This book has been updated for **KNIME 2.10**.

For information regarding permissions and sales, write to:

KNIME Press
Technoparkstr. 1
8005 Zurich
Switzerland

knimepress@knime.com

ISBN: 978-3-033-02850-0

Table of Contents

Foreword

Predictive analytics and data mining are becoming mainstream applications, fueling data-driven insight across many industry verticals. However, in order to continuously improve analytical processes it is crucial to be able to integrate heterogeneous data sources with tools from various origins. In addition, it is equally important to be able to uniformly deploy the results in operational systems and reuse models across applications and processes.

To address the challenges that users face in the end-to-end processing of complex data sets, we need a comprehensive platform to perform data extraction, pre-processing, statistical analysis and visualization. In this context, open source solutions offer the additional advantage that it is often easier to integrate legacy tools since the underlying code base is open. Therefore, KNIME is in a unique position to facilitate cross-platform, multi-vendor solutions which ultimately bring numerous benefits to the analytics industry, fostering common processes, agile deployment and exchange of models between applications. In support of its vision for open standards in the analytics industry, KNIME is also a member of the Data Mining Group (DMG) which develops the Predictive Model Markup Language (PMML), the de-facto standard for model exchange across commercial and open source data mining tools.

I predict that, as you read through this book and become an Expert in KNIME, you will find that your data mining solutions will not only follow a standards-based approach but also foster reuse of knowledge among all constituents involved in the analytics process, from data extraction, sophisticated statistical analysis to real-time business process integration.

As the first book for entry level users of KNIME, this book breaks ground with a comprehensive introduction which guides the reader through the multitude of analysis nodes, algorithms and configuration options. Supplemented with many examples and screen shots, it will make you productive with KNIME in no time.

Michael Zeller (CEO Zementis, Inc., PhD)

Acknowledgements

First of all I would like to thank the whole KNIME Team for their patience in dealing with me and my infinite questions.

Among all others in the KNIME Team I would like to specifically thank Peter Ohl for having reviewed this book in order to find any possible aspects that were not compatible with KNIME best practice.

I would also like to thank Bernd Wiswedel for all his help in the reporting section, Thomas Gabriel for his precious advices in the database jungle, and Dominik Morent for the answers about data modeling implementations.

I would like to thank Meta Brown for encouraging me in the first steps of developing the embryonic idea of writing this book.

Many thanks finally go to Heather Fyson for reviewing the book's English.

Chapter 1. Introduction

1.1. Purpose and structure of this book

We live in the age of data! Every purchase we make is dutifully recorded; every money transaction is carefully registered; every web click ends up in a web click archive. Nowadays everything carries an RFID chip and can record data. We have data available like never before. What can we do with all these data? Can we make some sense out of it? Can we use it to learn something useful and profitable? We need a tool, a surgical knife that can empower us to cut deeper and deeper into our data, to look at it from many different perspectives, to represent its underlying structure.

Let's suppose then that we have this huge amount of data already available, waiting to be dissected. What are the options for a professional to enter the world of Business Intelligence (BI) and data analytics? The options available are of course multiple and growing rapidly. If our professional does not control an excessive budget he could turn to the world of open source software. Open source software, however, is more than a money driven choice. In many cases it represents a software philosophy for resource sharing that many professionals would like to support.

Inside the open source software world we can find a few data analysis and BI tools. KNIME software represents an easy choice for the non-initiated professional. It does not require learning a specific script and it offers a graphical way to implement and document analysis procedures. In addition - and this is not a secondary advantage - KNIME can work as an integration platform into which many other BI and data analysis tools can be plugged. It is then not only possible but even easy to analyze data with KNIME and to build dashboards on the same processed data with a different BI tool.

Even though KNIME is very simple and intuitive to use, any beginner would profit from an accelerated orientation through all of KNIME's nodes, categories, and settings. This book represents the beginner's luck, because it is aimed to help any beginner to gear up his/her learning process. This book is not meant to be an exhaustive guide to the whole KNIME software. It does not cover implementations under the KNIME Server, which is not open source, or topics which are considered advanced. Flow Variables, for example, and implementations of database SQL queries are not discussed here.

The book is divided into six chapters. The first chapter covers the basic concepts of KNIME, while chapter two takes the reader by the hand into the implementation of a very first analysis procedure. In the third chapter we investigate data analysis in a more in depth manner. The third chapter indeed explains how to perform some data visualization, in terms of the nodes and processing flow. Chapter four is dedicated to data modeling. It covers a few demonstrative approaches to data modeling, from naïve Bayesian networks to decision trees and artificial neural networks. Finally, chapters five and six are dedicated to reporting. Usually the results of an investigation based on data visualization or, in a later phase, on data modeling have to be shown at some point to colleagues, management, directors, customers, or external workers. Reporting represents a very

important phase at the end of the data analysis process. Chapter five shows how to prepare the data to export into a report while chapter six shows how to build the report itself.

Each chapter guides the reader through a data manipulation or a data analysis process step by step. Each step is explained in details and offers some explanations about alternative employments of the current nodes. At the end of each chapter a number of exercises are proposed to the reader to test and perfect what he/she has learned so far.

Examples and exercises in this book have been implemented using KNIME 2.10. They should also work under subsequent KNIME versions, although there might be slight differences in their appearance. Updates and further discussions about the contents of this book can be found at the author's blog site http://dataminingreporting.com/blog.html.

1.2. The KNIME community

Web Links	
http://www.knime.org	This is the first place to look for information about KNIME products. The open source KNIME Analytics Platform can be downloaded here.
http://www.knime.org/learning-hub	This is the landing page to learn more specific KNIME functionalities. Learning material - as web sites, videos, webinars, courses, and more - is organized by topic, like text mining or chemistry, or basic KNIME functionalities, etc...
http://tech.knime.org/forum	In the www.knime.org site you can find a number of resources. What I find particularly useful is the KNIME Forum. Here you can ask questions about how to use KNIME or about how to extend KNIME with new nodes. Someone from the KNIME community answers always and quickly.
http://tech.knime.org/knime-labs	This site contains nodes still under development; i.e. the beta version of new nodes. You can already download them and use them, but they are not of product/release quality yet.
http://knime.org/supporters-0	This is the site where all contributing supporters (partners, providers, and sponsors) are listed.

Courses, Events, and Videos

KNIME User Training (Basic and Advanced)	KNIME periodically offers Basic and Advanced User Training Courses. To check for the next available date/place and to register, just go to the KNIME Training web site http://knime.org/trainings.
KNIME Webinars	A number of webinars are also available since May 2013 on specific topics, like chemistry nodes, text mining, integration with other analytics tools, and so on. To know about the next scheduled webinars, check the KNIME Training web page at http://knime.org/trainings.
KNIME Meetups and User Days	KNIME Meetups and KNIME User Days are held periodically all over the world. These are always good chances to learn more about KNIME, to get inspired about new data analytics projects, and to get to know other people from the KNIME Community (http://knime.org/about/events)
KNIME TV Channel on YouTube	KNIME has its own video channel on YouTube, named KNIME TV. There, a number of videos are available to learn more about many different topics and especially to get updated about the new features in the new KNIME releases (http://www.youtube.com/user/KNIMETV)

Books

KNIME Platform	For the advanced use: Rosaria SIlipo, Mike Mazanetz, "The KNIME Cookbook: Recipes for the Advanced User" (http://www.knime.org/knimepress/the-knime-cookbook) For a general summary: Gabor Bakos, "KNIME Essentials" (http://www.packtpub.com/knime-essentials/book)
Reporting Suite	The KNIME Reporting Suite is based on BIRT, another open source tool for reporting. Here is a basic guide on how to use BIRT: *D. Peh, N. Hague, J. Tatchell, "BIRT. A field Guide to Reporting.", Addison-Wesley, 2008*
Data Analysis and KNIME	To apply KNIME to the most common problems in data analysis and data mining: *Berthold M.R., Borgelt C., Höppner F., Klawonn F.,"Guide to intelligent data analysis", Springer 2010.*

1.3. Download and install the KNIME Analytics Platform

To start playing with KNIME, first, you need to download it to your computer.
There are two available versions of KNIME:

- the open source KNIME Analytics Platform, which can be downloaded free of charge at www.knime.org under the GPL version 3 license
- the Enterprise KNIME server, which can be purchased at http://knime.org/enterprise-server

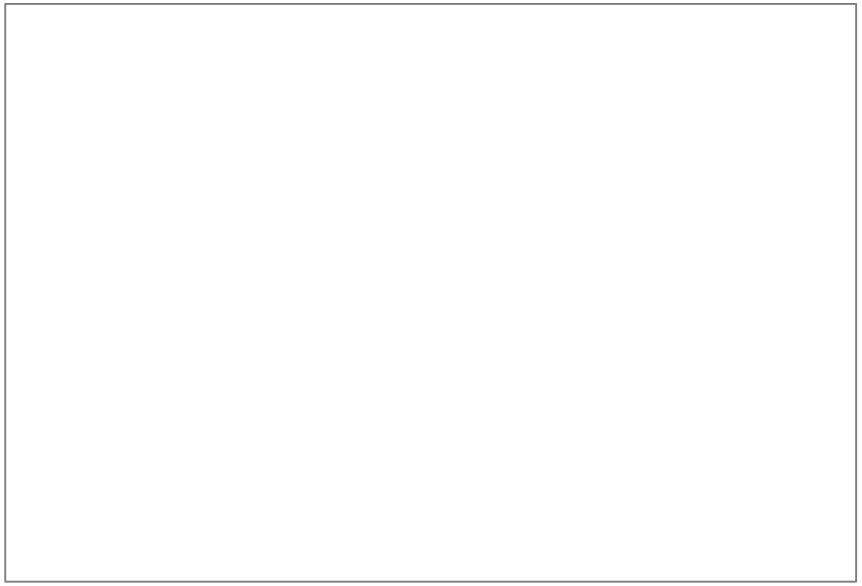
The functionalities of the two versions are very similar. The Enterprise version includes a few more centralized operations and collaboration tools that better fit a large enterprise environment. In this book we work with KNIME Analytics Platform (open source) version 2.10.

Download KNIME Analytics Platform

- Go to www.knime.org
- In the lower part of the main page, click “Download Now”
- If you wish, provide a little information about yourself (that is appreciated), otherwise select “Download Now”
- Choose the version that suits your environment (Windows/Mac/Linux, 32 bit/64 bit)
- Accept the terms and conditions
- Start downloading
- You will end up with a zipped (*.zip), a self-extracting archive file (*.exe), or an Installer application
- For .zip and .exe files, just unpack it in the destination folder on your machine
- If you selected the installer version, just run it and follow the installer instructions

If you want to move your installation to a different location, you can just move the “KNIME _2.x.y” folder to the selected location.

1.1. The KNIME Download web page



1.4. The workspace

To start KNIME, open the folder “KNIME_2.x.y” where KNIME has been installed and run knime.exe (or knime on a Linux/Mac machine).

After the splash screen, the “Workspace Launcher” window requires you to enter the path of the workspace.

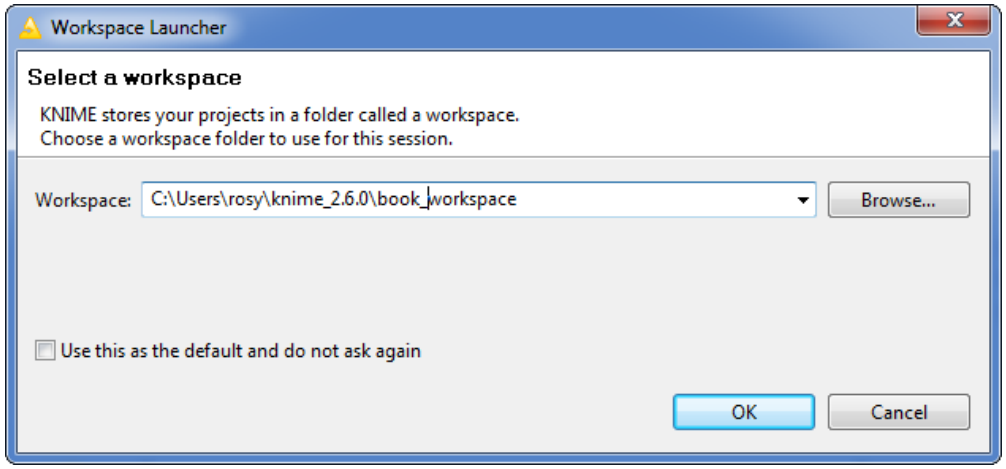
The” Workspace Launcher”

The **workspace** is the folder where all current workflows and preferences are saved for the next KNIME session.

The workspace folder can be located anywhere on the hard-disk.

By default, the workspace folder is “..\knime-workspace”. However, you can easily change that, by changing the path proposed in the “Workspace Launcher” window, before starting the KNIME working session.

1.2. The „Workspace Launcher“ window



From inside KNIME, you can change the workspace, by selecting “File” in the top menu and then “Switch Workspace”. After selecting the new workspace, KNIME closes and reopens, showing then the workflow list from the selected workspace. Notice that if the workspace folder does not exist, it will be automatically created.

If I have a large number of projects or of customers or of both, I use a new workspace for each new project/customer. This keeps my work space clean and tidy and protects me from mixing up information by mistake. For this project I used the workspace “KNIME_2.x.y\book_workspace”.

1.5. Download the KNIME extensions

The KNIME Analytics Platform is an open source product. As every open source product, it benefits from the feedback and the side utilities that the open source community develops. A number of extensions are available for the KNIME Analytics Platform and the KNIME Enterprise products alike. However, the KNIME Extensions are not available together with the KNIME Analytics Platform: they need to be installed separately on a running KNIME.

To install a new KNIME extension, there are two possible ways.

1. From the Top Menu, select "**File" -> "Install KNIME Extensions",** select the extension, click the "**Next**" button and follow the wizard instructions.

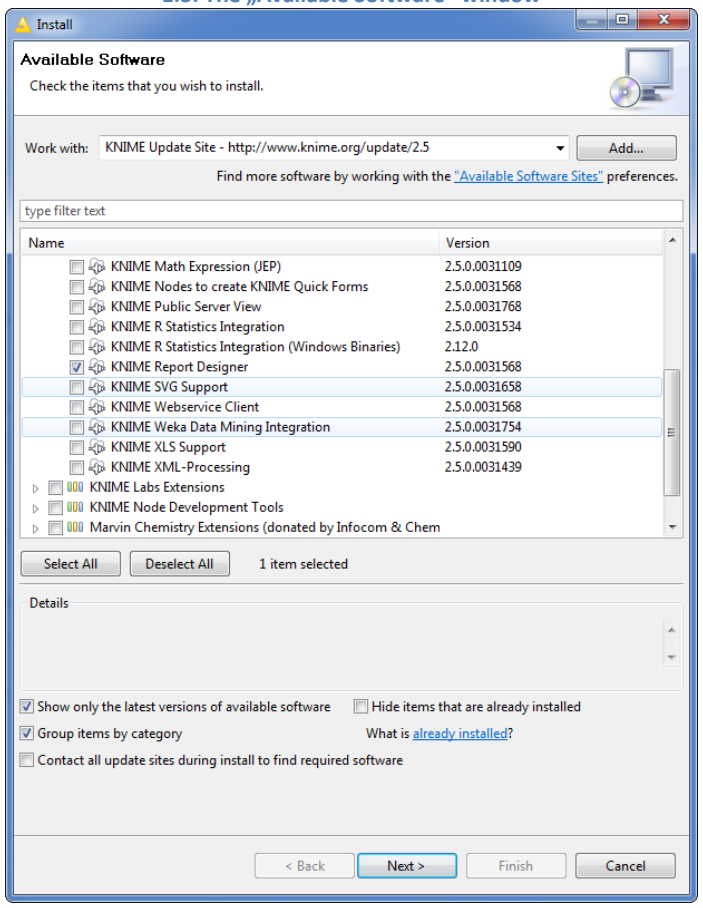
OR

2. From the Top Menu, select "**Help" -> "Install New Software".** In the "Available Software" window, in the "Work with" textbox, select the URL with the KNIME update site (usually named "KNIME Update Site" - http://www.knime.org/update/2.x). Then select the extension, click the "**Next**" button and follow the wizard instructions.

Once the KNIME extension has been installed and KNIME has been restarted, you should see the new node/category in the "Node Repository" in the KNIME workbench (see a few sections ahead).

For example, after installing the KNIME Report Designer extension, you should see a category "Reporting" in the "Node Repository" panel on the left.

1.3. The „Available Software" window



In the "Available Software" window there are some extension groups available: KNIME & Extensions, KNIME Labs Extensions, KNIME Development Tools, Sources, and more. "KNIME &Extensions" contains all extensions provided by KNIME for the current release; "KNIME Labs Extensions" contains

a number of packages developed by KNIME, ready to use, but not yet of release quality; "KNIME Development Tools" contains packages with some useful tools for the node developers; and "Source" contains the KNIME source code. Specific packages donated by third parties or community entities might are also available in the list of extensions.

1.6. The KNIME workflow

KNIME does not work with scripts, it works with workflows.

What is a workflow

A workflow is an **analysis flow**, i.e. the **sequence of analysis steps** necessary to reach a given result. It is the pipeline of the analysis process, something like:

Step 1. Read data
Step 2. Clean data
Step 3. Filter data
Step 4. Train a model

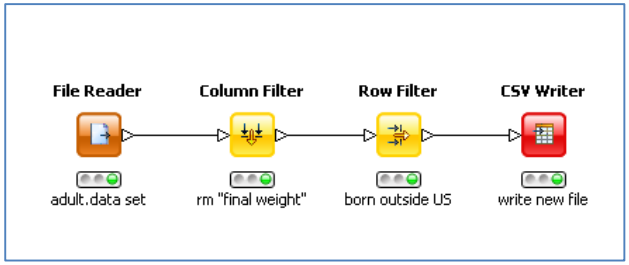
KNIME implements its workflows **graphically**. Each step of the data analysis is implemented and executed through a little box, called **node**. A sequence of nodes makes a workflow.

In the KNIME whitepaper [1] a workflow is defined as follows: *"Workflows in KNIME are essentially graphs connecting nodes, or more formally, direct acyclic graphs (DAG)."* (http://www.kdd2006.com/docs/KDD06_Demo_13_Knime.pdf)

Below is an example of a KNIME workflow, with:

- a node to read data from a file,
- a node to exclude some data columns,
- a node to filter out some data rows,
- a node to write the processed data into a file.

1.4. Example of a KNIME workflow



Note. A workflow is a data analysis sequence, which in a traditional programming language would be implemented by a series of instructions and calls to functions. KNIME implements it graphically. This graphical representation is more intuitive to use, lets you keep an overview of the analysis process, and makes for the documentation as well.

What is a node

A node is the **single processing unit** of a workflow.

A node takes a data set as input, processes it, and makes it available at its output port. The "processing" action of a node ranges from modeling, like an Artificial Neural Network Learner node, to data manipulation, like transposing the input data matrix, from graphical tools, like a scatter plot, to reading/writing operations.

Every node in KNIME has 3 states:

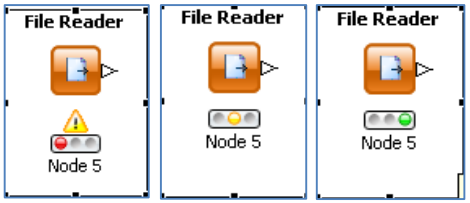
- Inactive and not yet configured → **red** traffic light
- Configured but not yet executed → **yellow** traffic light
- Executed successfully → **green** traffic light

If the node is executed with errors (unsuccessfully), its status stays at the yellow traffic light.

Nodes containing other nodes are called **meta nodes**.

Below are three examples of the same node (a File Reader node) in each one of the three states.

1.5. File Reader node with different status



1.7. The KNIME workbench

After accepting the workspace path, the KNIME workbench opens on a "Welcome to KNIME" page. This page provides a few links to get started and to documentation. It also shows a link to create a new workflow, to the "Learning Hub" web page where you can find links to all available tutorials, videos, and other learning material, to the EXAMPLES workflows, to the extensions, and to all most recently used workflows. By selecting "Go to my workflows", you then reach the workflow editor.

The KNIME workbench was developed as an Eclipse Plug-in and many of its features are inherited from the Eclipse environment, i.e. many items on the workbench are actually referring to a Java programming environment and are not necessarily of interest for KNIME beginners. I will warn the reader, when the item on the KNIME workbench is not directly related to the creation of KNIME workflows. The "KNIME Workbench" consists of a top menu, a tool bar, and a few panels. Panels can be closed and moved around.

Let's have a closer look at the KNIME workbench.

The KNIME workbench

Top Menu: File, Edit, View, Node, Help

Tool Bar: New, Save (Save As, Save All), Undo/Redo, Open Report (if reporting was installed), Align selected nodes vertically/horizontally, Auto layout, Configure, Execute options, Cancel execution options, Reset, Edit node name and description, Open node's first out port table, Open node's first view, Open the "Add Meta node" Wizard, , Hide all node names, Loop execution options, Insert grid in workflow editor

KNIME Explorer

This panel shows the list of workflow projects available in the selected workspace (LOCAL) or on the EXAMPLES server or on other connected KNIME servers.

Favorite Nodes

This panel stores the nodes that are used most often or most recently or that for some other reason you want to keep at hand.

Workflow Editor

The central area consists of the "Workflow Editor" itself.

A node can be selected from the "Node Repository" panel and dragged and dropped here, in the "Workflow Editor" panel.

Nodes can be connected by clicking the output port of one node and releasing the mouse either at the input port of the next node or at the next node itself.

Node Description

If a node is selected in the "Workflow Editor" or in the "Node Repository", this panel displays a summary description of the selected node's functionalities.

Node Repository

This panel contains all the nodes that are available in your KNIME installation. It is something similar to a palette of tools when working in a report or with a web designer software. There we use graphical tools, while in KNIME we use data analytics tools.

Outline

The "Outline" panel contains a small overview of the contents of the "Workflow Editor". The "Outline" panel might not be of so much interest for small workflows. However, as soon as the workflows reach a considerable size, all the workflow's nodes may no longer be visible in the "Workflow Editor" without scrolling. The "Outline" panel, for example, can help you locate newly created nodes.

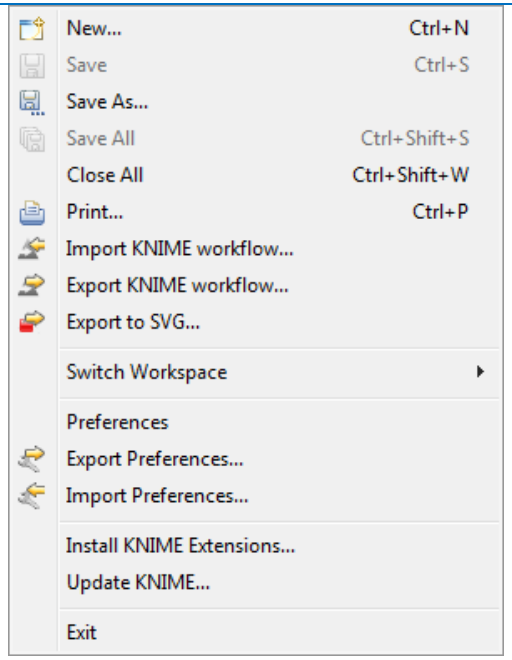
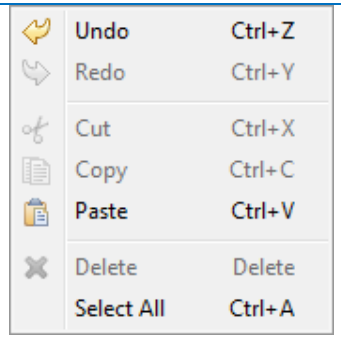
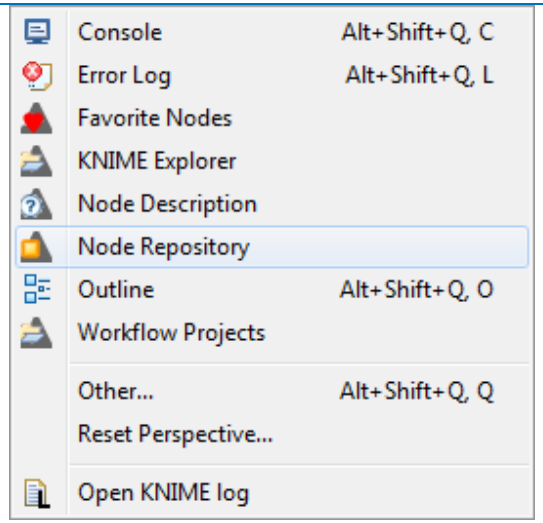
Console

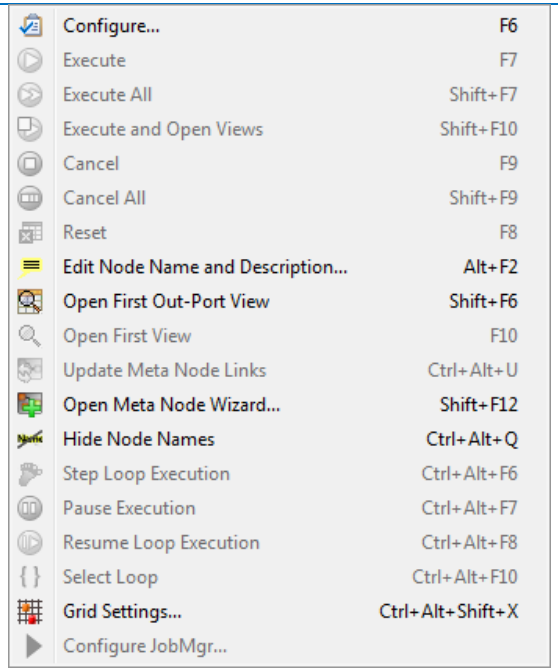
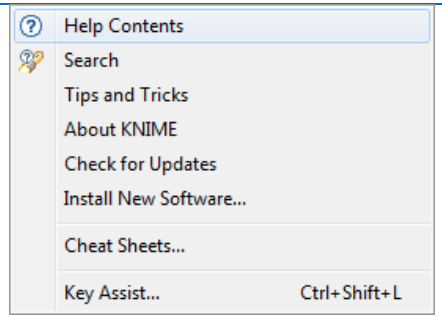
The "Console" panel displays error and warning messages to the user.

This panel also shows the location of the log file, which might be of interest when the console does not show all messages.

There is a button in the tool bar as well to show the log file associated with this KNIME instance.

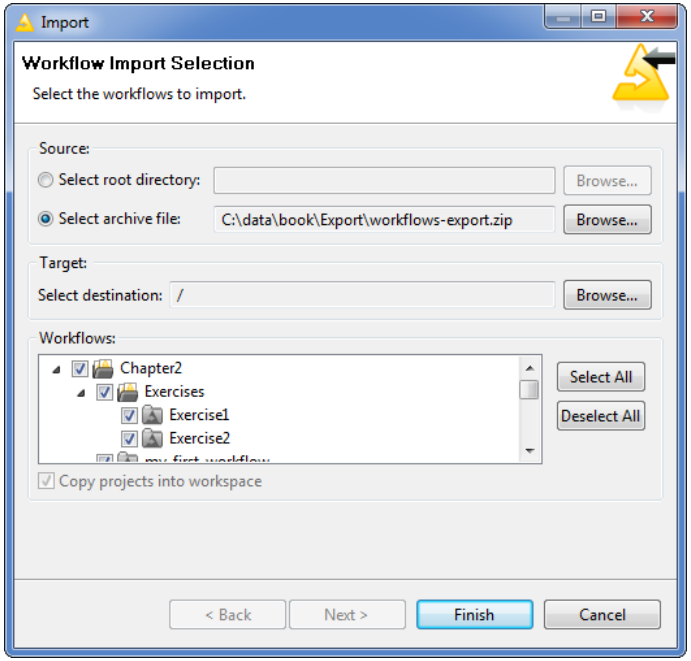
Top menu

File	Edit	View
		
File includes the traditional File commands, like "New" and "Save", in addition to some KNIME specific commands, like: - Import/Export KNIME workflow… - Export to SVG - Switch Workspace - Preferences - Export/Import Preferences - Install KNIME Extensions - Update KNIME	**Edit** contains the usual edit commands. **Cut, Copy, Paste,** and **Delete** refer to selected nodes in the workflow. **Select All** selects all the nodes of the workflow in the workflow editor.	**View** contains the list of all panels that can be opened in the KNIME workbench. A closed panel can only be re-opened here. Also, when the panel disposition is messed up, the option "Reset Perspective" re-creates the original panel layout of KNIME when it was started for the first time. Option "Other" opens additional views useful to customize the workbench.

Node	Help
	
Node refers to all possible operations that can be performed on a node. A node can be: - Configured - Executed - Cancelled (stopped during execution) - Reset (resets the results of the last “Execute” operation) - Given a name and description - Set to show its View (if any) Options are only active if they are possible. For example, an already successfully executed node is not re-executed unless it is reset or its configuration has been changed. The “Cancel” and “Execute” options are then inactive. Option “Open Meta Node Wizard” starts the wizard to create a new meta node in the workflow editor.	**Help Contents** provides general Help about the Eclipse Workbench, BIRT, and KNIME. **Search** opens a panel on the right of the “Node Description” panel to search for specific Help topics or nodes. **Install New Software** is the door to install KNIME Extensions from the KNIME Update sites. **Cheat Sheets** offer tutorials on specific Eclipse topics: the reporting tool, cvs, Eclipse Plug-ins. **Key Assist** summarizes all keyboard commands for the workflow editor.

Let's now go through the most frequently used items in the Top Menu.

1.7. Window „Import workflows"



"File" -> "Import KNIME workflow" reads workflows and copies them into the current workspace.

Option "Select root directory" copies the workflow from a folder into the current workspace (LOCAL).
Option "Select archive file" reads a workflow from a zipped file into the current workspace (LOCAL). Zipped files are created by the specular option "File"-> "Export KNIME workflow"

"File" -> "Export KNIME workflow" exports the selected workflow to a zip file.

Option "Exclude data from export" enables the export of nodes only, without the associated data. This generates considerably smaller export files.

Simply copying a workflow from one folder to another can create a number of problems related to internal KNIME updates. Copying workflows by using the option "Import KNIME workflow" is definitely safer.

"File" -> "Install KNIME Extensions" and "**Help" -> "Install New Software"** both link to the dialog window for the installation of KNIME Extensions from the KNIME Update sites (see Fig. 1.3).

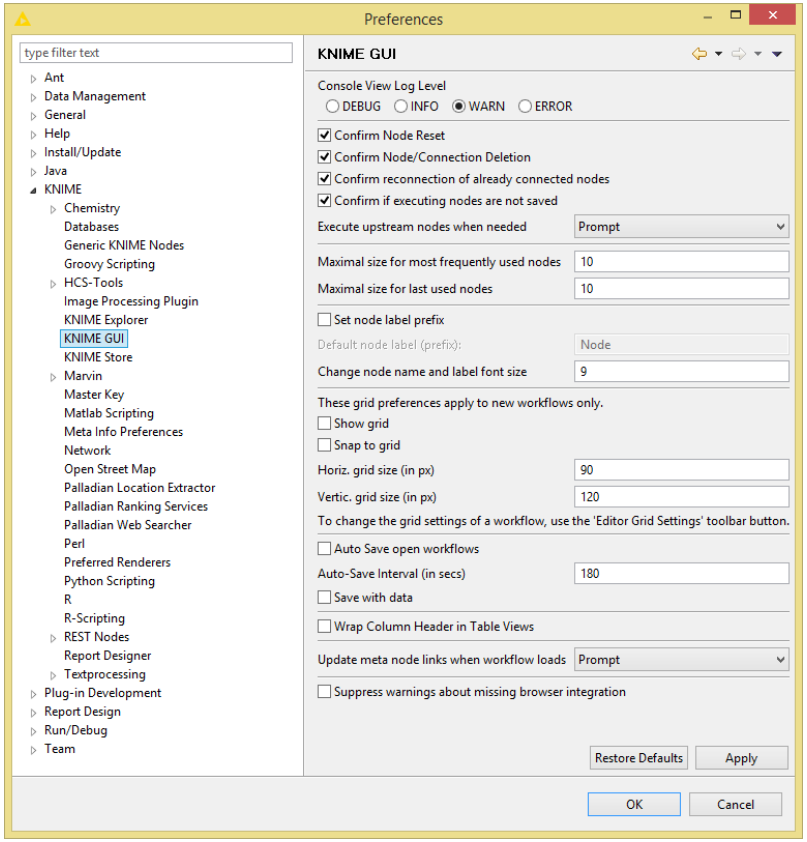
"File" -> "Switch Workspace" changes the current workspace with a new one.

"File" -> "Export to SVG" exports a KNIME workflow into an SVG file.

"File" -> "Preferences" brings you to the window where all KNIME settings can be customized. Let's check these KNIME Settings.

- **Chemistry** has settings related to the KNIME Renderers in the chemistry packages.
- **Databases** specifies the location of specific database drivers, not already available within KNIME. Indeed, the most common and most recent database drivers should already available within KNIME. Thus, if you work with some rare or old database version, the path to the database driver should be set here.

1.8. The "Preferences" window



- **KNIME Explorer** allows to access shared repositories via KNIME TeamSpace and KNIME Server. Both of these are available as add-on products from KNIME.com.
- **KNIME GUI** contains a few settings about the KNIME workbench options and layout.
- **KNIME Store** contains the credentials to access the KNIME Store web page http://tech.knime.org/knime-store from the KNIME Analytics Platformto check for new products, licenses, etc … You need to be signed up into the KNIME Forum to access the KNIME Store page.
- **Master Key** contains the master key to be used in nodes with an encryption option, like database connection nodes. Since KNIME 2.3 database passwords are passed via the "Credentials" workflow variables and the Master Key preference has been deprecated. You can still find it in the Preferences menu for backward compatibility.
- In **Meta Info Preferences** you can upload meta-info template for nodes and workflows.
- After Meta Info Preferences, you can find the preference settings for the installed packages. If the R Extension had been installed, you would find an R menu item. In figure 1.8 you can see the preferences pages for R, Report Designer, Perl, Text Processing, Open Street Maps, Network Graphs, and more extension packages.

Export Preferences and **Import Preferences** in the "File" menu respectively exports and imports the "Preferences" settings into and from a *.epf file. These two commands come in handy when, for example, a new version of KNIME is installed and we want to import the old preferences settings.

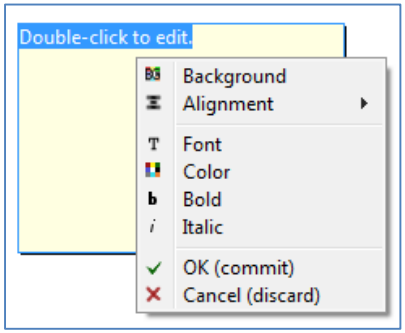
The "**Auto Layout**" button in the tool bar automatically adjusts the position of the nodes in the workflow to produce a clean, ordered, and easy to explore workflow. This auto-layout operation becomes particularly useful when, for examplke after a long development session, the workflow overview has become difficult.

1.9. The "Auto Layout" button in the tool bar

When building a workflow, it is also possible to include workflow **annotations** to comment and explain the function of each node or group of nodes. The result is an improved overview of the workflow general goal and single sub-tasks.

Workflow Annotations

1.10. The Annotation Editor



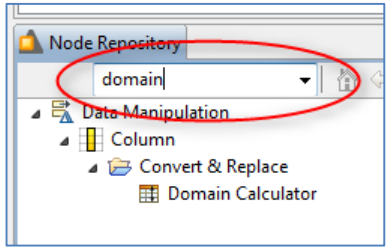
To insert a new annotation:

- right-click anywhere in the workflow editor
- select “New Workflow Annotation”
- a pale-yellow small frame appears with written “Double-click to edit”: this is the default annotation frame
- double-click the frame to edit it
- the context menu of the annotation contains the annotation editor options. Right-click anywhere in the annotation frame and use the context menu to edit text style, font color, background color, and text alignement.

Note that fonts related options are disabled in the context menu if no text has been selected in the annotation frame.

Node Repository Search box

1.11. The Search box in the "Node Repository" panel



In the “Node Repository” panel there is a **search box**.

If you type a keyword in the search box and then hit “Enter”, you obtain the list of nodes with that keyword in the name. Press the “Esc“ key to see all nodes again.

For example, here we searched for all nodes with the keyword “domain” in their name.

For all keyboard lovers, most KNIME commands can also run via **hotkeys**. All hotkeys are listed in the KNIME menus on the side of the corresponding commands or in the tooltip messages of the icons in the Tool Bar under the Top Menu. Here are the most frequently used hotkeys.

Hotkeys

Node Configuration

- **F6** opens the configuration window of the selected node

Node Execution

- **F7** executes selected configured nodes
- **Shift + F7** executes all configured nodes
- **Shift + F10** executes all configured nodes and opens all views

Stop Node Execution

- **F9** cancels selected running nodes
- **Shift + F9** cancels all running nodes

To move nodes

- **Ctrl + Shift + Arrow** moves the selected node in the arrow direction

Node Resetting

- **F8** resets selected nodes

Save Workflows

- **Ctrl + S** saves the workflow
- **Ctrl + Shift + S** saves all open workflows
- **Ctrl + Shift + W** closes all open workflows

Meta-Node

- **Shift + F12** opens Meta Node Wizard

To move Annotations

- **Ctrl + Shift + PgUp/PgDown** moves the selected annotation in the front or in the back of all the overlapping annotations

Note. The "KNIME Explorer" can also host data. Just create a folder under the workspace folder, fill it with data files, and select "Refresh" in the context-menu of the "KNIME Explorer" panel.

1.8. Customizing the KNIME Workbench

KNIME also offers a few options to customize the KNIME workbench, to better suites your personality.

The first possible customization is to switch on the grid in the workflow editor. The grid feature contains two options:

1. "Show grid lines". This shows grid lines in the workflow editor and allows to better align nodes and annotations manually.

2. “Snap to grid”. This option attaches nodes and annotations to the closest available corner of the grid. It gives you less manual freedom, but the result is cleaner and more ordered in shorter time.

Grid in Workflow Editor

There are two spots from where to switch the grid on:

- The last button in the Tool Bar (see Fig. 1.9)
- The last option, named “Grid Settings”, in the menu “Node” in the top menu

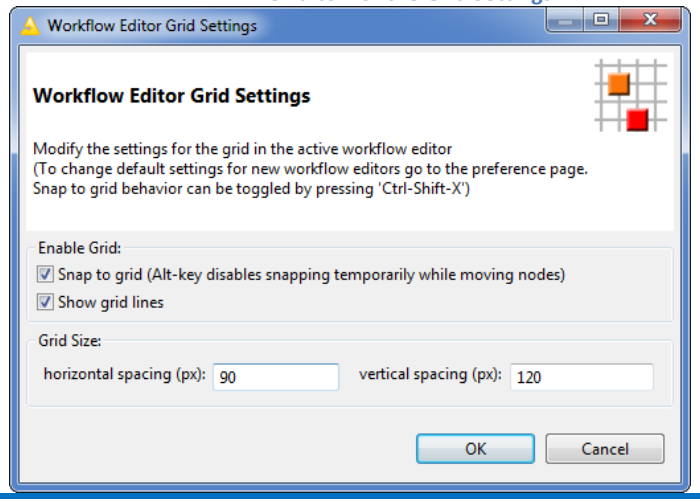
Both options lead to the “Workflow Editor Grid Settings” window.

Here, you can enable:

- one or both grid options (“show grid lines” and “snap to grid”)
- the grid size

If the grid is enabled, it will be shown in the workflow editor.

1.12. The Editor for the Grid Settings



Another source of customization can be found in the “View” menu in the top menu. Two very popular views are the “Node Monitor”, the “Custom Node Repository”, and the “Licenses” and Server related views, if you have a connected server.

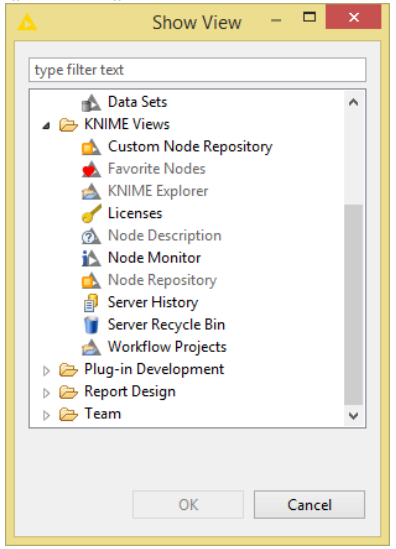
All these extra views can be found in the Top Menu under “View” -> “Other” -> “KNIME Views”.

The “Node Monitor” view helps, especially during the development phase, to monitor and debug the workflow execution.

The “Custom Node Repository allows for a customized “Node Repository” with only a subset of nodes available.

“Licenses” allows to monitor your license situation, if you have one.

1.13. Additional KNIME Views from „View“ -> „Other“ -> “KNIME Views”

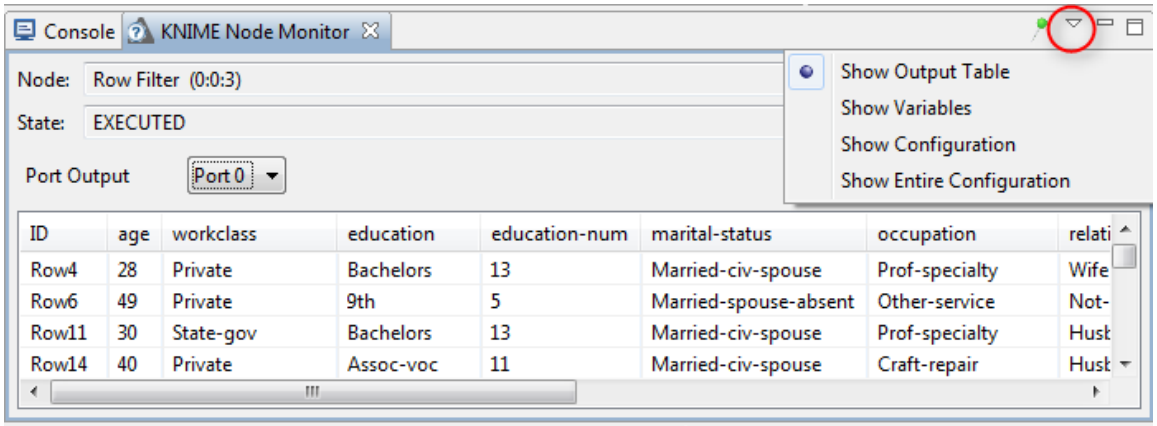


KNIME Node Monitor

To insert the “KNIME Node Monitor” panel in the workbench:

- Select “View” in the top menu
- Select “Other…”
- In the “Show View” window, expand the “KNIME Views” item and double-click “Node Monitor” (Fig. 1.13)
- A panel, named “Node Monitor”, appears on the side of the “Console” panel
- The panel shows the values for the output flow variables or for the output data or for the configuration settings of the selected node in the workflow editor.
- You can decide what to show (data, configuration, variables), via the menu in the top right corner.

1.14. The Annotation Editor



Mounting Servers in the KNIME Explorer

You now need to add KNIME instances (servers or teamspaces) to the "KNIME Explorer" panel.

- Select the "KNIME Explorer" panel
- Click the "Configure Explorer View" button (Fig. 1.15).
- The "Preferences (Filtered)" window opens on the "KNIME Explorer" page.
- In the "Preferences (Filtered)" window lists all KNIME Servers and Teamspaces uploaded in this KNIME instance. The first two KNIME Servers uploaded by default on every KNIME instance are the local KNIME Analytics Platform "LOCAL" and the KNIME public Server "EXAMPLES".

Use the "New" and the "Remove" button to add /remove connections to remote servers.

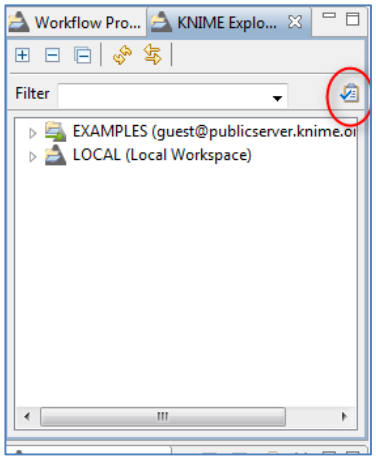
When installing a mountpoint for a new server (the "New" button), in the "Select New Content" window, use the "Test Connection" button to automatically retrieve the default mountpoint for the selected server (Fig. 1.17).

The same KNIME Explorer "Preferences" page in figure 1.16 can be reached via "File" in the top menu -> "Preferences" -> "KNIME Explorer".

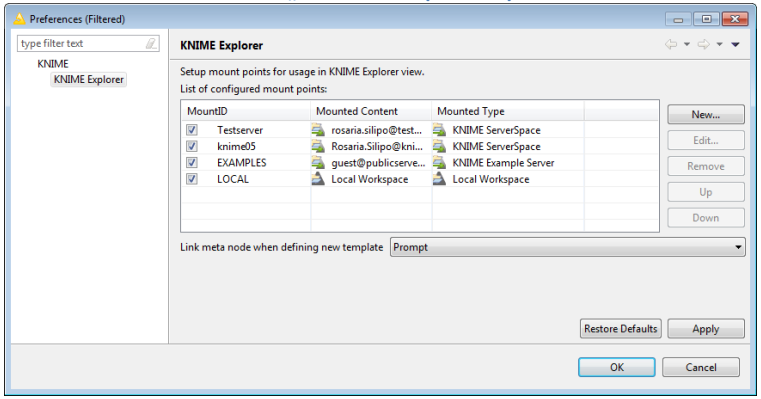
To login into any of the available servers in the "KNIME Explorer" panel in the upper left corner of the KNIME workbench:

- right-click the server name in the "KNIME Explorer" panel
- select "Login"
- provide the credentials

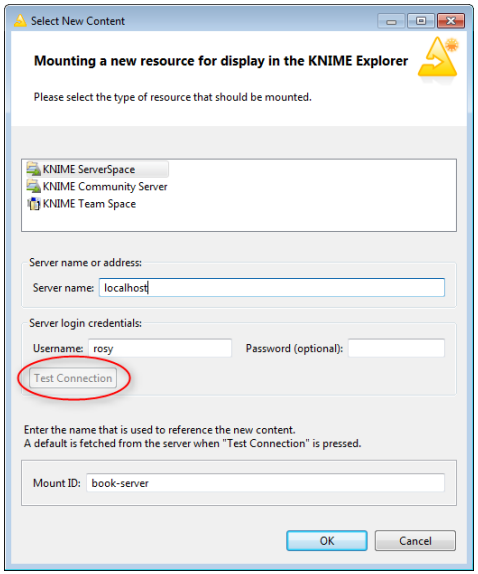
1.15. The „Configure KNIME Explorer" button



1.16. The „Preferences (Filtered)" window



1.17. The “Select New Content”” window



The EXAMPLES Server

A link to the KNIME Public Server (EXAMPLES) is available in the “KNIME Explorer” panel. This is a server provided by KNIME to all users for tutorials and demos. There you can find a number of useful examples on how to implement specific tasks with KNIME. To connect to the EXAMPLES Server:

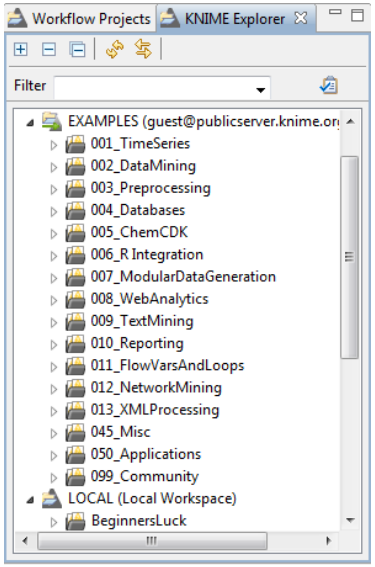
- right click “EXAMPLES” in the “KNIME Explorer” panel
- select “Login”

You should be automatically logged in as a guest.

To transfer example workflows from the EXAMPLES Server to your LOCAL workspace, just drag and drop or copy and paste (Ctrl-C, Ctrl-V in Windows) them from “EXAMPLES” to “LOCAL”.

You can also open the EXAMPLES workflows in the workflow editor, however only temporarely and in read-only mode. A yellow warning box on top warns that this workflow copy will not be saved.

1.18. Example Workflows in the KNIME Public Server



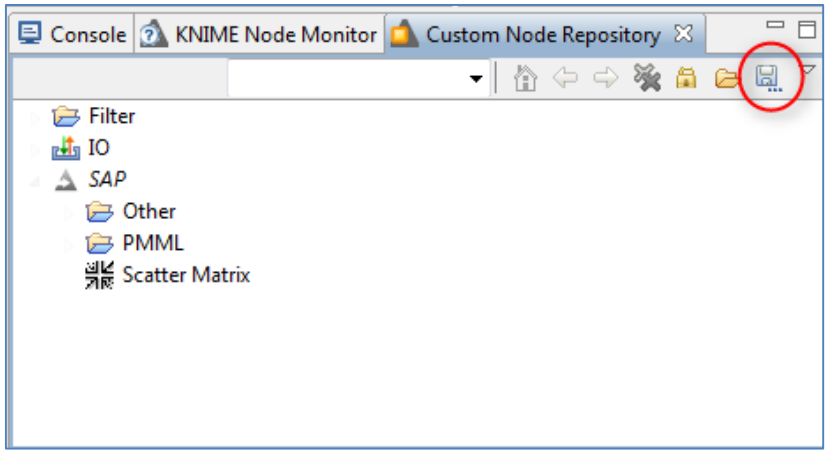
Another possible customization lies in the “Favorite Nodes” panel in the middle part of the left side bar, where you can store all the nodes that you want to keep at hand. Sometimes though, this is not enough. Sometimes KNIME users in the same group might need to share the same view of nodes. If you have a license for one of the KNIME commercial products, you can build a customized “Node Repository”.

Custom Node Repository

To insert the “Custom Node Repository” panel in the workbench:

- Select “View” in the top menu
- Select “Other…”
- In the “Show View” window (Fig. 1.13), expand the “KNIME Views” item and double-click “Custom Node Repository”
- A panel, named “Custom Node Repository”, appears on the side of the “Console” panel

1.19. The Customizable Node Repository



In this panel:

- You can drag and drop nodes, categories, and subcategories from the “Node Repository” panel.
- You can even create your own categories and fill them up with specific nodes.
- Finally, you can export your “Custom Node Repository” (Fig. 1.19) and pass it to other users.

1.9. Data and workflows for this book

In the course of this book we will implement a few workflows to show how KNIME works. In each chapter we will build one or more workflows and we will expect the reader to build a few more in the exercises. The data and workflows used and implemented here are available in the “Download Zone” of this book. There you will find a folder for each chapter containing the chapter’s demonstrative workflows and a subfolder “Exercises”. The subfolder “Exercises” contains the solutions to the exercises proposed in the same chapter.

Structure of the "Download Zone"

Data

- Data1.txt
- Projects.txt
- SoccerWorldCup2006.txt
- SoccerWorldCup2010.txt
- Adult.data
- Iris.data
- Yellow-small.data (Balloons)
- Wine.data
- Web site 1.txt

Chapter 2

- my_first_workflow.zip
- Exercises
  - Exercise1.zip
  - Exercise2.zip

Chapter 3

- writeToDB.zip
- my_first_data_exploration.zip
- Exercises
  - Exercise1.zip
  - Exercise2.zip
  - Exercise3.zip

Chapter 4

- data_preparation.zip
- my_first_data_model.zip
- my_first_ANN.zip
- Clustering and Regression.zip
- Exercises
  - Exercise1.zip
  - Exercise2.zip
  - Exercise3.zip

Chapter 5

- Projects.zip
- Exercises
  - Exercise1.zip
  - Exercise2.zip
  - Exercise3.zip

The data used for the exercises and for the demonstrative workflows of this book were either generated by the author or downloaded from the UCI Machine Learning Repository, a public data repository (http://archive.ics.uci.edu/ml/datasets). If the data set belongs to the UCI Repository, a full link is provided here to download it. Data generated by the author, that is not public data, are located in the "Download Zone" in the "Data" folder.

Data from the UCI Machine Learning Repository:

- Adult.data: http://archive.ics.uci.edu/ml/datasets/Adult
- Iris data: http://archive.ics.uci.edu/ml/datasets/Iris
- Yellow-small.data (Balloons) http://archive.ics.uci.edu/ml/datasets/Balloons
- Wine data: http://archive.ics.uci.edu/ml/datasets/Wine

1.10. Exercises

Exercise 1

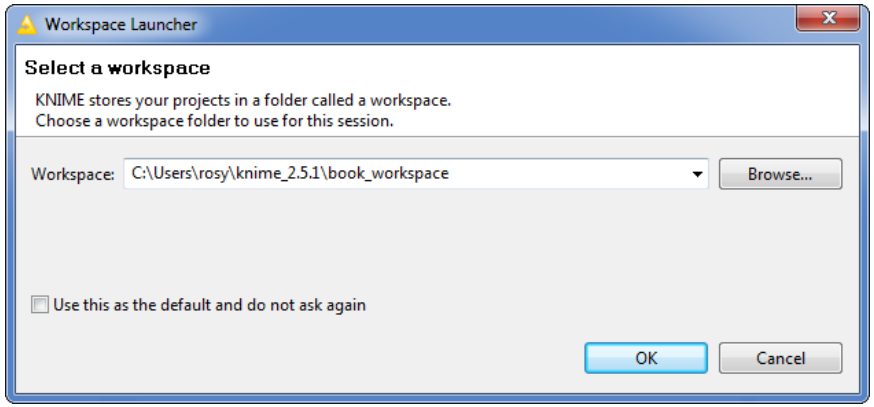
Create your own workspace and name it “book_workspace”. You will use this workspace for the workflows and the exercises of this book.

Solution to Exercise 1

2. Launch KNIME
3. Click “Browse”
4. Select the path for your new workspace
5. Click “OK”

To keep this as your default workspace, enable the option on the bottom left.

1.20. Exercise 1: Create workspace "book_workspace"



Exercise 2

Install the following extensions:

- KNIME Math Expression Extension (JEP)
- KNIME External Tool Node
- KNIME Report Designer

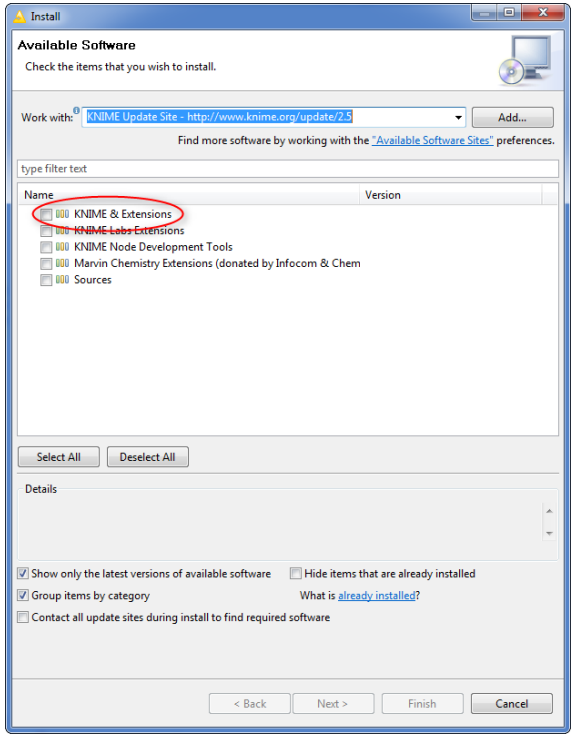
Solution to Exercise 2

From Top Menu, select **“File“ -> “Install KNIME Extensions”**

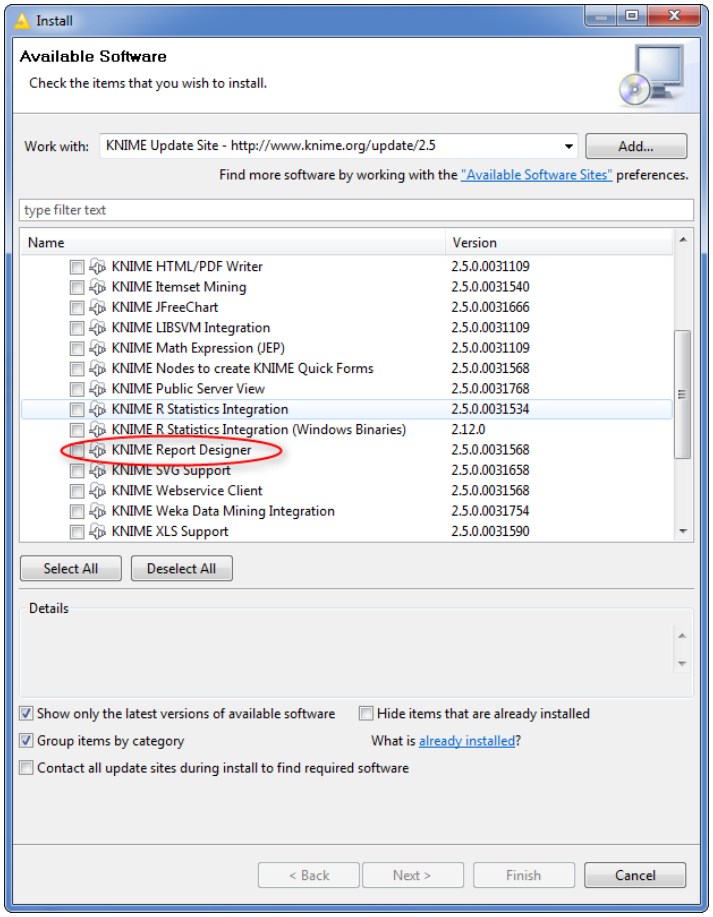
Select Extensions

Click **“Next”**

1.21. Exercise 2: List of KNIME Extensions



1.22. Exercise 2: KNIME Report Extension



Exercise 3

Search all “Row Filter” nodes in the Node Repository.

From the “Node Description” panel, can you explain what the difference is between a “Row Filter”, a “Reference Row Filter”, and a “Nominal Value Row Filter”?

Show the node effects by using the following data tables:

Original Table

Position	name	team
1	The Black Rose	4
2	Cynthia	4
3	Tinkerbell	4
4	Mother	4
5	Augusta	3
6	The Seven Seas	3

Reference Table

Ranking	scores
1	22
3	14
4	10

Solution to Exercise 3

Row Filter

The node allows for row filtering according to certain criteria. It can include or exclude: certain ranges (by row number), rows with a certain row ID, and rows with a certain value in a selectable column (attribute). In the example below we used the following filter criterion: `team > 3`

Original table

Position	name	team
1	The Black Rose	4
2	Cynthia	4
3	Tinkerbell	4
4	Mother	4
5	Augusta	3
6	The Seven Seas	3

Filtered table

Position	name	team
1	The Black Rose	4
2	Cynthia	4
3	Tinkerbell	4
4	Mother	4

Reference Row Filter

This node has two input tables. The first input table, connected to the top port, is taken as the reference table; the second input table, connected to the bottom port, is the table to be filtered. You have to choose the reference column in the reference table and the filtering column in the second table. All rows with a value in the filtering column that also exists in the reference column are kept, if the option "include" is selected; they are removed if the option "exclude" is selected.

Reference Table

Ranking	scores
1	22
3	14
4	10

Filtering Table

Position	name	team
1	The Black Rose	4
2	Cynthia	4
3	Tinkerbell	4
4	Mother	4
5	Augusta	3
6	The Seven Seas	3

Resulting Table

Position	name	team
1	The Black Rose	4
3	Tinkerbell	4
4	Mother	4

In the example above, we use "Ranking" as the reference column in the reference table and "Position" as the filtering column in the filtering table. We have chosen to include the common rows.

Nominal Value Row Filter

Filters the rows based on the selected value of a nominal attribute. A nominal column and one or more nominal values of this attribute can be selected as the filter criterion. Rows that have these nominal values in the selected column are included in the output data. Basically it is a Row Filter applied to a column with nominal values. Nominal columns are string columns and nominal values are the values in it.

In the example below, we use "name" as the nominal column and "`name = Cynthia`" as the filtering criterion.

Original table

Position	name	team
1	The Black Rose	4
2	Cynthia	4
3	Tinkerbell	4
4	Mother	4
5	Augusta	3
6	The Seven Seas	3

Filtered table

Position	name	team
2	Cynthia	4

Chapter 2. My first workflow

2.1. Workflow operations

If you have started KNIME for the first time, your "KNIME Explorer" panel on the top left corner of the KNIME workbench should only contain one workflow named "Example Workflow". This "Example Workflow" shows a few basic operations with KNIME: import data, assign visual properties, calculate basic statistical measures, partition data, train and evaluate a model, and finally display and interactively explore the error values.

In order to keep your space clean, workflow groups can help you collect workflows by topic or project. I usually define a workflow group for each project or subproject I work on, into which I put all the workflows related to the same topic. Let's create then a new workflow group and call it "Chapter 2".

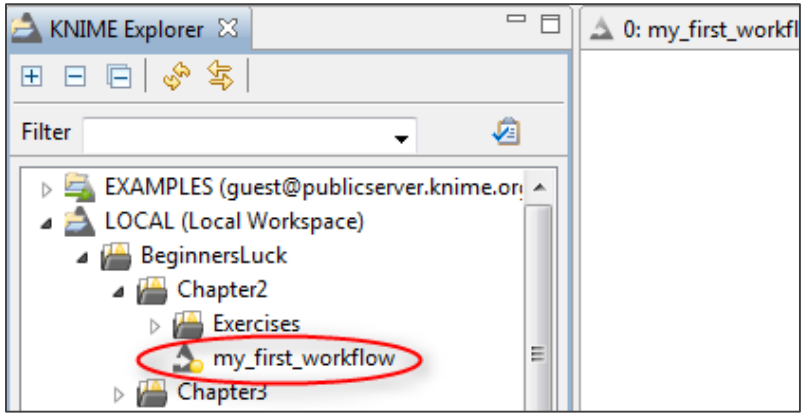
Once this has been done, we need to populate the newly created workflow group with a new workflow, let's call it "my_first_workflow".

Eventually in the "KNIME Explorer" panel, you will see workflow group "Chapter2" with a workflow named "my_first_workflow" in it.

For now, "my_first_workflow" is an empty workflow. If you double-click it, the workflow editor opens to an empty page.

Let's see now how to perform all these operations.

2.1. Workflow Structure in the „KNIME Explorer" panel

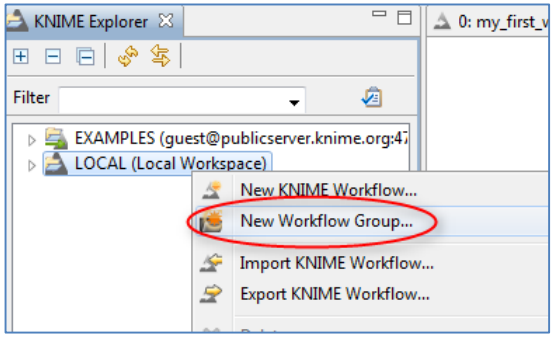


Create a new Workflow Group

In the “KNIME Explorer” panel:

- Right-click anywhere in the LOCAL workspace (or in a server space)
- Select “New Workflow Group”

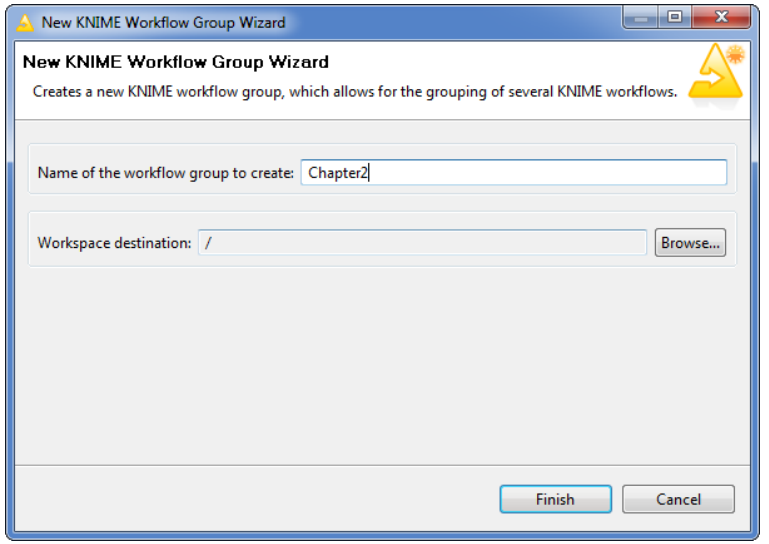
2.2. Create a new workflow group



In the “New KNIME Workflow Group Wizard” dialog:

- Enter the name of the workflow group
- Click “Finish”

2.3. Create a new workflow group named "Chapter2"



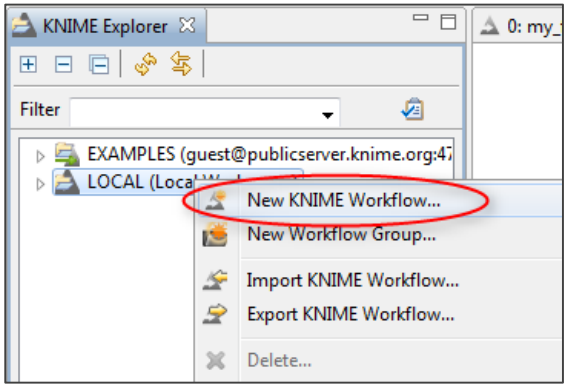
Note. If you already have workflow groups in your workspace, you will also need to select the final destination, for example in the root or in some of the already existing workflow groups. To visualize the possible workflow group destinations, click the “Browse” button.

Create a new workflow

- Right-click a workflow group in the “KNIME Explorer” panel

- Select “New KNIME Workflow”

2.4. Create a new workflow



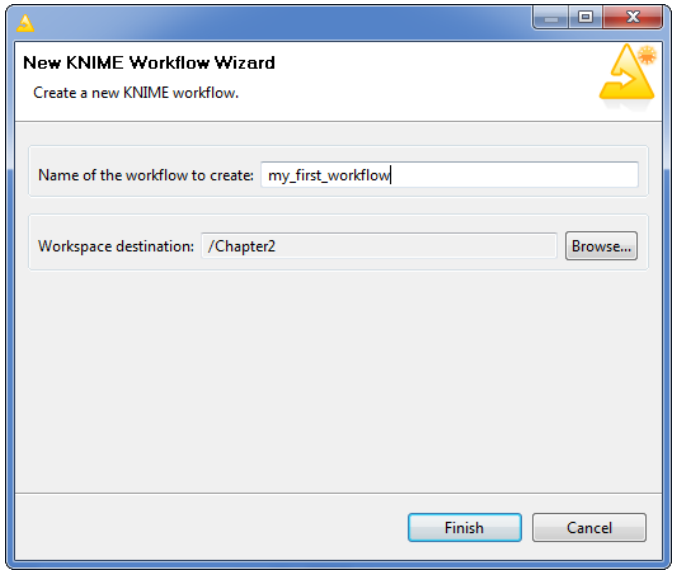
In the “New KNIME Workflow Wizard” dialog

- Enter the name of the new workflow

- Specify where it should be located, for example under an existing workflow group, by using the “Browse” button

- Click “Finish”

2.5. Create a new workflow named "my_first_workflow" and placed under "Chapter2"

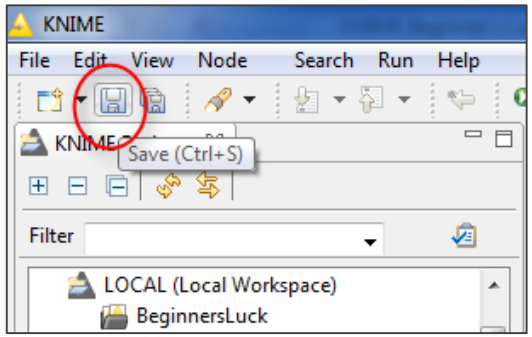


Save a workflow

To save a workflow, click the disk icon in the Top Menu.

Saving the workflow saves the workflow architecture, the nodes' configuration, and the data at the output of each node.

2.6. Save a workflow



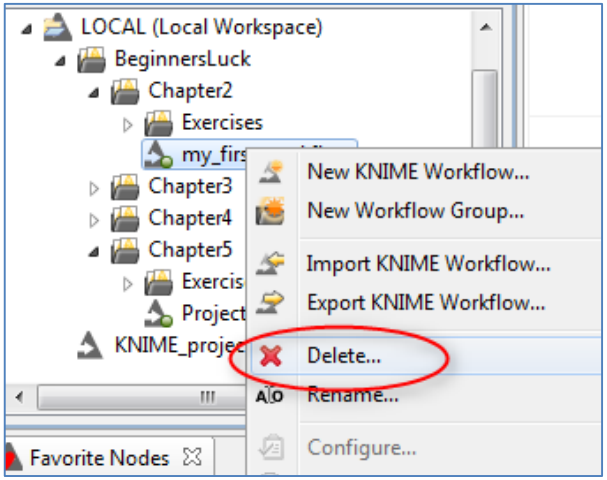
Delete a workflow

To delete a workflow

- Right-click the workflow in the "KNIME Explorer" panel
- Select "Delete"

In the "Confirm Deletion" dialog, you will be asked if you really want to delete the workflow. Indeed, the "Delete" command removes the workflow project physically from the hard disk. There is no way to get it back.

2.7. Delete a workflow



2.2. Node operations

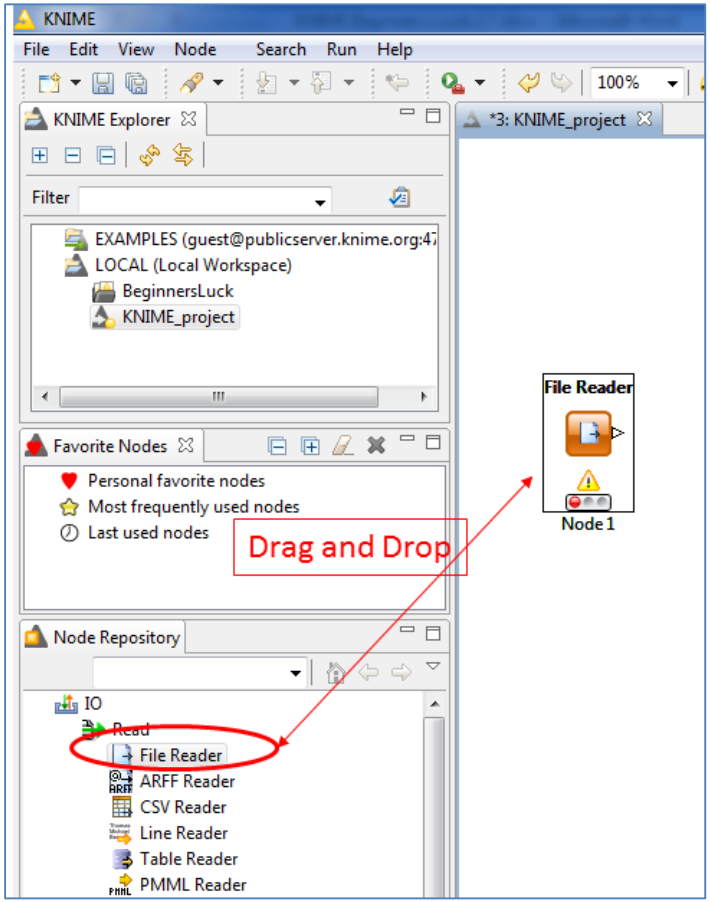
In Chapter 1, we have seen that a node is the basic computational unit in a KNIME workflow. We have also seen that nodes are available in the "Node Repository" panel on the bottom left, organized in categories. And we have seen that every node has three states: not yet configured (red), configured (yellow), and successfully executed (green).

In this section we are going to explore: how to add a new node to a workflow (status = inactive, not configured; **red light**), how to configure the node (status = configured, not executed; **yellow light**), and how to execute the node (status = successfully executed; **green light**).

Create a new node

- To create a new node, drag and drop the node from the "Node Repository" panel into the workflow editor OR double-click the node in the "Node Repository" panel. The node is usually imported with red traffic light status.
- To connect the node with existing nodes, click the output triangle of the first node and release the mouse at the [input triangle of the] second node. To move this connection to the next input port, Shift + double click the destination node.
- The node now needs configuration.
- To know what the node task and settings are, you can check the "Node Description" view on the right. This has a short description of the node functionality.
- If the "Node Description" view is not yet open, then:
    - Open the "View" item in the Top Menu,
    - Select "Node Description"

2.8. Drag and drop the node into the workflow editor to create a new node



Configure a node

- Double-click the node

OR

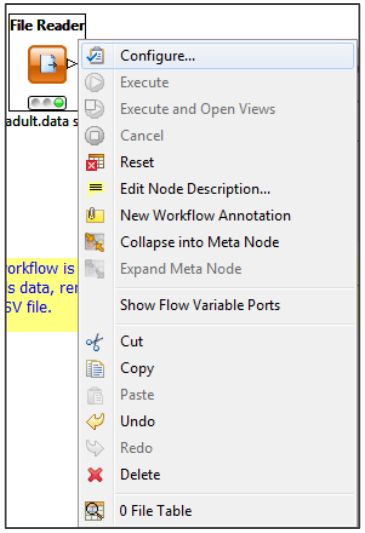
- Right-click the node and select “Configure”

If all input ports are connected, the configuration dialog appears for you to fill in the configuration settings.

Every node has a different configuration dialog, since every node performs a different task.

After a successful configuration, the node switches its traffic light to yellow.

2.9. Right-click the node and select "Configure" to configure the node



Execute a node

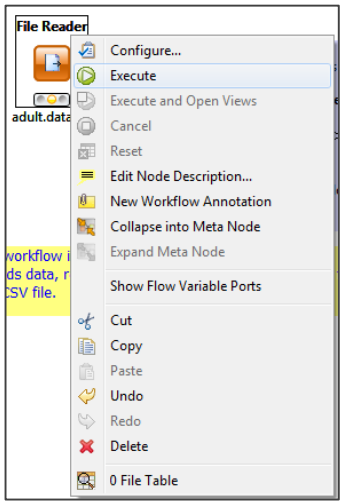
The node is now configured, which means it knows what to do.

In order to actually make it perform its task, we need to execute it.

- Right-click the node
- Select “Execute”.

If execution is successful, the node switches its traffic light to green.

2.10. Right-click the node and select "Execute" to run the node



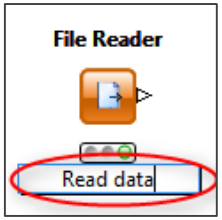
Finally we need to give a meaningful name to this node for documentation purposes, to easily recognize which task it is performing inside the workflow. The default name assigned to a new node by KNIME would be “Node n”, where “n” is a progressive number. The node name however can be customized. This, together with the workflow annotations described in chapter 1, keeps the overview of the workflow clear and fulfills the purpose of the workflow documentation.

Node Name

In order to change the node’s name:

- Double-click the node name under the node
- The node name becomes editable
- Write the new name for the node. You can use more lines just by inserting an “Enter” in the node name
- Click outside the node to commit the name change

2.11. Double-click the node name and enter a new name text

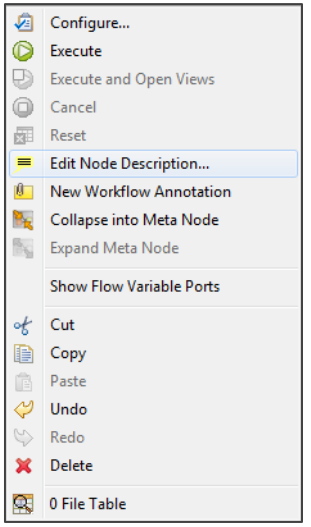


Node Description

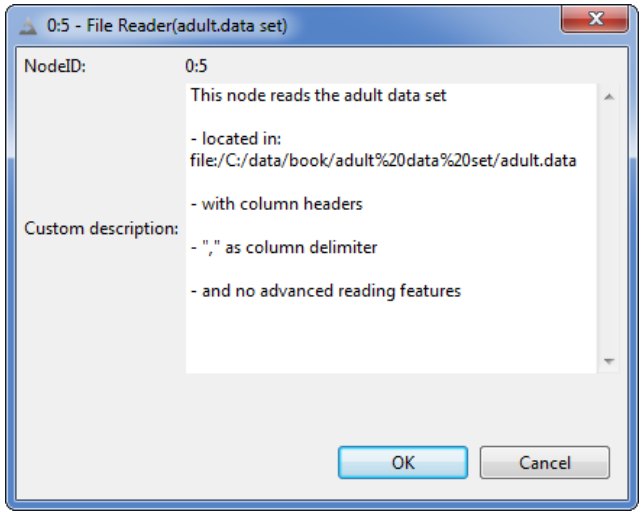
Besides the node name, you can also insert a quick description of the node task. In the context menu of the node, select the option “Edit Node Description”, that is:

- Right-click the node
- Select option “Edit Node Description …”
- In the “Node Description” window (Fig. 2.13)
    1. In the field “Custom Description”, write the node description
    2. Click “OK”

2.12. Double-click the node name and enter a new name text



2.13. The "Node Description" window with the node custom description text



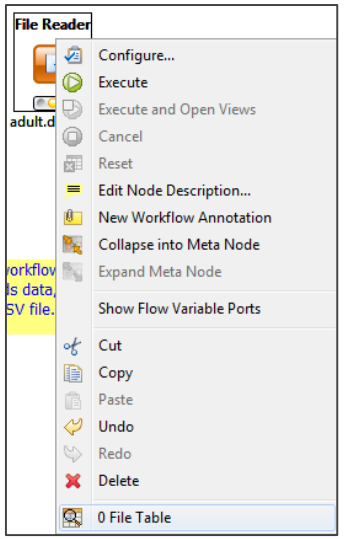
View the processed data

If the execution was successful (**green light**), you can see the processed data by right-clicking the node and selecting the last option in the context menu.

The option to view the processed data is the last item of the context menu (right-click menu) for all nodes with output data, but it takes on different names for different nodes.

The data table with the processed data is then shown.

2.14. Right-click the node and select the last option in the context menu to visualize the processed data



2.3. Read data from a file

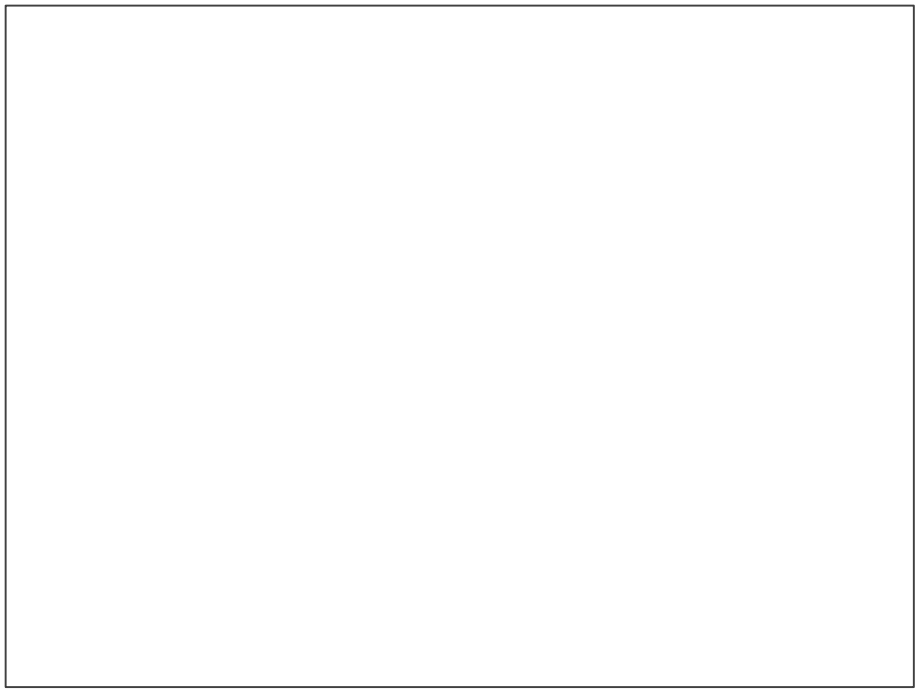
The first step in all data analysis works consists of reading data. Data is usually read from a file or from a database. In this chapter we describe how to read and write data from and to a text file. Reading and writing data from and to a database is described in section 3.7 “Database Operations”.

Create a “File Reader” node

In the “Node Repository” on the bottom left panel

- Expand the “IO” category and then the “Read” sub-category OR alternatively type “File Reader” in the search box in the “Node Repository” panel
- Drag and drop the “File Reader” node into the workflow editor
- If the “Node Description” panel on the right is enabled, it shows the description of the “File Reader” node.
- To activate the “Node Description” panel, go to the Top Menu, open “View” and select “Node Description”.

2.15. Create a "File Reader" node



Note. Below the newly created “File Reader” node you might notice a little yellow warning triangle. If you hover over it with the mouse, the following tooltip appears: “No Settings available”. This is because the File Reader node has not yet been configured (it needs at least the filename!). At the moment, the node is in the red traffic light state: not even configured.

Configure the “File Reader” node

- Double-click the node

OR

- Right-click the node and select “Configure”

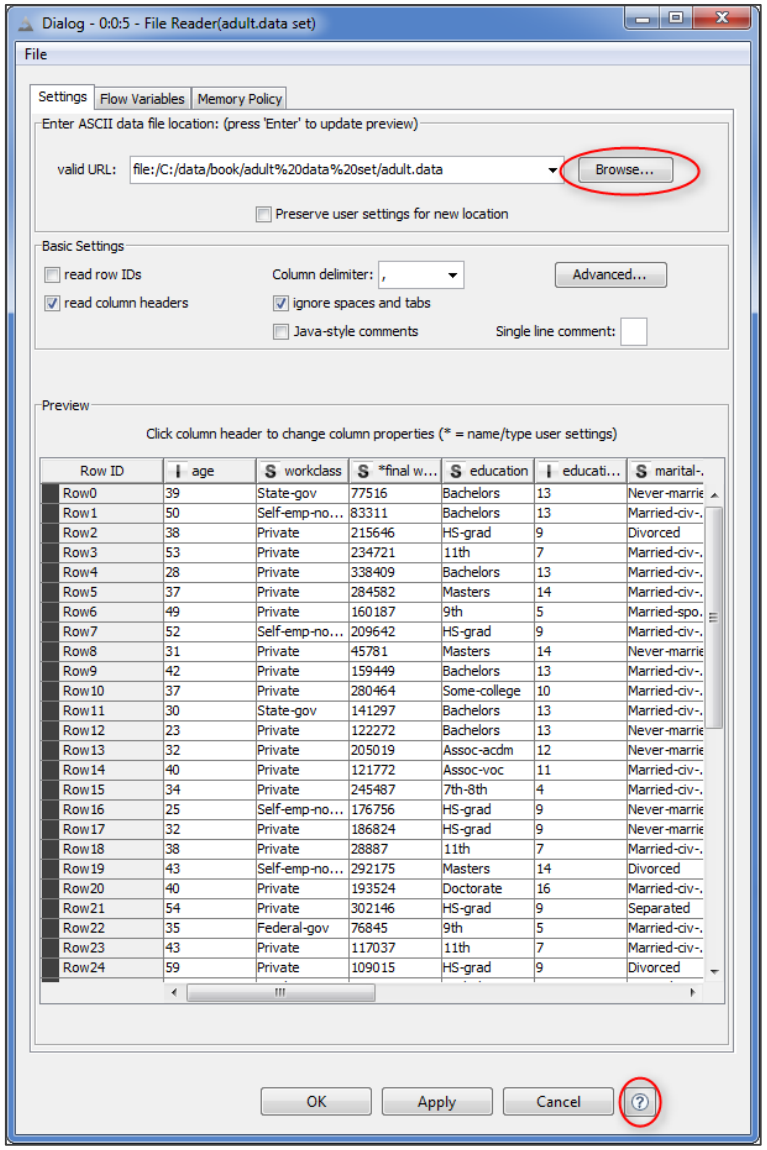
- Specify the file path in the configuration dialog, by typing or by using the “Browse” button. For this example, we used the adult.data file, downloadable from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Adult).
  If it is a comma or tab separated file, the “File Reader” node automatically detects the structure.

- Next, enable/disable the number of respective checkboxes according to the structure of the data in the file.

A preview of the data is available on the bottom part of the window and reports possible reading errors.

On the right of the acceptance/cancellation buttons in the bottom part of the window, you might notice a small button carrying a question mark icon. This is the help button and leads to a new window containing the node description.

2.16. "File Reader" node configuration window



Customize Column Properties

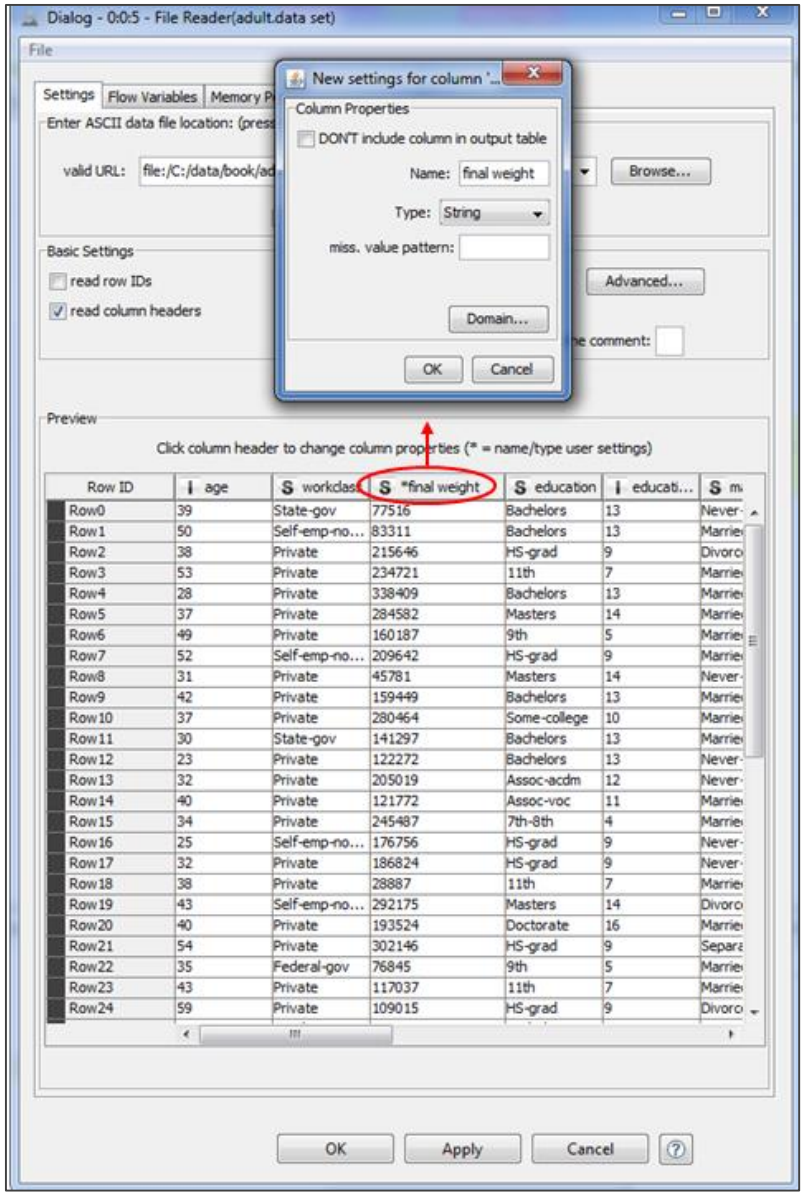
It is possible to customize the way that column data is read.

For example, let's read column "fnlwgt" as a String instead of as an Integer and set "final weight" as column header instead of "fnlwgt".

- Click the column header of column "fnlwgt"
- The window to customize the column properties opens

Here you can:

- Change the column name (= column header) from "fnlwgt" to "final weight"
- Change the column data type from Integer to String or Double
- Introduce a special character to represent missing values



Advanced Reading Options

Under the "Advanced" button you will find some more reading options. Each option tab is extensively described. I provide brief comments here on only three of them.

To read the adult.data file however we do not need to enable any of those options, as the adult.data file is pure ASCII.

"Quote Support" allows the definition of special quote characters.

Quote characters are essential because they define the start and the end of a special string. The default quote characters are \" and \'. Quoted strings spanning multiple lines are allowed if the corresponding flag is enabled.

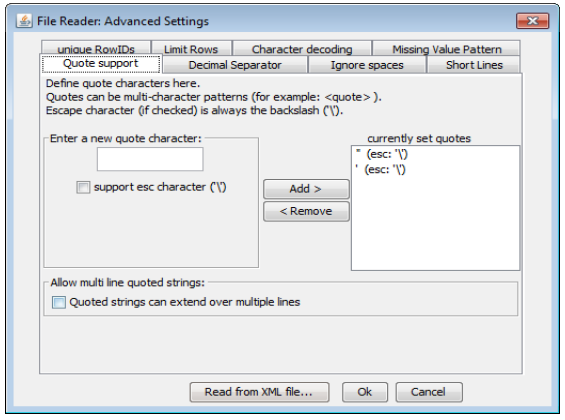
"Limit Rows" allows to read/skip only a limited number of rows.

If the first lines of a text file for example are just comments, you might want to skip them. The same, if only the first N rows are interesting and the rest is garbage like when reading a log file.

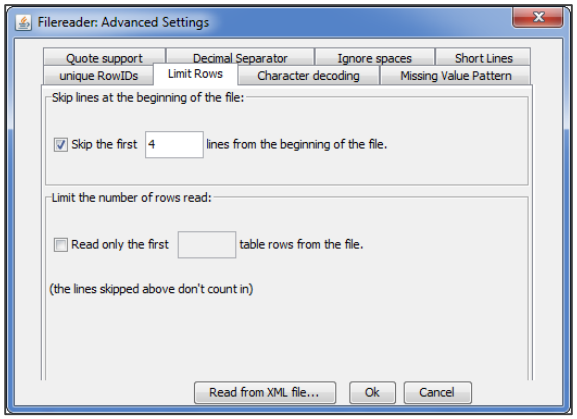
"Character Decoding" offers you the possibility of reading files encoded with a different locale than just ASCII (default).

Files with text in a different language than English need to have an encoding enabled, in order to be read properly.

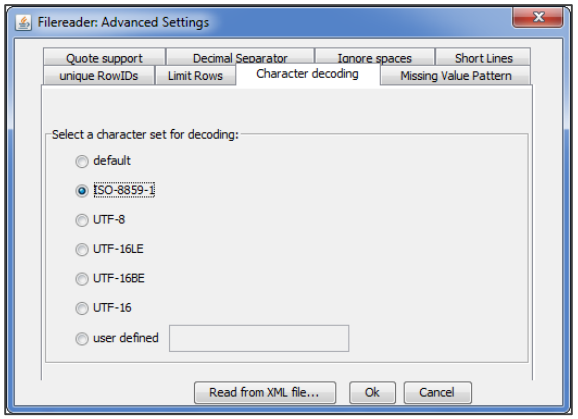
2.18. Advanced settings: Quote Support



2.19. Advanced settings: Limit Rows



2.20. Advanced settings: Character Decoding



Click "OK" at the bottom of the "Advanced Settings" window.

If you are satisfied with the data preview and your settings, click "OK" at the bottom of the "File Reader Configuration" window.

Note. After configuring the "File Reader" node, its state moves to the yellow traffic light.

Note. If a data folder is displayed in your "KNIME Explorer" panel, dragging and dropping a file with known extension (like .csv for example) into the workflow editor automatically creates the appropriate reader node, appropriately configured, in the workflow editor space.

We also need to give this node a meaningful name so that we can easily recognize what its task is in the workflow. The default name of our "File Reader" is "Node 1" because it is the first node created in the workflow. In order to change the node's name:

1. Click the "Node 1" label under the "File Reader" node
2. Enter the node's new name (for example "Adult data set")
3. Click elsewhere

We have now changed the name of the "File Reader" node from "Node 1" to "Adult data set".

In order to make the node read the file, we need to execute it. Proceed as follows:

- Right-click the node
- Select "Execute"

Note. If the reading process has no errors, the node switches its traffic light to green.

Note. On every configuration window you will find a tab, called "Flow Variables". Flow Variables are used to pass external variables from one node to another. However, we are not going to work with Flow Variables in this book, since they belong to a more advanced course on KNIME usage.

The "IO" -> "Read" category in the "Node Repository" contain a number of other nodes to read files in different formats, like Excel files, CSV files, KNIME proprietary format files, and more. The "IO"/"File Handling" category has additional nodes to read special formats and special files, like for example zip files, remote files, etc….

2.4. KNIME data

If the node execution was successful, you can now see the resulting data.

- Right-click the "File Reader" node
- Select option "File Table"

A table with the read data appears. Let's have a look at this table now to understand how data is structured inside KNIME.

First of all, data in KNIME is organized as a **table**.

Each row is identified by a **Row ID**. By default, Row IDs are strings like "Row n" where "n" is a progressive number. But RowIDs can be forced to be anything, with the only condition that they must be unique. Not unique RowIDs produce an error.

Columns are identified by **column headers**. If no column headers are available, default column headers like "Col n" - where "n" is a progressive number - are assigned by KNIME. In adult.data file column headers were included. We enabled the checkbox "Read column headers" in the configuration window of the "File Reader" node and we now have a header for each column in the final data table. Even column headers need to be unique. If a column header occurs more than once, KNIME adds a suffix "(#n)" (n = progressive number) to each multiple occurrence of the column header.

Each column contains data with a set **data type**. A few common data types are defined in KNIME:

- Double ("**D**")
- Integer ("**I**")
- String ("**S**")
- Date/time (calendar + clock icon)
- Unknown ("**?**")
- Other specific domain related types (like Document in the text processing extension or Smiles in chemistry extensions)

Date/time type is never generated by reading from a file. Dates are read from a text file just as strings. You need a "Time To String" node in the "Time Series" category to convert a String into a date/time data column.

Unknown type refers to columns with mixed data types, for example String and Integer, or with an indefinable data type, for example with all missing values.

Missing values are data cells with a special "missing value" status and are displayed by default with a question mark ("?"), unless the display character for the missing values was set otherwise in the "File Reader" node configuration.

Note. Missing values are **represented** by default by question marks. They are not question marks. Question marks in the text file are read as question marks, but they are not missing data. Missing values could be represented by anything else as defined in the configuration window of the "File Reader" node.

KNIME data structure

Data in KNIME is organized as a **table** with a fixed number of columns.

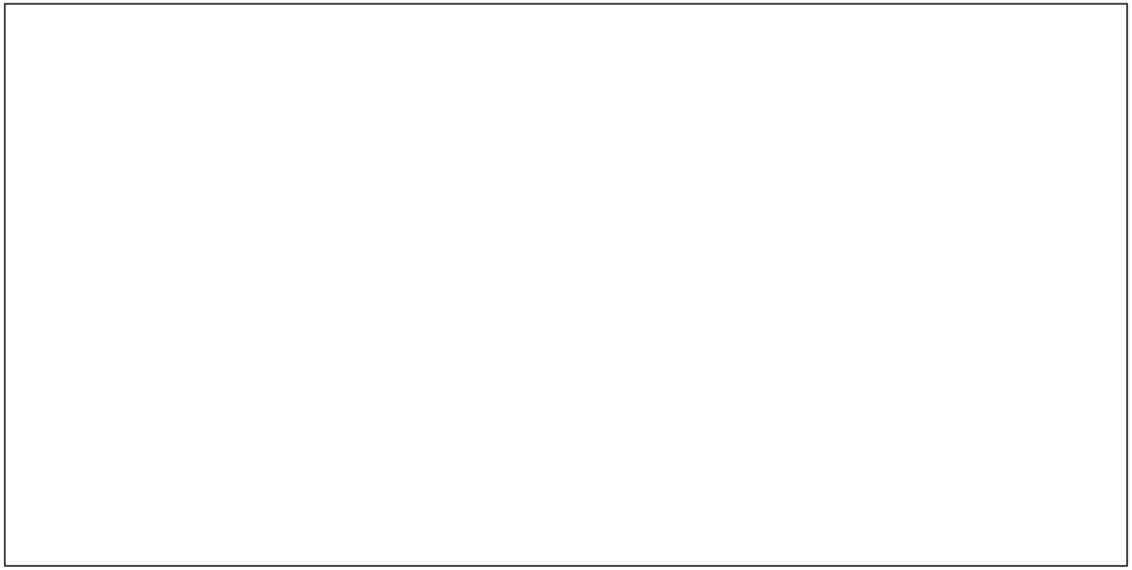
Each row is identified by a **Row ID**.

Columns are identified by **column headers**.

Each column represents a **data type**:

- Double ("**D**")
- Integer ("**I**")
- String ("**S**")
- Date/time (calendar + clock icon)
- Unknown ("**?**")
- Other domain related types

2.21. The KNIME data structure



Clicking the header of a data column allows to sort the data rows in an ascending / descending order.

Right-clicking the header of a data column allows to visualize the data using specific renderers. For Double/Integer data, for example, the "Bars" renderer displays the data as bars with a proportional length to their value and on a red/green heatmap.

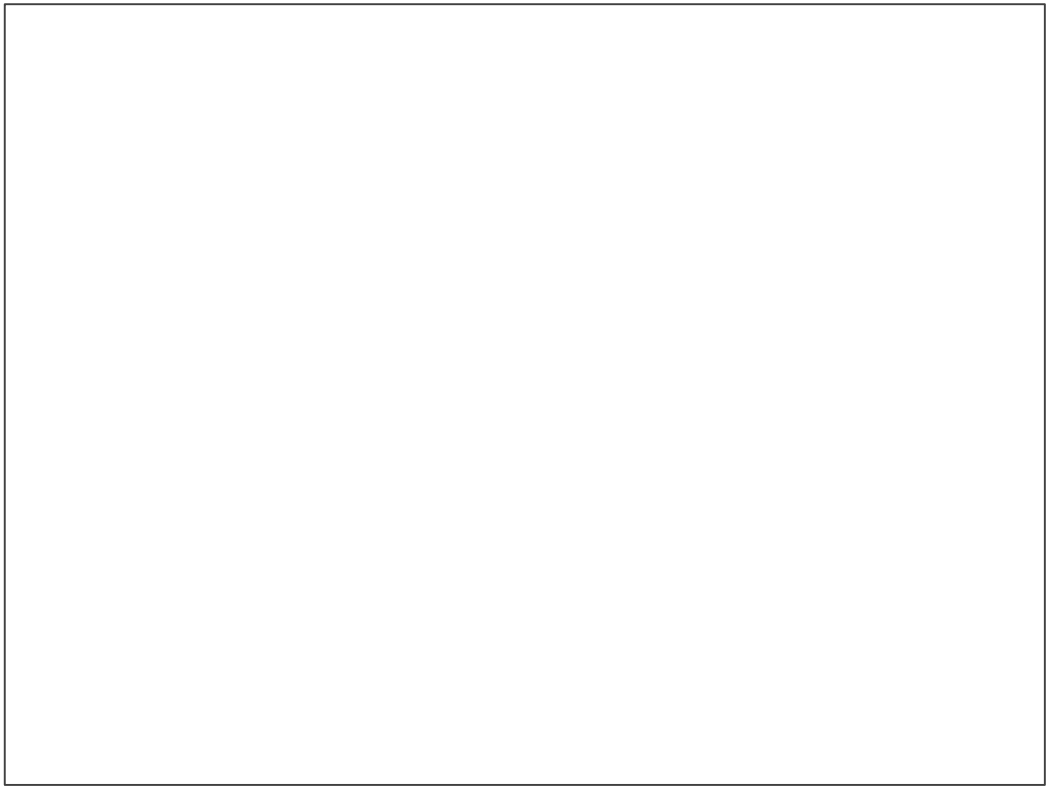
2.5. Filter Data Columns

In the next step, we want to filter out the column "final weight" from the read data set. In the "Node Repository" panel, on the bottom left, there is a whole category called "Data Manipulation" with nodes dedicated to managing the data structure. This category includes operations on columns, rows, and on the full data matrix.

Create a “Column Filter” node

- In the “Node Repository” panel on the bottom left, find the node “Column Filter” under “Data Manipulation” -> “Column” -> “Filter” or search for “Column Filter” in the search box.
- Drag and drop (or double-click) the “Column Filter” node from the “Node Repository” panel to move it into the workflow editor
- The description for this node appears in the “Node Description” panel on the right.
- Connect the “Column Filter” node with the previous node (in our workflow, the “File Reader” node named “adult.data set”)

2.22. Creating and configuring a "Column Filter" node: the “Manual Selection” option



To configure the node then:

- Double-click the node or right-click the node and select “Configure”
- The configuration window opens. The node’s configuration window contains all settings for that particular node.
- Set the node configuration settings
- Click “OK”

Configure the “Column Filter” node

The first setting in the configuration window is the type of filtering (Fig. 2.22). You can select and retain columns manually, by type, and by name, according to the radio buttons at the top of the configuration window.

Manual Selection

If the “Manual Selection” option is selected, the configuration window shows 2 sets of columns (Fig. 2.22):

- The columns to be included in the data table (**“Include”** set on the right)
- The columns to be excluded from the data table (**“Exclude”** set on the left)

Two **“Search”** buttons allow to search for the specific column.

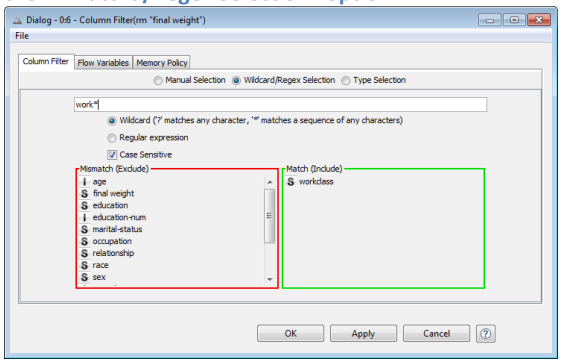
You can add and remove columns from one set to the other using the buttons **“Add”** and **“Remove”**.

- **“Enforce Inclusion”** keeps the “Include” set fixed. If one more input column is added from the previous node, this new column is automatically inserted into the “Exclude” set.
- **“Enforce Exclusion”** keeps the “Exclude” set fixed. If one more input column is added from the previous node, this new column is automatically inserted into the “Include” set.

Wildcard/Regex Selection

In case the “Wildcard/Regex Selection” option is enabled, the configuration window presents a textbox to edit the desired wildcard or regular expression (Fig. 2.23).
The radio button below the textbox specifies whether this is a regular expression or a wildcard expression. An additional checkbox enables a case sensitive match.
Columns with name matching the expression will be included in the output data table.

2.23. Configuration window of "Column Filter" node: the “Wildcard/Regex Selection” option

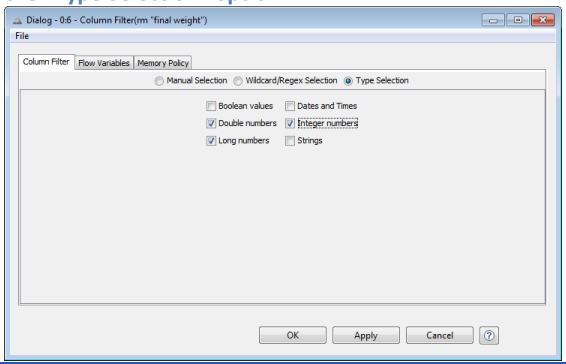


Type Selection

If the “Type Selection” option is enabled, you are presented with a series of checkboxes about the column types to keep in the output data table (Fig. 2.24).

Selecting Double, Integer, and Long Numbers, for example, will keep all numerical columns in the node’s output table.

2.24. Configuration window of "Column Filter" node: the “Type Selection” option

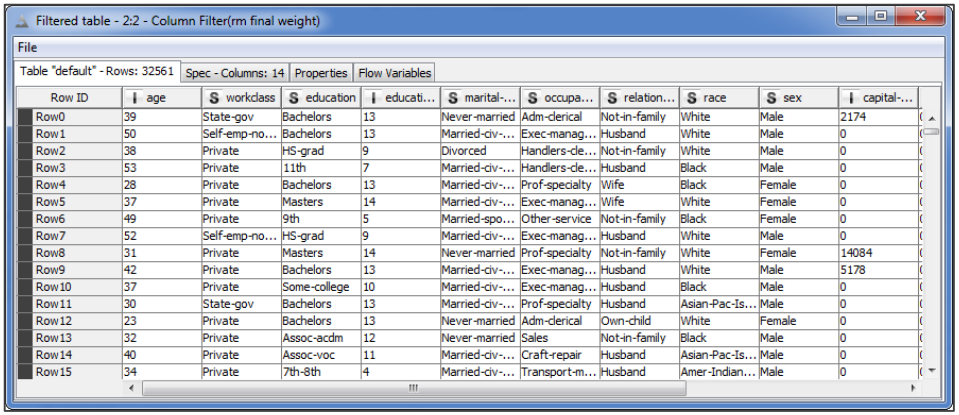


Remember this column selection frame, because we will find it again in all nodes requiring column selection.

In our example workflow, we want to remove only the “final weight” column. We set the column filter mode to “manual Selection” and to “Enforce Exclusion”, because we want to keep all possible new input columns besides “final weight”. After completing the configuration, right-click the “Column Filter” node and give it a name. We named it “rm ‘final weight’”. Right-click the node and select “Execute” to run the column filter.

To see the final processed data, right-click the “rm final weight” node and select “Filtered Table”. The column “final weight” should no longer be in the data table.

2.25. The column filtered table does not contain column "final weight"



Row ID	age	workclass	education	educati...	marital-...	occupa...	relation...	race	sex	capital-...
Row0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174
Row1	50	Self-emp-no...	Bachelors	13	Married-civ-...	Exec-manag...	Husband	White	Male	0
Row2	38	Private	HS-grad	9	Divorced	Handlers-cle...	Not-in-family	White	Male	0
Row3	53	Private	11th	7	Married-civ-...	Handlers-cle...	Husband	Black	Male	0
Row4	28	Private	Bachelors	13	Married-civ-...	Prof-specialty	Wife	Black	Female	0
Row5	37	Private	Masters	14	Married-civ-...	Exec-manag...	Wife	White	Female	0
Row6	49	Private	9th	5	Married-spo...	Other-service	Not-in-family	Black	Female	0
Row7	52	Self-emp-no...	HS-grad	9	Married-civ-...	Exec-manag...	Husband	White	Male	0
Row8	31	Private	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084
Row9	42	Private	Bachelors	13	Married-civ-...	Exec-manag...	Husband	White	Male	5178
Row10	37	Private	Some-college	10	Married-civ-...	Exec-manag...	Husband	Black	Male	0
Row11	30	State-gov	Bachelors	13	Married-civ-...	Prof-specialty	Husband	Asian-Pac-Is...	Male	0
Row12	23	Private	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0
Row13	32	Private	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0
Row14	40	Private	Assoc-voc	11	Married-civ-...	Craft-repair	Husband	Asian-Pac-Is...	Male	0
Row15	34	Private	7th-8th	4	Married-civ-...	Transport-m...	Husband	Amer-Indian...	Male	0

2.6. Filter Data Rows

Now let’s retain only those data rows that refer to people born outside of the United States that is only those rows with “native-country” other than “United States”.

Create a “Row Filter” node

In the “Node Repository” panel on the bottom left, open the node category “Data Manipulation” and navigate to the node “Row Filter” in `“Data Manipulation” -> “Row” -> “Filter”` or search for “Row Filter” in the search box.

Drag and drop (or double-click) the “Row Filter” node in the “Node Repository” to create a new instance in the workflow editor panel.

The task and settings description for this node can be found in the “Node Description” panel on the right or clicking the help button in the configuration window at the right of the “Cancel” button.

Connect the “Row Filter” node with the “Column Filter” node named “rm final weight ”.

Configure the “Row Filter” node

- Double-click the “Row Filter” node to open its configuration window.

The node implements three filter criteria, which are shown on the side. Each of these criteria can be used to **include** or to **exclude rows**.

- Implement your row filter criterion
- Click “OK”

Row Filter criteria

- Select rows **by attribute value** (pattern matching)
- Select rows **by row number**
- Select rows **by RowID** (pattern matching on RowID)

Row filter criteria

By attribute value

All rows, for which the value in a given column matches a given pattern, are filtered out or kept.

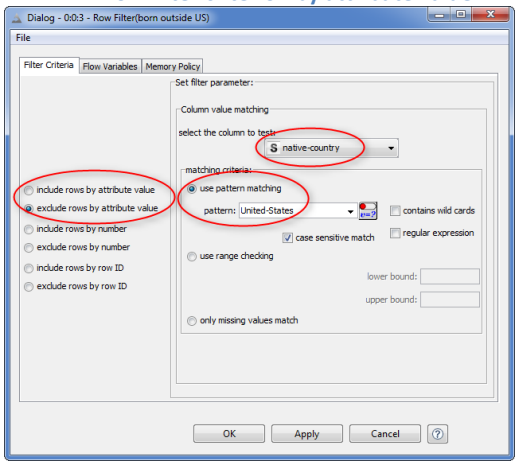
After you "select the column to test", you need to define the matching mode.

For **String values**, "use pattern matching" requires the given pattern to be either entered manually or selected from a list of pre-defined patterns.
A matching value with wildcards * (for example "United*") or with a regular expression is also possible.

For **Integer values**, "use range checking" requires a lower boundary and/or an upper boundary, which will be the same value if the condition is equality.

For **Missing values**, choose the last matching option.

2.27. Row filter criterion by attribute value

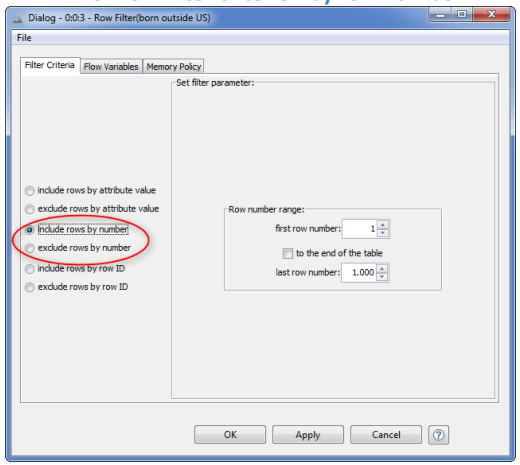


By row number

If you know where your desired or undesired rows are, you can just enter the **row number range** to be filtered out.

For example, if I know that the first 10 rows are garbage, I would select the filter criterion "exclude row by number" and set the row number range 1-10.

2.28. Row filter criterion by row number

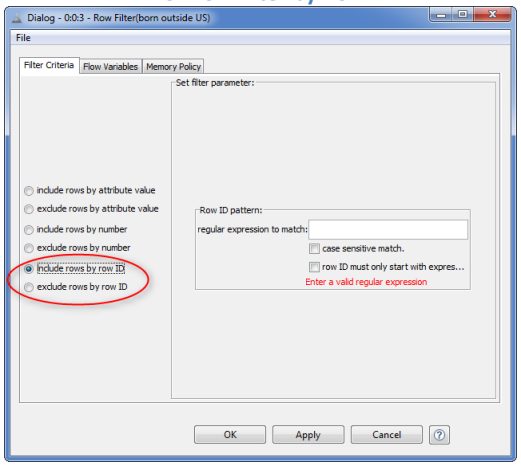


By RowID

A special row filter by attribute value runs on the RowIDs.

Here the matching pattern is given by a regular expression. The regular expression has to match the whole RowID or just its beginning,

2.29. Row filter by RowID



In order to retain all rows with data referring to people born outside of the United States, we need to:

- Set filter mode “exclude row by attribute value”
- Set the column to test to “native-country”
- Enable “use pattern matching”, because it is a string comparison
- Set pattern “United States”

We have just implemented the following filter criterion: `native-country != "United States"`

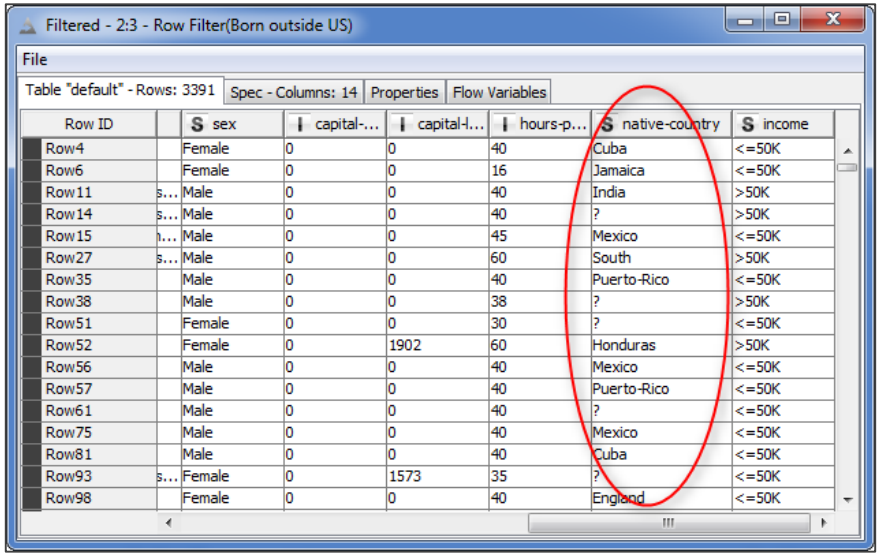
- Give the “Row Filter” node a meaningful name. We named it “born outside US”. The name of a node is important for documentation purposes. Since KNIME is a graphical tool, it is easy to keep an overview of what a workflow does, if the name of each node gives a clear indication of its task.

- Right-click the node and select “Execute” to run the row filter

To see the final processed data, right-click the node “born outside US” and select “Filtered”.

There should be no “United States” in column native-country.

2.30. The row filtered table has no pattern "United States" in column "native-country“



Row ID		sex	capital-...	capital-l...	hours-p...	native-country	income
Row4		Female	0	0	40	Cuba	<=50K
Row6		Female	0	0	16	Jamaica	<=50K
Row11	s...	Male	0	0	40	India	>50K
Row14	s...	Male	0	0	40	?	>50K
Row15	n...	Male	0	0	45	Mexico	<=50K
Row27	s...	Male	0	0	60	South	>50K
Row35		Male	0	0	40	Puerto-Rico	<=50K
Row38		Male	0	0	38	?	>50K
Row51		Female	0	0	30	?	<=50K
Row52		Female	0	1902	60	Honduras	>50K
Row56		Male	0	0	40	Mexico	<=50K
Row57		Male	0	0	40	Puerto-Rico	<=50K
Row61		Male	0	0	40	?	<=50K
Row75		Male	0	0	40	Mexico	<=50K
Row81		Male	0	0	40	Cuba	<=50K
Row93	s...	Female	0	1573	35	?	<=50K
Row98		Female	0	0	40	England	<=50K

2.7. Write Data to a File

Now we want to write the processed data to a file. There are many nodes that can write to a file. Let's choose the easiest and most standard format: the CSV (Comma Separated Values) format.

Create a "CSV Writer" node

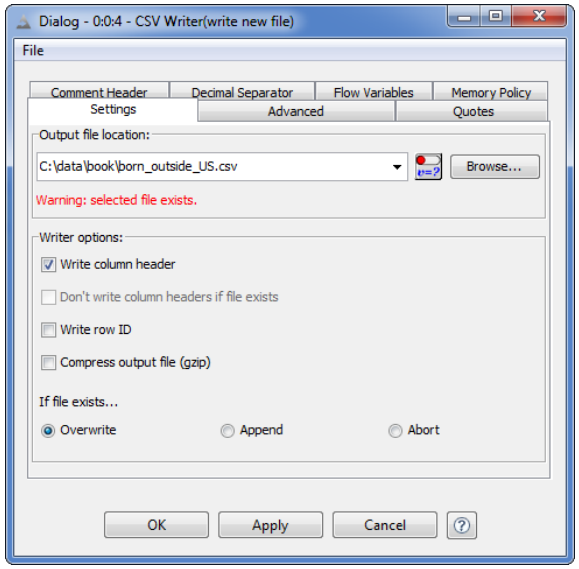
In the "Node Repository" in the bottom left panel:

- Expand category "IO"/"Write" or search for "CSV Writer" in the search box in the "Node Repository"
- Drag and drop (or double-click) the node "CSV Writer" to create a new node in the workflow editor
- If the "Node Description" panel on the right is enabled, it fills up with the description of the "CSV Writer" node. To activate the "Node Description" panel, go to the Top Menu, open "View" and select "Node Description".
- Right-click the node and select "Configure" to open its configuration window.

2.31. Create a "CSV Writer" node

Configure the "CSV Writer" node

2.32. Configuring a "CSV Writer" node



"Settings" is the most important tab of this configuration window. It requires:

- The path of the output file
- A few options, such as:
    1. Write column headers and/or RowID in output file
    2. Writing mode if file already exists
        - Overwrite
        - Append
        - Abort (does not write to file)

There are a few more tabs in this configuration window:

- **"Advanced"** allows specification of a different separation character other than "," and of a different missing value character.
- **"Quotes"** is for setting quote characters other than the default.
- **"Comment Header"** is to write a header with comments on top of the data.
- **"Decimal Separator"** is to specify a new decimal separator (default is ".")
- **"Memory Policy"** offers a few strategies to handle memory and data. It comes in useful when the workflow deals with a large amount of data.
- In this book we do not investigate the tab "Flow variables".

Note. Writing in mode "Append" can be tricky, because it just appends the new data to an existing file without checking the data structure nor matching the column by name. So, if the data table structure has changed, for example because of new or deleted columns, the output CSV file is not consistent anymore.

You want to select “Abort” as writing mode, in order to avoid overwriting the existing file.

Let’s now change the node’s name:

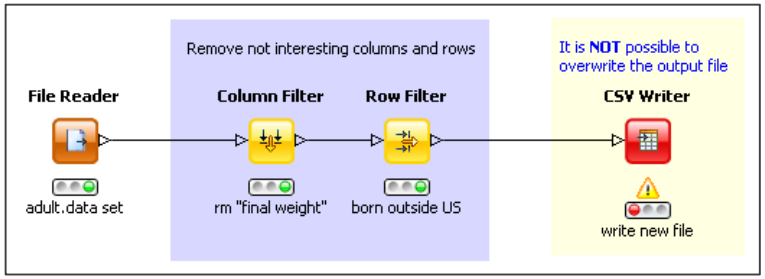
3. Click the node label under the node
4. Enter the node’s new name (for example “write new file”)
5. Click elsewhere
6. Right-click the node and select “Execute”

At this point we also add a few annotations to make even clearer what each node or group of nodes does.

We have created our first workflow to read data from a file, reorganize rows and columns, and finally write the data to an output file.

This is how the final workflow looks like.

2.33. Workflow "my_first_workflow"



2.8. Exercises

Exercise 1

In a workflow group “Exercises” under the existing workflow group “Chapter2” create an empty workflow “Exercise1”.

Workflow “Exercise1” should perform the following operations:

- Read file data1.txt (from the “Download Zone”) with column “ranking” as String and named “marks”;
- Remove initial comments from data read from file;
- Remove column “class”
- Write final data to file in CSV format (for example with name “data1_new.csv”) using character “;” as separator

Enter a short description for all nodes in the workflow.

Save and execute workflow “Exercise1”. Execution must be without errors (green lights for all nodes).

Solution to Exercise 1

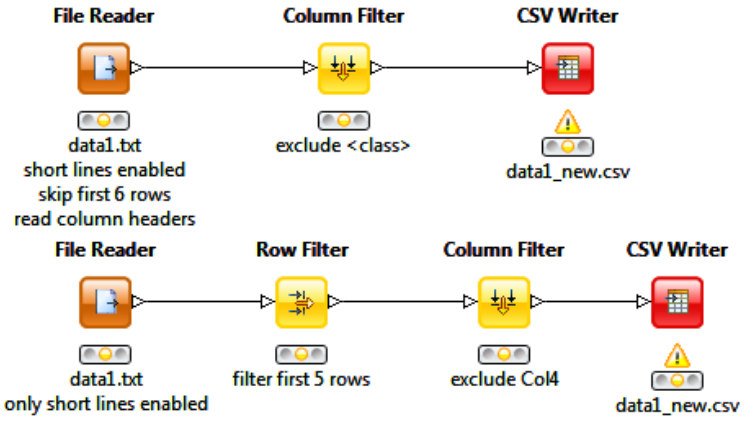
The file has some comments at the very beginning, which of course do not have the same length as the other lines in the file.

First, you need to enable the option “allow short lines” in the “Advanced” tab.

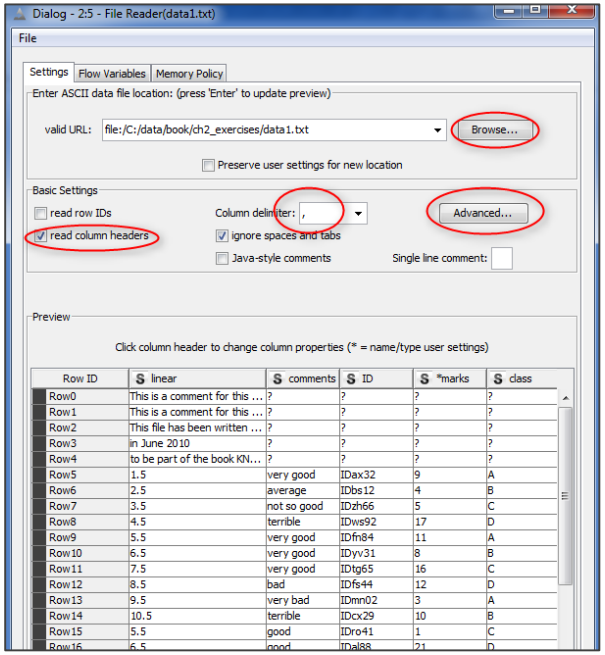
Then you can either enable the “Skip the first 6 lines …” in the “Limit Rows” tab of the “Advanced” window and the “read column headers” option in the basic configuration window or you can use a “Row Filter” node to exclude the first 5 rows of the read data. The difference between these two approaches is in the reading of the column headers from the file.

The “File Reader” node in both workflows changes the name of the 4th column to “marks”, as required.

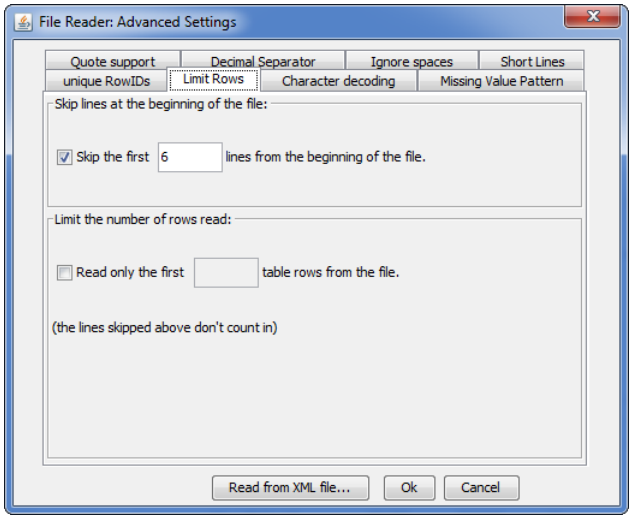
2.34. Exercise 1: The two possible workflows



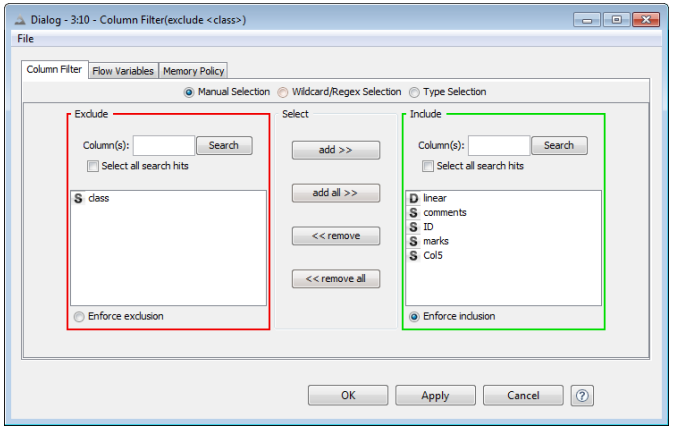
2.35. Exercise1: “File Reader” configuration settings



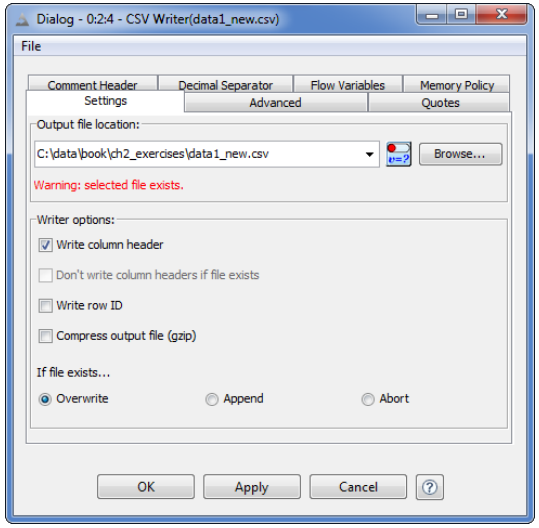
2.36. Exercise 1: "File Reader" “Limit Rows” tab configuration



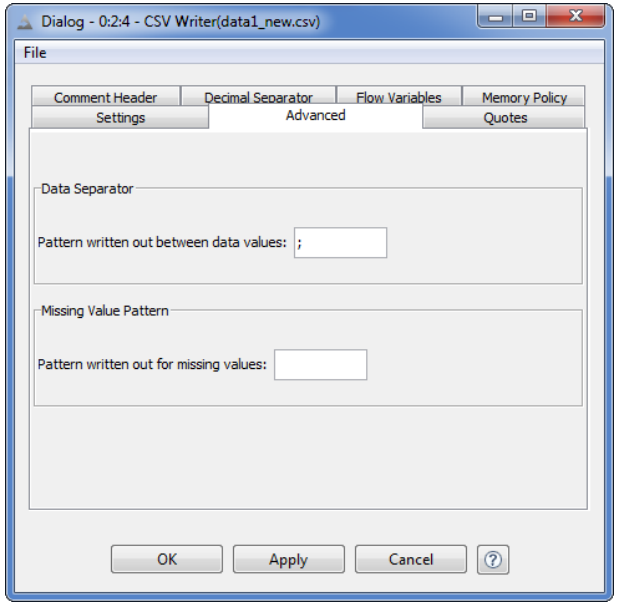
2.37. Exercise 1: “Column Filter” configuration options



2.38. Exercise 1: “CSV Writer” configuration settings



2.39. Exercise 1: "CSV Writer": "Advanced" tab



Exercise 2

In the workflow group "Chapter2\Exercises" create a workflow "Exercise2" to perform the following operations:

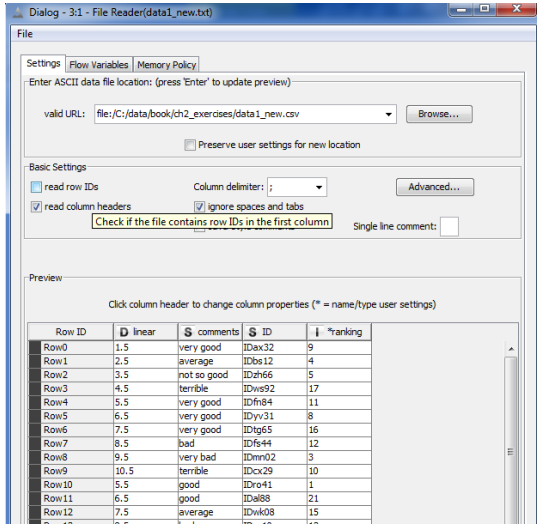
- Read the CSV file written in Exercise1 ("data1_new.csv") and rename column "marks" to "ranking"
- Filter out rows with comments = "average" in data column "ranking"
- Exclude Integer type columns
- Write final data to file in "Append" mode and with a tab as a separating character

Rename all nodes where necessary. Save and execute workflow "Exercise2".

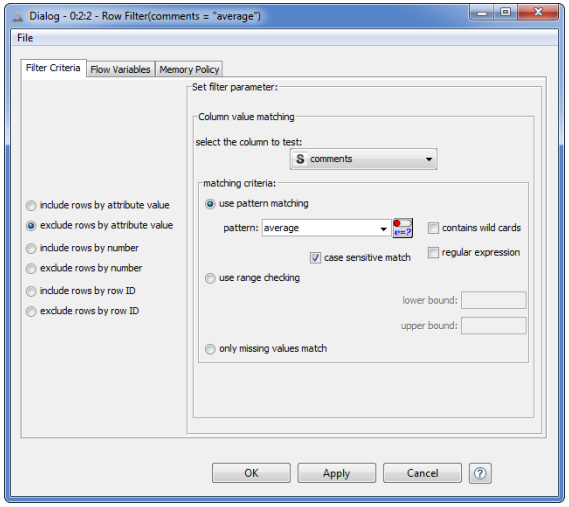
Solution to Exercise 2

We recycled the workflow structure from the workflow created in Exercise 1. That is we did a "Copy and Paste" operation (Ctrl-C, Ctrl-V) on the whole "Exercise 1" workflow from the workflow editor for "Exercise 1" into the workflow editor for "Exercise 2".

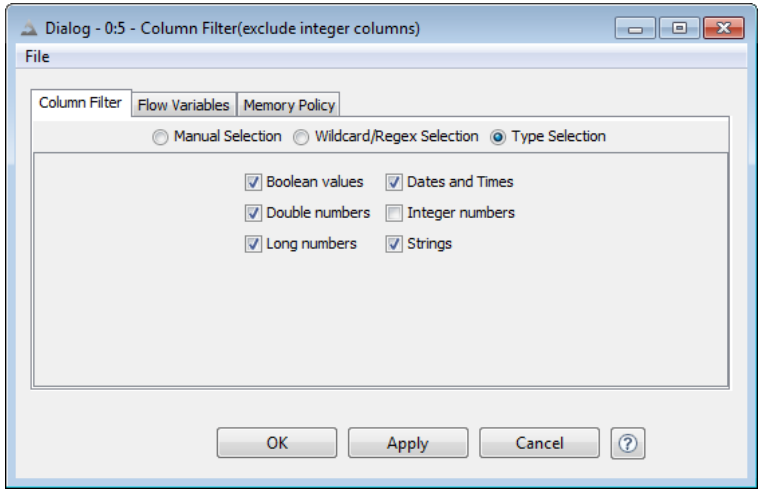
2.40. Exercise 2: “File Reader” configuration options



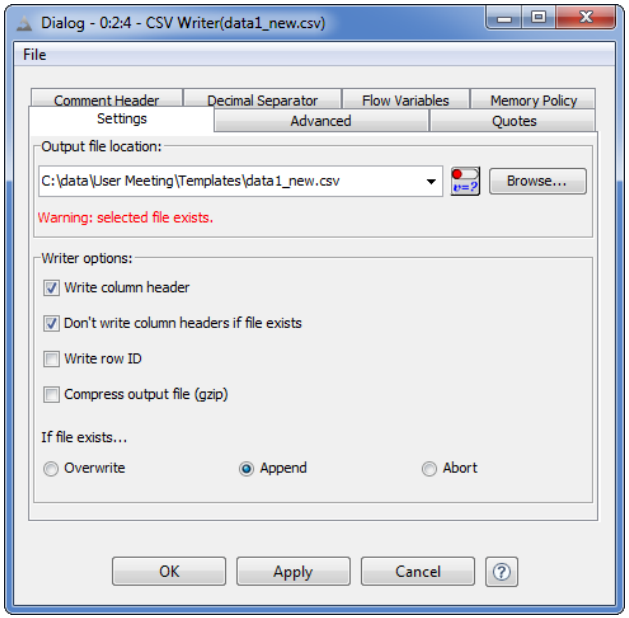
2.41. Exercise 2: “Row Filter” configuration options



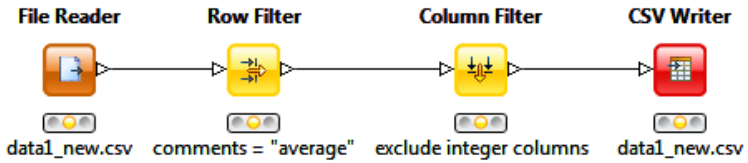
2.42. Exercise 2: “Column Filter” configuration options



2.43. Exercise 2: “CSV Writer” configuration options



2.44. Exercise 2: The workflow



Note. After copying the "File Reader" from Exercise 1, you need to disable the option "**Limit Rows**" in the "Advanced Settings" window, because this file has no initial comments.

Note. Notice the **yellow triangle** under the "Column Filter" node. This is a warning message that comes from the copy of the workflow and that remains even when the node has the green light. Hovering over the yellow triangle shows the warning message *"Some columns are not available: marks"*. This is correct: column "marks" is not there anymore, because we have renamed the column "marks" as "ranking". However, the column filter still works; only a warning message is issued. If we open the configuration window of the column filter, we see that the column "ranking" was automatically inserted into the "Exclude" set. This is because the option "Enforce Inclusion" was enabled. Clicking the "OK" button accepts the current configuration settings and makes the warning yellow triangle disappear.

Note. We saved the data in **"Append" mode** into the CSV file. The data from Exercise2 has only 3 columns, while the existing data in the file has 4 columns. The "CSV Writer" node does not check the consistency of the number and the position of the columns to be written with the number and the positions of the existing columns. It is then possible to write inconsistent data to a file. You need to be careful when working in "Append" mode with a "CSV Writer" node.

Chapter 3. My first data exploration

3.1. Introduction

This chapter describes two workflows: “WriteToDB” and “my_first_data_exploration”. “WriteToDB” writes data into a database and “my_first_data_exploration” reads the same data from the database and allows you to graphically explore the data.

The goal here is to become familiar with:

- nodes and options for database handling
- the “Data Views” category containing nodes for graphical data exploration
- a few more column operation nodes, like nodes for string manipulation and missing value handling

We start from the very well known “Iris” Dataset from the UCI Machine Learning Repository web site (http://archive.ics.uci.edu/ml/datasets/Iris) to build the data to write into the database. The Iris dataset describes a number of iris plants by means of 4 attributes:

- the sepal length
- the sepal width
- the petal length
- the petal width

The plants described in the data set belong to three different iris classes: Iris setosa, Iris versicolor, and Iris virginica.

3.1. The iris data set

File Table - 3:1 - File Reader(iris.data set)

File

Table "iris.data" - Rows: 150 | Spec - Columns: 5 | Properties | Flow Variables

Row ID	D Col0	D Col1	D Col2	D Col3	S Col4
Row0	5.1	3.5	1.4	0.2	Iris-setosa
Row1	4.9	3	1.4	0.2	Iris-setosa
Row2	4.7	3.2	1.3	0.2	Iris-setosa
Row3	4.6	3.1	1.5	0.2	Iris-setosa
Row4	5	3.6	1.4	0.2	Iris-setosa
Row5	5.4	3.9	1.7	0.4	Iris-setosa
Row6	4.6	3.4	1.4	0.3	Iris-setosa
Row7	5	3.4	1.5	0.2	Iris-setosa
Row8	4.4	2.9	1.4	0.2	Iris-setosa
Row9	4.9	3.1	1.5	0.1	Iris-setosa
Row10	5.4	3.7	1.5	0.2	Iris-setosa

This dataset has been used for many years as a standard for classification. The three classes are not all linearly separable. Only two of the three iris classes can be separated by using a linear threshold on two of the four numeric attributes. For the third class we need to use something more sophisticated than a linear separation. The separation of the first two classes can be clearly seen by using graphic plots. This is the reason why we use this dataset to illustrate the KNIME nodes for visual data exploration.

We will use this chapter also to explore string manipulation and how to create new rule-based values from the existing columns' values.

We create now a new workflow group named "Chapter3" in the "KNIME Explorer" panel. This workflow group will contain all workflows created in this chapter of the book. Under workflow group "Chapter3" we create two empty workflows: "WriteToDB" and "my_first_data_exploration". As we said at the beginning of this section, "WriteToDB" illustrates how to build a new dataset and how to write it into a database, while "my_first_data_exploration" describes how to perform a visual data exploration of the data. The workflow group and its workflows can be found in the "Download Zone".

Let's start with shaping the "WriteToDB" workflow.

After reading the Iris dataset file (iris.data) with a "File Reader" node, we get the data table in figure 3.1. We name the "File Reader" node "iris.data set" for a quick overview of the node task.

Note. The iris dataset file does not contain any names for the data columns. The "File Reader" node then assigns to each column a default name like "Col0", "Col1", "Col2", "Col3", and "Col4". Besides "Col4" where we can see that this is the iris class, we need to read the file specifications in file "iris.names" to understand which column represents which numerical attribute.

3.2. Replace Values in Columns

After reading the description of the iris data set in the iris.name file, we discover that the five columns are organized as follows:

1. Sepal length in cm
2. Sepal width in cm
3. Petal length in cm
4. Petal width in cm
5. class

And that there are no missing values in the data set.

Thus, the first step is to rename the data set's columns, in order to be able to talk clearly about what we are doing on the data. KNIME has a node "Column Rename" to be used exactly for this purpose.

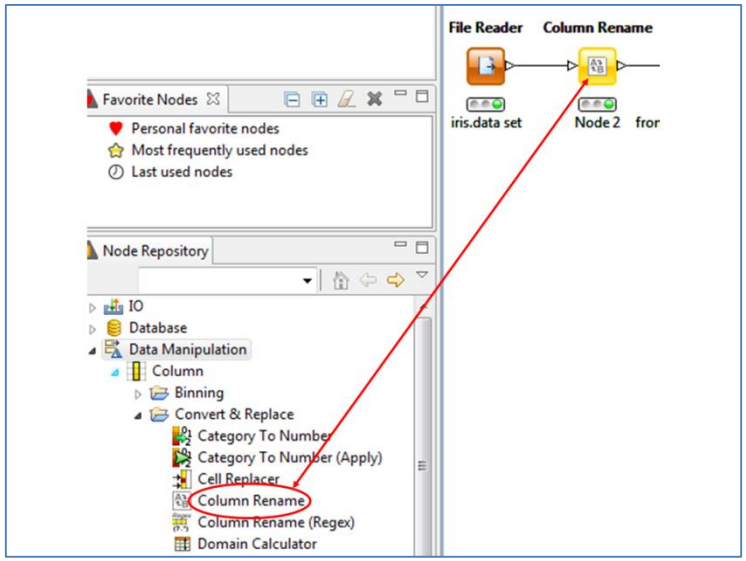
Column Rename

The node “Column Rename” can be found in the “Node Repository” panel under:

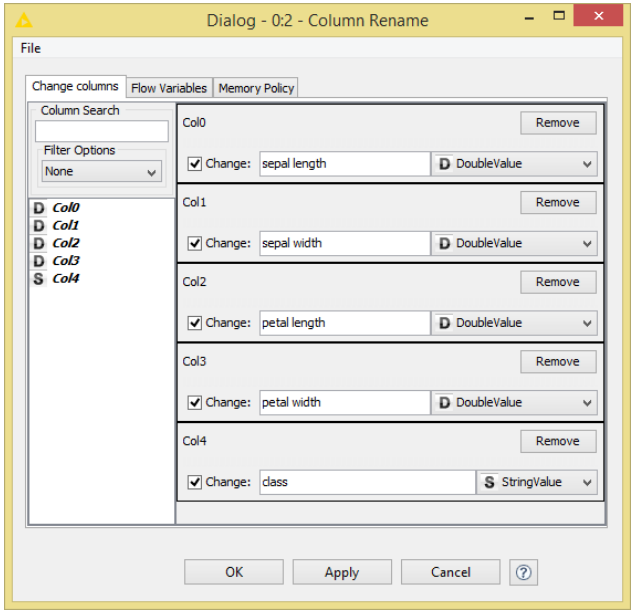
```
“Data Manipulation” -> “Column” -> “Convert & Replace”
```

The “Column Rename” node allows for the renaming of columns in the input data table.

3.2. Create a "Column Rename" node



3.3. Configuration dialog of the "Column Rename" node



The configuration dialog requires:

- To select the columns on which to operate
- to flag the columns whose name or type needs changing
- the new column names
- optionally the new column types

We created a “Column Rename” node and we connected it to the “File Reader” node “iris.data”. We then assigned new names to the data table columns according to the” Iris.name” specification file and we run the “Execute” command.

Let’s suppose now that the iris names in the “class” column are too long or too complex for our task. We would like to have just class numbers: “class 1”, “class 2”, and “class 3”. That is, we would like to add a column where “Iris-setosa” from column “class” is translated into “class 1”, “Iris-versicolor” into “class 2”, and finally all remaining instances belong to a “class 3”.

KNIME has a very practical node: the “Rule Engine” node. This node defines a set of rules on the values of the input data columns and generates new values according to the defined rule set. The new values can form a new column to append to the existing ones in the input data table or replace an existing data column.

The rule set that we would like to implement in this case is the following:

IF class = “Iris-setosa”	THEN	class 1
IF class = “Iris-versicolor”	THEN	class 2
ELSE		class 3

The “Rule Engine” node uses the following syntax to express this rule set:

```
$class$ = "Iris-setosa" => "class 1"
$class$ = "Iris-versicolor" => "class 2"
TRUE => "class 3"
```

Where $class$ indicates values in input data column “class”, “Iris-setosa” is the match value for the “=” operator, “=>” introduces the consequent of the rule, and “class 1” is the consequent value.

Note. String values need to be encapsulated in between quotation marks to be correctly interpreted as strings by the “Rule Engine” node.

The final keyword “TRUE” represents the ELSE value in our list of rules, i.e. the value that is always true if no other rule is applied first.

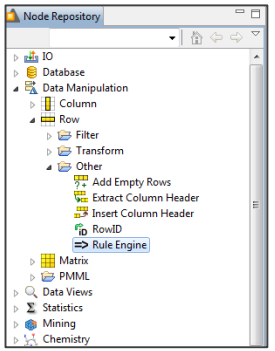
Note. To insert a new constant value in a data column, just use TRUE => <new constant value> with no other rule.

Rule Engine

The node “Rule Engine” is located in the “Node Repository” panel in the category `“Data Manipulation” -> “Row” -> “Other”.`

It defines a set of rules and creates new values based on the values in the existing columns.

3.4. Location of the "Rule Engine" node in the "Node Repository" panel



3.5. The Configuration window for the "Rule Engine" node

The configuration dialog includes:

- The list of available input data columns
- The list of available functions and operators
- A description panel to describe the usage and task of the selected function/operator
- A rule editor where to create the set of rules
- The option to create a new column in the output data table or to replace an existing one
- The list of available flow variables. However, Flow Variables are considered an advanced concept and we will ignore them in this book.

Column List

The first panel on the left upper corner of the “Rule Engine” configuration window shows all available columns from the input data table. Those are the columns our set of rules is going to work on.

Flow Variable List

The panel right below the “Column List” panel contains all flow variables available to the node. However, Flow Variables are not treated in this book and we will ignore flow variables when setting up our set of rules.

Function

The “Function” panel contains a list of functions and logical operators available to create the rule set. The “Category” menu on top of the “Function” list, allows to reduce the function list to a smaller subset.

Description

If a function or an operator is selected in the “Function” list, this panel provides a description of its task and usage.

Expression

The “Expression” panel is the rule editor. Here you can type your set of rules. If you need to involve a data column or a function, just double-click the desired item in the respective panel and it will appear in the rule editor.

Every rule consists of a condition including a function or an operator and of a consequence value. The symbol “=>” leads the condition to the consequence value, like: <condition> => <consequence value>. “TRUE” in the last rule leads to the default value, when none of the previous conditions apply. The rule can be edited and changed at any time.

To build our rule set, we typed in the set of rules described above at page 73.

Append Column / Replace Column

At the bottom of the configuration window, there are the options to choose whether to create a new data column or replacing an existing one. The default option is “Append Column” and the default name for the new column is “prediction”. We selected the default option and we named the new column “class nr”.

After configuration, we named the Rule Engine node “from iris names to class nr” and we run the “Execute” command.

3.4. String Splitting

In this section we explore how to perform string manipulation with KNIME. For example, how can we split the column "class" in a way as to have "Iris" in one column and "setosa", "versicolor", or "virginica" in another column? Vice versa, how can I build a key to uniquely identify each row of the data table?

In KNIME there are 3 nodes to split string cells.

- **"Cell Splitter by Position"** splits each string based on character position. The column to split contains a number of strings. The node splits all strings in the column in k substrings, each of length n1, n2, n3,… nk, where n1+n2+n3 +… nk is the length of the original string. Each substring is then placed in an additional column.

- **"Cell Splitter [by Delimiter]"** uses a delimiter character to separate the substrings in the original string. The column to split can also contain strings with variable length. If the delimiter character is found, the substring before and after will end up in two different additional columns. The name of the node is actually just "Cell Splitter". However, since it uses a delimiter character I will call it "Cell Splitter [by Delimiter]".

- **"Regex Split"** is a Cell Splitter by Regex. It uses a Regular Expression rule to recognize substrings. After the substrings have been recognized the node splits the original string into the recognized substrings and places them into different additional columns.

Unlike the split column operation, there is only one node to combine string columns: the "Column Combiner" node.

- The **"Column Combiner"** node concatenates the strings from two or more columns and puts the result into a new appended column.

Note. All string manipulation nodes, like the "Cell Splitter" nodes and the "Column Combiner" node, are located in the "Node Repository" panel in:
`"Data Manipulation" -> "Column" -> "Split & Combine"`

In the column "class" we want to separate substring "Iris" from the remaining substrings "setosa", "versicolor", or "viriginica".

- If we split by position, we need to split at the 4th character (at the end of "Iris" and before the rest of the string) and at the 5th character (before "setosa", "versicolor", or "virginica").
- If we split by delimiter, we need to split around character "-".

- Finally, if we split by RegEx, we need to find a Regular Expression rules to express “Iris”, “-“, and the remaining letters. A possible regular expression could be: `((Iris)[\-]*([A-Za-z]*)`.

Let’s see now in details how to use the three “Cell Splitter” nodes to do that.

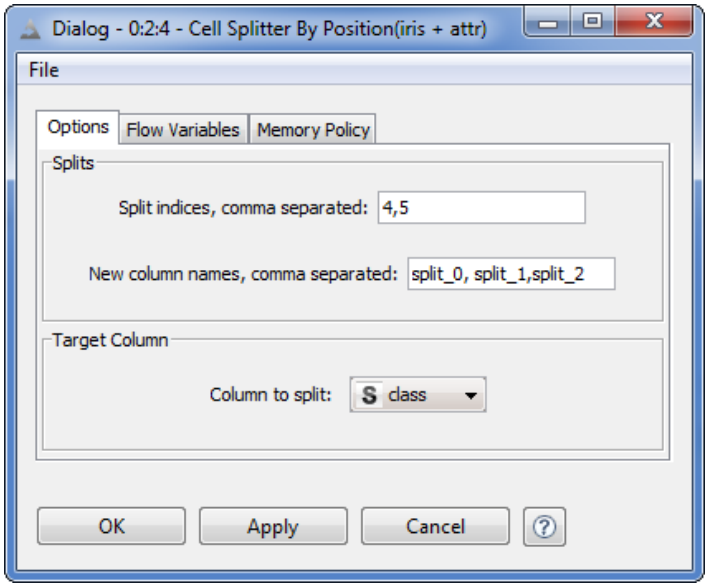
Cell Splitter by Position

This node splits the column string values based on character position. The result will be as many new columns as many delimiter characters have been found plus 1.

The configuration window asks for:

- The split indices (the character positions inside the string on which to split)
- The new column names (the new column names are always one more than the number of split indices)
- The name of the string column on which to perform the split

3.6. Configuration dialog for the "Cell Splitter by Position" node



We selected:

- Split indices 4 (at the end of word “Iris”) and 5 (after “-“)
- We will have then 3 substrings: “Iris” in column “split_0”, “-“ in column “split_1”, and “setosa”/”virginica”/”versicolor” in column “split_2”
- The column to perform the split on is “class”

Column “split_1” will contain only strings “-“ . We can always remove it later on by means of a “Column Filter” node.

Cell Splitter [by Delimiter]

This node splits the column string values at the delimiter character. The result will be as many new columns as many delimiter characters have been found plus one.

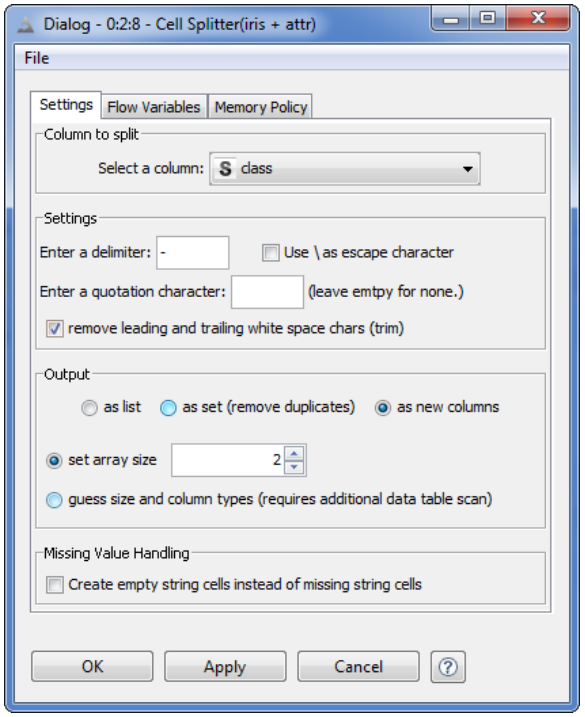
The configuration window requires the following settings:

- The name of the column on which to perform the split
- The delimiter character
- The output type:
    1. As new columns to append to the data table (here you need to set the array size)
    2. As one column only containing the list/set of sub-strings (a set of strings is like a list but without duplicate values)

If the array size is smaller than the number of detected substrings, the last splits will be ignored.

On the other side if the array size is bigger than the number of detected substrings, the last new columns will be empty.

3.7. Configuration dialog for the "Cell Splitter" node



We selected:

- Column to split = “class”
- Delimiter character = “-“
- Array size = 2

The substrings will be stored in column “<original_column_name>_Arr[0]” and “<original_column_name>_Arr[1]”, that is based on our configuration settings “class_Arr[0]” and “class_Arr[1]”.

Note. Here there is no column with only strings “-“. All characters “-“ are lost.

RegEx Split (= Cell Splitter by RegEx)

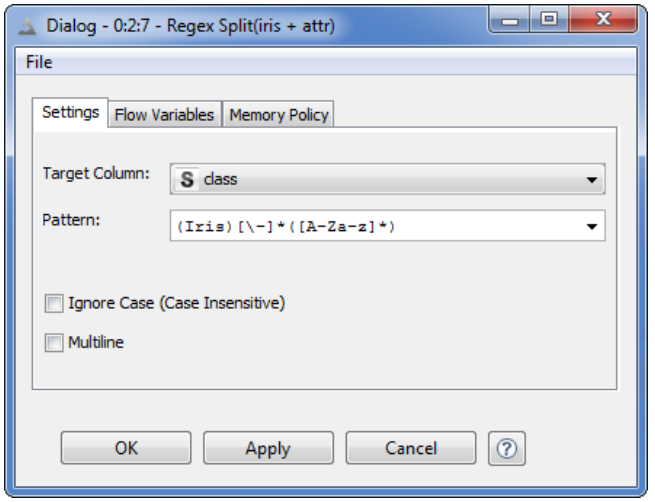
This node identifies substrings in a selected string column on the basis of a Regular Expression.

Substrings are represented as Regular Expressions inside parenthesis. The original string is then split in the substrings identified by the regular expressions. Each substring will create a new column.

The configuration window requires:

- The name of the column to split
- The Regular Expression pattern to identify the substrings, with the substrings included in parenthesis and the delimiter characters placed between the parenthesis groups

3.8. Configuration dialog for the "RegEx Split" node



To separate the word “Iris” from the rest of the string in column “class” by using a “RegEx Split” node, we selected:

- Column to split (Target Column) = class
- Regular Expression: `((Iris)[\-]*([A-Za-z]*)`, which means:
    1. First substring in parenthesis contains the word “Iris”
    2. Then comes a “-“ not to be used as a substring, since it is not in parenthesis
    3. The second substring can contain any alphabetical character

The result are two substrings named “split_0” and “split_1”, one containing the word “Iris” and the other containing the remaining word “setosa”, “versicolor”, or “virginica”.

The same result could have been obtained with a more general Regular Expression, like for example ([A-Za-z]*)[\-]*(.*$), which means:

4. First substring in parenthesis contains any alphabetical character
5. Then comes a “-“ not to be used as a substring, since it is not in parenthesis
6. The second substring can contain any alphanumerical character

These “Cell Splitter” nodes have all been named “iris + attr”, which describes the split between word “iris” and the following attribute “versicolor”, “virginica”, or “setosa”.

3.5. String Manipulation

Let’s suppose now that we want to rebuild the iris class name but with a different string structure, for example “<attribute>:IRIS”, with the word IRIS all in capital letters. We need then to replace the string “Iris” with “IRIS” and to recombine it with the <attribute> string. In KNIME there are many nodes to perform all kinds of string manipulation. One in particular, though, can perform most of the needed string manipulation tasks: the “String Manipulation” node.

String Manipulation

The “String Manipulation” node can perform a number of string manipulation tasks, like to calculate a string length, to compare two strings, to change a string into only uppercase or lowercase characters, to replace a substring or all occurrences of a character inside a string, to capitalize the string words, to find the positions of a character or substring occurrence, to extract a substring from a string, and so on.

The configuration window of the “String Manipulation” node is similar to the one of the “Rule Engine” node. The “**Expression Editor**” is located again in the central bottom part of the configuration window. Here a number of string functions can be combined together to obtain the desired string transformation.

The available string functions are listed above in the “**Function List**” panel. The “**Description**” panel on the right explains the task of the selected function.

Functions can also be visualized in smaller groups, by selecting a category in the “**Category List**” menu over the “Function List” panel.

On the left, in the “**Column List**” panel, all available data columns are displayed.

Double-clicking a column or a function automatically inserts it in the “Expression Editor”. String values have to be reported in quotation marks, for example “abc”, when introduced in the “Expression Editor”.

The “**Insert Missing As Null**” flag enables the production of a null string, instead of an empty data cell, when there are missing values in the input data. The configuration window finally requires the name of the new or of the existing column, depending on whether the resulting string has to overwrite existing data.

The “String Manipulation” node that we introduced in the “WriteToDB” workflow follows the “Cell Splitter” node and combines (string function “join()”) the <attribute> part of the class name with “:” and with an uppercase version (string function “uppercase()”) of the word “iris” (Fig. 3.9). The result is “setosa:IRIS” for the original “iris:setosa”.

Note. Notice the “toInt()”, “to Long()”, “toNull()”, “toDouble()”, “toBoolean()” functions which convert a string respectively into an integer, a double, and so on. They can be used to produce a non-string output column

An alternative processing consists of using a sequence of single dedicated nodes. In this way, the workflow becomes more crowded with nodes, but this approach might avoid us the interpretation of the many available string functions in the “Function List” of the “String Manipulation” node.

To change from lower to upper case or vice versa, KNIME has a “Case Converter” node in the category `“Data Manipulation” -> “Column” -> “Transform”`.

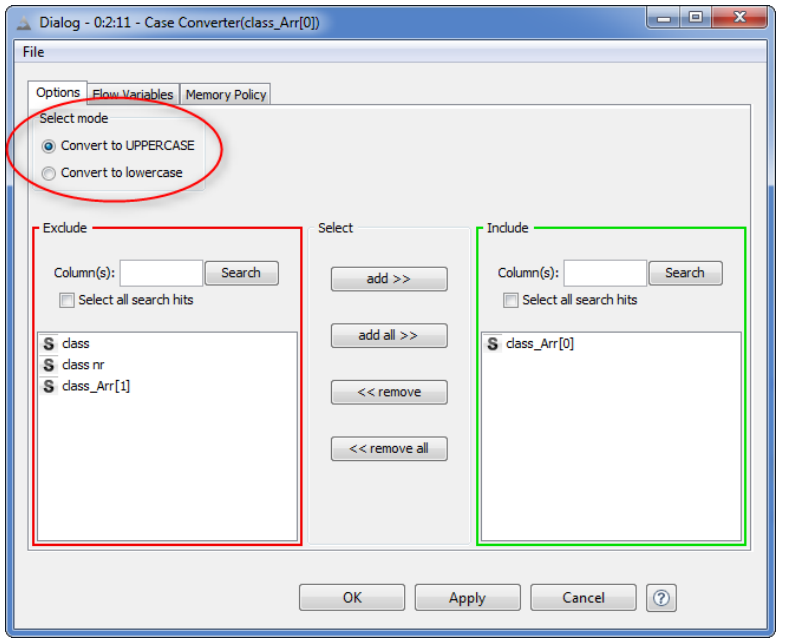
Case Converter

This node transforms the string characters into lowercase or uppercase depending on the “Select mode” flag.

The configuration window requires:

- “Select mode”: “UPPERCASE” or “lowercase”
- The names of the columns to transform. These columns are listed in the frame “Include”. All other columns that will not be affected by the transformation are listed in the frame “Exclude”.
- To move from frame “Include” to frame “Exclude” and vice versa, use buttons “add” and “remove”. To move all columns to one frame or the other use buttons “add all” and “remove all”.

3.10. Configuration window for the "Case Converter" node



We connected a “Case Converter” node to the output port of the “Cell Splitter” node. Of course we could have connected the “Case Converter” node to the output port of any of the “Cell Splitter” nodes. We chose the “Cell Splitter” node just as an example. Then we configured the “Case Converter” node like that:

- “Select mode” = “Convert to UPPERCASE”
- Columns to change = “class_Arr[0]”, which is the column containing the word “Iris”, in the “Include” set

To replace a string in general, KNIME has a “String Replacer” node in `“Data Manipulation” -> “Column” -> “Convert & Replace”`.

This node has a variant “String Replace (Dictionary)” that performs a spelling check based on a previously formatted dictionary text file.

String Replacer

The “String Replacer” node replaces a pattern in a string column.

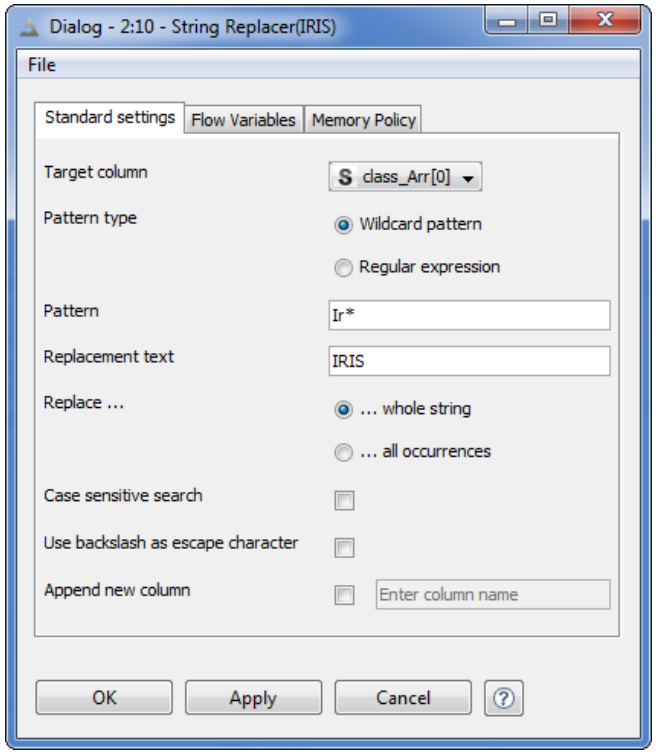
The configuration window requires:

- The name of the column where the pattern has to be replaced
- The pattern to replace (wildcards in the pattern are also allowed)
- The new pattern to override the old one with

And a few more options:

- Whether the pattern to be replaced contains wildcards or a regular expression
- Whether the replacement text must replace all occurrences of the pattern or just isolated strings matching the pattern
- Whether the pattern match has to be case sensitive
- Whether escape characters are indicated though a backslash
- Whether the result replaces the original column (default) or creates a new column

3.11. Configuration window for the „String Replacer“ node



To change string “Iris” into string “IRIS”, we can connect a “String Replacer” node to the output port of the “Cell Splitter” node and use the following configuration:

- Target column = “class_Arr[0]”, which contains string “Iris”
- Pattern to be replaced can be “Iris” or more generally “Ir*” with a wildcard “*”
- The new pattern to replace the old one = “IRIS”

As a result column “class_Arr[0]” contains all strings “IRIS” like the column generated with the “Case Converter” node.

Finally, we want to combine all those substrings in a new string column named “iris name” and containing strings structured as: “<attribute>:IRIS”. To combine two or more string columns, there is a “Column Combiner” node under `“Data Manipulation” -> “Column” -> “Split & Combine”`.

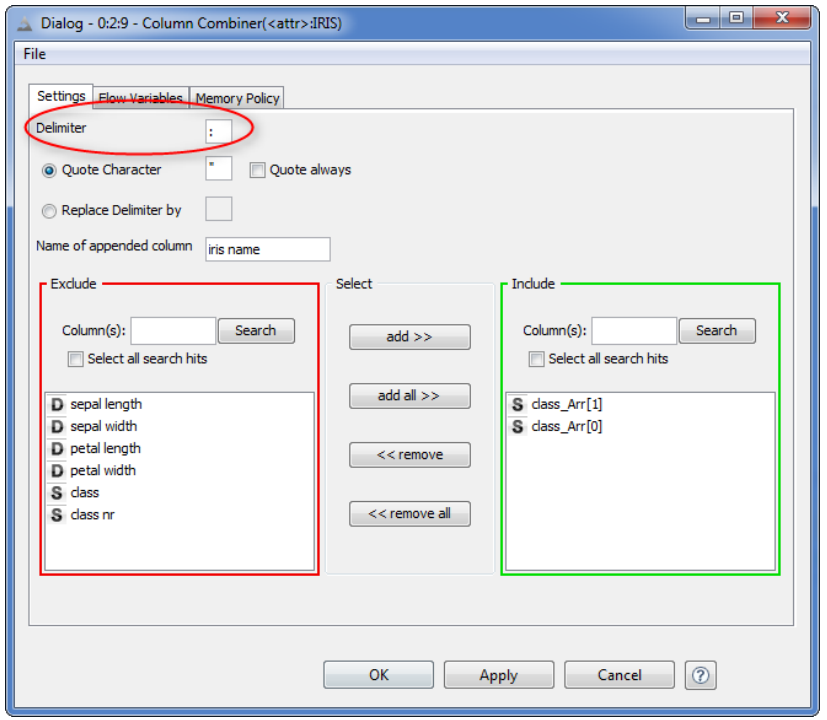
Column Combiner

The “Column Combiner” node combines two or more string columns into a single string column.

The configuration window requires:

- The delimiter character (if any, this field can also be empty)
- If we want to include the original substrings in quotes, then flag “Quote always” must be enabled and the “Quote character” must be supplied
- The name of the new column
- The names of the columns to combine. These columns are listed in the frame “Include”. All other columns that will not be used for the combination are listed in the frame “Exclude”.
- To move from frame “Include” to frame “Exclude” and vice versa, use buttons “add” and “remove”. To move all columns to one frame or the other use buttons “add all” and “remove all”.

3.12. Configuration window for the „Column Combiner“ node



To obtain the final String “<attribute:IRIS>”, we need a “Column Combiner” node with the following settings:

- Delimiter = “:”
- Columns to combine in the “Include” frame: “class_Arr[1]” and “class_Arr[0]”
- No quotes, that is flag “Quote always” is disabled

Note. In the "Column Combiner" node it is not possible to arrange the columns' concatenation order. Columns are combined following their order in the data table.

For example, column "class_Arr[0]" comes before column "class_Arr[1]" in the data table and the resulting combined strings will be "class_Arr[0]:class_Arr[1]", that is: "IRIS:<attribute>", which is not exactly what we wanted.

To change the columns' order in the data table, we can use a "Column Resorter" node located in `"Data Manipulation" -> "Column" -> "Transform"`.

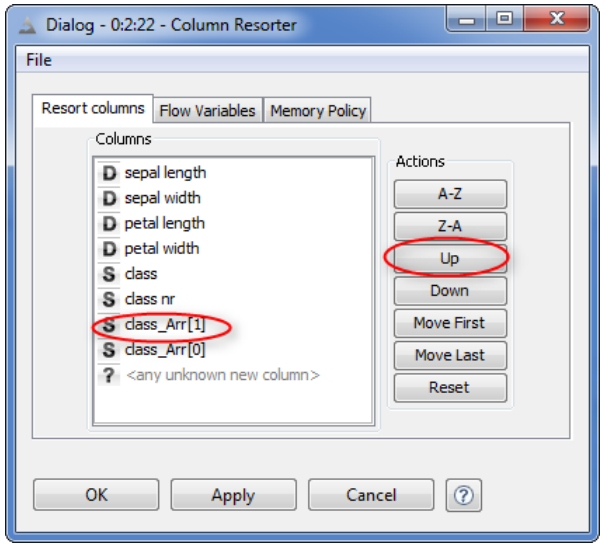
Column Resorter

The node "Column Resorter" changes the order of the columns in the input data table.

The list of columns with their order (left-to-right becomes top-to-bottom) is presented in the configuration window.

- To move one column down, select the column in the list and click button "Down". Same procedure to move one column up with button "Up".
- To make one column the first of the list, select the column and click "Move First". Same procedure to make one column the last of the list with button "Move Last".
- To use an alphabetical order on the column names, click button "A-Z" for descending order and "Z-A" for ascending order.

3.13. Configuration window of the "Column Resorter" node



We connected a "Column Resorter" node to the output port of the "Case Converter" node. We moved column "class_Arr[0]" one position down in the configuration window. We then named the "Column Resorter" node "class_Arr[0] after class_Arr[1]" and we connected its output port to the "Column Combiner". Now the "Column Combiner" has the input columns in the right order to get the final strings structured as "<attribute>:IRIS". The "Column Combiner" node was then named "<attr>:IRIS".

Note. The "Column Combiner" node is particularly useful to build unique keys to identify data rows.

3.6. Type Conversions

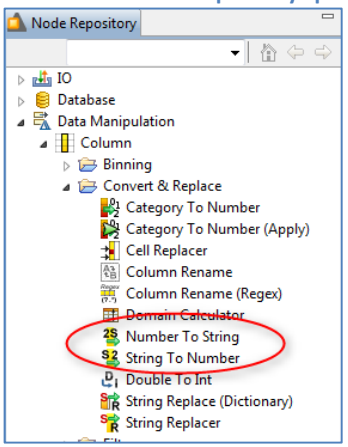
In the previous section we went through the string manipulation as performed by KNIME. Before moving to the database section, I would like to spend a little time showing the "Type Conversion" nodes.

In this book we will not work with Type DateTime. Excluding the Type DateTime, there are three basic "Type Conversion" nodes: "Number To String" node, "String To Number" node, and "Double To Int" node. All these nodes are located in the "Node Repository" panel in:

`"Data Manipulation" -> "Column" -> "Convert & Replace".`

In order to show how these Type Conversion nodes work, we will pretend that we want to change one of the data columns, for example "petal width", from Type Double to Type String. We will use a "Number To String" node for that.

3.14. Location of the "Type Conversion" node in the "Node Repository" panel



Number To String

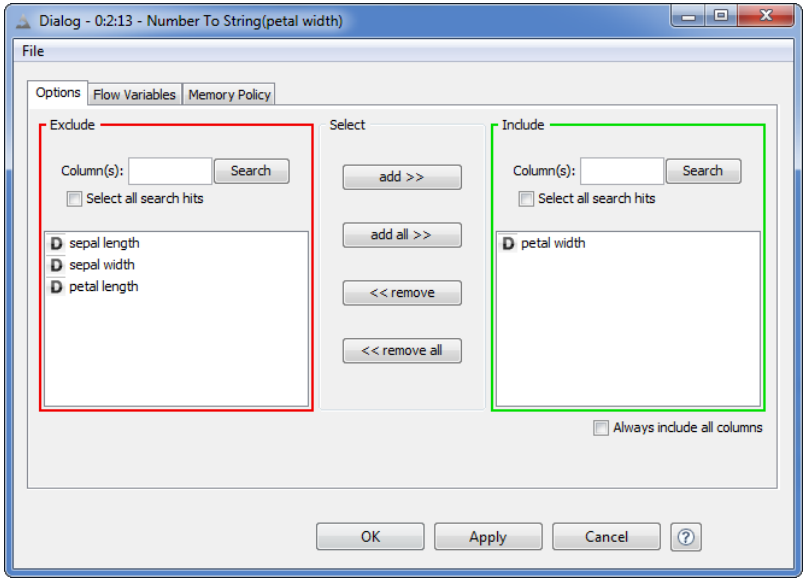
The "Number To String" node converts all cells of a column from Type "Double" or "Int" to Type "String".

The configuration window requires:

- The names of the columns to be converted to Type String. These columns are listed in the frame "Include". All other columns are listed in the frame "Exclude".
- To move from frame "Include" to frame "Exclude" and vice versa, use buttons "add" and "remove". To move all columns to one frame or the other use buttons "add all" and "remove all".

The "number to String" node is equivalent to the function "string()" in the "String Manipulation" node.

3.15. Configuration window of the "Number To String" node



We inserted only the column “petal width” in the frame “Include” to be converted from Type Double to Type String. The node has then been named just “petal width”, since it is already clear what its function is.

Now for demonstration sake, let’s suppose that we want to isolate the floating and the integer part of column “petal width”. Since now column “petal width" is of Type String, we will use a “Cell Splitter” node with delimiter character “.”. We named this node “int(petal width)”. At this point we have:

- The original String column “petal width”
- The first substring “petal width_Arr[0]” containing the integer part of “petal width” value
- The second substring “petal width_Arr[1]” containing the floating part of “petal width” value

Still for demonstration sake, let’s reconvert the integer and the floating part of “petal width” from a String Type to a Number Type (Double or Int). In order to do that, we can use the “String To Number” node.

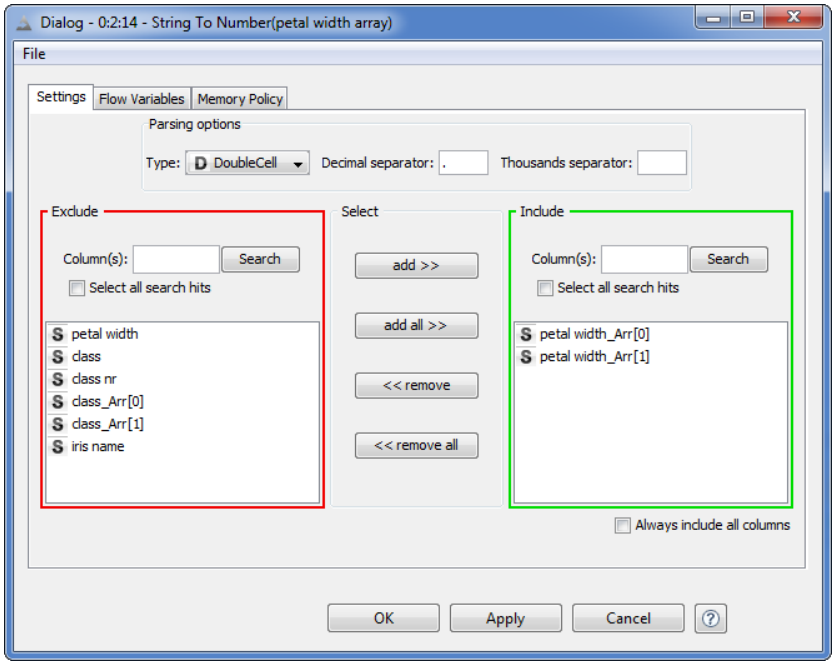
String To Number

The “String To Number” node converts all cells of a column from Type “String” to Type “Double”.

The configuration window requires:

- The final column Type: Double or Int
- The decimal separator and the thousands separator (if desired)
- The names of the columns to be converted to the selected Type. These columns are listed in the frame “Include”. All other columns are listed in the frame “Exclude”.
- To move from frame “Include” to frame “Exclude” and vice versa, use buttons “add” and “remove”. To move all columns to one frame or the other use buttons “add all” and “remove all” accordingly.

3.16. Configuration dialog of the "String To Number" node



This node has been named “petal width array” just to make it clear that it is working on the substring arrays generated from “petal width” column. Now the two columns “petal width_Arr[0]” and “petal width_Arr[1]” are of Type Double.

Let’s still suppose, for the sake of nodes demonstration, that we have converted the “petal width” array columns into the Type Double, but that actually we wanted to have Type Int. Let’s ignore the fact that it would be enough to change option “Type” in the configuration window of the “String To Number” node and let’s experiment with a new node: the “Double To Int” node.

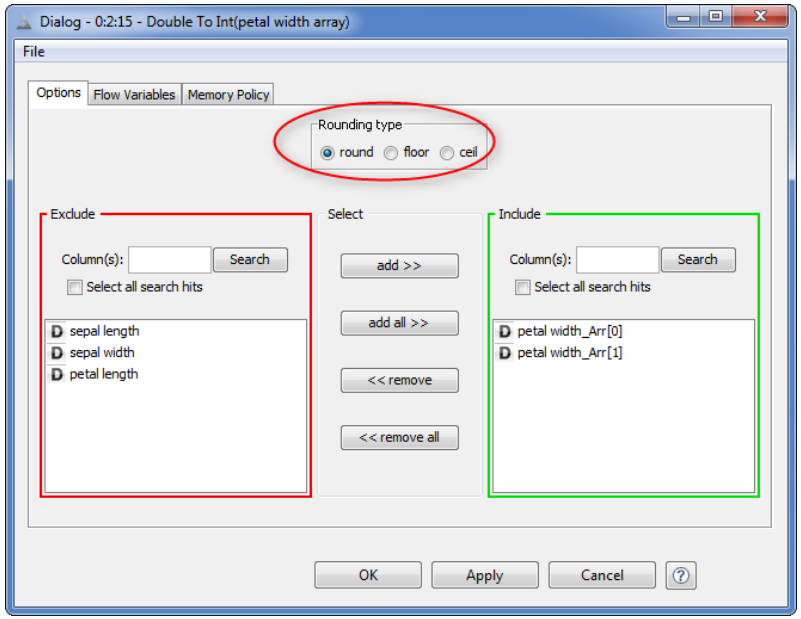
Double To Int

The “Double To Int” node converts all cells of a column from Type “Double” to Type “Int”.

The configuration window requires:

- The rounding type: round, floor, or ceil. “round” is the standard rounding, “floor” rounds up to the next smaller integer, “ceil” rounds up to the next bigger integer
- The names of the columns to be converted to Type Integer. These columns are listed in the frame “Include”. All other columns are listed in the frame “Exclude”.
- To move from frame “Include” to frame “Exclude” and vice versa, use buttons “add” and “remove”. To move all columns to one frame or the other use buttons “add all” and “remove all”.

3.17. Configuration window of the "Double To Int" node



3.7. Database Operations

We have only showed the Type Conversion nodes to illustrate KNIME’s potentials. We did not actually need these type conversions.

We will now write the data table feeding the “Type Conversion” nodes into a database. In the “Node Repository” panel there is a whole category called `“Database”` containing all nodes that perform operations on databases. However, before we start using any node to work on a database, we need to enable the protocol between KNIME and the database with the specific Database Driver. The Database Driver is a file that enables KNIME, or better Java programs in general, to interface with the database. We installed PostgreSQL (http://www.postgresql.org/), open source database

software, on our machine and we created a database “Book” in which to write the data table. We also defined a special user, with username “knime_user1” and password “knime_user1” to access the database.

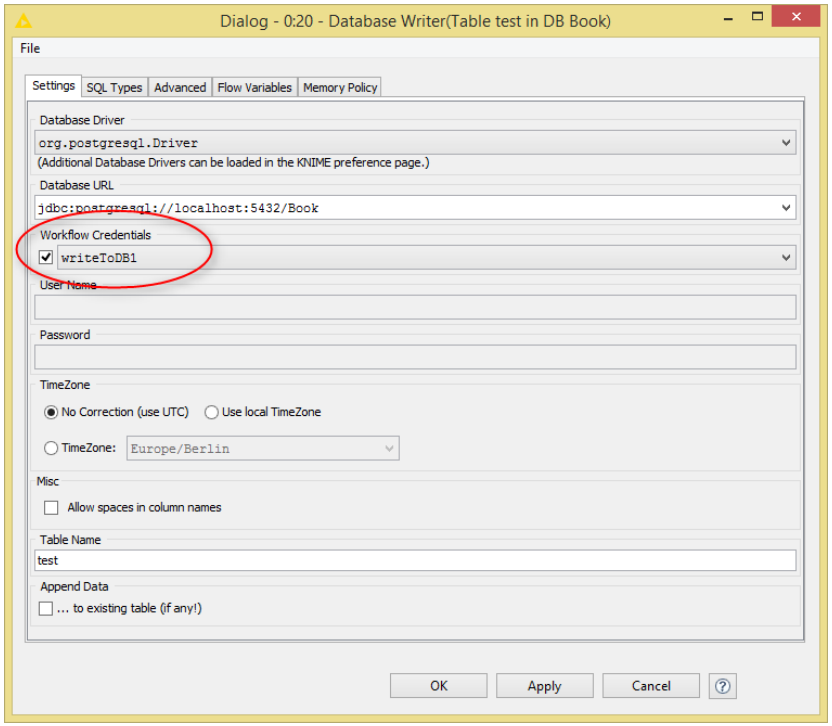
The goal of the “WriteToDB workflow is to write the final results in to a database table. In order to do that, we need the “Database Writer” node.

Database Writer

The node “Database Writer”, located in the `“Database”` category, writes the input data table into a database table in either the “Append” mode or the “Overwrite” mode. If the table does not exist in the database, it is created. The configuration window requires the following settings.

- The Database Driver. A few database drivers are preloaded and available in the database driver menu. If you cannot find the database driver for you, you need to upload it via the Preferences page (see below “Import a database Driver”).
- The URL location of the database, with server name (<host>), <port> on the server and <database name>.
- The credentials to access the database, i.e. username and password. The access credentials are provided by the database administrator. The database credentials can be supplied:
    1. Via the “Workflow Credentials” (recommended)
    2. Or directly with the traditional “User name” and “Password” fields.
- The specific Time Zone, if any
- The name of the table in the database
- The flag for the “Append” mode. The default writing mode is “Override”

3.18. Configuration window of the „Database Writer“ node



Workflow Credentials

You can set all usernames and passwords that you need for your workflow from the workflow's context menu.

- Right-click the workflow name in the "KNIME Explorer" panel
- Select "Workflow Credentials"

The "Workflow Credentials …" window opens.

To add a new workflow credential -- that is a new pair (Username, Password) -- :

- Click the "Add" button

- The window "Add/Edit Credentials" opens.

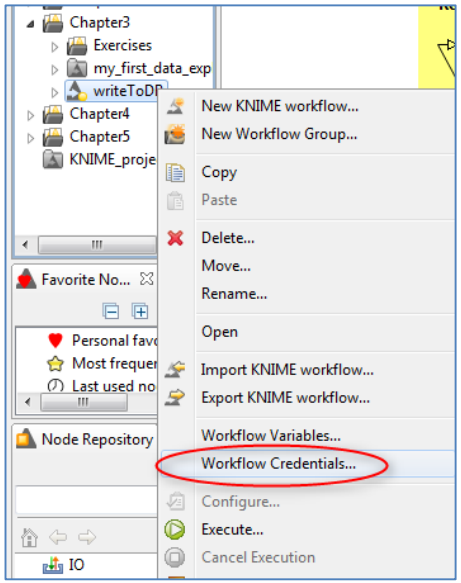
  - Set the credential ID, the User Login, and the User Password

  - Click the "OK" button

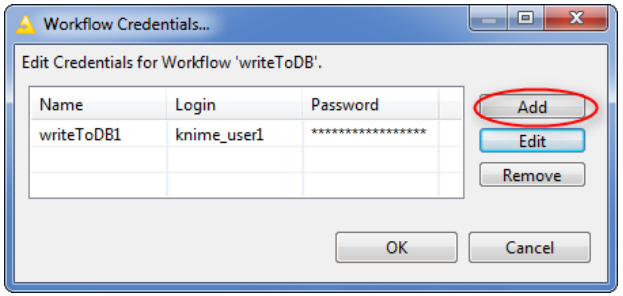
- Insert as many credentials as you need for your workflow

- Click the "OK" button

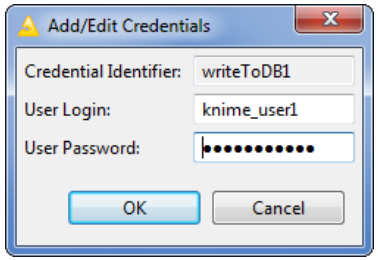
3.19. Set the "Workflow Credentials" from the workflow's context menu



3.20. The "Workflow Credentials..." window



3.21. The window "Add/Edit Credentials"



Note. The "Workflow Credentials" are automatically encrypted by KNIME. Database access via workflow credentials is then more secure and therefore recommended.

In order to write the processed iris data, a “Database Writer” node was connected to the output port of the “Column Combiner” node “<attr>:IRIS”. In the configuration window of the “Database Writer” node, we set the server name, the port number and the database name. The port number (=5432) was supplied by the database administration tool. We selected to write into the database called “Book”. We need now to set the username and password to access the database in the “Workflow Credentials”.

Master Key (deprecated)

The Master Key is simply a string that encrypts the password when the workflow, including the credentials, is saved on disk. Only you should know the value of the Master Key.

You can set the Master Key directly in the “Preferences” window.

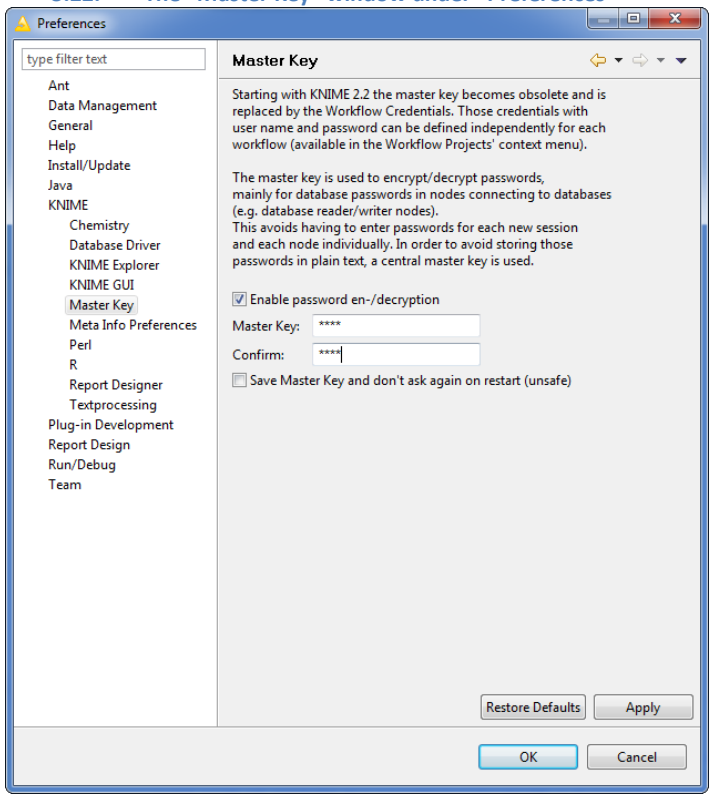
To open the “Preferences” window, in the Top Menu:

- Select “File”
- Select “Preferences”

In the “Preferences” window:

- Expand “KNIME”
- Select “Master Key”
- Enter the value for the Master Key
- Confirm the value for the Master Key
- Click “OK”

3.22. The "Master Key" window under "Preferences"



The ID list of all created workflow credentials is now available in the “Database Writer” node in the menu for the “Workflow Credentials” input. We selected the ID referring to the credential for database Book, named the final database table “test”, and opted for “Override” since we are not appending the data to an existing table (if any) (Fig. 3.18). Then we run the “Execute” command. If you now use the database administrator tool to check the “Book” database, you should see a table called “test” containing the Iris dataset. We named the “Database Writer” node “Table test in DB Book”.

In all database nodes it is also possible to supply directly the values for the “User name” and “Password”. By default, the contents of these fields are not encrypted when the workflow is saved, unless a value has been assigned to the “Master Key” variable in the “Preferences” window. Even though possible, this approach is considered obsolete since already KNIME 2.2. It is kept only for backward compatibility with workflows developed with older versions of KNIME.

Import a Database Driver

The most common and recent database drivers are already included in the database nodes. However, it might happen that you need a specific database driver not available in the database driver menu of the node. In this case, you need to upload the required database driver. Usually the Database Driver file (*.jar) can be found in the database installation or can be requested at the vendor’s site as an accessory to the database installation. In the second part of the installation of PostgreSQL we were indeed asked if we wanted to install drivers for this database. In order to load a database driver into KNIME, the driver location must be specified in the KNIME “Preferences” window.

Open “Preferences” window

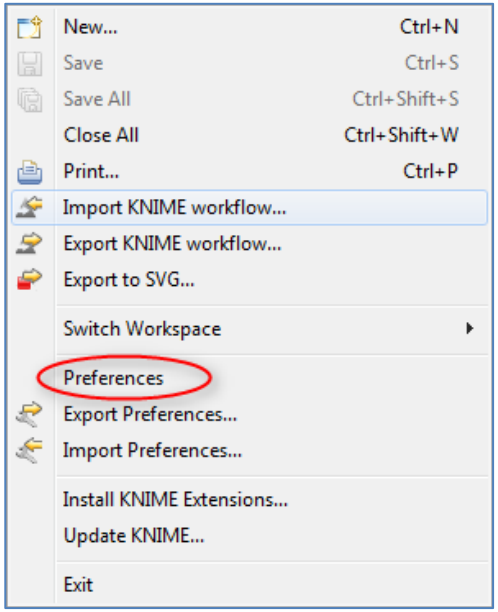
Database Drivers locations are specified in the “Preferences” window.

To access the “Preferences” window, in the Top Menu:

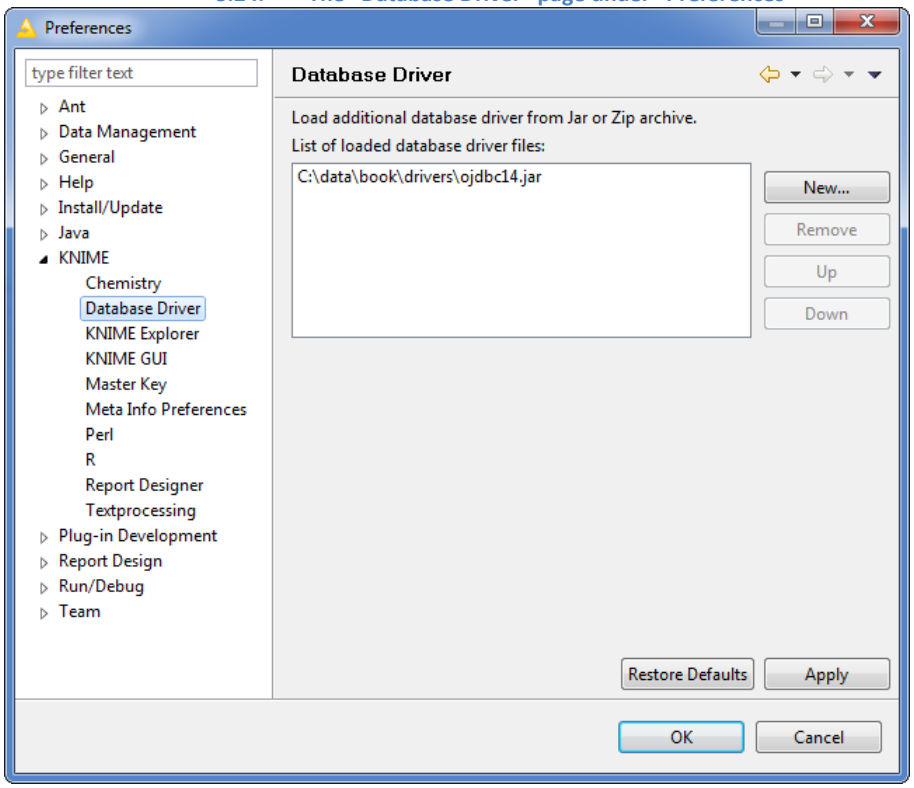
- Select “File”
- Select “Preferences”

The “Preferences” window opens.

3.23. Option "Preferences", located under "File" in the Top Menu, leads to the "Preferences" page



3.24. The “Database Driver” page under "Preferences"



Specify the location of a new Database Driver

The “Preferences” window sets the value for a number of general items, like “Help”, “Plug-in”, “Java” and so on. All these items are grouped in the list on the left in the “Preferences” window.

We are interested in the preferences for the item “KNIME” and inside item “KNIME” we are looking for the sub-item, “Database Driver”.

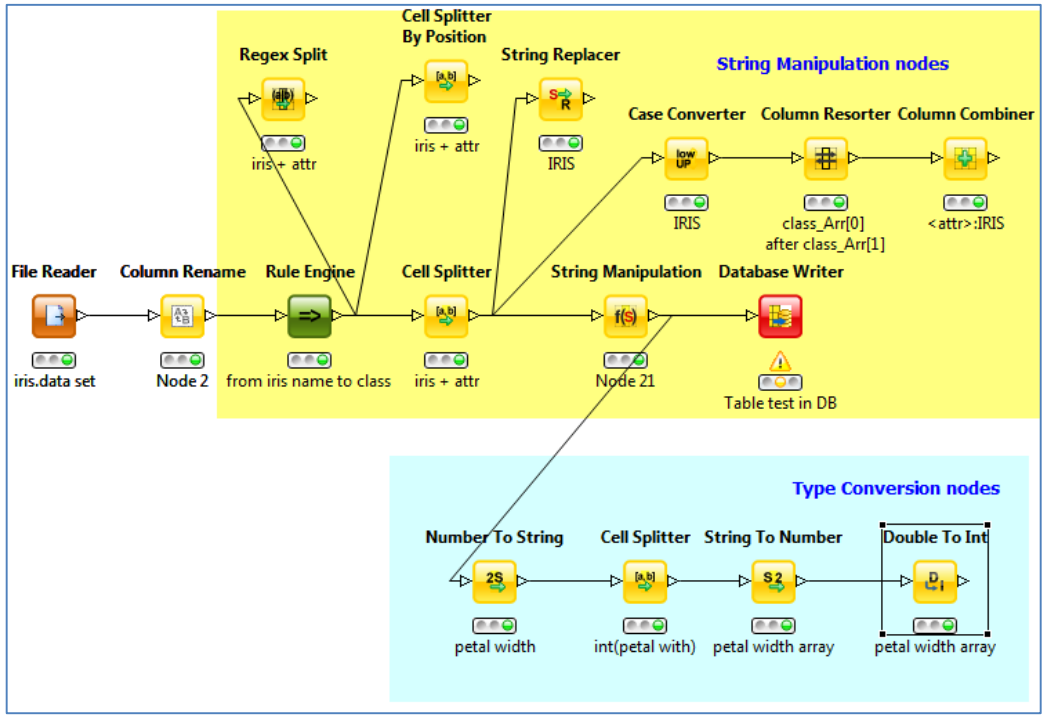
- Expand item “KNIME”
- Select sub-item “Database Driver”

The panel on the right displays the “Database Driver” settings.
In order to add a new database driver:

- Click the “New” button
- Select the *.jar or *.zip file that contains the database driver
- The new driver appears in the “List of loaded database driver files” in the center

We just finished the workflow “WriteToDB”, where the Iris Dataset was read and we performed a number of string manipulations and some type conversions. The data table emerging from the string manipulation node was written into a database table called “test”. The final workflow is shown in figure 3.25.

3.25. Workflow "WriteToDB"



Let's now read the same data to perform some visual data exploration. Next to the "Database Writer" node in the "Node Repository" panel, we find a "Database Reader" node. The "Database Reader" node will be used to read the data from the database table "test" created in the previous workflow.

We created a new workflow "my_first_data_exploration" and we used a "Database Reader" node in it. We named the "Database Reader" node "DB Book, Table test". We also created a "Workflow Credential" identifier, named "ID1", with username "knime_user1" and password "knime_user1", as defined at the beginning of this section.

Database Reader

The "Database Reader" node, located in the category "Database", reads data from a database table.

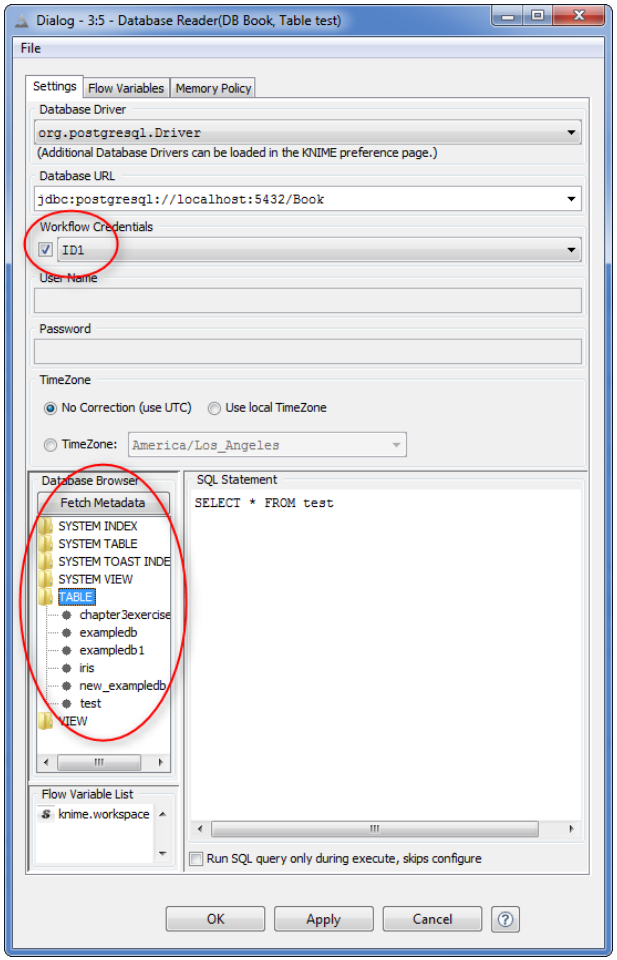
The configuration window requires the following information, whereby the first 5 options are the same as for the "Database Writer" node.

- The Database Driver. A few database drivers are preloaded and available in the database driver menu. If you cannot find the database driver for you, you need to upload it via the Preferences page (see below "Import a database Driver").
- The URL location of the database, in terms of server name (host), the port on the server and the database name.
- The credentials to access the database, i.e. Username and Password. These are supplied by your database administrator. Username and Password can be either supplied directly or via "Workflow Credentials".
- The Time Zone, if required.
- The SQL query (SELECT) to read the data is the only box that is present here, but not found in the "Database Writer" node configuration.

The SQL SELECT query can also be constructed using the help of the "Database Browser" panel on the left. Double-clicking a table or a table field in the "Database Browser" panel automatically inserts the corresponding object in the SQL SELECT query.

If it is not necessary to read the whole data table in our workflow, we may upload only the columns and/or the rows we are interested in via the "SELECT" statement.

3.26. Configuration window of the „Database Reader" node



The node reads all the columns from the table "test" in the database "Book" on the server "localhost". These are the 9 columns:

- 4 columns -- "sepal width", "sepal length", petal width", and "petal length" -- of data type "Double" come directly from the Iris dataset

- 1 column -- “class” -- represents the iris class and comes from the Iris dataset
- 1 column specifies the class number (“class 1”, “class 2”, and “class 3”) and was introduced to show how the “Rule Engine” node works
- The remaining 3 columns are substrings or combination of substrings of the column called “class”. They were introduced as examples of KNIME string manipulation operations.

3.8. Data Views

The goal of this chapter is to introduce the KNIME graphic tools and how data can be visually explored in KNIME. In this section, we show how some graphical nodes can be used to visually investigate the structure of the data. However, before continuing with the investigation of graphical nodes, I would like to describe an important feature of the KNIME data analysis tool, known as **Hiliting**.

Hiliting

“Hiliting” is a state that can be associated to data rows in a data table.

If a data row has been “hilited”, the “hilited” status follows the data row throughout the entire workflow (back and forth). Nodes enabled to recognize and display the “hilited” status, color the “hilited” data rows/points in orange. “Hiliting” can be set in one node for a few data rows; more data rows can then be “hilited” in another node’s view; all “hilited” data rows can be displayed in a third node; and finally the “hilite” status can be turned off in a forth node.

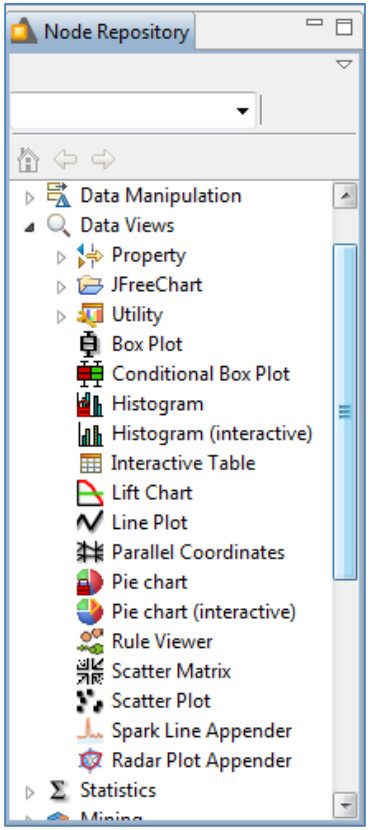
The “hilite” status is recognized by most graphical nodes, some model nodes, and even a few aggregation nodes, like the “GroupBy” node. For example, you can hilite the result/class of an aggregation/model node. All data rows falling into the result/class are hilited automatically and retroactively.

Note. The “Hilite” status remains with the data row until it is explicitly un-hilited or cleared. Even if you connect a new node to the workflow, the view of this new node shows the data rows with their previously set “hilited” status.

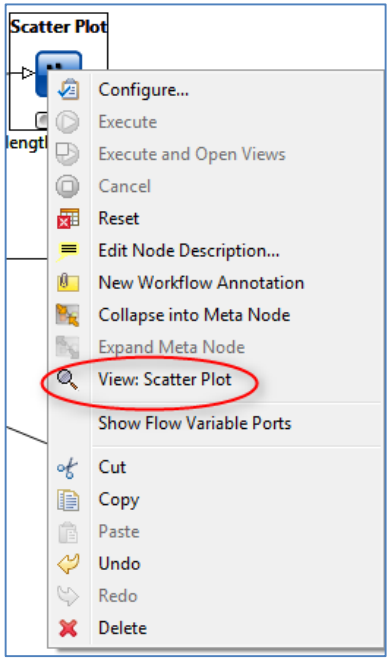
Let’s proceed now with the investigation of all those nodes that allow a graphical exploration of the data.

All graphical nodes are located in the same category called “`Data Views`” in the “Node Repository” panel (Fig. 3.27).

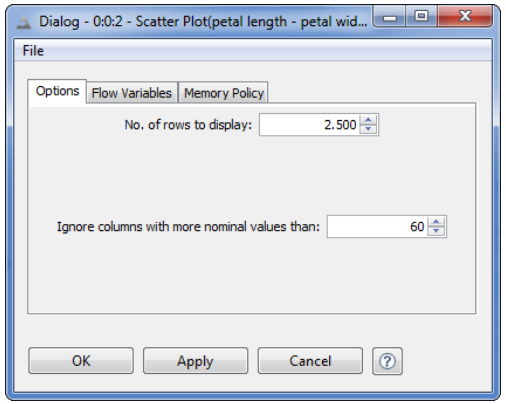
3.27. Location of the graphical nodes in the "Node Repository" panel



3.28. Context menu of a „Scatter Plot" node. The option to visualize the data is "View: Scatter Plot"



3.29. Configuration window for the „Scatter Plot" node



All view nodes have a "View" option available via the context-menu, that is when you right-click the node. For example, the "View" option of a "Scatter Plot" node is "View: Scatter Plot" (Fig. 3.28). The View shows the data in a special interactive window where you can change attributes and perspective.

For these graphical nodes the setting options are moved from the configuration window to the interactive View window. Therefore most graphical nodes present a very essential configuration window (Fig. 3.29). The configuration window usually contains:

- The number of rows to display
- The maximum number of nominal values allowed in a column: columns with more nominal values than the maximum allowed are ignored.

Let’s have a look at the View window of a few graphical nodes.

Scatter Plot

The “Scatter Plot” node plots each data row as a dot in a graphic by using two of its attributes as coordinates on the X-axis and the Y-axis.

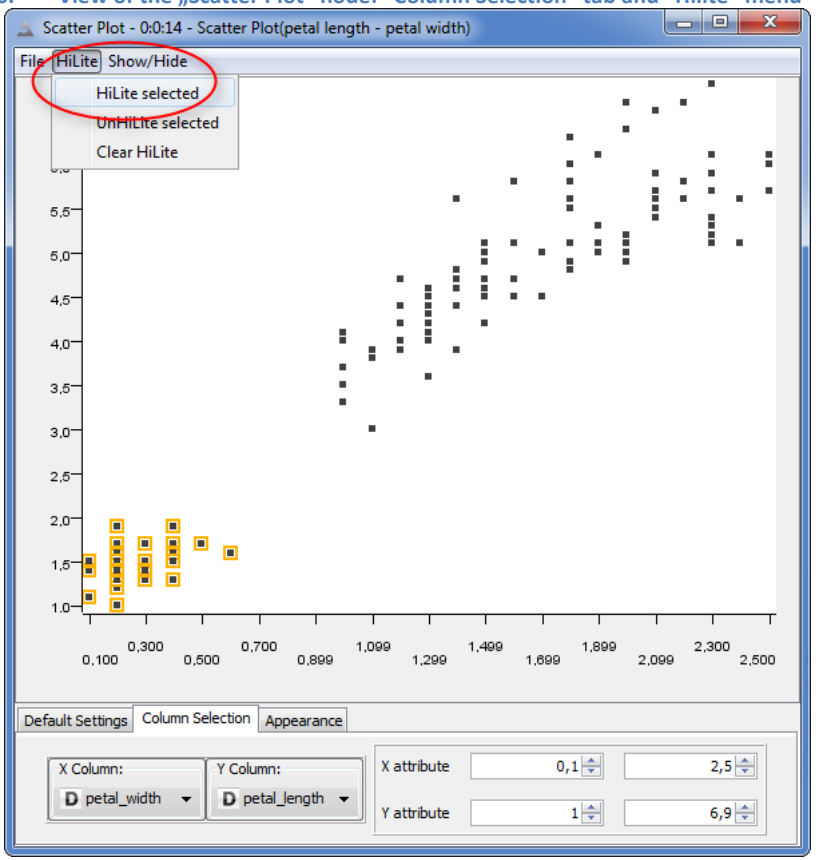
You can change the display of the plot by using the options in the tabs displayed on the bottom: “Default Settings”, “Column Selection” “Appearance”.

“**Default Settings**” contains the settings for the mouse mode, the size, and the background color of the plot.

“**Appearance**” contains the parameters to display the points in the plot.

“**Column Selection**” enables you to select the columns you want to be displayed in the plot. In the “Column Selection” tab you can select which columns are reported on the X- and Y-axis (see boxes “X-Column” and “Y-Column”). The 4 boxes on the bottom right define the range for the X and the Y attributes.

3.30. View of the „Scatter Plot“ node: “Column Selection” tab and “Hilite” menu

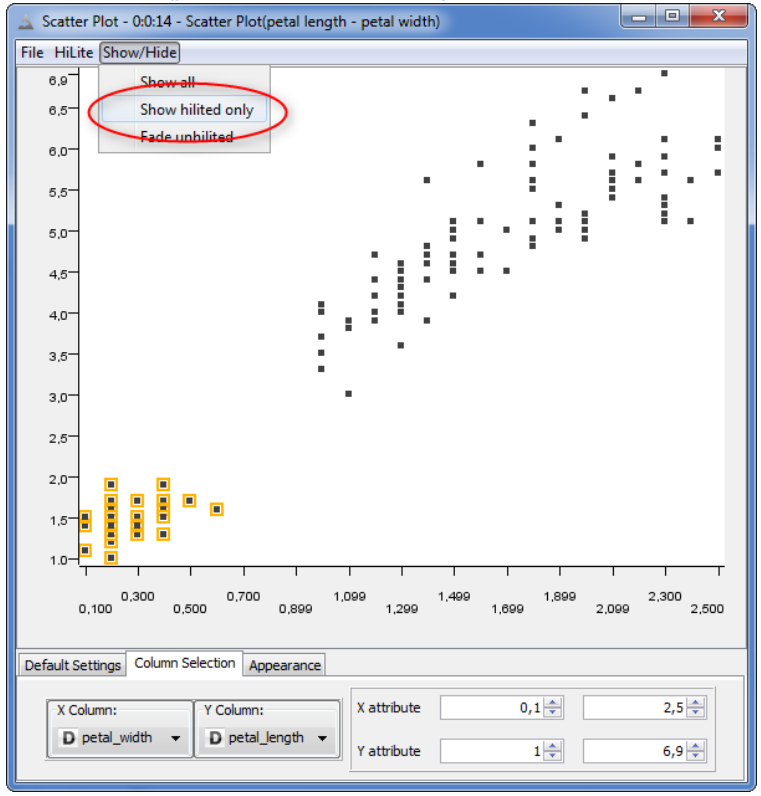


Top Menu of the Scatter Plot node's View

The Top Menu of the Scatter Plot node's View contains 3 items.

- **File.** "File" menu item allows the plot to be made always visible as "always on top" of the other windows, saving the plot as an image, or closing the window.
- **Hilite.** "Hilite" menu offers three hiliting options. You can "hilite selected" data points, "unhilite selected" data points, or "clear hilite" completely. Data points in the plot can be selected with the mouse and "hilited" via Top Menu -> "Hilite". "Hilited" means that the record underlying the data point is given a "hilite" status.
- **Show/Hide**. This item enables the data points to be selectively displayed, i.e. "all" data points, "hilited only" data points, or to "fade out unhilited" data points by using a less intense color.

3.31. View of the „Scatter Plot" node: "Show/Hide" menu



Hilite Data Points

- Select the data points in the plot, by clicking each data point or by drawing a rectangle around a group of data points
- Click on "Hilite" in the Top Menu of the plot view
- Click on "Hilite selected"

Selected data points have a black border; hilited data points have an orange border.

A data point stays hilited until "Unhilite" or "Clear Hilite" is selected.

Save plot as image

You can export the plot as a graphical file.

In the Top Menu of the Scatter Plot View:

- Select "File"
- Select "Export as"
- Select "PNG" or "SVG" for the graphical format

Scatter Matrix

A “Scatter Matrix” node plots a matrix of scatter plots by reporting all selected attributes once on the Y-axis and once on the X-axis.

To select the attributes for the scatter plots, you will find a classical “Exclude-Include” frame at the bottom of the plot view.

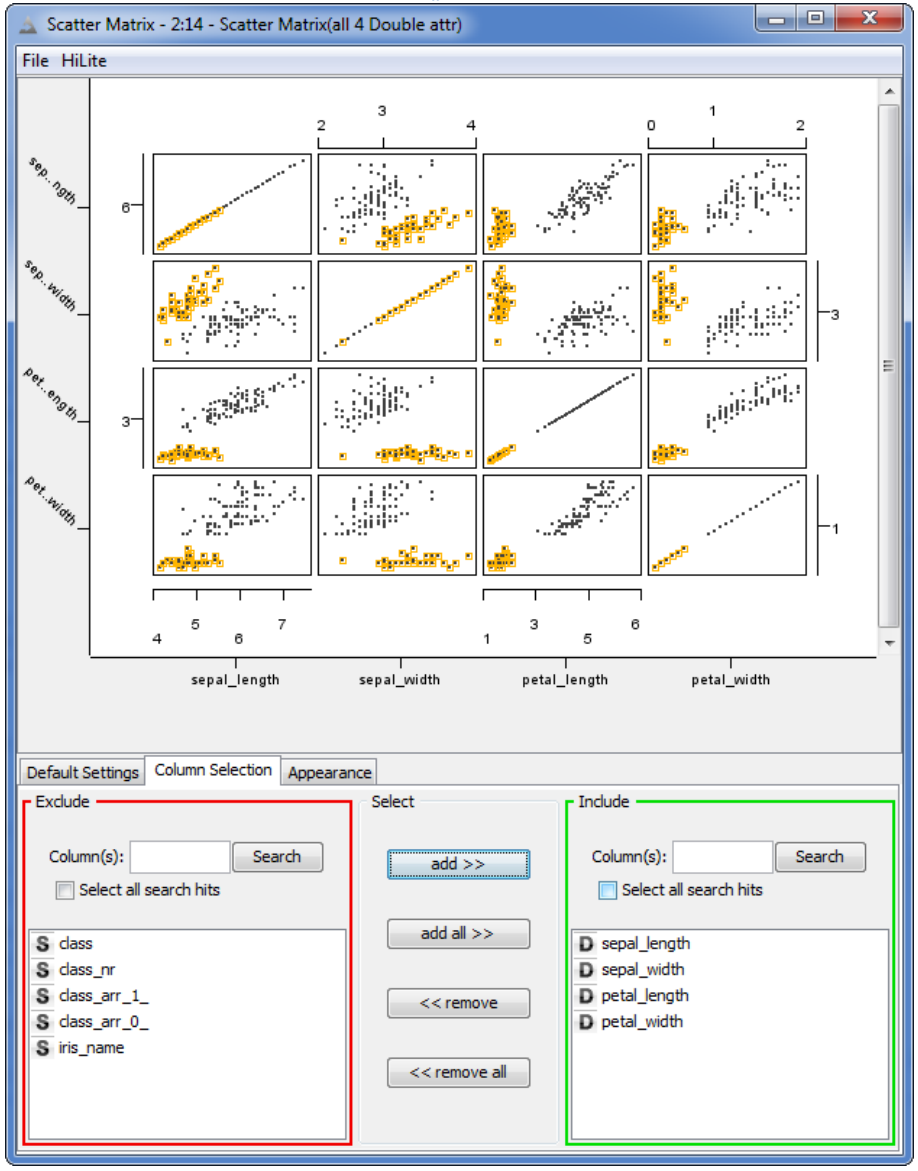
- The names of the columns to use to build the plots are listed in the frame “Include”. All other columns are listed in the frame “Exclude”.
- To move from the frame “Include” to the frame “Exclude” and vice versa, use buttons “add” and “remove”. To move all columns to one frame or the other use buttons “add all” and “remove all”.

The Top Menu of the plot view contains the same commands as the plot view for the “Scatter Plot” node.

- “File” -> “Export as” and select “PNG” or “SVG” as graphical format
- “Hilite” -> “Hilite Selected” to hilite selected points
- There are no “Show/Hide” options in comparison to the view of the “Scatter Plot” node.

“Default Settings” and “Appearance” tabs contain the settings for background, mouse mode, and point appearance in the plot.

3.32. View of the „Scatter Matrix“ node



In the “Scatter Matrix” node we chose to plot all attributes of type Double— “petal length”, “petal width”, “sepal length”, and “sepal width”—, in this way obtaining 16 scatter plots of the Iris data. We named the “Scatter Matrix” node “All 4 Double attr” to remind the user which attributes were used for the scatter plots.

The two separable classes of the Iris dataset are easily distinguished in the scatter plot node, by setting X-column = “petal length” and Y-column = ”petal width”. The two classes can be distinguished even more easily in some other scatter plots of the “Scatter Matrix” node.

Note. Generally plot views do not save the settings. Every time we open a plot view the settings are reset to their default values. Only the configuration values are saved.

We have found that the separation between two of the three classes of the Iris data set is best observed in a scatter plot if we use “petal length” as the X-column and “petal width” as the Y-column. However, the plot view does not store the X-column and Y-column settings, and every time we open the “Scatter Plot” node’s plot view, we have to re-set the X-column and the Y-column values to “petal length” and “petal width” without re-executing the node. When we open the plot view of the “Scatter Plot” node, the default columns are always “sepal length” and “sepal width”, i.e. the first 2 attributes of the data table.

Note. The “Hilited” status is recognized across plots, models, and interactive tables. Indeed data points with the hilited status keep this status across plots, interactive tables, and models nodes until their status is manually cleared by the user.

When we built the scatter plot we hilited the group of points close to the axis origin (displayed in orange). Afterwards when we built the scatter matrix plots, we did not hilite any point, but we still have hilited points (in orange). The hilited points in all plots of the scatter matrix have inherited their hilited status from the “Scatter Plot” node.

If we connect the original data table -- that is the data table coming from the “Database Reader” node -- to an “Interactive Table” node, we see that the points that are “hilited” in the plots are also “hilited” in the “Interactive Table”. The “Interactive Table” node allows navigating the records in the data table, such as jumping to a particular row or isolating the hilited or non-hilited records. If we connect a model node to the original data table, we could also distinguish the hilited records in the rules or clusters generated by the model.

We will have a look now at the node called “Interactive Table”. Model nodes are investigated in the next chapter.

Interactive Table

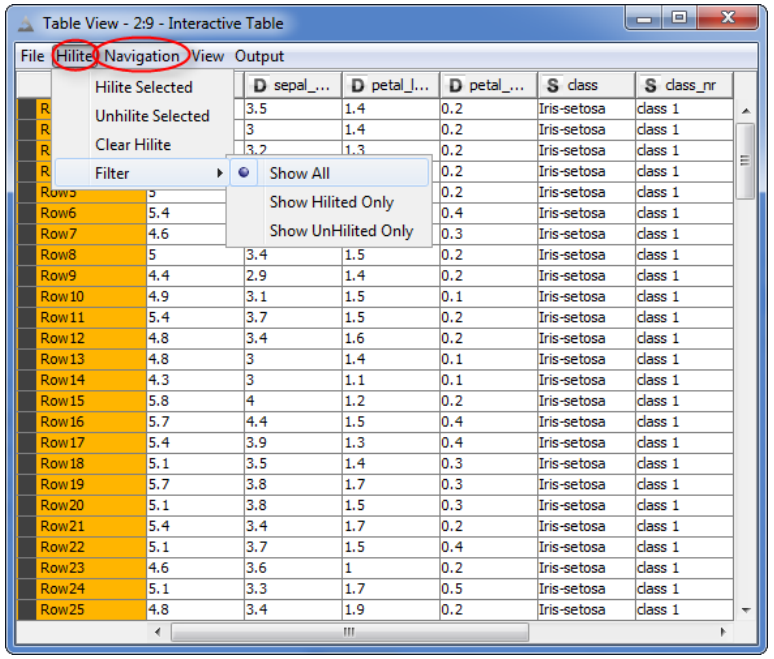
The “Interactive Table” node provides a View in which data rows can be searched and “hilited”. The “Interactive Table” node is located in the “Node Repository” panel in the `“Data Views”` category.

No configuration is necessary (the “Configure” option in the context- menu is disabled). The “View: Table View” option takes us to an interactive display of the data table.

In the Top Menu of the “Table View” window, we find two interesting items: “Hilite” and “Navigation”.

Top Menu items “File”, “View”, and “Output” contain options to export the table as a CSV file, as a PNG or SVG image, and to adjust the size of rows and columns.

3.33. Table View of the “Interactive Table” node



Hilite

“Hilite” contains commands to:

- Hilite/unhilite selected data rows
- Filter and show only hilited/unhilited data rows

Hilited rows might have been set to the “hilited” status by other nodes. RowKey cells of hilited rows are displayed in orange.

Navigation

“Navigation” contains commands to:

- Jump to a row with a specific row number
- Find a specific string in the rows

Note. If data points have been previously hilited in a plot, in a model or in another interactive table, the same data points will still be hilited (orange RowKey cells) in a newly created interactive table. This is called “Interactive Brushing”.

Another useful plot node available in the “Data Views” category is the “Line Plot” node. A line plot connects data points (=data rows) sequentially following their order in the data table. The row sequence represents the X-axis, while the corresponding attribute values are plotted on the Y-axis. A line plot is usually developed over time, i.e. the row sequence represents a time sequence. This is not the case of the iris data set, where rows represent only different iris examples and have no temporal relationship. Nevertheless, we are going to use this workflow to show how a “Line Plot” node works.

Line Plot

The “Line Plot” node displays data points using the row sequence as the X-axis.

The “Line Plot” node is located in the “Node Repository” panel in the `"Data Views"` category.

Like most of the “Data Views” nodes, it requires only minimal configuration settings. The “Plot View” is the window in which all plot settings are located.

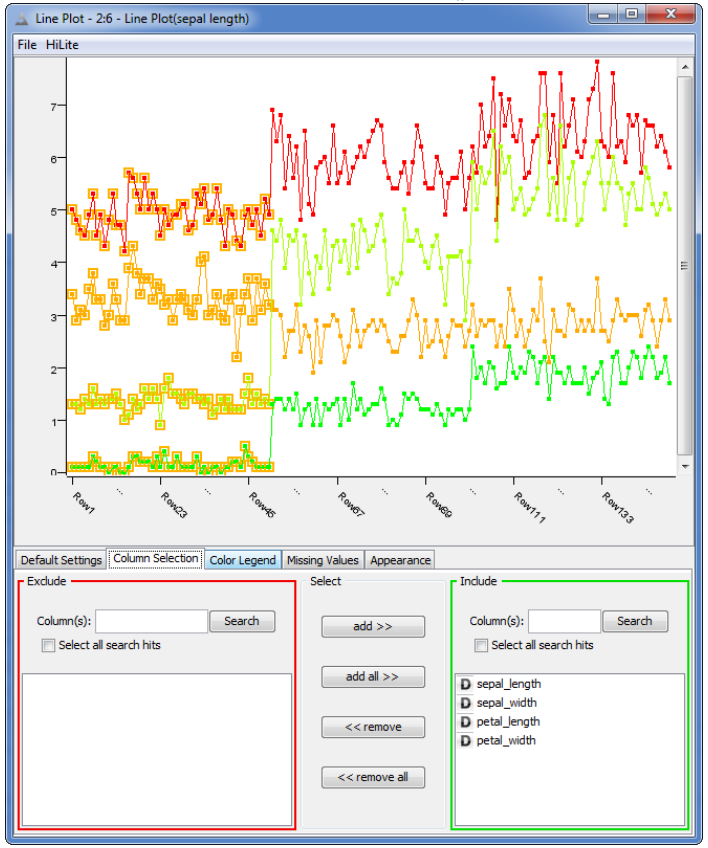
In particular, the tab “Column Selection” has an “Exclude/Include” frame that can be used to select the columns to plot.

- The columns to be plotted are listed in the “Include” frame. All other columns are listed in the frame called “Exclude”.
- To move from frame “Include” to frame “Exclude” and vice versa, use buttons “add” and “remove”. To move all columns to one frame or the other use buttons “add all” and “remove all”.

The “Color Legend” tab allows the plot color palette to be changed.

The “Default Settings”, “Missing Values”, and “Appearance” tabs include options affecting mouse usage, plot appearance and how to handle missing values.

3.34. “Line Plot View” window of the „Line Plot“ node



Of all the plots that are available to visually investigate the structure of the data, we cannot leave out the histogram. The histogram visualizes how often values in a given range (bin) are encountered in the value series.

First of all, we need to define the bins for the histogram view. In order to do that, we can use the "Numeric Binner" node.

Numeric Binner

The "Numeric Binner" node defines a series of bins (i.e. ranges) and assigns each value of a selected column to its bin.

The "Numeric Binner" node is located in the "Node Repository" panel in the `"Data Manipulation -> Column -> Binning"` category.

The configuration window requires the following:

- The list of bin intervals
- The numerical column to be binned
- A flag to indicate whether the binned values should appear in a new column or replace the original column

To define a new bin interval you must:

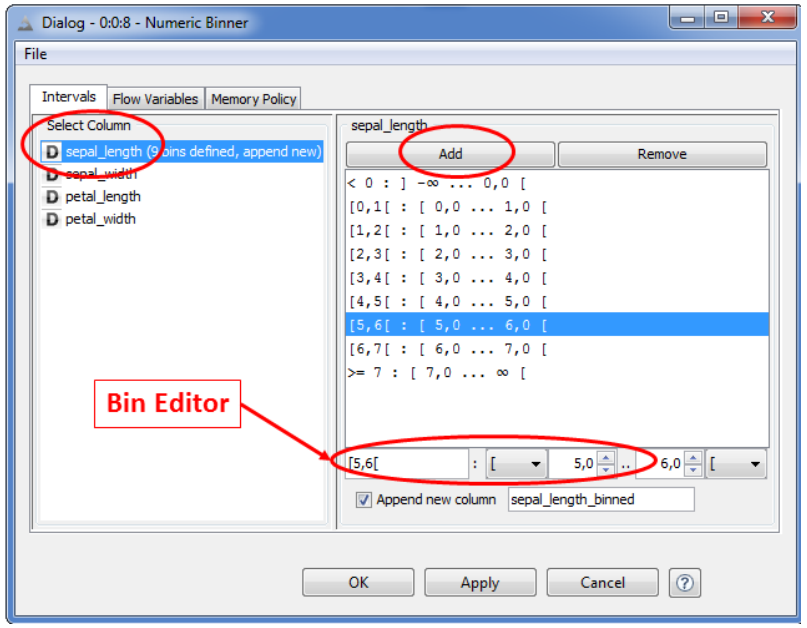
- Click the "Add" button
- Customize the bin range in the Bin Editor

To edit an existing bin interval:

- Select the bin interval in the list of bin intervals
- Customize the bin range in the Bin Editor

You can build a bin representation for other numerical columns in your data set by selecting a new column and repeating the binning procedure.

3.35. Configuration window for the „Numeric Binner" node



We chose to build the histogram on the values of the "sepal length" column only. We defined 9 bin intervals: "< 0", "[0,1[", "[1,2[", "[2,3[", "[3,4[", "[4,5[", "[5,6[", "[6,7[", and ">= 7". A square bracket at the outside of the interval means that the delimiting point does not belong to the interval. We also decided to create a new column for the binned values.

Histogram

The “Histogram” node displays how many times the values of a given column occur in a given interval (bin). The “Histogram” node is located in the “Node Repository” panel in the `“Data Views”` category.

Configuration window of the “Histogram” node’s

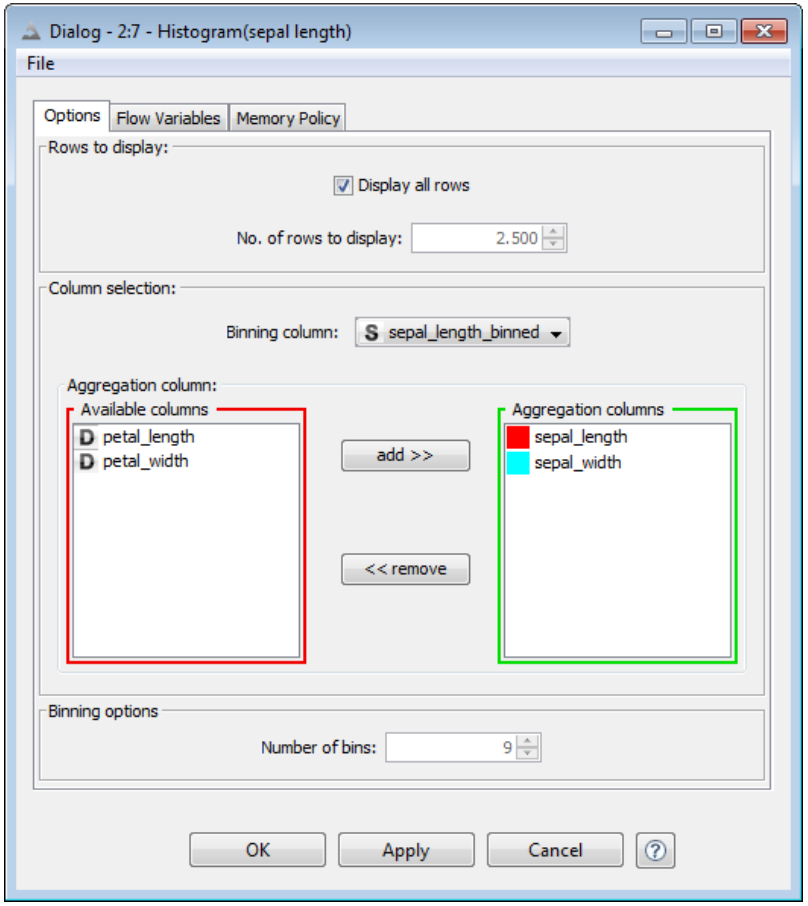
This is the only node in the “Data Views” category with a more sophisticated configuration window. The required configuration settings are as follows:

- The column(s) on which to count the value occurrences (“Aggregation columns”)
- The column on which to build the bin intervals (either an already binned column or a numerical column)
- The number of bins, providing the bins have to be built automatically on a numerical column
- A flag indicating whether all rows should be displayed (time and memory consuming) or only a limited number of rows
- The number of data rows to be displayed in the event that the user does not want to display all data rows

The “Aggregation columns” frame follows the “Exclude/Include” concept:

- The columns to be processed and displayed in the histogram are listed in the frame “Aggregation columns”. All other columns are listed in the frame “Available columns”.
- To move from frame “Aggregation columns” to frame “Available columns” and vice versa, use buttons “add” and “remove”. To move all columns to one frame or the other use buttons “add all” and “remove all”.

3.36. Configuration window of the „Histogram“ node

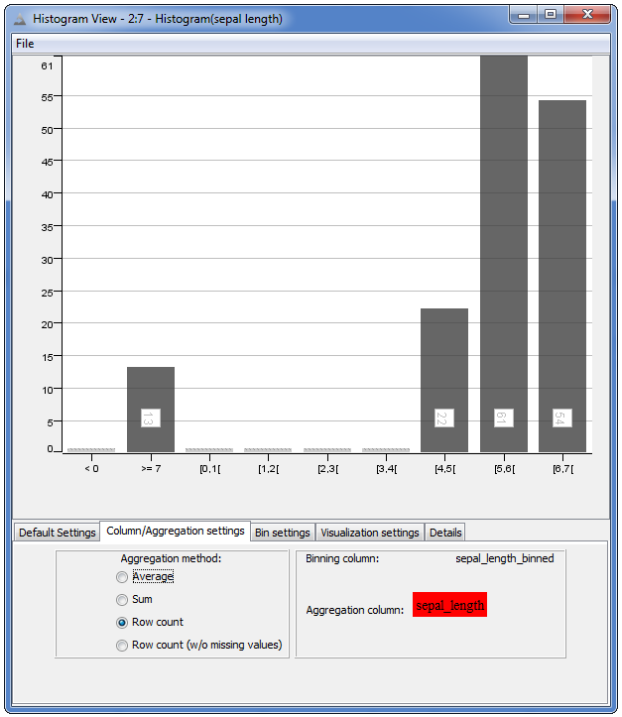


View window of "Histogram" node

The View window of the "Histogram" node includes a few interesting tabs:

- "**Column/Aggregation Settings**" enables you to change the aggregation method to calculate the histogram's bars. The classical numbers come from the row count, but a sum or an average of the values can also be displayed.
- "**Bin Settings**" contains a flag to display empty bins.
- "**Visualization settings**" includes a number of graphical options, like the bin size, the bar layout, and the labels settings, to better visualize the histogram. In figure 3.37 we have visualized the labels for "All Elements".
- "**Details**" allows the visualization of details of the histogram. For example, after clicking on a histogram bar, in "Details" we see what number exactly the histogram represents.
- "**Default Settings**" contains settings affecting the mouse usage.

3.37. View window of the „Histogram" node. The figure represents a histogram built on the prepared "sepal_length_binned" column

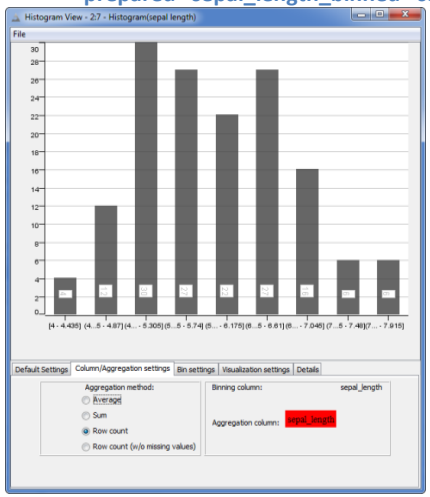


Note. In the "Histogram" node, the "RowCount" aggregation method counts the number of rows with the binned column's value in a given binning interval. Therefore with this definition the resulting number is independent of the selected aggregation column, since it always performs the counting in the binning column.

We selected:

- "sepal length" on the "sepal length_binned" bins created by the previous "Column Binner" node (in the configuration window)
- "Row Count" as aggregation method in the "Column/Aggregation Settings" tab (in the View window)
- checkbox "Show empty bins" enabled in the "Bin Settings" tab (in the View window)
- "All Elements" and "Vertical" in the "Labels" section in the "Bin Settings" tab (in the View window)

3.38. Histogram built on the “sepal_length“ column instead of the prepared “sepal_length_binned” column



Since the View does not remember the settings, we named the “Histogram” node “sepal length” to document that the histogram was built on the “sepal length” column.

Instead of preparing the bin intervals ourselves, we could have delegated this task to the “Histogram” node. That is, instead of using the prepared column “sepal length_binned”, we could have selected “sepal length” as the “Binning Column” and “9” as the “Number of Bins” in the configuration window. In this case, the “Histogram” node calculates the domain of the “sepal length” column and automatically builds 9 bins on it.

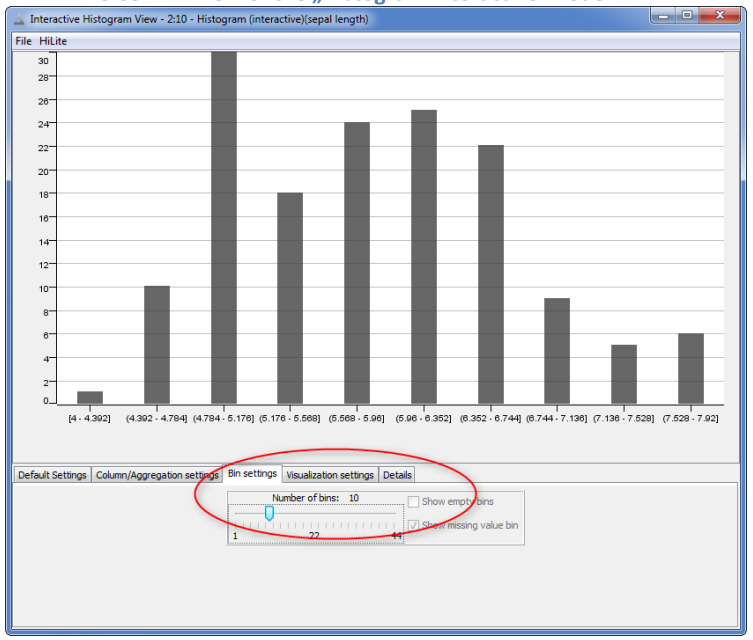
If we use this automatic approach, however, we end up with bins like (4.434-4.868] which might not be intuitive to interpret.

Histogram Interactive

The “Histogram Interactive” node is essentially the same as the “Histogram” node (see “Histogram” node).

The only difference is that the “Binning Column” and the “Number of Bins” parameters can be changed interactively in the “Histogram View” window, and precisely in the “Bin settings” tab.

3.39. View of the „Histogram interactive“ node



Parallel Coordinates

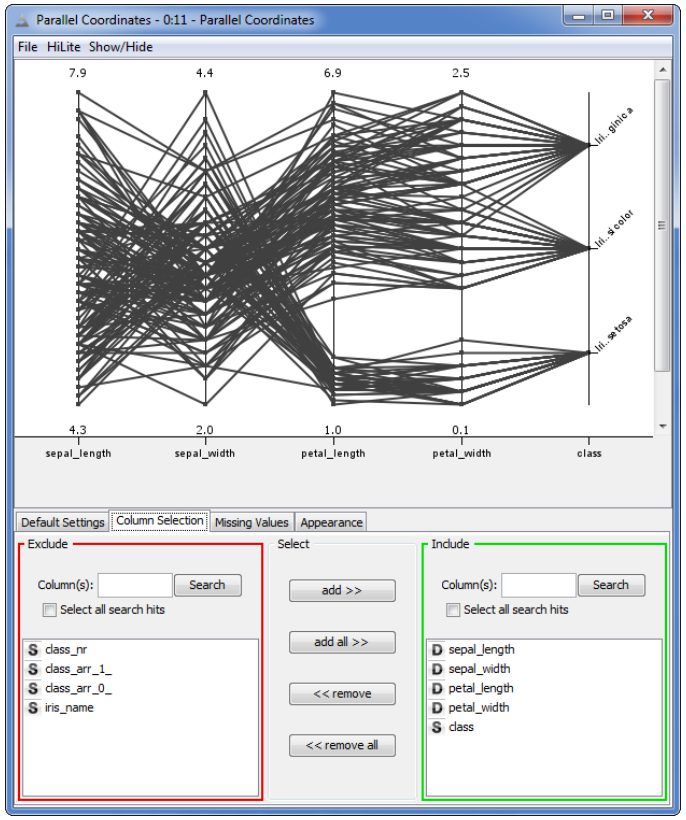
The "Parallel Coordinates" node displays the data table in a parallel coordinates plot. A parallel coordinates plot unfolds the attributes on the X-axis and displays each attribute value as a point with X=attribute and Y=value. As a result a data point is mapped as a line connecting values across attributes.

The "Parallel Coordinates" node is located in the "Node Repository" panel in the `"Data Views"` category.

With the "Parallel Coordinates" node we are back to the "Data Views" nodes that do not require much configuration. Almost all settings are set in the View.

- The "**Column Selection**" tab contains an "Exclude/Include" frame to add more columns to the parallel coordinates plot.
    1. The columns to be plotted are listed in the "Include" frame. All other columns are listed in the "Exclude" frame.
    2. To move from frame "Include" to frame "Exclude" and vice versa, use buttons "add" and "remove". To move all columns to one frame or the other use buttons "add all" and "remove all".
- The "**Missing Values**" tab provides options to deal with missing values
- The "**Appearance**" tab sets graphical options
- The "**Default Settings**" tab changes the mouse usage

3.40. View of the „Parallel Coordinates" node



This is where we finish our description of the list of nodes with data visualization functionalities. There are a few more interesting nodes in the "Data Views" category, such as the "Lift Chart" or the "Box Plot". However, we will leave them for the user to discover, based on the assumption that all nodes in this category follow the same scheme: minimal configuration settings and interactive views. If you are unsure of what each node does, consult the "Node Description" panel for more information.

3.9. Data Views Properties

Output plots of “Data Views” nodes can be customized in terms of color, shape, and size of the plot’s markers. KNIME has 3 nodes to customize plot appearance: the “Color Manager”, “Size Manager”, and “Shape Manager” nodes. These nodes can be found in the “`Data Views`” -> “`Property`” category in the “Node Repository” panel.

These nodes take a data table as input. In the settings you select one column and based on this column’s values you define a set of properties for the plots’ markers. These nodes have two output ports:

- One port contains the data table from the input port, but with the additional information of color, size, and shape assigned to each data row.
- The other port contains the parameters selected for plot customization: color, shape, and size. These parameters can be passed to an “Appender” node and reused for another data set.

Let’s have a look at the “Color Manager” node as an example of how this group of nodes works.

Color Manager

The “Color Manager” node assigns a color to each row of a data table depending on its value in a given column.

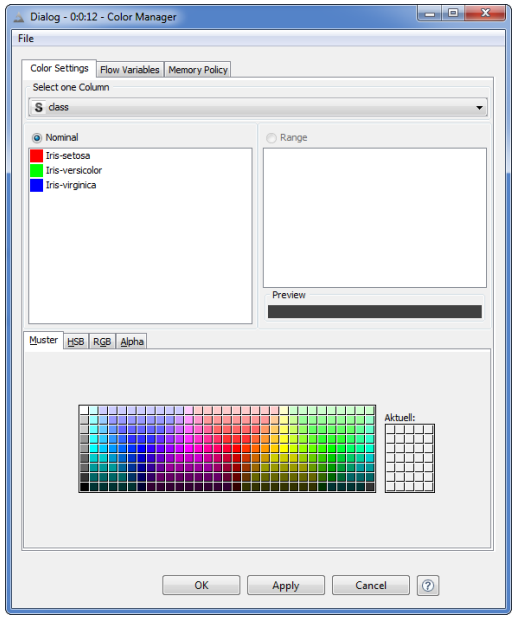
If it is a nominal column, colors are assigned in the “Configuration” dialog to each one of the nominal values. If it is a numerical column, different colors are assigned to some pre-defined value ranges.

The “Color Manager” node is located in the “Node Repository” panel in the “`Data Views” -> “Property`” category.

The configuration window requires:

- The column on which to build data groups
- The data groups to be marked with colors. Data groups are built on values for nominal columns and on ranges of values for numerical columns.
- The color for each value (nominal column) or range of values (numerical column)

3.41. Configuration window of the „Color Manager“ node



In our workflow “my_first_data_exploration” a “Color Manager” node was placed between the “Database Reader” and the “Scatter Matrix” node.

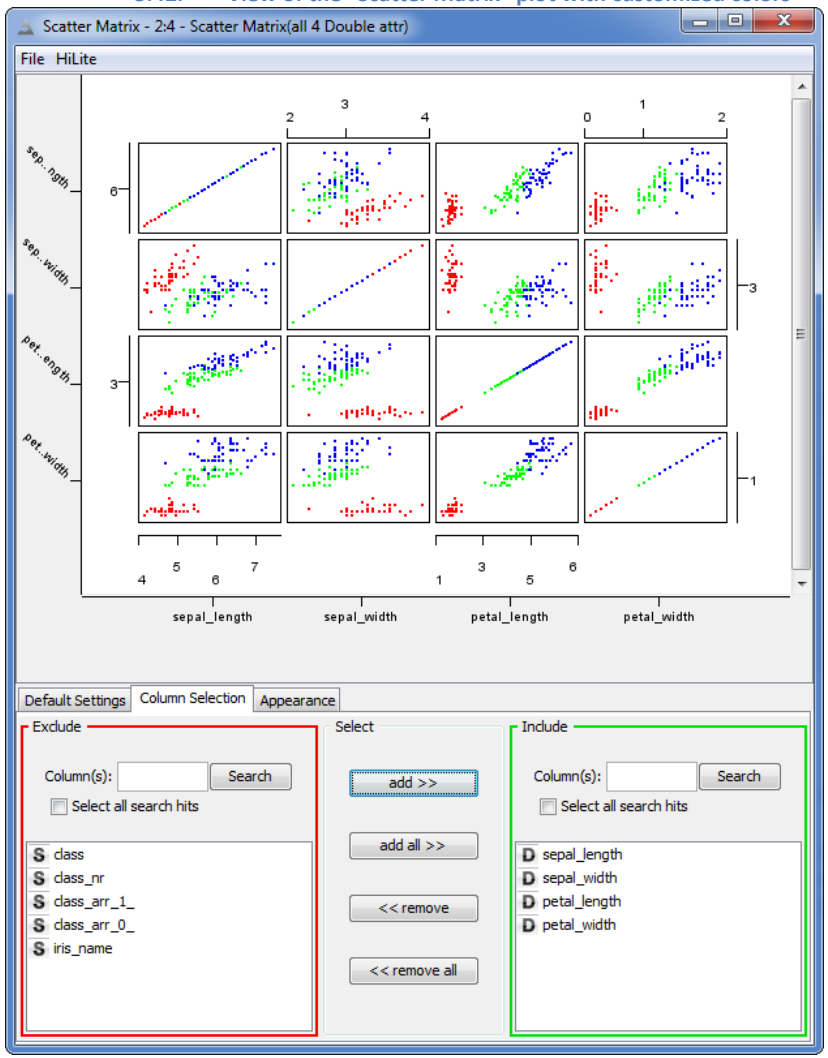
The goal here is to plot the iris classes with different colors. We chose the “class” column to receive the color scheme. This is a nominal column. After selecting the “class” column, KNIME automatically lists the three nominal values found in the “class” column and assigns a default color (red, green, and blue) to each nominal value. They are shown in the frame below the column selection box. Default colors can be changed by selecting the current color and choosing a new color from the color map in the lower area of the frame.

We kept the default colors offered by the configuration window for the three class labels found in the “class” column. We then ran the “Execute” command for both this node and the “Scatter Matrix” node. The color scheme was subsequently propagated to successor nodes from the “Color Manager” node to the “Scatter Matrix” node.

Now let’s open the view for the “Scatter Matrix” node and see what has changed. The data records belonging to the three classes are plotted with the three colors we selected. To be more precise, blue dots represent the “Iris-virginica” data, red dots the “Iris-setosa” data, and green dots the “Iris-versicolor” data (Fig. 3.42). It is pretty safe to state that the visualization of the data classes has been considerably improved in comparison to when the data was displayed in black and white.

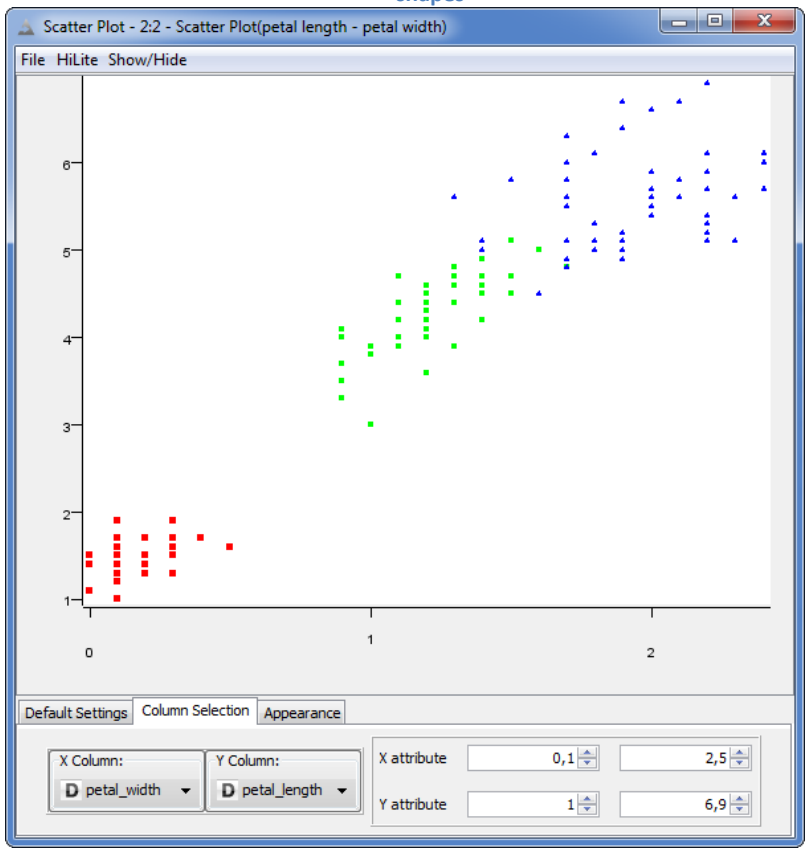
A “Shape Manager” node was also inserted between the “Color Manager” node and the “Scatter Plot” node to additionally customize the marker shape of each one of the three iris classes.

3.42. View of the “Scatter Matrix” plot with customized colors

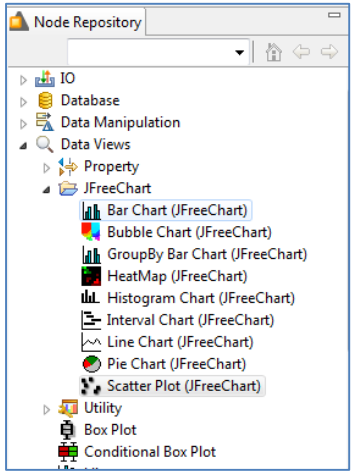


In the configuration window we selected the “class” column for the marker assignments and we allocated rectangle, circle, and triangle to each one of the three class labels found in the “class” column. Here, shape can be changed by clicking on the shape row itself. A menu appears showing all the possible alternative shapes. After running “Execute” for the “Shape Manager” node and the “Scatter Plot” node, the View of the “Scatter Plot” node shows the three iris classes with different colors and shapes (Fig. 3.43).

3.43. View of the "Scatter Plot" node with customized colors and markers' shapes



3.44. The "JFreeChart" sub-category



3.10. JFreeChart Nodes

There is one more sub-category in the "Data Views" category: "JFreeChart" (Fig. 3.44). The "JFreeChart" category contains a duplicate of most "Data Views" nodes: scatter plot, line plot, histogram, etc … The difference between these nodes and the "Data Views" nodes consists in the output and in the configuration window.

Scatter Plot (JFreeChart)

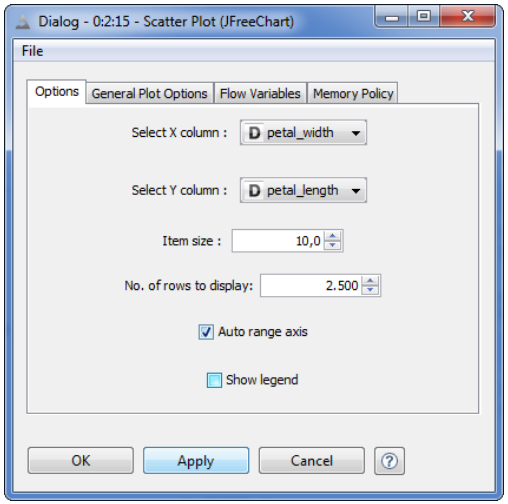
The “Scatter Plot (JFreeChart)” node produces a static image of a scatter plot.

Its configuration settings include 2 tabs: “Options” and “General Plot Options”.

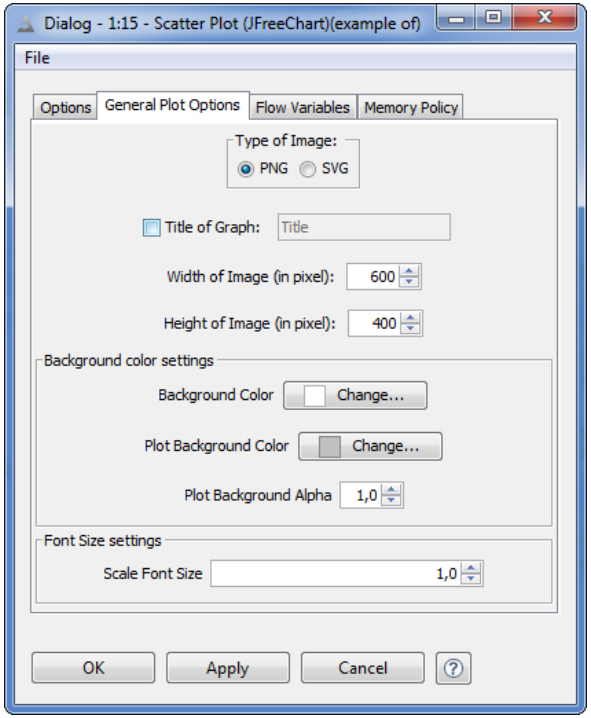
The “Options” tab sets:

- The two data columns to use as Y and X coordinates
- The dot size (“item size”)
- The number of rows to display (this parameter becomes necessary to reduce the waiting time in case of very large data)
- Whether the axis range has to be calculated automatically
- Whether the legend has to be shown

3.45. Configuration window of the „Scatter Plot (JFreeChart)“ node: the “Options” tab



3.46. Configuration window of the „Scatter Plot (JFreeChart)“ node: the “General Plot Options” tab



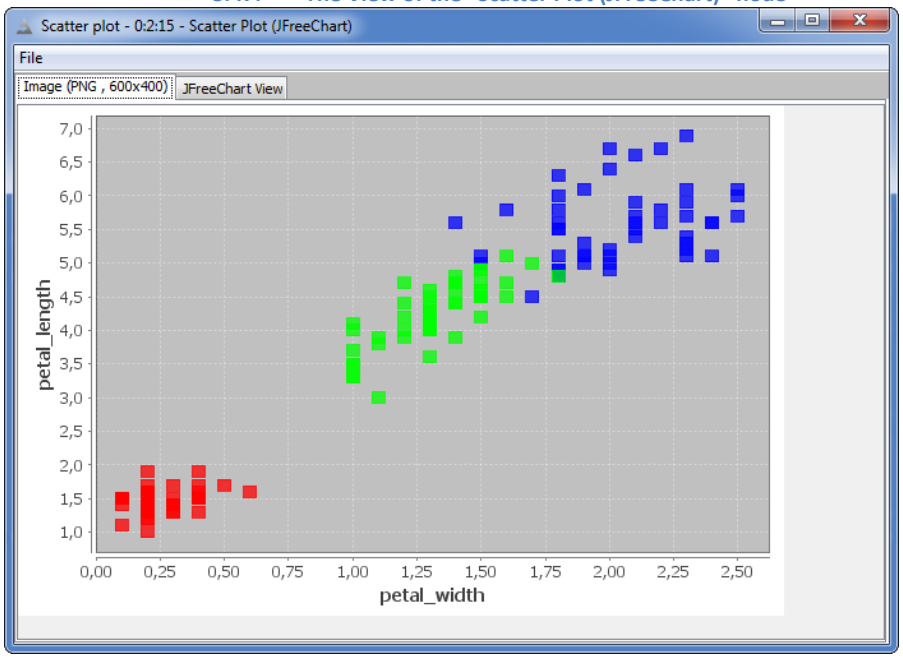
The “General Plot Options” tab sets all plotting parameters:

- Graphical output type
- Width and Height
- Title
- Background Properties
- Font Properties

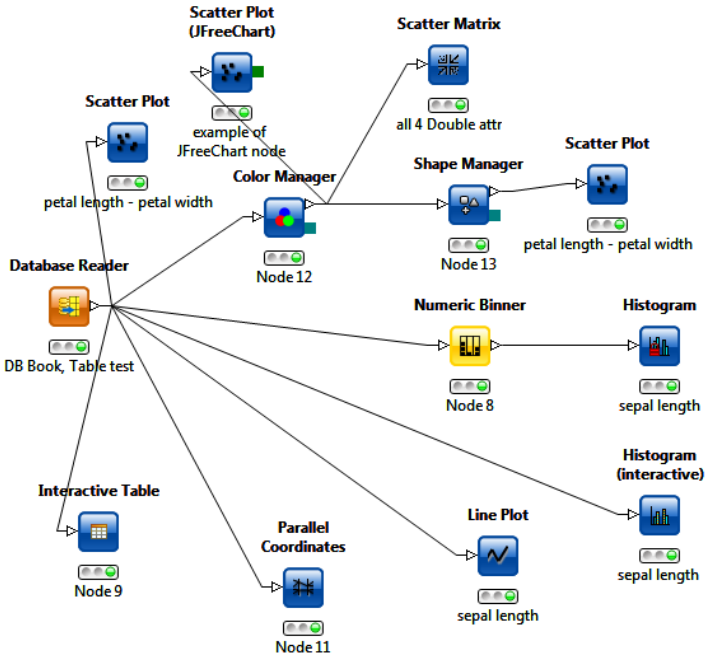
The JFreeChart nodes do not offer an interactive output view. They all produce an image at the output port and a view to display the static image that was produced. You cannot play with the data displayed in the image. This however has good and bad sides. On one side you need to know since the beginning what chart has to be generated and on which input features. On the other side, the image is fixed and ready to be exported for example in a report. Nevertheless these nodes keep a view feature, but the only interactivity in the view window concerns the image export type.

As an example, let's produce here a JFreeChart scatter plot and let's compare it with the scatter plot obtained with the corresponding node in the "Data Views" category.

3.47. The View of the "Scatter Plot (JFreeChart)" node



3.48. The final version of the "my_first_data_exploration" workflow



The view of the image produced by the "Scatter Plot (JFreeChart)" node can be seen in the figure 3.47 and compared with figure 3.43. It is of course the same scatter plot, but it is not interactive and its plot settings are saved with the configuration status of the node. The image displayed with this view is available at the output port of the JFreeChart node.

The final workflow "my_first_data_exploration" is shown in figure 3.48.

3.11. Exercises

Exercise 1

Read file “yellow-small.data” from the Balloons Data Set (you can find this file in the book’s Download Zone or you can download it from: http://archive.ics.uci.edu/ml/datasets.html).

This file has 5 columns: “Color”, “Size”, “Act”, “Age”,” Inflated (True/False)”. Rename the data columns accordingly.

Add the following classification column:

IF Color = yellow AND Size = Small => class =inflated

ELSE class = not inflated

Add a final column called “full sentence” that says:

“inflated is T”

OR

“not inflated is F”

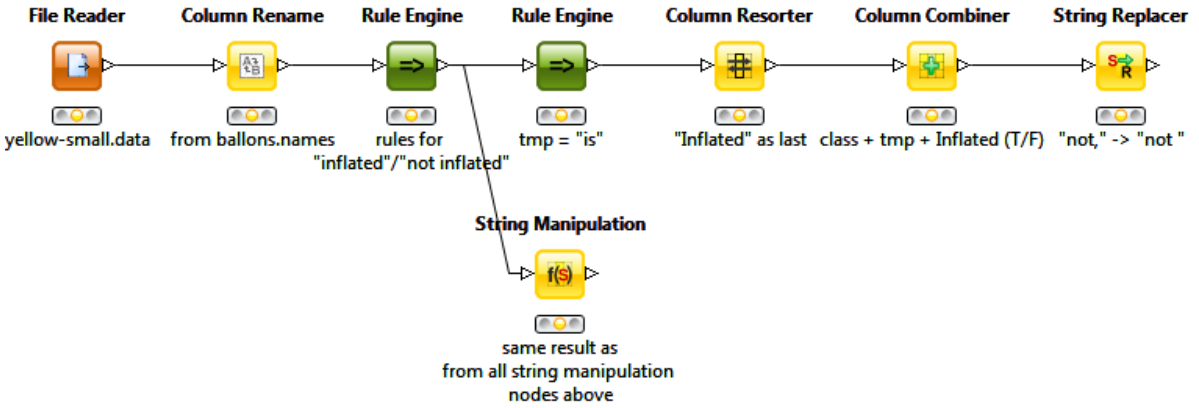
where “inflated/not inflated” comes from the “class” column and “T/F” from the “Inflated (True/False)” column.

Solution to Exercise 1

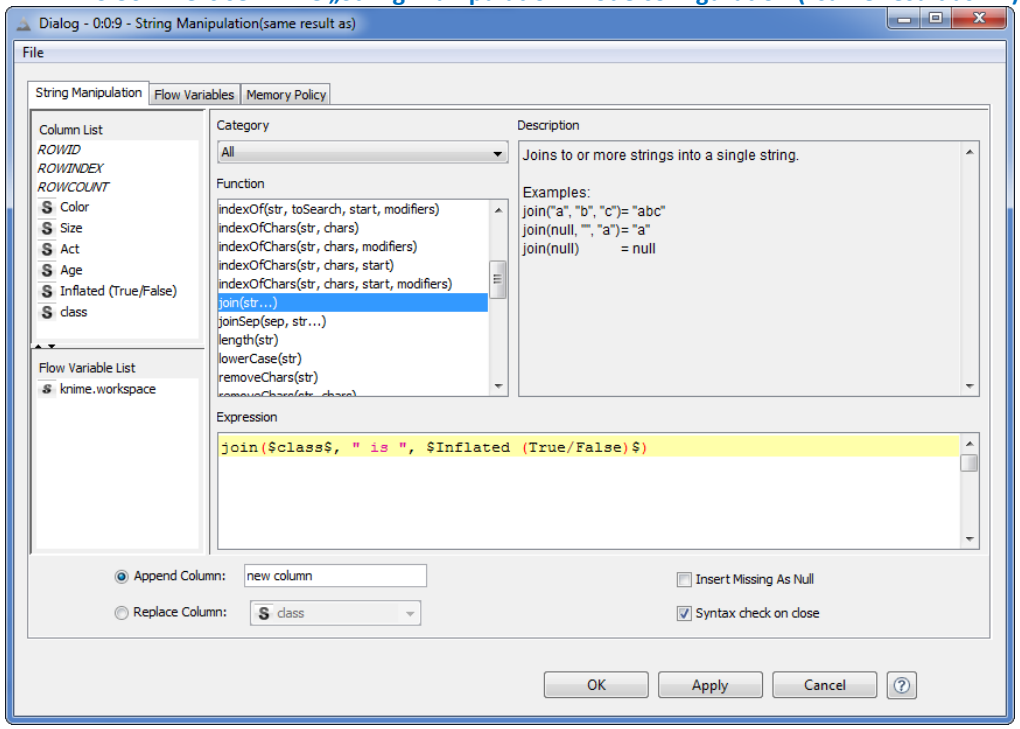
There are two ways to proceed in this exercise.

1. With a series of dedicated “String Manipulation” and “Rule Engine” nodes
2. With one “Rule Engine” and “String Manipulation” node and its functions

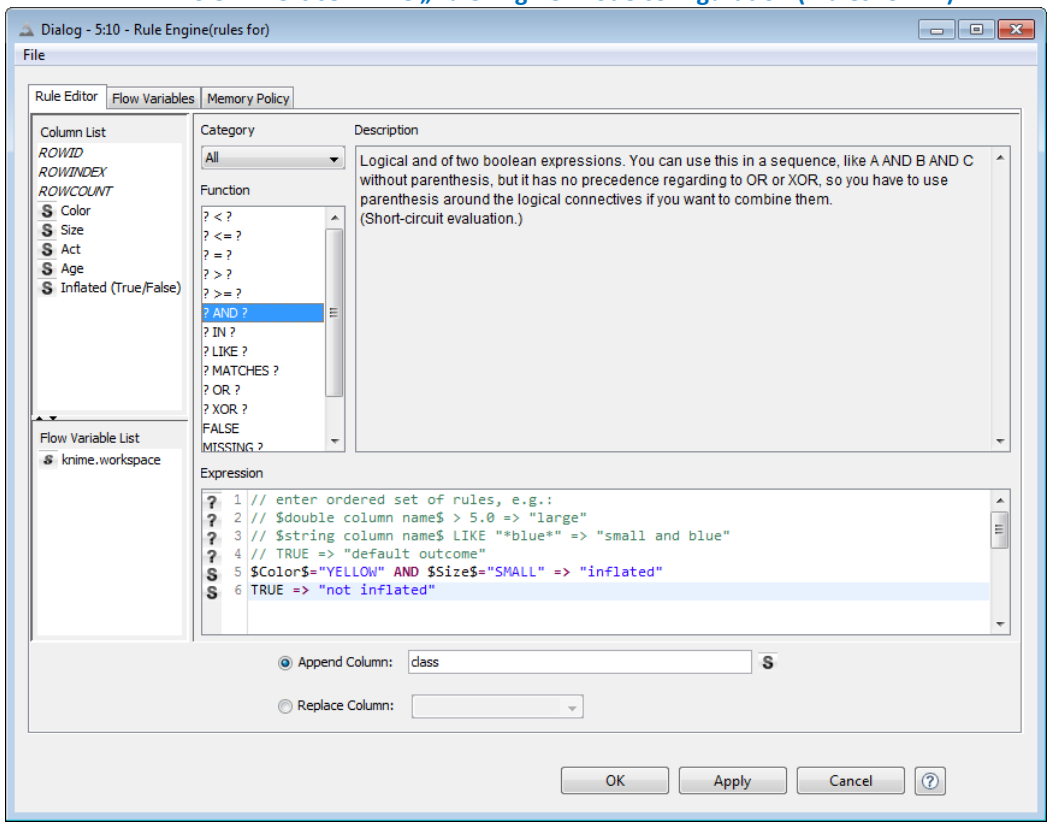
3.49. Exercise 1: Workflow



3.50. Exercise 1: The „String Manipulation“ node configuration (“same result as …”)



3.51. Exercise 1: The „Rule Engine“ node configuration (“rules for …”)



Exercise 2

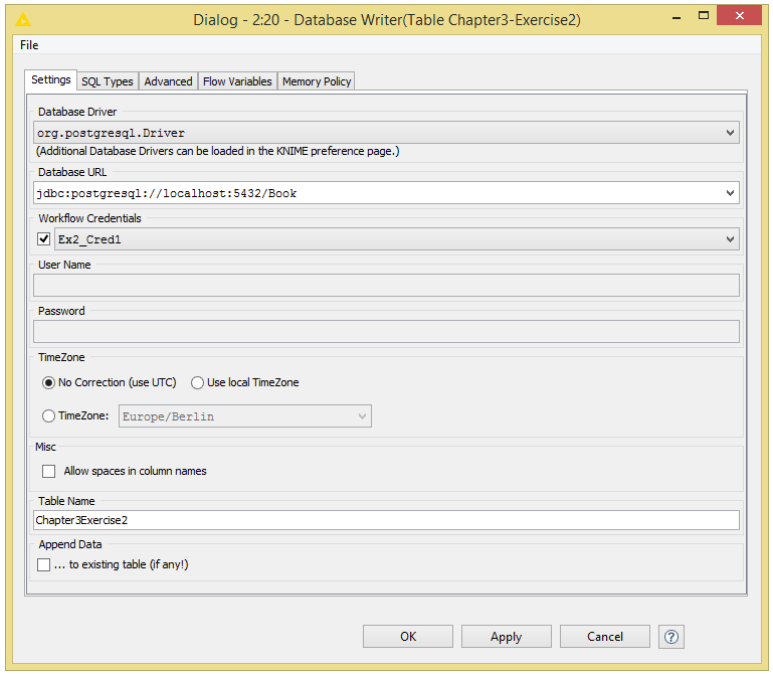
This exercise is an extension of Exercise 1 above.

Download and install the latest version of a common database (mySQL: http://dev.mysql.com/downloads/ or PostrgreSQL: http://www.postgresql.org/download/)

Write the last data table of workflow Exercise 1 into a table called “Chapter3Exercise2” in a database “Book”.

Solution to Exercise 2

3.52. Exercise 2: Configuration Window of the Database Writer node for PostGreSQL



Exercise 3

From the **adult.data** input data,
Display three histograms:

"Age" histogram
"Work class" histogram
"Average(capital gain)" vs. "work class"

Build these histograms using a “Histogram” node and a “Histogram (Interactive)” node
Plot age vs. hours per week in a scatter plot
Find the outlier points in terms of age and hilite them
Check how many outliers there are and how high their age is

Solution to Exercise 3

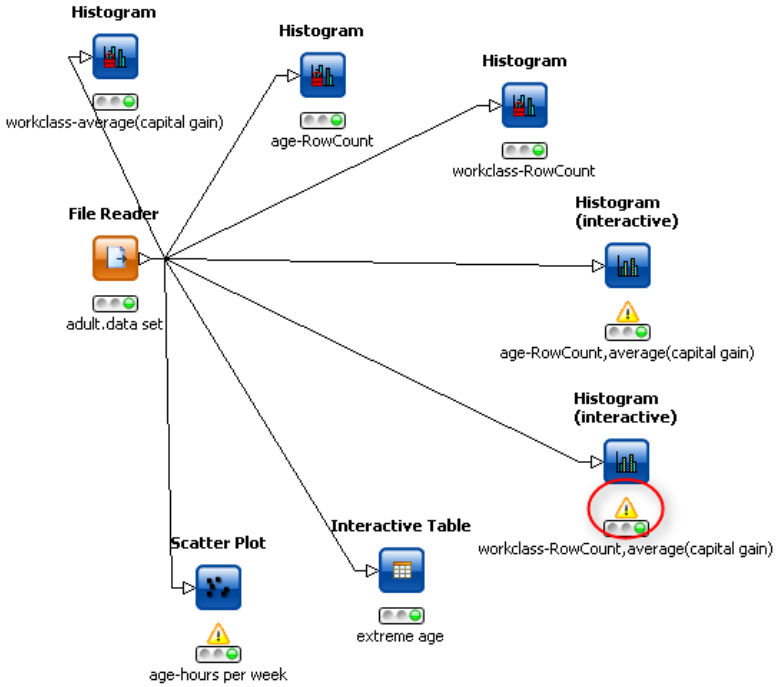
Figure 3.53 shows the workflow built for Exercise 3.

Notice the yellow triangle under some nodes.

If you hover on it, you will see that only some rows have been imported into the “View”. This is because some nodes limit the rows in their default settings to only a few thousand, in order to improve the “View” performance.

With respect to the outliers analysis, in the first 2500 rows there are 4 people of age 90 who still work in the “Private” category. These records are located at Row222, Row1040, Row1935, and Row2303.

3.53. Exercise 3: Workflow



Chapter 4. My First Data Model

4.1. Introduction

We have finally reached the heart of the KNIME platform: data modeling. There are two categories of nodes in the “Node Repository” panel fully dedicated to data modeling: “Statistics” and “Mining”.

The “Statistics” category contains the nodes to model a linear and a polynomial regression and a few more nodes to calculate statistical parameters and perform statistical tests.

The “Mining” category contains data mining algorithms, from Artificial Neural Networks to Bayesian Classifiers, from clustering to Support Vector Machines, and more.

Data modeling algorithms are usually implemented by two nodes: a “Learner” node and a “Predictor” node.

The “Learner” node reproduces the learning phase of the algorithm on a dedicated learning data set. The “Predictor” node classifies new unknown data by using the model learned by the “Learner” node. For example, the “Mining” -> “Bayes” category implements naïve Bayesian classifiers. The “Naïve Bayes Learner" node learns the Bayes rules on the learning (or training) data set and stores them in the model. The “Naïve Bayes Predictor” node reads the Bayes rules from the model and applies them to new data.

Therefore for most data modeling algorithms we need a learning set on which the model can learn its rules. We also need a test set of new data to test how well the model generalizes to unknown data. We need to divide the original data set in two smaller data sets: the learning data set and the test set. To partition, reorganize, and re-unite the data sets we use nodes from the “Data Manipulation” -> “Row” -> “Transform" category.

Sometimes problems can be incurred when there are missing values in the data. Not all data modeling algorithms can deal with missing data. The model might also require that the data set follows a normal distribution. To remove missing data from the data sets and to normalize values in a column, we can use more nodes from the category “Data Manipulation” -> “Column” -> “Transform”.

In this chapter, we will provide an overview of data modeling nodes, Learner nodes, Predictor nodes, and nodes to manipulate rows and transform values in columns. We will work on the adult data set that we have already seen in the previous chapters. Let’s create a new workflow group “Chapter4” and, inside that, a new workflow called “data_preparation”. We will use this workflow to prepare the data for further data modeling operations. The first step of this workflow is to read the adult data set with a “File Reader” node.

4.2. Split and Combine Data Sets

Since many models need a learning set and a separate test set, these two data sets have to be set up before we can start modeling the data. In order to extract two data sets - one for training and one for testing - from the original data set, we can apply the "Partitioning" node. If we only need a training set and not a test set and if we do not need to use the whole data set, we can apply the "Row Sampling" node.

Row Sampling

The "Row Sampling" node extracts a sample (= a subset of rows) from the input data.

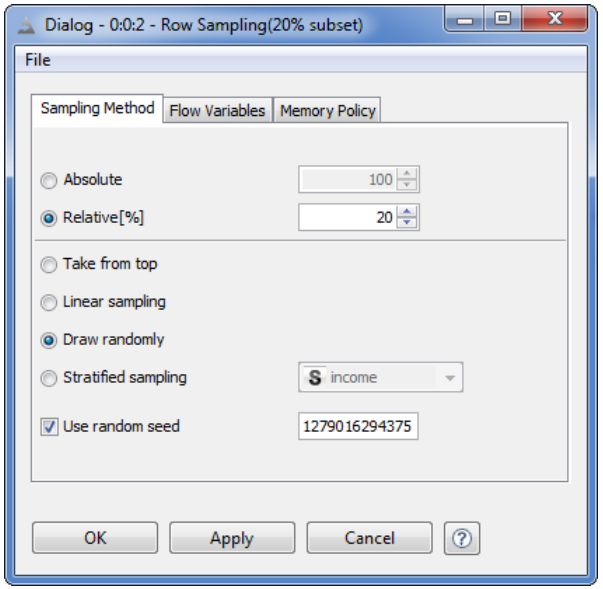
The "Row Sampling" node is located in the "Node Repository" panel in the `"Data Manipulation" -> "Row" -> "Transform"` category.

The configuration window enables you to specify:

- The sample size as an absolute number of rows or as a percentage of the original data set;
- The extraction mode
  - "Take from the top" means the first rows of the original data set
  - "Linear Sampling" takes the first and the last row and samples between rows at regular steps
  - "Draw randomly" extracts rows at random
  - "Stratified sampling" extracts rows whereby the distribution of values in the selected column is approximately retained in the output table

For "Draw randomly" and "Stratified sampling" a random seed can be defined so that the random extraction is reproducible (you never know when you need to recreate the same learning set).

4.1. Configuration window for the "Row Sampling" node



Here, we selected a size of 20% of the original data set for the learning set. Rows were extracted randomly from the original data set. A size of 20% of the original data set is probably too small; afterwards we should check that all the work classes we want to predict are actually represented in the learning set.

Note. The “Row Sampling” node only produces one data subset that we can use either to train or to test a model, but not both. If we want to generate two data subsets, the first one according to our specifications in the configuration window, and the second one with the remaining rows, we need to use the “Partitioning” node.

Partitioning

The “Partitioning” node performs the same task as the “Row Sampling” node: it extracts a sample (= a subset of rows) from the input data. It also builds a second data set with the remaining rows and makes it available at the other output port.

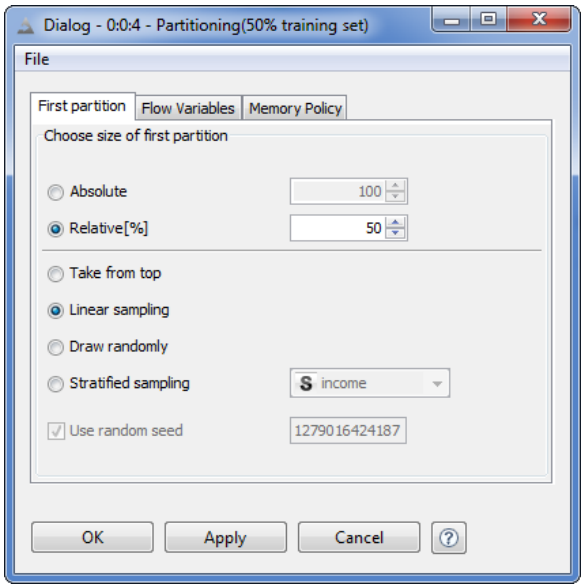
The “Partitioning” node is located in the “Node Repository” panel in the category: `“Data Manipulation” -> “Row” -> “Transform”.`

The configuration window enables you to specify:

- The sample size as an absolute number of rows or as a percentage of the original data set;
- The extraction mode
  - “Take from the top” means the first rows of the original data set
  - “Linear Sampling” takes the first and the last row and samples between rows at regular steps
  - “Draw randomly” extracts rows at random
  - “Stratified sampling” extracts rows whereby the distribution of values in the selected column is approximately retained in the output table

For “Draw randomly” and “Stratified sampling” a random seed can be defined so that the random extraction is reproducible (you never know when you need to recreate the same learning set).

4.2. Configuration window of the "Partitioning" node



Here, we selected a size of 50% of the original data set for the learning set plus a linear extraction mode. The learning set was made available at the output port on the top; the remaining data were made available at the output port on the bottom.

In the linear sampling technique, rows adhere to the order defined in the original data set. Sometimes it is not advisable to present rows to a model's Learner node in a specific order; otherwise the model might learn the row order among all other underlying patterns. To be sure that data rows are presented to the model's Learner node in a random order, we can extract them in the random mode or apply the "Shuffle" node.

Shuffle

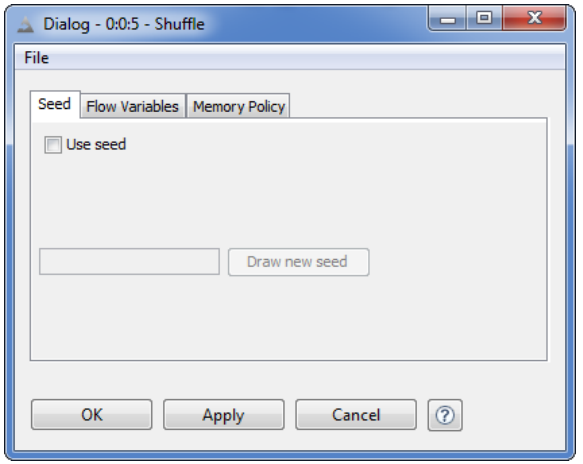
The "Shuffle" node shuffles the rows of the input table putting them in a random order.

The "Shuffle" node is located in the "Node Repository" panel in the `"Data Manipulation" -> "Row" -> "Transform"` category.

In general, the "Shuffle" node does not need to be configured. If we want to be able to repeat exactly the same random shuffling of the rows, we need to use a seed, as follows:

- Check the "Use seed" flag
- Click the "Draw new seed" button to create a seed for the random shuffling and recreate it at each run

4.3. Configuration window for the "Shuffle" node



We only applied the "Shuffle" node to the training set. It does not make a difference whether the data rows of the test set are presented into a pre-defined order or not.

Now we have a learning set and a test set. But what if we want to recreate the original dataset by reunifying the training and the test set? KNIME has a "Concatenate" node that comes in handy for this task.

Concatenate

The “Concatenate” node has two input ports, each one for a data set. The “Concatenate” node appends the data set at the second port (at the bottom) to the data set at the first port (at the top).

The “Concatenate” node is located in the “Node Repository” panel in category `“Data Manipulation” -> “Row” -> “Transform”.`

The configuration window deals with the following:

- What to do with rows with the same ID
  - skip the rows from the second data set
  - rename the RowID with an appended suffix
  - abort execution with an error (This option can be used to check for unique RowIDs)
- Which columns to keep
  - all columns from the second and first data set (union of columns)
  - only the intersection of columns in the two data sets (i.e. columns contained in both tables)
- Hiliting of data

4.4. Configuration window for the "Concatenate" node

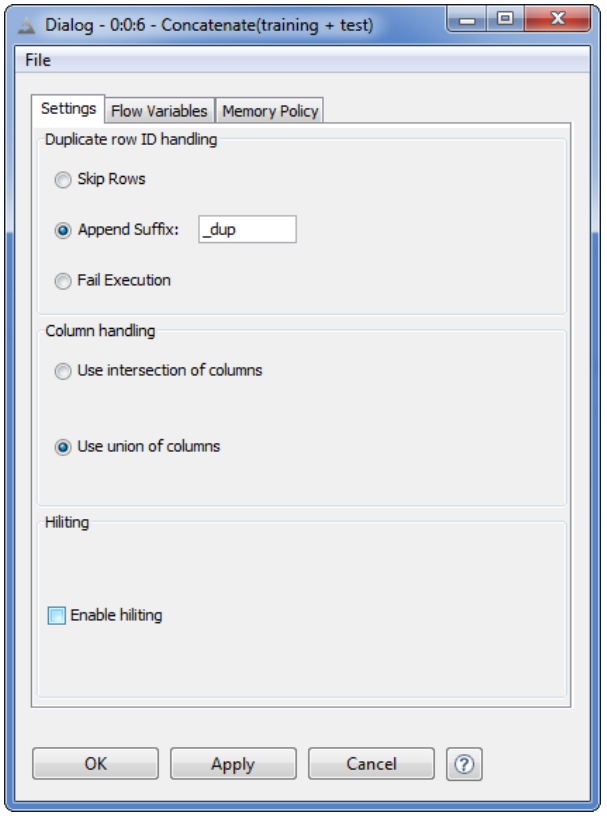


Figure 4.5 shows an example of how the “Concatenate” node works, when the following options in the configuration window are enabled:

- append suffix to RowID in rows with duplicate RowID
- use union of columns

4.5. This is an example of how the "Concatenate" node works

First Data Table

RowID	scores
Row1	22
Row3	14
Row4	10

Second Data Table

RowID	name	scores
Row1	The Black Rose	23
Row2	Cynthia	2
Row5	Tinkerbell	4
Row6	Mother	6
Row7	Augusta	8
Row8	The Seven Seas	3

***Concatenated** Table*

RowID	name	scores
Row1	?	22
Row3	?	14
Row4	?	10
Row1_dup	The Black Rose	23
Row2	Cynthia	2
Row5	Tinkerbell	4
Row6	Mother	6
Row7	Augusta	8
Row8	The Seven Seas	3

4.3. Transform Columns

In this section we explore two more data manipulation nodes that can help to prepare the data for the model.

The first node deals with missing values. KNIME data cells can have a special “missing value” status. By default, missing values are displayed in the table view with a question mark (“?”). However, not all nodes, including models, plots, and data manipulation nodes, can handle missing values appropriately. The “Missing Value” node replaces all missing values with values of your choice.

Missing Value

The “Missing Value” node replaces all missing values in a data set with values of your choice.

The “Missing Value” node is located in the “Node Repository” panel in the `“Data Manipulation” -> “Column” -> “Transform”` category.

In **tab “Default”**, replacement values can be defined separately for columns of Integer, Double, and String type and applied to the whole table (Fig. 4.6). The configuration settings require the replacement values for missing values in Integer columns, missing values in Double columns, and missing values in String columns.

The configuration window also allows the removal of rows with missing values in columns where the type is “Unknown”.

In **tab “Individual”**, a replacement value can be defined for each column and applied only to that column (Fig.4.7).

To define the missing value in a column:

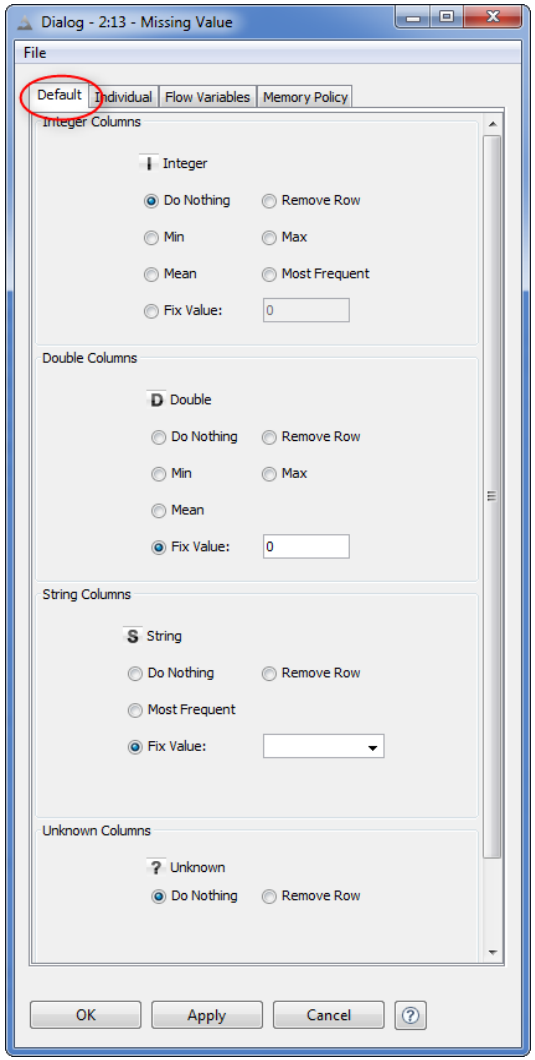
- Double-click the column in the list on the left

OR

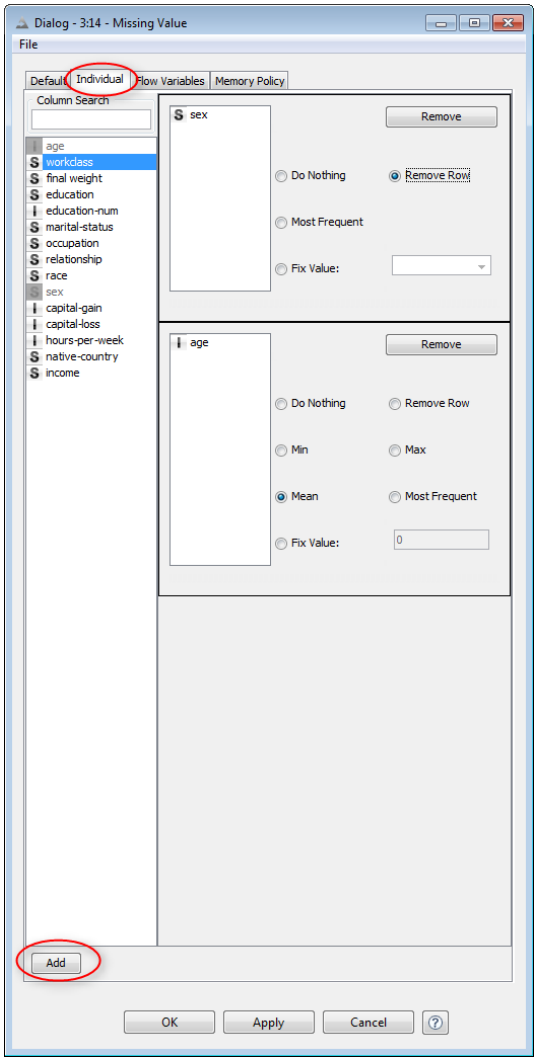
- Select the column from the list on the left
- Click the “Add” button at the bottom

Then, select the desired missing value handling strategy. A “Column Search” box is provided to help to find columns among many. A “Remove” button is also provided in the data column frame to remove the individual missing value handling strategy for the selected column.

4.6. Configuration window for „Missing Value“ node: Tab “Default”



4.7. Configuration window for „Missing Value“ node: Tab “Individual”



We used the following replacement values for the missing values: 0 for Double and Integer data cells and an empty string for String data cells. We applied the "Missing Value" node after the "File Reader" node. However, we did not use the results because the data models, that we are going to use in this chapter, can deal with missing values. We introduced the "Missing Value" node here for demonstrative purposes only.

Sometimes a data model requires normalized input attribute values, for the data to be either normalized to follow the Gaussian distribution or just to fall into the [0,1] interval. In order to comply with this requirement, we use the "Normalizer" node.

Normalizer

The "Normalizer" node normalizes data; i.e. it transforms the data to fall into a given interval or to follow a given statistical distribution.

The "Normalizer" node is located in the "Node Repository" panel in the `"Data Manipulation" -> "Column" -> "Transform"` category.

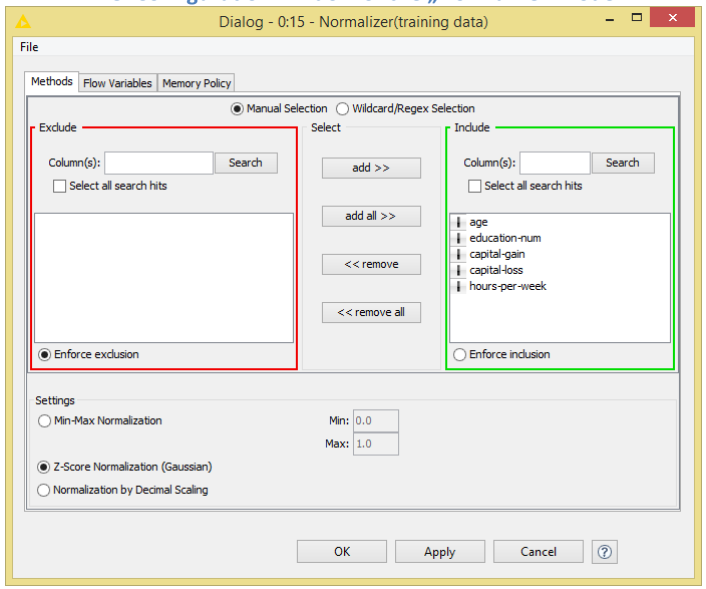
The configuration window requires:

- the normalization method
- the list of numerical data columns to be normalized

The column selection is performed by means of an "Exclude"/"Include" frame, by manual selection or Wildcard/RegEx selection. For manual selection:

- The columns to be normalized are listed in the "Normalize" frame. All other columns are listed in the "Do not normalize" frame.
- To move from frame "Normalize" to frame "Do not normalize" and vice versa, use buttons "add" and "remove". To move all columns to one frame or the other use buttons "add all" and "remove all".

4.8. Configuration window of the „Normalizer" node



The "Normalizer" node has 2 output ports:

- At the first port we find the normalized data
- At the second port the transformation parameters are provided to repeat the same normalization on other data (green square port)

Note. Triangular ports offer/read data. Squared ports offer/read parameters: model's parameters, normalization parameters, transformation parameters, color/shape parameters, etc …

Normalization Methods

Min-Max Normalization
This is a linear transformation whereby all attribute values in a column fall into the [min, max] interval and min and max are specified by the user.

Z-score Normalization
This is also a linear transformation whereby the values in each column are Gaussian-(0,1)-distributed, i.e. the mean is 0.0 and the standard deviation is 1.0.

Normalization by Decimal Scaling
The maximum value in a column is divided j-times by 10 until its absolute value is smaller or equal to 1. All values in the column are then divided by 10 to the power of j.

Normalizer (Apply)

This "Normalizer (Apply)" node normalizes data; that is it transforms data to fall into a given interval or to follow a given statistical distribution. It does not calculate the transformation parameters though; it obtains them from a "Normalizer" node previously applied to a similar data set.

The "Normalizer (Apply)" node has two input ports: one for the data to be normalized and one for the normalization parameters.

The "Normalizer(Apply)" node is located in the "Node Repository" panel in the `"Data Manipulation" -> "Column" -> "Transform"` category.

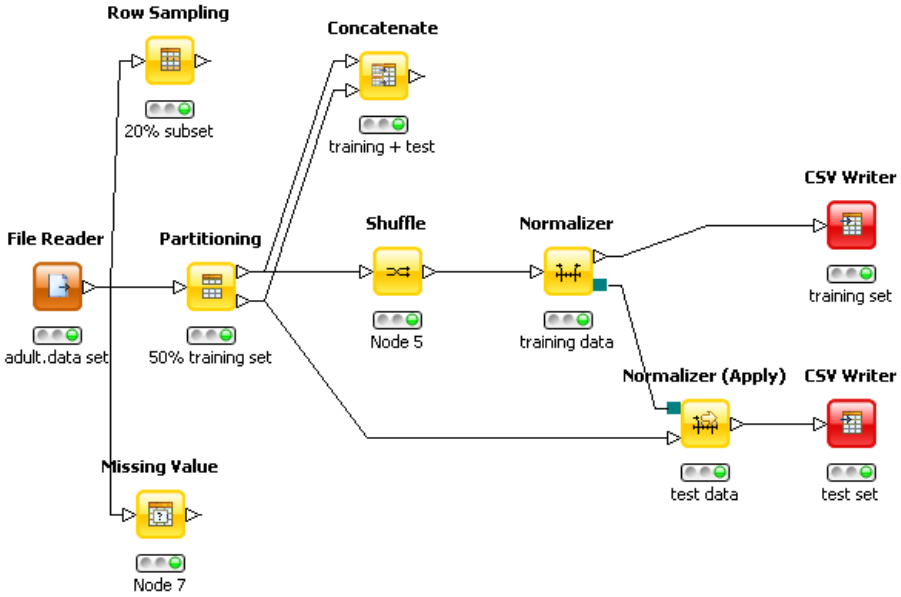
No additional configuration is required.

We applied the "Normalizer" node to the training set from the output port of the "Partitioning" node, in order to normalize the training set and to define the normalization parameters.

We then created a “Normalizer (Apply)” node to read the normalization parameters defined by the previous “Normalizer” node and to normalize the remaining data from the “Partitioning” node (2nd output port).

Let’s now write the processed training set and test set into a CSV file, “training_set.csv” and “test_set.csv” respectively. We used two “CSV Writer” nodes: one to write the training set into “training_set.csv” file and one to write the test set into “test_set.csv” file. These last 2 nodes conclude the “data_preparation” workflow that you can see depicted in figure 4.9.

4.9. The “data_preparation” workflow



4.4. Data Models

Now let’s create a new workflow and call it “my_first_data_model”. We will use this workflow to show how data models can be trained on a set of data and then applied to new data. To give an overview we will go through some standard data analysis method paradigms. Standard here refers to the way the paradigms are implemented in KNIME -- for example with one node as the Learner and a separate node as the Predictor/Applier -- rather than with regard to the quality of the algorithm itself.

The first two nodes in this new workflow are two “File Reader” nodes: one to read the training set and one to read the test set that was saved in two CSV files in the “data_preparation” workflow at the end of the last section.

In this workflow “my_first_data_model”, we want to predict the “workclass” label of the adult data set by using a few different data models. This section does not intend to compare those data models in terms of accuracy or performance. Indeed not much work has been spent to optimize these models to become the most accurate predictors. Contrarily, the goal here is to show how to create and configure such models. How to optimize the model parameters to ensure that they will be as accurate as possible is a problem that can be explored elsewhere [3] [4] [5].

In every prediction/classification problem, you need a training set with a number of attributes where each row of the training set has been manually assigned to a given class. These output classes of the data rows are contained in a column of the data set: this is the class column.

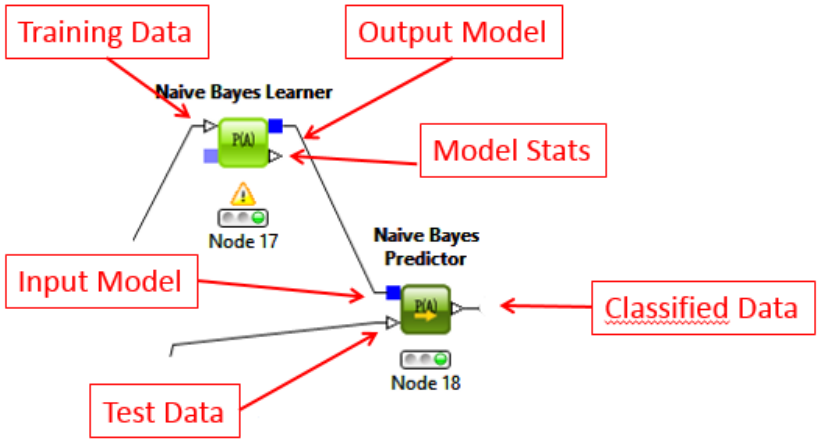
Most data models consist of two nodes: a Learner and a Predictor.

The Learner node defines the model’s parameters and rules that make the model suitable to perform a given classification/prediction task. The Learner node uses the input data table as the training set to define these parameters and rules. The output of this node is then a set of rules and/or parameters: the model.

The Predictor node uses the model built in the previous step and applies it to a set of unknown (i.e. new unclassified) data to perform exactly the classification/ prediction task for which it was built.

The Learner node requires a data table as input and provides a model as output. The output port of the Learner node is represented as a green square rather than a white triangle. This is the symbol for a model port.

4.10. Learner and Predictor Nodes



The Predictor node takes a data table and a model at the input ports (a white triangle for the data and a green square for the model) and provides a data table containing the classified data on the output port.

Naïve Bayes Model

Let’s start with a naïve Bayes model. A Bayesian model defines a set of rules, based on the Gaussian distributions and on the conditional probabilities of the input data, to assign a data row to an output class [3][4][5]. In the “Node Repository” panel in the “Mining” -> “Bayes” category we find two nodes: “Naïve Bayes Learner" and “Naïve Bayes Predictor”.

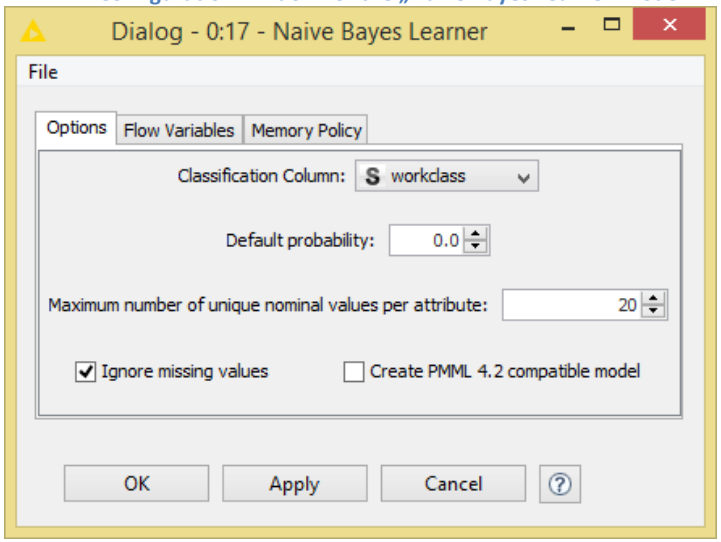
Naïve Bayes Learner

The "Naïve Bayes Learner" node creates a Bayesian model from the input training data. It calculates the distributions and probabilities to define the Bayesian model's rules from the training data. The output ports produce the model and the model parameters respectively.

In the configuration window you need to specify:

- The class column (= the column containing the classes)
- How to deal with missing values (skip vs. keep)
- The maximum number of unique nominal values allowed per column. If a column contains more than this maximum number of unique nominal values, it will be excluded from the training process.
- Compatibility of the output model with PMML 4.2

4.11. Configuration window for the „Naive Bayes Learner" node



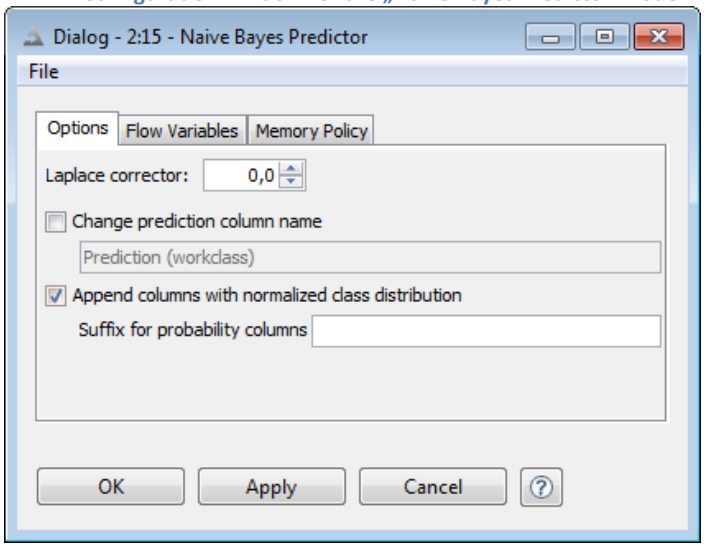
Naïve Bayes Predictor

The "Naïve Bayes Predictor" node applies an existing Bayesian model to the input data table.

In the configuration window you can:

- Append the normalized class distribution values for all classes to the input data table
- Use a Laplace corrector as the initial count for nominal columns with 0 count. A 0 value indicates no correction.
- Customize the column name for the predicted class

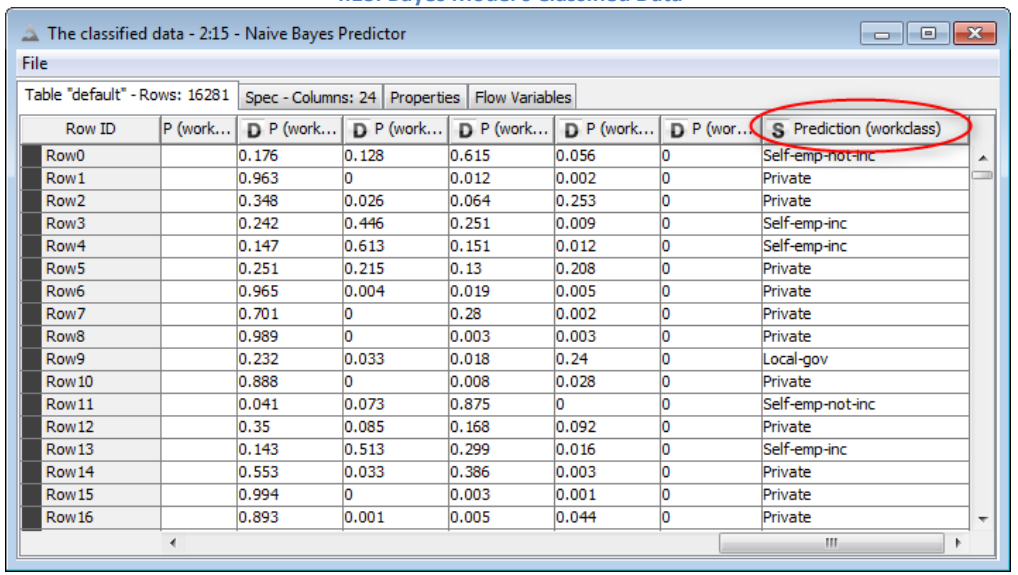
4.12. Configuration window for the „Naive Bayes Predictor" node



In the “my_first_data_model” workflow we connected a “Naïve Bayes Learner” node to the “File Reader” node that reads the training data set and it is named “training set”. In the configuration window of the “Naïve Bayes Learner”, we specified “workclass” as the class column, we opted to skip rows with missing values in the model estimation and to skip a column if more than 20 nominal values were found.

After this configuration, a yellow triangle appears under the “Naïve Bayes Learner” to say that column “native country” in the input data set has too many (> 20) nominal values and will be ignored. We can then run the “Execute” option for the “Naïve Bayes Learner” node.

4.13. Bayes Model's Classified Data


The classified data - 2:15 - Naive Bayes Predictor

File

Table "default" - Rows: 16281 | Spec - Columns: 24 | Properties | Flow Variables

Row ID	P (work...	P (work...	P (work...	P (work...	P (work...	P (wor...	Prediction (workclass)
Row0		0.176	0.128	0.615	0.056	0	Self-emp-not-inc
Row1		0.963	0	0.012	0.002	0	Private
Row2		0.348	0.026	0.064	0.253	0	Private
Row3		0.242	0.446	0.251	0.009	0	Self-emp-inc
Row4		0.147	0.613	0.151	0.012	0	Self-emp-inc
Row5		0.251	0.215	0.13	0.208	0	Private
Row6		0.965	0.004	0.019	0.005	0	Private
Row7		0.701	0	0.28	0.002	0	Private
Row8		0.989	0	0.003	0.003	0	Private
Row9		0.232	0.033	0.018	0.24	0	Local-gov
Row10		0.888	0	0.008	0.028	0	Private
Row11		0.041	0.073	0.875	0	0	Self-emp-not-inc
Row12		0.35	0.085	0.168	0.092	0	Private
Row13		0.143	0.513	0.299	0.016	0	Self-emp-inc
Row14		0.553	0.033	0.386	0.003	0	Private
Row15		0.994	0	0.003	0.001	0	Private
Row16		0.893	0.001	0.005	0.044	0	Private

The next step involves connecting a “Naïve Bayes Predictor” node to the “File Reader” node to read the test set – named “test set” – through the data port; the “Naïve Bayes Predictor” node is then also connected to the output port of the “Naïve Bayes Learner” node through the model port.

After execution, the “Naïve Bayes Predictor” shows a new column appended to the output table: “Prediction(workclass)”. This column contains the class assignments for each row performed by the Bayesian model. How correct these assignments are, that is how good the performance of the model is, can only be evaluated by comparing them with the original labels in “workclass”.

If the flag to append the probability values for each output class was enabled, in the final data table there will be as many new columns as there are values in the class column; each column contains the probability for a given class value according to the trained Bayesian model.

KNIME has a whole “Mining” ->“Scoring” category with nodes that measure the classifiers’ performances. We will use the “Scorer” node to measure the performances of the Bayesian classifier.

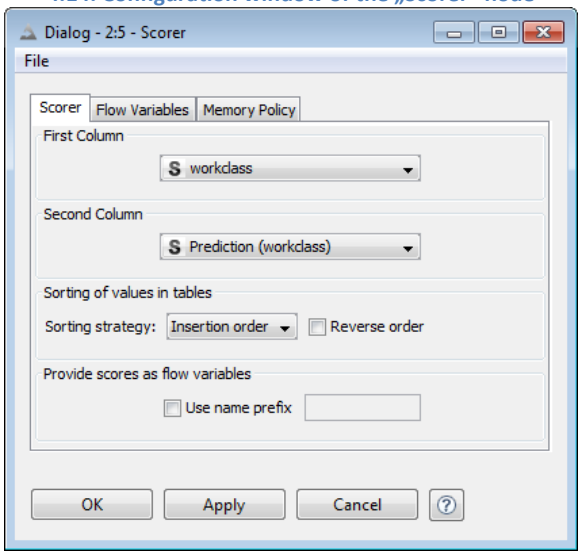
Scorer

The “Scorer” node is located in the “Node Repository” panel in the `“Mining” -> “Scoring”` category. It compares the values of two columns (reference column and prediction column) in the data table and shows the confusion matrix and some accuracy measures.

The “Scorer” node has a View option, where the confusion matrix is displayed and the “Hilite” functionality is available. Thus, the user can hilite the cells of the confusion matrix to isolate the underlying rows and to see them, for example, by means of an “Interactive Table” node.

The configuration window requires the selection of the two columns to compare. It also provides a flag to enable the storing of the score values as flow variables (flow variables though are not explained in this beginner’s book).

4.14. Configuration window of the „Scorer“ node



We added a “Scorer” node into the workflow “my_first_data_model”. The node is connected to the data output port of the “Naïve Bayes Predictor”. The first column with the original reference values is “workclass”; the second column with the class estimation is the column called “Prediction(workclass)” which is produced by the “Naïve Bayes Predictor” node.

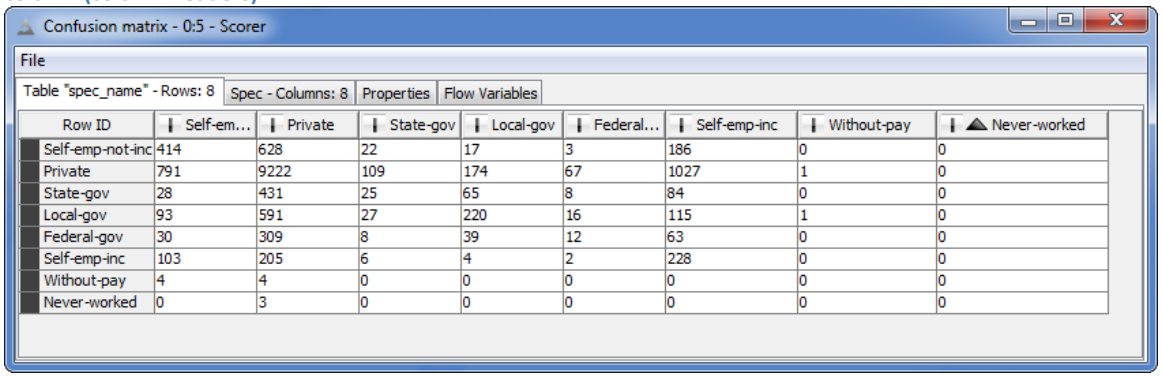
We can see the confusion matrix and the accuracy measures for the compared columns by selecting either the last two items or item “View Confusion Matrix” in the context-menu of the “Scorer” node.

Confusion Matrix

In Figure 4.15, you can see the confusion matrix generated by the "Scorer" node. The confusion matrix shows how many rows have a given value in the first column (reference column) and another given value in the second column (prediction column).

The values found in the first column are reported as RowIDs; the values found in the second column are reported as column headers.

4.15. Confusion Matrix from the "Scorer" node. The confusion matrix shows the number of rows belonging to one class in the reference column "Workclass" (RowIDs) and assigned to the same or a different class in the prediction column (column headers).



Row ID	Self-em...	Private	State-gov	Local-gov	Federal...	Self-emp-inc	Without-pay	Never-worked
Self-emp-not-inc	414	628	22	17	3	186	0	0
Private	791	9222	109	174	67	1027	1	0
State-gov	28	431	25	65	8	84	0	0
Local-gov	93	591	27	220	16	115	1	0
Federal-gov	30	309	8	39	12	63	0	0
Self-emp-inc	103	205	6	4	2	228	0	0
Without-pay	4	4	0	0	0	0	0	0
Never-worked	0	3	0	0	0	0	0	0

In our "Scorer" node, we selected the first column as the target classification column "workclass" and the second column as the output column of the Bayesian classifier. Thus, this confusion matrix says that, for example, 780 data rows were classified as "Private" and belong to the class "self-employed-not-incorporated" in the original data set; 9750 data rows were correctly classified as "Private"; 60 data rows were classified as "state-government employees" while they were actually "self-employed-not-incorporated" … and so on. The numbers in the diagonal cells of the confusion matrix show the correct classifications. The higher the diagonal numbers are, the better the performance of the classifier.

The sum across one row of the confusion matrix indicates the total number of data in one class according to the original data set classification. The sum across one column indicates the number of data rows assigned to one class by the Bayesian model. The sum of all columns and the sum of all rows must therefore be the same, since they represent the total number of data.

Accuracy Measures

The second port of the "Scorer" node presents a number of accuracy measures [6] [7].

True Positives is the number of data rows belonging to a class in the original data set and correctly classified as belonging to that class.
True Negatives is the number of data rows that do not belong to a class in the original data set and are classified as belonging to a different class.
False Positives is the number of data rows that do not belong to a class but are classified as if they do.
False Negatives is the number of data rows that belong to a class but are assigned to a different class by the model.

4.16. True Positives, False Negatives, True Negatives, and False Positives in the Confusion Matrix for the class "Self-emp-not-inc".

If we consider the confusion matrix shown in Figure 4.16, for each class we can see that:

The *True Positives* are in the diagonal cells, where the number of data rows is correctly classified

The *False Negatives* are in all the other cells in a row besides the diagonal cell

The *False Positives* are all the numbers in a column excluding the diagonal cell

The *True Negatives* are in all the remaining cells

In the second output port, the "Scorer" node provides a few more accuracy measures derived from the True Positives, True Negatives, False Positives, and False Negatives.

Sensitivity = True Positives / (True Positives + False Negatives)
Specificity = True Negatives / (True Negatives + False Positives)

"Sensitivity" measures the model's capability to recognize one class correctly. If all instances of a given class are recognized correctly, the result is 0 "False Negatives" for that class; which means that no items of that class are assigned to another class. "Sensitivity" is then 1.0 for that class.

"Specificity" measures the model's capability of recognizing what does not belong to a given class. If the model recognizes what does not belong to that class, the result is 0 "False Positives"; which means no extraneous data rows are misclassified in my class. "Specificity is then 1.0 for that class.

In a two-class problem, "Sensitivity" and "Specificity" are used to plot the ROC Curves (see "ROC Curve" later on in this section).

Recall = True Positives / (True Positives + False Negatives) = Sensitivity
Precision = True Positives / (True Positives + False Positives)

"Precision" and "Recall" are two widely used statistical accuracy measures. "Precision" can be seen as a measure of exactness or fidelity, whereas "Recall" is a measure of completeness.

In a classification task, the "Precision" for a class is the number of "True Positives" (i.e. the number of items correctly labeled as belonging to that class) divided by the total number of elements labeled as belonging to that class. "Recall" is defined as the number of "True Positives" divided by the total number of elements that actually belong to that class. "Recall" has the same definition as "Sensitivity".

F-measure = 2 x Precision x Recall / (Precision + Recall)

The F-measure can be interpreted as a weighted average of "Precision" and "Recall", where the F-measure reaches its best value at 1 and worst score at 0.

Accuracy = (Sum(TP)+ Sum (TN)) / (Sum(TP)+ Sum(FP)+ Sum(FN)+ Sum(TN))

being TP = True Positives, FP = False Positives, TN = True Negatives, and FN = False Negatives.

Cohen's Kappa is a measure of inter-rater agreement as [(Sum(TP)+Sum(FP)) – (P(chance))] / (1-P(chance)) with P(chance) coming from the probability of positive events (one of the rater) and the probability of true events (the other rater).

"Accuracy" is an *overall measure* and is calculated across all classes. An accuracy of 1.0 means that the classified values are exactly the same as the original class values.

All these accuracy measures are reported in the data table in the second port at the bottom of the "Scorer" node and give us information about the correctness and completeness of our model.

4.17. Accuracy Statistics Table from the „Scorer" node with the accuracy measures for each class

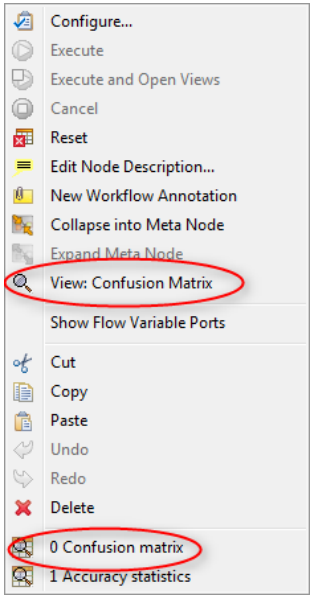
Data Hiliting

The context menu of the "Scorer" node offers 2 possibilities to visualize the confusion matrix:

Item "View: Confusion Matrix"

Item "0: Confusion Matrix"

4.18. The context menu of the "Scorer" node



These two items lead to a slightly different visualization of the same confusion matrix. The last item leads to the confusion matrix display that we have seen above. The first item "View: Confusion Matrix" includes a few more options. Let's have a look at the "View: Confusion Matrix" window (Fig. 4.19).

The total percentages of correctly classified data and of wrongly classified data are shown at the bottom of the table view.

The Top Menu includes an item for file operations and an item for "hiliting".

The "**File**" item in the Top Menu includes the following options:

"Always on top" makes sure that the table is always visible on the screen

"Export as PNG" exports the table as an image. This latter option can be useful if we want to include the confusion matrix in a report.

The "**Hilite**" item in the Top Menu was not present in the other view. Similarly to other plot views, "Hilite" allows the hiliting of cells in the confusion matrix. Data rows underlying those cells are subsequently hilited in all other views of the workflow, for example in the view of an "Interactive Table".

In this view of the confusion matrix, cells are selectable by clicking them (use Shift-click and Ctrl-click to select multiple cells). When a cell in the confusion matrix has been selected, we can hilite it by using the items of the "Hilite" sub-menu. To "hilite" a cell of the confusion matrix means to hilite the data rows to which this cell refers.

- "**Hilite Selected**" hilites the selected cells of the confusion matrix, i.e. it hilites the data that these cells represent
- "**Unhilite Selected**" unhilites all selected cells in the confusion matrix and all data underlying such cells
- "**Clear Hilite**" resets hiliting to a status where no cell is hilited

Note. The “Hilite” option was not present in the confusion matrix window shown in figure 4.17 and reached via the context menu item “0: Confusion Matrix”.

As an example, in the context menu of the “Scorer” node select “View: Confusion Matrix”. The Confusion Matrix view opens (Fig. 4.19).

Select the cell containing the number 6; this is the number of data rows belonging to class “self-emp-inc” and misclassified as “State-gov” .

In addition select (Ctrl-Click) the cell containing the number 4; this is the number of data rows belonging to class “self-emp-inc” and misclassified as “Local-gov”.

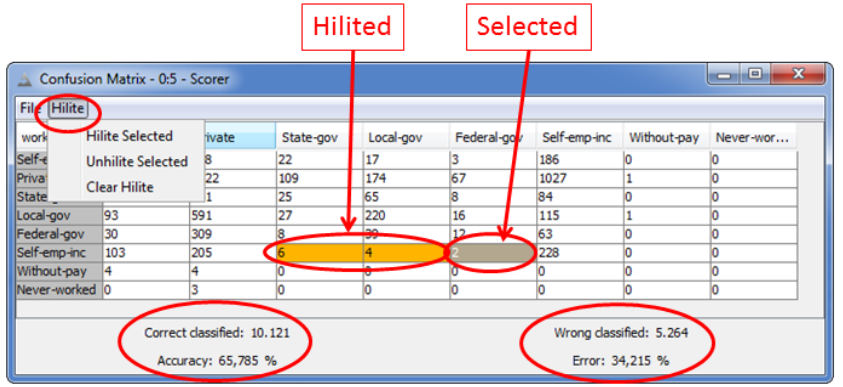
- Select “Hilite” in the Top Menu.
- Select “Hilite Selected”.

The two selected cells change to a pale yellow color. If we now select another cell, the pale yellow of the two hilited cells changes into a bright orange.

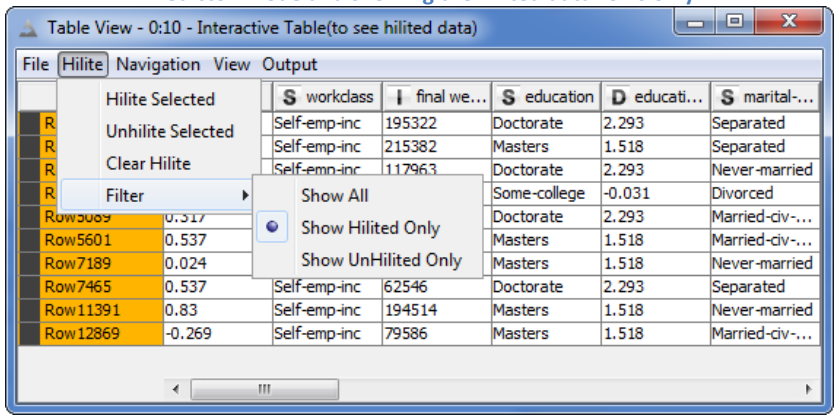
Let’s now insert an “Interactive Table” node from the “Data Views” category. The “Interactive Table” node is connected to the data output port of the “Naïve Bayes Predictor” node. Indeed, in order to be able to see the hilited data underlying the cells of the confusion matrix, the “Interactive Table” node must have access to the same data as the “Scorer” node. It therefore has to be connected to the “Naïve Bayes Predictor” output as the “Scorer” node is also connected.

- Right-click the “Interactive Table” node
- Select “View: Table View”
- In the Top Menu select “Hilite”
- Select “Filter” –> “Show Hilited Only”

4.19. Hilite functionality of window „View: Confusion Matrix”



4.20. Table View of an „Interactive Table” node connected to the “Naïve Bayes Predictor” node and showing the hilited data rows only



Only the data rows corresponding to the two cells of the confusion matrix hilited above appears in the view of the “Interactive Table” node. That is, only the data rows belonging to the class “Self-emp-inc” and misclassified in either the “State-gov” or “Local-gov” class will appear in the table view.

The “hilited” status is inherited by the “Scorer” node and represented by the bright orange color of their RowID cells.

Decision Tree

Using the same workflow “my_first_data_model”, let’s now apply another quite popular classifier: a decision tree [8] [9].

The Decision Tree algorithm consists of two phases – training and testing - like the Naïve Bayes classifier that we have seen in the previous section. The decision tree is implemented in KNIME with two nodes: one node for training and one node for testing, i.e.:

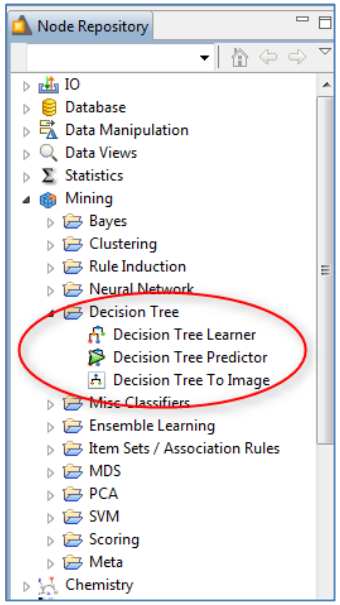
- The “Decision Tree Learner” node
- The “Decision Tree Predictor” node

The “Decision Tree Learner” node takes a data set as input (white triangle), learns the rules necessary to perform the desired task, and produces the final model at the output port (blue square). Let’s connect a “Decision Tree Learner” node to the “File Reader” node named “training set”.

Let’s also create a “Decision Tree Predictor” node to follow the “Decision Tree Learner” node. The “Decision Tree Predictor” node has two inputs:

- A data input (white triangle) with new data to be classified
- A model input (blue square) with the model parameters produced by a “Decision Tree Learner” node

4.21. Two nodes implement a Decision Tree: the “Decision Tree Learner” node and the “Decision Tree Predictor” node



The “Decision Tree Predictor” node has only one output table, consisting of the original data set with the appended prediction column and optionally the columns with the probability for each class to be the right class.

The sub-category “Mining” -> “Decision Tree” also includes a “Decision Tree To Image” node. This node converts the visualization of a decision tree model into an image that can then be exported into a report.

Our “Decision Tree Predictor” node reads the data from the “File Reader” “test set” node and the model from the “Decision Tree Learner” node.

Decision Tree Learner

The "Decision Tree Learner" node builds a decision tree from the input training data. The algorithm can run in multiple threads, and thus, exploit multiple processors or cores.

In the configuration window you need to specify:

The *class column.* The target attribute must be nominal.

The *quality measure* for split calculation: "Gini Index" or "Gain Ratio".

The *pruning method*: "No Pruning" or a pruning based on the "Minimum Description Length (MDL)" principle [8] [9].

The option *Reduced Error Pruning*, if checked, applies a simple post-processing pruning.

The *stopping criterion*: the minimum number of records in each decision tree's node. If one node has fewer records than this minimum number, the algorithm stops.

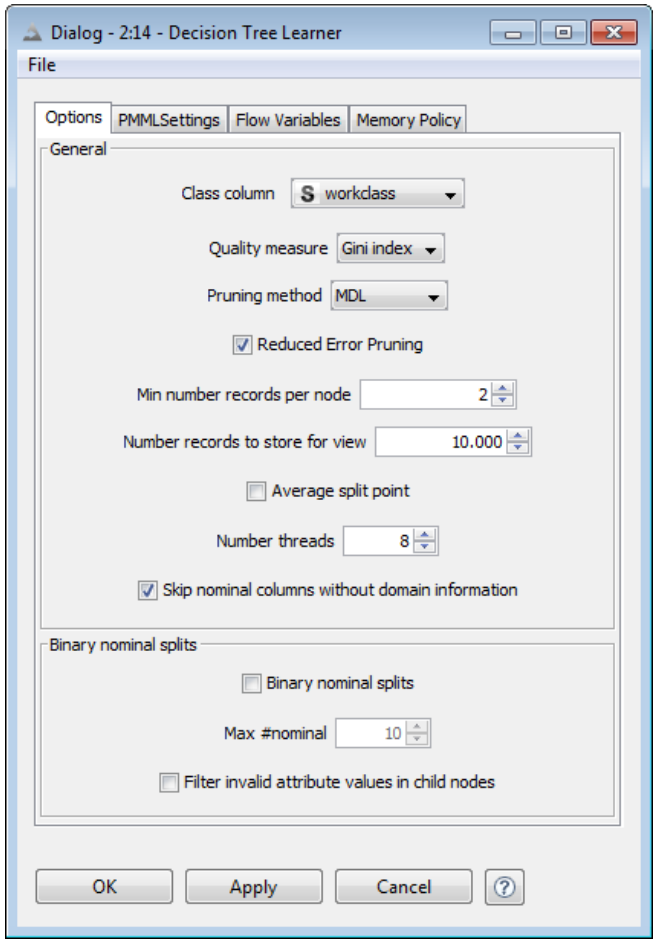
The *number of records to store for view*: the maximum number of rows to store for the hilite functionality.

The "*Average Split Point"* flag. For numerical attributes, the user has to choose one of two splitting strategies:

- The split point is calculated as the mean value between the two partitions' attributes ("*Average Split Point"* flag enabled)

- The split point is set to the largest value of the lower partition ("*Average Split Point"* flag disabled)

4.22. Configuration window for the „Decision Tree Learner" node



Whether *Binary nominal splits* apply to nominal attributes. In this case:

- The *Maximum # of nominal* splits, to reduce the computational load for this node.

- The *Number of threads* on which to run the node (default Number threads = number of processors available to KNIME).

We trained the "Decision Tree Learner" node with:

Class column = “workclass”
Gini Index as quality measure
Pruning = MDL
Stopping criterion = 2 data points per node

We accepted all the default values for the other configuration settings; i.e.:

Number of records for hiliting = 10000
No split point calculated as the average point between the two partitions
No binary splits for nominal values
Since the previous flag is not enabled, the maximum number of nominal splits is irrelevant here
2 as number of threads, since we are working on a 2-core machine

We can now run the “Execute” command and therefore train our decision tree model. At the end of the training phase the model is available at the output port (blue square) of the “Decision Tree Learner” node.

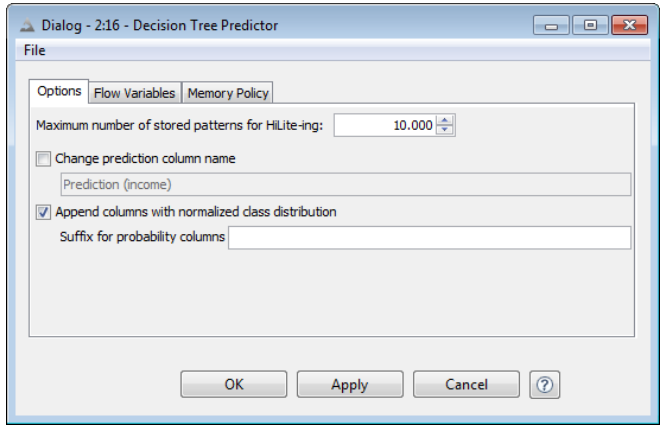
Decision Tree Predictor

The “Decision Tree Predictor” node assigns a Decision Tree model to the input port and applies it to the input data table.

In the configuration window you can:

- Define the maximum number of records for hiliting (sometimes it is not advisable to store all data records for hiliting for performance reasons)
- Define a custom name for the output column with the predicted class
- Append the columns with the normalized distribution of each class prediction to the output data set

4.23. Configuration window of the „Decision Tree Predictor“ node



In our “Decision Tree Predictor” node we enabled the option to append the normalized class distribution at the end of the prediction data table. We used the default maximum number of records for hiliting of 10000.

We then ran “Execute” which produced a data table with the class prediction and the class distributions on which the prediction was based (Fig. 4.24).

4.24. Output data table from the “Decision Tree Predictor” node. The last column is the prediction column. In between the original data columns and the prediction column you can see the columns with the normalized distribution values for each class

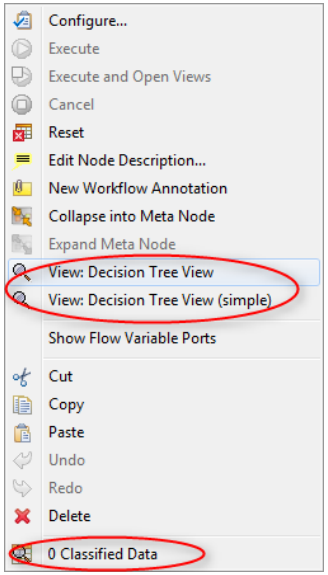
Decision Tree Views

In the context menu of both the “Decision Tree Predictor” node (Fig. 4.25) and the “Decision Tree Learner” node, we can see two options to visualize the decision tree rules:

- “View: Decision Tree View (simple)”
- “View: Decision Tree View ”

They correspond to two different views of the rules implemented by the decision tree. Both views are interactive and allow opening the decision tree branches to see their contents in terms of data rows and implemented rule.

4.25. Context menu of the “Decision Tree Predictor” node



The more complex view displays each branch as a rectangle. The data covered by this branch are shown inside the rectangle and the rule implementing the branch is displayed on top of the rectangle (Fig. 4.26).

The simpler view represents each branch as a circle fraction, where the fraction indicates how much of the underlying data is covered by the label assigned by rule. The rule is displayed at the side of the branch (Fig. 4.27).

In both views of the decision trees, a branch can be characterized by a little sign “+” indicating that more branches are connected to it.

The decision tree always starts with a “Root” branch that contains all training data. The “Root” branch in our decision tree is labeled “Private”, because it covers 11305 “Private” records out of 15340 from the training set. A majority vote is used to label each branch of the decision tree. In the simpler view, the label’s cover factor is visualized by the circle drawn on the side of the branch. The “Root” branch has a ¾ circle, meaning that its label covers three quarters of the incoming training records.

The first split happens in the “Occupation” column. From the “Root” branch the data rows are separated into a number of sub-branches according to their value of the “Occupation” attribute. Each branch is labeled with a predicted class coming from its cover factor. For example, the branch defined by the split condition “Occupation = Sales” is labeled as “Private” since during the training phase it covered 1477 “Private” records out of its incoming 1846.49 training patterns. Fraction values as a total number of training patterns can occur when missing values are encountered during training. In this event only fractions of the patterns are passed down the following branches.

Inside each branch more splits are performed and data rows are separated into different branches and so on deeper and deeper into the tree until the final leaves. The final leaves produce the final prediction/class.

In the decision tree views a branch of the decision tree can be selected by clicking it. Selected branches are shown with a black rectangle border (simple view) or with a darker background (complex view). A selection of multiple branches is not possible.

The “Decision Tree View” window has a top menu with three items.

“**File**” has the usual options:

- “Always on top” ensures that this window is always visible
- “Export as PNG” exports this window as a picture to be used in a report for example
- “Close” closes the window

“**Hilite**” contains all the hilite functionalities:

- “Hilite Selected Branch” hilites the selected branch and the data being classified by the selected branch
- “Unhilite Selected Branch” unhilites the selected branch and the data being classified by the selected branch
- “Clear Hilite” resets of the hilite status

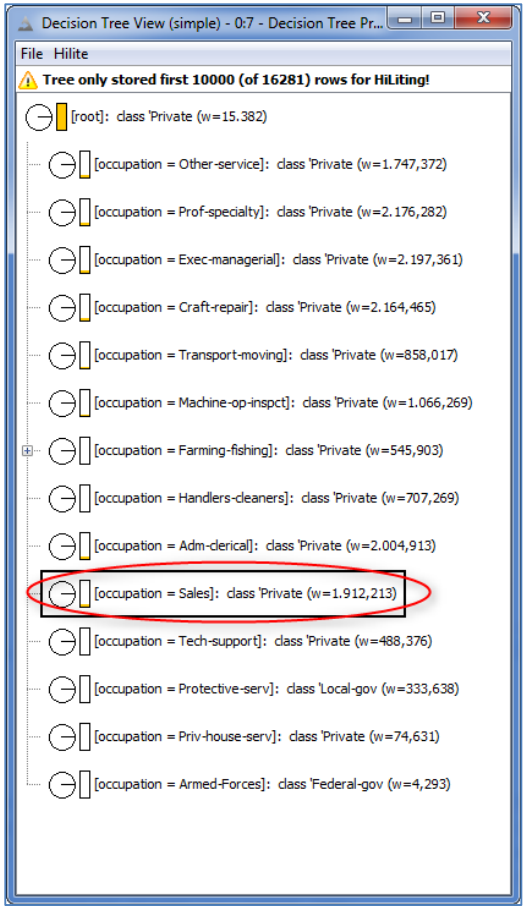
“**Tree**” offers the commands to expand and collapse the tree branches:

- “Expand Selected Branch” opens the sub-branches, if any, of a selected branch of the tree
- “Collapse Selected Branch” closes the sub-branches, if any, of a selected branch of the tree

On the right side, there is an overview of the decision tree. This is particularly useful if the decision tree is big and very bushy. In the same panel on the bottom, there is a zoom functionality to explore the tree with the most suitable resolution.

4.26. „Decision Tree View“ window

4.27. „Decision Tree View (simple)“ window



In the “Decision Tree View (simple)” (Fig. 4.27) of our “Decision Tree Predictor” node we hilited the split condition “Occupation = Sales”.

We then connected an “Interactive Table” node to the “Decision Tree Predictor” node and opened its “Interactive Table View”. Here we selected “Hilite” -> “Filter” -> “Show Hilited Only”. In the Table View we could then see all the input data rows that have been classified by this leaf (“Occupation = Sales”) in the decision tree. The maximum number of input records that can be hilited is defined in the configuration window of the “Decision Tree Predictor” node.

A similar hilite functionality can also be found in the “Decision Tree Learner” node and refers to its input data.

In order to measure how well our decision tree has performed, we can use an “Entropy Scorer” node. The “Entropy Scorer” node works in the same way as the “Scorer” node. It just offers a different set of quality measures.

Entropy Scorer

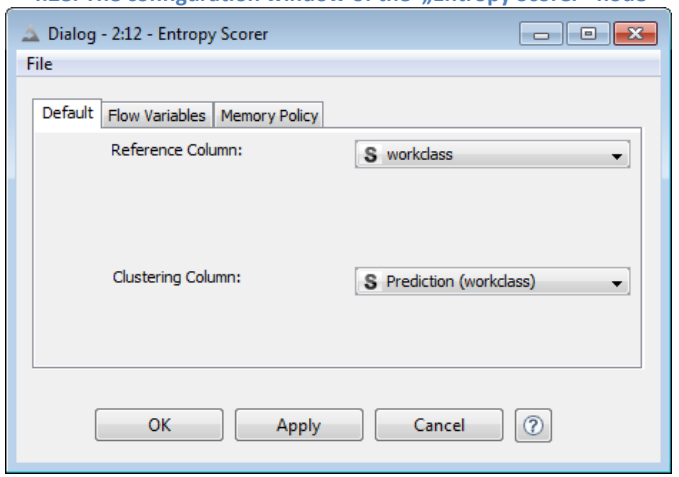
The “Entropy Scorer” node compares the values of two columns (reference column and clustering column) from two different data tables (reference table and clustering table) and calculates a number of quality measures.

The “Entropy Scorer” node has no hilite functionality.

The “Entropy Scorer” node is located in the “Node Repository” panel in the `“Mining” -> “Scoring”` category.

In the configuration window you are required to specify the two columns to be compared.

4.28. The configuration window of the „Entropy Scorer“ node



Since we host the clustering column (that is the prediction column of the “Decision Tree Predictor” node) and the original classification column in the same table, the “Entropy Scorer” node is connected to the same table on both input ports. We now select the column called “workclass” as the reference column and the column called “Prediction(workclass)” as the clustering column.

The “Entropy Scorer” node has two options in the context menu (Fig. 4.29) to see the quality measures:

- “View: Statistics View” shows the quality measures plus a statistics summary (Fig. 4.30)
- “Quality Table” only shows the quality measures

Quality Measures

The quality measures implemented by the "Entropy Scorer" node follow the definitions adopted in [10]. A cluster here refers to a predicted class of the decision tree.

Size. This is the cluster size; i.e. how many input records belong to the final predicted class.

Entropy. This is the accumulated entropy of all identified clusters, weighted by the relative cluster size. It is not normalized and may be greater than 1. The lower the entropy, the better the classification performance.

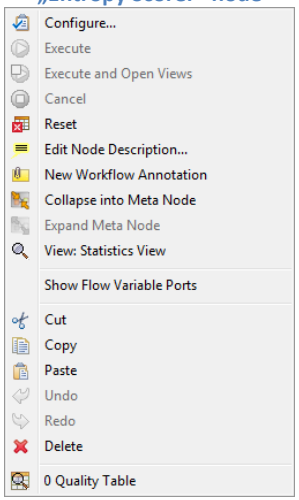
Normalized Entropy. This is the entropy value divided by `log2(number of original classes)`. The resulting value is then scaled to fall into the interval [0,1].

Quality. This is a measure defined over all records and all clusters. The quality of a single cluster is calculated as:

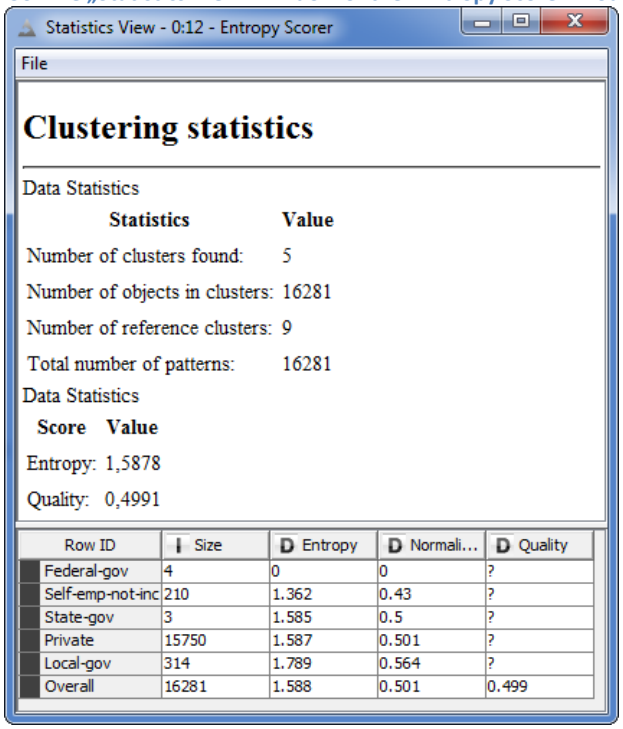
```
Quality(cluster) = (1-normalized_entropy(cluster))
```

The overall quality is the sum of the qualities of the individual clusters weighted by the cluster size. The domain of the quality value is [0,1], with 1 being the best possible value.

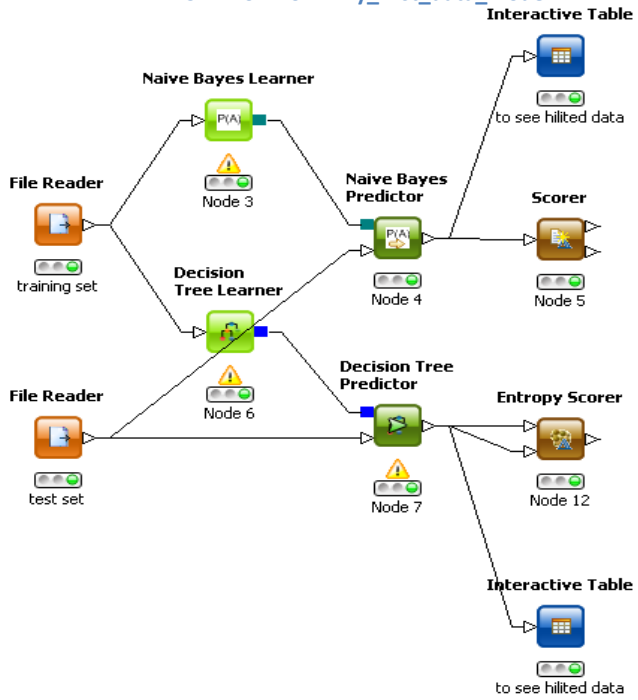
4.29. Context menu of the „Entropy Scorer" node



4.30. The „Statistics View" window of the "Entropy Scorer" node



4.31. Workflow "my_first_data_model"



Artificial Neural Network

We move on now from a multi-class classification task to a two-class classification task. We would like to show how to train and apply a very basic neural network, with Multilayer Perceptron architecture, one hidden layer, and the Back Propagation learning algorithm.

The neural network paradigm available in the "Mining" category is divided into:

- A learner node ("RProp MLP Learner")
- A predictor node ("Multilayer Perceptron Predictor")

The learner node learns the rules to separate the input patterns of the training set, packages them into a model, and assigns them to the output port.

The predictor node applies the rules from the model built by the learner node to a data set with new records.

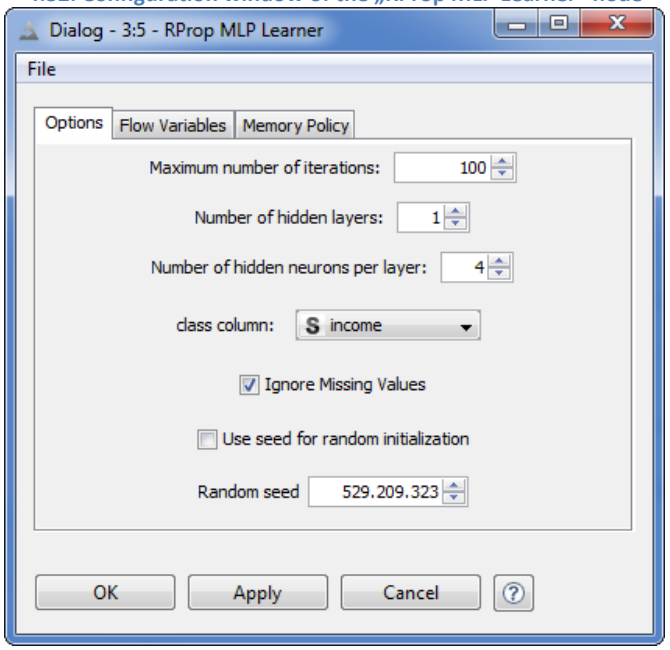
RProp MLP Learner

The "RProp MLP Learner" node builds and trains a Multilayer Perceptron with the BackPropagation algorithm. In the configuration window you need to specify:

The *maximum number of iterations* for the Back Propagation algorithm
The *number of hidden layers*
The *number of neurons per each hidden layer*
The *class column,* i.e. the column containing the classes
You also need to specify what to do with missing values. The algorithm does not work if missing values are present. If you have missing values you need to either transform them earlier on in your workflow or ignore them during training. To ignore the missing values just mark the checkbox at the bottom of the configuration window.
Finally, you need to specify a seed to make repeatable the weight random initialization.

The "RProp MLP Learner" only accepts numerical inputs.

4.32. Configuration window of the „RProp MLP Learner" node



We created a new workflow in the workflow group “Chapter4” and named it “my_first_ANN”. We also used the data sets “training set” and “test set” derived from the adult.data data set in the “data_preparation” workflow. We changed the classification/prediction task from predicting the “workclass” column to predicting the kind of income. “Income” is a string column with only two values: “>50K” and “<=50K”.

First, we inserted two “File Reader” nodes: one to read the training set and one to read the test set prepared by the “data_preparation” workflow earlier on in this chapter.

Of all the string attributes in the adult data set, we decided to keep only the attribute “sex”, since we think that sex is an important discriminative variable in predicting the income of a person. Of course we also kept the “Income” column to be the reference class. We removed all other string attributes. The attribute “sex” has been converted into a binary variable “sex_01”, according to the following rule:

IF sex = “Male” => sex_01 = “-1”
IF sex = “Female” => sex_01 = “+1”

In order to implement this rule, we used a “Rule Engine” node. “sex_01” is the newly created string column containing the binary values for sex. We then used a “StringToNumber” node to convert “sex_01” from String to Double and a “Column Filter” node to exclude all remaining string columns besides “Income”.

A “Normalizer” node was placed after the “Column Filter” node to normalize “sex_01” values to follow a Gaussian normalized distribution, i.e. we applied a “Z-score normalization” to the “sex_01” column. This sequence of transformations (“Rule Engine” on “sex”, “Column Filter” to keep only numerical attributes and “Income”, and “Z-score normalization” on “sex_01”) was applied on both training and test set.

We then applied an “RProp MLP Learner” node to build an MLP neural network with 6 input variables, 1 hidden layer with 4 neurons, and 2 output neurons, i.e. one output neuron for each “Income” class. We trained it on the training set data with a maximum number of iterations of 100. After training, we applied the MLP model to the test set’s data by using a “Multilayer Perceptron Predictor” node.

The neural network’s predictor node applies the rules from the model built by the learner node to a data set with new records. The predictor node has two input ports:

- A data input (white triangle) with the new data to be classified
- A model input (green square) with the model parameters produced by a “RProp MLP Learner” node

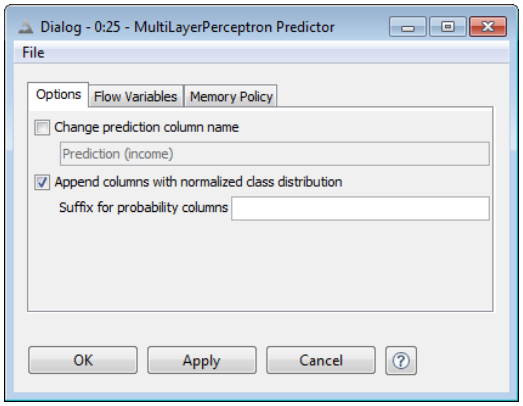
Multilayer Perceptron Predictor

The "Multilayer Perceptron Predictor" node takes an MLP model, generated by an "RProp MLP Learner" node, at the model input port (green square) and applies it to the input data table at the input data port (white triangle).

The "Multilayer Perceptron Predictor" node can be found in the "Node Repository" in the `"Mining" -> "Neural Network" -> "MLP"` category.

The only settings required for its configuration are a checkbox to append the normalized class distributions to the input data table and a possible customized name for the output class column

4.33.Configuration window of the „MultiLayer Perceptron Predictor" node



Classified Data

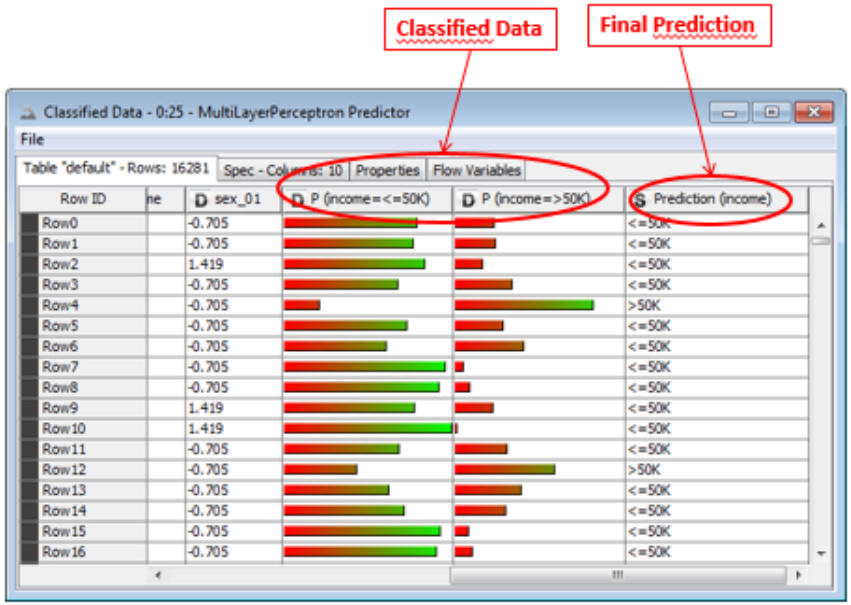
We can visualize the results of the MLP Prediction:

- Right-click the "Multilayer Perceptron Predictor" node
- Select "Classified data"

The "Classified Data" data table (Fig. 4.34) contains the final prediction class in the "Prediction (income)" column and the values of the two output neurons in the columns "P (Income >50K)" and "P(Income<=50K)".

The firing value of the two output neurons is represented by a red-green bar instead of a double number. Red means a low number (< 0.5), green a high number (> 0.5). The highest firing output neuron decides the prediction class of the data row.

4.34. Output Data Table of the „Multilayer Perceptron Predictor" node



Note. No data hiliting is available for learner and predictor nodes of the "Neural Network" -> "MLP" Category.

ROC Curve

The "ROC Curve" node draws the ROC Curve for a two-class classification problem. To draw an ROC curve the target classification column has to contain two class labels only. One of them is identified as the positive class. A threshold is then incrementally applied to the column containing the probabilities for the positive class, therefore defining the true positives rate and the false positives rate for each threshold value [5]. At each step, the classification is performed as:

```
IF    Probability of positive class > threshold   =>  positive class
ELSE                                              =>  negative class
```

The ROC Curve, in particular the area below the ROC curve, gives an indication of the predictor performance. The "ROC Curve" node does not have a hilite functionality.

The "ROC Curve" node is located in the "Node Repository" panel in the "Mining" -> "Scoring" category. The configuration window requires:

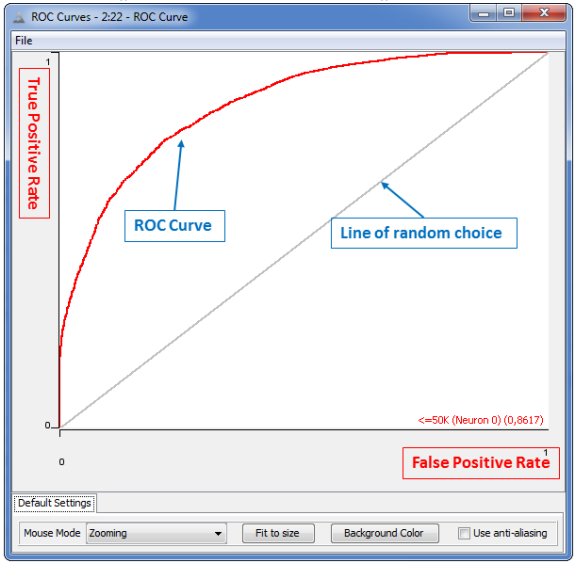
- The column containing the reference class ; only two class labels are allowed for a two-class classification problem
- The class label representing the positive class
- The column(s) with the probabilities for the positive class

It is possible to display multiple curves for different columns in the ROC Curve View, if we want to compare the performances of more than one classifier.

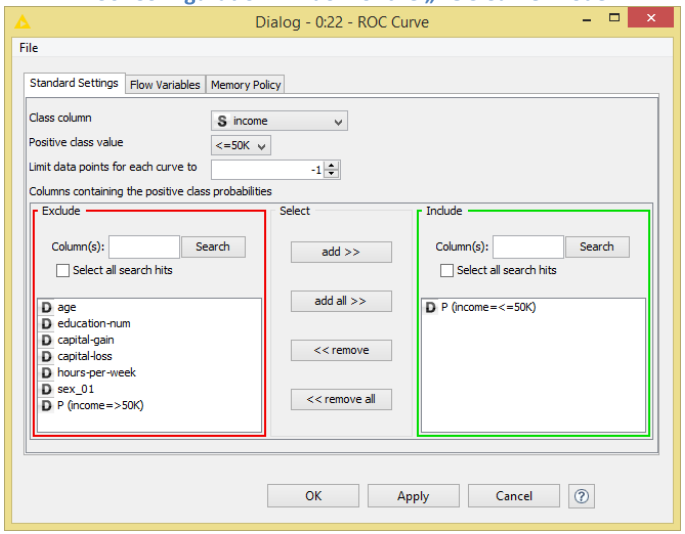
If the data set is very large, the option "Limit data points for each curve to" allows to work on a smaller subset, improving the execution time.

The selection of the columns with the probabilities for the positive class is performed by means of an "Exclude"/"Include" frame.

4.35. The „ROC Curve View" for the „ROC Curve node



4.36. Configuration window of the „ROC Curve" node



We have used the Neural Network paradigm in a two-class classification problem (“Income > 50K” or “Income <=50K”). We can now apply an “ROC Curve" node to the results of the “Multilayer Perceptron Predictor” node. We identified:

The class column as column “Income”
The label for the positive class as value “<=50K” in column “Income”
The column “P(Income <=50K)” as the column containing the probability/score for the positive class

The resulting ROC Curve is shown in Figure 4.35. The closer the curve is to the point [0,1], the better the classifier performance is. This can be seen in the ROC Curve itself, but also in the area under the ROC curve. The optimal performance has an area of 1.0 under the ROC Curve. The last item of the context menu of the “ROC Curve” node leads to the “Area under curve”. The “ROC Curve” node following the “Multilayer Perceptron Predictor” node shows an underlying area of 0.819.

Write/Read Models to/from file

Once we have trained a model and ascertained that it works well enough for our expectations, it would be nice if we could reuse the same model in other similar applications on new data. This means that we should be able to recycle the model in other workflows as well. KNIME offers two nodes to write a model to a file and two nodes to read a model from a file.

Model Writer

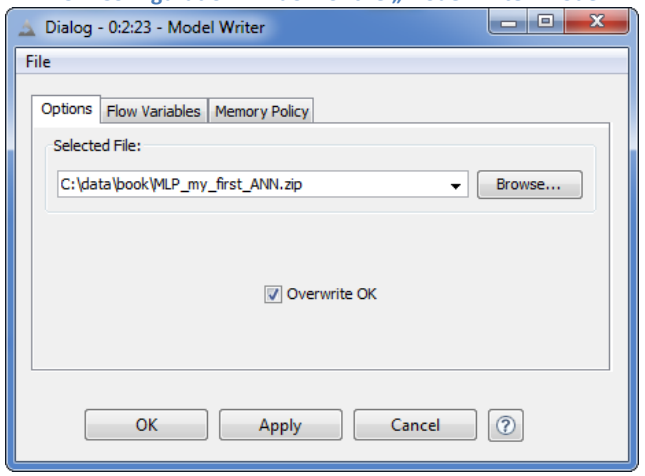
The “Model Writer” node takes a model at the input port (gray square) and writes it into a file by using the KNIME internal format.

The “Model Writer” node is located in the `"IO" -> "Write"` category in the “Node Repository” panel.

The configuration window only requires:

- The path of the output file (*.zip file)
- The flag to override the file if the file exists

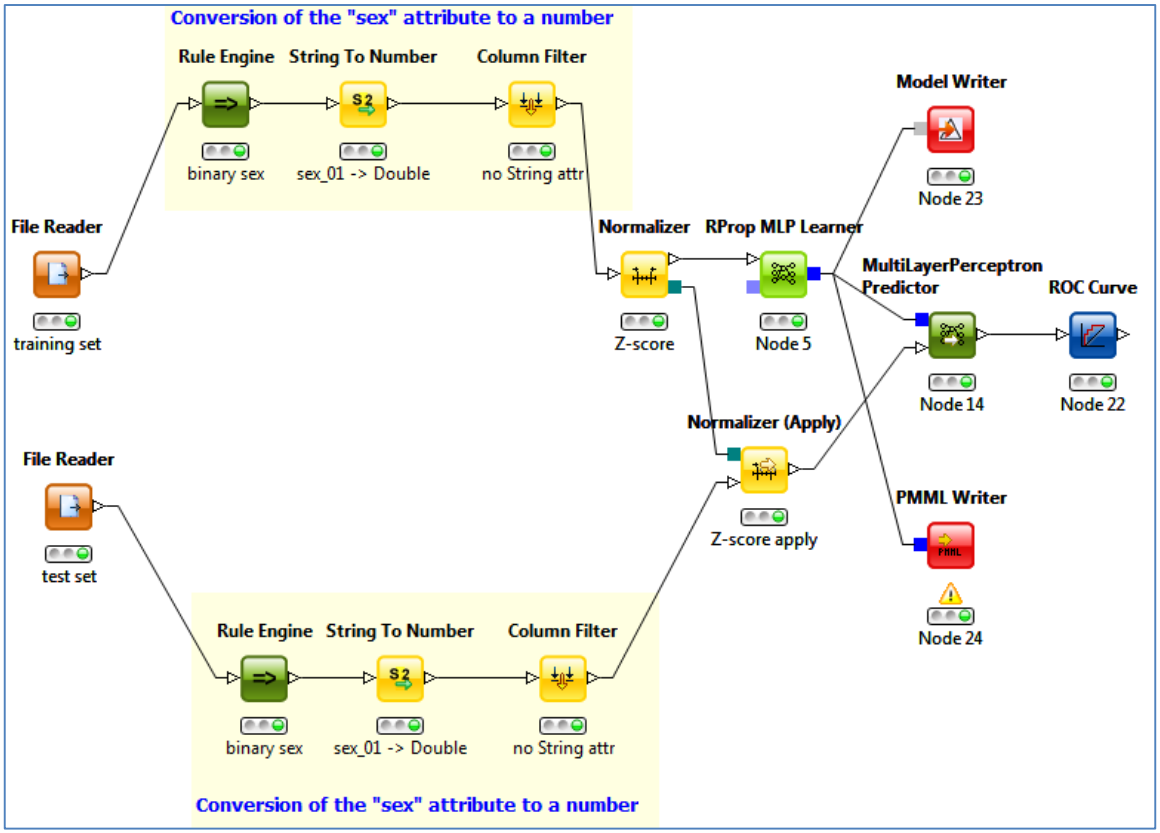
4.37. Configuration window of the „Mode Writer“ node



Note. In the same "IO" -> "Write" category, you will find a similar node: the "PMML Writer" node. The "PMML Writer" node writes a model into a file with extension .pmml using the PMML standard format.

The final workflow "my_first_ANN" is shown in figure 4.38.

4.38. Workflow „my_first_ANN"



At the same time, KNIME also provides two nodes to read a model from a file: the "Model Reader" node and the "PMML Reader" node. Both nodes are located in the "IO" -> "Read" category in the "Node Repository" panel.

Model Reader

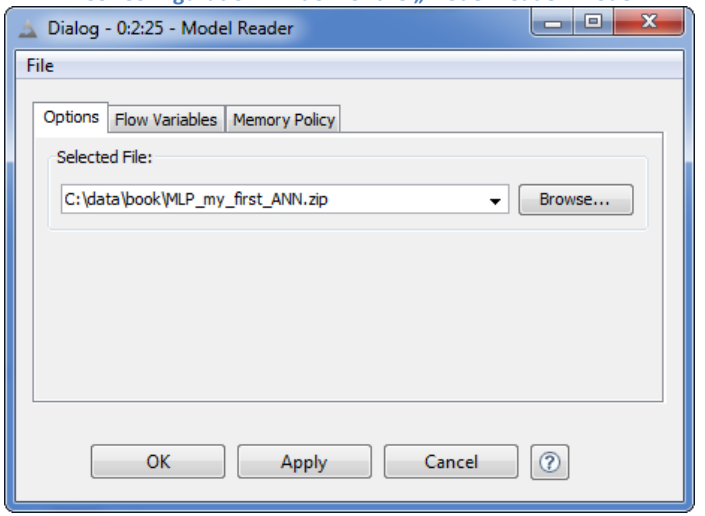
The “Model Reader” node reads a model from a file using the KNIME internal format and makes it available at the output port (gray square).

The “Model Writer” node is located in the `"IO" -> "Read"` category in the “Node Repository” panel.

The configuration window only needs:

- The path of the input file (*.zip file)

4.39. Configuration window of the „Model Reader“ node



Note. Similar to the “Model Reader” node, the “PMML Reader” node reads a model from a file with extension .pmml, but requires the PMML standard format.

In this last part of the chapter, we would like to show a few more nodes that are commonly used in data analysis. We will build a new workflow, named “Clustering and Regression”, in the workflow group “Chapter4” to host these nodes.

We will use the same data that we used for the previous two workflows, “training set” and “test set”, created in the “data_preparation” workflow. The first two nodes in the workflow will then be two “File Reader” nodes, one to read the training set and one to read the test set data.

Statistics

The “Statistics” node calculates statistic variables on the input data, such as:

For numerical columns (available in a table on output port 0):		For nominal columns (available in a table on output port 1 and 2):
minimum	variance	number of occurrences of nominal values
maximum	median	Number of missing values
Mean	overall sum	Histogram of nominal values
Standard deviation	kurtosis	
skewness	number of NaN/missing values	
Histogram		

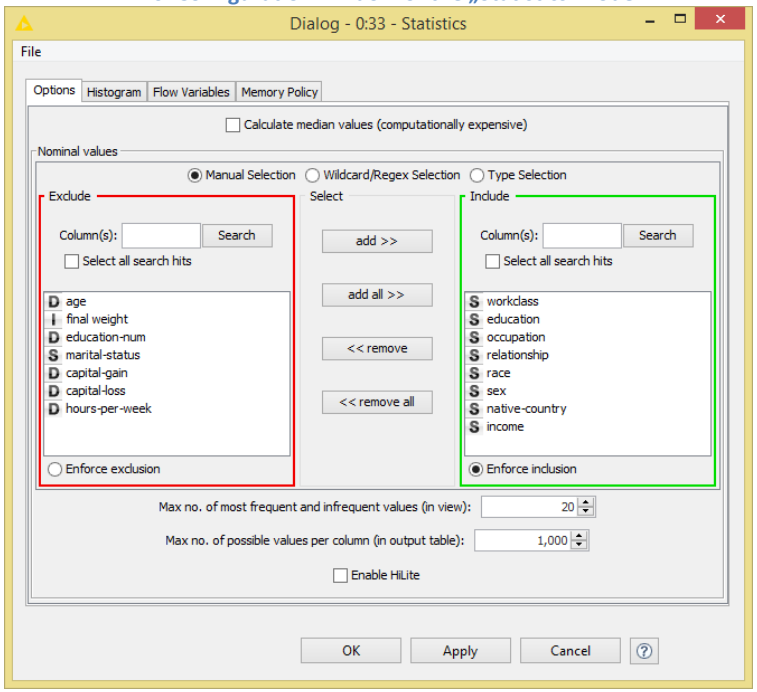
The “Statistics” node is located in the “Node Repository” panel in the “Statistics” category.

The configuration window requires:

- The selection of the nominal columns on which to calculate the statistical measures (the statistical measures for numerical variables are calculated on all numerical columns by default). The column selection is performed by means of an “Exclude”/”Include” frame.
- The maximum number of most frequent and infrequent values to display in the view
- The maximum number of possible values per column. This is to avoid long lists of nominal values.
- Whether to calculate the median value
- Whether to preserve the hilite for the following nodes

All the statistical measures, described in the table above, are available at the node output ports as well as in the node View. Note that the “Statistics View” option in the context menu has no hilite functionality.

4.40. Configuration window of the „Statistics“ node



The "Statistics" node has two visualization options: the "Statistics View" and the data tables on the output ports. Both visualization options are reachable via the context menu.

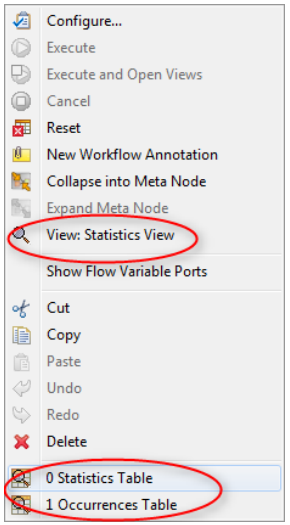
The node has three output ports and the "Statistics View" has three specular tabs. Output port "Statistics Table" corresponds to tab "Numeric" in the view; output port "nominal Statistical Values" corresponds to tab "Nominal" in the view; and output port "Occurrences Table" corresponds to tab "Top/Bottom" in the view.

Tab "Numeric" contains a number of statistical measures calculated on all numerical columns, with an approximate histogram. Each row then with all the statistical measures and the rough histogram offers an idea of the statistical properties and distribution of the values in a numeric data column.

Similarly, the "Nominal" and the "Top/Bottom" tabs give an idea of the statistical properties of the values in a nominal data column.

Note. The statistics of nominal columns is calculated only for the nominal columns included in the configuration window.

4.41. Context menu of the "Statistics" node

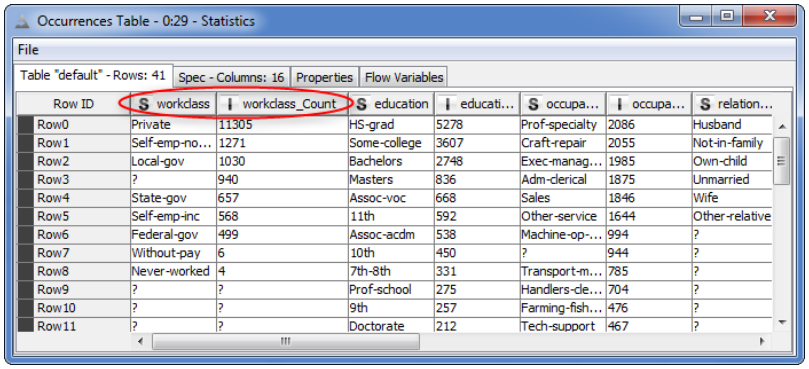


In our workflow's node we excluded the column "marital-status" from the nominal columns and the corresponding column pair "marital-status" and "marital status_Count" is not in the "Occurrences Table".

In "Statistics View" -> tab "Nominal Columns" we find the same information, but the lists of nominal values are sorted by frequency. For each column we find two table cells: one at the top for the most frequent (top 20) nominal values in the column and one at the bottom for the least frequent (bottom 20) nominal values in the column.

4.42. The "Numeric" tab of the "Statistics" node view displays statistical measures and histogram calculated on all numerical columns.

4.43. The “Occurrences Table” contains the number of occurrences of nominal values calculated only on the selected nominal columns



Row ID	workclass	workclass_Count	education	educati...	occupa...	occupa...	relation...
Row0	Private	11305	HS-grad	5278	Prof-specialty	2086	Husband
Row1	Self-emp-no...	1271	Some-college	3607	Craft-repair	2055	Not-in-family
Row2	Local-gov	1030	Bachelors	2748	Exec-manag...	1985	Own-child
Row3	?	940	Masters	836	Adm-clerical	1875	Unmarried
Row4	State-gov	657	Assoc-voc	668	Sales	1846	Wife
Row5	Self-emp-inc	568	11th	592	Other-service	1644	Other-relative
Row6	Federal-gov	499	Assoc-acdm	538	Machine-op-...	994	?
Row7	Without-pay	6	10th	450	?	944	?
Row8	Never-worked	4	7th-8th	331	Transport-m...	785	?
Row9	?	?	Prof-school	275	Handlers-cle...	704	?
Row10	?	?	9th	257	Farming-fish...	476	?
Row11	?	?	Doctorate	212	Tech-support	467	?

4.44. „Statistics View“ -> Tab „Top/Bottom” with the number of occurrences of nominal values calculated only on the selected nominal columns and sorted in descending order

Regression

Another very common task in data analysis is the calculation of the linear regression [3] [4] [5]. In the “Node Repository” panel, in the “Statistics” -> “Regression” category, there are two learner nodes to learn the regression parameters: one node performs a multivariate linear regression, the other node a multivariate polynomial regression. Both regression learner nodes share the predictor node.

Regression Learner nodes have two input ports and two output ports. At input, the node is fed with the training data and optionally with a pre-existing model. After execution, the node produces the regression model and the statistical properties of the model in a data table.

The predictor node takes the regression model, linear or polynomial, as input and applies it to new input data rows to predict their response. In this book, we will only show how to implement the linear regression.

Linear Regression (Learner)

The "Linear Regression (Learner)" node performs a multivariate linear regression on a target column, i.e. the response.

The "Linear Regression (Learner)" node can be found in the "Node Repository" in category "Statistics" -> "Regression"

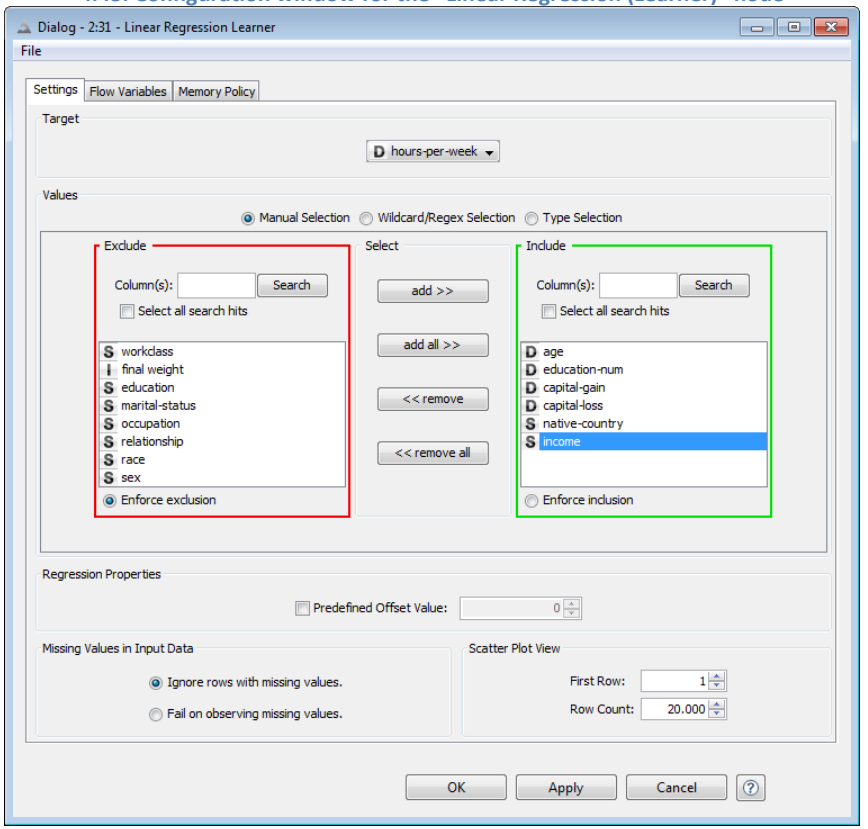
In the configuration window you need to specify:

- The *target column* for which the regression is calculated
- The *columns* to be used *as independent variables* in the linear regression
- The number of the starting row and the number of rows to be visualized in the node's scatter plot view
- The missing value handling strategy
- A default offset value to use

Selection of the input data columns is performed by means of the column selection framework: by manual selection with "Include/Exclude" panels; by type selection, by Wildcard/Regex expression selection.

The node outputs the regression model as well as the model's coefficients and statistics.

4.45. Configuration window for the "Linear Regression (Learner)" node



We connected a "Linear Regression (Learner)" node to the "File Reader" with the training data. We wanted to predict the columns "hours-per-week" by using the columns "age", "education-num", "capital-gain", "capital-loss", "native-cuntry", and "income" as independent variables for the linear regression.

Note. The "Linear Regression (Learner)" node can only deal with numerical values. Nominal columns are automatically discretized using a dummy coding available for categorical variables in regression http://en.wikipedia.org/wiki/Categorical_variable#Categorical_variables_in_regression.

The “Linear Regression (Learner)” node produces the regression model at the node’s output port. The regression model is subsequently fed into a “Regression Predictor” node and used to predict new values for a different data set.

Regression (Predictor)

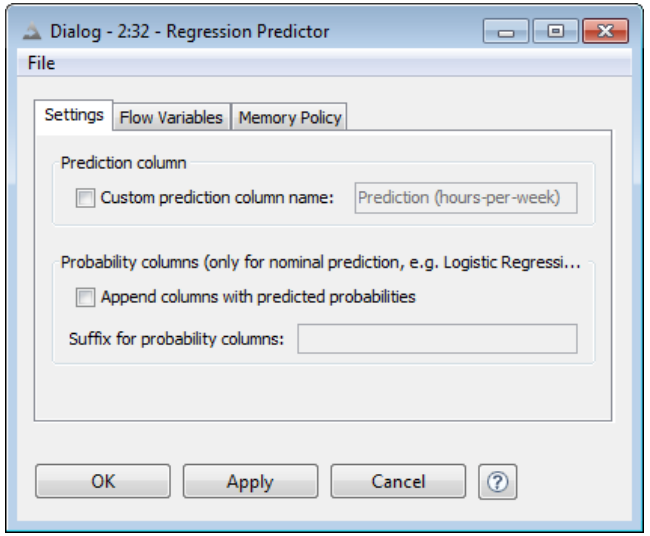
The “Regression (Predictor)” node obtains a regression model from one of its input ports (blue square) and data from the other input port (white triangle). It uses the model and the data to make a data based prediction.

The “Regression (Predictor)” node needs input data in the same format as the data that was used to build the regression model, i.e. with the same columns as the training data.

Since all information is already available in the model, this node only needs the minimal predictor settings: possible customized name for the output classification column and a flag to append the output class distribution columns to the input data table.

The “Regression (Predictor)” node is located in the `“Statistics” -> “Regression”` category in the “Node Repository” panel.

4.46. Configuration window for the “Regression (Predictor)“ node



Clustering

The last topic that we want to discuss in this chapter is clustering. There are many clustering techniques around and KNIME has implemented a number of them.

As in data models we already looked at, we have a trainer node and a predictor node for the clustering models. The learner nodes implement a clustering algorithm; that is they build a number of clusters by grouping together similar patterns and calculate their representative prototypes. The predictor then assigns a new data vector to the cluster with the nearest prototype. Such a predictor is not specific to only one clustering technique, but it works for any clustering algorithm that requires a cluster assignment on the basis of a distance function in the prediction phase. This leads to many specific clustering learner nodes (implementing different clustering procedures) but to only one clustering predictor node.

A learner node could implement the k-Means algorithm, for example. The k-Means procedure builds k clusters on the training data, where k is a predefined number [3] [4] [5]. The algorithm iterates multiple times over the data and terminates when the cluster assignments no longer change. Note that the k clusters are only built on the basis of a similarity (distance) criterion. k-Means does not take into account the real class of each data

row: it is an unsupervised classification algorithm. The predictor performs a crisp classification that assigns a data vector to only one of the k clusters which were built on the training data; in particular it assigns the data vector to the cluster with the nearest prototype.

We will focus on the k-Means algorithm to give you an example of how clustering can be implemented with KNIME (see the "Clustering and Regression" workflow).

k-Means

The "k-Means" node groups input patterns into k clusters on the basis of a distance criterion and calculates their prototypes. The prototypes are built as the mean value of the cluster patterns. This node takes the training data on the input port and presents the model at the blue squared output port and the training data with cluster assignment on the data output port (white triangle).

The "k-Means" node can be found in the "Node Repository" in the `"Mining" -> "Clustering"` category.

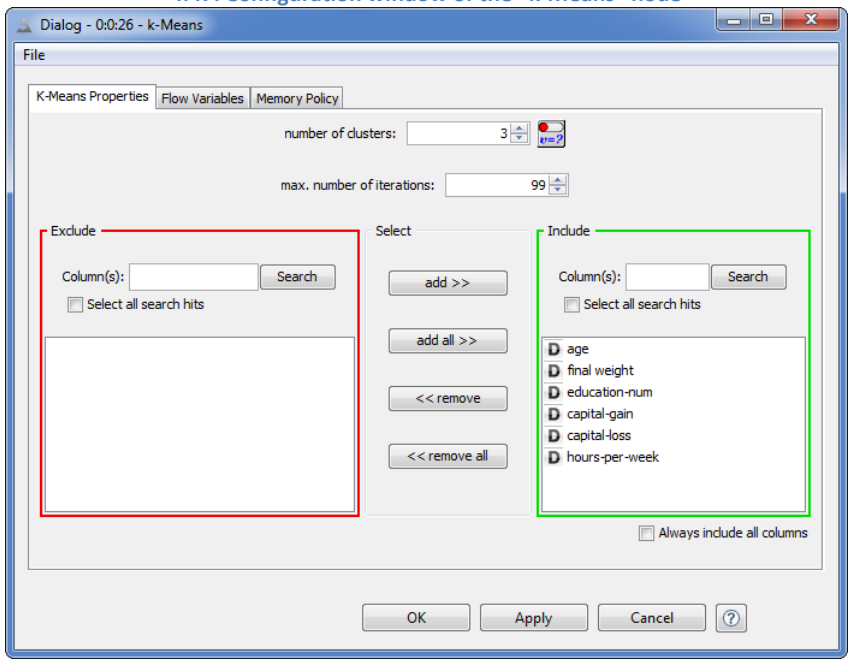
In the configuration window you need to specify:

- The final *number of clusters k*
- The *maximum number of iterations* to ensure that the learning operation converges within a reasonable time
- The *columns* to be used to calculate the distance and the prototypes
- Flag "Always include all columns" is alternative to the column selection frame.

Column selection is performed by means of an "Exclude"/"Include" frame.

- The columns to be used for the distance calculation are listed in frame "Include". All other columns are listed in frame "Exclude".
- To move from frame "Include" to frame "Exclude" and vice versa, use buttons "add" and "remove". To move all columns to one frame or the other use buttons "add all" and "remove all".

4.47. Configuration window of the "k-Means" node



The “K-Means” node has a “Cluster View” option in the context-menu: “View: Cluster View”. The Cluster View shows the prototypes of the k clusters.

Note. Since clustering algorithms are based on distance, a normalization is usually required to make all feature ranges comparable. In the “Clustering and Regression” workflow, we normalized the input features all into [0,1] by using a “Normalizer” node.

The “K-Means” algorithm though just defines the clusters in the input space on the basis of a representative subset of the same input space. Once the set of clusters is defined, new data rows need to be scored against it to find the cluster they belong to. To do that, we use the “Cluster Assigner” node.

Cluster Assigner

The “Cluster Assigner” node assigns test data to an existing set of prototypes that have been calculated by a clustering node such as the “k-Means” node. Each data row is assigned to its nearest prototype.

The node takes a clustering model and a data set as inputs and produces a copy of the data set with an additional column containing the cluster assignments.

The “Cluster Assigner” node is located in the `“Mining” -> “Clustering”` category in the “Node Repository” panel.

It does not need any configuration settings specific to its cluster assignment task.

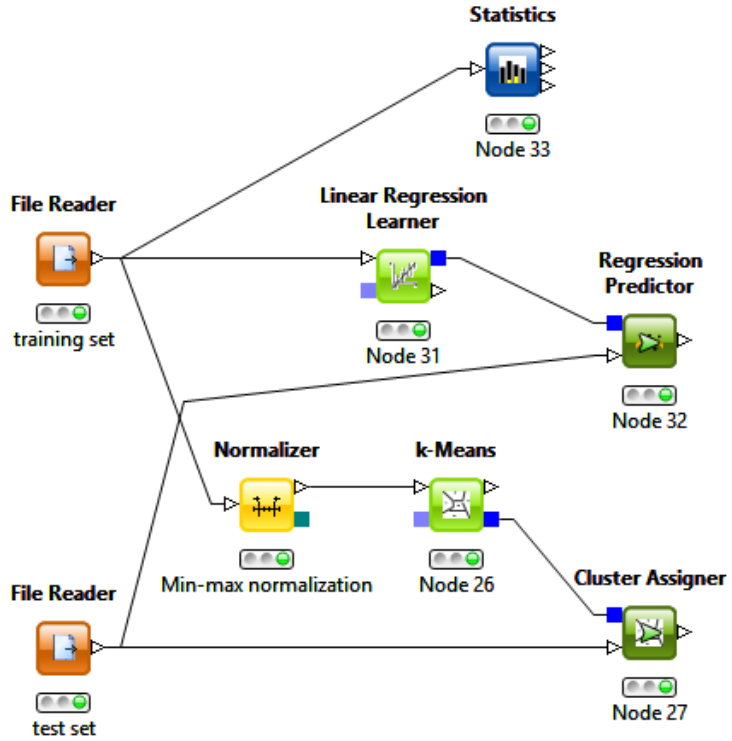
Note. The “Cluster Assigner” node is not specific for the “k-Means” node. It performs the cluster assignment task from a cluster set based with any of the available clustering algorithms.

The final workflow “Clustering and Regression” is shown in figure 4.48.

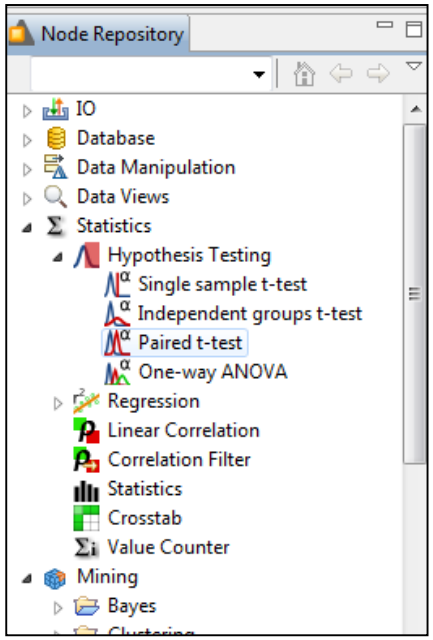
Hypothesis Testing

A few nodes are available in KNIME to perform classical statistical hypothesis testing. Most of them can be found in “Statistics” -> “Hypothesis Testing”: the single sample t-test, the paired t-test, the one way ANOVA, and the independent group t-test. Only the node performing the chi-square test is located outside of the “Hypothesis Testing” sub-category into the “Crosstab” node.

4.48. Workflow „Clustering and Regression“



4.49. The "Hypothesis Testing" sub-category



4.5. Exercises

Exercise 1

Using the wine.data file (training set = 80% and test set = 20%) train a decision tree to recognize the class to which each wine belongs.

Run the decision tree on the wine test set and measure the decision tree performance. In particular, we are interested in finding out how many false negatives for class 2 there are.

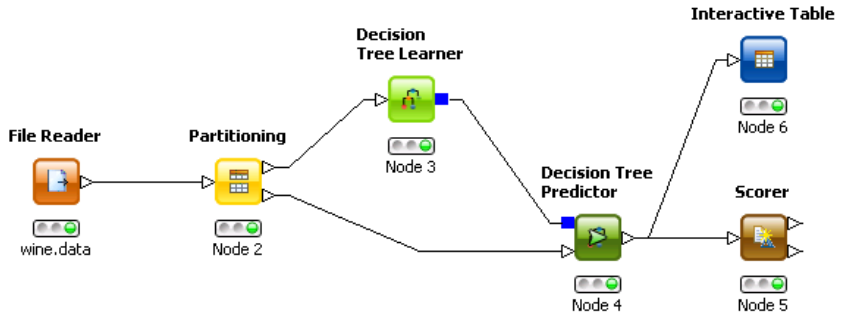
We would also like to identify the row number in the original test set of the wine belonging to class 2 that has been erroneously classified into class 3.

Solution to Exercise 1

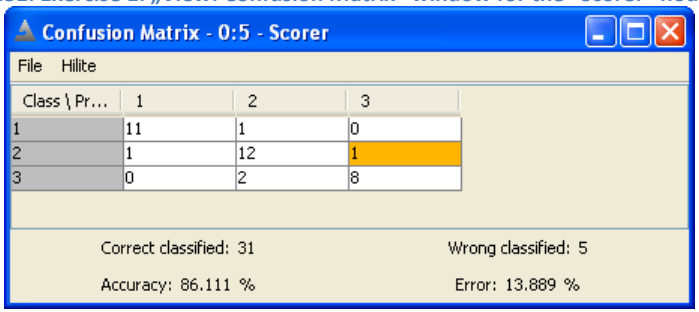
In the "Decision Tree Learner" node we used column "class" as the class column.

We then used a "Scorer" node to see how many False Negatives were produced in the accuracy statistics and/or in the confusion matrix.

4.50. Exercise 1: workflow



4.51. Exercise 1: „View: Confusion Matrix" window for the "Scorer" node

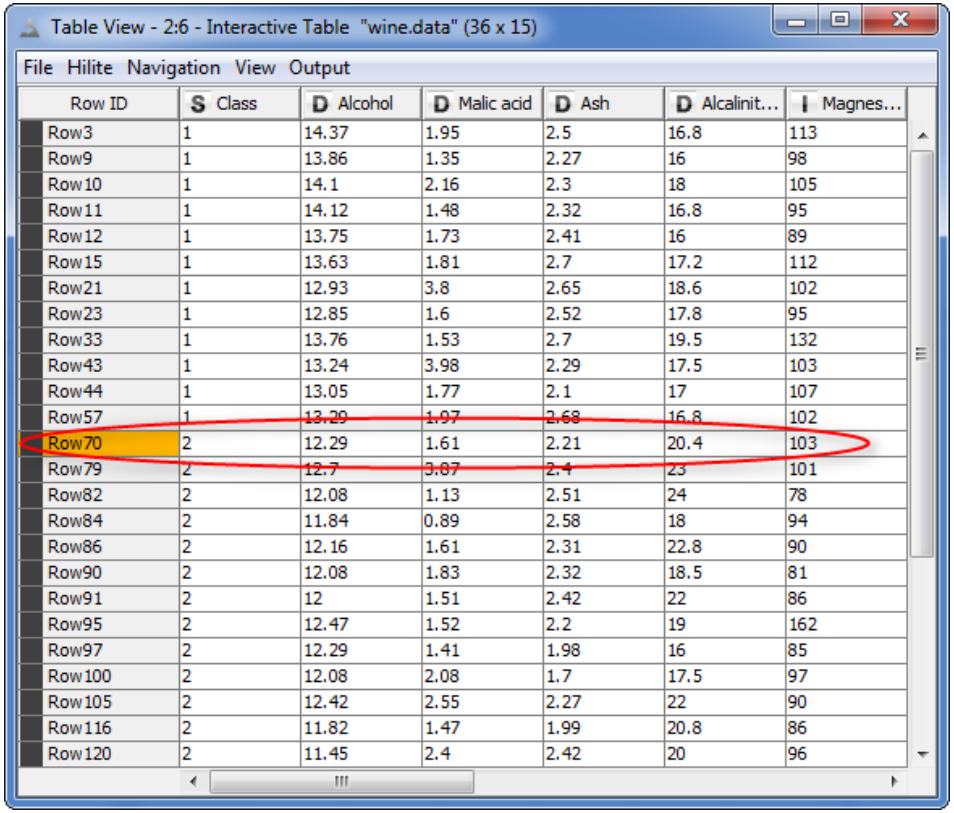


We open the "View: Confusion Matrix" option in the context menu and look for the number of records belonging to class 2 (Y-axis) that are misclassified as class 3 (X-axis).

There is only one record that has been misclassified to class 3 from class 2. We can use the hiliting function to find this record in the original test data table.

We hilited the cell in the confusion matrix that contains the number of such misclassified records. Next, we added an "Interactive Table" node to find the data row belonging to class 2 that has been assigned to class 3 in the data set resulting from the "Decision Tree Predictor" node. As a result of the interactive brushing property supported by the "Interactive Table" node and the "Scorer" node, the hilited cells in the Confusion Matrix View of the "Scorer" node hilite the corresponding records in the view of the "Interactive Table" node. The "Interactive Table" node was connected to the output port of the "Decision Tree Predictor" node. If we now open the option "View: Table View" in the context menu of the "Interactive Table" node, we find that one record is hilited. This is the misclassified record we have been looking for.

4.52. Exercise 1: Table View of the "Interactive Table" node.
The hilited row is the row producing the one misclassified record in the confusion matrix of the "Scorer" node.



Row ID	Class	Alcohol	Malic acid	Ash	Alcalinit...	Magnes...
Row3	1	14.37	1.95	2.5	16.8	113
Row9	1	13.86	1.35	2.27	16	98
Row10	1	14.1	2.16	2.3	18	105
Row11	1	14.12	1.48	2.32	16.8	95
Row12	1	13.75	1.73	2.41	16	89
Row15	1	13.63	1.81	2.7	17.2	112
Row21	1	12.93	3.8	2.65	18.6	102
Row23	1	12.85	1.6	2.52	17.8	95
Row33	1	13.76	1.53	2.7	19.5	132
Row43	1	13.24	3.98	2.29	17.5	103
Row44	1	13.05	1.77	2.1	17	107
Row57	1	13.29	1.97	2.68	16.8	102
Row70	2	12.29	1.61	2.21	20.4	103
Row79	2	12.7	3.87	2.4	23	101
Row82	2	12.08	1.13	2.51	24	78
Row84	2	11.84	0.89	2.58	18	94
Row86	2	12.16	1.61	2.31	22.8	90
Row90	2	12.08	1.83	2.32	18.5	81
Row91	2	12	1.51	2.42	22	86
Row95	2	12.47	1.52	2.2	19	162
Row97	2	12.29	1.41	1.98	16	85
Row100	2	12.08	2.08	1.7	17.5	97
Row105	2	12.42	2.55	2.27	22	90
Row116	2	11.82	1.47	1.99	20.8	86
Row120	2	11.45	2.4	2.42	20	96

Note. "Decision Tree" nodes need at least one nominal value to be used as the classification column.

The "File Reader" node reads the wine data class as Integer, since the classes are "1", "2", and "3". To make this a nominal value, go to the configuration window of the "File Reader" node and right-click the column header and select Type = "String".

Exercise 2

Build a training set (80%) and a test set (20%) from the wine.data. Train a Multilayer Perceptron (MLP) on the training set to classify the data according to the values in column "Class".

Next, apply the MLP to the test set and measure the classification entropy.

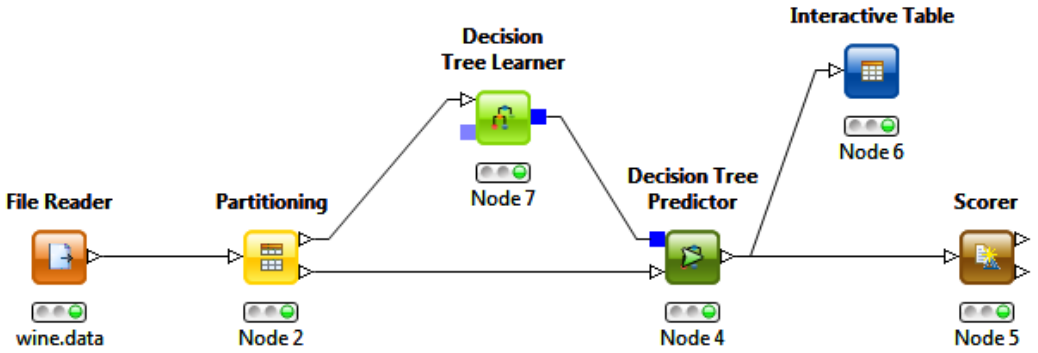
Solution to Exercise 2

We use a "Normalizer" node to scale the data before feeding them into the MLP.

Since the wine dataset is very small, we used the whole data set to define the normalization parameters.

The next step involved using a neural network with only one output neuron to model the three class values: "1", "2", and "3".

4.53. Exercise 2: workflow



As a neuron has a continuous output value, its output has to be post-processed to assign a class in the form of "1", "2", and "3" to each data row. To do this we used a "Rule Engine" node that implements the following rule:

```
IF       $neuron output$ <= 0.3       => class 1
ELSE IF  $neuron output$ > 0.3  AND  $neuron output$ < 0.6      => class 2
ELSE IF  $neuron output$ >= 0.6       => class 3
```

The entropy measure can be found in the "Entropy Scorer" node, where we need to compare the original class values with the neural network's predicted class values.

Exercise 3

Read the data "web site 1.txt" with a File Reader node. This data set describes the number of visitors to a web site for the year 2013.

Compute a few statistical parameters on the number of visitors, such as the mean and the standard deviation. Train a Naïve Bayesian Network on the number of visitors to discover whether a specific data row refers to a weekend or to a business day.

Finally, draw the ROC curve to visualize the Naïve Bayesian Classifier performance.

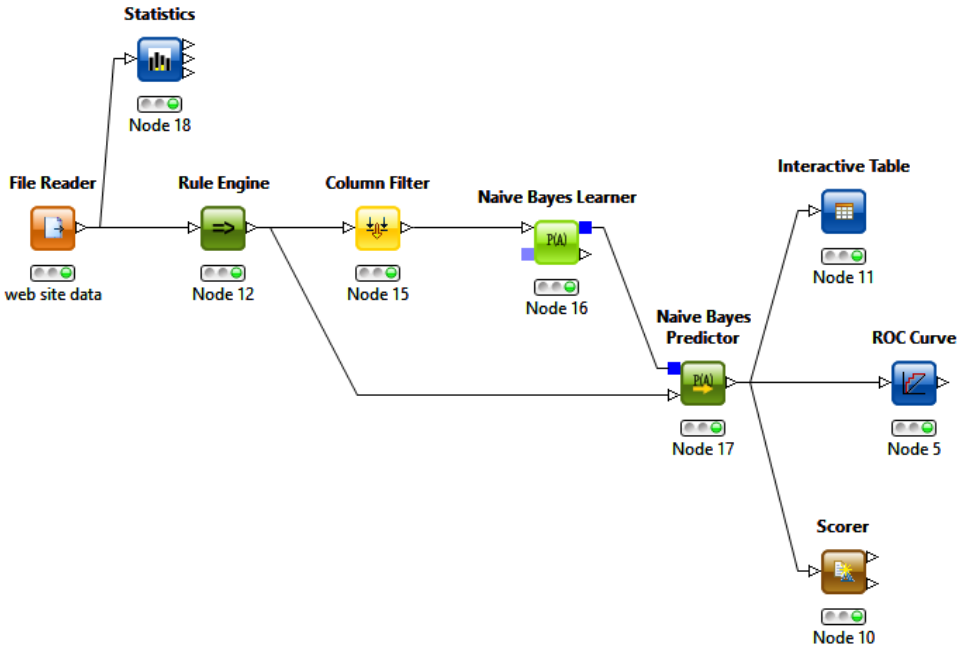
Solution to Exercise 3

We used a "Rule Engine" node to translate the column called "Day of Week" into a "weekend/not weekend" binary class.

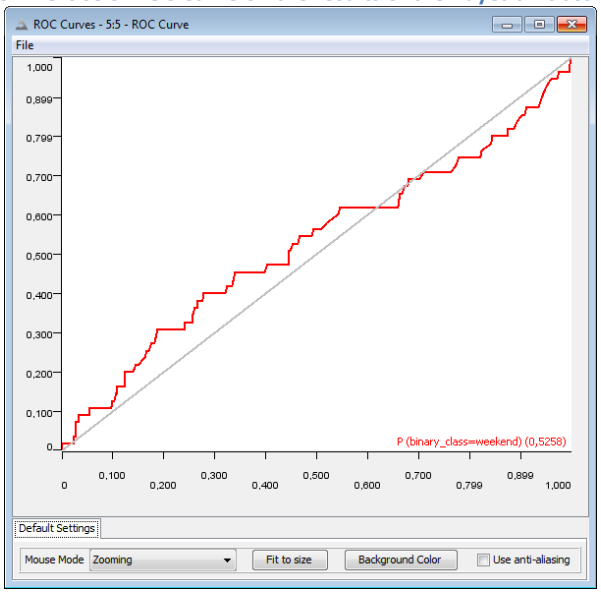
We filtered out the “Day of Week” column as not to make the classification task too easy for the Naïve Bayesian Classifier.

We trained the Bayesian Network on the binary class “weekend/not weekend” and we built the ROC curve on the “weekend” class probability (Fig. 4.55).

4.54. Exercise 3: workflow



4.55. Exercise 3: ROC Curve on the results of the Bayesian classifier



Chapter 5. The Workflow for my First Report

5.1. Introduction

The KNIME Report Designer is based on BIRT (Business Intelligence Reporting Tool), which is open source software for reporting.

BIRT and KNIME communicate inside the same platform. For each workflow I can open a default pre-set report and from each report I can go back to the original workflow just by clicking a button. The KNIME workflow generates the data, while BIRT takes this data and puts them inside a report layout. The data generated by the KNIME workflow and used by the BIRT report are automatically updated every time I move into the report space.

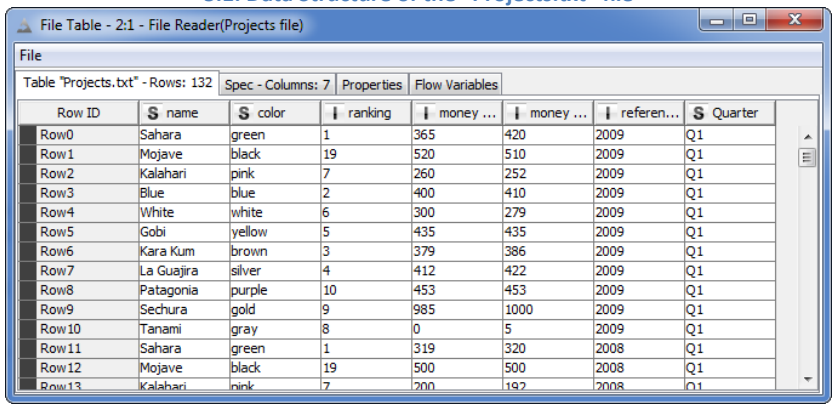
Usually reporting is only one part of a bigger data analysis structure. For example it can be used to show the model scores to the management board or to quantify performances for your boss. In this case, it is convenient to save the intermediate data into some history files, in order to be able to easily replicate the reports or to proceed with further data analysis later on. Generally, the KNIME workflow underlying the report is implemented to produce the data in almost the exact shape that is required by the report. A number of KNIME nodes are available to help us in this data manipulation task.

In this chapter we show how to build a report; that is how to implement a workflow to prepare the data for the report

In the next chapter we learn how to shape the report itself in the reporting tool; i.e., how to edit the report layout and how to switch back and forth from the KNIME environment to the BIRT environment.

During this chapter we build an example workflow to demonstrate how to generate data for a report. Before continuing, let's create a new workflow group "Chapter5" and open a new workflow with name "Projects".

5.1. Data Structure of the "Projects.txt" file



Row ID	name	color	ranking	money ...	money ...	referen...	Quarter
Row0	Sahara	green	1	365	420	2009	Q1
Row1	Mojave	black	19	520	510	2009	Q1
Row2	Kalahari	pink	7	260	252	2009	Q1
Row3	Blue	blue	2	400	410	2009	Q1
Row4	White	white	6	300	279	2009	Q1
Row5	Gobi	yellow	5	435	435	2009	Q1
Row6	Kara Kum	brown	3	379	386	2009	Q1
Row7	La Guajira	silver	4	412	422	2009	Q1
Row8	Patagonia	purple	10	453	453	2009	Q1
Row9	Sechura	gold	9	985	1000	2009	Q1
Row10	Tanami	gray	8	0	5	2009	Q1
Row11	Sahara	green	1	319	320	2008	Q1
Row12	Mojave	black	19	500	500	2008	Q1
Row13	Kalahari	pink	7	200	192	2008	Q1

We will use the data "Projects.txt" available in the book's "Download Zone". This file describes the evolution, in terms of money, of 11 fictitious projects across 2007, 2008, and 2009.

Each project is identified by the name of a different desert. Each project also has a unique color and a unique priority ranking, which remain the same over the years. Finally, the file describes the amount of money assigned to each project and the amount of money actually used by each project for each quarter of each year. After reading the file with a "File Reader" node, we get the data structure as in Figure 5.1. We also assign the name "Projects File" to the File Reader node, in order to quickly understand which data this node is applied to.

5.1. Installing the Report Designer Extension

The "KNIME Report Designer" is not included in the basic standalone version of KNIME. It can be downloaded as a separate extension package from the "KNIME.com Extensions" in the "Help" -> "Install New Software" window.

To install the "KNIME Report" package:

- Start KNIME
- In the Top Menu click "Help" -> "Install New Software …"
- In the "Available Software" window in the text box "Work with", select "KNIME Update Site – http://www.knime.org/update/2.x"
- Expand "KNIME & Extensions"
- Select the package "KNIME Reporting"
- Click the button "Next" on the bottom

If the installation runs correctly, after KNIME is restarted, you should have:

- A new category "Reporting" in the "Node Repository" panel with a "Data To Report" node
- A new option called "Edit Report" in the context menu of the workflows in the "KNIME Explorer" panel

5.2. Transform Rows

The goal of this chapter is to prepare the data for a report. Usually, data needs to reach the report in a predefined form. In this section we explore a few KNIME nodes that can help us to reach the desired data set structure.

The data for the report comes from the file "Projects.txt", which contains a list of projects and details how much money has been assigned to or used by each project during the years 2007, 2008, and 2009. We want to show 3 tables and 2 charts in the report.

The first table should show the project names in the row headers, the years in the column headers, and how much money has been assigned in total to each project for each year in the table cells (Fig. 5.2).

The second table has the same structure, but it shows how much money has been used in total by each project for each year in the table cells (Fig. 5.2).

The third table has the same structure as the two tables above and shows the remaining amount of money (= money assigned - money used) for each project for each year (Fig. 5.2).

The first chart should show the total amount of money assigned to each project (y-axis) over the three years (x-axis). The BIRT chart report item is fed with a data set where the values for x-axis and the values for y-axis are listed in two different columns; that is a data set where the year and the corresponding sum of money belong to the same row (Fig. 5.3).

The second chart has the same structure as the first chart, but shows the total amount of money used instead of the total amount of money assigned. That is, the chart must show the total amount of money used by each project (y-axis) over the three years (x-axis). For this reason, it needs a data set with year and total money used by each project separated into different columns (Fig. 5.3).

5.2. Data structure required for the tables in the report ("Pivoting" node)

Project Name \ year	2007	2008	2009
Project 1	Sum (money) for project 1 in year 2007	Sum (money) for project 1 in year 2008	Sum (money) for project 1 in year 2009
Project 2	Sum(money)for project 2 in year 2007	…	
Project 3	..		

5.3. Data structure required for the charts in the report ("GroupBy" node)

Project Name	year	Sum(money)
Project 1	2009	Sum (money) for project 1 in year 2009
Project 2	2009	Sum (money) for project 2 in year 2009
Project 3	2009	Sum (money) for project 3 in year 2009
…	…	…

For both tables and charts structures, we need to calculate the sum of money (assigned, used, or remaining). However, we need to insert this sum into a different data structure. For example, the years are the column headers in one case and the column values in the other.

In KNIME we can produce an aggregation based on variables – in this case the sum of money based on year and project – and we can report the final aggregation values on tables with different structure by using two different nodes: the **GroupBy** node and the **Pivoting** node.

Both nodes (“GroupBy” and “Pivoting”) are located in the “Node Repository” panel in the `“Data Manipulation” -> “Row” -> “Transform"` category.

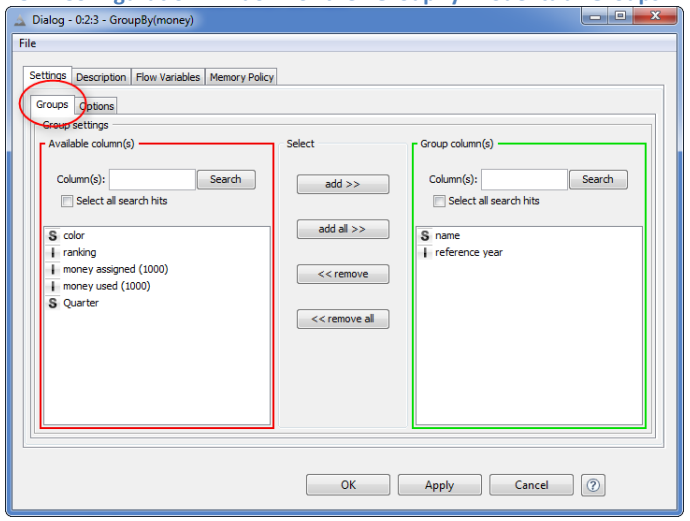
GroupBy

The “GroupBy” node finds groups of data rows by using the combination of values in one or more columns (**Group Columns**); it subsequently aggregates the values in other columns (**Aggregation Columns**) across those groups. Column values can be aggregated in the form of a sum or a mean or just a count of occurrences (**Aggregation Method)**.

The GroupBy node is located in category: `“Data Manipulation” -> “Row” -> “Transform"`

The configuration window of the “GroupBy” node consists of two tabs: “**Groups**” and “**Options**”.

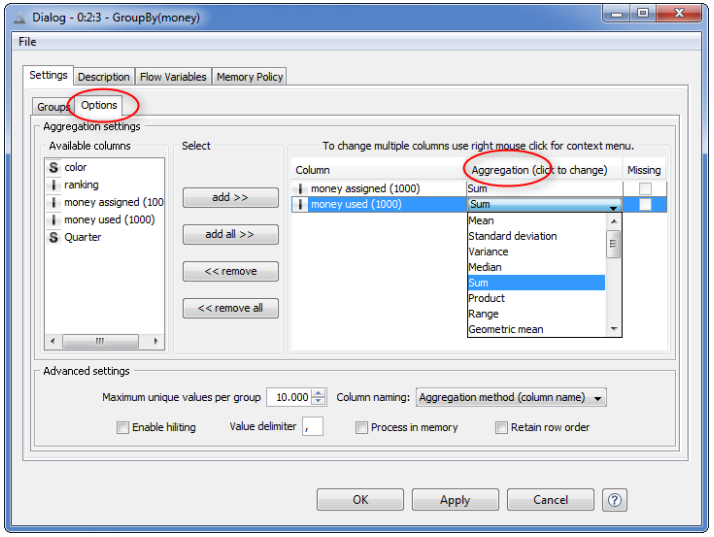
5.4. Configuration window for the "GroupBy" node: tab “Groups”



Tab **“Groups”** defines the grouping options. That is, it selects the group column(s) by means of an “Exclude”/”Include” frame:

- The still available columns for grouping are listed in the frame “Available column(s)”. The selected columns are listed in the frame “Group column(s)”.
- To move from frame “Available column(s)” to frame “Group column(s)” and vice versa, use the “add” and “remove” buttons. To move all columns to one frame or the other use the “add all” and “remove all” buttons.

5.5. Configuration window for the "GroupBy" node: tab “Options”



Tab **“Options”** defines the aggregation settings, that is:

- *The aggregation column(s)*
- *The aggregation method (one for each aggregation column)*
- *The name of the resulting column*

Several aggregation methods are available in the “Options” tab. The most frequently used are: Maximum, Minimum, Mean, Sum, Variance, Percent, and Count.

The name of the result column is by default “*Aggregation Method(column name)”*. We can either change it to “*Column name(Aggregation Method)*” or keep the original “*column name*”.

Pivoting

The “Pivoting” node finds groups of data rows by using the combination of values from **two or more** columns: the “**Pivot**” columns and the “**Group**” columns. It subsequently aggregates the values from a third group of columns (**Aggregation Columns**) across those groups. Column values can be aggregated in the form of a sum or a mean or just a count of occurrences (**Aggregation Methods)**.

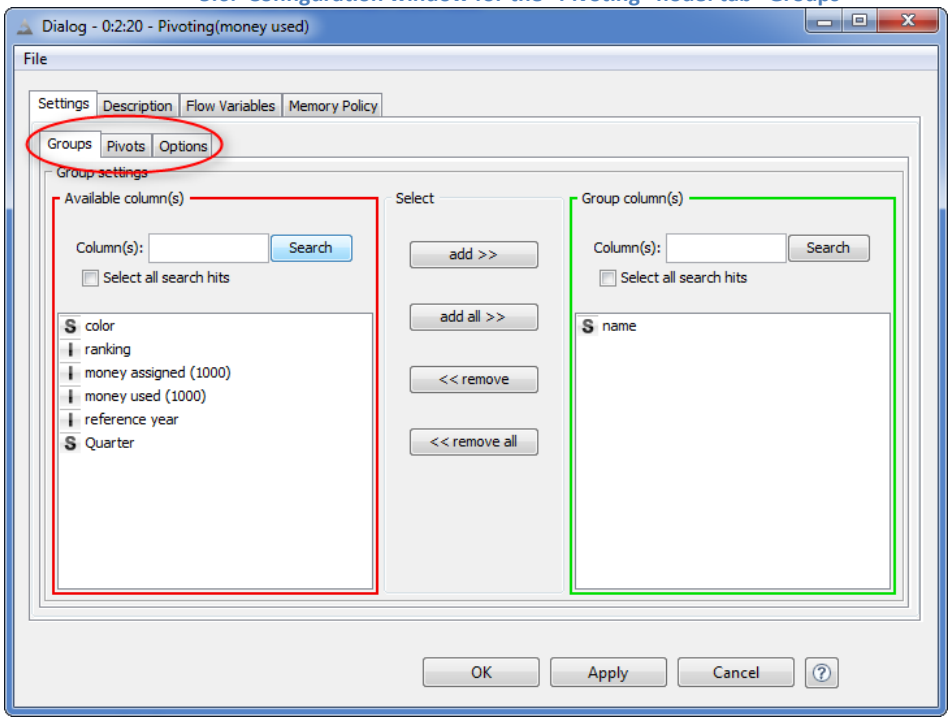
Once the aggregation has been performed, the data rows are reorganized in a matrix with “Pivot” column values as column headers and “Group” column values in the first columns.

The “Pivoting” node is located in the category `"Data Manipulation" -> "Row" -> "Transform"`

The “Pivoting” node has one input port and three output ports:

- The input port receives the data
- The first output port produces the pivot table
- The second output port produces the totals by group columns
- The third output port presents the totals by pivot columns

5.6. Configuration window for the "Pivoting" node: tab “Groups”



The “Pivoting” node is configured by means of three tabs: **“Groups”, “Pivots”,** and **“Options”.**

Tab “**Groups**” (Fig. 5.6) defines the group columns by means of an “Exclude”/”Include” frame:

- The still available columns for grouping are listed in the frame “Available column(s)”. The selected columns are listed in the frame “Group column(s)”.
- To move from frame “Available column(s)” to frame “Group column(s)” and vice versa, use the “add” and “remove” buttons. To move all columns to one frame or the other use the “add all” and “remove all” buttons.

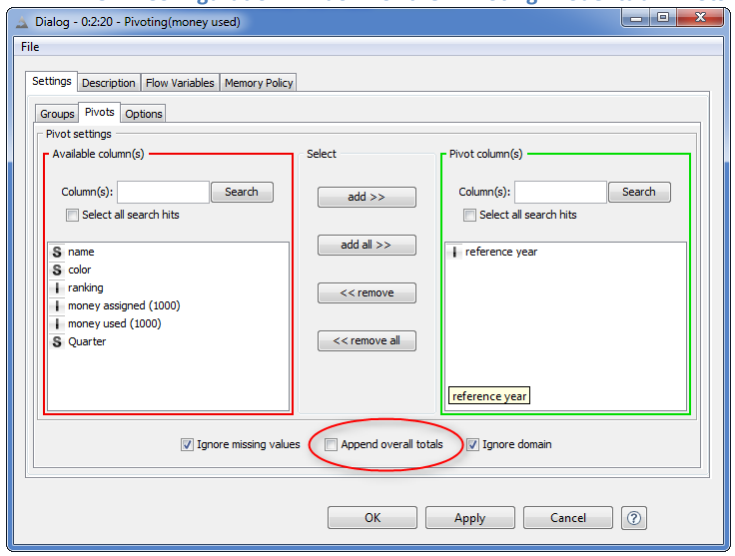
Tab “**Pivots**” (Fig. 5.7) defines the “Pivot” columns again by means of an “Exclude”/”Include” frame:

- The still available columns for grouping are listed in the frame “Available column(s)”. The selected columns are listed in the frame “Group column(s)”.
- To move from frame “Available column(s)” to frame “Pivot column(s)” and vice versa, use the “add” and “remove” buttons. To move all columns to one frame or the other use the “add all” and “remove all” buttons.

At the end of this tab window there are three flags:

- **“Ignore missing values”** ignores missing values while grouping the data rows
- **“Append overall totals”** appends the overall total in the output table “Pivot totals”
- **“Ignore domain”** groups data rows on the basis of the real values of the group and pivot cells and not on the basis of the data domain. This might turn out useful when there is a discrepancy between the real data values and their domain values (for example after using a node for string manipulation).

5.7. Configuration window for the "Pivoting" node: tab “Pivots”



Tab **“Options”** selects the aggregation columns and the aggregation method for each aggregation column. The column selection is again performed by means of an “Exclude”/”Include” frame:

For each selected aggregation column in the right frame, you need to choose an aggregation method.

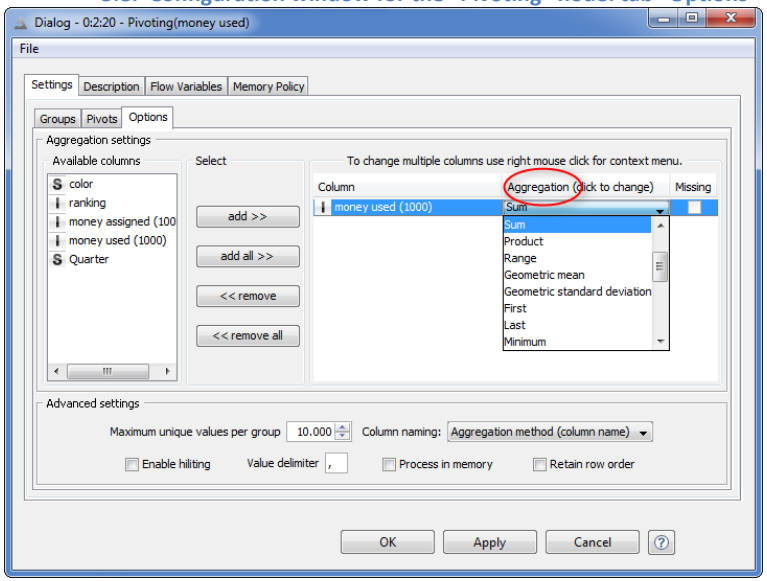
Several aggregation methods are available. The most frequently used are: Maximum, Minimum, Mean, Sum, Variance, Percent, and Count.

The “Column naming” box defines the naming criterion for the output columns, following the same naming criterion as for the “GroupBy” node.

Once the aggregation has been performed, the data rows are reorganized in the pivot table as follows:

- Column headers = <pivot columns distinct values> + <aggregation variable name selected criterion>
- First columns contain the group columns distinct values.

5.8. Configuration window for the "Pivoting" node: tab "Options"



In our “GroupBy” node, we calculated the sum (= aggregation method) of the values in the column *“money assigned (1000)”* and in the column *“money used (1000)”* (= multiple aggregation columns) for each group of rows defined by the combination of distinct values in the columns *“reference year”* and *“name”* (= Group Columns).

In the resulting data table, the first two columns contained all combinations of distinct values in the “*name*” and “*reference year*” columns. The aggregations were then run over the groups of data rows defined by each (“*name*”, “*reference year*”) pair. The aggregated values are displayed in two new columns “*Sum(money assigned (1000))*” and “*Sum(money used (1000))*”.
We named the new “GroupBy” node *“money by project by year”.*

5.9. Output data table of the GroupBy node

Group table - 2:3 - GroupBy(money by project by year)

File

Table "default" - Rows: 33 | Spec - Columns: 4 | Properties | Flow Variables

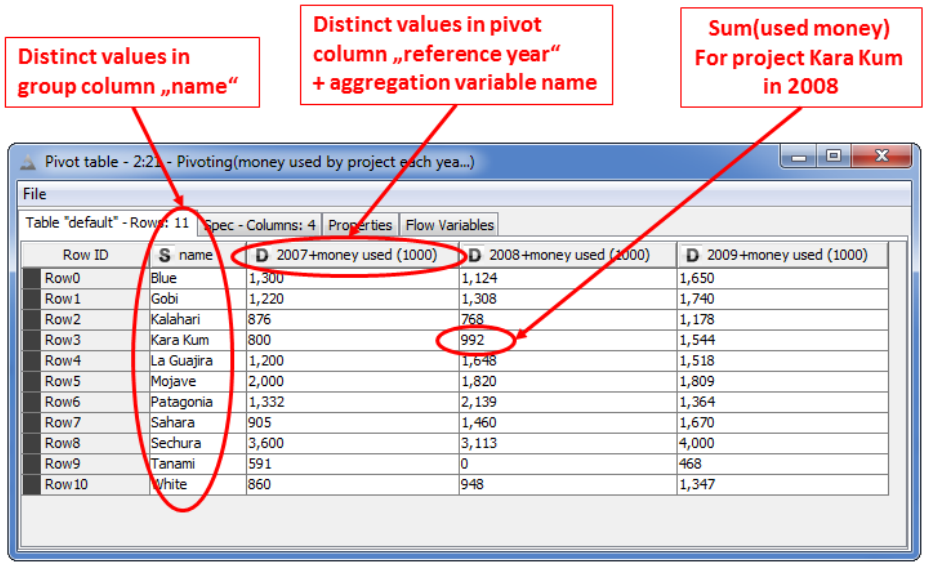
Row ID	name	reference year	Sum(money assigned (1000))	Sum(money used (1000))
Row0	Blue	2007	1,360	1,300
Row1	Blue	2008	1,277	1,124
Row2	Blue	2009	1,565	1,650
Row3	Gobi	2007	1,203	1,220
Row4	Gobi	2008	1,424	1,308
Row5	Gobi	2009	1,740	1,740
Row6	Kalahari	2007	630	876
Row7	Kalahari	2008	800	768
Row8	Kalahari	2009	1,192	1,178
Row9	Kara Kum	2007	800	800
Row10	Kara Kum	2008	888	992
Row11	Kara Kum	2009	1,516	1,544
Row12	La Guajira	2007	1,020	1,200
Row13	La Guajira	2008	1,404	1,648
Row14	La Guajira	2009	1,496	1,518
Row15	Mojave	2007	1,800	2,000
Row16	Mojave	2008	1,819	1,820
Row17	Mojave	2009	1,860	1,809

5.10. Output pivot table of the „Pivoting” node

We used two “Pivoting” nodes in the workflow named “Projects”.

In one “Pivoting” node we calculated the sum (= aggregation method) of the values in column *“money assigned(1000)”* (= aggregation column) for each combination of values in the *“reference year”* (= pivot column) and *“name”* (= group column) columns. In the other “Pivoting” node we calculated again the sum (= aggregation method) of the values in column *“money used(1000)”* (= aggregation column) for each combination of values in the *“reference year”* (= pivot column) and *“name”* (= group column) columns.

In both “Pivoting” nodes, we chose to keep the original names in the “Column naming” box.



The aggregated values are then displayed in a pivot table with <year + aggregation variable> as column headers, the project names in the first column, and the sum of "*money assigned(1000)"* or "*money used(1000)"* for each project and for each year as the cell content.

We named the new Pivoting nodes *"money assigned to project each year"* and *"money used by project each year"*.

In order to make the pivot table easier to read, we then moved the values of the project name column to become the RowIDs of the data table and we renamed the pivot column headers with only the reference year value. In order to do that, we used a "RowID" node and a "Column Rename" node respectively.

RowID

The RowID node can be found in the "Node Repository" panel in the `"Data Manipulation" -> "Row" -> "Other"` category.

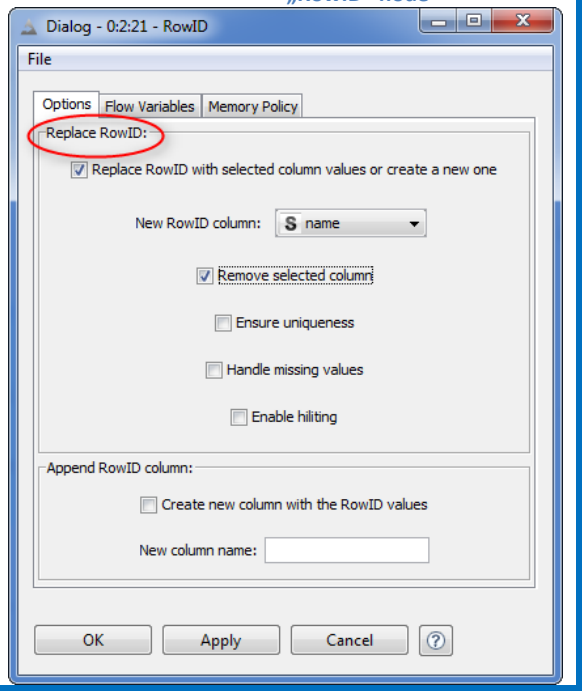
The RowID node allows the user to:

- Replace the current RowIDs with the values of another column (top half of the configuration window)
- Copy the current RowIDs into a new column (bottom half of the configuration window)

When replacing the current RowIDs a few additional options are supported.

- "Remove selected column" removes the column that has been used to replace the RowIDs.
- "Ensure uniqueness" adds an extension "(1)" to duplicate RowIDs. Extension becomes "(2)" or "(3)" etc… depending on how many duplicate values are encountered for this RowID.
- "Handle missing values" replaces missing values in RowIDs with default values.
- "Enable hiliting" keeps a map between the old and the new RowIDs to keep hiliting working in other nodes.

5.11. Configuration window for the „RowID" node



In the "Node Repository" panel, under the node "Pivoting" you can find a node "Unpivoting".

Although we are not going to use the "Unpivoting" node in our example workflow, it is worth it having a look at it.

Unpivoting

The “Unpivoting” node rotates the content of the input data table. This kind of rotation is shown in figures 5.13 and 5.14. Basically, it produces a “GroupBy”-style output data table from the “pivoted” input data table.

The “Unpivoting” node is located in the `“Data Manipulation” -> “Row” -> “Transform"` category.

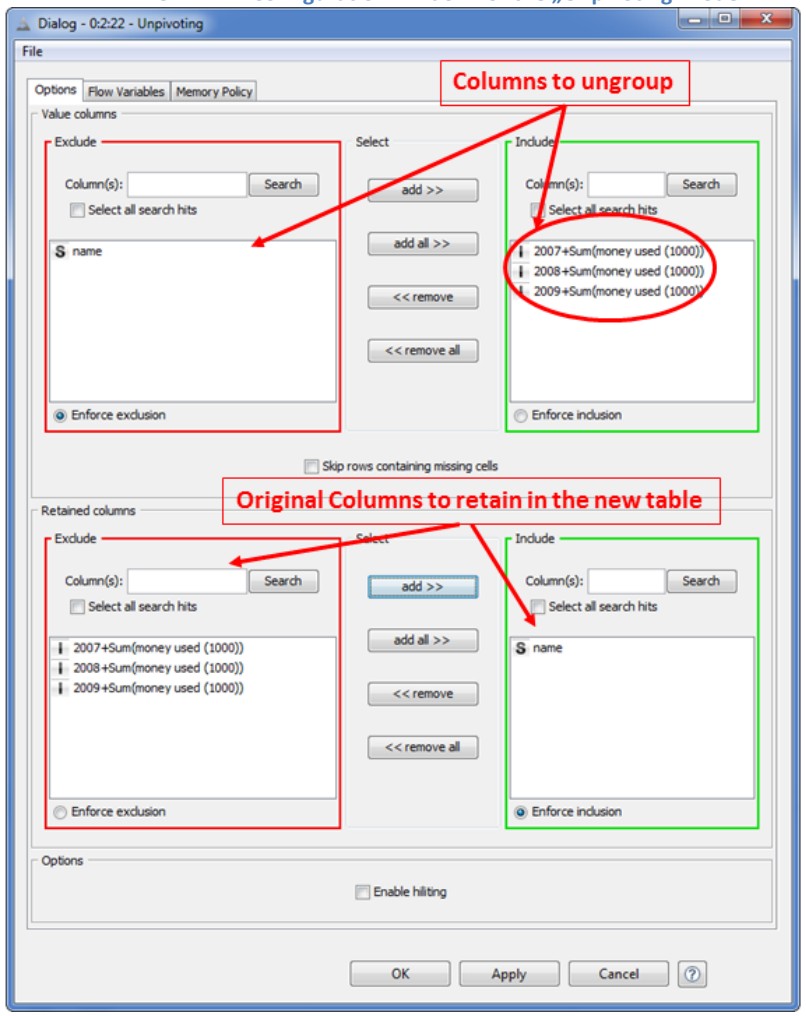
In the settings you need to define:

- Which columns should be used for the cells redistribution
- Which columns should be retained from the original data set

The unpivoting process produces 3 new columns in the data table:

- One column called “RowIDs”, which contains the RowIDs of the input data table
- One column called “ColumnNames”, which contains the column headers of the input data table
- One column called “ColumnValues”, which reconnects the original cell values to their RowID and column header

5.12. Configuration window for the „Unpivoting“ node



The column selection follows the already seen an “Exclude”/”Include” frame:

- The still available columns for grouping are listed in the frame “Available column(s)”. The selected columns are listed in the frame “Group column(s)”.
- To move from frame “Available column(s)” to frame “Group column(s)” and vice versa, use the “add” and “remove” buttons. To move all columns to one frame or the other use the “add all” and “remove all” buttons.

“Enforce Inclusion/Exclusion” keeps the included/excluded columns as fixed and adds possible new columns to the other column set.

5.13. Input Data Table

	Col1	Col2	Col3
ID 1	1	3	5
ID 2	2	4	6

5.14. Unpivoted Data Table

	RowIDs	ColumnNames	ColumnValues
Row 1	ID 1	Col1	1
Row 2	ID 1	Col2	3
Row 3	ID 1	Col3	5
Row 4	ID 2	Col1	2
Row 5	ID 2	Col2	4
Row 6	ID 2	Col3	6

Note. Notice that: **Pivoting + Unpivoting = GroupBy**

The output data tables of the "GroupBy" node and of the "Pivoting" node are sorted by the group columns' values (Fig. 5.9 and 5.10). If this were not the case, like in versions of KNIME previous to 2.4, we would need to sort the rows alphabetically according to their value in the "name" column.

The "Sorter" node, like the "Pivoting" and the "GroupBy" node, is another node that is frequently used for reporting. For demonstrative purposes we briefly show here the "Sorter" node.

Sorter

The “Sorter” node sorts the rows of a data table by sorting the values of one of its columns. In the settings you need to select:

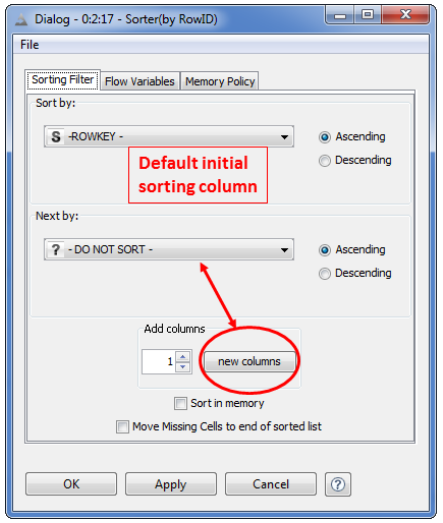
- The column(s) to be sorted (the RowIDs column is also accepted)
- Whether to sort in ascending or descending order

It is possible to sort the table by multiple columns. To add a new sorting column:

- Click the “new columns” button
- Select the new column

The first column (in the top part of the configuration window) gives the primary sorting; the second column gives the secondary sorting, and so on.

5.15. Configuration window for the„Sorter“ node



5.3. Joining Columns

After applying the two “Sorter” nodes, we now have two data tables with the same column structure:

- RowID containing the project names
- “2009” column with the used/assigned money for year 2009
- “2008” column with the used/assigned money for year 2008
- “2007” column with the used/assigned money for year 2007

Now, it would be useful to

- Have all values for used and assigned money over the years together for each project in the same row, for example:

RowID	assigned 2009	assigned 2008	assigned 2007	used 2009	used 2008	used 2007
<project name>	…	…	…	…	…	…

- Calculate the remaining money for each year for each project, as: *remain <year> = assigned <year> -used<year>*

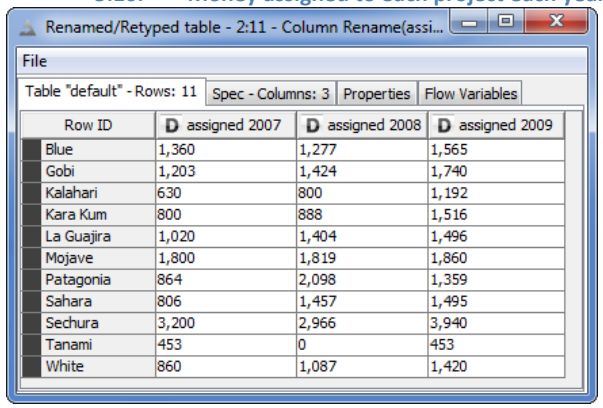
Basically, we want to join the two data tables, the table with the values for the assigned money and the table with the values for the used money, into one single table. After that, we want to calculate the remaining money values.

First of all, in order to be able to perform the table join without confusion, we need different columns to bear different names. We will see that actions need to be taken in case of a join of tables with columns with the same name. Let's connect a "Column Rename" node to each "RowID" node.

In the table resulting from the node "money used by project each year", let's rename the column called "2009 + money used(1000)" as just "used 2009", the column called "2008 + money used(1000)" to "used 2008", and the column called "2007 + money used(1000)" to "used 2007". In the table resulting from the node "money assigned to project each year", let's rename the column called "2009 + money assigned(1000)" to "assigned 2009", the column called "2008 + money assigned(1000)" to "assigned 2008", and the column called "2007 + money assigned(1000)" to "assigned 2007".

The data tables we want to join now have the structure reported in figures 5.16 and 5.17.

5.16. Money assigned to each project each year

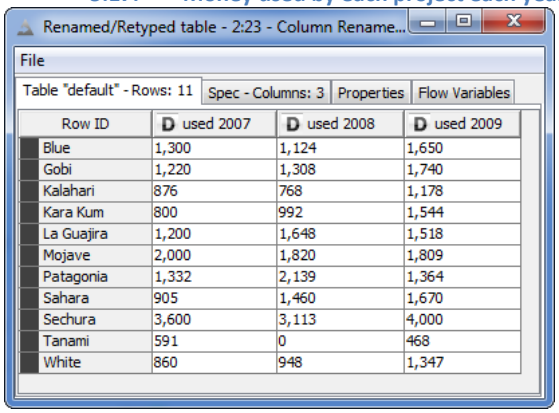

Renamed/Retyped table - 2:11 - Column Rename(assi...

File

Table "default" - Rows: 11 | Spec - Columns: 3 | Properties | Flow Variables

Row ID	assigned 2007	assigned 2008	assigned 2009
Blue	1,360	1,277	1,565
Gobi	1,203	1,424	1,740
Kalahari	630	800	1,192
Kara Kum	800	888	1,516
La Guajira	1,020	1,404	1,496
Mojave	1,800	1,819	1,860
Patagonia	864	2,098	1,359
Sahara	806	1,457	1,495
Sechura	3,200	2,966	3,940
Tanami	453	0	453
White	860	1,087	1,420

5.17. Money used by each project each year


Renamed/Retyped table - 2:23 - Column Rename...

File

Table "default" - Rows: 11 | Spec - Columns: 3 | Properties | Flow Variables

Row ID	used 2007	used 2008	used 2009
Blue	1,300	1,124	1,650
Gobi	1,220	1,308	1,740
Kalahari	876	768	1,178
Kara Kum	800	992	1,544
La Guajira	1,200	1,648	1,518
Mojave	2,000	1,820	1,809
Patagonia	1,332	2,139	1,364
Sahara	905	1,460	1,670
Sechura	3,600	3,113	4,000
Tanami	591	0	468
White	860	948	1,347

Now that we have the right data structure, we need to perform a table join. We want to join the cells to be in the same row based on the project name; i.e. in this case this is the RowID. In fact we want the row of used money for project "Blue" to be appended at the end of the corresponding row of the table with the assigned money. KNIME has a very powerful node that can be used to join tables, known as the "Joiner" node.

Joiner

The “Joiner” node is located in the “Node Repository” panel in `“Data Manipulation” -> “Column” -> “Split & Combine”`

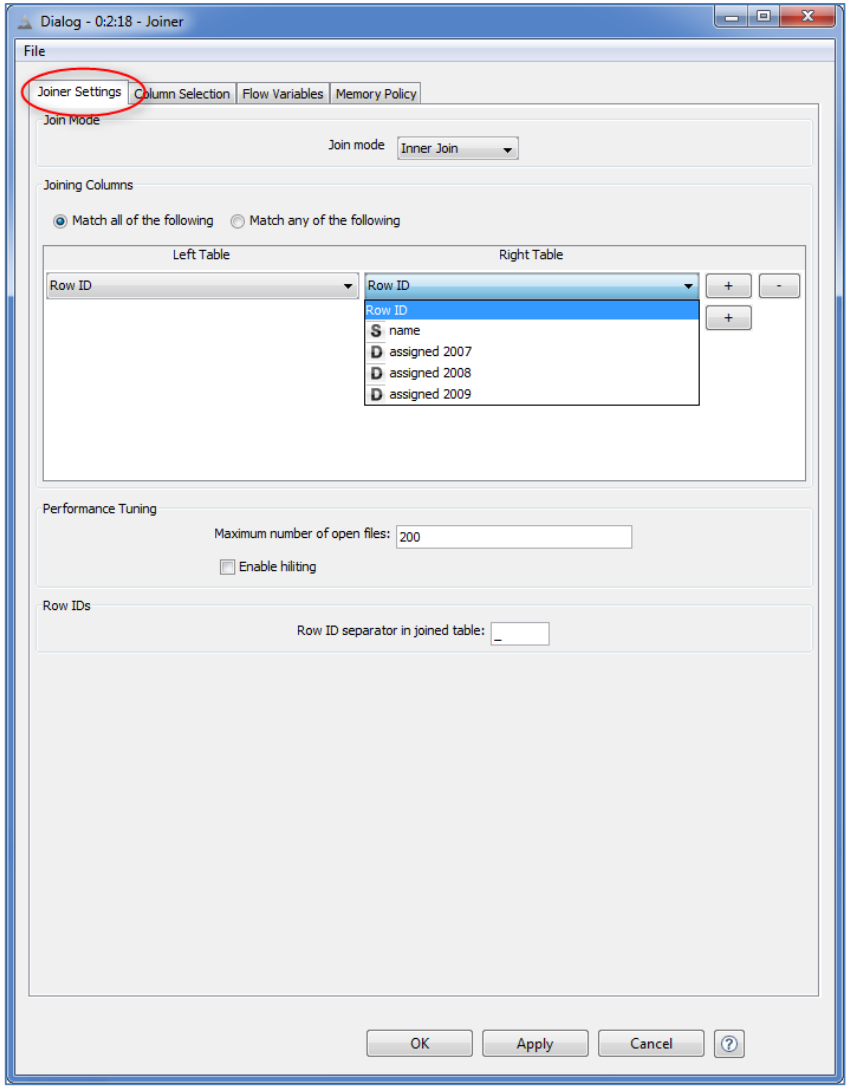
The “Joiner” node takes two data tables on the input ports and matches a column of the table on the upper port (left table) with a column of the table on the bottom port (right table). These columns can also be the RowID columns.

There are two tabs to be filled in, in the “Joiner” node’s configuration window.

- The “Joiner Settings” tab contains all the settings pertaining to:
  - The join mode
  - The columns to be matched
  - Other secondary parameters

- The “Column Selection” tab contains all settings pertaining to:
  - Which columns from the two tables to be included in the joined table
  - How to handle duplicate columns (i.e. columns with the same name)
  - Whether to filter out the joining column from the left and/or from the right data table or none at all

5.18. Configuration window for the „Joiner“ node, which includes two tabs to be filled in: “Joiner Settings” and “Column Selection”



Joiner node: the „Joiner Settings" tab

The "Joiner Settings" tab sets the basic joining properties, like the "join mode", the "joining columns", the "matching criterion", and so on.

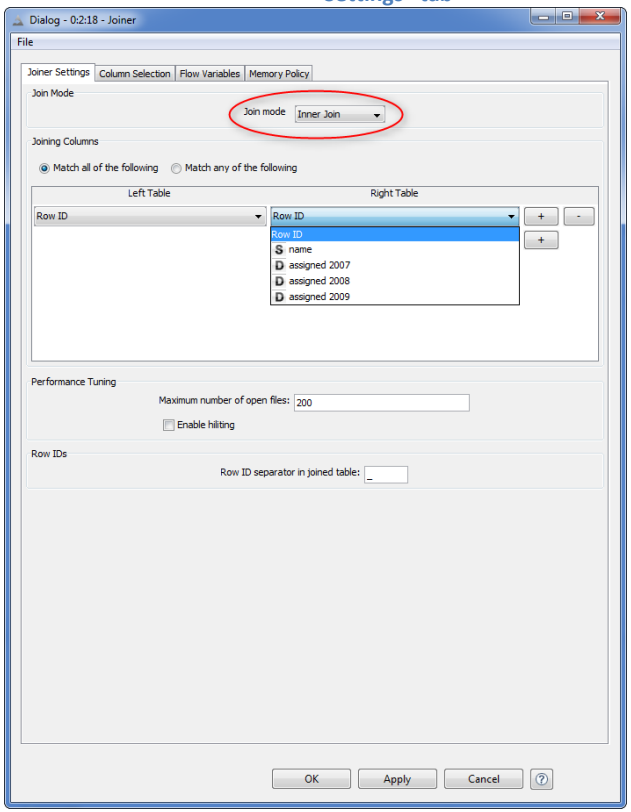
The first setting is the **join mode** (Fig. 5.21).

- **Inner join** keeps all rows where the two joining columns match;
- **Left join** keeps all rows from the left table and matches them with the rows of the right table, if the match exists;
- **Right join** keeps all rows from the right table and matches them with the rows of the left table, if the match exists;
- **Outer join** keeps all rows from the left and right tables and matches them with each other, if the match exists.

The **columns to match** can be selected in the "Joining Columns" panel. Joining on multiple columns is supported. To add a new pair of joining columns:

- Click the "+"button;
- Select the joining column for the left table and the joining column for the right table.

5.19. Configuration window for the „Joiner" node: "Joiner Settings" tab



There are two **matching criteria**.

- **Match all of the following.** A row from the left input table and a row from the right input table match if the values of all specified column pairs match.
- **Match any of the following:** A row from the left input table and a row from the right input table match if the values of at least one specified column pair match.

Other settings:

- "**Maximum number of open files**" is a performance parameter and defines the maximum number of temporary files open at the same time.
- If tables are not joined on the RowIDs, a new RowID is created by concatenating the two original RowIDs as string. "**RowID separator in joiner table**" sets the separator character in between the two original RowIDs.

Joiner node: the "Column Selection" tab

Tab "Column Selection" defines how to handle the columns that are not involved in the match process.

Once we have two rows that match we can keep some or all of the columns with unique headers from the left and the right table.

The c**olumn selection"** panel is applied to both input data tables (left and right) by means of an "Exclude"/"Include" frame.

- The columns to be kept in the new joined table are listed in frame "Include". All other columns are listed in frame "Exclude".
- To move from frame "Include" to frame "Exclude" and vice versa, use buttons "add" and "remove". To move all columns to one frame or the other use buttons "add all" and "remove all".

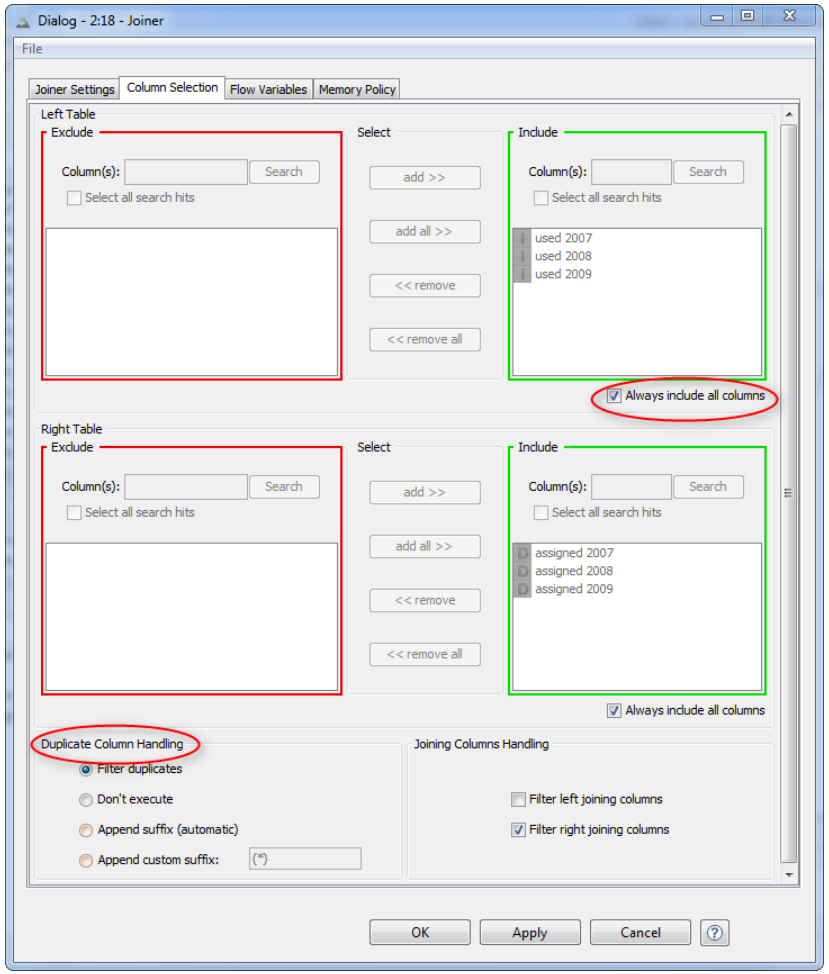
Under each "Column Selection" frame there is a flag called "**Always include all columns"**. If this flag is enabled, the "Column Selection" frame is disabled and all columns from this table are retained in the joined output data table.

The **"Duplicate Column Handling"** panel offers a few options to deal with the problem of columns with the same header (= duplicate columns) in the two tables.

- "**Filter duplicates**" filters out the duplicate columns of the right table and keeps those of the left table
- "**Do not execute**" produces an error
- "**Append suffix**" appends a suffix, default or customized, to the name of the duplicate columns in the right table

"Joining Columns Handling" defines whether the joining columns of the left and/or of the right table or none are filtered out.

5.20. Configuration window for the „Joiner" node: "Column Selection" tab



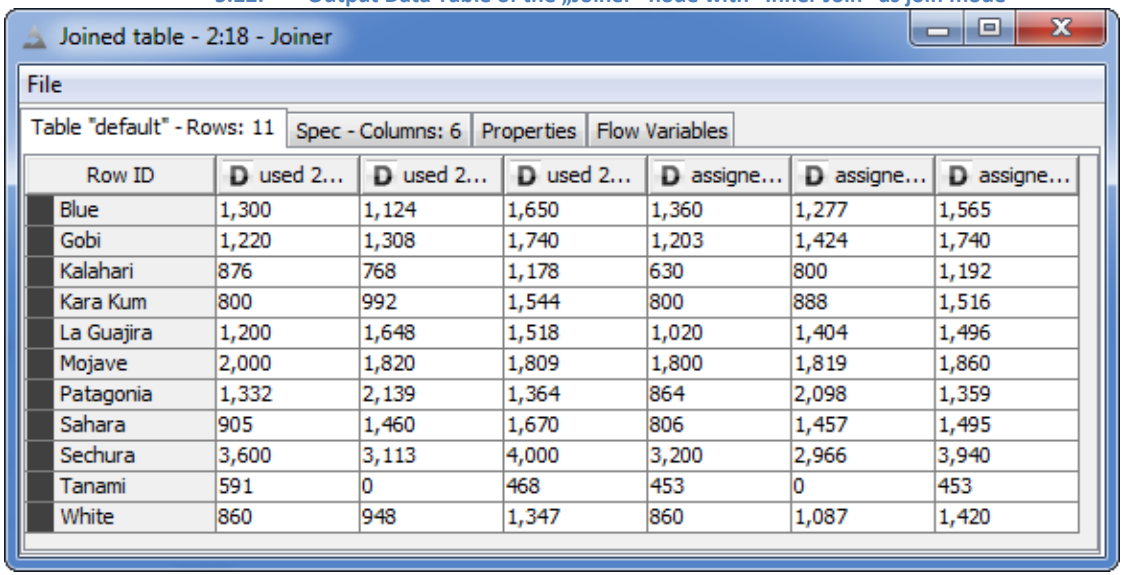
Join mode

We joined the two tables (money assigned and money used) using the RowIDs as the joining column for both; we chose to filter out the columns from the left table with the same name as the columns from the right table; and we chose the inner join as join mode.

The resulting data is shown in figure 5.22. You can see that now the "assigned money" values and the "used money" values are on the same row for each project.

It is of course possible to make the join on different columns than the RowIDs columns. However, the joining on RowID allows the user to keep the original RowID values which might be important for some subsequent data analysis or data manipulation. In this case we need the RowIDs to contain the joining keys. In order to manipulate the RowID values, KNIME has a "RowID" node.

5.22. Output Data Table of the „Joiner" node with "Inner Join" as join mode



Row ID	D used 2...	D used 2...	D used 2...	D assigne...	D assigne...	D assigne...
Blue	1,300	1,124	1,650	1,360	1,277	1,565
Gobi	1,220	1,308	1,740	1,203	1,424	1,740
Kalahari	876	768	1,178	630	800	1,192
Kara Kum	800	992	1,544	800	888	1,516
La Guajira	1,200	1,648	1,518	1,020	1,404	1,496
Mojave	2,000	1,820	1,809	1,800	1,819	1,860
Patagonia	1,332	2,139	1,364	864	2,098	1,359
Sahara	905	1,460	1,670	806	1,457	1,495
Sechura	3,600	3,113	4,000	3,200	2,966	3,940
Tanami	591	0	468	453	0	453
White	860	948	1,347	860	1,087	1,420

5.4. Misc Nodes

In our report we want to include the remaining money for each year, calculated as: `<remaining value> = <assigned value> - <used value>`. There are two ways to calculate this value: the "Math Formula" node and the "Java Snippet" nodes. All of these nodes are located in the "Misc" category.

The "Java Snippet" nodes allow the user to execute pieces of Java code. We can then use a "Java Snippet" node to calculate the amount <remaining value>. Actually, we will use three "Java Snippet" nodes: one to calculate the <remaining value 2009>, a second one to calculate the <remaining value 2008>, and a third one to calculate the <remaining value 2007>. We name the three "Java Snippet" nodes "remain 2009", "remain 2008", and "remain 2007".

There are two types of "Java Snippet" nodes: the "Java Snippet" node and the "Java Snippet (simple)" node. The functionality is the same: run a pice a Java code. However, the "Java Snippet" node has a more complex and more flexible GUI, while the "Java Snippet (simple)" node offers a more simplified GUI. That is, the "Java Snippet" node is for more expert users and more complex pieces of code, while the "Java Snippet (simple)" node is for medium expert users and more simple pieces of Java code.

Java Snippet (simple)

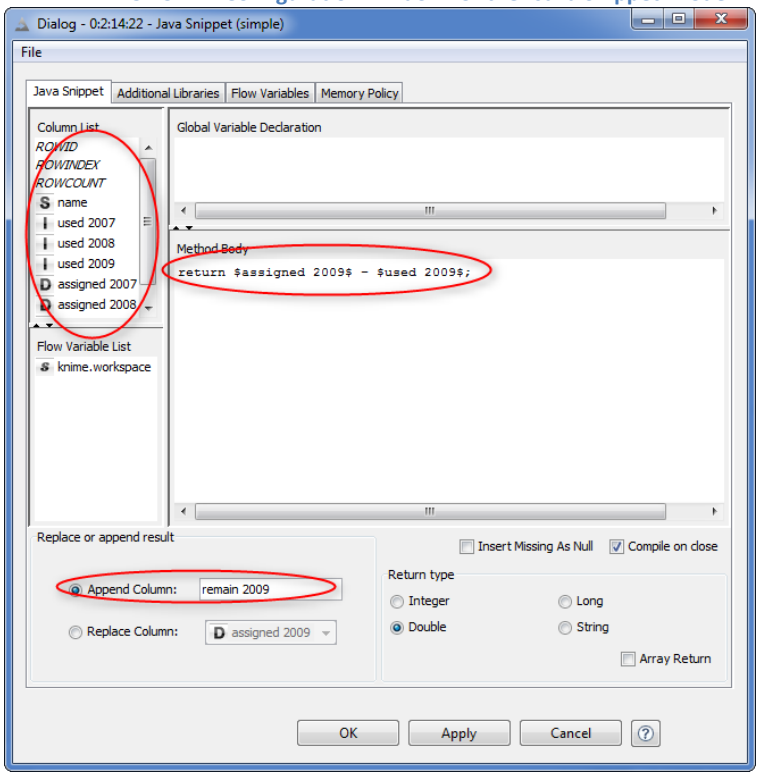
A “Java Snippet (simple)” node allows the execution of a piece of Java code and places the result value into a new or existing data column. The code has to end with the keyword “return” followed by the name of a variable or expression.

The “Java Snippet (simple)” node is located in the “Node Repository” panel in the `“Misc” -> “Java Snippet”` category.

When opening the node’s dialog for configuration, a window appears including a number of panels.

- The **Java editor** is the central part of the configuration window. This is where you write your code. Please remember that the code has to return some value and therefore it has to finish with the keyword “return” followed by a variable name or expression. Multiple “return” statements are not permitted in the code.
- The **list of column names** is on the top left-hand side. The column names can be used as variables inside the Java code. After double-clicking the column name, the corresponding variable appears in the Java editor. Variables carry the type of their original column into the java code, i.e.: Double, Integer, String and Arrays. Array variables come from columns of the type Collection Type.

5.23. Configuration window for the “Java Snippet” node



- The **name and type of the column** to append or to replace. The column type can be “Integer”, “Double”, or “String”. If the column type does not match the type of the variable that is being returned, the Java snippet code will not compile.
- It is also possible to **return arrays** instead of single variables (see the checkbox on the bottom right). In this case a number of columns (as many as the array length) will be appended to the output data table.
- In the “**Global Variable Declaration**” panel global variables can be created, to be used recursively in the code across the table data rows.
- The “**Additional Libraries**” tab allows the inclusion of non-standard Java Libraries.
- In the “**Global Variable Declaration**” panel global variables can be created, to be used recursively in the code across the table data rows.

For node “remain 2009” we used the Java code: `return $assigned 2009$ - $used 2009$`

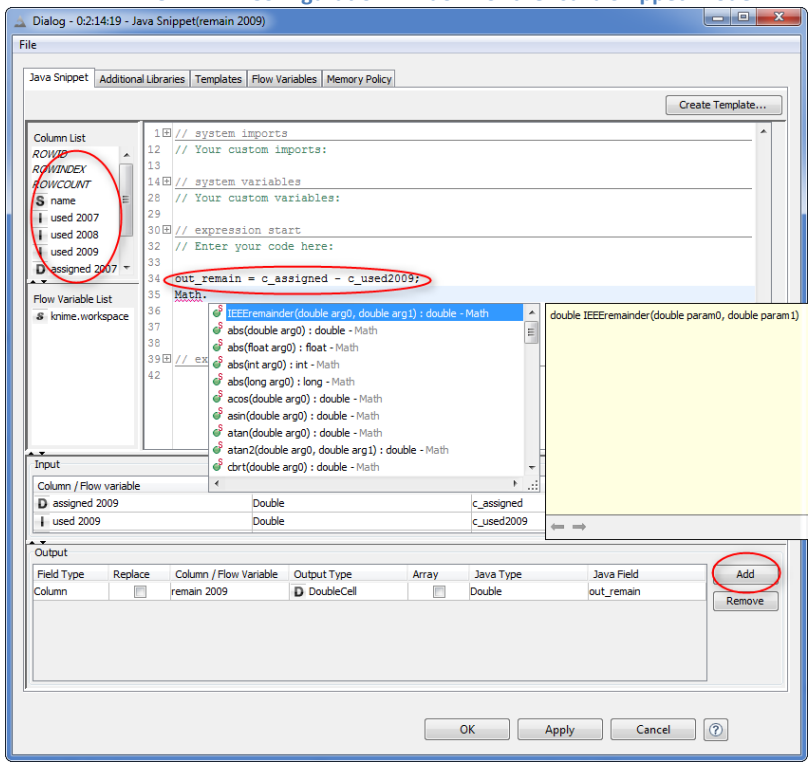
The same code could be used with other two Java snippet nodes to calculate “remain 2008” and “remain 2009”. The same task could have been accomplished with a “Java Snippet” node.

Java Snippet

Like the “Java Snippet (simple)” node, the “Java Snippet” node allows the execution of a piece of Java code and places the result value into a new or existing data column. The node is located in the “Node Repository” panel in the `“Misc” -> “Java Snippet”` category.

The configuration window of the “Java Snippet” node also contains:

5.24. Configuration window for the “Java Snippet” node



- The **Java editor.** This is the central part of the configuration window and it is the main difference with the “Java Snippet (simple)” node. The editor has sections reserved for: variable declaration, imports, code, and cleaning operations at the end.
    - The **“expression_start”** section contains the code.
    - The **“system variables”** section contains the global variables, those whose value has to be carried on row by row. Variables declared inside the “expression_start” section will reset their value at each row processing.
    - The “**system imports**” section is for the library import declaration.

  Self-completion is also enabled, allowing for an easier search of methods and variables. One or more output variables can be exported in one or more new or existing output data columns.
- The **table named “Input”.** This table contains all the variables derived from the input data columns. Use the “Add” and “Remove” buttons to add input data columns to the list of variables to be used in the Java code.
- The **list of column names** on the top left-hand side. The column names can be used as variables inside the Java code. After double-clicking the column name, the corresponding variable appears in the Java editor and in the list of input variables on the bottom. Variables carry the type of their original column into the java code, i.e.: Double, Integer, String and Arrays. Their type though can be changed (where possible) by changing the field “Java Type” in the table named “Input”.
- The **table named “Output”**. This table contains the output data columns to be created as new or to be replaced by the new values. To add a new output data column, click the “Add” button. Use the “Add” and “Remove” button to add and remove output data columns. Enable the flag “Replace” if the data column is to override an existing column. The data column type can be “Integer”, “Double”, or “String”. If the column type does not match the type of the variable that is being returned, the Java snippet code will not compile. It is also possible to **return arrays** instead of single variables, just by enabling the flag “Array”. Remember to assign a value to the output variables in the Java code zone.

Both “Java Snippet” nodes are very powerful nodes, since they allow the user to deploy the power of Java inside KNIME. However, most of the times, such powerful nodes are not necessary. All mathematical operations, for example, can be performed by the “Math Formula” node. The “Math Formula” node is optimized for mathematical operations and therefore tends to be faster than the “Java Snippet” nodes.

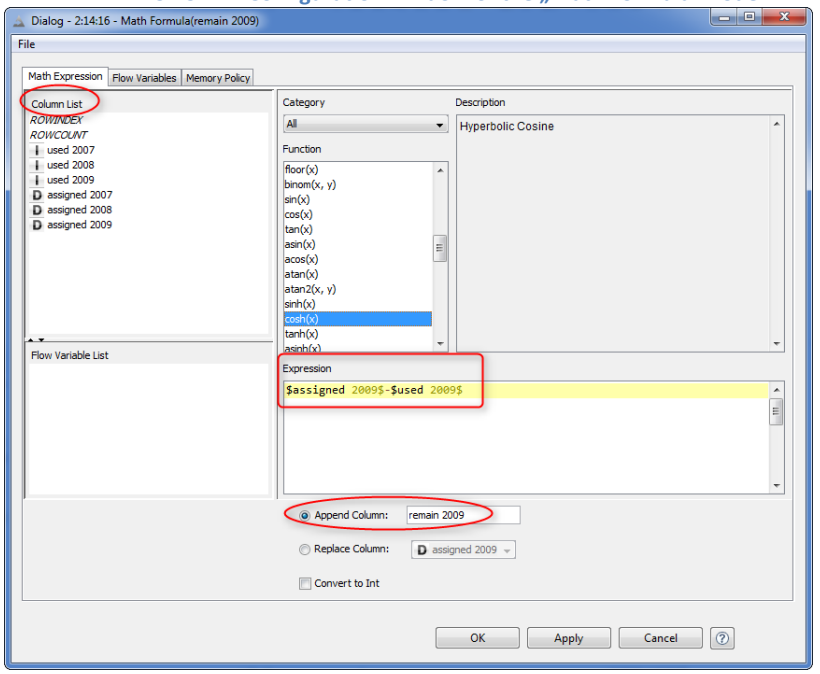
Math Formula

The “Math Formula” node enables mathematical formulas to be implemented and works similarly to the “String Manipulation” node (see section 3.5).
The “Math Formula” node is not part of the basic standalone KNIME. It has to be downloaded with the extension package (see par. 1.5) “*KNIME Math Expression Extension (JEP)*”. Once the extension package has been installed, the Math Formula node is located in the “Node Repository” panel in the “Misc” category.

In the configuration window there are 3 lists:

- The **list of column names** from the input data table
- The list of variables (to see in the “KNIME Cookbook”)
- A list of **mathematical functions,** e.g. log(x).
- The **expression editor**

5.25. Configuration window of the „Math Formula“ node



Double-clicking data columns from the list on the left automatically inserts them in the expression editor. You can complete the math expression by typing in what’s missing. Here, like in the “Java Snippet” nodes, $<column_name>$ indicates the usage of a data column.

A number of functions to build a mathematical formula are available in the central list.

At the bottom you can insert the **name of the column** to be appended or replaced.
The node exports double type data, but integer type data can also be exported by enabling the “Convert to Int” option.

In the configuration window shown in figure 5.23 we have implemented the same calculation of values <remain 2009> mentioned in the “Java Snippet” node earlier in this chapter. We simply needed to:

- Double-click the two columns $used 2009$ and $assigned 2009$ in the column list
- Type a character "-"between the two data column names in the expression editor.

In the "Projects" workflow we decided to use this implementation of the remaining values with the "Math Formula" nodes. That is, we used 3 "Math Formula" nodes, one after the other, to calculate the remaining values for 2009, the remaining values for 2008, and the remaining values for 2007 respectively.

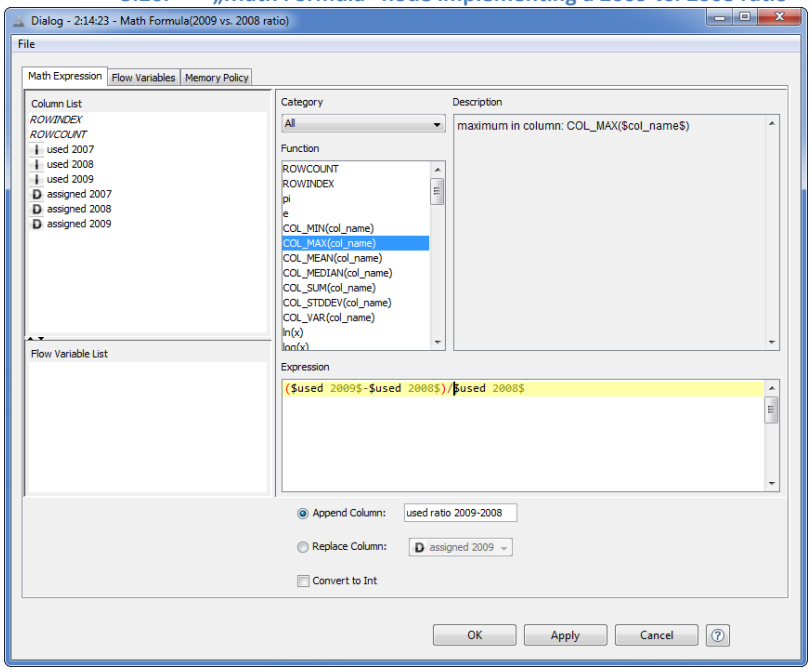
We also kept the implementation with the Java Snippet nodes for demonstration purposes. However, only the sequence of "Math Formula" nodes has been connected to the next node. We gave the "Math Formula" nodes the same names that we used for the Java Snippet nodes, to indicate that they are performing exactly the same task.

Let's have a look at another example (Fig. 5.26). The following ratio is implemented:

```
<used ratio 2009-2008> = (<used 2009> - <used 2008>)/<used 2008>
```

- Type a parenthesis "(" in the expression editor
- Double-click the column name "used 2009" in the "Column list" panel. Variable $used 2009$ appears in the expression editor.
- Type a minus sign "-" in the expression editor after $used 2009$.
- Double-click the column name "used 2008" in the "Column list" panel.
- Type a closed parenthesis and a division sign ")/" in the expression editor after $used 2008$.
- Double-click the column name "used 2008" again in the "Column list" panel
- Select "Append Column" and enter a name to append the new column to the data table.

5.26. „Math Formula" node implementing a 2009 vs. 2008 ratio



5.5. Marking Data for the Reporting Tool

We built the “Projects” workflow to produce a report. The KNIME reporting tool is a different application in comparison to the workflow editor, even though it is integrated into the KNIME platform. The idea is that the KNIME workflow prepares the data for the KNIME Report Designer, while the KNIME Report Designer displays this data in a graphical layout.

The two applications, the workflow editor and the reporting tool, need to communicate with each other; in particular the workflow needs to pass the data to the reporting tool. This communication between workflow and reporting tool happens via the “Data to Report” node.

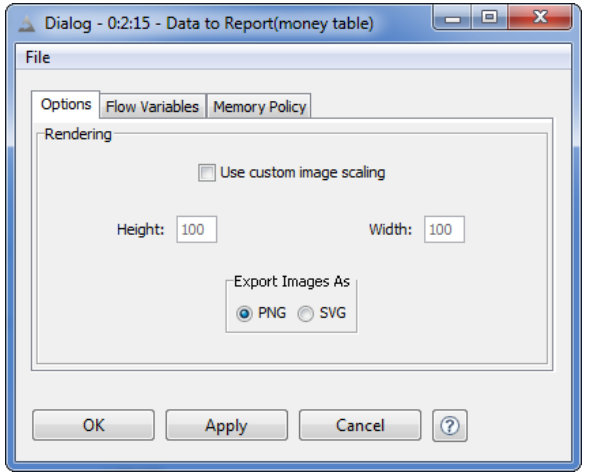
Data to Report

The “Data to Report” node can be found in the “Node Repository” panel in the `"Reporting"` category . The “Data to Report” node marks the KNIME data table at the input port to be recognized as a data set by the KNIME Report Designer.

When switching from the workflow editor to the reporting tool, all data tables marked by a “Data to Report” node are automatically imported as data sets. Each data set carries the name of the originating “Data to Report” node. Therefore the name of the “Data to Report” node is important! It has to be a meaningful name to facilitate the identification of the contents of the derived data set in the report environment.

Since the “Data to Report” node is only a marker for a data table, it does not need much configuration. The configuration window contains just a flag “use custom image scaling” to scale images in the data to a custom size. Default image size is the renderer size.

5.27. Configuration window of the "Data To Report" node.



We used two “Data to Report” nodes in our workflow. One is connected to the sequence of “Math Formula” nodes and exports the data for the tables in the report. The second one is connected to the “GroupBy” node named “money by project by year” and exports the data for the charts in the report. We named the first node “money table” and the second node “money chart”. When we switch to the reporting tool, we will find there two data sets called “money table” and “money chart” respectively. We therefore know immediately which data to use for the tables and which data for the charts.

In the category “Reporting” there is also the “Image to Report” node. The “Image to Report” node works similarly to the “Data to Report” node, it only applies specifically to images.

5.6. Cleaning Up the Final Workflow

The “Projects” workflow is now finished, however we can see that it is very crowded with nodes, especially if we want to keep all the “Java Snippet” nodes and the “Math Formula” nodes. To make the workflow more readable we can group all nodes that belong to the same task in a “Meta-node”. For example, we can create meta-node “remaining money” that groups together all the nodes for the remaining values calculations.

Create a Meta-node from scratch

A meta-node is a node that contains a sub-workflow of nodes. A meta-node does not perform a specific task; it is just a container of nodes.

To **create a “Meta-node”**:

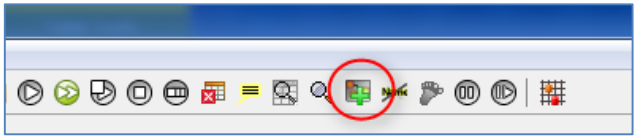
- In the Tool Bar click the “Meta-node” icon

OR

- In the Top Menu click “Node” and select “Open Meta Node Wizard”

You can choose between a number of pre-defined meta-node structures (1 input - 1 output, 2 inputs - 1 output, and so on). In addition, the “Customize” button enables you to adjust any selected meta-node structure.

5.28. "Meta-node" icon in the Tool Bar



To open a “Meta-node”:

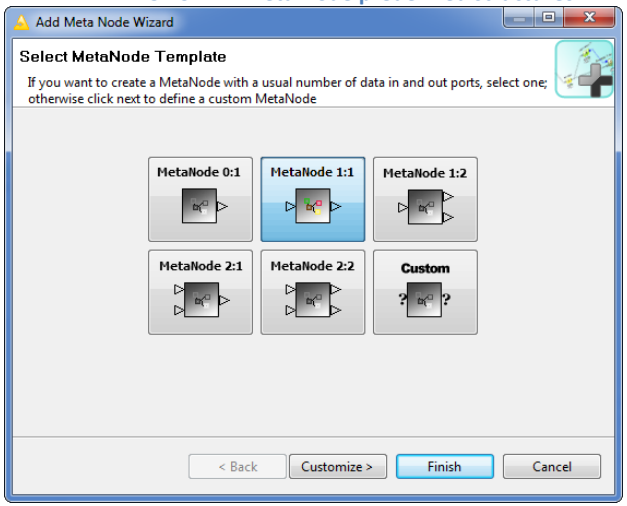
- Double-click the “Meta-node”

OR

- Right-click the “Meta-node” and select “Open sub-workflow editor”

- A new editor window opens for you to edit the associated sub-workflow contained in the “Meta-node”.

5.29. Meta-node predefined structures



To fill a “Meta-node” with nodes:

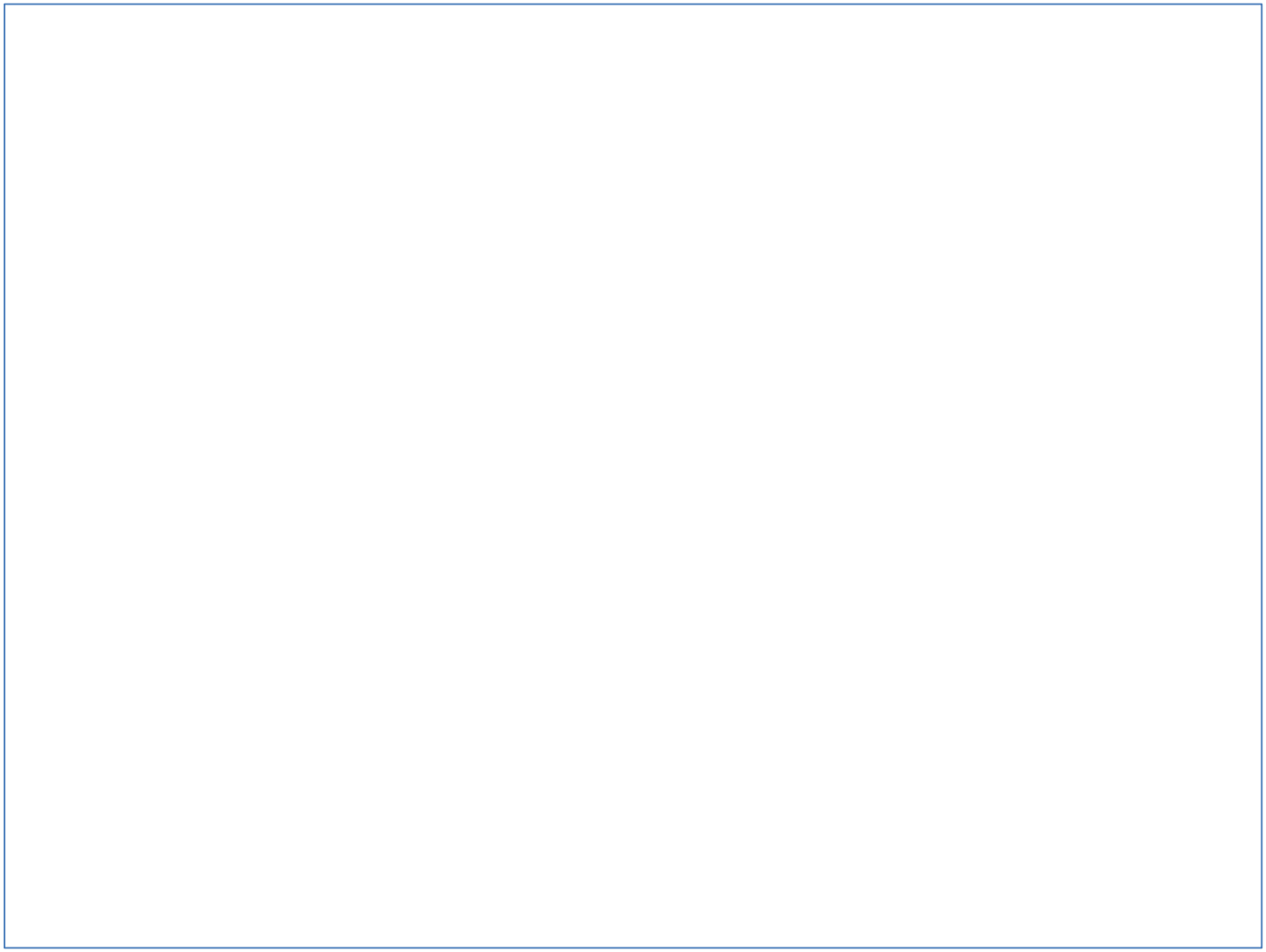
- Drag and drop nodes from the “Node Repository” panel as you would do with a normal workflow

OR

- Cut nodes that already exist in your workflow and paste them into the sub-workflow editor window

Note. The “Configure” option is disabled in a meta-node, as there is nothing to configure. All the other node commands, such as “Execute”, “Reset”, “Node name and description” etc…, are applied in the familiar manner, as for every other node. In particular, the “Execute” and “Reset” respectively run and reset all nodes inside the meta-node.

5.30. Workflow „Projects“



Collapse pre-existing nodes into a Meta-node

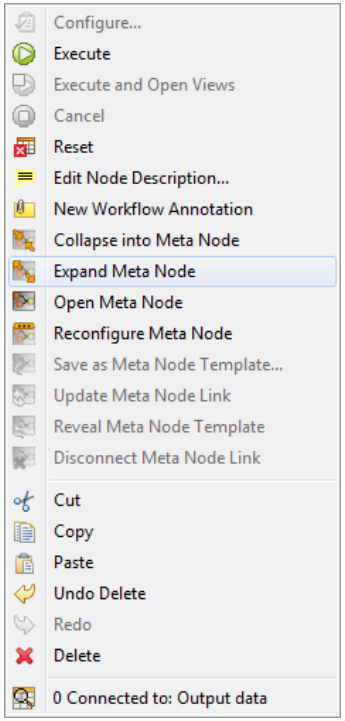
- In the workflow editor, select the nodes that will be part of the meta-nodes (to select multiple nodes in Windows use the Shift and Ctrl keys)
- Right-click any of the selected node
- Select “Collapse into Meta Node”
- A new meta-node is created with the sub-workflow of selected nodes

Option “Expand Meta Node” performs exactly the opposite task by extracting the sub-workflow from the meta node and inserting it into the parent workflow.

From the context menu of a meta-node it is also possible to change the structure of the meta-node itself.

- Right-click an existing meta-node
- Select “Reconfigure Meta Node”
- The Meta Node Wizard for custom meta-node (Fig. 5.32) then opens allowing to add / move/ remove input and output ports.

5.31. "Collapse into Meta Node" option in the context menu of selected nodes

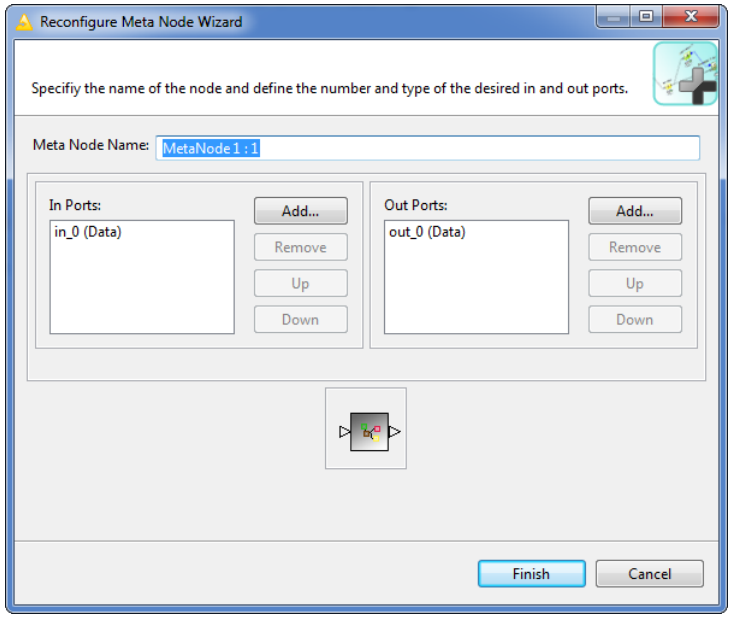


We created a new “Meta-node”, and named it “remaining money”. We connected its input port to the output port of the “Joiner” node and its output port to the input port of the “Data to Report” node named “money table”. We then cut the “net 2009”, “net 2008”, and “net 2007” nodes (both “Java Snippet” nodes and “Math Formula” nodes) from the main workflow and we pasted them into the meta-node’s sub-workflow. The output data table of the “remaining money” meta-node now contains the same results as the previous output data table of the last “Java Snippet” or the last “Math Formula” node of the “remaining money” sequence.

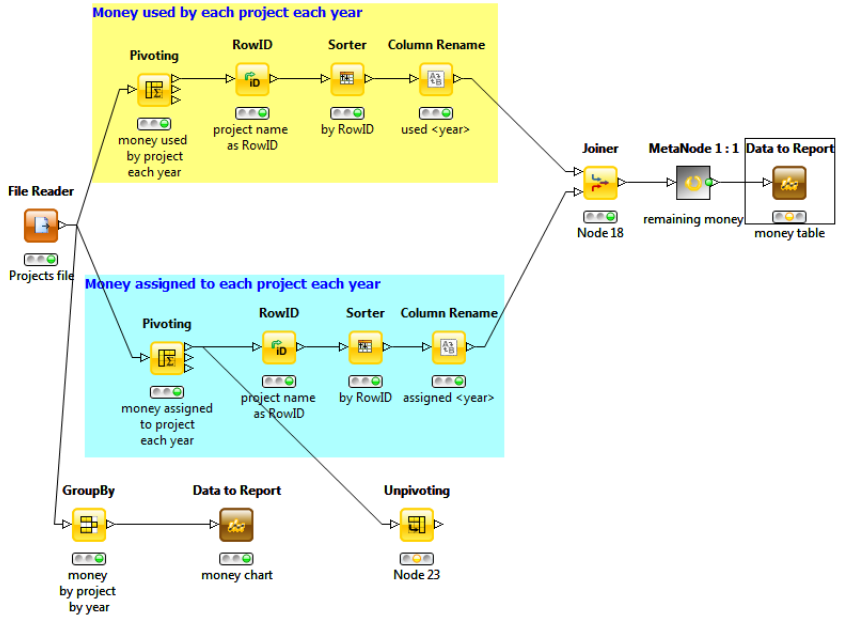
There are a number of pre-packaged meta-nodes in the KNIME “Node Repository” panel in category `"Meta"`. The “Meta” category contains useful meta-node implementations for loops and features elimination.

Figure 5.33 shows the final “Projects” workflow.

5.32. The Meta Node Wizard for custom meta-nodes



5.33. The "Projects" workflow



5.7. Exercises

Exercise 1

Use the input data **adult.data** to do the following:

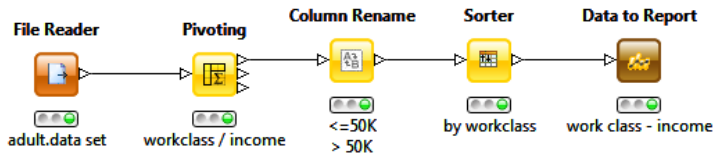
- Calculate the total number of people with income > 50K and the total number of people with income <= 50K for each work class
- Sort rows on the “work class” column in alphabetical order
- Create a data table structured as follows:

Work class	Nr of people with Income > 50K	Nr of people with Income <= 50K
Work class 1		
…		
Work class n		

- Mark this data set for reporting

Solution to Exercise 1

5.34. Exercise 1: The workflow



1. Read the data.
2. Use a “Pivoting” node to build the data table in the requested format. The “workclass” column should be the group column and the “Income” column should be the pivot column.
3. Optionally rename the column headers to make the table easier to read.
4. Attach a “Data to Report” node to be able to export the data into a report later on

Exercise 2

Extend Exercise 1

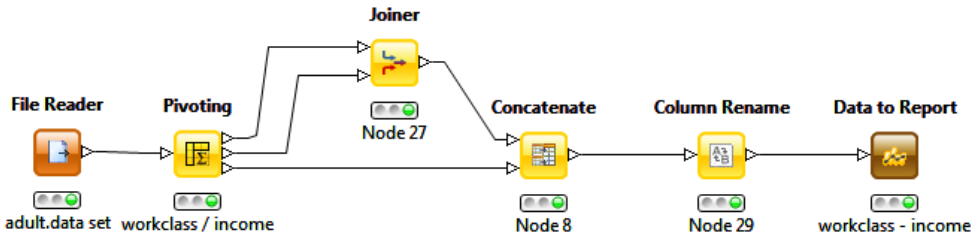
- Calculate the total number of people with income > 50K and with income <= 50K
- Calculate the total number of people for each work class
- Calculate the total number of people
- Extend the data table produced for exercise 1 as follows:

Work class	Nr of people with Income > 50K	Nr of people with Income <= 50K	Nr of people
Work class 1			
…			
total	Sum(nr of people with Income > 50K)	Sum(nr of people with Income <= 50K)	Sum(nr of people)

Solution to Exercise 2

1. To calculate the number of people for each “workclass” and each income class, we use the “Pivoting” node built in Exercise 1. The “Pivoting” node has three outputs: the pivot table, the totals by row, and the totals by column. Remember to enable “Append overall totals” in the “Pivots” tab.
2. We then join on the “workclass” values the pivot table with the totals by row using a “Joiner” node.
3. We then concatenate the data table resulting from the “Joiner” node with the totals by column of the “Pivoting” node.
4. Finally attach a “Data to Report” node and name it “workclass-income”

5.35. Exercise 2: The workflow



Exercise 3

Read the csv file SoccerWorldCup2006.txt from the “Download Zone”. This file describes the results of soccer games during the soccer world cup 2006 (www.fifa.com). The second semifinal game for the third and the fourth placement is not reported.

For each team calculate:

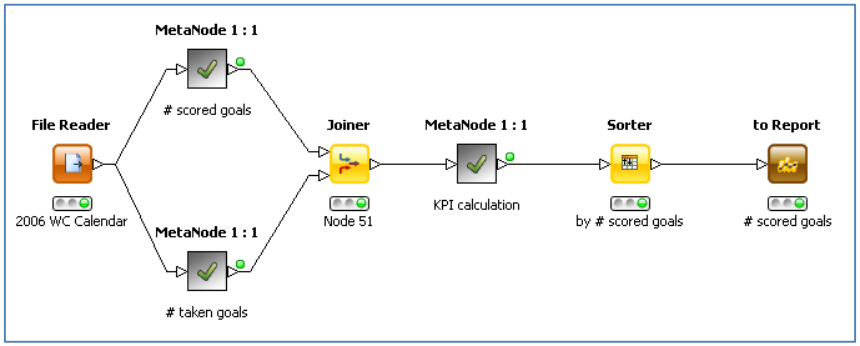
- The total number of played games
- The total number of scored goals
- The total number of taken goals
- The average number of scored goal per game
- The average number of taken goal per game
- A fit measure as: `(total number of scored goals – total number of taken goals)/number of played games`

Document each step with the appropriate node’s name and description.

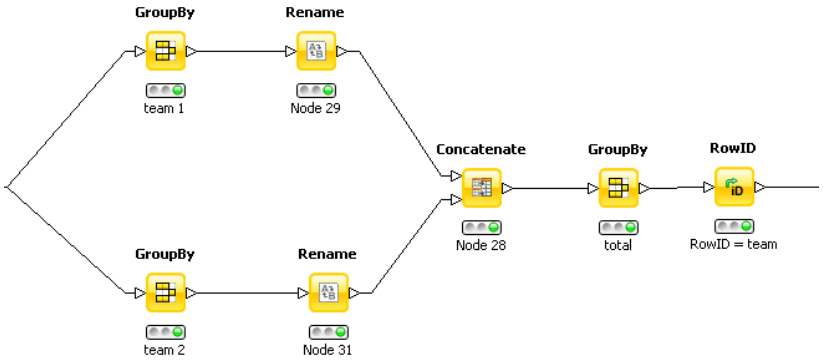
Make the workflow readable by using meta-nodes.

Solution to Exercise 3

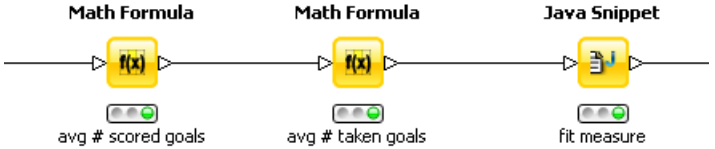
5.36. Exercise 3: workflow



5.37. Meta-node "# scored goals"/"# taken goals"



5.38. Meta-node “KPI Calculation”



In the “# scored goals” meta-node, first we sum team 1’s scores over all team 1, then the sum of team 2’s score over all team 2’s, and finally we sum the total scores of team 1 and team 2 when team 1 = team 2.

Meta-node “# taken goals” has the same structure as Meta-node “# scored goals”. The only difference lies in the aggregation variable of the first two “GroupBy nodes. In the meta-node “# scored goals” the first “GroupBy” node sums the “score of team 1” for all “team 1” values and the second “GroupBy” node sums the “score of team 2” for all “team 2” values. In meta-node “# taken goals” the first “GroupBy “node sums the “score of team 2” for all “team 1” values and the second “GroupBy” node sums the “score of team 1” for all “team 2” values.

In the “KPI calculation” meta-node we used 2 “Math Formula” nodes and one “Java Snippet” node. It could have been any other combination of “Java Snippet” and “Math Formula” nodes.

Chapter 6. My First Report

6.1. Switching from KNIME to BIRT and back

In the previous chapter we have shown how to build a workflow to generate data ready to use in a report. In this chapter we will show how to use the KNIME Report Designer to read the data produced by the workflow, to shape the report layout and produce the final document.

The KNIME Reporting tool is based on BIRT (Business Intelligence Reporting Tool), which is open source software for reporting. BIRT and KNIME are two different tools using the same environment with customized properties. In KNIME we develop workflows for data manipulation and modeling. In BIRT we create and shape the report to represent the workflows' data.

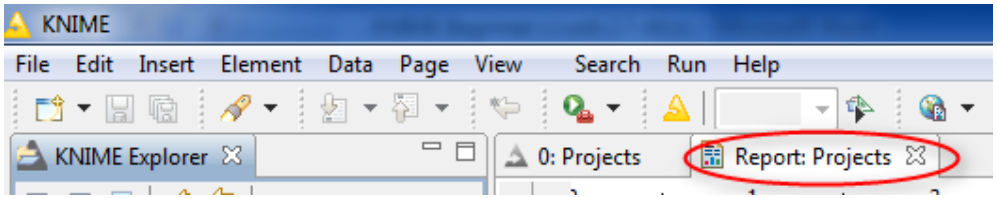
Only one report is associated to one workflow and vice versa. It is not possible to associate more than one report to one workflow. When we move into the BIRT environment, we open the report associated with the workflow. If it is the first time that we open the report associated with the workflow, the report will be empty. From a KNIME workflow, you can switch to the BIRT environment and open the associated report by:

- Opening the workflow in the "KNIME Explorer" panel
- Clicking the "Report" icon in the tool bar.

6.1. The Report icon in the Tool

The BIRT report editor then opens the report associated with the selected workflow. The report editor creates a new tab in the KNIME Editor window.

6.2. The new tab in the KNIME Editor for the selected report



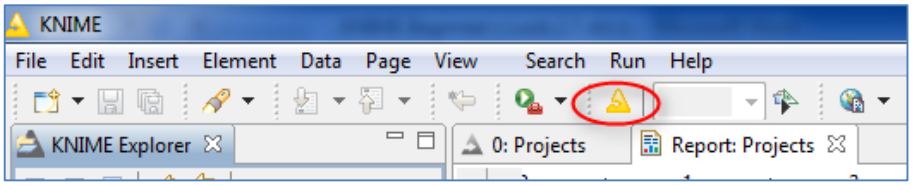
To go back from the report to the workflow editor, you can:

- Select the workflow's tab

or

- Click the KNIME icon in the tool bar

6.3. The KNIME icon in the Tool Bar



This will take you back to the more familiar KNIME environment.

Let's continue our work on the "Projects" workflow created in the previous chapter. We have the data, now we want to put together an appealing report to show it.

To open the report, select the "Projects" workflow in the "KNIME Explorer" panel; next right-click the workflow and select "Edit Report". This takes you to the BIRT environment, to a default empty report.

6.2. The BIRT Environment

BIRT is developed as an Eclipse Plug-In as KNIME is. This means that they both inherit a few properties and tools from the Eclipse platform. As a consequence, the BIRT report editor and the KNIME workflow editor are very similar, which makes our learning process easier for the reporting tool. In this section we provide a quick overview of the BIRT report editor. For more information on the BIRT software, the book listed in [2] gives a detailed overview of BIRT potentials.

Let's have a look at the different windows in the BIRT environment with an empty report.

The "**KNIME Explorer**" panel is still in the top left corner and it still contains the list of available KNIME workflows.

Under the "KNIME Explorer" panel, we find the "**Data Set View**" panel. This panel contains all data sets that are available for the report.

Under the "Data Set View" panel, we find the list of all available **report items** to create our report, like Table, Label, Chart, and so on.

In the center, as for the KNIME workflow editor, we find the **report editor**. Like in KNIME, where we built workflows by "dragging and dropping" the nodes into the workflow editor, here we can compose the report by "dragging and dropping" the report items into the report editor.

Finally in the center bottom of the window there are a few tabs, of which three are interesting for our work: Layout, Master Page, and Preview.

Layout is the page editor, where the single report page is processed.

Master Page, as in PowerPoint Master Page, defines a template for every page of the report. This is where the page header and footer are designed.

Finally, **Preview** allows the user to get a rough idea of what the report will look like. It is useful to see an intermediate view of the work without having to generate the final document for every little change.

6.3. Master Page

We now have an empty report to fill with tables and charts. First of all let's define its basic properties, such as paper size, borders, running headers, footers and so on. Basically, we want to define its Master Page.

Right below the report editor, there are three tabs: “Layout”, “Master Page”, and “Preview”. Let’s select tab “Master Page”.

Now the report editor in the center has become the Master Page editor and, below the three tabs, you can see the Master Page’s Properties Editor. There are 5 property groups: “General”, “Margins”, “Header/Footer”, “Comments”, and “Advanced”.

We would like to prepare a report to be exported into slides in PowerPoint format. We also want to have a running title with a logo on all the slides.

Usually PowerPoint slides have a landscape orientation. To change the paper orientation, we go to the “Orientation” field under the property “General”. We change it to “Landscape”.

To create a running title, we should change the header in the Master Page. The property “Header/Footer” offers only check boxes about showing or not showing the header and the footer. In order to actually change the header and the footer, we need to work in the Master Page editor itself. In the top part of the Master Page editor there is a dashed rectangle. This is the header editor.

To insert a logo in the header of each slide go to the Master Page editor:

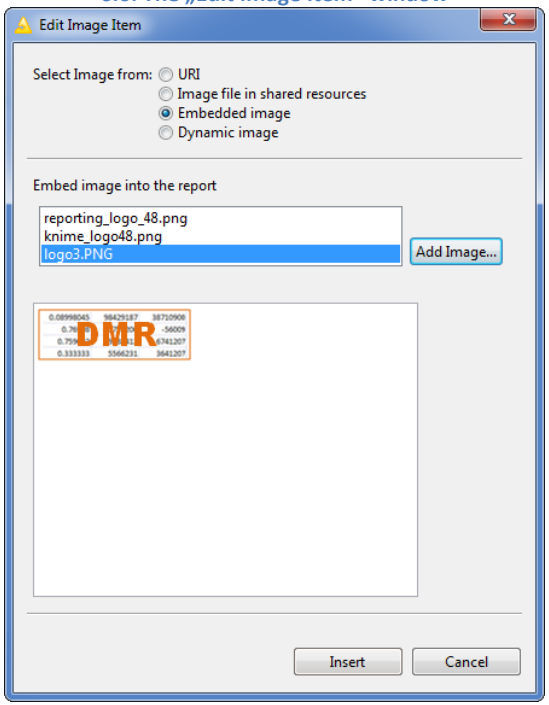
- Right-click the header editor
- Select “Insert”
- Select “Image”
- In the “Edit Image Item” window upload your image, for example as an embedded file

The logo image will appear in the top left corner of the header editor.

Instead of an image you can insert a “Label” in the header editor to have a running title in your slides. You can also combine both, a running title and a logo, in the header editor. However, you can only combine more report items side by side by using the “Grid” report item.

Now click the “Preview” tab. You should get a rough idea of what your report with the new header will look like. For the moment it is just the logo in the top left corner and the footer with the KNIME advertisement.

6.6. The „Edit Image Item“ window



6.4. Data Sets

Data sets contain the data available for the report. Each report is linked to one and only one workflow. In the integration of BIRT inside KNIME, data sets are automatically imported from the data tables marked by a “Data to Report” node in the underlying workflow. In the integrated version there is no other way to generate data sets in the reporting environment.

Let’s have a look at the data sets available for the report of workflow “Projects”.

In the “Data Set View” panel you should see two data sets, named “money chart” and “money table”. These were the names of the two “Data to Report” nodes in the “Projects” workflow. Indeed, when switching from the KNIME workflow editor to the BIRT report editor, the data of the “Data to Report” nodes are automatically exported as data sets into the report environment. For this reason, it is important to give meaningful names to the “Data to Report” nodes, so that when switching into the report editor we are not confused by data sets with obscure names.

If you cannot remember which “Data to Report” node the data set has been generated from or to check that the data set got exported correctly, you might need to preview the data in the data set. In order to do that:

- Double-click the data set

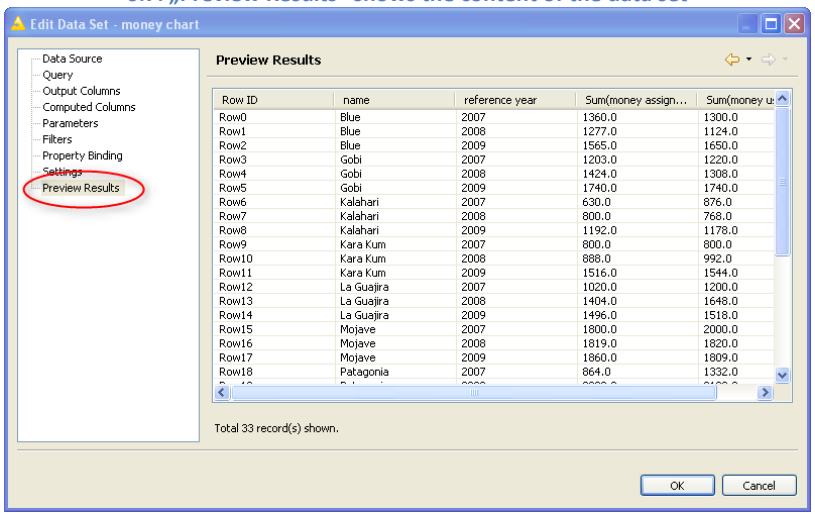
OR

- Right-click the data set and select “Edit”

then

- In the “Edit Data Set” window select “Preview Results”

6.7. „Preview Results“ shows the content of the data set



6.5. Title

Let’s now start assembling the report. Click the “Layout” tab to move away from the Master Page editor and back to the Report editor. What we see now is an empty page. First of all, we would like to have a title for our report, something like “Project Report: Money Flow” for example. We are going to place tables, charts, and more explicative labels under the main title.

To build a title:

- Drag and drop the “Label” report item from the “Report Items” panel in the bottom left corner into the Report editor
- Double click the label and enter the title ”Project Report: Money Flow”
- Select the whole label by clicking its external contour
- In the “Property” editor under the Report editor, go to the tab called “General” and select the properties for your title: font, font size, font style, font color, background color, and so on.

We chose font “Cambria”, color “green”, size “24 points”, style “bold”, and adjustment “centered”.

Note. The font size settings consist of 2 parameters: the number and the measure unit (%, cm, in, points, etc…). Be sure to set both of them consistently! If you set the number to 24 and the unit to “%” you will not see your title label anymore and will wonder what happened to it.

6.8. Drag and drop a “Label” item into the Report Editor to create the report title

6.6. Grid

I am sure you have noticed that the title label has been automatically placed at the top of the page and that it spans the complete width of the page. You cannot move it around to place it anywhere else nor shrink it to occupy only a part of the page width. This automatic adjustment (full page width and first available spot in the page from the top) will affect all report items that are dragged from the "Report Items" panel and dropped directly into the Report editor. For the title item this is not so bad, since the title usually spans the whole page width and is placed at the page top. It is however undesirable for most other report items.

In our report we would like to have three tables: two tables at the top describing the amount of money assigned and used for each project each year, and one table in the middle of the page under the two previous tables to show the remaining money. It would also be nice if all tables had the same size; i.e. something less than the half of the page width. Under the tables we would like to place two bar charts side by side to show respectively how the money has been assigned and used. In order to have the freedom to place report items anywhere in the report page and to give them an arbitrary size, we need to place them inside a "Grid".

A "Grid" is a report item, something like a table that creates cells in the report page with customizable location and size to contain other report items.

For our report, we need:

- one row with two cells: one for the assigned money table and one for the used money table
- one row with only one cell for the remaining money table
- one row with two cells again for the 2 bar charts

We therefore want to create a "Grid" with 3 rows and 2 columns and merge the two cells of the second row into one cell only.

To create the "Grid":

- Drag and drop the "Grid" report item from the "Report Items List" panel into the Report editor under the title label
- Enter 2 for the number of columns and accept 3 for the number of rows
- Select both cells in the second row by clicking the external left border of the row
- Right-click the two-cells selection
- Select the "Merge cells" option

Note. Sometimes I use over-detailed grids. That means I define grids with more columns and rows than necessary. This gives me more freedom in adjusting distances between report items and other margins.

6.7. Tables

To create a table we can follow the standard procedure:

- Drag and drop the “Table” report item into the report editor
- Bind the “Table” to a data set

- Bind each data cell to a data set field

OR we can:

- Drag and drop the data set into the report editor
- Remove any undesired columns

The second method is easier especially for big tables.

6.10. Drag and drop a data set from the "Data set View" panel to produce a table with as many columns as many data set's fields

In the report layout a table is composed of three rows:

- a header row
- a data cell row
- a footer row

The header row and the footer row contain only labels or other static report items and appear in the final report only once at the beginning and end of the table respectively. The data cell row contains the data set fields. In the real report, the data cell row multiplies into as many rows as there are in the data set.

After dragging and dropping the Data Set into the report editor, we see a table with as many columns as there are fields in the data set. The column headers are automatically set as labels with the data set field's name. The footer row is empty. The data cell row contains the data set fields. Let's now adjust the look of the table.

Remove unwanted columns

- Select the whole table. If you hover over the left bottom corner of the table with the mouse, a small gray rectangle with the word "Table" appears. To select the whole table, click that rectangle.
- Select the unwanted column. To select a whole column click the gray rectangle above the column's header.
- Right-click the top of the unwanted column
- Select "Delete"

Change the column header

The header of each column is an editable label

- Double click the header label
- Change the text

Change column position

- Select the whole table
- Right-click the top of the column (the gray rectangle) that you want to move
- Select "Cut"
- Select the column to be positioned on the left; do this right-clicking the gray rectangle at the top of the column
- Select "Insert Copied Column"

Change font properties

- As for "Labels", in the "Properties" window ("General" tab) you can change font, font size, alignment, style, etc…

Format number

- Select a cell containing a number
- In the “Properties” editor, select the “Format Number” tab
- Choose the format for the number in your cell

Define width and height

- Select a row or a column
- In the “Properties” window, go to the “General” tab and change the height and width

Set borders

- Select the item that needs borders (full table, row, or single cell).
- In the “Properties” editor, select the “Border” tab
- Choose the desired border

Note. The property “Border” is not available for columns.

Set table size

- Select the whole table
- In the “Properties” window, select the “General” tab
- Choose the desired width and height

Note. For the font, cell, and table size, the height and width can be expressed in different measure units. Verify that the unit you are using is a meaningful one. BIRT performs some kind of automatic adjustment on the width and height of the cells. You must define a suitable height and width for the full table first for the height and width of the single cells to become effective.

We dragged and dropped the “money table” data set into each one of the two cells in the first row and into the only cell in the second row of the “Grid”. The table on the left of the first row will show the assigned money. We then deleted all “*used*” and “*remain*” columns. The table on the right of the first row will show the used money. We then deleted all “*assigned*” and “*remain*” columns. The table in the second row will show the remaining values. Here we deleted all “*used*” and “*assigned*” columns.

In each table, the “RowID” column contains the project name. We therefore changed the header label to “Name”. The data and header cell for the “Name” column were left aligned while the last 3 cells were all right aligned. The tables had a green border running around it and also a green border between the header row and the data row.

The size of the first two tables was set to 80% (= 80% of the grid cell) and the size of the third table, which in a grid cell is double the size of the previous two, was set to 40% (= 40% of the grid cell). The alignment property of the three grid cells was set to “Center”.

In the first table, we then set the font to “Cambria” and font size to “10 points” in both header and data cells. The header’s font style was also set to “bold” and the color to “green”. Finally, the data cells containing numbers were formatted with “Format Number” set to “Fixed” with 2 decimal places and 1000s separator. All these operations should be repeated for the second and the third table as well.

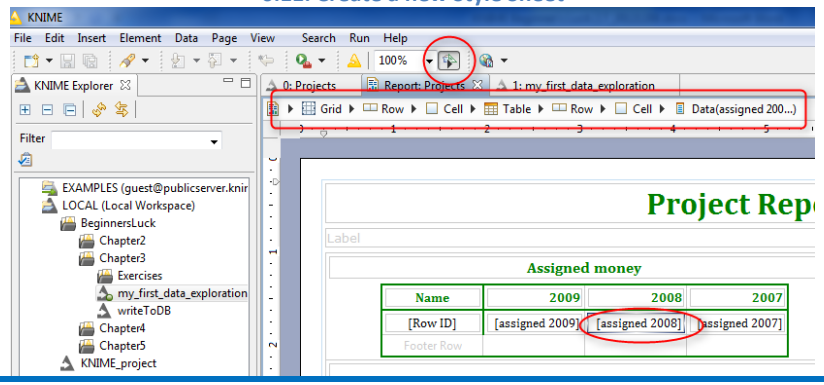
Toggle Breadcrumb

In the top bar you can find the “Toggle breadcrumb” button.

This button displays the hierarchy of a report item over the layout, for example the hierarchy of the “assigned 2008”data cell as:

Grid -> Row -> Cell -> Table -> Row -> Cell -> <data set field name>

6.11. Create a new Style Sheet



6.8. Style Sheets

Sometimes it can be tedious to format all single elements of a report item, especially if many of these report items have to be formatted with the same style. For example, in the previous section we were supposed to format all data cells and header cells of three tables in the same way. To avoid having to repeat such tedious operations, we can use the style sheets.

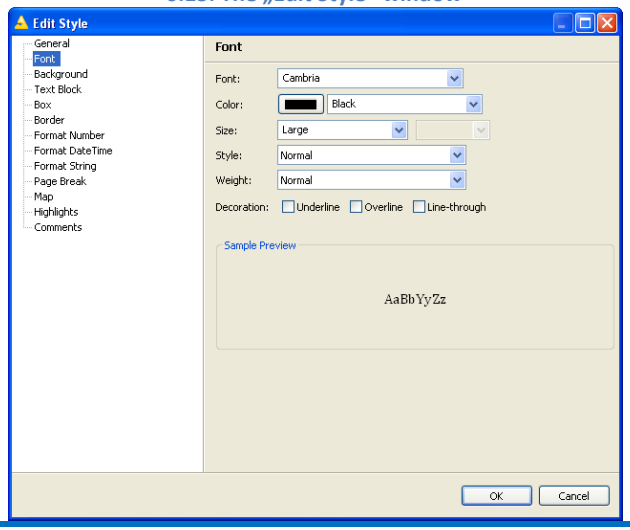
Style sheets are widely used in web programming to share style specifications across the many elements of web pages. Similarly, the KNIME reporting tool supports style sheets which can be used to apply style attributes to multiple report items.

Create a new Style Sheet

- Right-click anywhere on the report editor
- Select “Style”
- Select “New Style”

The “New Style” window opens.

6.13. The „Edit Style“ window



In the “New Style” window, you need to define:

- The name of the style sheet in the “General” tab
- The font properties in the “Font” tab
- The number properties in the “Format Number” tab
- And so on with more properties in other tabs

Taking the tables in the previous section as an example, it is easy to see that there are two groups of cells for each table:

- Header cells with font “Cambria”, font size “10 points”, font style “bold”, and font color “green”
- Data cells with font “Cambria”, font size “10 points”, and number format with 2 decimal places and 1000s separator

We then built two style sheets, one for the data cells and one for the header cells with the properties listed above. We chose “large” font size for both Style Sheets, named them “data cell” and “header cell” and applied them to each header cell and each data cell of the three tables.

Note. Not all font sizes are available in the Style Sheet editor as in the Property Editor. Only a few pre-defined font sizes can be used in a Style Sheet.

6.14. Apply a Style Sheet to a report item for example a data cell

Apply a Style Sheet

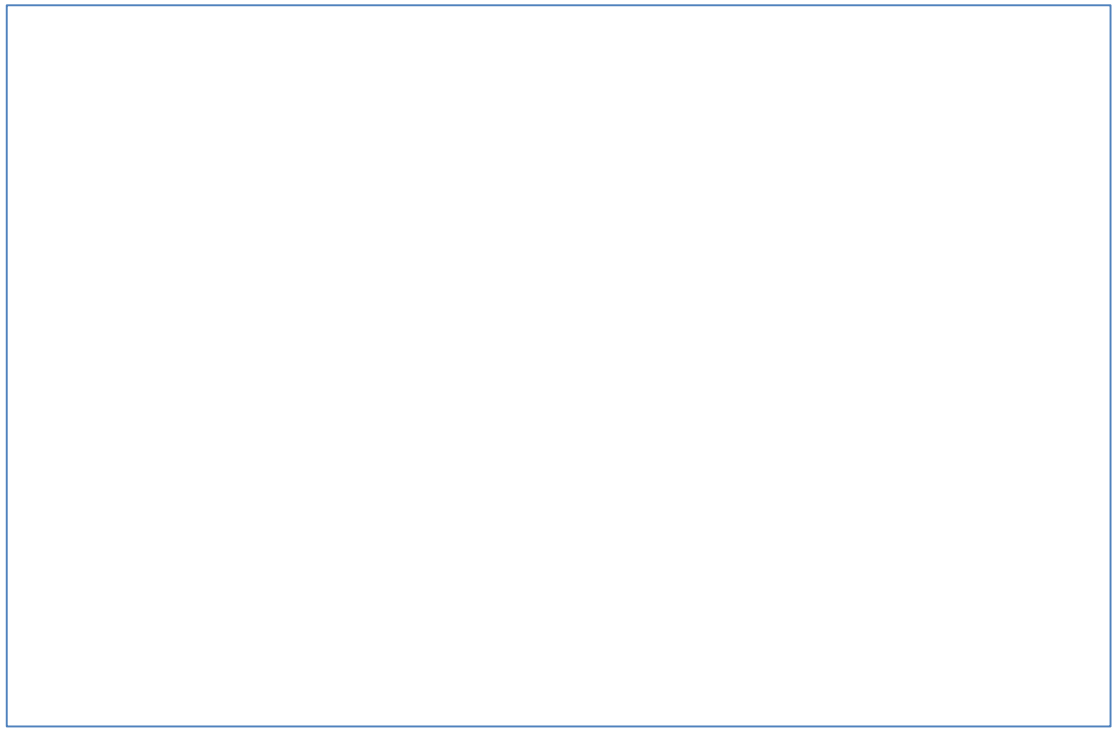
- Right-click the report item (table cell, label, etc…)
- Select “Style”
- Select “Apply Style”
- Select the name of the Style Sheet you want to apply

Let’s now run a “Preview” (“Preview” tab above the Property editor frame) to have a rough idea of what the report will look like. Probably the “large” font size we have chosen for the data cells and header cells will be too big for the tables to nicely fit into one page. We can easily reduce the font size

by setting it to “small” in one or both Style Sheets. This will automatically apply to all those table cells that have been formatted by these Style Sheets. This is one of the big advantages of using Style Sheets.

Let’s put a label on top of each table to say what the table is representing: “assigned money”, “used money”, and “remaining money”. We can then change the column headers from “<assigned/used/remain> <year>” to just “<year>”, for example “assigned 2009” to just “2009” and so on. Let’s also add a few empty labels after each table to make the report layout more spacious. If we run a Preview now, the report will look similar to the one shown in figure 6.15.

6.15. Report Preview after creating and formatting the three tables



6.9. Maps

Sometimes, we might want to map numeric values to descriptive values. For example in a financial report, we can map one column with numeric values as:

Values < 0	to	“negative”
Values = 0	to	“zero”
Values > 0	to	“positive”

The mapping functionality is found in the “Maps” tab in the “Properties” editor of table report items (Fig. 6.16); that is cells, rows, columns, and even the whole table.

Select the data cell, row, column, or table to which you want to apply your mapping

Select the “Maps” tab in the “Properties” editor

Click the “Add” button to add a new mapping rule

The “New Map Rule” editor opens.

Build your condition in the “Map Rule Editor”, for example:

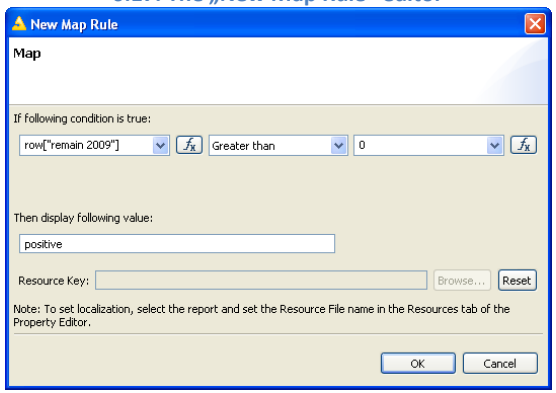
row[”remain 2009”] Greater than 0 -> “positive”

Click “OK”

Click the “Preview” tab to see the new mapped values.

6.16. The Tab „Maps“ in the table Properties Editor defines text mapping for a table item

6.17. The „New Map Rule“ editor



6.10. Highlights

The "Highlights" property works similarly to the "Maps" property, only that it affects the cell and row layout rather than cell text content.

The "Highlights" property is located in the "Highlights" tab in the Property editor of the "Table" report items: cells, rows, columns, and the whole table.

For example, we would like to mark all the cells with a "remain 2009" value smaller than 0 in red.

- Select the data cell [remain 2009] (or another cell, a row, or a column where the highlighting should occur)
- Click the "Highlights" tab in the Property editor
- Click the "Add" button

The "New Highlight" editor opens.

In the "Condition" section:

- Enter the rule for the highlight, for example:

  Row[net 2009] smaller than 0

  To build the rule you can also use the "Expression Editor" which is explained later in this chapter.

- In the "Format" section, enter the formatting you want to be applied, when the condition is true.

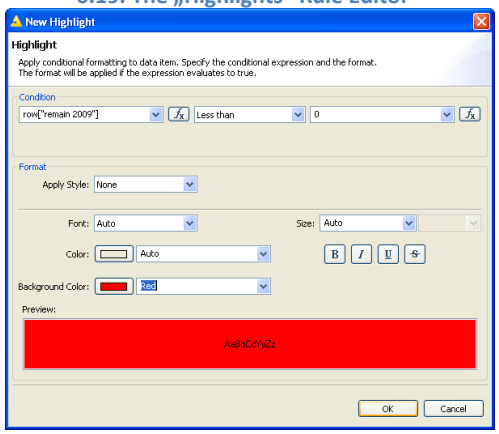
  Do this by clicking the button next to "Background Color" and then select red in the color dialog.

- Click "OK"

After closing the Highlight dialog, click the "Preview" tab to see the new red highlighted cells.

6.18. The „Highlights" tab in The Properties Editor defines conditional properties for a table item

6.19. The „Highlights" Rule Editor"



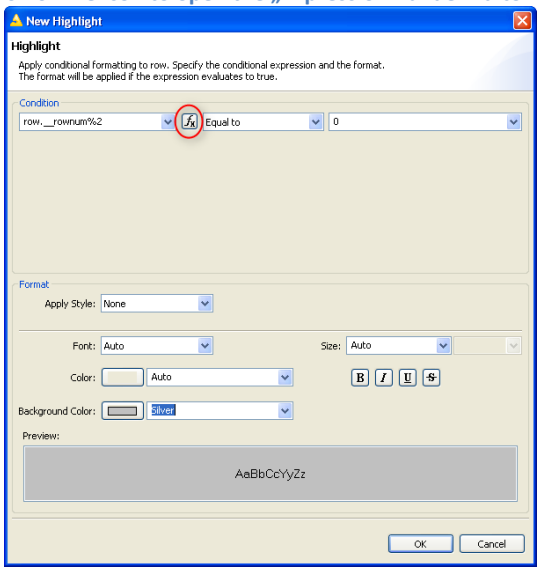
The zebra style

The zebra style is very popular for tables in reports. This is where the table's rows have alternating colors. To produce a zebra style table, you need to add the following condition in the "New Highlight" editor:

- Select the whole data row in the table, by selecting the gray rectangle on the left of the table row
- Select the "Highlights" tab in the Property editor
- Select the "Expression Builder" icon. This is the icon with "fx" close to the "Condition" input box
- In the "Expression Builder" dialog, select "Available Column Bindings" and then "Table"
- Double-click "RowNum" in the right column of the "Expression Builder" table
- row.__rownum appears in the "Expression Builder Editor"
- Write "row.__rownum % 2" in the "Expression Builder " dialog and click "OK"
- Select "Equal to" and enter "0" in the "New Highlights" editor
- In the "Format" section, select the background color "gray" or "silver" in the consequent field of the rule
- Click "OK"

Click "Preview" to see the zebra style table.

6.20. The icon to open the „Expression Builder Editor



6.21. The zebra style table

Net money

Name	2009	2008	2007
Blue	-85.00	153.00	60.00
Gobi	0.00	116.00	-17.00
Kalahari	14.00	32.00	-246.00
Kara Kum	-28.00	-104.00	0.00
La Guajira	-22.00	-244.00	-180.00
Mojave	51.00	-1.00	-200.00
Patagonia	-5.00	-41.00	-468.00
Sahara	-175.00	-3.00	-99.00
Sechura	-60.00	-147.00	-400.00
Tanami	-15.00	0.00	-138.00
White	73.00	139.00	0.00

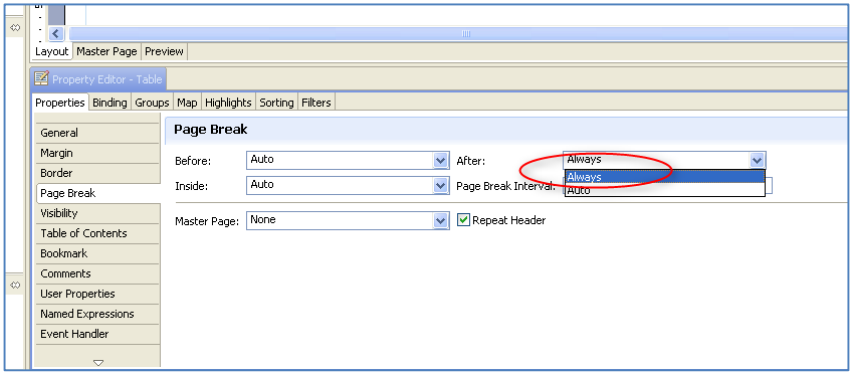
6.11. Page Break

We want to export the final report to Powerpoint. This first part of our report fits nicely into a Powerpoint slide. A page break at this point would be very useful to prevent undesired page format effects in the final document.

To insert a page break after a report item:

- Select the report item
- In the Property editor, select the “Page Break” tab
- Set your page break by changing the page break option from “Auto” to “Always”

6.22. „Page Break” tab in the Property editor



In the example workflow, the page break was set after the “remaining money” table.

6.12. Charts

The final part of this report consists of two charts to be placed side by side in the last row of the grid. One chart shows assigned money over the years and the other chart shows used money over the years. The two charts should have an identical look.

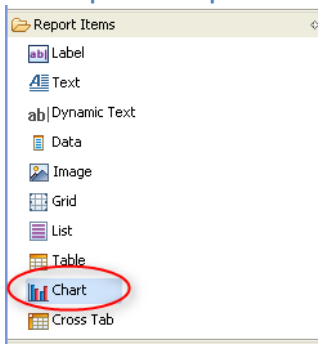
To create a chart, drag and drop the “Chart” report item from the “Report Item List” into the report editor. After the chart has been dropped, the “Chart Wizard" opens to guide you in setting the right properties for the chart.

The “Chart Wizard” covers three main steps for all types of charts:

- Select the Chart Type
- Select the Data
- Format the Chart

The “Chart Wizard” can be reopened at any moment by double-clicking the chart.

6.23. „Chart” report item in the “Report Items” panel



Select Chart Type

The first step of the “Chart Wizard” consists of selecting the chart type.

There are many chart types available and each chart type has a number of chart subtypes.

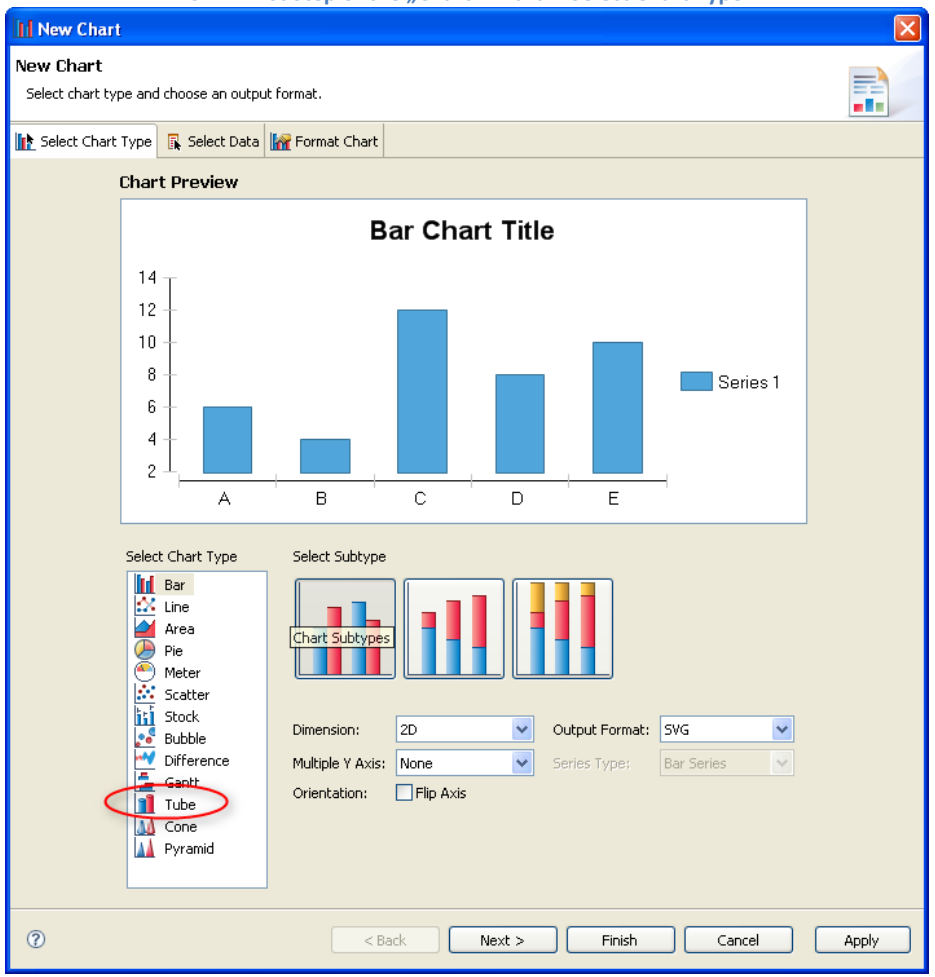
In addition, each chart can be rendered in 2D, in 2D with depth or in full 3D.

Flip Axis will change the orientation of the chart. The X-axis will then be vertical and Y-axis horizontal.

- Select your chart type
- Click “Next” to proceed to the next chart wizard’s step.

We chose a “Tube” chart type in a simple 2D dimension.

6.24. First Step of the „Chart Wizard”: Select Chart Type

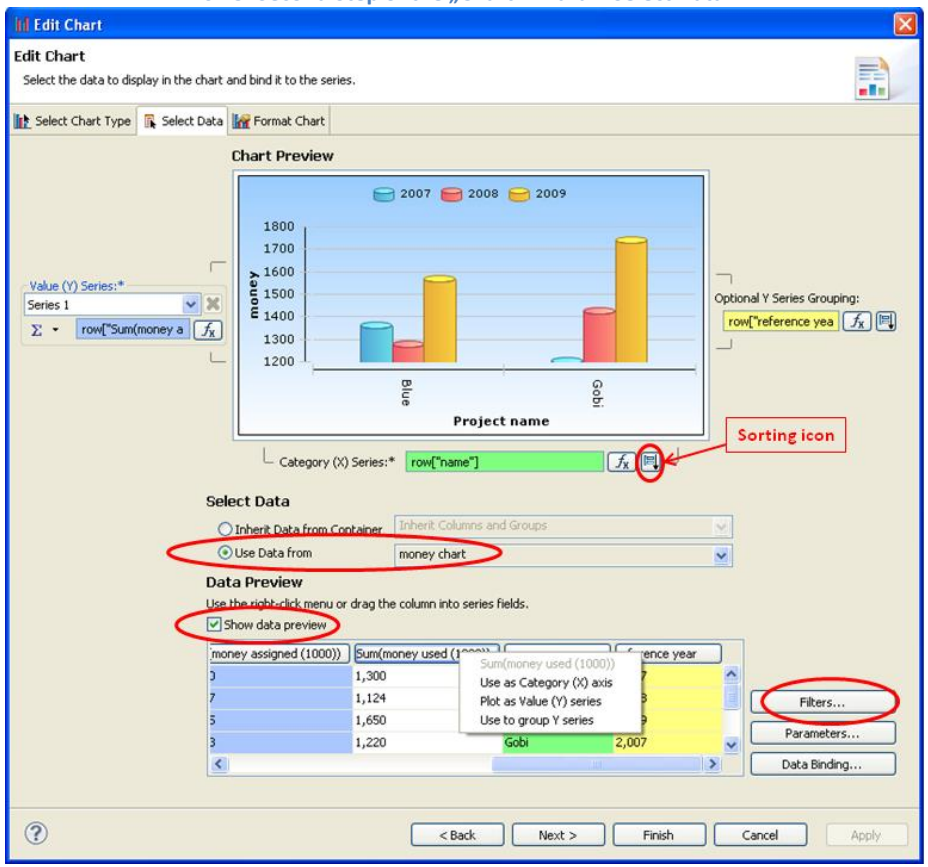


Select Data

To connect the chart to a Data Set is the second step.

- Bind the chart with a Data Set with option “Use Data from”.
- In the data preview table select the column data to be on the X-axis or on the Y-axis or to work as group data. Right-click on the column header and select one of those options:
  - Use as Category (X) axis
  - Plot as Value (Y) series
  - Use to group Y-series
- If you need additional Y-series, select “<New Series …>” in the menu called “Value (Y) Series”.
- Category data are sorted on the X-axis in descending order by default. If you do not want any sorting, click the sorting icon (the one with the down arrow on the side of the “Category (X) Series:” text box) and disable “Grouping”.
- Sometimes not all data rows from the data set need to be shown in a chart. To filter out rows from the data set, click the “Filters” button on the bottom right and add rules to include or exclude rows from the data set (see below).
- Click “Next” to move to the next wizard’s step.

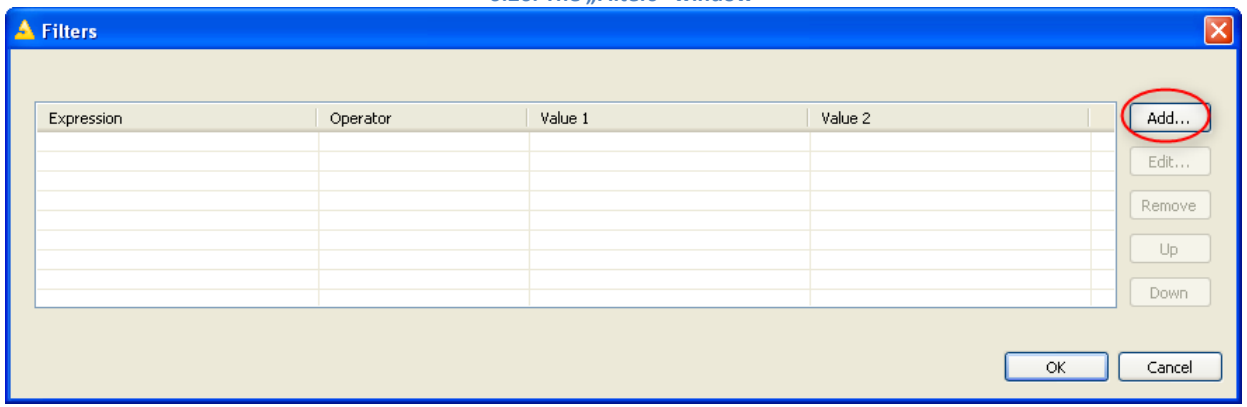
6.25. Second Step of the „Chart Wizard”: Select Data



To filter rows in the data set:

- Click the “Filters” button
- In the “Filters” window, click the “Add” button

6.26. The „Filters“ window

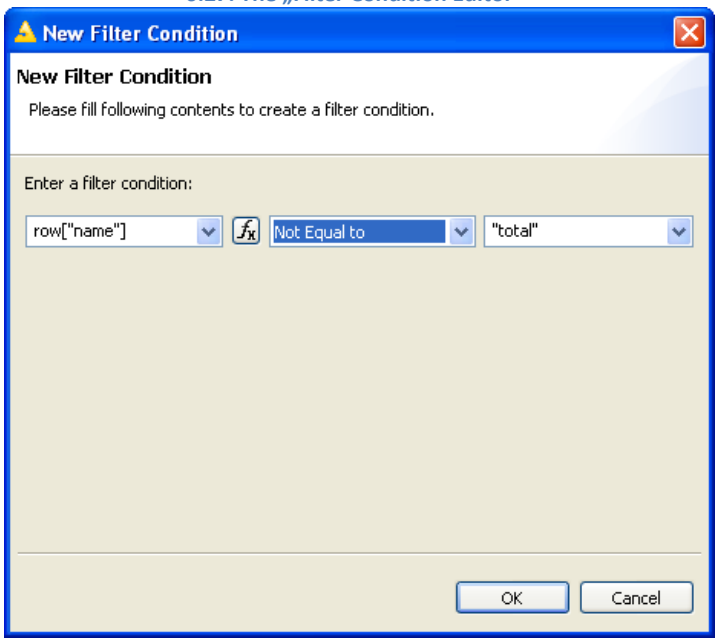


The “New Filter Condition” window appears.

Insert your filtering rule in the “New Filter Condition” window.

Here on the right is an example of a filtering rule that excludes all rows where column “name” = “total”. Notice that “total” is inside quotation marks. Do not forget the quotation marks in a string comparison, since BIRT needs quotation marks to recognize strings.

6.27. The „Filter Condition Editor“



The first chart is supposed to show the assigned money over the years. We selected:

- Data set “money chart”
- Column “name” as Category Series (X-axis) unsorted
- Column “Sum(assigned money(1000))” as Y-Series
- Column “reference year” to group the Y-series

We have only represented one Y-series in this chart and no filter was applied to the data set rows.

Format Chart

The last Wizard step guides you through the chart layout configuration.

On the left, a tree shows the formatting options for the chart.

In "**Series**" you can change the name of the Y-series. The default names are just "Series 1", "Series 2 ", etc….

In "**Value (Y) Series**" you can add and format labels on top of each point of the chart.

Under "**Chart Area**" you can define the background color and style.

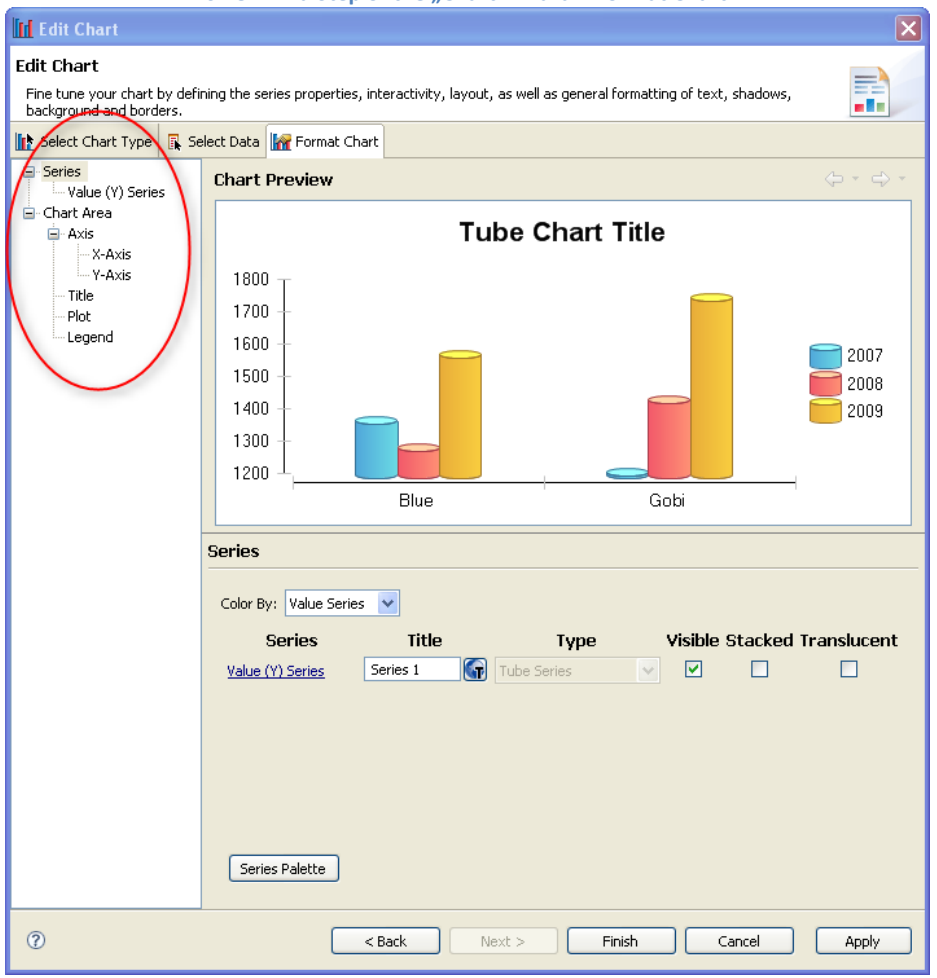
Under "**Axis**", you can define labels, scale, gridlines and everything else related to the chart axis (X-axis or Y-axis).

"Title" has options for the title text, layout, and font.

"**Plot**" is similar to "Chart Area", but refers only to the plotting space.

"**Legend**" helps you with the position, layout, font properties and everything else related to the chart legend.

6.28. Third Step of the „Chart Wizard": Format Chart



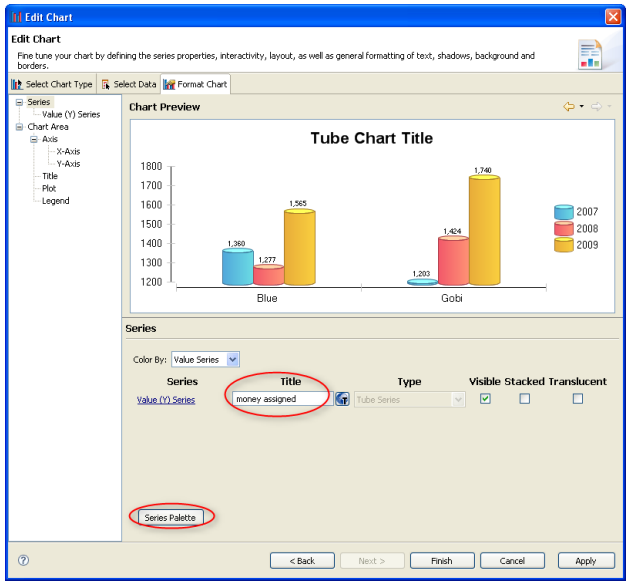
Series

In “Series” you can change the name (labeled as “Title”) of each Y-series. The default names are just “Series 1”, “Series 2 “, etc…. which are not very meaningful. The Y-series can be hidden by disabling the checkbox “Visible” on the right of the “Title” textbox.

The “Series Palette” button leads to a choice of colors for the Y-series. You can select a different color for each one of the Y-series values.

We changed the name of the Y-series from “Series 1” to “money assigned”. This name will appear in the legend. We kept the default series palette.

6.29. Chart Format: Series



Value (Y) Series

In “Value (Y) Series” you can add labels on top of each point of the plot, by enabling the option “Show Series Labels”.

The “Labels” button opens the “Series Labels” window to format the series labels.

Series Label window

The “Series Labels” window helps us format the labels on top of each point in the plot, providing we choose to make them visible.

Here you can define the label position, font, background, shadow, outline, and even inset points.

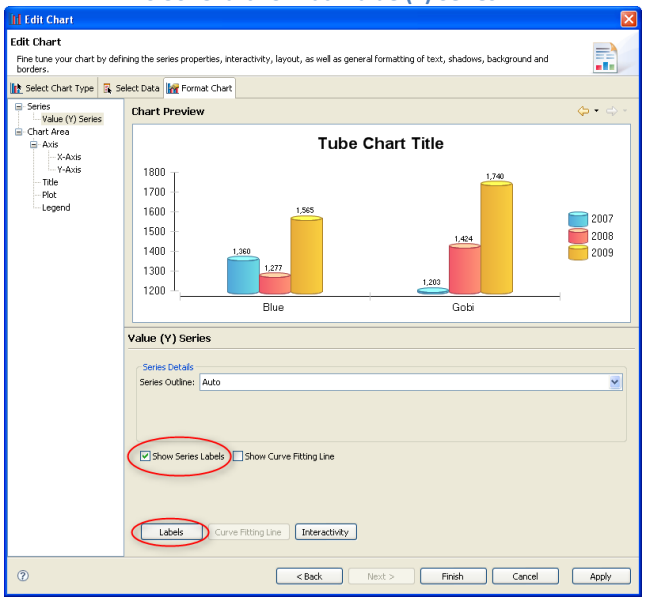
You can also define which values you want shown on top of each point: current Y-value, percent Y-value, X-value, or series name. The label can also be built around the shown value with a prefix, a suffix, and a separator.

The small button with an “A” inside leads to the “Font Editor”.

The “Format” button leads to the “Format Editor” (you need to select an item in the “Values” list to enable this button).

There is no “OK” or “Cancel” button in this “Series Labels” dialog. The new settings are applied immediately. For the “Projects” report we decided to make the series labels visible.

6.30. Chart Format: Value (Y) Series



6.31. The „Series Labels“ window. The „A” button opens the “Font Editor”. The “Format” button opens the “Format Editor”

Font Editor

The “Font Editor” is a standard window that you will find in the “Format Chart” step anywhere, where it is possible to change a font format. It contains the usual font format options: font name, size, style, color. A new option is “Rotation”.

“Rotation” rotates the label by the required number of degrees. “0 degrees” (= the red dot in the tachymeter) corresponds to horizontally written labels. “-90 degrees” writes labels vertically from top to bottom. “+90 degrees” writes labels still vertically but from bottom to top. “-45 degrees” writes labels on a 45 degrees pending line from top to bottom. And so on … The option “Rotation” is very useful for crowded charts or for very long labels.

For the charts in the report “Projects” the only setting we made was to specify the series labels font size as 7.

Format Editor

The “Format Editor” is used to format numeric values, dates, and even strings. The most common usage is however to format numbers.

There are 4 possible number formats: none, standard, advanced, fraction. A multiplier is used to represent numbers with smaller strings, for example money in million units rather than in real currency. The fraction digits are the digits after the comma. Prefix and suffix are also available to format strings and are used to build a label around the basic value.

In our chart we formatted the series labels on “Value data” (that is the data of the series) using 2 decimal digits.

6.32. Chart Format: Chart Area

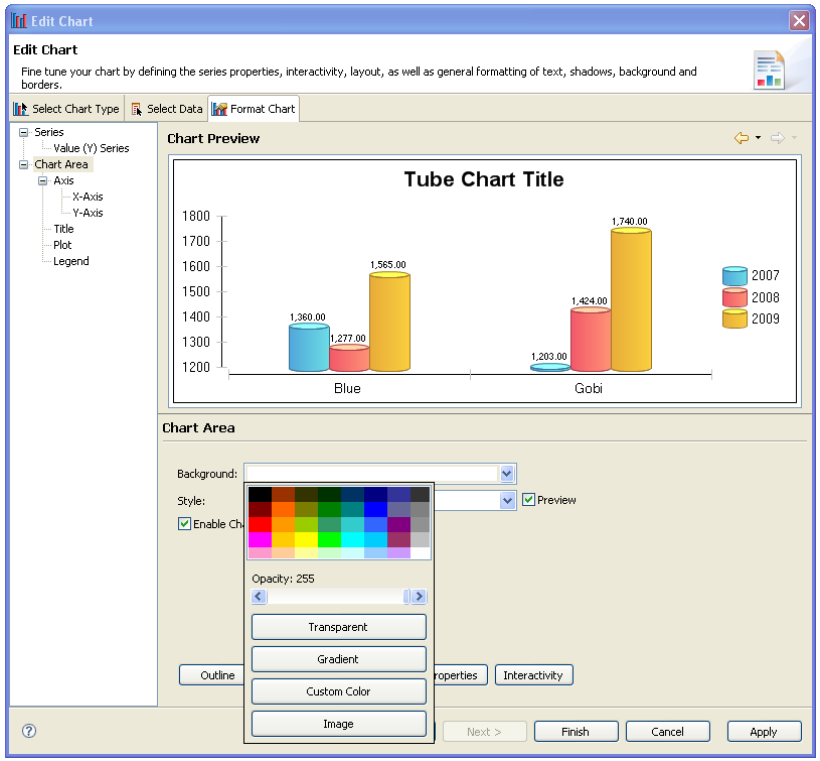


Chart Area

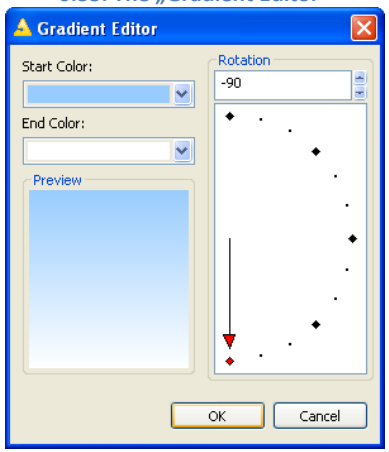
In the “Chart Area” you can define the background color and style of the chart.

If you click the “Background” menu, you are shown a number of options you can use to set the background:

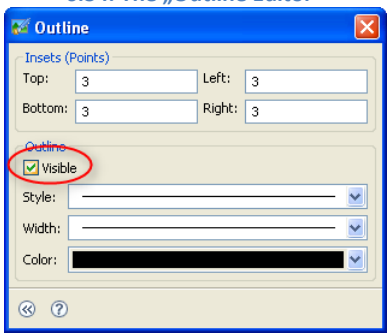
- A simple color
- “Transparent” which means no background color
- A gradient between two colors
- A custom color
- An image

We selected the “Gradient” option. The “Gradient Editor” needs the start and end color and the gradient direction expressed in degrees. Finally, we made the chart outline visible by clicking the “Outline” button and enabling the option “Visible” in the “Outline Editor”.

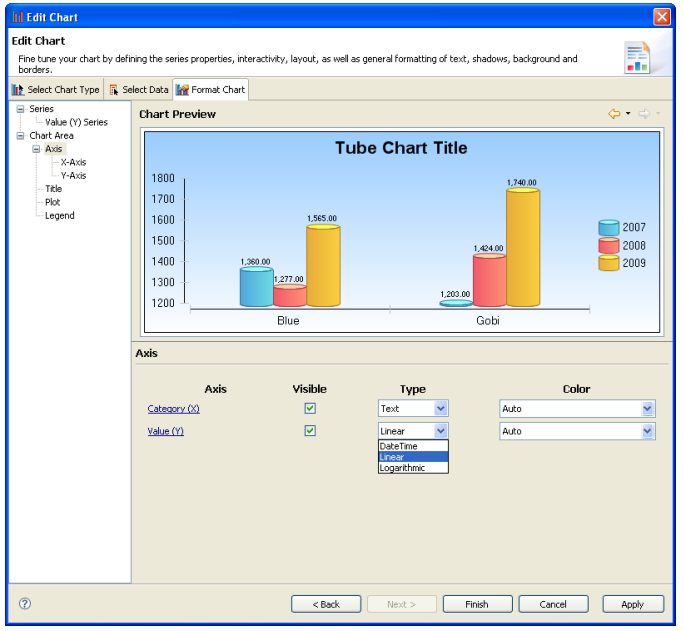
6.33. The „Gradient Editor“



6.34. The „Outline Editor“



6.35. Chart Format: Axis



Axis

Under “Axis”, you can define the type and color of both X-axis and Y-axis. There are a few axis types available depending on the value types displayed on the axis (Text, Number, or Datetime). Linear and logarithmic axes apply only to numerical values.

Let’s leave the default linear scale for the value(Y) axis.

All other axis settings, like fonts, gridlines, and scale can be defined for each axis separately. The two windows for X-axis settings and Y-axis settings are almost identical, besides two category options in the X-axis frame.

X-Axis / Y-Axis

Here the user can set an appropriate title and make it visible.

The most important part is to define the axis labels: format, font, and layout. The usual “A” button leads the user to the “Font Editor”.

The button with the Format icon leads to the “Format Editor”.

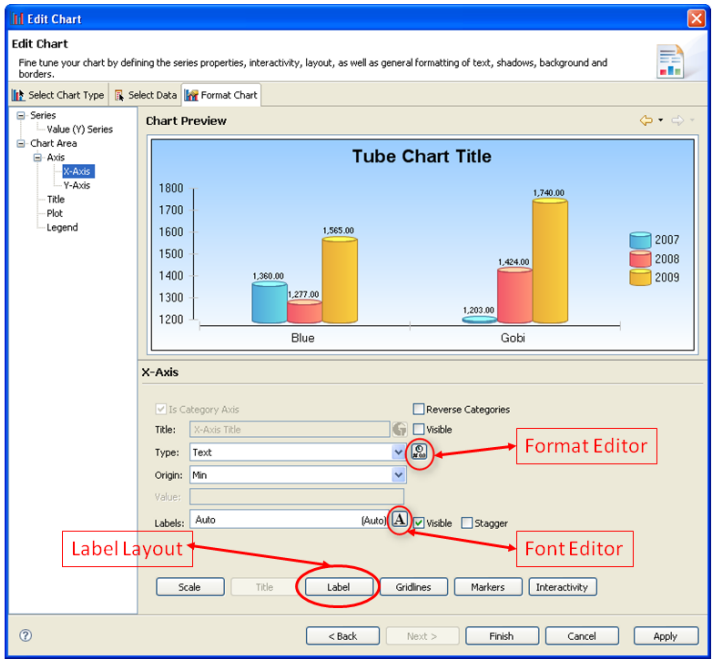
The “Label” button leads to the “Label Layout Editor”, where we can define the label position, background, outline, etc….

The “Scale” button defines the step size for numerical values on the axis. It is disabled for text values.

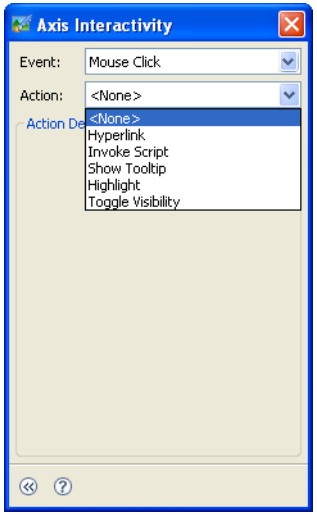
The “Title” button defines font and layout of the axis title if the checkbox was enabled to make the title visible.

The “Markers” button introduces lines to mark areas of the plot.

6.36. Chart Format: X-axis



6.37. The „Axis Interactivity“ window



The “Interactivity” button opens the “Axis Interactivity” window where you can set an action to follow an event. This is used for dashboards or html reports. For example a mouse-click can start a Java script. Many events, such as the mouse-click, and many actions, such as a hyperlink or a script, are available.

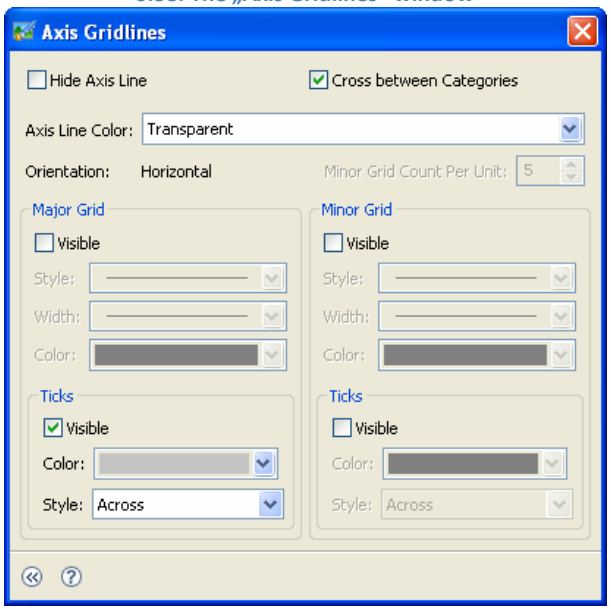
The “Gridlines” button opens the “Axis Gridlines” window to enable gridlines for this axis; that is horizontal gridlines for the Y-axis and vertical gridlines for the X-axis.

There are major and minor grids on the plot as well as ticks on the axis.

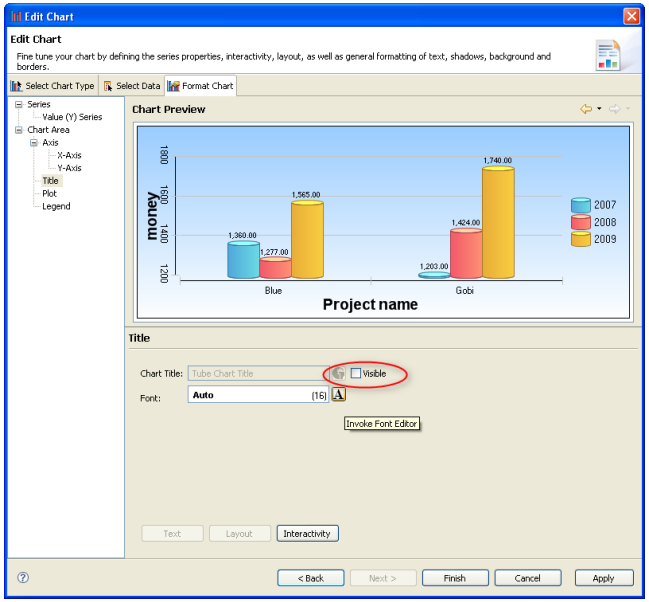
For the “Projects” report we enabled the following:

- Gridlines on the Y-axis, major grid and major grid ticks only. We overlooked the minor grid not to make the chart too crowded.
- Labels with font size 7 and rotated to -90 degrees on the X-axis
- Title visible on both axis with “Project name” as text for the X-axis and “money” for the Y-axis, font size is set to 8 and rotated to -90 degrees on the Y-axis
- No interactivity
- No markers
- No scale

6.38. The „Axis Gridlines“ window



6.39. Chart Format: Title



Title

“Title**”** sets a title in the chart. If you enable the title to be visible, the “Title” frame has options for the title layout, font, and interactivity. I usually do not set the title to be visible, because it takes space from the chart. I use a label on top of the chart in the report layout to act as the chart title.

In the “Projects” report we have disabled the title.

Plot

"Plot" is similar to "Chart Area", but refers only to the plotting space.

Legend

"**Legend**" helps you with the position, the layout, the font properties and everything else related to the chart legend.

If you decide to include a legend in the chart, first of all you need to make the legend visible in the legend frame ("Visible" checkbox at the very beginning of the "Legend" frame).

After that, you need to define the legend layout ("Layout" button) and font properties ("Entries" button).

6.40. Chart Format: Legend. Button „Layout" leads to the "Legend Layout" window. Button "Entries" leads to the "legend Entries" window.

In the "Projects" report we set the following properties for the legend:

- Font size: 7
- Orientation: horizontal
- Direction: left to right
- Position: above

When you are finished formatting the chart, click "Finish". The chart wizard takes you back to the report.

Resize the chart to fit the grid cell. Insert a label above the chart to make the chart title, for example where the text is "money assigned per year to each project".

How to change the chart properties

Change a format property

Run the “Preview”. If you do not like what the chart looks like, just go back to the “Layout” tab, double-click the chart and change the settings that you did not like. In the “Projects” report, for example, the “Series Labels” look a bit too crowded. To disable the “Series Labels”:

- Double-click the chart
- At the top, select the “Format Chart” tab
- Select “Value (Y) Series”
- Disable the “Show Series Labels” checkbox
- Click the “Finish” button

Change data assignment

We need to create an identical chart on the right cell of the grid, but with reference to the money used instead of the money assigned.

- Copy and paste the chart and its title label from the cell on the left to the cell on the right
- Double-click the chart on the right
- Select the “Select Data” tab
- In “Data Preview”, right-click the header of column “Sum(money used (1000))”
- Select “Plot as Value Y Series”
- Click the “Finish” button

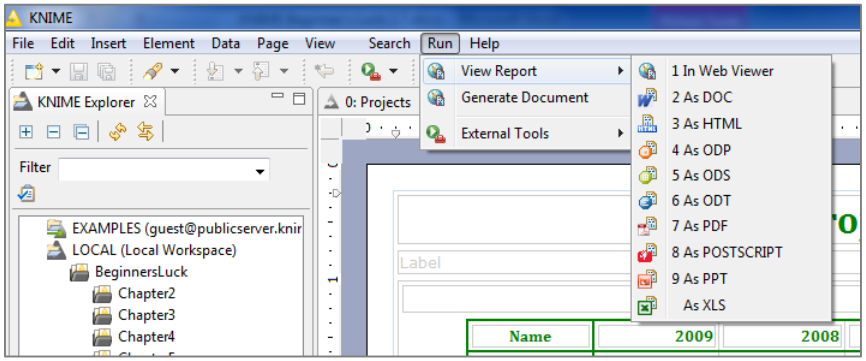
6.13. Generate the final document

In order to generate the final document, go to the Top Menu:

- Select “Run”
- Select “View Report”
- Select the format for your report, for example “PPT” for Powerpoint

BIRT generates your document in the desired format.

6.41. Generate the final document



6.14. Exercises

The exercises for this chapter follow on from the exercises in Chapter 5. In particular, they require shaping a report layout for the data sets built in Chapter 5 exercises.

Exercise 1

Using the workflow built in Chapter 5\Exercise 1, build a report with:

- A title “income by work class”
- A table on the left side like:

Work class	Income <= 50K	Income > 50K
[work class]	[nr <= 50K]	[nr > 50K]

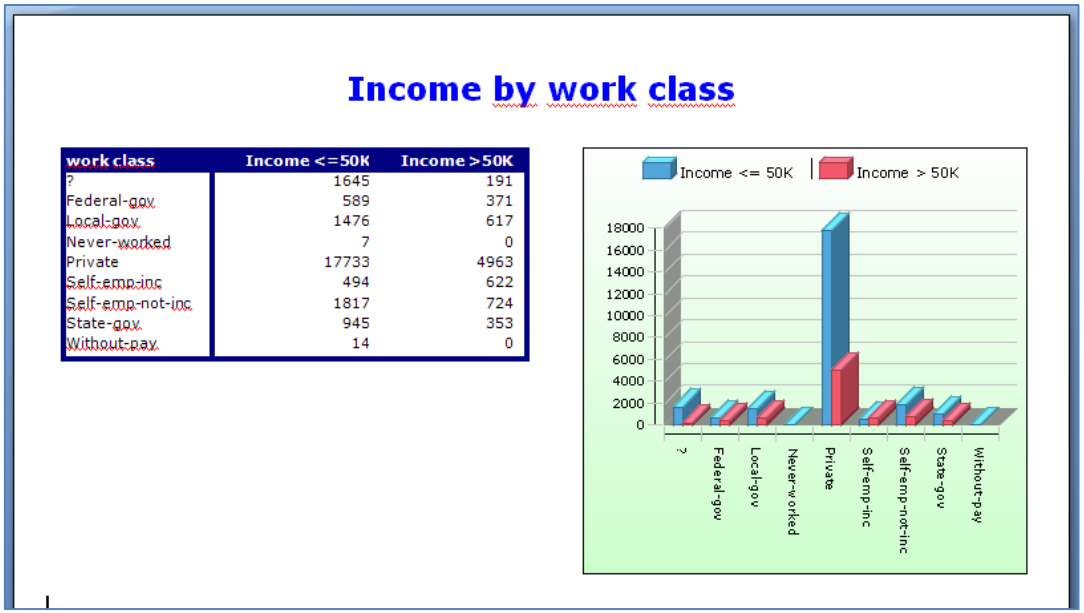
- A bar chart with:
  - Work class on the X-axis
  - “Income <= 50K” and “Income > 50K” on the Y-axis
  - Background gradient style
  - Font size 7 on the axis
  - Font size 8 in the legend
  - Legend placed above the plot and running horizontally

- No title
- No axis titles

Export as Word document

Solution to Exercise 1

6.42 Exercise 1: the final report



Exercise 1a

In the chart in Exercise 1, the “Private” work class causes the scale too large to see the other work class incomes.

Thus, extend Exercise 1 to:

- Define style sheets for table cells and table headers
- Apply the style sheets to the table. The resulting table must look the same as the original one
- Remove the “Private” work class from the chart

Solution to Exercise 1a

6.43. Exercise 1a: the final report without work class "Private"

Exercise 2

From the workflow developed in Chapter5/Exercise2 build a report with:

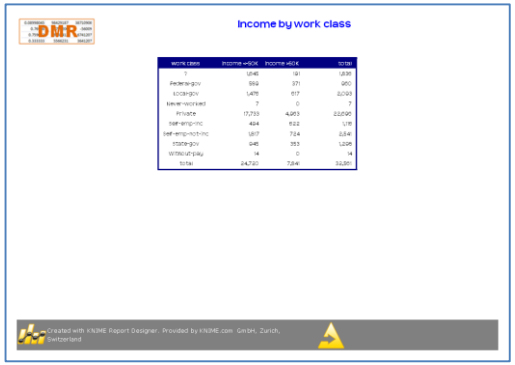
- A running header with a logo and a running title "Income by work class"
    - The logo is an arbitrary image
    - Title has font "Teen", font size "18 points", font color "Blue", and font style "Bold"
- Landscape format to be exported into PPT slides
- Table centered in the first slide, same layout as in exercise 1, but font "Teen"
- 2 charts in the second slide side by side
    - Chart on the left shows "total nr of people" on the Y-axis and "work class" on the X-axis
        - Use line chart
        - Remove "total" values from the chart
        - Set all fonts as "Teen"

        - No legend
        - Change the title to “Nr of people by work class”
        - Show labels on each chart point
        - Show axis titles

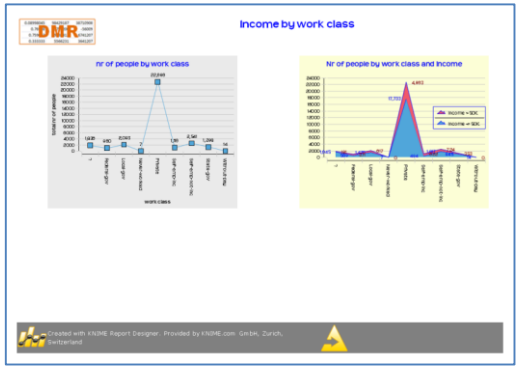
    - Chart on the right shows the nr of people by work class with income <= 50K and with income > 50K
        - Use Area chart
        - Remove “total” row
        - Set all fonts as “Teen”
        - Set legend inside the chart on the right with outline
        - Change title to “nr of people by work class and income”
        - Show labels on each chart point,
            - use the same color as used for the corresponding area
            - place labels on the left and right of the point to make them more readable
        - Show axis titles
- Export as ppt slides

Solution to Exercise 2

6.44 Exercise 2: the final report, slide 1



6.45. Exercise 2: the final report, slide 2



Exercise 3

With the workflow designed in Chapter5/Exercise 3, build the following report.

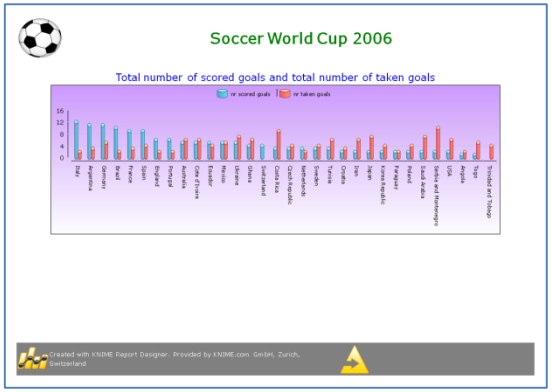
- Running title + logo
    - Use an arbitrary image for the logo
    - Running title “Soccer World Cup 2006”

List of charts one on top of the other:

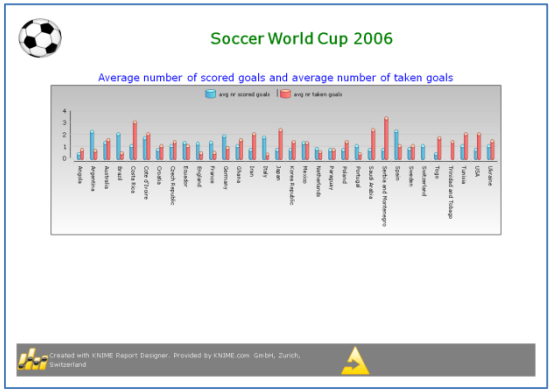
- Bar/Tube Chart with total number of scored goals and total number of taken goals (Y-axis) vs. team (X-axis)
- Bar/Tube Chart with average number of scored goals and average numbers of taken goals (Y-axis) vs. team (X-axis)
- Average number of scored goals vs. average number of taken goals with team name as label on each chart point
- Format legends, axis, axis titles, etc … so that the charts are readable
- Display the table with:
    - Team name
    - Number of played games
    - Number of scored goals
    - Number of taken goals
    - Average number of scored goals
    - Average number of taken goals
    - Fit measure
- Use different font colors for teams with fit measure > 0 (green), fit measure = 0 (orange), fit measure < 0 (red).
- Format the report to have only one chart/table per page

Solution to Exercise 3

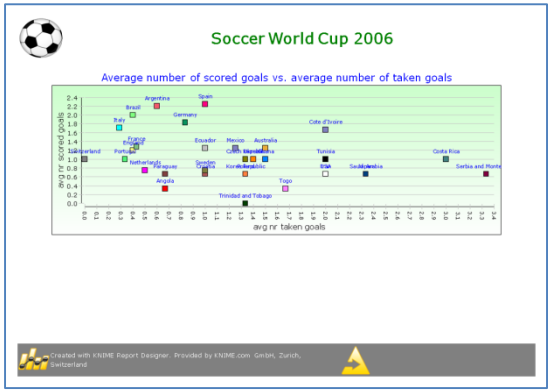
6.46. Exercise 3: final report, slide 1



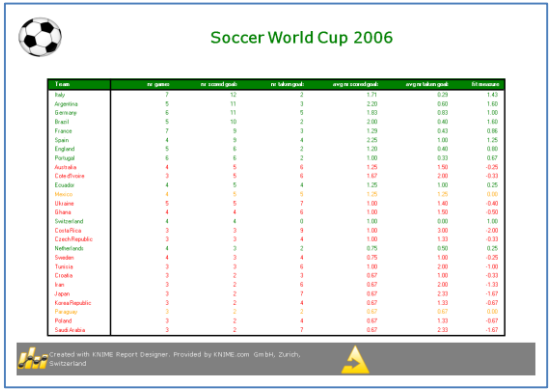
6.47. Exercise 3: final report, slide 2



6.48. Exercise 3: final report, slide 3



6.49. Exercise 3: final report, slide 4



6.50. Exercise 3: final report, slide 5

Note. In the Table's "Page Break" property you can also find the number of rows for each page. You need to adjust this number to fit your table nicely in each page of the report.

References

[1] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Koetter, T. Meinl, P. Ohl, C. Sieb, and B. Wiswedel, “KNIME: The Konstanz Information Miner”. KDD 2006 (http://www.kdd2006.com/docs/KDD06_Demo_13_Knime.pdf)

[2] D. Peh, N. Hague, J. Tatchell, “BIRT. A field Guide to Reporting”, Addison-Wesley, 2008

[3] C.M. Bishop, “Pattern Recognition and Machine Learning”, Springer (2007)

[4] M.R. Berthold, D.J. Hand, “Intelligent Data Analysis: An Introduction“, Springer Verlag, 1999

[5] M.R. Berthold**,** C. Borgelt , F**.** Höppner**,** F. Klawonn, **“**Guide to intelligent data analysis”, Springer 2010

[6] D. L. Olson, D. Delen, **“**Advanced Data Mining Techniques” Springer; 2008

[7] D.G. Altman, J.M. Bland, “Diagnostic tests. 1: Sensitivity and specificity” *BMJ* 308 (6943): 1552; 1994

[8] J.R. Quinlan, "C4.5 Programs for machine learning", Morgan Kaufmann Publishers Inc. , 1993

[9] J. Shafer, R. Agrawal, M. Mehta, "SPRINT: A Scalable Parallel Classifier for Data Mining", Proceedings of the 26th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc. ,1996 (http://citeseer.ist.psu.edu/shafer96sprint.html)

[10] B. Wiswedel, M.R. Berthold, “Fuzzy Clustering in Parallel Universes” , International Journal of Approximate Reasoning, Elsevier Inc., 2007

Node and Topic Index

H

I

J

K

L

M

N

P

Q

R

S

T

U

V

W

Z

KNIME press

KNIME Beginner's Luck

This book is born from a series of lectures on KNIME and KNIME Reporting. It gives a quite detailed overview of the main tools and philosophy of KNIME data analysis platform. The goal is to empower new KNIME users with the necessary knowledge to start analyzing, manipulating, and reporting even complex data. No previous knowledge of KNIME is required. The book has been updated for KNIME 2.10.

The book shows:

- how to move inside (and install) the KNIME platform (Chapter 1);
- how to build a workflow (Chapter 2);
- how to manipulate data (Chapters 2,3,4,and 5);
- how to perform a visual data exploration (Chapter 3);
- how to build models from data (Chapter 4);
- how to design and run reports (Chapters 5 and 6).

About the Author

Dr Rosaria Silipo has been mining data since her master degree in 1992. She kept mining data throughout all her doctoral program, her postdoctoral program, and most of her following job positions. She has many years of experience in data analysis, reporting, business intelligence, training, and writing. In the last few years she has been using KNIME for all her consulting work, becoming a KNIME certified trainer and an expert in the KNIME Reporting tool.

ISBN: 978-3-033-02850-0