PharmaSUG 2021 - Paper AD-175

The SASSY System: Making R Easier for SAS® Programmers

David J. Bosak, r-sassy.org

ABSTRACT

The **sassy** system is an R meta-package that make R easier for programmers whose primary experience is with SAS® software. These functions provide the ability to create data libraries, format catalogs, data dictionaries, and a traceable log. The system includes a datastep function that recreates the most basic functionality of the SAS® data step. The package also includes reporting capabilities reminiscent of those found in SAS®, and can output reports in text, rich-text, and PDF file formats. All combined, these functions allow you to write programs with an overall flow and thought-process that is more similar to a SAS® program. This paper will provide a brief overview of the system.

INTRODUCTION

SAS® Programmers who come to R are often frustrated with how difficult R can be to perform common tasks. Tasks which take a small amount of code in SAS® can take dozens or hundreds of lines of code in R. Instead of focusing on their analysis, programmers spend many hours trying to accomplish a simple task like creating a log, loading data files into memory, or writing a report. The **sassy** system was built for these programmers. The system offers a set of packages that make programming in R much more similar to programming in SAS®.

The **sassy** system includes four separate R packages. While the packages can be installed and used individually, they were designed to complement each other. The system contains the following packages:

- **logr**: To create a traceable log
- **fmtr**: To create formats and format catalogs
- **libr**: To create a libname, a data dictionary, and perform a data step
- **reporter**: To create statistical reports

Together, the above packages constitute a coherent and well-designed system for managing and reporting on data in R. The system was developed in the context of creating statistical reports in the pharmaceutical industry. However, the functions are generalized such that they can be used in any vertical. The purpose of this paper is to give an overview of the system.

Note that the **sassy** system of packages was written independently, and the authors have no association with, approval of, or endorsement by SAS® Institute or RStudio®.

INSTALLATION

The **sassy** meta-package and its sub-packages are published on CRAN. The entire set can be installed by installing the **sassy** package, as follows:

```
install.package("sassy")
```

Once installed, the packages can be loaded with the following command:

```
library(sassy)
```

Note that if you want to use the PDF capabilities of the **reporter** package, you should also install pandoc and the miktex latex renderer. Text and RTF output formats require no additional installations or configuration.

HOW TO USE

The following section will give a brief overview of the major functionality of the **sassy** system.

CREATE A LOG

Programmers coming from SAS® are usually surprised to find there is no automatic logging in R. The **logr** package was written to provide a logging system that anyone can use, and make it as automatic as possible.

There are three steps to creating a log:

1. Open the log
2. Write to the log
3. Close the log

Below is a simple example that illustrates how to create a log with the **logr** package. The `put()` function writes any R object to the log, similar to a SAS® `%put` statement:

```
library(sassy)

# Open the log
log_open("Example1.log")

# Write to the log
put("Here is something to send to the log.")

# Close the log
log_close()
```

The generated log looks like this:

```
================================================================================
Log Path: ./log/Example1.log
Working Directory: C:/packages/Testing
User Name: dbosa
R Version: 4.0.3 (2020-10-10)
Machine: DESKTOP-IGGQ6UV x86-64
Operating System: Windows 10 x64 build 19041
Log Start Time: 2021-01-18 12:16:36
================================================================================

Here is something to send to the log.

NOTE: Log Print Time:  2021-01-18 12:16:37
NOTE: Elapsed Time in seconds: 1.70827198028564

================================================================================
Log End Time: 2021-01-18 12:16:41
Log Elapsed Time: 0 00:00:05
================================================================================
```

For a complete explanation of the capabilities of the **logr** package, see the logr documentation site.

CREATE A LIBNAME

The **sassy** system provides a `libname()` function that is quite similar to a SAS® `libname` statement. The function will import a set of related datasets, and assign a name to the entire set. The following code imports a set of related csv files and assigns them to the name "sdtm":

```
library(sassy)

# Get sample data directory
dir <- system.file("extdata", package = "sassy")

# Define library
libname(sdtm, dir, "csv")
```

Once the datasets have been imported, the libname can be loaded into the workspace and accessed using two-level dot notation, as you would do in SAS®. The following code loads the datasets into the workspace, prints the structure of the `sdtm.DM` dataset, and then unloads the library from the workspace:

```
# Loads data into workspace
lib_load(sdtm)

# Print structure of DM dataset
str(sdtm.DM)

# Unload library from workspace
lib_unload(sdtm)
```

The above library functions are found in the **libr** package. This package also contains several functions for manipulating data libraries. See the libr documentation for additional information.

DATA FORMATTING

The idea of a *format* is another foundational concept in SAS® software. In R, formatting is spread over many functions. The **fmtr** package aims to consolidate and simplify formatting in R, and make it more similar to the way you work with formats in SAS®. The package contains functions to create a user-defined format, apply formats to vectors and data frames, and create format catalogs.

The following example shows you how to create a user-defined format, and apply it to a column of data. Notice that the `value()` function is similar to the statements inside a SAS® `proc format` procedure, and that the `fapply()` function is defined in a way that is reminiscent of a SAS® data step `put()` function:

```
library(sassy)

# Get sample data directory
dir <- system.file("extdata", package = "sassy")

libname(sdtm, dir, "csv")

# Loads data into workspace
lib_load(sdtm)

# Create user-defined format
fmt_sex <- value(condition(is.na(x), "Missing"),
                 condition(x == "M", "Male"),
                 condition(x == "F", "Female"),
                 condition(TRUE, "Other"))
```

```
# Apply format to data
sdtm.DM$SEXF <- fapply(sdtm.DM$SEX, fmt_sex)

# View a few rows of data
print(sdtm.DM[1:5, c("USUBJID", "SEX", "SEXF")])
# # A tibble: 5 x 3
#   USUBJID    SEX   SEXF
#   <chr>      <chr> <chr>
# 1 ABC-01-049 M     Male
# 2 ABC-01-050 M     Male
# 3 ABC-01-051 M     Male
# 4 ABC-01-052 F     Female
# 5 ABC-01-053 F     Female

# Unload library from workspace
lib_unload(sdtm)
```

The `value()` and `fapply()` functions are a small part of the very capable **fmtr** package. See the fmtr documentation for a complete list of functions.

PERFORM A DATASTEP

In SAS®, the data step is the go-to procedure for manipulating data. The data step provides row-by-row processing of a dataset. In R, however, data processing is typically done column-by-column. This change in orientation is difficult for SAS® programmers to adjust to. Also, there are some types of processing that are easy to do in a data step, but are quite difficult to do with column-wise processing.

For these reasons, the **sassy** system offers a `datastep()` function. It is part of the **libr** package. The `datastep()` function simulates the most basic functionality of a SAS® data step. Like a SAS® data step, it allows you to reference data.frame variables unquoted, and to make up variables on the fly. It also provides basic data shaping parameters like keep, drop, rename, and retain. The example below demonstrates a simple data step that performs age categorization on the sdtm.DM dataset.

```
library(sassy)

# Get sample data directory
dir <- system.file("extdata", package = "sassy")

libname(sdtm, dir, "csv")

# Loads data into workspace
lib_load(sdtm)

# Perform data step
dm_mod <- datastep(sdtm.DM, keep = c("USUBJID", "AGE", "AGECAT"), {
    if (AGE >= 18 & AGE <= 24)
          AGECAT <- "18 to 24"
    else if (AGE >= 25 & AGE <= 44)
          AGECAT <- "25 to 44"
    else if (AGE >= 45 & AGE <= 64)
          AGECAT <- "45 to 64"
    else if (AGE >= 65)
          AGECAT <- ">= 65"
    })

# Print dm_mod to console
print(dm_mod)
```

```
# # A tibble: 87 x 3
#    USUBJID      AGE AGECAT
#    <chr>      <dbl> <chr>
#  1 ABC-01-049    39 25 to 44
#  2 ABC-01-050    47 45 to 64
#  3 ABC-01-051    34 25 to 44
#  4 ABC-01-052    45 45 to 64
#  5 ABC-01-053    26 25 to 44
#  6 ABC-01-054    44 25 to 44
#  7 ABC-01-055    47 45 to 64
#  8 ABC-01-056    31 25 to 44
#  9 ABC-01-113    74 >= 65
# 10 ABC-01-114    72 >= 65
# # ... with 77 more rows

# Unload library from workspace
lib_unload(sdtm)
```

The `datastep()` function, like the `libname()` function, is one of several powerful functions in the **libr** package. See the libr documentation for additional examples and extended discussion.

WRITE A REPORT

If you want to create an HTML report in R, there are many packages to choose from. For paged file formats such as text, rich-text, and PDF, however, options are much more limited. The reporter package provides easy page-based reporting with a choice of output types. It is currently the closest R package to matching the capabilities of SAS `proc report` and `ODS`.

There are four steps to creating a **reporter** report:

- Create report content
- Create report
- Add content to report
- Write the report

Below is an example of a simple data listing using the **libr** and **reporter** packages. Observe that the reporter package will automatically set column widths, wrap pages, and put page breaks at the appropriate locations:

```
library(sassy)
library(magrittr)

# Get sample data directory
dir <- system.file("extdata", package = "sassy")

# Define data library
libname(sdtm, dir, "csv")

# Loads data into workspace
lib_load(sdtm)

# Create report content
tbl <- create_table(sdtm.DM)

# Create report and add content
rpt <- create_report("example.txt") %>%
    page_header("Sponsor: Company", "Study: ABC") %>%
```

```
    titles("Listing 1.0", "Sample Demographics Data") %>%
    add_content(tbl) %>%
    page_footer(Sys.time(), "CONFIDENTIAL", "Page [pg] of [tpg]")

# Write out the report
write_report(rpt)
# # A report specification: 9 pages
# - file_path: 'example.txt'
# - output_type: TXT
# - units: inches
# - orientation: landscape
# - line size/count: 108/45
# - page_header: left=Sponsor: Company right=Study: ABC
# - title 1: 'Listing 1.0'
# - title 2: 'Sample Demographics Data'
# - page_footer: left=2021-01-18 16:07:52 center=CONFIDENTIAL right=Page
[pg] of [tpg]
# - content:
#   # A table specification:
#   - data: tibble 'sdtm.DM' 87 rows 24 cols
# - show_cols: all
# - use_attributes: all
```

Here is the first page of the report created by the above example code:

```
Sponsor: Company                                                                                                Study: ABC
                                                    Listing 1.0
                                                Sample Demographics Data

STUDYID DOMAIN USUBJID    SUBJID    RFSTDTC    RFENDTC RFXSTDTC RFXENDTC    RFICDTC  RFPENDTC DTHDTC DTHFL
-------------------------------------------------------------------------------------------------------------
ABC     DM     ABC-01-049 049    2006-11-07                              2006-10-25
ABC     DM     ABC-01-050 050    2006-11-02                              2006-10-25
ABC     DM     ABC-01-051 051    2006-11-02                              2006-10-25
ABC     DM     ABC-01-052 052    2006-11-06                              2006-10-31
ABC     DM     ABC-01-053 053    2006-11-08                              2006-11-01
ABC     DM     ABC-01-054 054    2006-11-16                              2006-11-07
ABC     DM     ABC-01-055 055    2006-12-06                              2006-10-31
ABC     DM     ABC-01-056 056    2006-11-28                              2006-11-21
ABC     DM     ABC-01-113 113    2006-12-05                              2006-11-28
ABC     DM     ABC-01-114 114    2006-12-14                              2006-12-01
ABC     DM     ABC-02-033 033    2006-10-25                              2006-10-16
ABC     DM     ABC-02-034 034    2006-10-26                              2006-10-16
ABC     DM     ABC-02-035 035    2006-10-31                              2006-10-16
ABC     DM     ABC-02-036 036    2006-11-01                              2006-10-23
ABC     DM     ABC-02-037 037    2006-11-01                              2006-10-24
ABC     DM     ABC-02-038 038    2006-11-13                              2006-10-30
ABC     DM     ABC-02-039 039    2006-11-15                              2006-11-06
ABC     DM     ABC-02-040 040    2006-12-07                              2006-11-29
ABC     DM     ABC-02-109 109    2006-12-07                              2006-11-28
ABC     DM     ABC-02-110 110    2006-12-18                              2006-12-11
ABC     DM     ABC-02-111 111    2006-12-19                              2006-12-11
ABC     DM     ABC-02-112 112    2006-12-19                              2006-12-12
ABC     DM     ABC-03-001 001    2006-10-10                              2006-09-26
ABC     DM     ABC-03-002 002    2006-10-11                              2006-09-27
ABC     DM     ABC-03-003 003    2006-10-11                              2006-09-27
ABC     DM     ABC-03-004 004    2006-10-12                              2006-09-27
ABC     DM     ABC-03-005 005    2006-10-12                              2006-09-27
ABC     DM     ABC-03-006 006    2006-10-25                              2006-10-18
ABC     DM     ABC-03-007 007    2006-10-31                              2006-10-27
ABC     DM     ABC-03-008 008    2006-11-02 2006-11-09                   2006-10-18 2006-11-09
ABC     DM     ABC-03-089 089    2006-11-21                              2006-11-15
```

```
ABC     DM     ABC-03-090 090    2006-12-06                              2006-11-30
ABC     DM     ABC-03-091 091    2006-12-20                              2006-12-11
ABC     DM     ABC-04-073 073    2006-10-31                              2006-10-10
ABC     DM     ABC-04-074 074    2006-11-01 2006-11-20                   2006-10-16 2006-11-20
ABC     DM     ABC-04-075 075    2006-11-09                              2006-11-02
ABC     DM     ABC-04-076 076    2006-11-15                              2006-10-26

2021-01-18 16:07:52                          CONFIDENTIAL                                Page 1 of 9
```

See the reporter documentation site for more examples and complete function documentation.

INTEGRATED EXAMPLE

Now let's look at an integrated example. This example will demonstrate how the **sassy** functions work together, and allow you to write programs in a manner similar to how you would write them in SAS®.

In this example, we will use the above techniques to create a simple Demographics table. The example program will pull data from the sample "sdtm" data library, perform a data step, create a user-defined format, calculate summary statistics, and create a report in RTF format. All of these activities will be captured in a traceable log.

Note the following about this example:

- The `options("logr.autolog" = TRUE)` statement turns on the useful "autolog" feature of the **logr** package. This feature performs automatic logging for many functions in the **sassy** system.
- The `sep()` function from the **logr** package creates a distinctive section break in the log.
- The `fmt_*` functions are from the **fmtr** package, and include common formats for statistical reports.
- The **reporter** package contains many functions to easily control report layout.

CODE

Here is the code for the integrated example:

```
library(sassy)
library(tidyverse)


# Turn on autolog and turn off notes
options("logr.autolog" = TRUE,
        "logr.notes" = FALSE)

# Open log
log_open("example2.log")

# Get sample data directory
dir <- system.file("extdata", package = "sassy")


# Load and Prepare Data ---------------------------------------------

sep("Prepare Data")

# Define data library
```

```
libname(sdtm, dir, "csv")

# Loads data into workspace
lib_load(sdtm)

# Prepare data
dm_mod <- sdtm.DM %>%
  select(USUBJID, SEX, AGE, ARM) %>%
  filter(ARM != "SCREEN FAILURE") %>%
  datastep({

     if (AGE >= 18 & AGE <= 24)
       AGECAT <- "18 to 24"
     else if (AGE >= 25 & AGE <= 44)
       AGECAT <- "25 to 44"
     else if (AGE >= 45 & AGE <= 64)
       AGECAT <- "45 to 64"
     else if (AGE >= 65)
       AGECAT <- ">= 65"

   }) %>% put()

put("Get ARM population counts")
arm_pop <- count(dm_mod, ARM) %>% deframe() %>% put()


# Age Summary Block ------------------------------------------------

sep("Create summary statistics for age")

age_block <-
  dm_mod %>%
  group_by(ARM) %>%
  summarise( N = fmt_n(AGE),
             `Mean (SD)` = fmt_mean_sd(AGE),
             Median = fmt_median(AGE),
             `Q1 - Q3` = fmt_quantile_range(AGE),
             Range  = fmt_range(AGE)) %>%
  pivot_longer(-ARM,
               names_to  = "label",
               values_to = "value") %>%
  pivot_wider(names_from = ARM,
              values_from = "value") %>%
  add_column(var = "AGE", .before = "label") %>%
  put()


# Age Group Block --------------------------------------------------------

sep("Create frequency counts for Age Group")


put("Create age group frequency counts")
ageg_block <-
  dm_mod %>%
  select(ARM, AGECAT) %>%
  group_by(ARM, AGECAT) %>%
```

```
  summarize(n = n()) %>%
  pivot_wider(names_from = ARM,
              values_from = n,
              values_fill = 0) %>%
  transmute(var = "AGECAT",
            label =  factor(AGECAT, levels = c("18 to 24",
                                               "25 to 44",
                                               "45 to 64",
                                               ">= 65")),
            `ARM A` = fmt_cnt_pct(`ARM A`, arm_pop["ARM A"]),
            `ARM B` = fmt_cnt_pct(`ARM B`, arm_pop["ARM B"]),
            `ARM C` = fmt_cnt_pct(`ARM C`, arm_pop["ARM C"]),
            `ARM D` = fmt_cnt_pct(`ARM D`, arm_pop["ARM D"])) %>%
  arrange(label) %>%
  put()


# Sex Block ------------------------------------------------------------

sep("Create frequency counts for SEX")

# Create user-defined format
fmt_sex <- value(condition(is.na(x), "Missing"),
                 condition(x == "M", "Male"),
                 condition(x == "F", "Female"),
                 condition(TRUE, "Other")) %>% put()

# Create sex frequency counts
sex_block <-
  dm_mod %>%
  select(ARM, SEX) %>%
  group_by(ARM, SEX) %>%
  summarize(n = n()) %>%
  pivot_wider(names_from = ARM,
              values_from = n,
              values_fill = 0) %>%
  transmute(var = "SEX",
            label =   fct_relevel(SEX, "M", "F"),
            `ARM A` = fmt_cnt_pct(`ARM A`, arm_pop["ARM A"]),
            `ARM B` = fmt_cnt_pct(`ARM B`, arm_pop["ARM B"]),
            `ARM C` = fmt_cnt_pct(`ARM C`, arm_pop["ARM C"]),
            `ARM D` = fmt_cnt_pct(`ARM D`, arm_pop["ARM D"])) %>%
  arrange(label) %>%
  mutate(label = fapply(label, fmt_sex)) %>%
  put()

put("Combine blocks into final data frame")
final <- bind_rows(age_block, ageg_block, sex_block) %>% put()

# Report ---------------------------------------------------------------


sep("Create and print report")

var_fmt <- c("AGE" = "Age", "AGECAT" = "Age Group", "SEX" = "Sex")

# Create Table
```

```
tbl <- create_table(final, first_row_blank = TRUE) %>%
  column_defaults(from=`ARM A`, to=`ARM D`, align="center", width=1.25) %>%
  stub(vars = c("var", "label"), "Variable", width = 2.5) %>%
  define(var, blank_after = TRUE, dedupe = TRUE, label = "Variable",
         format = var_fmt,label_row = TRUE) %>%
  define(label, indent = .25, label = "Demographic Category") %>%
  define(`ARM A`,  n = arm_pop["ARM A"]) %>%
  define(`ARM B`,  n = arm_pop["ARM B"]) %>%
  define(`ARM C`,  n = arm_pop["ARM C"]) %>%
  define(`ARM D`,  n = arm_pop["ARM D"])

# Create report
rpt <- create_report("example2.rtf", output_type = "RTF") %>%
  set_margins(top = 1, bottom = 1) %>%
  page_header("Sponsor: Company", "Study: ABC") %>%
  titles("Table 1.0", "Analysis of Demographic Characteristics",
         "Safety Population") %>%
  add_content(tbl) %>%
  footnotes("Program: Table1_0.R",
        "NOTE: Denominator based on number of non-missing responses.") %>%
  page_footer(paste0("Date Produced: ",fapply(Sys.time(), "%d%b%y %H:%M")),
              right = "Page [pg] of [tpg]")

# Write out report
write_report(rpt)


# Unload library from workspace
lib_unload(sdtm)

# Close log
log_close()
```

LOG

Here is the log produced by the above code:

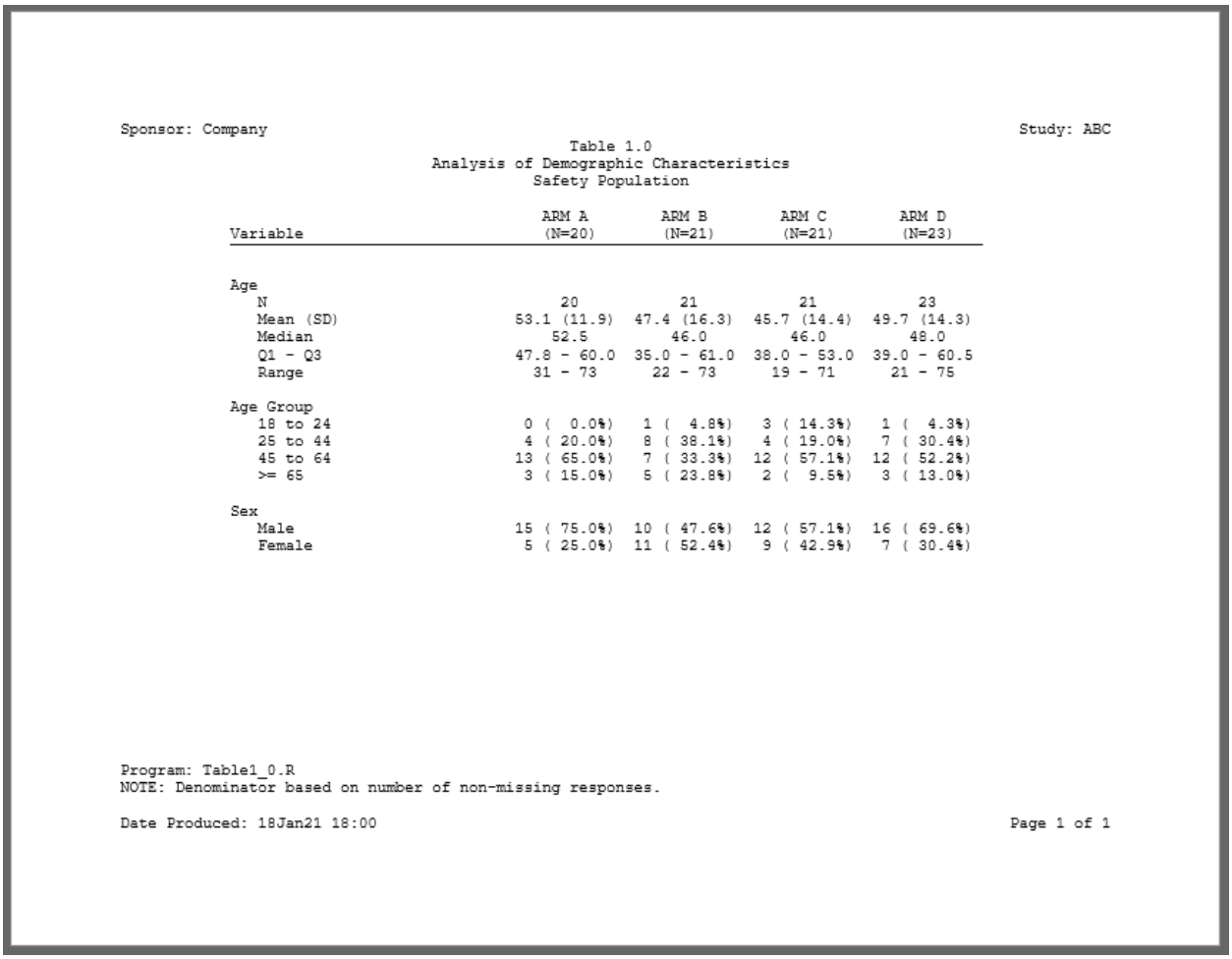
```
================================================================================
Log Path: ./log/example2.log
Working Directory: C:/packages/Testing
User Name: dbosa
R Version: 4.0.3 (2020-10-10)
Machine: DESKTOP-IGGQ6UV x86-64
Operating System: Windows 10 x64 build 19041
Log Start Time: 2021-01-18 18:10:18
================================================================================

================================================================================
Prepare Data
================================================================================

# library 'sdtm': 8 items
- attributes: csv not loaded
- path: C:/Users/dbosa/Documents/R/win-library/4.0/sassy/extdata
- items:
  Name Extension Rows Cols     Size        LastModified
1   AE       csv  150   27  88.1 Kb 2021-01-16 20:48:35
2   DA       csv 3587   18 527.8 Kb 2021-01-16 20:48:35
3   DM       csv   87   24  45.2 Kb 2021-01-16 20:48:35
4   DS       csv  174    9  33.7 Kb 2021-01-16 20:48:35
```

```
5   EX       csv   84   11    26 Kb 2021-01-16 20:48:35
6   IE       csv    2   14    13 Kb 2021-01-16 20:48:35
7   SV       csv  685   10  69.9 Kb 2021-01-16 20:48:35
8   VS       csv 3358   17   467 Kb 2021-01-16 20:48:35

lib_load: library 'sdtm' loaded

datastep: columns decreased from 4 to 5

# A tibble: 85 x 5
   USUBJID    SEX     AGE ARM   AGECAT
   <chr>      <chr> <dbl> <chr> <chr>
 1 ABC-01-049 M        39 ARM D 25 to 44
 2 ABC-01-050 M        47 ARM B 45 to 64
 3 ABC-01-051 M        34 ARM A 25 to 44
 4 ABC-01-052 F        45 ARM C 45 to 64
 5 ABC-01-053 F        26 ARM B 25 to 44
 6 ABC-01-054 M        44 ARM D 25 to 44
 7 ABC-01-055 F        47 ARM C 45 to 64
 8 ABC-01-056 M        31 ARM A 25 to 44
 9 ABC-01-113 M        74 ARM D >= 65
10 ABC-01-114 F        72 ARM B >= 65
# ... with 75 more rows

Get ARM population counts

ARM A ARM B ARM C ARM D
   20    21    21    23

================================================================================
Create summary statistics for age
================================================================================

# A tibble: 5 x 6
  var   label     `ARM A`     `ARM B`     `ARM C`     `ARM D`
  <chr> <chr>     <chr>       <chr>       <chr>       <chr>
1 AGE   N         20          21          21          23
2 AGE   Mean (SD) 53.1 (11.9) 47.4 (16.3) 45.7 (14.4) 49.7 (14.3)
3 AGE   Median    52.5        46.0        46.0        48.0
4 AGE   Q1 - Q3   47.8 - 60.0 35.0 - 61.0 38.0 - 53.0 39.0 - 60.5
5 AGE   Range     31 - 73     22 - 73     19 - 71     21 - 75

================================================================================
Create frequency counts for Age Group
================================================================================

Create age group frequency counts

# A tibble: 4 x 6
  var    label    `ARM A`     `ARM B`     `ARM C`     `ARM D`
  <chr>  <fct>    <chr>       <chr>       <chr>       <chr>
1 AGECAT 18 to 24 0 (  0.0%)  1 (  4.8%) 3 ( 14.3%)  1 (  4.3%)
2 AGECAT 25 to 44 4 ( 20.0%)  8 ( 38.1%) 4 ( 19.0%)  7 ( 30.4%)
3 AGECAT 45 to 64 13 ( 65.0%) 7 ( 33.3%) 12 ( 57.1%) 12 ( 52.2%)
4 AGECAT >= 65    3 ( 15.0%)  5 ( 23.8%) 2 (  9.5%)  3 ( 13.0%)

================================================================================
Create frequency counts for SEX
================================================================================

# A user-defined format: 4 conditions
  Name Type Expression   Label Order
1    x    U   is.na(x) Missing    NA
2    x    U   x == "M"    Male    NA
3    x    U   x == "F"  Female    NA
4    x    U       TRUE   Other    NA

# A tibble: 2 x 6
  var   label  `ARM A`     `ARM B`     `ARM C`     `ARM D`
```

```
  <chr> <chr>  <chr>       <chr>       <chr>       <chr>
1 SEX   Male   15 ( 75.0%) 10 ( 47.6%) 12 ( 57.1%) 16 ( 69.6%)
2 SEX   Female 5 ( 25.0%)  11 ( 52.4%) 9 ( 42.9%)  7 ( 30.4%)

Combine blocks into final data frame

# A tibble: 11 x 6
   var    label     `ARM A`      `ARM B`      `ARM C`      `ARM D`
   <chr>  <chr>     <chr>        <chr>        <chr>        <chr>
 1 AGE    N         20           21           21           23
 2 AGE    Mean (SD) 53.1 (11.9)  47.4 (16.3)  45.7 (14.4)  49.7 (14.3)
 3 AGE    Median    52.5         46.0         46.0         48.0
 4 AGE    Q1 - Q3   47.8 - 60.0  35.0 - 61.0  38.0 - 53.0  39.0 - 60.5
 5 AGE    Range     31 - 73      22 - 73      19 - 71      21 - 75
 6 AGECAT 18 to 24  0 (  0.0%)   1 (  4.8%)   3 ( 14.3%)   1 (  4.3%)
 7 AGECAT 25 to 44  4 ( 20.0%)   8 ( 38.1%)   4 ( 19.0%)   7 ( 30.4%)
 8 AGECAT 45 to 64  13 ( 65.0%)  7 ( 33.3%)   12 ( 57.1%)  12 ( 52.2%)
 9 AGECAT >= 65     3 ( 15.0%)   5 ( 23.8%)   2 (  9.5%)   3 ( 13.0%)
10 SEX    Male      15 ( 75.0%)  10 ( 47.6%)  12 ( 57.1%)  16 ( 69.6%)
11 SEX    Female    5 ( 25.0%)   11 ( 52.4%)  9 ( 42.9%)   7 ( 30.4%)

================================================================================
Create and print report
================================================================================

# A report specification: 1 pages
- file_path: 'example2.rtf'
- output_type: RTF
- units: inches
- orientation: landscape
- margins: top 1 bottom 1 left 1 right 1
- line size/count: 107/41
- page_header: left=Sponsor: Company right=Study: ABC
- title 1: 'Table 1.0'
- title 2: 'Analysis of Demographic Characteristics'
- title 3: 'Safety Population'
- footnote 1: 'Program: Table1_0.R'
- footnote 2: 'NOTE: Denominator based on number of non-missing responses.'
- page_footer: left=Date Produced: 18Jan21 18:10 center= right=Page [pg] of [tpg]
- content:
# A table specification:
- data: tibble 'final' 11 rows 6 cols
- show_cols: all
- use_attributes: all
- stub: var label 'Variable' width=2.5 align='left'
- define: var 'Variable' dedupe='TRUE'
- define: label 'Demographic Category'
- define: ARM A
- define: ARM B
- define: ARM C
- define: ARM D

lib_sync: synchronized data in library 'sdtm'

lib_unload: library 'sdtm' unloaded

================================================================================
Log End Time: 2021-01-18 18:10:30
Log Elapsed Time: 0 00:00:11
================================================================================
```

OUTPUT

And here is the output report produced by the integrated example:



Sponsor: Company Study: ABC

Table 1.0
Analysis of Demographic Characteristics
Safety Population

Variable	ARM A (N=20)	ARM B (N=21)	ARM C (N=21)	ARM D (N=23)
Age				
N	20	21	21	23
Mean (SD)	53.1 (11.9)	47.4 (16.3)	45.7 (14.4)	49.7 (14.3)
Median	52.5	46.0	46.0	48.0
Q1 - Q3	47.8 - 60.0	35.0 - 61.0	38.0 - 53.0	39.0 - 60.5
Range	31 - 73	22 - 73	19 - 71	21 - 75
Age Group				
18 to 24	0 (0.0%)	1 (4.8%)	3 (14.3%)	1 (4.3%)
25 to 44	4 (20.0%)	8 (38.1%)	4 (19.0%)	7 (30.4%)
45 to 64	13 (65.0%)	7 (33.3%)	12 (57.1%)	12 (52.2%)
>= 65	3 (15.0%)	5 (23.8%)	2 (9.5%)	3 (13.0%)
Sex				
Male	15 (75.0%)	10 (47.6%)	12 (57.1%)	16 (69.6%)
Female	5 (25.0%)	11 (52.4%)	9 (42.9%)	7 (30.4%)

Program: Table1_0.R
NOTE: Denominator based on number of non-missing responses.

Date Produced: 18Jan21 18:00 Page 1 of 1

GETTING HELP

If you need help with the **sassy** family of packages, the best place to turn to is the r-sassy web site. This web site offers many examples, and full documentation on every package and function.

If you need additional help, please consult stackoverflow.com. The stackoverflow community will be very willing to answer your questions.

If you want to look at the code for the **sassy** packages, visit the github page.

If you encounter a bug or have a feature request, please submit an issue here.

CONCLUSION

By providing a similar conceptual framework, the **sassy** system makes it much easier for a SAS® programmer to work in R. The package offers R versions of many familiar SAS® concepts. Those concepts include data libraries, data dictionaries, a data step, formats and format catalogs, and a traceable log. The system also provides a reporting package that is closer to the reporting capabilities

found in SAS®. By introducing these concepts, the **sassy** system can greatly increase the efficiency and satisfaction of writing programs in R.

CONTACT INFORMATION

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

David J. Bosak
dbosak01@gmail.com
www.r-sassy.org