KNIME Training @ Syngenta

Monday, September 11, 2023

Dr. Alice Krebs

Content

Topic	Details
(Chemical) Data Processing	Manipulating Data – Cleaning, Data types, Conditions/Rule Engine, Regex Aggregating Data – GroupBy, Pivoting
Visualization & Output	Visualizing Data Creating Interactive & Composite Views, Components Output & File download, presentations, Smartsheet, Excel
Loops & Flow Variables, Workflow Control	What is a Flow Variable? - Creating and using Flow Variables, Path Flow Variables Repeating Workflow Parts – Loops Switches, Try-Catch, Error handling
KNIME Python Integration	Executing Python code in KNIME Conda Environment Propagation Integrating Jupyter notebooks, Code adaptions to match KNIME Python API
Customized KNIME Nodes in Python	Basics on the example template node
Chemistry Use Cases	Multiparameter Optimization (MPO), Sketcher, Similarity Search, Fingerprints, Clustering

(Chemical) Data Processing

Selected Commercial Extensions for Cheminformatics

- BioSolveIT Nodes
  - CoLibri (Chemistry Spaces)
  - IO
  - Assess Affinity with Hyde in SeeSAR
  - Compute FTrees Similarity
  - Compute FlexS Alignments
  - Compute LeadIT Docking
  - Convert Molecules with Naomi
  - FTrees Query Generator
  - Filter Molecules with Naomi
  - FlexX Docking
  - Generate 3D Coordinates
  - Generate Protomers / Tautomers with Naomi
  - Interactive BioSolveIT Table
  - Interactive SeeSAR Viewer
  - Prepare Receptor with LeadIT
  - Run ReCore Interactively
  - Search FTrees Fragment Space
  - SeeSAR Project Generator

- ChemAxon / Infocom
  - JChem
    - IO
    - Converter
    - Marvin
    - Calculator Plugins
    - JChem Base
    - JChem Cartridge
    - Standardizer
    - Structure Checker
    - Name to Structure
    - Screen
    - JKlustor
    - Reactor
    - Markush Viewer
    - Metabolizer
    - Fragmenter
  - Marvin

- Cresset
  - Forge
    - Models
    - Project
    - Forge Align
    - Activity Miner
    - FieldTemplater
  - Spark
    - Spark Fragment Selector
    - Generate Spark Database
    - Spark Database Search
  - XedTools
    - XedMin
    - XedeX
  - Torch/Forge Molecule Viewer

- MOE
  - Input
  - Output
  - Convert
  - Transform
  - Process
  - Calculate
  - QuaSAR
  - Fingerprints
  - Simulations
  - Bioinformatics
  - Fragment Based Design
  - CombiChem
  - Miscellaneous
  - Pharmacophore
  - Materials

- Schrödinger
  - Readers/Writers
  - Converters
  - Ligand Preparation
  - Property Generation
  - Cheminformatics
  - Pharmacophore Modeling
  - Protein Structure Prediction
  - Docking and Scoring
  - Molecular Mechanics
  - Molecular Dynamics
  - Quantum Mechanics
  - Workflows
  - Filtering
  - Reporting
  - Scripting
  - Tools

Selected Open Source Extensions for Cheminformatics

- RDKit
  - Converters
  - Modifiers
  - Calculators
    - RDKit Descriptor Calculation
    - RDKit Calculate Charges
  - Geometry
    - RDKit Generate Coords
    - RDKit Optimize Geometry
    - RDKit Add Conformers
    - RDKit Open 3D Alignment
    - RDKit RMSD Filter
  - Fingerprints
    - RDKit Fingerprint
    - RDKit Count-Based Fingerprint
    - RDKit Fingerprint Reader
    - RDKit Fingerprint Writer
    - RDKit Diversity Picker
  - Fragments
  - Searching
  - Reactions
  - Viewing
    - RDKit Interactive Table
    - RDKit SMILES Headers
    - RDKit Molecule Highlighting
  - Experimental

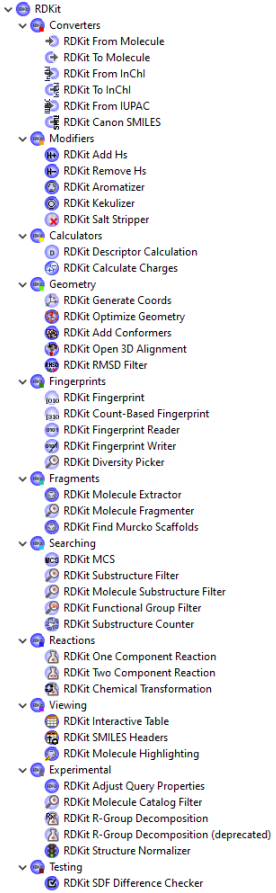
- CDK
  - 3D
    - 3D Coordinates
    - 3D D-Moments
    - 3D D-Similarity
    - 3D RMSD
    - 3D Viewer
    - 3D WHIM
  - AMBIT
  - I/O
  - 2D Coordinates
  - Atom Signatures
  - ChemSpider
  - Connectivity
  - Depiction
  - Element Filter
  - Fingerprint Similarity
  - Fingerprints
  - Hydrogen Manipulator
  - Lipinski's Rule-of-Five
  - Mass Calculator
  - Molecular Properties
  - OPSIN
  - SMARTS Query
  - Structure Sketcher
  - Substructure Search
  - Sugar Remover
  - Sum Formula
  - Symmetry
  - XLogP

- Erlwood Nodes
  - IO
  - Structure Data Format Converters
  - Structure Similarity
    - Fingerprint Similarity
  - Structure Properties
    - Plane of Best Fit Calculator
  - Virtual Screening
    - Virtual Screening Metrics
  - Evaluation and Ranking
    - Desirability Ranking
    - Pareto Ranking
  - SAR Analysis
    - Automated Matched Pairs
    - Free-Wilson Matched Pairs
  - Viewers
    - 2D/3D Scatterplot
    - Activity Cliffs Viewer
    - Similarity Viewer
  - Testing

- Vernalis
  - Databases
  - European PubMed Central
    - European PubMed Central Advanced Search
  - Fingerprints
  - Flow Control
  - IO
  - Matched Molecular Pairs (MMPs)
    - Filtering
    - Fragmentation
    - Pair Generation
    - Rendering
    - Transforms
    - MMP Calculate Maximum Cuts (RDKit)
    - MMP Fragmentation Type Loop Start
    - Uniquify IDs
  - Principal Moments of Inertia (PMIs)
  - RCSB PDB Tools
    - PDB Connector
    - PDB Connector (XML Query)
    - PDB Connector Combine XML Queries
    - PDB Connector Custom Report
    - PDB Connector Query Only
    - PDB Connector Query Only (XML Query)
    - PDB Connector XML Query Builder
    - PDB Describe Heterogens
    - PDB Downloader (Source)
    - PDB SMILES Query
    - PDB Downloader
  - Local PDB Tools
  - Sequence Tools
  - Speedy SMILES
  - Testing
  - Miscellaneous

What is RDKit?

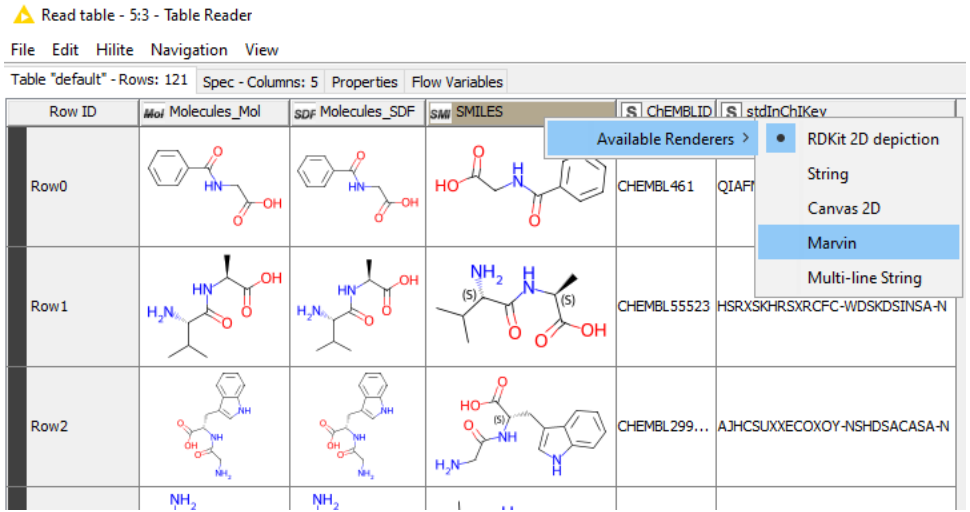
- Open source cheminfo library in C++
- Wrappers for KNIME maintained by the open source community
- Useful for:
  - Descriptor calculation
  - Cleaning structures
  - InChI conversion
  - Canonical SMILES
  - Fingerprints
  - Scaffolds/substructures
  - Reaction simulation
  - and more…



http://www.rdkit.org

Overview of Types in KNIME

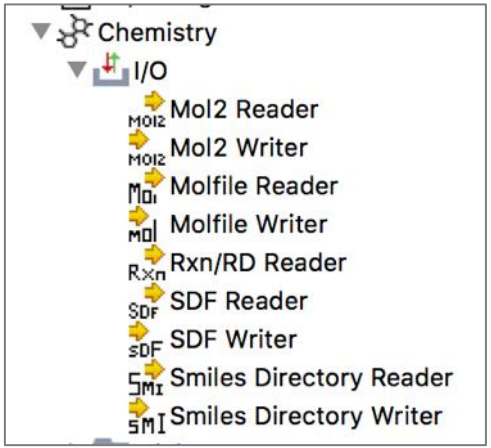
- Basic KNIME data types
  - String, integer, double
- KNIME core chemistry data types:
  - SMILES, SMARTS, SDF, Mol, Mol2
  - Structures in these formats can be rendered in KNIME tables
- Change the rendering by clicking on the column header
  - Available renderers depend on installed extensions

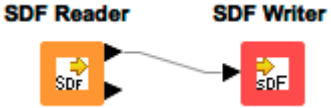


Nodes for Reading and Writing files

Reader and writers provided for:

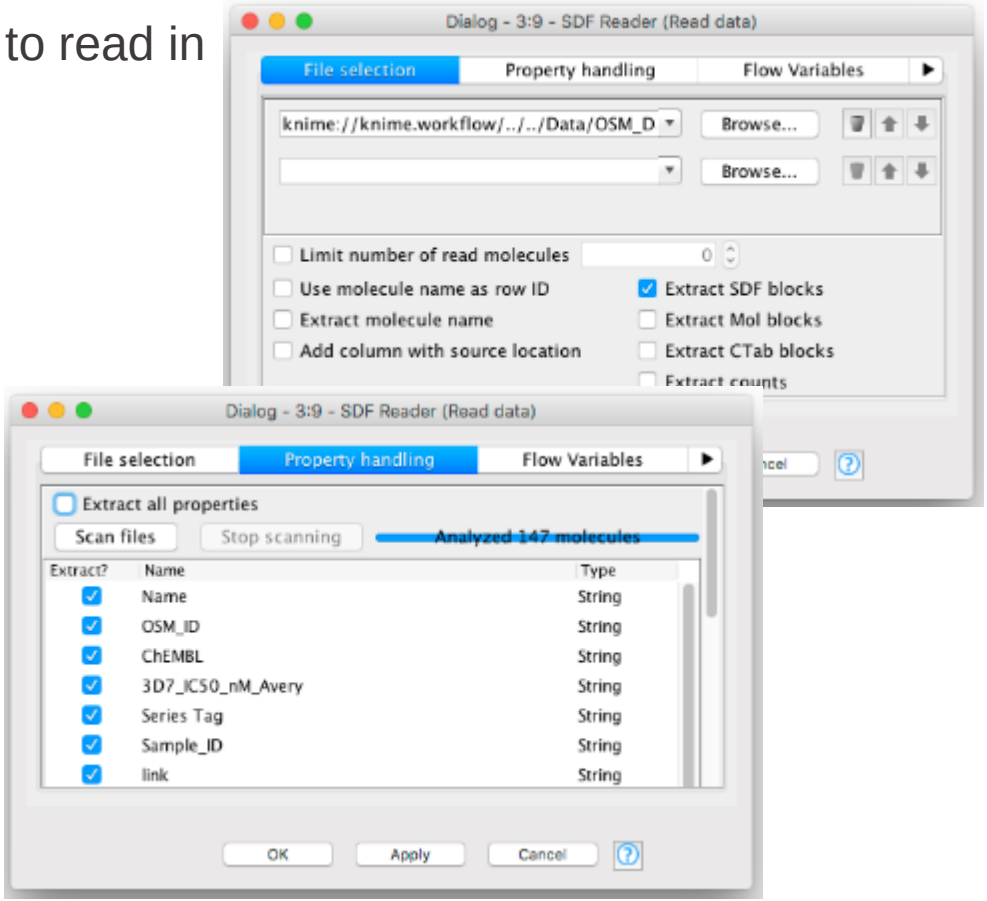
- SDF, SMILES, Mol, Mol2





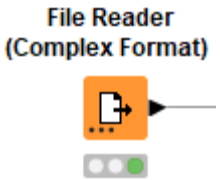
Reading SD files

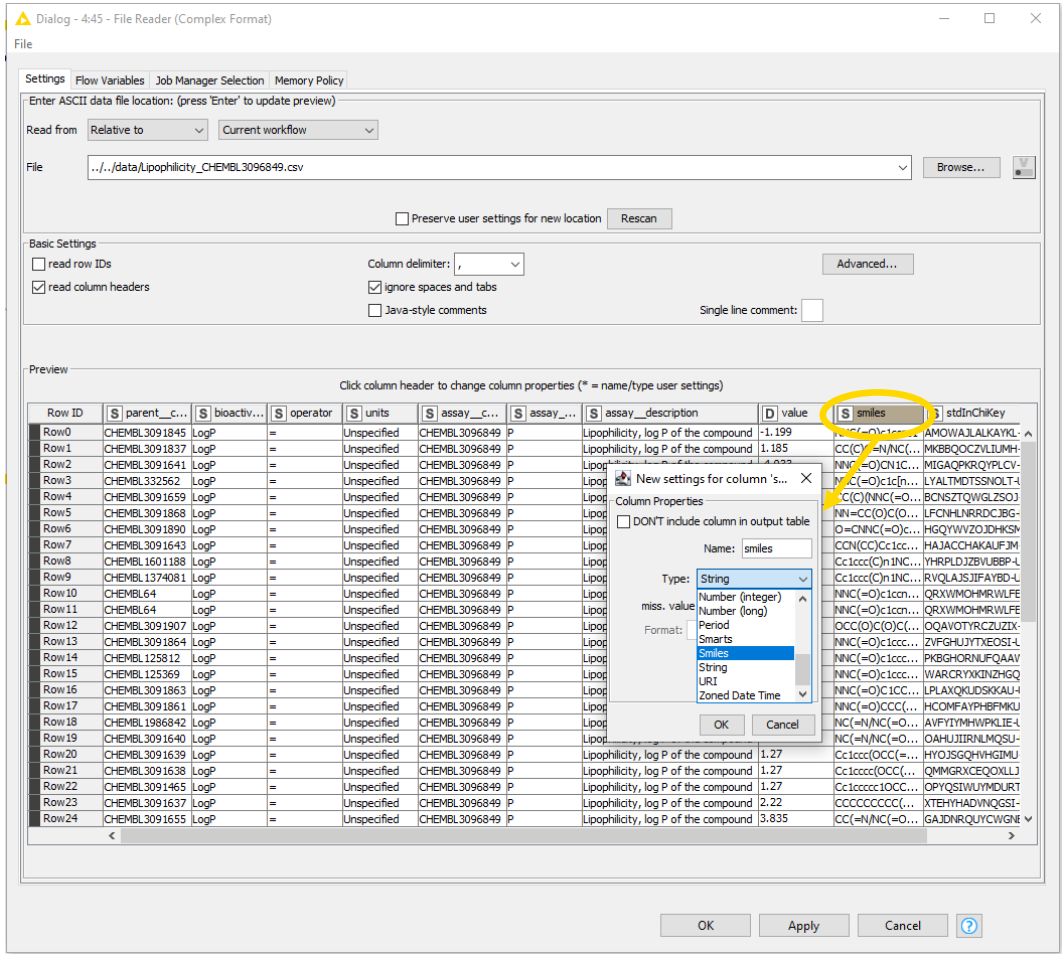
- Configure the property handling tab to read in columns with “other” data



File Reader (Complex Format)

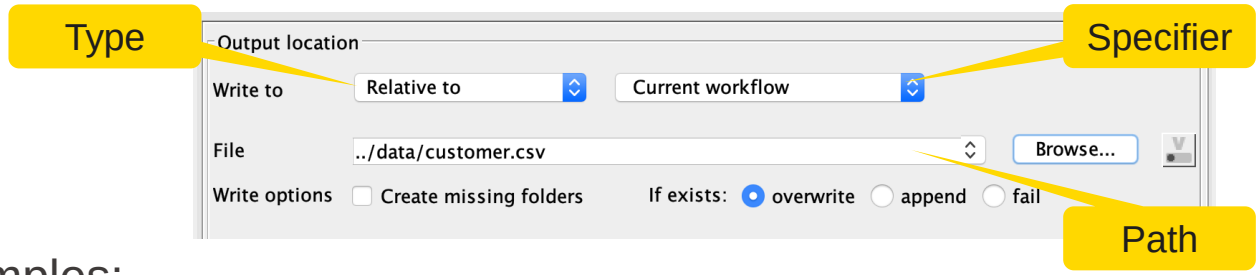
- Allows to directly define data type upon import
- Click on column header to set type





Common Settings: File Path

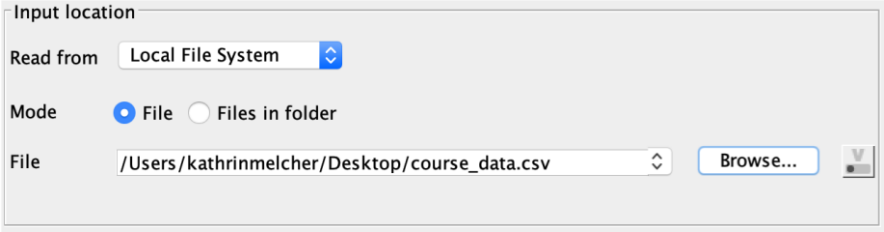
- A path consists of three parts:
  - **Type**: Specifies the file system type e.g. local, relative, mountpoint, custome_url or connected.
  - **Specifier**: Optional string with additional file system specific information e.g. relative to which location (knime.workflow)
  - **Path**: Specifies the location within the file system



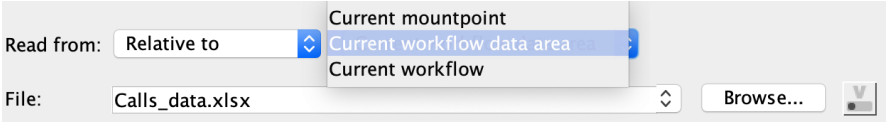
- Examples:
  - (LOCAL, , C:\Users\username\Desktop)
  - (RELATIVE, knime.workflow, file1.csv)
  - (MOUNTPOINT, MOUNTPOINT_NAME, /path/to/file1.csv)
  - (CONNECTED, amazon-s3:eu-west-1, /mybucket/file1.csv)

Common Settings: 4 Default File Systems

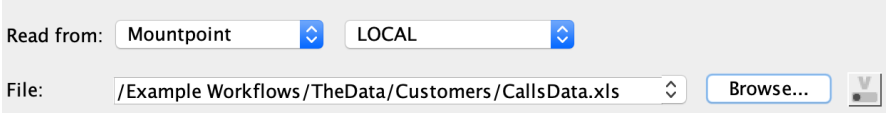
- ## Local File System



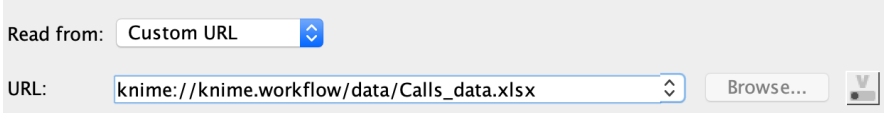
- ## Relative to …



- ## Mountpoint

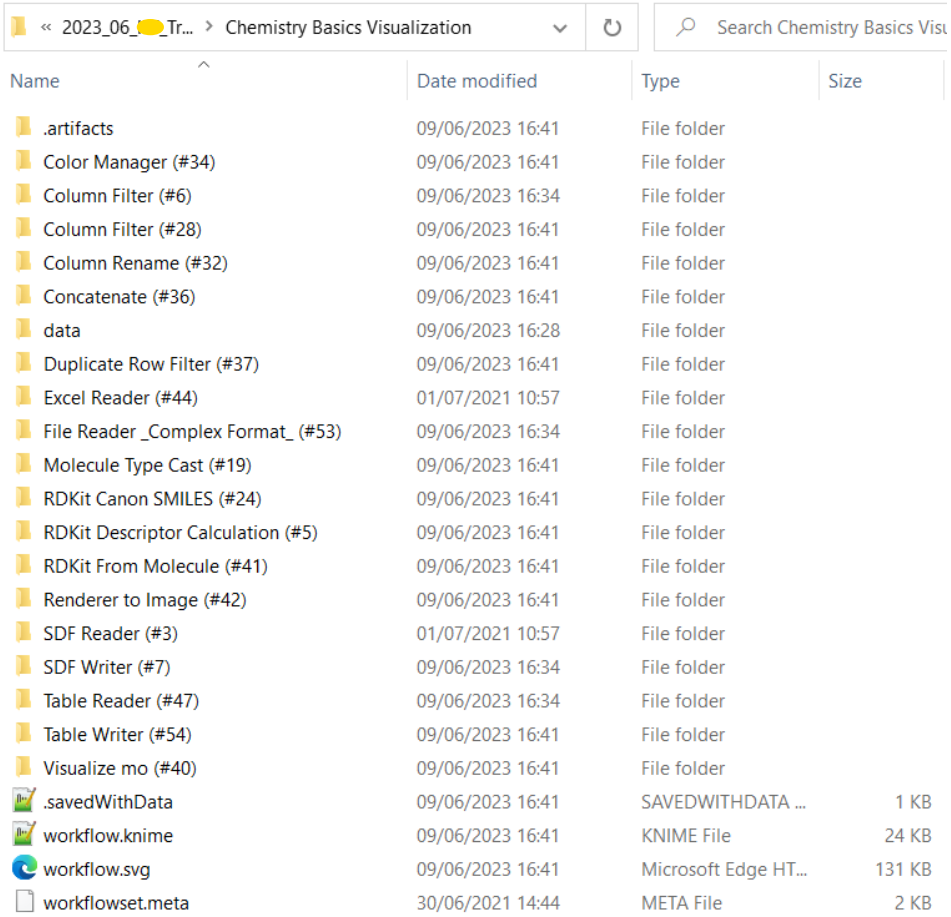


- ## Custom URL



Workflow Data Area

- Workflows are folders in the workspace folder
- Data area is a folder inside the workflow
- Easy way to share and store data with a workflow



« 2023_06_[redacted]_Tr... › Chemistry Basics Visualization

Search Chemistry Basics Visu

Name	Date modified	Type	Size
.artifacts	09/06/2023 16:41	File folder	
Color Manager (#34)	09/06/2023 16:41	File folder	
Column Filter (#6)	09/06/2023 16:34	File folder	
Column Filter (#28)	09/06/2023 16:41	File folder	
Column Rename (#32)	09/06/2023 16:41	File folder	
Concatenate (#36)	09/06/2023 16:41	File folder	
data	09/06/2023 16:28	File folder	
Duplicate Row Filter (#37)	09/06/2023 16:41	File folder	
Excel Reader (#44)	01/07/2021 10:57	File folder	
File Reader _Complex Format_ (#53)	09/06/2023 16:34	File folder	
Molecule Type Cast (#19)	09/06/2023 16:41	File folder	
RDKit Canon SMILES (#24)	09/06/2023 16:41	File folder	
RDKit Descriptor Calculation (#5)	09/06/2023 16:41	File folder	
RDKit From Molecule (#41)	09/06/2023 16:41	File folder	
Renderer to Image (#42)	09/06/2023 16:41	File folder	
SDF Reader (#3)	01/07/2021 10:57	File folder	
SDF Writer (#7)	09/06/2023 16:34	File folder	
Table Reader (#47)	09/06/2023 16:34	File folder	
Table Writer (#54)	09/06/2023 16:41	File folder	
Visualize mo (#40)	09/06/2023 16:41	File folder	
.savedWithData	09/06/2023 16:41	SAVEDWITHDATA ...	1 KB
workflow.knime	09/06/2023 16:41	KNIME File	24 KB
workflow.svg	09/06/2023 16:41	Microsoft Edge HT...	131 KB
workflowset.meta	30/06/2021 14:44	META File	2 KB

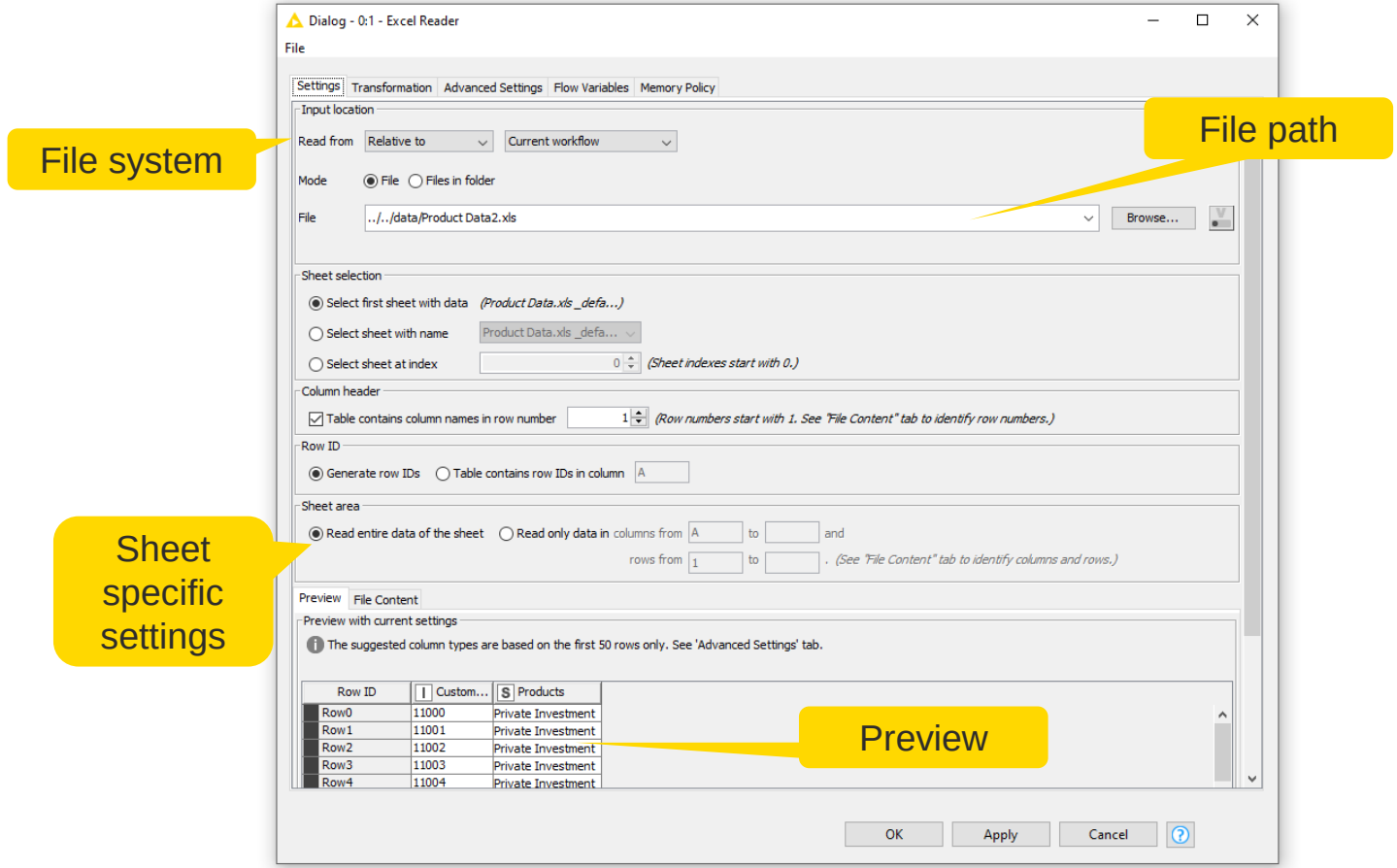
Open for Innovation KNIME

Excel Reader (XLS)

- Reads .xls and .xlsx file from Microsoft Excel
- Supports reading from multiple sheets

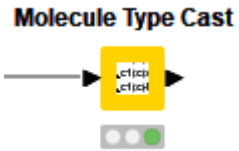
Excel Reader

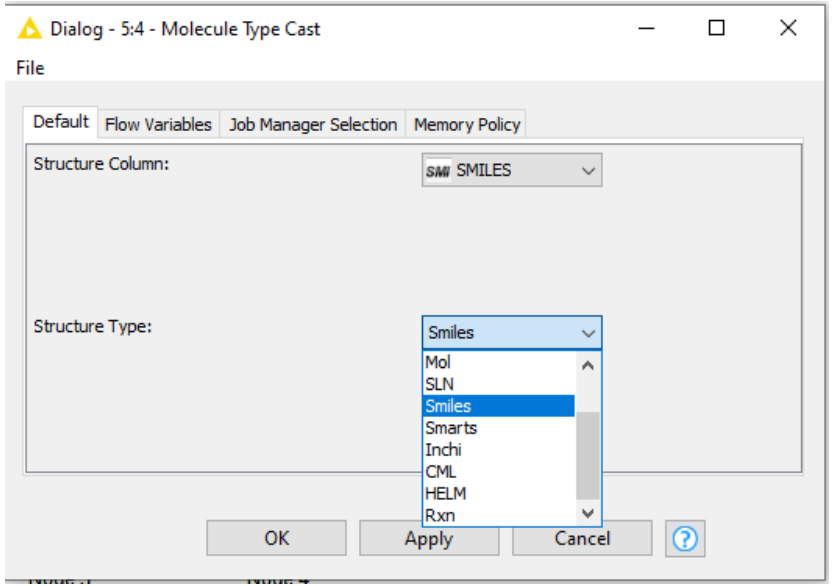
Excel Reader



Molecule Type Cast

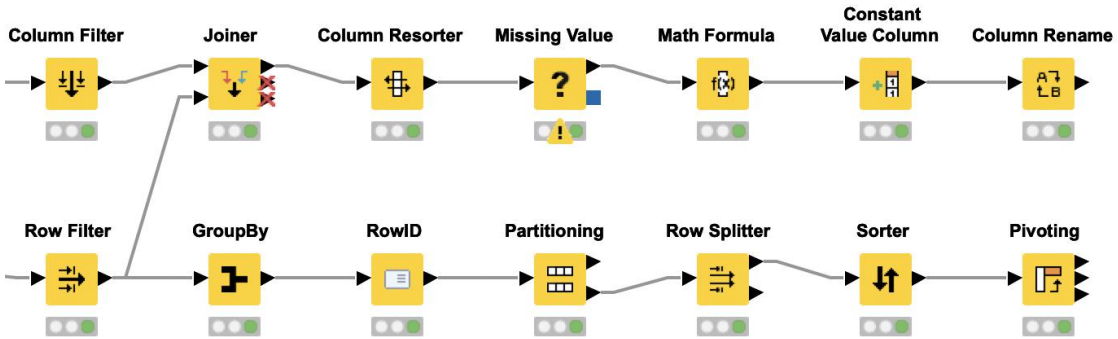
- Chemical data can be read in as string with all other Reader nodes
- Molecule Type Cast node converts of a string column into the chosen molecule type
- Variety of molecule types available:
  - e. g. Mol2, PDB, SDF, CML, HELM, SLN, Smiles, Smarts, or Rxn
- No sanity checking

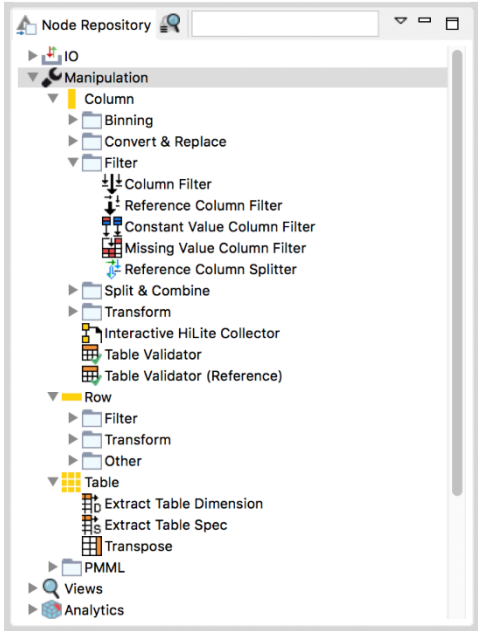




Data Manipulation Nodes

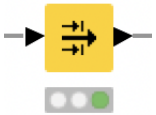
- Yellow color with a variety of input and output ports
- Apply a transformation to input data
- Many, many nodes!

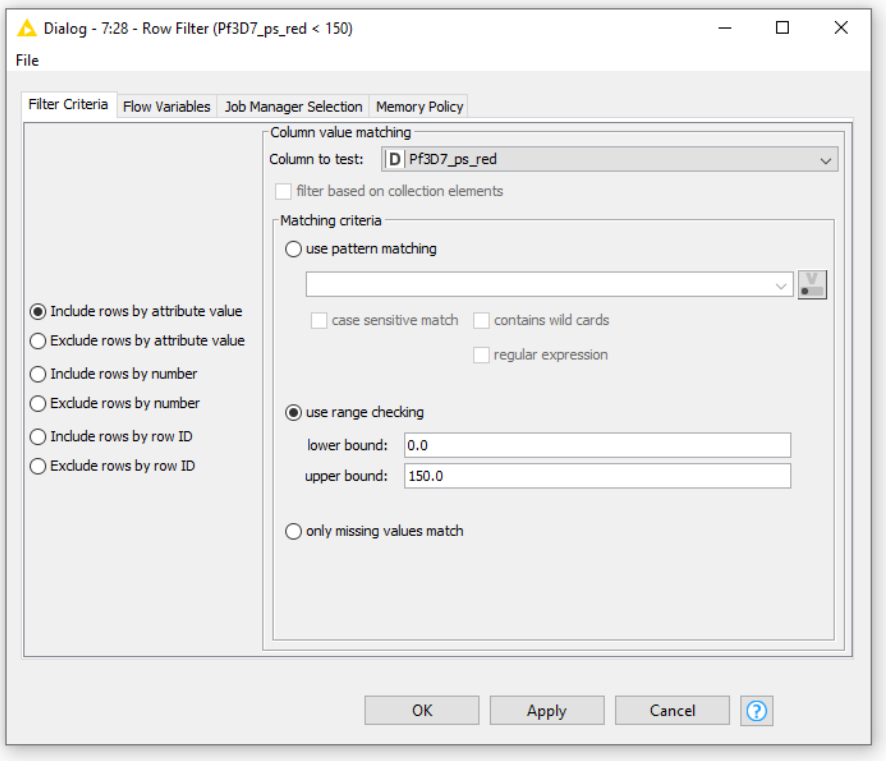


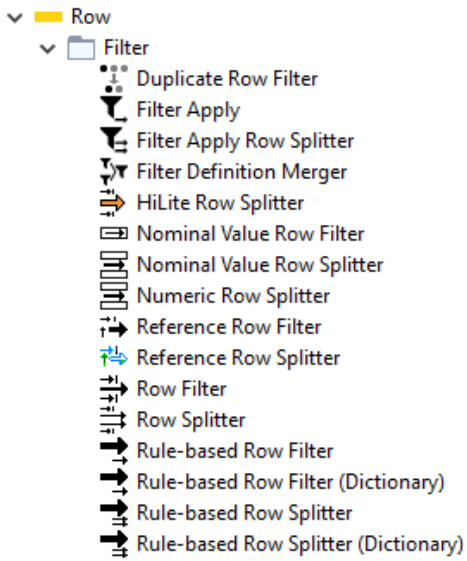


Row Filter

Row Filter







Data Aggregation (Pivoting)

Type	Name	Safety
NSAIDs	paracetamol	irritant
NSAIDs	aspirin	irritant
NSAIDs	ibuprofen	health hazard
NSAID	diclofenac	acute toxic
PPIs	omeprazole	irritant
PPIs	pantoprazole	irritant
SSRIs	fluoxetine	acute toxic
SSRIs	paroxetine	health hazard
SSRIs	citalopram	health hazard
SSRIs	sertraline	health hazard

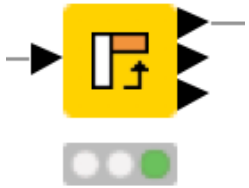
Type	Acute toxic	Health hazard	Irritant
NSAIDs	1	1	2
PPIs	?	?	2
SSRIs	1	3	?

Pivoting Node: Group - Pivot - Aggregate

Pivoting with Two Aggregation Methods

Row ID	S Type	S Safety	S Name
Row0	NSAID	irritant	paracetamol
Row1	NSAID	irritant	aspirin
Row2	NSAID	health hazard	ibuprofen
Row3	NSAID	acute toxic	diclofenac
Row4	PPIs	irritant	omeprazole
Row5	PPIs	irritant	pantoprazole
Row7	SSRIs	acute toxic	fluoxetine
Row8	SSRIs	health hazard	paroxetine
Row9	SSRIs	health hazard	citalopram
Row10	SSRIs	health hazard	sertraline

Pivoting



Row ID	S Type	I acute toxic+Count(Name)	I health hazard+Count(Name)	I irritant+Count(Name)
Row0	NSAID	1	1	2
Row1	PPIs	?	?	2
Row2	SSRIs	1	3	?

Pivoting Node: Group - Pivot - Aggregate

Pivoting

I mol..	D standard_value	S standard_relation	S standard_units	S standard_type	D pchemb.	S target_chemb
152260	6.1	=	nM	Ki	8.21	CHEMBL214
152137	7.8	=	nM	Ki	8.11	CHEMBL214
152134	120	=	nM	Ki	6.92	CHEMBL214
152080	210	=	nM	Ki	6.68	CHEMBL214
52338	47	=	nM	Ki	7.33	CHEMBL214
40962	2,759.7	=	nM	Ki	5.56	CHEMBL214
40961	10,000	>	nM	Ki	?	CHEMBL214
40178	10,000	>	nM	Ki	?	CHEMBL214
40145	10,000	>	nM	Ki	?	CHEMBL214
40144	3,967.7	=	nM	Ki	5.4	CHEMBL214
40143	405.6	=	nM	Ki	6.39	CHEMBL214
40142	10,000	>	nM	Ki	?	CHEMBL214
40141	10,000	>	nM	Ki	?	CHEMBL214
40138	10,000	>	nM	Ki	?	CHEMBL214
40060	990.7	=	nM	Ki	6	CHEMBL214
37553	31.1	=	nM	Ki	7.51	CHEMBL214
37476	5.05	=	nM	Ki	8.3	CHEMBL214
37323	40.7	=	nM	Ki	7.39	CHEMBL214
36988	14.4	=	nM	Ki	7.84	CHEMBL214
36732	170	=	nM	Ki	6.77	CHEMBL214
36730	10,000	>	nM	Ki	?	CHEMBL214
36357	355	=				
36324	1.8	=				
36287	139	=				
36144	319	=				
36144	48.8	=				
35706	778	=				
35706	142	=				
35672	554	=				
35623	10,000	>				
35324	1.3	=				
33781	80	=				
33781	80	=				
33681	168	=				

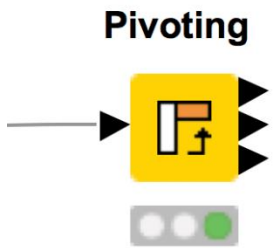
Row ID	I molregno	S CHEMBL214+pref_name	S CHEMBL214+standard_type	D CHEMBL214+standard_value	S CHEMBL214+standard_relation	S CHEMBL214+standard_units	D CHEMBL214+pchembl_value	S CHEMBL214+assay_chembl_id
Row0	2214	Serotonin 1a (5-HT1a) receptor						
Row1	2246	Serotonin 1a (5-HT1a) receptor	Ki	6,620.9	=	nM	5.18	CHEMBL616124
Row2	2261	Serotonin 1a (5-HT1a) receptor	Ki	6.38	=	nM	8.2	CHEMBL2026267
Row3	3587	Serotonin 1a (5-HT1a) receptor	Ki	6	=	nM	8.22	CHEMBL870197
Row4	3599	Serotonin 1a (5-HT1a) receptor	Ki	12	=	nM	7.92	CHEMBL1030618
Row5	3854	Serotonin 1a (5-HT1a) receptor	Ki	296	=	nM	6.53	CHEMBL615779
Row6	3859	Serotonin 1a (5-HT1a) receptor	Ki	3,600	=	nM	5.44	CHEMBL3366152
Row7	4149	Serotonin 1a (5-HT1a) receptor	Ki	0.17	=	nM	9.77	CHEMBL1034993
Row8	4181	Serotonin 1a (5-HT1a) receptor	Ki	0.6	=	nM	9.22	CHEMBL1046676
Row9	4634	Serotonin 1a (5-HT1a) receptor	Ki	17.2	=	nM	7.76	CHEMBL3820056
Row10	6291	Serotonin 1a (5-HT1a) receptor	Ki	2	=	nM	8.7	CHEMBL839708
Row11	6446	Serotonin 1a (5-HT1a) receptor	Ki	6.7	=	nM	8.17	CHEMBL884575
Row12	7065	Serotonin 1a (5-HT1a) receptor	Ki	24	=	nM	7.62	CHEMBL616370
Row13	7714	Serotonin 1a (5-HT1a) receptor	Ki	190	=	nM	6.72	CHEMBL3226682
Row14	8543	Serotonin 1a (5-HT1a) receptor	Ki	22.4	=	nM	7.65	CHEMBL616120
Row15	9128	Serotonin 1a (5-HT1a) receptor	Ki	344	=	nM	6.46	CHEMBL615460
Row16	10841	Serotonin 1a (5-HT1a) receptor	Ki	7,943	>	nM	?	CHEMBL615975
Row17	12494	Serotonin 1a (5-HT1a) receptor	Ki	4	=	nM	8.4	CHEMBL616370
Row18	[illegible]	Serotonin 1a (5-HT1a) receptor	Ki	0.371	=	nM	9.43	CHEMBL1035001

Pivoting Node: Group - Pivot - Aggregate

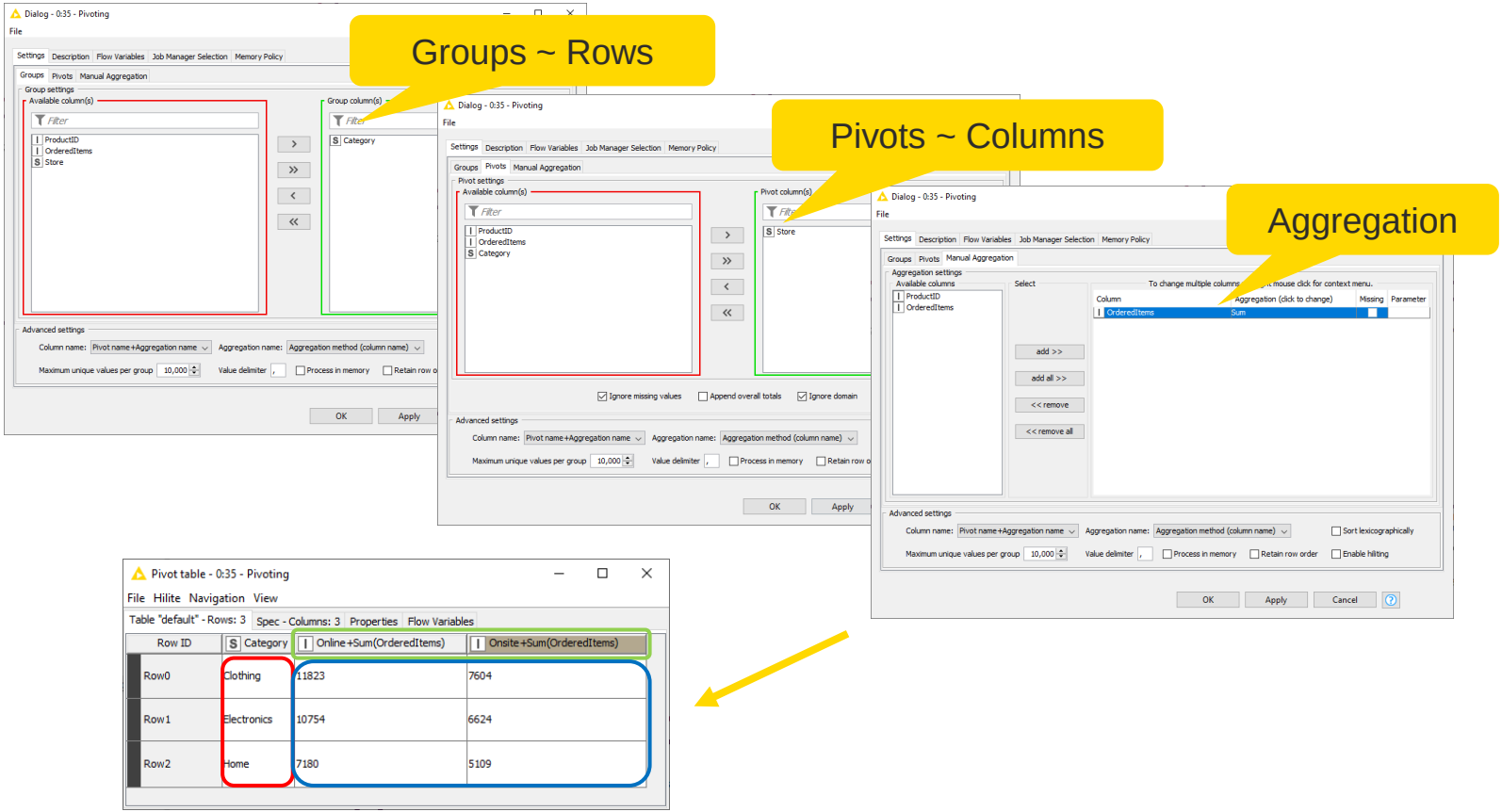
Pivoting

Performs pivoting on selected columns for grouping and pivoting

- Values of group columns become unique rows
- Values of the pivot columns become unique columns for each set of column combination together with each aggregation
- Many aggregation methods are provided (similar to GroupBy)

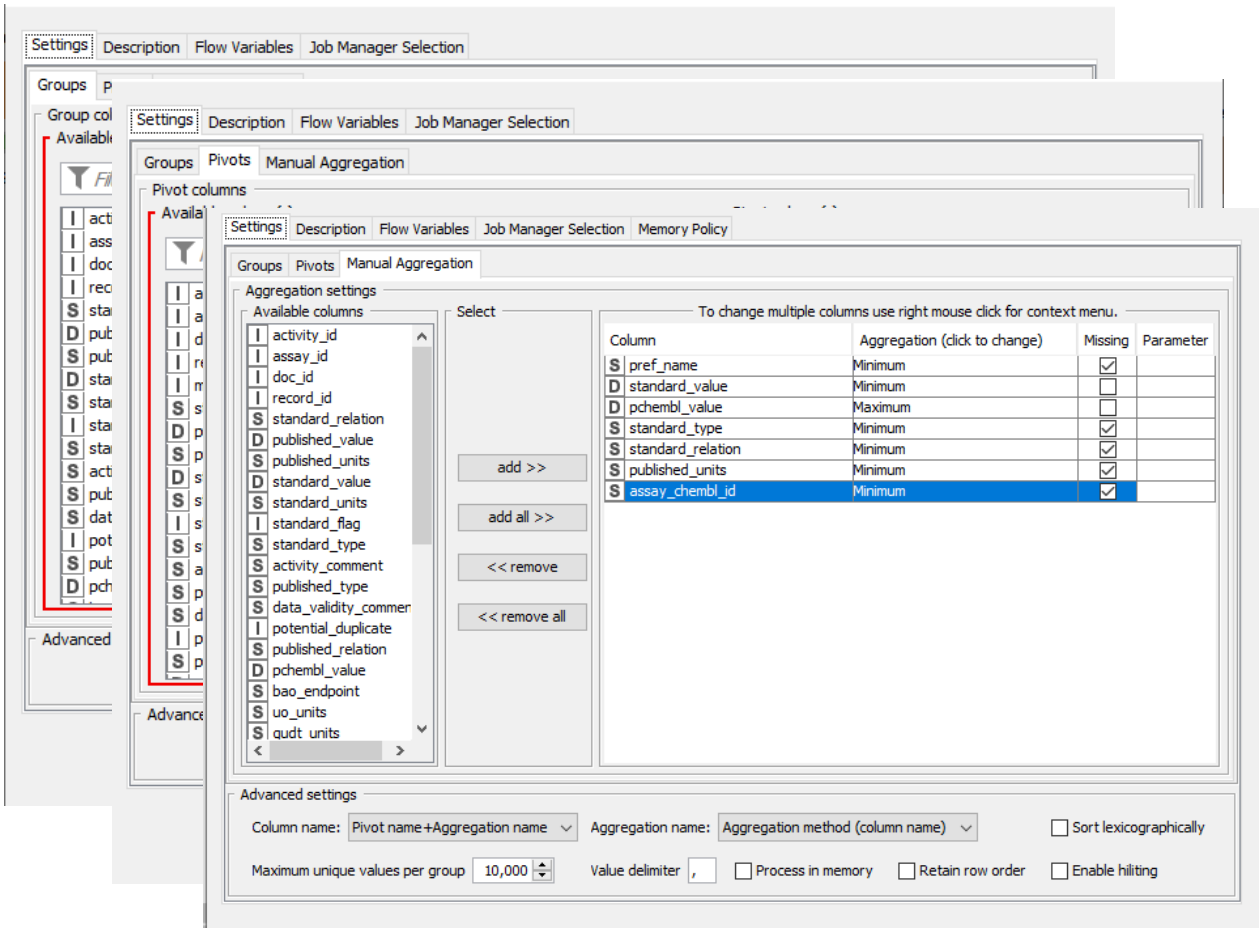


Pivoting

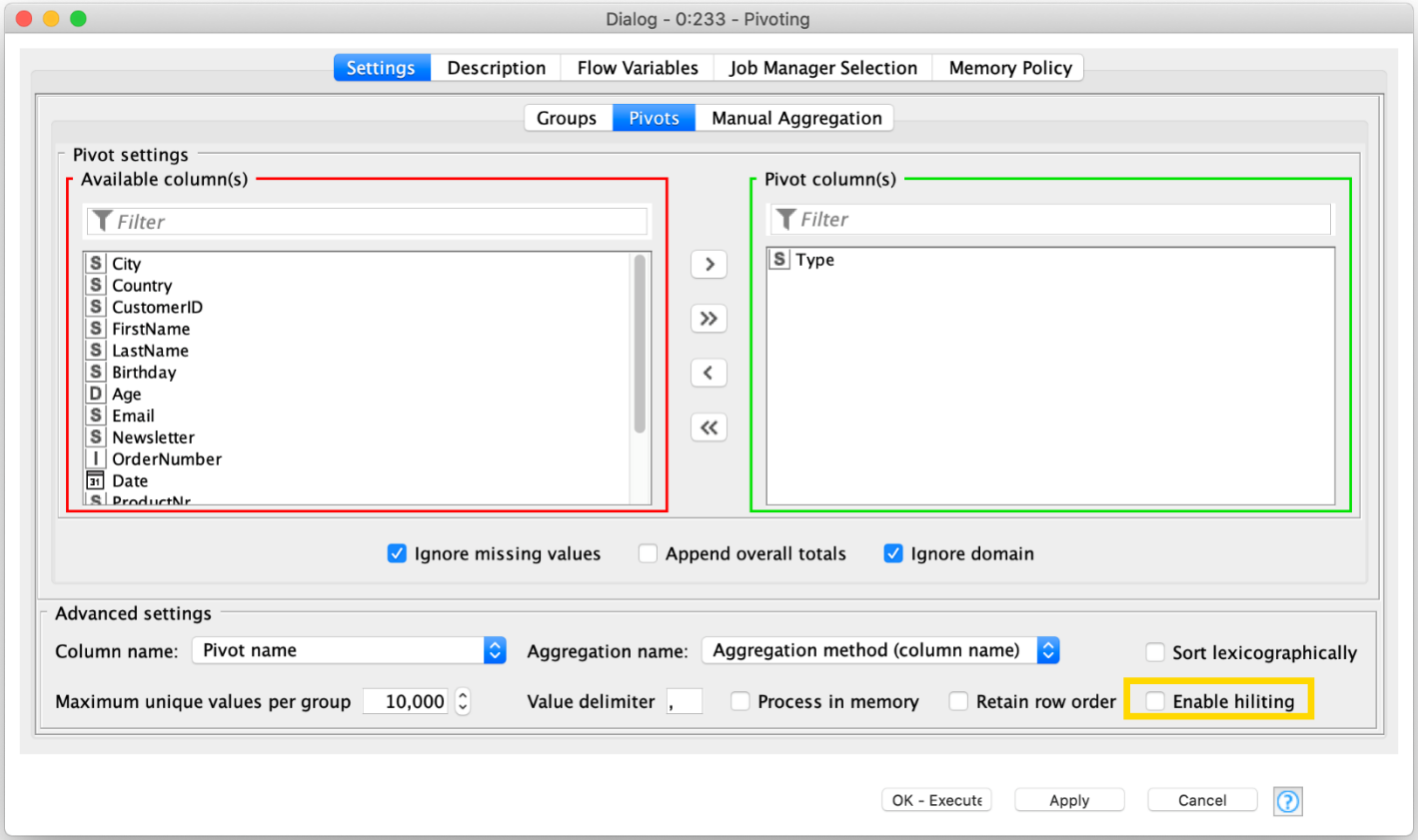


Data Pivoting

DB Pivot



Enable Hiliting!



Data Aggregation (GroupBy)

Type	Name	Weigt
NSAID	paracetamol	151.17
NSAID	aspirin	180.16
NSAID	ibuprofen	206.29
NSAID	diclofenac	296.15
PPI	omeprazole	345.42
PPI	pantoprazole	383.38
SSRI	fluoxetine	309.33
SSRI	paroxetine	329.37
SSRI	citalopram	324.40
SSRI	sertraline	342.70

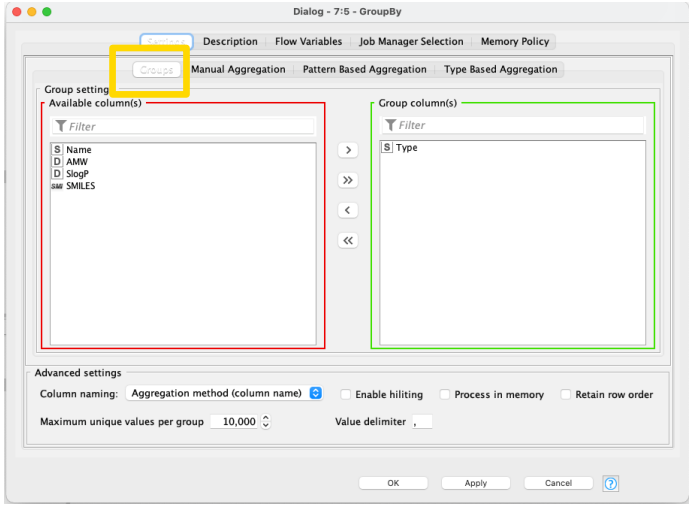
Type	Count(Name)	Mean(Weight)
NSAID	4	208.44
PPI	2	364.40
SSRI	4	326.45

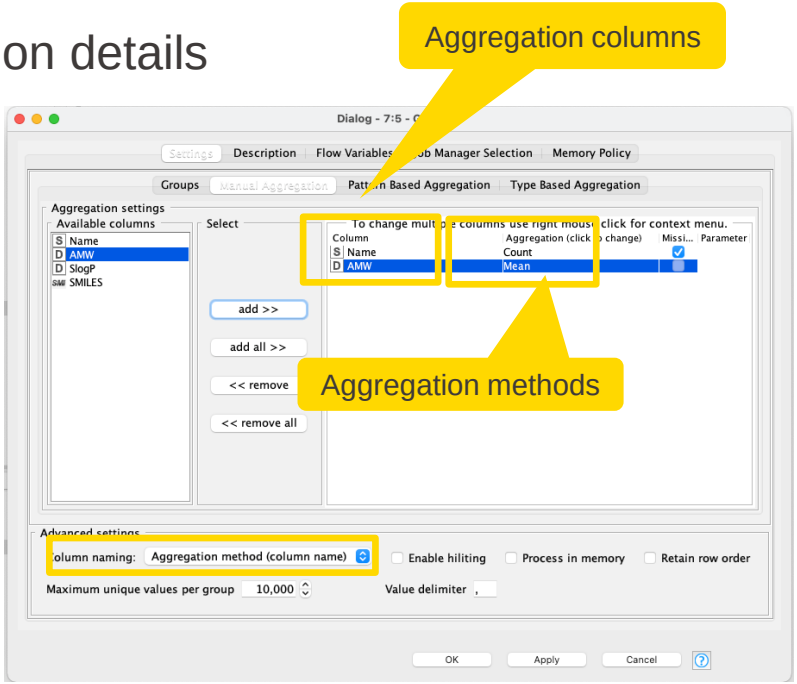
Aggregated on Type (group) by Count (aggregation method) and Mean (aggregation method)

GroupBy

Aggregate to summarize data

- First tab provides grouping options
- Second tab provides control over aggregation details

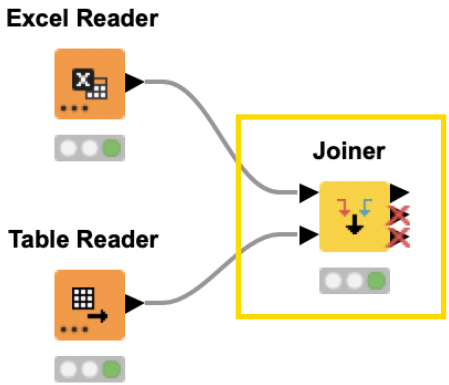




YouTube KNIME TV video: https://youtu.be/bDwF-TOMtWw

Joiner

- Combines columns from two different tables
  - Top input port: “Left” data table
  - Bottom input port: “Right” data table



- Outputs:
  - Top port: Resulting joined table
  - Middle port: Unmatched rows from the left input table (top input port)
  - Bottom port: Unmatched rows from the right input table (bottom input port)

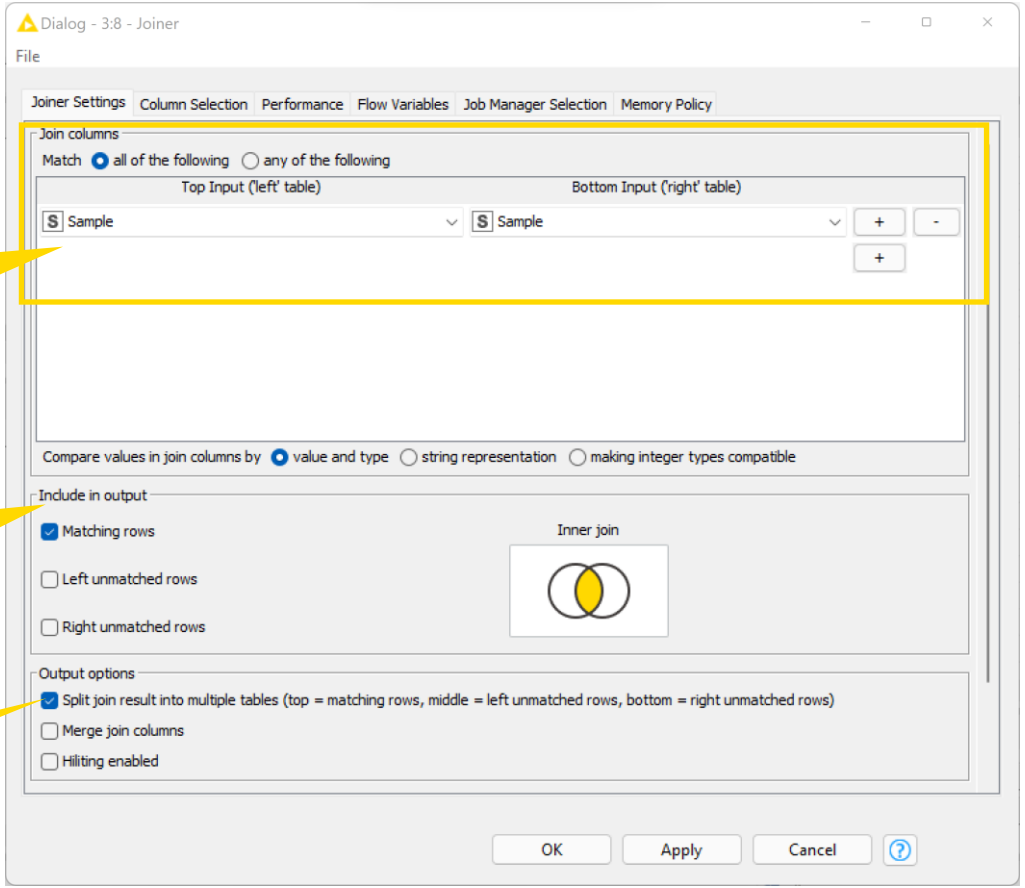
- By default the two bottom output ports are deactivated

Joiner Configuration – Linking Rows

Values to join on. Multiple joining columns are allowed

Select the rows which should be included in the joined table

Activate this checkbox to activate the bottom output ports



Joining Columns of Data – Inner Join

Left Table

molregno	chembl_id	SMILES
22	CHEMBL1794855	CCCN(CCC)
24	CHEMBL278751	CCN(C)
15	CHEMBL103772	CCCN1CC
10	CHEMBL328107	C1CN(CCN1)

Right Table

molregno	Ki_value	Ki_relation	Ki_unit
17	76.0	=	nM
65	6.56	=	nM
35	100	>	nM
15	8	=	nM
10	95.8	=	nM

Inner Join

molregno	chembl_id	SMILES	Ki_value	Ki_relation	Ki_unit
15	CHEMBL103772	CCCN1CC	8	=	nM
10	CHEMBL328107	C1CN(CCN1)	95.8	=	nM

Joining Columns of Data – Left Outer Join

Left Table

molregno	chembl_id	SMILES
22	CHEMBL1794855	CCCN(CCC)
24	CHEMBL278751	CCN(C)
15	CHEMBL103772	CCCN1CC
10	CHEMBL328107	C1CN(CCN1)

Right Table

molregno	Ki_value	Ki_relation	Ki_unit
17	76.0	=	nM
65	6.56	=	nM
35	100	>	nM
15	8	=	nM
10	95.8	=	nM

Left Outer Join

molregno	chembl_id	SMILES	Ki_value	Ki_relation	Ki_unit
22	CHEMBL1794855	CCCN(CCC)	?	?	?
24	CHEMBL278751	CCN(C)	?	?	?
15	CHEMBL103772	CCCN1CC	8	=	nM
10	CHEMBL328107	C1CN(CCN1)	95.8	=	nM

Joining Columns of Data – Right Outer Join

Left Table

molregno	chembl_id	SMILES
22	CHEMBL1794855	CCCN(CCC)
24	CHEMBL278751	CCN(C)
15	CHEMBL103772	CCCN1CC
10	CHEMBL328107	C1CN(CCN1)

Right Table

molregno	Ki_value	Ki_relation	Ki_unit
17	76.0	=	nM
65	6.56	=	nM
35	100	>	nM
15	8	=	nM
10	95.8	=	nM

Right Outer Join

molregno	chembl_id	SMILES	Ki_value	Ki_relation	Ki_unit
17	?	?	76.0	=	nM
65	?	?	6.56	=	nM
35	?	?	100	>	nM
15	CHEMBL103772	CCCN1CC	8	=	nM
10	CHEMBL328107	C1CN(CCN1)	95.8	=	nM

Joining Columns of Data – Full Outer Join

Left Table

molregno	chembl_id	SMILES
22	CHEMBL1794855	CCCN(CCC)
24	CHEMBL278751	CCN(C)
15	CHEMBL103772	CCCN1CC
10	CHEMBL328107	C1CN(CCN1)

Right Table

molregno	Ki_value	Ki_relation	Ki_unit
17	76.0	=	nM
65	6.56	=	nM
35	100	>	nM
15	8	=	nM
10	95.8	=	nM

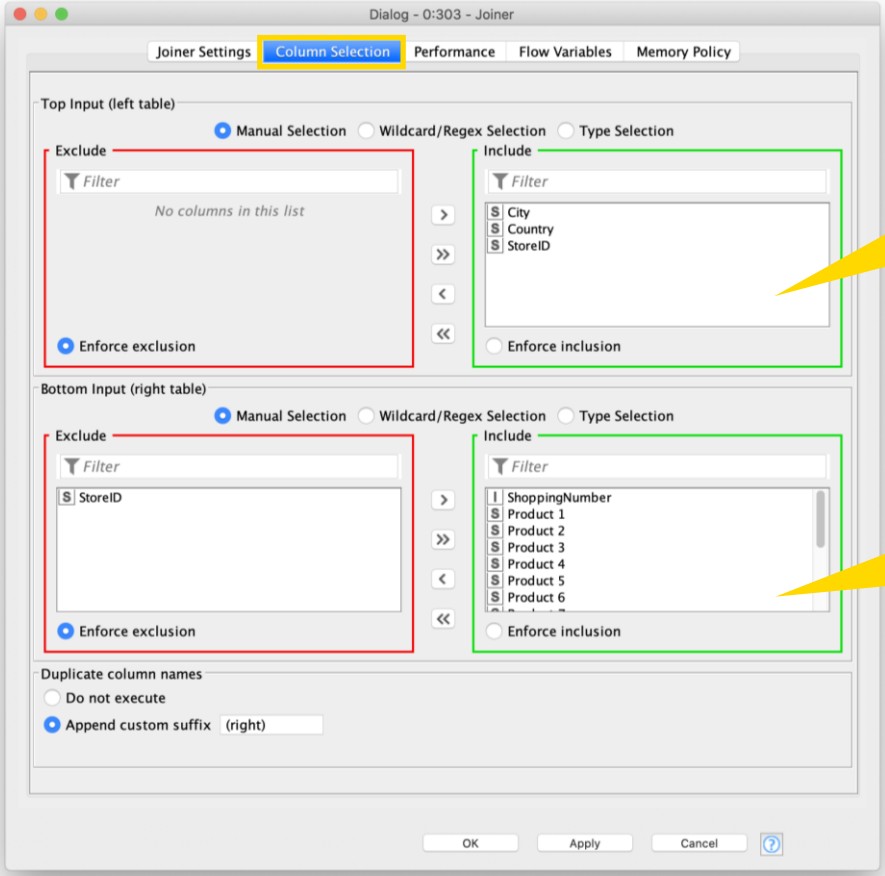
Values missing in the left table

Full Outer Join

molregno	chembl_id	SMILES	Ki_value	Ki_relation	Ki_unit
17	?	?	76.0	=	nM
65	?	?	6.56	=	nM
35	?	?	100	>	nM
15	CHEMBL103772	CCCN1CC	8	=	nM
10	CHEMBL328107	C1CN(CCN1)	95.8	=	nM
22	CHEMBL1794855	CCCN(CCC)	?	?	?
24	CHEMBL278751	CCN(C)	?	?	?

Values missing in the right table

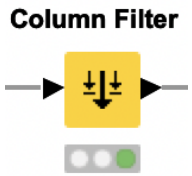
Joiner Configuration – Column Selection

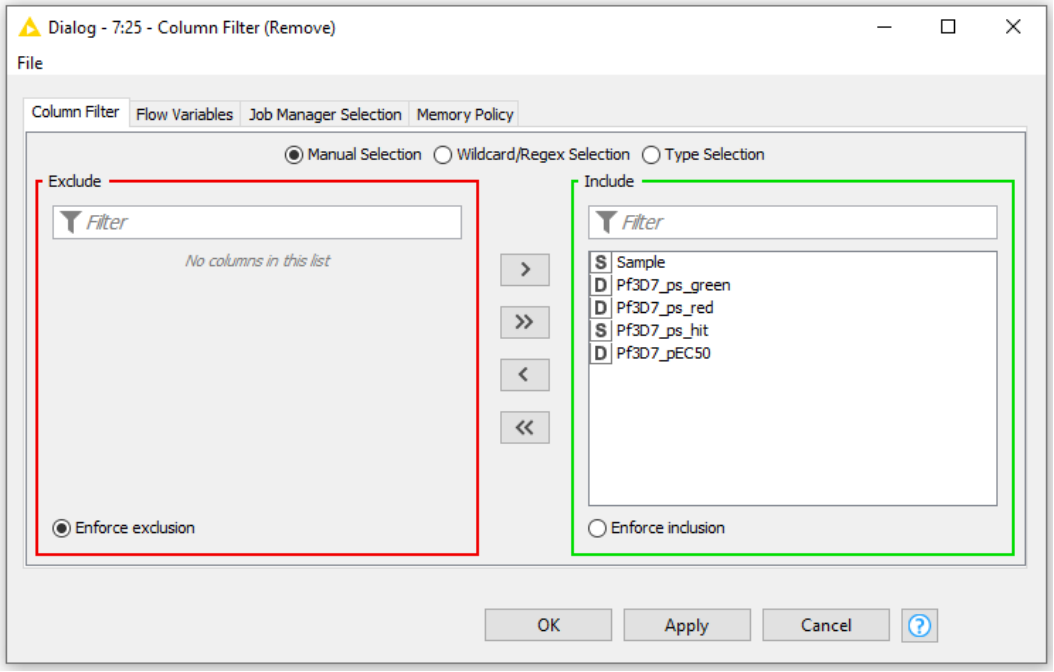


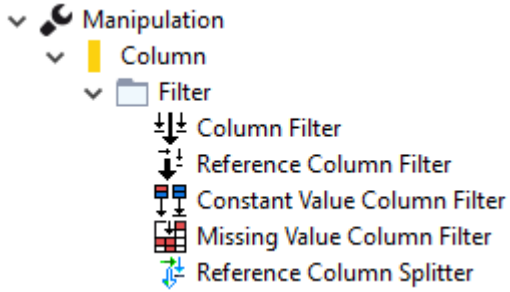
Columns from top table for joined table

Columns from lower table for joined table

Column Filter

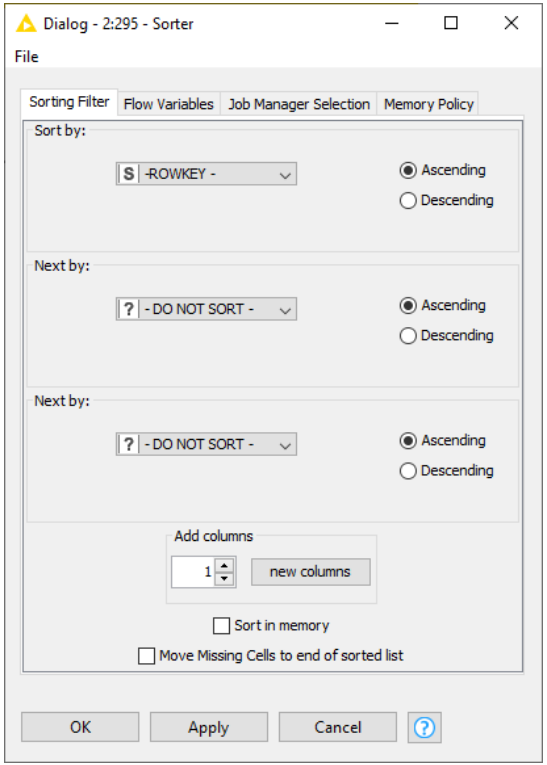






Other Useful Nodes for Row and Column Handling

Sorter



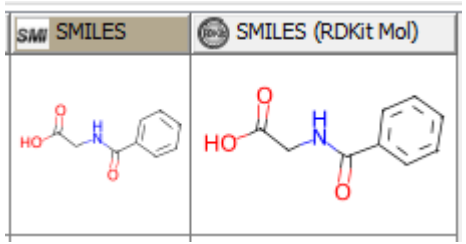
Column Resorter

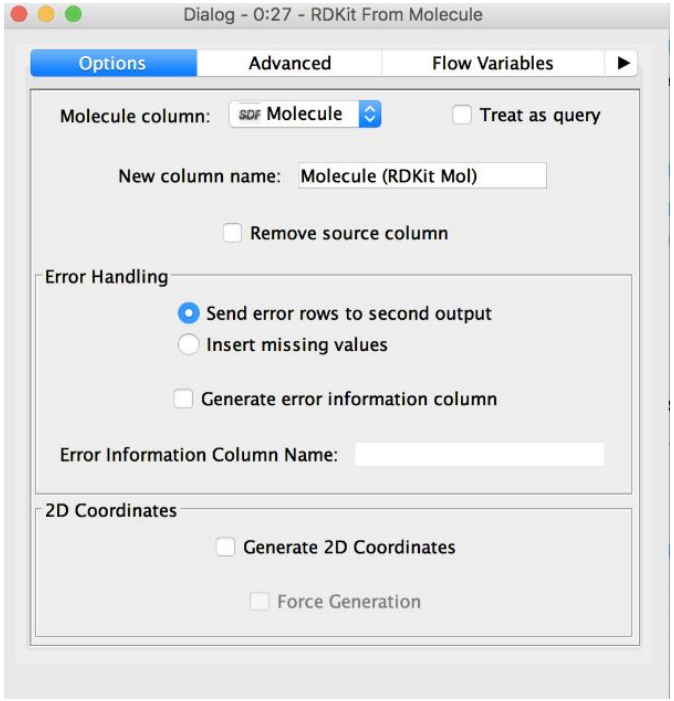
Column Splitter

Row Splitter

RDKit From Molecule

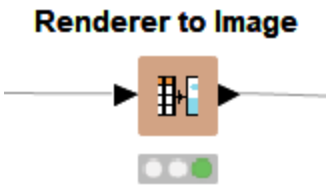
- Generates RDKit molecule column from a molecule string representation (SMILES, SDF or SMARTS)
- Can generate 2D coordinates
- Does some “chemical sanity check”

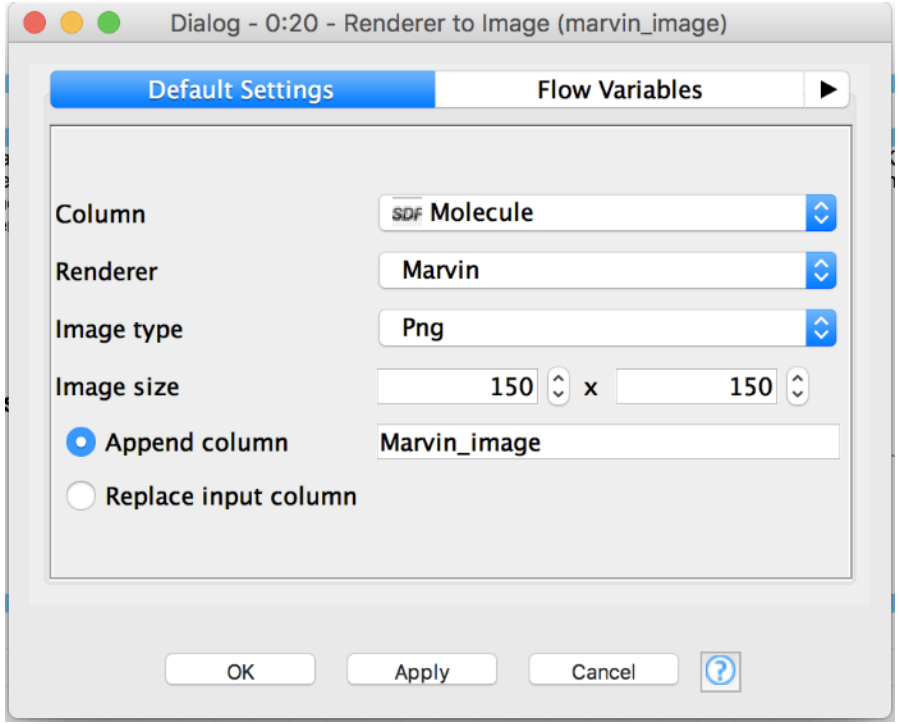




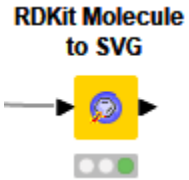
Renderer to Image

- Needed to show rendered molecules in Interactive Views
- Convert chemical structure to an image
- Various image types



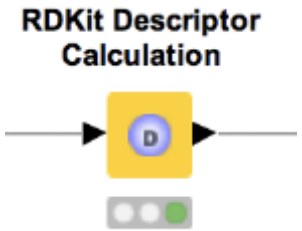


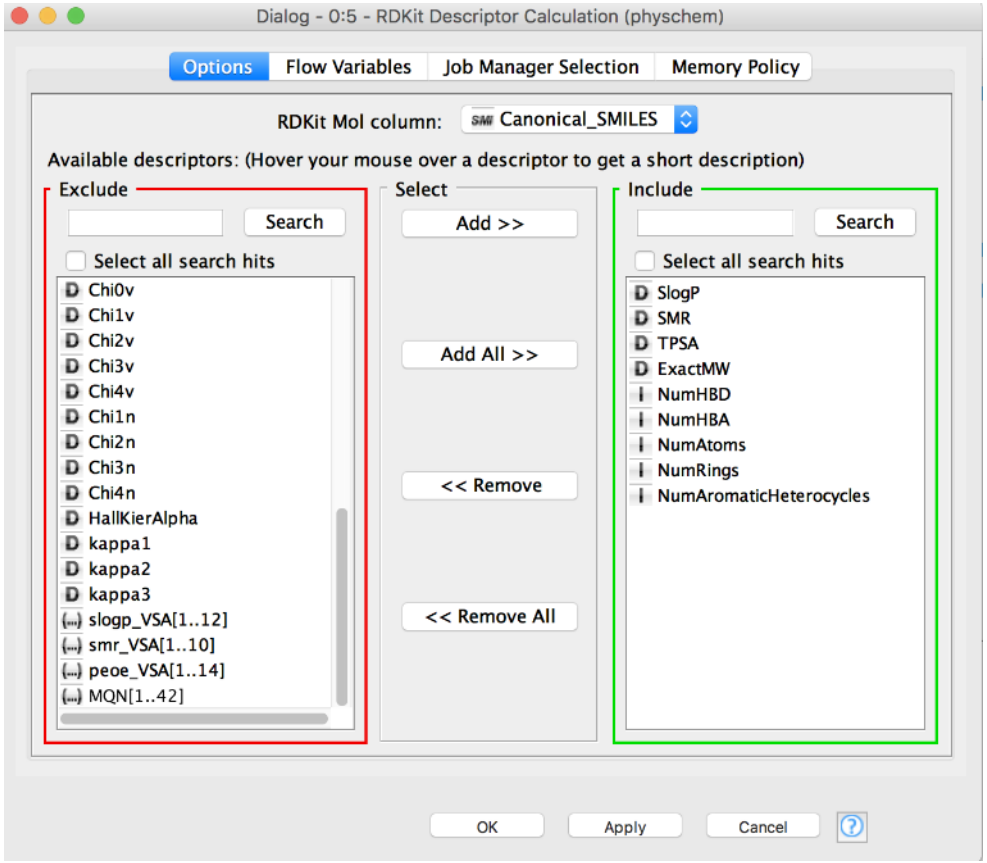
- Alternative:



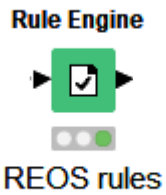
Compute Descriptors

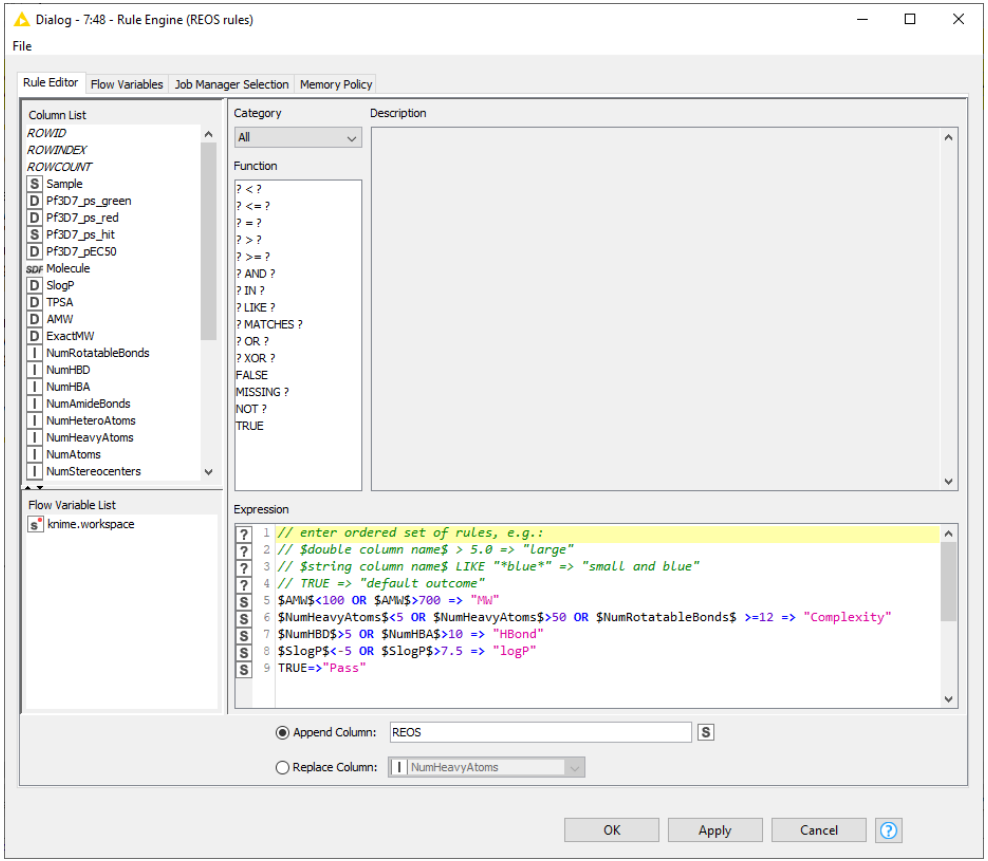
- Not all descriptors from the Python library available in the node





Rule Engine





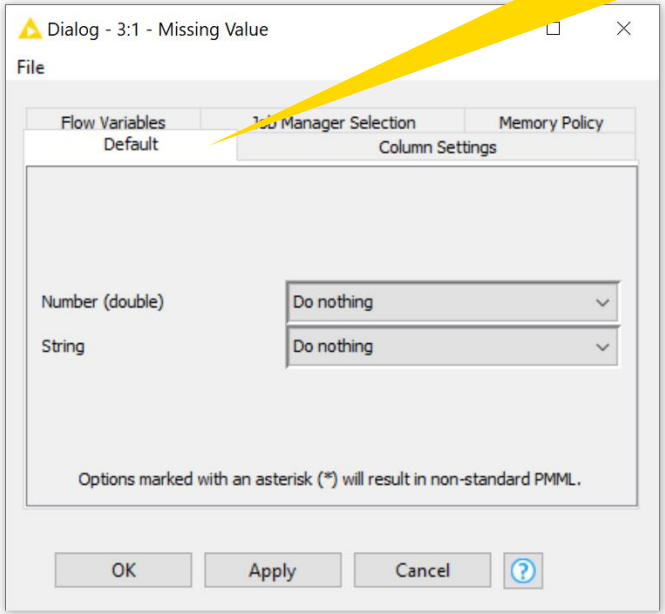
IF => THEN
ELSE

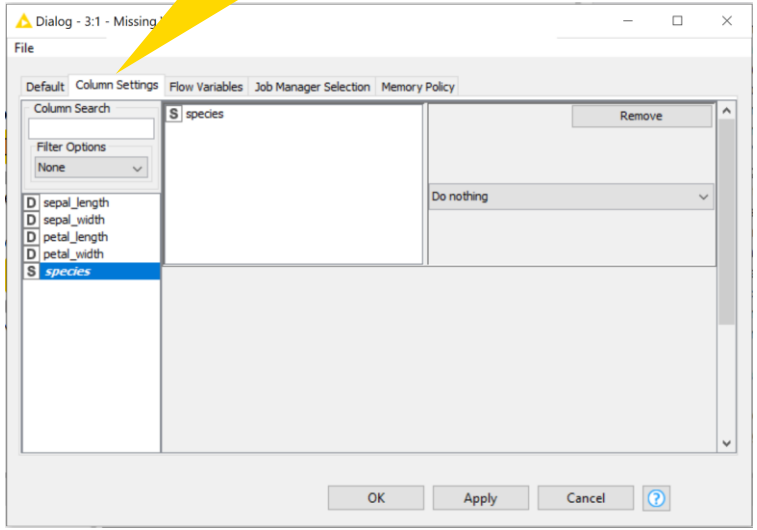
Missing Value

helps to handle missing values

Define default values for all columns of specific type

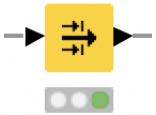
Define values for every single column

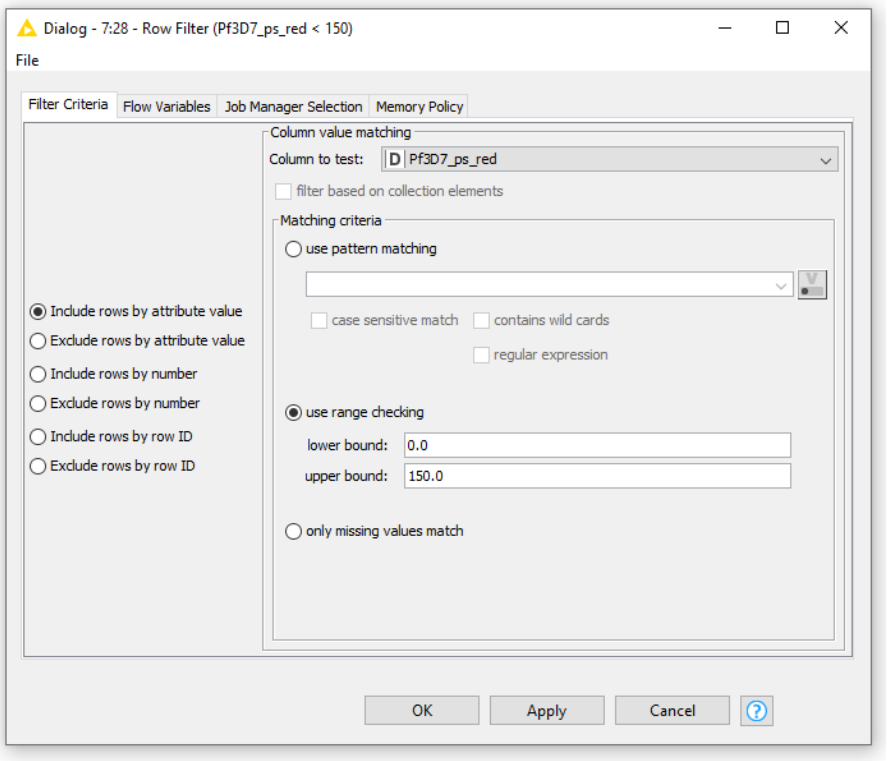


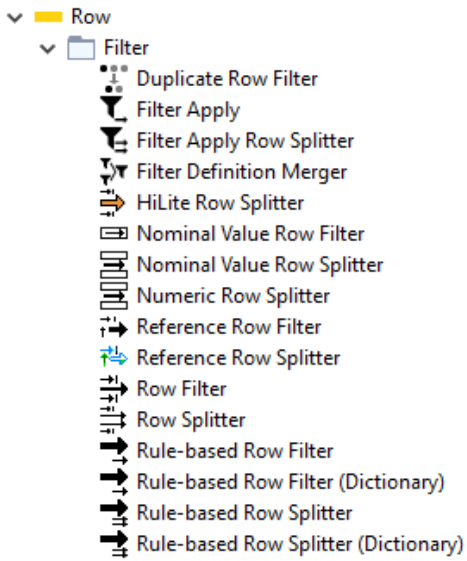


Row Filter

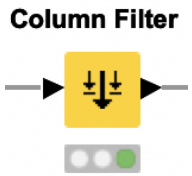
Row Filter

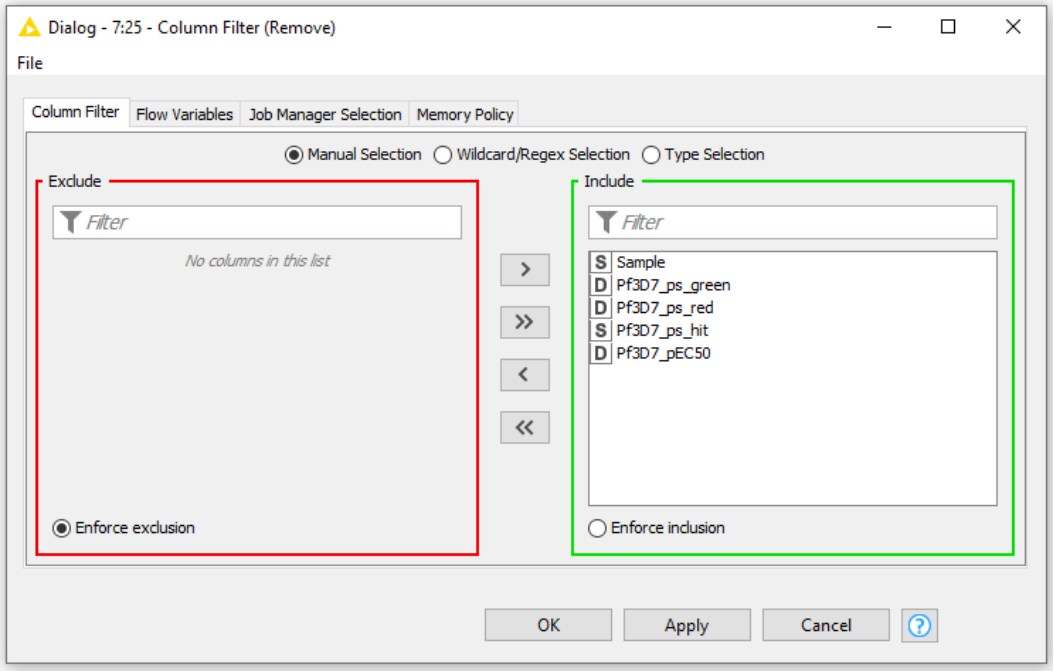


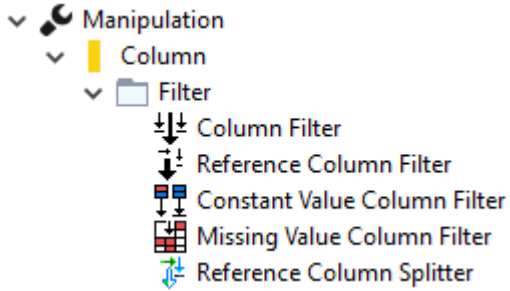




Column Filter

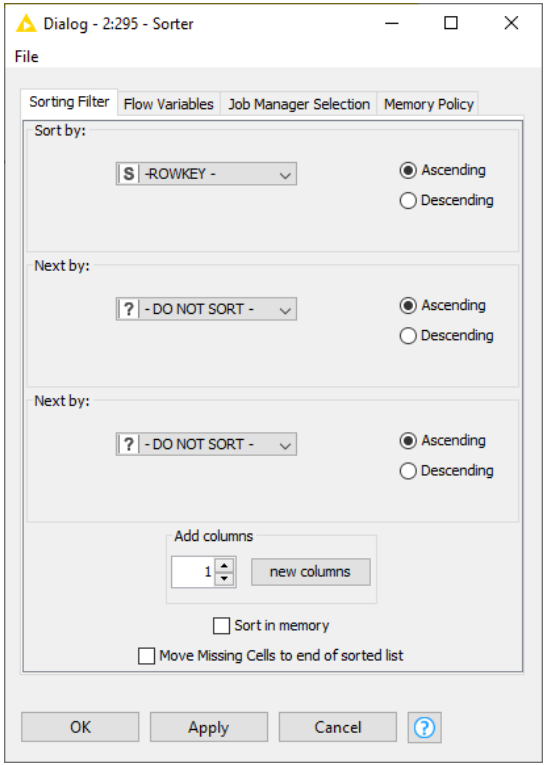






Other Useful Nodes for Row and Column Handling

Sorter



Column Resorter

Column Splitter

Row Splitter

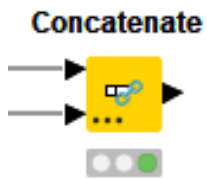
Concatenate Tables

Table A

RowID	Mol Reg No	Chembl ID	Ki value
0	35	CHEMBL15435	100.0
1	15	CHEMBL1794855	8.0

Table B

RowID	Mol Reg No	Chembl ID
0	15	CHEMBL1794855
1	10	CHEMBL278751
2	22	CHEMBL103772



union of columns

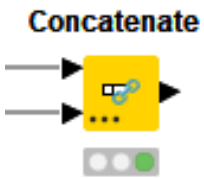
RowID	Mol Reg No	Chembl ID	Ki value
0	35	CHEMBL15435	100.0
1	15	CHEMBL1794855	8.0
0_dup	15	CHEMBL1794855	
1_dup	10	CHEMBL278751	
2	22	CHEMBL103772	

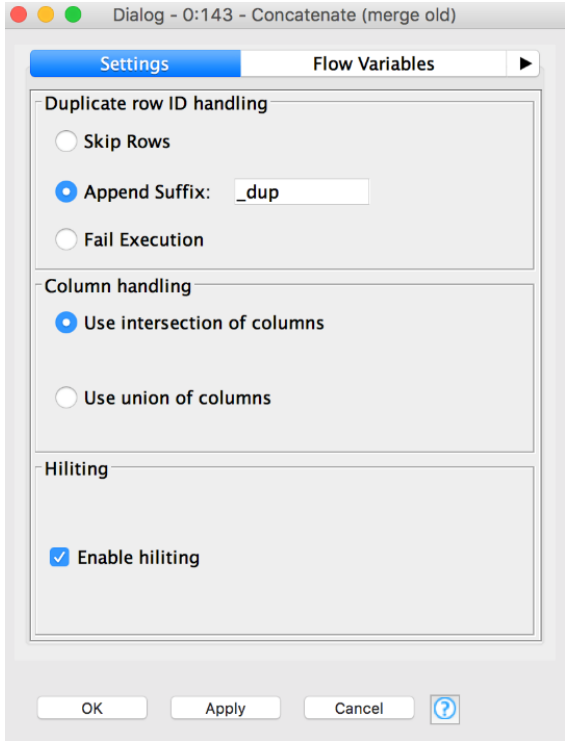
intersection of columns

Concatenate

Combine rows from 2 or more tables with shared columns

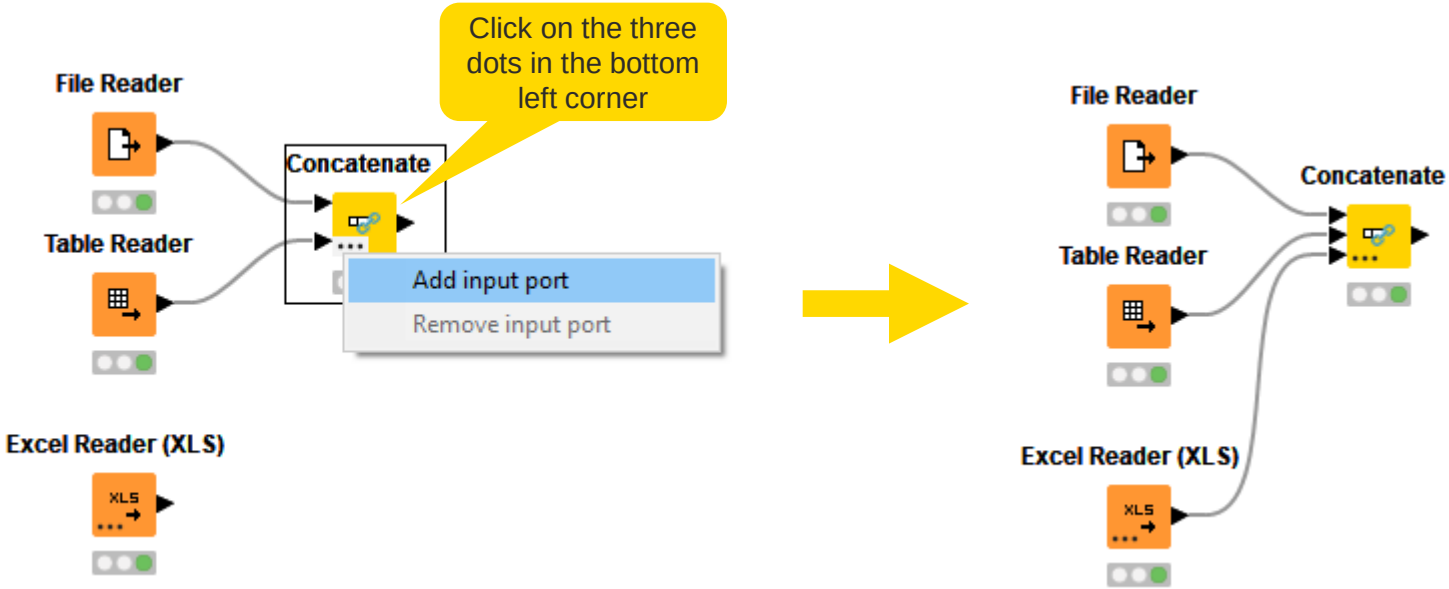
- Handles duplicate row keys gracefully
- Take the union or intersection of columns





Dynamic Ports

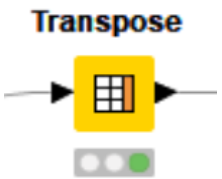
Add and remove node ports based on your needs, e.g. in order to concatenate three or more tables

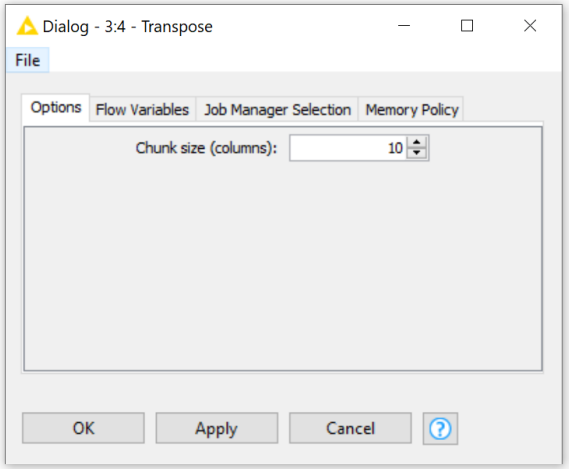


Transpose

Row ID	S column1	S column2	S column3
Row0	1	2	3
Row1	1	2	3
Row2	1	2	3
Row3	1	2	3

Row ID	S Row0	S Row1	S Row2	S Row3
column1	1	1	1	1
column2	2	2	2	2
column3	3	3	3	3

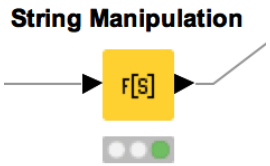


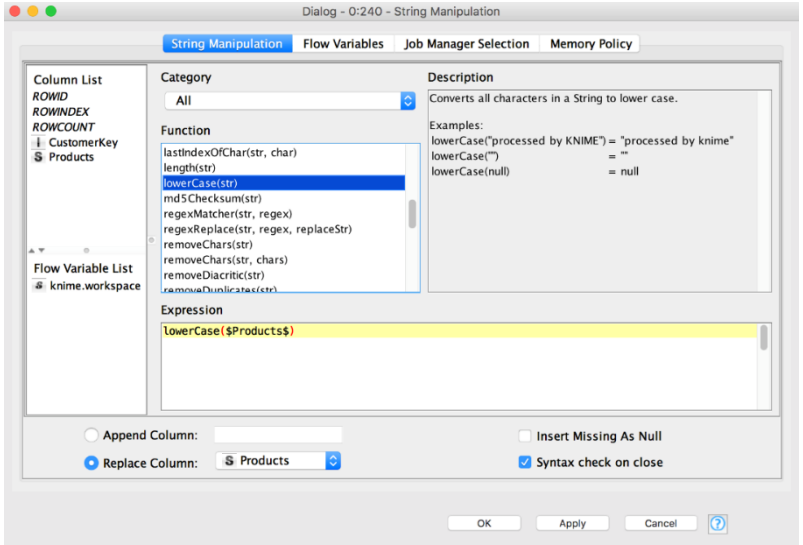


String Manipulation

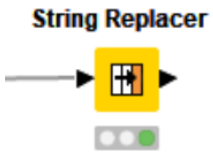
Create and edit values in String columns

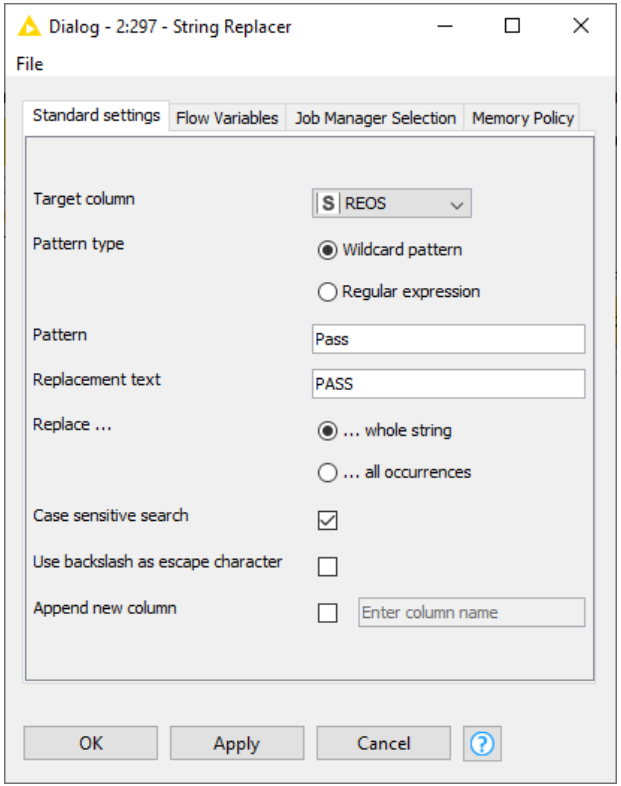
- Clean up capitalization (e.g. Lowercase)
- Replace strings
- Modify existing strings or create new columns

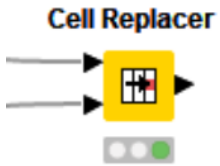


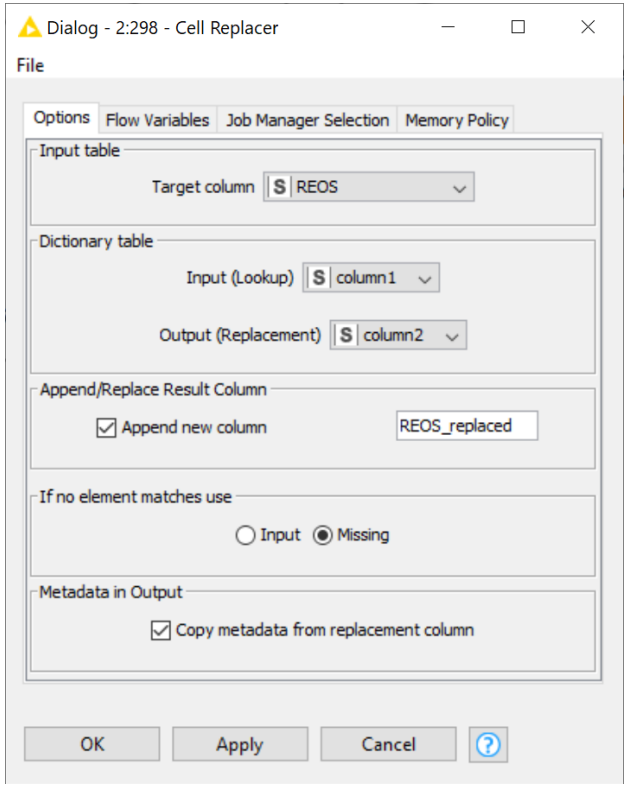


More Nodes

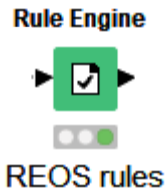


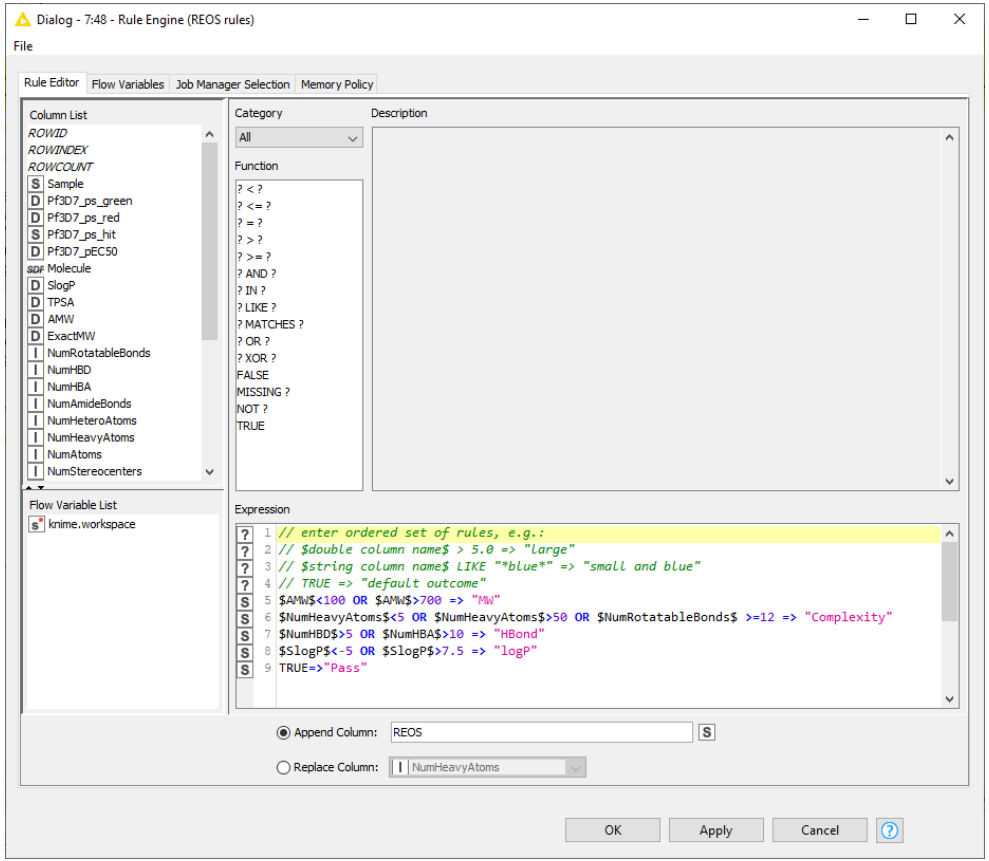






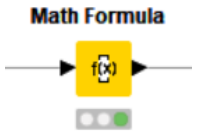
Rule Engine

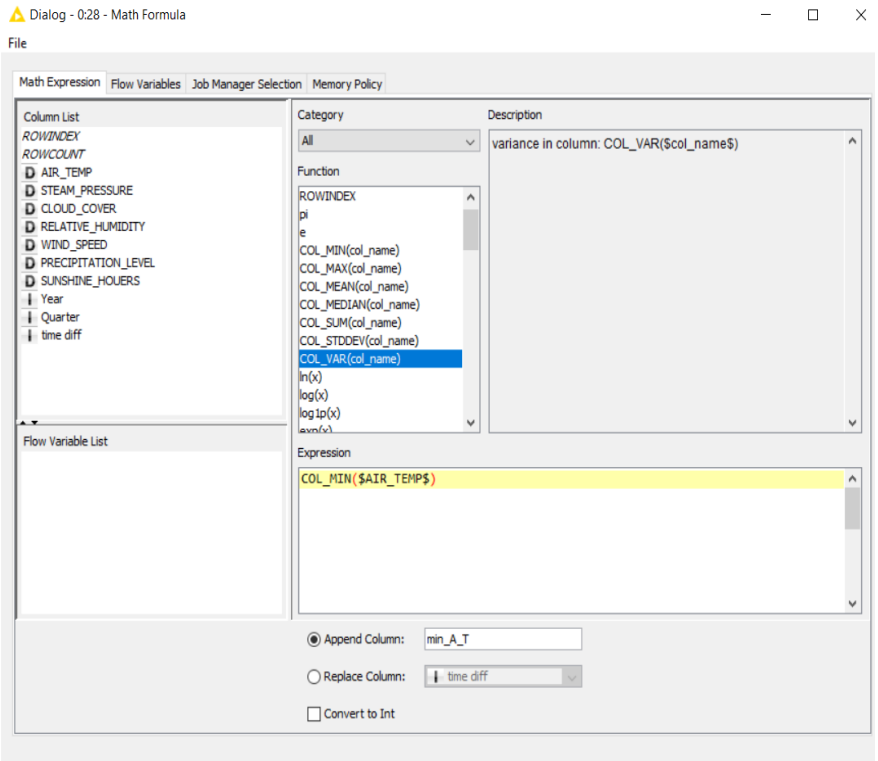




Math Formula

- Row-wise calculations
- Some column-wise statistics
- Lots of mathematical functions
- Double click on function, then select column





Column Expression

- Append or modify an arbitrary number of columns using expressions
- Many different functions are available
- No restriction on number of lines per expression allow to write complex expressions
- Part of the KNIME Labs extension

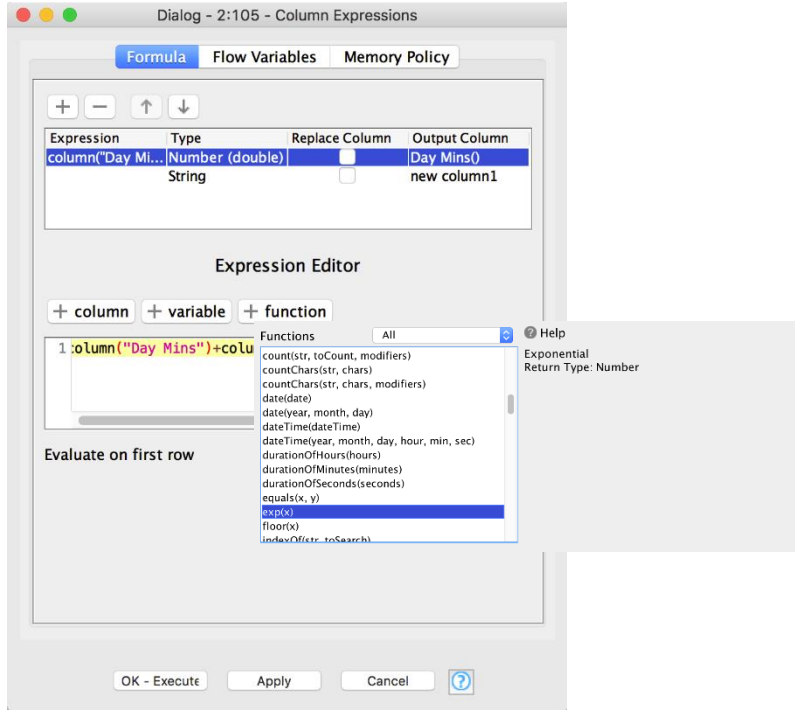
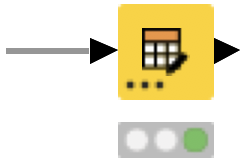


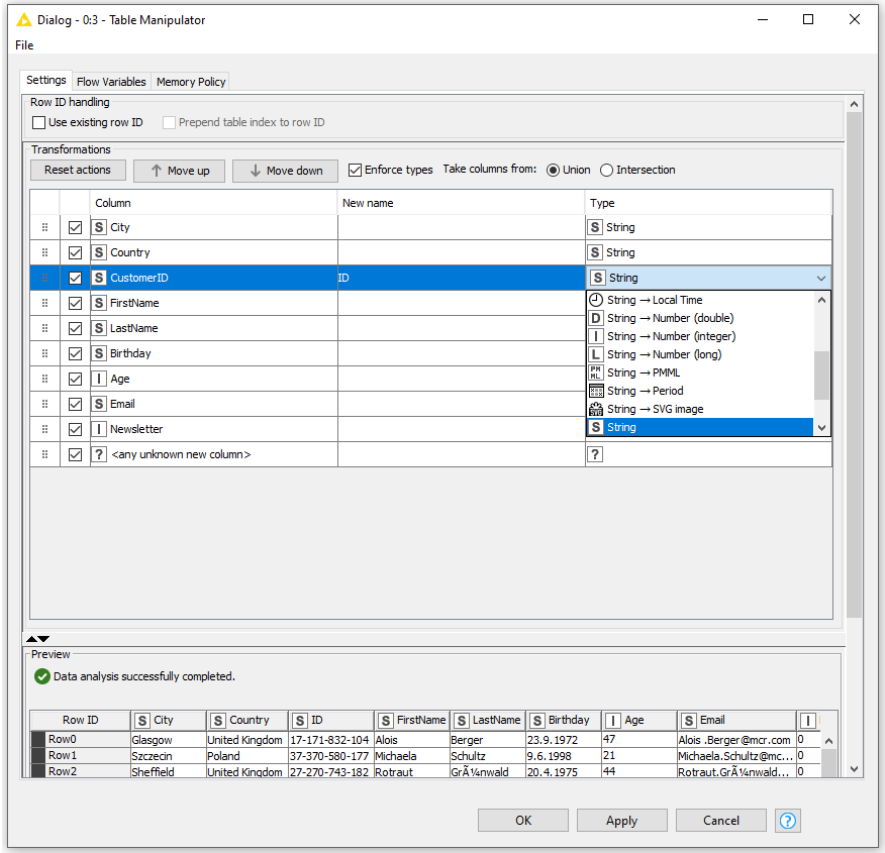
Table Manipulator

Allows for

- Concatenation of multiple files/tables
- Column filtering
- Column sorting
- Column renaming
- Column type mapping

Table Manipulator

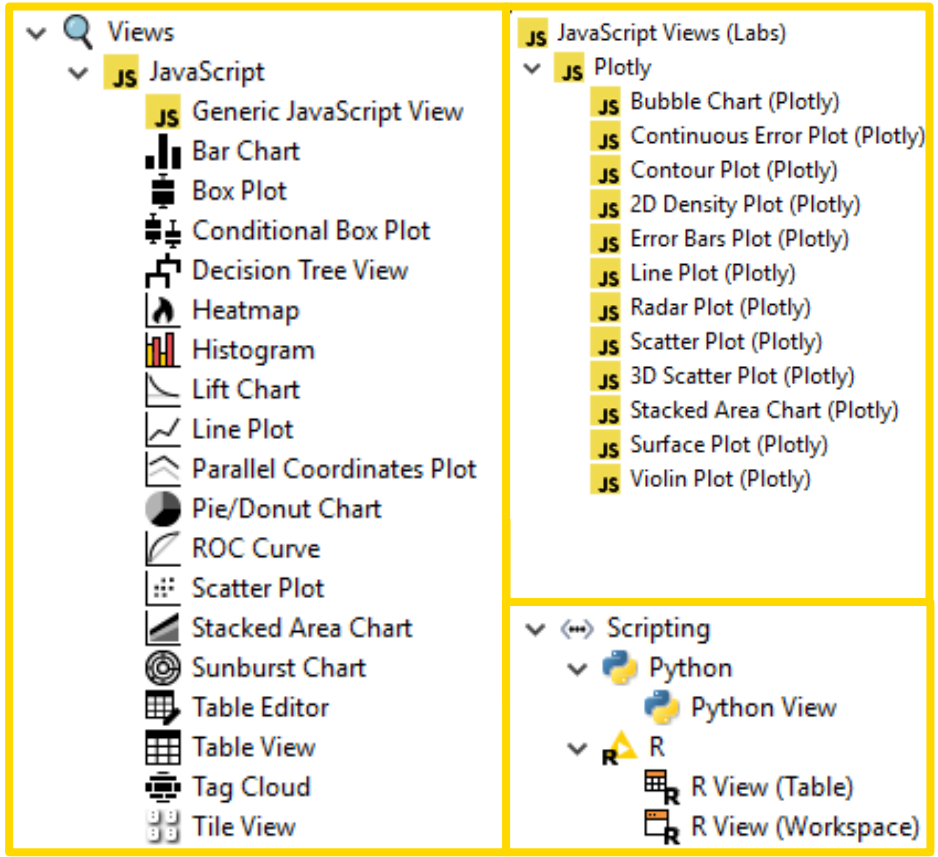




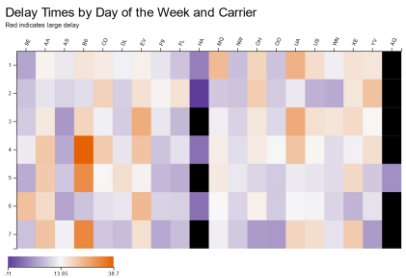
Data visualisation

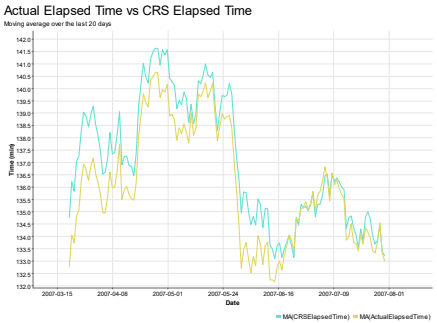
Data Visualization

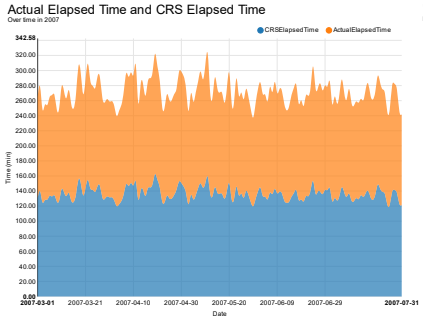
- Large selection of easy to use visualization nodes
  - Web-based and interactive
  - Dedicated nodes, no scripting required
- Plotly nodes
  - Similar but integrated from an external library
- R and Python View nodes for highly customizable graphics
  - Require scripting

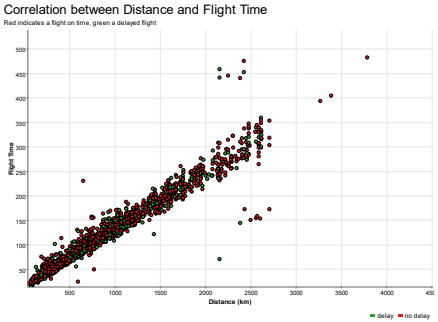


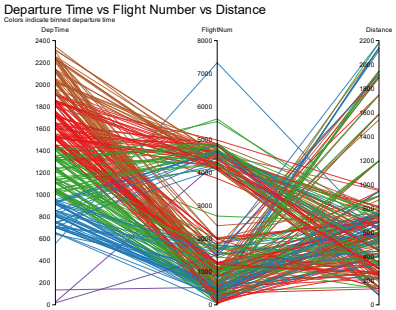
Visualization

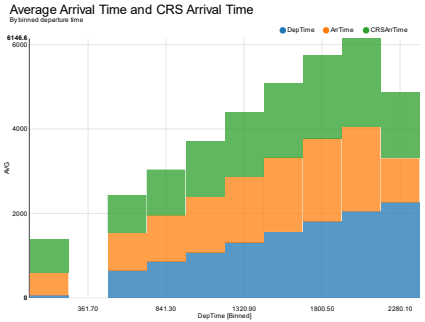


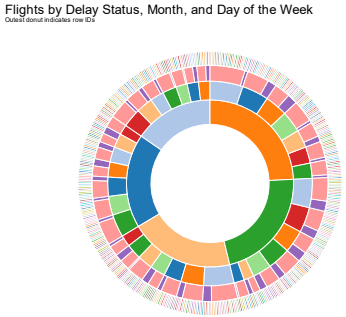












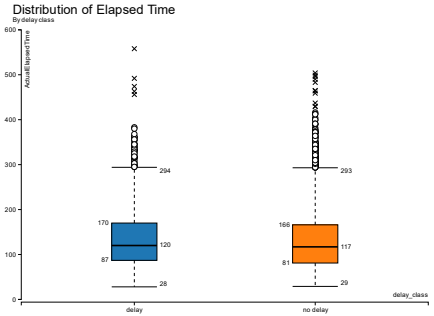
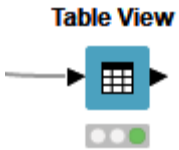
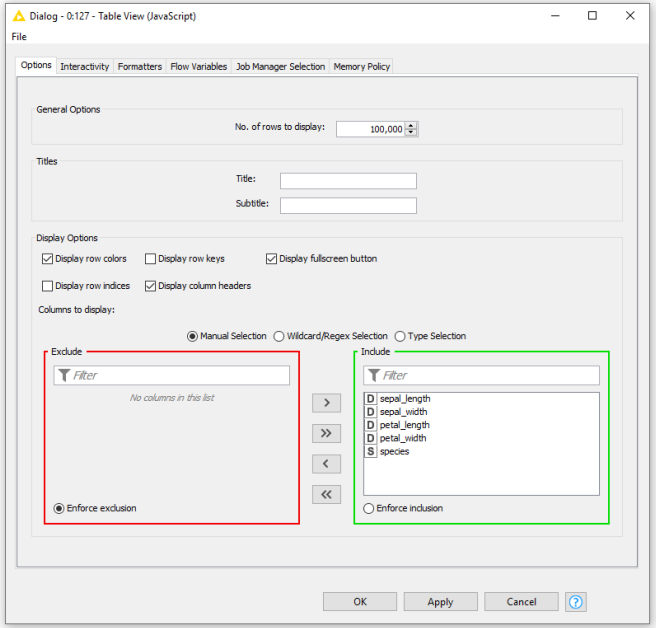
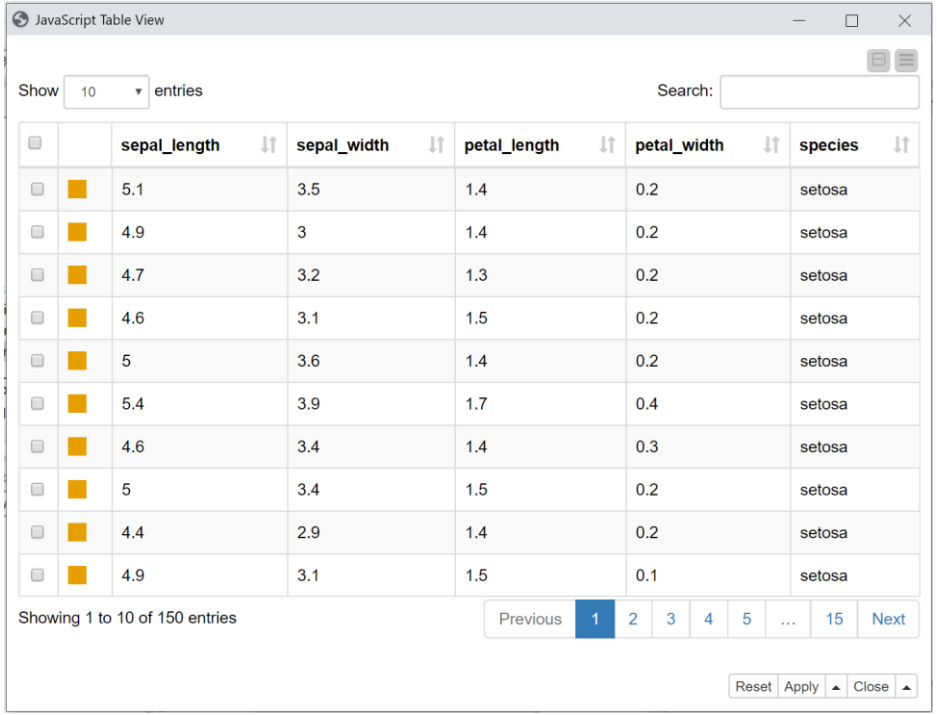


Table View

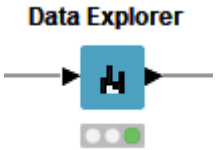






sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

Data Explorer



Data Explorer View

Numeric | Nominal | Data Preview

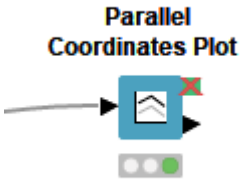
Search:

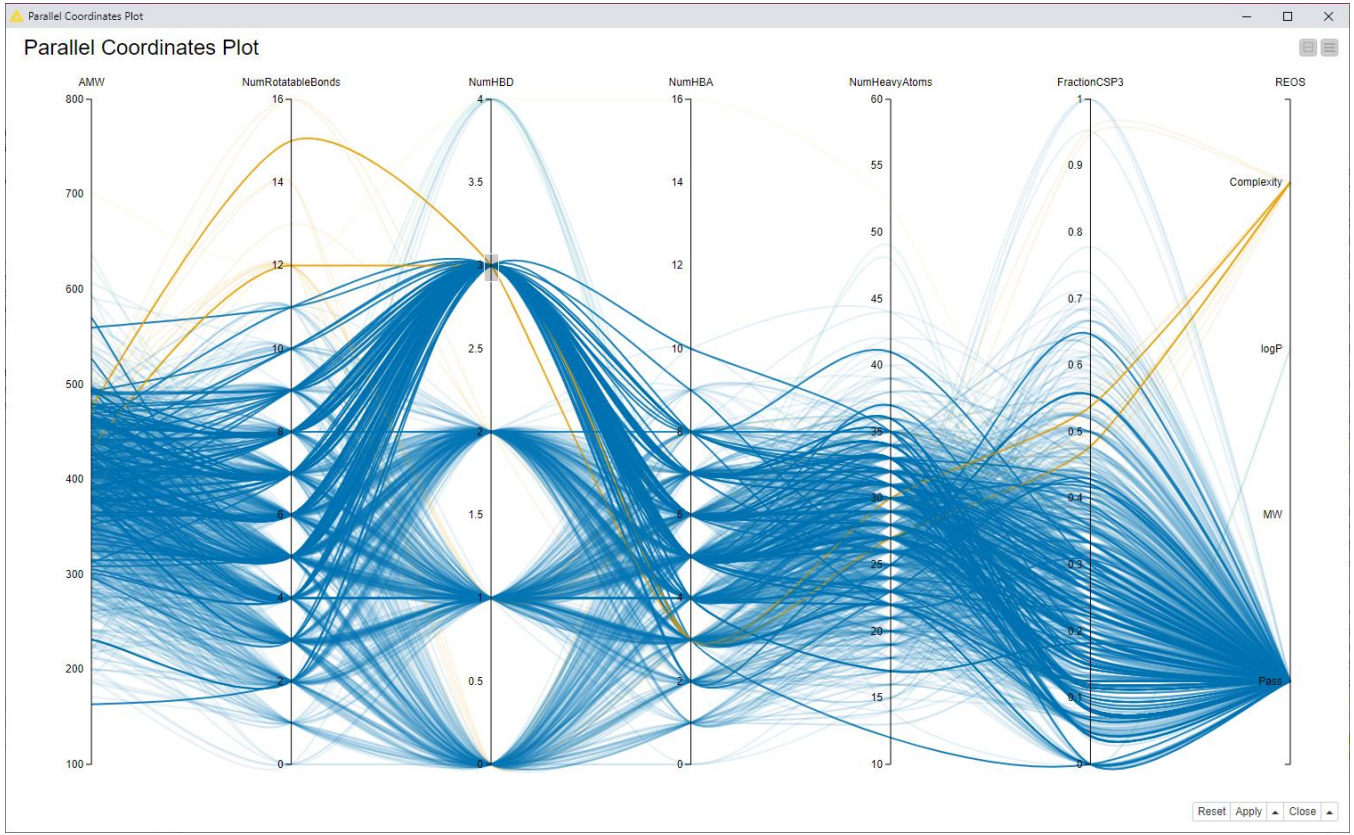
Column	Exclude Column	Minimum	Maximum	Mean	Standard Deviation	Variance	Skewness	Kurtosis	Overall Sum	No. zeros	No. missings	No. NaN	No. +∞	No. -∞	Histogram
AMW	☐	163.184	700.812	390.278	69.540	4835.749	-0.215	0.364	596344.599	0	0	0	0	0	
NumRotatableBonds	☐	0	16	5.228	2.242	5.027	0.558	0.926	7988	6	0	0	0	0	
NumHBD	☐	0	4	1.302	0.895	0.800	0.325	-0.386	1990	285	0	0	0	0	
NumHBA	☐	0	16	4.694	1.721	2.963	0.198	0.880	7172	3	0	0	0	0	
NumHeavyAtoms	☐	12	52	27.425	4.977	24.772	0.021	0.549	41905	0	0	0	0	0	
FractionCSP3	☐	0	1	0.261	0.148	0.022	0.837	1.786	398.142	45	0	0	0	0	

Showing 1 to 6 of 6 entries

Reset | Apply | Close

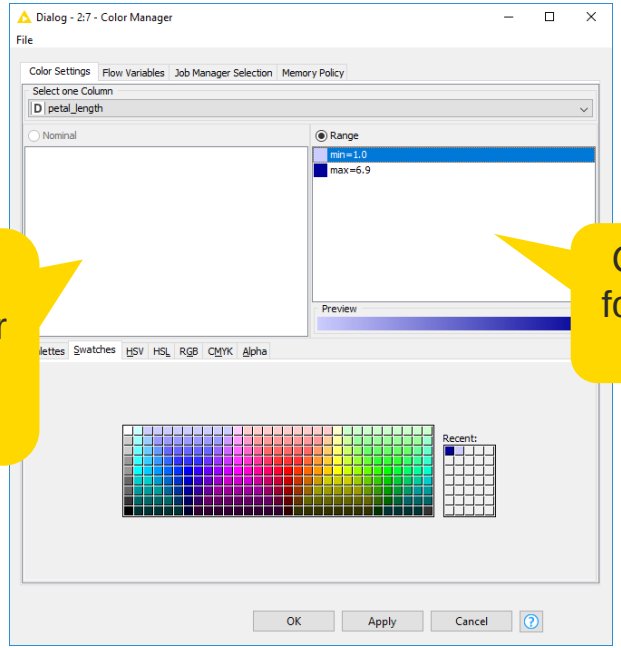
Parallel Coordinates Plot





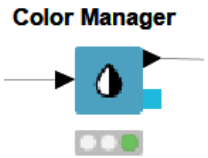
Color Manager

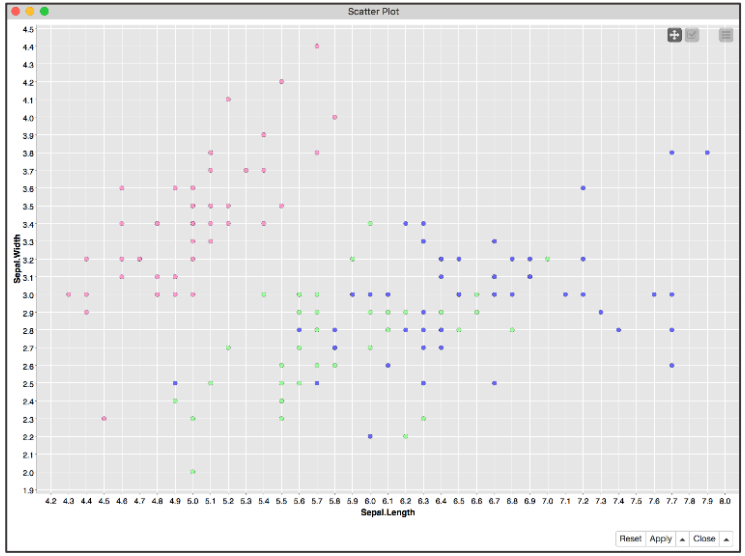
- Color by nominal or continuous values
- Sync colors between views using the color model port and Color Appender node



Discrete colors for nominal values

Color range for numerical values

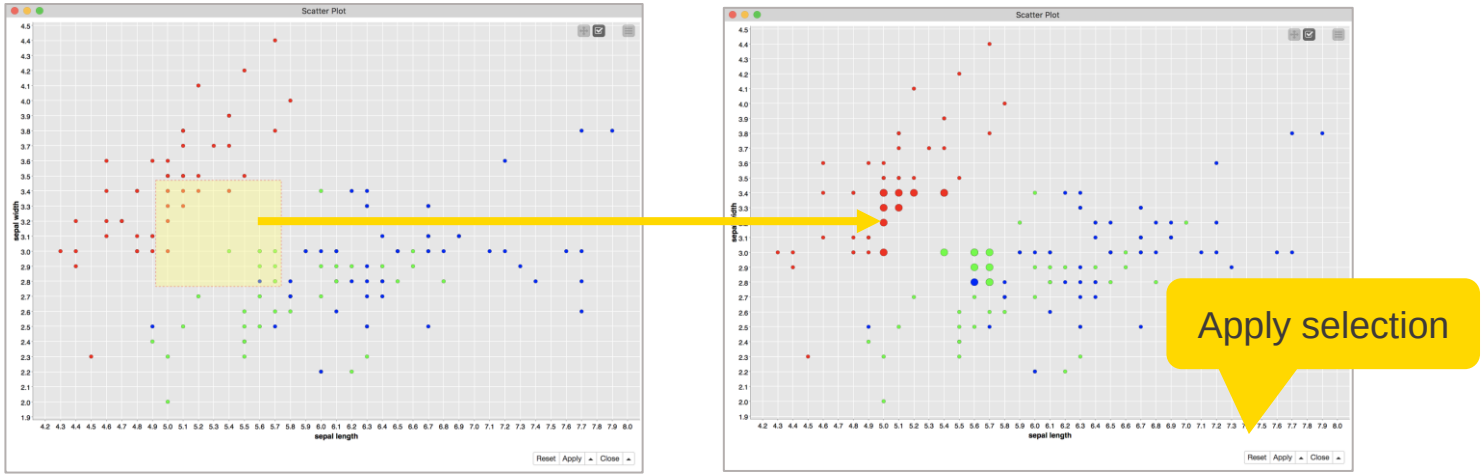




Selection & Filtering in JavaScript Views

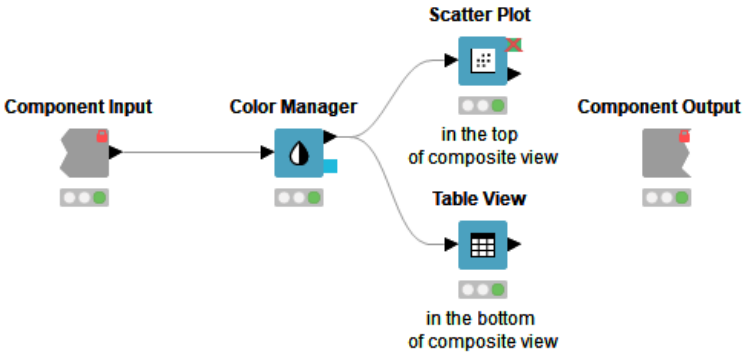
Interactivity allows you to select data points in views

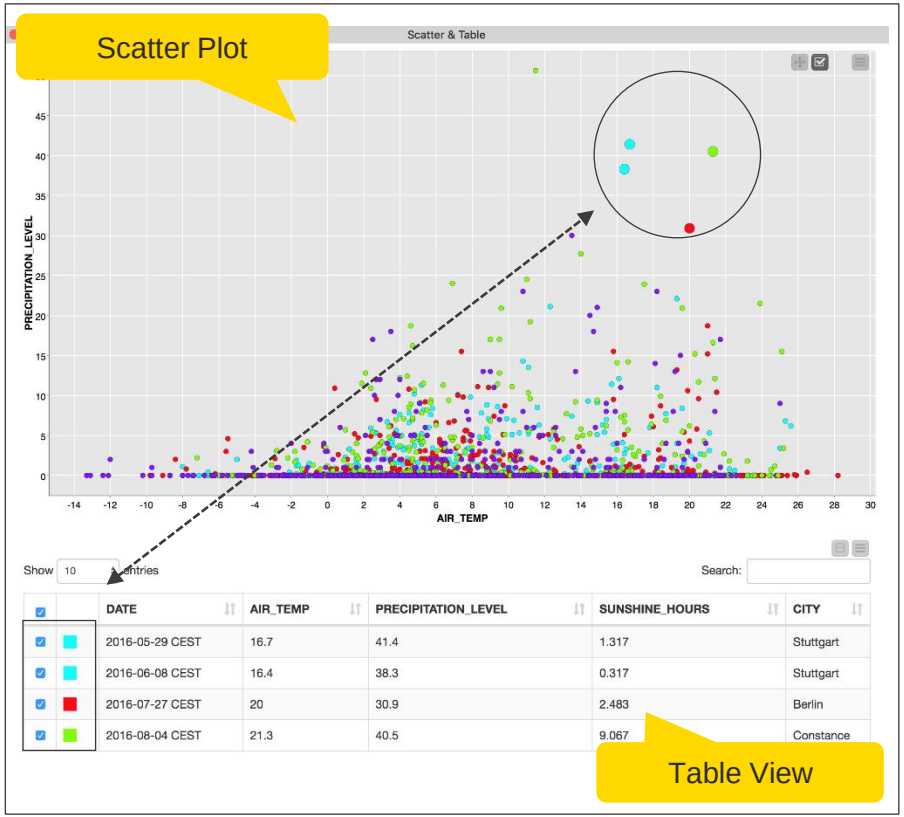
- Selection is propagated to other views.
- Highlight selected rows or filter them
- Click “Apply” to add column to data that indicates selection (true/false) for use in downstream nodes



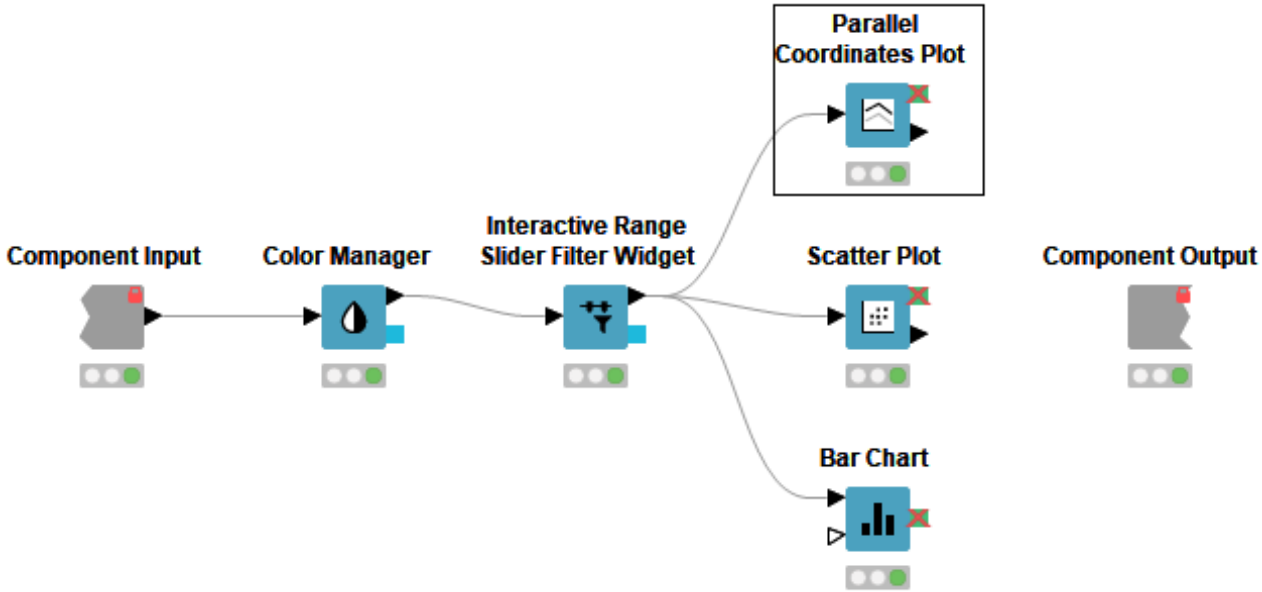
Components – Combined Views

- Multiple JavaScript View nodes can be combined in Components
- Selections are transmitted to all other views
- Also for use on the KNIME WebPortal

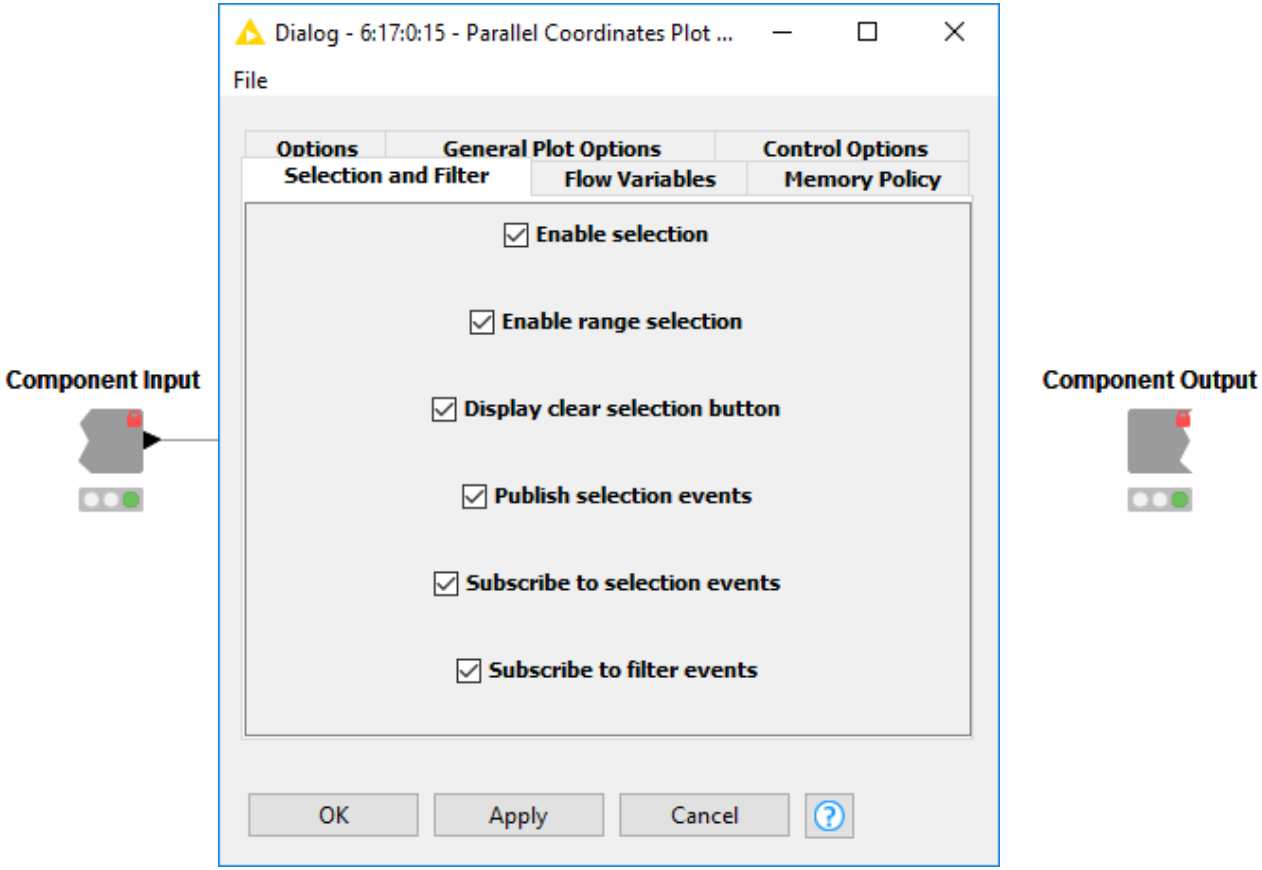




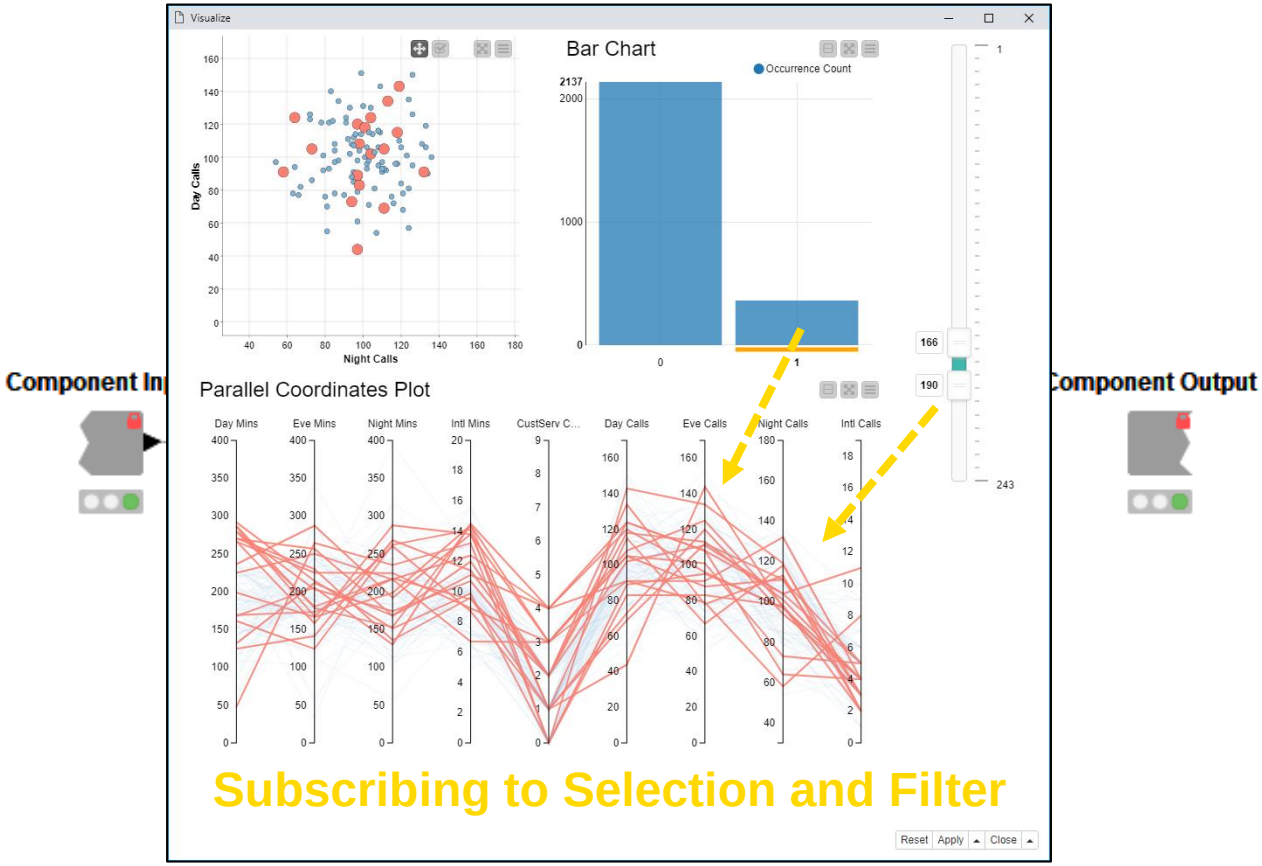
Interactivity across Charts: Selection and Filter Events



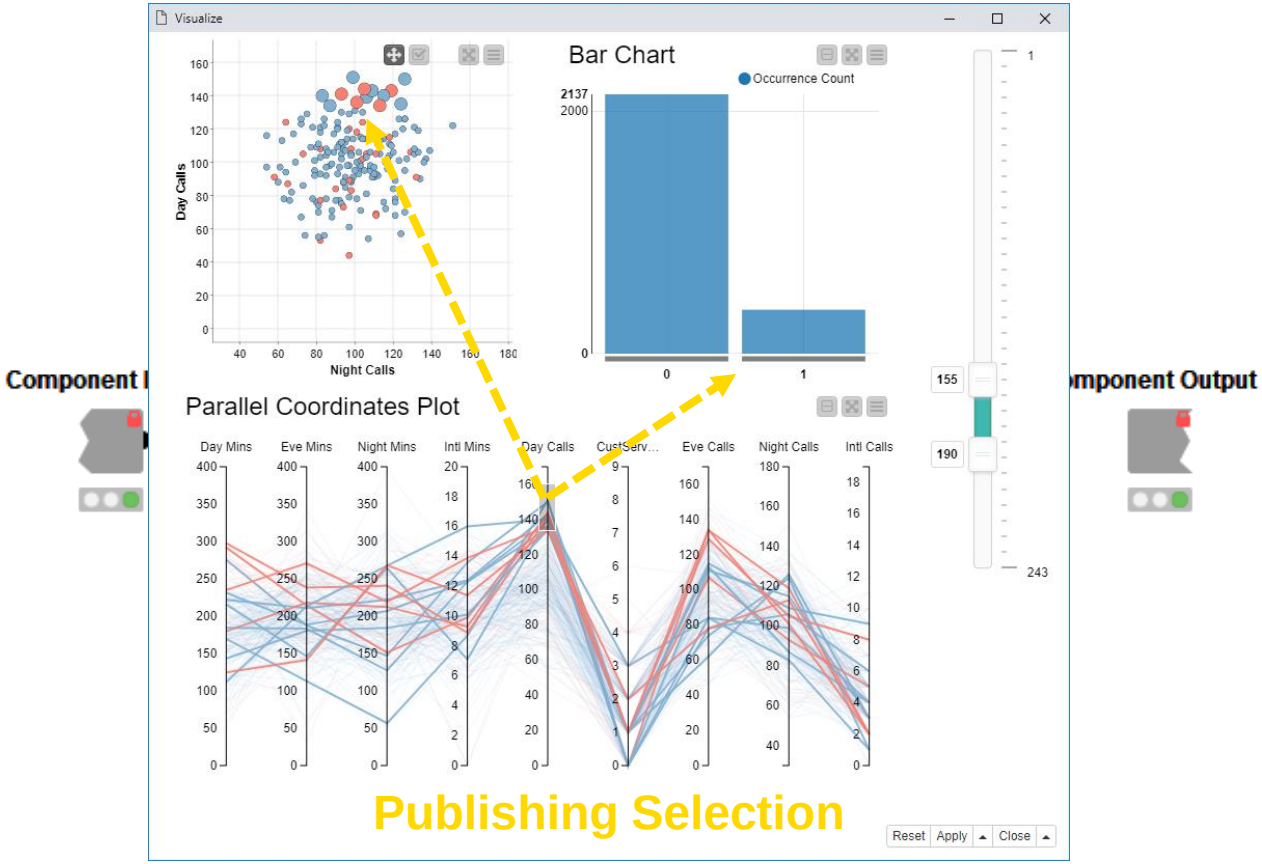
Interactivity across Charts: Selection and Filter Events



Interactivity across Charts: Selection and Filter Events

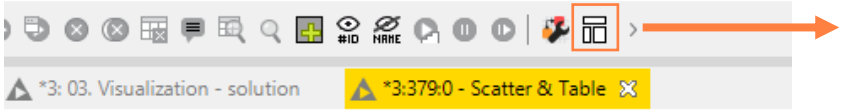


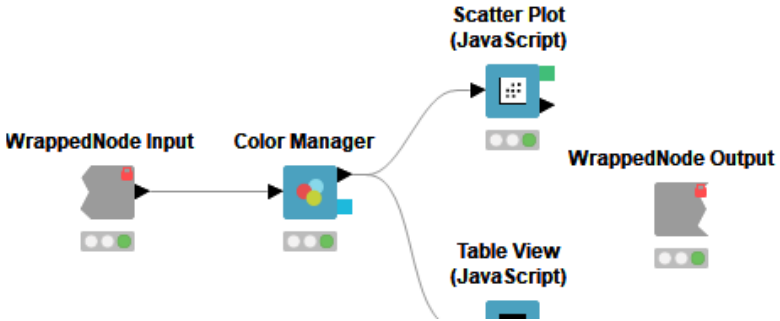
Interactivity across Charts: Selection and Filter Events

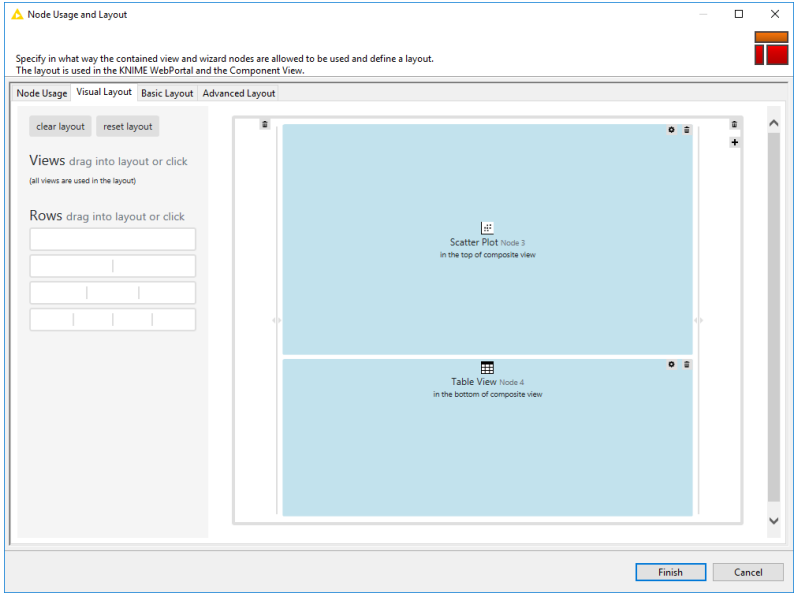


Configure Content and Views Layout

- Click layout button when inside Component to assign views to rows and columns
  - Add views and rows via *drag&drop*
  - Add columns using **+** buttons

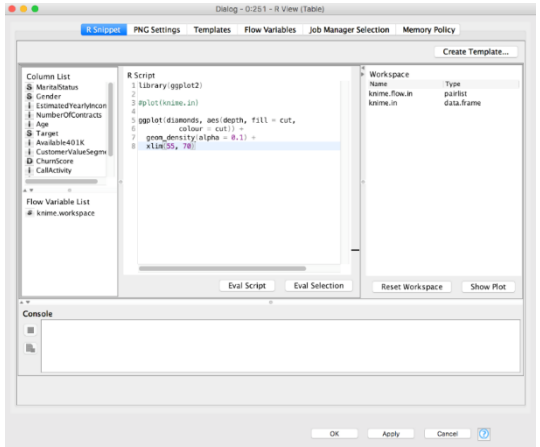


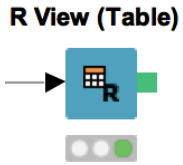


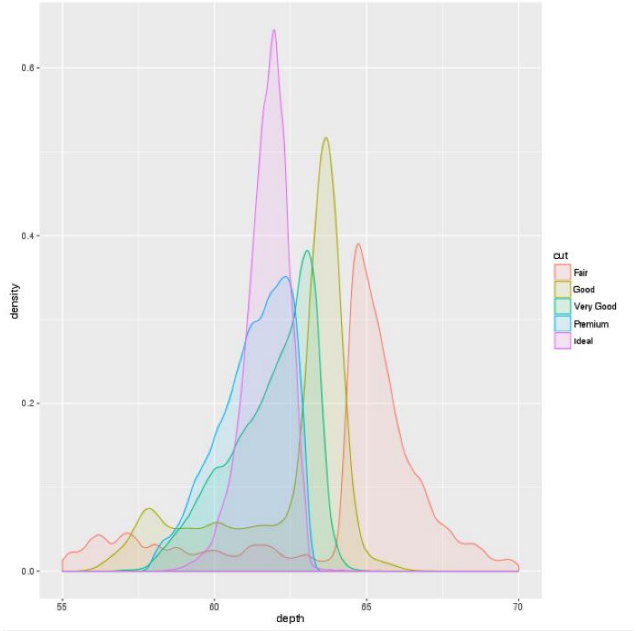


Script-based View Nodes

- R View nodes for greater customizability
  - Use your favorite libraries, e.g. ggplot2
- If you prefer Python: Python View node
- For JS developers: Generic JavaScript View



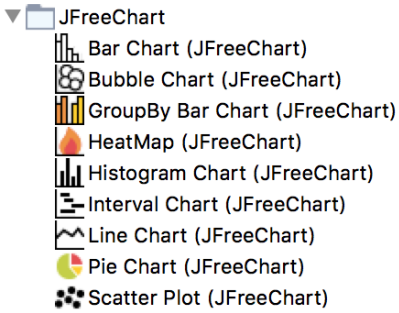




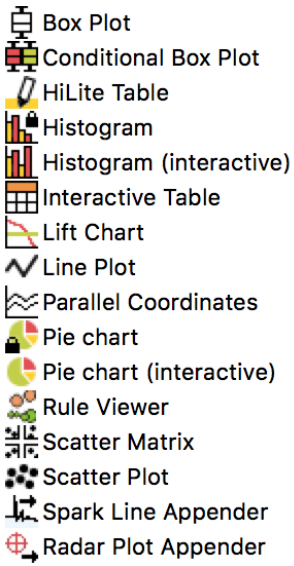
Legacy View Nodes: JFreeChart & KNIME Views

- KNIME provides three types of visualizations
  - **JavaScript Views**
  - JFreeChart Views
  - Local Views

- Active development only for JavaScript Views -> use those!
- JFreeChart and Local Views still useful when visualizing locally



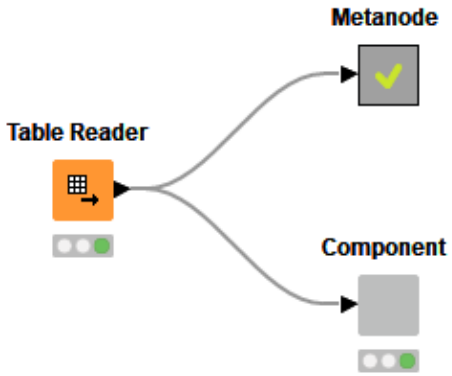
JFreeChart Views



Local Views

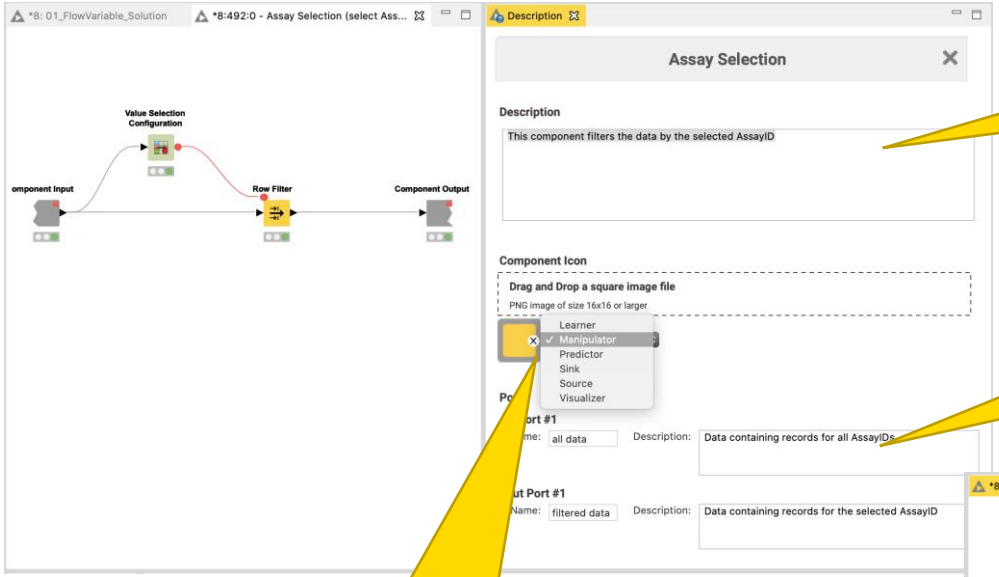
Components

- Components encapsulate functionality for reuse and sharing
- Components main features:
  - Local Flow Variable scope
  - Configurable via Configuration nodes
- Components are the key to advanced functionality in KNIME products:
  - Components corresponds to a KNIME WebPortal page
  - Configurations on a WebPortal page are defined using Widget nodes
  - Possibility to be shared via KNIME Hub



Component Description

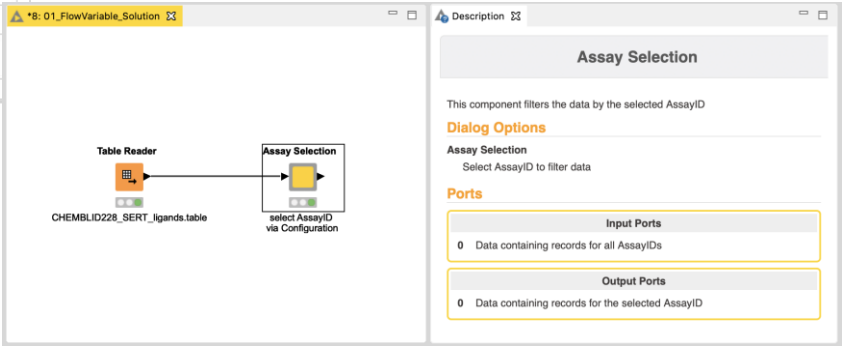
- Make your component look like a KNIME node



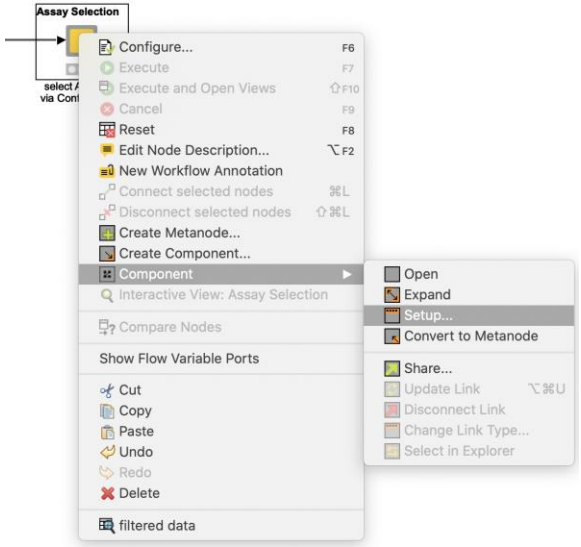
Add description of the component

Add description of the input and output ports

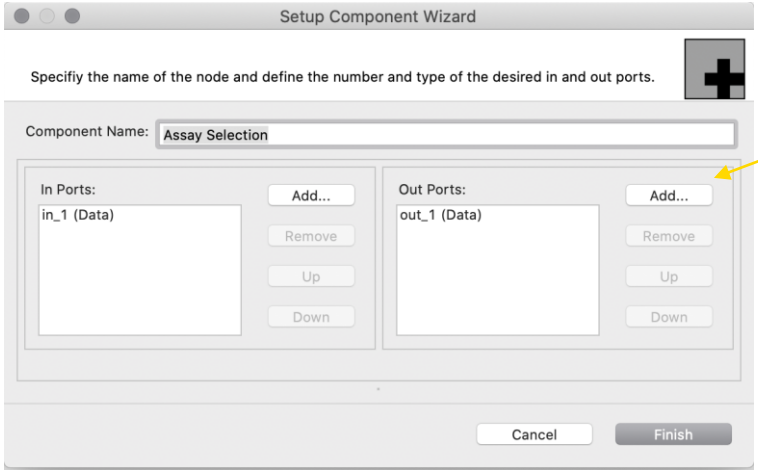
Add background color or icon

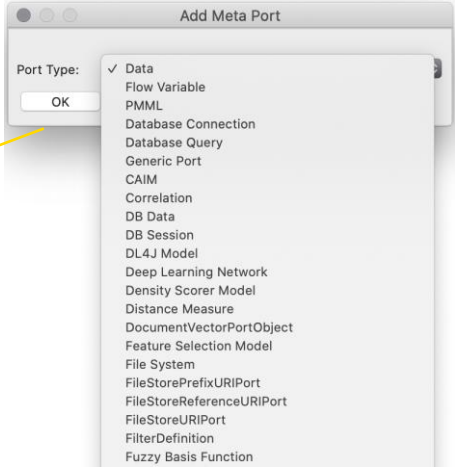


Configure Component Ports



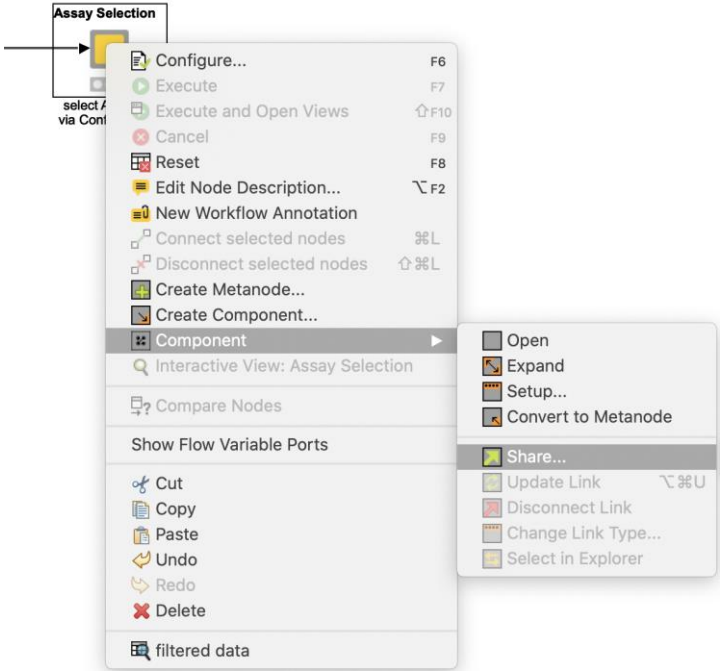
- Add input and output points to Metanodes/Components
- Remove ports to adapt to changes after creation of Metanodes/Components





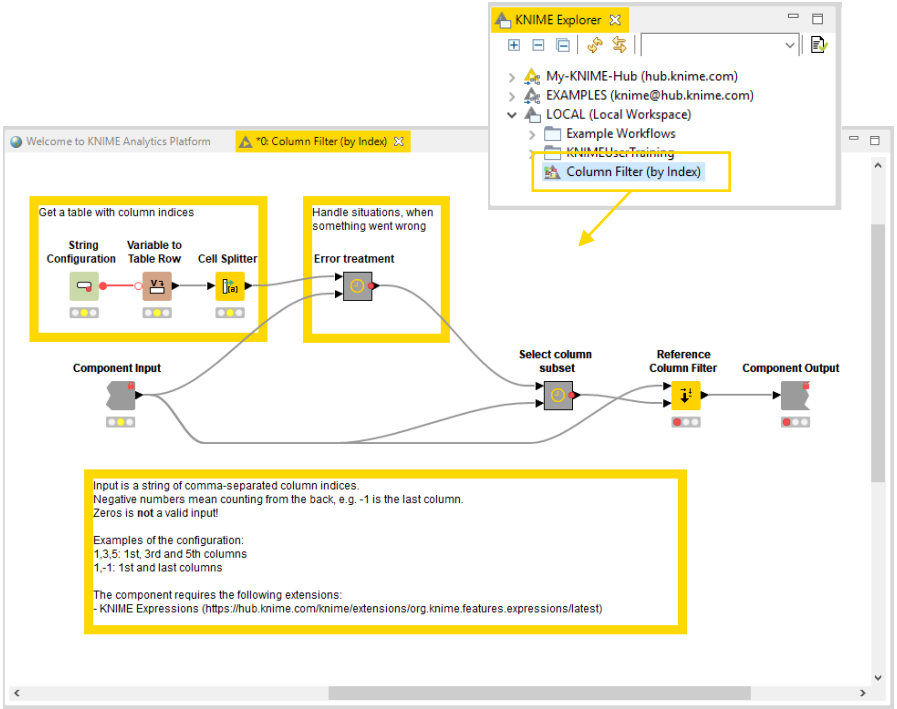
What is a shared component?

- Components can be saved in your KNIME workspace, KNIME Server or the KNIME Hub for later reuse
- To do this, simply right-click any Component and select “Share…”
- Shared Components are read-only instances of a Component
- Public Shared Components are available on EXAMPLES Server and on the KNIME Hub



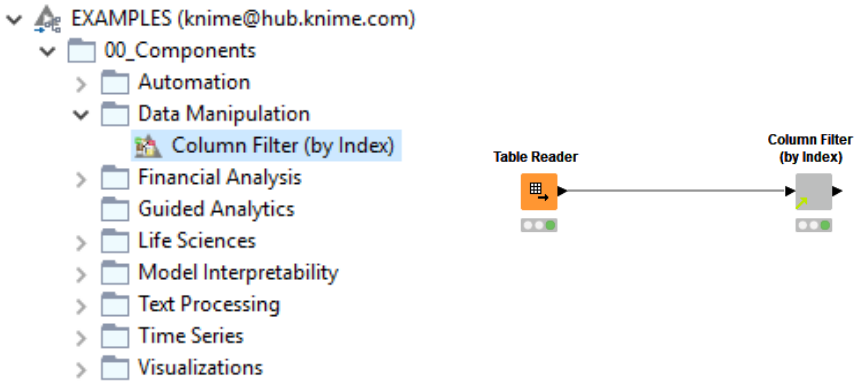
How can you edit a shared component?

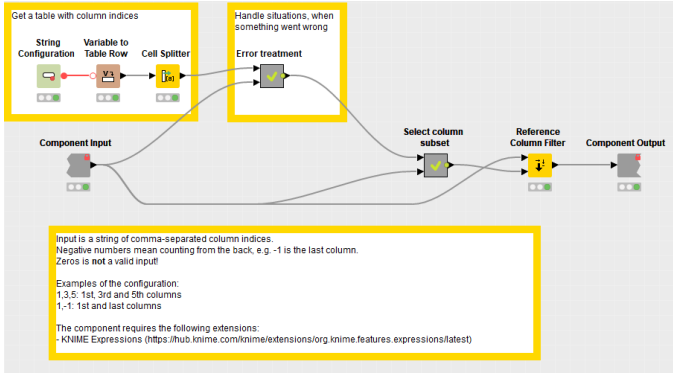
- Components can be edited using the Component Editor similar to workflows
- To edit a Component using the Component Editor, double-click the Component in its location in the KNIME Explorer
- To ensure Components are executable when opened in the Component Editor, chose the option “Include input data with component” when sharing it



How can you use a shared component?

- To use a Shared Component, drag and drop it to the workflow editor
- Instances of Shared Components can be updated either manually or when workflow is opened
- Shared Components can also be unlinked from its original location, which makes it editable in the workflow directly
- Update Shared Components by overwriting them

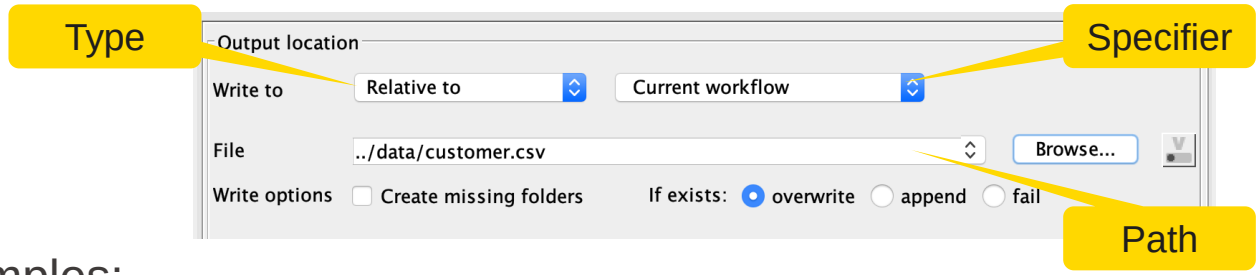




Flow Variables and Loops

Common Settings: File Path

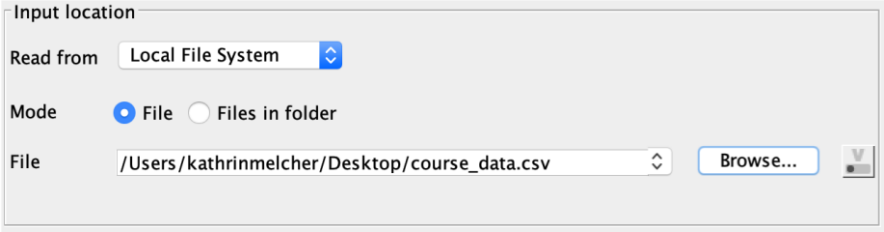
- A path consists of three parts:
  - **Type**: Specifies the file system type e.g. local, relative, mountpoint, custome_url or connected.
  - **Specifier**: Optional string with additional file system specific information e.g. relative to which location (knime.workflow)
  - **Path**: Specifies the location within the file system



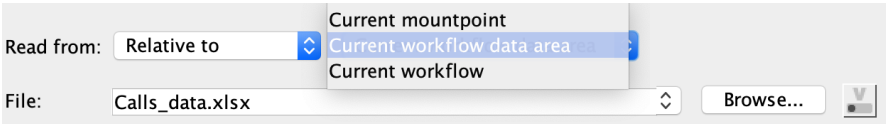
- Examples:
  - (LOCAL, , C:\Users\username\Desktop)
  - (RELATIVE, knime.workflow, file1.csv)
  - (MOUNTPOINT, MOUNTPOINT_NAME, /path/to/file1.csv)
  - (CONNECTED, amazon-s3:eu-west-1, /mybucket/file1.csv)

Common Settings: File Path Options

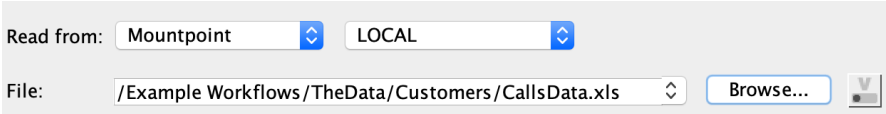
- ## Local File System



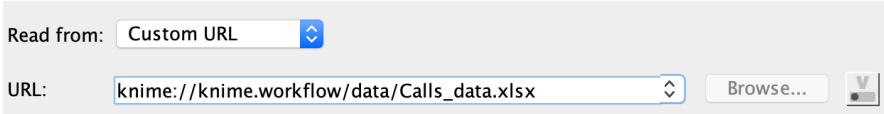
- ## Relative to …



- ## Mountpoint

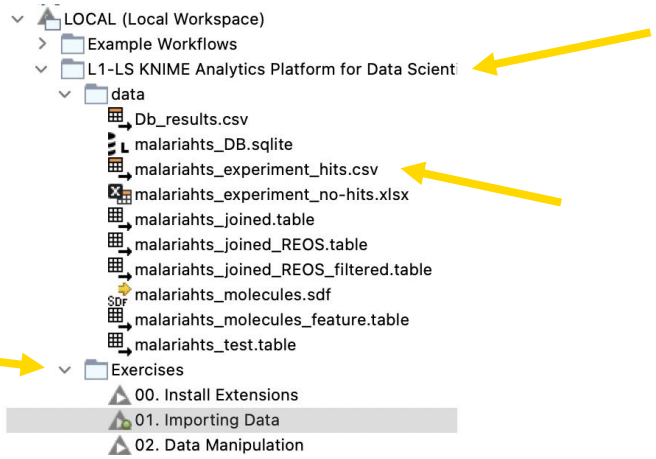


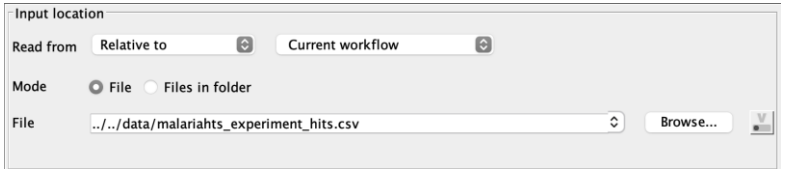
- ## Custom URL

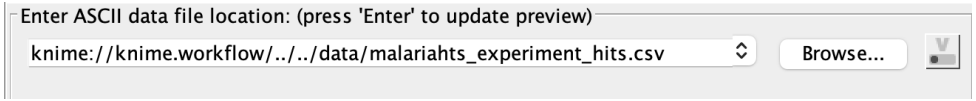


Workflow-Relative File Paths

- Best choice if workflows are to be shared
- Requires matching folder structure within workflow group
  - Independent of environment outside of workflow group
- Example: Path to „Sentiment Analysis.table“
  - Local path:
  - C:\Users\rb\knime-workspace\KNIMEUserTraining\data\malariahts_experiments_hits.csv
    - Workflow relative:



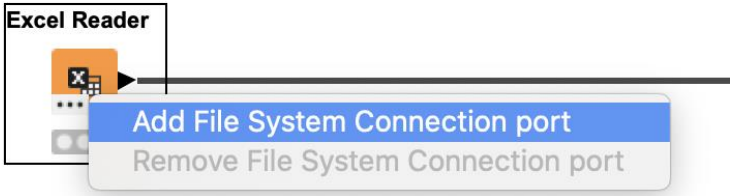




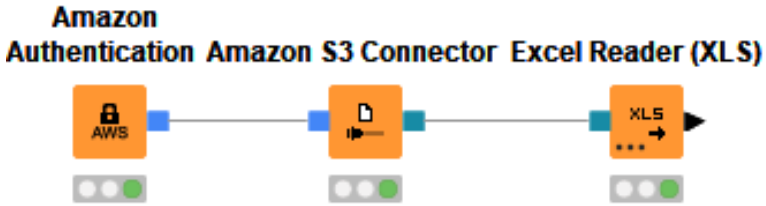
YouTube KNIME TV Channel:
https://youtu.be/IiZsOnhZgzk

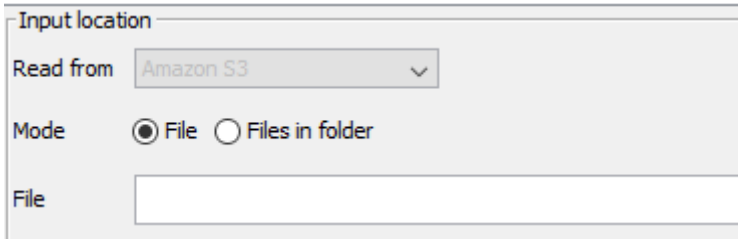
Common Settings: Connecting to other File Systems

- Add file system connection port to connect to another file system
  - Click on the three dots on the lower left to add or remove a dynamic port.



- Supported file systems
  - Microsoft Azure
  - Google
  - Amazon
  - Databricks
  - BigData file systems (hdfs, httpFS, …)
  - On-premise (e.g. ssh, ftp, …)

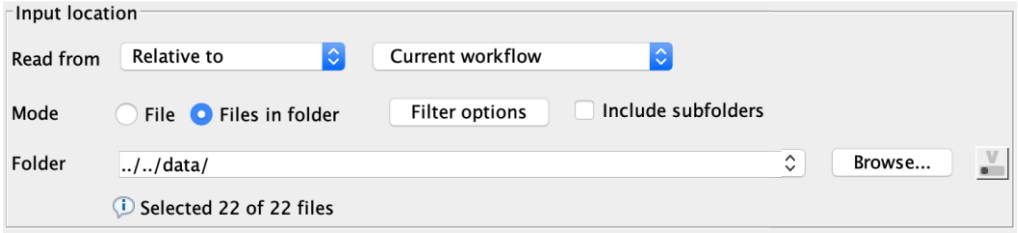




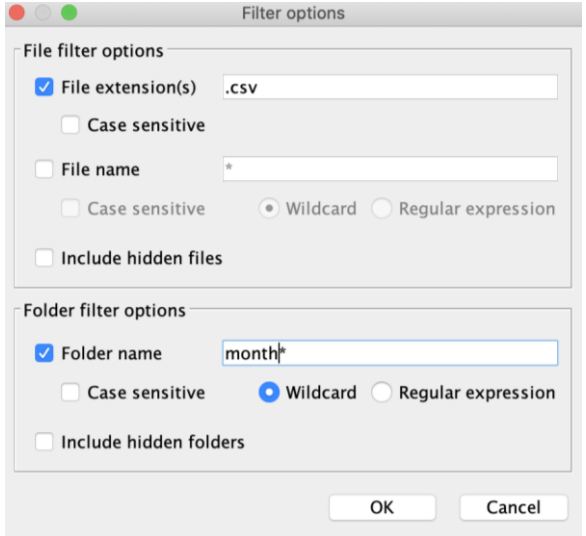
Common Settings: Read Single or Multiple Files

- Single file

- Files in a folder

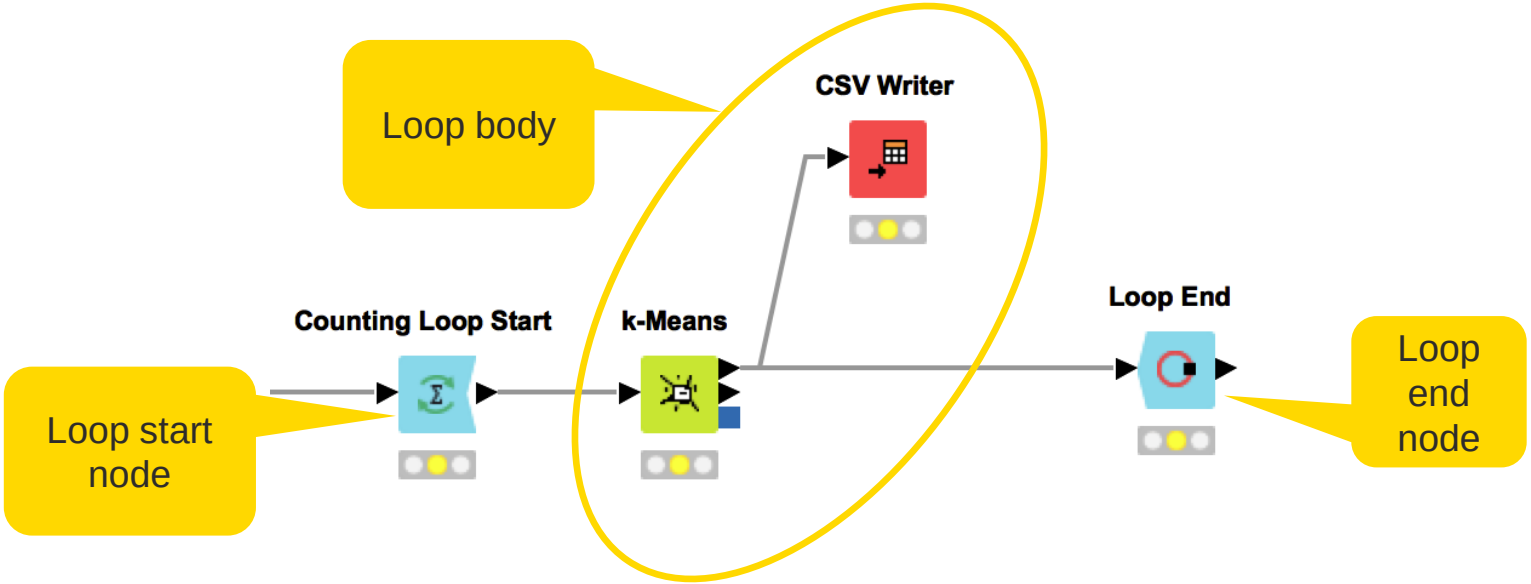


  - Option to include subfolder
  - Option to define filter criterions



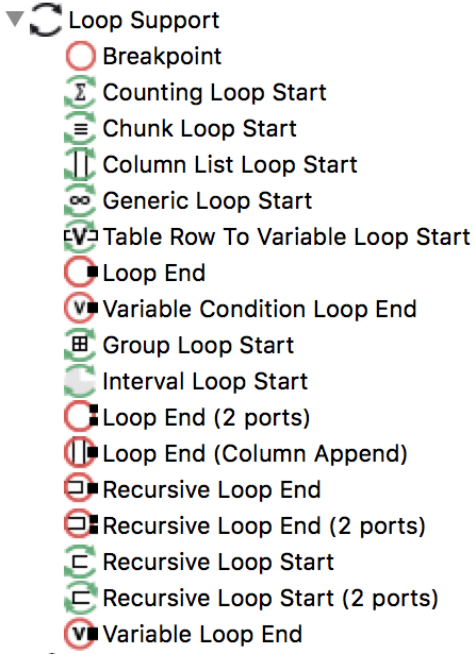
The Loop Block

- A loop block is defined by appropriate loop start and loop end nodes.
- Loop body = nodes in between (including side branches).



Loop Start and End Nodes

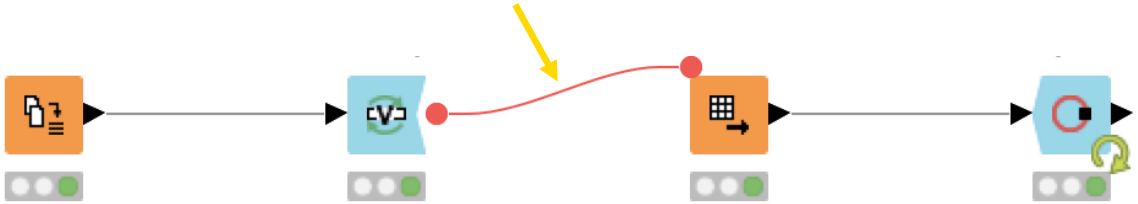
- For different tasks, there are different loop start and end nodes
- Nodes with circular arrows (green) = Start node
- Nodes with a closed circle (red) = End node
- Flow Variables are really helpful to build the loop body



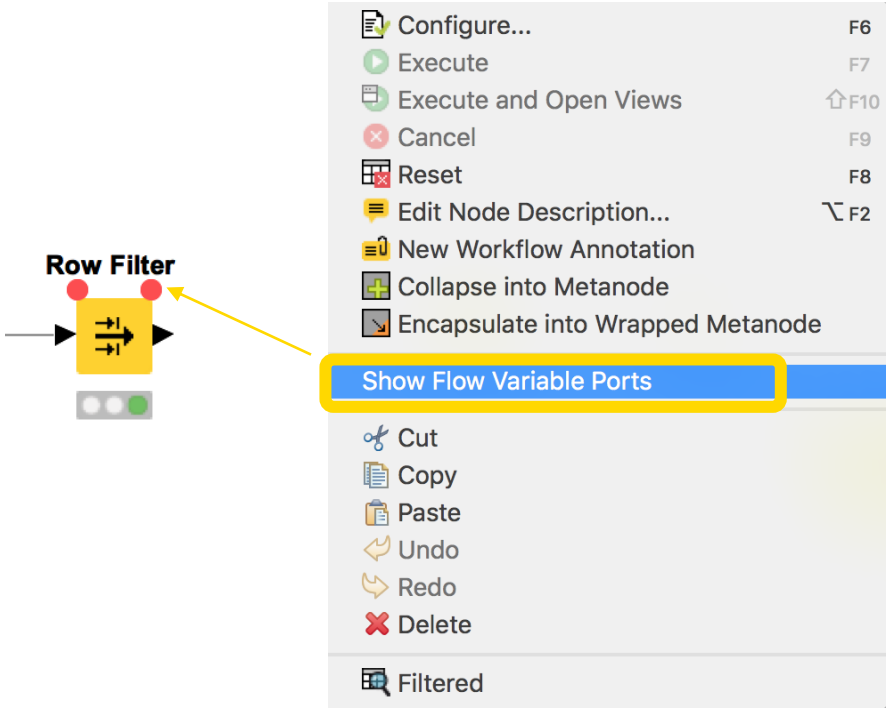
Flow variables

- Flow Variables are workflow parameters that are to overwrite existing node settings

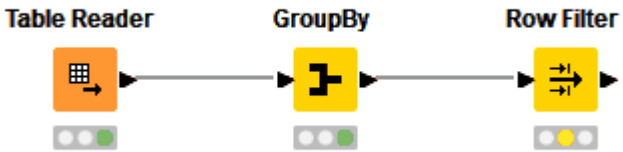
- Especially in loops they can hold different values, depending on the iteration



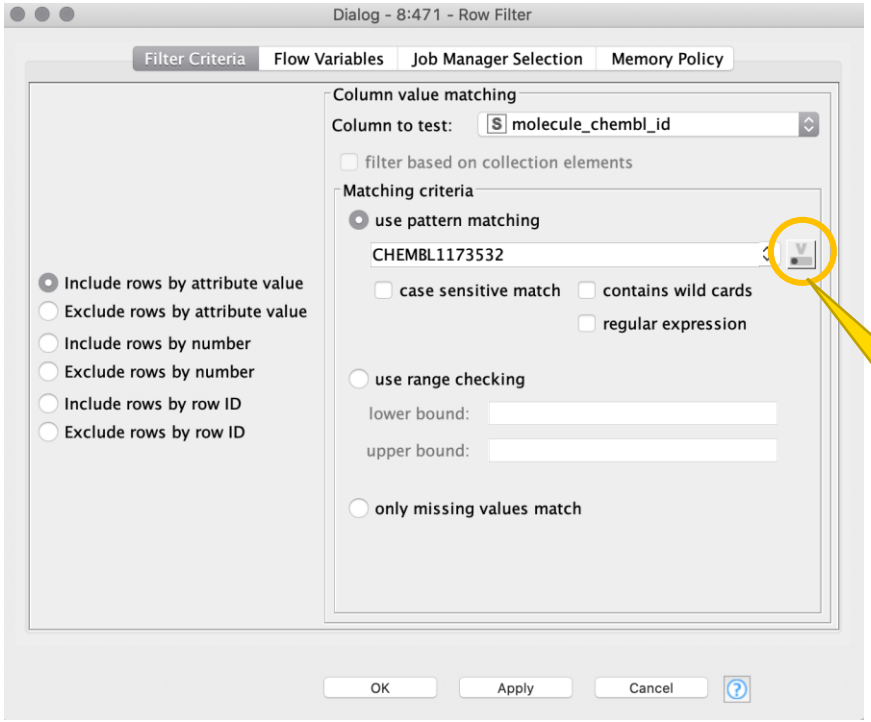
Flow Variable – Ports and connections

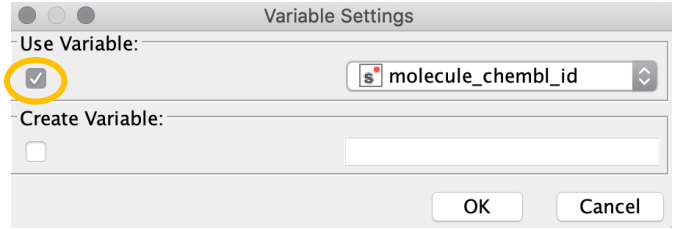


… or just drag and drop from one upper corner of a node to another



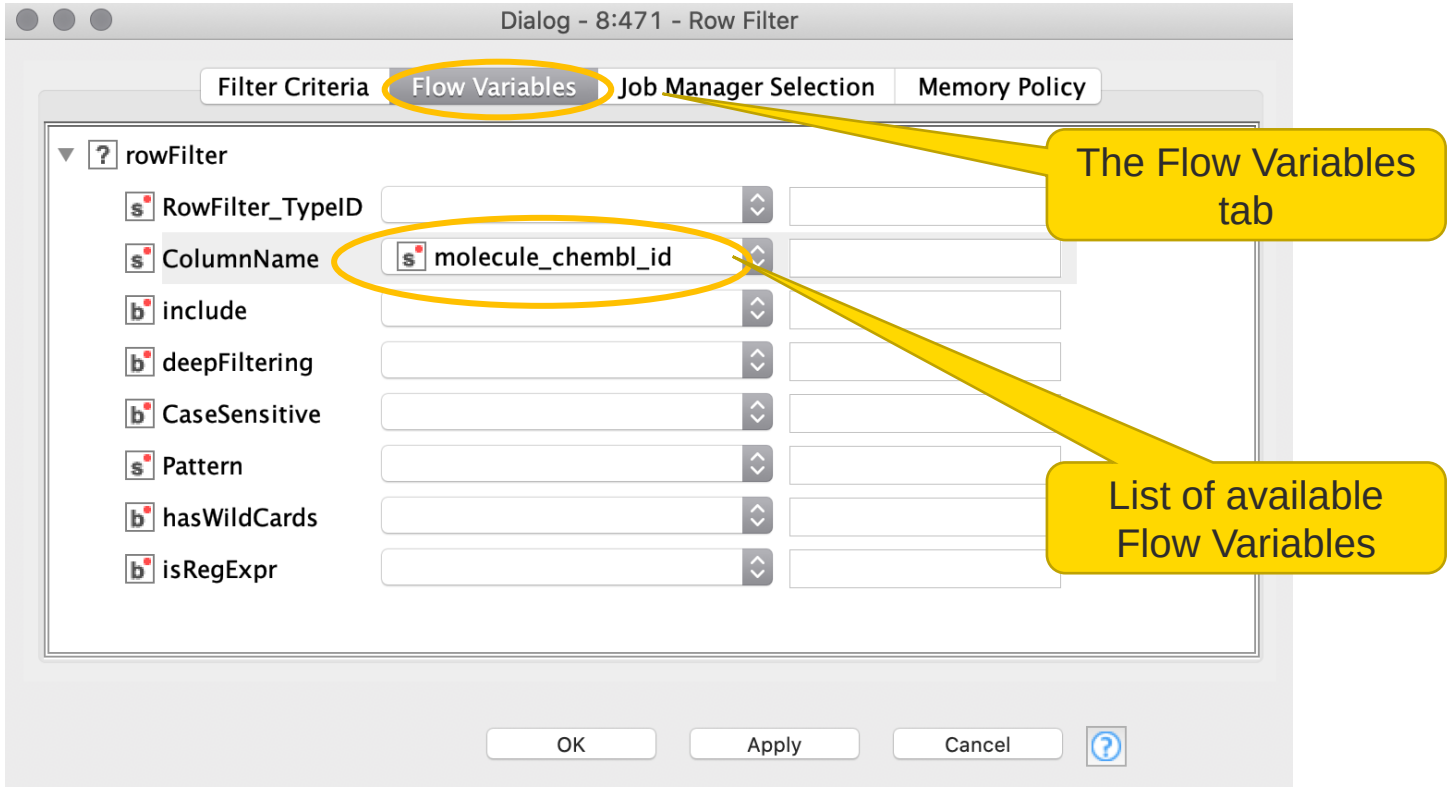
Apply a Flow Variable (Button)



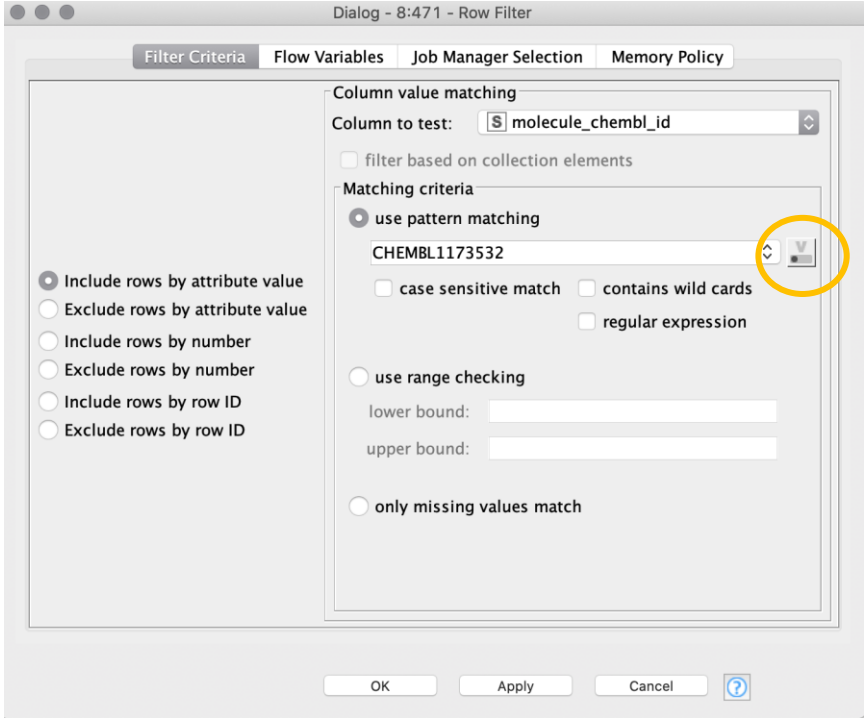


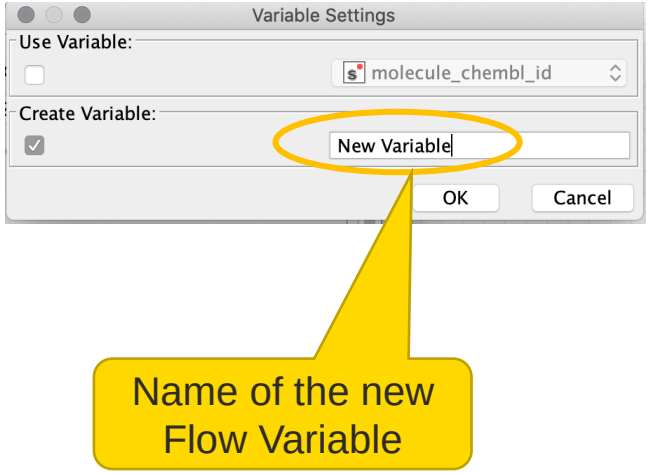
The Flow Variable button

Apply a Flow Variable (Advanced)



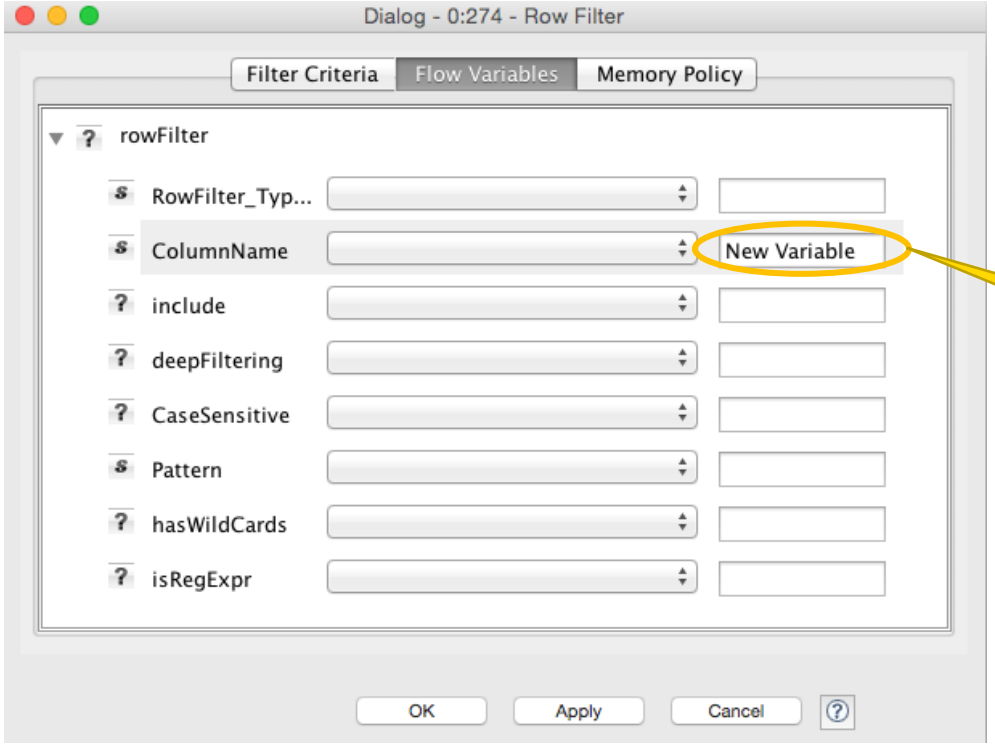
Create a Flow Variable (Button)





Create a Flow Variable (Advanced)

- Converting a setting value into a Flow Variable



Name of the new Flow Variable

Key Features: Flow Variables

- Flow Variables are workflow parameters used to overwrite existing node settings
- A Flow Variable is carried along workflow branches (parallel branches don’t share local Flow Variables)
- Flow Variables can be of type String, Integer, Double, Boolean, Long and Array
- Flow Variables can be created
  1. Using the ”Table Row to Variable” node
  2. In the “Flow Variable” tab of any node
  3. Using Configuration and Widget nodes

Example 1: Reading many Excel sheets from one file

- List all sheet names of an Excel file
- Convert sheet name into a flow variable (1 sheet name per iteration)
- In each iteration, read the spreadsheet with the current sheet name
- Close the loop and collect the results

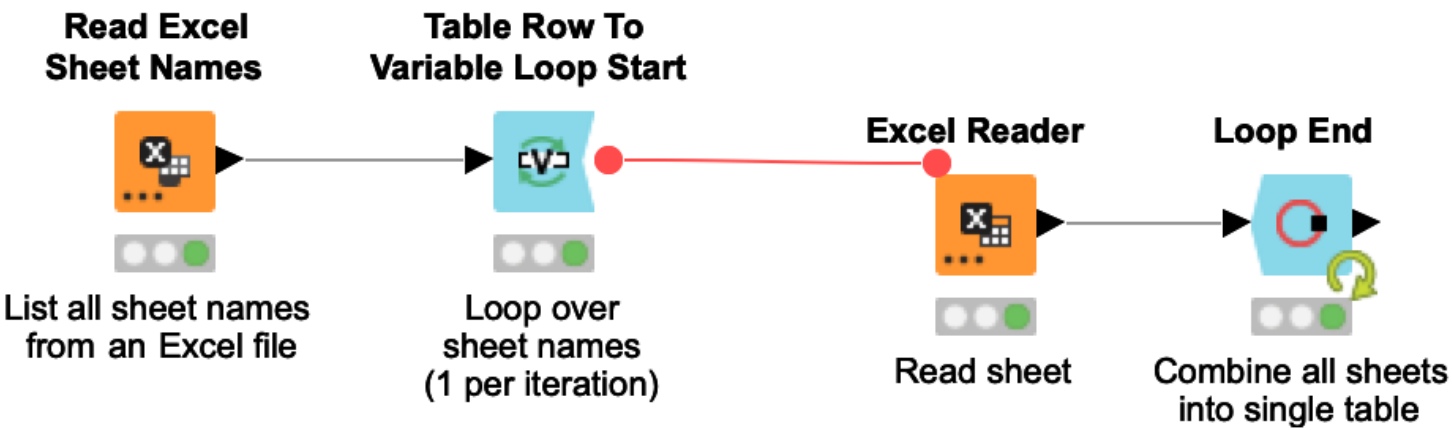
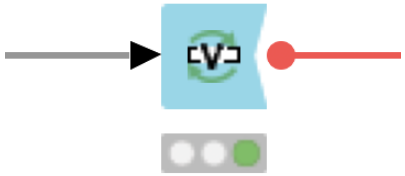
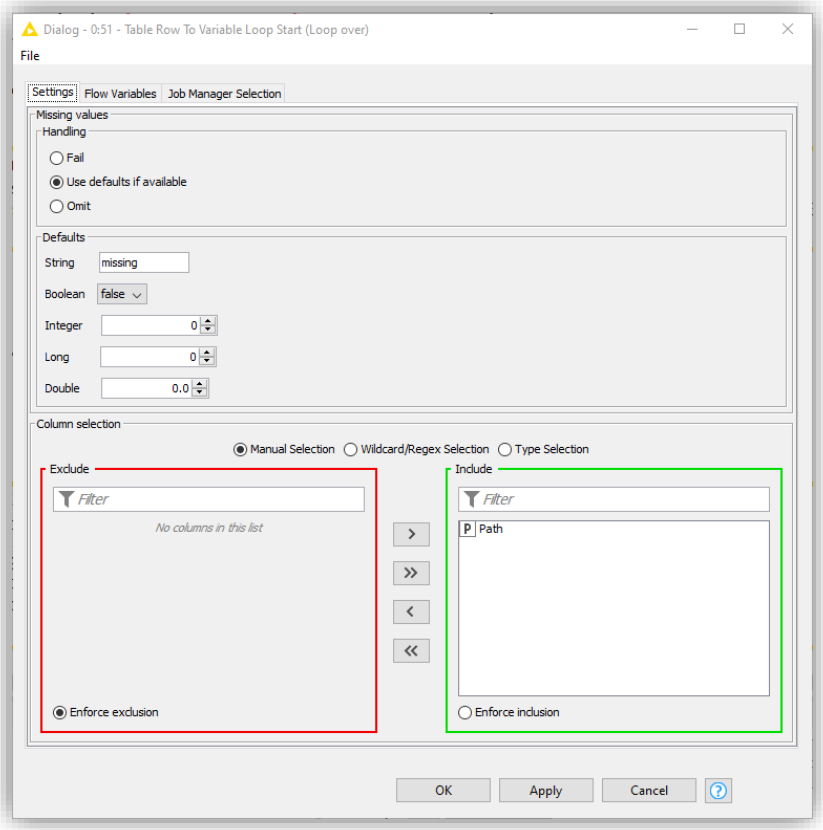


Table Row to Variable Loop Start

- Similar to the
  Table Row to Variable node
- Each iteration of the loop converts the next row of the input table into Flow Variables
- Injects variables into other nodes to re-execute subflows with a progression of settings

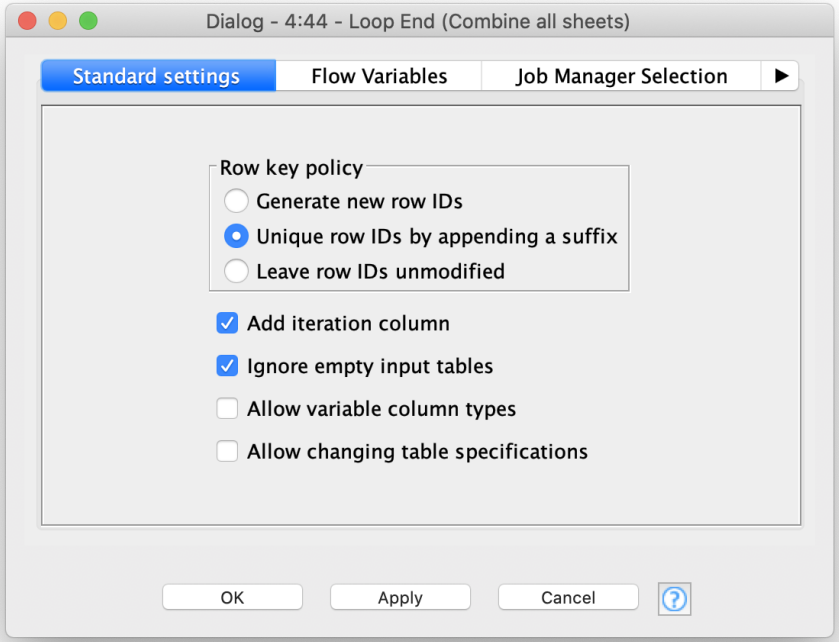
Table Row To Variable Loop Start

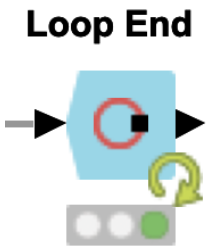




Loop End

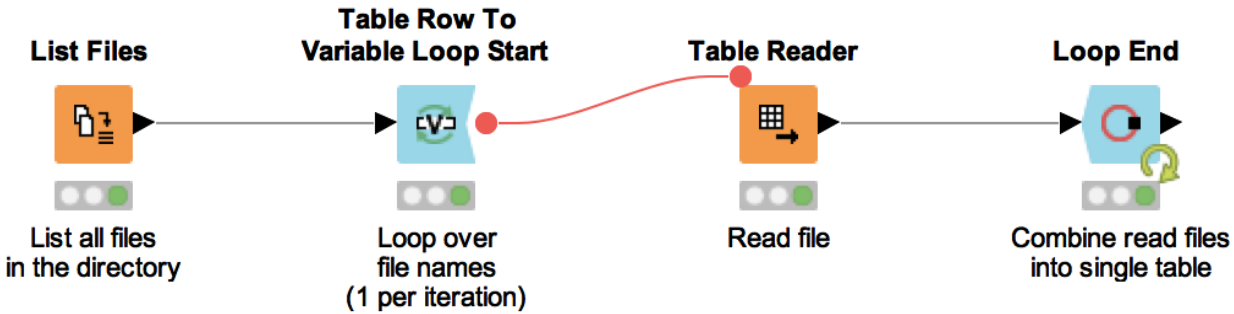
- Can be used to end of a loop
- Collects the results of the different iterations by row-wise concatenation of the incoming tables
- Provides options to:
  - Add a column with the iteration number
  - Allow variable column types
  - Allow changing table specifications



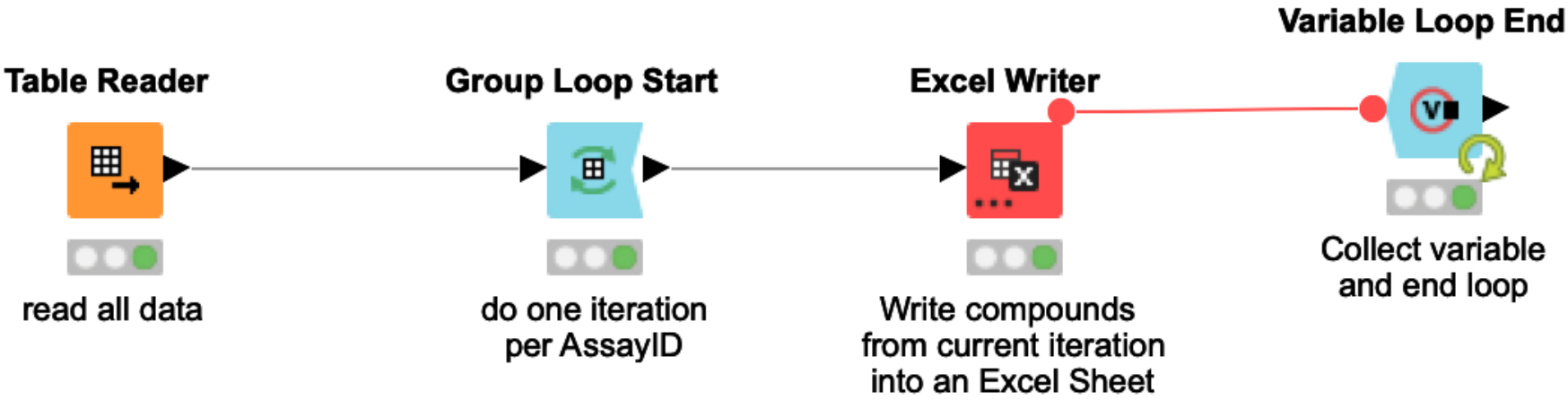


Similar: Reading Many Files

- List all files in a directory
- Convert each file path to a Flow Variable (1 per iteration)
- In each iteration, read the file and collect the results

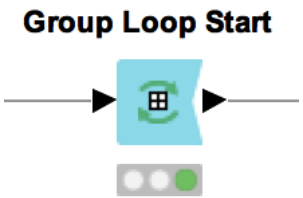


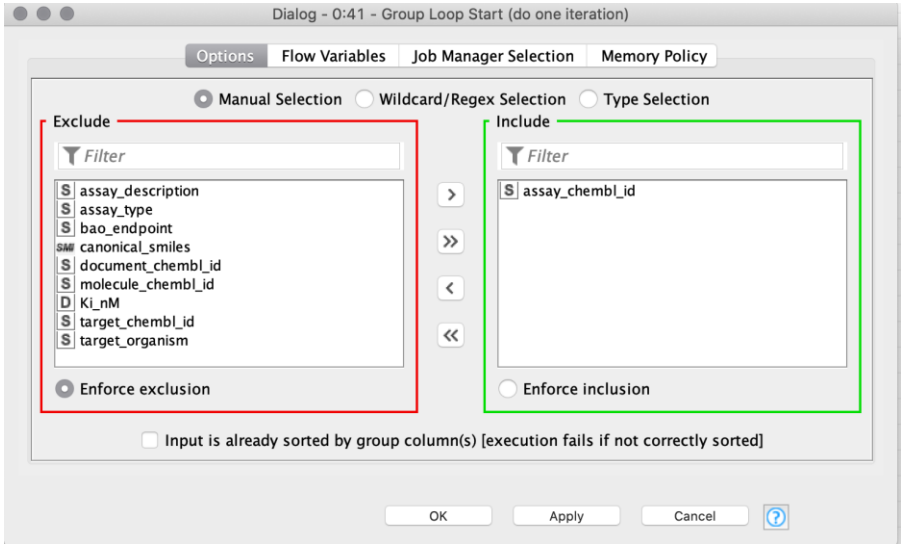
Example 2: Writing to multiple Excel Sheets in same file



Group Loop Start

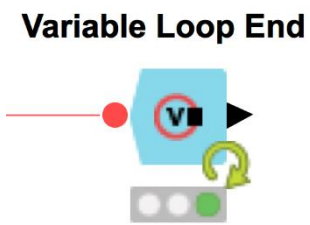
- Similar to GroupBy except without aggregation tab
- Each iteration of the loop passes the next group of rows
- You implement the aggregation task. It can be anything from a complex calculation to updating a database

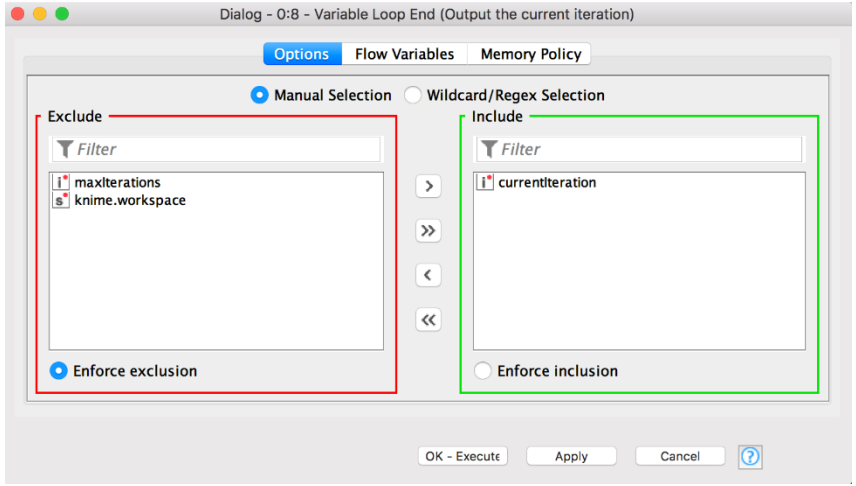




Variable Loop End

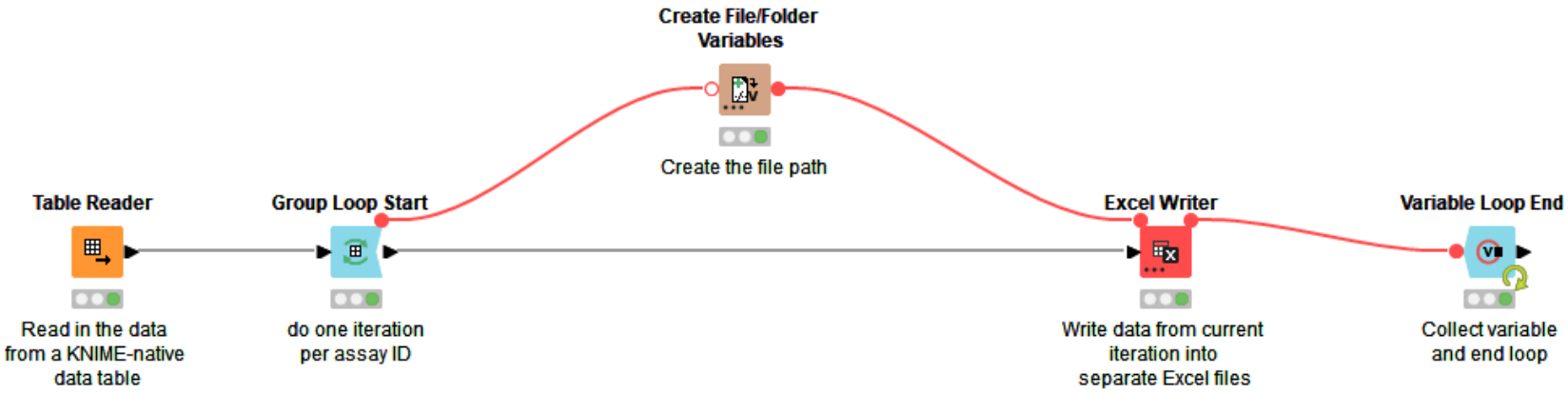
- Collects variables created in the loop
- Closes a loop without aggregating any data
  - Can be used to close loops where all the magic happens inside
  - e.g. in each iteration where the aggregated table is already written





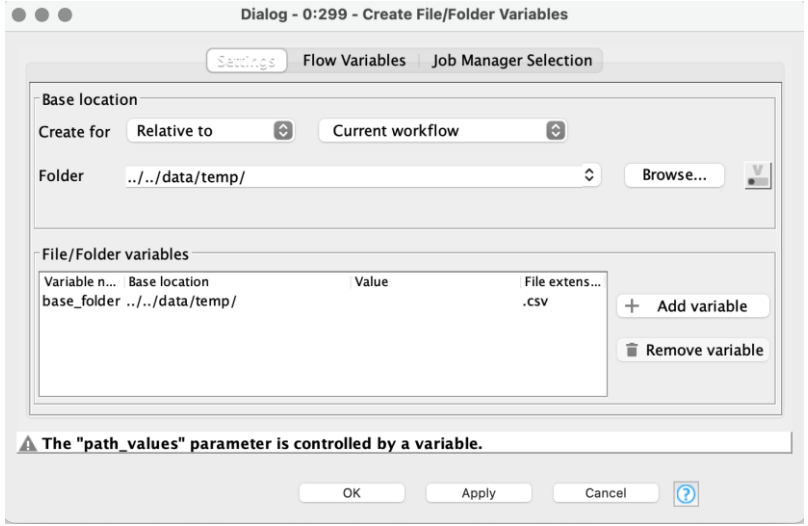
Example 3: Writing into separate files

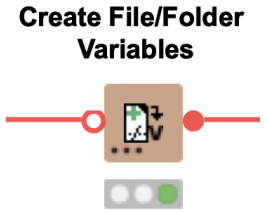
- Group Loop Start → Variable Loop End
- Group data by specific column values
- Iterate over all groups of data
- Create an appropriate path variable
- Write grouped data to Excel files with new file name



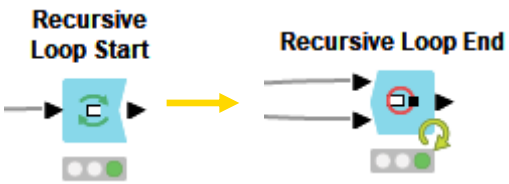
Create File/Folder Variables

- Creates one or multiple path flow variable(s) pointing to files / folders
- Inputs:
  - Base location
  - Flow variable name(s)
  - Value (file name or path relative to base location)
  - File extension (optional)
- Output variables can be used to control the output location in writer nodes.

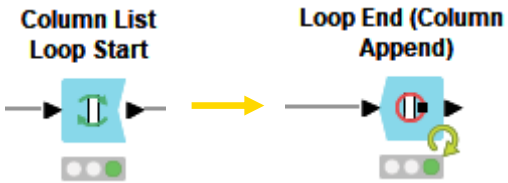




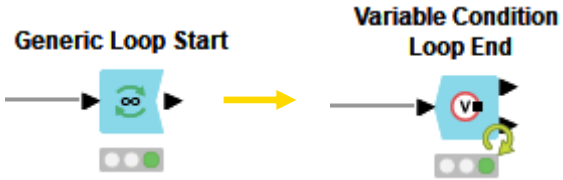
Loop node pairs that go often together



The Recursive Loop node pair enables the passing of data tables from the Recursive Loop End back to the Recursive Loop Start.



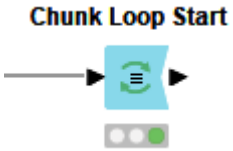
Takes one column at a time and joins the output (i.e. the column after modification) to the current input table.



Execute the body of the loop until a certain condition on one of the flow variables is met.

More loop start nodes

Starts a loop which is executed a predefined number of times



Each iteration processes another chunk of consecutive rows

Examples on the KNIME Hub

- Find more examples for loops on the [KNIME Hub]

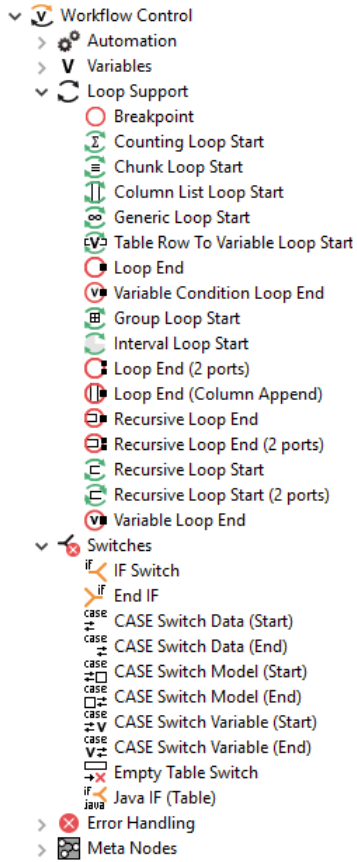
Home > 06_Control_Structures > 04_Loops

21_Parameter_optimization_loop
01_Loop_over_a_set_of_parameter_for_k_means
02_Example_for_Reading_a_List_of_Files
03_Looping_over_all_columns_and_manipulation_of_each
04_Looping_for_Multiple_Target_Prediction
05_Write_each_row_in_one_file
06_Writing_a_data_table_column_wise_to_multiple_csv_files
07_Example_for_Recursive_Looping
08_Example_for_Recursive_Replacement_of_Strings
09_Looping_over_Chunks_of_the_Data
10_Looping_a_fixed_number
11_Looping_over_Groups_of_the_Data
12_Using_TableRows_as_FlowVariables_in_Loop
13_Usage_of_Generic_Loop_Start
14_Looping_over_defined_Intervals
15_Collecting_Columns_in_Loop
16_Collecting_Variables_in_Loop
17_Usage_of_Breakpoint_in_Loops
18_Recover_from_Breakpoint_in_Loop
19_Forecasting_with_TimeDelay_Loop
20_Time_Series_Prediction_with_a_Recursive_Loop

Error handling

Workflow Control Structures

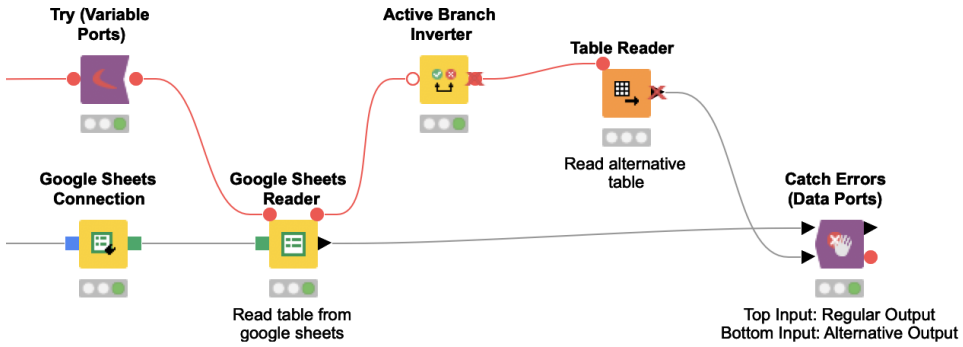
- **Try-Catch**
  - Handle workflow branches that may fail in execution and you don‘t know before execution
- **Active Branch Inverter**
- **Breakpoint**
- Switches
  - Direct the path of a workflow by selectively executing one or more workflow branches.



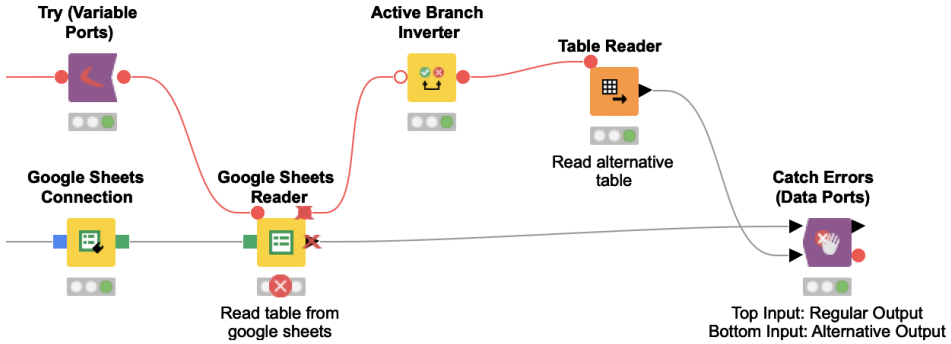
Try-Catch

- A way to catch errors in workflows
- Useful when it is hard to know if a node will execute (for example,
  when reading from a Google Sheet)
- KNIME tries to execute the nodes, but if it fails will fall back to
  an alternative branch

Regular Execution

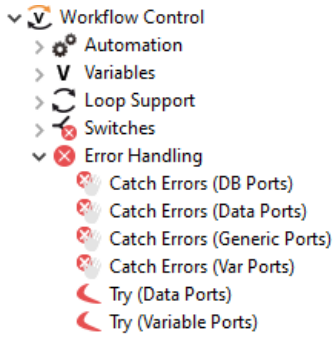


Alternative Execution



Try-Catch

- Try-Catch nodes available for different data types

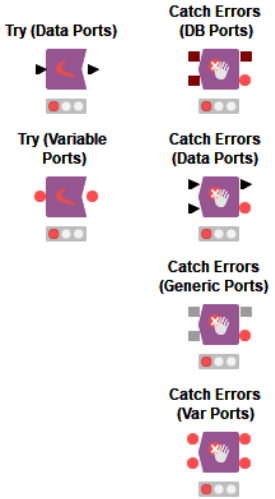


The original input

The input to be used when execution on the main branch failed; alternative input

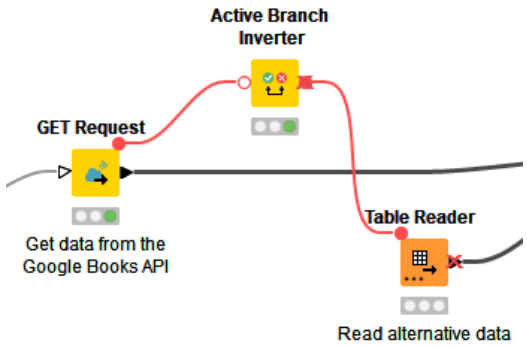
The original input or the alternative input, depending on node failure

Reasons for Failure (if any)



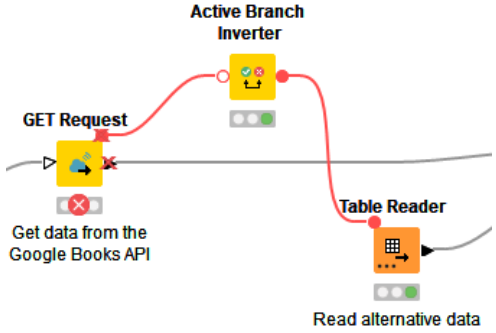
Active Branch Inverter

- Node execution can be triggered via flow variables: if they are connected to upstream nodes by flow variables, they will be automatically be executed



- If the node executes that is connected to the Active Branch Inverter, the alternative branch/node is not active

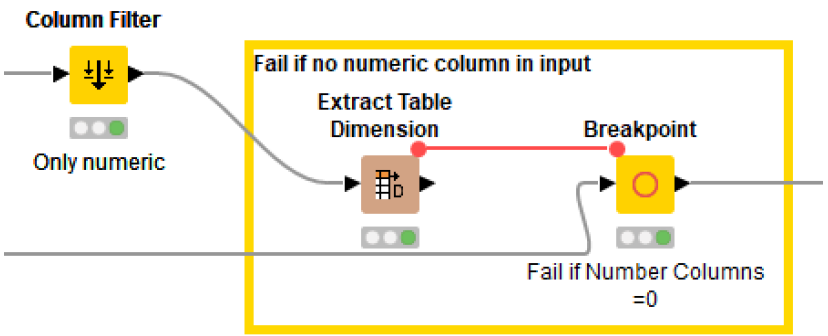
- If the node that is connected to the Active Branch Inverter does NOT execute, the execution of the alternative branch/node is triggered by flow variable

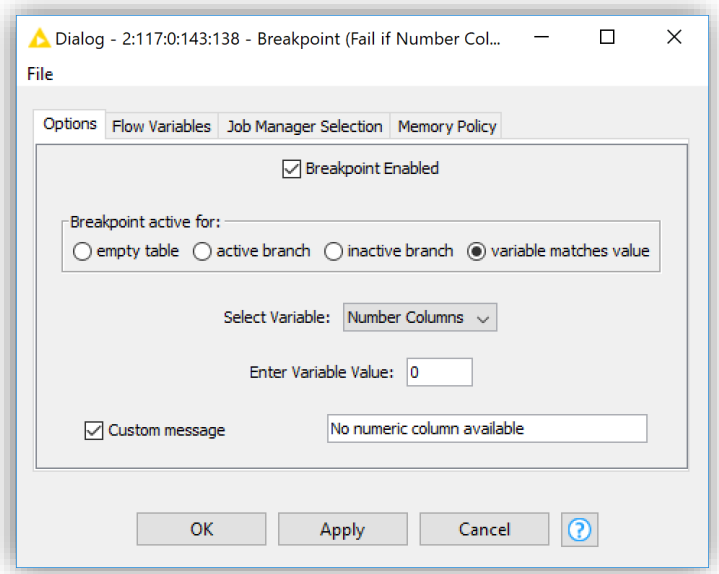


- like an If-switch for node execution, the condition is the execution of the upstream node

Breakpoint

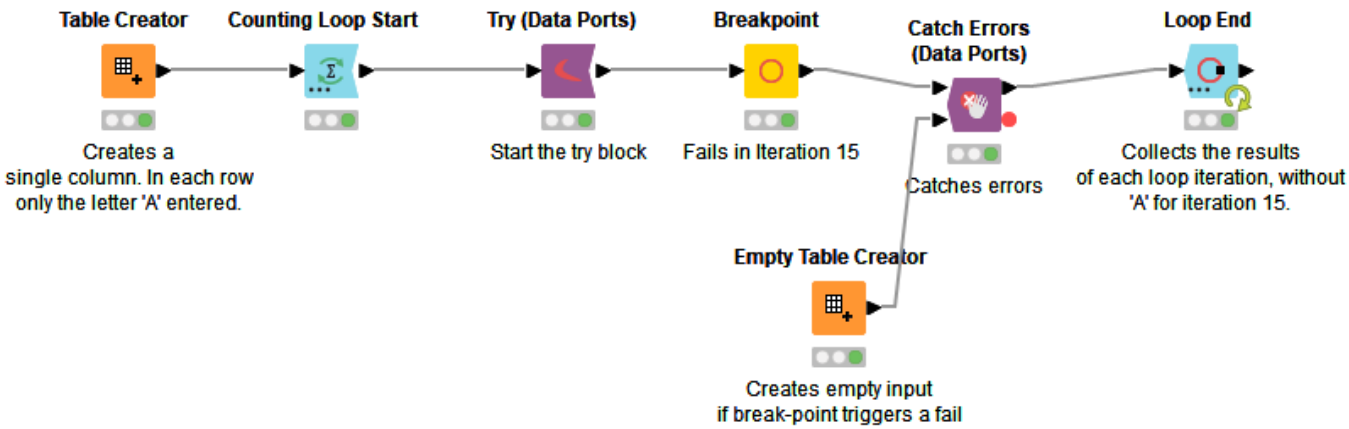
- Stops execution of a workflow branch
- Useful to stop the execution of a component and provide a custom error message
- Execution stops based on the selected condition:
  - Empty table
  - Active/Inactive branch
  - Flow Variable value





Breakpoint and Try-Catch

- Recover from a Breakpoint using the Try-Catch

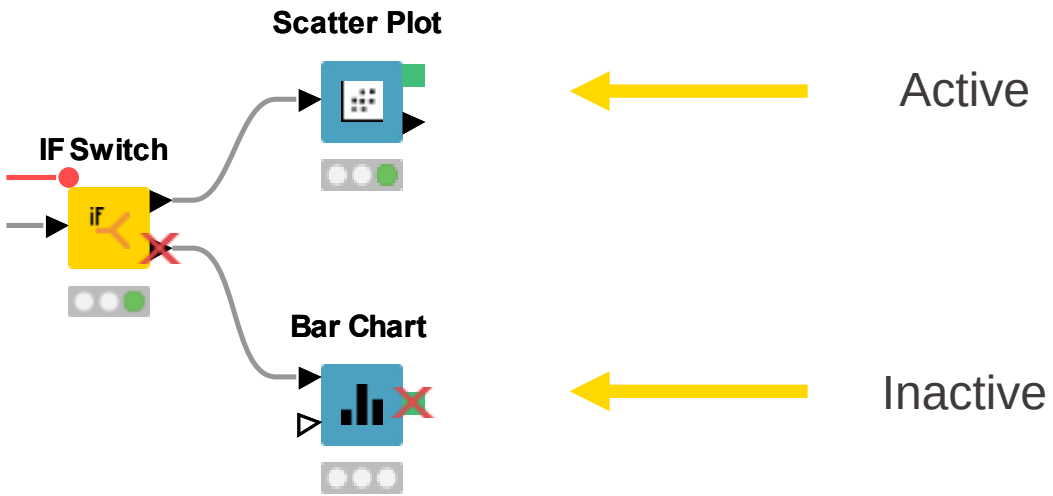


Row ID	S Text	I Iteration
Row0#0	A	0
Row0#1	A	1
Row0#2	A	2
Row0#3	A	3
Row0#4	A	4
Row0#5	A	5
Row0#6	A	6
Row0#7	A	7
Row0#8	A	8
Row0#9	A	9
Row0#10	A	10
Row0#11	A	11
Row0#12	A	12
Row0#13	A	13
Row0#14	A	14
Row0#15	?	15
Row0#16	A	16
Row0#17	A	17
Row0#18	A	18
Row0#19	A	19

- https://kni.me/w/yNkGIznDDl8YcVby

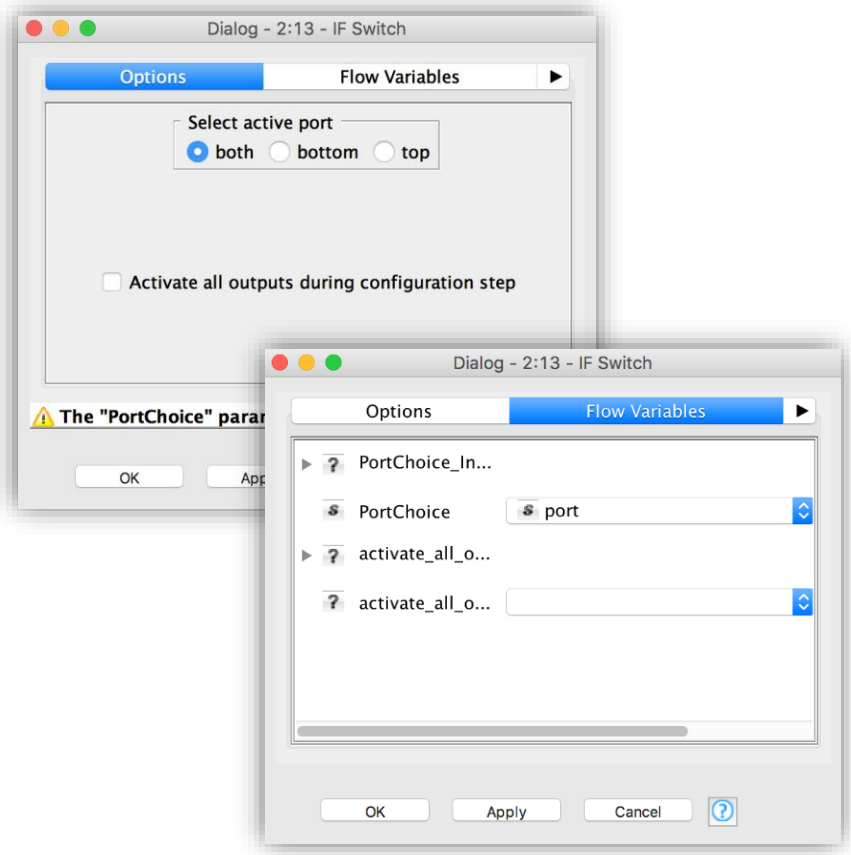
Switches

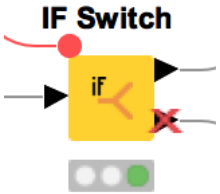
- A switch allows you to selectively activate branches of a workflow
- Inactive branches are marked with a red x on their output ports. Inactive nodes propagate down stream.



If Switch

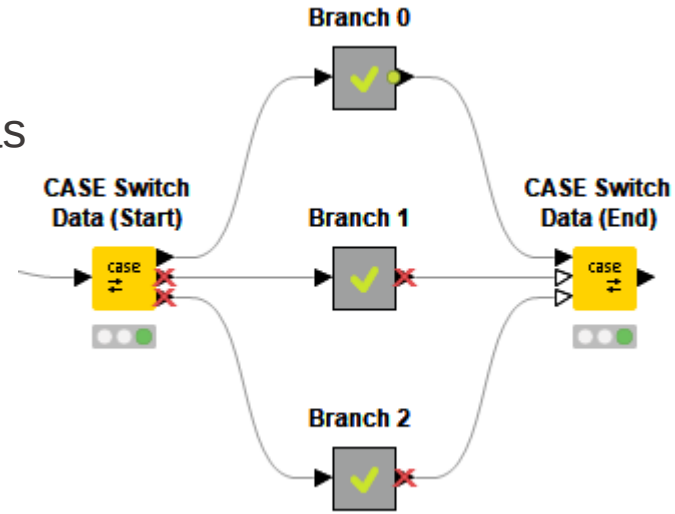
- Controls which branches of your workflow are active programmatically
- Controlled with a Flow Variable, setting the value to the literal Strings: “top”, “bottom”, “both”
- May be used in Flow Variables or tables (different nodes)





Case Switch Data

- Similar to If-Switch: Takes data from single input port and passes it to the active output port
- Nodes connected to inactive branches are not executed
- Configure via node dialog, or pass port index as Flow Variable
  - 0, 1, 2 for top, middle, and bottom port
- Case switches also available for Flow Variable and model ports



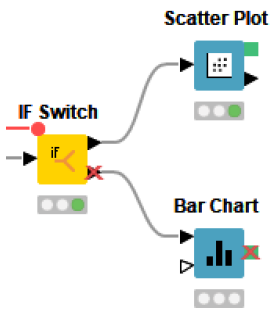
The difference between Loops and Switches

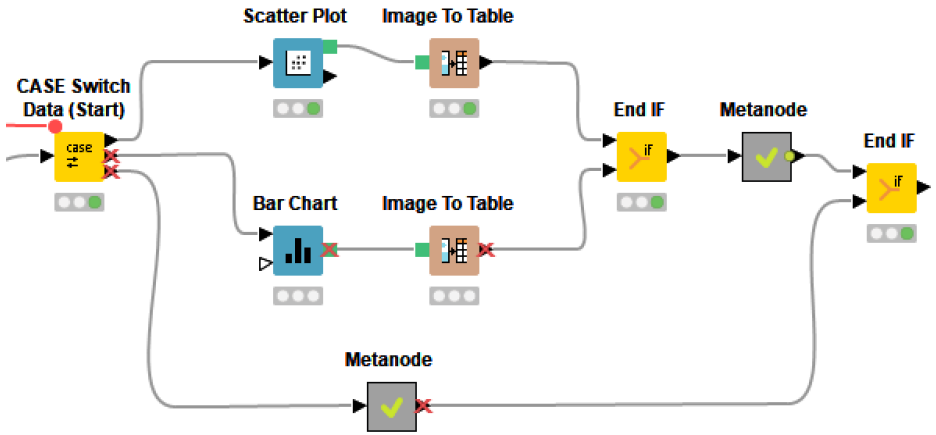
Loops

- The Loop Start is connected to the Loop End node, they form a pair.
- A loop iterates over a workflow part.

Switches

- A Switch Start can be used without a corresponding Switch End. They can also be combined.



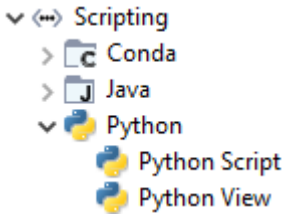


Output

KNIME Python Integration

KNIME Python Integration

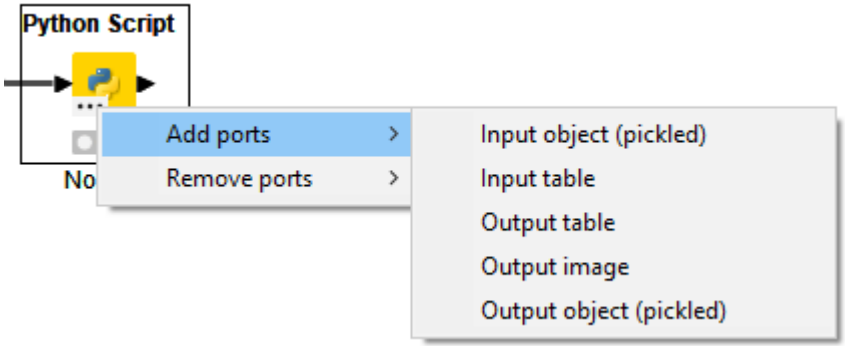
- Install the KNIME Python Integration for the nodes
- Since 4.7 comes with a bundled environment under the hood
- Includes the extensions on the right




```
- beautifulsoup4        # 4.11.1
- cloudpickle           # 2.2.0
- ipython               # 8.8.0
- jedi>=0.18.1          # 0.18.2
- matplotlib-base       # 3.6.2
- markdown
- nbformat              # 5.7.1
- nltk                  # 3.8.1
- nomkl                 # 1.0
- numpy>=1.22           # 1.24.1
- openpyxl              # 3.0.10
- pandas                # 1.5.2
- packaging             # 21.3
- pillow                # 9.4.0
- plotly                # 5.11.0
- py4j                  # 0.10.9.7
- pyarrow>=9            # 9.0.0
- python=3.9            # 3.9.15
- python-dateutil       # 2.8.2
- pytz                  # 2022.7
- pyyaml                # 6.0
- requests              # 2.28.1
- scikit-learn          # 1.2.0
- scipy                 # 1.10.0
- seaborn               # 0.12.2
- statsmodels           # 0.13.5
```

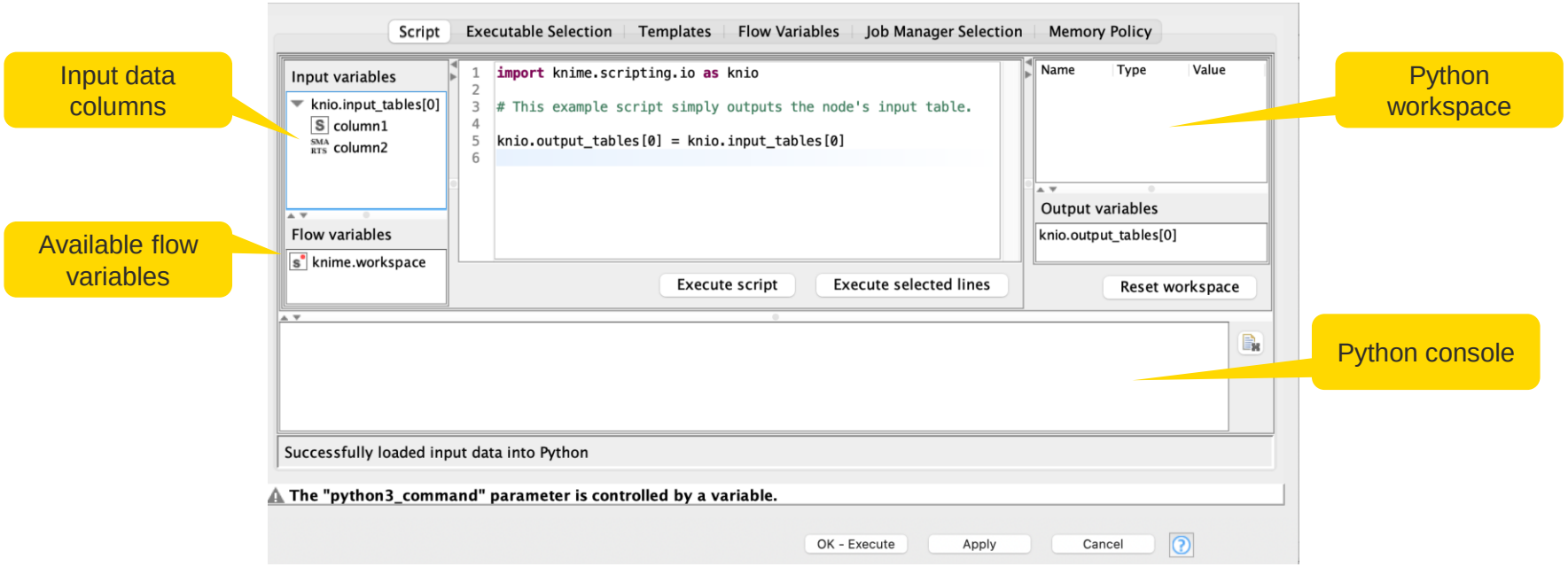
Python Scripting node

- Variable input and output ports
- Click on the three dots to change them



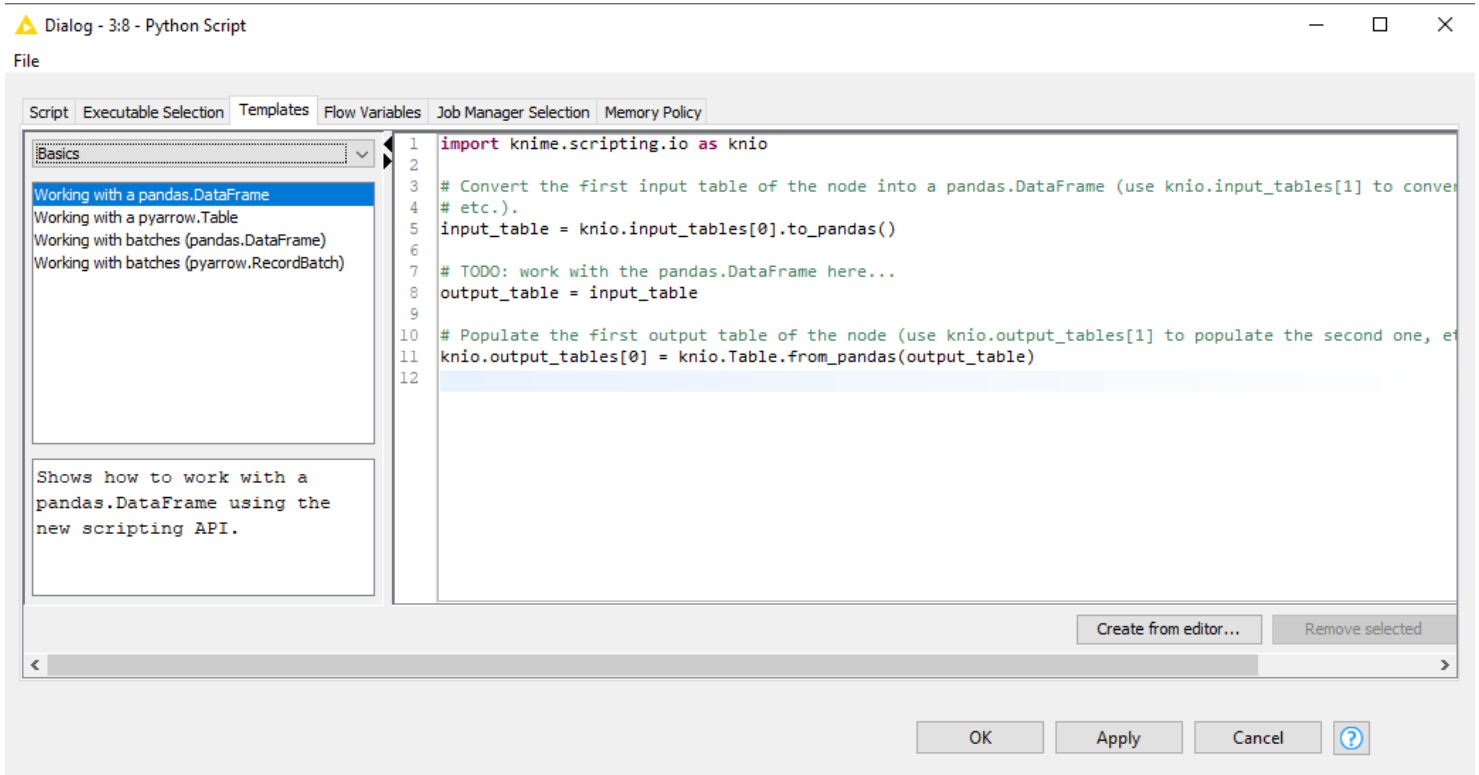
Python Scripting node

- Configuring it opens the code editor
- use the *knime.scripting.iomodule* (imported as *knio* by default) to access the node's input data and populate its output data



Python Scripting node

- Find code templates under the Templates tab



Accessing data

- tables
  - knio.input_tables[i] and knio.output_tables[i]
- objects
  - knio.input_objects[i] and knio.output_objects[i]
- images
  - knio.output_images[i]
  - must be either a string describing an SVG image or a byte array encoding a PNG image
- flow variables
  - knio.flow_variables['name_of_flow_variable']

- where i is the index of the corresponding table/object/image (starting with 0)

KNIME Tables to Python

- Pandas data frames
- To Pandas df:
  - df = knio.input_tables[0].to_pandas()
- From Pandas df:
  - knio.output_tables[0] = knio.Table.from_pandas(df)

- PyArrow tables
- To PyArrow table:
  - table = knio.input_tables[0].to_pyarrow()
- From PyArrow table
  - knio.output_tables[0] = knio.Table.from_pyarrow(table)

Porting Scripts from the Legacy Nodes

```
import knime.scripting.io as knio
input_table_1 = knio.input_tables[0].to_pandas()

# the script from the legacy nodes goes here

knio.output_tables[0] = knio.Table.from_pandas(output_table_1)
```

Jupyter Notebook

```
# Path to the folder containing the notebook, e.g. the folder 'data' contained
# in my workflow folder
notebook_directory = "knime://knime.workflow/data/"

# Filename of the notebook
notebook_name = "sum_table.ipynb"

# Load the notebook as a Python module
import knime.scripting.jupyter as knupyter
my_notebook = knupyter.load_notebook(notebook_directory, notebook_name)

# Print its textual contents
knupyter.print_notebook(notebook_directory, notebook_name)

# Call a function 'sum_each_row' defined in the notebook
output_table = my_notebook.sum_each_row(input_table)
```

https://docs.knime.com/latest/python_installation_guide/index.html#jupyter-notebooks

Jupyter Notebook

- Juypter notebook of the example

jupyter simple Last Checkpoint: 30 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted my-rdkit-env

Run Code

In [1]:
```
from rdkit import Chem
from rdkit.Chem import Descriptors
```

In [4]:
```
from rdkit import RDConfig
import sys, os
sys.path.append(os.path.join(RDConfig.RDContribDir,'SA_Score'))
import sascorer
```

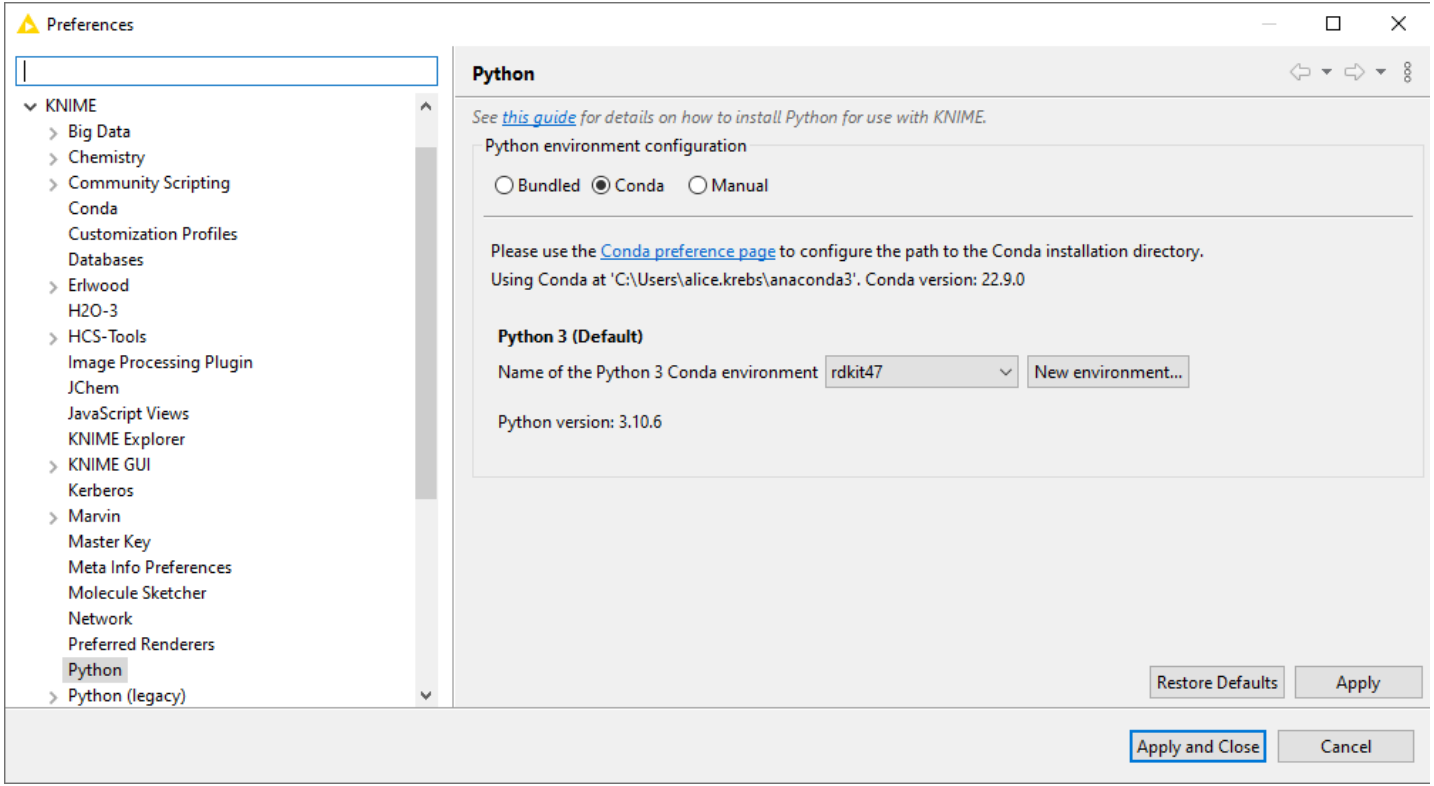
In [2]:
```
def return_descriptors(m):
    return Descriptors.qed(m),sascorer.calculateScore(m)
```

KNIME packages for environments

- *knime-python-base*
  - contains the basic packages which are always needed
- *knime-python-scripting*
  - contains *knime-python-base* and installs additionally the packages used in the Python Script node

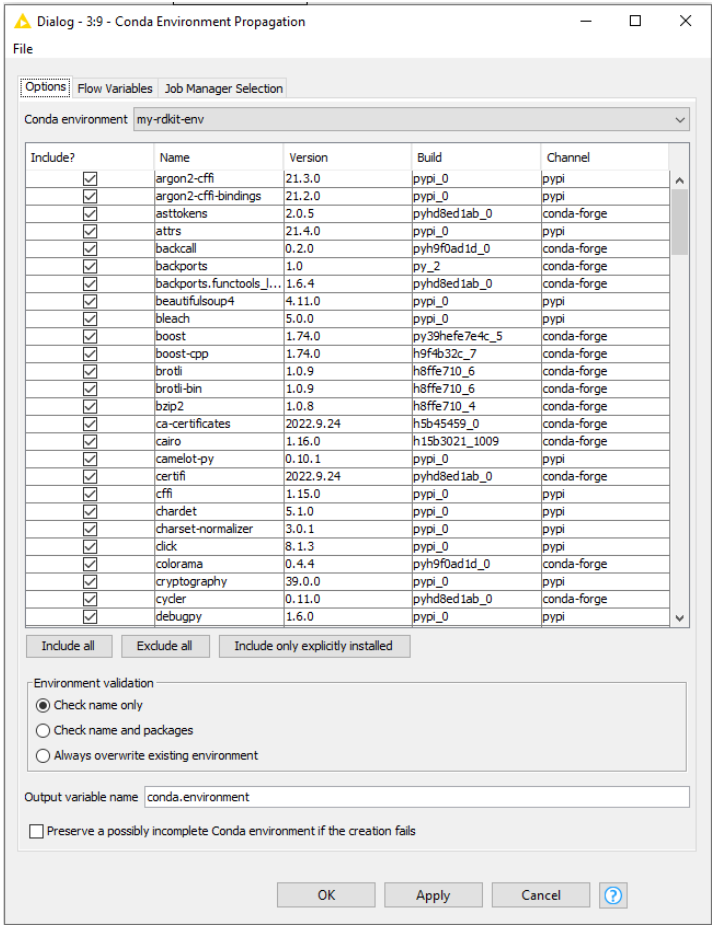
Pointing to Environments

- *KNIME > File > Preferences*



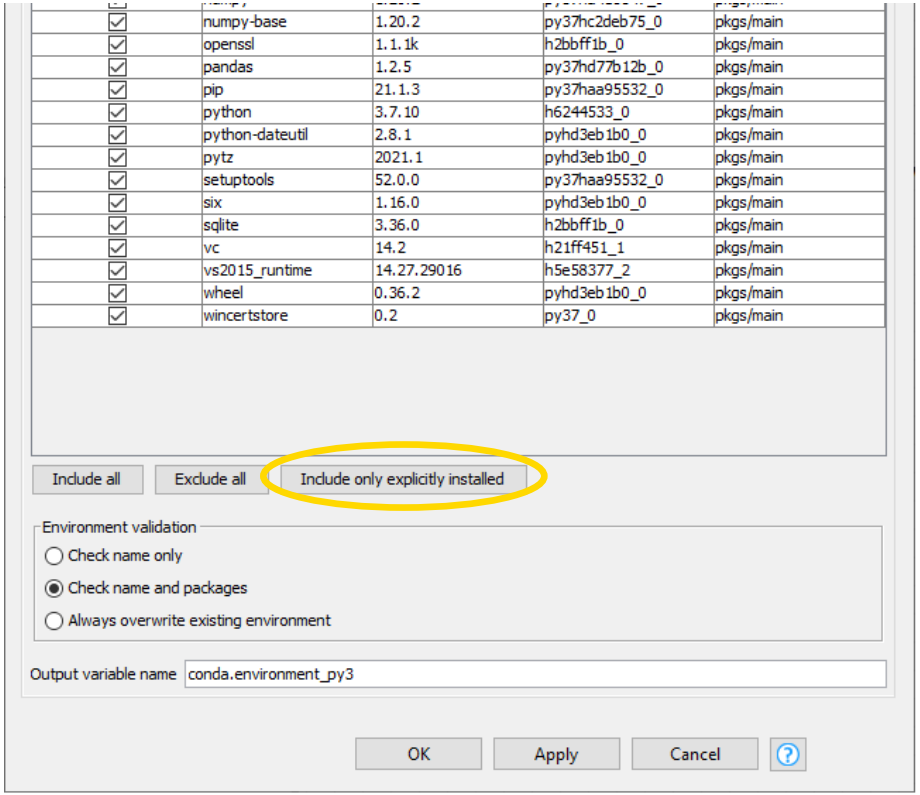
Conda Environment Propagation Node

- Configure custom Python environments to be used
- Bundle environments together with workflows for easier re-execution and workflow sharing
- Define Conda and custom Python environments under *Preferences* > *KNIME* > *Python* first



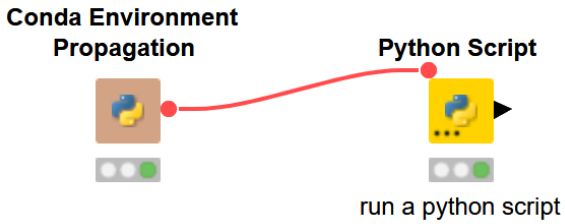
Conda Environment Propagation Node

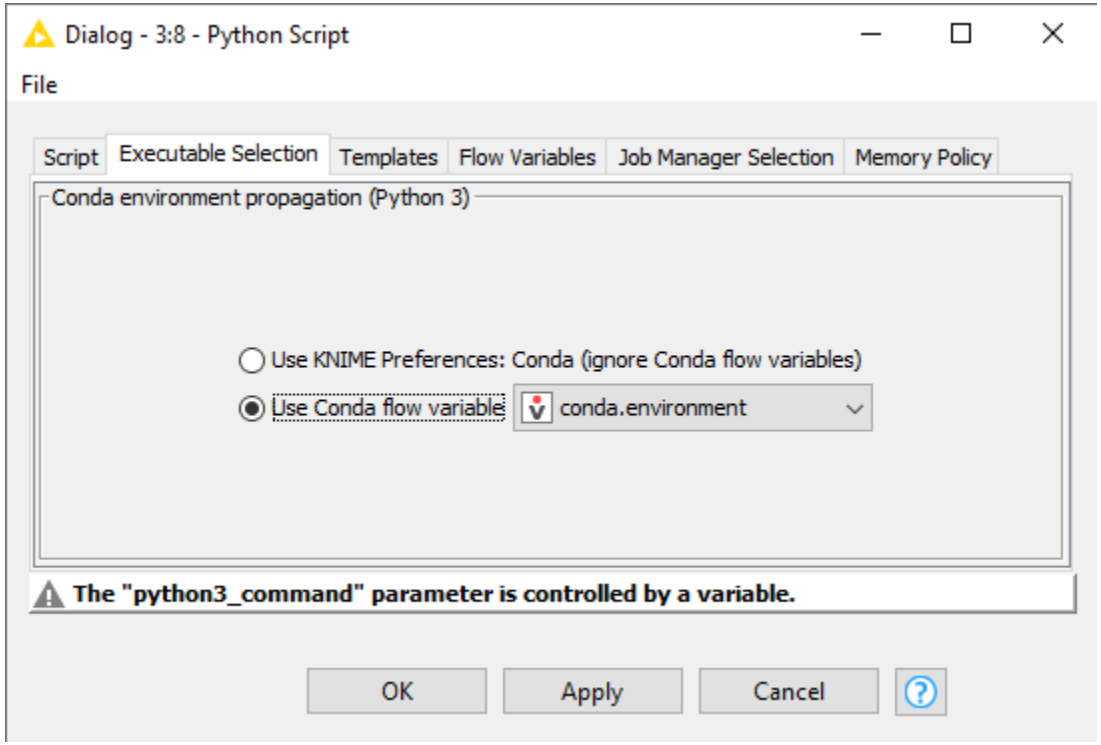
- *Tip:* choose only explicitly installed packages to avoid conflicts when using the workflow on different operating systems



Define the Environment to be used

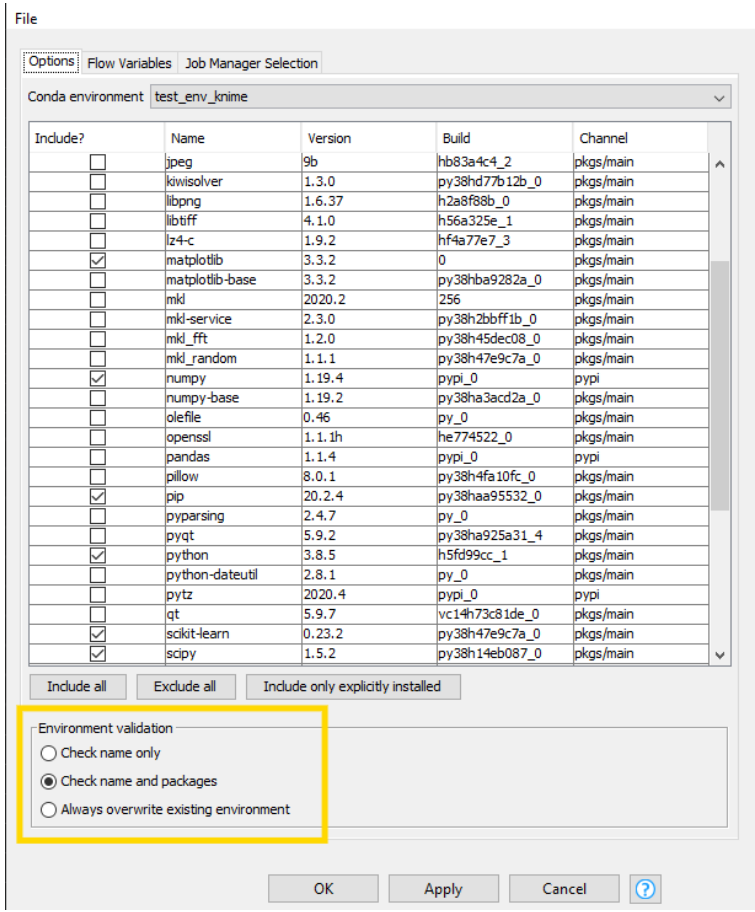
- The Conda Environment Propagation node allows using different envs in different nodes
- Set the flow variable in the scripting nodes





Establish the defined env on the target machine

- Conda needs to be set up and configured in the preferences
- the node will check whether a local Conda environment exists that matches its configured environment
- 3 different types of validation:
  - *Check name only:* env with the same name
  - *Check name and packages*: check both name and requested packages
  - *Always overwrite existing environment*: disregard the existence of an equal environment on the target machine and complete recreation



Chemistry in Python scripting

https://forum.knime.com/t/announcing-the-interoperability-between-rdkit-and-python-in-knime-4-7/60416

- Converting between molecule types in Python
- Import the library *knime.types.chemistry*
  - `import knime.types.chemistry as ktchem`
    - ktchem.SdfValue
    - ktchem.RxnValue
    - ktchem.SmilesValue
    - ktchem.SmartsValue

Chemistry in Python

AdapterValue - class
AdapterValueFactory - class
CDXMLValue - class
CDXMLValueFactory - class
CMLAdapterValue - class
CMLAdapterValueFactory - class
CMLValue - class
CMLValueFactory - class
CtabValue - class
CtabValueFactory - class
HELMAdapterValue - class
HELMAdapterValue - class
HELMAdapterValueFactory - class
HELMValue - class
HELMValueFactory - class
InchiAdapterValue - class
InchiAdapterValueFactory - class
InchiValue - class
InchiValueFactory - class
kt - module
Mol2AdapterValue - class
Mol2AdapterValueFactory - class

Mol2AdapterValue - class
Mol2AdapterValueFactory - class
Mol2Value - class
Mol2ValueFactory - class
MolAdapterValue - class
MolAdapterValueFactory - class
MolValue - class
MolValueFactory - class
RxnAdapterValue - class
RxnAdapterValueFactory - class
RxnValue - class

RxnAdapterValue - class
RxnAdapterValueFactory - class
RxnValue - class
RxnValueFactory - class
SdfAdapterValue - class
SdfAdapterValueFactory - class
SdfValue - class
SdfValueFactory - class
SlnValue - class

SlnValue - class
SlnValueFactory - class
SmartsAdapterValue - class
SmartsAdapterValueFactory - class
SmartsValue - class
SmartsValueFactory - class
SmilesAdapterValue - class
SmilesAdapterValueFactory - class
SmilesValue - class
SmilesValueFactory - class
StringBasedValueFactory - class

Useful links

- KNIME Python Integration Guide:
  - https://docs.knime.com/latest/python_installation_guide/index.html
- Blog post “KNIME and Jupyter”:
  - https://www.knime.com/blog/knime-and-jupyter
- Blog post “Manage Your Python Environments with Conda and KNIME”:
  - https://www.knime.com/blog/how-to-manage-python-environments-conda-and-knime
- KNIME Python Node Extension guide:
  - https://docs.knime.com/2022-06/pure_python_node_extensions_guide/index.html#introduction
- Find the documentation for the new API:
  - https://knime-python.readthedocs.io/en/stable/
- Get in touch with the developers at KNIME and provide feedback:
  - https://forum.knime.com/c/knime-development/
  - please add the tag “Python”
- Blog post “4 Steps for your Python Team to Develop KNIME Nodes”:
  - https://www.knime.com/blog/4-steps-for-your-python-team-to-develop-knime-nodes

Create your own KNIME nodes in Python

Needed extension

- Install the Python Extension Development (Labs)

Extension

KNIME Python Extension Development (Labs)

v 5.1.0

This extension allows the development and integration of nodes written in Python.

KNIME packages for environments

- *knime-python-base*
  - contains the basic packages which are always needed
- *knime-extension*
  - contains the node development API

- Requires -c knime -c conda-forge

Configurations

- ***config.yml:*** information that binds your extension and the corresponding Python environment with KNIME Analytics Platform

```yaml
org.knime.rdkit_extension: # {group_id}.{name} from the knime.yml
  src: C:\knime-python\basic-default\tutorial_extension # Path to folder containing the extension files
  conda_env_path: C:\Users\alice.krebs\anaconda3\envs\rdkit47 # Path to the Python environment to use
  debug_mode: true # Optional line, if set to true, it will always use the latest changes of execute/configure, when that method is used within
```

- my_conda_env.yml:

```yaml
name: rdkit_node_env
channels:
  - knime
  - conda-forge
dependencies:
  - packaging
  - python=3.9
  - knime-extension
  - knime-python-base
  - rdkit
```

```
.
├── tutorial_extension
│   ├── icon.png
│   ├── knime.yml
│   ├── LICENSE.TXT
│   └── my_extension.py
├── config.yml
├── my_conda_env.yml
├── Example_with_Python_node.knwf
└── README.md
```

- ***knime.yml:*** important metadata about your extension

```yaml
name: rdkit_extension # Will be concatenated with the group_id to an ID
author: Alice
env_yml_path: C:\knime-python\basic-default\my_conda_env.yml # This is necessary for bundling, but not needed during development
extension_module: new_rdkit_nodes # The .py Python module containing the nodes of your extension
description: RDKit Extension # Human readable bundle name / description
long_description: This extension has new RDKit nodes written in Python.
group_id: org.knime
version: 0.1.0 # Version of this Python node extension
vendor: KNIME AG, Zurich, Switzerland
license_file: LICENSE.TXT # Best practice: put your LICENSE.TXT next to the knime.yml; otherwise you would need to change to path/to/LICENSE.txt
```

Configurations

- Point KNIME AP to the *config.yml* in the knime.ini file, in order to allow it to use our extension and its Python environment

```
--add-opens=java.security.jgss/sun.security.jgss.krb5=ALL-UNNAMED
--add-exports=java.security.jgss/sun.security.jgss=ALL-UNNAMED
--add-exports=java.security.jgss/sun.security.jgss.spi=ALL-UNNAMED
--add-exports=java.security.jgss/sun.security.krb5.internal=ALL-UNNAMED
--add-exports=java.security.jgss/sun.security.krb5=ALL-UNNAMED
-DXXchromium.remote_debugging_port=8888
-Dknime.python.extension.config=C:/knime-python/basic-default/config.yml
--add-opens=java.base/sun.security.ssl=ALL-UNNAMED
--add-opens=java.base/sun.security.util=ALL-UNNAMED
-server
-Dsun.java2d.d3d=false
-Dosgi.classloader.lock=classname
```

- Add the following line to the ini file:
  - *-Dknime.python.extension.config=<path/to/your/config.yml>*

Simple Example – Python code for KNIME node

C: > knime-python > basic-default > basic > tutorial_extension > MolFormula.py > ...

```python
import logging
import knime_extension as knext
from rdkit import Chem
import rdkit.Chem.rdMolDescriptors
import pandas as pd
LOGGER = logging.getLogger(__name__)

@knext.node(name="Get Molecular Formula", node_type=knext.NodeType.LEARNER, icon_path="icon.png", category="/")
@knext.input_table(name="Input Data", description="Input table containing SMILES")
@knext.output_table(name="Output Data", description="Input table appended with a column containing the molecular
class MolFormula:
    """
    This node returns the molecular formula from SMILES in an appended column.
    """

    molecule_column = knext.ColumnParameter(label="Select the column containing SMILES", description="Choose the

    def configure(self, configure_context, input_schema_1):
        return input_schema_1.append(knext.Column(knext.string(), "MolFormula"))


    def execute(self, exec_context, input_1):
        input_1_pandas = input_1.to_pandas()

        mols = []
        for smi in input_1_pandas[self.molecule_column]:
            mols.append(Chem.MolFromSmiles(smi))

        MolFormula = []
        for x in mols:
            MolFormula.append(Chem.rdMolDescriptors.CalcMolFormula(x))

        input_1_pandas['MolFormula'] = MolFormula
        return knext.Table.from_pandas(input_1_pandas)
```

Import libraries

Manipulate data

Simple Example – Python code for KNIME node

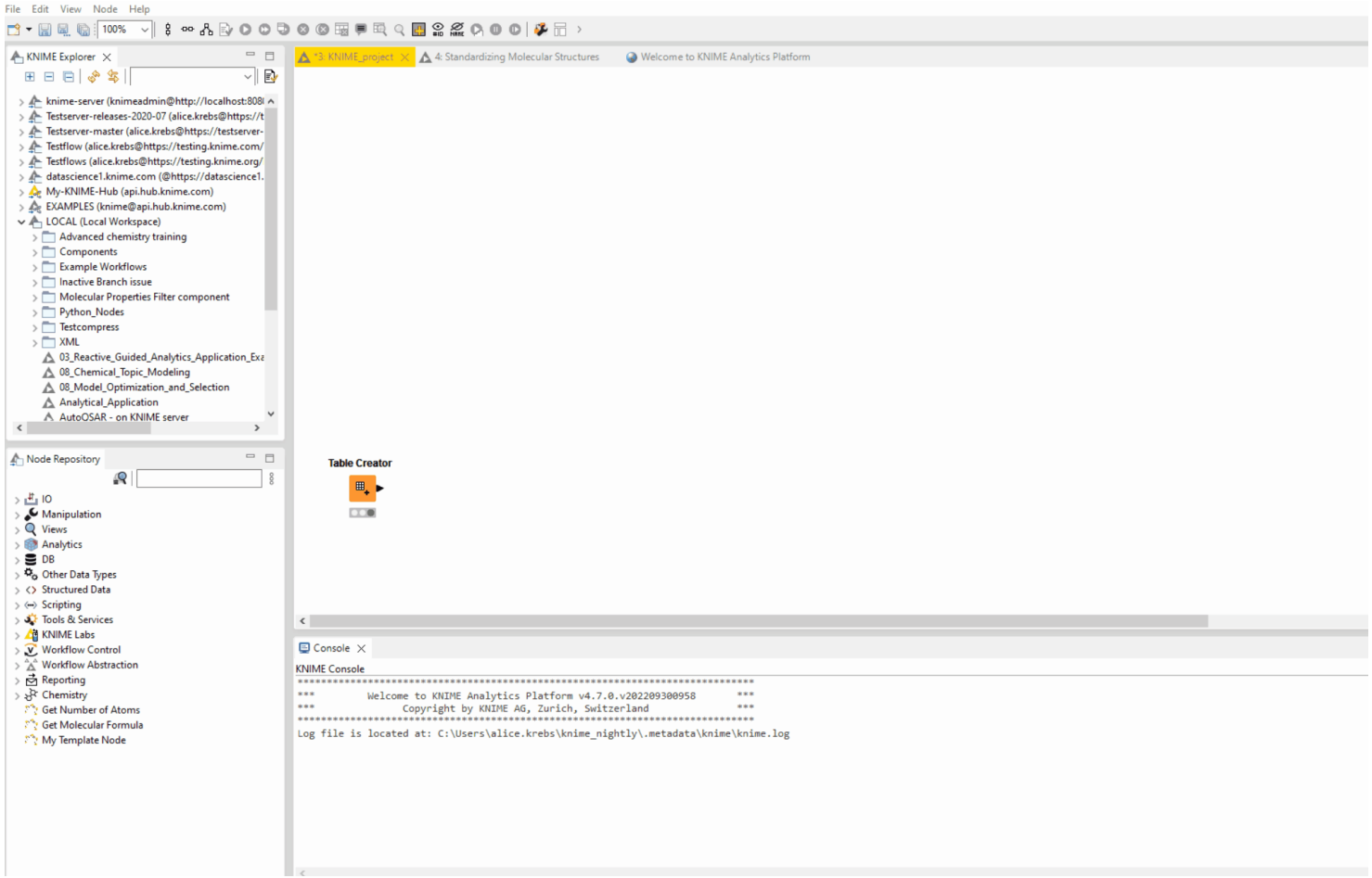
C: > knime-python > basic-default > basic > tutorial_extension > MolFormula.py > ...

```python
import logging
import knime_extension as knext
from rdkit import Chem
import rdkit.Chem.rdMolDescriptors
import pandas as pd
LOGGER = logging.getLogger(__name__)

@knext.node(name="Get Molecular Formula", node_type=knext.NodeType.LEARNER, icon_path="icon.png", category="/")
@knext.input_table(name="Input Data", description="Input table containing SMILES")
@knext.output_table(name="Output Data", description="Input table appended with a column containing the molecular
class MolFormula:
    """
    This node returns the molecular formula from SMILES in an appended column.
    """

    molecule_column = knext.ColumnParameter(label="Select the column containing SMILES", description="Choose the

    def configure(self, configure_context, input_schema_1):
        return input_schema_1.append(knext.Column(knext.string(), "MolFormula"))


    def execute(self, exec_context, input_1):
        input_1_pandas = input_1.to_pandas()

        mols = []
        for smi in input_1_pandas[self.molecule_column]:
            mols.append(Chem.MolFromSmiles(smi))

        MolFormula = []
        for x in mols:
            MolFormula.append(Chem.rdMolDescriptors.CalcMolFormula(x))

        input_1_pandas['MolFormula'] = MolFormula
        return knext.Table.from_pandas(input_1_pandas)
```

Define input and output ports

Define node dialogue, input and output table spec

Simple Example – Resulting node



3 steps to create a custom Python-based node

- Step 1: **Install** necessities
- Step 2: **Establish** connection between KAP and external Python script(s)
- Step 3: **Write** Python-KNIME specific code

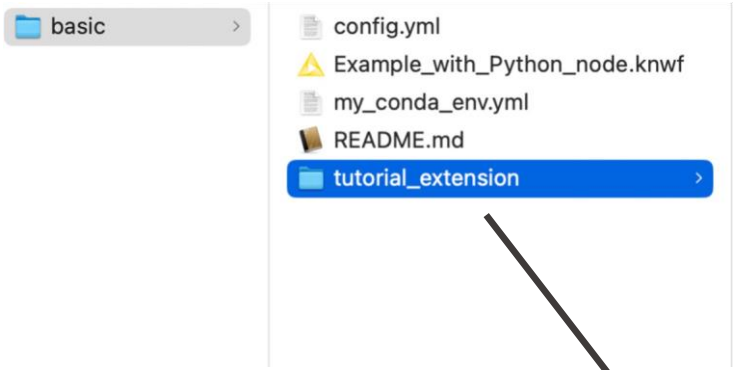
© 2023 KNIME AG. All rights reserved.

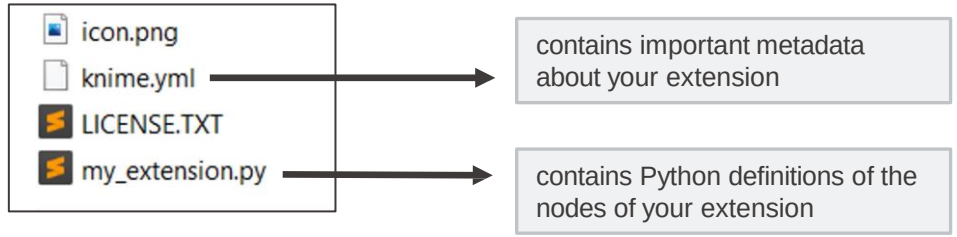
Step 1: Install Necessities

- Install KNIME
  - Install KNIME Analytics Platform version **4.6.0** or higher
- Install Extension
  - File > Install KNIME Extensions > "*KNIME Python Node Development Extension (Labs)*"
- Install (mini)conda
  - docs.conda.io/en/latest/miniconda.html
  - `conda create -n my_python_env python=3.9 packaging knime-python-base knime-extension -c knime -c conda-forge`
- Download the "basic" folder from [KNIME Docs]

© 2023 KNIME AG. All rights reserved.

“Basic” Folder Structure

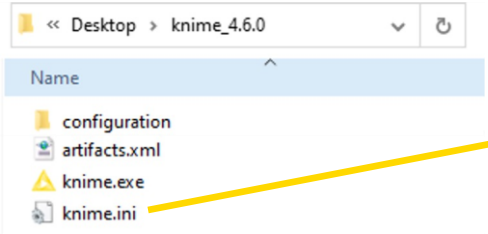




© 2023 KNIME AG. All rights reserved.

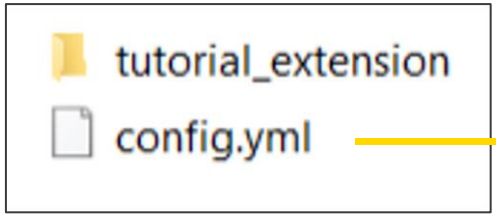
Step 2: Establish Connection

You will change 2 files: knime.ini and config.yml



You must add this line at the end of the file:

```
...
 -Dknime.python.extension.config=C:\Users\paolo\Desktop\basic\config.yml
```




```
org.knime.fluent_extension:
  src: path/to/folder/of/template
  conda_env_path:
  path/to/my_python_env debug_mode:
  true
```

Path to folder containing the extension files

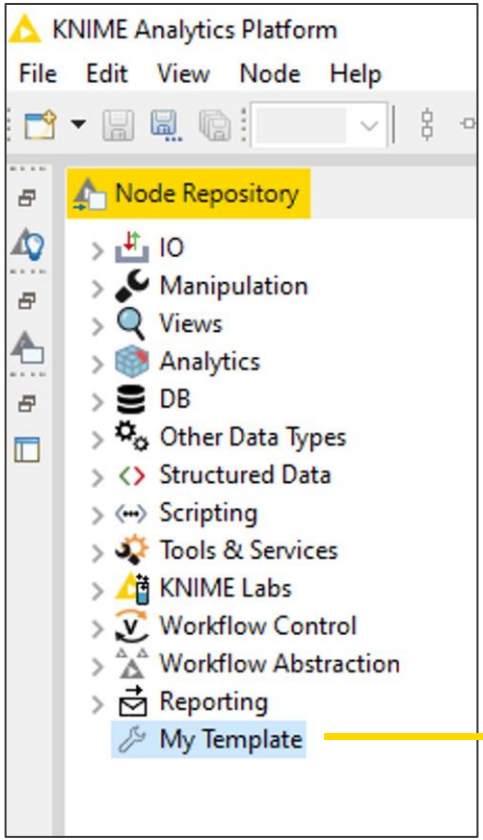
Path to the Python environment to use

```
conda env list
conda activate my_python_env
```

EXAMPLE:

```
org.knime.fluent_extension:
  src: C:\Users\paolo\Desktop\basic\tutorial_extension
  conda_env_path:
  C:\Users\paolo\miniconda3\envs\my_python_env debug_mode:
  true
```

Did it work so far? Start KNIME and See!

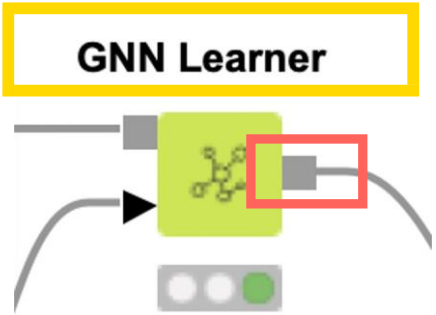


A new node should appear in your node repository!

© 2023 KNIME AG. All rights reserved.

Step 3: Write Code (node aesthetics)

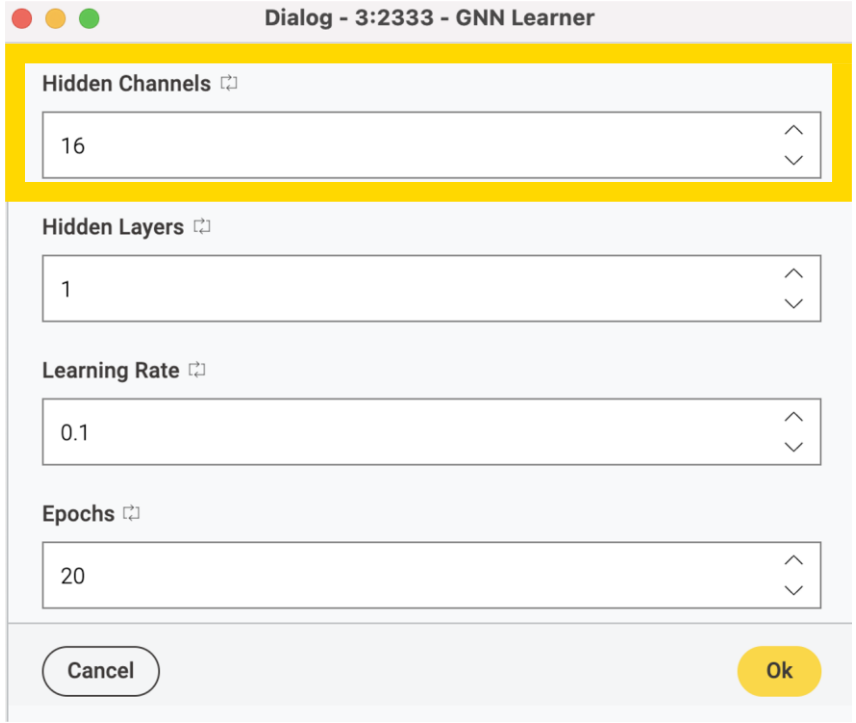
```
@knext.node(name="GNN Learner", node_type=knext.NodeType.LEARNER, icon_path="icon.png", category="/")
@knext.input_binary("Full Graph", "Pickled Graph", "org.knime.torch.graphcreator" )
@knext.input_table(name="Train Set",
                   description="A table that contains nodes need to be masked and passed downstream.")
```




```
@knext.output_binary(
    name="Model",
    description="Pytorch Geometric Model",
    id="org.knime.torch.learner",
)
class GNNLearner:
```

Step 3: Write Code (dialog)

```
class GNNLearner:
    hidden_channels = knext.IntParameter("Hidden Channels", "The number of hidden channels desired.", 1)
    number_of_hidden_layers = knext.IntParameter("Hidden Layers", "The number of hidden layers desired.
    learning_rate = knext.DoubleParameter("Learning Rate", "The learning rate to use in the optimizer",
    epochs = knext.IntParameter("Epochs", "The number of epochs to train for.", 1)
```



Step 3: Write Code (functionality)

Within the class, two functions are required: configure and execute

Configure defines specifications and rules:

```
def configure(self, configuration_context, input_table_1):

    if knext.string() not in [x.ktype for x in list(input_table_1)]:
        raise knext.InvalidParametersError(
            "The input table does not have any string columns. You need to have a string column for this node."
        )
```

Execute performs the operations:

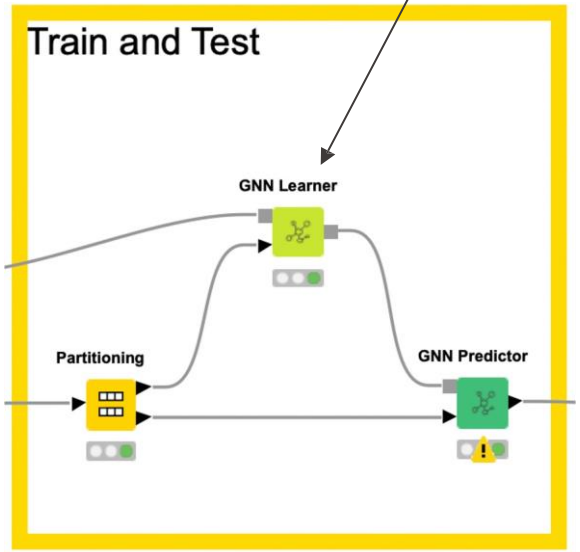
```
def execute(self, exec_context, graph_dict, train_set):
    train_set = train_set.to_pandas()
    graph_dict = pickle.loads(graph_dict)

    graph = graph_dict['graph']
    msk = AddMask(graph_dict['key'])
    masked_graph = msk(graph, train_set)
```


© 2023 KNIME AG. All rights reserved.

Python Script node vs Custom Python-based node

- Script node: easier to use for beginners and can help prototype
- Custom node: control, modularity, versioning, and collaboration

User cannot alter nodes within KNIME



Can write scripts in Visual Studio Code and version/collaborate with GitHub

```
class GNNLearner:
    """
    Train GNN
    """
    hidden_channels = knext.IntParameter("Hidden Channels", "The number of hidden channels desired.", 1
    number_of_hidden_layers = knext.IntParameter("Hidden Layers", "The number of hidden layers desired.
    learning_rate = knext.DoubleParameter("Learning Rate", "The learning rate to use in the optimizer",
    epochs = knext.IntParameter("Epochs", "The number of epochs to train for.", 1)
    criterion = nn.CrossEntropyLoss()

    def configure(self, configure_context, input_binary_1, input_schema_1):
        return knext.BinaryPortObjectSpec("org.knime.torch.learner")

    def execute(self, exec_context, graph_dict, train_set):
        train_set = train_set.to_pandas()
        graph_dict = pickle.loads(graph_dict)

        graph = graph_dict['graph']
        msk = AddMask(graph_dict['key'])
        masked_graph = msk(graph, train_set)
```

Resources for custom Python-based nodes

- Reminder: Must be using KNIME Analytics Platform 4.6+

Blog

Create

4 Steps for your Python Team to Develop KNIME Nodes

August 29, 2022 — by Paolo Tamagnini & Mahantesh Pattadkal

Geo Distances

Search or jump to... / Pull requests Issues

paolotamag / KNIME-Extension-Example Public

Code Issues Pull requests Actions Projects

Documentation

KNIME Documentation > KNIME Analytics Platform 4.6 > Create a New Python based KNIME Extension

KNIME Analytics Platform 4.6

Create a New Python based KNIME Extension

Graph Neural Networks

Custom GNN nodes for the KNIME Analytics Platform

Builing Graph Neural Network workflow based on torch_geometric library

Developers: Jinwei Sun, Paolo Tamagnini, & Victor Palacios

Chemistry in Pure Python Extensions

https://forum.knime.com/t/announcing-the-interoperability-between-rdkit-and-python-in-knime-4-7/60416

- Converting between molecule types in Python
- Using RDKit in Pure-Python nodes
  - Sending fingerprints from Python to KNIME
  - Working with KNIME-provided fingerprints in Python
  - Working with RXN reactions
-

Useful links

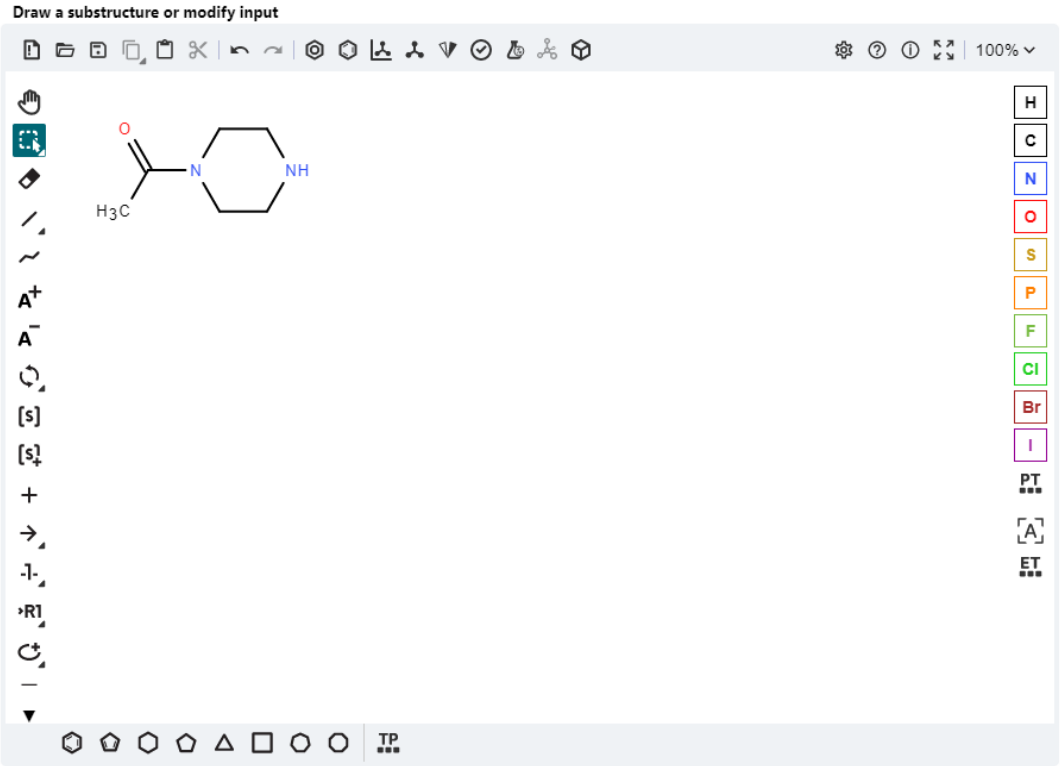
- KNIME Python Integration Guide:
  - https://docs.knime.com/latest/python_installation_guide/index.html
- Blog post “KNIME and Jupyter”:
  - https://www.knime.com/blog/knime-and-jupyter
- Blog post “Manage Your Python Environments with Conda and KNIME”:
  - https://www.knime.com/blog/how-to-manage-python-environments-conda-and-knime
- **KNIME Python Node Extension guide:**
  - **https://docs.knime.com/latest/pure_python_node_extensions_guide/index.html#introduction**
- Find the documentation for the new API:
  - https://knime-python.readthedocs.io/en/stable/
- Get in touch with the developers at KNIME and provide feedback:
  - https://forum.knime.com/c/knime-development/
  - please add the tag “Python”
- Blog post “4 Steps for your Python Team to Develop KNIME Nodes”:
  - https://www.knime.com/blog/4-steps-for-your-python-team-to-develop-knime-nodes

Chemistry Use Cases

Molecule Sketcher Widget

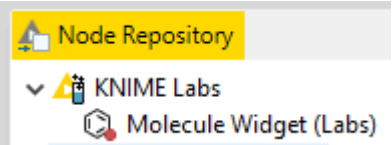
- Possible output formats
  - MOL
  - MOL V3000
  - RXN
  - RXN V3000
  - SMILES
  - Extended SMILES
  - SMARTS
  - CML
  - InChI
  - InChI with AuxInfo
  - KET

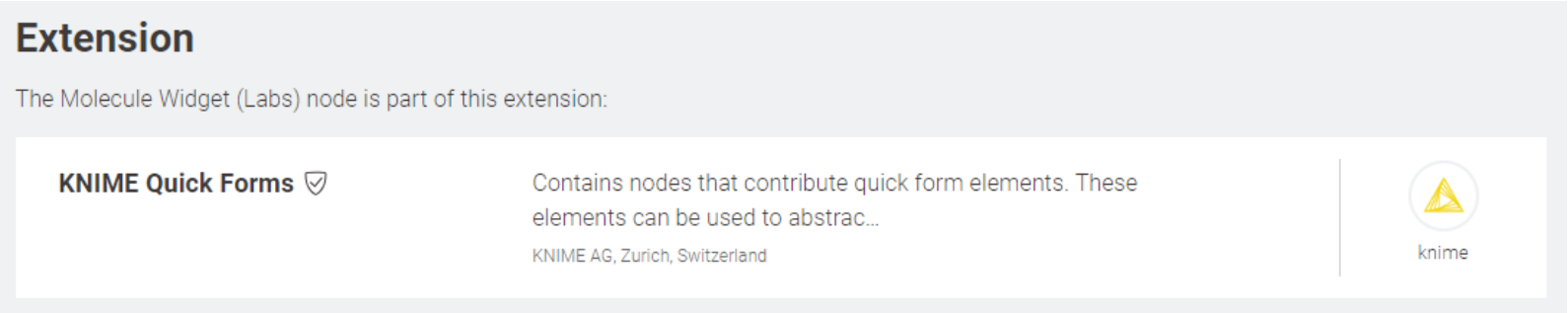
Molecule Widget (Labs)



Molecule Sketcher Widget

- Part of the Quick Forms extension
- Under *KNIME Labs* in the Node Repository
- Based on the EPAM Ketcher (version 2.7.2)

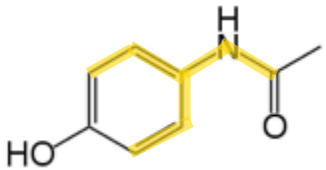




Today’s goal

- Perform hierarchical clustering based on molecular fingerprints and create an interactive view to pick interesting clusters
- Fingerprints
- Bit Vector Distances
- Hierarchical Clustering

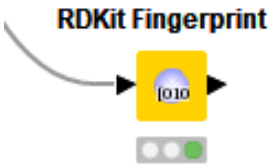
Molecular Fingerprints Encode Substructures

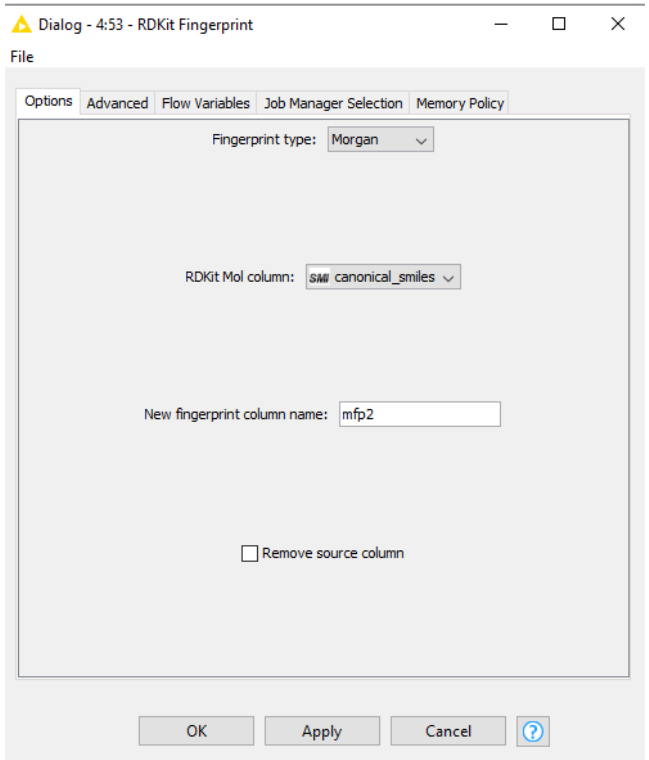


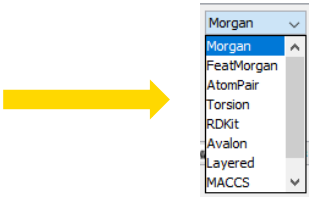
[00000**1**000000000000000001000…]

- Bit vectors, where each bit corresponds to a presence (1) or absence (0) of a certain atom and its neighbors

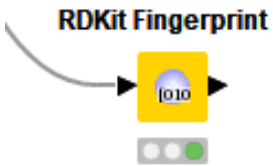
RDKit Fingerprint node

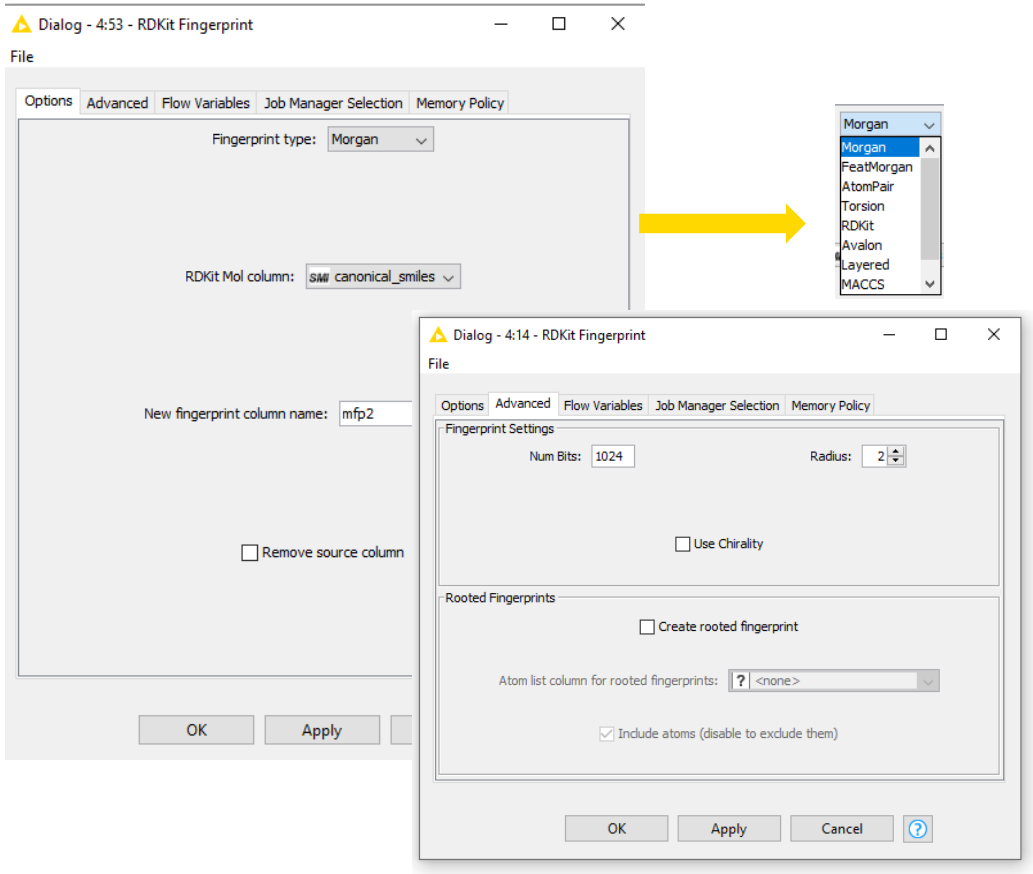




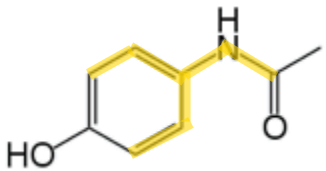


RDKit Fingerprint node





Tanimoto Similarity



[00000**1**0000000000000001000…]

$$\text{Tanimoto}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

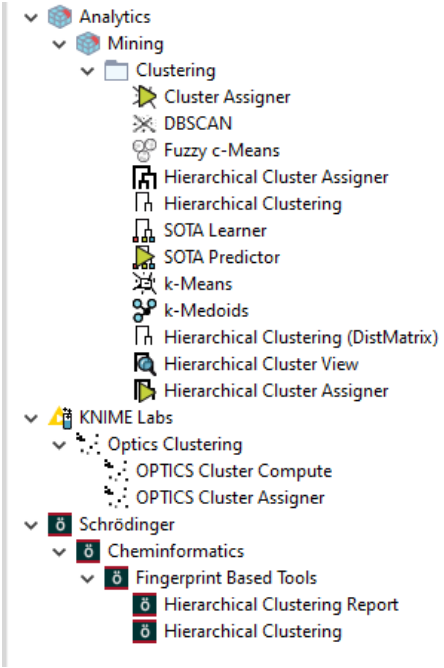
[0,1]
0 – no similarity
1 – “identical”

Goal of Cluster Analysis

Discover hidden structures in unlabeled data (unsupervised)

Clustering identifies a set of groups (*clusters*) $C_1, C_2 \cdots, C_k$ in the dataset such that:

- Objects within the *same* cluster C_i shall be as similar as possible
- Objects of *different* clusters C_i, C_j $(i \neq j)$ shall be as dissimilar as possible



Hierarchical Clustering

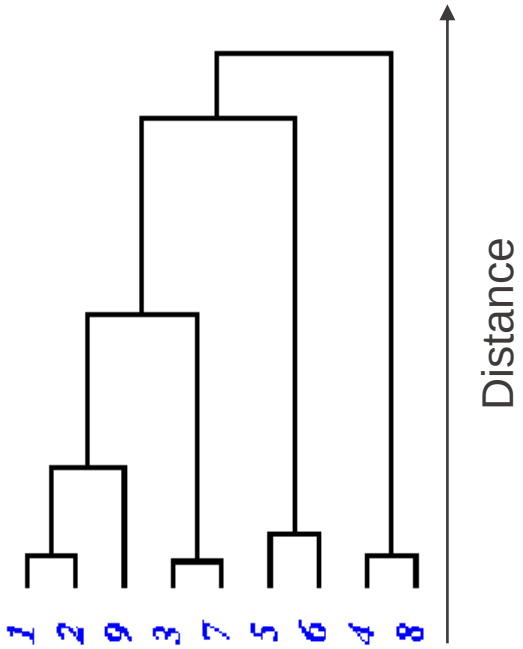
Linkage Hierarchies: Basics

Goal

- Construction of a hierarchy of clusters (*dendrogram*) by merging/separating clusters with minimum/maximum distance

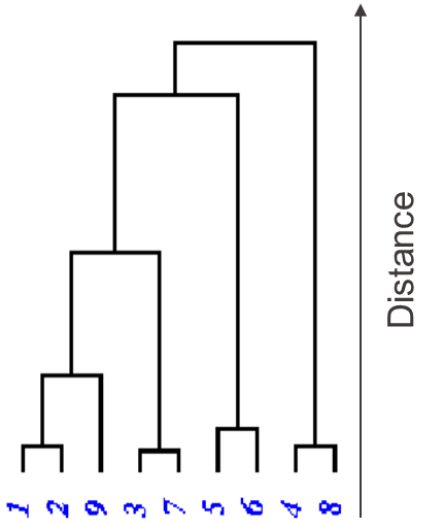
Dendrogram:

- A tree representing hierarchy of clusters, with the following properties:
  - Root: single cluster with the whole data set.
  - Leaves: clusters containing a single object.
  - Branches: merges / separations between larger clusters and smaller clusters / objects



Base Algorithm

1. Form initial clusters consisting of a single object and compute the distance between each pair of clusters.
2. Merge the two clusters having minimum distance.
3. Calculate the distance between the new cluster and all other clusters.
4. If there is only one cluster containing all objects:
   Stop, otherwise go to step 2.



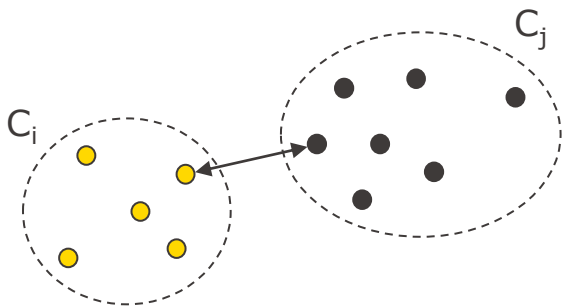
Single Linkage

- Distance between clusters (nodes):

$$Dist(C_1, C_2) = \min_{p \in C_1, q \in C_2} \{dist(p, q)\}$$

Distance of the closest two points, one from each cluster

- Merge Step: Union of two subsets of data points



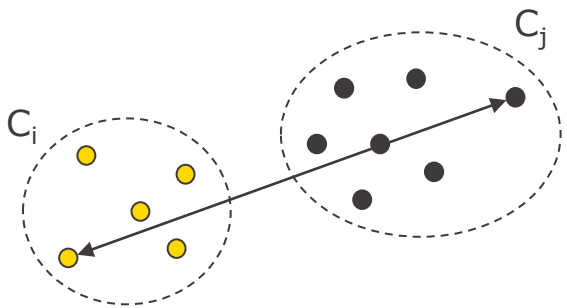
Complete Linkage

- Distance between clusters (nodes):

$$Dist(C_1, C_2) = \max_{p \in C_1, q \in C_2} \{dist(p, q)\}$$

Distance of the farthest two points, one from each cluster

- Merge Step: Union of two subsets of data points



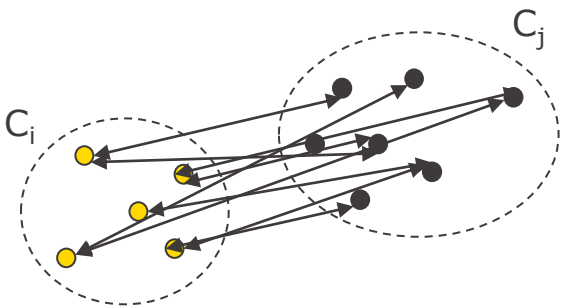
Average Linkage

- Distance between clusters (nodes):

$$Dist_{avg}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{p \in C_1} \sum_{p \in C_2} dist(p, q)$$

Average distance of all possible pairs of points between C_1 and C_2

- Merge Step: Union of two subsets of data points

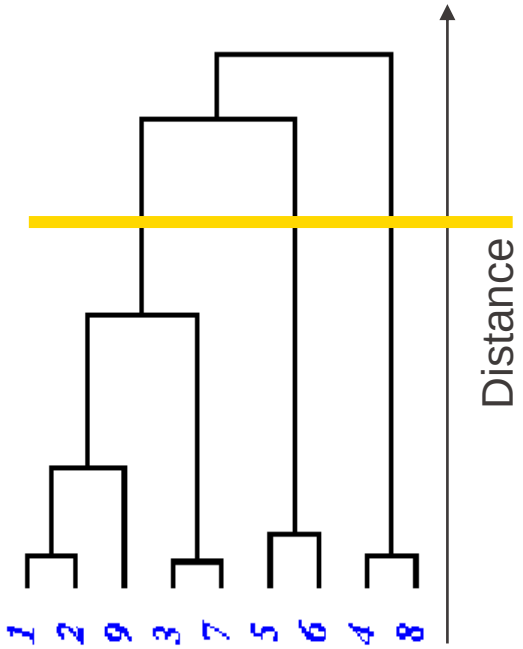


Comments on Single Linkage and Variants

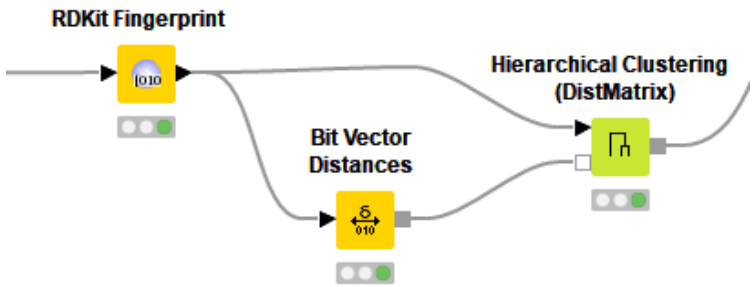
+ Finds not only a „flat“ clustering, but a hierarchy of clusters (dendrogram)

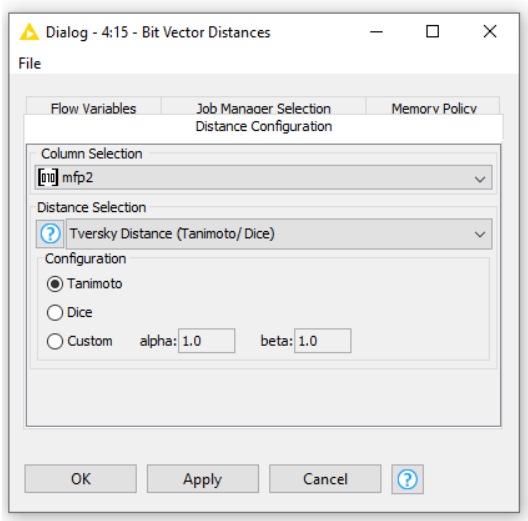
+ A single clustering can be obtained from the dendrogram (e.g., by performing a horizontal cut)

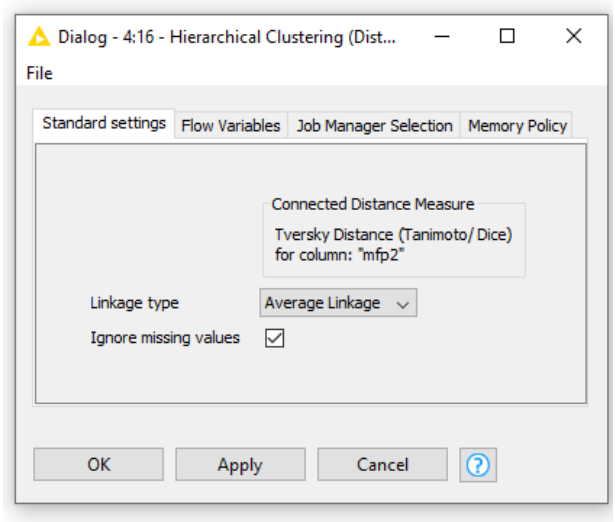
- Decisions (merges/splits) cannot be undone
- Sensitive to noise (Single-Link) (a „line“ of objects can connect two clusters)
- Inefficient → Runtime complexity at least $O(n^2)$ for n objects



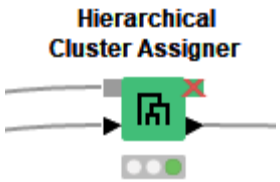
Bit Vector Distances and Hierarchical Clustering

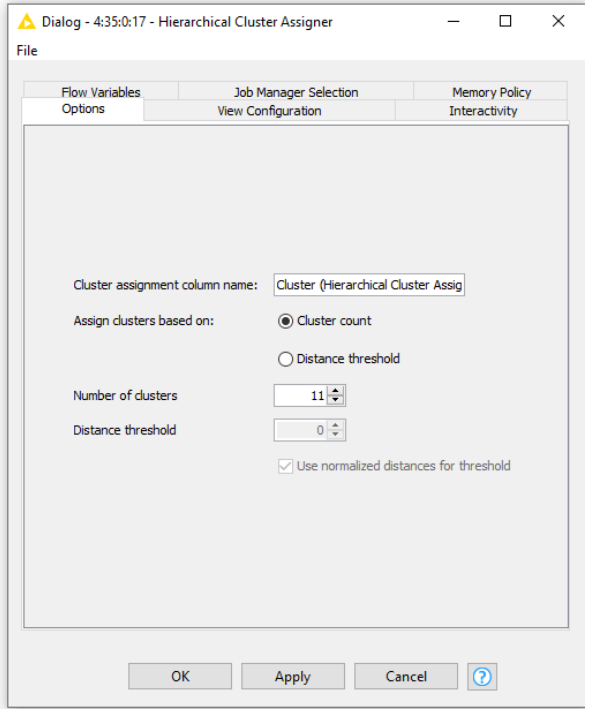


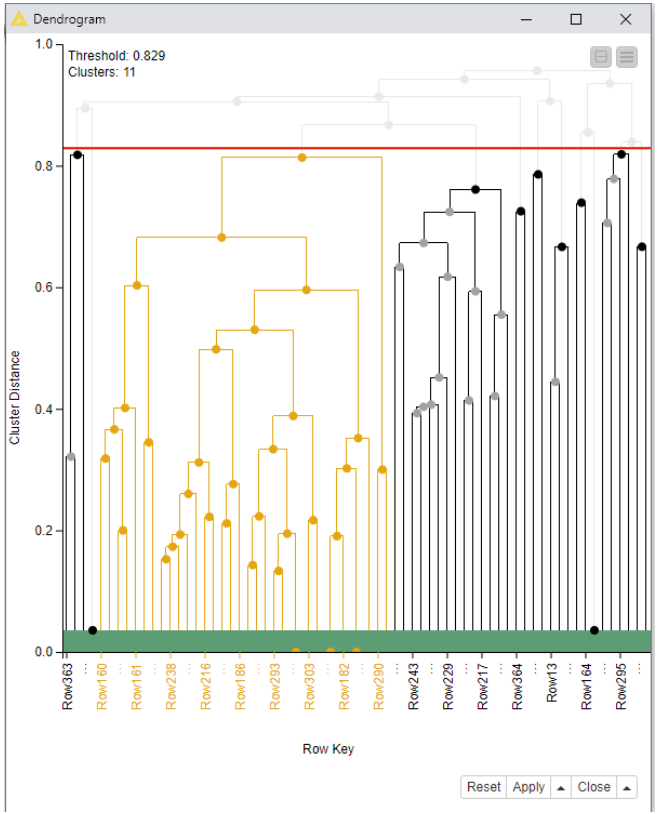




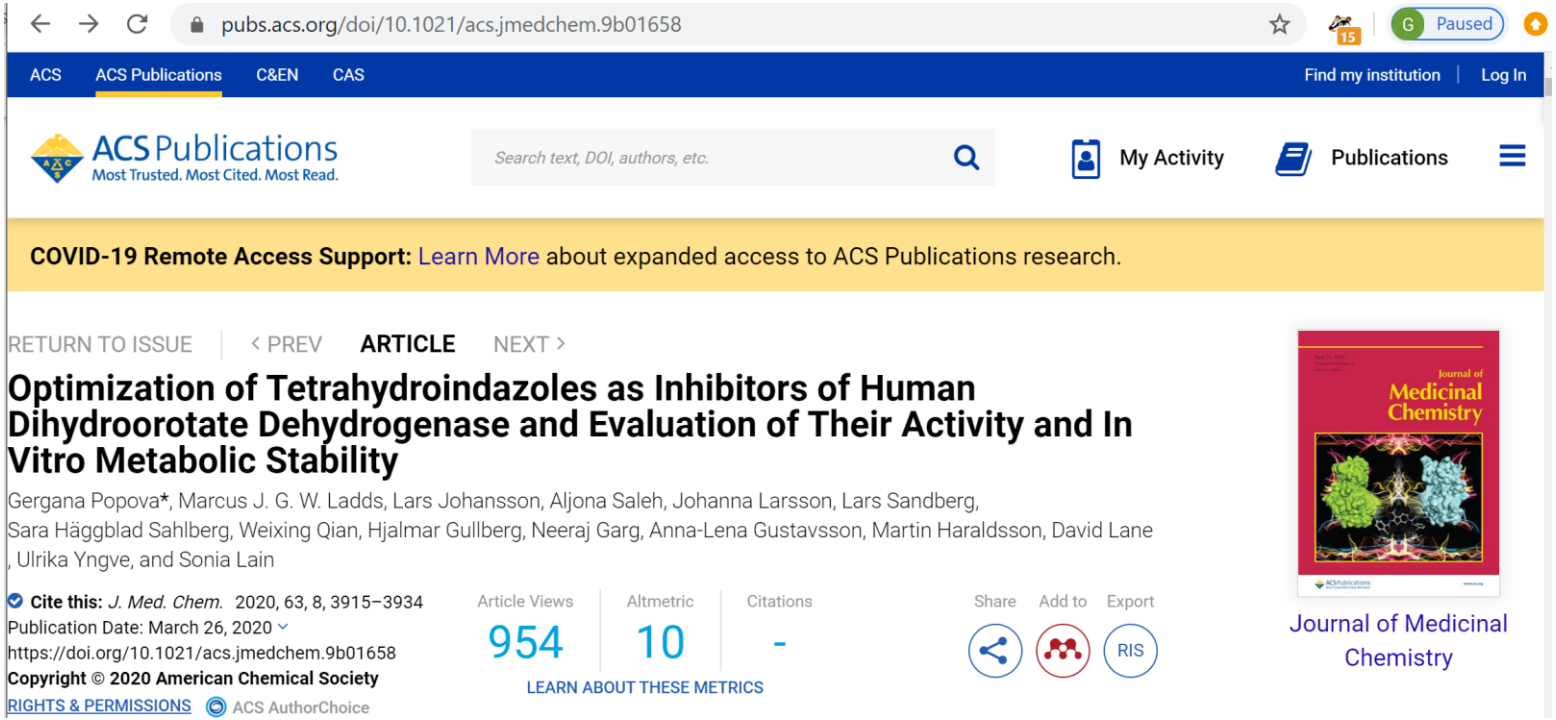
Hierarchical Cluster Assigner







The Data



Note: the paper is open access

https://doi.org/10.1021/acs.jmedchem.9b01658

Similarity search

Today’s goal

- Expand our results by finding building blocks similar to those we selected in the previous exercise (03_Clustering)

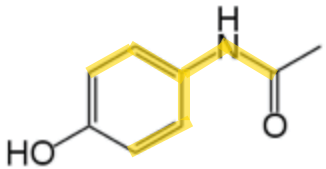
- The catalog of building blocks is taken from: https://zinc15.docking.org/catalogs/enaminebbe/substances/

- First calculate the fingerprints from the compounds and the building blocks. To ensure that the settings for this calculation are the same we will create and use a **shared component**. Then we perform a similarity search and end with a final selection and annotation step

Agenda for Today

- Similarity search
- Shared components
- Configuration nodes
- Search for functional groups
- Annotations in interactive view

Tanimoto Similarity

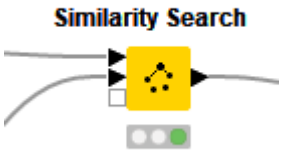


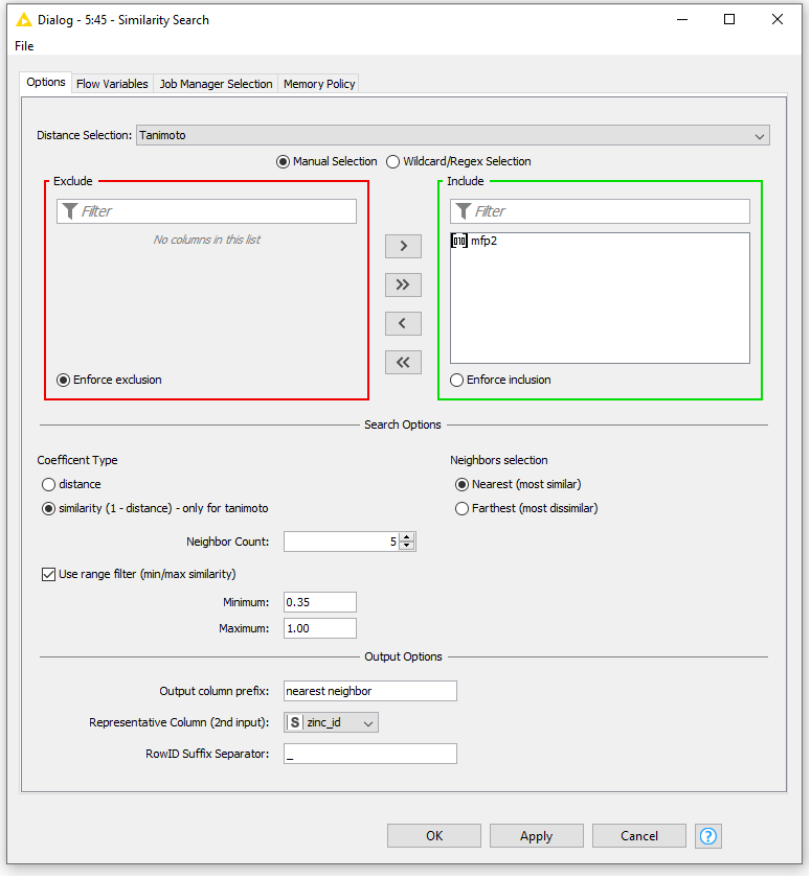
[00000**1**0000000000000001000…]

$$\text{Tanimoto}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

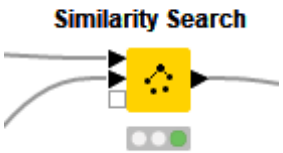
[0,1]
0 – no similarity
1 – “identical”

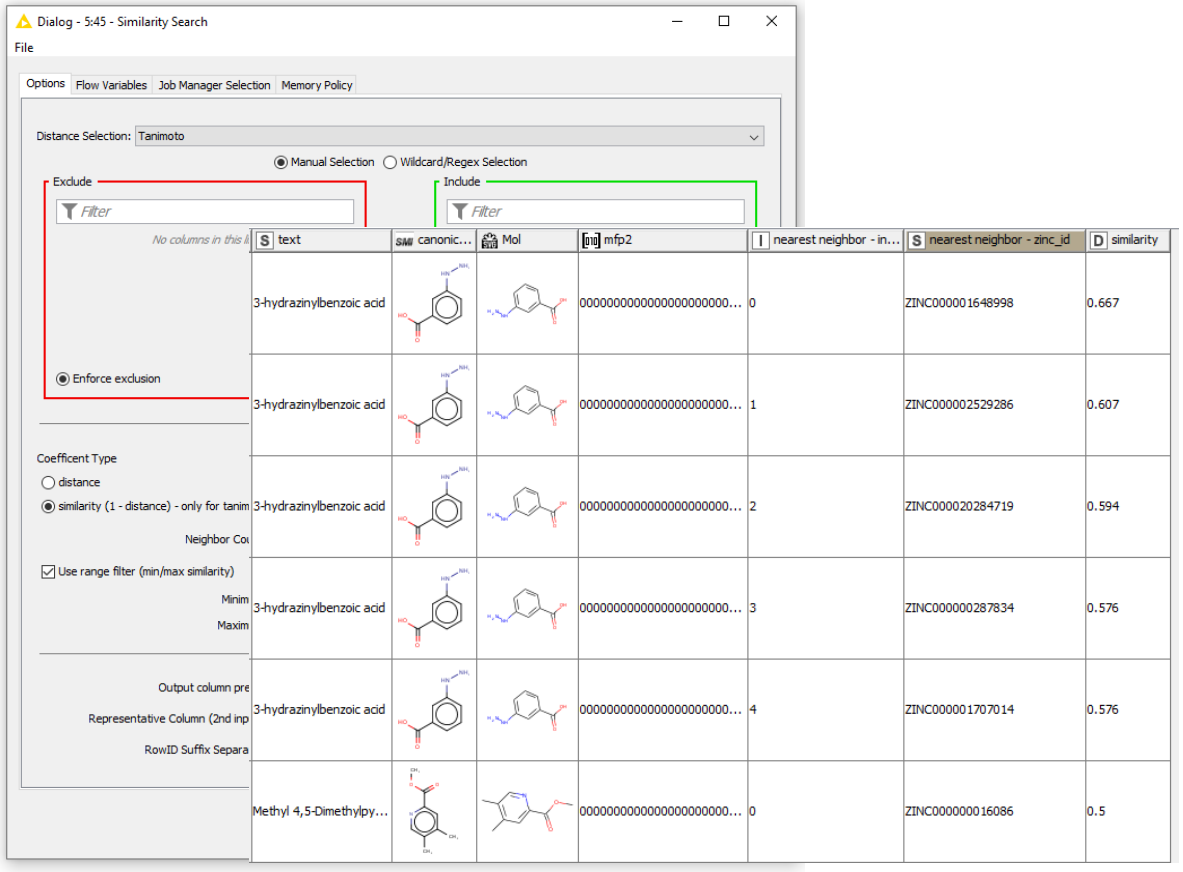
Similarity Search



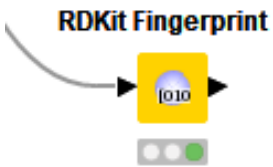


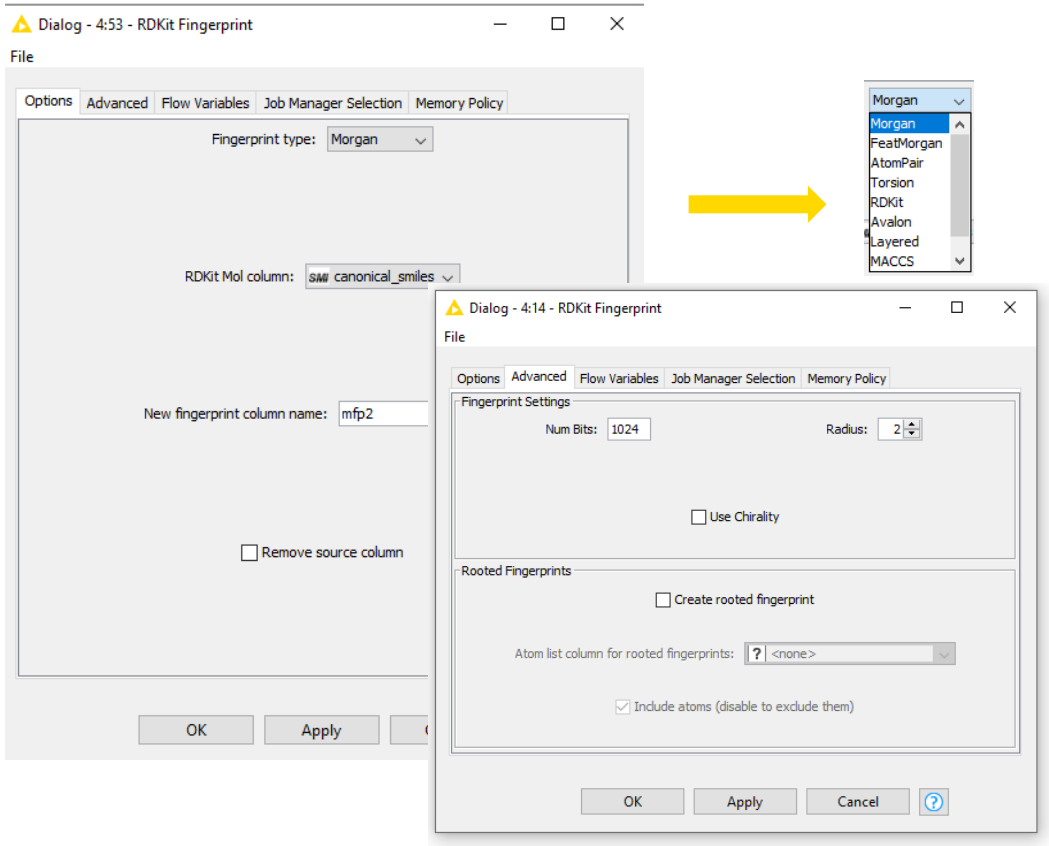
Similarity Search



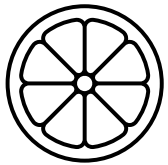


RDKit Fingerprints

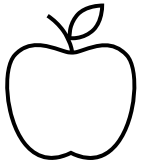


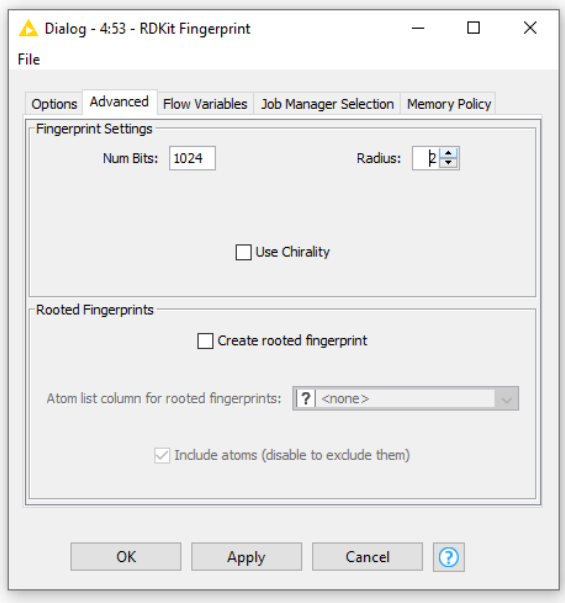


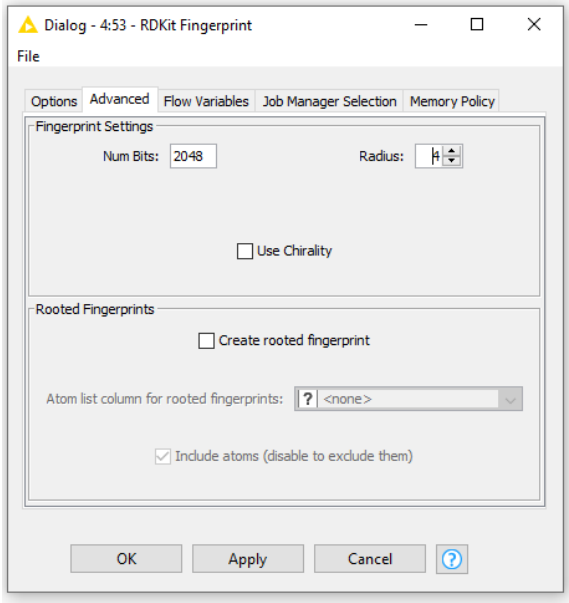
Similarity Search with Fingerprints



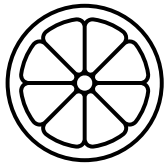
=



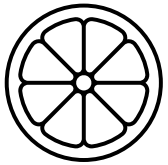




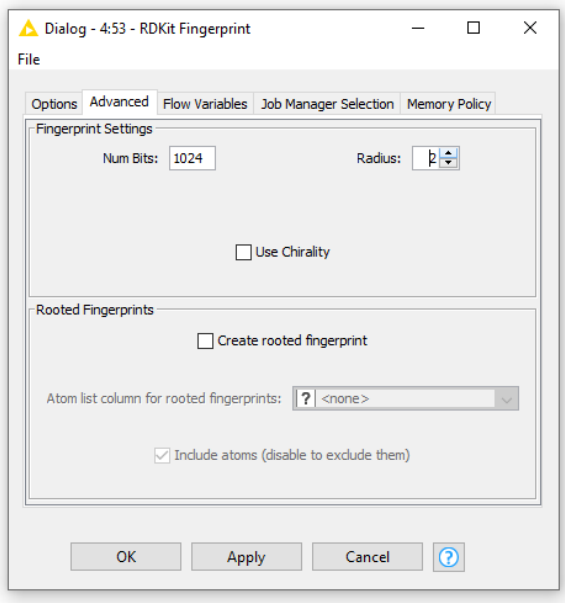
Similarity Search with Fingerprints

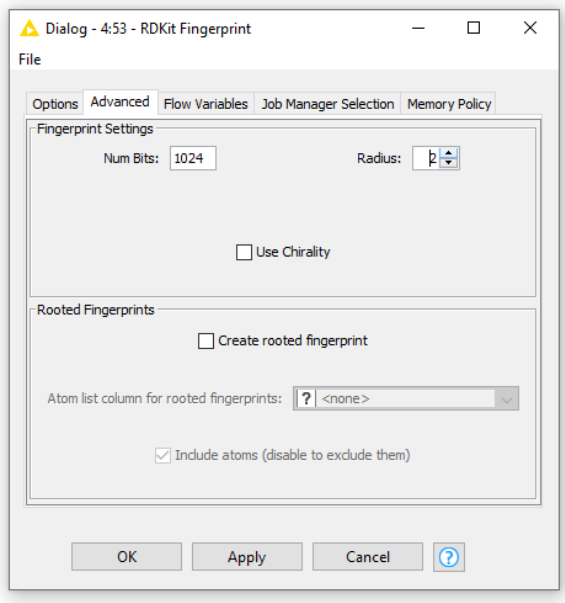


=



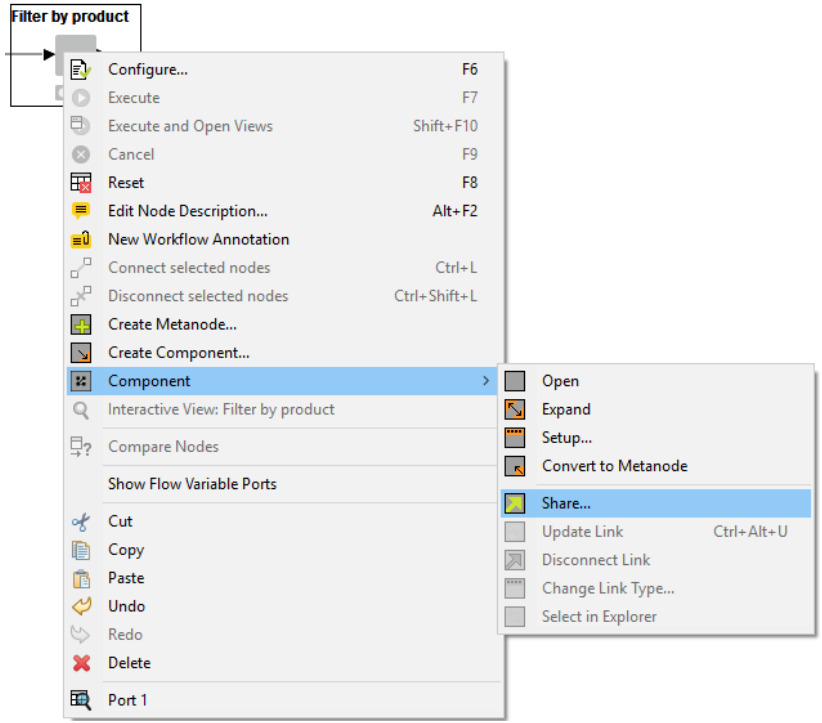
Reusable
Shared
Components!





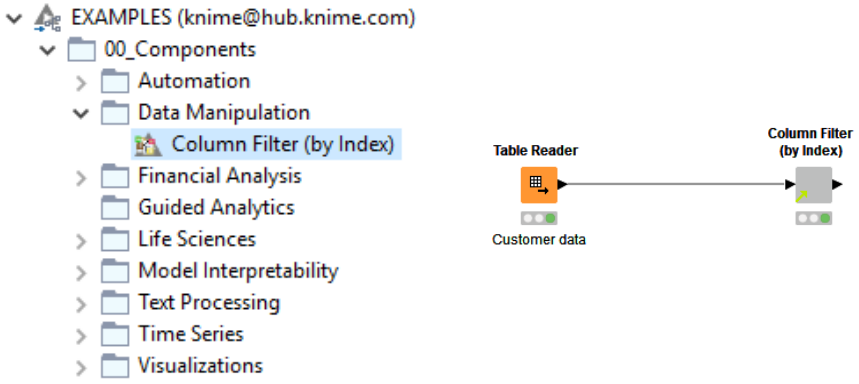
Shared Components

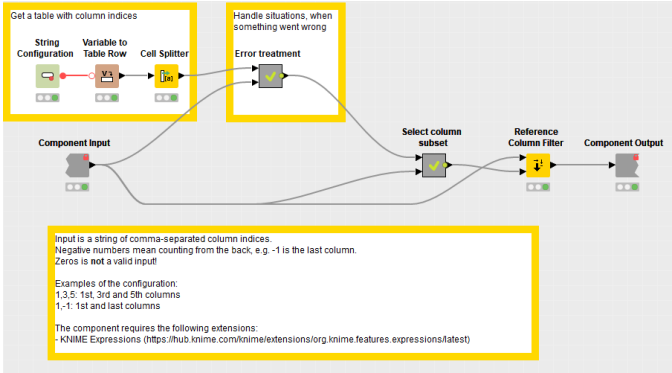
- Shared Components are read-only instances of a Component
- Components can be saved in your KNIME workspace, KNIME Server, or the KNIME Hub for later reuse
- To do this, simply right-click any Component and select “Share…”



Shared Components

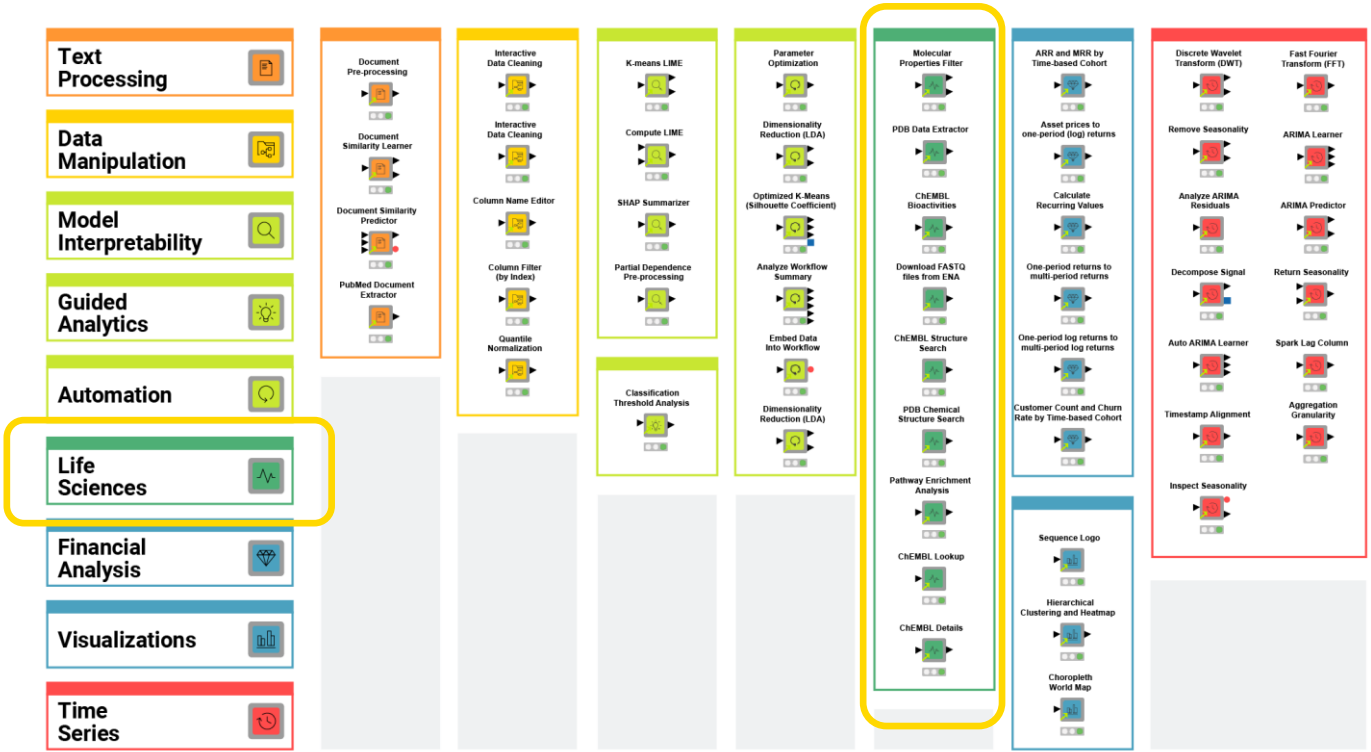
- To use a Shared Component, drag and drop it to the workflow editor
- Instances of Shared Components can be updated either manually or when workflow is opened
- Shared Component can also be unlinked from its original location, which makes it editable in the workflow directly
- Update Shared Components by overwriting them



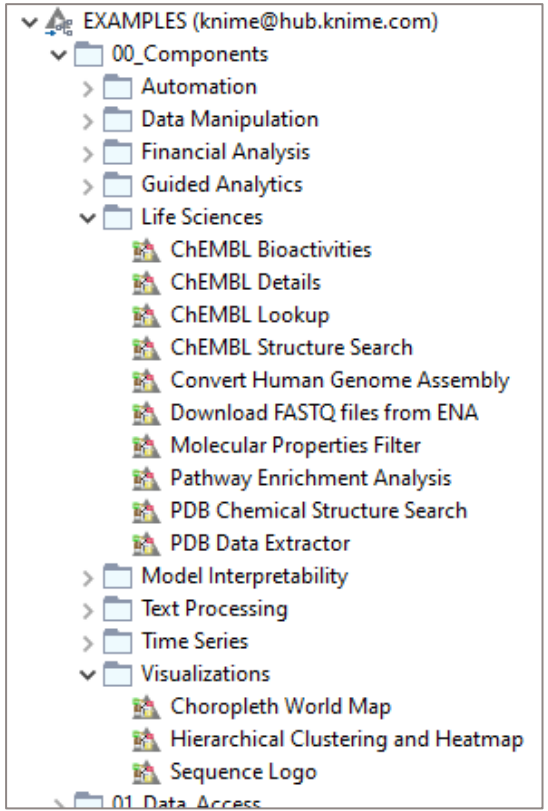


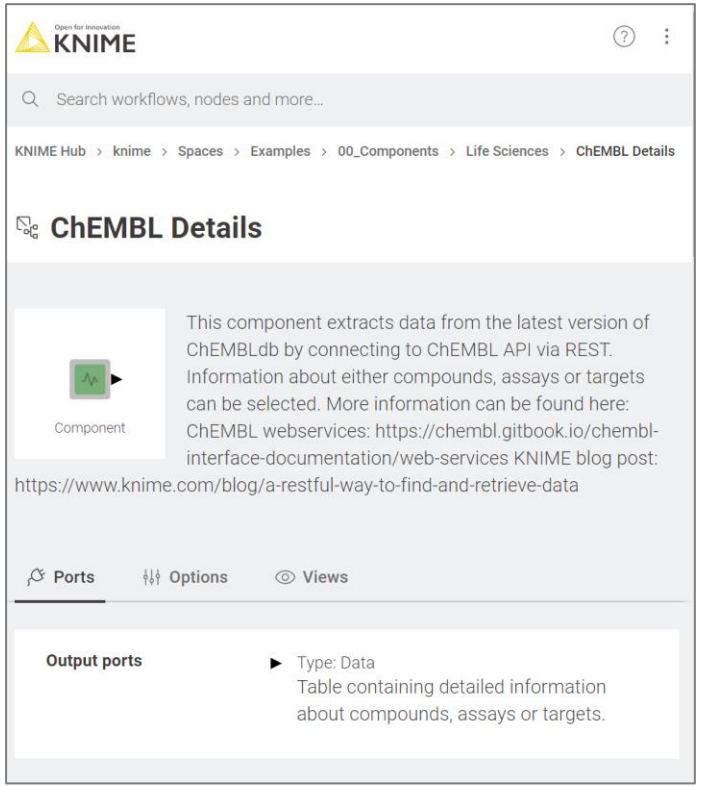
KNIME Verified Components

- Verified Components reuse bundled functionalities, verified by KNIME experts
- Released and updated on the KNIME Hub



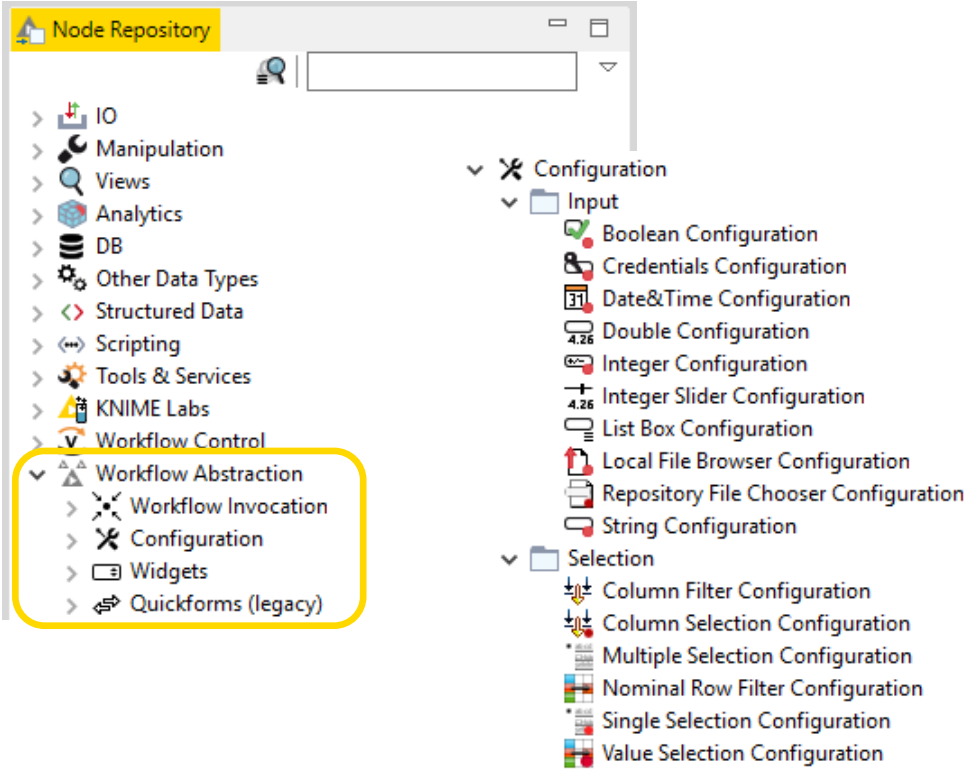
Publicly Shared Components

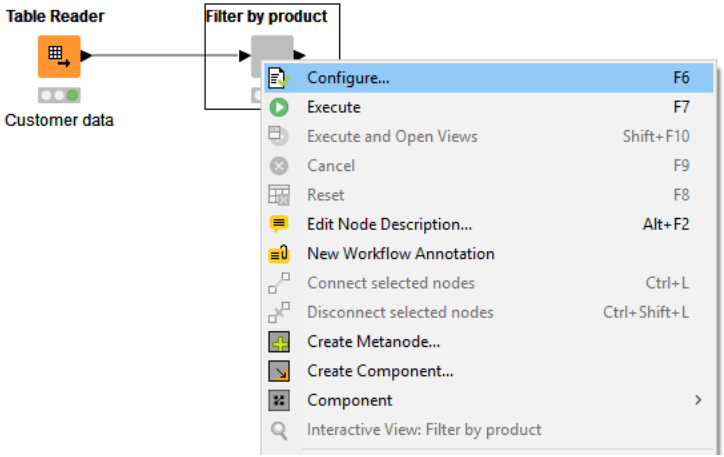




https://www.knime.com/blog/knime-analytics-platform-40-components-are-for-sharing

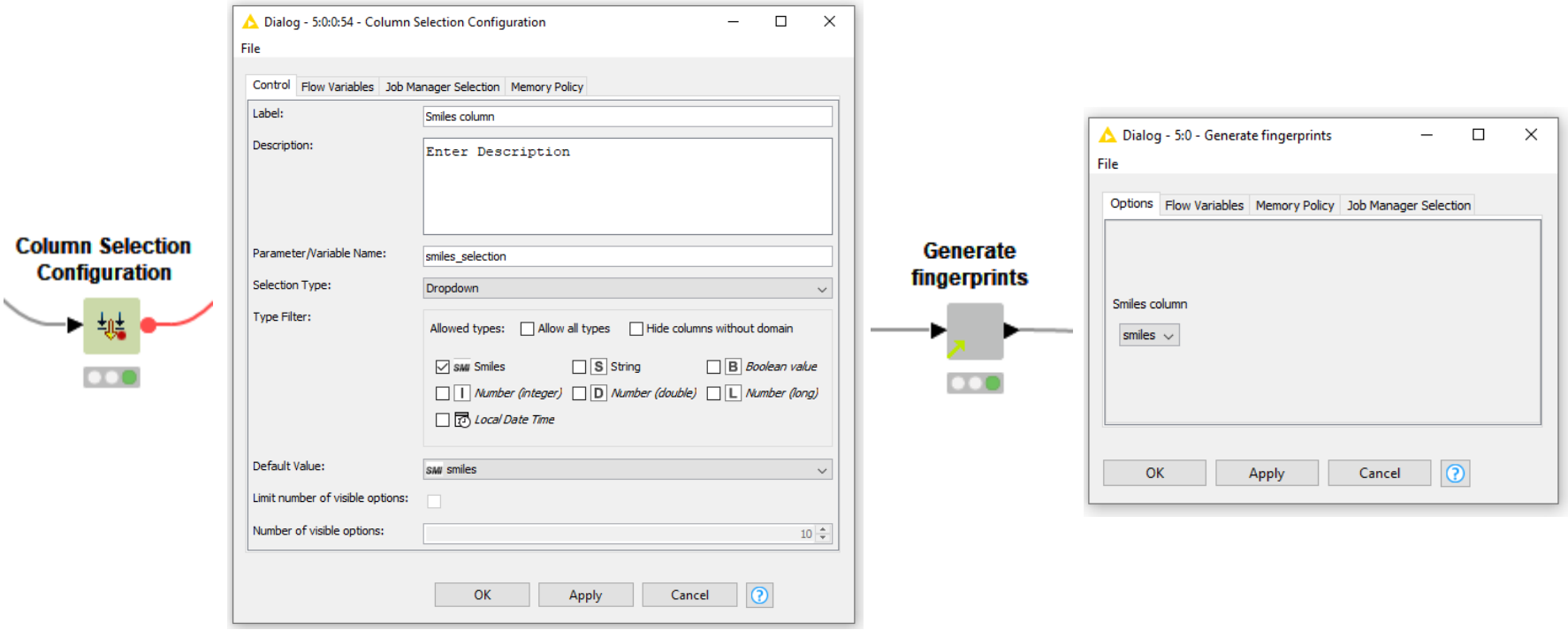
Configuration Nodes





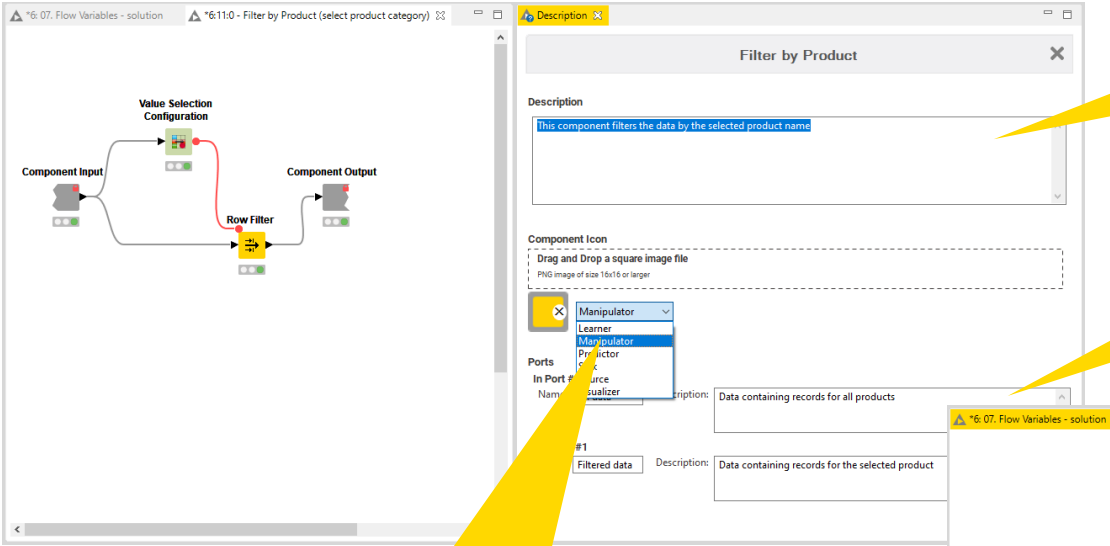
- Double click a component to configure

Column Selection Configuration



Component Description

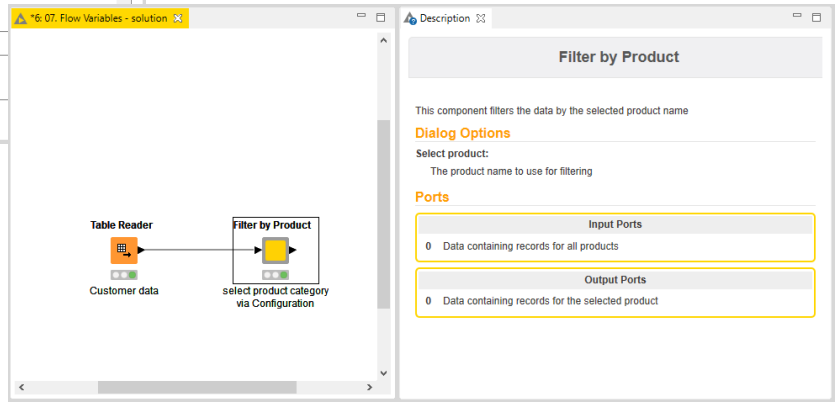
Make your component look like a KNIME node



Add description of the component

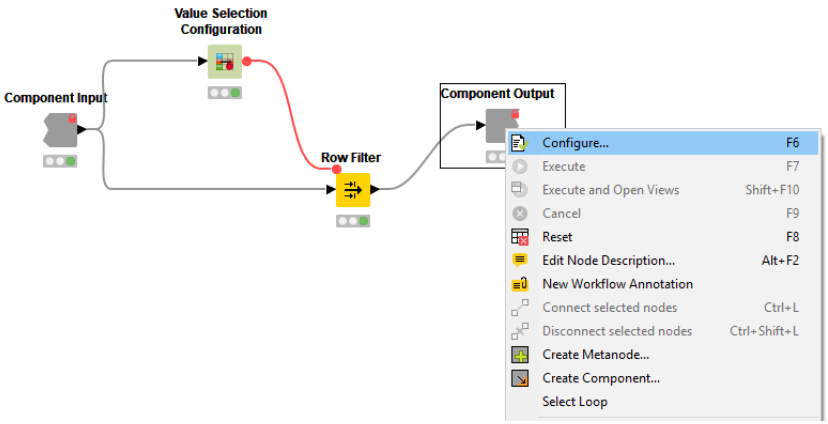
Add description of the input and output ports

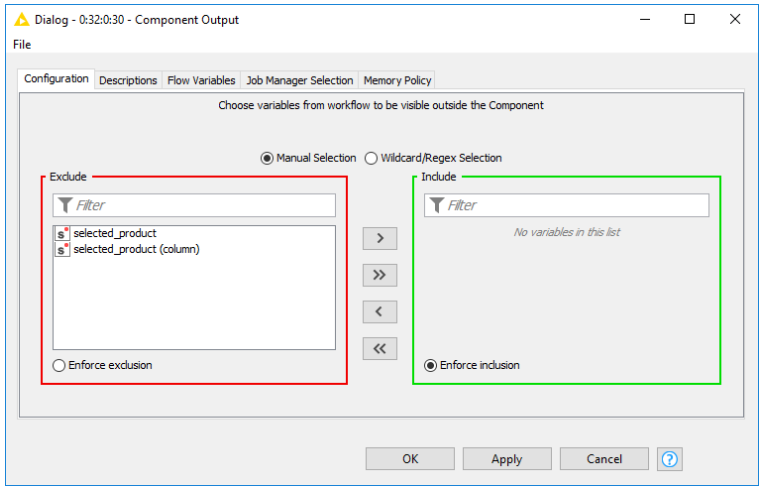
Add background color or icon



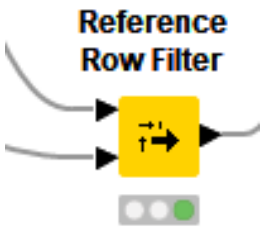
Passing Variables into/from Components

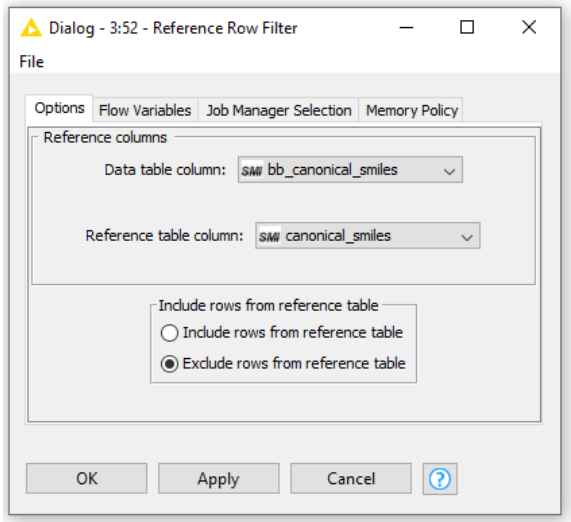
- Flow Variables by default only available locally inside Component
- Configure Component Input/Output to pass Flow Variables from/to outside Component





Reference Row Filter





Ports

Input Ports	
0	Table from which rows are to be included or excluded
1	Table rows used as reference filter

Output Ports	
0	Table with filtered rows

Search for Functional Groups

- Splits molecules matching and failing the filters

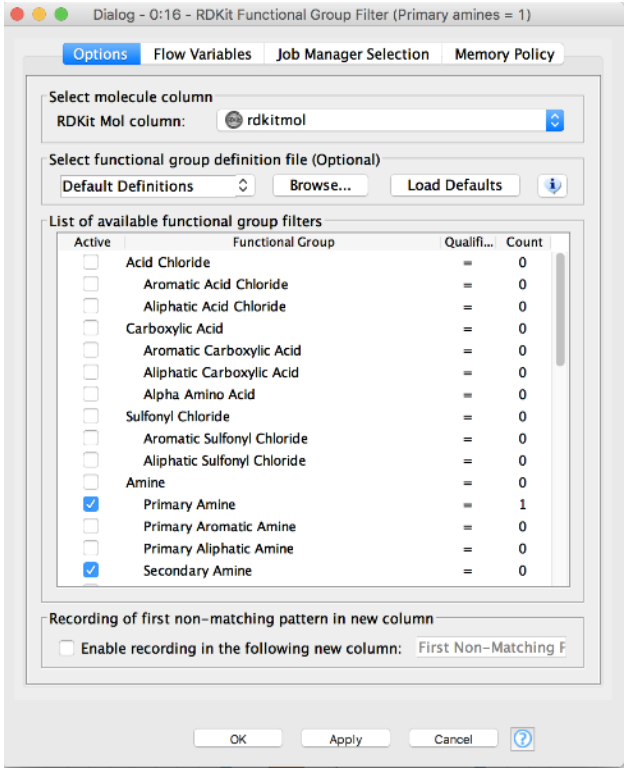
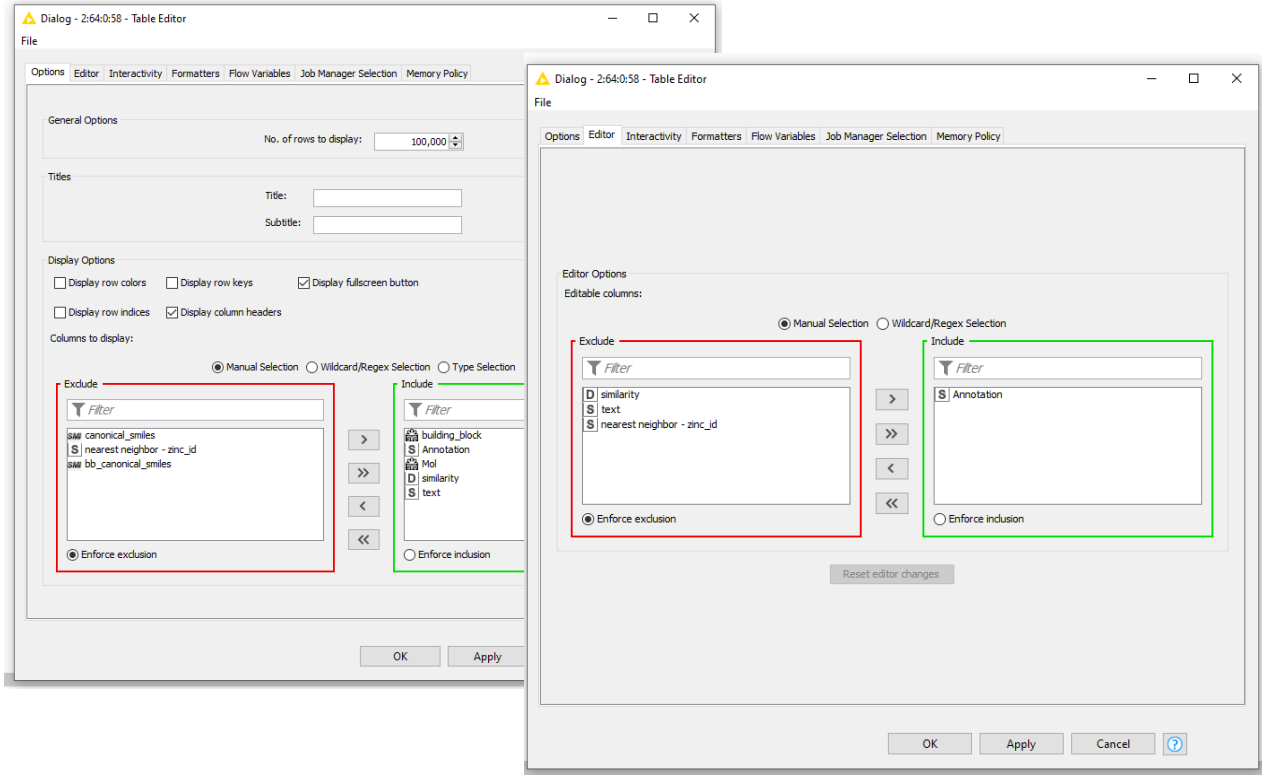


Table Editor

- Interactive table view and possibility to edit



04_SimilaritySearch

- Create shared component with configuration to calculate fingerprints
- Perform similarity search for compounds and building blocks
- Create an interactive view for final selection and annotation

Confirmation of Attendance and Survey

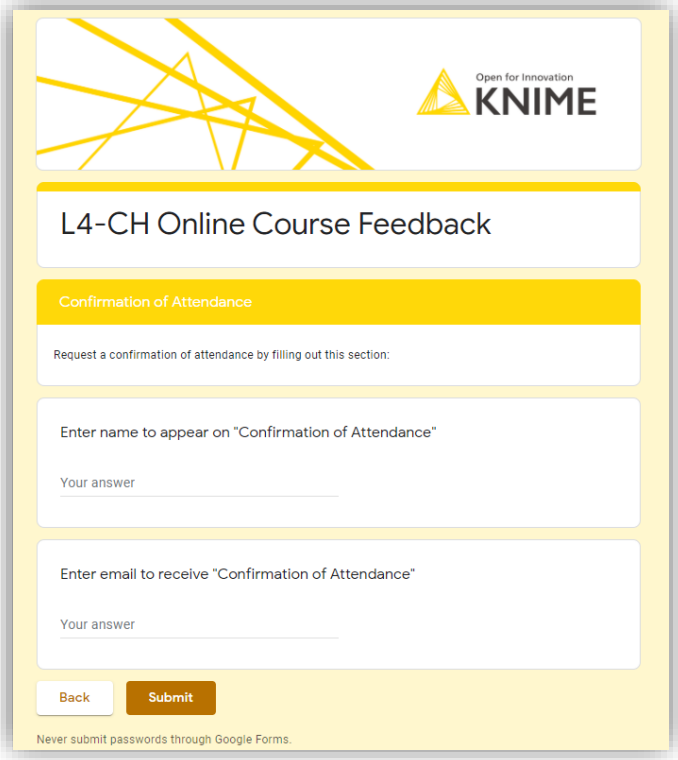
- If you would like to get a “Confirmation of Attendance” please click on the link below*

LINK

- The link also takes you to our course feedback survey. Filling it in is optional but highly appreciated!

Thank you! ☺

*Please send your request within 3 days

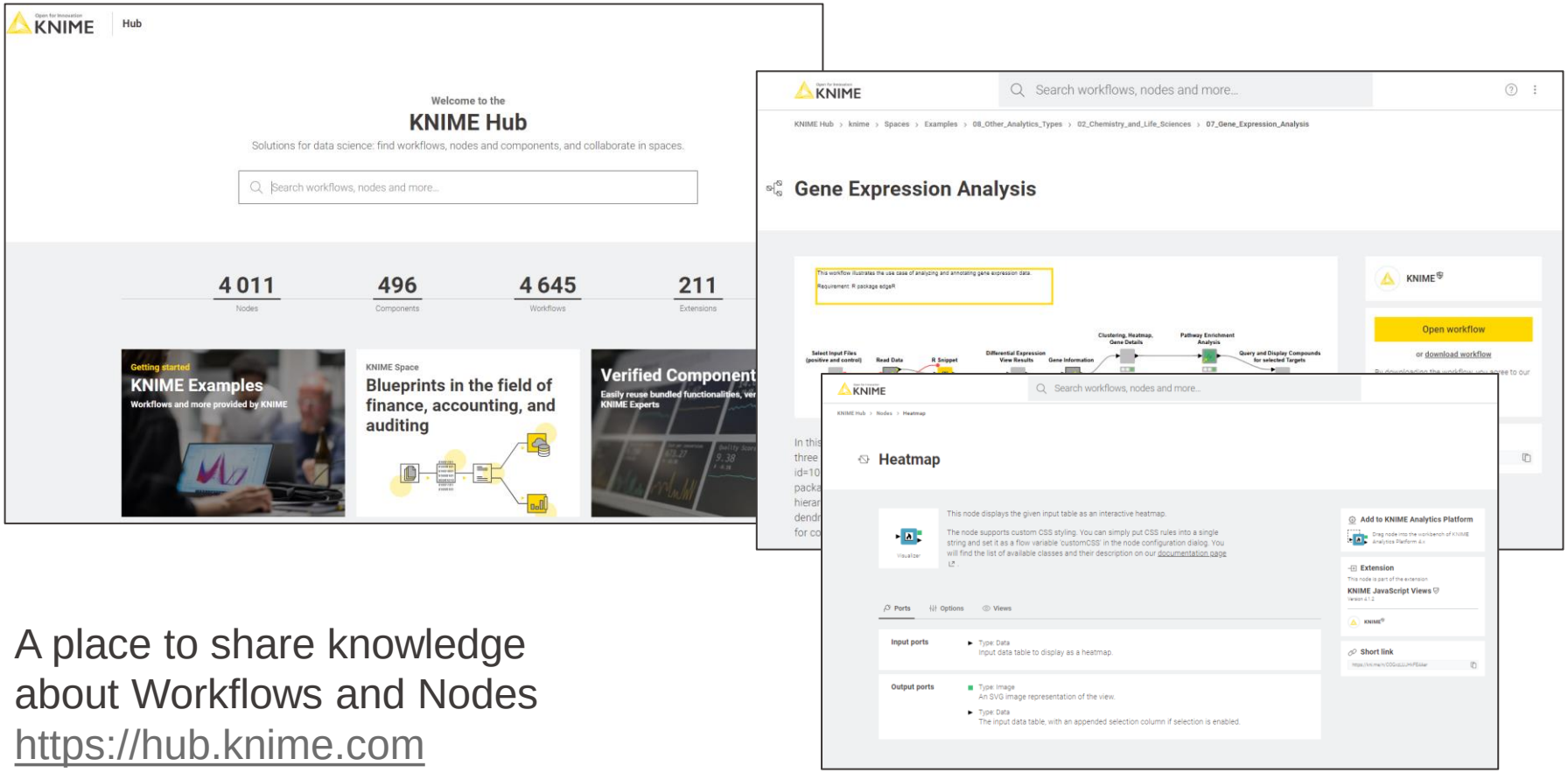


End and Backup

Hot Keys (for Future Reference)

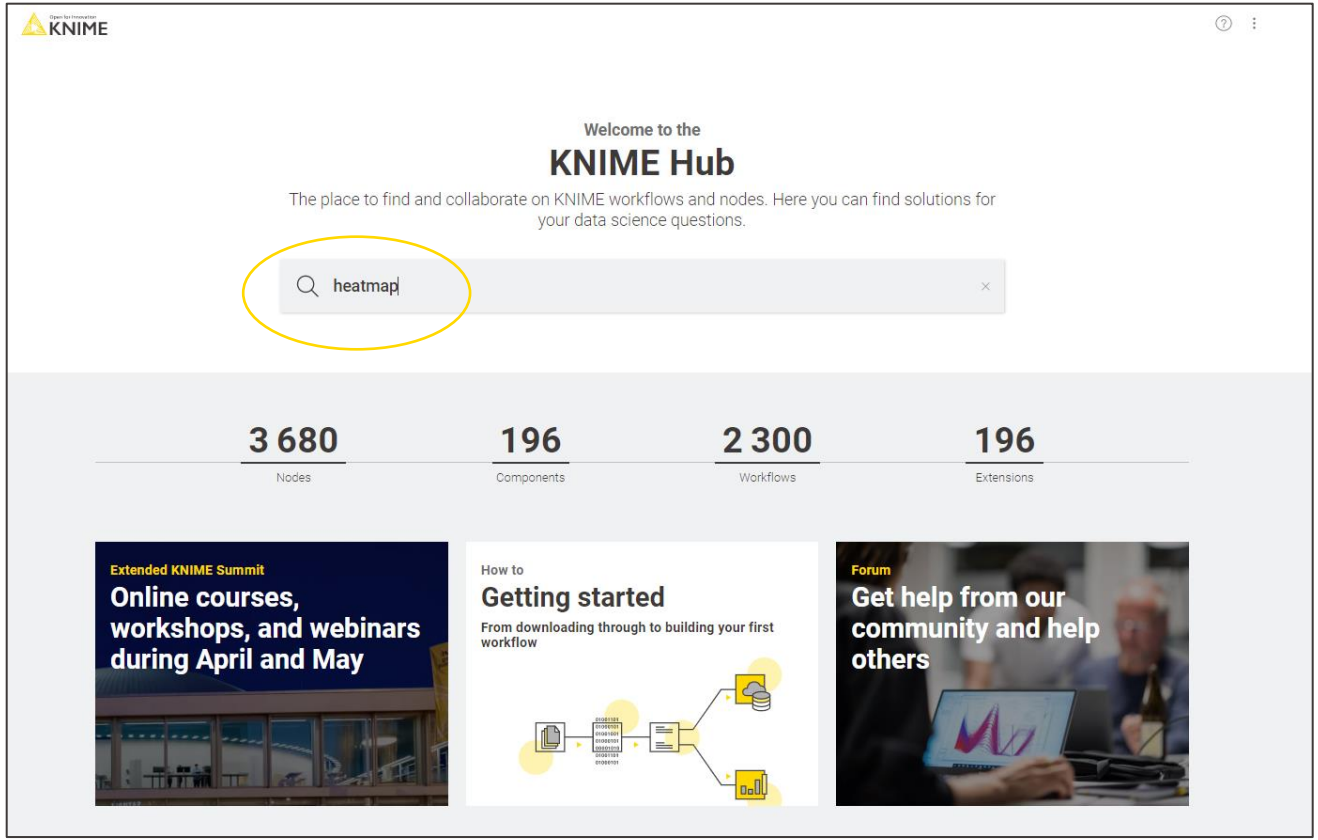
Task	Hot key	Description
Node Configuration	F6	opens the configuration window of the selected node
Node Execution	F7	executes selected configured nodes
	Shift + F7	executes all configured nodes
	Shift + F10	executes all configured nodes and opens all views
	F9	cancels selected running nodes
	Shift + F9	cancels all running nodes
Node Connections	Ctrl + L	connects selected nodes
	Ctrl + Shift + L	disconnects selected nodes
Move Nodes and Annotations	Ctrl + Shift + Arrow	moves the selected node in the arrow direction
	Ctrl + Shift + PgUp/PgDown	moves the selected annotation in the front or in the back of all overlapping annotations
Workflow Operations	F8	resets selected nodes
	Ctrl + S	saves the workflow
	Ctrl + Shift + S	saves all open workflows
	Ctrl + Shift + W	closes all open workflows
	Ctrl + F	search workflow for nodes
Metanode	Shift + F12	opens metanode wizard

KNIME Community Hub

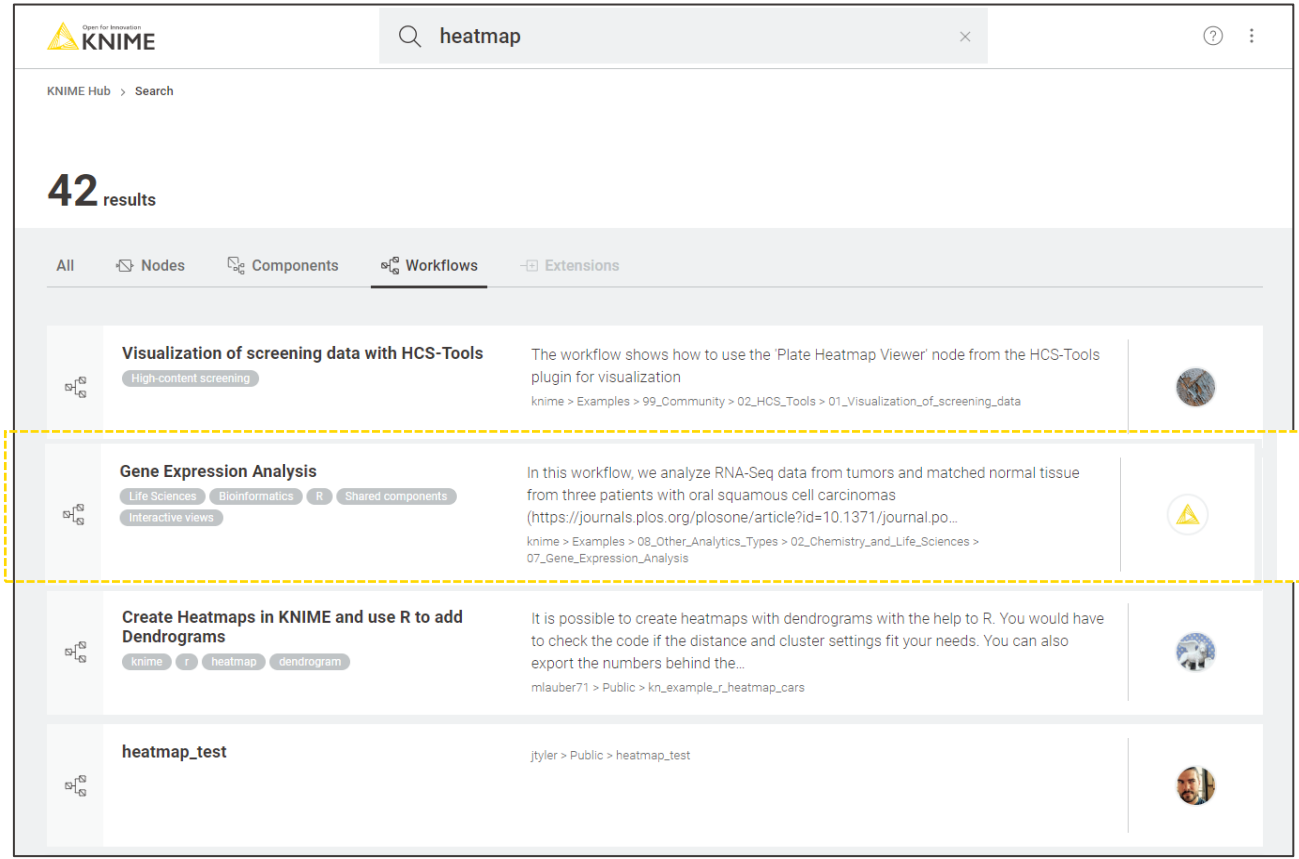


A place to share knowledge about Workflows and Nodes
https://hub.knime.com

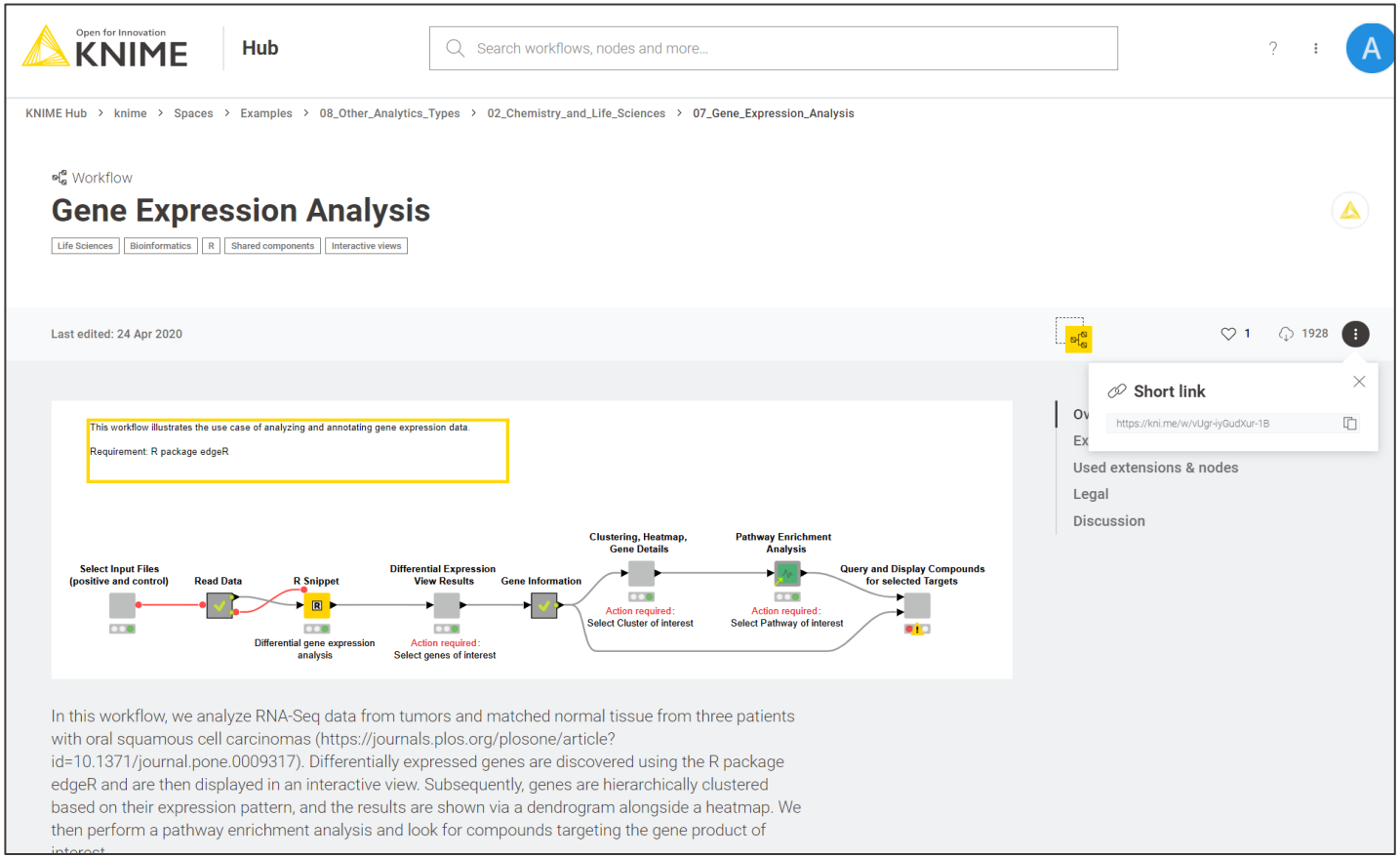
The KNIME Community Hub



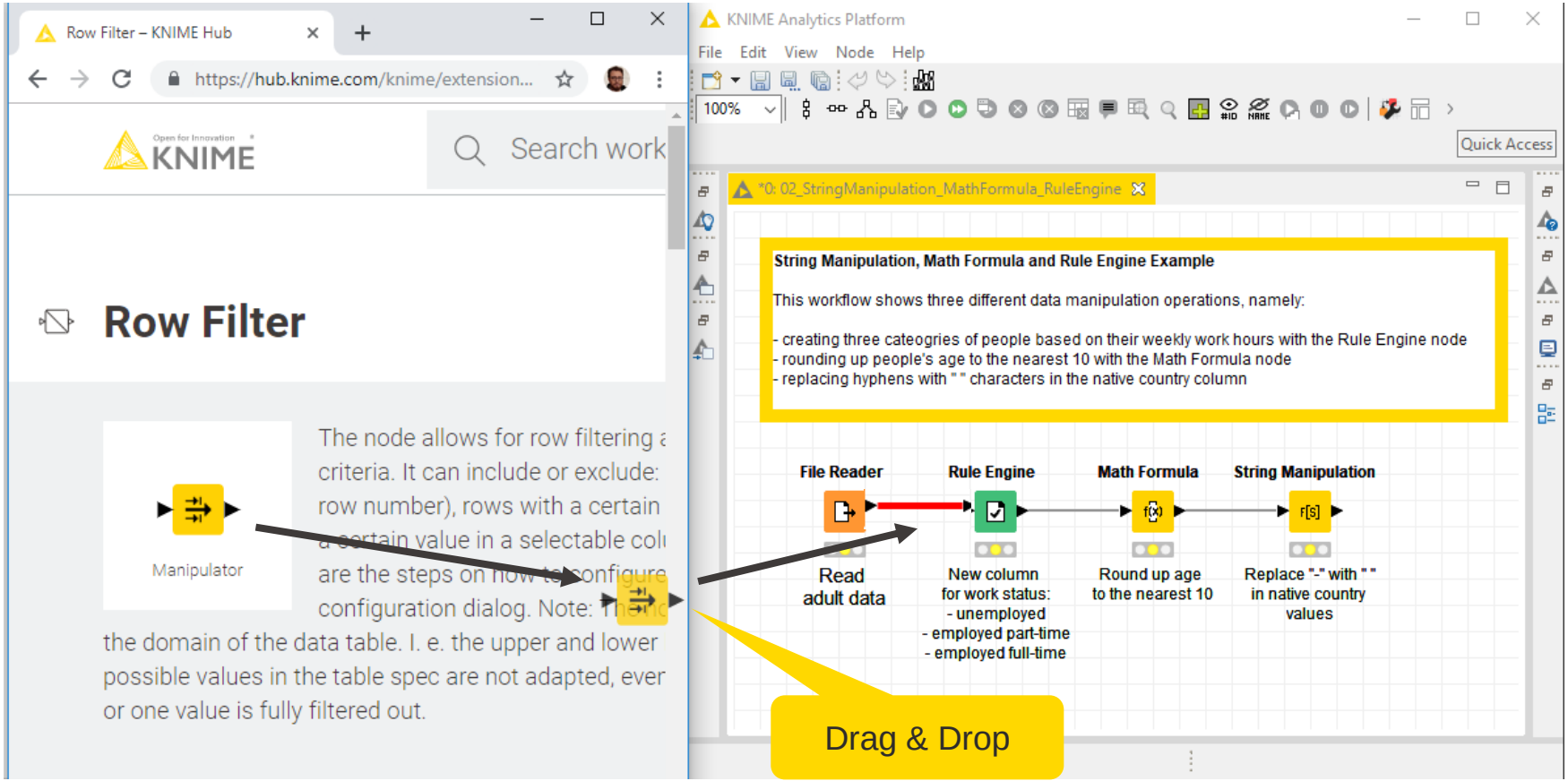
Searching Nodes and Workflows



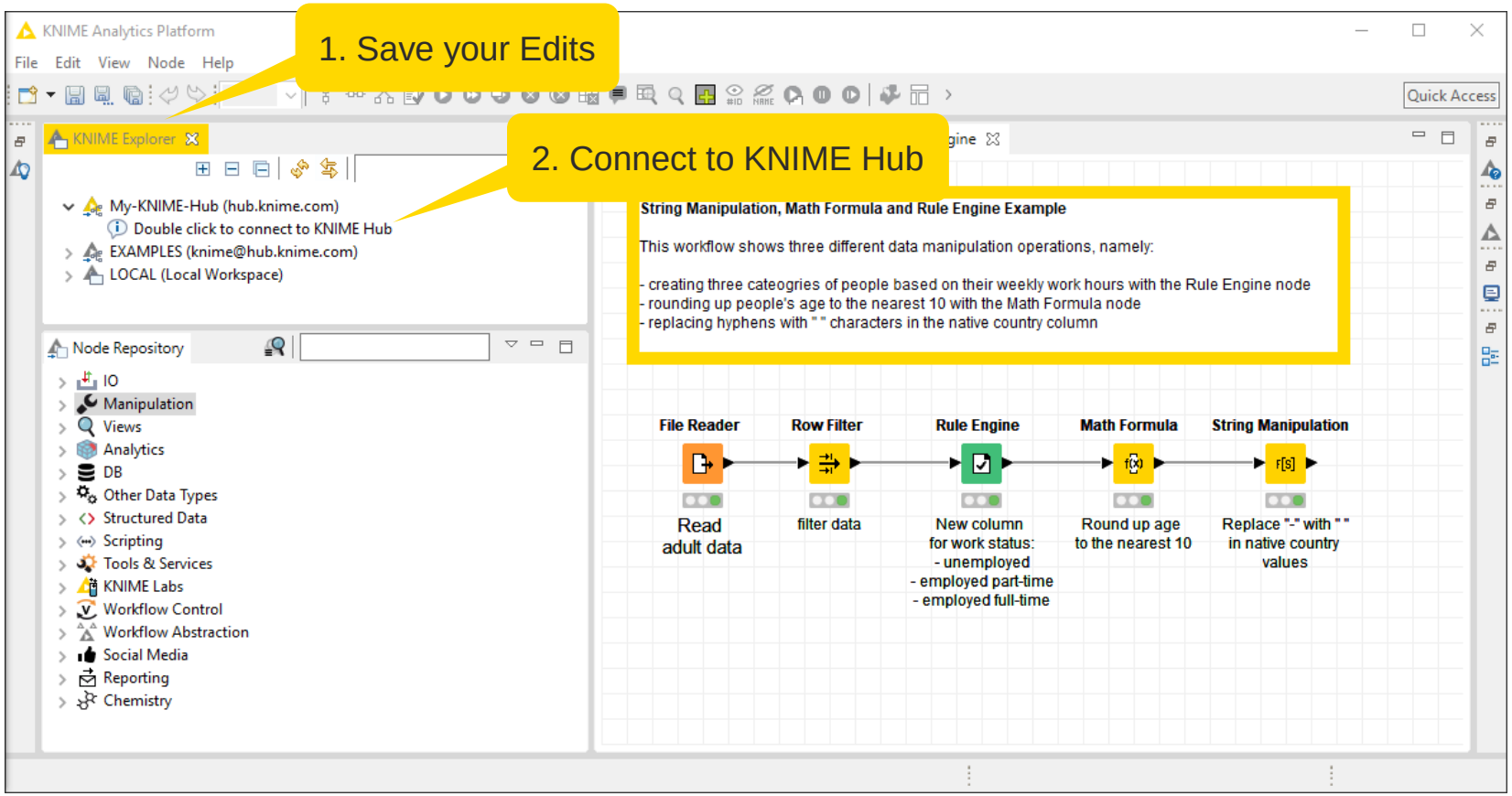
Opening a Workflow from the Hub



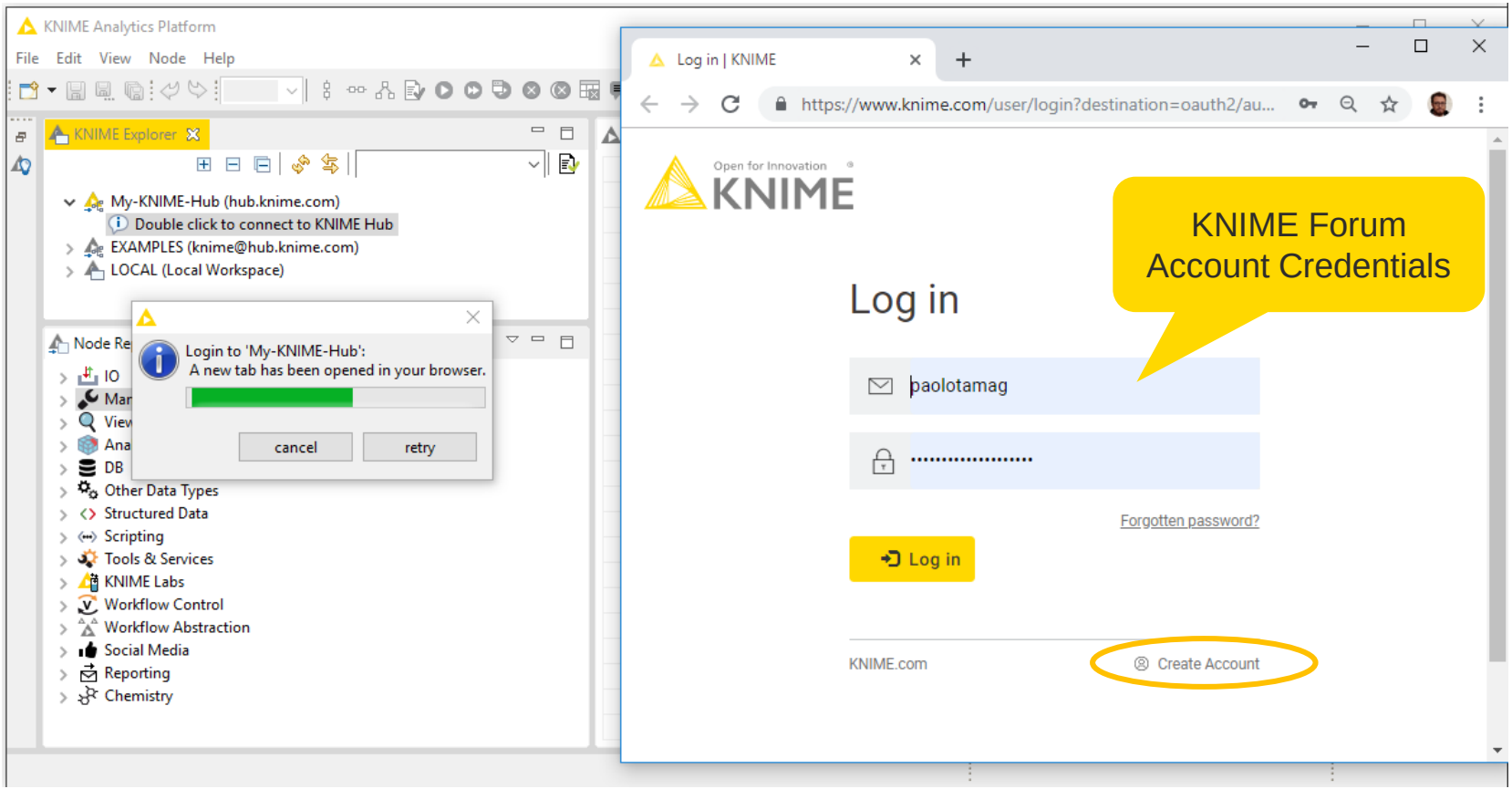
Edit the Workflow



Sharing the Workflow on the Hub



Log in the Hub



Stay connected with KNIME

Blog: knime.com/blog

Forum: forum.knime.com

KNIME Hub: hub.knime.com

KNIME E-Learning Course:
www.knime.com/e-learning-course

Follow us on social media:

Resources

- Self Paced E-Learning Data Science Courses
  - https://www.knime.com/knime-self-paced-courses
- KNIME certification
  - https://www.knime.com/certification-program
- KNIME Documentation
  - https://www.docs.knime.com
- YouTube KNIME TV
  - https://www.youtube.com/user/KNIMETV
- KNIME Events and instructor-lead courses
  - https://www.knime.com/learning/events
- KNIME Cheat Sheets
  - https://www.knime.com/learning/cheatsheets
- KNIME Books
  - https://www.knime.com/knimepress

Thank You!

Questions? Please reach out

alice.krebs@knime.com

johannes.judt@knime.com