PharmaSUG 2018 - Paper HT-04

A Gentle Introduction to R From A SAS Programmer's Perspective

Saranya Duraisamy, Nathan Mockler, Biogen

ABSTRACT

Just as a screwdriver cannot hammer in a nail, no single programming language can solve every single problem you will encounter in your programming career. Like a good craftsman, a skilled statistical programmer should know many techniques for analyzing and visualizing complex clinical data. While SAS has impressive analytic capabilities, it has its own limitations when it comes to complex analytics and visuals. That is why it is prudent to have another tool in your toolbox. R – an open source language, has made enormous strides in the past few years in the areas of data science and graphics. This talk will serve as an introduction of R's analytic capabilities including basic data manipulation techniques and the grammar of graphics that serves as the basis of the popular "ggplot2" capability. This will also be discussed from the perspective of a SAS programmer wanting to learn R as an additional language..

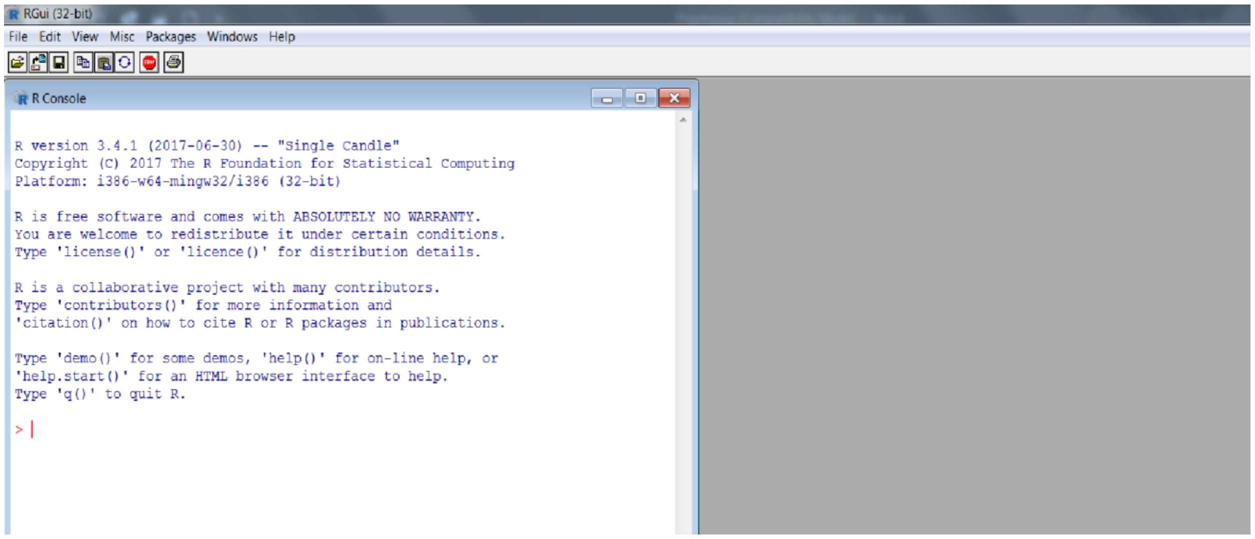
INTRODUCTION

No carpenter would want to be stuck with 1 tool; no band would want to be solely with 1 instrument. A well-rounded statistical analyst should be familiar enough with both languages in order to work with whatever problem comes their way. Especially with the rise of "big data", and "machine learning", at least having a base level knowledge means that if an opportunity comes, you can quickly dive in.

TALK LIKE A PIRATE DAY

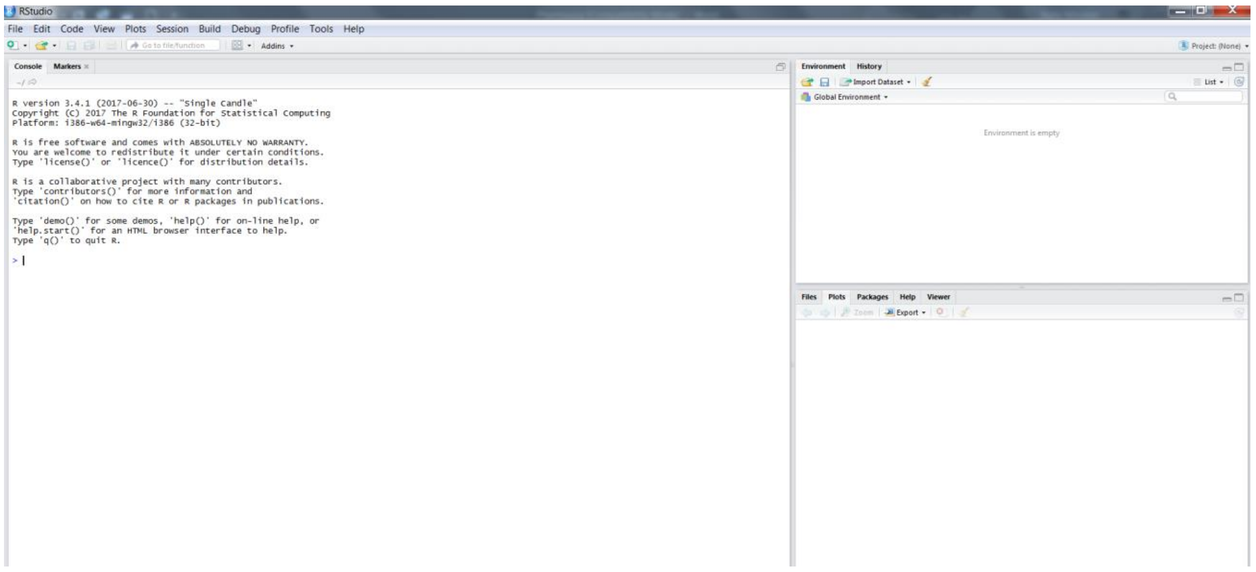
R is an open-source language. That means not only is it free as in beer, it's free as in speech. You could go, get the raw code and compile it yourself, not that I recommend that. It's a collaborative enterprise between academics, people in industry, and hobbyist determined. CRAN (The Comprehensive R Archive Network) is the home of the binaries, and the thousands of packages contained within.

Looking at the command-line can be intimidating, especially for people who have grown up with the comfort of the SAS Enhanced Editor. Fortunately (this will be a pattern), someone has realized this problem and come up with a solution. There are numerous GUIs that will allow you to view workspaces, submit code, and debug much easier. The most popular one at this time is RStudio, which is under very active development. Plus, it is open-source as well:



```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Output 1: Command line! Ahh, scary!



```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Output 2: Phew, that's better.

One of the biggest differences between SAS and R is that R is an object-orientated language (Technically, SAS has an object-oriented part, but that's going to get complicated). SAS is more of a procedural language. In SAS, the only object that you interact with is a dataset. The dataset is comprised of observations and variables. Every PROC interacts with a dataset, and if you want to manipulate the output, you're going to have to output it to a dataset. In R, that's not the case.

There are vectors, matrices, lists (that can contain lists within it), and the closest analogue to a dataset, the data.frame. We'll be using these mostly for the remainder of this, but the ecosystem of R is much larger than SAS.

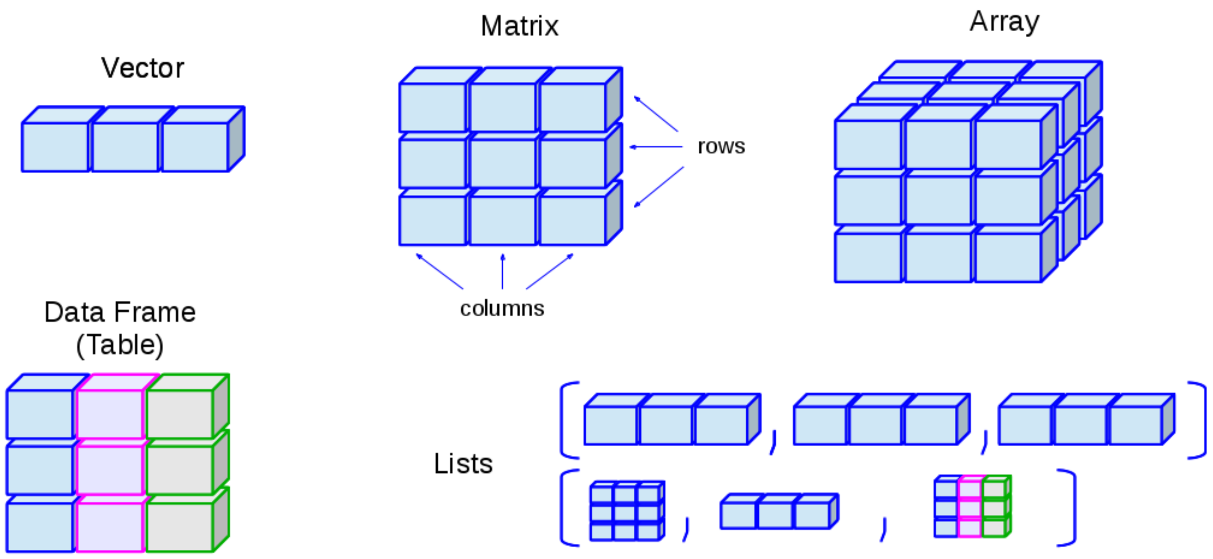


Figure 1: Objects in R (Each color corresponds to a type)

Figure 2: Objects in SAS (That's a dataset)

EXERCISE 1: GETTING YOUR FEET WET

Your mission, should you choose to accept it, is that you're a hotshot SAS programmer for Hooli Pharmaceuticals. You're living the elite SAS Programmer lifestyle: yachts, big corner office, etc, but the boss wants to look at some data, and the SAS license is expired. This is a tight deadline, so you'll have to use R. You've never used it before, but an experienced programmer like you should be able to pick it up fast, right? First, import the sas7bdata DM dataset and take the mean of age.

THE LIFE CHANGING MAGIC OF TIDY-ING UP

Of course, since R is open-source, it gives the opportunity for individuals to create packages that greatly extend the functionality. That leads us to Hadley Wickham, chief scientist at RStudio, and professor at Auckland University. One article called him "The Man who Revolutionized R"

https://priceonomics.com/hadley-wickham-the-man-who-revolutionized-r/

His compilation of packages is called the "tidyverse":

- ggplot2, for data visualisation.
- dplyr, for data manipulation.
- tidyr, for data tidying.
- readr, for data import.
- purrr, for functional programming.
- tibble, for tibbles, a modern re-imagining of data frames.

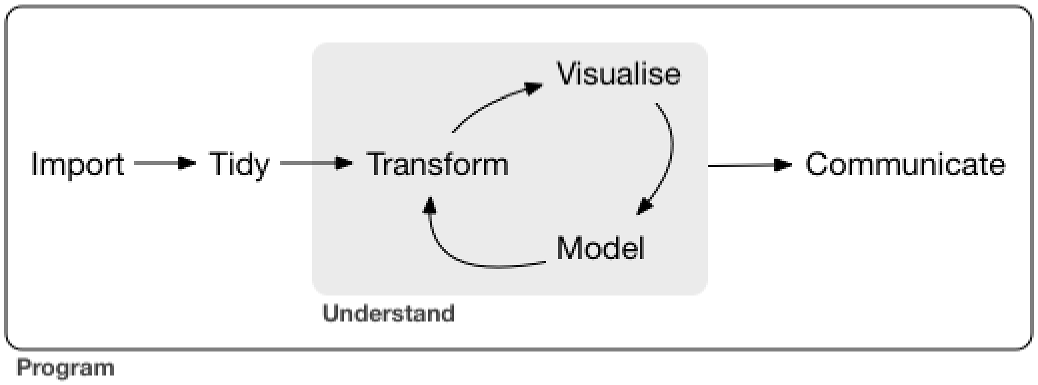


Figure 3: The Project Loop – tidy style (Wickham, 2017)

Wickham describes Tidy data as:

Tidying your data means storing it in a consistent form that matches the semantics of the dataset with the way it is stored. In brief, when your data is tidy, each column is a variable, and each row is an observation. Tidy data is important because the consistent structure lets you focus your struggle on questions about the data, not fighting to get the data into the right form for different functions. (Wickham, 2017)

Of course, coming from SAS, most of our data is in this format, but dealing with data such as JSON, XML, web scraping, etc won't be, so we need to get the data in this format.

Some of these packages that we'll explore are critical in getting the data in the format to analyze, and most R users these days use at least one of his packages. The packages we'll discuss are:

Tidyr: Just in case you missed PROC TRANSPOSE, it's here, with commands such as gather() and spread()

Ggplot2: We'll talk about this more in the next section, but it allows you to create complicated visualizations that you are unable to in SAS or even base R

Dplyr: Allows you subset observations, and rows, summarize, make new variables and combine data sets.

One of the best things about the tidyverse is the pipe, which allows you to string functions together without having to a) Create some nesting doll nightmare of parentheses or b) Create a lot of intermediate steps that can make the code unreadable. The key is that the object on the left is the first/only argument on the right

For example,

```
iris %>%
group_by(Species)
```

is equivalent to:

```
group_by(Iris, Species)
```

See how that's more readable? The benefits start to become more evident once you string multiple ones together.

```
car_data <-
  mtcars %>%
  subset(hp > 100) %>%
  aggregate(. ~ cyl, data = ., FUN = . %>% mean %>% round(2)) %>%
  transform(kpl = mpg %>% multiply_by(0.4251)) %>%
  print
```

Is equivalent to:

```
car_data <-
  transform(aggregate(. ~ cyl,
                      data = subset(mtcars, hp > 100),
                      FUN = function(x) round(mean(x, 2))),
            kpl = mpg*0.4251)
```

Which is much harder to read and debug.

One of the more one-to-one equivalents are PROC TRANSPOSE. This is done by the spread and collapse function.

If you're a fan of PROC SQL for joining, you're in luck, since dplyr has various join functions:
Left_join (a, b, by = "x1") is equivalent to PROC SQL;
Select * from a left join b on a.x1 = b.x1. The other types of joins (right, inner, outer) work accordingly.

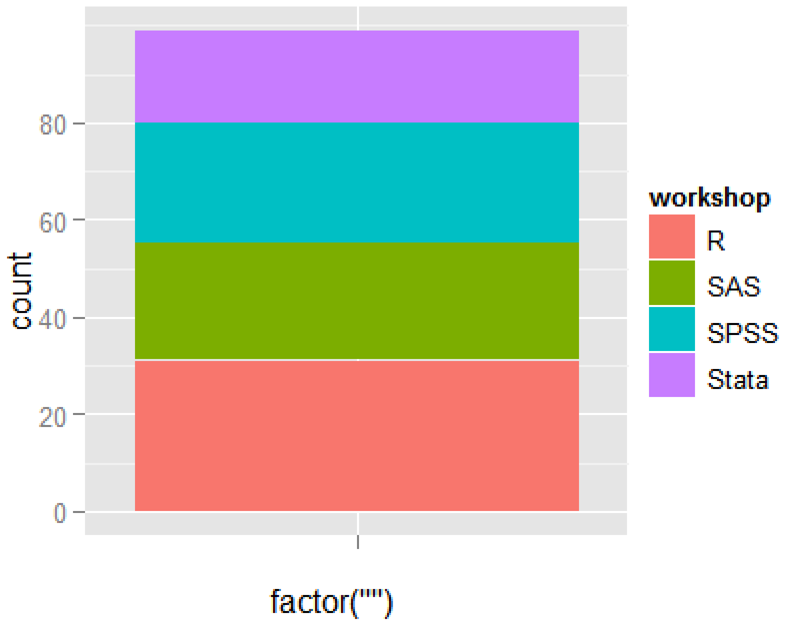
EXERCISE 2: STARTING TO PUT IT TOGETHER

Now for this exercise, we want you to combine two of the data sets – a demographics and Lab Data set, but it's going to be tricky. We want to create a lab data set over time data set merged with demographics… create a high lab value indicator variable, and then get a frequency of that high lab.

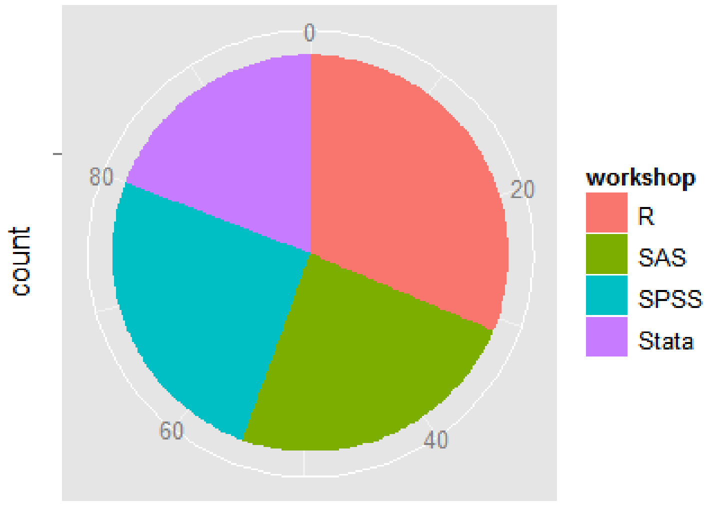
THE PLOT THICKENS

Now onto what many consider R's greatest strengths, its ability to do data visualization. While SAS can do a lot of visualizations, it can be limited even with using GTL. R has primarily 3 graphics packages: base, lattice, and ggplot2. For this exercise, we're going to focus on ggplot2.

GGPLOT2 is a package part of the tidyverse based on the "Grammar of Graphics". The grammar of graphics basically puts all visualization in a common language. It allows you to take a plot such as this



Output 3: A Stacked Bar Chart



Output 4: A Pie Chart; we're not so different, you and I

What the grammar of graphics shows that these are the exact same plot, and to change this only take 1 option change (from linear to a polar scale). An exploration of the grammar of graphics is beyond the scope of this paper, but I recommend exploring the ggplot2 documentation. The concept behind ggplot2 is based on stacking elements on top of each other, like a sandwich:

Figure 4: Delicious and Illustrative!

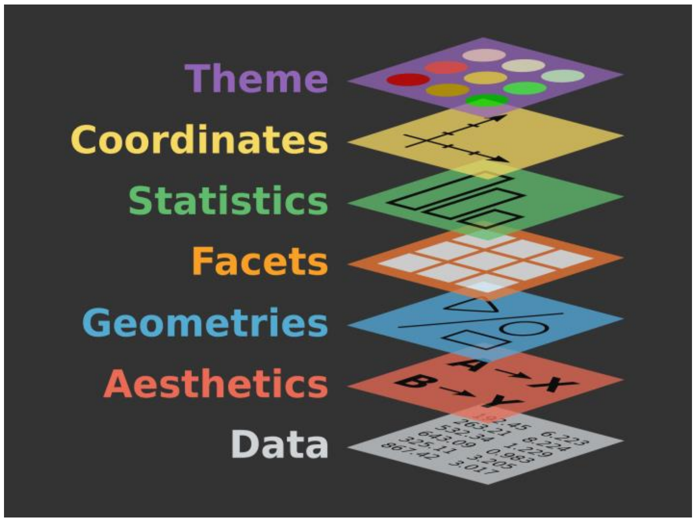
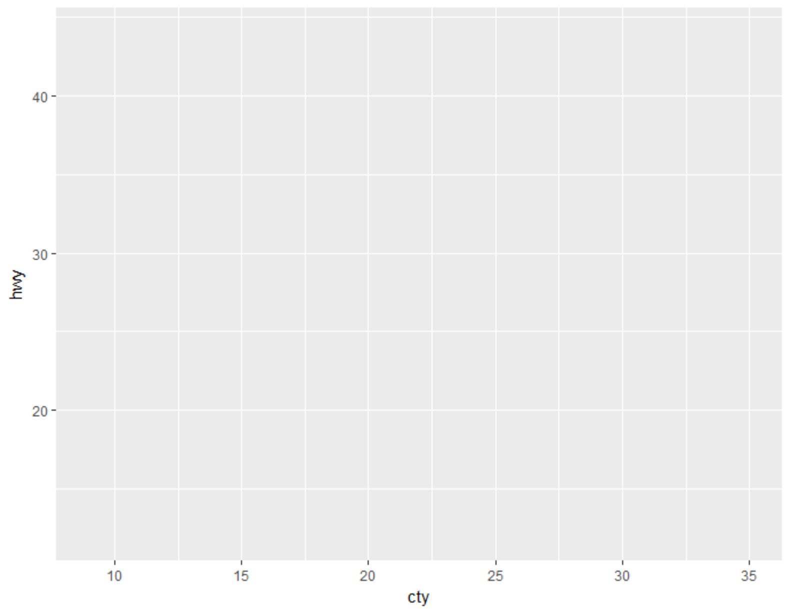


Figure 5: Less edible, but slightly more informative.

So now we're going to stack a graph together. This is in contrast to SAS, where all your options are done at once.

Let's start with a basic graph:

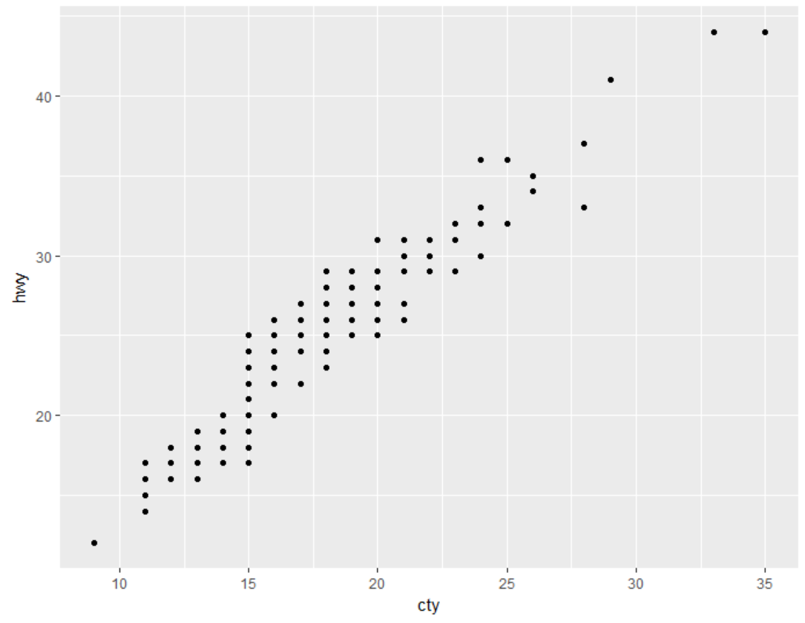
```
g <- ggplot(data = mpg, aes(x = cty, y = hwy))
```



Output 5: Not Much to look at, yet…

Easy enough, right? This creates a graph object that we can modify. Now let's add points:

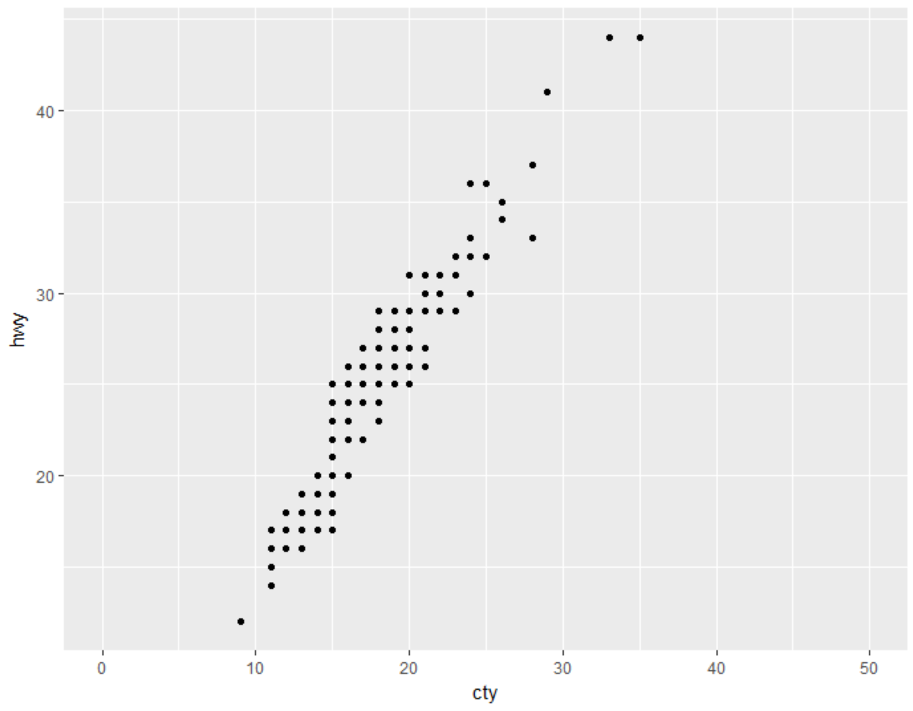
```
g <- ggplot(data = mpg, aes(x = cty, y = hwy)) + geom_point()
```



Output 6: Getting There….

Now let’s say we want to modify the x axis to be from 0 to 50. Rather than modify the original statement, we add a scale statement:

```
g <- ggplot(data = mpg, aes(x = cty, y = hwy)) + geom_point() +
scale_x_continuous(limits = c(0,50))
```

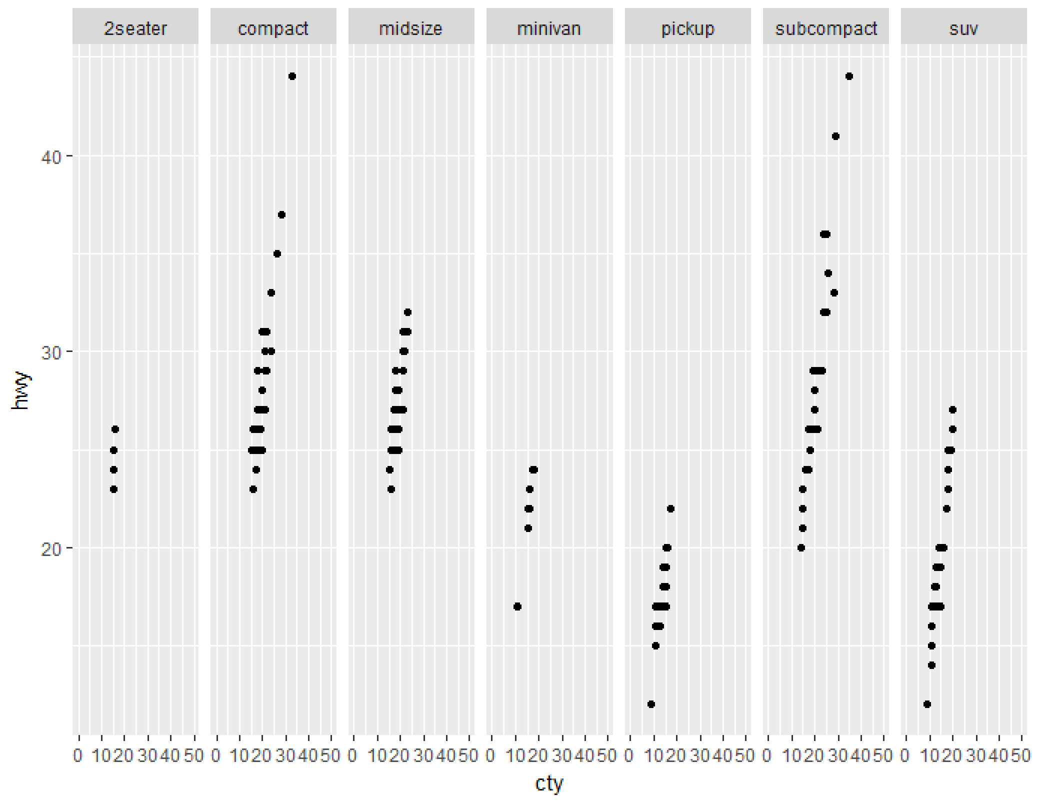


Output 7: Almost….

Okay, that sounds good. Now let’s say we want to create a separate graph (the equivalent of a by statement) for each type (the variable is class) of car. We can stack on that stack, and do:

```
g <- ggplot(data = mpg, aes(x = cty, y = hwy)) + geom_point() +
scale_x_continuous(limits = c(0,50)) + facet_wrap(. ~ class)
```

To get our final graph:



Output 8: Finally, our graph!

There is also work we can do with themes, legends, etc… using the same concepts. Please see the documentation for more information.

EXERCISE 3: THE PLOTS THE THING

Taking that data set you generated in Exercise 2, let's graph some of the work. Please graph lab value over time, adding labels, legends, and output to pdf.

CONCLUSION

This is but a taste of what R can do. From machine-learning to interactive dashboards, R has a lot of uses. Combined with the fact that it is free and open-source, as well as the fact anyone can contribute a package means whatever your question is, there probably is an R package that will do it. Any SAS Programmer should be at least be familiar with how to use it to add their tool belt..

REFERENCES

Muenchen, Robert A. 2011. *R for SAS and SPSS Users*. New York, NY :Springer

Priceanomics. 2015. "Hadley Wickham, the Man Who Revolutionized R." Accessed February 1, 2018. https://priceonomics.com/hadley-wickham-the-man-who-revolutionized-r/

Wickham, Hadley and Garret Grolemund. 2017. *R for Data Science*. Sepastopol, CA: O'Reilly Media, Inc.

CONTACT INFORMATION

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

Nate Mockler
Nate.Mockler@biogen.com

Saranya Duraisamy
saranya.duraisamy@biogen.com