Open for Innovation
KNIME

# KNIME Analytics Platform User Guide

KNIME AG, Zurich, Switzerland
Version 5.1 (last updated on 2023-10-17)

# Table of Contents

# Introduction

KNIME Modern UI is the new user interface for the KNIME Analytics Platform that is available with improved look and feel as the default interface, from KNIME Analytics Platform version 5.1.0 release.

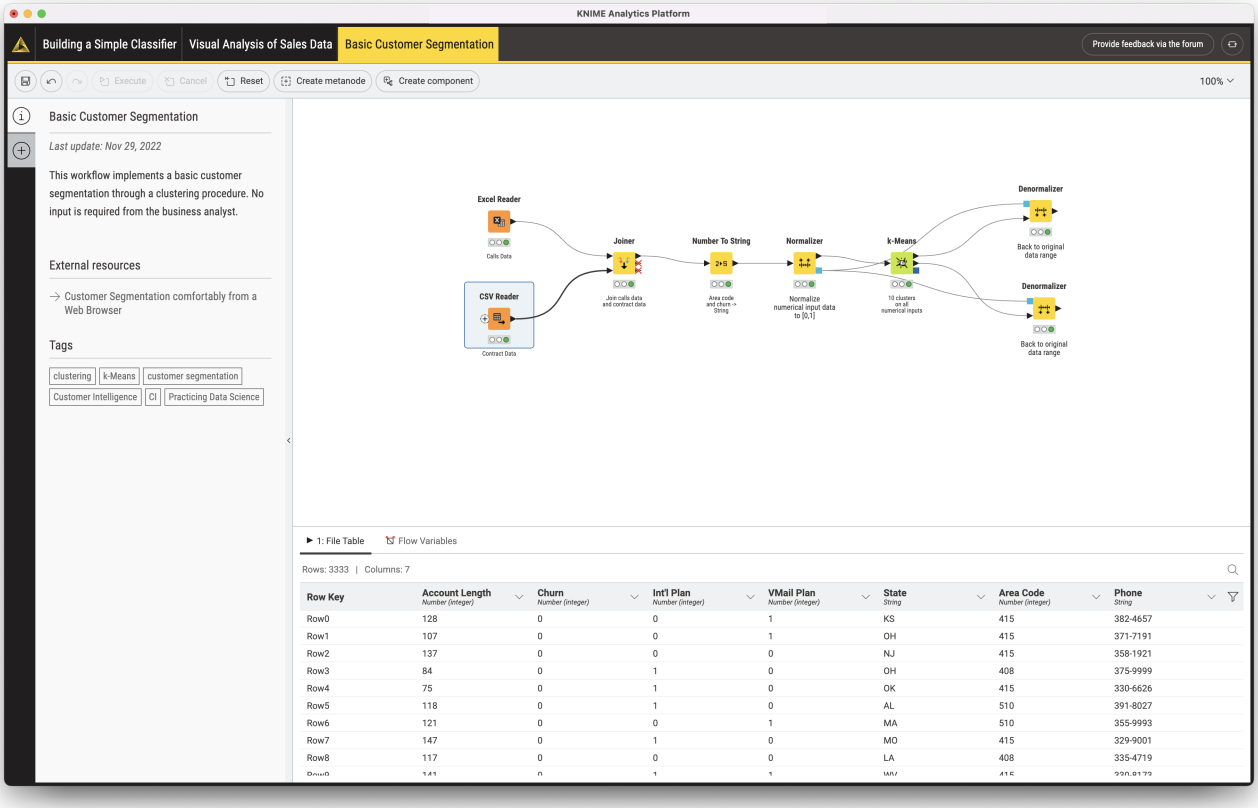



*Figure 1. KNIME Modern UI*

This guide covers the basics of the KNIME Analytics Plaform usage, guiding you in the first steps with the platform but also providing more advanced information about the most important concepts together with indication on how to configure the platform.

# Workspaces

When you start KNIME Analytics Platform, the KNIME Analytics Platform launcher window appears and you are asked to define the KNIME workspace, as shown in Figure 2.

The KNIME workspace is a folder on the local computer to store KNIME workflows, node settings, and data produced by the workflow.

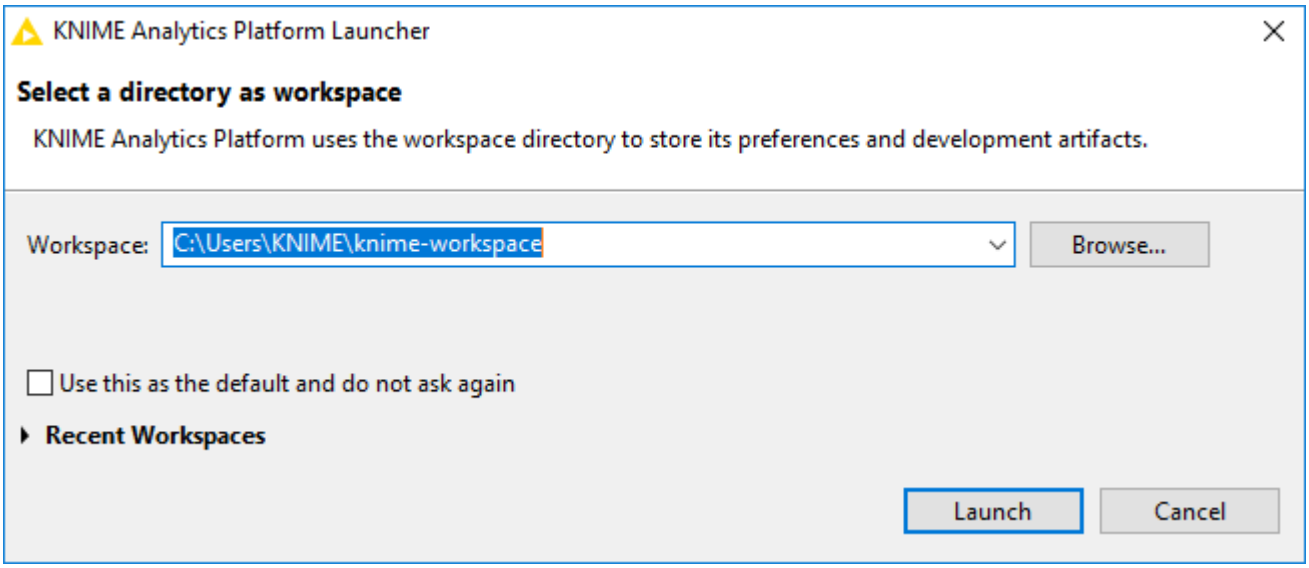



*Figure 2. KNIME Analytics Platform launcher*

The workflows, components and data stored in the workspace are available through the space explorer in the side panel navigation.

# User interface

After selecting a workspace for the current project, click *Launch*. The KNIME Analytics Platform user interface - the KNIME Workbench - opens.

The active workflow of the KNIME Analytics Platform will be displayed after switching from an opened workflow. If you have open multiple workflows before you switch the perspective, only the active workflow and all loaded workflow tabs of the current KNIME Analytics Platform will be displayed in the KNIME Modern UI. For each workflow you will see a workflow tab after switching. After clicking the first tab (with the KNIME logo) you end up at the entry page.

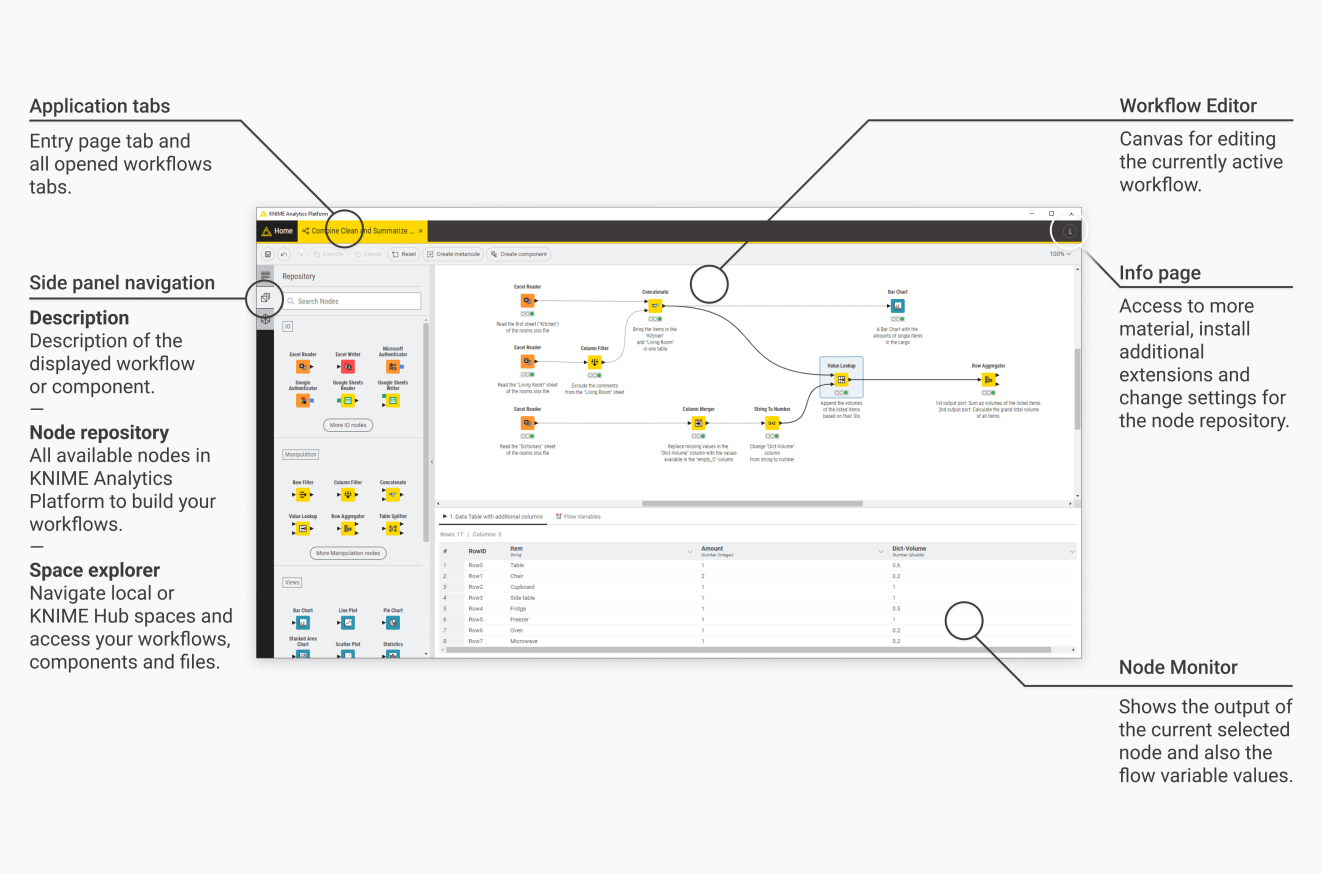



*Figure 3. General user interface layout — application tabs, side panel, workflow editor and node monitor*

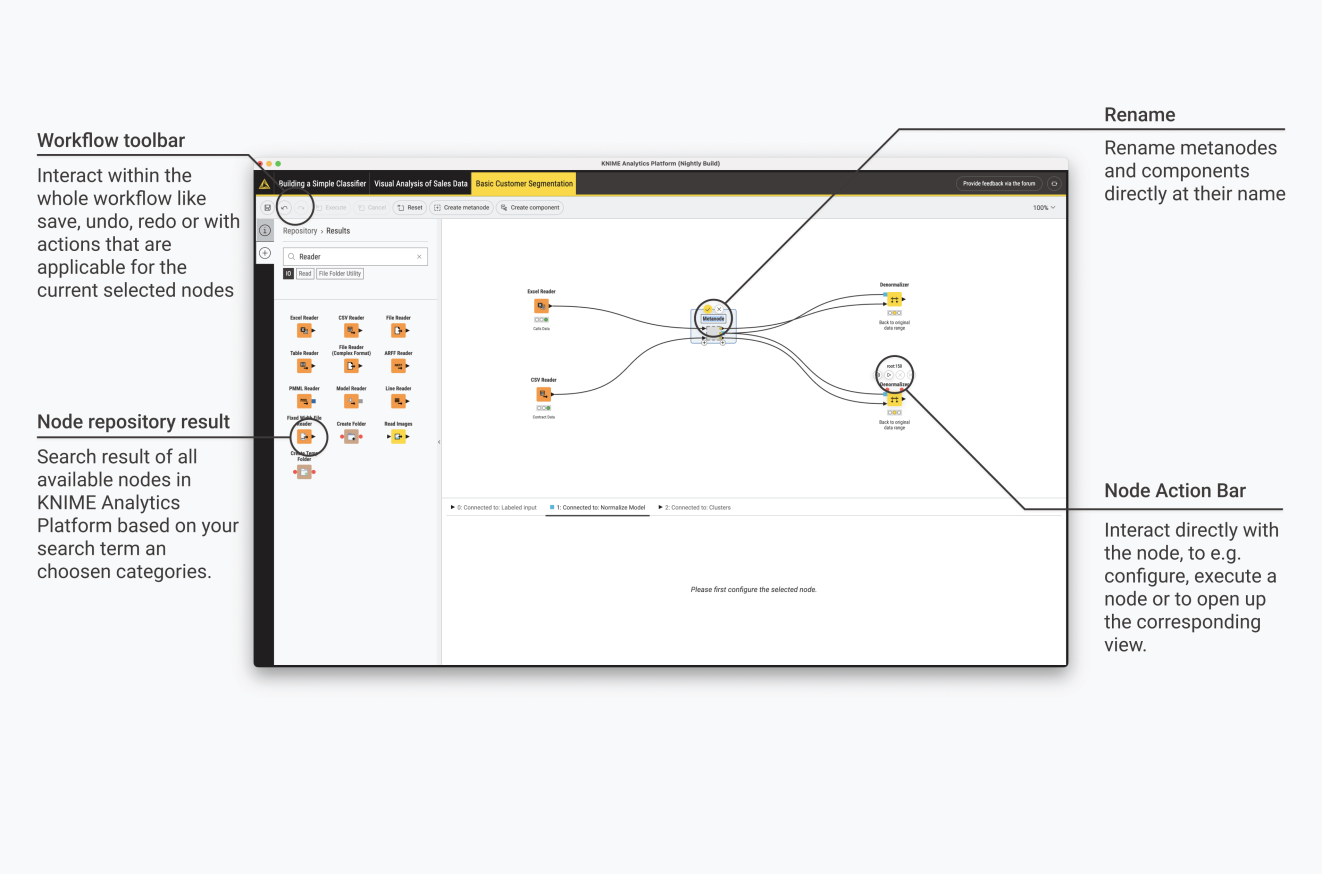



*Figure 4. User interface elements — workflow toolbar, node action bar, rename components and metanodes*

In the next few sections we explain the functionality of these components of the workbench:

- Entry page
- Workflow editor & nodes
- Connect to KNIME Hub
- Space explorer
- Node repository
- Node description
- Workflow description
- Node monitor
- Info page

# Entry page

The entry page is displayed by clicking the *Home* tab.

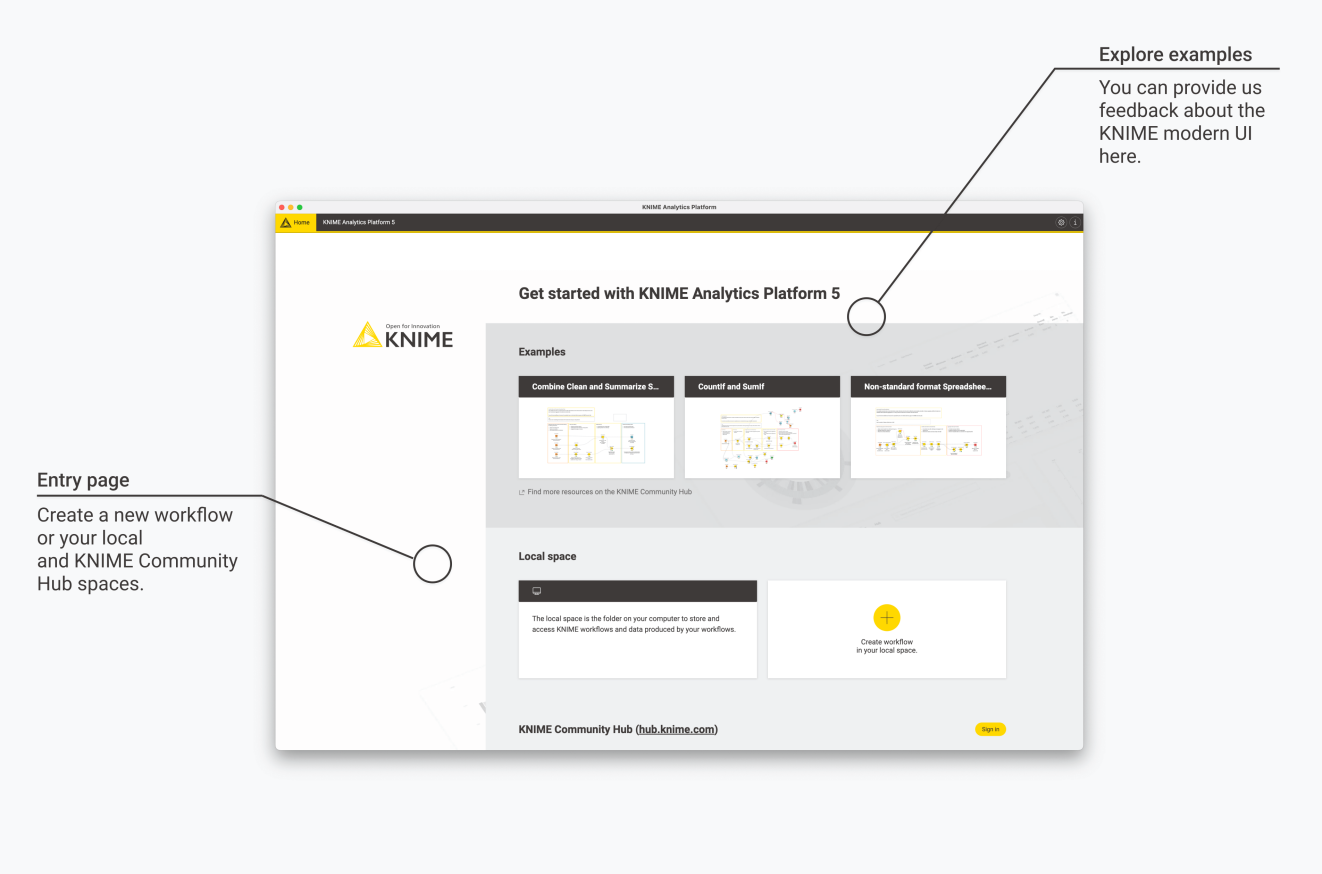



*Figure 5. Entry page to create or open workflows*

Here you will find:

- Three example workflows to help you get started
- Your local workspace - navigate your local workspace to find the workflow you want to work on
- Create a new workflow by clicking the yellow plus button
- Access one of the available mount points. Click Sign in, provide your credentials and start navigating the available spaces. By default only the local workspace, and the link to connect to your personal KNIME Community Hub space are visible. To add a new mount point follow the instructions in the Connect to KNIME Hub section.

Please be aware that in case you want to work with KNIME Server you need to switch to the classic user interface.

# Workflow editor & nodes

The workflow editor is where workflows are assembled. Workflows are made up of individual tasks, represented by nodes.

One way to create a new workflow is to go to the space explorer, click the three dots and select *Create workflow* from the menu. Give the workflow a name and click *Create*.

In the new empty workflow editor, create a workflow by dragging nodes from the node repository to the workflow editor, then connecting, configuring, and executing them.

## Nodes

In KNIME Analytics Platform, individual tasks are represented by nodes. Nodes can perform all sorts of tasks, including reading/writing files, transforming data, training models, creating visualizations, and so on.

### Facts about nodes

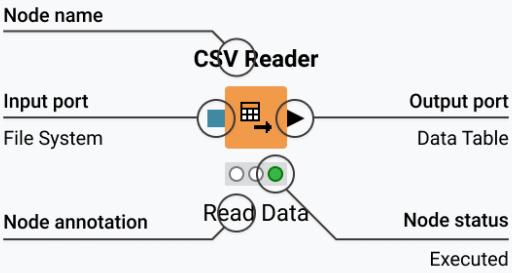



*Figure 6. A node in KNIME Analytics Platform*

- Each node is displayed as a colored box with input and output ports, as well as a status, as shown in Figure 6
- The input port(s) hold the data that the node processes, and the output port(s) hold the resulting datasets of the operation
- The data is transferred over a connection from the output port of one to the input port of another node.

For simplicity we refer to data when we refer to node input and output ports, but nodes can also have input and output ports that hold a model, a database query, or another type explained in Node Ports.

A node can me in different status as shown in the Figure 7.

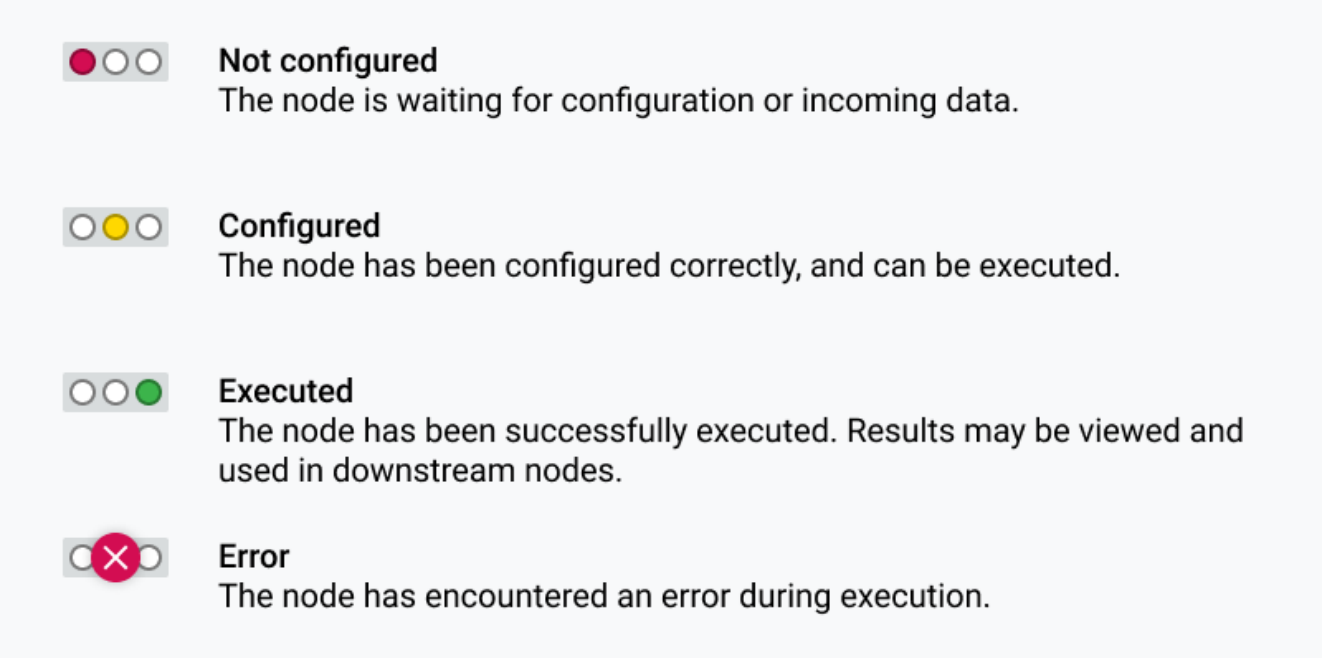



*Figure 7. A node can exist in different status*

### Changing the status of a node

The status of a node can be changed, either configuring, executing, or resetting it.

All these options can be found:

- In the node action bar - click the different icons to configure, execute, cancel, reset and when available open the view.

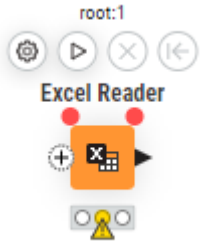



*Figure 8. Action bar of a node*

- In the context menu of a node - open the context menu by right clicking a node.

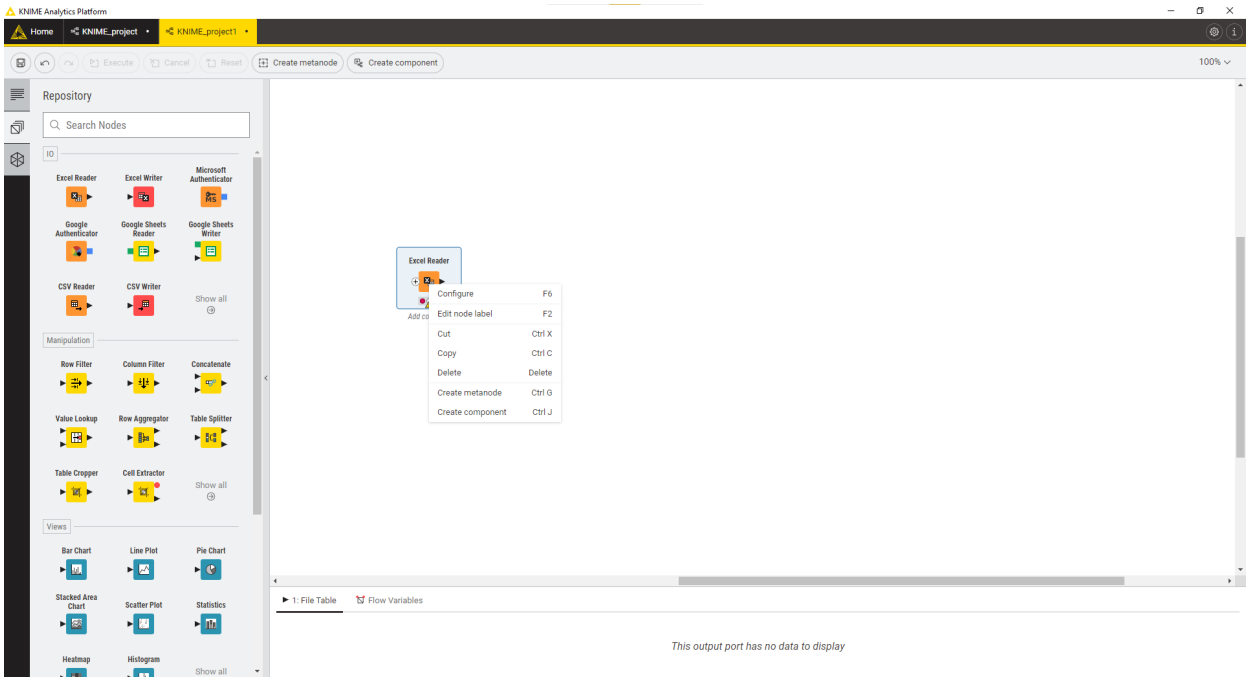



*Figure 9. Context menu of a node*

### Identifying the node status

The traffic light below each node shows the status of the node. When a node is configured, the traffic light changes from red to yellow, i.e. from "not configured" to "configured".

When a new node is first added to the workflow editor, its status is "not configured" - shown

by the red traffic light below the node.

### Configuring the node

The node can be configured by adjusting the settings in its configuration dialog.

Open the configuration dialog of a node by either:

- Double clicking the node
- Clicking the *Configure* button in the node action bar
- Right clicking a node and selecting *Configure* in the context menu
- Or, selecting the node and pressing F6

### Executing the node

Some nodes have the status "configured" already when they are created. These nodes are executable without adjusting any of the default settings.

Execute a node by either:

- Clicking the *Execute* button in the node action bar
- Right clicking the node and selecting *Execute*
- Or, selecting the node and pressing F7

If execution is successful, the node status becomes "executed", which corresponds to a green traffic light. If the execution fails, an error sign will be shown on the traffic light, and the node settings and inputs will have to be adjusted as necessary.

### Canceling execution of the node

To cancel the execution of a node click the *Cancel* button in the node action bar, or right click it and select *Cancel* or select it and press F9.

### Resetting the node

To reset a node click the *Reset* button in the node action bar, or right click it and select *Reset* or select it and press F8.

Resetting a node also resets all of its subsequent nodes in the workflow. Now, the status of the node(s) turns from "executed" into "configured", the nodes' outputs are cleared.

### Node ports

A node may have multiple input ports and multiple output ports. A collection of interconnected nodes, using the input ports on the left and output ports on the right, constitutes a workflow. The input ports consume the data from the output ports of the predecessor nodes, and the output ports provide data to the successor nodes in the workflow.

Besides data tables, input and output ports can provide other types of inputs and outputs. For each type the pair of input and output port looks different, as shown in Figure 10.

An output port can only be connected to an input port of the same type - data to data, model to model, and so on.

Some input ports can be empty, like the data input port of the Decision Tree View node in Figure 10. This means that the input is optional, and the node can be executed without the input. The mandatory inputs, shown by filled input ports, have to be provided to execute the node.

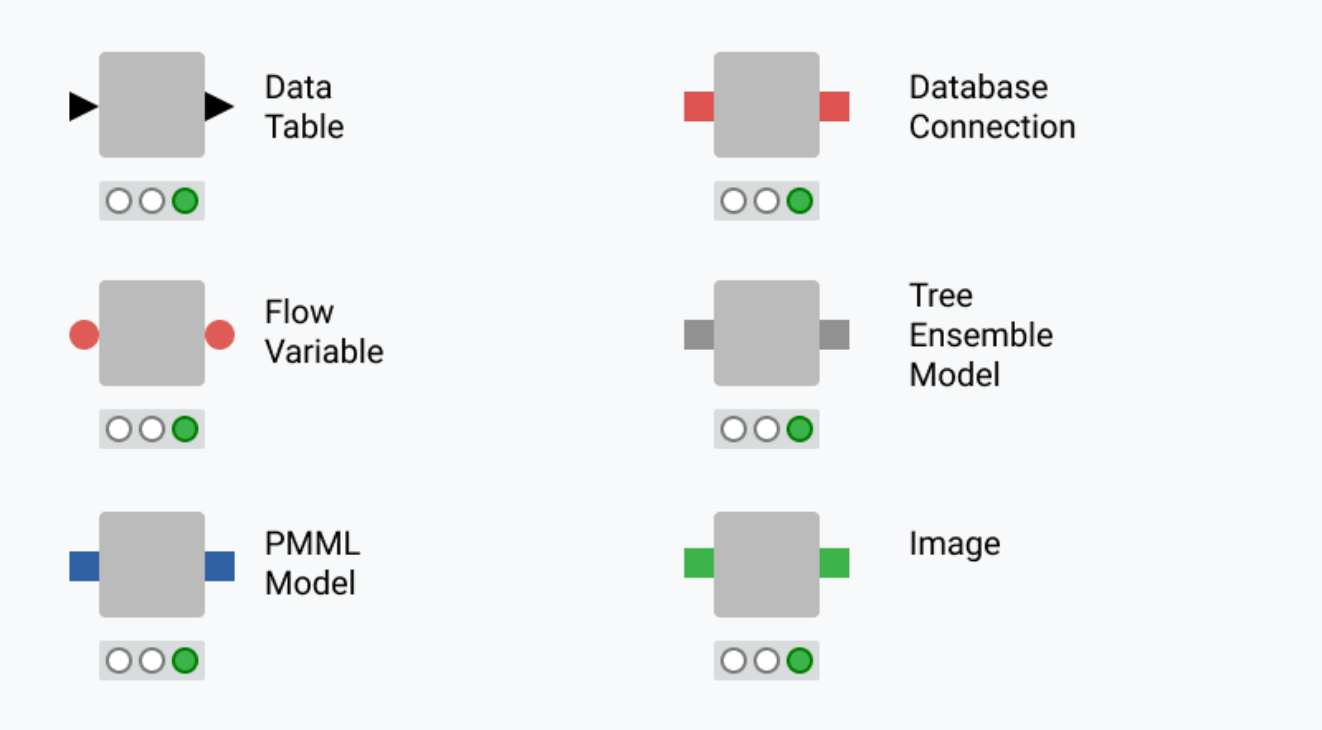



*Figure 10. Common port types*

A tooltip gives a short explanation of the input and output ports. If the node is executed, the

dimensions of the outcoming data are shown in its data output port. A more detailed explanation of the input and output ports is in the node description.

## Adding nodes to the canvas

Currently there are three ways of adding nodes to your canvas to build your workflow:

1. Drag and drop a node from the node repository,
2. double click on a node inside the node repository, or
3. drop a connection into an empty area inside the workflow canvas to display the quick nodes adding panel. Up to 12 recommended nodes are displayed inside this panel. Also you can search in the panel for all compatible nodes.Click the desired node to add it.

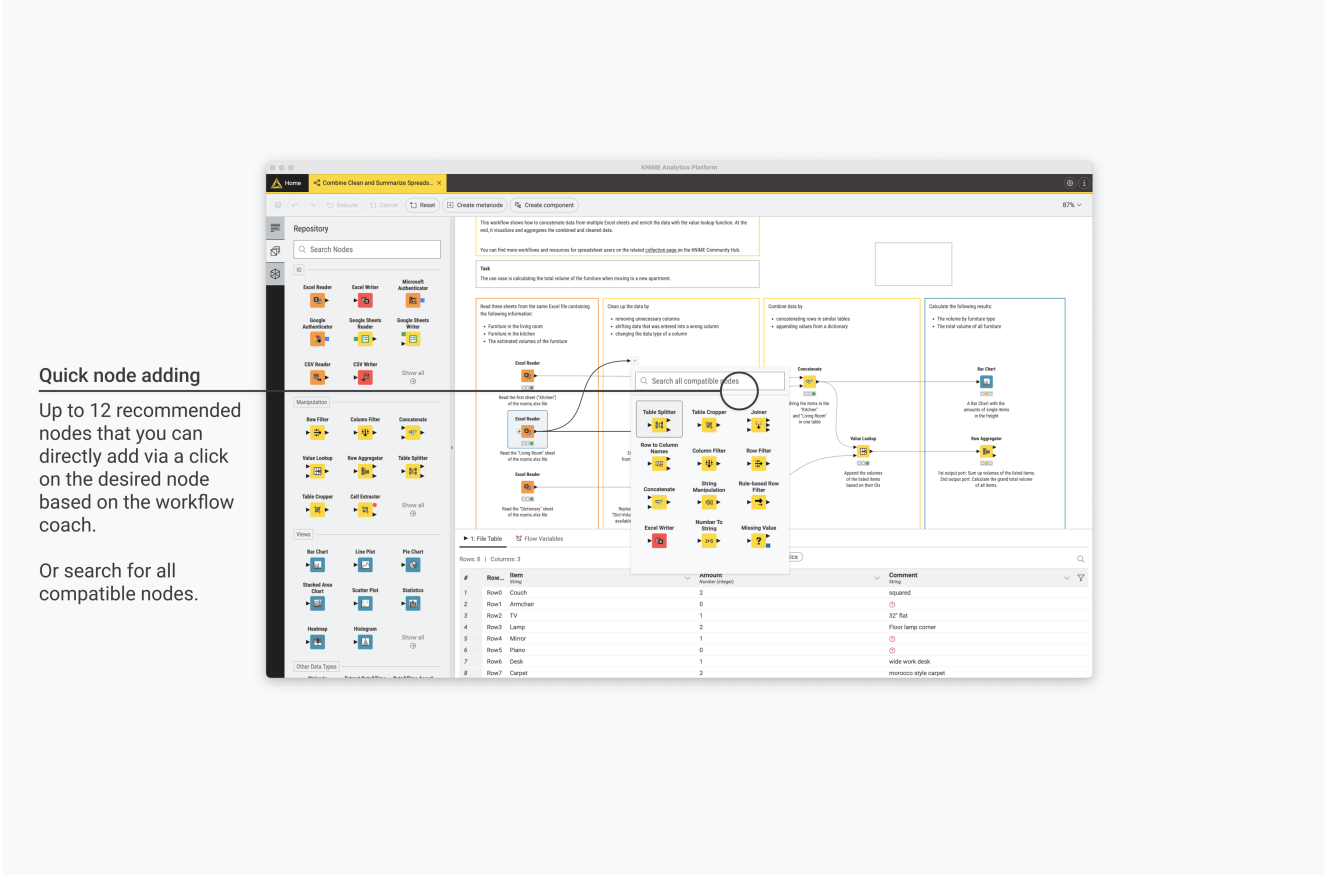



*Figure 11. Quick nodes adding with recommended nodes*

To use quick nodes adding you need to allow us to receive anonymous usage data. This is possible at the startup of the KNIME Analytics Platform or after switching to a new workspace by selecting *Yes* in the “Help improve KNIME” dialog.

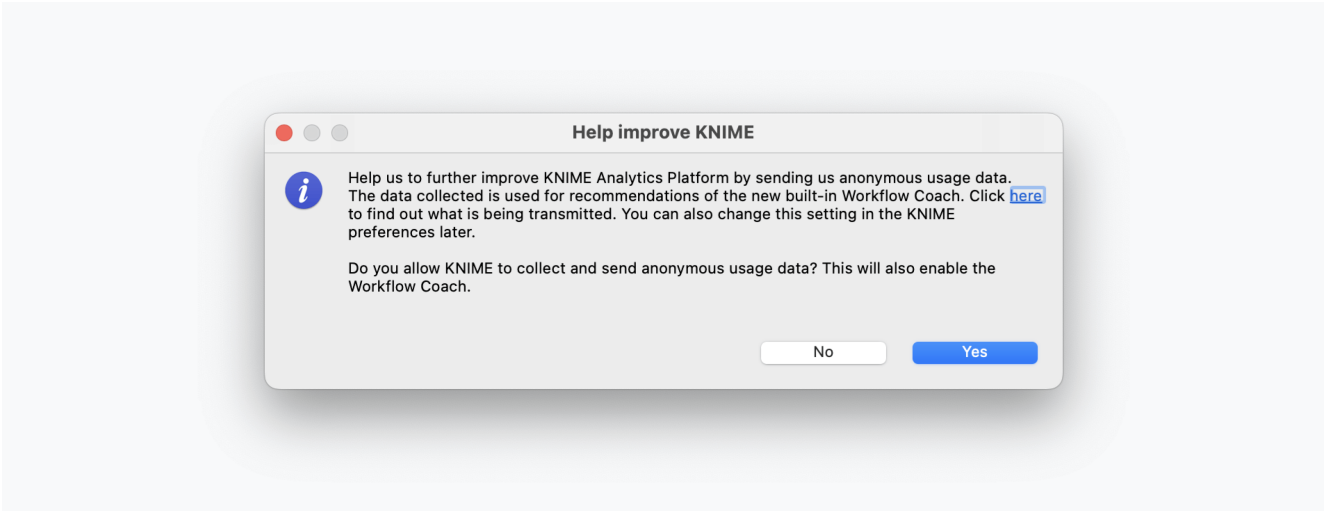



*Figure 12. "Help improve KNIME" dialog*

You can also activate it, via the *Open Preference* button that is displayed in the quick nodes adding panel.

Click here to find out what is being transmitted. If you don't want to do this anymore, you can deactivate it at any time in the KNIME Workflow Coach Preferences.

To open the preferences follow these steps:

1. Click the cog button in the top right corner of the Analytics Platform
2. Go to *KNIME → Workflow Coach*
3. Deactivate the setting *Node Recommendations by the Community*

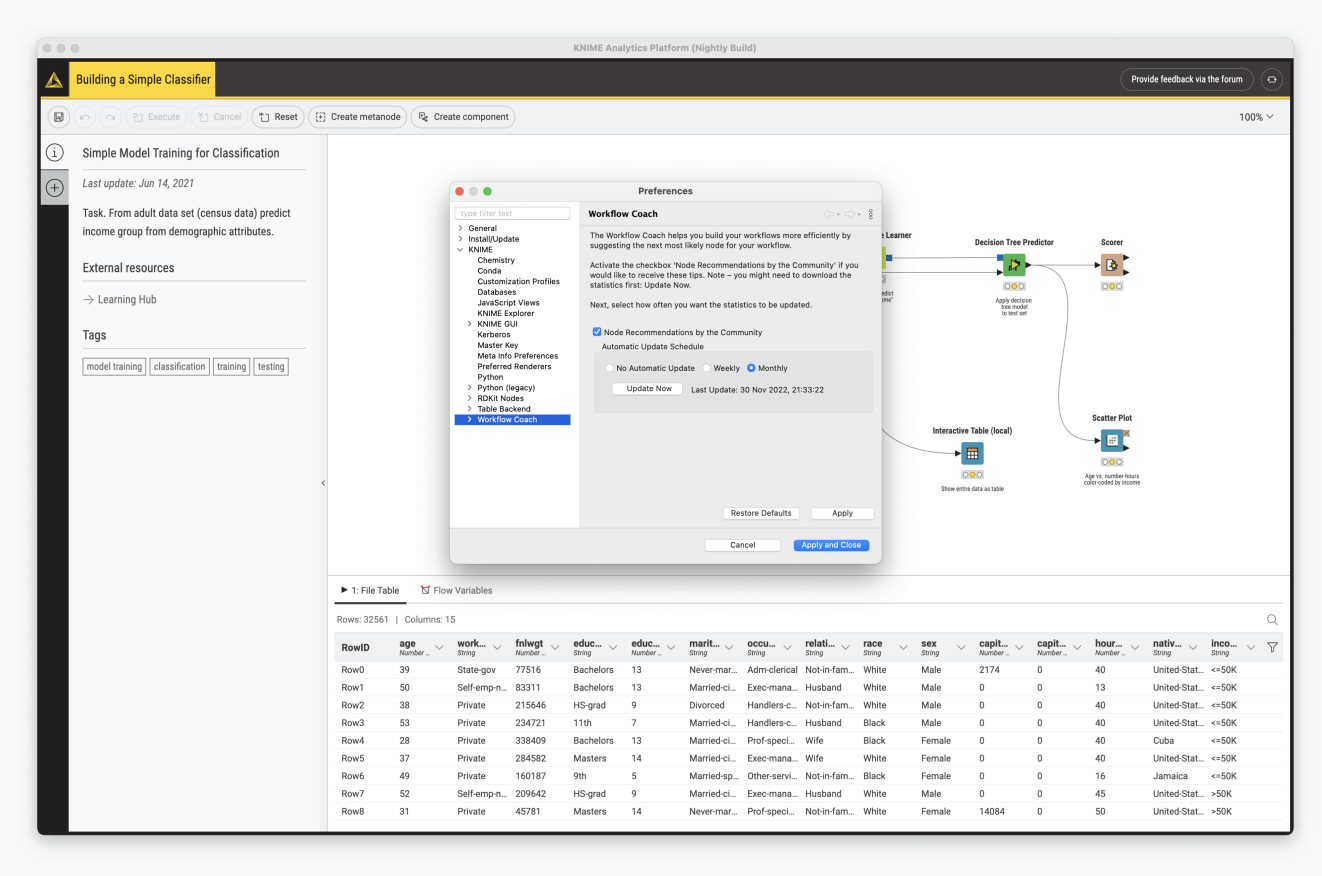



*Figure 13. Workflow Coach Preferences*

## How to select, move, copy, and replace nodes in a workflow

Nodes can be moved into the workflow editor by dragging and dropping them. To copy nodes between workflows, select the chosen nodes, right click the selection, and select *Copy* in the menu. In the destination workflow, right click the workflow editor, and select *Paste* in the menu.

To select a node in the workflow editor, click it once, and it will be surrounded by a border. To select multiple nodes, draw a rectangle over the nodes with the mouse.

Replace a node by dragging a new node onto an existing node. Now the existing node will be covered with a colored box with an arrow and boxes inside as shown in Figure 14. Releasing the mouse replaces the node.

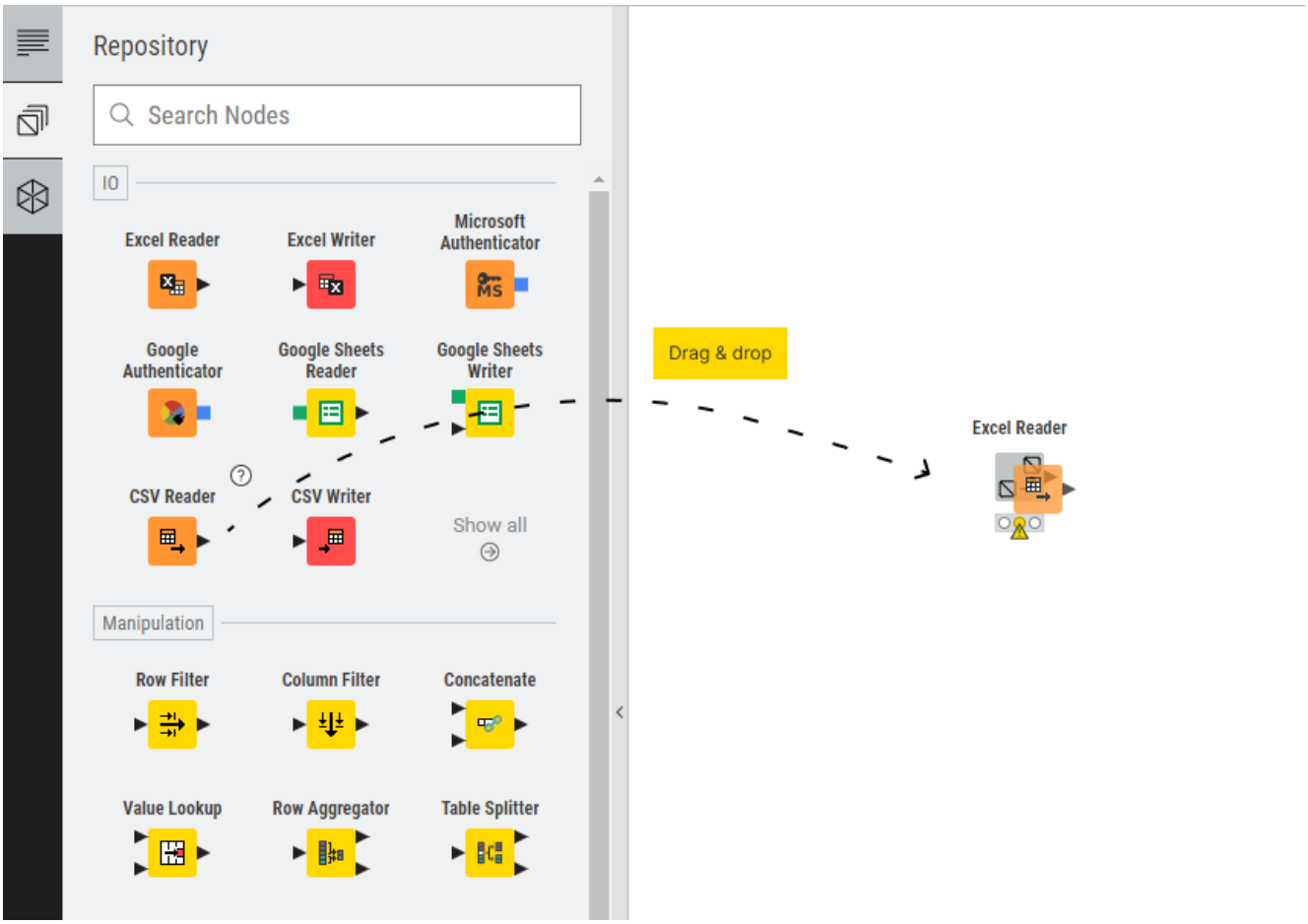



*Figure 14. Replacing a node in a workflow*

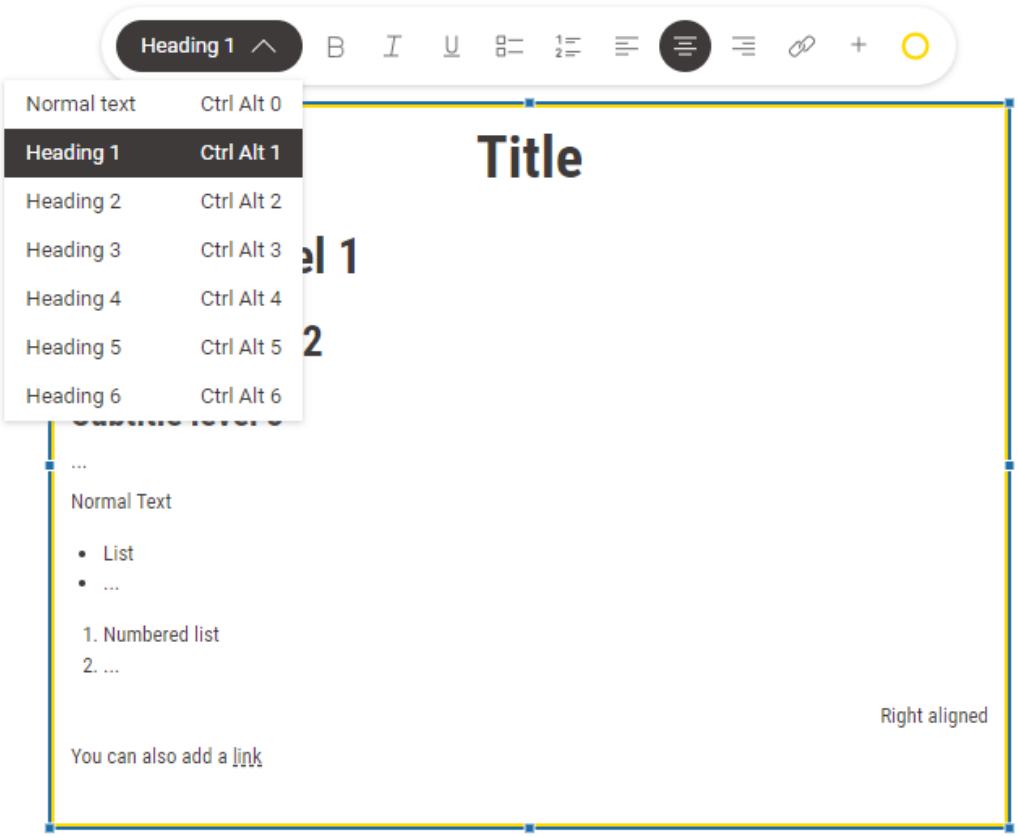
## Comments and annotations

You have two options in the workflow editor to document a workflow:

- Node label - Add a comment to an individual node by double clicking the text field below the node and editing the text


*Figure 15. Writing a node comment*

- Workflow annotation - Add a general comment to the workflow, right click the workflow editor and select *New workflow annotation* in the menu. Now a text box will appear in the workflow editor.




Double click the workflow annotation to add text and format the text and change the color of the annotation outline. To change the format you can use the annotation bar or use the following syntax:

- To create a heading, add number signs (#), followed by a space, in front of a word or phrase. The number of number signs you use should correspond to the heading level (<h1> to <h6>).
- To create a bullet list, add a star sign (*) followed by a space.
- To create a numbered list, add a number followed by a point (1.), followed by a space.
- To make a text bold, italic, or underlined, select the text and press CTRL+b, CTRL+i, CTRL+u.

Finally you can click outside the annotation and click the annotation once again to move it around the canvas or to change its dimensions.

## Connect to KNIME Hub

By default you can connect to your account on KNIME Community Hub from the *Home* tab.

It is possible to add a new KNIME Hub instance mount point by clicking the cog icon on the top right corner of the Analytics Platform. Go to *KNIME Explorer* section and click *New…*. In the window that opens select *KNIME Hub* and add your Hub URL. Then click *Apply*.

Now the new mount point will show up in the *Home* tab.

Sign in and select the space you want to work on. The content of the space and the related operations you can do on the items are visible in the space explorer.

## Switch back to KNIME classic user interface

If you want to switch back to the classic KNIME Analytics Platform user interface, go to the info page, by clicking the info button at the top right corner of the KNIME Analytics Platform. Here, in the section **Switch to classic user interface** click the button *Switch to KNIME classic user interface*,

You can switch back to KNIME Modern UI at any time by pressing the button *Open KNIME Modern UI* in the classic user interface, at the top right corner.

Really, really, *really* important disclaimer

Workflow elements such as connectors or annotations are visualized in a new way and may not look exactly like in the current KNIME Analytics Platform. Changes will therefore not look 100% the same.

## Space explorer

The space explorer is where you can manage workflows, folders, components and files in a space, either local or remote on a KNIME Hub instance. A space can be:

- Your local workspace you selected at the start up of the KNIME Analytics Platform
- One of your user's spaces on KNIME Community Hub
- One of your team's spaces on KNIME Business Hub

Please be aware that in case you want to work with KNIME Server you need to switch to the classic user interface.

You can switch to other spaces by:

- Going to the *Home* tab and selecting one of the available spaces
- On the top of the space explorer you can sign in to any of the Hub mount points and select a space.

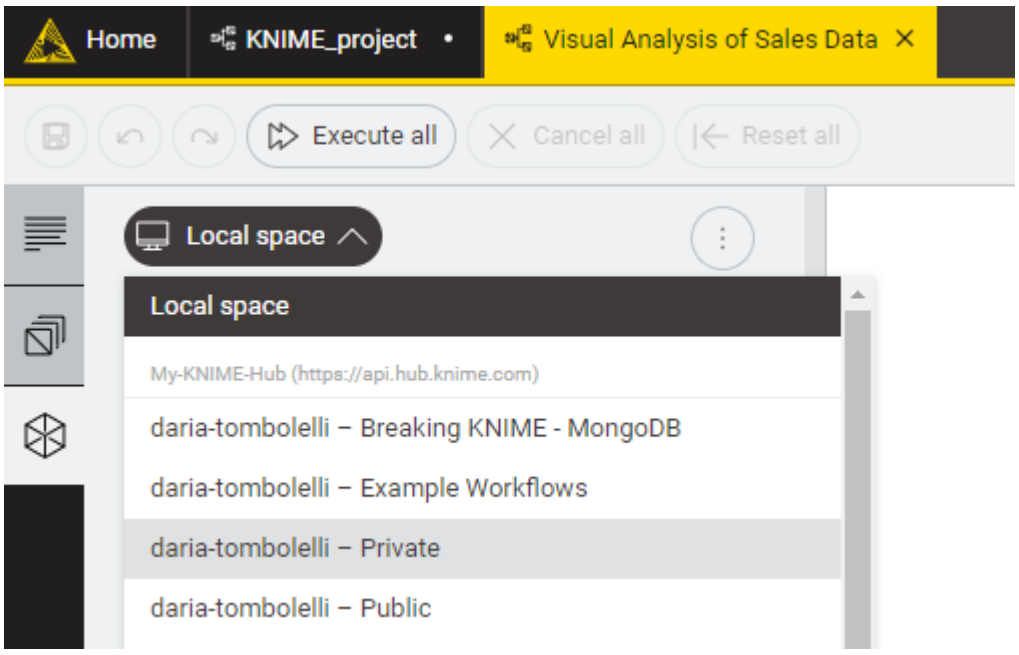



*Figure 16. Select a space to explore*

In the space explorer you can see:

- Workflows
- Folders
- Data files
- Components

- Metanodes

Double click a new empty workflow to open it in the workflow canvas and start adding nodes to the canvas from the node repository.

An overview on components and metanodes is available in the KNIME Components Guide.

Here you can click the three dots to select one of the following actions within the current space:

- Create a new folder or a new workflow
- Import a workflow
- Add a file

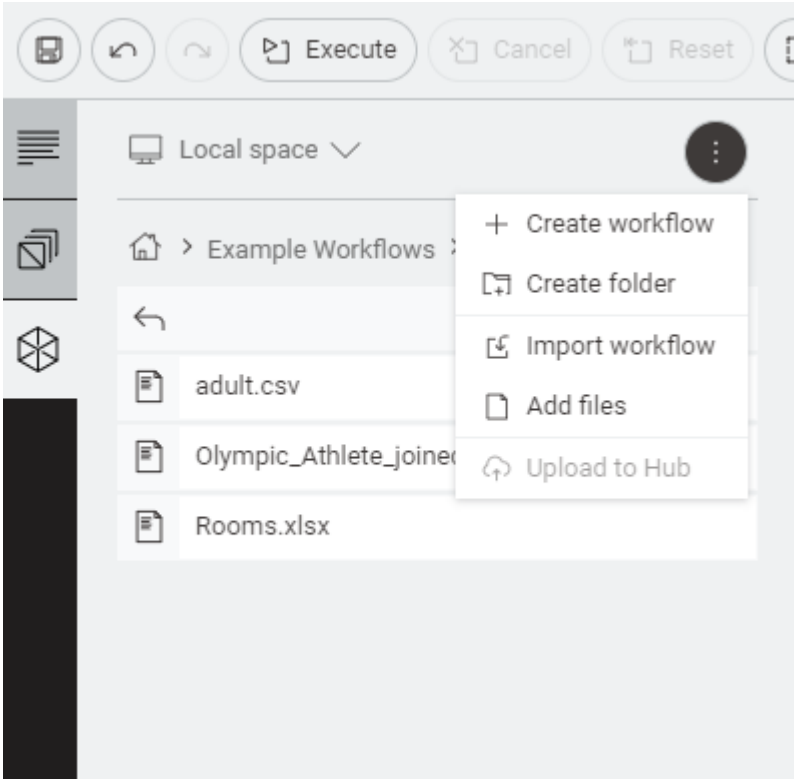



*Figure 17. Space context menu*

Also you can rearrange the items in the space - just drag and drop the item to the position you want to move it to.

You can also drop files to the canvas. KNIME will create the appropriate file reading node automatically and preconfigure it. Finally you can drop a component to the canvas to use the component in the current workflow.

Select an item from the current space and right click on it to access the item context menu. From here you can *Rename*, *Delete*, *Export* (only available for workflows), *Upload to Hub* (if

you are already connected to one of the available Hub mount points) or *Connect to Hub*.

## Building workflows

When you create a new workflow, the canvas will be empty.

To build the workflow you will need to add nodes to it by dragging them from the node repository and connecting them. Alternatively you can drag an output port of a node to show the workflow coach which will suggest you the compatible nodes and directly connect them.

Once two nodes are added to the workflow editor, they can be connected by clicking the output port of the first node and release the mouse at the input port of the second node. Now, the nodes are connected. For some nodes you might have the ability to add specific ports. When hovering over these nodes you will see a + sign appearing. Click it to add a port. If the nodes supports different types of these dynamic ports a list will appear for you to scroll down to select the type of port you want to add.

You can also add a node between two nodes in a workflow. To do so drag the node from the node repository, and release it at its place in the workflow.

## Node repository

Currently installed nodes are available in the node repository. You can add a node from the node repository into the workflow editor by drag and drop, as explained in the section Building Workflows.

Search for a node by typing a search term in the search field on top of the node repository, as shown in Figure 18.

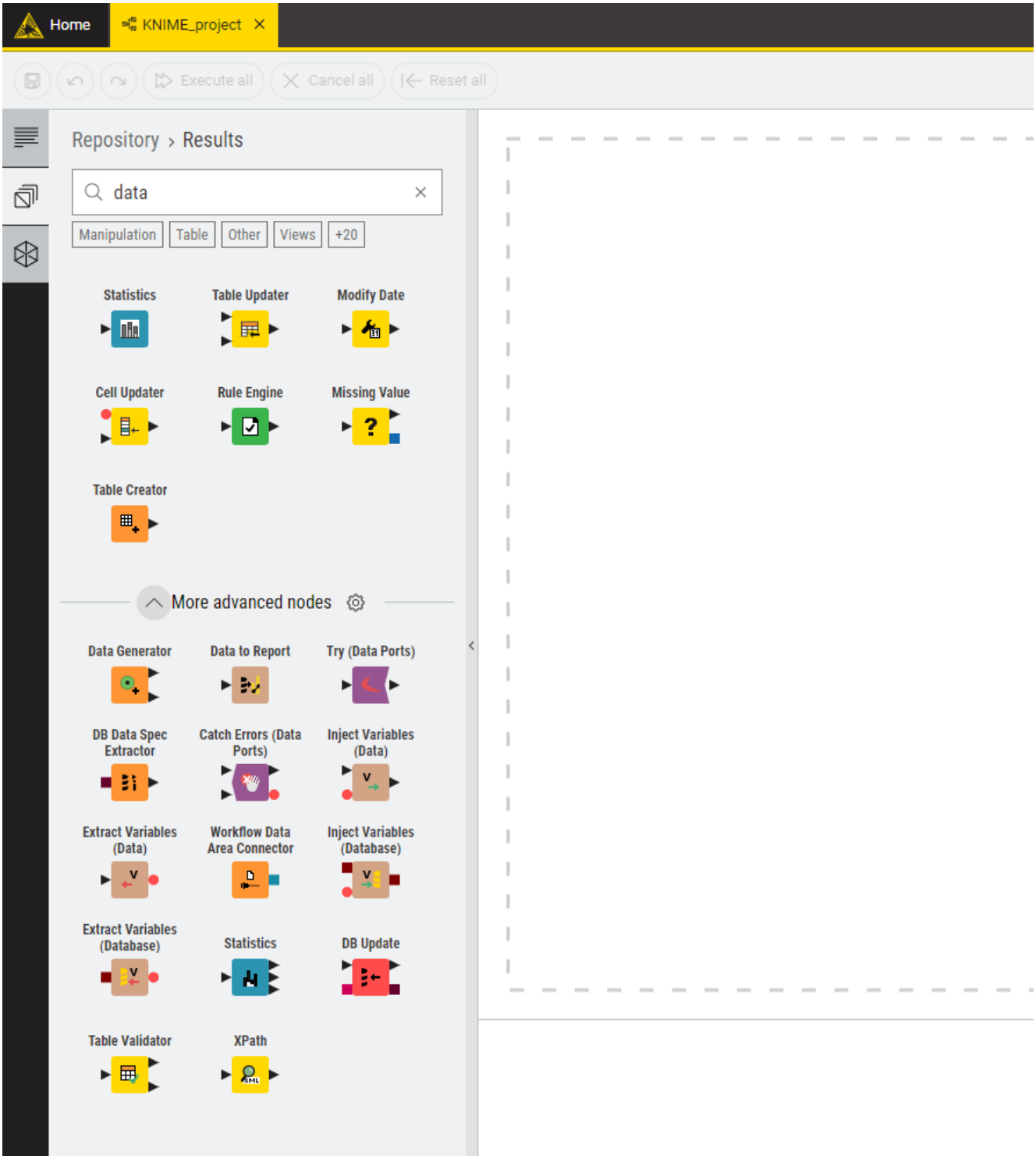



*Figure 18. Node repository with two search modes*

By default a specific set of nodes to help you get started with the KNIME Analytics Platform will be shown. You can expand the search results by clicking *More advanced nodes* in the node repository. Click the cog icon to go to the KNIME Modern UI to change the default of the nodes included in the node repository search results.

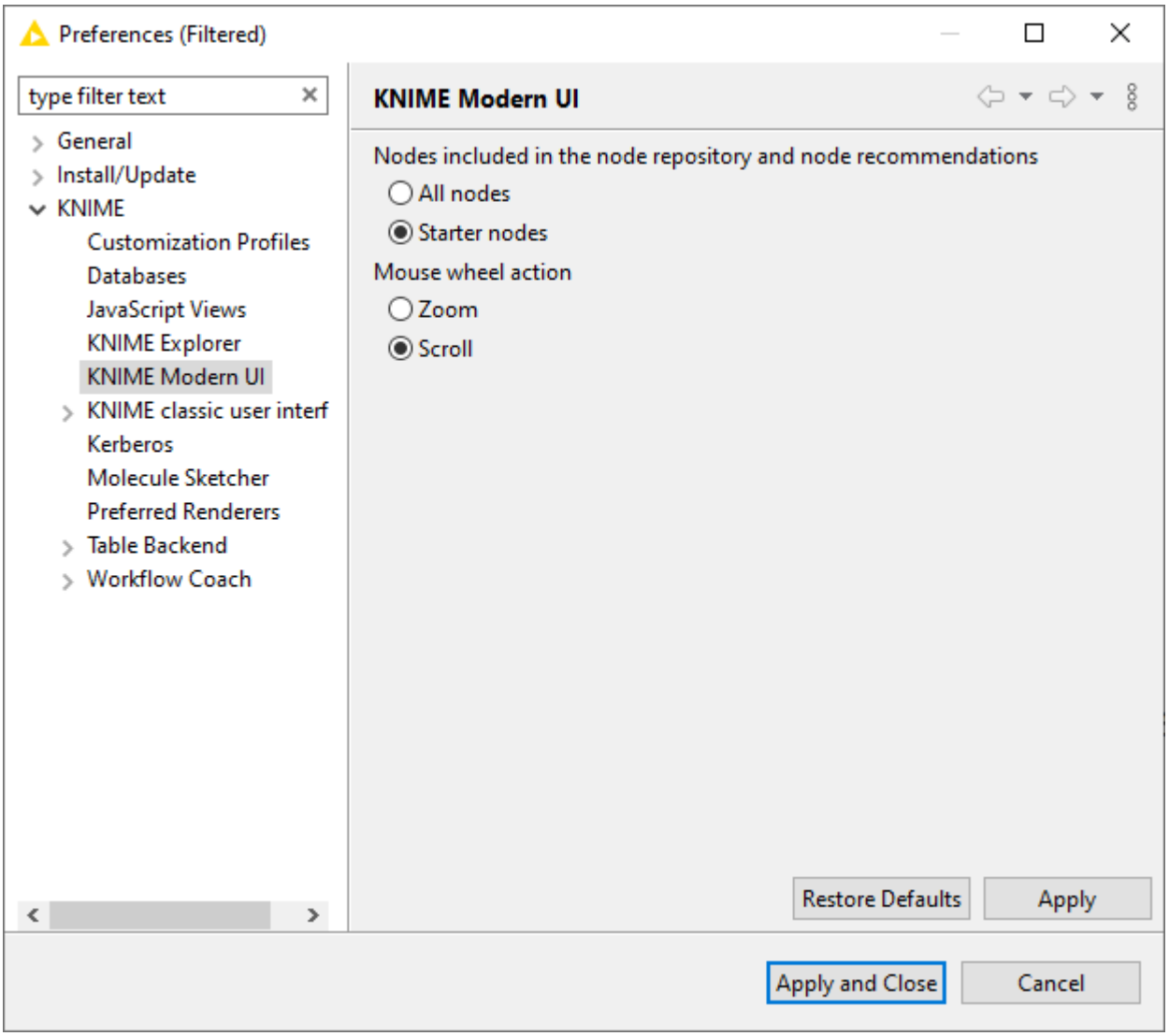



*Figure 19. Change the default of the node search results*

## Node description

You can access the node description, with information about the node function, the node configuration and the different ports available for the node in the following ways:

- Select a node you added in the canvas, go to the side panel navigation and select the first option
- Hover over a node in the node repository and click the info icon that appears. This will open the node description panel.

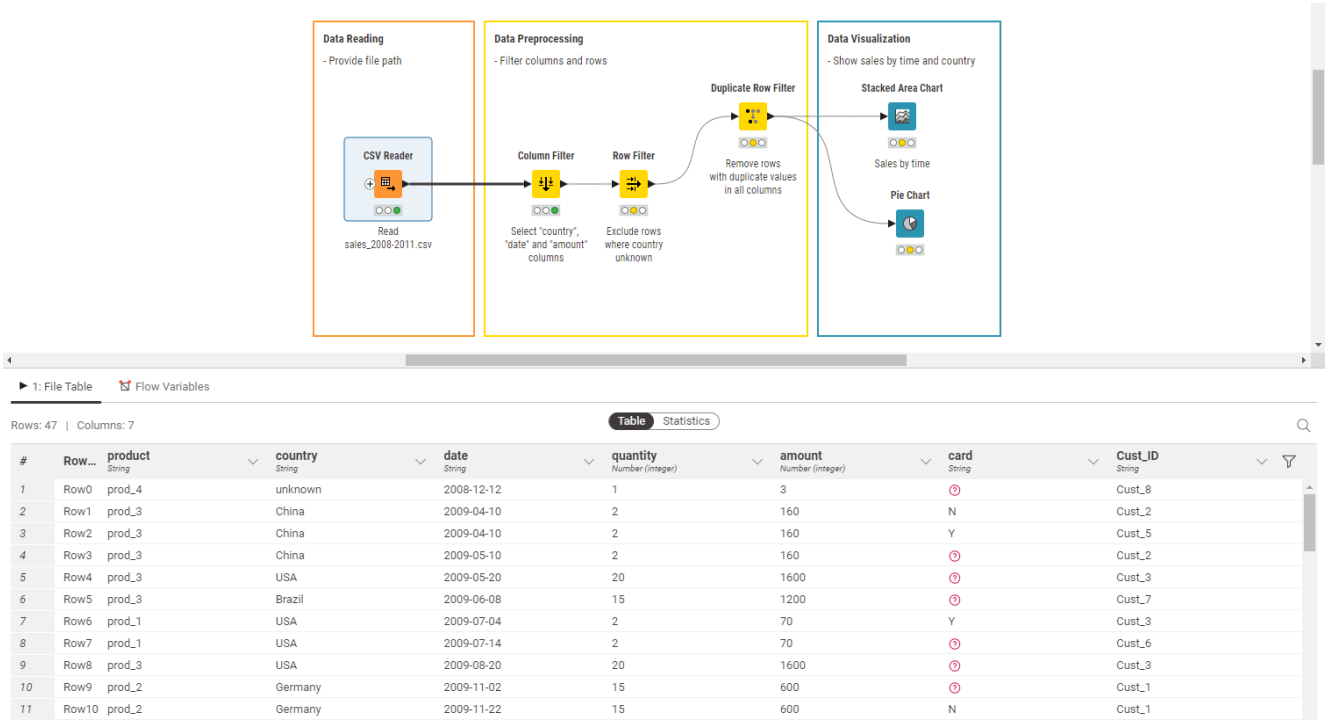
## Workflow description

The description panel on the left of the KNIME Analytics Platform shown in Figure 3 provides a description of the currently active workflow, or a selected component.

Click the pen icon to change the workflow description, add links to external resources and add tags.

## Node monitor

The Node Monitor tab is located on the bottom part of the KNIME Workbench shown in Figure 20. It is especially useful to inspect intermediate output tables in the workflow.




| # | Row... | product String | country String | date String | quantity Number (integer) | amount Number (integer) | card String | Cust_ID String |
|---|---|---|---|---|---|---|---|---|
| 1 | Row0 | prod_4 | unknown | 2008-12-12 | 1 | 3 | ? | Cust_8 |
| 2 | Row1 | prod_3 | China | 2009-04-10 | 2 | 160 | N | Cust_2 |
| 3 | Row2 | prod_3 | China | 2009-04-10 | 2 | 160 | Y | Cust_5 |
| 4 | Row3 | prod_3 | China | 2009-05-10 | 2 | 160 | ? | Cust_2 |
| 5 | Row4 | prod_3 | USA | 2009-05-20 | 20 | 1600 | ? | Cust_3 |
| 6 | Row5 | prod_3 | Brazil | 2009-06-08 | 15 | 1200 | ? | Cust_7 |
| 7 | Row6 | prod_1 | USA | 2009-07-04 | 2 | 70 | Y | Cust_3 |
| 8 | Row7 | prod_1 | USA | 2009-07-14 | 2 | 70 | ? | Cust_6 |
| 9 | Row8 | prod_3 | USA | 2009-08-20 | 20 | 1600 | ? | Cust_3 |
| 10 | Row9 | prod_2 | Germany | 2009-11-02 | 15 | 600 | ? | Cust_1 |
| 11 | Row10 | prod_2 | Germany | 2009-11-22 | 15 | 600 | N | Cust_1 |

*Figure 20. Node Monitor*

Here you can choose to show the flow variables or a preview of the output data at any port of a selected node in the active workflow.

Switch to *Statistics* in order to see some basic statistics of the data.

Read more about the data table shown in the node monitor in the KNIME tables section.

### Info page

By clicking the info icon on the top right corner of the KNIME Analytics Platform you can access to the info page.

Here you can find useful links like:

- The KNIME Forum to ask the community about workflow building, tips and tricks
- The KNIME Community Hub to find workflows, nodes and components, and collaborate in spaces
- Some learning resources like cheat sheets, getting started guide and documentation
- Access to the extensions installation and to check for updates
- Here you can also switch back to KNIME classic user interface

# Customizing the KNIME Workbench

### Reset and logging

When a node is reset, the node status changes from "executed" to "configured" and the output of the node is not available anymore. When saving a workflow in an executed state, the data used in the workflow are saved as well. That is, the larger the dataset, the larger the file size. Therefore, resetting workflows before saving them is recommended in case the dataset can be accessed without any restrictions.

A reset workflow only saves the node configurations, and not any results. However, resetting a node does not undo the operation executed before. All operations done during creation, configuration, and execution of a workflow are reported in the `knime.log` file.

The `knime.log` file is also located in the `knime`-folder inside the `.metadata`-folder, in the KNIME workspace folder defined when launching KNIME Analytics Platform. The `knime.log` file has a limited size, and after reaching it the rows will be overwritten from the top.

# Configuring KNIME Analytics Platform

## Preferences

With the release of KNIME Analytics Platform version 5.1 the preferences have been rearranged. It can not be open, from the Modern User Interface the cog icon on the top right corner of the Analytics Platform.

Here, a list of subcategories is displayed in the dialog that opens. Each category contains a separate dialog for specific settings like database drivers, available update sites, and appearance.

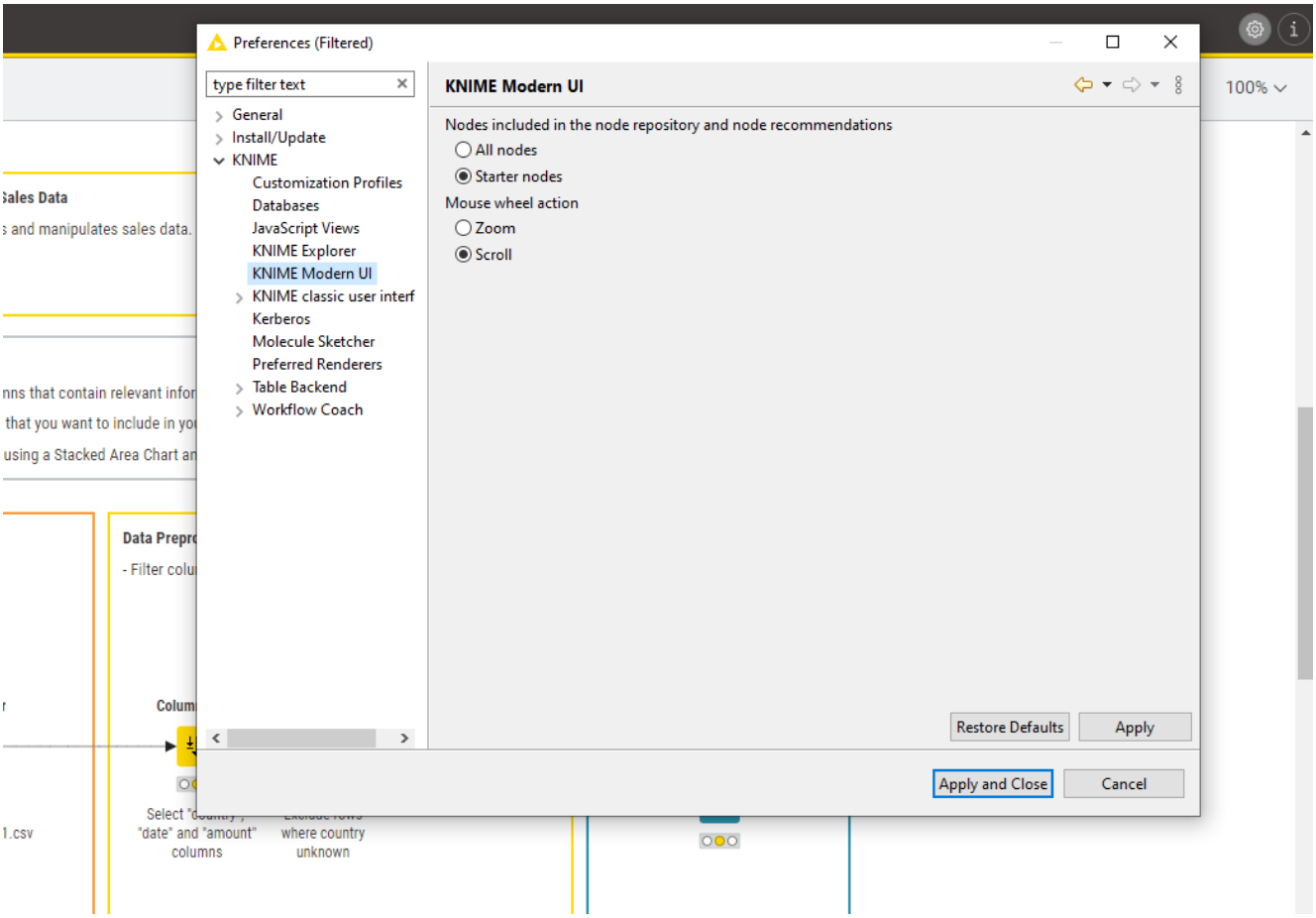



*Figure 21. Open KNIME Preferences window from KNIME Analytics Platform*

## KNIME

Selecting *KNIME* in the list of subcategories, allows you to define the log file log level. By default it is set to *DEBUG*. This log level helps developers to find reasons for any unexpected behavior.

Directly below, you can define the maximum number of threads for all nodes. Separate branches of the workflow are distributed to several threads to optimize the overall execution time. By default the number of threads is set to twice the number of CPUs on the running machine.

In the same dialog, you can also define the folder for temporary files.

Check the last option *Yes, help improve KNIME.* to agree to sending us anonymous usage data. This agreement activates the node recommendations by community in the Workflow Coach.

### KNIME Modern UI

In the KNIME Modern UI category you can:

- Select which nodes to include in the node repository and node recommendations
- Select which action is associated with the mouse wheel.

### KNIME classic user interface

The *KNIME* category, contains a subcategory *KNIME classic user interface*. In this dialog, you can define the console view log level. By default it is set to "WARN", because more detailed information is only useful for diagnosis purposes.

Further below, you can select which confirmation dialogs are shown when using KNIME Analytics Platform. Choose from the following:

- Confirmation after resetting a node
- Deleting a node or connection
- Replacing a connection
- Saving and executing workflow
- Loading workflows created with a nightly build

In the same dialog, you can define what happens if an operation requires executing the previous nodes in the workflow. You have these three options:

- Execute the nodes automatically
- Always reject the node execution
- Show a dialog to execute or not

The following options allow you to define whether workflows should be saved automatically

and after what time interval, also whether linked components and metanodes should be automatically updated. You can also define visual properties such as the border width of workflow annotations.

### Table backend

Starting with KNIME Analytics Platform version 4.3 a new Columnar Backend is introduced, in order to optimize the use of main memory in KNIME Analytics Platform, where cell elements in a table are represented by Java objects by reviewing the underlying data representation.

The KNIME Columnar Table Backend extension addresses these issues by using a different underlying data layer (backed by Apache Arrow), which is based on a columnar representation.

The type of table backend used can be defined:

- As default for all new workflows created. Open the *KNIME Preferences* and select *Table Backend* under KNIME in the left pane of the preferences window. Here you can select *Columnar Backend* as *Table backend for new workflows*, as shown in Figure 22.

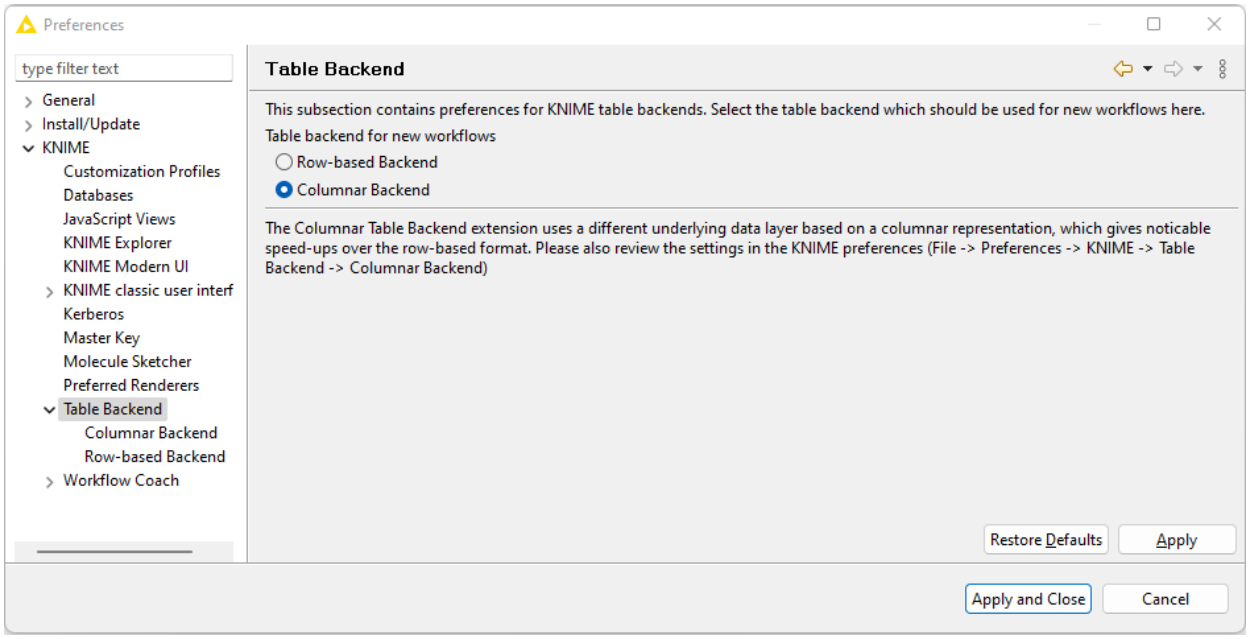



*Figure 22. The Table Backend preference page.*

The parameters relative to memory usage of the Columnar Backend can also be configured. Go to *File → Preferences* and select *Table Backend → Columnar Backend* under KNIME in the left pane of the preferences window, as shown in Figure 23.

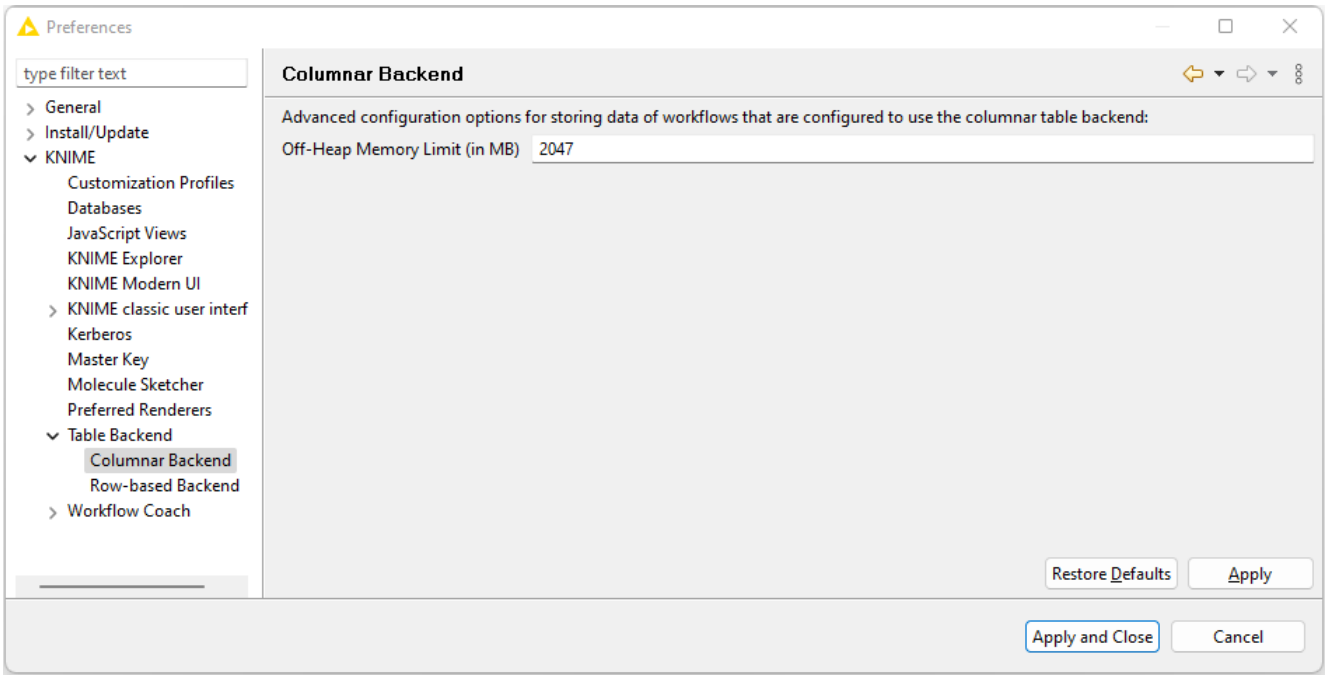



*Figure 23. The Columnar Backend preference page.*

Note that the caches of the Columnar Backend that reside in the off-heap memory region require an amount of memory in addition to whatever memory you have allotted to the heap space of your KNIME's Java Virtual Machine via the `-Xmx` parameter in the `knime.ini`. When altering the sizes of these cache via the preference page, make sure not to exceed your system's physical memory size as otherwise you might encounter system instability or even crashes.

For a more detailed explanation of the Columnar Backend technical background please refer to this post on KNIME Blog.

**High memory usage on Linux:** On some Linux systems KNIME Analytics Platform can allocate more system memory than expected when using the Columnar Backend. This is caused by an unfavorable interaction between the JVM and the glibc native memory allocator. There are multiple options to circumvent this issue.

- Option 1: Reduce the number of allowed malloc areas
    1. Run KNIME Analytics platform with the environment variable `MALLOC_ARENA_MAX` set to 1.
- Option 2: Use jemalloc
    1. Install jemalloc on your OS. For ubuntu: `apt install libjemalloc2` (link to package).
    2. Find the path to jemalloc: `$ ldconfig -p | grep jemalloc`. On Ubuntu 22.04 it is `/lib/x86_64-linux-gnu/libjemalloc.so.2`.

3. Start KNIME Analytics Platform with the environment variable `LD_PRELOAD=<path to libjemalloc.so from step 2>`.

- Option 3: Use tcmalloc
  1. Install tcmalloc on your OS. For ubuntu: `apt install google-preftools`.
  2. Find the path to tcmalloc: `$ ldconfig -p | grep tcmalloc`. On Ubuntu 22.04 it is `/lib/x86_64-linux-gnu/libtcmalloc.so.4`.
  3. Start KNIME Analytics Platform with the environment variable `LD_PRELOAD=<path to libtcmalloc.so from step 2>`.

## Setting up knime.ini

When installing KNIME Analytics Platform, configuration options are set to their defaults. The configuration options, i.e. options used by KNIME Analytics Platform, range from memory settings to system properties required by some extensions.

You can change the default settings in the `knime.ini` file. The `knime.ini` file is located in the installation folder of KNIME Analytics Platform.

To locate the `knime.ini` file on MacOS, open Finder and navigate to the installed Applications.
Next, right click the KNIME application, select *Show Package Contents* in the menu, and navigate to *Contents*, and open *Eclipse*.

Edit the `knime.ini` file with any plaintext editor, such as Notepad (Windows), TextEdit (MacOS) or gedit (Linux).

The entry `-Xmx1024m` in the `knime.ini` file specifies how much memory KNIME Analytics Platform is allowed to use. The setting for this value will depend on how much memory is available in the running machine. We recommend setting it to approximately one half of the available memory, but this value can be modified and personalized. For example, if the computer has 16GB of memory, the entry might be set to `-Xmx8G`.

Besides the memory available, you can define many other settings in the `knime.ini` file. Find an overview of some of the most common settings in Table 1 or in this complete list of the configuration options.

*Table 1. Common configuration settings in knime.ini file*

| Setting | Explanation |
|---|---|
| `-Xmx`<br>• default value: `1024m`<br>• example: `-Xmx16G` | Sets the maximum amount of memory available for KNIME Analytics Platform. |
| `-Dknime.compress.io`<br>• default value: `SNAPPY`<br>• possible values: `[SNAPPY\|GZIP\|NONE]`<br>• example: `-Dknime.compress.io=SNAPPY` | Determines which compression algorithm (if any) to use when writing temporary tables to disk. |
| `-Dorg.knime.container.cellsinmemory`<br>• default value: 5,000<br>• possible values: any value between 0 and 2,147,483,647<br>• example: `-Dorg.knime.container.cellsinmemory=100,000` | This setting defines the size of a "small table". Small tables are attempted to be kept in memory, independent of the Table Caching strategy. By increasing the size of a small table, the number of swaps to the disk can be limited, which comes at the cost of reducing memory space available for other operations. |
| `-Dknime.layout_editor.browser`<br>• default value version 4.7.2: swt<br>• possible values: `[cef\|swt]`<br>• example: -Dknime.layout_editor.browser=cef | This setting defines which browser should be used to display the layout editor. |

| Setting | Explanation |
| --- | --- |
| `-Dknime.table.cache`<br>• default value: `LRU`<br>• possible values: `[LRU\|SMALL]`<br>• example: `-Dknime.table.cache=SMALL` | Determines whether to attempt to cache large tables (i.e., tables that are not considered to be "small"; see setting `-Dorg.knime.container.cellsinmemory`) in memory. If set to `LRU`, large tables are evicted from memory in least-recently used (LRU) order or when memory becomes scarce. If set to `SMALL`, large tables are always flushed to disk. |
| `-Dknime.url.timeout`<br>• default value: 1,000 ms<br>• example: `-Dknime.url.timeout=100` | When trying to connect or read data from an URL, this value defines a timeout for the request. Increase the value if a reader node fails. A too high timeout value may lead to slow websites blocking dialogs in KNIME Analytics Platform. |
| `-Dchromium.block_all_external_requests`<br>• default value: false<br>• example: `-Dchromium.block_all_external_requests=true` | This configuration setting when set to `true` blocks all the external requests made by Chrmium Embedded Framework. |

## KNIME runtime options

KNIME's runtime behavior can be configured in various ways by passing options on the command line during startup. Since KNIME is based on Eclipse, all Eclipse runtime options also apply to KNIME.

KNIME also adds additional options, which are described below.

### Command line arguments

Listed below are the command line arguments processed by KNIME. They can either be specified permanently in the `knime.ini` in the root of the KNIME installation, or be passed to the KNIME executable. Please note that command line arguments must be specified *before* the system properties (see below) i.e. before the `-vmargs` parameter.

Note that headless KNIME applications, such as the batch executor, offer quite a few command line arguments. They are not described here but are printed if you call the application without any arguments.

`-checkForUpdates`

If this arguments is used, KNIME automatically checks for updates during startup. If new versions of installed features are found, the user will be prompted to install them. A restart is required after updates have been installed.

## Java system properties

Listed below are the Java system properties with which KNIME's behavior can be changed. They can either be specified permanently in the `knime.ini` in the root of the KNIME installation, or be passed to the KNIME executable. Please note that system properties must be specified *after* the `-vmargs` parameter. The required format is `-DpropName=propValue`.

## General properties

`org.knime.core.maxThreads=<number>`

Sets the maximum number of threads that KNIME is using for executing nodes. By default this number is 1.5 times the number of cores. This property overrides the value from the KNIME preference page.

`knime.tmpdir=<directory>`

Sets the default directory for temporary files KNIME files (such as data files). This property overrides the value from the preference pages and is by default the same as the `java.io.tmpdir` .

`knime.synchronous.io=(true|false)`

Can be used to enforce the sequential processing of rows for KNIME tables. By default, each table container processes its rows asynchronously in a number of (potentially re-used) threads. The default value is `false`. Setting this field to `true` will instruct KNIME to always handle rows sequentially and synchronously, which in some cases may be slower.

`knime.async.io.cachesize=<number>`

Sets the batch size for non-sequential and asynchronous handling of rows (see knime.synchronous.io). It specifies the amount of data rows that are handled by a single container thread. The larger the buffer, the smaller the synchronization overhead but the larger the memory requirements. This property has no effect if rows are handled sequentially. The default value is 10.

`knime.domain.valuecount=<number>`

The number of nominal values kept in the domain when adding rows to a table. This is only the default and may be overruled by individual node implementations.If no value is specified a default of 60 will be used.

`org.knime.container.threads.total=<number>`

Sets the maximum number of threads that can be used to write KNIME native output tables. By default this number equals the number of processors available to the JVM. *Note:* This value has to be greater than 0.

`org.knime.container.threads.instance=<number>`

Sets the maximum number of threads that can be used to write a *single* KNIME native output table. By default this number equals the number of processors available to the JVM. *Note:* This value has to be greater than 0 and cannot be larger than `org.knime.container.threads.total`.

`knime.discourage.gc=(true|false)`

If set to true, discourages KNIME from triggering a full stop-the-world garbage collection. Note that (a) individual nodes are allowed to disregard this setting and (b) the garbage collector may independently decide that a full stop-the-world garbage collection is warranted. Set to true by default.

`org.knime.container.minspace.temp=<number>`

Java property to specify the minimum free disc space in MB that needs to be available.
If less is available, no further table files & blobs will be created (resulting in an exception).

`knime.columnar.chunksize=<number>`

The columnar table backend horizontally divides tables into batches and vertically divides these batches into column chunks. This property controls the initial size of these chunks and thereby the number of rows per batch. A chunk is the smallest unit that must be materialized to access a single value. Changing this value can therefore impact memory footprint and overall performance. Do not change this value unless you have good reasons. The default value is 28,000.

`knime.columnar.reservedmemorymb=<number>`

The columnar table backend caches table data off-heap. To this end, it requires memory in addition to the JVM's heap memory, whose size is controlled via the -Xmx parameter. If no explicit cache sizes are set in the preferences, the default memory available for caching is computed as follows: Total physical memory minus reserved memory minus 1.25 times heap memory. The reserved memory size in this equation (in MB) can be configured via this property. The default is 4,096.

`knime.columnar.verbose=(true|false)`

Setting this property to true activates verbose debug logging in the columnar table backend.
-

`knime.disable.rowid.duplicatecheck=(true|false)`

Enables/disables row ID duplicate checks on tables. Tables in KNIME are supposed to have unique IDs, whereby the uniqueness is asserted using a duplicate checker. This property will disable this check.
**Warning:** This property should not be changed by the user.

`knime.disable.vmfilelock=(true|false)`

Enables/disables workflow locks. As of KNIME 2.4 workflows will be locked when opened; this property will disable the locking (allowing multiple instances to have the same workflow open).
**Warning:** This property should not be changed by the user.

`knime.database.timeout=<number>`

Sets the timeout in seconds trying to establish a connection to a database.
The default value is 15 seconds.

`knime.database.fetchsize=<number>`

Sets the fetch size for retrieving data from a database.
The default value depends on the used JDBC driver.

`knime.database.batch_write_size=<number>`

Sets the batch write size for writing data rows into a database.
The default value is 1, that is one row at a time.

`knime.database.enable.concurrency=(true|false)`

Used to switch on/off the database connection access (applies only for the same database connection).
Default is true, that is all database accesses are synchronized based on single connection; false means off, that is, the access is not synchronized and may lead to database errors.

`knime.logfile.maxsize=<number>[mk]`

Allows one to change the maximum log file size (default is 10 MB).
Values must be integer, possibly succeeded by "m" or "k" to denote that the given value is in mega or kilo byte.

`knime.settings.passwords.forbidden=(true|false)`

If *true*, nodes using passwords as part of their configuration (e.g. DB connection or SendEmail) will not store the password as part of the workflow on disc. Instead a null value is stored, which will cause the node's configuration to be incorrect (but valid) after the workflow is restored from disc. Default is *false*.

`knime.repository.non-instant-search=(true|false)`

Allows to disable the live update in the node repository search.
-

`knime.macosx.dialogworkaround=(true|false)`

Allows to disable the workaround for freezes when opening node dialogs under MacOSX.
-

`knime.data.bitvector.maxDisplayBits=<number>`

Sets the maximum number of bits that are display in string representations of bit vectors.
-

`knime.xml.disable_external_entities=(true|false)`

If set to true, all nodes that parse XML files will not read external entities defined via a DTD.
This is usually only useful when running as an executor on the server and you want prevent XXE attacks.

## Plug-in dependent properties

These properties only affect some plug-ins and are only applicable if they are installed.

`org.knime.cmlminblobsize=<number>[mMkK]`

Allows to change the minimum size in bytes (or kilobyte or megabytes) a CML molecule must have before it is stored in a blob cell. Otherwise it is stored inline. The latter is a bit faster but needs more memory. The default is 8kB.

`org.knime.ctabminblobsize=<number>[mMkK]`

Allows to change the minimum size in bytes (or kilobyte or megabytes) a Ctab molecule must have before it is stored in a blob cell. Otherwise it is stored inline. The latter is a bit faster but needs more memory. The default is 8kB.

`org.knime.mol2minblobsize=<number>[mMkK]`

Allows to change the minimum size in bytes (or kilobyte or megabytes) a Mol2 molecule must have before it is stored in a blob cell. Otherwise it is stored inline. The latter is a bit faster but needs more memory. The default is 8kB.

`org.knime.molminblobsize=<number>[mMkK]`

Allows to change the minimum size in bytes (or kilobyte or megabytes) a Mol molecule must have before it is stored in a blob cell. Otherwise it is stored inline. The latter is a bit faster but needs more memory. The default is 8kB.

`org.knime.rxnminblobsize=<number>[mMkK]`

Allows to change the minimum size in bytes (or kilobyte or megabytes) a Rxn molecule must have before it is stored in a blob cell. Otherwise it is stored inline. The latter is a bit faster but needs more memory. The default is 8kB.

`org.knime.sdfminblobsize=<number>[mMkK]`

Allows to change the minimum size in bytes (or kilobyte or megabytes) a SDF molecule must have before it is stored in a blob cell. Otherwise it is stored inline. The latter is a bit faster but needs more memory. The default is 8kB.

# KNIME tables

## Data table

Very common input and output ports of nodes are data input ports and data output ports, which correspond to the black triangles in Figure 24.

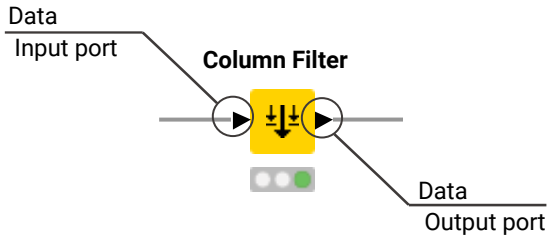


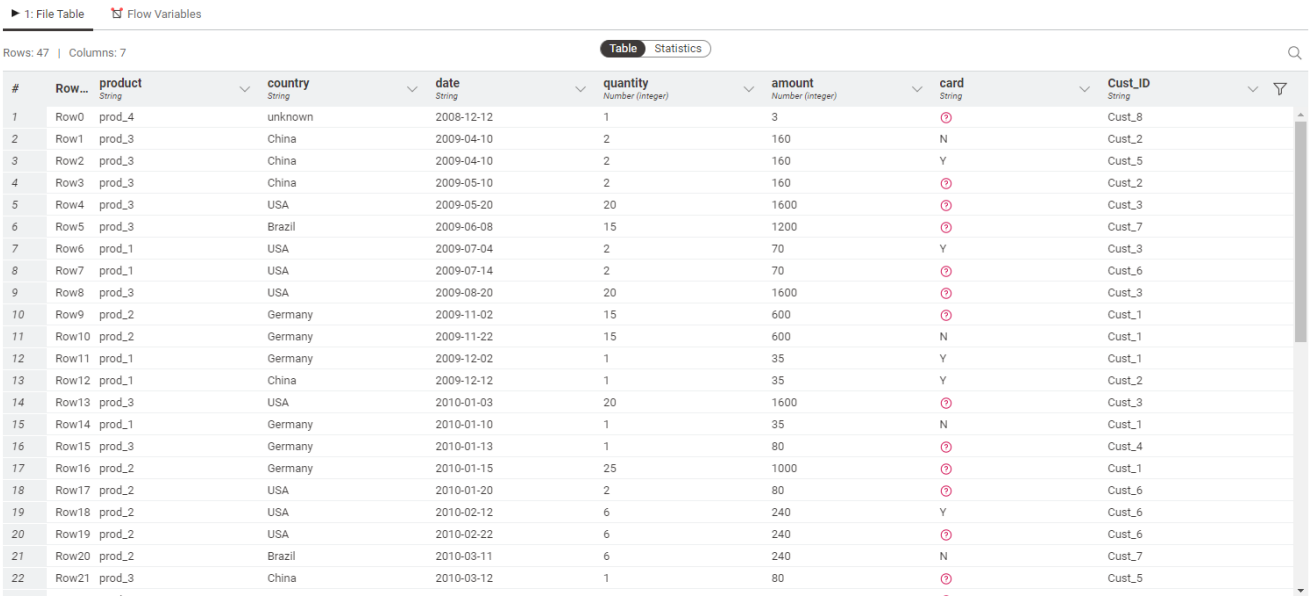

*Figure 24. Data input and output port*

A data table is organized by columns and rows, and it contains a number of equal-length rows. Elements in each column must have the same data type.

The data table shown in Figure 25 is produced by a CSV Reader node, which is one of the many nodes with a black triangle output port for data output. To open the table, click the node. Execute the node if it is not yet execute. The table will show up in the node monitor.

The output table has row numbers, unique RowIDs and column headers. The RowIDs are automatically created by the reader node, but they can also be defined manually. The RowIDs and the column headers can therefore be used to identify each data cell in the table. Missing values in the data are shown by a red question mark in a circle.

At the top of the node monitor you can select which output port you want to view via the tabs and the flow variable tab, which shows the available flow variables in the node output and their current values. Next row will indicate the table dimensions, meaning how many rows and how many columns there are in the table at that specific output port. Here you can also use the toggle to switch to *Statistics*. This tab shows the meta information of the table, like the columns names, columns types and some other statistics data.



1: File Table | Flow Variables

Rows: 47 | Columns: 7 — Table | Statistics

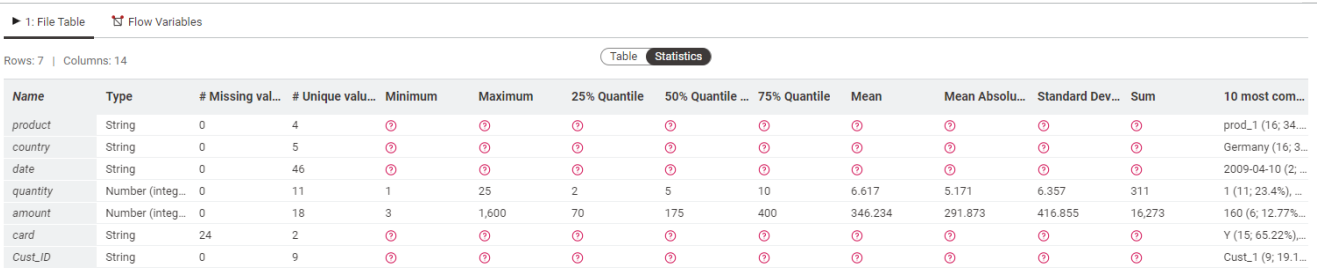
| # | Row... | product (String) | country (String) | date (String) | quantity (Number (integer)) | amount (Number (integer)) | card (String) | Cust_ID (String) |
|---|---|---|---|---|---|---|---|---|
| 1 | Row0 | prod_4 | unknown | 2008-12-12 | 1 | 3 | ? | Cust_8 |
| 2 | Row1 | prod_3 | China | 2009-04-10 | 2 | 160 | N | Cust_2 |
| 3 | Row2 | prod_3 | China | 2009-04-10 | 2 | 160 | Y | Cust_5 |
| 4 | Row3 | prod_3 | China | 2009-05-10 | 2 | 160 | ? | Cust_2 |
| 5 | Row4 | prod_3 | USA | 2009-05-20 | 20 | 1600 | ? | Cust_3 |
| 6 | Row5 | prod_3 | Brazil | 2009-06-08 | 15 | 1200 | ? | Cust_7 |
| 7 | Row6 | prod_1 | USA | 2009-07-04 | 2 | 70 | Y | Cust_3 |
| 8 | Row7 | prod_1 | USA | 2009-07-14 | 2 | 70 | ? | Cust_6 |
| 9 | Row8 | prod_3 | USA | 2009-08-20 | 20 | 1600 | ? | Cust_3 |
| 10 | Row9 | prod_2 | Germany | 2009-11-02 | 15 | 600 | ? | Cust_1 |
| 11 | Row10 | prod_2 | Germany | 2009-11-22 | 15 | 600 | N | Cust_1 |
| 12 | Row11 | prod_1 | Germany | 2009-12-02 | 1 | 35 | Y | Cust_1 |
| 13 | Row12 | prod_1 | China | 2009-12-12 | 1 | 35 | Y | Cust_2 |
| 14 | Row13 | prod_3 | USA | 2010-01-03 | 20 | 1600 | ? | Cust_3 |
| 15 | Row14 | prod_1 | Germany | 2010-01-10 | 1 | 35 | N | Cust_1 |
| 16 | Row15 | prod_3 | Germany | 2010-01-13 | 1 | 80 | ? | Cust_4 |
| 17 | Row16 | prod_2 | Germany | 2010-01-15 | 25 | 1000 | ? | Cust_1 |
| 18 | Row17 | prod_2 | USA | 2010-01-20 | 2 | 80 | ? | Cust_6 |
| 19 | Row18 | prod_2 | USA | 2010-02-12 | 6 | 240 | Y | Cust_6 |
| 20 | Row19 | prod_2 | USA | 2010-02-22 | 6 | 240 | ? | Cust_6 |
| 21 | Row20 | prod_2 | Brazil | 2010-03-11 | 6 | 240 | N | Cust_7 |
| 22 | Row21 | prod_3 | China | 2010-03-12 | 1 | 80 | ? | Cust_5 |

*Figure 25. Data output in KNIME Analytics Platform*

## Column types

The basic data types in KNIME Analytics Platform are `Integer`, `Double`, and `String`, along with other supported data types such as `Long`, `Boolean` value, `JSON`, `URI`, `Document`, `Date&Time`, `Bit vector`, `Image`, and `Blob`. KNIME Analytics Platform also supports customized data types, for example, a representation of a molecule.

Switch to the *Statistics* view in an output table, to see the data types of the columns in the data table, as shown in Figure 26. For numerical values, only the range of the values in the data is shown. For string values, the different values appearing in the data are shown.



1: File Table | Flow Variables

Rows: 7 | Columns: 14 — Table | Statistics

| Name | Type | # Missing val... | # Unique valu... | Minimum | Maximum | 25% Quantile | 50% Quantile ... | 75% Quantile | Mean | Mean Absolu... | Standard Dev... | Sum | 10 most com... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| product | String | 0 | 4 | ? | ? | ? | ? | ? | ? | ? | ? | ? | prod_1 (16; 34.... |
| country | String | 0 | 5 | ? | ? | ? | ? | ? | ? | ? | ? | ? | Germany (16; 3... |
| date | String | 0 | 46 | ? | ? | ? | ? | ? | ? | ? | ? | ? | 2009-04-10 (2; ... |
| quantity | Number (integ... | 0 | 11 | 1 | 25 | 2 | 5 | 10 | 6.617 | 5.171 | 6.357 | 311 | 1 (11; 23.4%), ... |
| amount | Number (integ... | 0 | 18 | 3 | 1,600 | 70 | 175 | 400 | 346.234 | 291.873 | 416.855 | 16,273 | 160 (6; 12.77%... |
| card | String | 24 | 2 | ? | ? | ? | ? | ? | ? | ? | ? | ? | Y (15; 65.22%),... |
| Cust_ID | String | 0 | 9 | ? | ? | ? | ? | ? | ? | ? | ? | ? | Cust_1 (9; 19.1... |

*Figure 26. Data types and data domain in "Spec" tab*

The reader nodes in KNIME Analytics Platform assign a data type to each column based on their interpretation of the content. If the correct data type of a column is not recognized by the reader node, the data type can be corrected afterwards. There are nodes available to convert data types. For example: String to Number, Number to String, Double to Int, String to Date&Time, String to JSON, and String to URI.

Many of the special data types are recognized as `String` by the reader nodes. To convert

these `String` columns to their correct data types, use the Column Type Auto Cast node.

When you use the File Reader node to read a file you can convert the column types directly via the node configuration dialog. To do so go to the Transformation tab in the configuration dialog and change the type of the desired column, as shown in Figure 27.

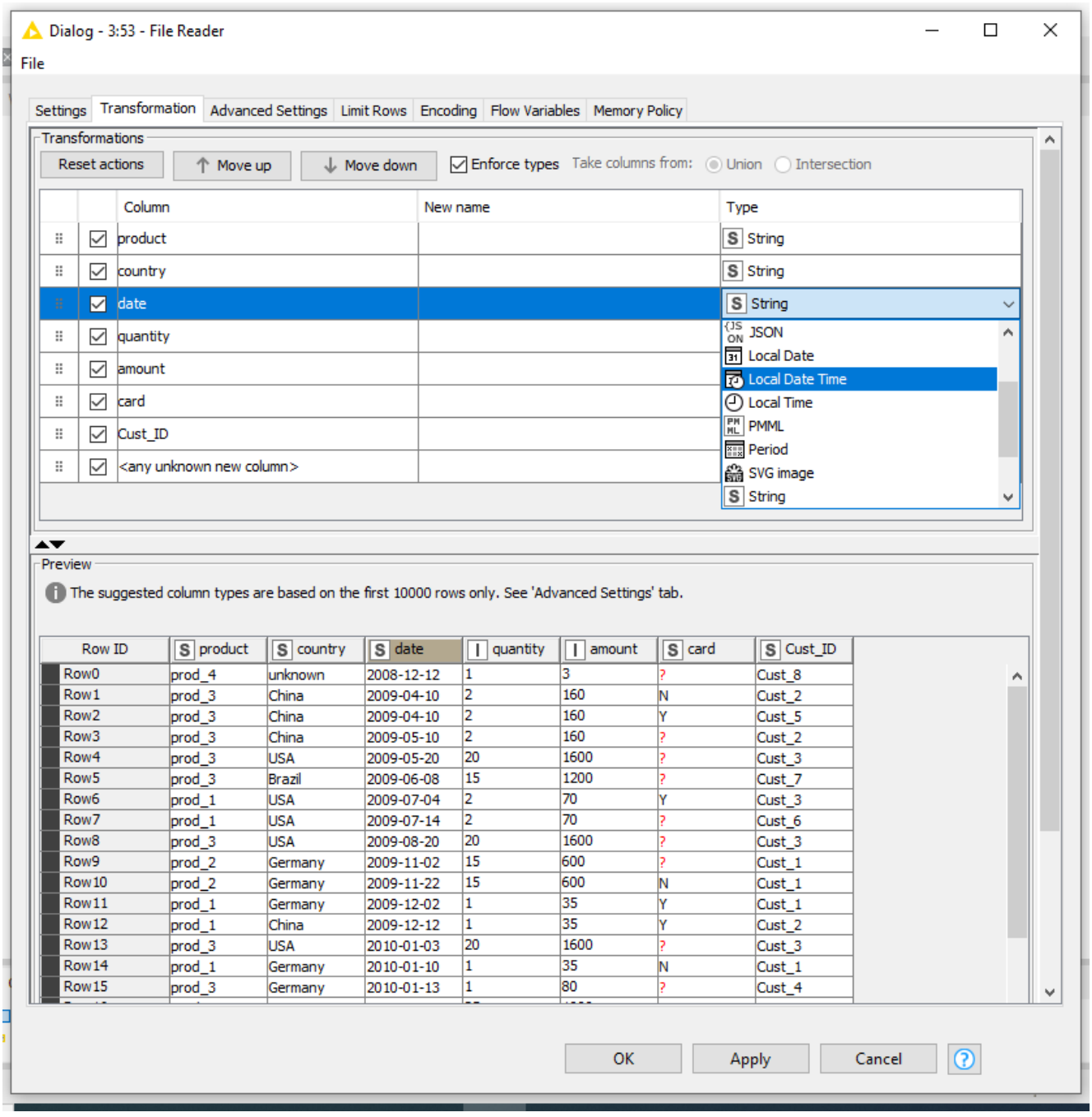



*Figure 27. Change column type in File Reader node*

## Sorting

Rows in the table view output can be sorted by values in one column by clicking the up (ascending) and down (descending) arrow that appears hovering over the column name in the header. Note that this sorting only affects the current output view and has no effect on the node output.

To sort rows in an output table permanently, use the Sorter node. Use the Column Resorter node to reorder columns.

## Column rendering

In a table view output, you can also change the way in which numeric values are displayed in a data table. For example, it is possible to display numeric values as percentages, with full precision, or replace digits by a grey scale or bars. To see these and other rendering options for a column, click the carat icon in the column header, and select the desired available renderer, as shown in Figure 28. Note that these changes are temporary and have no effect on the node output.

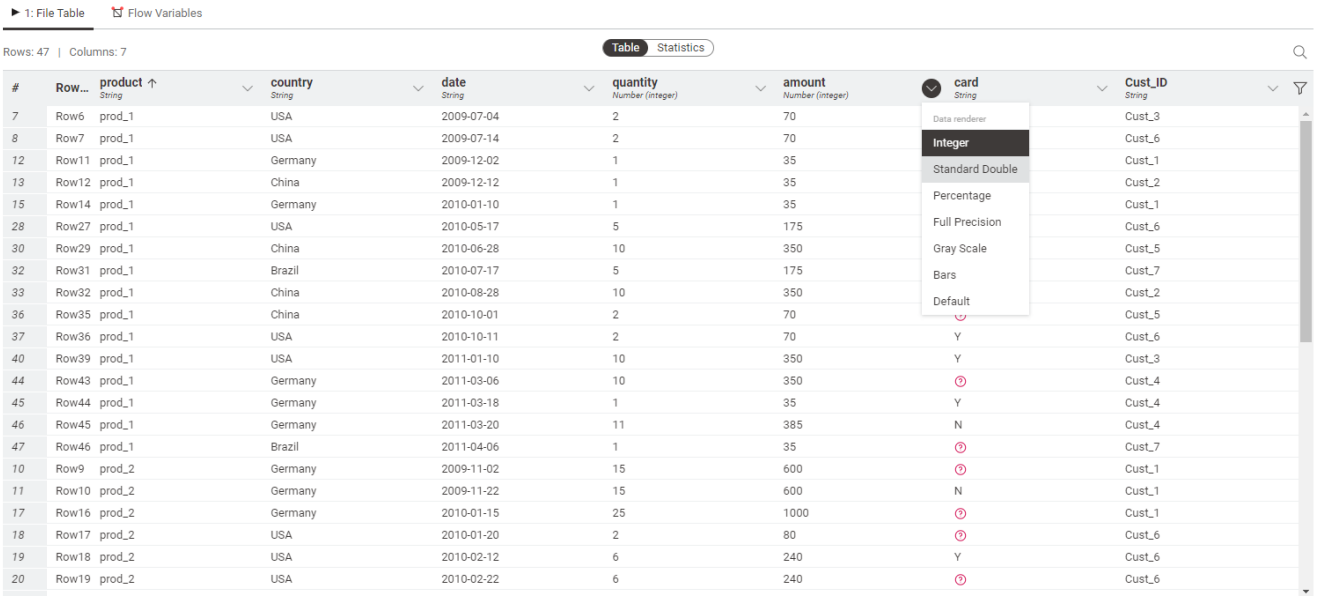



*Figure 28. Rendering data in table view*

## Table storage

When executed, many KNIME nodes generate and provide access to tabular data at their output ports. These tables might be small or large and, therefore, might fit into the main memory of the executing machine or not. Several options are available for configuring which tables to hold in memory as well as when and how to write tables to disk. These options are outlined in this section.

### In-memory caching

KNIME Analytics Platform differentiates between small and large tables. Tables are considered to be small (large) when they are composed of up to (more than) 5000 cells. This threshold of 5000 cells can be adjusted via the `-Dorg.knime.container.cellsinmemory` parameter in the `knime.ini` file. KNIME Analytics Platform always attempts to hold small tables in memory, flushing them to disk only when memory becomes scarce.

In addition, KNIME Analytics Platform attempts to keep recently used large tables in memory while sufficient memory is available. However, it writes these tables asynchronously to disk in the background, such that they can be dropped from memory when they have not been accessed for some time or when memory becomes scarce. You can configure the memory consumption of a specific node to never attempt to hold its tables in memory and, instead, write them to disk on execution. This is helpful if you know that a node will generate a table that cannot be held in memory or if you want to reduce the memory footprint of a node.

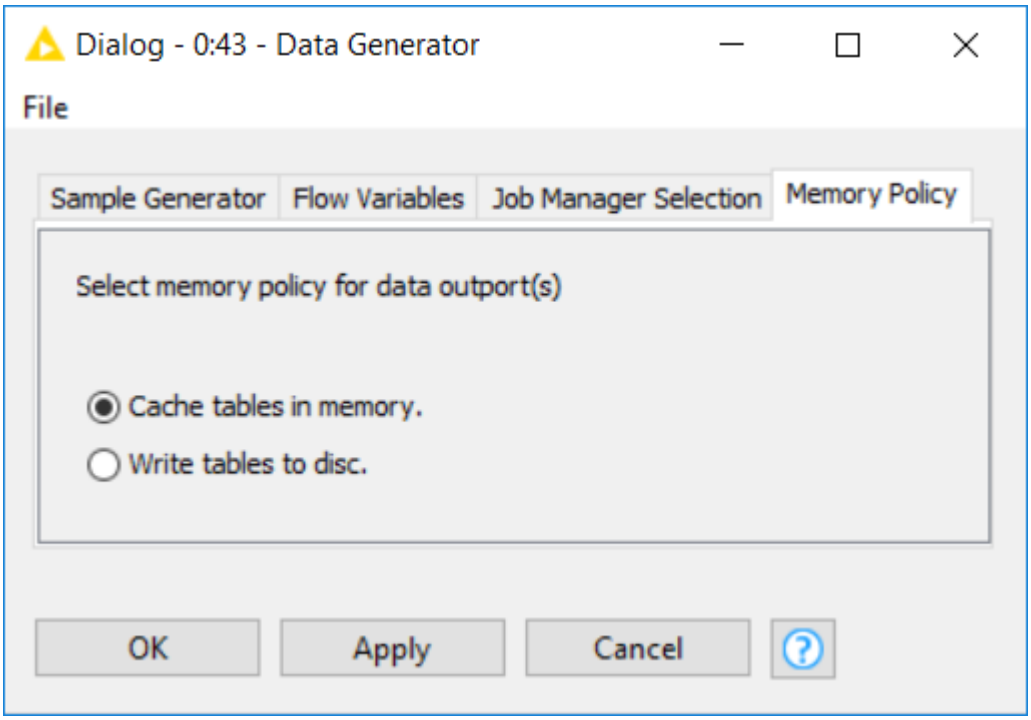



*Figure 29. Configuring a node's memory policy*

Alternatively, by putting the line `-Dknime.table.cache=SMALL` into the `knime.ini` file, KNIME Analytics Platform can be globally configured to use a less memory-consuming, albeit much slower caching strategy. This strategy only ever keeps small tables in memory.

### Disk storage

KNIME Analytics Platform compresses tables written to disk to reduce the amount of occupied disk space. By default, KNIME Analytics Platform uses the Snappy compression algorithm to compress its tables. However, you can configure KNIME Analytics Platform to use GZIP compression or no compression scheme at all via the `-Dknime.compress.io` parameter in the `knime.ini` file.

### Columnar Backend

Starting with KNIME Analytics Platform version 4.3 a new Columnar Backend is introduced. This extension addresses these issues by using a different underlying data layer (backed by Apache Arrow), which is based on a columnar representation.

For information on how to set up this type of backend please refer to the Table backend section.

# Shortcuts

The listed shortcuts are only working inside the KNIME Modern UI Preview. They cannot be changed at the moment. The eclipse preferences have no impact on them.

## General actions

*Table 2. The supported shortcuts*

| Action | Mac | Windows & Linux |
|---|---|---|
| Create workflow | ⌘ N | Ctrl + N |
| Open workflow | ⌘ O | Ctrl + O |
| Close workflow | ⌘ W | Ctrl + W |
| Save workflow | ⌘ S | Ctrl + S |
| Copy | ⌘ C | Ctrl + C |
| Cut | ⌘ X | Ctrl + X |
| Paste | ⌘ V | Ctrl + V |
| Undo | ⌘ Z | Ctrl + Z |
| Redo | ⌘ ⇧ Z | Ctrl + Shift + Z |
| Delete | ⌫<br>⌦ | Delete |

## Execution

*Table 3. The supported shortucts for execute nodes, reset and cancel node execution*

| Action | Mac | Windows & Linux |
|---|---|---|
| Execute | F7 | F7 |
| Cancel | F9 | F9 |
| Reset | F8 | F8 |
| Execute all nodes | ⇧ F7 | Shift + F7 |
| Cancel all nodes | ⇧ F9 | Shift + F9 |
| Reset all nodes | ⇧ F8 | Shift + F8 |
| Start step loop execution | ⌥ ⇧ F6 | Ctrl + Alt + F6 |
| Pause step loop execution | ⌥ ⇧ F7 | Ctrl + Alt + F7 |
| Resume step loop execution | ⌥ ⇧ F8 | Ctrl + Alt + F8 |

## Zooming and Panning

*Table 4. The supported shortcuts related to zooming and panning*

| Action | Mac | Windows & Linux |
|---|---|---|
| Fit to screen | ⌘ 1 | Ctrl + 1 |
| Fill entire screen | ⌘ 2 | Ctrl + 2 |
| 100% | ⌘ 0 | Ctrl 0 |

| Action | Mac | Windows & Linux |
|---|---|---|
| Zoom in | ⌘ \+ | Ctrl \+ |
| Zoom out | ⌘ - | Ctrl - |
| Panning workflow canvas | *hold* “SPACE” — *while pressing left mouse button canvas can be moved around*<br><br>right or middle click and move | *hold* “SPACE” — *while pressing left mouse button canvas can be moved around*<br><br>right or middle click and move |

## Component and metanode building

*Table 5. The supported shortcuts related to build components and metanodes*

| Action | Mac | Windows & Linux |
|---|---|---|
| Create component | ⌘ J | Ctrl + J |
| Expand component | ⌘ ⇧ J | Ctrl + Shift + J |
| Create metanode | ⌘ G | Ctrl + G |
| Expand metanode | ⌘ ⇧ G | Ctrl + Shift + G |
| Rename component or metanode | ⌘ F2 | Shift F2 |
| Open component layout editor | ⌘ D | Ctrl + D |
| Enter component or metanode | ⌘ ⌥ ⇧ | Ctrl + Alt + ⇧ |

| Action | Mac | Windows & Linux |
|---|---|---|
| Leave component or metanoder | ⌘ ⌥ ⇧ ⇧ | Ctrl + Alt + Shift + ⇧ |

## Node labels

*Table 6. The supported shortcuts related to add workflow documentation via node labels*

| Action | Mac | Windows & Linux |
|---|---|---|
| Edit node label | F2 | F2 |
| Apply changes and leave edit mode | ⌘ ⇧ | Ctrl + ⇧ |

## Workflow annotations

*Table 7. The supported shortcuts related to add workflow documentation via formated workflow annotations*

| Action | Mac | Windows & Linux |
|---|---|---|
| Normal text | ⌘ ⌥ 0 | Ctrl + Alt + 0 |
| Headline 1 | ⌘ ⌥ 1 | Ctrl + Alt + 1 |
| Headline 2 | ⌘ ⌥ 2 | Ctrl + Alt + 2 |
| Headline 3 | ⌘ ⌥ 3 | Ctrl + Alt + 3 |
| Headline 4 | ⌘ ⌥ 4 | Ctrl + Alt + 4 |
| Headline 5 | ⌘ ⌥ 5 | Ctrl + Alt + 5 |

| Action | Mac | Windows & Linux |
|---|---|---|
| Headline 6 | ⌘ ⌥ 6 | Ctrl + Alt + 6 |
| Bold | ⌘ B | Ctrl + B |
| Italic | ⌘ I | Ctrl + I |
| Underline | ⌘ U | Ctrl + U |
| Strikethrough | ⌘ ⇧ X | Ctrl + Shift + X |
| Ordered list | ⌘ ⇧ 7 | Ctrl + Shift + 7 |
| Bullet list | ⌘ ⇧ 8 | Ctrl + Shift + 8 |
| Add or edit link | ⌘ K | Ctrl + K |
| Bring workflow annotation to front | ⌘ ⇧ ↑ | Ctrl + Shift + ↑ |
| Bring workflow annotation forwards | ⌘ ↑ | Ctrl + ↑ |
| Send workflow annotation to the back | ⌘ ↓ | Ctrl + ↓ |
| Send workflow annotation backwards | ⌘ ⇧ ↓ | Ctrl + Shift + ↓ |

## Quick nodes adding

*Table 8. The supported shortcuts related to add workflow documentation via node labels*

| Action | Mac | Windows & Linux |
|---|---|---|
| Open quick nodes panel and switch through ports | . | Ctrl + . |
| Select node | via ↑ ← ↓ → | via ↑ ← ↓ → |
| Enter selected node | | |

KNIME AG
Talacker 50
8001 Zurich, Switzerland
www.knime.com
info@knime.com