1

Data Curation

Gilles Marcou and Alexandre Varnek

Goal: Identify and curate problematic chemical information from a data collection. The raw dataset is processed so that it will be ready to feed a relational database dedicated to the organoleptic properties of small organic molecules. Information is interpreted and re-encoded as categories or bit vectors when relevant.
Software: KNIME 3.0, ChemAxon
Data: The following files are provided in the tutorial:

- `thegoodscent_dup.csv` – The raw data formatted in a semicolon separated file extracted from the web site of The Good Scent Company. The data is prepared and the most visible errors and discrepancies are already corrected.
- `thegoodscent_dup.raw` – The raw data without any processing related to the tutorial.
- `MissingOdorTypes.csv` – Manually curated Odor Types provided for some difficult cases.
- `StructureCuration.csv` – File containing the curation rules for some deficient SMILES of the input.
- `TutoDataCuration.zip` – The final KNIME workflow. Unzip the archive in the KNIME workspace and it will appear in your LOCAL workflows.
- `Slurp.pl` – A Perl script exploring the website of The Good Scents Company in search of some chemical information.

The Good Scent Company is an online shop providing cosmetic, flavor, and fragrance ingredients. It provides information for the flavor, food, and fragrance industry since 1994, and sales ingredients since 1980.

Theoretical Background

Chemical datasets can be collected from literature, compendiums, web sites, lab-books, databases, and so on. Aggregation and automatic treatment of data represent additional sources of errors. Therefore, verification of quality and accuracy of chemical information is a crucial step of data valorization.[1]

Tutorials in Chemoinformatics, First Edition. Edited by Alexandre Varnek.

Companion website: www.wiley.com/go/varnek/chemoinformatics

The problem of the quality of publicly available chemical data can be illustrated on the searching the Web for the chemical structure of antibacterial compound Vancomycine, for which stereochemistry information is essential. One can suggest two possible queries using InChIKey notations:[2,3]

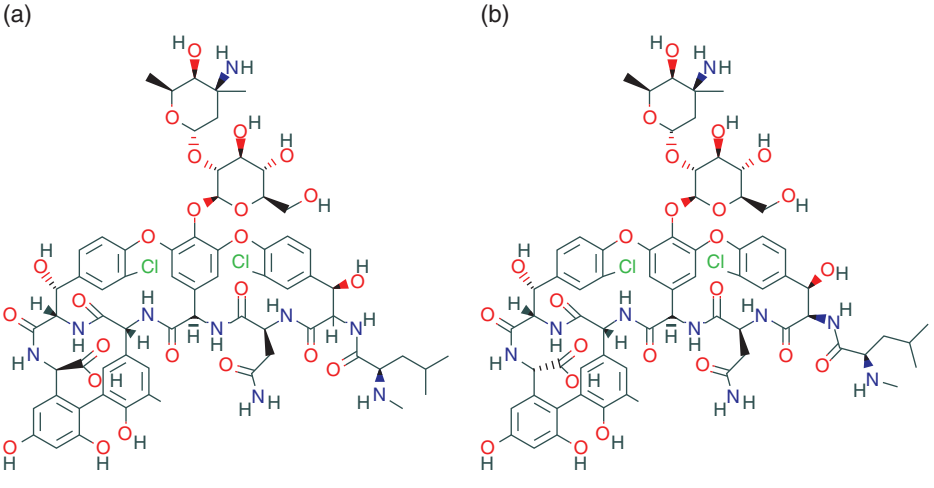
Query 1: "MYPYJXKWCTUITO" "Vancomycine"
Query 2: "MYPYJXKWCTUITO-LYRMYLQWSA-N" "Vancomycine"

Query 1 corresponds to the first layer of the InChI code of Vancomycine; it encodes only elemental constitution and atoms connectivity, whereas *Query 2* includes detailed stereochemistry information.

A search on Google (29/01/2016) retrieves 82 and 71 entries for *Queries 1* and *2*, respectively. Entries found with *Query 2* correspond to the correct chemical structure of Vancomycine, whereas all 11 additional entries retrieved with *Query 1* refer to its different enantiomers, see example on Scheme 1.1.

From this example, one can see that an estimate of the erroneous data associating Vancomycine to the wrong chemical structure is about 13%. Analysis of some 6800 publications in drug discovery[4] show that the average error rate of reported chemical structures is about 8% and, it seems, nothing has changed so far. Numerous examples and alerts about data curation problems, especially in public databases, can be found in the literature.[4–8]

In this tutorial, a dataset regarding organoleptic properties of cosmetic related chemicals was collected from the website http://thegoodscentscompany.com/(January 2016). The dataset contains eight records: the name of the chemical substance, the CAS number, an odor category and description, the source of the odor description, a taste description, the literature for the source of the taste description, and the SMILES encoding the chemical structure of the substance. The data were retrieved automatically



Scheme 1.1 Chemical structures of Vancomycine from PubChem. (a) PubChem CID 441141, InChIKey : MYPYJXKWCTUITO-UTHKAUQRSA-N. (b) PubChem CID 14969, InChIKey : MYPYJXKWCTUITO-LYRMYLQWSA-N. Notice that Vancomycine corresponds to structure (b), whereas structure (a) is, in fact, its enantiomer.

using a script provided with the tutorial (however, the script might need changes to work properly if the website has changed its structure in the meantime).

Each substance should be associated to exactly one organoleptic category, its odor type. Besides, some additional descriptions of the odor and tastes can be present. These textual descriptions are interpreted in terms of a dictionary of concepts used to describe the odors and tastes: the organoleptic semantic. With the help of this semantic each substance can be represented as a bit vector: each bit is related to an organoleptic descriptor. A bit is "on" if a particular description is relevant for the substance and it is "off" otherwise. Similarly, the chemical structures are interpreted in terms of MACCS fingerprints. In such a vector, a bit is "on" if the chemical structure of the substance possesses some feature (includes some element or chemical function for instance). Binary descriptions are suitable for further analysis, to compute distances or association rules.

Chemical structures and organoleptic descriptions, organoleptic category and bibliographic references are split into different files that can be loaded into separate tables and then merged into a relational database.

Software

KNIME is an Integrated Development Environment (IDE) and a workflow-programing language. Processing units, called nodes, are connected to each other. Data is directed from one node to another following the connections between them. By default, KNIME is divided into eight zones (Figure 1.1). The first one (**1**) is the *toolbar* of buttons for quick shortcuts. These buttons include creating a new project, saving the current projects, zooming and automatic cleanup of the workbench, running and managing the workflow. The second (**2**) area is the *workbench*, the place to drag and drop the nodes and to connect them in order to design a workflow. A miniature of the workbench is provided inside the sixth area (**6**), the *Outline*, in order to help navigating the workflow. The third area (**3**) it the *KNIME Explorer*, a storage area for workflows: it is divided by default into LOCAL and EXAMPLES. The EXAMPLES require an Internet connection to connect with a public KNIME server (login as guest, no password) where is found useful KNIME examples implementing solutions for many basic and advanced operations. The fourth (**4**) area is the *Node Repository*; this is the place where all nodes, representing data processing operations, are stored. Nodes are organized in a tree and a navigation bar provides a node search tool. The most frequently used nodes and the annotated ones are available inside the seventh area (**7**), the *Favorite Nodes*. The fifth area (**5**) is the *Node Description*. When a node is selected, it displays the help text describing the purpose of the node, its parameters, and the format of input and output. The eighth area (**8**) is the *Console* where errors and warning messages are displayed.

When KNIME is activated the first time, it requests a directory to use as workspace. This workspace is used to store temporary files and the workflows. The location and name of the workspace is up to the choice of the user. This choice can be changed later in the **Preferences** menu of KNIME.

Using KNIME consists in manipulating the following basic concepts:

- Drag and drop a node from the node repository into the workbench to use it.
- A node (Figure 1.2) has a main title describing its purpose, a traffic light describing the state of a node, and a custom name. On the side of a node are located handles. The left handles are input and right handles are output.

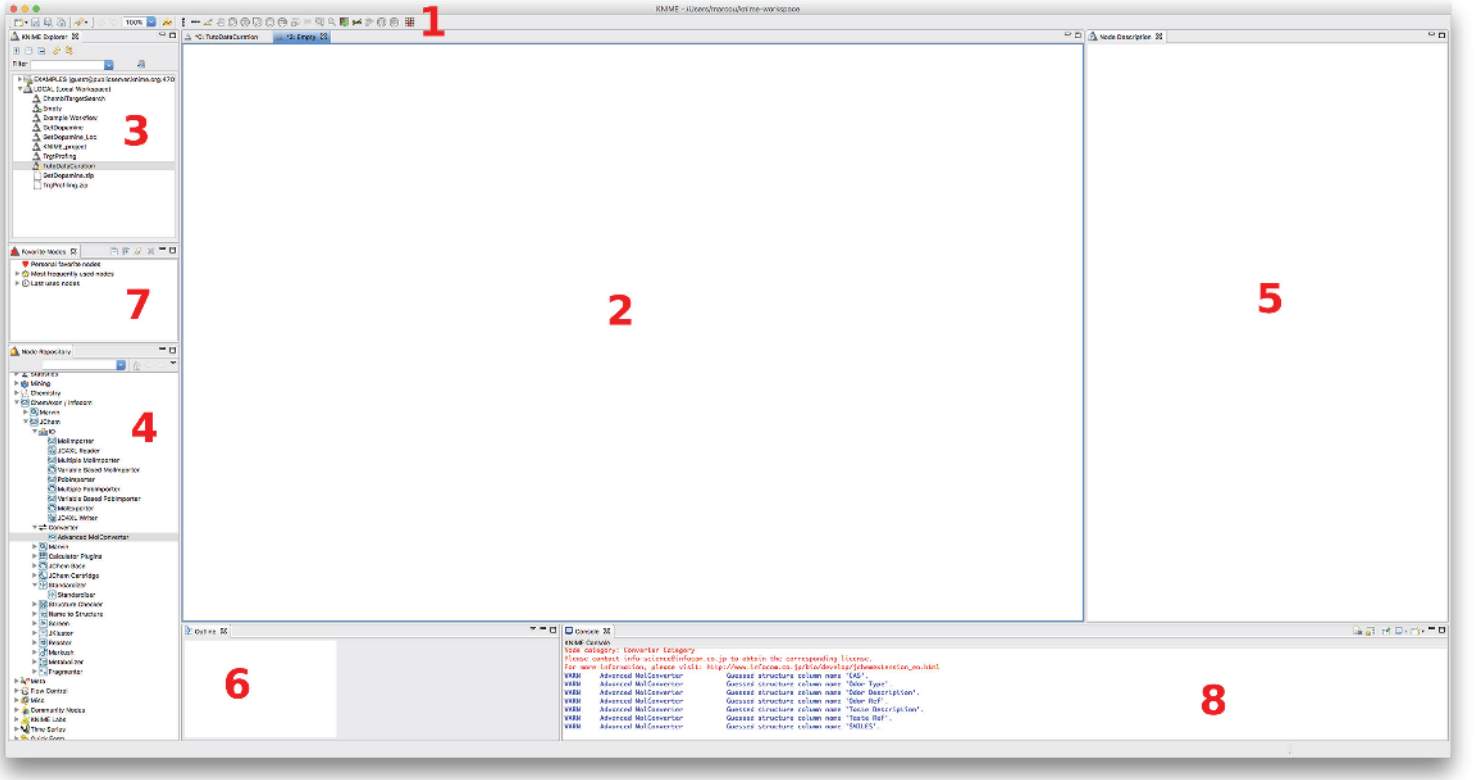


Figure 1.1 KNIME Overview. The interface is organized as follows: (1) the toolbar, (2) the workbench, (3) the KNIME Explorer, (4) the Node Repository, (5) the Node Description, (6) the Outline, (7) the Favorite Nodes, (8) the Console.

- The traffic light is red if the node is not configured, orange if the node is ready, green if the node was successful in processing the data. It is modified if the node generated an error or a warning.
- Click on a right handle of a node, pull and release the mouse button on a left handle triangle of another node to connect the two nodes. The connection represents the dataflow. The output of a node (right handed triangle) is the input (left handed triangle) of the next node.
- Right click on a node to open a popup menu. The main action of this menu is to configure the node. Other common actions are to execute the node, edit the tooltip message, or to get a preview of the data processing by the node.
- Lay the mouse over an in or out triangle of a node to get a snippet of the state of the data at this location of the workflow.
- Right click to an edge connecting two nodes to edit or delete it.
- It is recommended to find a particular node using the search tool of the *Node Repository.*

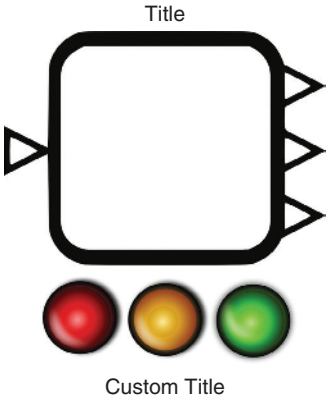


Figure 1.2 Schematic view of a KNIME node. The main title describes the data processing. To the left and right, the handles represent the input and output respectively. The traffic light indicates the node status and below is a custom title for the node.

Step-by-Step Instructions

The tutorial requires the installation of the ChemAxon nodes for KNIME (version 2.12 or later) called JChemExtension (version 2.8 or later). To use them, a ChemAxon[1] and a JChemExtension[2] license are needed. To install new nodes, proceed as follows:

1) In the **help** menu of KNIME chose **Install new software…**
2) In the pop-up window click the button **Add…**
3) Give a meaningful name (for instance JChemExtension) and the relevant URL (currently https://www.infocom.co.jp/bio/knime/update/3.1).
4) Click the **Next** button and continue through the installation wizard packages for verification and license agreement.

The JChemExtension license file is loaded through the **Preferences** menu of KNIME, then into the topic **KNIME, JChem**. ChemAxon license files are installed using the dedicated ChemAxon **License Manager** software.

At any time in the tutorial, it is recommended to use the **New Workflow Annotation** in a right-click popup menu in the workbench to add comments and colors to the workbench. This helps to order and clarify the workflow. It is also recommended to customize the names of the nodes so that the workflow becomes more self-explaining. A main objective of a workflow-programing environment such as KNIME is to provide a clear picture of the process.

1 https://www.chemaxon.com/products/
2 http://infocom-science.jp/product/detail/jchemextension_update_en.html

Instructions	Comments
• Launch KNIME and open a new workbench.	This starts a new project. The user is invited to give a meaningful name for their project.
• Use the node **CSV Reader** and configure it so that the *Column Delimiter* is a semicolon (;) and the *Quote Char* is a double quote (“). Uncheck the option *Has Column Header* and the option *Has Row Header*. Set the file source as `thegoodscent_dup.csv`	The input file was roughly formatted so that the column delimiter is a semicolon and all data element is delimited by double-quotes. This is usually a good choice, since these characters are usually not present in linear chemical notations like SMILES of InChI. There is no column title line in this file.
• Add the **Column Rename** node and connect its input to the output of the **CSV Reader**.	The columns of the raw CSV file need to be identified.
• Configure the **Column Rename** node to rename each column as follows ○ Col0: *Name* ○ Col1: *CAS* ○ Col2: *Odor Type* ○ Col3: *Odor Description* ○ Col4: *Odor Ref* ○ Col5: *Taste Description* ○ Col6: *Taste Ref* ○ Col7: *SMILES*	The configuration interface shall look like the Figure 1.3.
• Execute the workflow by clicking on the green lecture buttons of the tool bar. Check the *Console* window for errors and warnings.	Upon execution of the workflow, at least one error will occur. On line 11860 of the CSV file, the compound is named `6"-O-malonyl genistin`. The character `"`is taken for a word delimiter. The remedy is to use a text editor to replace inside the line 14003, the `"` by a `'`. In fact, many inconsistencies were present in the raw extract of the website: misinterpreted characters, fused words, confusion between l, I and 1, and O and 0.

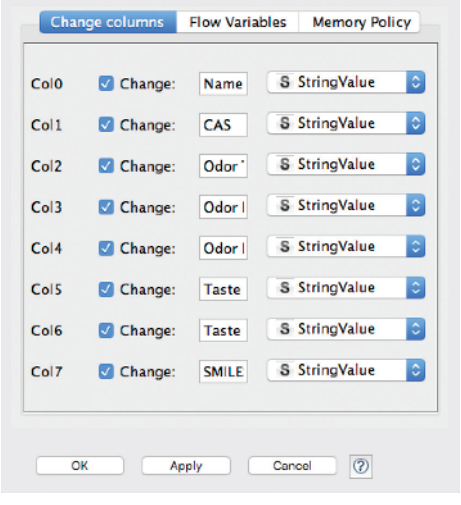


Figure 1.3 Configuration interface of the Column Rename node.

Instructions	Comments
• Right click on the node **Column Rename** and select the item *Renamed/Retyped table*. • Look at lines 7688-7798 to identify, by their name, some undesired entries. • Add a **Row Filter** node and configure it. • In the configuration interface, select *exclude rows by attribute value*, chose the *select the column to test* as *Name* and as *matching criteria* select the *use pattern matching* option. Then use as *pattern* the string: `peg-.*|ppg-.*` • Tick the box *regular expression*.	Browsing through the entries, for instance at lines 7688-7798, polyethylene glycol (PEG) and polypropylene glycol (PPG) can be found. Since the objective of the database is to collect organoleptic properties of small organic compounds, such substances should be excluded. Those compounds are found by their names, containing the substring "peg-" or "ppg-". In regular expression semantics supported by KNIME, a simple query "`peg-.*`" matches any line containing the substring "`peg-`" followed by any number of characters "`.*`" The "`|`" character is a logical OR. Thus, the query "`peg-.*|ppg-.*`" searches simultaneously any substring containing either "`peg-`" or "`ppg-`" (Figure 1.4). Many other PEG/PPG containing substances were present in the database (steareths, octoxynols, ceteths, oleths, isoceteths, poloxamines, trideceths, pareths, ceteareths) as well as lauryl ether sulfates. They are widely used as surfactant, emulsifying agents, cleansing agents, and solubilizing agents for cosmetics. They were removed from the dataset in this tutorial, but they illustrate the particularities of chemistry that must be taken care of when processing data.

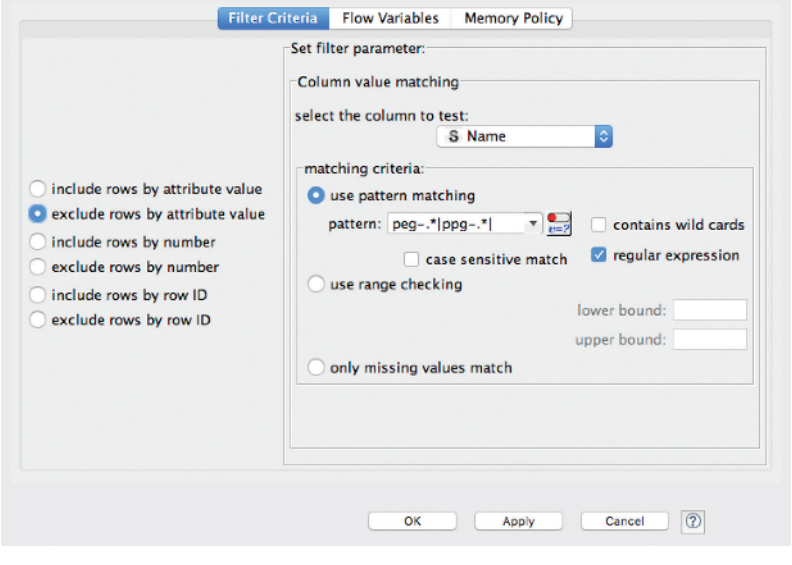


Figure 1.4 Configuration of the Row Filter node used to remove polyethylene glycol and polypropylene glycol.

(*Continued*)

Instructions	Comments
• Right click on the last node **Row Filter** and select the item *Filtered.* • Look at lines 2666-2668 to identify by their name, some undesired entries. • Add a **Row Filter** node and configure it. • In the configuration interface, select *exclude rows by attribute value*, chose the *select the column to test* as *Name* and as *matching criteria* select the *use pattern matching* option. Then use as *pattern* the string: `.*\bextract\b.*|.*\boil\b.*` • Tick the box *regular expression*. • Note that the node can be configured temporarily to *include rows by attribute value*. After execution, it allows to check those lines that are matched by the node.	Another specificity is the use of chemically ill-defined substances. For instance, looking at lines 2666-2668 some extracts are present. This node is dedicated to search and remove lines mentioning the words oil or extract. The regular expression is: `.*\bextract\b.*|.*\boil\b.*` The `\b` are word delimiters, in order to avoid matching accidentally the substring "oil" with words such as "foil," "boil," or "soiled" for instance. Perfumery and cosmetics uses natural extract, therefore, the raw data must be investigated for such keywords as "absolute," "resin," "concrete," "root," "wood," "leaf", "flower," "fruit," "juice," "seed," "bean," "tincture," "straw," or "water" for instance. Keyword filtering is often insufficient. It is also necessary to check what are the data matched by the keyword and expert decision and perhaps to include or exclude a data point.
• Add a **Rule-based Row Filter** node and connect its input to the output of the last **Row Filter** node of the workflow. • Configure the **Rule-base Row Filter** node. • In the Expression field use the following rule: `($Odor Type$ = "") AND ($Odor Description$ = "") AND ($Taste Description$ = "") => TRUE` • Select the *Exclude TRUE matches* option. • Execute the workflow	Entries with no recorded odor or taste information are not needed. This requires a test on several columns. Therefore, a more complex request must be build, motivating the use of the **Rule-based Row Filter** node. This node manages complicated logical rules (involving any combination of logical operators `NOT, AND, OR, XOR`) based on any columns. Columns are referred to by their name surrounded by `$`. The configuration of the node should look like Figure 1.5. So the rule matches lines with empty "Odor Type," "Odor Description," and "Taste Description" fields.

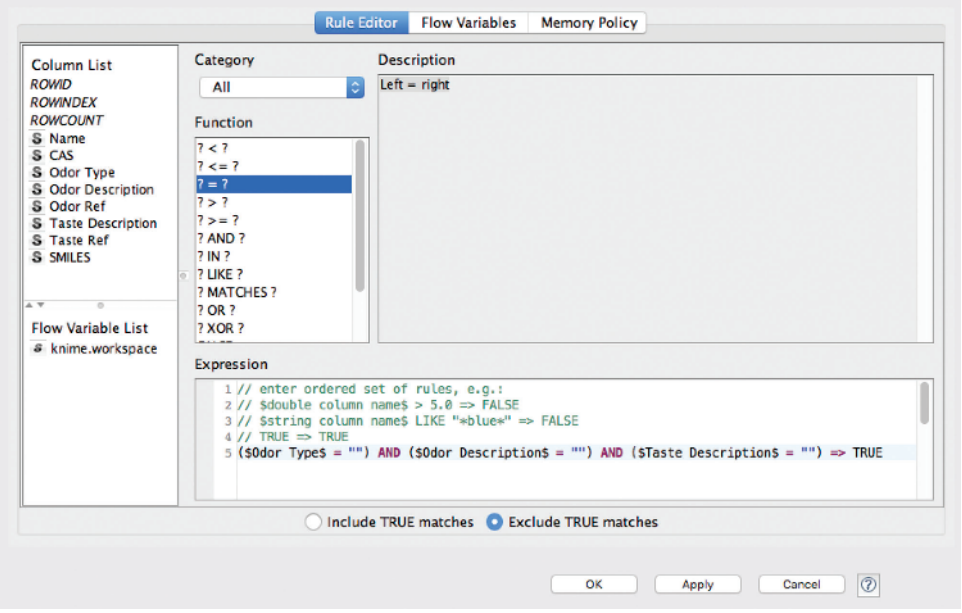


Figure 1.5 Configuration of the Rule-based Row Filter node to remove instances providing no information on the organoleptic properties.

Instructions	Comments

• Connect the input of a **Rule Engine** node to the output of the last **Rule-based Row Filter** node.
• Configure the **Rule Engine** as shown in Figure 1.6.
  ○ Add the following lines into the *Expression* text-edit region:
 	
`($Odor Type$ = "") AND ($Odor Description$ MATCHES "\b\w*\b") => $Odor Description$`
`($Odor Type$ = "") AND ($Taste Description$ MATCHES "\b\w*\b") => $Taste Description$`
`(NOT $Odor Type$ = "") => $Odor Type$`
 	
  ○ You can use the *Column List*, *Category* and *Description* elements of the configuration interface to help you write the above three rules.
  ○ Invalid syntaxes are underlined in red and the node cannot be executed.
  ○ Select the *Replace Column* option and choose the column *Odor Type*.
Validate and execute the node.
	The *Odor Type* field defines the organoleptic category to which the substance belongs. However, for a few hundreds of compounds, this information is missing.
	The first attempt to recover this information is to use the *Odor Description* or the *Taste Description* fields if they contain only one word.
	This manipulation requires the use of the **Rule Engine**. The node is programmed using three rules to compute a value, in decreasing order of priority:
	1) If the *Odor Type* is empty and the *Odor Description* is composed of one word, use the *Odor Description* as the *Odor Type*.
	2) If the *Odor Type* is empty and the *Taste Description* is composed of one word, use the *Taste Description* as the *Odor Type*.
	3) If the *Odor Type* is not empty, use the value of *Odor Type* as the *Odor Type*.
	The *Odor Description* and *Taste Description* fields are compared to the following regular expression:
	
	`\b\w*\b`
	
	It matches those fields composed of word characters (`\w`) in any number (`*`), as long as they are in one same unique word (enclosed by `\b`).
	Once a rule is applied the others are ignored.
	The third rule is important because the result overwrites the *Odor Type* column. If no rule applies to an instance the result of the operation is *undefined* (annotated as "?"), and the *undefined* value will overwrite the initial content of the *Odor Type* field. Using the third rule, the *Odor Type* value is kept unchanged by default.

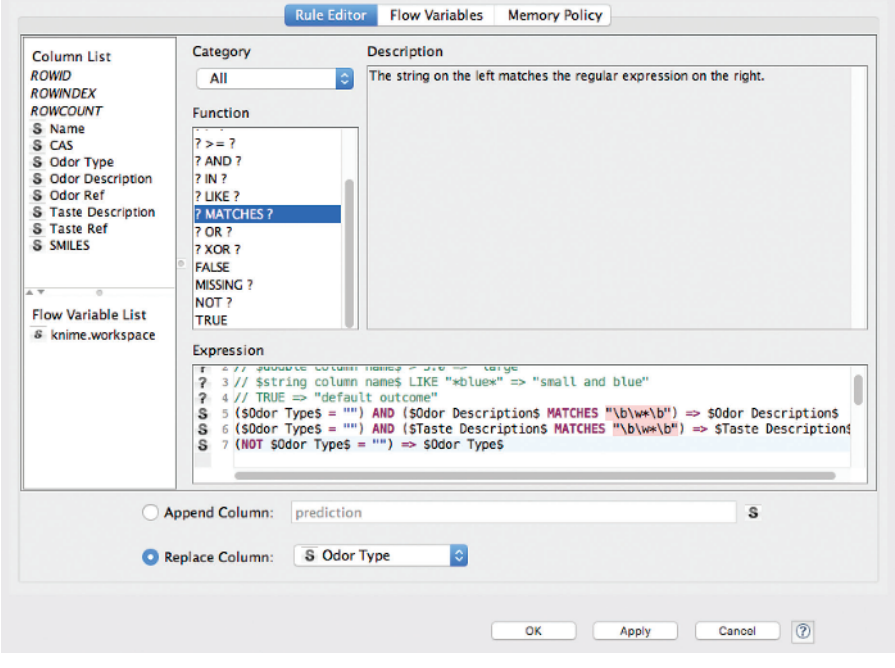


Figure 1.6 Configuration of Rule Engine node using the Odor Description and Taste Description fields to fill in the missing values of the Odor Type field.

(*Continued*)

Instructions	Comments
• Include and configure a **CSV Reader** node (Figure 1.7). It will read the file `MissingOdorTypes.csv`. The *Column Delimiter* is a semicolon (;) and the *Quote Char* is a double quote (“). Check the option *Has Column Header* and uncheck the option *Has Row Header*.	Yet, for about 300 substances, the *Odor Description* and *Taste Description* are too complicated to be used as an *Odor Type*. An expert had to review these items and manually corrected them. The corrections are collected into the file *MissingOdorTypes.csv*. This file is loaded with a **CSV Reader** node and generates a second data flow.

• Include a **Joiner** node. Connect the **CSV Reader** node output to the *Right table* input of the **Joiner** (the bottom left handle) and the **Rule Engine** output to the *Left table* input of the **Joiner** (the top left handle).
• Configure the **Joiner** node.
  ○ In the *Joiner Settings* tab (Figure 1.8) select the *Left Outer Join* option as the *Join mode*. Then in the *Joining Columns* menu, select the match all of the following option and select the column *SMILES* for both left and right table.
  ○ In the *Column Selection* tab (Figure 1.9), uncheck the *Always include all columns* option of the *Right table*. In the column selection menu interface of the *Right table* exclude all columns except the *Odor Type*.
	The field *SMILES*, which is common to the main dataflow (or left table) and the new dataflow (right table), is used to merge them via a **Joiner node**. The columns that come from the right table are filled up with missing values if no matching row exists in the right table. All columns from the left table are included while only the *Odor Type* field of the right table is added to the output.

• Use and configure a **Column Merger** node (Figure 1.10).
  ○ Select as *Primary Column* the column *Odor Type*.
  ○ Select as *Secondary Column* the column *Odor Type (#1)*.
  ○ Select the option *Replace primary column*.
	Then, a new **Column Merger** node is configured in order to merge the values of the *Odor Type* field from the left and from the right table. If the value is missing inside the *Primary Column*, the value is taken from the *Secondary Column*.

• Add and configure a **Column Filter** node (Figure 1.11). The column *Odor Type (#1)* should be inside the *Exclude* area while all other columns should be inside the *Include* area.	Finally, the **Column Filter** node discards the *Odor Type* from the left table, as it is no longer needed.
• Add to the workflow a **Domain Calculator** field (Figure 1.12). • Add the *Include* area, the columns *Odor Type*, *Odor Ref* and *Taste Ref*. • Set the *Restrict number of possible values* to 200.	Although the KNIME system does not have a Nominal type to describe the data (they are manipulated as normal strings), some data integrity checking can be enforced. The **Domain Calculator** node is used to list all words used in the *Odor Type*, *Odor Ref*, and *Taste Ref* fields. After execution, it can be investigated into the *Spec – Columns* tab of the output *Data table* view. This is the place to check for errors and misspelling (“floal” instead of “floral,” “cirus” instead of “citrus” for instance) that can occur during manual data entry.

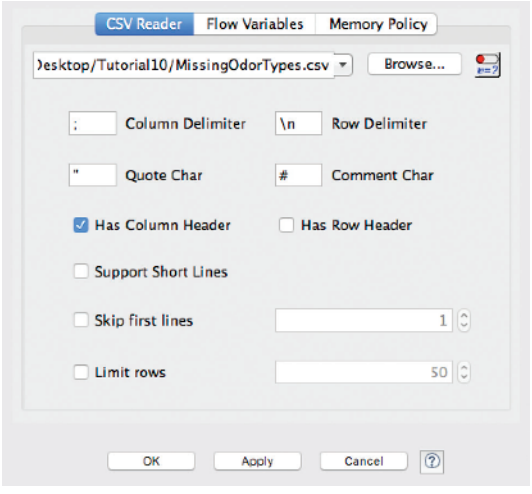


Figure 1.7 Configuration of the CSV Reader node, loading corrected Odor Type fields from the file MissingOdorTypes.csv.

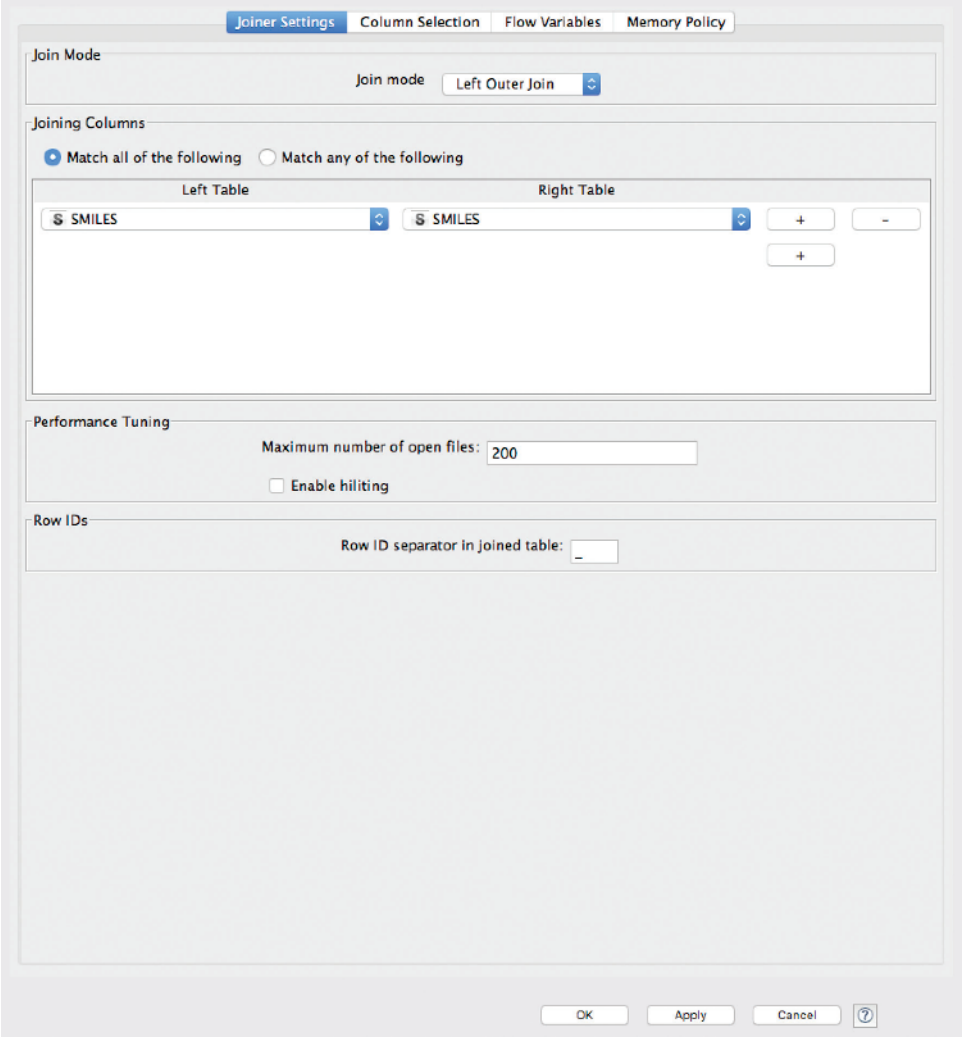


Figure 1.8 Joiner Settings of the Joiner node interface.

(*Continued*)

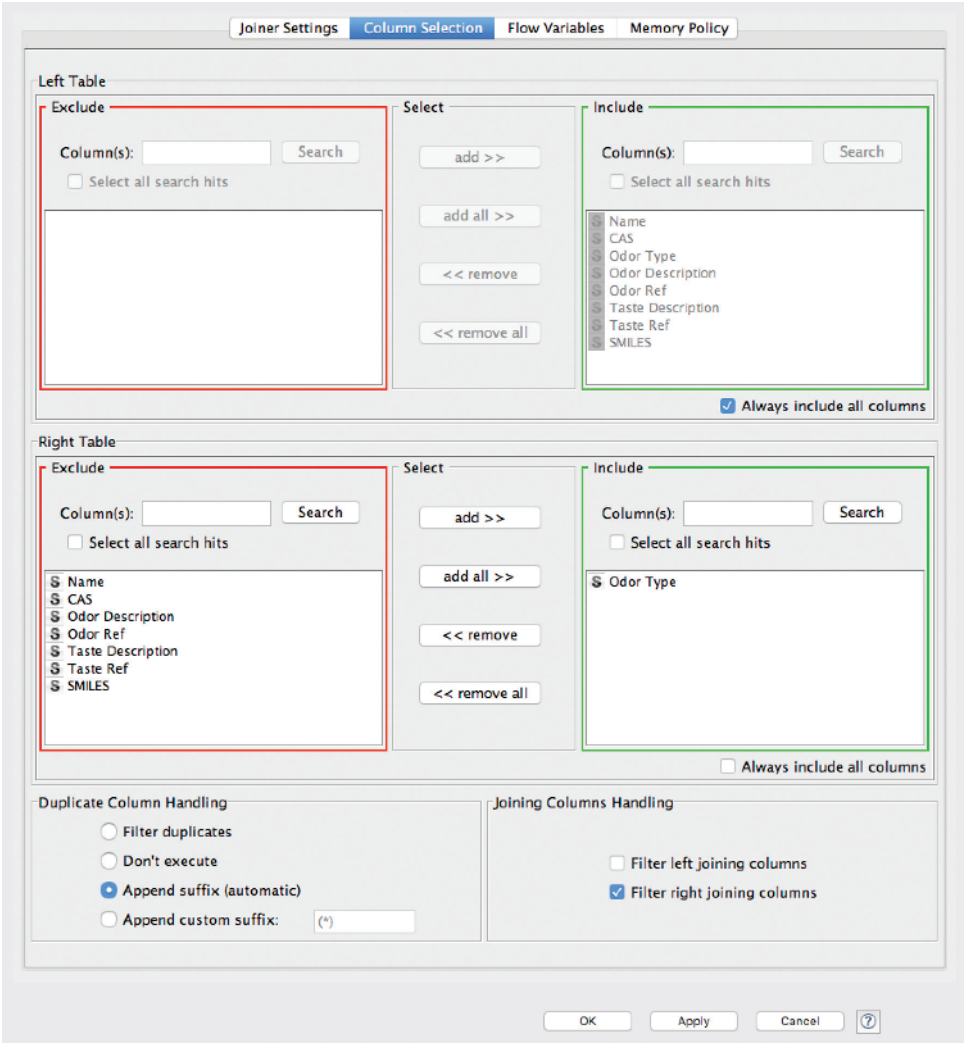
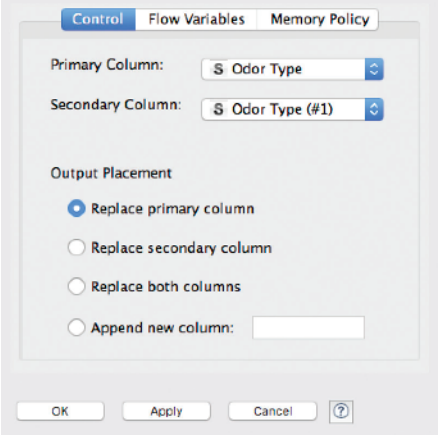


Figure 1.9 Column Selection interface of the Joiner node.

Figure 1.10 Configuration of the Column Merger node merging the manually curated Odor Type fields to fill in the missing values of the Odor Type field of the main dataflow.



Instructions	Comments
• Connect a **Column Combiner** node to the last node of the workflow. • Configure the node so that the *Include* area contains only the *Odor Type*, *Odor Description*, and *Taste Description*. • Set the *Delimiter* to "," and choose the *Replace Delimiter by* option. Use the space character inside the text box of this option. Set the *Name of appended column* to *Raw Organoleptic*.	Another part of the workflow shall be dedicated to the aggregation of *Odor Type*, *Odor Description*, and *Taste Description* in a general description of the organoleptic description of the substance. The first step is to merge the columns *Odor Type*, *Odor Description* and *Taste Description* (Figure 1.13).

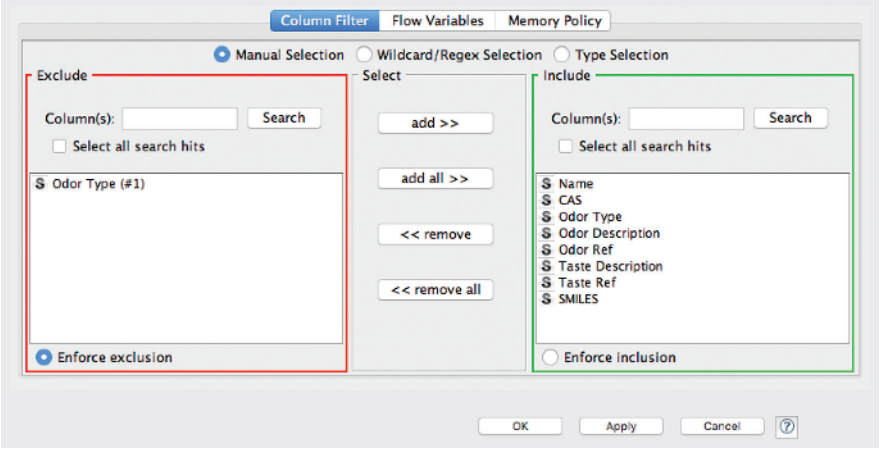


Figure 1.11 The Column Filter interface, used to discard the manually curated Odor Type field. It is now useless since the information is already merged into the main dataflow.

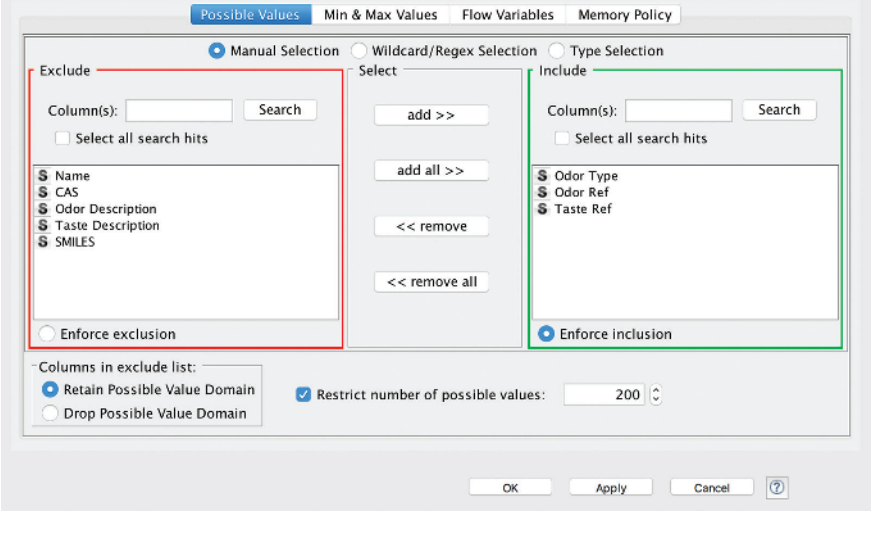


Figure 1.12 Domain Calculator node interface, used to enumerate the possible values of the Odor Type field, and the odor and taste descriptions references.

(*Continued*)

Instructions	Comments
• Connect a **String Replacer** node to the output of the **Column Combiner**. • Configure the **String Replacer** node (Figure 1.14). • Set the *Target Column* to *Raw Organoleptic*. • Choose the *Regular Expression* option. • Type in the following string as the *Pattern*: `,|\.|-|/|&|\d+` • Enter a white space as the *Replacement text*. • Choose the *all occurrences* option.	The new field, *Raw Organoleptic*, added to the workflow contains points, comma, ampersand, hyphens, and digits. These characters do not participate directly to the semantic of organoleptic properties. So, they are removed. In the regular expression, the dot character is represented by a "\." because otherwise the dot is a wild card for *any* character. The notation "`\d+`" matches a digit (`\d`) appearing one or more times (+).

• Add a **String Manipulation** node to the workflow.
• Configure the **String Manipulation** node (Figure 1.15).
  ○ Select the option *Replace Column* and choose *Raw Organoleptic*.
  ○ Inside the *Expression* text edit region, type the command:
`lowerCase($Raw Organoleptic$)`
	Additionally, the *Raw Organoleptic* field aggregates sentences describing an odor or a taste. So the field uses a mixture of lowercase and uppercase.

• Connect the output of the **String Manipulation** node to the input of a **String Replacer** node.	It also uses words that are not related to an odor description. These words are: "with," "a," "an," "the," "and," "of," "for," "in," "low," "high," "it," "like," "likes," "likely," "slight," "slightly," "nuance," "nuances," "enhanced," "reminiscent," "penetrating," "intense," "has," "all," "minutes," "breath," "mouth," "feel," and "feels." Therefore, one node is used to lowercase all characters, and a second node is used to remove the undesired words from the organoleptic description.

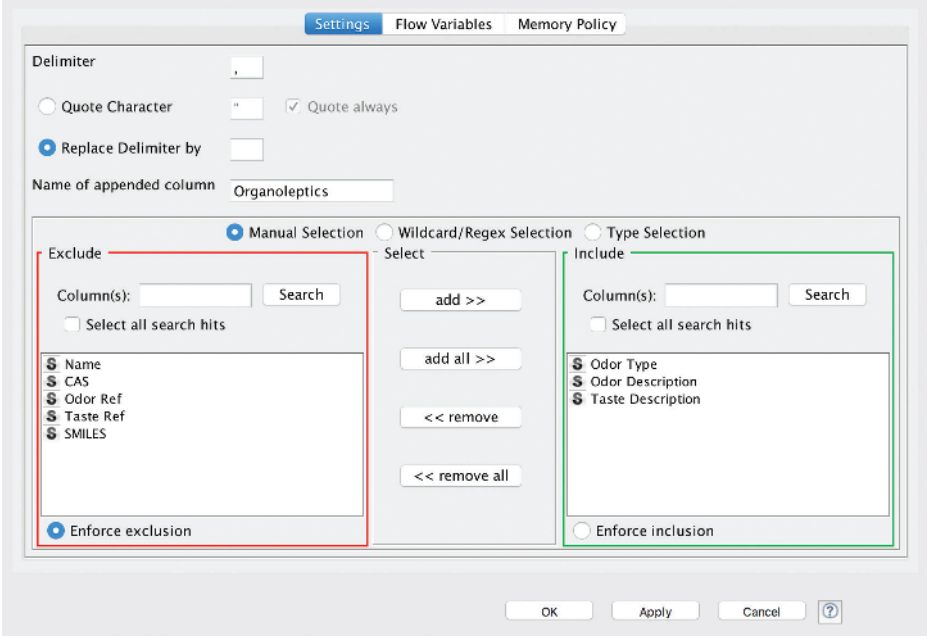


Figure 1.13 Configuration of the Column Combiner Node, used to agregate odor and taste description into a global organoleptic description.

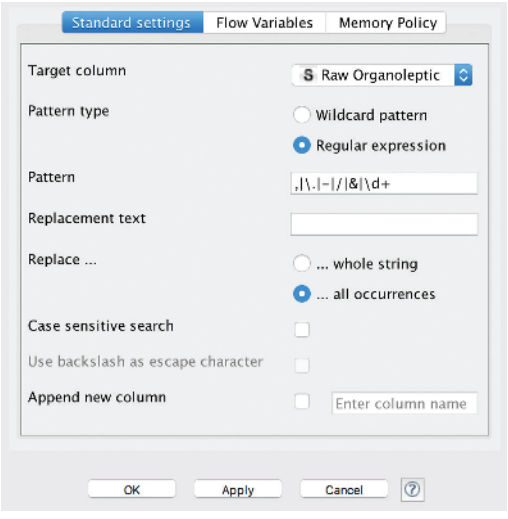


Figure 1.14 Configuration of the String Replacer node removing punctuation, special characters, and numbers from the Raw Organoleptic field.

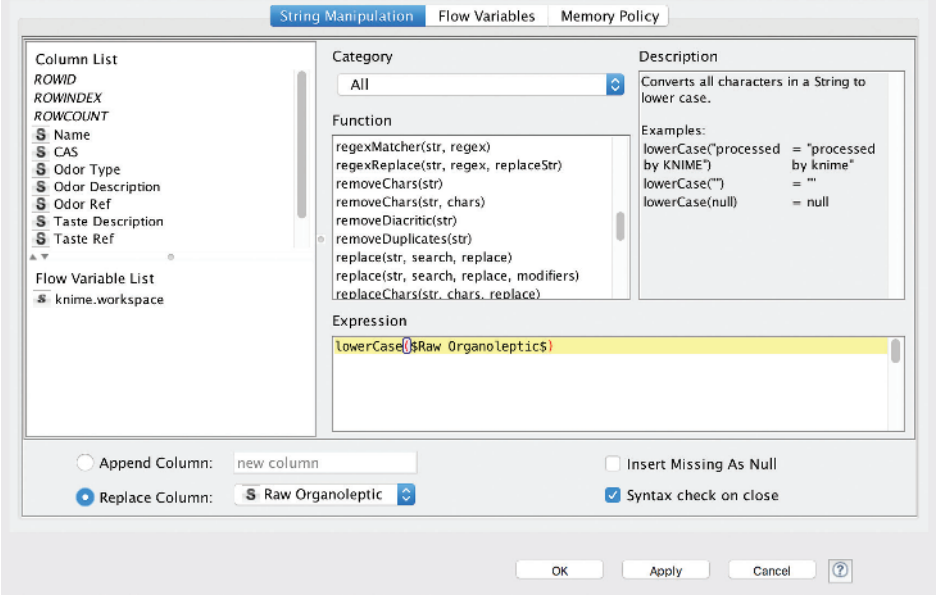


Figure 1.15 String Manipulation node, used to lowercase all characters of the Raw Organoleptic field.

(*Continued*)

Instructions	Comments

• Configure the **String Replacer** node (Figure 1.16).
  ○ Set the *Target column* to *Raw Organoleptic*. As *Pattern Type* select *Regular expression*. As *Replace* option select *all occurrences*.
  ○ Inside the *Pattern* text edit region, type in the following command:
`\bwith\b|\bthe\b|\band\b|\ball\b|\ban\b|\ba\b|\bof\b|\blike\b|\blikes\b|\bslight\b|\bslight.+\b|\breminiscent\b|\benhanced\b|\bnote\b|\bnote.+\b|\bin\b|\bbackground\b|\blasts\b|\bbreath\b|\bfor\b|\bminutes\b|\bmouth\b|\bfeel\b|\bfeels\b|\bpenetrating\b|\bnuance\b|\bnuance.+\b|\bit\b|\bhas\b|\bintense\b|\blow\b|\bhigh\b`	

• Add a **String Manipulation** node.
• Configure the **String Manipulation** node (Figure 1.17).
  ○ Select the option *Replace Column* and choose *Raw Organoleptic*.
  ○ Inside the *Expression* text edit region, type the command:
`removeDuplicates($Raw Organoleptic$)`
	These operations generate successions of white spaces. The last **String Manipulation** node removes these duplicate white spaces.

• Add a **Cell Splitter** node and configure it (Figure 1.18).
  ○ Select the column Raw Organoleptic.
  ○ Use a white space inside the field *Enter a delimiter*.
  ○ Tick the box *remove leading and trailing white space chars (trim)*. Chose the option *as set (remove duplicates)*.
	The *Raw Organoleptic* field needs to be converted to a set. Each word describing the organoleptic of a substance appears exactly once within a set. The organoleptic description of the substance reduces to the organoleptic semantic.
	The set of all these words constitute a compendium of the description of chemical substances. A chemical is associated to only some of these words.

• Connect a **Molecule Type Cast** to the workflow. • Configure the node so the *Structure Column* is *SMILES* and the *Structure Type* is *smiles*.	Chemical structures are stored as string and as such chemically specialized nodes cannot process them. The role of the **Molecule Type Cast** node is to annotate and interpret the part of the dataflow concerning chemical structure. This node is very permissive. No quality check of the chemical structure is done.
• Connect the output of the **Molecule Type Cast** node to the input of the **Structure Checker** node.	This is the role of the **Structure Checker** node. The actions of this node are performed in the same order as they are listed in the configuration interface.

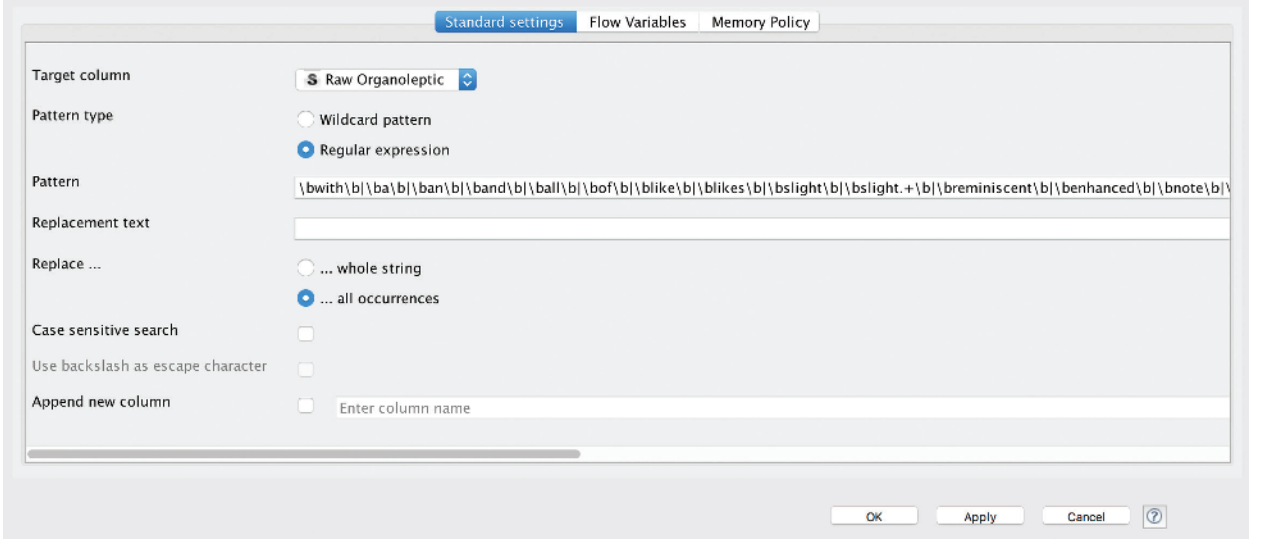


Figure 1.16 The String Replacer node, used to remove the undesired words from the Raw Organoleptic field using regular expressions.

(*Continued*)

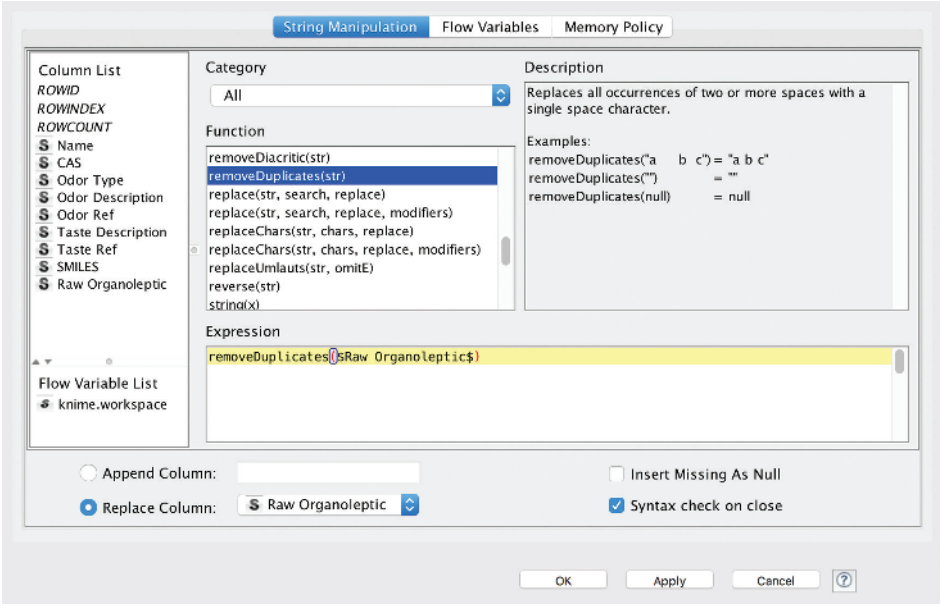


Figure 1.17 String Manipulation node, used to remove duplicate white spaces in the Raw Organoleptic field.

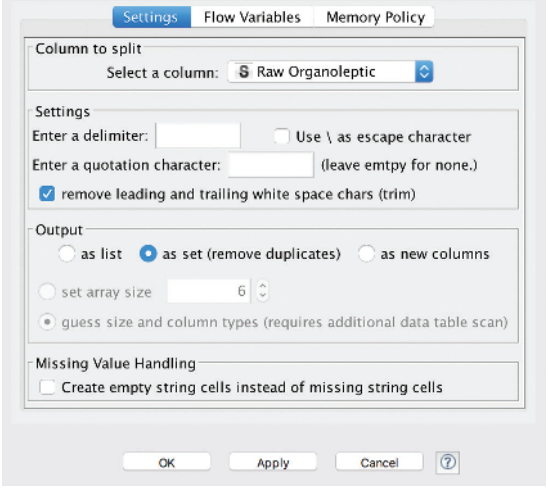


Figure 1.18 Configuration of the Cell Splitter node converting the organoleptic description to a set of descriptive words.

Instructions	Comments

• Configure the **Structure Checker** (Figure 1.19). Add the following actions to the right-hand window:
  ○ *Empty Structure Checker*
  ○ *Rare Element Checker*
  ○ *Covalent Counterion Checker*
  ○ *Valence Error Checker*
  ○ *Ring Strain Error Checker*
  ○ *Aromaticity Error Checker*
• It is usually possible to customize an action by clicking on it into the right-hand window. Customize the *Aromaticity Checker* so that it uses a *Basic* aromatization scheme and set the *Fix* option to *Dearomatize*.
• Connect the discarded structure output (the bottom right handle of the node) a **Table Writer** node. Configure the node so the output table file is called *Discarded.table*. Check to *Overwrite OK* box.
	It starts by checking that the chemical structure field contains a chemical structure.
	Then, the atoms of the structure are checked. Rare atoms are those not in the following list: H, Li, Na, K, Mg, Ca, B, C, N, O, F, Cl, Br, I, Al, P, S, Cr, Mn, Fe, Co, Ni, Cu, Zn.
	The third step is to check if an alkali metal or alkaline earth metal is covalently connected to an N, O, or S. If this is the case, the covalent bond is deleted and the atoms are charged.
	The fourth step searches for valence overflows (forbidden by the octet rule). Unless the problem is an additional H (which is then removed), the chemical structure cannot be automatically repaired.
	In the fifth step, impossible intracyclic bonds are searched for. These bonds are trans double bonds, successive double bonds and triple bonds. These bonds cannot exist within small organic molecule rings.
	Finally, the sixth step considers the aromatic representation of rings. The *Basic* style uses Hückel's rule.
	The node has two output handles. The bottom one is the dataflow containing chemical structures that are discarded by the node, while the top one can be safely processed.
	The discarded structures cannot be further processed. They should be fixed separately and this is why the **Table Write** node is connected immediately to the discarded structure handle.

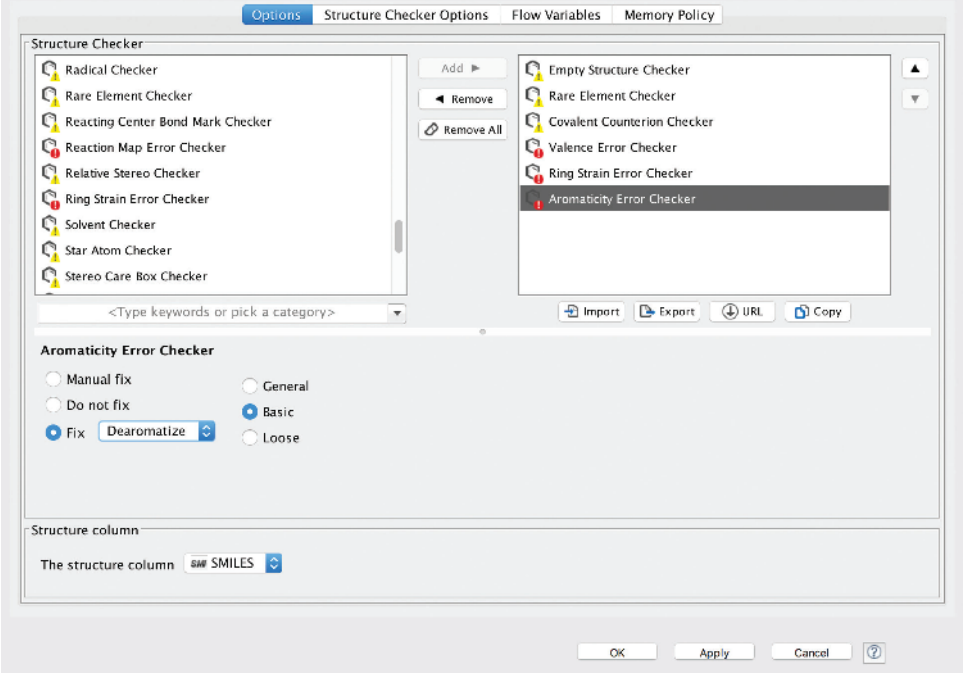


Figure 1.19 Configuration of the Structure Checker focused on atoms and valences of the chemical structures.

(*Continued*)

Instructions	Comments

• Edit the file *thegoodscent_dup.csv* to correct some errors.
  ○ Line 7607, replace "H2S" by "S"
  ○ Line 1673, replace "InChI = 1/C7H8S/c1-6-4-2-3-5-7(6)8/h2-5,8H,1H3" by "Cc1ccccc1S"
  ○ Line 6711, replace "SJWFXCIHNDVPSH-MRVPVSSYSA-N" by "CCCCCC[C@@H](C)O"
  ○ Line 6045, replace "CC[S-]C(=O)CNC(=O)C" by "CCSC(=O)CNC(=O)C"
  ○ Line 7645, replace "O = [Bi].Cl" by "O = [Bi]Cl"
  ○ Line 7164, replace "CC(=O)c1ccc(C)n1" by "CC(=O)c1ccc(C)[nH]1"
  ○ Line 15308 and 15896, replace "Cc1c(sc2c1c(nc(=O)[nH]2)N)C" by "Cc1sc2NC(=O)N = C(N)c2c1C".
• If you edited the CSV file directly, reset the first **CSV Reader** node of the workflow and re-execute the whole workflow.
• Alternatively, you can add a new **CSV Reader** configured to read the file *StructureCuration.csv*, using a semicolon (";") as *Column Delimiter* and with *Has Column Header* ticked and *Has Row Header* not ticked.
• Connect the output of this new **CSV Reader** to the dictionary table input (bottom left handle) of a new **Cell Replacer** node. Connect Table containing column (top left handle) to the output of the **Column Rename** node that is the second node of the workflow.
• Configure the **Cell Replacer** node (Figure 1.20) so the *Target Column* is set to *SMILES*, the *Input (Lookup)* is set to *LookUp* and *Output (Replacement)* is set to *Replacement*. Choose the option *Input* as *If no element matches use*.
• Connect the output (right handle of the node) to the input of the **Row Filter** that is the third node of the workflow.
• Re-execute the workflow.
	The procedure above should identify the following problems.
	• Some SMILES structures are wrong.
	  ○ Line 7607, hydrogen sulfide is coded "H2S" instead of "S."
	  ○ Line 1673, ortho-thiocresol is encoded with the InChI code.
	  ○ Line 6711, the (R)-(-)-2-octanol is encoded by its InChIKey.
	• The name and structure are misspelled.
	  ○ Line 6045, "S-ethyl 2-acetyl aminoethane thioate" should be understood as "S-ethyl 2-acetamidoethanethioate."
	  ○ Line 7645, "bismuth(III) oxychloride" is simply "bismuth oxychloride."
	• Warnings about loose definition of aromatic rings
	  ○ Line 7164, the nitrogen of 2-acetyl-5-methyl pyrrole has a proton that should be included into the SMILES code.
	  ○ Line 15308 and 15896, pyrimidin-2-one in 4-Amino-5,6-dimethylthieno(2,3-d) pyrimidin-2(1H)-one should not be aromatic.
	• Structures that attract attention.
	  ○ Line 2651, cyclohexasiloxane.
	  ○ Line 43778, vanadium pentoxide.
	  ○ silica and silicates at lines 7429, 15231, 4414, 4469, 4471, 4476.
	  ○ Pure elements, such as helium (line 7335).
	  ○ Metal oxides such as titanium oxide (line 8348), calcium oxide (line 4353), and magnesium oxide (line 4360).
	• The input file can be edited manually.
	• An alternative is to use the prepared file *StructureCuration.csv* containing the rules to curate these deficient SMILES codes. It requires to include in the workflow some dedicated nodes. This is the preferred solution if it is expected that the source be updated with the very same errors in the future.

Instructions	Comments

• Add a **Standardizer** node to the workflow.
• Configure the **Standardizer** node (Figure 1.21):
  ○ Add a *Dearomatize* action
  ○ Add a *Mesomerize* action
  ○ Add a *Tautomerize* action
  ○ Add an *Aromatize* action
  ○ Add a *Transform Nitro* action
  ○ Add a *Clean 2D* action
• As before, an action can be customized by clicking on the corresponding line in the right-hand space of the interface.
• Configure the *Aromatize* action in order to use a *Basic style aromatization*.
• Tick the box *Append column*.
	The chemical structures in the dataset are still heterogeneous. For instance, some aromatic rings are in a Kekulé form while others are in aromatic form, which would bias substructure search, similarity searching or any attempt to model the dataset based on the chemical structure.
	The goal of the **Standardizer** node is to define a set of rules according to which a chemical structure shall be drawn. The rules are applied to each chemical structure of the workflow following the order of the list defined inside the right-hand area of the configuration interface of the node.
	In the present situation, the first step is to dearomatize a structure. The dearomatized structure is used as a reference state to compute a standard mesomer and tautomer state of the compound. Then the structure is aromatized, using the *Basic* style. This implements an homogeneous representation of the chemical structure.
	Yet, some chemical functions can pose a challenge to any automatic procedure to cure the chemical structures. For instance, the PubChem database has enumerated about 60 different representation of a nitro group [9]. The rule integrated into the **Standardizer** node concerns only one of them: if the pentavalent nitrogen is found in a nitro group pattern, it is replaced by positively charged tetravalent nitrogen and one oxygen becomes negatively charged.
	The pattern matching and replacement strategy is becoming obsolete. Already in 2005, an early attempt to create a chemical ontology was referencing 231 chemical functions [10]. More recently [9], PubChem was analyzed in terms of atom environments (atoms plus the neighbor atoms with chemical bonds). Considering only an atom and the chemical bonds around, about 1600 unique atom environments were enumerated. With up to one neighbor atom, about 110 000 unique atom environments were found. Based on these numbers, the number of rules to exhaustively standardize chemical functions would be between 10 000 to millions.

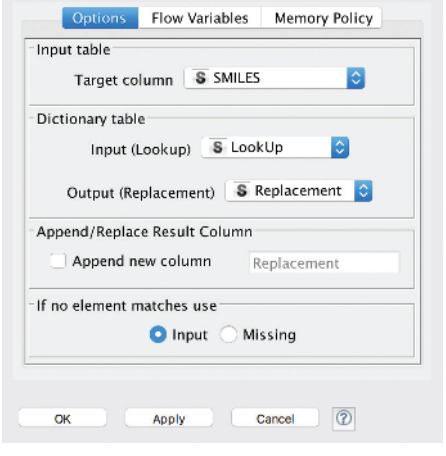


Figure 1.20 Configuration of the Cell Replacer node used to curate some SMILES of chemical structures.

(*Continued*)

Instructions	Comments
• Add a **MACCS Fingerprint** node. In the configuration interface (Figure 1.22), select *Binary* as *Output type* and *SMILES (Standardized)* as *The structure column.* • Add a **Naming** node (Figure 1.23). Choose the option *Traditional Name* and select *SMILES (Standardized)* as *The structure column*.	Once the chemical structures are better validated, some calculations can be done a priori to help the management of data. The first one is to compute molecular fingerprints. Yet, the chemical data to process refer to substances, not molecules. A possible relevant choice for such a situation are MACCS fingerprints [11]. These fingerprints are very qualitative, encoding for elemental analysis, presence of particular substructure and annotation about the nature of the chemical (molecule, substance, mixture, monomer…). Although in the present implementation, only a subset of 166 bits is used, they are robust enough to be relevant for organoleptic substances. Another useful computed property of the chemical structures is the name of the compound. As the compounds are fairly common, the common name is preferred.
• Add a **Category to Number** node and connect the *Data* handle to the output of the previous node of the workflow. • Configure the node (Figure 1.24). • Add to the *Include* section, the columns *Odor Type, Odor Ref*, and *Taste Ref.*	It is time to export the dataflow, using a format suitable for management into a relational database system. The export shall include at least four tables: • The odors references • The tastes references • The odor types • The substances.

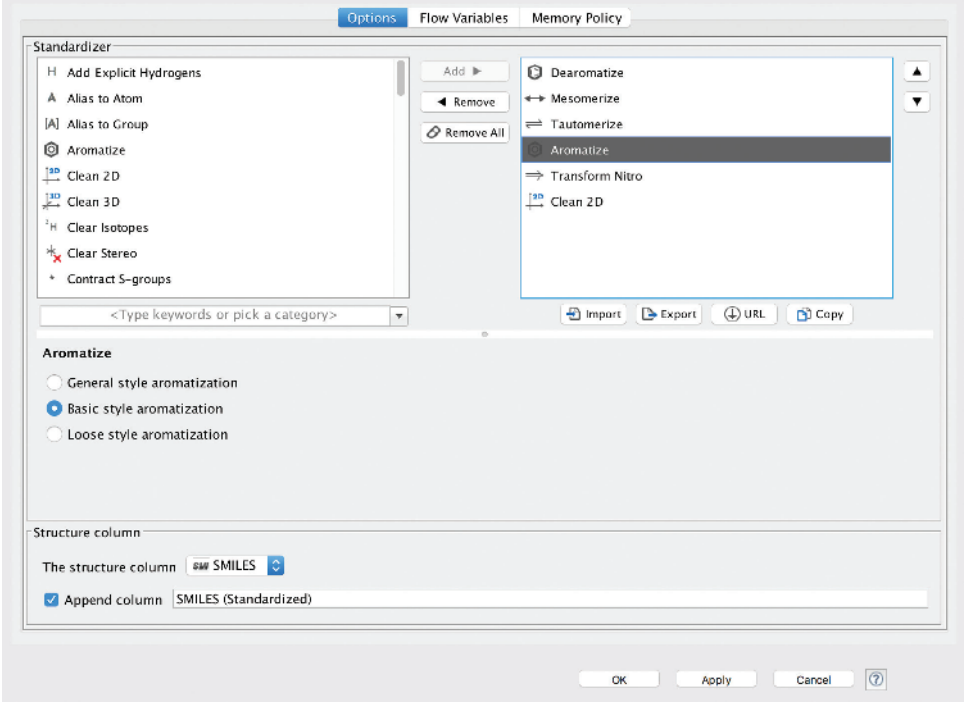


Figure 1.21 Standardizer node configuration interface setting rules for aromatic ring representation, taking into account mesomers and tautomers.

Figure 1.22 Configuration interface of the MACCS Fingerprint node.

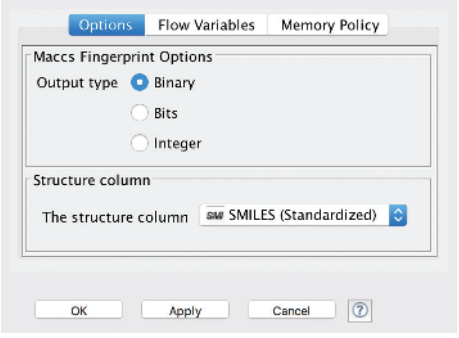
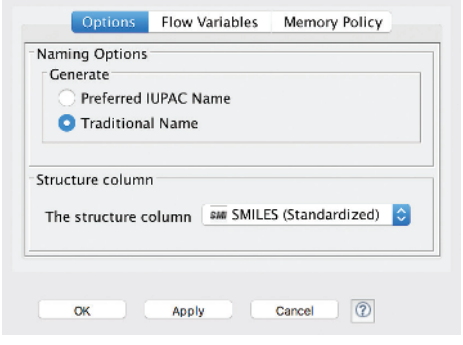


Figure 1.23 The configuration interface of the Naming node.



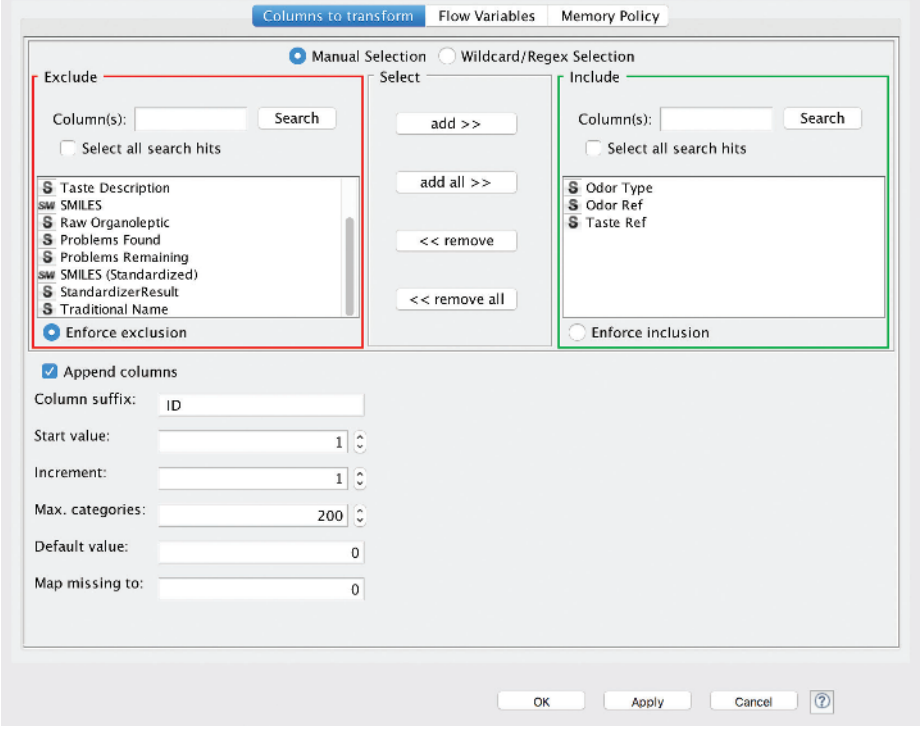


Figure 1.24 Configuration interface of the Category To Number node used to generate foreign keys.

(*Continued*)

Instructions	Comments
• Set the *Start value* to 1. Set the *Default value* and the *Map missing to* values to 0.	The table shall be connected through foreign keys. The goal of the **Category To Number** node is to map each category of nominal data to an integer. Here, each reference for the description of an odor (respectively as taste) is one category. The generated integer becomes the foreign key. The *Default value* and the *Map missing to* values are set so that in any case, the absence of information will also generate a valid foreign key for the future relational database. The bottom handles of the **Category To Number** node are dedicated to access to a PMML (Predictive Model Markup Language[12]) dataflow. The PMML is a general format to encode models. Since, KNIME manages dataflow using basing type and data structures, it is sometimes necessary to add a PMML dataflow that helps define a datatype more strictly. There are no intrinsic categorical data in KNIME (they are treated as general string). For instance, a closer look at the *Transformed PMML Input* (bottom left handle of the node), shows that the columns to which the **Domain Calculator** node was applied (*Odor Type*, *Odor Ref*, and *Taste Ref*) are described into the PMML dataflow using all the enumerated values. Those PMML handles are there, because the operation of this node is likely to affect the detailed description of the data encoded into the PMML. In the present case, the PMML flow is created by the node: it did not exist previously.
• Connect the output of the *Data* output (the top right handle) of the **Category To Number** node to three **GroupBy** nodes.	The three **GroupBy** nodes are used to split the dataflow to populate three tables of the organoleptic database.
• Configure the first node by adding only the columns *Odor Ref ID* to the *Include* section (Figure 1.25).	The advantage is that the references and the organoleptics categories defined by the *Odor Type* field can be maintainded separately. If a correction is needed on a reference, it is done only once inside the dedicated reference table.
• Then click on the *Manual Aggregation* tab and include the column *Odor Ref* into the area to the right of the interface. Keep the default aggregation method which is *First* (Figure 1.26).	It enforces also the integrity of data. If a new substance is added, it must belong to one of the existing categories. If the substance requires a dedicated category it shall be created before, inside the dedicated odor type table.
• Configure the second node in analogy to the first one. The column *Taste Ref ID* should be alone inside the *Include* section. In the *Manual Aggregation* tab, select the column *Taste Ref* to be inside the area to the right-hand side of the interface. Keep the default aggregation method. Configure the third node like the two previous ones. The column *Odor Type ID* should be alone inside the *Include* section. Switch to the *Manual Aggregation* tab. Add the column *Odor Type* in the area to the right-hand side of the interface. Keep the default aggregation method.	

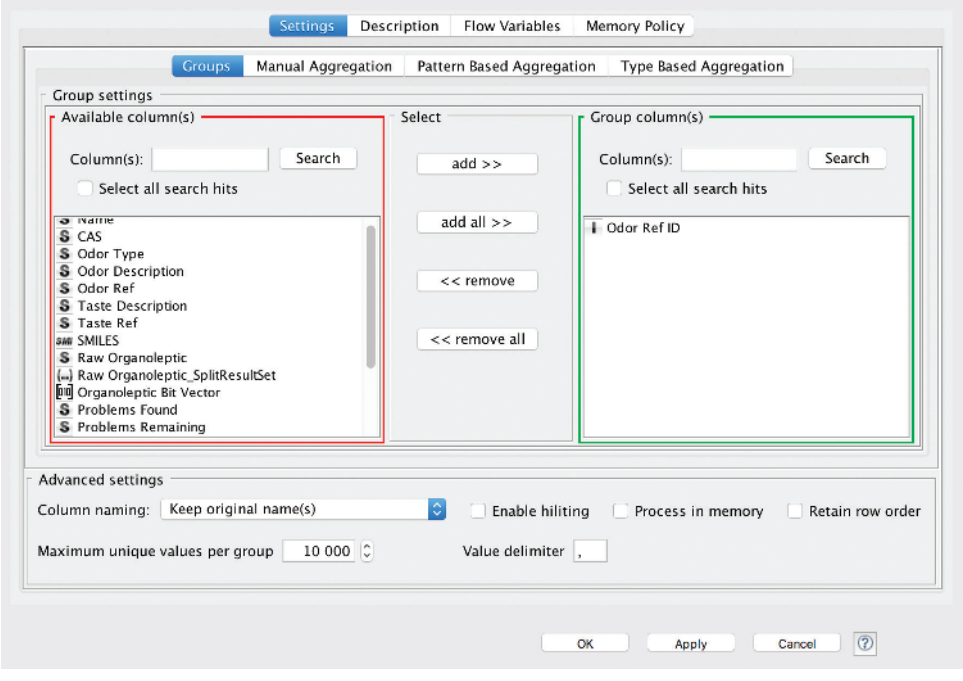


Figure 1.25 The main tab of the configuration interface of the GroupBy node using the Odor Ref ID for aggregation.

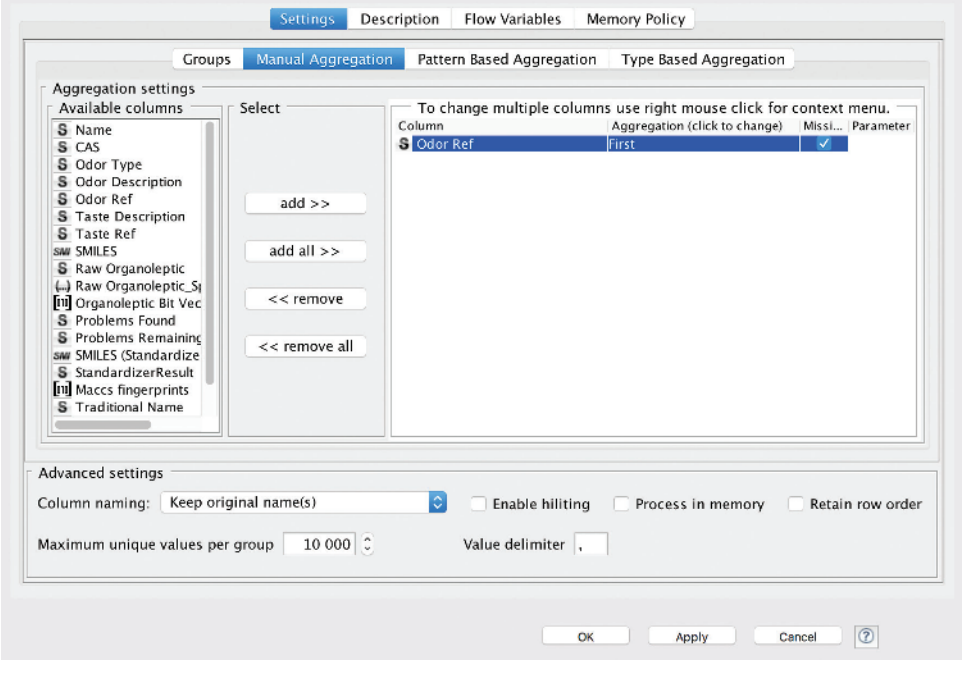


Figure 1.26 The Manual Aggregation tab of the configuration interface of the GroupBy node aggregating the Odor Ref column by keeping the first value.

(*Continued*)

Instructions	Comments

• Connect the output of each of the three **GroupBy** nodes to dedicated **CSVWrite** nodes.
• Configure the first **CSVWriter** node, the one connected to the dataflow concerning the odor references.
  ○ Inside the *Settings* tab (Figure 1.27), give a name for the output file: `OdorRefTable.csv`
  ○ Tick the box *Write column header* and select the option *Overwrite*.
  ○ Swith to the *Advanced* tab (Figure 1.28).
  ○ Use as *DataSeparator* a semicolon (";"). Select *System default* as *Line Endings*.
• Configure the **CSVWriter** node connected to the dataflow dedicated to taste descriptions references in the same way. Just change the name of the output file, it shall be: `TasteRefTable.csv`.
• Direct the dataflow dedicated to the odor type to the third **CSVWriter** node. Set the output file name to be: `OdorTypeTable.csv`. All other parameters should be identical to the other to **CSVWriter** nodes.
	The dataflows dedicated to odor descriptions references, taste descriptions references, and odor types must be writen to the hard drive into dedicated files: `OdorRefTable.csv`, `TasteRefTable.csv` and `OdorTypeTable.csv`
	The CSV format can be customized. It is recommended to use a semicolon as data separator. Some care must be given to this choice to avoid future problems with the generated file due to conventions and localization. For instance the comma "," can be confused with punctuation in a sentence or the French decimal separator.
	The line ending character is also important. The created data files are text files. However, Microsoft, Apple, and Linux do not encode the end of a line the same way. Using a wrong line ending can make the file difficult to manipulate. The safest choice, in the case that all further data treatment would be performed on the same system, is to use the local system conventions.

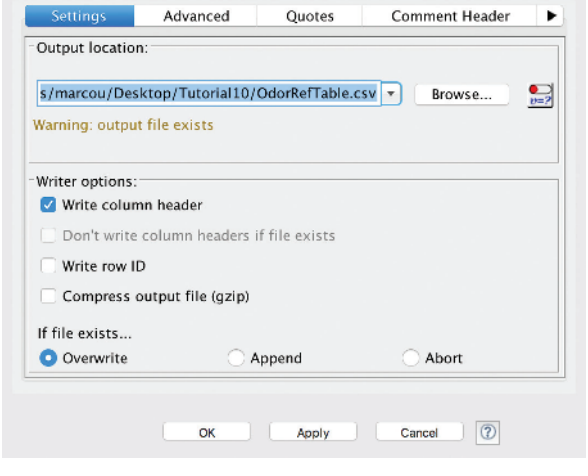


Figure 1.27 The Settings tab of the configuration interface of the CSVWriter node for output of the odor descriptions references.

Instructions	Comments
• Add a **Column Filter** node and connect its input to the data output of the **Category To Number** node (the upper right handle). • Configure the **Column Filter** node (Figure 1.29). The *Include* section should contain all the columns except the columns *Odor Type, Odor Ref, Taste Ref, Raw Organoleptic, Problem Founds, Problems Remaining,* and *StandardizerResult.*	The remaining columns of the data stream are not yet saved to a file. They should be dedicated to the substance table only. Therefore all information that is stored into separated tables (odor and taste descriptions references and odor types) must be discarded.
• Add a new **GroupBy** node to the workflow. • Select the field *SMILES (Standardized)* as the *Group column(s)* (Figure 1.30). • Select the *Keep original names* option as *Column naming*. • Switch to the *Manual Aggregation* tab (Figure 1.31). • Add the following fields into area to the right of the *Aggregation settings* with *List* option as *Aggregation*:	A closer examination of the standardized structures reveals that some chemical structures are duplicates. This happens because some substances have been added to the database at various concentrations of the perceptive substance, or substances got duplicated in the CAS. Other reasons can be errors of the chemical structure, essentially due to incorrect stereochemistry or stereoisomers that were confused with the racemic mixture.

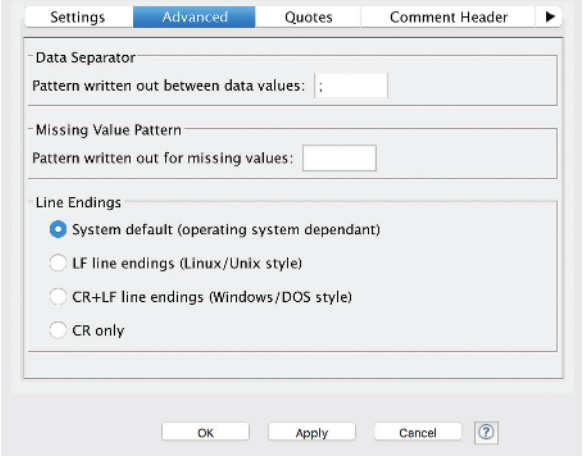


Figure 1.28 The Advanced tab of the configuration interface of the CSVWriter node for output of the odor descriptions references.

(*Continued*)

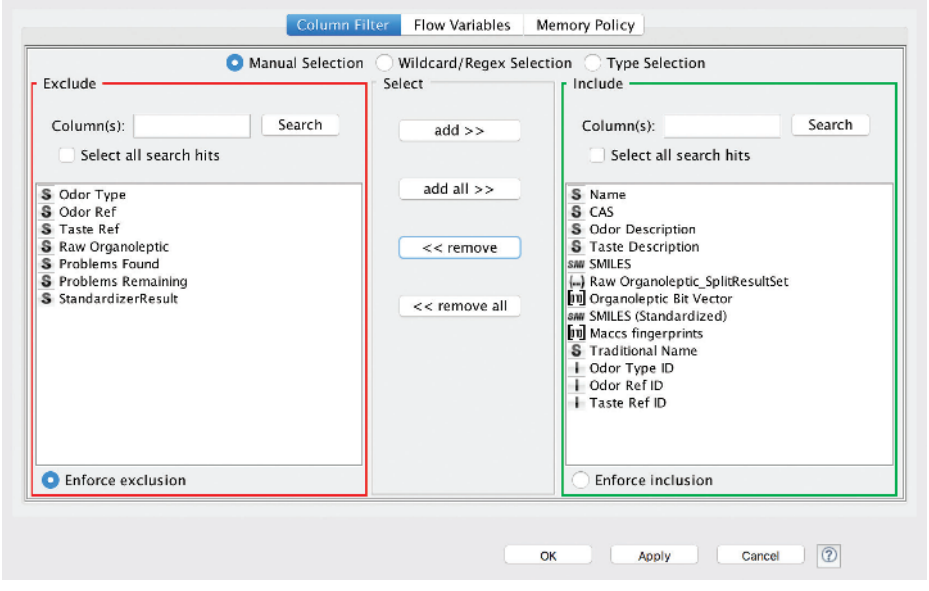


Figure 1.29 Configuration interface of the Column Filter used to process the dataflow that is planed to populate the chemical substance table of a database.

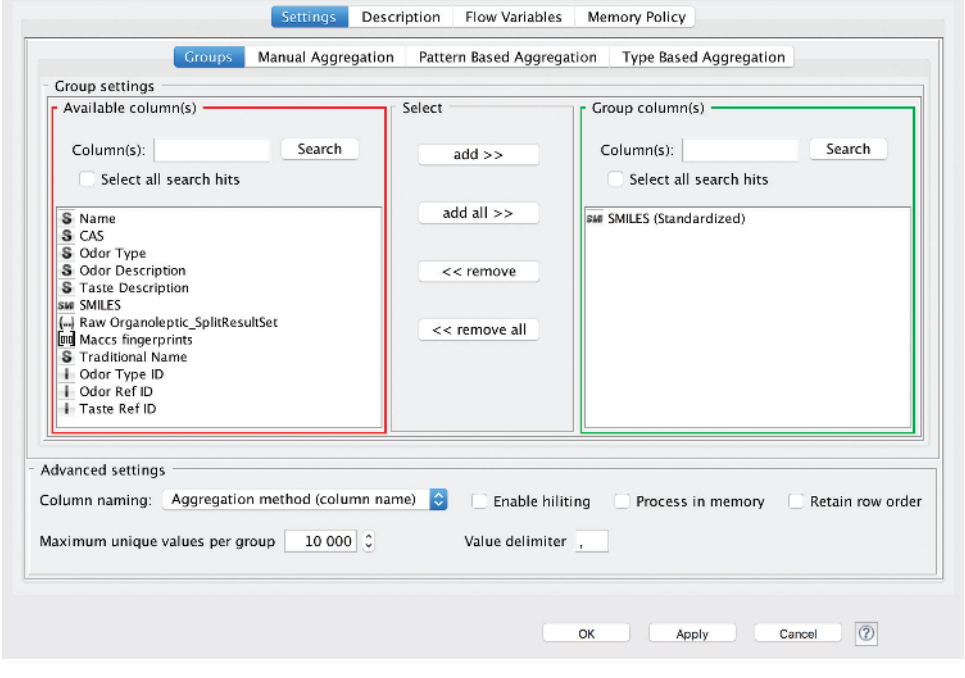


Figure 1.30 The GroupBy node grouping each line according to the chemical structures.

Instructions	Comments
○ *Name* ○ *CAS* ○ *Odor Description* ○ *Taste Description* ○ *SMILES* ○ *Odor Ref ID* ○ *Taste Ref ID* • Add the following fields into area to the right of the *Aggregation settings* with *First* option as *Aggregation*: ○ *Odor Type* ○ *Traditional Name* ○ *Odor Type ID* ○ *MACCS fingerprints* • Add the following fields into area to the right of the *Aggregation settings* with *Union* option as *Aggregation*: • *Raw Organoleptic_SplitResult*	If two items of the dataflow share the same chemical structure, the related information has to be merged. For fields such as *Name, CAS, Odor Decription, Taste Description, SMILES, Odor Ref ID*, and *Taste Ref ID* it is interesting to keep all information. Therefore, the data are aggregated into collections. For data specific to the chemical structure, the *Traditional Name, the MACCS fingerprints* the information is redundant and only the first encountered value is sufficient. For the *Odor Type* and *Odor Type ID* there should be some discrepancies. A duplicated chemical structure might have been categorized in different ways. So at this point, each discrepancy shall be analyzed and a decision must be taken by the user whether to discard a given chemical structure or to edit the *Odor Type* value. Finally the *Raw Organoleptic_SplitResult* may differ for the duplicates. However, any organoleptic description is equally interesting. Therefore, the lists of organoleptic terms are merged into one sole list using the *Union* action.

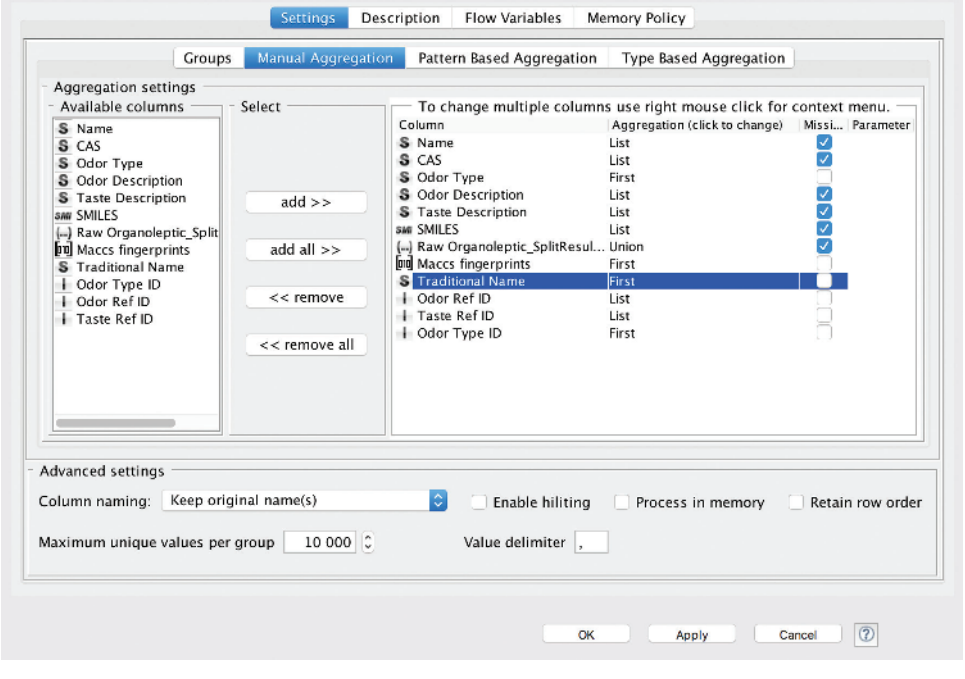


Figure 1.31 The GroupBy node Manual Aggregation interface managing the duplicate chemical structures in the dataset.

(*Continued*)

Instructions	Comments
• Add and configure a **Create Bit Vector** node (Figure 1.32). ○ Choose the option *Create bit vectors from a single collection column.* ○ Choose the column: *Raw Organoleptic_SplitResultSet.* ○ As *Output column* name it *Organoleptic Bit Vector.*	The *Raw Organoleptic_SplitResultSet* contains a synthetic description of the odors and tastes of the substance. If a substance appeared more than once, all descriptions have been merged into this field. Each word describing the organoleptic of a substance appears exactly once within a set. The set of all these words constitute a compendium of the description of chemical substances. A chemical substance is associated to a subset of these words. This relation can be efficiently encoded as a bit-vector. The bit-vectors generated by the node **Create Bit Vectors** are an efficient way to represent each chemical substance by its organoleptic properties. This is an alternative description to molecular fingerprint that are computed from the chemical structures.

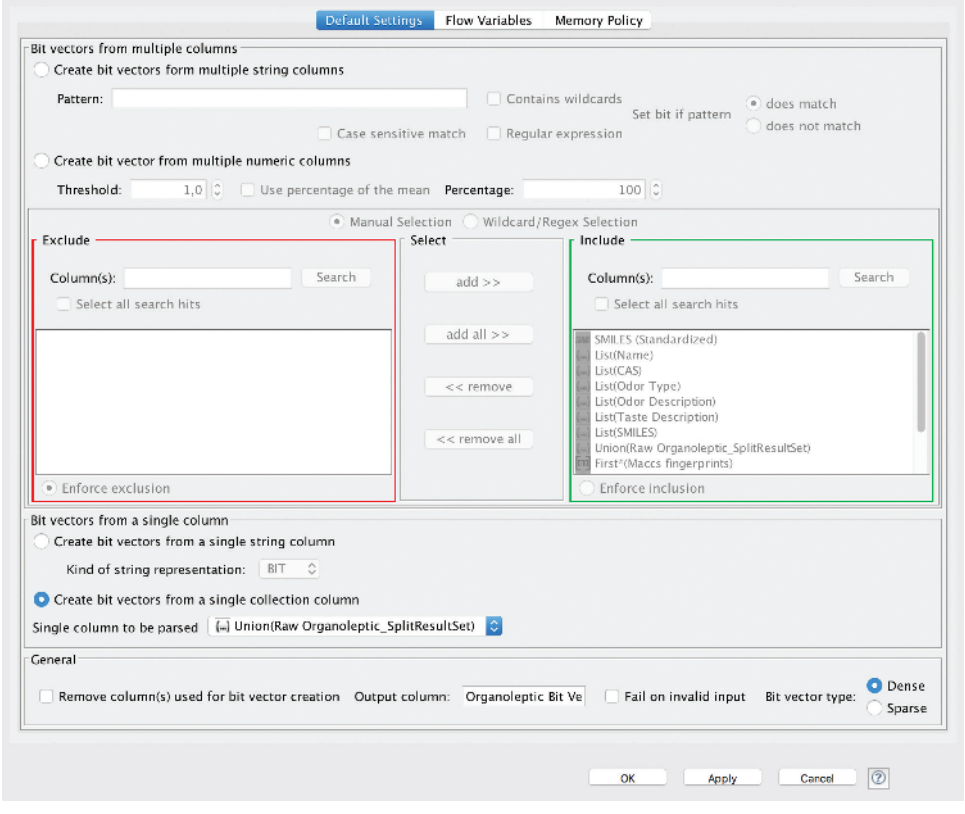


Figure 1.32 The Create Bit Vector node, providing an organoleptic description of the chemical substances as bitstrings.

Instructions	Comments
• Add a **Column Rename** to the workflow and configure it (Figure 1.33). • Double click on the following column titles, inside the Column Search area, to add them in the right-hand area of the interface: *SMILES (Standardized)*,	The dataflow contains fields that are formatted so they cannot be readily saved in a text file, such as a CSV. In particular, there are fingerprints represented as bit vectors. To save them, the value of each bit (on/off) must be translated into a 8 bit encoded character ("1"/"0").

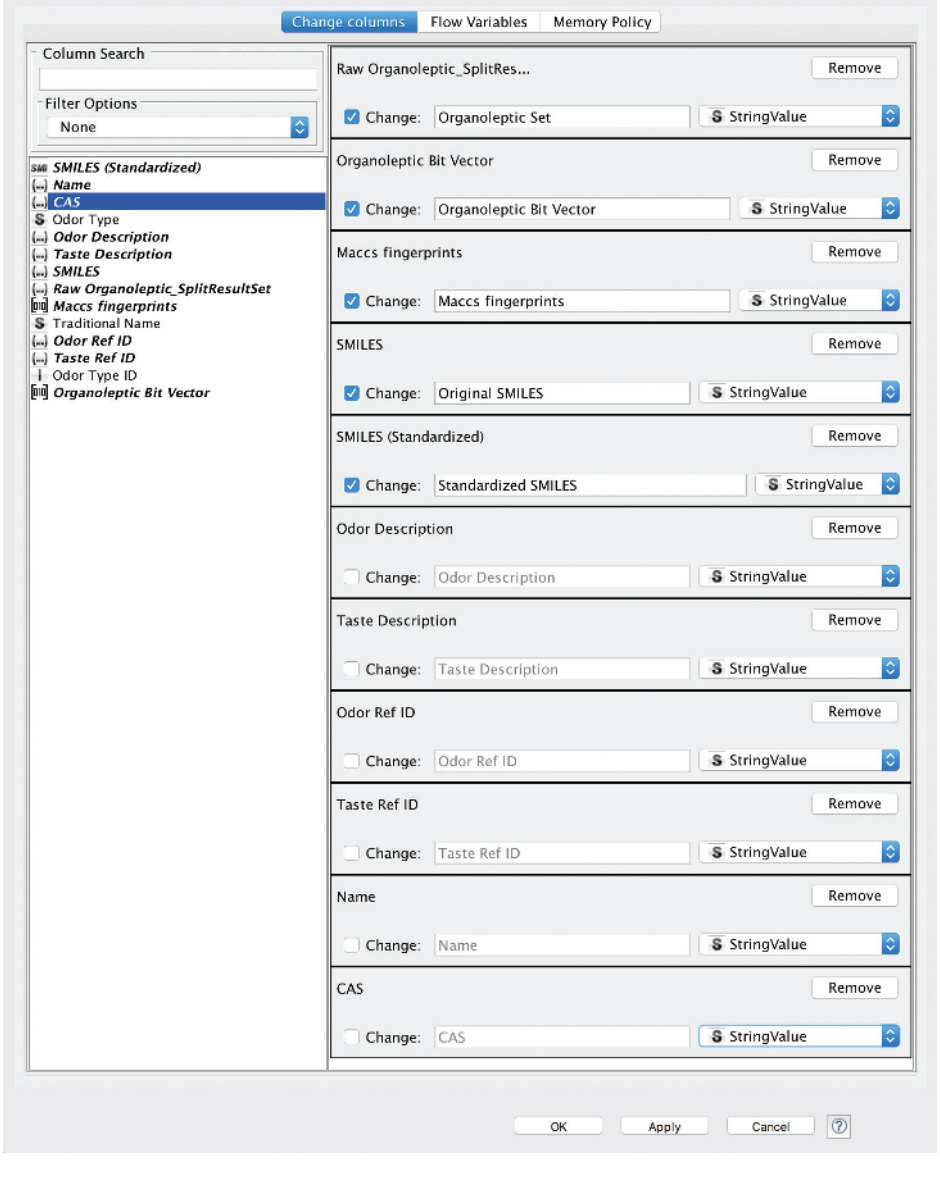


Figure 1.33 Configuration interface of the Column Rename node used to type cast the bit vectors and collections to standard strings.

(*Continued*)

Instructions	Comments

Name, *CAS*, *Odor Description*, *Taste Description*, *SMILES*, *Raw Organic_SplitResultSet*, *MACCS fingerprints*, *Odor Ref ID*, *Taste Ref ID*, *Organoleptic Bit Vector*.
• Tick the Change box in each of the added lines.
• Change the output type of each line to *StringValue*.
• Rename each line so the dataflow fields are more meaningful:
  ○ *Raw Organoleptic_SplitResultSet* as *Organoleptic Set*
  ○ *SMILES* as *Original SMILES*
  ○ *SMILES (Standardized)* as *Standardized SMILES*.
	Another difficult type are Collections. An example is the column *Raw Organoleptic_SplitResult*.
	A collection is a safe way to aggregate into one column lots of information. A collection encodes also the rules to split this information into separated columns. Yet in the present case, the collection content should be stored as a simple text.
	Here, the **Column Rename** is used to type cast those fields of the dataflow into standard strings that can be saved into a text file.
	Doing so, the data representation is less efficient and some information is lost. For instance, after the operation, it is no longer possible to safely split the initial collection into separated fields.

• Add a **CSV Writer** node to the workflow. • Configure the **CSV Writer** node like the others (Figure 1.27 and Figure 1.28). • Set the name of the output file as `SubstanceTable.csv`.	The dataflow dedicated to substance specific information is saved into the file `SubstanceTable.csv`.

Conclusion

The final workflow is shown in Figure 1.34. The process is very tedious but it is a prerequisite to chemical information management: databases, modeling, and so on. The raw input data file has been interpreted, scrutinized for errors, and particular caution was paid to the chemical structures. Finally, the substances information, odor types, odor and taste description references populate different CSV files with foreign keys. These files will be used to build a relational database dedicated to organoleptic properties of small organic compounds.

It is remarkable that many aspects of the dataflow are specific to this particular task of assembling a database of organoleptic properties of organic substances. It is extremely difficult to generalize the curation process. Each problem necessitates specific treatments. This is the main reason why solutions like flow programming languages, such as KNIME, are so efficient to deal with this task. A clear picture of the process alleviates the complexity of the data treatment. Many routine actions are repeatedly needed, although always in a different manner, during a curation process: splitting files, joining files, merging columns or rows, manipulating strings, and checking and standardizing the chemical structures. These benefit from abstract representation as configurable nodes in a workflow.

Finally, it is important to note that management of chemical structures is still an open question today. There is no satisfactory solution to this problem. The many valid possible representations of a compound as a graph still constitute a challenge today and there are no rules to decide which one is the more relevant in a particular context.

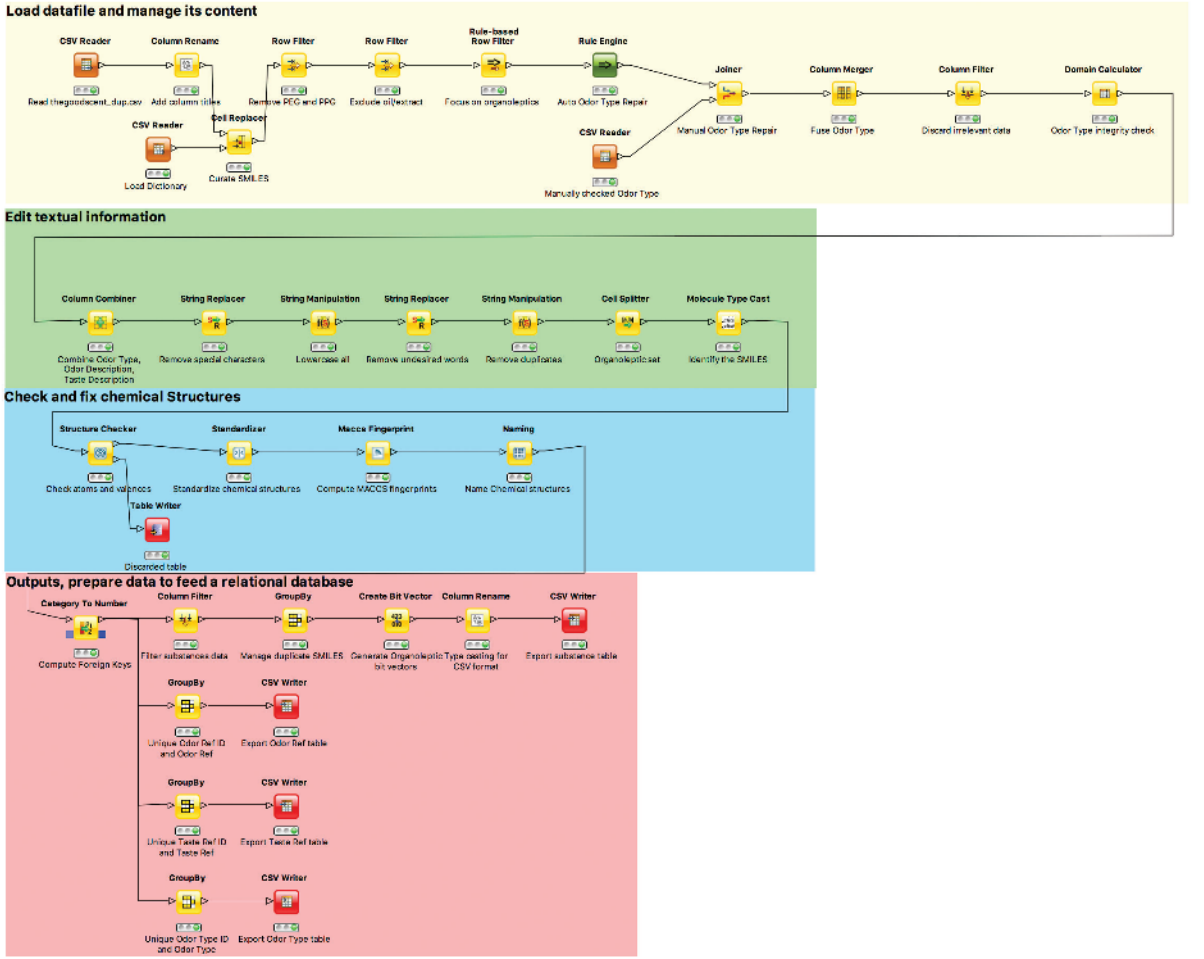


Figure 1.34 Overview of the KNIME workflow for the treatment of an organoleptic dataset aiming at populating a relational database.

Almost all software editors in chemoinformatics provide some automated solution for chemical structure curation. There are also some publications providing an easy to follow workflow [5]. It can give a false feeling of security and the problem is often disregarded. Checking and fixing chemical structures is a process that is an intrinsic part of a scientific process: it technically translates the opinion of an expert on a domain of chemistry. For instance, aggregating data on binders of a particular receptor protein, using the very same public sources, will lead to very different datasets depending on the expert working on it, with very different results concerning the outcome of projects based on these datasets.

Beyond the automatic processing of data, other solutions may be envisaged. An example of such alternative is the crowd sourced curation procedure that is used to validate the data in some main public databases [13].

References

1 Curry, E., A. Freitas, and S. O'Riáin, *The role of community-driven data curation for enterprises*, in *Linking enterprise data*. 2010, Springer. 25–47.
2 McNaught, A., *The IUPAC international chemical identifier*. Chemistry International, 2006, 12–14.
3 Heller, S.R. and A.D. McNaught, *The IUPAC international chemical identifier (InChI)*. Chemistry International, 2009, **31**(1), 7.
4 Oprea, T., et al., *On the propagation of errors in the QSAR literature*. EuroQSAR, 2002, 314–315.
5 Fourches, D., E. Muratov, and A. Tropsha, *Trust, but verify: on the importance of chemical structure curation in cheminformatics and QSAR modeling research*. Journal of Chemical Information and Modeling, 2010, **50**(7), 1189–1204.
6 Tropsha, A., *Best practices for QSAR model development, validation, and exploitation*. Molecular Informatics, 2010, **29**(6–7), 476–488.
7 Boser, B.E., I.M. Guyon, and V.N. Vapnik. *A training algorithm for optimal margin classifiers*. in *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, ACM.
8 Williams, A.J. and S. Ekins, *A quality alert and call for improved curation of public chemistry databases*. Drug Discovery Today, 2011, **16**(17), 747–750.
9 Hähnke, V.D., E.E. Bolton, and S.H. Bryant, *PubChem atom environments*. Journal of Cheminformatics, 2015, **7**(1), 1–37.
10 Feldman, H.J., et al., *CO: A chemical ontology for identification of functional groups and semantic comparison of small molecules*. FEBS letters, 2005, **579**(21), 4685–4691.
11 Durant, J.L., et al., *Reoptimization of MDL keys for use in drug discovery*. Journal of Chemical Information and Computer Sciences, 2002, **42**(6), 1273–1280.
12 Guazzelli, A., et al., *PMML: An open standard for sharing models*. The R Journal, 2009, **1**(1), 60–65.
13 Ekins, S. and A.J. Williams, *Reaching out to collaborators: crowdsourcing for pharmaceutical research*. Pharmaceutical Research, 2010, **27**(3), 393–395.