PharmaSUG 2026 - Paper DV-214

Introduction to Plotting with the PROCS Package

David J. Bosak, Archytas Clinical Solutions, LLC

ABSTRACT

The "procs" R package aims to simulate several SAS procedures in R. For instance, the package has functions like "proc_freq()", "proc_ttest()" and "proc_reg()" that replicate the basic functionality of the corresponding SAS procedures. A major feature missing from these replicas has been the ability to generate plots. That missing piece has been rectified in a recent release of the "procs" package. This paper will discuss which functions now support plots, and how to generate them. The paper will give several examples demonstrating plot options, and also show how they can be exported and used in a report.

INTRODUCTION

The **procs** R package is intended to bring some of the good design features of SAS to R. These good design features include high-level procedures that make it easy for the statistician/analyst to do their job. A typical SAS procedure produces not only statistics, but also any related plots. Recently, some of these plots have been replicated in the **procs** package.

Here is a short summary of the plots that the **procs** package can reproduce:

- **proc_freq()**: Basic frequency plots, including bar charts, dot plots, stacked bar charts, and clustered bar charts.
- **proc_ttest()**: The summary panel with combined histogram and box plot is replicated, along with the QQ plot, interval plot, agreement plot, and paired profiles plot. The summary panel histogram and box plot may also be generated independently.
- **proc_reg()**: Produces almost all the basic regression plots, including the diagnostics fit panel, QQ plot, fit plot, Cook's D, residual histogram, RStudent by leverage, RStudent by predicted, residual-fit spread plot, influence plots, and more.

These plots give the user a more complete and insightful understanding of the data and the analysis, and bring a small fraction of the power of SAS into R.

HOW IT WORKS

The plotting functions are closely integrated with the corresponding statistical procedure. When possible, the plotting function will use the same results generated by the statistical procedure to create the chart. In this way, the charts are an accurate representation of the tabular statistics.

The plot functions are interfaces to Base R plots. Base R plots are stable and adequate for the task at hand. The plot functions pass the statistical results to Base R plot functions, create a JPEG file in a temporary directory, and then display the graphic file in the RStudio viewer.

DEFAULT PLOTS

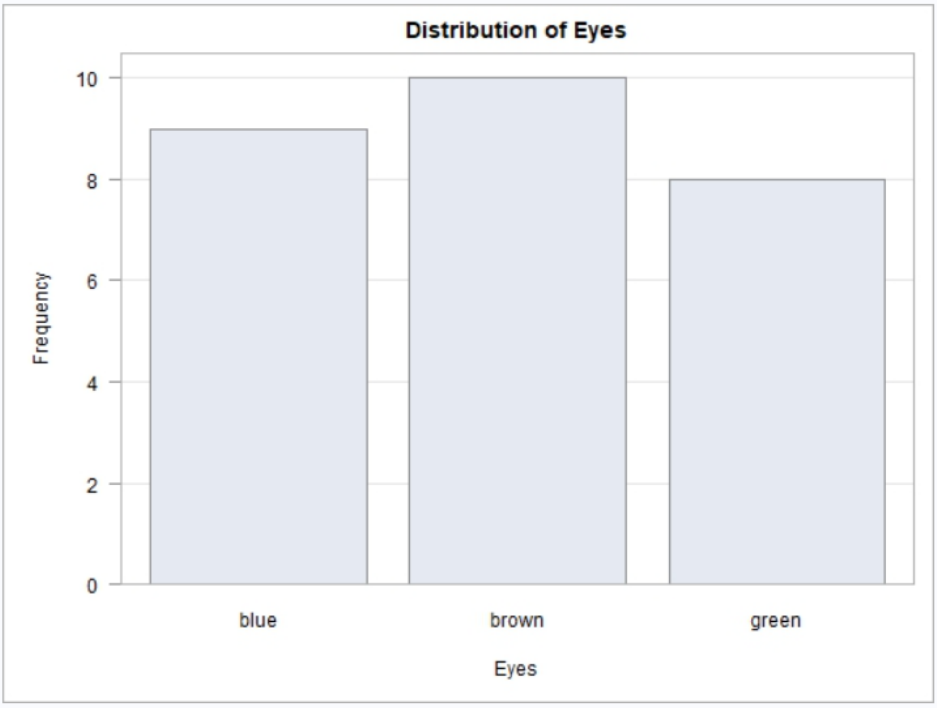
Each of the above functions now has a `plots` parameter. To get a default set of plots, simply set the `plots` parameter to TRUE. Here is an example showing how to get a default frequency plot from the `proc_freq()` function:

```
proc_freq(dat,
          tables = "Eyes",
          plots = TRUE)
```

Setting the `plots` parameter to TRUE will generate a frequency table and a bar chart that looks like this:

The FREQ Function

Table of Eyes					
Eyes	N	Frequency	Percent	Cumulative Frequency	Cumulative Percent
blue	27	9	33.33	9	33.33
brown	27	10	37.04	19	70.37
green	27	8	29.63	27	100.00



FREQUENCY PLOTS

In addition to the default plots, all functions have a special function that allows you to request customizations. For the `proc_freq()` function, the plot request function is called `freqplot()`. The syntax and parameters for the `freqplot()` function are below.

SYNTAX

```
proc_freq(<data>,
          tables = <table requests>,
          plots = freqplot(type = <plot type>,
                           orient = <TRUE or FALSE>,
                           twoway = <twoway keyword>,
                           groupby = <"row" or "column">
                           npanelpos = <number of plots per panel>))
```

PARAMETERS

Name	Description	Valid Values	Default
type	Indicates which type of plot to create.	"barchart", "dotplot"	"barchart"
orient	Specifies chart orientation.	"vertical", "horizontal"	"vertical"

scale	Specifies the type of scale for the Y axis.	“freq”, “percent”, “log”, “sqrt”, “grouppercent”	“freq”
twoway	Provides options for display of twoway interactions.	“groupvertical”, “grouphorizontal”, “stacked”, “cluster”, “grouppercent”	“groupvertical”
groupby	Controls how variables are arranged.	“column”, “row”	“column”
npanelpos	Limits how many plots are on a panel.	Integer between 1 and 10	4

EXAMPLES

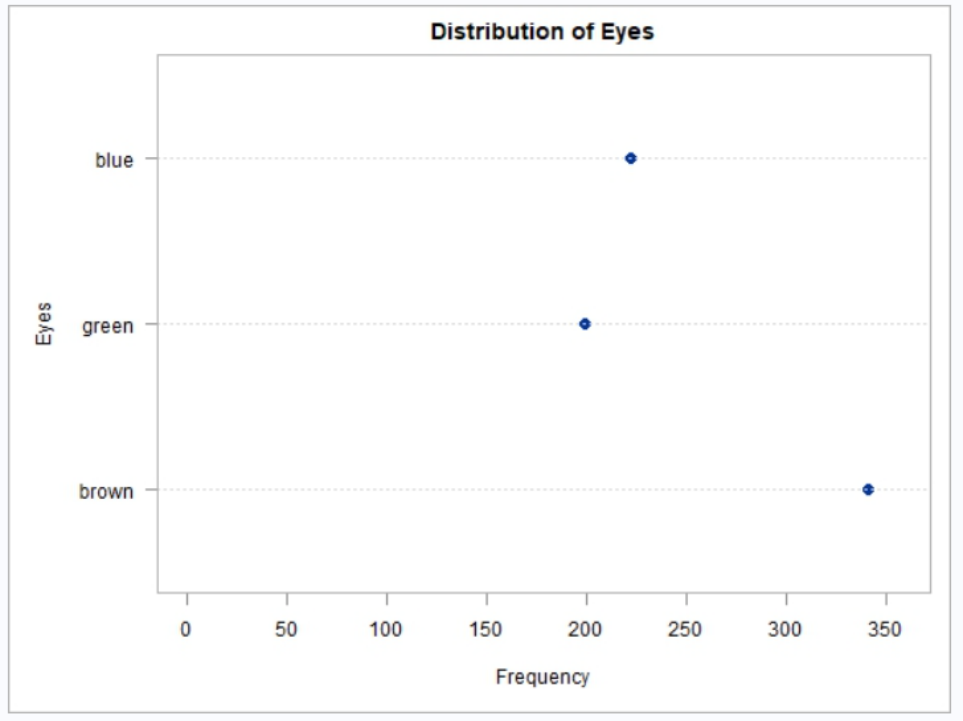
Let’s use the `freqplot()` function to create some different kinds of charts.

Dot Plot

Here is how to create a dot plot instead of a bar chart. In this example, we will also change the orientation to “horizontal” instead of “vertical”. The code will look like this:

```
proc_freq(dat,
          tables = "Eyes",
          weight = "Count",
          plots = freqplot(type = "dotplot",
                           orient = "horizontal"))
```

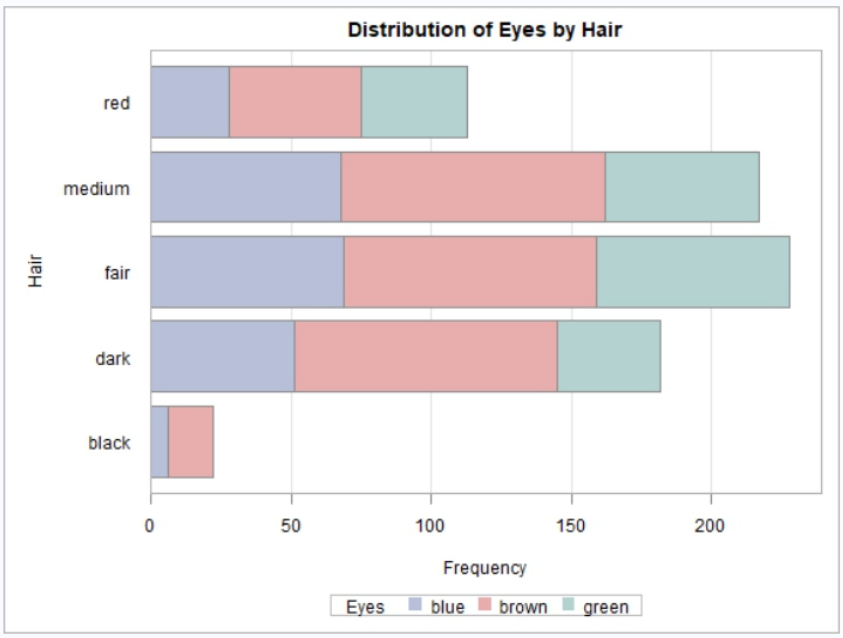
And the chart produced will look like this:



Stacked Bars

Some types of charts are available only to two-way interactions. Here is an example of a stacked bar chart:

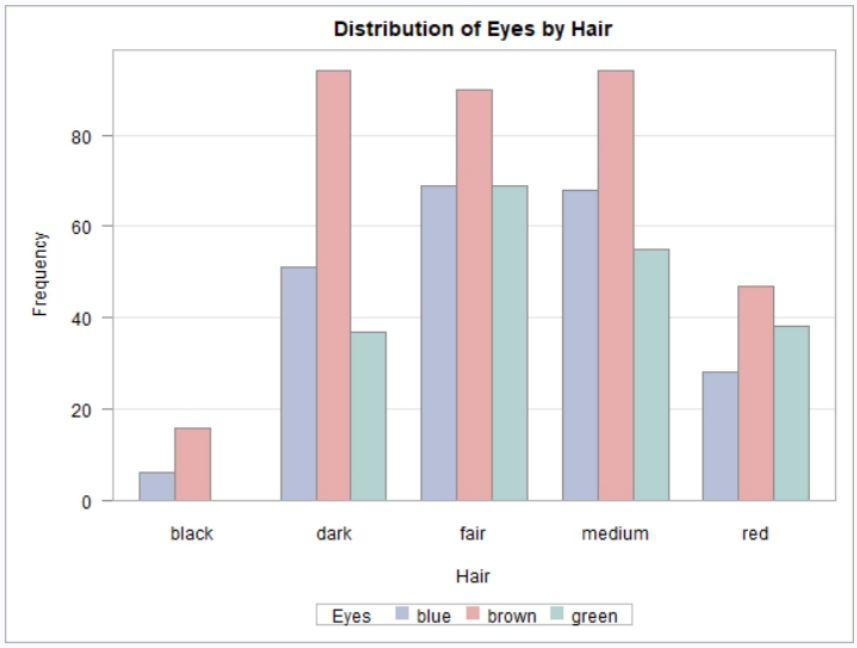
```
proc_freq(dat,
          tables = "Eyes * Hair",
          weight = "Count",
          plots = freqplot(type = "barchart",
                           orient = "horizontal",
                           twoway = "stacked"))
```



Clustered Bars

Likewise, the clustered bar chart is only for two-way interactions:

```
proc_freq(dat,
          tables = "Eyes * Hair",
          weight = "Count",
          plots = freqplot(type = "barchart",
                           orient = "vertical",
                           twoway = "cluster"))
```



To learn more about the `freqplot()` function, see the documentation here.

TTEST PLOTS

Plots for `proc_ttest()` are requested very much in the same way as the plots for `proc_freq()`. For any kind of T-Test, you may request a default set of plots by setting `plots = TRUE` on the `proc_ttest()` function. If you want to customize the plot output, use the function `ttestplot()`. Here are the syntax and parameters.

SYNTAX

```
proc_ttest(<data>,
           var = <One or two sample variable>,
           paired = <paired variable>,
           class = <class variable for two-sample test>,
           plots = ttestplot(type = <plot type>,
                             panel = <TRUE or FALSE>,
                             showh0 = <TRUE or FALSE>)
```

PARAMETERS

Name	Description	Valid Values	Default
type	The type(s) of plot to produce. You can pass multiple types in a vector.	"summary", "agreement", "boxplot", "histogram", "interval", "profiles", and "qqplot".	Depends on type of analysis.
panel	Whether or not to show the histogram and boxplot in one combined summary panel.	TRUE or FALSE	TRUE
showh0	Whether or not to show the h0 reference line in the box plot and interval plot.	TRUE or FALSE	FALSE

EXAMPLES

While the `freqplot()` function can request only a single chart, the `ttestplot()` function may request multiple charts. To request multiple charts, pass the desired chart names in a vector on the `type` parameter. Like this:

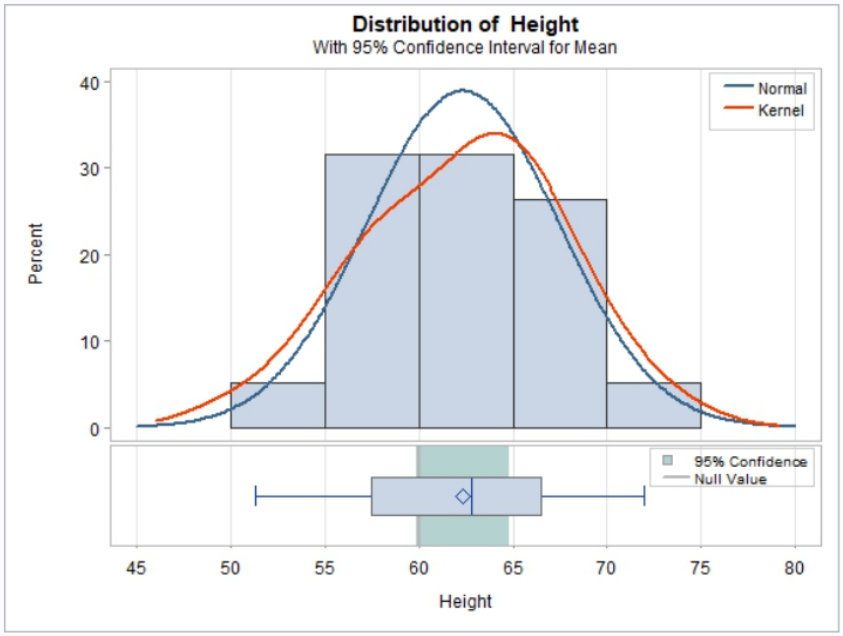
```
proc_ttest(cls,
           var = "Height",
           options = v(h0 = 60)
           plots = ttestplot(type = v(summary, qqplot, interval),
                             showh0 = TRUE)
```

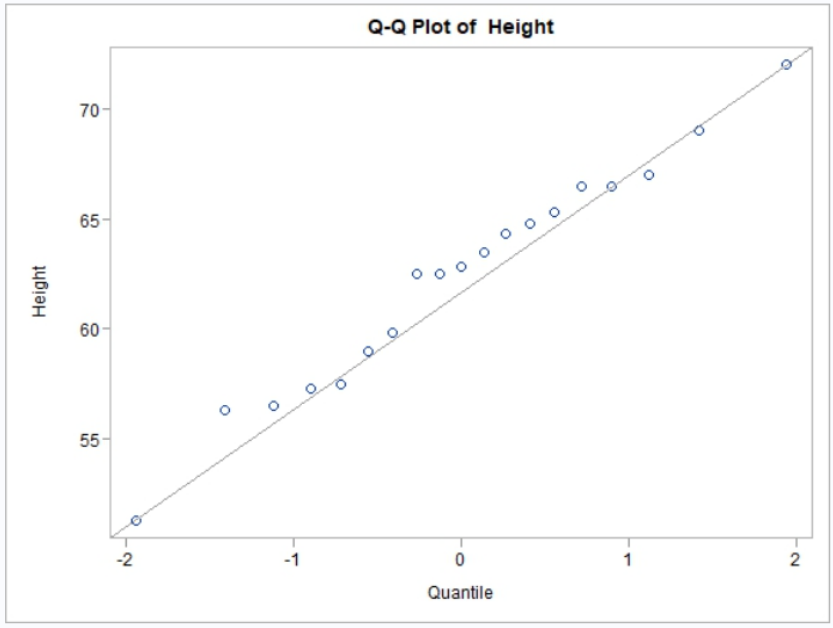
The TTEST Function
Variable: Height

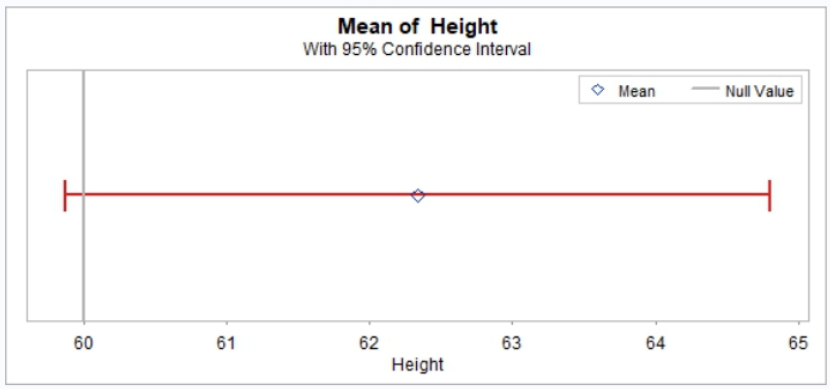
N	Mean	Std Dev	Std Err	Minimum	Maximum
19	62.3368	5.1271	1.1762	51.3000	72.0000

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
62.3368	59.8657	64.8080	5.1271	3.8741	7.5820

DF	t Value	Pr > |t|
18.000	1.99	0.0624


Distribution of Height
With 95% Confidence Interval for Mean
Normal
Kernel
Percent
40
30
20
10
0
95% Confidence
Null Value
45
50
55
60
65
70
75
80
Height


Q-Q Plot of Height
Height
70
65
60
55
-2
-1
0
1
2
Quantile


Mean of Height
With 95% Confidence Interval
Mean
Null Value
60
61
62
63
64
65
Height

To see more information about the `ttestplot()` function, navigate to the documentation here.

REGRESSION PLOTS

There are many types of regression plots. Some of these plots serve to better understand your data. Some are to understand the quality of the model. Still others help identify data points that influence the model. First, let's understand the syntax and parameters available, and then we can see an example:

SYNTAX

```
proc_reg(<data>,
         model = <model specification>,
         plots = regplot(type = <vector of plot types>,
                         panel = <TRUE or FALSE>,
                         stats = <vector of stats>)
```

PARAMETERS

Name	Description	Valid Values	Default
type	A vector of one or more plot types to generate.	"diagnostics", "cooksd", "dfbetas", "dffits", "fitplot", "observedbypredicted", "qqplot", "residuals", "residualbypredicted", "residualhistogram", "rfplot", "rstudentbyleverage", "rstudentbypredicted"	"diagnostics", "residuals", "fitplot"
panel	Whether to combine diagnostics charts into a single panel.	TRUE or FALSE	TRUE
stats	Which statistics to show on the diagnostics panel and the fit plot.	"adjrsq", "aic", "coeffvar", "depmean", "default", "edf", "mse", "nobs", "nparm", "rsquare", "sse", and "none".	"default"

EXAMPLES

Like the `proc_freq()` and `proc_ttest()` functions, you may request a default set of plots on `proc_reg()` by setting `plots = TRUE`. To customize your plot output, you can either pass a vector of plot names directly to the plots parameter, or pass a `regplot()` function. Here is an example passing a `regplot()` function:

```
proc_reg(cls,
         model = "Weight = Height",
         plots = regplot(type = v(diagnostics, residuals, fitplot),
                         label = TRUE, id = "Name"))
```

The REG Function
Model: MODEL1
Dependent Variable: Weight

	NOBS
Number of Observations Read	19
Number of Observations Used	19

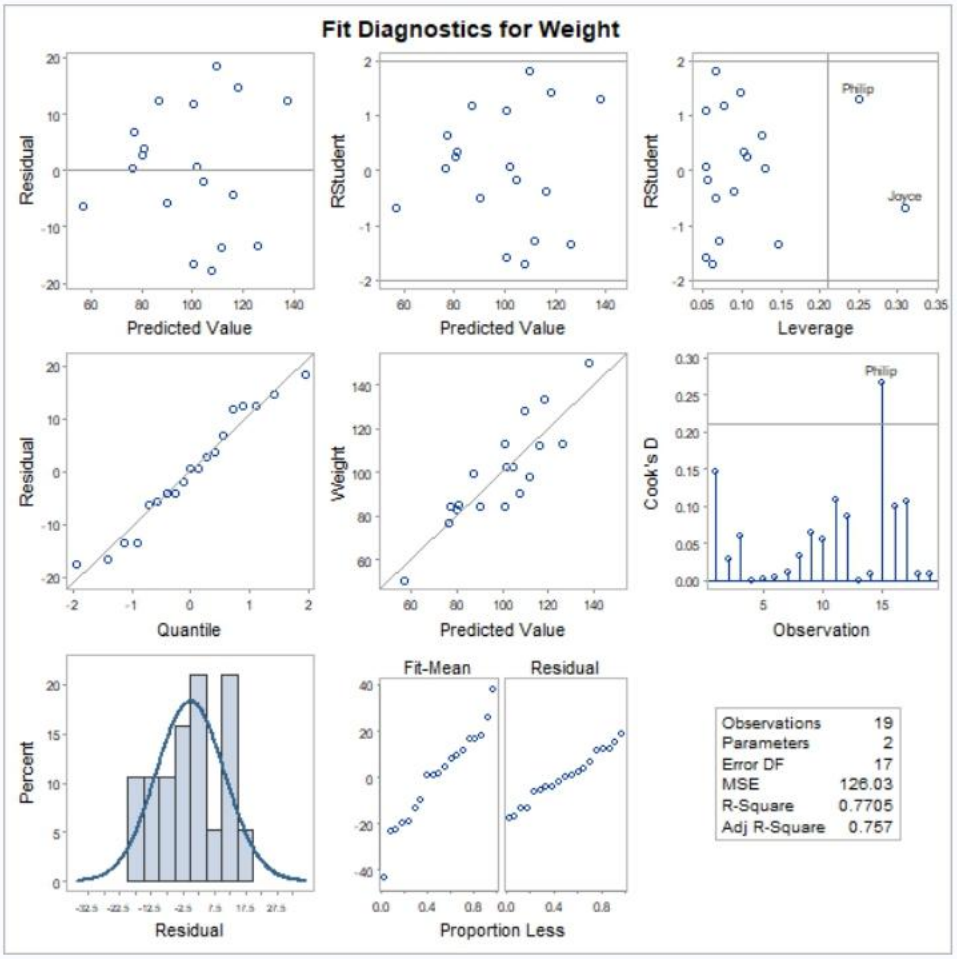
Analysis of Variance

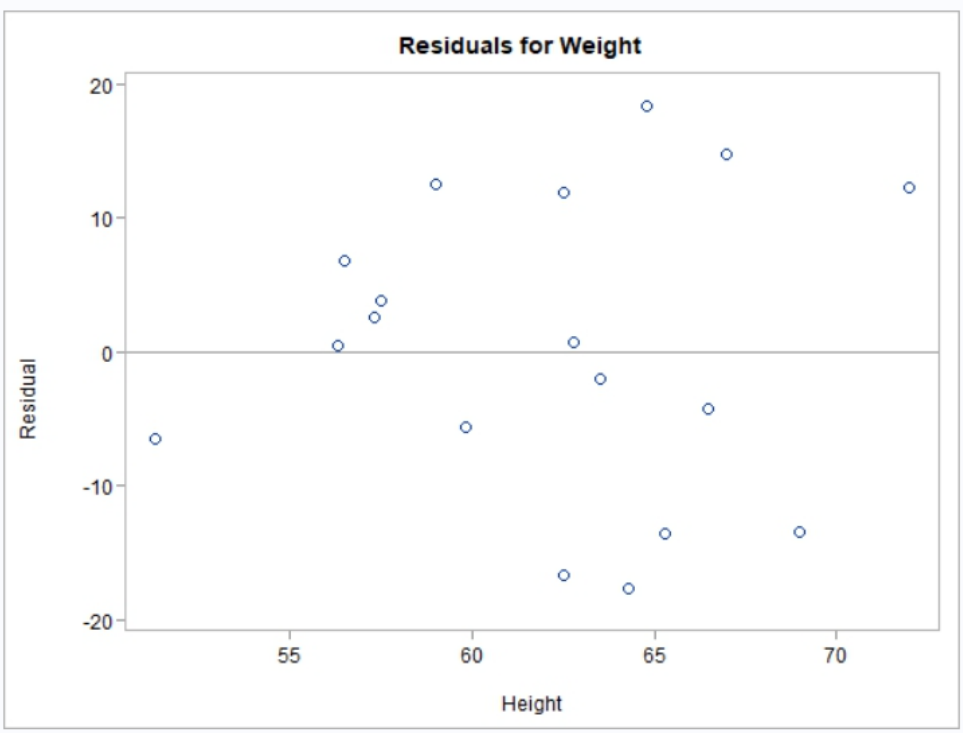
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	7193.24912	7193.24912	57.08	<.0001
Error	17	2142.48772	126.02869		
Corrected Total	18	9335.73684			

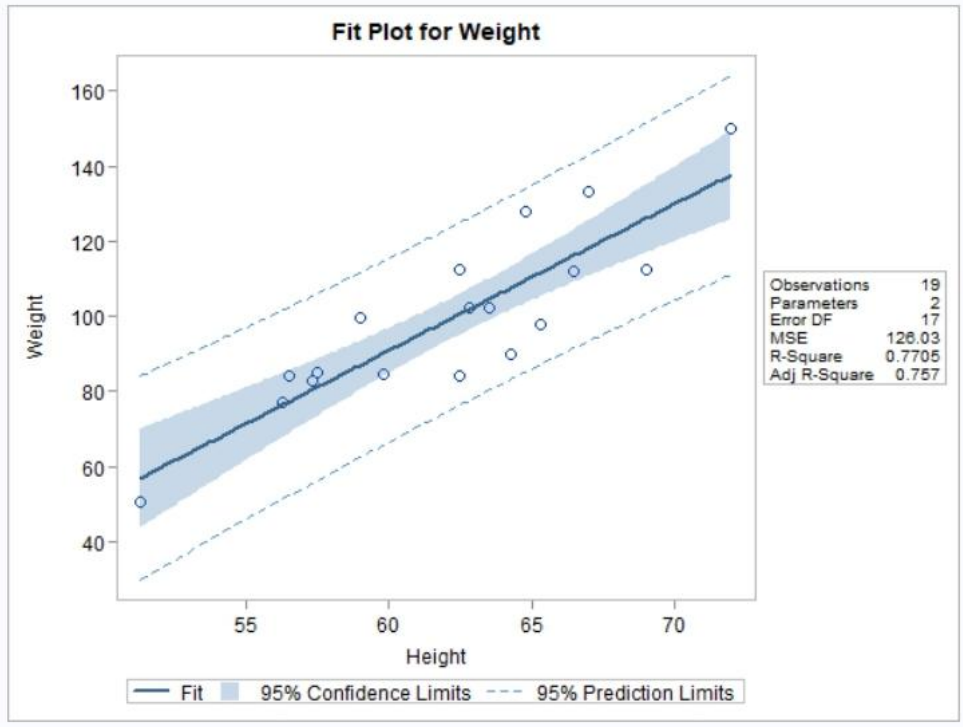
Root MSE	Dependent Mean	Coeff Var	R-Square	Adj R-Sq
11.22625	100.02632	11.22330	0.7705	0.7570

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







For more information about the `regplot()` function, see the documentation here.

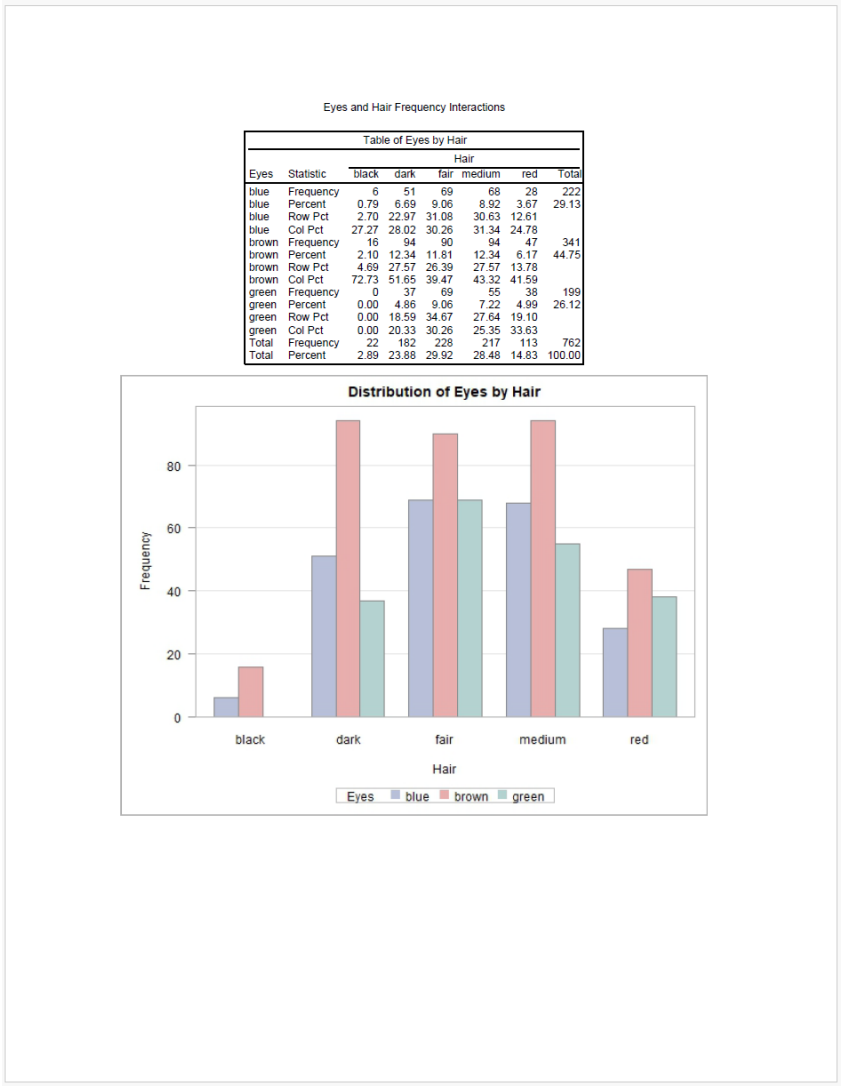
EXPORTING PLOTS

You can export any of the above plots to a report. To export to a report, simply return the report object from the target function, send to `proc_print()`, and specify the file name and output type. Here is an example:

```
# Return frequency results
res <- proc_freq(dat,
                 tables = "Eyes * Hair",
                 weight = "Count",
                 output = report,
                 plots = freqplot(type = "barchart",
                                  orient = "vertical",
                                  twoway = "cluster"))

# Print to file
proc_print(res, file_path = "c:/Projects/Archytas/PharmaSUG/2026/Test.pdf",
           output_type = "PDF",
           titles = "Eyes and Hair Frequency Interactions")
```

And here is a screen shot of the PDF report output:



Eyes and Hair Frequency Interactions

Table of Eyes by Hair

		Hair					
Eyes	Statistic	black	dark	fair	medium	red	Total
blue	Frequency	6	51	69	68	28	222
blue	Percent	0.79	6.69	9.06	8.92	3.67	29.13
blue	Row Pct	2.70	22.97	31.08	30.63	12.61	
blue	Col Pct	27.27	28.02	30.26	31.34	24.78	
brown	Frequency	16	94	90	94	47	341
brown	Percent	2.10	12.34	11.81	12.34	6.17	44.75
brown	Row Pct	4.69	27.57	26.39	27.57	13.78	
brown	Col Pct	72.73	51.65	39.47	43.32	41.59	
green	Frequency	0	37	69	55	38	199
green	Percent	0.00	4.86	9.06	7.22	4.99	26.12
green	Row Pct	0.00	18.59	34.67	27.64	19.10	
green	Col Pct	0.00	20.33	30.26	25.35	33.63	
Total	Frequency	22	182	228	217	113	762
Total	Percent	2.89	23.88	29.92	28.48	14.83	100.00

LIMITATIONS

Observe that not all plots available in SAS have been replicated in the **procs** package. There are many specialized frequency plots that are not available, and also specialized types of regression plots. The authors focused on the most commonly used plots from these three procedures. Other plots may be

added to the package in the future. For the latest information on the **procs** package, visit the package web site at https://procs.r-sassy.org.

CONCLUSION

Many years ago, SAS gave users the capability to produce plots along with tabular output. Plots can deepen understanding of your data and improve your analysis. This type of combined output has been missing in the R world, and specifically in the **procs** package. Now that missing capability has been rectified, and the procs package is able to produce the most commonly used plots for `proc_freq()`, `proc_ttest()`, and `proc_reg()`. Future enhancements will target more types of analysis and more plotting features.

REFERENCES

Bosak, David J. 2024. "Replicating SAS® Procedures in R with the PROCS Package." *Proceedings of PharmaSUG,* Baltimore, Maryland: Available at https://pharmasug.org/proceedings/2024/AP/PharmaSUG-2024-AP-295.pdf.

CRAN, Bosak, David J. "procs: Recreates some 'SAS®' Procedures in R", 2025, Available at https://cran.r-project.org/web/packages/procs/index.html

R Core Team, R Foundation for Statistical Computing, Vienna, Austria, 2025. Available at https://www.R-project.org/

ACKNOWLEDGMENTS

The author would like to thank Chang-Yu Huang for his valuable suggestions and feedback during the early stages of **procs** plot development.

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