PharmaSUG China 2019 - Paper CC-051

Exploration of SAS Output Solutions

Liao Wen, Dongsheng Guo, Zhengfang Du, and Peiqi Wang, Proswell Medical Company

ABSTRACT

This article explores three widely-used SAS output solutions under the SAS Output Delivery System (ODS), for users who need to generate native Excel output files. The topic methods include the PROC PRINT, the PROC EXPORT and the PROC REPORT procedure. To make the process more controllable and reliable, also for better accessibility, our exploration is based on real-world clinical data programming example. The contents of the example include coding, test running, experimental analysis, conclusion and recommendation.

INTRODUCTION

SAS provides powerful data processing procedures such as PROC SQL and DATA STEP for both large, concentrative data flow and small, scattering tables. However, any data analysis would inevitably need to produce a physical, touchable result, because your client may not know how to appreciate the beauty of your nested macro structure or the algorithm that you have figured out by sitting in front of the computer for hours. In most cases, people only need simple, straightforward result first, sometimes even a number, further explanations may depend on the quality of the first sight. In CRO area, one of the most frequently requested text outputs is the native excel table report. In this article, as the most recent destination in the third maintenance release for SAS 9.4 (TS1M3) will be treated as the main frame to support the comparison of the above-mentioned procedures.

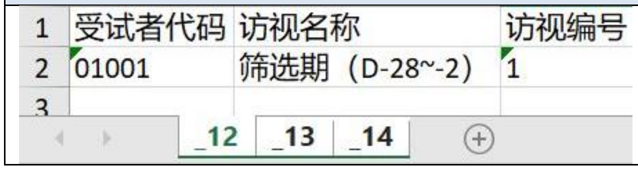
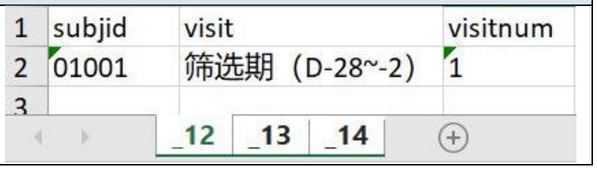
TOOLS USED

For demonstration purpose, we are using a specific module in one of our clinical data validation programs as an example (Liao Wen 2019, OD1.sas). This program is developed in Base SAS 9.4. The output of the program is a standard two-dimension table containing detailed clinical medical test information such as subject id, test center id, name of diseases, etc. For confidentiality reason, we would choose to keep only the subject id, visit date-range name and its corresponding id “visitnum” to de-identify some private information. Since this article would not refer much to the graphic design or any statistical analysis, it could be more efficient to pick one observation for easy-viewing.

Our expected output is one single excel workbook that contains three different sheets named 12, 13 and 14 related to three independent data procedures programmed in one single SAS file. In addition, the three variables in each worksheet should be in numeric, character and numeric form respectively. Moreover, the cell value of the second variable “visit”, is supposed to be a combination of number, character string and special characters rather than the number-only contents that the corresponding variable “visitnum” has. Any output result that does not match these rules will be marked as unwanted.

Without considering further decorations, the output table that we use as a benchmark should look precisely like **Table.1** bellow:

Table.1 Expected Output Data

With Labels	With Variable Names
	

1	受试者代码	访视名称	访视编号
2	01001	筛选期（D-28~-2）	1
3			

1	subjid	visit	visitnum
2	01001	筛选期（D-28~-2）	1
3			

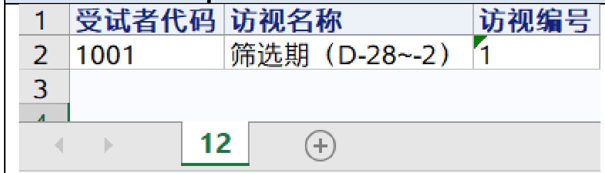
EXPERIMENTAL ANALYSIS

Table.2 to **Table.4** include the screenshots of the initial test output generated through the PROC PRINT, PROC EXPORT and the PROC REPORT procedures respectively. The first row of the three demonstration tables contains the major coding part of the output procedure. The column headers are displayed with both labels and variable names- i.e. in **Table.2**, the column headers in Chinese characters located on the left part of the second row sequentially conform to those written in English located on the right side.

Table.2 PROC PRINT procedure and output

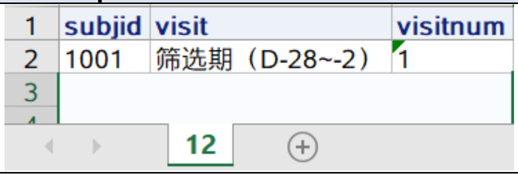
```
/*Output using PROC PRINT*/
 ods excel file="&results.\&xlsname..xlsx"
   options(sheet_name = "&sheetid");
 proc print data = listtb noobs;
 run;
```

Output table with labels



	A	B	C
1	受试者代码	访视名称	访视编号
2	1001	筛选期（D-28~-2）	1
3			

Output table with variable names

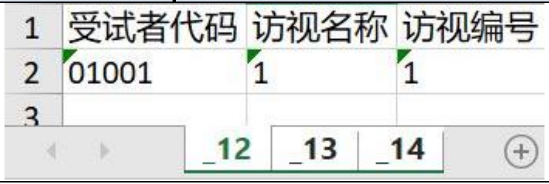


	A	B	C
1	subjid	visit	visitnum
2	1001	筛选期（D-28~-2）	1
3			

Table.3 PROC EXPORT procedure and output

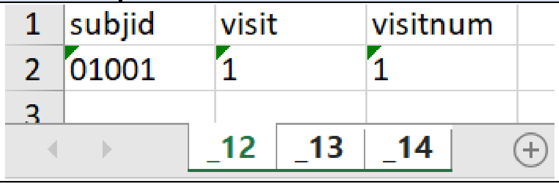
```
/*Output using PROC EXPORT*/
 proc export data = listview outfile="&results.\&xlsname..xlsx"
     dbms = excel label replace;
     sheet = "&sheetid";
 run;
```

Output table with labels



	A	B	C
1	受试者代码	访视名称	访视编号
2	01001	1	1
3			

Output table with variable names

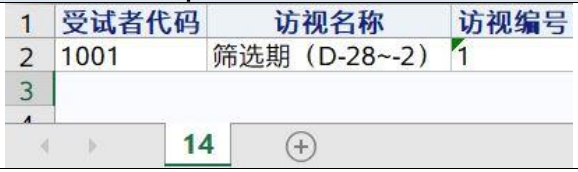


	A	B	C
1	subjid	visit	visitnum
2	01001	1	1
3			

Table.4 PROC REPORT procedure and output

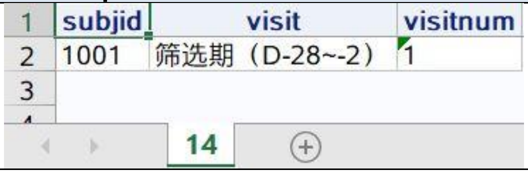
```
/*Output using PROC REPORT*/
 ods excel file="&results.\&xlsname..xlsx"
   options (sheet_name = "&sheetid ");
 proc report data = listtb nowd style(column header)={vjust=m};
 run;
```

Output table with labels



	A	B	C
1	受试者代码	访视名称	访视编号
2	1001	筛选期（D-28~-2）	1
3			

Output table with variable names



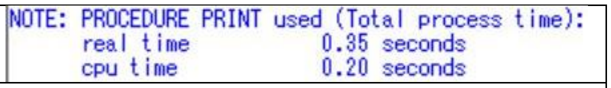
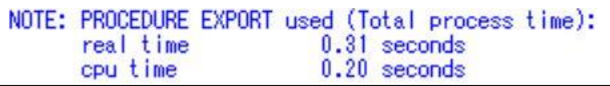
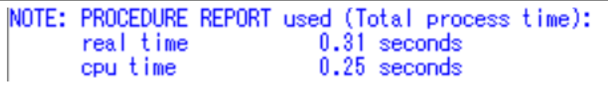
	A	B	C
1	subjid	visit	visitnum
2	1001	筛选期（D-28~-2）	1
3			

Above are the output tables produced by the three different procedures. By comparing the variable names, cell values and the sheet structures with the correct version in **Table.1**, we have found that both the PROC PRINT and the PROC REPORT procedures have produced only one worksheet, and the PROC EXPORT procedure has lost the correct format for the variable “visit” in its output table- Again, “visit” represents the actual visit date-range, which should be character type though its related id “visitnum” is in numeric form. Also, we have observed slight decorations such as the background color and font size in both the PROC

PRINT and the PROC EXPORT results while the PROC EXPORT shows only plain text in the output excel tables. Otherwise there is no significant difference found in the comparison.

Besides the accuracy and output effect, one of the most important factors that should be taken into account is the efficiency, which could be partly detected through the program execution time recorded in the SAS log. The first-round comparison is listed in **Table.5** as the following table:

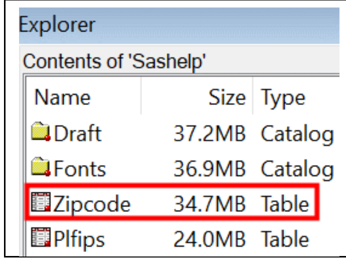
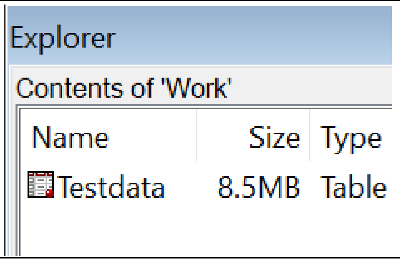
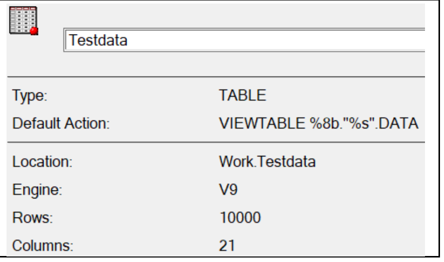
Table.5 Runtime comparison of the outputs

PROC PRINT	 NOTE: PROCEDURE PRINT used (Total process time): real time 0.35 seconds cpu time 0.20 seconds
PROC EXPORT	 NOTE: PROCEDURE EXPORT used (Total process time): real time 0.31 seconds cpu time 0.20 seconds
PROC REPORT	 NOTE: PROCEDURE REPORT used (Total process time): real time 0.31 seconds cpu time 0.25 seconds

Since our test output does not contain large amount of data, the processing time seemed not vary too much compared to each other. This is because the test data set is too small to present significant runtime difference. To collect more viewable results, we decided to import a larger data set, which is the “Zipcode” data set stored in the SAS built-in library SASHELP. By trimming it from 41,267 rows down to 10,000 rows while keeping the same amount of variable, we ended up to have a test data set of 8.5 megabytes without changing any variable names and labels (**Table.6**).

Table.6 Adopting larger data set for speed test

```
/*Trimming*/
 data testData; set sashelp.Zipcode (obs = 10000); run;
```

Data set from SASHELP	Trimmed data set	Property of the data set
 Explorer Contents of 'Sashelp' Name / Size / Type Draft 37.2MB Catalog Fonts 36.9MB Catalog Zipcode 34.7MB Table Plfips 24.0MB Table	 Explorer Contents of 'Work' Name / Size / Type Testdata 8.5MB Table	 Testdata Type: TABLE Default Action: VIEWTABLE %8b."%s".DATA Location: Work.Testdata Engine: V9 Rows: 10000 Columns: 21

The adjustment of the speed test phase did not require much rearrangement of the supporting program. As **Table.7** shows, only slight changes such as data set renaming, destination folder change were made to fit the output procedure.

Note that we are adding another two commonly used ODS destination formats during the entire speed test. Although the ODS Excel is considered the latest supported option, the ODS Tagset.ExcelXP and the ODS html are old members of the ODS family as early as when some of our young generation had not yet started playing SAS. The ODS html supports the output that needs to be displayed in .html format while the ODS Tagset.ExcelXP provides the XML output, which is formatted pretty much like the HTML file used in web development. The troublesome part of using the Tagset.ExcelXP to produce the desired excel worksheet is, you will keep getting the pop-up dialogue window to remind you that the file you create is not a true native excel file. Implicitly, it is an XML file (Christopher 2017, p. 2), because that is what Tagset.ExcelXP does. In many cases, we are still using these old destinations for specific professional supports such as complex data-descriptions or other data structuring work with strict limitations of content changing. However, the advancement of new technology increases the efficiency requirement, thus simple but flexible data manipulations are needed to make the reporting procedure as straightforward as possible, just like what ODS Excel does.

Table.7 Supporting program for the runtime comparison

```
/*-----------------------------PROC EXPORT-----------------------------*/
 proc export data = testData dbms=xlsx
     outfile="&currpath.\PROC EXPORT.xlsx" label;
     sheet="EXPORT";
     title "PROC EXPORT";
 run;
 ods _all_ close;
 dm odsresult 'clear' continue;

/*------------------------------ODS EXCEL------------------------------*/
 /*PROC REPORT*/
 ods excel file="&currpath.\PROC REPORT(ods excel).xlsx"
     options(sheet_name = "REPORT");
 proc report data = testData nowd style(column header)={vjust=m};
     title "PROC REPORT" justify = center;
 run;
 ods excel close;
 dm odsresult 'clear' continue;

 /*PROC PRINT*/
 ods excel file="&currpath.\PROC PRINT(ods excel).xlsx"
     options(sheet_name = "PRINT");
 proc print data = testData noobs label;
     title "PROC PRINT" justify = center;
 run;
 ods excel close;
 dm odsresult 'clear' continue;

/*------------------------ODS TARGETS EXCELXP------------------------*/
/*PROC REPORT*/
 ods tagsets.excelxp path="&currpath."
     file="PROC REPORT(ods tagsets.excelxp).xls" style=printer
     options(sheet_name = "REPORT");
 proc report data = testData nowd style(column header)={vjust=m};
     title "PROC REPORT" justify = center;
 run;
 ods tagsets.excelxp close;
 dm odsresult 'clear' continue;

 /*PROC PRINT*/
 ods tagsets.excelxp path="&currpath."
     file="PROC PRINT(ods tagsets.excelxp).xls" style=printer
     options(sheet_name = "PRINT");
 proc print data = testData noobs label;
     title "PROC PRINT" justify = center;
 run;
 ods tagsets.excelxp close;
 dm odsresult 'clear' continue;


/*------------------------------ODS HTML------------------------------*/
/*PROC REPORT*/
 ods html file="&currpath.\PROC REPORT(ods html).xls" style=default
     options(sheet_name = "REPORT");
 proc report data = testData nowd style(column header)={vjust=m};
```

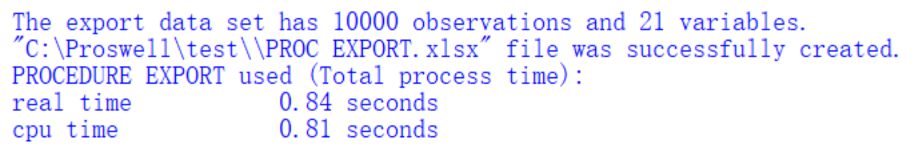
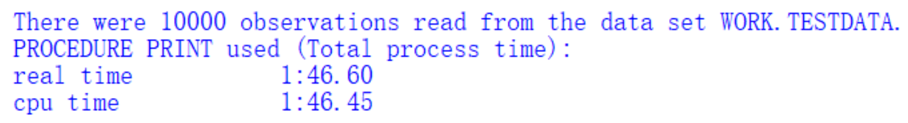
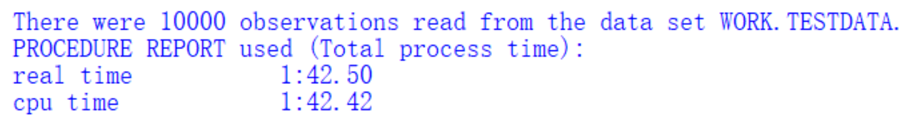
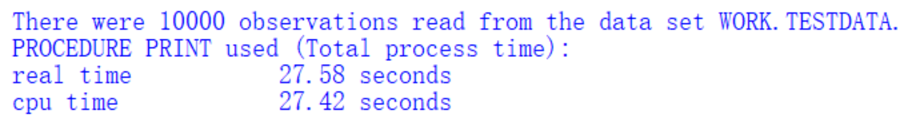
```
    title "PROC REPORT" justify = center;
run;
ods html close;
dm odsresult 'clear' continue;

/*PROC PRINT*/
ods html file="&currpath.\PROC PRINT(ods html).xls" style=default
    options(sheet_name = "PRINT");
proc print data = testData noobs label;
    title "PROC PRINT" justify = center;
run;
ods html close;
dm odsresult 'clear' continue;
```

This time SAS gives a more human-perceivable result as shown in **Table.8** and **Table.9** on page 5. The comparison through the two rounds also indicates that under the same hardware and software situation, the runtime difference keeps incrementing as the data size increases.

Note that the results of the other two choices- the ODS Tagset ExcelXP and the ODS html options were also collected and presented together with the ODS Excel destination. The reason that we take the PROC EXPORT as an independent option on top of the row is because this procedure supports nearly none of the ODS' sheet formatting for the target file. In another word, it does not belong to any specific ODS destination. We have already showed how it failed in **Table.3**, and we will bring it back to life in **Table.10**.

The runtime comparison clearly shows us that the PROC EXPORT reigns the speed record of 0.8 seconds while the PROC PRINT and the PROC REPORT procedures are way slower among all three different ODS processing. As we mentioned above, the PROC EXPORT produce plain text tables with no graphs, colors or any other effects. The ODS Tagset ExcelXP and the ODS html works slower but much faster than the ODS Excel. This is because as the latest production, the ODS Excel option remains certain balance between the sheet-formatting and the size-expanding for the destination file while the ODS Tagset ExcelXP and the ODS html are sacrificing large amount of disk space as **Table.9** shows. I tried to open the smallest output worksheet but decided to give up- with the file size of 6,921 kilobytes, it took me 30 seconds.

Table.8 Screenshots of speed comparison

PROC EXPORT	 The export data set has 10000 observations and 21 variables. "C:\Proswell\test\\PROC EXPORT.xlsx" file was successfully created. PROCEDURE EXPORT used (Total process time): real time 0.84 seconds cpu time 0.81 seconds
ODS Excel	
PROC PRINT	 There were 10000 observations read from the data set WORK.TESTDATA. PROCEDURE PRINT used (Total process time): real time 1:46.60 cpu time 1:46.45
PROC REPORT	 There were 10000 observations read from the data set WORK.TESTDATA. PROCEDURE REPORT used (Total process time): real time 1:42.50 cpu time 1:42.42
ODS Tagset ExcelXP	
PROC PRINT	 There were 10000 observations read from the data set WORK.TESTDATA. PROCEDURE PRINT used (Total process time): real time 27.58 seconds cpu time 27.42 seconds

PROC REPORT	There were 10000 observations read from the data set WORK.TESTDATA. PROCEDURE REPORT used (Total process time): real time 34.65 seconds cpu time 34.61 seconds
ODS html	
PROC PRINT	There were 10000 observations read from the data set WORK.TESTDATA. PROCEDURE PRINT used (Total process time): real time 8.18 seconds cpu time 8.15 seconds
PROC REPORT	There were 10000 observations read from the data set WORK.TESTDATA. PROCEDURE REPORT used (Total process time): real time 6.08 seconds cpu time 6.26 seconds

Table.9 Screenshot of file-size comparison

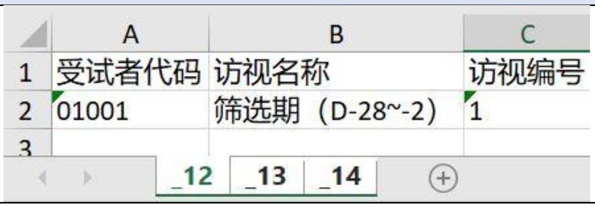
Output Worksheets	File Size
PROC EXPORT.xlsx	970 KB
PROC PRINT(ods excel).xlsx	1,010 KB
PROC REPORT(ods excel).xlsx	1,014 KB
PROC PRINT(ods html).xls	6,921 KB
PROC REPORT(ods html).xls	7,684 KB
PROC PRINT(ods tagsets.excelxp).xls	17,883 KB
PROC REPORT(ods tagsets.excelxp).xls	18,089 KB

Another key factor that significantly affects the output performance is the programming flexibility, because not everything could be designed as perfect as the final version. Until the last day of the project, programmers will be bothered by endless change requirements from their boss, their clients, their co-workers and other stakeholders. At the same time, it is not allowed to wait for the urgent moment to design the major part because QBD is everywhere. The more controllable the output phase is, the less time would be spent on repainting- a great example is the format loss found in **Table.3**. Of course, there are plenty of solutions such as adding another PROC SQL to create additional table or view right before the output step, then pick up the lost format $VISIT and install it back on the target variable “visit” (**Table.10**).

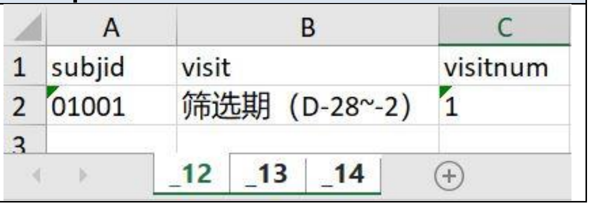
Table.10 Recover the lost format for PROC EXPORT

```
/*Prevent the format from losing*/
 proc sql; create view listview as select subjid,
   put(visit,$VISIT.) as visit label='访视名称', visitnum
   from listtb;
 quit;

/*Output using PROC EXPORT*/
 proc export data = listview outfile="&results.\&xlsname..xlsx"
     dbms = excel label replace;
     sheet = "&sheetid";
 run;
```

The revised PROC EXPORT output for Table.3



	A	B	C
1	受试者代码	访视名称	访视编号
2	01001	筛选期（D-28~-2)	1
3			



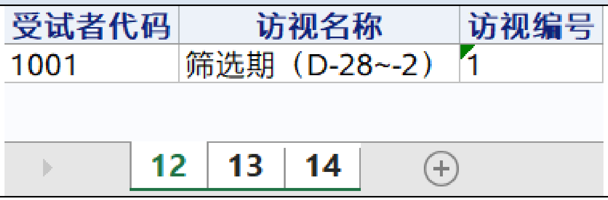
	A	B	C
1	subjid	visit	visitnum
2	01001	筛选期（D-28~-2)	1
3			

Compare with a lost format, missing an entire sheet of data could be much more disastrous as it destroys the upcoming program modules and creates a lot of unnecessary reprogramming especially the statistical related work. **Table.11** and **Table.12** seems to add only a few more dummy copies to expand the number of sheets of the same workbook, but if the old data such as the initial input tables and the preprocessed data sets were not properly dropped, the final output could be in chaos by continuously retaking identical variable names or even macro program names from the previous procedures. To solve this problem, the programmer has to go way back to several earlier steps to identify the suspicious pieces of codes. The least wanted move is to manually rename a bunch of conflicting tables and variables including those from external codes until the whole set of programs is cleaned.

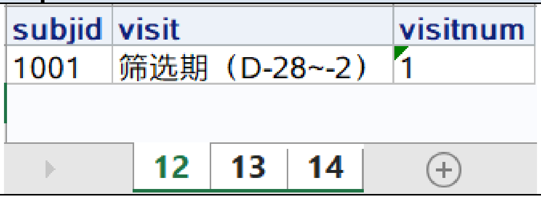
Table.11 Adding multiple printing steps in PROC PRINT

```
/*Output using PROC PRINT*/
 ods excel file="&results.\&xlsname..xlsx" options(sheet_name = "12");
 proc print data = listtb1 noobs ;
 run;
 ods excel options(sheet_name = "13");
 proc print data = listtb2 noobs ;
 run;
 ods excel options(sheet_name = "14");
 proc print data = listtb3 noobs ;
 run;
```

The revised PROC PRINT output for Table.2



受试者代码	访视名称	访视编号
1001	筛选期（D-28~-2）	1

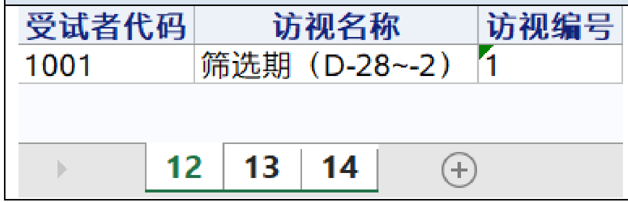


subjid	visit	visitnum
1001	筛选期（D-28~-2）	1

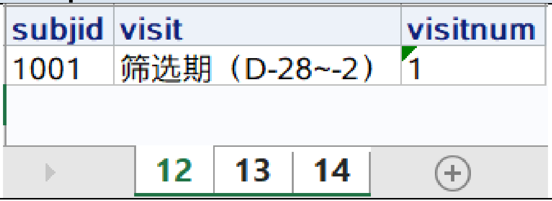
Table.12 Adding complex conditions in PROC REPORT

```
/*Output using PROC REPORT*/
 ods excel file="&results.\&xlsname..xlsx" options(sheet_name = "12");
 proc report data = listtb1 nowd style(column header)={vjust=m};
   column subjid visit visitnum;
   define subjid / display width=10 'subjid';
   define visit / display width=10 'visit';
   define visitnum / display width=10 'visitnum';
 run;
 ods excel options(sheet_name = "13");
 proc report data = listtb2 nowd style(column header)={vjust=m};
 run;
 ods excel options(sheet_name = "14");
 proc report data = listtb3 nowd style(column header)={vjust=m};
 run;
```

The revised PROC REPORT output for Table.4



受试者代码	访视名称	访视编号
1001	筛选期（D-28~-2）	1



subjid	visit	visitnum
1001	筛选期（D-28~-2）	1

DISCUSSION AND CONCLUSION

Table.13 Quantified Comparison

Aspects	PROC EXPORT	ODS Excel		ODS Tagsets.Excelxp		ODS html	
		PROC PRINT	PROC REPORT	PROC PRINT	PROC REPORT	PROC PRINT	PROC REPORT
Integrity	70	50	50	50	50	50	50
Effects	30	50	70	50	70	50	70
Speed	90	1	1	5	4	16	25
Flexibility	5	50	90	70	90	50	90
SpaceLeft	95	95	95	65	62	11	10
Overall	**200**	**246**	**306**	**240**	**276**	**177**	**245**

Based on the pros and cons of the discussed output methods, we have designed a summary tables to quantify the comparison. In **Table.13**, we have assigned a grading range from 1 to 100 as the key performance index to compare the ability of maintaining data integrity, readability, efficiency and controllability. For the evaluation of resource saving, we set a full remaining disk space of 20,000 kilobytes for each output file, then calculated the percentage of the free space left after the file was created, finally multiplied it by 100 to generate the grading score. By associating these values, we can clearly observe the result from the row named "Overall", which tells us that the PROC REPORT output procedure under the ODS Excel option has the highest score compare to others. However, utilizing the proper output techniques requires further analysis including programming habit, company SOPs, industry standards and additional project demands from the key stakeholders. Generally, the PROC EXPORT creates plain text sheets without graphical contents and color paintings but it is fast and less costly (Tom 2017, p. 1); The PROC REPORT could be the optimal choice for additional decorations (e.g. layout, background color, etc.) and the last-minute settings such as variable names, labels, conditional layouts; The PROC PRINT could be treated as something in between. Regarding some "outdating" ODS options, in many cases, people still need them for specific tasks rather than completely get rid of them. Just like some "non-mainstream" programming languages such as Pascal- it does not program that much by itself, but it teaches.

REFERENCES

SAS Institute. SAS® 9.4 Output Delivery System: User's Guide, Fifth Edition. April 26, 2019. http://documentation.sas.com/api/docsets/odsug/9.4/content/odsug.pdf.

Liao Wen, (2019, May 23). *Project PW9303(B003-101), Data Validation Plan (DVP). Version 2.0. OD1.sas*. Proswell Medical Company. https://www.proswell.com.cn.

Christopher J. Boniface, (2017). U.S. Census Bureau. *ODS TAGSETS.EXCELXP and ODS EXCEL SHOWDOWN*. SAS Global Forum 2017 proceedings. Paper 973.

Tom Bugg, (2017). Wells Fargo Home Mortgage. *Using the New ODS EXCEL Destination in SAS® 9.4 When Working with Remote Servers*. SAS Global Forum 2017 proceedings. Paper 169-2017. https://support.sas.com/resources/papers/proceedings17/0169-2017.pdf.

CONTACT INFORMATION

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

Name: Liao Wen
Proswell Medical Company
Email: vliaov@163.com
Website: https://www.proswell.com.cn