SESUG Paper 072-2021

The *reporter* package: A Powerful and Easy-to-use Reporting Package for R

David J. Bosak, r-sassy.org

ABSTRACT

SAS® programmers who come to R are often disappointed by the reporting options available. Creating a report that takes a few minutes in SAS® can take hours in R. Sometimes it appears impossible to create an equivalent report at all. The *reporter* package was built to overcome these difficulties. This package contains functions to create regulatory-style statistical reports. Originally designed to generate tables, listings, and figures (TLFs) for the pharmaceutical, biotechnology, and medical device industries, these reports are generalized enough that they could be used in any industry. The *reporter* package can output text, rich-text, and PDF file formats. The package specializes in printing wide and long tables with automatic page wrapping and splitting. Reports can be produced with a minimum of function calls, and without relying on other table packages. The package supports titles, footnotes, page headers, page footers, spanning headers, page by variables, and automatic page numbering. This paper will provide a brief overview of the *reporter* package. The reader should have some familiarity with the R language and RStudio®.

INTRODUCTION

The 2020 pandemic gave me an opportunity to do something I've been wanting to do for years: learn R. After some introductory lessons on basic R syntax, I tried to create a listing for a sample dataset I had sitting around. The dataset had about 300 rows and 56 columns. Obviously, this dataset wasn't going to fit on a single page. In SAS®, this task would have taken about five lines of code. Yet I seemed to be struggling to write the equivalent code in R. How hard could it be?

Six weeks later, I still couldn't do it. I had learned about *RMarkdown*, *pandoc*, *LaTeX*, *pander*, *flextable*, *officer*, *gt*, *dt*, *huxtable*, and probably a dozen more R packages. Yet my seemingly simple request still wasn't fulfilled: create a data listing with titles that repeat on every page and break/wrap the data so that everything fits without flowing into the margins. I was amazed at how difficult it was to accomplish this task in R.

That was the point at which I realized that R wasn't nearly as mature as I had expected. There were still a lot of opportunities for improvement. What I found exciting about R is that I could do the improvement myself. So I began writing the **reporter** package.

The *reporter* package aims to create printable reports in as few lines of code as possible. It specializes in printing large reports that don't fit on one page. It supports repeating titles, footnotes, page headers, and page footers. You can output in multiple file formats: TXT, RTF, and PDF. You can add *ggplot* graphics to create figures. And it supports page-by variables, X of Y page numbering, and many other features to make your life easier.

For those of you with a background in SAS®, you can think of the *reporter* package as a R equivalent to PROC PRINT or PROC REPORT. For those of you with a background in R, you can think of the *reporter* package as a pipe-enabled set of functions that creates reports directly, without using *RMarkdown*. This paper is a brief introduction to the package.

HOW TO INSTALL

The reporter package can be installed from the R console. Simply run the following command:

```
install.packages("reporter")
```

Then put the following line at the top of your program or script:

```
library(reporter)
```

To view the help pages for the package, run `?reporter` on the console.

CREATE A LISTING

After hundreds of hours developing the *reporter* package, you can now easily create the listing I attempted at the beginning of the pandemic. And it only takes five lines of code! Here they are:

```
library(reporter)

# Get temporary file path
tmp <- file.path(tempdir(), "example1.rtf")

# Create the report
rpt <- create_report(tmp,
                     orientation = "portrait",
                     output_type = "RTF") %>%
   page_header(left = "Sponsor: AHS",
               right = "Study: Iris") %>%
   titles("Listing 1.0", "IRIS Data Listing") %>%
   add_content(create_table(iris)) %>%
   page_footer(left = Sys.time(),
               center = "Confidential",
               right = "Page [pg] of [tpg]")

# Write the report
write_report(rpt)

# View the report
file.show(tmp)
```

The above lines of code create a three-page listing, shown here:

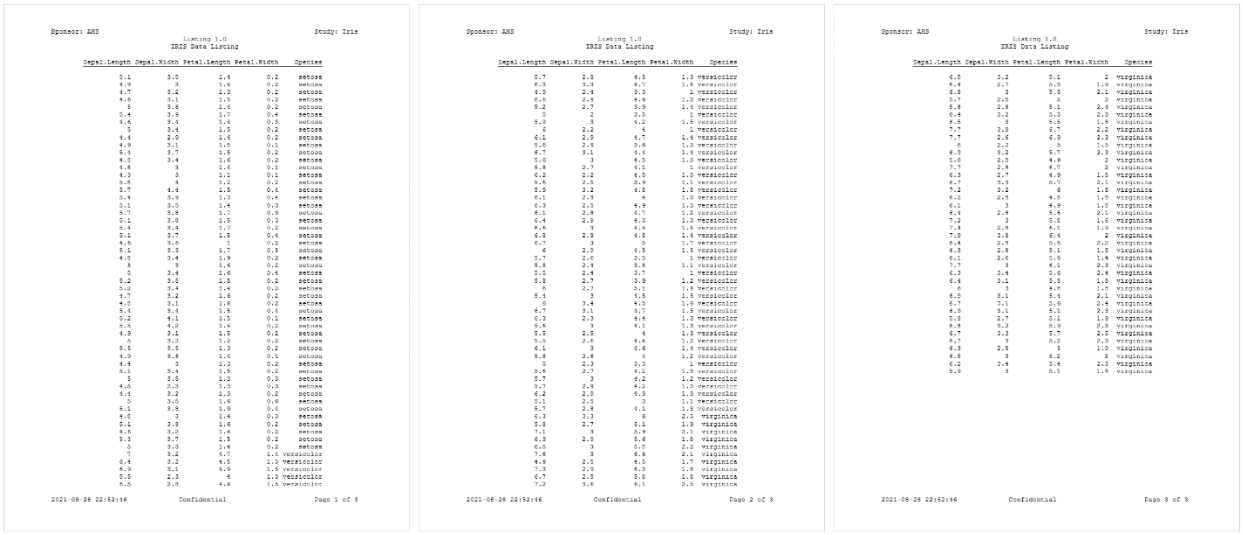


Figure 1. A Table with Spanning Headers

Note the following about this report:

- The titles, page header, and page footer repeat on every page automatically
- The column headers repeat on every page automatically
- Column widths are determined automatically

- The pages break at an appropriate place automatically
- The page numbers increment automatically
- If the table had more columns, it would break horizontally to the next page automatically
- You can output the same report in TXT or PDF by changing the `output_type` parameter and the file name

These are all features that currently do not exist as part of any other R reporting package. To users of SAS®, they are all expected features. But these features have not been available in R until now.

CREATE A TABLE

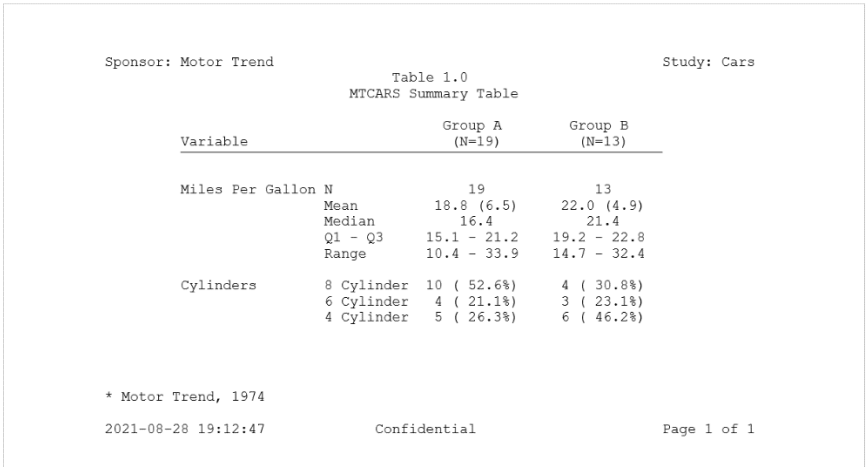
Further, you can create a summary table of statistics from the same package. Tables generally take the following steps:

1. Create the table content
2. Create the report and add the content
3. Write the report

The *reporter* package contains many functions and features to control the output.

A SIMPLE TABLE

Let us start with a simple table. The table will look like this:



Sponsor: Motor Trend — Study: Cars

Table 1.0
MTCARS Summary Table

Variable		Group A (N=19)	Group B (N=13)
Miles Per Gallon	N	19	13
	Mean	18.8 (6.5)	22.0 (4.9)
	Median	16.4	21.4
	Q1 - Q3	15.1 - 21.2	19.2 - 22.8
	Range	10.4 - 33.9	14.7 - 32.4
Cylinders	8 Cylinder	10 (52.6%)	4 (30.8%)
	6 Cylinder	4 (21.1%)	3 (23.1%)
	4 Cylinder	5 (26.3%)	6 (46.2%)

* Motor Trend, 1974

2021-08-28 19:12:47 Confidential Page 1 of 1

Figure 2. A Simple Table

To speed up the example, we will use prepared data. The data is this:

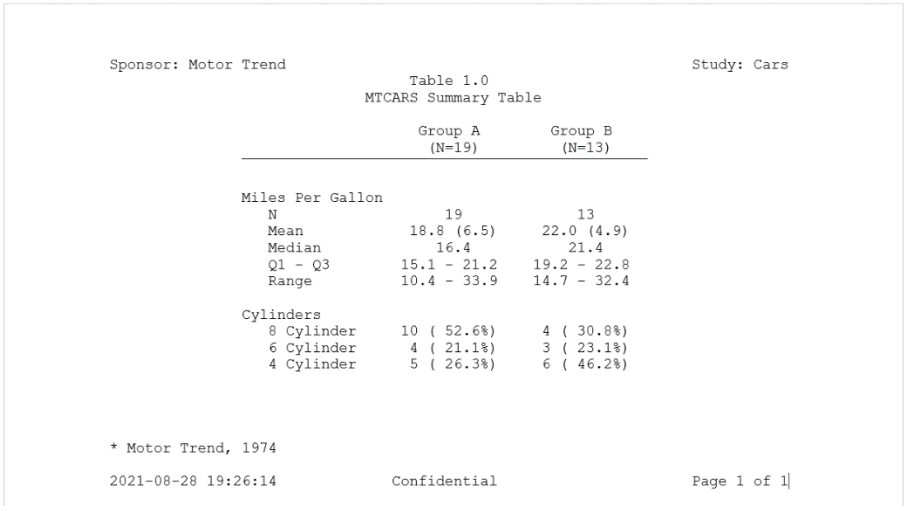
```
# Read in prepared data
df <- read.table(header = TRUE, text = '
      var     label         A              B
      "ampg"   "N"           "19"           "13"
      "ampg"   "Mean"        "18.8 (6.5)"   "22.0 (4.9)"
      "ampg"   "Median"      "16.4"         "21.4"
      "ampg"   "Q1 - Q3"     "15.1 - 21.2" "19.2 - 22.8"
      "ampg"   "Range"       "10.4 - 33.9" "14.7 - 32.4"
      "cyl"    "8 Cylinder" "10 ( 52.6%)" "4 ( 30.8%)"
      "cyl"    "6 Cylinder" "4 ( 21.1%)"  "3 ( 23.1%)"
      "cyl"    "4 Cylinder" "5 ( 26.3%)"  "6 ( 46.2%)"')
```

To create the report in Figure 2, it will be necessary to have greater control over the column definitions. The column definitions can be controlled using the `define()` function. This function allows you to set the column label, width, alignment, and much more. Examine the following example:

```
#library(reporter)

# Create temporary path
tmp <- file.path(tempdir(), "example2.rtf")

# Create table
tbl <- create_table(df, first_row_blank = TRUE) %>%
  define(var, label = "Variable", blank_after = TRUE, dedupe = TRUE,
         format = c(ampg = "Miles Per Gallon", cyl = "Cylinders")) %>%
  define(label, label = "") %>%
  define(A, label = "Group A", width = 1.25, align = "center", n = 19) %>%
  define(B, label = "Group B", width = 1.25, align = "center", n = 13)

# Create report and add content
rpt <- create_report(tmp, orientation = "portrait",
                     output_type = "RTF") %>%
  page_header(left = "Sponsor: Motor Trend", right = "Study: Cars") %>%
  titles("Table 1.0", "MTCARS Summary Table") %>%
  add_content(tbl) %>%
  footnotes("* Motor Trend, 1974") %>%
  page_footer(left = Sys.time(),
              center = "Confidential",
              right = "Page [pg] of [tpg]")

# Write out report
write_report(rpt)

# View the report
file.show(tmp)
```

A SIMPLE TABLE WITH STUB

Now let's modify the above example slightly to add a stub column. A stub column will nest the labels of the first two columns in a hierarchical manner, like so:



Sponsor: Motor Trend — Study: Cars

Table 1.0
MTCARS Summary Table

	Group A (N=19)	Group B (N=13)
Miles Per Gallon		
N	19	13
Mean	18.8 (6.5)	22.0 (4.9)
Median	16.4	21.4
Q1 - Q3	15.1 - 21.2	19.2 - 22.8
Range	10.4 - 33.9	14.7 - 32.4
Cylinders		
8 Cylinder	10 (52.6%)	4 (30.8%)
6 Cylinder	4 (21.1%)	3 (23.1%)
4 Cylinder	5 (26.3%)	6 (46.2%)

* Motor Trend, 1974

2021-08-28 19:26:14 — Confidential — Page 1 of 1

Figure 3. A Simple Table with Stub Column

The report in Figure 3 is created in a nearly identical manner to the report in Figure 2. The only differences are the addition of a `stub()` function to combine the first two columns of the report, and some adjustments to the `define()` statements for those two columns. The code for this report is as follows:

```
#library(reporter)

# Create temporary path
tmp <- file.path(tempdir(), "example3.rtf")

# Create table
tbl <- create_table(df, first_row_blank = TRUE) %>%
  stub(c("var", "label")) %>%
  define(var, blank_after = TRUE, label_row = TRUE,
         format = c(ampg = "Miles Per Gallon", cyl = "Cylinders")) %>%
  define(label, indent = .25) %>%
  define(A, label = "Group A", width = 1.25, align = "center", n = 19) %>%
  define(B, label = "Group B", width = 1.25, align = "center", n = 13)


# Create report and add content
rpt <- create_report(tmp, orientation = "portrait", output_type = "RTF")
%>%
  page_header(left = "Sponsor: Motor Trend", right = "Study: Cars") %>%
  titles("Table 1.0", "MTCARS Summary Table") %>%
  add_content(tbl) %>%
  footnotes("* Motor Trend, 1974") %>%
  page_footer(left = Sys.time(),
              center = "Confidential",
              right = "Page [pg] of [tpg]")

# Write out report
write_report(rpt)

# View the report
file.show(tmp)
```

SPANNING HEADERS

Most reporting tools in R do not support spanning headers. The *reporter* package does. Not only does it support spanning headers, you can define them in a very intuitive way. You simply add a `spanning_header()` to the table, tell it the columns you want to span, and supply a label. If you want more than one level of spanning header, you can define it using the `level` parameter.

Here is a sample report showing multiple spanning headers:

Table 1.0
MTCARS Spanning Headers

				Meta-Group					
	Group 1 (N=10)			Group 2 (N=10)			Group 3 (N=10)		
Vehicle	mpg	cyl	hp	drat	wt	qsec	vs	gear	carb
Mazda RX4	21.0	6	110	3.9	2.62	16.46	0	4	4
Mazda RX4 Wag	21.0	6	110	3.9	2.875	17.02	0	4	4
Datsun 710	22.8	4	93	3.85	2.32	18.61	1	4	1
Hornet 4 Drive	21.4	6	110	3.08	3.215	19.44	1	3	1
Hornet Sportabout	18.7	8	175	3.15	3.44	17.02	0	3	2
Valiant	18.1	6	105	2.76	3.46	20.22	1	3	1
Duster 360	14.3	8	245	3.21	3.57	15.84	0	3	4
Merc 240D	24.4	4	62	3.69	3.19	20	1	4	2
Merc 230	22.8	4	95	3.92	3.15	22.9	1	4	2
Merc 280	19.2	6	123	3.92	3.44	18.3	1	4	4

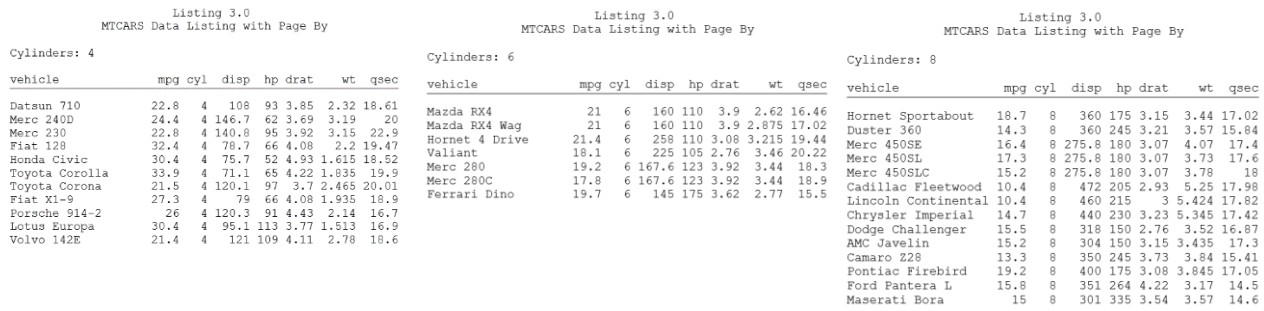
Figure 4. A Table with Spanning Headers

Here is the code that produces the above example:

```
library(reporter)

# Create temporary path
tmp <- file.path(tempdir(), "example4.rtf")

# Prepare Data
dat <- mtcars[1:10, ]
df <- data.frame(vehicle = rownames(dat), dat)

# Define Table with spanning headers
tbl <- create_table(df) %>%
  titles("Table 1.0", "MTCARS Spanning Headers") %>%
  spanning_header(from = "mpg", to = "hp", label = "Group 1", n = 10) %>%
  spanning_header(from = "drat", to = "qsec", label = "Group 2",n = 10) %>%
  spanning_header(from = "vs", to = "carb", label = "Group 3", n = 10) %>%
  spanning_header(from = "drat", to = "carb",
                  label = "Meta-Group", level = 2) %>%
  define(vehicle, label = "Vehicle") %>%
  define(mpg, format = "%.1f") %>%
  define(disp, visible = FALSE) %>%
  define(am, visible = FALSE)

# Create Report and add table
rpt <- create_report(tmp, output_type = "RTF") %>%
  add_content(tbl, align = "left")

# Write the report
write_report(rpt)

# View the report
file.show(tmp)
```

PAGE BY

The page-by is another feature that is not well-supported in other R packages. The page-by groups the report by the values of a particular variable. Here are three pages of a simple report with a page-by:



Listing 3.0
MTCARS Data Listing with Page By

Cylinders: 4

vehicle	mpg	cyl	disp	hp	drat	wt	qsec
Datsun 710	22.8	4	108	93	3.85	2.32	18.61
Merc 240D	24.4	4	146.7	62	3.69	3.19	20
Merc 230	22.8	4	140.8	95	3.92	3.15	22.9
Fiat 128	32.4	4	78.7	66	4.08	2.2	19.47
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.9
Toyota Corona	21.5	4	120.1	97	3.7	2.465	20.01
Fiat X1-9	27.3	4	79	66	4.08	1.935	18.9
Porsche 914-2	26	4	120.3	91	4.43	2.14	16.7
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.9
Volvo 142E	21.4	4	121	109	4.11	2.78	18.6

Listing 3.0
MTCARS Data Listing with Page By

Cylinders: 6

vehicle	mpg	cyl	disp	hp	drat	wt	qsec
Mazda RX4	21	6	160	110	3.9	2.62	16.46
Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44
Valiant	18.1	6	225	105	2.76	3.46	20.22
Merc 280	19.2	6	167.6	123	3.92	3.44	18.3
Merc 280C	17.8	6	167.6	123	3.92	3.44	18.9
Ferrari Dino	19.7	6	145	175	3.62	2.77	15.5

Listing 3.0
MTCARS Data Listing with Page By

Cylinders: 8

vehicle	mpg	cyl	disp	hp	drat	wt	qsec
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02
Duster 360	14.3	8	360	245	3.21	3.57	15.84
Merc 450SE	16.4	8	275.8	180	3.07	4.07	17.4
Merc 450SL	17.3	8	275.8	180	3.07	3.73	17.6
Merc 450SLC	15.2	8	275.8	180	3.07	3.78	18
Cadillac Fleetwood	10.4	8	472	205	2.93	5.25	17.98
Lincoln Continental	10.4	8	460	215	3	5.424	17.82
Chrysler Imperial	14.7	8	440	230	3.23	5.345	17.42
Dodge Challenger	15.5	8	318	150	2.76	3.52	16.87
AMC Javelin	15.2	8	304	150	3.15	3.435	17.3
Camaro Z28	13.3	8	350	245	3.73	3.84	15.41
Pontiac Firebird	19.2	8	400	175	3.08	3.845	17.05
Ford Pantera L	15.8	8	351	264	4.22	3.17	14.5
Maserati Bora	15	8	301	335	3.54	3.57	14.6

Figure 5. Three Pages of a Report with a Page-by

Notice in the report sample above that the data has been grouped by cylinders, and a page-break inserted for each group. The page-by label is displayed between the titles and the table header.

In other packages, it may take a considerable amount of work to get such a report. With the *reporter* package, it is easy! The code is hardly any different than the previous examples. The only significant difference is the addition of a `page_by()` function. You also have to order by data by the page-by variable before sending it to the reporting functions. Here is the code:

```
library(reporter)

# Create temporary path
tmp <- file.path(tempdir(), "example5.rtf")

# Prepare data
dat <- mtcars[order(mtcars$cyl), ]
dat <- data.frame(vehicle = rownames(dat), dat)

# Define table
tbl <- create_table(dat, show_cols = 1:8) %>%
  page_by(cyl, label="Cylinders: ")

# Create the report
rpt <- create_report(tmp, orientation = "portrait", output_type ="RTF") %>%
  titles("Listing 3.0", "MTCARS Data Listing with Page By") %>%
  add_content(tbl)

# Write the report
write_report(rpt)

# View the report
file.show(tmp)
```

CREATE A FIGURE

In addition to listings and tables, the *reporter* package can create figures. A figure is accomplished using the popular *ggplot2* package, and adding the plot to the report with the `create_plot()` and `add_content()` functions. Here is what a figure looks like:

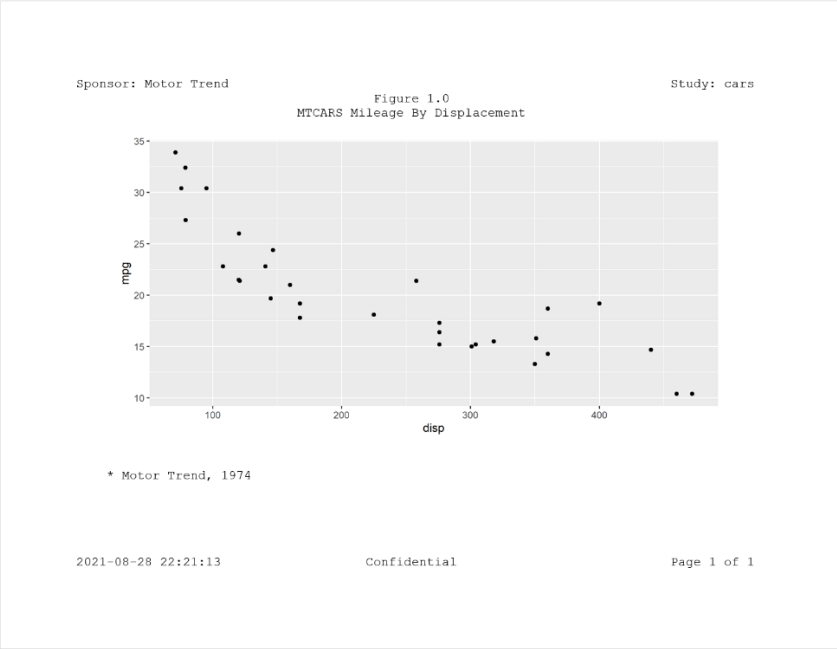


Figure 6. Example of a Figure

This figure is generated very much the same way as the tables and listings. You follow the same basic steps:

1. Create the plot content
2. Create the report and add the content
3. Write the report

Here is the code that generates the above figure:

```
library(reporter)
library(ggplot2)

# Create temporary path
tmp <- file.path(tempdir(), "example6.rtf")

# Generate plot
p <- ggplot(mtcars, aes(x=disp, y=mpg)) + geom_point()

# Define plot object
plt <- create_plot(p, height = 4, width = 8) %>%
  titles("Figure 1.0", "MTCARS Mileage By Displacement",
         blank_row = "none") %>%
  footnotes("* Motor Trend, 1974")

# Add plot to report
rpt <- create_report(tmp, output_type = "RTF") %>%
  set_margins(top = 1, bottom = 1) %>%
```

```
  options_fixed(font_size = 12) %>%
  page_header("Sponsor: Motor Trend", "Study: cars") %>%
  add_content(plt) %>%
  page_footer(Sys.time(), "Confidential", "Page [pg] of [tpg]")

# Write out report
write_report(rpt)

# View the report
file.show(tmp)
```

CONCLUSION

The **reporter** package was designed to overcome the limitations of existing R reporting packages. It allows you easily create data listings and many styles of tables and figures. The package requires no knowledge of *RMarkdown*, *LaTeX*, *pandoc*, or any other technology. It is the only reporting package with the intelligence to break/wrap your data when needed to prevent pages overflows. The code interface is intuitive and easy to use. Full documentation of the package and more examples are available online at reporter.r-sassy.org.

The *reporter* package is part of a system of packages called **sassy**. This system replicates many ideas from SAS® software into R. For more information, see the *sassy* site at r-sassy.org.

CONTACT INFORMATION

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

David J. Bosak
r-sassy.org
269-870-7343
dbosak01@gmail.com