PharmaSUG China 2023 - Paper AA-141

PDR technology framework – A unified approach to efficiently build complex tables and stakeholder reports across different platforms and file types

Ming Zou, 1. RepTik Analytics Solution; 2. University Hospital Basel, Switzerland

ABSTRACT

Current programming methods for customized data analysis and stakeholder reporting, like clinical study reports, are mostly via individual “brute force” approaches and have multiple longstanding unmet needs for the last 30 years. Under the traditional report programming paradigms, programmers are wasting a lot of recurrent efforts on supportive jobs like format /layout /shell design coding and associated debug, QC, modify, manual edits, and trainings. It makes the programmers reluctant to iterate with the report stakeholders for changes, leads to highly fragmented practices, and is very difficult to collaborate across workflows like different platforms, teams, and file types.

After investigating the workflows across multiple platforms (SAS, R, SQL…), database teams, and file types (Excel, Word, PowerPoint…), we designed a patents-pending multi-platform Presentation Data Referencing (PDR) technology framework to address the multiple longstanding pain points in one go. With this new backbone design, programmers can create, format, and modify the complex tables & stakeholder reports with the ease and power of different office software and fill in with data from different platforms automatically via some special referencing structures. The recurrent coding /debug /QC /training jobs on the format /layout /shell design can therefore be thoroughly eliminated.

Consequently, users can save up to 70% reporting time, iterate flexibly with stakeholders to adopt changes for more insightful analyses & reports, and maintain easily across workflows. PDR technology is capable of boosting productivity, collaboration, and inter-operability of analyses & reports across different programming platforms, team practices, and report file types.

In this paper, we will explain the design principle behind the new and powerful PDR technology framework, introduce a new application, the RepTik Analytics Solution, that applies the PDR technology, and present some examples that use SAS and RepTik to produce Excel and Word reports.

INTRODUCTION

Healthcare generates 30% of the world’s data volume. And we usually use customized programming to analyze the data and present the results in a stakeholder report for further decision making. To produce such a stakeholder report, like a clinical study report, we typically need to generate the value-adding results from the data and generate the professional shell design to present the results to stakeholders. Both parts are necessary, because we need the solid results and insights to guide our business forward and we need the good design looking and reproducibility to show our professionalism and high team quality. For example, we often design and program a table shell with layouts of rows and columns, labels of patient attributes and patient groups, and formats of table borders, text fonts /sizes /spacing. On the other hand, we also program corresponding numeric results for filling into the table shell.

However, current programming methods for such customized data analysis and stakeholder reporting are mostly via individual “brute force” approaches and have multiple longstanding unmet needs for the last 30 years (see Figure 1). Particularly for generating the appropriate shell design, under the traditional programming paradigms, programmers are wasting a lot of recurrent efforts on jobs like format /layout /shell design coding and associated debug, QC, modify, manual edits, and trainings. These supportive activities are often tedious, boring, time-consuming, and error prone.

Before the programming step, depending on the selected file types, the shell design often has already been generated with the Microsoft Office software in Excel, Word, or PowerPoint. Then, programmers need to re-do everything a second time in the selected programming platforms/languages, which commonly have limitations to implement different advanced and powerful features provided in Excel,

Word, or PowerPoint. Let alone the tedious efforts to learn different programming skills and details for the different file types and different complex team practices for such reports. Once implemented, programmers are reluctant to adopt useful new ideas or even small changes from the stakeholders and push back jobs because of the pain points.

Furthermore, since there's no common technology framework available across different programming platforms, team practices, and report file types, this field is highly fragmented and very difficult to collaborate or maintain across different workflows, within or outside of an organization. For example, in a federated multi-center study, each center will synthesize a "sub" report using their own workflows and then someone need to use the painful and error prone copy-pasting-editing process to combine all the results in a master study report.

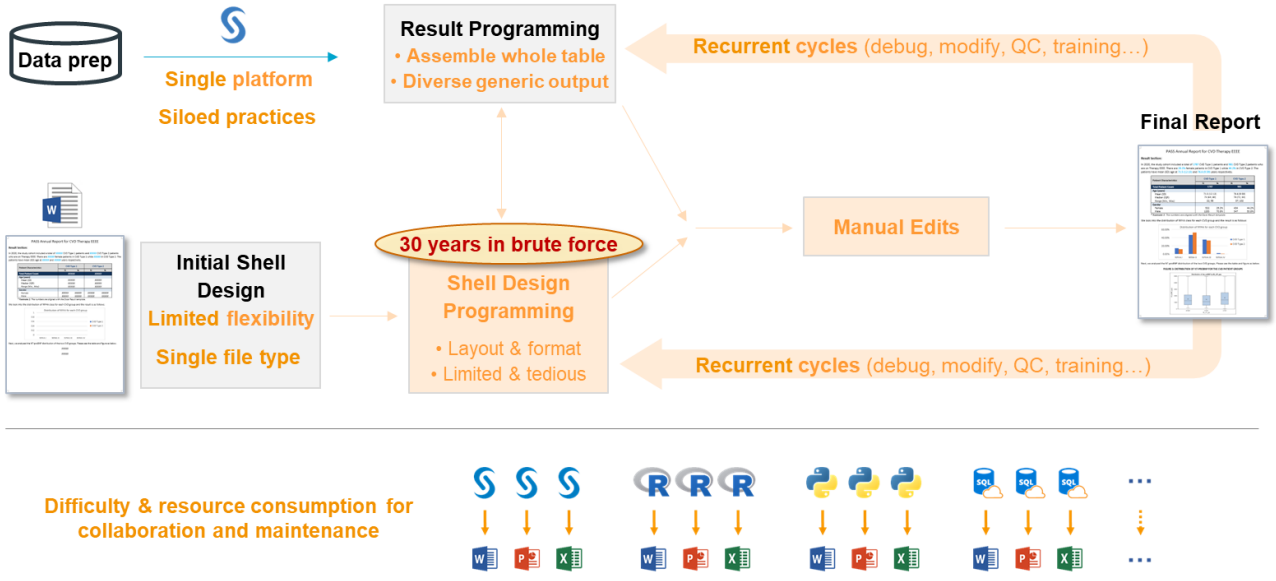


Figure 1. Longstanding pain points of traditional report programming paradigms

To satisfy different purposes, programmers have to tradeoff between quality, efficiency, flexibility, scalability (extendibility /compatibility), easiness to use (learn /maintain), reproducibility, stability and/or safety, etc.. They often have to stick to a specific limited solution for most parts of their daily jobs, and may further need to learn and maintain a zoo of platforms and solutions. In the end, the overheads of any aspects will all translate into high costs.

After investigating the workflows across multiple platforms (SAS, R, SQL…), database teams, and file types (Word, Excel, PowerPoint…), we designed a patents-pending multi-platform technology framework to address the multiple longstanding pain points in one go. We name this new technology framework as "Presentation Data Referencing (PDR)". In this paper, we will explain the design principle behind this new and powerful PDR technology framework, introduce a new application, the RepTik Analytics Solution, that applies the PDR technology, and present some examples that use SAS and RepTik to produce an Excel report and a Word report.

1. PDR TECHNOLOGY FRAMEWORK – A UNIFIED APPROACH

Under the traditional reporting workflows, both the shell design and the result generation have to be implemented in the same programming platform and workflow, and often inter-mingled with each other. Programmers have to be very careful so that the two can adapt with each other. A large amount of unnecessary efforts were wasted in implementing the shell design a second time in the programming platform, comparing to the little efforts spent on its first creation during the analysis planning via a dedicated software with powerful features (e.g. Excel, Word, PowerPoint, or special tools).

1.1 DESIGN PRINCIPLE

In our new technology design, we directly leverage the shell design generated at the first time via a powerful dedicated software, without implementing it a second time in the programming process. We consider a principle similar to how a computer use indexes or location references to operate information in a numeric array, and applies such a principle to operate result data to be presented in a stakeholder report (see Figure 2).

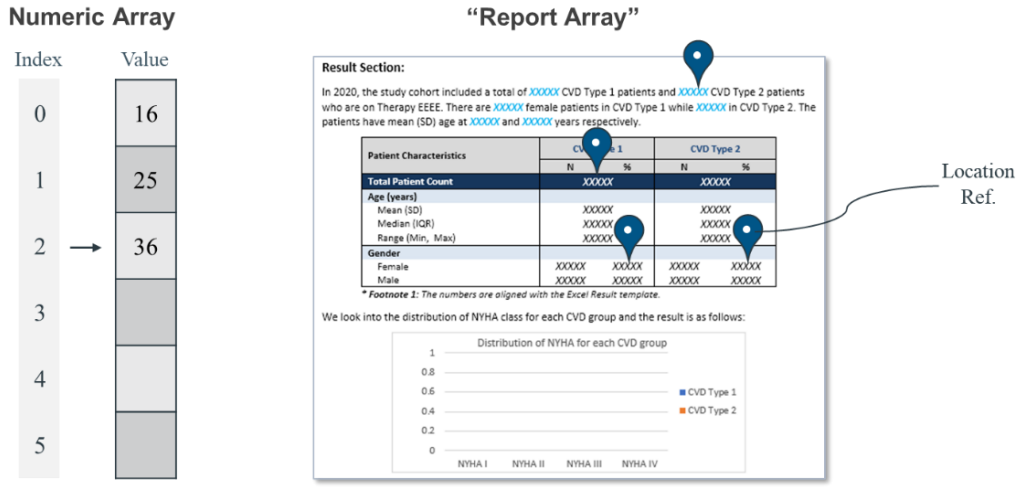


Figure 2. Design principle of the PDR technology

We view the initial shell design as a "report array" that contains a combination of contents like layouts, formats, and result data placeholders at different locations (e.g. number, text, images, etc.). After giving each result data placeholder a special reference and assigning the same reference to corresponding result data during the result programming, we can use automation to inject the individual result data directly into the result presenting locations and get the final view of the report (see Figure 3). We named this process or framework as the "Presentation Data Referencing (PDR)" technology.

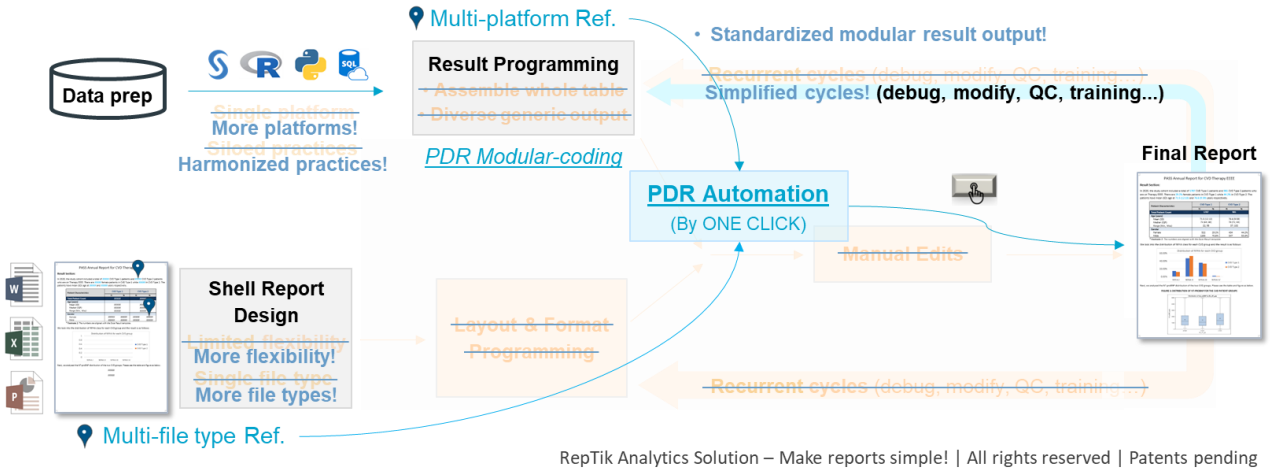


Figure 3. PDR technology framework – a unified approach to generate stakeholder reports

1.2 A MULTI-PLATFORM REFERENCING SYSTEM

To efficiently apply the PDR technology in actual heterogeneous production environments across different programming platforms and different report file types, we need an advanced referencing system that satisfies three conditions at the same time.

Firstly, the referencing system should work compatibly across platforms like SAS, R, Python, SQL etc. and file types like Excel, Word, PowerPoint, etc.. Secondly, this referencing system can be automatically calculated for different locations within the shell design (particularly the tables), can be automatically generated and coupled to different results when executing the result production codes, and can be automatically executed by a software application to produce the final report. Thirdly, this referencing system should be not only computer recognizable but also human readable or programmer friendly, so that the programmers can easily implement the location references for individual result values during result programing and easily conduct their jobs during debug /QC /changes etc..

To achieve this, we have designed a multi-platform referencing system, which typically can refer to individual result locations of different tables, worksheets, rows and columns, in shell designs across different report file types. The location references can be automatically generated by different programming software. Further, a programmer can easily read a location reference and identify both its location in the shell design and the specific code line to calculate the result. For example, the location reference “t1|6|2” refers to a location at table “t1”, row “6” and column “2” (see Figure 4 and Figure 18).

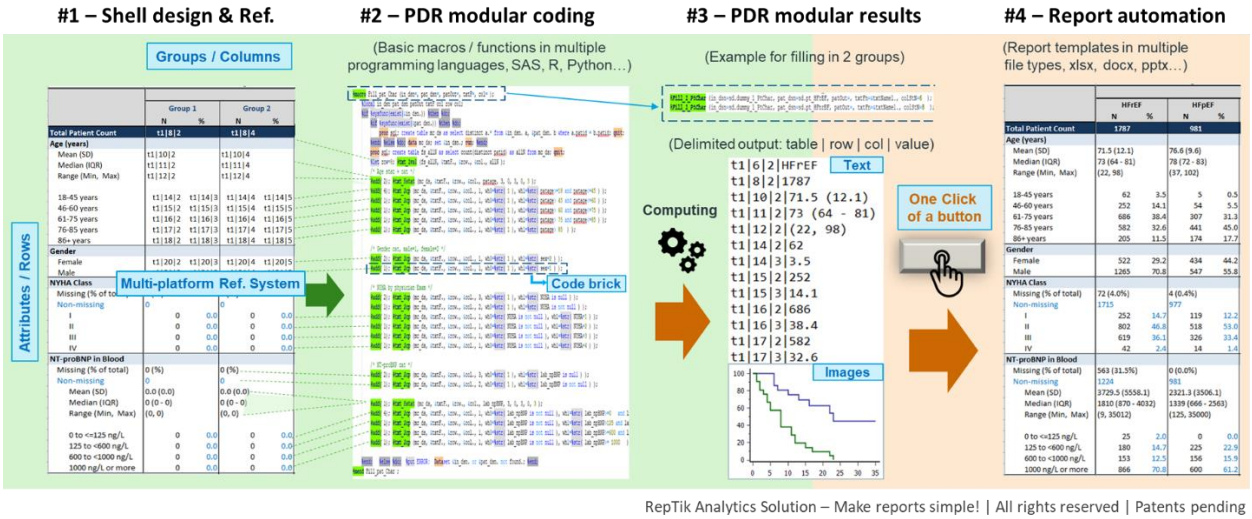


Figure 4. Multi-platform referencing system and modular coding workflow of PDR technology

1.3 A MODULAR CODING WORKFLOW WITH INTUITIVE 1-TO-1 ITERATIONS

The most basic and modular result “bricks” will provide the most flexibility to organize and present the results in the shell design, and allow generating the results in a modular way during programming without worrying about any “monolith” layout or format programming limitations. So, we adopted the modular coding concept in designing the basic macros or functions for generating the individual “brick” results and coupling such “brick” results with their corresponding location references. For calculating each attribute or row of results, one specific basic macro or function can be called (see Figure 4 and Figure 8).

Such a 1-to-1 modular coding workflow of the results and location references can actually be viewed as a new, much simpler and much intuitive iteration approach between the programmer and the computer, or between the different dedicated powerful shell design software and the analysis result programming software, by skipping all the layout & format programming steps. The users will have much less mental burden on keeping the shell design and the result programming working with each other.

1.4 STANDARDIZED MODULAR RESULT FORMAT

It is worthwhile to point out that, in most of current reporting techniques the programmers typically think of the result data as monolith “blocks” (e.g. a whole table or a section of the table), while the PDR technology drills down the result data into the most basic “bricks” (e.g. individual values for each cell of a table, like Lego bricks, see Figure 4).

Following this idea, we further developed some standardized modular formats for storing the basic “bick” result data output. For number/text values, we use a delimited line format like “table|row|column|value” and save in flat files named like “txt__SheetName__TextNote.txt”. For graph or figure output we save them as image files named like “img__SheetName__row__col__TextNote.png”, where the location reference is embedded in the file name. The double underscores “__” in the file names work as a delimiter. Such result format can be easily applied across the different programming or computing platforms (see Figure 4).

1.5 COMPARISON TO SAS-DDE TECHNIQUE

In some extent, the PDR technology is very similar to the diminished powerful SAS-DDE for Excel technique. But because SAS-DDE works only with SAS PC platform and only when Excel is co-installed in the same PC, and is unstable and unsafe, it was discouraged by Microsoft from 20 years ago. Even some follow-up applications have been developed to replace SAS-DDE, like SAS-OLE and SAS Add-In for Microsoft Office, their performance are not as satisfactory in many situations.

In contrast, the PDR technology framework is platform agnostic (applicable in SAS PC, SAS Grid, R, Python, SQL etc.) and report file type agnostic (applicable for Excel, Word, PowerPoint, PDF, and OpenDoc, etc.), and is much more stable and safer. In other words, you can inject results of numbers, text, and images generated from different platforms directly into Excel, and even Word and PowerPoint shell design files as simple and easy as what you can do with SAS-DDE only for Excel. You have no need to program and create the shell design file but directly modify an existing one. For this purpose, the SAS-DDE users may consider PDR technology as a better replacement option.

2. REPTIK ANALYTICS SOLUTION – APPLYING PDR

Based on the PDR technology framework, we have developed a new application, the RepTik Analytics Solution, to facilitate the programmers’ daily data analysis and stakeholder reporting workflows.

The RepTik application contains two main components. The first component is a software program with a very simple interface to assign location references automatically according to the shell design and to inject standardized modular results automatically into the shell design base upon the assigned location references (see Figure 5). The second component is a programming package of macros or functions compatible with the user’s programming platform to calculate results, couple results with the assigned location references, and output in a standardized modular format (see Figure 4).

The RepTik application has been validated in >20 formal production projects with robust performance.

2.1 REPTIK SOFTWARE PROGRAM

The simplicity design of the product is aiming to minimize the programmers’ mental burdens on generating the shell design, i.e., they can create, format, and modify the complex tables & shell design with the ease and power of office software and without tedious programming (including debug /QC /training etc.), and can generate the final view of the report with all results populated by ONE CLICK of a button. In this way, the programmers can save a lot of efforts from the shell design jobs and concentrate on the value-adding result generation jobs.

The current RepTik application software component is a minimal viable product that was built with VBA. It can deal with report shell design in file types like Excel (e.g. *.xlsx, *.xls), Word (e.g. *.docx, *.doc, *.rtf), and PowerPoint (e.g. *.pptx, *.ppt). The set of buttons in column “Add Tags” are to assign location references /tags automatically according to the shell design file. The set of buttons in column “Fill Results” are to inject standardized modular results automatically into the shell design based upon the assigned

location references. The button of “List Result wd” is to perform dynamic table listing, namely batch listing of result data with variable number of rows.

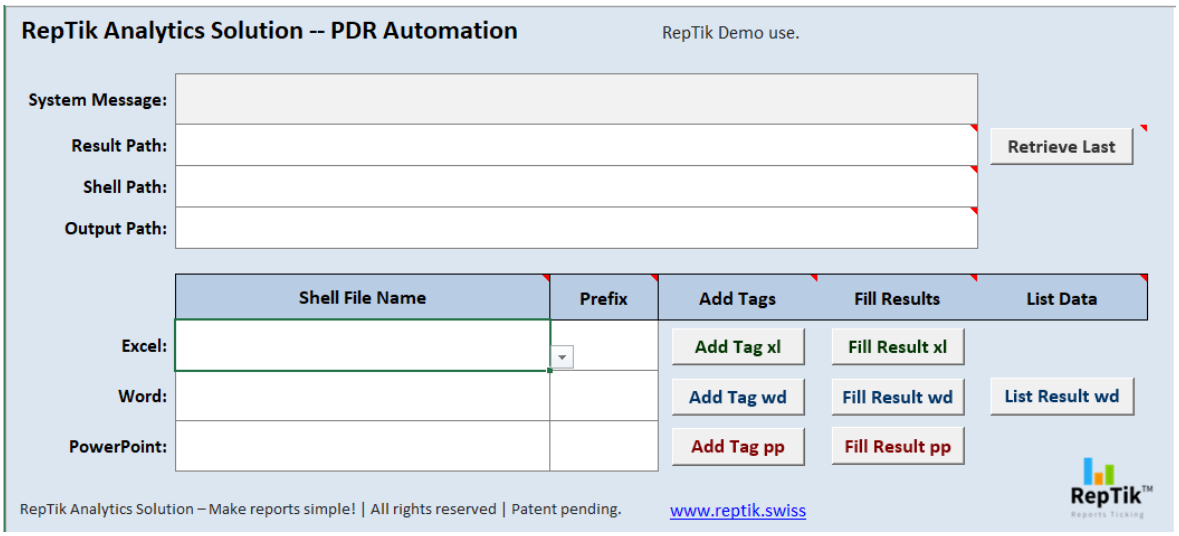


Figure 5. Software interface of RepTik Analytics Solution

2.2 REPTIK PROGRAMMING PACKAGE

The programming package component is currently available in SAS and R, with Python and SQL under planning. Depending on the user situation and preference, there’re two ways to use the package component: 1) plug RepTik into the last step of an existing workflow, that is to use your existing tools or code to generate “block” tables and figures, then in the last step output in RepTik modular format; or 2) integrate the RepTik PDR modular coding system into the existing workflow, that is to calculate individual basic “brick” values and figures with RepTik programming package tools and output the results directly in RepTik modular format (see Figure 4).

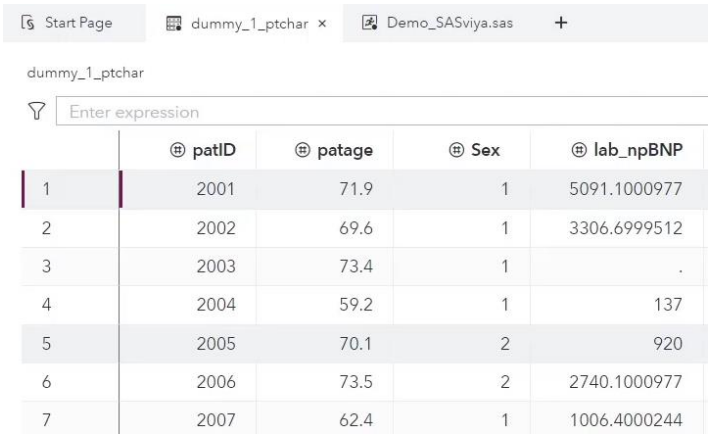
Particularly, the RepTik PDR modular coding system has a near 1-to-1 relationship between the analysis code lines and the result locations within the shell design, which gives the most flexibility to organize and present the results in the shell design (e.g. taking advantages of the advanced displaying features in Excel, Word, PowerPoint, etc.) and allows generating the desired results without worrying about any layout or format limitations casted by the programming platforms.

We would like to transform the package component into different open-source projects for free use and distribution among different programming platforms and communities. The aim is to let all RepTik users have 100% freedom to create their own PDR macros and functions, share efforts with each other and increase the diversity of tools for PDR technology across different areas.

3. EXAMPLES IN SAS FOR EXCEL AND WORD

With a high-level understanding of the PDR technology framework and the RepTik application workflow, we will now look into some detailed examples applying PDR technology in the SAS programming platform. Assuming we are running a phase IV post authorization safety study, it is on a few thousands of patients with a cardiovascular disease.

We have a dummy patient characteristics dataset in SAS format. For each patient ID, there are some characteristics variables like age, sex, baseline lab test values and diagnosis groups (see Figure 6). In the analysis, we are requested by an internal stakeholder, a clinical expert, to produce a table and a figure in an Excel report, as well as a table in a Word report. Both reports will be used for discussions with external stakeholders and therefore should look professional and be of good quality.



Start Page | dummy_1_ptchar | Demo_SASviya.sas

dummy_1_ptchar

Enter expression

	patID	patage	Sex	lab_npBNP
1	2001	71.9	1	5091.1000977
2	2002	69.6	1	3306.6999512
3	2003	73.4	1	.
4	2004	59.2	1	137
5	2005	70.1	2	920
6	2006	73.5	2	2740.1000977
7	2007	62.4	1	1006.4000244

Figure 6. A dummy patient characteristics dataset

3.1 EXAMPLE 1 – BUILD A NEW TABLE IN EXCEL

After discussion with the clinical expert, we agree to produce some statistical results of age and sex for CVD Type 1 and Type 2. The clinical expert has asked to put a heart logo, make some nice layout and formatting for the table shell design in Excel.

As compared to SAS-DDE for Excel technique, the way to build a table shell design in Excel and then to inject results directly into the shell design is quite similar. So, we can do all the required modifications easily with Excel software's powerful formatting and editing features – without any SAS programming, and no table headers or attribute text to be entered elsewhere for a second time.

Step 1: Shell design and location references

The table shell design created by Excel looks like that in Figure 7 and the worksheet name is "1_pat". Here we have no need to use RepTik to generate any explicit location references in the Excel shell design because each Excel cell has already got its worksheet, row and column referenced on the Excel software interface.

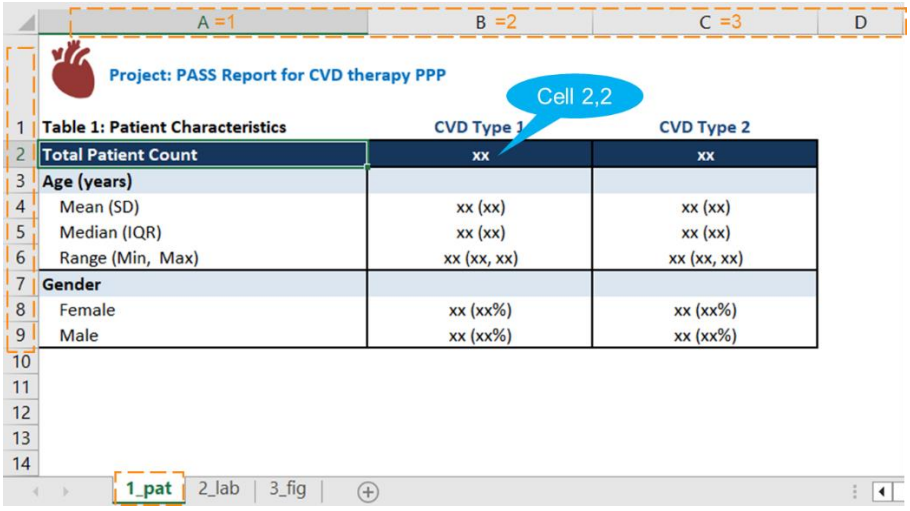


Figure 7. Shell design of a patient characteristics table in Excel

Step 2: PDR modular coding

Next, we use SAS to program the result output according to the table shell design from the dummy patient dataset (see Figure 8). RepTik's programming package has provided a series of basic "brick" macros for descriptive analysis, e.g. to output a value from a variable (macro #1, `%txt_1val()`), to calculate mean(SD) /median(IQR) /min-max (macro #3, `%txt_5stat()`), and to calculate count and percentage (macro #4, `%txt_3cp()`). We will use such basic "brick" macros to first assemble a modular code block for one group / column, and then wrap them up as a "block" macro for other groups (macro #5, `%Fill_1_pat()`).

To couple the correct location references with the results in modular coding, the table name, the starting row number and the starting column number of the first cell containing a result will be important. The rest cells' locations are all relative to the first cell (see Figure 7). In this way, the row number for the series of results of a column can be automatically calculated according to the relative position versus the last result value output – typically "shifted" by 1 or 2 (macro #2, `%add()`). We output the calculated result values and the location references in a flat file named "txt__1_pat__demo.txt". When going for the next group / next column, we shift the starting column number by 1 (macro #5, `%Fill_1_pat()`) and use corresponding patient ID dataset.

Sometimes, you may want to have 2 columns for each group, or swop the rows and columns in the shell design, you just assemble the basic "brick" macros accordingly and shift the row/column numbers accordingly, or further customize your own basic "brick" macros.

Note that for simplicity, some common error handling code snippets are omitted from the example.

```
%macro Fill_1_pat (txtFn=, tabNM=, colStN=, dsIN=, dsPAT= );
    %local row col;
    /*==== select subgroup patient data ====*/
    proc sql;
        create table mc_patPOP as select distinct a.* from &dsIN. a, &dsPAT. b where a.patid = b.patid;
    quit;
    %macro add( add );
        %local add; %let row=%eval(&row.+&add.);
    %mend add;
    /*==== START Fill... ====*/
    /* total N of input dataset */
    proc sql; create table mc_allN as select count(distinct patid) as allN from mc_patPOP; quit;
    %let row=2; %txt_1val( &txtFn.,&tabNM.,&row.,&colStN., mc_allN, allN );

    /* Attribute Age - num */
    %add( 2); %txt_5stat( &txtFn.,&tabNM.,&row.,&colStN., mc_patPOP, patage, meanSD=1, mediIQR=1, minmax=1);

    /* Attribute Gender - cat, male=1, female=2 */
    %add( 4); %txt_3cp( &txtFn.,&tabNM.,&row.,&colStN., mc_patPOP, 1, sql_wh0=%str( 1), sql_wh1=%str( sex=2));
    %add( 1); %txt_3cp( &txtFn.,&tabNM.,&row.,&colStN., mc_patPOP, 1, sql_wh0=%str( 1), sql_wh1=%str( sex=1));

%mend Fill_1_pat ;

/* Define patient groups with filters */
data pt_CVD1 (keep= patid); set sd.dummy_1_PtChar; where CVD_type = "CVD-1"; run;
data pt_CVD2 (keep= patid); set sd.dummy_1_PtChar; where CVD_type = "CVD-2a" or CVD_type = "CVD-2b"; run;

/* Results for Table 1 */
%let txtFile1 = &dir_study./01_result/txt__1_pat__demo.txt ;
%Fill_1_pat( txtFn=&txtFile1., tabNM=t1, colStN=2, dsIN=sd.dummy_1_PtChar, dsPAT=pt_CVD1 );
%Fill_1_pat( txtFn=&txtFile1., tabNM=t1, colStN=3, dsIN=sd.dummy_1_PtChar, dsPAT=pt_CVD2 );
```

Figure 8. Modular coding to calculate individual result values and generate paired location references, and then output to a flat file

Step 3: PDR modular results

We run the SAS code in Figure 8 and it generates the result file as shown in Figure 9. This step is automatic and straight forward. We spend no efforts.

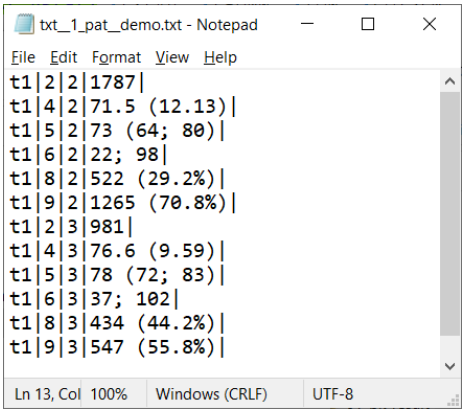


Figure 9. Modular results and location references generated by the modular code

Step 4: Report automation

Within the RepTik software, we enter the folder paths and shell file name, and then simply by ONE CLICK of the “Fill Result xl” button (see Figure 10). We immediately obtain the final Excel report with all results from the flat file created in Step 3 populated into the shell design (see Figure 11).

We can conduct further debug /QC /changes easily with the RepTik PDR workflow by repeating the activities in steps 1-4.

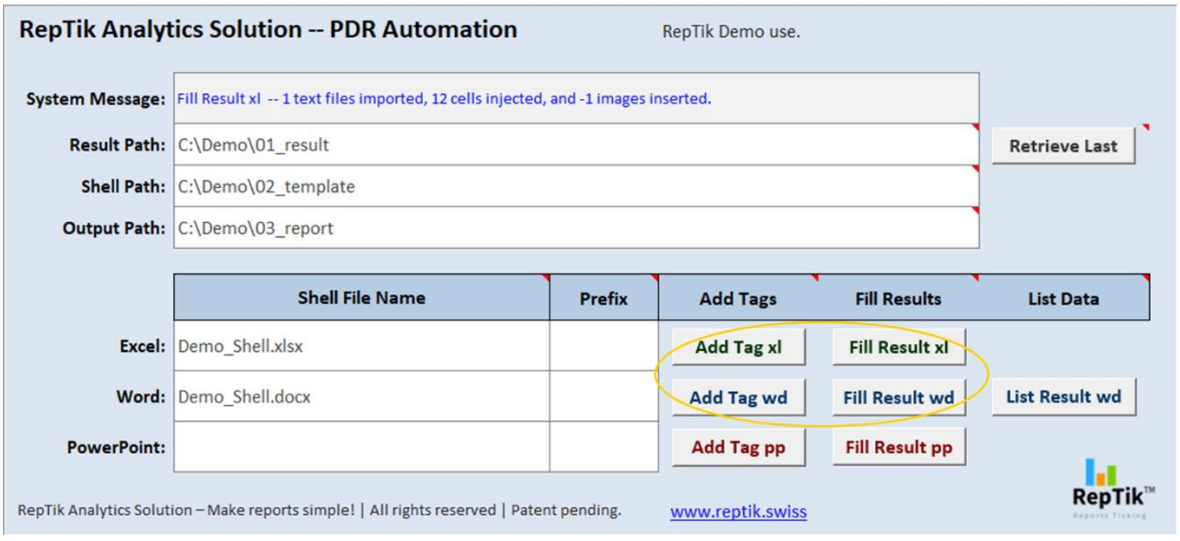
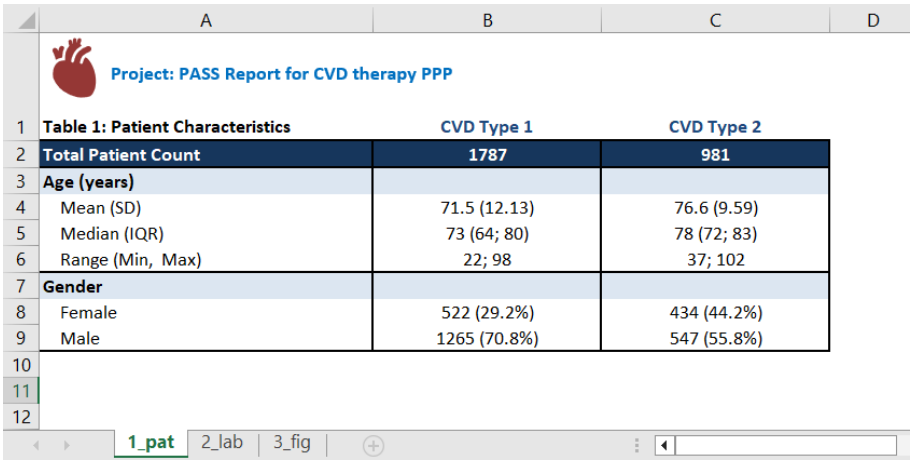


Figure 10. PDR report automation with the RepTik software by simple mouse clicks



Project: PASS Report for CVD therapy PPP

Table 1: Patient Characteristics	CVD Type 1	CVD Type 2
Total Patient Count	**1787**	**981**
Age (years)		
Mean (SD)	71.5 (12.13)	76.6 (9.59)
Median (IQR)	73 (64; 80)	78 (72; 83)
Range (Min, Max)	22; 98	37; 102
Gender		
Female	522 (29.2%)	434 (44.2%)
Male	1265 (70.8%)	547 (55.8%)

1_pat | 2_lab | 3_fig

Figure 11. Final Excel report generated by the RepTik software

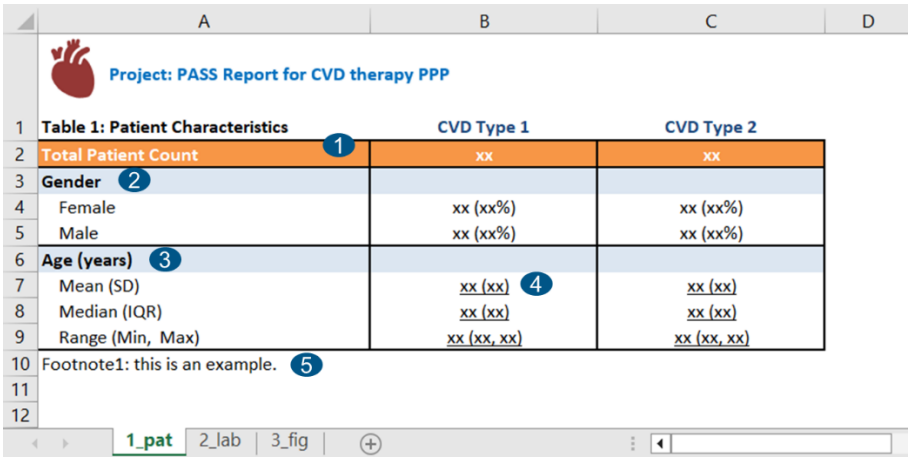
3.2 EXAMPLE 2 – MODIFY AN EXISTING TABLE IN EXCEL

The story continues. We deliver the final Excel report to the internal clinical expert and discuss whether there's any updates need to be done. The clinical expert finds that the color theme is better to use orange, the gender results are better to be placed above the age results, some footnotes need to be added, and some result highlight is necessary (see Figure 12). Some changes are important for the business and some changes will make the report looks more professional. We follow the instructions from the clinical expert, and repeat the steps 1-4 under the RepTik PDR workflow.

For step 1, we apply all the changes directly on the initial shell design using the Excel software, very simple tasks and no programming of the updated shell design (see Figure 12).

For step 2, we only need to do two little code edits. a) Swop the few "code brick" lines for the age attribute with the few "code brick" lines for the gender attribute (see the two orange boxes in Figure 8). The updated code is in Figure 13. b) Update the relative row shifts from "Total patient count" row to "Female" count and percentage row, that is from "4" to "2" (see the yellow circle in Figure 13). Then, run the code.

No action for step 3 and only ONE CLICK of the same button in RepTik software for step 4. Afterwards, we obtain the updated Excel report in Figure 14. All changes are immediately implemented in the report.



Project: PASS Report for CVD therapy PPP

Table 1: Patient Characteristics	CVD Type 1	CVD Type 2
Total Patient Count ①	xx	xx
Gender ②		
Female	xx (xx%)	xx (xx%)
Male	xx (xx%)	xx (xx%)
Age (years) ③		
Mean (SD)	xx (xx) ④	xx (xx)
Median (IQR)	xx (xx)	xx (xx)
Range (Min, Max)	xx (xx, xx)	xx (xx, xx)
Footnote1: this is an example. ⑤		

1_pat | 2_lab | 3_fig

Figure 12. Multiple edits of the shell design without tedious programming

```
/* Attribute Gender - cat, male=1, female=2 */
%add( 2); %txt_3cp( &txtFn.,&tabNM.,&row.,&colStN., mc_patPOP, 1, sql_wh0=%str( 1), sql_wh1=%str( sex=2));
%add( 1); %txt_3cp( &txtFn.,&tabNM.,&row.,&colStN., mc_patPOP, 1, sql_wh0=%str( 1), sql_wh1=%str( sex=1));

/* Attribute Age - num */
%add( 2); %txt_5stat( &txtFn.,&tabNM.,&row.,&colStN., mc_patPOP, patage, meanSD=1, mediIQR=1, minmax=1);
```

Figure 13. Simple update of the modular code according to the edited shell design

	A	B	C	D
	Project: PASS Report for CVD therapy PPP			
1	**Table 1: Patient Characteristics**	**CVD Type 1**	**CVD Type 2**	
2	**Total Patient Count**	**1787**	**981**	
3	**Gender**			
4	Female	522 (29.2%)	434 (44.2%)	
5	Male	1265 (70.8%)	547 (55.8%)	
6	**Age (years)**			
7	Mean (SD)	71.5 (12.13)	76.6 (9.59)	
8	Median (IQR)	73 (64; 80)	78 (72; 83)	
9	Range (Min, Max)	22; 98	37; 102	
10	Footnote1: this is an example.			
11				
12				

1_pat | 2_lab | 3_fig

Figure 14. Updated Excel report with RepTik software by one click

3.3 EXAMPLE 3 – ADD A FIGURE IN EXCEL

The clinical expert is satisfied with the updated table. He/she further ask for a scatter plot to be added to the report. The worksheet to place the figure is “3_fig” and we want to put the figure starting at row 5 and column 2. The shell design looks like that in Figure 15. We generate the figure with regular SAS coding and save it with file name “img__3_fig__5__2__scatter.png”, which contains the location reference for this figure.

After getting the result in SAS, by only ONE CLICK of the same button in RepTik software, we soon obtain the final view of the Excel report as shown in Figure 16.

	A	B	C	D	E	F	G	H	I
1									
2	Project: PASS Report for CVD therapy PPP								
3	**Figure: NT-proBNP distribution against age for different groups.**								
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									

1_pat | 2_lab | 3_fig

Figure 15. A shell design to depict the figure in Excel

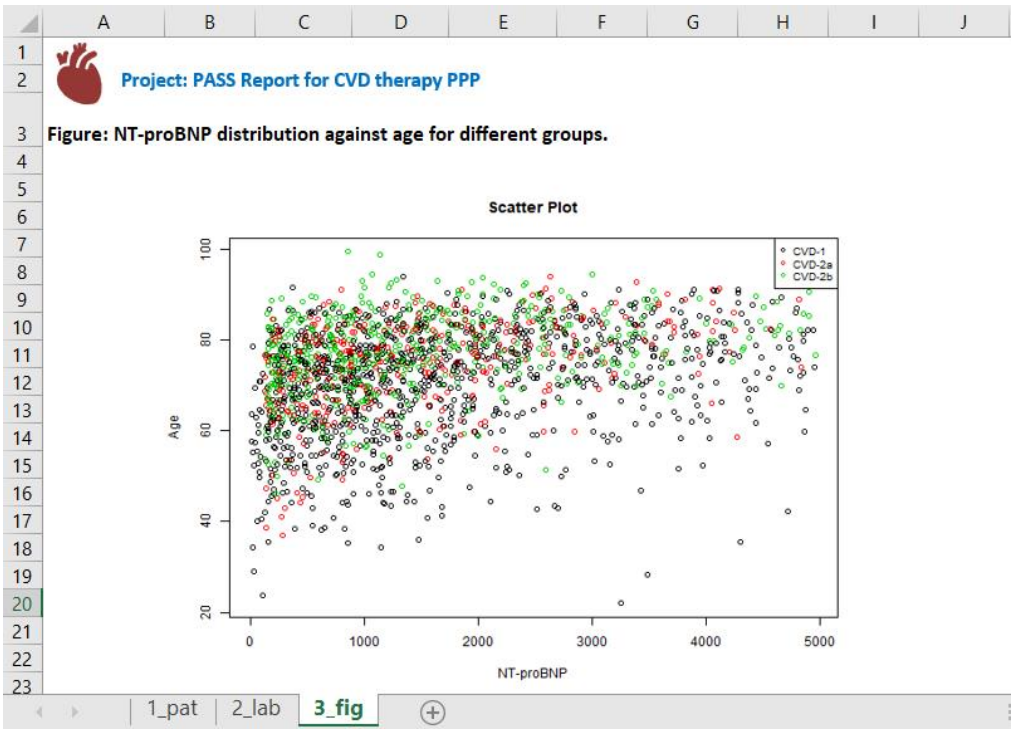


Figure 16. The figure was automatically inserted into the Excel report

3.4 EXAMPLE 4 – BUILD A NEW TABLE IN WORD

As requested by the clinical expert, he/she further needs the patient characteristics table to be presented in a Word report. The 4 steps in RepTik PDR workflow are essentially the same as in Example 1, except that in step 1 we will generate explicit location references for the table shells in Word file to facilitate our modular coding – because Word software interface doesn't show any explicit references for the cells within a table like Excel does.

Step 1: Shell design and location references

Here, the initial shell design can be built with Word easily and looks like that in Figure 17. We generate the location references by as simple as ONE CLICK of the button "Add Tag wd" on the RepTik software interface (see Figure 10). Then, we obtain immediately the intermediate "functional" shell design looks like that in Figure 18.

Table 1: Patient Characteristics	CVD Type 1	CVD Type 2
Total Patient Count	**xx**	**xx**
Age (years)		
Mean (SD)	xx (xx)	xx (xx)
Median (IQR)	xx (xx)	xx (xx)
Range (Min, Max)	xx (xx, xx)	xx (xx, xx)
Gender		
Female	xx (xx%)	xx (xx%)
Male	xx (xx%)	xx (xx%)

** **Footnote 1**: The ogeuzzfb skkfj llfuwii bbvgsfw.*

Figure 17. Shell design of the patient characteristics table in Word

Table 1: Patient Characteristics	CVD Type 1	CVD Type 2
Total Patient Count	**t1|2|2**	**t1|2|3**
Age (years)		
Mean (SD)	t1|4|2	t1|4|3
Median (IQR)	t1|5|2	t1|5|3
Range (Min, Max)	t1|6|2	t1|6|3
Gender		
Female	t1|8|2	t1|8|3
Male	t1|9|2	t1|9|3

***Footnote 1:** The ogeuzzfb skkfj llfuwii bbygsfw.*

Figure 18. Automatically generating PDR location references in the Word shell design with RepTik

Step 2: PDR modular coding

Based on the location references in Figure 18, we will now perform the PDR modular coding exactly the same as step 2 in Example 1 (see Figure 8).

This means, when we are planning to present the same results in different file types (e.g. one in Excel, one in Word, one in PowerPoint), with PDR technology we need to program only once of the results and RepTik can inject the same results to all the three file types. RepTik can help you avoid format and layout programming of across the three shell design.

Step 3: PDR modular results

We run the SAS code in Figure 8 and it generates the result file as shown in Figure 9. This step is automatic and straight forward. We spend no efforts.

Step 4: Report automation

This step is also automatic and straight forward. With RepTik software, we simply by ONE CLICK of the "Fill Result wd" button (see Figure 10). We immediately obtain the final Word report with all results from the flat file created in Step 3 populated into the shell design (see Figure 19).

We can conduct further debug /QC /changes easily with the RepTik PDR workflow by repeating the activities in steps 1-4, without programming any layouts and formats.

Table 1: Patient Characteristics	CVD Type 1	CVD Type 2
Total Patient Count	**1787**	**981**
Age (years)		
Mean (SD)	71.5 (12.13)	76.6 (9.59)
Median (IQR)	73 (64; 80)	78 (72; 83)
Range (Min, Max)	22; 98	37; 102
Gender		
Female	522 (29.2%)	434 (44.2%)
Male	1265 (70.8%)	547 (55.8%)

***Footnote 1:** The ogeuzzfb skkfj llfuwii bbygsfw.*

Figure 19. Automatically populating results into the final Word report with RepTik

CONCLUSION

We hope this paper has given you a feast for the mind – namely a new way of viewing a stakeholder report and a new technical angle for programming analysis results and generating such a stakeholder report. We can effectively throw away the recurrent iterative programming burden of shell designs (including all tedious layouts and formats) and concentrate on the more interesting and value-adding result programming jobs, yet yielding even higher quality better designed report deliverables.

Overall, with the PDR technology and the RepTik Analytics Solution, multiple users say they can save up to 70% reporting time, iterate flexibly with stakeholders to adopt changes for more insightful analyses & reports, and maintain easily across workflows. PDR technology is capable of boosting productivity, collaboration, and inter-operability of analyses & reports across different programming platforms, team practices, and report file types.

As a new backbone technology, there're a lot of new potentials or new features that can be developed. For example, adding further parameters like commands during the result output so that to perform specific actions on the final report.

You could find further information and some video examples, as well as user feedbacks on the PDR technology and the RepTik application in the website (https://www.reptik.swiss/).

REFERENCES

Vyverman, K. 2000. "Using Dynamic Data Exchange to Pour SAS® Data into Microsoft Excel." *Proceedings of the 18th SAS European Users Group International Conference*

Available at http://www.sas-consultant.com/professional/SEUGI18-Using-DDE-to-Pour-S.pdf

Bessler, LR. 2010. "SAS® and Excel, A Winning Combination, Part 2: Dynamic Data Exchange (DDE), a Popular Solution around the World" *MWSUG 2010 Conference Proceedings, Milwaukee, Wisconsin*

Available at https://www.mwsug.org/proceedings/2010/excel_db/MWSUG-2010-166.pdf

SAS Institute. 2017. "Introduction to SAS® Add-In for Microsoft Office" *SAS Communities* https://communities.sas.com/t5/Ask-the-Expert/Introduction-to-SAS-Add-In-for-Microsoft-Office/ta-p/357000

CONTACT INFORMATION

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

Ming Zou, MD PhD
RepTik Analytics Solution; University Hospital Basel
info@reptik.swiss
https://www.reptik.swiss/