Paper 185-2023

An Overview of the SASSY System

David J. Bosak, Archytas Clinical Solutions

ABSTRACT

This presentation will provide an overview for a set of R language packages called the **sassy** system. The **sassy** system is designed to make R easier for SAS® Programmers. The system includes tools to create a log, declare a libname, perform a datastep, define a format, generate a report, and more. The overall experience makes writing code in R more similar to writing code in SAS. If you are a SAS programmer who wants to be more comfortable and productive in R, then this presentation will be of interest to you. The presentation will be given by David J. Bosak, who is the author 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 six 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
- **procs**: Simulates several popular SAS procedures
- **common**: A collection of utility functions

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 **posit**®.

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.packages("sassy")
```

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

```
library(sassy)
```

The **sassy** packages are written almost entirely in Base R. They were written this way intentionally to minimize dependencies and make the system more stable. Note that there are no dependencies on **Tidyverse**®, **Quarto**®, or any other package from **posit**®.

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 simple logging system that anyone can use.

There are three steps to creating a log:

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

Below is a minimal 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:/Projects/Archytas/WUSS/Code
User Name: dbosa
R Version: 4.3.1 (2023-06-16 ucrt)
Machine: SOCRATES x86-64
Operating System: Windows 10 x64 build 22621
Base Packages: stats graphics grDevices utils datasets methods base Other
Packages: procs_1.0.3 reporter_1.4.2 libr_1.2.8 logr_1.3.4 fmtr_1.6.0
common_1.0.9 sassy_1.2.1
Log Start Time: 2023-10-01 00:20:10.31621
================================================================================

Here is something to send to the log.

NOTE: Log Print Time:  2023-10-01 00:20:10.491874
NOTE: Elapsed Time: 0.174062013626099 secs

================================================================================
Log End Time: 2023-10-01 00:20:11.291764
Log Elapsed Time: 0 00:00:00
================================================================================
```

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 from a specified directory, 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)

# Define library
libname(sdtm, "./data", "csv")
```

Once the datasets have been imported, the libname can be accessed using R list syntax. That means the datasets are now available from the libname using the dollar sign ($). In **RStudio**®, it looks like this:

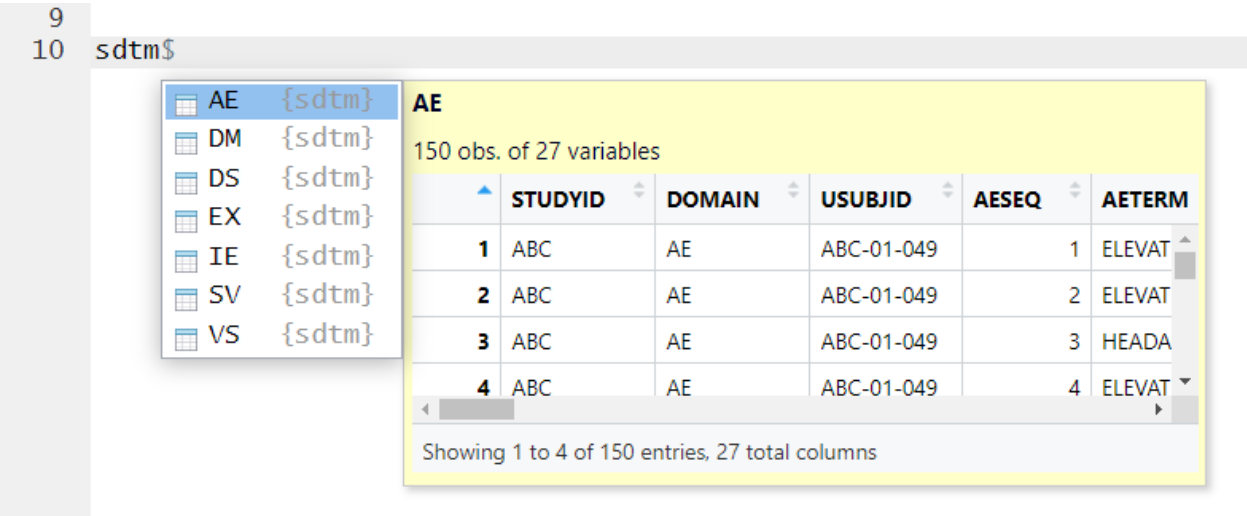


Figure 1. Accessing datasets in a libname

The **libr** package not only allows you to easily access a directory of datasets. It also allows you to add, remove, and edit datasets in the library. The `libname()` function can import SAS datasets, CSV files, R data files, Excel files, and more. 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)

#Create libname
libname(sdtm, "./data", "csv")

# Copy DM dataset
dat <- sdtm$DM
```

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

# Create a new column of formatted values
dat$SEXF <- fapply(dat$SEX, fmt_sex)

# View a few rows of data
print(dat[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
```

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)

#Create libname
libname(sdtm, "./data", "csv")


# 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
```

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, RTF, DOCX, 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`.

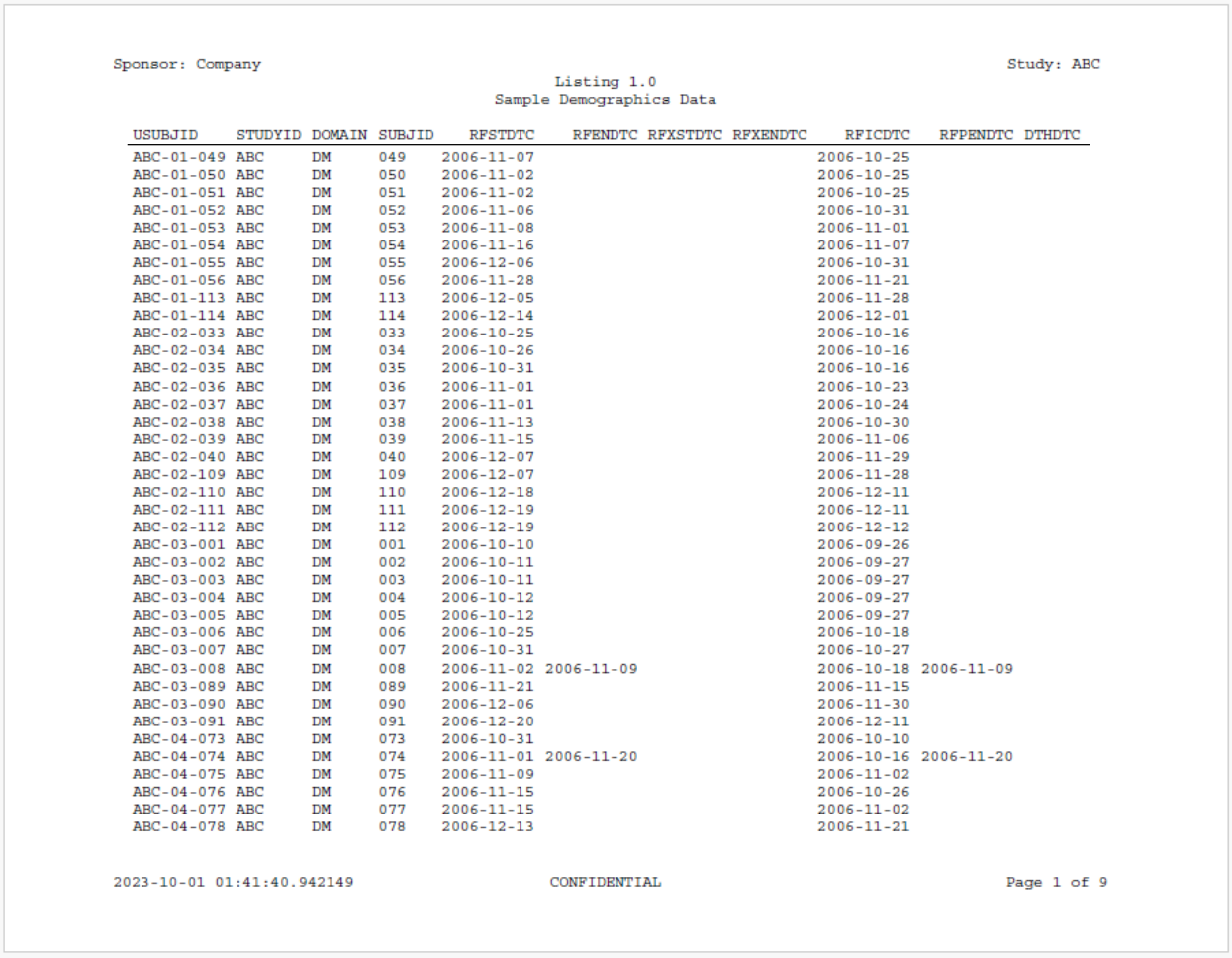
There are four steps to creating a **reporter** report:

- 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. If you want to override the automatic settings, you can use the `define()` function. Also note the ability to set an ID variable that is repeated on every page:

```
library(sassy)

# Define data library
libname(sdtm, "./data", "csv")

# Create report content
tbl <- create_table(sdtm$DM) |>
  define(RACE, width = 2.5) |>
  define(ETHNIC, width = 2.5) |>
  define(USUBJID, id_var = TRUE)

# Create report and add content
rpt <- create_report("./output/example.pdf", font = "Courier",
                     output_type = "PDF") |>
  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)
```

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



Sponsor: Company — Study: ABC

Listing 1.0
Sample Demographics Data

USUBJID	STUDYID	DOMAIN	SUBJID	RFSTDTC	RFENDTC	RFXSTDTC	RFXENDTC	RFICDTC	RFPENDTC	DTHDTC
ABC-01-049	ABC	DM	049	2006-11-07				2006-10-25		
ABC-01-050	ABC	DM	050	2006-11-02				2006-10-25		
ABC-01-051	ABC	DM	051	2006-11-02				2006-10-25		
ABC-01-052	ABC	DM	052	2006-11-06				2006-10-31		
ABC-01-053	ABC	DM	053	2006-11-08				2006-11-01		
ABC-01-054	ABC	DM	054	2006-11-16				2006-11-07		
ABC-01-055	ABC	DM	055	2006-12-06				2006-10-31		
ABC-01-056	ABC	DM	056	2006-11-28				2006-11-21		
ABC-01-113	ABC	DM	113	2006-12-05				2006-11-28		
ABC-01-114	ABC	DM	114	2006-12-14				2006-12-01		
ABC-02-033	ABC	DM	033	2006-10-25				2006-10-16		
ABC-02-034	ABC	DM	034	2006-10-26				2006-10-16		
ABC-02-035	ABC	DM	035	2006-10-31				2006-10-16		
ABC-02-036	ABC	DM	036	2006-11-01				2006-10-23		
ABC-02-037	ABC	DM	037	2006-11-01				2006-10-24		
ABC-02-038	ABC	DM	038	2006-11-13				2006-10-30		
ABC-02-039	ABC	DM	039	2006-11-15				2006-11-06		
ABC-02-040	ABC	DM	040	2006-12-07				2006-11-29		
ABC-02-109	ABC	DM	109	2006-12-07				2006-11-28		
ABC-02-110	ABC	DM	110	2006-12-18				2006-12-11		
ABC-02-111	ABC	DM	111	2006-12-19				2006-12-11		
ABC-02-112	ABC	DM	112	2006-12-19				2006-12-12		
ABC-03-001	ABC	DM	001	2006-10-10				2006-09-26		
ABC-03-002	ABC	DM	002	2006-10-11				2006-09-27		
ABC-03-003	ABC	DM	003	2006-10-11				2006-09-27		
ABC-03-004	ABC	DM	004	2006-10-12				2006-09-27		
ABC-03-005	ABC	DM	005	2006-10-12				2006-09-27		
ABC-03-006	ABC	DM	006	2006-10-25				2006-10-18		
ABC-03-007	ABC	DM	007	2006-10-31				2006-10-27		
ABC-03-008	ABC	DM	008	2006-11-02	2006-11-09			2006-10-18	2006-11-09	
ABC-03-089	ABC	DM	089	2006-11-21				2006-11-15		
ABC-03-090	ABC	DM	090	2006-12-06				2006-11-30		
ABC-03-091	ABC	DM	091	2006-12-20				2006-12-11		
ABC-04-073	ABC	DM	073	2006-10-31				2006-10-10		
ABC-04-074	ABC	DM	074	2006-11-01	2006-11-20			2006-10-16	2006-11-20	
ABC-04-075	ABC	DM	075	2006-11-09				2006-11-02		
ABC-04-076	ABC	DM	076	2006-11-15				2006-10-26		
ABC-04-077	ABC	DM	077	2006-11-15				2006-11-02		
ABC-04-078	ABC	DM	078	2006-12-13				2006-11-21		

2023-10-01 01:41:40.942149 — CONFIDENTIAL — Page 1 of 9

Figure 2. A basic listing

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

GENERATE FREQUENCIES AND SUMMARY STATISTICS

One of the most frustrating things about working with R is that the statistics don't always match SAS. It is not that the R statistics are wrong. It is that R functions often have different default settings or different algorithms, which lead to different results. Both tools are correct. But they are correct in different ways.

The **procs** package was developed to give you an easy way to produce statistics in R that match SAS. The two primary functions of the package are `proc_freq()` and `proc_means()`. For convenience, the package also includes `proc_sort()`, `proc_transpose()`, and `proc_print()`. Collectively, these functions represent the core of the **sassy** system.

Let us look at an example. In this example, we will generate frequencies and means on some variables in the DM dataset, combine them into a single list, and print everything to the viewer:

```
library(sassy)

# Define data library
libname(sdtm, "./data", "csv")

# Generate Frequencies
res <- proc_freq(sdtm$DM,
```

```
                    tables = c("ARM", "SEX", "SEX * ARM"),
                    output = "report")

# Generate Means and append to Frequencies
res[["AGE"]] <- proc_means(dat, var = "AGE",
                           class = "ARM",
                           output = "report")

# Print everything to viewer
proc_print(res, titles = "Analysis of Demographics Dataset")
```

When working in RStudio, the above code will send the following to the viewer:

Analysis of Demographics Dataset

Table of ARM

ARM	N	Frequency	Percent	Cumulative Frequency	Cumulative Percent
ARM A	87	20	22.99	20	22.99
ARM B	87	21	24.14	41	47.13
ARM C	87	21	24.14	62	71.26
ARM D	87	23	26.44	85	97.70
SCREEN FAILURE	87	2	2.30	87	100.00

Table of SEX

SEX	N	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	87	32	36.78	32	36.78
M	87	55	63.22	87	100.00

Table of SEX by ARM

		ARM					
SEX	Statistic	ARM A	ARM B	ARM C	ARM D	SCREEN FAILURE	Total
F	Frequency	5	11	9	7	0	32
	Percent	5.75	12.64	10.34	8.05	0.00	36.78
	Row Pct	15.62	34.38	28.12	21.88	0.00	
	Col Pct	25.00	52.38	42.86	30.43	0.00	
M	Frequency	15	10	12	16	2	55
	Percent	17.24	11.49	13.79	18.39	2.30	63.22
	Row Pct	27.27	18.18	21.82	29.09	3.64	
	Col Pct	75.00	47.62	57.14	69.57	100.00	
Total	Frequency	20	21	21	23	2	87
	Percent	22.99	24.14	24.14	26.44	2.30	100.00

ARM	Variable	N	Mean	Std Dev	Minimum	Maximum
ARM A	AGE	20	53.1500000	11.8999115	31.0000000	73.0000000
ARM B	AGE	21	47.3809524	16.2587705	22.0000000	73.0000000
ARM C	AGE	21	45.7142857	14.4192332	19.0000000	71.0000000
ARM D	AGE	23	49.7391304	14.3248588	21.0000000	75.0000000
SCREEN FAILURE	AGE	2	69.0000000	15.5563492	58.0000000	80.0000000

Figure 3. Procs viewer output

Note that you may also print the report to a file. Use the `file_path` and `output_type` parameters to specify where to print the report and what type of report to create. Available types are TXT, HTML, PDF, DOCX and RTF. See the procs package documentation for additional information.

USEFUL UTILITIES

The above packages comprise the main body of the **sassy** system. In addition, there is a package of utility functions called **common**. The functions in this package perform useful tasks that are not easily done using Base R. Here is a list of functions in the package and a short explanation of each:

- **v()**: A generalized NSE quoting function.
- **sort.data.frame()**: An overload to the sort function for data frames.
- **labels.data.frame()**: An overload to the labels function for data frames.
- **%p%**: An infix operator for the `paste0()` function.
- **%eq%**: An enhanced equality operator.
- **Sys.path()**: A function to return the path of the currently running program.
- **roundup()**: A rounding function that matches SAS® rounding.
- **file.find()**: A function to search for a file on the file system.
- **dir.find()**: A function to search for a directory on the file system.
- **find.names()**: A function to search for variable names on a data.frame.
- **copy.attributes()**: A function to copy column attributes from one data frame to another.
- **supsc()**: A function to get UTF-8 superscript characters.
- **subsc()**: A function to get UTF-8 subscript characters.
- **symbol()**: A function to lookup UTF-8 symbols.
- **spaces()**: A function to create a string of blank spaces.
- **changed()**: A function to identify changed values in a vector or data frame.

For more information, see the common package documentation.

VALIDATION

The **sassy** system is validated. For the **procs** package, the `proc_freq()` and `proc_means()` functions have been compared to SAS. The validation documentation is here.

In addition, there are IQ and OQ validation routines built into the **sassy** package itself. These routines can be used for on-site qualification of the packages in your local environment. See the `run_iq()` and `run_oq()` documentation in the **sassy** package.

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 **reporter** package has capabilities to easily control report layout.
- The **procs** package generates frequency and summary statistics in almost the same manner as SAS.

CODE

Here is the code for the integrated example:

```
library(sassy)

# 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")

# Prepare data
dm_mod <- sdtm$DM |>
  datastep(keep = v(USUBJID, ARM, SEX, AGE, AGECAT),
           where = expression(ARM != "SCREEN FAILURE"), {

    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("Get ARM population counts")
arm_pop <- proc_freq(dm_mod, tables = ARM,
                     options = v(nocum, nopercent, nonobs),
                     output = long)

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

sep("Create summary statistics for age")

put("Generate summary stats")
proc_means(dm_mod, var = AGE, class = ARM,
           stats = v(n, mean, std, median, Q1, Q3, min, max),
           options = v(notype, nofreq, nway)) |>
  datastep(drop = v(MEAN, STD, Q1, Q3, MIN, MAX), {

    MEANSD <- fapply2(MEAN, STD, "%4.1f", "(%5.2f)")
    QRANGE <- fapply2(Q1, Q3, "%3.1f", "- %3.1f")
    MINMAX <- fapply2(MIN, MAX, "%d", "- %d")
  }) |>
```

```
  proc_transpose(var = v(N, MEANSD, MEDIAN, QRANGE, MINMAX),
                 id = CLASS, copy = VAR, name = "LABEL") -> age_block

put("Format stat labels")
age_block$LABEL <- fapply(age_block$LABEL, c(N = "n",
                                             MEANSD = "Mean (SD)",
                                             MEDIAN = "Median",
                                             QRANGE = "Q1 - Q3",
                                             MINMAX = "Min - Max"))

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

sep("Create frequency counts for Age Group")

put("Generate frequencies")
proc_freq(dm_mod, tables = "AGECAT * ARM", options = v(nonobs)) |>
  datastep(rename = list(VAR1 = "VAR", CAT1 = "LABEL"),
           drop = v(VAR2, CNT, PCT), {

              CNTPCT <- fapply2(CNT, PCT, "%2d", "(%5.1f%%)")

           }) |>
  proc_transpose(var = v(CNTPCT), id = CAT2,
                 copy = VAR, by = LABEL)  |>
  datastep(drop = NAME, {}) -> ageg_block

put("Assign factor for sorting")
ageg_block$LABEL <- factor(ageg_block$LABEL,
                           levels = c("18 to 24", "25 to 44",
                                      "45 to 64", ">= 65"))

put("Sort by age group")
ageg_block <- proc_sort(ageg_block, by = LABEL)

put("Convert label to character so it can be bound")
ageg_block$LABEL <- as.character(ageg_block$LABEL)

put("Combine blocks into final dataset")
final <- rbind(age_block, ageg_block) |> put()

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


sep("Create and print report")

# Define variable format
var_fmt <- c("AGE" = "Age", "AGECAT" = "Age Group")

put("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"], label = "Placebo") |>
  define(`ARM B`,  n = arm_pop["ARM B"], label = "Treatment 1") |>
```

```
    define(`ARM C`,  n = arm_pop["ARM C"], label = "Treatment 2") |>
    define(`ARM D`,  n = arm_pop["ARM D"], label = "Treatment 3")

  put("Create report")
  rpt <- create_report("./output/example2.rtf", output_type = "RTF",
                       font = "Arial") |>
    set_margins(top = 1, bottom = 1) |>
    page_header("Sponsor: Company", "Study: ABC") |>
    titles("Table 1.0", "Analysis of Demographic Characteristics",
           "Safety Population", bold = TRUE) |>
    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]")

  put("Write out report")
  write_report(rpt)

  # Close log
  log_close()
```

LOG

Here is the log produced by the above program:

```
================================================================================
Log Path: ./log/Example2.log
Program Path: C:/Projects/Archytas/WUSS/Code/integrated.R
Working Directory: C:/Projects/Archytas/WUSS/Code
User Name: dbosa
R Version: 4.3.1 (2023-06-16 ucrt)
Machine: SOCRATES x86-64
Operating System: Windows 10 x64 build 22621
Base Packages: stats graphics grDevices utils datasets methods base Other
Packages: tidylog_1.0.2 procs_1.0.3 reporter_1.4.2 libr_1.2.8 logr_1.3.4
fmtr_1.6.0 common_1.0.9 sassy_1.2.1
Log Start Time: 2023-10-02 00:34:29.851869
================================================================================

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

# library 'sdtm': 7 items
- attributes: csv not loaded
- path: C:/Users/dbosa/AppData/Local/R/win-library/4.3/sassy/extdata
- items:
  Name Extension Rows Cols     Size        LastModified
1   AE       csv  150   27  88.5 Kb 2023-09-30 23:58:49
2   DM       csv   87   24  45.5 Kb 2023-09-30 23:58:49
3   DS       csv  174    9  34.1 Kb 2023-09-30 23:58:49
4   EX       csv   84   11  26.4 Kb 2023-09-30 23:58:49
5   IE       csv    2   14  13.4 Kb 2023-09-30 23:58:49
6   SV       csv  685   10  70.3 Kb 2023-09-30 23:58:49
7   VS       csv 3358   17 467.4 Kb 2023-09-30 23:58:49

datastep: columns decreased from 24 to 5

# A tibble: 85 × 5
   USUBJID    ARM   SEX     AGE AGECAT
```

```
   <chr>      <chr> <chr> <dbl> <chr>
 1 ABC-01-049 ARM D M        39 25 to 44
 2 ABC-01-050 ARM B M        47 45 to 64
 3 ABC-01-051 ARM A M        34 25 to 44
 4 ABC-01-052 ARM C F        45 45 to 64
 5 ABC-01-053 ARM B F        26 25 to 44
 6 ABC-01-054 ARM D M        44 25 to 44
 7 ABC-01-055 ARM C F        47 45 to 64
 8 ABC-01-056 ARM A M        31 25 to 44
 9 ABC-01-113 ARM D M        74 >= 65
10 ABC-01-114 ARM B F        72 >= 65
# ℹ 75 more rows
# ℹ Use `print(n = ...)` to see more rows

Get ARM population counts

proc_freq: input data set 85 rows and 5 columns
           tables: ARM
           output: long
           view: TRUE
           output: 1 datasets

# A tibble: 1 × 6
  VAR   STAT  `ARM A` `ARM B` `ARM C` `ARM D`
  <chr> <chr>   <dbl>   <dbl>   <dbl>   <dbl>
1 ARM   CNT        20      21      21      23

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

Generate summary stats

proc_means: input data set 85 rows and 5 columns
            class: ARM
            var: AGE
            stats: n mean std median Q1 Q3 min max
            view: TRUE
            output: 1 datasets

  CLASS VAR  N     MEAN      STD MEDIAN   Q1 Q3 MIN MAX
1 ARM A AGE 20 53.15000 11.89991   52.5 47.5 60  31  73
2 ARM B AGE 21 47.38095 16.25877   46.0 35.0 61  22  73
3 ARM C AGE 21 45.71429 14.41923   46.0 38.0 53  19  71
4 ARM D AGE 23 49.73913 14.32486   48.0 39.0 62  21  75

datastep: columns decreased from 10 to 7

  CLASS VAR  N MEDIAN       MEANSD      QRANGE  MINMAX
1 ARM A AGE 20   52.5 53.1 (11.90) 47.5 - 60.0 31 - 73
2 ARM B AGE 21   46.0 47.4 (16.26) 35.0 - 61.0 22 - 73
3 ARM C AGE 21   46.0 45.7 (14.42) 38.0 - 53.0 19 - 71
4 ARM D AGE 23   48.0 49.7 (14.32) 39.0 - 62.0 21 - 75

proc_transpose: input data set 4 rows and 7 columns
                var: N MEANSD MEDIAN QRANGE MINMAX
                id: CLASS
                copy: VAR
                name: LABEL
                output dataset 5 rows and 6 columns

  VAR  LABEL        ARM A        ARM B        ARM C        ARM D
1 AGE      N           20           21           21           23
2 AGE MEANSD 53.1 (11.90) 47.4 (16.26) 45.7 (14.42) 49.7 (14.32)
3 AGE MEDIAN         52.5         46.0         46.0         48.0
4 AGE QRANGE  47.5 - 60.0  35.0 - 61.0  38.0 - 53.0  39.0 - 62.0
5 AGE MINMAX      31 - 73      22 - 73      19 - 71      21 - 75

Format stat labels
```

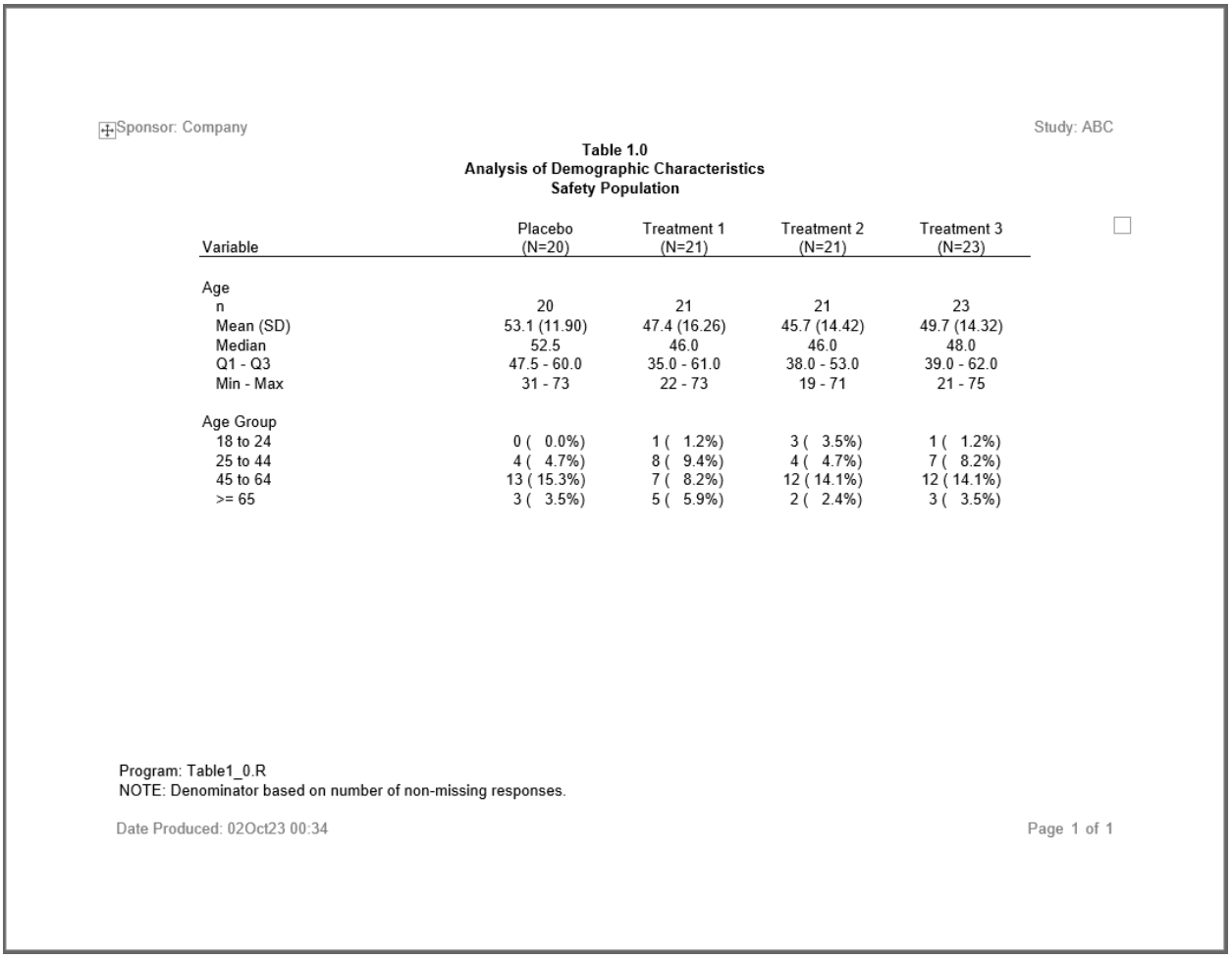
```
==========================================================================
Create frequency counts for Age Group
==========================================================================

Generate frequencies

proc_freq: input data set 85 rows and 5 columns
           tables: AGECAT * ARM
           view: TRUE
           output: 1 datasets

# A tibble: 16 × 6
   VAR1   VAR2  CAT1     CAT2    CNT   PCT
   <chr>  <chr> <chr>    <chr> <dbl> <dbl>
 1 AGECAT ARM   >= 65    ARM A     3  3.53
 2 AGECAT ARM   >= 65    ARM B     5  5.88
 3 AGECAT ARM   >= 65    ARM C     2  2.35
 4 AGECAT ARM   >= 65    ARM D     3  3.53
 5 AGECAT ARM   18 to 24 ARM A     0  0
 6 AGECAT ARM   18 to 24 ARM B     1  1.18
 7 AGECAT ARM   18 to 24 ARM C     3  3.53
 8 AGECAT ARM   18 to 24 ARM D     1  1.18
 9 AGECAT ARM   25 to 44 ARM A     4  4.71
10 AGECAT ARM   25 to 44 ARM B     8  9.41
11 AGECAT ARM   25 to 44 ARM C     4  4.71
12 AGECAT ARM   25 to 44 ARM D     7  8.24
13 AGECAT ARM   45 to 64 ARM A    13 15.3
14 AGECAT ARM   45 to 64 ARM B     7  8.24
15 AGECAT ARM   45 to 64 ARM C    12 14.1
16 AGECAT ARM   45 to 64 ARM D    12 14.1

datastep: columns decreased from 6 to 4

# A tibble: 16 × 4
   VAR    LABEL    CAT2  CNTPCT
   <chr>  <chr>    <chr> <chr>
 1 AGECAT >= 65    ARM A " 3 (  3.5%)"
 2 AGECAT >= 65    ARM B " 5 (  5.9%)"
 3 AGECAT >= 65    ARM C " 2 (  2.4%)"
 4 AGECAT >= 65    ARM D " 3 (  3.5%)"
 5 AGECAT 18 to 24 ARM A " 0 (  0.0%)"
 6 AGECAT 18 to 24 ARM B " 1 (  1.2%)"
 7 AGECAT 18 to 24 ARM C " 3 (  3.5%)"
 8 AGECAT 18 to 24 ARM D " 1 (  1.2%)"
 9 AGECAT 25 to 44 ARM A " 4 (  4.7%)"
10 AGECAT 25 to 44 ARM B " 8 (  9.4%)"
11 AGECAT 25 to 44 ARM C " 4 (  4.7%)"
12 AGECAT 25 to 44 ARM D " 7 (  8.2%)"
13 AGECAT 45 to 64 ARM A "13 ( 15.3%)"
14 AGECAT 45 to 64 ARM B " 7 (  8.2%)"
15 AGECAT 45 to 64 ARM C "12 ( 14.1%)"
16 AGECAT 45 to 64 ARM D "12 ( 14.1%)"

proc_transpose: input data set 16 rows and 4 columns
                by: LABEL
                var: CNTPCT
                id: CAT2
                copy: VAR
                name: NAME
                output dataset 4 rows and 7 columns

# A tibble: 4 × 7
  VAR    LABEL    NAME   `ARM A`       `ARM B`       `ARM C`       `ARM D`
  <chr>  <chr>    <chr>  <chr>         <chr>         <chr>         <chr>
1 AGECAT >= 65    CNTPCT " 3 (  3.5%)" " 5 (  5.9%)" " 2 (  2.4%)" " 3 (  3.5%)"
2 AGECAT 18 to 24 CNTPCT " 0 (  0.0%)" " 1 (  1.2%)" " 3 (  3.5%)" " 1 (  1.2%)"
3 AGECAT 25 to 44 CNTPCT " 4 (  4.7%)" " 8 (  9.4%)" " 4 (  4.7%)" " 7 (  8.2%)"
4 AGECAT 45 to 64 CNTPCT "13 ( 15.3%)" " 7 (  8.2%)" "12 ( 14.1%)" "12 ( 14.1%)"
```

```
datastep: columns decreased from 7 to 6

# A tibble: 4 × 6
  VAR    LABEL    `ARM A`       `ARM B`       `ARM C`       `ARM D`
  <chr>  <chr>    <chr>         <chr>         <chr>         <chr>
1 AGECAT >= 65    " 3 (  3.5%)" " 5 (  5.9%)" " 2 (  2.4%)" " 3 (  3.5%)"
2 AGECAT 18 to 24 " 0 (  0.0%)" " 1 (  1.2%)" " 3 (  3.5%)" " 1 (  1.2%)"
3 AGECAT 25 to 44 " 4 (  4.7%)" " 8 (  9.4%)" " 4 (  4.7%)" " 7 (  8.2%)"
4 AGECAT 45 to 64 "13 ( 15.3%)" " 7 (  8.2%)" "12 ( 14.1%)" "12 ( 14.1%)"

Assign factor for sorting

Sort by age group

proc_sort: input data set 4 rows and 6 columns
           by: LABEL
           keep: VAR LABEL ARM A ARM B ARM C ARM D
           order: a
           output data set 4 rows and 6 columns

# A tibble: 4 × 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 (  1.2%)" " 3 (  3.5%)" " 1 (  1.2%)"
2 AGECAT 25 to 44 " 4 (  4.7%)" " 8 (  9.4%)" " 4 (  4.7%)" " 7 (  8.2%)"
3 AGECAT 45 to 64 "13 ( 15.3%)" " 7 (  8.2%)" "12 ( 14.1%)" "12 ( 14.1%)"
4 AGECAT >= 65    " 3 (  3.5%)" " 5 (  5.9%)" " 2 (  2.4%)" " 3 (  3.5%)"

Convert label to character so it can be bound

Combine blocks into final dataset

     VAR     LABEL        ARM A        ARM B        ARM C        ARM D
1    AGE         n           20           21           21           23
2    AGE Mean (SD) 53.1 (11.90) 47.4 (16.26) 45.7 (14.42) 49.7 (14.32)
3    AGE    Median         52.5         46.0         46.0         48.0
4    AGE   Q1 - Q3  47.5 - 60.0  35.0 - 61.0  38.0 - 53.0  39.0 - 62.0
5    AGE Min - Max      31 - 73      22 - 73      19 - 71      21 - 75
6 AGECAT  18 to 24   0 (  0.0%)   1 (  1.2%)   3 (  3.5%)   1 (  1.2%)
7 AGECAT  25 to 44   4 (  4.7%)   8 (  9.4%)   4 (  4.7%)   7 (  8.2%)
8 AGECAT  45 to 64  13 ( 15.3%)   7 (  8.2%)  12 ( 14.1%)  12 ( 14.1%)
9 AGECAT     >= 65   3 (  3.5%)   5 (  5.9%)   2 (  2.4%)   3 (  3.5%)

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

Create Table

Create report

Write out report

# A report specification: 1 pages
- file_path: './output/example2.rtf'
- output_type: RTF
- units: inches
- orientation: landscape
- margins: top 1 bottom 1 left 1 right 1
- line size/count: 9/36
- 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: 02Oct23 00:34 center= right=Page [pg] of [tpg]
- content:
```

```
# A table specification:
- data: data.frame 'final' 9 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 'Placebo'
- define: ARM B 'Treatment 1'
- define: ARM C 'Treatment 2'
- define: ARM D 'Treatment 3'

================================================================================
Log End Time: 2023-10-02 00:34:31.940407
Log Elapsed Time: 0 00:00:02
================================================================================
```

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	Placebo (N=20)	Treatment 1 (N=21)	Treatment 2 (N=21)	Treatment 3 (N=23)
Age				
n	20	21	21	23
Mean (SD)	53.1 (11.90)	47.4 (16.26)	45.7 (14.42)	49.7 (14.32)
Median	52.5	46.0	46.0	48.0
Q1 - Q3	47.5 - 60.0	35.0 - 61.0	38.0 - 53.0	39.0 - 62.0
Min - Max	31 - 73	22 - 73	19 - 71	21 - 75
Age Group				
18 to 24	0 (0.0%)	1 (1.2%)	3 (3.5%)	1 (1.2%)
25 to 44	4 (4.7%)	8 (9.4%)	4 (4.7%)	7 (8.2%)
45 to 64	13 (15.3%)	7 (8.2%)	12 (14.1%)	12 (14.1%)
>= 65	3 (3.5%)	5 (5.9%)	2 (2.4%)	3 (3.5%)

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

Date Produced: 02Oct23 00:34 Page 1 of 1

Figure 4. Demographics table integrated example

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 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 system offers R versions of many familiar SAS concepts. Those concepts include data libraries, data dictionaries, a data step, formats and format catalogs, procedure replicas, 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.

REFERENCES

Bosak, David J. “The SASSY System” https://www.r-sassy.org, 2023.

ACKNOWLEDGMENTS

During development of this system, several people have provided ideas and encouragement. I’d like to thank the following people for their contributions: Duong Tran, Raphael Huang, Brian Varney, Kevin Kramer, and Yifei Chen.

CONTACT INFORMATION

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

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