PharmaSUG China 2018 – Paper AD-71

No Solution to Auto-generating acrf.pdf? Try to Use GROOVY Procedure!

Yin-Jhen Yan, PAREXEL International, Taipei, Taiwan
Kyle Chang, PAREXEL International, Taipei, Taiwan

ABSTRACT

Base SAS® is a flexible, extensible and powerful programming language, which enables programmers to accomplish tasks, including data management, analysis, and reporting. However, SAS® is limited when programmers attempt to develop automatic tools in order to improving efficiency and productivity. Beginning with SAS® 9.3, the GROOVY procedure enables to execute Groovy code on the Java Virtual Machine (JVM). Therefore, programmers are capable to run SAS code and Groovy code together inside a SAS program. PROC GROOVY will help programmers to get more things done. If you have no solution, let's try to use PROC GROOVY!

This paper provides automation approach that using PROC GROOVY to easily extracting necessary information from Case Report Form (CRF) to generate SDTM annotated Case Report Form (aCRF) for Rave electronic data capture (EDC) system.

INTRODUCTION

In pharmaceutical and biotechnology industry, sponsors send Electronic Submission (ESUB) to Food and Drug Administration (FDA) for drug applications. One of components for Study Data Tabulation Model (SDTM) ESUB is 'acrf.pdf', which includes the SDTM annotations indicate the dataset names and corresponding variable names in submitted data. In most cases, programmers routinely generate acrf.pdf by annotating annotations again and again, even though the CRFs are alike. Under the circumstances, programmers attempt to figure out how to use SAS to generate SDTM annotations automatically if the CRFs are similar.

You enable to refer to the 7 steps in this paper to develop a one-step macro to automatically generate SDTM annotations within 1 minute, which will dramatically reduce your time and keep consistency across your studies. Now, let's start to use PROC GROOVY as key step to automatically generate acrf.pdf, even if you don't know anything about Groovy code.

WHAT IS PROC GROOVY AND WHAT CAN IT DO?

Groovy is a dynamic language which runs on the JVM, the syntax is Java like, and most of the Java code can also work in Groovy. PROC GROOVY imply that programmers enable to run Groovy code on the JVM while run SAS code. There are hundreds of thousands of Java libraries on internet. To work with PDF documents, we are going to use the Apache PDFBox® library as an example in this paper. It is an open source Java PDF library and it allows creation of new PDF documents, manipulation of existing documents and the ability to extract content from documents. Namely, SAS programmers can use both SAS code and Groovy code for Apache PDFBox® under SAS environment, and capable to flexibly develop more automatic tools by using SAS. The application includes:

- Auto-generating annotations (comments) which are proposed in this paper.
- Combining multiple RTF/PDF file into one bookmarked PDF document for making review easier.
- Reading JSON data in order to analysis web data.

THE REQUIRED DOCUMENTS AND STEPS ABOUT GENERATING ACRF.PDF IN RAVE STUDY

You can simply understand the required documents and steps for generating acrf.pdf through below flow chart:

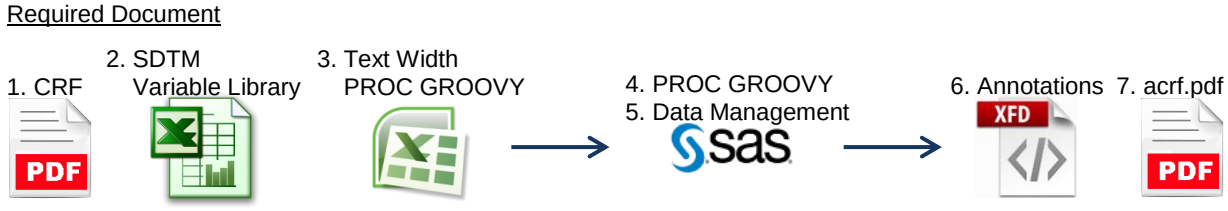

Figure 1. Flow chart about steps

The detail information is:

1. Getting Rave CRF with annotations in order to extract questions, corresponding options, variables, and position of texts.

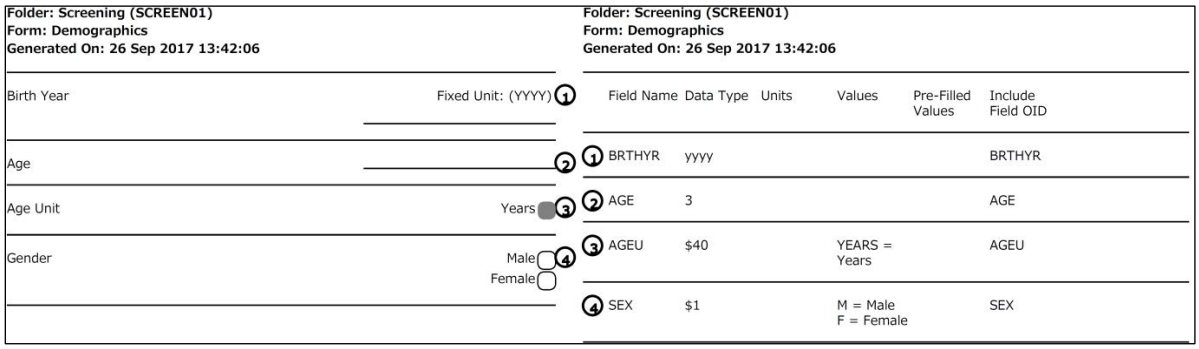

Figure 2. Sample of Rave CRF with annotations

2. Preparing standardized library include Rave variables and corresponding texts for annotations.

FormOID	FieldOID	SDTM Annotations
DEMOG	BRTHYR	BRTHDTC
DEMOG	AGE	AGE
DEMOG	AGEU	AGEU
DEMOG	SEX	SEX

Figure 3. Sample of standardize library for Rave variables

3. Using PROC GROOVY to generate file with the width of texts. The width of rectangular box (textbox) varied from different string if the font is not monospaced. We have to create such lookup table for SDTM annotations of the box.

DEC	FONT	STYLE	FONTSIZE	FONTWIDTH
66	Arial	Bold Italic	14	10.136871
66	Arial	Bold Normal	10	7.1911316
67	Arial	Bold Italic	14	10.136871
67	Arial	Bold Normal	10	7.1911316
68	Arial	Bold Italic	14	10.136871
68	Arial	Bold Normal	10	7.1911316

Figure 4. Sample of width for each text

4. Using PROC GROOVY to extract necessary information from CRF.

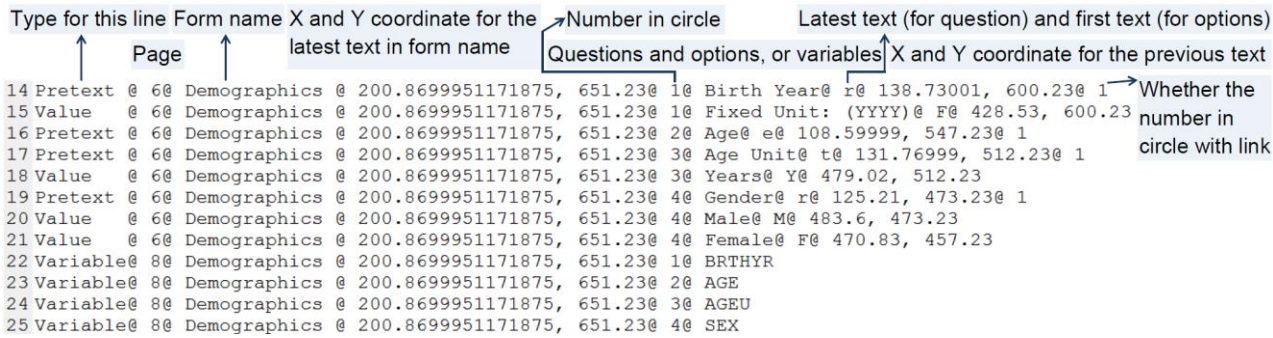

Figure 5. Sample of extracting necessary information

5. Managing above files to a SAS dataset.

	ANNOTATION	FREETEXT_RECT	FONT_SIZE	FONT_STYLE
9	DM = Demographics	90,702.23,231.3595566,720.23	14	italic
10	BRTHDTC	461.6632394,583.23,515.79004,596.23	10	normal
11	AGE	488.2067269,543.23,515.79004,556.23	10	normal
12	AGEU	444.2455553,508.23,479.02,521.23	10	normal
13	SEX	442.9000685,461.23,468.83,474.23	10	normal

	FONT_COLOR	FONT_NAME	BG_COLOR	BD_COLOR	FREETEXT_ROTATION	CRF_PAGES
9	#000000	Arial	#FFFFAA	0 0 0 rg	0	6
10	#000000	Arial	#FFFFAA	0 0 0 rg	0	6
11	#000000	Arial	#FFFFAA	0 0 0 rg	0	6
12	#000000	Arial	#FFFFAA	0 0 0 rg	0	6
13	#000000	Arial	#FFFFAA	0 0 0 rg	0	6

Figure 6. Sample of SAS dataset after data management

6. Converting the SAS dataset to XFDF file. For more details, please refer to “Developing annotated CRF: SAS, Excel and patience as your friends”, Ilias Pyrnokokis, PhUSE 2015.

```
107 ></freetext
108 ><freetext rect="90,702.23,231.3595566,720.23" creationdate="D:20180507224506" color="#FFFFAA" rotat
109 ><contents-richtext
110 ><body xmlns="http://www.w3.org/1999/xhtml" xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/" xfa:
111 style="font-size:14pt;text-align:left;color:#000000;font-weight:bold;font-style:italic;font-family:A
112 ><p style="text-align:CENTER"
113 >DM = Demographics</p
114 ></body
115 ></contents-richtext
116 ><defaultappearance
117 >1 1 0.6666666667 rg /Arial-BoldMT 10 Tf</defaultappearance
118 ><defaultstyle
119 >font: italic bold Arial 14pt; text-align:left; color:#000000 </defaultstyle
120 ></freetext
```

Figure 7. Sample of XFDF file

7. Import XFDF file into CRF and getting the automatic generating acrf.pdf.

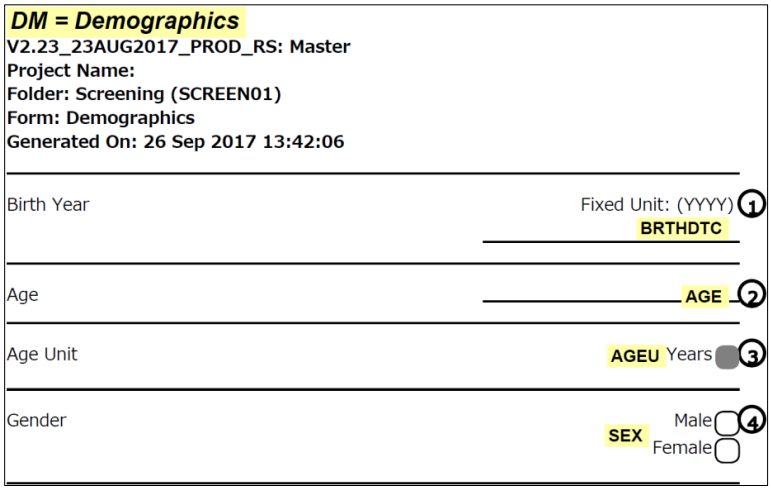

Figure 8. Sample of acrf.pdf

After preparing the documents from Step 1 to Step 3, you need to focus on developing programs about Step 4, 5, and 6. Next, you just import XFDF file into your CRF, you will get the acrf.pdf.

This paper proposes the approach that combining corresponding programs from Step 4 to Step 6, then a one-step SAS macro will be generated for auto-generating acrf.pdf. In other words, users only need to execute the one-step SAS macro (see the sample SAS macro below with required parameters) and import the XFDF file into the CRF. They will get the auto-generating acrf.pdf.

```
%autoSDTMaCRF(fileIn=<Rave CRF PDF file>, pathOut=<XFDF Output Path>);
```

THE KEY STEPS ABOUT EXTRACTING TEXTS, TEXT WIDTH AND TEXT POSITION

You enable to get or generate most of above files through Rave system, other papers in PharmaSUG, or by yourself. However, you may not have any idea about the file in Step 3. Using PROC GROOVY to generate file with the width of texts and Step 4. Using PROC GROOVY to extract necessary information from CRF. Thus, you can try to use some simple Java code in PROC GROOVY.

For example, you would like to extract position of texts, or extract texts from specific area, you can refer to the Apache PDFBox® (https://pdfbox.apache.org/) which allows you to deal with PDF documents. Before using Apache PDFBox®, you need to download pdfbox-app-version.jar and add class path in SAS options. The file of pdfbox-version-src.zip provides examples for your reference.

Please find the examples from the Apache PDFBox® website. Then, let's try to include Java code in PROC GROOVY in order to handle CRFs.

EXTRACTING WIDTH AND POSITION OF TEXTS

Extracting the position of "r" (from Year) and "F" (from Fixed) so that the position of annotations will automatically fit our expectation.

Figure 9. Sample of acrf.pdf for explaining the position of texts

1. Finding PrintTextLocations.java in the example folder or the link of this example in Apache PDFBox.
2. Copying and pasting the Java code into SAS.
3. Writing statements related to PROC GROOVY.

```
proc groovy;
    submit;
    Java code from PrintTextLocations.java
    endsubmit;
quit;
```

4. Providing the path and file name of CRF.

```
proc groovy;
    submit;
    Java code from PrintTextLocations.java
        public static void main( String[] args ) throws IOException
        {
        …
```

Provide PDF file name of Rave CRF

```
        public static void main( String[] args ) throws IOException
        {
            args = new String[1];
            args[0] = "/path/PDF file name of Rave CRF.pdf";
        …
    endsubmit;
quit;
```

5. Executing the program, and position of "r" and "F" is extracted from Rave CRF.

```
String[116.729996,191.77002 fs=10.0 xscale=10.0 height=7.3500004 space=3.39 width=6.300003]Y
String[123.03,191.77002 fs=10.0 xscale=10.0 height=7.3500004 space=3.39 width=5.7700043]e
String[128.8,191.77002 fs=10.0 xscale=10.0 height=7.3500004 space=3.39 width=5.800003]a
String[134.6,191.77002 fs=10.0 xscale=10.0 height=7.3500004 space=3.39 width=4.130005]r
String[428.53,191.77002 fs=10.0 xscale=10.0 height=7.3500004 space=3.39 width=5.7099915]F
String[434.24,191.77002 fs=10.0 xscale=10.0 height=7.3500004 space=3.39 width=2.5299988]i
```

Output 1. Output from above program for extracting position of texts

6. The width of text is the width as above, and you able to generate PDF file with all possible texts for your annotations. Then, using this Java code in PROC GROOVY, you will get the file with the width of texts.

EXTRACTING TEXTS FROM SPECIFIC AREA

Extracting the corresponding options so that the position of annotation should be adjusted to align the center across these options.

Figure 10. Sample of acrf.pdf for explaining the extracting texts from specific area

1. Finding ExtractTextByArea.java in the example folder or the link of this example in Apache PDFBox.
2. Copying and pasting the Java code into SAS.
3. Writing statements related to PROC GROOVY.

```
proc groovy;
    submit;
    Java code from ExtractTextByArea.java
    endsubmit;
quit;
```

4. Modifying the program and providing the path and file name of CRF.

```
proc groovy;
    submit;
Java code from ExtractTextByArea.java
    private ExtractTextByArea()
    …
    public static void main( String[] args ) throws IOException
    {
    …
            Rectangle rect = new Rectangle( 10, 280, 275, 60 );
    …
            PDPage firstPage = document.getPage(0);
    …
```

Modify the Java code and provide PDF file name of Rave CRF

```
    public ExtractTextByArea()
    …
    public static void main( String[] args ) throws IOException
    {
        args = new String[1];
        args[0] = "/path/PDF file name of Rave CRF.pdf";
    …
            Rectangle rect = new Rectangle( 470, 317, 31, 18 );
    …
            PDPage firstPage = document.getPage(the CRF page);
    …
    endsubmit;
quit;
```

The position is the area, which the texts will be extracted.

5. Executing the program, and the options are extracted from Rave CRF.

```
Text in the area:java.awt.Rectangle[x=470,y=317,width=31,height=18]
Male
Female
```

Output 2. Output from above program for extracting texts from specific area

CONCLUSION

To submit data for drug applications, the 'acrf.pdf' is required, but generating it is time-consuming if the CRFs are similar. Therefore, programmers attempt to develop an automatic tool for generating acrf.pdf so that they can increase efficiency and keep consistency of mapping rule across studies. This paper proposes a one-step automatic tool for generating acrf.pdf by using PROC GROOVY as a key step, which not only encourage programmers to develop this kind of tool, but also let programmers familiar with PROC GROOVY. After that, the extension of PROC GROOVY may be more and more, for example, checking the version, page size, font, and font size of PDF according to "Portable Document Format (PDF) Specifications", FDA.

REFERENCES

Hamilton, Jack. 2013. "Writing a Useful Groovy Program When All You Know about Groovy Is How to Spell It". SAS Global Forum 2013. Available at http://support.sas.com/resources/papers/proceedings13/493-2013.pdf.

Kennedy, John. 2016. "Reading JSON in SAS® Using Groovy". Available at http://support.sas.com/resources/papers/proceedings16/1660-2016.pdf.

Weiquan, Xin and John, Wang. 2016. "Utilizing SAS® and Groovy to combine multiple RTF/PDF reports to one bookmarked PDF document". PharmaSUG 2016. Available at https://www.lexjansen.com/pharmasug-cn/2016/PG/PharmaSUG-China-2016-PG21.pdf.

Apache PDFBox®. Available at https://pdfbox.apache.org/.

Luo, Haiqiang and Cao, Yong. 2015. "Automatic generating blankcrf.pdf for Rave Study". PharmaSUG 2015. Available at https://www.lexjansen.com/pharmasug-cn/2015/CD/PharmaSUG-China-2015-CD60.pdf.

Pyrnokokis, Ilias. 2015. "Developing annotated CRF: SAS, Excel and patience as your friends". PhUSE 2015. Available at https://www.lexjansen.com/phuse/2015/pp/PP29.pdf.

U.S. Food and Drug Administration. September 2016. "Portable Document Format (PDF) Specifications". Available at https://www.fda.gov/downloads/Drugs/UCM163565.pdf.

ACKNOWLEDGMENTS

Many thanks for Shelly Lee, Kai Yang, and Mark Lin of PAREXEL International for their great support during the period of developing auto-generating acrf.pdf, and encouraging us to attend PharmaSUG China 2018.

RECOMMENDED READING

- Base SAS® 9.3 Procedures Guide

CONTACT INFORMATION

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

Name: Yin-Jhen Yan
Enterprise: PAREXEL International
Address: 22F, Far Glory International Center, No. 200, Sec. 1, Keelung Road,
City, State ZIP: Taipei, Taiwan 11071, ROC
E-mail: Yin-Jhen.Yan@PAREXEL.com
Web: http://www.parexel.com/

Name: Kyle Chang
Enterprise: PAREXEL International
Address: 22F, Far Glory International Center, No. 200, Sec. 1, Keelung Road,
City, State ZIP: Taipei, Taiwan 11071, ROC
E-mail: Kyle.Chang@parexel.com
Web: http://www.parexel.com/