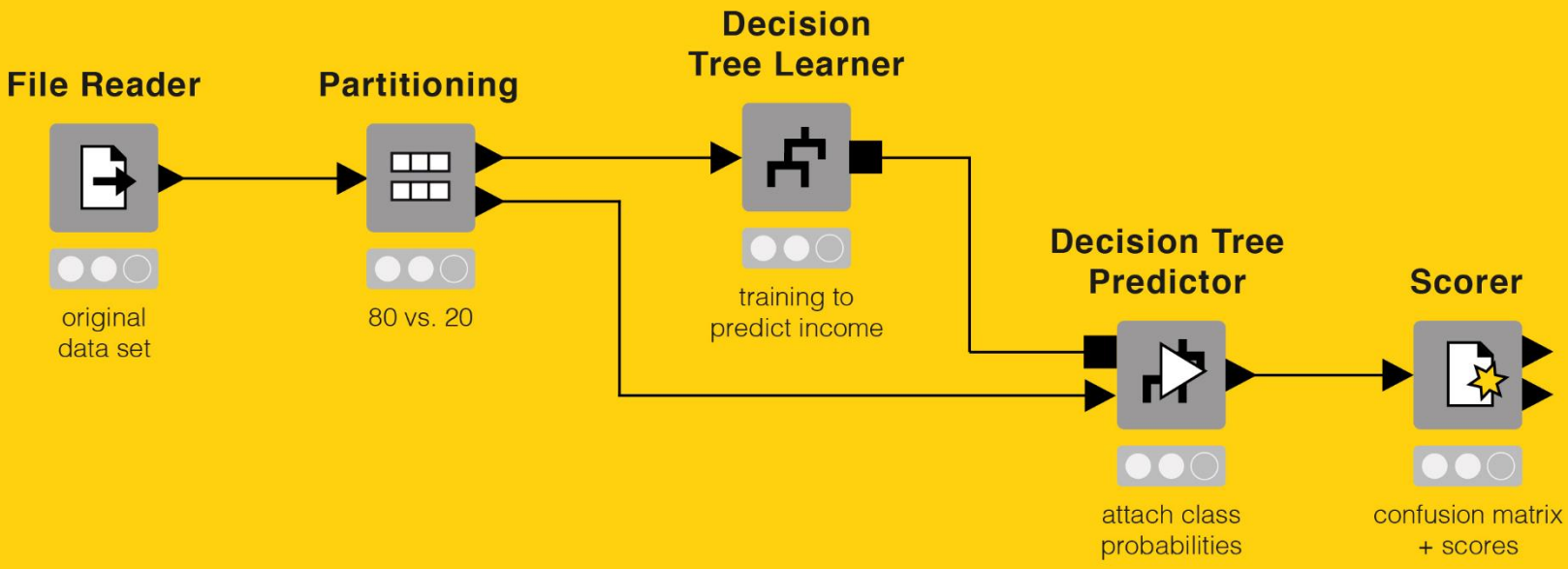
KNIME®
BEGINNER'S LUCK



A Guide to KNIME Analytics Platform for Beginners

Author: Rosaria Silipo

This book has been updated for **KNIME 3.7**.

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 advice 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 machine learning, 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 detail 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 3.7. They should also work under subsequent KNIME versions, although there might be slight differences in their appearance.

1.2. KNIME community

Web Links	
http://www.knime.org	The root page in the KNIME web site.
https://www.knime.com/knime-server	The first place to look for information about KNIME products. The open source KNIME Analytics Platform can be downloaded here.
https://www.knime.com/knime-introductory-course	The landing page to learn more about the specific KNIME functionalities. It covers the whole data science cycle from data access and data exploration to machine learning and control structures.
http://www.knime.org/learning-hub	This is a collection of learning material - as web sites, videos, webinars, courses, and more. It is organized by topic, like text mining or chemistry, or basic KNIME nodes, 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.

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 Course web site https://www.knime.com/courses
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 Events web page at https://www.knime.com/learning/events
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 (https://www.knime.com/learning/events)
KNIME TV Channel on YouTube	KNIME has its own video channel on YouTube, named KNIMETV. 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	For an overview of data analysis, data mining, and data science, please check: *Berthold M.R., Borgelt C., Höppner F., Klawonn F.,"Guide to intelligent data analysis", Springer 2010.*

1.3. Download and install 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 KNIME server, which is described at https://www.knime.org/knime-server

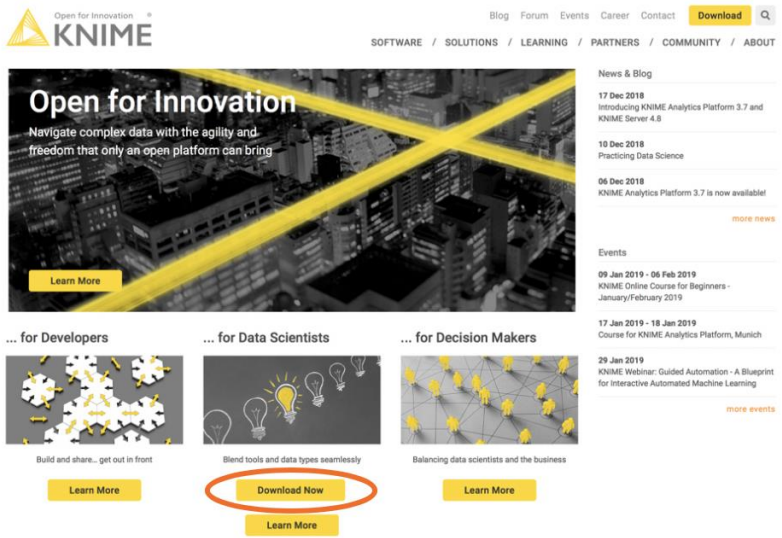
Analytically speaking, the functionalities of the two versions are the same. The KNIME Server includes a number of useful features for team collaboration, enterprise workflow development, data warehousing, integration, and scalability for the data science lab. In this book we work with the KNIME Analytics Platform (open source) version 3.7.

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 proceed to step 2 “Download KNIME” at the top of the page
- Choose the version that suits your environment (Windows/Mac/Linux, 32 bit/64 bit, with or without Installer for Windows) optionally including all free extensions
- 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 _3.x.y” folder to the selected location.

1.1. The KNIME Download web page



1.4. Workspace

To start KNIME, open the folder where KNIME has been installed and run knime.exe (or knime on a Linux/Mac machine). If you have installed KNIME using the Installer, then you can just click the icon on your desktop or on your Windows main menu.

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

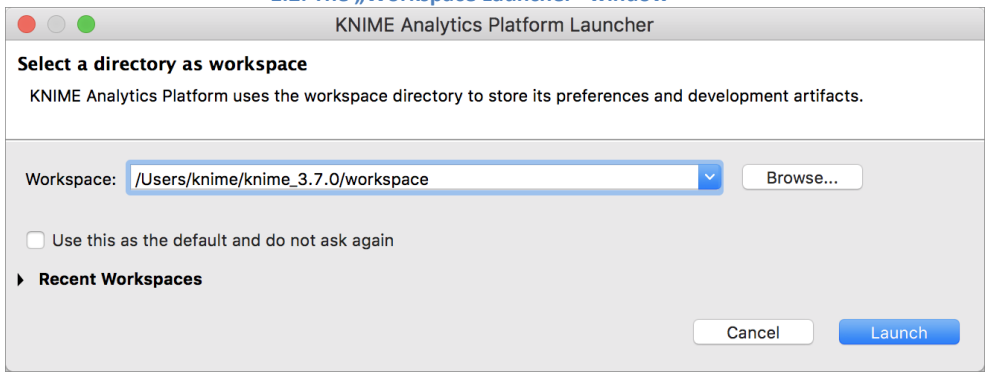
The "Workspace Launcher"

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



Once KNIME has been opened, from within the KNIME workbench you can switch to another workspace folder, by selecting "File" in the top menu and then "Switch Workspace". After selecting the new workspace, KNIME restarts, showing the workflow list from the newly selected workspace. Notice that if the workspace folder does not exist, it will be automatically created.

If I have a large number of customers for example, I can use a different workspace for each one of them. 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_3.x.y\workspace".

1.5. KNIME workflow

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

Small little boxes, called nodes, are dedicated each to implement and execute a given task. A sequence of nodes makes a workflow to process the data to reach the desired result.

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

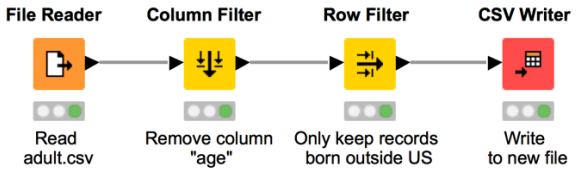
KNIME implements its workflows **graphically**. Each step of the data analysis is implemented and executed through a little box, called **node**. A sequence of nodes makes a workflow.

In the KNIME whitepaper [1] a workflow is defined as follows: *"Workflows in KNIME are 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

1.3. 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.

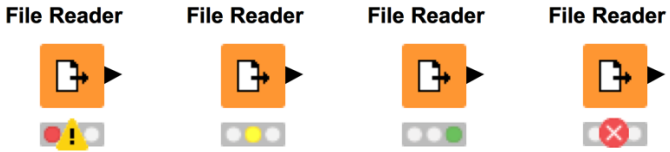
Every node in KNIME has 4 states:

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

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

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

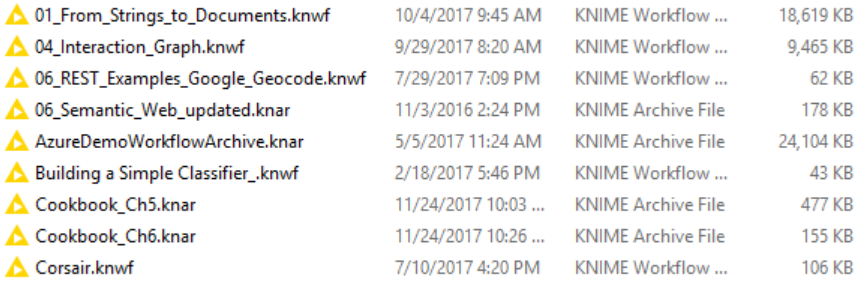
1.4. File Reader node with different states



1.6. .knwf and .knar file extensions

KNIME workflows can be packaged and exported in .knwf or .knar files. A .knwf file contains only one workflow, while a .knar file contains a group of workflows. Such extensions are associated with the KNIME Analytics Platform. A double-click opens the KNIME Analytics Platform and the workflow inside the platform.

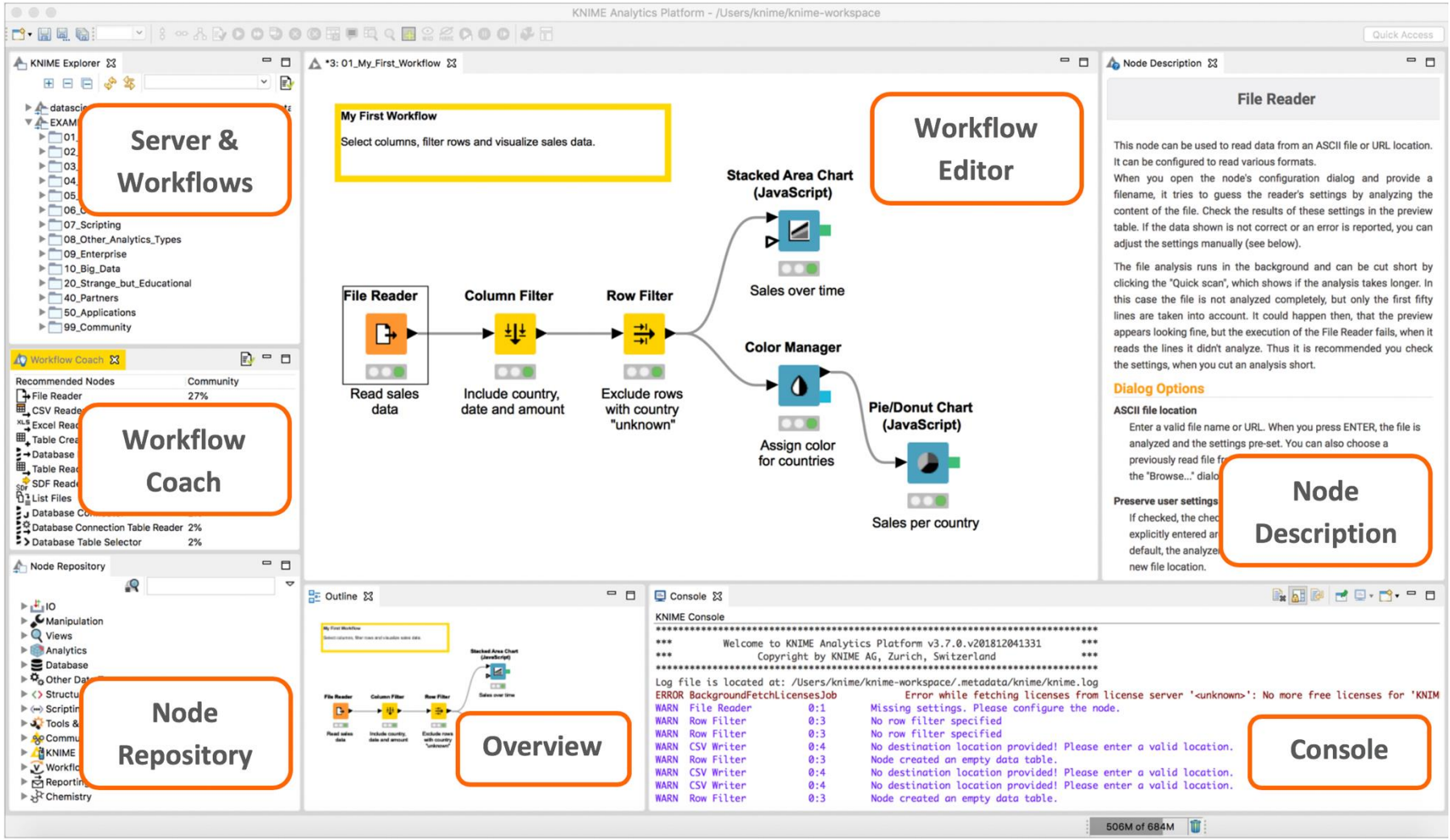
1.5. .knwf and .knar files are associated with KNIME Analytics Platform. A double-click opens them directly in the platform



1.7. 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 some documentation. It also shows a link to create a new workflow, to the “Learning Hub” web page where you can find links to 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 to 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, re-opened, 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, zoom (in %), 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, , Append IDs to node names, Hide all node names, Loop execution options, Change Workflow Editor Settings, Edit Layout in Wrapped Metanodes, configure job manager.

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.

Workflow Coach

This is a node recommendation engine. It will provide the list of the top most likely nodes to follow the currently selected node.

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.

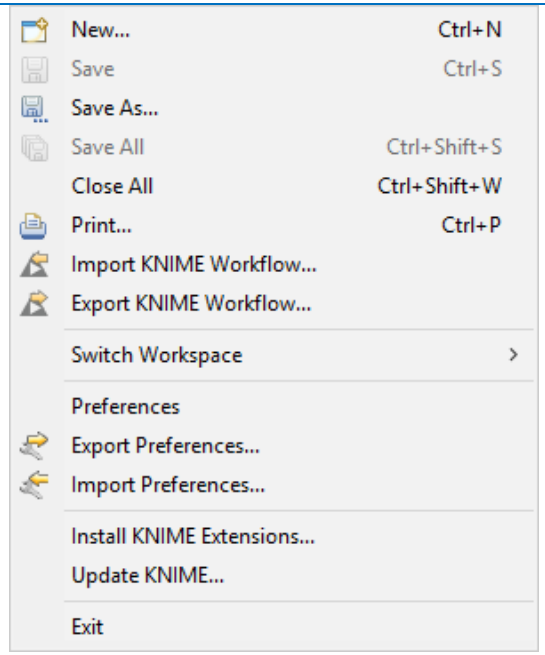
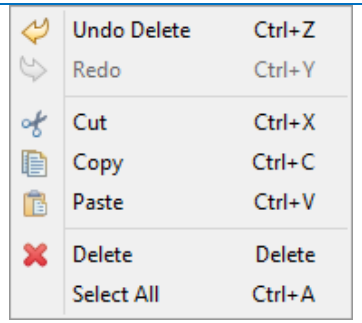
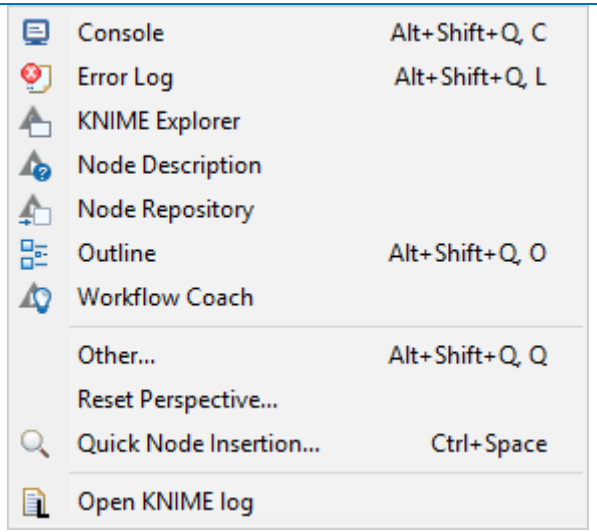
Console

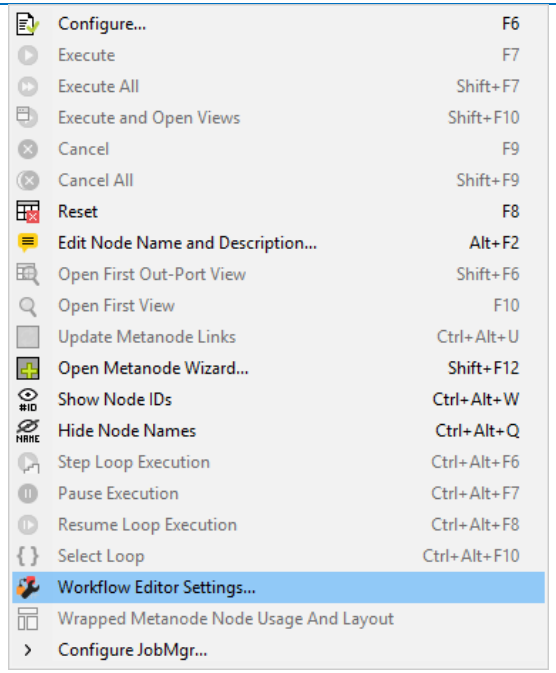
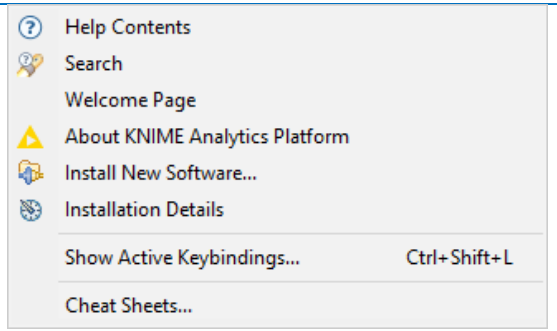
The "Console" panel displays error and warning messages to the user.

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.

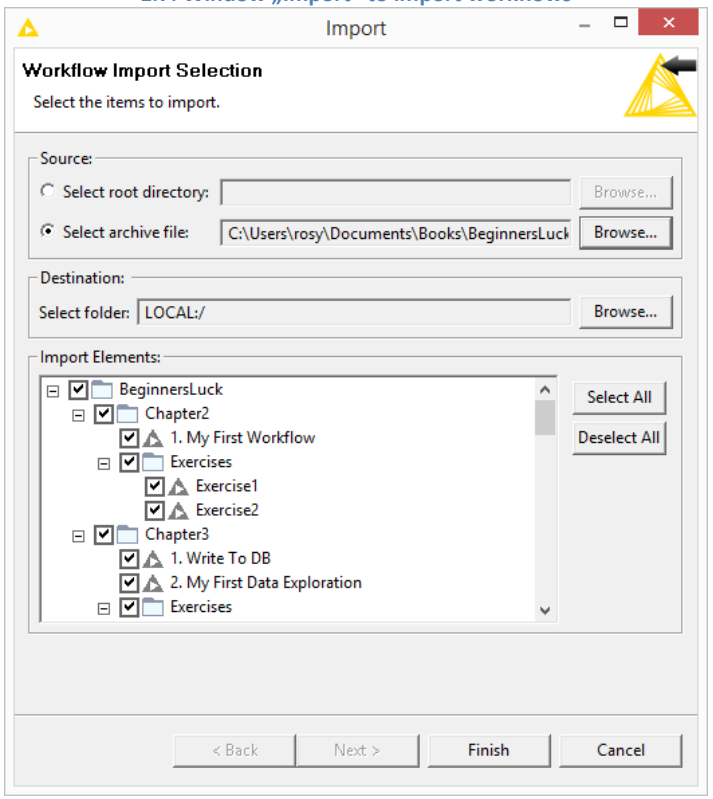
Top menu

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… - Switch Workspace - Preferences - Export/Import Preferences - Install KNIME Extensions - Update KNIME	**Edit** contains 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 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 cannot be re-executed unless it is first 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. **Show Active Keybindings** 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" to import workflows



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

Option "Select root directory" copies the workflow directly from a folder into the current workspace (LOCAL).
Option "Select archive file" reads a workflow from a .knwf or .knar file into the current workspace (LOCAL). .knwf /.knar files can be created through the option "File"-> "Export KNIME workflow".

"File" -> "Export KNIME workflow" exports the one selected workflow to a .knwf or the many selected workflows to a .knar file.

Option "Reset Workflow(s) before export" exports fully resetted workflows without the data produced by each node. 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" or by double-click is definitely safer.

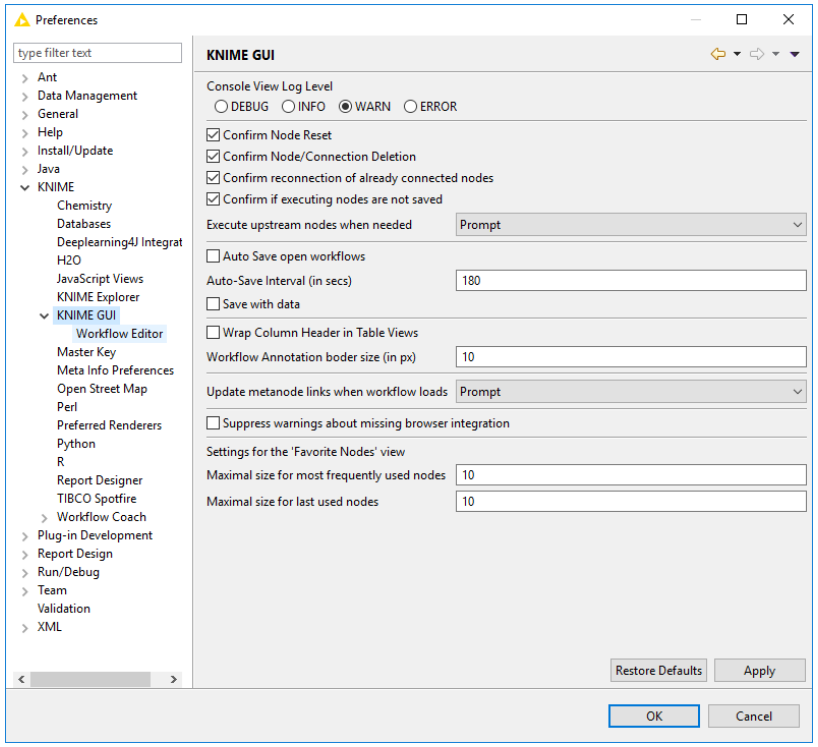
"File" -> "Install KNIME Extensions" and "**Help" -> "Install New Software"** both link to the dialog window for the installation of KNIME Extensions from the KNIME Update sites (see next sections).

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

"File" -> "Preferences" brings you to the window where all KNIME settings can be customized. They can be found under item "KNIME". Let's check them.

- **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 are already available in the driver menu of Database nodes. However, if you need some specific driver file, you can set its path here.

1.8. The "Preferences" window



- **KNIME Explorer** contains the list of the shared repositories via KNIME Server.
- **KNIME GUI** allows the customization of the KNIME workbench options and layout via a number of settings.
- **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.
- Here you can also find the preference settings for the **external packages**, like: H2O, R, Report Designer, Perl, Perl, Open Street Map, and others if you have them installed. In particular, for the external scripts, this page offers the option to set the path to the reference script installation.
- Finally, **Workflow Coach** contains the dataset to be used for the node recommendation engine: the community, a server workspace, or your own local workspace.

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.

Tool Bar

The tool bar is another important piece of the KNIME workbench.

From the right, we find the icon to create a new workflow, save the selected workflow, save as the selected workflow in another location, save all open workflows, undo and redo, switch to the reporting environment, zoom (in %), align selected nodes vertically, align selected nodes horizontally, auto-layout, configure the selected node, execute the selected node, execute all executable nodes, execute selected nodes and open the first data view, cancel selected running nodes, cancel all running nodes, reset selected nodes, edit description of selected node, open first data view of selected nodes, open views of selected nodes, open the Add Metanode Wizard, append IDs to node names, hide node names,

do one loop step, pause loop execution, resume loop execution, change workflow editor settings, open layout editor for wrapped metanodes, configure job manager for all selected nodes. We will see all these options along the course of this book.

For now, I just want to describe the "**Auto Layout**" button. The auto-layout button 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 example after a long development session, the workflow overview has become difficult.

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

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

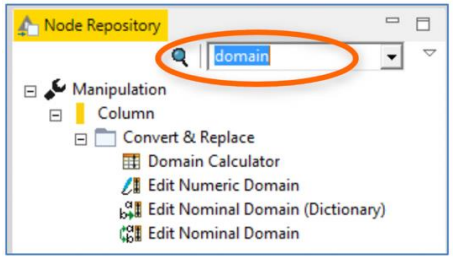
Node Repository

In the lower left corner we find the Node Repository, containing all installed nodes organized in categories and subcategories. KNIME Analytics Platform has accumulated by now more than 1500 nodes. It has become hard to remember the location of each node in the Node Repository. To solve this problem, two search options are available: by exact match and by fuzzy match, both in the search box placed at the top of the Node Repository panel.

Search box

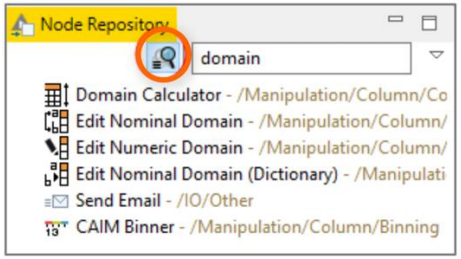
At the top of the "Node Repository" panel there is a **search box**. If you type a keyword in the search box and hit "Enter", you obtain the list of nodes containing an exact match of that keyword. Press the "Esc" key to see all nodes again.

1.10. Word search in the Node Repository panel: exact match mode



Clicking the lens on the left of the search box runs a fuzzy search algorithm leading to a wider matching result list than what found in the previous figure.

1.11. Word Search in the Node Repository panel: fuzzy match mode



KNIME Explorer

In the top left corner of the KNIME workbench, we find the KNIME Explorer panel. This panel contains:

- Under LOCAL the workflows that have been developed in the selected workspace
- The mount points to a number of KNIME Servers
- The workflows contained in the reference workspace of such servers

By default, the KNIME Explorer panel only contains LOCAL and EXAMPLES. As we already stated, LOCAL shows the content of the selected workspace. EXAMPLES points to a read-only public server, accessible via anonymous login. This server hosts a number of example workflows that you can use to jump start a new project.

When you open KNIME Analytics Platform for the first time, you will find a folder named “Example Workflows” containing the solutions to a few common data science use cases, comprehensive of data.

Folders in “KNIME Explorer”, containing workflows, are also called “Workflow Groups”.

Note. KNIME Explorer panel can also host data. Just create a folder under the workspace folder, fill it with data files, and select “Refresh” in the context-menu (right-click) of the “KNIME Explorer” panel.

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:

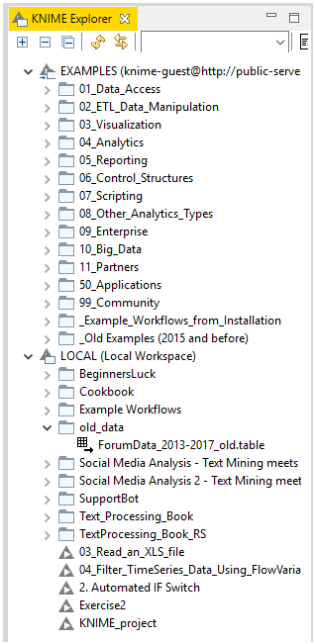
- right click “EXAMPLES” in the “KNIME Explorer” panel
- select “Login”

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 temporarily and in read-only mode. A yellow warning box on top warns that this workflow copy will not be saved.

1.12. KNIME Explorer panel. At the top the content of the EXAMPLES server; below the content of the LOCAL workspace



The KNIME Explorer panel can of course host more than one KNIME Server. It is enough to add server mount points to the list of the available KNIME servers in the KNIME Explorer panel.

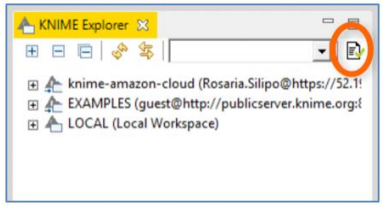
Mounting Servers in KNIME Explorer

To add KNIME instances (servers or teamspaces) to the “KNIME Explorer” panel:

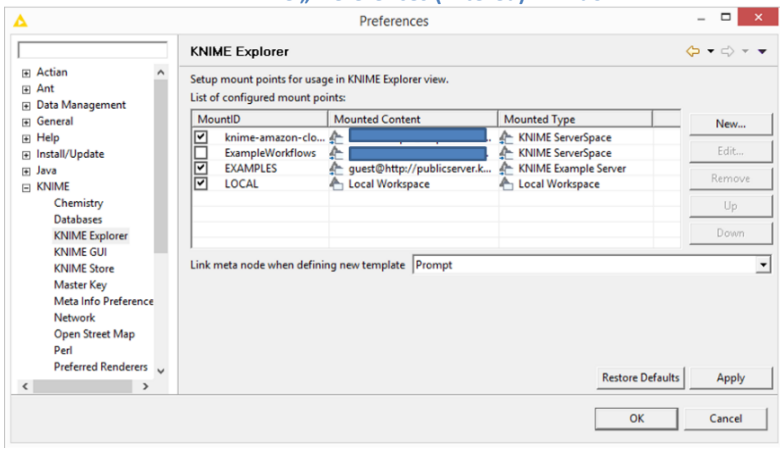
- Select the “KNIME Explorer” panel
- Click the “Configure Explorer View” button
- The “Preferences (Filtered)” window opens on the “KNIME Explorer” page and lists all KNIME Servers and Teamspaces uploaded in this KNIME instance. The two KNIME spaces uploaded by default on every KNIME instance are the local workspace “LOCAL” and the KNIME public Server space “EXAMPLES”.
- Use the “New” and the “Remove” button to add /remove connections to remote servers.
- After clicking the „New“ button, fill in the required information about the server in the “Select New Content” window (Fig. 1.15)
- Use the “Test Connection” button to automatically retrieve the default mountpoint for the selected server.

The same KNIME Explorer “Preferences” page in figure 1.14 can be reached via “File” in the top menu -> “Preferences” -> “KNIME Explorer”.

1.13. The „Configure KNIME Explorer“ button



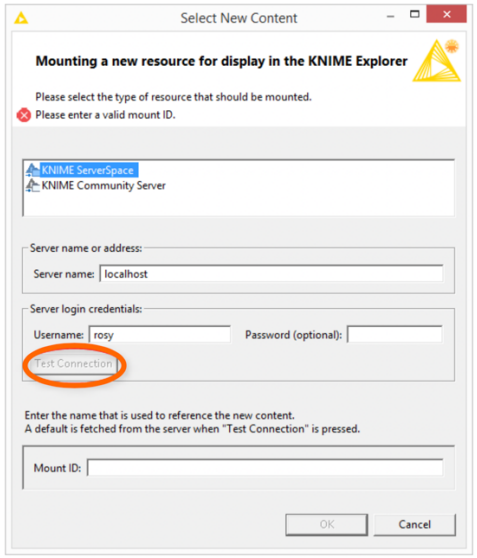
1.14. The „Preferences (Filtered)“ window



To login into any of the available servers in the “KNIME Explorer” panel:

- right-click the server name
- select “Login”
- provide the credentials

1.15. The “Select New Server”” window



Workflow Editor

The central piece of the KNIME workbench consists of the workflow editor itself. This is the place where a workflow is built by adding one node after the other. Nodes are inserted in the workflow editor by drag and drop or double-click. The workflow building process will be described widely in the next sections of this book. In this section here, we will describe how to customize and probably improve the canvas role of the workflow editor space.

In particular, we will describe two options: change the canvas appearance with grids and different connections; introducing annotations to comment the work.

Adding a grid to the canvas and curved connections to the workflows

Almost towards the end, on the right of the tool bar, you can see the “Change Workflow Editor Settings” button. If you click it, the “Workflow Editor Settings” window opens.

1.16. "Change Workflow Editor Settings" Button in Tool Bar

Customizing the Workflow Editor

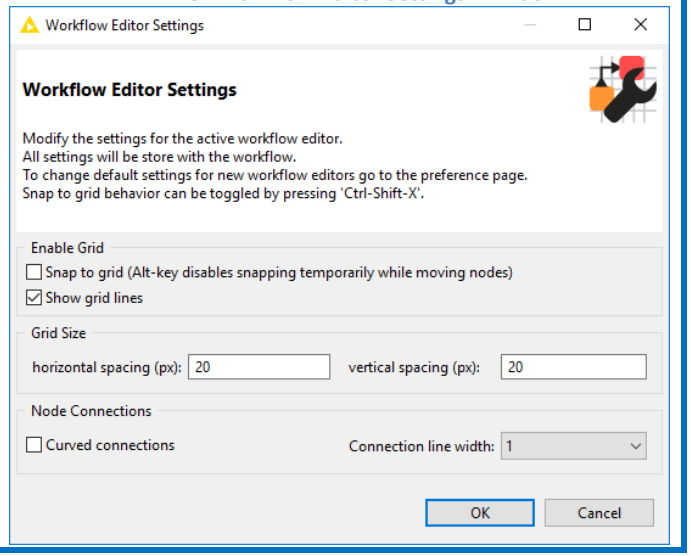
The grid feature contains a few options:

1. "Show grid lines". This shows grid lines in the workflow editor and allows to better align nodes and annotations manually. If this option is enabled, you can set the grid size below.

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.

3. "Node Connections". Here you can enable node connections to follow a curve rather than straight lines. This might leads to more appealing workflow graphics.

1.17. The "Workflow Editor Settings" window



Adding annotations to the canvas

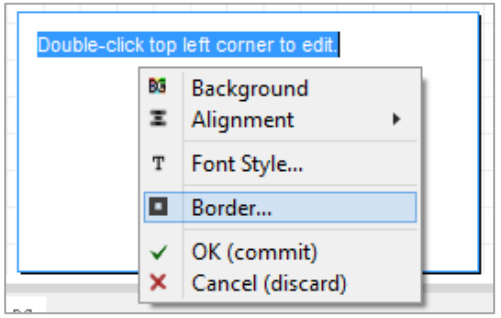
It is also possible to include **annotations** in the workflow editor. Annotations can help to explain the task of the workflow and the function of each node or group of nodes. The result is an improved overview of the workflow general goal and of the single sub-tasks.

Workflow Annotations

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 alignment.

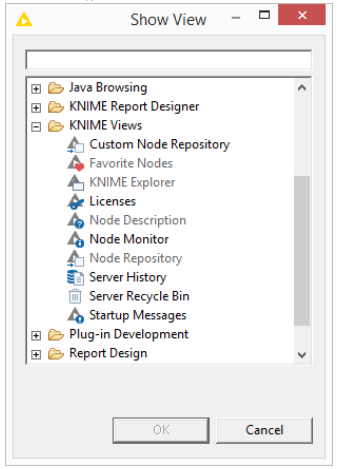
1.18. The Annotation Editor



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

Other Workbench Customizations

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



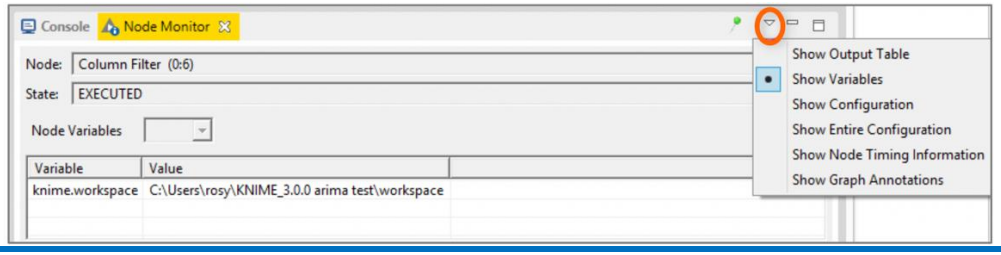
Another possibility for customization consists of adding views. Available views are found in the “View” item in the Top menu.
Popular views, for example, are the “Node Monitor”, the “Custom Node Repository”, and the “Licenses” and “Server” views, if you have a connected server.
All these extra views 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.
“Licenses” allows to monitor your license situation, if you have any.

Node Monitor View

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

- Select “View”-> “Other…” in the top menu
- In the “Show View” window, expand the “KNIME Views” item and double-click “Node Monitor”; a panel, named “Node Monitor”, appears on the side of the “Console” panel; the panel shows the values for the output flow variables, the output data, or 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.

1.20. The Node Monitor View



1.8. Download the KNIME Extensions

KNIME Analytics Platform is an open source product. As every open source product, it benefits from the feedback and the functionalities that the open source community develops. A number of extensions are available for KNIME Analytics Platform. If you have downloaded and installed KNIME Analytics Platform including all its free extensions, you will see the corresponding categories in the Node Repository panel, such as KNIME Labs, Text Processing, R Integration, and many others.

However, if at installation time, you have chosen to install the bare KNIME Analytics Platform without the free extensions, you might need to install them separately at some point on a running KNIME.

Installing KNIME Extensions

To install a new KNIME extension, there are two options.

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

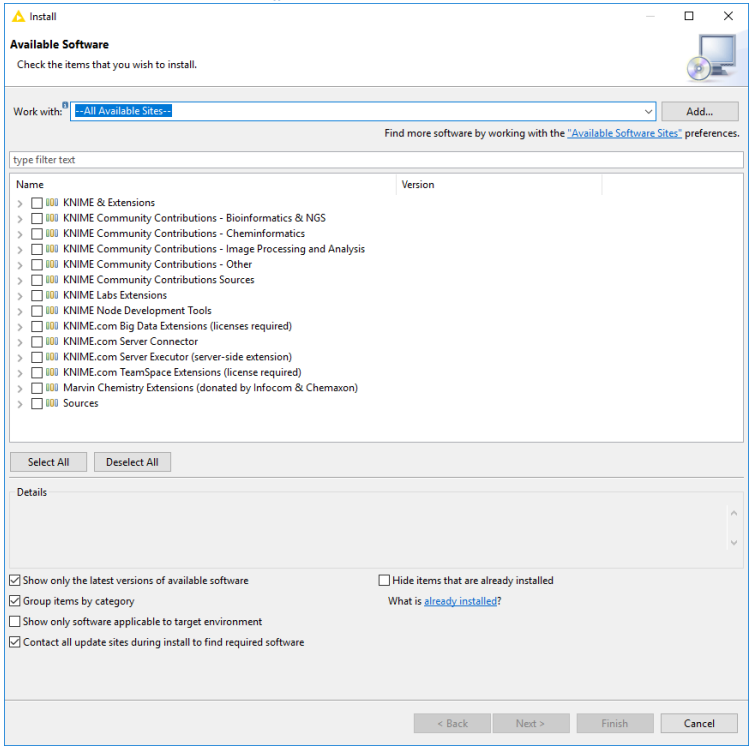
   OR

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/3.x). Then select the extension, click the "**Next**" button and follow the wizard instructions.

Once the selected KNIME extension(s) has/have been installed and KNIME has been restarted, you should see the new category, corresponding to the installed extension, in the "Node Repository" in the KNIME workbench.

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

1.21. The „Available Software" window



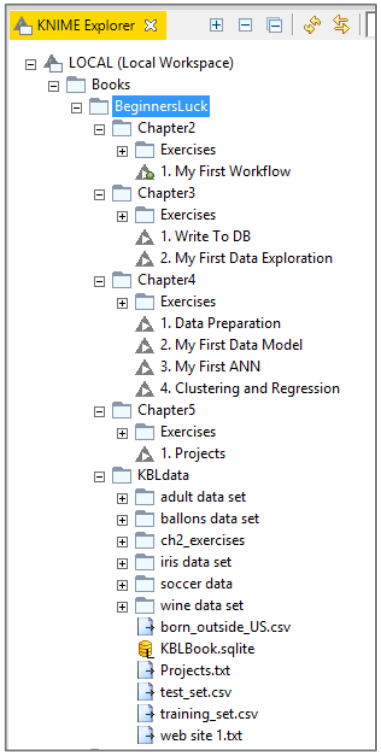
In the “Available Software” window you can find some extension groups: KNIME & Extensions, KNIME Labs Extensions, KNIME Node Development Tools, Sources, and more. “KNIME &Extensions” contains all extensions provided by KNIME for the current release; “KNIME Labs Extensions” contains a number of extensions developed by KNIME, ready to use, but not yet of x.1 release quality; “KNIME Node Development Tools” contains packages with some useful tools for java programmers to develop nodes; “Sources” contains the KNIME source code. Specific packages donated by third parties or community entities might also be available in the list of extensions. They are usually grouped under “Community” categories.

My advice is to install all extensions, even the cheminformatics ones. Many of them contain a number of useful nodes of general usage and not necessarily restricted to that particular domain.

1.9. Data and workflows for this book

When you purchased this book, in the same email with the link to this pdf file, you should also have found a link to the Download Zone file. The Download Zone file is a .knar file that contains the data and workflows used and implemented throughout this book.

1.22. Workflows and data imported from the Download Zone .knar file



- Download the Download Zone .knar file onto your machine. Then:
- Double click it
  OR
  import it into the KNIME Explorer via Select File -> Import KNIME Workflow …

At the end of the import operation, in the KNIME Explorer panel you should find a BeginnersLuck folder containing Chapter2, Chapter3, Chapter4 and Chapter5 subfolders, each one with workflows and exercises to be implemented in the next chapters. You should also find a KBLdata folder containing the required data.

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 for download. Data generated by the author, that is not public data, are located in the "Download Zone" in the KBLData 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

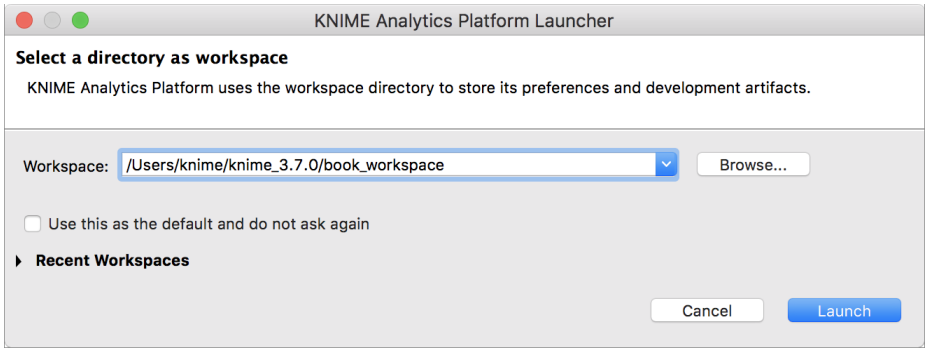
Create your own workspace and name it "book_workspace". You will use this workspace for the next workflows and exercises.

Solution to Exercise 1

- Launch KNIME
- In Workspace Launcher window, click “Browse”
- Select the path for your new workspace
- Click “OK”

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

1.23. Exercise 1: Create workspace "book_workspace"



Exercise 2

Install the following extensions:

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

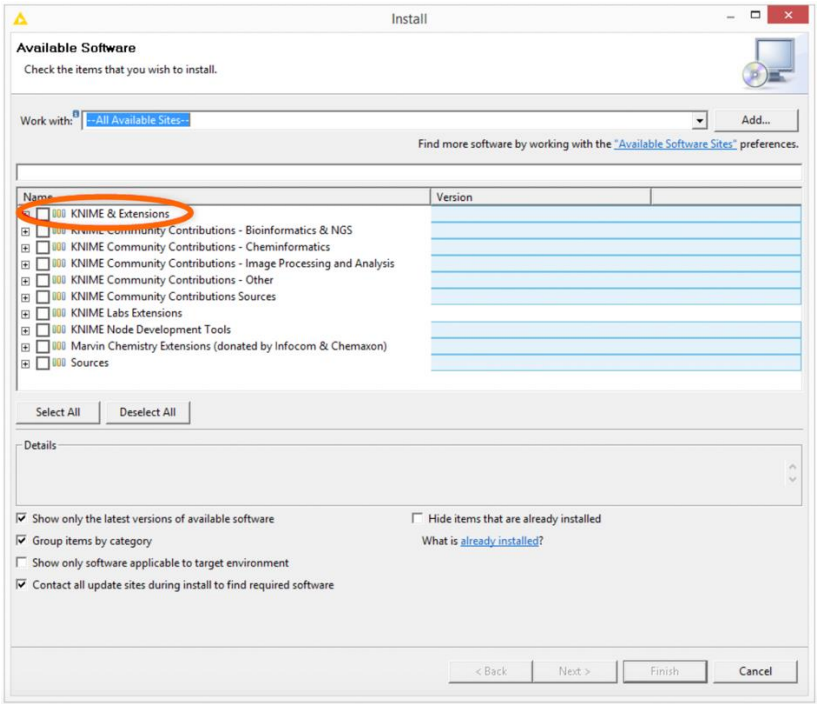
Solution to Exercise 2

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

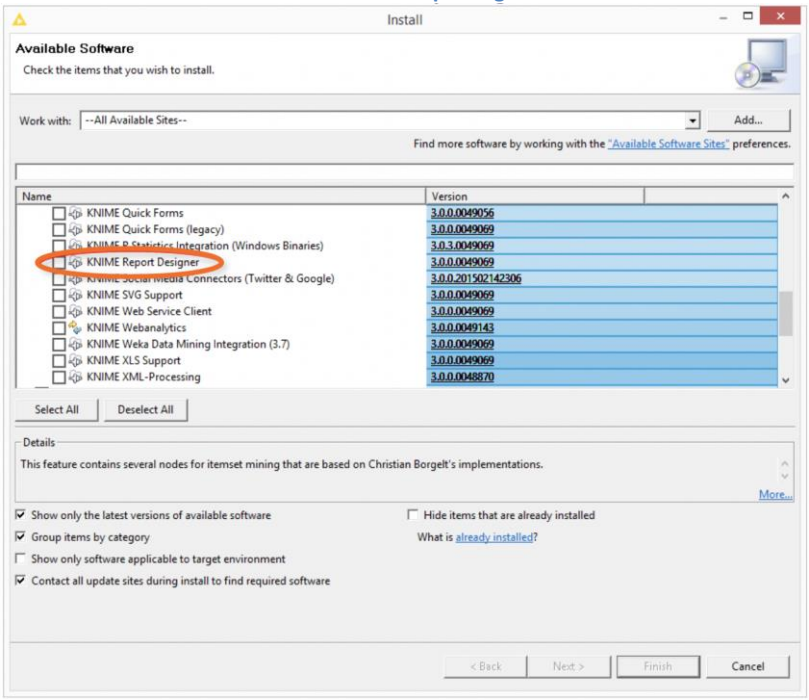
Select required Extensions

Click **“Next”** and follow instructions

1.24 . Exercise 2: List of KNIME Extensions



1.25. Exercise 2: Reporting 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 contains only one workflow group (folder) named "**Example Workflows**". This "Example Workflows" folder contains a number of sub-folders, each with basic workflows for very common use cases:

- ***Basic Examples***. Workflows in "*Basic Examples*" sub-folder show *basic general operations*, like import data, data blending, ETL, train and evaluate a model, and finally display results in a simple report.
- ***Customer Intelligence***. Basic workflows for *churn prediction, credit scoring, and customer segmentation* are available inside sub-folder "*Customer Intelligence*".
- ***Retail***. A *recommendation engine* is built in sub-folder "*Retail*".
- ***Social Media***. An example of *social media analysis* is available in "*Social Media*".

These example workflows can be reused and readapted for your own application. However, in this chapter we want to build our own very first workflow, to perform the following basic operations:

- Read data from a text file
- Filter out undesired rows
- Filter out undesired columns
- Write resulting data to a CSV file

We will use this first workflow to explore data structures and data types, node and workflow commands, debugging and data inspection possibilities, commenting options, configuration windows and execution commands, and other features available inside the KNIME workbench.

In order to keep our space clean, we use workflow groups to organize workflows by chapter or topic. Let's create now a new workflow group and call it "Chapter2". 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 should see workflow group "Chapter2" with a workflow named "My First Workflow" in it. For now, "My First Workflow" is an empty workflow. Indeed, if you double-click it, the workflow editor opens to an empty page.

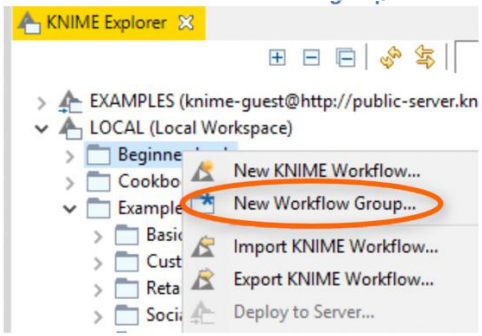
Let's see now how to perform some workflow operations, including creating, saving, and deleting a workflow.

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”

2.1. Create a new workflow group

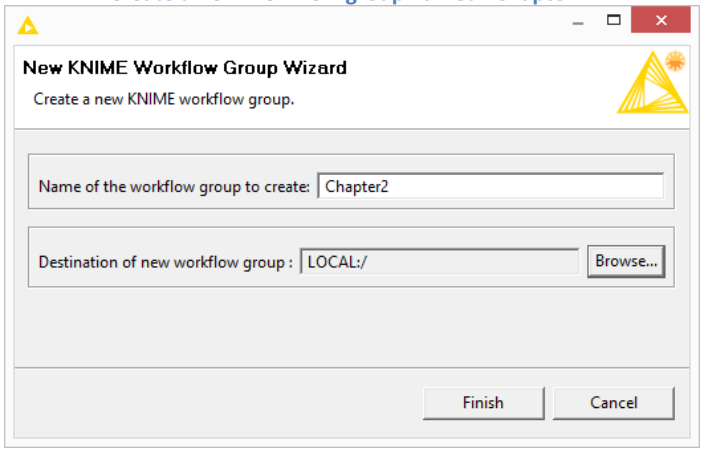


In the “New KNIME Workflow Group Wizard” dialog:

- Enter the name of the workflow group
- Enter its destination within the KNIME Explorer panel. To visualize the possible workflow group destinations, click the “Browse” button
- Click “Finish”

Note. If you select an existing workflow group in KNIME Explorer, right-click, and start the “New KNIME Workflow Group Wizard”, the default destination will be the selected workflow group.

2.2. Create a new workflow group named "Chapter2"

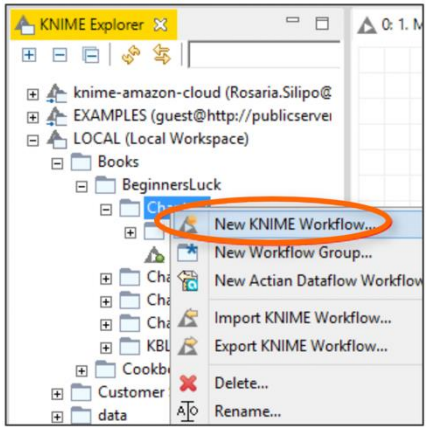


Create a new workflow

In the “KNIME Explorer” panel:

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

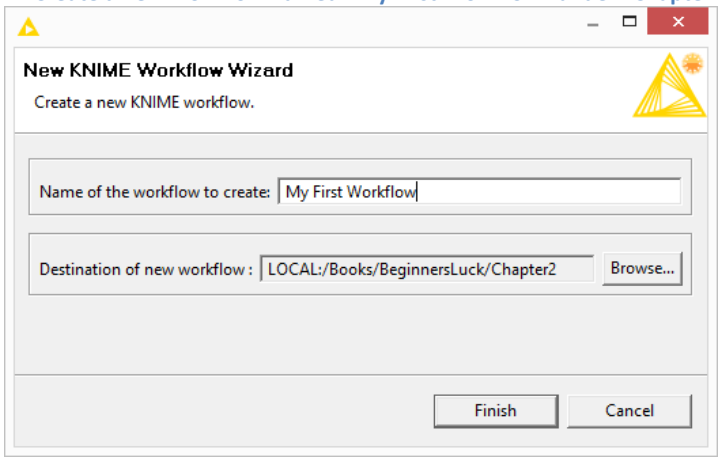
2.3. Create a new workflow



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 if needed
- Click “Finish”

2.4. Create a new workflow named "My First Workflow" under "Chapter2"



Note. If you select an existing workflow group in KNIME Explorer, right-click, and start the “New KNIME Workflow Wizard”, the default destination will be the selected workflow group.

To open a workflow, just double-click the workflow in the KNIME Explorer

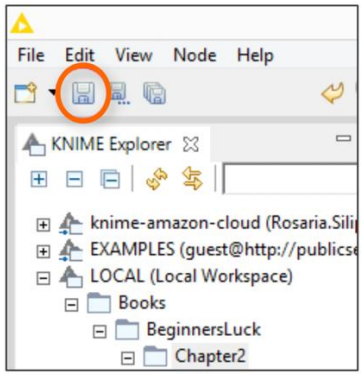
Save a workflow

To save a workflow, click the disk icon in the Top Menu. This only saves the selected workflow open in the workflow editor. Saving the workflow saves the workflow architecture, the nodes' configuration, and the data produced at the output of each node.

If you want to save a copy of the currently selected workflow, you need to click the "Save as …" disk icon on the right to the "Save" single disk icon.

If you want to save ALL open workflows and not only the selected one, you need to click the "Save All" stack of disks icon on the right to the "Save as …" disk icon.

2.5. "Save", "Save as…", and "Save all" options to save workflows



Delete a workflow

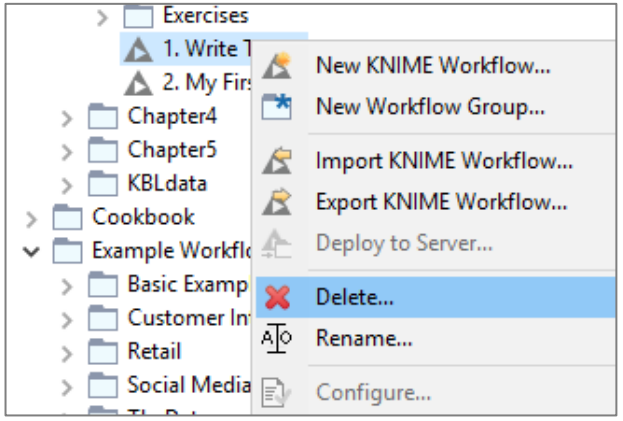
To delete a workflow

- 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.

Beware! The "Delete" command removes the workflow project physically from the hard disk. Once it is deleted, there is no way to get it back.

2.6. 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, organized by categories, in the "Node Repository" panel in the lower left corner of the KNIME workbench. 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 (final status = inactive, not configured; **red light**), how to configure the node (final status = configured, not executed; **yellow light**), and how to execute the node (final status = successfully executed; **green light**).

Create a new node

To create a new node, you have two options:

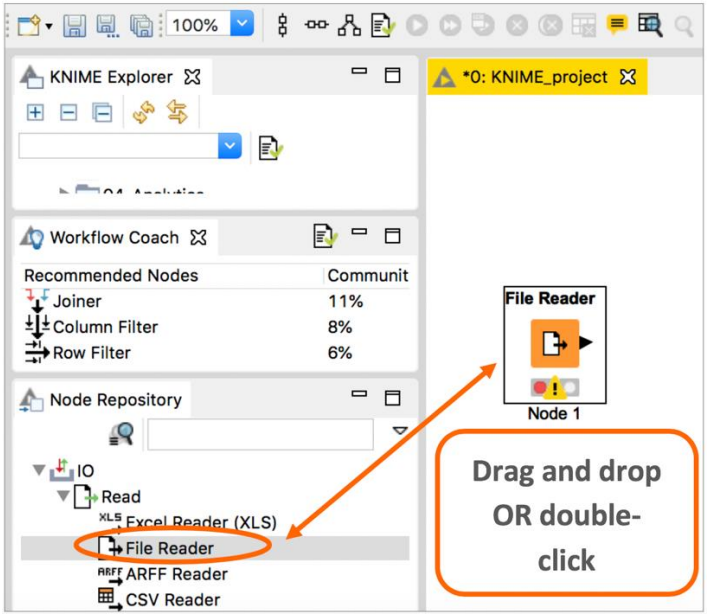
- drag and drop the node from the "Node Repository" panel into the workflow editor
- double-click the node in the "Node Repository" panel

The node is usually imported with red traffic light status.

To connect a node with existing nodes, there are two more options:

- click the output port of the first node and release the mouse at the input port of the second node
- select a node in the workflow and double-click a node in the Node Repository: this creates a new node and automatically connects its first input port to the first output port of the existing node. Shift + double clicking the new node moves the connection to the next input port.

2.7. Drag and drop or double-click the node to create a new node in the workflow editor



Once the node has been created, we need to configure it, i.e. to set the parameters needed for the node task to be executed.

Let's then open the node configuration window and let's configure the node.

Configure a node

To configure an existing node:

- Double-click the node

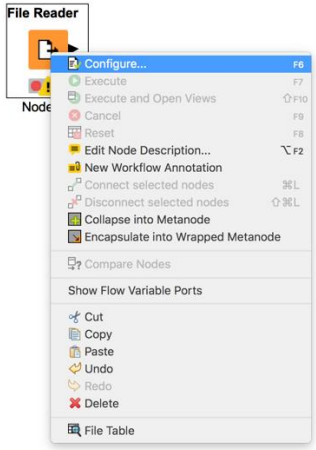
OR

- 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.

2.8. Right-click the node and select "Configure" or double-click the node to configure it



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.

To execute a node and let it run its task:

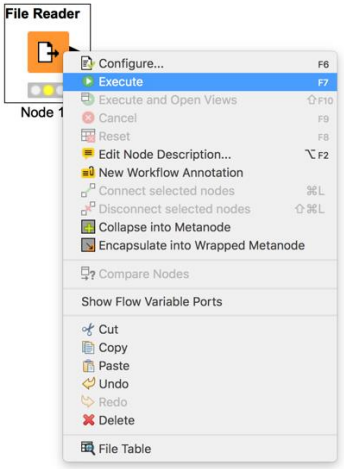
- Right-click the node & select “Execute”

OR

- Select the node and click the single green arrow in the tool bar

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

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



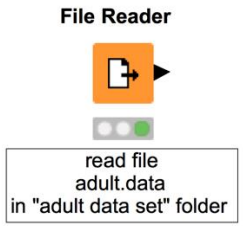
Finally, we need to associate a meaningful description to this node for documentation purposes, to easily recognize which task it is performing inside the workflow. Each node is created with a default text underneath as “Node n”, where “n” is a progressive number. This node text can be customized. This, together with the workflow annotations described in chapter 1, keeps the overview of the workflow clear and fulfills the purpose of workflow documentation.

Node Text

In order to change the text located under the node:

- Double-click the node text, so that it becomes editable
- Write the new text. The text can span more lines, separated by “Enter”
- Click outside the node to commit the text change

2.10. Double-click the node name to edit it

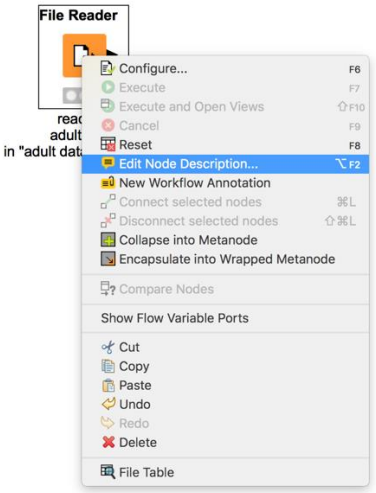


Node Description

Besides the node text, you can also insert a quick hidden 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
  - In the field “Custom Description”, write the node description
  - Click “OK”

2.11. Option “Edit Node Description” to insert a hidden node description



View the processed data

If the execution was successful (**green light**), you can inspect the processed data.

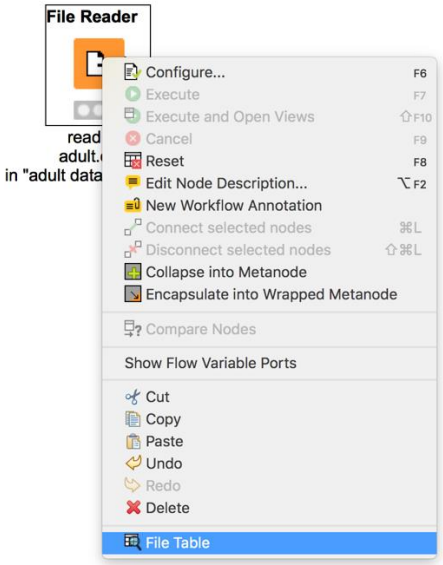
- Right-click the node
- Select the last option in the context menu
- The data table with the processed data should then appear.

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 depending on their task.

Some nodes might produce more than one output data set. In this case, there is more than one view item in the lowest part of the node context menu.

Nodes present as many output ports (in this case, a black triangle) as many output datasets are produced.

2.12. 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 analytics projects consists of reading data. Local data is usually read from a file or from a database.

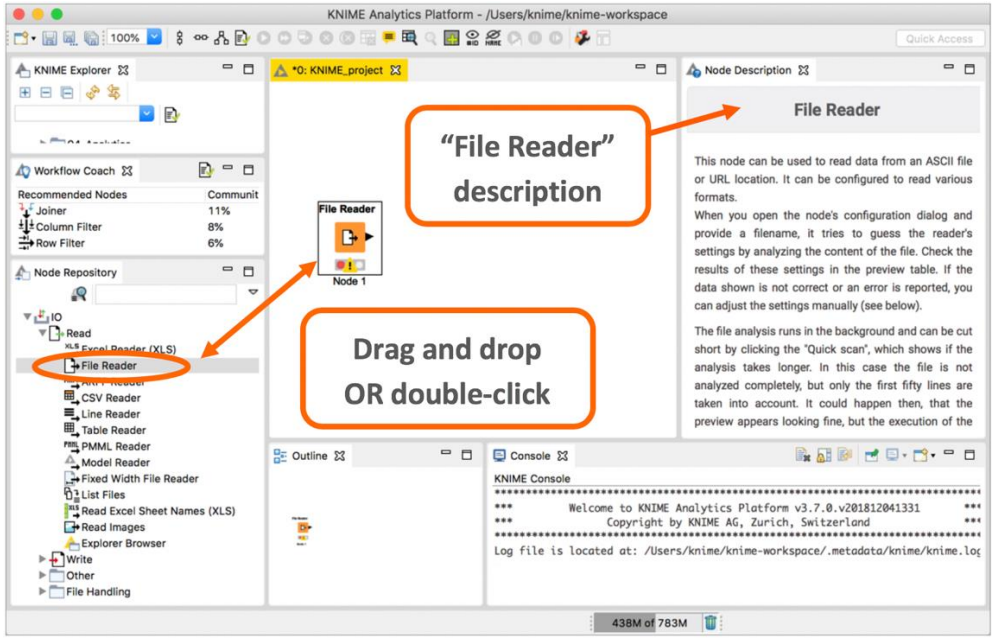
In this chapter we describe how to read and write data from and to a text file. Reading and writing data from and to a database is described in Chapter 3 in section “Database Operations”.

Create a “File Reader” node

In the “Node Repository” panel in the bottom left corner:

- Expand the “IO” category and then the “Read” sub-category OR alternatively type “File Reader” in the search box
- Drag and drop the “File Reader” node into the workflow editor (or double-click it)
- If the “Node Description” panel on the right is enabled, it shows all you need to know about the “File Reader” node: task, output port, and required settings.
- To activate the “Node Description” panel, go to the Top Menu, open “View” and select “Node Description”.

2.13. Create a "File Reader" node



Note. Under 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 file path!). 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”

In the configuration dialog:

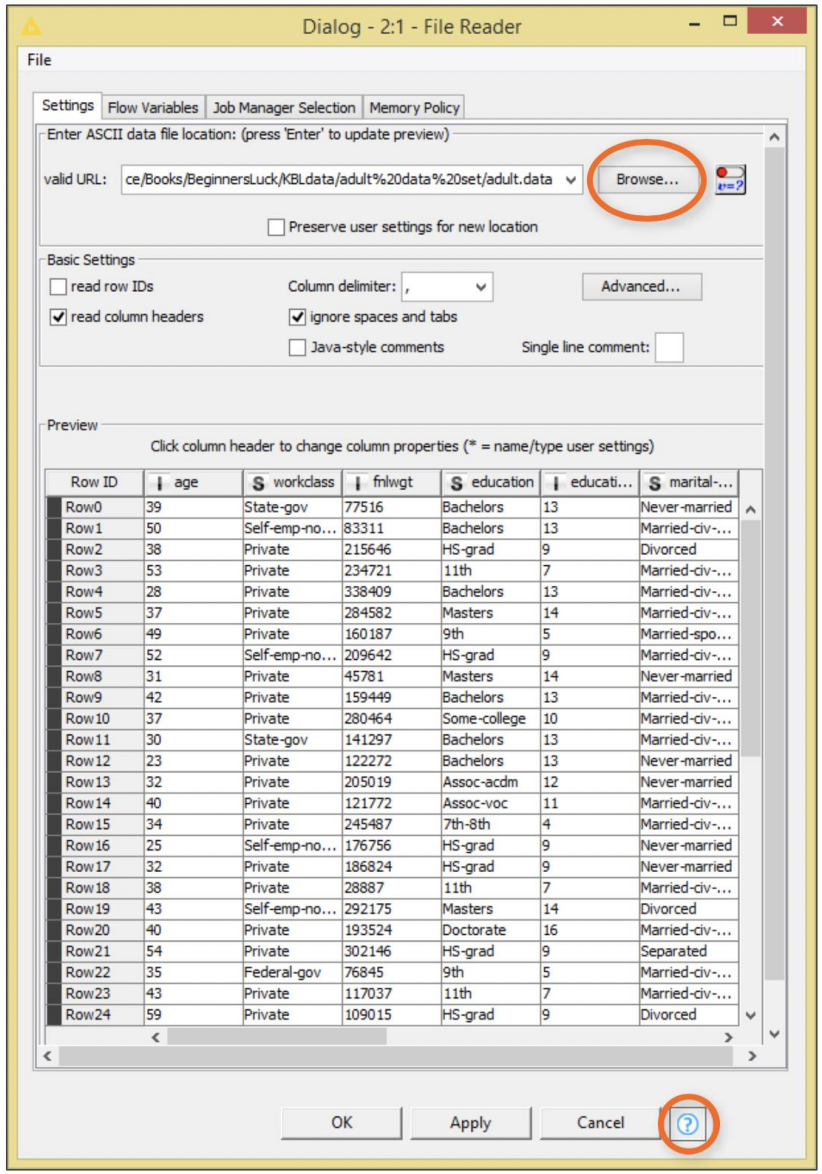
- Specify the file path, by typing or by using the “Browse” button. For this example, we used the adult.data file, downloadable from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Adult) or available in KBLdata/adult data set/adult.data.

- In most cases, the “File Reader” node automatically detects the file structure.
- If this is not one of most cases and the File Reader node has not guessed the file structure exactly, then enable/disable all required checkboxes according to the file data structure.

- A preview of the read data is available in the lower part of the configuration window and shows possible reading errors.

On the right of the OK/Cancel buttons in the lower part of the window, there is a small button carrying a question mark icon. This is the help button and leads to a new window containing the node description.

2.14. "File Reader" node configuration window



Customizing Column Properties

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

For example, the adult.data file contains a field unclearly named “fnlwgt”. We can change this column header to a more meaningful “final weight”.

In order to change a column’s header - or other column properties - during reading, follow these steps.

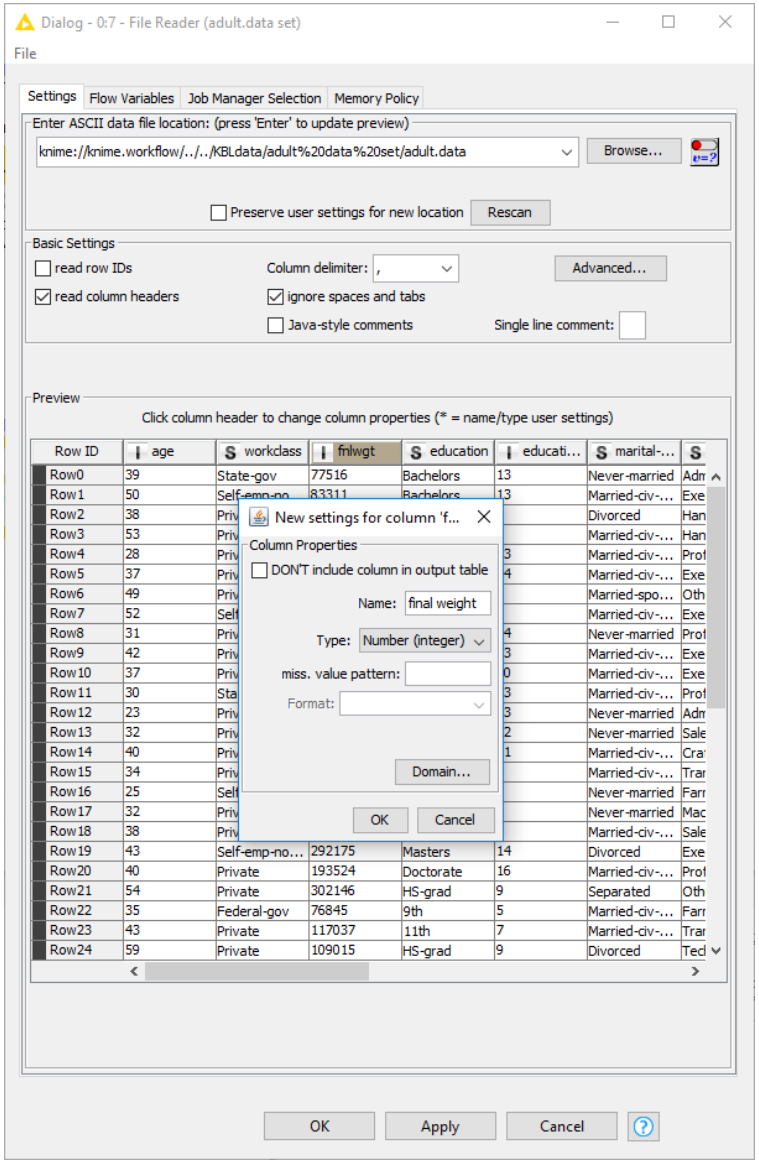
In the Preview pane in the File Reader configuration window

- Click the column header you want to change (in this case “fnlwgt”)
- The sub-window to customize the column properties opens

In the “New Settings for column …” sub-window, you can:

- Change the column name. In this case, you would change the Name text from “fnlwgt” to “final weight”
- Change the column data type (Integer, String, Double, …)
- Introduce a special character to represent missing values (default character is “?”)
- Add a format pattern for date & time columns

2.15. Customize how to read columns



Advanced Reading Options

Under the “Advanced” button, you will find some more reading option tabs. Each option tab is extensively described in the description view. Here some comments are provided only for three of the option tabs. Notice that to read the adult.data file you do not need to enable any of those options, as the adult.data file is pure ASCII.

“Quote Support” allows for 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.

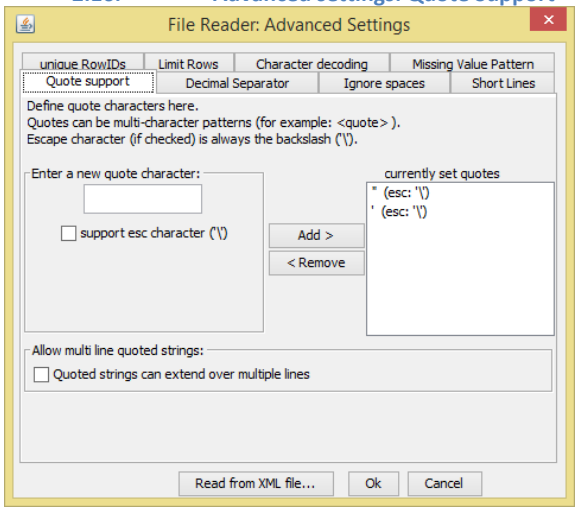
“Limit Rows” allows to read/skip a limited number of rows.

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 not interesting, like in a log file.

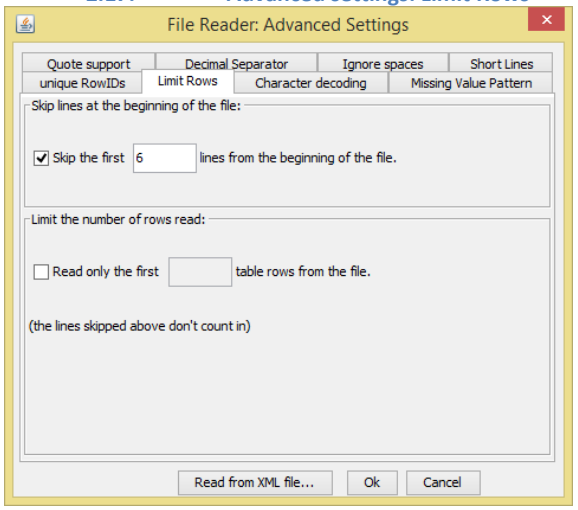
“Character Decoding” offers the possibility of reading files encoded with a different locale than just ASCII (default).

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

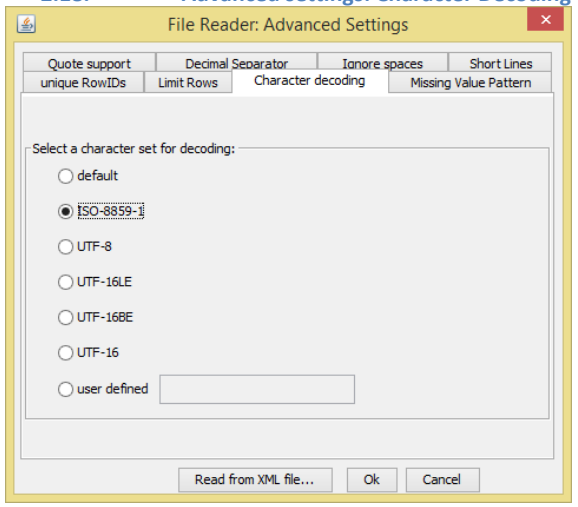
2.16. Advanced settings: Quote Support



2.17. Advanced settings: Limit Rows



2.18. Advanced settings: Character Decoding



Once you are done setting advanced options, click “OK” at the bottom of the “Advanced Settings” window.

If you are satisfied with the data preview, then 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.

What we have described here is the long way to get a File Reader node configured! However, since this (node creation and configuration) is a way that works for all nodes, we could not avoid going through it. However, if the file has a known extension, such as .csv or .txt, there might be a faster way.

Note. Instead of manually writing the full file path in the valid URL field in the File Reader configuration window, you can just drag and drop the file from the KNIME Explorer panel into the workflow editor. If the file has a known extension (like .csv for example), this automatically creates the appropriate reader node and configures it.

Notice that when you create a File Reader this way, you will get a different protocol in the URL box. By browsing to a file, you get a *file://* protocol; by drag and drop you get a *knime://* protocol.

The *knime://* protocol

The *knime://* protocol is a special protocol that allows you to reference the local workspace or the local workflow in a path. This then permits the creation of relative paths making you independent of the workspace folder absolute URL, but just dependent on the workspace folder structure.

This feature is particularly useful when moving workflows around to other workspaces or even to other machines. As long as the folder structure of data and workflow is preserved, the File Reader will keep finding the file and reading it.

The knime:// protocol works on all reader and writer nodes.

2.19. Possible paths using knime:// protocol

knime://LOCAL/	refers to the current workspace location
knime://LOCAL/../../knime-workspace	moves two levels up from the current workspace location to a new workspace folder named knime.workspace
knime://knime.workflow/	refers to the current workflow location
knime://knime.workflow/../../data	moves two levels up from the current workflow location to a new folder named data
knime://<knime-mountID>/	refers to a KNIME Server available in the KNIME Explorer panel
knime://<knime-mountID>/<path>/data	moves to the <path>/data folder on the referenced KNIME Server

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

- Double-click the “Node 1” label under the “File Reader” node
- Enter the node’s new comment (for example “Adult data set”)
- Click elsewhere

We have now changed the comment under the “File Reader” node from “Node 1” to “Adult data set with file:/ protocol”.

After configuration, in order to make the node really read the file, we need to execute it. Thus, 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 parameters 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 functionalities.

The “IO” -> “Read” category in “Node Repository” contains a number of additional nodes to read files in different formats, like Excel, CSV, KNIME proprietary format, and more. The “IO”/”File Handling” category has nodes to read special formats and special files, like for example ZIP files, remote files, etc.

2.4. KNIME data structure and data types

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 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. 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 Analytics Platform adds a suffix "(#n)" (n = progressive number) to each multiple occurrence of the column header.

Each column contains data with a set **data type**. Available data types are:

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

Date&Time type can come from importing data from a database. It does not appear from reading data from a file. In text files, dates and times are read just as strings. You then need a "String to Date&Time" node to convert a String into a Date&Time type column.

Unknown type refers to columns whose type could not be determined, like for example with mixed data types or 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 with question marks. They are not question marks, they are missing and are represented with question marks. Question marks in the text file are correctly 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 are organized as a **table** with a fixed number of columns.

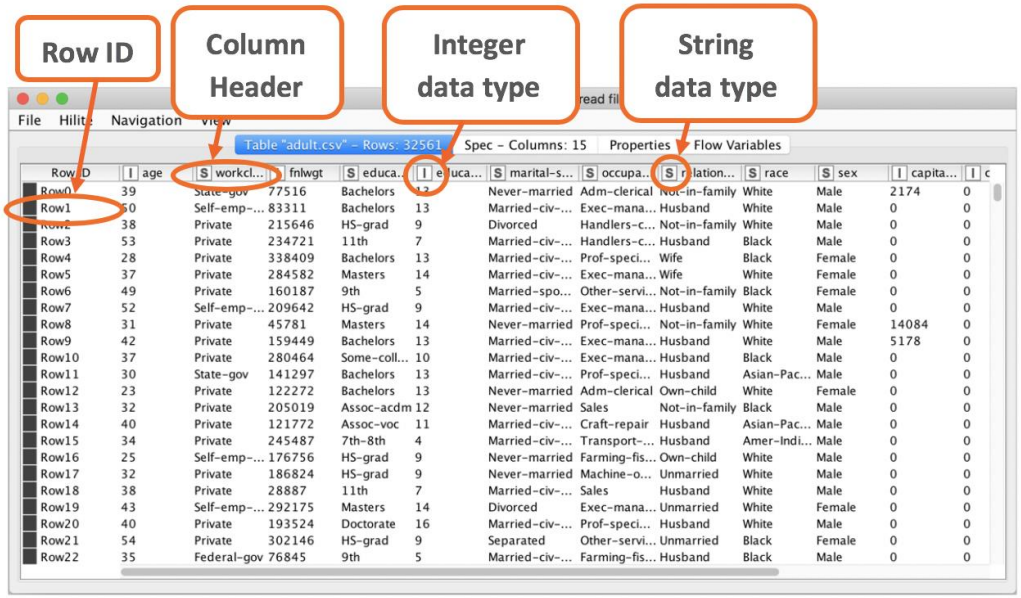
Each row is identified by a **Row ID**.

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

2.20. The KNIME data structure



Row ID	age	workcl...	fnlwgt	educa...	marital-s...	occupa...	race	sex	capita...
Row0	39	State-gov	77516	Bachelors	Never-married	Adm-clerical	White	Male	2174
Row1	50	Self-emp-...	83311	Bachelors	Married-civ-...	Exec-mana...	White	Male	0
Row2	38	Private	215646	HS-grad	Divorced	Handlers-c...	White	Male	0
Row3	53	Private	234721	11th	Married-civ-...	Handlers-c...	Black	Male	0
Row4	28	Private	338409	Bachelors	Married-civ-...	Prof-speci...	Black	Female	0
Row5	37	Private	284582	Masters	Married-civ-...	Exec-mana...	White	Female	0
Row6	49	Private	160187	9th	Married-spo...	Other-servi...	Black	Female	0
Row7	52	Self-emp-...	209642	HS-grad	Married-civ-...	Exec-mana...	White	Male	0
Row8	31	Private	45781	Masters	Never-married	Prof-speci...	White	Female	14084
Row9	42	Private	159449	Bachelors	Married-civ-...	Exec-mana...	White	Male	5178
Row10	37	Private	280464	Some-coll...	Married-civ-...	Exec-mana...	Black	Male	0
Row11	30	State-gov	141297	Bachelors	Married-civ-...	Prof-speci...	Asian-Pac...	Male	0
Row12	23	Private	122272	Bachelors	Never-married	Adm-clerical	Own-child	White	Female	0
Row13	32	Private	205019	Assoc-acdm	Never-married	Sales	Not-in-family	Black	Male	0
Row14	40	Private	121772	Assoc-voc	Married-civ-...	Craft-repair	Husband	Asian-Pac...	Male	0
Row15	34	Private	245487	7th-8th	Married-civ-...	Transport-...	Husband	Amer-Indi...	Male	0
Row16	25	Self-emp-...	176756	HS-grad	Never-married	Farming-fis...	Own-child	White	Male	0
Row17	32	Private	186824	HS-grad	Never-married	Machine-o...	Unmarried	White	Male	0
Row18	38	Private	28887	11th	Married-civ-...	Sales	Husband	White	Male	0
Row19	43	Self-emp-...	292175	Masters	Divorced	Exec-mana...	Unmarried	White	Female	0
Row20	40	Private	193524	Doctorate	Married-civ-...	Prof-speci...	Husband	White	Male	0
Row21	54	Private	302146	HS-grad	Separated	Other-servi...	Unmarried	Black	Female	0
Row22	35	Federal-gov	76845	9th	Married-civ-...	Farming-fis...	Husband	Black	Male	0

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.

You can temporarily sort the data by clicking the column header and select the kind of sorting. You can temporarily change the data renderer by right-clicking the column header and change to a different renderer either numerical or bar based. Both those operations are just temporary. If you close the data table and reopen it, the default window will be back.

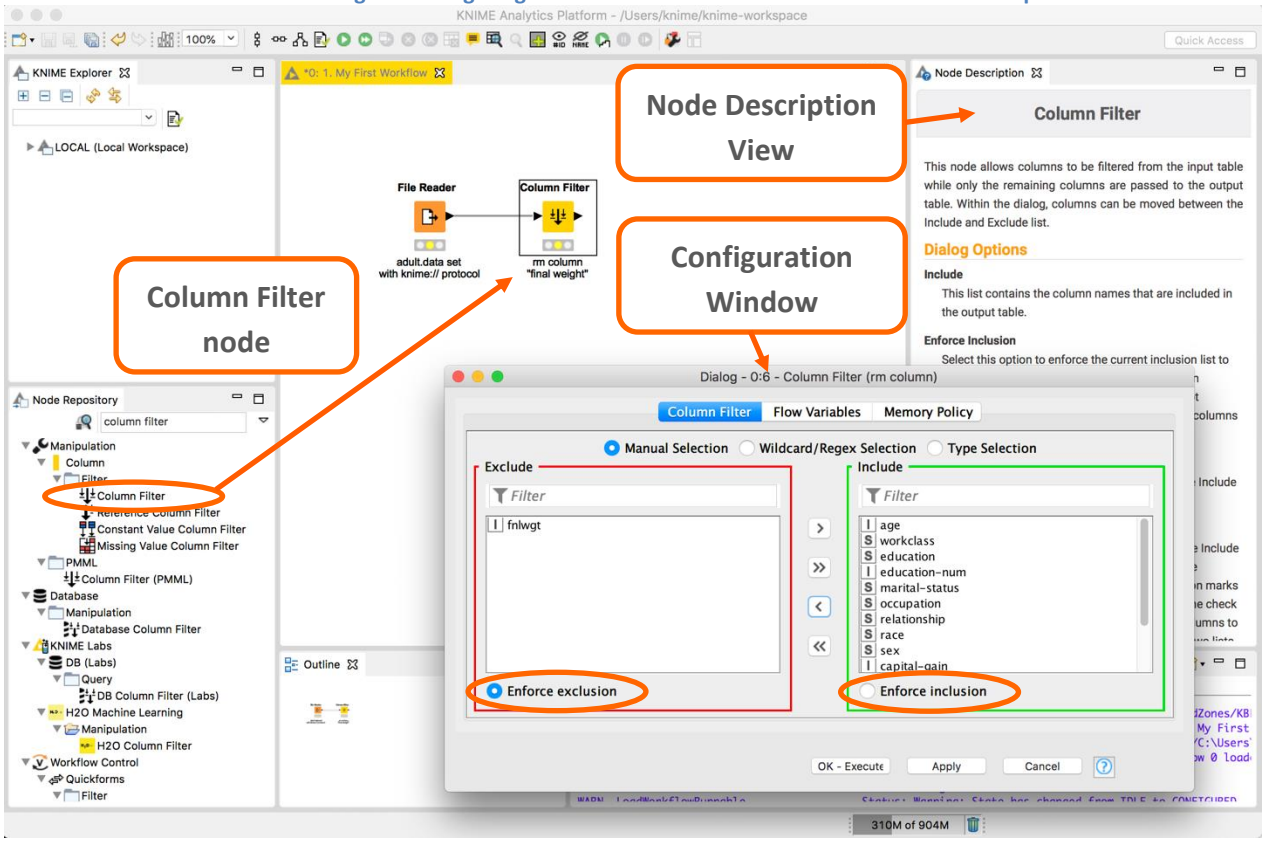
2.5. Filter Data Columns

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, find the node “Column Filter” under `“Data Manipulation” -> “Column” -> “Filter”` or search for “Column Filter” in the search box.
- Drag and drop the “Column Filter” node from the “Node Repository” to the workflow editor or double-click it in the “Node Repository”
- 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) by clicking at the output of the “File Reader” node and releasing at the input of the “Column Filter” node.

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



To configure the node:

- 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. You can select and retain columns **manually**, **by type**, or **by name**, according to the radio buttons at the top of the configuration window (Fig. 2.21).

Manual Selection

If the “Manual Selection” option is selected, the configuration window shows 2 sets of columns (Fig. 2.21):

- 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 a specific column.

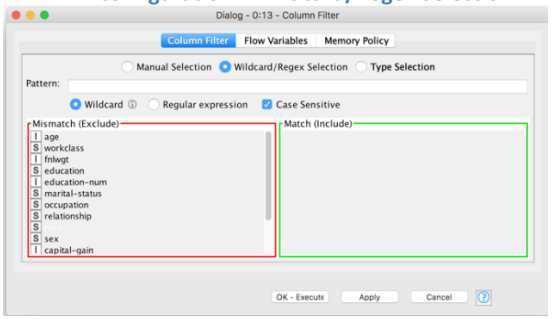
You can add and remove columns from one set to the other using the buttons **“Add”** and **“Remove”**.

- **“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.
The radio buttons below the textbox specify 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.

2.22. "Column Filter" node configuration: “Wildcard/Regex Selection”

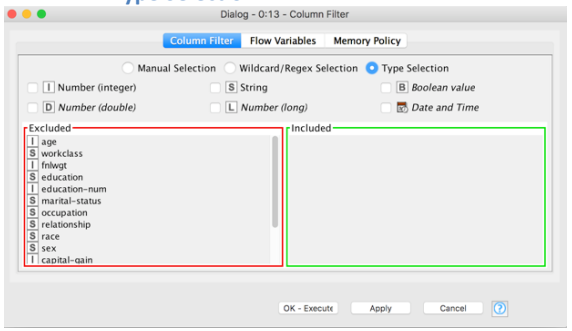


Type Selection

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

Selecting all Numbers checkboxes, for example, will keep all numerical columns only in the node’s output table.
Selecting String will keep all String type columns only in the output table.

2.23. "Column Filter" node configuration: “Type Selection”



Remember this column selection frame that comes with the “Manual Selection” option, because it will show up again in all those nodes requiring column selection.

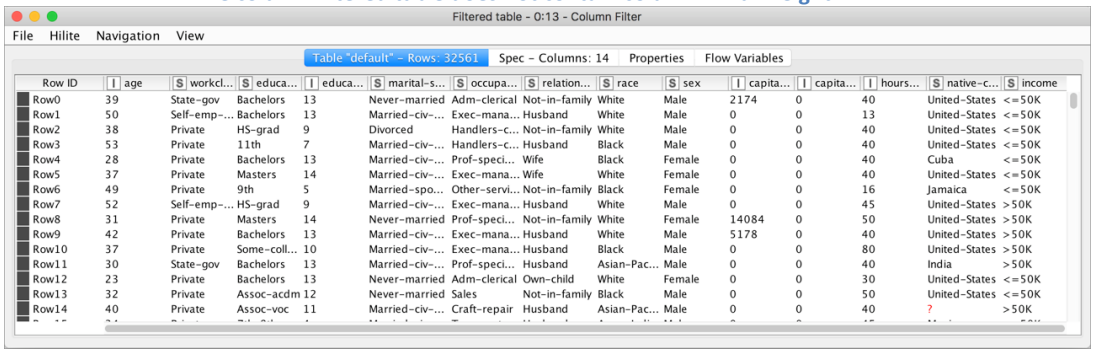
In our example workflow “my First Workflow”, we wanted to remove the “final weight” column.

We set the column filter mode to “Manual Selection” and we populated the Exclude panel with column “final weight”. We then enabled “Enforce Exclusion”, because we wanted to keep all new input columns, if any, and always exclude just “final weight”.

After completing the configuration, we right-clicked the “Column Filter” node and commented it with “rm column ‘final weight’”.

We finally right-clicked the node and selected “Execute” to run the column filter.

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



Row ID	age	workcl...	educa...	educa...	marital-s...	occupa...	relation...	race	sex	capita...	capita...	hours...	native-c...	income
Row0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
Row1	50	Self-emp-...	Bachelors	13	Married-civ-...	Exec-mana...	Husband	White	Male	0	0	13	United-States	<=50K
Row2	38	Private	HS-grad	9	Divorced	Handlers-c...	Not-in-family	White	Male	0	0	40	United-States	<=50K
Row3	53	Private	11th	7	Married-civ-...	Handlers-c...	Husband	Black	Male	0	0	40	United-States	<=50K
Row4	28	Private	Bachelors	13	Married-civ-...	Prof-speci...	Wife	Black	Female	0	0	40	Cuba	<=50K
Row5	37	Private	Masters	14	Married-civ-...	Exec-mana...	Wife	White	Female	0	0	40	United-States	<=50K
Row6	49	Private	9th	5	Married-spo...	Other-servi...	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
Row7	52	Self-emp-...	HS-grad	9	Married-civ-...	Exec-mana...	Husband	White	Male	0	0	45	United-States	>50K
Row8	31	Private	Masters	14	Never-married	Prof-speci...	Not-in-family	White	Female	14084	0	50	United-States	>50K
Row9	42	Private	Bachelors	13	Married-civ-...	Exec-mana...	Husband	White	Male	5178	0	40	United-States	>50K
Row10	37	Private	Some-coll...	10	Married-civ-...	Exec-mana...	Husband	Black	Male	0	0	80	United-States	>50K
Row11	30	State-gov	Bachelors	13	Married-civ-...	Prof-speci...	Husband	Asian-Pac...	Male	0	0	40	India	>50K
Row12	23	Private	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0	0	30	United-States	<=50K
Row13	32	Private	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0	50	United-States	<=50K
Row14	40	Private	Assoc-voc	11	Married-civ-...	Craft-repair	Husband	Asian-Pac...	Male	0	0	40	?	>50K

To see the final processed data, we right-clicked the “rm column “final weight”” node and selected option “Filtered Table”. The column “final weight” was not to be found in the Column Filter’s output data table.

2.6. Filter Data Rows

If you have had a deeper look into the data we are currently analyzing, you have seen that each record describes a person in terms of age, job, education, and other general demographic information. We have seen how to remove a data column from a data table. Let’s see now how to exclude data rows from a data table.

Let’s suppose that we want to retain all records of people born outside of the United States. That is, we want to retain only those rows with “native-country” other than “United States”. We need to use a Row Filter node.

Create a "Row Filter" node

In the "Node Repository" panel, 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 previously created.

Configure the "Row Filter" node

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

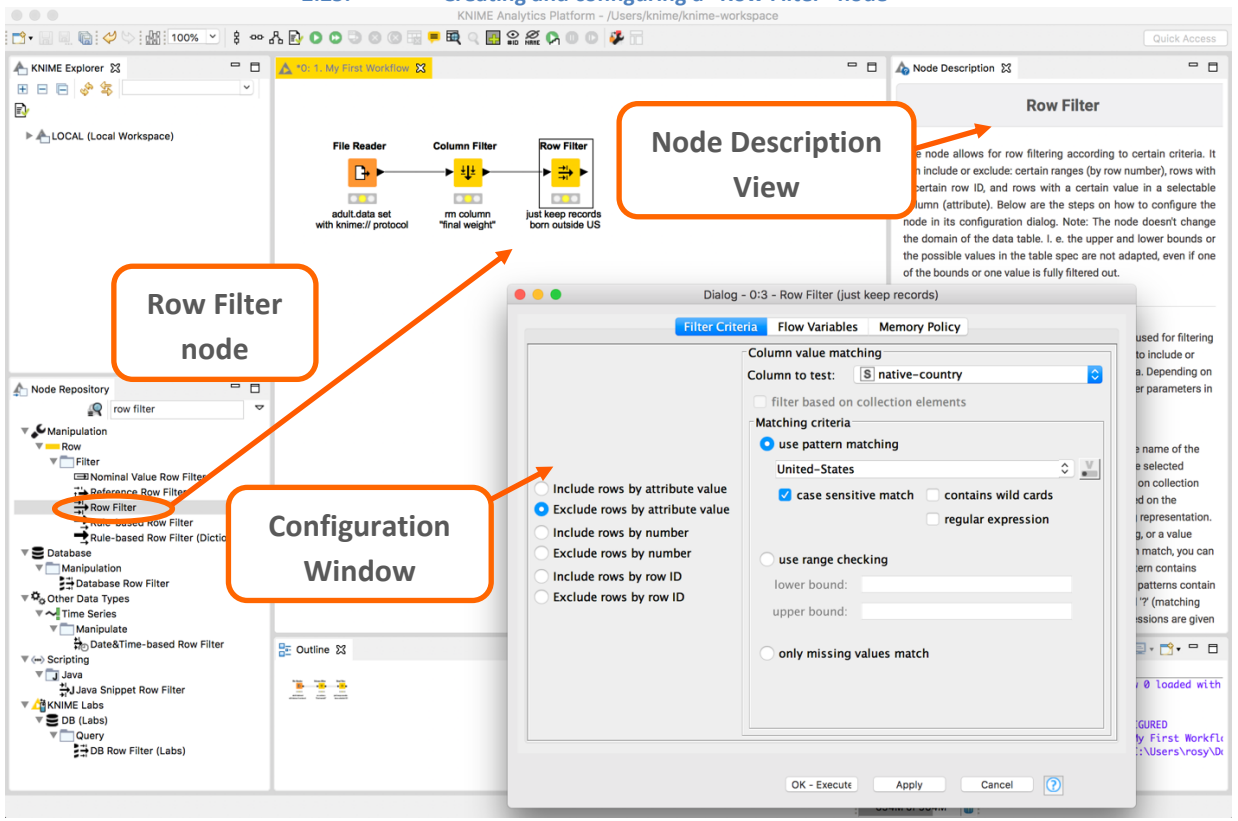
The node implements three filter criteria:

- Select rows **by attribute value** (pattern matching)
    - Value matching: column value matching some pre-defined pattern value (wild-cards and regular expression are allowed in pattern definition)
    - Range checking for numerical columns: column value above or below a given value
    - Missing Value Matching
- Select rows **by row number**
- Select rows **by RowID** (pattern matching on RowID)

Each of these criteria can be used to **include** or to **exclude rows**.

- Implement your row filter criterion
- Click "OK"

2.25. Creating and configuring a "Row Filter" node



Below you can find a more detailed description of the row filter criteria available in the Row Filter node configuration.

The Row Filter node is not the only way to perform row filtering in KNIME, even though probably it is the easiest one and works for 80% of your row filtering needs. Other row filtering options are offered by:

- "Nominal Value Row Filter" node for multiple pattern matching in "OR mode" (example: native-country = "United Stated" OR native-country="Canada" OR native-country="Puerto Rico");
- "Rule Based Row Filter" node to define an arbitrarily complex set of IF-THEN row filtering rules, even spanning multiple columns;
- "Geo-location Row Filter" node in KNIME Labs category for row filtering based on geographical coordinates;
- "Date&Time-based Row Filter" node to perform a row filtering on a Date&Time type column;
- "Database Row Filter" node to implement a row filtering SQL query to run directly on the database.

Row filter criteria

By attribute value

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

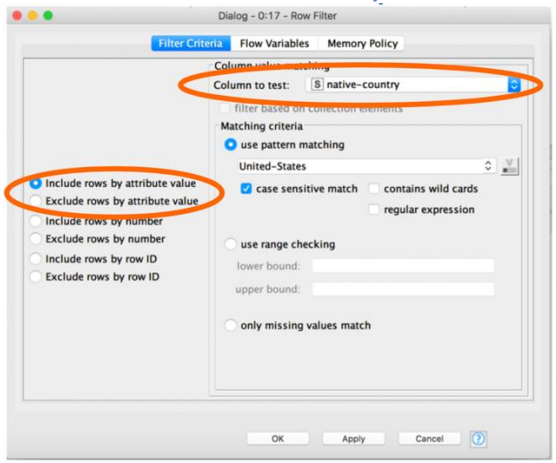
After you "select the column to test", you need to define the matching mode.

For **String/Integer/Date&Time values**, "use pattern matching" requires the given pattern to be either entered manually or selected from a menu populated with the column values as possible pattern values.
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 coincide if the condition is equality.

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

2.26. Row filter criterion by attribute value

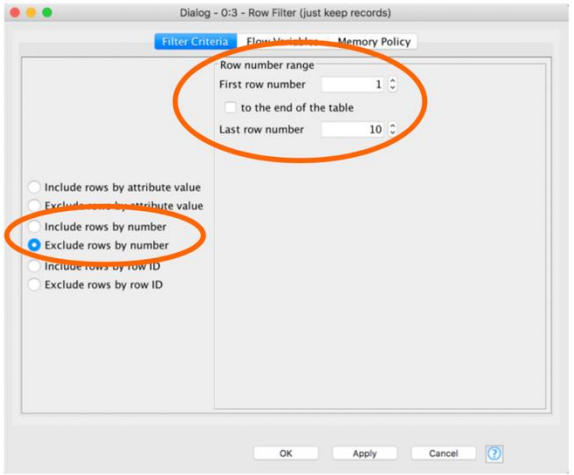


By row number

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 comments or just garbage, I would select the filter criterion "exclude row by number" and set the row number range 1-10.

2.27. Row filter criterion by row number

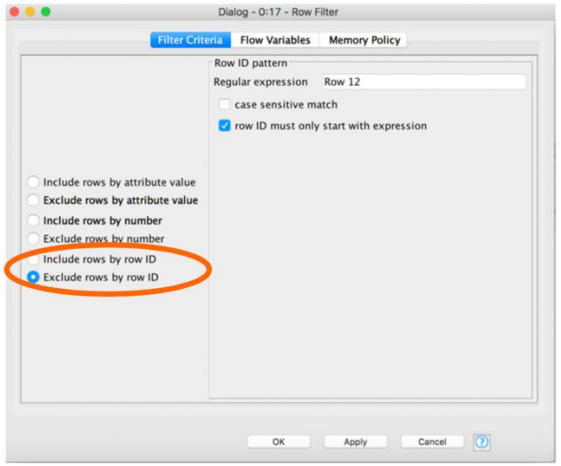


By RowID

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,

2.28. 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”`

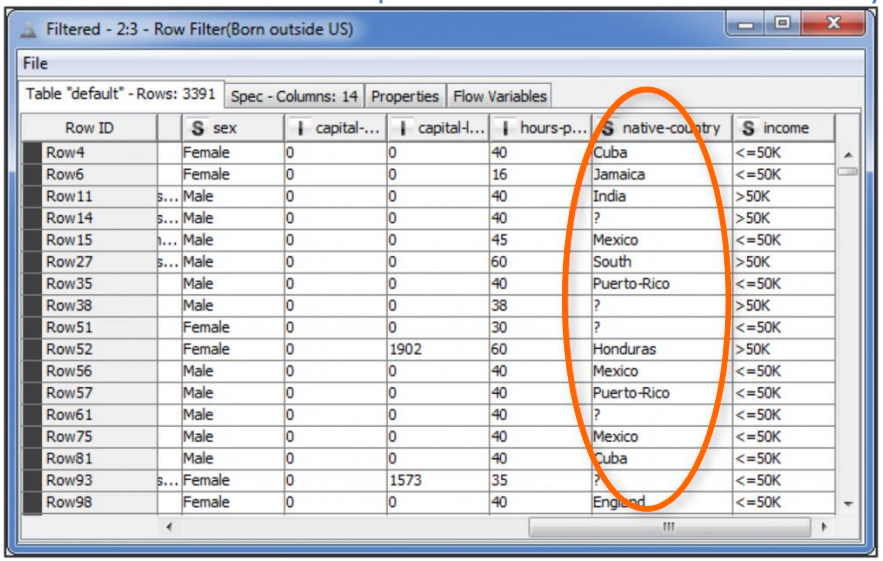
- Give the “Row Filter” node a meaningful comment. We commented it with “just keep records born outside US”. The comment on a node is important for documentation purposes. Since KNIME is a graphical tool, it is easy to keep an overview of what a workflow does, if the name of each node gives a clear indication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

2.29. 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	h...	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

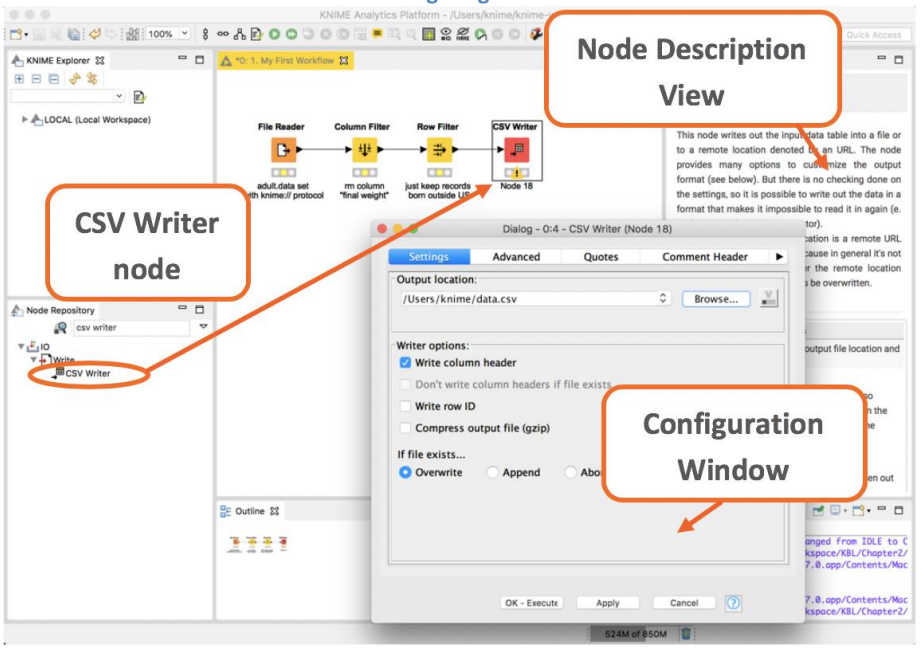
Now we want to write the processed data table to a file. There are many nodes that can write to a file. Let's choose the easiest and most standard format for now: the CSV (Comma Separated Values) format.

Create a "CSV Writer" node

In the "Node Repository":

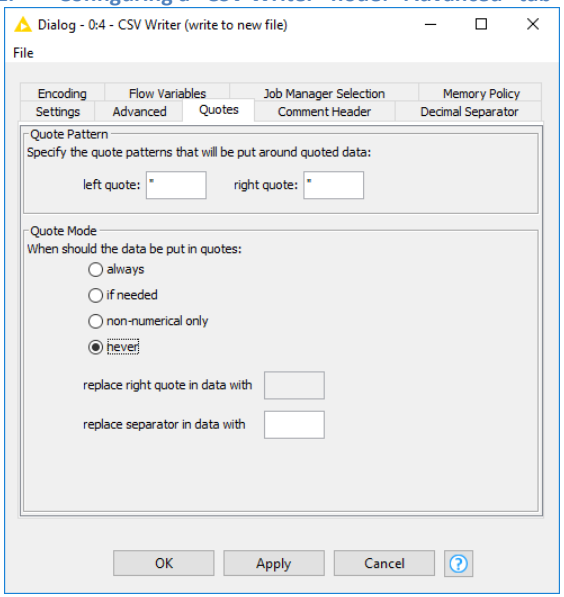
- Expand category "IO"/"Write" or search for "CSV Writer" in the search box
- Drag and drop (or double-click) the node "CSV Writer" to create a new node in the workflow editor
- If the "Node Description" panel on the right is on, it fills up with the description of the tasks and settings for the "CSV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

2.30. Create and configuring a "CSV Writer" node



Configure the “CSV Writer” node

2.31. Configuring a "CSV Writer" node: “Advanced” tab



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

- The path of the output file (the knime:// protocol is also supported)
- A few additional options about the data structure, such as:
    - Write column headers and/or RowID in output file
    - 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. In figure 2.31 the “Quotes” tab is shown. The default quote setting mode is to quote all string values. For most purposes you do not need that much quoting and you can set it to “never”. Therefore no values, either String or Integer or other type values, will be written in quotes in the output file.
- **“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.
- **“Encoding”** is to choose the appropriate encoding for the text.
- Tab **“Memory Policy”** contains a few options that might speed up the node execution. This tab is common to the configuration window of all nodes.
- In this book we do not investigate the tab “Flow variables” and “Job Manager Selection”.

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 will not be consistent anymore.

In some cases, you might want to select "Abort" as over-writing mode, in order to avoid overwriting the existing file.

Let's now change the node's comment:

- Click the node label under the node
- Enter the node's new comment (for example "write new file")
- Click elsewhere
- Right-click the node and select "Execute"

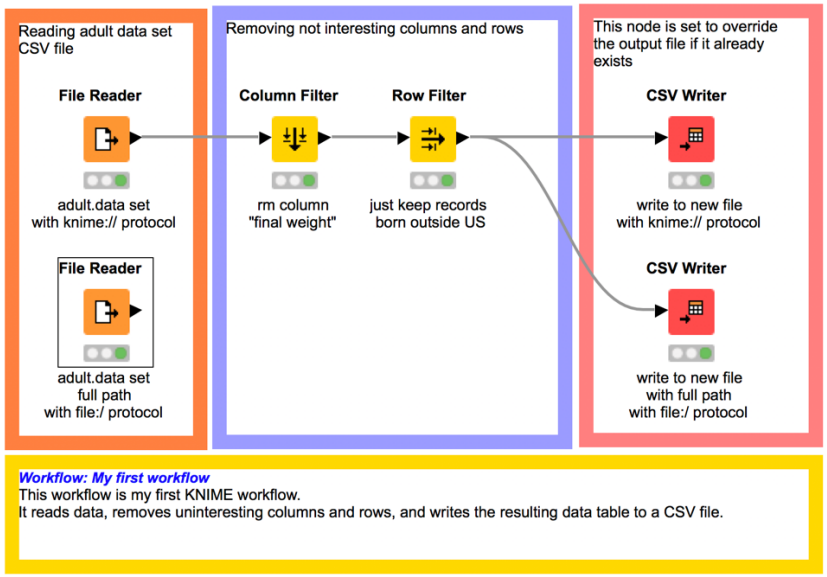
You can also make the node comments more verbose, if you want to add more information about the node settings and implemented task.

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.

In figure 2.32 is shown the final workflow named "My First Workflow".

2.32. 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

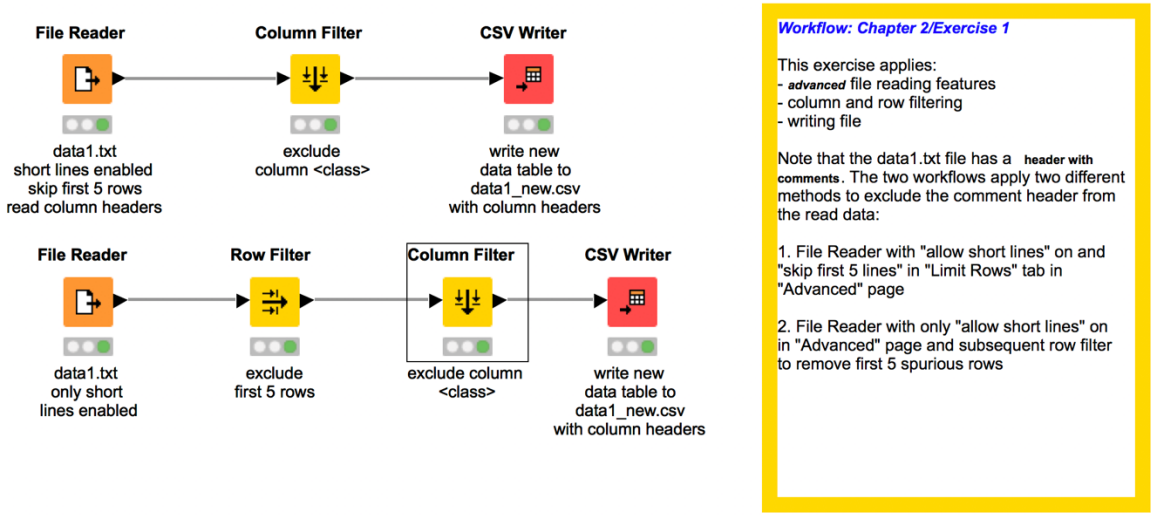
The file has some comments at the very beginning, which of course do not have the same length as the other lines in the file.

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

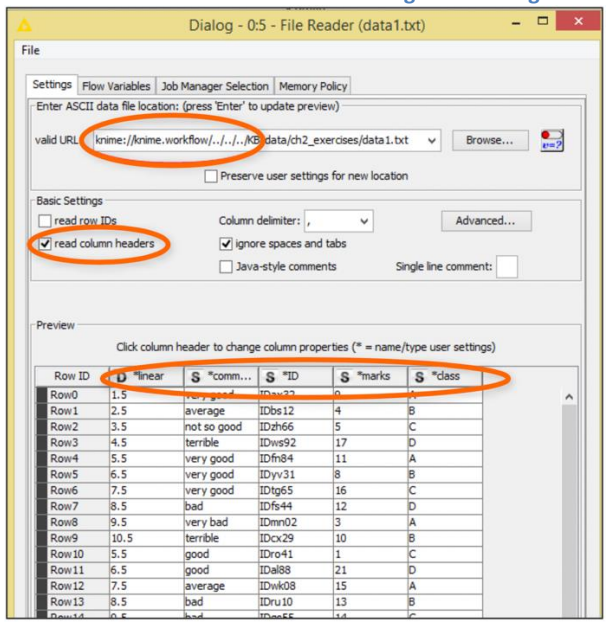
Then you can either enable the “Skip the first 6 lines …” in the “Limit Rows” tab of the “Advanced” window and the “read column headers” option in the basic configuration window or you can use a “Row Filter” node to exclude the first 5 rows of the read data. The difference between these two approaches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

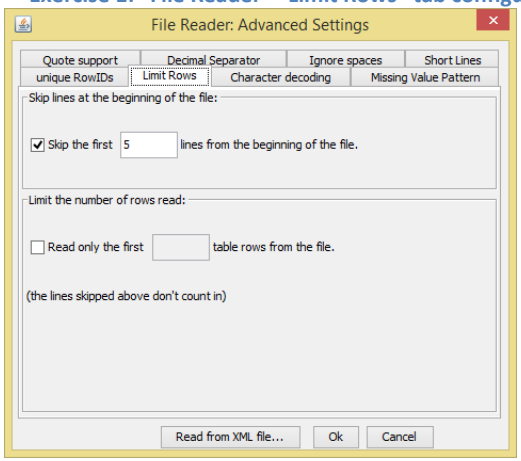
2.33. Exercise 1: Two possible solution workflows



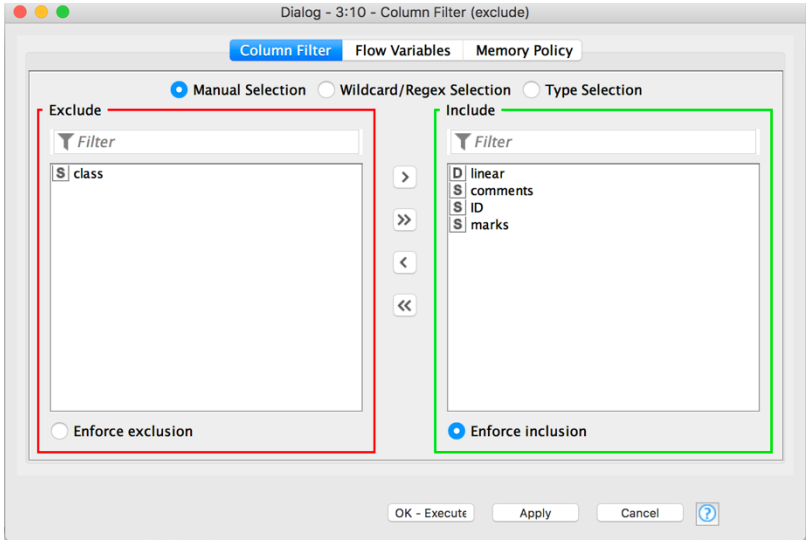
2.34. Exercise1: “File Reader” “Settings” tab configuration



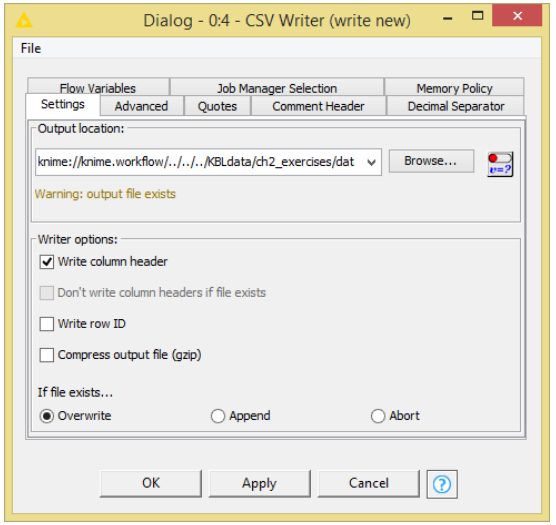
2.35. Exercise 1: "File Reader" “Limit Rows” tab configuration



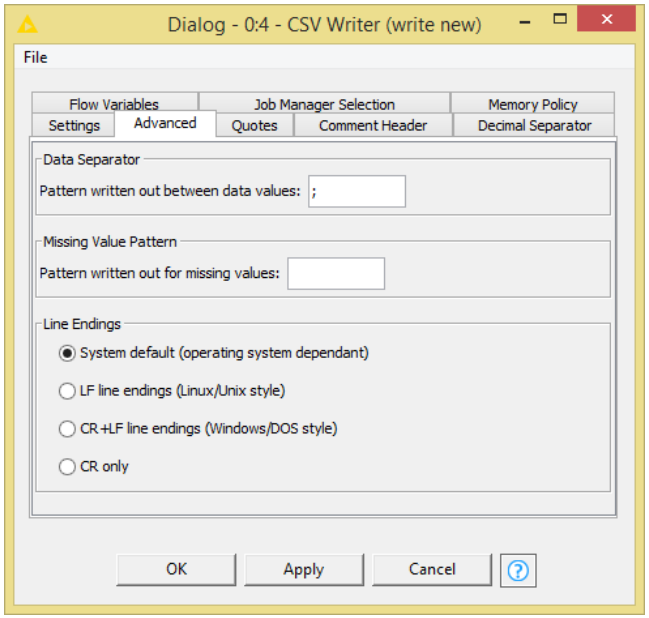
2.36. Exercise 1: “Column Filter” configuration



2.37. Exercise 1: “CSV Writer” “Settings” tab configuration



2.38. Exercise 1: “CSV Writer” “Advanced” tab configuration



Exercise 2

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

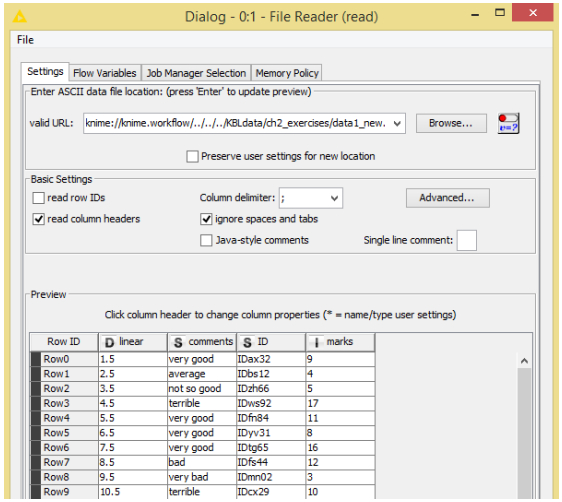
- Read the CSV file written in Exercise1 (“data1_new.csv”) and rename column “marks” to “ranking”
- Filter out rows with comments = “average” in data column “ranking”
- Exclude Integer type columns
- Write final data to file in “Append” mode and with a tab as a separating character

Rename all nodes where necessary. Save and execute workflow “Exercise2”.

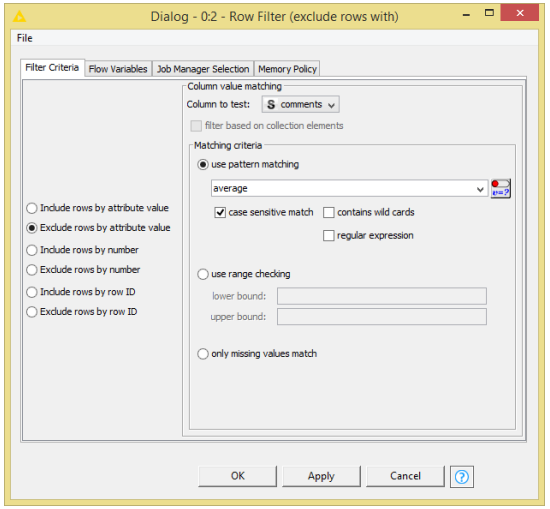
Solution to Exercise 2

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”.

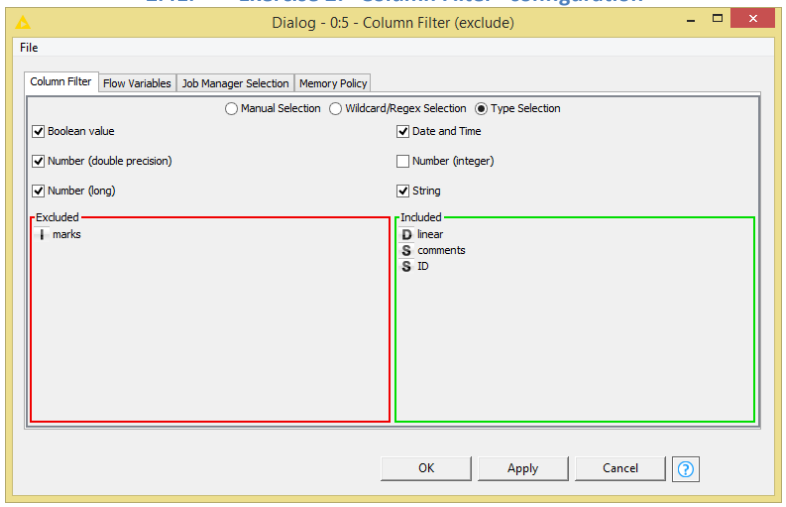
2.39. Exercise 2: “File Reader” “Settings” configuration



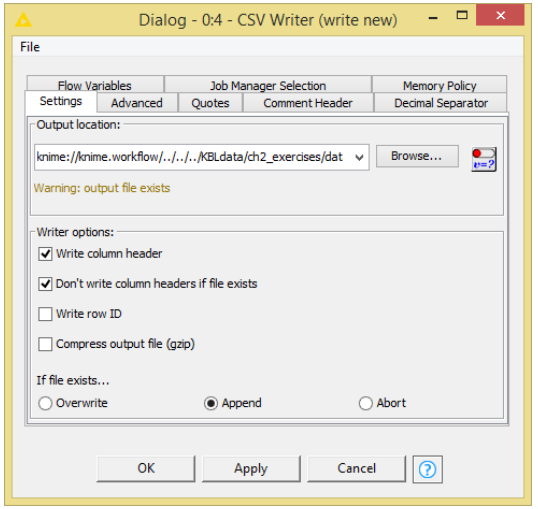
2.40. Exercise 2: “Row Filter” configuration



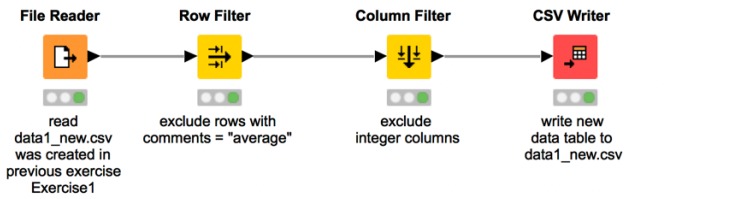
2.41. Exercise 2: “Column Filter” configuration

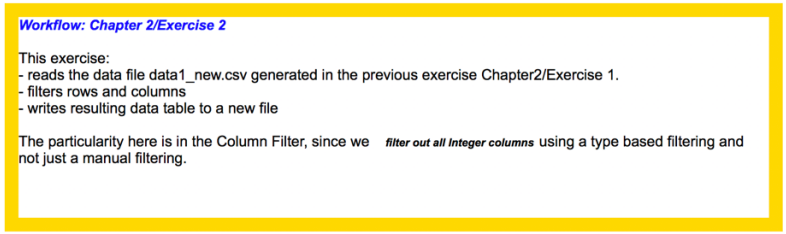


2.42. Exercise 2: “CSV Writer” configuration



2.43. Exercise 2: The workflow





Note. After copying the "File Reader" node 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: “Write To DB” and “My First Data Exploration”. “Write To DB” writes data into a database and “My First Data Exploration” reads the same data from the database and graphically explores the data.

The goal of this chapter is to become familiar with:

- nodes and options for database handling
- the “Views/Javascript” 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, downloaded from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Iris) and available under the KBLdata folder, to prepare the data for the next graphical exploration. 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 function on two of the four numeric attributes. For the third class we need to use something more sophisticated than a linear separation. The first two classes and their possible linear separation can be clearly identified 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 existing columns' values.

In the "KNIME Explorer" panel, we create now a new workflow group named "Chapter3", to contain all workflows created in this chapter of the book. Under workflow group "Chapter3" we create two empty workflows: "Write To DB" and "My First Data Exploration". As we said at the beginning of this section, "Write To DB" will show how to build a new dataset and how to write it into a database, while "My First Data Exploration" will describe how to perform a visual exploration of the data. The workflow group and its workflows can be found in the folder imported from the "Download Zone" file.

Let's start with shaping the "Write To DB" workflow.

After reading the Iris dataset file (iris.data) with a "File Reader" node, we get the data table in figure 3.1. We write the comment "read data from iris.data set from KBLdata folder" under the "File Reader" node, for a quick overview of the node task.

Note. The iris dataset file does not contain column names. 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.

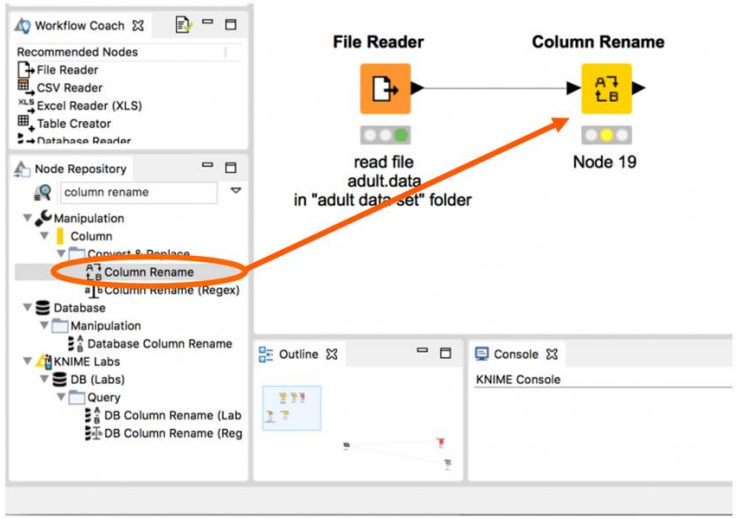
Column Rename

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

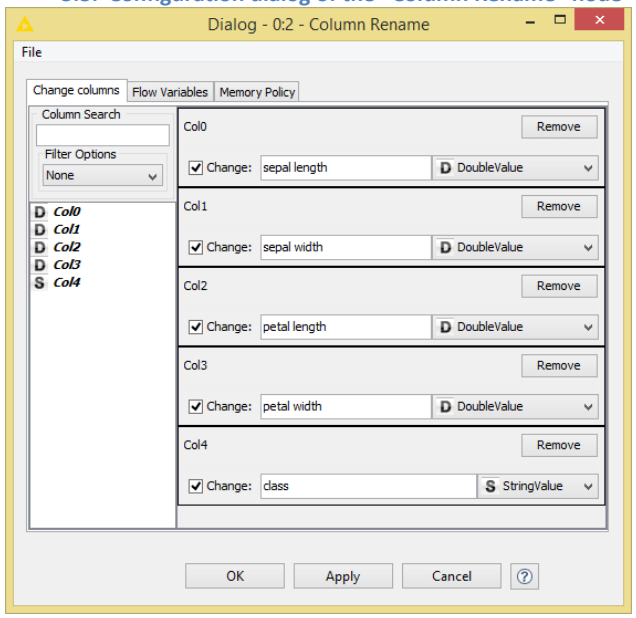
```
“Data Manipulation” -> “Column” -> “Convert & Replace”
```

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



In the configuration window we have on the left the list of candidate columns for renaming; on the right the list of the columns actually selected for renaming.

The configuration dialog requires:

- To select the columns on which to operate by double-click in the left panel
- to flag the columns whose name or type needs changing (checkbox)
- to provide the new column names
- and optionally the new column types
- The button “Remove” is there to remove an already selected column from the renaming panel

We created a “Column Rename” node and we connected it to the “File Reader” node. 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 and that 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 same rule set:

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

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

Note. Fixed 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.

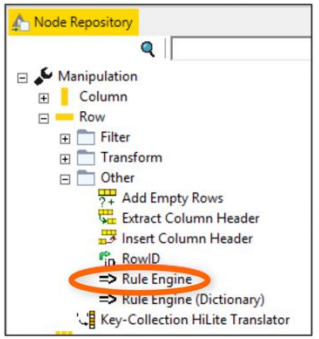
Note. To insert a constant value in a data column, just use TRUE => <new constant value> with no other rule in a “Rule Engine” node. Alternatively, you can use the “Constant Value Column” node.

Rule Engine

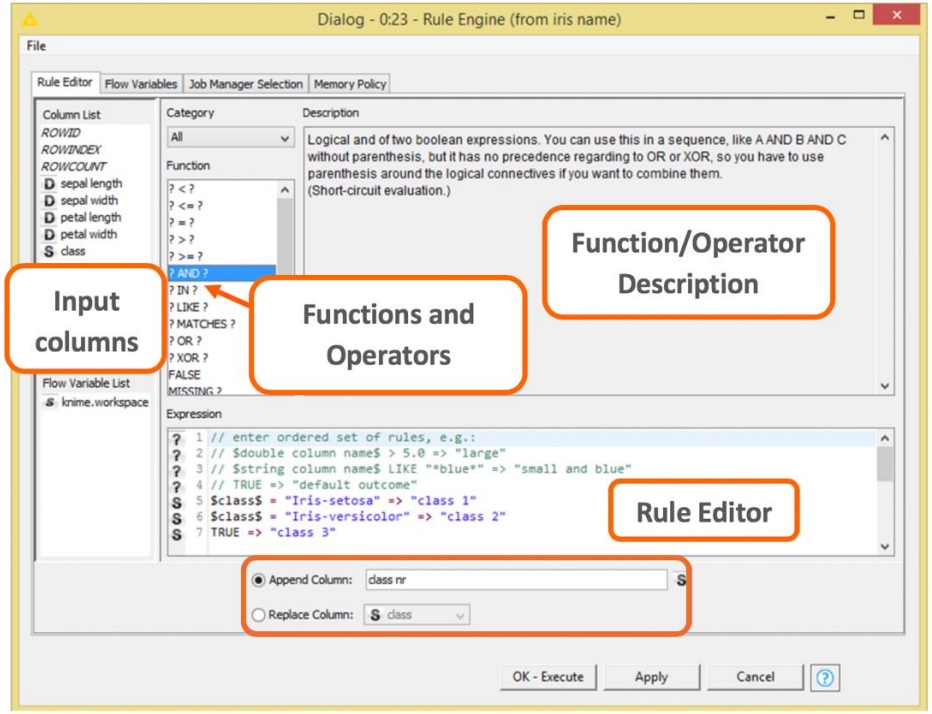
The node “Rule Engine” is located in the “Node Repository” panel in the category `“Data Manipulation” -> “Row” -> “Other”.`

This node defines a set of rules and creates new values based on the set of rules and the input column values.

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 edit 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 them 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 with the right syntax.

Every rule consists of a condition (antecedent), including a function or an operator, and of a consequence value. The symbol “=>” leads the condition to the consequence value, like: <antecedent> => <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.

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 commented the Rule Engine node as “from iris names to class no” and we run the “Execute” command.

3.3. 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 string values. The node splits all strings in the column in k substrings, each of length n1, n2, n3,… nk, where n1+n2+n3 +… nk = L is the length of the original strings. Each substring is then placed in an additional column. Notice that for this node all input strings need to be at least L-character long.

- **"Cell Splitter [by Delimiter]"** uses a delimiter character to extract substrings from the original strings. Strings can be of 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.

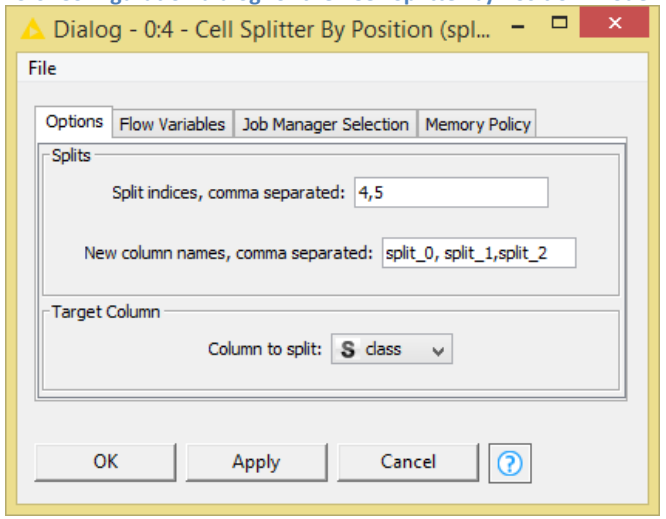
Cell Splitter by Position

This node splits the column string values based on character position. The result consists of as many new columns as many position indices plus 1.

The configuration window asks for:

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

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



We selected:

- Split position indices 4 (at the end of word “Iris”) and 5 (after “-“)
- We will obtain 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]

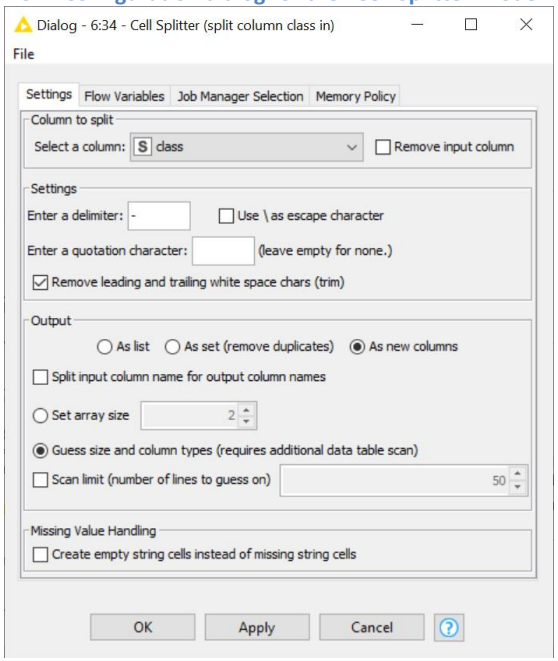
This node splits the column string values at a delimiter character. The result will be as many new columns as many delimiter characters have been found plus one. The configuration window requires the following settings:

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

If many splits are forecasted, the first option can quickly add too many new columns to the output data set and become unmanageable. On the opposite, the second option adds only one additional column to the output data set, compacting all substrings into a collection type column.

The size of the resulting array of substrings can be set a priori or, if we do not know it, we can let the node guess the best size. This last option works for most of the string splitting tasks. For more complex tasks, we might need to set the array size manually ourselves.

3.7. Configuration dialog for the "Cell Splitter" node



In case we set a fixed array size, 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. We selected:

- Column to split = “class”
- Delimiter character = “-“
- Array size = 2 and substrings to be output as new columns

The substrings will be stored in new columns named “<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)

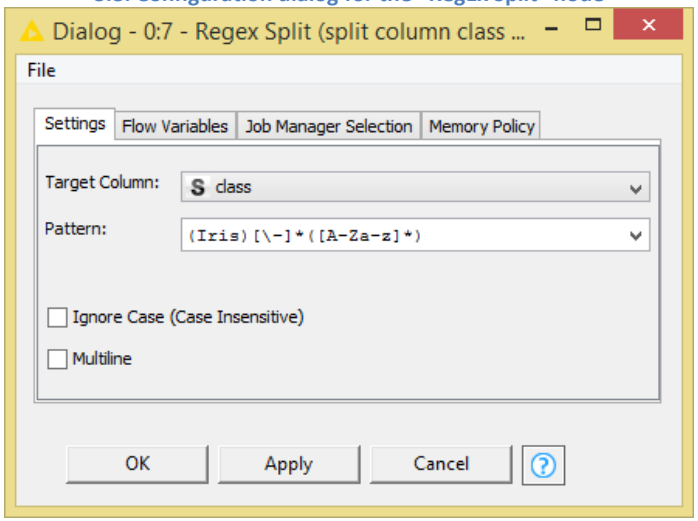
This node identifies substrings in a selected string column on the basis of a Regular Expression.

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

The configuration window requires:

- The name of the column to split
- The Regular Expression patterns to identify the substrings, with the substrings included in parenthesis
- A few additional options to consider multi-line strings and use a case sensitive/insensitive match

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:
  - First substring in parenthesis contains the word “Iris”
  - Then comes a “-“ not to be used as a substring, since it is not in parenthesis
  - 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:

- First substring in parenthesis contains any alphabetical character
- Then comes a “-“ not to be used as a substring, since it is not in parenthesis
- 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”.

3.4. 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 and <attribute> being “virginica”, “setosa”, or “versicolor”. 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 node 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 lower part of the configuration window. Here a number of string functions can be nested and combined together to obtain the desired string transformation.

The available string functions are listed above in the “**Function List**” panel. Functions can also be visualized in smaller groups, by selecting a category in the “**Category List**” menu over the “Function List” panel.

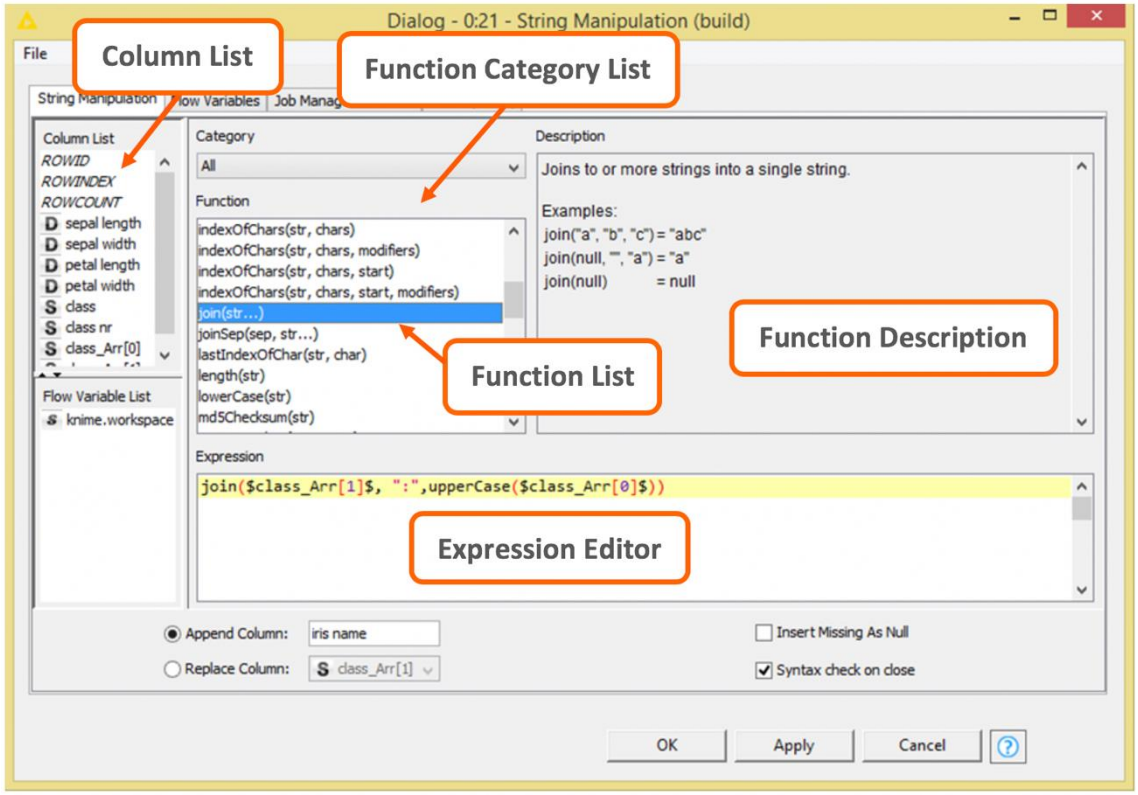
The “**Description**” panel on the right explains the task of the selected function.

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” with the correct syntax. Fixed 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 the string manipulation function is somehow unsuccessful.

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.

3.9. Configuration dialog for the "String Manipulation" node



The “String Manipulation” node that we introduced in the “Write To DB” workflow follows the “Cell Splitter” node and combines (*function “join()”*) the <attribute> part of the class name in column class_Arr[1] with fixed string “:” and with the uppercase version (*function “uppercase()”*) of the word “iris”. The result is for example “setosa:IRIS” for the original string “iris:setosa”.

Note. Functions “toInt()”,“toDouble()”, “toBoolean()”, “to Long()”, “toNull()”convert a string respectively into an integer, a double, and so on. They can be used to produce a non-string output column at the output port of the String Manipulation node.

The String Manipulation node is particularly useful when we want to combine a number of different string functions into a single more complex one. However, an alternative processing uses a sequence of single dedicated nodes. This approach leads to a more crowded workflow, but it provides an easier interpretation of all used string manipulation functions.

To switch from lower to upper case or vice versa, the “Case Converter” node in the category “Data Manipulation” -> “Column” -> “Transform” can be used.

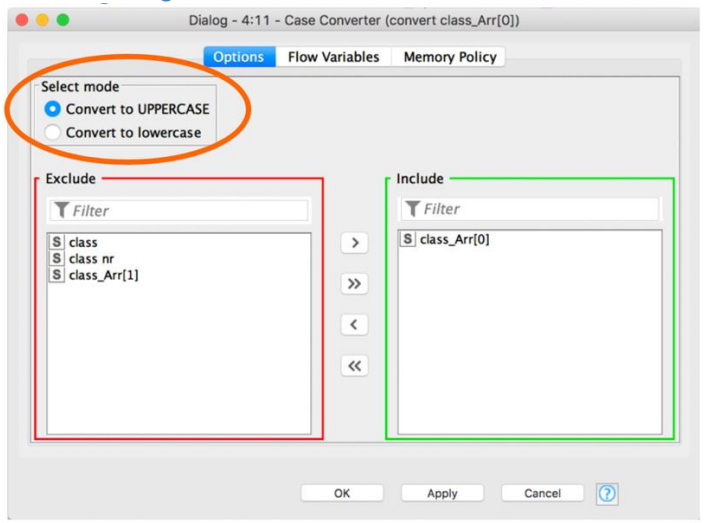
Case Converter

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

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

This node has a variant “String Replace (Dictionary)” that performs the string replacements based on a previously formatted dictionary text file. This node can be useful to replace multiple strings and substrings with the same string value.

String Replacer

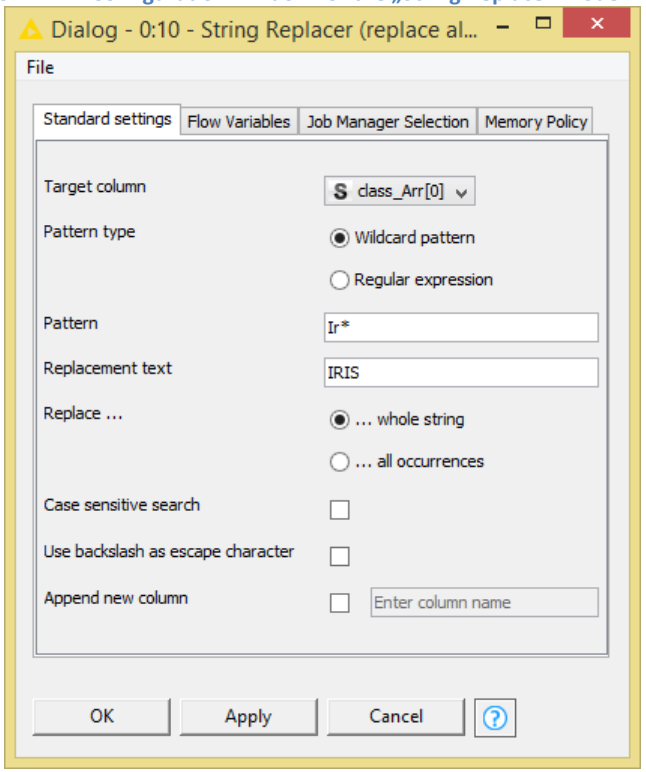
The “String Replacer” node replaces a pattern in the values of a string type column. The configuration window requires:

- The name of the column where the pattern has to be replaced
- The pattern to match and replace (wildcards in the pattern are allowed)
- The new pattern to overwrite the old one

And a few more options:

- Whether the pattern to be matched and replaced contains wildcards or is a regular expression
- Whether the replacement text must replace all occurrences of the pattern as isolated strings only or as substrings as well
- Whether the pattern match has to be case sensitive
- Whether escape characters are indicated through 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 connect a “String Replacer” node to the output port of the “Cell Splitter” node and use the following configuration:

- Target column is “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 overwrite the old one is “IRIS”

Result column “class_Arr[0]” contains all “IRIS” strings, exactly like the column generated with the “Case Converter” node.

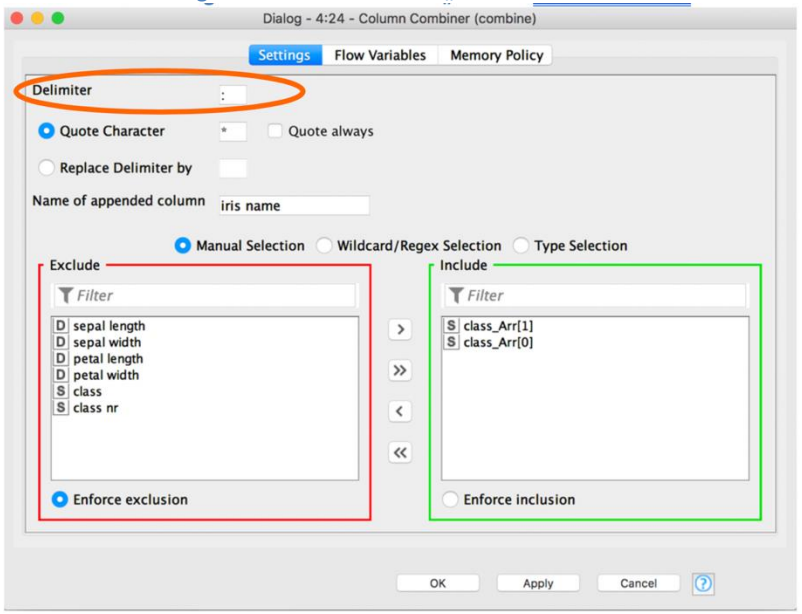
Finally, we want to combine all those substrings in a new string column named “iris name” and containing strings structured as: “<attribute>:IRIS”. To combine two or more string columns, there is the “Column Combiner” node under `“Data Manipulation” -> “Column” -> “Split & Combine”`.

Column Combiner

The "Column Combiner" node combines two or more string columns into a single string column, optionally joining them through a delimiter character. 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 values "<attribute:IRIS>", we need a "Column Combiner" node with the following settings:

- Delimiter is ":"
- Columns to combine in the "Include" frame are "class_Arr[1]" and "class_Arr[0]"
- No quotes around the original strings, that is flag "Quote always" is disabled
- Name of the new column is "iris name". Notice that this node has no option to replace an input column with the new values

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

For example, column “class_Arr[0]” comes before column “class_Arr[1]” in the input data table and therefore 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 substring order, we need to change the column order in the input data table.

To change the columns’ order in the input data table, we use a “Column Resorter” node located in `“Data Manipulation” -> “Column” -> “Transform”`.

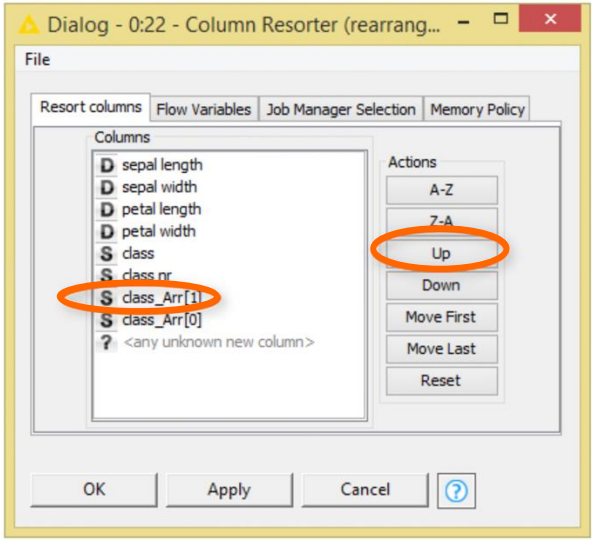
Column Resorter

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

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

- To move one column up or down, select the column in the list and click button “Up” or “Down”.
- 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, that is after column “class_Arr[1]”. After commenting the “Column Resorter” node with “rearrange column order for next node column combiner”, we connected its output port to the “Column Combiner” node. Now the “Column Combiner” has the input columns in the right order to get the final strings structured as “<attribute>:IRIS”.

Note. The “Column Combiner” node is useful to build unique keys to identify data rows.

3.5. Type Conversions

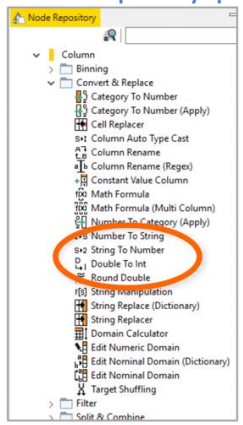
In the previous section we went through the string manipulation functionalities available in KNIME Analytics Platform. Before moving to the database section, I would like to spend a little time showing the “Type Conversion” nodes.

In this book we will not work with data type Date&Time. Excluding this data type, there are three basic type conversion nodes: “Number To String”, “String To Number”, and “Double To Int”. 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

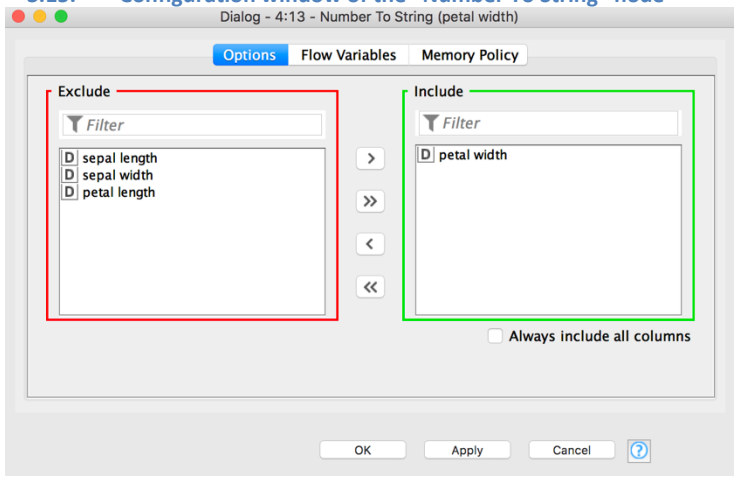
The “Number To String” node converts all cells of a column from type “Double” or “Int” to type “String”.

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.

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

To convert values in the opposite direction of the Number To String node, we find the String To Number node. For demonstration sake, let’s reconvert “petal width” from a String type to a Number type (Double, Long, or Int). In order to do that, we can use the “String To Number” node.

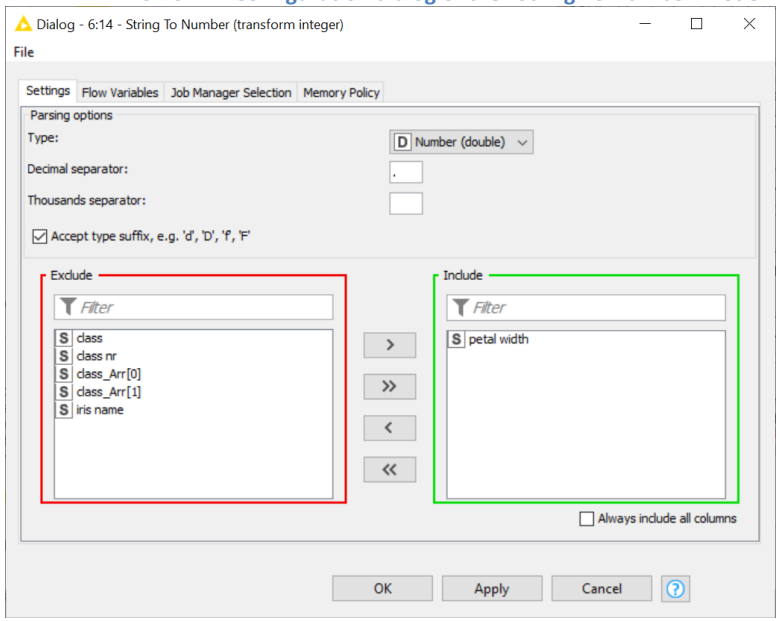
String To Number

The “String To Number” node converts all cells of a column from type “String” to type “Double”, “Long” or “Int”. The configuration window requires:

- The final column type: Double, Long, or Int
- The decimal separator and the thousands separator (if any)
- 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.

The “String to Number” node is equivalent to toInt(), toDouble(), toLong(), and similar functions in the “String Manipulation” node.

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



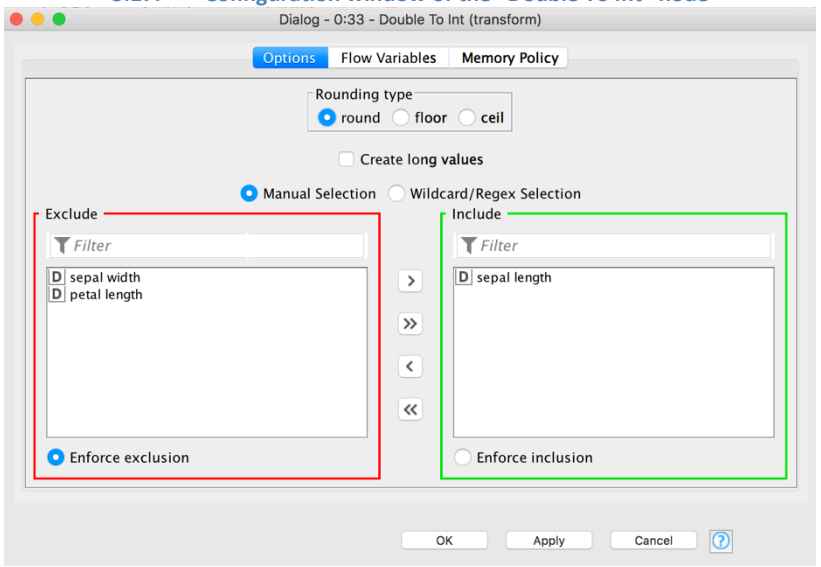
Let's still suppose, for the sake of nodes demonstration, that we have converted the "petal width" array columns to type Double, but that actually we wanted to have them of type Int. Let's ignore the fact that it would be enough to change option "Type" in the configuration window of the "String To Number" 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 selection of the columns to be converted to type Integer. Selection can be set manually or using wildcard or regex. For both selections:
    - Columns to be transformed into type Int 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".

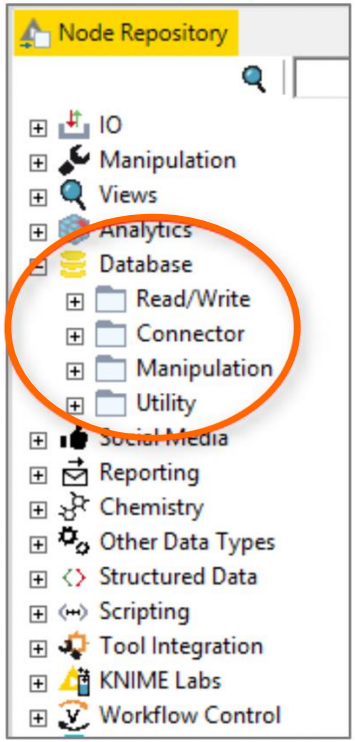
3.17. Configuration window of the "Double To Int" node



3.6. Database Operations

We have only showed the type conversion nodes to illustrate KNIME's potentials. We did not actually need these type conversions to prepare the data for the visualization part. The data produced by the String Manipulation is what we will use in the next workflow for visualization.

3.18. "Database" Category in the Node Repository



We need now to write the data table generated by the String Manipulation node into a database. In the "Node Repository" panel there is a whole category called "Database" containing all nodes that perform operations on databases.

There are two ways to access a database with KNIME:

- We establish a connection to the database with a connector node and along that connection we use a Database Writer or a Database Reader node to write or read the data table;
- We do everything within one node: a Database Writer or a Database Reader.

With both approaches we need a database already installed, the corresponding JDBC driver file, and the credentials – username and password - to access it.

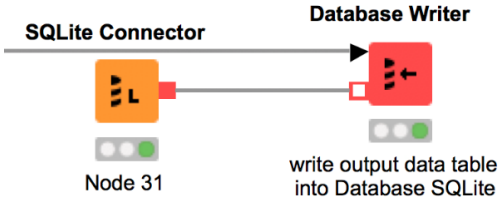
The nodes that KNIME provides to access databases come with pre-loaded JDBC drivers for the most commonly used and most recent database versions, such as MySQL, SQLite, Vertica, Hadoop Hive, Ingres, PostgreSQL, and more.

For this example workflow we use the SQLite database (https://www.sqlite.org/). SQLite is a self-contained, serverless, zero-configuration, transactional file based database which does not require authentication. This makes it easy for the distribution of the workflows associated with this book, since no installation and no configuration of a separate database are required. The database is contained in the file named "KBLBook.sqlite" in the KBLdata folder. Just remember that a similar procedure, including authentication, with a similar sequence of nodes should be followed when using other databases.

If the JDBC driver for your database is not in the list of pre-installed drivers, you can always add it through the Preferences page (see later on in this chapter).

Let's start with the first approach: first we establish the connection to the database and then along this connection we write the data table to the database. For the first task – establishing the connection to a database – we use a connector node. For the second task – writing the data table into the database – we use a Database Writer node.

3.19. SQLite Connector node + Database Writer node



Database Connector Nodes: SQLite Connector

Under category “Database”/”Connectors” you find a number of connector nodes to establish a connection between KNIME Analytics Platform and a database.

Some of those nodes are dedicated connectors, which means they refer to a pre-loaded JDBC driver file. Only the “Database Connector” node is a generic connector to be used, when the dedicated connector for the database of choice is not available.

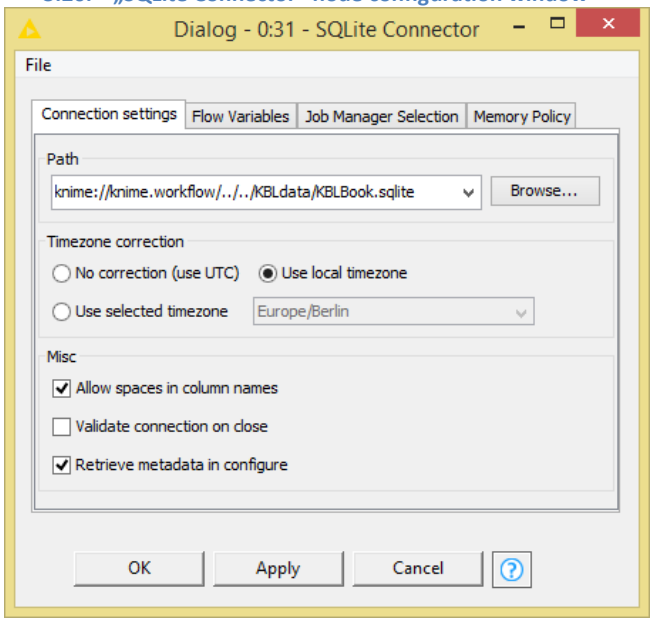
For the SQLite database, a dedicated connector node is available: the SQLite Connector node. Its configuration window just requires the path to the sqlite file and no password. The JDBC driver for the SQLite database is not required, because it has been pre-loaded.

For all connector nodes, it is necessary to specify the time zone we want to use to make sure we import the right Date&Time values.

Other database dedicated connectors might require hostname, port, database name, and full credentials.

The generic “Database Connector” node also requires the JDBC driver file for the database of choice.

3.20. „SQLite Connector“ node configuration window



Note. Did you notice the full red square as input port? So far, we have seen only white triangle as input or output ports. A white triangle means data. A full red square means a database connection. An empty red square means an optional database connection. A full brown square means an SQL statement. There are many different ports, each one exporting or importing a different kind of object.

The Database Writer node has two input ports: one for the data (black triangle) and one for the optional connection from a database connector node (red lined square). The Database Writer node reduces its configuration settings when it receives a database connection. Indeed, the database connection at its input provides already the information about the database driver, the hostname, and the port. In this case, only the table name is required in the configuration settings.

In the following text we have provided a description of the configuration window of a Database Writer node when it is preceded by a Connector node and when it is used as standalone.

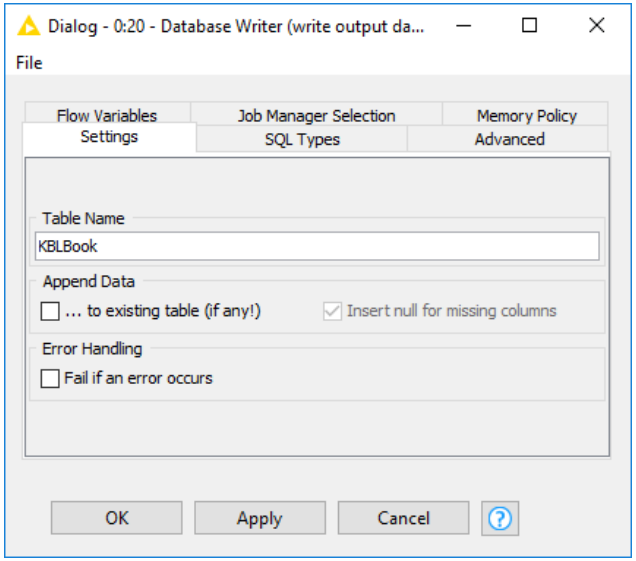
Database Writer following a Database Connector

The node "Database Writer", located in the `"Database"/"Read/Write"` category, writes the input data table into a database table in either "Append" mode or "Overwrite" mode. If the table does not exist in the database, it is created.

If the "Database Writer" node is connected to a connector node, it does not need to establish a connection to the database and therefore all information about the database connection (hostname URL, credentials, database name, etc…) is not required. The only required settings are:

- The name of the table in the database
- The flag for the "Append" mode. The default writing mode is "Overwrite"
- The flag for failure in case of error

3.21. Configuration of the „Database Writer" node when following a Database Connector node



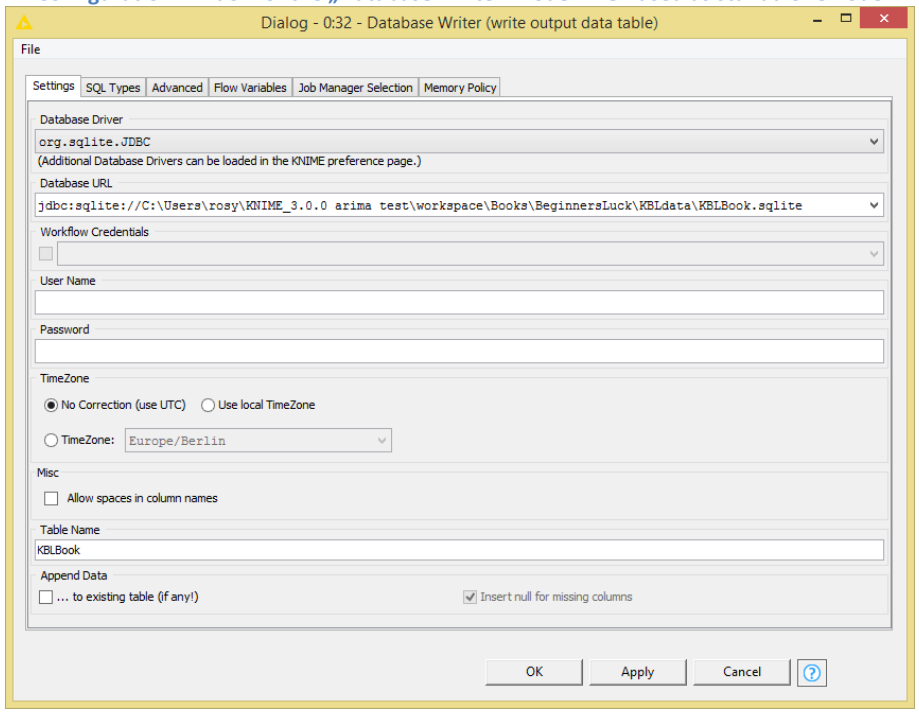
Let's explore now how to use the Database Writer node as a standalone node to establish the database connection and run the writing operation. The configuration window of the Database Writer node (as well as of the Database Reader node) changes and acquires a number of additional required settings mainly about the database connection.

Database Writer used in standalone mode

The node “Database Writer”, located in the `“Database”/”Read/Write”` 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 pre-loaded 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>, and <database name>.
- The credentials to access the database, i.e. username and password, as provided by the database administrator. The database credentials can be supplied:
    - Via the “Workflow Credentials” (recommended)
    - 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 “Overwrite”

3.22. Configuration window of the „Database Writer“ node when used as standalone node



Since we use a SQLite database, username and password are not required. However, most of the other databases require credentials. Credentials can supplied as username and password directly into the configuration window or as credentials for the whole workflow.

Workflow credentials are automatically encrypted and therefore more secure. Providing username and password into the configuration window requires an extra step for security: we need to define a master key. This master key will then be used to automatically encrypt usernames and passwords when provided into configuration windows. Both options will be shown in this chapter.

Workflow Credentials

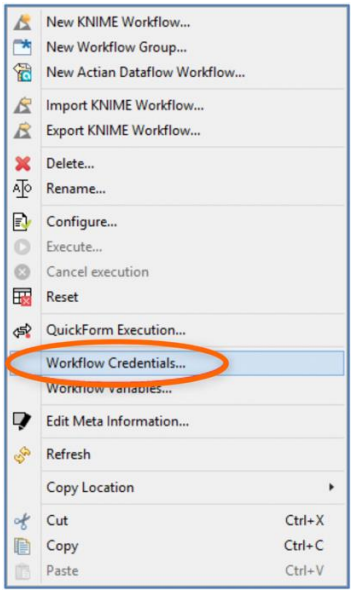
You can set all usernames and passwords that you need for your workflow as workflow credentials from the workflow's context menu.

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

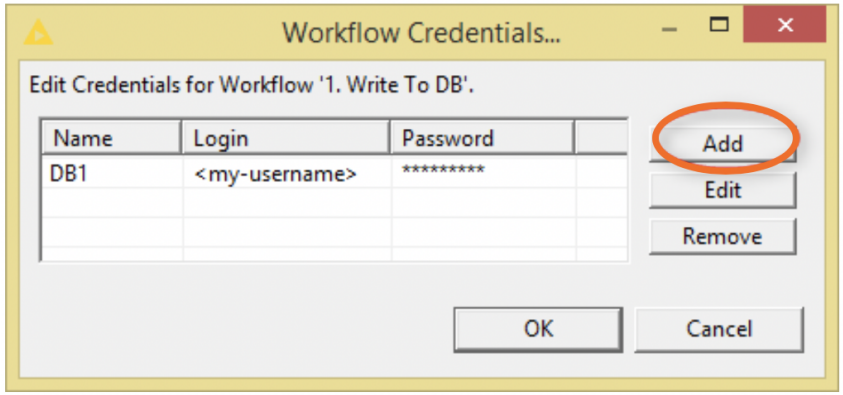
In the "Workflow Credentials …" window, add a new workflow credential -- that is a new pair (Username, Password) -- :

- Click the "Add" button
- In the window "Add/Edit Credentials":
  - Set credential ID, Username (User Login), and User Password
  - Click "OK"
- Insert as many credentials as you need for your workflow
- Click the "OK" button

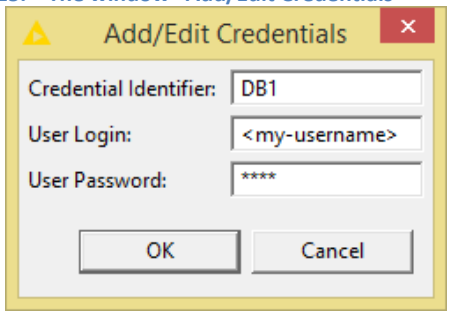
3.23. Set the "Workflow Credentials" from the workflow's context menu



3.24. The "Workflow Credentials..." window



3.25. The window "Add/Edit Credentials"



Note. Workflow credentials are automatically encrypted. Database access via workflow credentials is then more secure and therefore recommended.

The list of all created credential IDs is then available in the menu for the “Workflow Credentials” setting in the configuration window of all database nodes that implement a connection to a database.

Master Key (deprecated)

The Master Key is simply a string used to encrypt passwords when the workflow is saved on disk. Only you should know the value of this Master Key.

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

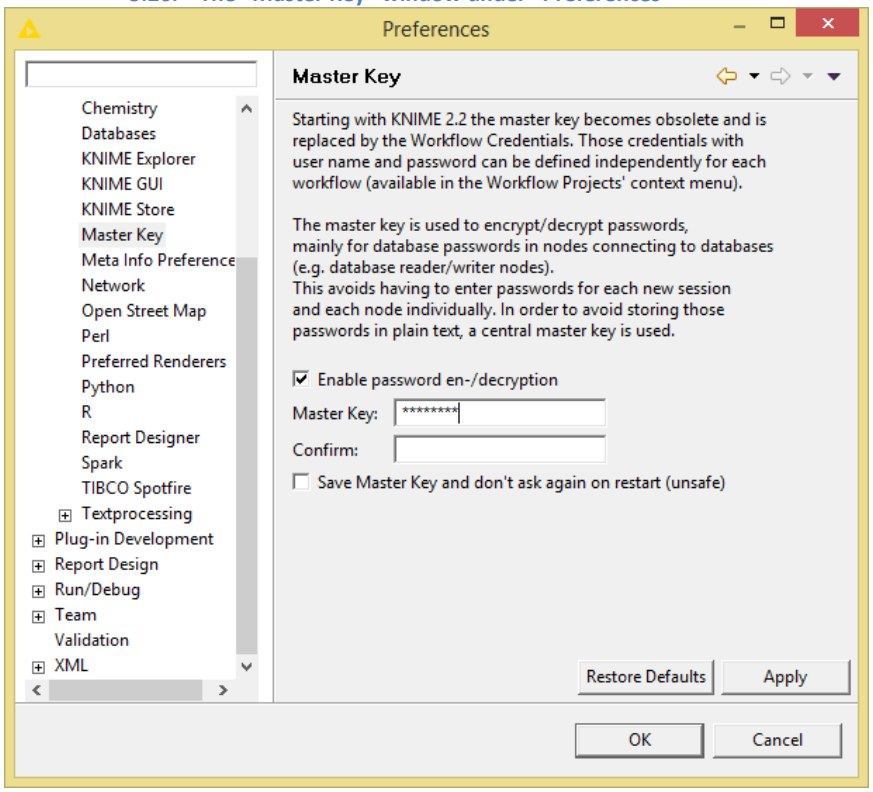
To open the “Preferences” window, in the Top Menu:

- 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”

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

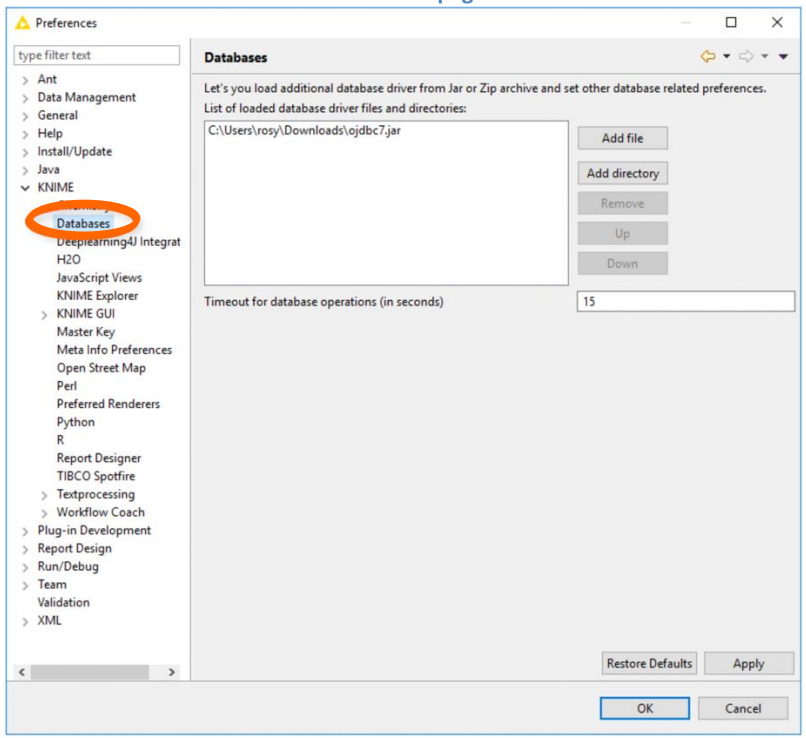


In order to write the processed iris data to the KBLBook.sqlite database, a “Database Writer” node was connected to the output port of the “String Manipulation” node named “build <attr>:IRIS”. In the configuration window of the “Database Writer” node, we set the JDBC driver for SQLite and the path to the database file. For other databases, we might have needed to supply hostname, port, database name, and access credentials.

Import a JDBC Database Driver

The JDBC drivers for the most common and recent databases are already pre-loaded and available in the database nodes. However, it might happen that the JDBC driver for a specific database is not available. In this case, you need to upload the required database driver onto KNIME Analytics Platform. Usually the JDBC 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 order to load a database driver into KNIME, the driver file location must be specified in the KNIME "Preferences" window.

3.27. The "Database Driver" page under "Preferences"



In the Top Menu, select "File" -> "Preferences".
The "Preferences" window opens.

The "Preferences" window sets the values 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 of the "Preferences" window.

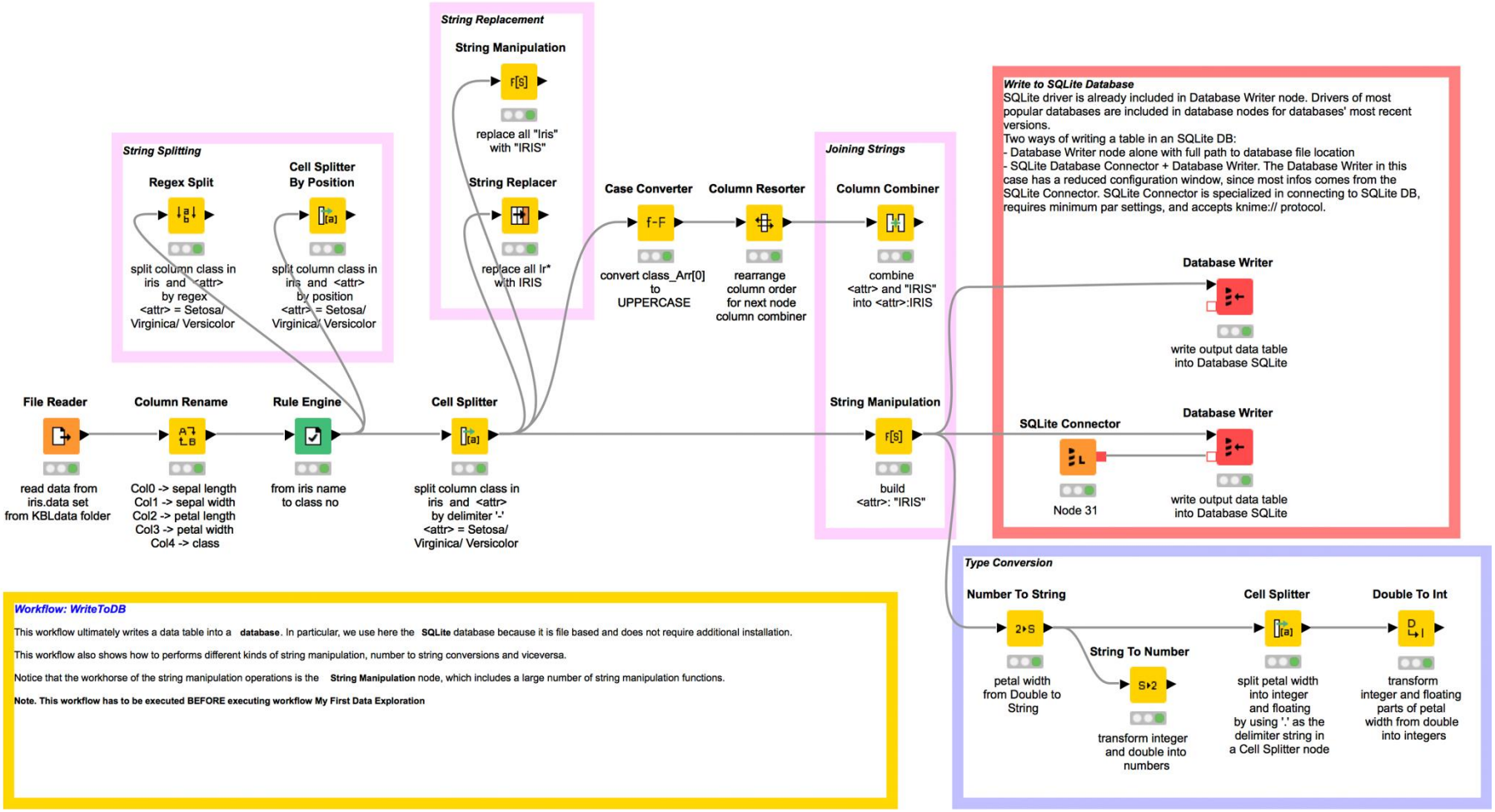
- Expand item "KNIME"
- Select sub-item "Databases"

The panel on the right displays the "Databases" settings.
In order to add a new database driver:

- Click the "Add File" or "Add directory" button
- Select the *.jar or *.zip file that contains the JDBC database driver
- The new JDBC driver appears in the "List of loaded database driver files and directories" in the center and becomes available for all database nodes.

We just completed the workflow "Write To DB", where we read the Iris Dataset and we performed a number of string manipulations and some type conversions. The data table emerging from the string manipulation node has been written into a SQLite database.

3.28. Workflow "Write To DB"

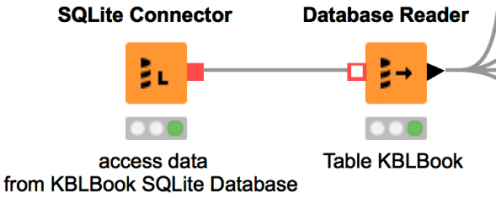


Let's now read from the SQLite database the data we have just written, to perform some visual data exploration. Next to the "Database Writer" node in the "Node Repository" panel, we find the "Database Reader" node, which we will use to read the data from the database table created in the previous workflow.

We create a new workflow "My First Data Exploration" and we place a "Database Reader" node in it. We named the "Database Reader" node "Table KBLBook".

Alternatively, we can establish the connection with the database with a connector node (in this case a “SQLIte Connector” node) and read the data coming from that connection with a “Database Reader” node. Like for the “Database Writer” node, the configuration window of the “Database Reader” node changes depending on whether the node is used in combination with a connector node or in standalone mode.

3.29. SQLite Connector node + Database Reader node



Database Reader following a Database Connector

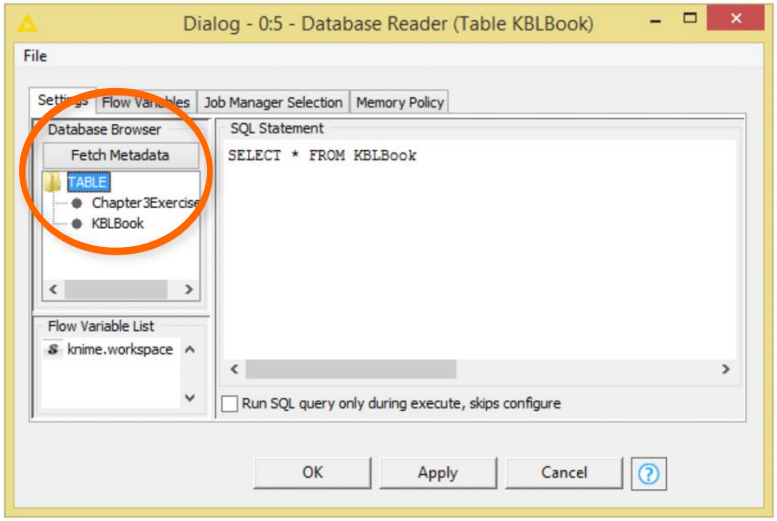
The node “Database Reader”, located in the `"Database"/"Read/Write"` category, reads a table from a database and imports it into a workflow.

If the “Database Reader” node is connected to a “SQLite Connector” node, it does not need to establish a connection to the database and therefore all information about the database connection (hostname URL, credentials, database name, etc…) is not required. The only required settings are:

- The SQL query to extract the data from the selected database table

The “**Database Browser”** on the left side can help you browse the tables and fields in the database. If you click “**Fetch Metadata**”, you will get the database table tree. Double-clicking any table or field in the Database Browser automatically exports it in the SQL Statement editor with the right syntax.

3.30. Configuration of the „Database Writer“ node when following a Database Connector node



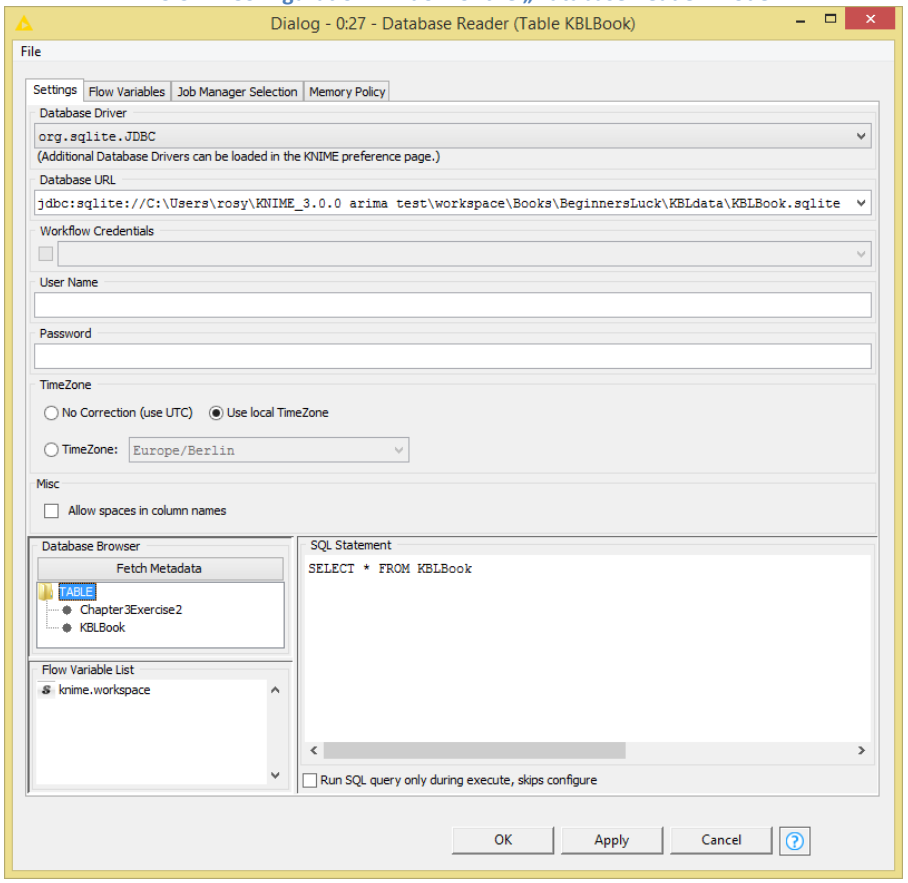
Database Reader used in standalone mode

The "Database Reader" node, located in category `"Database"/"Read/Write"`, 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 pre-loaded 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 above "Import a Database Driver").
- The URL location of the database, in terms of server name (host), port, and 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 any.
- The SQL query (SELECT) to extract the data.

3.31. Configuration window of the „Database Reader" node



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 with the right syntax.

If it is not necessary to read the whole data table in your workflow, you might want to upload only the columns and/or the rows of interest via the SELECT SQL statement.

The node reads all the columns from the table KBLBook in database KBLBook.sqlite:

- 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 earlier 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 string manipulation operations.

3.7. Aggregations and Binning

As an example, let’s investigate the distribution of feature sepal_length across the whole data set. We will approximate this distribution visually with a histogram. The histogram needs ranges of values (bins) on which to count the number of occurrences. So, before proceeding with the drawing of the histogram, we define such bins on sepal_length value range. To do that, we use a “Numeric Binner” node.

We chose to build the histogram on the values of sepal_length 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. The column containing the bins was named “sepal_length_binned”.

We now want to count the number of iris plants for each species and with the “sepal_length” measure falling in one of the bins; that is we want to count the number of iris plants by “sepal_length_binned” and by “class”.

In KNIME we can produce an aggregation of values based on groups 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.

Both nodes are quite important in the KNIME node landscape, since they are quite flexible and allow for a number of different aggregation operations, from simple row counting to the calculation of statistical measures, from correlation to value concatenation.

Both nodes groups the input data according to the values in some selected columns and on the defined groups calculate a number of aggregation measures. The only difference is in the shape of the aggregated output data table. In the results of the “GroupBy” node each aggregation group is identified by the values in the first columns, while the final column contains the aggregated measure relatively to that group. In the resulting table of the “Pivoting” node, each cell contains the aggregation measure for the group identified by the values in its column header and in its RowID. Given the importance of both nodes, we used both of them.

We set “sepal_length_binned” and “class” to identify the different group and we used “count” as aggregation measure on “sepal_length” column. “count” counts the rows in the defined group, that is for all irises in “class” iris-virginica with “sepal_length” between 6 and 7.

Numeric Binner

The “Numeric Binner” node - located in the “Node Repository” panel in “Data Manipulation” -> ”Column” -> “Binning” category - defines a series of intervals (i.e. bins) and assigns each column value to its bin.
The configuration window requires the following:

- The numerical column to be binned
- The list of bin intervals
- 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:

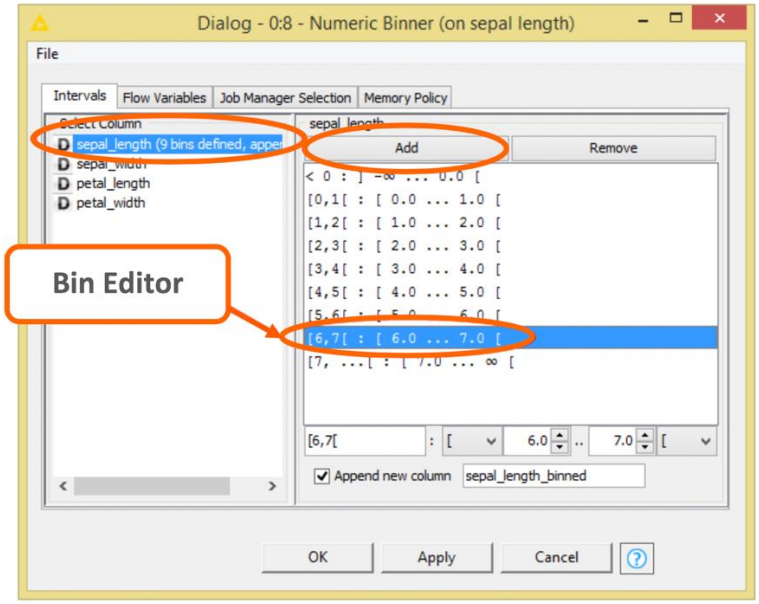
- Click the “Add” button
- Customize the bin range in the Bin Editor

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 new bin representation by selecting another column and repeating the binning procedure.

3.32. Configuration window for the „Numeric Binner“ node



Note. Aggregation method “count” just counts the rows in the group. It makes no difference which column it uses to count the rows, if we do not exclude those with missing values. However, this is the only aggregation method with this particularity. All other methods, such as average or sum or standard deviation, will of course produce different results when applied to different columns.

GroupBy: “Groups” tab

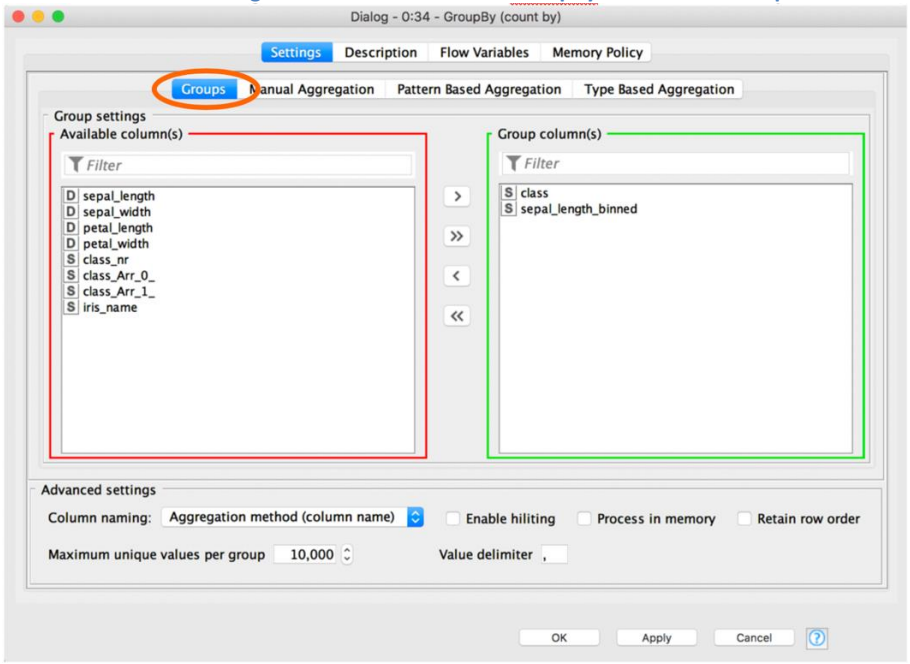
The “GroupBy” node finds groups of data rows by using the combination of values in one or more columns (**Group Columns**); it subsequently aggregates the values in other columns (**Aggregation Columns**) across those groups. Column values can be aggregated in the form of a sum, a mean, just a count of occurrences, or using other aggregation methods (**Aggregation Method)**.

The configuration window of the “GroupBy” node consists of a number of tabs. Here we check the tab named “**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.

3.33. Configuration window for the "GroupBy" node: tab “Groups”

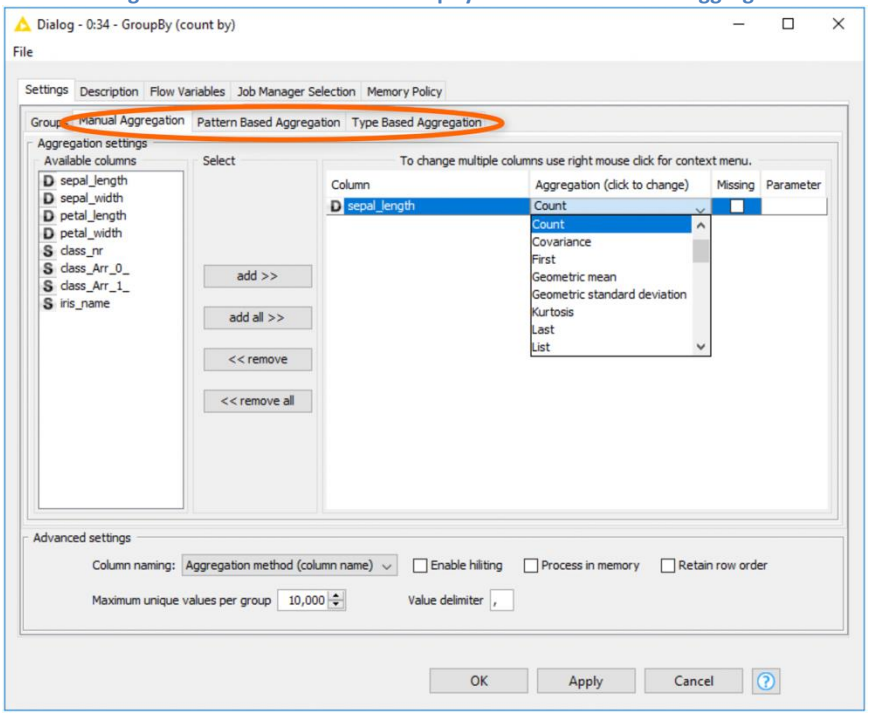


The lower part of the configuration window

- sets the name of the new column
- keeps the row order or resorts them in alphabetical order
- rejects columns with too many different distinct values (default 10000), therefore generating too many different distinct groups
- option “Enable hiliting” refers to a feature available in the old “Data Views” node.

GroupBy: Aggregation tabs

3.34. Configuration window for the "GroupBy" node: tab “Manual Aggregation”



The remaining tabs in the configuration window define the aggregation settings, that is:

- *The aggregation column(s)*
- *The aggregation method (one for each aggregation column)*

The different tabs select the columns on which to perform the aggregation using different criteria:

- Manually, one by one, through an “Exclude”/”Include” frame: all columns selected will be used for aggregation
- Based on a regex or wildcard pattern: all columns with name matching the pattern will be used for aggregation
- Based on column type: all columns of the selected type will be used for aggregation

Several aggregation methods are available in all aggregation tabs. All available aggregation methods are described in detail in the “Description” tab.
Aggregation methods differ for numerical columns (including here statistical measures, like mean, variance, skewness, median, etc…) and for String columns (including unique count for example).

Notice that aggregation methods “Count” and “Percent” just count the number of data rows for a group and its percent value with respect to the whole data set. That means that whichever aggregation column is associated with these two aggregation methods, the results will not change, since counting data rows of one group and its percentage does not depend on the aggregation column but only on the data group.

Aggregation methods “First” and “Last” respectively extracts the first and last data row of the current group.

The most frequently used aggregation methods for numerical columns are: Maximum, Minimum, Mean, Sum, Variance, and Sum. The most frequently used aggregation methods for nominal columns are: Concatenate, [Unique] List, and Unique Count.

Pivoting

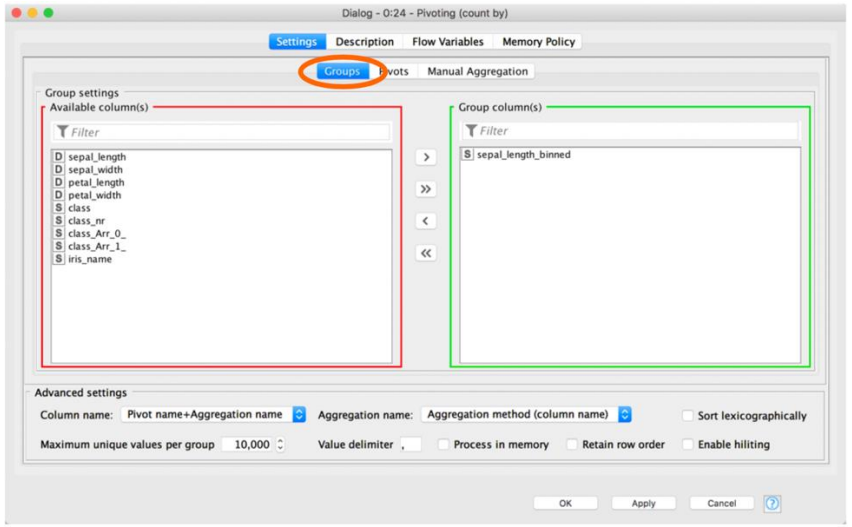
The "Pivoting" node finds groups of data rows by using the combination of values from **two or more** columns: the "**Pivot**" columns and the "**Group**" columns. It subsequently aggregates the values from a third group of columns (**Aggregation Columns**) across those groups. Column values can be aggregated in the form of a sum, a mean, just a count of occurrences, or a number of other aggregation methods (**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 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 column
- The third output port presents the totals by pivot column

3.35. "Pivoting" node: tab "Groups"



The "Pivoting" node is configured by means of three tabs: **"Groups", "Pivots",** and **"Manual Aggregation".**

Tab "**Groups**" 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.

The lower part of the configuration window

- sets the name of the new column
- keeps the row order or resorts them in alphabetical order
- rejects columns with too many different distinct values (default 10000), therefore generating too many different distinct groups
- option "Enable hiliting" refers to a feature available in the old "Data Views" nodes

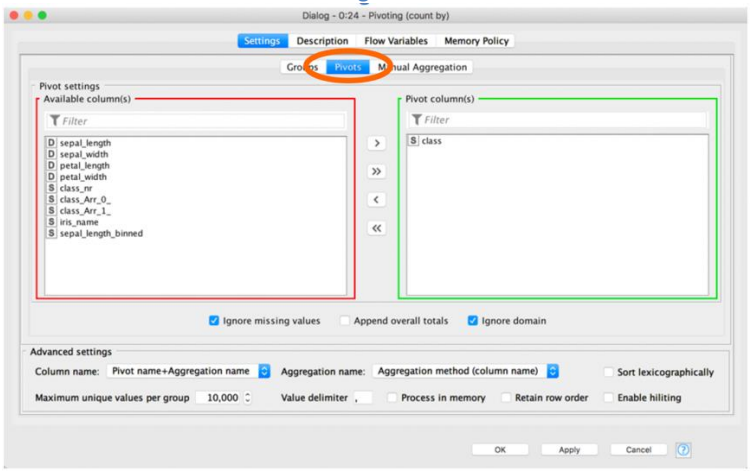
Tab “**Pivots**” defines the Pivot columns by means of an “Exclude”/”Include” frame:

- 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 “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).

3.36. "Pivoting" node: tab “Pivots”



Tab **“Manual Aggregation”** 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:

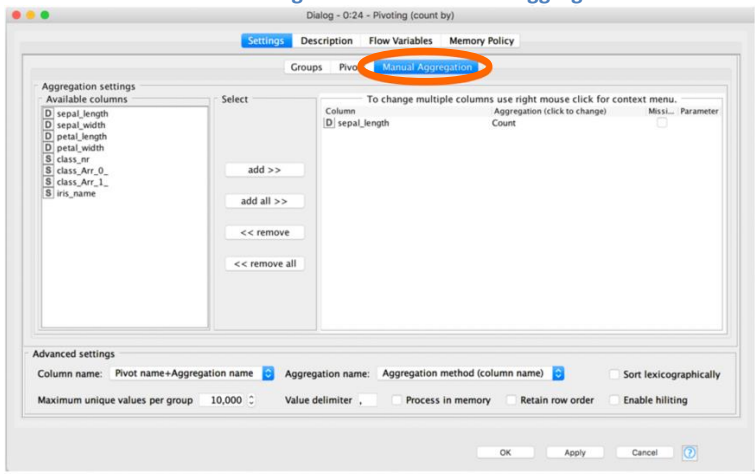
For each selected aggregation column, you need to choose an aggregation method. Several aggregation methods are available. They are all described in the “Description” tab.

Aggregation methods “Count” and “Percent” just counts the number of data rows in a group and therefore they are independent of the associated aggregation column.

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 = distinct values in the group columns

3.37. "Pivoting" node: tab "Manual Aggregation"



3.8. Nodes for Data Visualization

Let's move now into the data exploration and data visualization part through the graphic functionalities of KNIME Analytics Platform. For historical reasons, there are three possible ways to graphically represent data in KNIME: Data Views nodes, JFreeChart based nodes, and Javascript based view nodes.

Data Views nodes are located in category "Views" in the "Node Repository". These nodes get a data table as input and produce a temporary graphical representation of the data, i.e. a view. Those are the oldest graphical nodes in KNIME Analytics Platform, creating a less powerful and less detailed graphical representation.

JFreeChart nodes are located under "Views"/"JFreeChart" category in the "Node Repository". These nodes are based on the Java JFreeChart graphical libraries. They are similar in contents and tasks to the Data Views nodes, but they produce a static image rather than a temporary view of the data graphical representation. The static image is exported into the KNIME workflow and can be used later on for reports, but not for interactive exploration of the data structure.

The newest baby in the data visualization node sets in the KNIME Analytics Platform consists of the **Javascript based** nodes. These nodes, located in "Views"/"Javascript", are based on Javascript graphical libraries and therefore allow for better graphics and a higher interaction level than the previous Data Views nodes. These nodes produce a data table and a static image. The output data table is a copy of the input data table plus a column containing the selection flag for each data point. The output image is a screenshot of the node graphical view. It can be exported into the workflow for reporting or other usage. Because of their better graphics and higher interactivity, in this section we will focus on Javascript based nodes for data visualization.

3.9. Scatter Plot (Javascript)

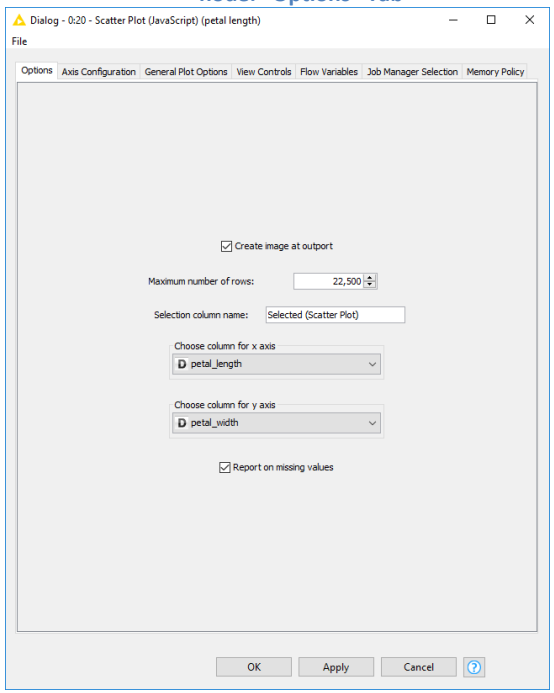
Let's start our data exploration with a classic scatter plot. The node to use here is the "Scatter Plot (Javascript)" node.

The "Scatter Plot (Javascript)" node plots each data row as a dot by using two of its attributes as coordinates on the X-axis and the Y-axis. After reading the iris data set from the KBLBook.sqlite database, we want to produce a scatter plot of petal length vs. petal width, which is the view where the three groups of iris flowers are best recognizable.

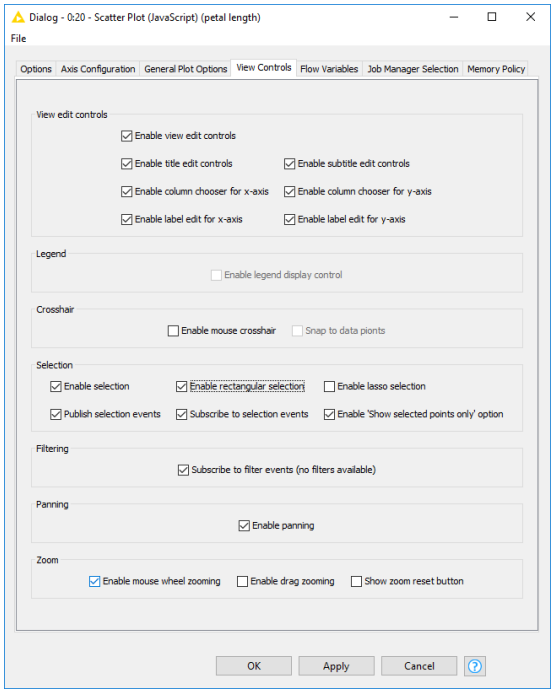
The configuration window of a "Scatter Plot (Javascript)" node covers 4 option tabs: "Options", "Axis Configuration", General Plot Options", and "View Controls". Tab **"Options"** defines the columns to report on the x- and y-axis, the name of the output column for the selected points, the emergency criterion for the maximum number of data rows to visualize, a flag to reproduce the view into an image at the output port, and a flag to produce a warning in case of missing values. Tab **"General Plot Options"** specifies the image options, such as size, title, features, colors, and background. Tab "**Axis**

Configuration” defines the axis options for the view and the image, such as labels, range, and format. The **“View Controls”** tab defines the allowed interactivity on the final view, such as the possibility to edit title and labels, change displayed columns, zooming, and select points.

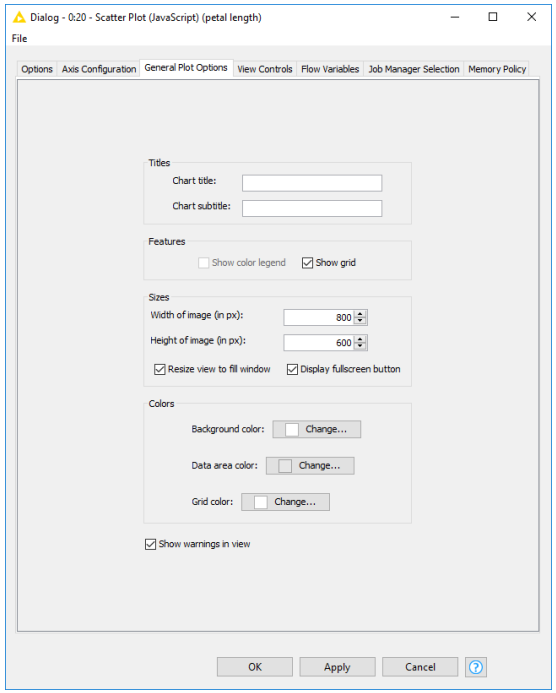
3.38. Configuration of “Scatter Plot (Javascript)” node: “Options” Tab



3.39. Configuration of “Scatter Plot (Javascript)” node: “View Controls” Tab



3.40. Configuration of “Scatter Plot (Javascript)” node: “View Controls” Tab



After execution, the node produces an interactive view. Right-click the node and select “Interactive View: Scatter Plot”. The level of interactivity of this view was decided in the settings of the “View Controls” tab of the node configuration window. Let’s explore this view and let’s see the kind of interactivity it allows.

The view of the “Javascript Scatter Plot” node opens using the settings of the configuration window. In our case, opens on petal length vs. petal width, with such axis label, no title, wheel zooming, and simple and rectangular selection enabled, as defined in the “View Controls” tab. Depending on the options you have enabled in the “View Controls” tab of the configuration window, the view of the “Javascript Scatter Plot” node will be more or less interactive.

Scatter Plot (Javascript): Interactive View

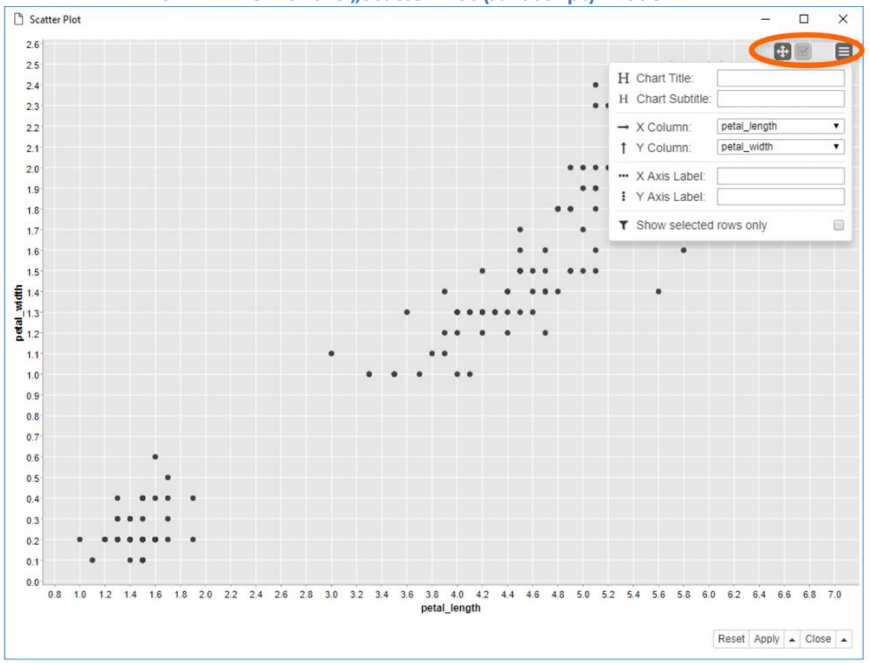
This on the side is the view of the “Scatter Plot (Javascript)” node, where you can see the dots of the scatter plot.

There are three buttons in the upper right corner. Those are the interactivity buttons.

Starting from the far right, we have the button that allows to change the plot settings, such as axis labels, columns for x-axis and y-axis, and title.

The second button from the right puts the mouse-click into selection mode. When enabled, clicking a point or drawing a rectangle in the plot selects the corresponding points.

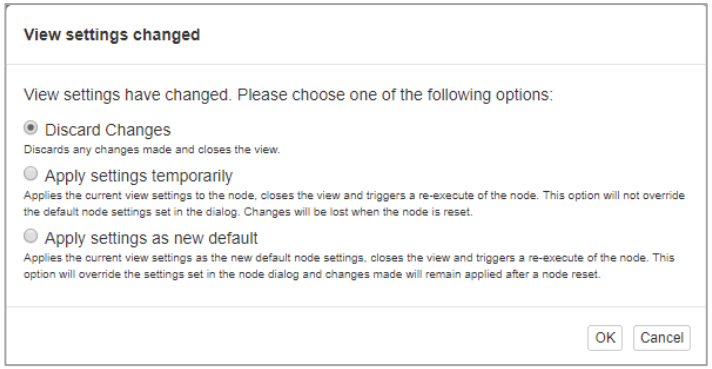
3.41. View of the „Scatter Plot (Javascript)“ node



After selecting points or changing settings, if we click the button “Close” in the lower right corner, a window appears asking whether we want to keep the new settings, i.e. the selected points, either temporarily or permanently. With this last option we practically overwrite the node settings.

The last button from the right allows for panning, that is zooming and moving around the plot.

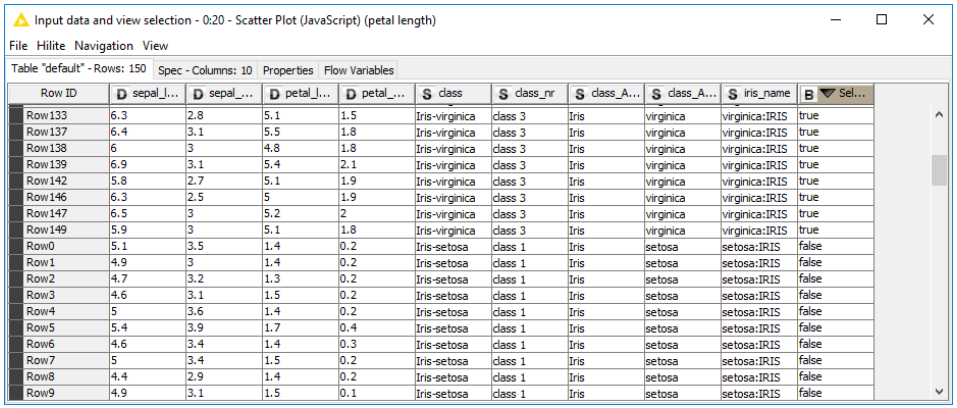
3.42. Confirmation on Changes after click on “Close” button



Note. The flag “create image at output port” in the “Options” tab might slow down the node execution if the image is built on a larger number of input records. In this case, you might consider disabling this flag in the interest of speed execution.

After selecting some points in the plot, closing the view, and accepting the changes, in the output data table the additional column named “Selected Scatter Plot” shows a series of “true” and “false” values. “true” is associated to all selected records, “false” to all others.

3.43. "true" and "false" values in the additional column named "Selected Scatter Plot" and produced by the "Scatter Plot (Javascript)" node. "true" is associated to all selected records. "false" indicates a not selected records and it is the default value



Row ID	sepal_l...	sepal_...	petal_l...	petal_...	class	class_nr	class_A...	class_A...	iris_name	Sel...
Row133	6.3	2.8	5.1	1.5	Iris-virginica	class 3	Iris	virginica	virginica:IRIS	true
Row137	6.4	3.1	5.5	1.8	Iris-virginica	class 3	Iris	virginica	virginica:IRIS	true
Row138	6	3	4.8	1.8	Iris-virginica	class 3	Iris	virginica	virginica:IRIS	true
Row139	6.9	3.1	5.4	2.1	Iris-virginica	class 3	Iris	virginica	virginica:IRIS	true
Row142	5.8	2.7	5.1	1.9	Iris-virginica	class 3	Iris	virginica	virginica:IRIS	true
Row146	6.3	2.5	5	1.9	Iris-virginica	class 3	Iris	virginica	virginica:IRIS	true
Row147	6.5	3	5.2	2	Iris-virginica	class 3	Iris	virginica	virginica:IRIS	true
Row149	5.9	3	5.1	1.8	Iris-virginica	class 3	Iris	virginica	virginica:IRIS	true
Row0	5.1	3.5	1.4	0.2	Iris-setosa	class 1	Iris	setosa	setosa:IRIS	false
Row1	4.9	3	1.4	0.2	Iris-setosa	class 1	Iris	setosa	setosa:IRIS	false
Row2	4.7	3.2	1.3	0.2	Iris-setosa	class 1	Iris	setosa	setosa:IRIS	false
Row3	4.6	3.1	1.5	0.2	Iris-setosa	class 1	Iris	setosa	setosa:IRIS	false
Row4	5	3.6	1.4	0.2	Iris-setosa	class 1	Iris	setosa	setosa:IRIS	false
Row5	5.4	3.9	1.7	0.4	Iris-setosa	class 1	Iris	setosa	setosa:IRIS	false
Row6	4.6	3.4	1.4	0.3	Iris-setosa	class 1	Iris	setosa	setosa:IRIS	false
Row7	5	3.4	1.5	0.2	Iris-setosa	class 1	Iris	setosa	setosa:IRIS	false
Row8	4.4	2.9	1.4	0.2	Iris-setosa	class 1	Iris	setosa	setosa:IRIS	false
Row9	4.9	3.1	1.5	0.1	Iris-setosa	class 1	Iris	setosa	setosa:IRIS	false

The interactivity is nice, but the view of the scatter plot looks a bit sad in its black and white simplicity.

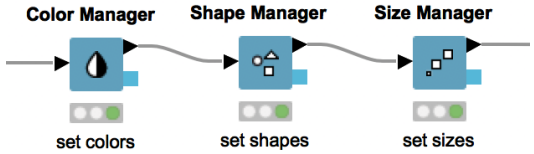
Note. It is not possible to define graphical properties such as dot color, size, and shape through the scatter plot node itself. You need a property manager node, such as “Color Manager”, “Size Manager”, or “Shape Manager” node.

3.10. Graphical Properties

Graphical plots in node views can be customized with color, shape, and size of the plot’s markers. KNIME Analytics Platform has three nodes, in “Views” -> “Property” in the “Node Repository” panel, to customize plot appearance: “Color Manager”, “Size Manager”, and “Shape Manager”. These nodes take a data table as input and produce two objects at two separate output ports.

- The first output port contains the same data table from the input port, with the additional graphical properties as color, size, and/or shape assigned to each data row.
- The second output port contains the graphical model; that is the color, shape, or size adopted for each record. This graphical model can be passed to an “Appender” node and applied to another data set.

3.44. The three views property nodes, to set color, shape, and size of plot markers



Let's have a look at the "Color Manager" node as an example of how these graphical property nodes work.

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 a nominal column is selected in the configuration dialog, colors are assigned to each one of the nominal values.

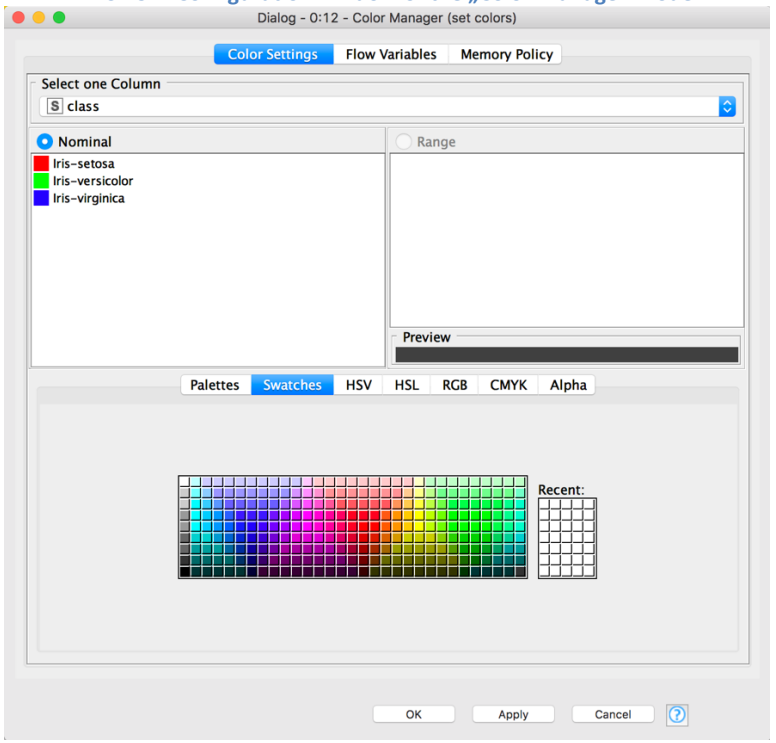
If a numerical column is selected, a color heat map spans the column numerical range.

The configuration window requires:

- The column from which to extract values (nominal columns) or ranges (numerical columns)
- The color map for each list of values or range of values

A default color map is assigned by default to the list / range of values. This can be changed by selecting the value / range and then assigning a different color from the color map displayed in the lower part of the configuration window.

3.45. Configuration window of the „Color Manager" node



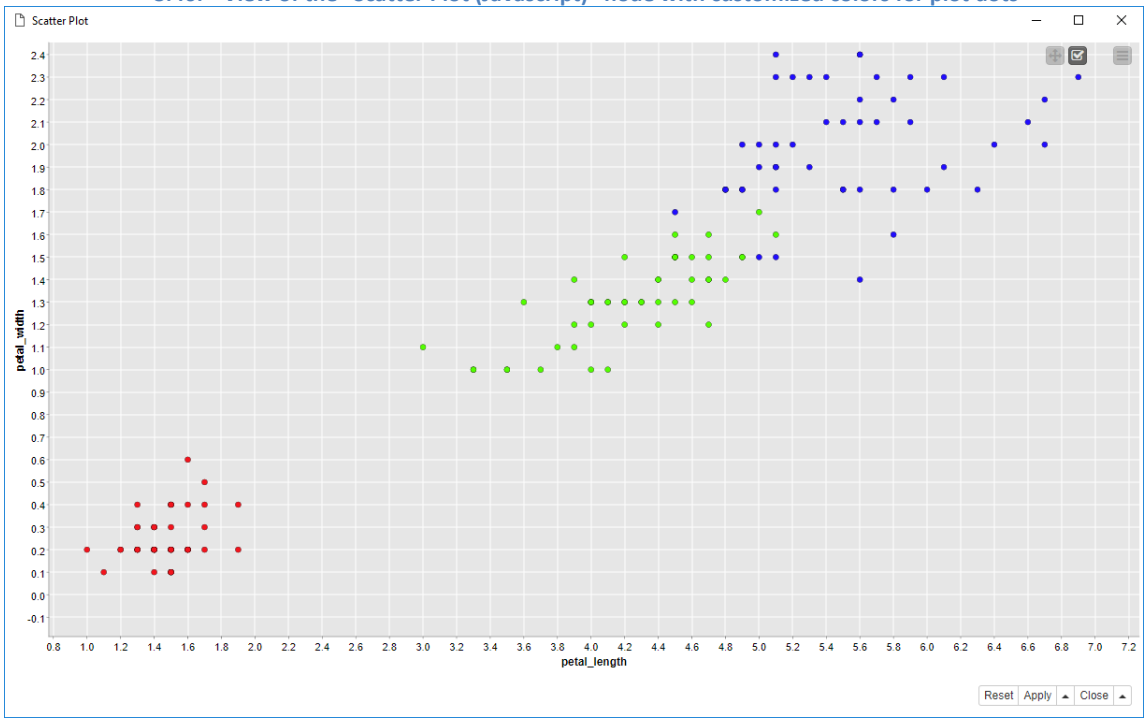
Similarly to the "Color Manager" node, in the configuration window of the "Shape Manager" node, shape can be changed by clicking the row with the desired column value and assigning a shape from the menu list on the right.

The “Size Manager” node on the opposite uses a multiple of an input numerical column to scale the size of the plot markers. Its configuration window then requires the numerical column and the factor to use for the scaling operation.

Warning. As of KNIME Analytics Platform 3.5, the “Size Manager” node and the “Shape Manager” node are not supported by the Javascript based visualization nodes.

This time, a “Color Manager” node was applied to the original iris data before feeding the scatter plot node. In the configuration window we selected the “class” column for the marker assignment and we allocated different colors to each one of the three iris labels found in the “class” column. The introduction of this graphical property transforms the scatter plot - reported above in black and white - into the following scatter plot.

3.46. View of the "Scatter Plot (Javascript)" node with customized colors for plot dots



3.11. Line Plots and Parallel Coordinates

Another useful plot is the line plot, to draw time series and other evolving phenomena along one dimension only. A line plot connects attribute values sequentially, i.e. following their order in the input data table. The row sequence represents the X-axis, while the corresponding attribute values are plotted on the Y-axis. Multiple lines, i.e. multiple columns, can be reported in the plot.

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 (Javascript)" node works.

Line Plot (Javascript)

The "Line Plot (Javascript)" node displays a line plot, using one column as X-axis and one or more column values as Y-axis.

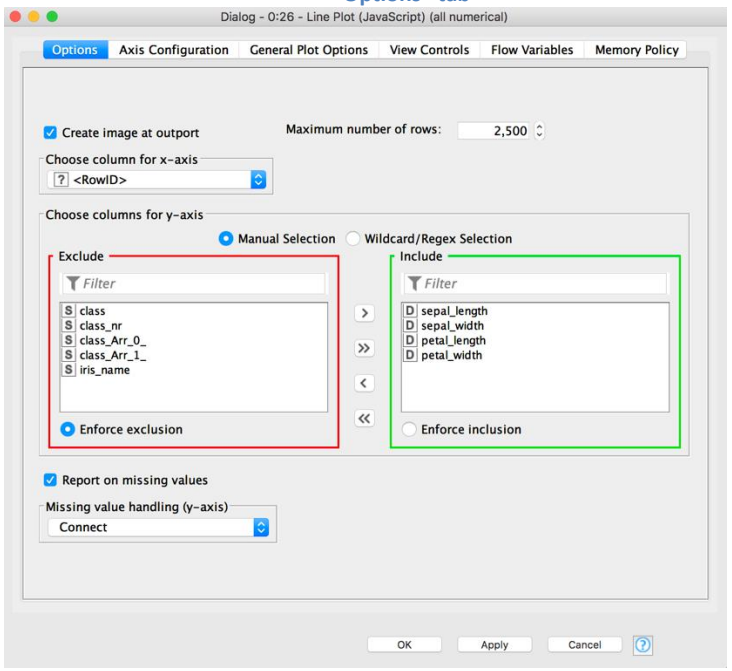
As for the previous Javascript based visualization nodes, the configuration window of the "Line Plot (Javascript)" node has four tabs: ***"Options"*** for the data; ***"Axis Configuration"*** and ***"General Plot Options"*** for the plot details; and ***"View Controls"*** for the interactivity features.

The main difference is in the "Options" tab, where an "Include"/"Exclude" frame allows to select the columns for the plot.

A number of missing values handling strategies are also available: just ignore the missing value and connect the closest two ones; leave an empty gap; or remove the whole column if it contains missing values.

Unlike other Javascript based visualization nodes, the "Line Plot (Javascript)" node has a second optional input port for the color scheme. In this input map column names are associated with colors. In the final plot, each column will then be drawn using the associated color in the map.

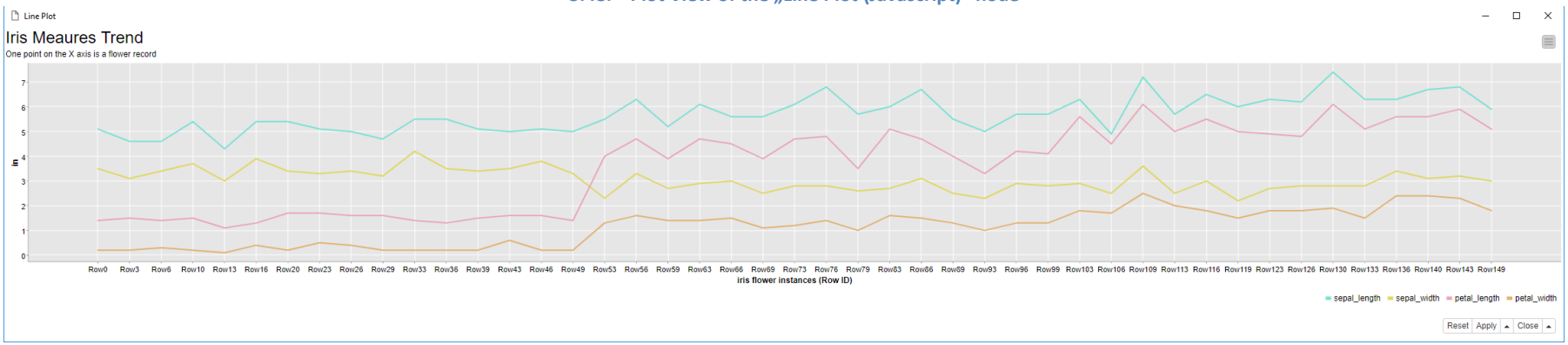
3.47. Configuration window of the „Line Plot (Javascript)" node: "Options" tab



The final view of the “Line Plot (Javascript)” node is shown in the following figure, where RowIDs are displayed on the X-axis and iris measures are displayed on the Y-axis. We used here RowIDs for the X-axis, but we could have used any other column for that. Whatever had been chosen to be reported in the X-axis, the plot would have still drawn the column values in sequence, in order of appearance in the input data table.

Warning. As of KNIME Analytics Platform 3.5, the “Line Plot (Javascript)” node does not allow for much interactivity.

3.48. Plot View of the „Line Plot (Javascript)“ node



Another interesting plot is the Parallel Coordinates plot. Parallel coordinate visualization plots are useful to get an idea of pattern groups across columns. For example, for our iris data set, we can see that one of the iris classes gets easily separated from the other two along the coordinates “petal length” and “petal width”.

In the parallel coordinates plot, one column is one coordinate, i.e. one Y-axis. Multiple column values can be visualized on multiple coordinates, that is on multiple Y-axis. The data disposition along each axis can tell us some stories about the groups in the data set. The node that produces a parallel coordinate plot is the “Parallel Coordinates (Javascript)” node.

Parallel Coordinates (Javascript)

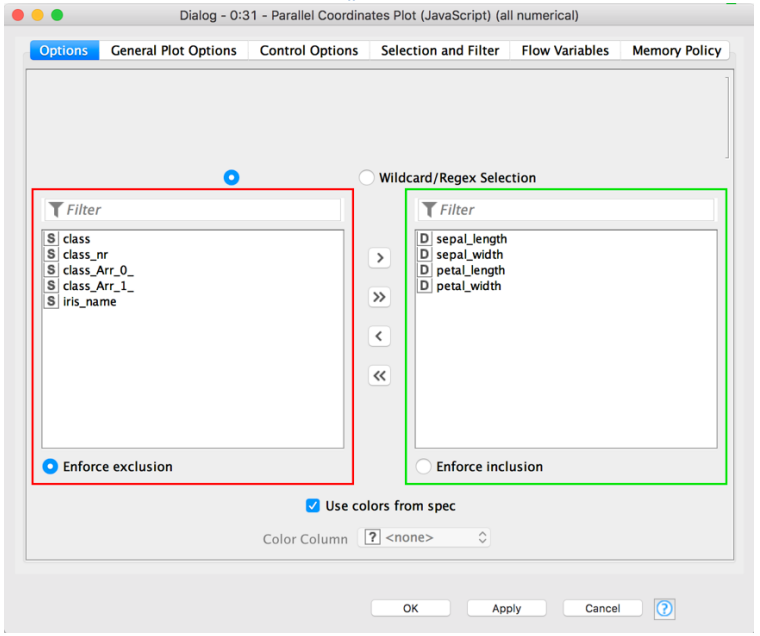
The "Parallel Coordinates (Javascript)" node displays the input data table in a parallel coordinates plot. A parallel coordinates plot unfolds the column names along the X-axis and displays each column value on a separated Y-axis. As a result a data point is mapped as a line connecting values across attributes.

The configuration window of this node has three tabs.

- "**Options**" tab contains an "Exclude/Include" frame to insert/remove more columns (i.e. Y-axis) into/from the parallel coordinates plot.
- "**General Plot Options**" tab defines general settings for the plot and the output image
- "**Control Options**" tab sets the interactivity level for the final view

Line colors can come from a specific column containing the color as a graphical property (that is the result of the "Extract Color" node) or just from the graphical property associated to each row (flag "use color from spec").

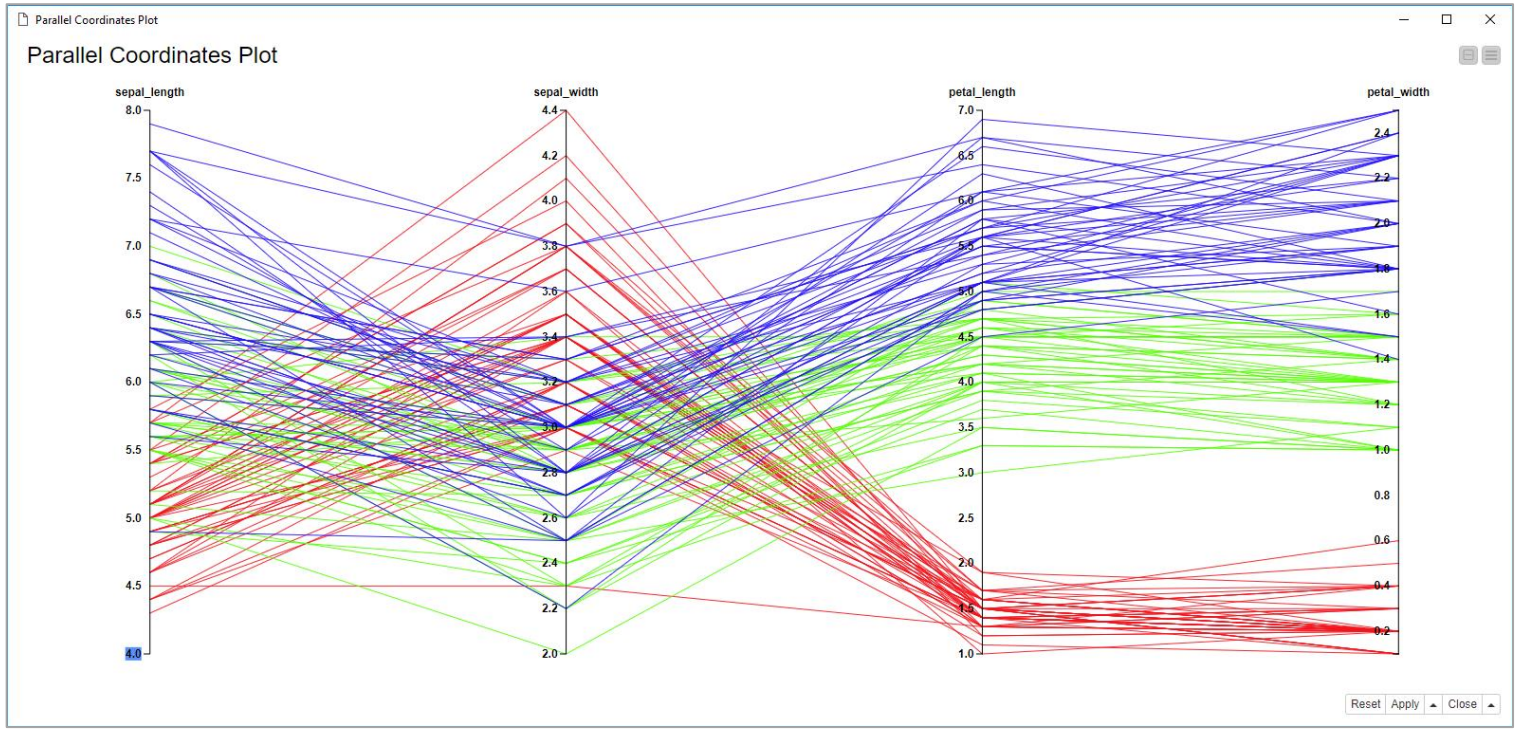
3.49. View of the „Parallel Coordinates" node



Below is the view of the "Parallel Coordinates (Javascript)" node. As Y-axis we find: sepal_length, sepal_wodth, petal_length, petal_width. Each iris plant is then described by the line connecting its sepal length, sepal_width, petal_length, and petal_width values. Line colors are determined by the color associated to each data row – i.e. to each iris plant – by the preceding "Color Manager" node.

Interactivity in the "Parallel Coordinates (Javascript)" node is also reduced with respect to, for example, the "Scatter Plot (Javascript)" node.

3.50. View of the "Parallel Coordinates (Javascript)" plot, where the 4 iris measures are displayed on the 4 Y-axis. One line corresponds to one iris plant



3.12. Bar Charts and Histograms

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. This section briefly takes a look at histograms and bar charts.

Properly speaking, there is not a dedicated Javascript based node to draw a histogram plot. The histogram drawing functionality is hidden in the “Bar Chart (Javascript)” node.

We already binned the “sepal_length” attribute in 9 bins. Now each data row of the input data table is assigned to a given bin according to the value of its “sepal_length” attribute. To build the histogram of attribute “sepal_length”, it is enough to count the number of occurrences in each “sepal_length_binned” interval with a “Pivoting” node.

Bar Chart (Javascript)

The “Bar Chart (Javascript)” node creates a generic bar chart. To do that, it needs:

- A category column, which in case of a histogram is the binned column
- An aggregation column and an aggregation method.

In case of a histogram the aggregation method is “Occurrence Count, This just counts the data rows falling in each bin and therefore does not require a specific aggregation column.

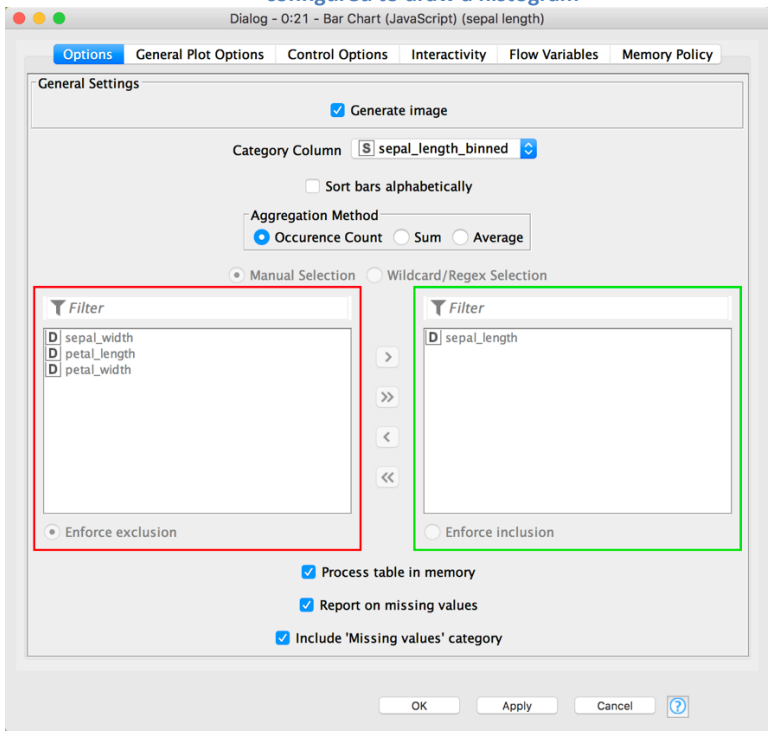
These settings are all defined in the tab “Options” of the configuration window. Two additional tabs “General Plot Options” and “Control Options” defines respectively the plot graphical details and the enabled view controls.

“General Plot Options” tab includes preferences for title, axis labels, plot orientation, legend, and output image size.

“Control Options” tab includes zooming, plot orientation change, title and label editing, bar stacking/grouping, and label stacking.

“Bar Chart (Javascript)” node has an optional input port for a color map.

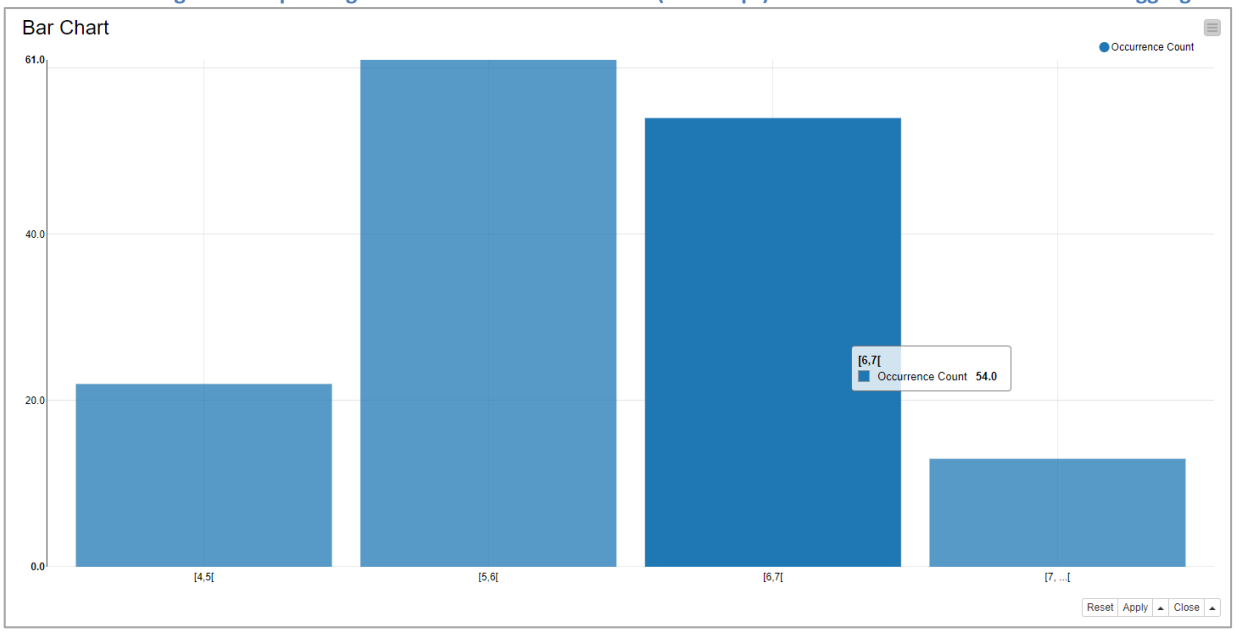
3.51. „Bar Chart (Javascript)“ node configuration window: “Options” tab configured to draw a histogram



The “Histogram” view displays how many times the values of a given column occur in a given interval (bin). The final histogram view is shown below.

Note. The “Bar Chart (Javascript)” node does not sort the string categories on the X-axis. They are displayed in occurrence order. If we want them to be sorted, like in our case of binning intervals, a “Sorter” node needs to precede the “Bar Chart (Javascript)” node.

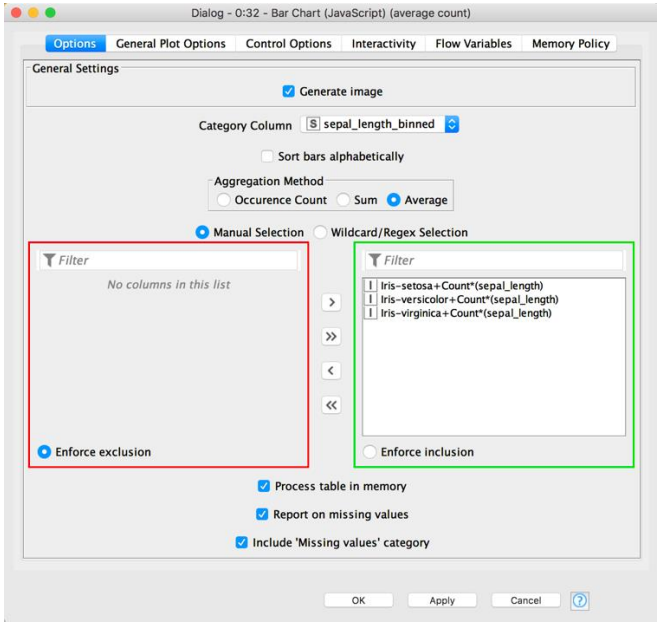
3.52. View of the Histogram of sepal length obtained with a "Bar Chart (Javascript)" node with "Occurrence Count" as aggregation method



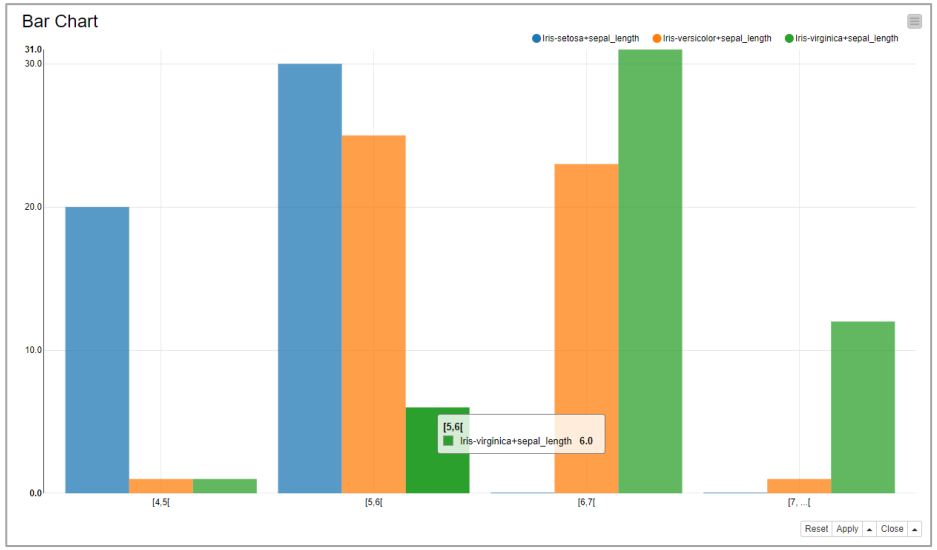
This histogram covers all instances of iris plants represented in the input data set. However, let's suppose we want to isolate and compare the same histogram for the three separate classes: iris-setosa, iris-versicolor, and iris-virginica.

First, we need to separate the three groups and count the number of occurrences for each group and for each bin in sepal_length ("Pivoting" node); finally we need to draw the counts into a bar chart ("Bar Chart (Javascript)" node with aggregation method "Average" on all three classes). Configuration window of "Bar Chart (Javascript)" node and consequent histograms view are reported below.

3.53. Configuration window of "Bar Chart (Javascript)" node: "Options" tab with aggregation method "Average on count of sepal_length in bin for all three iris classes



3.54. View of "Bar Chart (Javascript)" node showing the sepal length histograms for all three iris classes



The last node we would like to consider in this section is the “Table View (Javascript)” node. This node just displays the input data in a table.

Table View (Javascript)

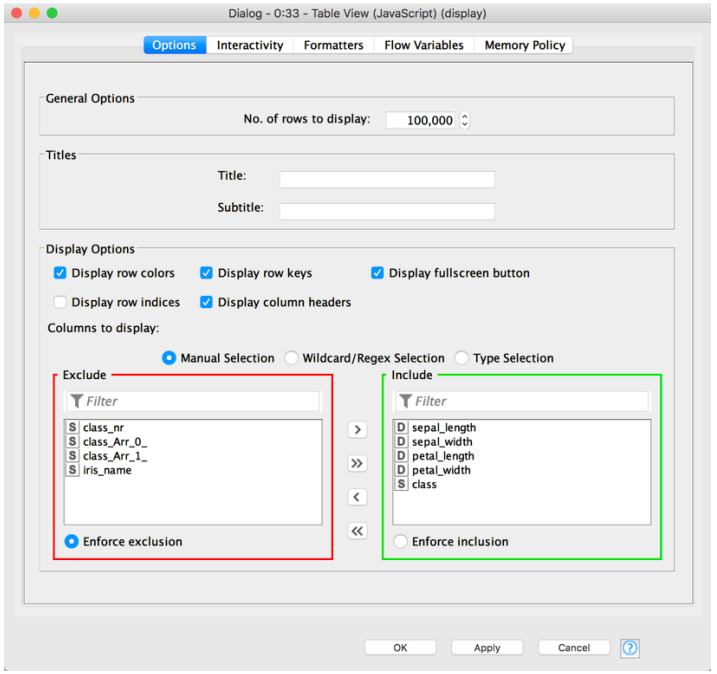
The “Table View (Javascript)” node displays the input data in a table.

The configuration window consists of three tabs:

- “***Options***” tab defines the data selection,for example which columns to display
- “***Interactivity***” tab contains the usual settings to determine the level of interactivity in the produced view
- “Formatters” tab provides a few formatting options for numbers, Strings, and Dates.

Depending on the settings in the “Interactivity” tab, the rows in the table view present a selection box on the left. In this way, it is possible to select only some of them. Selected rows will exhibit the flag “true” in the “Selected Javascript Table View” appended column at the node output port.

3.55. Table View of the “Interactive Table” node

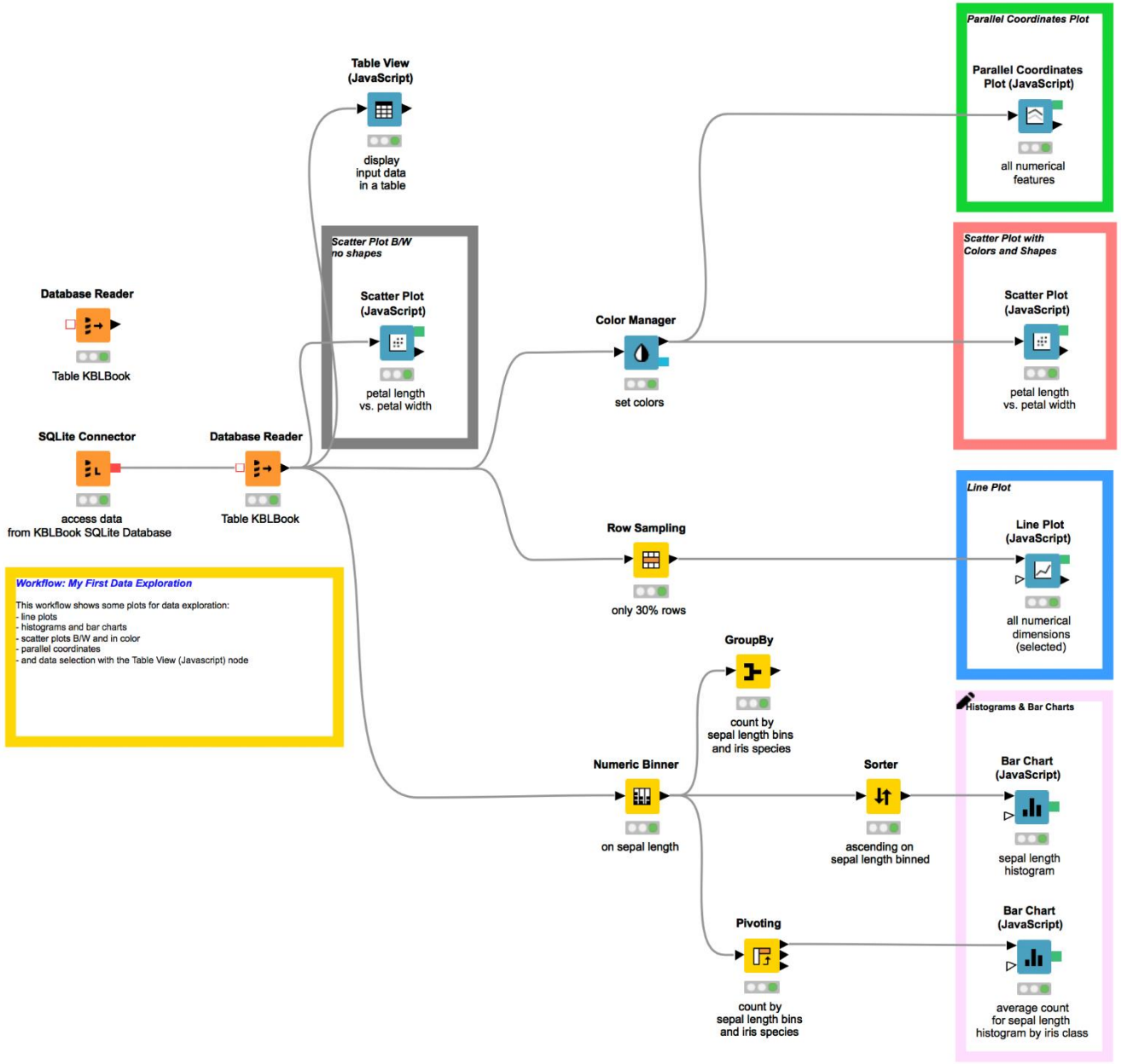


This is where we finish our description of the nodes available in KNIME Analytics Platform for data visualization. There are a few additional interesting visualization nodes, such as “Lift Chart”, “Box Plot”, “ROC Curve”, “Pie Chart”, etc ….

In particular, the “Generic Javascript View” node allows for free Javascript code. If you are a Javascript expert and/or you prefer to use some specific Javascript libraries, this is the node that allows to create arbitrarily complex Javascript based graphics.

This is the final workflow “My First Data Exploration”.

3.56. The final version of the workflow "My First Data Exploration"



3.13. Exercises

Exercise 1

Read file "yellow-small.data" from the Balloons Data Set (you can find this file in the KBLdata folder or you can download it from: http://archive.ics.uci.edu/ml/datasets.html).

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

Add the following classification column and name it "class":

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"

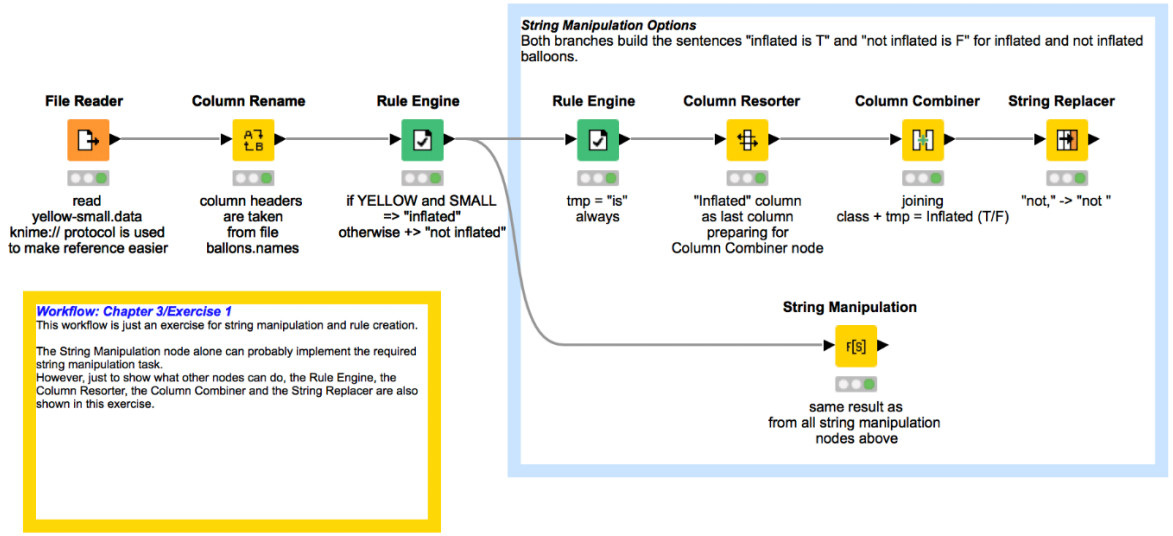
where "inflated/not inflated" comes from the "class" column and "T/F" from the "Inflated (True/False)" column.

Solution to Exercise 1

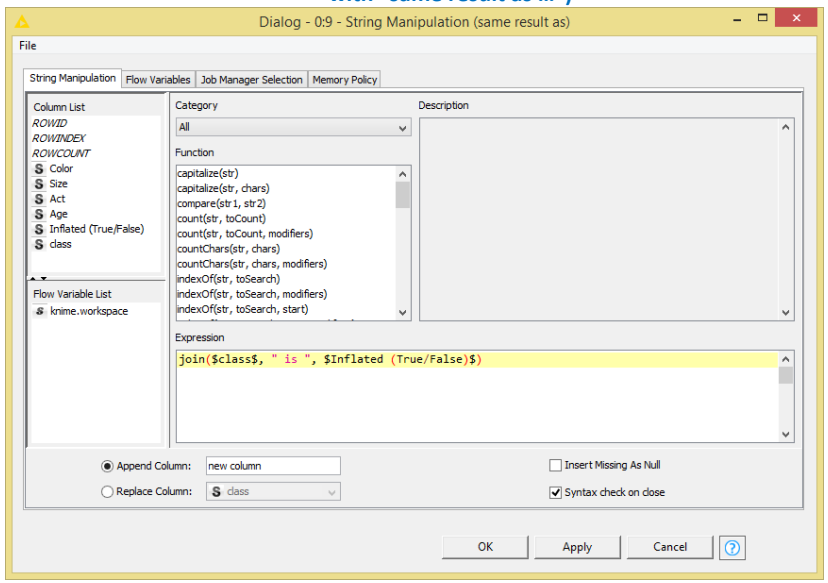
There are two ways to proceed in this exercise.

- With a series of dedicated "String Manipulation" and "Rule Engine" nodes
- With one "Rule Engine" and one "String Manipulation" node with its functions

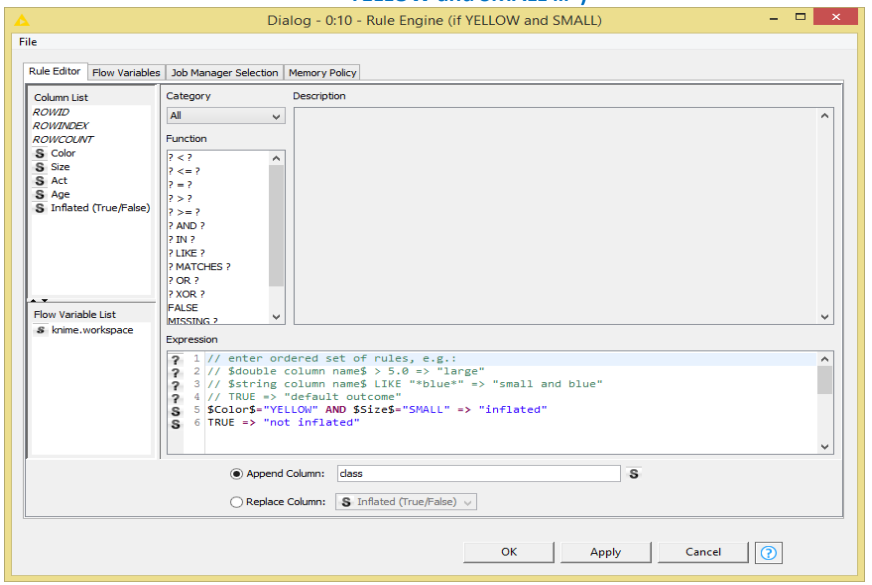
3.57. Exercise 1: Workflow



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



3.59. Exercise 1: The „Rule Engine“ node configuration (node commented with “if YELLOW and SMALL …”)



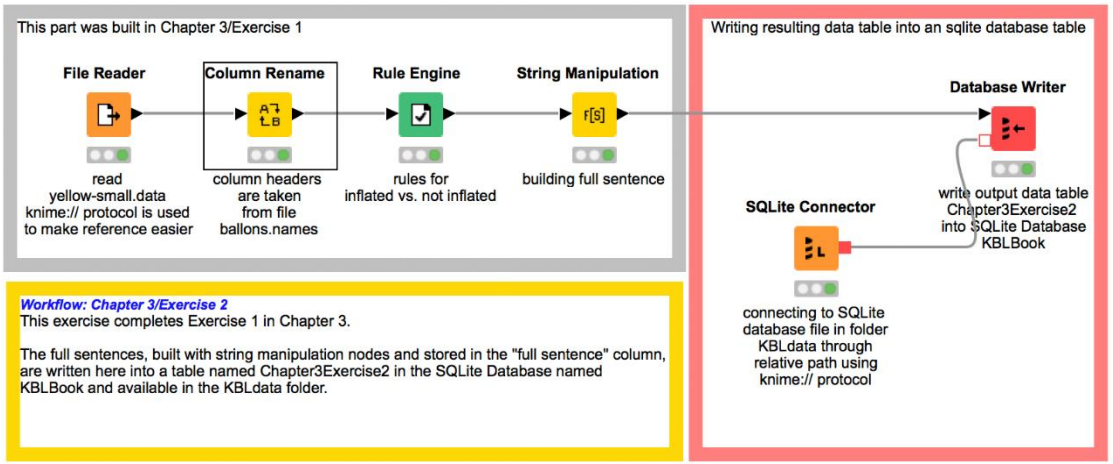
Exercise 2

This exercise is an extension of Exercise 1 above.

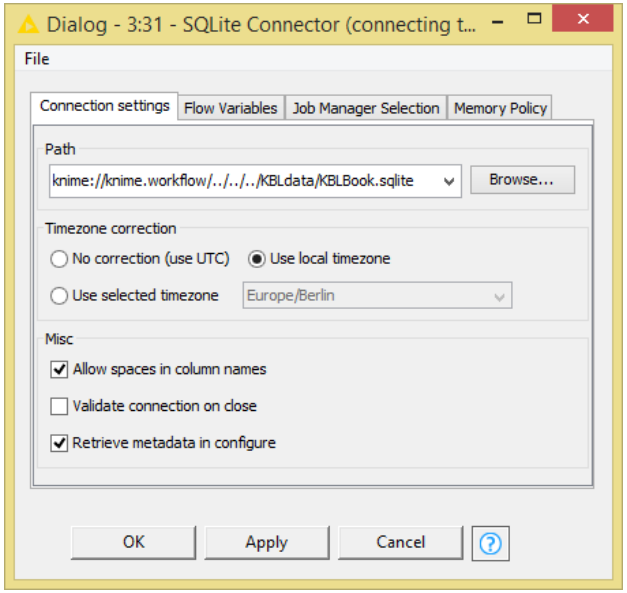
Write the last data table of workflow Exercise 1 into a table called “Chapter3Exercise2” in the SQLite database “KBLBook.sqlite”, using the “SQLite Connector” and the “Database Writer” node.

Solution to Exercise 2

3.60. Exercise 2: Solution Workflow



3.61. Exercise 2: Configuration Window of the “SQLIte Connector” node



Exercise 3

Read the **adult.data** file. From this data set display three plots:

- “Age” Histogram by sex on 10 age bins
- “Work class” Bar Chart as number of occurrences for each work class value
- “Average(capital gain)” vs. “hours per week” Scatter Plot

Build the histogram and the bar chart using a “Bar Chart (Javascript)” node and the scatter plot using a “Scatter Plot (Javascript)” node.
In the “age” vs. “hours per week” scatter plot, select all points with “age” = 90 and extract them with a “Row Filter” node on column “Selected”(…)” = “true”.
How many 90-year old people are included in the data set?

Solution to Exercise 3

Scatter Plot “age” vs. “hours per week”.

- In order to make sure that all records are plotted we need to change the default value of the setting “Maximum Number of Rows” in the “options” tab of the configuration window of the “Scatter Plot (Javascript)” node. We need to make sure that this number is bigger than the number of records in the input data set. Plotting all records instead of only the default number will of course require a longer execution time.
- In “Views Control” tab we need to enable rectangular selection. We open the node view, enable the selection button on the top right corner, and draw a rectangle around our 90-year old people on right of the scatter plot (if “age” has been placed on the x-axis). Then we click button “Close” in the lower right corner of the view and accept the changes.
- A “Row Filter” node finally extracts the records with “Selected (…)” column = true. 35 points representing 90-year old people have been selected.
- Optionally, we colored the dots in blue for male records and in red for female records with a “Color Manager” node.

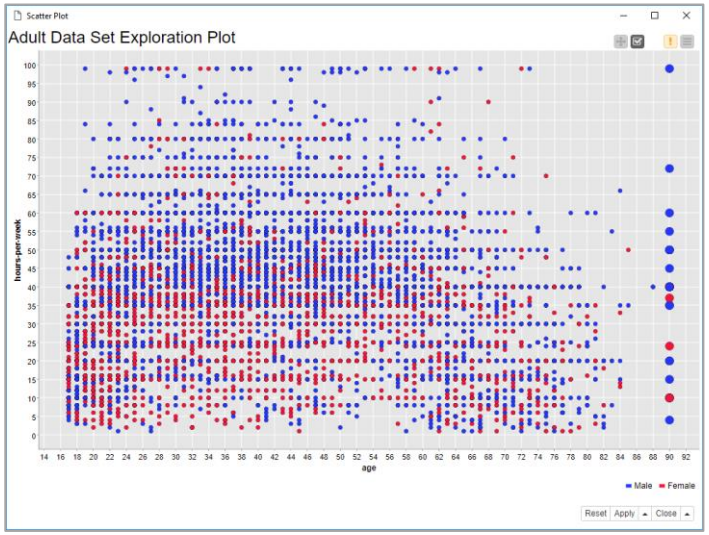
Bar Chart on Number of Occurrences in each work class.

- Here we used just a “Br Chart (Javascript)” node counting number of occurrences on category “workcalss”

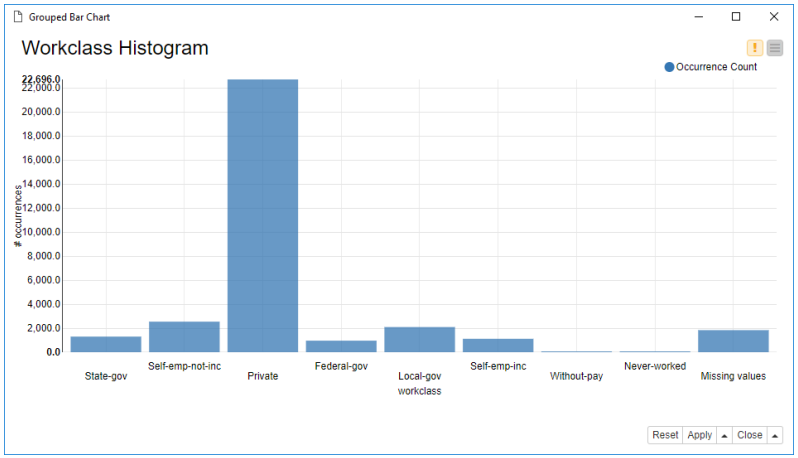
Age Histogram for Males and Females

- First we automatically build 10 age bins using the “Auto-Binner” node
- Then we use a “Pivoting” node to count the number of occurrences for men and women in the different age bins
- Using a “String Manipulation” node we change “[“ into (“ for sorting purposes and then we sort the age bins in ascending order
- Finally, a “Bar Chart (Javascript)” node displays the 2 numbers side by side for women and men. The side to side effect was obtained selecting “Grouped” as “Chart Type” setting in the “General Plot Options” tab in the node configuration window.

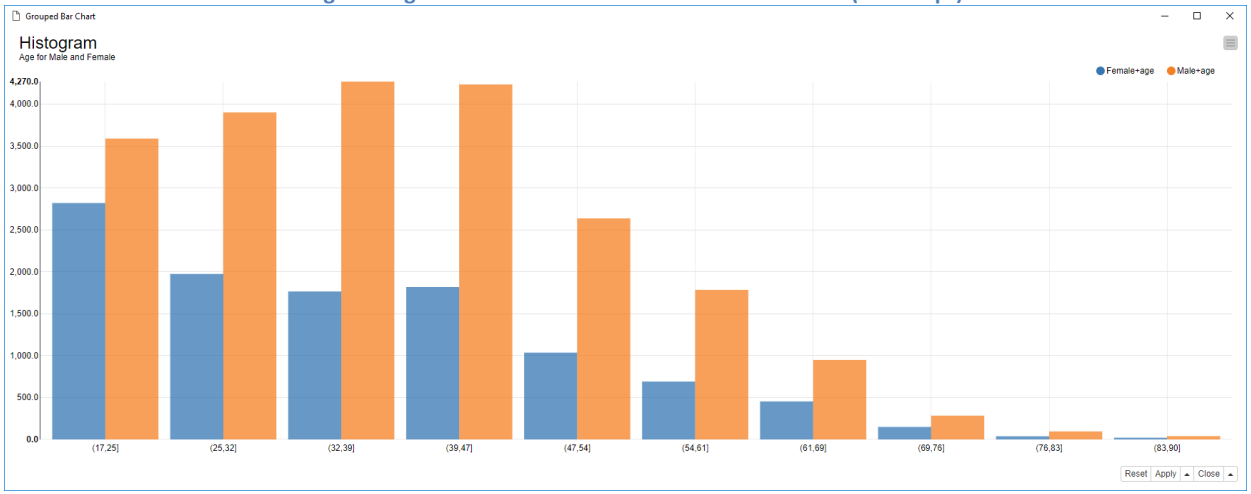
3.62. Scatter Plot of "age" vs. "hours per week" for the adult data set. 90-year old people have been selected.



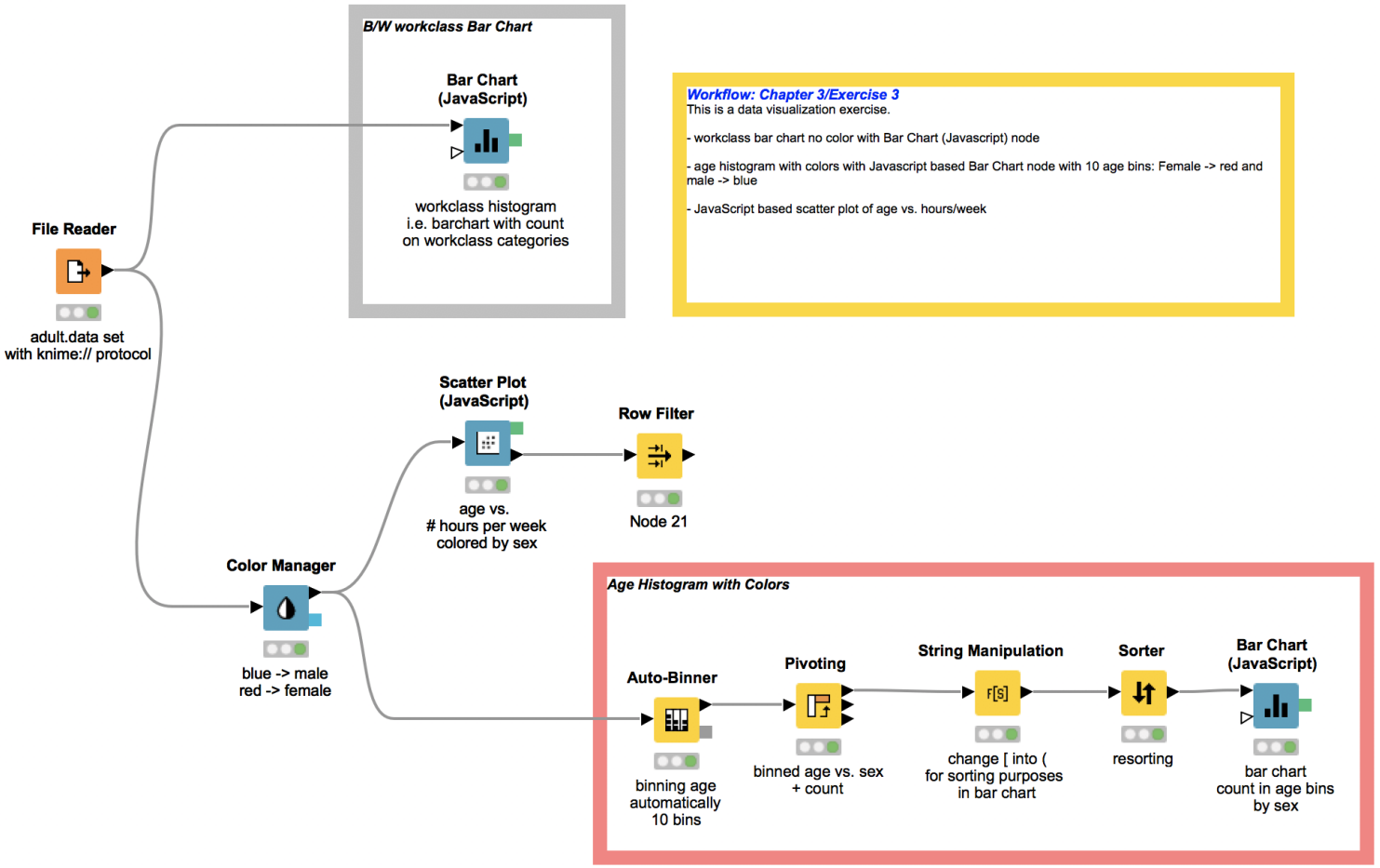
3.63. Bar Chart of number of occurrences of "work class" values in the adult data set.



3.64. Age Histogram for men and women from a “Bar Chart (Javascript)” node.



3.65. Exercise 3: Solution Workflow



Chapter 4. My First Model

4.1. Introduction

We have finally reached the heart of the KNIME Analytics platform: data modeling. There are two categories of nodes in the “Node Repository” panel fully dedicated to data modeling: “Analytics” -> “Statistics” and “Analytics” -> “Mining”. The “Statistics” category contains 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 consists of two phases: training the model on a set of data (the training data set) and applying the model to a set of new data (live data or a test data set). Complying with these two phases, data modelling algorithms in the KNIME Analytics Platform are implemented with two nodes: a “Learner” node to train the model and a “Predictor” node to apply the model. The “Predictor” node takes on another name when we are dealing with unsupervised training algorithms.

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

All data modeling algorithms need a training data set to build the model. Usually, after building the model, it is useful to evaluate the model quality, just to make sure we are not believing predictions produced by a poor model. For evaluation purposes, a new data set, named test data set, is used. Of course, the test data set has to contain different data from the training data set, to allow for the evaluation of the model capability to work properly onto unknown new data. For evaluation purposes, then, all modelling algorithms need a test data set as well.

In order to provide a training set and a test set for the algorithm, usually the original data set is partitioned in two smaller data sets: the learning/training data set and the test data set. To partition, reorganize, and re-unite 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. Indeed, not all modeling algorithms can deal with missing data. The model might also require the data set to have 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 “Data Manipulation” -> “Column” -> “Transform” category.

In this chapter, we provide an overview of data mining nodes, i.e. Learner and Predictor nodes, and of nodes to manipulate rows and transform values in columns. We work on the adult data set, already used in the previous chapters. Here we create a new workflow group “Chapter4” and, inside that, a new workflow called “Data Preparation”. We 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 training data and separated test data, these two data sets have to be set up before modeling the data. In order to extract two data sets - one for training and one for testing - from the original data set, the “Partitioning” node can be used. If only a training set is needed and not a test set or if the original data set is too big to be used wholly, we can use the “Row Sampling” node.

Row Sampling

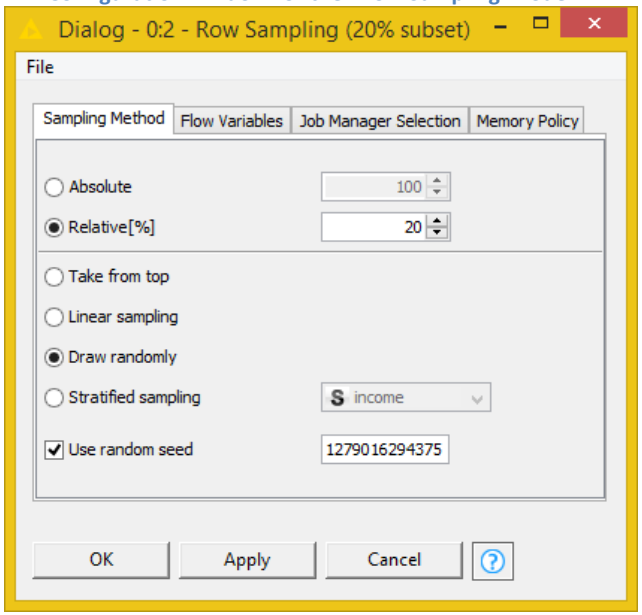
The “Row Sampling” node extracts a sample (= a subset of rows) from the input data.

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 top rows of the original data set
  - “Linear Sampling” takes the first and the last row and samples in between these rows at regular steps
  - “Draw randomly” extracts rows at random
  - “Stratified sampling” extracts rows randomly 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 exact same random training 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

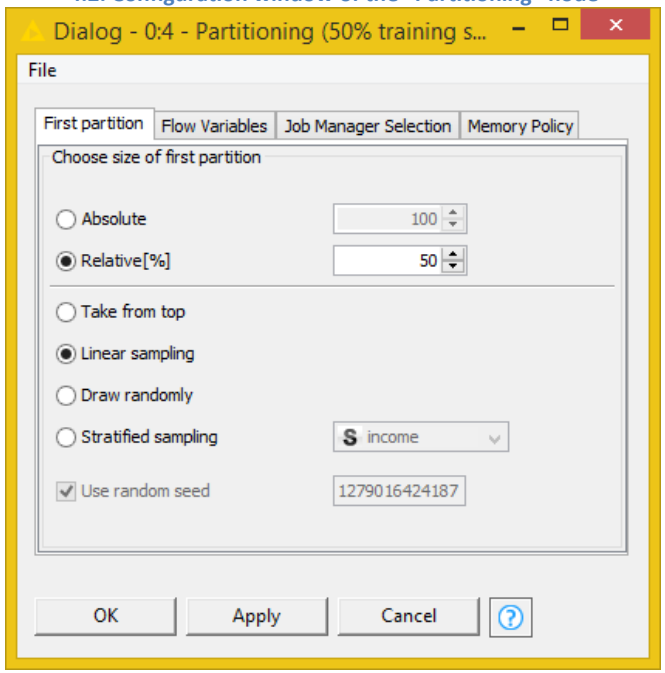
The "Partitioning" node performs the same task as the "Row Sampling" node: it extracts a sample (= a subset of rows) from the input data. It also builds a second data set with the remaining rows and makes it available at the lower output port.

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 training set plus a linear extraction mode. The training set was made available at the upper output port; the remaining data were made available at the lower output port. In the linear sampling technique, rows adhere to the order defined in the original data set.

Sometimes it is required to present the data rows in the original order to the training algorithm, for example, when dealing with time series prediction. The row order, in this case, is a temporal order and is used by the model to represent temporal sequences.

Sometimes, however, it is not advisable to present rows to a Learner node in a specific order; otherwise the model might learn the row order among all other underlying patterns. For example, the customer order in the database does not mean anything more than assigning a sequential identifying key to each customer. To be sure that data rows are presented to the model's Learner node in a random order, we can extract them randomly or apply the "Shuffle" node.

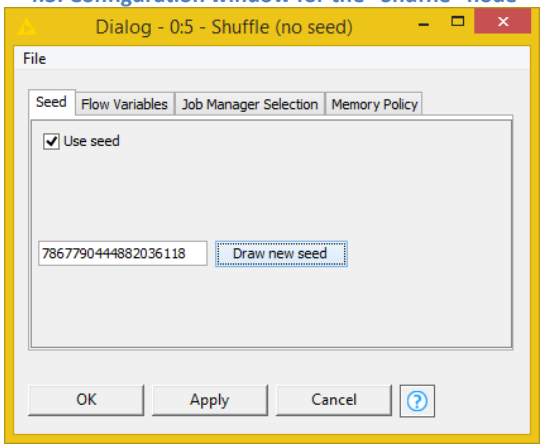
Shuffle

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

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 training data set and a test data 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 lower input port to the data set at the upper input port.

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)
- Option “Enable hiliting” refers to the hiliting property available in the old “Data Views” nodes.

4.4. Configuration window for the "Concatenate" node

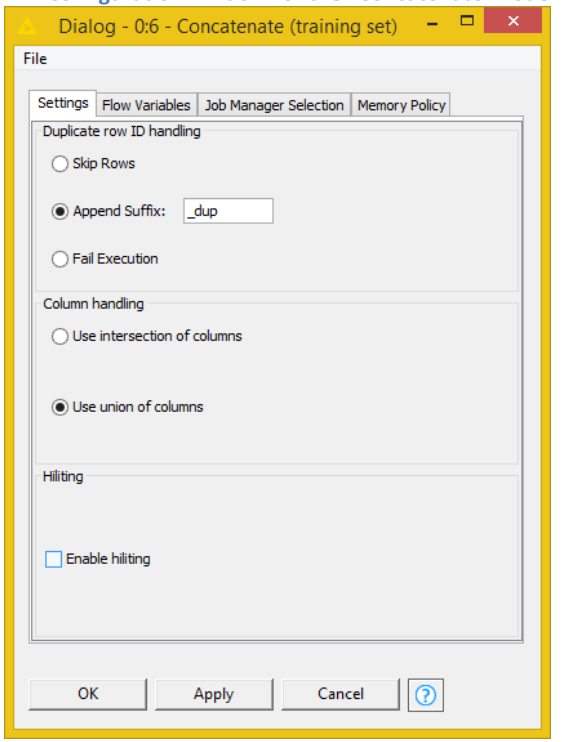


Figure 4.4 shows an example of how the “Concatenate” node works, when the following options in the configuration window are enabled:

- append suffix to RowID in rows with duplicate RowID
- use union of columns
- no hiliting enabled

A similar node to the “Concatenate” node is the “Concatenate (Optional in)” node. The “Concatenate (Optional in)” node works exactly the same as the “Concatenate” node, but allows to concatenate up to 4 data sets at the same time.

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

We have successfully derived a training set and a test set from the original data set. The original data set, though, contained missing values in some of its data columns and most data mining algorithms cannot deal with missing values. KNIME data cells, indeed, can have a special “missing value” status. By default, missing values are displayed in the table view with a question mark (“?”).

Some of the Learner nodes might ignore data rows containing missing values, therefore reducing the data basis they are working on; and some of the Learner nodes will just fail when encountering a missing value. In the last case, a strategy to deal with missing values is required; but even in the first case, a strategy to deal with missing values is advisable to expand the data basis for the future model.

There are many strategies to deal with missing values and books have been written about which strategy is best to use in which context. Each strategy consists in substituting the missing value in question with another plausible value. What is the most plausible value depends on the context and often on the expert knowledge.

The KNIME Analytics Platform implements the most common strategies to deal with missing values, such as using the data column mean value, moving average, maximum/minimum, most frequent value, linear and average interpolation, previous or next value, a fixed value, and probably by now more. Of course, the option of removing the data row containing a missing value is always available.

The node that deals with missing values is named “Missing Value”. The “Missing Value” node takes a data table as input and replaces missing values everywhere or only in selected columns with a value of your choice. The new data table with replaced missing values is then produced at the upper output port. Indeed, this node has two output ports. The lower output port is in the shape of a square blue rather than the usual white triangle. A blue square port means a PMML compliant model.

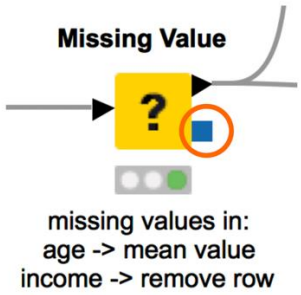
PMML

PMML (Predictive Model Markup Language) is a standard XML-based structure that enables the storage of predictive models and data transformations.

Since it is a standard structure, it enables the portability of models and transformations across platforms and applications.

KNIME Analytics platform supports PMML for models and transformations. The blue squares as input and output ports in KNIME nodes identify PMML compliant objects, be it predictive models or ETL transformations.

4.6. The „Missing Value“ node



In KNIME it is not only possible to export models and single transformations as PMML structures, but also to modularly concatenate them so that the final PMML structure contains the sequence of transformations and the model created in the workflow and fed into the PMML structure. Two nodes are key for modular PMML: “PMML Transformation Appender” and “PMML Model Appender”.

Note. Some of the missing value strategies are marked with an asterisk in the menus of the configuration window in the “Missing Value” node. The asterisk indicates that such transformations are not PMML supported.

Missing Value

The “Missing Value” node replaces missing values in a data set everywhere or only in selected columns with a value of your choice.

In **tab “Default”**, replacement values are defined separately for numerical and string type columns and applied to the all data columns of the same type.

In **tab “Column Settings”**, a replacement value is defined specifically for each selected data column and applied only to that column. To define the replacement value for a column:

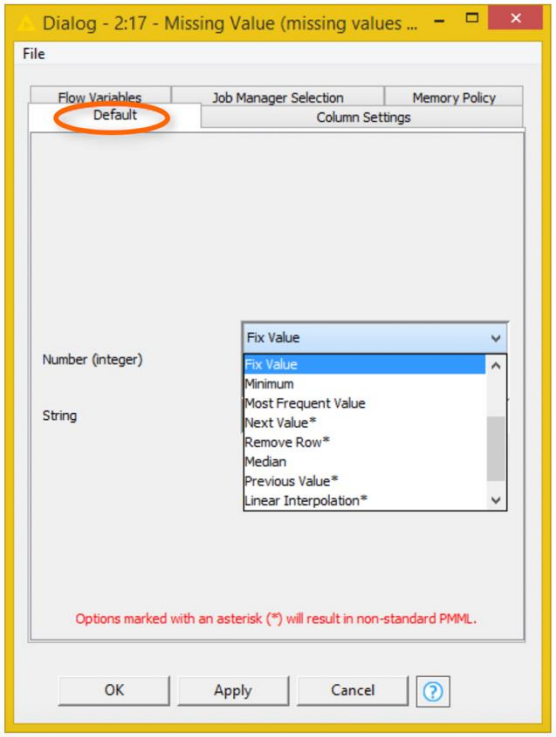
- Double-click the column in the list on the left

OR

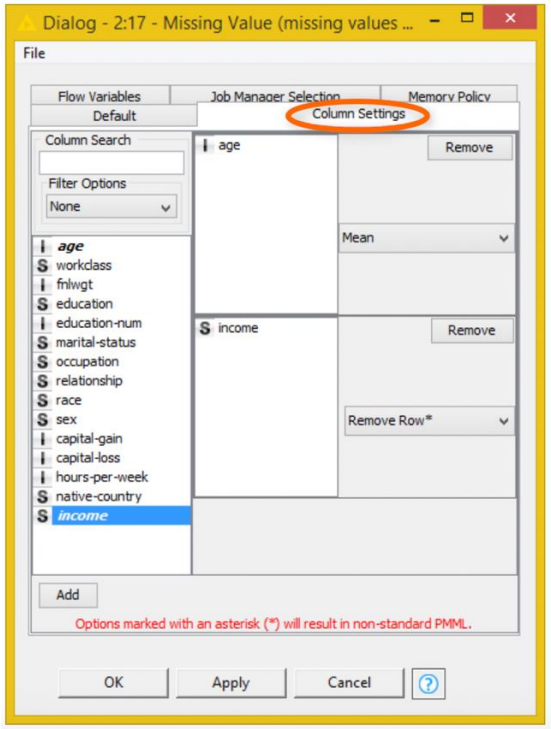
- Select the column from the list on the left
- Click the “Add” button under the list

Then, select the desired missing value handling strategy.

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



4.8. Configuration window for „Missing Value“ node: Tab “Column Settings”



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.

We introduced a “Missing Value” node prior the “Partitioning” node in our “Data Preparation” workflow. Here we set 0 as the fixed value to replace missing values in all numerical columns and “Do nothing” for missing values in String columns. Then, for column “age” (Integer) and “income” (String), we set individual replacement strategies for missing values. In column “age” missing values are replaced by the data column mean value; in column “income”, rows with missing values are simply removed. While the missing value strategy for “age” is purely demonstrative, the missing value strategy for “income” is necessary, since we want to predict the “income” value given all other census attributes for each person.

Some data models - such as neural networks, clustering, or other distance based models - require 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.

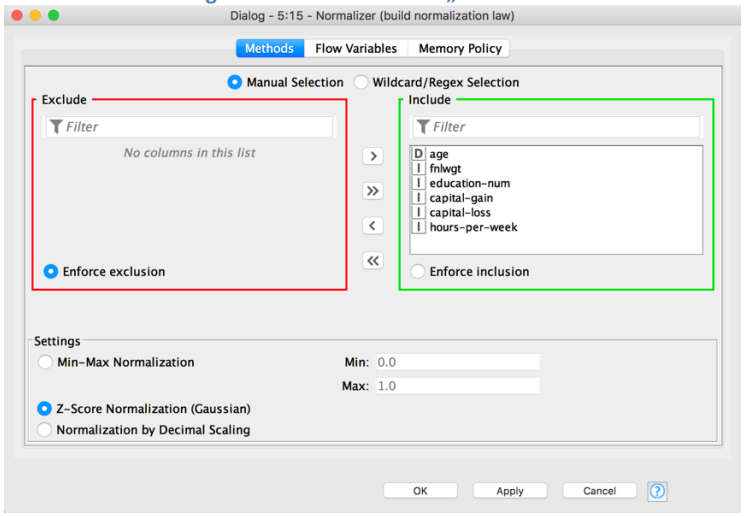
The configuration window requires:

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

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 viceversa, use buttons "add" and "remove". To move all columns to one frame or the other use buttons "add all" and "remove all".

4.9. Configuration window of the „Normalizer" node



The "Normalizer" node has 2 output ports:

- At the upper port we find the normalized data
- At the lower port the transformation parameters are provided to repeat the same normalization on other data (light blue/dark blue square port)

Note. Triangular ports output/read data. Squared ports output/read parameters: model's parameters, normalization parameters, transformation parameters, graphics parameters, etc …

There are two normalizer nodes: "Normalizer" and "Normalizer (PMML)" node. They perform exactly the same task using the same settings. The only difference is in the exported parameter structure: KNIME proprietary structure (light blue square) or PMML compliant structure (dark blue square).

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.

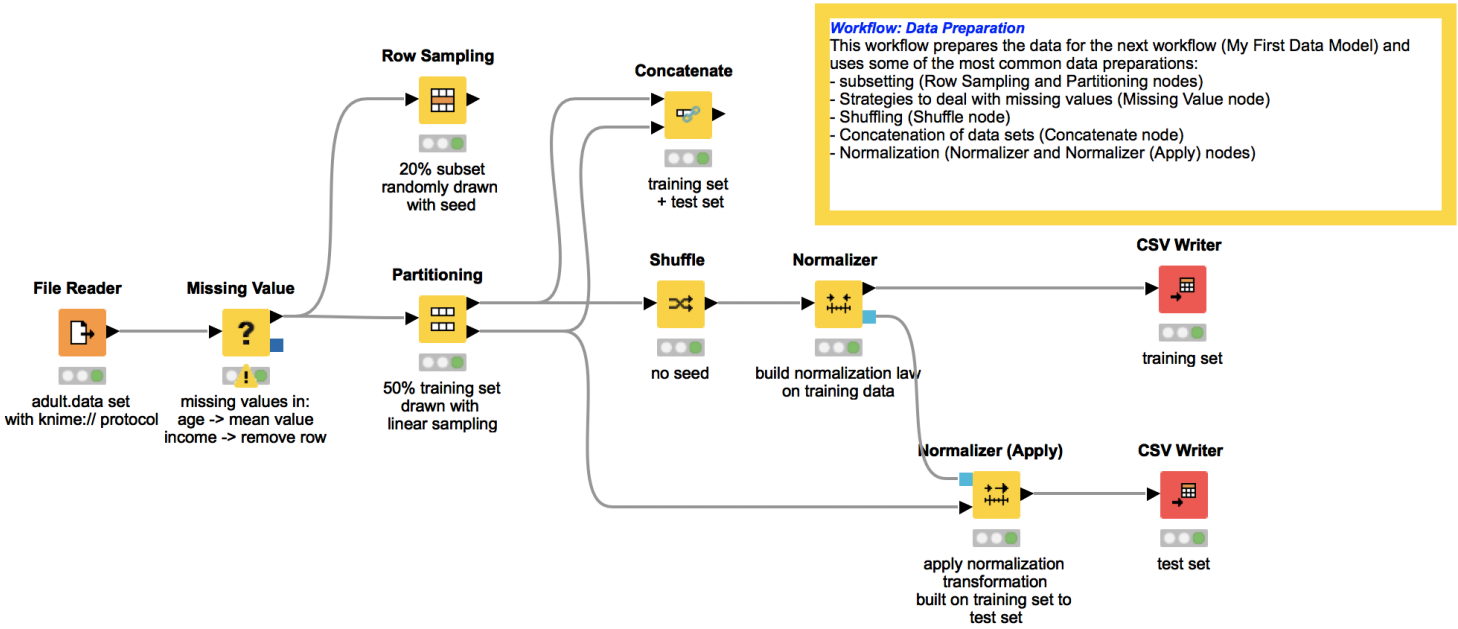
The "Normalizer(Apply)" node is located in the "Node Repository" panel in the "Data Manipulation" -> "Column" -> "Transform" category.

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 introduced a "Normalizer (Apply)" node to read the normalization parameters and to use them to normalize the remaining data from the "Partitioning" node (2nd output port).

Let's now write the processed training data set and test data set into CSV files, named "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.

4.10. "Data Preparation" workflow



4.4. Data Models

Now let's create a new workflow and call it "My First Model". We will use this workflow to show how 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 "income" label of the adult data set by using the other attributes and based on a few different models. This section does not intend to compare those 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 supervised prediction/classification problem, we need a labelled training set; that is a training set where each row has been 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 or target column.

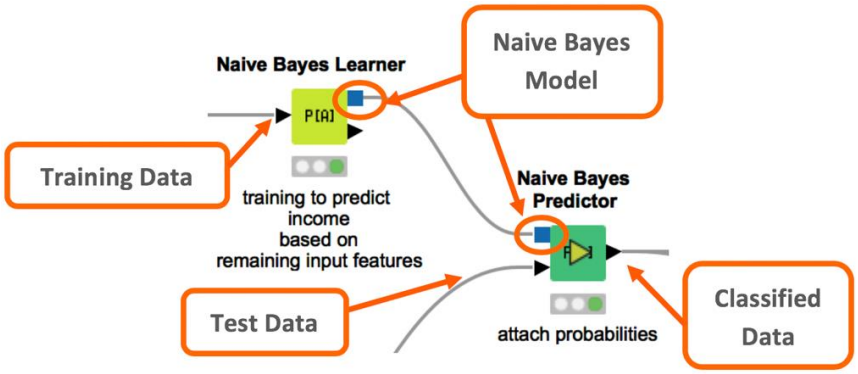
Most data mining and statistics paradigms consist of two nodes: a Learner and a Predictor.

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 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 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 blue square, which is the symbol for a PMML compliant model.

4.11. 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 blue square for the model) and provides a data table containing the classified data at the output port.

Naïve Bayes Model

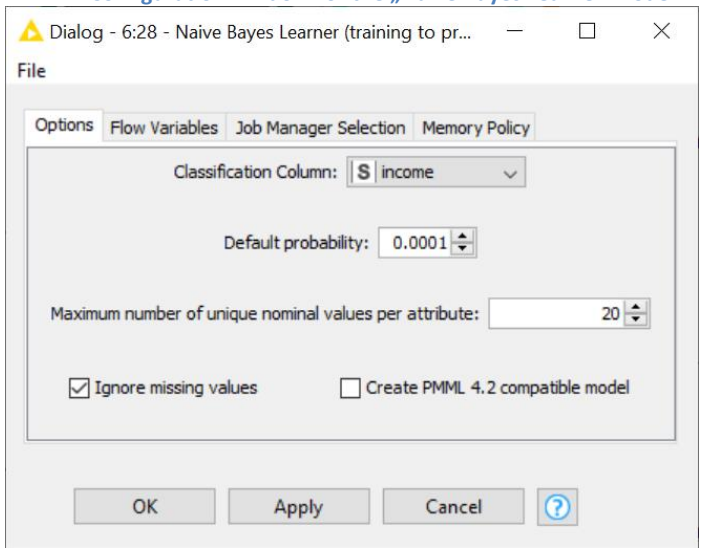
Let's start with a naïve Bayes model. A Bayesian model defines a set of rules, based on the Gaussian distributions and on the conditional probabilities of the input data, to assign a data row to an output class [3][4][5]. In the "Node Repository" panel in the "Mining" -> "Bayes" category we find two 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)
- The default probability (almost 0)
- 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.
- How to deal with missing values (skip vs. keep)
- Compatibility of the output model with PMML 4.2

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



Naïve Bayes Predictor

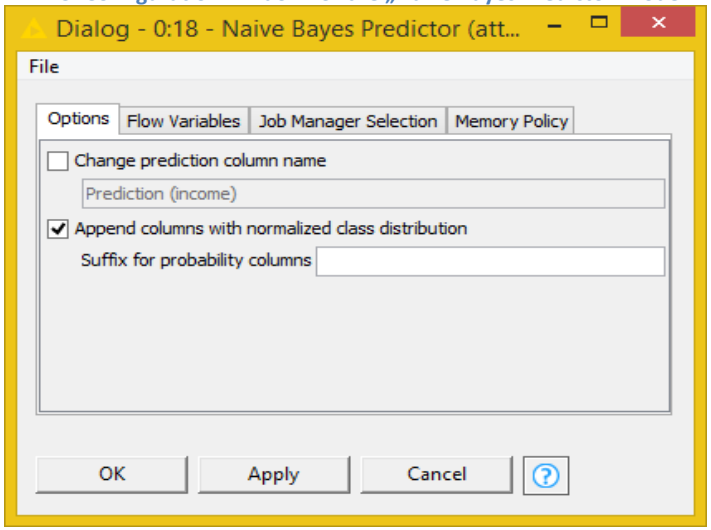
The "Naïve Bayes Predictor" node applies an existing Bayesian model to the input data table.

All necessary configuration settings are available in the input model.

In the configuration window you can only:

- Append the normalized class distribution values for all classes to the input data table
- Customize the column name for the predicted class

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



Note. All predictor nodes expose the same configuration window: one option to append predicted class probabilities/normalized distributions and one option to change the default prediction class column name.

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. In the configuration window of the “Naïve Bayes Learner”, we specified “income” as the class/target 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 setting 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 as from configuration settings) nominal values and will be ignored. We then run the “Execute” option for the “Naïve Bayes Learner” node.

The next step involves connecting a “Naïve Bayes Predictor” node to the “File Reader” node to read the 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.

4.14. Bayes Model's Classified Data

The classified data - 0:18 - Naive Bayes Predictor (attach probabilities)

File

Table "default" - Rows: 15363 | Spec - Columns: 18 | Properties | Flow Variables

Row ID	hours-p...	native-...	income	P (inco...	P (income=>50K)	Prediction (income)
Row0	-2.346	United-States	<=50K	0.149	0.851	>50K
Row1	-0.085	United-States	<=50K	0.99	0.01	<=50K
Row2	-0.085	United-States	<=50K	0.047	0.953	>50K
Row3	0.334	United-States	>50K	0.16	0.84	>50K
Row4	-0.085	United-States	>50K	0	1	>50K
Row5	-0.085	India	>50K	0.055	0.945	>50K
Row6	0.753	United-States	<=50K	0.99	0.01	<=50K
Row7	0.334	Mexico	<=50K	0.995	0.005	<=50K
Row8	-0.085	United-States	<=50K	0.999	0.001	<=50K
Row9	0.334	United-States	>50K	0.674	0.326	<=50K
Row10	-1.76	United-States	<=50K	1	0	<=50K
Row11	-0.085	United-States	<=50K	0	1	>50K
Row12	-0.085	United-States	>50K	0.034	0.966	>50K
Row13	3.266	United-States	<=50K	0.941	0.059	<=50K
Row14	0.92	United-States	<=50K	0.979	0.021	<=50K
Row15	-0.085	United-States	<=50K	0.006	0.994	>50K
Row16	-2.179	United-States	<=50K	0.998	0.002	<=50K

After execution, the “Naïve Bayes Predictor” shows a new column appended to the output table: “Prediction (income)”. 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 “income”.

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 “Analytics” -> “Mining” -> “Scoring” category with nodes that measure the classifiers’ performances. The most straightforward of these evaluation nodes is the “Scorer” node. We will use the “Scorer” node to measure the performances of the Bayesian classifier.

Scorer (JavaScript)

The “Scorer (JavaScript)” node compares the values of two columns (target column and prediction column) in the data table; based on this comparison it shows the confusion matrix and some accuracy measures.

This node produces three output data tables: the confusion matrix, the statistics of correctly identified rows for each class, and the overall accuracy measures as set in the configuration window.

This node has a View option, where the confusion matrix and some accuracy metrics are displayed.

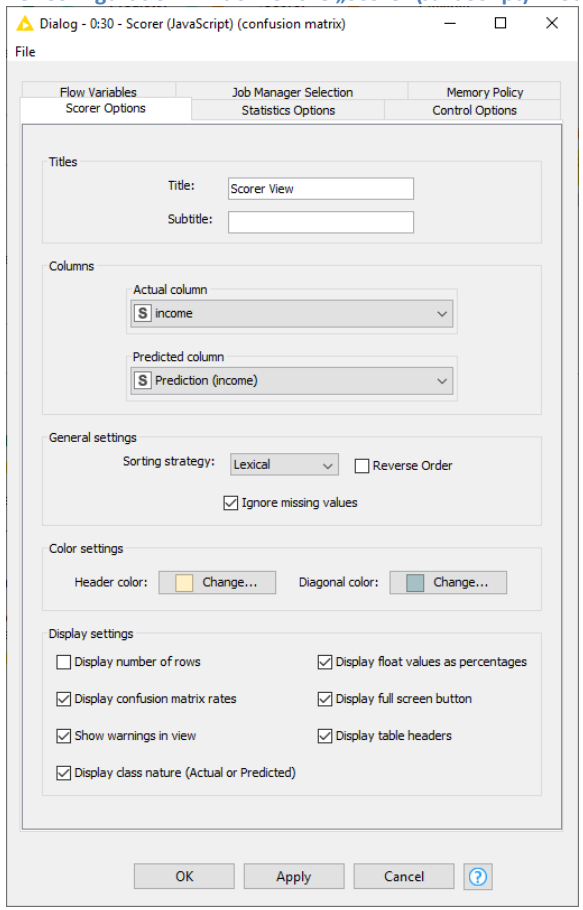
The configuration window has three tabs: “Scorer Options”, “Statistics Options”, “Control Options”.

Tab “Scorer Options” requires the selection of the two columns to compare (“Actual Column” and “Predicted Column”) and the sorting to be used in the evaluation strategy. The flag “Ignore missing values”, if unchecked, makes the node fail if missing values are encountered in one of the two columns to compare. All other options are related to the display of the node view.

Tab “Statistics Options” includes all accuracy measures and false/true positive/ negative numbers to calculate.

Like all JavaScript based nodes, this node produces a view with some degree of interactivity. Interactivity options are defined in tab “Control Options”.

4.15. Configuration window of the „Scorer (JavaScript)“ node



We added a “Scorer (JavaScript)” node into the workflow “My First Model”. The node is connected to the data output port of the “Naïve Bayes Predictor”. The first column with the original reference values is “income”; the second column with the class estimation is the column called “Prediction (income)” which is produced by the “Naïve Bayes Predictor” node. During execution, values are then compared row by row and the confusion matrix and the consequent accuracy measures are calculated.

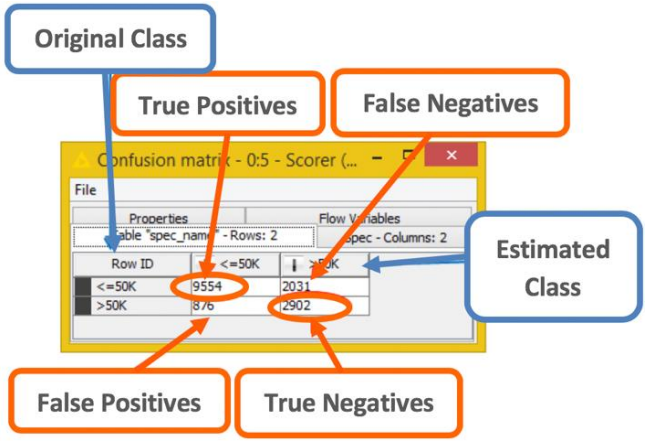
We can see the confusion matrix and the accuracy measures for the compared columns by selecting either the last three items or the item “Interactive View: Confusion Matrix” in the context-menu of the “Scorer (JavaScript)” node.

Confusion Matrix

In Figure 4.16, you can see the confusion matrix generated by the "Scorer" node. The confusion matrix shows the number of matches between the values in the target column and the values in the predicted column.

The values found in the target column are reported as Row IDs; the values found in the predicted column are reported as column headers. Since "income" has only two possible values – ">50K" and "<=50K" – the reading of the confusion matrix is quite simple.

4.16. True Positives, False Negatives, True Negatives, and False Positives in the Confusion Matrix for "<=50K" as the positive class



The first cell contains the number of data rows that had an income "<=50K" and were correctly classified as having an income "<=50K". The last cell, the one identified as (">50K", ">50K"), contains the number of data rows with an income ">50K" and that were correctly classified as having an income ">50K". The other two cells represent the number of data rows with original income "<=50K" and incorrectly classified as having an income ">50K" and viceversa.

The cells along the diagonal from the top left corner to the lower right corner contains the numbers of correctly classified events. The opposite diagonal, the one from the top right corner to the lower left corner, contains the numbers of incorrectly classified events, that is the errors that we want to minimize.

The sum across one row of the confusion matrix indicates the total number of data rows in one class according to the labels in the original data set. The sum across one column indicates the number of data rows assigned to one class by the 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.

In our "Scorer" node, we selected the first column as the target classification column "income" and the second column as the output column of the Bayesian classifier. Thus, this confusion matrix says that 9554 data rows were correctly classified as having an income "<=50K; 2902 were correctly classified as having an income ">50K"; and 876 and 2031 data rows were incorrectly classified.

Accuracy Measures

The second port of the "Scorer" node presents a number of accuracy measures [6] [7]. In a binary classification (or in any classification), we need to choose one of the classes as the positive class. Such choice is completely arbitrary and usually dictated by the data context. Once one of the classes has been assumed as the positive one, the following definitions can take place:

True Positives is the number of data rows belonging to the positive 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 the positive class in the original data set and are classified as not belonging to that class.

False Positives is the number of data rows that do not belong to the positive class but are classified as if they do.
False Negatives is the number of data rows that belong to the positive class but are assigned to a different class by the model.

In our case, if we arbitrarily choose "<=50K" as the positive class, the *True Positives* are in the first cell, identified by ("<=50K", "<=50K"); the *False Negatives* are in the adjacent cell; the *False Positives* are below it; and the *True Negatives* are in the remaining diagonal cell.

On the basis of these True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) numbers, a number of correctness measures can be defined, each measure enhancing some aspect of the correctness of the classification task.

These accuracy measures are provided in the lower output port of the "Scorer" node.

Let's see how they are defined.

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))

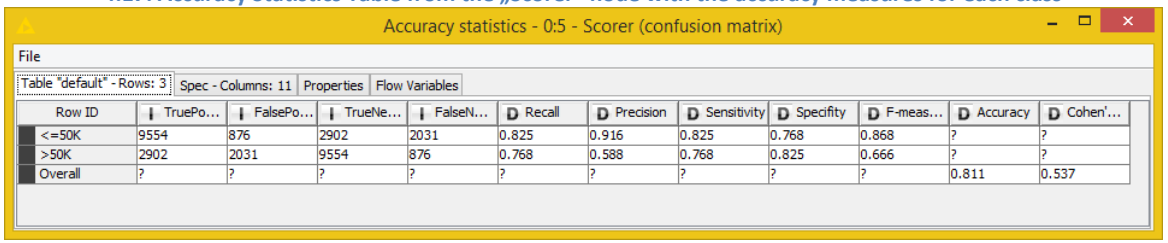
being TP = True Positives, FP = False Positives, TN = True Negatives, and FN = False Negatives.

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). The Cohen's kappa gives a more balanced accuracy estimation in case of strong differences in the class distributions.

"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.

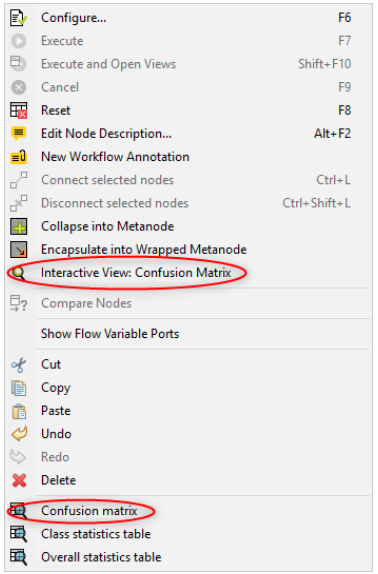
All these accuracy measures are reported in the data table in the second port at the bottom of the "Scorer" node and give us information about the correctness and completeness of our model.

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



Row ID	TruePo...	FalsePo...	TrueNe...	FalseN...	Recall	Precision	Sensitivity	Specifity	F-meas...	Accuracy	Cohen'...
<=50K	9554	876	2902	2031	0.825	0.916	0.825	0.768	0.868	?	?
>50K	2902	2031	9554	876	0.768	0.588	0.768	0.825	0.666	?	?
Overall	?	?	?	?	?	?	?	?	?	0.811	0.537

4.18. The context menu of the "Scorer" node



View: Confusion Matrix

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

Item "Interactive View: Confusion Matrix"

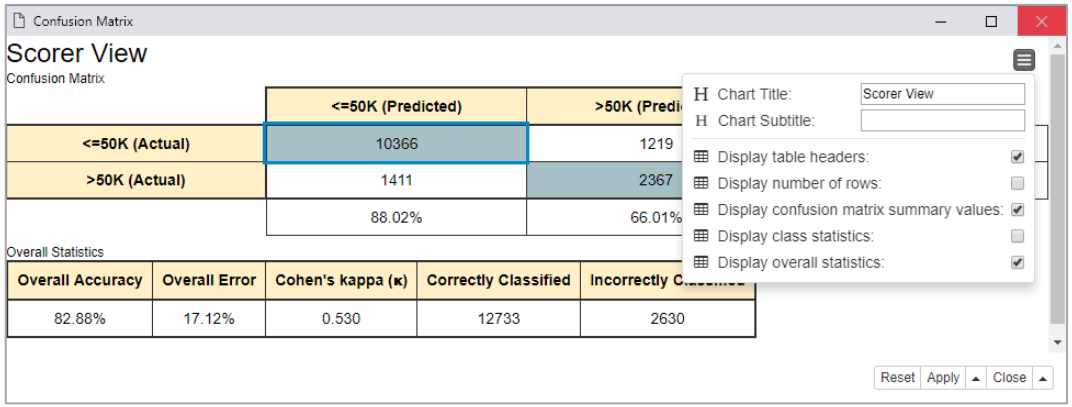
Item "Confusion Matrix"

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 "Interactive View: Confusion Matrix" includes a few more options. Let's have a look at the "Interactive View: Confusion Matrix" window.

The View includes a title and a subtitle - as defined in the configuration window – in the top left corner. Then right below come the confusion matrix, performance statistics by class, and the overall statistics measures, at least those that have been selected in the tab "Statistics Options" of the configuration window.

Like all JavaScript based nodes in KNIME Analytics Platform, also this node includes a menu button in the top right corner. The menu opening, after clicking this button, allows you to change the view display, such as title and subtitle, but also to include (or not) summary values, class statistics, overall statistics and so on. Finally, cell content in the confusion matrix is selectable.

4.19. Confusion Matrix from the Interactive View of the "Scorer (JavaScript)" node.



Decision Tree

Using the same workflow "My First Model", let's now apply another quite popular classifier: a decision tree [8] [9].

The Decision Tree algorithm is a supervised algorithm and therefore 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.:

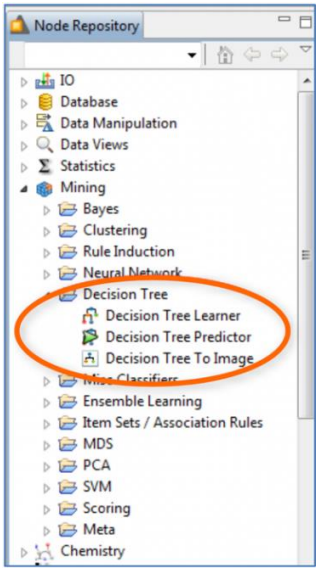
- The "Decision Tree Learner" node
- The "Decision Tree Predictor" node

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

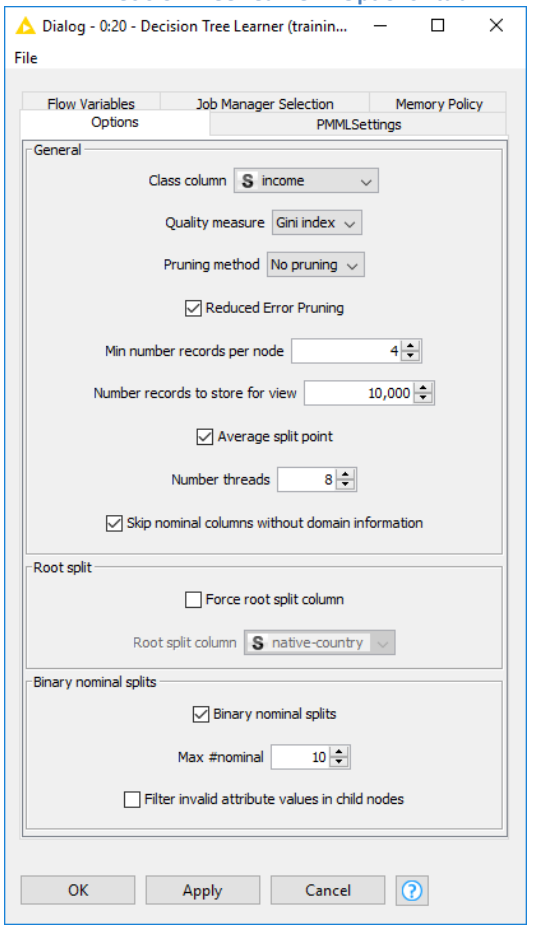
4.20. Two nodes implement a Decision Tree: the "Decision Tree Learner" and the "Decision Tree Predictor"



- A model input (blue square) with the model parameters produced by a “Decision Tree Learner” node

Decision Tree Learner: Options Tab

4.21. Decision Tree Learner: “Options” tab



The “Decision Tree Learner” node builds a decision tree from the input training data. In the configuration window you need to specify:

General

The *class column.* The target attribute must be nominal (String).

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 further splitting of this branch. The higher the number, the shallower the tree.

The *number of records to store for view*: the maximum number of rows to store for the hilite functionality. A high number slows down the algorithm execution.

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)

The *Number of threads* on which to run the node (default Number threads = 2 * number of processors available to KNIME).

Root Split

If you know that one attribute must be important for the classification, you can force it on the root node of the tree, by enabling “Force root split column” and selecting the “Root split column”.

Binary nominal splits

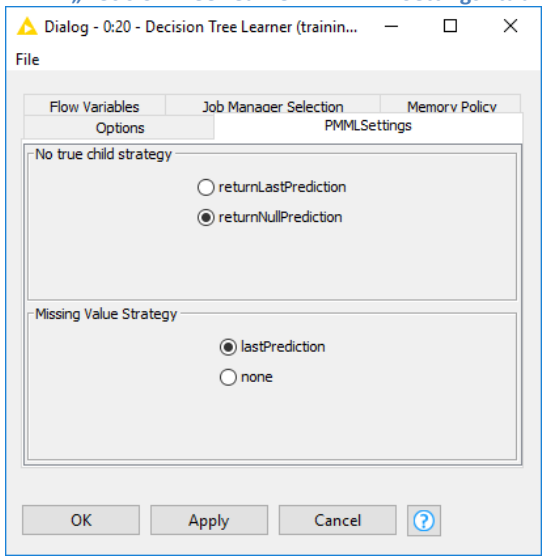
Here you can define whether *Binary nominal splits* apply to nominal attributes. In this case you can set the threshold *Maximum # of nominal* splits, up to which an accurate split is calculated instead of just a heuristic. The heuristic, though less precise, reduces the computational load.

"Filter invalid attribute values …" inspects the tree at the end of the training procedure and removes possible duplicates and incongruences.

Decision Tree Learner: PMML Settings Tab

The configuration window of the "Decision Tree Learner" node offers two tabs: "Options" (described above) and "PMML Settings". The "PMML Settings" tab deals with settings for the final PMML model.

4.22. „Decision Tree Learner": "PMML Settings" tab



How to deal with the no true child problem

Sometimes, the evaluation process reaches a node in the tree for which the required attribute shows an out of training domain value. In this case, for the predicted class you can:

- Use the majority class in the previous node ("returnLastPrediction" option)
- Return a missing value ("returnNullPrediction" option)

How to deal with missing values

Sometimes, the evaluation process reaches a node in the tree for which the required attribute shows a missing value. In this case, for the predicted class you can:

- Use the majority class in the previous node ("lastPrediction" option)
- Revert to the no true child strategy ("none" option)

We trained the "Decision Tree Learner" node with:

Class column = "income"
Gini Index as quality measure
Pruning = No Pruning
Stopping criterion = 4 data points per node
Number of records for hiliting = 10000
Split point calculated as the average point between the two partitions
Binary splits for nominal values
Maximum number of distinct nominal value allowed in a column = 10

8 as number of threads, since we are working on a 4-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.

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, like all other predictor nodes. The “Decision Tree Predictor” node was introduced to get the test data from the “File Reader” node and the model from the “Decision Tree Learner” node, with the option to append the normalized class distribution at the end of the prediction data table.

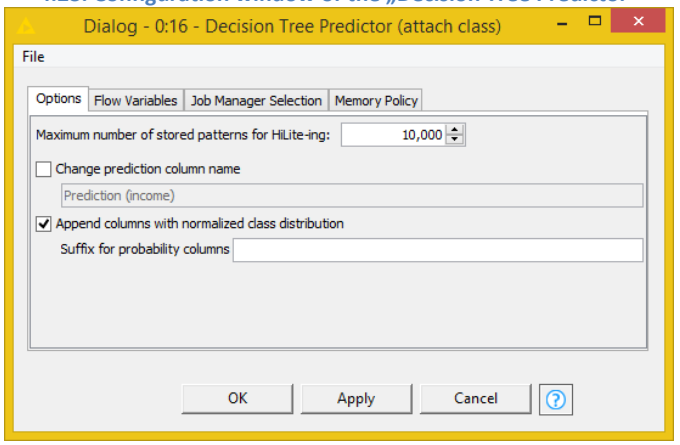
Decision Tree Predictor

The “Decision Tree Predictor” node imports a Decision Tree model from 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 (again a heritage of the old “Data Views” visualization nodes)
- 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“



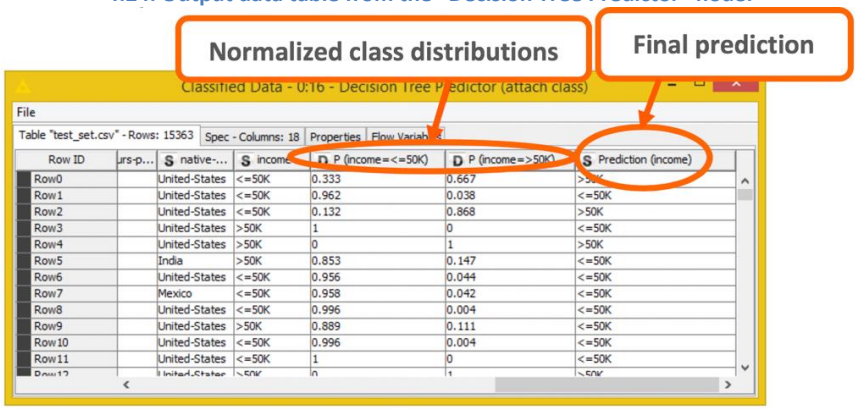
Decision Tree Views

In the context menu of both the “Decision Tree Predictor” node 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 ”

Sub-category “Mining” -> “Decision Tree” also includes a “Decision Tree To Image” node and a “Decision Tree to Ruleset” node. The “Decision Tree To Image” node converts the view of the decision tree model into an image. The “Decision Tree to Ruleset” node converts the decision tree splits into a set of rules.

4.24. Output data table from the “Decision Tree Predictor” node.



Let’s have a look at the decision tree views.

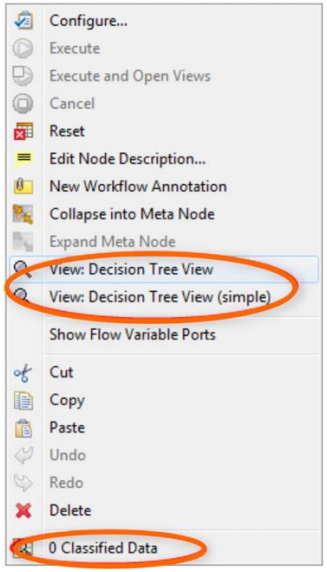
The more complex view (“View: Decision Tree View”) displays each branch of the decision tree as a rectangle. The data covered by this branch are shown inside the rectangle and the rule implementing the branch is displayed on top of the rectangle.

The simpler view (“View: Decision Tree View (simple)”) represents each branch as a circle fraction, where the fraction indicates how much of the underlying data is covered by the label assigned by the corresponding rule. The rule is displayed at the side of the branch.

In both views of the decision trees, a branch can show a little sign “+” indicating the possibility to expand the branch with more nodes.

The decision tree always starts with a “Root” branch that contains all training data. The “Root” branch in our decision tree is labeled “<=50K”, because it covers 11490 “<=50K” records out of 153362 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.

4.25. Context menu of the “Decision Tree Predictor” node



The first split happens in the “relationship” column. From the “Root” branch the data rows are separated into a number of sub-branches according to their value of the “relationship” attribute. Each branch is labeled with a predicted class coming from its cover factor. For example, the branch defined by the split condition “relationship = Wife, Husband” is labeled as “<=50K” since during the training phase it covered 3788 “<050K” records out of its

incoming 7074 training patterns. Fraction values in the 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 the hilite commands to work together with the "Data Views" nodes.

"**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. Nodes in the decision tree view can be colored using a "Color Manager" node. The final decision tree shown with color distributions for male (blue) and female (red) is reported in figure 4.26.

The "Scorer" node at the end measures the success performance for the decision tree as well, which amounts to 83% accuracy and 53% Cohen's kappa. The Naive Bayes classifier produced 81% accuracy and 54% Cohen's kappa. This means that the two model performances are comparable, even though the decision tree performs slightly better on one of the two classes, probably the more populated one.

Another possible visualization for a decision tree consists of the interactive view produced by the "Decision Tree View (Javascript)" node. This is another one of the Javascript based visualization nodes and it is dedicated to visualize the splits in a decision tree model (Fig. 4.29).

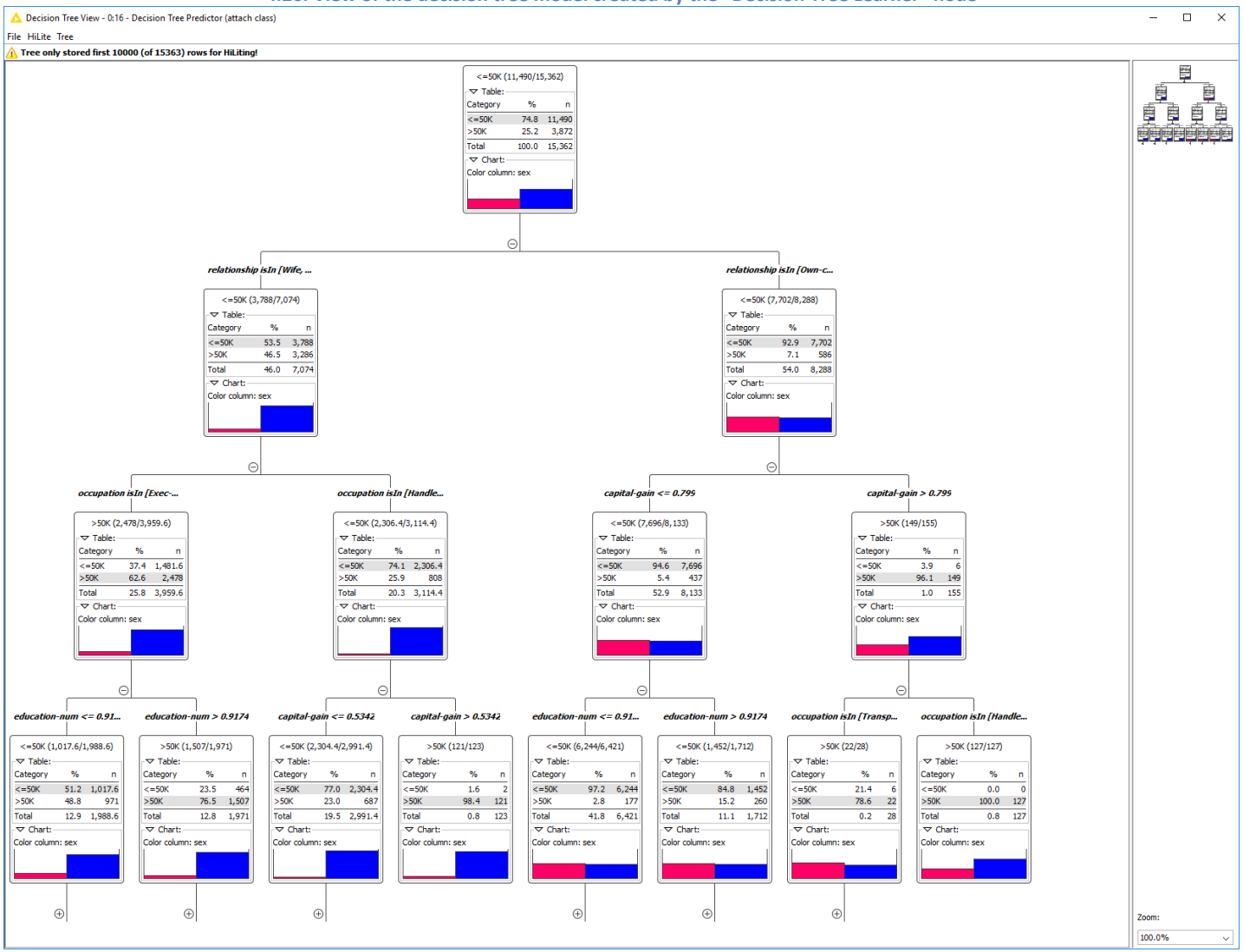
Another possible evaluation of the model performance could be achieved through an ROC curve. Actually, both models, the naïve Bayes and the decision tree, could be evaluated and compared by means of an ROC curve. Of course there is a Javascript based node that produces an interactive

visualization of a number of ROC curves. 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:

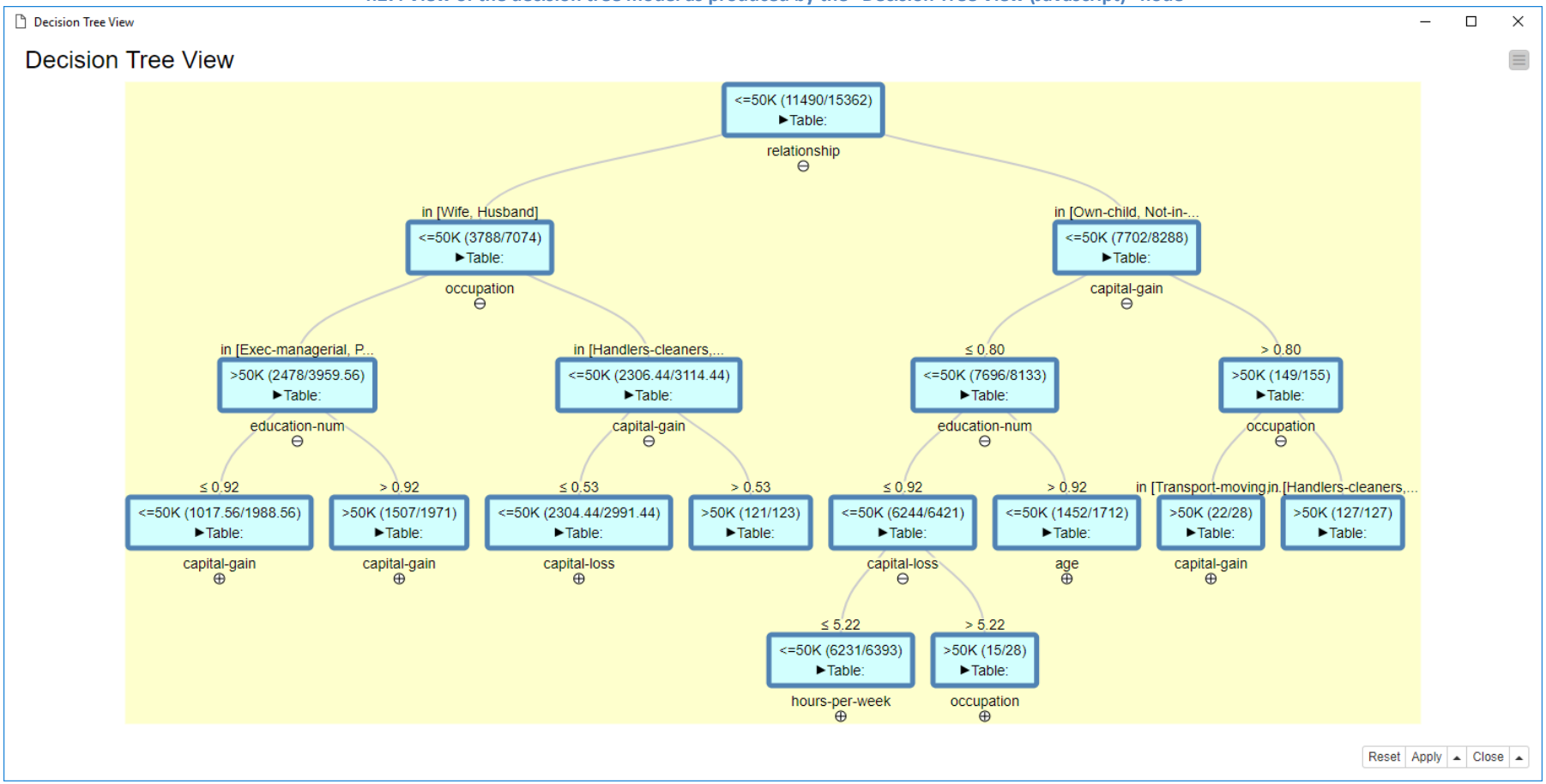
```
IF    Probability of positive class  >  threshold      =>   positive class
ELSE                                                  =>   negative class
```

The ROC Curve, in particular the area below the ROC curve, gives an indication of the predictor performance. 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. From figure 4.29 we can see that the two classification models have very similar performances (Area under the Curve = 89% for Naïve Bayes, Area under the Curve = 85% for the decision tree).

4.26. View of the decision tree model created by the “Decision Tree Learner” node



4.27. View of the decision tree model as produced by the “Decision Tree View (Javascript)” node



Decision Tree View (Javascript)

As for all Javascript based visualization nodes, the configuration window of this node contains three tabs: “Decision Tree Plot Options”, “General Plot Options”, and “View Controls”.

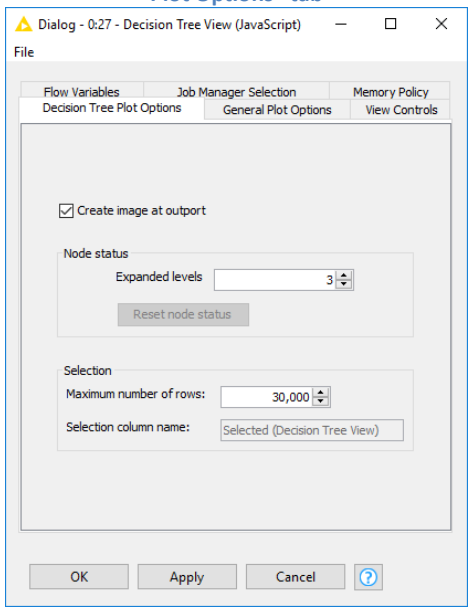
“Decision Tree Plot Options” tab defines the content to plot, such as the number of rows and the number of levels to be expanded already at view opening. Of course, the higher the number of rows to visualize, the slower the node execution. It also contains the flag to create an image from the produced view.

“Decision Tree Plot Options” tab defines general plot properties, such as background color, tree area background color, node color, title and subtitle, and formatting.

“View Controls” tab sets the plot interactivity, like zoom and title and subtitle editing.

Figure 4.27. shows a possible final view of the decision tree generated with a “Decision Tree View (Javascript)” node.

4.28. Configuration window of the „Decision Tree View (Javascript)“ node: “Decision Tree Plot Options” tab



ROC Curve (Javascript)

The “ROC Curve (Javascript)” node draws a number of ROC curves for a two-class classification problem. The configuration window covers four tabs: “ROC Curve Settings”, “General Plot Options”, “Axis Configuration”, and “View Controls”.

ROC Curve Settings

- The column containing the reference class
- The positive value of the class (arbitrarily assumed as positive)
- The column(s) with the probabilities for the positive class
- The limit on the number of points to plot. Remember less points less accurate curve, more points slower execution.

The selection of the columns with the probabilities for the positive class is performed by means of an “Exclude”/”Include” frame.

General Plot Options

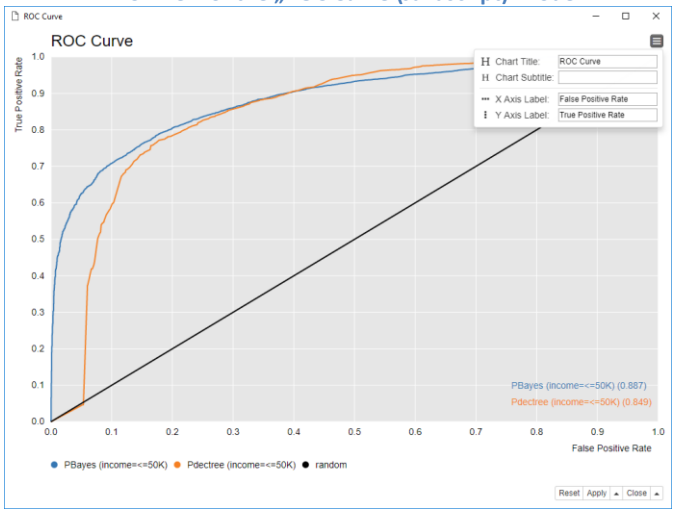
Here all plot settings are required: image size, formatting, background colors, etc …

Axis Control contains all settings about the plot axis

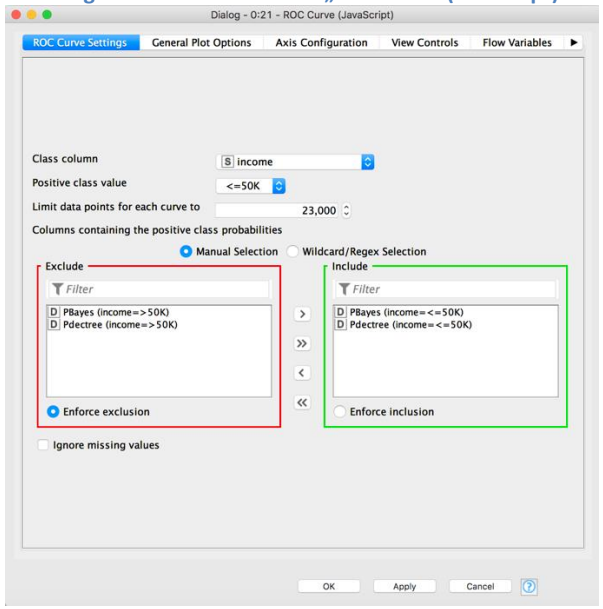
View Controls defines the level of interactivity of the curve view, such as label or title editing.

The node outputs the image (optionally) of the produced ROC curve and the Area under the Curve (AuC) for the probability columns.

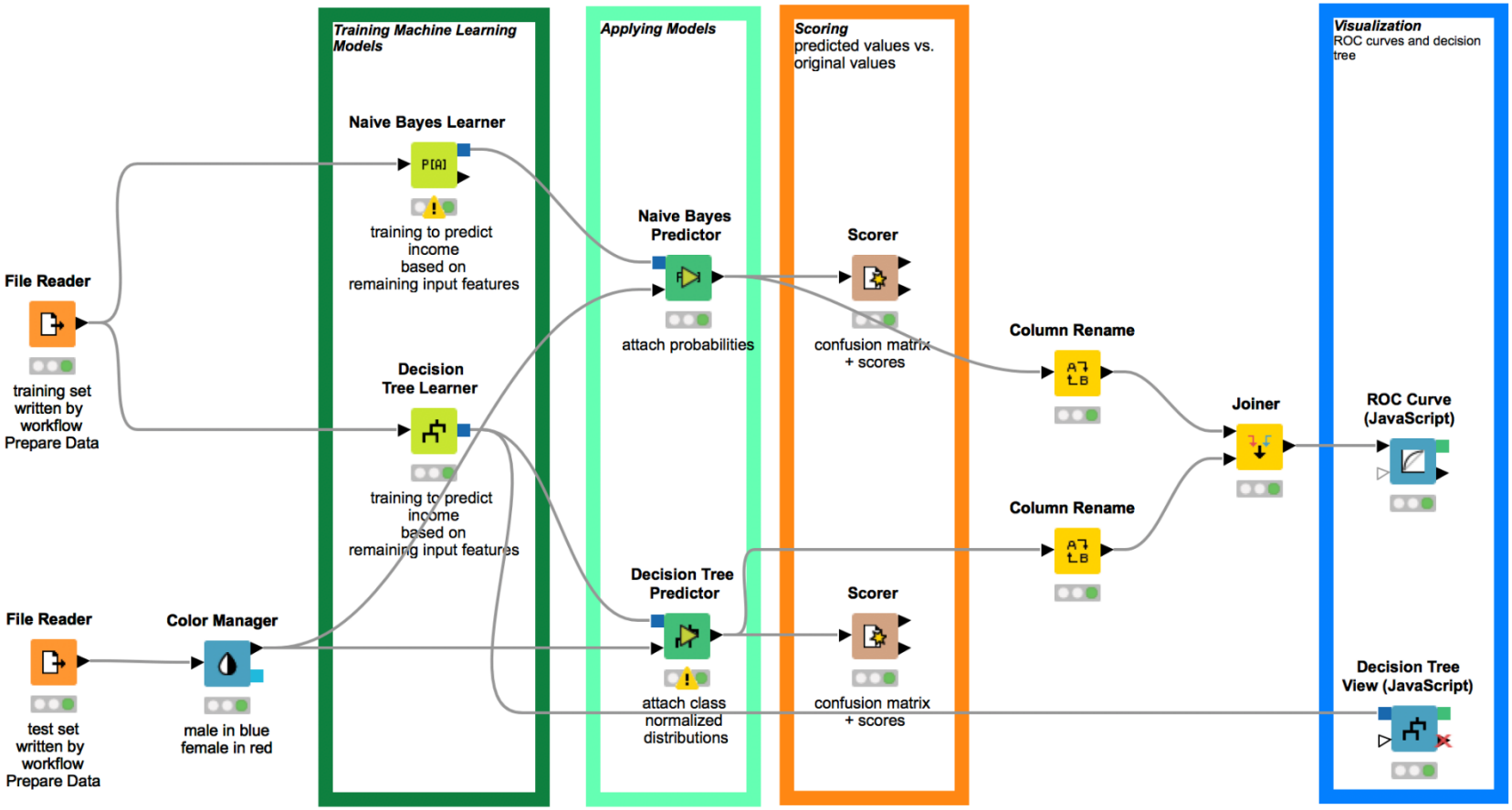
4.29. View of the „ROC Curve (Javascript)” node



4.30. Configuration window of the „ROC Curve (Javascript)“ node



4.31. Workflow "My First Data Model"



Artificial Neural Network

We move on now to a neural network and specifically to a Multilayer Perceptron (MLP) architecture, with one hidden layer, and the Back Propagation learning algorithm. The neural network paradigm is available in the "Mining" category and consists of:

- 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.

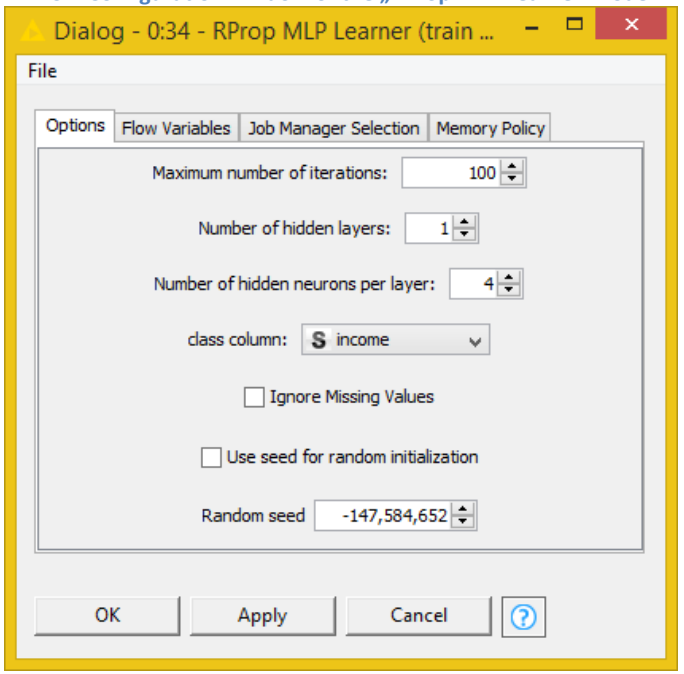
RProp MLP Learner

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* of the neural architecture
The *number of neurons per each hidden layer*
The *class column,* i.e. the column containing the target classes. The class column has to be of type String (nominal values)
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 corresponding checkbox in the configuration window.
Finally, you need to specify a seed to make the weight random initialization repeatable.

The "RProp MLP Learner" only accepts numerical inputs. String data columns will not be processed as input attributes.

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 set the classification/prediction task to predict the kind of income each person/record has. “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”, being of type String, could not be used as it was and it 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 Integer column containing the binary values for sex. We then used a “Column Filter” node to exclude all remaining string columns besides “Income”.

Multilayer Perceptron requires numerical data in the [0,1] range. In order to comply with that, a “Normalizer” node was placed after the “Column Filter” node to normalize all numerical data columns to fall into the range [0,1]. This sequence of transformations (“Rule Engine” on “sex”, “Column Filter” to keep only numerical attributes and the “Income” data column, and the [0,1] normalization”) was applied on both training and test set.

We then applied the “RProp MLP Learner” node to build an MLP neural network with 6 input variables (age, education-num, fnlwgt, capital-gain, capital-loss, hours-per-week, sex_01), 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 (blue square) with the model parameters produced by a “RProp MLP Learner” node

The predictor node has one output port, where the original data set plus the predicted classes and optionally the class distributions are produced.

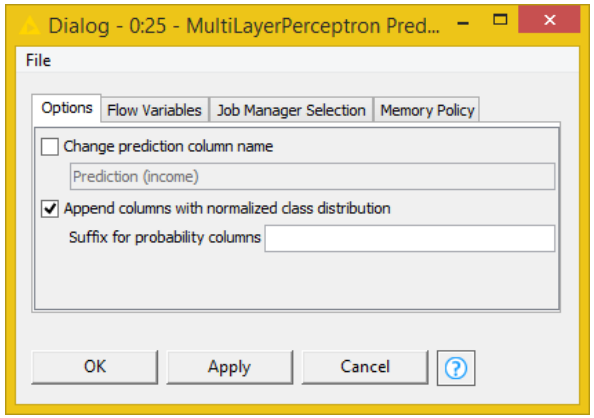
Multilayer Perceptron Predictor

The “Multilayer Perceptron Predictor” node takes an MLP model, generated by an “RProp MLP Learner” node, at the model input port (blue square) and applies it to the input data table at the input data port (white triangle).

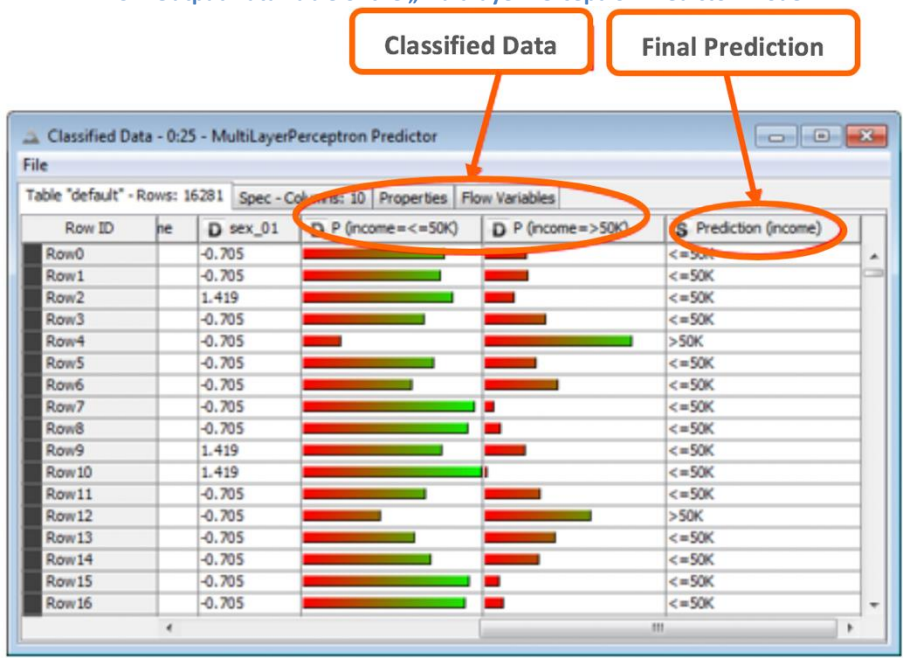
The “Multilayer Perceptron Predictor” node can be found in the “Node Repository” in the “Analytics” ->“Mining” -> “Neural Network” -> “MLP” category.

The only settings required for its configuration, like for all other predictor nodes, 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



4.34. Output Data Table of the „Multilayer Perceptron Predictor” node



Classified Data
Let’s visualize the results of the MLP Prediction:

- Right-click the “Multilayer Perceptron Predictor” node
- Select “Classified data”

The “Classified Data” data table contains the final predicted classes 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.
To change the rendering of the neuron firing values, you right-click the column header and select a new rendering under “Available Renderers”, like for example “Standard Double”.

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 (Javascript)" node to the results of the "Multilayer Perceptron Predictor" node. We identified:

The class column as column "Income"
The positive value "<=50K" in class column "Income"
The column "P(Income <=50K)" as the column containing the probability/score for the positive class

The resulting ROC Curve shows an Area under the Curve around 0.85.

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.

PMML Writer

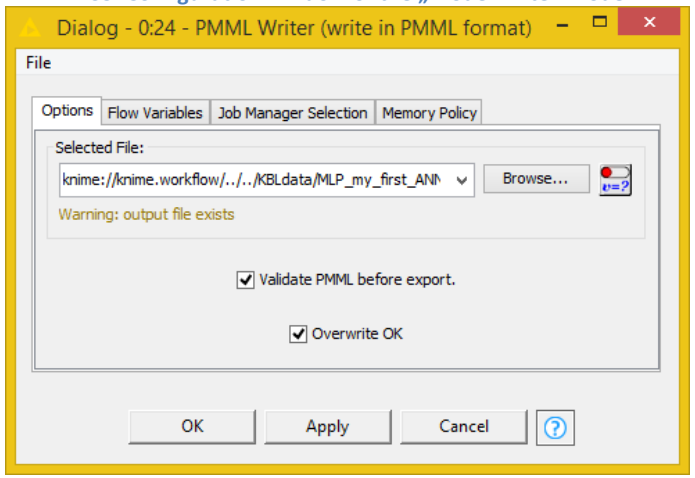
The "PMML Writer" node takes a PMML structured model at the input port (blue square) and writes it into a file by using the PMML format.

The "PMML 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 (*.pmml) (knime:// protocol is also accepted)
- The flag to override the file if the file exists
- The flag to validate the PMML structure

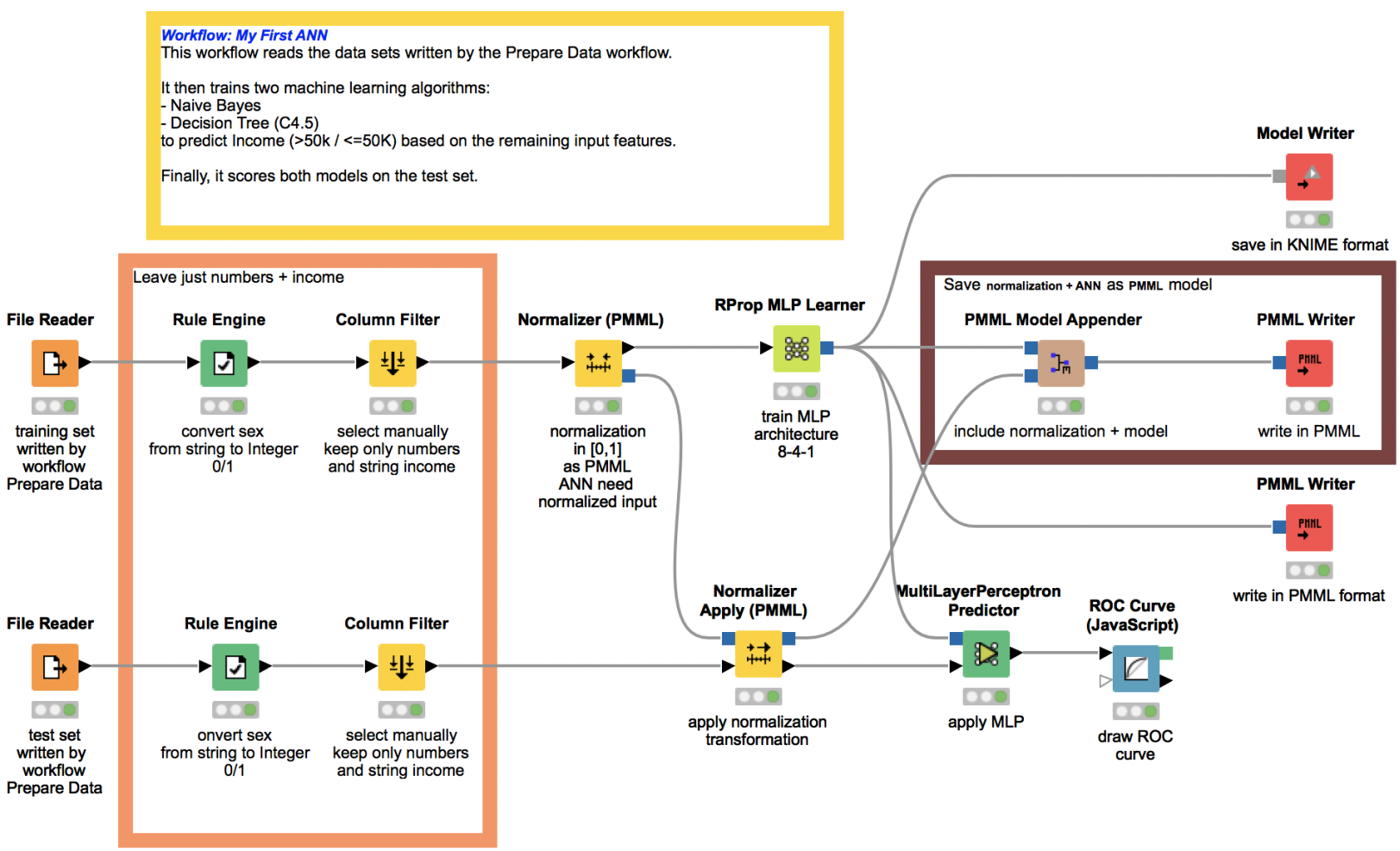
4.35. Configuration window of the „Mode Writer" node



Note. In the same "IO" -> "Write" category, you can find a similar node: the "Model Writer" node. The "Model Writer" node writes a model into a file with extension .model using a KNIME internal format (gray square).

The final workflow "My First ANN" is shown in below.

4.36. 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.

PMML Reader

The “PMML Reader” node reads a model from a file using the PMML standard format and makes it available at the output port (blue square).

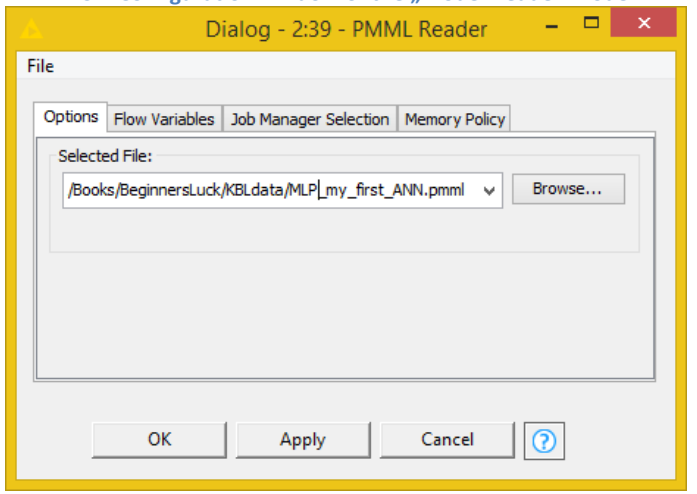
The node has an optional PMML input port to collect previous PMML data transformation structures.

The configuration window only needs:

- The path of the input file (*.pmml) (knime:// protocol is also accepted)

Drag and drop of a PMML file from a data folder automatically creates a “PMML Reader” node with the right configuration settings.

4.37. Configuration window of the „Model Reader“ node



Note. Similar to the “PMML Reader” node, the “Model Reader” node reads a model from a file with extension .model, but requires the model to be structured according to the KNIME internal format for models (gray square).

In this last part of the chapter, we would like to show a few more nodes that are commonly used in data analytics. We will build a new workflow, named “Clustering and Regression”, in the workflow group “Chapter4” to explain 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, as in the previous workflows.

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 “Analytics” -> “Statistics” category.

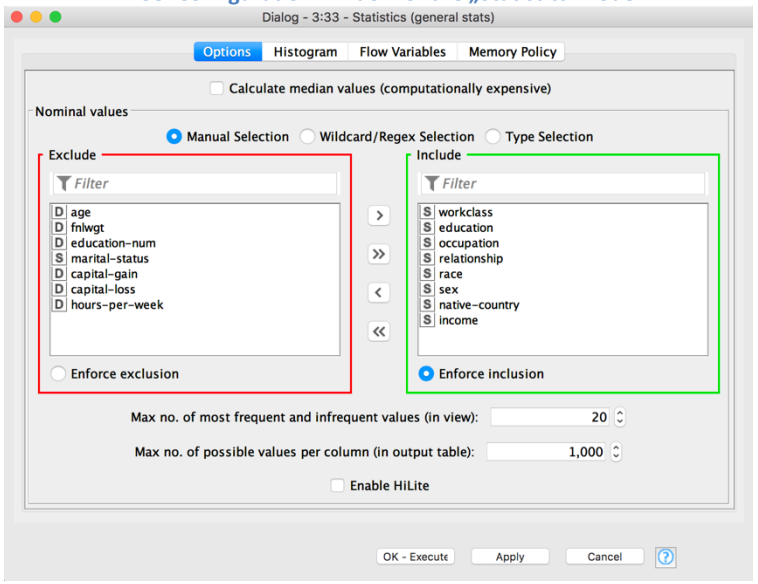
The configuration window requires:

- 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 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

All the statistical measures, described in the table above, are available at the node output ports as well as in the node View.

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.

4.38. 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.

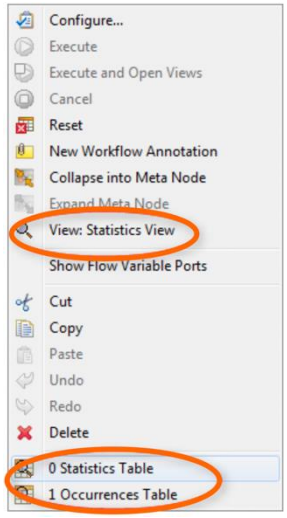
The node has three output ports and the “Statistics View” has three specular tabs. Output port “Statistics Table” corresponds to tab “Numeric” in the view; output port “nominal Statistical Values” corresponds to tab “Nominal” in the view; and output port “Occurrences Table” corresponds to tab “Top/Bottom” 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.

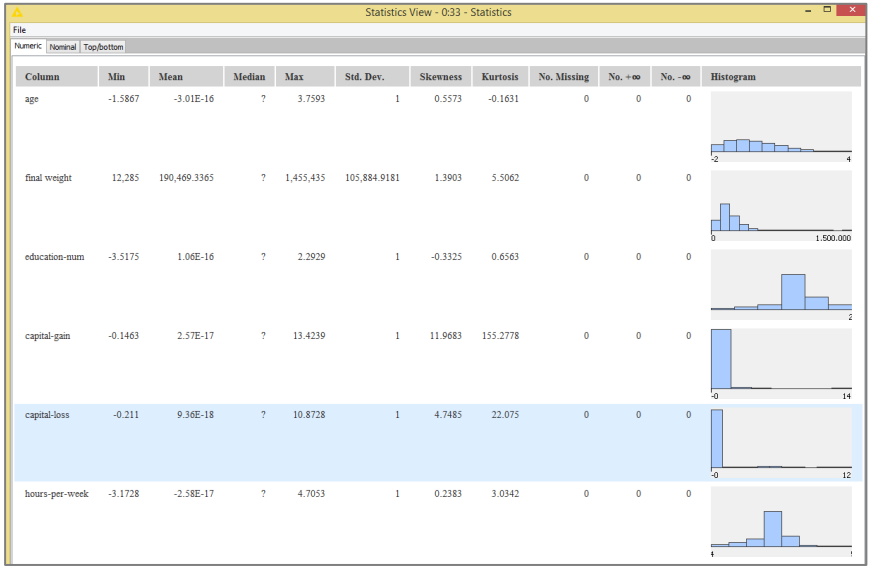
Note. The statistics of nominal columns is calculated only for the nominal columns included in the configuration window.

4.39. Context menu of the “Statistics” node

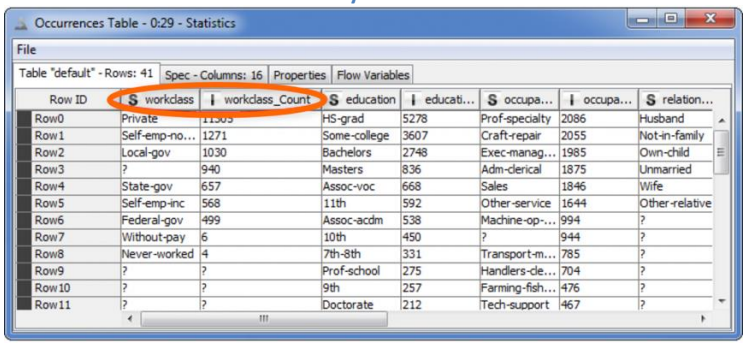


In our workflow’s node we excluded the column “marital-status” from the nominal columns and the corresponding column pair “marital-status” and “marital status_Count” is not in the “Occurrences Table”.
In “Statistics View” -> tab “Nominal Columns” we find the same information, but the lists of nominal 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.

4.40. The “Numeric” tab of the “Statistics” node view displays statistical measures and histogram calculated on all numerical columns.

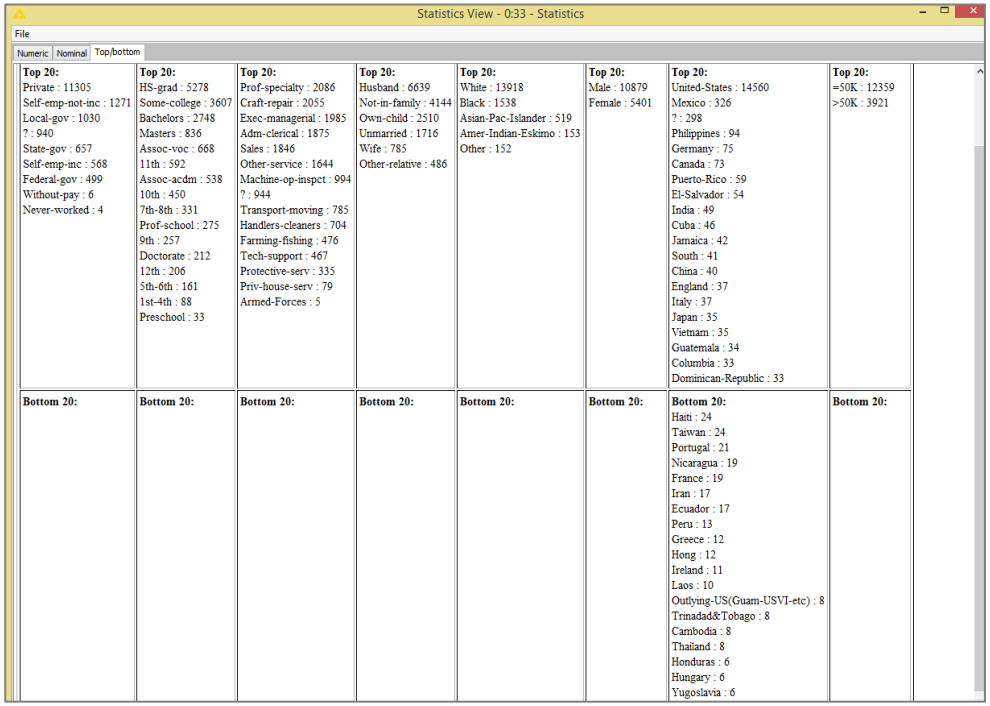


4.41. The “Occurrences Table” contains the number of occurrences of nominal values calculated only on the selected nominal columns



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	?

4.42. „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



Top 20:	Top 20:	Top 20:	Top 20:	Top 20:	Top 20:	Top 20:	Top 20:
Private : 11305 Self-emp-not-inc : 1271 Local-gov : 1030 ? : 940 State-gov : 657 Self-emp-inc : 568 Federal-gov : 499 Without-pay : 6 Never-worked : 4	HS-grad : 5278 Some-college : 3607 Bachelors : 2748 Masters : 836 Assoc-voc : 668 11th : 592 Assoc-acdm : 538 10th : 450 7th-8th : 331 Prof-school : 275 9th : 257 Doctorate : 212 12th : 206 5th-6th : 161 1st-4th : 88 Preschool : 33	Prof-specialty : 2086 Craft-repair : 2055 Exec-managerial : 1985 Adm-clerical : 1875 Sales : 1846 Other-service : 1644 Machine-op-inspct : 994 ? : 944 Transport-moving : 785 Handlers-cleaners : 704 Farming-fishing : 476 Tech-support : 467 Protective-serv : 335 Priv-house-serv : 79 Armed-Forces : 5	Husband : 6639 Not-in-family : 4144 Own-child : 2510 Unmarried : 1716 Wife : 785 Other-relative : 486	White : 13918 Black : 1538 Asian-Pac-Islander : 519 Amer-Indian-Eskimo : 153 Other : 152	Male : 10879 Female : 5401	United-States : 14560 Mexico : 326 ? : 298 Philippines : 94 Germany : 75 Canada : 73 Puerto-Rico : 59 El-Salvador : 54 India : 49 Cuba : 46 Jamaica : 42 South : 41 China : 40 England : 37 Italy : 37 Japan : 35 Vietnam : 35 Guatemala : 34 Columbia : 33 Dominican-Republic : 33	=50K : 12359 >50K : 3921
Bottom 20:	**Bottom 20:**	**Bottom 20:**	**Bottom 20:**	**Bottom 20:**	**Bottom 20:**	**Bottom 20:** Haiti : 24 Taiwan : 24 Portugal : 21 Nicaragua : 19 France : 19 Iran : 17 Ecuador : 17 Peru : 13 Greece : 12 Hong : 12 Ireland : 11 Laos : 10 Outlying-US(Guam-USVI-etc) : 8 Trinadad&Tobago : 8 Cambodia : 8 Thailand : 8 Honduras : 6 Hungary : 6 Yugoslavia : 6	**Bottom 20:**

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 “Analytics” -> “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.

The models we have seen so far were classifiers; that is they were trying to predict nominal values (classes) for each data row. The linear regression is a fitting model; that is a model that tries to predict numerical values. In this case, the target data column must be a numerical column with numerical values to be approximated through the linear regression fitting.

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 "Analytics" -> "Mining" -> "Regression".

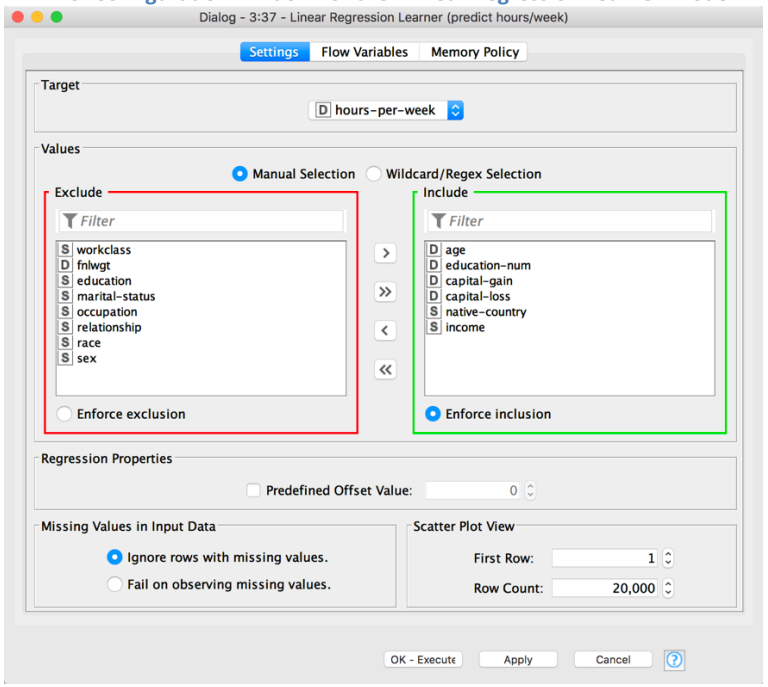
In the configuration window you need to specify:

- 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 (if any)

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.

4.43. Configuration window for the "Linear Regression Learner" node



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.

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-country", and "income" as independent variables for the linear 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.

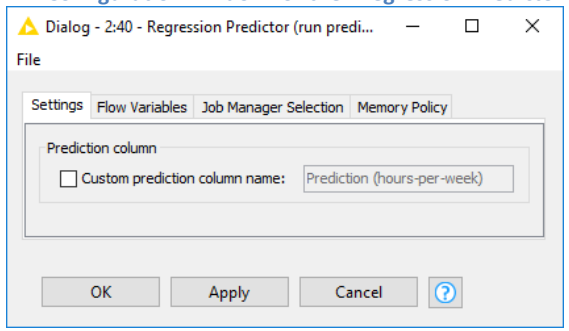
Regression Predictor

The "Regression Predictor" node obtains a regression model from one of its input ports (blue square) and data from the other input port (black triangle). It uses the model and the data to make a data based prediction.

Since all information is already available in the model, this node only needs the minimal predictor settings: an alternative customized name for the output classification column.

The "Regression Predictor" node is located in the "Analytics" -> "Mining" -> "Regression" category in the "Node Repository" panel.

4.44. Configuration window for the "Regression Predictor"



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 “Analytics” -> “Mining” -> “Clustering” category.

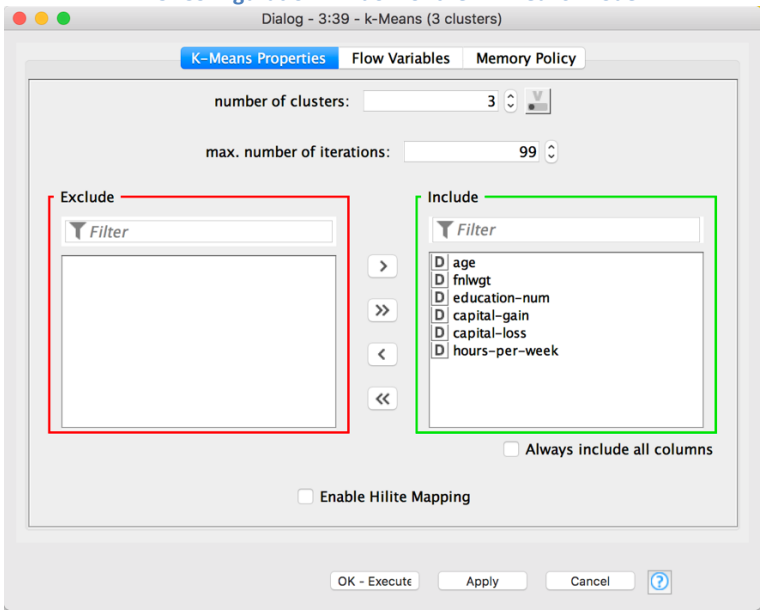
In the configuration window you need to specify:

- 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”.

4.45. 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 "Analytics" -> "Mining" -> "Clustering" category in the "Node Repository" panel.

It does not need any configuration settings specific to its cluster assignment task.

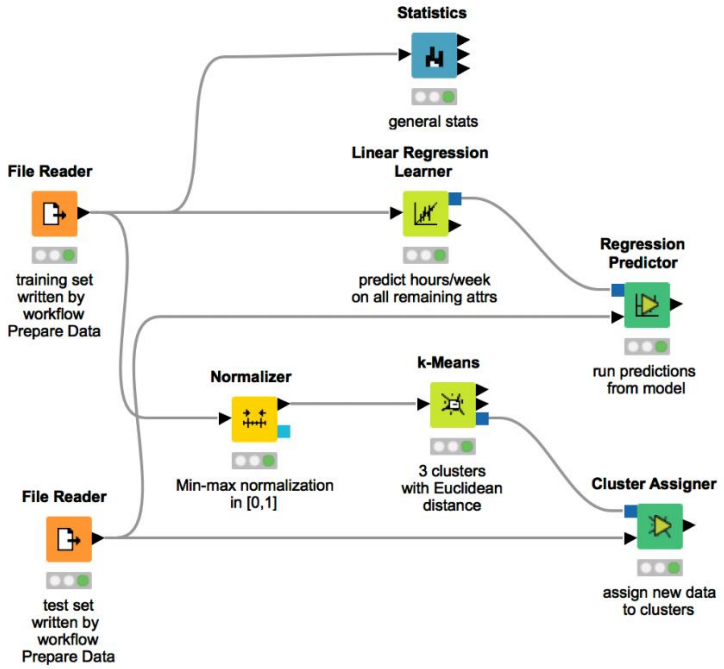
Note. The "Cluster Assigner" node is not specific for the "k-Means" node. It performs the cluster assignment task from a cluster set based with any of the available clustering algorithms.

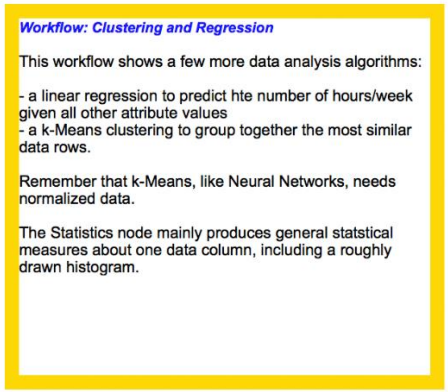
Hypothesis Testing

A few nodes are available in KNIME to perform classical statistical hypothesis testing. Most of them can be found in "Analytics" -> "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.

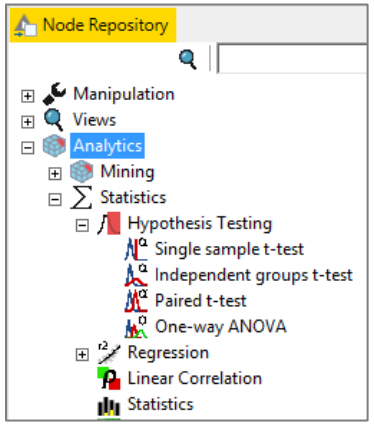
Additional newer nodes for statistical hypothesis testing are available under "KNIME Labs" -> "Statistics" in the "Node Repository" panel.

4.46. Workflow „Clustering and Regression“





4.47. 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.

Solution to Exercise 1

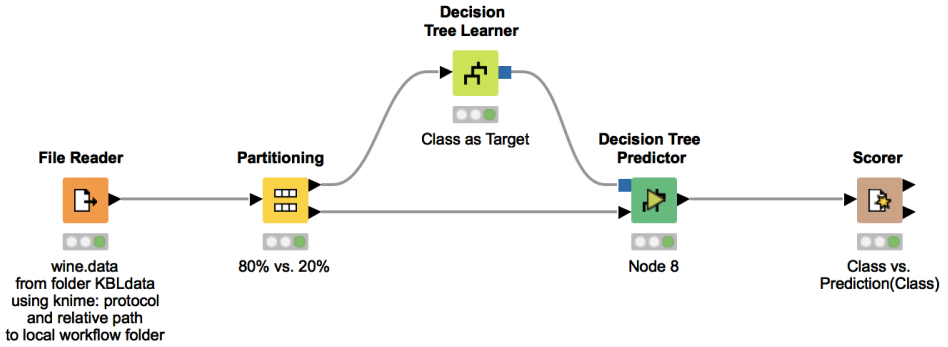
In the "Decision Tree Learner" node we used column "class" as the class column. By default, the "File Reader" node reads the wine data class as Integer, since the classes are "1", "2", and "3".

If you use a decision tree, as we did, for the final classification, you need the column "Class" to be of nominal values, i.e. to be of String type. You have two options for that:

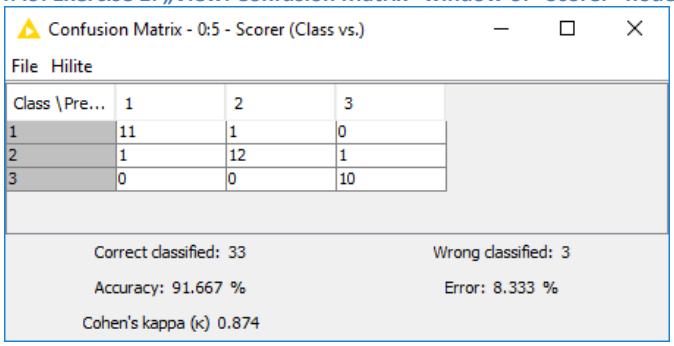
- You read "Class" as String. In the "File Reader" configuration window, right-click column "Class" and change type from "Integer" to "String"

- You leave the default settings in the "File Reader" and then you use a "Number To String" node for the conversion

4.48. Exercise 1: workflow



4.49. Exercise 1: „View: Confusion Matrix" window of "Scorer" node



We then used a "Scorer" node to see how many False Negatives were produced in the accuracy statistics and/or in the confusion matrix.

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.

Note. "Decision Tree Learner" node needs at least one nominal value to be used as classification column.

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 model performance.

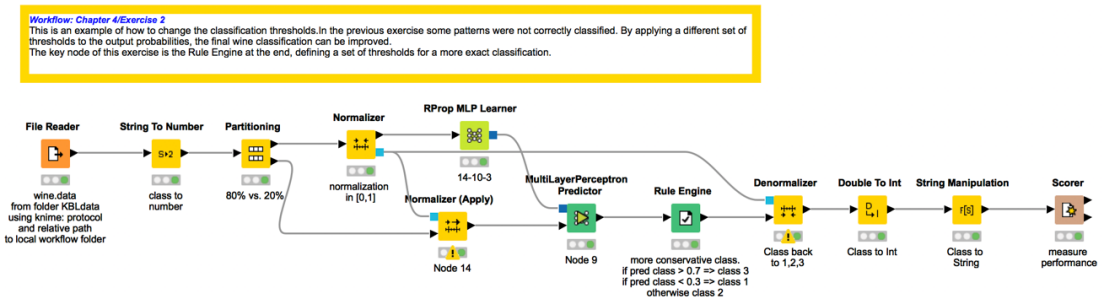
Solution to Exercise 2

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”.

4.50. 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

Model performance is measured with a “Scorer” node.

Alternatively, you can explore the “Numerical Scorer” node to measure performances with numerical distances.

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 Bayes model 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.

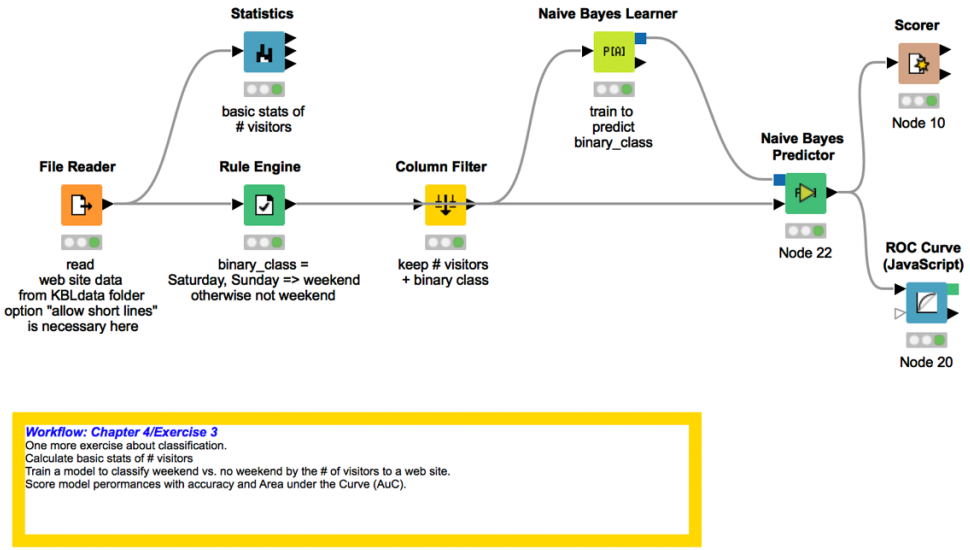
Solution to Exercise 3

We used a “Rule Engine” node to translate the column called “Day of Week” into a “weekend/not weekend” binary class.

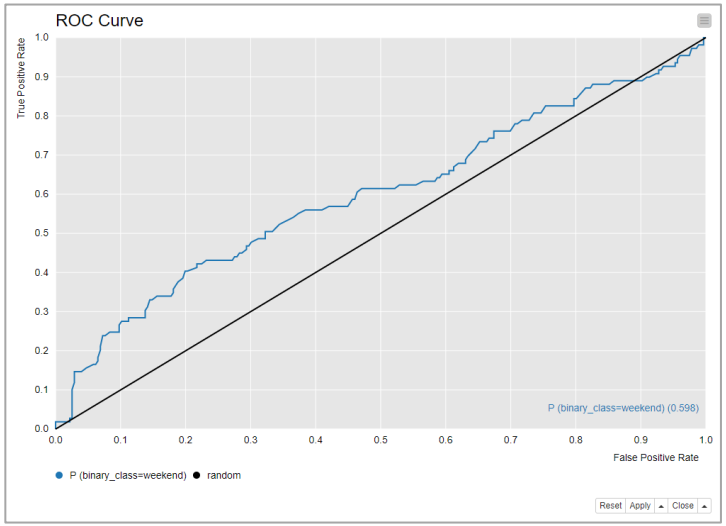
We filtered out the “Day of Week” column as not to make the classification task too easy for the Naïve Bayesian Classifier.

We trained the Bayesian Network on the binary class “weekend/not weekend” and we built the ROC curve on the “weekend” class probability.

4.51. Exercise 3: workflow



4.52. 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, http://www.eclipse.org/birt/), which is an Eclipse based 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

File Table - 2:1 - File Reader(Projects file)

File

Table "Projects.txt" - Rows: 132 | Spec - Columns: 7 | Properties | Flow Variables

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 data folder "KBLData" from the "Download Zone" file. The file "Projects.txt" 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.

5.2. Installing the Report Designer Extension

The “KNIME Report Designer” suite is not included in the basic standalone version of KNIME. It can be downloaded as a separate extension package from the “KNIME & Extensions” link in the “Help” -> “Install New Software” window.

To install the “KNIME Report” package:

- Start KNIME
- In the Top Menu click “Help” -> “Install New Software …” OR “File” -> “Install KNIME Extensions…”
- In the “Available Software” window in the text box “Work with”, select “KNIME Update Site – http://www.knime.org/update/3.x”
- Expand “KNIME & Extensions”
- Select the package “KNIME Report Designer”
- Click the button “Next” on the bottom and follow the installation instructions

If the installation runs correctly, after KNIME is restarted, you should have a new category “Reporting” in the “Node Repository” panel with two nodes: “Data To Report” and “Image To Report”

5.3. 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. In the report, we want to show 3 tables, with structure as the one shown in figure 5.2, and 2 charts, from a data table as the one reported in figure 5.3.

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.

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.

The third table has the same structure as the two tables described above and shows the remaining amount of money (= money assigned - money used) for each project for each year.

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.

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.

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
…	…	…

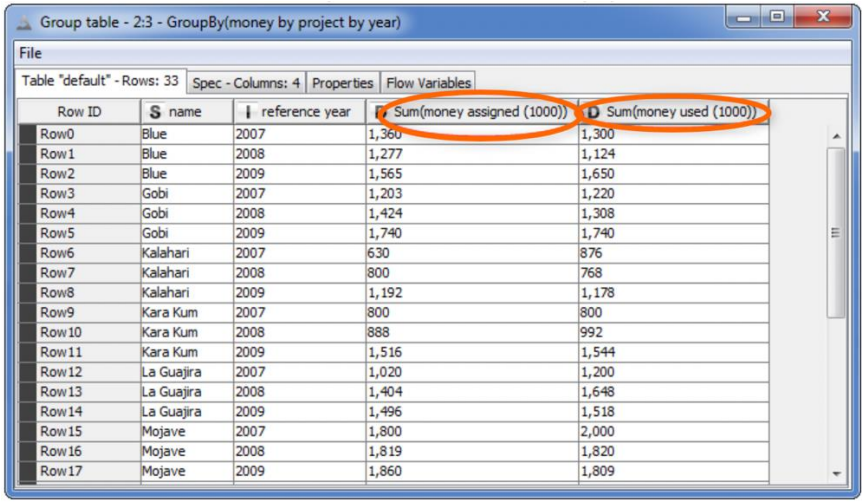
For both table and chart structures, we need to calculate the sum of money (assigned, used, or remaining), but we need to report it on a different data structure. For example, in one case we want the years to be the column headers and in the other case we want the years to be the column values.

The first data table (Fig. 5.2) could be obtained with a “Pivoting” node, while the second data table (Fig. 5.3) with a “GroupBy” node. We then introduced a “GroupBy” node and two “Pivoting” nodes in the “Projects” workflow.

In the "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).

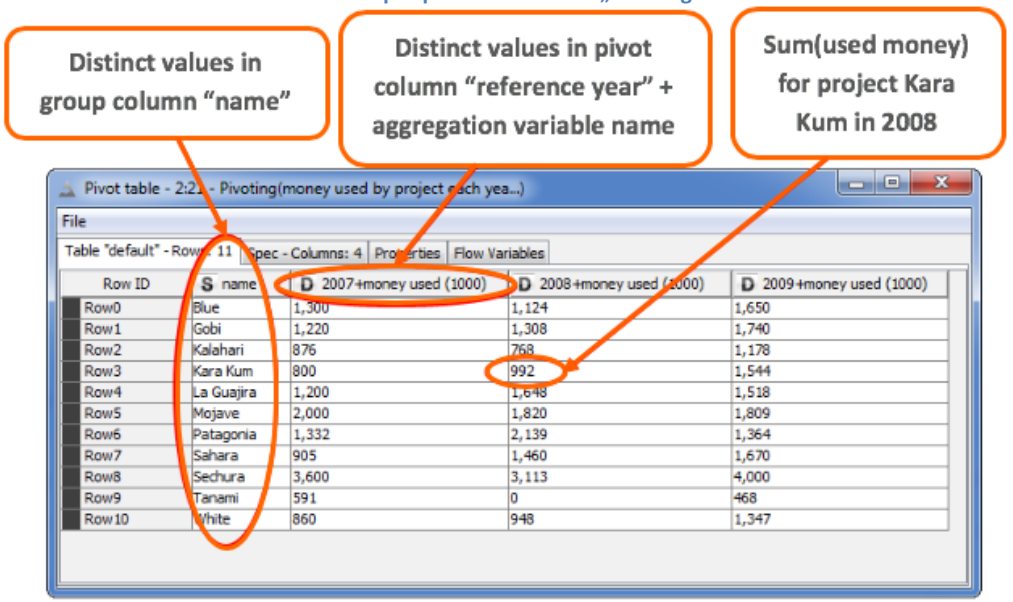
In the resulting data table, the first two columns contained all combinations of distinct values in the "*name*" and "*reference year*" columns. The aggregations were then run over the groups of data rows defined by each ("*name*", "*reference year*") pair. The aggregated values are displayed in two new columns "*Sum(money assigned (1000))*" and "*Sum(money used (1000))*".
We named the new "GroupBy" node *"money by project by year".*

5.4. Output data table of the GroupBy node



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

5.5. Output pivot table of the „Pivoting" node



Row ID	name	2007+money used (1000)	2008+money used (1000)	2009+money used (1000)
Row0	Blue	1,300	1,124	1,650
Row1	Gobi	1,220	1,308	1,740
Row2	Kalahari	876	768	1,178
Row3	Kara Kum	800	992	1,544
Row4	La Guajira	1,200	1,648	1,518
Row5	Mojave	2,000	1,820	1,809
Row6	Patagonia	1,332	2,139	1,364
Row7	Sahara	905	1,460	1,670
Row8	Sechura	3,600	3,113	4,000
Row9	Tanami	591	0	468
Row10	White	860	948	1,347

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.

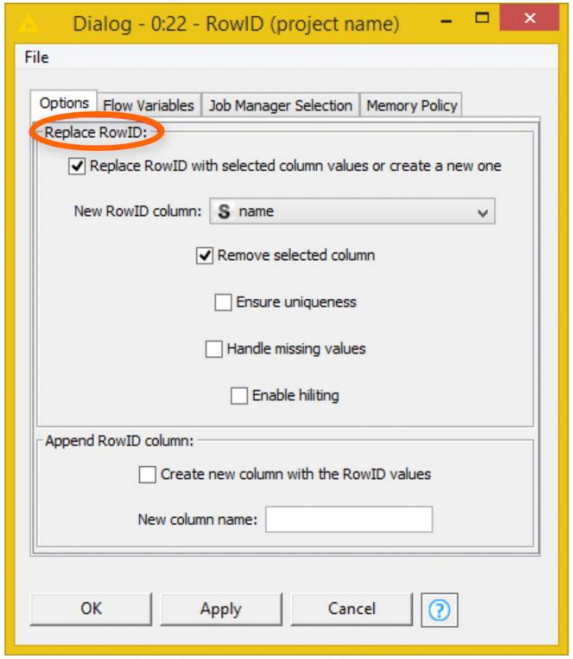
The RowID node allows the user to:

- 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.

5.6. Configuration window for the „RowID" node



In the "Node Repository" panel, close to the "Pivoting" node you can find a "Unpivoting" node.

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 5.8 and 5.9. 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.

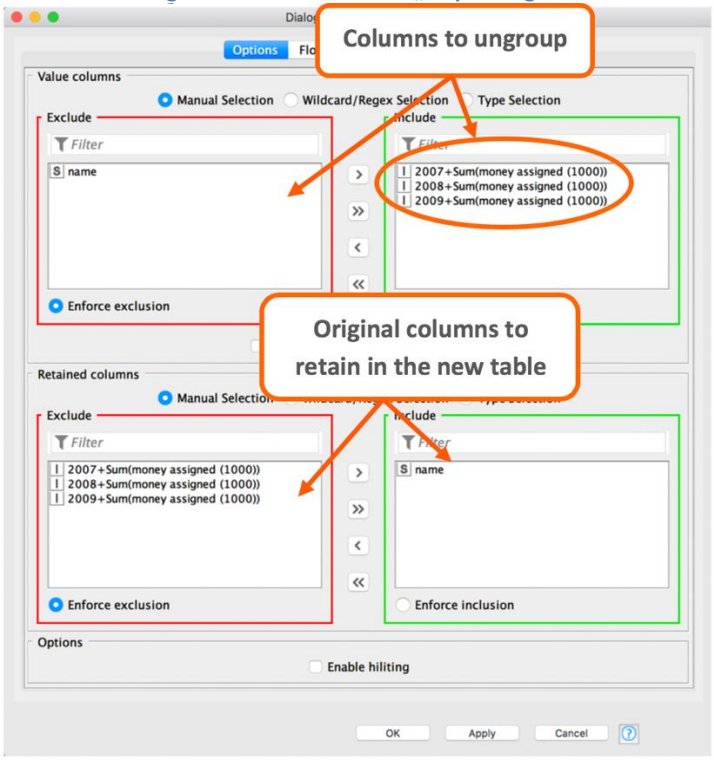
In the settings you need to define:

- 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

5.7. 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.

5.8. Input Data Table

	Col1	Col2	Col3
ID 1	1	3	5
ID 2	2	4	6

5.9. 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. **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:

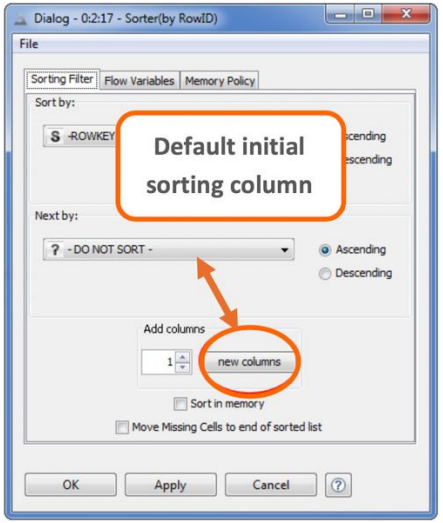
- The column(s) to be sorted (the RowIDs column is also accepted)
- Whether to sort in ascending or descending order

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.

5.10. Configuration window for the „Sorter“ node



5.4. 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 below.

5.11. 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

5.12. 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”`

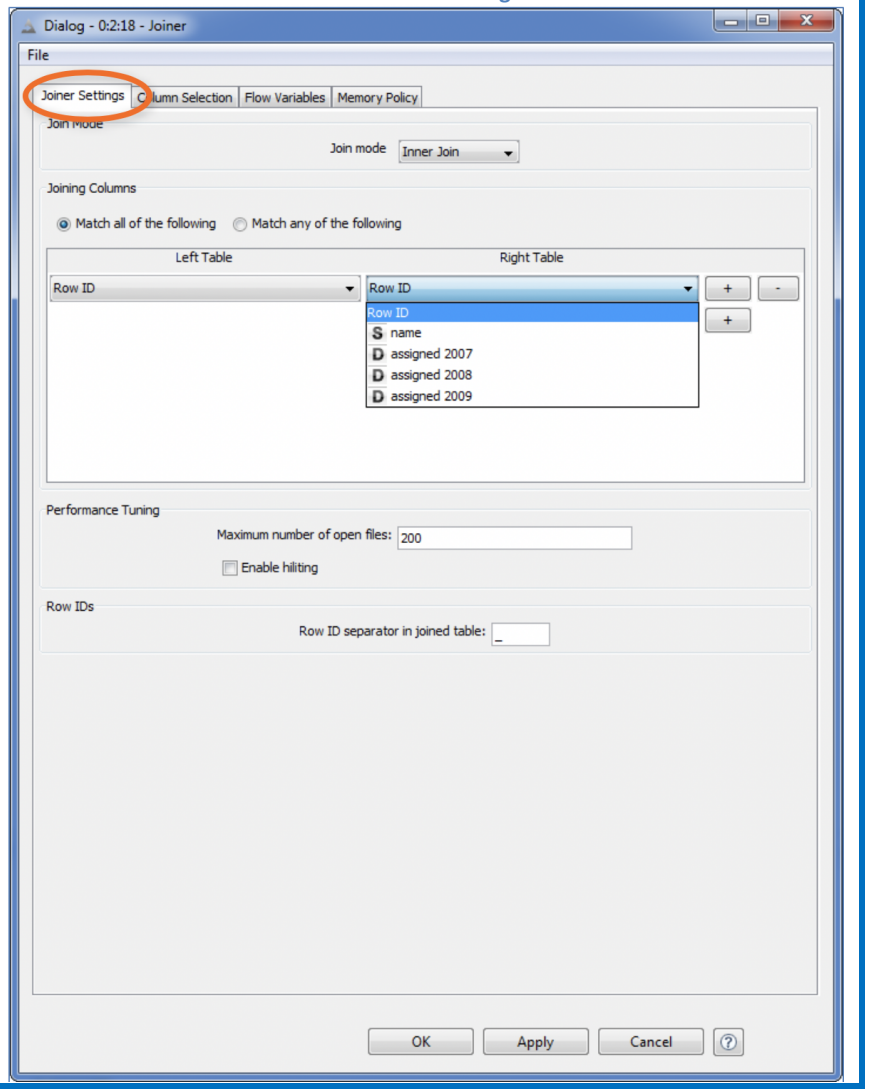
The “Joiner” node takes two data tables on the input ports and matches a column of the table on the upper port (left table) with a column of the table on the bottom port (right table). These columns can also be the RowID columns.

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

5.13. 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.

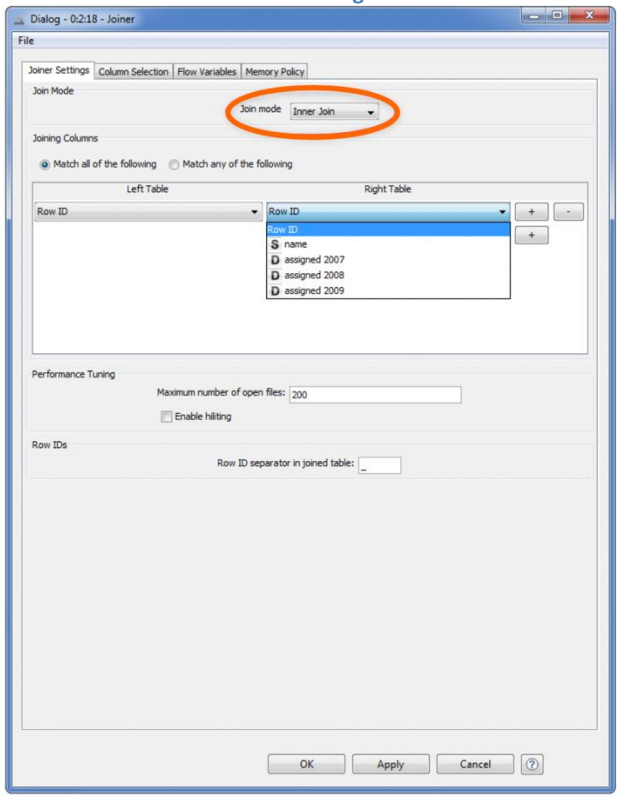
The first setting is the **join mode** (Fig. 5.21).

- **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.

5.14. 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".

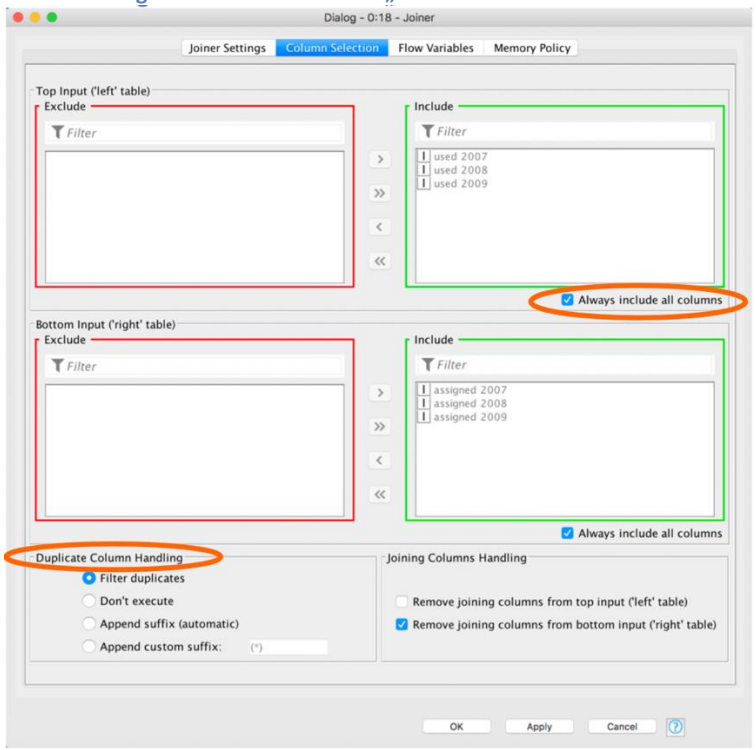
Under each "Column Selection" frame there is a flag called "**Always include all columns"**. If this flag is enabled, the "Column Selection" frame is disabled and all columns from this table are retained in the joined output data table.

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.

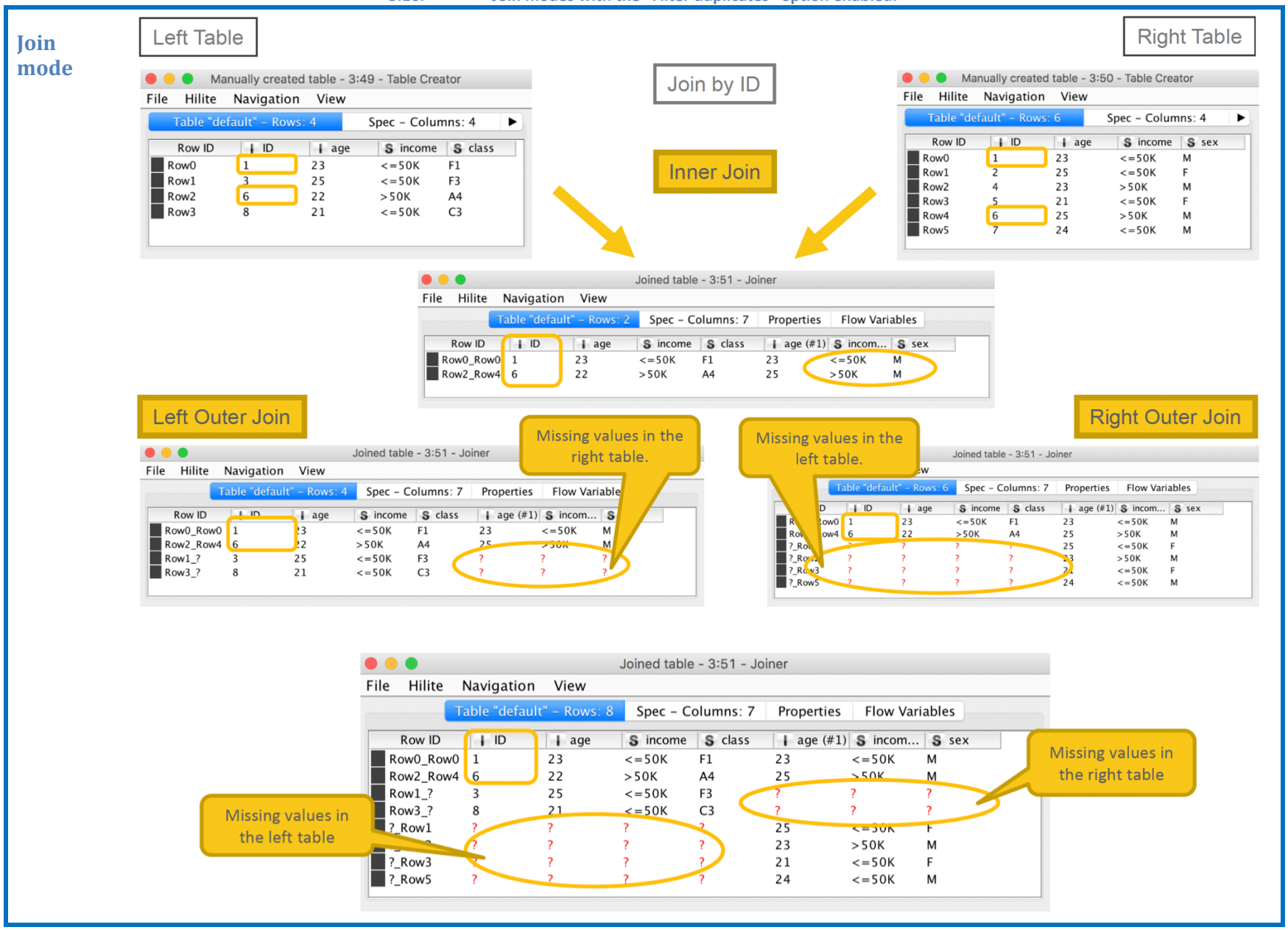
- "**Filter duplicates**" filters out the duplicate columns of the right table and keeps those of the left table
- "**Do not execute**" produces an error
- "**Append suffix**" appends a suffix, default or customized, to the name of the duplicate columns in the right table

"Joining Columns Handling" defines whether the joining columns of the left and/or of the right table or none are filtered out.

5.15. Configuration window for the „Joiner" node: "Column Selection" tab



5.16. Join modes with the “Filter duplicates” option enabled.

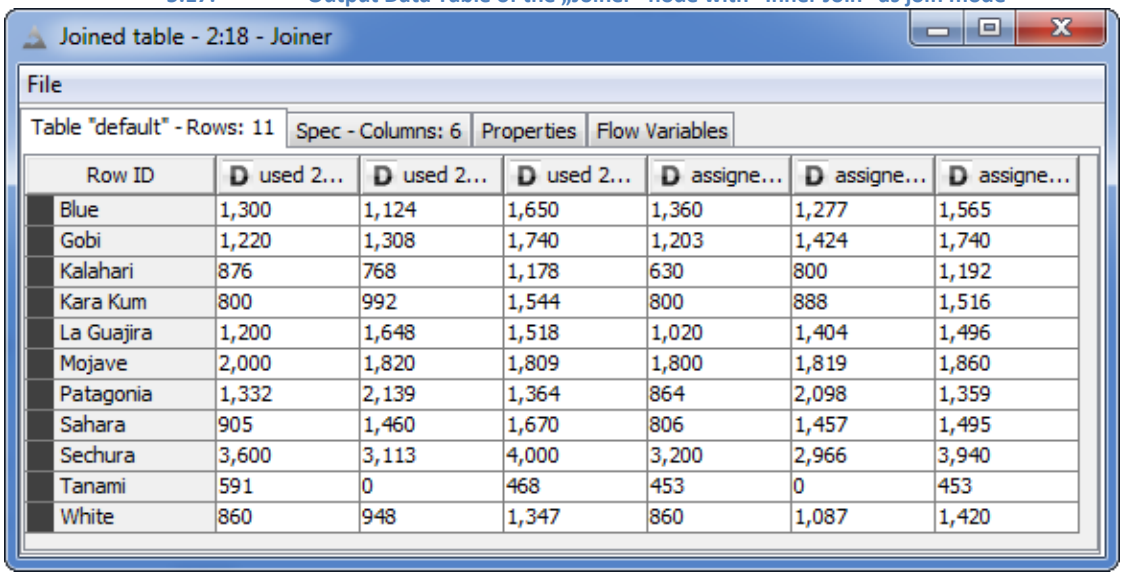


We joined the two tables (money assigned and money used) using the RowIDs as the joining column for both; we chose to filter out the columns from the left table with the same name as the columns from the right table; and we chose the inner join as join mode.

The resulting data is shown in figure 5.17. 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.

5.17. Output Data Table of the „Joiner" node with "Inner Join" as join mode



Row ID	used 2...	used 2...	used 2...	assigne...	assigne...	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

5.5. 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)

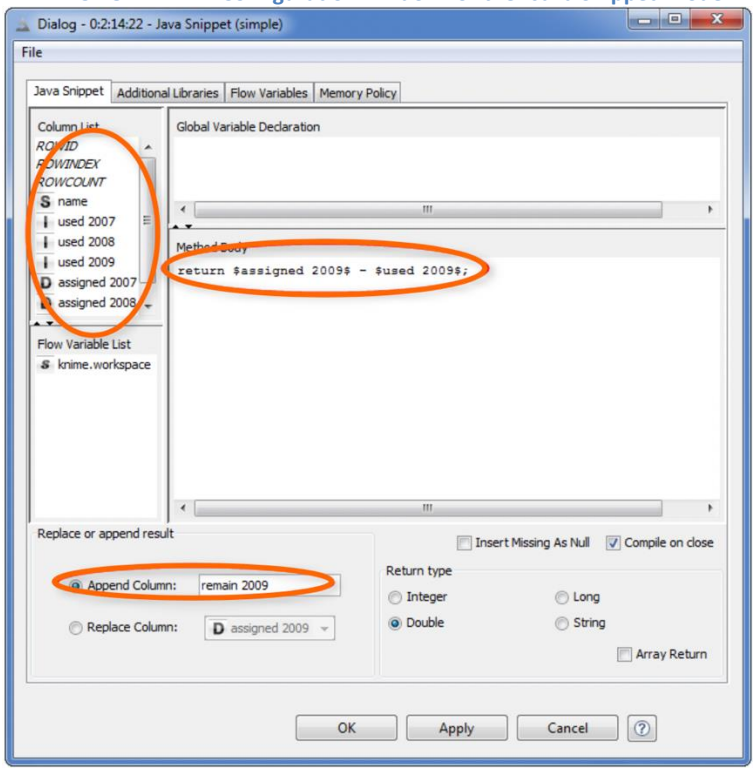
A “Java Snippet (simple)” node allows the execution of a piece of Java code and places the result value into a new or existing data column. The code has to end with the keyword “return” followed by the name of a variable or expression.

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.

5.18. 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.

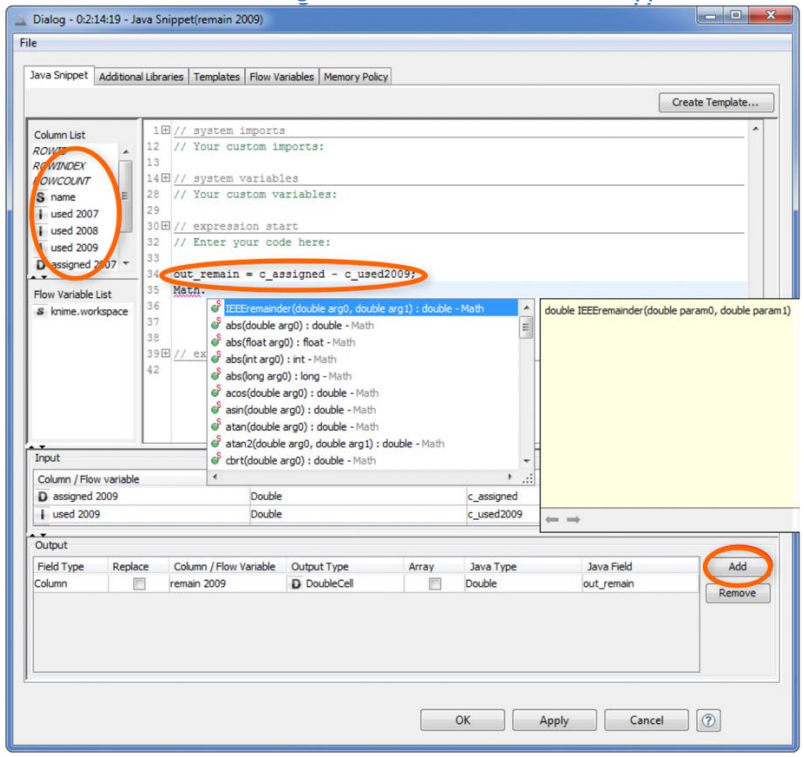
Java Snippet

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:

- 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.

5.19. Configuration window for the “Java Snippet” node



- 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

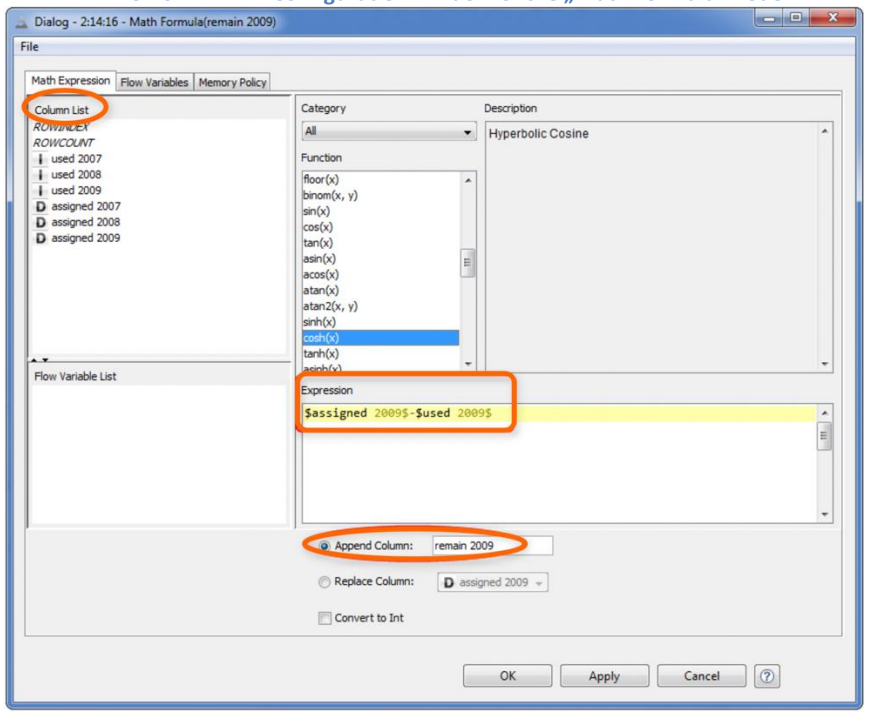
The “Math Formula” node enables mathematical formulas to be implemented and works similarly to the “String Manipulation” node (see section 3.5).
The “Math Formula” node is not part of the basic standalone KNIME. It has to be downloaded with the extension package (see par. 1.5) *“KNIME Math Expression 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**

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.

5.20. Configuration window of the „Math Formula“ node



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 of the Math Formula node introduced in the “Projects” workflow, we 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.

Another type of "Math Formula" node is the "Math Formula (Multi Column)" node.

Math Formula (Multi Column)

The "Math Formula (Multi Column)" node allows to implement the same mathematical formula on a list of columns, as specified in the configuration window.

In the upper part of the configuration window, we choose the list of columns on which to implement the formula, through an Exclude/Include frame.

After that, we have the same configuration as for the simpler "Math Formula" node.

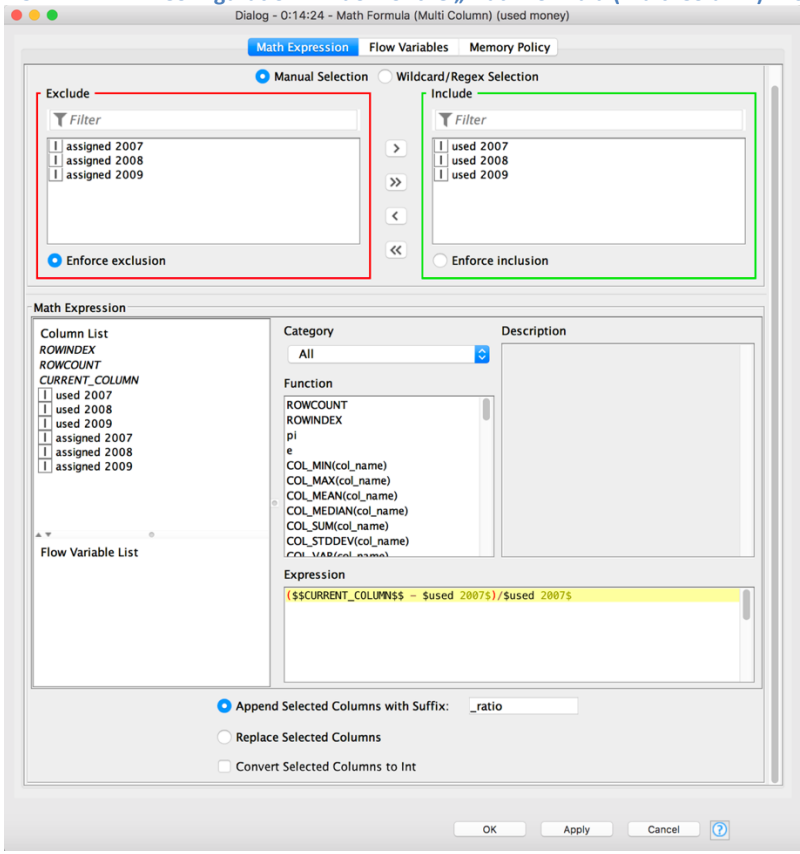
- The **list of column names** from the input data table
- The list of variables (to see in the "KNIME Cookbook")
- A list of **mathematical functions,** e.g. log(x) and their descriptions
- The **mathematics expression editor**

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.

The last three options include:

- A suffix to add to the original column names to form the output column names, if we want to create new columns;
- The option of overwriting the values in the original columns
- The flag to convert all results into Integer numbers

5.21. Configuration window of the „Math Formula (Multi Column)" node



5.6. 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.

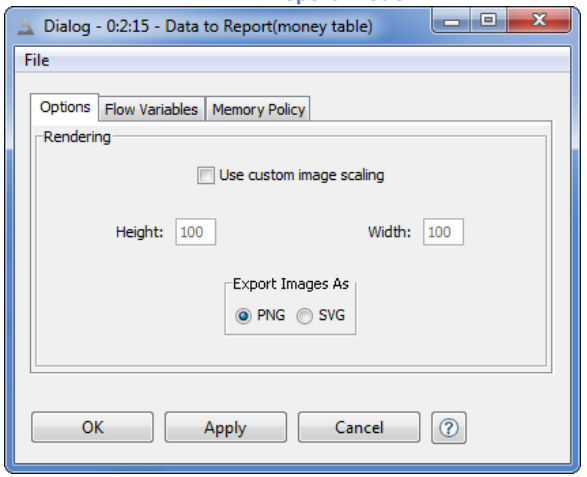
Data to Report

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.

5.22. 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.

5.7. 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.

5.23. "Meta-node" icon in the Tool Bar

To open a “Meta-node”:

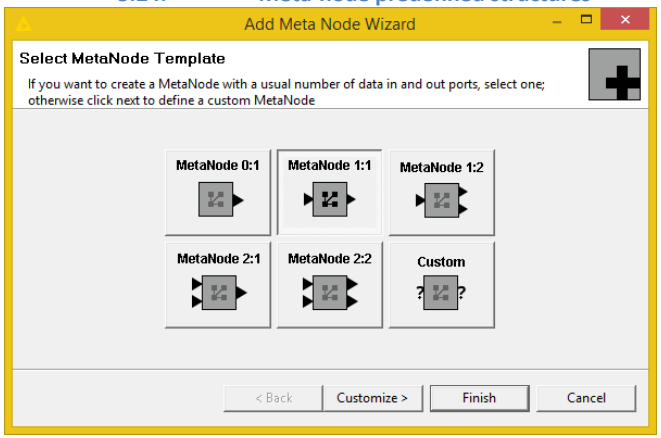
- Double-click the “Meta-node”

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”.

5.24. Meta-node predefined structures



To fill a “Meta-node” with nodes:

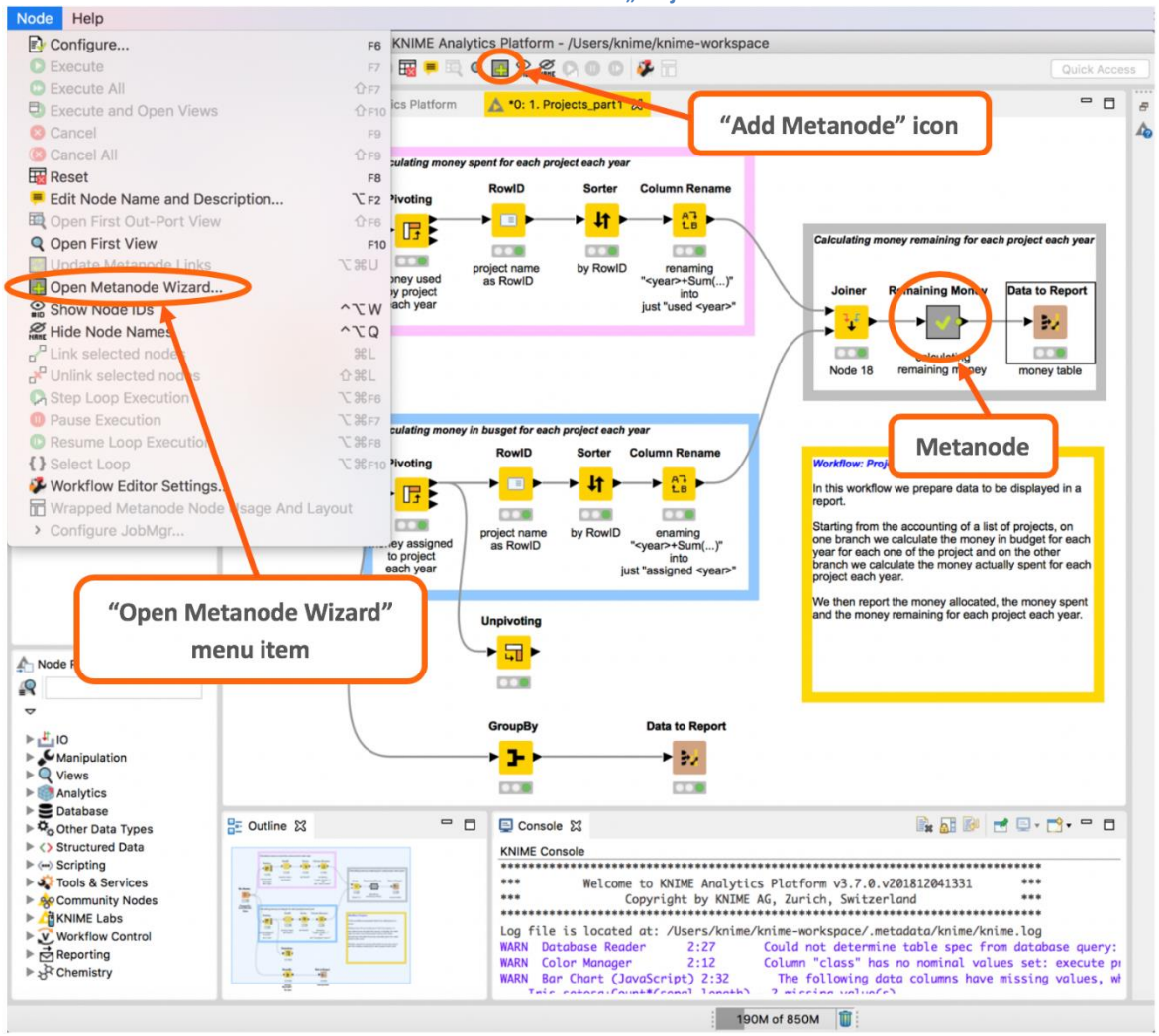
- Drag and drop nodes from the “Node Repository” panel as you would do with a normal workflow

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.

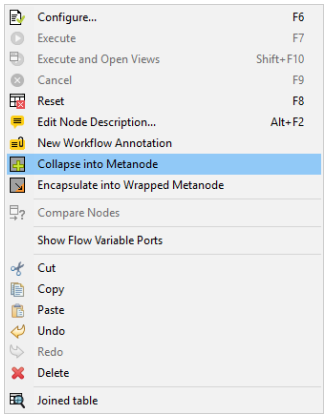
5.25. Workflow „Projects“



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

5.26. "Collapse into Meta Node" option in the context menu of selected nodes

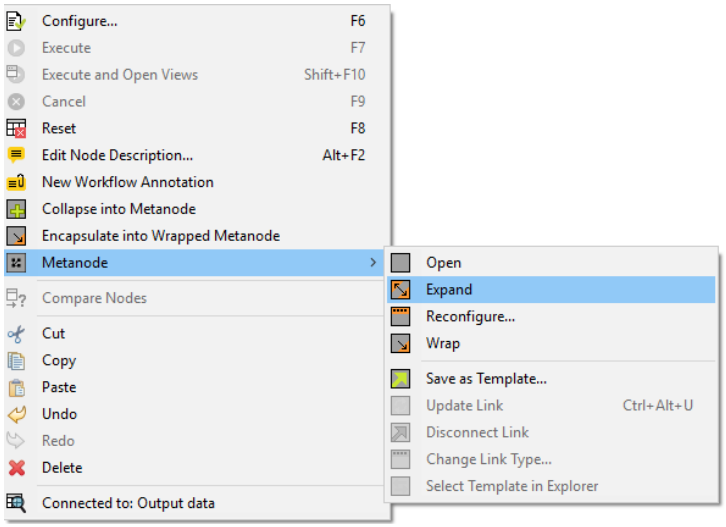


Expand and Reconfigure a Meta-node

Once the meta-node exists, it is possible to interact with it (reconfigure, expand, etc…) through its context menu.
The context menu of a meta-node is completely similar to the context menu of any other node, besides the “Meta Node” option. The “Meta Node” option opens a sub-menu with commands applicable only to a meta-node, such as:

- “Open”, to open the meta-node content in the workflow editor
- “Expand”, to reintroduce the meta-node content into the main workflow and get rid of the meta-node container
- “Reconfigure”, to change the meta-node in terms of input/output ports and name
- “Wrap”, to transform the meta-node into a sub-node with configuration window options
- “Save as Template”, to save the current meta-node as a template into a central repository and allow other users to use it with the right permissions (this option is only available with a commercial license)

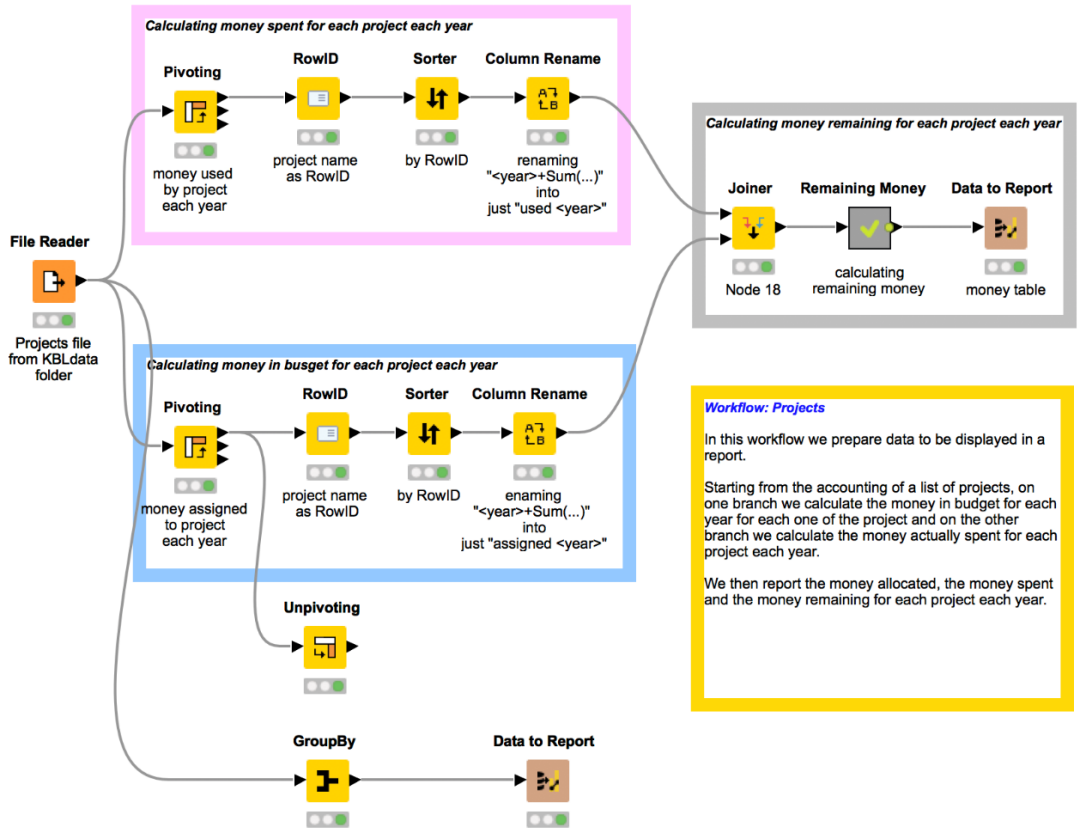
5.27. Meta-node context menu



You might have noticed the presence of items involving “Wrapped Meta-nodes” in the context menu. A wrapped meta-node is a special kind of self-contained meta-node.

Note. The main difference between meta-nodes and wrapped meta-nodes involves the usage of flow variables and the execution of Javascript based nodes on the KNIME WebPortal.

5.28. The "Projects" workflow



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 some sub-categories "Meta Nodes" under some main categories, like "Workflow Control", "Mining", "R", "Time Series", and others. The "Meta Nodes" categories contain useful pre-packaged meta-node implementations for that particular category.

5.8. 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:

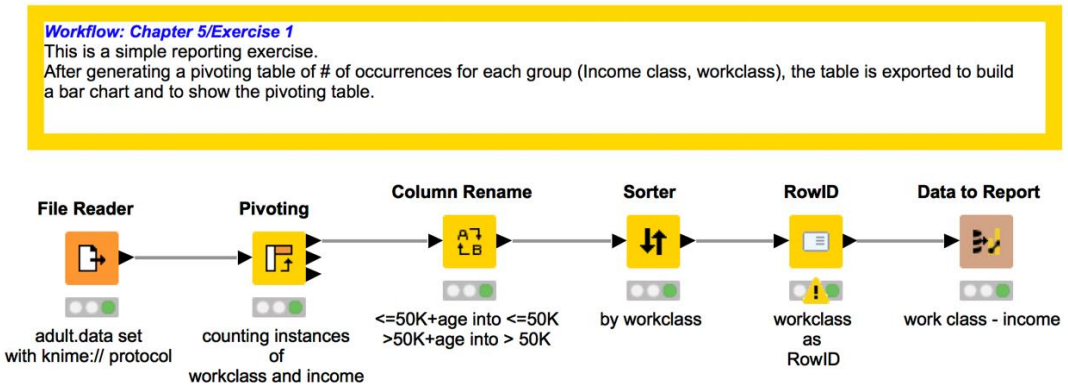
Work class	Nr of people with Income > 50K	Nr of people with Income <= 50K
Work class 1		
…		
Work class n		

- Mark this data set for reporting

Solution to Exercise 1

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

5.29. Exercise 1: The workflow



Exercise 2

Extend Exercise 1

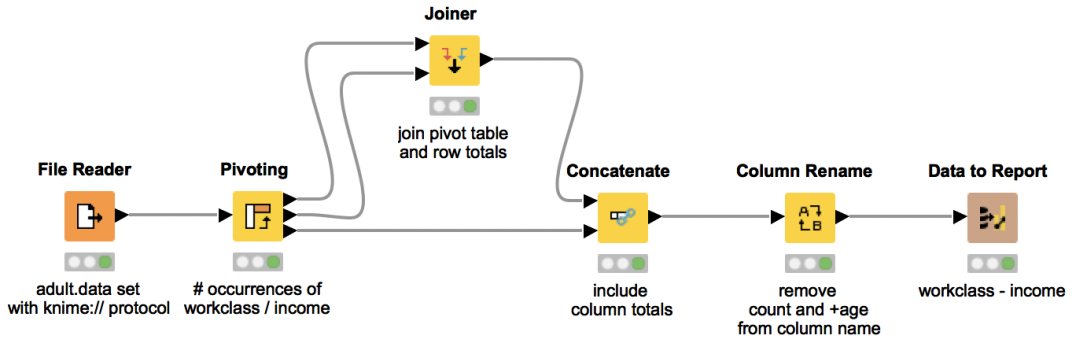
- Calculate the total number of people with income > 50K and with income <= 50K
- Calculate the total number of people for each work class
- Calculate the total number of people
- Extend the data table produced for exercise 1 as follows:

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”

5.30. Exercise 2: The workflow



Workflow: Chapter 5/Exercise 2
This is another simple reporting exercise.
After generating a pivoting table of # of occurrences for each group (Income class, workclass), the table is exported to build a simple line plot, an aggregated line plot, and to show the pivoting table.

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:

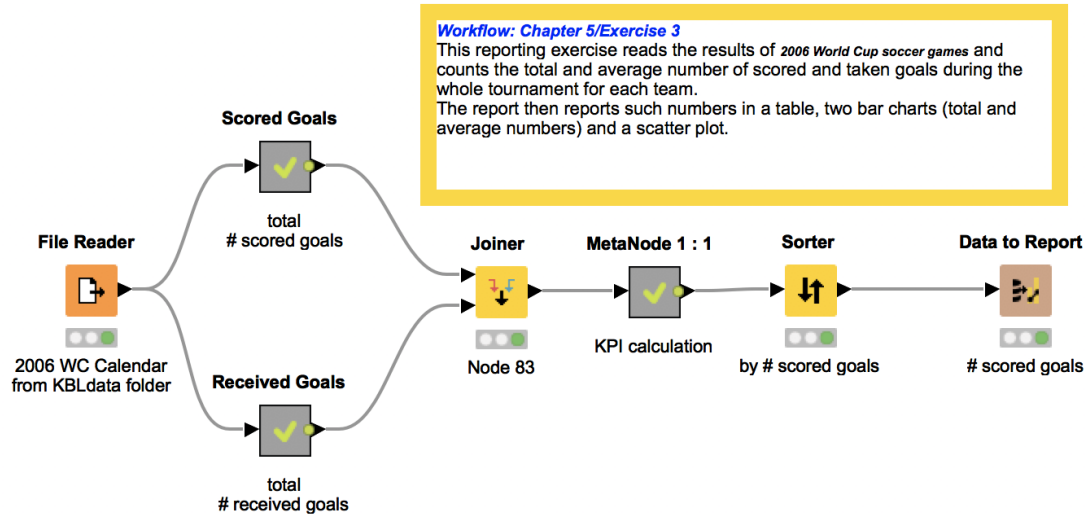
- The total number of played games
- The total number of scored goals
- The total number of taken goals
- The average number of scored goal per game
- The average number of taken goal per game
- A fit measure as: `(total number of scored goals – total number of taken goals)/number of played games`

Document each step with the appropriate node's name and description.

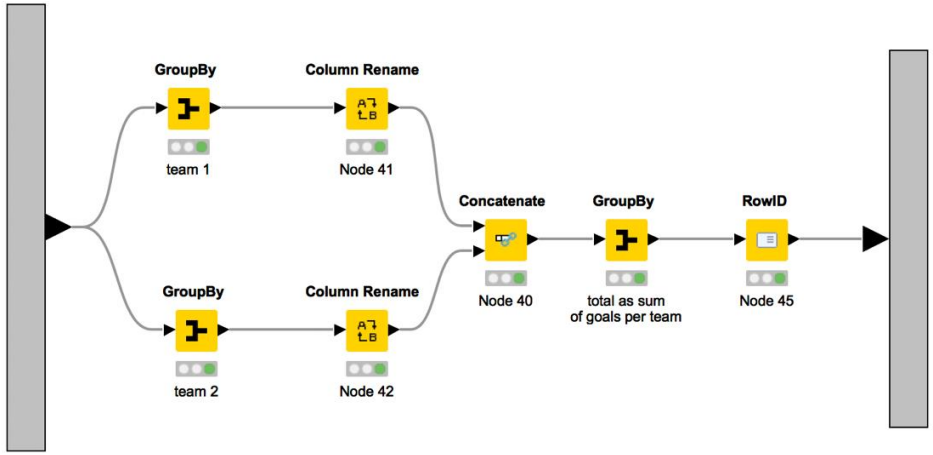
Make the workflow readable by using meta-nodes.

Solution to Exercise 3

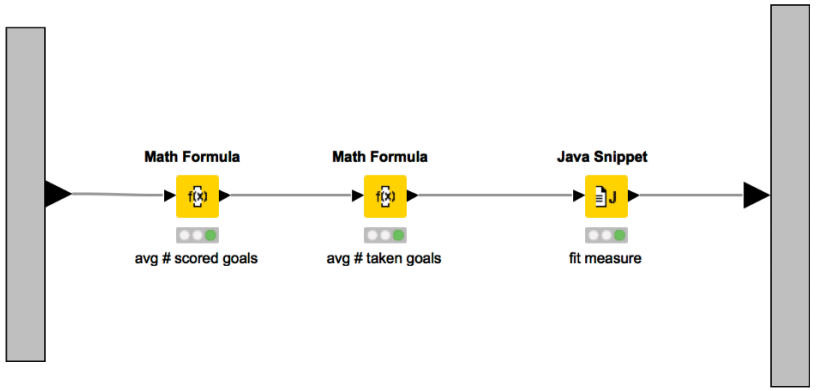
5.31. Exercise 3: workflow



5.32. Meta-node "# scored goals"/"# taken goals"



5.33. 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.

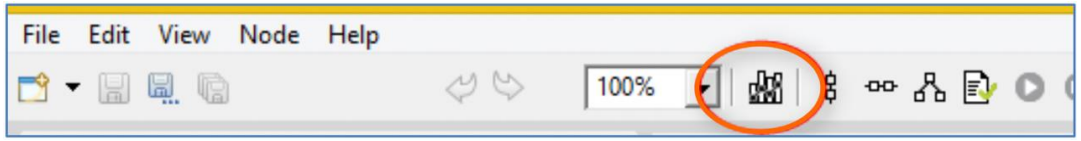
The KNIME Reporting tool is based on BIRT (Business Intelligence Reporting Tool), which is open source software for reporting. BIRT and KNIME are two different tools using the same environment with customized properties. In KNIME we develop workflows for data manipulation and modeling. In BIRT we create and shape the report to represent the workflows' data.

The KNIME integration with BIRT is not part of the core package of KNIME Analytics Platform. To get it, you need to install the reporting extension located in: "KNIME & Extensions" -> "KNIME Report Designer".

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, it will be empty. From a KNIME workflow, you can switch to the BIRT environment and open the associated report by:

- Opening the workflow from the "KNIME Explorer" panel into the workflow editor
- 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 Workflow Editor window.

6.2. The new tab in the KNIME Workflow Editor for the selected report

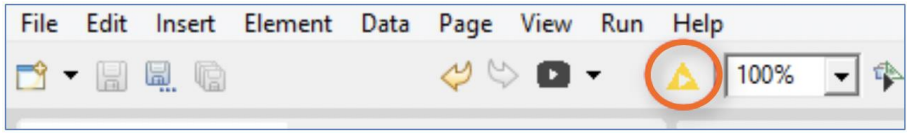
To go back from the report to the workflow editor, you can:

- Select the workflow tab

or

- Click the KNIME icon in the tool bar

6.3. The KNIME icon in the Tool Bar when a report is open



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, double-click the "Projects" workflow in the "KNIME Explorer" panel to open it; then select the report icon in the tool bar (Fig. 6.1). 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.

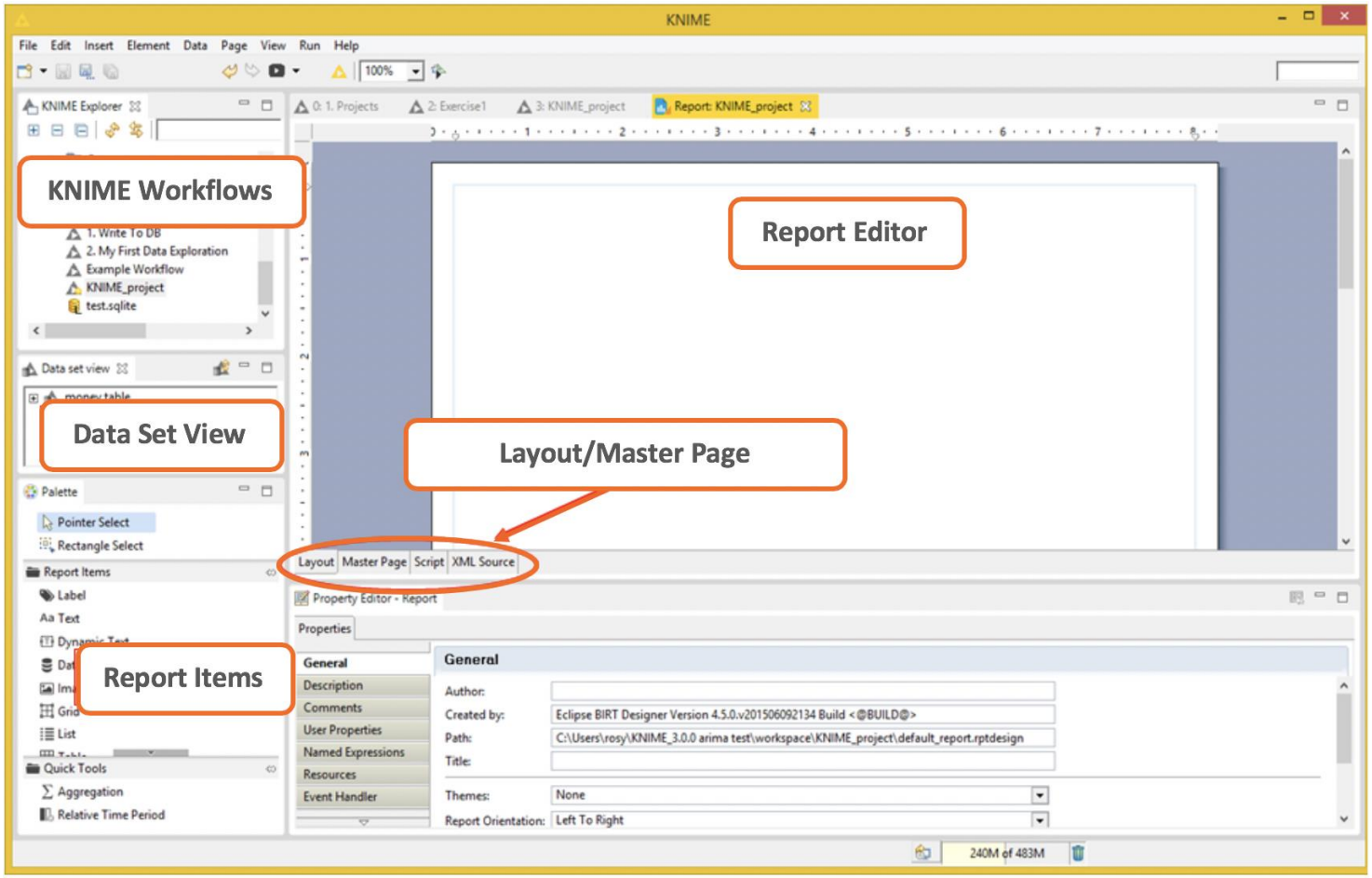
In the center, as for the KNIME workflow editor, we find the **report editor**. Like in KNIME, where we built workflows by "dragging and dropping" the nodes into the workflow editor, here we can compose the report by "dragging and dropping" the report items into the report editor.

Finally in the center bottom of the window there are a few tabs, of which only two are interesting for our work: Layout and Master Page.

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.

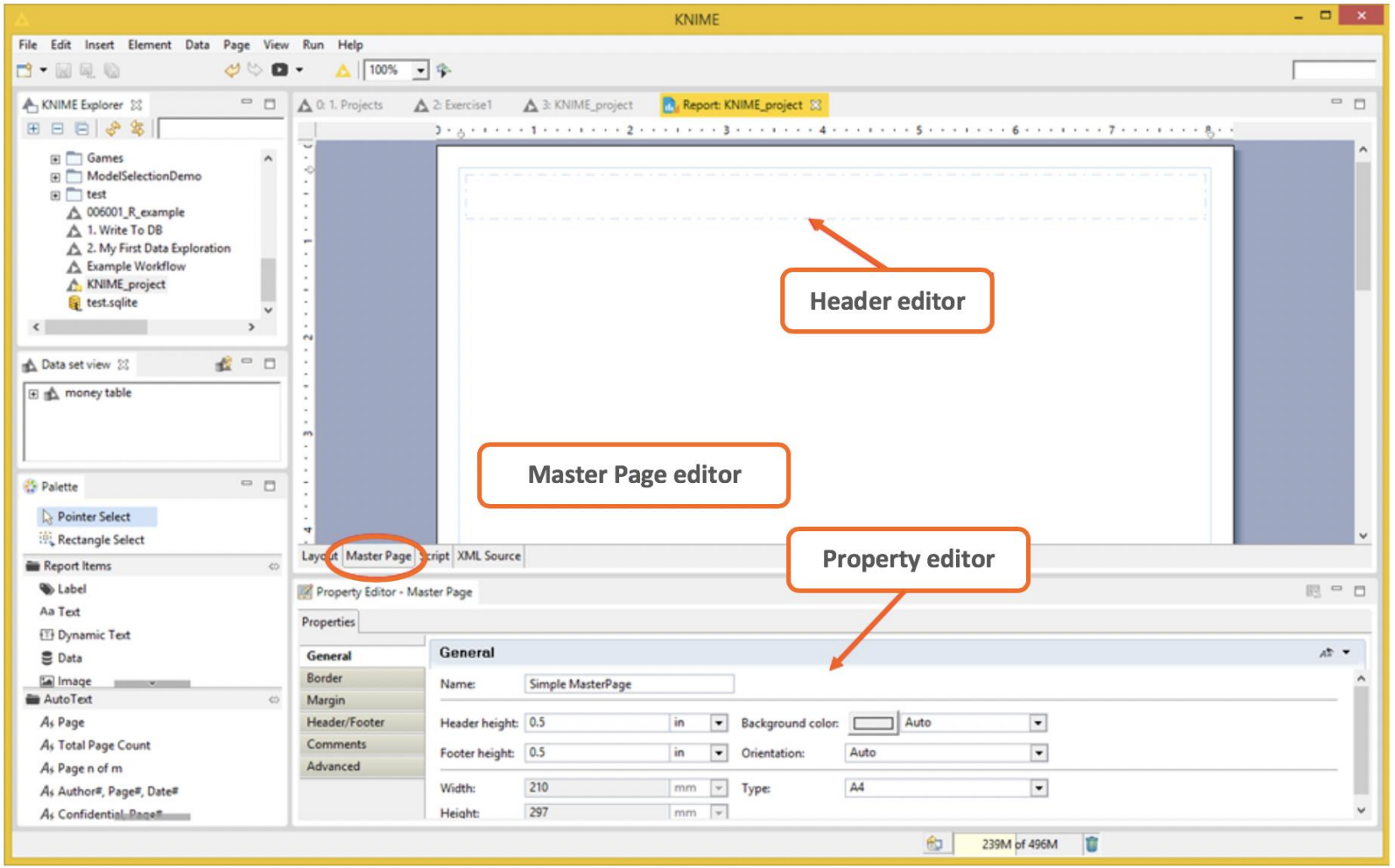
6.4. The Report Editor in the BIRT environment



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 page size, borders, running headers, footers and so on. Basically, we want to define its Master Page.

6.5. The Header Editor inside the Master Page Editor



Right below the report editor, there are a few tabs: “Layout”, “Master Page”, and others. Let’s select tab “Master Page”.

Now the report editor in the center has become the Master Page editor and, below the tabs, you can see the Master Page’s Properties Editor. There are 6 property groups: “General”, “Border”, “Margin”, “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:

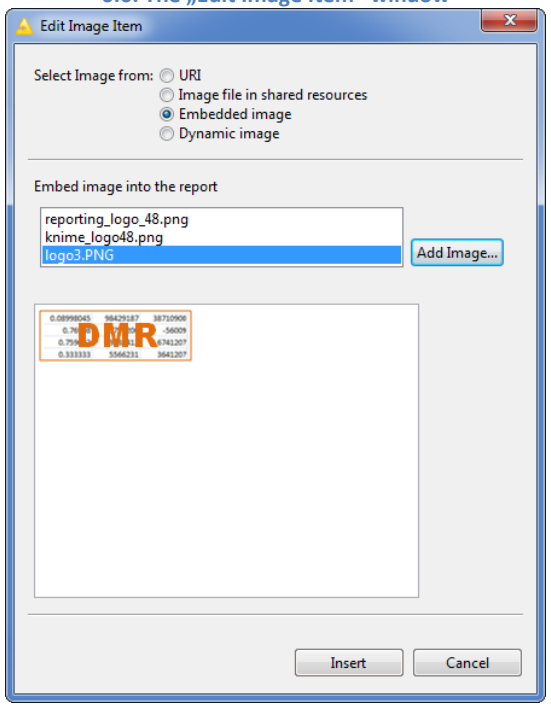
- Right-click the header editor
- Select “Insert”
- Select “Image”
- In the “Edit Image Item” window upload your image, for example as an embedded file

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.

There used to be a “Preview” tab, together with the “layout” and the “MasterPage” tab. Unfortunately, the “Preview” tab has been removed with BIRT 4.0. To see how the report will look like, you need to select “Run” -> “View Report” in the top menu and then your output format. This generates the real report. For a quick preview you can choose “In Web Viewer” for a quick creation of the HTML report page. 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

The panel named “Data set view” contains 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

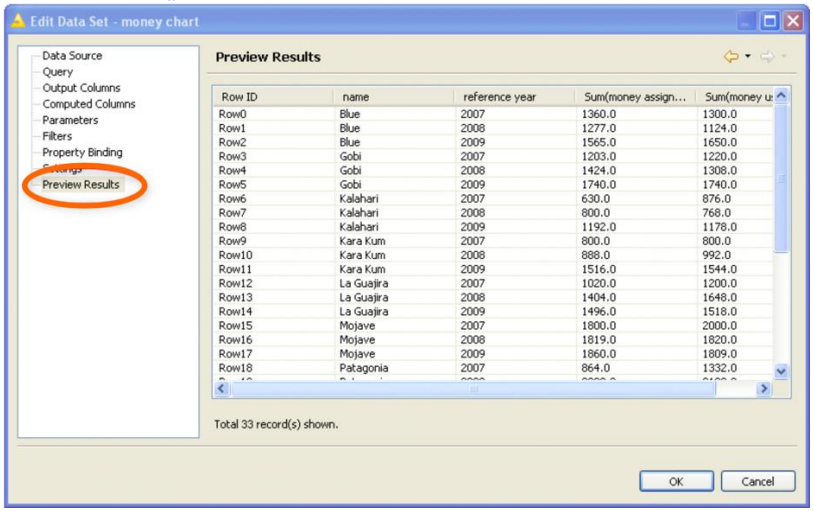
OR

- 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.

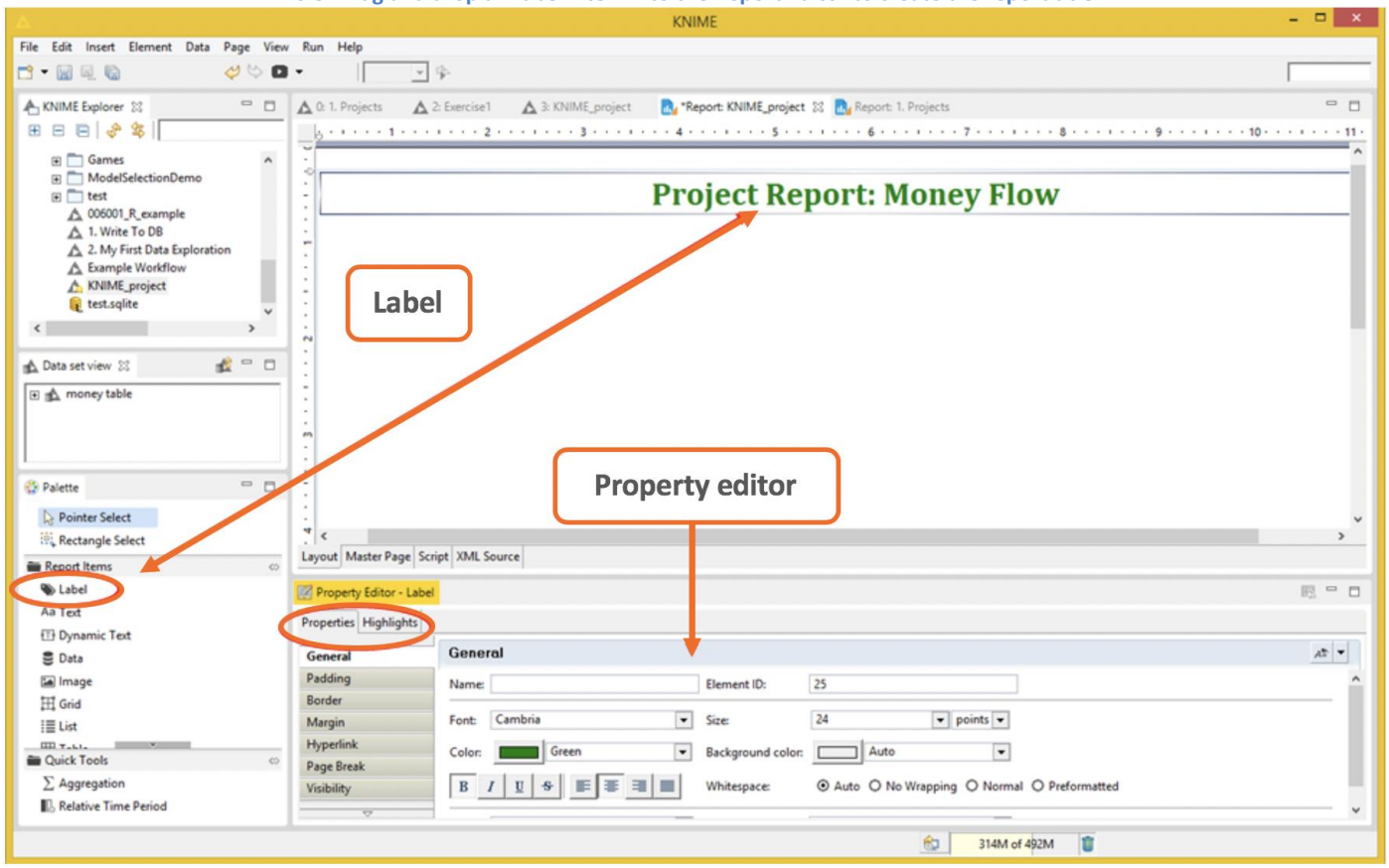
To build a title:

- 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:

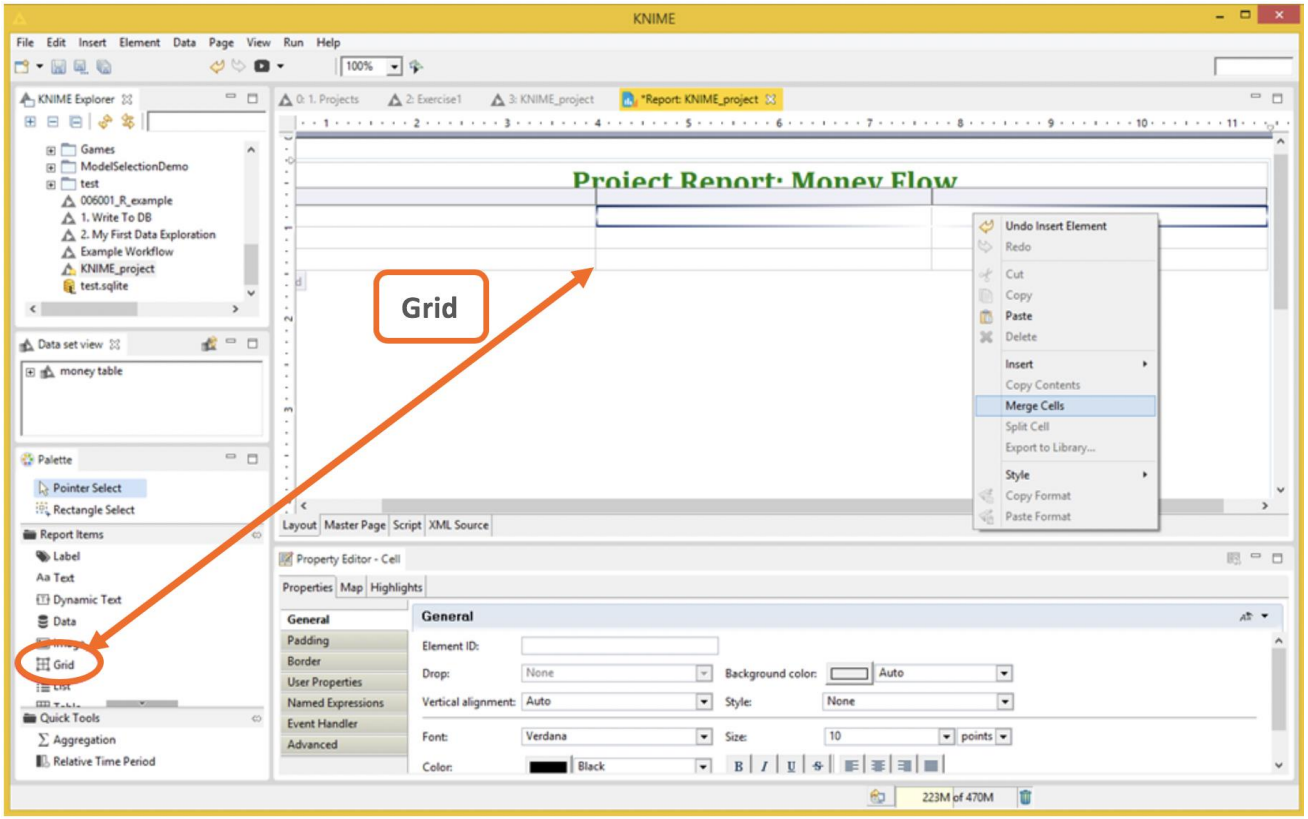
- one row with two cells: one for the assigned money table and one for the used money table
- one row with only one cell for the remaining money table
- one row with two cells again for the 2 bar charts

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

6.9. Drag and drop the “Grid” report item into the Report editor, select 3 rows and 2 columns, and merge the two cells in the middle row



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:

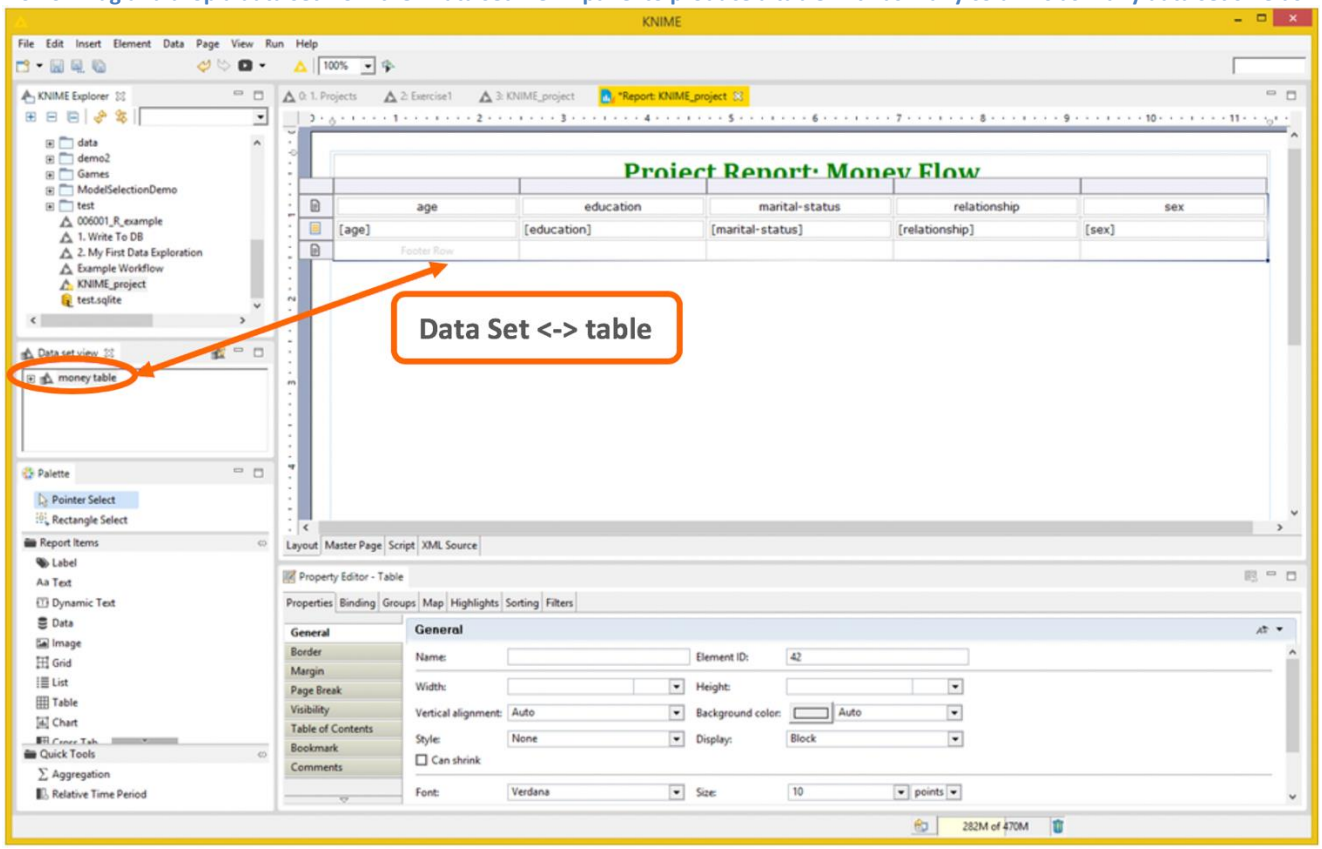
- Drag and drop the “Table” report item into the report editor
- Bind the “Table” to a data set
- Bind each data cell to a data set field

OR we can:

- Drag and drop the data set into the report editor
- In the next window, select the data columns you want to appear in the final report

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.

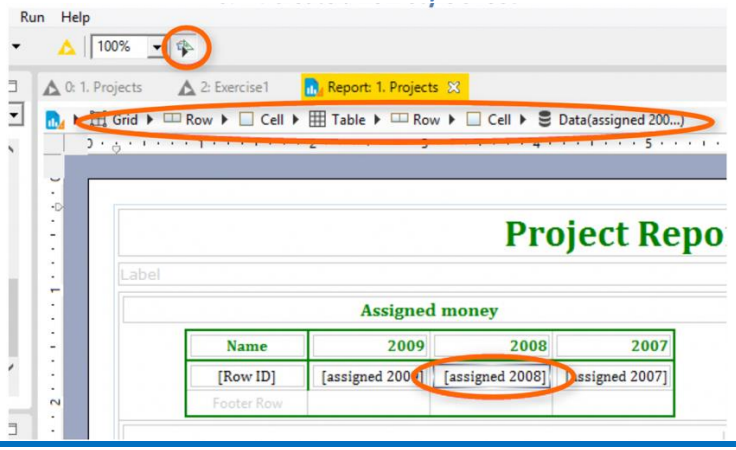
Toggle Breadcrumb

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.

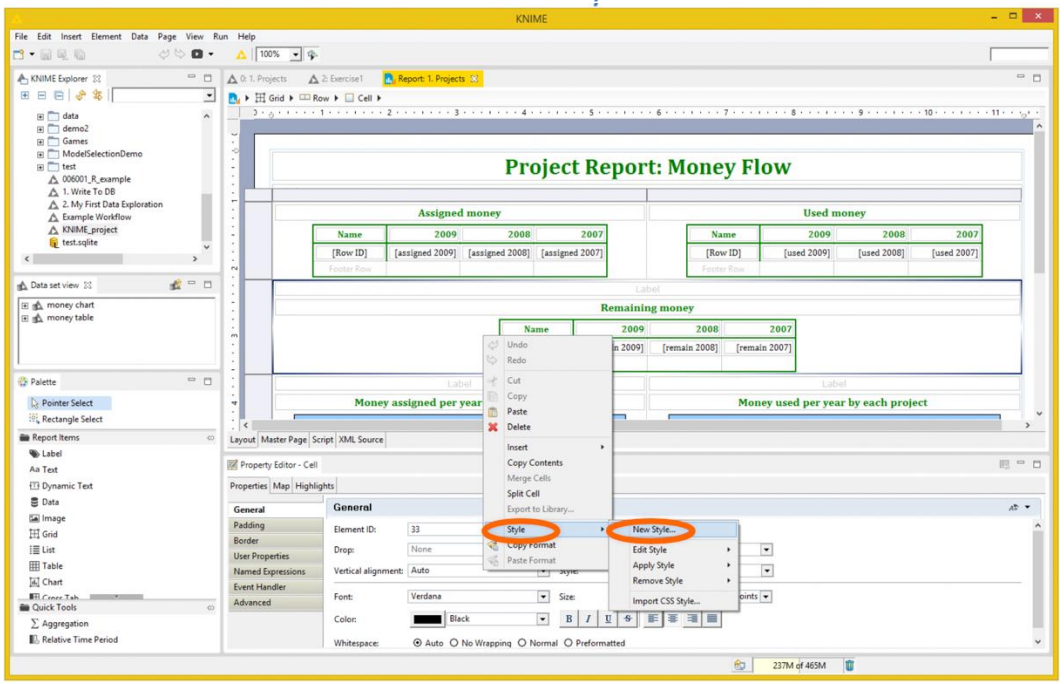
Style sheets are widely used in web programming to share style specifications across the many elements of web pages. Similarly, the KNIME reporting tool supports style sheets which can be used to apply style attributes to multiple report items.

Create a new Style Sheet

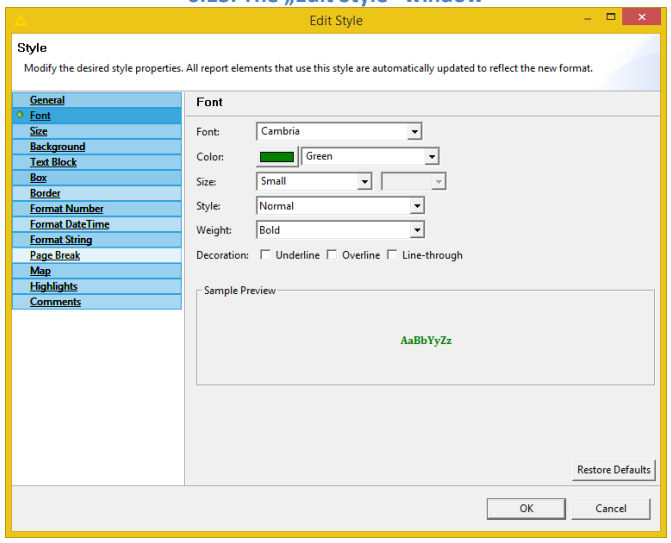
- Right-click anywhere on the report editor
- Select “Style”
- Select “New Style”

The “New Style” window opens.

6.12. Create a new Style Sheet



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

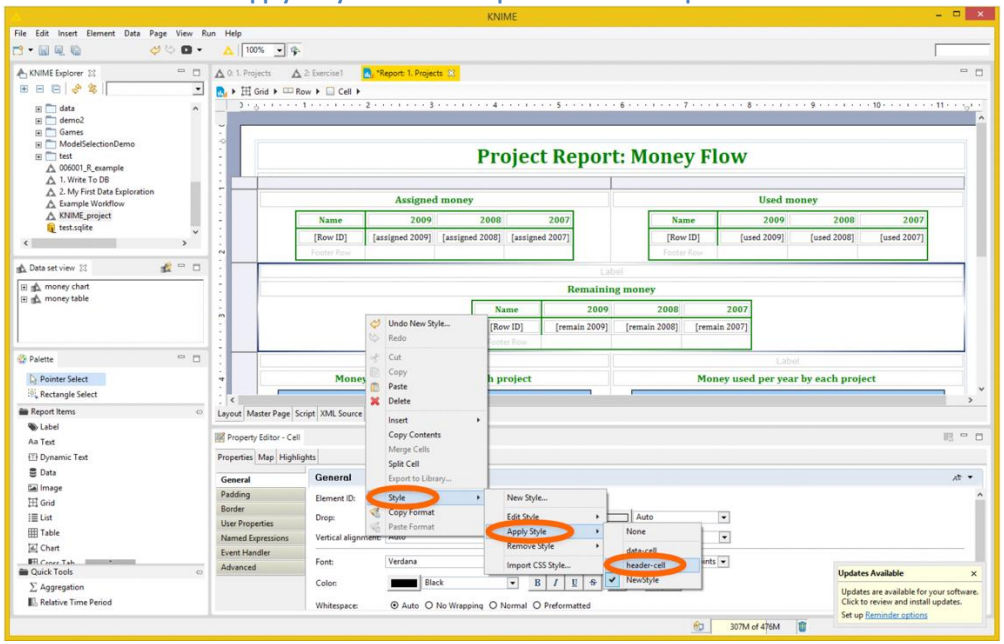
We then built two style sheets, one for the data cells and one for the header cells with the properties listed above. We chose “large” font size for both Style Sheets, named them “data cell” and “header cell” and applied them to each header cell and each data cell of the three tables.

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.

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

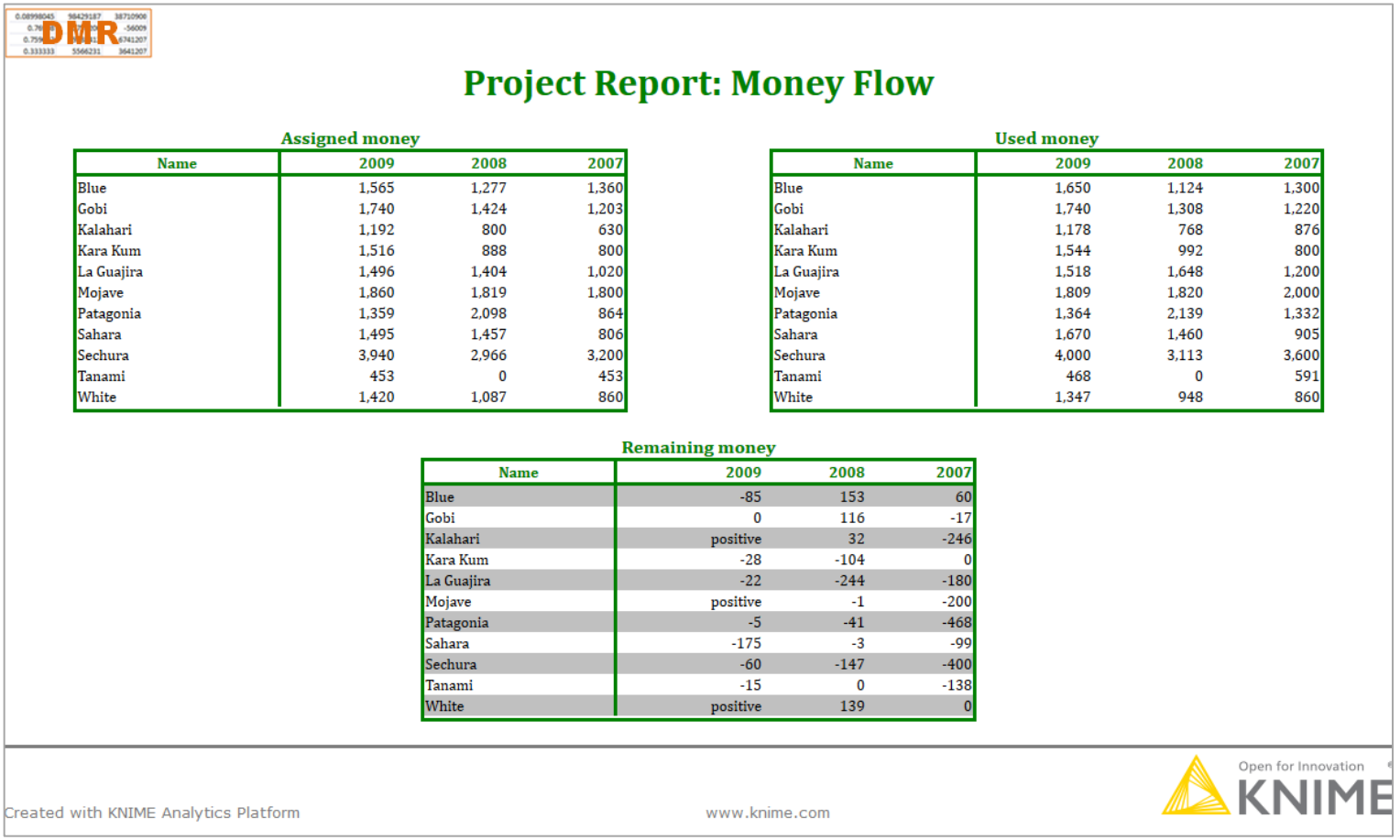
6.14. Apply a Style Sheet to a report item for example a data cell



Let’s now create the report (from top menu “Run” -> “View Report” -> “In Web Viewer”) 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 below.

6.15. Report View on a web browser after creating and formatting the three tables



Project Report: Money Flow

Assigned money

Name	2009	2008	2007
Blue	1,565	1,277	1,360
Gobi	1,740	1,424	1,203
Kalahari	1,192	800	630
Kara Kum	1,516	888	800
La Guajira	1,496	1,404	1,020
Mojave	1,860	1,819	1,800
Patagonia	1,359	2,098	864
Sahara	1,495	1,457	806
Sechura	3,940	2,966	3,200
Tanami	453	0	453
White	1,420	1,087	860

Used money

Name	2009	2008	2007
Blue	1,650	1,124	1,300
Gobi	1,740	1,308	1,220
Kalahari	1,178	768	876
Kara Kum	1,544	992	800
La Guajira	1,518	1,648	1,200
Mojave	1,809	1,820	2,000
Patagonia	1,364	2,139	1,332
Sahara	1,670	1,460	905
Sechura	4,000	3,113	3,600
Tanami	468	0	591
White	1,347	948	860

Remaining money

Name	2009	2008	2007
Blue	-85	153	60
Gobi	0	116	-17
Kalahari	positive	32	-246
Kara Kum	-28	-104	0
La Guajira	-22	-244	-180
Mojave	positive	-1	-200
Patagonia	-5	-41	-468
Sahara	-175	-3	-99
Sechura	-60	-147	-400
Tanami	-15	0	-138
White	positive	139	0

Created with KNIME Analytics Platform www.knime.com Open for Innovation KNIME

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; 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.

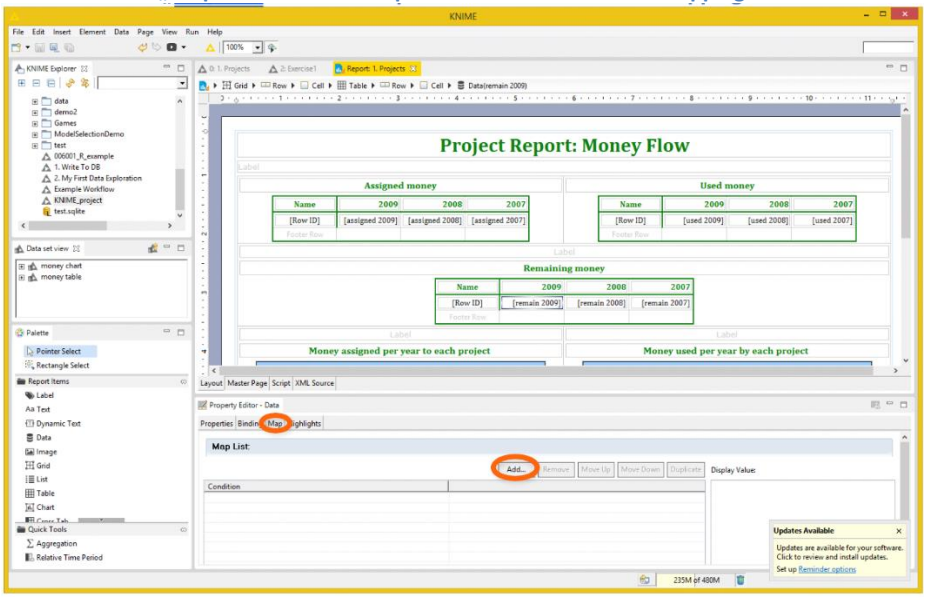
Build your condition in the "Map Rule Editor", for example:

row["remain 2009"] Greater than 0 -> "positive"

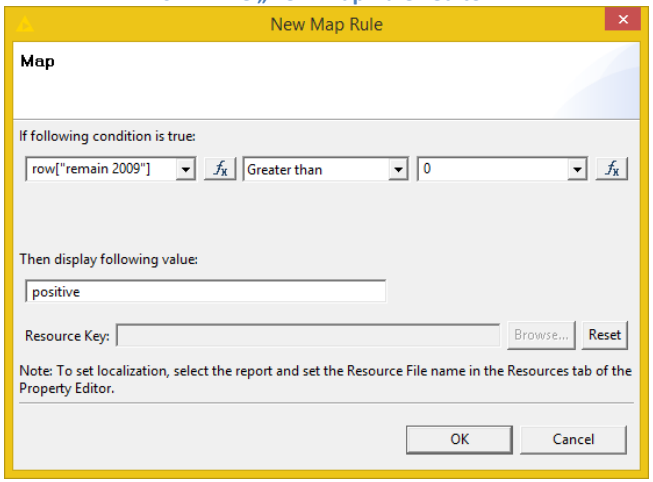
Click "OK"

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[remain 2009] smaller than 0

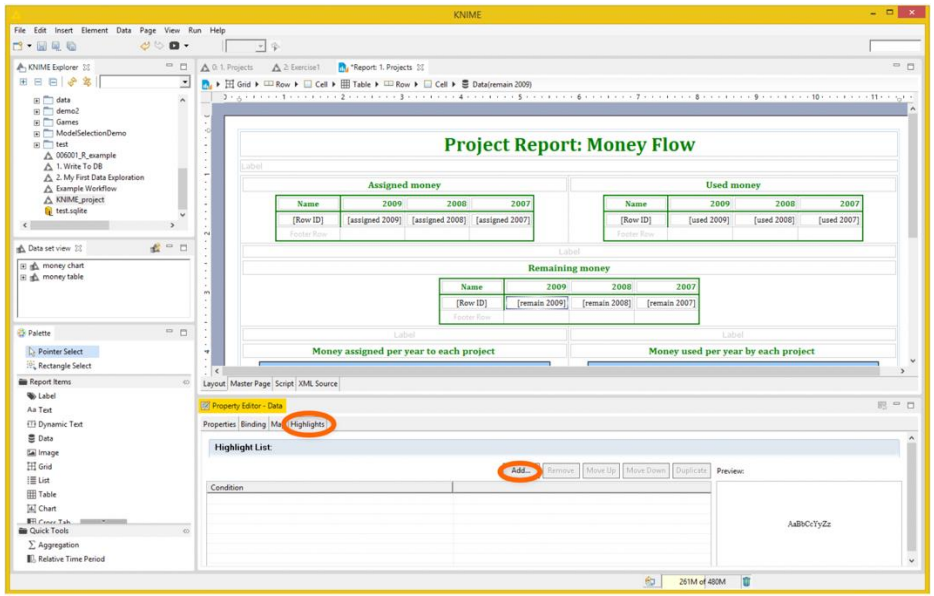
  To build the rule you can also use the “Expression Editor” which is explained later in this chapter.

- In the “Format” section, enter the formatting you want to be applied, when the condition is true.

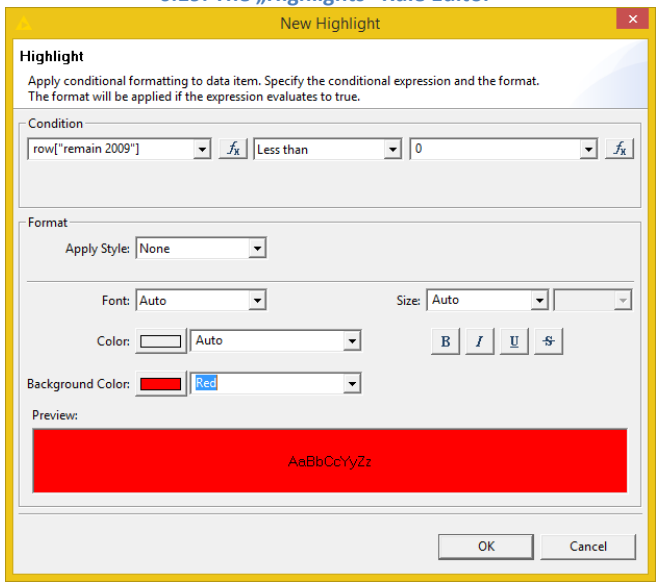
  Do this by clicking the button next to “Background Color” and then select red in the color dialog.

- Click “OK”

6.18. The „Highlights” tab in The Properties Editor defines conditional properties for a table item



6.19. The „Highlights“ Rule Editor“



After closing the Highlight dialog, run a view of the document to see the new red highlighted cells.

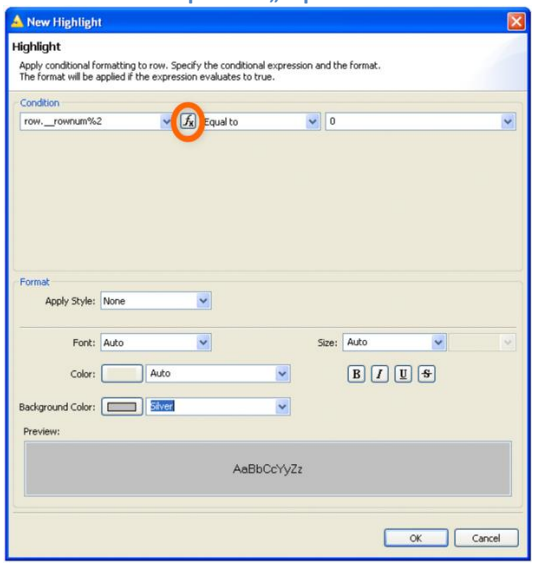
The zebra style

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"

Run a preview of the document 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

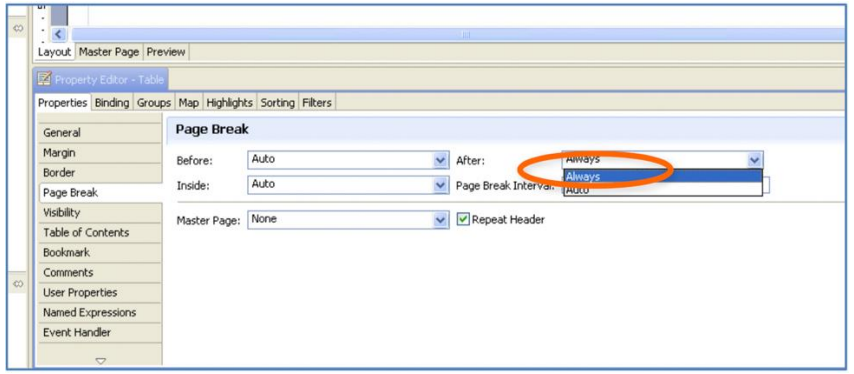
We want to export the final report to Powerpoint. This first part of our report fits nicely into a Powerpoint slide. A page break at this point would be very useful to prevent undesired page format effects in the final document.

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”

In the example workflow, the page break was set after the “remaining money” table.

6.22. „Page Break” tab in the Property editor



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.

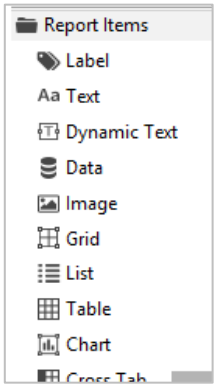
To create a chart, drag and drop the “Chart” report item from the “Report Item List” into the report editor. After the chart has been dropped, the “Chart Wizard" opens to guide you in setting the right properties for the chart.

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.

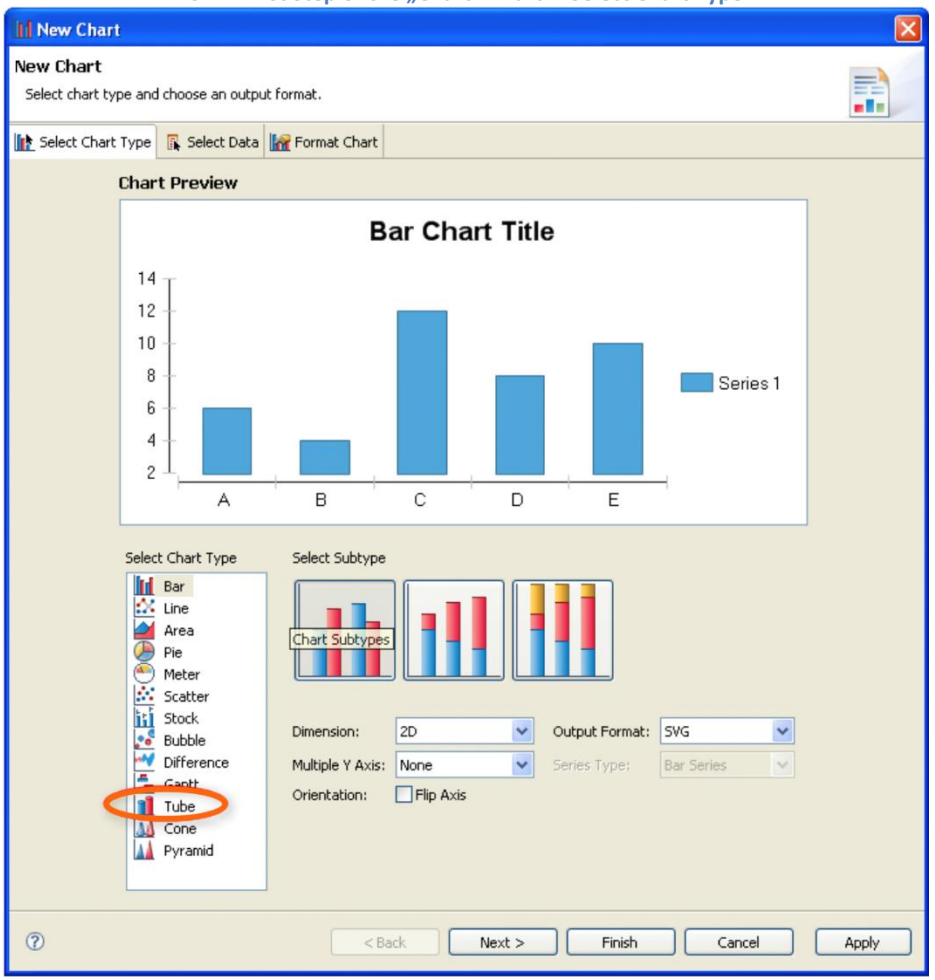
In addition, each chart can be rendered in 2D, in 2D with depth or in full 3D.

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

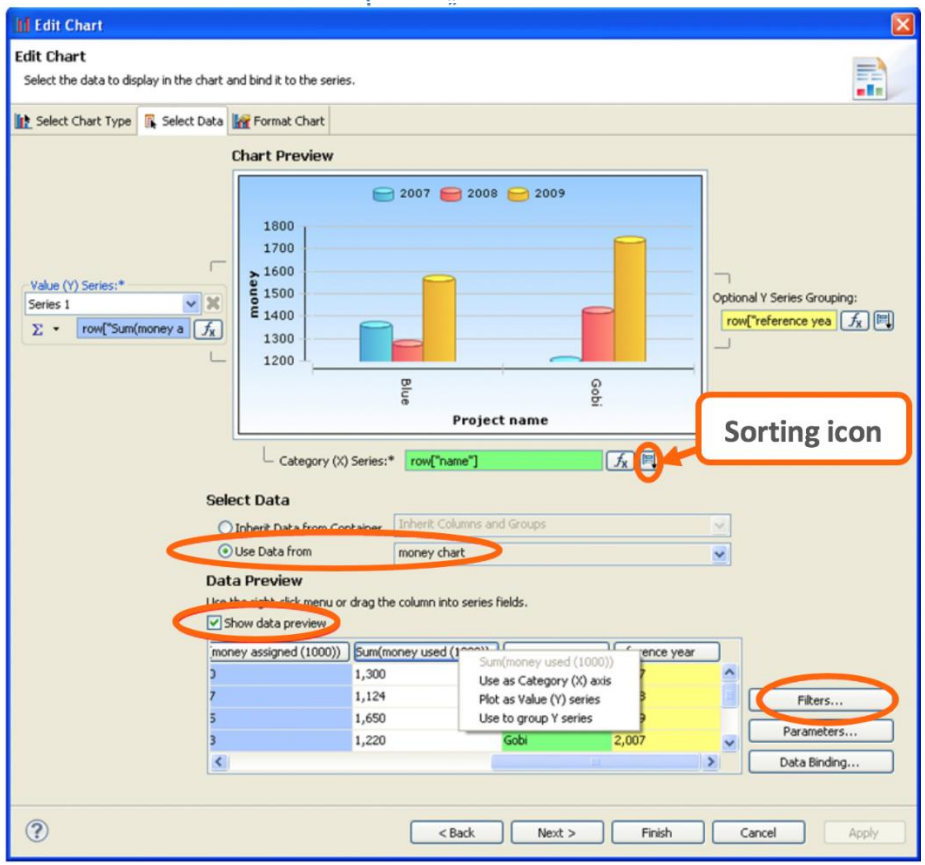


Select Data

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



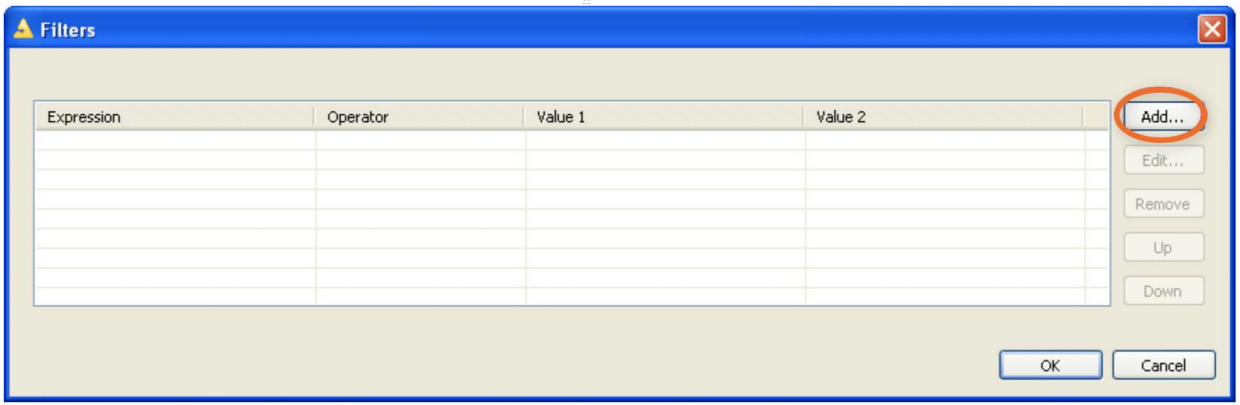
To filter rows in the data set:

- Click the “Filters” button
- In the “Filters” window, click the “Add” button

The “New Filter Condition” window appears.

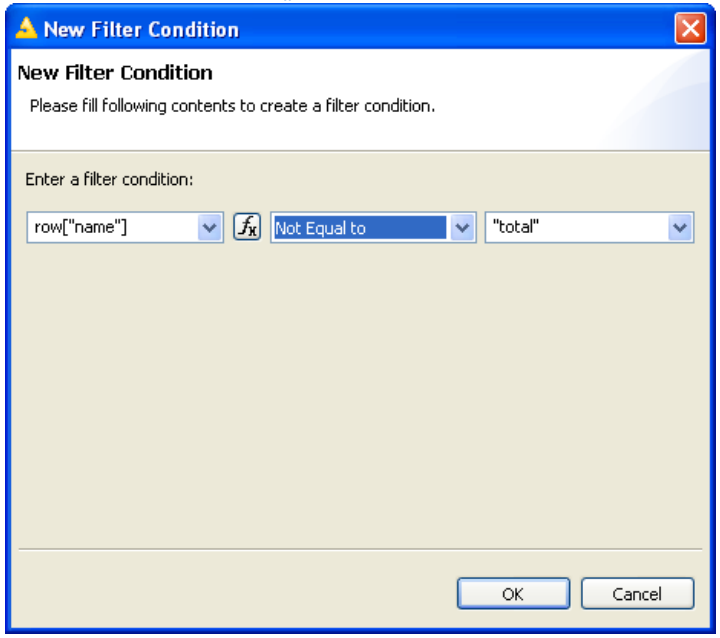
Insert your filtering rule in the “New Filter Condition” window.

6.26. The „Filters“ 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.

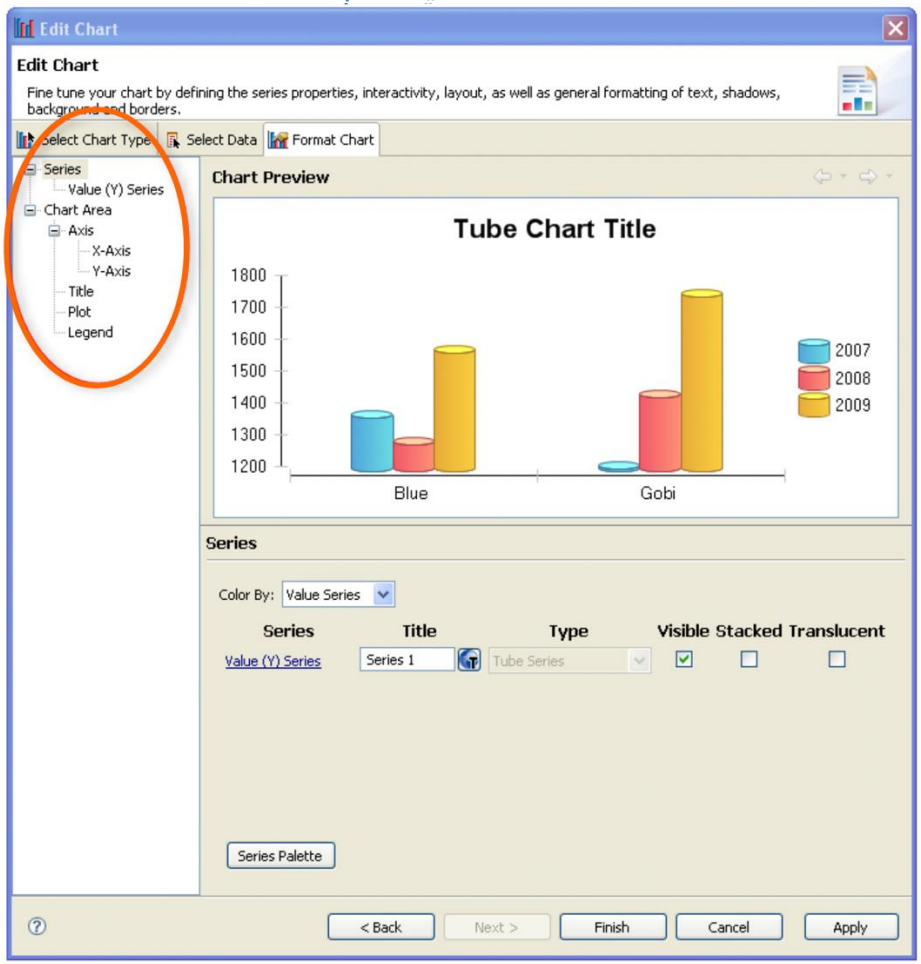
Under "**Axis**", you can define labels, scale, gridlines and everything else related to the chart axis (X-axis or Y-axis).

"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



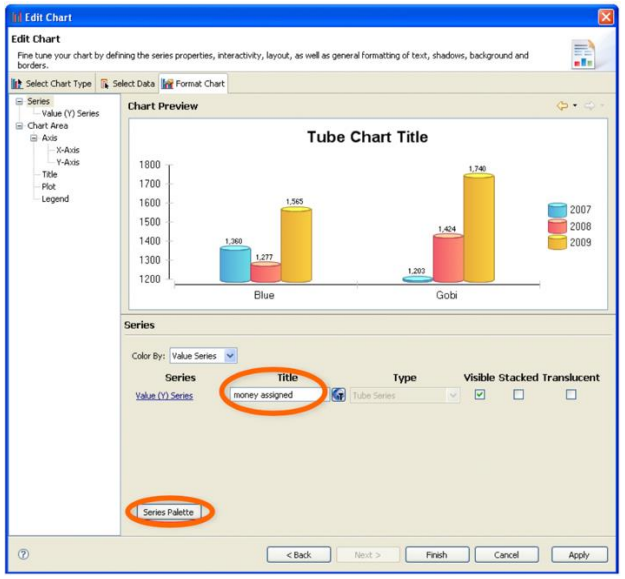
Series

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.

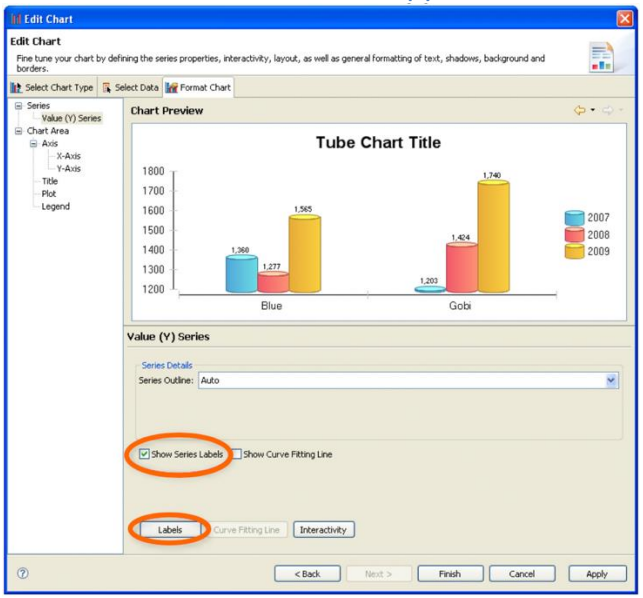
You can also define which values you want shown on top of each point: current Y-value, percent Y-value, X-value, or series name. The label can also be built around the shown value with a prefix, a suffix, and a separator.

The small button with an “A” inside leads to the “Font Editor”.

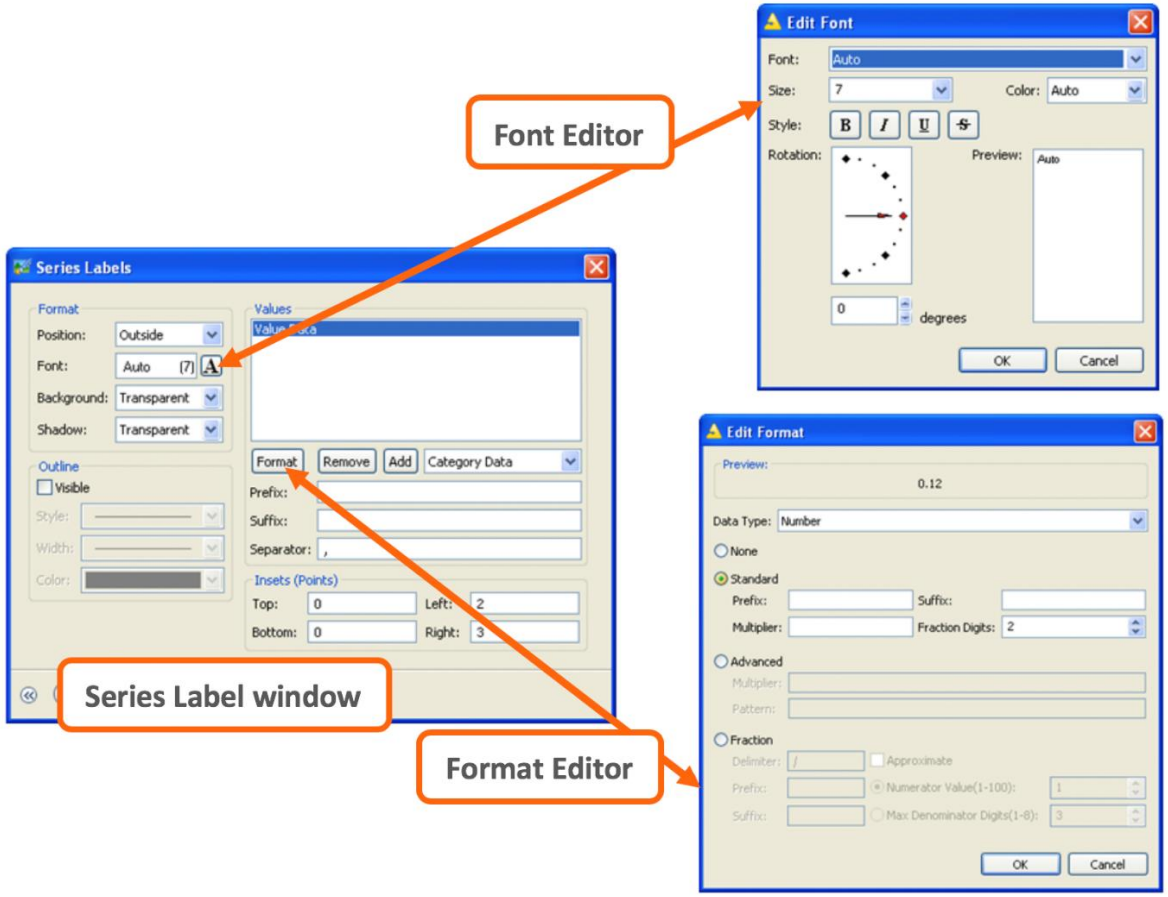
The “Format” button leads to the “Format Editor” (you need to select an item in the “Values” list to enable this button).

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.

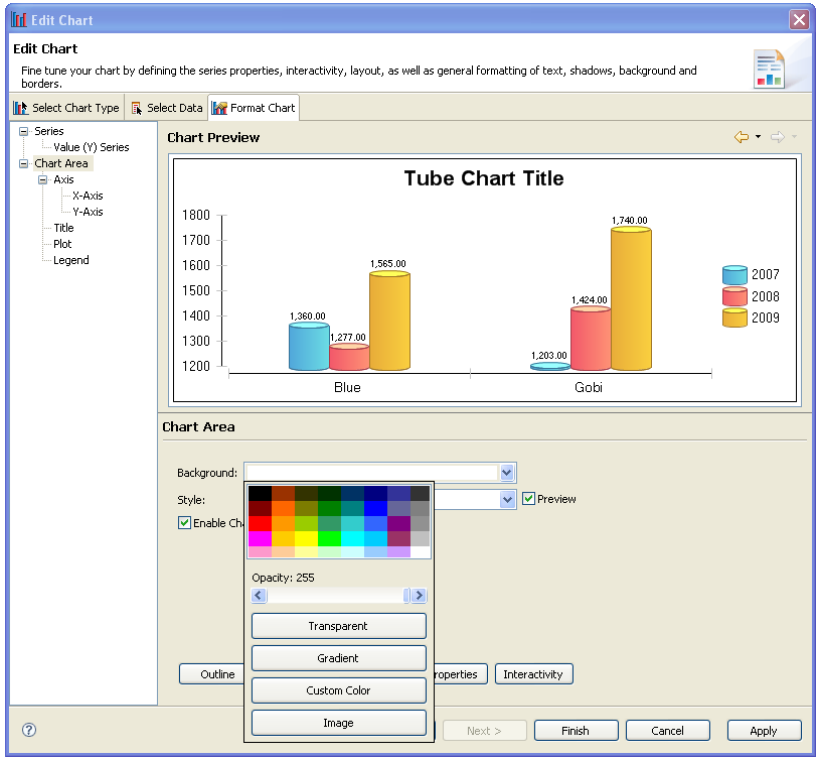
Chart Area

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:

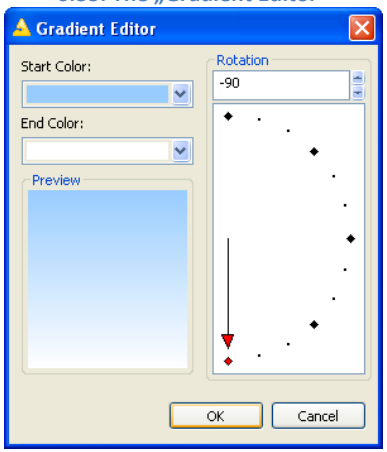
- A simple color
- “Transparent” which means no background color
- A gradient between two colors
- A custom color
- An image

6.32. Chart Format: Chart Area

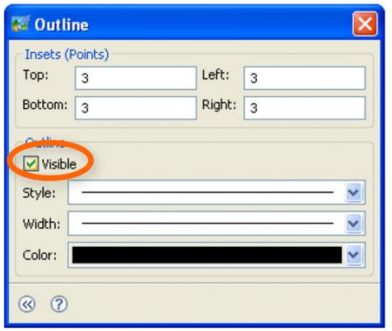


We selected the “Gradient” option. The “Gradient Editor” needs the start and end color and the gradient direction expressed in degrees. Finally, we made the chart outline visible by clicking the “Outline” button and enabling the option “Visible” in the “Outline Editor”.

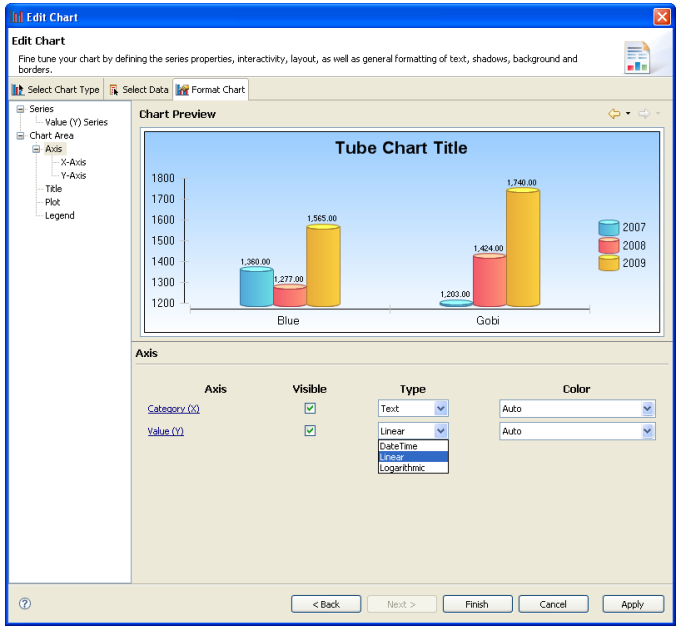
6.33. The „Gradient Editor“



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”.

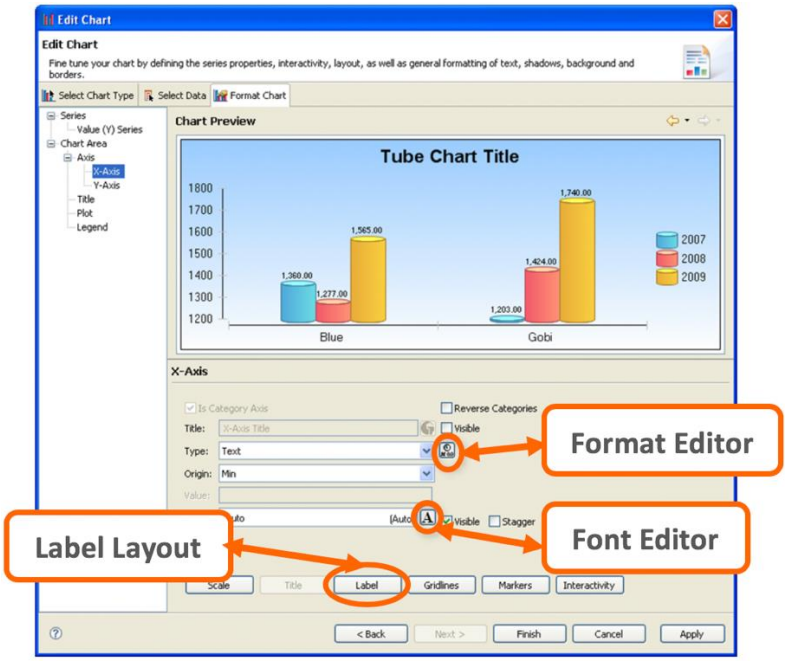
The “Label” button leads to the “Label Layout Editor”, where we can define the label position, background, outline, etc….

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.

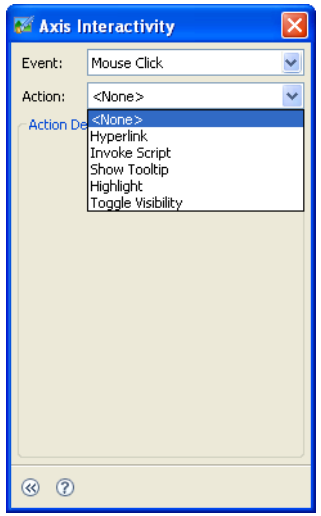
The “Markers” button introduces lines to mark areas of the plot.

6.36. Chart Format: X-axis



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.

6.37. The „Axis Interactivity“ window



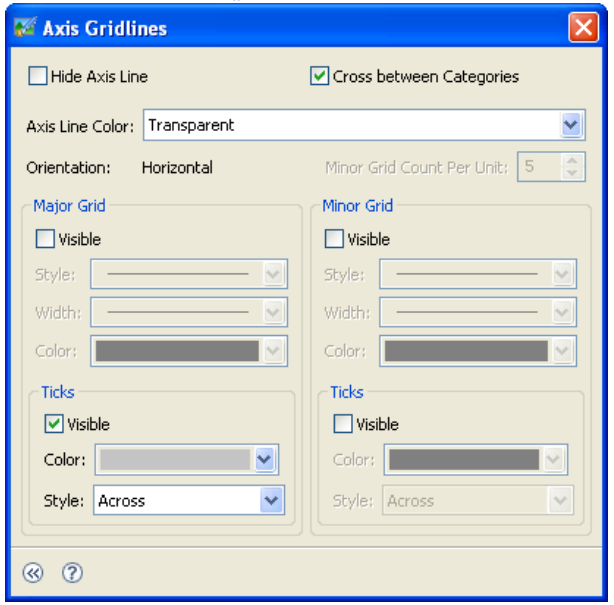
The “Gridlines” button opens the “Axis Gridlines” window to enable gridlines for this axis; that is horizontal gridlines for the Y-axis and vertical gridlines for the X-axis.

There are major and minor grids on the plot as well as ticks on the axis.

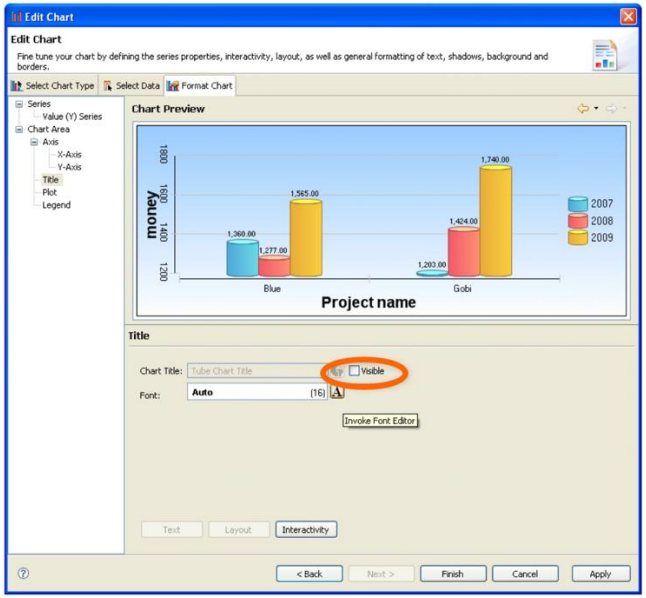
For the “Projects” report we enabled the following:

- 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.

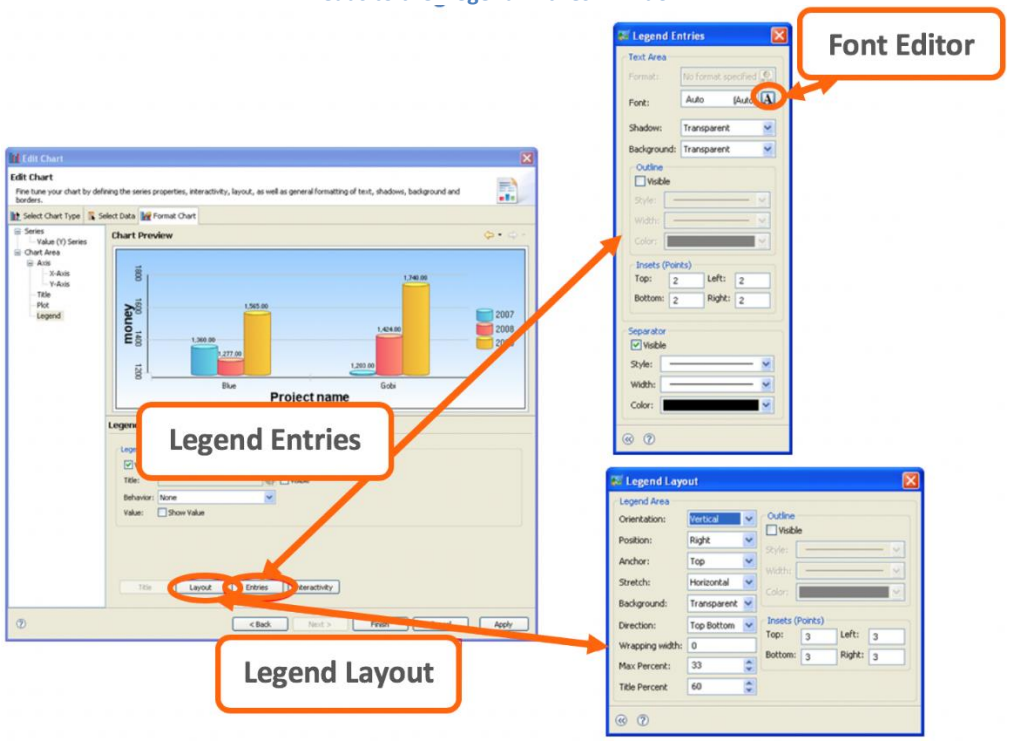
Legend

"**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 a preview of the document. 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 "Chart Preview", right-click the header of column "Sum(money used (1000))"
- Select "Plot as Value Y Series"
- Click the "Finish" button

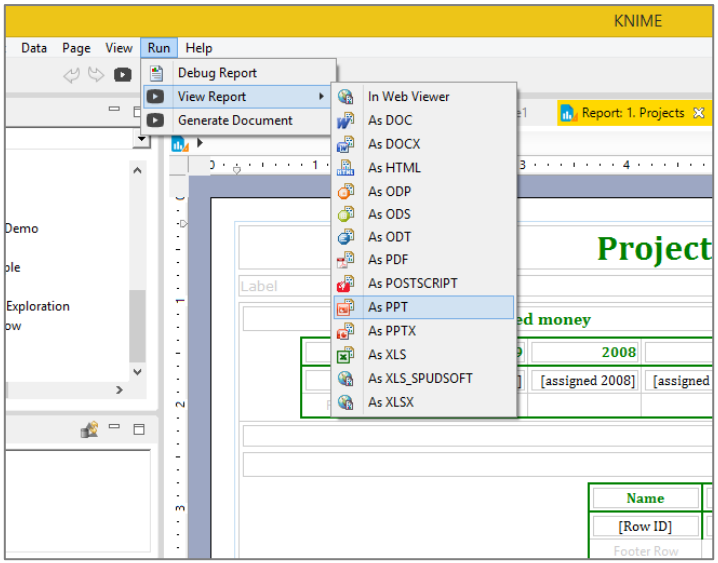
6.13. Generate the final document

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.

Alternatively, “Run” -> “Generate Document” directly generates the file in the preferred 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]

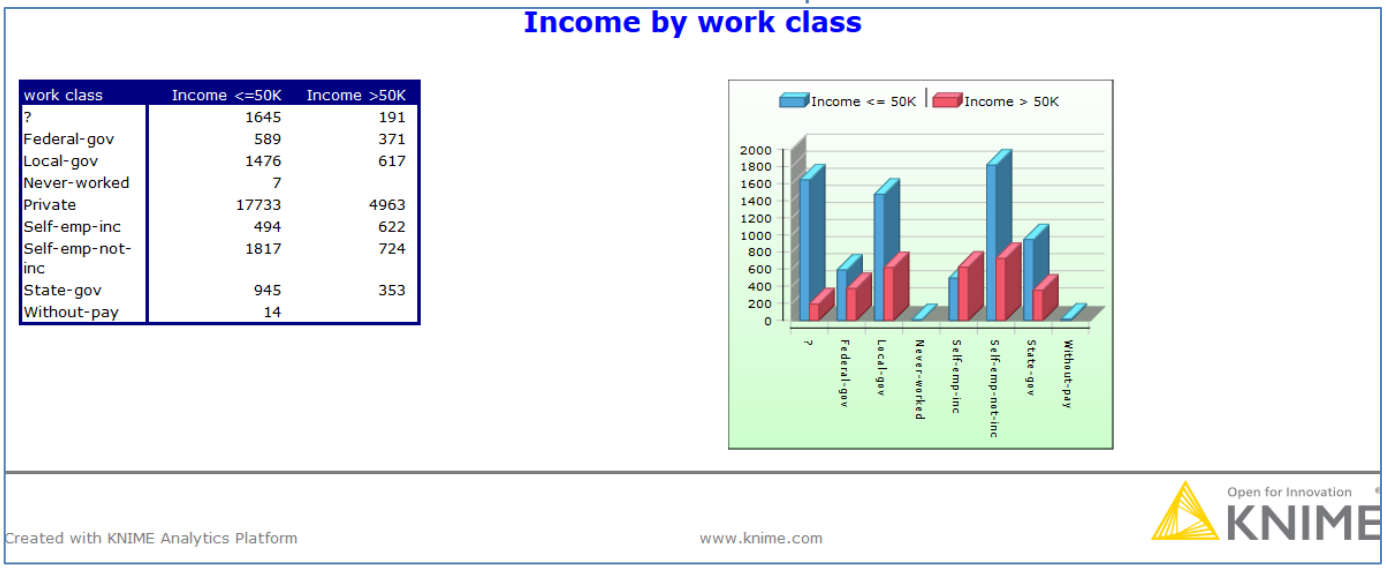
- A bar chart with:
    - Work class on the X-axis

- “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

The report must look the same as in the previous exercise.

Exercise 2

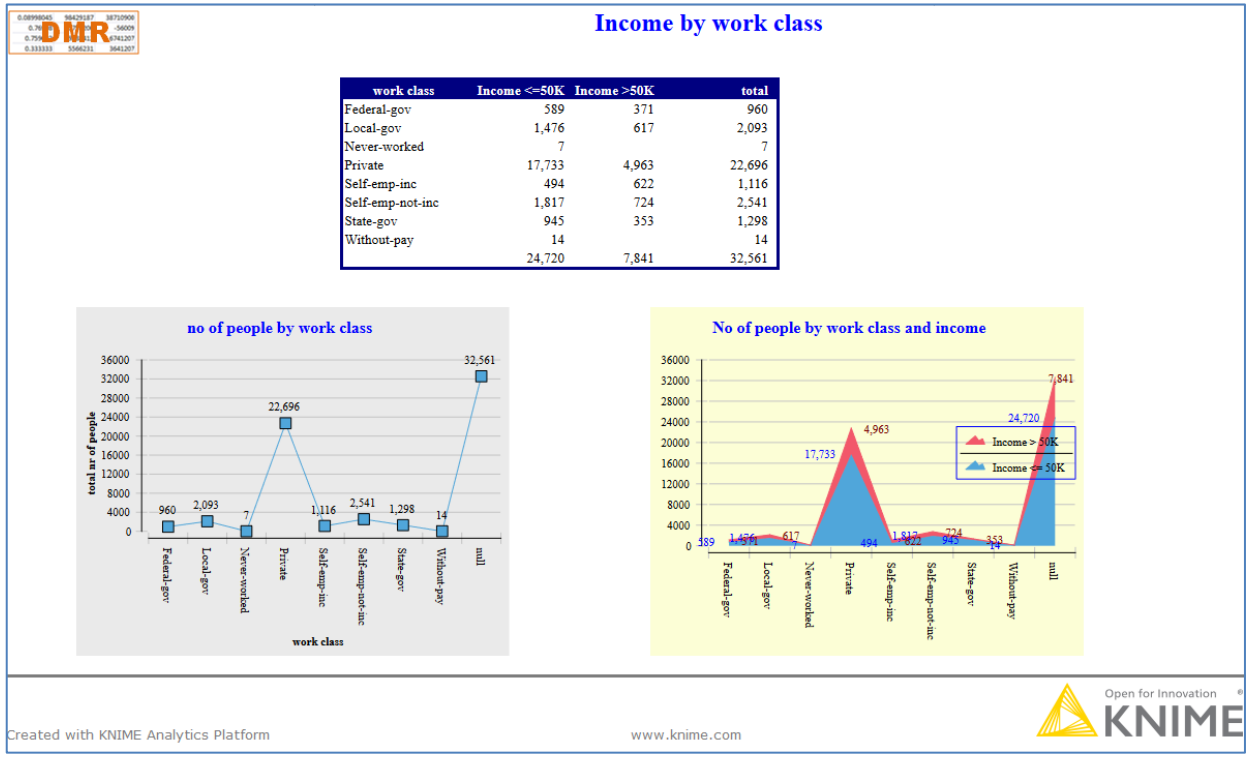
From the workflow developed in Chapter5/Exercise2 build a report with:

- 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 HTML

Solution to Exercise 2

6.43 Exercise 2: the final report as HTML



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)

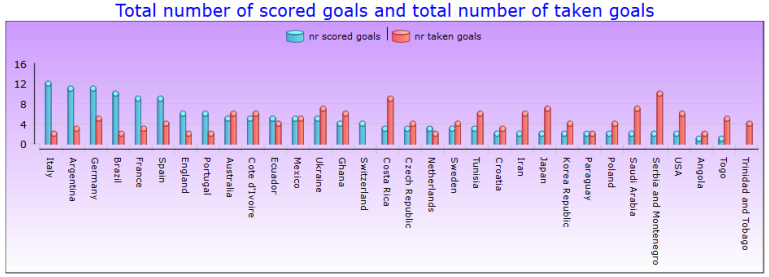
- Bar/Tube Chart with average number of scored goals and average numbers of taken goals (Y-axis) vs. team (X-axis)
- Average number of scored goals vs. average number of taken goals with team name as label on each chart point
- Format legends, axis, axis titles, etc … so that the charts are readable
- 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).
- Export as HTML

Solution to Exercise 3

6.44. Exercise 3: final report, part 1

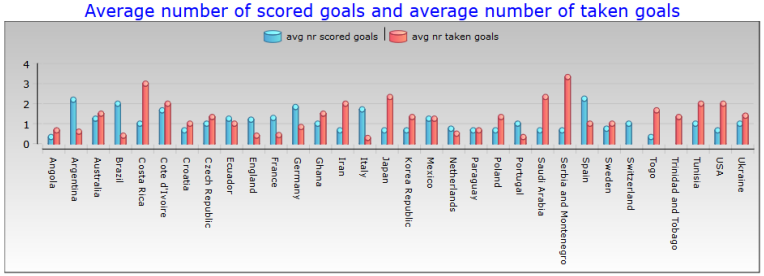
Soccer World Cup 2006

Total number of scored goals and total number of taken goals



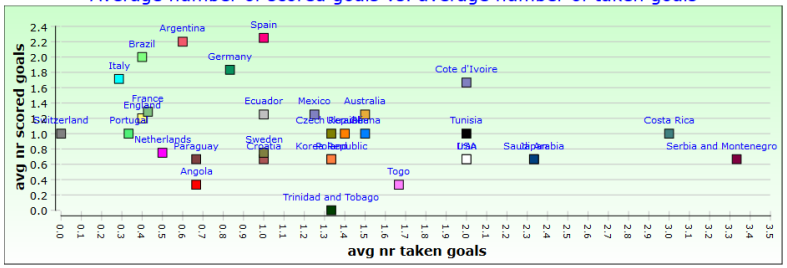
6.45. Exercise 3: final report, part 2

Average number of scored goals and average number of taken goals



6.46. Exercise 3: final report, part 3

Average number of scored goals vs. average number of taken goals



6.47. Exercise 3: final report, part 4

Team	nr games	nr scored goals	nr taken goals	avg nr scored goals	avg nr taken goals	fit measure
Italy	7	12	2	1.71	0.29	1.43
Argentina	5	11	3	2.20	0.60	1.60
Germany	6	11	5	1.83	0.83	1.00
Brazil	5	10	2	2.00	0.40	1.60
France	7	9	3	1.29	0.43	0.86
Spain	4	9	4	2.25	1.00	1.25
England	5	6	2	1.20	0.40	0.80
Portugal	6	6	2	1.00	0.33	0.67
Australia	4	5	6	1.25	1.50	-0.25
Cote d'Ivoire	3	5	6	1.67	2.00	-0.33
Ecuador	4	5	4	1.25	1.00	0.25
Mexico	4	5	5	1.25	1.25	0.00
Ukraine	5	5	7	1.00	1.40	-0.40
Ghana	4	4	6	1.00	1.50	-0.50
Switzerland	4	4	0	1.00	0.00	1.00
Costa Rica	3	3	9	1.00	3.00	-2.00
Czech Republic	3	3	4	1.00	1.33	-0.33
Netherlands	4	3	2	0.75	0.50	0.25
Sweden	4	3	4	0.75	1.00	-0.25
Tunisia	3	3	6	1.00	2.00	-1.00
Croatia	3	2	3	0.67	1.00	-0.33
Iran	3	2	6	0.67	2.00	-1.33
Japan	3	2	7	0.67	2.33	-1.67
Korea Republic	3	2	4	0.67	1.33	-0.67
Paraguay	3	2	2	0.67	0.67	0.00
Poland	3	2	4	0.67	1.33	-0.67
Saudi Arabia	3	2	7	0.67	2.33	-1.67
Serbia and Montenegro	3	2	10	0.67	3.33	-2.67
USA	3	2	6	0.67	2.00	-1.33
Angola	3	1	2	0.33	0.67	-0.33
Togo	3	1	5	0.33	1.67	-1.33
Trinidad and Tobago	3	0	4	0.00	1.33	-1.33

Created with KNIME Analytics Platform www.knime.com

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

E

F

G

H

I

J

K

L

M

N

P

R

S

T

U

V

W

Z

KNIME Beginner's Luck: A Guide to KNIME Analytics Platform for Beginners

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 3.7.

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's 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 data science consulting work, becoming a KNIME trainer and an expert in the KNIME Reporting tool.

ISBN: 978-3-033-02850-0