Spécial Foire aux questions

Ce numéro spécial de la foire aux questions de l'OASUS a été créé pour les rencontres SAS de Montréal (MONSUG) et Québec du printemps 2015. Il s'agit d'un regroupement de quelques unes des meilleures questions répondues lors des rencontres de l'OASUS à Ottawa.

Question 1 : Comment garder plusieurs copies d’un dataset?

Often, one needs to maintains multiple versions of a given dataset for historical/archival as a well as backup/recovery purposes (there may be other reasons not as obvious). There are different ways this could be done in SAS. For example, one could program a custom solution in a SAS application that would enable versioning one or many datasets. This custom solution would certainly add some complexity to the application.

Another approach could be to use the SAS audit trail facility which logs all the modifications to a SAS data set during its lifecycle. It does not maintain multiple versions of a data set however.

SAS has yet another facility called the *Generation Data Sets*. A generation data set is an archived data set that is part of a generation data group. Under a generation data group, each time a data set is replaced by a new one under the same name, a copy of the old version is kept under a specific generation number. The most recent version is called the *base version*. A generation data group is created when the dataset option *GENMAX* is specified with a number greater than 0 (0 is the default and means that the generation data sets feature is not in effect). This option specifies the maximum number of versions to keep. The following example creates a generation data group with a maximum number of versions of 4.

```
/* Dataset creation and first generation */
data testgen(genmax=4);
   input var1 $ var2 $;
   datalines;
a11 b11
a12 b12
;
```

Except for the current version, each historical version of a data set in a generation data group is stored using an absolute version number appended to its member name. The absolute version starts with # and is followed with 3 digits. For example, for data set A:

A	base (current) version
A#003	most recent historical version
A#002	second most recent historical version
A#001	oldest historical version.

When the maximum number of generations is reached, SAS will delete the oldest version in order to keep only the number copies as specified by the *GENMAX* option.

Once you have a generation group created, how do you reference the various data sets associated with it? First rule, except for the current version, never use the physical name with the #. This will lead to a syntax error since data set names cannot contain a #! Instead use the *GENNUM* data set option as followed:

GENNUM=0 : to access the current version. This is the default, so you don't need to specify it.
GENNUM=-x : to access the version x generations back. This is relative reference.
GENNUM=x : to access version x. This is an absolute reference.

The following example illustrates the concept.

```
/* Print most recent version */
title 'most recent version';
proc print data=testgen;
run;

/* Print generation back from the current version */
title 'Version -2';
proc print data=testgen(gennum=-2);
run;

/* Print generation #003 */
title 'Version #003';
proc print data=testgen(gennum=3);
run;

/* This file reference is invalid */
/* A valid SAS file name cannot contain an # */
title 'Version #003';
proc print data=testgen#003;
run;
```

To manage generation groups, it is recommended to use *PROC DATASETS*. *PROC DATASETS* allows you to:

- Change the number of versions to be kept.
- Delete specific versions and an entire generation group.
- Rename versions

SAS generation data sets are a very handy and easy-to-use facility to keep multiple versions of a dataset. Keep in mind however that the storage requirements of your SAS application may increase quickly.

Reference :
SAS Global Forum Paper 253-31, Exploring SAS Generation Data Sets, Kirk Paul Lafler

Question 2 : Les variables FIRST et LAST d’une étape DATA sont très pratiques. Quel est l’équivalent avec PROC SQL?

The FIRST.variable and the LAST.variable relates to BY-group processing. These variables identify the beginning and the end of a BY group, enabling special processing for specific observations.

SQL is a set language and does not have advanced procedural constructs as is the case for the DATA step. However, there are techniques to achieve similar results. In general, they tend to be more complex that a simple BY group processing DATA step construct and can possibly be much slower with large amount of data.

Here is an example using the SASHELP.CARS dataset. From that dataset, we want to find the lowest 3 retail prices (variable: MSRP) by origin of the car (variable: Origin).

First, let’s look how you would do it with a DATA step. It is fairly simply for anyone comfortable with the BASE SAS language. The code is intuitive and efficient.

```
/* Retrieve the lowest 3 MSRP by Origin using a DATA step. */

proc sort data=sashelp.cars out=CarsSorted;
      by Origin MSRP;
run;

data CarsResults(keep=Origin MSRP);
      retain group_count;
      set CarsSorted;
      by origin MSRP;

      if first.origin then
            group_count = 0;

      group_count = group_count + 1;

      if group_count =< 3 then
            output;
run;
```

To achieve the same task in SQL requires some advanced notions of SQL. The code is not as intuitive as for the DATA step and it does not perform very well with a large amount of data.

```
/* Retrieve the lowest 3 MSRP by Origin using a SQL. */

proc sql;

create table CarsResults2 as
      select  Origin, MSRP
            from sashelp.cars as tdo
            where (
            (select count(*) from sashelp.cars as tdi
                  where
                        tdi.origin=tdo.origin
                        and tdi.MSRP < tdo.MSRP )  <=2 )
      order by Origin, MSRP;

quit;
```

There are other SQL techniques that can be used but exploring them is beyond the scope of this exercise. The important point to be made is that SAS has a number of tools and one should always select the tool that would perform best for the job at hand. The resulting solution should also be relatively easy to comprehend for anyone who would need to support it in the future.

PROC SQL is a tremendous tool but it does not replace the DATA step. As soon as one needs to perform procedural processing, the DATA step should be considered first.

Question 3 : Est-ce possible d’appeler une étape DATA à partir d’une autre étape DATA ou même une PROC à partir d’une étape DATA ?

Yes it is possible !

The simplest way is to use the functions DOSUB or DOSUBL available in SAS 9.4 (experimental in SAS 9.3).

DOSUB and DOSUBL are BASE SAS functions and submit any code that is passed to them. Contrary to CALL EXECUTE, the code is executed immediately even within a DATA step that is already running. The code can consist of one or many steps including DATA steps and PROC steps! The difference between DOSUB and DOSUBL is that DOSUB will execute code contained in a file whereas DOSUBL will execute code passed as a string.

DOSUB and DOSUBL return a value of zero if SAS code was able to execute, and return a nonzero value if SAS code was not able to execute.

Call a DATA inside a DATA step

```
data _null_;
      rc = dosubl('data x; y=1; run; %let abc=1;');
      abc=symget('abc');
      put abc=;
run;

data _null_;
      set x;
      put _all_;
run;
```

Call a PROC inside a DATA step

```
data _null_;
  rc = dosubl('proc sql;
                create table work.regions as
                      select distinct region
                      from sashelp.shoes;');
  putlog rc=;
run;
```

Use of DOSUBL mimics a PROC FCMP function

```
/* This macro returns a list of variables contained in the */
/* in the specified dataset. */
%MACRO ExpandVarList(data=_LAST_, var=_ALL_);
      %if %upcase(%superq(data)) = _LAST_ %then
           %let data = &SYSLAST;
      %let rc = %sysfunc(dosubl(%str(
           proc transpose data=&DATA(obs=0)
           out=ExpandVarList_temp;
           var &VAR;
           run;

           proc sql noprint;
           select _name_ into :temp_varnames separated by ' '
           from ExpandVarList_temp
           ;

           drop table ExpandVarList_temp;
           quit;
           )));
      &temp_varnames
%MEND ExpandVarList;

/* Use it as a macro "function" */
data newclass(keep=newvar1
%ExpandVarList(data=sashelp.class));
      newvar1=1;
      tempvar=0;
      set sashelp.class;
run;
```

References

Langston, Rick. 2013. "Submitting SAS® Code On The Side". Paper 032-2013. SAS Global Forum 2013, San Francisco, California. http://support.sas.com/resources/papers/proceedings13/032-2013.pdf

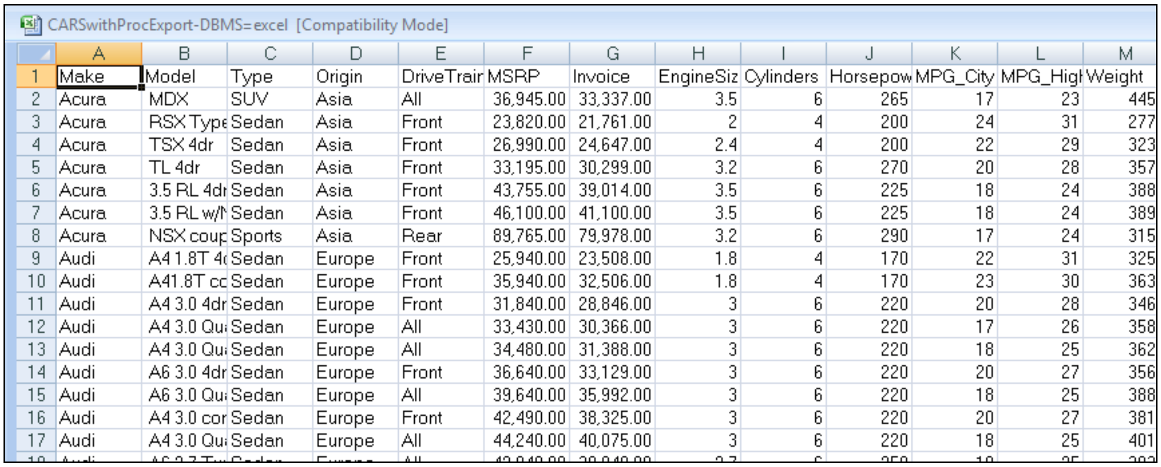
Question 4 : Avec PROC EXPORT, est-ce possible de créer un chiffrier Excel avec des colonnes dont la largeur s'ajuste automatiquement?

PROC EXPORT creates basic Excel spreadsheets. It is not possible to issue Excel commands such as autofit using it. To do so you require using DDE which is no longer recommended since its support by Microsoft is uncertain (and it does not work well on servers).

PROC EXPORT with DBMS=EXCEL

If you use PROC EXPORT with DBMS=EXCEL, SAS will use the Microsoft ACE or Jet engine via the LIBNAME engine to do the conversion and will not resize the columns. The only solution in this case is to post-process the spreadsheet (either manually or with some Excel programming).

```
/* PROC EXPORT with DBMS=excel does not autofit */
proc export data=sashelp.cars
outfile='c:\temp\CARSwithProcExport-DBMS=excel.xlsx' dbms =
excel replace;
run;
```

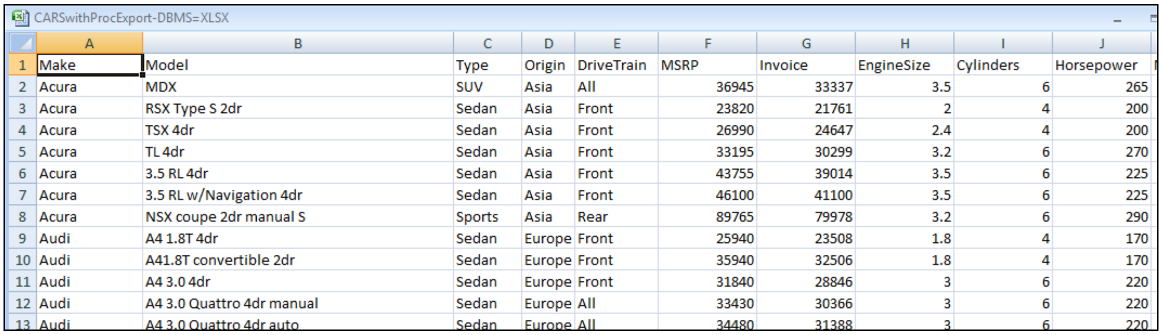


	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Make	Model	Type	Origin	DriveTrai	MSRP	Invoice	EngineSiz	Cylinders	Horsepow	MPG_City	MPG_Higl	Weight
2	Acura	MDX	SUV	Asia	All	36,945.00	33,337.00	3.5	6	265	17	23	445
3	Acura	RSX Type	Sedan	Asia	Front	23,820.00	21,761.00	2	4	200	24	31	277
4	Acura	TSX 4dr	Sedan	Asia	Front	26,990.00	24,647.00	2.4	4	200	22	29	323
5	Acura	TL 4dr	Sedan	Asia	Front	33,195.00	30,299.00	3.2	6	270	20	28	357
6	Acura	3.5 RL 4dı	Sedan	Asia	Front	43,755.00	39,014.00	3.5	6	225	18	24	388
7	Acura	3.5 RL w/N	Sedan	Asia	Front	46,100.00	41,100.00	3.5	6	225	18	24	389
8	Acura	NSX coup	Sports	Asia	Rear	89,765.00	79,978.00	3.2	6	290	17	24	315
9	Audi	A4 1.8T 4	Sedan	Europe	Front	25,940.00	23,508.00	1.8	4	170	22	31	325
10	Audi	A41.8T cc	Sedan	Europe	Front	35,940.00	32,506.00	1.8	4	170	23	30	363
11	Audi	A4 3.0 4dı	Sedan	Europe	Front	31,840.00	28,846.00	3	6	220	20	28	346
12	Audi	A4 3.0 Qu	Sedan	Europe	All	33,430.00	30,366.00	3	6	220	17	26	358
13	Audi	A4 3.0 Qu	Sedan	Europe	All	34,480.00	31,388.00	3	6	220	18	25	362
14	Audi	A6 3.0 4dı	Sedan	Europe	Front	36,640.00	33,129.00	3	6	220	20	27	356
15	Audi	A6 3.0 Qu	Sedan	Europe	All	39,640.00	35,992.00	3	6	220	18	25	388
16	Audi	A4 3.0 cor	Sedan	Europe	Front	42,490.00	38,325.00	3	6	220	20	27	381
17	Audi	A4 3.0 Qu	Sedan	Europe	All	44,240.00	40,075.00	3	6	220	18	25	401

PROC EXPORT with DBMS=XLSX

If you use PROC EXPORT with DBMS=XLSX, then PROC EXPORT will use its own translation engine which does a pretty good job at sizing the different columns properly. This may be an interesting approach if you don't need to "stick" with the older Excel format since it only supports the newer xlsx format.

```
/* PROC EXPORT with DBMS=xlsx does autofit */
proc export data=sashelp.cars
outfile='c:\temp\CARSwithProcExport-DBMS=XLSX.xlsx' dbms = xlsx
replace;
run;
```


CARSwithProcExport-DBMS=XLSX

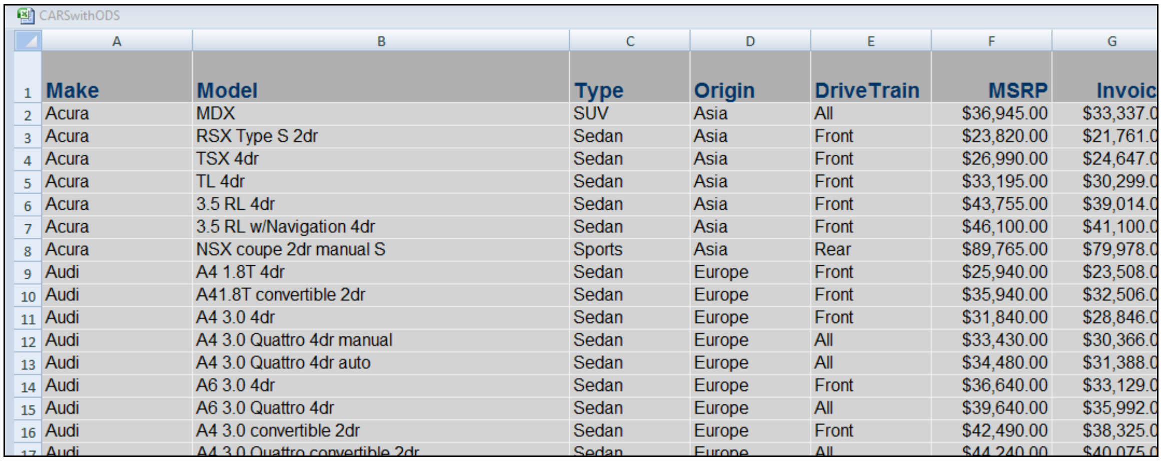
	A	B	C	D	E	F	G	H	I	J
1	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower
2	Acura	MDX	SUV	Asia	All	36945	33337	3.5	6	265
3	Acura	RSX Type S 2dr	Sedan	Asia	Front	23820	21761	2	4	200
4	Acura	TSX 4dr	Sedan	Asia	Front	26990	24647	2.4	4	200
5	Acura	TL 4dr	Sedan	Asia	Front	33195	30299	3.2	6	270
6	Acura	3.5 RL 4dr	Sedan	Asia	Front	43755	39014	3.5	6	225
7	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	46100	41100	3.5	6	225
8	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	89765	79978	3.2	6	290
9	Audi	A4 1.8T 4dr	Sedan	Europe	Front	25940	23508	1.8	4	170
10	Audi	A41.8T convertible 2dr	Sedan	Europe	Front	35940	32506	1.8	4	170
11	Audi	A4 3.0 4dr	Sedan	Europe	Front	31840	28846	3	6	220
12	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	33430	30366	3	6	220
13	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	34480	31388	3	6	220

ODS TAGSETS.EXCELXP

Another option is to switch to ODS TAGSETS.EXCELXP and specify column widths or a default column width to use when the file is opened in Excel. This works because ODS TAGSETS.EXCELXP creates a Spreadsheet Markup Language XML file which contains syntax for specifying column widths. This is different from PROC EXPORT, which exports SAS data to Excel binary format, but without any formatting instructions (such as font, color or column widths). TAGSETS.EXCELXP, on the other hand, has a way to specify DEFAULT_COLUMN_WIDTH, as well as ABSOLUTE_COLUMN_WIDTH -- which isn't exactly the same as AUTOFIT, but it does get you a bit closer to making columns wider when they need to be.

```
ods tagsets.excelxp file="c:\temp\CARSwithODS.xml"

options(absolute_column_width="10,25,8,8,8,8,8,8,8,8,8,8,8,8,8"
                                autofit_height="yes");
proc print data=sashelp.cars noobs;
run;
ods tagsets.excelxp close;
```



	A	B	C	D	E	F	G
1	**Make**	**Model**	**Type**	**Origin**	**DriveTrain**	**MSRP**	**Invoic**
2	Acura	MDX	SUV	Asia	All	$36,945.00	$33,337.0
3	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820.00	$21,761.0
4	Acura	TSX 4dr	Sedan	Asia	Front	$26,990.00	$24,647.0
5	Acura	TL 4dr	Sedan	Asia	Front	$33,195.00	$30,299.0
6	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755.00	$39,014.0
7	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	$46,100.00	$41,100.0
8	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765.00	$79,978.0
9	Audi	A4 1.8T 4dr	Sedan	Europe	Front	$25,940.00	$23,508.0
10	Audi	A41.8T convertible 2dr	Sedan	Europe	Front	$35,940.00	$32,506.0
11	Audi	A4 3.0 4dr	Sedan	Europe	Front	$31,840.00	$28,846.0
12	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	$33,430.00	$30,366.0
13	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	$34,480.00	$31,388.0
14	Audi	A6 3.0 4dr	Sedan	Europe	Front	$36,640.00	$33,129.0
15	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	$39,640.00	$35,992.0
16	Audi	A4 3.0 convertible 2dr	Sedan	Europe	Front	$42,490.00	$38,325.0
17	Audi	A4 3.0 Quattro convertible 2dr	Sedan	Europe	All	$44,240.00	$40,075.0

Question 5 : Est-ce possible de créer un fichier zip en SAS ?

Yes, it is possible!

With the ODS package destination.

```
ods package(ZipOut) open nopf;

/* Add files to the package. You can also include a path if you
desire */
ods package(ZipOut)  add file="&DataDir\air.txt" path='sashelp';
ods package(ZipOut)  add file="&DataDir\class.txt" path='sashelp';
ods package(ZipOut)  add file="&DataDir\cars.txt" path='sashelp';

/* Write out the package as a zip file */
ods package(ZipOut) publish archive
     properties(archive_name='ZipwithODS.zip'
                archive_path='c:\temp');

/* Close the package destination. Your zip file is now ready to
use! */
ods package(ZipOut) close;
```

With the FILENAME statement

Starting in SAS 9.4, it is also possible to write to (and to read from!) a zip file via the FILENAME statement.

```
filename infile "&DataDir";
filename zipfile ZIP 'c:\temp\ZipwithFilename.zip';

/* Write one text file to the zip file */

data _null_;
     infile infile(&File..txt) ;
     file zipfile(&File..txt);
     input;
     put _infile_;
 run;



%mend;
```

References :

Hamilton, Jack. 2013. "Creating ZIP Files with ODS". Proceedings of the SAS Global Forum 2013 Conference. Cary NC: SAS Institute Inc.

Langston, Rick. 2014. “Reading and Writing ZIP Files with SAS®”, Proceedings of the SAS Global Forum 2014 Conference. Cary NC: SAS Institute Inc.

Question 6 : Qu'est-ce PROC DOCUMENT? Pouvez –vous donner un exemple?

PROC DOCUMENT is part of the ODS DOCUMENT facility which allows you to:

- Save an ODS output into a ODS document which is a repository for ODS output, graphs and datasets;
- Replay the output to a specific format;
- Search and filter an ODS document;
- Add notes and change titles, footnotes, etc.

Before you can use PROC DOCUMENT, you have to create an ODS document by directing your output to an ODS document destination instead of a destination such as pdf or html that you would normally specify. The ODS document is saved in a special SAS item store file but you can use the normal <libref.>member-name nomenclature to designate the file. The following code provides an example:

```
ods document name=work.quickstart;
proc contents data=sashelp.cars ;
run;
proc means nmiss mean max min data=sashelp.cars;
class origin type;
run;
ods document close;
```

Now that you have an ODS document, you use PROC DOCUMENT to replay the output to another ODS destination:

```
/* replay the ODS document into an html output */
/*ods html file="c:\temp\output1.html";*/
proc document name=work.quickstart;
     replay;
     run;
quit;
/*ods html close;*/
```

You can keep adding to an existing ODS document by referring it as an ODS destination in other blocks of code:

```
/* Adding new content to a new ODS document */
ods document name=work.quickstart;
proc freq data=sashelp.cars;
     table origin  * type / list;
     run;
ods document close;
```

You can list the content of an existing ODS document:

```
/* Summary listing of the content of an ODS document*/
proc document name=work.quickstart;
     list;
     run;
quit;
/* Detailed listing of the content of an ODS document*/
proc document name=work.quickstart;
     list / levels=all details  ;
     run;
quit;
```

You can replay only certain ODS objects stored in the ODS document. In order to do that you need to understand how ODS names those objects and how they are organized. Typically, each run of a procedure has a corresponding folder in the ODS document which is named according to the name of the procedure and is assigned a sequence number (example Freq#1). Sub-folders can also exist depending on the output produced by the procedure and eventually individual ODS objects are stored in a specific folders. You can replay specific object or all the objects contained within a folder.

```
/* Replaying only the objects generated by Freq#1 */
ods html file="c:\temp\output2.html";
proc document name=work.quickstart;
     replay freq#1;
     run;
quit;
ods html close;
```

There are many other operations you can do with PROC DOCUMENT. The book listed in reference is highly recommended to learn more.

Reference:

Tuchman, Michael. 2012. **PROC DOCUMENT by Example Using SAS**®. Cary, NC: **SAS** Institute Inc.