PhUSE 2008

Paper AD02

SAS® Data via .NET®: "The Power to *Sh*ow"

Nicolas Rouillé, Merck Serono International S.A., Geneva, Switzerland

ABSTRACT

Presentation of an application whose purpose is to visualize SAS Data Set (displaying exact numerical values, coded and decoded values, variables attributes, formats description, selection of variables, freeze panes functionality, …), then to explore data (quantitative and qualitative descriptive analysis of variables -or any block of selected cells, search functionality, SQL filter, auto-generation of SAS Code, SAS Code submission, …).

This copyrighted software has been developed for Windows platform using the .NET framework (language C#), a Microsoft technology.

The development of this User-Interface driven Application was motivated by:

- a need to increase productivity for SAS Programmers,
- a need to facilitate the knowledge of data by Statisticians, being SAS users or not: they can directly access and explore SAS data sets with no knowledge of SAS syntax.
- ... and, subjectively, by the difficulty to find on the market a satisfying SAS data set viewer!

INTRODUCTION

We are working in a complex environment, doing complex things… But some simple things need a reminder:
A basic question concerning the Data Set we are looking to: *"Is what we see really what we get (when deriving variables or when receiving external data)?"* In other words, does the displayed data represent the stored value?
Furthermore, how much time do we spend on doing repetitive "simple things" like sorting data, filtering data, frequency or quantitative analysis: how many "Proc Print", "Proc Sort", "Proc Freq", "Where/If" clauses at the end of each day? To save time, some of these elementary steps are sometimes simply not realized.

Many SAS Tools to Visualize/Explore data already exist:

- SAS Viewer V8.2 / V9 (problem with filter, decoded variables, reordered variables, scrolling, etc…[5])
- SAS Explorer Window
- SAS/AF applications (cost for development/deployment, flexibility limitations)
- SAS FSP (not really user-friendly!)

So, What are our Pretensions? Mainly 3:

- A) Visualize data,
- B) Visualize data,
- C) Visualize data.

The first part of this paper identifies the requirements. Then, the technical development is approached.

VISUALIZE DATA IS A RIGOROUS INITIATIVE

VISUALIZE… THE EXACT STORED VALUE

Here is a basic data step syntax, submitted within a SAS session:

```
*--- Data Sample: expected numerical value is 0.3;
Data InputData;
x=0.3; output;
x=100.3-100; output;
x=3*0.1; output;
x=3*1/10; output;
Run;
```

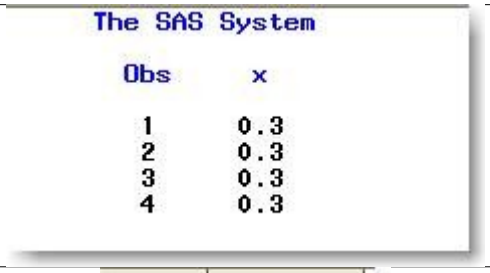
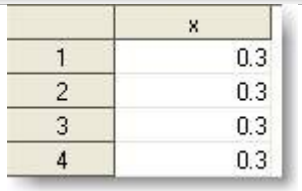
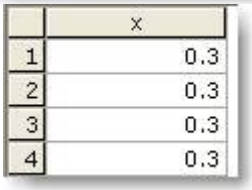
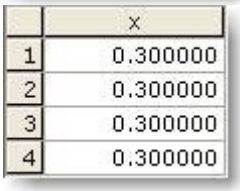
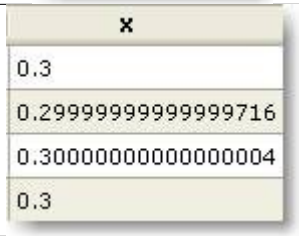
SAS Output *(Proc Print)*	
SAS Data Explorer	
SAS Viewer *(formatted values)*	
SAS Viewer *(unformatted values)*	
The developed "Data Set Viewer" application	

Figure 1.1 : 5 different layouts of common data visualization tools representing how the 4 numerical records of "x" are displayed.

As expected, all the SAS tools used to visualize data are displaying values of 0.3, excepted for the developed "Data Set Viewer" application...

Hummm… am I Wrong?

This following syntax submitted within a SAS session will definitely status on which tool is leading to a wrong display of the stored data, or not:

```
*--- Test some (un)equalities;
Data Test_It;
Set InputData;
Format x 22.20;
if x=0.3 then Flag1=1; else Flag1=0;
if x=0.29999999999999716 then Flag2=1; else Flag2=0;
if x=0.30000000000000004 then Flag3=1; else Flag3=0;
Run;
```

Using the same tools as previously, let's see

- How the 4 numerical records are displayed with a 20 decimals format.
- How will be resolve the equality test of "*Is equal to 0.3?*" (See "*Flag1*" variable)
- If the displayed data is equal or not to the stored value (See "*Flag2*" and "*Flag3*" variables)

SAS Output *(Proc Print)*

The SAS System

Obs	x	Flag1	Flag2	Flag3
1	0.30000000000000000000	1	0	0
2	0.29999999999999000000	0	1	0
3	0.30000000000000000000	0	0	1
4	0.30000000000000000000	1	0	0

SAS Data Explorer

	x	Flag1	Flag2	Flag3
1	0.30000000000000000000	1	0	0
2	0.29999999999999000000	0	1	0
3	0.30000000000000000000	0	0	1
4	0.30000000000000000000	1	0	0

SAS Viewer *(formatted values)*

	x	Flag1	Flag2	Flag3
1	0.3	1	0	0
2	0.2999999999	0	1	0
3	0.3	0	0	1
4	0.3	1	0	0

SAS Viewer *(unformatted values)*

	x	Flag1	Flag2	Flag3
1	0.300000	1.000000	0.000000	0.000000
2	0.300000	0.000000	1.000000	0.000000
3	0.300000	0.000000	0.000000	1.000000
4	0.300000	1.000000	0.000000	0.000000

The developed "Data Set Viewer" application

x	Flag1	Flag2	Flag3
0.3	1	0	0
0.29999999999999716	0	1	0
0.30000000000000004	0	0	1
0.3	1	0	0

Figure 1.2 : 5 different layouts of common data visualization tools testing a set of equalities.

Referring to the SAS tech. note TS-654 [1], we learn that *"the decimal values 0.1 and 0.3 do not have exact binary representations. In decimal arithmetic 3*0.1 is exactly equal to 0.3, but this equality does not hold in binary arithmetic. Although there are various means to store numbers, SAS uses floating point, or real binary, representation to store all numeric values."*

SAS Institute represents data like they were supposed to be instead of what they really are.

This makes sense as it is the expected result, but what happen if you receive an external Data Set containing a decimal variable? Will you be able to directly distinguish the 4 records presented above?

- If we have to derive from the variable "x" in our sample, a primary endpoint equal to "*Yes*" when lower or equal to 0.3 and to "*No*" when greater than 0.3, this will probably lead to misunderstood (in the sample why the record number 3 is not derived as "*No*"?). Even if, in a better case, you are using a rounding function, you have to measure the impact of this step (see Chapter 16, "Propagation of Rounding Error," in Elements of Numerical Analysis by Peter Henrici [6]).

Best, and simplest solution is to be able to visualize the exact stored value of the data.

VISUALIZE… THE EXACT RESULT OF ANY REQUEST/FILTER

No application can be guarantied "bug-free", the goal in this section is to present some problems that have been considered as critical, because of the absence of error/warning message, even if a result is returned to the user.

Some SAS Viewer bugs related to the filter function are well known [5], for instance:

- Conflict between the variable name "missing" and the value "missing".
- SAS comparison operator "!=" can lead to trouble, furthermore mnemonic operators can not be used. It is necessary to use a documented well-known language for request syntax: the easiest is maybe to allow the use of SAS SQL syntax in our data visualizer.

A strategy that can be used, to ensure the result of your request is exactly corresponding to what you would have in a SAS Session, is to submit the text of the request in batch mode, soliciting SAS/Base. You will also have the opportunity to visualize the Log of your submission and to control the accuracy of the submitted request.

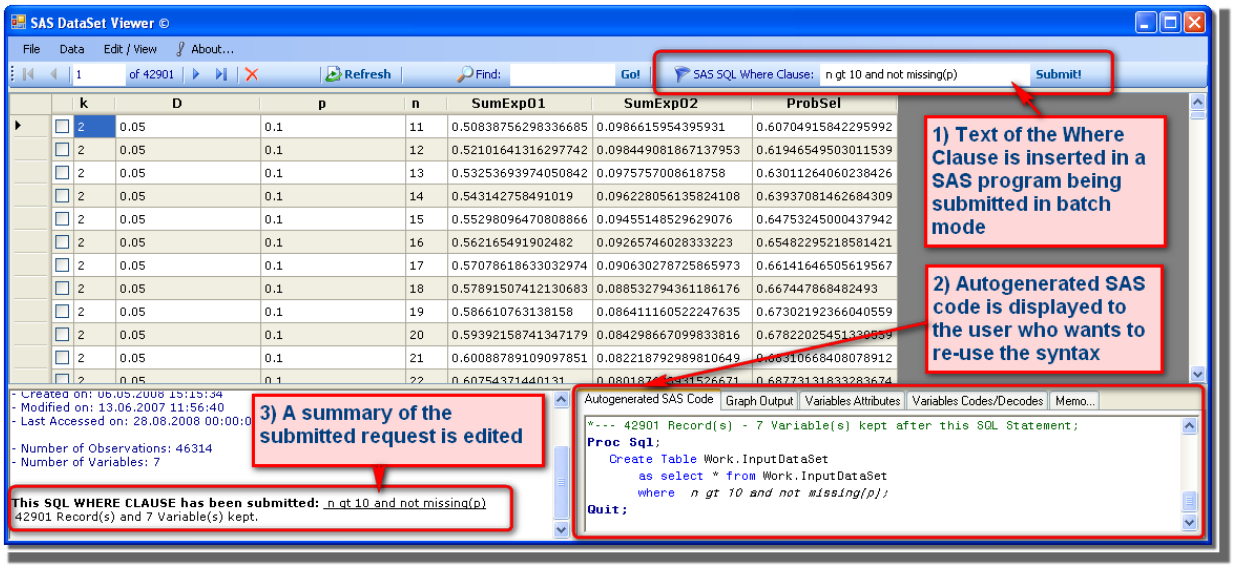


Figure 2 : Basic subset of data in the "Data Set Viewer" application.

VISUALIZE… IN THE SAME TIME CODED/DECODED VARIABLES

Why choose between a layout with formatted values or a layout with unformatted values?

Print coded and decoded variables is useful in particular :

- with Start/End dates if you want to easily estimate duration,
- to avoid any confusion on dichotomous variables coded using 0 and 1, or 1 and 2 ("*Yes*"/"*No*" variables : Is the code used for "*No*" equal to 0 or equal to 2?).

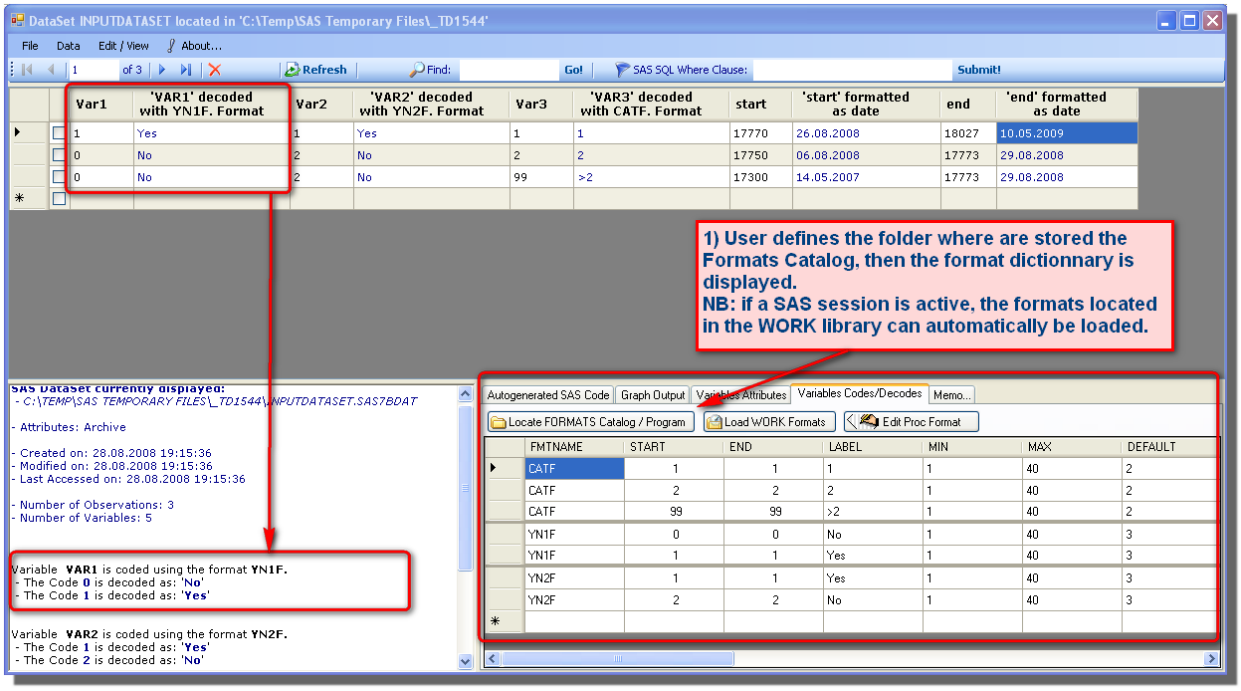


Figure 3 : Reliability between coded and non-coded variables

SOME OTHER "GOOD-TO-HAVE" FEATURES

Additionally to the previous "must-have" features of a data visualizer, here is a list of some -subjectives- useful functions a Data Set Visualizer could have:

- Comfort to navigate trough Data Set using a freeze pane functionality, alternate lines color, variables selection or hide some variables, move or sort variables.
- Possibility to select a Data Set stored in the “Work” library of a current SAS session.
- Track any action made by user (event Log window).
- Auto-generated SAS Code to store actions realized by user (e.g. after a query, an SQL statement is generated, this allow user to retrieve this selection in a SAS Program after a copy-and-paste)
- Short keys for descriptive analysis that allow quick checks on data (consistency)
- Generate SAS Syntax like Proc Report Statements, or code to export the displayed Data Set into Excel controlling labels and formats, …
- Display a graphical description of data
- Save an auto-generated PDF report of your session (containing the generated SAS code and outputs)
- Possibility to submit SAS code (useful for derivation, consistency checks on a Data Set)
- Export the displayed Data Set in CSV, TAB or XML format.

All this in one-click...

...How to do it?

A DEVELOPMENT USING .NET® TECHNOLOGY

WHAT IS .NET®?

The Microsoft .NET Framework is a software technology that is available with several Microsoft Windows operating systems. It includes a large library of pre-coded solutions to common programming problems, a runtime or virtual machine that manages the execution of programs written specifically for the framework, and a set of tools for configuring and building applications. The .NET Framework is a key Microsoft offering and is intended to be used by most new applications created for the Windows platform.

This free development platform can use these languages: C#, Visual Basic, Visual C++, Jscript, J#.

An other point of interest: the .NET framework is well documented.

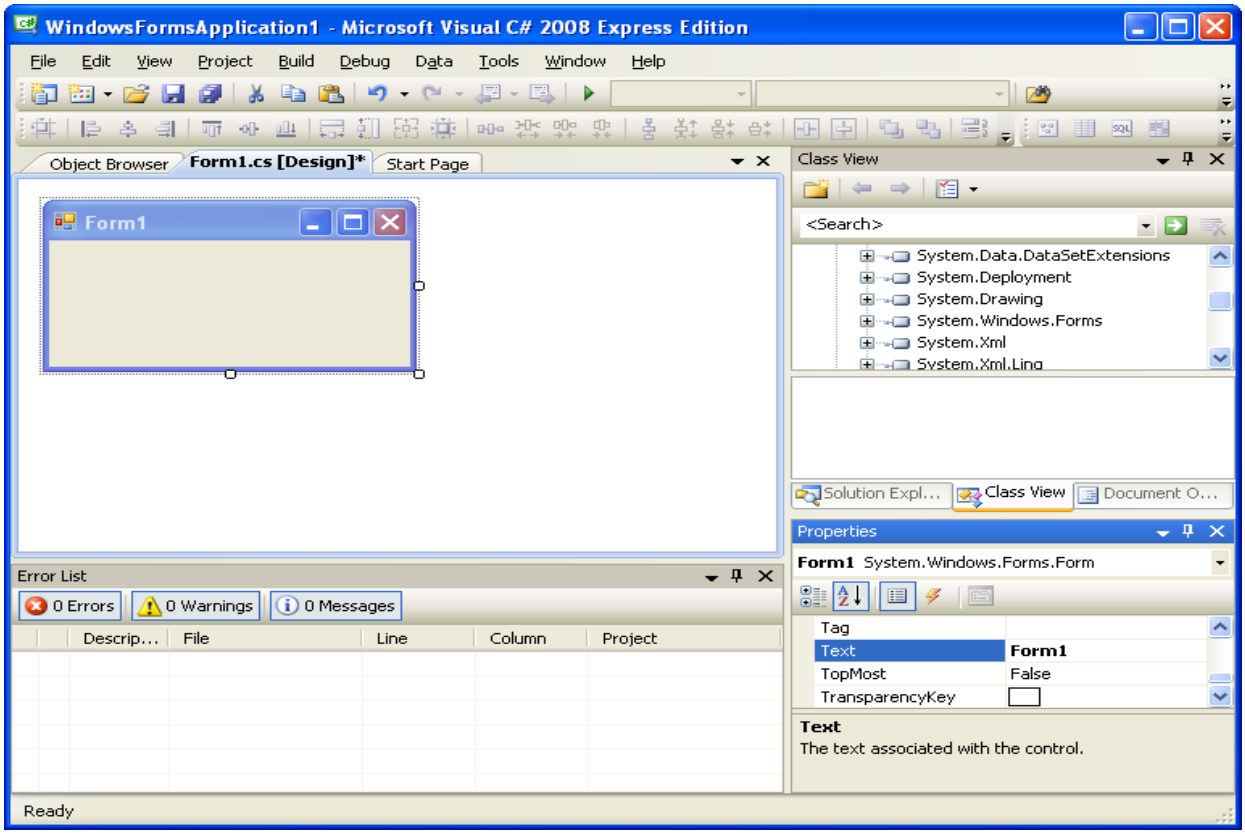


Figure 4 : Start page to design a .NET application in Microsoft Visual C# 2008 Express Edition.

TWO STRATEGIES TO INTEGRATE SAS WITHIN THE .NET ENVIRONMENT

To visualize SAS Data Sets, run SAS programs, view SAS Log or SAS Lst files, you can mainly consider two options:

- To specifically visualize a SAS Data Set within an .NET object, use an OLE DB adapter. You can use the batch mode to execute your SAS programs (specifying in your command line the destination of Log and Lst files).
- To use SAS/Integration Technologies and its SAS Integrated Object Model (IOM). At first, a SAS Workspace is created (COM/DOM/IOM Bridge); then it allows you to take benefit of one of these services: DataService, FileService, Utilities, LanguageServive. We can particularly note some useful features offered by this set of services, very flexible and powerfull:
  - DataService allows you, between others, to manage librefs(assign/deassign/iterate)
  - and within the Utilities Service, you will find a FormatService that can applied SAS Formats to returned data.

Your choice can be based on cost (SAS/Integration Technologies must be licensed with V8 and higher if you choose the SAS IOM option), performance, maintenance criteria ... or your level of SAS addiction : this implies your dependence of future SAS versions, and that your data visualizer tool is dedicated to SAS Objects/Services.

IMPORT SAS DATA SET IN THE FRAMEWORK ENVIRONMENT

- At first, it is better to automatically copy locally the selected Data Set you want to display: this allows you (or anyone else!) to continue to work on this data in your (or any) SAS session with no error message concerning the Data Set accessibility. You can update the displayed data set with a “refresh” button.

```
// To copy the data set to be vizualized under a local "TEMP" Folder
File.Copy(SelectedFile_Public, "C:\\Temp\\_" + currentFile + currentExt, true);
// To set the copied file as Hidden
File.SetAttributes("C:\\Temp\\_" + currentFile + currentExt, FileAttributes.Hidden);
```

Figure 5.1 : The .NET class “System.IO.File” provides a static method to copy an existing file to a new file (overwriting a file of the same name is allowed)

- Then Import SAS Data Set into the .NET object: DataGridView

```
// Reprensent an in-memory cache of data
System.Data.DataSet sasDs = new System.Data.DataSet();

// Represent and open connection to the data source (the local copy of the SAS data file is opened)
OleDbConnection sas = new OleDbConnection(@"Provider=sas.LocalProvider; Data Source='C:\Temp\';");
sas.Open();

// Represent a stored procedure (or a SQL comamnd!) to execute against a data source
OleDbCommand sasCommand = sas.CreateCommand();
sasCommand.CommandType = CommandType.TableDirect;
sasCommand.CommandText = "_" + FILESTRING;

// Define the OLE DB adapter
OleDbDataAdapter da = new OleDbDataAdapter(sasCommand);

// Add or refresh rows in "sasDs" to macth those in the data source
da.Fill(sasDs, "SasData");

// Close the connection to the data source
sas.Close();
DataGridView1.DataSource = sasDs.Tables["SasData"];
```

Figure 5.2: Sample of a C# syntax used to import a SAS Data Set using an OLE DB adapter. SAS Integration Technologies offers an other very powerful way to integrate SAS objects.

– Manipulate your DataGridView Object. The .NET Framework Class Library is documented on the MSDN Website [3]: sample code are provided for a wide variety of programming language environments such as C#, Visual C++, Jscript, J#.

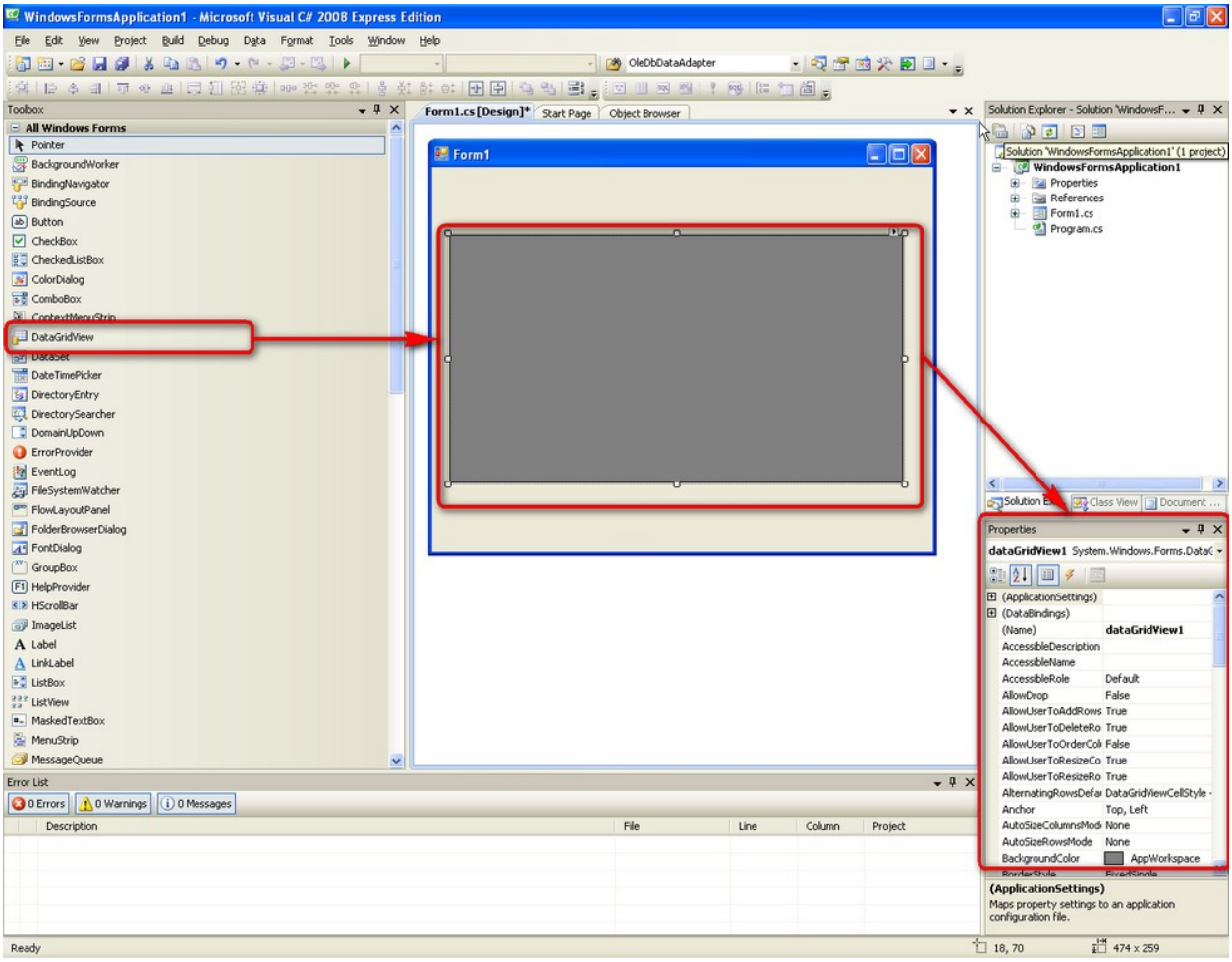


Figure 6 : The DataGridView Object that has been dragged-and-dropped from the Windows Forms library can then be customized using the properties listed in the "Properties" frame.

– Create your personalized application following your wishes… a set of Windows Forms is available (Common controls, container, Menus and Toolbars, Data, Dialogs, ...); you can also enrich your application with the different contributions of the .NET programmers community, very active.

Dedicated in a first time to SAS Data Sets, this tool can integrate other useful data formats. The presented application allows to visualize and explore S+, Excel, TAB, CSV data files (additionally to the functionality of export into CSV, TAB, XML format).
Visualization, exploration, descriptive analysis and export of data are independent from the native data format. For the user, there is no apparent conversion of the data file, what could be the format of the file.

CONCLUSION

The developed application is a sample of what can be done using the .NET Framework technology. Defining the need is crucial: technically, you can create a flexible and reliable application that ensures a high level of quality in the data visualization, respects data integrity and finally allows a better knowledge of your data.

This application can be used by developers, statistical programmers, statisticians, medical writers: ... any one who manipulates SAS Data Sets, in the pharmaceutical industry or not.

Why focus on the development of a Data Set visualization application? Maybe because the "*power to know*" starts with "*the power to show*"¨!

REFERENCE

[1] Technical Note TS-654 on numerical precision: “http://support.sas.com/techsup/technote/ts654.html”
[2] .NET Homepage: “http://www.microsoft.com/NET/”
[3] .NET Framework Class Library “http://msdn.microsoft.com/en-us/library/ms229335.aspx”
[4] SAS Integration Technologies: http://support.sas.com/documentation/onlinedoc/inttech/index.html
[5] SAS-L Forum concerning Error using SAS Viewer or SAS Explorer Window
[6] "Propagation of Rounding Error," in Elements of Numerical Analysis by Peter Henrici

ACKNOWLEDGMENTS

For his review and encouragement, Dominique MOYSE, M.D.

RECOMMENDED READING

- Technical Note TS-654 on numerical precision: “http://support.sas.com/techsup/technote/ts654.html”
- Technical Note TS-230 on representation issues: “http://support.sas.com/techsup/technote/ts230.html”
  - “*Floating point representation has been chosen as the best alternative, but has the known disadvantage of representation error. This brings up the need to determine when representation error will be a problem in your applications. Since integer numbers up to a certain magnitude can be represented exactly, you will not encounter the problem when working with these numbers. This could include operations like control counters, frequency counters, and other math operations which involve whole numbers”*
  - “*The only SAS format that displays the exact value of a variable is the HEX16. Format*”

- Henrici, P., Elements of Numerical Analysis, New York: John Wiley & Sons, Inc.
- Knuth, D.E., The Art of Computer Programming, Volume 2, Seminumerical
- Algorithms, Reading MA: Addison-Wesley. Langston, R.D., "Numeric Precision Considerations in SAS Software"

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

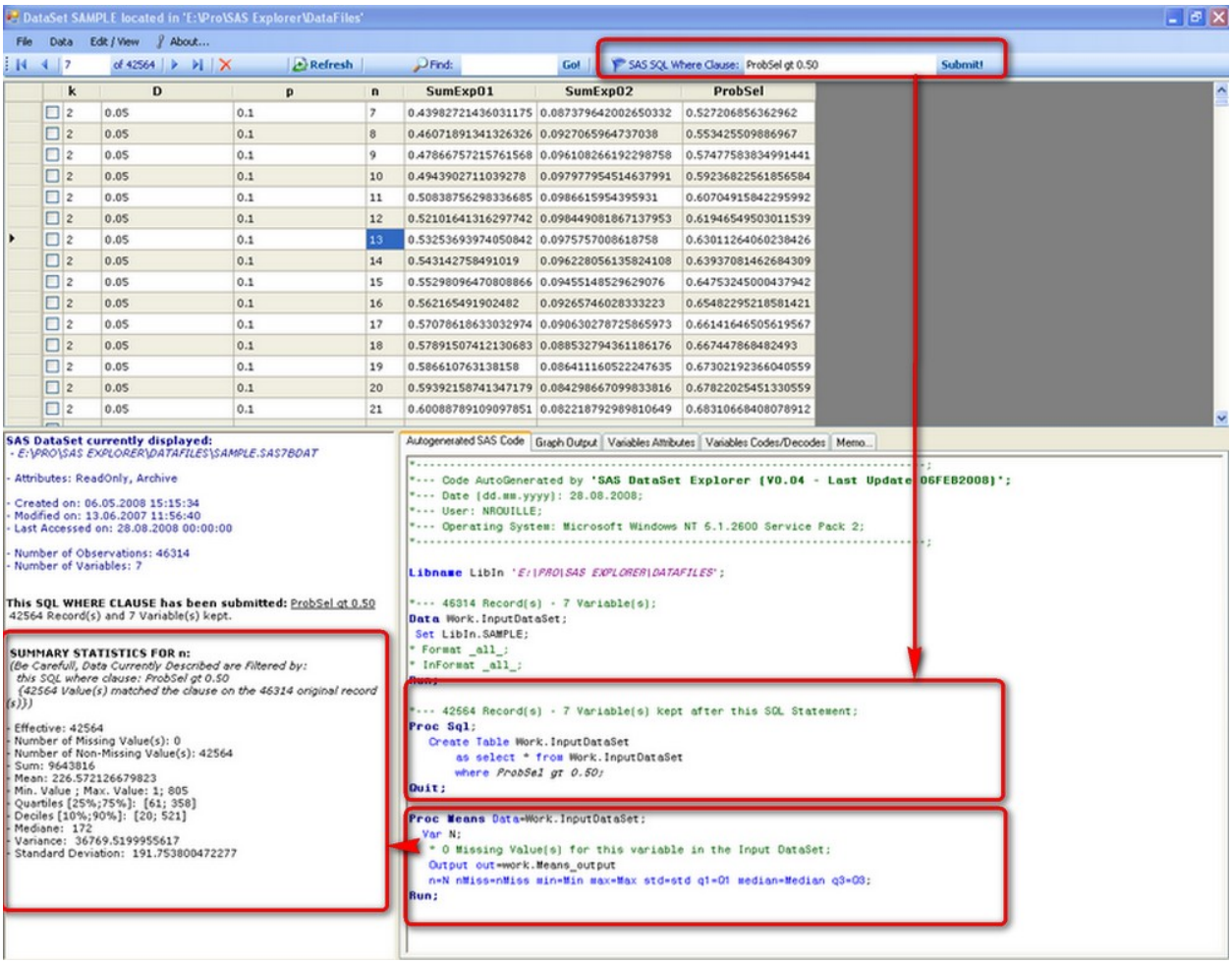
Nicolas Rouillé
Merck Serono International S.A.
9 Chemin des Mines
1202 Geneva, Switzerland
Phone: +41 22 414 31 86
Fax: +41 22 414 33 30
Email : nicolas.rouille@merckserono.NET
Web:http://www.merckserono.NET/

APPENDIX

A DAY IN THE LIFE OF A SAS DATA SET USER...

Here are some tasks of what some of us are daily faced to:

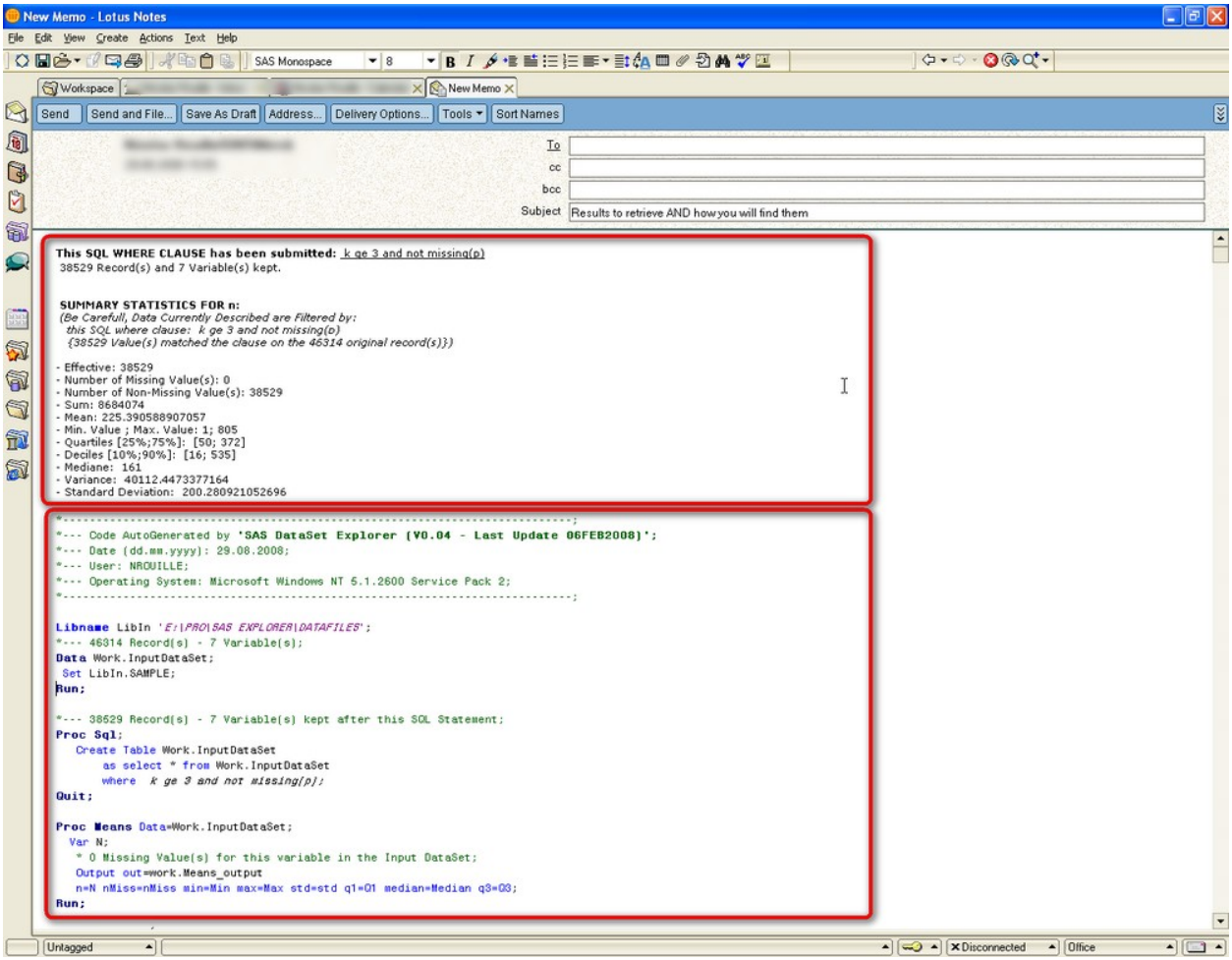
1. Retrieve how was calculated a value reported in an analysis table : e.g. what was the condition used to calculate the published median?



In this sample, 2 functions of the application are successively used:

- data filter: the user writes the SQL syntax, then submit the request (top of the window). The resulting subset of data is displayed, then the equivalent SAS code (“Proc Sql” with a where clause) is automatically generated.
- via the menu (Data>Summarize Variable) or by pressing a short key combination, the user requires a quantitative description of the variable “n”: the equivalent SAS syntax (“Proc means”) is automatically generated to allow further analysis. The summary statistics are displayed in the same time on the bottom left window which tracks every action of the user.

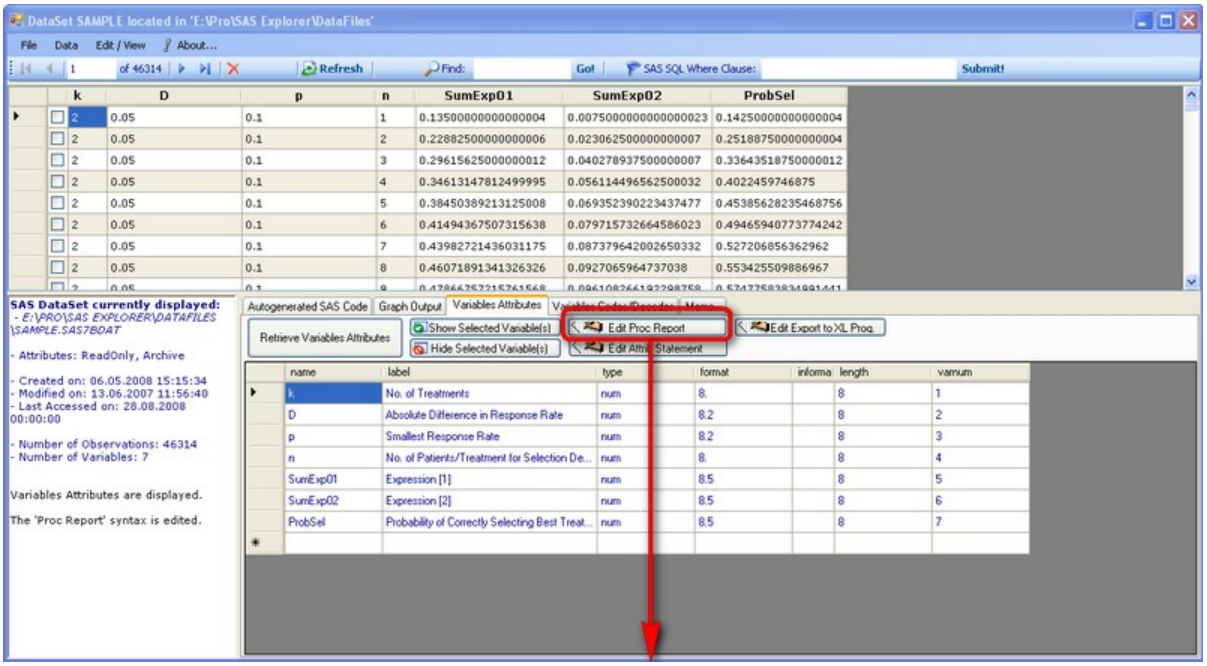
2. A Statistician e-mails specification to a SAS Programmer concerning derivation of a variable

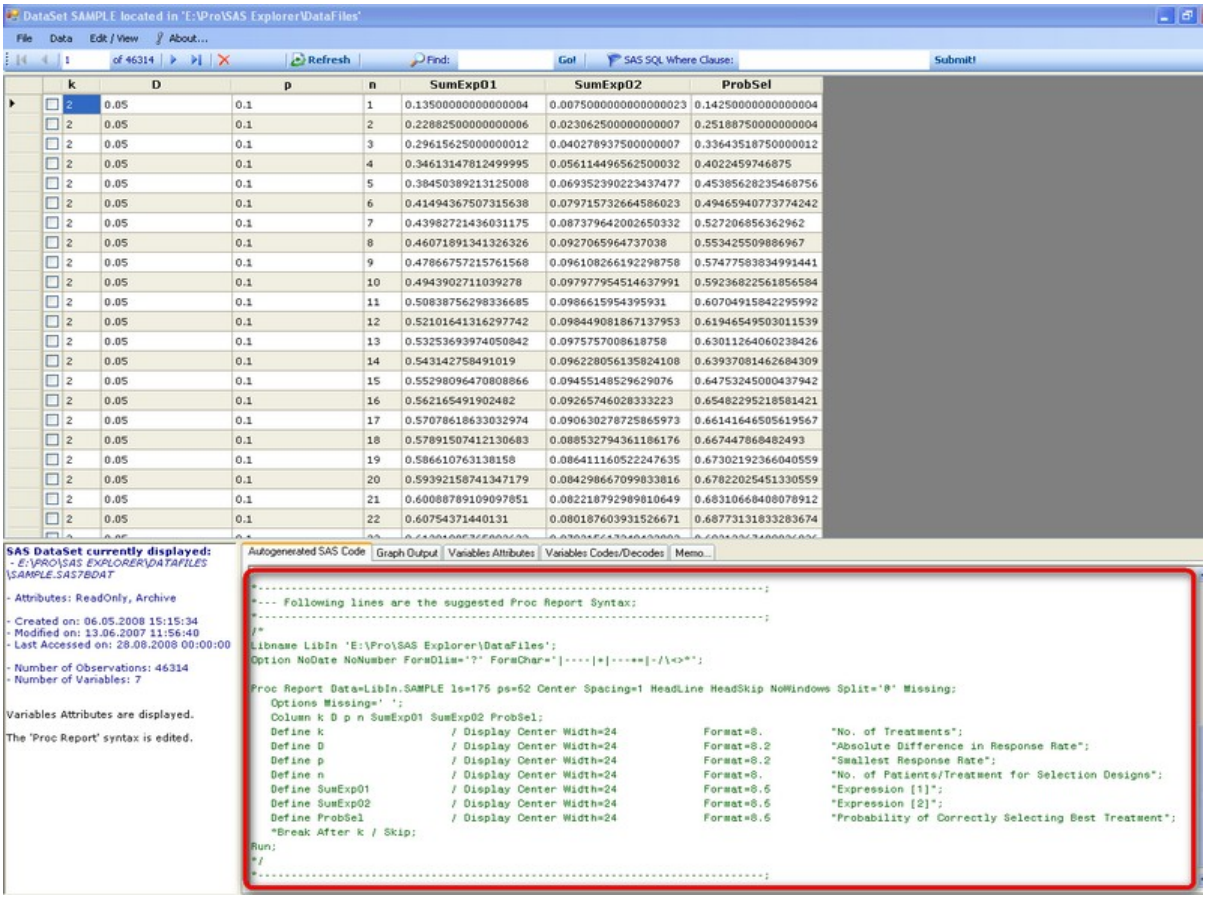


In this mail, sections highlighted with red rectangles are directly copied-and-pasted from the “Data Set Viewer” application, successively:

- In the first rectangle, the text published in the Log Event Window of the application:
  - name, location and attributes of the data set displayed
  - summary of the successive filters applied with a count of excluded records and selected records
  - result of the descriptive analysis.
- In the second rectangle, the auto-generated SAS syntax that allows the user who copy-and-paste it into a SAS session to reproduce all the actions and results found in the “Data Set Viewer” application.

3. A SAS programmer wants to generate the code to edit a “Proc report” (raw/derived Data Set listings)



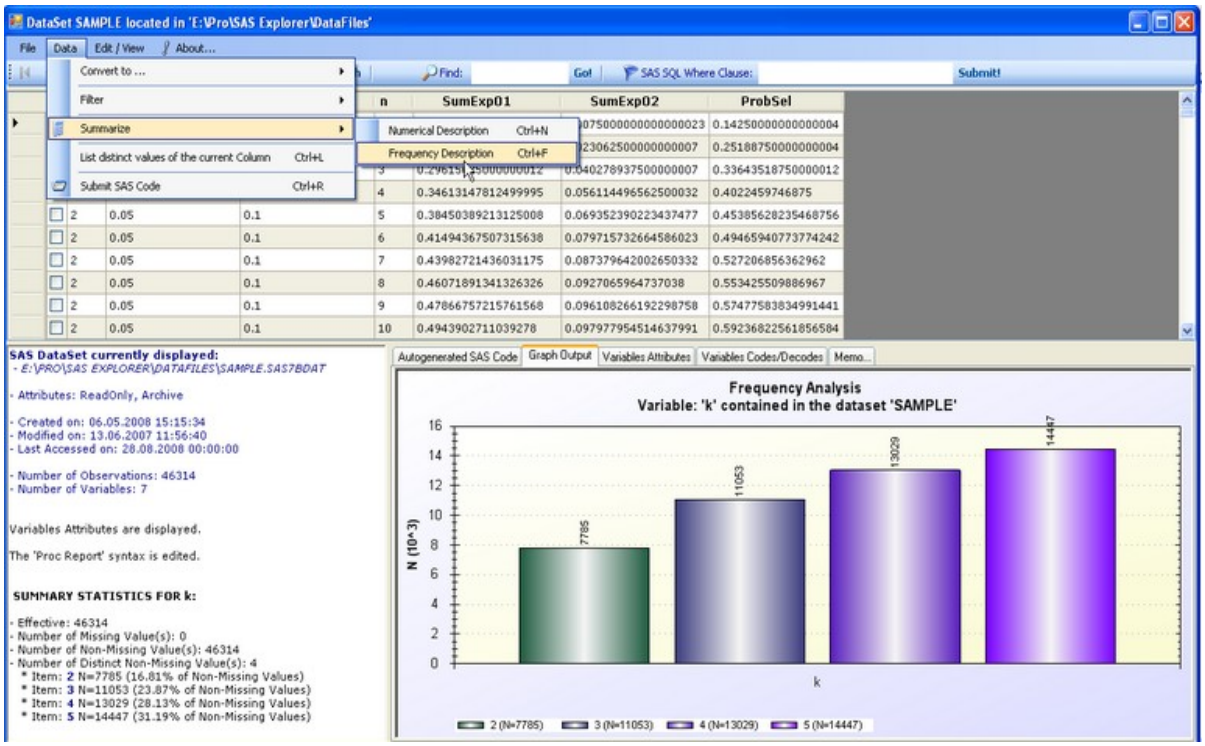


N.B. : with the same method, it is possible to generate the SAS program to use for an export of the Data Set into an Excel format.

4. A SAS user wants to realize univariate analysis

The published outputs are standardized, in order to minimize the number of necessary actions from the user (for instance, the frequency analysis of a selected variable is produced with the short key “CTRL+F”).

4.1. Frequency Analysis



4.2 Quantitative Analysis