Intro to RRG Part 1

Basics

- RRG has 2 parts: SAS (macros) part and Java part
- You do not need to know anything about Java to use RRG

- SAS part has 2 sub-parts: **table** module and **listing** module

- Both **table** module and **listing** module produce sas dataset with the content of the report.
    - This SAS dataset has 3 parts, identified by __datatype variable
        - __datatype=RINFO has formatting instructions, titles, footnotes etc
        - __datatype=HEAD has information about the column header in the table or listing
        - __datatype=TBODY has the content of the table (what goes below the header)

- Java part is used to create RTF and/or PDF file and render this dataset in the desired format

Table Module

- Table module performs basic calculations for summary stats – what is available in PROC MEANS, some combined statistics (e.g.
  “Mean (SD)”,
  “MIN, Max”,
  (or whatever combination of statistics from PROC MEANS you may want),
  counts and percentages,
  and some basic stat models (ANCOVA, CMH, FISHER, CHI-SQ, BINOMEX, etc)
- Other - less basic - models can be written as RRG plugins (SAS macros).

Listing module

- Does not perform any calculations
- It is used in similar way as a very basic PROC REPORT

- Apart from producing listings, it may be used to produce statistical tables which are not easily generated with built-in RRG table functionality (example: MMRM)

Java component

- Uses formatting instructions from dataset created by table or listing module
- Calculates column widths and page breaks based on the user-specified preference (row in the table/listing module created dataset where __datatype=RINFO
- Row-based or cell-based formatting instructions (e.g. indentation, bolding of some rows, or adding cell borders, etc) can also be provided.

- You do not need to know anything about Java to use it
- But you need to provide formatting instructions that Java component will understand

Pre- and Post- processing

- %rrg_codebefore macro can be used to do data manipulation prior to calling RRG table or listing macros
- %rrg_codeafter (for tables) and %rrg_codeafterlist (for listings) can be used to make changes to the RRG-produced dataset before Java rendering of the output dataset happens.

  For example you may want to remove some rows from the display, or provide row or cell based formatting instructions, or change the text in some cells, or reorder the rows.

RRG listing macros

- %rrg_initlist
- %rrg_codebefore
- %rrg_deflist
- %rrg_defcol
- %rrg_genlist --- a macro parameter can be used to indicate that post-processing (%rrg_codeafterlist) will happen before output is rendered

- %rrg_codeafterlist

- %rrg_finalizelist

RRG Tables macros

- %rrg_init
- %rrg_codebefore
- %rrg_defreport
- %rrg_addtrt
- %rrg_maketrt
- %rrg_addgroup
- %rrg_addvar
- %rrg_addcatvar
- %rrg_addcond
- %rrg_codeafter
- %rrg_generate
- %rrg_finalize

Other RRG macros and functionality

- There are other RRG utility macros that can be used, e.g.
  - %Rrg_joinds to set or merge files (with an option to use cross join)
  - %rrg_inc to include external programs
  - %rrg_maketrt to create pooled treatments

- You can append one table to another table (e.g. appending lengthy footnotes at the end of the table)
- Formating options, titles, footnotes and defaults for table/listing macros can be specified in configuration file

RRG is a code generator

- RRG generates sas code (free of proprietary macros) in the folder you specify.
  The recommended location is to have a subfolder ../pgm/generated where this code is saved
- This saved code can be used to troubleshot issues
- The generated code goes as far as producing final dataset – but not the rendering of pdf/rtf output (unless the user has RRG installed)
- In order for generated code to work properly, care is needed about defining libnames and %include -- they must be placed within RRG macros "sandbox".
- If you do not follow recommendation in the previous bullet, you can easily modify generated code to define libnames and add %include(s) in the beginning of the generated code.

To install RRG:

1. Install sasshiato (see https://github.com/ipeszek/sasshiato)
2. Clone RRG project https://github.com/ipeszek/RRG. We will refer to root folder of cloned RRG project as simply RRG
3. Compile the RRG macros located in RRG\src folder into a folder of your choice
4. Create config.ini file (see RRG\docs\config.ini for example). The name is just an example, you can use whatever name you want.
5. Create init.sas file. You need to specify the location of compiled macros (rrgmacropath), the location of output files (rrgoutpath) , the location of the generated code (rrgpgmpath) and the location of config.ini file (rrg_configpath) and the location of sasshiato directory (__sasshiato_home). Please see RRG\docs\init.sas for example). This init.sas file has to be invoked in every RRG program. The name is just an example, you can use whatever name you want.

And you are done!