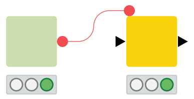
Cheat sheet: Control and orchestration with KNIME Analytics Platform

Open for Innovation KNIME

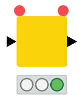
Control

Flow variables

Flow Variables allow for the parameterization of a workflow. A Flow Variable is a parameter that can assume different values at different execution points in the workflow & overwrite configuration settings in upcoming nodes.



Hidden Flow Variable Ports
Each node has two hidden Flow Variable ports to accept incoming Flow Variables & to propagate them to the upcoming nodes. To make these ports visible, hover your cursor over the node. To configure a node's flow variables right-click the node and select **Configure flow variables**.



Creating a Flow Variable
1. Use a Configuration or a Widget node to create a Flow Variable at any point in your workflow.
2. Use any of the nodes converting data into Flow Variables.
3. Via the node configuration window in the ***Flow Variables*** tab, fill in a blank box with the name of the Flow Variable.

Math Formula (Variable)

Math Formula (Variable): This node is the Flow Variables version of the String Manipulation node. Similarly, other nodes have their own version for Flow Variables like the Rule Engine Variable node & the Math Formula (Variable) node.

Merge Variables

Merge Variables: Combines Flow Variables from two or more separate branches. To add a branch click the plus in the bottom left corner. If Flow Variables with the same name are collected, the Flow Variable in the top most connection is retained.

Inject Variables (Data)

Inject Variables (Data): Adds (injects) the Flow Variables at its Flow Variable input port into the data table at the top input port. The input data table is forwarded (unaltered) in the node output port.

Extract Variables (Data)

Extract Variables (Data): Extracts the Flow Variables coming in through the input data port & produces them as standalone Flow Variables at the output port.

Widgets & configuration nodes

Widget and configuration nodes create one or more new flow variables & make them available at the output port. Widget nodes create a UI item for the composite view or the composite view/data app to create and control the flow variable. Configuration nodes create a UI item in the configuration dialog of a component & are not visible in the composite view or the composite view/data app.

Column Selection Widget

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.

Interactive Range Slider Filter Widget

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 views from other JavaScript based nodes in the same composite view.

Text View

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

Credentials Widget

Credential Widget: Creates fields to enter credentials (username & password) in the form of text boxes. The text in the password box is masked. The node produces these credentials in a flow variable at its output port.

Single Selection Configuration

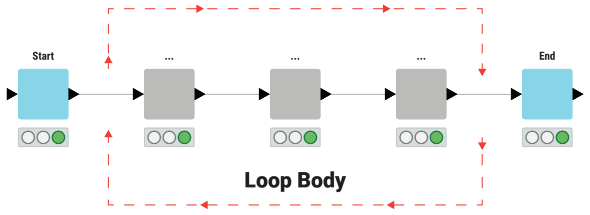
Single Selection Configuration: Creates a list of options of type String in the form of a menu or radio buttons. These options can be defined in the configuration dialog together with the selected default value. The node produces the value of the selected option in a flow variable at its output port.

Booelan Configuration

Boolean Configuration: Creates a boolean selection for an enabled/disabled flag (1/0) in the form of a checkbox. The node produces the value of the selected option in a flow variable at its output port (1 if the checkbox is enabled, 0 if disabled).

Loops

A loop is a sequence of operations that is repeated until a condition is met. It has a start, an end, & a loop body of operations. A loop is implemented via a Loop Start node, a Loop End node, & a number of nodes in between for the body of operations. Different Loop Start nodes provide alternative ways to iterate on the input data. Different Loop End nodes provide alternative ways to collect results. The end condition can be defined in either the Loop Start node or the Loop End node, depending on the kind of loop. Some nodes to start & end a loop work on data & others on Flow Variables. These nodes can be paired up freely - loops can start with data & end up with Flow Variables & vice versa.



Generic Loop Start

Generic Loop Start: Starts a loop. It should be paired with a loop end node defining the end condition.

Group Loop Start

Group Loop Start: Starts a loop iterating over groups of input data rows i.e. each iteration works on a different group of the input data. Groups are extracted from values in selected columns, as in the configuration window of the GroupBy node.

Column List Loop Start

Column List Loop Start: Starts a loop iterating over a selected list of columns. At each iteration, the current column & the remaining columns are passed into the loop body.

Table Row to Variable Loop Start

Table Row to Variable Loop Start: Starts a loop iterating over input data rows one at a time i.e. each iteration is dedicated to just one row. At each iteration, the row values are converted into Flow Variables & named after the column headers.

Loop End

Loop End: Ends a loop by concatenating the resulting rows from each iteration.

Loop End (Column Append)

Loop End (Column Append): Ends a loop by joining together the resulting columns from each iteration on the column containing the row IDs.

Variable Loop End

Variable Loop End: Loop End nodes do not only work on data. At the end of a loop you might want to pass the results as a Flow Variable - like in the Variable Loop End node.

Variable Condition Loop End

Variable Condition Loop End: Ends a loop when a condition is met, i.e. a specific value in a Flow Variable.

Parallel Chunk Start

Parallel Chunk Start: Starts a parallel loop iterating over input data rows in chunks of equal size which get processed in parallel in the loop body. The number of parallel iterations, (chunks), is defined either automatically based on the size of the input data, or manually.

Parallel Chunk End

Parallel Chunk End: Ends a parallel loop by concatenating the resulting data rows from each chunk.

Recursive Loop Start

Recursive Loop Start: Starts a recursive loop iterating on the updated input table. The input data table for each iteration is the output data table from the previous iteration. The first iteration works on the data table provided at the input port of the node. This is the only loop where the updated data table feeds the next iteration.

Recursive Loop End

Recursive Loop End: Ends a recursive loop. The top input port collects the resulting data rows. The lower input port collects the updated data table to be passed back to the Recursive Loop Start node. Also defines the loop end condition, such as max. number of iterations, min. number of rows, & a specific value for a Flow Variable.

Feature Selection Loop Start (1:1)

Feature Selection Loop Start (1:1): Starts a feature selection loop iterating on a set of input columns (features) to extract the subset that optimizes a given Flow Variable. Used for feature selection against a given model & against a given performance metric. Together with the Feature Selection Loop End node, it's used in the Backward/Forward Feature Selection metanode.

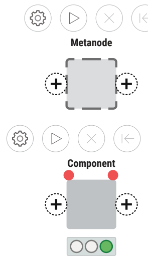
Feature Selection Loop End

Feature Selection Loop End: Ends a feature selection loop. It sets the variable with the metric to be maximized/minimized for best feature selection. The top output port of the node produces the evaluation metrics for all column subsets. The bottom output port produces a summary of the different feature sets with the associated scores.

Metanodes & components

A Metanode or Component is a node that contains other nodes.

Creating a Metanode or Component
Select all relevant nodes, right-click and select **Create metanode** for a metanode or **Create component** for a component. Right-clicking a metanode or component opens the context menu with a number of options such as expand or configure. To add input or output ports to a metanode or component click the plus on the left side for additional input ports, and the plus on the right side for additional output ports.

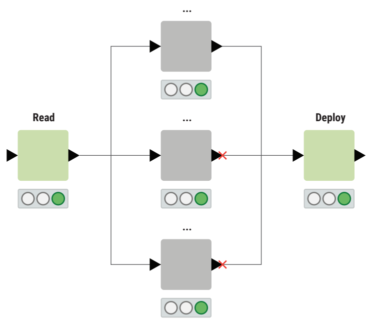


Metanodes just collect nodes inside and are an efficient way to clean up your workflow.

Components encapsulate & abstract functionality, can have their own dialog and can have their own sophisticated, interactive views. They can be reused in your own workflows but also shared with others: via KNIME Business Hub or KNIME Community Hub. They can also represent web pages in a Data App deployed to others via KNIME Business Hub. Flow Variables cannot enter or exist a component, unless explicitly configured in the component's input and output nodes.

Switches

A switch construct allows you to conditionally execute different sequences of operations via nodes located on different workflow branches. All start with a Switch Start node and optionally end with a Switch End node. In between, a number of parallel branches implement various operation sequences.



A switch construct works on data, flow variables, database queries, or any other port type defined via the dynamic ports of the start and end nodes. You can pair up different port types.

CASE Switch Start

CASE Switch Start: Selectively activates only one of its three output ports, enabling three alternative paths for the input. The active output port can be configured manually or automatically via the value of a Flow Variable. The IF Switch node performs the same task but with only two alternative output ports (both can be active at the same time).

CASE Switch End

CASE Switch End: Collects the results from the active one among the branches connected to its input ports. The End IF node works similarly & is paired with the IF Switch node.

Catch Errors (Data Ports)

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

Try (Data Ports)

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 secondary branch is activated. It must be closed by a Catch node.

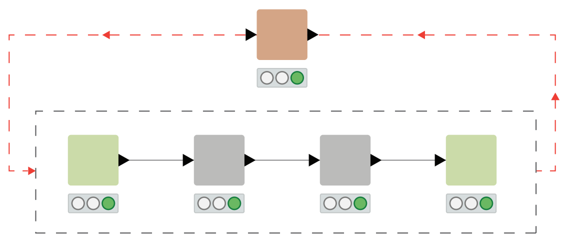
Empty Table Switch

Empty Table Switch: Provides an alternative path for the data flow in case the main branch has no data rows. It activates the top output port & deactivates the bottom output port if the input table has at least one data row. It deactivates the top output port & activates the bottom output port if the input table is empty.

Active Branch Inverter

Active Branch Inverter: Changes the activity status of the branch. If the input port is active, the output port becomes inactive & vice versa. It's often used to force a branch to produce an output even if it's inactive & vice versa (to deactivate a branch even if it's active).

Orchestration



Call Workflow Service

Call Workflow Service: Triggers the execution of a workflow stored in the LOCAL workspace or on KNIME Business Hub. Data exchange with the triggered workflow can happen via any port type via the Workflow Service Input/Output nodes.

Workflow Service Input

Workflow Service Input: Receives data via any port type from the caller workflow via the Call Workflow Service node. Enables an efficient data exchange between workflows only, excluding third-party software. The corresponding "Workflow Service Output" node returns the results to the caller workflow.

Call Workflow (Table Based)

Call Workflow (Table Based): Triggers the execution of a workflow stored in the LOCAL workspace or on KNIME Business Hub. Data exchange with the triggered workflow can happen via data tables, flow variables, or credentials via the Container Input/Output (Table), (Variable), or (Credentials) nodes.

Container Input (Table)

Container Input (Table): Receives a data table from the caller workflow or from a third-party software. If no input is provided, the template default data table is used. Similar nodes are available to exchange flow variables & credentials. The corresponding "Container Output (Table)" node returns the results as a data table.

Call Workflow (Row Based)

Call Workflow (Row Based): Triggers the execution of a workflow stored in the LOCAL workspace or on KNIME Business Hub. Data exchange with the triggered workflow happens via JSON format via the Container Input/Output (Row) or (JSON) nodes.

Container Input (JSON)

Container Input (JSON): Receives a JSON data structure from the caller workflow or from a third-party software. If no input is provided, the template default JSON structure is used. The corresponding "Container Output (JSON)" node returns the results as a JSON structure.

GET Request

GET Request: Calls a REST service in GET mode. The node can send one single service request set in the configuration window, or multiple service requests stored in a column of the input table. Responses are saved in the output data table. Options to set authentication, request header, & response header are available.

KNIME Server Connector

KNIME Server Connector: Connects to a KNIME Server using the server URL & credentials. After the connection has been created, new directories on the server can be created & remote files can be accessed, created, & deleted.

Timer Info

Timer Info: Reports the number of executions & execution times for each node in a workflow. Both single node & total workflow execution time are reported. Execution times for nodes inside metanodes can also be reported.

Send Email

Send Email: Sends HTML or text formatted emails using an external SMTP to a recipient - including the message & possible attachments.

Save Workflow

Save Workflow: Saves the (also partially - up to here) executed workflow.

Create Directory

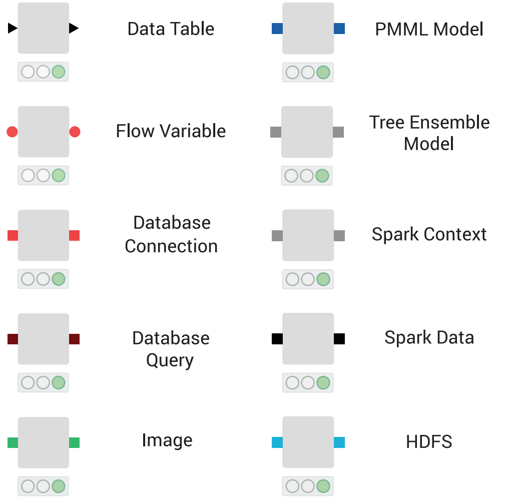
Create Directory: Creates a new folder and outputs the folder location as a flow variable of type Path.

Create File/Folder Variables

Create File/Folder Variables: Creates a list of Path type Flow Variables pointing to files/folders relative to a selected base location.

Node ports

Different types of data pass through different node ports. Only ports of the same type can be connected. Here are some examples of ports for frequently used data types.

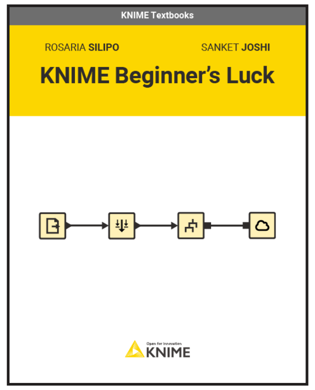


Resources

- **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:** Browse and share workflows, nodes, and components or access collection pages for dedicated topics 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***.

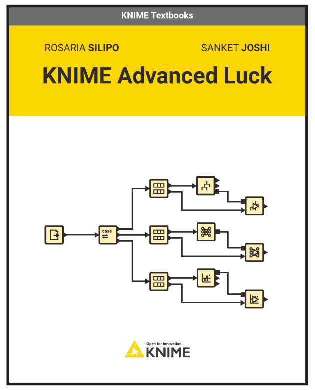
KNIME Press

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.


KNIME Textbooks

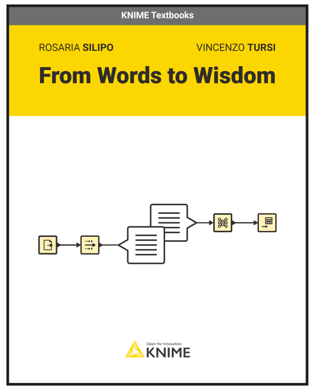
ROSARIA SILIPO SANKET JOSHI

KNIME Beginner's Luck


KNIME Textbooks

ROSARIA SILIPO SANKET JOSHI

KNIME Advanced Luck


KNIME Textbooks

ROSARIA SILIPO VINCENZO TURSI

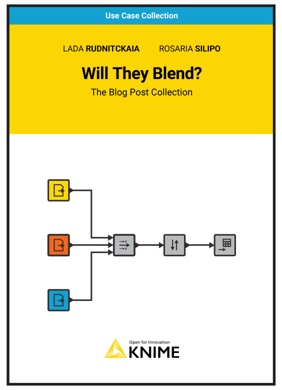
From Words to Wisdom

Use Case Collections

Edited by
ROBERTO CADILI & FRANCISCO VILLARROEL ORDENES

Meet Your Customers

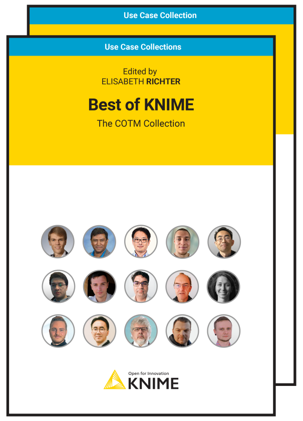
The Marketing Analytics Collection


Use Case Collection

LADA RUDNITCKAIA ROSARIA SILIPO

Will They Blend?

The Blog Post Collection

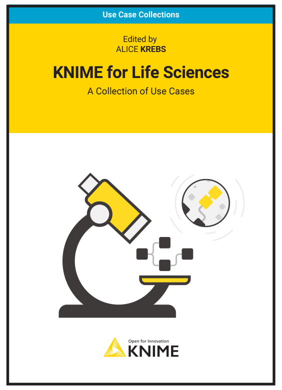

Use Case Collection

Use Case Collections

Edited by
ELISABETH RICHTER

Best of KNIME

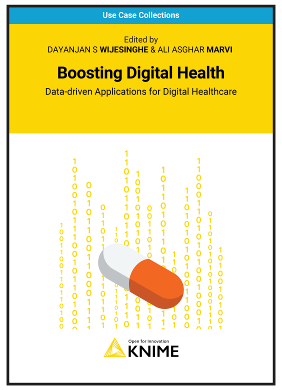
The COTM Collection


Use Case Collections

Edited by
ALICE KREBS

KNIME for Life Sciences

A Collection of Use Cases


Use Case Collections

Edited by
DAYANJAN S WIJESINGHE & ALI ASGHAR MARVI

Boosting Digital Health

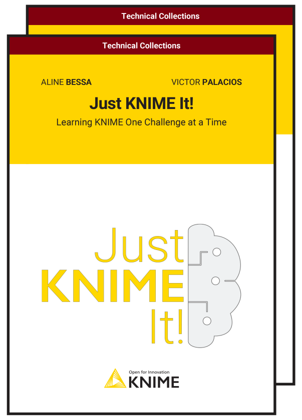
Data-driven Applications for Digital Healthcare

Technical Collections

MAARIT WIDMANN ALFREDO ROCCATO

Scoring Metrics

Evaluating Machine Learning Models


Technical Collections

Technical Collections

ALINE BESSA VICTOR PALACIOS

Just KNIME It!

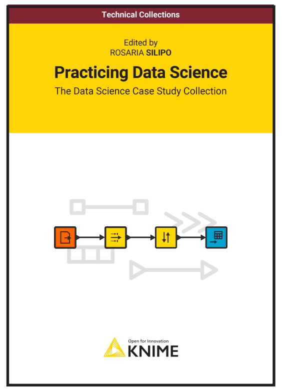
Learning KNIME One Challenge at a Time

Technical Collections

PAOLO TAMAGNINI

Guided Labeling

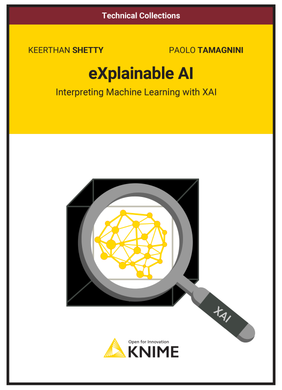
Generating Labels with
Active Learning & Weak Supervision


Technical Collections

Edited by
ROSARIA SILIPO

Practicing Data Science

The Data Science Case Study Collection


Technical Collections

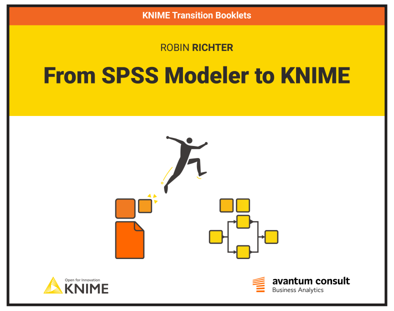
KEERTHAN SHETTY PAOLO TAMAGNINI

eXplainable AI

Interpreting Machine Learning with XAI

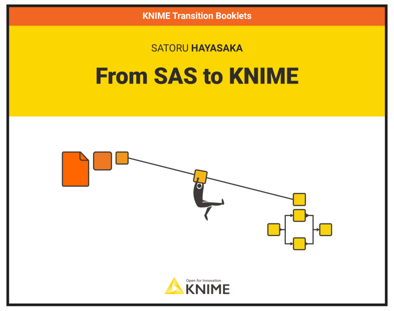
from
Excel®
to
KNIME®

Author: Kathrin Melcher


KNIME Transition Booklets

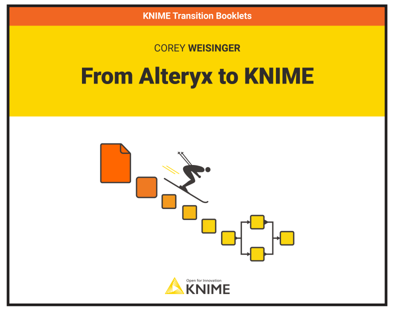
ROBIN RICHTER

From SPSS Modeler to KNIME


KNIME Transition Booklets

SATORU HAYASAKA

From SAS to KNIME


KNIME Transition Booklets

COREY WEISINGER

From Alteryx to KNIME

Need help?
Contact us!