Assignment7

Chase Darlington

October 17, 2018

BSDS 100: Intro to Data Science with R

Assignment 7

by Chase S Darlington (University of San Francisco)

Directions: For all questions in this assignment, write complete sentences and fully answer any question that is asked, and use R to answer each question. Provide all R code and solutions by knitting your final RStudio file into a single file. Late assignments will automatically have 10 points deducted. Assignments turned in after the answer key is posted (one week after the due date) will not receive any points. Assignment: For this assignment, please do the following:

1. (8 pts) We have now discussed several libraries from the tidyverse, including tibble, ggplot2, tidyr, and dplyr. Select two and describe what they do, why you might use them, how to use one function from each, and why you might use that function.

- Tidyr
- makes it easy to "tidy" or clean data
- for example, you can seperate columns into 2 columns using seperate()
- another example, you can reshape messy data with gather(), which makes key-value pairs based on criteria

1.1 Tidyr tidys and cleans/reorganizes data 1.2 Tidyr::gather can be used to make key-value pairs and is normally used like this: gather(data, key [the new coded column], value(s), and exceptions). Also illustrated below. 1.3 Tidyr::gather is used to translate variables (that aren't actually variables) to values in a dataset. In general, tidyr can be used to expand & consolidate datasets to more accurately portray information.

```
library(tidyr)
library(tibble)

messydata <- data.frame(
  kids = c("Jack", "Jill", "John", "Jane", "Jake", "Jennifer"),
  X= c(7.2, 8, 9, 10, 6.3, 5.9), Y= c(6, 5.4, 4.9, 6.6, 5.5, 7.0)
)
head(messydata)
```

```
##       kids    X   Y
## 1     Jack  7.2 6.0
## 2     Jill  8.0 5.4
## 3     John  9.0 4.9
## 4     Jane 10.0 6.6
## 5     Jake  6.3 5.5
## 6 Jennifer  5.9 7.0
```

```
gather(messydata, type, X:Y, -kids)
```

```
##        kids type  X:Y
## 1      Jack    X  7.2
## 2      Jill    X  8.0
```

```
## 3      John    X  9.0
## 4      Jane    X 10.0
## 5      Jake    X  6.3
## 6  Jennifer    X  5.9
## 7      Jack    Y  6.0
## 8      Jill    Y  5.4
## 9      John    Y  4.9
## 10     Jane    Y  6.6
## 11     Jake    Y  5.5
## 12 Jennifer    Y  7.0
```

```
stocks <- tibble(
  time = as.Date('2009-01-01') + 0:9,
  X = rnorm(10, 0, 1),
  Y = rnorm(10, 0, 2),
  Z = rnorm(10, 0, 4)
)
head(stocks)
```

```
## # A tibble: 6 x 4
##   time            X      Y      Z
##   <date>      <dbl>  <dbl>  <dbl>
## 1 2009-01-01 -2.46   1.39   -7.63
## 2 2009-01-02 -0.492 -2.47   -1.25
## 3 2009-01-03  0.266 -1.18  -10.9
## 4 2009-01-04 -1.61   0.488  -1.01
## 5 2009-01-05  0.893 -3.23   -5.26
## 6 2009-01-06  0.726 -3.56    1.53
```

```
gather(stocks, stock, price, -time)
```

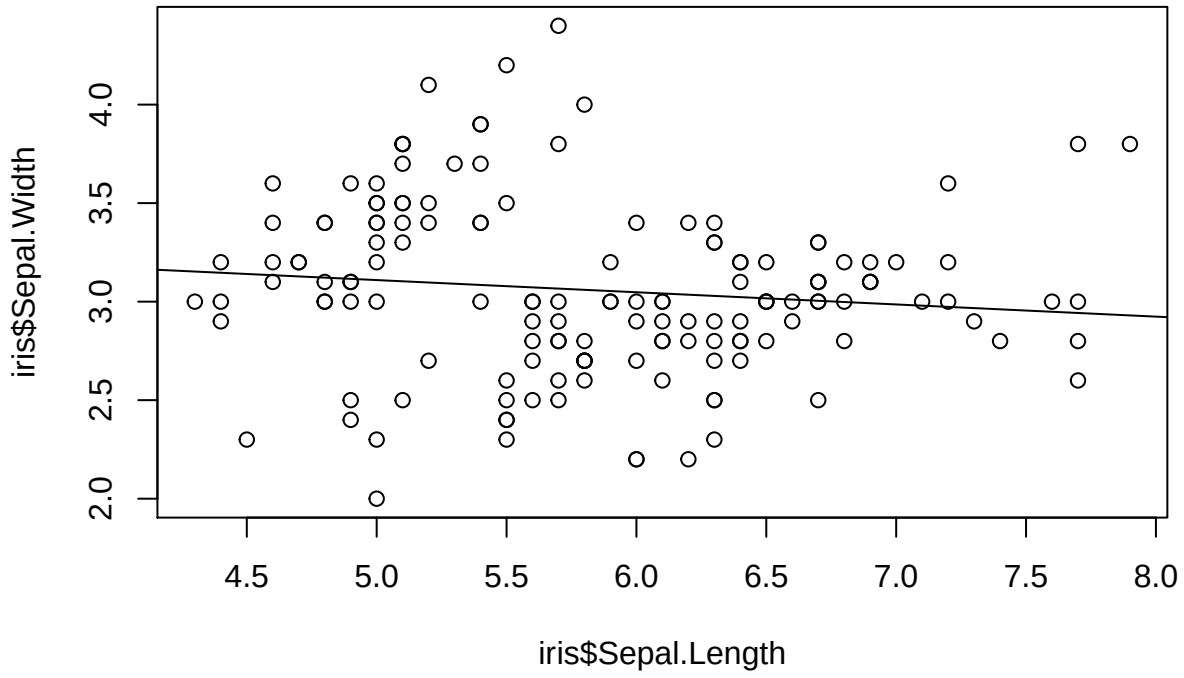
```
## # A tibble: 30 x 3
##    time       stock   price
##    <date>     <chr>   <dbl>
##  1 2009-01-01 X     -2.46
##  2 2009-01-02 X     -0.492
##  3 2009-01-03 X      0.266
##  4 2009-01-04 X     -1.61
##  5 2009-01-05 X      0.893
##  6 2009-01-06 X      0.726
##  7 2009-01-07 X      0.703
##  8 2009-01-08 X      0.204
##  9 2009-01-09 X      0.0582
## 10 2009-01-10 X     -1.52
## # ... with 20 more rows
```

- ggplot2: ggplot: a visualization function like base R "plot"
- the "plot" function is a very simple plot function that allows users to plot data on a 2D, Euclidian plane.
- the "ggplot" function, on the other hand, sets itself apart in 4 key ways. Ultimately, ggplot makes it easier to visualize data–group values, define appearance of geometric points, define color and fill, and more features are much easier to accomplish with ggplot. Notably, the same can be said of the entire ggplot2 library.
  - integration with dataframes
  - ggplot works best with dataframes, which are more common than matrices (best data type for plot function)

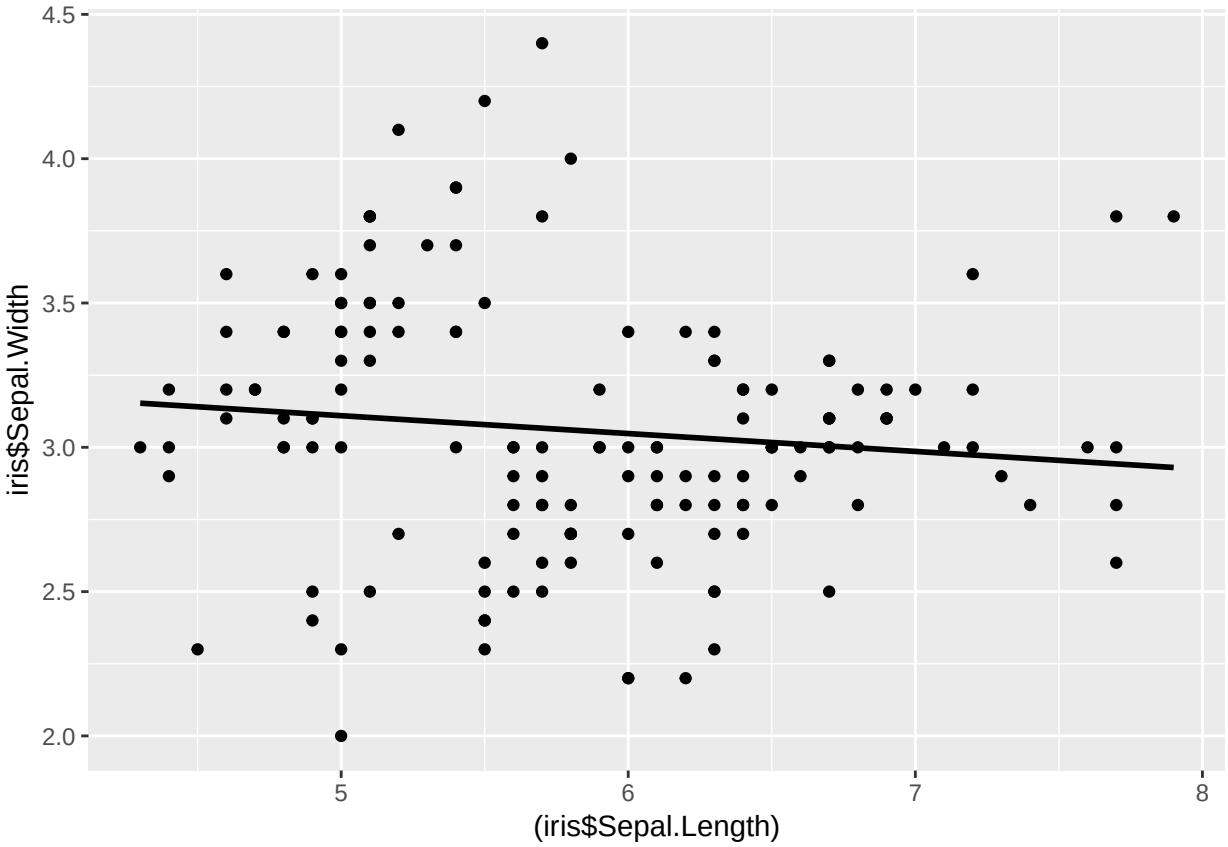
- default colors and organization
- a unique "grammar of graphics"
- one can piece together components of various visualization types into one graph/model/visual.
- concatenated with a "+"
- ggplot has built-in data management tools
- this is what allows one to group/bin values
- one would have to manipulate the data outside of "plot" in order to do the same.

```
library(ggplot2)

plot(iris$Sepal.Length, iris$Sepal.Width, abline(lm(iris$Sepal.Width ~ iris$Sepal.Length)))
```

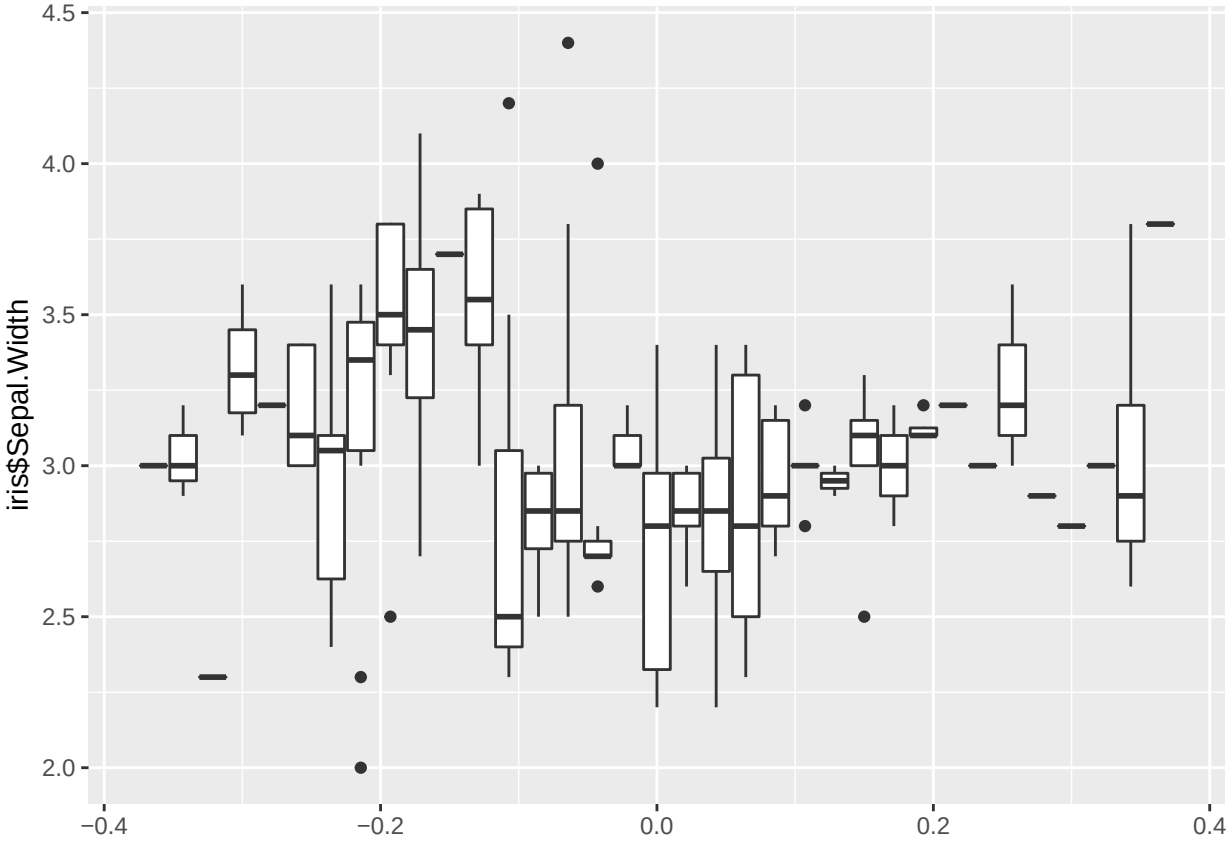



```
ggplot(iris, aes((iris$Sepal.Length), iris$Sepal.Width))+ geom_point() + geom_smooth(method="lm",se=F, c
```




```
#ggplot has out-of-the-box features (organization and colors) that plot does not
#ggplot uses "+" to combine different visualization types

ggplot(iris, aes(group=iris$Sepal.Length, y=iris$Sepal.Width)) + geom_boxplot()
```

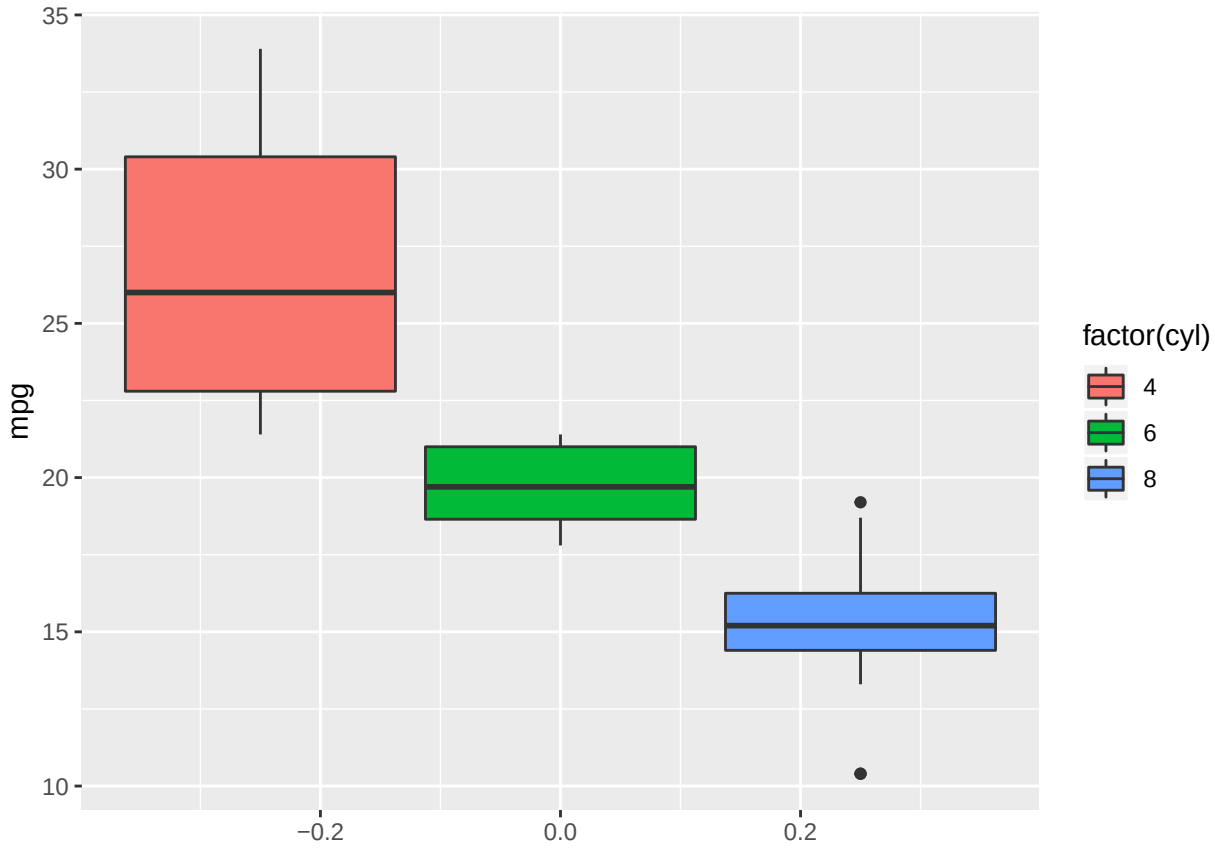



```
#and ggplot has built-in data management tools that help you group data
```

2. (4 pts) Review the notes on different types of plots. Select two types of plots and describe what they are useful for. Then describe which functions you would use to make each.

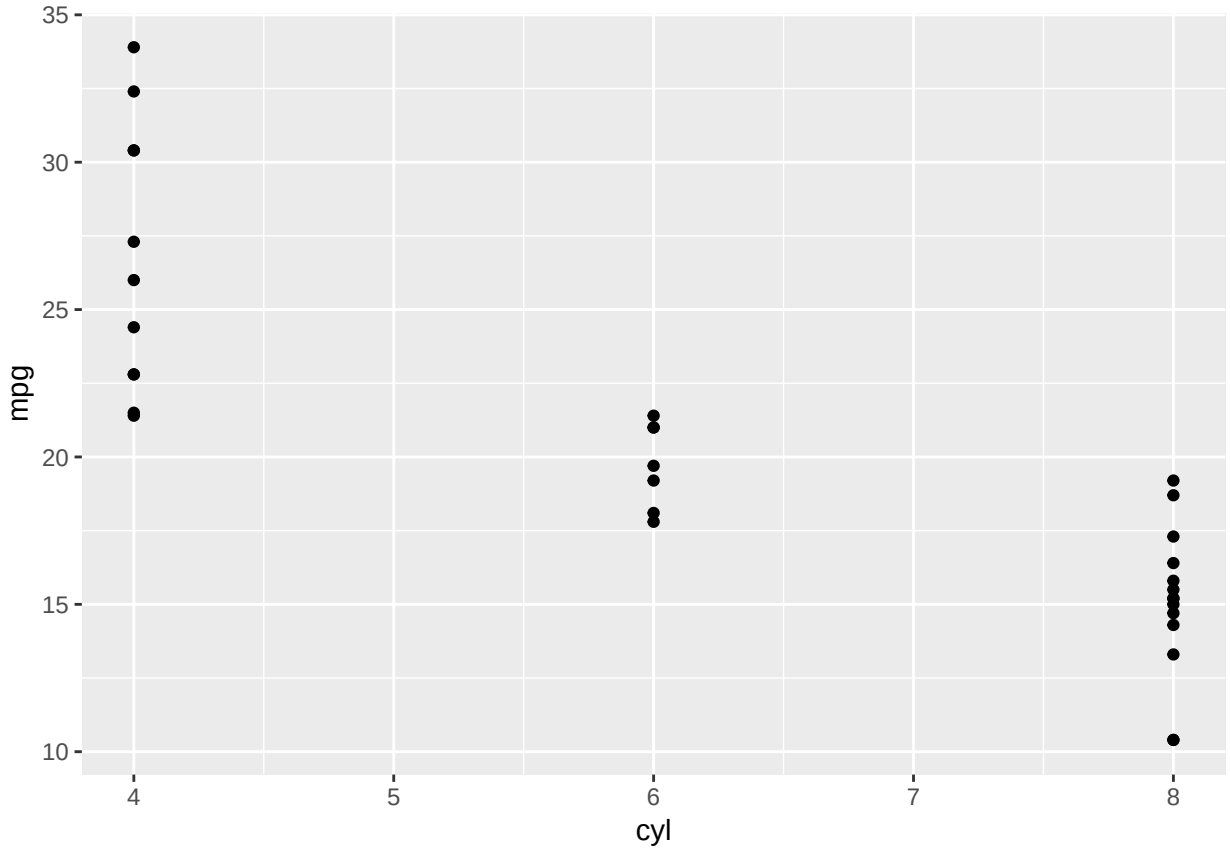
- Box Plot
- very useful when analyzing range, average, and quartile metrics.
- in ggplot, use geom_boxplot()

```
ggplot(mtcars, aes(group=factor(cyl), y=mpg, fill=factor(cyl)))+ geom_boxplot()
```



- Scatter Plot
- very useful when analyzing distribution, trends, and correlation.
- in ggplot, use geom_point()

```
ggplot(mtcars, aes(cyl, mpg)) + geom_point()
```



3. (10 pts) Load the ggplot2 library. Read Sections 3.1 through 3.5 in the R for Data Science text on Data Visualization here http://r4ds.had.co.nz/data-visualisation.html. Complete Exercises 3.2.4 numbers 1-5. You will need to include code and sentence descriptions for each of the five questions, except question 4, which only requires code.

```
library(ggplot2)


#1. Run ggplot(data = mpg). What do you see?
ggplot(data=mpg)
```

```
#A blank, white sheet

#2. How many rows are in mpg? How many columns?
nrow(mpg)
```

```
## [1] 234
```

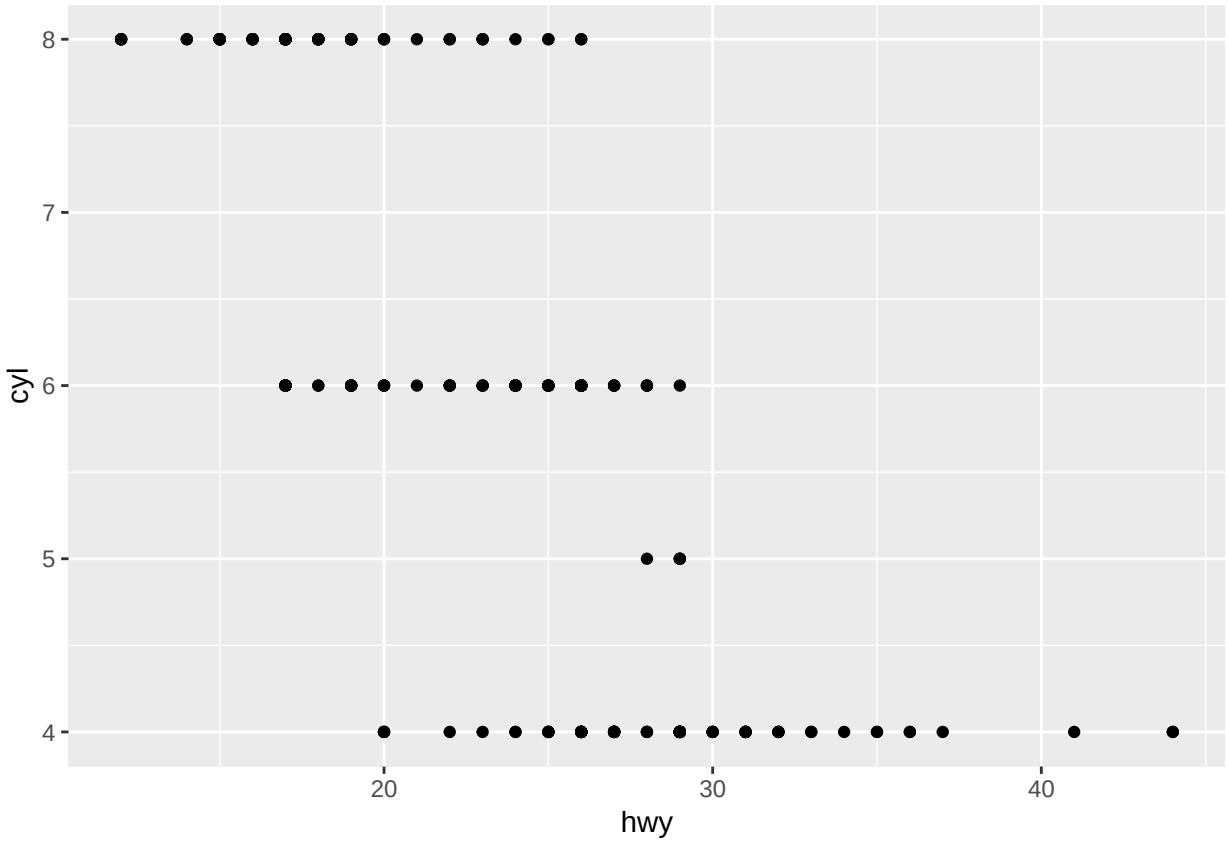
```
ncol(mpg)
```

```
## [1] 11
```

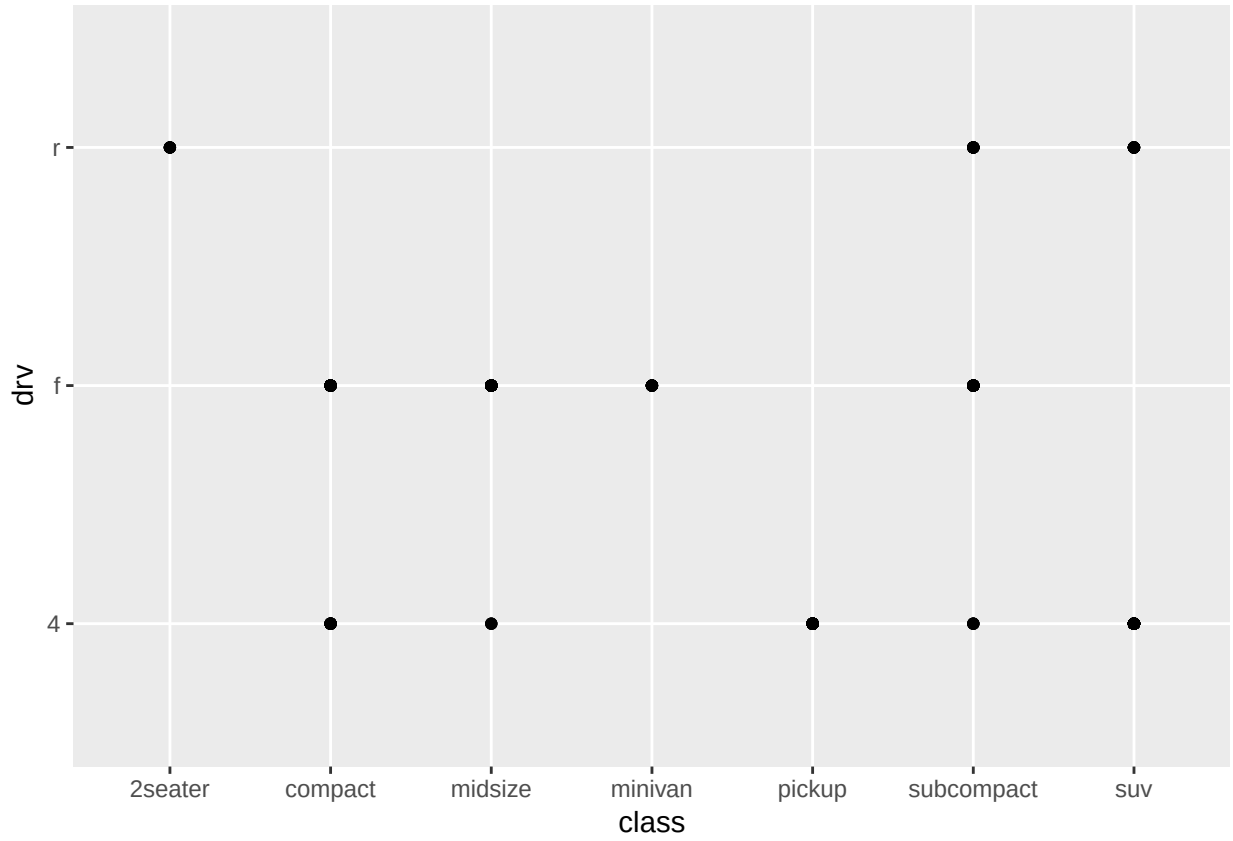
```
#234 rows; 11 columns

#3. What does the drv variable describe? Read the help for ?mpg to find out.
#drv denotes the drivetrain, whether it be front-wheel, rear-wheel or 4-wheel/all-wheel drive

#4. Make a scatterplot of hwy vs cyl.
ggplot(mpg, aes(hwy, cyl)) + geom_point()
```




```
#5. What happens if you make a scatterplot of class vs drv? Why is the plot not useful?
ggplot(mpg, aes(class, drv)) + geom_point()
```




```
#this plot shows drivetrains (in this data set) for different vehicle classes
#this plot is unhelpful because it doesn't provide further insight beyond common knowledge, industry st
#this plot could be considered useful because it indicates what data is in the set and what standard dr
```

4. (8 pts) Load the tibble library and the tidyr library. Tidy the tibble below. Do you need to spread or gather it? What are the variables before tidying? After tidying? You must include code and sentence descriptions to receive full credit.

You do not need to spread or gather the data.

Before tidying, the variables are: ~species, ~weight_when_admitted, ~weight_when_adopted, ~supplement,

After tidying, the variables should still be: ~species, ~weight_when_admitted, ~weight_when_adopted, ~supplement,

```
library(tibble)
library(tidyr)

animals <- tribble(
~species, ~weight_when_admitted, ~weight_when_adopted, ~supplement,
"cat", 8, 15, "turkey",
"cat", 9, 11, "kibbles",
"dog", 18, 27, "turkey"
)
animals
```

```
## # A tibble: 3 x 4
##   species weight_when_admitted weight_when_adopted supplement
##   <chr>                  <dbl>               <dbl> <chr>
```

```
## 1 cat                       8                 15 turkey
## 2 cat                       9                 11 kibbles
## 3 dog                      18                 27 turkey
```

```
# gather: not needed: none of the column headers are inappropriately placed "values" -- all of the colu
# spread: not needed: none of the values in the dataset are innappropriately placed "column variables"


#species - a changing variable
#weight_when_admitted - a changing variable
#weight_when_adopted - a changing variable
#supplement - a changing variable
#note: none of the column variables (headers) are values for another changing variable
#note: none of the dataframe values should be considered column variables (or headers) because none of
```