from SAS® to KNIME®

Author: Rosaria Silipo
## Table of Contents

Foreward ............................................................................................................................................. 7

Data and Script Structure .................................................................................................................. 9
  PROC and DATA Steps - Workflows and Nodes ............................................................................ 10
  LIBNAME - Workspace .................................................................................................................. 11

The KNIME Workbench .................................................................................................................... 12

PROC PRINT - display data table at node’s output port ................................................................. 13
DELETE - delete data table / increase the heap space ................................................................. 14

Input/Output ..................................................................................................................................... 15
  INFILE - File Reader ................................................................................................................... 16

INPUT ... datalines - Table Creator ............................................................................................... 17

connect to /disconnect from - Database Connector\Database Reader\ ........................................... 18

PROC IMPORT ... DBMS=EXCEL - Excel Reader (XLS) ............................................................ 19

Read SAS datasets from KNIME .................................................................................................... 20

Grouping / Sorting ............................................................................................................................ 21
  PROC SUMMARY/MEANS - GroupBy ......................................................................................... 22
  PROC TABULATE - Pivoting ......................................................................................................... 23
  PROC SORT - Sorter .................................................................................................................... 24
  PROC SORT nodup - GroupBy node with First/Last as aggregation method ............................... 25
  BY - Sorter .................................................................................................................................. 26

count - enumeration variable as ROWINDEX ................................................................................ 27
first.<variable> - finding the first row meeting a condition in a row group .................................. 28
nmiss() - count missing values in a row ....................................................................................... 29

Join/Concatenate ............................................................................................................................. 30
  SET - Concatenate and Concatenate (Optional in) ..................................................................... 31
  MERGE - Joiner .......................................................................................................................... 32
<table>
<thead>
<tr>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Math Operations</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Date&amp;Time Manipulation</strong></td>
<td>day() / year() / month() - Extract Date&amp;Time Fields</td>
</tr>
<tr>
<td></td>
<td>today() - Create Date&amp;Time Range</td>
</tr>
<tr>
<td></td>
<td>mdy() - String Manipulation – join() and String to Date&amp;Time</td>
</tr>
<tr>
<td><strong>Logical Operations</strong></td>
<td>if - then - Rule Engine</td>
</tr>
<tr>
<td></td>
<td>in() - Rule Engine – IN(...)</td>
</tr>
<tr>
<td></td>
<td>missing() - Rule Engine - MISSING and Missing Value</td>
</tr>
<tr>
<td></td>
<td>do ... end - Loops</td>
</tr>
<tr>
<td></td>
<td>macro - Metanode</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>String Manipulation</strong></td>
<td>right() - String Manipulation – stripEnd($column_name$)</td>
</tr>
<tr>
<td></td>
<td>left() - String Manipulation – stripStart($column_name$)</td>
</tr>
<tr>
<td></td>
<td>trim() - String Manipulation – strip($column_name$)</td>
</tr>
<tr>
<td></td>
<td>strip() - String Manipulation – replace($column_name$, ““,”’”’)</td>
</tr>
<tr>
<td></td>
<td>upcase()/lowcase()/propcase() - String Manipulation – lowerCase($col$), upperCase($col$), capitalize($col$)</td>
</tr>
<tr>
<td></td>
<td>substr() - String Manipulation – replace($column$, “old”, “new”) - and Cell Splitter By Position</td>
</tr>
<tr>
<td></td>
<td>scan() - Cell Splitter</td>
</tr>
<tr>
<td></td>
<td>countw() - String Manipulation – count($col$, “String”)</td>
</tr>
<tr>
<td></td>
<td>index ()/indexw() - String Manipulation – indexOf($col$, “String”)</td>
</tr>
<tr>
<td></td>
<td>put() - Column Rename &amp; other nodes</td>
</tr>
<tr>
<td></td>
<td>PROC FORMAT - Cell Replacer, String Replace (Dictionary), Rule Engine</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Rule Engine</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>int()</code>/<code>ceil()</code>/<code>round()</code>/<code>floor()</code></td>
<td>Math Formula – <code>round()</code>, <code>ceil()</code>, <code>floor()</code></td>
</tr>
<tr>
<td><code>max()</code>/<code>min()</code></td>
<td>Math Formula – <code>max_in_args()</code>, <code>min_in_args()</code></td>
</tr>
<tr>
<td><code>max()</code>/<code>min()</code></td>
<td>Math Formula (Variable)</td>
</tr>
<tr>
<td><code>mean()</code>/<code>sum()</code></td>
<td>Math Formula – <code>mean()</code> and <code>+</code></td>
</tr>
<tr>
<td>TRANSPOSE</td>
<td>Transpose</td>
</tr>
<tr>
<td>Basic Statistics</td>
<td></td>
</tr>
<tr>
<td>PROC FREQ</td>
<td>Crosstab and Statistics</td>
</tr>
<tr>
<td>Reporting</td>
<td></td>
</tr>
<tr>
<td>PROC REPORT</td>
<td>Data to Report and Report Designer (BIRT)</td>
</tr>
<tr>
<td>Include other script languages</td>
<td></td>
</tr>
<tr>
<td>PROC SQL</td>
<td>Database Query and Java, R, and Python Snippet</td>
</tr>
</tbody>
</table>
OK, I admit to being a 34 year SAS addict – 25 of them working directly for SAS – so when I was first introduced to KNIME Analytics Platform I was skeptical. Open source? Community supported? Everything in graphical workflows? FREE full function desktop Version? VERY different from the SAS I know! To me it sounded more like “bells and whistles” than “solid and robust” and I wondered why the Gartner group gave it “Cool Vendor Status” and what all those pharmaceutical companies saw in it. Now, after using the product myself for several major projects, I can truly say that KNIME is very powerful indeed!

Working with Rosaria Silipo meant that I could climb the software learning curve very quickly – she knows both products well, and her publishing this booklet means that YOU can now get up to speed very quickly.

I appreciate the fact that Rosaria has focused on comparing CORE SAS functionality to KNIME. Any experienced SAS user knows that a SAS application is, at the end of the day, a program that combines data steps, procs, macro language, and a front end – combined with the knowledge of how to USE them in an application created for a specific business area. Looking at core functionalities is how we as SAS users most effectively learn something new.

There are some basic similarities: both SAS and KNIME pull data sensibly into memory and optimize the interplay between I/O and processors and clusters, because both suppliers know that at some point, no matter how fast in-memory manipulation is big data will require loading and processing. Both fundamentally understand that it’s not just about statistics and reports, it’s about accessing data – any data – and manipulating that data to get it back out in a usable form. Both believe that a platform that integrates across desired components is the right way to go.

Both also try to provide as many techniques as possible in the analytics area. But while SAS employs its large, dedicated team of thousands of developers to generate everything possible in its proprietary infrastructure (and only once it’s developed and rolled out is it, in fact, possible), KNIME takes the approach that there will ALWAYS be some unforeseen requirement for an esoteric or specialized technique, so it simply insures that it’s easy to pull that technique directly from another source – whether that be R or Weka or you name it.

Both SAS and KNIME allow beginners to start out small and focused while learning. But once up to speed, you really need the ability to scale up quickly to accommodate many users and a lot of data without having to rewrite programs; the two platforms also have this positive trait in common. But there are a few major philosophical differences that, when understood, will help you get going with KNIME faster.

Most important is probably the paradigm difference: fundamentally, SAS is a line-oriented scripting language. Everything on top of base SAS, including all the specific applications successfully developed in the last several years, generates that line-oriented code at the end of the day.

But let’s focus on actually building a SAS program or a KNIME workflow to achieve a particular task. In SAS, your approach is always effectively iterating through data steps and procs back and forth until you finally have what you need. You manage the programming logic with macros, and can surface that application via a variety of different interfaces. KNIME is a node-workflow oriented approach – for absolutely EVERYTHING. You don’t have to learn an application interface, then the data step language, then the PROC syntax, then a MACRO language for packaging code, and possibly even one of the application languages (Java, .net or proprietary SAS) to build and surface programs to end users – it’s ALL done in nodes and workflows. That means in the beginning you may have to search for the correct nodes and combinations thereof to achieve what you need to do, but once you are up and running you can quickly mix and match the different types of nodes as needed.
Another major difference with KNIME: what you program = what you run = what you document = what you can visualize to your audience. Once you’ve built a workflow, it’s available for all required purposes – you don’t have to jump over to a graphics tool and mock up a flow or draw a picture to explain what’s actually going on behind that exotic-looking macro code!

Probably the biggest difference though is in the approach to packaging and expanding your application. The KNIME concept of a metanode – taking a series of linked nodes in a workflow and creating ONE reusable node out of them, with all inputs/outputs automatically defined – is extremely powerful.

Let’s be clear what KNIME is not: it is not attempting to be the ultimate in-memory BI reporting packages; rather, KNIME makes sure that, if needed, you can surface everything through your favorite reporting tool, whatever that is. KNIME is also not trying to be the end-all in MIS batch reporting, although the open source BIRT platform provided with KNIME will get you started. And while KNIME already has a huge number of business-oriented examples (check out the website!), it doesn’t (yet?) strive to make end-to-end, complete applications. What KNIME does do is make it easy to include and reuse other applications, whether that be from the popular R package or your favorite coding language such as Java. You can also easily CALL other applications – even SAS – to build exactly what you require. And one last tip from me: start with this booklet and your favorite SAS dataset. Using the special SASB7DAT node, KNIME will read that data in with just one click – which is not only super-simple and fast but also gives you an edge in allowing you to start learning with data that you already know.

I hope you enjoy using KNIME as much as I do!
And in the interest of total transparency – I am so convinced I have joined the board of KNIME.com.

Phil Winters
Strategic Advisor, Peppers & Rogers Group
CIAgenda
Data and Script Structure
A SAS Base analysis is performed by means of a script, which consists mainly of DATA steps and PROC steps.

A number of SAS products use now a more visual approach. However, the graphic layout is translated into the corresponding SAS Base script before execution.

**Example:**

```
DATA xyz
   set x y z;
RUN;

PROC TABULATE data= xyz;
   class = education, gender;
   var income;
   table gender, education * N;
RUN;
```

**Note.** Nodes have four possible status displayed by a little traffic light under the node itself:
- Not yet configured  -> red light
- Configured but not executed  -> yellow light
- Successfully executed  -> green light
- Executed with error  -> red light with cross

For more details about KNIME Analytics Platform, check:
- R. Silipo, *KNIME Beginner’s Luck*, KNIME Press, 2018
- R. Silipo, J. Prinz, *KNIME Advanced Luck*, KNIME Press, 2018
### LIBNAME

<table>
<thead>
<tr>
<th>SAS Base</th>
<th>KNIME Analytics Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>libname</strong> <code>&lt;lib_name&gt; &lt;lib_path&gt;;;</code></td>
<td><strong>Workspace</strong> defines the folder where all workflows and intermediate data are saved. The path to the workspace is selected at the very beginning, after starting <code>KNIME Analytics Platform</code>.</td>
</tr>
</tbody>
</table>

**Example:**

```r
libname mylib 'C:\Data\Course';
```

---

**Workspace**

You can still change the workspace after KNIME has been launched, by selecting “File” in the top menu and then “Switch Workspace”.

![Switch Workspace](image)

![Workspace Selection](image)
The KNIME Workbench

Start KNIME Analytics Platform from the desktop or from the installation folder. The KNIME workbench opens, which includes: “Workflow Editor”, “KNIME Explorer”, “Node Repository”, “Node Description”, “Outline”, and “Console”.

“KNIME Explorer” shows the list of currently available workflow for the selected workspace.

“Node Repository” panel contains all available nodes. A “Search” box is available on top of this panel to search for nodes.

“Workflow Editor” in the center allows for the creation and editing of workflows.

“Workflow Coach” offers hints for the next node to insert after the selected one in the Workflow Editor.

“Node Description” shows text describing the node task and configuration settings, for the selected node either in the “Workflow Editor” or in the “Node Repository” panel.

The “Outline” panel offers an overview of the workflow and the “Console” panel shows execution messages.

KNIME workflows are created by drag&drop of nodes from the “Node Repository” panel into the “Workflow Editor”.

Nodes are connected to each other through their input and output ports. Just click the output port of the first node and release on the input port of the second node. Nodes that have been just created have a red light status: not yet configured. To configure a node, right-click the node and select the option “Configure” or alternatively double-click the node. The node “Configuration” window opens. Configure the node. If the configuration is successful, the node status changes to a yellow traffic light. The node is now configured, but not yet executed. To execute the node, right-click the node and select the “Execute” option. If the execution is successful, the node changes its status to a green light.
### PROC PRINT

**SAS Base**

```
PROC PRINT
   data=<dataset-name>
   <options>;
run;
```

**Example:**

```
PROC PRINT data=xyz;
RUN;
```

### display data table at node's output port

**KNIME Analytics Platform**

The data tables produced after node execution are always available. To see them:
- Right-click the node in the workflow
- Select the last option in the context menu

```
Row ID  Column Header  Integer data type  String data type
-------  -------------  ------------------  ---------------
1        Name          Type               Sex             Age
2        State-gov     Elementary         Married         39
3        Self-emp-III  H-School          Married         50
4        Private       H-School          Married         53
5        Private       Florida           Married         55
6        Private       MA                 Married         62
7        Private       MD                 Single          60
8        Private       TX                 Single          65
9        Private       CA                 Single          55
10       Private       MO                 Single          60
11       Private       LA                 Single          65
12       Private       NY                 Single          55
13       Private       DC                 Single          60
```

**Note.** Some nodes, like plotting and modeling nodes, have also a more complex “View” function. The option leading to this “View” is usually displayed in the middle of the context menu.
### DELETE

<table>
<thead>
<tr>
<th>SAS Base</th>
<th>KNIME Analytics Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>proc datasets;</td>
<td></td>
</tr>
<tr>
<td>delete &lt;dataset-name&gt;;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>proc datasets;</td>
<td></td>
</tr>
<tr>
<td>delete xyz;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**delete data table / increase the heap space**

KNIME memory management ensures disposal of tables that are not used anymore, so that no equivalent to the `DELETE`-command in SAS is needed.

Memory to be used by KNIME Analytics Platform is allocated at installation time. If you wish to increase the memory available to KNIME Analytics Platform just change option `–Xmx` in the knime.ini file in the KNIME installation folder.
Input/Output
INFILE

SAS Base

data <dataset-name>
infile <filepath> delimiter=<char>;
input <col_1> <col_2> ... <col_n>;
run;

Example:

data xyz;
  infile 'C:\data\CRM.txt' delimiter=';';
  input CustomerID ContractID date_start date_end amount;
  format date_start date9.;
  format date_end date9.;
run;

Note 1. There are many possible reading formats. Basic formats are: Integer, Double, and String. A date must be read as String and later on converted to Date&Time type with a “String To Date&Time” node.

Note 2. KNIME has also a specific “CSV Reader” node dedicated to reading CSV files.
INPUT ... datalines

**SAS Base**

```sas
data <dataset-name>;
input [position] <col_1> [format] [position] <col_2> [format] ... [position]<col_n> [format];
datalines;
<data>
run;

Example:

data xyz;
  input ContractID $ CustID $ nr @17 ReceivedDate date9. @27 Price;
  format ReceivedDate date9.;
datalines;
A123 1ABCR 34 28jul1998 45.00
B123 2ABCDF 98 20jul1997 345.88
C123 3ABCD5 121 19may1999 100.99
D123 4A 43 10aug1999 27.37
E123 5ABC 80 16aug1999 9.65
F123 6ERA 6 18jun1999 19.67
G123 7ABC 383 09jan2000 14.89
H123 8ABC 26 03aug2000 44.50
run;
```

**Table Creator**

**KNIME Analytics Platform**

Note. There are many possible reading formats. Basic formats are: Integer, Double, and String. A date must be read as String and later on converted to Date&Time type with a “String To Date&Time” node.
connect to /disconnect from

**SAS Base**

```sas
proc sql;
    connect to <database-name> as <connection-alias> ([connection credentials]);
    create table <table-name> as select * from connection to <database-name> ([sql statements]);
    disconnect from <connection-alias>;
quit;
```

**Example:**

```sas
proc sql;
    connect to oracle as dblink (user='xxxx', password='xxxx', path='ORCL');
    create table xyz as select * from connection to oracle (select * from employees);
    disconnect from dblink;
quit;
```

**Database Connector\Database Reader...**

**KNIME Analytics Platform**

**Note 1.** The approach to database reading is modular. The three phases (connect, SELECT, extract) can be run all together in a Database Reader node or built step by step with a Database Connector, a Database Table Selector, and a Database Connection Table Reader node. Other Database nodes are available to help build the SQL query.

**Note 2.** A “Database Disconnector” node is not necessary. The “Database Reader” node connects to the database, reads the table by means of a SELECT query, and disconnects.

**Note 3.** Database nodes only builds the SQL query, do not execute it. Only the final node extracting the data executes the SQL query.
**PROC IMPORT ... DBMS=EXCEL**

**SAS Base**

**PROC IMPORT**

OUT= <dataset-name>
DATAFILE= <file path>
DBMS=EXCEL REPLACE;
SHEET=<sheet-name>;
GETNAMES=YES/NO;
MIXED=YES/NO;
USEDATE=YES/NO;
SCANTIME=YES/NO;
RUN;

**Example:**

**PROC IMPORT**

OUT= work.data1
DATAFILE= "C:\Months.xls"
DBMS=EXCEL REPLACE;
SHEET="Tabelle1";
GETNAMES=YES;
MIXED=YES;
USEDATE=YES;
SCANTIME=YES;
RUN;

**Note.** Possible reading formats are: Integer, Double, and String. A date must be read as String and later on converted to Date&Time type with a “String To Date&Time” node.
KNIME Labs has a dedicated node to read SAS datasets, named “SAS7BDAT(DSREAD)”. To install this extension from the KNIME Labs, in the top menu:

- Click “Help”
- Select “Install New Software …”
- Select the update site
- Open the “KNIME Labs Extensions” category
- Select “KNIME SAS7BDATA Reader (Windows Only)”
- Click “Next” and follow the installation instructions
Grouping / Sorting
## PROC SUMMARY/MEANS

### SAS Base

```
proc summary data= <dataset-name>;
  class = <col_1>, <col_2>, ... <col_n>;
  var <col_x>;
  OUTPUT <OUT=SAS-data-set>;
  <output-statistic-list>
    <statistic(variable)=new-var-name>
    <statistic(variable)=new-var-name>
run;
```

**Example:**

```
PROC SUMMARY DATA= xyz;
CLASS ContractID;
VAR Price;
OUTPUT OUT=example1
  SUM(Price) = tot_sales
  MEAN(Price) = mean_sales
  NMISS(Price) = bad_sales
;
RUN;
```

### KNIME Analytics Platform

The “GroupBy” node is configured via two tabs:
- “Groups” defines the group columns
- “Options” defines the aggregation variables and the aggregation methods

Columns selected in tab “Groups” correspond to columns selected by “class”. Columns selected in tab “Options” correspond to columns selected by “var”. Aggregation methods selected in tab “Options” correspond to: “<statistic(variable)>”.

---

![Diagram](image)
PROC TABULATE

SAS Base

```sas
proc tabulate data= <dataset-name> out=<dataset-name>;
  class = <col_1>, <col_2>, ... <col_n>;
  var <col_x>;
  table <col_i> * <stat_i>, ..., <col_j> * <stat_j>;
run;
```

Pivoting

KNIME Analytics Platform

The “Pivoting” node is configured via three tabs:
- “Groups” defines the group columns (final row IDs)
- “Pivots” defines the pivoting columns (final column headers)
- “Options” defines the aggregation variables and the aggregation methods

Columns selected in “Groups” and “Pivots” correspond to columns selected by “class”. Columns selected in “Options” correspond to columns selected by “var”. Aggregation methods selected in “Options” correspond to “<stat_i>”.

The “Pivoting” node produces three output tables: the pivot table and the total values for columns and rows.

Example:

```sas
proc tabulate data= xyz out=xyz;
  class = education, gender;
  var number;
  table gender, education * N;
run;
```
### PROC SORT

#### SAS Base

```sas
PROC SORT DATA=<input-data-set> OUT=<output-data-set>;
   BY <key> ;
RUN;
```

**Example:**

```sas
PROC SORT DATA=xyz OUT=xyz2 ;
   BY CustID, ContractID, DateReceived ;
RUN ;
```

### Sorter

#### KNIME Analytics Platform

**Note.** The “Sorter” node has no option to filter out duplicates. To filter out duplicates you need to use a “RowID”, a “Joiner” node, or a “GroupBy” node.
PROC SORT nodup DATA=<input-dataset> OUT=<output-dataset>;
  BY <key> ;
RUN ;

Example:

PROC SORT nodup DATA=xyz OUT=xyz2 ;
  BY sex;
RUN ;

GroupBy node with First/Last as aggregation method

The “GroupBy” node is configured via two tabs:
- “Groups” defines the group columns
- “Options” defines the aggregation variables and the aggregation methods

Aggregation method „First” or „Last” keeps only the first or last value of the selected column for the defined group. All other values for the group are removed. As a result the group is represented by one row only with the first or last value for the selected variable.
<table>
<thead>
<tr>
<th>BY</th>
<th>SAS Base</th>
<th>Sorter</th>
<th>KNIME Analytics Platform</th>
</tr>
</thead>
</table>
|    | data <dataset-name>;
    | set <dataset-name>;
    | by <col_1> <col_2> … <col_n>;
    | run;                  |

**Example:**

```
data datal;
  set xyz;
  by class gender;
run;
```

Aggregation nodes, like “GroupBy” or “Pivoting”, sort the output data by group in alphabetical order.

If you need a different kind of sorting for the output data or if you need any operation to run on groups in a certain order, you will need to use a “Sorter” node after or before the desired node.
To just count the number of rows in a value group, the “Value Counter” node or the “GroupBy” node with aggregation method “count” would be enough. Besides that, every node has a flow variable named “ROWINDEX” available. This flow variable contains the number of the current row. “ROWINDEX” can be used for example in a “Math Formula” node to create the enumeration column. A similar flow variable is “ROWCOUNT” and contains the total number of rows of the input data table. An enumeration variable can also be obtained with a “Java Snippet” node.

Example:

```sas
data xyz;
  set xyz;
  count + 1;
run;
```

```java
// Your custom imports:
10 // system imports
15 // your custom variables:
int count = 0;
16 // expression start
21 // enter your code here:
22 count++;
23 // expression end
```
To find the first row meeting a condition, we can use a “Rule Engine” node. The “Rule Engine” node evaluates a condition on each row and produces a consequent output value. If input data have been previously sorted, the counter=1 will happen at the beginning of each group.

If we want to get an enumeration inside each group, we need a group loop cycle, starting with a “Group loop Start” node and ending with a “Loop End” node.

Example:

```sas
data dataset-name;
    set dataset-name;
    count + 1;
    by col_1 col_2 ... col_n;
    if condition then count = 1;
run;
```

```sas
data xyz;
    set xyz;
    count + 1;
    by sex;
    if first.sex then count = 0;
run;
```
### nmiss()

#### count missing values in a row

<table>
<thead>
<tr>
<th><strong>SAS Base</strong></th>
<th><strong>KNIME Analytics Platform</strong></th>
</tr>
</thead>
</table>
| data <dataset-name>;
  set <dataset-name>;
  <new_column> = nmiss(<col_1>, <col_2>, ..., <col_n>); run; | This can be obtained in many ways. |
| **Example:** | |
| data xyz;
  set data1;
  nr_missing = nmiss(sex, CustID, ContractID);
run; | |

#### Note
- Similarly, you can use a “Missing Value” node to assign a special value to missing values and then count them with a “Rule Engine” and a “Math Formula” node.

- “For small data sets, we could use a “Transpose” node + “GroupBy” node with “Missing value count” as aggregation method.

#### Note
- If you are working on many columns, in tab “Groups” you should use “Pattern Based Aggregation” with regex “.*” or “Type Based Aggregation” including all types. It would be too tedious to add all columns manually in the “Manual Aggregation” tab.
Join/Concatenate
**SET**

<table>
<thead>
<tr>
<th>SAS Base</th>
<th>KNIME Analytics Platform</th>
</tr>
</thead>
</table>
| **Concatenate**

```
data <dataset-name>;
   set <dataset1> <dataset2>;
run;
```

**Example:**

```
data xyz;
   set x y z;
run;
```

**Note 1.** The “Concatenate (Optional in)” node has a maximum of 4 input. If we need to concatenate more than four data sets, we need to use more “Concatenate (Optional in)” nodes.

**Note 2.** The simpler “Concatenate” node has only two input ports and can concatenate only two data tables.
**MERGE**

### SAS Base

```sas
data <dataset-name>;
  merge <dataset1>( in=<label1> )
    <dataset2>( in=<label2> );
by <column-name>;
run;
```

### KNIME Analytics Platform

**Note 1.** “Duplicate Column Handling” is defined in the “Column Selection” tab. Here you can choose between: skip duplicate columns, stop the node execution if a column with the same name is found in both tables, or append a suffix to the name of duplicate columns.

**Note 2.** There is no merging option in handling duplicate columns. That is: unlike “merge” in SAS, the “Joiner” node in KNIME does not fill the missing values in one column in one table with the non-missing values from the column with the same name in the other table. To merge values from two columns in the joined data table, you will need to use the “Column Merger” or the “Rule Engine” node.

**Example:**

```sas
data xyz;
  merge sales( in=a )
    contracts( in=b );
  by id;
run;
```
### WHERE

**SAS Base**

```sas
proc ... data=<dataset-name>
    ... where <condition>; run;

data <dataset-name>
    set <dataset-name>;
    where <condition>; run;

Example:

proc means data=xyz;
    where x<5 and y>3;
run;

data xyz;
    set xyz;
    where ContractID=123;
run;
```

**KNIME Analytics Platform**

![Row Filter Configuration](image)

**Note.** You need to build the `<where>` condition in the configuration window of the “Row Filter” node. Notice that it is also possible to EXCLUDE the rows matching the pattern (right side of the node configuration window).
### KEEP/DROP

<table>
<thead>
<tr>
<th>SAS Base</th>
<th>Column Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>proc ... data=&lt;dataset-name&gt;</code></td>
<td><img src="image" alt="Column Filter" /></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td><strong>keep</strong> <code>&lt;col1&gt; &lt;col2&gt; ... &lt;coln&gt;;</code></td>
<td></td>
</tr>
<tr>
<td><code>[drop &lt;col1&gt; &lt;col2&gt; ... &lt;coln&gt;;]</code></td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><code>data &lt;dataset-name&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>set &lt;dataset-name&gt;</code></td>
<td></td>
</tr>
<tr>
<td><strong>keep</strong> <code>&lt;col1&gt; &lt;col2&gt; ... &lt;coln&gt;;</code></td>
<td></td>
</tr>
<tr>
<td><code>[drop &lt;col1&gt; &lt;col2&gt; ... &lt;coln&gt;;]</code></td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>proc means data=xyz;</code></td>
<td></td>
</tr>
<tr>
<td><strong>keep</strong> <code>x, y;</code></td>
<td></td>
</tr>
<tr>
<td><code>[drop z;]</code></td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><code>data xyz;</code></td>
<td></td>
</tr>
<tr>
<td><code>set xyz;</code></td>
<td></td>
</tr>
<tr>
<td><strong>keep</strong> <code>ContractID, CustID, Price;</code></td>
<td></td>
</tr>
<tr>
<td><code>[drop nr,DateReceived;]</code></td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
String Manipulation
**right()**

### SAS Base

```sas
data <dataset-name>;
  set <dataset-name>;
  <new column> = right(<column-name>);
run;
```

**Example:**

```sas
data xyz;
  set data1;
  CustID = right(CustID);
run;
```

### KNIME Analytics Platform

**String Manipulation – stripEnd($column_name$)**

**Note.** Argument of the function can be more than one column, if separated by commas.
### SAS Base

```
data <dataset-name>;
  set <dataset-name>;
  <new column> = left(<column-name>);
run;
```

**Example:**

```
data xyz;
  set data1;
  CustID = left(CustID);
run;
```

### KNIME Analytics Platform

**Note.** Argument of the function can be more than one column, if separated by commas.
<table>
<thead>
<tr>
<th>trim()</th>
<th>String Manipulation – strip($column_name$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SAS Base</strong></td>
<td><strong>KNIME Analytics Platform</strong></td>
</tr>
</tbody>
</table>
| \[
\begin{align*}
\text{data } & \text{<dataset-name>;} \\
\text{set } & \text{<dataset-name>;} \\
\text{<new column>} & = \text{trim(<column-name>);} \\
\text{run;}
\end{align*}
\] | ![String Manipulation](image) |

*Example:*

```sas
\[
\begin{align*}
\text{data xyz;} \\
\text{set data1;} \\
\text{CustID} & = \text{trim(CustID);} \\
\text{run;}
\end{align*}
\]```

*Note.* Argument of the function can be more than one column, if separated by commas.
### strip()

**SAS Base**

```sas
data <dataset-name>;
set <dataset-name>;
<new column> = strip(<column-name>);
run;
```

**KNIME Analytics Platform**

![String Manipulation in KNIME](image)

**Note.** SAS function “strip()” is a particular instance of KNIME function “replace()”. In particular it is the instance where all blank spaces are replaced by nothing.

**Example:**

```sas
data xyz;
set data1;
CustID = strip(CustID);
run;
```
**String Manipulation – lowerCase($col$), upperCase($col$), capitalize($col$)**

<table>
<thead>
<tr>
<th>SAS Base</th>
<th>KNIME Analytics Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>data &lt;dataset-name&gt;;</code>&lt;br&gt;  <code>&lt;new column&gt; = upcase(&lt;column-name&gt;);</code>&lt;br&gt;  <code>&lt;new column&gt; = lowcase(&lt;column-name&gt;);</code>&lt;br&gt;  <code>&lt;new column&gt; = propcase(&lt;column-name&gt;);</code>&lt;br&gt;  <code>run;</code></td>
<td><img src="image" alt="KNIME String Manipulation" /></td>
</tr>
</tbody>
</table>

**Example:**

```sas
data xyz;
  ContractID = upcase(ContractID);
  ContractID = lowcase(ContractID);
  ContractID = propcase(ContractID);
run;
```
**substr()**

**SAS Base**

1. \( \text{substr(<string>, <position>, <length>)} = \text{characters-to-replace} \)

OR

2. \(<\text{variable}> = \text{substr(<string>, <position>, <length>)} \)

**Example:**

1. \( a='KNIME'; \)
   \( \text{substr}(a,4,1)='F'; \)

RESULT: \( a = "KNIFE" \)

OR

2. \( \text{newStr='Information Mining';} \)
   \( \text{newStr1=substr(newStr,1,4);} \)
   \( \text{newStr2=substr(newStr,12,6);} \)

RESULT: \( \text{newStr1 = "Info" newStr2 = "Mining"} \)

**KNIME Analytics Platform**

**Note.** KNIME has 3 “Cell Splitter” nodes: “Cell Splitter by Position” based on indices, “Cell Splitter” based on delimiter matching, and “Regex Split” based on Regular Expression matching.
data <dataset-name>;
   set <database-name>;
   <word_string> = scan(<string>, <int>, <delimiter>, <modifier>);
   end;
run;

Example:

data xyz;
   set data1;
   delim = ',';
   modif = 'mo';
   nwords = countw(string, delim, modif);
   do count = 1 to nwords;
      word = scan(string, count, delim, modif);
      output;
   end;
run;
countw()  

String Manipulation – count($col$, “String”)

**SAS Base**

```sas
data <dataset-name>;
    set <database-name>;
    <int> = countw(<string>, <delimiter>, <modifier>);
    end;
run;
```

**Example:**

```sas
data xyz;
    set data1;
    delim = ',';
    modif = ''; 
    nwords = countw(custID, delim, modif);
    do count = 1 to nwords;
        word = scan(string, count, delim, modif);
        output;
    end;
run;
```

**KNIME Analytics Platform**

![Image showing the KNIME Analytics Platform interface for string manipulation](image-url)
### SAS Base

```sas
data <dataset-name>;
  set <dataset-name>;
  <new column int> = index(<column-name>,<string>);
  <new column int> = indexw(<column-name>,<string>,<del>);
run;
```

**Example:**

```sas
data xyz;
  set data1;
  found = index(custID, "123");
  found = indexw(custID, "123", "%");
run;
```

### KNIME Analytics Platform

**String Manipulation - indexOf($col$, “String”)**

![String Manipulation in KNIME](image)

---

**Description:**

Gives the first position of `nSearch` in the string or -1 if `nSearch` is not found in the string. The returned value is an integer between 0 (the first character) and length(str) - 1 (the last character).

**Examples:**

- `indexOf("abcde", "a") = 0`
- `indexOf("abcde", "c") = 2`
- `indexOf("abcde", "d") = 1`
... 

```sas
put();
source = <value>;
ff = <format>;
<output> = put(source, format);
...
```

**Example:**

```sas
...
source = "12.01.2011";
ff= date8.;
source_date = put(source, date8.);
...
```

For more specific type conversions you can use one of the following nodes:

- Number To String
- String To Number
- Double To Int
- String To Date&Time
- Time To String

**Note.** String Manipulation node functions toInt(), toDouble(), string() perform the same conversion resp. from String to Integer, from String to Double, from Number to String.
### PROC FORMAT

**SAS Base**

```sas
proc format library=<library-name>;
  value <column-name> [range1]=text1
  [range2]=text2
  [range3]=text3;
  value $<column-name>
  <valuesList1>=text4
  <valuesList2>=text5;
  ...
run;
```

**Example:**

```sas
proc format library=my;
  value education 1-11 = 'till High School'
  12 = 'High School'
  12<-high = 'above High School';
  value $sex 'M','m' = 'Male'
  'F','f' = 'Female';
run;
```

### Cell Replacer, String Replace (Dictionary), Rule Engine

**KNIME Analytics Platform**

**In BIRT (via Report Designer Extension):**

In Tab „Property Editor“ (panel under the Report Layout editor):
- select Tab “Map” to display values or ranges as text strings.

**Cell Replacer**

**String Replace (Dictionary)**

**In workflow to replace String from Dictionary:**

**Rule Engine**

**In workflow to replace String according to rule map:**
Logical Operations
**if - then**

<table>
<thead>
<tr>
<th>SAS Base</th>
<th>Rule Engine</th>
</tr>
</thead>
</table>
| data <dataset-name>; 
  set <dataset-name>; 
  **if** <condition1> **then** <consequence1>; 
  **if** <condition2> **then** <consequence2>; 
  **else** <default-consequence>; |
| ![Rule Engine](image) |
| **Example:** |
| data xyz; 
  set xyz; 
  **if** Price < 50 **then** label='discounted'; 
  **else if** Price > 200 **then** label = 'overpriced'; 
  **else** label='normal priced'; |
| ![Rule Editor](image) |
| **Note.** Column values can be inserted in the rule editor by double-click in the “Column List” panel. |
### in() in SAS Base

```sas
data <dataset-name>;
  set <dataset-name>;
  if/where <col-name> in(<list-of-values>)
    then <consequence>;
run;
```

**Example:**

```sas
data xyz;
  set xyz;
  if class in (‘A’,’B’,’C’)
    then prediction = ‘Y’;
run;
```
to impute missing values:

Example:

data xyz;
  set data1;
  if missing(ContractID) then label=CustID;
  else label=ContractID;
run;
data <dataset-name> do <condition> ... end; run;

Example:

data iris; do species=1 to 3; ... end; run;

In the „Flow Control“ - „Loop Support“ category, KNIME offers a number of different loop nodes to start and to end a loop. All loops in KNIME start with a “<name> Loop Start” node and end with a “<name> Loop End” node. All nodes between the loop start node and the loop end node make the loop body.

- **Cycle “for” with a fixed number of iterations.** The “Counting Loop Start” and the “Interval Loop Start” node, in combination with a “Loop End” node, implement a “for” cycle.
- **Cycle “while”.** The “Generic Loop Start”, in combination with a “Variable Condition Loop (End)”, implements a “while” cycle.
- **Looping on a list of values.** The “TableRow To Variable Loop Start” node, together with a “Loop End” node, loops across all rows of a table and transforms each row into a flow variable. This is particularly useful, for example, to loop on a list of unique values.
- **Looping on groups/chunks of data rows.** The “Group/Chunk Loop Start” loops across groups/chunks of the input rows.
- **Loop with memory.** Notice that all previous loops act at each iteration on the original input data set. The only loop using the processed dataset from the previous iteration is the Recursive loop (“Recursive Loop Start” node – “Recursive Loop End” node)
- **Note.** Loop End nodes benefit from a number of extra options in their context menu to start, stop, resume, and execute step by step the loop cycle.
### SAS Base

%macro <macro-name>(par1=value1, par2=value2);
... Code
... %mend <macro-name>;

### KNIME Analytics Platform

KNIME nodes can be grouped together inside a meta-node like code lines inside a macro. Just select the meta-nodes, right-click and select “Collapse into Meta Node” or “Encapsulate into Wrapped Metanode”.

Alternatively you can create a meta-node from scratch by clicking the “Add Meta Node Wizard” button in the top bar and following the meta-node wizard instructions.

Input parameters can be introduced in a meta-node by means of “Quick Form” nodes.

**Note.** The same meta-node can be stored centrally and called by different workflows, workspaces, and even users. This is a metanode template. The metanode instance calling to the metanode template is a linked metanode. This is available in KNIME Analytics Platform (the free open-source version) as well as in KNIME Server.
Date&Time Manipulation
### Extract Date&Time Fields

#### SAS Base

```sas
DATA <dataset-name>;
  INPUT  <var-name> <format> <pos> <datetime-name> <date-format>;
  M = MONTH(<datetime-name>);
  D = DAY(<datetime-name>);
  Y = YEAR(<datetime-name>);
  wk_d = WEEKDAY(<datetime-name>);
  q = QTR(<datetime-name>);
...
RUN;
```

#### KNIME Analytics Platform

![Extract Date&Time Fields](image)

**Example:**

```sas
DATA dates;
  INPUT  name $ 1-4 @6 bday mmddyy11.;
  M = MONTH(bday);
  D = DAY(bday);
  Y = YEAR(bday);
  wk_d = WEEKDAY(bday);
  q = QTR(bday);
...
RUN;
```

**Note.** The “Extract Date&Time Fields” node works on Date&Time data type. Since a date is always read as a String, it has to be converted to the Date&Time type before the “Extract Date&Time Fields” node is applied. For the conversion, use a “String to Date&Time” node.
### SAS Base

```sas
data <dataset-name>;
  <column-name> = today();
run;
```

### Example:

```sas
data _null_
  current_date = today();
  if (current_date - datedue) > 30 then do;
    put 'As of ' current_date date9. 'the payment of invoice #'
      Invoice_nr 'is expired.';
  end;
run;
```

### KNIME Analytics Platform

Generate 1 Date&Time value with “Use execution date&time” as Starting Point.

![Create Date&Time Range](image)
### mdy()

<table>
<thead>
<tr>
<th><strong>SAS Base</strong></th>
<th><strong>KNIME Analytics Platform</strong></th>
</tr>
</thead>
</table>

```
data <dataset-name>;
  set <database-name>;
  <date> = mdy(<month>,<day>,<year>);
run;
```

**Example:**

```
data xyz;
  set data1;
  birthday = mdy(11, 13, 1965);
run;
```

**String Manipulation – join() and String to Date&Time**

Starting from 3 columns “day”, “month”, and “year”, combine them with the “String Manipulation” node using the join function.

Use a “String To Date&Time” node to convert the combined String to a Date&Time type.

**Note.** The Date&Time format in the menu of the “String To Date&Time” node is editable. If you do not find the format that you need, you can always edit one.

**Note.** For a conversion to Date&Time you need day and month in a two-char format, like “03” for March. If you start from day and month as Integer, the “Number To String” node does not supply a fixed format with zero-padding for the conversion. So, Integer “3” becomes String “3” while you need “03” for a conversion to Date&Time.
Math Operations
<table>
<thead>
<tr>
<th>int()/ceil()/round()/floor()</th>
<th>Math Formula – round(), ceil(), floor()</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SAS Base</strong></td>
<td><strong>KNIME Analytics Platform</strong></td>
</tr>
<tr>
<td>data &lt;dataset-name&gt;;</td>
<td><img src="image" alt="Math Formula Dialog" /></td>
</tr>
<tr>
<td>set &lt;dataset-name&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;matrix1&gt; = int(&lt;matrix&gt;);</td>
<td></td>
</tr>
<tr>
<td>&lt;matrix2&gt; = round(&lt;matrix&gt;);</td>
<td></td>
</tr>
<tr>
<td>&lt;matrix3&gt; = floor(&lt;matrix&gt;);</td>
<td></td>
</tr>
<tr>
<td>&lt;matrix4&gt; = ceil(&lt;matrix&gt;);</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

data xyz;
set data1;

    Price = int(Price);
    Price = round(Price);
    Price = floor(Price);
    Price = ceil(Price);
run;
### SAS Base

```sas
data <dataset-name>;
set <database-name>;
<variable-name> = max/min(<matrix_1>, ..., <matrix_n>);
run;
```

### Example:

```sas
data xyz;
set data1;
y = max(nr, Price);
run;
```

### KNIME Analytics Platform

Math Formula - max_in_args() min_in_args()

- **max_in_args()**
  - **Description:** Maximum in argument list `max_in_args(x, y, z)`, if all of them is NaN, the result is also NaN

- **min_in_args()**
  - **Description:** Minimum in argument list `min_in_args(x, y, z)`, if all of them is NaN, the result is also NaN

![Math Formula Dialog](image-url)
### SAS Base

```sas
data <dataset-name>;
  set <database-name>;
  <variable-name> = max/min(<matrix_1>, ..., <matrix_n>);
run;
```

**Example:**

```sas
data xyz;
  set data1;
  y = max(x1, x2, x3);
run;
```

### KNIME Analytics Platform

![Image of KNIME Analytics Platform](image)

**Math Formula (Variable)**

- **Function:** `max`, `min`.
- **Description:**
  - `max`: Maximum of a list of arguments.
  - `min`: Minimum of a list of arguments.
- **Example:**
  ```java
  max_arg1, arg2, arg3
  ```

**Dialog - 677: Math Formula (Variable)**

- **Input:**
  - Variable set:
    - `max`: Arguments: `arg1, arg2, arg3`.
  - Additional arguments can be added.
- **Options:**
  - **Append Variable:** `new_variable`.
  - **Replace Variable:** `variable2`.
  - **Convert to List:**
```
mean()/sum()

### SAS Base

```sas
data <dataset-name>;
set <database-name>;
<new column> = mean/sum(col_1, col_2, ..., col_n);
<new column> = mean/sum(of col1-coln);
<new column> = mean/sum(of col_1, col_2, ...
run;
```

### KNIME Analytics Platform

```
Expression
average({column1$,$column2}) + {column1$,$column2}
```

### Example:

```sas
data xyz;
set xyz;
avg = mean(x, y, z);
avg = mean(of x1-x10);
avg = mean(of x, y, z);
run;
```
### Transpose

<table>
<thead>
<tr>
<th><strong>SAS Base</strong></th>
<th><strong>KNIME Analytics Platform</strong></th>
</tr>
</thead>
</table>
| **proc transpose**  
  data=<dataset-name> out=<dataset-name>;  
  [by <grouping-column>;]  
  run; | **Note.** The “Transpose” node performs a simple transposition (that is, without the “BY” statement) of the input data table. It has no configuration settings, besides a few memory options. |

**Example:**

**proc transpose** data=indata out=outdata;  
  by location date;  
  run;
Basic Statistics
### PROC FREQ

#### SAS Base

```sas
proc freq data = <dataset-name>;
   tables <col_1>*<col_2> / <option>;
run;
```

**Example:**

```sas
proc freq data = xyz;
   tables class*value / chisq;
   TITLE 'Simple Example of PROC FREQ';
run;
```

Default freq calculations: N, %, cumulative N, cumulative %

The **chisq** option to the tables statement performs the standard Pearson chi-square test on the table(s) requested.

The **expected** option prints the expected number of observations in each cell under the null hypothesis.

The **exact** option requests Fisher's exact test for the table(s). This is automatically computed for 2 x 2 tables.

### Crosstab and Statistics

#### KNIME Analytics Platform

The default freq calculations and the statistics of the proc freq can be found in the “Crosstab” node.

The “Crosstab” node analyzes the relation of two columns with categorical data by displaying the frequency distribution (N, %, deviation, cell chisquare, etc...) of the categorical variables in one of the output tables.

The “Crosstab” node also provides chi-square test statistics and, in case of a cross tabulation of dimension 2x2, the Fisher's exact test. The results of these tests are displayed in the second output table.

#### Statistics

The node “Statistics” also calculates a number of statistics variables for the input data table columns.
Reporting
**PROC REPORT**  

**SAS Base**

```sas
proc report data=<dataset-name>;
column <column-names>;
define <column-name> / <stat> <format>;
<options>;
run;
```

**Example:**

```sas
proc report data=xyz nowindows;
column x y z;
define x / group format=$xfmt.;
define y / analysis sum format=dollar9.2;
define z / group format=$zfmt.;
...
run;
```

**Data to Report and Report Designer (BIRT)**

**KNIME Analytics Platform**

- In the workflow connect a “Data to Report” node to the data to be displayed in the report.
- Click the “Reporting” button in the tool bar on the top.
- The KNIME Reporting Designer (which is BIRT) opens.
- Create your report.

![Data to Report Workflow](image)
Include other script languages
### PROC SQL

#### SAS Base

```sql
proc sql;
cREATE TABLE <table-name> AS
SELECT <column-name>
FROM <dataset-name>
WHERE <condition>
ORDER BY <grouping-column>;
QUIT;
```

**Example:**

```sql
proc sql;
cREATE TABLE Filter AS
SELECT Price
FROM xyz
WHERE nr > 13
ORDER BY CustID;
QUIT;
```

### Database Query and Java, R, and Python Snippet

#### KNIME Analytics Platform

KNIME offers a SQL editor in the Database Query node and a few Snippet nodes to write and execute pieces of Java, R, Python, and Perl code.
From a personal experience, I know how difficult it can be to switch from one software tool to another. Even though both tools provide the same functionalities, a change of mind set is needed to discover where and how such functionalities are implemented in the new software tool.

This book is a quick guide on how to use KNIME for users coming from the SAS experience. It is not an introduction to KNIME, since it is assumed that the user is already familiar with the basic concepts of data manipulation, analysis, and reporting in KNIME. It is more a map of the most commonly used SAS functions and techniques into their KNIME equivalents.

For a complete introduction to KNIME, please refer to my book “KNIME Beginner’s Luck” available from KNIME Press under https://www.knime.com/knimepress

About the Author

Dr Rosaria Silipo has been mining data since her master degree in 1992. She kept mining data throughout all her doctoral program, her postdoctoral program, and most of her following job positions. She has many years of experience in data analysis, reporting, business intelligence, training, and writing. In the last few years she has been using KNIME for her consulting work, till she started working for KNIME as a Principal Data Scientist in 2013.

ISBN: 978-3-9523926-1-4