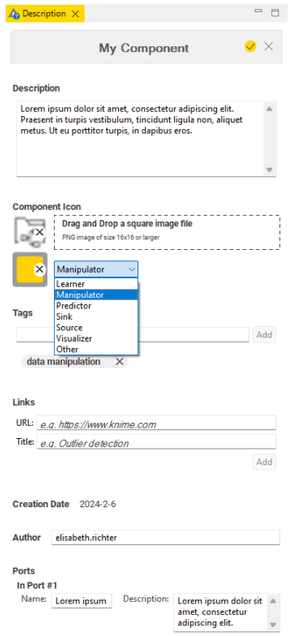
Cheat sheet: Components with KNIME Analytics Platform

Open for Innovation
KNIME

Components description



Documents the purpose and usage of the component and defines the appearance of the component. To access and edit the component description, open the Description panel from inside the component and select the empty canvas.

Description: Provides the text describing the use case, requirements, licensing and copyrights, disclaimers, and any other extra information you would like to add.

Icon: Drag and drop a PNG 16x16 px from your local file system. The icon will appear on the component, giving it a unique look.

Color: Usually the color of nodes and components is associated with a precise category. Pick the category that best fits your use case: The selected color appears behind the logo.

Tags: The keywords by which the search results in KNIME Community/Business Hub can be filtered.

Links: If the description of the component needs to provide links to external resources.

Author: The name of the component builder can be mentioned here.

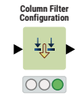
In/out ports: Describe the ports requirements here. For example: What kind of input column types are supported? Are there new rows/columns in the output? Does the input/output connect only to a precise other node or component?

Options: The component description also lists all the settings available inside the component dialog. To add text describing the settings usage open each configuration node dialog and enter it there node by node.

Configuration description

Single Selection Configuration: Creates a list of options of type String for a menu or radio buttons. Define options in the configuration dialog together with the selected default value. This node produces the value of the selected option, which can be used inside the component to configure other node settings.

Boolean Configuration: Creates a boolean selection for an enabled/disabled flag (1/0) in the form of a checkbox. This node produces the value of the selected option in a flow variable at its output port. Usually adopted to configure switch nodes and trigger different component modes.

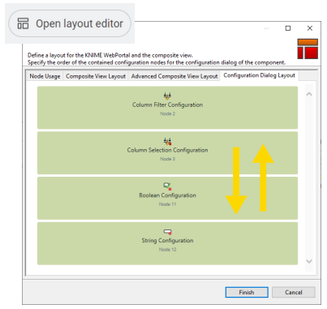


Column Filter Configuration: Selects the columns of the component input table. Set the node to include or exclude all columns by default. Useful to remove columns that the component should not use in its execution.

String Configuration: Creates a string flow variable for configuring other nodes inside the component. Pick a default value so that the user understands how this component setting works. The string flow variable is useful to determine how to rename the component output columns or for text displayed in the component's composite view.

Dialog layout panel

The component dialog stacks one setting for each configuration node in the component. To access it, right-click the component and select "Component > Open layout editor" or click the "Open layout editor" button in the workflow toolbar inside the component. In the panel select the Configuration Dialog Layout tab. Drag and drop tiles to change the component settings based on sequential steps the user should follow, from top to bottom.

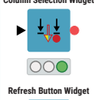


Widget nodes

Image View: Displays image output provided by a user. Useful to display images in Interactive Views or Data Apps.

Interactive Range Slider Filter Widget: Creates a slider to filter data to only include rows with values in the selected column within the specified range. The slider can interact with most of the other views inside the component when connected to its output.

Text View: Displays text output provided by a user. Useful to create text or number infographics in Interactive Views or Data Apps.



Column Selection Widget: Creates a list of selectable columns from the input data table in the form of a menu or radio buttons. The node produces the name of the selected column in a flow variable at its output port.

Refresh Button Widget: Offers a button to be placed in the composite view to trigger workflow execution. When clicked, the nodes after this widget re-execute and if any of them are widgets or views they are also updated.

View nodes

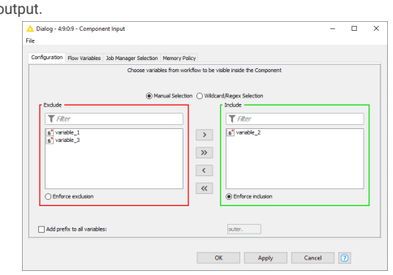
Heatmap: Visualizes tabular data using a color-coded matrix, revealing patterns and correlations.

Parallel Coordinates Plot: Displays one curve for every row and one parallel axis for each included column, both numerical and categorical. Useful to explore data points over several dimensions, looking for interesting patterns. It can automatically publish and subscribe to interactive events when it shares the same input with other views and/or interactive widgets in the component.

Line Plot (Plotly): Displays a curve for each selected column on the y axis. The x axis in between the curves is based on another column or the RowID. This view comes from the KNIME Plotly Integration, a JavaScript based open source visualization library.

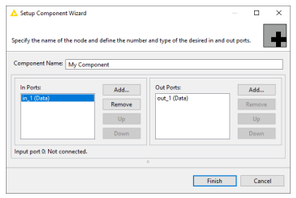
Flow variable filter

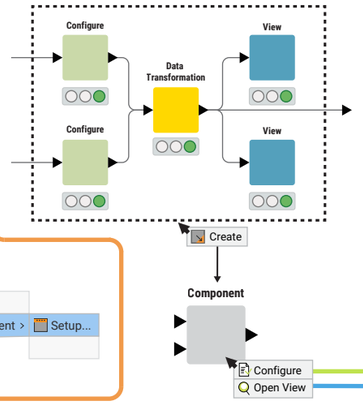
Filters the flow variables at the input and output of the component. Double-click the Input and Output ports inside the component to access a dialog. By default, flow variables pre-existing upstream of the component input cannot be used inside. Similarly, by default, flow variables generated within the component cannot be used downstream of the component output.

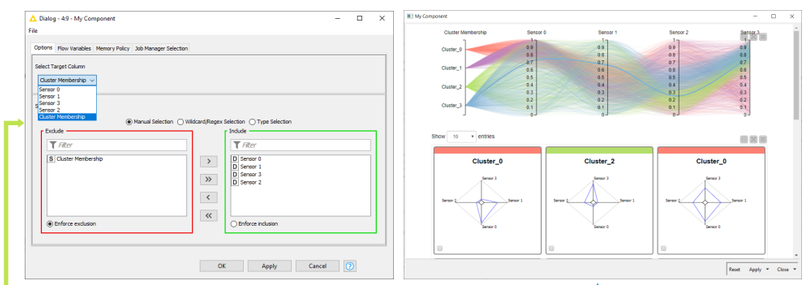


Component setup

You can still change component names and ports after component creation. The panel offers a text field to edit the title, buttons to change the order, remove or add. When adding a new port, a drop down menu appears to define the port type.







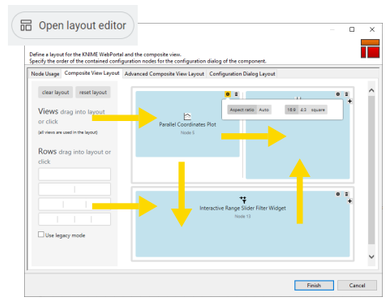
Create: To create a component multiple-select all the nodes you want to encapsulate in the component. Now right-click the selection and select "Create Component".

Configure: If you included a Configuration node in the component, open the configuration dialog. Change the component settings here before executing, just like any KNIME node.

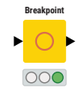
Open view: After executing a component that contains a Widget or View node, you can open its interactive view. In the opened composite view, interactivity across the single views and widgets nodes in the component is activated by default: visualizations sharing the same input table are usually connected via selection and filter events.

Composite view layout

By default KNIME stacks views and widgets from top to bottom in the composite view. Access the Composite View Layout tab via the Node Usage Layout panel from inside the component. Define the layout of the composite view with tables: reset/clear the current layout (top left buttons); drag rows from the left; add columns using "+"; drag the views/widgets on the left into the empty structure. Optionally set the ratio or pixel sizes of individual views/widgets via the gear button.



Error handling



Breakpoint: Fails on purpose when a certain condition is met. A custom error message appears on the node and on the outside of the component. Use it to detect whether the input or configurations of the component satisfy minimum requirements and provide the user with an intuitive message on what should be fixed after making the component fail on purpose.

Try (Data Ports): Starts a try-catch construct to enable an alternative path for the data flow in case of failure in the main branch. One branch is defined as the main branch while the other is set as the secondary branch. If execution fails in the main branch, the seconaary branch is activated. It must be closed by a Catch node.

Catch Errors (Data Ports): Closes a try-catch construct started with a Try node and collects the results from the active branch."

Verified Components

Verified Components behave like actual nodes and are regularly released on the KNIME Hub. Browse them all at knime.com/verified-component. Each new component is assigned to one of the 9 categories, represented by a color and symbol.

Automation Example - AutoML: Trains classification models and outputs the best one in an automated fashion. The component adopts Integrated Deployment to capture the end-to-end process as an optimized workflow.

Guided Analytics Example - Model Monitor View: Once a classifier model is trained and deployed, it can be monitored. The component generates a chart with performance over time and controls for re-training of the model.

Time Series Example - SARIMA Learner: Trains in Python a SARIMA (Seasonal ARIMA) model. This model is a powerful option when designing forecasts on data with seasonal or cyclic patterns.

Visualizations Example - Animated Bar Chart: Visualizes a bar chart changing over time via a Generic JavaScript View node. Each visual bar represents a different category, racing in range with the others in a smooth animation.

Finance Analytics Example - Measure Fractional Years: Computes the fraction of the year of the difference between two dates, similar to the YEARFRAC function in Microsoft Excel.

Text Processing Example - Web Text Scraper : Uses a Java based library to extract the main textual content from a web page to analyze online trends for Search Engine Optimization (SEO).

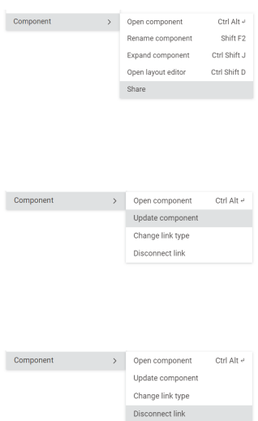
Data Manipulation Example - Python Transform: A series of transformations have been implemented via the KNIME Python Integration. Use it to learn how to reliably package your Python scripts for other KNIME users.

Model Interpretability Example - Global Feature Importance: Computes and visualizes global feature importance for an input model with four available model-agnostic XAI techniques.

Life Sciences Example - Molecular Properties Filter : Selects a subset of molecules based on molecular properties via RDKit nodes. The Interactive View depicts the properties and structural formula of the selected molecules.

Shared components

Components can be reused as your personal customized KNIME nodes or they can also be shared with others via KNIME Hub and KNIME Server.



To share a component choose the destination for the shared component in the window that opens. Now the component is shared and it can only be edited in its new location: A green arrow appears on top of the component to represent the link between the instance and the shared component in the Space Explorer. The link can be of relative type when the component is shared locally.

Manual update is useful to make sure your component is still up to date. A check for updates can be automated or prompted based on your KNIME Preferences. If an update is found but you opt out, the green arrow becomes dashed. If the link between the instance and the shared component is no longer valid the green arrow will become a red cross.

The owner of the shared component can edit its new location in the Space Explorer. After editing, KNIME can update the shared component instances in other workflows. If you are using someone else's shared component you can still edit it, but first you need to break the link: note that then you won't be able to get updates anymore.

Scripted components

Component builders can optionally implement component functionalities through coding. KNIME supports a number of coding frameworks as well as ways to ship dependencies with the component.

Generic JavaScript View: Offers a code editor for JavaScript to implement a custom view. Optionally feed in data to visualize it based on your implementation. The node offers checkboxes for a few dependencies (d3.js, ..) as well as a CSS editor.

R Snippet: Offers a code editor for R to process a KNIME table. KNIME executes the R installation configured either in the node settings and/or in KNIME Preferences. More nodes with different ports are available from the KNIME Interactive R Statistics Integration

Python Script: Offers a code editor for Python to process any number and type of inputs into outputs: three dots on the outside of the node means you can add ports. KNIME executes the Python installation configured either in the node settings and/or in KNIME Preferences.

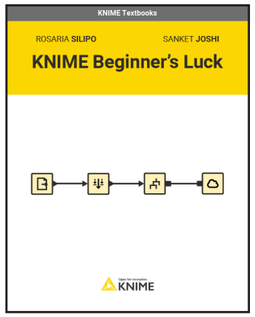
Conda Environment Propagation: Automatically installs the Conda environment necessary for your component to execute the downstream R/Python nodes. The environment usually includes the R/Python installation plus precise versions of the libraries. Useful to share scripted components with the necessary dependencies. Requires installation of Anaconda or Miniconda and minimal configuration in KNIME Preferences.

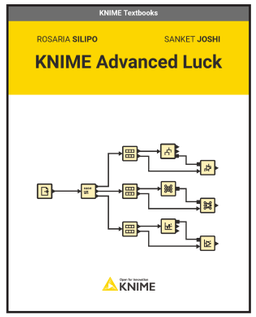
Resources

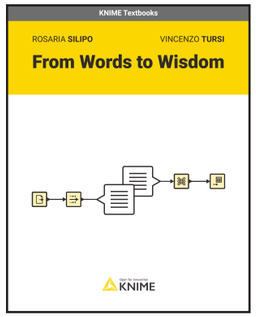
- **KNIME Press:** Access various data science books and other cheat sheets at ***knime.com/knimepress***, including beginner and advanced topics.
- **KNIME blog:** Engaging topics, challenges, industry news, & knowledge nuggets at ***knime.com/blog***.
- **Self-paced courses:** Take our free online self-paced courses to learn about data analysis, data engineering, or data science with KNIME (with hands-on exercises) at ***knime.com/learning***.
- **KNIME Community Hub:** Store, version, automate, and collaborate on private workflows, or explore and share public workflows with the KNIME Community at ***hub.knime.com.***
- **KNIME Forum:** Join our global community & engage in conversations at ***forum.knime.com***.
- **KNIME Business Hub:** For team-based collaboration, automation, management, & deployment check out KNIME Business Hub at ***knime.com/knime-business-hub***.

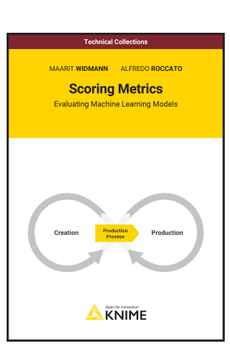
KNIME Press

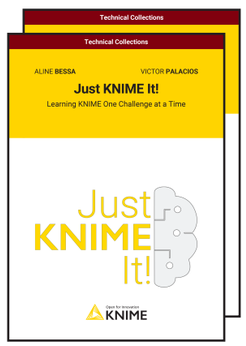
Extend your KNIME knowledge with our collection of books from KNIME Press. For beginner and advanced users, through to those interested in specialty topics such as topic detection, data blending, and classic solutions to common use cases using KNIME Analytics Platform - there's something for everyone. Available for download at www.knime.com/knimepress.

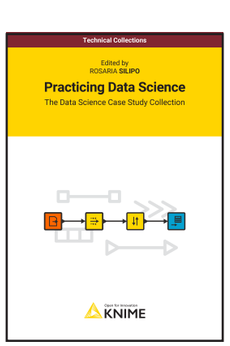


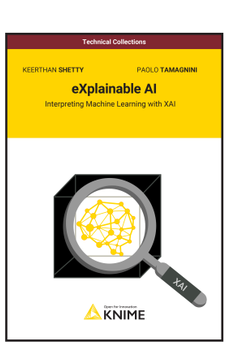


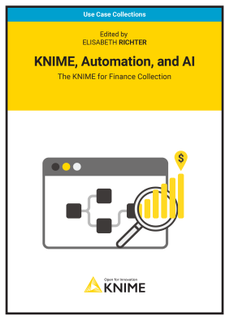


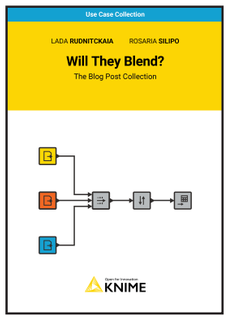


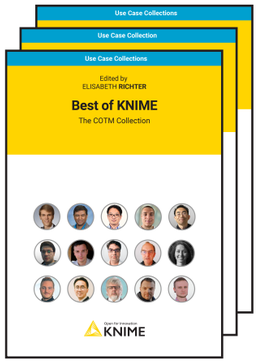


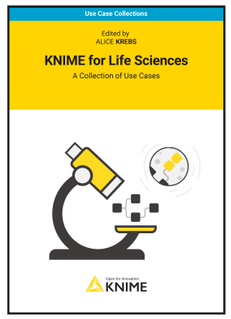


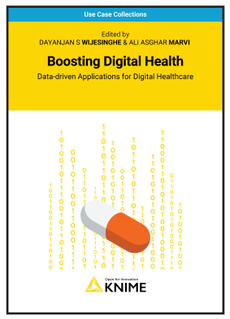


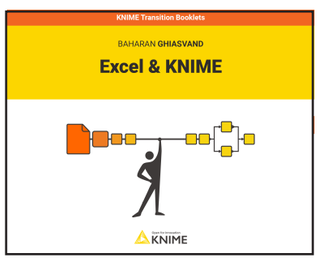


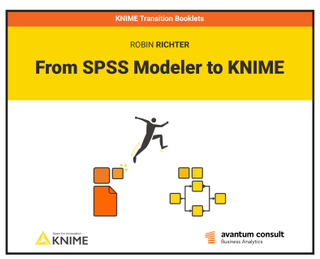


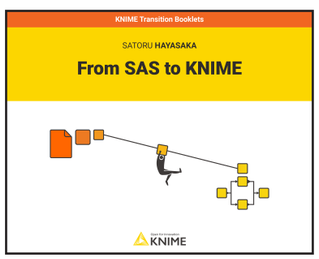


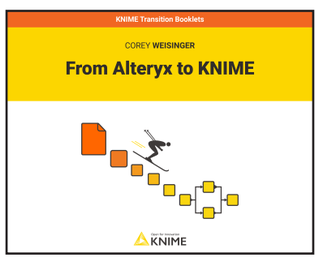


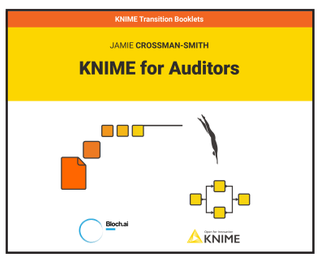












Need help?
Contact us!