



WPS Workbench ***user guide***

Version: 4.2.9

(c) 2023 World Programming

www.worldprogramming.com

Contents

Introduction.....	7
About WPS.....	7
WPS and SAS® software.....	8
Workbench components.....	8
Workspace Launcher.....	10
Migrating existing programs or projects.....	11
Importing an existing project.....	11
Migrating from previous versions of WPS.....	12
Migrating programs from SAS® software into the WPS SAS Language Environment.....	12
Code Analyser.....	13
Analysing program compatibility.....	14
Analysing language usage.....	15
Viewing or exporting an analysis report.....	16
Analysis restrictions.....	17
Online help.....	18
Cheat sheets.....	19
Environments.....	22
The SAS Language Environment.....	22
The Workflow Environment.....	22
Perspectives.....	24
Opening a perspective.....	24
Closing a perspective.....	25
Resetting a perspective.....	25
Saving a perspective.....	25
Deleting a perspective.....	26
WPS Processing Engines.....	27
Configuring a local WPS installation.....	28
Connecting to a remote Processing Engine.....	29
Creating a new remote host connection.....	30
Defining a new Processing Engine.....	32

Specifying startup options for a Processing Engine.....	34
Default Processing Engine.....	34
Local host connection properties.....	35
Processing Engine Properties.....	35
Processing Engine LOCALE and ENCODING settings.....	36
General text file encoding.....	37
Licence key.....	39
Applying a licence key.....	39
Database connectivity.....	39
Connect to Oracle.....	40
Connect to DB2.....	41
Connect to SQL Server.....	44
Connect to MySQL.....	44
Connect to a database using ODBC.....	46
Restarting Processing Engines.....	47
Restarting a WPS Server.....	48
Restarting Workflow Engines.....	48
Workbench Views.....	49
Working with views.....	49
View stacks.....	49
Opening a view.....	51
Detaching and reattaching a view.....	52
Generic views.....	52
Project Explorer.....	52
File Explorer.....	70
Properties.....	73
Link Explorer and Workflow Link Explorer.....	73
Log.....	75
SAS Language Environment views.....	78
Editor.....	78
WPS Server Explorer.....	78
Bookmarks.....	81
Tasks.....	82
Outline.....	83
Output Explorer.....	84
Results Explorer.....	85
Console.....	85
Search.....	86
Using Hub Authentication Domains and Library Definitions with the SAS Language Environment.....	86
Workflow Environment views.....	88
Editor.....	89
Database Explorer view.....	93
Data Profiler view.....	96
Dataset File Viewer.....	105

Bookmarks view and Tasks view.....	110
Using Hub Library Definitions with the Workflow Environment.....	111

Working in the SAS Language Environment..... 113

Creating a new program.....	113
Creating a new program file in a project.....	115
Entering WPS code via templates.....	116
WPS Code Injection.....	116
Using Program Content Assist.....	117
WPS syntax colouring.....	117
Running a program.....	117
Running a program in Workbench.....	118
Command line mode.....	121
Libraries and datasets.....	122
Setting the WORK location.....	123
Catalogs.....	124
Datasets.....	124
Working with program output.....	145
Dataset generation.....	146
Managing ODS output.....	146
Text-based editing features.....	152
Working with editors.....	153
Jumping to a particular project location.....	155
Navigation between multiple project files.....	156
Searching and replacing strings.....	157
Undoing and redoing your edits.....	157

Working in the Workflow Environment..... 159

Creating a new Workflow.....	159
Adding blocks to a Workflow.....	160
Connecting blocks in a Workflow.....	161
Removing blocks from a Workflow.....	162
Deleting Workflow blocks.....	162
Cutting, Copying and pasting blocks.....	163
Duplicating blocks.....	164
Undoing Workflow actions.....	164
Redoing Workflow actions.....	164
Deleting a working dataset.....	164
Workflow execution.....	165
Database References.....	165
Adding a DB2 database reference.....	166
Adding a MySQL database reference.....	167
Adding an Oracle database reference.....	168
Adding an ODBC database reference.....	168
Adding a PostgreSQL database reference.....	169
Adding an SQL Server database reference.....	170

Adding a Snowflake database reference.....	171
Adding an AWS Redshift database reference.....	171
Adding an MS Azure SQL Data Warehouse database reference.....	172
Parameters.....	173
Adding a parameter from its place of use.....	174
Adding a parameter from the Workflow Settings tab.....	174
Viewing parameters.....	175
Editing a parameter.....	175
Deleting a parameter.....	176
Binding a parameter to an option.....	176
Editing a parameter binding.....	177
Removing a parameter binding.....	177
Workflow block reference.....	177
Blocks.....	178
Block Workflow palette.....	180
Workbench Preferences.....	347
Using Workbench Preferences.....	347
General preferences.....	347
Changing shortcut key preferences.....	348
Backing up Workbench preferences.....	348
Importing Workbench preferences.....	349
Workflow Environment preferences.....	349
Data panel.....	350
Data Profiler panel.....	350
Workflow panel.....	351
Using Git with Workbench.....	359
Installing Workbench Git support.....	359
Using Workbench with Git.....	360
Opening the Workbench Git Staging view.....	360
Opening the Workbench Git perspective.....	360
Configuration files.....	361
AutoExec file.....	364
WPS Analytics tips and tricks.....	365
WPS Analytics troubleshooting.....	367
Making technical support requests.....	368



Legal Notices..... 369

Introduction

This guide will help you gain familiarity with *WPS Workbench*, the graphical user interface of WPS (the World Programming System). WPS Workbench has two environments: The *SAS Language Environment* (SLE), for developing traditional text based SAS language programs, and the *Workflow Environment* (WFE), a drag-and-drop graphical development tool. Each environment comes with its own *perspective*, that tailors Workbench's features and views to each environment.

The SAS Language Environment enables you to create, edit and run SAS language programs, along with their resulting datasets, logs and other output.

The Workflow Environment is a graphical development environment with features for data mining, predictive modelling tasks, and a range of Machine Learning capabilities.

For an overview of some commonly-used features, see *WPS Analytics tips and tricks* [↗](#) (page 365). For information to help you resolve some previously-reported problems see *WPS Analytics troubleshooting* [↗](#) (page 367).

About WPS

A description of the main two components of the World Programming System (WPS): an Integrated Development Environment (IDE) and a compiler/interpreter.

The World Programming System (WPS) consists of the following components:

- An Integrated Development Environment (IDE) – *WPS Workbench*. This environment utilises the *Eclipse IDE* and provides facilities to create and manage SAS language programs, and then to run these programs using the WPS server.
- A compiler/interpreter – the WPS Processing Engine, known as *WPS Server* when in the SAS Language Environment and the *Engine* when in the Workflow Environment. When using Workbench, the compiler is run as a server process and is used to process and execute programs.

WPS server or WPS Engine process

To run SAS language programs, Workbench requires a connection to a licensed Processing Engine (for more information, see *WPS Processing Engines* [↗](#) (page 27)). This process may be running either on the local workstation (a *local server* or *local engine*), or on an installation of WPS on a remote machine (a *remote server* or *remote engine*).

WPS and SAS[®] software

An overview of the relationship between WPS and SAS[®] software.

If you are accustomed to using other products related to the SAS language, you will find that the language support in WPS is familiar. You can expect to find much of the same syntax in terms of procedures, formats, macros, DATA steps, and so on.

WPS provides other recognisable features and objects such as logs, datasets, or the `WORK` library. Other features may be new to you, such as the Workbench environment itself and the way in which it handles or displays objects. You will find that the Workbench has help and reference material to assist you in migrating to WPS.

Compatibility with SAS[®] software

Besides being able to run, modify, share and save programs written in the SAS language, WPS is also able to read and write data files used by SAS software, for example `SAS7BDAT` files. WPS also includes a wide selection of library engines to allow you to access many leading third party databases, data warehouses and Hadoop big data environments.

WPS uses a proprietary dataset file format known as `WPD`. Temporary datasets written to the `WORK` library use this `WPD` format.

Supported language elements

WPS does not yet support every element in the SAS language. Workbench provides a code analysis tool (see [Code Analyser](#) (page 13)) to help determine if any of your existing SAS programs contain unknown language elements. Details of the SAS language elements currently-supported in WPS can be found in the *WPS Reference for Language Elements*.

Existing SAS programs

WPS uses the terms *SAS language program*, or *program* to describe scripts, programs and applications written in the SAS language.

Workbench components

Brief descriptions of the types of objects managed by the Workbench.

Projects

The fundamental unit of organisation for SAS language programs and related objects. For example, you might have a project for applications under development, or another for monthly reporting jobs. For more information, see [Projects](#) (page 53).

SAS language programs

Create or modify SAS language programs using the SAS language editor. Files containing SAS language programs use either `.wps` or `.sas` extensions. For more information see [SAS language editor](#) (page 153).

Workflow

The Workflow Environment is a drag-and-drop graphical development environment with features for data mining, predictive modelling tasks, and a range of Machine Learning capabilities.

Log output

When you run a SAS language program, the information generated is stored in a log file. This file can be viewed, printed and saved from within the Workbench. The log generated is cumulative and contains the results of each program run since opening Workbench, restarting the Processing Engine, or clearing the log. For more information, see [Log](#) (page 75).

Listing output

This contains the printed output from any programs that you have run. For example, this could be a table of data generated by a `PROC PRINT` statement. Listing output can be viewed, printed and saved in the Workbench. The listing output generated is cumulative and contains the output of each program run since opening Workbench, restarting the Processing Engine, or clearing the listing output. For more information, see [Listing output](#) (page 147).

ODS output

As described, the ODS (Output Delivery System) can be used to produce text listing and HTML output. Each program can specify when and where its output is stored but it is also possible to allow the Workbench to manage the process automatically. The default option is for the Workbench to generate HTML output. For more information see [Managing ODS output](#) (page 146).

Datasets

The data generated from running a program is stored in datasets. You can browse or edit a dataset using the dataset viewer. For more information, see [Datasets](#) (page 124).

Host Connection

A computer, local or remote that Workbench can access. The local host connection enables you to run SAS language programs and Workflows on the Processing Engine installed on your local machine. Connections can also be made to remote WPS servers, see [Connecting to a remote Processing Engine](#) (page 29) for more information.

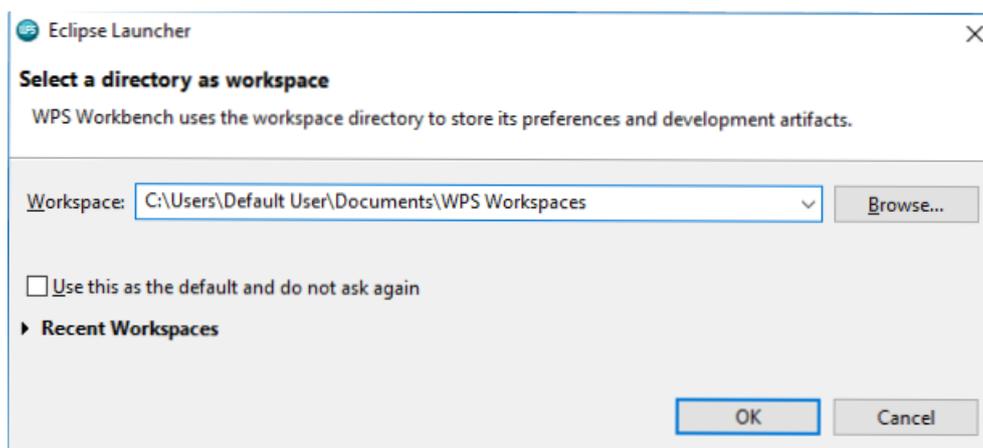
WPS Processing Engine

Executes SAS language programs and Workflows and generates the resulting output, such as the log, listing output, results files, and datasets. For more information, see [WPS Processing Engines](#) (page 27).

Workspace Launcher

A description of the **Eclipse Launcher** that is displayed when Workbench is opened, allowing you to choose which *workspace* to use.

When you open Workbench, the **Eclipse Launcher** dialog box is displayed.



A *workspace* is the parent folder used to hold one or more projects (see [projects](#) (page 53)). It is possible to use any folder on any drive that your computer can access as a workspace, and the workspace you use is the default location for newly-created projects.

You do not have to use projects to manage your Workbench resources; any files you can access (on your local system or a remote computer) can be accessed in Workbench through the File Explorer (see [File explorer](#) (page 70)).

On Windows, the default location of the workspaces is your *My Documents* folder, and, on UNIX and Linux platforms, it is your *home* directory. In all cases, by default the sub-directory is called *WPS Workspaces* and the workspaces are by default named incrementally, starting at *Workspace1*.

To use a different workspace either select the name in the **Workspace** list, select a workspace from the **Recent Workspaces** list, or click **Browse** and navigate to the workspace folder in the **Select Workspace Directory** dialog box.

The **Eclipse Launcher** dialog box is displayed whenever you start Workbench. To hide the dialog box at start up and use the selected workspace every time you start Workbench, select the **Use this as the default and do not ask again** check box.

Migrating existing programs or projects

Details of how to use existing projects or SAS language programs with the latest version of WPS.

Importing an existing project

Importing an existing project into Workbench.

To import an existing project into the Workbench:

1. Click the **File** menu, click **Import** to open the **Import** wizard.
2. On the **Select** page, expand the tree under the **General** node, select **Existing Projects into Workspace** and click **Next**
3. Select the import method:
 - To import an existing project, click **Select root directory** and either select the project from the list or click **Browse** to navigate to the folder where the project is located.
 - To import an archived project, for example a project stored as a Zip `.zip` file, click **Select archive file**. Select the archive file from the list or click **Browse** to navigate to the folder where the archive file is located.

If the archive file or folder contains a valid project, the project name is added to the **Projects** list.

4. Click **Finish** to import the selected project.

Use this method to import the samples project supplied with Workbench. Sample projects are available in each supported language, and the project archive file `samples.zip` is located in the `doc/<language>` folder in your Workbench installation.

Migrating from previous versions of WPS

Workspaces from previous versions of WPS can be opened using the latest version, possibly requiring an automated migration step to be confirmed when doing so.

Workspaces opened with previous versions of WPS can be opened with the current version, so there are no migration steps to be performed to use a workspace from a previous version of WPS with the latest version. However, the first time that you open a workspace that was created using an earlier version of WPS, an Eclipse dialog is displayed asking for your confirmation that it is **OK** for the workspace to be upgraded automatically. Such automatic upgrades should not cause any problems.

Existing  programs do not need to be modified to work with the latest version of WPS. In addition, projects that were created with earlier versions of WPS can be opened and used by later versions of WPS without any additional action.

Migrating programs from SAS[®] software into the WPS SAS Language Environment

Using existing programs written in the SAS language with the Workbench

If you already have programs written in the SAS language, there is no conversion process to undertake in order to use these programs with WPS Analytics. Any file with the `.wps` or `.sas` extension is assumed to be a program that the Workbench can open, edit and run.

Accessing your existing programs

You can access your existing programs from the **File Explorer** view.

Alternatively, you can use Workbench projects, to manage your files enabling you to use other features such as local history.

Checking program compatibility

Opening programs in the Workbench causes unknown or unsupported language elements to be displayed in red. However, before trying to execute your existing programs with WPS Analytics, it is also recommended that you use the Workbench *Code Analyser* [↗](#) (page 13) for further verification.

Analysing programs, even hundreds of programs at a time, can take less than a few minutes to complete and can therefore be quicker than trying to execute long-running programs. Program analysis is available from both the **File Explorer** view or the **Project Explorer** view.

Code Analyser

Use the Workbench SAS Language Environment's *Code Analyser* tool to scan the code of existing programs written in the SAS language.

The Code Analyser is a feature of the Workbench and can therefore only be used on platforms on which Workbench is supported. The Code Analyser generates reports that indicate which of your existing programs will run unchanged in WPS Analytics, which programs may require modification to run, and which programs contain language elements not yet supported by WPS.

The Code Analyser enables you to:

- **Analyse Program Compatibility** – determines whether your existing programs or projects contain any unsupported language elements.
- **Analyse Language Usage** – lists language elements used in the selected programs or projects. The analysis indicates which elements are supported and which are not.

Analysing mainframe programs

The *Code Analyser* is a feature of the Workbench and can therefore only be used on platforms on which Workbench is supported. To analyse programs designed to run on a mainframe, copy the required programs from the mainframe to a workstation running the Workbench. When these programs are analysed, the *Code Analyser* will identify SAS language elements specific to the z/OS platform.

Before analysing programs copied from a mainframe to a workstation:

- We recommended you remove sequence numbers from your jobs.
- Ensure each program file has a file extension of `.sas`
- Package up the jobs on the mainframe using *XMIT*.
- Once transferred to the workstation, use *XMIT Manager* to un-XMIT the jobs for analysis.

Note:

XMIT Manager can only handle PDSs (Partitioned Datasets) and not PDSEs (Extended Partitioned Datasets).

When downloading the *XMIT* files from mainframe to your workstation, you must specify FB (Fixed Block) with an LRECL (Logical Record Length) of 80 and with no ASCII/EBCDIC conversion, truncation or CRLF translation.

Running analyses

The Code Analyser can quickly analyse single programs, all programs in one or more projects, or all programs in one or more folders.

Programs that might normally be executed on multiple different platforms can be analysed together. Gather the programs from these other environments, copy them to your workstation and then run the analysis tools from Workbench.

Analysing program compatibility

You can analyse one or more programs to identify language elements that are not yet supported by WPS.

To analyse SAS language programs:

1. Select the files to be analysed:
 - Select the required file or files to analyse in the **Project Explorer** or **File Explorer**. You can highlight programs in different projects or folders within the particular view.
 - Select one or more projects in the **Project Explorer** to analyse all contained programs.
 - Select one or more folders in the **File Explorer** to analyse all contained programs.
2. In the view corresponding to your selection, right-click on the selected items and from the short cut menu click **Analyse**, and then click **Program Compatibility**.

When the analysis has finished, a **Program Compatibility Report** automatically opens in the Workbench.

Compatibility Analysis Results

[Export results to Excel](#)
[File details](#)

Please note that due to the nature of the language of SAS it is not possible to guarantee these static analysis results are 100% accurate

A total of 197 programs analyzed
A total of 184 analysed programs are without problems (93.4%)
A total of 53 files were ignored

A total of 13 analysed programs had problems (6.6%)
A total of 11 distinct problems were identified
A total of 62 individual problems were identified

▼ **Language Element Individual Problems Table**

Element	Problems
Procedure	47
Procedure option	15

This report contains details about unsupported language elements used in programs. It does not report on any supported language elements that were used in the programs.

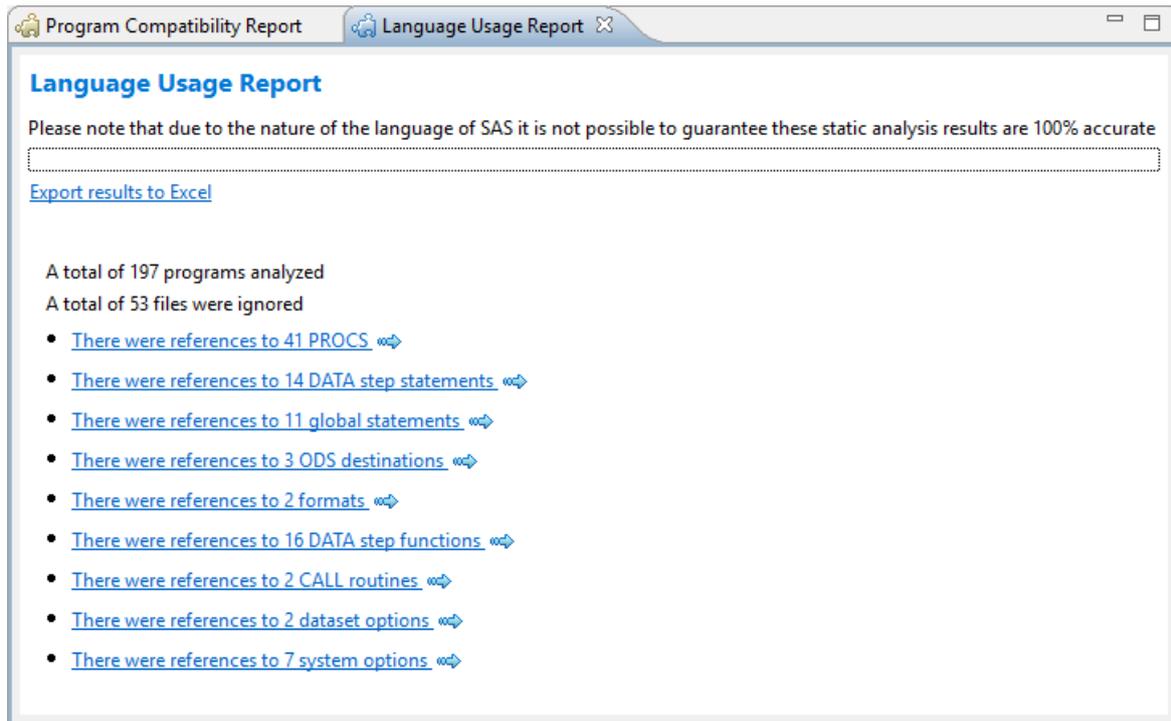
Analysing language usage

You can analyse one or more programs to identify supported language elements used in one or more programs.

To analyse SAS language usage in programs:

1. Select the files to be analysed:
 - Select the required file or files to analyse in the **Project Explorer** or **File Explorer**. You can highlight programs in different projects or folders within the particular view.
 - Select one or more projects in the **Project Explorer** to analyse all contained programs.
 - Select one or more folders in the **File Explorer** to analyse all contained programs.
2. In the view corresponding to your selection, right-click on the selected items and from the short cut menu click **Analyse**, and then click **Language Usage**.

When the analysis has finished, a **Language Usage Report** automatically opens in the Workbench.



This report details the SAS language elements used in the selected programs, and you can explore the detail of this report in the Workbench, or export the content to Microsoft Excel

Viewing or exporting an analysis report

You can view the detail of a *Code Analyser* report in Workbench, or export the report detail to Microsoft Excel.

When viewing a report in Workbench, you can navigate to more detail, and ultimately to the source program where an issue is located.

- The program compatibility report summary links to a list of programs analysed. From this list you can access a list of problem elements, see where those elements are used within a program, and navigate to the element location within the file.
- The language usage report summary displays the SAS language elements used in the analysed programs. From this list you can find the frequency of SAS language element usage, which programs contain the elements, and navigate to the element location within the file.

Exporting analysis results to Microsoft Excel

The results of a program compatibility report or language usage report can be exported to a Microsoft Excel workbook for either further analysis or to preserve the report information. To export a report:

1. In the report summary page, click **Export results to Excel**.

2. In the **Save as** window, enter the required name for the workbook, and browse to the required location before saving it.

Analysis restrictions

Limitations of the Code Analyser.

Because of the nature of the SAS language, the result of the analysis cannot be guaranteed, and reports should be treated as a guide for further analysis.

The Code Analyser has some limitations:

- The Code Analyser will not report incorrect syntax.
- SAS language elements that are not yet supported in WPS are now shown in the Compatibility Report as unknown.
- The analysis reports do not currently contain information about the use of macro language elements and library engine (data access) elements. Their use is however supported in WPS.

Online help

The context-sensitive **Help** view automatically displays relevant help items as you select different Workbench views.

This view is not open by default. To add this view to the current perspective click the **Help** menu and then click **Show Contextual Help** or press **F1**.

Help view

The **Help** view can also display the content of the WPS Analytics documentation; to do this click the **Help** menu and then click **Help Contents**. The **Help** view provides features to help you navigate through the documentation:

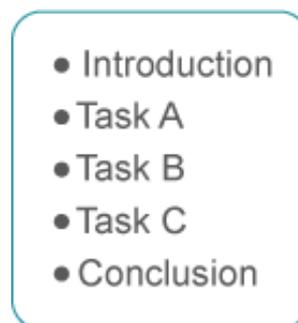
- **Show in Contents** – (, at top right of help window) synchronises the table of contents with the help topic you are reading.
- **Bookmark Document** – (, at top right of help window) adds a shortcut to a specific page in the documentation.
- **Search** – (top left of help window) search the help for specific keywords and phrases.

Cheat sheets

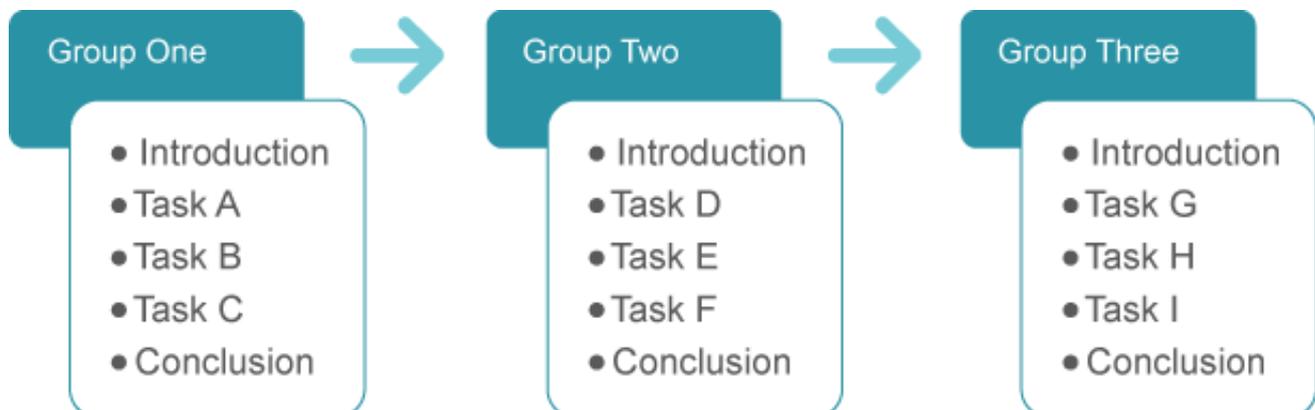
The Workbench contains tutorials known as *cheat sheets* designed to help you start using WPS by introducing specific tasks and features.

Cheat sheets can be one of two different types: 📄 *simple* or 📖 *composite*.

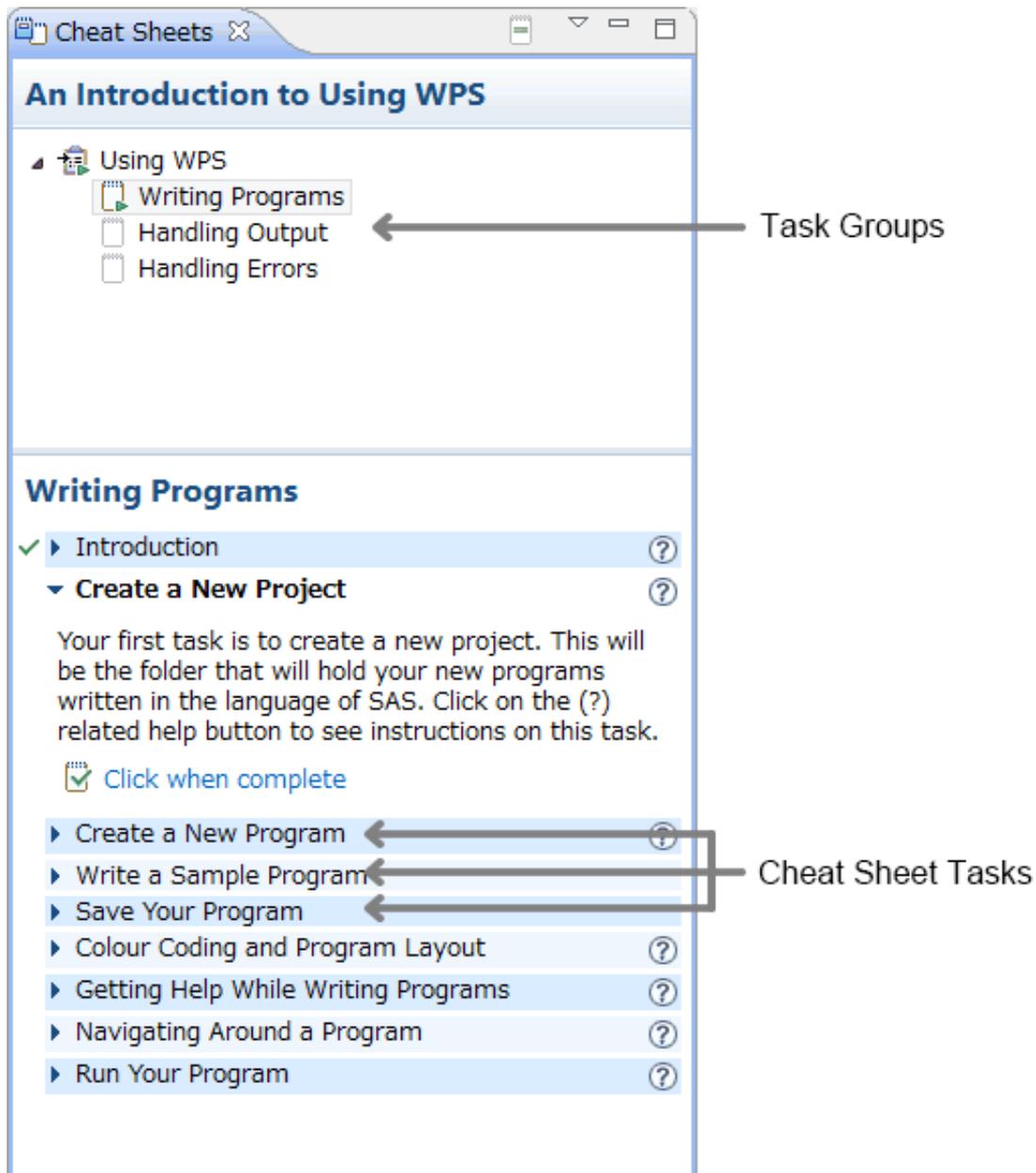
A simple cheat sheet, designed to guide you through a single task, has an introduction to set the scene, followed by a list of tasks designed to be performed one after another:



A composite cheat sheet is designed to guide you through a more complex problem. The problem is broken down into smaller manageable groups, each group consisting of an introduction and conclusion:



Composite cheat sheets have two panes in the user interface. One shows the groups of tasks, the other shows the tasks in each group.



Open a cheat sheet

Cheat sheets are opened from the help menu in the Workbench.

1. Click the **Help** menu and then select **Cheat Sheets**.
2. If necessary, expand the **WPS Workbench** grouping.
3. Select the cheat sheet from the list, and click **OK** to open.

Close a cheat sheet

You can close the active cheat sheet by selecting the **Close** on the cheat sheet's tab. The active cheat sheet saves its completion status when it is closed so that you can continue where you left off when you next reopen it.

Working through a cheat sheet

When you open a new cheat sheet, the introduction is expanded so that you can read a brief description about the selected cheat sheet.

In a simple cheat sheet, click **Click to Begin** at the bottom of the introductory step. The next step is then expanded and highlighted. You should also see action buttons at the bottom of the next step, for example **Click to Complete**.

In a composite cheat sheet, read the introduction and then click the **Go to...** link at the bottom of the first sheet. The introduction for the first group is displayed, click the **Start working...** link to begin. You progress through a group in the same manner as a simple cheat sheet.

In simple cheat sheets or a composite group, when you have finished a particular step, click **Click when complete** to move to the next step. A check mark  appears in the left margin of each completed step.

Composite cheat sheets have a conclusion, and also the option to review the task, or to progress onto the next group of tasks. Starting the next task will take you to the introduction for the next group.

You can open any step out in a cheat sheet by clicking the title of the section. If you are working through each step in the sheet, click  **Collapse All Items but Current** to collapse all opened steps except the current active step waiting to be completed.

In composite cheat sheets, you can review any previously-completed group by clicking the group in the task group pane. The displayed point in the group is where you left that group, for example if you completed the group tasks, the conclusion is displayed.

A simple cheat sheet is completed when you finish the last step. A composite cheat sheets is complete when all task groups have been finished.

To restart from the first step, open the first step and click **Click to Restart**. A Composite cheat sheet allows you to reset task groups. Right-click on the task group in the groups pane and click **Reset**.

Environments

Workbench has two environments: The *SAS Language Environment* (SLE), for developing traditional text based SAS language programs, and the *Workflow Environment* (WFE), a higher level graphical development tool.

The SAS Language Environment

The SAS Language Environment enables you to create, edit and run SAS language programs, and view the resulting datasets, logs and other output.

Opening the SAS Language Environment Perspective

To open the SAS Language Environment perspective, click on the **SAS Language Environment** button at the top right of the screen: . This button can be made clearer by right-clicking on it and selecting **Show Text**; this displays the name of the button on the button itself:  **SAS Language Environment**.

The Workflow Environment

The Workflow Environment is a drag-and-drop graphical development environment with features for data mining, predictive modelling tasks, and a range of Machine Learning capabilities.

Opening the Workflow Environment Perspective

To open the Workflow Environment perspective, click on the **Workflow Environment** button at the top right of the screen: . This button can be made clearer by right-clicking on it and selecting **Show Text**; this displays the name of the button on the button itself:  **Workflow Environment**.

Overview

The Workflow Environment operates in a completely different way to how Workbench operates when you develop SAS language programs using the SLE.

The **Workflow Editor** view provides a palette of drag-and-drop interactive blocks that you combine to create Workflows that connect to a data source, filter and manipulate the data into a smaller subset using **Data Preparation** blocks. You can save the data to an external dataset for later use. You can use the Machine Learning capabilities available in the **Model Training** blocks to discover predictive relationships in your filtered and manipulated data.

Once created, a Workflow is re-usable, so can be used with different input datasets to generate output datasets filtered or manipulated in the same way.

A Workflow can auto-generate error-free code from the models, ready for deployment and execution in production.

The **Data Profiler** view is a graphical tool that enables you to explore WPS Analytics datasets used in a Workflow, or external to a Workflow. You can use the **Data Profiler** view to interact with and explore your data through graphical views and predictive insights.

Many of the Data Science features available in the **Workflow Editor** view are enabled by World Programming's built-in SAS language capabilities. Although coding is not a prerequisite for creating a Workflow, those in the team who have the requisite programming skills can carry out more advanced tasks in a Workflow, using not only the SAS language code block, but also R, Python and SQL.

Who should use the Workflow Environment

If you are familiar with the Cross-Industry Standard Process for Data Mining (CRISP-DM), you can use Workflows to follow this process in the preparation and modelling of data sources of all sizes. The Workflow Environment tools enable a team of people with different skill sets to work on the same data in a collaborative environment. For example, a single Workflow could be created that enables:

- Data analysts to blend the prepared data through joining, transformation, partitioning, and so on, to create raw datasets of varying sizes.
- Data scientists to use machine learning algorithms to build, explore and validate reproducible predictive models, including scorecards, from proven datasets.

Perspectives

The positions of available views, windows, view stacks, menu items, and toolbars, together with any other items that make up the general layout of Workbench, together constitute a  *perspective*.

There are two default perspectives provided with Workbench, one for each environment:

- **Workflow Environment perspective:** A collection of views suited to producing Workflows. Some views are applicable to just the Workflow Environment, some to the SAS Language Environment and some to both.
- **SAS Language Environment perspective:** A collection of views suited to producing SAS language code. Some views are applicable to just the Workflow Environment, some to the SAS Language Environment and some to both.

The two default perspectives can each be customised by moving, resizing, adding or removing different views. You can save customised perspectives and to switch between different Workbench perspectives.

When you open Workbench, the windows views, toolbars and the so on, are displayed in the same arrangement as when closed. This ensures that your preferred perspective is visible when you open Workbench.

Opening a perspective

Opening a perspective in Workbench.

To open a different perspective, click the **Window** menu click **Open Perspective** and then click **Other**:

- To open one of the default perspectives, click either **Workflow Environment perspective** or **SAS Language Environment perspective**. These default perspectives can also be opened using the quick access icons at the top right of the Workbench window.
- To open another perspective, click **Other**, then select the required perspective from the list and click **OK**.

The selected perspective is displayed in Workbench. If you have other perspectives available, they are listed in top right of the Workbench, so that you can click switch between them quickly.

Closing a perspective

Closing a perspective in Workbench.

To close a perspective:

1. If you have more than one perspective open, ensure that the Workbench is displaying the one that you want to close.

Note:

To switch perspective, look in the area to the right of the **Open Perspective**  icon, and then click on the perspective required.

2. Select **Window > Perspective > Close Perspective**.

Note:

To close all the perspectives, so that only the Default WPS Perspective remains available, select **Window > Perspective > Close All Perspectives** Your interface will go blank.

Note:

To restore the default perspective, select **Window > Perspective > Open Perspective > SAS Language Environment**.

Resetting a perspective

Resetting a perspective in Workbench back to its default layout.

To reset the current perspective back to its default layout:

1. Select the **Window** menu, click **Perspective**, then **Reset Perspective**.
2. You are prompted for confirmation to reset the perspective. Click **Yes** to confirm that you wish to do so.

Saving a perspective

Saving a perspective in Workbench.

If you have changed the layout of a perspective, you can save it under a user defined name for future use, as follows:

1. Select the **Window** menu, then click **Save Perspective As....**
2. Enter a name for the perspective at the top of the **Save Perspective As...** dialog.

3. Click **OK** to continue.

The new perspective appears in the top right of the Workbench, to the right of the **Open Perspective**  icon.

Deleting a perspective

Deleting a perspective in Workbench.

To delete a perspective:

1. Select the **Window** menu, then **Preferences**.
2. From the left hand tree view of the **Preferences** window, select General, then **Perspectives**.
3. From the list of **Available perspectives**, click on the perspective that you want to delete.
4. Click **Delete** to remove the perspective permanently.

Note:

You can only delete perspectives that you have created. You cannot delete a perspective that was supplied with WPS, including the SAS Language Environment perspective.

5. Click **OK** to close the **Preferences window**.

WPS Processing Engines

Workbench uses one or more Processing Engines to run SAS language programs and Workflows. In the SAS Language Environment, Processing Engines are referred to as *WPS Servers*; whereas in the Workflow Environment Processing Engines are referred to as *Engines*. Processing Engines can exist locally, on the same host as the Workbench installation, or remotely, on another host.

Viewing servers and connections

The list of defined connections and Processing Engines is stored in a workspace, and is visible through the **Link Explorer** (when in the SAS Language Environment perspective) or the **Workflow Link Explorer** (when in the Workflow Environment perspective).

Location of Processing Engines

A Processing Engine can either be on the local workstation or on a remote machine with an installation of WPS.

- A Processing Engine on a local machine (a *local server* or *local engine*) can be accessed through a *local host connection*.
- A Processing Engine on a remote host (a *remote server* or *remote engine*) can be accessed through a *remote host connection*.

Setting up remote servers can give you access to the processing power of remote server machines from the Workbench on your workstation. Multiple servers can run under a single remote host connection. For more information, see [Connecting to a remote Processing Engine](#) (page 29).

When WPS Workbench is first installed, a single local host connection called **Local** is created. This local connection hosts the Processing Engines: a **Local Server** for the SAS Language Environment and a **Local Engine** for the Workflow Environment. This connection is started by default when the Workbench is started, and terminates when the Workbench is closed.

WPS server data

As well as running SAS language programs, WPS servers in the SAS Language Environment contain data associated with programs they have run, such as the log, output results, file references, library references and datasets. This does not apply to WPS engines, which exist purely to run Workflows.

Creation of new Processing Engines

In the SAS Language Environment, further local servers can be created if required, or the local server can be deleted or uninstalled. Creating multiple local servers requires no further licensed WPS products and the only restriction on the number of local servers is the local machine resources.

In the Workflow Environment, only one engine per host can be manually defined. Workbench will automatically create and manage multiple engines as and when required to run concurrent Workflow branches. The number of engines currently running is shown by a number in brackets after the engine name in Workflow Link Explorer. All engines on a host will use the original engine's properties as a template.

Changing workspaces

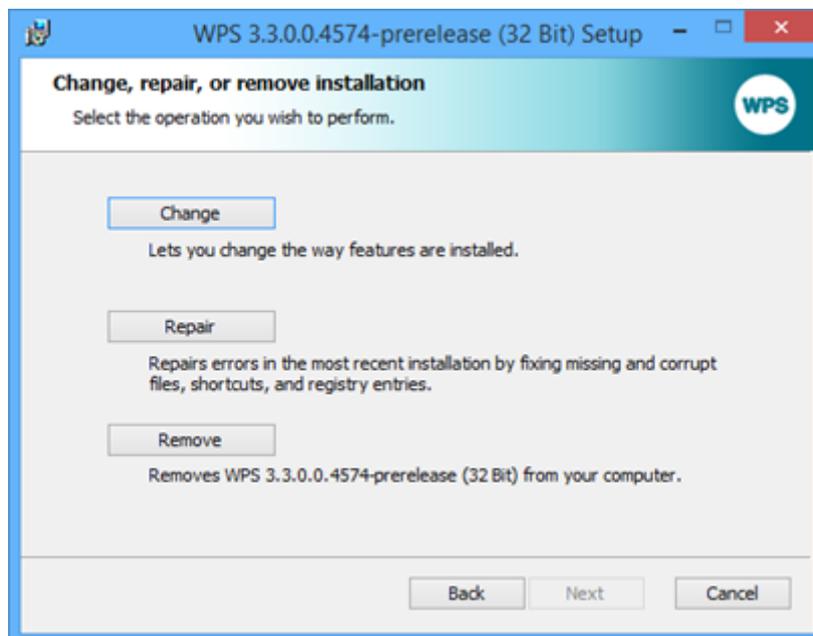
If you change workspace, you may have different connections and Processing Engines. To share server definitions in a work group, or between workspaces, export (see [Exporting a WPS server definition](#) (page 33)), and import (see [Importing a WPS server definition](#) (page 33)) Processing Engine definitions to or from a file.

Configuring a local WPS installation

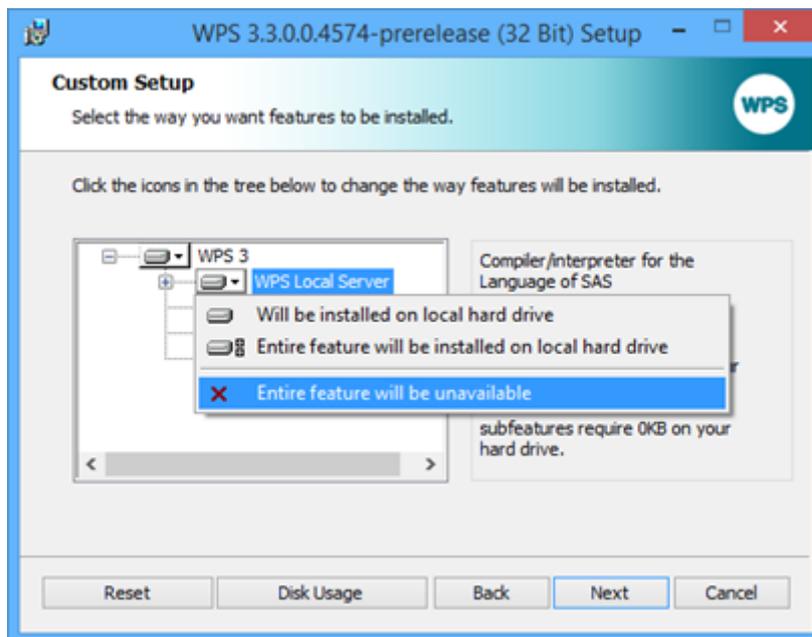
If you only run SAS language programs and Workflows on a remote Processing Engine, you can modify your Workbench installation to remove the local Processing Engine. This option is only available for Workbench installations on Microsoft Windows.

To remove the local Processing Engine:

1. Open the **Programs and Features** control panel item and select **WPS** in list of installed programs.
2. Click **Change** at the top of the **Programs and Features** window. In the **Change, repair, or remove installation**, then Next at the first page of the Wizard. At the next screen, click **Change**:



3. In the **Custom Setup** window, clear **WPS Local Server** and complete the amended installation.



On a non-Windows platform, the WPS server cannot be removed from the Workbench installation. To remove the local server access from Workbench, in the **Link Explorer** view, select the **Local Server** and click **Delete**.

Connecting to a remote Processing Engine

How to create a link from Workbench to a WPS server on a remote machine.

The WPS server must exist before a connection can be made from Workbench. The remote server requires a licensed copy of the WPS server. If the remote server is Windows, it requires an SSH server; this is not necessary for Linux due to native SSH support.

Client/server installation summary

Before connecting to a remote server, you require the hostname for the remote server, the user name and password log on credentials, and the full WPS server installation path on the remote server.

To create a connection to a remote server:

1. Create a connection to the remote host. See [Creating a new remote host connection](#) (page 30).
2. Add a WPS server to this remote host. See [Defining a new Processing Engine](#) (page 32).

Creating a new remote host connection

A remote host connection enables you to access a remote host's file system using the **File Explorer** view and to link to the Processing Engine installed on the remote host to run Workflows or SAS language programs. New remote hosts will display in both **Link Explorer** view and **Workflow Link Explorer**, but links to WPS Servers can only be viewed in the SAS Language Environment perspective, and Workflow Engines in the Workflow Environment.

Before creating a new connection, you will need:

- SSH access to the server machine that has a licensed WPS server installation.
- The installation directory path for the WPS server installation.

You may need to contact the administrator to obtain this information, and to ensure that you have access. For more details about SSH authentication, see the *WPS Link user guide and reference*.

To create a new remote host connection:

1. Click the **WPS** menu, click **Link** and then click **New Remote Host Connection**. The **New Server Connection** dialog is displayed.
2. Select **New SSH Connection (3.2 and later -- UNIX, MacOS, Windows)** and click **Next**.
3. Complete the **New Remote Host Connection (3.2 and later)** dialog box as follows:
 - a. In **Hostname**, enter the domain name or IP address of the remote server machine.
 - b. Unless modified by your system administrator, the default **Port** value (22) should be left unchanged.
 - c. In **Connection name** enter a unique name to be displayed in Workbench for this connection. By default this entry will be copied from your entry in the **Hostname** box.
 - d. In **User name**, enter your user ID on the remote host.

Click the required check box:

Option	Description
Enable compression	Controls whether or not data sent between the Workbench and the remote connection is compressed.
Verify Hostname	Confirms whether the host you have specified in the Hostname entry exists.
Open the connection now	Whether the connection is automatically opened immediately
Open the connection automatically on Workbench startup	Controls whether the connection is automatically opened when the Workbench is started.

4. Click **Next** to define the connection directory shortcuts (if required):
 - a. Click **Add** to display the **Directory Shortcut** dialog box.
 - b. In **Directory Name** enter the displayed shortcut name, and in **Directory Path** the full path of the target directory. Click **OK** to save the changes.
 - c. Enter the displayed shortcut name in **Directory Name**, and the full path in **Directory Path** and click **OK**.
5. Click **Finish** to save the changes.

If you have not previously validated the authenticity of the remote host, an **SSH2 Message** dialog box is displayed requesting confirmation of the new connection.

6. In the **Password Required** dialog enter your **password** for the remote machine and click **OK**.

The **Link Explorer** view contains an entry for the new connection. To run WPS Workflows or SAS language programs, you will now need to define a new processing engine (see *Defining a new Processing Engine* [↗](#) (page 32)).

Advanced connection options

Controlling the environment variables available to a remote Processing Engine.

By default, the Processing Engine process inherits the interactive shell environment for the user under whose ID the server is being started. The default environment may not include everything required by the Processing Engine, for example database client shared libraries may not be defined for the server.

On Unix/Linux systems, this might require that any special environment variables for the Processing Engine are added to the `LD_LIBRARY_PATH`; or the `/etc/profile` or `~/.profile` modified so that database client shared libraries can be loaded by the Processing Engine.

An alternative on Unix/Linux is to place a `wpsenv.sh` file in the WPS installation directory, or in your home directory. This file is automatically run before the server is launched

On Windows, any special environment variables for the WPS server should be configured through the **Control Panel**, as either system or user environment variables.

Note:

All processes running as the user will see the created variables, and there is no way to specify that they should only be visible to the WPS process.

Exporting a host connection definition

A host connection definition can be exported from Workbench to enable sharing in a workgroup or to copy connection definitions between workspaces.

To export host connection definitions:

1. In the **Link Explorer**, select the connection for which definitions will be exported.
2. Right-click the connection and click **Export Host Connection** in the shortcut menu.
3. In the **Export to File** dialog box, navigate to the save location, enter a **File name** and click **Save**. The exported file `.cdx` contains the connection definition in XML format.

The host connection definition is saved to the selected file, which can be shared in the workgroup.

You can select multiple connections in the **Link Explorer** view and export them all to the same definition file to share more than one connection definition.

See *Importing a host connection definition* [↗](#) (page 32) for how to import the connection definitions into a workspace.

Importing a host connection definition

A host connection definition can be imported into Workbench to use consistent host definitions in a workgroup or between workspaces.

This task assumes that you have previously exported some connection definitions to a file, or have been provided with an export file created by someone else in your work group.

1. Click the **WPS** menu, click **Link**, and then click **Import Remote Host Connection**.
2. In the **Import Host Connection** dialog box, select the required connection definition file (`.cdx`) and click **Open**.

The host connection definition is imported from the selected file.

If there are any name clashes, the imported connection definition is automatically renamed so that it has a unique name within your list of connections.

Defining a new Processing Engine

In the SAS Language Environment, Workbench allows you to manually create multiple WPS servers on a local or remote host connection. In the Workflow Environment, only one Engine can be created for each host.

New Processing Engines can only be defined for an active host connection.

To define a new Processing Engine:

1. In Workflow Link Explorer or **Link Explorer** view, right-click on the host you wish to create an engine on, and select **New WPS Server** (SAS Language Environment), or **New Engine** (Workflow Environment).

2. In **New WPS Server** or **New Remote Engine** dialog box, enter a unique display name in **Server name** or **Engine Name** (or accept the default suggestion).

If you are defining a new remote WPS server, then in the **Base WPS install directory** box enter the path to the WPS base installation directory on the remote server.

3. Click **Finish** save the changes.

The **Link Explorer** view contains an entry for the new WPS server. If required, you can configure the startup options for the new server, see *WPS server properties* [↗](#) (page 79).

Exporting a WPS server definition

In the SAS Language Environment, a WPS server definition can be exported from Workbench to enable sharing in a workgroup or to copy connection definitions between workspaces.

To export a WPS Server definition:

1. In the **Link Explorer** view, select the server for which the definition will be exported.
2. Right-click the server and click **Export Host Connection** in the shortcut menu.
3. In the **Export to File** dialog box, navigate to the save location, enter a **File name** and click **Save**. The exported file `.sdx` contains the connection definition in XML format.

The WPS server definitions will be saved to the selected file, which can be shared in the workgroup.

You can select multiple servers in the **Link Explorer** view and export them all to the same definition file to share more than one connection definition.

See *Importing a WPS server definition* [↗](#) (page 33) for how to import the server definitions into a workspace .

Importing a WPS server definition

In the SAS Language Environment, a WPS server definition can be imported into Workbench to use consistent definitions in a workgroup or between workspaces.

A WPS server definition can only be imported into a host connection. If you need to create a host definition, see *Creating a new remote host connection* [↗](#) (page 30).

To import a WPS server definition

1. Click the **WPS** menu, click **Link**, and then click **Import Remote Host Connection**.
2. In the **Import from File** dialog box, select the required server definition file (`.sdx`) and click **Open**.

The WPS server definitions are imported from the selected file.

If there are any name clashes, the imported server definition is automatically renamed so that it has a unique name within your list of WPS servers.

Specifying startup options for a Processing Engine

You can specify the settings applied to a Processing Engine and use these startup options to change the behaviour of the engine.

The startup options for Processing Engines are WPS Analytics system options. You can control items such as system resource usage or language settings. Not all system options can be set as startup options. A list of system options that can be set when a Processing Engine starts, their effect, and their supported values, can be found in the *WPS Reference for Language Elements*.

All startup options are set in Workbench in the same way. For example, to set the value of the `ENCODING` startup option for a Processing Engine:

1. In the Workflow Link Explorer or **Link Explorer** view, right-click the required Processing Engine and click **Properties**.

A **Properties** dialog box is displayed.

2. Expand **Startup**, and then click **System Options**.
3. In the **System Options** panel, click **Add**.

The **Startup Option** dialog box is displayed.

4. For this example, enter `ENCODING` in the **Name** field.

If you do not know or are unsure of the name of a system option, you can click **Select**, which displays the **Select Startup Option** dialog box from which you can select a system option. Click **OK** in this dialog box to select the option.

5. Enter `UTF-8` in the **Value** field.
6. Click **OK** to save the changes and when prompted, click **Yes** to restart the Workflow Engine for the changes to take effect.

Default Processing Engine

Setting or changing the Processing Engine used by default when running SAS language programs or Workflows in Workbench.

One of the available Processing Engine connections defined in Workbench must be identified as the default WPS server to be used when a SAS language program is executed or a Workflow is run.

When Workbench is first installed, the **Local Server** or **Local Engine** is the default WPS server (depending on environment). You can select any available server to be the default Workbench server.

To set the default server:

1. Open the **Link Explorer** or **Workflow Link Explorer** (depending on perspective) and expand the connection containing the required WPS server.
2. Right-click on the required server and click **Set as Default Server** on the shortcut menu.

Local host connection properties

The local host is created during workbench installation, and the only properties available for a local host connection are directory shortcuts.

To access local host connection properties, in the **Link Explorer**, right-click on the **Local** host connection and click **Properties** in the shortcut menu.

A *directory shortcut* is a shortcut to a directory on the local file system. The local shortcuts that are created by default differ in accordance with the operating system on which you are running the Workbench:

- On Microsoft Windows operating systems, there will be *Home* and *Workspace* shortcuts, and also a shortcut for each available local drive. For example, if you have drives *C:* and *D:* on your computer, there will be four shortcuts by default.
- On all other operating systems, only the *Home (~)* and *Root (/)* shortcuts are created by default.

Create a new shortcut

To create a new directory shortcut:

1. In the **Link Explorer** view, right-click on the **Local** host connection and click **Properties** in the shortcut menu.
2. Click **Add** to display the **Directory Shortcut** dialog box.
3. In **Directory Name**, enter the displayed shortcut name, and in **Directory Path** the full path of the target directory. Click **OK** to save the changes.

Processing Engine Properties

Viewing information about a Processing Engine.

To display Processing Engine properties, right-click on the Processing Engine and then click **Properties**. Processing Engine properties are displayed as follows:

- **Code Submission:** An option to change the working directory to the program directory when code is submitted to the Processing Engine.
- **Engine Instances:** Controls how many engines are created to run Workflows.
- **Environment:** Details of the Processing Engine's environment (working directory, process ID and environment variables).
- **Macro Variables:** The full list of automatic and global macro variables used by the server.
- **Startup:** Flag to indicate whether or not the server is to be started automatically on connection startup, the initial current directory for the server process (for remote servers), startup environment variables (for local servers), and startup system options.
- **System Options:** The currently applied system options. For more information about this topic, please refer to the **Configuration Files** section of the **WPS Workbench** user guide.
- **WPS License Information:** Full details about your licence key.
- **WPS Software Information:** Details about the WPS software, including the version number.

Processing Engine **LOCALE** and **ENCODING** settings

Specifying locale and encoding options for a WPS server.

Any **LOCALE** option defined in a configuration file is ignored when running SAS language programs or Workflows from Workbench. To be effective, the **LOCALE** must be set as a Workbench start-up option.

In order to display and save programs and other files that contain characters for your selected **LOCALE**, you may need to set a General text file encoding [🔗](#) (page 37) value for the Workbench.

Set **LOCALE** for a WPS server

The locale used by WPS is set individually by the particular server. To set the language system option for the locale for a server:

1. In the **Link Explorer** or **Workflow Link Explorer**, ensure the required Processing Engine is running.
2. Right-click the server and click **Properties** on the shortcut menu to display the **Properties** dialog box.
3. In the properties list, click **Startup** and then click **System Options**.
4. On the **System Options** panel, click **Add** to display the **Startup Option** dialog box.
5. In the **Name** field, type **LOCALE**, and in the **Value** field enter the required locale value.

A list of valid **LOCALE** values can be found in the *WPS Reference for Language Elements*.

6. Click **OK** to save the changes restart the server to apply your changes.

Set ENCODING for a WPS server

You may also need to set an appropriate ENCODING value on the server (for example to execute a WPS installation containing non-ASCII characters). As an example, proceed as follows to set UTF-8 on the server:

1. In the **Link Explorer** or **Workflow Link Explorer**, ensure the required Processing Engine is running.
2. Right-click the server and click **Properties** on the shortcut menu to display the **Properties** dialog box.
3. In the properties list, click **Startup** and then click **System Options**.
4. On the **System Options** panel, click **Add** to display the **Startup Option** dialog box.
5. In the **Name** field, type ENCODING, in the **Value** field type UTF-8 and click **OK**.

A list of valid ENCODING values can be found in the *WPS Reference for Language Elements*.

6. Restart the server when prompted.

General text file encoding

To display and save programs and files that contain international characters, you may need to set an appropriate text file encoding.

Text file encoding can be done at two levels:

- Global, for all Workbench projects and associated files.
- Project, for programs contained in the specific project. This can be used where a different encoding is required to the inherited global encoding option.

Set the encoding at a global level

To set a global encoding value:

1. Click the **Window** menu and then click **Preferences** to display the **Preferences** dialog box.
2. In the preferences list, expand **General** and then click **Workspace**.
3. In the **Workspace** panel, under **Text File Encoding**, click **Other** and select the required value from the list.

If the required encoding option is not in the drop-down list, enter the encoding name, for example `Shift_Jis` in the **Other** field.

4. Click **OK** to apply your change.

Check the locale is set for your country so that data is handled correctly by the WPS server. The locale is displayed in the bottom right hand corner of Workbench (for example, to `FR_FR` if you are in French territories). If not set, specify the `LOCALE` and `ENCODING` settings. For more information, see [WPS server LOCALE and ENCODING settings](#) (page 36).

Setting encoding at a project level

To set a project encoding value:

1. In the **Project Explorer** view, select the required project, right-click and on the shortcut menu click **Properties**.
2. In the **Properties for...** dialog box, in the properties list, click **Resource**.
3. On the left of the **Properties** window, ensure the **Resource** option is selected.
4. In the **Resource** panel, under **Text File Encoding**, click **Other** and select the required value from the list.

If the required encoding option is not in the drop-down list, enter the encoding name, for example UTF-8 in the **Other** field.

5. Click **OK** to apply your change.

Setting the encoding at a global level

Changing text file encoding for the whole of Workbench.

To set a global encoding value:

1. Click the **Window** menu and then click **Preferences** to display the **Preferences** dialog box.
2. In the preferences list, expand **General** and then click **Workspace**.
3. In the **Workspace** panel, under **Text File Encoding**, click **Other** and select the required value from the list. If the required encoding option is not in the drop-down list, enter the encoding name, for example `Shift_Jis` in the **Other** field.
4. Click **OK** to apply your change.

Check the locale is set for your country so that data is handled correctly by the Processing Engine. The locale is displayed in the bottom right hand corner of Workbench (for example, to `FR_FR` if you are in French territories). If not set, specify the `LOCALE` and `ENCODING` settings. For more information, see WPS server `LOCALE` and `ENCODING` settings [↗](#) (page 36).

Setting encoding at a project level

Changing text file encoding for a project.

To set a project encoding value:

1. In the **Project Explorer** view, select the required project, right-click and on the shortcut menu click **Properties**.
2. In the **Properties for...** dialog box, in the properties list, click **Resource**.
3. On the left of the **Properties** window, ensure the **Resource** option is selected.

4. In the **Resource** panel, under **Text File Encoding**, click **Other** and select the required value from the list. If the required encoding option is not in the drop-down list, enter the encoding name, for example UTF-8 in the **Other** field.
5. Click **OK** to apply your change.

Licence key

A valid licence key is required to run programs on a Processing Engine. Each Processing Engine, whether local or remote, requires a separate licence key.

A licence key is provided in a file ending in **.wpskey** that is separate from the WPS installation file.

Licenses may only be applied locally from the host that is running the Processing Engine.

Applying a licence key

How to apply a new licence key to your local WPS server.

If you are using a Microsoft Windows operating system, ensure that you have administrator privileges before applying a licence key.

1. Click the **Help** menu and then click **Apply Licence**.
2. In the **Import licence for server Local Server** dialog box, either:
 - Copy and paste the entire contents of your licence file into the window.
 - Click **Import from file**. Find your licence key file and click **Open**.
3. Click **Finish** to apply the licence.

Following installation, you can view the details of the licence. To do so click the **Help** menu and click **View Licence**.

Database connectivity

WPS can be used in conjunction with many different database management systems.

WPS can be used in conjunction with many different database management systems. This section covers installation of ODBC (Open Database Connectivity) and some of the most commonly-used native clients:

- Oracle, through the Oracle Instant Client.
- DB2, through the IBM Data Server Runtime Client.

- SQL Server, through the SQL Server Native Client.
- MySQL through the MySQL Connector/C.
- Sybase Adaptive Server Enterprise through the Sybase ASE Client

Wherever possible, you should use the native database client to obtain the best support for database options and the SAS language. If you are unable to use a native database client, WPS supported database connectivity using ODBC. For more information about creating an ODBC connection on both Microsoft Windows and Linux systems.

Supported client drivers

The following client connector versions are supported:

Engine	Client version
Oracle	Instant Client 12
DB2	IBM Data Server Runtime Client 10.5
SQL Server	Microsoft SQL Server 2012 Native Client
MySQL	Connector/C 6.1
Sybase ASE	Sybase ASE 15.7

If you use ODBC on Linux, WPS only supports unixODBC version 2.3.2 or later

Connect to Oracle

How to install the Oracle Instant Client on Microsoft Windows.

Before installing the Oracle Instant Client software, ensure that:

- You have installed and licensed WPS.
- You have the requisite administrator privileges to install both WPS and the Oracle Instant Client.
- You download the version of the Oracle Instant Client that matches the version of the Oracle database to which you are connecting.

When installing the Oracle Instant Client, you must match the WPS Workbench installation type :

- The Windows (32-bit) client is required for the 32-bit version of WPS Workbench
- The Windows (x64) client is required for the 64-bit version of WPS Workbench.

The Oracle Instant Client must be installed on the same PC or server as the WPS server on which your SAS language programs will be run.

To install the Oracle Instant Client:

1. Download the basic Oracle Instant Client from the Instant Client download page of the Oracle website.

2. Once the appropriate installation (.zip) file has been downloaded, create a folder for the instant client on your PC, for example C:\oracle, and unzip the content of the file into that folder:
3. Add the Oracle Instant Client installation directory to the system *Path* variable, as follows:
 - a. In the **Control Panel**, select the **System and Security** group. Select **System** and click **Advanced System Settings**.
 - b. In the **Advanced** tab, click **Environment Variables**. Select the *Path* entry in the **System variables** list and click **Edit...**
 - c. In the **Edit environment variable** dialog, click **New** and enter the installation directory path.
 - d. Click **OK** in the **Edit environment variable** dialog to save the changes, and close the remaining system and security dialogs.
4. Start WPS Workbench to ensure that the *Path* environment changes to the path are available.

You can view the *Path* variable in WPS Workbench using the following SAS language program:

```
DATA _null_;  
  
LENGTH pathname $ 32760;  
pathname = SYSGET('PATH');  
PUT pathname= ;  
RUN;
```

The log output should contain the Oracle Instant Client installation directory.

Once the client has been installed, test the connection using the following:

```
LIBNAME DATASRC ORACLE USER=<user_name> password=<password> PATH='<remote-id>/TNS';  
PROC DATASETS LIBRARY=DATASRC;  
RUN;
```

Replace *<user_name>* and *<password>* with your user name and password for the server. The *PATH* option contains the server name (specified as *<remote-id>*) and, optionally, a TNS name.

The *PROC DATASETS* statement returns the names of all tables in the selected database; for databases with large numbers of tables, this program may take some time to run.

Connect to DB2

How to install the IBM Data Server Runtime Client, on Microsoft Windows.

Before installing the IBM Data Server Runtime Client, ensure that:

- You have installed and licensed WPS.
- You have the requisite administrator privileges to install both WPS and the IBM Data Server Runtime Client,.

When installing the IBM Data Server Runtime Client,, you must match the WPS Workbench installation type :

- The Windows 32-bit AMD and Intel x86 client is required for the 32-bit version of WPS Workbench.
- The Windows AMD64 and Intel EM64T client is required for the 64-bit version of WPS Workbench.

The IBM Data Server Runtime Client, must be installed on the same PC or server as the WPS server on which your SAS language programs will be run.

To install the IBM Data Server Runtime Client,:

1. Download the IBM Data Server Runtime Client, from the Download Clients and Drivers page of the IBM website.
2. Once downloaded, locate and run the executable installation file. Follow the instructions in the wizard for the *Typical* installation type.

Note:

For ease of future use, you are advised to ensure that the installation folder is on the system path.

3. Set the *DB2CMDEXE* environment variable to point to the *db2cmd.exe* file (typically *C:\Program Files\IBM\SQLLIB\BIN\db2cmd.exe*), as follows:
 - a. In the **Control Panel**, select the **System and Security** group. Select **System** and click **Advanced System Settings**.
 - b. In the **Advanced** tab, click **Environment Variables**. In the **System variables** section, click **New...**
 - c. Create a system variable with a **Variable name** of *DB2CMDEXE*, and a **Variable value** of the IBM Data Server Runtime Client, installation directory, for example *C:\Program Files\IBM\SQLLIB\BIN\db2cmd.exe*

4. Configure the connection from the IBM Data Server Runtime Client, to the server you will access:
 - a. Open a command prompt and run DB2CMD.
 - b. In the new window, type DB2 to access the command line processor for the DB2 client.
 - c. Create a *node* to reference the remote database, using the following command:

```
catalog TCPIP node <node_name> REMOTE <server> SERVER <port_service>
```

Where:

- *<node_name>* is the local connection name.
 - *<server>* is the IP address or name of the server hosting the database.
 - *<port_service>* is either the server port number or the database instance name on the server.
- d. Create a database instance reference (that you will connect to through the *<node_name>*), using the following command:

```
catalog database <database_name> AS <alias_name> AT NODE <node_name>
```

Where:

- *<database_name>* is the instance of the database on the DB2 server.
 - *<alias_name>* is the name you will use in the DB2 client to connect to the database instance.
 - *<node_name>* is the previously-created connection name.
- e. Connect to the local connection name (*<node_name>*) using the following command:

```
connect to <node_name> user <user_name> using <password>
```

Where:

- *<node_name>* is the previously-created connection name.
- *<user_name>* and *<password>* are your user ID and password for the DB2 server.

If the connection is successful, the database information is displayed and you can close the command line processor for the DB2 client:

```
Database Connection Information
Database server          = DB2/LINUX
SQL authorization ID    = TEST
Local database alias    = TESTDB2
```

Once the client has been installed, test the connection using the following:

```
LIBNAME DATASRC DB2 user=<user_name> password=<password> DSN=TESTDB2;
PROC DATASETS LIBRARY=DATASRC;
RUN;
```

Replace *<user_name>* and *<password>* with your user name and password for the server.

The PROC DATASETS statement returns the names of all tables in the selected database; for databases with large numbers of tables, this program may take some time to run.

Connect to SQL Server

How to install the Microsoft SQL Server Native Client on Microsoft Windows

Before installing the Microsoft SQL Server Native Client, ensure that:

- You have installed and licensed WPS.
- You have the requisite administrator privileges to install both WPS and the Microsoft SQL Server Native Client.

When installing the Microsoft SQL Server Native Client, you must match the WPS Workbench installation type

- The X86 client is required for the 32-bit version of WPS Workbench.
- The X64 client is required for the 64-bit version of WPS Workbench.

The Microsoft SQL Server Native Client must be installed on the same PC or server as the WPS server on which your SAS language programs will be run.

To install the Microsoft SQL Server Native Client:

1. Download the Microsoft SQL Server Native Client installer from the SQL Server Native Client page of the Microsoft website.
2. Once the file has been downloaded, execute the file and follow the onscreen instructions to install the Microsoft SQL Server Native Client.

Once the client has been installed, test the connection using the following:

```
LIBNAME DATASRC SQLSrvr user=<user_name> password=<password> server=<remote-id>;  
PROC DATASETS LIBRARY=DATASRC;  
RUN;
```

Replace <user_name> and <password> with your user name and password for the <remote-id> server.

The PROC DATASETS statement returns the names of all tables in the selected database; for databases with large numbers of tables, this program may take some time to run.

Connect to MySQL

How to install the MySQL Connector/C client on Microsoft Windows.

Before installing the MySQL Connector/C client software, ensure that:

- You have installed and licensed WPS.
- You have the requisite administrator privileges to install both WPS and the MySQL Connector/C client.

When installing the MySQL Connector/C client, you must match the WPS Workbench installation type:

- The Windows (x86 32-bit) client is required for the 32-bit version of WPS Workbench
- The Windows (x86 64-bit) client is required for the 64-bit version of WPS Workbench.

Note:

The library names for the different versions of the MySQL Connector/C client are identical, so you can only have one version of the client (either 32-bit or 64-bit) registered on the *Path* system variable.

The MySQL Connector/C client must be installed on the same PC or server as the WPS server on which your SAS language programs will be run.

To install the MySQL Connector/C client

1. Download the MySQL Connector/C client installer (.msi) from the Download Connector/C page of the MySQL website.
2. Once the installation (.msi) file has been downloaded:
 - a. Double-click the downloaded file to begin installation (you will need to read and accept the MySQL licence agreement).
 - b. Select a *Typical* installation, and click **Install**.
3. Add the MySQL Connector/C client installation directory containing `libmysql.dll` to the system *Path* variable, as follows:
 - a. In the **Control Panel**, select the **System and Security** group. Select **System** and click **Advanced System Settings**.
 - b. In the **Advanced** tab, click **Environment Variables**. Select the *Path* entry in the **System variables** list and click **Edit...**
 - c. In the **Edit environment variable** dialog, click **New** and enter the installation directory path.
 - d. Click **OK** in the **Edit environment variable** dialog to save the changes, and close the remaining system and security dialogs.
4. Start WPS Workbench to ensure that the *Path* environment changes are available.

You can view the *Path* variable in WPS Workbench using the following SAS language program:

```
DATA _null_;
  LENGTH pathname $ 32760;
  pathname = SYSGET('PATH');
  PUT pathname= ;
RUN;
```

The log output should contain the MySQL Connector/C client installation directory.

Once the client has been installed, test the database connectivity by using the following in WPS Workbench:

```
LIBNAME DATASRC MYSQL USER=<user-name> PASSWORD=<password>
  SERVER=<remote-id> DATABASE=<dbase>;
PROC DATASETS LIBRARY=DATASRC;
RUN;
```

In the `LIBNAME` statement, replace `<user-name>` and `<password>` with your user name and password for the `<remote-id>` server, and replace `<dbase>` with the name of the MySQL database.

Note:

You must supply a `DATABASE` name in for WPS to successfully connect to the MySQL server.

The `PROC DATASETS` statement returns the names of all tables in the selected database; for databases with large numbers of tables, this program may take some time to run.

Connect to a database using ODBC

ODBC (Open Database Connectivity) is a database-independent connection option for WPS. You should use ODBC as the connection option where no native connection client exists.

ODBC provides the facility to create a SAS language program that stores data in one RDBMS, and then to modify the database storing the data without the need to modify your program significantly.

The version of ODBC driver you select must match the WPS installation type:

- If you have the 32-bit version of WPS installed, select a 32-bit ODBC driver.
- If you have the 64-bit version of WPS installed, select a 64-bit ODBC driver.

Using ODBC may limit the functionality that would otherwise be available through a native client, for example when using *Explicit passthrough* commands in `PROC SQL`. (*Explicit passthrough* uses SQL as understood by the server to which you are connecting, and passes those statements to the RDBMS verbatim.)

Each ODBC driver requires the creation of a DSN (Data Source Name), and the information required when creating this will vary between databases.

The ODBC Client must be installed, and DSN created on the same PC or server as the WPS server on which your SAS language programs will be run.

The following example creates a DSN for the SQLite ODBC driver:

1. Download the ODBC driver for SQLite from the SQLite ODBC Driver website.
2. Once downloaded, double-click the executable file and follow the on-screen instructions to install the driver.
3. Create an empty file on your PC to hold the database, for example `E:\data\mydata.sqlite`.

4. Create a DSN for the ODBC driver, as follows.
 - a. In the **Control Panel**, select **Administrative tools** and then choose either **ODBC Data Sources (32-bit)** or **ODBC Data Sources (64-bit)** as appropriate.
 - b. On the **User DSN** tab, click **Add...** and select **SQLite3 ODBC Driver** in the **Create New Data Source** dialog.
 - c. In the **SQLite3 ODBC DSN Configuration** dialog, enter the following information:
 - **Data Source Name** – the name to be referenced by the SAS language programs run in WPS, for example `SQLITE`.
 - **Database Name** – the file name for the database, for example `E:\data\mydata.sqlite`.

Other information is optional, but may provide better data management within the database. For example, to enforce foreign key constraints, select **Foreign Keys** in the **Configuration** window. Further details on the options can be found in the SQLiteODBC documentation.
 - d. Once all required options have been selected, click **OK** to create the DSN, and close the **ODBC Data Sources** dialog.

Once the DSN has been created, test the database connectivity by using the following:

```
PROC SQL;
  CONNECT TO ODBC (DATASRC=<dsn-name>);
  EXECUTE (CREATE TABLE test (ID INTEGER, name TEXT)) BY ODBC;
  DISCONNECT FROM ODBC;
QUIT;
```

Replace `<dsn-name>` with the **Data Source Name** entered in the **DSN configuration** dialog.

If successful, the WPS log should contain the following information:

```
CONNECT TO ODBC (DATASRC=SQLITE);
NOTE: Successfully connected to database ODBC as alias ODBC.
EXECUTE (CREATE TABLE test (ID INTEGER, name TEXT)) BY ODBC;
The statement completed successfully.
NOTE: Successfully passed statement to database ODBC.
```

Restarting Processing Engines

Processing Engines may be manually restarted at any time.

The consequences of restarting a Processing Engine depend on the environment:

- **SAS Language Environment:** WPS Servers may be restarted as a troubleshooting step, as well as a quick and easy way to clear data associated with a WPS Server (the log, all the output results, file references, library references, datasets, and so on).

- **Workflow Environment:** Engines may be restarted as a troubleshooting step. This will not clear logs and datasets, as these are held by the Workflow upon execution, rather than by the Processing Engine (as in the SAS Language Environment). Because multiple Workflow engines are managed automatically by Workbench, you cannot close individual engines from a set. Instead, a restart action closes all open engines and starts up a new single engine, from which point Workbench decides afresh whether to open multiple engines to run the Workflow.

Restarting a WPS Server

Restarting a WPS Server, both simply and with advanced server actions.

Each time that the Workbench is closed, the temporary results associated with the session (the log, all the output results, file references, library references, datasets, and so on) are cleared. If you want to clear the same temporary results prior to the end of the Workbench session, you can restart the WPS server. Restarting the server means that programs will no longer be able to interact with temporary objects created in the previous server session.

Alternatively, you can clear the log or results and keep all the other temporary output.

To restart the server:

1. Click the **WPS** menu, click **Restart Server** (SAS Language Environment) or **Restart Engine** (Workflow Environment) and then click the server or engine you want to restart. If you are restarting the local server or engine, you can use **Ctrl + Alt + S**.
2. A confirmation dialog box may be displayed. If so, click **OK** to restart the server. You will not see the confirmation dialog if you have previously selected the **Do not show this confirmation again** option in this dialog box.

Restarting with Advanced Server Actions

If you have selected **Show advanced server actions** in the **WPS Preferences** page of the **Preferences** window, the shortcut menu for the WPS server or engine in the **WPS Server Explorer**, **Workflow Link Explorer**, **Output Explorer** and **Results Explorer** views displays alternative restart options:

- **Restart, new WORK** – restart and clear the contents of the `WORK` library.
- **Restart, keep WORK** – restart and keep the contents of the `WORK` library.

Restarting Workflow Engines

Restarting a Workflow Engine.

To restart a Workflow engine, click the **WPS** menu, click **Restart Engine** and then click the engine you want to restart.

Workbench Views

The layout of the items in Workbench is known as a *perspective* and comprises individual windows that contain one or more *views*. These views display specific types of information and provide specific functions. Some views also have context-specific toolbars located in the top right of each view.

- Some views are the same in both SAS Language Environment and Workflow Environment (see [Generic views](#) (page 52)).
- Some views are specific to the SAS Language Environment (see [SAS Language Environment views](#) (page 78)).
- Some views are specific to the Workflow Environment (see [Workflow Environment views](#) (page 88)).

Working with views

This section describes how to manipulate the Workbench views both individually and as part of overall view stacks.

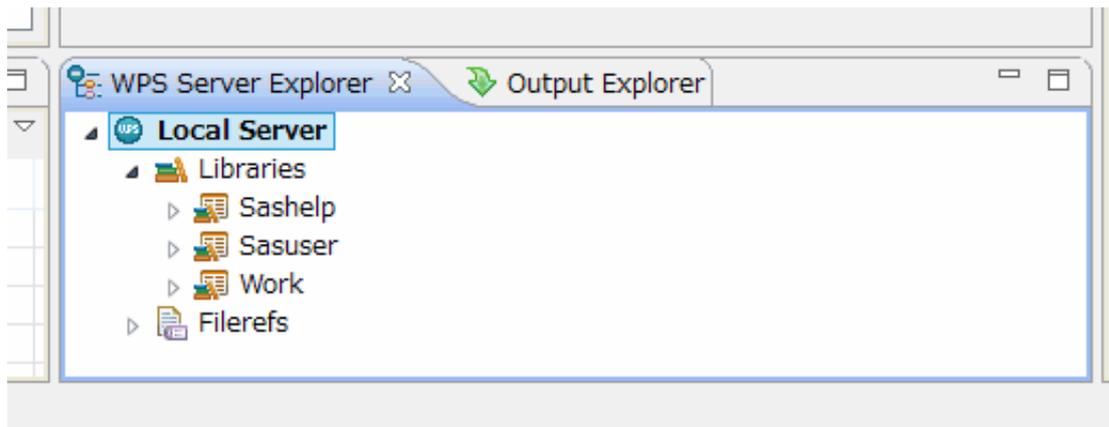
For more information on view stacks, see [view stacks](#) (page 49).

View stacks

A view stack is a window formed by a collection of views that are all allocated the same area in a perspective.

You can move, minimize and maximize views using a view stack. When a view is reopened, that view opens in the view stack to which it was last attached.

For example, the view stack below contains two views:



Resizing a view

Resizing a view by resizing its view stack.

In order to resize a view, you have to resize its associated view stack, which will then resize all the views within it.

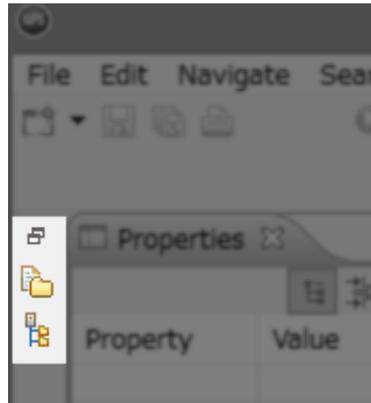
1. Move your mouse over the border between the particular view stack or window, and its adjacent view stack or window (whether this is to the side, above or below), until the cursor changes to the \leftrightarrow or \updownarrow resize icon.
2. Press your left mouse button and keep it pressed while you drag the border side to side, or up and down.
3. Release the mouse button to complete the resize operation.

Minimising a view

Minimising a view by minimising its view stack.

To minimise a view, you have to minimise its associated view stack. To minimise a view stack or window, click the **Minimise** button on the view stack or window.

Once minimised, the views are represented by icons on either the left or right of the Workbench, so that you can restore them:



Maximising a view

Maximising a view by maximising its view stack.

To maximise a view, you have to maximise its associated view stack, which fills the Workbench interface.

Maximising a view causes it to occupy all available space in the Workbench window. All other view stacks are minimised to the edges of the Workbench window, where they are represented as icons.

To maximise a view stack or window, click the **Maximise** button on the view stack or window.

Restoring a view

Restoring a view by restoring its view stack.

To restore a view to its 'normal' state after it has been minimised or maximised, you have to restore its associated view stack.

To restore a view stack or window that has been maximised, click on the **Restore** button on the view stack or window.

To restore a view stack or window that has been minimised, click the **Restore** button above the minimised view stack to restore all the views in that stack.

Opening a view

Opening a view.

To open a view:

1. Click the **Window** menu, click **Show View** and then click **Other**.

2. In the **Show View** dialog box, either enter the view name or select the required view from the list of views. Click **OK** to display the view

The view is opened in the view stack to which it was last attached. If the required view is already open, it will become active in Workbench.

Detaching and reattaching a view

Detaching a view so it can be moved to a different view stack.

You can move a view between view stacks in Workbench or into a new detached window.

To detach a view:

Click on the view tab or title bar and drag the view to a different view stack. Alternatively, drag the view away from the Workbench window to create a new view stack. If you close a detached view stack, it will still be detached when you next open it.

To attach a view to a view stack:

Click on the view tab or title bar and drag the view over the required view stack. Use the guides to position the view in the view stack.

Generic views

Views that apply to both the SAS Language Environment and the Workflow Environment.

Project Explorer

The **Project Explorer** view is used to edit and manage SAS language programs and Workflows, along with related files, programs or datasets; all organised into projects.

Projects can store any kind of local file, and support file system operations such as running programs. Objects within projects can be managed with Local History features (see [Local history](#) (page 59)), and they can also be exported to archives or to other folders in the local file system. The **Project** view cannot be used to manage files stored on a remote file system or files on a local file system that are not part of the project.

The **Project Explorer** view only displays projects that are in your current workspace. You can create several Projects in a workspace (see [Projects](#) (page 53)), and use each project for a specific task.

Selecting an item in the **Project Explorer** displays information about that item in the **Properties** view.

When in Workflow Environment perspective, if you change the default Workflow Engine to a remote host, the location of the projects and Workflow files does not change and remains on the local host. Any references to datasets in the workspace or project cannot be used with a remote Workflow Engine, and must be moved to the remote host using the **File Explorer** view, and any references in the Workflow changed to the appropriate filepath for the remote host.

Displaying the view

To display the **Project Explorer** view:

From the **Window** menu, click **Show View** and then click **Project Explorer**.

Operating system tools

As well as the Workbench's tools, you can use those supplied by your operating system to delete, cut, copy and paste objects into a project. If you do use the operating system tools rather than the Workbench tools, modifications are not preserved in the local history.

You may need to refresh the **Project Explorer** view to ensure that objects manipulated in this way are displayed correctly. To do this right-click the project and, click **Refresh** on the shortcut menu (or select the file and press **F5**).

Projects

A *project* is a Workbench representation of a folder, containing objects such as programs, datasets, files and folders.

WPS Analytics uses a project to hold the files and folders that relate to each other. For example, you might have one project containing programs that are still under development and another project for monthly reporting applications. There is no limit to the number of projects you can have open within Workbench.

Each project is defined by a project definition file named *.project* stored in the root directory of a project. This file can only be viewed in the system file browser and must not be deleted, edited, moved from the project folder or copied into another project.

Only one workspace can be open at a time in Workbench, but you can switch between workspaces as required. The **Project Explorer** view is used to display and manage the current workspace and projects in the active workspace.

Creating a new project

Creating a new project in Workbench.

To create a new project:

1. Click the **File** menu, click **New** and then click **Project** to launch the **New Project** wizard.
2. Expand the **General** folder, click **Project** and click **Next**.
3. Enter the new **Project name**.
4. To create the new project in a location other than the current workspace folder, clear **Use default location** and specify a new directory in the **Location** field.
5. Click **Finish** to create the new project.

The new project is displayed in the **Project Explorer** view. If the new project location is not in the workspace folder, the path to the project folder is displayed in the **Properties** view.

You can also use the Workbench's copy and paste facility to create a new project based on an existing project.

Creating a new folder in a project

Creating a new folder in a project.

To create a new folder:

1. On the **File** menu, click **New** and then click **Folder** to launch the **New Folder** wizard.
2. The parent for the folder defaults to either the current project or folder selected in the **Project Explorer** view.
3. Enter a **Folder name** for the new folder.
4. To change the location of the folder on the file system click **Advanced** and select one of:
 - **Use default location.** The folder is created as a file system folder, in the same location as the parent folder or workspace.
 - **Folder is not located in the file system (Virtual Folder).** The folder only exists as a workspace artifact, no folder is created on the file system. You can drag-and-drop your other project resources into the virtual folder to create links to other files and folders in this new folder, and also add other virtual folders. It is not possible to create physical files and folders within virtual folders.
 - **Link to alternate location (Linked Folder).** Creates the folder on the file system in a different location to the parent folder or workspace. Click **Browse** to select the location on your file system to which to link.
5. Click **Finish** to create the new folder.

Moving or copying a project file

Moving any folder, file, program, or any other object, between projects, or between folders within the same project.

1. In the **Project Explorer** view, select the object you need to move.

2. Click the **File** menu, and then click **Move**. In the **Move Resources** dialog box, select the new destination project or folder and click **OK** to move the resource.

If you want to duplicate rather than move the object, click the **Edit** menu and then click **Copy**; select the target folder or project, click the **Edit** menu and then click **Paste**. If you duplicate the object in the same project or folder as the original, you must provide a different name for the copied object.

Note:

Files can also be dragged and dropped between folders.

To return the object to its original location, click the **Edit** menu and then click either **Undo Move Resource** if the object was moved, or **Undo Copy Resource** if the object was duplicated.

Linking the active file to the Project Explorer

If you have several files from different projects open in the **Editor** view, you can configure the **Project Explorer** view to highlight the program in the project or folder each time that you switch the focus to that program in the **Editor** view.

To link the **Project Explorer** view and the **Editor** view, in the **Project Explorer** view click **Link with Editor** .

Deleting an object from a project

Deleting (and undoing deletion for) an object in a project.

To delete an object from a project:

1. In the **Project Explorer** view, select the required object to delete.
2. Click the **Edit** menu, and then click **Delete**. In the **Delete Resources** dialog box, click **OK**.

To return the object to its original location, click the **Edit** menu and click either **Undo Delete Resource**.

Renaming a project or project file

Renaming a project or project file.

To rename a project, or an object in a project:

1. In the **Project Explorer** view, select the required object or project.
2. Click the **File** menu and then click **Rename**.
3. Enter the **New name** in the **Rename Resource** dialog box and click **OK**.

Copying a project

You can use the Workbench's  **Copy** and  **Paste** facility to create a new project based on an existing project.

1. In the **Project Explorer** view, select the required project.
2. Click the **Edit** menu, and then click **Copy**.
3. Click the **Edit** menu, click **Paste** and in the **Copy Project** dialog box, enter a new **Project name** .

If you want to copy the project to a location other than the current workspace, clear **Use default location** and enter an alternate **Location** for the new project.

4. Click **OK** to create the project.

Importing files or archives

You can import files from either an archive file (such as a ZIP file) or from the file system into an existing project.

To import files:

1. Click the **File** menu and then click **Import** to open the **Import** wizard.
2. Expand the tree under the **General** node and select one of the following options:
 - **File System** – to import files and folders not archived in a single file.
 - **Archive File** – to import files and folders that are archived in one file (for example, a *zip*, or *tar.gz*).
3. Click **Next** and click **Browse** to select either the archive file or directory.
4. You will see a list of the files and folders that are available to import. Ensure that all the objects that you want to import are selected.

Note:

Do not select the parent folder of the files to import; if you do, your imported files are added to a subfolder in your project.

Note:

Do not import a project definition file (*.project*); if you do the imported file will overwrite the project definition file in the existing project

5. Click **Browse** and specify the **Into folder** – the target folder for the imported files.
6. Click **Finish** to import the selected files.

Closing and reopening a project

You can close or open a project to control the number of available projects in a workspace.

Closing a project collapses the tree under project in the **Project Explorer** view and closes any programs or other files that were open in the Workbench **Editor** view.

To close a project, in the **Project Explorer**, right-click an open project (📁) and click **Close Project** on the shortcut menu.

If you want to open or reopen a closed project, in the **Project Explorer**, right-click a closed project (📁) and click **Open Project** on the shortcut menu.

Exporting a project

Making a copy or backup of a project and all the files and folders contained within it.

1. In the **Project Explorer** view, select the project to export.
2. Click the **File** menu and then click **Export** to display the **Export** wizard.
3. On the **Select** page, expand the **General** node and select
 - **Archive File** to create *zip* file in the export location.
 - **File System** to makes a copy of your project folder and its contents.
4. Click **Next** and select the folders and files to export from the selected project
5. If you are exporting to an archive, enter the path for file, including filename in **To archive file**. If you are exporting to the file system, enter the folder path in **To directory**.
6. Click **Finish** to export the selected folders and files.

Deleting a project

Deleting a project.

1. In the **Project Explorer** view, select the required project to delete.
2. Click the **Edit** menu and click **Delete**.
3. In the **Delete Resources** dialog box, choose whether to delete the reference to the project in the current workspace, or delete the reference and the folder contents on disk:
 - a. to remove the workspace reference to the project without deleting the project folder and contents, clear **Delete project contents on disk (cannot be undone)**.
 - b. To remove the workspace reference and folder contents from the disk, select **Delete project contents on disk (cannot be undone)**.
4. Click **OK**.

Comparing and merging multiple files

You can carry out either two or three-way comparisons between different versions of a file. The comparison tool is purely text based, so will not produce a meaningful result for Workflow files.

Three-way comparisons show the differences between three different versions of a file, for example the differences between the resource in the Workbench, the version of the resource that is committed in a branch, and the *common ancestor* on which the two conflicting versions are based. If a *common ancestor* cannot be determined, for example because a resource with the same name and path was created and committed by two different developers, the comparison becomes a two-way comparison.

1. In the **Project Explorer**, highlight the files that you want to compare. You can select up to three different files.
2. Right-click, and on the shortcut menu, click **Compare With** and then click **Each Other**.
 - In a two-way comparison, the **Text Compare** window displays the differences (marked in grey) between the two files.
 - In a three-way comparison, select the *common ancestor* – the file against which comparisons will be made. The **Text Compare** window displays the two file versions. To display the *common ancestor* at the same time, click **Show Ancestor Pane**.
3. Make the required changes to the files

You can navigate through the files using the text compare toolbar:

- Click **Next Difference** to show the first difference between the files *after* the cursor location or highlighted text.
- Click **Previous Difference** to show the first difference between the files *before* the cursor location or highlighted text.
- Click **Next Change** to show the first change between file versions *after* the cursor location or highlighted text.
- Click **Previous Change** to show the first difference between file versions *before* the cursor location or highlighted text.

You can merge changes from one file to the other using the text compare toolbar:

- Click **Copy Current Change from Right to Left** to overwrite the highlighted text in the left file with the highlighted text in the right file.
 - Click **Copy All from Right to Left** to replace the content of the left file with the content of the right file.
 - Click **Copy Current Change from Left to Right** to overwrite the highlighted text in the right file with the highlighted text in the left file.
 - Click **Copy All from Left to Right** to replace the content of the right file with the content of the left file.
4. Once you have made your changes, click the **File** menu and then click **Save**.
 5. Click the **File** menu and then click **Close** to close the **Text Compare** window.

Local history

The Workbench maintains a local history of modifications to programs or other project files changed in the **Project Explorer** view.

Each time you save a project file in the Workbench, a snapshot of the current contents of the file is added to the Workbench's local history. This history provides the ability to take any of your current programs and compare or replace them with local history.

History of Deletions

Each time you delete a program or file, the edit history of the item being deleted is added to the local history of the containing project or folder. This enables you to restore state from local history, or to select which particular state from the item's history to restore. See *Restoring from local history* [↗](#) (page 61) and *Replacing from local history* [↗](#) (page 61) for more information.

Local History Preferences

Local history preferences allow you to control how long to keep the local history, how many entries to keep in the history, and the maximum file size that can be recorded. See *Local history preferences* [↗](#) (page 61) for more information.

Comparing with local history

Project Explorer's comparison tool allows a current file to be compared with a version from local history. The comparison tool is purely text based, so will not produce a meaningful result for Workflow files.

To compare a current saved project file with a previous version:

1. In the **Project Explorer** view, select the required file, right-click, click **Compare With** and then click **Local History** on the shortcut menu.

A **History** window opens listing program revisions with revision times.

2. Double-click the revision with which you want to compare the current version of the file.

The **Text Compare** window opens showing the differences (marked in grey) between the two versions. The current version is shown on the left, with the local history version on the right.

3. Make the required changes to the files.

You can navigate through the files using the text compare toolbar:

- Click **Next Difference** to show the first difference between the files *after* the cursor location or highlighted text.
- Click **Previous Difference** to show the first difference between the files *before* the cursor location or highlighted text.
- Click **Next Change** to show the first change between file versions *after* the cursor location or highlighted text.
- Click **Previous Change** to show the first difference between file versions *before* the cursor location or highlighted text.

You can merge changes from one file to the other using the text compare toolbar:

- Click **Copy Current Change from Right to Left** to overwrite the highlighted text in the left file with the highlighted text in the right file.
- Click **Copy All from Right to Left** to replace the content of the left file with the content of the right file.

4. Once you have made your changes, click the **File** menu and then click **Save**.

5. Click the **File** menu and then click **Close** to close the **Text Compare** window.

Replacing the current version with the previous version from local history

How to revert a program in a project back to the previously saved version.

To replace the current version with the previous version, in the **Project Explorer** right-click the required file and, on the shortcut menu, click **Replace With** and then click **Previous from Local History**.

Replacing the current version with a version from local history

How to revert a program in a project back to a previous saved version.

To replace the current version with a version from the local history:

1. In the **Project Explorer** right-click the required file and, on the shortcut menu, click **Replace With** and then click **Local History**.
2. In the **Compare** dialog, double-click a revision to review and select the required version. Click **Replace** to use the selected version.

Replacing from local history

How to revert a program in a project back to a previous saved version.

You can either replace the current version with the previous version or select a version from the local history.

1. To replace the current version with the previous version, in the **Project Explorer** right-click the required file and, on the shortcut menu, click **Replace With** and then click **Previous from Local History**
2. To replace the current version with a version from the local history:
 - a. In the **Project Explorer** right-click the required file and, on the shortcut menu, click **Replace With** and then click **Local History**.
 - b. In the **Compare** dialog, double-click a revision to review and select the required version. Click **Replace** to use the selected version.

Restoring from local history

Recovering a project file that has been deleted in the **Project Explorer** view.

To restore a project file:

1. In the **Project Explorer** view, right-click either the project or folder that the file contained, and then on the shortcut menu click **Restore from Local History**.
2. In the **Restore from Local History**, select the files to restore:
 - a. Click the check box next to the project file name To restore the last version of that file.
 - b. If a selected file has a saved local history, select the file name and choose the required version of the file in the **Select an edition of a file** panel.
3. Click **Restore** to recover the selected files and close the **Restore from Local History** dialog box.

Local history preferences

Changing the time period for retention of local history for each project file.

By default, the Workbench keeps seven days of local history for each individual project file, with a maximum of fifty versions for each file. There is also a default of 1 MB of storage allocated for each program's local history.

The above settings can be changed as follows:

1. Select **Window > Preferences... > General > Workspace > Local History**.

The **Local History** dialog is displayed.

2. Set any of the following as required:

- **Days to Keep Files** - The number of days that you want to keep records for any one project file. For example, if you type 20, then a history of saved versions from the last twenty days will be kept. The default value is 7 days.
- **Maximum Entries per File** - The number of versions to keep for any one project file. If this number is exceeded, the oldest changes are discarded to make room for the new changes. The default is 50 versions per file.
- **Maximum File Size (MB)** - The maximum file size (in MB) for a project file's local history. If this size is exceeded, then no more local history is kept for the file. The default size is 1MB.

Note:

The **Days to Keep Files** and **Maximum Entries per File** are only applied when the local history is compacted on server shutdown.

Switching to a different workspace

Switching to a different workspace to use a different group of projects.

1. Click the **File** menu, click **Switch Workspace** and click **Other**.
2. In the **Workspace Launcher**, select the required **Workspace** or expand the **Recent Workspaces** list and select the required workspace if it appears in the list.
3. Click **OK** to restart Workbench displaying the selected workspace.

Hub Program Packages

Workbench can be used to create and manage Hub *program packages*. Hub Program Packages are collections of programs deployable as one unit within Hub. Programs can be in the SAS language, R language, or Python language.

Alongside their constituent programs, Hub program packages contain a YAML descriptor file for the package, and individual YAML descriptor files for each program. Workbench can create and manage all of these file types.

Creating a new Hub program package

Creating a new WPS Hub program package from within Workbench.

1. Click the **File** menu, click **New** and then click **Other** to launch the **New** wizard.
2. Under **WPS**, select **Hub Deployment Services Package Project**.
3. Click **Next**.

4. Under **Project name** type a name for the project.
5. Next, choose a storage location for the program package:
 - Select **Use default location** to store the program package in the current Workspace.
 - Clear **Use default location** and specify a location for the program package.
6. Next, choose whether to add the program package to working sets:
 - Select **Add project to working sets** and then choose from the **Working sets** list. To create a new working set, click **New**.
 - Clear **Add project to working sets** to not add the program package to working sets.
7. Click **Finish**.

The new WPS Hub program package is now visible in the **Project Explorer** view. No accompanying YAML file is created at this stage, because an empty program package has no programs for the YAML file to describe.

Creating a new Hub program

Creating a new WPS Hub program from within Workbench.

1. In **Project Explorer** view, locate the program package that you want to create the program within.
2. Right-click the selected program package and select **New**. Now choose from:
 - a.  **SAS Hub Program**
 - b.  **Python Hub Program**
 - c.  **R Hub Program**
3. Type a **Name** for the Hub program and click **OK**.

The new WPS Hub program is now visible underneath its program package in the **Project Explorer** view. If this is the first program in the package, a YAML descriptor file is also created describing the program. If a program or programs already exist, the existing YAML descriptor file is updated to include the new program.

Editing a Hub program

Editing a WPS Hub program from within Workbench. These changes will modify the parent program package's YAML descriptor file.

To edit a Hub program, locate it in **Project Explorer**, then right-click on it and click **Open**. The program will open in a new tab, which has three sub-tabs:

- **Overview**: Defines the program's label, description, categories and a file containing the program code.

- **Parameters:** Defines the variables in the program, allowing their creation, editing and deletion.
- **Results:** Defines the output of the program, allowing their creation, editing and deletion.

These sub-tabs are slightly different for each type of program and are described in detail in the following sections:

- *Editing a Hub SAS language program* [↗](#) (page 64)
- *Editing a Hub Python language program* [↗](#) (page 66)
- *Editing a Hub R language program* [↗](#) (page 68)

Note:

For a more detailed explanation of Hub program options, see the Hub documentation.

Editing a Hub SAS language program

Editing a Hub SAS language program from within Workbench.

To edit a SAS language Hub program, locate it in **Project Explorer**, then right-click on it and click **Open**, or double-click on the program. The program will open in a new tab, with two sub-tabs: **Overview** and **Description**.

Overview sub-tab

The **Overview** sub-tab is divided into four areas, each covered under a heading below.

General Information

Label: The label under which this program is published in the WPS Hub directory. This is only relevant if the program is included in the list of programs to be published within the package descriptor file. If not specified, the program's file name, without extension, is used.

Program File: Specifies the file containing the program code. Two options are available:

- **Browse** allows you to specify an existing file from your workspace containing the program code. Clicking this option opens a **Select program file** window, from which you can select a program from within the workspace and click **OK** to return to the **Overview** sub-tab.
- **Program file** allows you to create a new program file. Clicking this option opens a **New File** window, from which you can browse to a location for the file and specify the name. Finish will create the file and return to the **Overview** sub-tab.

Parameters

Parameter style allows you to specify a parameter style for the . Choose from **Dataset** or **Macro variables**.

Below **Parameter style** is a list of all the parameters for the program. Actions for this list are as follows:

- **Add.** Clicking on this button opens an Add parameter window, where you can enter:
 - **Name:** The variable name used in the program code.
 - **Label:** The label for the variable presented to the user when running the program.
 - **Prompt:** A word or phrase displayed to the user asking them to make their choice.
 - **Hidden:** Hides this variable once the program is published.
 - **Required:** Forces the user to choose this input variable to continue running the program. If the variable is not supplied, an HTTP 400 (bad request) response is given.
 - **Type:** Choose from: **String**, **Password**, **Choice**, **Integer**, **Float**, **Boolean**, **Stream**, **Date**, **Time**, and **Date & time**.

Each **Type** displays its own set of optional settings. With the exception of **Choice**, these optional settings consist of some or all of: a **default value** (value taken if no other value is provided), a **minimum** to validate user input against, and a **maximum** to validate user input against. If **Choice** is specified, further options are presented to create and manage a list of choices, with each choice consisting of a **label** and a **value**.

- **Edit:** With a parameter selected from the list, this button opens an **Edit parameter** window. Options presented are the same as described above for **Add**.
- **Remove:** With a parameter selected from the list, this button deletes that parameter.
- **Up:** With a parameter selected from the list, this button moves that parameter up the list.
- **Down:** With a parameter selected from the list, this button moves that parameter down the list.

Results

The **Results** section contains a list of the program's outputs. Actions for this list are as follows:

- **Add.** Clicking on this button opens an **Add result** window, where you can enter:
 - **Name:** A name for the output.
 - **Result type:** Choose from **Dataset** or **Stream**.
- **Edit:** With a result selected from the list, this button opens an **Edit parameter** window. Options presented are the same as described above for **Add**.
- **Remove:** With a result selected from the list, this button deletes that result.
- **Up:** With a result selected from the list, this button moves that result up the list.
- **Down:** With a result selected from the list, this button moves that result down the list.

Categories

The **Categories** section contains a list of categories (Hub program metadata) under which the program will be listed in the directory, if it is to be published.

Actions for the **Categories** list are as follows:

- **Add:** Adds a new category to the list. To edit the category, click on it and type.
- **Remove:** Removes the selected category or categories from the list.

Description sub-tab

The **Description** sub-tab contains a free text description of the program to place in the directory. This is only relevant if the program is included in the list of programs to be published within the package descriptor file.

Editing a Hub Python language program

Editing a WPS Hub Python language program from within Workbench.

To edit a Python language Hub program, locate it in **Project Explorer**, then right-click on it and click **Open**, or double-click on the program. The program will open in a new tab, with two sub-tabs: **Overview** and **Description**.

Overview sub-tab

The **Overview** sub-tab is divided into four areas, each covered under a heading below.

General Information

Label: The label under which this program is published in the WPS Hub directory. This is only relevant if the program is included in the list of programs to be published within the package descriptor file. If not specified, the program's file name, without extension, is used.

Program File: Specifies the file containing the program code. Two options are available:

- **Browse:** allows you to specify an existing file from your workspace containing the program code. Clicking this option opens a **Select program file** window, from which you can select a program from within the workspace and click **OK** to return to the **Overview** sub-tab.
- **Program file:** allows you to create a new program file. Clicking this option opens a **New File** window, from which you can browse to a location for the file and specify the name. Finish will create the file and return to the **Overview** sub-tab.

Initialisation program file: specifies an initialisation program file. Two options are available:

- **Browse:** allows you to specify an existing file from your workspace. Clicking this option opens a **Select initialisation program file** window, from which you can select a file from within the workspace and click **OK** to return to the **Overview** sub-tab.
- **Initialisation program file:** allows you to create a new Initialisation program file. Clicking this option opens a **New file** window, from which you can browse to a location for the file and specify the name. Finish will create the file and return to the **Overview** sub-tab.

Parameters

Parameter style allows you to specify a parameter style. Choose from **Dataset** or **Macro variables**.

Below **Parameter style** is a list of all the parameters for the program. Actions for this list are as follows:

- **Add.** Clicking on this button opens an Add parameter window, where you can enter:
 - **Name:** The variable name used in the program code.
 - **Label:** The label for the variable presented to the user when running the program.
 - **Prompt:** A word or phrase displayed to the user asking them to make their choice.
 - **Hidden:** Hides this variable once the program is published.
 - **Required:** Forces the user to choose this input variable to continue running the program. If the variable is not supplied, an HTTP 400 (bad request) response is given.
 - **Type:** Choose from: **String**, **Password**, **Choice**, **Integer**, **Float**, **Boolean**, **Stream**, **Date**, **Time**, and **Date & time**.

Each **Type** displays its own set of optional settings. With the exception of **Choice**, these optional settings consist of some or all of: a **default value** (value taken if no other value is provided), a **minimum** to validate user input against, and a **maximum** to validate user input against. If **Choice** is specified, further options are presented to create and manage a list of choices, with each choice consisting of a **label** and a **value**.

- **Edit:** With a parameter selected from the list, this button opens an **Edit parameter** window. Options presented are the same as described above for **Add**.
- **Remove:** With a parameter selected from the list, this button deletes that parameter.
- **Up:** With a parameter selected from the list, this button moves that parameter up the list.
- **Down:** With a parameter selected from the list, this button moves that parameter down the list.

Results

The **Results** section contains a list of the program's outputs. Actions for this list are as follows:

- **Add.** Clicking on this button opens an **Add result** window, where you can enter:
 - **Name:** A name for the output.
 - **Result type:** Choose from **Dataset** or **Stream**.
- **Edit:** With a result selected from the list, this button opens an **Edit parameter** window. Options presented are the same as described above for **Add**.
- **Remove:** With a result selected from the list, this button deletes that result.
- **Up:** With a result selected from the list, this button moves that result up the list.
- **Down:** With a result selected from the list, this button moves that result down the list.

Categories

The **Categories** section contains a list of categories (Hub program metadata) under which the program will be listed in the directory, if it is to be published.

Actions for the **Categories** list are as follows:

- **Add:** Adds a new category to the list. To edit the category, click on it and type.
- **Remove:** Removes the selected category or categories from the list.

Description sub-tab

The **Description** sub-tab contains a free text description of the program to place in the directory. This is only relevant if the program is included in the list of programs to be published within the package descriptor file.

Editing a Hub R language program

Editing a WPS Hub R language program from within Workbench.

To edit an R language Hub program, locate it in **Project Explorer**, then right-click on it and click **Open**, or double-click on the program. The program will open in a new tab, with two sub-tabs: **Overview** and **Description**.

Overview sub-tab

The **Overview** sub-tab is divided into four areas, each covered under a heading below.

General Information

Label: The label under which this program is published in the WPS Hub directory. This is only relevant if the program is included in the list of programs to be published within the package descriptor file. If not specified, the program's file name, without extension, is used.

Program File: Specifies the file containing the program code. Two options are available:

- **Browse:** allows you to specify an existing file from your workspace containing the program code. Clicking this option opens a **Select program file** window, from which you can select a program from within the workspace and click **OK** to return to the **Overview** sub-tab.
- **Program file:** allows you to create a new program file. Clicking this option opens a **New File** window, from which you can browse to a location for the file and specify the name. Finish will create the file and return to the **Overview** sub-tab.

Initialisation program file: specifies an initialisation program file. Two options are available:

- **Browse:** allows you to specify an existing file from your workspace. Clicking this option opens a **Select initialisation program file** window, from which you can select a file from within the workspace and click **OK** to return to the **Overview** sub-tab.
- **Initialisation program file:** allows you to create a new Initialisation program file. Clicking this option opens a **New file** window, from which you can browse to a location for the file and specify the name. Finish will create the file and return to the **Overview** sub-tab.

Parameters

The **Parameters** section contains a list of parameters for the program. Actions for this list are as follows:

- **Add.** Clicking on this button opens an Add parameter window, where you can enter:
 - **Name:** The variable name used in the program code.
 - **Label:** The label for the variable presented to the user when running the program.
 - **Prompt:** A word or phrase displayed to the user asking them to make their choice.
 - **Hidden:** Hides this variable once the program is published.
 - **Required:** Forces the user to choose this input variable to continue running the program. If the variable is not supplied, an HTTP 400 (bad request) response is given.
 - **Type:** Choose from: **String**, **Password**, **Choice**, **Integer**, **Float**, **Boolean**, **Stream**, **Date**, **Time**, and **Date & time**.

Each **Type** displays its own set of optional settings. With the exception of **Choice**, these optional settings consist of some or all of: a **default value** (value taken if no other value is provided), a **minimum** to validate user input against, and a **maximum** to validate user input against. If **Choice** is specified, further options are presented to create and manage a list of choices, with each choice consisting of a **label** and a **value**.

- **Edit:** With a parameter selected from the list, this button opens an **Edit parameter** window. Options presented are the same as described above for **Add**.
- **Remove:** With a parameter selected from the list, this button deletes that parameter.
- **Up:** With a parameter selected from the list, this button moves that parameter up the list.
- **Down:** With a parameter selected from the list, this button moves that parameter down the list.

Results

The **Results** section contains a list of the program's outputs. Actions for this list are as follows:

- **Add.** Clicking on this button opens an **Add result** window, where you can enter:
 - **Name:** A name for the output.
 - **Result type:** Choose from **Dataset** or **Stream**.
- **Edit:** With a result selected from the list, this button opens an **Edit parameter** window. Options presented are the same as described above for **Add**.
- **Remove:** With a result selected from the list, this button deletes that result.
- **Up:** With a result selected from the list, this button moves that result up the list.
- **Down:** With a result selected from the list, this button moves that result down the list.

Categories

The **Categories** section contains a list of categories (Hub program metadata) under which the program will be listed in the directory, if it is to be published.

Actions for the **Categories** list are as follows:

- **Add:** Adds a new category to the list. To edit the category, click on it and type.
- **Remove:** Removes the selected category or categories from the list.

Description sub-tab

The **Description** sub-tab contains a free text description of the program to place in the directory. This is only relevant if the program is included in the list of programs to be published within the package descriptor file.

File Explorer

The **File Explorer** view is used to edit and manage SAS language programs and Workflows, along with related files, programs or datasets on your file system.

Because the **File Explorer** view does not manage content through projects, you cannot use the local history, nor use Workbench functionality to import or export projects or archives.

The **File Explorer** view enables access to all files that are available on your local file system, and on any connected remote hosts. The view also displays the following object types:

- *Connections*: The root node that represents a connection to a host machine. This can be either local or remote.
- *Shortcuts*: A directory on a file system selected for management via the  **File Explorer** view.
- *Symbolic linked file* (SAS Language Environment only): A file that points to another file in the host file system via a symbolic link.
- *Symbolic linked directory* (SAS Language Environment only): A file that points to another directory in the host file system via a symbolic link.

Selecting an item (other than a connection) in the **File Explorer** displays information about that item in the **Properties** view.

Displaying the view

To display the **File Explorer** view:

From the **Window** menu, click **Show View** and then click **File Explorer**.

Connections and shortcuts

It is only possible to view files on remote systems when you have an open host connection. The host connection is opened from the **Link Explorer** or **Workflow Link Explorer** views, or when starting WPS servers in the **WPS Server Explorer** view.

Each host connection can have several directory shortcuts, each providing access to the files and folders on the associated file system.

To create a new directory shortcut:

1. In the **File Explorer** view, right-click on the required folder.
2. Click **New**, and then click **Directory Shortcut**.

3. Enter a **Directory Name** in the **Directory Shortcut** dialog box.

Working with files and folders

It is possible to perform the following operations on files and folders within the **File Explorer**:

- Create a new directory shortcut, directory or file.
- Selected files and directories to be moved, copied or deleted.
- Run a SAS language program on a selected WPS server.
- Rename a selected file or directory.
- Analyse the contents of *SAS Language Environment* programs, and folders containing programs (see *Code Analyser* [↗](#) (page 13) for more information).
- Display the selected file in the system explorer (for example, Windows Explorer in Windows). To do this, right-click on a file and select **Show in System Explorer**.

File Explorer preferences

To display hidden files and folders in the **File Explorer** view: in the **Preferences** dialog box, expand the **WPS** group, select **File Explorer** and in the **File Explorer** panel select **Show hidden files and folders**.

Managing files in the File Explorer

The **File Explorer** view can be used to manipulate the various objects in available local and remote file systems.

When you open files from the **File Explorer** view, the file is displayed in the most appropriate **Editor** view:

- Program files (those with `.wps` and `.sas` file extensions) are opened in the **SAS editor**.
- Text files (`.txt`) are opened in the **Text Editor**.
- XML files (`.xml`) are opened in the **Text Editor**.
- HTML files (`.html`) are opened in Workbench where possible (this is platform-dependent).

On some operating systems, the Workbench attempts to open the application associated with the file, for example PDF files, even if the file cannot be opened in Workbench itself.

Creating a new folder in a file system

Folders can be created wherever you have appropriate operating system privileges for the particular local or remote host.

1. Click the **Window** menu, click **Show View** and then click **File Explorer** to display the **File Explorer** view.
2. In the **File Explorer** view, right-click the required parent directory shortcut or Folder, click **New** and click **Folder** on the shortcut menu.
3. Enter a new **Folder name** and click **Finish**.

Creating a new file or program in a file system

Creating a new file or program in a file system using Workbench.

1. In the **File Explorer** view, right-click the Directory Shortcut or Folder, click **New** and then click **File** on the shortcut menu.

If you are creating a new SAS language program and are in the SAS Language Environment perspective, click **New** and then click **Program** on the shortcut menu.

If you are creating a new Workflow and are in the Workflow Environment perspective, click **New** and then click **Workflow** on the shortcut menu.

2. In the dialog box, enter a **Filename**, including file extension, and click **Finish**.

If you are creating a new SAS language program, the **Filename** has a default extension of `.sas`, but you can change this to `.wps` if you prefer.

Moving a file in a file system

Moving a file using the **File Explorer** view.

You can move files to other folders in your local and remote connections. To move objects in the **File Explorer** view:

1. Click the **Window** menu, click **Show View** and then click **File Explorer** to display the view.
2. In the **File Explorer** view, right-click the required object to move and click **Cut** on the shortcut menu.
3. Right-click the target directory shortcut or folder to which you want to move the object and click **Paste** on the shortcut menu.

Alternatively, you can select the object you want to move and drag the object to the required directory shortcut or Folder.

Renaming an object in a file system

Renaming an object using the **File Explorer** view.

To rename a file system object:

1. In the **File Explorer** view, right-click the required object to rename and click **Rename** on the shortcut menu.
2. In the **Rename File**, enter a new name for the file and click **OK**.

You may need to refresh the connection in the **Link Explorer** view to see the change.

Deleting an object from a file system

Deleting an object using the **File Explorer** view.

To delete an object from a file system, proceed as follows:

1. In the **File Explorer** view, Right-click on the required object to delete and click **Delete** on the shortcut menu.
2. In the **Confirm Delete** dialog box, click **OK** to confirm deletion.

Once you confirm the deletion of the object from the file system, the operation cannot be undone.

Properties

The **Properties** view displays the properties of a selected object.

Select an object in one of the following views to show the object details in the **Properties** view:

- **File Explorer** view
- **Link Explorer** view
- **Project Explorer** view
- **Search** view
- **WPS Server Explorer** view

Displaying the view

To display the **Properties** view

From the **Window** menu, click **Show View** and then click **Properties**.

Link Explorer and Workflow Link Explorer

The **Link Explorer** in the SAS Language Environment allows you to manage connections to WPS Analytics hosts and any WPS Servers on those connections. Similarly, the **Workflow Link Explorer** in

the Workflow Environment allows you to manage connections to WPS Analytics hosts and any WPS Engines on those connections.

The **Link Explorer** and **Workflow Link Explorer** views are the key views if you are using *WPS Link* – the collective term for the technology used to provide a client/server architecture. These views allow you to connect to a remote machine with a WPS server installed to run SAS language programs or Workflows. The output from the remote server can be viewed and manipulated in Workbench, in the same manner as a locally executed program or Workflow.

Displaying the Link Explorer view or Workflow Link Explorer view

To display the **Link Explorer** or **Workflow Link Explorer**: from the **Window** menu click **Show View** and then click **Link Explorer** (🔗) when in the SAS Language Environment perspective, or **Workflow Link Explorer** (🔗) when in the Workflow Environment perspective.

Objects displayed

There are two node types visible in this view:

1. A **Connection node** (🔗): this is the root node and represents a connection to a host machine. The connection can either be a local connection or a connection to a remote machine. The connection can contain one or more WPS server nodes. There can only be one local connection, but many remote connections.
2. A **Server node** (🔗), representing the Processing Engine installation where you can run your SAS language programs or Workflows.

Managing the connections and servers

If you right-click on an open connection, the following options are available:

- Open or close a local or remote host connection.
-
- Export a host connection (see [Exporting a host connection definition](#) (page 31)).
- Define a new WPS server (see [Defining a new Processing Engine](#) (page 32)).
- Import or export a WPS server definition (see [Importing a WPS server definition](#) (page 33) or [Exporting a WPS server definition](#) (page 33)).

Right-clicking on a WPS server gives you options to stop, start or restart that server (see [Restarting Processing Engines](#) (page 47)).

A WPS server can be moved or copied to a different host connection. A drag-and-drop operation moves the server from its current host connection to the target host connection. Press and hold the **Ctrl** key at the same time to copy the WPS server to the new host connection.

Right-clicking on an item and selecting **Properties** (if enabled), will display further information on that connection in the default **Remote Connection Options** tab. The **properties** window for servers also has a tab for **Directory Shortcuts**, allowing you to enter a full path to a host's filing system, which will then appear in the **File Explorer** window.

Note:

Within the properties window there is an option **Automatically open connection**. If you are going to be using the connection regularly, then you are strongly advised to select this option.

Commands (top right corner of the view)

The view contains the following commands

- **Collapse All** – collapse the view to show just the root nodes.
- **Create a new remote host connection** – add a new remote connection.
- **Import a host connection definition** –import a connection definition file. Host connections can be also imported by dragging Connection Definition Files (with the suffix *.cdx) onto the **Link Explorer**.

Log

All SAS language programs, and Workflow blocks that run SAS language code in the background, produce a log output that can be viewed in the Editor view. The log output is a record of events occurring during the execution of a SAS language program, along with information about the run environment.

In the SAS Language Environment, the log file contains a record of all activity for the current session. If multiple programs or Workflows are executed on the same server or engine during a single server session, the output for all programs is appended to the same log. If you have more than one WPS server registered in the Workbench, each WPS server creates to its own log file.

In the Workflow Environment, the log only displays log information for the block it was opened from.

There are system options that enable you to customise the log output. A description of these system options can be found in the WPS Reference for Language Elements.

Opening the log - SAS Language Environment

Opening a log in the SAS Language Environment for a particular WPS server.

1. Run a SAS language program.
2. From the **Window** menu click **Show View** and then click **Output Explorer**. **Output Explorer** may already be visible; if so, click on it to bring it to focus.
3. Under the WPS Server you ran the program on, double-click on **Log**.

If you have run several programs, log information for all programs is concatenated into a single log file. To help locate information, you can use the **Outline** view to navigate the log, see the Outline view [🔗](#) (page 83) for more information.

Opening the log - Workflow Environment

Opening a log in the Workflow Environment for a particular WPS server.

1. Run a Workflow.
2. Right-click on a Workflow block that runs SAS language code and click **Open Log**. Blocks that run code will have a logo with a coloured background.

The log only displays log information for the block it was opened from.

Saving the log

Saving the log as a text file.

1. Open the log. For more information, see *Opening the log - SAS Language Environment* [↗](#) (page 75) or (*Opening the log - Workflow Environment* [↗](#) (page 76).
2. Right-click on the log and select **Save As**.
A **Save As** window opens.
3. Choose a location for the file and click **Save**.

Printing the log

Printing the log.

1. Open the log. For more information, see *Opening the log - SAS Language Environment* [↗](#) (page 75) or (*Opening the log - Workflow Environment* [↗](#) (page 76).
2. Right-click on the log and select **Print**.
A **Print** window opens.
3. Choose your required print options, then click **Print**.

Finding an item in a log

Finding an item in a log by searching for a string of characters.

To find an item in a log:

1. Open the log. For more information, see *Opening the log - SAS Language Environment* [↗](#) (page 75) or (*Opening the log - Workflow Environment* [↗](#) (page 76).

2. Right-click on the log and select **Find**.
A **Find in Log** window opens.
3. Enter a string you wish to search for and then select the search **Direction**, and any additional **Options**.
4. Click **Find next** to start the search.

Clearing the log

Logs in the *SAS Language Environment* can be cleared. This does not apply to *Workflow Environment* logs.

To clear the log, right-click on the log and select **Clear log**.

Note:

To automatically clear the log on each run, see [Log preferences](#) (page 77).

Log preferences

Changing log preferences, including log syntax colouring and changes to the behaviour of the log.

To change log preferences:

1. Open the log. For more information, see [Opening the log - SAS Language Environment](#) (page 75)) or ([Opening the log - Workflow Environment](#) (page 76)).
2. Right-click on the log and select **Preferences**.
A **Preferences** window opens.
3. By default you are presented with the **Log syntax coloring** page, which allows you to set colours for each item within the log, along with whether the text appears in bold or not. For options concerning the behaviour of the log, select **WPS** in the left hand pane.
4. To save preferences and exit, click **OK**.

Autoscroll

Autoscroll automatically scrolls to the bottom of the log as new lines are appended. This feature can be enabled or disabled.

To change the Autoscroll status:

1. Open the log. For more information, see *Opening the log - SAS Language Environment* [↗](#) (page 75)) or (*Opening the log - Workflow Environment* [↗](#) (page 76)).
2. Right-click on the log.

If **Autoscroll** is enabled, a tick will be next to the option.

3. To change the Autoscroll status, click on it once.

SAS Language Environment views

An overview of views specific to the SAS Language Environment, together with a description of the main actions that are carried out within each view.

To open the SAS Language Environment perspective, click the **Window** menu, click **Perspective**, then **Open Perspective** and finally **SAS Language Environment**. A faster way to access the SAS Language Environment perspective is to click on the **SAS Language Environment** button at the top right of the screen: . This button can be made clearer by right-clicking on it and selecting **Show Text**; this displays the name of the button on the button itself:  **SAS Language Environment**

Editor

When you open any type of file in the Workbench, the file is opened as a tab in the **Editor** view.

The **Editor** view is the main interactive view for creating and editing SAS language programs or Workflows. The view is also used to display output such as result datasets or log information.

The **Editor** can be used to edit datasets opened from the **WPS Server Explorer** view, as well as files opened from the **Project Explorer** or **File Explorer** views. Logs and output files can be viewed but not edited.

The **Editor** view can be split to view or modify two or more files together. To split the view, from the **Window** menu, click **Editor** and then click **Toggle Split Editor (Vertical)**. The editor can also be split horizontally by using **Toggle Split Editor (Horizontal)**, and you can open another copy of the current file by using the **Clone** option.

WPS Server Explorer

The **WPS Server Explorer** view is used to display all *Filerrefs*, libraries, catalogs and datasets generated from running a SAS language program on a WPS server.

While the WPS server is running, the *Filerrefs* and library contents generated from previous SAS language programs are preserved. To remove these items, you need to restart the server.

Displaying the view

To display **WPS Server Explorer** view

From the **Window** menu, click **Show View** and then click **WPS Server Explorer**.

Objects displayed

The view contains a WPS Server node that represents the WPS server where your SAS language program has run. Under this node:

- **Libraries** – a parent node for individual library items.

The libraries node contains a library node called *Work*. This is the default temporary library used if one is not specified in a program.

- **Library** – a collection of catalogs and datasets.

If you use a `LIBNAME` statement in a SAS language program to specify your own library name, then you will see a library node with the specified name also listed in the view.

- **Catalog** – a collection of user-defined items such as formats and informats.
- **Dataset** – a data file containing numeric and/or character fields.
- **Dataset View** – a dataset created by invoking a view in the source data.
- **Numeric Dataset Field** – a dataset field that contains numeric values.
- **Character Dataset Field** – a dataset field that contains character values.
- **Filerefs** – The parent node for Fileref items.
- **Fileref** – an individual reference to a file.

Filerefs are normally generated automatically, but you can create a Fileref using the `FILENAME` statement, for example:

```
filename shops 'c:\Users\fred\Desktop\4.1-Shops.xls';
```

The link between a Fileref and its associated external file lasts for the duration of the current WPS session, or until you change it or discontinue it. To keep a Fileref from session to session, then save the `FILENAME` statement in the relevant program(s).

If you select an item in the **WPS Server Explorer** view, information about that item displayed in the **Properties** view.

WPS server properties

Properties and information available for a WPS server.

Right-click on a WPS server and then click **Properties** to see the following information about that server:

Code Submission

Sets the server working directory to the directory from which the program is opened.

Environment

A read-only list describing the server's environment (working directory, process ID and environment variables). This information is only available when the server is running.

Macro Variables

A list of the automatic and global macro variables used by the server, and the associated values. This information is only available when the server is running. Existing variables cannot be modified, but new variables can be created.

Startup

The main page has an option called **Start the server automatically on connection startup**. Ensure that this is selected to automatically start the server whenever the associated connection is launched. The **Initial current directory for server process** option specifies the working directory for the server, determining where the output is generated. If you have `%INCLUDE` statements in your program using relative paths, it is important that you start the server in the correct directory to allow those `%INCLUDE` statements to find the files.

The **Startup** page contains:

System Options

Enables you to specify any WPS options to be processed by the server on startup. Click **Add...** to specify the name and value of the option. You can then use **Edit...** and **Remove** to modify and delete any previously-created options and values.

Environment Variables

For a local WPS server only, used to specify any environment variables to be set before the server is started. Click **New...**, to specify the name and value of the variable. You can use **Edit...** and **Remove** to modify and delete previously-created environment variables.

System Options

When the WPS server is running, provides a read-only list of system options and their values currently set on the server. This is the same information as would be provided via the `OPTIONS` procedure.

WPS Licence Information

Displays the licence information for the server. The information cannot be modified and is only available when the server is running.

WPS Software Information

Displays details about the WPS software. The information cannot be modified and is only available when the server is running.

Bookmarks

The **Bookmarks** view lists the bookmarks added to any program or file in a project.

You can add bookmarks on any line in any program or file you can open from the **Project Explorer** view and edit with the **Editor** view.

The **Bookmarks** view lists bookmarks for all programs in all open projects. Bookmarks cannot be created for programs not managed through the **Project** view.

To open a file at the bookmark, double-click the bookmark in the **Bookmarks** view. The file is opened in the **Editor** view, with the bookmarked line highlighted, or at the first line in the file if the entire file was bookmarked

Displaying the view

To display the **Bookmarks** view:

From the **Window** menu, click **Show View** and then click **Bookmarks**.

Bookmark anchors

A bookmark is an anchor that you can specify to navigate back to that point at any time.

There is no limit to the number of bookmarks that you can create either against a particular line in the program, or a program file as a whole.

You can list, and navigate between, all your bookmarks using the **Bookmarks** view. If you export a project containing bookmarks, the individual bookmarks are not exported with it.

Adding a bookmark

A bookmark can be added to a particular line in a program.

To add a new bookmark on a particular line in a program:

1. Open the program or file in the **Editor** view.
2. Right-click in the grey border to the left of the required line and click **Add Bookmark** on the shortcut menu:
3. Enter a bookmark name in the **Add Bookmark** dialog box and click **OK**.

A bookmark tag () is displayed in the border, and a corresponding entry made in the **Bookmarks** view.

You can also add a bookmark to a whole file: select the file in the **Project Explorer** view, click **Edit** menu and then click **Add Bookmark**.

Deleting a bookmark

Deleting a bookmark from a program.

To delete a bookmark:

1. Click the **Window** menu, click **Show View** and then click **Bookmarks**.
2. In the **Bookmarks** view, right-click the bookmark's description and click **Delete** on the shortcut menu.

Tasks

The **Tasks** view lists any tasks added to any program or file in a project.

You can add tasks on any line in any program or file you can open from the **Project Explorer** view, and edit with the **Editor** view.

The **Tasks** view lists tasks associated with all programs in all open projects. Tasks cannot be created for programs not managed through the **Project** view.

You can use the **Tasks** view to change the priority levels, modify descriptions, and mark them as completed or leave them uncompleted.

You can add *Task markers* [↗](#) (page 82) (reminders) against any line in a program that has been opened from the *Project Explorer* [↗](#) (page 52), and associate each of them with notes and a priority level. To list, update and navigate between any tasks you have added, you need to use the  **Tasks** view.

To open a file associated with a task, double-click the task in the **Tasks** view. The file is opened in the **Editor** view, with the relevant line highlighted.

Displaying the view

To display the **Tasks** view:

From the **Window** menu, click **Show View** and then click **Tasks**.

Task markers

A task marker represents a reminder about a work item.

Task markers can only be added to programs in a project. You can add a task and associate a short description and a priority level (one of High, Normal or Low) against any line in a program. You can list and navigate between all your tasks in the **Tasks** view.

There is no limit to the number of task markers that you can add to a program. However, if you export a project, the individual tasks are not exported with it.

Adding a task

Adding a new task marker on a particular line in a program.

To add a task:

1. Open the program or file in the **Editor** view.
2. Right-click in the grey border to the left of the required line and click **Add Task** on the shortcut menu.

A **Properties** window appears, quoting the program name and location and the line number you have selected for the task.

3. In the **Properties** dialog, enter a **Description** and select the **Priority** for the task.
4. Click **OK** to save the task

A task marker () is displayed in the border, and a corresponding entry appears in the **Tasks** view.

Deleting a task

Deleting a new task marker from a particular line in a program.

To delete a task:

1. Click the **Window** menu, click **Show View** and then click **Tasks**.
2. Right-click on the task's description, and click **Delete** on the shortcut menu.

Outline

The **Outline** view displays structural elements for a file that can be used to locate the line associated with the element.

Opening or giving focus to a program, log, listing or HTML result file, automatically refreshes the **Outline** view to display the structural elements that are relevant to the active item.

Displaying the view

To display the **Outline** view:

From the **Window** menu, click **Show View** and then click **Outline**.

Displayed structural elements

Viewing programs displays the following structural elements in the **Outline** view:

-  Global statement

-  DATA step
-  Procedure
-  Macro

Viewing the log file displays the following structural elements in the **Outline** view:

-  An individual log entry.
-  Error
-  Warning

Viewing the output results from program execution displays the following structural elements in the **Outline** view:

-  Results parent node
-  HTML tabular output
-  Listing tabular output

You can control which structural elements are displayed in the **Outline** view using the toolbar at the top of the view panel.

Output Explorer

The **Output Explorer** view displays the output files generated from running a SAS language program.

Output items are associated with the server on which they were generated, and may be one of the following types:

- Listing output
- HTML output
- PDF output

Log output is always shown for each server.

To open any output, double-click on it.

Displaying the view

To display the **Output Explorer** view:

Once you have run a SAS language program, from the **Window** menu click **Show View** and then click **Output Explorer**.

View output

Double-click on one of the output nodes to open the output type in its associated editor. For output types other than log output the individual result elements are listed in the **Results Explorer** view when the output is opened.

Results Explorer

The **Results Explorer** view displays the results from every SAS language program that has been executed.

The view contains hierarchical links to the results, grouped by server, since the last time that the server was started.

Displaying the view

To display the **Results Explorer** view: from the **Window** menu, click **Show View** and then click **Results Explorer**.

Output

Once a program has been executed, an entry appears under the relevant server for every procedure that created output. For example `PROC PRINT` will produce an entry named *The PRINT Procedure*. Expand this link to find each type of output that has been generated.

Console

The **Console** view displays all system standard output and system standard error messages produced by a WPS server.

The **Console** view is automatically opened when a system error is reported.

Displaying the view

To display the **Console** view:

From the **Window** menu, click **Show View** and then click **Console**.

Search

The **Search** view displays the results of searches carried out using the **Search** dialog box.

When you click an item in the **Results** view, the file is opened in the **Editor** view, with the relevant line highlighted.

To open the **Search** dialog box, click the **Search** menu and then click **Search**. The **Search** view opens automatically to display the results.

Using Hub Authentication Domains and Library Definitions with the SAS Language Environment

Hub is an Enterprise Management Tool that, amongst other things, provides centralised management of access to data sources.

Hub provides two features for use with the SAS Language Environment:

- *Authentication Domains* securely store access credentials for data sources within Hub, which can then be accessed from SAS language code by name, removing the need to hard code credentials in SAS Language code. This improves security and also eliminates multiple code changes should the credentials change.
- *Library Definitions* securely store details of libraries and their options within Hub. These library definitions can then be accessed from SAS language code using Hub user specific Libname Bindings, which are keywords defined in Hub. These Libname Bindings can take the place of Libname statements in SAS language code.

To use either of these features, the SAS language code will need to access Hub when it runs, which can be done in two ways:

- The SAS language code is run from within Workbench, with Workbench logged in to the Hub instance hosting the Authentication Domains or Library Definitions you require.
- The code includes an `OPTIONS` statement, specifying Hub connection details. For example:

```
OPTIONS hub_user="HubUser123" hub_pwd="pineapple" hub_port=8443  
hub_server="https://1cen7x64d01.abcde.local";
```

Logging in to Hub from Workbench

To use Hub features in Workbench, you will need to log into a Hub installation from Workbench.

Before connecting to Hub, you require details of a Hub host and access credentials for it. These details will be provided by your Hub administrator.

To log in to a Hub instance from Workbench:

1. On the **WPS** drop-down menu in Workbench, click **Hub**, then **Log in**.
2. In the **Hub Login** dialog box complete the required information:
 - a. Select the required **Protocol**, either `HTTP` or `HTTPS`.
 - b. Enter the **Host** name provided by your Hub administrator.
 - c. Enter the **Port** for Hub. By default, Hub configured for `HTTPS` will use port `8443`, and when configured for `HTTP`, port `8181`, but your Hub administrator may change these.
 - d. Enter the Hub **Username** provided by your Hub administrator.
 - e. Enter the Hub **Password** provided by your Hub administrator.
3. Click **OK** to connect.

If the connection is successful, a message stating `Hub: Logged in as username` is displayed in the Workbench status bar. If the connection fails, an error message is displayed in the **Hub Login** dialog box.

Logging in to Hub from SAS language code

To use Hub features in SAS language code, you will need to include Hub login details in an `OPTIONS` statement.

To connect a piece of SAS language code with Hub, the code needs to include an `OPTIONS` statement specifying Hub connection details.

For example:

```
OPTIONS hub_user="HubUser123" hub_pwd="pineapple" hub_port=8443 hub_server="https://  
lcn7x64d01.abcd.local";
```

Hub Authentication Domains

Authentication Domains securely store access credentials for data sources within Hub, which can then be accessed from SAS language code by name, removing the need to hard code credentials in SAS Language code. This improves security and also eliminates multiple code changes should the credentials change.

To use Authentication Domains, SAS language code must either be run from an instance of Workbench connected to Hub (see [Logging in to Hub from Workbench](#) (page 86)), or the code itself must log in to Hub (see [Logging in to Hub from SAS language code](#) (page 87)).

SAS Language statements that require credentials, for example `LIBNAME`, will also accept an `AUTHDOMAIN` option where you can specify the name of an Authentication Domain present in a connected Hub instance.

For example, the following libname statement connects to an Oracle database using an Authentication Domain contained in a connected Hub instance:

```
libname      L33 oracle
             path='oracle-ab-c12'
             authdomain="OraAB-AD"
             schema='TEST';
```

In the above example, the Authentication Domain OraAB-AD is stored within the connected Hub instance and contains credentials to connect to the oracle-ab-c12 database.

Hub Library Definitions in the SLE

Library Definitions securely store details of libraries and their options within Hub. These library definitions can then be accessed from SAS language code using Hub user specific Libname Bindings, which are keywords defined in Hub. These Libname Bindings can take the place of Libname statements in SAS language code.

To use Library Definitions, SAS language code must either be run from Workbench connected to an instance of Hub (see [Logging in to Hub from Workbench](#) (page 86)), or the code itself must log in to Hub (see [Logging in to Hub from SAS language code](#) (page 87)).

In Workbench, Libname Bindings from a connected Hub instance are displayed as Libraries in **WPS Server Explorer** and may therefore be quoted in SAS language code statements. Libname Bindings are Hub user specific, so the Libname Bindings displayed in Workbench depend on the Hub user that you are logged in as.

Although Library Definition libraries are defined in Hub, the actual library call is still being made by the user logged in to Workbench, so that user will need appropriate software installed to access the library in question.

Workflow Environment views

The Workflow Environment perspective contains the views and tools required to create and manage Workflows. To open the Workflow Environment perspective, click on the **Workflow Environment** button at the top right of the screen: .

Workflow files are created and managed in projects using the **Project Explorer** view, or on a host to which you have access using the **File Explorer** view.

The **Workflow Editor** view is used to create and run Workflows. The **Data Profiler** view is used to inspect the contents of a dataset in the Workflow.

The Workflow Link Explorer and **File Explorer** view enable you to connect to a remote host and specify a default Workflow Engine on a remote host to run your Workflows.

The Workflow perspective also enables you to create and view tasks and bookmarks in a Workflow.

The SAS language library feature for creating datasets, such as the temporary `WORK` library, is not available in a Workflow. *Working datasets* are created as part of a Workflow, and the **Data Profiler** view is used to explore the dataset in the Workflow.

To open the Workflow Environment perspective, click the **Window** menu, click **Perspective**, then **Open Perspective** and finally **Workflow Environment**. A faster way to access the Workflow Environment perspective is to click on the **Workflow Environment** button at the top right of the screen: . This button can be made clearer by right-clicking on it and selecting **Show Text**; this displays the name of the button on the button itself: .

Note:

Some views and features from the SAS Language Environment are visible when viewing Workflows in the Workflow Environment, even though they don't function for Workflows (for example, the **Search** facility).

Editor

When you open any type of file in the Workbench, the file is opened as a tab in the **Editor** view.

The **Editor** view is the main interactive view for creating and editing SAS language programs or Workflows. The view is also used to display output such as result datasets or log information.

The **Editor** can be used to edit datasets opened from the **WPS Server Explorer** view, as well as files opened from the **Project Explorer** or **File Explorer** views. Logs and output files can be viewed but not edited.

The **Editor** view can be split to view or modify two or more files together. To split the view, from the **Window** menu, click **Editor** and then click **Toggle Split Editor (Vertical)**. The editor can also be split horizontally by using **Toggle Split Editor (Horizontal)**, and you can open another copy of the current file by using the **Clone** option.

Workflow tab

The **Workflow** tab in the **Workflow Editor** view contains the canvas on which a Workflow is created. This tab is displayed by default when opening a Workflow file.

To display the **Workflow** tab, ensure that you are in the Workflow Environment and displaying a Workflow file, then click **Workflow** at the base of the screen.

The **Workflow** tab has a palette of operations, such as data import, data preparation, machine learning, coding, scoring, and so on. Each operation is represented by a block. All blocks are available in a palette on the left hand side of the Workflow tab. To select the required operation, drag the corresponding block onto the canvas, where you can connect it with other blocks already on the canvas to create a Workflow.

Most types of blocks that can be added to a Workflow require you to specify properties. Properties are set in the configuration dialog box that can be accessed through the shortcut menu for a block. To view or specify block properties, right-click the required block in the canvas and click **Configure**.

By default, the Workflow automatically runs as blocks are added or edited. You can also right-click on a block and choose to run the Workflow up to that block. Working datasets created as part of the Workflow are deleted when the **Workflow Editor** view is closed. If you automatically run a Workflow, reopening a saved Workflow automatically recreates all working datasets in the Workflow.

Adding comments to a Workflow and blocks

You can add comments to a Workflow, describing the purpose of the Workflow or identifying the individual steps in a Workflow when the Workflow is shared.

- To create a new comment for the Workflow, right-click in the **Workflow Editor** view and click **Add comment**.
- To add a comment to a block, right-click the required block and click **Add comment**.

Workflow navigation

The **Workflow Editor** view enables you to zoom in and out of a Workflow, to relocate blocks in a Workflow and move the Workflow within the editor view.

Zoom in and zoom out features in the **Workflow Editor** view can be controlled from the **Zoom** tools, keyboard, or mouse. The zoom tools are grouped in the Workbench toolbar:



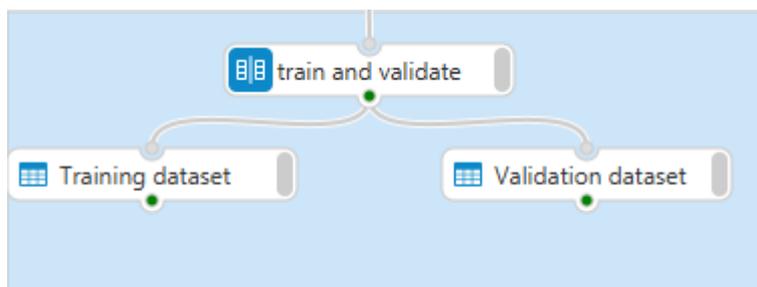
To change the scale of the Workflow view:

- Click **Zoom in** () to increase the scale of the Workflow view by 10% of the current scale. Alternatively, press **Ctrl+Plus Sign** to do the same.
- Click **Zoom out** () to decrease the scale of the Workflow view by 10% of the current scale. Alternatively, press **Ctrl+ Minus Sign** to do the same.
- Click the zoom list () and then select **Page** to scale the Workflow view to fit the available screen area in the **Workflow Editor**.
- Press **Ctrl+0** (zero) to set the scale of the Workflow view to 100%.
- Click the zoom list () and then select **Height** to scale the Workflow view to fit the height in the available screen area in the **Workflow Editor**. If the Workflow design is too wide for the scale, some blocks may be partly-visible or not visible at all after scaling.
- Click the zoom list () and then select **Width** to scale the Workflow view to fit the width in the available screen area in the **Workflow Editor** view. If the Workflow design is too tall for the scale, some blocks may be partly-visible or not visible at all after scaling.

Moving Workflow blocks in the view

The position of multiple blocks can be changed at the same time in the Workflow view. If blocks are connected to other blocks in the Workflow, the connections are maintained between the blocks. To relocate multiple blocks in the Workflow:

1. Click the **Workflow Editor** view.
2. Press **Ctrl** and drag the pointer to cover all the blocks:



The blocks become highlighted (their outlines become blue).

3. Click one of the highlighted blocks, and drag the group of blocks to the required place in the Workflow.

Move the visible area of the Workflow

You can move the Workflow canvas view to enable you to focus on a specific part of the Workflow. To move the view, click on a blank area and drag until the required area of the Workflow canvas is in view.

Saving a Workflow

After modifying a Workflow, you should save it to ensure no loss of data. A Workflow can be saved in a project in the active workspace, or to a location on a local or remote host.

Saving a Workflow to a different file

You can save a Workflow to an alternative file either in the current workspace or to a folder on any host to which you have access. To save a Workflow to an alternative file, click the **File** menu and then select **Save As**.

- If you use the **Project Explorer** view to manage files, you can save the Workflow as a different file in any project in your current workspace, but cannot save a Workflow to a folder that can only be accessed through the **Project Explorer** view.
- If you use the **File Explorer** view to manage files, you can save the Workflow as a different file to any available folder structure that you can access on each host, but cannot save a Workflow to a project in the current workspace.

Updating a saved Workflow

To save a Workflow, click the **File** menu and then click **Save**. The Workflow file is updated with any changes you have made.

Exporting to SAS language program

The **Workflow Editor** view allows you to export a Workflow as SAS language code, either to the clipboard or to a specified file.

Note:

To export a Workflow to SAS language code, the Workflow must not contain any failed or out of date blocks.

Note:

Model Training blocks cannot be exported to SAS language procedures.

To export a Workflow as SAS language code:

1. Ensure that your Workflow is displayed and runs successfully.
2. Right-click on the Workflow background and then click **Export to SAS Language Program**.

An **Export Workflow to SAS Language Program** dialog box appears.

3. In **Temporary library name**, enter a name for the library that will be defined at the start of the SAS language code.
4. In **Directory for temporary datasets**, enter a directory for the library to be defined at the start of the SAS language code.
5. Click **Next**.

SAS language code for the Workflow is displayed.

6. You can now either copy the code to the clipboard or save it to a file:
 - To copy the code to the clipboard, click **Copy Code**. You can now close the dialog box by clicking **Cancel**.
 - To save the code to a file, click **Next**, choose either your **Workspace** or an **External** location, and then click **Browse** to choose the location and type a filename, including the file extension (.sas or .wps).
7. Click **Finish**.

Workflow Settings tab

The **Workflow Editor** view **Settings** tab contains tools for viewing and managing a Workflow's database references and parameters.

To display the **Settings** tab, ensure that you are in the Workflow Environment and displaying a Workflow file, then click **Settings** at the base of the screen.

For more information on database references, see *Database References* [↗](#) (page 165).

For more information on parameters, see *Parameters* [↗](#) (page 173).

Database Explorer view

The **Database Explorer** view enables you to create a reference to a database and to access references created and stored in Hub.

A *reference* specifies the details required to access a database. If you have multiple Workflows that require access to the same data, you can create a library reference and import data into a Workflow using the **Database Import** block.

Displaying the view

To display the **Database Explorer** view:

1. Click the **Window** menu option, select **Show View** and then click **Database view**.

Objects displayed

There are two node types visible in this view, a host node (1) and a data source node (2):



A *host node* represents the location where the data source connection information is stored, either **Hub** or **Workbench**.

- The **Hub** node displays all references available in the authorisation domain in WPS Analytics to which you have connected. For more information, see *Using Hub Library Definitions with the Workflow Environment* [↗](#) (page 111).
- The **Workbench** node contains all references you have created or imported. These references are stored in the current project, but can be shared with a Workflow through the Workflow **Settings** view.

A *database node* represents the reference element specifying a connection to a specific database. Previously-created references can be imported from a data source definition file. The settings for any node created in the **Workbench** host can be exported to a database definition file.

Adding a new database reference for Workbench

To add a new reference to a database, click the **Create a new database** button () at the top right of the **Database Explorer** view. The **Add Database** wizard starts. For guidance on completing the wizard, see *Database References* [↗](#) (page 165).

Importing a database reference for Workbench

An existing database reference can be added to the **Database Explorer** view if that reference has previously been saved as a database definition. To add an existing reference you import the database definition:

1. Click the **Import database definitions** button () at the top right of the **Database Explorer** view.

An **Open** dialog is displayed.

2. Browse to the folder that contains the database definition.

A database definition has the file extension `.dsx`.

3. Select the definition.

The **Open** dialog is closed, and a database reference is added to the list of references under the **Workbench** host. The name of the reference is extracted from the definition. If the name is the same as a name already in the list of references, the name is made unique by appending a number.

Renaming a Workbench database reference

A database reference can be renamed. To do this:

1. Right-click the reference you want to rename.
2. In the pop-up menu, click **Rename**.

The **Enter new database name** dialog is displayed.

3. Enter a new name in the text box.
4. Click **OK**.

Click **Cancel** to close the dialog box and keep the current name.

Exporting an existing Workbench database reference

An existing database reference can be added to the **Database Explorer** view if that reference has previously been saved as a database definition. To add an existing reference you import the database definition:

1. Click the **Import database definitions** button () at the top right of the **Database Explorer** view.

A **Save As** dialog is displayed.

2. Browse to the folder where you want to save contains the database definition.

3. Enter a name for the definition in **File name**.

A database definition has the file extension `.dsx`.

4. Click **Save**.

The **Save As** dialog is closed.

Deleting a Workbench database reference

An existing database reference can be deleted from the **Database Explorer** view. To delete a database reference:

1. Right-click the reference you want to delete.
2. In the pop-up menu, click **Delete**.

The **Confirm Remove** dialog is displayed.

3. Click **OK** to delete the reference, or **Cancel** to keep the reference and close the dialog box.

Testing a database connection

You can test whether an existing database reference can connect to a database. You can do this for databases defined under the **Workbench** or **Hub** nodes. To do this:

1. Right-click the reference you want to test.
2. In the pop-up menu, click **Test**.

The **Data Connection Test** dialog is displayed. This tells you if a connection was successfully made to the database.

3. Click **OK** to close the dialog box.

If you cannot connect to the database, check whether it is running, or whether connection details have changed since you created the reference.

Adding a database or table to a Workflow by dragging

You can drag a database name or table name from the **Database Explorer** view onto the Workflow canvas.

If you drag a database name and drop it on the canvas, a Database Import block is created. You can then double click on the block to open the **Database Import** configuration dialog to select the required tables.

If you drag a table name and drop it on the canvas, a Table block is created. This is a pointer to the table on the database server, and does not copy data into memory until you connect one or more blocks and run the Workflow, or view the contents of the table using the Dataset File Viewer.

Data Profiler view

The **Data Profiler** view enables you to view content and details about a dataset.

To open the **Data Profiler** view, select the required dataset, right-click **Open With** and then click **Data Profiler** in the shortcut menu.

Summary View ↗	96
The Summary View lists information about the dataset, including the number of observations and variables in the dataset as well as variable characteristics.	
Data ↗	98
The Data panel lists all observations in a dataset and enables you to view, filter, or sort those observations.	
Univariate View ↗	98
The Univariate view tab displays summary statistics for all numeric univariate variables.	
Univariate Charts ↗	98
The Univariate Charts tab enables you to view the frequency distribution table and graphs for a selected dataset variable.	
Correlation Analysis ↗	99
The Correlation Analysis tab enables you to view the strength of the correlation between numeric variables in the dataset.	
Predictive Power ↗	101
The Predictive Power tab enables you to view the predictive power of independent variables in the dataset in relation to a selected dependent variable.	

Summary View

The **Summary View** lists information about the dataset, including the number of observations and variables in the dataset as well as variable characteristics.

The **Summary View** contains two panels: **Summary** and **Variables**.

Summary

The **Summary** panel displays summary information about the dataset: the dataset name, the total number of observations, and the total number of variables in the dataset.

Variables

The **Variables** panel displays information about the variables in the dataset as follows:

Variable

The name of the variable in the dataset.

Label

The alternative display name for the variable.

Type

The type of the variable. This can be one of:

- *Numeric*, for numbers and date and time values.
- *Character* for character and string data.

Classification

The category of the variable; this can be one of:

- *Categorical*. A non-numeric variable that can contain a limited number of possible values, and the limit is below the specified classification threshold.
- *Continuous*. A numeric variable that can contain an unlimited number of possible values. This classification is used for numeric variables where the number of distinct values in the variable is greater than the specified classification threshold.
- *Discrete*. A numeric variable that can contain a limited number of possible values. This classification is used for character variables where the number of distinct values in the variable is less than or equal to the specified classification threshold.

The classification threshold defines a number of unique values above which a variable is classified as continuous, and below which a variable is either categorical (if non-numeric) or discrete (if numeric). The classification threshold is specified on the **Data** panel of the **Preferences** dialog box.

Length

The size required to store the variable values. For character types, the number represents the maximum number of characters found in a variable value. For numeric types, the number represents the maximum number of bytes required to store the value.

Format

How the variable is displayed when output. For more information about formats see the section *Formats* in the *WPS Reference for Language Elements*.

Informat

The formatting applied to the variable when imported into WPS Analytics. For more information, about informat, see the section *Informats* in the *WPS Reference for Language Elements*.

Distinct Values

The number of unique values in the variable. If the variable classification is Continuous, the display indicates there are more unique values than the specified classification threshold.

Missing Values

The number of missing values in the variable.

Frequency Distribution

For each value in a non-continuous variable, displays a chart showing the number of occurrences of each value in the variable.

Data

The **Data** panel lists all observations in a dataset and enables you to view, filter, or sort those observations.

The **Data** panel in the **Data Profiler** view is a read-only version of the **Dataset Viewer**. You can modify the view, or sort and filter data in the **Data** panel. If you want to edit values in the dataset, you need to use the **Dataset Viewer**. For more information about sorting and filtering observations, and about the **Dataset Viewer** see *Dataset Viewer* [↗](#) (page 105).

Univariate View

The **Univariate view** tab displays summary statistics for all numeric univariate variables.

The columns displayed in the **Univariate View** tab are determined using the **Calculate Statistics** dialog box. Click **Configure Statistics**  to open **Preferences** and select the statistics you want displayed:

Quantiles

Lists quantile points.

Variable Structure

Lists statistics describing the variable, such as number of missing values, and minimum and maximum values.

Measures of Central Tendency

Lists measures used to identify the central tendency in the values of the variable (mean, median and mode).

Measures of Dispersion

Lists measures used to show the dispersion of a variable.

Others

Lists other statistics that can be displayed in the statistics table.

Univariate Charts

The **Univariate Charts** tab enables you to view the frequency distribution table and graphs for a selected dataset variable.

The information displayed is determined by the variable selected in the **Variable Selection** list. Each section in the tab displays the frequency of values occurring in the selected variable.

Frequency Table

Displays the frequencies of the values in the specified variable, the percentage of the total number of observations for each value, and cumulative frequencies and percentages.

If the variable type is categorical or discrete, the table displays one row for each value. If the variable type is continuous, the variable is binned and frequency information is displayed for each bin.

Frequency Chart

Displays the frequency of values as the percentage of the total number of observations for the variable, either as a histogram, line chart or pie chart.

The chart can be edited and saved. Click **Edit chart**  to open the Chart Editor from where the chart can be saved to the clipboard.

Correlation Analysis

The **Correlation Analysis** tab enables you to view the strength of the correlation between numeric variables in the dataset.

Options

Specifies the coefficient type to use when comparing variables, and which numeric variables in the dataset are to be compared.

Coefficient

Specifies the type of coefficient used to compare values.

Pearson

Specifies the Pearson correlation coefficient, which is defined as:

$$r_{x,y} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

where:

- $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ is the mean of variable x
- $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ is the mean of variable y

Spearman's Rho

Specifies the Spearman's rank correlation coefficient, which is defined as:

$$r_s = \frac{\sum_{i=1}^n (R_{x_i} - \bar{R}_x)(R_{y_i} - \bar{R}_y)}{\sqrt{\sum_{i=1}^n (R_{x_i} - \bar{R}_x)^2 \sum_{i=1}^n (R_{y_i} - \bar{R}_y)^2}}$$

where:

- R_{x_i} is the rank of x_i
- $\bar{R}_x = \frac{1}{n} \sum_{i=1}^n R_{x_i}$ is the mean of the ranked variable R_x
- R_{y_i} is the rank of y_i
- $\bar{R}_y = \frac{1}{n} \sum_{i=1}^n R_{y_i}$ is the mean of the ranked variable R_y

The values of the variables are first ranked and the ranks compared. Using this coefficient may therefore be more robust to outliers in the data.

Kendall's Tau

Specifies the Kendall rank correlation coefficient, which is defined as:

$$\tau_b = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}}$$

where:

- $n_0 = \frac{n(n-1)}{2}$
- $n_1 = \sum_{i=1}^n \frac{t_i(t_i-1)}{2}$
- $n_2 = \sum_{j=1}^n \frac{u_j(u_j-1)}{2}$
- n_c is the number of concordant pairs
- n_d is the number of discordant pairs
- t_i is the number of tied values in the i th group of tied values for the first variable
- u_j is the number of tied values in the j th group of tied values for the second variable
- The difference in the number of concordant and discordant pairs can be expressed as:

$$n_c - n_d = \sum_{i < j} \text{sign}(x_i - x_j) \text{sign}(y_i - y_j)$$

The values of the variables are first ranked and the ranks compared. Using this coefficient may therefore be more robust to outliers in the data.

Select Variables

Displays a list of numeric variables in the dataset, from which you can select the variables to compare.

Correlation Coefficient Matrix

The matrix is shown in coloured blocks, with the size and colour of the blocks indicating the relationship between the compared variables. The scale for the items displayed ranges from 1 a strong positive correlation to -1 where there is a strong negative correlation.

Correlation Statistics

Displays the correlation statistics and scatter plot for the block selected in the Correlation Coefficient Matrix. The table displays the variables being compared, the coefficient type and the p-value for the correlation coefficient being non-zero. The scatter plot displays all values in the dataset, using the same colour scale as the matrix, as either a plot or heat map if the number observations is very high.

Predictive Power

The **Predictive Power** tab enables you to view the predictive power of independent variables in the dataset in relation to a selected dependent variable.

The information displayed is determined by the variable selected as the **Dependent Variable**. Each section displays the relationship between the independent variables in the dataset and the specified **Dependent Variable**.

Statistics Table

The **Statistics Table** displays all the variables in the dataset and a series of predictive power statistics. The statistics displayed are:

- **Entropy Variance.** Displays the *Entropy Variance* value for each variable in relation to the specified **Dependent Variable**.
- **Chi Sq.** Displays the *Chi-Squared* value for each variable in relation to the specified **Dependent Variable**.
- **Gini.** Displays the *Gini Variance* value for each variable in relation to the specified **Dependent Variable**.

For more information about how these values are calculated, see [Predictive power criteria](#) (page 102).

Entropy Variance Chart

Displays the bar chart of independent variables' entropy variances in relation to the specified **Dependent Variable**. The variables are displayed in order from the highest entropy variance to the lowest. The number of variables displayed in the chart is determined by the preferences set in the **Data Profiler** panel of the **Preferences** dialog box.

The chart can be edited and saved. Click **Edit chart**  to open the Chart Editor, from where the chart can be saved to the clipboard.

Frequency Chart

Displays the frequency of each value of the independent variable selected in the **Statistics Table** for each value of the specified Dependent Variable.

- Click **View whole data**  to display the overall predictive relationship between values of an independent variable selected in the **Statistics Table** and the specified **Dependent Variable**.
- Click **View breakdown data**  to display the frequency of each value of an independent variable, and its relationship to the outcomes for the specified **Dependent Variable**.

The chart can be edited and saved. Click **Edit chart**  to open a the Chart Editor from where the chart can be saved to the clipboard.

Predictive power criteria

Predictive power is a way of measuring how well a particular input variable can predict the target variable.

Pearson's chi-squared statistic

Pearson's chi-squared statistic is a measure of the likelihood that the value of the target variable is related to the value of the predictor variable.

Each observation in the dataset is allocated to a cell in a contingency table, according to the values of the predictor and target variables. Pearson's chi-squared statistic is calculated as the normalised sum of the squared deviations between the actual number of observations in each cell, and the expected number of observations in each cell if there were no relationship between the predictor and target variables.

If a predictor variable has a high Pearson's chi-squared statistic, it means that the variable is a good predictor of the target variable, and is likely to be a good candidate to use to split the data in a binning or tree-building algorithm.

Pearson's chi-squared statistic for a discrete target variable is calculated as

$$\chi^2 = \sum_{i=1}^r \sum_{j=1}^c \frac{(n_{ij} - \mu_{ij})^2}{\mu_{ij}}$$

$$\mu_{ij} = \frac{n_{i*}n_{*j}}{N}$$

where

- r is the number of distinct values of the predictor variable X (these are the rows in the contingency table)
- c is the number of distinct, discrete values of the target variable Y (these are the columns in the contingency table)
- n_{ij} is the number of observations for which the predictor variable X has the i th value, X_i , and the target variable Y has the j th value, Y_j (these are the values in the cells of the contingency table)
- μ_{ij} is the expected value of n_{ij}
- n_{i*} is the total number of observations for which the predictor variable X has the i th value, X_i
- n_{*j} is the total number of observations for which the target variable Y has the j th value, Y_j
- N is the total number of observations in the dataset

Entropy variance

Entropy variance is a measure of how well the value of a predictor variable can predict the value of the target variable.

If a variable in a dataset has a high entropy variance, it means that the variable is a good predictor of the target variable, and is likely to be a good candidate to use to split the data in a binning or tree-building algorithm.

Entropy variance for a discrete target variable is calculated as

$$E_r = 1 - \frac{\sum_{i=1}^r \left(\frac{n_{i*}E_i}{N} \right)}{E}$$

$$E_i = -\frac{1}{\log(c)} \sum_{j=1}^c \frac{n_{ij}}{n_{i*}} \log\left(\frac{n_{ij}}{n_{i*}}\right)$$

$$E = -\frac{1}{\log(c)} \sum_{j=1}^c \frac{n_{*j}}{N} \log\left(\frac{n_{*j}}{N}\right)$$

where

- r is the number of distinct values of the predictor variable, X
- n_{i*} is the total number of observations for which the predictor variable X has the i th value, X_i

- E_i is the entropy calculated for just the observations where the predictor variable is X_i
- N is the total number of observations in the dataset
- E is the entropy calculated for all the observations
- c is the number of distinct, discrete values of the target variable, Y
- n_{ij} is the number of observations for which the predictor variable X has the i th value, X_i , and the target variable Y has the j th value, Y_j
- n_{*j} is the total number of observations for which the target variable Y has the j th value, Y_j

Gini variance

Gini variance is a measure of how well the value of a predictor variable can predict the target variable.

If a variable in a dataset has a high Gini variance, it means that the variable is a good predictor of the target variable, and is likely to be a good candidate to use to split the data in a binning or tree-building algorithm.

Gini variance for a discrete target variable is calculated as

$$G_r = 1 - \frac{\sum_{i=1}^r \left(\frac{n_{i*} G_i}{N} \right)}{G}$$

$$G_i = 1 - \frac{\sum_{j=1}^c n_{ij}^2}{n_{i*}^2}$$

$$G = 1 - \frac{\sum_{j=1}^c n_{*j}^2}{N^2}$$

where

- r is the number of distinct values of the predictor variable, X
- n_{i*} is the total number of observations for which the predictor variable X has the i th value, X_i
- G_i is the Gini impurity calculated for just the observations where the predictor variable is X_i
- N is the total number of observations in the dataset
- G is the Gini impurity calculated for all the observations
- c is the number of distinct, discrete values of the target variable, Y
- n_{ij} is the number of observations for which the predictor variable X has the i th value, X_i , and the target variable Y has the j th value, Y_j
- n_{*j} is the total number of observations for which the target variable Y has the j th value, Y_j

Information value

Information value is a measure of the likelihood that the value of the target variable is related to the value of the predictor variable. The information value measure is only applicable for binary target variables (that is, target variables that can take one of exactly two values).

If a predictor variable has a high information value, it means that the variable is a good predictor of the target variable, and is likely to be a good candidate to use to split the data in a binning or tree-building algorithm.

The information value statistic is calculated as

$$IV = \sum_{i=1}^r \left(\frac{n_{i0}}{n_{*0}} - \frac{n_{i1}}{n_{*1}} \right) WOE_i$$

$$WOE_i = \ln \left(\frac{n_{i0}}{n_{*0}} + \alpha \right) - \ln \left(\frac{n_{i1}}{n_{*1}} + \alpha \right)$$

where:

- r is the number of distinct, discrete values of the predictor variable X (these are the rows in the contingency table)
- n_{i0} is the number of observations for which the predictor variable X has the i th value, X_i , and the target variable Y has the value, Y_0 (these are the values in the cells of the Y_0 column in the contingency table)
- n_{i1} is the number of observations for which the predictor variable X has the i th value, X_i , and the target variable Y has the value, Y_1 (these are the values in the cells of the Y_1 column in the contingency table)
- Y_0 and Y_1 are the two possible values of the binary target variable Y
- n_{*0} is the total number of observations for which the target variable Y has the value, Y_0
- n_{*1} is the total number of observations for which the target variable Y has the value, Y_1
- WOE_i is the WOE value for observations where the predictor variable is X_i
- $\alpha \ll 1$ is the weight of evidence (WOE) adjustment, a small positive number to avoid infinite values when $n_{i0} = 0$ or $n_{i1} = 0$

Dataset File Viewer

Enables you to view, filter, sort or modify observations in a dataset.

The contents of a dataset are shown in a grid. The rows of the grid represent dataset observations, and the columns represent dataset variables.

You can display labels in the column headers of a dataset, re-organise the view by moving columns, and hide datagrid columns that are not relevant.

By default, a dataset is opened in *browse* mode. If required, you can change to *edit* mode if you want to make changes to the data. You cannot, however, edit a working dataset created by a block.

To open the **Dataset File Viewer**, select the required working dataset, right-click **Open With** and then click **Data File Viewer** in the shortcut menu.

Modifying the dataset view

You can modify the dataset view to change the column order and hide variables that are not of interest.

Changes made to the displayed layout in the **Dataset File Viewer** do not change the underlying data.

Show labels

You can specify whether to use labels as column headings rather than column names. To use labels:

1. Right-click the column header of the variable that you want to show labels for, and select **Preferences**.

The **Preferences** dialog box is displayed.

2. Expand the **WPS** group and click **Dataset Viewer**.
3. Click **Show labels for column names**.
4. Click **OK**.

The setting is saved, and the **Preferences** dialog closed.

Hide variables

To hide a datagrid column:

1. Right-click the column header for the variable you want to hide.
2. Click **Hide column**.

If you hide a column for which filtering is currently active, you need to confirm that you want to remove the filter and hide the column.

Show hidden variables

To show previously hidden dataset variables:

1. Right-click the dataset header row.
2. Click **Show / Hide Columns** to display the **Show / Hide Columns** dialog box.
3. Select the required columns and click **OK**.

Move columns

To move a column in the dataset view, click the column and drag to the new location in the view. You can only move one column at a time.

Editing a dataset

An imported dataset can be opened in *edit* mode enabling you to add new observations, delete observations, or edit existing values. Datasets that are part of a Workflow cannot be edited.

To edit an imported dataset, right-click in the grid and click **Toggle edit mode**. Any changes you make in the Dataset File Viewer are not written to the dataset until you save the changes. If saving your changes fails, then any changes that were not written remain visible and details of the failure are written to the Workbench log.

Modifying values

The Dataset Viewer enables you to edit values in the dataset.

1. Double-click the value that you want to edit. The variable type affects how the value is edited:
 - Numeric or character types: The value is displayed in the cell, where you can enter a new value. If the variable has formatting applied, to see the current value as a tool tip press and hold **Shift+F8**.
 - Date, datetime or time formatted types: Click on an individual element of a date or time value, and either enter the required value or use the up or down arrows to increase or decrease it in steps.
2. When you have completed your edits, press **Enter**.

The entered value is displayed in bold type, and an asterisk (*) is appended to the observation number in the left margin of the grid.

3. To save the changes to the original dataset, click the **File** menu, and click **Save**.

Adding observations

The **Dataset Viewer** enables you to add new observations to the end of an existing dataset.

To add a new observation to the dataset:

1. Right-click the datagrid and click **Add Observation**.
2. A new row appears at the end of the dataset. Double-click each cell in the observation and modify the content as required.
3. To save the changes to the original dataset, click the **File** menu, and click **Save**.

Deleting observations

The **Dataset Viewer** enables you to remove observations from an existing dataset.

To delete an observation from a dataset:

1. Select the observation that you want to delete by clicking on its observation number in the left hand column.
2. Right-click the observation and click **Delete Observation(s)**.
3. To save the changes to the original dataset, click the **File** menu, and click **Save**.

Missing values

Enables you to set the value of a variable to missing.

1. Select the variable value that you want to set to missing.
2. Right-click the selected cell, and click **Set Missing**.

If the cell is a:

- Numeric variable type, the **Set Missing Value** dialog is displayed. The default . (full stop) is used as the value. Alternatively, you can change the missing value to one of: . (full stop), . _ (full stop followed by an underscore), or .A to .Z.
 - Character variable type, the cell is set to a single space (' ') character.
3. To save the changes to the original dataset, click the **File** menu, and then click **Save**.

Filtering a dataset

You can apply one or more filters to restrict the view to show only the data you are interested in.

To filter your dataset view:

1. Click the filter button (🔍) below the variable heading for the column you want to filter.
2. Select the criteria by which you want to filter the view and complete as required. The column is now filtered and shows a generated filter expression next to the filter button (🔍).

Dataset filter expressions

You can modify a generated filter expression applied to numeric and character variable types. The expressions available depend on the variable type. A generated filter expression applied to a variable with a date, datetime or time format cannot be modified, only cleared and re-entered (to clear the filter for a variable, click the filter button and then click **Clear Filter**).

Note:

You can only apply a filter expression to a dataset open in browse mode. Filter criteria can be set on multiple columns and, as you apply filters, the dataset contents are automatically updated.

The following table shows the supported expression syntax for numeric values.

Criteria	Expression	Example
Equal to	EQ X	Is equal to 100 EQ 100
Not equal to	NE X	Is not equal to 100 NE 100
Less than	LT X	Is less than 100 LT 100
Greater than	GT X	Is greater than 100 GT 100
Less than or equal to	LE X	Is less than or equal to 100 LE 100
Greater than or equal to	GE X	Is greater than or equal to 100 GE 100
Between (inclusive)	BETWEEN X AND y	Is between 100 and 200 BETWEEN 100 AND 200
Not between (inclusive)	NOT BETWEEN X AND y	Is not between 100 and 200 NOT BETWEEN 100 AND 200
Is missing	IS MISSING	IS MISSING
Is not missing	IS NOT MISSING	IS NOT MISSING
In	IN (x, y)	Is one of the values 100, 200 or 300 (numeric) IN (100,200,300)

The following table shows the supported expression syntax for character values.

Criteria	Expression	Example
Equal to	EQ X	Is equal to "Blanco" (string) EQ "Blanco"
Not equal to	NE X	Is not equal to "Blanco" (string) NE "Blanco"

Criteria	Expression	Example
In	IN (x, y)	Is one of the values "Blanco", "Jones" or "Smith" (character) IN ("Blanco", "Jones", "Smith")
Starts with	LIKE " s%"	Starts with the string "Bla" LIKE "Bla%"
Ends with	LIKE " %S"	Ends with the string "nco" LIKE "%nco"
Contains	LIKE " %s%"	Contains the string "an" LIKE "%an%"
Is missing	IS MISSING	IS MISSING
Is not missing	IS NOT MISSING	IS NOT MISSING

Sorting a dataset

The dataset displayed in the **Dataset Viewer** can be sorted using the values in one or more variables.

You can only sort a dataset that is open in Browse mode. Variables that are part of an active sort have an icon representing the direction of the sort in the header.

To sort a dataset:

1. Right-click the column header for the variable that you want to sort by primarily.
2. Click **Ascending Sort** or **Descending Sort** on the shortcut menu to perform the sort.
3. If required, you can apply further sorting (secondary, tertiary etc) by selecting another column and clicking **Ascending Sort** or **Descending Sort**.

To remove any column from the sort, right-click the required column and click **Clear Sort**. The dataset will remove that sort and revert to any previous sorts, if present.

Bookmarks view and Tasks view

The **Bookmarks** view lists bookmarks added to a Workflow and the **Tasks** view lists tasks added to a Workflow. Both bookmarks and tasks can be used to identify parts of the Workflow that require further investigation.

To add a bookmark:

1. In the **Project Explorer** view, select the required Workflow.

2. Click the **Edit** menu and then click **Add Bookmark**.
3. Enter a **Description** in the **Bookmark Properties** dialog box.
4. Click **OK**.

The new bookmark is displayed in the **Bookmarks** view.

To add a task:

1. In the **Project Explorer** view, select the required Workflow.
2. Click the **Edit** menu and then click **Add Task**.
3. Enter a **Description** in the **Properties** dialog box.
4. Click **OK**.

The new task is displayed in the **Tasks** view.

You can double-click on a bookmark in the **Bookmarks** view, or a task in the **Tasks** view to open the Workflow in the **Workflow Editor** view.

Using Hub Library Definitions with the Workflow Environment

Hub is an Enterprise Management Tool that provides centralised management of access to data sources.

Hub *Library Definitions* can be used with the Workflow Environment. *Library Definitions* securely store details of libraries and their options within Hub.

Logging in to Hub from Workbench

To use Hub features in Workbench, you will need to log into a Hub installation from Workbench.

Before connecting to Hub, you require details of a Hub host and access credentials for it. These details will be provided by your Hub administrator.

To log in to a Hub instance from Workbench:

1. On the **WPS** drop-down menu in Workbench, click **Hub**, then **Log in**.

2. In the **Hub Login** dialog box complete the required information:
 - a. Select the required **Protocol**, either `HTTP` or `HTTPS`.
 - b. Enter the **Host** name provided by your Hub administrator.
 - c. Enter the **Port** for Hub. By default, Hub configured for `HTTPS` will use port `8443`, and when configured for `HTTP`, port `8181`, but your Hub administrator may change these.
 - d. Enter the Hub **Username** provided by your Hub administrator.
 - e. Enter the Hub **Password** provided by your Hub administrator.
3. Click **OK** to connect.

If the connection is successful, a message stating `Hub: Logged in as username` is displayed in the Workbench status bar. If the connection fails, an error message is displayed in the **Hub Login** dialog box.

Hub Library Definitions in the WFE

Using Hub Library Definitions in the WFE.

Once Workbench is logged into an instance of Hub, Library Definitions available to that Hub User will be displayed in the Workbench **Database Explorer** view. Library Definitions can be dragged from the Database view onto the Workflow canvas.

Although Library Definition libraries are defined in Hub, the actual library call is still being made by the user logged in to Workbench, so that user will need appropriate software installed to access the library in question.

Working in the SAS Language Environment

The SAS Language Environment enables you to create, edit and run SAS language programs, along with their resulting datasets, logs and other output.

To open the SAS Language Environment perspective, click on the **SAS Language Environment** button at the top right of the screen: .

Creating a new program

You can create a new program under either the Project Explorer or File Explorer.

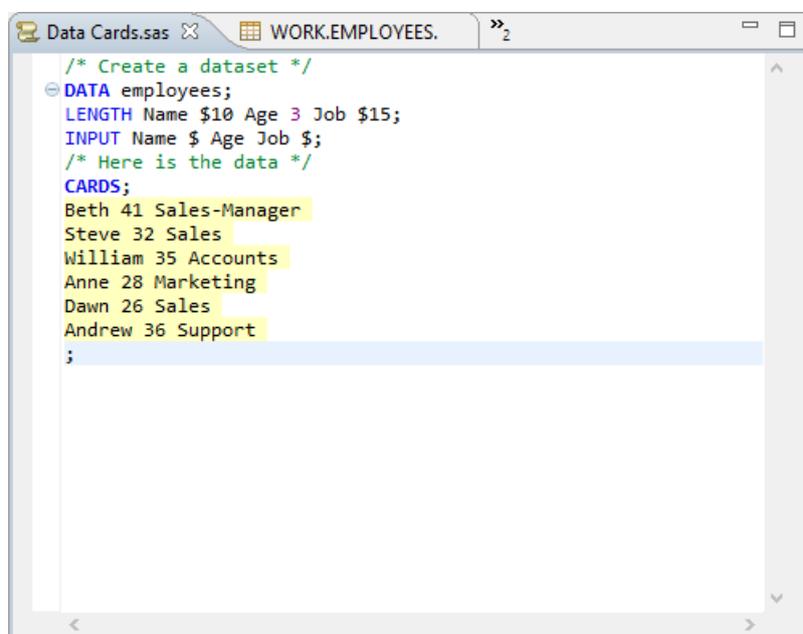
To create an empty program file in Workbench:

1. Click the **File** menu, click **New** and then click **Untitled Program**.

2. Enter the following example to create a simple dataset:

```
/* Create a dataset */  
DATA employees;  
LENGTH Name $10 Age 3 Job $15;  
INPUT Name $ Age Job $;  
/* Here is the data */  
CARDS;  
Beth 41 Sales-Manager  
Steve 32 Sales  
William 35 Accounts  
Anne 28 Marketing  
Dawn 26 Sales  
Andrew 36 Support  
;
```

Workbench displays the language elements in the program using colour coding to help you to see the structure of the code, for example:

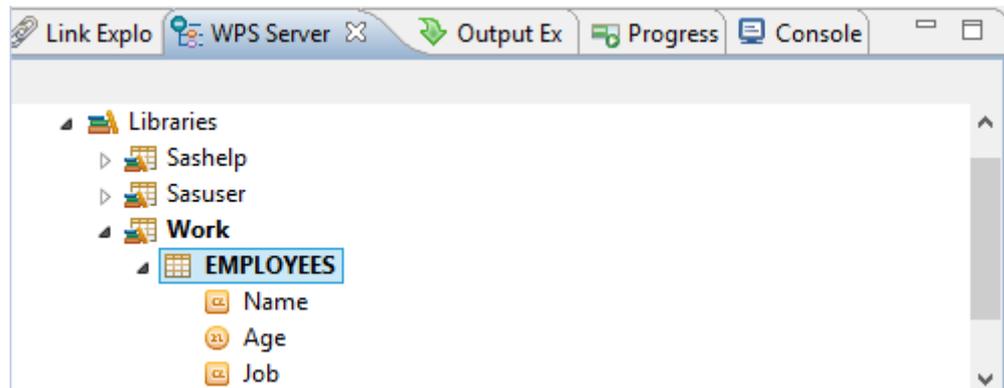


```
/* Create a dataset */  
DATA employees;  
LENGTH Name $10 Age 3 Job $15;  
INPUT Name $ Age Job $;  
/* Here is the data */  
CARDS;  
Beth 41 Sales-Manager  
Steve 32 Sales  
William 35 Accounts  
Anne 28 Marketing  
Dawn 26 Sales  
Andrew 36 Support  
;
```

3. Click the **File** menu and then click **Save as** to save your program. Use the **Save as** dialog box to select the parent folder and enter a **File name** for the program.

4. Click the **WPS** menu and then click **Run File <file name>** to run the program on the default server. Alternatively, you can click **Run File <file name> On** and then choose another server.

The generated dataset is created in the **WORK** library of the server upon which the program is run, displayed in the **WPS Server Explorer view**:



You can view the dataset properties, or open and edit the dataset in the **Editor** view (see *Editing a dataset* [↗](#) (page 140) for more information).

Creating a new program file in a project

Creating a new program.

To create a new program file:

1. Click the **File** menu, click **New** and then click **Program** to launch the **New Program** wizard.
2. Select the required parent folder at the top of the dialog, and enter a **File name**.
3. If you are creating a new program in a project, you can create a link to an existing program on the file system:
 - a. Click **Advanced** and select **Link to file in the file system**.
 - b. Click **Browse** and use the **Select Link Target** window to locate the file and click **Open**.
4. Click **Finish** to complete the task.

The new program is automatically opened in the **Editor** view.

Alternatively, you can create an untitled program, and save the program when required. To create an untitled program, click the **File** menu, click **New** and then click **Untitled Program**.

Entering WPS code via templates

Defining templates that can be entered as code shortcuts into programs.

You can define templates that can be entered as code shortcuts into programs. These templates can be accessed through the normal code completion key sequence (typically, **CTRL+Space**). For example if you define a code template with the name **template1**, you can type `temp` followed by **CTRL+Space** and the code completion feature will execute the following actions:

1. Match `temp` against the list of code templates.
2. Find your template called *template1*.
3. Enter the *Pattern* associated with the template into your program.

The *Pattern* containing the code to be entered into your program can be as simple or as complex as you like. It could just be a single word. For example, you could define a code template to expand to the name of a frequently used dataset for which the name is difficult to type. A code template could also define a large amount of text, for example, a boilerplate invocation of a particular procedure. If you are enter a variable into a *Pattern*, it needs to be enclosed within pairs of '\$\$' characters.

You can export code templates to an external file, or import them from a previously exported file. This allows you to share code templates, or to copy code templates between workspaces.

WPS Code Injection

Creating SAS language code to be run before or after your program using the Workbench code injection feature.

You can create SAS language code to be run before or after your program using the Workbench code injection feature. The code to run is entered in the **Code Injection** panel of the **Preferences** window.

Enable custom pre-code injection

When this option is selected, the code in the text box is executed immediately prior to the running of a program. This option is similar to the `INITSTMT` system option, except that this code is automatically run before each submitted program. (The `INITSTMT` system option is only processed on server startup).

Enable custom post-code injection

When this option is enabled, the code in the text box is executed immediately following the running of a program. This option is similar to the `TERMSTMT` system option except that this code is automatically run after each submitted program. (The `TERMSTMT` system option is only processed on server shutdown.)

Using Program Content Assist

The **SAS Language** editor provides a content assistance feature that suggests language elements.

The **SAS Language** editor provides a content assistance feature that suggests language elements. When you select these elements, they are automatically added to your program:

1. Type a keyword into a program that may be associated with language elements.

For example, type the word `PROC`.

2. After the word, type a single space character. and press **CTRL+SPACEBAR**.

If there are any expected language elements after the word you typed, a pop-up list is displayed of these elements.

3. You can filter the list by typing in the first few letters of the language element that you want to use.
4. Double-click the language element you require and it will added to your program.

WPS syntax colouring

Customising the colour scheme used by the SAS language editor.

To modify SAS language editor syntax colouring:

1. Click the **Window** menu and then click **Preferences**.
2. In the **Preferences** window, expand the **WPS** group and select **Syntax Colouring**.
3. For each different type of language item, specify attributes relating to **Color**, **Background Color** and whether the item should be displayed in **Bold** or **Italic**.
4. Click **OK** to save the changes and close the **Preferences** window.

Running a program

Running a program; either from inside the Workbench, or from a command line outside of the Workbench.

All libraries, datasets and macro variables are *persistent*. This means that, once they are assigned during the execution of a program, they are available to other programs that are run during the same Workbench session.

If required you can clear all previous libraries, datasets, variables and output on a WPS server for the current session by restarting the server (see [Restarting a WPS Server](#) (page 48)). Alternatively you can use the **Clear Log** , **Clear Results** options on the **WPS** menu to reset the log and program output respectively.

For more information about controlling the output (datasets, logs, results, and so on) generated from the execution of a program see *Working with program output* [↗](#) (page 145).

Running a program in Workbench

Running a program in Workbench.

Note:

When you run a program, the state from previous program executions run on the same WPS server, in the same server session, is preserved, so the program can interact with the datasets, Filerefs, and so on, that have been previously created.

1. Ensure that the program to be run is open and active in the **Editor** view.
2. Click the **WPS** menu and click **Run File <filename>** . Alternatively use the **Ctrl + R** shortcut key, or the run button () on the toolbar.

To run the program on a specified server, choose **Run File <pathname> On** or click the down arrow next to the run button, and then choose the server.

Alternatively, you can select and run an unopened SAS language program in either the **Project Explorer** or **File Explorer**. To do this, select the program and click the **WPS** menu, followed by **Run File <filename>**, or **Run File <pathname> On** and choosing the server.

Running a selected section of a program

Executing part of a SAS language program.

1. Ensure that the relevant program is open and active in the **Editor** view.
2. Highlight the section of code in the program that you want to run.
3. Click the **WPS** menu, then click **Run selected code from <filename>** and then click **Local Server** (Substitute the name of your remote WPS server in place of **Local Server**). Alternatively use the **Ctrl + R** shortcut key.

To run the selected section of code on a specified server, choose **Run selected code from <pathname> On** or click the down arrow next to the run button, and then choose the server.

Running a step

Running a step from a SAS language program.

1. Ensure that the relevant program is open and active in the **Editor** view.

2. Place the cursor in the step that you want to run.
3. Right-click and, from the context menu, click **Run Step**. Alternatively use the **Ctrl + Alt + R** shortcut key.

To run the step on a specified server, choose **Run Step On** and then choose the server.

Stopping program execution

Stopping a SAS language program before it has finished.

1. Click the **WPS** menu, click **Cancel** and then click **Local Server** (Substitute the name of your remote WPS server in place of **Local Server**) to stop program execution.
2. Once the progress indicator in the lower right corner of the Workbench disappears, this indicates that the program has stopped running.

Depending on the particular step that the program is executing, the program may not stop immediately.

When the program stops, a message similar to *ERROR: Execution was cancelled* is displayed in the log.

WPS Preferences for running programs

WPS Preferences can be set to control the behaviour of Workbench during program execution.

The **WPS** page of the **Preferences** contains options affecting how SAS language programs are run. Before running any programs, select your preferred settings from the page:

Save all modified files before running a program

Specifies whether modified programs open in the **Editor** view are saved before a program is run. The behaviour can be one of: **Always**, **Never** or **Prompt**. The default, **Prompt**, displays the **Save and Run** dialog box listing modified files.

- If you select **Always save resources before run**, all modified files open in the **Editor** view are saved before the program is run.
- If you select **Never save resources before run**, no changes to files open in the **Editor** view are saved before the program is run. Errors may occur if your program relies on other files open in Workbench that have been modified but not saved.

Close all open datasets before running a program

Specifies whether open datasets are closed before running a program. The behaviour can be one of: **Always**, **Never** or **Prompt**. The default, **Prompt**, displays the **Datasets are open** dialog box listing all opened datasets.

- If you select **Always close datasets before run**, all open datasets are closed before the program is run.

- If you select **Never close datasets before run**, all datasets are left open when the program is run. This may cause errors if the program attempts to modify an open dataset.

Show log when an error is encountered running a program

Specifies whether the log file is displayed when an error is encountered in a program. The behaviour can be one of: **Always**, **Never** or **Prompt**. The default, **Prompt**, displays the **Confirm Log Display** dialog box:

- If you select **Remember my decision**, the log file is always shown whenever an error is encountered.
- If you clear **Remember my decision**, the log file is not shown, even when an error is encountered.

Close all open datasets when deassigning a library

Specifies whether open datasets are closed when the containing library is deassigned. The behaviour can be either **Always** or **Prompt**. The default, **Prompt**, displays a dialog listing open datasets that must be closed to de-assign the library.

Confirm deletions

Specifies whether confirmation is required when deleting an item in the **WPS Server Explorer** view.

Confirm server restart

Specifies whether confirmation is required when you restart a server in the **WPS Server Explorer** view.

Autoscroll the log

Specifies whether the log output is automatically scrolled (see [Autoscroll](#) (page 77)).

Autoscroll the listing

Specifies whether the listing output is automatically scrolled (see [Viewing the listing output](#) (page 147)).

Show advanced server actions

Controls the context menu options available when restarting WPS servers (see [Restarting with Advanced Server Actions](#) (page 48)).

Temporary file location

This field points to the directory where the Workbench stores temporary files. The log files created during execution of a program are stored in this directory, as are any ODS HTML results generated by programs.

You should be aware that output files, particularly logs, can be sizeable, so you may find that you need to modify this setting if the field points by default to a location with space restrictions.

Command line mode

Running SAS language programs on a WPS server from the command line.

To use the WPS server from a command line, you need to execute the `wps` program. This executable file is located in the `bin` directory of your WPS installation.

When running the WPS server from a command line, you cannot compile programs into run-time jobs.

Options

Running `wps` on its own will not do anything, and so you will need to pass some additional instructions in the following form:

```
wps <options> <programFileName>
```

The `<options>` arguments can be a list of any of the following:

- `-optionname [optionalValue]` – specifies a WPS system option.
- `-config <configFileName>` – specifies a configuration file containing WPS system options.
- `-set <envVarName> <envVarValue>` – specifies an environment variable that takes effect when the WPS server is run.

You can define any number of `<options>` arguments, but the `<programFileName>` must always be the last argument on the command line.

System options

To set a system option specify:

```
wps -optionname [optionalValue] <programFileName>
```

You can pass as many `-optionname` system options to the WPS server as required. However, you can only pass single value options, so must specify `-optionname [optionalValue]` for each system option.

Configuration files

System options can be defined in a configuration file, and this file referenced when running the WPS server:

```
wps -config <configFileName> <programFileName>
```

For more information about configuration files and the order in which they are processed, see *Configuration files* [↗](#) (page 361).

Environment variables

To set an environment variable to use with the WPS server:

```
wps.exe -set <envVarName> <envVarValue> <programFileName>
```

You can set as many environment variables as you require. However, you can only pass single value options, so must specify `-set <envVarName> <envVarValue>` for each environment variable.

Log output

When you run the WPS server from the command line, log information is sent to the standard output (*stdout*) for the operating system. Once the program has finished running this log information is lost. To preserve the log output, redirect *stdout* to a file, for example:

```
wps program1.sas > program1.log
```

Results output

When you run the WPS server from the command line, any results generated by the program are automatically captured into a file called *<programFileName>.lst* created in the same directory as the program. For example, a file called *Program1.lst* is output if you run a program called *Program1.sas*.

Libraries and datasets

A library is used to store WPS Analytics data files (including datasets), folder locations and database connections via the `LIBNAME` statement.

On the majority of operating systems, a library maps directly to an operating system folder. The operating system folder may contain non-WPS Analytics files, but only the files with extensions that WPS Analytics recognises will be visible in Workbench.

The default **Libraries** that are visible under each server in the **WPS Server Explorer** view are:

- `SASHELP` – a read-only library containing a variety of system metadata.
- `SASUSER` – used to store datasets and other information that you want to keep beyond the end of your WPS Analytics session. To use this location, add `SASUSER.` in front of the name of the relevant dataset in your program.

- `WORK` – where your datasets, catalogs and Filerefs are stored by default. `WORK` is a 'temporary working space' that is automatically cleared of all data when your WPS server session ends.

To preserve the output that your program creates, you can redefine the work location to a non-temporary location (see [Setting the `WORK` location](#) (page 123)), or use the `LIBNAME` statement to write output you want to keep to your own permanent location (such as a directory or database connection). For example:

```
LIBNAME mylib '/directories/mydirectory/';
```

Any directory or database connection that you associate with the alias must already exist before you assign it. Once a dataset has been created and stored in this location, the associated alias (`mylib`) must be used to retrieve the dataset.

Setting the `WORK` location

Change the `WORK` location when SAS language programs are run in Workbench.

To set the pathname for the `WORK` location that is used every time you start up a particular server:

1. In the **WPS Server Explorer** view, right-click the required server and click **Properties** in the shortcut menu.
2. In the properties list, expand the **Startup** group and select **System Options**.
3. On the **System Options** panel, click **Add** to display the **Startup Option** dialog box.
4. In **Name** enter `WORK`, and in **Value** enter the full pathname for the new `WORK` location.
5. Click **OK** to save the system option, and click **OK** on the **Properties** dialog box to restart the server and apply your changes.

The `WORK` location is redefined when using Workbench to run SAS language programs, but not if WPS is run from the command line. When running programs in batch mode, the `WORK` system option must be set in a configuration file.

Catalogs

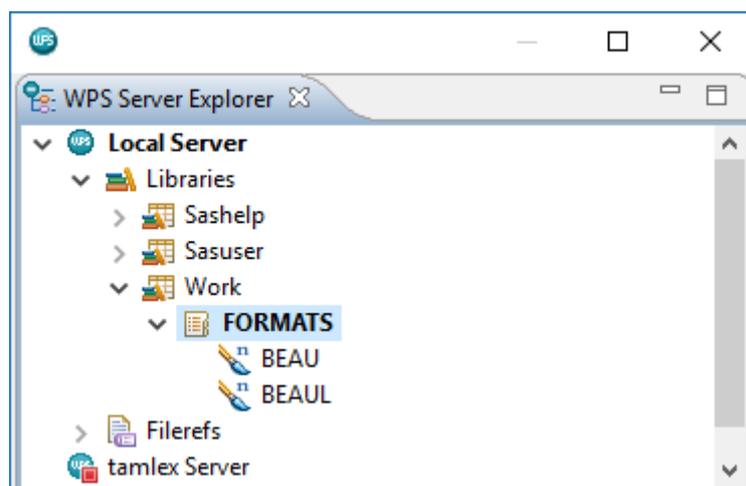
A *catalog* is a folder in which you can store different types of information (called *catalog entries*), for example formats, informats, macros, and graphics output.

For example, you could write a program to generate a catalog (`FORMATS`) in which to store `PICTURE` formats:

```
PROC FORMAT;  
  
PICTURE beau 0-HIGH = '00 000 000.09' ;  
PICTURE beaul 0-HIGH = '00 000 000.09' ;  
RUN;  
  
PROC CATALOG CATALOG = WORK.FORMATS;  
CONTENTS;  
RUN;
```

The `PICTURE` statement allows you to create templates for printing numbers by defining a format that allows for special characters, such as: leading zeros, decimal and comma punctuation, fill characters, prefixes, and negative number representation.

When you run the program, the catalogs are created under the `WORK` folder:



To view the *entries* associated with the catalogs, right-click on them and click **Properties**.

Datasets

A dataset is a file created or imported by a WPS server, and stored in a library.

A dataset contains rows and columns, known as *observations* and *variables* respectively.

- Observations (rows) usually relate to one specific entity (for example, an order or person).
- Variables (columns) describe attributes of an entity (for example, an item ID in an order).

Example Dataset

In this example dataset, each observation is an item in the order. The variables are *Order ID*, *Item ID*, *Quantity* and *Unit Price*.

Order ID	Item ID	Quantity	Unit Price
10001	47853	3	30.75
10001	23104	10	4.90
10002	62091	1	89.99

Open a dataset

You can open a dataset in the **WPS Server Explorer** view. Expand the required server node, expand **Libraries** and double-click the required dataset.

Delete a dataset

You can delete a dataset stored on disk in the **WPS Server Explorer** view. Expand the required server node, expand **Libraries** node and expand the required library. Select the dataset, right-click and click **Delete** in the shortcut menu.

Importing datasets

In addition to the datasets created by running SAS language programs, datasets can be created by importing data from external files.

Supported formats for datasets are:

- Delimited
- Fixed Width
- Microsoft Excel Spreadsheet Files

The **Dataset Import Wizard** enables you to configure the dataset as it is imported. It is not always necessary to go through all of the steps in the wizard; you can import a dataset from a file and click **Finish** on the first page to accept all the default options. If you import a dataset with the same name as an existing one, you will overwrite it.

When importing delimited files, Workbench deduces the delimiter character from the data. If this delimiter is not correct, you can change it prior to importing the dataset.

When importing fixed width files, Workbench deduces column start and end from the data. If this widths are not correct, you can change the widths prior to importing the dataset.

Spreadsheet files may contain several worksheets. By default, Workbench will select the first sheet that has data for importing. A sheet may also have *names* that refer to cell ranges; it is possible to import data from these names and also from the clipboard.

You can drag a dataset file from the **Project Explorer** or **File Explorer** to the required library on the WPS Server.

- You can only drag files from the **Project Explorer** to the Local Server.
- You can drag datasets from any folder accessed through the same connection node as the required WPS server. You cannot, for example, drag a dataset from your Local connection to a remote connection. To do this first use the **File Explorer** to copy the dataset to the required server, and then drag the dataset from the new location.

You can also import a dataset using one of the following methods:

- In SAS language code, for example using the XLSX engine:

```
LIBNAME xlsxlib XLSX 'myfile.xlsx' HEADER=YES;  
DATA WORK.IMPORTED;  
SET xlsxlib.Sheet1;  
RUN;
```

- The **Dataset Import Wizard**. When importing Excel Workbooks in either XLS or XLSX format, this method uses the EXCEL engine and you will therefore need to have the *Microsoft ACE OLEDB* engine installed.

If you do not have Microsoft Excel installed, you will need to install the *Microsoft Access Database Engine Redistributable* to import datasets.

Importing datasets from files

Importing datasets from files using **WPS Server Explorer**.

To import files:

1. In the **WPS Server Explorer** view, right-click local server and click **Import Dataset** on the shortcut menu.

Alternatively, you can select a file in from the file system and either drag the file to the local server or copy and paste the file to the local server.

2. In the **Dataset Import** dialog box, enter a file name or click **Change** and open the required file.
3. Select a **Library Name** for the dataset and, if required, change the **Member Name** to rename the dataset.
4. The type of format is inferred from the type of file imported. If required, change the type of **Data Format** that best describes the dataset content.

5. Click **Next** and Configure the imported dataset depending on the type:
 - If you select **Fixed width**, for each **Column** modify the **Start** value and **Width** value to alter the width of the exported variable.
 - If you select **Delimited**:
 - Select the required delimiter. To define your own delimiter, select **Other** and enter the delimiter character.
 - Select **Include field names on first row** to include the names of the columns.
 - If you select **Excel**:
 - If required, select a different worksheet in **Whole Worksheet**. You can also select an existing **Named Range** from the existing or new worksheet.
 - Select **Include first row names as column headers** to use the first row as variable names.
6. Click **Next** and, if required, edit the dataset column properties.
 - To change a column name, select a column header in the preview and edit the **Column Name**.
 - To change a column label, select a column header in the preview and edit the **Column Label**.
 - To change the column data type, select a column header in the preview edit the **Column Data Format**.
7. Click **Finish** to import the dataset.

Importing spreadsheet data

You can use the **Dataset Import Wizard** to import a range of cells directly from a Microsoft Excel spreadsheet.

Note:

Data imported will have the last-saved value. Cell formulae are not reevaluated when the spreadsheet is opened, and the last-saved value is used for these cells.

1. Select the contiguous range in your Excel spreadsheet that you want to import and copy and paste the content to the required server in the **WPS Server Explorer** view.

Alternatively, you can drag the selected range to the required library in the required server.
2. In the **Dataset Import** dialog box, enter a file name or click **Change** and open the required file.
3. Select a **Library Name** for the dataset and, if required, change the **Member Name** to rename the dataset.
4. The type of format is inferred from the type of file imported. If required, change the type of **Data Format** that best describes the dataset content.

5. Click **Next** and Configure the imported dataset depending on the type:
 - If you select **Fixed width**, for each **Column** modify the **Start** value and **Width** value to alter the width of the exported variable.
 - If you select **Delimited**
 - Select the required delimiter. To define your own delimiter, select **Other** and enter the delimiter character.
 - Select **Include field names on first row** to include the names of the columns.
6. Click **Next** and, if required, edit the dataset column properties.
 - To change a column name, select a column header in the preview and edit the **Column Name**.
 - To change a column label, select a column header in the preview and edit the **Column Label**.
 - To change the column data type, select a column header in the preview edit the **Column Data Format**.
7. Click **Finish** to import the dataset.

Exporting datasets

WPS datasets can be exported as delimited and fixed width text, and also as Microsoft Excel Workbooks. The **Dataset Export Wizard** allows you to fine tune the contents of export file as required.

To export a dataset:

1. In the **WPS Server Explorer** view, expand the required server, right-click the required dataset and click **Export Dataset** on the shortcut menu.
2. In the **Dataset Export Wizard**, select the export type, one of: **Delimited**, **Fixed Width** or **Excel**. For **Excel**, select the required file format and click **Next**.
3. Configure the exported dataset depending on the export type:

If you select fixed width, for each **Column** modify the **Start** value and **Width** value to alter the width of the exported variable.

If you select Delimited:

- Select the required delimiter. To define your own delimiter, select **Other** and enter the delimiter character.
 - Select **Include field names on first row** to include the names of the columns.
4. In the **Select Destination File** panel, enter a file name for the exported dataset and click **Finish**.

Removing dataset references

How to remove a dataset reference.

You cannot remove a dataset reference on its own. To remove a dataset reference, you must restart the required server, which also removes any ODS output, log entries, other libraries, and file references. For more information, see *Restarting a WPS Server* [↗](#) (page 48).

Data Profiler view

The **Data Profiler** view enables you to view content and details about a dataset.

To open the **Data Profiler** view, select the required dataset, right-click **Open With** and then click **Data Profiler** in the shortcut menu.

Summary View

The **Summary View** lists information about the dataset, including the number of observations and variables in the dataset as well as variable characteristics.

The **Summary View** contains two panels: **Summary** and **Variables**.

Summary

The **Summary** panel displays summary information about the dataset: the dataset name, the total number of observations, and the total number of variables in the dataset.

Variables

The **Variables** panel displays information about the variables in the dataset as follows:

Variable

The name of the variable in the dataset.

Label

The alternative display name for the variable.

Type

The type of the variable. This can be one of:

- *Numeric*, for numbers and date and time values.
- *Character* for character and string data.

Classification

The category of the variable; this can be one of:

- *Categorical*. A non-numeric variable that can contain a limited number of possible values, and the limit is below the specified classification threshold.
- *Continuous*. A numeric variable that can contain an unlimited number of possible values. This classification is used for numeric variables where the number of distinct values in the variable is greater than the specified classification threshold.
- *Discrete*. A numeric variable that can contain a limited number of possible values. This classification is used for character variables where the number of distinct values in the variable is less than or equal to the specified classification threshold.

The classification threshold defines a number of unique values above which a variable is classified as continuous, and below which a variable is either categorical (if non-numeric) or discrete (if numeric). The classification threshold is specified on the **Data** panel of the **Preferences** dialog box.

Length

The size required to store the variable values. For character types, the number represents the maximum number of characters found in a variable value. For numeric types, the number represents the maximum number of bytes required to store the value.

Format

How the variable is displayed when output. For more information about formats see the section *Formats* in the *WPS Reference for Language Elements*.

Informat

The formatting applied to the variable when imported into WPS Analytics. For more information, about informat, see the section *Informats* in the *WPS Reference for Language Elements*.

Distinct Values

The number of unique values in the variable. If the variable classification is Continuous, the display indicates there are more unique values than the specified classification threshold.

Missing Values

The number of missing values in the variable.

Frequency Distribution

For each value in a non-continuous variable, displays a chart showing the number of occurrences of each value in the variable.

Data

The **Data** panel lists all observations in a dataset and enables you to view, filter, or sort those observations.

The **Data** panel in the **Data Profiler** view is a read-only version of the **Dataset Viewer**. You can modify the view, or sort and filter data in the **Data** panel. If you want to edit values in the dataset, you need to use the **Dataset Viewer**. For more information about sorting and filtering observations, and about the **Dataset Viewer** see *Dataset Viewer* [↗](#) (page 105).

Univariate View

The **Univariate view** tab displays summary statistics for all numeric univariate variables.

The columns displayed in the **Univariate View** tab are determined using the **Calculate Statistics** dialog box. Click **Configure Statistics**  to open **Preferences** and select the statistics you want displayed:

Quantiles

Lists quantile points.

Variable Structure

Lists statistics describing the variable, such as number of missing values, and minimum and maximum values.

Measures of Central Tendency

Lists measures used to identify the central tendency in the values of the variable (mean, median and mode).

Measures of Dispersion

Lists measures used to show the dispersion of a variable.

Others

Lists other statistics that can be displayed in the statistics table.

Univariate Charts

The **Univariate Charts** tab enables you to view the frequency distribution table and graphs for a selected dataset variable.

The information displayed is determined by the variable selected in the **Variable Selection** list. Each section in the tab displays the frequency of values occurring in the selected variable.

Frequency Table

Displays the frequencies of the values in the specified variable, the percentage of the total number of observations for each value, and cumulative frequencies and percentages.

If the variable type is categorical or discrete, the table displays one row for each value. If the variable type is continuous, the variable is binned and frequency information is displayed for each bin.

Frequency Chart

Displays the frequency of values as the percentage of the total number of observations for the variable, either as a histogram, line chart or pie chart.

The chart can be edited and saved. Click **Edit chart**  to open the Chart Editor from where the chart can be saved to the clipboard.

Correlation Analysis

The **Correlation Analysis** tab enables you to view the strength of the correlation between numeric variables in the dataset.

Options

Specifies the coefficient type to use when comparing variables, and which numeric variables in the dataset are to be compared.

Coefficient

Specifies the type of coefficient used to compare values.

Pearson

Specifies the Pearson correlation coefficient, which is defined as:

$$r_{x,y} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

where:

- $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ is the mean of variable x
- $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ is the mean of variable y

Spearman's Rho

Specifies the Spearman's rank correlation coefficient, which is defined as:

$$r_s = \frac{\sum_{i=1}^n (R_{x_i} - \bar{R}_x)(R_{y_i} - \bar{R}_y)}{\sqrt{\sum_{i=1}^n (R_{x_i} - \bar{R}_x)^2 \sum_{i=1}^n (R_{y_i} - \bar{R}_y)^2}}$$

where:

- R_{x_i} is the rank of x_i
- $\bar{R}_x = \frac{1}{n} \sum_{i=1}^n R_{x_i}$ is the mean of the ranked variable R_x
- R_{y_i} is the rank of y_i
- $\bar{R}_y = \frac{1}{n} \sum_{i=1}^n R_{y_i}$ is the mean of the ranked variable R_y

The values of the variables are first ranked and the ranks compared. Using this coefficient may therefore be more robust to outliers in the data.

Kendall's Tau

Specifies the Kendall rank correlation coefficient, which is defined as:

$$\tau_b = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}}$$

where:

- $n_0 = \frac{n(n-1)}{2}$
- $n_1 = \sum_{i=1}^n \frac{t_i(t_i-1)}{2}$
- $n_2 = \sum_{j=1}^n \frac{u_j(u_j-1)}{2}$
- n_c is the number of concordant pairs
- n_d is the number of discordant pairs
- t_i is the number of tied values in the i th group of tied values for the first variable
- u_j is the number of tied values in the j th group of tied values for the second variable
- The difference in the number of concordant and discordant pairs can be expressed as:

$$n_c - n_d = \sum_{i < j} \text{sign}(x_i - x_j) \text{sign}(y_i - y_j)$$

The values of the variables are first ranked and the ranks compared. Using this coefficient may therefore be more robust to outliers in the data.

Select Variables

Displays a list of numeric variables in the dataset, from which you can select the variables to compare.

Correlation Coefficient Matrix

The matrix is shown in coloured blocks, with the size and colour of the blocks indicating the relationship between the compared variables. The scale for the items displayed ranges from 1 a strong positive correlation to -1 where there is a strong negative correlation.

Correlation Statistics

Displays the correlation statistics and scatter plot for the block selected in the Correlation Coefficient Matrix. The table displays the variables being compared, the coefficient type and the p-value for the correlation coefficient being non-zero. The scatter plot displays all values in the dataset, using the same colour scale as the matrix, as either a plot or heat map if the number observations is very high.

Predictive Power

The **Predictive Power** tab enables you to view the predictive power of independent variables in the dataset in relation to a selected dependent variable.

The information displayed is determined by the variable selected as the **Dependent Variable**. Each section displays the relationship between the independent variables in the dataset and the specified **Dependent Variable**.

Statistics Table

The **Statistics Table** displays all the variables in the dataset and a series of predictive power statistics. The statistics displayed are:

- **Entropy Variance.** Displays the *Entropy Variance* value for each variable in relation to the specified **Dependent Variable**.
- **Chi Sq.** Displays the *Chi-Squared* value for each variable in relation to the specified **Dependent Variable**.
- **Gini.** Displays the *Gini Variance* value for each variable in relation to the specified **Dependent Variable**.

For more information about how these values are calculated, see [Predictive power criteria](#) (page 102).

Entropy Variance Chart

Displays the bar chart of independent variables' entropy variances in relation to the specified **Dependent Variable**. The variables are displayed in order from the highest entropy variance to the lowest. The number of variables displayed in the chart is determined by the preferences set in the **Data Profiler** panel of the **Preferences** dialog box.

The chart can be edited and saved. Click **Edit chart**  to open the Chart Editor, from where the chart can be saved to the clipboard.

Frequency Chart

Displays the frequency of each value of the independent variable selected in the **Statistics Table** for each value of the specified Dependent Variable.

- Click **View whole data**  to display the overall predictive relationship between values of an independent variable selected in the **Statistics Table** and the specified **Dependent Variable**.
- Click **View breakdown data**  to display the frequency of each value of an independent variable, and its relationship to the outcomes for the specified **Dependent Variable**.

The chart can be edited and saved. Click **Edit chart**  to open a the Chart Editor from where the chart can be saved to the clipboard.

Predictive power criteria

Predictive power is a way of measuring how well a particular input variable can predict the target variable.

Pearson's chi-squared statistic

Pearson's chi-squared statistic is a measure of the likelihood that the value of the target variable is related to the value of the predictor variable.

Each observation in the dataset is allocated to a cell in a contingency table, according to the values of the predictor and target variables. Pearson's chi-squared statistic is calculated as the normalised sum of the squared deviations between the actual number of observations in each cell, and the expected number of observations in each cell if there were no relationship between the predictor and target variables.

If a predictor variable has a high Pearson's chi-squared statistic, it means that the variable is a good predictor of the target variable, and is likely to be a good candidate to use to split the data in a binning or tree-building algorithm.

Pearson's chi-squared statistic for a discrete target variable is calculated as

$$\chi^2 = \sum_{i=1}^r \sum_{j=1}^c \frac{(n_{ij} - \mu_{ij})^2}{\mu_{ij}}$$

$$\mu_{ij} = \frac{n_{i*} n_{*j}}{N}$$

where

- r is the number of distinct values of the predictor variable X (these are the rows in the contingency table)
- c is the number of distinct, discrete values of the target variable Y (these are the columns in the contingency table)
- n_{ij} is the number of observations for which the predictor variable X has the i th value, X_i , and the target variable Y has the j th value, Y_j (these are the values in the cells of the contingency table)
- μ_{ij} is the expected value of n_{ij}
- n_{i*} is the total number of observations for which the predictor variable X has the i th value, X_i
- n_{*j} is the total number of observations for which the target variable Y has the j th value, Y_j
- N is the total number of observations in the dataset

Entropy variance

Entropy variance is a measure of how well the value of a predictor variable can predict the value of the target variable.

If a variable in a dataset has a high entropy variance, it means that the variable is a good predictor of the target variable, and is likely to be a good candidate to use to split the data in a binning or tree-building algorithm.

Entropy variance for a discrete target variable is calculated as

$$E_r = 1 - \frac{\sum_{i=1}^r \left(\frac{n_{i*} E_i}{N} \right)}{E}$$

$$E_i = -\frac{1}{\log(c)} \sum_{j=1}^c \frac{n_{ij}}{n_{i*}} \log\left(\frac{n_{ij}}{n_{i*}}\right)$$

$$E = -\frac{1}{\log(c)} \sum_{j=1}^c \frac{n_{*j}}{N} \log\left(\frac{n_{*j}}{N}\right)$$

where

- r is the number of distinct values of the predictor variable, X
- n_{i*} is the total number of observations for which the predictor variable X has the i th value, X_i

- E_i is the entropy calculated for just the observations where the predictor variable is X_i
- N is the total number of observations in the dataset
- E is the entropy calculated for all the observations
- c is the number of distinct, discrete values of the target variable, Y
- n_{ij} is the number of observations for which the predictor variable X has the i th value, X_i , and the target variable Y has the j th value, Y_j
- n_{*j} is the total number of observations for which the target variable Y has the j th value, Y_j

Gini variance

Gini variance is a measure of how well the value of a predictor variable can predict the target variable.

If a variable in a dataset has a high Gini variance, it means that the variable is a good predictor of the target variable, and is likely to be a good candidate to use to split the data in a binning or tree-building algorithm.

Gini variance for a discrete target variable is calculated as

$$G_r = 1 - \frac{\sum_{i=1}^r \left(\frac{n_{i*} G_i}{N} \right)}{G}$$

$$G_i = 1 - \frac{\sum_{j=1}^c n_{ij}^2}{n_{i*}^2}$$

$$G = 1 - \frac{\sum_{j=1}^c n_{*j}^2}{N^2}$$

where

- r is the number of distinct values of the predictor variable, X
- n_{i*} is the total number of observations for which the predictor variable X has the i th value, X_i
- G_i is the Gini impurity calculated for just the observations where the predictor variable is X_i
- N is the total number of observations in the dataset
- G is the Gini impurity calculated for all the observations
- c is the number of distinct, discrete values of the target variable, Y
- n_{ij} is the number of observations for which the predictor variable X has the i th value, X_i , and the target variable Y has the j th value, Y_j
- n_{*j} is the total number of observations for which the target variable Y has the j th value, Y_j

Information value

Information value is a measure of the likelihood that the value of the target variable is related to the value of the predictor variable. The information value measure is only applicable for binary target variables (that is, target variables that can take one of exactly two values).

If a predictor variable has a high information value, it means that the variable is a good predictor of the target variable, and is likely to be a good candidate to use to split the data in a binning or tree-building algorithm.

The information value statistic is calculated as

$$IV = \sum_{i=1}^r \left(\frac{n_{i0}}{n_{*0}} - \frac{n_{i1}}{n_{*1}} \right) WOE_i$$

$$WOE_i = \ln \left(\frac{n_{i0}}{n_{*0}} + \alpha \right) - \ln \left(\frac{n_{i1}}{n_{*1}} + \alpha \right)$$

where:

- r is the number of distinct, discrete values of the predictor variable X (these are the rows in the contingency table)
- n_{i0} is the number of observations for which the predictor variable X has the i th value, X_i , and the target variable Y has the value, Y_0 (these are the values in the cells of the Y_0 column in the contingency table)
- n_{i1} is the number of observations for which the predictor variable X has the i th value, X_i , and the target variable Y has the value, Y_1 (these are the values in the cells of the Y_1 column in the contingency table)
- Y_0 and Y_1 are the two possible values of the binary target variable Y
- n_{*0} is the total number of observations for which the target variable Y has the value, Y_0
- n_{*1} is the total number of observations for which the target variable Y has the value, Y_1
- WOE_i is the WOE value for observations where the predictor variable is X_i
- $\alpha \ll 1$ is the weight of evidence (WOE) adjustment, a small positive number to avoid infinite values when $n_{i0} = 0$ or $n_{i1} = 0$

Dataset Viewer

You can view, filter or sort a dataset within the viewer. You can also edit observation variables, remove observations, and also add new observations.

The contents of the dataset are shown in a datagrid. The rows of the grid represent the dataset's observations, and the columns represent dataset variables.

You can display labels in the column headers of a dataset, re-organise the view by moving columns, and if some of the columns are irrelevant for a task, you can hide them.

Mode

A dataset is opened in a read-only *Browse* mode, but can be converted to *Edit* mode if you want to make changes to the data. To edit the dataset content, click the **Data** menu and then click **Toggle Edit Mode**.

Organising the dataset

Manipulating how a dataset appears.

You can manipulate the view of the dataset so that you hide variables you do not need, and move columns so that they are in a different order. Any changes made in this view do not change the underlying data.

Show labels

You can specify whether to display labels for columns rather than column names. In the **Preferences** window click **WPS**, click **Dataset Viewer** and then click **Show labels for column names**.

Hide variables

To hide a datagrid column:

1. Right-click the column header for the variable you want to hide.
2. Click **Hide column(s)**.

The datagrid column disappears from the view. If you hide a column for which filtering is currently active, you need to confirm that you want to remove the filter and hide the column.

Show hidden variables

To show previously hidden dataset variables:

1. Right-click the dataset header row.
2. Click **Show / Hide Columns**.
3. In the **Show / Hide Columns** dialog, select the columns to show and click **OK**.

Move columns

You can move columns in the dataset view:

1. Left-click and hold the mouse button on the column header.
2. Drag the column to where you want it to be.

You will see a thick grey line appear between columns, indicating where the column will appear.

3. Release the mouse button, and the column appears in the indicated location.

Editing a dataset

Editing or deleting observations from a dataset.

You can edit observations, and also add and delete them, if the dataset is in *Edit* mode.

Controls for editing a dataset are available from the **Data** menu, from the toolbar, and also from the dataset context menu.

If the dataset is open in *Browse* mode, you can switch to Edit mode by clicking **Edit**:

	Variable	Value	Density	Count
1	x1	-3.401969236	0.0199450552	1
2	x1	-3.385345503	0.0209959695	0
3	x1	-3.368721769	0.0220857068	0
4	x1	-3.352098035	0.0232148506	0
5	x1	-3.335474301	0.0243839356	0
6	x1	-3.318850567	0.0255934866	0
7	x1	-3.302226834	0.0268440165	0
8	x1	-3.2856031	0.0281359545	0
9	x1	-3.268979366	0.0294697465	0

You can make as many changes as required. Changes are not automatically written to the dataset, you must choose to save changes. If saving your changes fails, then any changes that were not written remain visible and details of the failure are written to the Workbench log

Modifying observations

Editing observation variables (cells) as raw values.

You can edit observation variables (cells) as raw values. If your variables have any formatting applied, you will not see this during editing; once edit is complete the formatted observation is displayed.

1. Double-click the cell that you want to edit:

Variable Type

Numeric

Description

The raw, unformatted value is displayed in the cell where you can enter a new value. If you want to see how the current cell appears when formatted, press **Shift +F8** and it will be shown as a tooltip.

DATE, DATETIME or TIME

DATE, DATETIME and TIME values are edited as their respective types. You can click on any individual element of a date or time (such as a year or hour) in-place, and use the keyboard arrow keys or mouse wheel to increase or decrease them in steps. Numeric elements can also be overtyped with replacement values.

Variable Type**Description**

You can use the alternative edit control associated with each type. DATE values have a  calendar, times have a  time picker, and DATETIME values have a  DATETIME picker combination. Click on the button to the right of the in-place value and the appropriate editor will appear.

String

The raw, unformatted value is displayed in the cell where you can enter a new value. If you want to see how the current cell appears when formatted, press **Shift +F8** and it will be shown as a tooltip.

2. When you have completed your edits, press **Enter**.

If the value you entered is different from the original saved value, then the observation value is displayed in bold type with an asterisk (*) next to the observation number in the left margin of the grid.

3. Changes are not saved to the dataset immediately. To save the changes to the original dataset, click the **File** menu, and then click **Save**. If the changes have been made in error, you can undo all edits (see *Canceling changes* [↗](#) (page 142)).

Adding observations

Adding new observations to a dataset.

New observations can be added to a dataset. They are always added to the end of the dataset, regardless of your current position within it. To add a new observation:

1. Click the **Data** menu and then click **Add Observation**.
2. The new row will appear in bold type, and a plus character (+) will appear next to the observation number in the left margin of the grid. Double-click each cell in the observation and modify the content as required.
3. Changes are not saved to the dataset immediately. To save the changes to the original dataset, click the **File** menu, and then click **Save**. If the changes have been made in error, you can undo all edits (see *Canceling changes* [↗](#) (page 142)).

Deleting observations

Deleting observations from a dataset.

To delete an observation from a dataset:

1. Select the observation that you want to delete by clicking on its observation number in the left hand column. To delete multiple observations, hold down the **CTRL** key while selecting the observations you want to delete.

2. Click the **Data** menu, and then click **Delete Observation**.
3. Changes are not saved to the dataset immediately. To save the changes to the original dataset, click the **File** menu, and then click **Save**. If the changes have been made in error, you can undo all edits (see [Cancelling changes](#) (page 142)).

Cancelling changes

Cancelling unsaved changes to a dataset.

You can cancel any changes that you have made but not yet saved. To cancel unsaved changes:

1. Click the **Data** menu and then click **Cancel edits**.
2. In the **Confirm Cancel Edits**, click **Yes** to undo any changes.

Missing values

Setting an observation as *missing*.

You can set an observation variable to missing using the **Set Missing** option.

Note:

When numeric variables are formatted as DATE, DATETIME or TIME, this is the only way to set these variables as missing.

1. Select the observation variable (cell) you want to set as missing.
Only one variable at a time can be set to missing in a single operation.
2. Right-click the selected cell, and click **Set Missing** in the shortcut menu.
 1. If the cell is a character variable type, the cell is set to a single space (' ') character.
 2. If the cell is a numeric variable type, the **Set Missing Value** dialog is used to set the missing value. If the observation variable is already set to a missing value, then the current raw value is shown. If the observation variable is not set to missing, the default . (full stop) value is used. If required, you can change the missing value to the value to one of: . (full stop), . _ (full stop followed by an underscore), or .A to .Z.
3. Changes are not saved to the dataset immediately. To save the changes to the original dataset, click the **File** menu, and then click **Save**. If the changes have been made in error, you can undo all edits (see [Cancelling changes](#) (page 142)).

Filtering a dataset

You can filter the contents of the dataset by variable to show only the data that you need.

Filters are cumulative, if you filter on more than one variable, all filters apply. You can cancel any of the filters at any time.

You can set filter criteria on multiple columns and, as you apply filters, the dataset contents are automatically updated.

To filter your dataset view:

1. Click the filter  button below the variable heading for the column that you want to filter.
2. Select the criteria by which you want to filter the view and Complete the criteria for the filter by entering the appropriate details.

Data Type	Criteria
DATE, DATETIME or TIME	Most expressions based on a date or time variable are set in another dialog. Use the calendar and clock controls to set the date criteria. For DATETIME values, click the clock  button to set the time component of the filter value.
Other numeric or string	The expression for the filter is shown below the variable header. Enter the value or values necessary to complete the expression in the edit box.

For a list of supported filter expressions, see [Dataset filter expressions](#)  (page 143).

To clear the filter for a variable, click the filter button and then click **Clear Filter**.

Dataset filter expressions

Syntax for editing dataset filter expressions.

For variable types other than DATE, DATETIME or TIME formats, you can edit the filter expression that is generated. You cannot edit the expressions of DATE, DATETIME or TIME variable filters; these can only be cleared and re-entered.

Filter Expression Syntax

The table shows the supported expression syntax.

Criteria	Expression	Examples
Equal to	EQ <input type="text" value="X"/> or EQ <input type="text" value="S"/>	Is equal to 100 (numeric) EQ 100 Is equal to "Blanco" (string) EQ "Blanco"

Criteria	Expression	Examples
Not equal to	NE <input type="text" value="x"/> or NE <input type="text" value="s"/>	<p>Is not equal to 100 (numeric)</p> <p>NE 100</p> <p>Is not equal to "Blanco" (string)</p> <p>NE "Blanco"</p>
Less than	LT <input type="text" value="x"/>	<p>Is less than 100</p> <p>LT 100</p>
Greater than	GT <input type="text" value="x"/>	<p>Is greater than 100</p> <p>GT 100</p>
Less than or equal to	LE <input type="text" value="x"/>	<p>Is less than or equal to 100</p> <p>LE 100</p>
Greater than or equal to	GE <input type="text" value="x"/>	<p>Is greater than or equal to 100</p> <p>GE 100</p>
Between (inclusive)	BETWEEN <input type="text" value="x"/> AND <input type="text" value="y"/>	<p>Is between 100 and 200</p> <p>BETWEEN 100 AND 200</p>
Not between (inclusive)	NOT BETWEEN <input type="text" value="x"/> AND <input type="text" value="y"/>	<p>Is not between 100 and 200</p> <p>BETWEEN 100 AND 200</p>
Is missing	IS MISSING	IS MISSING
Is not missing	IS NOT MISSING	IS NOT MISSING
In	IN (<input type="text" value="x"/> , <input type="text" value="y"/>) or IN (<input type="text" value="s1"/> , [<input type="text" value="s2"/>])	<p>Is one of the values 100, 200 or 300</p> <p>IN (100,200,300)</p> <p>Is one of the values "Blanco", "Jones" or "Smith"</p> <p>IN ("Blanco", "Jones", "Smith")</p>
Starts with	LIKE " <input type="text" value="s%"/>	<p>Starts with the string "Bla"</p> <p>LIKE "Bla%"</p>
Ends with	LIKE " <input type="text" value="%s"/>	<p>Ends with the string "nco"</p> <p>LIKE "%nco"</p>
Contains	LIKE " <input type="text" value="%s%"/>	<p>Contains the string "an"</p> <p>LIKE "%an%"</p>

Sorting a dataset

Sorting a dataset by one or many variables.

You can sort the contents of a dataset on several variables. Variables that are part of an active sort have an icon representing the direction of the sort in the header: ▲ for ascending sort and ▼ for descending sort. The size of the icon indicates the significance of the variable in the sort.

Sorting is not available if the dataset is open in *Edit* mode. If you attempt to switch to *Edit* mode while a sort is currently active, the entire dataset to be rewritten to match the current sort order, to allow it to be edited in-place.

To sort a dataset:

1. Right-click the column header for the variable that you want to sort by primarily.
2. Click **Ascending Sort** or **Descending Sort** on the shortcut menu to perform the sort.
3. If required, you can apply further sorting (secondary, tertiary etc) by selecting another column and clicking **Ascending Sort** or **Descending Sort**.

To remove any column from the sort, right-click the required column and click **Clear Sort**. The dataset will remove that sort and revert to any previous sorts, if present.

Working with program output

An overview of the various types of program output.

Output from programs is accessible from the following views:

- [WPS Server Explorer](#) (page 78) – Provides access to the datasets, catalogs, library references (for example, the default Work library), file references, and so on, generated for the particular server.
- [Output Explorer](#) (page 84) – Provides access to the logs and output generated by the Output Delivery System (ODS), such as the listing, HTML or PDF output formats.
- [Results Explorer](#) (page 85) – Provides access to the output results of procedures run as part of the SAS language program.
- [Outline](#) (page 83) – Provides access to the structural elements of the currently selected program, log, listing or HTML output.

Log output for a server in the SAS Language Environment is cumulative and shows information and errors from each program that has been run during the current session. For more information on logs, see [Log](#) (page 75).

Listing output is cumulative for a server and shows the results of every program that has been executed during the current session.

HTML and PDF output show the results of the programs, but neither are cumulative.

The WPS server itself can be restarted without restarting Workbench (see [Restarting a WPS Server](#) (page 48) for more information). This clears the log, listing and HTML output, and deletes temporary files library references, datasets and macro variables set up on the server.

Dataset generation

If your program produces or amends a dataset, then this can be viewed from the relevant library in WPS server explorer.

For example, each time you run the following sample program the associated dataset is updated in the *Work* library:

```
DATA longrun;
  DO I = 1 TO 5000000000;
    OUTPUT;
  END;
RUN;
```

To view the details associated with the dataset, in **WPS Server Explorer** view, expand the **Work** library under the **Local** server, right-click on the dataset name and click **Properties** on the shortcut menu.

For more information about the use of libraries on a server, see [Libraries and datasets](#) (page 122). For details about viewing and amending datasets, see [Dataset Viewer](#) (page 138).

Managing ODS output

Working with the Output Delivery System.

Once you have run a program that produces some output using the ODS (Output Delivery System), you can view it in the **Results Explorer** view.

The ODS *destinations* and the location of the generated output, can either be automatically managed by Workbench, or controlled through statements in the SAS language program.

Automatically manage ODS destinations

Managing ODS destinations for different result types.

Preferences to control the ODS destinations are automatically created when a SAS language program is run in Workbench. To output the ODS destinations in a program not running in Workbench, you will need to add the required ODS creation statements to your program.

Automatically Manage Result Types

Click **Yes** to create ODS destinations for each of the result types selected. Click **No** to manage the output within the SAS language program.

Result Types

Select the ODS destination to be automatically created. Workbench creates a temporary file for each destination type for each SAS language program run. When a different program is run, or the same program run again, new temporary files are created.

Show generated injected code in the log

Click to show the automatically created code for each ODS destination in the log output.

Listing output

This section of the guide looks at the  listing output of results.

Listing Colouring Preferences

Setting preferences for listing results.

The preferences that you can set regarding the different items contained in listing results are found under **Window > Preferences... > WPS > Listing Syntax Coloring**:

For each different type of item contained in a listing, you can specify attributes for **Colour**, **Background Colour**, and whether or not the item should be displayed in **Bold**.

Viewing the listing output

Viewing the **Output Explorer** listing output.

To view the listing output:

1. Ensure you have the  **Output Explorer** open (**Window > Show View >  Output Explorer**).
2. If there is more than one server, open the required server node to display the associated output.
3. Either double-click the  **Listing** node, or right-click it and select  **Open** from the context menu.

The listing output opens and is given focus.

If you have performed several runs, the previous listing output will be concatenated into the existing results. If you want to clear the output, to ensure that when you next run a program the listing only contains results for that run, then you can proceed in accordance with either of the following:

-  *Clearing the results output* [↗](#) (page 151) (which will also clear any *HTML output* [↗](#) (page 149)).
-  *Restarting a WPS Server* [↗](#) (page 48) (which will clear the log and all results, datasets, library and Filerefs).

Navigating the listing output

Navigating the output contained in the listing file.

If you have executed several programs, or several instances of the same program, and have not cleared the results for the particular server by either [Clearing the results output](#) (page 151) or [Restarting a WPS Server](#) (page 48), then each instance of listing output will have been appended to the existing listing file in order of execution.

To navigate the output contained in the listing file:

1. Open the results in accordance with [Viewing the listing output](#) (page 147).
2. Ensure that you have the [Outline](#) (page 83) view open (**Window** > **Show View** > **Outline**).
3. In this view you will see nested output for the various programs, with nodes for the different output elements. Click on any of these nodes and the corresponding section in the listing file will be displayed.

Saving the listing output to a file

Saving the **Output Explorer** listing output to a file.

To save the listing output to a file:

1. Ensure that you have the **Output Explorer** open (**Window** > **Show View** > **Output Explorer**).
2. Do one of the following:
 - Right-click on the **Listing** node, and, from the context menu, select **Save As...**
 - Open the listing in the editor (see [Viewing the listing output](#) (page 147)), and then right-click in the editor window and select **Save As...**
 - Open the listing in the editor and then select **File** > **Save As...** from the main menu.
3. Use the **Save As** dialog to save the file to the required location.

Printing the listing output

Printing the listing output for a server.

To print the listing output for a particular server:

1. Ensure that you have the **Output Explorer** open (**Window** > **Show View** > **Output Explorer**).
2. If there is more than one server, open the required server node to display the associated output.

3. Do one of the following:

- Right-click on the  **Listing** node, and, from the context menu, select  **Print Results...**
- Open the listing in the editor (see [Viewing the listing output](#) (page 147)), and then right-click in the editor window and either select  **Print...** or press **Ctrl+P** on Windows (**Cmd+P** on MacOS).
- Open the listing in the editor and then select **File** >  **Print...** from the main menu.

4. Use the **Print** dialog to send the results to your required printer.

HTML output

This section of the guide looks at the  HTML output of results.

Viewing the HTML output

Viewing the **Output Explorer** HTML output.

To view the HTML output:

1. Ensure you have the  **Output Explorer** open (**Window** > **Show View** >  **Output Explorer**).
2. If there is more than one server, open the required server node to display the associated output.
3. Either double-click the  **HTML** node, or right-click it and select  **Open** from the context menu.

The HTML output file opens and is given focus.

Navigating the HTML output

Navigating the HTML output.

If you opted to have the HTML output automatically managed by the Workbench (see [Automatically manage ODS destinations](#) (page 146)), then each run will create a new HTML file. However, if you have executed several programs, or several instances of the same program, and have not cleared the results for the particular server by either  [Clearing the results output](#) (page 151) or  [Restarting a WPS Server](#) (page 48), then the previous HTML output will still be available in the  [Outline](#) (page 83) view.

To navigate the output contained in the HTML file:

1. Open the results in accordance with [Viewing the HTML output](#) (page 149).
2. Ensure that you have the  [Outline](#) (page 83) view open (**Window** > **Show View** >  **Outline**).

3. In this view you will see nested output for the various programs, with nodes for the different output elements. Click on any of these nodes and the corresponding section in the HTML output will be displayed.

Printing the HTML output

Printing the **Output Explorer** HTML output.

To print the HTML output:

1. Ensure that you have the  **Output Explorer** open (**Window** > **Show View** >  **Output Explorer**).
2. If there is more than one server, open the required server node to display the associated output.
3. Do one of the following:
 - Right-click on the  **HTML** node, and, from the context menu, select  **Print Results...**
 - Open the HTML file in the editor (see [Viewing the HTML output](#) (page 149)), and then right-click in the editor window and either select  **Print...** or press **Ctrl+P** on Windows (**Cmd-P** on MacOS).
 - Open the HTML file in the editor and then select **File** >  **Print...** from the main menu.
4. Use the **Print** dialog to send the results to your required printer.

Note:

A single HTML output file is created for each run within the Workbench. However, you can only print the HTML output of one run at a time.

PDF output

This section of the guide looks at the PDF output of results.

To set up the automatic management of PDF output, refer to [Automatically manage ODS destinations](#) (page 146).

The default settings for PDF output are as specified in the `wps.cfg` file (see [Configuration files](#) (page 361)), and are as follows:

- `-BOTTOMMARGIN 1cm`
- `-LEFTMARGIN 1cm`
- `-RIGHTMARGIN 1cm`
- `-TOPMARGIN 1cm`
- `-ORIENTATION portrait`

The `PAPERSIZE` is determined automatically by the locale of Workbench (see *Processing Engine LOCALE and ENCODING settings* [↗](#) (page 36)). It can be reset using unmanaged output (see below).

The `startpage` option, which determines whether or not the output from each PROC starts on a new page, is not configurable inside Workbench. It is initially set to `STARTPAGE=yes` in the ODS software, which means that each PROC does start on a new page. The option can also be reset using unmanaged output (see below).

You create unmanaged output by entering code directly into your programs, as per the example below:

```
options orientation=landscape;
options papersize=A4;
options topmargin=0.75in;
options bottommargin=0.5in;
options leftmargin=0.5in;
options rightmargin=0.5in;
ods pdf
  file="body.pdf"
  startpage=no;
  /* Startpage=no means that new pages are not generated between PROCs */
  /* Insert PROCs to generate output */
ods_all_ close;
```

Note:

Relevant parts of the code can also be used as WPS code injections [↗](#) (page 116).

Clearing the results output

Clearing the results output for a server.

1. To save your output before clearing it, save your PDF elsewhere if you have been using the  PDF destination, or proceed as in *Saving the listing output to a file* [↗](#) (page 148).
2. Clear the output results for the required server by doing one of the following:
 - From the main menu, select **WPS** >  > **Clear Results** > **Local Server** (or substitute the name of your remote WPS server in place of **Local Server**).
 - Use the keyboard shortcut **Ctrl + Alt + O** to remove the results from the default server.

Note:

If you have more than one WPS server registered, then the above action will clear the results for the *Default Processing Engine* [↗](#) (page 34). If in doubt as to which is the default server, use the tooltip for the toolbar button  to show which server's results will be cleared.

- From the toolbar, click  to clear the results from the default server, or click the drop-down  next to this button to select a non-default server.
- Ensure that you have the  **Output Explorer** open (**Window** > **Show View** >  **Output Explorer**), right-click on the  listing output,  HTML output or  PDF output for the required server, and from the context menu, select  **Clear All Results**.

All output will now be blank.

Note:

The effects of this option are different from *Restarting a WPS Server* [↗](#) (page 48) in that you preserve the context of what you are currently doing, and do not lose your temporary work.

Text-based editing features

Text based features in Workbench that help you write and modify programs and other project files.

You can use these features with the *SAS Editor* [↗](#) (page 153) to help you write and modify programs and other project files. The same features are available with the *Text Editor* [↗](#) (page 154), with the exception of **Colour Coding of Language Elements**. These features are not available when managing files through the **File Explorer** view.

Left hand borders

The left hand borders are used to display different features:

- In the outer left grey border, you will see annotations.
- In the inner left white border, you will see the expand and collapse controls for blocks of related language items. The white border is also used to display quick difference indicators.

Annotations

These can consist of the following displayed in the outer left grey border of a file:

-  *Bookmark anchors* [↗](#) (page 81)
-  *Task markers* [↗](#) (page 82)
-  Search results

Quick difference indicators

A quick difference indicator in the inner left white border of a program shows that something has changed in that line of code since it was last saved:

-  Indicates that a change has been made to the line of code.

If you hover the mouse over this coloured block, a pop-up window will show you the code as it was prior to the change.

-  Indicates that there is a new, additional line of code.
-  Indicates the position where one or more lines of code have been deleted.

When you save a program, the quick difference indicators will be cleared from the margin.

Navigating between annotations and quick difference indicators

You can navigate between annotations within an individual program by using the **Navigate** menu options. The **Next** option moves to the next annotation, the **Previous** option moves to the previous annotation.

You can also use **Next Annotation** and **Previous Annotation** on the Workbench's main button bar to move between annotations and quick difference indicators.

Layout preferences

The display controls for a program, including the current line, foreground and background colours, and whether to display line numbers, are user defined. See *Text Editor Preferences* [↗](#) (page 155) for more details.

Colour Coding of Language Elements

The colours used to display language elements in a program can be controlled via the **Preferences**. See *WPS Syntax Colouring* [↗](#) (page 117) for more information.

Working with editors

In any Workbench perspective, the **Editor** window is always displayed, so that you can continue to write and modify files. This section describes the various ways of opening files in the **Editor** window with the different editors.

Only one file can be active at any one time. This is the file that currently has focus within the **Editor** window.

You can open programs using either the **Project Explorer** view or **File Explorer** view. The **Project Explorer** view offers more control than the **File Explorer** view, which determines the most suitable editor automatically, based on the file name extension of the file being opened.

Editing files within projects allows you to use local history to help you manage changes. See *Local history* [↗](#) (page 59) for more information

SAS Editor

The **SAS Editor** view provides features suitable for editing SAS language programs, such as syntax colouring.

Programs can be opened either via a project or directly from one of the available file systems. Only files with an extension *.wps* or *.sas* can be opened in this editor.

To open a program in the **SAS Editor** view:

1. Click the **Window** menu, click **Show View** and then click **Project Explorer**.

If you are using the **File Explorer** view, Click the **Window** menu, click **Show View** and then click **File Explorer**.

2. Navigate to the program you want to open and double-click on the program name.

Text Editor

The Workbench has a basic **Text Editor** view. You can open a program with this editor but the contents are treated as regular text file.

To open a file in the **Text Editor** view :

1. Click the **Window** menu, click **Show View** and then click **Project Explorer**.

If you are using the **File Explorer** view, Click the **Window** menu, click **Show View** and then click **File Explorer**.

2. Navigate to the file you want to open. If the file is a text file, then double-click on it. Otherwise, right-click on the file, and, from the context menu, select **Open With** >  **Text Editor**.

System Editor

If a file has an extension associated with an application installed on your computer (for example, *.doc* may be associated with Microsoft Word), the System Editor will launch that application with the file in a new window outside the Workbench. This type of editor is known as a System Editor.

To open a file with the System Editor:

1. Click the **Window** menu, click **Show View** and then click **Project Explorer**.

If you are using the **File Explorer** view, Click the **Window** menu, click **Show View** and then click **File Explorer**.

2. Navigate to the program you want to open and double-click on the program name.

In-Place Editor

The Workbench supports OLE (Object Linking and Embedding) document editing. For example, if you have a Microsoft Word document in your project and use the **In-Place Editor**, the Word document will open in the Workbench's *Editor* [↗](#) (page 78), with Microsoft Word's pull-down menu options being integrated into the menu bar of the view.

To open a file with the **In-Place Editor**:

1. Click the **Window** menu, click **Show View** and then click **Project Explorer**.
2. Right-click on the required file, click **Open With** and then click **In-Place Editor** on the shortcut menu.

If the file has an associated OLE editor, the file is opened in the **In-Place Editor** in the **Editor** view and the menus and options associated with the linked application are available.

Text Editor Preferences

Text editor preferences can be set from the main Workbench preferences window.

To edit text editor preferences:

1. Click the **Window** menu and then click **Preferences**.
2. In the **Preferences** window, expand the **General** group, expand the **Editors** group and select the **Text Editors** group.

You can use the preferences to determine how programs are displayed in the SAS language editor and the text editor. For example, to set the background colour, in the **Appearance colour options**, select **Background colour**, clear **System Default**, and then select the required **Colour**.

These preferences only apply to files opened within the **Project Explorer** view.

Some of the **Preferences** that can be configured are:

- **Undo history size** – The number of operations able to be undone.
- **Displayed tab width** – the width of the tab used in files
- **Show line numbers** – whether line numbers are displayed in the margin
- **Colours** – current line, foreground, background, and so on.

Closing files

Closing files from the **Editor** view.

Any file that is opened in the **Editor** view has a tab labelled with the file name. To stop displaying the file in the **Editor** view click the **File** menu and then click **Close**. Alternatively click  **Close** on the tab.

Jumping to a particular project location

Jumping to a particular location in a project file using a bookmarked location, a task, a line number, or the location of the last edit.

You can jump to a particular location in a project file:

To jump to a bookmarked location:

1. Open the  Bookmarks view (**Window > Show View >  Bookmarks**).
2. Either double-click on the required bookmark description, or right-click on it, and, from the context menu, select  **Go To**.
3. The relevant file will then be opened (if it was closed), and the bookmarked line will be highlighted (or else the first line in the file if the entire file was bookmarked).

To jump to a task:

1. Open the  Tasks view (**Window > Show View >  Tasks**).
2. Either double-click on the required task description, or right-click on it, and, from the context menu, select  **Go To**.
3. The relevant file will then be opened (if it was closed), and the line containing the task will be highlighted.

To jump to a particular line number:

1. Select **Navigate > Go to Line...**
2. Enter the required number in the **Go to Line** dialog.
3. Either press **Enter** on your keyboard, or click **OK**, to complete the operation and close the dialog.

To jump to the last edit:

1. Select  **Last Edit Location** from either the toolbar or **Navigate** menu.

The relevant file will be opened (if it was closed), the last line you edited will be highlighted, and the cursor will be placed at the end of the last piece of text you edited.

Navigation between multiple project files

Switching between different project files.

You can have as many projects open in the Workbench as you like and each of these projects can contain any number of programs.

You can open any number of programs and other files from the different projects in the Editor view (see *Editor*  (page 78)). For each program that is open, a corresponding  tab is displayed. An open program can be given focus in the **Editor** view by clicking on its tab.

Moving backwards and forwards between programs

The main menu options **Navigate >  Back** and **Navigate >  Forward** (or  and  on the toolbar) are analogous to the back and forward buttons on a web browser. However, unlike a web browser, when you close and reopen Workbench, it still has the navigational information.

-  **Back** navigates to the previous resource that was viewed in the **Editor** window.

-  **Forward** displays the program that was active prior to selection of the previous **Back** command.

Searching and replacing strings

Finding a text string in an open program and, if required, replacing that string with another.

You can find any text string in an open program, as follows:

1. Ensure that the program to be searched is open in the  **File Explorer** (**Window > Show View > File Explorer**).
2. Display the **Find/Replace** dialog by either selecting **Edit > Find/Replace...** from the menu, or pressing **CTRL + F** on your keyboard.

The **Find/Replace** dialog is displayed, which allows you to specify the term to be found, and, if required, the term with which it is to be replaced.

Note:

The default **Scope** is **All**, which means that the whole file is searched (you can amend this to **Selected lines** if required). The other options are those that you would expect to find in any search dialog and are self-explanatory.

3. Click **Find** to find the first instance of the word and then proceed as required, clicking **Find** to move to each new instance of the text string.

Note:

If you select **Close** to remove the dialog, then the string is remembered, and selecting **Edit > Find Next** and **Edit > Find Previous** will conduct the same search, forwards and then backwards, without re-opening the dialog.

Undoing and redoing your edits

Undoing and redoing changes to a project file.

There is a multiple undo facility for project file edits, as follows:

1. Either select **Edit >  Undo** from the menu, or press **CTRL + Z** on your keyboard.

The last change you made is undone.

2. Repeat the above step for each previous edit that you want to undo.

Note:

The number of undo actions that can be performed is determined by **Undo history size** in the Text Editor Preferences [🔗](#) (page 155).

Note:

You can also redo any undone edits on an incremental basis, by either selecting **Edit > 🖱️ Redo** from the menu, or pressing **CTRL + Y** on your keyboard, the requisite number of times.

Working in the Workflow Environment

The Workflow Environment is a graphical development environment with features for data mining, predictive modelling tasks, and a range of Machine Learning capabilities.

To open the Workflow Environment perspective, click on the **Workflow Environment** button at the top right of the screen: .

Creating a new Workflow

Creating a new Workflow in either the **Project Explorer** view or the **File Explorer** view.

A Workflow is a program sequence visually represented by connected, colour-coded graphical elements (blocks) through which the background code runs from start to end.

A Workflow enables collaboration between the project stakeholders; it provides a project framework so the projects can be easily replicated or audited and can be used as a framework to carry out projects in standardised manner.

Any datasets you need to use with your Workflow must be located on the same host as the Workflow Engine you will use to run the Workflow. If your Workflow requires a connection to a database, you must have the appropriate database client software installed on the same host as the Workflow Engine.

You can create a new Workflow in either the **Project Explorer** view or the **File Explorer** view.

- If you select **Project Explorer** view, the Workflow can be created in any folder in the current workspace, and the file is saved on your local workstation.
 - If you select **File Explorer** view, the Workflow can be created in any folder on the local workstation, or any remote hosts in which you have permission to create files.
1. Open the Workflow Environment perspective, by clicking on the **Workflow Environment** button at the top right of the screen: . This button can be made clearer by right-clicking on it and selecting **Show Text**; this displays the name of the button on the button itself: .
 2. Click the **File** menu, click **New** and click **Workflow**.
 3. In the Workflow pane, select the required folder for the new Workflow, specify a **File name** and click **Finish**.

Adding blocks to a Workflow

Adding blocks to a Workflow.

A Workflow visually represents a program's logical steps using blocks. The blocks include colours, icons, and labels to identify the block type. Blocks can be connected together to represent program steps and specific operations within each step.

Most Workflows begin by importing a dataset, for example by using a **Permanent Dataset** block.

To add a **Permanent Dataset** block to a Workflow:

1. Ensure that you are in the Workflow Environment.
2. Add the block to the Workflow canvas. This may be done in one of four ways:
 - Expand the block's group (**Import** in this case) from the palette on the left hand side, then click and drag the **Permanent Dataset** block on to the Workflow canvas.
 - Right-click the **Workflow Editor** view, in the shortcut menu click **Add**, click **Import**, and then click **Permanent Dataset**.
 - Double-click on the Workflow canvas, then either scroll to find the block or use the search facility.
 - In the case of the Permanent Dataset block, a WPD file may also be dragged and dropped onto the Workflow canvas from a Windows Explorer window. This also applies to CSV files and Excel files for their import blocks.
3. Right-click the **Permanent Dataset** block and click **Rename Block** in the shortcut menu.
4. In the **Rename** dialog box, enter a **Block label** to describe the dataset, for example *Base dataset* and click **OK**. The name entered is displayed as the label of the block on the **Workflow Editor** view.
5. Right-click the **Permanent Dataset** block and click **Configure** in the shortcut menu (or just double-click on the block).
6. In the **Configure Permanent Dataset** dialog box, select a file location, either **Workspace** or **External**, and click **Browse** to locate an existing WPD-format dataset.
7. Click **OK** to save the changes.

If the dataset is successfully loaded, the execution status in the **Output** port of the block is green:



You can then add other blocks to the editor and connect them together to create a Workflow.

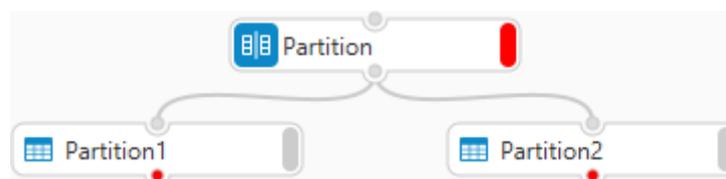
Connecting blocks in a Workflow

Blocks can be connected together using connectors to create a Workflow.

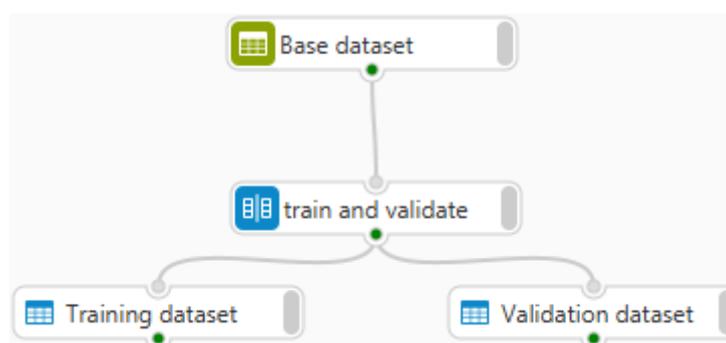
Connectors are linked to the block at either the **Input** port located at the top of a block, or an **Output** port located at the bottom of the block. You can connect the output from the **Output** port of one block to the **Input** port of one or more other blocks.

For example, you can randomly divide a dataset into two or more datasets using a **Partition** block. To partition the *Base dataset* created in the section *Add blocks to a Workflow*:

1. In the **Workflow Editor** view, click the **Data Preparation** group in the group palette.
2. Select the **Partition** block, and drag onto the working area. By default, two partitions are created and the **Partition** block has two working datasets:



3. Right-click the **Partition** block and click **Rename Block** in the shortcut menu. In the **Rename** dialog box, enter a **Block label** to describe the partition, for example *Train and validate* and click **OK**.
4. Click the **Base dataset** block **Output** port and drag towards the **Input** port of the **Train and validate** block. A connector appears as you drag. Drag this until it connects to the **Input** port.
5. When the **Base dataset** block is connected to the **Train and validate** block:
 - a. Select the **Partition1** dataset and change the **Block label** to *Training dataset*.
 - b. Select the **Partition2** dataset and change the **Block label** to *Validation dataset*.



More blocks can be connected to the output datasets following the same approach to create a Workflow that produces, for example, a credit risk scorecard.

Removing blocks from a Workflow

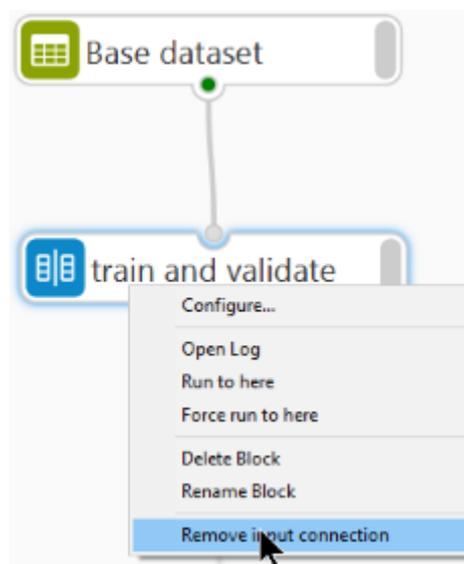
Blocks can be excluded from a Workflow path by deleting the connectors leading to the **Input** port of the block. Blocks can also be permanently deleted from a Workflow.

Connections between blocks can be removed from the block at either end of the connection using the shortcut menu for the block. Where multiple connectors are linked to the **Input** port of a block, the shortcut menu displays all connectors identified by the block label.

Connectors leading to the **Input** port of a working dataset cannot be removed.

To exclude the **train and validate** block from the Workflow:

1. In the **Workflow Editor** view, right-click the **train and validate** block.



2. Click **Remove input connection** in the shortcut menu.

Note:

To completely and permanently remove a block from a Workflow, right-click on it and select **Delete**.

Removing blocks enables you to alter the Workflow, for example to attach a different dataset to the **train and validate** partition block to create new training and validation datasets for use in the rest of the Workflow.

Deleting Workflow blocks

Blocks and any associated working datasets can be deleted from a Workflow. To delete a block, select the required block, right-click and click **Delete Block** in the shortcut menu.

Deleting a block deletes:

- Any associated working datasets created by the block.
- Any connectors leading to the **Input** port.
- Any connectors leading from the **Output** port of the block.
- If the block automatically creates working datasets, any connectors leading from the **Output** port for every working dataset created by the block.

To delete one or many blocks:

1. Select one or many blocks.
 - To select a single block, click on it. This will also select the block's output if applicable.
 - To select multiple blocks, hold down **Ctrl** and then either use the left mouse button to drag a box around the blocks you want to select, or click individually on each block. Keep **Ctrl** held down.
2. With the block or blocks selected, click the **Edit** drop-down menu and click **Delete**. You may also press **Delete** on the keyboard, or right-click and click **delete** from the pop-up menu, but note that this latter method does not work with multiple block selections.

Cutting, Copying and pasting blocks

Blocks can be cut, copied and pasted, either within the same Workflow or to another Workflow.

If a cut or copied block has options or variables configured which are specific to a source dataset, then when the block is pasted and connected to another dataset, if the datasets are similar then the Workflow Editor will attempt to retain those configured options or variables. It is always recommended that you check the configuration of a pasted block.

Copy and paste will only function on the block itself and not the connections to and from the block. To copy and paste the connections with a single block, use the duplicate function (see [Duplicating blocks](#) (page 164)).

Comments attached to blocks will be copied and pasted along with the block.

To cut or copy and then paste one or more Workflow blocks with any relevant connections:

1. Select one or many blocks.
 - To select a single block, click on it. This will also select the block's output if applicable.
 - To select multiple blocks, hold down **Ctrl** and then either use the left mouse button to drag a box around the blocks you want to select, or click individually on each block. Keep **Ctrl** held down.
2. With the block or blocks selected, click the **Edit** drop-down menu and click **Copy** if you wish to store the selection in the clipboard and retain the selection on the Workflow canvas, or **Cut** if you want to do this but remove the selection from the Workflow canvas. You may also press **Ctrl+C** on the keyboard for copy, or **Ctrl+X** for cut (these shortcuts may vary, depending on your operating system). Alternatively, right-click and click **copy** or **cut** from the pop-up menu, but note that this latter method does not work with multiple block selections.

3. Right-click where you want to paste the block and select **Paste**. You may also click the **Edit** drop-down window and click **Paste**, or use the keyboard shortcut **Ctrl+V**. If using **Ctrl+V**, the block will be pasted at the top left of the current Workflow canvas view.

Duplicating blocks

Individual blocks may be duplicated, which performs a copy and paste in one action and also preserves connections to and from the block.

To duplicate a block, right-click on the block and select **Duplicate**.

Undoing Workflow actions

Many actions in the Workflow Environment can be undone.

To undo an action, select **Undo** from the **Edit** menu, or press **Ctrl+Z** (this shortcut may vary, depending on your operating system). Actions that can be undone include, but are not limited to: creating blocks, deleting blocks, changing block preferences and setup, adding connections and typing words in some comment boxes and expression boxes. If you have a configuration window open, the **Edit** menu will not be accessible and pressing **Ctrl+Z** is the only undo method available.

Redoing Workflow actions

Any undone Workflow action may be redone, restoring the action that undo reversed.

To redo an action, select **Redo** from the **Edit** menu, or press **Ctrl+Y** (this shortcut may vary, depending on your operating system). If you have a configuration window open, the **Edit** menu will not be accessible and pressing **Ctrl+Y** is the only redo method available.

Note:

If you have undone changes, then once you edit the Workflow, you will lose the ability to redo those undone changes.

Deleting a working dataset

Working datasets cannot be deleted; to delete a working dataset you must either delete the block that created the dataset, or change the number of working datasets created by the block.

For example, to remove a partition working dataset created using the **Partition** block:

1. Right-click the **Partition** block and in the shortcut menu click **Configure**.
2. In the **Configure** dialog box, select the partition to be removed and click **Remove Partition**.

Note:

You cannot delete the default partitions created when you drag a **Partition** block onto the **Workflow Editor** view.

Workflow execution

Workflows can run automatically or manually, as specified in the Workflow preferences in the **Workflow** panel of the Workbench **Preferences** dialog box.

Automatically running Workflows

If a Workflow runs automatically, additions to the Workflow are evaluated when you connect new blocks to a Workflow. If you make any changes to blocks, the Workflow is automatically re-run for blocks downstream of the change. Individual blocks can have their automatic running disabled if required (for example, if a block is known to take a long time to run), by right-clicking on the block and deselecting **Auto Run**.

Manually running Workflows

If the Workflow is manually run, right-click on the Workflow canvas and select **Run**. This will run the Workflow, using any output from previously-run blocks in the Workflow.

Alternatively, you can manually run sections of the Workflow by right-clicking on a block and selecting:

- **Run to here** – evaluates the additions to the Workflow up to the block clicked on, but uses the output from any previously-run blocks in the Workflow.
- **Force run to here** – reruns the whole Workflow up to the block clicked on, evaluating all blocks in the Workflow and recreating all working datasets.

Database References

Workbench can store access details for databases (known as Database References), which can then be used in Workflows. The **Add Database** wizard guides you through the steps to create a new database reference to a local or remote database for subsequent access throughout Workflow.

The **Add Database** wizard can be accessed from a number of points in the **Workflow Environment**: The **Settings** tab, the **Database Explorer** view, the **Database Import** block and the **Database Export** block.

The path through the wizard is slightly different depending on which database you want to access:

- DB2: see *Adding a DB2 database reference* [↗](#) (page 166).
- MySQL: see *Adding a MySQL database reference* [↗](#) (page 167).
- Oracle: see *Adding an Oracle database reference* [↗](#) (page 168).
- ODBC: see *Adding an ODBC database reference* [↗](#) (page 168).
- PostgreSQL: see *Adding a PostgreSQL database reference* [↗](#) (page 169).
- SQL Server: see *Adding an SQL Server database reference* [↗](#) (page 170).
- Snowflake: see *Adding a Snowflake database reference* [↗](#) (page 171)
- AWS Redshift: see *Adding an AWS Redshift database reference* [↗](#) (page 171)
- MS Azure SQL Data Warehouse: see *Adding an MS Azure SQL Data Warehouse database reference* [↗](#) (page 172)

Adding a DB2 database reference

Using the **Add Database** wizard to add a new DB2 database reference.

Before adding a new reference to a DB2 database, you must ensure that a DB2 database client connector is installed and accessible from the Workflow Engine used to run the Workflow.

To create a new DB2 database reference, initiate the **Add Database** wizard and then follow these steps:

1. Select **DB2** from the **Database type** list.
2. Click **Next**.
3. Enter the **Database Name**.
4. Choose the authentication method from the **Authentication** drop-down list.
 - If you choose **Credentials**, then enter a **Username** and **Password**.
 - If you choose **Auth domain**, then choose an **Auth domain** from the drop-down list.
5. Click **Connect**.

If the connection is successful, a **Connection successful** message is displayed. Click **OK** to close this message and continue at the next step. If the connection is unsuccessful, a **Connection failed** message is displayed. Click **Details** to view details of the connection failure and **OK** to close this message and try the previous step again.

6. Click **Next**.
7. Choose the **Schema**.
8. Click **Next**.
9. In the **Name** box, enter a name to use to refer to the database in the Workflow Environment.
10. Click **Finish**.

Adding a MySQL database reference

Using the **Add Database** wizard to add a new MySQL database reference.

Before adding creating a new reference to a MySQL database, you must ensure that a MySQL database client connector is installed and accessible from the Workflow Engine used to run the Workflow.

To create a new MySQL database reference, initiate the **Add Database** wizard and then follow these steps:

1. Select **MySQL** from the **Database type** list.
2. Click **Next**.
3. Enter the **Host name**.
4. Enter the **Port**.
5. Choose the authentication method from the **Authentication** drop-down list.
 - If you choose **Credentials**, then enter a **Username** and **Password**.
 - If you choose **Auth domain**, then choose an **Auth domain** from the drop-down list.
6. To use SSL for the connection, select **Use SSL** and then specify all four of the fields beneath:
 - **SSL CA**: The file path and file name of the Certificate Authority (CA) file in PEM format.
 - **SSL CA path**: The path to a folder that contains SSL CA files in PEM format.
 - **SSL certificate**: The X.509 SSL certificate in PEM format. This will either be the client public key certificate or server public key certificate.
 - **SSL key**: the SSL private key file in PEM format. This will either be the client private key or server private key.
7. Click **Connect**.

If the connection is successful, a **Connection successful** message is displayed. Click **OK** to close this message and continue at the next step. If the connection is unsuccessful, a **Connection failed** message is displayed. Click **Details** to view details of the connection failure and **OK** to close this message and try the previous step again.
8. Choose the **Database** from the drop-down list.
9. Click **Next**.
10. In the **Name** box, enter a name to use to refer to the database in the Workflow Environment.
11. Click **Finish**.

Adding an Oracle database reference

Using the **Add Database** wizard to add a new Oracle database reference.

Before adding creating a new reference to an Oracle database, you must ensure that an Oracle database client connector is installed and accessible from the Workflow Engine used to run the Workflow.

To create a new Oracle database reference, initiate the **Add Database** wizard and then follow these steps:

1. Select **Oracle** from the **Database type** list.
2. Click **Next**.
3. Enter the **Net service** name.
4. Choose the authentication method from the **Authentication** drop-down list.
 - If you choose **Credentials**, then enter a **Username** and **Password**.
 - If you choose **Auth domain**, then choose an **Auth domain** from the drop-down list.
5. Click **Connect**.

If the connection is successful, a **Connection successful** message is displayed. Click **OK** to close this message and continue at the next step. If the connection is unsuccessful, a **Connection failed** message is displayed. Click **Details** to view details of the connection failure and **OK** to close this message and try the previous step again.

6. Click **Next**.
7. Choose the **Schema**.
8. Click **Next**.
9. In the **Name** box, enter a name to use to refer to the database in the Workflow Environment.
10. Click **Finish**.

Adding an ODBC database reference

Using the **Add Database** wizard to add a new ODBC database reference.

Before adding creating a new reference to an ODBC database, you must ensure that a ODBC database client connector is installed and accessible from the Workflow Engine used to run the Workflow.

To create a new ODBC database reference, initiate the **Add Database** wizard and then follow these steps:

1. Select **ODBC** from the **Database type** list.
2. Click **Next**.
3. Enter the **Data source name**.

4. Choose the **Authentication** method from the drop-down list:

- If you choose **Credentials**, then optionally enter a **Username** and **Password** (if required by the specified database).
- If you choose **Auth domain**, then choose an **Auth domain** from the drop-down list.
- **Use ODBC Credentials.**

5. Click **Connect**.

If the connection is successful, a **Connection successful** message is displayed. Click **OK** to close this message and continue at the next step. If the connection is unsuccessful, a **Connection failed** message is displayed. Click **Details** to view details of the connection failure and **OK** to close this message and try the previous step again.

6. Click **Next**.

7. Optionally, and if supported, choose the **Schema**.

8. Click **Next**.

9. In the **Name** box, enter a name to use to refer to the database in the Workflow Environment.

10. Click **Finish**.

Adding a PostgreSQL database reference

Using the **Add Database** wizard to add a new PostgreSQL database reference.

Before adding creating a new reference to a PostgreSQL database, you must ensure that a PostgreSQL database client connector is installed and accessible from the Workflow Engine used to run the Workflow.

To create a new PostgreSQL database reference, initiate the **Add Database** wizard and then follow these steps:

1. Select **PostgreSQL** from the **Database type** list.
2. Click **Next**.
3. Enter the **Host name**.
4. Enter the **Port**.
5. Choose the authentication method from the **Authentication** drop-down list.
 - If you choose **Credentials**, then enter a **Username** and **Password**.
 - If you choose **Auth domain**, then choose an **Auth domain** from the drop-down list.
6. Click **Connect**.

If the connection is successful, a **Connection successful** message is displayed. Click **OK** to close this message and continue at the next step. If the connection is unsuccessful, a **Connection failed** message is displayed. Click **Details** to view details of the connection failure and **OK** to close this message and try the previous step again.

7. Click **Next**.
8. Choose the **Database** from the drop-down list.
9. Click **Next**.
10. In the **Name** box, enter a name to use to refer to the database in the Workflow Environment.
11. Click **Finish**.

Adding an SQL Server database reference

Using the **Add Database** wizard to add a new SQL Server database reference.

Before adding creating a new reference to a SQL Server database, you must ensure that a SQL Server database client connector is installed and accessible from the Workflow Engine used to run the Workflow.

To create a new SQL Server database reference, initiate the **Add Database** wizard and then follow these steps:

1. Select **SQL Server** from the **Database type** list.
2. Click **Next**.
3. Enter the **Host name**.
4. Enter the **Port**.
5. Choose the authentication method from the **Authentication** drop-down list.
 - If you choose **Credentials**, then enter a **Username** and **Password**.
 - If you choose **Auth domain**, then choose an **Auth domain** from the drop-down list.
6. Click **Connect**.

If the connection is successful, a **Connection successful** message is displayed. Click **OK** to close this message and continue at the next step. If the connection is unsuccessful, a **Connection failed** message is displayed. Click **Details** to view details of the connection failure and **OK** to close this message and try the previous step again.

7. Click **Next**.
8. Choose the **Database** from the drop-down list.
9. Click **Next**.
10. In the **Name** box, enter a name to use to refer to the database in the Workflow Environment.
11. Click **Finish**.

Adding a Snowflake database reference

Using the **Add Database** wizard to add a new Snowflake database reference.

Before adding a new reference to a Snowflake database, you must ensure that a Snowflake database client connector is installed and accessible from the Workflow Engine used to run the Workflow.

To create a new Snowflake database reference, initiate the **Add Database** wizard and then follow these steps:

1. Select **Snowflake** from the **Database type** list.
2. Click **Next**.
3. Enter the **Server** and **Warehouse**.
4. Choose the authentication method from the **Authentication** drop-down list.
 - If you choose **Credentials**, then enter a **Username** and **Password**.
 - If you choose **Auth domain**, then choose an **Auth domain** from the drop-down list.
5. Click **Connect**.

If the connection is successful, a **Connection successful** message is displayed. Click **OK** to close this message and continue at the next step. If the connection is unsuccessful, a **Connection failed** message is displayed. Click **Details** to view details of the connection failure and **OK** to close this message and try the previous step again.

6. Click **Next**.
7. Choose the **Database** and **Schema** from the drop-down lists.
8. Click **Next**.
9. In the **Name** box, enter a name to use to refer to the database in the Workflow Environment.
10. Click **Finish**.

Adding an AWS Redshift database reference

Using the **Add Database** wizard to add a new AWS Redshift database reference.

Before adding a new reference to a AWS Redshift database, you must ensure that a AWS Redshift database client connector is installed and accessible from the Workflow Engine used to run the Workflow.

To create a new AWS Redshift database reference, initiate the **Add Database** wizard and then follow these steps:

1. Select **Amazon Redshift** from the **Database type** list.
2. Click **Next**.
3. Enter the **Host name**.

4. Enter the **Port**.
5. Enter the **Database**.
6. Choose the authentication method from the **Authentication** drop-down list.
 - If you choose **Credentials**, then enter a **Username** and **Password**.
 - If you choose **Auth domain**, then choose an **Auth domain** from the drop-down list.
7. Click **Connect**.

If the connection is successful, a **Connection successful** message is displayed. Click **OK** to close this message and continue at the next step. If the connection is unsuccessful, a **Connection failed** message is displayed. Click **Details** to view details of the connection failure and **OK** to close this message and try the previous step again.

8. Click **Next**.
9. Choose the **Schema** from the drop-down list.
10. Click **Next**.
11. In the **Name** box, enter a name to use to refer to the database in the Workflow Environment.
12. Click **Finish**.

Adding an MS Azure SQL Data Warehouse database reference

The **Add Database** wizard can be used to add a new MS Azure SQL Data Warehouse database reference.

Before adding a new reference to a MS Azure SQL Data Warehouse database, you must ensure that a MS Azure SQL Data Warehouse database client connector is installed and accessible from the Workflow Engine used to run the Workflow.

To create a new MS Azure SQL Data Warehouse database reference, initiate the **Add Database** wizard and then follow these steps:

1. Select **Sql Azure Warehouse** from the **Database type** list.
2. Click **Next**.
3. Enter the **Host name**.
4. Enter the **Port**.
5. Choose the authentication method from the **Authentication** drop-down list.
 - If you choose **Credentials**, then enter a **Username** and **Password**.
 - If you choose **Auth domain**, then choose an **Auth domain** from the drop-down list.

6. Click **Connect**.

If the connection is successful, a **Connection successful** message is displayed. Click **OK** to close this message and continue at the next step. If the connection is unsuccessful, a **Connection failed** message is displayed. Click **Details** to view details of the connection failure and **OK** to close this message and try the previous step again.

7. Click **Next**.

8. Choose the **Database** and **Schema** from the drop-down list.

9. Click **Next**.

10. In the **Name** box, enter a name to use to refer to the database in the Workflow Environment.

11. Click **Finish**.

Parameters

A Workflow can store global variables, known as parameters, which can be used throughout that Workflow in block configuration windows in place of user specified values. Parameters can represent open values that can be used in any character or numeric entry box, or specific values from a drop-down list. Parameters are saved with the Workflow they are created in and can only be used with that Workflow.

Parameters can be added either from the location in the user interface that you want the parameter to be used from, or from the **Settings** tab of a Workflow. Parameters associated with drop-down lists can only be added from their location in the user interface. The **Settings** tab is also used to manage parameters.

A parameter consists of:

- **Name**: Used to refer to the parameter when using or managing it.
- **Type**, which is one of:
 - **Character**: Alphanumeric and used in fields that accept a character input (such as names and labels).
 - **Number**: Numeric and used in fields that accept a numeric input.
 - **Boolean**: True or false.
 - A type specific to a particular drop-down list. For example, the type **Variable Treatment** in the **Decision Forest** block.
- **Value**: The value of the parameter.

Adding a parameter from its place of use

Adding a parameter from the location in the user interface that you want the parameter to be used. This is the only way to add a parameter for use with a drop-down list.

To add a parameter from its place of use:

1. Open an existing Workflow file in Workbench, or create a new one.
2. Locate the Workflow block configuration window that contains the entry box you want to use a parameter with. Click the small blue ring to the right of the box: .

3. Click **Add Binding**.

A **Bind Workflow Parameters** window appears.

4. Click **Add parameter** ().

A **Create Parameter** window opens.

5. Complete the **Create Parameter** window as follows:

- a. **Name:** Enter a name for the parameter. This will be used to refer to the parameter elsewhere in Workflow.

- b. **Type:** This will be greyed out if only one type is applicable. Otherwise, choose the type of the variable from:

- **Character:** Alphanumeric and used in fields that accept a character input (such as names and labels).
- **Number:** Numeric and used in fields that accept a numeric input.
- **Boolean:** True or false.

- c. **Value:** Enter the value you want the parameter to take. If you are setting a parameter for a drop-down list with set values, choose the value you want from the drop-down list.

6. Click **OK**.

7. At the **Bind Workflow Parameters** window, click **OK**.

Adding a parameter from the Workflow Settings tab

Adding a parameter from the **Settings** tab of a Workflow file.

To add a parameter for use with a text or numeric entry box:

1. Open an existing Workflow file in Workbench, or create a new one.
2. Click the **Settings** tab at the base of the screen.
3. From the left hand pane, click **Parameters** (.

4. Click **Add Parameter**.

A **Create Parameter** window opens.

5. Complete the **Create Parameter** window as follows:

a. Name: Enter a name for the parameter. This will be used to refer to the parameter elsewhere in Workflow.

b. Type: This will be greyed out if only one type is applicable. Otherwise, choose the type of the variable from:

- **Character:** Alphanumeric and used in fields that accept a character input (such as names and labels).
- **Number:** Numeric and used in fields that accept a numeric input.
- **Boolean:** True or false.

c. Value: Type the value that you want the parameter to have.

6. Click **OK**.

The new parameter appears in the list of parameters.

Viewing parameters

Viewing parameters from the **Settings** tab of a Workflow file.

To view a list of parameters:

1. Open an existing Workflow file in Workbench, or create a new one.
2. Click the **Settings** tab at the base of the screen.
3. From the left hand pane, click **Parameters** (⚙️).

A list of parameters is displayed.

Editing a parameter

Editing parameters using the **Settings** tab of a Workflow file.

Parameter names and values can be changed, but not parameter types (which could conflict with their usage throughout the interface).

To edit a parameter:

1. Open an existing Workflow file in Workbench, or create a new one.
2. Click the **Settings** tab at the base of the screen.
3. From the left hand pane, click **Parameters** (⚙️).

4. Select a Parameter and click **Edit Parameter**.

An **Edit Parameter** window opens.

5. Change the **Name** or **Value** of the Parameter as required.
6. Click **OK**.

The parameter has now been edited.

Deleting a parameter

Deleting parameters from the **Settings** tab of a Workflow file.

To delete a parameter:

1. Open an existing Workflow file in Workbench, or create a new one.
2. Click the **Settings** tab at the base of the screen.
3. From the left hand pane, click **Parameters** (⚙️).
4. Click the parameter that you want to delete.
5. Click **Remove Parameter**.

The parameter has now been deleted.

Binding a parameter to an option

Selected options throughout the Workflow Environment perspective can have *parameters* bound to them, so that they take on the value of that parameter. This allows the values of variables to be defined once in a central location, and therefore edited only once if necessary. Parameters are defined for each Workflow file, and can only be used within that file.

Before you begin, ensure that the Workflow file already has an appropriate parameter defined.

To bind an existing parameter to a variable:

1. Open an existing Workflow file in Workbench, or create a new one.
2. Locate an input box that is compatible with the parameters function, which can be identified by a small blue ring to the right of the box: .
3. Click the blue ring and then either:
 - Click a parameter from the displayed list of compatible parameters.
 - Click **Add Binding** to display a **Bind Workflow Parameter** window, where you can choose a **Parameter** and view its current **Value**. Once you have selected the parameter you want, click **OK**.

The chosen input box will now be greyed out and show the current value of the selected parameter. The blue ring will become a solid circle (). The chosen variable and parameter are now linked; so if the parameter is changed, the value of the chosen variable will take on this new value automatically.

Editing a parameter binding

Editing an existing parameter binding, assigning a different parameter to the variable in question.

To edit a parameter binding:

1. Open an existing Workflow file in Workbench, or create a new one.
2. Locate a bound variable, which can be identified by a solid blue circle to the right of an input box:



3. Click the blue circle and then either:
 - Click a parameter from the displayed list of compatible parameters.
 - Click **Edit Binding** to display a **Bind Workflow Parameter** window, where you can choose a **Parameter** and view its current **Value**. Once you have selected the parameter you want, click **OK**.

The parameter binding has now been edited.

Removing a parameter binding

Removing an existing parameter binding, returning the Workflow interface option to an editable box. When a parameter binding is removed, the last entered value for the option is restored.

To remove a parameter binding:

1. Open an existing Workflow file in Workbench, or create a new one.
2. Locate a bound variable, which can be identified by a solid blue circle to the right of an input box:



3. Click the blue circle and then click **Remove Binding**.

The parameter binding has now been removed.

Workflow block reference

This section is a guide to the blocks currently supported by the **Workflow Editor** view.

Blocks

Blocks are the individual items that make up a Workflow.

Blocks are used to represent:

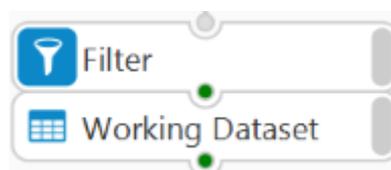
- Datasets, either imported into the Workflow or created by the Workflow.
- Programming code such as Python, R or the SAS language.
- Functionality that manipulates data to prepare a dataset for analysis.
- Modelling operations.

Blocks can be connected together to create a Workflow using connectors, or by docking blocks together. Blocks typically contain an **Output** port and an **Input** port. These ports enable you to link the output from one block to become the input to one or more other blocks. For example, **Permanent Dataset** blocks have an **Output** port located at the bottom of the block:



Block input and output ports

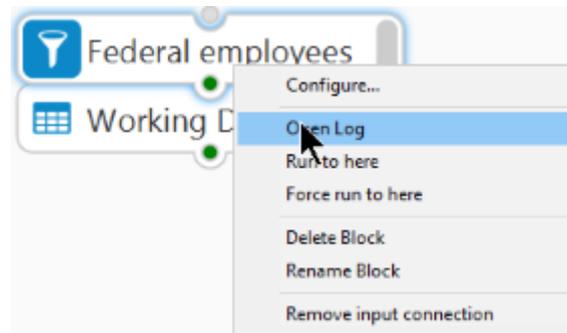
Blocks that manipulate a dataset have an **Input** port located at the top of a block, and an **Output** port located at the bottom of either the block, or the automatically-created output dataset of the block:



The execution status in the **Output** port of the block indicates the state of the block when the Workflow is run.

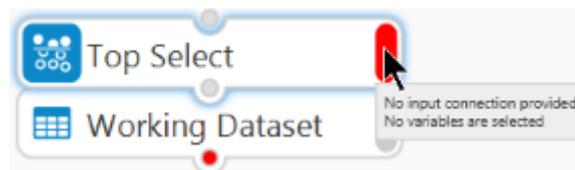
- If there are no errors and the block has run, the execution status is green.
- If the execution status is grey:
 - There is an error in a connected upstream block.
 - The block does not yet have an input.
 - The block has **Auto Run** disabled and a change has occurred.

- If the execution status is red there are errors in the block. You can review the error by reading the log output. Blocks that execute SAS language code each contain their own log file; to view such a log, right-click the block and click **Open Log** on the shortcut menu:



Block configuration status

The configuration status (the right-hand bar of a block) indicates whether all the required details have been specified for the block. If the configuration status is grey, the block is complete and can be used in a Workflow. If the configuration status is red, then some details are missing. In this event, hover over the bar to see what information is required to complete the block:



Block shortcut menu

You can Configure, modify and delete blocks functionality using the block's shortcut menu. To do this, right-click the required block and choose the appropriate option:

- **Copy Block** - Copy the block to the clipboard so it can be subsequently pasted elsewhere, either within the current Workflow or within another Workflow.
- **Delete Block** – remove the block from the Workflow. All connectors leading to and from the deleted block are also removed.
- **Rename Block** – enter a new display label for the block.
- **Remove output connection** – delete a connector linked to the **Output** port of the block. If there are multiple connections, you will see the option **Remove connection to**, which when highlighted opens a further menu to select the connection required.
- **Remove input connection** – delete a connector linked to the **Input** port of the block. If there are multiple connections, you will see the option **Remove connection from**, which when highlighted opens a further menu to select the connection required.
- **Configure** – specify the details of the block. For example, dataset location for **Import** blocks.
- **Configure Outputs**. Some blocks have multiple possible outputs; clicking this option enables you to choose which outputs you want the block to produce.

Some blocks have additions to the shortcut menu. For example, the **Permanent Dataset** block and working datasets have **Open** and **Open with** options that enable you to view the dataset content with either the **Dataset File viewer** or in the **Data Profiler** view. You specify which viewer opens when you click **Open** on the shortcut menu using the **File Associations** panel in the **Preferences** dialog box.

Where the block runs some SAS language code, for example the **Partition** block, the menu contains an **Open Log** option that enables you to view the log created by the Workflow Engine.

Block Workflow palette

The Workflow palette provides tools to import and prepare datasets for analysis, and to then use those datasets to create and train models for use in, for example, scoring applications.

Import group ↗	180
Contains blocks that enable you to import data into your Workflow.	
Data Preparation group ↗	186
Contains blocks that enable you to work with the datasets in your Workflow to create new, modified datasets.	
Code Blocks group ↗	234
Contains blocks that enable you to add a new program to the Workflow.	
Model Training group ↗	240
Contains blocks that enable you to discover predictive relationships in your data.	
Scoring group ↗	329
Contains blocks that enable you to analyse and score models.	
Export group ↗	333
Contains blocks that enable you to export data from a Workflow.	

Import group

Contains blocks that enable you to import data into your Workflow.

Database Import block ↗	181
Enables you to use a data source reference to connect to a relational database management system (RDBMS) to import datasets from tables and views into a Workflow.	
Excel Import block ↗	181
Enables you to import a worksheet from a Microsoft Excel workbook.	
Permanent Dataset block ↗	183
Enables you to specify a WPD-format dataset to import into the Workflow.	
Text File Import block ↗	184
Enables you to import a text file into the Workflow. The file may have any file extension.	

Database Import block

Enables you to use a data source reference to connect to a relational database management system (RDBMS) to import datasets from tables and views into a Workflow.

You can use the **Database Import** block to import multiple tables and views into a Workflow in a single step. The library reference used and available library connections stored with the Workflow are managed using the **Configure Database Import** dialog box.

To open the **Configure Database Import** dialog box, double-click the **Database Import** block. If no database is currently configured, an **Add Database** wizard will start to guide you through connecting to a database. For help using the wizard, see *Database References* [↗](#) (page 165).

Database

Specifies the source database used. If any databases have already been configured with Workflow, these are available to choose from in the drop-down list. To add another database, click the **Create a new database** button () to open an **Add Database** wizard. For help using the wizard, see *Database References* [↗](#) (page 165).

Tables

Displays the tables defined in the selected database.

Views

Displays the views defined in the selected database. Views are a table-like object displaying the data from one or more tables that meet the selection requirements of the view.

Excel Import block

Enables you to import a worksheet from a Microsoft Excel workbook.

The **Excel Import** block may be invoked in two ways: either as with any other block, or by dragging an Excel file onto the Workflow canvas. Once the **Excel Import** block is on the Workflow canvas, double-click it to open the **Configure Database Import** dialog box. If you choose to invoke the block by dragging an Excel file onto the Workflow canvas, before the block can be used you must first open the **Configure Database Import** dialog box and confirm its settings by clicking **OK**.

You can import data from a single sheet or a named range in a Workbook. Headers can be imported as the names of variables and the data types modified during import.

Options in the **Configure Database Import** dialog box are as follows:

File Location

Specifies where the dataset is located.

Workspace

Specifies the location of file containing the dataset is the current workspace.

Workspace datasets are only accessible when using the *Local Engine*.

External

Specifies the location of the file containing the dataset is on the file system accessible from the device running the Workflow Engine.

External files are accessible when using either a *Local Engine* or a *Remote Engine*.

Path

Specifies the path and file name for the file containing the required dataset. If you enter the **Path**:

- When importing a dataset from the Workspace, the root of the path is the Workspace. For example, to import a file from a project named `datasets`, the path is `/datasets/filename`.
- When importing a dataset from an external location, the path is the absolute (full) path to the file location.

The **Path** to the file is only valid with the Workflow Engine in use when the workflow is created. This path will need to be re-entered if the workflow is used with a different Workflow Engine.

If you do not know the path to the file, click **Browse** and navigate to the required dataset in the **Choose file** dialog box.

Data Format

Specifies the import range and formats of the variables in the working dataset.

Sheets

Import data from a sheet in the workbook. Available sheets are listed in the **Sheets** box.

Named Ranges

Import data from a named range on a sheet in the workbook. Available named-ranges are listed in the **Name** box.

Use first row as headers

Specifies the data contains column labels in the first row that are variable labels and not part of the imported data. The headers become the variable name and label displayed in the **Data Profiler** view.

Column Types

Specifies the type of variables imported. If required, select the column and modify the format. You might, for example, change a numeric column to a date in the working dataset.

Character

Specifies the column as a character type in the working dataset.

Numeric

Specifies the column as a numeric type in the working dataset.

Date

Specifies the column as a date type in the working dataset. The **Format** specified determines how the value is displayed in the working dataset:

- MMDDYY specifies the display format is MM/DD/YYYY.
- DDMMYY specifies the display format is DD/MM/YYYY.
- YYMMDD specifies the display format is YYYY-MM-DD.

🔗	183
🔗	183
🔗	183

1.

Permanent Dataset block

Enables you to specify a WPD-format dataset to import into the Workflow.

This block may be invoked by dragging a WPD file onto the Workflow canvas, then opening the block's configuration page and confirming the configuration.

The dataset can be imported from a project in the current workspace, or from a location on the file system. Importing a dataset is configured using the **Configure Permanent Dataset** dialog box.

To open the **Configure Permanent Dataset** dialog box, double-click the **Permanent Dataset** block.

File Location

Specifies where the dataset is stored on disk.

Workspace

Specifies the location of the dataset is the current workspace.

Workspace datasets can only be imported when using the *Local Engine*.

External

Specifies that the location of the dataset is on the file system accessible from the device running the Workflow Engine.

External files can be imported when using either a *Local Engine* or a *Remote Engine*.

Path

Specifies the path and file name for file containing the WPD-format dataset. If you enter the **Path**:

- When importing a dataset from the Workspace, the root of the path is the Workspace. For example, to import a file named `mydata.wpd` from a project named `datasets`, the path is `/datasets/mydata.wpd`.
- When importing a dataset from an external location, the path is the absolute (full) path to the file location.

The **Path** to the file is only valid for the Workflow Engine enabled when the Workflow is created. This path will need to be re-entered if the Workflow is used with a different Workflow Engine.

If you do not know the path to the file, click **Browse** and navigate to the required dataset in the **Choose file** dialog box.

Text File Import block

Enables you to import a text file into the Workflow. The file may have any file extension.

This block may be invoked by dragging a text file onto the Workflow canvas, then opening the block's configuration page and confirming the configuration.

Importing a dataset is configured using the **Configure Text File Import** dialog box. To open the **Configure Text File Import** dialog box, double-click the **Text File Import** block

File Location

Specifies where the text file is stored on disk.

Workspace

Specifies the location of the dataset is the current workspace.

Workspace datasets can only be imported when using the *Local Engine*.

External

Specifies that the location of the dataset is on the file system accessible from the device running the Workflow Engine.

External files can be imported when using either a *Local Engine* or a *Remote Engine*.

URL

Specifies that the location of the dataset is hosted on a web site accessible from the device running the Workflow Engine. Only files specified using the HTTP or HTTPS protocols can be imported.

External files can be imported when using either a *Local Engine* or a *Remote Engine*.

Path

Specifies the path to file containing the dataset.

- When importing a dataset from the Workspace, the root of the path is the Workspace. For example, to import a file named `mydata.csv` from a project named `datasets`, the path is `/datasets/mydata.csv`.
- When importing a dataset from an external location, the path is the absolute (full) path to the file location.

The **Path** to the file is only valid with the Workflow Engine in use when the Workflow is created. This path will need to be re-entered if the Workflow is used with a different Workflow Engine.

Alternatively, click **Browse** and navigate to the required dataset in the selection dialog box.

Data Format

Specifies the file delimiter and formats of the variables in the working dataset.

Delimiter

Specifies the character used to mark the boundary between variables in an observation of the imported dataset.

If the required delimiter is not listed, select `Other` and enter the character to use as the delimiter for the imported dataset.

Use first row as headers

Specifies the data contains column labels in the first row that are variable labels and not part of the imported data. The headers become the variable name and label displayed in the **Data Profiler** view.

Column Types

Specifies the type of variables imported. If required, select the column and modify the format to, for example, change a numeric column to a date in the working dataset.

Character

Specifies the column as a character type in the working dataset. The **Width** specified determines the maximum length of the character string in the working dataset.

Numeric

Specifies the column as a numeric type in the working dataset.

Date

Specifies the column as a date type in the working dataset. The **Format** specified determines how the value is displayed in the working dataset:

- `MMDDYY` specifies the display format is `MM/DD/YYYY`.
- `DDMMYY` specifies the display format is `DD/MM/YYYY`.
- `YYMMDD` specifies the display format is `YYYY-MM-DD`.

Data Preparation group

Contains blocks that enable you to work with the datasets in your Workflow to create new, modified datasets.

Aggregate block ↗	187
Enables you to apply a function to create a single value from a set of variable values grouped together using other variables in the input dataset.	
Binning block ↗	189
Enables you to bin a variable in a dataset. This block can then output a dataset including a new variable showing a bin allocation for each observation. The block can also be configured to output a binning model.	
Filter block ↗	191
Enables you to reduce the total number of observations in a large dataset by selecting observations based on the value of one or more variables.	
Impute block ↗	198
Enables you to assign or calculate values to fill in missing values in a variable.	
Join block ↗	204
Enables you to combine observations from two datasets into a single working dataset.	
Merge block ↗	208
Enables you to combine two datasets into a single working dataset.	
Mutate block ↗	209
Enables you to modify or add new variables to the working dataset. These variables can be independent of, or derived from variables in the existing dataset.	
Partition block ↗	212
Enables you to divide the observations in a dataset into two or more working datasets, where each working dataset contains a random selection of observations from the input dataset with a specified weighting for the split.	
Query block ↗	213
Enables you to create SQL code to join and interrogate database tables or datasets.	
Rank block ↗	224
Enables you to rank observations in an input dataset using one or more numeric variables.	
Sampling block ↗	229
Enables you to create a working dataset that contains a small representative sample of observations in the input dataset.	
Select block ↗	230
Enables you to create a new dataset containing specific variables from an input dataset.	
Sort block ↗	230
Enables you to sort a dataset.	
Top Select block ↗	231
Enables you to create a working dataset containing a dependent variable and its most influential independent variables.	

Transpose block [↗](#)..... 232
 Enables you to create a new dataset with transposed variables.

Aggregate block

Enables you to apply a function to create a single value from a set of variable values grouped together using other variables in the input dataset.

Block Overview

The aggregation of values is configured using the **Configure Aggregate** dialog box. To open the **Configure Aggregate** dialog box, double-click the **Aggregate** block.

The **Configure Aggregate** dialog box has two tabs:

- Expressions
- Grouping Variable Selection

Expressions

Variable

Specifies the variable to which the aggregation function is applied.

Function

Specifies the aggregation function to use. Before the function is applied, the dataset is collected into groups, and the function is applied to the values in the specified variable in the required grouping. Only functions that apply to the specified aggregation variable are displayed.

- **Average:** Returns the arithmetic mean of the specified variable. This function can only be used with numeric values.
- **Count:** Returns the number of occurrences of a numeric value or string in the specified variable.
- **Count(*):** Returns the number of observations in a dataset.
- **Count (Distinct):** Returns the number of unique occurrences of a numeric value or string in the specified variable.
- **Minimum:** Returns the minimum value in the input dataset for the specified variable. For character values, the returned value is the lowest value when the values are put in lexicographical order.
- **Maximum:** Returns the maximum value in the input dataset for the specified variable. For character values, the returned value is the highest value when the values are put in lexicographical order.
- **Sum:** Returns the total of all values in the specified variable. This function can only be used with numeric values.
- **Number of missings:** Returns the number of missing values in the specified variable.

- **Range (maximum - minimum):** Returns the difference between Maximum and Minimum.
- **Standard deviation:** Returns the standard deviation for values in the specified variable.
- **Standard error:** Returns the standard error for values in the specified variable.
- **Variance:** Returns the variance for values in the specified variable.
- **Skewness:** Returns the skewness for values in the specified variable.
- **Kurtosis:** Returns the kurtosis for values in the specified variable.
- **1st Percentile:** Returns a value equivalent to the 1st percentile of the specified variable.
- **5th Percentile:** Returns a value equivalent to the 5th percentile of the specified variable.
- **10th Percentile:** Returns a value equivalent to the 10th percentile of the specified variable.
- **25th Percentile / Q1:** Returns a value equivalent to the 25th percentile of the specified variable.
- **75th Percentile / Q3:** Returns a value equivalent to the 75th percentile of the specified variable.
- **90th Percentile:** Returns a value equivalent to the 90th percentile of the specified variable.
- **95th Percentile:** Returns a value equivalent to the 95th percentile of the specified variable.
- **99th Percentile:** Returns a value equivalent to the 99th percentile of the specified variable.
- **Median:** Returns the median for values in the specified variable.
- **Mode:** Returns the mode for values in the specified variable.
- **Corrected sum of squares:** Returns the corrected sum of squares for values in the specified variable.
- **Coefficient of variation:** Returns the coefficient of variation for values in the specified variable.
- **Lower confidence limit:** Returns the lower confidence limit for values in the specified variable.
- **Upper confidence limit:** Returns the upper confidence limit for values in the specified variable.
- **Two tailed p-value for Student's t statistic:** Returns the two tailed p-value for the Student's t statistic for values in the specified variable.
- **Quartile range (Q3 - Q1):** Returns the difference between the values equivalent to the Q3/75th and Q1/25th percentiles for the specified variable.
- **Student's t statistic for zero mean:** Returns the value of the student's t statistic for a zero mean for the specified variable.
- **Uncorrected sum of squares:** Returns the value of the uncorrected sum of squares for the specified variable.

New Variable

Specifies the name for the aggregated variable in the output dataset.

Grouping Variable Selection

Specifies the variables to use when grouping observations in the dataset.

Two lists are presented: **Unselected Grouping Variables** showing variables available for grouping, and **Selected Grouping Variables** showing variables selected for grouping. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Binning block

Enables you to bin a variable in a dataset. This block can then output a dataset including a new variable showing a bin allocation for each observation. The block can also be configured to output a binning model.

Block Overview

If the binning type is optimal, then you can set values that effect optimal binning. You can view the result of the binning, and adjust the settings if necessary. The block creates a working dataset in which a new variable is created for each observation. This variable contains the value for the bin into which the observation is placed.

The aggregation of values is configured using the **Configure Binning** dialog box. To open the **Configure Binning** dialog box, double-click the **Binning** block.

Configure Binning dialog

Use the **Configure Binning** dialog to specify the details required by block.

Binning variables

Specifies the variable or variables to be binned. The upper list of **Unselected Variables** shows variables that can be specified for binning and the lower list of **Selected Variables** shows variables that have been specified for binning. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Binning type

Specifies the type of binning. The available types depend on the format of the variable. Click on the box to display a list of types, and then select a type. There might only be one type listed. The types of binning that can be selected are:

Optimal.

Bins are created using optimal binning.

Equal Height

Bins are created using equal height binning.

Equal Width

Bins are created using equal width binning.

Winsorised

Bins are created after the data has been Winsorised.

Optimal Binning

The controls in this group are only displayed if you select Optimal as the binning type for a variable.

Dependent variable

Specifies the dependent variable used for optimal binning. Click the box to display a list variables, and then select the required variable.

Variable treatment

Specifies how the variable should be treated by the binning process. The supported treatments are:

Binary

Specifies a dependent variable that can take one of two values.

Nominal

Specifies a discrete dependent variable with no implicit ordering.

Ordinal

Specifies a discrete dependent variable with an implicit category ordering.

Creating the bins

When you have selected the variable to be binned and specified the parameters for optimal binning, if required, you can create the bins. To do this, click **Bin Variables**. The bins are then created and displayed in the **View Bins** panel. The bins are created using values specified in the Preferences dialog. See the section **Binning** panel [↗](#) (page 351) for details.

The list box underneath the control lists the bins created by the binning process. These bins might be adequate, in which case you can click **OK**. The **Configure Binning** dialog is then closed, and a working dataset containing the binning variable is created.

If you decide the bins are not adequate, you can change the binning preferences. To do this, click **Binning Preferences** (). This opens the **Binning** panel in **Workflow** in **Preferences**. See the section **Binning** panel [↗](#) (page 351) for details.

View Bins

The **View Bins** panel displays the bins you have created.

To split a bin, right-click on a bin heading and select **Split**. After performing a split, you can selectively combine the resulting items back into a bin by selecting the required items (press **Shift** and click for a contiguous list, or hold **Ctrl** and click to create a non-contiguous list), then right-clicking and selecting **Combine**.

To remove items from a bin, select them as required (press **Shift** and click for a contiguous list, or hold **Ctrl** and click to create a non-contiguous list), then right-click and select **Remove**.

Binning Output

Output types available for the **Binning** block are:

- **Working dataset:** The same dataset as input, but with an additional variable showing a bin allocation for each observation.
- **Binning Model.**

To specify the outputs, right-click on the **Binning** block and select **Configure Outputs**. Two boxes are presented: **Unselected Output**, showing outputs that are not selected, and **Selected Output**, showing selected outputs. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Filter block

Enables you to reduce the total number of observations in a large dataset by selecting observations based on the value of one or more variables.

When you filter a dataset using the **Filter** block, the input dataset remains unchanged and a working dataset is created containing only the observations selected with the filter.

Filtering datasets

To filter datasets:

1. Drag the **Filter** block onto the Workflow canvas, and connect the required dataset to the **Input** port of the block.
2. Double-click the **Filter** block.

The **Filter Editor** view displays and contains two tabs:

- The **Basic** tab that enables you to create a filter that is either a logical conjunction (logical AND) or a logical disjunction (logical OR) of expressions.
- The **Advanced** tab that enables you to create a complex filter statements that combines both the logical conjunction and logical disjunction of expressions.

3. Using either the **Basic** or **Advanced** filter tabs, specify a filter for the input dataset.

Basic filter

Creating a single filter that is either a logical OR expression or a logical AND expression.

The **Basic** view cannot be used to create a filter containing a combination of logical OR and logical AND expressions.

Create a filter expression

To create a filter expression, click **Add expression**  to create a child node, and complete the **Expression Properties** for the node. When a filter contains two or more expressions, select **AND** for a logical AND filter, or select **OR** for a logical OR filter.

Expression settings

Expressions define individual nodes as part of a larger filter.

Variable

The variable in the input dataset to be filtered.

Operator

Specifies the operation used to select observations. The available operators are determined by the variable type.

The following operators are available for character data variables.

Equal to	Includes observations in the working dataset if the value of the variable is equal to the specified Value specified.
Not equal to	Includes observations in the working dataset if the value of the variable is not equal to the specified Value .
In	Includes observations in the working dataset if the value of the variable is contained in the defined list of values. The list is specified in the Edit dialog box accessed from the Configure Filter dialog box.
Starts with	Includes observations in the working dataset if the value of the variable starts with the specified Value .
Ends with	Includes observations in the working dataset if the value of the variable ends with the specified Value .

The following filter operators are available for numeric data variables.

=	Includes observations in the working dataset if the value of the variable is equal to the specified Value specified.
!=	Includes observations in the working dataset if the value of the variable is not equal to the specified Value .

In	Includes observations in the working dataset if the value of the variable is contained in the defined list of values. The list is specified in the Edit dialog box accessed from the Configure Filter dialog box.
<	Includes observations in the working dataset if the value of the variable is less than the specified Value .
<=	Includes observations in the working dataset if the value of the variable is less than, or equal to, the specified Value .
>	Includes observations in the working dataset if the value of the variable is greater than the specified Value .
>=	Includes observations in the working dataset if the value of the variable is greater than, or equal to, the specified Value .

Value

The variable value used as the test in the filter. All character data entered in the **Value** box is case-sensitive.

Drawing Connections

By clicking and dragging from node to node you can draw connections between existing filter blocks. For example, you can take the output of two blocks and combine into another block for further filtering.

Advanced filter

Creating a complex filter statement using *simple* or *expression* nodes.

Overview

A filter is a sequence visually represented by connected nodes in a similar manner to a Workflow. The filter path for each branch represents a logical AND expression sequence. Each branch from dataset root are joined together as a logical OR sequence in the filter.

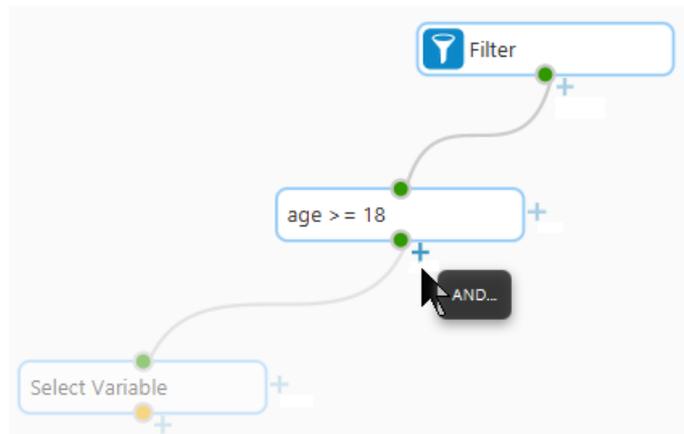
The **Advanced** filter contains the following:

- The filter canvas, which contains a visual representation of the filter.
- A box across the bottom of the filter canvas, which contains the version of the filter that can be copied to a SAS language program for testing or production use.
- A box on the right, which enables you specify the properties for a node in the filter in two forms:
 - **Simple** node: A simple statement containing a variable, an operator and a value. For example:
age >= 65.
 - **Expression** node: A SQL language expression.

Creating a filter node

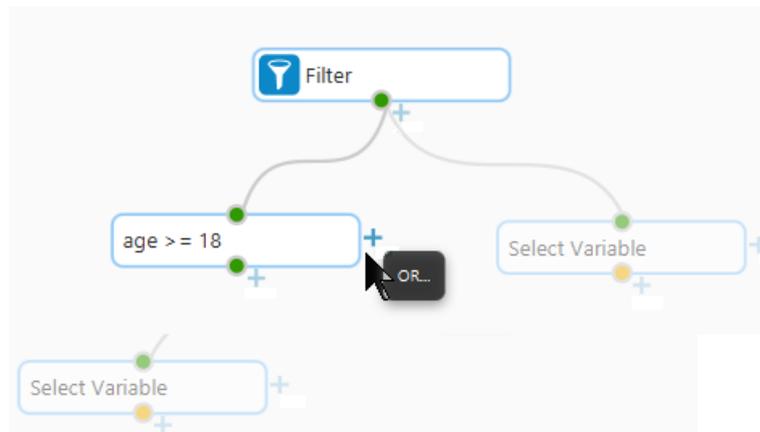
You can create complex filters by connecting the ports on the nodes in the filter canvas.

To add a logical AND expression, hover over the plus icon (+) below the node. The location of the new node is shown as a child of the current node on the filter canvas:



If the node is being added in the correct location, click the plus icon (+) below the node and complete the settings for the node.

To add a logical OR node, hover over the plus icon (+) to the right of the node. The location of the new node is shown as a child of the parent node on the filter canvas:



If the node is being added in the correct location, click the plus icon (+) icon to the right of the node and complete the settings for the node.

Drawing Connections

By clicking and dragging from node to node you can draw connections between existing filter blocks. For example, you can take the output of two blocks and combine into another block for further filtering.

Advanced filter simple nodes

Filters data with simple statements that contain a variable, an operator and a value.

Editing the filter node

To edit the filter node's properties, click the node and modify the properties for that node in the **Edit Node** section.

To delete the expression either click the  icon in the **Edit Node** box, or right-click the node in the filter canvas and click **Delete expression**.

Edit Node Settings

If the **Node Type** is set to **Simple**, the node is defined by three parameters as follows:

Variable

The variable in the input dataset to be filtered.

Operator

Specifies the operation used to select observations. The available operators are determined by the variable type.

The following operators are available for character data variables.

Equal to	Includes observations in the working dataset if the value of the variable is equal to the specified Value specified.
Not equal to	Includes observations in the working dataset if the value of the variable is not equal to the specified Value .
In	Includes observations in the working dataset if the value of the variable is contained in the defined list of values. The list is specified in the Edit dialog box accessed from the Configure Filter dialog box.
Starts with	Includes observations in the working dataset if the value of the variable starts with the specified Value .
Ends with	Includes observations in the working dataset if the value of the variable ends with the specified Value .

The following filter operators are available for numeric data variables.

=	Includes observations in the working dataset if the value of the variable is equal to the specified Value specified.
!=	Includes observations in the working dataset if the value of the variable is not equal to the specified Value .
In	Includes observations in the working dataset if the value of the variable is contained in the defined list of values. The list is specified in the Edit dialog box accessed from the Configure Filter dialog box.

- < Includes observations in the working dataset if the value of the variable is less than the specified **Value**.
- <= Includes observations in the working dataset if the value of the variable is less than, or equal to, the specified **Value**.
- > Includes observations in the working dataset if the value of the variable is greater than the specified **Value**.
- >= Includes observations in the working dataset if the value of the variable is greater than, or equal to, the specified **Value**.

Value

The variable value used as the test in the filter. All character data entered in the **Value** box is case-sensitive.

Example

Node Type

- Simple
- Expression

Edit Node

Variable: workclass

Operator: Equal to

Value: pt

```
(age <= 25 AND workclass = pt)
OR
(age >= 65 AND workclass = pt)
```

In the above example, each path is a different filter for the dataset. The alternate paths are joined at the point closest to the dataset root. Expressions can be shared between filter paths:

```
age <= 25 AND workclass='pt'
OR
age >= 25 AND workclass='pt'
```

Advanced filter expression nodes

Filters data with SQL language statements. Each node is an SQL SELECT WHERE expression.

Editing the filter node

To edit the filter node's properties, click the node and modify the properties for that node in the **Edit Node** section.

To delete the expression either click the icon in the **Edit Node** box, or right-click the node in the filter canvas and click **Delete expression**.

It is possible to create a node initially as a **Simple** node and then switch to **Expression** for more advanced editing.

Edit Node Settings

If the **Node Type** is set to **Expression**, the node is defined by an SQL expression entered in the provided box.

Click **Open in Editor** to open the **Filter Expression Builder** window. At the top of the window is a text box where an SQL expression can be entered manually. Alternatively, an expression can be created by double-clicking on or dragging the input variables, functions and operators provided below the text box. When typing manually, press **Ctrl-Spacebar** to show a list of suggested terms.

Example

In the above example, census data is filtered into two streams: one where `education` contains the strings `nth`, `st`, `nd` or `rd`, capturing school grades reached (1st, 2nd, 3rd, etc); and another where `education` contains `"grad"`, capturing graduate status.

Impute block

Enables you to assign or calculate values to fill in missing values in a variable.

The available methods for replacing missing values are determined by variable type and are specified in the **Configure Impute** dialog box. You can define expressions to replace missing values in all variables in a single **Impute** block. To add expressions to the block click **Add Expression**  and complete the information for the expression.

Imputation of missing values is configured using the **Configure Impute** dialog box. To open the **Configure Impute** dialog box, double-click the **Impute** block.

Configure Impute

Variable

Specifies a variable in the dataset with missing values. **Variable** is pre-populated with variables in the dataset

Method

Specifies the impute method used for missing values in the specified variable.

Character variables

The following impute methods are available for character variables.

Constant

Replaces missing values for the variable with the specified value. Specify the replacement in **Value**.

Distribution

Replaces missing values in the specified variable with randomly selected values present elsewhere in the variable. The replacement values are selected based on their probability of occurrence in the input dataset.

Mode

Replaces missing values in the specified variable with the mode of the specified variable.

Numeric variables

The following imputation methods are available for numeric variables. In the SAS language, numeric variables include all date-, datetime- and time-formatted variables.

Gaussian simulation

Replaces missing values in the specified variable with a random number from a Normal distribution based on the mean and standard deviation of that distribution. The mean and standard deviation are calculated from the input dataset and displayed in the **Mean** and **Standard deviation** fields respectively.

You can specify different mean and standard deviation values in the fields, but altering the values too far from those calculated might increase the amount of noise in the data.

If you choose a *Gaussian simulation* imputation method and output a model from the **Impute** block, then the **Mean** and **Standard deviation** values used will be from the original dataset connected to the **Impute** block or specified in the original imputation model, not from the new dataset that the model is applied to using the **Score** block.

Max

Replaces missing values in the specified variable with the maximum value of the specified variable.

Mean

Replaces missing values in the specified variable with the mean of the specified variable.

Median

Replaces missing values in the specified variable with the median of the specified variable.

Min

Replaces missing values in the specified variable with the minimum value of the specified variable.

Constant

Replaces missing values for the variable with a specified value. Specify the replacement in **Value**.

Distribution

Replaces missing values in the specified variable with randomly-selected values present in the variable. Replacement values are selected based on the frequency with which they occur in the input dataset.

Mode

Replaces missing values in the specified variable with the mode of the specified variable.

Trimmed mean

Replaces missing values in the specified variable with the mean value calculated, after removing the specified proportion of largest and smallest values. For example, if you specify a proportion of 0.25, the smallest and largest quarters of values are removed from the calculation. The interquartile mean value replaces all missing values in the variable.

The percentage of values removed from the calculation is specified in the **Percentage** field.

Uniform simulation

Replaces missing values in the specified variable with a random number from a Uniform distribution. By default, the distribution range is based on the minimum and maximum values for the specified variable, displayed in the **Min** field and **Max** field.

Only values within the specified boundaries are used to replace missing values in the variable, not the values specified in **Min** and **Max**.

Winsorized mean

Replaces missing values in the specified variable with the calculated Winsorized mean value, after replacing the specified proportion of largest and smallest values. For example, if you specify 0.05, the largest 5% of values are replaced with the value of the 95th percentile; the lowest 5% of values are replaced with the value of the fifth percentile; and all non-missing values in the specified variable are used to calculate the Winsorized mean.

The percentage of values replaced before the calculation occurs is specified in the **Percentage** field.

Seed

Specifies whether the same set of imputed values is generated each time the **Impute** block is executed.

- To generate the same sequence of observations, set **Seed** to a positive value greater than or equal to 1.
- To generate a different sequence of observations, set **Seed** to 0 (zero).

Block Outputs

The **Impute** block can generate one or both of the following outputs:

- **Working Dataset**: A new dataset containing the imputed variables.
- **Impute Model**: The imputation model, suitable for use with a **Score** block.

To specify the outputs, right-click on the **Impute** block and select **Configure Outputs**. Two boxes are presented: **Unselected Output**, showing outputs that are not selected, and **Selected Output**, showing selected outputs. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Impute block basic example: imputing missing values

This example demonstrates how the **Impute** block can be used to fill in missing values in a dataset.

In this example, an incomplete dataset of books named `LIB_BOOKS` is imported into a blank Workflow. The dataset is incomplete due to some missing values in its *Price* variable. The **Impute** block is used to fill in the missing values and output a complete dataset named `BOOKS COMPLETE`.

Workflow Layout



Dataset used

The file used in this example can be found in the samples distributed with WPS Analytics.

lib_books.csv dataset

This example uses a dataset of books, named `lib_books.csv`. Each observation in the dataset describes a book, with variables such as *Title*, *ISBN*, *Author* etc.

Using the Impute block to impute missing values

Using the **Impute** block to impute missing values in a dataset.

1. Import the `lib_books.csv` dataset onto a blank Workflow canvas. Importing a CSV file is described in [Text File Import block](#) (page 184).
2. Expand the **Data Preparation** group in the Workflow palette, then click and drag an **Impute** block onto the Workflow canvas, beneath the `lib_books.csv` dataset.
3. Click on the `lib_books.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Impute** block.

Alternatively, you can dock the **Impute** block onto the dataset block directly.

4. Double-click on the **Impute** block.

A **Configure Impute** dialog box opens.

5. From the **Configure Impute** dialog box:
 - a. From the **Variable** drop-down list, select **Price**.
 - b. From the **Method** drop-down list, select **Distribution**.
 - c. Click **OK**.

The **Configure Impute** dialog box closes. A green execution status is displayed in the **Output** ports of the **Impute** block and of the new dataset **Working Dataset**.

6. Right-click on the new dataset **Working Dataset**, click **Rename** and type `books_complete`.

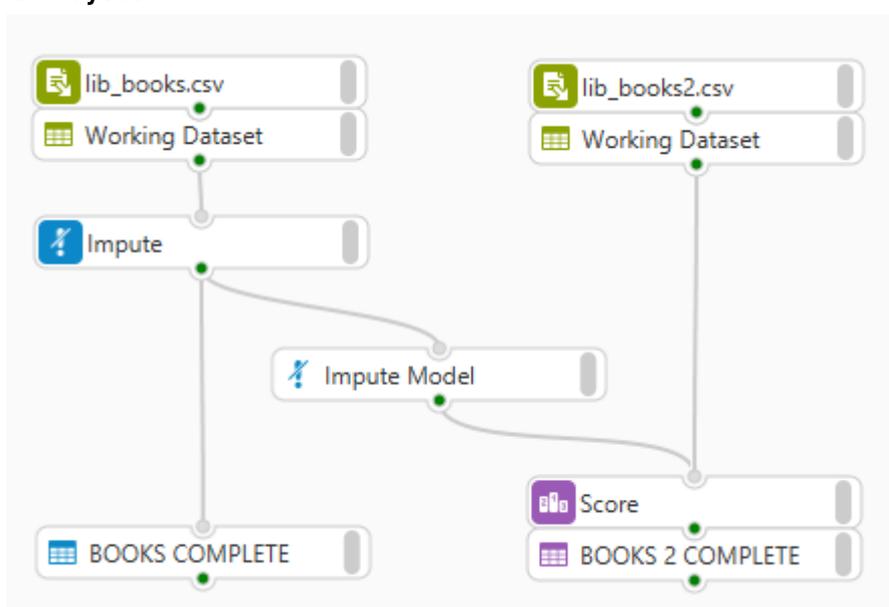
You now have a new dataset, `BOOKS COMPLETE`, with no missing values in the variable `Price`.

Impute block example: using the model output

This example demonstrates how the **Impute** block's optional model output can be applied to another dataset.

In this example, an **Impute** block is used to fill in missing values in a `lib_books.csv` dataset. The **Impute** block is then configured to output its model, which is then applied to another dataset, `lib_books2.csv`, using a **Score** block. The **Score** block generates an output dataset `BOOKS 2 COMPLETE`.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

lib_books.csv dataset

This example uses a dataset of books, named `lib_books.csv`. Each observation in the dataset describes a book, with variables such as *Title*, *ISBN*, *Author* etc.

lib_books2.csv dataset

A second dataset of books is used in this example, with the same variables as the `lib_books.csv` dataset.

Using the Impute block's model output

Imputing missing values in a dataset using the **Impute** block and then using the **Impute** block's model output to impute missing values in another dataset.

1. Import the `lib_books.csv` dataset onto a blank Workflow canvas. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Drag an **Impute** block onto the Workflow canvas, beneath the `LIB_BOOKS` dataset.
3. Draw a connection between the `lib_books.csv` dataset block output port and the **Impute** block input port.
4. Double-click on the **Impute** block to open the **Configure Impute** dialog box, and then configure as follows:
 - a. Set **Variable** to **Price**.
 - b. Set **Method** to **Distribution**.
 - c. Click **OK**.

A green execution status is displayed in the **Output** ports of the **Impute** block and of the new dataset **Working Dataset**.

5. Import the `lib_books2.csv` dataset into a blank Workflow canvas.
6. Right-click on the **Impute** block and select **Configure Outputs**.

An **Outputs** window is displayed. By default, the **Working Dataset** is in **Selected Output** and the **Impute Model** is in **Unselected Output**.

7. Double-click on **Impute Model** to move it into the **Selected Output** list.

Selected Output lists both **Working Dataset** and **Impute Model**.

8. Click **OK**.

The **Outputs** window is closed, and on the Workflow canvas the **Impute** block can be seen to have both a dataset and model output.

9. From the **Scoring** grouping in the Workflow palette, click and drag a **Score** block onto the Workflow canvas, below the new **Impute Model** block and the `LIB_BOOKS 2` dataset block.

A **Score** block and empty dataset block are created on the Workflow canvas. Note that the **Score** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires a model and a dataset input.

10. Click on the `LIB_BOOKS_2` dataset block's **Output** port and drag a connection towards the **Input** port of the **Score** block.

11. Click on the **Impute Model Output** port and drag a connection towards the **Input** port of the **Score** block.

The execution status of the **Score** block turns green and its dataset output is populated.

12. Right-click on the **Score** block's output dataset block and click **Rename**. Type `BOOKS_2_COMPLETE` and click **OK**.

The imputation model output from the **Impute** block has imputed the missing values in the `LIB_BOOKS_2` dataset and generated a complete dataset, now named `BOOKS_2_COMPLETE`.

Join block

Enables you to combine observations from two datasets into a single working dataset.

Rows are joined using a common column. For example, if the two tables have observations that specify an identifier (perhaps using the variable `ID` in one, and `identifier` in another), the observations in which the identifiers have the same value are joined together in the same observation.

Datasets can be joined in various ways, and the condition used to join records does not have to be one of equality.

Joining datasets

To join datasets:

1. Drag the **Join** block onto the Workflow canvas, and connect it to the datasets you want to join.

2. Double-click the **Join** block.

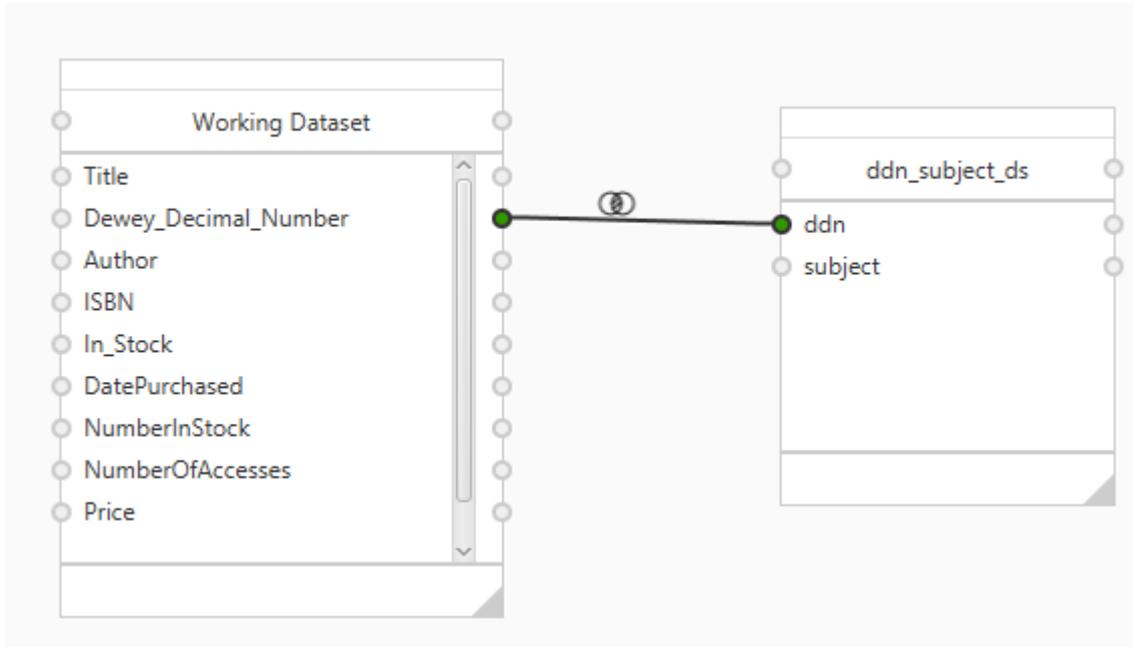
The **Join Editor** is displayed. This contains a representation of the datasets and variables to be joined. For example:



Here, the datasets to be joined are **Working Dataset** and `ddn_subject_ds`.

- Click on the connector  that corresponds to the variable you want to join in one dataset, and drag the cursor to the corresponding connector in the other dataset.

For example, if you want to join the datasets above on the `Dewey_Decimal_Number` variable, click the connector corresponding to that in the left-hand dataset, and then drag the cursor to the corresponding variable in the other dataset. For example:



This creates an inner join between the datasets. To create a different kind of join, you need to configure the join.

The dataset on which you click first is the left-hand table in the join. For example, with the two datasets above, clicking `ddn` in the right-hand dataset and dragging a connector to `Dewey_Decimal_Number` in the left-hand dataset makes the right-hand dataset the left dataset in the join.

You can also specify the variables to connect, and the left dataset, using the **Join Properties** dialog.

Configuring the join properties

To configure the join properties, click the symbol above the connector between variables. This displays the **Join Properties** dialog. The symbol represents the type of join, and is by default .

The following properties can be configured:

Join type

Specifies the type of join used to combine the values in the left and right datasets. Inner and outer joins use a condition to select observations, enabling you to select the variables you want to compare. The variable condition is defined using **Join operator**.

Inner

Creates a working dataset that includes observations from both datasets where the specified condition matches for the specified variable.

Left Outer

Creates a working dataset that contains all observations in the left dataset and the values of those variables from observations in the right dataset where the specified condition matches for the specified variable.

Right Outer

Creates a working dataset that contains all observations in the right dataset and the values of those variables from observations in the left dataset where the specified condition matches for the specified variable.

Full Outer

Creates a working dataset that contains all observations from either or both of the right dataset and the left dataset where the specified variable matches the specified condition.

Cross

Creates a working dataset that consists of all observations in the left dataset joined with every observation in the right dataset.

Natural

Creates a working dataset using commonly-named variables in the left and right datasets. The dataset only includes observations where the variable in the left dataset has a value that matches the variable in the right dataset. Any other observations are ignored.

Join operator

Specifies the type of operator for the join. The condition uses the operator and the specified columns to define which observations are included or excluded by a join type. The operators available are:

=	Equal to
!=	Not equal to
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Left dataset

Identifies the left dataset in the join, and the columns on which to join. Although the left dataset is initially defined when you connect the dataset in the **Join Editor**, you can change the left dataset here:

Dataset

Select the dataset to be used as the left dataset. This is set by default if you connect datasets using the dataset representation in the **Join Editor**.

Column

Specify the variable in the left dataset to be used in the join. This is set by default if you connect datasets using the dataset representation in the **Join Editor**.

Right dataset

Identifies the right dataset in the join and the columns. Although the left dataset is initially defined when you connect the dataset in the **Join Editor**, you can change the left dataset here:

Dataset

Select the dataset to be used as the right dataset. This is set by default if you connect datasets using the dataset representation in the **Join Editor**.

Column

Specify the variable in the right dataset to be used in the join. This is set by default if you connect datasets using the dataset representation in the **Join Editor**.

Merge block

Enables you to combine two datasets into a single working dataset.

Block Overview

A **Merge** block takes two or more connections from datasets and joins those datasets together. To configure a merge block, double-click to open the **Configure Merge** dialog box.

Options tab

Dataset order

Lists the order that the datasets will be combined in. To move a dataset in the order, click on that dataset and use the arrow keys to move it.

Merge operation

Concatenate

Joins one dataset onto another. Variables from each dataset will remain distinct with no merging between observations. In both cases, variables and observations in datasets will be concatenated in the order specified in **Dataset order**.

Interleave

Available for selection if source datasets contain common variables (dictated by variable name and type). Observations from common variables will be joined so that they belong to the same variable. Variables and observations in datasets will be interleaved in the order specified in **Dataset order**.

One-to-one

Joins datasets together so that observations are merged. Variables and observations in datasets will be joined in the order specified in **Dataset order**.

One-to-one including excess rows

Joins datasets together so that observations are merged. Variables and observations in datasets will be joined in the order specified in **Dataset order**.

Excess observations are included, such that the smaller dataset contains blank observations for rows beyond its original size.

By Variables

Allows you to specify variables to sort by. Two lists are presented: **Unselected By Variables** showing variables available, and **Selected By Variables** showing variables selected. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Mutate block

Enables you to modify or add new variables to the working dataset. These variables can be independent of, or derived from variables in the existing dataset.

The new variable name, type and format are set and the content based on an SQL expression. You can, for example, use the expression to duplicate the content of a variable into the new variable, or apply a function to create a new variable based on the value of an existing variable in the dataset.

The input dataset remains unchanged, and any variables created using the **Mutate** block are appended to the output working dataset.

New variables are defined using the **Configure Mutate** dialog box. To open the **Configure Mutate** dialog box, double-click the **Mutate** block.

Mutated Variables

The main panel in this section gives a list of new mutated variables created by this block. Each variable displays its **Variable Name**, **Type** and **Format**. The list can be sorted by clicking on any of the header fields.

Plus icon

Add a new mutated variable. This will enable the fields in the **Variable Definition** section and add a new entry to the Mutated Variables list.

Delete icon

Deletes a selected variable from the **Mutated Variables** list.

Variable Definition

Describes the new mutated variable to be created.

Name

Specifies the variable name displayed when the working dataset is viewed in the **Data Profiler** view or the Dataset File Viewer.

Type

Select the SAS language type for the new variable:

- **Character**, for all character- and string-based variables.
- **Numeric**, for all numeral-based variables including any date, time or datetime variables.

Label

Specifies the display name that can be used when outputting graphical interpretation of the variable.

Length

Specifies the maximum length of a variable.

Format

The display format for the new variable. The format applied does not affect the underlying data stored in the `DATA` step. For more information see the section *Formats* in the *WPS Reference for Language Elements*.

Informat

The format applied to data as it is read. For more information see the section *Informats* in the *WPS Reference for Language Elements*.

SQL Expression

Defines the data content of the new variable. The SQL expression can either be written into this area manually, or created by double-clicking on or dragging the input variables, functions and operators given below the **SQL Expression** entry area, which will populate the SQL Expression area for you. When typing manually, press **Ctrl-Spacebar** to show a list of suggested terms.

The SQL expression can be used to duplicate an existing variable by entering that variable's name; conditionally create new values using the SQL procedure `CASE` expression, or create values that are independent from existing variables.

Most SAS language `DATA` step functions can also be entered.

For example, you can create a variable containing the current date and time by manually entering the SAS language function `DATETIME`. Specifying a format such as `DATETIME.` for the variable enables the content to be displayed in a readable date and time format.

Input Variable

Lists the existing variables in the dataset being mutated. Double-clicking an **Input Variable** will add it to the **SQL Expression**. Alternatively, input variables can be dragged into the **SQL Expression** area. Variable names with spaces are automatically enclosed in quotes and suffixed with an 'n' to define them as name literals.

Function

Lists supported SQL functions. Double-clicking an SQL function will add it to the **SQL Expression** area. Alternatively, functions can be dragged into the **SQL Expression** area. Supported functions are:

- **AVG** and **MEAN**: Both return the arithmetic mean of a list of numeric values in a specified variable. **AVG** is an alias of **MEAN**.
- **COUNT**: Can be used in three forms:
 - `Count (*)`: returns the total number of observations.
 - `Count(variable)`: returns the number of non-missing values in the specified variable.
 - `Count(variable, "string")`: returns true (1) for each observation where the specified string occurs in the specified variable, otherwise returns false (0).
- **FREQ** and **N**: Both return a count of the number of observations for a specified variable, with behaviour dependent on the variable type:
 - If a numeric variable is specified, missing values are included.
 - If a string variable is specified, missing values are excluded.

FREQ is an alias of **N**.

- **CSS**: Returns the corrected sum of squares of a list of numeric values.
- **CV**: Returns the coefficient of variation (standard deviation as a percentage of the mean) for a list of numeric values.
- **MAX**:
 - If a numeric variable is specified, returns the maximum value.
 - If a string variable is specified, returns the longest string.
- **MIN**:
 - If a numeric variable is specified, returns the minimum value.
 - If a string variable is specified, returns the shortest string.
- **NMISS**: Returns the number of missing values in a list of string or numeric values.
- **PRT**: Returns the probability that a randomly drawn value from the Student T distribution is greater than the T statistic for a numeric variable.
- **RANGE**: Returns the difference between the maximum and minimum value in a list of numeric values.
- **STD**: Returns the standard deviation of a list of numeric values.
- **STDERR**: Returns the standard error of the mean of a list of numeric values.
- **SUM**: Returns the sum of values from a list of numeric values.
- **SUMWGT**: Returns the sum of weights.
- **T**: Returns the t statistic for a variable.
- **USS**: Returns the uncorrected sum of squares of a list of numeric values.
- **VAR**: Returns the variance of a list of numeric values.

- **COALESCE**: Returns the first non-missing value in a list of numeric values.
- **COALESCEC**: Returns the first string that contains characters other than all spaces or null.
- **MONOTONIC**: Can only be used without arguments. Generates a new variable populated with observation numbers, starting at 1 and finishing at the final observation.

Operators

Lists SQL operators that can be used with Input Variables and Functions. Clicking an operator will add it to the **SQL Expression** area above.

Partition block

Enables you to divide the observations in a dataset into two or more working datasets, where each working dataset contains a random selection of observations from the input dataset with a specified weighting for the split.

Block Overview

A **Partition** block is typically used to create *training* and *validation* working datasets for model training.

By default, the weighting is set to 50:50, so two working datasets are created that each contain roughly 50% of the input dataset observations. You can add or remove output datasets in the **Configure** dialog box, alter the weighting to reflect the relative importance of the output datasets created, and introduce a seed to replicate the split of datasets.

Data partitioning is configured using the **Configure Partition** dialog box. To open the **Configure Partition** dialog box, double-click the **Partition** block.

Defining partitions

The **Configure Partition** dialog box contains a list of all partitions that will be created.

- To create a new partition, click **Add Partition**. The new partition has a weight of 1.00.
- To delete a partition, select the partition in the **Partition** column and click **Remove Partition**.
- To rename a partition, select the partition in the **Partition** column and enter a new name.
- To change the weighting applied to a partition, select the value in **Weight** column and enter a new weighting value.

Controlling observations in a partition

Every time the **Partition** block is executed, the partition weighting is persisted, but the sequence of observations in the working datasets may change. The **Seed** field enables you to recreate the same sequence of observations each time the **Partition** block is executed.

- To generate the same sequence of observations, set **seed** to a positive value greater than or equal to 1.

- To generate a different sequence of observations, set **seed** to 0 (zero).

Query block

Enables you to create SQL code to join and interrogate database tables or datasets.

Block Overview

A **Query** block enables you to create SQL code that you can use to join database tables or datasets, to get specific data from one or more database tables or datasets. The terms dataset and tables, and variables and columns, are equivalent for this block. To create the SQL code, double-click the block to open the **Query** full screen editor. The editor consists of three tabs:

- **Query** – Select this tab to create your query. The tab is described in more detail below.
- **SQL** – Select this tab to view the SQL code that results from the query you create.
- **Preview** – Select this tab to preview the results of the query you have created. This displays the data that is written to the working dataset by the block when you save it.

Query tab

The **Query** tab contains the following panels:

- **Tables** – This shows the database tables that have been connected to the block, enables you to select columns for the query and to join the tables if you have connected more than one table to the block.
- **Columns** – Lists the columns from the tables that the query will use.
- **Where** – Enables you to specify `WHERE` clauses for the columns in the query.
- **Having** – Enables you to specify a `HAVING` clause for grouped columns. This panel is only displayed if you select **Group** for a column in the **Columns** panel.

The working dataset created by this block contains the selected columns from an input table or from joined tables. You can create expressions based on a selected column and create new columns based on the results, or write the result of the expression into the same column.

Saving the query

To save, either click the Save button () or press **Ctrl+S**.

Tables

Enables you to select columns (variables) to be used in the query, and to join tables if more than one has been connected to the block.

This panel displays the tables connected to the block. You can:

- Join the tables, if there is more than one.
- Select columns from the tables for the query.

To join tables, select a column in one table and drag it on to a column in another table. The tables are then joined on these two columns. This creates an inner join between the tables. The table on which you click first is the left-hand table in the join. To create a different kind of join, you need to configure the join. You can configure the join using the **Join Properties** dialog. You can also specify the columns to connect, and the left table, using the **Join Properties** dialog.

If you want to create a natural or cross join for the tables, select the heading of one table, and drag it onto the heading of the other table. This created a natural join by default. If you want to change it to a cross join, use the **Join Properties** dialog.

Configuring the join properties

To configure the join properties, double-click the line that connects the tables. This displays the **Join Properties** dialog.

The following properties can be configured:

Join type

Specifies the type of join used to combine the values in the left and right tables. Inner and outer joins use a condition to select rows, enabling you to select the columns you want to compare. The column condition is defined using **Join operator**.

Inner

Creates a working dataset that includes rows from both tables where the specified condition matches for the specified column.

Left Outer

Creates a working dataset that contains all rows in the left table and the values of those columns from rows in the right table where the specified condition matches for the specified column.

Right Outer

Creates a working dataset that contains all rows in the right table and the values of those columns from rows in the left table where the specified condition matches for the specified column.

Full Outer

Creates a working dataset that contains all rows from either or both of the right table and the left table where the specified column matches the specified condition.

Cross

Creates a working dataset that consists of all rows in the left table joined with every row in the right table.

Natural

Creates a working dataset using commonly-named columns in the left and right tables. The table only includes rows where the column in the left table has a value that matches the column in the right dataset. Any other rows are ignored.

You can display the type of join that currently applies between tables can be displayed if you hover the mouse pointer over the line connecting the tables.

Join operator

Specifies the type of operator for the join. The condition uses the operator and the specified columns to define which rows are included or excluded by a join type. The operators available are:

=	Equal to
!=	Not equal to
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Left dataset

Identifies the left dataset (table) in the join, and the columns on which to join.

Dataset

Select the table to be used as the left table. This is set by default if you have already connected the tables by dragging.

Column

Specify the column in the left table to be used in the join. This is set by default if you have already connected tables using the table representation in the **Join Editor**.

Right dataset

Identifies the right dataset (table) in the join and the columns on which to join.

Dataset

Select the table to be used as the right table. This is set by default if you connect tables using the table representation in the **Join Editor**.

Column

Specify the column in the right table to be used in the join. This is set by default if you have already connected tables using the table representation in the **Join Editor**.

Columns

Lists the columns that will be included in the SQL query and enables you to create expressions.

The **Columns** pane specifies the columns used in the query. You can also define expressions to be applied to the columns.

Columns panel

The **Columns** panel lists the columns that are used to populate the working dataset that results from this block.

You can add a column to this panel by:

Dragging

Click a column name in a table in the **Tables** panel and then drag it to the **Columns** panel. A dragged column is added to the end of those already listed. If you need more control over the order of columns in the output, use the **Add Columns** dialog box instead.

You can add all columns from a table by dragging the table header into the **Columns** panel. This adds an entry to **Column** of the form *table-name.**, where *table-name* is the name of the table you dragged, and *** is the wildcard that specifies all columns in that table.

Double clicking

You can enter a column name in the **Columns** panel by double clicking a column in a table in the **Tables** panel. The column is added to the end of those already listed. If you need more control over the order of columns in the output, use the **Add Columns** dialog box instead.

You can add all columns from a table by double clicking the table header. This adds an entry to **Column** of the form *table-name.**, where *table-name* is the name of the table you dragged, and *** is the wildcard that specifies all columns in that table.

Using the Add Columns dialog box

You can enter a column name in the **Columns** panel by clicking on the **Add Columns**  icon in the panel. This opens the **Add Columns** dialog box.

The **Add Columns** dialog box enables you to specify which columns are to be included. Two lists are presented:

- **Available columns**, which shows the columns available for selection.
- **Columns to add**, which shows the columns that have been selected.

To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the **Select all** button () or **Deselect all** button () to move all the items from one list to the other. Use

the filter box (🔍) on either list to limit the variables displayed, either by a simple text filter, or by a regular expression; as you enter text or an expression in the box, only those variables containing the text or expression are displayed in the list. The filter for **Available columns** provides a dropdown list of predefined filters.

The list of variables contains `*`, which can be used to specify that all variables from all tables are used in the query, and `table-name.*`, which can be used to specify that all variables from the table `table-name` are included in the query.

Editing a row in Columns

You can double click in **Column** in an empty row to enter a column name directly. When you have entered the column name, press the Return key. The **Select**, **Sort** and **Group** controls then become available, and you can edit the corresponding **Alias**.

The working dataset consists of the columns listed (if they remain selected), and any columns created as a result of SQL expressions you create using the Expression Builder.

The **Columns** panel contains the following:

Column

Specifies the name of a column that is used in the SQL query to create the working dataset.

Alias

Specifies an alternative name for a column. If you do not specify an alias using this column, then the name specified in `Column` is the name used in the working dataset.

If you create an SQL expression based on the column name, a new column is created with a system-generated name to hold the result; **Alias** remains blank by default. However, if you enter a name in this cell, the result is held in a column with that name. Alternatively, if the column name specified here is the same as the name in **Column**, then the values created using an SQL expression are written to that column rather than to a column with a system-generated name; that is, the result of the expression replaces the existing value of the column.

Select

The column is used in the SQL `SELECT` statement in the SQL query that is created to generate the working dataset. By default, when you add a column to the **Columns** pane, it is selected. However, you can deselect it if required.

Sort

Specifies the sort order for the column. Click on the box to cycle through the choices: ascending (↑), descending (↓), or none (–); none is the default.

Group

Specifies that the column is an SQL `GROUP BY` column, enabling you to group results.

You can rearrange the order of the columns using the up (▲) or down (▼) controls. The order of the columns affects the order of the columns in the `SELECT` clause of the SQL query, and might therefore affect the result of the query.

Expression Builder

The Expression Builder enables you to specify an SQL expression that creates a new value for the table. This value is written to a column with a system-generated name, written to a column with a user-defined name, or replaces the value in the column for which the expression is created. The SQL expression can be created either entirely manually, or by using the controls in the Expression Builder.

To open the Expression Builder you can do one of the following:

- Double click a column name in the **Columns** panel.
- Right click a column in the **Columns** panel. A pop-up is displayed. Click **Edit with Expression Builder**. The Expression Builder dialog box opens.

At the top of the dialog box is the **SQL Expression** area, into which you can manually enter an SQL expression. You can also enter most SAS language DATA step functions in this box. For example, you can create a column that contains the current date and time by manually entering the SAS language function `DATETIME`. Specifying a format such as `DATETIME.` for the column enables the content to be displayed in a readable date and time format.

If your SQL expression creates another value, this value is stored in a temporary column with a system-generated name, unless you specify a column name in **Alias**.

The other controls in this dialog box enable you to create the elements of an expression in the SQL Expression box. To enter an element into the area, you can double click an element, or drag it. The elements available are provided in two lists:

Operators

Lists SQL operators that can be used with columns and functions. Click an operator to add it to the **SQL Expression** area.

Input variables

The columns (variables) that can be used in the SQL expression are listed in **Input variables**. If the query consists of more than one table, the column name is prefixed with the table name, so the column name has the form *table-name.column-name*.

Functions

Lists supported SQL functions. Double-clicking an SQL function will add it to the **SQL Expression** area. Alternatively, functions can be dragged into the **SQL Expression** area. Supported functions are:

AVG

This is an alias of `MEAN`.

MEAN

Returns the arithmetic mean of a list of numeric values in a specified column.

COUNT

Can be used in three forms:

- `Count (*)`: returns the total number of rows.
- `Count (column)`: returns the number of non-missing values in the specified column.

- `Count(column, "string")`: returns true (1) for each observation where the specified string occurs in the specified column, otherwise returns false (0).

FREQ

This is an alias of **N**.

N

Returns a count of the number of rows for a specified column, with behaviour dependent on the column type:

- If the column contains numeric data, missing values are included.
- If the column contains character data, missing values are excluded.

CSS

Returns the corrected sum of squares of a list of numeric values.

CV

Returns the coefficient of variation (standard deviation as a percentage of the mean) for a list of numeric values.

MAX

- If a numeric column is specified, returns the maximum value.
- If a string column is specified, returns the longest string.

MIN

- If a numeric column is specified, returns the minimum value.
- If a string column is specified, returns the shortest string.

NMISS

Returns the number of missing values in a list of string or numeric values.

PRT

Returns the probability that a randomly drawn value from the Student T distribution is greater than the T statistic for values in the column.

RANGE

Returns the difference between the maximum and minimum value in a list of numeric values.

STD

Returns the standard deviation of a list of numeric values.

STDERR

Returns the standard error of the mean of a list of numeric values.

SUM

Returns the sum of values from a list of numeric values.

SUMWGT

Returns the sum of weights.

T

Returns the t statistic for a column.

USS

Returns the uncorrected sum of squares of a list of numeric values.

VAR

Returns the variance of a list of numeric values.

COALESCE

Returns the first non-missing value in a list of numeric values.

COALESCEC

Returns the first string that contains characters other than all spaces or null.

MONOTONIC

Can only be used without arguments. Generates a new column populated with observation numbers, starting at 1 and finishing at the final observation.

WHERE and HAVING

Enables you to specify a `WHERE` or `HAVING` clause for the SQL query.

The SQL query you create acts on every row of the table unless you specify a `WHERE` or `HAVING` clause to limit the rows selected. Both clauses specify that only those rows where a condition is met are used for further processing.

You specify a `WHERE` clause using the **WHERE** edit box. You directly enter the required clause in this box. The **HAVING** edit box appears if you select the **GROUP** option for a column. You use `HAVING` with grouped rows, and `WHERE` with individual rows. You can use both `WHERE` and `HAVING` in the same query.

Query block example: selecting specified rows

This example demonstrates how the **Query** block can be used to create a query that results in only the selected rows being written to an output dataset.

In this example, a comma separated variable file of books named `LIB_BOOKS.CSV` is imported into a blank Workflow. The output required is a working dataset that contains the titles of books by a particular author that are no longer in the library collection.

Workflow Layout



Dataset used

The files used in this example can be found in the samples distributed with WPS Analytics.

LIB_BOOKS.CSV

Each record in the file describes a book, with fields such as `Title`, `ISBN`, `Author`, and so on.

DDN_SUBJECTS.CSV

Each record in the file links a number in the Dewey Decimal Number system to a subject.

Using the Query block to create a basic query

Creating a query that writes only the specified rows and variables to the working dataset.

1. Import the file `LIB_BOOKS.CSV` into a blank Workflow canvas. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Drag a **Query** block onto the Workflow canvas, beneath the `LIB_BOOKS.csv` block.
3. Draw a connection between the **Working Dataset** output port of the `LIB_BOOKS.csv` block and the **Query** block input port.
4. Double-click the **Query** block.

The **Query** editor is displayed.

5. In the **Tables** panel, in the Working Dataset, double click **Title** and then **In_stock**.

The `Title` and `In_stock` rows are added to the **Columns** panel.

6. Ensure the columns are included in the query by selecting **Select**, if necessary. This control is selected by default when you enter a column in the **Columns** table.

7. Enter the following in the **Where** box:

```
author EQ "Rendell, Ruth" AND In_Stock EQ "n"
```

This specifies that only those rows where the `Author` column contains the text `Rendell, Ruth` and the `In_Stock` column contains `n`, are output to the working dataset.

8. To save, either click the Save button () or press **Ctrl+S**.

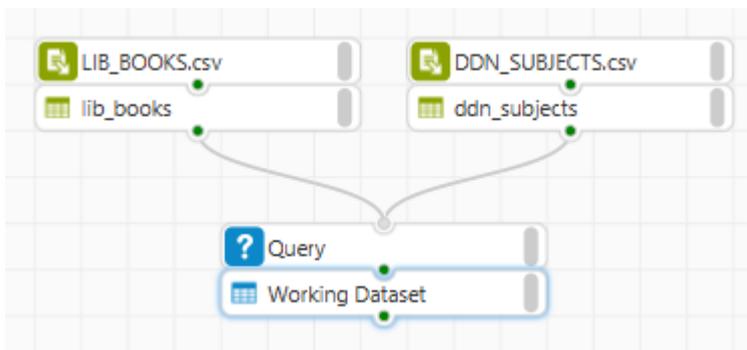
The working dataset contains the selected values.

Query block example: joining tables and selecting rows

This example demonstrates how the **Query** block can be used to create a query that joins two tables and selects specified rows from the joined table.

In this example, there are two comma separated variable files; one contains a list of books and information about those books (`LIB_BOOKS.CSV`), the other contains information about book subjects (`DDN_SUBJECTS.CSV`). These are imported into a blank Workflow. The working dataset contains the output required from the input file, which is a list non-fiction book subjects and the number of books with that subject, for a specified author.

Workflow Layout



Datasets used

The files used in this example can be found in the samples distributed with WPS Analytics.

LIB_BOOKS.CSV

Each record in the file describes a book, with fields such as `Title`, `ISBN`, `Author`, and so on.

DDN_SUBJECTS.CSV

Each record in the file links a number in the Dewey Decimal Number system to a subject.

Using the Query block to join tables and to group output

Creating a query that joins two tables, acts only on the specified rows, and outputs grouped data.

1. Import the `LIB_BOOKS.CSV` and the `DDN_SUBJECTS.CSV` files into a blank Workflow canvas. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).

The working datasets of both datasets created by the blocks is called **Working Dataset**. Because this might be confusing, you can rename the datasets. To do this for each working dataset, right click-on it, click **Rename** and enter a suitable name. In this example, `lib_books` and `ddn_subjects` have been used for the corresponding working datasets.

2. Drag a **Query** block onto the Workflow canvas, beneath the Text File Import blocks.
3. Draw a connection between the output ports of the working datasets and the **Query** block input port.
4. Double-click the **Query** block.

The **Query** editor is displayed.

5. In the **Tables** panel, in the **Subjects** table, drag the column **DDN** over the variable **Dewey_Decimal_Number** in the **Lib_books** table.

An inner join is created between the two tables. This is the default join. The `Subjects` table is the left dataset in the join. A line is drawn between the variables in the two tables.

6. Double click **Subject** in the **Subjects** table.

The **Subject** row is added to the **Columns** panel.

7. Right click on **Subject**.

The **Edit with Expression Builder** context menu is displayed.

8. Click **Edit with Expression Builder**.

The **Expression Builder** is displayed. The **SQL Expression** editor box contains the column name `Select`.

9. Click **Count** in the **Function** list.

The function `COUNT()` is added to the **SQL Expression** editor.

10. Edit the contents of **SQL Expression** so that the expression is `COUNT(Subject)`.

This expression provides a total count of subject names. The expression replaces the column name in the **Columns** pane. The value returned by `COUNT` is written into a column with a system-generated name, unless you specify an alias. The next steps shows how to do this.

11. Click the **Alias** box in the row that now contains `COUNT(Subject)`. The box becomes editable. Type a name for the column: `Number in category`. This replaces the system-generated name.

12. Enter the following in the **Where** box:

```
author EQ "Wilson, Colin"
```

This specifies that the only rows used in the query are those where the `Author` column contains the text `Wilson, Colin`.

13. In the first row of **Column**, click **Group**.

This ensures that the rows are grouped by the values of `Subject`; the total number of each group of subjects is then returned to the column `Number in category`.

The **Having** edit box is also displayed. This enable you to enter an SQL `HAVING` clause for the group if required.

14. Enter the following in the **Having** box:

```
subject NE "English fiction"
```

This ensures that only non-fiction subjects for the specified author are counted.

15. Double click **Subject** in the **Subjects** table again.

The **Subject** row is added as another row in the **Columns** panel, beneath the first row.

16. To save, either click the Save button () or press **Ctrl+S**.

The following SQL query is created:

```
select COUNT(Subject) as 'Number in category'n,  
Subjects.Subject  
from Subjects as Subjects inner join lib_books as lib_books on  
lib_books.Dewey_Decimal_Number = Subjects.DDN  
where author EQ "Wilson, Colin"  
group by Subject  
having subject NE "English fiction"
```

You can see this by clicking the **SQL** tab in the Query editor. The working dataset contains the selected values:

Subject	Number in category
Communities	1
Conscious mental processes & intelligence	1
English essays	1
Social interaction	2
Specific topics in parapsychology & occultism	4

Rank block

Enables you to rank observations in an input dataset using one or more numeric variables.

The input dataset remains unchanged, and any rank variables created using the **Rank** block are appended to the output working dataset.

Rank scores are defined using the **Configure Rank** dialog box. To open the **Configure Rank** dialog box, double-click the **Rank** block.

Variable Selection panel

Specifies which variables are to be ranked. The upper list of **Unselected Variables** shows variables that can be specified for ranking and the lower list of **Selected Variables** shows variables that have been specified for ranking. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Parameters panel

Enables you to specify how the rank scores for observations in the dataset are calculated.

General

Specifies ranking order and how identical results are ranked.

Rank order

Specifies the order in which ranking scores are applied to observations ordered by the variables specified in the **Variable Selection** panel.

Ascending

Rank scores are specified in ascending order, with the lowest rank applied to the smallest observation.

Descending

Rank scores are specified in descending order, with the highest rank applied to the smallest observation.

Use default treatment for ties

When selected, **Tie Treatment** is chosen automatically based on the **Ranking Method** selected:

- If **Ranking Method** is specified as **Fractional**, **Fractional (N + 1)**, or **Percent**, **Tie Treatment** is automatically set to **High**.
- If **Ranking Method** is specified as **Ordinal**, **Savage**, **Groups**, or **Normal**, then **Tie Treatment** is automatically set to **Mean**.

Tie Treatment

Specifies the ranking of identical results.

The following input dataset contains the result of the *IAAF 2015 World Championships 100 Meters Men – Final*, and is used in the examples below.

Name	Time
Bolt, Usain	9.79
Bromell, Trayvon	9.92
De Grasse, Andre	9.92

Name	Time
Gatlin, Justin	9.80
Gay, Tyson	10.00
Powell, Asafa	10.00
Vicaut, Jimmy	10.00
Rodgers, Mike	9.94
Su, Bingtian	10.06

Dense

Ranks the variable using the dense tie resolution. The method assigns the same rank value to all tied variables and the next variable is assigned the immediately-following rank.

Applying `Dense` to the input dataset above, creates the following working dataset:

Name	Time	Time_RANK
Bolt, Usain	9.79	1
Bromell, Trayvon	9.92	3
De Grasse, Andre	9.92	3
Gatlin, Justin	9.80	2
Gay, Tyson	10.00	5
Powell, Asafa	10.00	5
Vicaut, Jimmy	10.00	5
Rodgers, Mike	9.94	4
Su, Bingtian	10.06	6

High

Ranks the variable using the high tie resolution. The method assigns the highest rank value to all tied variables.

Applying `High` to the input dataset described above creates the following working dataset:

Name	Time	Time_RANK
Bolt, Usain	9.79	1
Bromell, Trayvon	9.92	4
De Grasse, Andre	9.92	4
Gatlin, Justin	9.80	2
...

Bromell, Trayvon and De Grasse, Andre have the same time, and ranks 3 and 4. Specifying **High** gives both a rank of 4.

Low

Ranks the variable using low tie resolution. The method assigns the lowest rank value to all tied variables.

Applying `LOW` to the input dataset described above creates the following dataset:

Name	Time	Time_RANK
Bolt, Usain	9.79	1
Bromell, Trayvon	9.92	3
De Grasse, Andre	9.92	3
Gatlin, Justin	9.80	2
...

Bromell, Trayvon and De Grasse, Andre have the same time, and ranks 3 and 4. Specifying **Low** gives both a rank of 3.

Mean

Ranks the variable using mean tie resolution. The method assigns the mean rank value to all tied variables.

Applying `Mean` to the input dataset described above creates the following working dataset:

Name	Time	Time_RANK
Bolt, Usain	9.79	1
Bromell, Trayvon	9.92	3.5
De Grasse, Andre	9.92	3.5
Gatlin, Justin	9.80	2
...

Bromell, Trayvon and De Grasse, Andre have the same time, and ranks 3 and 4. Specifying **Mean** gives both a rank of 3.5.

Ranking Method

Specifies how the rank value for each observation is calculated.

Ordinal

Assigns an incrementing rank number to each observation.

Fractional

Each rank value is calculated as the ordinal ranking method divided by the number of observations in the dataset.

Fractional (N+1)

Each rank value is calculated as the ordinal ranking method divided by the one plus the number of observations in the dataset.

Percent

Each rank value is calculated as the ordinal ranking method divided by the number of observations in the dataset, expressed as a percentage.

Savage

The rank values for the observations are transformed to an exponential distribution using the Savage method. Subtracting one from the transformation result centers the rank score around zero (0):

$$t_{RN} = \left[\sum_{r=1}^R \frac{1}{(N-r+1)} \right] - 1 \text{ for } (R = 1, \dots, N)$$

Where:

- R is the rank score.
- N is the number of observations.

Groups

Divides the observations into the specified number of groups based on the ranking score. Observations with tied ranking scores are allocated to the same group.

Normal

The rank values for the observations are transformed to a standard normal distribution using the selected method:

Blom

Creates the standard normal distribution of rank scores using a Blom transformation:

$$F_i = \frac{\left(R_i - \frac{3}{8}\right)}{\left(N + \frac{1}{4}\right)}$$

Where:

- R is the rank score.
- N is the number of observations.

Tukey

Creates the standard normal distribution of rank scores using a Tukey transformation:

$$F_i = \frac{\left(R_i - \frac{1}{3}\right)}{\left(N + \frac{1}{3}\right)}$$

Where:

- R is the rank score.
- N is the number of observations.

VW

Creates the standard normal distribution of rank scores using a Van der Waerden transformation:

$$F_i = \frac{R_i}{(N+1)}$$

Where:

- R is the rank score.
- N is the number of observations.

Sampling block

Enables you to create a working dataset that contains a small representative sample of observations in the input dataset.

A sample is configured using the **Configure Sampling** dialog box. To open the **Configure Sampling** dialog box, double-click the **Sampling** block.

Sample Type

Specifies the type of samples, either *Random* or *Stratified*.

Random Sampling

A working dataset is created by selecting a random set of observations to match the required sample size.

Sample Size

Specifies how the sample is constructed:

- **Percentage of obs** specifies the percentage of observations in the input dataset to include in the output working dataset.
- **Fixed number of obs** specifies the absolute number of observations in the input dataset to include in the output working dataset.

Stratified Sampling

The dataset is divided in groups where observations contain similar characteristics. Sampling is applied to each group, ensuring that each group is represented in the output working dataset.

Strata Variable

Specifies the variable in the dataset to use when dividing the whole dataset into groups.

Sample Size

Specifies how the sample is constructed:

- **Percentage of stratum obs** specifies the percentage of each group to include in the output working dataset.

- **Custom** enables you to determine the number of observations from each group, allowing you to increase or decrease the weight of each group in the output working dataset.

Use Probability Proportional to Size Sampling

Specifies that samples are taken using the same probability used to divide the input dataset into groups.

Size variable

Specifies a variable that indicates sample sizes.

Seed

Specifies whether the same set of observations is included in the output working dataset each time the **Sampling** block is executed. A **Seed** can be applied to either a random sample or a stratified sample.

- To generate the same sequence of observations, set **Seed** to a positive value greater than or equal to 1.
- To generate a different sequence of observations, set **Seed** to 0 (zero).

Select block

Enables you to create a new dataset containing specific variables from an input dataset.

Variables included in the working dataset are specified using the **Configure Select** dialog box. To open the **Configure Select** dialog box, double-click the **Select** block.

The **Configure Select** dialog box enables you to specify which variables are to be included. Two lists are presented: **Unselected Variables** showing variables available for selection, and **Selected Variables** showing variables selected. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Sort block

Enables you to sort a dataset.

You can sort a dataset using one or more variables. For example, if you were sorting a dataset containing book titles and author names, you could first sort by author name, to get all authors in alphabetical order, and then title, so that the titles for each author are also sorted alphabetically. You can sort in ascending or descending order, and you can specify what happens with duplicate records and keys.

The sorting of values is configured using the **Configure Sort** dialog box. To open the **Configure Sort** dialog box, double-click the **Sort** block.

The dataset is sorted according to the variables and sort order you specify, and a working dataset containing the sorted dataset is created.

Variable list

Two lists are presented: **Unselected Variables** showing variables available for sorting, and **Selected Variables** showing variables selected for sorting. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Remove Duplicate Keys

Specifies that, for observations that have duplicate keys, only the first of the observations with a duplicated key is included in the sort and written to the working dataset.

Remove Duplicate Records

Specifies that, for observations that are exactly the same, only the first of the duplicated observations is included in the sort and written to the working dataset.

Maintain Order of Duplicates

Specifies that, for observations that are exactly the same, the order of the duplicate observations in the sorted dataset remain the same as in the original dataset.

Top Select block

Enables you to create a working dataset containing a dependent variable and its most influential independent variables.

Creating a working dataset enables you to reduce the number of variables as input to model training, for example when growing a decision tree or for logistic regression.

The dataset is configured using the **Configure Top Select** dialog box. To open the **Configure Top Select** dialog box, double-click the **Top Select** block.

Choose a **Dependent Variable** from the drop-down list.

To choose an independent variable, two lists are presented: **Unselected Independent Variables** showing available independent variables, and **Selected Independent Variables** showing chosen independent variables. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous

list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

When selecting independent variables:

- You should avoid selecting an independent variable with an entropy variance of 1 (one) as the values will typically be too random to provide a good predictive power and may lead to overfitting a model.
- You should avoid selecting an independent variable where the entropy value is 0 (zero) as this has a very low predictive power.
- Typically, you would select a group of independent variables where the difference in the entropy variance of the variables is small.

For more information about how the entropy variance is calculated see *Predictive power criteria* [↗](#) (page 102).

Transpose block

Enables you to create a new dataset with transposed variables.

Enables a dataset to be transposed, converting columns (variables) to be rows (observations) and vice versa.

Note:

WPS imposes a 32,000 variable (column) limit on datasets. If a transpose would result in this limit being breached, the transpose block will warn you that the dataset will be truncated.

Configure Transpose

Double-click on the transpose block to open the **Configure Transpose** dialog box. It has three tabs for variable selection:

- **Transpose Variables:** Allows you to specify which variables from the source dataset to Transpose. Only specified variables will appear in the output dataset.
- **Keep Variables:** Allows you to specify which variables from the source dataset to keep the same.

- **By Variables:** Specifying a source dataset variable creates a new dataset with the first variable as this specified **By Variable** and the observations grouped by the **By Variable**. An additional column (or columns) is added to the end of the dataset, listing corresponding observations from variables that match each **By Variable** value.

The above tabs all work in the same way, presenting lists of **Unselected Variables** and **Selected Variables**. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Variable Names

Allows variables created by the Transpose block to be named.

- **Output Variable Names:** Defines the name of each output variable.
 - **Prefix:** Adds the specified string to the beginning of each new column number. If both prefix and suffix are left blank, the variables will be labelled as COL1, COL2 etc.
 - **Suffix:** Adds the specified string to the end of each new column number. If both prefix and suffix are left blank, the variables will be labelled as COL1, COL2 etc.
 - **Use ID variable:** Specifies which variable's observations from the source dataset are used to name the variables in the output dataset.

When setting the above values, if any resulting variable name would be too long, a warning will be given.

- **Transposed Variables:** Defines:
 - **Names variable:** Sets the name of the names variable in the new dataset.
 - **Labels variable:** Sets the name of the labels variable in the new dataset.

Examples

Below are examples for each transpose function. In all examples, the **Names variable** and **Labels variable** have been left as their default values of `_NAME_` and `_LABEL_`. The prefix and suffix are left blank, resulting in output columns being labelled as COL1, COL2, etc.

Input Dataset

The input dataset is a simple list of fruits with variables (columns) for fruit, colour and weight:

Fruit	Colour	Weight
banana	yellow	100
apple	green	120
blackberry	black	5
strawberry	red	15

Transpose Variables

When every variable in the input dataset is selected for transposition, the output dataset is as follows:

NAME	_LABEL_	COL1	COL2	COL3	COL4
Fruit	Fruit	banana	apple	blackberry	strawberry
Colour	Colour	yellow	green	black	red
Weight	Weight	100	120	5	15

When only `Colour` is selected for transposition, the output dataset is as follows:

NAME	_LABEL_	COL1	COL2	COL3	COL4
Colour	Colour	yellow	green	black	red

Keep Variables

This example selects every variable in the input dataset for transposition (as in the first example above) and in addition to this, chooses `Colour` as a **Keep Variable**.

Colour	_NAME_	_LABEL_	COL1	COL2	COL3	COL4
yellow	Fruit	Fruit	banana	apple	blackberry	strawberry
green	Colour	Colour	yellow	green	black	red
black	Weight	Weight	100	120	5	15
red						

By Variables

This example uses an extended dataset with five additional fruits that share their colours with fruits in the original dataset. This example transposes `Fruit` and chooses `Colour` as a **By Variable**.

Colour	_NAME_	_LABEL_	COL1	COL2	COL3	COL4
black	Fruit	Fruit	blackberry			
green	Fruit	Fruit	apple	pear		
red	Fruit	Fruit	strawberry	cherry	raspberry	mango
yellow	Fruit	Fruit	banana	lemon		

Code Blocks group

Contains blocks that enable you to add a new program to the Workflow.

Although programming is not a prerequisite for creating a Workflow, if you have the required programming skills you can carry out more advanced tasks in a Workflow, using one or more of the available coding blocks.

Python Language block ↗	235
Enables you to use Python language programs in a Workflow.	
R Language block ↗	236
Enables you to use R language programs in a Workflow.	
SAS Language block ↗	238
Enables you to use SAS language programs in a Workflow.	
SQL Language block ↗	239
Enables you to create a dataset using the SQL language <code>SELECT</code> statements in a Workflow.	

Python Language block

Enables you to use Python language programs in a Workflow.

To use the **Python Language** block, you must have a Python interpreter installed and correctly configured on the local or remote host on which the Workflow Engine is run. As a minimum, you must include the *pandas* module in the Python installation.

The **Python Language** block can be used to access either existing Python language modules such as *scikit-learn* or your own modules for your Workflows.

Reference names for the input dataset are displayed in the **Inputs** list. Output working dataset names are displayed in the **Outputs** list.

Workflow interaction with the Python language, including managing input and output datasets is configured using the **Configure Python Language** dialog box. To open the **Configure Python Language** dialog box, double-click the **Python Language** block

Python Language editor

This editor is used to enter the code for the Python language program. Program code must follow the same indentation rules as Python language programs created outside a Workflow.

All code for the program must be present in the editor, you cannot import program code from an external file. Modules located in the Python site package folders can be imported into your program.

Inputs

Lists the input datasets and variables in the dataset. You can drag a name label to the Python Language editor rather than typing a label.

Outputs

Lists the pandas DataFrames created in the Python language program that are made available to other blocks in the Workflow. You can drag a name label to the Python Language editor rather than typing a label.

Input datasets

Input datasets displayed in the **Inputs** list represent the working datasets connected to the **Python Language** block.

To create a new input dataset, on the Workflow canvas, drag a connector from the **Output** port of the working dataset to the **Input** port of the **Python Language** block

To modify the dataset name click **Edit** () , and enter a new name for the dataset in the **Inputs** list. If the dataset is already referenced in your Python language program, the references are not automatically updated and must be manually modified for the **Python Language** block to run successfully.

Output datasets

Output datasets displayed in the **Outputs** list represent the mapping between a dataset created or modified in the Python language program, and the working dataset made available to other blocks in the Workflow.

To create a new output working dataset, click **Add** () . To use the new output dataset reference in the Python Language editor, drag the label to the editor.

To delete the output dataset name click **Delete** () . If the dataset is already referenced in your Python language program, the references are not automatically updated and the references to the dataset must be manually modified for the **Python Language** block to run successfully.

To modify the dataset name click **Edit** () , and enter a new name for the dataset in the **Outputs** list. If the dataset is already referenced in your Python language program, the references are not automatically updated and must be manually modified for the **Python Language** block to run successfully.

R Language block

Enables you to use R language programs in a Workflow.

To use the **R Language** block, you must have an R interpreter installed and correctly configured on the local or remote host on which the Workflow Engine is run.

The **R Language** block can be used to access existing R language modules or create your own modules for your Workflows.

Workflow interaction with the R language, including managing input and output datasets is configured using the **Configure R Language** dialog box. To open the **Configure R Language** dialog box, double-click the **R Language** block

R language editor

Used to enter the code for the R language program.

All code for the program must be present in the editor, you cannot import program code from an external file using the `source()` command. Packages located in folders referenced in the R `.libPaths` variable can be imported into your program.

Inputs

Lists the input datasets and variables in the dataset. You can drag a name label to the R Language editor rather than typing a label

Outputs

Lists the data frames created in the R language program that are made available to other blocks in the Workflow. You can drag a name label to the R Language editor rather than typing a label.

Input datasets

Input datasets displayed in the **Inputs** list represent the working datasets connected to the **R Language** block.

To create a new input dataset, on the Workflow canvas, drag a connector from the **Output** port of the working dataset to the **Input** port of the **R Language** block

To modify the dataset name click **Edit** () , and enter a new name for the dataset in the **Inputs** list. If the dataset is already referenced in your R language program, the references are not automatically updated and must be manually modified for the **R Language** block to run successfully.

Output datasets

Output datasets displayed in the **Outputs** list represent the mapping between a dataset created or modified in the R language program, and the working dataset made available to other blocks in the Workflow.

To create a new output working dataset, click **Add** () . To use the new output dataset reference in the R Language editor, drag the label to the editor.

To delete the output dataset name click **Delete** () . If the dataset is already referenced in your R language program, the references are not automatically updated and the references to the dataset must be manually modified for the **R Language** block to run successfully.

To modify the dataset name click **Edit** () , and enter a new name for the dataset in the **Outputs** list. If the dataset is already referenced in your R language program, the references are not automatically updated and must be manually modified for the **R Language** block to run successfully.

SAS Language block

Enables you to use SAS language programs in a Workflow.

The **SAS Language** block provides a self-contained environment that supports all of the SAS language syntax available in the WPS engine.

Input and output datasets must be referred to in your SAS language program using macro syntax label names. For example, to use an input dataset named `Input_1`, you must refer to the dataset as `&Input_1`.

Macro reference names for the input dataset are displayed in the **Inputs** list. Output working dataset names are displayed in the **Outputs** list.

The SAS language program input and output datasets are configured using the **Configure SAS Language** dialog box. To open the **Configure SAS Language** dialog box, double-click the **SAS Language** block.

SAS Language editor

Used to enter the code for the SAS language program. All code for the program must be present in the editor, you cannot import program code from an external file.

Inputs

Lists the input datasets and variables in the dataset. You can drag a name label to the SAS Language editor rather than typing a label.

Outputs

Lists the datasets created in the SAS language program to be made available to other block in the Workflow. You can drag a name label to the SAS Language editor rather than typing a label.

Input datasets

Input datasets displayed in the **Inputs** list represent the working datasets connected to the **SAS Language** block.

To create a new input dataset, on the Workflow canvas, drag a connector from the **Output** port of the working dataset to the **Input** port of the **SAS Language** block

To modify the dataset name click **Edit** () , and enter a new name for the dataset in the **Inputs** list. If the dataset is already referenced in your SAS language program, the references are not automatically updated and must be manually modified for the **SAS Language** block to run successfully.

Output datasets

Output datasets displayed in the **Outputs** list represent the mapping between a dataset created or modified in the SAS language program, and the working dataset made available to other blocks in the Workflow.

To create a new output working dataset, click **Add** (). To use the new output dataset reference in the SAS Language editor, drag the label to the editor.

To delete the output dataset name click **Delete** (). If the dataset is already referenced in your SAS language program, the references are not automatically updated and the references to the dataset must be manually modified for the **SAS Language** block to run successfully.

To modify the dataset name click **Edit** (), and enter a new name for the dataset in the **Outputs** list. If the dataset is already referenced in your SAS language program, the references are not automatically updated and must be manually modified for the **SAS Language** block to run successfully.

SQL Language block

Enables you to create a dataset using the SQL language `SELECT` statements in a Workflow.

The **SQL Language** block is configured using the **Configure SQL Language** dialog box. To open the **Configure SQL Language** dialog box, double-click the **SQL Language** block.

SQL Language editor

The editor is used to enter SQL language `SELECT` statements.

Input datasets must be referred to in your SQL language code using SAS language macro syntax label names. For example, to use an input dataset named `Input_1`, you must refer to the dataset as `&Input_1`.

Inputs

Lists the input datasets and variables in the datasets. You can drag a name label to the SQL Language editor rather than typing a label.

Input datasets

Input datasets displayed in the **Inputs** list represent the working datasets connected to the **SQL Language** block.

To create a new input dataset, on the Workflow canvas, drag a connector from the **Output** port of the working dataset to the **Input** port of the **SQL Language** block

To modify the dataset name click **Edit** (), and enter a new name for the dataset in the **Inputs** list. If the dataset is already referenced in your SQL language statements, the references are not automatically updated and must be manually modified for the **SQL Language** block to run successfully.

Model Training group

Contains blocks that enable you to discover predictive relationships in your data.

Decision Forest block ↗	240
Provides an interface to create a <i>decision forest</i> prediction model.	
Decision Tree block ↗	249
Provides an interface to create a decision tree for classification purposes.	
Hierarchical Clustering block ↗	264
Provides an interface to create a hierarchical cluster model.	
K-Means Clustering block ↗	271
Enables you to cluster data using <i>k</i> -means clustering.	
Linear Regression block ↗	278
Enables you to create a linear regression model.	
Logistic Regression block ↗	284
Fits a logistic model between a set of effect variables and a binary dependent variable.	
MLP block ↗	293
Enables you to build a Multi-Layer Perceptron (MLP) neural network from an input dataset and use the trained network to analyse other datasets.	
Reject Inference block ↗	308
Enables you to use a <i>reject inference</i> method to address any inherent selection bias in a model by including a rejected population. The Reject Inference block takes two inputs: one from a logistic regression model and another from a dataset containing rejected records. The Reject Inference block then outputs a logistic regression model, labelled the Inference Model , that has been trained on both the performance of the accepted records and the inferred performance of the rejected records.	
Scorecard Model block ↗	315
Enables the generation of a standard scorecard (credit scorecard) and its deployment code that can be copied into a SAS language program for testing or production use.	
WoE Transform block ↗	321
Enables you to measure the influence of an independent variable on a specified dependent variable. This calculates the Weight of Evidence, or WoE.	

Decision Forest block

Provides an interface to create a *decision forest* prediction model.

A decision forest is a prediction model that uses a collection of *decision trees* to predict the value of a dependent (target) variable based on other independent (input) variables in the input dataset.

A decision forest is created and edited using the **Configure Decision Forest** dialog box. To open the **Configure Decision Forest** dialog box, double-click the **Decision Forest** block.

The **Decision Forest** block generates a report containing the summary statistics for the model and the scoring code generated by the model. To view this report, right-click the **Decision Forest Model** block and click **View Result**. The generated scoring code can be copied into a SAS language program for testing or production use.

Variable Selection panel

Enables you to specify the dependent (target) variable and independent (input) variables.

Dependent Variable

Specifies the dependent (target) variable for which the category is being calculated by the decision tree.

Dependent Variable Treatment

Specify the type of the dependent (target) variable. The supported variable types are:

Interval

Specifies a continuous dependent (target) variable.

Nominal

Specifies a discrete dependent (target) variable with no implicit ordering.

Ordinal

Specifies a discrete dependent (target) variable with an implicit category ordering.

Independent Variables

The two boxes in the **Variable Selection** panel show **Unselected Independent Variables** and **Selected Independent Variables**. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

For categorical dependent variables, the *Entropy Variance* column shows how well each independent variable can predict the value of the selected dependent (target) variable. *Entropy Variance* is in the range of 0 – 1. If a variable in a dataset has a high entropy variance, it means that the variable is a good predictor of the dependent variable. However where the entropy variance is very high, or 1, the variable might not be a good predictor as using this variable could lead to over-fitting the model.

The Variable Treatment for each variable enables you to specify that variable's type. Only applicable types are available to choose from. The full list of types is:

Interval

Specifies a continuous independent (input) variable.

Nominal

Specifies a discrete independent (input) variable with no implicit ordering. When partitioning this variable into nodes in a tree, any category can be merged with any other category.

Ordinal

Specifies a discrete independent (input) variable with an implicit category ordering. When partitioning this variable into nodes in a tree, only adjacent categories can be merged together.

Preferences panel

Specifies the options for the creation of the decision forest.

Number of trees

Specifies the number of decision trees in the forest.

Minimum improvement

Specifies the minimum improvement (decrease) in Gini Impurity required for a node to be split.

Exclude Missings

When selected, excludes observations with missing values when building the forest.

Classifier combination

Choose **Vote** to classify observations by a majority vote.

Choose **Mean Probability** to classify observations by a combination of the probabilities calculated by each tree.

This option is unavailable if the dependent variable is an **Interval** variable (a regression forest is used).

Criterion

Specifies how the variable from the subset of specified dependent variables that gives the largest overall decrease in *Gini Impurity* is used to split nodes in all trees.

Note:

This setting is for information only and cannot be changed.

Bootstrap

Specifies that the trees in the forest are constructed using bootstrap re-sampling of the input dataset.

Note:

This setting is for information only and cannot be changed.

Inputs at split

Defines the number of input variables to randomly select each time a node is split.

Use default

Specifies the default number of input variables is selected.

For N selected independent variables, the default value is $\lfloor \sqrt{N} \rfloor$ for classification trees and $\lfloor N/3 \rfloor$ for regression trees.

Number of inputs at each split

Specifies the number of input variables.

Split size

Enables you to specify the minimum number of observations for a node to be split either as a proportion of the input dataset or an absolute number.

Minimum split size

Specifies the absolute minimum number of observations in a node.

Minimum split size as ratio

Specifies the minimum number of observations as a proportion of the input dataset.

Frequency and weight variables**Frequency variable**

Specifies a variable in the input dataset giving the frequency associated with each observation.

Weight variable

Specifies a variable in the input dataset giving the prior weight associated with each observation.

Decision Forest Model

Displays a summary of the decision forest model and its results, as well as providing scoring code in the SAS language.

Decision Forest Results tab

Model Information

Provides a summary of the model.

Target Summary

Shows the dependent (target) variable that the decision forest is required to predict.

Input Summary

Summarises the independent (input) variables, their types, categories and the other properties.

Out-of-Bag Classification Table

A confusion matrix that shows the out-of-bag (OOB) error estimate from the decision forest. This is the number of observations in the training dataset that were correctly and incorrectly classified using OOB classification.

To calculate the predicted OOB classification for an observation, the observation is classified using each tree for which the observation was out-of-bag (that is, each tree for which the observation wasn't selected to train that tree). The most popular value is chosen as the predicted OOB classification for that observation. The OOB error estimate is the number of observations which are correctly and incorrectly classified using this algorithm.

Scoring Code tab

The **Scoring Code** tab displays code for the decision forest in a specified language with specified variable names. SAS language code can be used in a `DATA` step in the **SAS Language** block. In place of `xxxx .xxxx`, insert a reference to the saved model (see [Saving the Model](#) (page 244)).

Saving the Model

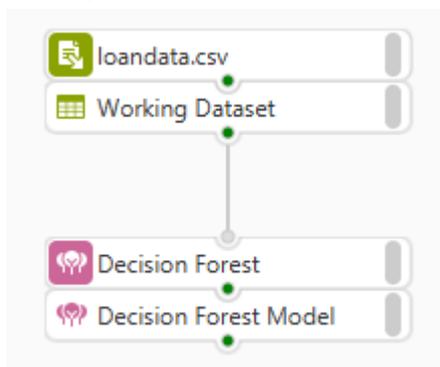
To save the Decision Forest model, right-click on the **Decision Forest Model** block and select **Save Model**. Specify a filename and location.

Decision Forest block basic example

This example demonstrates how the **Decision Forest** block can be used to model a binary variable.

In this example, a dataset containing information about loans, named `loandata.csv` is imported into a blank Workflow. The dataset contains information about the default status of loans using a binary *Default* variable. The **Decision Forest** block uses this dataset to estimate the likelihood for each loan to default.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Loan dataset

This example uses a dataset about loans, named `loandata.csv`. Each observation in the dataset describes a past loan and the person who took loan out, with variables such as *Income*, *Loan_Period*, *Other_Debt* etc.

Using the Decision Forest block to create a Decision Forest model.

Using the **Decision Forest** block to predict loan default.

1. Import the `loandata.csv` dataset onto a blank Workflow canvas. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **Decision Forest** block onto the Workflow canvas, beneath the `loandata.csv` dataset.
3. Click on the `loandata.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Decision Forest** block.

Alternatively, you can dock the **Decision Forest** block onto the dataset block directly.

4. Double-click on the **Decision Forest** block.

A **Configure Decision Forest** dialog box opens.

5. From the **Configure Decision Forest** dialog box:
 - a. In the **Dependent variable** drop-down list, select **Default**.
 - b. In the **Dependent variable treatment** drop-down list, select **Nominal**.
 - c. From the **Unselected Independent Variables** box, double-click on the variables *Age*, *Part_Of_UK*, *Income* and *Other_Debt* to move them to the **Selected Independent Variables** box.
 - d. Click **OK**.

The **Configure Decision Forest** dialog box closes. A green execution status is displayed in the **Output** port of the **Decision Forest** block and the model results, `Decision Forest Model`.

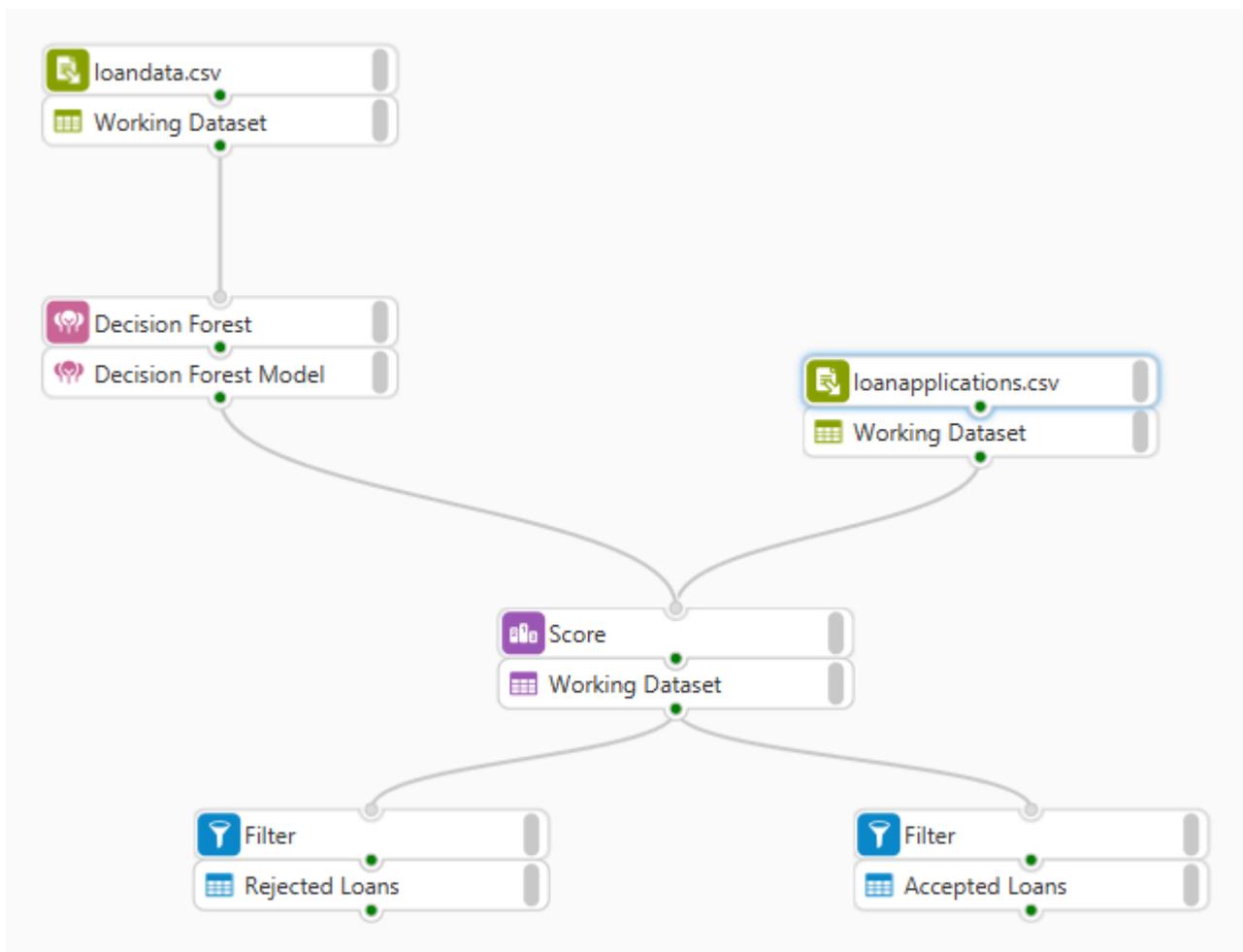
You now have a decision forest model that predicts the likelihood of each loan defaulting.

Decision Forest block example: Accepting or rejecting loan applications

This example demonstrates how the **Decision Forest** block can make predictions on loan defaulting, assisting with the acceptance or rejection of loan applications.

In this example, a dataset containing information about loans, and a dataset of loan applications, named `loandata.csv` and `loanapplications.csv` respectively are imported into a blank Workflow. The **Decision Forest** block is then configured to output a model that estimates the likelihood for a loan to default. The model is then applied to another dataset, `loanapplications.csv`, using a **Score** block. Two **Filter** blocks are then used to divide the dataset of loan applications into two datasets: one describing accepted loans and the other rejected loans, based on the value of the defaulting prediction. These new datasets are named `Rejected Loans` and `Accepted Loans3` accordingly.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Loan dataset

This example uses a dataset about loans, named `loandata.csv`. Each observation in the dataset describes a past loan and the person who took loan out, with variables such as *Income*, *Loan_Period*, *Other_Debt* etc.

Loan applications dataset

This example uses a dataset about loan applications, named `loanapplications.csv`. Each observation in the dataset describes an application for a loan and the person taking the loan out. It contains the same variables as `loandata.csv` except for the lack of a *Default* variable.

Using the Decision Forest block to accept or reject loan applications.

Training a **Decision Forest** block and using it to make predictions on new data.

1. Import the `loandata.csv` and `loanapplications.csv` datasets into a blank Workflow. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **Decision Forest** block onto the Workflow canvas, beneath the `loandata.csv` dataset.
3. Click on the `loandata.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Decision Forest** block.

Alternatively, you can dock the **Decision Forest** block onto the dataset block directly.

4. double-click on the **Decision Forest** block.

A **Configure Decision Forest** dialog box opens.

5. From the **Configure Decision Forest** dialog box **Variable selection** tab:
 - a. In the **Dependent variable** drop-down list, select **Default**.
 - b. In the **Dependent variable treatment** drop-down list, select **Nominal**.
 - c. From the **Unselected Independent Variables** box, double-click on the variables *Age*, *Part_Of_UK*, *Income* and *Other_Debt* to move them to the **Selected Independent Variables** box.
6. Click on the **Preferences** tab and change the following:
 - a. **Number of trees** to 20.
 - b. **Classifier combination** as **Mean Probability**.
 - c. Select **Minimum split size as ratio** and set its value to 5.
 - d. Click **OK**.

The **Configure Decision Forest** dialog box saves and closes. A green execution status is displayed in the **Output** port of the **Decision Forest** block and the model results, *Decision Forest Model*.

7. From the **Scoring** grouping in the Workflow palette, click and drag a **Score** block onto the Workflow canvas, below the new **Decision Forest Model** block and the `loanapplications.csv` dataset block.

A **Score** block and empty dataset block are created on the Workflow canvas. Note that the **Score** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires a model and a dataset input.

8. Click on the `loanapplications.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Score** block.
9. Click on the **Decision Forest Model Output** port and drag a connection towards the **Input** port of the **Score** block.

The execution status of the **Score** block turns green and its dataset output is populated with the original dataset and the scored variables.

10. From the **Data Preparation** grouping in the Workflow palette, click and drag two **Filter** blocks onto the Workflow canvas.
11. Click on the **Score** block **Output** port and drag a connection towards the **Input** port of each **Filter** block.

12. Configure the first **Filter** block:

- a. Double-click on a **Filter** block.

A **Filter Editor** view opens.

- b. Click on the **Basic** tab.
- c. From the **Variable** drop-down box, select **P_1**.
- d. From the **Operator** drop-down box, select **<**.
- e. In the **Value** box, enter `0.5`.
- f. Save your **Filter** block. To save, either click the Save button () or press **Ctrl+S**.
- g. Close the **Filter Editor** view.

The **Filter Editor** view closes. A green execution status is displayed in the **Output** port of the **Filter** block and the filtered dataset, **Working Dataset**.

- h. Right-click on the **Filter** block's output dataset block and click **Rename**. Type `Rejected Loans` and click **OK**.

13. Configure the second **Filter** block:

- a. Double-click on the other **Filter** block.

A **Filter Editor** view opens.

- b. Click on the **Basic** tab.

- c. From the **Variable** drop-down box, select **P_1**.

- d. From the **Operator** drop-down box, select **>=**.

- e. In the **Value** box, enter **0.5**.

- f. Save your **Filter** block. To save, either click the Save button () or press **Ctrl+S**.

- g. Close the **Filter Editor** view.

The **Filter Editor** view closes. A green execution status is displayed in the **Output** port of the **Filter** block and the filtered dataset, **Working Dataset**.

- h. Right-click on the **Filter** block's output dataset block and click **Rename**. Type **Accepted Loans** and click **OK**.

You now have a decision forest model that predicts if a loan applicant will default and filters data into acceptances and rejections.

Decision Tree block

Provides an interface to create a decision tree for classification purposes.

Decision trees are created and edited using the **Decision Tree Editor** view and **Decision Tree Preferences** dialog box. Double-clicking a **Decision Tree** block will open the **Decision Tree Editor** view; if the block is new with no basic information configured, the **Decision Tree Preferences** dialog box will also open. Once basic information to build a tree has been configured, double-clicking the **Decision Tree** block will just open the **Decision Tree Editor** view. To open the **Decision Tree Preferences** dialog box manually, click the preferences icon () at the top right of the **Decision Tree Editor** view.

You can use the **Decision Tree Editor** view **Tree** tab to grow the tree manually based on the optimal binning measure of the selected independent variables, or automatically grow one level of the tree at a time or a full decision tree using the growth preference algorithm specified in the **Decision Tree Preferences** dialog box.

Growing a decision tree requires a connected source dataset, although without a connected dataset, trees may still be viewed and pruned (Unsplit). If a dataset is reconnected after a period of disconnection, the decision tree will be recalculated.

Decision Tree editor view

Displays graphical, tabular and coded representations of the decision tree and information about specific nodes or tree levels as the tree is grown. The decision tree editor also stores a history of changes to the decision tree.

Decision Tree preferences

Enables you to specify the independent and dependent variables and set options for creating the decision tree.

Opening the Decision Tree preferences dialog box

The Decision Tree **Preferences** dialog box is opened in three ways:

- When first creating a decision tree, the Decision Tree **Preferences** dialog box will open automatically.
- By clicking the Preferences () button in the **Decision Tree Editor** view **Tree** tab.
- By right-clicking on the Decision Tree block in the Workflow and clicking **Edit Decision Tree**.

The contents of the Decision Tree **Preferences** dialog box are described below.

Variable Selection panel

Enables you to specify the dependent (target) variable and category you want to predict based on other independent (input) variables in the input dataset.

Dependent Variable

Specifies the dependent (target) variable for which the category is calculated by the decision tree.

Tree type

Choose from **Classification** or **Regression**. Choice will be limited depending on the type of the **Dependent Variable** chosen.

Target Category

For classification trees, specifies the classification of the dependent variable the decision tree will predict using the selected independent (input) variables selected in the **Independent Variables** list.

Weight variable

To weight every node's calculated frequencies, select the check box and choose a weight variable from the drop-down list.

Independent Variables

The two boxes in the **Variable Selection** panel show **Unselected Independent Variables** and **Selected Independent Variables**. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

For categorical dependent variables, the *Entropy Variance* column shows how well each independent variable can predict the value of the selected dependent (target) variable. *Entropy Variance* is in the range of 0 – 1. If a variable in a dataset has a high entropy variance, it means that the variable is a good predictor of the dependent variable. However where the entropy variance is very high, or 1, the variable might not be a good predictor as using this variable could lead to over-fitting the model.

The Variable Treatment for each variable enables you to specify that variable's type. Only applicable types are available to choose from. The full list of types is:

Interval

Specifies a continuous independent (input) variable.

Nominal

Specifies a discrete independent (input) variable with no implicit ordering. When partitioning this variable into nodes in a tree, any category can be merged with any other category.

Ordinal

Specifies a discrete independent (input) variable with an implicit category ordering. When partitioning this variable into nodes in a tree, only adjacent categories can be merged together.

Growth Preferences

Specifies the decision tree growth algorithm and associated growth preferences. The **Growth Algorithm** specifies the method used to construct the decision tree, and can be one of C4.5, CART, or BRT, depending on the type of the chosen dependent variable. Each algorithm has its own specific options.

C4.5

Specifies the C4.5 algorithm should be used to build the decision tree. You can then specify options for the algorithm as follows:

- **Maximum depth:** Specifies the maximum depth of the decision tree.
- **Minimum node size (%):** Specifies the minimum number of observations in a decision tree node, as a proportion of the input dataset.
- **Merge categories:** When selected, merges categories into single leaf nodes.
- **Prune:** Select to specify that nodes which do not significantly improve the predictive accuracy are removed from the decision tree to reduce tree complexity.

- **Prune confidence level:** Specifies the confidence level for pruning, expressed as a percentage.

CART

Specifies the CART algorithm should be used to build the decision tree. Leaf nodes will be merged according to the algorithm's methodology. You can specify options for the algorithm as follows:

- **Criterion:** Specifies the criterion used to split nodes in the decision tree. The supported criteria are:
 - **Gini:** Specifies that *Gini Impurity* is used to measure the predictive power of variables.
 - **Ordered twoing:** Specifies that the *Ordered Twoing Index* is used to measure the predictive power of variables.
 - **Twoing:** Specifies that the *Twoing Index* is used to measure the predictive power of variables.
- **Prune:** Select to specify that nodes which do not significantly improve the predictive accuracy are removed from the decision tree to reduce tree complexity.
- **Pruning Method:** Specifies the method to be used when pruning a decision tree. The supported methods are:
 - **Holdout:** Specifies that the input dataset is randomly divided into a test dataset containing one third of the input dataset and a training dataset containing the balance. The output decision tree is created using the training dataset, then pruned using risk estimation calculations on the test dataset.
 - **Cross validation:** Specifies that the input dataset is divided as evenly as possible into ten randomly-selected groups, and the analysis repeated ten times. The analysis uses a different group as test dataset each time, with the remaining groups used as training data. Risk estimates are calculated for each group and then averaged across all groups. The averaged risk values are used to prune the final tree built from the entire dataset.
- **Maximum Depth:** Specifies the maximum depth of the decision tree.
- **Minimum node size (%):** Specifies the minimum number of observations in a decision tree node, as a proportion of the input dataset.
- **Minimum improvement:** Specifies the minimum improvement in impurity required to split decision tree node.
- **Exclude missings:** Specifies that observations containing missing values are excluded when determining the best split at a node.

BRT

Specifies the algorithm options for binary response trees. Specifies the BRT algorithm should be used to build a binary decision tree. Leaf nodes will be merged according to the algorithm's methodology. You can specify options for the algorithm:

- **Criterion:** Specifies the criterion used to split nodes in the decision tree. The supported criteria are:
 - **Chi-squared:** Specifies that *Pearson's Chi-Squared* statistic is used to measure the predictive power of variables.
 - **Entropy variance:** Specifies that *Entropy Variance* is used to measure the predictive power of variables.
 - **Gini variance:** Specifies that *Gini Variance* is used to measure the predictive power of variables by measuring the strength of association between variables.
 - **Information value:** Specifies that *Information Value* is used to measure the predictive power of variables.
- **Maximum depth:** Specifies the maximum depth of the decision tree.
- **Select minimum node size by ratio:** When selected enables you to specify the minimum number of observations in a node as a proportion of the input dataset. When cleared, enables you to specify the absolute minimum number of observations in a node.
- **Minimum node size (%):** Specifies the minimum number of observations in a decision tree node, as a proportion of the input dataset.
- **Minimum node size:** When selected enables you to specify the minimum split size of a decision tree node as a proportion of the input dataset. When cleared, enables you to specify the absolute minimum split size of a decision tree node.
- **Select minimum split size by ratio:** When selected enables you to specify the minimum number of observations for the node to be split as a proportion of the input dataset. When cleared, enables you to specify the absolute minimum number of observations in a node.
- **Minimum split size (%):** Specifies the minimum number of observations a decision tree node must contain for the node to be split, as a proportion of the input dataset.
- **Minimum split size:** Specifies the absolute minimum number of observations a decision tree node must contain for the node to be split.
- **Allow same variable split:** Specifies that a variable can be used more than once to split a decision tree node.
- **Open left:** For a continuous variable, when selected, specifies that the minimum value in the node containing the very lowest values is $-\infty$ (minus infinity). When clear, the minimum value is the lowest value for the variable in the input dataset.
- **Open right:** For a continuous variable, when selected, specifies that the maximum value in the node containing the very largest values is ∞ (infinity). When clear, the maximum value is the largest value for the variable in the input dataset.
- **Merge missing bin:** Specifies that missing values are considered a separate valid category when binning data.
- **Monotonic Weight of Evidence:** Specifies that the Weight of Evidence value for ordered input variables is either monotonically increasing or monotonically decreasing.
- **Exclude missings:** Specifies that observations containing missing values are excluded when determining the best split at a node.
- **Initial number of bins:** Specifies the number of bins available when binning variables.

- **Maximum number of optimal bins:** Specifies the maximum number of bins to use when optimally binning variables.
- **Weight of Evidence adjustment:** Specifies the adjustment applied to weight of evidence calculations to avoid invalid results for pure inputs.
- **Maximum change in predictive power:** Specifies the maximum change allowed in the predictive power when optimally merging bins.
- **Minimum predictive power for split:** Specifies the minimum predictive power required for a decision tree node to be split.

Tree tab

Displays the decision tree and associated information.

Tree

The main part of the **Tree** tab displays a graphical representation of the decision tree as it grows. Each node is numbered, with numbering starting at zero at the root, with each subsequent level then numbered from left to right following on from the previous level. For discrete dependent variables, each node shows a frequency breakdown of the target categories. For continuous variables, each node shows the mean and LSD (least squares deviation) of the dependent variable for that node.

Tree nodes are colour coded as follows:

- **Regression Trees:** Nodes are shaded according to their dependent variable's mean value. The node with a mean closest to the minimum value is shaded blue, and the node with a mean closest to the maximum is shaded red, with gradations in between.
- **Classification Trees:** Nodes are shaded according to the percentage of observations for that node that satisfy the target category. The node with the lowest percentage is shaded blue, and the node with the highest percentage is shaded red, with shades in between to represent other percentages.

The growth of the tree is determined by options specified in the **Node properties** panel. Other Decision Tree settings are configured using the Preferences window, accessed by clicking the **Preferences** button (two cogs symbol).

The Node Information panel shows information about the selected node. If a continuous dependent variable is being used, two other panels are available: **Node Frequencies** and **Peer Comparison**. The **Node Frequencies** panel shows a frequency breakdown of the target categories for the selected node. The **Peer Comparison** panel, which is collapsed by default, displays information about a selected node compared to other nodes on the same tree level.

To prune a tree (remove child branches), right-click on a parent branch and click **Unsplit**.

To switch between the tree showing unweighted frequencies or weighted frequencies, right-click on the tree itself and select **show unweighted frequencies** or **show weighted frequencies**. This will also change which frequency value is displayed in the **Node Frequencies** panel.

Node Information panel

Displays information about the node currently-selected.

Node

Displays the label of the node. If the node is the root of the tree, the label `<root>` is displayed, otherwise the **Node label** specified in **Node Properties** is displayed.

Size

Displays the total number of observations in the current node.

% of Population

Displays the size of the node as a percentage of the input dataset.

Node Frequencies panel (continuous dependent variables only)

Displays the frequency or weighted frequency of the dependent variable categories in the selected node. Switching between weighted and unweighted frequencies is done by right-clicking on the tree and selecting **show unweighted frequencies** or **show weighted frequencies**.

The frequency panel can show either a histogram showing the frequency percentage for each category, or a table showing the frequency and associated observation count. Click the icons at the top of the panel to switch between the histogram and table views.

The histogram chart can be edited and saved. Click **Edit chart**  to open a the Chart Editor from where the chart can be saved to the clipboard. Frequency data for the selected node can be saved, click **Copy data to clipboard**  to save the table of data.

Peer comparison panel (continuous dependent variables only)

Displays the frequency of the dependent variable categories in the selected node and other nodes at the same tree level (peer nodes). The frequency for each category in all peer nodes can be displayed as either a histogram showing the frequency percentage for each category, or a table showing the frequency and associated observation count in each node.

The peer comparison chart can be edited and saved. Click **Edit chart**  to open a the Chart Editor from where the chart can be saved to the clipboard. Frequency data for the selected node and peer nodes can be saved, click **Copy data to clipboard**  to save the table of data.

Node Properties panel

Enables you to grow a tree and specify how new tree level nodes are split.

Split

Displays the independent variable from the **Split Variable** list used to split the parent node to create the tree level containing the selected node.

Node Label

Specifies the label displayed in the graphical decision tree and the **Node Information** panel for the node selected in the graphical tree. The default label is the value or values of the displayed split variable used to create the node. If you modify the label, click **Default** to return the label to the Workbench-generated value.

Grow 1 Level

Grows the next level of the tree using the **Growth Algorithm** specified in the **Growth Preferences** panel of the Decision tree preferences.

Grow

Grows a complete decision tree using the **Growth Algorithm** specified in the **Growth Preferences** panel of the Decision tree preferences.

Optimal split

Will grow one level of the tree using the Optimal Binning **Measure** specified in the **Binning** panel of the **WPS** section of the **Preferences** dialog box.

Split Variable

Displays the variable used to split the selected node and the categories used to determine the split points. You can change the created nodes using one of the available binning methods or by selecting a different **Split variable**. The available binning methods, depending on variable type, are optimal binning , equal width binning , equal height binning , or Winsorized binning .

Map panel

The map panel at the bottom right shows how the tree display area, represented by a grey rectangle, relates to the tree as a whole, shown with blue and red blocks. The main tree display area can be moved by clicking and dragging the grey rectangle.

Summary tab

Gives information about the decision tree.

Model Information

Provides a summary of the Decision Tree, giving its type, number of nodes and leaf nodes, and maximum width and depth (measured in nodes).

Dependent Variable

Gives details of the dependent variable

Independent Variable

Gives details of the independent variable or variables.

Classification Table

Gives predicted category frequencies from the decision tree.

Node Table

The Node Table is a tabular representation of the decision tree, with every leaf node as a row.

Sorting the table

The table can be sorted by clicking on the variable you want to sort by.

Exporting to Excel

The **Export to excel** button will export the table's contents to an MS Excel `xlsx` file with a specified location and filename. The sort order of the table view has no influence on the sorted order of the generated Excel file.

Dependent variable value

Sets the value of the dependent variable. The dependent variable is specified in *Decision Tree preferences* [↗](#) (page 250).

Column Definitions

Node Table Column	Description
ID	
Condition	A logical statement defining the leaf node (row).
Population	The total number of observations matching the condition.
% of Population	The percentage of the total population that match the condition.
Cumulative Population	The cumulative total of the population that match the condition.
Cumulative % of Population	The cumulative percentage of the total population that match the condition.
<dependent variable> = <dependent variable value>	The number of observations that match the condition and this statement. The dependent variable is specified in the <i>Decision Tree preferences</i> ↗ (page 250) window and the Dependent variable value is specified in the drop-down list above the table.

Node Table Column	Description
% <dependent variable> = <dependent variable value>	The percentage of observations in the entire population that match the condition and this statement. The dependent variable is specified in the <i>Decision Tree preferences</i> ↗ (page 250) window and the Dependent variable value is specified in the drop-down list above the table.
Cumulative % <dependent variable> = <dependent variable value>	The cumulative percentage of observations in the entire population that match the condition and this statement. The dependent variable is specified in the <i>Decision Tree preferences</i> ↗ (page 250) window and the Dependent variable value is specified in the drop-down list above the table.
Proportion % <dependent variable> = <dependent variable value>	The percentage of observations in the node that match the condition and this statement. The dependent variable is specified in the <i>Decision Tree preferences</i> ↗ (page 250) window and the Dependent variable value is specified in the drop-down list above the table.
Lift % <dependent variable> = <dependent variable value>	The lift percentage of observations that match the condition and this statement. Lift is defined as the ratio between matches for the node and the entire population.
Cumulative lift % <dependent variable> = <dependent variable value>	The cumulative lift percentage of observations that match the condition and this statement. Lift is defined as the ratio between matches for the node and the entire population.

Scoring Code tab

Displays code for the decision tree in a specified language with specified variable names. SAS language code can be used in a `DATA` step in the **SAS Language** block.

Choosing the language

From the **Select Language** list box, select the required language as either **SAS** or **SQL**.

Choosing score variable names

Click **Options**, then in **Source Generation Options**, click the score variable you want to change and type the new name. When you are happy with all the score variable names, click **OK**.

History tab

Stores all changes to the decision tree.

Events

Lists decision tree change events and has three columns:

Event

Lists events by number, starting with the oldest item (labelled '1') and concluding with the newest.

Action

Describes what took place in each event.

Node

If applicable, gives the node for the event.

Click on a column heading to numerically sort entries.

Clicking on an event will display data from that even in the other four panels, described below.

Click the copy to clipboard button () to copy the entire contents of the panel to the clipboard.

Growth Settings

Lists the growth settings in place for the selected event. These settings are set in the **Growth Preferences** tab of *Decision Tree preferences* [↗](#) (page 250).

Click the copy to clipboard button () to copy the entire contents of the panel to the clipboard.

Decision Tree Settings

Lists the variables selection for the selected event. These settings are set in the **Variable Selection** tab of *Decision Tree preferences* [↗](#) (page 250).

Click the copy to clipboard button () to copy the entire contents of the panel to the clipboard.

Binning Settings

Lists the binning settings for the selected event. These settings are set in the **Binning Preferences** tab of *Decision Tree preferences* [↗](#) (page 250).

Click the copy to clipboard button () to copy the entire contents of the panel to the clipboard.

Dataset Variables

Lists any changes in the Decision Tree block's input dataset variables.

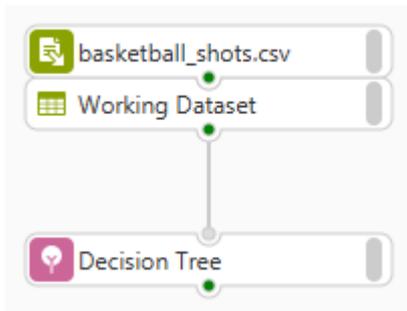
Click the copy to clipboard button () to copy the entire contents of the panel to the clipboard.

Decision Tree block basic example

This example demonstrates how the **Decision Tree** block can be used to model a binary variable.

In this example, a dataset containing information about basketball shots, named `basketball_shots.csv` is imported into a blank Workflow. The dataset contains information about a basketball shot and whether it scored through the *Scored* variable. The **Decision Tree** block uses this dataset to estimate the likelihood for each basketball shot to score.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Basketball dataset

This example uses a dataset about basketball shots, named `basketball_shots.csv`. Each observation in the dataset describes an attempt to score and the person shooting, with variables such as *height*, *weight*, *distance_feet* etc.

Using the Decision Tree block to create a Decision Tree model

Using the **Decision Tree** block to predict loan default.

1. Import the `basketball_shots.csv` dataset onto a blank Workflow canvas. Importing a CSV file is described in [Text File Import block](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **Decision Tree** block onto the Workflow canvas, beneath the `basketball_shots.csv` dataset.
3. Click on the `basketball_shots.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Decision Tree** block.

Alternatively, you can dock the **Decision Tree** block onto the dataset block directly.

4. Double-click on the **Decision Tree** block.

A **Decision Tree Editor** view opens, along with the Decision Tree **Preferences** dialog box.

5. From the **Preferences** window:

- a. In the **Dependent variable** drop-down list, select **Score**.
- b. In the **Target category** drop-down list, select **1**.
- c. From the **Unselected Independent Variables** box, double-click on the variables *distance_feet*, *angle*, *height*, *weight* and *position* to move them to the **Selected Independent Variables** box.
- d. Click **OK**. The Decision Tree **Preferences** dialog box closes. Right-click on the **0:Score** node and click **Grow (C4.5)** to train the model.

The **Decision Tree Editor** view is populated with further nodes that make up the decision tree model.

- e. Save your **Decision Tree** block. To save, either click the Save button () or press **Ctrl+S**.
- f. Close the **Decision Tree Editor** view.

The **Decision Tree** window closes. A green execution status is displayed in the **Output** port of the **Decision Tree** block.

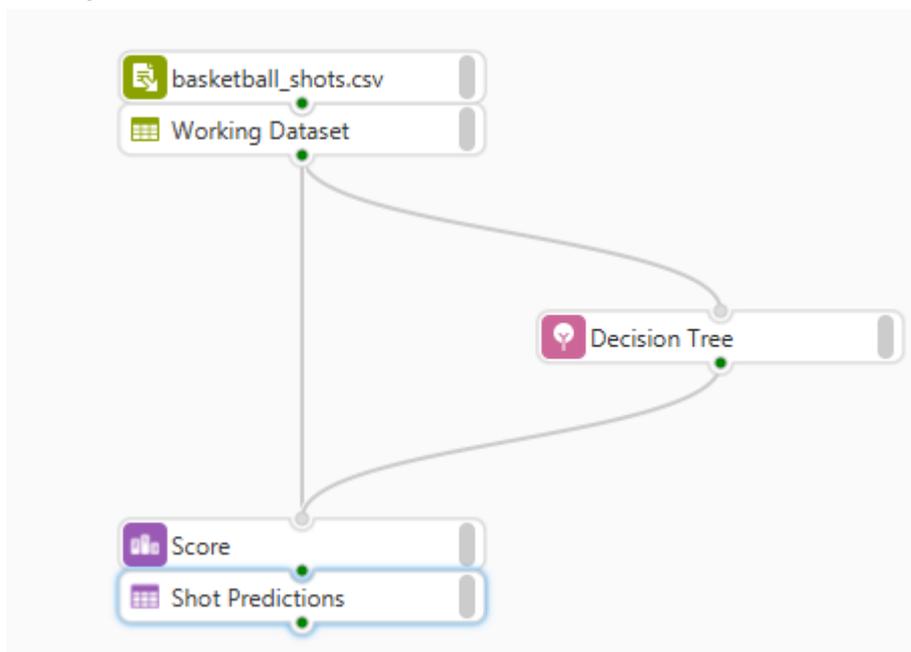
You now have a decision tree model that predicts whether a basketball shot will score.

Decision Tree block example: Manually building a decision tree

This example demonstrates how the **Decision Tree** block can be manually built.

In this example, a dataset containing information about basketball shots, named `basketball_shots.csv` is imported into a blank Workflow. The dataset contains information about a basketball shot and whether it score through the *Scored* variable. The **Decision Tree** block uses this dataset to estimate the likelihood for each basketball shot to score. The model is then applied to `basketball_shots.csv`, using a **Score** block. The **Score** block generates an output dataset `Shot Predictions`.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Basketball dataset

This example uses a dataset about basketball shots, named `basketball_shots.csv`. Each observation in the dataset describes an attempt to score and the person shooting, with variables such as *height*, *weight*, *distance_feet* etc.

Using the Decision Tree block to create a manual decision tree model

Manually building a **Decision Tree** block in the **Decision Tree Editor** view to predict a binary variable.

1. Import the `basketball_shots.csv` dataset onto a blank Workflow canvas. Importing a CSV file is described in [Text File Import block](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **Decision Tree** block onto the Workflow canvas, beneath the `basketball_shots.csv` dataset.
3. Click on the `basketball_shots.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Decision Tree** block.

Alternatively, you can dock the **Decision Tree** block onto the dataset block directly.

4. Double-click on the **Decision Tree** block.

A **Decision Tree Editor** view opens, along with the Decision Tree **Preferences** dialog box.

5. From the Decision Tree **Preferences** dialog box:
 - a. In the **Dependent variable** drop-down list, select **Score**
 - b. In the **Target category** drop-down list, select **1**
 - c. From the **Unselected Independent Variables** box, double-click on the variables *distance_feet*, *angle* and *height* to move them to the **Selected Independent Variables** box.
 - d. Click **OK**. The Decision Tree **Preferences** dialog box closes.
6. In the **Decision Tree Editor** view:
 - a. From the **Split variable** drop-down box select **distance_feet** and click on the **Calculate optimal binning** button (.
 - b. Click on the decision tree node labelled **5: > 19.3 AND <=20.9**, from the **Split variable** drop-down box select **angle** and click on the **Calculate equal width binning** button (.
 - c. Click on the decision tree node labelled **12: > 14.0 AND <=28.0**, from the **Split variable** drop-down box select **height** and click on the **Calculate optimal binning** button (.

The **Decision Tree Editor** view is populated with further nodes that make up the decision tree model.

- d. Save your **Decision Tree** block. To save, either click the Save button () or press **Ctrl+S**.
- e. Close the **Decision Tree Editor** view.

The **Decision Tree Editor** view window closes. A green execution status is displayed in the **Output** port of the **Decision Tree** block.

7. From the **Scoring** grouping in the Workflow palette, click and drag a **Score** block onto the Workflow canvas, below the **Score** block and the *basketball_shots.csv* dataset block.

A **Score** block and empty dataset block are created on the Workflow canvas. Note that the **Score** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires a model and a dataset input.

8. Click on the *basketball_shots.csv* dataset block's **Output** port and drag a connection towards the **Input** port of the **Score** block.
9. Click on the **Decision Tree** blocks **Output** port and drag a connection towards the **Input** port of the **Score** block.

The execution status of the **Score** block turns green and its dataset output is populated with the original dataset and the scored variables.

10. Right-click on the **Score** block's output dataset block and click **Rename**. Type *Shot Predictions* and click **OK**.

You now have a manually configured decision tree model that predicts scoring accuracy of a basketball shot.

Hierarchical Clustering block

Provides an interface to create a hierarchical cluster model.

A hierarchical cluster model is created and edited using **Hierarchical Clustering** view. Required information to build the model is set using the hierarchical clustering **Preferences** dialog box. The hierarchical clustering **Preferences** dialog box is displayed the first time you open a new **Hierarchical Clustering** view.

To open the **Hierarchical Clustering** view, double-click the **Hierarchical Clustering** block.

Hierarchical clustering preferences

Enables you to specify the variables and set options for creating the cluster model.

Variable Selection

Enables you to specify the variables to be used in the clustering model.

The two boxes in the **Variable Selection** panel show **Unselected Variables** and **Selected Variables** that will be used in the clustering model. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Configuration

Enables you to specify settings for the hierarchical model created using variables selected in the **Variable Selection** panel.

Standardize

Standardises the variables to normal form, and uses these values when calculating clusters.

Impute

Replaces any missing values with a calculated variable in the input variables specified in the **Variable Selection** panel.

Perform pre-clustering

Specifies that input observations are allocated to a smaller number of clusters to reduce the problem complexity. Observations are assigned to the clusters using k-means clustering.

Method

Specifies how cluster distances and cluster centres are calculated.

Average linkage

Specifies the group average method to create data clusters.

Centroid method

Specifies the centroid method to create data clusters.

Complete linkage

Specifies the complete-linkage method to create data clusters.

Density linkage

Specifies the density-linkage method to create data clusters.

Flexible-beta method

Specifies the flexible-beta method to create data clusters, as described in Lance, G.N., and Williams, W.T., 1967. A general theory of classificatory sorting strategies, 1. Hierarchical systems in *Computer Journal*, Volume 9, pp 373–380.

McQuitty similarity analysis

Specifies that McQuitty-similarity analysis is used to create, as described in McQuitty, L.L., 1966. Similarity Analysis by Reciprocal Pairs for Discrete and Continuous Data, in *Educational and Psychological Measurement*, Volume 26, pp 825–831.

Median Method

Specifies the median method of centroids to create data clusters.

Single linkage

Specifies a single linkage or nearest neighbour calculation is used to create data clusters.

Ward minimum variance model

Specifies that the Ward minimum variance model is used to create data clusters, as described in Ward, H.J., 1963. Hierarchical Grouping to Optimize an Objective Function, in *Journal of the American Statistical Association*, Volume 58, pp. 236–244

Suppress squaring distances

Suppresses the squaring of distances in distances calculations. This option can be specified for the *Average linkage*, *Centroid*, *Median*, and *Ward minimum variance* models.

Mode

Specifies the minimum number of observations required for a cluster to be distinguished as a *modal cluster*. A modal cluster is a region of high density separated from another region of high density by a region of low density.

Beta

Specifies the beta value required to implement the flexible-beta method.

Dimensionality

Specifies the number of dimensions to use when calculating density estimates in the *density linkage* model.

Frequency variable

Specifies a variable selected in the **Value Selection** panel that defines the frequency of each observation.

K Nearest Neighbours

Specifies the number of nearest neighbours to each observation in cluster calculations when using the *Density linkage* method.

Radius of Sphere for uniform-kernel

Specifies the radius of the a sphere around each observation used to compute the nearest neighbours in cluster calculations when using the *Density linkage* method.

Hierarchical Clustering view

Displays a graphical representation of the cluster model and information about specific clusters in the model.

The **Hierarchical Clustering** view displays information using the following tabs.

Clustering

Displays a dendrogram graph, which is a tree displaying the calculated clusters of observations and the distances between clusters. The graphs show the Cubic Clustering Criteria (CCC), Pseudo F and Pseudo T-Square statistics for the generated clusters. On all graphs, the vertical dotted line shows the number of clusters currently selected.

The view contains two slider controls:

- **Dendrogram details** enables you to adjust the level of detail in the tree.
- **Dendrogram Height Variable** enables you to choose the height variable used.
- **Number of clusters** enables you to select the number of clusters in the final model. The number selected determines the number of clusters listed in both the **Univariate View** and **Distribution Comparison**.

Univariate View

Displays the summary statistics for the input dataset and the same summary statistics for each cluster in the model. These statistics can be based on either the data values in the dataset (the **Input Space**) or a standardised normal distribution of the variable values (the **Standardised Space**).

The view displays a frequency distribution for a specified variable in the overall or cluster lists. The frequency distribution of the specified variable can be displayed as a histogram, line chart or pie chart. In all cases, the charts display the frequency of values as the percentage of the total number of observations for the variable.

The chart can be edited and saved. Click **Edit chart**  to open a Chart Editor from where the chart can be saved to the clipboard.

Distribution Comparison

Enables you to select a variable and view the frequency for a specified variable in one or more of the specified clusters. The displayed frequency can be based on either the data values in the dataset (the **Input Space**) or a standardised normal distribution of the variable values (the **Standardised Space**).

The frequency distribution of the specified variable and clusters can be displayed as a histogram, line chart or pie chart. In all cases, the charts display the frequency of values as the percentage of the total number of observations for the variable.

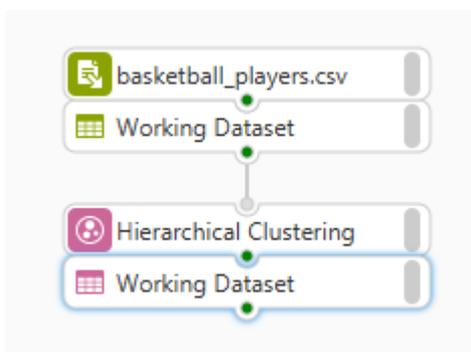
The chart can be edited and saved. Click **Edit chart**  to open a Chart Editor from where the chart can be saved to the clipboard.

Hierarchical Clustering block basic example

This example demonstrates how the **Hierarchical Clustering** block can be used to cluster observations in a dataset.

In this example, a dataset containing information about basketball players, named `basketball_players.csv` is imported into a blank Workflow. The dataset contains attributes of various basketball players. The **Hierarchical Clustering** block uses this dataset to cluster observations together.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Basketball players dataset

This example uses a dataset about basketball players, named `basketball_players.csv`. Each observation in the dataset describes a basketball player, with variables such as *height*, *weight*, *firstName* etc.

Using the Hierarchical Clustering block to create a Hierarchical Clustering model

Using the **Hierarchical Clustering** block to cluster observations.

1. Import the `basketball_players.csv` dataset into a blank Workflow canvas. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **Hierarchical Clustering** block onto the Workflow canvas, beneath the `basketball_players.csv` dataset.
3. Click on the `basketball_players.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Hierarchical Clustering** block.

Alternatively, you can dock the **Hierarchical Clustering** block onto the dataset block directly.

4. Double click on the **Hierarchical Clustering** block.

A **Hierarchical Clustering** view opens, along with a hierarchical clustering **Preferences** dialog box.

5. From the hierarchical clustering **Preferences** dialog box:
 - a. From the **Unselected Independent Variables** box, double click on the variables *height*, *weight*, and *goals_scored* to move them to the **Selected Independent Variables** box.
 - b. Click **OK**. The hierarchical clustering **Preferences** dialog box closes. Drag the **Number of clusters** slider to **3**.

Several graphs appears showing various details of the clustering.

- c. Save your **Hierarchical Clustering** block. To save, either click the Save button () or press **Ctrl +S**.
- d. Close the **Hierarchical Clustering** view.

The **Hierarchical Clustering** view closes. A green execution status is displayed in the **Output** port of the **Hierarchical Clustering** block and the resulting dataset, **Working Dataset** contains the application of the clustering model to `basketball_players.csv`.

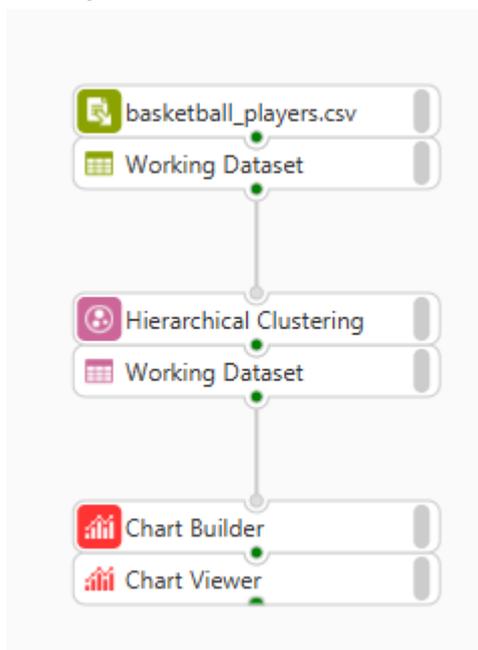
You now have a hierarchical clustering model that clusters similar observations together.

Hierarchical Clustering block example: analysing clustered observations

This example demonstrates how the **Hierarchical Clustering** block can be used as part of a cluster analysis.

In this example, a dataset containing information about basketball players, named `basketball_players.csv` is imported into a blank Workflow. The dataset contains attributes of various basketball players. The **Hierarchical Clustering** block uses this dataset to cluster observations together. The **Chart Builder** block is used to plot attributes of the dataset by its cluster, creating a visual representation for how the data has been clustered together. An output chart, **Chart Viewer** is generated.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Basketball players dataset

This example uses a dataset about basketball players, named `basketball_players.csv`. Each observation in the dataset describes a basketball player, with variables such as *height*, *weight*, *firstName* etc.

Using the Hierarchical Clustering block to plot clusters

Using the **Hierarchical Clustering** block and **Chart Builder** block to plot clusters as part of a cluster analysis.

1. Import the `basketball_players.csv` dataset into a blank Workflow canvas. Importing a CSV file is described in [Text File Import block](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **Hierarchical Clustering** block onto the Workflow canvas, beneath the `basketball_players.csv` dataset.
3. Click on the `basketball_players.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Hierarchical Clustering** block.

Alternatively, you can dock the **Hierarchical Clustering** block onto the dataset block directly.

4. Double click on the **Hierarchical Clustering** block.

A **Hierarchical Clustering** view opens, along with a hierarchical clustering **Preferences** dialog box.

5. From the hierarchical clustering **Preferences** dialog box:
 - a. From the **Unselected Independent Variables** box, double click on the variables *height*, *weight*, *goals_scored* and *shot_accuracy* to move them to the **Selected Independent Variables** box.
 - b. Click on the **Configuration** tab.
 - c. Clear the **Perform pre-clustering(recommended for large datasets)** checkbox.
 - d. Click **OK**.

The hierarchical clustering **Preferences** dialog box closes.

6. In the **Hierarchical Clustering** view:
 - a. Drag the **Number of clusters** slider to **5**.

Several graphs appears showing various details of the clustering.
 - b. Save your **Hierarchical Clustering** block. To save, either click the Save button () or press **Ctrl+S**.
 - c. Close the **Hierarchical Clustering** view.

The **Hierarchical Clustering** view closes. A green execution status is displayed in the **Output** port of the **Hierarchical Clustering** block and the resulting dataset, **Working Dataset** contains the application of the clustering model to `basketball_players.csv`.

7. Expand the **Export** group in the Workflow palette, then click and drag a **Chart Builder** block onto the Workflow canvas, beneath the **Hierarchical Clustering** block.
8. Click on the **Hierarchical Clustering** block **Working Dataset Output** port and drag a connection towards the **Input** port of the **Chart Builder** block.

Alternatively, you can dock the **Chart Builder** block onto the dataset block directly.

9. Double click on the **Chart Builder** block.

A **Chart Builder** view opens.

10. Configure the **Chart Builder** block:
 - a. Under the **Plots** pane drop-down box, select **Scatter**.

A series of configuration boxes appear for customising the scatter plot.
 - b. In the **X** drop down box, select **height**.
 - c. In the **Y** drop down box, select **goals_scored**.
 - d. In the **Options** pane, under the **Scatter** section, click on the **Group** drop down box and select **CLUSTER**.
 - e. Save your **Chart Builder** block. To save, either click the Save button () or press **Ctrl+S**. Then close the **Chart Builder** view window.

Saving generates a preview of the chart, closing the **Chart Builder** view then takes you back to the Workflow canvas. A green execution status is displayed in the **Output** port of the **Chart Builder** block and the resulting chart, **Chart Viewer** contains the full view of the chart.

11. Double click on the **Chart Viewer** to view the generated chart

The results are displayed in the **Chart Viewer**, showing how the different data points are assigned to each cluster.

You now have a hierarchical clustering model that clusters similar observations together.

K-Means Clustering block

Enables you to cluster data using *k*-means clustering.

Clustering, or cluster analysis, is an exploratory data analysis technique that divides data into groups. The groups contain items that are more similar to each other than they are to items in other groups. *k*-means clustering is a method for creating such groups.

To open the **Configure K-Means Clustering** dialog box, double-click the **K-Means Clustering** block.

The **K-Means Clustering** block generates a report displaying summary and graphical information about of the cluster model. To view this report, right-click the **K-Means Cluster Model** block and click **View Result**.

Configure K-Means Clustering

Enables you to specify the variables and set options for creating the K-Means clustering model. Only numeric data can be clustered.

Variable Selection

Specifies which variables are to be clustered. Only numeric data can be clustered, therefore only numeric variables are listed and can be selected.

The two boxes in the **Variable Selection** panel show **Unselected Variables** and **Selected Variables** that will be used in the clustering model. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Options

Standardize

Specifies that input variables are standardised; that is, the scales of variables are adjusted so that they are in a similar range.

Impute

Specifies that the value of missing values is imputed, based on the mean value for the variable.

Number of Clusters

Specifies the number of clusters into which the data is to be partitioned.

Max iterations

Specifies the maximum number of iterations to be performed by the clustering algorithm before terminating.

Convergence

Specifies a convergence threshold at which the clustering algorithm terminates.

Frequency Variable

Specifies a variable that contains the frequency associated with another variable.

Weight Variable

Specifies a variable that contains the prior weight associated with each variable.

K-Means Clustering Report view

Displays a summary and graphical representation of the cluster model and information about specific clusters in the model.

The **K-Means Clustering Report** view is accessed by double-clicking on the K-Means Cluster Model that is output by the K-Means Clustering block and displays information using the following tabs.

Clustering Results

Displays summary statistics for the dataset and each specified cluster. The **Summary** section shows the Cubic Clustering Criterion (CCC), Pseudo F and Pseudo T-Square for the input dataset. The **Clusters** section shows summary statistics for each cluster, and can be based on either the data values in the dataset (the **Input Space**) or a standardised normal distribution of the variable values (the **Standardised Space**).

Univariate View

Displays the summary statistics for the input dataset and the same summary statistics for each cluster in the model. These statistics can be based on either the data values in the dataset (the **Input Space**) or a standardised normal distribution of the variable values (the **Standardised Space**).

The view displays a frequency distribution for a specified variable in the overall or cluster lists. The frequency distribution of the specified variable can be displayed as a histogram, line chart or pie chart. In all cases, the charts display the frequency of values as the percentage of the total number of observations for the variable.

The chart can be edited and saved. Click **Edit chart**  to open a Chart Editor from where the chart can be saved to the clipboard.

Distribution Comparison

Enables you to select a variable and view the frequency for a specified variable in one or more of the specified clusters. The displayed frequency can be based on either the data values in the dataset (the **Input Space**) or a standardised normal distribution of the variable values (the **Standardised Space**).

The frequency distribution of the specified variable and clusters can be displayed as a histogram, line chart or pie chart. In all cases, the charts display the frequency of values as the percentage of the total number of observations for the variable.

The chart can be edited and saved. Click **Edit chart**  to open a Chart Editor from where the chart can be saved to the clipboard.

Scoring Code

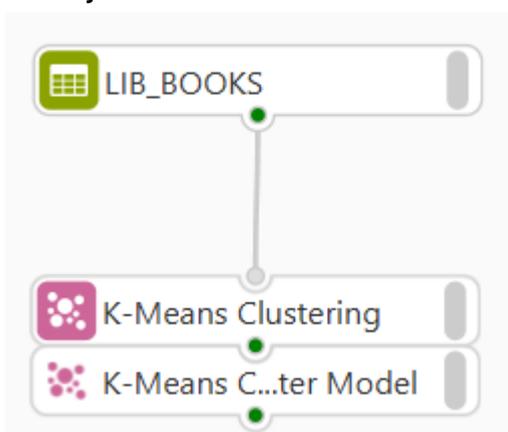
Displays the K-means model code. The code is available as SAS language code to be used in a **SAS Language** block.

K-Means Clustering block basic example

This example demonstrates how the **K-Means Clustering** block can be used to cluster observations in a dataset.

In this example, a dataset containing information about books, named `lib_books.csv`, is imported into a blank Workflow. The **K-Means Clustering** block uses this dataset to cluster observations together based on Euclidean distances of the numeric variables.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

lib_books.csv dataset

This example uses a dataset of books. Each observation in the dataset describes a book, with variables such as *Title*, *ISBN*, *Author* etc.

Using the K-Means Clustering block to create a K-means Clustering model

Using the **K-Means Clustering** block to cluster observations.

1. Import the `lib_books.csv` dataset into a blank Workflow canvas. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **K-Means Clustering** block onto the Workflow canvas, beneath the `lib_books.csv` dataset.
3. Click on the `lib_books.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **K-Means Clustering** block.

Alternatively, you can dock the **K-Means Clustering** block onto the dataset block directly.

4. Double click on the **K-Means Clustering** block.

A **Configure K-Means Clustering** dialog box opens.

5. From the **Configure K-Means Clustering** dialog box:
 - a. From the **Unselected Independent Variables** box, double click on the variables *DatePurchased*, *NumberInStock*, *Price* and *LastAccessed* to move them to the **Selected Independent Variables** box.
 - b. Click **OK**.

The **Configure K-Means Clustering** dialog box closes. A green execution status is displayed in the **Output** port of the **K-Means Clustering** block and the model results, **K-Means Clustering Model**.

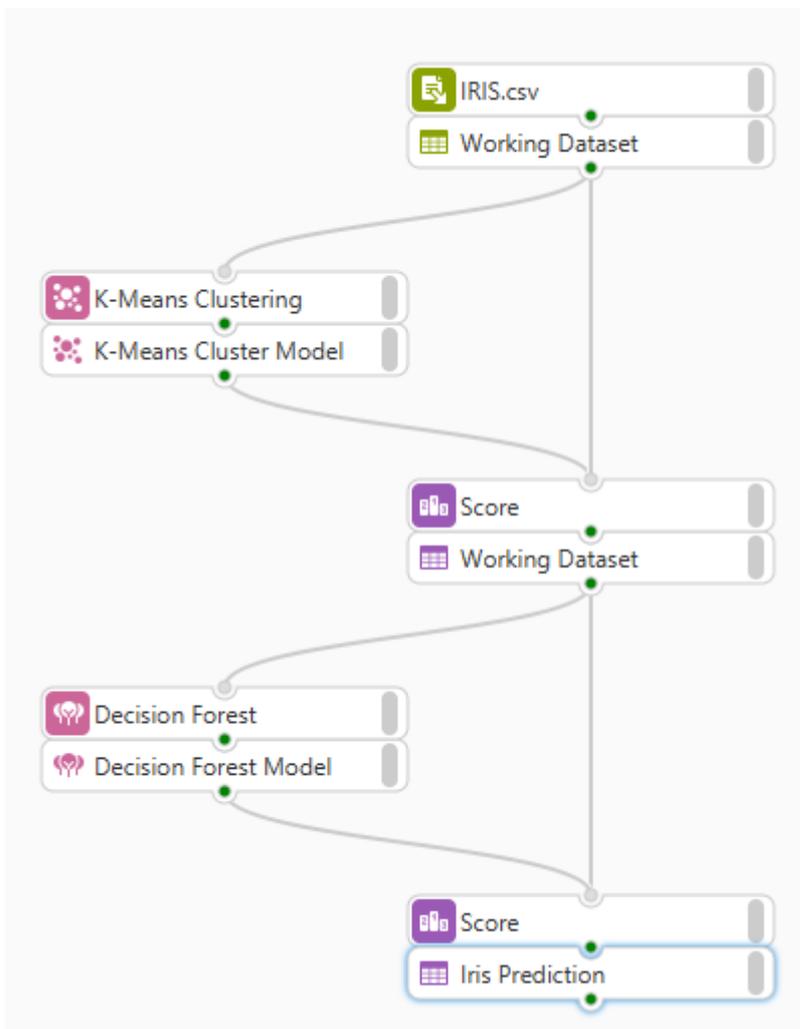
You now have a K-Means clustering model that clusters similar observations together.

K-Means Clustering block example: clustering observations as an input variable for a predictive model.

This example demonstrates how the **K-Means Clustering** block can be used to cluster observations in a dataset and use the variable that represents cluster assignment as an independent variable to a predictive model

In this example, a dataset containing information about iris flowers, named `IRIS.csv` is imported into a blank Workflow. The **K-Means Clustering** block uses this dataset to cluster observations together based on the length and width of the petals. The model is then applied to the dataset, using a **Score** block. A **Decision Forest** block is then used to generate a model that predicts the `species` variable using the generated variable that represents cluster assignment, `cluster`. The decision forest model is then applied to the clustered dataset, using a second **Score** block. This **Score** block generates an output dataset, `Iris Prediction`. This contains the predictions for the `species` variable with 96% accuracy.

Workflow Layout



Datasets used

The dataset used in this example is the publicly available iris flower dataset (Ronald A. Fisher, "The use of multiple measurements in taxonomic problems." *Annals of Eugenics*, vol.7 no.2, (1936); pp. 179--188.)

IRIS dataset

Each observation in the dataset describes an iris flower, with variables such as *species*, *sepal_length*, *petal_width* etc.

Using the K-Means Clustering block to assign clusters for model input

Using the **K-Means Clustering** block to cluster observations.

1. Import the `IRIS.csv` dataset into a blank Workflow canvas. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **K-Means Clustering** block onto the Workflow canvas, beneath the `IRIS.csv` dataset.
3. Click on the `IRIS.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **K-Means Clustering** block.

Alternatively, you can dock the **K-Means Clustering** block onto the dataset block directly.

4. Double click on the **K-Means Clustering** block.

A **Configure K-Means Clustering** dialog box opens.

5. From the **Configure K-Means Clustering** dialog box:
 - a. From the **Unselected Independent Variables** box, double click on the variables *petal_length* and *petal_width* to move them to the **Selected Independent Variables** box.
 - b. Click on the **Options** tab. Set the **Number of clusters** to 3.
 - c. Click **OK**.

The **Configure K-Means Clustering** window closes. A green execution status is displayed in the **Output** port of the **K-Means Clustering** block and the model results, **K-Means Clustering Model**.

6. From the **Scoring** grouping in the Workflow palette, click and drag a **Score** block onto the Workflow canvas, below the new **K-Means Clustering Model** block and the `IRIS.csv` dataset block.

A **Score** block and empty dataset block are created on the Workflow canvas. Note that the **Score** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires a model and a dataset input.

7. Click on the `IRIS.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Score** block.
8. Click on the **K-Means Clustering Model Output** port and drag a connection towards the **Input** port of the **Score** block.

The execution status of the **Score** block turns green and its dataset output is populated with the original dataset and the scored variables.

9. Click and drag a **Decision Forest** block onto the Workflow canvas.
10. Click on the **Score** block **Working DatasetOutput** port and drag a connection towards the **Input** port of the **Decision Forest** block.

Alternatively, you can dock the **Decision Forest** block onto the dataset block directly.

11. Double-click on the **Decision Forest** block.

A **Configure Decision Forest** dialog box opens.

12. From the **Configure Decision Forest** dialog box **Variable selection** tab:

- a. In the **Dependent variable** drop down list, select **species**.
- b. In the **Dependent variable treatment** drop down list, select **Nominal**.
- c. From the **Unselected Independent Variables** box, double click on the *cluster* variable, to move it to the **Selected Independent Variables** box.

13. Click on the **Preferences** tab and change the following:

- a. **Number of trees** to 20.
- b. **Classifier combination** as **Mean Probability**.
- c. Click **OK**.

The **Configure Decision Forest** dialog box saves and closes. A green execution status is displayed in the **Output** port of the **Decision Forest** block and the model results, *Decision Forest Model*.

14. From the **Scoring** grouping in the Workflow palette, click and drag a **Score** block onto the Workflow canvas, below the **Decision Forest** block.

A **Score** block and empty dataset block are created on the Workflow canvas. Note that the **Score** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires a model and a dataset input.

15. Click on the **Score** block **Working Dataset Output** port and drag a connection towards the **Input** port of the **Score** block.
16. Click on the **Decision Forest Model Output** port and drag a connection towards the **Input** port of the **Score** block.

The execution status of the **Score** block turns green and its dataset output is populated with the original dataset and the scored variables.

17. Right-click on the **Score** block's output dataset block, **Working Dataset** and click **Rename**. Type *Iris Prediction* and click **OK**.

You now have a predictive model that utilises input from a K-Means clustering model.

Linear Regression block

Enables you to create a linear regression model.

Linear regression attempts to model the linear relationship between multiple independent (regressor) variables and a dependent (response) variable. When you have found the linear relationship, you can use this to estimate the value of other response variables.

The **Linear Regression** block generates a report, diagnostic information, and scoring code. The report contains details of the variance in the dataset and parameter estimates. The diagnostic charts can be used to identify outlier points in the dataset. The generated scoring code can be copied into a SAS language program for testing or production use.

The model details are specified in the Variables included in the working dataset, using the **Configure Linear Regression** dialog box. To open the **Configure Linear Regression** dialog box, double-click the **Linear Regression** block.

Configure Linear Regression

Enables you to specify the variables used to create the linear regression model and also the parameters that define the relationship between the dependent and linear regressors in the model.

Variable Selection

Enables you to specify the variables used to create the linear regression model. Only numeric variables can be specified.

Dependent Variable

Specifies the dependent variable.

Unselected Regressors and Selected Regressors

Allows you to specify regressors for the linear regression model. The two boxes show **Unselected Regressors** and **Selected Regressors**. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Frequency Variable

Specifies a variable that defines the relative frequency of each observation.

Weight Variable

Specifies a variable that defines the prior weight associated with each observation.

Effect Selection

Enables you to specify parameters for the linear regression model.

Effect Selection panel

Enables you to specify parameters for the linear regression model.

Method

Specifies the method used to build the linear regression. The method can be:

Forward

Specifies the use of forward model selection.

The model will initially contain the intercept (if specified in the **Properties** panel) and the selected regressor (independent) variables added one at a time at each iteration of the model. The most significant variable (the effect variable with the smallest probability value and below the **Entry Significance Level**) is added at each iteration, and iterations continue until no specified effect variable is below the **Entry Significance Level**.

Backward

Specifies the use of backward model selection.

All selected regressor (independent) variables are included in the model, the variables are tested for significance, and the least significant dropped at each iteration of the model. The model is recalculated and the least-significant variable is dropped again.

A variable will not be dropped, however, if it is below the value specified for this method in **Exit Significance Level**. When all variables are below this level, the model stops.

Stepwise

Specifies that the model uses a combination of forward model and backward model selection.

The model initially contains an intercept, if specified, and one or more forward steps are taken to add regressors to the model. Backward and forward steps are used to remove or add regressors from or to the model until no further improvement can be made.

Maxr

At each iteration of the model selection process, this option chooses the variable that gives the largest increase in the R^2 statistic (coefficient of determination).

Minr

At each iteration of the model selection process, this option chooses the variable that gives the smallest increase in the R^2 statistic (coefficient of determination).

Rsquare

Specifies that the linear regression should be defined using the coefficient of determination (R^2). This finds a combination of regressors for the model which has highest R^2 .

Adjrsq

Specifies that the linear regression should be defined using the adjusted coefficient of determination (adjusted R^2). This finds a combination of regressors for the model which has highest Adjusted R^2 .

CP

Specifies that the linear regression should be defined using Mallows' C_p .

Frequency Variable

Specifies a variable that defines the relative frequency of each observation.

Weight Variable

Specifies a variable that defines the prior weight associated with each observation.

Start

Specifies the number of regressor (independent) variables to be included in the initial model for forward and stepwise model selection.

Stop

Specifies the maximum number of regressor (independent) variables the model can contain.

Entry Significance Level

Specifies the value below which a variable is included in the *Forward* or *Stepwise* methods.

Exit Significance Level

Specifies the value above which a variable is removed in the *Backward* or *Stepwise* methods.

Properties panel

Enables you to specify properties for the linear regression model.

Singular Value

Specifies the tolerance value used to test for linear dependency among the effect (independent) variables.

Intercept

Specifies that an intercept should be used.

Linear Regression Report view

Displays a summary and graphical information to help evaluate the linear regression model, along with SAS language scoring code.

The **Linear Regression Report** view is accessed by double-clicking on the Linear Regression Model that is output by the Linear Regression block. The report view displays information across two tabs and code in another.

Linear Regression Report tab

Displays a summary of the linear regression applied to the input dataset.

The **Model Information** section shows the Selection Method and Dependent Variable chosen in the Linear Regression block configuration, as well as the number of observations read and used by the model.

The **Summary Statistics** section gives statistics summarising the model used, including Root MSE (Means Squared Error), Dependent mean, R^2 (coefficient of determination), Adjusted R^2 (adjusted for the number of predictors in the model), and Coeff var (the coefficient of variation).

Diagnostics Panel tab

Displays plots of statistics relating to the regression model. Plots displayed are: predicted against residual, predicted against R-Student, leverage against R-Student, quantile-quantile, prediction against actual value, observation against Cook's Distance, and a residual histogram. Click on a plot to view it enlarged in the right hand pane.

Scoring code tab

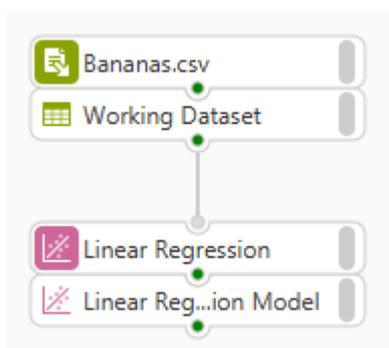
Displays the linear regression model code. The code is available as SAS language code to be used in a **SAS Language** block.

Linear Regression block basic example

This example demonstrates how the **Linear Regression** block can be used for a basic linear regression analysis.

In this example, a dataset named `Bananas.csv` is imported into a blank Workflow. The dataset contains the length and mass of bananas, with each observation representing an individual banana. The **Linear Regression** block is configured to produce a linear regression model from this dataset.

Workflow Layout



Dataset used

The file used in this example can be found in the samples distributed with WPS Analytics.

Bananas dataset

This example uses a dataset named `Bananas`. Each observation in the dataset describes a banana, with numeric variables `Length(cm)` and `Mass(g)` giving each banana's measured length in centimetres and mass in grams respectively.

Using the Linear Regression block to create a linear regression model

Using the **Linear Regression** block to carry out a basic linear regression analysis.

1. Import the `Bananas.csv` dataset into a blank Workflow. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **Linear Regression** block onto the Workflow canvas, beneath the `Bananas` dataset.
3. Click on the `Bananas` dataset block's **Output** port and drag a connection towards the **Input** port of the **Linear Regression** block.

Alternatively, you can dock the **Linear Regression** block onto the dataset block directly.

4. Double-click on the **Linear Regression** block.

A **Configure Linear Regression** dialog box opens.

5. From the **Variable Selection** tab:
 - a. From the **Dependent Variable** drop-down list, select **Mass(g)**.
 - b. From the **Unselected Regressors** list, double click **Length(cm)** to move it to the **Selected Regressors** list.
6. Click **OK**.

The **Configure Linear Regression** dialog box closes. A green execution status is displayed in the **Output** port of the **Linear Regression** block and the **Output** port of the **Linear Regression Model**.

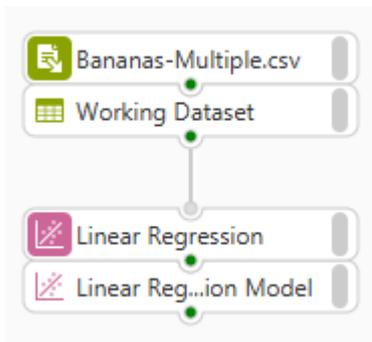
7. Double-click the **Linear Regression Model** to view the results.

Linear Regression block example: using multiple regressors

This example demonstrates how the **Linear Regression** block can be used for a linear regression analysis that includes multiple regressor variables.

In this example, a dataset named `Bananas-Multiple.csv` is imported into a blank Workflow. The dataset contains the length and mass of bananas from different regions of the world. The **Linear Regression** block is configured to produce a linear regression model from this dataset.

Workflow Layout



Dataset used

The file used in this example can be found in the samples distributed with WPS Analytics.

Bananas-Multiple dataset

This example uses a dataset named `Bananas.csv`. Each observation in the dataset describes a length of banana, with the numeric variable *Length(cm)* giving the length in centimetres, with the corresponding masses for that length from different regions given by *Mass(g)-In*, *Mass(g)-Ch*, *Mass(g)-Ph*, *Mass(g)-Col*, and *Mass(g)-Ec*.

Using the Linear Regression block to create a linear regression model with multiple regressors

Using the **Linear Regression** block to carry out a linear regression analysis with multiple regressor variables.

1. Import the `Bananas-Multiple.csv` dataset into a blank Workflow. Importing a CSV file is described in [Text File Import block](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **Linear Regression** block onto the Workflow canvas, beneath the `Bananas` dataset.
3. Click on the `Bananas` dataset block's **Output** port and drag a connection towards the **Input** port of the **Linear Regression** block.

Alternatively, you can dock the **Linear Regression** block onto the dataset block directly.

4. Double-click on the **Linear Regression** block.

A **Configure Linear Regression** dialog box opens.

5. From the **Variable Selection** tab:
 - a. From the **Dependent Variable** drop-down list, select **Length(cm)**.
 - b. From the **Unselected Regressors** list, click **Select all** () to move every mass variable to the **Selected Regressors** list.

6. From the **Model Selection** tab:
 - a. From the **Method** drop-down list, select **Forward**.
 - b. In **Entry Significance Level** enter 0.7.
7. Click **OK**.

The **Configure Linear Regression** dialog box closes. A green execution status is displayed in the **Output** port of the **Linear Regression** block and the **Output** port of the **Linear Regression Model**.

8. Double-click the **Logistic Regression Model** to view the results.

Logistic Regression block

Fits a logistic model between a set of effect variables and a binary dependent variable.

The **Logistic Regression** block generates a report and deployment code. The report contains details of the Model fit statistics, Maximum likelihood estimate analysis, odds ratio estimates, and predicted probability and observed responses. The generated deployment code is a `DATA` step that can be copied into a SAS language program for testing or production use.

The model details are specified in the Variables included in the working dataset are specified using the **Configure Logistic Regression** dialog box. To open the **Configure Logistic Regression** dialog box, double-click the **Logistic Regression** block.

Configure Logistic Regression

Enables you to specify the variables used to create the logistic regression model and also the parameters that define the relationship between the effect variables and binary dependent variable.

Variable Selection

Enables you to specify the variables used to create the logistic regression model.

Dependent variable

Specifies the binary dependent variable for the calculation.

Event

Specifies the target category for which the probability of occurrence is to be calculated.

Effect Variables

Allows you to specify effect variables. The two boxes show **Unselected Effect Variables** and **Selected Effect Variables**. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Frequency Variable

Specifies a variable that defines the frequency of each observation.

Weight Variable

Specifies a variable that defines the prior weight associated with each observation.

Model Selection

Enables you to specify parameters for the logistic regression model.

Method

Specifies the model effect variable selection method. Choose from:

- **None:** Specifies that the fitted model includes every selected effect (independent) variable.
- **Forward:** Specifies the use of forward model selection.

The model initially contains intercept, if specified, and the selected effect (independent) variables added one at a time at each iteration of the model. The most significant variable (the effect variable with the smallest probability value and below the **Entry Significance Level**) is added at each iteration, and iterations continue until no specified effect variable is below the **Entry Significance Level**.

- **Backward:** Specifies the use of backward model selection.

All selected effect (independent) variables are included in the model, the variables are tested for significance, and the least significant dropped at each iteration of the model. The model is recalculated and the least-significant variable is dropped again. A variable will not be dropped, however, if it is below the value specified for this method in **Stay Significance Level**. When all variables are below this level, the model stops.

- **Stepwise:** Specifies that the model uses a combination of forward model and backward model selection.

The model initially contains an intercept, if specified, and one or more forward steps are taken to add effect variables to the model. Backward and forward steps are used to remove effect variables from and add effect variables to the model until no further improvement can be made to the model.

- **Score:** Specifies that the model returned has the best likelihood score statistic.

The model is created using a branch-and-bound method. Initially a model is created using the specified effect (independent) variable with the best individual likelihood score statistic. A second model is created using the two specified effect variables with the best combined likelihood score statistic. The model with the best likelihood score statistic is considered the best current solution and becomes the bound for the next iteration.

At each subsequent iteration, an effect variable is added to the model where the combination of all effect variables has the best likelihood score statistic. The model is compared to the best current solution and if the new model likelihood score statistic is better than the current solution, the new model becomes the bound for the next iteration.

When the best likelihood score statistic cannot be improved, the model is returned.

Fast

Available if **Method** is selected as **Backward**. Implements a quicker approximation for the model.

Link Function

Specifies how effect (independent) variables are linked to the dependent variable. Choose from:

- **CLOGLOG.** Specifies the *cloglog* (complementary log-log) function is used:

$$f(p) = \log(-\log(1-p))$$

Where p is the probability of the **Event** occurring.

- **LOGIT.** Specifies the *logit* function (the inverse of the logistic function) is used:

$$f(p) = \log\left(\frac{p}{1-p}\right)$$

Where p is the probability of the **Event** occurring.

- **PROBIT.** Specifies the *probit* function is used:

$$f(p) = \Phi^{-1}(p)$$

Where Φ^{-1} is the inverse cumulative distribution function of the standard normal distribution where the mean = 0 and variance = 1.

Entry Significance Level

Specifies the value below which a variable is included in the **Forward** or **Stepwise** methods.

Stay Significance Level

Specifies the value above which a variable is removed in the **Backward** or **Stepwise** methods.

Properties

Singular Tolerance

Specifies the tolerance value used to test for linear dependency among the effect (independent) variables.

Intercept

If selected, includes an intercept in the model. If cleared, the model will not include an intercept.

Logistic Model Report view

Displays summary and graphical information to help evaluate the Logistic Regression model.

The **Logistic Model Report** view is accessed by double-clicking on the Logistic Regression Model that is output by the Logistic Regression block. The report view displays information in one tab and code in another.

Logistic Regression Results tab

Displays a summary of the logistic regression applied to the input dataset.

The **Model Information** section confirms the dataset and response variable used along with the number of response levels. Also shown are the model and optimisation technique used and finally the number of observations read and used.

The **Model Fit Statistics** section shows the output of three tests of model fit for relative comparisons: AIC (Akaike Information Criterion), SC (Schwarz Criterion) and -2 Log L.

The **Testing Global Null Hypothesis: BETA=0** section shows the output of three tests: Likelihood ratio, score and Wald.

The **Analysis of Maximum Likelihood Estimates** section shows an Analysis of Maximum Likelihood Estimates for each observation in the dataset.

The **Odds Ratio Estimates** section shows Odds Ratio Estimates for each observation in the dataset.

The **Association of Predicted Probabilities and Observed Responses** section shows a comparison of observations in the dataset and the predicted values from the logistic regression, showing information on the concordant, discordant and tied pairs.

Scoring code tab

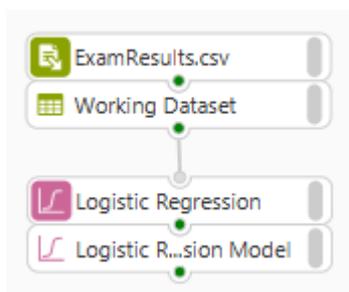
Displays the logistic regression model code. The code is available as SAS language code to be used in a **SAS Language** block.

Logistic Regression block basic example

This example demonstrates how the **Logistic Regression** block can be used for a basic logistic regression analysis.

In this example, a dataset of exam results named `ExamResults.csv` is imported into a blank Workflow. The dataset contains details of the hours students spent studying for an exam and whether they passed or failed the exam. The **Logistic Regression** block is configured to produce a logistic regression model from this dataset.

Workflow Layout



Dataset used

The file used in this example can be found in the samples distributed with WPS Analytics.

ExamResults dataset

This example uses a dataset of exam results, named `ExamResults.csv`. Each observation in the dataset describes a student, with a numeric variable *HoursStudy* giving their hours of study for an exam, and a binary variable *Pass?* indicating whether they passed the exam or not.

Using the Logistic Regression block to create a logistic regression model

Using the **Logistic Regression** block to carry out a simple logistic regression analysis.

1. Import the `ExamResults.csv` dataset into a blank Workflow. Importing a CSV file is described in [Text File Import block](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **Logistic Regression** block onto the Workflow canvas, beneath the `ExamResults.csv` dataset.
3. Click on the `ExamResults` dataset block's **Output** port and drag a connection towards the **Input** port of the **Logistic Regression** block.

Alternatively, you can dock the **Logistic Regression** block onto the dataset block directly.

4. Double-click on the **Logistic Regression** block.

A **Configure Linear Regression** dialog box opens.

5. From the **Variable Selection** tab:
 - a. From the **Dependent Variable** drop-down list, select **Pass?**.
 - b. From the **Event** drop-down list, select **1**.
 - c. From the **Unselected Effect Variables** list, double-click **HoursStudy**.

HoursStudy is moved to the **Selected Effect Variables** list.

- d. Click **OK**.

The **Configure Logistic Regression** window closes. A green execution status is displayed in the **Output** ports of the **Logistic Regression** block and of its output **Logistic Regression Model**.

6. Double-click the **Logistic Regression Model** to view the results.

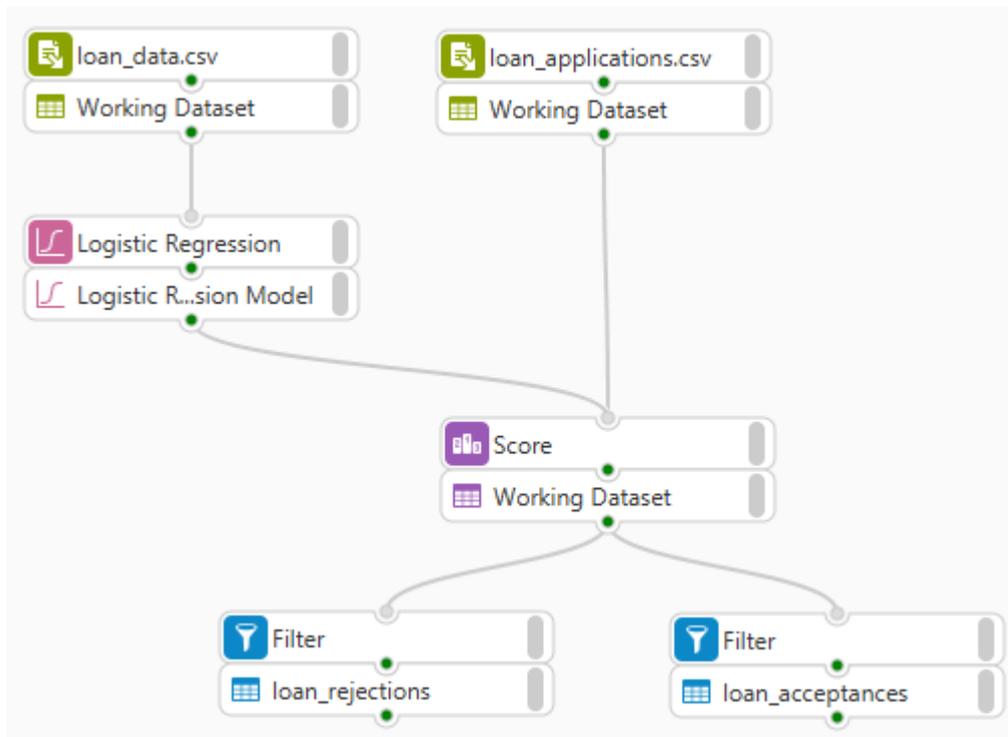
You now have a logistic regression model that predicts the likelihood of a student passing an exam based on their hours of study.

Logistic Regression block example: a more complex logistic regression and application of the model to a new dataset

This example demonstrates how the **Logistic Regression** block can be used for a logistic regression analysis of loan data with multiple effect variables. The model generated is then applied to another dataset of loan applications, and the results filtered into loan rejections and loan acceptances.

In this example, a dataset of exam results named `loan_data.csv` is imported into a blank Workflow. The dataset contains details of past loan applications and whether or not each loan defaulted. A **Logistic Regression** block is used to build a logistic regression model, looking at a selection of effect variables and how they relate to whether or not each loan defaulted. The **Logistic Regression** block is configured to output its model and, using a **Score** block, this is applied against a dataset of new loan applications called `loan_applications.csv`. The **Score** block then outputs the `loan_applications.csv` dataset appended with the probability of defaulting for each loan application, which is then filtered, using the probabilities, into two datasets of loan rejections and loan acceptances.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Loan dataset

This example uses a dataset about loans, named `loan_data.csv`. Each observation in the dataset describes a loan and the person who took the loan out, with variables such as *Income*, *Loan_Period*, *Other_Debt* etc.

Loan applications dataset

This example uses a dataset about loan applications, named `loan_applications.csv`. Each observation in the dataset describes an application for a loan. This dataset contains the same variables as `LOAN_DATA`, but without a *Default* variable.

Using the Logistic Regression block to create a logistic regression model and then apply that model to another dataset

Using the **Logistic Regression** block to carry out a logistic regression analysis and then applying the trained model to another dataset.

1. Import the `loan_data.csv` and `loan_applications.csv` datasets into a blank Workflow. Importing a CSV file is described in [Text File Import block](#) (page 184).

2. Expand the **Model Training** group in the Workflow palette, then click and drag a **Logistic Regression** block onto the Workflow canvas, beneath the `LOAN_DATA` dataset.
3. Click on the `LOAN_DATA` dataset block's **Output** port and drag a connection towards the **Input** port of the **Logistic Regression** block.

Alternatively, you can dock the **Logistic Regression** block onto the dataset block directly.

4. Double-click on the **Logistic Regression** block.

A **Configure Logistic Regression** dialog box opens.

5. From the **Variable Selection** tab:
 - a. From the **Dependent Variable** drop-down list, select **Default**.
 - b. From the **Event** drop-down list, select **1**.
 - c. From the **Unselected Effect Variables** list, double-click **Employment_Type**, **Income**, **Other_Debt**, **Housing_Situation**, and **Has_CCJ**.

All chosen variables are moved to the **Selected Effect Variables** list.

6. From the **Model Selection** tab:
 - a. From **Method**, select **Forward**.
 - b. From **Link function**, select **LOGIT**.
 - c. In **Entry Significance Level**, type `0.07`.

7. Click **OK**.

The **Configure Logistic Regression** dialog box closes. A green execution status is displayed in the **Output** port of the **Logistic Regression** block and the **Output** port of the **Logistic Regression Model**.

8. From the **Scoring** grouping in the Workflow palette, click and drag a **Score** block onto the Workflow canvas, below the new **Decision Forest Model** block and the `loan_applications` dataset block.

A **Score** block and empty dataset block are created on the Workflow canvas. Note that the **Score** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires a model and a dataset input.

9. Click on the `loan_applications` dataset block's **Output** port and drag a connection towards the **Input** port of the **Score** block.
10. Click on the **Logistic Regression Model Output** port and drag a connection towards the **Input** port of the **Score** block.

The execution status of the **Score** block turns green and its dataset output is populated with the original dataset and the scored variables.

11. From the **Data Preparation** grouping in the Workflow palette, click and drag two **Filter** blocks onto the Workflow canvas.
12. Click on the **Score** block **Output** port and drag a connection towards the **Input** port of each **Filter** block.

13. Configure the first **Filter** block:

- a. Double-click on a **Filter** block.
A **Configure Filter** dialog box opens.
- b. Click on the **Basic** tab.
- c. From the **Variable** drop-down box, select **P_1**.
- d. From the **Operator** drop-down box, select **<**.
- e. In the **Value** box, enter **0.5**.
- f. Click the **Save** button at the top left of the screen, then close the **Filter** configuration window.

The **Filter** configuration window closes. A green execution status is displayed in the **Output** port of the **Filter** block and the filtered dataset, **Working Dataset**.

- g. Right-click on the **Filter** block's output dataset block and click **Rename**. Type `loan_rejections` and click **OK**.

14. Now configure the second **Filter** block:

- a. Double-click on the other **Filter** block.
A **Filter** configuration window opens.
- b. Click on the **Basic** tab.
- c. For the **Variable** drop-down box, select **P_1**.
- d. For the **Operator** drop-down box, select **>=**.
- e. In the **Value** box, enter **0.5**.
- f. Save the **Filter** block. To save, either click the Save button () or press **Ctrl+S**.
- g. Close the **Filter Editor** view.

The **Filter** configuration window closes. A green execution status is displayed in the **Output** port of the **Filter** block and the filtered dataset, **Working Dataset**.

- h. Right-click on the **Filter** block's output dataset block and click **Rename**. Type `loan_acceptances` and click **OK**.

You now have a logistic regression model that predicts if a loan applicant will default and an application of that model to another dataset, with results filtered into acceptances and rejections.

MLP block

Enables you to build a Multi-Layer Perceptron (MLP) neural network from an input dataset and use the trained network to analyse other datasets.

MLP neural networks are created and edited using the **Multilayer Perceptron** view and MLP **Preferences** dialog box. Double-clicking a **MLP** block will open the **Multilayer Perceptron** view; if the block is new with no basic information configured, the MLP **Preferences** dialog box will also open. Once basic information to build an MLP neural network has been configured, double-clicking the **MLP** block will just open the **Multilayer Perceptron** view. To open the MLP **Preferences** dialog box manually, click the preferences icon () at the top right of the **Multilayer Perceptron** view.

Multilayer perceptron neural networks are a class of non-linear machine learning algorithms that can be used for classification and regression.

Networks are usually conceptualised as layers of non-linear functions analogous to layers of neurons in biological neural networks connected by layers of weights analogous to synapses.

Training algorithms are used to find the weights that enable an MLP to approximate relationships between the effect variables and a response variable, making it possible to create a network that can predict unknown responses from known effects.

For more information about MLP neural networks, see *MLP procedure* in the *WPS Machine Learning* section in *WPS Reference for Language Elements*.

Multilayer perceptron preferences

Enables you to specify the preferences used to create a trained network.

Dataset Roles

Defines how the input datasets are used in the trained network.

Train

Specifies the dataset used to train the network.

Validate

Specifies a validation dataset. If specified, the result of training is the network that achieved the minimum error against this dataset.

Test

Specifies a dataset used to test the network at the end of training.

Variable Selection

Enables you to specify the dependent (target) variable and independent (input) variables in the input dataset to use the trained network.

Dependent variable

Specifies the dependent (target) variable for the trained network.

Independent variables

Allows you to specify effect (independent) variables. The two boxes show **Unselected Independent Variables** and **Selected Independent Variables**. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Optimiser

Specifies the optimisation algorithm to be used to train the network.

The following optimisers run using minibatches from the training dataset. A minibatch is a subset of observations, where all subsets are used in one iteration (epoch) when training the network:

- *ADADELTA* [↗](#) (page 294)
- *ADAM* [↗](#) (page 295)
- *NADAM* [↗](#) (page 296)
- *RMSPROP* [↗](#) (page 297)
- *SGD* [↗](#) (page 298)
- *SMORMS3* [↗](#) (page 298)

The following optimisers run using the whole training dataset in each iteration (epoch) when training the network:

- *RPROP* [↗](#) (page 297)

ADADELTA

Specifies an adaptive step size minibatch learning algorithm that reduces sensitivity to the choice of learning rate.

Epsilon

Specifies a value for the epsilon parameter.

Smoothing factor

Specifies a value for the smoothing factor of the network.

Minibatch type

Specifies how observations are selected for minibatches.

DYNAMIC

Specifies that observations are selected randomly without replacement.

STATIC

Specifies that observations are selected in the order they appear in the training dataset.

BOOTSTRAP

Specifies that observations are selected randomly with replacement.

Minibatch size

Specifies the number of observations in each minibatch.

Learning rates

Specifies a learning rate schedule as a series of minibatch number-learning rate pairs. To define learning rates, enter a **minibatch** number and the required **Learning rate** for that minibatch and click **Add**. Only one learning rate can be specified per minibatch number. If you specify a single learning rate, that learning rate is used for all iterations.

To change a learning rate, select the row in the **Learning rates** box, change the rate and click **Update**.

To remove a specified minibatch number-learning rate pair, select the required row in the **Learning rates** box and click **Delete**.

ADAM

An algorithm that attempts to maximize speed of convergence by using estimates of lower-order moments.

Epsilon

Specifies a value for the epsilon parameter.

Beta1

Specifies a value for the beta1 parameter.

Beta2

Specifies a value for the beta2 parameter.

Minibatch type

Specifies how observations are selected for minibatches.

DYNAMIC

Specifies that observations are selected randomly without replacement.

STATIC

Specifies that observations are selected in the order they appear in the training dataset.

BOOTSTRAP

Specifies that observations are selected randomly with replacement.

Minibatch size

Specifies the number of observations in each minibatch.

Learning rates

Specifies a learning rate schedule as a series of minibatch number-learning rate pairs. To define learning rates, enter a **minibatch** number and the required **Learning rate** for that minibatch and click **Add**. Only one learning rate can be specified per minibatch number. If you specify a single learning rate, that learning rate is used for all iterations.

To change a learning rate, select the row in the **Learning rates** box, change the rate and click **Update**.

To remove a specified minibatch number-learning rate pair, select the required row in the **Learning rates** box and click **Delete**.

NADAM

A version of the **ADAM** optimiser method that incorporates *Nesterov* momentum.

Epsilon

Specifies a value for the epsilon parameter.

Beta1

Specifies a value for the beta1 parameter.

Beta2

Specifies a value for the beta2 parameter.

Minibatch type

Specifies how observations are selected for minibatches.

DYNAMIC

Specifies that observations are selected randomly without replacement.

STATIC

Specifies that observations are selected in the order they appear in the training dataset.

BOOTSTRAP

Specifies that observations are selected randomly with replacement.

Minibatch size

Specifies the number of observations in each minibatch.

Learning rates

Specifies a learning rate schedule as a series of minibatch number-learning rate pairs. To define learning rates, enter a **minibatch** number and the required **Learning rate** for that minibatch and click **Add**. Only one learning rate can be specified per minibatch number. If you specify a single learning rate, that learning rate is used for all iterations.

To change a learning rate, select the row in the **Learning rates** box, change the rate and click **Update**.

To remove a specified minibatch number-learning rate pair, select the required row in the **Learning rates** box and click **Delete**.

RMSPROP

An adaptive step size minibatch learning algorithm.

Epsilon

Specifies a value for the epsilon parameter.

Smoothing factor

Specifies a value for the smoothing factor.

Minibatch type

Specifies how observations are selected for minibatches.

DYNAMIC

Specifies that observations are selected randomly without replacement.

STATIC

Specifies that observations are selected in the order they appear in the training dataset.

BOOTSTRAP

Specifies that observations are selected randomly with replacement.

Minibatch size

Specifies the number of observations in each minibatch.

Learning rates

Specifies a learning rate schedule as a series of minibatch number-learning rate pairs. To define learning rates, enter a **minibatch** number and the required **Learning rate** for that minibatch and click **Add**. Only one learning rate can be specified per minibatch number. If you specify a single learning rate, that learning rate is used for all iterations.

To change a learning rate, select the row in the **Learning rates** box, change the rate and click **Update**.

To remove a specified minibatch number-learning rate pair, select the required row in the **Learning rates** box and click **Delete**.

RPROP

An adaptive step size full batch learning algorithm.

Initial delta

Specifies the initial step size.

Min delta

Specifies the minimum step size.

Max delta

Specifies the maximum step size.

Step size increment (ETA+)

Specifies the factor by which the step size is increased.

Step size decrement (ETA-)

Specifies the factor by which the step size is reduced.

SGD

A fixed step size minibatch learning algorithm that supports *Classical* and *Nesterov* momentum.

Momentum

Specifies the type of momentum, either `NESTEROV` or `CLASSICAL`.

Value

Specifies the value of the selected momentum.

Minibatch type

Specifies how observations are selected for minibatches.

DYNAMIC

Specifies that observations are selected randomly without replacement.

STATIC

Specifies that observations are selected in the order they appear in the training dataset.

BOOTSTRAP

Specifies that observations are selected randomly with replacement.

Minibatch size

Specifies the number of observations in each minibatch.

Learning rates

Specifies a learning rate schedule as a series of minibatch number-learning rate pairs. To define learning rates, enter a **minibatch** number and the required **Learning rate** for that minibatch and click **Add**. Only one learning rate can be specified per minibatch number. If you specify a single learning rate, that learning rate is used for all iterations.

To change a learning rate, select the row in the **Learning rates** box, change the rate and click **Update**.

To remove a specified minibatch number-learning rate pair, select the required row in the **Learning rates** box and click **Delete**.

SMORMS3

An adaptive step size minibatch learning algorithm that automatically adjusts the amount of smoothing to improve the trade-off between rate of convergence and stability.

Epsilon

Specifies a value for the epsilon parameter.

Minibatch type

Specifies how observations are selected for minibatches.

DYNAMIC

Specifies that observations are selected randomly without replacement.

STATIC

Specifies that observations are selected in the order they appear in the training dataset.

BOOTSTRAP

Specifies that observations are selected randomly with replacement.

Minibatch size

Specifies the number of observations in each minibatch.

Learning rates

Specifies a learning rate schedule as a series of minibatch number-learning rate pairs. To define learning rates, enter a **minibatch** number and the required **Learning rate** for that minibatch and click **Add**. Only one learning rate can be specified per minibatch number. If you specify a single learning rate, that learning rate is used for all iterations.

To change a learning rate, select the row in the **Learning rates** box, change the rate and click **Update**.

To remove a specified minibatch number-learning rate pair, select the required row in the **Learning rates** box and click **Delete**.

Regularisers

Multiple regularisers can be specified, so that more than one type of regularisation can be applied simultaneously. To add a regulariser, click **Add Regulariser** and fill in the required details for the regulariser type.

Regulariser

Specifies the type of regularisation to be used during training.

LNNORM

Specifies that a norm-based regulariser be applied to all non-bias weights.

DROPOUT

Specifies that dropout regularisation should be used on all hidden layers and that neurons will be dropped out during training with the specified probability.

Probability

Specifies the probability of neurons being dropped out during training using the **DROPOUT** regulariser.

Strength

Specifies the strength of the **LNNORM** regulariser.

Power

Specifies the power of the **LNNORM** regulariser.

Stopping Criteria

Defines when model training stops.

Max training time (s)

Specifies a target maximum training time in seconds.

Number of epochs

Specifies that training should terminate if the likelihood cannot be computed for the specified number of successive epochs.

Max unimproved validation assessment

Specifies the maximum number of validation assessments to that can be run without termination where the validation error does not improve.

Validation interval

Specifies the interval between validation set assessments. If you specify the `RPROP` optimiser, the interval is the number of epochs. For all other optimisers, the interval is the number of minibatches.

Multilayer Perceptron view

Enables you to create a Multilayer Perceptron (MLP), export the resulting trained network code and view the tabular output of the trained network.

The **Multilayer Perceptron** view is used to create a trained network based on the settings specified in the MLP **Preferences** dialog box. The **Multilayer Perceptron** tab enables you to create a trained network and view the progress as the network is trained. The **Scoring Code** tab enables you to export the trained network SAS language code. The **Report** tab displays tabular output information from the **MLP** block.

Multilayer Perceptron Tab

Enables you define the layers of the trained network. The **Layers** panel contains the detail of the trained network, and the **Error** panel displays the current status of the network as it is created.

To create a trained network, click **Train**.

Configure Multilayer Perceptron

Layers

Displays the layers in the trained network. Every network requires an Input, an Output and at least one Hidden layer. To add a new hidden layer to the trained network, click **Add Layer** and complete the information for the new row in the list. To remove a Hidden layer, click **Delete Layer**.

You can specify the **Number of Neurons** for each hidden later and the **Activation Function** to specify how the activity y of a neuron is calculated from its post-synaptic potential x . The available Activation functions are:

ARCTAN	$y = \arctan(x)$
ELLIOTT	$y = \frac{x}{1 + x + 1 }$
LEAKYRECTIFIEDLINEAR	$y = x$ if $x > 0$ $y = 0.01x$ otherwise
LINEAR	$y = x$
LOGISTIC	$y = \frac{1}{1 + \exp(-x)}$
RECTIFIEDLINEAR	$y = x$ if $x > 0$ $y = 0$ otherwise
SOFTMAX	$y_i = \frac{\exp(x_i)}{\sum_{n=1}^N \exp(x_n)}$ where N is the number of network outputs.
SOFTPLUS	$y = \log 1 + e^x $
SOFTSIGN	$y = \frac{x}{1 + x }$
TANH	$y = \tanh(x)$

Error

Displays the performance of the **MLP** block as it runs and automatically updates the history of any training error and validation error at each iteration.

Scoring Code tab

Displays the trained network code. The code is available as SAS language code to be used in a **DATA** step in the **SAS Language** block.

Report tab

Displays the tabular output of the trained network.

Configuration

The configuration table records the specified configuration used during this execution of the **MLP** block.

Input Encoding

Shows the mapping between the independent variables and the input neuron in the MLP. Numeric variables are mapped to a single input node. Categorical variables are mapped to multiple neurons, and the range is displayed in this table.

Input Lengths

Shows the weights of each input neuron.

Input Mapping

Shows the mapping between independent variables and input neuron for each neuron in the input layer of the MLP. Numerical variables have a single row in the table showing the neuron to which the variable is mapped. Categorical variables have multiple rows in the table showing which neuron each category is mapped to.

Input Scaling

Shows the original range in values of numerical values before being scaled to a consistent range, typically -1 to 1.

Network Architecture

The network architecture table summarises the MLP generated during the execution of the **MLP** block.

Results

The results table displays the results of training.

Stopping reasons

The stopping reason table lists all the reasons why training stopped.

Training History

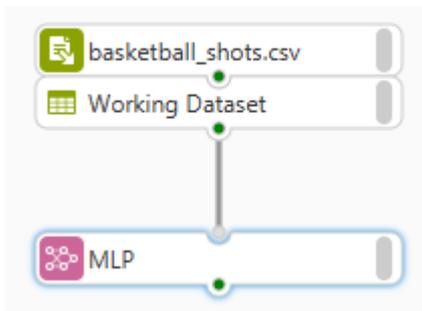
The training history table shows how training, validation and regularisation errors have changed during training.

MLP block basic example

This example demonstrates how the **MLP** block can be used to model a binary variable.

In this example, a dataset containing information about basketball shots, named `basketball_shots.csv` is imported into a blank Workflow. The dataset contains observations of each basketball shot and whether it scored through the *Scored* variable. The **MLP** block uses this dataset to estimate the likelihood for each basketball shot to score.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Basketball dataset

This example uses a dataset about basketball shots, named `basketball_shots.csv`. Each observation in the dataset describes an attempt to score and the person shooting, with variables such as *height*, *weight*, *distance_feet* etc.

Using the MLP block to create an MLP model.

Using the **MLP** block to model shooting accuracy in basketball

1. Import the `basketball_shots.csv` dataset onto a blank Workflow canvas. Importing a CSV file is described in [Text File Import block](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **MLP** block onto the Workflow canvas, beneath the `basketball_shots.csv` dataset.
3. Click on the `basketball_shots.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **MLP** block.

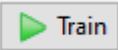
Alternatively, you can dock the **MLP** block onto the dataset block directly.

4. Double-click on the **MLP** block.

A **Multilayer Perceptron** view opens, along with the **MLP Preferences** dialog box.

5. From the MLP **Preferences** dialog box:
 - a. In the **Train** drop-down list, select **Working Dataset**.
 - b. Click on the **Variable Selection** tab.
 - c. In the **Dependent variable** drop-down list, select **Score**.
 - d. From the **Unselected Independent Variables** box, double-click on the variables *distance_feet*, *angle*, *height*, *weight* and *position* to move them to the **Selected Independent Variables** box.
 - e. Click on the **Optimiser** tab, from the **Optimiser** drop-down list, select **ADAM**.
 - f. Click on the **Stopping Criteria** tab, set the **Max training time (s)** to 10.
 - g. Click **OK**.

The MLP **Preferences** dialog box closes.

- h. Click on the **Train** button () to train the MLP model.

A graph appears on the right hand side showing the live training error during model training.

- i. Save your **MLP** block. To save, either click the Save button () or press **Ctrl+S**.
- j. Close the **Multilayer Perceptron** view.

The **Multilayer Perceptron** view closes. A green execution status is displayed in the **Output** port of the **MLP** block.

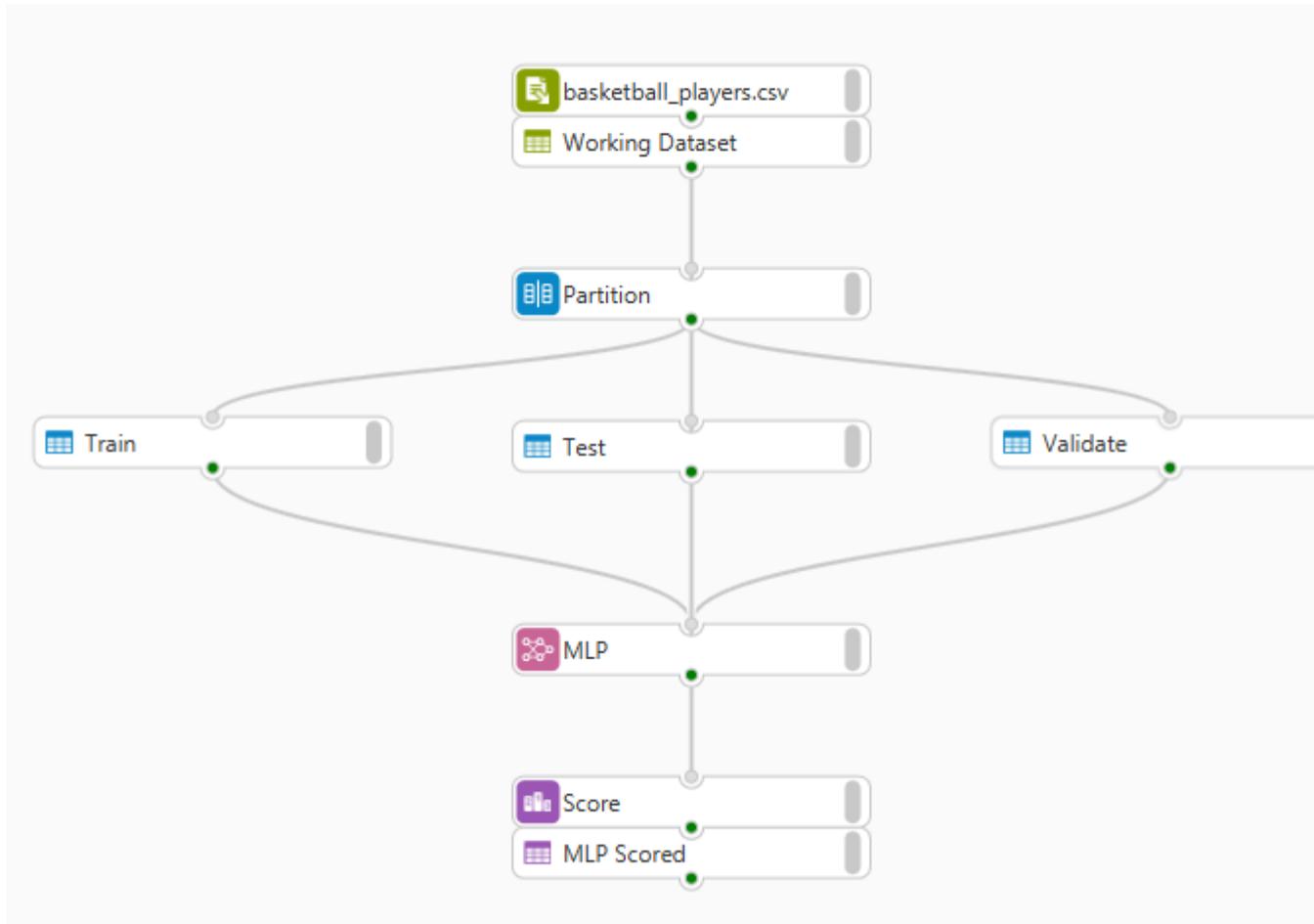
You now have an MLP model that predicts the probability of a basketball shot scoring.

MLP block example: Predicting a multinomial variable

This example demonstrates how the **MLP** block can be used to incorporate training, testing and validation data to model a non-binary class variable.

In this example, a dataset containing information about basketball players, named `basketball_players.csv` is imported into a blank Workflow. The dataset contains observations about each basketball player and their position through the *position* variable. The **Partition** block splits the dataset into a training, testing and validation dataset, named `Train`, `Test` and `Validation` respectively. The **MLP** block uses these datasets to estimate the position of the basketball player. The model is then applied to the original dataset, `basketball_players.csv`, using a **Score** block. The **Score** block generates an output dataset **MLP Scored**.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Basketball players dataset

This example uses a dataset about basketball players, named `basketball_players.csv`. Each observation in the dataset describes a basketball player, with variables such as *height*, *weight*, *firstName* etc.

Using an MLP block to predict a multinomial variable

Using the **MLP** block to predict the position of a player in basketball.

1. Import the `basketball_players.csv` dataset onto a blank Workflow canvas. Importing a CSV file is described in [Text File Import block](#) (page 184).
2. Expand the **Data Preparation** group in the Workflow palette, then click and drag a **Partition** block onto the Workflow canvas, beneath the `basketball_players.csv` dataset.

3. Double-click on the **Partition** block.

A **Configure Partition** dialog box opens.

4. Click on the **Add partition** button () do the following:

A third partition named **Partition3** is added to the **Configure Partition** dialog box.

5. Rename your partitions:

- a. Click on **Partition1** and type `Train` to rename the partition. Click on its corresponding weight and type 85.
- b. Click on **Partition2** and type `Validate` to rename the partition. Click on its corresponding weight and type 10.
- c. Click on **Partition3** and type `Test` to rename the partition. Click on its corresponding weight and type 5.
- d. Click **OK**.

The **Configure Partition** dialog box saves and closes. A green execution status is displayed in the **Output** port of the **Partition** block and the resulting datasets, `Train`, `Test` and `Validate`.

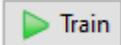
6. Expand the **Model Training** group in the Workflow palette, then click and drag an **MLP** block onto the Workflow canvas, beneath the `basketball_shots.csv` dataset.
7. Click on the `Train`, `Test` and `Validate` dataset block's **Output** port and drag a connection towards the **Input** port of the **MLP** block for each dataset.
8. Double-click on the **MLP** block.

A **Multilayer Perceptron** view opens, along with the **MLP Preferences** dialog box.

9. From the **MLP Preferences** dialog box:

- a. In the **Train** drop-down list, select **Train**.
- b. In the **Validate** drop-down list, select **Validate**.
- c. In the **Test** drop-down list, select **Test**.
- d. Click on the **Variable Selection** tab.
- e. In the **Dependent variable** drop-down list, select **position**.
- f. From the **Unselected Independent Variables** box, double-click on the variables *height*, *weight* and *goals_scored* to move them to the **Selected Independent Variables** box.
- g. Click on the **Optimiser** tab, from the **Optimiser:** drop-down list, select **ADAM**.
- h. Click on the **Stopping Criteria** tab, set the **Max training time (s)** to 15.
- i. Click **OK**.

The **MLP Preferences** dialog box closes.

- j. Click on the **Train** button () to train the MLP model.

A graph appears on the right hand side showing the live training error during model training.

- k. Save your **MLP** block. To save, either click the Save button () or press **Ctrl+S**.

- l. Close the **Multilayer Perceptron** view.

The **Multilayer Perceptron** view closes. A green execution status is displayed in the **Output** port of the **MLP** block.

10. From the **Scoring** grouping in the Workflow palette, click and drag a **Score** block onto the Workflow canvas, below the **MLP** block.

A **Score** block and empty dataset block are created on the Workflow canvas. Note that the **Score** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires a model and a dataset input.

11. Click on the `basketball_players.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Score** block.

12. Click on the **MLP** block **Output** port and drag a connection towards the **Input** port of the **Score** block.

The execution status of the **Score** block turns green and its dataset output is populated with the original dataset and the scored variables.

13. Right-click on the **Score** block's output dataset block and click **Rename**. Type `MLP Scored` and click **OK**.

You now have an MLP model that predicts the probability of the position of a player.

Reject Inference block

Enables you to use a *reject inference* method to address any inherent selection bias in a model by including a rejected population. The **Reject Inference** block takes two inputs: one from a logistic regression model and another from a dataset containing rejected records. The **Reject Inference** block then outputs a logistic regression model, labelled the **Inference Model**, that has been trained on both the performance of the accepted records and the inferred performance of the rejected records.

Configure Reject Inference

The **Configure Reject Inference** dialog box enables you to configure the **Reject Inference** block.

Opening the Configure Reject Inference dialog box

To open the **Configure Reject Inference** dialog box, double-click the **Reject Inference** block.

Inference Options

Inference Method

Choose from:

- **Proportional Assignment:** Assigns a good or bad status to each rejected record at random.
- **Simple Augmentation:** Assigns a good or bad status to each rejected record based on a threshold.
- **Fuzzy Augmentation:** Scores the rejects using the existing model.

Bad event

The category of the accepted dataset's dependent variable that defines 'bad' (for example, a boolean determining defaulting on a loan).

Inferred factor

The rate at which rejected records are considered bad compared with accepted records. If not specified, defaults to 3.

Seed

Used for random assignment operations within the model. If zero is specified, then seeding will be determined by the system clock during execution.

Iterations

Available if **Fuzzy Augmentation** is selected. The number of times to iterate the modelling process; higher numbers give greater accuracy, but take more processing time.

Convergence threshold

Available if **Fuzzy Augmentation** is selected. Determines the point at which the model is considered converged, so no more iterations are performed and the modelling process terminates early (before it reaches the **Iterations** specified). Convergence is defined by similarity between coefficients of parameters between successive models.

Model Variables

Dependent Variable

Read-only. Displays the binary dependent variable of the attached Logistic Regression model.

Event

Read-only. Displays the target category for which the probability of occurrence has been calculated.

Effect Variables

Allows you to specify effect variables. The two boxes show **Unselected Effect Variables** and **Selected Effect Variables**. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

Frequency Variable

Specifies a variable that defines the frequency of each observation.

Weight Variable

Specifies a variable that defines the prior weight associated with each observation.

Model Options

Configures the logistic model generated by the block. The options are the same as those in the **Logistic Regression** block.

Method

Specifies the model effect variable selection method. Choose from:

- **None**: Specifies that the fitted model includes every selected effect (independent) variable.
- **Forward**: Specifies the use of forward model selection.

The model initially contains intercept, if specified, and the selected effect (independent) variables added one at a time at each iteration of the model. The most significant variable (the effect variable with the smallest probability value and below the **Entry Significance Level**) is added at each iteration, and iterations continue until no specified effect variable is below the **Entry Significance Level**.

- **Backward:** Specifies the use of backward model selection.

All selected effect (independent) variables are included in the model, the variables are tested for significance, and the least significant dropped at each iteration of the model. The model is recalculated and the least-significant variable is dropped again. A variable will not be dropped, however, if it is below the value specified for this method in **Stay Significance Level**. When all variables are below this level, the model stops.

- **Stepwise:** Specifies that the model uses a combination of forward model and backward model selection.

The model initially contains an intercept, if specified, and one or more forward steps are taken to add effect variables to the model. Backward and forward steps are used to remove effect variables from and add effect variables to the model until no further improvement can be made to the model.

- **Score:** Specifies that the model returned has the best likelihood score statistic.

The model is created using a branch-and-bound method. Initially a model is created using the specified effect (independent) variable with the best individual likelihood score statistic. A second model is created using the two specified effect variables with the best combined likelihood score statistic. The model with the best likelihood score statistic is considered the best current solution and becomes the bound for the next iteration.

At each subsequent iteration, an effect variable is added to the model where the combination of all effect variables has the best likelihood score statistic. The model is compared to the best current solution and if the new model likelihood score statistic is better than the current solution, the new model becomes the bound for the next iteration.

When the best likelihood score statistic cannot be improved, the model is returned.

Fast

Available if **Method** is selected as **Backward**. Implements a quicker approximation for the model.

Link Function

Specifies how effect (independent) variables are linked to the dependent variable. Choose from:

- **CLOGLOG.** Specifies the *cloglog* (complementary log-log) function is used:

$$f(p) = \log(-\log(1-p))$$

Where p is the probability of the **Event** occurring.

- **LOGIT.** Specifies the *logit* function (the inverse of the logistic function) is used:

$$f(p) = \log\left(\frac{p}{1-p}\right)$$

Where p is the probability of the **Event** occurring.

- **PROBIT.** Specifies the *probit* function is used:

$$f(p) = \Phi^{-1}(p)$$

Where Φ^{-1} is the inverse cumulative distribution function of the standard normal distribution where the mean = 0 and variance = 1.

Entry Significance Level

Specifies the value below which a variable is included in the *Forward* or *Stepwise* methods.

Stay Significance Level

Specifies the value above which a variable is removed in the *Backward* or *Stepwise* methods.

Singular Tolerance

Specifies the tolerance value used to test for linear dependency among the effect (independent) variables.

Intercept

If selected, includes an intercept in the model. If cleared, the model will not include an intercept.

Block Outputs

The **Reject Inference** block can generate one or more of the following outputs:

- **Inferred Rejects:** A new dataset containing the inferred rejects.
- **Inference Report:** The report containing a comparison of the rejected dataset and the accepted dataset.
- **Inference Model:** A report containing the results of the Logistic Regression.

To specify the outputs, right-click on the **Reject Inference** block and select **Configure Outputs**. Two boxes are presented: **Unselected Output**, showing outputs that are not selected, and **Selected Output**, showing selected outputs. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

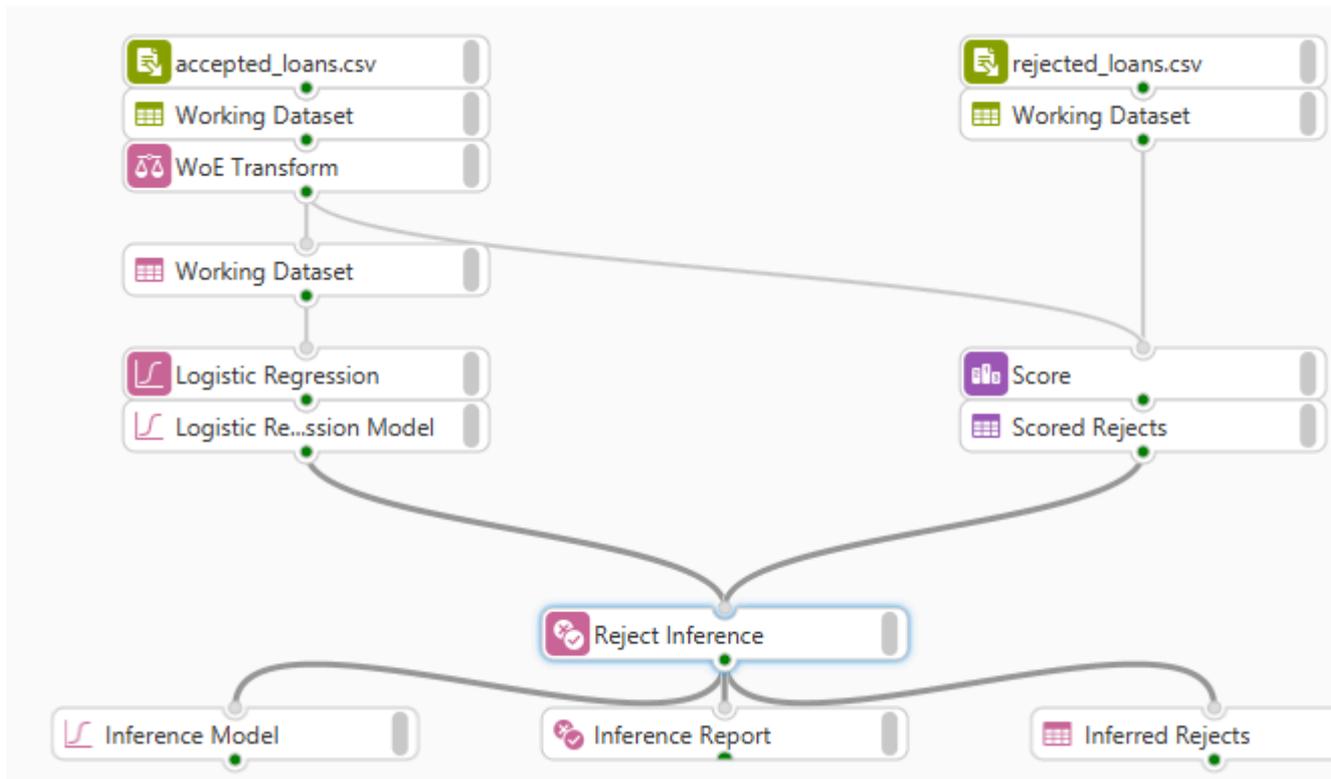
Reject Inference block example: building a model with rejected loan applications

This example demonstrates how the **Reject Inference** block is used to create a model and analyse rejected loan applicants that are otherwise unconsidered in traditional models.

In this example, a dataset of completed loans named `accepted_loans.csv` is imported into a blank Workflow. The dataset contains details of accepted loan applications and whether or not each loan defaulted. Another dataset of rejected loan applicants named `rejected_loans.csv` is also imported. A **WoE Transform** block uses the `accepted_loans.csv` dataset to transform several independent variables into several bins, using the optimal binning technique. A **Logistic Regression** block is used

to build a logistic regression model using the output variables from the **WoE Transform** block to target whether or not each loan defaulted. The **Score** block is then used to apply the **WoE Transform** model to the `rejected_loans.csv` dataset and is named `Scored Rejects`. The **Reject Inference** block then outputs an inference model using the `Scored Rejects` dataset and logistic regression model.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Accepted loan dataset

This example uses a dataset about loans, named `accepted_loans.csv`. Each observation in the dataset describes a completed loan and the person who took the loan out, with variables such as *Income*, *Loan_Period*, *Other_Debt* etc.

Rejected loan dataset

This example uses a dataset about rejected loan applications, named `rejected_loans.csv`. Each observation in the dataset describes the loan details and the loan applicant. This dataset contains the same variables as `accepted_loans.csv`, but without a *Default* variable.

Using the Reject Inference block to build an inference model

Using the **Reject Inference** block and rejected loan applications to build an inference model.

1. Import the `accepted_loans.csv` and `rejected_loans.csv` datasets into a blank Workflow. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **WoE Transform** block onto the Workflow canvas, beneath the `accepted_loans.csv` dataset.
3. Click on the `accepted_loans.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **WoE Transform** block.

Alternatively, you can dock the **WoE Transform** block onto the dataset block directly.

4. Double-click on the **WoE Transform** block.

A **WoE Transform Editor** view opens.

5. From the **Variable Selection** tab:
 - a. From the **Dependent Variable** drop-down list, select **Default**.
 - b. From the **Target Category** drop-down list, select **1**.
 - c. From the **Independent Variables** list, click on the drop-down boxes under **Treatment** and set **Other_Debt** to **Interval**, **Income** to **Interval**, **Sector** to **Nominal**, **Age** to **Interval** and **Housing_Situation** to **Nominal**.
 - d. Click on the **Optimisation** tab and click on **Apply optimal binning to all variables**.
 - e. Save your **WoE Transform** block. To save, either click the Save button () or press **Ctrl+S**.
 - f. Close the **WoE Transform Editor** view.

The **WoE Transform Editor** view closes. A green execution status is displayed in the **Output** ports of the **WoE Transform** block and the dataset that contains the applied transformation, **Working Dataset**.

6. Click and drag a **Logistic Regression** block onto the Workflow canvas, beneath the **WoE Transform** block.
7. Click on the **WoE Transform** block **Working Dataset**'s **Output** port and drag a connection towards the **Input** port of the **Logistic Regression** block.

Alternatively, you can dock the **Logistic Regression** block onto the dataset block directly.

8. Double-click on the **Logistic Regression** block.

A **Configure Logistic Regression** dialog box window opens.

9. From the **Variable Selection** tab:

- a. From the **Dependent Variable** drop-down list, select **Default**.
- b. From the **Event** drop-down list, select **1**.
- c. From the **Unselected Effect Variables** list, double-click **Other_Debt_WOE**, **Income_WOE**, **Sector_WOE**, **Age_WOE** and **Housing_Situation_WOE**.

All chosen variables are moved to the **Selected Effect Variables** list.

- d. From the **Selected Effect Variables** list, under the **Class** column, clear the checkbox for each variable.

10. From the **Model Selection** tab:

- a. From **Method**, select **Forward**.
- b. Click **OK**.

The **Configure Logistic Regression** dialog box closes. A green execution status is displayed in the **Output** ports of the **Logistic Regression** block and of its output **Logistic Regression Model**.

11. Click and drag a **Score** block onto the Workflow canvas, below the `rejected_loans.csv` dataset.

A **Score** block is created on the Workflow canvas. Note that the **Score** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires a model and a dataset input.

12. Click on the **WoE Transform** block **WoE Transform** and `rejected_loans.csv` dataset's **Output** port and drag a connection towards the **Input** port of the **Score** block.

The execution status of the **Score** block turns green and its dataset output is populated with the original dataset and the scored variables.

13. Right-click on the **Score** block's output dataset block and click **Rename**. Type `Scored Rejects` and click **OK**.

14. Click and drag a **Reject Inference** block onto the Workflow canvas, beneath the **Logistic Regression** block and **Score** block.

15. Click on the **Score** block **Scored Rejects Output** port and drag a connection towards the **Input** port of each **Reject Inference** block.

16. Click on the **Logistic Regression** block **Logistic Regression Models Output** port and drag a connection towards the **Input** port of the **Reject Inference** block.

17. Double-click on the **Reject Inference** block.

A **Configure Reject Inference** dialog box opens.

18. From the **Inference Options** tab:

- a. From the **Inference Method** drop-down list, select **Proportional Assignment**.
- b. From the **Bad Event** drop-down list, select **1**.
- c. Click **OK**.

The **Configure Reject Inference** dialog box closes. A green execution status is displayed in the **Output** ports of the **Reject Inference** block and of its outputs **Inference Model**, **Inference Report** and **Inferred Rejects**.

You now have a reject inference model, that predicts loan defaulting with rejected applications considered.

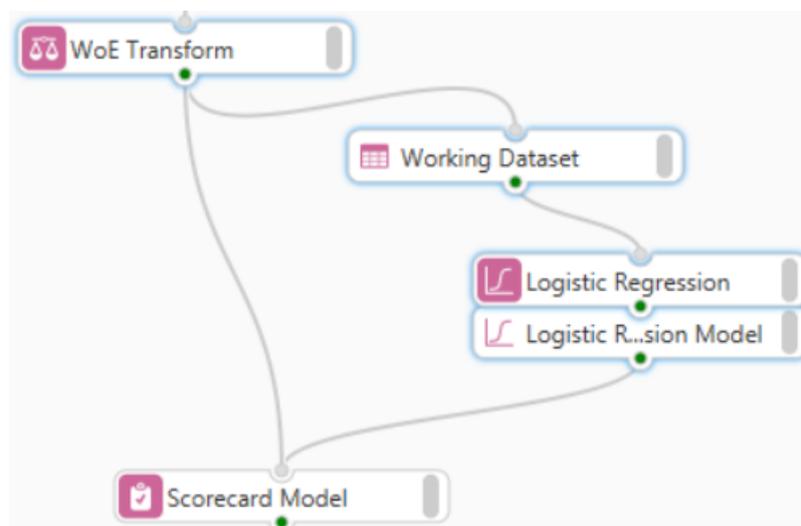
Scorecard Model block

Enables the generation of a standard scorecard (credit scorecard) and its deployment code that can be copied into a SAS language program for testing or production use.

The **Scorecard Model** block creates a scorecard model consisting of a set of attributes each with an assigned weighted score (either positive or negative). The sum of those scores equals the final credit score representing the lending risk.

The **Scorecard Model** block requires two inputs:

- The transformation model from a **WoE Transform** block.
- A logistic regression model created from the working dataset of the same **WoE Transform** block.



The **Scorecard Model** block is configured using the **Scorecard Editor** view, which can be opened by double-clicking the **Scorecard Model** block.

Point Allocation

Scaling Parameters

Base Points

Specifies the number of points to represent the **Base Odds**.

Base Odds

Specifies the number of *Bad* category entries for each *Good* category entry.

Points to Double the Odds

Specifies the number of points decrease in score at which the probability of default is doubled.

Higher Score Category

Specifies which category of borrower the higher scores are allocated to.

- *Good* indicates borrowers that have a lower credit risk
- *Bad* indicates borrowers that have a higher credit risk.

Scorecard

Displays the variables and associated scores making up the credit scorecard.

Scoring Code

Displays the scorecard model code. The code is available as SAS language or SQL language code, to be used in a **SAS Language** block or **SQL Language** block respectively.

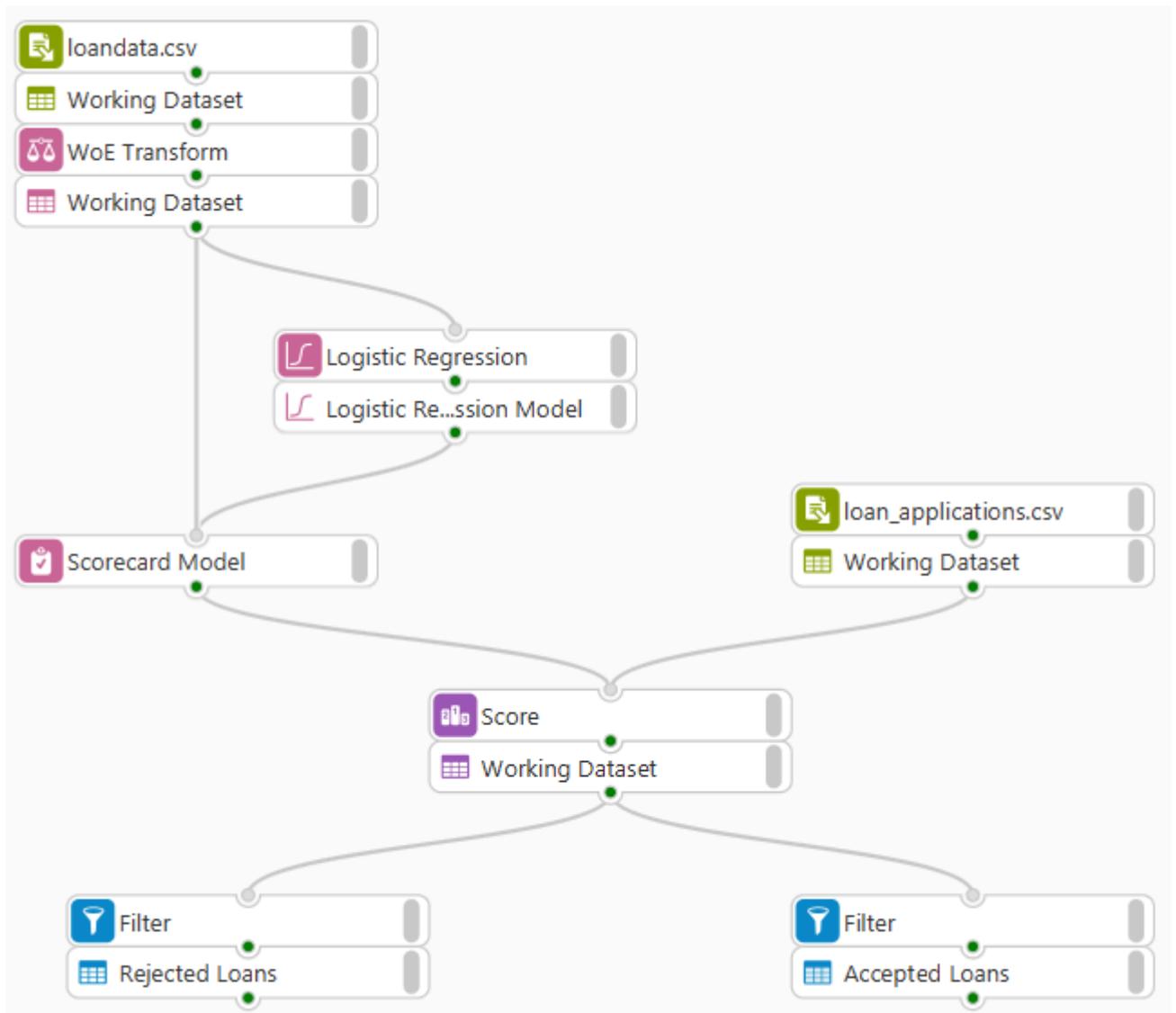
The final generated code can be copied using **Copy generated text to clipboard** .

Scorecard Model block example: building a scorecard process

This example demonstrates how the **Scorecard Model** block is used to construct a scorecard process. The scorecard generated is applied to a dataset of loan applications, and the results filtered into loan rejections and loan acceptances.

In this example, a dataset of completed loans named `loandata.csv` is imported into a blank Workflow. The dataset contains details of past loan applications and whether or not each loan defaulted. A **WoE Transform** block is used to transform several independent variables into bins, using an optimal binning technique. A **Logistic Regression** block is used to build a logistic regression model using the output variables from the **WoE Transform** block to target whether or not each loan defaulted. The **Scorecard Model** block is then configured to output a scorecard model and, using a **Score** block, this is applied against a dataset of new loan applications called `loan_application.csv`. The **Score** block then outputs a dataset appended with the probability of defaulting for each loan application, which, using the score, is then filtered with two **Filter** blocks into two datasets of loan rejections and loan acceptances.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Loan dataset

This example uses a dataset about loans, named `loandata.csv`. Each observation in the dataset describes a past loan and the person who took loan out, with variables such as *Income*, *Loan_Period*, *Other_Debt* etc.

Loan applications dataset

This example uses a dataset about loan applications, named `loan_applications.csv`. Each observation in the dataset describes an application for a loan. This dataset contains the same variables as `loandata.csv`, but without a *Default* variable.

Using the Scorecard Model block for the scorecard modelling process

Using several blocks to set up the the scorecard modelling process and the **Scorecard Model** block to output the results.

1. Import the `loandata.csv` and `loan_application.csv` datasets into a blank Workflow. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **WoE Transform** block onto the Workflow canvas, beneath the `loandata.csv` dataset.
3. Click on the `loandata.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **WoE Transform** block.

Alternatively, you can dock the **WoE Transform** block onto the dataset block directly.

4. Double-click on the **WoE Transform** block.

A **WoE Transform Editor** view opens.

5. From the **Variable Selection** tab:
 - a. From the **Dependent Variable** drop-down list, select **Default**.
 - b. From the **Target Category** drop-down list, select **1**.
 - c. From the **Independent Variables** list, click on the drop-down boxes under **Treatment** and set **Other_Debt** to **Interval**, **Income** to **Interval**, **Sector** to **Nominal**, **Age** to **Interval** and **Housing_Situation** to **Nominal**.
 - d. Click on the **Optimisation** tab and click on **Apply optimal binning to all variables**.
 - e. Save your **WoE Transform** block. To save, either click the Save button () or press **Ctrl+S**.
 - f. Close the **WoE Transform Editor** view.

The **WoE Transform Editor** view closes. A green execution status is displayed in the **Output** ports of the **WoE Transform** block and the dataset that contains the applied transformation, **Working Dataset**.

6. Click and drag a **Logistic Regression** block onto the Workflow canvas, beneath the **WoE Transform** block.
7. Click on the **WoE Transform** block **Working Dataset's** **Output** port and drag a connection towards the **Input** port of the **Logistic Regression** block.

Alternatively, you can dock the **Logistic Regression** block onto the dataset block directly.

8. Double-click on the **Logistic Regression** block.

A **Configure Logistic Regression** dialog box window opens.

9. From the **Variable Selection** tab:

- a. From the **Dependent Variable** drop-down list, select **Default**.
- b. From the **Event** drop-down list, select **1**.
- c. From the **Unselected Effect Variables** list, double-click **Other_Debt_WOE**, **Income_WOE**, **Sector_WOE**, **Age_WOE** and **Housing_Situation_WOE**.

All chosen variables are moved to the **Selected Effect Variables** list.

- d. From the **Selected Effect Variables** list, under the **Class** column, clear the checkbox for each variable.

10. From the **Model Selection** tab:

- a. From **Method**, select **Forward**.
- b. Click **OK**.

The **Configure Logistic Regression** dialog box closes. A green execution status is displayed in the **Output** ports of the **Logistic Regression** block and of its output **Logistic Regression Model**.

11. Click and drag a **Scorecard Model** block onto the Workflow canvas, below the new **Logistic Regression** block.

A **Scorecard Model** block is created on the Workflow canvas. Note that the **Scorecard Model** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires input.

12. Click on the **WoE Transform** block **WoE Transform** and **Logistic Regression** block **Logistic Regression Model Output** port and drag a connection towards the **Input** port of the **Scorecard Model** block.

13. Double-click on the **Scorecard Model** block.

A **Scorecard Editor** view opens.

14. From the **Point Allocation** tab:

- a. In the **Base points** box, type 500.
- b. In the **Points to double the odds** box, type 60.
- c. In the **Higher Score Category** pane, select **Good**.
- d. Save your **Scorecard Model** block. To save, either click the Save button () or press **Ctrl+S**.
- e. Close the **Scorecard Editor** view.

The **Scorecard Editor** view closes. A green execution status is displayed in the **Output** ports of the **Scorecard Model** block.

15. From the **Scoring** grouping in the Workflow palette, click and drag a **Score** block onto the Workflow canvas, below the **Scorecard Model** block and the `loan_applications.csv` dataset.

A **Score** block and empty dataset block are created on the Workflow canvas. Note that the **Score** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires a model and a dataset input.

16. Click on the `loan_applications.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Score** block.

17. Click on the **Scorecard Model** block **Working Dataset Output** port and drag a connection towards the **Input** port of the **Score** block.

The execution status of the **Score** block turns green and its dataset output is populated with the original dataset and the scored variable.

18. From the **Data Preparation** grouping in the Workflow palette, click and drag two **Filter** blocks onto the Workflow canvas.

19. Click on the **Score** block **Output** port and drag a connection towards the **Input** port of each **Filter** block.

20. Configure the first **Filter** block:

a. Double-click on a **Filter** block.

A **Filter Editor** view opens.

b. Click on the **Basic** tab.

c. From the **Variable** drop-down box, select **Score**.

d. From the **Operator** drop-down box, select **<**.

e. In the **Value** box, enter 540.

f. Save your **Filter** block. To save, either click the Save button () or press **Ctrl+S**.

g. Close the **Filter Editor** view.

The **Filter Editor** view. A green execution status is displayed in the **Output** port of the **Filter** block and the filtered dataset, **Working Dataset**.

h. Right-click on the **Filter** block's output dataset block and click **Rename**. Type `Rejected Loans` and click **OK**.

21. Now configure the second **Filter** block:

a. Double-click on the other **Filter** block.

A **Filter Editor** view opens.

b. Click on the **Basic** tab.

c. For the **Variable** drop-down box, select **Score**.

d. For the **Operator** drop-down box, select **>=**.

e. In the **Value** box, enter 540.

f. Save your **Filter** block. To save, either click the Save button () or press **Ctrl+S**.

g. Close the **Filter Editor** view.

The **Filter Editor** view closes. A green execution status is displayed in the **Output** port of the **Filter** block and the filtered dataset, **Working Dataset**.

h. Right-click on the **Filter** block's output dataset block and click **Rename**. Type `Accepted Loans` and click **OK**.

You now have a scorecard process that scores loan application, which are then sorted into acceptances and rejections.

WoE Transform block

Enables you to measure the influence of an independent variable on a specified dependent variable. This calculates the Weight of Evidence, or WoE.

You can use the **WoE Transform** block to measure risk, for example to test hypotheses such as the risk of loan default based on area of residence if your dataset is grouped by geographic areas.

A **WoE Transform** block is edited through the **WoE Transform Editor** view. To open the **WoE Transform Editor** view, double-click the **WoE Transform** block.

The **WoE Transform Editor** view is used to select a dependent variable, its target category, and the most influential independent variables and their treatment.

Variable Selection view

Dependent Variable

Specifies the target variable for the calculation.

Target Category

Specifies the target category for which the probability of occurrence is to be calculated.

Frequency Variable

Specifies a variable that defines the frequency of each observation.

Independent Variables

Enables you to select the most influential independent variables for the WoE transform.

Variable

Displays the name of all independent variables in the input dataset.

Entropy Variance

Displays the *Entropy Variance* value for each variable in relation to the specified **Dependent Variable**. For more information, see *Predictive power criteria* [↗](#) (page 102)

Treatment

Enables you to specify the type of each variable.

Excluded

Specifies the variable is excluded from Weight of Evidence calculations.

Interval

Specifies a continuous independent (input) variable.

Nominal

Specifies a discrete independent (input) variable with no implicit ordering.

Ordinal

Specifies a discrete independent (input) variable with an implicit category ordering.

Monotonic WoE

Specifies that the Weight of Evidence value for the variable is either monotonically increasing or monotonically decreasing.

Frequency Chart

Shows the effect the selected independent variable has on each potential target category for the specified dependent variable.

Optimisation view

Displays the output from transforming the specified independent variables using optimal binning.

Independent variable

Specifies the variable to use in the weight of evidence transformation. The list contains the variables selected in the **Variable Selection** view.

Binning Definition

Displays the effects of binning the specified independent variable. Click **Calculate optimal binning**  to optimally bin the selected independent variable. The table then displays the classes used for each bin and the Workbench-generated label for the bin, which can be modified if required.

WoE Transformation

Displays a table with information about each optimal bin created including the:

- Number of observations in the bin.
- Number of observations in the bin matching the **Target Category** specified for the dependent variable.
- WoE for each bin.
- Information value for each bin.

The panel contains graphs showing the percentage of the specified target category to the non-target category for all bins, the Weight of Evidence for each bin, and the total number of observations (count) in each bin.

The charts can be edited and saved. Click **Edit chart**  at the top of the required chart to open the Chart Editor from where the chart can be saved to the clipboard. The frequency data used to create each node can be saved by clicking **Copy data to clipboard**  at the top of the required chart to save the table of data.

Transformation Code

Displays the transformation code for the WoE calculations. The code is available as either SQL, for use in the **SQL Language** block, or as SAS language code to be used in a `DATA` step in the **SAS Language** block.

Click **Options** to open the **Source Generation Options** dialog box to modify the names of the transformation variables in the generated code.

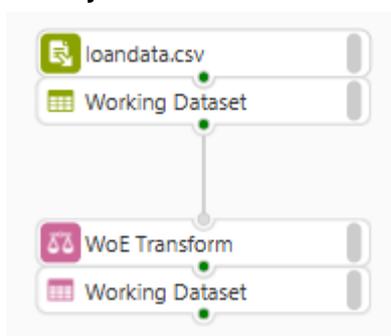
The final generated code can be copied using **Copy generated text to clipboard** .

WoE Transform block basic example

This example demonstrates how the **WoE Transform** block can be used to transform variables.

In this example, a dataset of completed loans named `loandata.csv` is imported into a blank Workflow. The dataset contains details of past loan applications and whether or not each loan defaulted. A **WoE Transform** block is used to transform several variables into several bins, using an optimal binning technique.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Loan dataset

This example uses a dataset about loans, named `loandata.csv`. Each observation in the dataset describes a past loan and the person who took loan out, with variables such as *Income*, *Loan_Period*, *Other_Debt* etc.

Using a WoE Transform block to transform variables

Using the **WoE Transform** block to transform variables into a finite number of bins and output the results.

1. Import the `loandata.csv` dataset into a blank Workflow. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **WoE Transform** block onto the Workflow canvas, under the `loandata.csv` dataset.

3. Click on the `loandata.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **WoE Transform** block.

Alternatively, you can dock the **WoE Transform** block onto the dataset block directly.

4. Double-click on the **WoE Transform** block.

A **WoE Transform Editor** view opens.

5. From the **Variable Selection** tab:
 - a. From the **Dependent Variable** drop-down list, select **Default**.
 - b. From the **Target Category** drop-down list, select **1**.
 - c. From the **Independent Variables** list, click on the drop-down boxes under **Treatment** and set **Income** to **Interval** and **Housing_Situation** to **Nominal**.
 - d. Click on the **Optimisation** tab and click on **Apply optimal binning to all variables**.
 - e. Save your **WoE Transform** block. To save, either click the Save button () or press **Ctrl+S**.
 - f. Close the **WoE Transform Editor** view.

The **WoE Transform Editor** view closes. A green execution status is displayed in the **Output** ports of the **WoE Transform** block and the dataset that contains the applied transformation, **Working Dataset**.

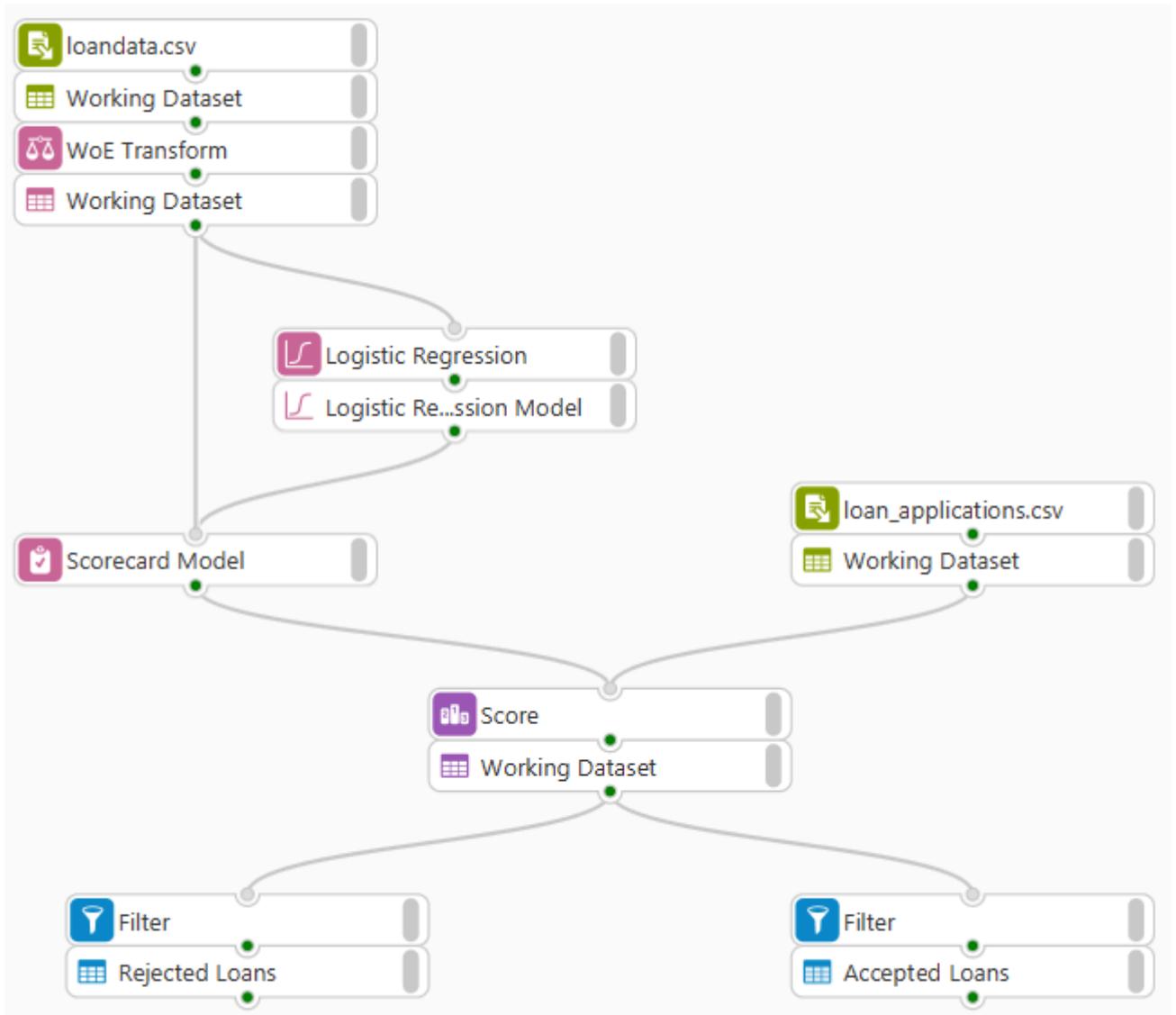
You now have a WoE transformation model that outputs transformed variables.

WoE Transform block example: use as part of a scorecard process

This example demonstrates how the **WoE Transform** block can be used as part of a scorecard process. The workflow generates a scorecard, which is then applied to a dataset of loan applications, and the results filtered into loan rejections and loan acceptances.

In this example, a dataset of completed loans named `loandata.csv` is imported into a blank Workflow. The dataset contains details of past loan applications and whether or not each loan defaulted. A **WoE Transform** block is used to transform several independent variables into bins, using an optimal binning technique. A **Logistic Regression** block is used to build a logistic regression model using the output variables from the **WoE Transform** block to target whether or not each loan defaulted. The **Scorecard Model** block is then configured to output a scorecard model and, using a **Score** block, this is applied against a dataset of new loan applications called `loan_application.csv`. The **Score** block then outputs a dataset appended with the probability of defaulting for each loan application, which, using the score, is then filtered with two **Filter** blocks into two datasets of loan rejections and loan acceptances.

Workflow Layout



Datasets used

The file used in this example can be found in the samples distributed with WPS Analytics.

Loan dataset

This example uses a dataset about loans, named `loandata.csv`. Each observation in the dataset describes a past loan and the person who took loan out, with variables such as *Income*, *Loan_Period*, *Other_Debt* etc.

Loan applications dataset

This example uses a dataset about loan applications, named `loan_applications.csv`. Each observation in the dataset describes an application for a loan. This dataset contains the same variables as `loandata.csv`, but without a *Default* variable.

Using the WoE Transform block as part of a scorecard process.

Using the **WoE Transform** block with other blocks to create a scorecard modelling process.

1. Import the `loandata.csv` and `loan_application.csv` datasets into a blank Workflow. Importing a CSV file is described in *Text File Import block* [↗](#) (page 184).
2. Expand the **Model Training** group in the Workflow palette, then click and drag a **WoE Transform** block onto the Workflow canvas, under the `loandata.csv` dataset.
3. Click on the `loandata.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **WoE Transform** block.

Alternatively, you can dock the **WoE Transform** block onto the dataset block directly.

4. Double-click on the **WoE Transform** block.

A **WoE Transform Editor** view opens.

5. From the **Variable Selection** tab:
 - a. From the **Dependent Variable** drop-down list, select **Default**.
 - b. From the **Target Category** drop-down list, select **1**.
 - c. From the **Independent Variables** list, click on the drop-down boxes under **Treatment** and set **Other_Debt** to **Interval**, **Income** to **Interval**, **Sector** to **Nominal**, **Age** to **Interval** and **Housing_Situation** to **Nominal**.
 - d. Click on the **Optimisation** tab and click on **Apply optimal binning to all variables**.
 - e. Save your **WoE Transform** block. To save, either click the Save button () or press **Ctrl+S**.
 - f. Close the **WoE Transform Editor** view.

The **WoE Transform Editor** view closes. A green execution status is displayed in the **Output** ports of the **WoE Transform** block and the dataset that contains the applied transformation, **Working Dataset**.

6. Click and drag a **Logistic Regression** block onto the Workflow canvas, beneath the **WoE Transform** block.
7. Click on the **WoE Transform** block **Working Dataset's** **Output** port and drag a connection towards the **Input** port of the **Logistic Regression** block.

Alternatively, you can dock the **Logistic Regression** block onto the dataset block directly.

8. Double-click on the **Logistic Regression** block.

A **Configure Logistic Regression** dialog box window opens.

9. From the **Variable Selection** tab:

- a. From the **Dependent Variable** drop-down list, select **Default**.
- b. From the **Event** drop-down list, select **1**.
- c. From the **Unselected Effect Variables** list, double-click **Other_Debt_WOE**, **Income_WOE**, **Sector_WOE**, **Age_WOE** and **Housing_Situation_WOE**.

All chosen variables are moved to the **Selected Effect Variables** list.

- d. From the **Selected Effect Variables** list, under the **Class** column, clear the checkbox each variable.

10. From the **Model Selection** tab:

- a. From **Method**, select **Forward**.
- b. Click **OK**.

The **Configure Logistic Regression** dialog box closes. A green execution status is displayed in the **Output** ports of the **Logistic Regression** block and of its output **Logistic Regression Model**.

11. Click and drag a **Scorecard Model** block onto the Workflow canvas, below the new **Logistic Regression** block.

A **Scorecard Model** block is created on the Workflow canvas. Note that the **Scorecard Model** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires input.

12. Click on the **WoE Transform** block **WoE Transform** and **Logistic Regression** block **Logistic Regression Model Output** port and drag a connection towards the **Input** port of the **Scorecard Model** block.

13. Double-click on the **Scorecard Model** block.

A **Scorecard Editor** view opens.

14. From the **Point Allocation** tab:

- a. In the **Base points** box, type 500.
- b. In the **Points to double the odds** box, type 60.
- c. In the **Higher Score Category** pane, select **Good**.
- d. Save your **Scorecard Model** block. To save, either click the Save button () or press **Ctrl+S**.
- e. Close the **Scorecard Editor** view.

The **Scorecard Editor** view closes. A green execution status is displayed in the **Output** ports of the **Scorecard Model** block.

15. From the **Scoring** grouping in the Workflow palette, click and drag a **Score** block onto the Workflow canvas, below the **Scorecard Model** block and the `loan_applications.csv` dataset.

A **Score** block and empty dataset block are created on the Workflow canvas. Note that the **Score** block's configuration status (the right-hand bar of the block) is red and displays a screentip stating that the block requires a model and a dataset input.

16. Click on the `loan_applications.csv` dataset block's **Output** port and drag a connection towards the **Input** port of the **Score** block.

17. Click on the **Scorecard Model** block **Working Dataset Output** port and drag a connection towards the **Input** port of the **Score** block.

The execution status of the **Score** block turns green and its dataset output is populated with the original dataset and the scored variable.

18. From the **Data Preparation** grouping in the Workflow palette, click and drag two **Filter** blocks onto the Workflow canvas.

19. Click on the **Score** block **Output** port and drag a connection towards the **Input** port of each **Filter** block.

20. Configure the first **Filter** block:

a. Double-click on a **Filter** block.

A **Filter Editor** view opens.

b. Click on the **Basic** tab.

c. From the **Variable** drop-down box, select **Score**.

d. From the **Operator** drop-down box, select **<**.

e. In the **Value** box, enter `540`.

f. Save your **Filter** block. To save, either click the Save button () or press **Ctrl+S**.

g. Close the **Filter Editor** view.

The **Filter Editor** view. A green execution status is displayed in the **Output** port of the **Filter** block and the filtered dataset, **Working Dataset**.

h. Right-click on the **Filter** block's output dataset block and click **Rename**. Type `Rejected Loans` and click **OK**.

21. Now configure the second **Filter** block:

a. Double-click on the other **Filter** block.

A **Filter Editor** view opens.

b. Click on the **Basic** tab.

c. For the **Variable** drop-down box, select **Score**.

d. For the **Operator** drop-down box, select **>=**.

e. In the **Value** box, enter `540`.

f. Save your **Filter** block. To save, either click the Save button () or press **Ctrl+S**.

g. Close the **Filter Editor** view.

The **Filter Editor** view closes. A green execution status is displayed in the **Output** port of the **Filter** block and the filtered dataset, **Working Dataset**.

h. Right-click on the **Filter** block's output dataset block and click **Rename**. Type `Accepted Loans` and click **OK**.

You now have a scorecard process that scores loan application, which are then sorted into acceptances and rejections.

Scoring group

Contains blocks that enable you to analyse and score models.

Analyse Models block ↗	329
Enables analysis of a scored dataset to determine performance of a model.	
Score block ↗	330
Enables you to test multiple models against the same dataset, or to apply an existing model to a new dataset.	
PSI block ↗	331
Population Stability Index (PSI) assesses how a scorecard has changed over time.	

Analyse Models block

Enables analysis of a scored dataset to determine performance of a model.

Overview

The **Analyse Models** block accepts a connection from a scored dataset. To test the same model with multiple datasets, create one **Analyse Models** block and connect each scored dataset to the same block.

Configure Analyse Models

Double-clicking on a new **Analyse Models** block will open the **Configure Analyse Models** window, which has two tabs:

Analysis Type

Choose the type of model analysis:

- **Classification:** Performs a classification analysis on a predicted variable, comparing a chosen truth category against the predicted probability from the dataset.
- **Regression:** Performs a regression analysis on a predicted variable, comparing it against the observed value from the dataset.

Variable Selection

If you have selected a classification analysis, select the following for each dataset:

- **True class:** the variable that's being predicted.
- **Truth category:** the value of the **True class** variable that the model is aiming to predict.

- **Predicted probability:** probability of occurrence for the truth category value.

If you have selected a regression analysis, for each dataset, select:

- **Outcome variable:** the variable that's being predicted.
- **Predicted result:** the model's prediction for the outcome variable.

Report

The **Analyse Models** block generates a report showing charts and statistics. To display the report, **right-click** on the report and select **Open Report**.

Classification Analysis

If you have selected a classification analysis, the report contains a **Summary** tab showing all charts and statistics, along with a tab for each chart: **Gains Chart**, **K-S Chart**, **ROC Chart** and **Lift Chart** for the model. When in its full screen tab view, the **K-S Chart** allows you to toggle its constituent plots on and off by clicking on the legend items.

Regression Analysis

If you have selected a regression analysis, the report contains a **Summary** tab showing all statistics, along with a tab for each chart: **Normal Q-Q plots**, **Predicted vs Outcome Q-Q Plots**, **Residual Histograms**, **Residual Scatterplots** and **Predicted vs Outcome Scatterplots** for the model.

Score block

Enables you to test multiple models against the same dataset, or to apply an existing model to a new dataset.

Testing multiple models

To test multiple models, create one **Score** block for each model you want to test, and connect the same dataset to each **Score** block. The output from the multiple blocks will enable you to identify the preferred model to deploy.

Applying an existing model to a new dataset

To apply an existing model to a new dataset, configure a model output from a compatible block, for example a block used against a training dataset, and connect that model output to the score block along with a connection from the new dataset. The score block will then output a new dataset that is the result of the connected model being applied to the connected dataset.

PSI block

Population Stability Index (PSI) assesses how a scorecard has changed over time.

Block inputs

The **PSI** block requires two inputs: a *reference dataset* containing the expected distribution of scores, in labelled bins; and a second *current dataset* containing the current distribution of scores, also in labelled bins. The block will analyse the bin label variable from each dataset, counting the observations for each bin in the reference dataset and comparing it to that of the current dataset.

Block outputs

The **PSI** block has the following outputs:

- The block label can be set to show the primary metric for the block (either PSI or p-value).
- A coloured border around the **PSI** block's comment indicator indicating the PSI and how it compares to specified thresholds.
- An automatically generated comment for the **PSI** block, giving a summary of the block's results.
- A **PSI Report** view, selected as an output by default, but can be removed by right-clicking on the PSI block and selecting **Configure Outputs**.
- A **Distributions Dataset**, giving the block's outputs as a dataset. This is not selected by default, but can be selected by right-clicking on the PSI block and selecting **Configure Outputs**.
- **Summary Statistics**. This is not selected by default, but can be selected by right-clicking on the PSI block and selecting **Configure Outputs**.

Configure PSI

Enables you to configure the PSI block.

The **Configure PSI** window allows you to specify the following:

Reference dataset

Choose the reference dataset against which comparisons will be made.

Reference binned variable

Choose the reference binned variable from the reference dataset. This variable will be a list of bin labels, for example assigned from another continuous variable.

Current binned variable

Choose the binned variable from the current dataset to compare with the reference binned variable from the reference dataset. This variable will be a list of bin labels, for example assigned from another continuous variable.

Automatically update block label and comment

When selected, this will display the primary metric (specified below) as a label on the block in the Workflow, and automatically generate a comment with further detail. Both of these indicators will be updated if the data changes. This automatic update will overwrite any manual edits you have made to the block label or comment.

When not selected, the automatic block label and comment will not be created.

Minor change threshold

Sets the threshold for a minor change in the p-value. When the p-value falls below this value, this will be noted as a minor change in the block's outputs.

Major change threshold

Sets the threshold for a major change in the p-value to be noted in the block's outputs. When the p-value falls below this value, this will be noted as a major change in the block's outputs.

Primary metric

Choose between PSI and p-value to display on the block label and as the primary metric in the report view.

PSI Report view

Displays the block's output in detail.

Summary

The summary at the top of the report view shows:

- A colour coded indicator showing:
 - Green: **No significant change**. The p-value is above the specified **Minor change threshold**.
 - Amber: **Minor change**. The p-value is below the specified **Minor change threshold**.
 - Red: **Major change**. The p-value is below the specified **Major change threshold**.
- **PSI**.
- **p-value**.

Current and Reference Distributions

A table with a row for each reference bin, displaying:

- **Reference Bin**: The label for the bin.
- **Reference Frequency**: The number of items in the bin for the reference dataset.
- **Reference Percent**: The percentage of the total items that this bin's items represent for the reference dataset.
- **Current Frequency**: The number of items in the bin for the current dataset.

- **Current Percent:** The percentage of the total items that this bin's items represent for the current dataset.
- **Current - Reference:** The difference between the current and reference percentages.
- **Index:** The PSI for each bin.

Export group

Contains blocks that enable you to export data from a Workflow.

Data exported can be saved in either the current workspace or to the file system accessible through the active Workflow link.

Chart Builder block ↗	333
Enables you to export a working dataset to a variety of different charts.	
Delimited File Export block ↗	341
Enables you to export a working dataset to a text file, specifying the field delimiters.	
Excel Export block ↗	342
Enables you to export a dataset from a Workflow to an Excel Workbook.	
Database Export block ↗	343
Enables you to export a dataset from a Workflow to a specified database. This block has Auto Run disabled by default.	
Tableau Export block ↗	345
The Tableau Export block allows you to export a Workbench dataset to Tableau Data Extract file format (.tde) and, if required, uploaded to a Tableau server. The .tde file can be stored either within a Workbench workspace, or in a specified external folder.	

Chart Builder block

Enables you to export a working dataset to a variety of different charts.

Overview

The **Chart Builder** block requires a single dataset input and can generate one or many charts from variables contained within that dataset.

Charts are created and edited using the Chart Builder editor, which is opened by double-clicking on the **Chart Builder** block. The Chart Builder editor is divided into five areas:

Note:

Changes to a chart must be saved for them to appear in the preview area or Chart Viewer output.

Chart Setup

At the top of the Chart Builder tab are the following options to configure labels and size for the chart:

Title

A title, displayed above the chart.

Footnote

A footnote, displayed below the chart.

Width

The width of the chart, in pixels.

Height

Specifies the height of the chart, in pixels.

Plots

Allows you to add, delete and configure one or many plots to appear on the chart.

If there are multiple plots, the order of the plots can be changed, which determines their display order on the chart.

Only plots compatible with the input dataset are displayed. If adding additional plots to a chart, the list of available plots is further constrained to plots compatible with existing plots on that chart.

Each plot type has its own set of definable variables and parameters, which appear in the **Options** panel.

Plot Type	Plot Variables and Parameters
Band: Plots a series plot (line graph) together with a shaded band.	<p>X: the main series to be plotted.</p> <p>Upper: the upper limit of the band.</p> <p>Lower: the lower limit of the band.</p>
Bubble: Plots isolated points from two variables, one assigned to the x axis and one to the y axis. Each point is represented by a bubble.	<p>X: the variable to be plotted on the x axis.</p> <p>Y: the variable to be plotted on the y axis.</p> <p>Size: the variable used to determine the size of each bubble.</p>
Density Plots a density plot for a specified variable, with the x axis showing variable values, and the y axis showing probability density.	<p>Variable: the variable to be plotted.</p>
Ellipse: When selected in addition to a scatter plot, draws an ellipse around values that fall within a specified percentile of the total observations.	<p>X: the variable to be plotted on the x axis.</p> <p>Y: the variable to be plotted on the y axis.</p>

Plot Type	Plot Variables and Parameters
<p>High Low: Plots three series plots (line graphs) on one axes: a main series and an upper and lower series.</p>	<p>X: the main series to be plotted.</p> <p>Upper: the upper series.</p> <p>Lower: the lower series.</p>
<p>Histogram: Plots a histogram for a specified variable, with the x axis showing automatically binned variable values, and the y axis showing percentages of the dataset's observations.</p>	<p>Variable: the variable to be plotted.</p>
<p>Horizontal Bar: Plots a horizontal bar graph for a specified variable, with the y axis showing variable values and the x axis showing frequency of occurrence for those values.</p>	<p>Response variable: the variable to be plotted.</p>
<p>Horizontal Bar (parameterized): Plots a horizontal bar graph for a specified variable, categorised by another variable. The x axis shows variable values and the y axis shows the categories.</p>	<p>Category: the variable to be used for categorisation.</p> <p>Response: the variable to be plotted.</p>
<p>Horizontal Box: Plots a single variable on the x axis, drawing a box centered on the variable's median value, with sides set at the 25th and 75th percentile. Also shown are whiskers on the left and right sides of the box. The lower (left side) whisker is the lowest value still within 1.5 times the interquartile range of the lower quartile. The upper (right side) whisker is the highest value still within 1.5 times the interquartile range of the upper quartile.</p>	<p>Response variable: the variable to be plotted.</p>
<p>Horizontal Line: Plots a line graph for a specified variable, with the y axis showing variable values, and the x axis showing frequency of occurrence.</p>	<p>Response variable: the variable to be plotted.</p>
<p>LOESS: Plots isolated points from two variables, one assigned to the x axis and one to the y axis. Each point is represented by a diamond. Also performs a LOESS (Locally Weighted Scatterplot Smoothing) fit to the data and plots this as a dotted line on the same axes.</p>	<p>X: the variable to be plotted on the x axis.</p> <p>Y: the variable to be plotted on the y axis.</p>
<p>Needle: Plots isolated points from two variables, one assigned to the x axis and one to the y axis. Each point is joined to the x axis with a line.</p>	<p>X: the variable to be plotted on the x axis.</p> <p>Y: the variable to be plotted on the y axis.</p>

Plot Type	Plot Variables and Parameters
<p>Penalized B-Spline: Plots isolated points from two variables, one assigned to the x axis and one to the y axis. Each point is represented by a diamond. Also performs a Penalized B-Spline fit on the dataset and plots this as a dotted red line on the same axes.</p>	<p>X: the variable to be plotted on the x axis. Y: the variable to be plotted on the y axis.</p>
<p>Regression: Plots isolated points from two variables, one assigned to the x axis and one to the y axis. Each point is represented by a diamond. Also performs a linear regression on the dataset and plots the resultant model as a dotted red line on the same axes .</p>	<p>X: the variable to be plotted on the x axis. Y: the variable to be plotted on the y axis.</p>
<p>Scatter: Plots isolated points from two variables, one assigned to the x axis and one to the y axis. Each point is represented by a diamond.</p>	<p>X: the variable to be plotted on the x axis. Y: the variable to be plotted on the y axis.</p>
<p>Series: Plots points from two variables, one assigned to the x axis and one to the y axis. All points are joined directly with lines.</p>	<p>X: the variable to be plotted on the x axis. Y: the variable to be plotted on the y axis.</p>
<p>Step: Plots points from two variables, one assigned to the x axis and one to the y axis. All points are joined with horizontal and vertical lines, creating steps.</p>	<p>X: the variable to be plotted on the x axis. Y: the variable to be plotted on the y axis.</p>
<p>Vector: Plots points from two variables, one assigned to the x axis and one to the y axis. Each point is treated as the end-point of a vector, plotted with an arrow; with the start point as the origin.</p>	<p>X: the variable to be plotted on the x axis. Y: the variable to be plotted on the y axis.</p>
<p>Vertical Bar: Plots a vertical bar graph for a specified variable, with the x axis showing variable values and the y axis showing frequency of occurrence for those values.</p>	<p>Response variable: the variable to be plotted.</p>
<p>Vertical Bar (parameterized): Plots a horizontal bar graph for a specified variable, categorised by another variable. The y axis shows variable values and the x axis shows the categories.</p>	<p>Category: the variable to be used for categorisation. Response: the variable to be plotted.</p>
<p>Vertical Box: Plots a single variable on the y axis, drawing a box centered on the variable's median value, with sides set at the 25th and 75th percentile. Also shown are whiskers above and below the box. The lower whisker is the lowest value still within 1.5 times the interquartile range of</p>	<p>Response variable: the variable to be plotted.</p>

Plot Type	Plot Variables and Parameters
the lower quartile. The upper whisker is the highest value still within 1.5 times the interquartile range of the upper quartile.	
Vertical Line: Plots a line graph for a specified variable, with the y axis showing variable values, and the x axis showing frequency of occurrence.	Response variable: the variable to be plotted.

Options

The options panel displays options specific to the currently selected plot type. To select a plot type, select its check box in the **Plots** panel.

Plot specific options

This section has the same title as the plot type selected (for example: Scatter, Series etc) and lists options tailored to that plot type. The table below lists and describes every plot option.

Plot Option	Description
Group	Chooses which variable to group plotted elements by, with the grouping shown on the plot by style and colour.
Legend Label	The text that appears next to the chart's legend.
Marker or Markers	When enabled, uses a larger marking for each data point.
Missing	Displays missing values on the chart.
No missing variable	Displays missing variables on the chart.
No missing group	Displays missing groups on the chart.
Frequency	Specifies a frequency variable in the input data set, determining how many times each observation counts. Only accepts integer values - decimal values will be truncated.
Break	Allows a break in data points from missing values when drawing a contiguous plot.
Data label	Specifies a variable to use as a label for each data point.
Statistic	Selects a statistic to use for the plot.
Response variable	Specifies a response variable.
Alpha	The percentile of the total observations that lie outside the plotted area.
Smooth	The smoothing parameter. Must be positive.
Category	Specifies a category variable.

Plot Option	Description
Type	Specifies the distribution type.
Clip	Clips an ellipse plot, if necessary, so that it doesn't extend past the plot axes.
Weight	Specifies a weighting variable in the input data set, determining the degree by which each observation counts. Accepts positive integer and decimal values.

Style Options

The list of style options presented in this section is tailored to the currently selected plot type. The table below lists and describes every option.

Style Option	Description
Transparency	Transparency for the plot, where 1.00 is completely transparent and 0.00 is completely opaque.
Fill	Select to fill the plot areas (for example, bars for a histogram) with a colour. If no fill style is chosen, a default colour will be allocated to the bars.
Fill style	select to enable fill colour and transparency options. If not selected, a default fill style will be chosen.
Fill colour	Click to choose the fill colour.
Fill transparency	Transparency for the fill, where 1.00 is completely transparent and 0.00 is completely opaque.
Outline	Draws a border around the plot.
Line style	Select to enable line colour and thickness options. If not selected, a default line style and colour will be chosen.
Line colour	Click to choose the plot line colour.
Line thickness	Click to choose the plot line thickness.

Axes Options

Some plot types allow a **second X axis** and or a **second Y axis** to be specified. If selected, each option flips the axis to a location opposite to its default position.

Style Options

The list of style options presented in this section is tailored to the currently selected plot type. The table below lists and describes every option.

Chart Builder Preferences

Allows you to set axes and basic charting options.

By Variables

Generates separate charts for each unique value in the specified variables. The charts can be viewed in the Chart Builder preview pane using the left and right arrow buttons, whereas Chart Viewer will generate thumbnails for each chart that can then be selected to view the full chart. To use the By Variables function, the selected variables must be sorted, so that all observations are grouped together by value.

X Axis, Y Axis, Second X Axis, and Second Y Axis

Configures the X Axis, Y Axis, Second X Axis, and Second Y Axis. There is a tab for each axis, each with the same options, as described below.

Display

Check boxes are used to choose the following axis features:

- **Label:** A label for the axis, placed centrally and on the side of the axis outside the plot area.
- **Line:** Displays an axis line. When a chart has a border, this line option may not be noticeable.
- **Ticks:** Markers for each numeric value shown adjacent to the axis.
- **Values:** Numeric values shown adjacent to the axis. These values are chosen automatically and cannot be edited.
- **Grid:** Lines extending across the plot area from each numeric value.

Axis

Allows you to specify the following axis features:

- **Type:** Choose from:
 - **Discrete:** Dataset values determine tick placement.
 - **Linear:** Ticks are spaced evenly based on minimum and maximum values.
 - **Log:** Logarithmic scale with base 10.
 - **Time:** If the axis variable is formatted as date or time in the input dataset, this option formats the axis as date or time, as applicable. If this option is not selected, the axis will display numerical values instead.
- **Tick placement:** When a logarithmic axis type is chosen, tick placement can either also be logarithmic at intervals of $\log(x)$ or $\log(y)$ (where x or y are integers); or linear, where ticks are spaced linearly and chosen automatically based on minimum and maximum values.

- **Log base:** Choose the log base for the tick placement.
- **Label:** Enter text for the axis label, which will be centered on the axis. There is no limit for the length of the label, although if the label exceeds the axis length it will be truncated at both ends.
- **Min:** Specifies a starting value for the axis. If this field is left blank, the smallest value from the plotted variable will be chosen.
- **Max:** Specifies an ending value for the axis. If this field is left blank, the largest value from the plotted variable will be chosen.
- **Offset min:** Specifies an offset for the start of the axis plot area away from the origin. Enter a fractional value equal to the proportion the offset is to be relative to the total axis range.
- **Offset max:** Specifies an offset for the end of the axis plot area away from the origin. Enter a fractional value equal to the proportion the offset is to be relative to the total axis range.
- **Integer:** Forces ticks to be at integer values.
- **Minor:** Inserts minor ticks between major ticks. Only applies when **Type** is set to **Log** or **Time**.
- **Reverse:** Reverses the axis, so the last observation in the dataset is plotted first (on the left hand end of the axis), and the first observation plotted last (on the right hand end of the axis).
- **Fit Policy.** Allows you to choose how large axis labels are fitted in the event of them overlapping. Choose from:
 - **None:** Leaves the labels overlapped.
 - **Rotate:** If labels overlap, this option will rotate them by 45 degrees.
 - **Rotate thin:** If labels overlap, this option first tries rotating them by 45 degrees. If labels still overlap after the rotation, alternate labels will be removed.
 - **Stagger:** If labels overlap, this option creates two rows for labels and splits the labels between the rows alternately.
 - **Stagger rotate:** If labels overlap, this option tries creating two rows for the labels, splitting labels between the rows alternately. If the labels still overlap, this option will then try rotating the labels by 45 degrees.
 - **Stagger thin:** If labels overlap, this option tries creating two rows for the labels, splitting labels between the rows alternately. If the labels still overlap, this option will then try removing alternate labels.
 - **Thin:** If labels overlap, this option removes alternate labels.

Style

Allows you to specify font and colour for axis labels and axis values.

Penalized B-Spline and Regression plots also have a **Show individual prediction limits** option, which displays 95% prediction limits for each point. The default label for individual prediction limits is **95% Prediction Limits**, although this can be changed by entering text in the **Individual prediction limits text** box.

Chart Builder Chart View

Displays the results of the **Chart Builder** block.

To open the Chart Builder Chart View, double-click on the **Chart Builder** block output. If there are multiple charts, specified by using the **By Variables** function in Chart Builder, thumbnails of these are displayed on the left hand side, with the full chart displayed on the right hand side when each thumbnail is selected.

Note:

Changes to a chart must be saved for them to appear in the Chart Viewer output.

Chart Editor

Double-click the **Resize Chart** button at the top right of the Chart Viewer to open the **Chart Editor** window.

To copy the chart to the clipboard, click **Copy Chart**. The size of the copied chart is determined by the size of the **Chart Editor** window when the chart is copied.

Delimited File Export block

Enables you to export a working dataset to a text file, specifying the field delimiters.

The **Delimited File Export** block requires a single dataset input, and creates a file containing the exported dataset.

Exporting a dataset is configured using the **Configure Delimited File Export** dialog box. To open the **Configure Delimited File Export** dialog box, double-click the **Delimited File Export** block.

Format

Delimiter

Specifies the character used to mark the boundary between variables in an observation of the exported dataset.

If the required delimiter is not listed, select `Other` and enter the character to use as the delimiter for exported dataset.

Write headers to first row

Specifies that the names of the variables in the input dataset are written as column headings to the first row of the exported file.

File Location

Specifies where the file containing the output dataset is located.

Workspace

Specifies the location of file containing the dataset is the current workspace.

Workspace datasets are only accessible when using the *Local Engine*.

External

Specifies the location of the file containing the dataset is on the file system accessible from the device running the Workflow Engine.

External files are accessible when using either a *Local Engine* or a *Remote Engine*.

Path

Specifies the path and file name for the file containing the output dataset. If you enter the **Path**:

- When exporting a dataset to a file in the Workspace, the root of the path is the Workspace. For example, to export a file to a project named `datasets`, the path is `/datasets/filename`.
- When exporting a dataset to an external location, the path is the absolute (full) path to the file location.

The export **Path** for the file is only valid with the Workflow Engine in use when the workflow is created. The path will need to be re-entered if the workflow is used with a different Workflow Engine.

If you do not know the path to the file, click **Browse** and navigate to the required dataset in the **Choose file** dialog box.

Excel Export block

Enables you to export a dataset from a Workflow to an Excel Workbook.

The **Excel Export** block requires a single dataset input, and creates an Excel workbook with a single worksheet.

Exporting a dataset to Microsoft Excel is configured using the **Configure Excel Export** dialog box. To open the **Configure Excel Export** dialog box, double-click the **Excel Export** block.

Format

Specifies the Excel Workbook version of the saved file containing the exported dataset. The format options are:

- Excel 2007-2013 Workbook (.xlsx)
- Excel 97-2003 Workbook (.xls)

Write headers to first row

Specifies that the names of the variables in the input dataset are written as column headings to the first row of the exported file.

File Location

Specifies where the file containing the output dataset is located.

Workspace

Specifies the location of file containing the dataset is the current workspace.

Workspace datasets are only accessible when using the *Local Engine*.

External

Specifies the location of the file containing the dataset is on the file system accessible from the device running the Workflow Engine.

External files are accessible when using either a *Local Engine* or a *Remote Engine*.

Path

Specifies the path and file name for the file containing the output dataset. If you enter the **Path**:

- When exporting a dataset to a file in the Workspace, the root of the path is the Workspace. For example, to export a file to a project named `datasets`, the path is `/datasets/filename`.
- When exporting a dataset to an external location, the path is the absolute (full) path to the file location.

The export **Path** for the file is only valid with the Workflow Engine in use when the workflow is created. The path will need to be re-entered if the workflow is used with a different Workflow Engine.

If you do not know the path to the file, click **Browse** and navigate to the required dataset in the **Choose file** dialog box.

Database Export block

Enables you to export a dataset from a Workflow to a specified database. This block has **Auto Run** disabled by default.

First steps

If any database references have already been created for Workflow, for example added from the Workflow **Settings** tab, then double-clicking on a new **Database Export** block will open the *Configure Database Export* [🔗](#) (page 344) tab.

If no database references have been created for Workflow, then double-clicking on a new **Database Export** block will take you straight to the **Add Database** wizard, which will guide you through creating a new database reference. For guidance on completing the wizard, see *Database References* [🔗](#) (page 165). The newly added database will then appear in the *Configure Database Export* [🔗](#) (page 344) tab.

Configure Database Export

Enables configuration of the export of a Workbench dataset to a target database.

Export Type

Specifies the type of export to be carried out. Choose from:

- **Append to existing table:** the dataset will be added to an existing **Target Table** within the **Target Database**.
- **Truncate existing table and insert:** Selecting this option will delete all existing data in the **Target Table** (but retain that table's structure) and replace it with the exported data.
- **Create new table:** Selecting this option will use the exported data to create a new **Target Table** within the **Target Database**. If the specified **Target Table** already exists, it will be deleted completely. If the specified **Target Table** does not exist, a new table will be created. When this option is selected, a warning is displayed. To suppress this warning, deselect **Warn when switching to destructive export types.**, or deselect this same option from Workbench **Preferences** under **Workflow** and then **Database Export**.

The **Target Table** and **Target Database** are specified in the **Database Parameters** panel.

Database Parameters

Target Database

Specifies the target database from a list of database references known to Workflow. To add a new database reference, click the **Create a new database** button (📄+), which launches the **Add Database** wizard. For guidance on completing the wizard, see [Database References](#) (page 165).

Target table

If **Export type** is set to **Append to existing table** or **Truncate existing table and insert**, displays a list of all tables within the specified **Target Database**. Choose the table that the dataset will be exported to.

If **Export type** is set to **Create new table**, allows you to specify a name for the new table. If the specified name already exists, that existing table will be deleted and replaced. If the specified name does not exist, a new table will be created.

Input Columns and Exported Columns

Input Columns lists variables from the input dataset which can be added as columns to the target database. **Exported Columns** lists variables from the input dataset that will be added to the target database. To move an item from one list to the other, double-click on it. Alternatively, click on the item to select it and use the appropriate single arrow button. A selection can also be a contiguous list chosen using **Shift** and click, or a non-contiguous list chosen using **Ctrl** and click. Use the double arrows to move all the items from one list to the other. Use the filter button on either list to limit the variables displayed, either by a simple text filter, or by a regular expression (enabled by selecting the **Regular expression** checkbox).

If you are appending or truncating, **Target Column** allows you to specify the column the variable will be exported to (from compatible columns) and **Target Type** displays the data type. If a column in the **Target database's Target table** matches the variable name and type, this will be automatically selected. If you are creating a new table, the **Target Column** and **Target Type** must be specified.

Tableau Export block

The **Tableau Export** block allows you to export a Workbench dataset to Tableau Data Extract file format (.tde) and, if required, uploaded to a Tableau server. The .tde file can be stored either within a Workbench workspace, or in a specified external folder.

Configure Tableau Export

Enables configuration of the export of a Workbench dataset to a Tableau Data Extract file format (.tde) and, if required, upload to a Tableau server. This block has auto-run enabled.

Opening the Configure Tableau Export dialog box

To open the **Configure Tableau Export** dialog box, double-click the **Tableau Export** block.

File Location

Choose from:

- **Workspace**, to use a Workbench workspace.
- **External**, to use an external folder.

Path

Type a filename in the box and click **Browse** to open either a Workspace browser or file system browser, depending on which option is selected above.

Click **OK** to export the file.

Server Upload

If **Perform Upload** is selected, when the block is run the Tableau file will be uploaded to a Tableau Server with specified details:

- **User name**
- **Password**
- **Site ID**

- **Hostname**
- **Project**
- **Datasource**

Workbench Preferences

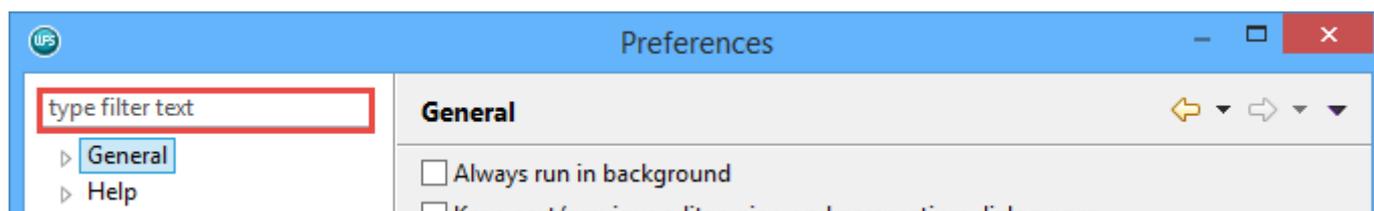
Enables you to configure how Workbench operates in the SAS Language Environment, the Workflow Environment, and generically.

Using Workbench Preferences

How to find and navigate Workbench Preferences.

There are many user settings, controls and defaults that you can control using the **Preferences** window. To open the window, click the **Window** menu and then click **Preferences**.

The **Preferences** window has a filter function that can help you find preference pages quickly. Once you have opened the window, you will see the filter at the top of the preferences list.



General preferences

The **General** page of the **Preferences** window contains controls that affect general aspects of Workbench.

Always Run in Background

Specifies that long running tasks to be run without displaying the **Executing** window. This allows you to carry on using Workbench for other tasks.

Show Heap Status

When selected, an indicator is displayed in Workbench showing information about usage of the current *Java heap* memory used by the Workbench interface. This indicator appears in the bottom of the window to the left of the server status.

To increase your *Java heap* storage, modify the `workbench.ini` in the `eclipse` folder of your installation, to, for example, `-vmargs -Xmx1024m`. This sets the maximum *Java heap* value of 1024 megabytes.

Workbench save interval (in minutes)

Specifies how often the state of the Workbench, including perspective layouts, is automatically saved to disk. Set to 0 (zero) to disable automatic saving.

Open Mode

Specifies the method used to open objects in Workbench.

- **double-click** – open an object with a double-click, select by single click.
- **Single click (Select on hover)** – Open an object with a single click, select by hovering the mouse cursor over the resource.
- **Single click (Open when using arrow keys)** – Open an object with a single click, select by hovering the mouse cursor over the resource. When you use the arrow keys to open an object with a single click, selecting a resource with the arrow keys will automatically open it in an editor.

Changing shortcut key preferences

Workbench provides shortcut key mappings for the most commonly-used commands. You can modify these mappings, or create new mappings.

To change a shortcut key mapping:

1. Click the **Window** menu and then click **Preferences**.
2. In the **Preferences** window, expand the **General** group and select **Keys**.
3. Either type the command name, for example *Cancel edits* in the filter, or select the required command in the list.
4. Select **Binding** and type the shortcut key, for example **Ctrl= 5**.
5. If no conflicts are reported, click **OK** to save the new mapping.

To delete a binding, select the required command and click **Unbind Command**.

Backing up Workbench preferences

Saving your current preferences to file to preserve preferences or to import into a different Workbench installation.

To save your preferences to file:

1. Click the **File** menu and then click **Export**.
2. In the **Export** dialog, expand the **General** group and click **Preferences**.

3. Click **Next** and in the **Export Preferences** page, select the required preferences and enter a preference file name.

If you are updating an existing file, you can select **Overwrite existing files without warning** to avoid warning dialogs during export.

4. Click **Finish** to create the backup file.

Importing Workbench preferences

Importing saved preferences from file to a Workbench installation.

To import preferences from file:

1. Click the **File** menu and then click **Import**.
2. In the **Import** dialog, expand the **General** group and click **Preferences**.
3. Click **Next** and in the **Import Preferences** page, select the required preferences file and choose the preference group to import. and enter a preference filename.
4. Click **Finish** to import the preferences.

Workflow Environment preferences

The Workflow Environment preferences in the **WPS** section of the **Preferences** dialog box specify defaults and preferences when using the Workflow Environment perspective.

Data panel ↗	350
Specifies the default level for classifying variables and whether a subset of the dataset is used for profiling or growing decision trees.	
Data Profiler panel ↗	350
Specifies the default settings for profiling datasets in the Data Profiler view.	
Workflow panel ↗	351
Specifies default settings for running a Workflow and where intermediate datasets are stored.	

Data panel

Specifies the default level for classifying variables and whether a subset of the dataset is used for profiling or growing decision trees.

Classification threshold

Specifies the number of unique values in a variable above which the variable classification is Continuous. If the number of unique values is below the threshold, the classification is Categorical for character type variables or Discrete for numeric type variables. The classification is displayed in the **Summary View** tab of the **Data Profiler** view.

Limit cells processed to

Specifies the maximum number of cells processed when creating statistics in the **Data Profiler** view or **Decision Tree Editor** view. If left empty, all cells in the dataset are used when creating statistical information or growing a decision tree.

Reducing the number of processed cells in a dataset can help reduce the time taken to view information about a dataset or grow a decision tree.

The restriction can be applied to one or both views enabling you to, for example, use a subset of the dataset in the **Data Profiler** view and use the whole dataset to grow a decision tree.

Data Profiler panel

Specifies the default settings for profiling datasets in the **Data Profiler** view.

Summary View preferences

Enables you to specify whether the frequency graph for variables is displayed in the Summary View panel. In a dataset with a large number of variables displaying the frequency graph may reduce the responsiveness of the **Data Profiler** view.

To display the graph, select **Show Frequency Graph**. To hide the graph, clear **Show Frequency Graph**.

Predictive power preferences

Enables you to specify the number of variables displayed in the Entropy chart of the Predictive Power panel. Variables displayed are those calculated to have the highest Entropy Variance. If the number of displayed variables is too small, the graph may only display variables with a value variance that is too random to enable a good selection of dependent variables.

Workflow panel

Specifies default settings for running a Workflow and where intermediate datasets are stored.

Execution

Specifies whether the Workflow is run automatically or manually. To run the Workflow manually, on the **File** menu, click **Run Workflow**. This option will only run blocks that have changed. If you need to regenerate all working datasets in the Workflow, on the **File** menu click **Force Run Workflow**.

Auto Run Workflow

Select to run a Workflow when a new block is added, or an existing block updated. If left clear the Workflow needs to be run manually when modified.

Temporary Resources

Specifies whether working datasets and temporary resources used by the Workflow are saved to disk, and where the resources are located.

Persist to disk

Select to save working datasets in a temporary location on disk. If left clear, working datasets are stored in memory while the Workflow is open; this may lead to a Workflow failing if the datasets produced are large.

Location

Specifies the folder where working datasets and other temporary resources are located. By default this folder is in your user profile. To change the location, click **Browse**, and select an alternative folder.

Time-To-Live

Specifies the number of minutes, hours or days the temporary resources are kept for before being deleted. The default value is 30 days and the maximum value is 180 days.

Clear temporary resources

Click to remove all working datasets and other temporary resources saved to disk.

Binning panel

Specifies the default number of bins created, and the maximum number of bins that can be created.

Default bin count

Specifies the number of bins to be used in equal-width or equal-height binning.

Winsorrate

Specifies the minimum and maximum percentile value for winsorized binning. All values below or above this percentile of observations are set to the lower or upper observational values at this point.

For example, if you specify 0.05; prior to the data being split into bins, all values below the 5th percentile are set to the lower observational value at the 5th percentile, and all values above the 95th percentile are set to the upper observational value at the 95th percentile.

Optimal binning

Specifies the default settings used to determine how a dataset is split into optimal groups for further analysis.

Measure

Specifies the split measure used when performing optimal binning. for more information about these measures, see Predictive power criteria [↗](#) (page 102).

Chi-squared

Specifies that Pearson's Chi-Squared Test is used to measure the predictive power of variables by measuring the likelihood that the value of the dependent variable is related to the value of the independent variable.

Gini Variance

Specifies that Gini Variance is used to measure the predictive power of variables by measuring the strength of association between variables.

Entropy Variance

Specifies that Entropy Variance is used to measure the predictive power of variables by measuring how well an independent variable value can predict the dependent variable value.

Information Value

Specifies that Information Value is used to measure the predictive power of variables by measuring the likelihood that the dependent variable value is related to the value of the independent variable.

Information Value

Specifies the default settings for the Information Value measure.

Minimum information value

Specifies the minimum value for the information variable. If the information value of a variable falls below this limit it is not added to the bin.

Maximum change when merging

Specifies the maximum change allowed in the predictive power when optimally merging bins.

WoE adjust

Specifies the adjustment applied in weight of evidence calculations to avoid invalid results for pure inputs.

Entropy Variance

Specifies the default settings for the Entropy Variance measure.

Minimum value

Specifies the minimum value for the Entropy Variance. If the Entropy Variance of a variable falls below this limit it is not added to the bin.

Maximum change when merging

Specifies the maximum change allowed in the predictive power when optimally merging bins.

Gini Variance

Specifies the default settings for the Gini Variance measure.

Minimum value

Specifies the minimum value for the Gini Variance. If the Gini Variance of a variable falls below this limit it is not added to the bin.

Maximum change when merging

Specifies the maximum change allowed in the predictive power when optimally merging bins.

Chi-squared

Specifies the default settings for the Chi-squared measure.

Minimum value

Specifies the minimum value for the Chi-squared measure. If the Chi-squared measure of a variable falls below this limit it is not added to the bin.

Maximum change when merging

Specifies the maximum change allowed in the predictive power when optimally merging bins.

Initial number of bins

Specifies the number of bins into which data is merged before the optimal binning begins.

For example, if you specify 50, then before starting the optimal binning process, a numerical variable is equally binned into 50 bins; if the number of categories in a variable is greater than 50, similar categories are merged to create 50 bins.

Maximum number of bins

Specifies the maximum number of bins to use when optimally binning variables.

Exclude Missing

Excludes missing values.

Merge a bin with missing values

Specifies that missing values are merged into the bin containing the most similar values. If left clear, a separate bin is created for missing values.

Monotonic Weight of Evidence

Specifies that the Weight of Evidence value for ordered input variables is either monotonically increasing or monotonically decreasing.

Minimum bin size

Specifies the minimum number of observations for bin as either a proportion of the input dataset or an absolute value.

Count

Specifies the absolute minimum number of observations a bin can contain.

Percent

Specifies the minimum size of each bin as a proportion of the number of input observations.

Chart panel

Specifies preferences for the **Correlation Analysis** view of the **Data Profiler** view.

Scatter plot observations threshold

Specifies the limit above which a heat map of values is shown in the **Scatter Plot** panel of the **Correlation Analysis** view.

Correlation Matrix

Specifies how items are displayed in the **Correlation Coefficient Matrix** chart in the **Correlation Analysis** view.

Vary matrix item size by coefficient magnitude

Select to display the size of the items in the **Correlation Coefficient Matrix** chart relative to the coefficient of the two variables being compared. Clear to display all items in the chart at the same size.

Matrix item shape

Specifies the shape of the item on the **Correlation Coefficient Matrix** chart, either `Circle` or `Square`

Decision Tree panel

Specifies the default tree growth preferences for a decision tree.

The **Decision Tree** block uses the information specified in this panel when growing a decision tree using the `DECISIONTREE` procedure in WPS Analytics. For more information see the *DECISIONTREE procedure* in the *WPS Reference for Language Elements*

Default Growth

Specifies the default growth algorithm for a new decision tree, one of:

- `C4.5`. Specifies the C4.5 algorithm is used for generating a classification decision trees.
- `CART`. Specifies the CART algorithm is used for generating classification and regression decision trees.
- `BRT`. Specifies the BRT algorithm is used for generating binary response decision trees.

C4.5

Specifies the default preferences for a decision tree grown using the C4.5 method. A decision tree grown with this method always uses the Entropy Variance to determine when to split a node in the tree.

Maximum depth

Specifies the maximum depth of the decision tree.

Minimum node size (%)

Specifies the minimum number of observations in a decision tree node, as a proportion of the input dataset.

Merge categories

Specifies that discrete independent (input) variables are grouped together to optimize the dependent (target) variable value used for splitting.

Prune

Select to specify that nodes which do not significantly improve the predictive accuracy are removed from the decision tree to reduce tree complexity.

Prune confidence level

Specifies the confidence level for pruning, expressed as a percentage.

Exclude missings

Specifies that observations containing missing values are excluded when determining the best split at a node.

CART

Specifies the default preferences for a decision tree grown using the CART method.

Criterion

Specifies the criterion used to split nodes in the decision tree. The supported criteria are:

Gini

Specifies that Gini Impurity is used to measure the predictive power of variables.

Ordered twoing

Specifies that the Ordered Twoing Index is used to measure the predictive power of variables.

Twoing

Specifies that the Twoing Index is used to measure the predictive power of variables.

Prune

Select to specify that nodes which do not significantly improve the predictive accuracy are removed from the decision tree to reduce tree complexity.

Pruning method

Specifies the method to be used when pruning a decision tree. The supported methods are:

Holdout

Specifies that the input dataset is randomly divided into a test dataset containing one third of the input dataset and a training dataset containing the balance. The output decision tree is created using the training dataset, then pruned using risk estimation calculations on the test dataset.

Cross validation

Specifies that the input dataset is divided as evenly as possible into ten randomly-selected groups, and the analysis repeated ten times. The analysis uses a different group as test dataset each time, with the remaining groups used as training data.

Risk estimates are calculated for each group and then averaged across all groups. The averaged risk values are used to prune the final tree built from the entire dataset.

Maximum depth

Specifies the maximum depth of the decision tree.

Minimum node size (%)

Specifies the minimum number of observations in a decision tree node, as a proportion of the input dataset.

Minimum improvement

Specifies the minimum improvement in impurity required to split decision tree node.

Exclude missings

Specifies that observations containing missing values are excluded when determining the best split at a node.

BRT

Specifies the default preferences for a decision tree grown using the BRT method.

Criterion

Specifies the criterion used to split nodes in the decision tree. The supported criteria are:

Chi-squared

Specifies that Pearson's Chi-Squared statistic is used to measure the predictive power of variables.

Entropy variance

Specifies that Entropy Variance is used to measure the predictive power of variables.

Gini variance

Specifies that Gini Variance is used to measure the predictive power of variables by measuring the strength of association between variables.

Information value

Specifies that Information Value is used to measure the predictive power of variables.

Maximum depth

Specifies the maximum depth of the decision tree.

Select minimum node size by ratio

When selected enables you to specify the minimum number of observations in a node as a proportion of the input dataset. When cleared, enables you to specify the absolute minimum number of observations in a node.

Minimum node size (%)

Specifies the minimum number of observations in a decision tree node, as a proportion of the input dataset.

Minimum node size

When selected enables you to specify the minimum split size of a decision tree node as a proportion of the input dataset. When cleared, enables you to specify the absolute minimum split size of a decision tree node .

Select minimum split size by ratio

When selected enables you to specify the minimum number of observations for the node to be split as a proportion of the input dataset. When cleared, enables you to specify the absolute minimum number of observations in a node.

Minimum split size (%)

Specifies the minimum number of observations a decision tree node must contain for the node to be split, as a proportion of the input dataset.

Minimum split size

Specifies the absolute minimum number of observations a decision tree node must contain for the node to be split.

Allow same variable split

Specifies that a variable can be used more than once to split a decision tree node.

Open left

For a continuous variable, when selected, specifies that the minimum value in the node containing the very lowest values is $-\infty$ (minus infinity). When clear, the minimum value is the lowest value for the variable in the input dataset.

Open right

For a continuous variable, when selected, specifies that the maximum value in the node containing the very largest values is ∞ (infinity). When clear, the maximum value is the largest value for the variable in the input dataset.

Merge missing bin

Specifies that missing values are considered a separate valid category when binning data.

Monotonic Weight of Evidence

Specifies that the Weight of Evidence value for ordered input variables is either monotonically increasing or monotonically decreasing.

Exclude missings

Specifies that observations containing missing values are excluded when determining the best split at a node.

Initial number of bins

Specifies the number of bins available when binning variables.

Maximum number of optimal bins

Specifies the maximum number of bins to use when optimally binning variables.

Weight of Evidence adjustment

Specifies the adjustment applied to weight of evidence calculations to avoid invalid results for pure inputs.

Maximum change in predictive power

Specifies the maximum change allowed in the predictive power when optimally merging bins.

Minimum predictive power for split

Specifies the minimum predictive power required for a decision tree node to be split.

Using Git with Workbench

Git support can be installed into Workbench, enabling a Git Staging view and Git perspective, which allow management of Git files from within Workbench.

Installing Workbench Git support

Setting up Workbench to work with Git.

Git support can be installed into any version of Workbench. Before you start, obtain the URL of the eGit update site (at the time of writing, this is <https://download.eclipse.org/egit/updates>).

To install Git into Workbench:

1. Open Workbench.
2. From the **Help** drop-down menu, select **Install New Software**.
3. In **Work with**, enter the URL of the eGit update site and press **Return**.
4. Expand the **Git integration for Eclipse** folder and then select **Git integration for Eclipse**.
5. Click **Next**.

The progress bar shows **Calculating requirements and dependencies**. After a short delay, you will be asked to **Review the items to be installed**.

6. Click **Next**.
7. Select **I accept the terms of the license agreements** and click **Finish**.

An **Installing Software** progress window is displayed.

8. At the **Do you trust these certificates?** page, select the certificate at the top of the list and click **OK**.

The **Installing Software** progress window continues.

9. When asked to restart Workbench, click **Yes**.

Workbench restarts.

Workbench Git support is now installed.

Using Workbench with Git

Workbench provides two ways to integrate with Git: the Git Staging view and the Git perspective.

Opening the Workbench Git Staging view

The Workbench Git Staging view offers basic Git integration, such as simple commit operations.

To use the Git Staging view, you must have installed Git Workbench support (see *Installing Workbench Git support* [↗](#) (page 359)).

To display the Git Staging view:

1. From Workbench, select the **Window** menu, then **Show View** and click **Other**.

A **Show View** window is displayed.

2. From the **Git** folder, select **Git Staging** and click **OK**.

The Git Staging view is displayed.

Opening the Workbench Git perspective

The Workbench Git perspective offers full Git support within Workbench.

To use the Git perspective, you must have installed Git Workbench support (see *Installing Workbench Git support* [↗](#) (page 359)).

To display the Git perspective:

1. From Workbench, select the **Window** menu, then **Perspective** and click **Other**.
2. From the list of perspectives, click **Git** and click **OK**.

The Workbench Git perspective is displayed.

Configuration files

Configuration files are files incorporating system options that control the initial WPS environment. Configuration files are used during WPS initialisation when WPS is run from the command line, and also when the Workbench is being used.

You can open a configuration file as a basic text file in order to add, remove or change system options to suit your needs.

There is a base configuration file that is included with the WPS installation, called `wps.cfg` and it is located in the directory where WPS is installed. It is recommended that this file be left untouched and one of the override mechanisms below be used if any modifications need to be made.

Note:

A number of configuration files can be processed during WPS initialisation, each of which may set some of the same options. If an option is set in more than one place, the last one processed will have precedence. Options set on the command line, or options set in the **Startup Options** configuration panel for a server, will override any settings that are set in the configuration files.

WPS initialisation procedure for Windows

When WPS is invoked, the following initialisation procedure occurs:

1. The `WPS_SYS_CONFIG` operating system environment variable is evaluated and, if it exists, the specified configuration file is processed. The processing of this file does not affect whether the default configuration files are loaded.
2. Any files specified via the `-config` option on the command line or in the startup options for a server are processed.

If any configuration files are specified via the `-config` option, the default configuration files are *not* loaded, and the `WPS_USER_CONFIG` is processed instead.

Take care when overriding the default configuration file processing, specifically the loading of the `wps.cfg` file. There are many required options in the base `wps.cfg` file. If you use the `-config` option, it is recommended that you either take a copy of the base `wps.cfg` file and modify it to suit, or that you make sure to include the base `wps.cfg` file in your custom configuration file and then override any settings that you need, for example:

```
-set wpsHOME 'somewhere'  
-config !wpsHOME/wps.cfg  
/* And now set any options you want */
```

3. The default configuration files are processed in the following order:
 - a. `wps.cfg` in the WPS installation directory
 - b. `My Documents/My WPS Files/.wps.cfg`
 - c. `My Documents/My WPS Files/wps.cfg`
 - d. `My Documents/My WPS Files/.wpsv3.cfg`
 - e. `My Documents/My WPS Files/wpsv3.cfg`
 - f. `wps.cfg` in the current directory
 - g. `wpsv3.cfg` in the current directory
4. The `WPS_USER_CONFIG` environment variable is evaluated and, if it exists, the specified configuration file is processed.
5. The `WPS_OPTIONS` environment variable is evaluated and, if it exists, any options specified are processed. The syntax of this environment variable is the same as if the options are specified on the command line.
6. Any options specified on the command line are processed.

WPS initialisation procedure for UNIX platforms

When WPS is invoked, the following initialisation procedure occurs:

1. Any files specified via the `-config` option on the command line or in the startup options for the server are processed.

If any configuration files are loaded in this way, the default configuration files are *not* loaded.

2. The `WPS_OPTIONS` and `WPS_V3OPTIONS` environment variables are evaluated and any contained `-config` options are processed.

If any configuration files are loaded in this way, the default configuration files are *not* loaded.

Note:

This step is skipped if the `-NOCONFIG` option is specified on the command line.

3. The `WPS_CONFIG` and `WPS_V3CONFIG` environment variables are evaluated and, if they exist, the specified configuration files are processed.

If any configuration files are loaded in this way, the default configuration files are *not* loaded.

Note:

This step is skipped if the `-NOCONFIG` option is specified on the command line.

4. If no configuration files have yet been loaded, the default configuration files are loaded in the following order:
 - a. `wps.cfg` in the WPS installation directory
 - b. `wps_local.cfg` in the WPS installation directory. This file can be edited to provide site-specific overrides of various options.
 - c. `.wps.cfg` in your home directory
 - d. `wps.cfg` in your home directory
 - e. `.wpsv3.cfg` in your home directory
 - f. `wpsv3.cfg` in your home directory
 - g. `wps.cfg` in the current directory
 - h. `wpsv3.cfg` in the current directory
5. The `WPS_OPTIONS` environment variable is evaluated and, if it exists, any options specified are processed. The syntax of this environment variable is the same as if the options are specified on the command line.
6. The `WPS_V3OPTIONS` environment variable is evaluated and, if it exists, any options specified are processed. The syntax of this environment variable is the same as if the options are specified on the command line.
7. Any options specified on the command line are processed.

There are also "restricted" configuration files that will *always* be processed. Any options in these files cannot be changed by any other method. The files are based on the user's `userid` and `groupid`. For example, if the user is `<user>` and the group is `<group>`, the following configuration files will be processed:

1. `misc/rstropts/rwps.cfg` in the WPS installation directory
2. `misc/rstropts/groups/<group>_rwps.cfg` in the WPS installation directory
3. `misc/rstropts/users/<user>_rwps.cfg` in the WPS installation directory

Get information about the startup procedure

It is possible to check which configuration files have been processed by checking the `CONFIG` system option. You can do this by invoking a `PROC OPTIONS` statement as follows:

```
PROC OPTIONS OPTION=CONFIG;
RUN;
```

You can also set the `VERBOSE` system option on the command line, or in the startup options for the server, and WPS will then generate information in the normal log output indicating which configuration files were processed and which system options were set.

AutoExec file

An *AutoExec* file contains SAS language statements or a program that is automatically executed when a Processing Engine is started, or restarted from Workbench. The file can be used for a number of purposes, such as setting up common `LIBNAMES` so that they do not need to be added to every program that is run.

Windows

The WPS server automatically runs a file called `autoexec.sas`. The WPS server searches for the `autoexec.sas` file in the following directories, in the order shown:

1. The current directory
2. `My Documents` in your user profile
3. Paths specified in the `PATH` environment variable
4. The root directory of the drive, for example `C:\`
5. The WPS installation directory.

Other operating systems

The WPS server automatically runs a file called `autoexec.sas`. The WPS server searches for the `autoexec.sas` file in the following directories, in the order shown:

1. The current directory
2. Your user home (`$HOME`) directory
3. The WPS installation directory.

Specifying a different file

To automatically run a different file, or a file not in the searched directory hierarchy, set the `AUTOEXEC` system option to point to the file (see [Configuration files](#) (page 361)).

You can also set the `AUTOEXEC` system option through Workbench:

1. In the **Link Explorer** or **Workflow Link Explorer** view, right-click the server name and click **Properties** from the shortcut menu.
2. In the **Properties** list, double-click **Startup**, click **System Options**.
3. On the **System Options** panel, click **Add**.
4. In the **Name** field, enter `AUTOEXEC`, and in the **Value** field enter the path to the required configuration file and click **OK**.
5. Click **OK** on the **Properties** dialog and you will be prompted to restart the server.

WPS Analytics tips and tricks

Tips and tricks for general setup, file management, running SAS language programs and datasets.

General Setup

- Use WPS Link connect to a remote server solution and run SAS language programs (see *Connecting to a remote Processing Engine* [↗](#) (page 29)).
- You can change the Workbench font, background appearance, or change the shortcut key bindings (see *Using Workbench Preferences* [↗](#) (page 347)).
- You can export and import preferences (including templates) between workspaces.
- You can use configuration files to control the system options used in the WPS Analytics environment (see *Configuration files* [↗](#) (page 361)).
- You can use an AutoExec file to set up common `LIBNAME` statements and use them any program running on the WPS server (see *AutoExec file* [↗](#) (page 364)).
- You can define multiple servers the same machine and distribute programs between those servers (see *Defining a new Processing Engine* [↗](#) (page 32)).

File Management

- To display or create a different group of projects, you can switch to a different workspace (see *Switching to a different workspace* [↗](#) (page 62)).
- You can restore a file deleted in the **Project Explorer** from the project's local history (see *Restoring from local history* [↗](#) (page 61)).
- You can use the **Project Explorer** to compare and merge files (see *Comparing and merging multiple files* [↗](#) (page 58)).
- You can create directory shortcuts in the **File Explorer** to access files on the host file system (see *Creating a new remote host connection* [↗](#) (page 30)).

Running SAS language programs

- You can specify code that is to be executed automatically before and after each program (see *WPS Code Injection* [↗](#) (page 116)).
- You can create templates containing code snippets that can be re-used (see *Entering WPS code via templates* [↗](#) (page 116)).
- You can run part of a program from the Workbench (see *Running a selected section of a program* [↗](#) (page 118)).
- You can control whether or not the ODS destinations are to be controlled automatically, and, if so, which specific types of output are to be used (see *Automatically manage ODS destinations* [↗](#) (page 146)).

- You can set a permanent location for the `WORK` library to preserve datasets when Workbench is closed (see *Setting the WORK location* [↗](#) (page 123)).

Datasets

- You can edit datasets from within the Workbench, using in-place context-aware editing and column re-ordering.
- You can export datasets as delimited and fixed width text, and also as Microsoft Excel spreadsheets.

WPS Analytics

troubleshooting

A list of potential problems and how to resolve them.

Why are the icons greyed out so that I cannot run any programs?

The icons will be greyed out if:

- The server on which you are running programs is not licensed or has been uninstalled. See *WPS Processing Engines* [↗](#) (page 27) for more information.
- The name of the program file open in the **Editor** view does not use the `.sas` or `.wps` file name extension.
- The focus of the mouse is in a view other than the **Editor** view.

Why do I see a dialog saying that there is no default server on which to submit the program?

Usually because the default WPS server has been removed. If you have other WPS servers defined, select one of those as the new default (see *Default Processing Engine* [↗](#) (page 34)). Alternatively, you can restore the WPS server that was removed (see *Defining a new Processing Engine* [↗](#) (page 32)) and set it as the default.

Why are the characters in my files not being displayed correctly?

This is likely to be an encoding issue. You should follow all the steps in *Processing Engine LOCALE and ENCODING settings* [↗](#) (page 36) and *General text file encoding* [↗](#) (page 37), and restart Workbench. If you have imported the files that are displayed incorrectly, then you may need to re-import them.

Why can't I start the Workbench?

If the Workbench usually starts, but then fails, you may have run out of disk space, causing one or more files in the workspace `.metadata` directory to become corrupted. Rename the `.metadata` folder in the corrupt workspace directory to something else, restart WPS, and select to the workspace again. The **Project Explorer** contains no entries, but the project content and location are unaffected. You can reimport any existing projects as required. See *Importing an existing project* [↗](#) (page 11) for more information.

How do I increase my Java heap storage in the event of Java Runtime errors?

You can set the `-Xmx` and `-Xms` Java options to change the amount of heap memory available to the Java Runtime. See *Show Heap Status* [↗](#) (page 347) for more information.

Making technical support requests

Technical support procedures for WPS, including guidance for submitting log files.

How you access technical support for your WPS software depends on how you purchased your software. If you made a standard purchase of WPS software you have an annual subscription licence that entitles you to unlimited free upgrades throughout the twelve months, and free email support via support@worldprogramming.com.

Larger customers may have additional support arrangements in place. In this case, the handling of technical support requests may be carried out via one or more *super users* in your organisation; you need to be aware who the super users are to progress support issues.

Due to the complex nature of the SAS language, extended interaction between World Programming and yourselves may be required to identify and resolve the issue. Resolutions to problems may include suggested workarounds or may require you to update your WPS software installation.

WPS server log files

For certain types of issue, you may be requested to provide the log file. If the log is open in Workbench, right-click on the view and click **Save as** from the shortcut menu. If the log is not open, the `.log` file is stored in the `.metadata` directory in the active workspace.

Because `.metadata` and `.log` start with a period, they will normally be hidden and you will need to ensure you can see these types of file.

Legal Notices

(c) 2023 World Programming

This information is confidential and subject to copyright. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system.

Trademarks

WPS and World Programming are registered trademarks or trademarks of World Programming Limited in the European Union and other countries. (r) or ® indicates a Community trademark.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

All other trademarks are the property of their respective owner.

General Notices

World Programming Limited is not associated in any way with the SAS Institute.

WPS is not the SAS System.

The phrases "SAS", "SAS language", and "language of SAS" used in this document are used to refer to the computer programming language often referred to in any of these ways.

The phrases "program", "SAS program", and "SAS language program" used in this document are used to refer to programs written in the SAS language. These may also be referred to as "scripts", "SAS scripts", or "SAS language scripts".

The phrases "IML", "IML language", "IML syntax", "Interactive Matrix Language", and "language of IML" used in this document are used to refer to the computer programming language often referred to in any of these ways.

WPS includes software developed by third parties. More information can be found in the THANKS or acknowledgments.txt file included in the WPS installation.