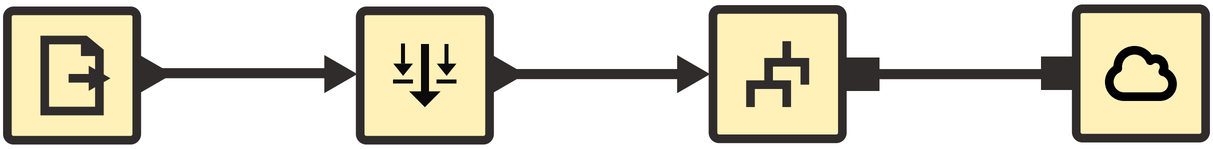
KNIME Textbooks

ROSARIA **SILIPO** SANKET **JOSHI**

KNIME Beginner’s Luck

KNIME v5.2



Open for Innovation
KNIME

This book has been updated for **KNIME 5.2**.

For information regarding permissions and sales, write to:

KNIME Press
Talacker 50
8001 Zurich
Switzerland

knimepress@knime.com

ISBN: 978-3-9523926-5-2

www.knime.com

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 Casiana Rimbu for the help in providing the most beautiful, clear, and artistic screenshots I could ever imagine and 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.

Preface

This is the first book I wrote in 2010 for the KNIME Press on how to use KNIME Analytics Platform. Since then, we (the KNIME Press Team and I) have been constantly updating the book twice a year every year, following each new release of KNIE Analytics Platform; not immediately after – but close enough.

That is right! KNIME Beginner's Luck, like all other e-books from KNIME Press, is a live e-book, constantly changing to fit the newest version of the software – which is also true for the new and improved UX/UI that came with KNIME Analytics Platform Version 5 in summer 2023. This liveness of the e-book is also the reason why it has only rarely been printed. Updating printed pages is undoubtedly harder than updating a pdf file!

As this is the first book, it is inevitably about the basics: the basics of KNIME Analytics Platform of course and the basics of a data science project. This book guides you through the most important access functions, data transformation operations, and of course machine learning nodes available in KNIME Analytics Platform. Supplemented with many example workflows, exercises, and screenshots, it will quickly familiarize you with the basic functions of the software. If you are looking for more advanced topics, you won't find them here, instead….

If you want to learn more about advanced machine learning algorithms, flow variables, or loops, check the sequel to this book: "KNIME Advanced Luck". If you want to learn more about text processing, have a look at the book, "From Words To Wisdom". If you come from that school of thoughts where reading manuals or instructions is overrated, you can start directly with reading about solutions to case studies in various application fields in our collection "Practicing Data Science". If your job is more about integrating and blending different data sources and data types, then the book for you is the "Will they blend?" collection. More useful booklets are available on the KNIME Press page, if you are transitioning from Excel, Alteryx, SPSS Modeler, or SAS.

All this is to say that the KNIME Press team and I have been working hard to provide you with the learning material, books, and tutorials, to become progressively more and more productive with KNIME Software and data science concepts.

Rosaria Silipo

Table of Contents

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 this 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 Science (DS)? The options available are of course multiple and growing rapidly. If our professional does not control an excessive budget, he or she 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 and control that many professionals support.

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

Even though KNIME Analytics Platform is very simple and intuitive to use, any beginner would profit from an accelerated orientation through all of the 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 Business Hub, which is not open-source, or topics which are considered advanced. Flow Variables, for example, and implementations of database SQL queries are discussed in the sequel book "KNIME Advanced Luck".

This book is divided into six chapters. The first chapter covers the basic concepts of KNIME Analytics Platform, while chapter two takes the reader by the hand into the implementation of the very first KNIME application. From the third chapter, we start the exploration of data science concepts in a more in-depth manner. The third chapter indeed explains how to perform some basic data exploration and visualization, in terms of nodes and processing flow. Chapter four is dedicated to data modeling. It covers a few demonstrative approaches to machine learning, Naïve Bayes, decision trees, and artificial neural networks. Finally, chapters five, six, and seven are dedicated to reporting. Usually, the results of an investigation based on data visualization or, in a later phase, on data modeling must be shown at some point to colleagues, management, directors, customers, or external workers. Thus, reporting is 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 an ETL or a machine learning (ML) 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 several 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 5.2. They should also work under subsequent KNIME versions, although there might be slight differences in their appearance.

1.2. The KNIME Community

Being an open-source software, KNIME Analytics Platform benefits a number of forums and groups of KNIME users all around the world. This is a good safety net for advice, hints, and learning material. We report below the most popular sites and groups.

Useful Web Pages

- *KNIME Website:* The root page on the KNIME website.

  https://www.knime.com

- *Software Overview:* The first place to look for an overview of all KNIME products. The open source KNIME Analytics Platform can be downloaded here.

  https://www.knime.com/software-overview

- *Learning Hub:* A central spot to access education material to get you started with KNIME.

  https://www.knime.com/learning

- *KNIME Community Hub:* The perfect place to search for nodes or example workflows when you're not quite sure what you need yet.

  https://hub.knime.com

- *KNIME Forum:* Come here to engage in community discussion, submit feature requests, ask for help, or help others yourself!

  https://forum.knime.com

- *Events and Courses:* Information on all our upcoming events including courses, webinars, learnathons, and summits.

  https://www.knime.com/events

- *Blogs:* A collection of blog posts covering data science with KNIME, a great space to learn what KNIME can really do.

  https://www.knime.com/blog

- *FAQ:* A collection of some of our most commonly asked questions, check out the forum if your answer isn't here!

  https://www.knime.com/faq

- *KNIME Press:* Information on all our available books, like this one!

  https://www.knime.com/knimepress

Courses, Events, and Videos

- *Courses for KNIME Analytic Platform:* KNIME periodically offers onsite and online courses for the KNIME software. This includes basic and advanced elements. To check for the next available date and to register, just go to the KNIME Events web page (https://www.knime.com/events) and select the tab "Online Course".

- *KNIME Webinars:* A number of webinars are also frequently available on specific topics, like chemistry nodes, text mining, integration with other analytics tools, automated machine learning, deep learning, time series analysis, best practices, and so on. To know about the next scheduled webinars, check the KNIME Events web page (https://www.knime.com/events) and select the tab "Webinars".
- *KNIME Data Connects, KNIME Data Talks, and KNIME Summits:* KNIME Data Connects, KNIME Data Talks and KNIME Summits are held periodically all over the world. These are always good chances to learn more about the KNIME software, to get inspired about new data science projects, and to get to know other people from the KNIME Community. To check for the next upcoming events, just go to the KNIME Events web page (https://www.knime.com/events) and select the tabs "Data Talks", "Summit", or "Data Connect".
- *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 specially to get updated about the new features in the new KNIME releases (http://www.youtube.com/user/KNIMETV).

Books

- *Advanced Features in KNIME Analytics Platform:*

  For the advanced use: Rosaria Silipo & Victor Palacios, "KNIME Advanced Luck", KNIME Press

  https://www.knime.com/knimepress/advanced-luck

- *Data Science and KNIME*:

  For an overview of data science, data mining, and data analytics, please check: Berthold, M.R., Borgelt, C., Höppner, F., Klawonn, F., Silipo, R., "Guide to Intelligence Data Science", Springer 2020

  https://www.datascienceguide.org/

- *Codeless Deep Learning with KNIME*:

  It is possible to implement deep learning solutions also within KNIME Analytics Platform:

  Kathrin Melcher & Rosaria Silipo, "Codeless Deep Learning with KNIME", Packt, 2020

- *Codeless Time Series Analysis with KNIME*:

  A book explaining the main steps for time series analysis using the KNIME time series components.

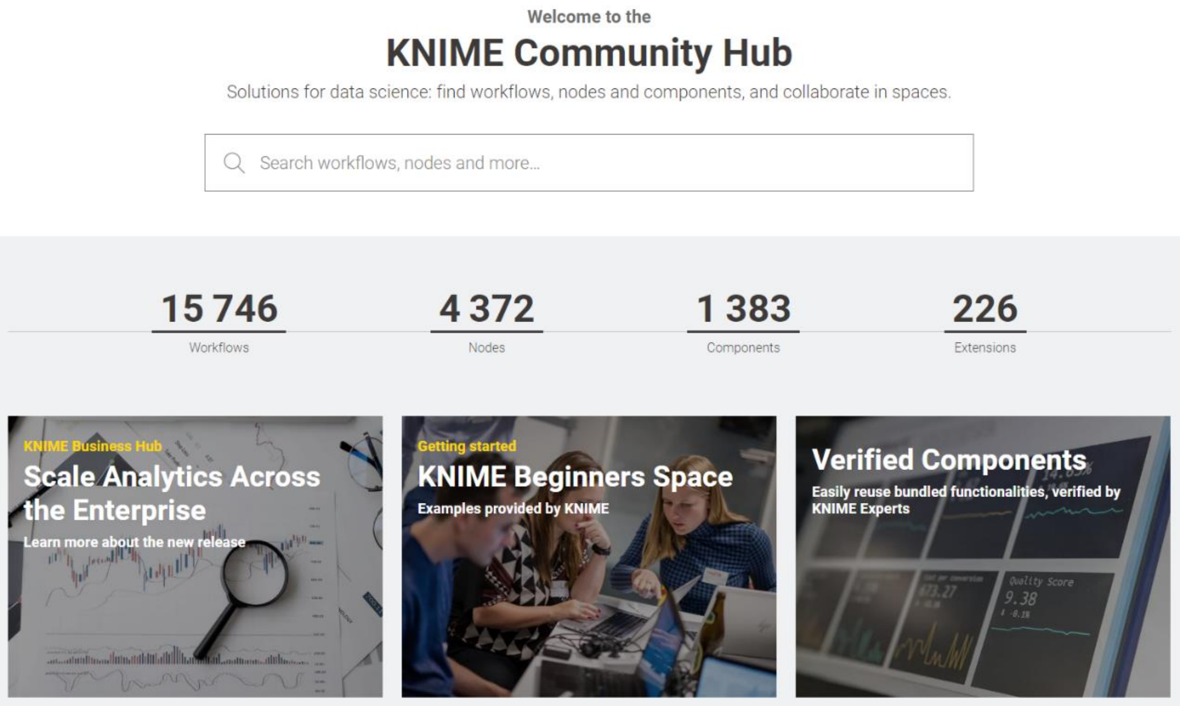
  Corey Weisinger, Maarit Widmann, Daniele Tonini, "Codeless Time Series Analysis with KNIME", Packt, 2022

KNIME Community Hub

However, there is a privileged place where to find information about KNIME nodes and example workflows for your next projects: the KNIME Community Hub (https://hub.knime.com/).

The KNIME Community Hub is a repository of applications, components, and nodes to recycle, reuse, and assemble on KNIME Analytics Platform. Or as it says on the home page: The KNIME Community Hub is "the place to find and collaborate on KNIME workflows and nodes. Here you can find solutions for your data science questions."

When you access the KNIME Community Hub the first time, you end up with the page in Figure 1.1. This page offers a few links to the starting guide documentation, the KNIME Forum, and



Welcome to the

KNIME Community Hub

Solutions for data science: find workflows, nodes and components, and collaborate in spaces.

Search workflows, nodes and more...

15 746 Workflows

4 372 Nodes

1 383 Components

226 Extensions

KNIME Business Hub

Scale Analytics Across the Enterprise

Learn more about the new release

Getting started

KNIME Beginners Space

Examples provided by KNIME

Verified Components

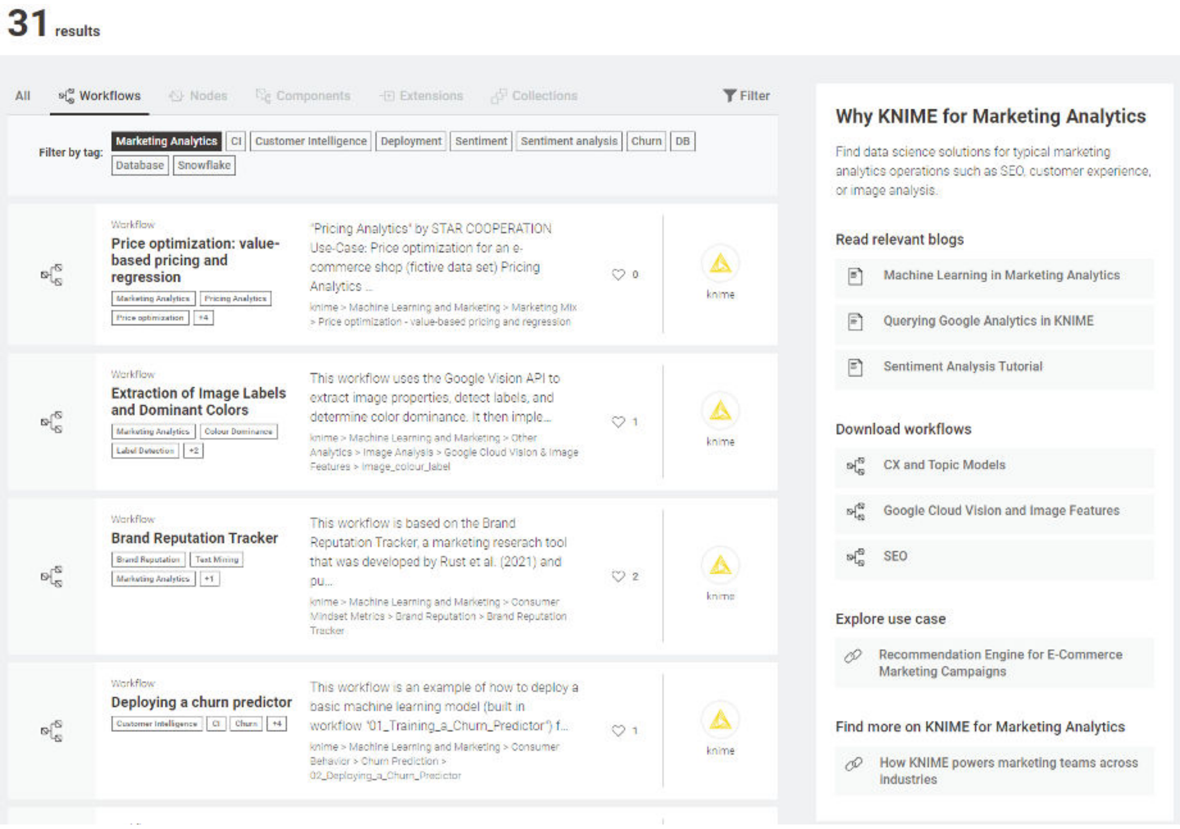
Easily reuse bundled functionalities, verified by KNIME Experts

Figure 1.1. The KNIME Community Hub home page at https://hub.knime.com.

the KNIME blog. Most importantly at the top it offers a search box to find applications, nodes, and components uploaded by KNIME users in this shared place of the KNIME community.

If we type "Customer Intelligence" in the search box, we end up with a list of nodes and workflows related to Customer Intelligence. If you then select just "Workflows" in the top menu, you will see a list of applications (workflows) implementing some aspects of Customer Intelligence - and appropriately tagged - as uploaded by users of the KNIME community. Indeed, you can upload your own developed applications on the KNIME Community Hub. All you need is an account with the KNIME Forum.

On the KNIME Community Hub you can also find official dedicated spaces with their own description and landing page. Try for example to type "Marketing" and select the tag "Marketing Analytics". You will get 31 marketing analytics related solutions hosted in the marketing analytics space and described on the landing page on the right, including relevant links.



Workflow
Price optimization: value-based pricing and regression
Marketing Analytics | Pricing Analytics | Price optimization | +4
"Pricing Analytics" by STAR COOPERATION Use-Case: Price optimization for an e-commerce shop (fictive data set) Pricing Analytics ...
knime > Machine Learning and Marketing > Marketing Mix > Price optimization - value-based pricing and regression
♡ 0

Workflow
Extraction of Image Labels and Dominant Colors
Marketing Analytics | Colour Dominance | Label Detection | +2
This workflow uses the Google Vision API to extract image properties, detect labels, and determine color dominance. It then imple...
knime > Machine Learning and Marketing > Other Analytics > Image Analysis > Google Cloud Vision & Image Features > image_colour_label
♡ 1

Workflow
Brand Reputation Tracker
Brand Reputation | Text Mining | Marketing Analytics | +1
This workflow is based on the Brand Reputation Tracker, a marketing reserach tool that was developed by Rust et al. (2021) and pu...
knime > Machine Learning and Marketing > Consumer Mindset Metrics > Brand Reputation > Brand Reputation Tracker
♡ 2

Workflow
Deploying a churn predictor
Customer Intelligence | CI | Churn | +4
This workflow is an example of how to deploy a basic machine learning model (built in workflow '01_Training_a_Churn_Predictor') f...
knime > Machine Learning and Marketing > Consumer Behavior > Churn Prediction > 02_Deploying_a_Churn_Predictor
♡ 1

Why KNIME for Marketing Analytics

Find data science solutions for typical marketing analytics operations such as SEO, customer experience, or image analysis.

Read relevant blogs
- Machine Learning in Marketing Analytics
- Querying Google Analytics in KNIME
- Sentiment Analysis Tutorial

Download workflows
- CX and Topic Models
- Google Cloud Vision and Image Features
- SEO

Explore use case
- Recommendation Engine for E-Commerce Marketing Campaigns

Find more on KNIME for Marketing Analytics
- How KNIME powers marketing teams across industries

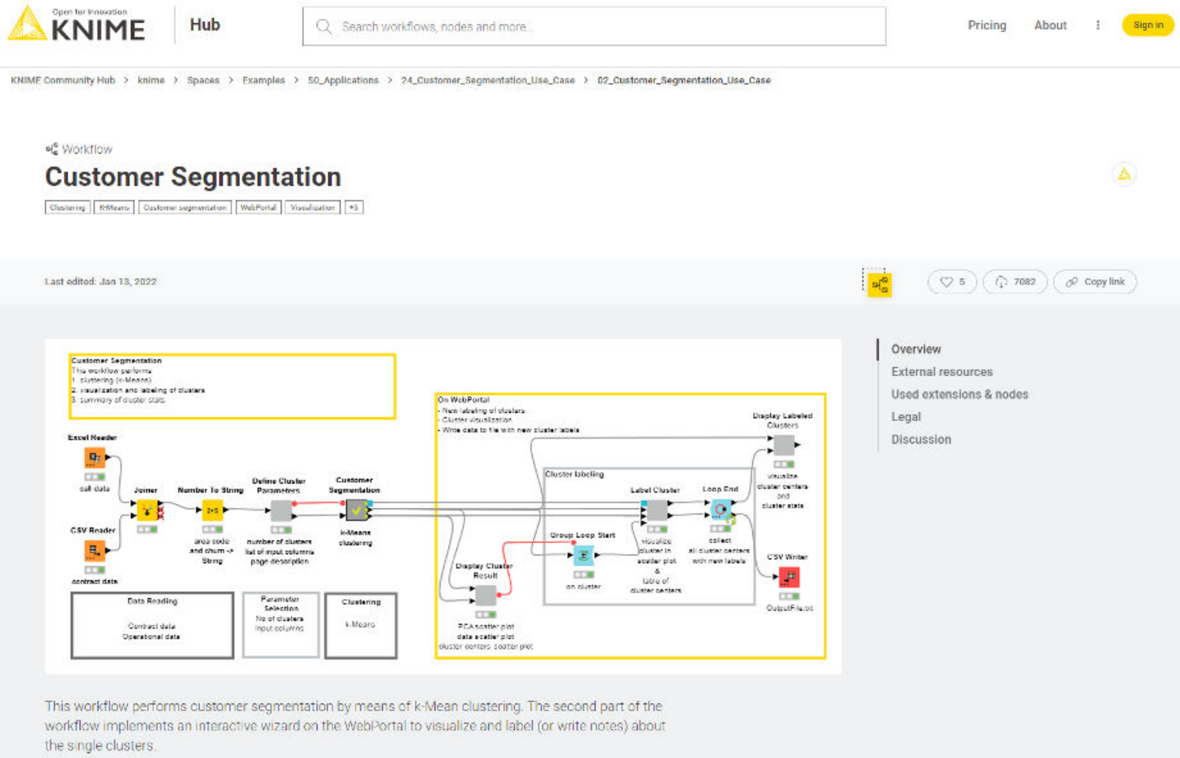
Figure 1.2. The list of applications (workflows) related (and tagged) with "Marketing Analytics" on the KNIME Community Hub.

Clicking one of the applications in the list opens the corresponding page (Figure 1.3), with a nice explanatory picture of the implemented workflow.

On the top right corner, you can see the button to log in with your KNIME account. Being logged in allows you to upload, download, comment, like, and update the spaces and workflows for which you have permissions. Below that, you can find the author picture and below that a number of utility buttons: to download the workflow, to like it, to drag & drop it into KNIME Analytics Platform, and to copy the short permanent link for this workflow to share.

If you hover over the author image, and if you have editing permissions for this Hub space, a pen will appear. Clicking on it will allow you to give other KNIME users permission to upload and change this space.

Notice that the KNIME Community Hub is a repository for workflows, but also for nodes, components, and extensions.



KNIME Community Hub > knime > Spaces > Examples > 50_Applications > 24_Customer_Segmentation_Use_Case > 02_Customer_Segmentation_Use_Case

Customer Segmentation

Last edited: Jan 13, 2022

Customer Segmentation
This workflow performs
1. clustering (k-Means)
2. visualization and labeling of clusters
3. summary of cluster stats

On WebPortal
- New labeling of clusters
- Cluster visualization
- Write data to file with new cluster labels

This workflow performs customer segmentation by means of k-Mean clustering. The second part of the workflow implements an interactive wizard on the WebPortal to visualize and label (or write notes) about the single clusters.

Figure 1.3. The page dedicated to the application named "Customer Segmentation" on the KNIME Community Hub, with short link https://kni.me/w/37cHxqru6dblIUeP.

1.3. Download and Install KNIME Analytics Platform

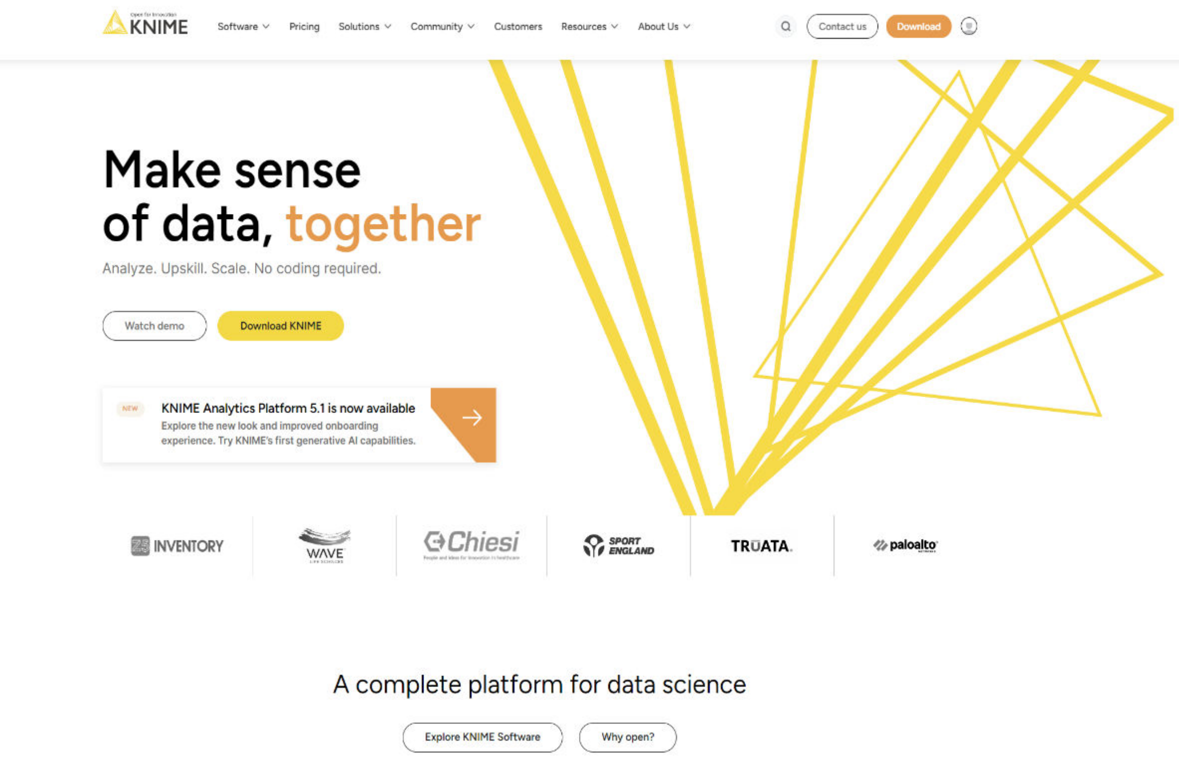
There are two available KNIME products:

- the open source KNIME Analytics Platform, which can be downloaded free of charge at https://www.knime.com/software-overview under the GPL version 3 license
- the KNIME Business Hub, which is described at https://www.knime.com/knime-business-hub

Analytically speaking, the functionalities of the two products are the same. The KNIME Business Hub in addition includes a number of useful IT features for team collaboration, enterprise workflow deployment and management, data warehousing, integration, and scalability for the data science lab. In this book, however, we will work with KNIME Analytics Platform (open source). To start playing with KNIME Analytics Platform, first, you need to download it to your machine.

Download KNIME Analytics Platform

- Go to www.knime.com
- In the upper right corner of the main page, click “Download”
- Provide a little information about yourself (that is appreciated), then proceed to step 2 “Download KNIME”
- 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. If you selected the installer version, just run it and follow the installer instructions.



Make sense of data, together

Analyze. Upskill. Scale. No coding required.

KNIME Analytics Platform 5.1 is now available
Explore the new look and improved onboarding experience. Try KNIME's first generative AI capabilities.

A complete platform for data science

Figure 1.4. The KNIME web page.

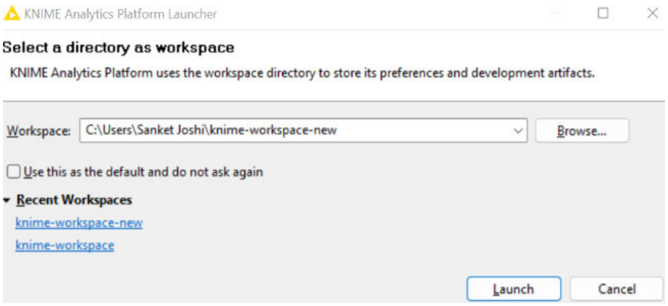
1.4. Workspace

To start KNIME Analytics Platform, 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.

If you are starting KNIME Analytics Platform for the first time, you will be asked if you want to share your usage statistics with KNIME. These numbers will be used to fuel the best practice recommendation engine provided within KNIME Analytics Platform workbench: the Workflow Coach. No personal information will ever reach KNIME and your anonymous statistics will never be shared with anybody.

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

The Workspace Launcher



Select a directory as workspace

KNIME Analytics Platform uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\Sanket Joshi\knime-workspace-new

□ Use this as the default and do not ask again

Figure 1.5. The "Workspace Launcher" window.

The *workspace* is a folder where all preferences and applications (workflows), both developed and currently under development, 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.

Once KNIME Analytics Platform 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 Analytics Platform restarts, showing the workflow list from the newly selected workspace. Notice that if the workspace folder does not exist, it will be automatically created.

When having a large number of customers for example, different workspaces can be used for each one of them. This keeps each workspace clean and tidy and protects from mixing up information by mistake.

For this project I used the workspace "Knime-workspace-new".

1.5. KNIME Workflow

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

Small little icons, 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 Analytics Platform 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)."

On the right 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

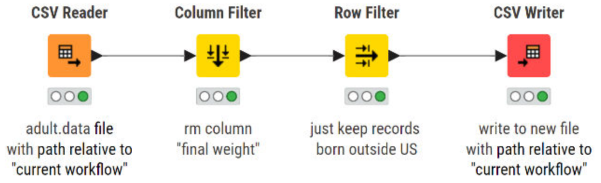

CSV Reader
Column Filter
Row Filter
CSV Writer
adult.data file with path relative to "current workflow"
rm column "final weight"
just keep records born outside US
write to new file with path relative to "current workflow"

Figure 1.6. 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 Analytics Platform 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.

[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).*

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* or *components*.

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

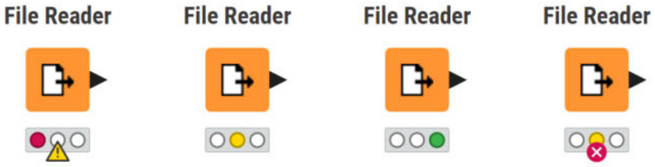
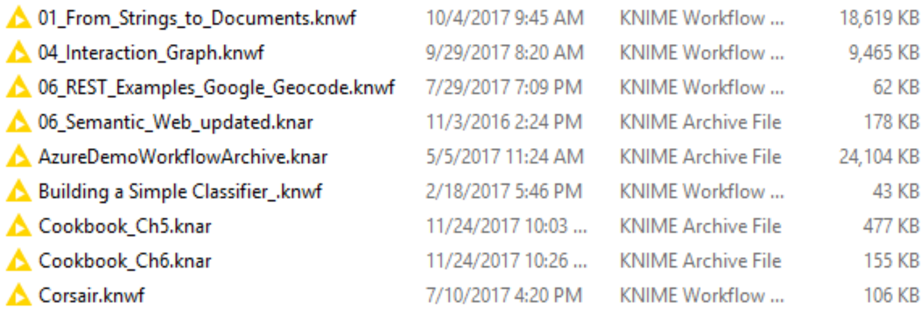


Figure 1.7. The File Reader node in different states.

1.6. File Extensions: *.knwf* and *.knar*

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 KNIME Analytics Platform. A double-click opens the workflow inside KNIME Analytics Platform.



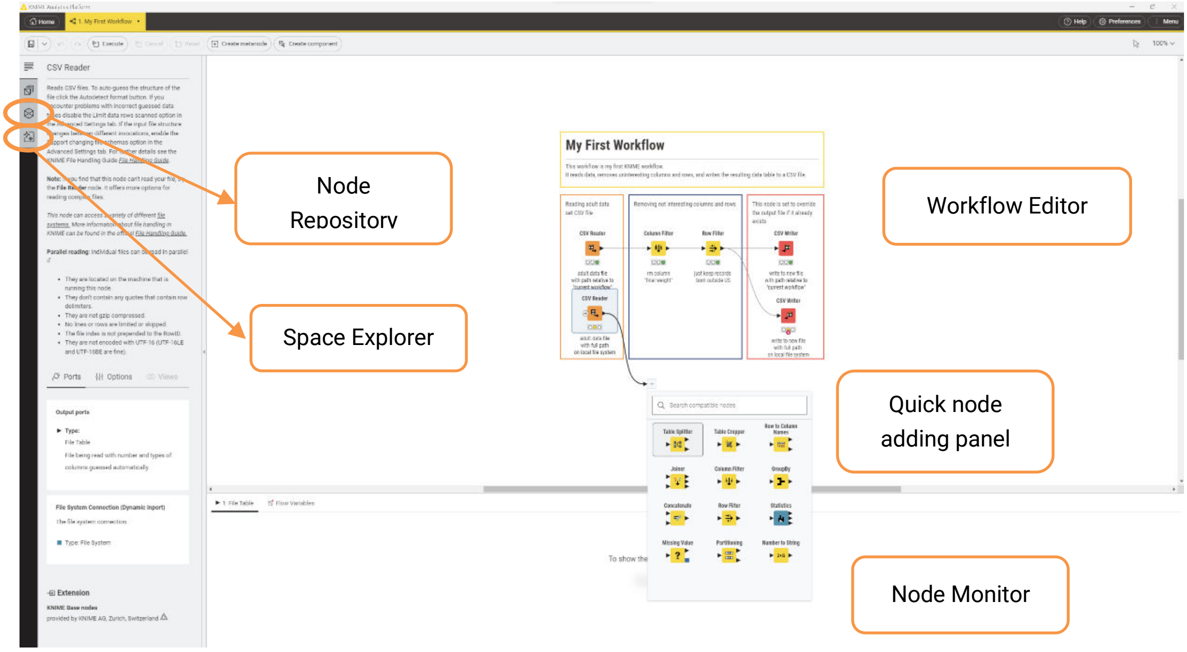
01_From_Strings_to_Documents.knwf	10/4/2017 9:45 AM	KNIME Workflow ...	18,619 KB
04_Interaction_Graph.knwf	9/29/2017 8:20 AM	KNIME Workflow ...	9,465 KB
06_REST_Examples_Google_Geocode.knwf	7/29/2017 7:09 PM	KNIME Workflow ...	62 KB
06_Semantic_Web_updated.knar	11/3/2016 2:24 PM	KNIME Archive File	178 KB
AzureDemoWorkflowArchive.knar	5/5/2017 11:24 AM	KNIME Archive File	24,104 KB
Building a Simple Classifier_.knwf	2/18/2017 5:46 PM	KNIME Workflow ...	43 KB
Cookbook_Ch5.knar	11/24/2017 10:03 ...	KNIME Archive File	477 KB
Cookbook_Ch6.knar	11/24/2017 10:26 ...	KNIME Archive File	155 KB
Corsair.knwf	7/10/2017 4:20 PM	KNIME Workflow ...	106 KB

Figure 1.8. .knwf and .knar files are associated with KNIME Analytics Platform. A double-click opens the workflow(s) directly inside the platform.

1.7. KNIME User Interface

After accepting the workspace path, the KNIME UI opens on a “Getting started with KNIME Analytics Platform 5” entry page. This page provides access to the local space or create a new workflow. This page also provides the mount points. You can connect to them for usage by clicking the sign-in button. The “KNIME User Interface” consists of a top menu, a tool bar, and a few panels. Panels can be closed, re-opened.

- ***Home button:*** This panel shows the list of workflow projects available in the selected workspace (LOCAL), on the EXAMPLES, on the My-KNIME-Hub (your own space on the KNIME Community Hub), or on other connected KNIME Hub spaces.
- ***Quick node adding panel:*** This is a node recommendation engine. It will provide the list of the compatible nodes to follow the currently selected node. You will also be able to search for the other available nodes by typing in keywords in the search field. Double-click the node you want to add to the workflow, and it will be added and connected automatically.
- ***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.



CSV Reader

Reads CSV files. To auto-guess the structure of the file click the Autodetect format button. If you encounter problems with incorrect guessed data types disable the Limit data rows scanned option in the Advanced Settings tab. If the input file structure changes between different invocations, enable the Support changing file schemas option in the Advanced Settings tab. For further details see the KNIME File Handling Guide File Handling Guide.

Note: If you find that this node can't read your file, try the **File Reader** node. It offers more options for reading complex files.

This node can access a variety of different file systems. More information about file handling in KNIME can be found in the official File Handling Guide.

Parallel reading: Individual files can be read in parallel if:

- They are located on the machine that is running this node.
- They don't contain any quotes that contain row delimiters.
- They are not gzip compressed.
- No lines or rows are limited or skipped.
- The file index is not prepended to the RowID.
- They are not encoded with UTF-16 (UTF-16LE and UTF-16BE are fine).

Figure 1.9. The KNIME user Interface.

- ***Workflow Editor:*** Workflow Editor is the central part of the KNIME UI. 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.
- ***Space Explorer:*** To navigate to local or KNIME Hub spaces and access the workflows, components, and files, you can switch to the "Space Explorer" panel.
- ***Node Description:*** If a node or a workflow is selected, the "Node Description" panel on the left displays a summary description of the node's functionalities or the workflow's meta information.

1.8. Help

The "Help" button is one of the newer options added to the KNIME User Interface in version 5.2.

It includes a few more buttons; if you click on those buttons, you will be redirected to the respective website.

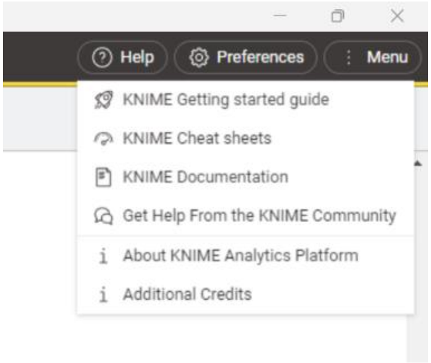

Help
Preferences
Menu
KNIME Getting started guide
KNIME Cheat sheets
KNIME Documentation
Get Help From the KNIME Community
About KNIME Analytics Platform
Additional Credits

Figure 1.10. The "Help" button in KNIME.

1.9. Preferences

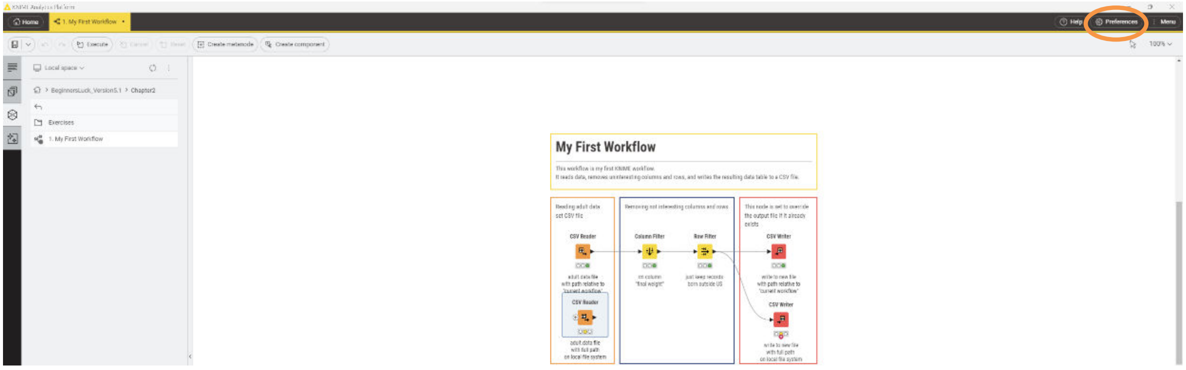
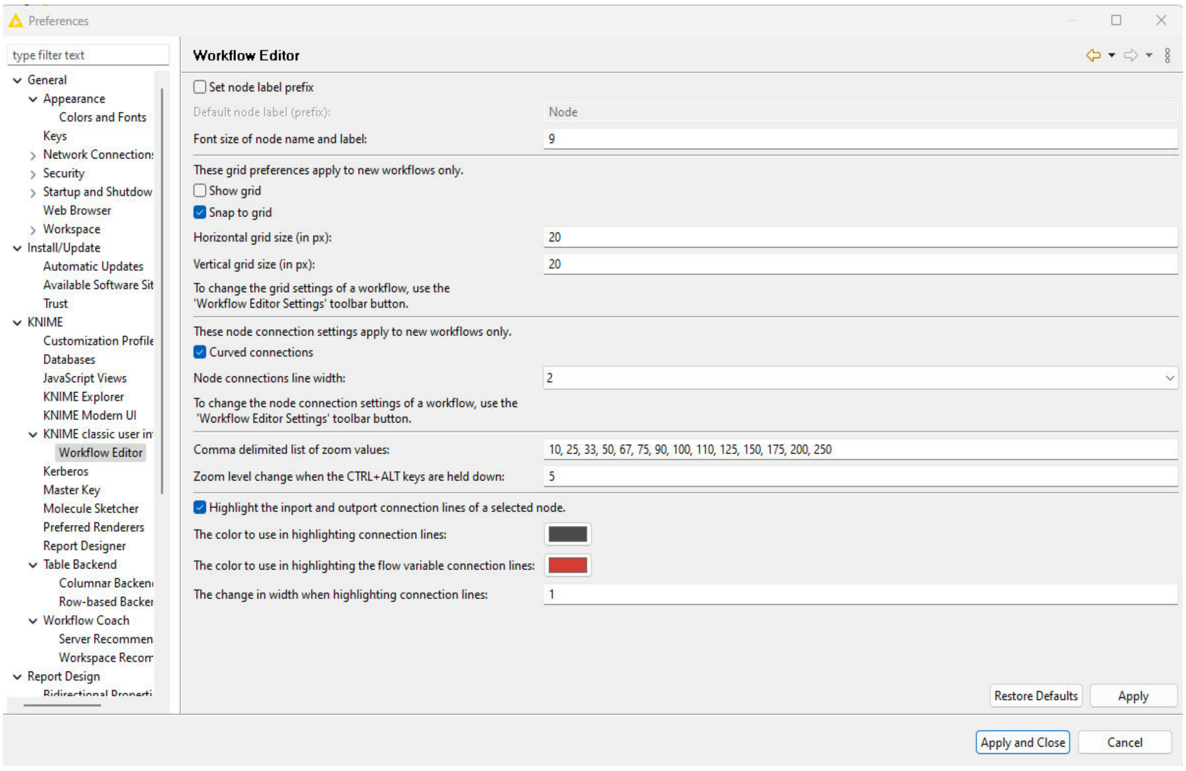


Figure 1.11. The "Preferences" option in the KNIME User Interface.

Preferences brings you to the window where all KNIME settings can be customized. Under the item "KNIME" can be found:

- ***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.
- ***Space Explorer*** – contains the list of the shared repositories via KNIME Hub spaces.
- ***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 Hub workspace, or your own local workspace.



Workflow Editor

□ Set node label prefix

Default node label (prefix): Node

Font size of node name and label: 9

These grid preferences apply to new workflows only.

□ Show grid

☑ Snap to grid

Horizontal grid size (in px): 20

Vertical grid size (in px): 20

To change the grid settings of a workflow, use the 'Workflow Editor Settings' toolbar button.

These node connection settings apply to new workflows only.

☑ Curved connections

Node connections line width: 2

To change the node connection settings of a workflow, use the 'Workflow Editor Settings' toolbar button.

Comma delimited list of zoom values: 10, 25, 33, 50, 67, 75, 90, 100, 110, 125, 150, 175, 200, 250

Zoom level change when the CTRL+ALT keys are held down: 5

☑ Highlight the inport and outport connection lines of a selected node.

The color to use in highlighting connection lines: ■

The color to use in highlighting the flow variable connection lines: 🟥

The change in width when highlighting connection lines: 1

Figure 1.12. The "Preferences" window.

Tool Bar

The tool bar is another important piece of the KNIME User Interface.

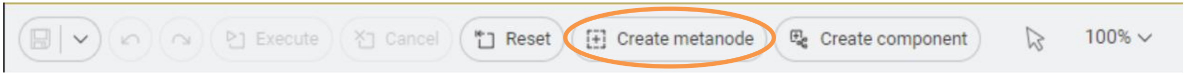


Figure 1.13. The "Create Metanode" button in the tool bar.

From the left, we see

- an icon to *save* or *save as* the current workflow. To the left, we find the icon to save the selected workflow
- *undo* and *redo* the changes that were made to the workflow
- *execute* or *execute all* (which executes all the selected nodes)
- *cancel* the executing node(s)
- *reset* all the node(s) to the original state
- *create Metanode*

- *create Component*,
- the arrow icon which provides the option of selecting, using the annotating mode or using the panning mode
- and on the right is the zoom (in %) level selection.

For now, let's have a look at the ***Create Metanode*** button. The *Create Metanode* button creates a metanode for all the selected nodes. It creates one node inside which you can find all the selected nodes. This is particularly useful to create a clean and easy-to-use workflow, for example, by compressing all the nodes used for data cleaning within a *Pre-processing* metanode.

Hotkeys

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.

Node Configuration

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

Node Execution

- `F7` executes selected configured nodes
- `Shift + F7` executes all configured nodes

Stop Node Execution

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

Node Resetting

- `F8` resets selected nodes

Save Workflows

- `Ctrl + S` saves the workflow
- `Ctrl + Shift + S` saves all open workflows

- `Ctrl + W` closes all open workflows

Metanode

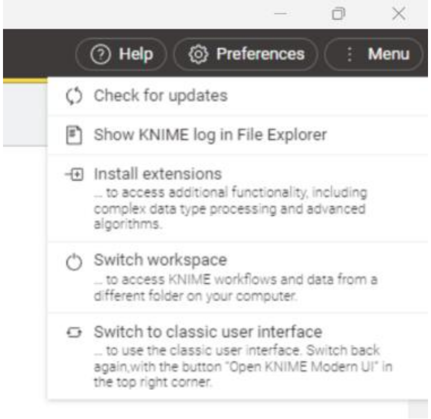
- `Shift + F12` opens Meta Node Wizard

To move Annotations

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

1.10. Menu

The “Menu” option button is another option added to the KNIME User Interface in KNIME Analytics Platform v5.2.



Check for updates

Show KNIME log in File Explorer

Install extensions
... to access additional functionality, including complex data type processing and advanced algorithms.

Switch workspace
... to access KNIME workflows and data from a different folder on your computer.

Switch to classic user interface
... to use the classic user interface. Switch back again,with the button "Open KNIME Modern UI" in the top right corner.

Figure 1.14. The "Menu" button in the tool bar.

Once you click this button, you will see multiple options, and we will see what each option does.

- ***Check for Updates:*** On clicking this option, you can see if any updates are available for your KNIME Analytics Platform.
- ***Show KNIME log in File Explorer:*** This option will open a window in your local system where the KNIME log file is stored.
- ***Install Extensions:*** On clicking this option, you will be able to see a list of extensions that you could install.
- ***Switch Workspace:*** This will allow you to change the working directory.
- ***Switch to Classic User Interface:*** This option will first ask you to save or not save the current workflow and then switch to the classic UI. You can also switch back to the Modern UI from the classic UI by clicking on the “Open KNIME Modern UI” option in the top right corner of the Analytics Platform.

1.11. 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.

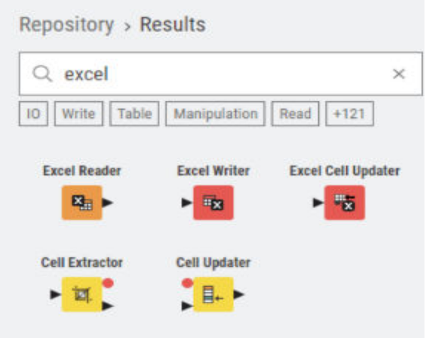


Figure 1.15. Word search in the Node Repository panel: exact match mode.

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.

1.12. Space Explorer

We find the Space Explorer panel below the Home button, one of the four panels on the KNIME User Interface. This panel contains:

- Under LOCAL the workflows that have been developed in the selected workspace
- The mount points to a number of KNIME Hub spaces.
- The workflows contained in the reference workspace of such Hub spaces.
- The access to the My-KNIME-Hub, that is to your space on the KNIME Community Hub. Remember that you need an account with the KNIME Forum to access this space.

At the beginning, the Space Explorer panel only contains LOCAL, My-KNIME-Hub, and EXAMPLES. As we already stated, LOCAL shows the content of the selected workspace. EXAMPLES points to a read-only public repository, accessible via anonymous login. This repository hosts a number of example workflows that you can use to jump start a new project. My-KNIME-Hub allows to access your space on the KNIME Community Hub.

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 “Space Explorer”, containing workflows, are also called “Workflow Groups”.

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

My-KNIME-Hub

From the Space Explorer panel, you can access your spaces on the KNIME Community Hub and upload and update new or existing content in there.

By default, an authenticated KNIME user has a public space, for material to share publicly, and a private space to park his/her own material for further usage. However, new private or public spaces can be created with a right-click on My-KNIME-Hub in the Space Explorer panel and then a selection of the option “Create new Space…”.

By default, you are the only owner of your own spaces. However, when accessing this space from a web browser, after hovering on your image in the top right corner, a pen appears. This will allow us to add colleagues and teammates as contributors to the space.

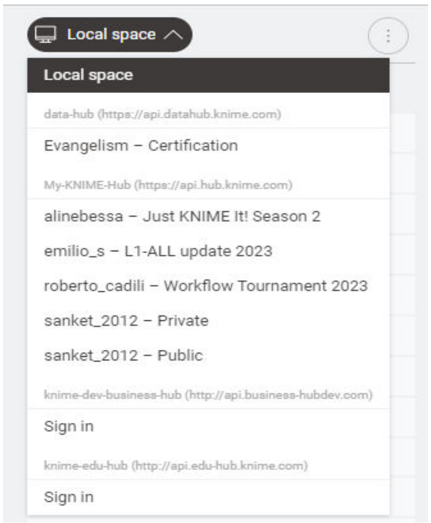

Local space
Local space
data-hub (https://api.datahub.knime.com)
Evangelism – Certification
My-KNIME-Hub (https://api.hub.knime.com)
alinebessa – Just KNIME It! Season 2
emilio_s – L1-ALL update 2023
roberto_cadili – Workflow Tournament 2023
sanket_2012 – Private
sanket_2012 – Public
knime-dev-business-hub (http://api.business-hubdev.com)
Sign in
knime-edu-hub (http://api.edu-hub.knime.com)
Sign in

Figure 1.16. Space Explorer panel. At the top the content of the EXAMPLES; below the content of the LOCAL workspace.

EXAMPLES

A link to EXAMPLES is available in the “Space Explorer” panel. This is a repository 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:

- double click “EXAMPLES” in the “Space Explorer” panel
- double click “Double click to connect…”

You should be automatically logged in as a guest.

To transfer example workflows from the EXAMPLES 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.

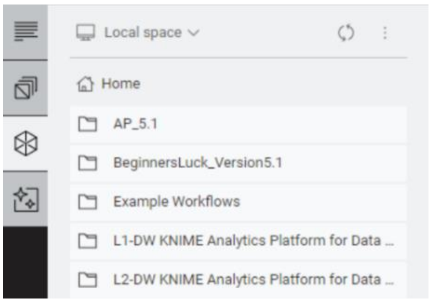

Local space
Home
AP_5.1
BeginnersLuck_Version5.1
Example Workflows
L1-DW KNIME Analytics Platform for Data ...
L2-DW KNIME Analytics Platform for Data ...

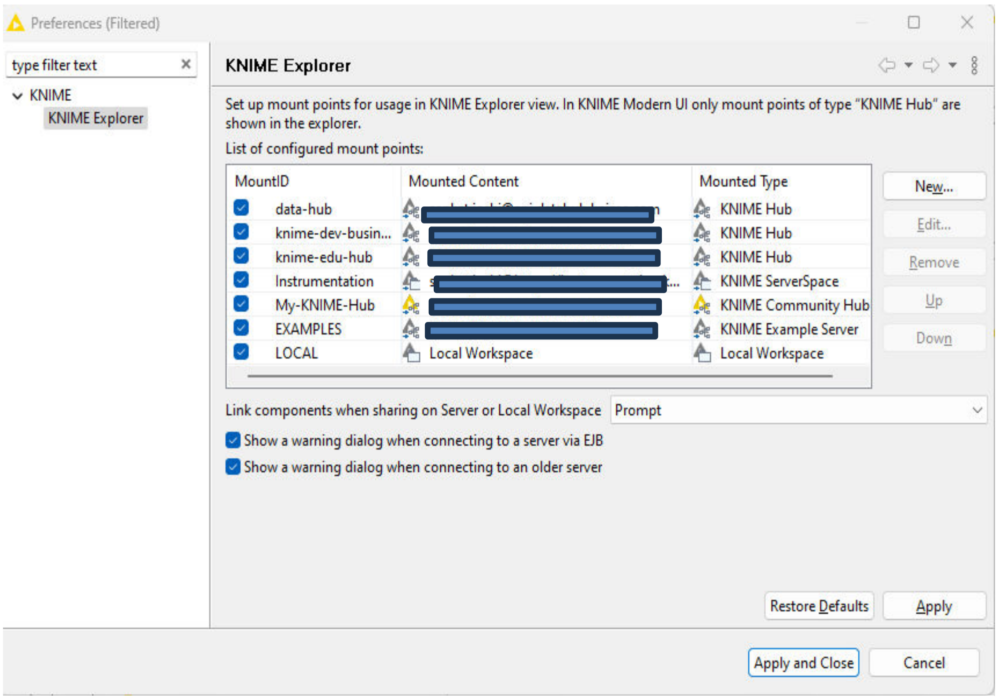
Figure 1.17. Space Explorer panel.

The Space Explorer panel can of course host more than one KNIME Hub space. It is enough to add mountpoints to the list of the available KNIME Hub spaces.

Mounting KNIME Business Hub in Space Explorer

To add KNIME Business Hubs to the "Space Explorer" panel, in the modern UI, click on "Preferences" -> KNIME -> KNIME Explorer and:

- The "Preferences (Filtered)" window opens on the "KNIME Explorer" page and lists all KNIME spaces already mounted in this KNIME instance. The three KNIME spaces available by default on every KNIME instance are the local workspace "LOCAL", the KNIME "EXAMPLES", and the My-KNIME-Hub located on the KNIME Community Hub (hub.knime.com).
- Use the "New" and the "Remove" button to add /remove connections to remote Hub spaces.
- After clicking the "New" button, fill in the required information about the KNIME Business Hub in the "Select New Content" window (Figure 1.18).



KNIME Explorer

Set up mount points for usage in KNIME Explorer view. In KNIME Modern UI only mount points of type "KNIME Hub" are shown in the explorer.

List of configured mount points:

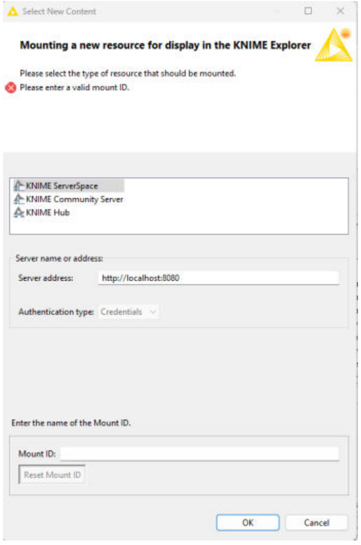
	MountID	Mounted Content	Mounted Type
☑	data-hub	[redacted]	KNIME Hub
☑	knime-dev-busin...	[redacted]	KNIME Hub
☑	knime-edu-hub	[redacted]	KNIME Hub
☑	Instrumentation	[redacted]	KNIME ServerSpace
☑	My-KNIME-Hub	[redacted]	KNIME Community Hub
☑	EXAMPLES	[redacted]	KNIME Example Server
☑	LOCAL	Local Workspace	Local Workspace

Link components when sharing on Server or Local Workspace: Prompt

☑ Show a warning dialog when connecting to a server via EJB

☑ Show a warning dialog when connecting to an older server

Figure 1.18. The "Preferences (Filtered)" window.



Mounting a new resource for display in the KNIME Explorer

Please select the type of resource that should be mounted.

Please enter a valid mount ID.

Server name or address:

Server address: http://localhost:8080

Authentication type: Credentials

Enter the name of the Mount ID.

Mount ID:

Figure 1.19. The "Select New Content" window.

The same KNIME Explorer "Preferences" page can be reached via *File > Preferences > KNIME Explorer.*

To login into any of the available KNIME Business Hubs in the "Space Explorer" panel:

- right-click or double-click the Hub space name
- provide the credentials

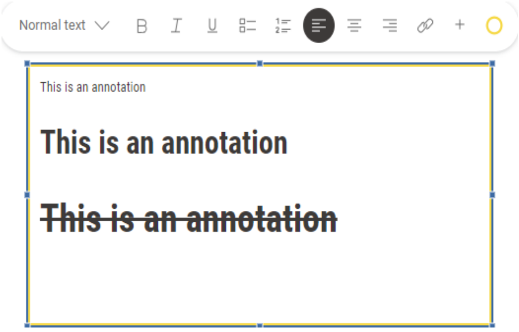
Workflow Editor

The central piece of the KNIME User Interface 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 from the Node Repository or the Quick node adding panel. The workflow building process will be described widely in the next sections of this book. Here, we will describe how to customize and probably improve the canvas role of the workflow editor space. We will describe two options:

- change the canvas appearance with grids and different visualizations for the connections;
- introducing annotations to comment the work.

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 documentation-like overview of the workflow general task and of the single sub-tasks.



This is an annotation

This is an annotation

~~This is an annotation~~

Figure 1.20. The Annotation Editor.

Workflow Annotations

To insert a new annotation:

- right-click anywhere in the workflow editor and select “New Workflow Annotation”
- a gray small frame appears; this is the default annotation frame
- double-click the frame to edit its content
- Notice the tool bar appearing at the top to edit text style, text size, background color, text alignment, and border properties (color, thickness)
- To reopen an annotation, just double-click anywhere on the annotation

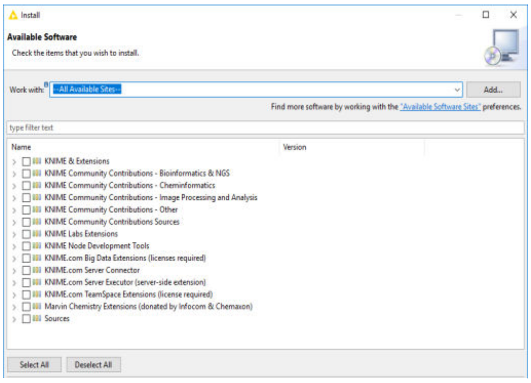
1.13. Download the KNIME Extensions

KNIME Analytics Platform is an open-source product. As every open-source product, it benefits from the feedback and the functionalities that the 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 instance.

Installing KNIME Extensions

To install a new KNIME extension from within KNIME Analytics Platform, there are three options.

1. From the Top right options, select Menu → "Install KNIME Extensions", select the desired extension, click the "Next" button and follow the wizard instructions.
2. IF you want install extension from the classic UI, 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 Analytics Platform 5.x Update Site" - `http://update.knime.com/analytics-platform/4.x`). Then select the extension, click the "Next" button and follow the wizard instructions.
3. Search the KNIME Community Hub, on a web browser or from the KNIME Community Hub panel. When the desired extension is found, drag and drop the extension icon from the browser to the Workflow Editor.



Install

Available Software

Check the items that you wish to install.

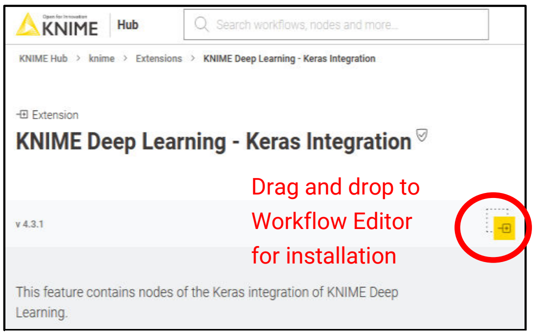
Work with: --All Available Sites--

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
□ KNIME & Extensions	
□ KNIME Community Contributions - Bioinformatics & NGS	
□ KNIME Community Contributions - Cheminformatics	
□ KNIME Community Contributions - Image Processing and Analysis	
□ KNIME Community Contributions - Other	
□ KNIME Community Contributions Sources	
□ KNIME Labs Extensions	
□ KNIME Node Development Tools	
□ KNIME.com Big Data Extensions (licenses required)	
□ KNIME.com Server Connector	
□ KNIME.com Server Executor (server-side extension)	
□ KNIME.com TeamSpace Extensions (license required)	
□ Marvin Chemistry Extensions (donated by Infocom & Chemaxon)	
□ Sources	

Figure 1.21. The "Available Software" window.



KNIME
Hub
Search workflows, nodes and more...
KNIME Hub > knime > Extensions > KNIME Deep Learning - Keras Integration
Extension

KNIME Deep Learning - Keras Integration

v 4.3.1

Drag and drop to Workflow Editor for installation

This feature contains nodes of the Keras integration of KNIME Deep Learning.

Figure 1.22. An extension from the KNIME Community Hub.

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 "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 for the current release; "KNIME Labs Extensions" contains a number of extensions 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. These are usually grouped under "Community" categories. My advice is to install all extensions, even the cheminformatics ones. Many of them contain several useful nodes not necessarily restricted to a particular domain.

1.14. Data and Workflows for this Book

This book builds a few examples and provides the solutions to the exercises. The workflows are accessible via the KNIME Community Hub and are stored in the KNIME Press space – look for the respective book and KNIME version. To download material from the KNIME Community Hub, you need to be logged in with your KNIME account (see how to create a KNIME account). After entering the KNIME Community Hub, in order to download the workflows, just click on the cloud icon. Download the whole folder onto your machine from the KNIME Beginner's Luck space, which will result in a *.knar* file. Then double click it OR import it into the KNIME Explorer via Select "File" → "Import KNIME Workflow…".

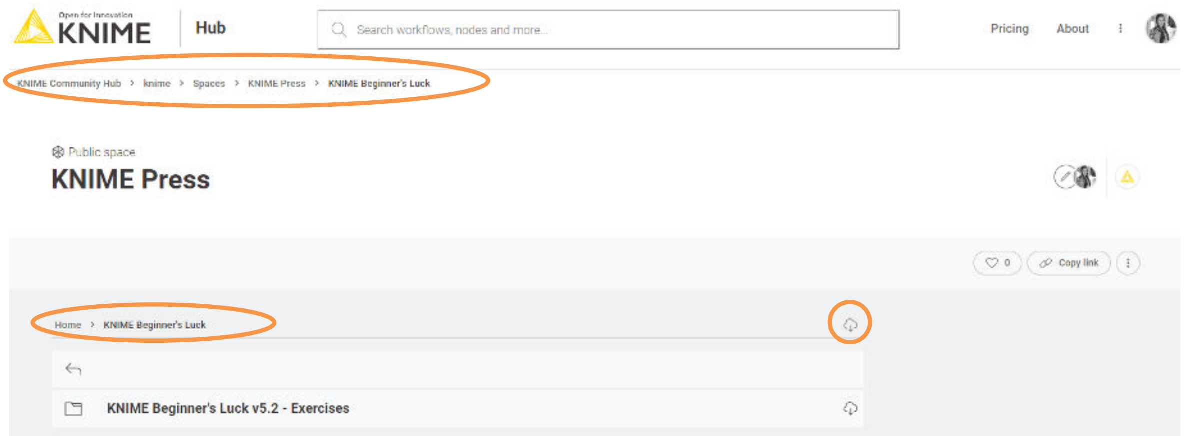

Open for Innovation
KNIME
Hub
Search workflows, nodes and more...
Pricing
About
KNIME Community Hub > knime > Spaces > KNIME Press > KNIME Beginner's Luck
Public space
KNIME Press
0
Copy link
Home > KNIME Beginner's Luck
KNIME Beginner's Luck v5.2 - Exercises

Figure 1.23. Beginners Luck space on the KNIME Community Hub.

At the end of the import operation, in the Space Explorer panel you should find a *KNIME Beginner's Luck v5.2 - Exercises* folder containing Chapter2, Chapter3, Chapter4, Chapter5, Chapter6, and Chapter7 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 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

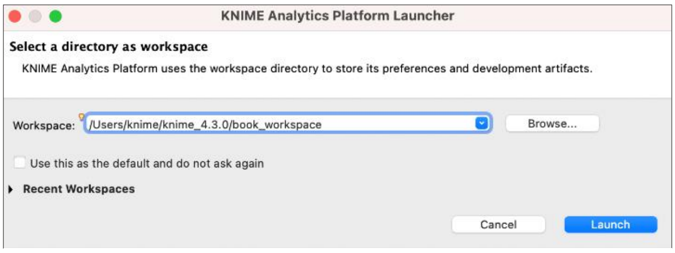
1.15. Exercises

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 "Launch"



KNIME Analytics Platform Launcher

Select a directory as workspace

KNIME Analytics Platform uses the workspace directory to store its preferences and development artifacts.

Workspace: /Users/knime/knime_4.3.0/book_workspace

□ Use this as the default and do not ask again

Figure 1.24. Exercise 1: Create workspace "book_workspace".

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

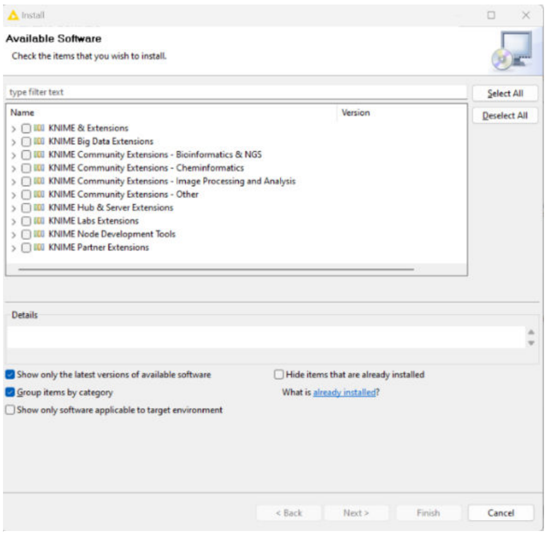
Exercise 2

Install the following extensions:

- KNIME Database
- KNIME Javascript Views
- KNIME Report Designer

Solution to Exercise 2

From the Top right corner options, select Menu → "Install Extensions". Search and select the required Extensions. Click "Next" and follow the instructions.



Install

Available Software

Check the items that you wish to install.

type filter text

Name	Version
› □ KNIME & Extensions	
› □ KNIME Big Data Extensions	
› □ KNIME Community Extensions - Bioinformatics & NGS	
› □ KNIME Community Extensions - Cheminformatics	
› □ KNIME Community Extensions - Image Processing and Analysis	
› □ KNIME Community Extensions - Other	
› □ KNIME Hub & Server Extensions	
› □ KNIME Labs Extensions	
› □ KNIME Node Development Tools	
› □ KNIME Partner Extensions	

Details

☑ Show only the latest versions of available software

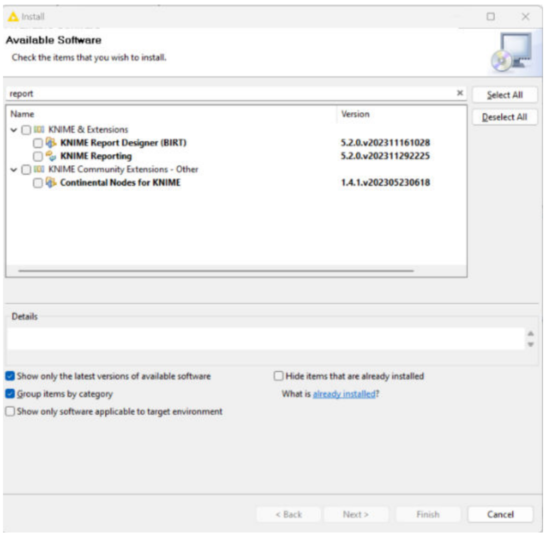
☑ Group items by category

□ Show only software applicable to target environment

□ Hide items that are already installed

What is already installed?

Figure 1.25. Exercise 2: List of KNIME Extensions.



Install

Available Software

Check the items that you wish to install.

report

Name	Version
⌄ □ KNIME & Extensions	
□ **KNIME Report Designer (BIRT)**	5.2.0.v202311161028
□ **KNIME Reporting**	5.2.0.v202311292225
⌄ □ KNIME Community Extensions - Other	
□ **Continental Nodes for KNIME**	1.4.1.v202305230618

Details

☑ Show only the latest versions of available software

☑ Group items by category

□ Show only software applicable to target environment

□ Hide items that are already installed

What is already installed?

Figure 1.26. 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 bottom port, is taken as the reference table; the second input table, connected to the top 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 “Space Explorer” panel on the top left corner of the KNIME User Interface 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 User Interface.

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 “Space 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

If you click on the Home Button:

- Click on the LOCAL workspace
- Select “Create Folder”

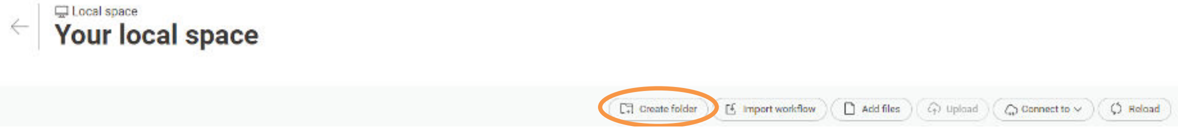


Figure 2.1. Create new workflow group.

In the “Create Folder” dialog:

- Enter the name of the workflow group

> ***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 under the selected workflow group.

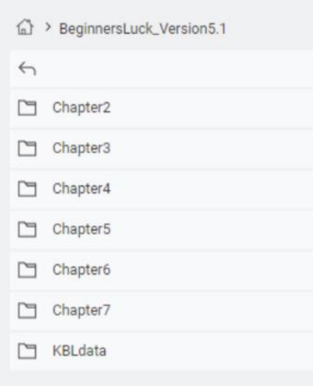


Figure 2.2. Create a new workflow group named for every chapter.

Create a New Workflow

In the “Space Explorer” panel:

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

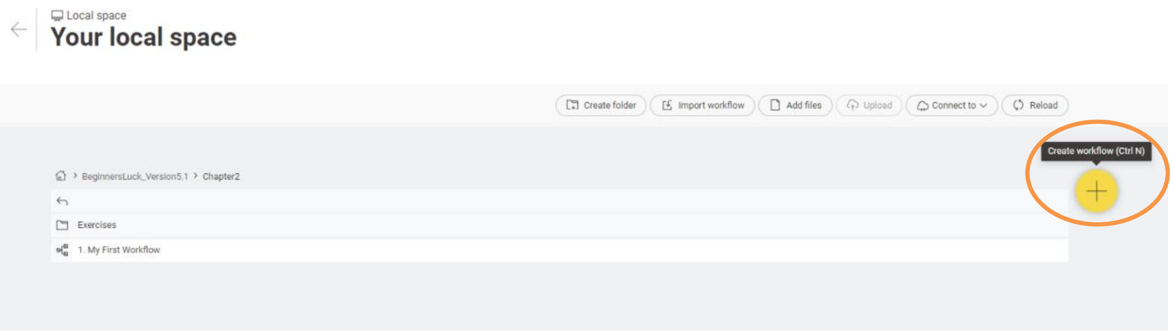


Figure 2.3. Create new workflow.

In the "New KNIME Workflow Wizard" dialog

- Enter the name of the new workflow
- Click "Create"

> ***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 in the selected workflow group.

To create a new workflow from the Space Explorer, click on the black button with three dots, and a window will pop up with multiple options like Create workflow, Create folder, Import workflow, and Connect to Hub.

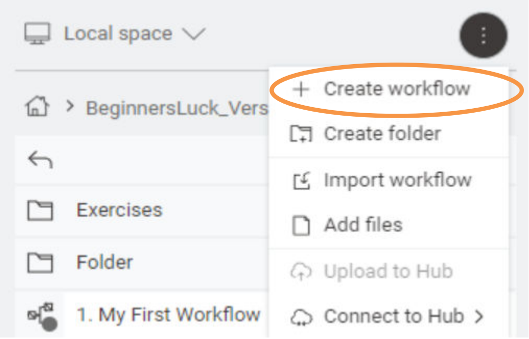


Figure 2.4. Create a new workflow in Space Explorer named "My First Workflow" under "Chapter2".

Save a Workflow

To save a workflow, click the disk icon in the toolbar. This either only saves the selected workflow open in the workflow editor or you can ***Save as*** the workflow with a different name as well. Saving the workflow saves the workflow architecture, the nodes' settings, and the data produced at the output of each node.

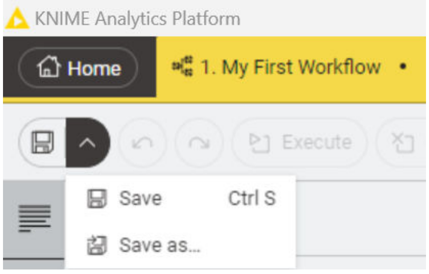


Figure 2.5. "Save" options to save workflow.

Delete a Workflow

To delete a workflow

- Right-click the workflow in the “Space 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.

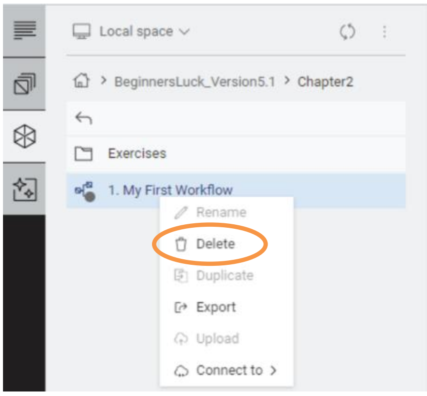


Figure 2.6. Delete a workflow.

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 side panel of the KNIME User Interface. And we have seen that every node has four states: not yet configured (red), configured (yellow), successfully executed (green), and executed with error (red with cross).

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.
- In the Modern UI, clicking the node's output port gives the + in a dotted square symbol and shows all the compatible nodes to the current node. If you click one of the nodes from the panel, a connection is established between the two nodes.

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 configure 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.

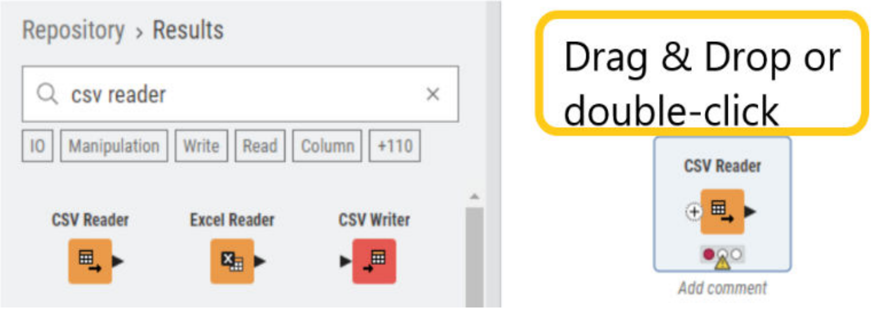


Figure 2.7.Drag and drop or double-click the node to create a new node in the workflow editor.

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.

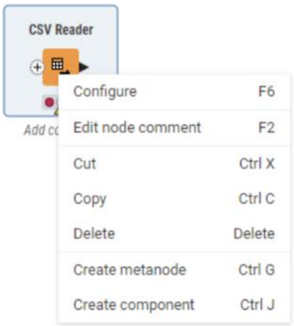


Figure 2.8. Right-click the node and select "Configure" or double-click the node to configure.

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.

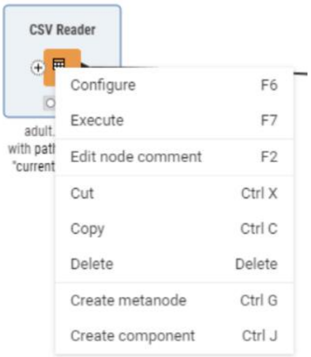


Figure 2.9. Right-click the node and select "Execute" to run the node

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

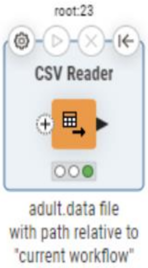


Figure 2.10. Double-click the node name to edit it.

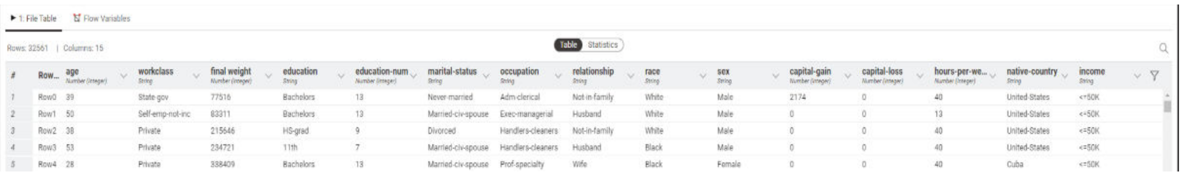
View the Processed Data

If the execution was successful (green light), you can inspect the processed data. You can either right-click on the node and select the open output port and in that, you can select the table which will open a new window with the output table or you will find the output table below the "workflow editor.

The output table shows the processed output and statistics of all the columns in the “Statistics” tab.

Some nodes might produce more than one output data set.

There is a tab beside the “Output Table” tab “Flow Variable”, which lists the flow variables. The concept of “Flow Variable” is explained in detail in the book KNIME Advanced Luck.



Rows: 32561 | Columns: 15

#	Row...	age Number (integer)	workclass String	final weight Number (integer)	education String	education-num Number (integer)	marital-status String	occupation String	relationship String	race String	sex String	capital-gain Number (integer)	capital-loss Number (integer)	hours-per-we... Number (integer)	native-country String	income String
1	Row0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
2	Row1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
3	Row2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
4	Row3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
5	Row4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

Figure 2.11. 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”.

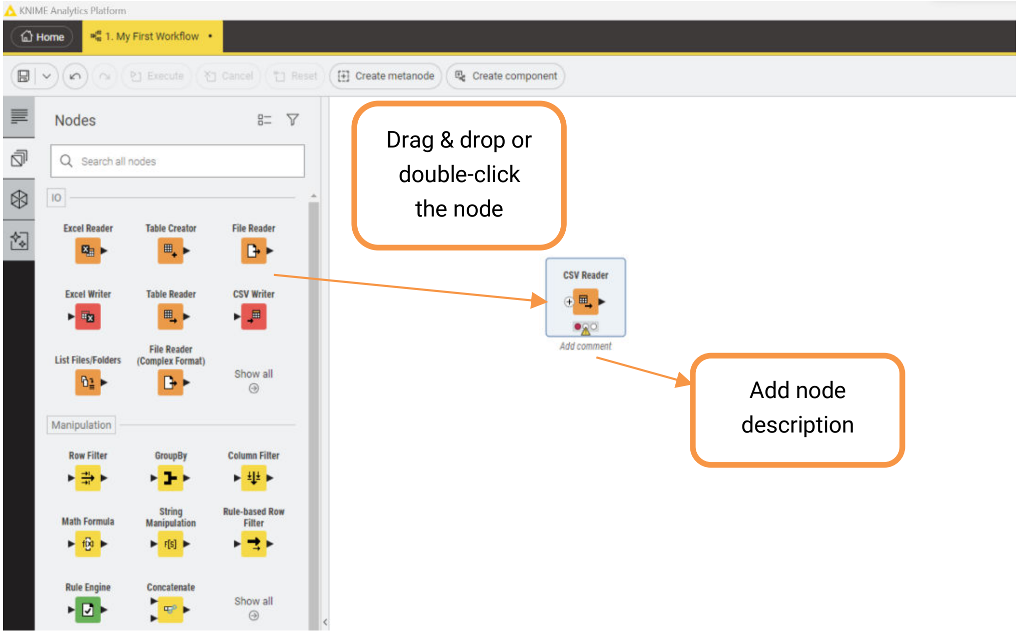
Most common file format is the Comma Separated Values (CSV) format, that covers all text files where fields are separated by a special character. To read this kind of files of data, the CSV Reader node is the most versatile node in KNIME Analytics Platform.

Create a *CSV 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 “CSV Reader” in the search box
- Drag and drop the “CSV Reader” node into the workflow editor (or double-click it)
- If the “Description” panel on the right is enabled, it shows all you need to know about the “CSV Reader” node: task, output port, and required settings.
- To activate the “Description” panel, go to the Top Menu, open “View” and select “Description”.

Note. Under the newly created "CSV 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 CSV 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.



Drag & drop or double-click the node

Add node description

Figure 2.12. Create a CSV Reader node.

Configure the *CSV Reader* Node

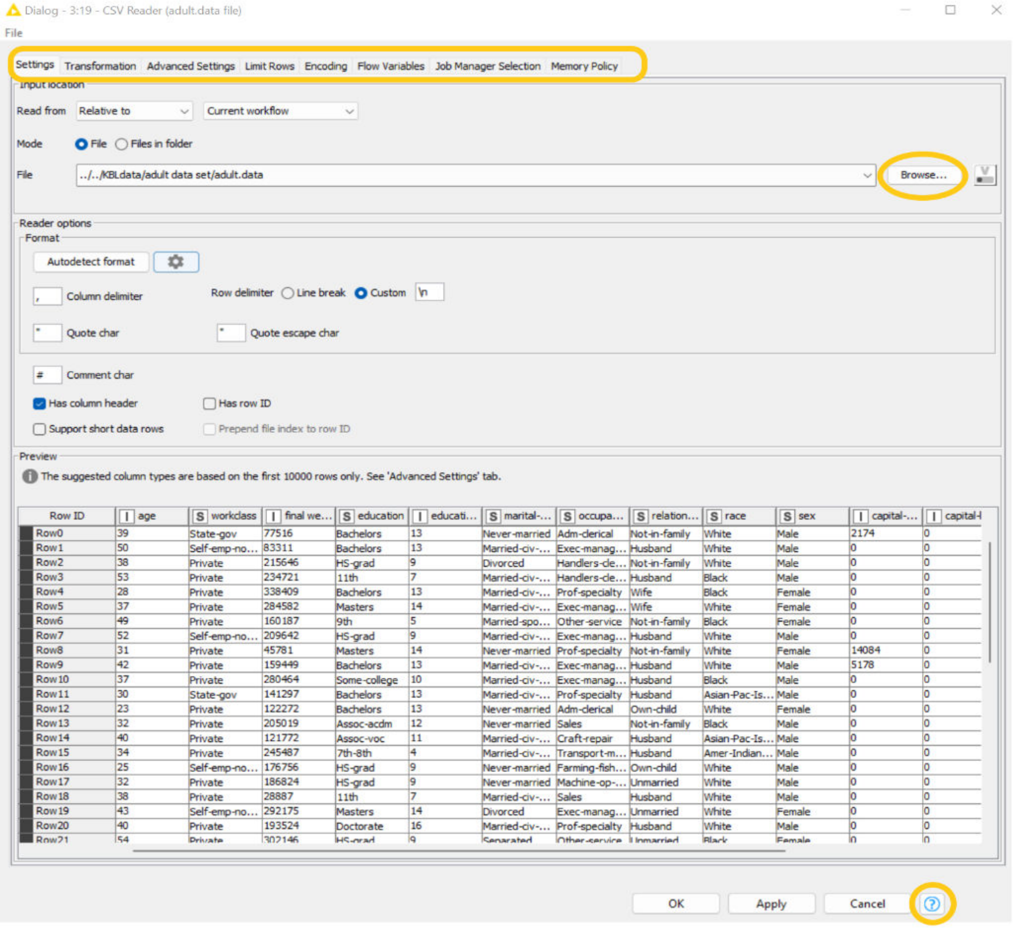
To configure the *CSV Reader* node,

- 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.csv file, downloadable from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Adult) or available in KBLdata/adult data set/adult.csv.
- In most cases, the "CSV Reader" node automatically detects the file structure.

- If this is not one of most cases and the CSV Reader node has not guessed the file structure exactly, then tray with the button "Autodetect format" and/or enable/disable all required checkboxes in the tabs at the top, 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.



Read from: Relative to | Current workflow

Mode: ● File ○ Files in folder

File: ../../KBLdata/adult data set/adult.data

Column delimiter: ,

Row delimiter: ○ Line break ● Custom \n

Quote char: "

Quote escape char: "

Comment char: #

☑ Has column header

□ Has row ID

□ Support short data rows

□ Prepend file index to row ID

The suggested column types are based on the first 10000 rows only. See 'Advanced Settings' tab.

Row ID	I age	S workclass	I final we...	S education	I educati...	S marital-...	S occupa...	S relation...	S race	S sex	I capital-...	I capital-l
Row0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0
Row1	50	Self-emp-no...	83311	Bachelors	13	Married-civ-...	Exec-manag...	Husband	White	Male	0	0
Row2	38	Private	215646	HS-grad	9	Divorced	Handlers-cle...	Not-in-family	White	Male	0	0
Row3	53	Private	234721	11th	7	Married-civ-...	Handlers-cle...	Husband	Black	Male	0	0
Row4	28	Private	338409	Bachelors	13	Married-civ-...	Prof-specialty	Wife	Black	Female	0	0
Row5	37	Private	284582	Masters	14	Married-civ-...	Exec-manag...	Wife	White	Female	0	0
Row6	49	Private	160187	9th	5	Married-spo...	Other-service	Not-in-family	Black	Female	0	0
Row7	52	Self-emp-no...	209642	HS-grad	9	Married-civ-...	Exec-manag...	Husband	White	Male	0	0
Row8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0
Row9	42	Private	159449	Bachelors	13	Married-civ-...	Exec-manag...	Husband	White	Male	5178	0
Row10	37	Private	280464	Some-college	10	Married-civ-...	Exec-manag...	Husband	Black	Male	0	0
Row11	30	State-gov	141297	Bachelors	13	Married-civ-...	Prof-specialty	Husband	Asian-Pac-Is...	Male	0	0
Row12	23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0	0
Row13	32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0
Row14	40	Private	121772	Assoc-voc	11	Married-civ-...	Craft-repair	Husband	Asian-Pac-Is...	Male	0	0
Row15	34	Private	245487	7th-8th	4	Married-civ-...	Transport-m...	Husband	Amer-Indian...	Male	0	0
Row16	25	Self-emp-no...	176756	HS-grad	9	Never-married	Farming-fish...	Own-child	White	Male	0	0
Row17	32	Private	186824	HS-grad	9	Never-married	Machine-op-...	Unmarried	White	Male	0	0
Row18	38	Private	28887	11th	7	Married-civ-...	Sales	Husband	White	Male	0	0
Row19	43	Self-emp-no...	292175	Masters	14	Divorced	Exec-manag...	Unmarried	White	Female	0	0
Row20	40	Private	193524	Doctorate	16	Married-civ-...	Prof-specialty	Husband	White	Male	0	0
Row21	54	Private	302146	HS-grad	9	Separated	Other-service	Unmarried	Black	Female	0	0

Figure 2.13. Configuration window of the CSV Reader node.

Customizing Column Properties

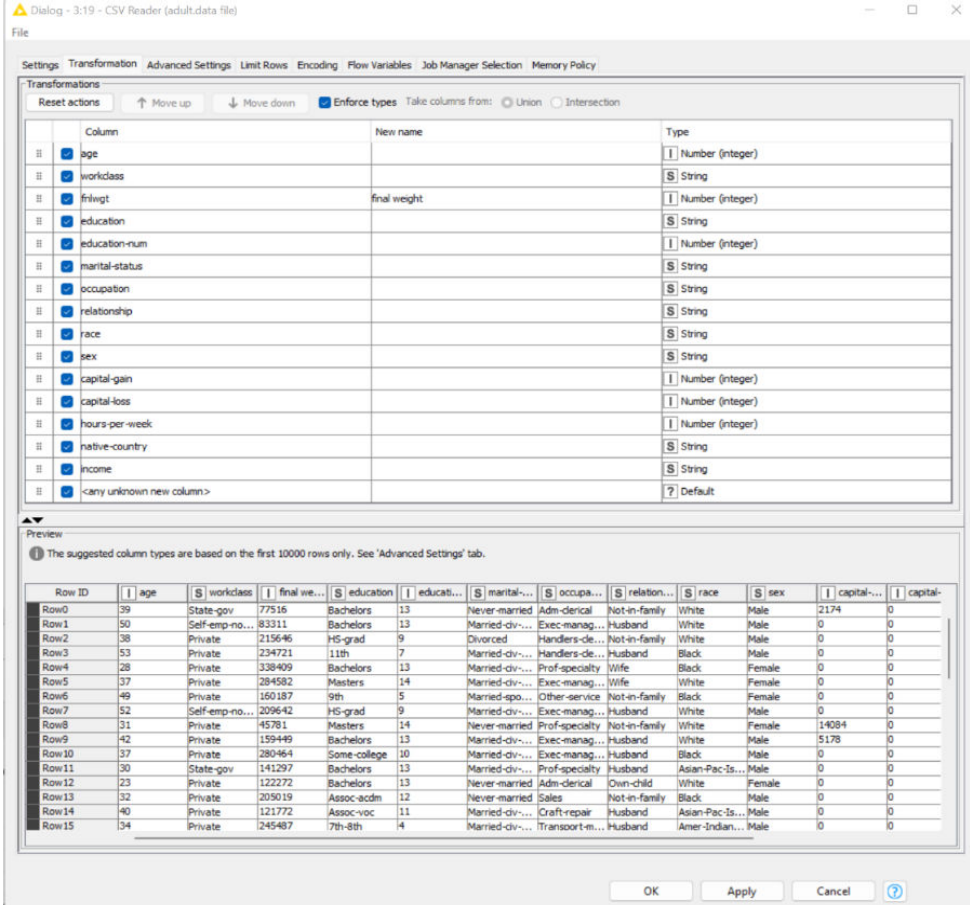
It is possible to customize the way that each column data is read.

For example, the *adult.csv* file contains a field unclearly named “fnlwgt”. We can change this column header to a more meaningful “final weight” in the ***Transformation tab*** (Figure 2.14), just by writing the new name in the corresponding New Name field.

In the ***Settings tab*** (Figure 2.13), we can also set: the delimiter character that separates the column, whether to use the first data row for the column headers, whether to use the first column as RowIDs, and if we can tolerate shorter data rows.

The ***Limit Rows tab*** allows to skip the first lines in the text file. This is usually very useful when dealing with files with a header text.

The ***Advanced Settings tab*** offers a few additional settings, and the ***Encoding tab*** allows to set the right encoding for the text.



☑ Enforce types   Take columns from: ● Union ○ Intersection

		Column	New name	Type
⠿	☑	age		I Number (integer)
⠿	☑	workclass		S String
⠿	☑	fnlwgt	final weight	I Number (integer)
⠿	☑	education		S String
⠿	☑	education-num		I Number (integer)
⠿	☑	marital-status		S String
⠿	☑	occupation		S String
⠿	☑	relationship		S String
⠿	☑	race		S String
⠿	☑	sex		S String
⠿	☑	capital-gain		I Number (integer)
⠿	☑	capital-loss		I Number (integer)
⠿	☑	hours-per-week		I Number (integer)
⠿	☑	native-country		S String
⠿	☑	income		S String
⠿	☑	<any unknown new column>		? Default

Preview

The suggested column types are based on the first 10000 rows only. See 'Advanced Settings' tab.

Row ID	I age	S workclass	I final we...	S education	I educati...	S marital-...	S occupa...	S relation...	S race	S sex	I capital-...	I capital-
Row0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0
Row1	50	Self-emp-no...	83311	Bachelors	13	Married-civ-...	Exec-manag...	Husband	White	Male	0	0
Row2	38	Private	215646	HS-grad	9	Divorced	Handlers-cle...	Not-in-family	White	Male	0	0
Row3	53	Private	234721	11th	7	Married-civ-...	Handlers-cle...	Husband	Black	Male	0	0
Row4	28	Private	338409	Bachelors	13	Married-civ-...	Prof-specialty	Wife	Black	Female	0	0
Row5	37	Private	284582	Masters	14	Married-civ-...	Exec-manag...	Wife	White	Female	0	0
Row6	49	Private	160187	9th	5	Married-spo...	Other-service	Not-in-family	Black	Female	0	0
Row7	52	Self-emp-no...	209642	HS-grad	9	Married-civ-...	Exec-manag...	Husband	White	Male	0	0
Row8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0
Row9	42	Private	159449	Bachelors	13	Married-civ-...	Exec-manag...	Husband	White	Male	5178	0
Row10	37	Private	280464	Some-college	10	Married-civ-...	Exec-manag...	Husband	Black	Male	0	0
Row11	30	State-gov	141297	Bachelors	13	Married-civ-...	Prof-specialty	Husband	Asian-Pac-Is...	Male	0	0
Row12	23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0	0
Row13	32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0
Row14	40	Private	121772	Assoc-voc	11	Married-civ-...	Craft-repair	Husband	Asian-Pac-Is...	Male	0	0
Row15	34	Private	245487	7th-8th	4	Married-civ-...	Transport-m...	Husband	Amer-Indian...	Male	0	0

Figure 2.14. Customize how to read columns.

Notice the three dots in the lower left corner of the CSV Reader icon. These dots indicate dynamic port. By clicking on the three dots, you can add one or more input ports to this node to connect to an external file system, like Amazon S3, HDFS, Databricks, Micrsoft Azure, and so on …

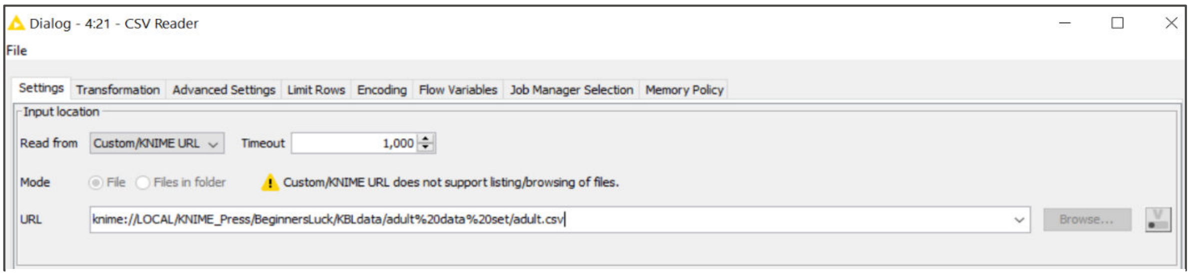
The three dots in a node always mean dynamic ports.

> ***Note.*** After configuring the *CSV Reader* node, its state moves to the yellow traffic light.

What we have described here is the long way to get a *CSV Reader* node configured! 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 Input Location/File field in the CSV Reader configuration window, we could just drag and drop the file from the Space 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 *CSV 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 (Figure 2.15). This special protocol is paired with the option “Custom/KNIME URL” in the Read From field.


Dialog - 4:21 - CSV Reader

Input location

Read from: Custom/KNIME URL　Timeout: 1,000

Mode: ● File ○ Files in folder　⚠ Custom/KNIME URL does not support listing/browsing of files.

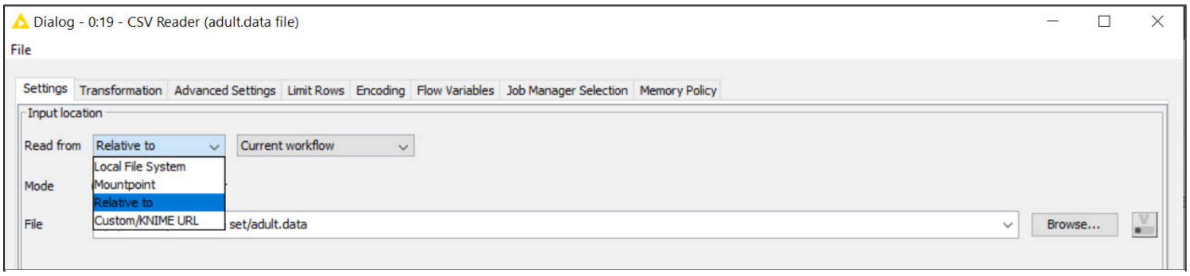
URL: knime://LOCAL/KNIME_Press/BeginnersLuck/KBLdata/adult%20data%20set/adult.csv

Figure 2.15. Automatic configuration of the Input Location part in the Settings tab of a CSV Reader node created via drag & drop of a file from the KNIME Explorer panel into the workflow editor panel.

Other options available to locate a file in the Input Location part of the configuration window of a CSV Reader node are from *Local File System*, from *a specific Mountpoint*, from *a relative location*, and *the Custom/KNIME URL* that we have already seen. The first three options define the starting point for the file path.

- “Local File System” treats the file path as an absolute path.
- “Mountpoint” defines your path as a relative path to the Mountpoint selected in the secondary menu on the right, that is: LOCAL as the current workspace folder, My KNIME Hub (you need to be logged in to see the option), another workspace, or the EXAMPLES.

- “Relative to” defines the path of the file relatively to a given location, as specified in the secondary menu on the right: the current workflow folder, the data folder within the current workflow folder, current workspace, and current mountpoint.
- “Custom/ KNIME URL” refers to the usage of the *knime://* protocol


Dialog - 0:19 - CSV Reader (adult.data file)

Input location

Read from: Relative to — Current workflow

Mode

File: set/adult.data

Figure 2. 16. Options available in the “Read from” field in the Input Location part in the Settings tab of a CSV Reader

The Browse button can help you find the file while the “Read from” option will write it in the right format.

Notice that in Reader and Writer nodes that still do not have the “read from” field, the *knime://* protocol still works. Let’s have a look at the different *knime://* protocol options.

The *knime://* Protocol

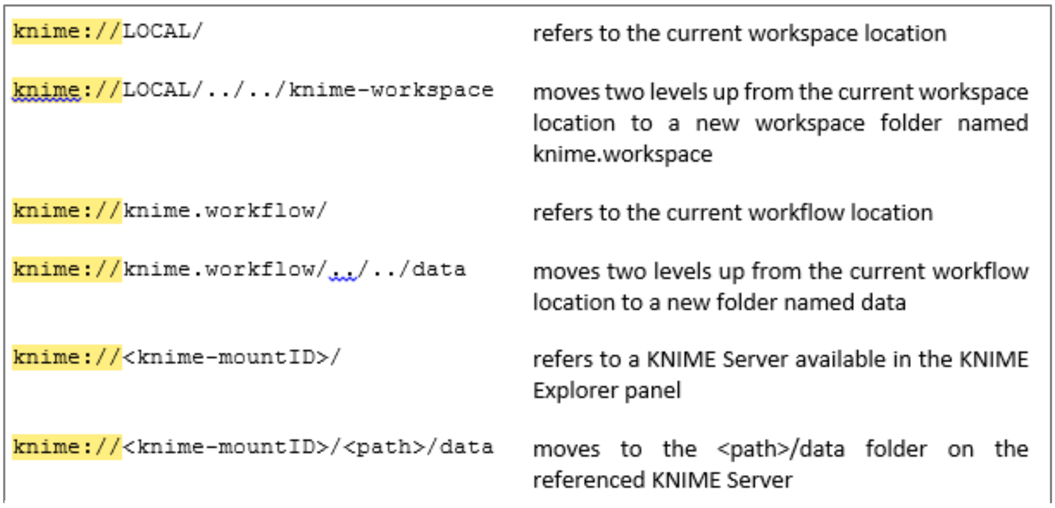
The *knime://* protocol is a special protocol that allows you to reference the local workspace or the local workflow in a path. This then permits the creation of relative paths making you independent of the workspace folder absolute URL, but just dependent on the workspace folder structure.

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 *CSV Reader* will keep finding the file and reading it.

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

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 *CSV 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 *CSV Reader* node
- Enter the node’s new comment (for example “Adult data set”)
- Click elsewhere



`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

Figure 2.17. Possible paths using knime:// protocol.

We have now changed the comment under the *CSV Reader* node from "Node 1" to "Adult data set with *file:// protocol*".

Note. Notice the three dots in the lower left corner of the node. Clicking on them allows you to connect to an external file system and access files from there.

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 “CSV 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 *CSV 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 occurrences 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 CSV Reader node configuration.

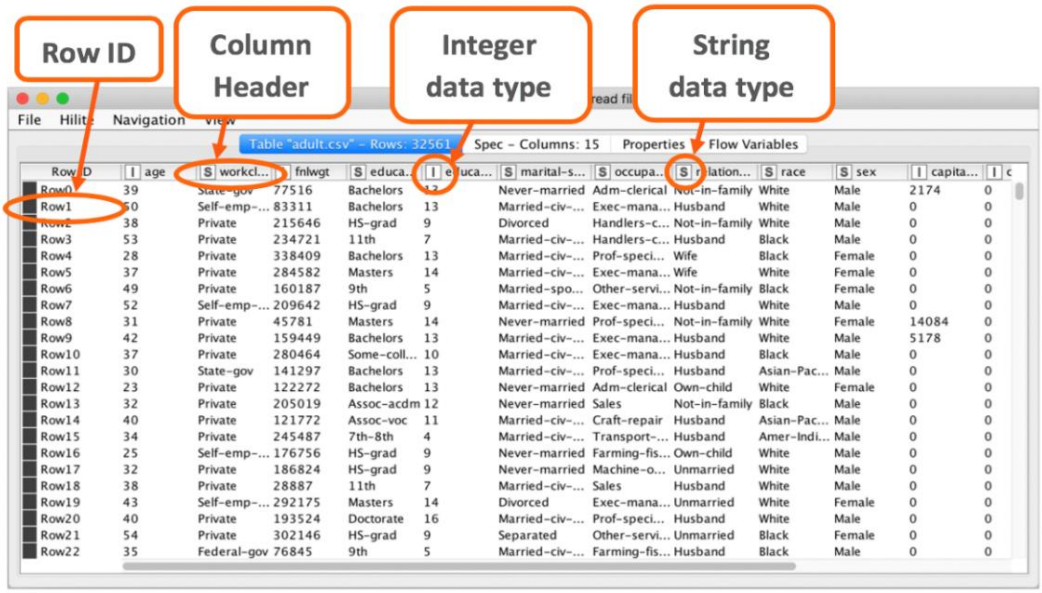
Note. Missing values are represented by default with question marks. They are not question marks, they are missing and are represented with question marks. Question marks in the text file are correctly read as question marks, but they are not missing data. Missing values could be represented by anything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

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.



Row ID	I age	S workcl...	fnlwgt	S educa...	I educa...	S marital-s...	S occupa...	S relation...	S race	S sex	I capita...	I c
Row0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0
Row1	50	Self-emp-...	83311	Bachelors	13	Married-civ-...	Exec-mana...	Husband	White	Male	0	0
Row2	38	Private	215646	HS-grad	9	Divorced	Handlers-c...	Not-in-family	White	Male	0	0
Row3	53	Private	234721	11th	7	Married-civ-...	Handlers-c...	Husband	Black	Male	0	0
Row4	28	Private	338409	Bachelors	13	Married-civ-...	Prof-speci...	Wife	Black	Female	0	0
Row5	37	Private	284582	Masters	14	Married-civ-...	Exec-mana...	Wife	White	Female	0	0
Row6	49	Private	160187	9th	5	Married-spo...	Other-servi...	Not-in-family	Black	Female	0	0
Row7	52	Self-emp-...	209642	HS-grad	9	Married-civ-...	Exec-mana...	Husband	White	Male	0	0
Row8	31	Private	45781	Masters	14	Never-married	Prof-speci...	Not-in-family	White	Female	14084	0
Row9	42	Private	159449	Bachelors	13	Married-civ-...	Exec-mana...	Husband	White	Male	5178	0
Row10	37	Private	280464	Some-coll...	10	Married-civ-...	Exec-mana...	Husband	Black	Male	0	0
Row11	30	State-gov	141297	Bachelors	13	Married-civ-...	Prof-speci...	Husband	Asian-Pac...	Male	0	0
Row12	23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0	0
Row13	32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0
Row14	40	Private	121772	Assoc-voc	11	Married-civ-...	Craft-repair	Husband	Asian-Pac...	Male	0	0
Row15	34	Private	245487	7th-8th	4	Married-civ-...	Transport-...	Husband	Amer-Indi...	Male	0	0
Row16	25	Self-emp-...	176756	HS-grad	9	Never-married	Farming-fis...	Own-child	White	Male	0	0
Row17	32	Private	186824	HS-grad	9	Never-married	Machine-o...	Unmarried	White	Male	0	0
Row18	38	Private	28887	11th	7	Married-civ-...	Sales	Husband	White	Male	0	0
Row19	43	Self-emp-...	292175	Masters	14	Divorced	Exec-mana...	Unmarried	White	Female	0	0
Row20	40	Private	193524	Doctorate	16	Married-civ-...	Prof-speci...	Husband	White	Male	0	0
Row21	54	Private	302146	HS-grad	9	Separated	Other-servi...	Unmarried	Black	Female	0	0
Row22	35	Federal-gov	76845	9th	5	Married-civ-...	Farming-fis...	Husband	Black	Male	0	0

Figure 2.18. The KNIME Data Structure.

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.

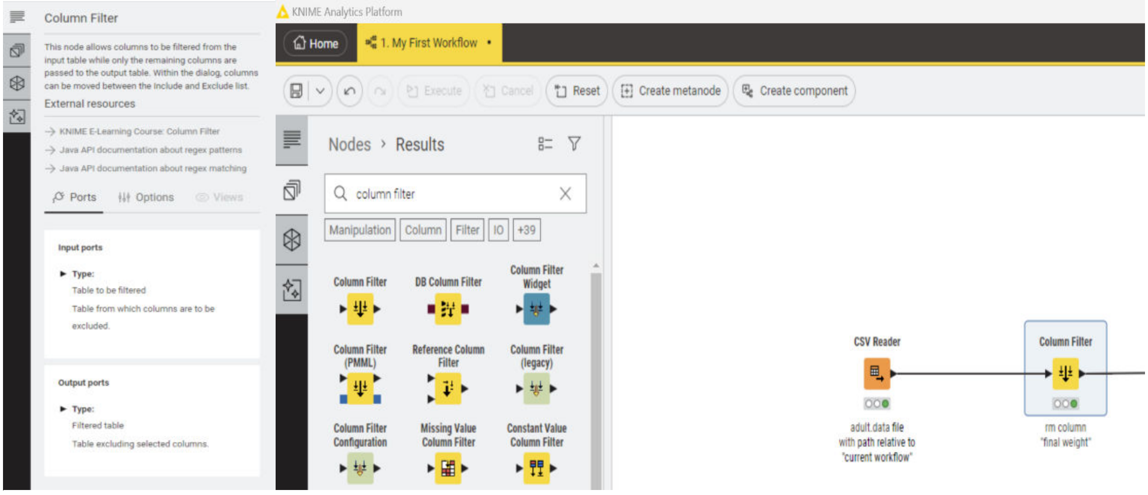
2.5. Filter Data Columns

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 left, there is a whole category called "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 "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 “CSV Reader” node) by clicking at the output of the “CSV Reader” node and releasing at the input of the “Column Filter” node.



Column Filter

This node allows columns to be filtered from the input table while only the remaining columns are passed to the output table. Within the dialog, columns can be moved between the Include and Exclude list.

External resources

→ KNIME E-Learning Course: Column Filter
→ Java API documentation about regex patterns
→ Java API documentation about regex matching

Input ports

► Type:
Table to be filtered
Table from which columns are to be excluded.

Output ports

► Type:
Filtered table
Table excluding selected columns.

Figure 2.19. Creating a Column Filter node.

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

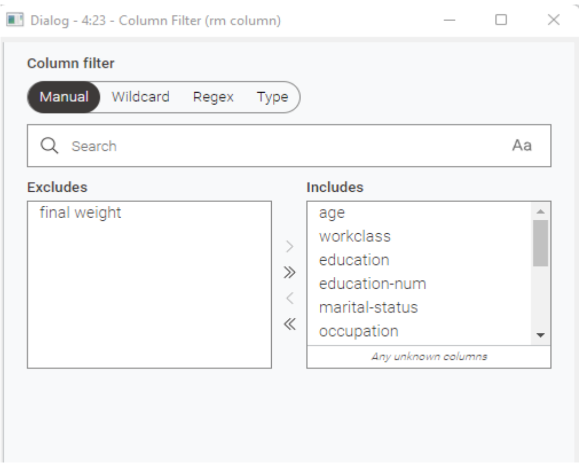
The first setting in the configuration window is the type of filtering. You can select and retain columns **manually**, **by type**, or **by name**, according to the options at the top of the configuration window (Figure 2.20).

Manual Selection:

If the "Manual Selection" option is selected, the configuration window shows 2 sets of columns (Figure 2.20):

- The columns to be included in the data table ("Includes" set on the right)
- The columns to be excluded from the data table ("Excludes" set on the left)

The "**Search**" bar allows to search for a specific column.



Excludes: final weight

Includes: age, workclass, education, education-num, marital-status, occupation

Figure 2.20. Configuration window of the Column Filter

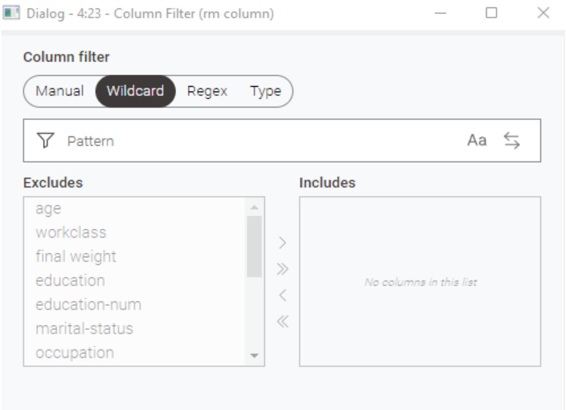
You can add and remove columns from one set to the other on double click.

- "Includes" 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.
- "Excludes" 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.

Columns with names matching the expression will be included in the output data table.



Excludes: age, workclass, final weight, education, education-num, marital-status, occupation

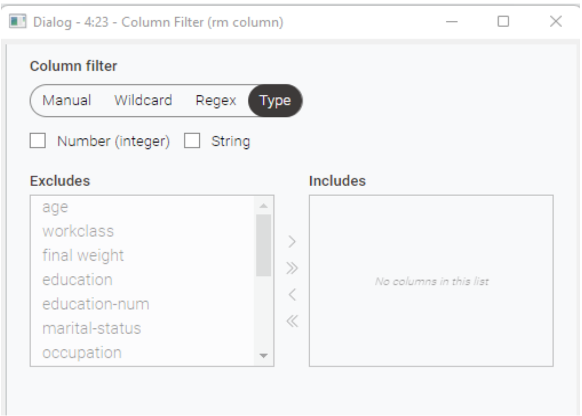
Figure 2.21. Column Filter node Configuration: "Wildcard/Regex Selection".

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, **Number(integer)** will keep all numerical columns only in the node’s output table.

Selected String will keep all String type columns only in the output table.



Dialog - 4:23 - Column Filter (rm column)

Column filter

Manual Wildcard Regex **Type**

□ Number (integer) □ String

Excludes: age, workclass, final weight, education, education-num, marital-status, occupation

Includes: No columns in this list

Figure 2.22. Column Filter node configuration: "Type.

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.

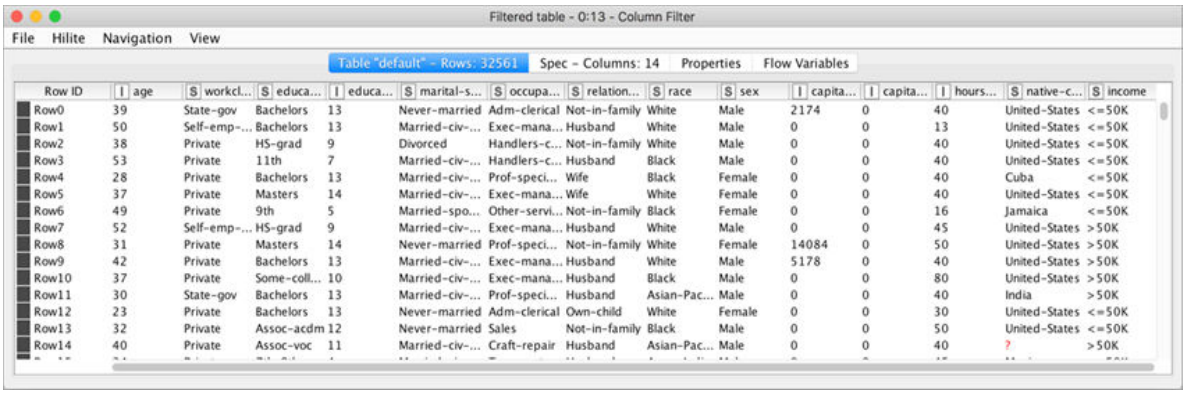
In our example workflow “my First Workflow”, we wanted to remove the “final weight” column.

We set the column filter mode to “Manual Selection” and we populated the Exclude panel with column “final weight”.

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.

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



Filtered table - 0:13 - Column Filter

File Hilite Navigation View

Table "default" - Rows: 32561 | Spec - Columns: 14 | Properties | Flow Variables

Row ID	I age	S workcl...	S educa...	I educa...	S marital-s...	S occupa...	S relation...	S race	S sex	I capita...	I capita...	I hours...	S native-c...	S 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

Figure 2.23. The column filtered table does not contain the column "finale weight".

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 "Manipulation" and navigate to the node "Row Filter" in "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.

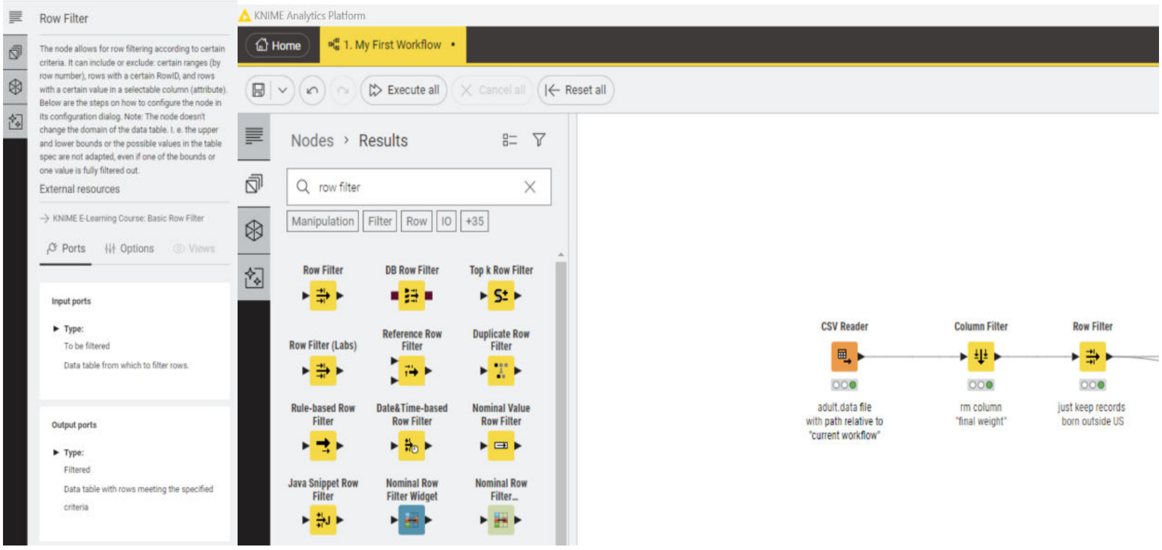
The node implements three filter criteria:

- 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”



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 RowID, and rows with a certain value in a selectable column (attribute). Below are the steps on how to configure the node in its configuration dialog. Note: The node doesn't change the domain of the data table. I. e. the upper and lower bounds or the possible values in the table spec are not adapted, even if one of the bounds or one value is fully filtered out.

External resources

→ KNIME E-Learning Course: Basic Row Filter

Input ports

Type: To be filtered

Data table from which to filter rows.

Output ports

Type: Filtered

Data table with rows meeting the specified criteria

Figure 2.24. Creating 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-Coordinate 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;

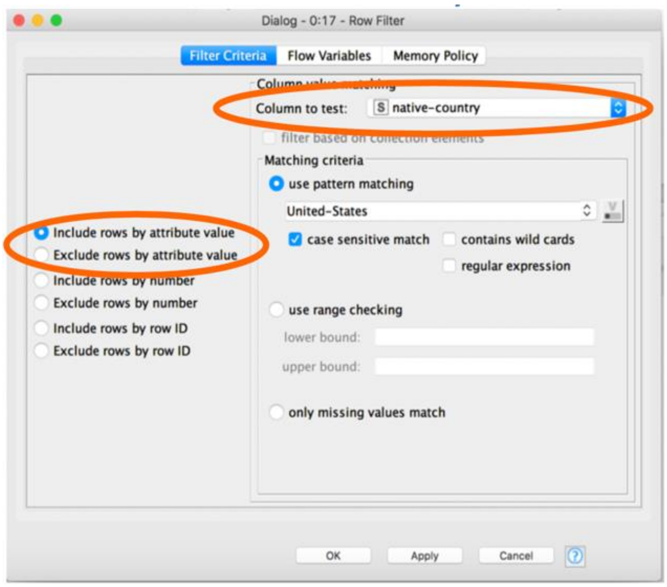
- "Database Row Filter" node to implement a row filtering SQL query to run directly on the database.

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.



Dialog - 0:17 - Row Filter

Filter Criteria | Flow Variables | Memory Policy

Column value matching

Column to test: S native-country

□ filter based on collection elements

Matching criteria

● use pattern matching

United-States

☑ case sensitive match

□ contains wild cards

□ regular expression

○ use range checking

lower bound:

upper bound:

○ only missing values match

● Include rows by attribute value

○ Exclude rows by attribute value

○ Include rows by number

○ Exclude rows by number

○ Include rows by row ID

○ Exclude rows by row ID

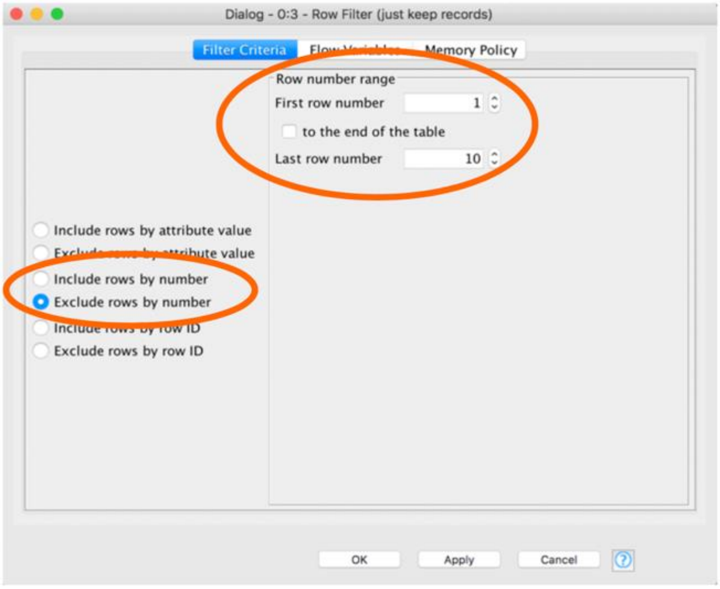
Figure 2.25. Row Filter criterion by attribute value.

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.

By Row Number:

If you know where your desired or undesired rows are, you can just



Dialog - 0:3 - Row Filter (just keep records)

Filter Criteria | Flow Variables | Memory Policy

Row number range

First row number 1

□ to the end of the table

Last row number 10

○ Include rows by attribute value

○ Exclude rows by attribute value

○ Include rows by number

● Exclude rows by number

○ Include rows by row ID

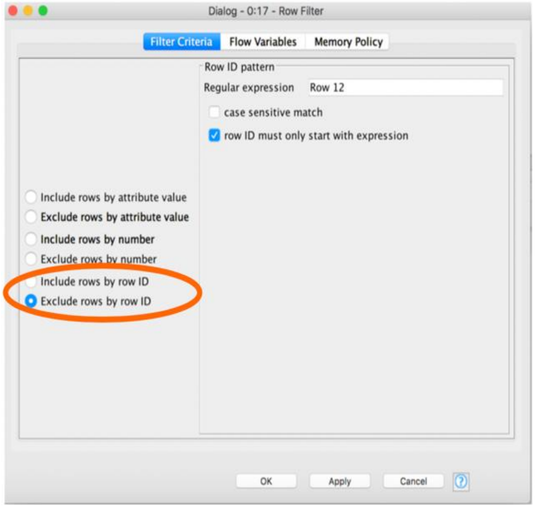
○ Exclude rows by row ID

Figure 2.26. Row Filter criterion by row number.

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.

By RowID:



Row ID pattern

Regular expression: Row 12

☐ case sensitive match

☑ row ID must only start with expression

○ Include rows by attribute value
○ Exclude rows by attribute value
○ Include rows by number
○ Exclude rows by number
○ Include rows by row ID
● Exclude rows by row ID

Figure 2.27. Row Filter criterion 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.

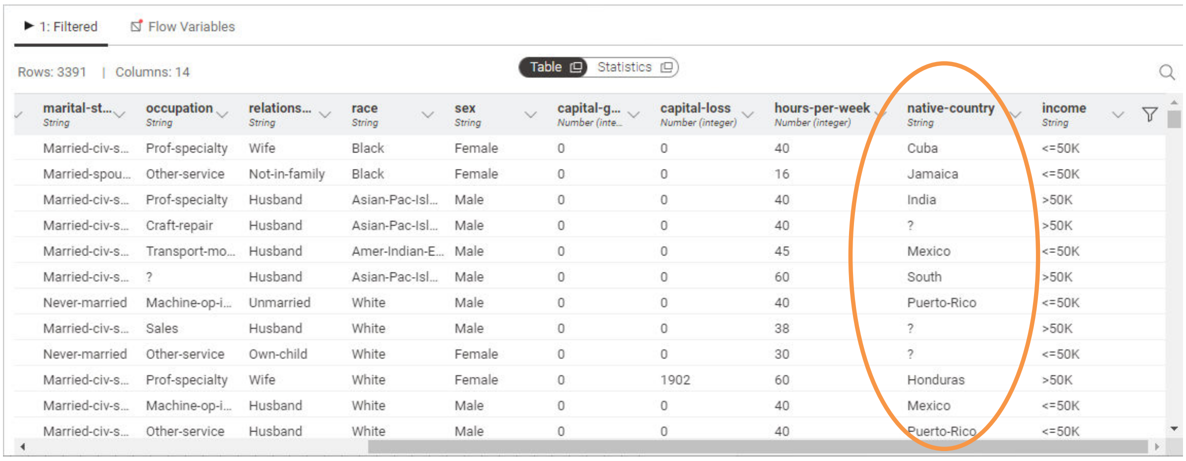
In order to retain all rows with data referring to people born outside of the United States, we need to:

- 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 see the filtered table below in the Node Monitor. There should be no “United States” in column native-country.



Rows: 3391 | Columns: 14

marital-st... String	occupation String	relations... String	race String	sex String	capital-g... Number (inte...	capital-loss Number (integer)	hours-per-week Number (integer)	native-country String	income String
Married-civ-s...	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
Married-spou...	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
Married-civ-s...	Prof-specialty	Husband	Asian-Pac-Isl...	Male	0	0	40	India	>50K
Married-civ-s...	Craft-repair	Husband	Asian-Pac-Isl...	Male	0	0	40	?	>50K
Married-civ-s...	Transport-mo...	Husband	Amer-Indian-E...	Male	0	0	45	Mexico	<=50K
Married-civ-s...	?	Husband	Asian-Pac-Isl...	Male	0	0	60	South	>50K
Never-married	Machine-op-i...	Unmarried	White	Male	0	0	40	Puerto-Rico	<=50K
Married-civ-s...	Sales	Husband	White	Male	0	0	38	?	>50K
Never-married	Other-service	Own-child	White	Female	0	0	30	?	<=50K
Married-civ-s...	Prof-specialty	Wife	White	Female	0	1902	60	Honduras	>50K
Married-civ-s...	Machine-op-i...	Husband	White	Male	0	0	40	Mexico	<=50K
Married-civ-s...	Other-service	Husband	White	Male	0	0	40	Puerto-Rico	<=50K

Figure 2.28. The row filtered table has no pattern "United States" in column "native-country".

2.7. Write Data to a File

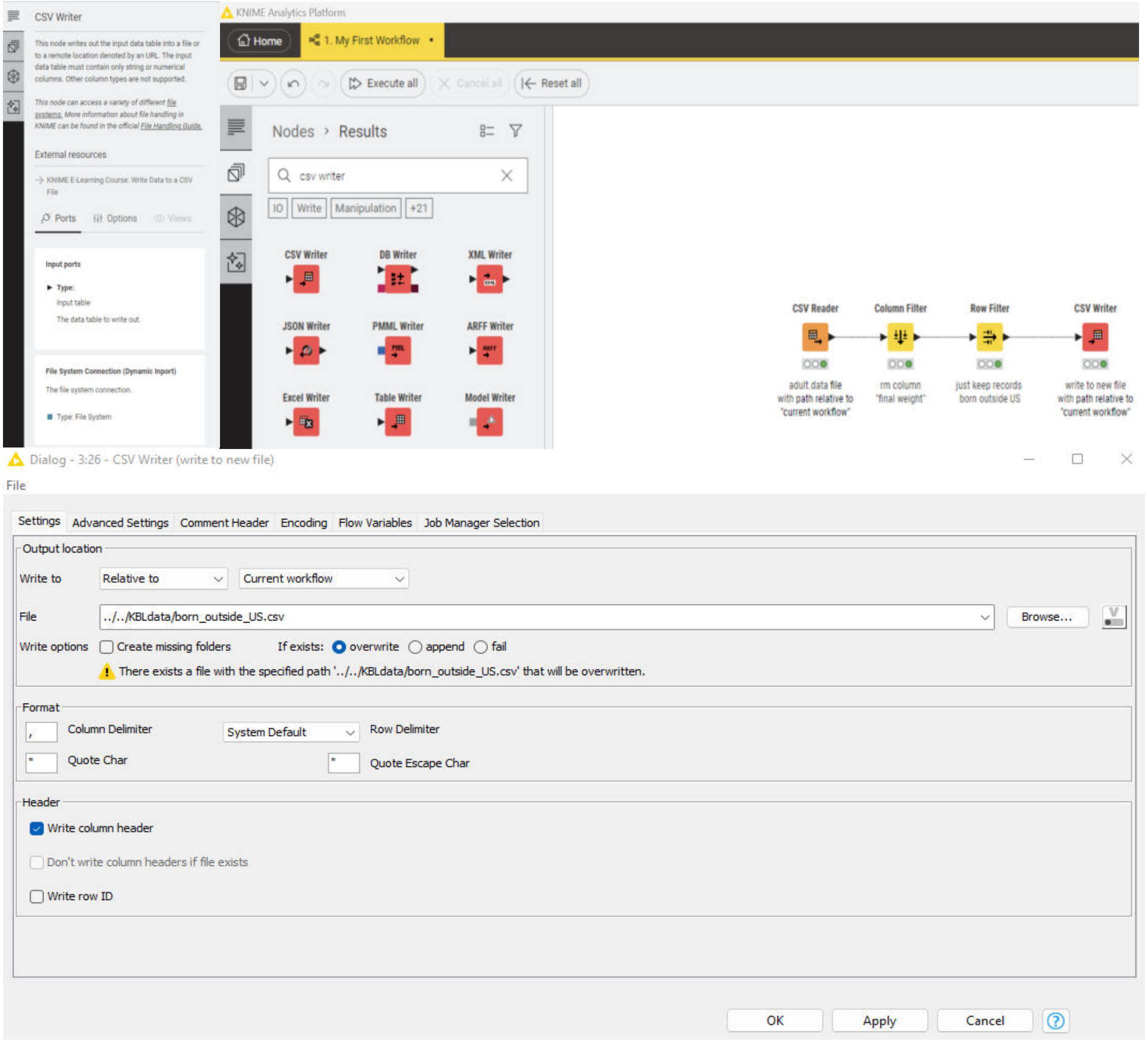
Now we want to write the processed data table to a file. There are many nodes that can write to a file. Let's choose the easiest and most standard format for now: the CSV (Comma Separated Values) format.

Create *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
- Right-click the "CSV Writer" node and select "Configure" or double-click it to open its configuration window.
- Configuration window has four tabs: Settings, Advanced Settings, Comment Header, and Encoding

The configuration window of the CSV Writer node is similar to the configuration window of the CSV Reader node.



This node writes out the input data table into a file or to a remote location denoted by an URL. The input data table must contain only string or numerical columns. Other column types are not supported.

This node can access a variety of different file systems. More information about file handling in KNIME can be found in the official File Handling Guide.

External resources

→ KNIME E-Learning Course: Write Data to a CSV File

Input ports

▸ Type: Input table

The data table to write out.

File System Connection (Dynamic Inport)

The file system connection.

Type: File System

csv writer

adult.data file with path relative to "current workflow"

rm column "final weight"

just keep records born outside US

write to new file with path relative to "current workflow"

Dialog - 3:26 - CSV Writer (write to new file)

Output location

Write to: Relative to | Current workflow

File: ../../KBLdata/born_outside_US.csv

Write options: □ Create missing folders   If exists: ● overwrite ○ append ○ fail

⚠ There exists a file with the specified path '../../KBLdata/born_outside_US.csv' that will be overwritten.

Format

Column Delimiter: ,   Row Delimiter: System Default

Quote Char: "   Quote Escape Char: "

Header

☑ Write column header

□ Don't write column headers if file exists

□ Write row ID

Figure 2.29. Create and configure a "CSV Writer" node.

Configure the *CSV Writer* Node

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

- The path of the output file in Output Location. Notice that Output Location offers the same options as the Input Location in the CSV Reader node.
- A few additional options about the data structure, such as:
  - Whether to write the column headers and/or RowID in output file
  - The Column Delimiter character

  - The writing mode if file already exists: Overwrite, Append, Fail (does not write to file)

The **Advanced Settings tab** allows for a few more specs, like the quote character, the decimal separator, or the compression to a gzip file. The **Comment Header tab** allows to automatically write a header with comments on top of the data. The **Encoding tab** chooses the appropriate encoding for the text. The **Memory Policy** tab 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 **Flow Variables** tab and the **Job Manager Selection** tab.

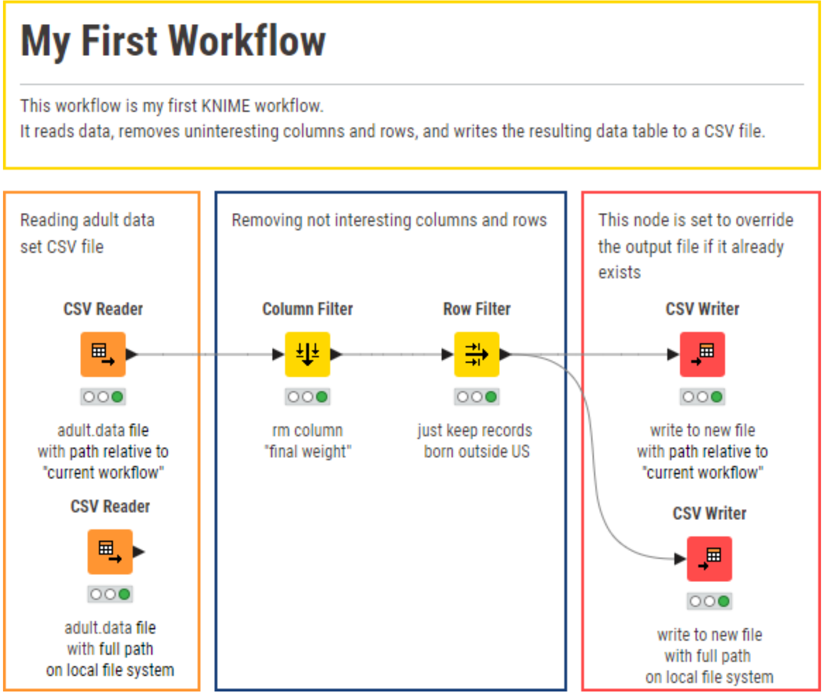
Note. Writing in mode "Append" can be tricky, because it just appends the new data to an existing file without checking the data structure nor matching the column by name. So, if the data table structure has changed, for example because of new or deleted columns, the output CSV file will not be consistent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

In some cases, you might want to select "Fail" 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.



My First Workflow

This workflow is my first KNIME workflow.
It reads data, removes uninteresting columns and rows, and writes the resulting data table to a CSV file.

Reading adult data set CSV file

Removing not interesting columns and rows

This node is set to override the output file if it already exists

CSV Reader
adult.data file with path relative to "current workflow"
CSV Reader
adult.data file with full path on local file system
Column Filter
rm column "final weight"
Row Filter
just keep records born outside US
CSV Writer
write to new file with path relative to "current workflow"
CSV Writer
write to new file with full path on local file system

Figure 2.30. 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 KBLdata folder) 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

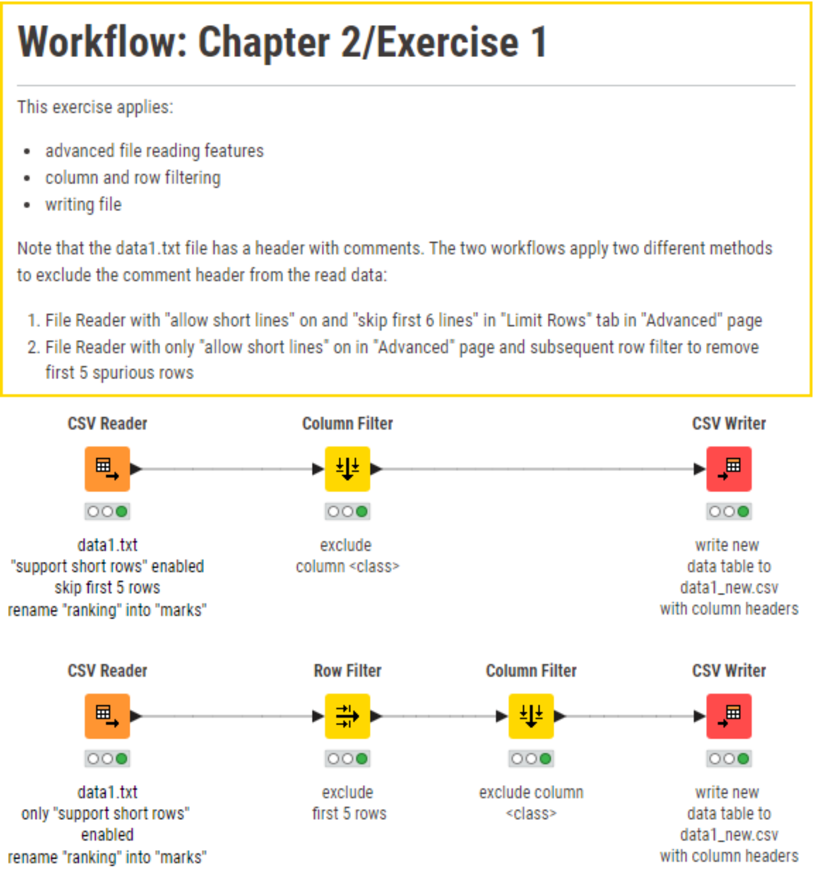
The file has some comments at the very beginning, which of course do not have the same length as the other lines in the file.

First, you need to enable the options “has column headers” and “support short data rows” in the “Settings” tab and rename column “ranking” as “marks” in the “Transformation” tab.

Then you can do one of the two approaches:

1. Set the “Skip first data rows …” to 5 in the “Limit Rows” tab.
2. Use the “Row Filter” node to exclude the first 5 rows of the read data.

Again, the solution workflow is available in the folder of workflows you downloaded from the KNIME Community Hub.



Workflow: Chapter 2/Exercise 1

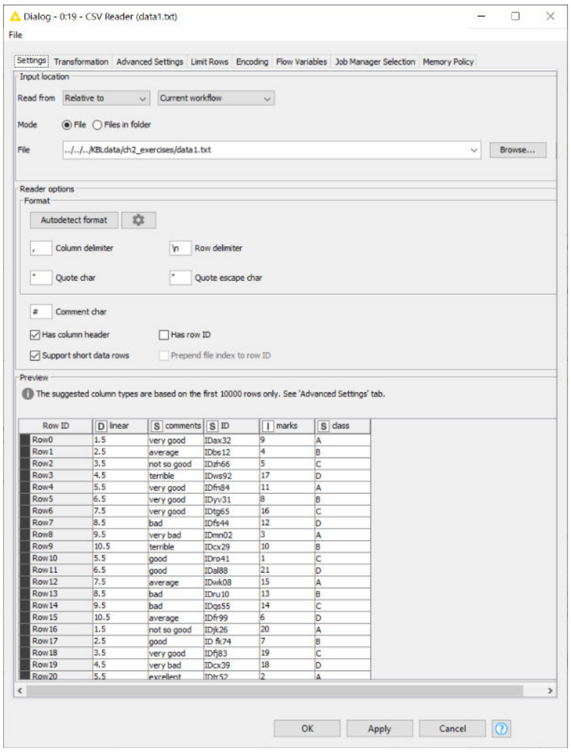
This exercise applies:

- advanced file reading features
- column and row filtering
- writing file

Note that the data1.txt file has a header with comments. The two workflows apply two different methods to exclude the comment header from the read data:

1. File Reader with "allow short lines" on and "skip first 6 lines" in "Limit Rows" tab in "Advanced" page
2. File Reader with only "allow short lines" on in "Advanced" page and subsequent row filter to remove first 5 spurious rows

Figure 2.31. Exercise 1: Two possible solution workflows.



Input location

Read from: Relative to | Current workflow

Mode: ● File ○ Files in folder

File: ../../../KBLdata/ch2_exercises/data1.txt

Reader options

Format

, Column delimiter   \n Row delimiter

" Quote char   " Quote escape char

# Comment char

☑ Has column header   □ Has row ID

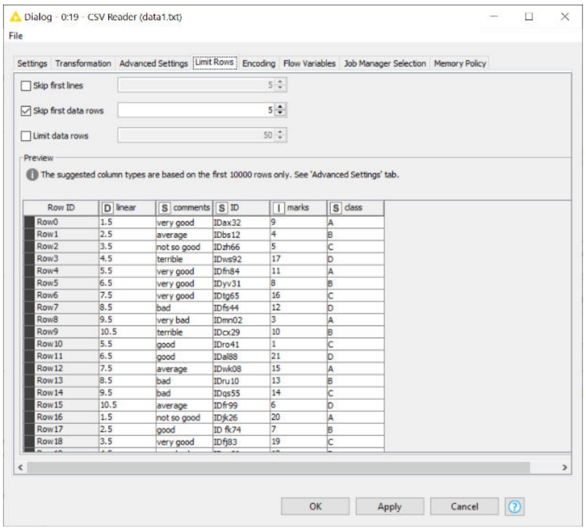
☑ Support short data rows   □ Prepend file index to row ID

Preview

The suggested column types are based on the first 10000 rows only. See 'Advanced Settings' tab.

Row ID	D linear	S comments	S ID	I marks	S class
Row0	1.5	very good	IDax32	9	A
Row1	2.5	average	IDbs12	4	B
Row2	3.5	not so good	IDzh66	5	C
Row3	4.5	terrible	IDws92	17	D
Row4	5.5	very good	IDfn84	11	A
Row5	6.5	very good	IDyv31	8	B
Row6	7.5	very good	IDtg65	16	C
Row7	8.5	bad	IDfs44	12	D
Row8	9.5	very bad	IDmn02	3	A
Row9	10.5	terrible	IDcx29	10	B
Row10	5.5	good	IDro41	1	C
Row11	6.5	good	IDal88	21	D
Row12	7.5	average	IDwk08	15	A
Row13	8.5	bad	IDru10	13	B
Row14	9.5	bad	IDqs55	14	C
Row15	10.5	average	IDfr99	6	D
Row16	1.5	not so good	IDjk26	20	A
Row17	2.5	good	ID fk74	7	B
Row18	3.5	very good	IDfj83	19	C
Row19	4.5	very bad	IDcx39	18	D
Row20	5.5	excellent	IDtr52	2	A

Figure 2.32. Exercise 1: CSV Reader "Settings" tab configuration.



□ Skip first lines 5

☑ Skip first data rows 5

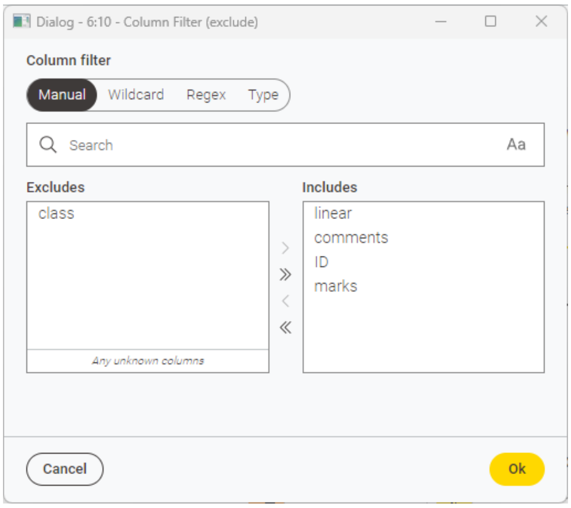
□ Limit data rows 50

Preview

The suggested column types are based on the first 10000 rows only. See 'Advanced Settings' tab.

Row ID	D linear	S comments	S ID	I marks	S class
Row0	1.5	very good	IDax32	9	A
Row1	2.5	average	IDbs12	4	B
Row2	3.5	not so good	IDzh66	5	C
Row3	4.5	terrible	IDws92	17	D
Row4	5.5	very good	IDfn84	11	A
Row5	6.5	very good	IDyv31	8	B
Row6	7.5	very good	IDtg65	16	C
Row7	8.5	bad	IDfs44	12	D
Row8	9.5	very bad	IDmn02	3	A
Row9	10.5	terrible	IDcx29	10	B
Row10	5.5	good	IDro41	1	C
Row11	6.5	good	IDal88	21	D
Row12	7.5	average	IDwk08	15	A
Row13	8.5	bad	IDru10	13	B
Row14	9.5	bad	IDqs55	14	C
Row15	10.5	average	IDfr99	6	D
Row16	1.5	not so good	IDjk26	20	A
Row17	2.5	good	ID fk74	7	B
Row18	3.5	very good	IDfj83	19	C

Figure 2.33. Exercise 1: CSV Reader "Limit Rows" tab configuration.



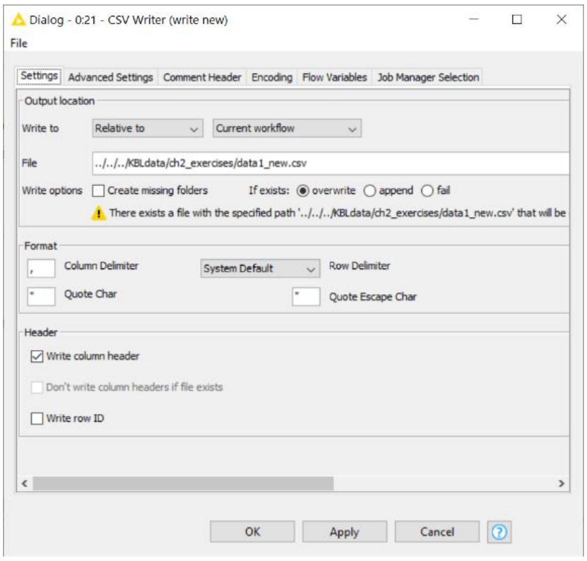
Column filter

Excludes: class

Any unknown columns

Includes: linear, comments, ID, marks

Figure 2.34. Exercise 1: Column Filter configuration.



Output location

Write to: Relative to | Current workflow

File: ../../../KBLdata/ch2_exercises/data1_new.csv

Write options: □ Create missing folders   If exists: ● overwrite ○ append ○ fail

There exists a file with the specified path '../../../KBLdata/ch2_exercises/data1_new.csv' that will be

Format

, Column Delimiter   System Default Row Delimiter

" Quote Char   " Quote Escape Char

Header

☑ Write column header

□ Don't write column headers if file exists

□ Write row ID

Figure 2.35. Exercise 1: CSV Writer "Settings" tab configuration.

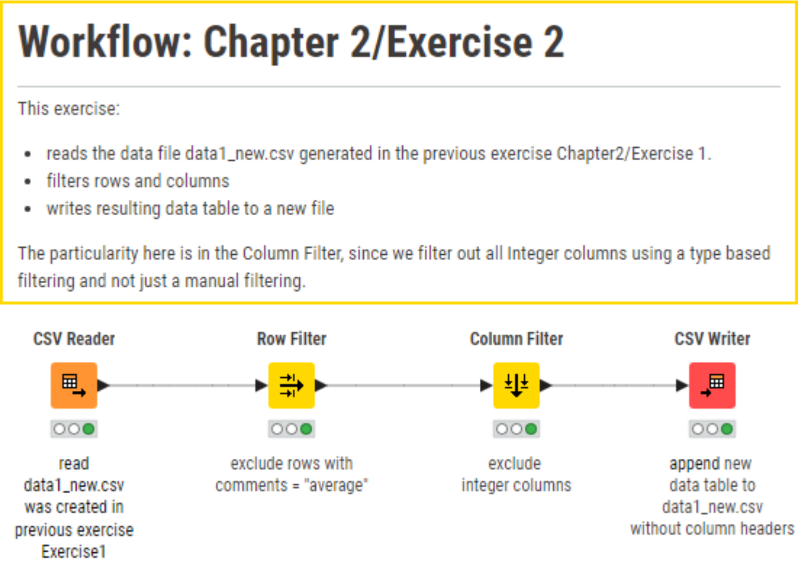
Exercise 2

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 value ‘average’ in the ‘comments’ column
- 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”.



Workflow: Chapter 2/Exercise 2

This exercise:

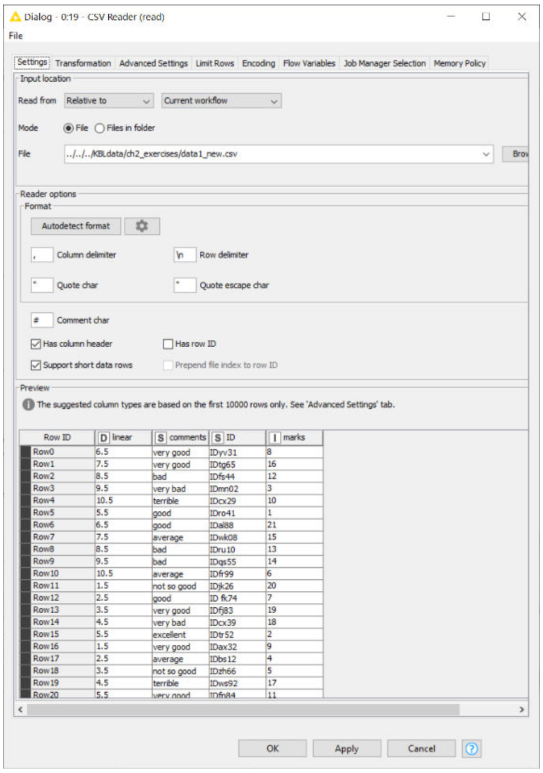
- reads the data file data1_new.csv generated in the previous exercise Chapter2/Exercise 1.
- filters rows and columns
- writes resulting data table to a new file

The particularity here is in the Column Filter, since we filter out all Integer columns using a type based filtering and not just a manual filtering.

CSV Reader
read data1_new.csv was created in previous exercise Exercise1
Row Filter
exclude rows with comments = "average"
Column Filter
exclude integer columns
CSV Writer
append new data table to data1_new.csv without column headers

Figure 2.36. Exercise 2: The workflow.

Note. After copying the “CSV 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.


Dialog - 0:19 - CSV Reader (read)

File

Settings | Transformation | Advanced Settings | Limit Rows | Encoding | Flow Variables | Job Manager Selection | Memory Policy

Input location

Read from: Relative to | Current workflow

Mode: ● File ○ Files in folder

File: ../../../KBLdata/ch2_exercises/data1_new.csv

Reader options

Format

Autodetect format

, Column delimiter | \n Row delimiter

" Quote char | " Quote escape char

Comment char

☑ Has column header | □ Has row ID

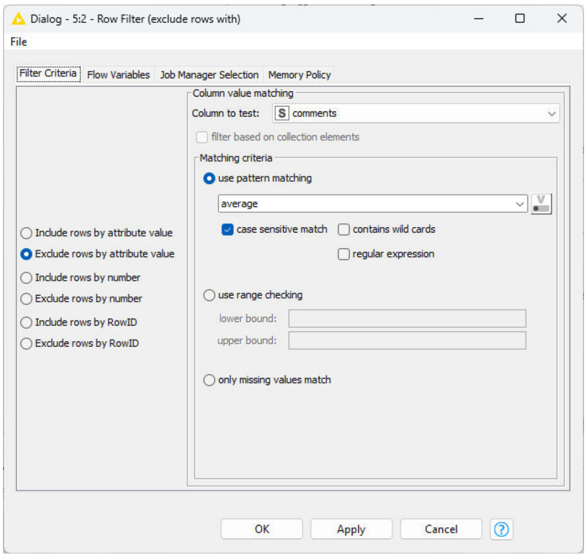
☑ Support short data rows | □ Prepend file index to row ID

Preview

The suggested column types are based on the first 10000 rows only. See 'Advanced Settings' tab.

Row ID	D linear	S comments	S ID	I marks
Row0	6.5	very good	IDyv31	8
Row1	7.5	very good	IDtg65	16
Row2	8.5	bad	IDfs44	12
Row3	9.5	very bad	IDmn02	3
Row4	10.5	terrible	IDcx29	10
Row5	5.5	good	IDro41	1
Row6	6.5	good	IDal88	21
Row7	7.5	average	IDwk08	15
Row8	8.5	bad	IDru10	13
Row9	9.5	bad	IDqs55	14
Row10	10.5	average	IDfr99	6
Row11	1.5	not so good	IDjk26	20
Row12	2.5	good	ID fk74	7
Row13	3.5	very good	IDfj83	19
Row14	4.5	very bad	IDcx39	18
Row15	5.5	excellent	IDtr52	2
Row16	1.5	very good	IDax32	9
Row17	2.5	average	IDbs12	4
Row18	3.5	not so good	IDzh66	5
Row19	4.5	terrible	IDws92	17
Row20	5.5	very good	IDfn84	11

Figure 2.37. Exercise 2: CSV Reader "Settings" configuration.


Dialog - 5:2 - Row Filter (exclude rows with)

File

Filter Criteria | Flow Variables | Job Manager Selection | Memory Policy

○ Include rows by attribute value
● Exclude rows by attribute value
○ Include rows by number
○ Exclude rows by number
○ Include rows by RowID
○ Exclude rows by RowID

Column value matching

Column to test: S comments

□ filter based on collection elements

Matching criteria

● use pattern matching

average

☑ case sensitive match □ contains wild cards

□ regular expression

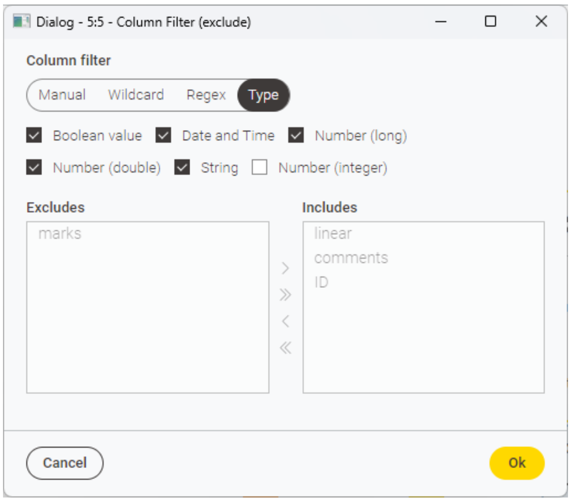
○ use range checking

lower bound:

upper bound:

○ only missing values match

Figure 2.38. Exercise 2: Row Filter configuration.


Dialog - 5:5 - Column Filter (exclude)

Column filter

Manual | Wildcard | Regex | **Type**

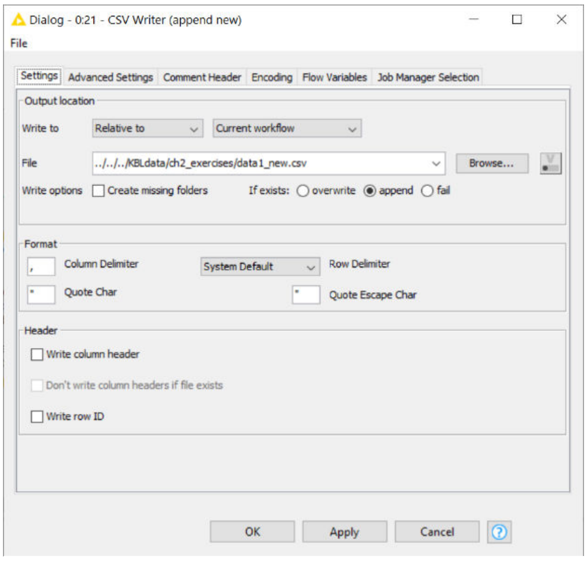
☑ Boolean value ☑ Date and Time ☑ Number (long)

☑ Number (double) ☑ String □ Number (integer)

Excludes: marks

Includes: linear, comments, ID

Figure 2.39. Exercise 2: Column Filter configuration.


Dialog - 0:21 - CSV Writer (append new)

File

Settings | Advanced Settings | Comment Header | Encoding | Flow Variables | Job Manager Selection

Output location

Write to: Relative to | Current workflow

File: ../../../KBLdata/ch2_exercises/data1_new.csv | Browse...

Write options: □ Create missing folders | If exists: ○ overwrite ● append ○ fail

Format

, Column Delimiter | System Default Row Delimiter

" Quote Char | " Quote Escape Char

Header

□ Write column header

□ Don't write column headers if file exists

□ Write row ID

Figure 2.40. Exercise 2: CSV Writer configuration.

Note. We saved the data in "Append" mode into the CSV file. The data from Exercise 2 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 a few data wrangling nodes useful to bring the data into the desired shape, involving data transformation, string manipulation, rule application, and other similar tasks. For that, we will use three workflows: “Column Rename Example”, “Write To DB” and “My First Data Exploration”. “Column Rename Example” is a very simple workflow showing the usage of the Column renamer node, “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.

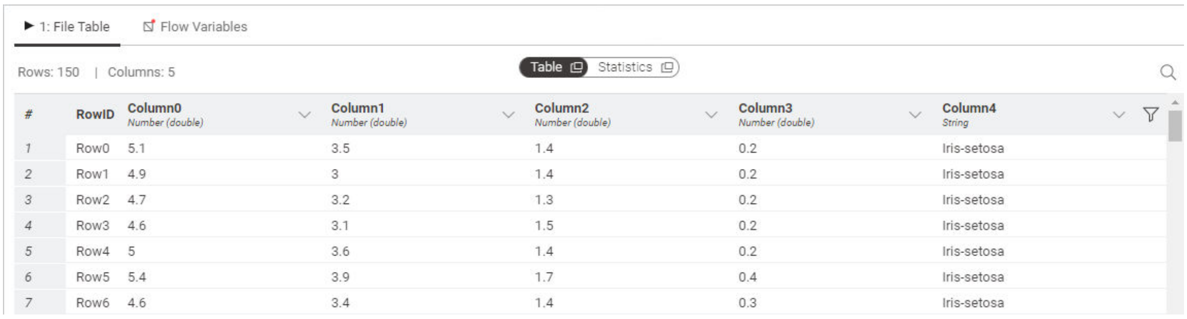
The goal of this chapter is to become familiar with:

- nodes and options for database handling
- the “Views” 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.



Rows: 150 | Columns: 5

#	RowID	Column0 Number (double)	Column1 Number (double)	Column2 Number (double)	Column3 Number (double)	Column4 String
1	Row0	5.1	3.5	1.4	0.2	Iris-setosa
2	Row1	4.9	3	1.4	0.2	Iris-setosa
3	Row2	4.7	3.2	1.3	0.2	Iris-setosa
4	Row3	4.6	3.1	1.5	0.2	Iris-setosa
5	Row4	5	3.6	1.4	0.2	Iris-setosa
6	Row5	5.4	3.9	1.7	0.4	Iris-setosa
7	Row6	4.6	3.4	1.4	0.3	Iris-setosa

Figure 3.1. The iris dataset.

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.

Let's start with reading the data in the "Column Rename Example" workflow. Since the Iris dataset file (iris.data) is not in a standard data format, we cannot directly drag and drop it to the workflow editor. Instead, we read it with a "CSV Reader" node. If we do not change the name of the data columns in the Transformation tab, then the node will read a data table like the one in figure 3.1. We write the comment "read the iris.data file from KBLdata folder" under the "CSV Reader" node, for a quick overview of the node task.

Note. The iris dataset file does not contain column names. The "CSV Reader" node then assigns to each column a default name like "Column0", "Column1", "Column2", "Column3", and "Column4". Besides "Column4", 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 Renamer

The node "Column Renamer" can be found in the "Node Repository" panel under "Manipulation" → "Column" → "Convert & Replace".

The "Column Renamer" node allows for the renaming of columns in the input data table.

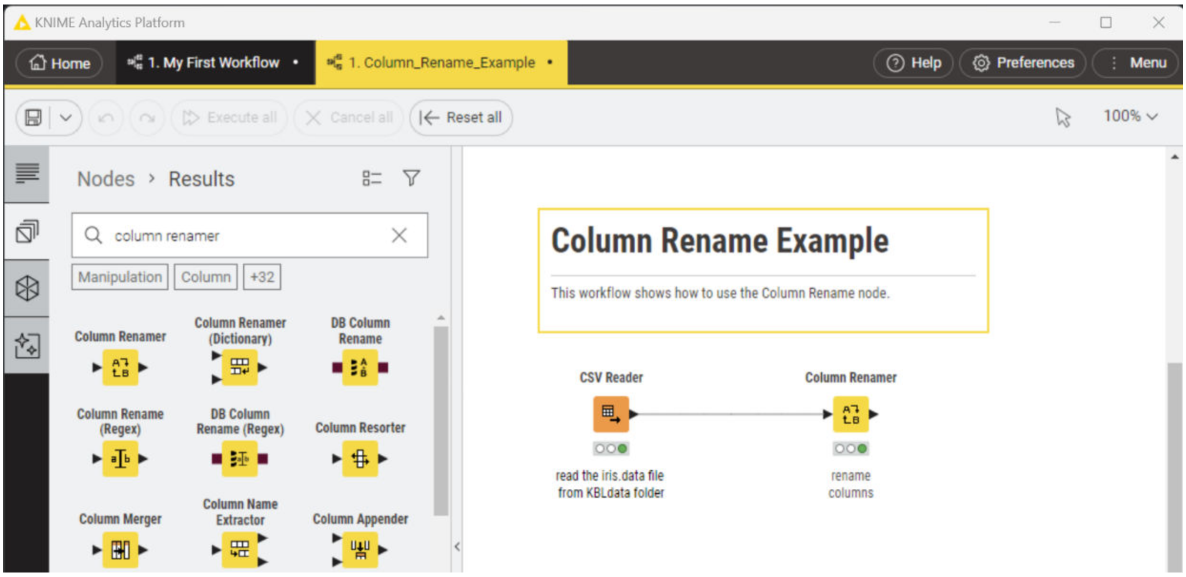
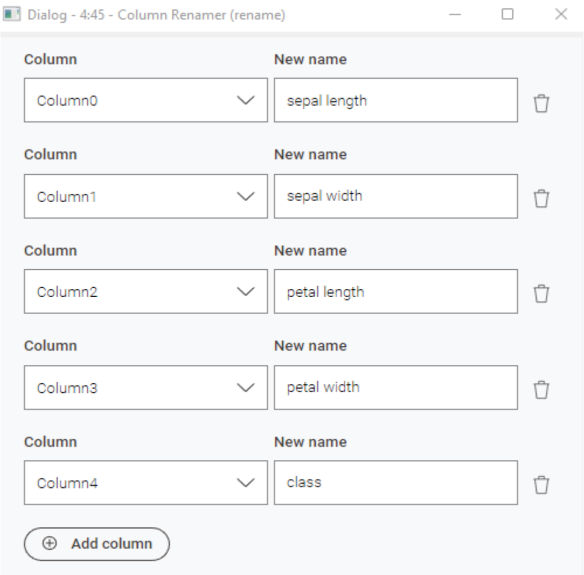


Figure 3.2. Create a Column Renamer node.

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


Dialog - 4:45 - Column Renamer (rename)

Column	New name
Column0	sepal length
Column1	sepal width
Column2	petal length
Column3	petal width
Column4	class

Figure 3.3. Configuration dialog of the Column Renamer node. The button with "Trash" symbol is there to remove an already selected column from the renaming panel.

We created a *Column Renamer* node, and we connected it to the *CSV Reader* node. We then assigned new names to the data table columns according to the "Iris.name" specification file and we ran the "Execute" command. The same operation could have been executed in the **Transformation tab** of the *CSV Reader* node. This is what we did in the second workflow of this chapter "Write To DB".

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, you can 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 “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.

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

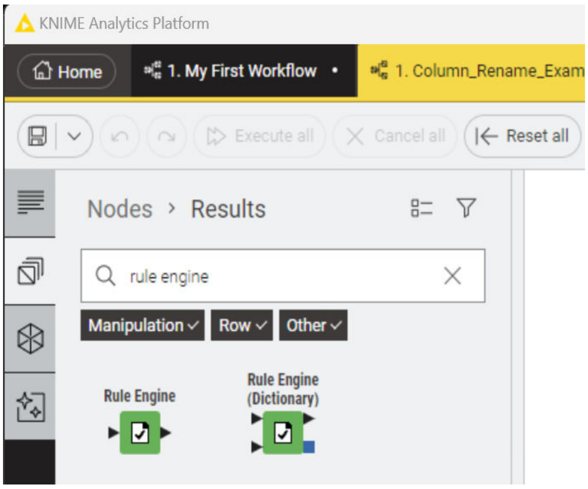
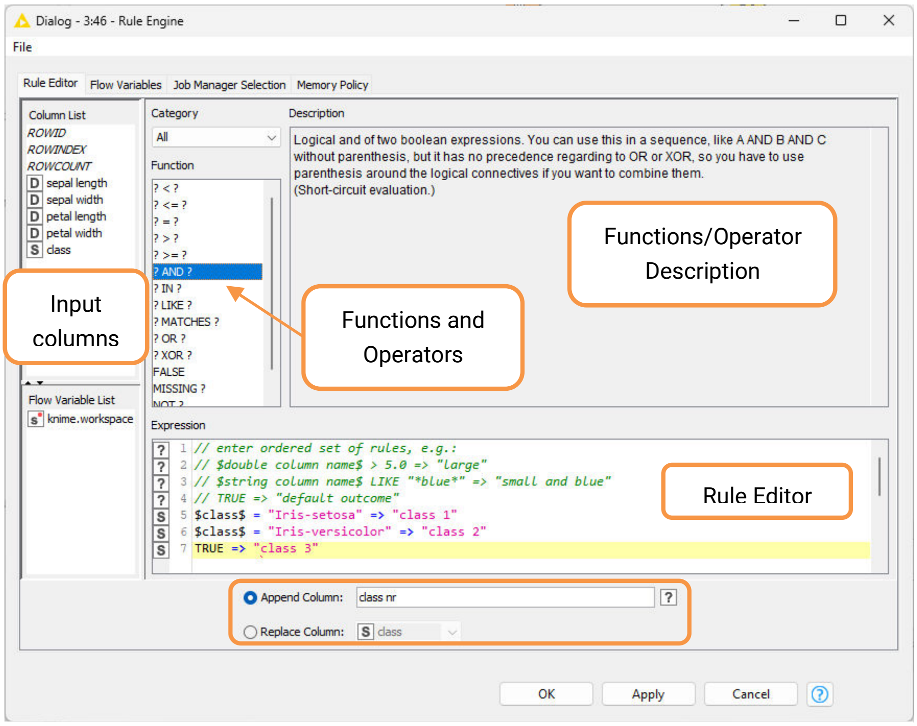


Figure 3.4. Location of the "Rule Engine" node in the "Node Repository" panel.

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.



Logical and of two boolean expressions. You can use this in a sequence, like A AND B AND C without parenthesis, but it has no precedence regarding to OR or XOR, so you have to use parenthesis around the logical connectives if you want to combine them. (Short-circuit evaluation.)

```
// enter ordered set of rules, e.g.:
// $double column name$ > 5.0 => "large"
// $string column name$ LIKE "*blue*" => "small and blue"
// TRUE => "default outcome"
$class$ = "Iris-setosa" => "class 1"
$class$ = "Iris-versicolor" => "class 2"
TRUE => "class 3"
```

● Append Column: class nr

○ Replace Column: S class

Figure 3.5. The configuration window of the "Rule Engine" node.

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 ran 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: "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 detail how to use the three "Cell Splitter" nodes to do that.

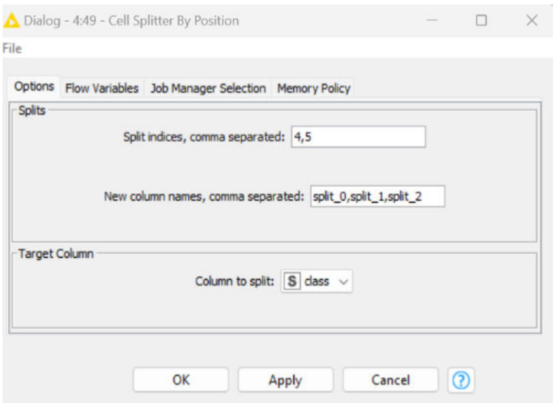
Cell Splitter by Position

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.

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"



Split indices, comma separated: 4,5

New column names, comma separated: split_0,split_1,split_2

Column to split: S class

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

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)

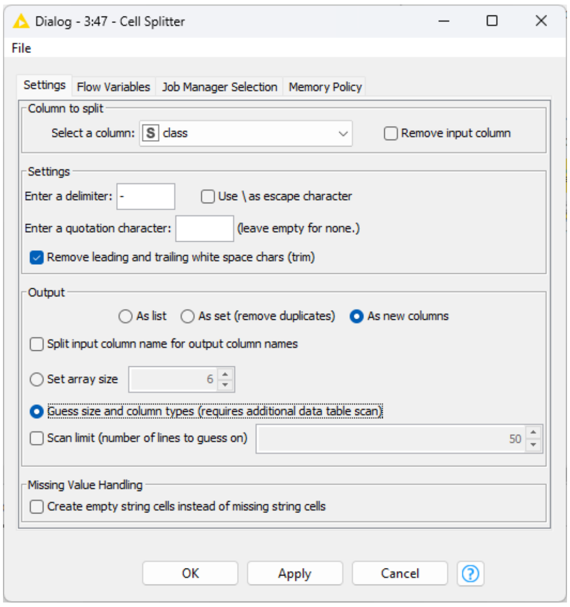
If many splits are forecasted, the first option can quickly add too many new columns to the output data set and become unmanageable. On the opposite, the second option adds only one additional column to the output data set, compacting all substrings into a collection type column.

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.

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 col umns 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]”.



Column to split

Select a column: S class ☐ Remove input column

Settings

Enter a delimiter: - ☐ Use \ as escape character

Enter a quotation character: (leave empty for none.)

☑ Remove leading and trailing white space chars (trim)

Output

○ As list ○ As set (remove duplicates) ● As new columns

☐ Split input column name for output column names

○ Set array size 6

● Guess size and column types (requires additional data table scan)

☐ Scan limit (number of lines to guess on) 50

Missing Value Handling

☐ Create empty string cells instead of missing string cells

Figure 3.7. Configuration dialog for the Cell Splitter node.

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

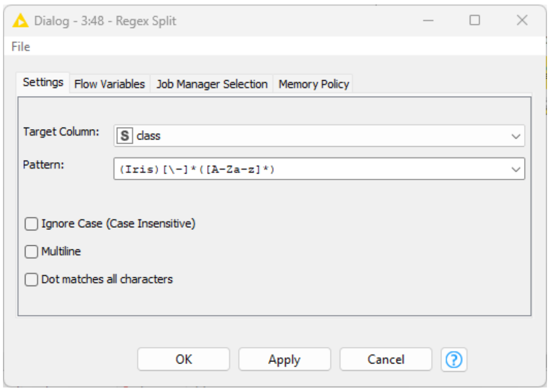
RegEx Split (= Cell Splitter by RegEx)

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

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”.



Dialog - 3:48 - Regex Split

Target Column: S class

Pattern: (Iris)[\-]*([A-Za-z]*)

☐ Ignore Case (Case Insensitive)

☐ Multiline

☐ Dot matches all characters

Figure 3.8. Configuration dialog for the RegEx Split node.

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”.

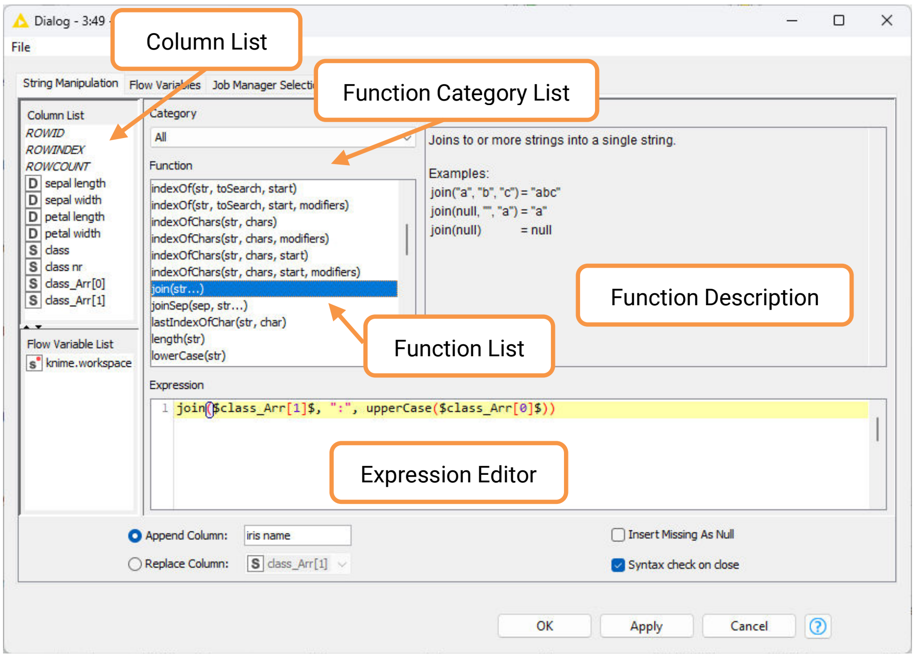
3.4. String Manipulation

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.



Joins to or more strings into a single string.

Examples:

```
join("a", "b", "c") = "abc"
join(null, "", "a") = "a"
join(null)          = null
```

```
join($class_Arr[1]$, ":", upperCase($class_Arr[0]$))
```

● Append Column: iris name

○ Replace Column: S class_Arr[1]

□ Insert Missing As Null

☑ Syntax check on close

Figure 3.9. Configuration dialog for the String Manipulation 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.

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”. Notice that the splitting of the original string into the substrings contained in class_Arr[] could have also been obtained inside the String Manipulation node using a substr() function.

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.

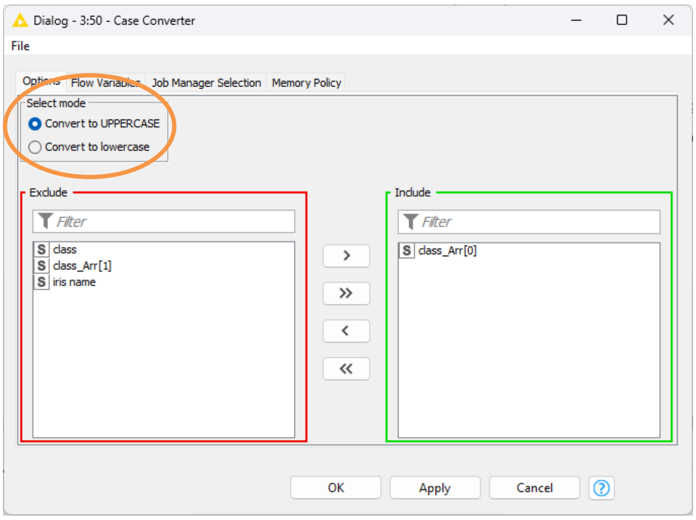
The String Manipulation node is particularly useful when we want to combine a number of different string functions into a single more complex one. However, an alternative processing uses a sequence of single dedicated nodes. This approach leads to a more crowded workflow, but it provides an easier interpretation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

To switch from lower to upper case or vice versa, the “Case Converter” node in the category “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"



Select mode

● Convert to UPPERCASE

○ Convert to lowercase

Exclude

S class

S class_Arr[1]

S iris name

Include

S class_Arr[0]

Figure 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 "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.

The corresponding function in the String Manipulation node would be *upperCase()*.

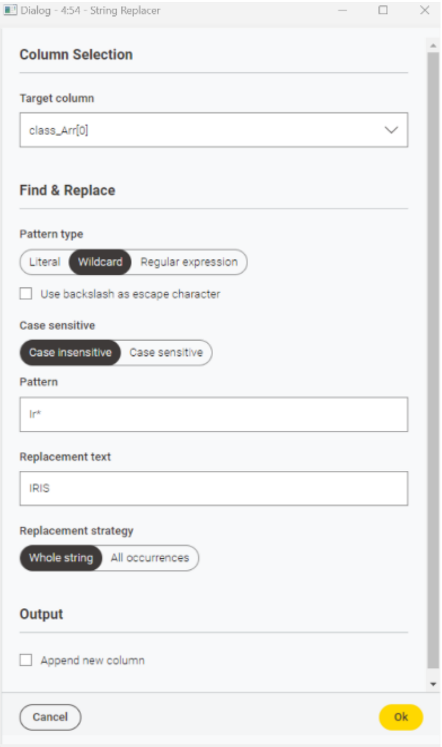
String Replacer

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



Dialog - 4:54 - String Replacer

Column Selection

Target column: class_Arr[0]

Find & Replace

Pattern type: Literal | [✓ Wildcard] | Regular expression

□ Use backslash as escape character

Case sensitive: [✓ Case insensitive] | Case sensitive

Pattern: Ir*

Replacement text: IRIS

Replacement strategy: [✓ Whole string] | All occurrences

Output

□ Append new column

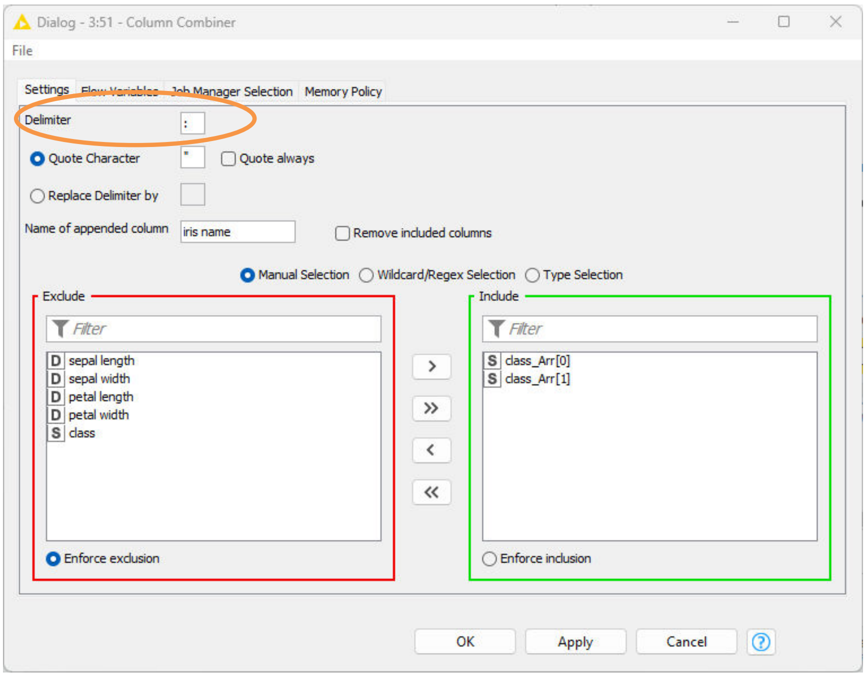
Figure 3.11. Configuration window for the String Replacer node.

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 "Manipulation" → "Column" → "Split & Combine".

The corresponding function in the String Manipulation node would be *replace()*.



Delimiter :

● Quote Character " □ Quote always

○ Replace Delimiter by

Name of appended column iris name □ Remove included columns

● Manual Selection ○ Wildcard/Regex Selection ○ Type Selection

Exclude
- D sepal length
- D sepal width
- D petal length
- D petal width
- S class

● Enforce exclusion

Include
- S class_Arr[0]
- S class_Arr[1]

○ Enforce inclusion

Figure 3.12. Configuration window for the Column Combiner node.

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”.

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

The corresponding function in the String Manipulation node would be *join()*.

> ***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 “Manipulation” → “Column” → “Transform”.

Column Resorter

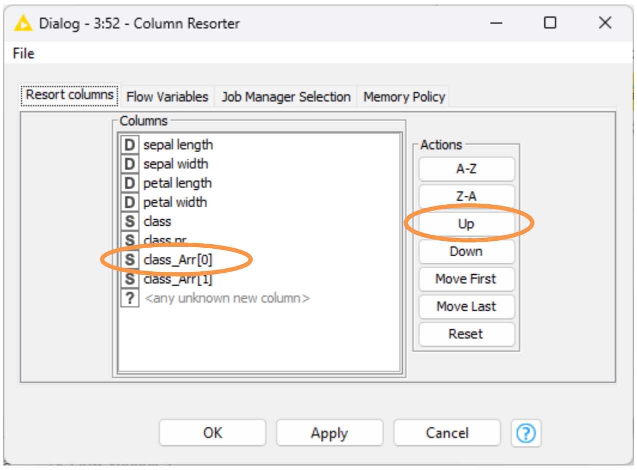
The “Column Resorter” node changes the order of the columns in the input data table.

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.

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”.



D sepal length
D sepal width
D petal length
D petal width
S class
S class nr
S class_Arr[0]
S class_Arr[1]
? <any unknown new column>

Figure 3.13. The configuration window of the Column Resorter node.

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

3.5. Type Conversion

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: “Manipulation” → “Column” → “Convert & Replace”.

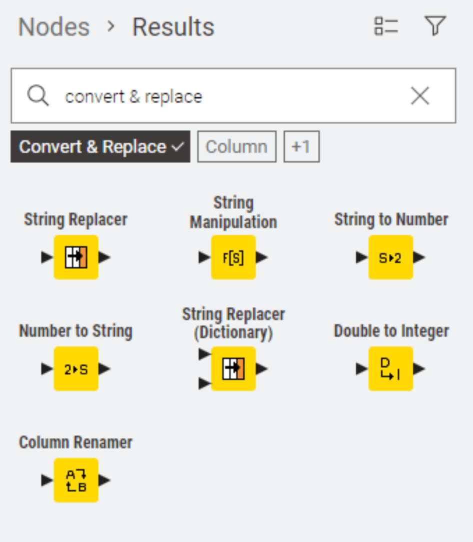


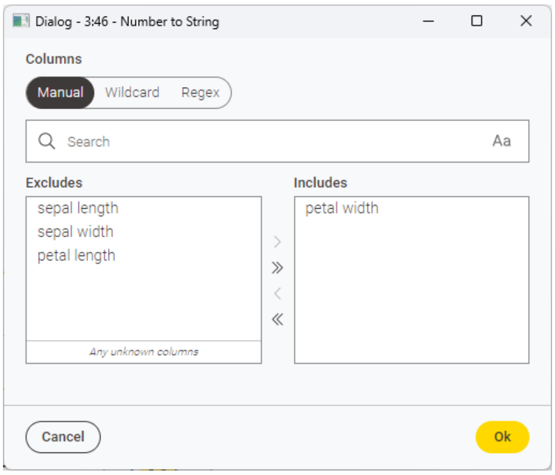
Figure 3.14. Location of the Type Conversion node in the Node Repository panel.

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.

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 "Includes". All other columns are listed in the frame "Excludes".
- To move from frame "Includes" to frame "Excludes" and vice versa, double click the column name or select the column and then click on arrows in between the two windows.
- The "Number to String" node is equivalent to the function *string()* in the String Manipulation node.



Dialog - 3:46 - Number to String

Columns

Manual | Wildcard | Regex

Excludes: sepal length, sepal width, petal length

Includes: petal width

Figure 3.15. Configuration window of the Number to String node.

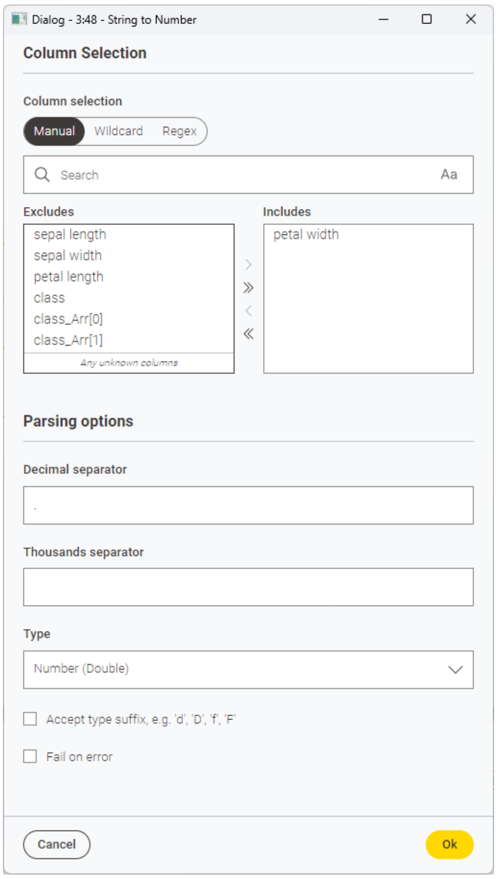
We inserted only the column "petal width" in the frame "Include" to be converted from type Double to type String.

Now for demonstration's 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's 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



Dialog - 3:48 - String to Number

Column Selection

Column selection: Manual

Excludes: sepal length, sepal width, petal length, class, class_Arr[0], class_Arr[1]

Includes: petal width

Parsing options

Decimal separator: .

Thousands separator:

Type: Number (Double)

□ Accept type suffix, e.g. 'd', 'D', 'f', 'F'

□ Fail on error

Figure 3.16. Configuration dialog of the String to Number node.

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 "Includes". All other columns are listed in the frame "Excludes".
- To move from frame "Includes" to frame "Excludes" and vice versa, double click the column name or select the column and then click on arrows in between the two windows.
- The "String to Number" node is equivalent to *toInt()*, *toDouble()*, *toLong()*, and similar functions in the "String Manipulation" node.

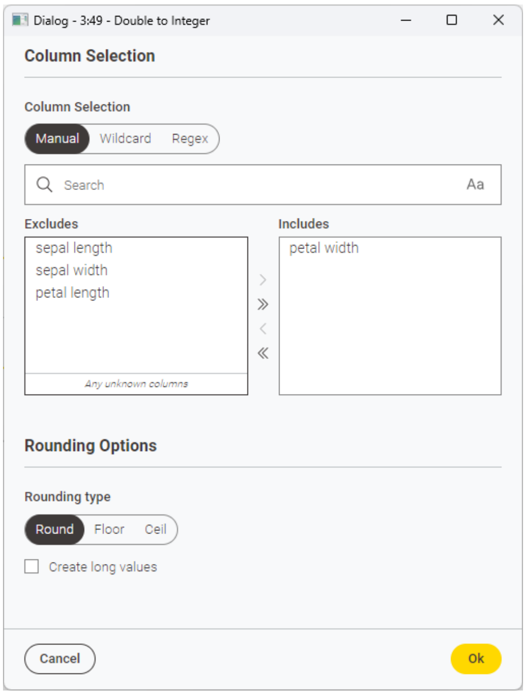
Let's still suppose, for the sake of nodes demonstration, that we have converted the "petal width" array columns to type Double, but that actually we wanted to have them of type Int. Let's ignore the fact that it would be enough to change option "Type" in the configuration window of the "String To Number"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

Double to Integer

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 “Includes”. All other columns are listed in frame “Excludes”.

To move from frame “Includes” to frame “Excludes” and vice versa, double click the column name or select the column and then click on arrows in between the two windows.



Dialog - 3:49 - Double to Integer

Column Selection

Column Selection

Manual | Wildcard | Regex

Search Aa

Excludes

sepal length
sepal width
petal length

Any unknown columns

Includes

petal width

Rounding Options

Rounding type

Round | Floor | Ceil

□ Create long values

Cancel | Ok

Figure 3.17. Configuration window of the Double to Integer node.

> ***Note.*** A node that covers many of the conversion operations listed in this section is the “Column Auto Type Cast”. This node scans a column to automatically define its data type. A quick scan is also possible, faster but riskier.

3.6. Database Operations

We have only shown 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 functions in the *String Manipulation* node would have been sufficient. In the next workflow for visualization, we will use indeed just the data produced by the *String Manipulation* node.

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 “DB” containing all nodes that perform operations on databases.

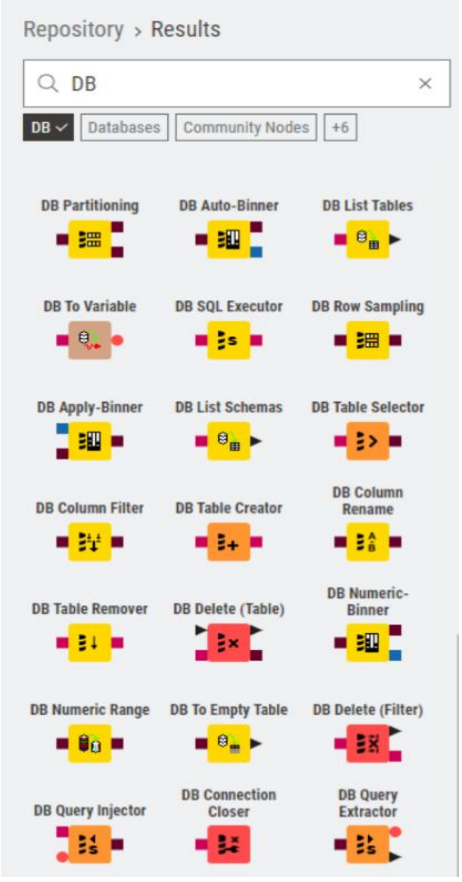


Figure 3.18. "DB" category in the Node Repository.

In order to access a database with KNIME Analytics Platform we first establish a connection to the database with a connector node and along that connection we use a *DB Writer* or a *DB Table Selector* followed by a *DB Reader* node to write or read the data table.

The connector nodes that KNIME provides to access databases come with pre-loaded JDBC drivers. The connector nodes cover the most commonly used and most recent database versions, such as MySQL, SQLite, Vertica, Hadoop Hive, H2, PostgreSQL, and more.

If the connector node for your database is not available, you can always use a generic DB Connector node. Here you will need to provide the driver file for your database. If this is not already in the list of pre-loaded driver files, you can always add it via the "File" → "Preferences" → "Databases" page (see later on in this chapter).

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.

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 the DB Writer node.

Under category "DB"/"Connector" you find a number of connector nodes to establish a connection to a database. Some of those nodes are dedicated connectors, which means they contain a pre-loaded JDBC driver file and present a customized User Interface. Only the "DB

Connector" node is a generic connector, to be used when the dedicated connector for the database of choice is not available.

SQLite Connector

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 already pre-loaded.

In the configuration window of all connector nodes, there are five tabs: "Connection Settings", "JDBC Parameters", "Advanced", "Input Type Mapping", "Output Type Mapping".

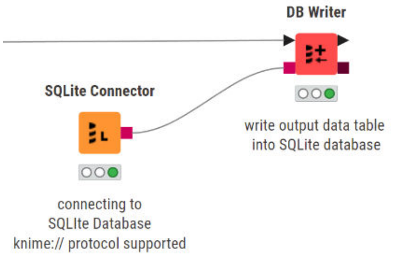
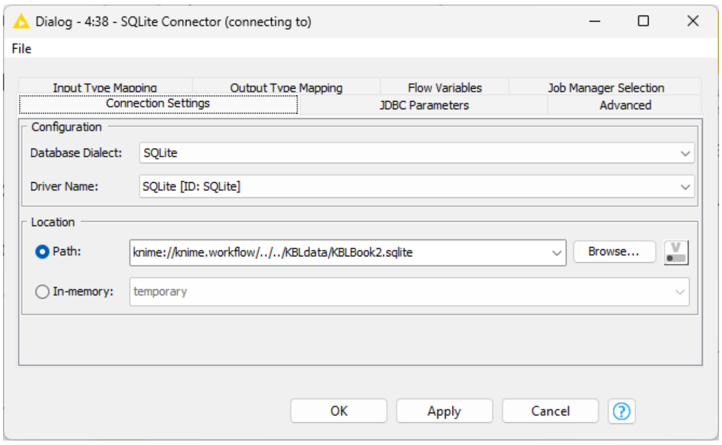


Figure 3.19. SQLite Connector node + DB Writer node.

- "Connection Settings" contains all necessary settings to connect to the database: JDBC driver, hostname, port, database name, and full credentials where required.
- Tabs "JDBC Parameters" and "Advanced" allow you to set specific commands to connect to the database; while tabs "Input Type Mapping" and "Output Type Mapping" allow to correctly map all data types from KNIME to the database and viceversa.



Database Dialect: SQLite

Driver Name: SQLite [ID: SQLite]

● Path: knime://knime.workflow/../../KBLdata/KBLBook2.sqlite

○ In-memory: temporary

Figure 3.20. SQLite Connector node: "Connection Settings" tab in the configuration window.

Note. Did you notice the full red square as input port? So far, we have seen only black triangles as input or output ports. A black 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 types of ports, each one exporting or importing a different kind of object.

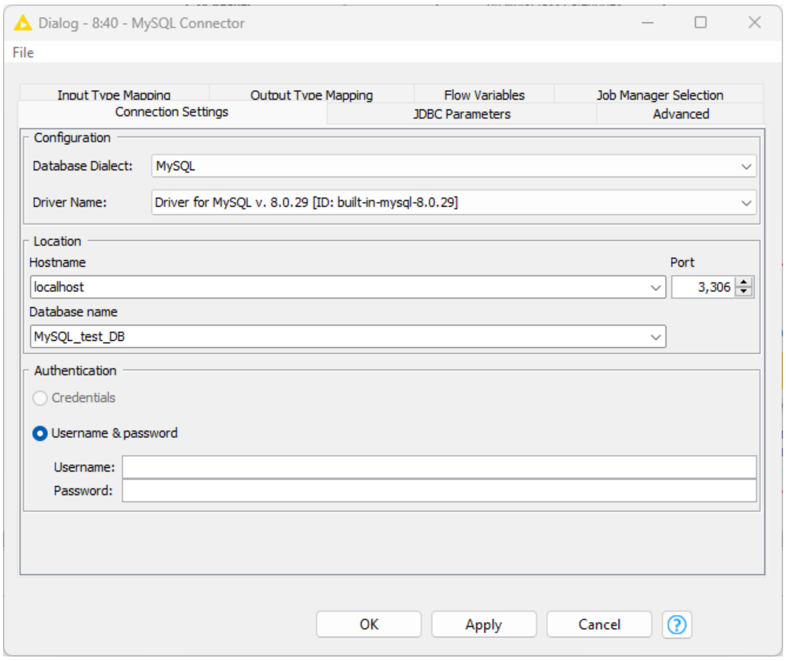
The SQLite database is a simple file-based database, requiring no credentials, which leads to a very simple configuration window. Let's check the configuration window of the connector node to a more complex database: the MySQL Connector node.

MySQL Connector

The MySQL Connector node connects to a MySQL database and requires:

- The database driver (pre-loaded)
- The hostname and database name
- The username and password for the authentication

Credentials can be supplied as username and password by enabling the option "Username & password". Another option is to define them as credentials at the workflow level.



Database Dialect: MySQL

Driver Name: Driver for MySQL v. 8.0.29 [ID: built-in-mysql-8.0.29]

Hostname: localhost

Port: 3,306

Database name: MySQL_test_DB

○ Credentials

● Username & password

Username:

Password:

Figure 3.21. MySQL Connector node: "Connection Setting" tab.

The other tabs “JDBC Parameters”, “Advanced”, and “Mappings” include the same functionalities as for the SQLite Connector node.

Credentials at the workflow level are automatically encrypted and therefore more secure. Username and password provided directly into the configuration window are not automatically encrypted and require an extra step for security: a master key. This master key will then be used to encrypt usernames and passwords when provided into configuration windows.

DB Writer

The node “DB Writer”, located in the “DB”/”Read/Write” category, writes the input data table into a database table. If the table does not exist, it is created. If the table already exists, column names in the KNIME data have to match exactly the column names in the table. The only required settings are:

- The name and optionally schema of the table in the database. The button “Select a table” allows you to browse the database content.
- The size of the batch of data to write at each time
- The columns to transfer into the database from the KNIME data through an Include/Exclude frame
- The flag for failure in case of error
- The flag to append the status of the writing operation for each data rows
- The flag “Remove existing table” allows to write in “Append” mode or in “Overwrite” mode
- The “Output Type Mapping” tab contains the specifications on the mapping of the KNIME data types on the database data types.

In order to write the processed iris data to the KBLBook.sqlite database, a “DB Writer” node was connected to the output port of the “String Manipulation” node named “build <attr>:IRIS”. In the configuration window of the “DB Writer” node, we set the data columns that we want to transfer from KNIME into the SQLite database.

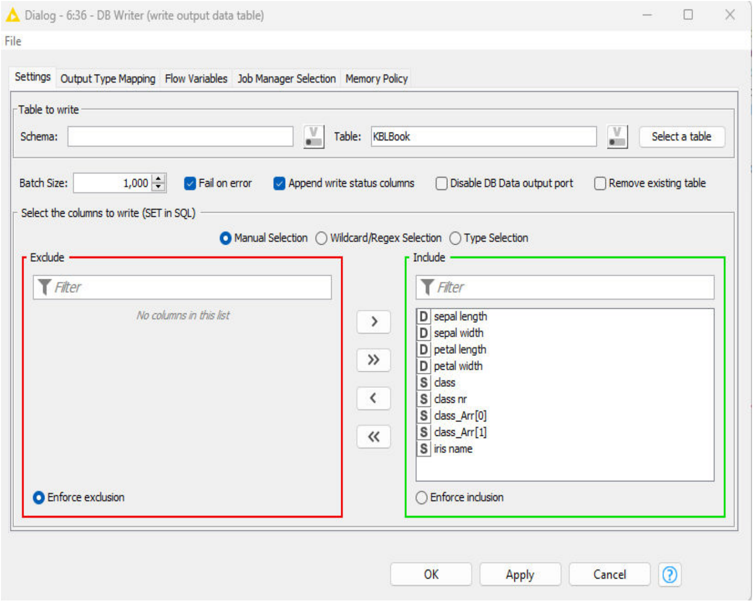


Table to write

Schema:

Table: KBLBook

Batch Size: 1,000

☑ Fail on error ☑ Append write status columns □ Disable DB Data output port □ Remove existing table

Select the columns to write (SET in SQL)

● Manual Selection ○ Wildcard/Regex Selection ○ Type Selection

Exclude

Filter

No columns in this list

● Enforce exclusion

Include

Filter

D sepal length
D sepal width
D petal length
D petal width
S class
S class nr
S class_Arr[0]
S class_Arr[1]
S iris name

○ Enforce inclusion

Figure 3.22. Configuration of the DB Writer node.

Import a JDBC Database Driver

The JDBC drivers for the most common and recent databases are already pre-loaded and available in the connector 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.

In the Top Menu, select “Settings” (beside the information(i) button). 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.

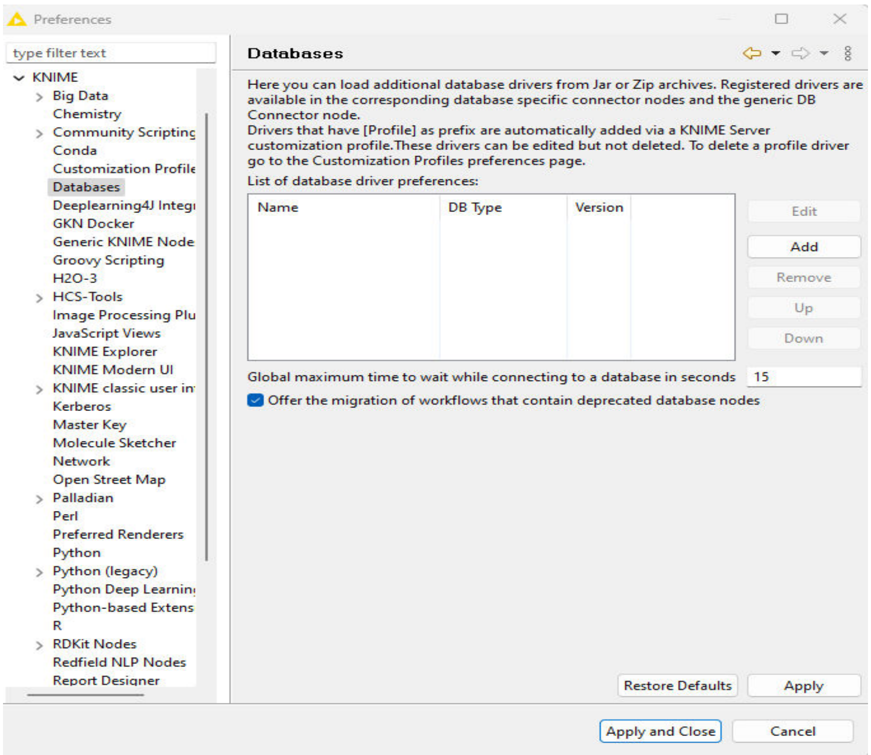
- Expand item “KNIME”
- Select sub-item “Databases”

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.



Here you can load additional database drivers from Jar or Zip archives. Registered drivers are available in the corresponding database specific connector nodes and the generic DB Connector node.
Drivers that have [Profile] as prefix are automatically added via a KNIME Server customization profile.These drivers can be edited but not deleted. To delete a profile driver go to the Customization Profiles preferences page.
List of database driver preferences:

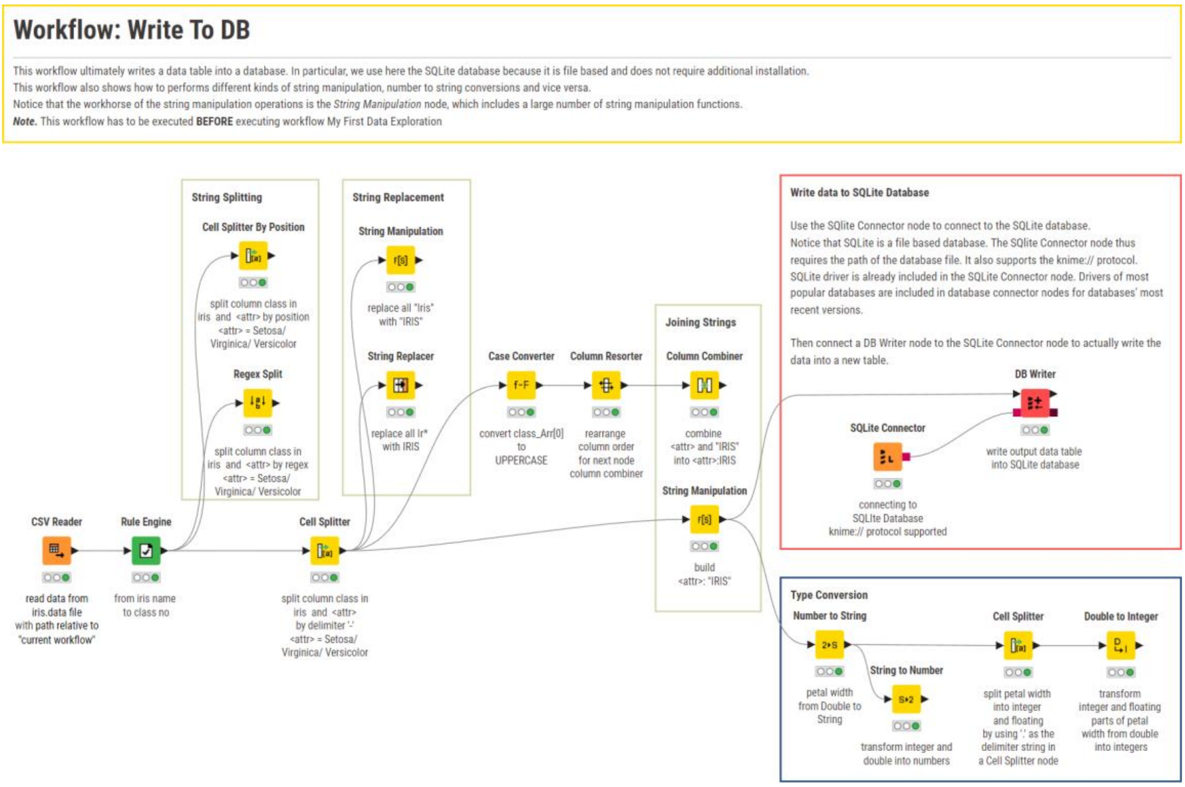
Global maximum time to wait while connecting to a database in seconds: 15

☑ Offer the migration of workflows that contain deprecated database nodes

Figure 3.23. The "Database Driver" page under "Preferences".

Let’s now read from the SQLite database the data we have just written, to perform some visual data exploration. Next to the “DB Writer” node in the “Node Repository” panel, we find the “DB 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 establish the connection with the database with a connector node (in this case a “SQLIte Connector” node), the select the database table to read from with a DB Table Selector node, and then read the data with a “DB Reader” node.



Workflow: Write To DB

This workflow ultimately writes a data table into a database. In particular, we use here the SQLite database because it is file based and does not require additional installation.
This workflow also shows how to performs different kinds of string manipulation, number to string conversions and vice versa.
Notice that the workhorse of the string manipulation operations is the *String Manipulation* node, which includes a large number of string manipulation functions.
Note. This workflow has to be executed **BEFORE** executing workflow My First Data Exploration

Write data to SQLite Database

Use the SQlite Connector node to connect to the SQLite database.
Notice that SQLite is a file based database. The SQlite Connector node thus requires the path of the database file. It also supports the knime:// protocol.
SQLite driver is already included in the SQLite Connector node. Drivers of most popular databases are included in database connector nodes for databases' most recent versions.

Then connect a DB Writer node to the SQLite Connector node to actually write the data into a new table.

String Splitting
Cell Splitter By Position
split column class in iris and <attr> by position <attr> = Setosa/ Virginica/ Versicolor
Regex Split
split column class in iris and <attr> by regex <attr> = Setosa/ Virginica/ Versicolor
String Replacement
String Manipulation
replace all "Iris" with "IRIS"
String Replacer
replace all Ir* with IRIS
CSV Reader
read data from iris.data file with path relative to "current workflow"
Rule Engine
from iris name to class no
Cell Splitter
split column class in iris and <attr> by delimiter '-' <attr> = Setosa/ Virginica/ Versicolor
Case Converter
convert class_Arr[0] to UPPERCASE
Column Resorter
rearrange column order for next node column combiner
Joining Strings
Column Combiner
combine <attr> and "IRIS" into <attr>:IRIS
String Manipulation
build <attr>: "IRIS"
DB Writer
write output data table into SQLite database
SQLite Connector
connecting to SQLIte Database knime:// protocol supported
Type Conversion
Number to String
petal width from Double to String
String to Number
transform integer and double into numbers
Cell Splitter
split petal width into integer and floating by using '.' as the delimiter string in a Cell Splitter node
Double to Integer
transform integer and floating parts of petal width from double into integers

Figure 3.24. Workflow "Write To DB".

DB Table Selector

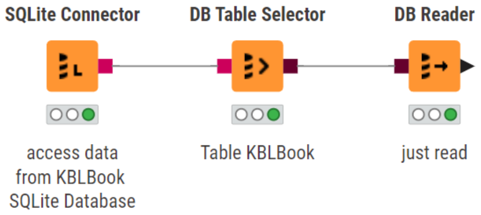
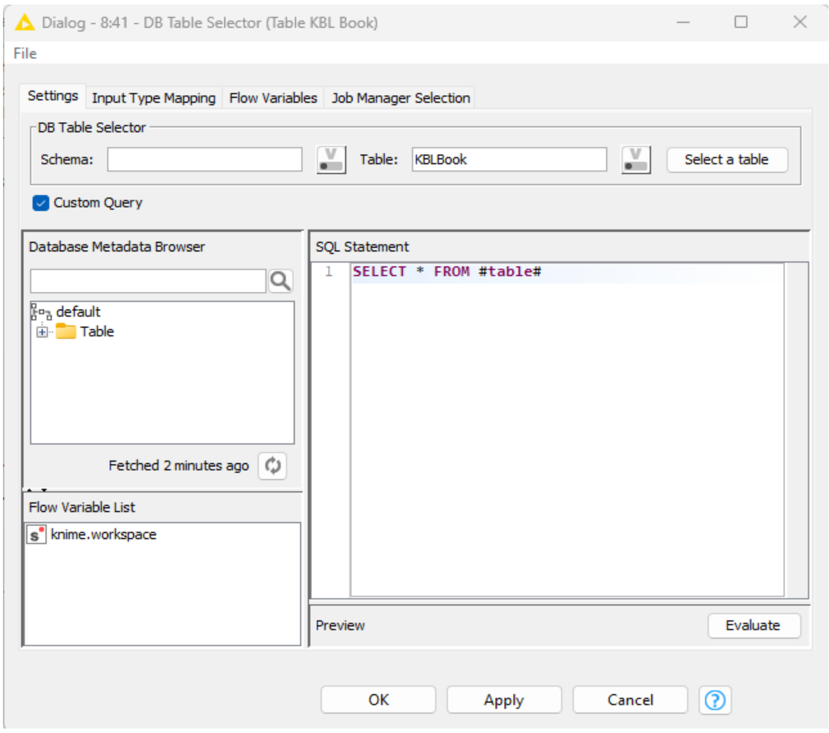


Figure 3.25. SQLite Connector node + DB Table Selector node + Database Reader node to read data from an SQLite database.

The node "DB Table Selector", located in "DB"/"Query", selects a table from the database provided with the database connection at its input port. Configuration settings require:

- The name and optionally schema of the database table. The "Select a table" button allows you to browse the content of the database to select the desired table.
- The default query is "SELECT * FROM #table#" where #table# is the selected table, which includes all data rows and all data columns of the table.
- It is also possible to extract a part or a transformation of the original #table#, by creating a custom query. The query editor appears when the flag "Custom Query" is selected.

- In the SQL editor (if available) you can then write your customized query to extract the data from #table#



Schema:
Table: KBLBook
☑ Custom Query

```
SELECT * FROM #table#
```

Figure 3.26. Configuration of the Database Table Selector when following a Database Connector node.

After the SQLite Connector node, the DB Table Selector node selects the table named KBLBook inside the database using the default query. We will work on the data of this table from now on.

DB Reader

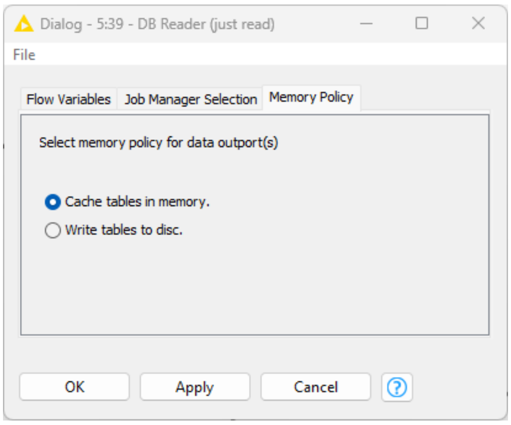
The node “DB Reader”, located in the “DB”/”Read/Write” category, reads a table from a database according to the input SQL query and imports it into a workflow.

Since the “DB Reader” node receives all information (connection to the database and SQL query) at the input port, it does not require any additional settings to run.

So, the only settings in the configuration window refer to the storage of the resulting data table.

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.



Select memory policy for data outport(s)

● Cache tables in memory.

○ Write tables to disc.

Figure 3.27. Configuration window of the DB Reader node.

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 structures 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 “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 group 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 relative 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 "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

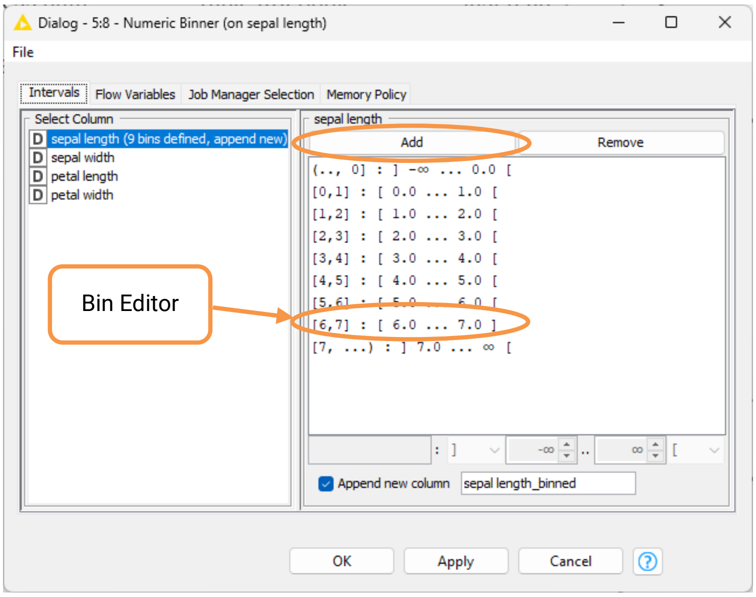
To define a new bin interval:

- 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.



```
(.., 0] : ] -∞ ... 0.0 [
[0,1] : [ 0.0 ... 1.0 [
[1,2] : [ 1.0 ... 2.0 [
[2,3] : [ 2.0 ... 3.0 [
[3,4] : [ 3.0 ... 4.0 [
[4,5] : [ 4.0 ... 5.0 [
[5,6] : [ 5.0 ... 6.0 [
[6,7] : [ 6.0 ... 7.0 ]
[7, ...) : ] 7.0 ... ∞ [
```

☑ Append new column: sepal length_binned

Figure 3.28. Configuration window for the Numeric Binner node.

Note. The 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

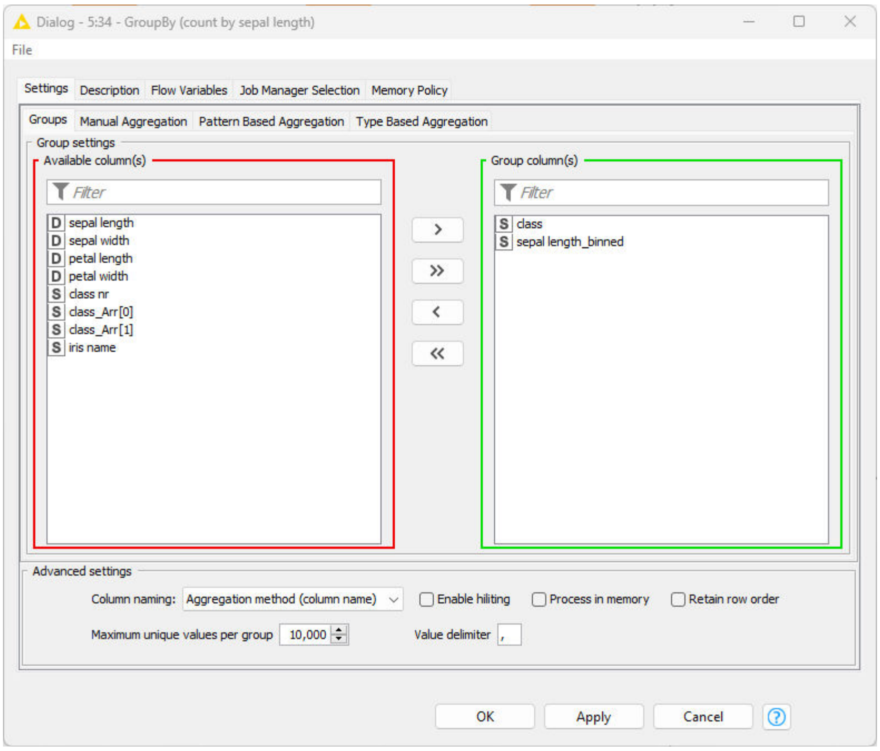
The "GroupBy" node finds groups of data rows by using the combination of values in one or more columns (Group Columns); it subsequently aggregates the values in other columns (Aggregation Columns) across those groups. Column values can be aggregated in the form of a sum, a mean, just a count of occurrences,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

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 hilting” refers to a feature available in the old “Data Views” node.



Dialog - 5:34 - GroupBy (count by sepal length)

Available column(s)

- D sepal length
- D sepal width
- D petal length
- D petal width
- S class nr
- S class_Arr[0]
- S class_Arr[1]
- S iris name

Group column(s)

- S class
- S sepal length_binned

Advanced settings

Column naming: Aggregation method (column name)   □ Enable hiliting   □ Process in memory   □ Retain row order

Maximum unique values per group: 10,000   Value delimiter: ,

Figure 3.29. Configuration window for the GroupBy node: the "Groups" tab.

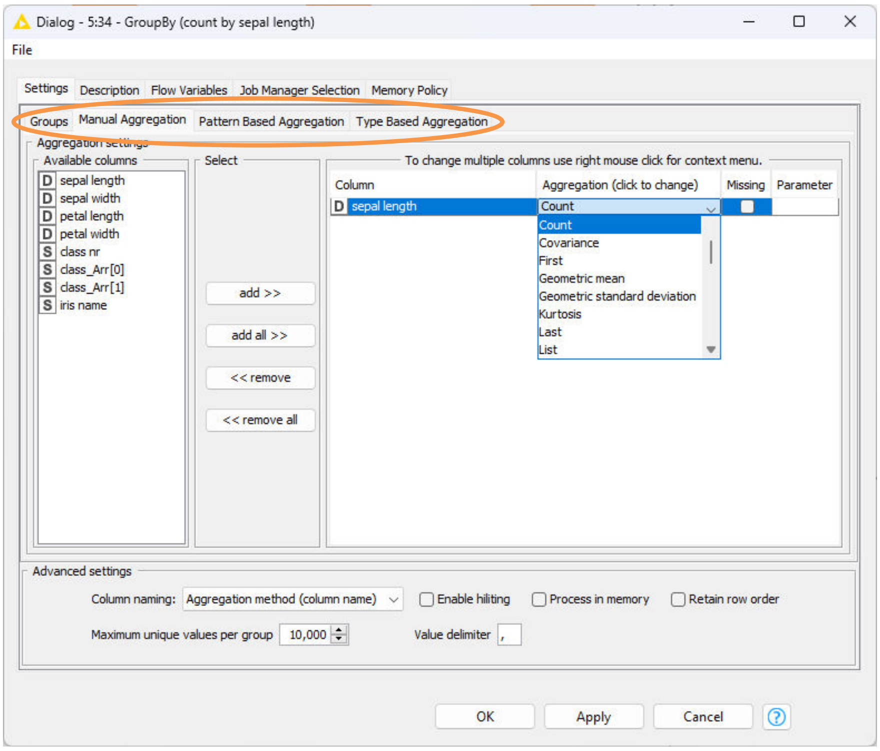
"Aggregation" Tabs

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, by clicking on add button or clicking on add all button to select all the columns 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



Dialog - 5:34 - GroupBy (count by sepal length)

To change multiple columns use right mouse click for context menu.

Column	Aggregation (click to change)	Missing	Parameter
D sepal length	Count	□	

Column naming: Aggregation method (column name)

□ Enable hiliting □ Process in memory □ Retain row order

Maximum unique values per group 10,000

Value delimiter ,

Figure 3.30. Configuration window for the GroupBy node: the "Manual Aggregation" tab.

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

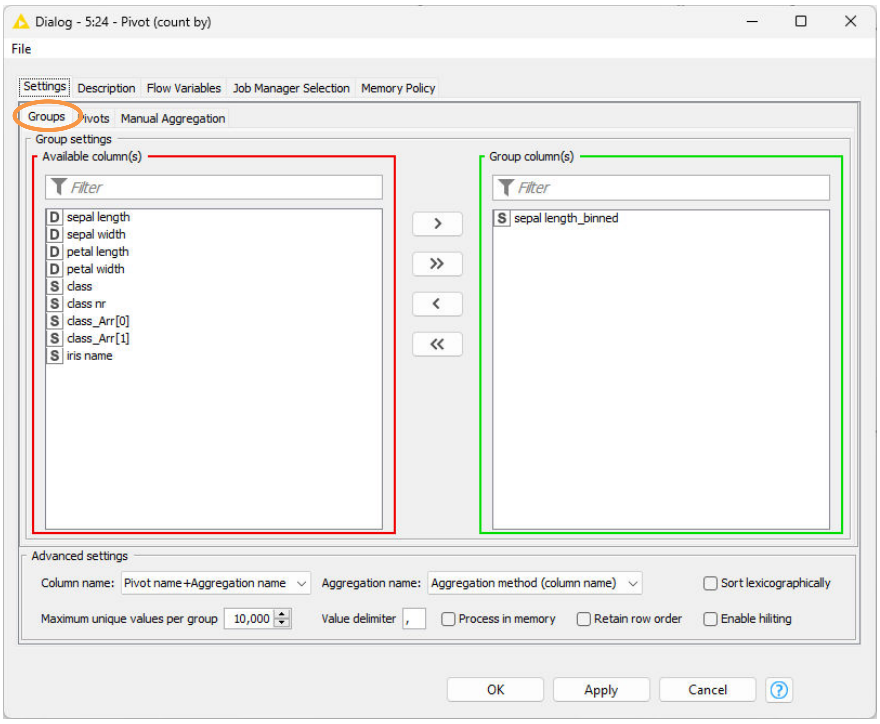
The “Pivoting” node is configured by means of three tabs: “**Groups**”, “**Pivots**”, and “**Manual Aggregation**”.

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



Group settings

Available column(s)

- D sepal length
- D sepal width
- D petal length
- D petal width
- S class
- S class nr
- S class_Arr[0]
- S class_Arr[1]
- S iris name

Group column(s)

- S sepal length_binned

Advanced settings

Column name: Pivot name+Aggregation name Aggregation name: Aggregation method (column name) □ Sort lexicographically

Maximum unique values per group 10,000 Value delimiter , □ Process in memory □ Retain row order □ Enable hiliting

Figure 3.31. Configuration window of the Pivoting node: the "Groups" tab.

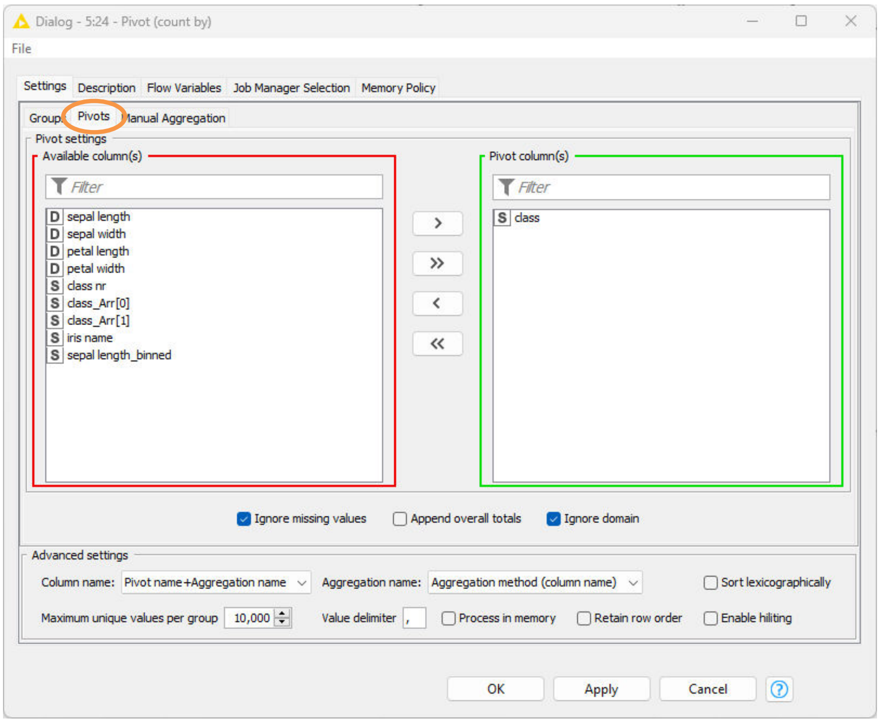
Tab “**Pivots**” defines the Pivot columns by means of an “Exclude”/”Include” frame:

- 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).



Dialog - 5:24 - Pivot (count by)

File

Settings | Description | Flow Variables | Job Manager Selection | Memory Policy

Groups | Pivots | Manual Aggregation

Pivot settings

Available column(s)

- D sepal length
- D sepal width
- D petal length
- D petal width
- S class nr
- S class_Arr[0]
- S class_Arr[1]
- S iris name
- S sepal length_binned

Pivot column(s)

- S class

☑ Ignore missing values ☐ Append overall totals ☑ Ignore domain

Advanced settings

Column name: Pivot name+Aggregation name
Aggregation name: Aggregation method (column name)
☐ Sort lexicographically

Maximum unique values per group 10,000
Value delimiter ,
☐ Process in memory ☐ Retain row order ☐ Enable hiliting

Figure 3.32. Configuration window of the Pivoting node: the "Pivots" tab.

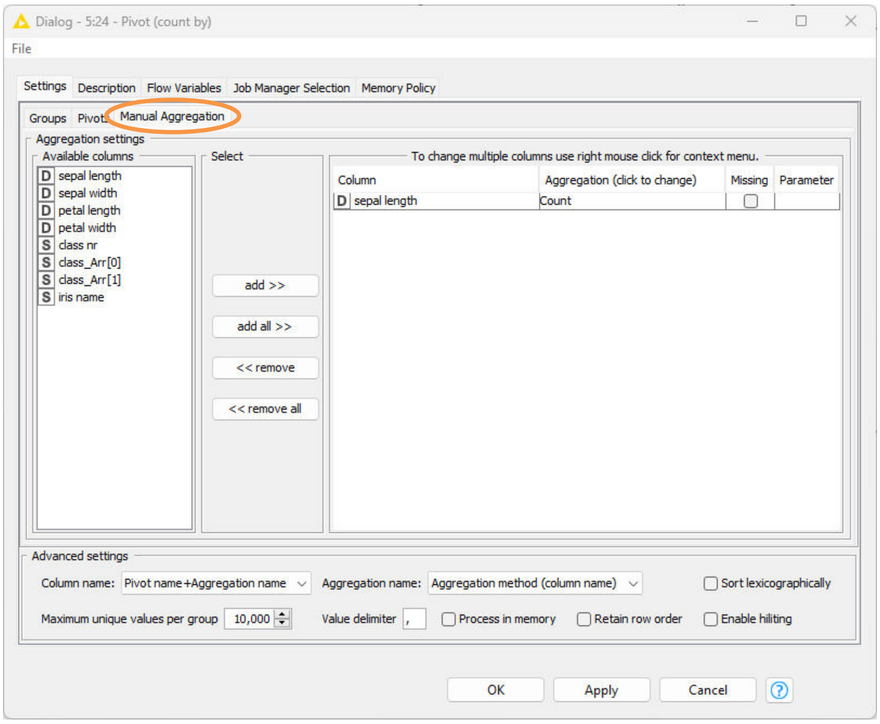
Tab “**Manual Aggregation**” selects the aggregation columns and the aggregation method for each aggregation column. The column selection is again performed by means of clicking on the arrow buttons between the two windows.

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 count 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



Column	Aggregation (click to change)	Missing	Parameter
D sepal length	Count	□	

Column name: Pivot name+Aggregation name

Aggregation name: Aggregation method (column name)

□ Sort lexicographically

Maximum unique values per group 10,000

Value delimiter ,

□ Process in memory

□ Retain row order

□ Enable hiliting

Figure 3.33. Configuration window of the Pivoting node: the "Manual Aggregation" tab.

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 the newest nodes in KNIME Analytics Platform 5. They are located in the 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. Only the new visualization nodes are compatible with the new reporting framework.

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.

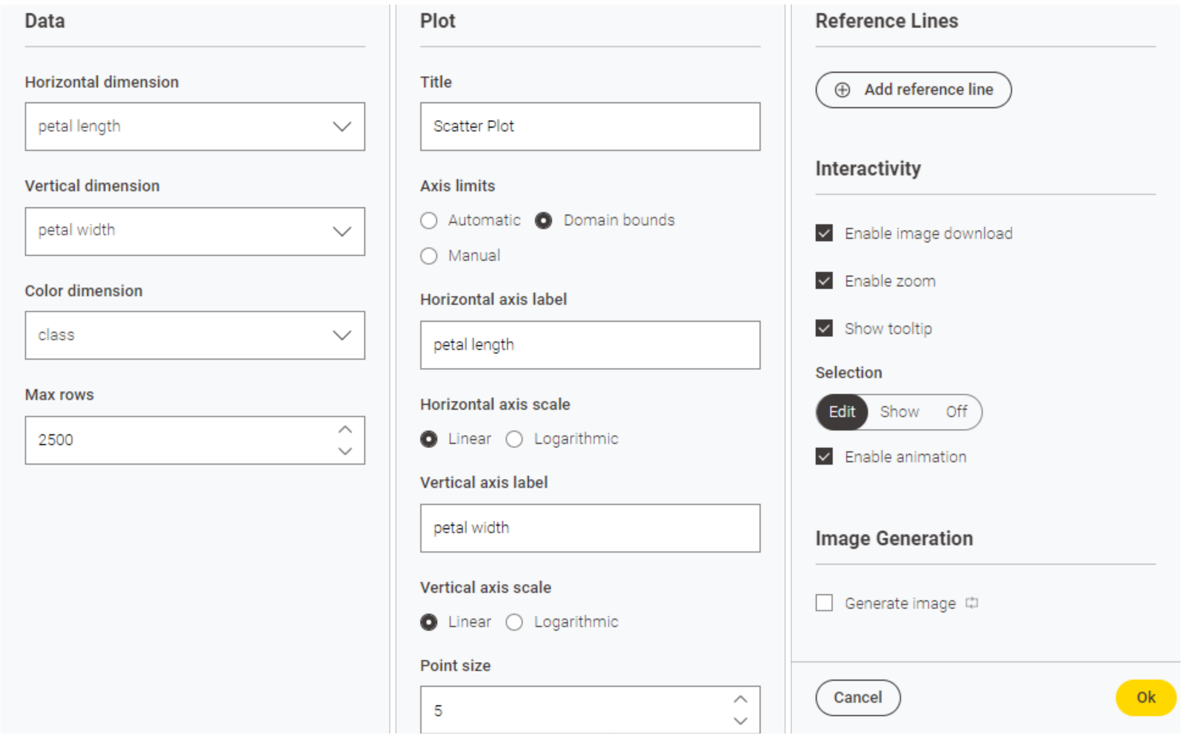
KNIME Analytics Platform also consists of the JavaScript based nodes. These nodes, located in "Views"/"JavaScript", are based on JavaScript graphical libraries. These nodes produce a data table and a static image. The output data table is a copy of the input data table plus a column containing the selection 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 using BIRT, but not with the new reporting framework.

Scatter Plot

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

The "Scatter Plot" 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" node covers 5 sections: "Data", "Plot", "Reference Lines", "Interactivity", and "Image Generation". The "Data" section defines columns to report to the x- and y-axis, columns to report on "color dimension", and the maximum number of data rows to be visualized. The "Plot" section specifies the image options, such as the title, the axes limits, the choice of selecting axes scales, and the axes labels. The "Reference Lines" section allows to plot a reference line at a specific value on the y-axis. This section allows to change the line's label, border style, color, and size. The "Interactivity" section defines the allowed interactivity on the final view, such as the possibility of downloading the image, zooming, information display on hovering, and selection of points. The "Image Generation" section is a checkbox to reproduce a view into an image at the output port.



Data

Horizontal dimension: petal length

Vertical dimension: petal width

Color dimension: class

Max rows: 2500

Plot

Title: Scatter Plot

Axis limits: ○ Automatic ● Domain bounds ○ Manual

Horizontal axis label: petal length

Horizontal axis scale: ● Linear ○ Logarithmic

Vertical axis label: petal width

Vertical axis scale: ● Linear ○ Logarithmic

Point size: 5

Reference Lines

Interactivity

☑ Enable image download

☑ Enable zoom

☑ Show tooltip

Selection: [✓ Edit] Show Off

☑ Enable animation

Image Generation

□ Generate image

Figure 3.34. Configuration window of the Scatter Plot node

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 "Interactivity" section of the node configuration window. Let's explore this view and let's see the kind of interactivity it allows.

The view of the "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.

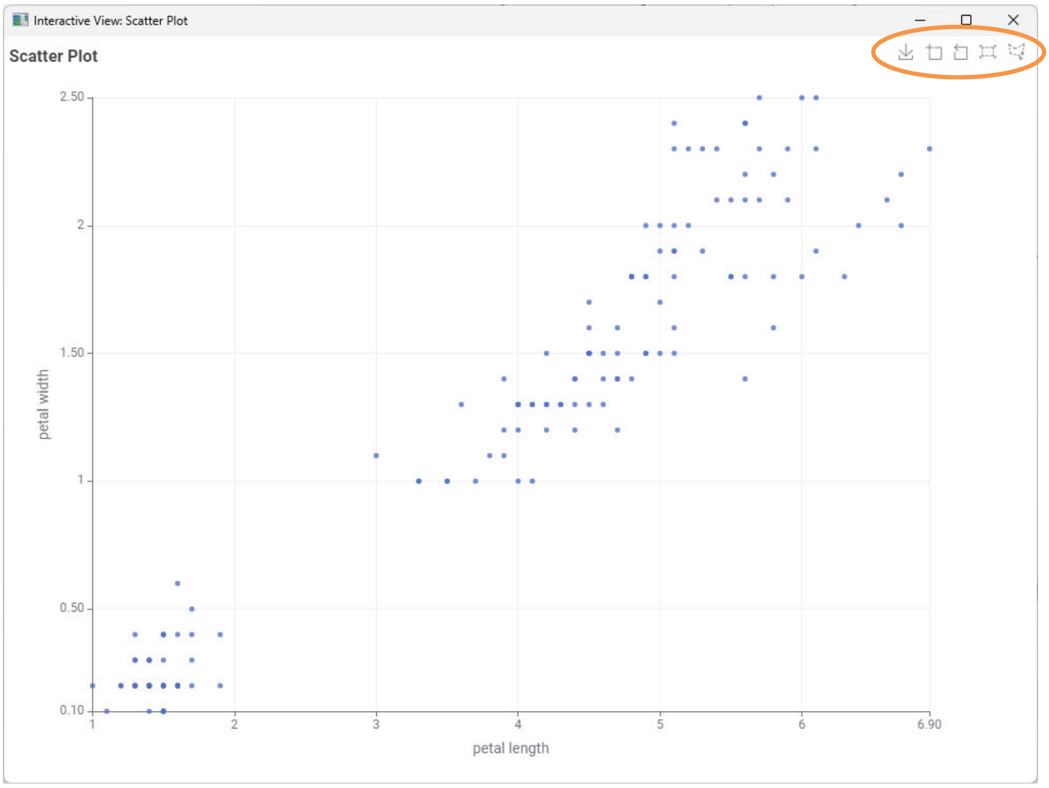
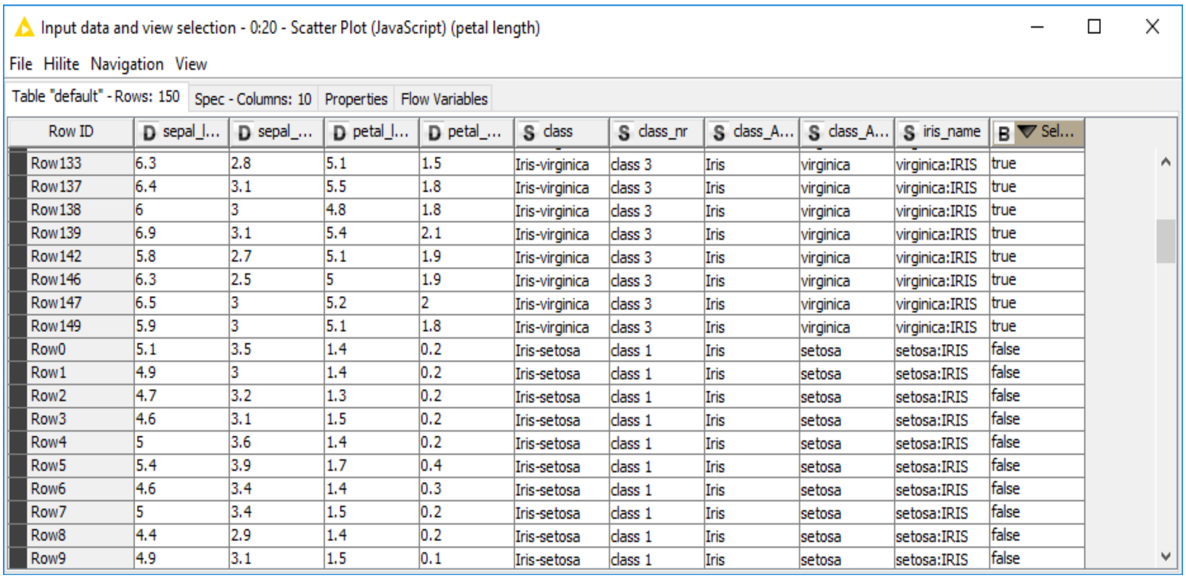


Figure 3.35. The Interactive View of the Scatter Plot node.

Scatter Plot: Interactive View

This is the view of the *Scatter Plot* node, where you can see the dots of the scatter plot. There are five buttons in the upper right corner. Those are the interactivity buttons. There are five buttons in the upper right corner. Those are the interactivity buttons. Starting from the far left, the "Save Image" button lets you download the image locally. The second button, the "Zoom" button, zooms in when you click on a specific point in the plot. The third button, the "Zoom reset" button, resets the zoomed-in part to the default view. The fourth button from the right, "Box select", allows a box-shaped selection on the plot. The fifth button is the "Lasso select" button, which allows us to select specific parts in a freeform. Additionally, scrolling back and forth on the plot in the interactive view lets you zoom in and out.

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.



Input data and view selection - 0:20 - Scatter Plot (JavaScript) (petal length)

File Hilite Navigation View

Table "default" - Rows: 150 | Spec - Columns: 10 | Properties | Flow Variables

Row ID	D sepal_l...	D sepal_...	D petal_l...	D petal_...	S class	S class_nr	S class_A...	S class_A...	S iris_name	B 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

Figure 3.36. "true" and "false" values in the additional column named "Selected Scatter Plot" and produced by the Scatter Plot node. "true" is associated to all selected records. "false" indicates a not selected records and it is the default value.

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.

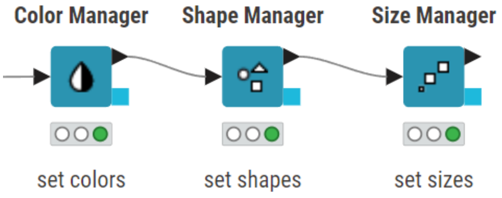


Figure 3.37. The three views properties nodes, to set color, shape, and size of plot markers.

- 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 on to the "Size Appender" node and then can be applied to another data set.

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.

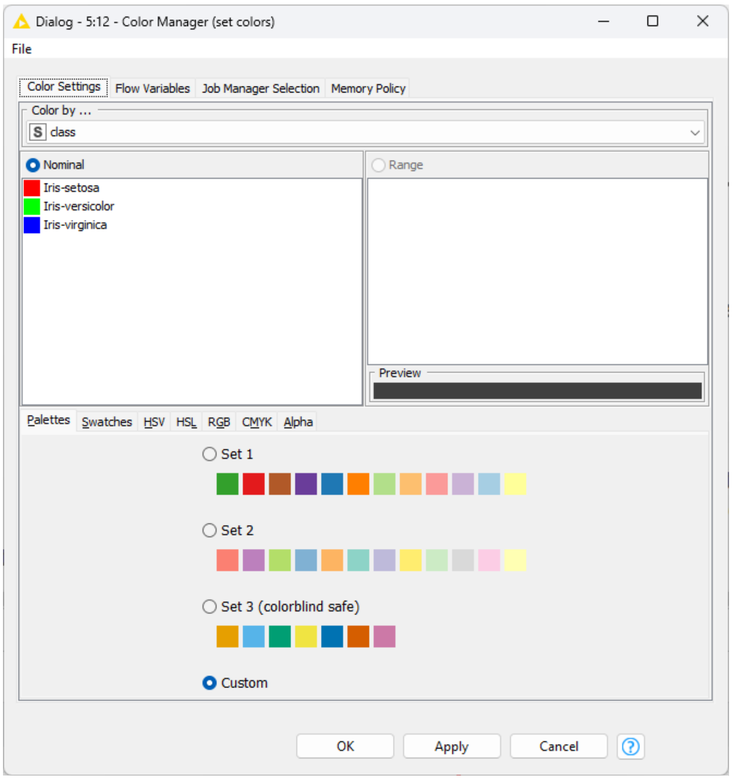
If a numerical column is selected, a color heat map spans the column numerical range.

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.

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.



Dialog - 5:12 - Color Manager (set colors)

Color by ...: S class

● Nominal

🟥 Iris-setosa
🟩 Iris-versicolor
🟦 Iris-virginica

○ Range

○ Set 1

○ Set 2

○ Set 3 (colorblind safe)

● Custom

Figure 3.38. Configuration window of the Color Manager node.

Note. As of KNIME Analytics Platform 5.2, the *Size Manager* node and the *Shape Manager* node are supported by the 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.

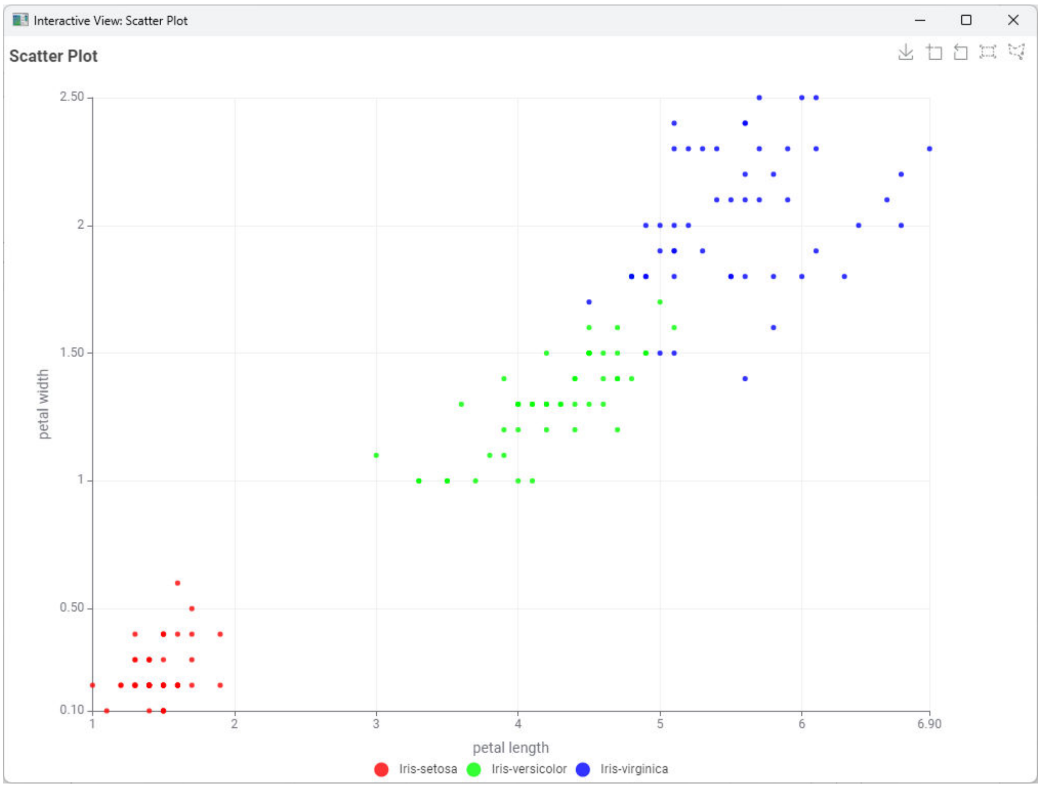


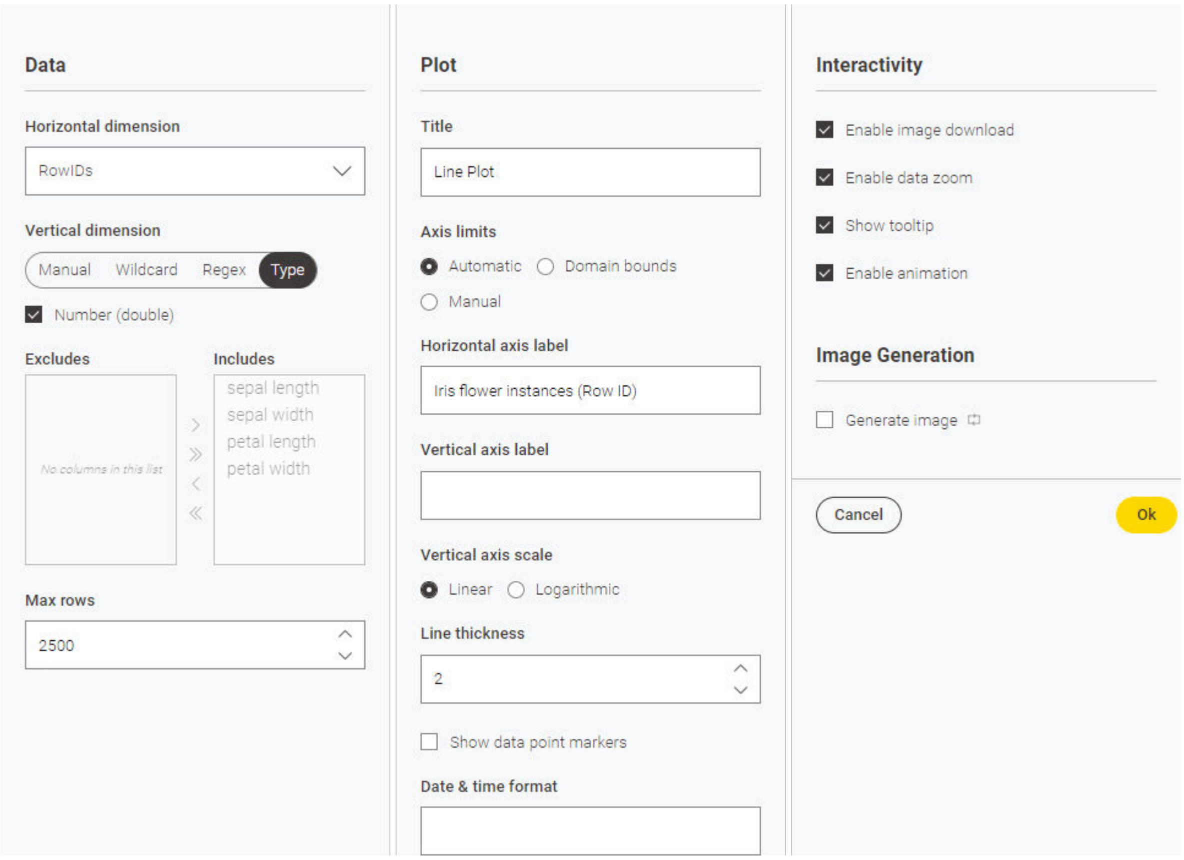
Figure 3.39. The view of the Scatter Plot node with customized colors for plot dots.

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 with the iris data set, where rows represent only different iris examples and have no temporal relationship. Nevertheless, we are going to use this workflow to show how a *Line Plot* node works.

Line Plot



Data

Horizontal dimension: RowIDs

Vertical dimension: Manual | Wildcard | Regex | **Type**

☑ Number (double)

Excludes: *No columns in this list*

Includes: sepal length, sepal width, petal length, petal width

Max rows: 2500

Plot

Title: Line Plot

Axis limits: ● Automatic ○ Domain bounds ○ Manual

Horizontal axis label: Iris flower instances (Row ID)

Vertical axis label:

Vertical axis scale: ● Linear ○ Logarithmic

Line thickness: 2

□ Show data point markers

Date & time format:

Interactivity

☑ Enable image download

☑ Enable data zoom

☑ Show tooltip

☑ Enable animation

Image Generation

□ Generate image

Cancel | Ok

Figure 3.40. Configuration window of the Line Plot node: the "Options" tab.

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

As for the previous new visualization nodes, the configuration window of the "Line Plot" node has four sections: "**Data**"; "**Plot**" for the plot details and "**Interactivity**"; and "**Image Generation**" for the generating the image at the output port.

The main difference is in the "Data" section, where an "Includes"/"Excludes" frame allows to select the columns for the plot.

The final view of the *Line Plot* 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.

Note. As of KNIME Analytics Platform 5.2, the *Line Plot* node does allow interactivity.

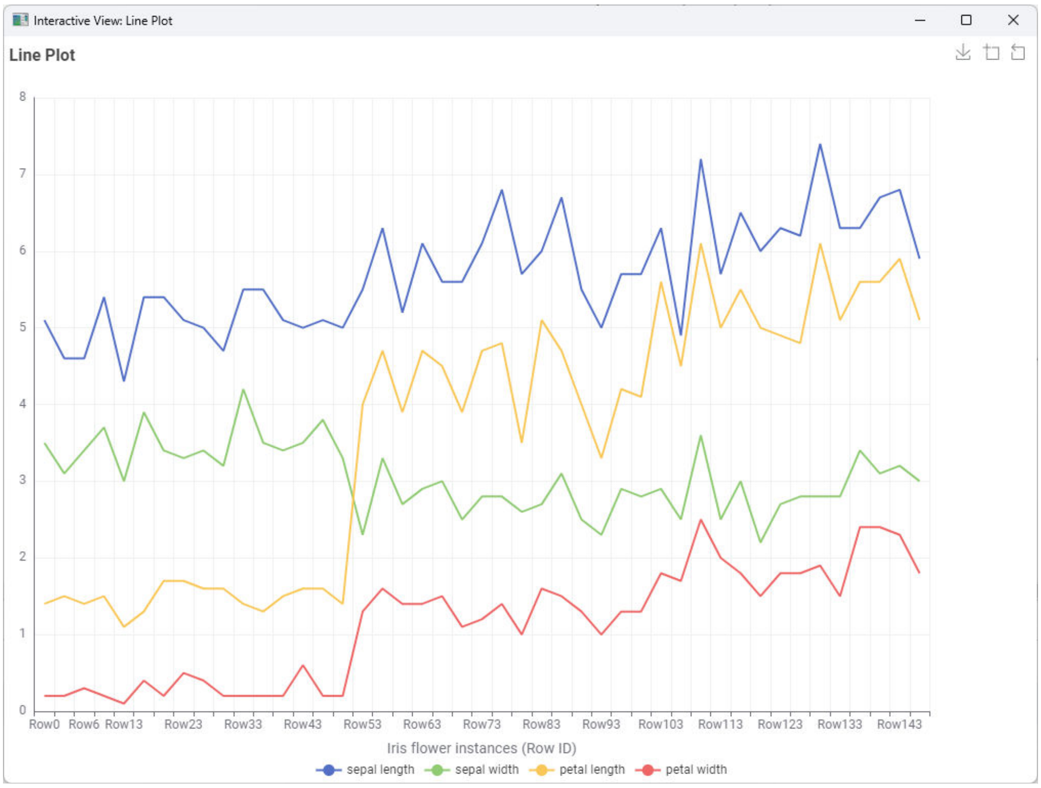


Figure 3.41. Plot view of the Line Plot node.

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”.

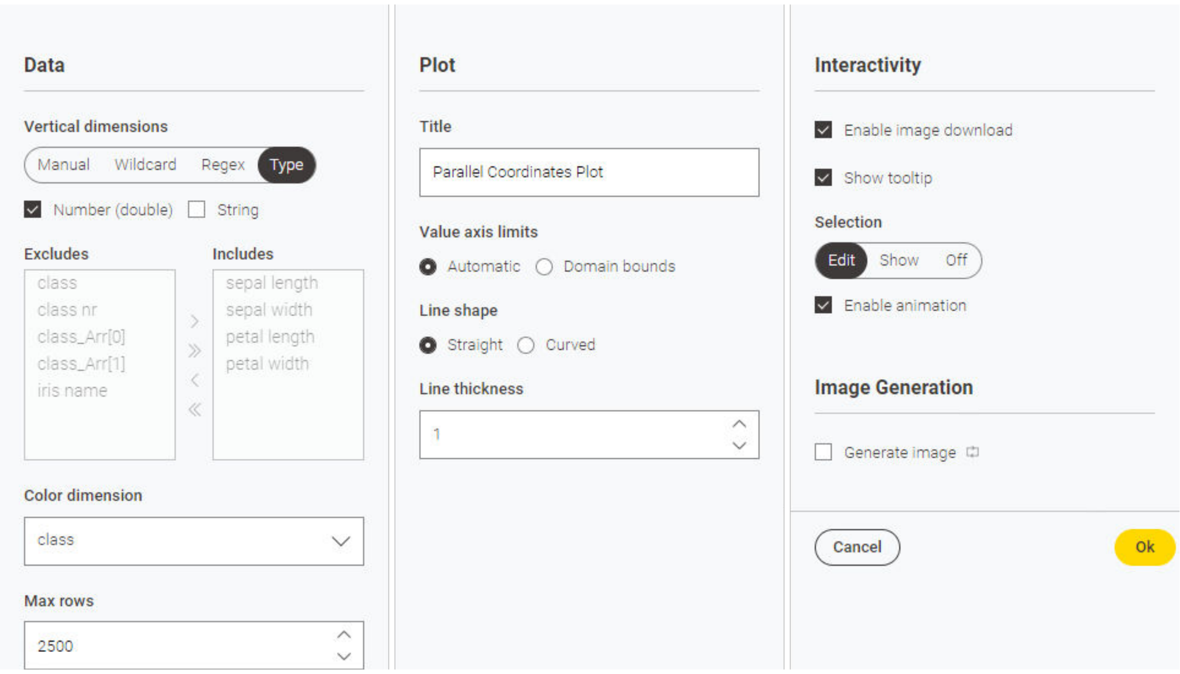
In the parallel coordinates plot, one column is one coordinate, i.e., one Y-axis. Multiple column values can be visualized on multiple coordinates, that is on multiple Y-axis. The data disposition along each axis can tell us some stories about the groups in the data set. The node that produces a parallel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

Parallel Coordinates Plot

The “Parallel Coordinates” 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 four sections.

- “**Data**” section contains an “Excludes/Includes” frame to insert/remove more columns (i.e. Y-axis) into/from the parallel coordinates plot. This section allows to decide the “color dimension” and maximum number of data rows to be included.
- “**Plot**” section defines general settings for the plot like the title, value axis limits, line shape, and line thickness.
- “**Interactivity**” section sets the interactivity level for the final view
- “**Image Generation**" section sets the option to generate the image at the output port.
- 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”).



Data

Vertical dimensions: Manual | Wildcard | Regex | **Type**

☑ Number (double) □ String

Excludes: class, class nr, class_Arr[0], class_Arr[1], iris name

Includes: sepal length, sepal width, petal length, petal width

Color dimension: class

Max rows: 2500

Plot

Title: Parallel Coordinates Plot

Value axis limits: ● Automatic ○ Domain bounds

Line shape: ● Straight ○ Curved

Line thickness: 1

Interactivity

☑ Enable image download

☑ Show tooltip

Selection: **Edit** | Show | Off

☑ Enable animation

Image Generation

□ Generate image

Figure 3.42. Configuration window of the Parallel Coordinates node.

Below is the view of the *Parallel Coordinates* node. As Y-axis we find: “sepal_length”, “sepal_width”, “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* node is also reduced with respect to, for example, the *Scatter Plot* node.

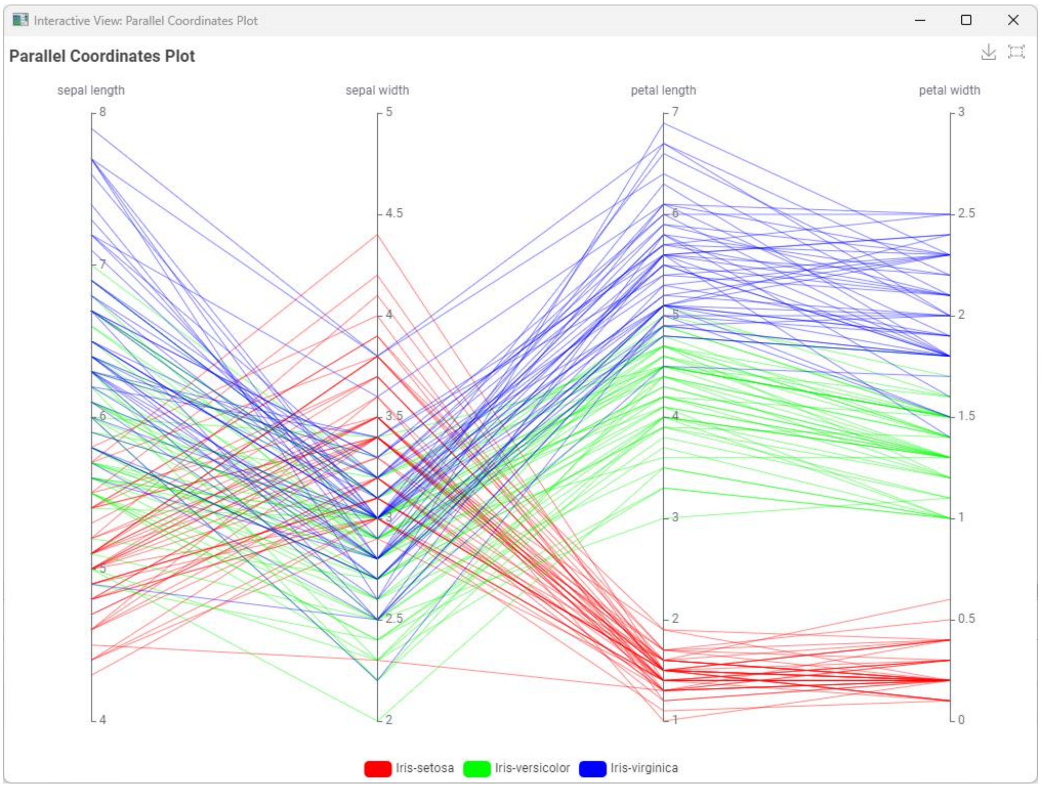


Figure 3.43. View of the Parallel Coordinates plot, where the four Iris measures are displayed on the four Y-axis. One line corresponds to one Iris plant.

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” node.

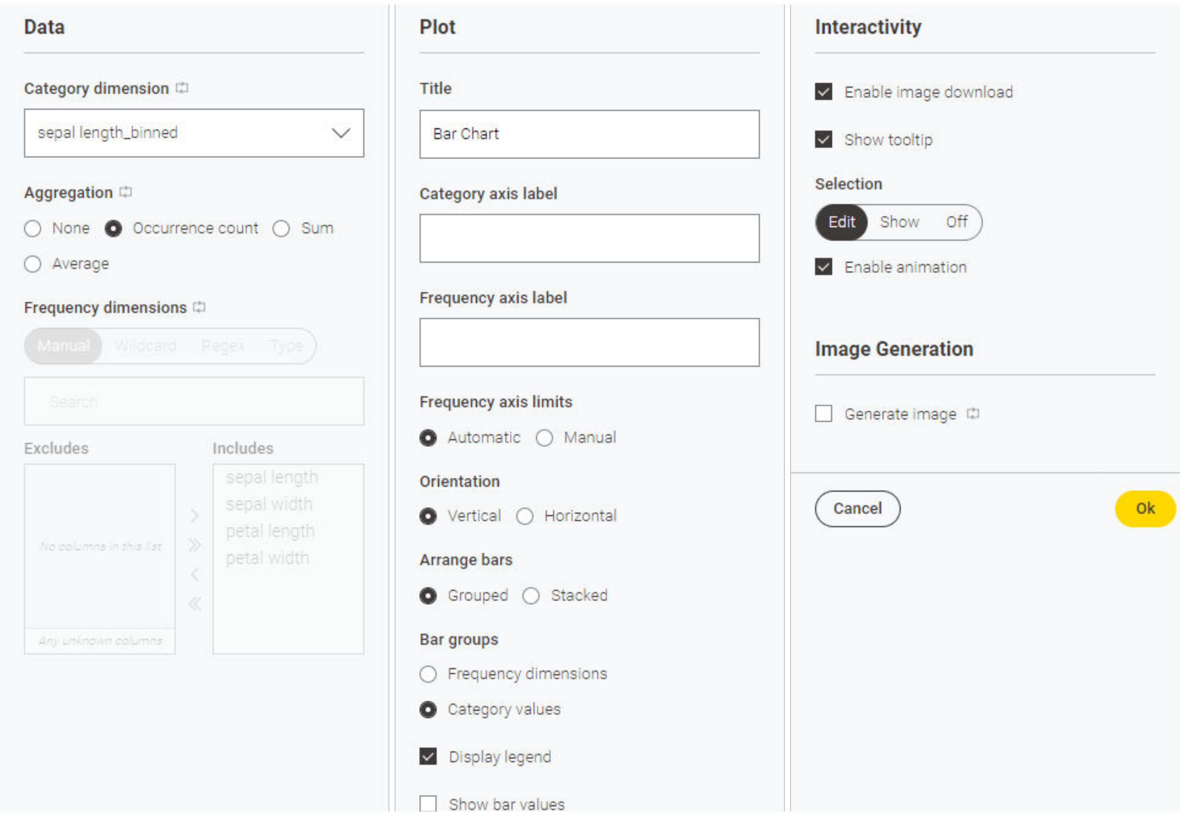
We already binned the “sepal_length” attribute in 9 bins. Now each data row of the input data table is assigned to a given bin according to the value of its “sepal_length” attribute. To build the histogram of attribute “sepal_length”, it is enough to count the number of occurrences in each “sepal_length_binned”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

Bar Chart

The “Bar Chart” 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 the 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.



Data

Category dimension: sepal length_binned

Aggregation: ○ None ● Occurrence count ○ Sum ○ Average

Frequency dimensions: Manual | Wildcard | Regex | Type

Search

Excludes: No columns in this list; Any unknown columns

Includes: sepal length, sepal width, petal length, petal width

Plot

Title: Bar Chart

Category axis label:

Frequency axis label:

Frequency axis limits: ● Automatic ○ Manual

Orientation: ● Vertical ○ Horizontal

Arrange bars: ● Grouped ○ Stacked

Bar groups: ○ Frequency dimensions ● Category values

☑ Display legend

☐ Show bar values

Interactivity

☑ Enable image download

☑ Show tooltip

Selection: Edit | Show | Off

☑ Enable animation

Image Generation

☐ Generate image

Figure 3.44. Configuration window of the Bar Char node: the "Options" tab, configured to draw a histogram.

These settings are all defined in the sections “Data” of the configuration window. Two additional tabs “Plot”, “Interactivity”, and “Image Generation” define respectively the plot graphical details, enabled interactive view controls, and the option to generate the image at the output port. "Plot" section includes preferences for title, axis labels, plot orientation, legend, and an option to show the bar values. "Interactivity" section enabling image download, option to show tool tip when hovered on the chart, subscribing and publishing to selection views, and enabling animation. The "Image Generation" section allows to generate the image at the output port.

The "Bar Chart" node does not have an optional input port for a color map.

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

> ***Note.*** The *Bar Chart* 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* node.

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" node with aggregation method "Average" on all three classes). The configuration window of the *Bar Chart* node and consequent histogram view are reported below.

The last node we would like to consider in this section is the "Table View" node. This node just displays the input data in a table.

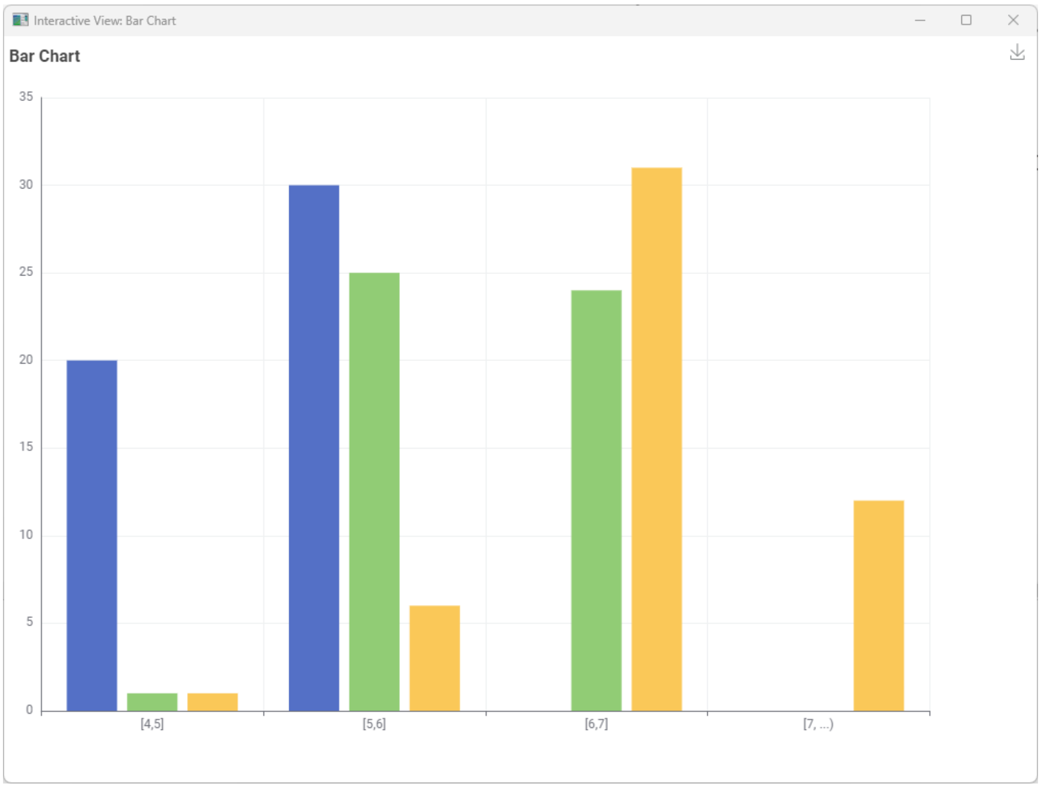


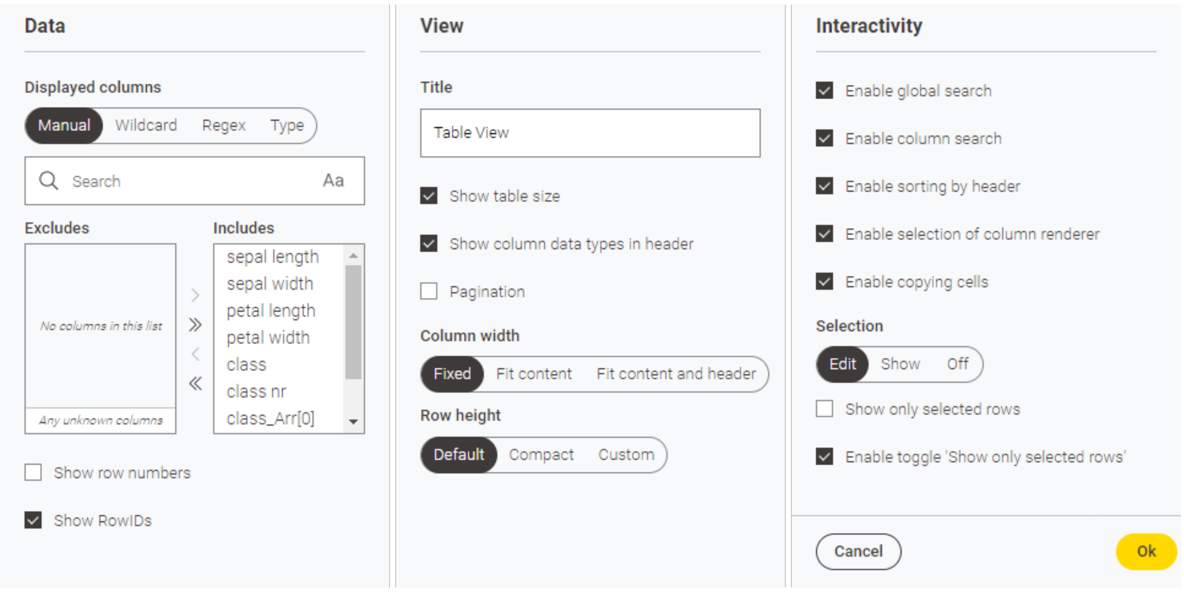
Figure 3.45. The view of the histogram of "septal_length" obtained with a Bar Chart node, using "Occurence Count" as aggregation method.

Table View

The “Table View” node displays the input data in a table. The configuration window consists of three tabs:

- “**Data**” section defines the data selection, for example which columns to display
- “**View**” section contains the usual settings to change the title, adjust table pagination, and an option to compact rows in the table view.
- “**Interactivity**” section contains the usual settings to determine the level of interactivity in the produced view.

- Depending on the settings in the "Interactivity" section, 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.



Data

Displayed columns

Manual | Wildcard | Regex | Type

Search Aa

Excludes: No columns in this list; Any unknown columns

Includes: sepal length, sepal width, petal length, petal width, class, class nr, class_Arr[0]

☐ Show row numbers

☑ Show RowIDs

View

Title: Table View

☑ Show table size

☑ Show column data types in header

☐ Pagination

Column width: Fixed | Fit content | Fit content and header

Row height: Default | Compact | Custom

Interactivity

☑ Enable global search

☑ Enable column search

☑ Enable sorting by header

☑ Enable selection of column renderer

☑ Enable copying cells

Selection: Edit | Show | Off

☐ Show only selected rows

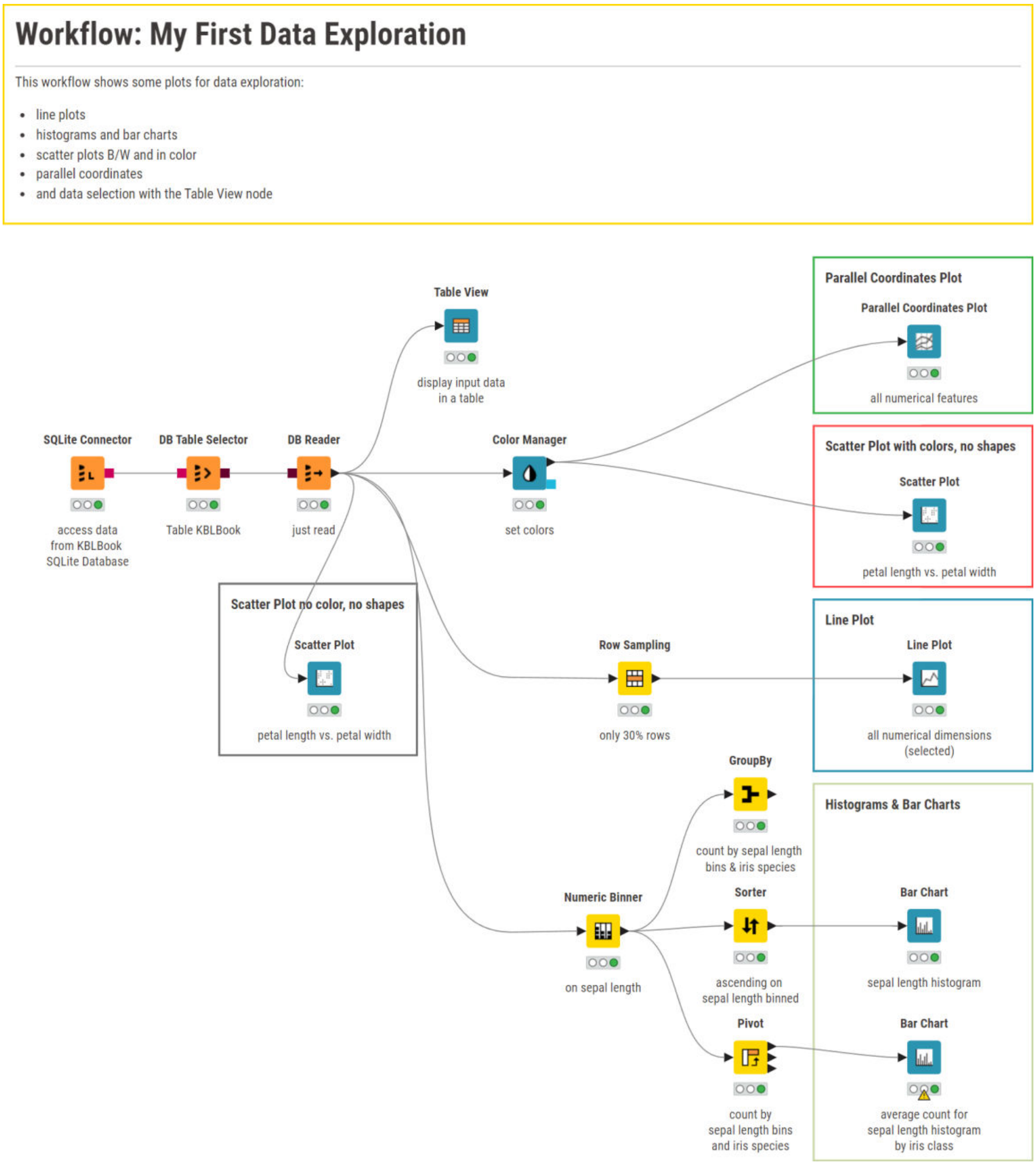
☑ Enable toggle 'Show only selected rows'

Figure 3.46. Configuration window of the Table View 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".



Workflow: My First Data Exploration

This workflow shows some plots for data exploration:

- line plots
- histograms and bar charts
- scatter plots B/W and in color
- parallel coordinates
- and data selection with the Table View node

Table View
display input data in a table
SQLite Connector
access data from KBLBook SQLite Database
DB Table Selector
Table KBLBook
DB Reader
just read
Color Manager
set colors
Parallel Coordinates Plot
Parallel Coordinates Plot
all numerical features
Scatter Plot with colors, no shapes
Scatter Plot
petal length vs. petal width
Scatter Plot no color, no shapes
Scatter Plot
petal length vs. petal width
Row Sampling
only 30% rows
Line Plot
Line Plot
all numerical dimensions (selected)
GroupBy
count by sepal length bins & iris species
Histograms & Bar Charts
Numeric Binner
on sepal length
Sorter
ascending on sepal length binned
Bar Chart
sepal length histogram
Pivot
count by sepal length bins and iris species
Bar Chart
average count for sepal length histogram by iris class

Figure 3.47. The final version of the workflow "My First Data Exploration".

3.9. Exercises

Exercise 1

Read file “yellow-small.data” from the Balloons dataset (you can find this file in the KBLdata folder or you can download it from: https://archive.ics.uci.edu/ml/machine-learning-databases/balloons/). This file has 5 columns: “Color”, “Size”, “Act”, “Age”, and “Inflated”. 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 “Final sentence” that says:

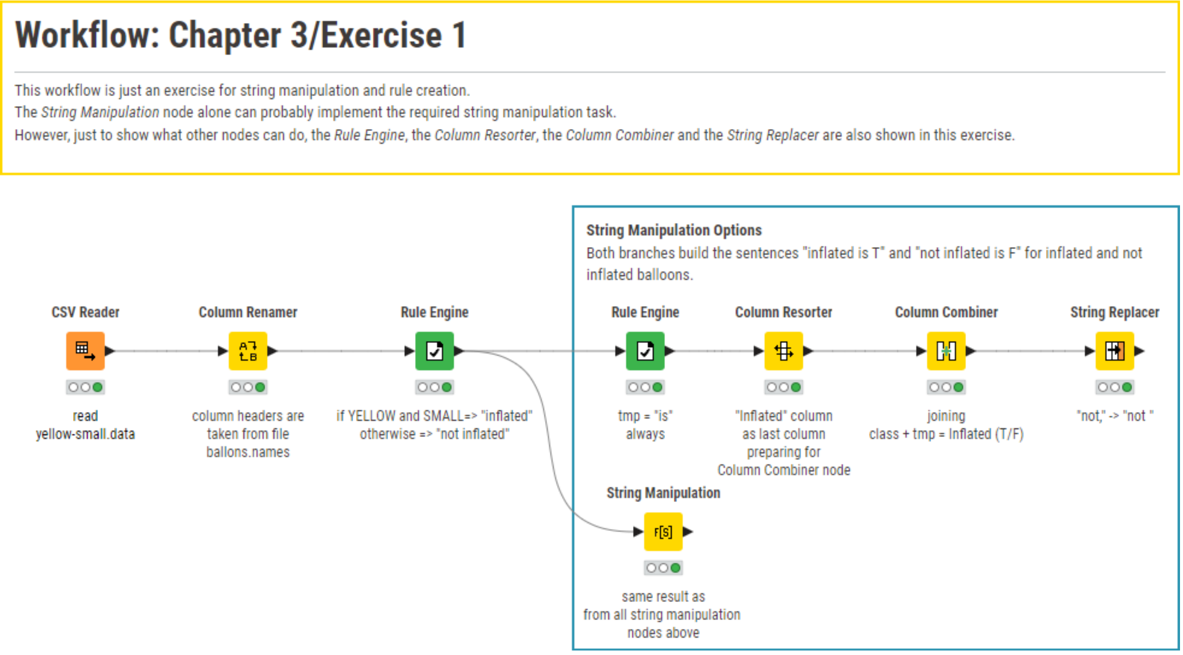
```
“inflated is T”
OR
“not inflated is F”
```

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

Solution to Exercise 1

There are two ways to proceed in this exercise.

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



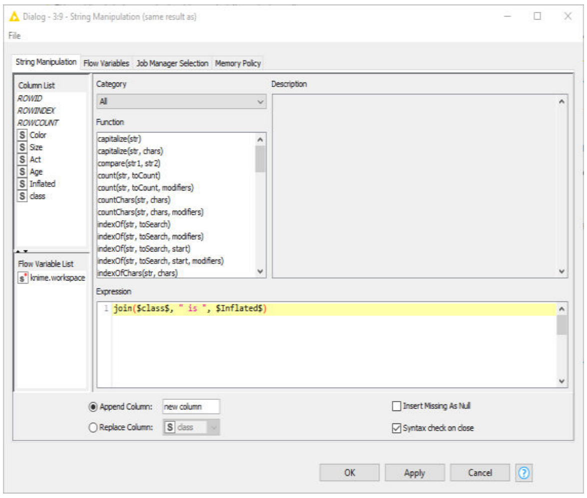
Workflow: Chapter 3/Exercise 1

This workflow is just an exercise for string manipulation and rule creation.
The *String Manipulation* node alone can probably implement the required string manipulation task.
However, just to show what other nodes can do, the *Rule Engine*, the *Column Resorter*, the *Column Combiner* and the *String Replacer* are also shown in this exercise.

String Manipulation Options

Both branches build the sentences "inflated is T" and "not inflated is F" for inflated and not inflated balloons.

Figure 3.48. Exercise 1: Workflow.



Dialog - 3:9 - String Manipulation (same result as)

Expression

```
join($class$, " is ", $Inflated$)
```

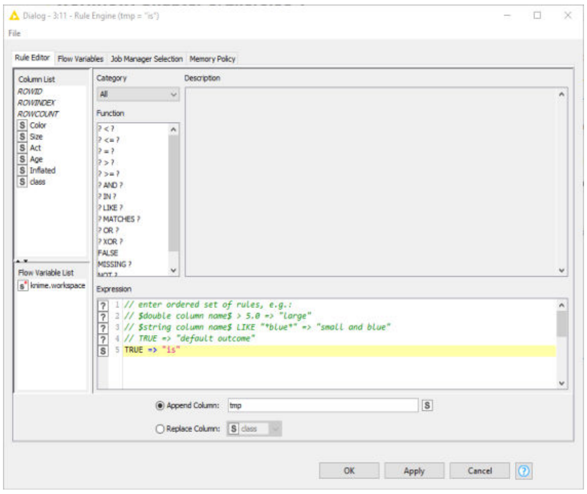
● Append Column: new column

○ Replace Column: S class

□ Insert Missing As Null

☑ Syntax check on close

Figure 3.49. Exercise 1: The String Manipulation node configuration (node commented with "same result as...").



Dialog - 3:11 - Rule Engine (tmp = "is")

Expression

```
// enter ordered set of rules, e.g.:
// $double column name$ > 5.0 => "large"
// $string column name$ LIKE "*blue*" => "small and blue"
// TRUE => "default outcome"
TRUE => "is"
```

● Append Column: tmp

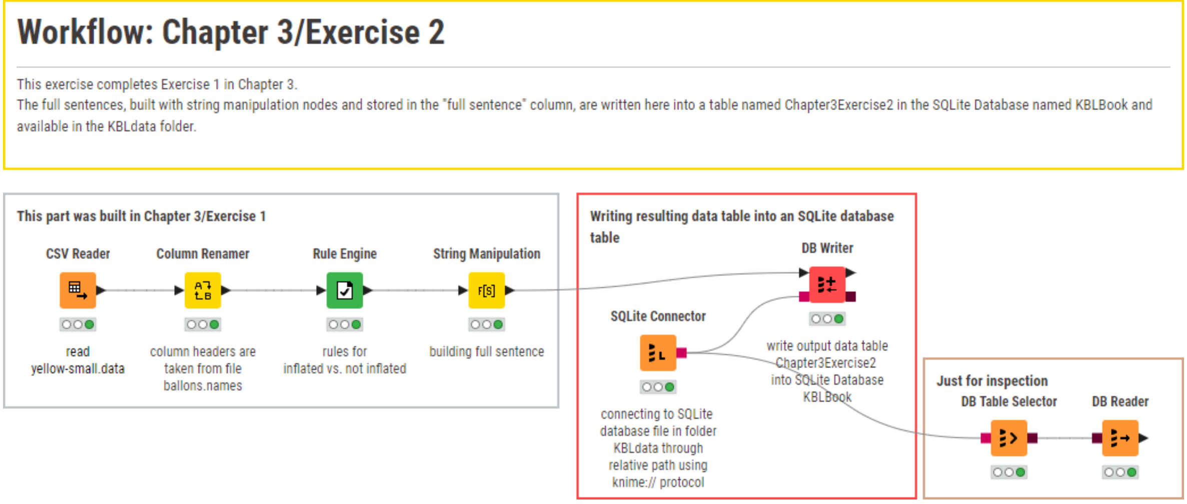
○ Replace Column: S class

Figure 3.50. 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* node and the *Database Writer* node.

Solution to Exercise 2

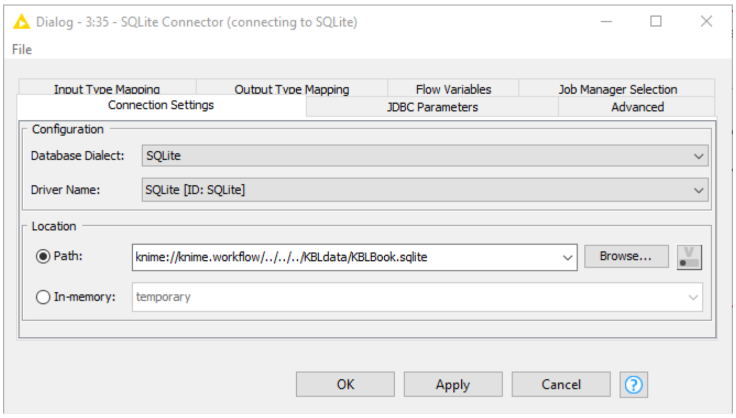


Workflow: Chapter 3/Exercise 2

This exercise completes Exercise 1 in Chapter 3.
The full sentences, built with string manipulation nodes and stored in the "full sentence" column, are written here into a table named Chapter3Exercise2 in the SQLite Database named KBLBook and available in the KBLdata folder.

This part was built in Chapter 3/Exercise 1
CSV Reader
read yellow-small.data
Column Renamer
column headers are taken from file ballons.names
Rule Engine
rules for inflated vs. not inflated
String Manipulation
building full sentence
Writing resulting data table into an SQLite database table
DB Writer
write output data table Chapter3Exercise2 into SQLite Database KBLBook
SQLite Connector
connecting to SQLite database file in folder KBLdata through relative path using knime:// protocol
Just for inspection
DB Table Selector
DB Reader

Figure 3.51. Exercise 2: Solution Workflow.



Dialog - 3:35 - SQLite Connector (connecting to SQLite)

Configuration
Database Dialect: SQLite
Driver Name: SQLite [ID: SQLite]

Location
● Path: knime://knime.workflow/../../../KBLdata/KBLBook.sqlite
○ In-memory: temporary

Figure 3.52. 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
- "Age" vs. "hours per week" Scatter Plot

Build the histogram and the bar chart using a *Bar Chart* node and the scatter plot using a *Scatter Plot* 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" 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. 43 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.

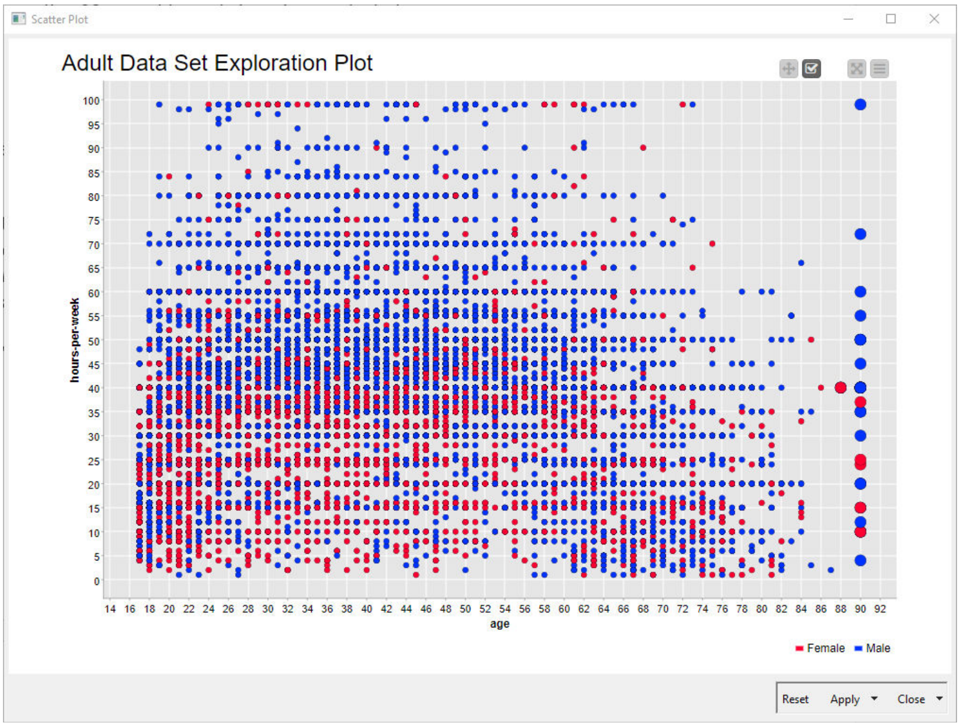


Figure 3.53. Scatter Plot of "age" vs. "hours per week" for the adult dataset. 90-year-old people have been selected.

Bar Chart on Number of Occurrences in each work class:

Here we used just a *Bar Chart* node counting number of occurrences on category "workclass"

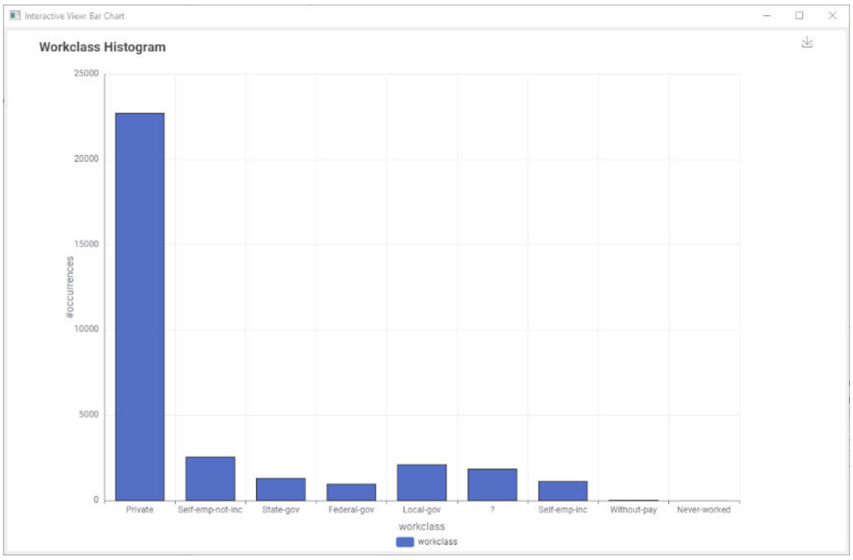


Figure 3.54. Bar Chart of number of occurrences of "work class" values in the adult data set.

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* 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.

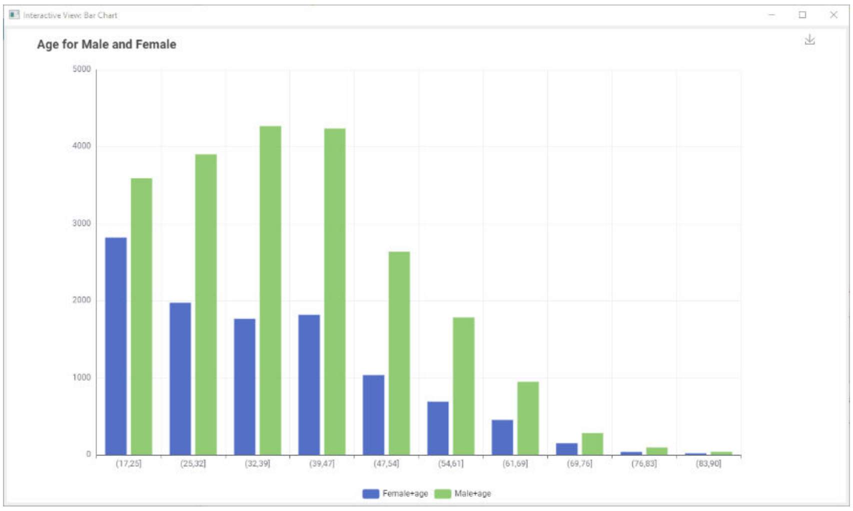
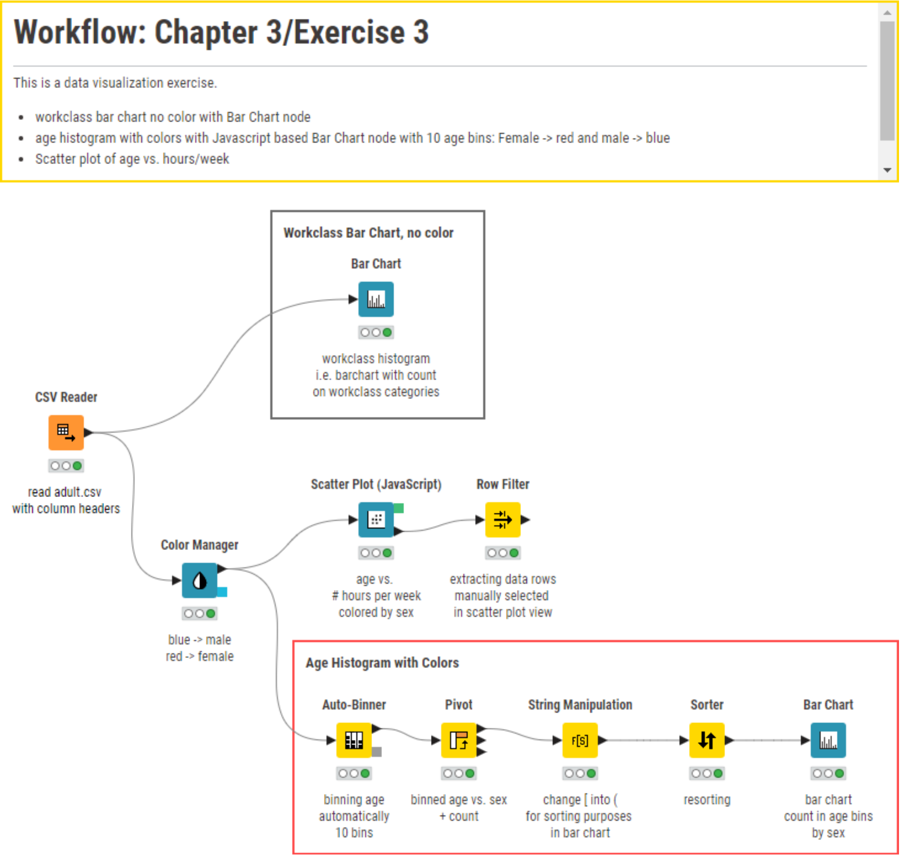


Figure 3.55. Age Histogram for men and women from a Bar Chart node.



Workflow: Chapter 3/Exercise 3

This is a data visualization exercise.

- workclass bar chart no color with Bar Chart node
- age histogram with colors with Javascript based Bar Chart node with 10 age bins: Female -> red and male -> blue
- Scatter plot of age vs. hours/week

Figure 3.56. 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 mainly machine learning 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 dataset) and applying the model to a set of new data (live data or a test dataset). Complying with these two phases, data modelling algorithms in 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 dataset. 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) dataset 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 dataset 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-quality model. For evaluation purposes, a new data set, named test dataset, is used. Of course, the test dataset has to contain different data from the training dataset, 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 dataset 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 dataset and the test dataset. To partition, reorganize, and re-unite datasets, we use nodes from the "Manipulation" → "Row" → "Transform" category.