Homework Assignment #8

Matt Kline

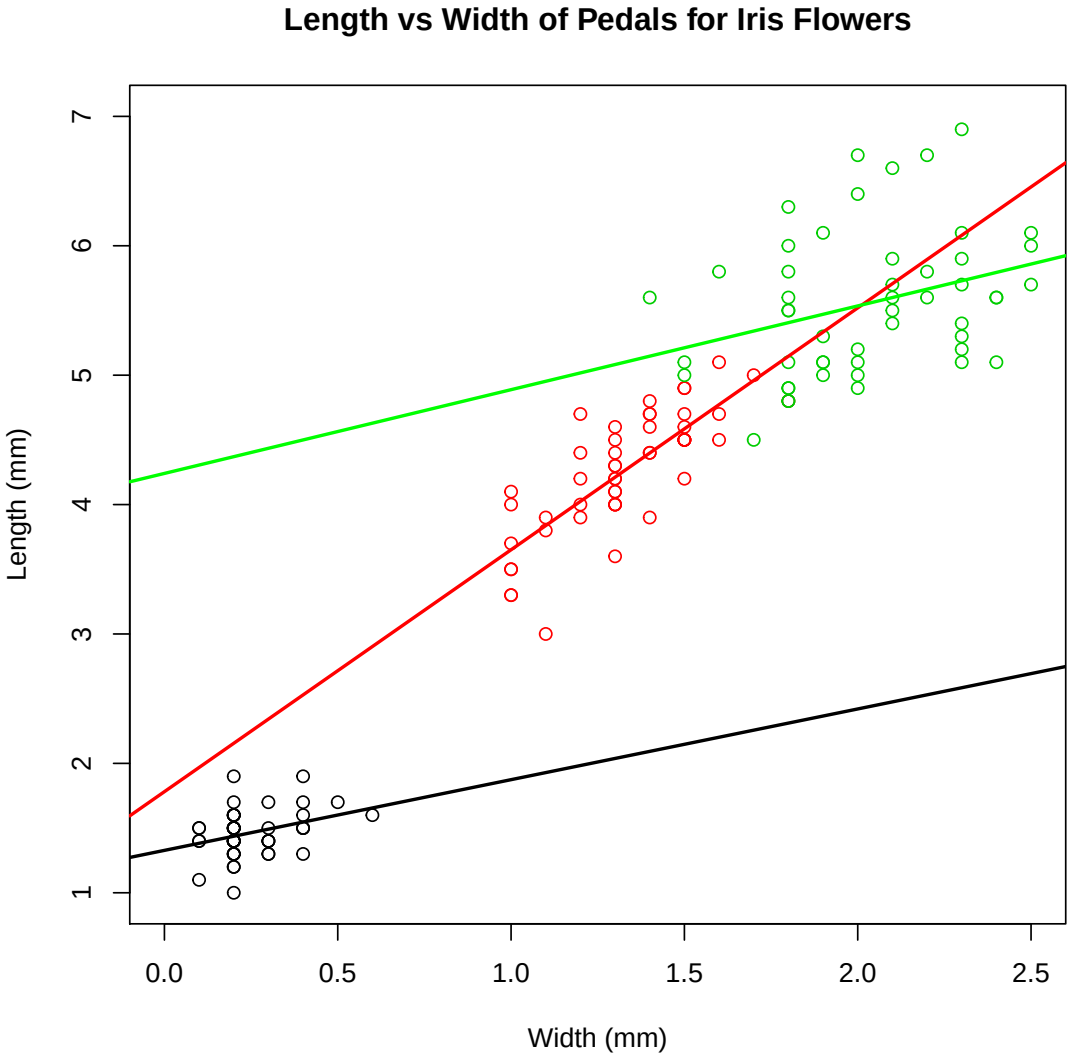
April 11, 2017

Part 1 Statistics, Math and Simulations:

1.) The built-in dataset iris contains data analyzed by the statistician R. A. Fisher on the lengths and widths of petals of three different species of iris flowers, Iris Setosa, Iris Verginica and Iris Versicolor.

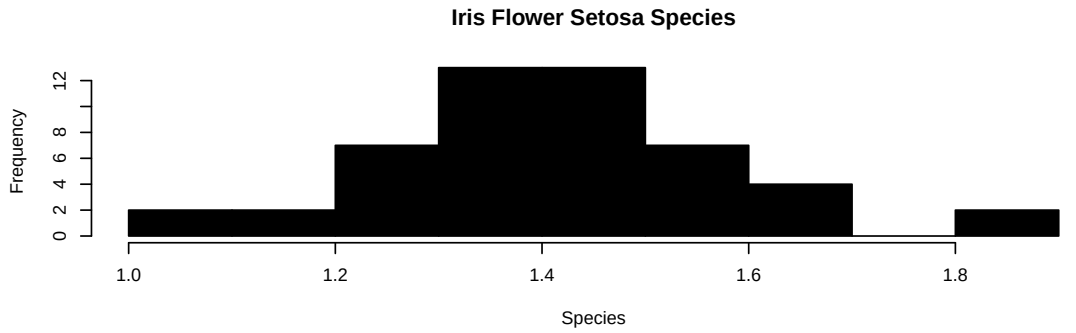
a. Carry out three separate linear regression analyses, one for each species, with Petal.Length as the response (y) and Petal.Width as the predictor (x). Then add the three regression lines to a scatterplot of the data, with species represented by different colors, as below.

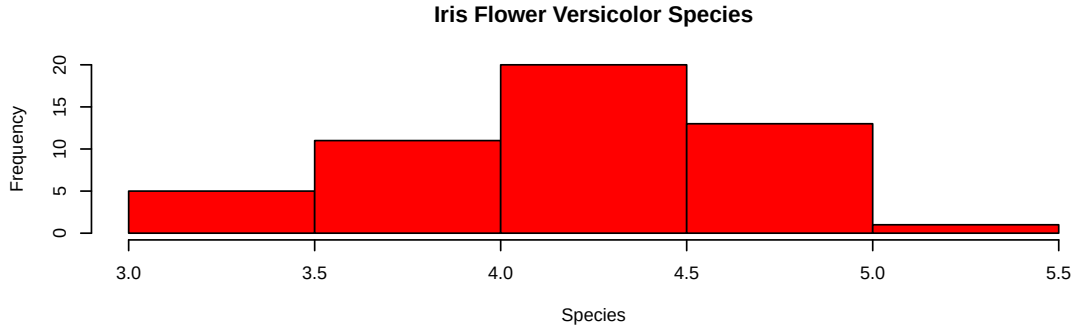
```
myReg1 <- lm(Petal.Length[iris$Species == "setosa"] ~
               Petal.Width[iris$Species == "setosa"],
             data = iris)
myReg2 <- lm(Petal.Length[iris$Species == "versicolor"] ~
               Petal.Width[iris$Species == "versicolor"],
             data = iris)
myReg3 <- lm(Petal.Length[iris$Species == "virginica"] ~
               Petal.Width[iris$Species == "virginica"],
             data = iris)

plot(x = iris$Petal.Width, y = iris$Petal.Length,
     main = "Length vs Width of Pedals for Iris Flowers",
     xlab = "Width (mm)", ylab = "Length (mm)", type = "p",
     xlim = c(0, 2.5), ylim = c(1, 7), col = iris$Species)

abline(myReg1, col = "black", lwd = 2)
abline(myReg2, col = "red", lwd = 2)
abline(myReg3, col = "green", lwd = 2)
```

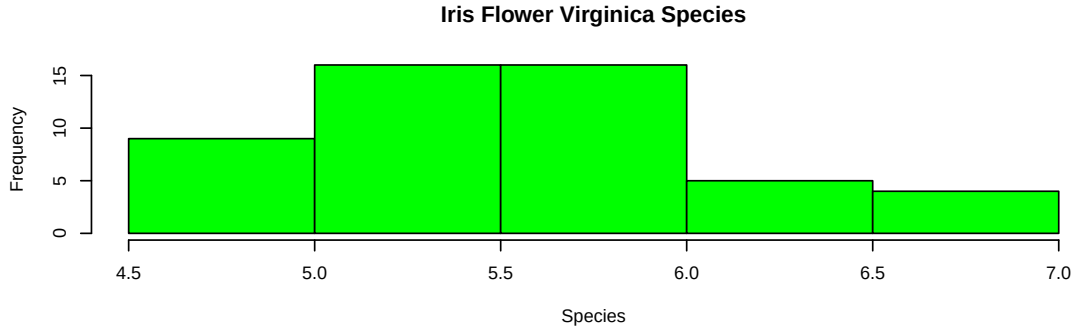


b. Now make histograms of the three sets of residuals. Use par(mfrow = c(3, 1)) to arrange the three histograms together in a multi-figure array with three rows and one column.

```
par(mfrow = c(3, 1))
hist(x = iris$Petal.Length[iris$Species == "setosa"],
     xlab = "Species", col = "black",
     main = "Iris Flower Setosa Species")
hist(x = iris$Petal.Length[iris$Species == "versicolor"],
     xlab = "Species", col = "red",
     main = "Iris Flower Versicolor Species")
hist(x = iris$Petal.Length[iris$Species == "virginica"],
     xlab = "Species", col = "green",
     main = "Iris Flower Virginica Species")
```







2.) Use the following command to create a data frame containing a "standard deck of 52 cards"

```
cards <- data.frame(Rank = rep(1:13, times = 4),
                    Suit = rep(c("Clubs", "Diamonds", "Hearts", "Spades"),
                               each = 13))
```

a. Write an R command that "shuffles" the cards, placing them in random order in a new data frame.

```
cards <- cards[sample(nrow(cards), nrow(cards)), ]
```

b. Write an R command that randomly "draws" 5 cards from the deck (without replacement).

```
cards[sample(nrow(cards), 5, replace = F), ]

##    Rank     Suit
## 15    2 Diamonds
## 43    4   Spades
## 31    5   Hearts
```

```
## 2     2    Clubs
## 18    5 Diamonds
```

c. How would you modify the command of Part b so that the cards are drawn with replacement?

```
cards[sample(nrow(cards), 5, replace = T), ]

##    Rank     Suit
## 16    3 Diamonds
## 48    9   Spades
## 13   13    Clubs
## 15    2 Diamonds
## 1     1    Clubs
```

3.) Random number generators can create lots of numbers for you with a minimal amount of typing. How might you use runif() (and round()) to create a vector with numbers similar to 7.3, 6.8, 9.0, 12.0, 2.4, 18.9

```
round(runif(6, 0, 20), 1)

## [1] 13.0  3.9 19.1  3.6 15.0  0.7
```

4.)

a. Use rnorm() to generate random samples of sizes n = 10, n = 100, n = 1, 000, n = 10, 000, and n = 100, 000 from a normal distribution with mean $\mu = 40 and standard deviation \sigma = 5, and compute the sample mean \bar{x} for$

```
mean(rnorm(10, 40, 5))

## [1] 39.86948

mean(rnorm(100, 40, 5))

## [1] 39.55882

mean(rnorm(1000, 40, 5))

## [1] 40.04339

mean(rnorm(10000, 40, 5))

## [1] 40.0625

mean(rnorm(100000, 40, 5))

## [1] 40.01666
```

b. What tends to happen to the discrepancy between and $\mu as n gets large? (This is called the Law of Large Numbers.)$
The two values are becoming closer and closer

5.)

a. Use punif() to calculate the probability that a uniformly distributed random variable on the interval from 0 to 1 will be smaller than 0.75.

```
punif(0.75, 0, 1)

## [1] 0.75
```

b. Use punif() to calculate the probability that a uniformly distributed random variable on the interval from 10 to 30 will be smaller than 13.

```
punif(13, 10, 30)

## [1] 0.15
```

c. Use pnorm() to calculate the probability that a normally distributed random variable with mean $\mu = 0 and standard deviation \sigma = 1 will be larger than 3$

```
pnorm(3, 0, 1)

## [1] 0.9986501
```

d. Use pnorm() to calculate the probability that a normally distributed random variable with mean $\mu = 35 and standard deviation \sigma = 6 will be larger than 42$.

```
pnorm(42, 35, 6)

## [1] 0.8783275
```

6.)

a. Use rnorm() to generate 10,000 random numbers from a normal probability distribution whose mean and standard deviation are $\mu = 20 and \sigma = 5$

```
#rnorm(10000, 20, 5)
```

b. Determine the proportion of your 10,000 randomly numbers that are less than 15 and compare it to the theoretical probability returned by pnorm().

```
1 - sum(rnorm(10000, 20, 5) >= 15) / 10000

## [1] 0.1523

pnorm(15, 20, 5)

## [1] 0.1586553
```

The two values are pretty close in size. There is a .13% difference

Part 2 Object-Oriented Programming

7.) Determine the class of Inf, NA, NaN, "", and NULL.

```r
class(Inf)

## [1] "numeric"

class(NA)

## [1] "logical"

class(NaN)

## [1] "numeric"

class("")

## [1] "character"

class(NULL)

## [1] "NULL"
```

8.) The harmonic mean is defined as the reciprocal of the arithmetic mean of the reciprocals of the values in a data set, i.e. 1/mean(1/x), where x is a vector of positive values.

a. Write a harmonic mean function hm() that takes a vector argument x and gives appropriate feedback when the input is not numeric or contains nonpositive values. Hint: The function is.numeric() can be used to check whether x is numeric.

```r
hm <- function(vector){
  if (is.numeric(vector)) {
    if (any(vector <= 0)) stop("Value is negative")
    1/mean(1/vector)
  } else stop("Value is not numeric")
}
```

b. Test your hm() function by passing it 1) a vector containing one or more nonpositive values, 2) a character vector, and 3) a numeric vector of positive values.

```r
hm(c(10, -2, 19, -24, 6, 23, 0, 24, 54, 77))

## Error in hm(c(10, -2, 19, -24, 6, 23, 0, 24, 54, 77)):  Value is negative

hm(c("a", "red", "firetruck", "yellow", "zebra", "house"))

## Error in hm(c("a", "red", "firetruck", "yellow", "zebra", "house")):  Value is not
numeric

hm(c(10, 2, 19, 24, 6, 23, 47, 24, 54, 77))

## [1] 10.01109
```

c. Modify your harmonic mean function so that the return value has the class "harmonic".

```
class(hm) <- "harmonic"
```

d. Now write an S3 print() method for this class that displays the message "The harmonic mean is y" where y is the harmonic mean. Hint: The function cat() can be used inside your print() method to print the message along with the value of y.

```
print.S3 <- function(value){
  cat("The harmoic mean is ", hm(value))
}
```

e. Test your print() method, for example by typing:

```
y <- hm(c(3, 2, 2, 5))
print.S3(y)

## The harmoic mean is  2.608696
```