STAT216 Activity 1: Intro to RStudio

Synopsis

The goal of this lab is to introduce you to R and RStudio, which you'll be using throughout the course both to learn the statistical concepts discussed in the textbook and also to analyze real data and come to informed conclusions. To straighten out which is which: R is the name of the programming language itself and RStudio is a convenient interface.

As the labs progress, you are encouraged to explore beyond what the labs dictate; a willingness to experiment will make you a much better programmer. Before we get to that stage, however, you need to build some basic fluency in R. Today we begin with the fundamental building blocks of R and RStudio: the interface, reading in data, and basic commands.

Getting started

Follow these few steps to get started:

Step 1. If you haven't already done so, create a free account at Rstudio.cloud using your FVCC email address. (if the link doesn't work, right click and select "open in new tab")

Step 2. In the upper right corner of the page, select "New Project," then "New RStudio Project." Call your project something evocative like "STAT216." We'll work on all of our activities in this project.

Step 3. Now you see an RStudio workspace with three windows in it: Console, Environment, and a Files. The console window is a place to perform quick calculations and install/load packages. An R package is a collection of commands, datasets, etc. that someone in the R community has authored to make our lives easier. In this class we will use two main packages, the "tidyverse" and "openintro." The tidyverse package will help us manage data easily and the openintro package gives us access to all the datasets in our textbook (among other things). In the console type:

```
install.packages("tidyverse")
```

then hit "enter." Enter evaluates the commands in the console. Now evaluate

```
install.packages("openintro")
```

Step 4. After you install a package, you always have to load it. This is done with the following commands:

```
library(openintro)
library(tidyverse)
```

Step 5. You're going to complete this lab using an R Markdown file. Download the file ***STAT216_A1_lab_report.Rmd*** from Eagle Online. You can find this file in the Activity 1 assignment folder or [BLAHBLAHBLAH]

Step 6. In the file window of your RStudio workspace, you'll see an "Upload" button. Select "Choose file" and find ***STAT216_A1_lab_report.Rmd.*** Select this file then hit OK.

Step 7. You should now see this file listed in the file window. Click the file to open your lab report.

Step 8. Watch the following two videos (also linked in Eagle Online) about using R Markdown for lab reports :

- Why we're using R Markdown

- You can start watching this video from about 40 seconds on. Using R Markdown for Lab Reports

Once you've completed these tasks, you should be ready to start working on your lab report!

RStuio as a big calculator

At its most basic, R is just a fancy, big, computer based calculator. For instance, we can calculate the following:

```
2 + 2
```

```
## [1] 4
```

```
2^127 -1
```

```
## [1] 1.701412e+38
```

```
sqrt(9)
```

```
## [1] 3
```

```
log(10, 10)
```

```
## [1] 1
```

1. In your lab report there is a section for you to make three calculations on your own. We will learn more about R as a "big calculator" as the semester proceeds, but play around with at least three calculations on your own.

RStudio and data sets

For the rest of this activity we will be working with some new car data from 1993. This data set is called `cars93`. To start with, evaluate

```
?cars93
```

in the ***console.*** Note that it brings up a description of the data set and gives some examples. In general, a question mark before a data set or function will bring up the help documentation for the object in question.

The 'head' command below displays the first 6 observations in the data set.

```
head(cars93)
```

```
## # A tibble: 6 x 6
##   type    price mpg_city drive_train passengers weight
##   <fct>   <dbl>    <int> <fct>            <int>  <int>
## 1 small    15.9       25 front                5   2705
## 2 midsize  33.9       18 front                5   3560
## 3 midsize  37.7       19 front                6   3405
## 4 midsize  30         22 rear                 4   3640
## 5 midsize  15.7       22 front                6   2880
## 6 large    20.8       19 front                6   3470
```

Note that we can view the data set like in Excel, but we cannot edit the data set. This takes a little getting used to, but ends up preventing many errors down the road.

2. Answer the following questions:
    - How many variables are there in this data set?
    - What are they and what type are they?
    - How many observations are there in the data set? Hint: The function `dim` will help.

Data analysis: categorical variables

Let's suppose first that we want to analyze the `type` variable in `cars93.` This variable is categorical, so we may firs be interested in the values it takes on and to do so without staring at the actual dataset. You can do exactly that with the following code.

```
unique(cars93$type)
```

```
## [1] small   midsize large
## Levels: large midsize small
```

Here note that the `$` calls only the `type` variable from the dataset.

Now let's figure out the frequency of each of the 3 values of `type.` The function `summary` will do exactly this for us.

```
summary(cars93$type)
```

```
##   large midsize   small
##      11      22      21
```

Similarly, we can look at the proportion of each type of car by dividing the results of the `summary` command by the `length` of the data vector:

```
summary(cars93$type)/length(cars93$type)
```

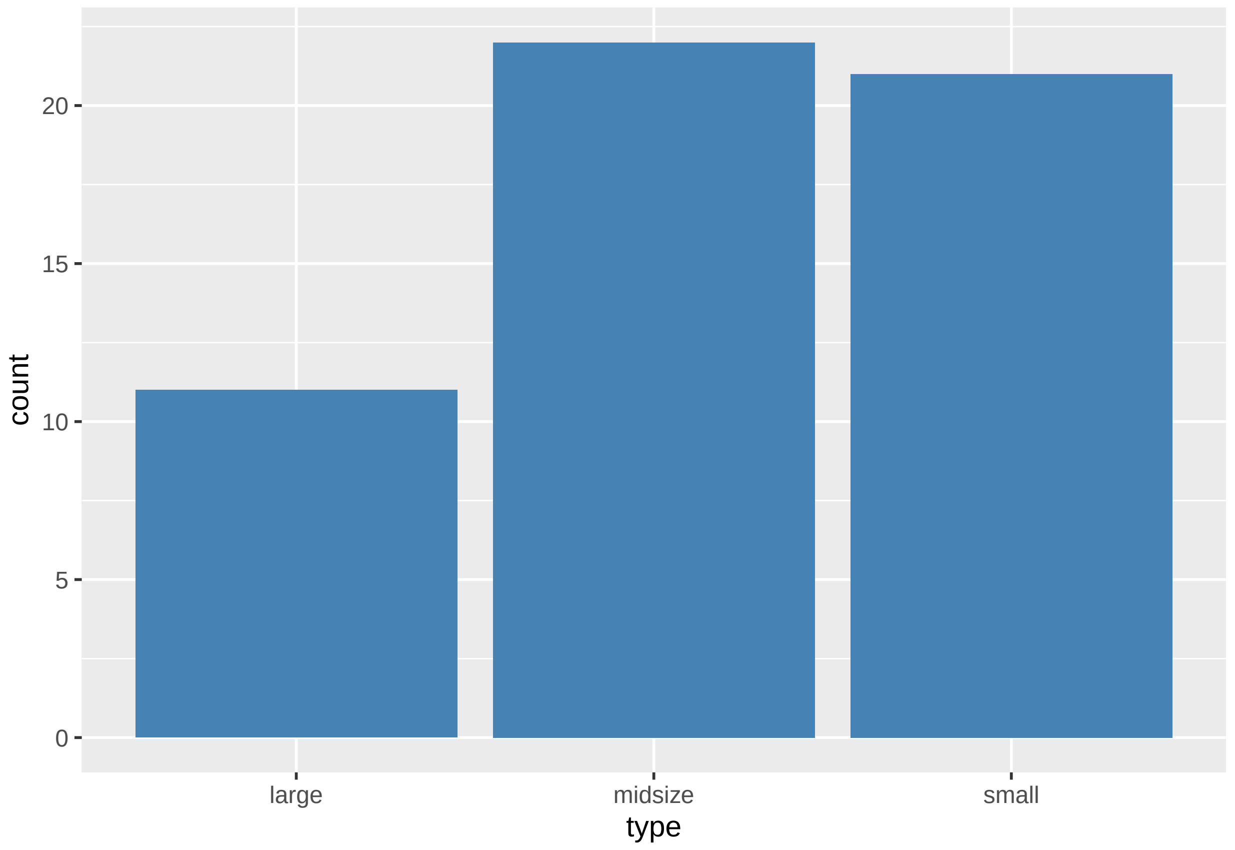
```
##     large   midsize     small
## 0.2037037 0.4074074 0.3888889
```

3. - What types of drive trains do the cars in this data set have?
   - How many cars in this data set are rear wheel drive?
   - What proportion of cars in the data set are 4WD.

Visualizations: categorical variables

Visualizing data using barcharts, histograms, scatterplots, etc. can be essential for data exploration and communication. Fortunately, RStudio makes many visualizations easy to construct. As an example, let's see how to construct a bar chart to visualize the numbers of each type of car in the `cars93` dataset.

```
p <- ggplot(cars93, aes(x=type)) + geom_bar(stat="count", fill="steelblue")
p
```



This example brings up at least 3 important points here.

Point 1. The `<-` arrow defines `p` to be whatever is to the *right* of it. As another example `x <- 10` assigns to the variable `x` the value of 10.

Point 2. In this case, we won't go into the details too much about `ggplot` is, but `ggplot(cars93, aes(x=type))` defines `p` to be a plot with the explanatory variable as `type` and `+ geom_bar(stat ="count", fill="steelblue")` specifies a bar chart with the count of each category as the height (`fill` is just the color of the bars). The `+` symbol essentially adds different layers to the plot.

Point 3. We have to include the `p` on the second line when running this code. The first line *defines* `p`, but we have to actually call `p` again on the second line in order to print the graph.

4. Mimic the plot above to create a bar chart to visualize the number of cars in the data set with each type of drive train.

Data analysis: numerical variables

Now let's analyze the `price` variable from `cars93`, noting that this is a continuous numerical variable and that this gives the price in thousands of dollars. `R` can calculate many summary statistics for us quickly and compactly. For example to compute the mean and standard deviation of the price, we can run the following:

```
mean(cars93$price)
```

```
## [1] 19.99259
```

```
sd(cars93$price)
```

```
## [1] 11.50645
```

It is important to note here that the standard deviation is that of a sample, meaning that the denominator is $n-1$.

The `summary` function also applies to numerical variables and returns the min/max values, the quartiles, and the mean.

```
summary(cars93$price)
```

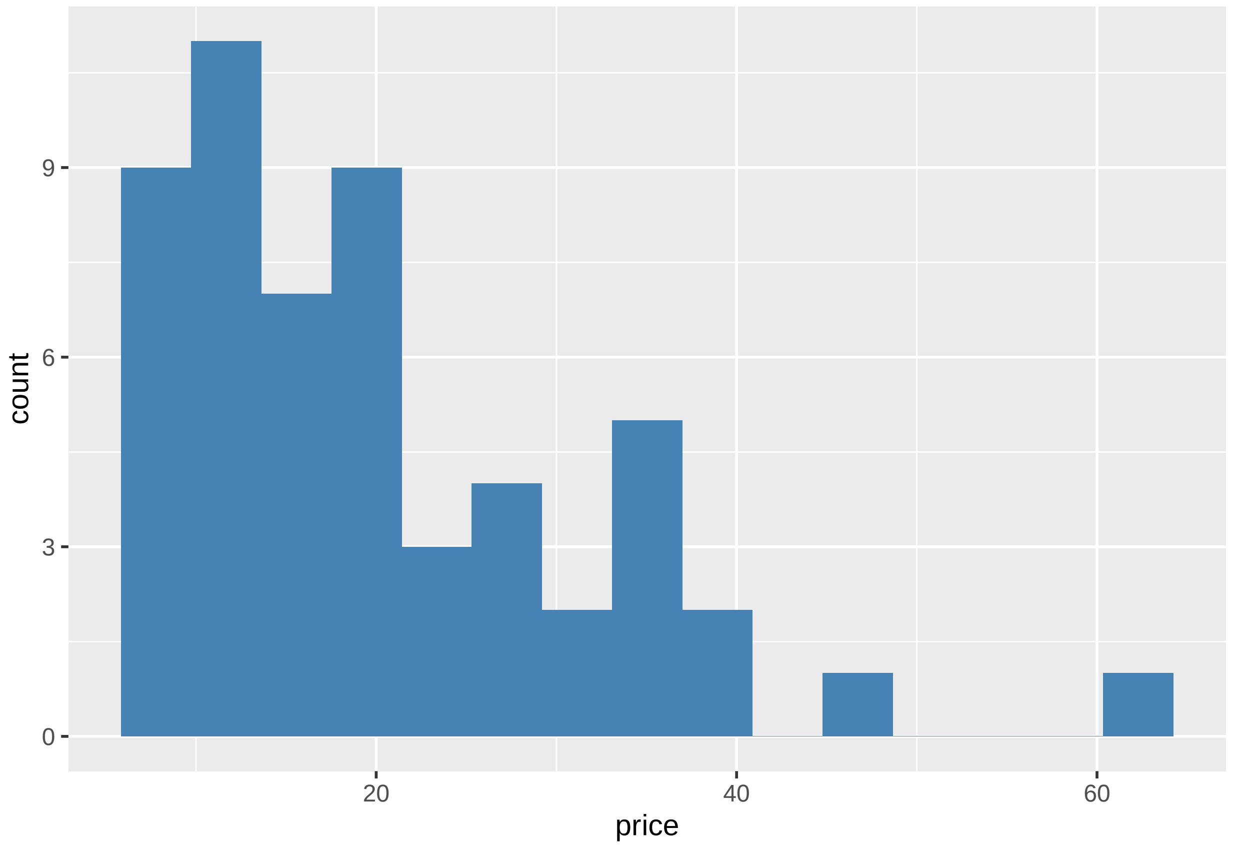
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7.40   10.95   17.25   19.99   26.25   61.90
```

5. Compare the mean and median weight of cars in `cars93`. Is there a difference between these measures of center? If so, why do you think that is the case?

Visualizations: a single numerical variable

We may want to visualize the shape or distribution of a single numerical variable. The most standard way to do this is with a histogram. Here, we'll investigate the distribution of the `price` variable.

```
p <- ggplot(cars93, aes(x=price)) + geom_histogram(bins = 15, fill = "steelblue")
p
```



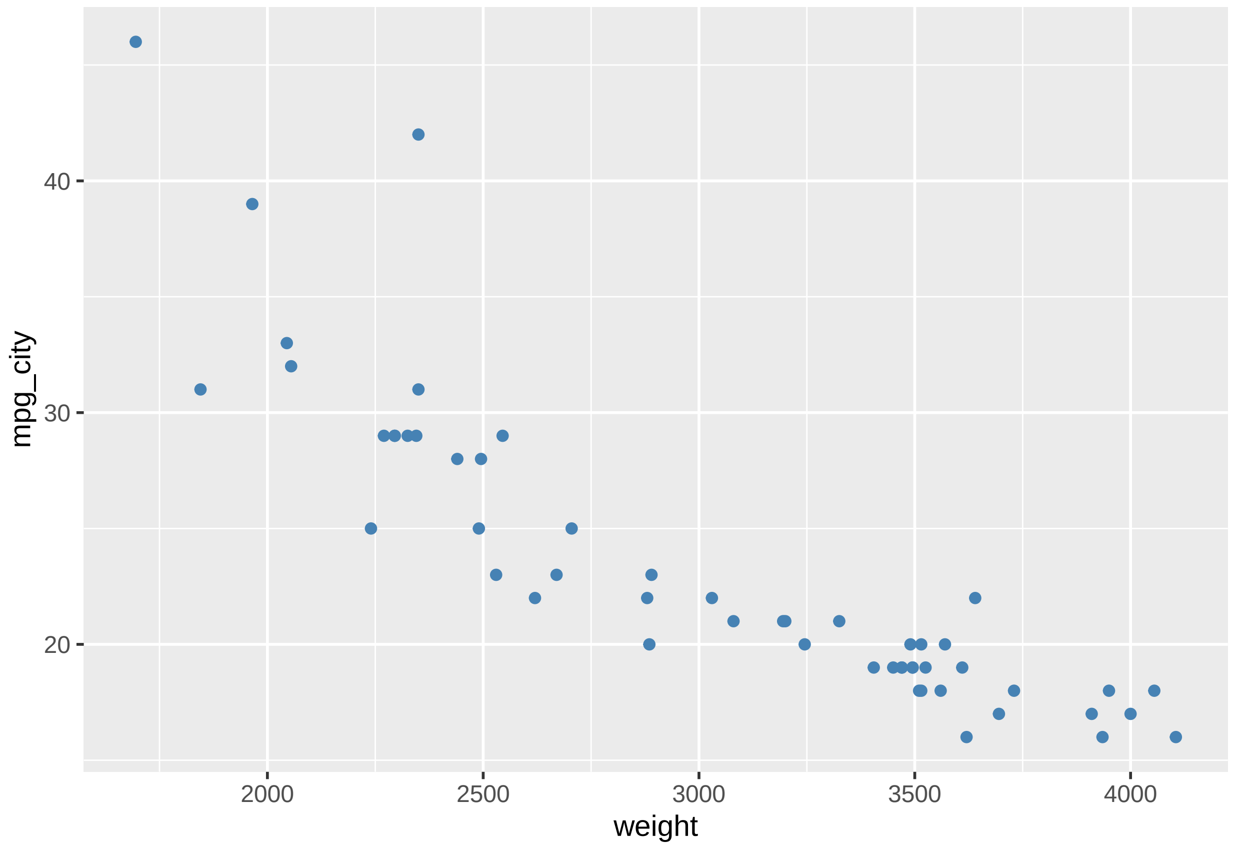
In the code above `geom_histogram` told RStudio to make a histogram and `bin = 15` specified there to be a total of 15 bins. The program makes the bid width calculation automatically. If you do not specify the number of bins, the default value is 30.

6. Construct a histogram to visualize the distribution of the `weight` variable. How would you describe this distribution?

Visualizations: 2 numerical variables

We will often want to explore relationships between multiple variables. For instance, as the weight of a car increase, we suspect the fuel efficiency to decrease. But how exactly does the fuel economy decrease? We can begin to study this question by looking at a scatter plot that has `weight` as the explanatory varibale and `mpg_city` as the response variable. The code below constructs such a plot for us:

```
p <- ggplot(data=cars93, aes(x = weight, y = mpg_city)) + geom_point(color="steelblue")
p
```



As with the plots before this one, you can change the aesthetics of your plots quite a bit. We may spend more time on this later in the semester, but for now this plot demonstrates that our suspicion was correct: as the weight of a vehicle incearses, the fuel economy decreases.

7. Is there any evidence of a correlation between the price of vehicle and it's fuel economy? If so, how would you describe this relationship?

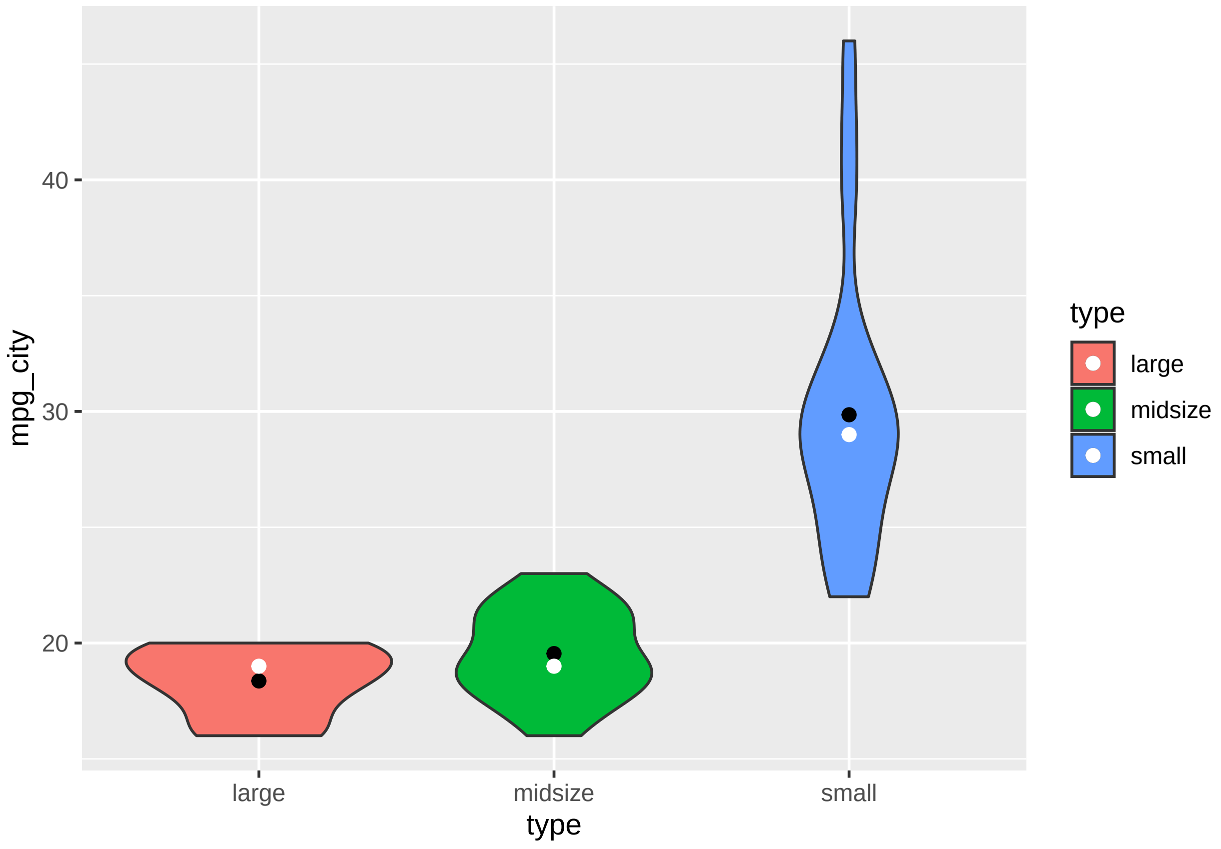
Visualizations: A categorical and numerical variable

We may want to investigate the relationship between a categorical variable and a numerical variable. For instance, we may want to compare fuel economy across the different types of cars in the data set.

There are a variety of ways we could do this. Box plots are historically the main visualization for comparing numerical variables across categories, but there are better options available now. One such option is a **violin plot**.

The next code chunk will generate violin plots to compare `mpg_city` by `type`.

```
p <- ggplot(data=cars93, aes(x = type, y = mpg_city, fill = type)) +
  geom_violin() +
  stat_summary(fun=mean, geom="point",  size=2, color="black") +
  stat_summary(fun=median, geom="point",  size=2, color="white")
p
```



We will not explicity describe what a violin plot shows (see Exercise 8 below), but note the layering in the plot above. Each of the lines beginning with `stat_summary(...)` adds a layer to the plot. In particular, the black dot is the mean value and the white dot gives the median value.

8. Describe, in your own words, what a violin plot shows. You may want to look this up; feel free to simply search "violin plot" on the web and see what you find.

Resources for learning R and working in RStudio

That was a short introduction to R and RStudio, but we will provide you with more functions and a more complete sense of the language as the course progresses.

In this course we will be using the suite of R packages from the **tidyverse**. The book R For Data Science by Grolemund and Wickham is a fantastic resource for data analysis in R with the tidyverse. If you are Goggling for R code, make sure to also include these package names in your search query. For example, instead of Goggling "scatterplot in R", Goggle "scatterplot in R with the tidyverse".

These may come in handy throughout the semester:

- RMarkdown cheatsheet
- Data transformation cheatsheet
- Data visualization cheatsheet

Note that some of the code on these cheat sheets may be too advanced for this course. However the majority of it will become useful throughout the semester.

The first and last sections of this activity, along with the overall template, were taken from this Openintro Stats Lab.