Lab Assignment #1

Dr. Wynne's Partial Solutions

January 27, 2023

Instructions

```
library(ISLR2)
library(ggplot2)
```

Problem 2: More Basic R Commands

Part b (Explanation: 1 pt)

Explain in your own words the purpose of the `set.seed()` function. What happens if you *don't* include this line before running the `rnorm` function?

Explanation

The random number generator in R is not a *true* random number generator; that is, there is an algorithm that tells R exactly what "random" numbers should be produced. By setting the seed, we ensure that the same "random" numbers are produced every time. If we don't run the `set.seed()` function first, then we will likely get a different set of numbers every time (this is extremely frustrating when you write something based on your output, and then you knit your file to turn it in and get completely different numbers!). Thus, setting a seed first ensures that our work is reproducible.

Problem 3: Indexing Data

Part b (Code: 0.5 pts)

When you import data into R, it usually will be stored in a *data frame* object instead of a matrix. There are a variety of ways to extract an individual column from a data frame.

```
Auto2 <- Auto[1:10,]  # first 10 rows
```

In the code chunks below, find two ways to subset the `Auto2` data to get a two-column dataset containing `mpg` and `horsepower` (columns 1 and 4). Note that `drop = FALSE` is only necessary when you want a one-column data frame.

Solution

```
Auto2[,c(1,4)]
```

```
##    mpg horsepower
## 1   18        130
## 2   15        165
## 3   18        150
```

```
## 4   16        150
## 5   17        140
## 6   15        198
## 7   14        220
## 8   14        215
## 9   14        225
## 10  15        190
```

```
Auto2[c("mpg", "horsepower")]
```

```
##    mpg horsepower
## 1   18        130
## 2   15        165
## 3   18        150
## 4   16        150
## 5   17        140
## 6   15        198
## 7   14        220
## 8   14        215
## 9   14        225
## 10  15        190
```

Problem 4: Hypothesis Testing in R

Part a (Code: 1 pt)

Fix the chunk below to create a vector `x` consisting of 50 random numbers from a normal distribution with mean 100 and standard deviation 15 and a vector `y` consisting of 50 random numbers from a normal distribution with mean 90 and standard deviation 15. (Refer back to the end of Problem 2 if you need help.)

```
set.seed(199)
x <- rnorm(n = 50, mean = 100, sd = 15)
y <- rnorm(n = 50, mean = 90, sd = 15)
```

Now, run the chunk below to perform a two-sample t-test to determine whether the population means of `x` and `y` are different, and store the output in a variable called `t_test_sim`:

```
t_test_sim <- t.test(x,y, alternative = "t") # "t" for two-sided
```

After you run this chunk, the variable `t_test_sim` should appear in your Environment tab as a "List of 10." In R, the output of a hypothesis test is an *htest* object containing information about the methods and results of the inference. Let's see what information we can get out of the `t_test_sim` object:

```
names(t_test_sim)
```

```
##  [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
##  [6] "null.value"  "stderr"      "alternative" "method"      "data.name"
```

We can extract the p-value for our test using the `$` operator:

```
t_test_sim$p.value
```

```
## [1] 0.01104232
```

Part b (Code: 1 pt)

Sometimes our data is in this "wide" format (where each column represents the values from a group and we can use the individual vectors as the arguments), but more often our data is in "long" format (where each

column represents a variable). Let's see what this data would look like in long format:

```
group_values <- rep(c("x", "y"), each = 50) # 50 x followed by 50 y
numerical_values <- c(x, y)
sim_df_long <- data.frame(
  group = group_values,
  value = numerical_values
)
```

Write code in the chunk below that finds the number of rows and columns in the `sim_df_long` data frame.

```
dim(sim_df_long)
```

```
## [1] 100   2
```

```
nrow(sim_df_long)
```

```
## [1] 100
```

```
ncol(sim_df_long)
```

```
## [1] 2
```

Write code in the chunk below that finds the names of the variables in the data frame.

```
names(sim_df_long)
```

```
## [1] "group" "value"
```

```
colnames(sim_df_long)
```

```
## [1] "group" "value"
```

To perform a two-sample t-test when the data is in "long" format, we use a *formula* argument of the form `response ~ explanatory`:

```
t_test_sim_long <- t.test(value ~ group, data = sim_df_long, alternative = "t")
```

Part c (Explanation: 0.5 pts)

Write a chunk to extract the p-value from `t_test_sim_long`. Do you get the same p-value that you got originally?

Solution

```
t_test_sim_long$p.value
```

```
## [1] 0.01104232
```

Yes, it's the same p-value.

Part d (Explanation: 1.5 pts)

What information is contained the `statistic`, `parameter`, and `conf.int` objects within `t_test_sim` and `t_test_sim_long`? (You can write and run code chunks to verify your answer.)

Solution

```
t_test_sim$statistic
```

```
##        t
## 2.591615
```

```
t_test_sim_long$statistic
```

```
##        t
## 2.591615
```

The `statistic` object contains the observed value of the test statistic (in this case a t-statistic).

```
t_test_sim$parameter
```

```
##       df
## 96.03834
```

```
t_test_sim_long$parameter
```

```
##       df
## 96.03834
```

The `parameter` object actually contains the degrees of freedom in the appropriate t-distribution.

```
t_test_sim$conf.int
```

```
## [1]  1.707007 12.877920
## attr(,"conf.level")
## [1] 0.95
```

```
t_test_sim_long$conf.int
```

```
## [1]  1.707007 12.877920
## attr(,"conf.level")
## [1] 0.95
```

The `conf.int` object contains the bounds of a 95% (as indicated by the `conf.level` attribute) confidence interval for the parameter we are testing about (in this case the difference in population means).

Problem 5: Graphing with Base R

For parts (a) through (c):

- Create the requested plot using the *Auto* dataset. Make sure each plot has appropriate labels for the x-and y-axis. You can find all of the relevant code in ISLR Lab 2.3.5.
- Write a short paragraph describing the graph to someone who does not know anything about the *Auto* dataset. When I describe a graph, I use the TAME strategy to orient the reader/listener:

  Topic: what data (observational units and variables) you are graphing

  Axes: what variables are being mapped to the x-axis and y-axis (and other properties, e.g., color)

  Main point: the most important takeaway (for one variable, this usually deals with the mode of the distribution; for two variables, this usually deals with the association)

  Extra information (optional): anything else that you find interesting to point out

Part a (Code: 0.5 pts; Explanation: 1 pt)

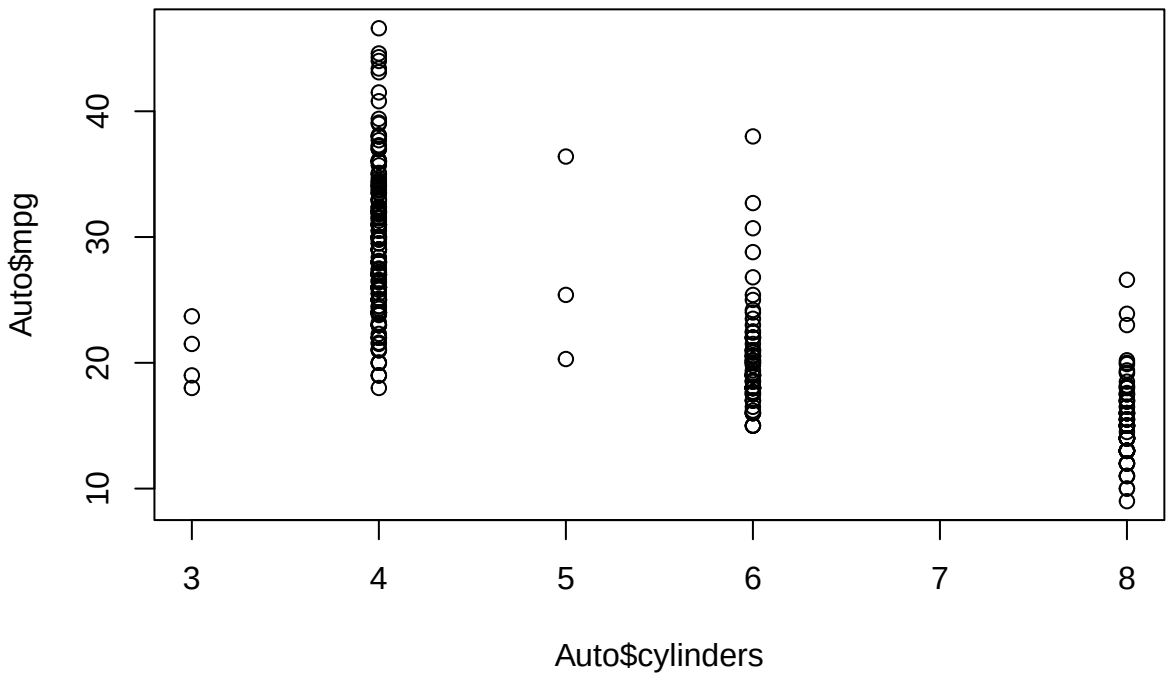
Create and describe a scatterplot of `mpg` (response) against `cylinders` (numerical predictor). Do not attach the dataset; use the `$` sign to select the variables.

Solution

```
dim(Auto)
```

```
## [1] 392   9
```

```
plot(Auto$cylinders, Auto$mpg)
```



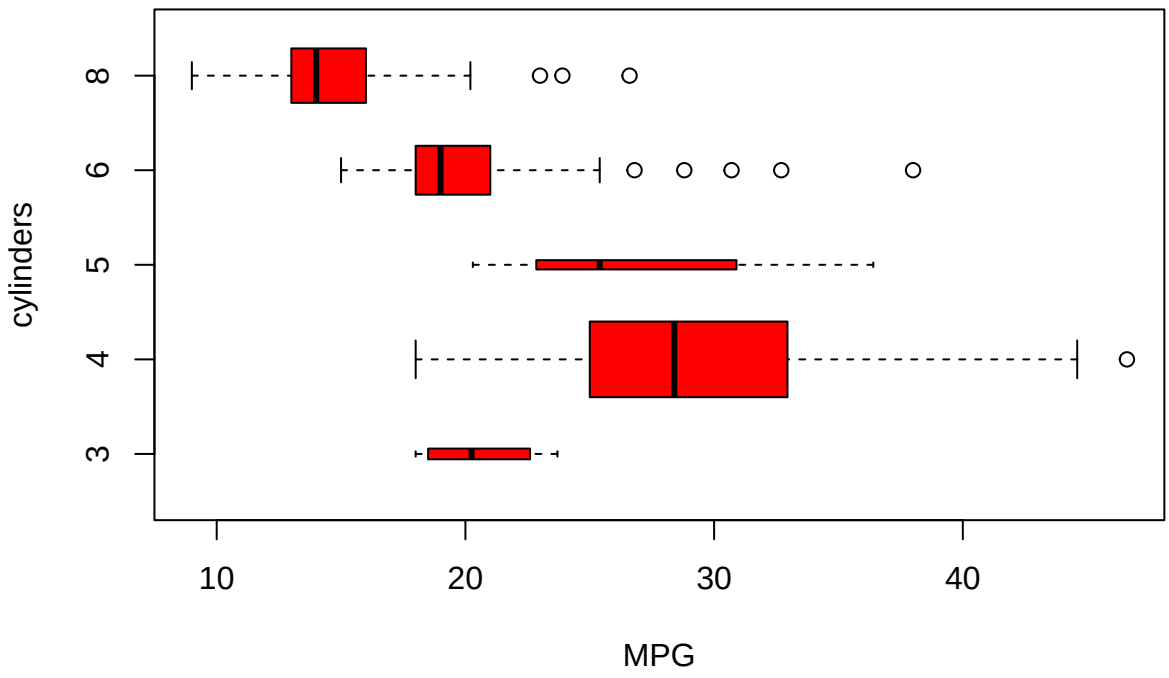
This graph shows how the miles per gallon varies with the number of cylinders for 392 cars. The mpg is on the y-axis and the number of cylinders is on the x-axis. In general, cars with more cylinders tend have lower gas mileage.

Part b (Code: 0.5 pts; Explanation: 1 pt)

Create and describe a boxplot of `mpg` (response) by `cylinders` (categorical predictor). The boxes should be horizontal and red.

Solution

```
Auto$cylinders <- as.factor(Auto$cylinders)
plot(Auto$cylinders, Auto$mpg, col = "red", horizontal = T, varwidth = T,
     xlab = "MPG", ylab = "cylinders")
```

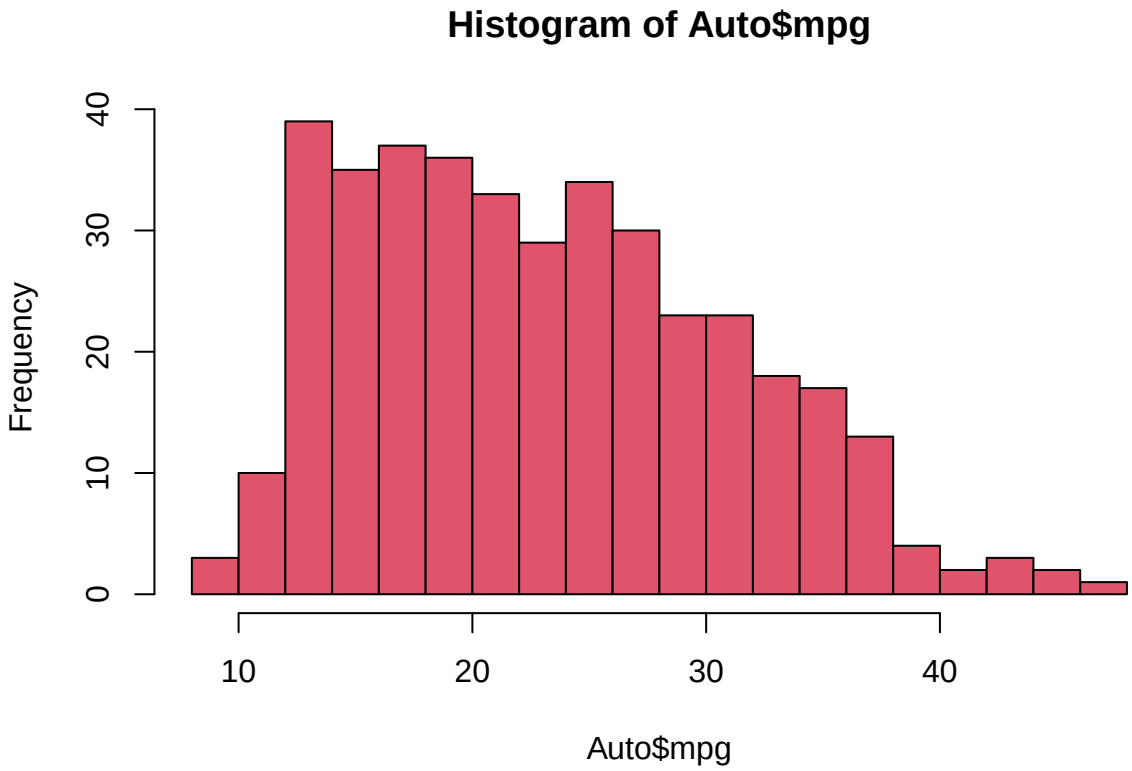


This graph also shows how the miles per gallon varies with the number of cylinders for 392 cars. The mpg is now on the x-axis and the number of cylinders is on the y-axis. Cars with 8 cylinders tend to have the lowest gas mileage and cars with 4 cylinders tend to have the highest gas mileage. The variability in gas mileage appears to be greatest for the 4-cylinder cars. Almost all the cars have 4, 6, or 8 cylinders. There are several cars with unusually high MPG for their number of cylinders.

Part c (Code: 0.5 pts; Explanation: 1 pt)

Create and describe a histogram of `mpg`. The bars should be red and there should be roughly 15 bars.

```
hist(Auto$mpg, col = 2, breaks = 15)
```



This graph shows the distribution of gas mileage for 392 cars in the `Auto` dataset. The x-axis shows the miles per gallon and the y-axis shows the number of cars with that gas mileage. Each bar covers a 2-mpg interval. The graph appears to be skewed right with a mode around 15-20 mpg.