PharmaSUG 2024 - Paper AP- 295

Replicating SAS® Procedures in R with the PROCS Package

David J. Bosak, Archytas Clinical Solutions

ABSTRACT

The **procs** package aims to simulate some commonly used SAS® procedures in R. The purpose of simulating SAS procedures is to make R easier to use and match statistical results. Another important motivation is to provide stable tools to work with in the pharmaceutical industry. The package replicates several of the most frequently used procedures, such as PROC FREQ, PROC MEANS, PROC TTEST, and PROC REG. The package also contains some data manipulation procedures like PROC TRANSPOSE and PROC SORT. This paper will present an overview of the package and provide demonstrations for each function.

INTRODUCTION

The **procs** R package was written to provide a stable, familiar set of functions for SAS programmers looking to get something done in R. The package has been validated, and the statistics match SAS. The package is also published on CRAN. Although the package contains only a small subset of SAS procedures, and even a subset of the functionality of those procedures, the package provides a minimally viable set of tools for basic analysis and reporting in R. The package can help improve your R programming and speed up development.

The **procs** package contains the following functions. Each function replicates the basic features of the corresponding procedure in SAS:

- `proc_freq()`
- `proc_means()`
- `proc_ttest()`
- `proc_reg()`
- `proc_transpose()`
- `proc_sort()`
- `proc_print()`

The proceeding sections will describe these functions, and provide some simple examples. For additional examples, see the **procs** web site at https://procs.r-sassy.org.

SAMPLE DATA

The examples below will use the following sample data. This data is identical to the *sashelp.class* dataset:

```
cls <- read.table(header = TRUE, text = '
Name    Sex Age Height Weight
Alfred    M  14    69.0  112.5
Alice     F  13    56.5   84.0
Barbara   F  13    65.3   98.0
Carol     F  14    62.8  102.5
Henry     M  14    63.5  102.5
James     M  12    57.3   83.0
Jane      F  12    59.8   84.5
Janet     F  15    62.5  112.5
Jeffrey   M  13    62.5   84.0
John      M  12    59.0   99.5
Joyce     F  11    51.3   50.5
Judy      F  14    64.3   90.0
```

```
  Louise    F  12   56.3   77.0
  Mary      F  15   66.5  112.0
  Philip    M  16   72.0  150.0
  Robert    M  12   64.8  128.0
  Ronald    M  15   67.0  133.0
  Thomas    M  11   57.5   85.0
  William   M  15   66.5  112.0')
```

STATISTICAL FUNCTIONS

THE FREQUENCY FUNCTION

The `proc_freq()` function in the **procs** package replicates some of the functionality of SAS PROC FREQ. It can perform one and two-way frequencies, and supports some of the options of the SAS procedure. Unlike typical R functions, `proc_freq()` produces both dataset results and an interactive report. The interactive report is organized similar to the HTML results produced by SAS. The output datasets are also similar to SAS. Let's take a look at a simple example:

```
# Run frequencies
res <- proc_freq(cls,
                 tables = v(Sex, Age, Sex * Age))
```

In RStudio®, the above code produces an interactive report in the viewer. It looks like this:

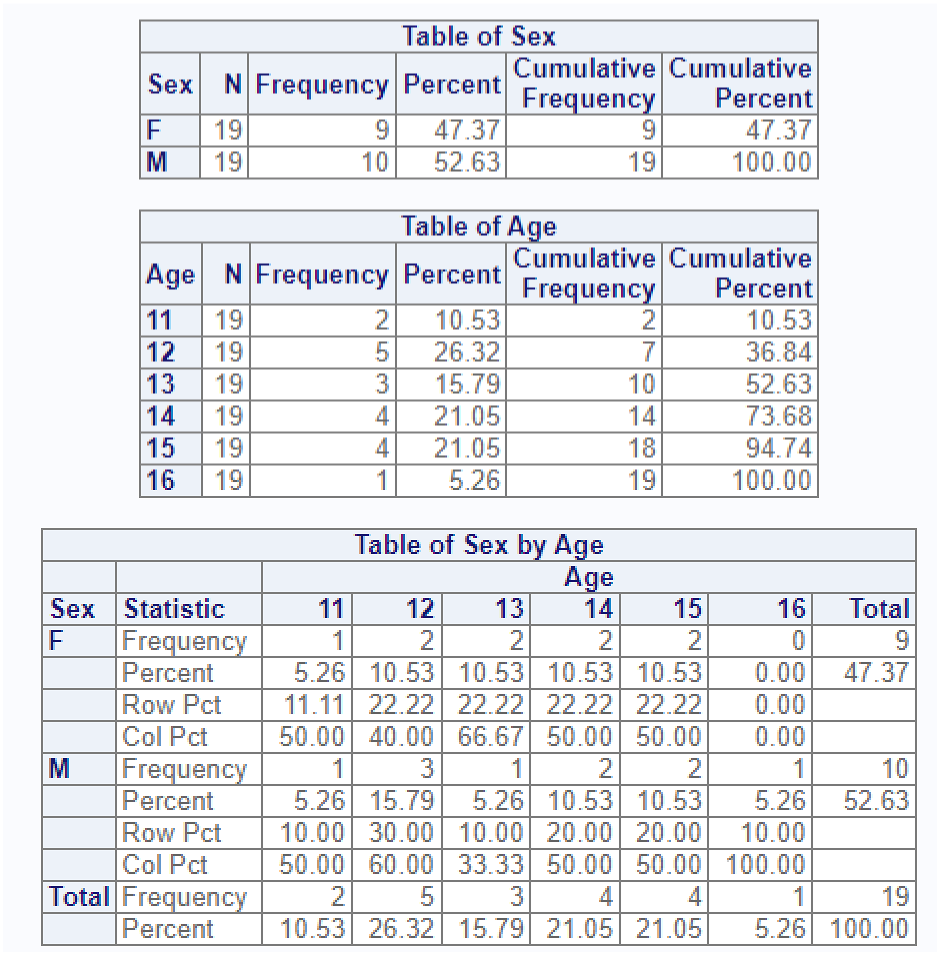


Table of Sex

Sex	N	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	19	9	47.37	9	47.37
M	19	10	52.63	19	100.00

Table of Age

Age	N	Frequency	Percent	Cumulative Frequency	Cumulative Percent
11	19	2	10.53	2	10.53
12	19	5	26.32	7	36.84
13	19	3	15.79	10	52.63
14	19	4	21.05	14	73.68
15	19	4	21.05	18	94.74
16	19	1	5.26	19	100.00

Table of Sex by Age

		Age						
Sex	Statistic	11	12	13	14	15	16	Total
F	Frequency	1	2	2	2	2	0	9
	Percent	5.26	10.53	10.53	10.53	10.53	0.00	47.37
	Row Pct	11.11	22.22	22.22	22.22	22.22	0.00	
	Col Pct	50.00	40.00	66.67	50.00	50.00	0.00	
M	Frequency	1	3	1	2	2	1	10
	Percent	5.26	15.79	5.26	10.53	10.53	5.26	52.63
	Row Pct	10.00	30.00	10.00	20.00	20.00	10.00	
	Col Pct	50.00	60.00	33.33	50.00	50.00	100.00	
Total	Frequency	2	5	3	4	4	1	19
	Percent	10.53	26.32	15.79	21.05	21.05	5.26	100.00

Notice that the above results are quite similar to the interactive results produced by SAS. The appearance and organization of these results is far superior to the results typically produced by an R statistical function.

The above code also returned datasets. Datasets are returned from the function by default. If there is more than one dataset to return, they are passed back as a list of datasets. The datasets may be examined by viewing the value "res":

```
# View results
res
# $Sex
#   VAR CAT  N CNT      PCT
# 1 Sex   F 19   9 47.36842
# 2 Sex   M 19  10 52.63158
#
# $Age
#   VAR CAT  N CNT       PCT
# 1 Age  11 19   2 10.526316
# 2 Age  12 19   5 26.315789
# 3 Age  13 19   3 15.789474
# 4 Age  14 19   4 21.052632
# 5 Age  15 19   4 21.052632
# 6 Age  16 19   1  5.263158
#
# $`Sex * Age`
#    VAR1 VAR2 CAT1 CAT2  N CNT       PCT
# 1   Sex  Age    F   11 19   1  5.263158
# 2   Sex  Age    F   12 19   2 10.526316
# 3   Sex  Age    F   13 19   2 10.526316
# 4   Sex  Age    F   14 19   2 10.526316
# 5   Sex  Age    F   15 19   2 10.526316
# 6   Sex  Age    F   16 19   0  0.000000
# 7   Sex  Age    M   11 19   1  5.263158
# 8   Sex  Age    M   12 19   3 15.789474
# 9   Sex  Age    M   13 19   1  5.263158
# 10  Sex  Age    M   14 19   2 10.526316
# 11  Sex  Age    M   15 19   2 10.526316
# 12  Sex  Age    M   16 19   1  5.263158
```

These datasets provide an easy way for you to extract values from the function, and use them for subsequent analysis. All the functions in the **procs** package operate in a similar manner. Note that the values in any returned dataset are not formatted or rounded in any way, to provide you with the most accurate results.

Conveniently, the `proc_freq()` function has capabilities for by groups, a weight variable, and several options like "nlevels", "missing", and "chisq". Documentation on the `proc_freq()` function can be found here: https://procs.r-sassy.org/reference/proc_freq.html.

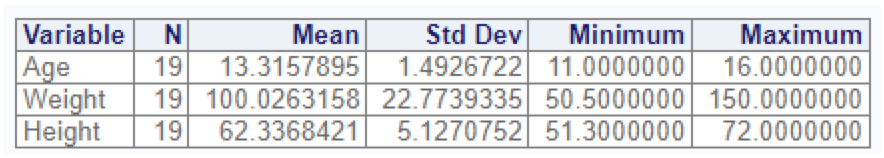
THE MEANS FUNCTION

The `proc_means()` function provides many of the statistics options of the PROC MEANS procedure in SAS. The function supports by and class variables, and options like "completetypes" and "nway". Here is an example with the default statistics options:

```
# Default stats
res <- proc_means(cls,
                  var = v(Age, Weight, Height))

# View results
res
#   TYPE FREQ    VAR  N      MEAN       STD  MIN MAX
# 1    0   19    Age 19  13.31579  1.492672 11.0  16
# 2    0   19 Weight 19 100.02632 22.773933 50.5 150
# 3    0   19 Height 19  62.33684  5.127075 51.3  72
```

Here is the interactive report from the above call:



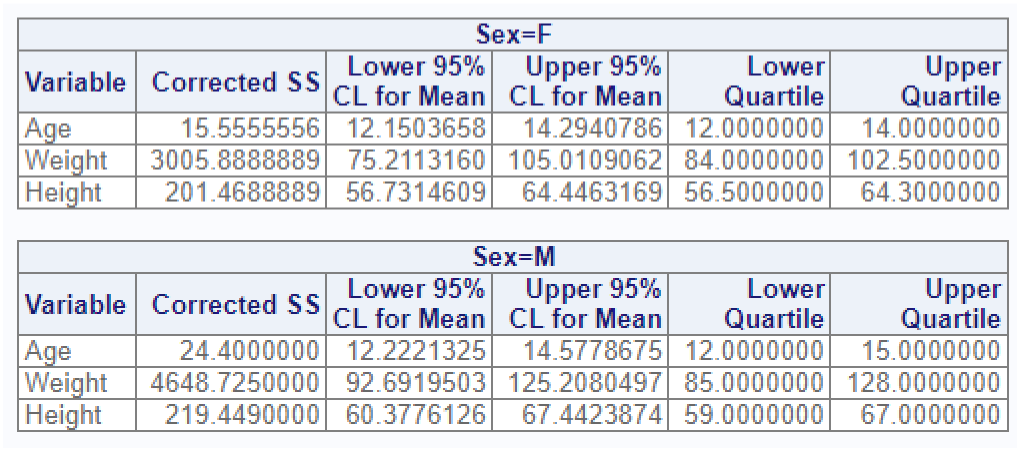
Variable	N	Mean	Std Dev	Minimum	Maximum
Age	19	13.3157895	1.4926722	11.0000000	16.0000000
Weight	19	100.0263158	22.7739335	50.5000000	150.0000000
Height	19	62.3368421	5.1270752	51.3000000	72.0000000

Here is another call with the “by” parameter and some different statistics keywords:

```
# By variable and custom stats
res <- proc_means(cls,
                  var = v(Age, Weight, Height),
                  by = Sex,
                  stats = v(css, clm, q1, q3))

# View results
res
#   BY TYPE FREQ    VAR        CSS     LCLM      UCLM   Q1    Q3
# 1  F    0    9    Age   15.55556 12.15037  14.29408 12.0  14.0
# 2  F    0    9 Weight 3005.88889 75.21132 105.01091 84.0 102.5
# 3  F    0    9 Height  201.46889 56.73146  64.44632 56.5  64.3
# 4  M    0   10    Age   24.40000 12.22213  14.57787 12.0  15.0
# 5  M    0   10 Weight 4648.72500 92.69195 125.20805 85.0 128.0
# 6  M    0   10 Height  219.44900 60.37761  67.44239 59.0  67.0
```

And here is the interactive display:



Sex=F					
Variable	Corrected SS	Lower 95% CL for Mean	Upper 95% CL for Mean	Lower Quartile	Upper Quartile
Age	15.5555556	12.1503658	14.2940786	12.0000000	14.0000000
Weight	3005.8888889	75.2113160	105.0109062	84.0000000	102.5000000
Height	201.4688889	56.7314609	64.4463169	56.5000000	64.3000000

Sex=M					
Variable	Corrected SS	Lower 95% CL for Mean	Upper 95% CL for Mean	Lower Quartile	Upper Quartile
Age	24.4000000	12.2221325	14.5778675	12.0000000	15.0000000
Weight	4648.7250000	92.6919503	125.2080497	85.0000000	128.0000000
Height	219.4490000	60.3776126	67.4423874	59.0000000	67.0000000

Documentation for `proc_means()` is here: https://procs.r-sassy.org/reference/proc_means.html.

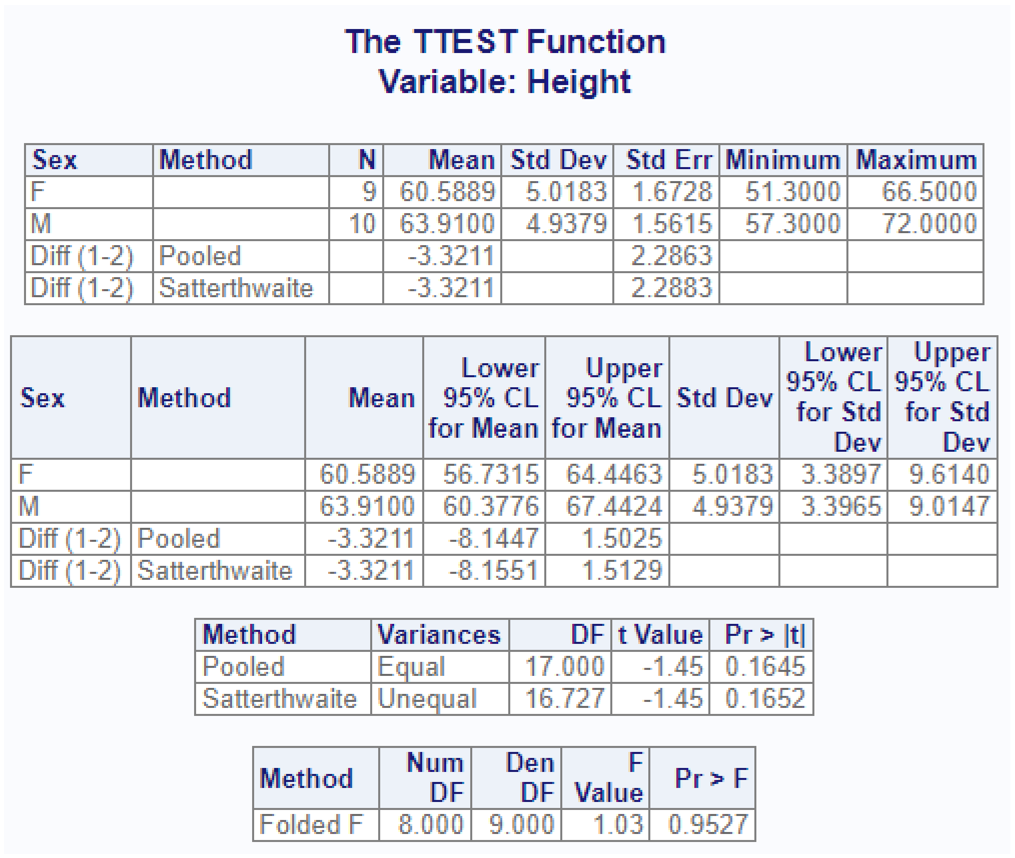
THE TTEST FUNCTION

There are three main types of hypothesis testing supported by SAS PROC TEST: One Sample, Two Sample, and Paired Samples. The `proc_ttest()` function in the **procs** package supports all three of them. To do so, the function has "paired" and "class" parameters similar to SAS. The function also provides an "h0=" setting on the "options" parameter to set the hypothesis value for a One Sample Test.

The following example shows a Two Sample t-test:

```
# Two Sample Test
res <- proc_ttest(cls, var = Height, class = Sex)
```

The interactive report looks like this:



The TTEST Function
Variable: Height

Sex	Method	N	Mean	Std Dev	Std Err	Minimum	Maximum
F		9	60.5889	5.0183	1.6728	51.3000	66.5000
M		10	63.9100	4.9379	1.5615	57.3000	72.0000
Diff (1-2)	Pooled		-3.3211		2.2863		
Diff (1-2)	Satterthwaite		-3.3211		2.2883		

Sex	Method	Mean	Lower 95% CL for Mean	Upper 95% CL for Mean	Std Dev	Lower 95% CL for Std Dev	Upper 95% CL for Std Dev
F		60.5889	56.7315	64.4463	5.0183	3.3897	9.6140
M		63.9100	60.3776	67.4424	4.9379	3.3965	9.0147
Diff (1-2)	Pooled	-3.3211	-8.1447	1.5025			
Diff (1-2)	Satterthwaite	-3.3211	-8.1551	1.5129			

Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	17.000	-1.45	0.1645
Satterthwaite	Unequal	16.727	-1.45	0.1652

Method	Num DF	Den DF	F Value	Pr > F
Folded F	8.000	9.000	1.03	0.9527

The function also returned four datasets corresponding to the results of the interactive report. We can access one of them using dollar sign ($) syntax, as follows:

```
# View results
res$TTests
#      VAR        METHOD VARIANCES       DF         T     PROBT
# 1 Height        Pooled     Equal 17.00000 -1.452625 0.1645363
# 2 Height Satterthwaite   Unequal 16.72695 -1.451319 0.1651880
```

Like `proc_means()`, `proc_ttest()` allows one or more by groups. This is a very useful feature, and does not exist in the Base R equivalent. Full documentation on the `proc_ttest()` function is here: https://procs.r-sassy.org/reference/proc_ttest.html.

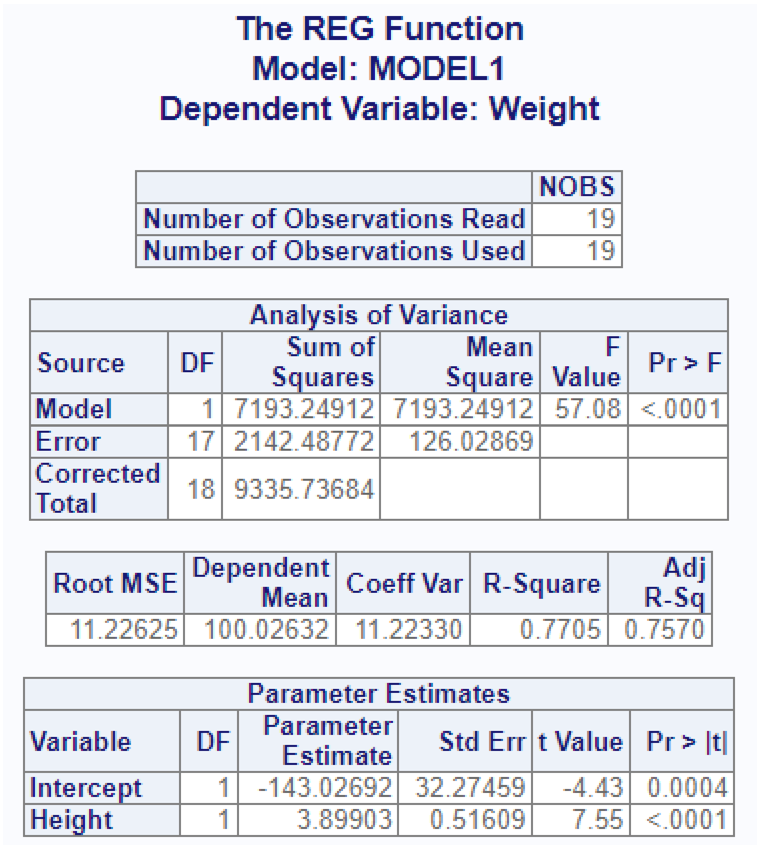
THE REG FUNCTION

The `proc_reg()` function calculates one or more linear regression models. The function supports either SAS or R model syntax. It contains several valuable options like “tableout”, “outseb”, and “press“. The options passed on the “model” statement in SAS are passed on the “stats” parameter on `proc_reg()`. Here is an example showing how to request a simple model using SAS syntax:

```
# Request model with "outseb" option
res <- proc_reg(cls,
                model = "Weight = Height",
                options = outseb)

# View results
res
#    MODEL  TYPE DEPVAR     RMSE  Intercept    Height Weight
# 1 MODEL1 PARMS Weight 11.22625 -143.02692 3.8990303     -1
# 2 MODEL1   SEB Weight 11.22625   32.27459 0.5160939     -1
```

Here are the displayed results:



The REG Function
Model: MODEL1
Dependent Variable: Weight

	NOBS
Number of Observations Read	19
Number of Observations Used	19

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	7193.24912	7193.24912	57.08	<.0001
Error	17	2142.48772	126.02869		
Corrected Total	18	9335.73684			

Root MSE	Dependent Mean	Coeff Var	R-Square	Adj R-Sq
11.22625	100.02632	11.22330	0.7705	0.7570

Parameter Estimates					
Variable	DF	Parameter Estimate	Std Err	t Value	Pr > |t|
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

The `proc_reg()` function also allows by groups and weighted observations. See the documentation at https://procs.r-sassy.org/reference/proc_reg.html for additional information.

DATA MANIPULATION FUNCTIONS

In addition to the statistical functions described above, the **procs** package includes three other functions to help manipulate and report on your data.

THE TRANSPOSE FUNCTION

SAS PROC TRANSPOSE is a commonly used procedure for pivoting data from rows to columns. Pivoting data from rows to columns is often desirable for reporting purposes. In the **procs** package, the equivalent function is `proc_transpose()`. The R function provides many of the same parameters as the SAS procedure: `id`, `var`, `by`, `copy`, `prefix`, `suffix`, etc. Here is an example:

```
# Get summary statistics
res <- proc_means(cls,
                  var = v(Age, Weight, Height),
                  by = Sex,
                  options = v(notype, nofreq))

# View results
res
#   BY    VAR  N      MEAN       STD  MIN   MAX
# 1  F    Age  9  13.22222  1.394433 11.0  15.0
# 2  F Weight  9  90.11111 19.383914 50.5 112.5
# 3  F Height  9  60.58889  5.018328 51.3  66.5
# 4  M    Age 10  13.40000  1.646545 11.0  16.0
# 5  M Weight 10 108.95000 22.727186 83.0 150.0
# 6  M Height 10  63.91000  4.937937 57.3  72.0

# Perform transpose
res_t <- proc_transpose(res, id = BY,
                        by = VAR)
# View results
res_t
#       VAR NAME          F          M
# 1     Age    N   9.000000  10.000000
# 2     Age MEAN  13.222222  13.400000
# 3     Age  STD   1.394433   1.646545
# 4     Age  MIN  11.000000  11.000000
# 5     Age  MAX  15.000000  16.000000
# 6  Height    N   9.000000  10.000000
# 7  Height MEAN  60.588889  63.910000
# 8  Height  STD   5.018328   4.937937
# 9  Height  MIN  51.300000  57.300000
# 10 Height  MAX  66.500000  72.000000
# 11 Weight    N   9.000000  10.000000
# 12 Weight MEAN  90.111111 108.950000
# 13 Weight  STD  19.383914  22.727186
# 14 Weight  MIN  50.500000  83.000000
# 15 Weight  MAX 112.500000 150.000000
```

See the `proc_transpose()` documentation for a full description of the parameters and additional examples: https://procs.r-sassy.org/reference/proc_transpose.html.

THE SORT FUNCTION

PROC SORT is one of the most frequently used procedures in SAS. The **procs** version of this procedure mimics the most commonly used features. It allows you to control the sort order, which variables are kept, and eliminates duplicates when needed. Here are some basic features of the `proc_sort()` function:

```
# Sort by age
res <- proc_sort(cls, by = Age,
                 keep = v(Name, Sex, Age))

# View results
res
#       Name Sex Age
# 11   Joyce   F  11
# 18  Thomas   M  11
# 6    James   M  12
# 7     Jane   F  12
# 10    John   M  12
# 13  Louise   F  12
# 16  Robert   M  12
# 2    Alice   F  13
# 3  Barbara   F  13
# 9  Jeffrey   M  13
# 1   Alfred   M  14
# 4    Carol   F  14
# 5    Henry   M  14
# 12    Judy   F  14
# 8    Janet   F  15
# 14    Mary   F  15
# 17  Ronald   M  15
# 19 William   M  15
# 15  Philip   M  16


# Get unique ages
res2 <- proc_sort(cls, by = Age,
                  keep = Age,
                  options = nodupkey)

# View results
res2
#    Age
# 11  11
# 6   12
# 2   13
# 1   14
# 8   15
# 15  16
```

Documentation on the `proc_sort()` function is here: https://procs.r-sassy.org/reference/proc_sort.html.

Note that this function is often used in conjunction with the `datastep()` function from the **libr** package. Documentation of that function is here: https://libr.r-sassy.org/reference/datastep.html.

THE PRINT FUNCTION

The `proc_print()` function is intended to provide quick print capabilities with little control over the formatting. This function should be used when you want to dump your statistical output to the viewer or to a file. Multiple statistical outputs can be combined into a list for a single printable output. Here is an example:

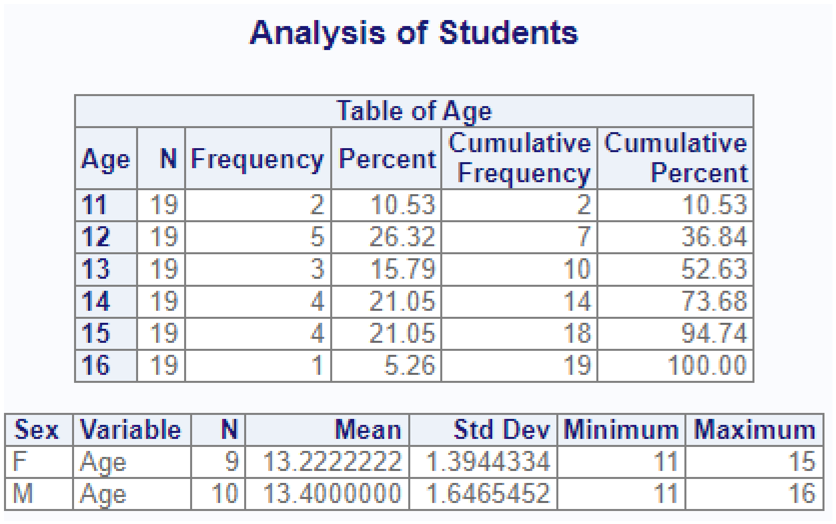
```
# Get Age frequencies
res1 <- proc_freq(cls, table = Age, output = report)

# Get Age summary stats
res2 <- proc_means(cls, var = Age, class = Sex,
                   options = v(nofreq, notype),
                   output = report)

# Combine into one list
res3 <- list(res1, res2)

# Print analysis
proc_print(res3, titles = "Analysis of Students")
```

In the above code, note that the `output = report` option returns the datasets used for the interactive report. These datasets are often more suitable for viewing. Here are the printed results:



Analysis of Students

Table of Age					
Age	N	Frequency	Percent	Cumulative Frequency	Cumulative Percent
11	19	2	10.53	2	10.53
12	19	5	26.32	7	36.84
13	19	3	15.79	10	52.63
14	19	4	21.05	14	73.68
15	19	4	21.05	18	94.74
16	19	1	5.26	19	100.00

Sex	Variable	N	Mean	Std Dev	Minimum	Maximum
F	Age	9	13.2222222	1.3944334	11	15
M	Age	10	13.4000000	1.6465452	11	16

Also note that the `proc_print()` function can print results to a file using the `file_path` and `output_type` parameters. This feature allows you to easily store your analysis to a permanent file. Complete documentation on the `proc_print()` function is here: https://procs.r-sassy.org/reference/proc_print.html.

As stated above, `proc_print()` is a quick print function with a limited feature set. For more full-featured reporting, see the reporter package at https://reporter.r-sassy.org/.

CONCLUSION

The **procs** package was written to give a familiar interface to SAS programmers when working in R. The package encapsulates commonly used statistical procedures, such as PROC FREQ, PROC MEANS, PROC TTEST, and PROC REG. The package also replicates PROC TRANSPOSE, PROC SORT, and a bit of PROC PRINT. The functions in **procs** have been validated to match SAS. They make working in R much more comfortable than using Base R functions.

REFERENCES

Bosak D (2024). *procs: Recreates Some 'SAS®' Procedures in 'R'*. R package version 1.0.6, https://github.com/dbosak01/procs, https://procs.r-sassy.org

Bosak D (2024). Package Validation for "procs". R package version 1.0.6. https://r-sassy.org/validation/Procs_Validation.pdf

Bosak D (2023). *An Overview of the SASSY System*, WUSS Paper 185-2023, https://www.lexjansen.com/wuss/2023/WUSS-2023-Paper-185.pdf

Bae K (2024). *sasLM: 'SAS' Linear Model.* R package version 0.10.2, https://CRAN.R-project.org/package=sasLM

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, Brian Varney, Kevin Kramer, and Yifei Chen.

The `proc_ttest()` and `proc_reg()` functions in the **procs** package rely heavily on the **sasLM** package by Kyun-Seop Bae. Special acknowledgement is given to him for his foundational work on that package to provide statistical routines that match SAS.

CONTACT INFORMATION

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

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

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.