# Data Science Module 5 Exercises

Brady Lamson

2/28/2022

## 9: Statistical Foundations

### 9.3: Simulations

**Exercise 1:**

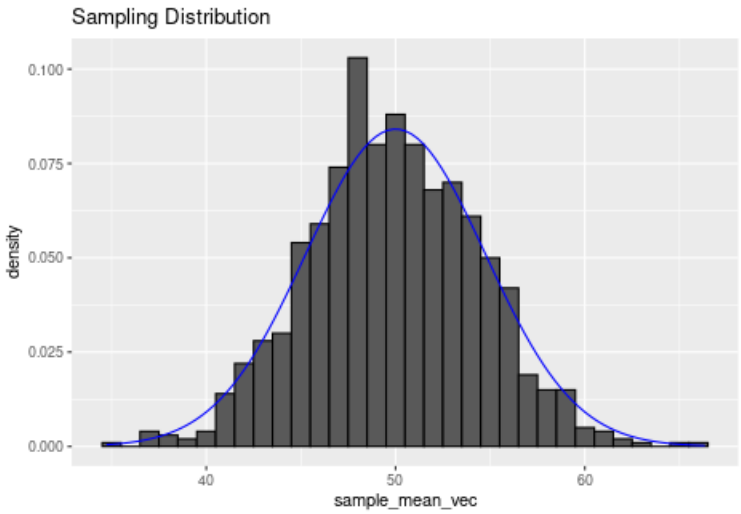
```
# A
sample_mean_vec <- c()

for(i in 1:1000) {
    sim_sample <- rnorm(n = 10, mean = 50, sd = 15)
    sample_mean_vec <- c(sample_mean_vec, mean(sim_sample))
}
```

```
#B
sample_mean <- mean(sample_mean_vec) %>% round(digits = 3)
sample_standard_error <- sd(sample_mean_vec) %>% round(digits = 3)

glue::glue("
    The mean of the sample mean vector is approximately {sample_mean},
    and the standard error of the vector is approximately {sample_standard_error}.
")
```

```
## The mean of the sample mean vector is approximately 49.876,
## and the standard error of the vector is approximately 4.499.
```

c) We can see that both values we got are very close to their theoretical values. The theoretical mean would be 50 and we can calculate the theoretical standard error, $\sigma/\sqrt{n} = 15/\sqrt{10} = 4.74342$.

```
# D

ggplot(data = data.frame(sample_mean_vec)) +
    geom_histogram(
        mapping = aes(x = sample_mean_vec, y = stat(density)),
        binwidth = 1,
        color = "black") +
    geom_function(
        fun = dnorm,
        args = list(mean = 50, sd = 15/sqrt(10)),
        color = "blue") +
    labs(title = "Sampling Distribution")
```




The blue line represents an idealized normal distribution. What we can see is that our simulation comes incredibly close. The center is right around 50, the shape follows the same curve and the density matches as well.

---

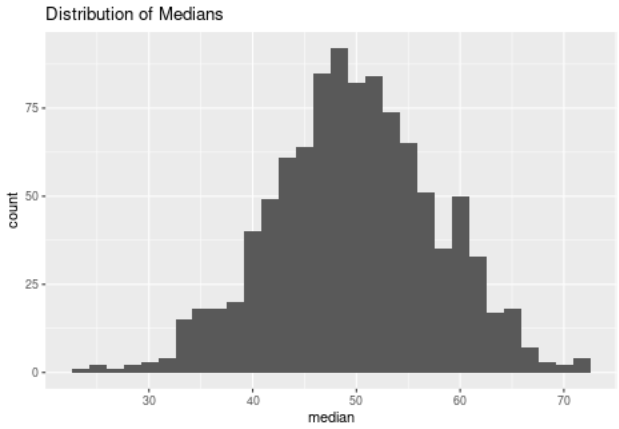
**Exercise 2**

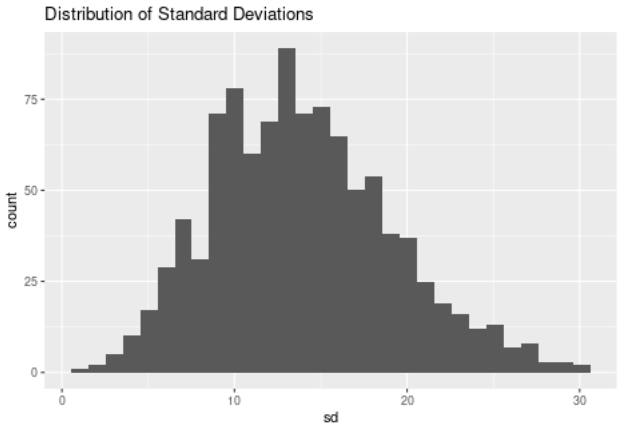
```
sample_mean_vec <- c()
sample_median_vec <- c()
sample_sd_vec <- c()
sample_min_vec <- c()
sample_max_vec <- c()
sim_vec <- c()

for(i in 1:1000) {
    sim_sample <- rnorm(n = 5, mean = 50, sd = 15)
    sim_vec <- c(sim_vec, sim_sample)
    # simulated statistic vectors
    sample_mean_vec <- c(sample_mean_vec, mean(sim_sample, na.rm = TRUE))
    sample_median_vec <- c(sample_median_vec, median(sim_sample, na.rm = TRUE))
    sample_sd_vec <- c(sample_sd_vec, sd(sim_sample, na.rm = TRUE))
    sample_min_vec <- c(sample_min_vec, min(sim_sample, na.rm = TRUE))
    sample_max_vec <- c(sample_max_vec, max(sim_sample, na.rm = TRUE))
}
```

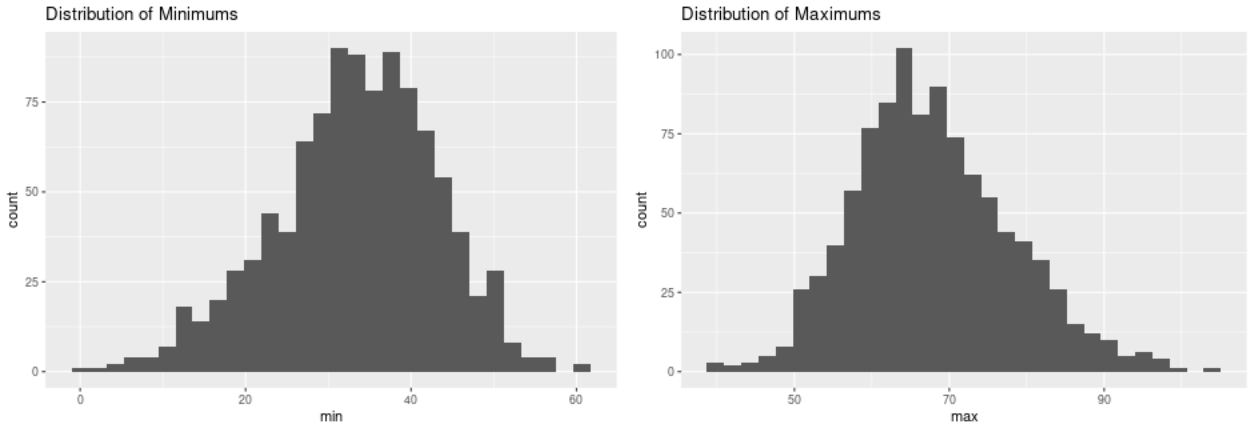
```
simulation_stats <- tibble(
    median = sample_median_vec,
    sd = sample_sd_vec,
    min = sample_min_vec,
    max = sample_max_vec
)
```

```
data.frame(
    average = sapply(simulation_stats, FUN = mean),
    standard_error = sapply(simulation_stats, FUN = sd)
) %>%
    kbl()
```

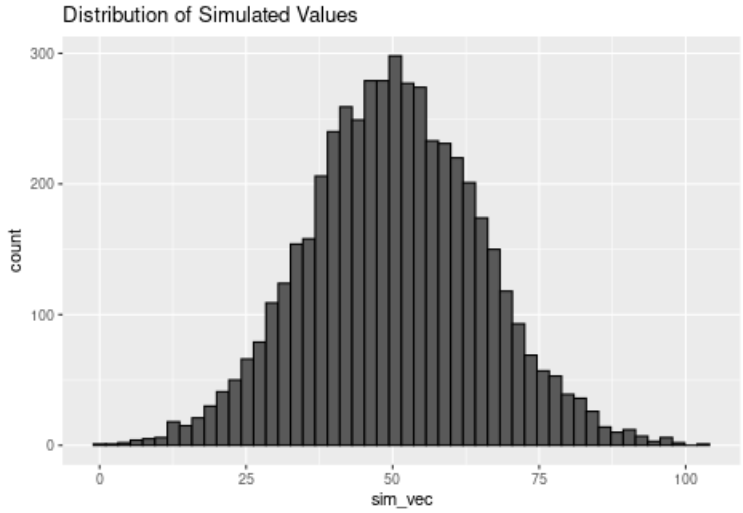
| | average | standard_error |
|---|---|---|
| median | 49.85544 | 7.897012 |
| sd | 13.94853 | 5.194970 |
| min | 33.32671 | 9.785064 |
| max | 67.84086 | 10.205027 |






```
ggplot() +
    aes(x = sim_vec) +
    geom_histogram(bins = 50, color = "black") +
    labs(title = "Distribution of Simulated Values")
```




The distribution of the simulated values seems very much normal. The center is right around 50 and the count dips as the values get further and further away from the mean.

---

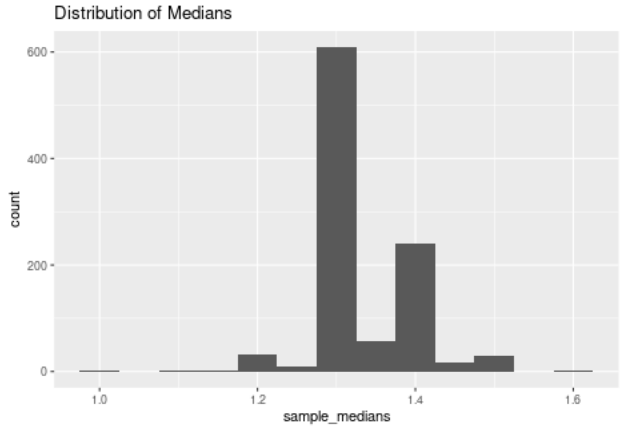
## 9.4: The Bootstrap

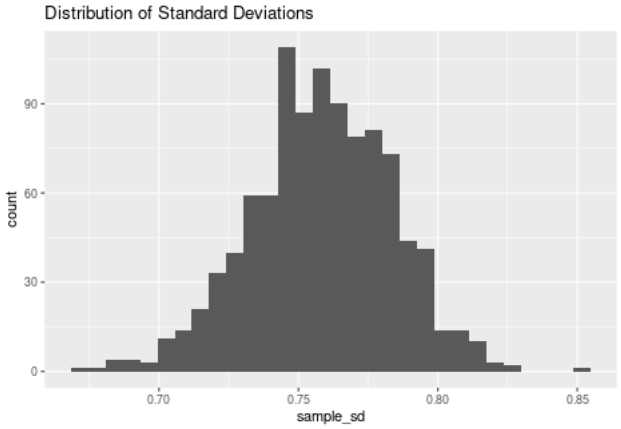
### Exercise 3

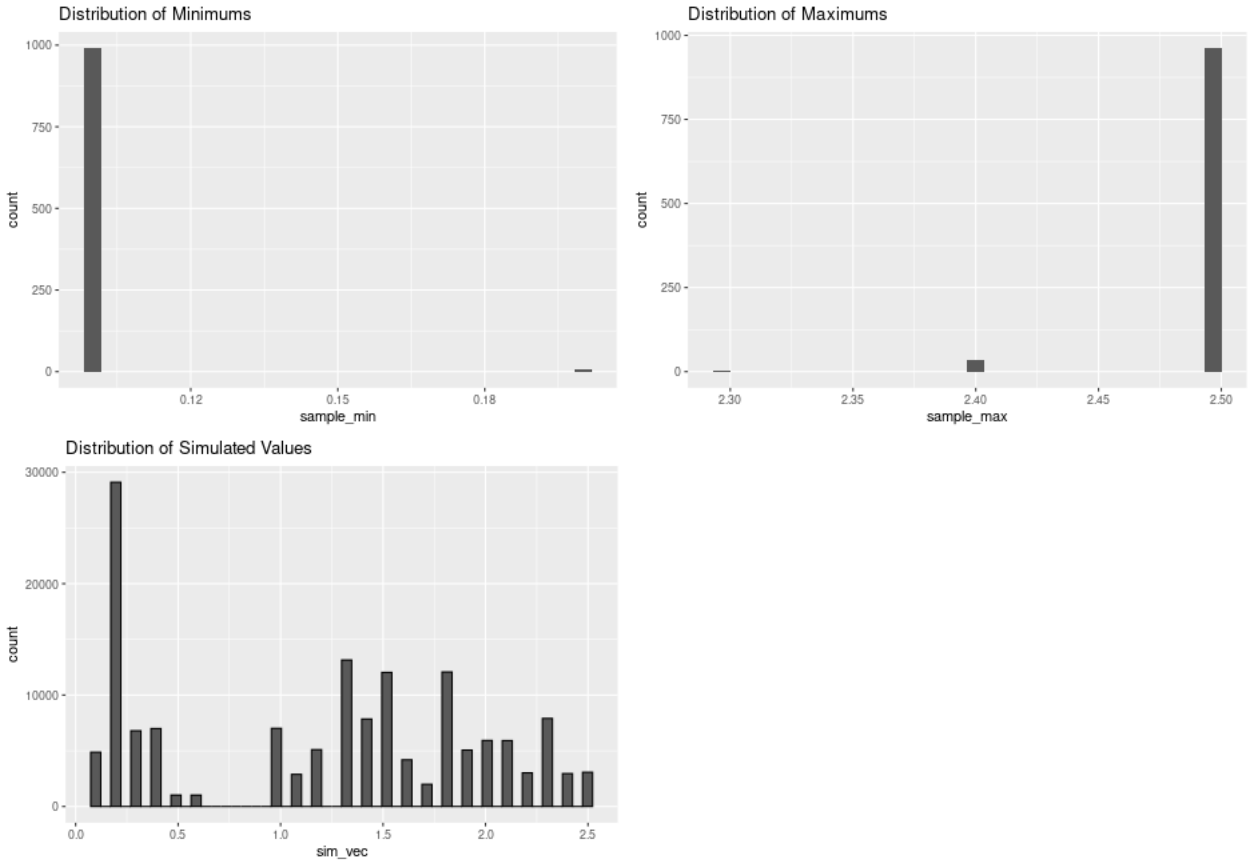
```
B <- 1000
sample_df <- tibble(
    sample_medians = rep(NA, B),
    sample_sd = rep(NA, B),
    sample_min = rep(NA, B),
    sample_max = rep(NA, B),
)

sim_vec <- c()

for(i in 1:B) {
    resamp <- slice_sample(.data = iris,
                           n = 150,
                           replace = TRUE)

    # Simulated values
    sim_vec <- c(sim_vec, resamp$Petal.Width)

    # Simulated statistics
    sample_df$sample_medians[i] <- median(resamp$Petal.Width)
    sample_df$sample_sd[i] <- sd(resamp$Petal.Width)
    sample_df$sample_min[i] <- min(resamp$Petal.Width)
    sample_df$sample_max[i] <- max(resamp$Petal.Width)
}
```

```
data.frame(
    average = sapply(sample_df, FUN = mean),
    standard_error = sapply(sample_df, FUN = sd)
) %>%
    kbl()
```

| | average | standard_error |
|---|---|---|
| sample_medians | 1.3308000 | 0.0618076 |
| sample_sd | 0.7583686 | 0.0253813 |
| sample_min | 0.1009000 | 0.0094488 |
| sample_max | 2.4958000 | 0.0219737 |






All of the plots here are very much **not** normal. The medians have two values that are far more frequent than any others, whereas the minimums and maximums only really have 1 value that occurs at all. The standard deviations seem to have a relatively normal distribution though, with a slight left skew. As for the general simulated values, There's a very non normal distribution with a peak close to 0 and a fair frequency of values between 1.3 and 1.8.

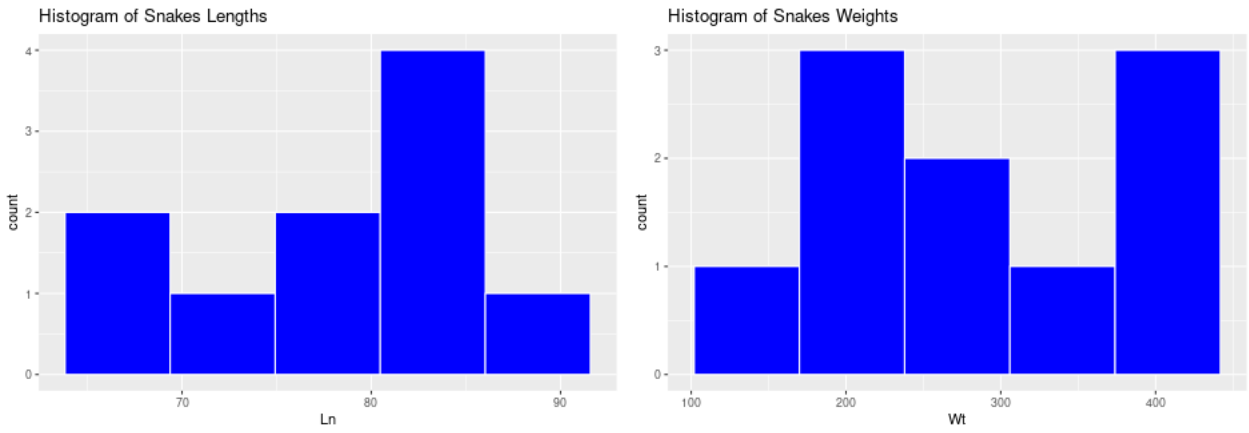
---

## 9.5: Outliers

### Exercise 4

```
SnakeID <- 1:10
Ln <- c(85.7, 64.5, 84.1, 82.5, 78.0, 65.9, 81.3, 71.0, 86.7, 78.7)
Wt <- c(331.9, 121.5, 382.2, 287.3, 224.3, 380.4, 245.2, 208.2, 393.4, 228.3)
Snakes <- data.frame(SnakeID, Ln, Wt)
```

```
ggplot(data = Snakes) +
    geom_histogram(mapping = aes(x = Ln),
                   fill = "blue",
                   color = "white",
                   bins = 5) +
    ggtitle("Histogram of Snakes Lengths")

ggplot(data = Snakes) +
    geom_histogram(mapping = aes(x = Wt),
                   fill = "blue",
                   color = "white",
                   bins = 5) +
    ggtitle("Histogram of Snakes Weights")
```

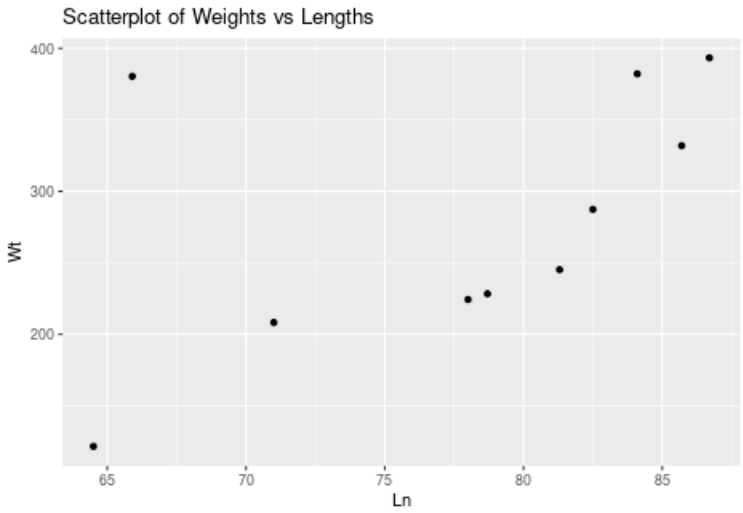



a)

From these two histograms I absolutely cannot tell what the outlier is. There's too few bins to properly differentiate here.

b)

```
ggplot(data = Snakes) +
    geom_point(mapping = aes(x = Ln, y = Wt)) +
    ggtitle("Scatterplot of Weights vs Lengths")
```




Here it is a lot easier to notice the outlier, it's the value in the top left.

---

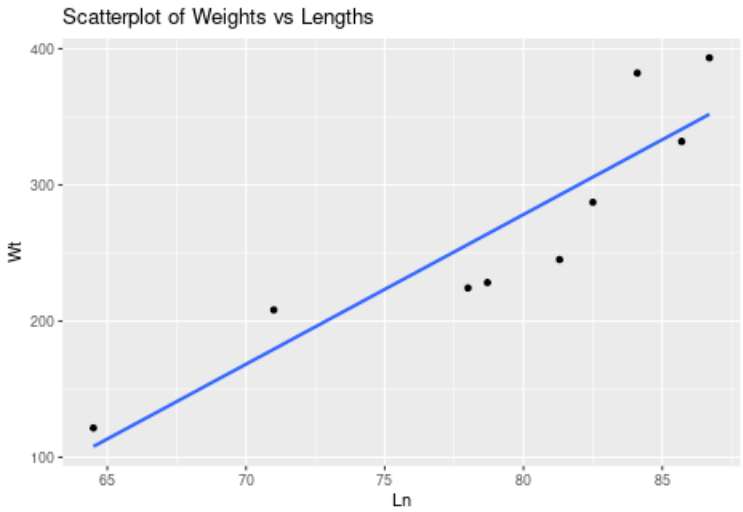
## 9.6: Statistical Models: Explaining Variation

### Exercise 5

```
SnakeID <- 1:9
Ln <- c(85.7, 64.5, 84.1, 82.5, 78.0, 81.3, 71.0, 86.7, 78.7)
Wt <- c(331.9, 121.5, 382.2, 287.3, 224.3, 245.2, 208.2, 393.4, 228.3)
Snakes <- data.frame(SnakeID, Ln, Wt)
```

```
ggplot(data = Snakes, mapping = aes(x = Ln, y = Wt)) +
geom_point() +
geom_smooth(method = "lm", se = FALSE) +
ggtitle("Scatterplot of Weights vs Lengths")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
my.reg <- lm(Wt ~ Ln, data = Snakes)
summary(my.reg)
```

```
##
## Call:
## lm(formula = Wt ~ Ln, data = Snakes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.395 -32.020  -9.061  28.826  58.827
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -601.083    154.333  -3.895 0.005939 **
## Ln            10.992      1.942   5.660 0.000767 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.94 on 7 degrees of freedom
## Multiple R-squared:  0.8207, Adjusted R-squared:  0.795
## F-statistic: 32.03 on 1 and 7 DF,  p-value: 0.0007667
```

From the output of summary(), the equation of the fitted line is:

$$\hat{Y} = -601.08 + 10.99x$$

Obtain the predicted weight for a snake whose length is 80cm in two ways:

1: By plugging 80 into the equation for x.

```
fitted_line <- function(x) {
    -601.08 + (10.99*x)
}

fitted_line(80)
```

```
## [1] 278.12
```

2: By using predict

```
predict(my.reg, newdata = data.frame(Ln = 80))
```

```
##        1
## 278.3047
```

b) Each additional cm of elongation typically adds around 10.99 units of weight to the snake. 5cm will add 55 units of weight.

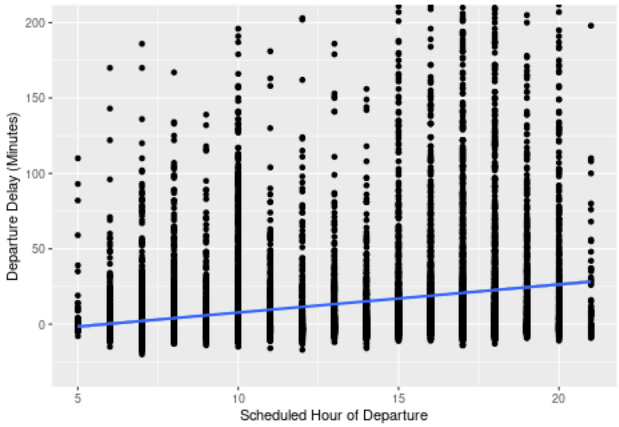
---

**Exercise 6**

```
SF <- dplyr::filter(
    .data = nycflights13::flights,
    dest == "SFO",
    !is.na(arr_delay)
)
```

```
ggplot(data = SF, mapping = aes(x = hour, y = dep_delay)) +
    geom_point() +
    geom_smooth(method = "lm") +
    xlab("Scheduled Hour of Departure") + ylab("Departure Delay (Minutes)") +
    coord_cartesian(ylim = c(-30, 200))
```

```
## `geom_smooth()` using formula 'y ~ x'
```




a) Use lm() and summary() to obtain the equation of the fitted regression line, with dep_delay as the response and hour as the explanatory variable.

```
flight_reg <- lm(dep_delay ~ hour, data = SF)
summary(flight_reg)
```

```
##
## Call:
## lm(formula = dep_delay ~ hour, data = SF)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -37.33 -17.42  -8.74  -1.10 991.39
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10.95593    1.03591  -10.58   <2e-16 ***
## hour          1.86451    0.07692   24.24   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.34 on 13171 degrees of freedom
## Multiple R-squared:  0.0427, Adjusted R-squared:  0.04263
## F-statistic: 587.5 on 1 and 13171 DF,  p-value: < 2.2e-16
```

The equation of the line is:

$$\hat{Y} = -10.956 + 1.86X$$

b) Obtained the predicted departure delay for a flight whose departure hour is 15 in two ways. Plug in 15 into the equation and use the predict command.

```
-10.956 + (1.86 * 15)
```

```
## [1] 16.944
```

```
predict(flight_reg, newdata = data.frame(hour = 15))
```

```
##        1
## 17.01166
```

c) The departure delay increases by around 1.86 minutes for every hour that passes.

---

**Exercise 7**

```
SnakeID <- 1:9
Ln <- c(85.7, 64.5, 84.1, 82.5, 78.0, 81.3, 71.0, 86.7, 78.7)
Wt <- c(331.9, 121.5, 382.2, 287.3, 224.3, 245.2, 208.2, 393.4, 228.3)
Snakes <- data.frame(SnakeID, Ln, Wt)
```

```
snake_reg <- lm(Wt ~ Ln, data = Snakes)
```

a) What class of object is returned by lm()

```
class(snake_reg)
```

```
## [1] "lm"
```

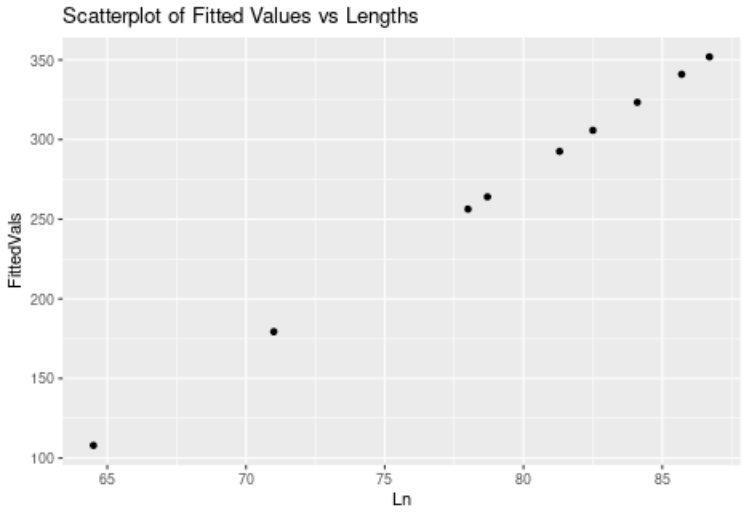
```
#B

is.list(snake_reg)
```

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

```
#C

names(snake_reg)
```

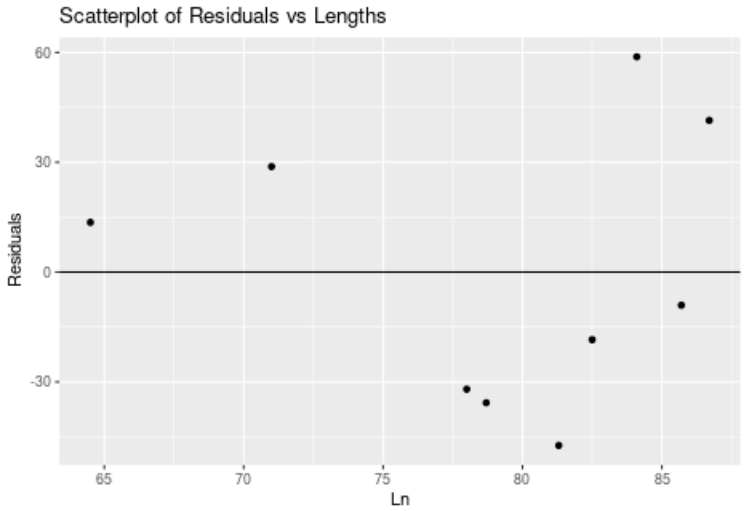
```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
# D

Snakes <- dplyr::mutate(
    Snakes, FittedVals = snake_reg$fitted.values
)

ggplot(
    Snakes, aes(x = Ln, y = FittedVals)
) +
    geom_point() +
    ggtitle("Scatterplot of Fitted Values vs Lengths")
```



The plot of fitted values looks pretty much like a straight line, this makes sense considering we got these values from a linear regression.

```
# E

Snakes <- dplyr::mutate(
    Snakes, Residuals = snake_reg$residuals
)

Snakes %>%
    ggplot(aes(x = Ln, y = Residuals)) +
    geom_point() +
    geom_hline(yintercept = 0) +
    ggtitle("Scatterplot of Residuals vs Lengths")
```




What we see is a scattering of points above or below the x-axis. This scattering of points just shows how far away an actual data point was from the predicted value.

```
# F

sum(Snakes$Residuals)
```

```
## [1] -1.065814e-14
```

We get a value here that is absurdly close to 0, it likely just isn't 0 due to rounding error.

---

**Exercise 8**

a) According to the output of summary the residual standard error appears to be 39.34 with 13171 degrees of freedom.

b) The value of $R^2$ seems to be 0.0427.

c) According to the provided table this linear model would be a very poor fit for the data.

---

**Exercise 9**

```
Sales <- c(174.4, 164.4, 244.2, 154.6, 181.6, 207.5, 152.8, 163.2,
           145.4, 137.2, 241.9, 191.1, 232.0, 145.3, 161.1, 209.7,
           146.4, 144.0, 232.6, 224.1, 166.5)
Under16 <- c(68.5, 45.2, 91.3, 47.8, 46.9, 66.1, 49.5, 52.0, 48.9,
             38.4, 87.9, 72.8, 88.4, 42.9, 52.5, 85.7, 41.3, 51.7,
             89.6, 82.7, 52.3)
Income <- c(16.7, 16.8, 18.2, 16.3, 17.3, 18.2, 15.9, 17.2, 16.6, 16.0,
            18.3, 17.1, 17.4, 15.8, 17.8, 18.4, 16.5, 16.3, 18.1, 19.1,
            16.0)
portraitSales <- data.frame(Sales, Under16, Income)
```

```
# A

sales_reg <- lm(Sales ~ Under16 + Income, data = portraitSales)

sales_reg %>% summary()
```

```
##
## Call:
## lm(formula = Sales ~ Under16 + Income, data = portraitSales)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.4239  -6.2161   0.7449   9.4356  20.2151
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -68.8571    60.0170  -1.147   0.2663
## Under16       1.4546     0.2118   6.868    2e-06 ***
## Income        9.3655     4.0640   2.305   0.0333 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.01 on 18 degrees of freedom
## Multiple R-squared:  0.9167, Adjusted R-squared:  0.9075
## F-statistic:  99.1 on 2 and 18 DF,  p-value: 1.921e-10
```

The equation we get for this set of data is:

$$\hat{Y} = -68.867 + 1.455x_1 + 9.366x_2$$

```
# B1

-68.867 + (1.455 * 45) + (9.366 * 17)
```

```
## [1] 155.83
```

```
# B2

predict(sales_reg, newdata = data.frame(Under16 = 45.0, Income = 17))
```

```
##        1
## 155.8116
```

c)

Using the equation of the plane, we can surmise that sales increases by approximately **1.455** per 1.0 thousand people under 16.

d)

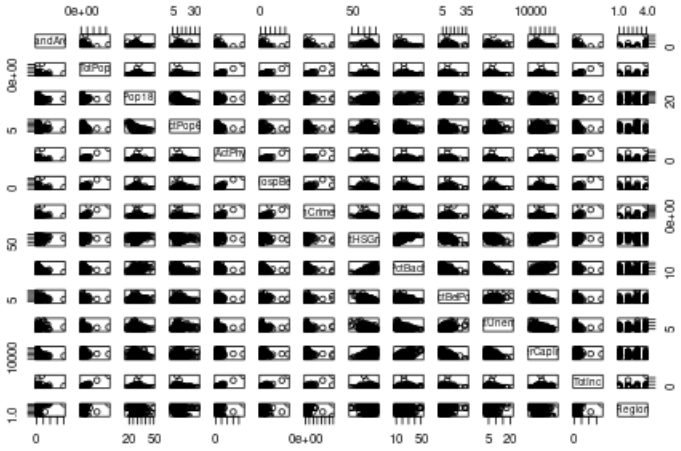
Using the same equation, we can say sales increase by approximately **9.366**.

---

### Exercise 10

```
cdi <- read.table("CDI.txt", header = TRUE)
```

```
# A

cdi %>%
    dplyr::select(-c(ID, County, State)) %>%
    pairs()
```



```
# B

cdi %>%
    dplyr::select(-c(ID, County, State)) %>%
    cor() %>%
    kableExtra::kbl() %>%
    kableExtra::kable_classic() %>%
    kable_styling(latex_options=c("scale_down", "hold_position"))
```

| | LandArea | TotPop | PctPop18_34 | PctPop65 | nActPhys | nHospBeds | nCrimes | PctHSGrad | PctBach | PctBelPov | PctUnemp | PerCapInc | TotInc | Region |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LandArea | 1.0000000 | 0.1730834 | -0.0548781 | 0.0057709 | 0.0780747 | 0.0730473 | 0.1294754 | -0.0985981 | -0.1372377 | 0.1713433 | 0.1992093 | -0.1877151 | 0.1270743 | 0.3628682 |
| TotPop | 0.1730834 | 1.0000000 | 0.0783721 | -0.0290374 | 0.9402486 | 0.9237384 | 0.8863318 | -0.0174269 | 0.1468138 | 0.0380195 | 0.0053517 | 0.2356102 | 0.9867476 | 0.0694371 |
| PctPop18_34 | -0.0548781 | 0.0783721 | 1.0000000 | -0.6163096 | 0.1196992 | 0.0745319 | 0.0899406 | 0.2505843 | 0.4560970 | 0.0339755 | -0.2785271 | -0.0316484 | 0.0711615 | 0.0524141 |
| PctPop65 | 0.0057709 | -0.0290374 | -0.6163096 | 1.0000000 | -0.0031286 | 0.0532784 | -0.0352903 | -0.2682518 | -0.3392288 | 0.0065785 | 0.2363094 | 0.0185907 | -0.0227332 | -0.1732916 |
| nActPhys | 0.0780747 | 0.9402486 | 0.1196992 | -0.0031286 | 1.0000000 | 0.9504644 | 0.8204595 | -0.0042481 | 0.2367655 | 0.0641363 | -0.0505161 | 0.3161346 | 0.9481106 | 0.0246461 |
| nHospBeds | 0.0730473 | 0.9237384 | 0.0745319 | 0.0532784 | 0.9504644 | 1.0000000 | 0.8568499 | -0.1119164 | 0.1004265 | 0.1727938 | 0.0075240 | 0.1948082 | 0.9020615 | -0.0031069 |
| nCrimes | 0.1294754 | 0.8863318 | 0.0899406 | -0.0352903 | 0.8204595 | 0.8568499 | 1.0000000 | -0.1063284 | 0.0770765 | 0.1644057 | 0.0435568 | 0.1175391 | 0.8430980 | 0.0913073 |
| PctHSGrad | -0.0985981 | -0.0174269 | 0.2505843 | -0.2682518 | -0.0042481 | -0.1119164 | -0.1063284 | 1.0000000 | 0.7077867 | -0.6917505 | -0.5935958 | 0.5229961 | 0.0433557 | -0.0100551 |
| PctBach | -0.1372377 | 0.1468138 | 0.4560970 | -0.3392288 | 0.2367655 | 0.1004265 | 0.0770765 | 0.7077867 | 1.0000000 | -0.4084238 | -0.5409069 | 0.6953619 | 0.2222301 | 0.0202990 |
| PctBelPov | 0.1713433 | 0.0380195 | 0.0339755 | 0.0065785 | 0.0641363 | 0.1727938 | 0.1644057 | -0.6917505 | -0.4084238 | 1.0000000 | 0.4369472 | -0.6017250 | -0.0387393 | 0.2709848 |
| PctUnemp | 0.1992093 | 0.0053517 | -0.2785271 | 0.2363094 | -0.0505161 | 0.0075240 | 0.0435568 | -0.5935958 | -0.5409069 | 0.4369472 | 1.0000000 | -0.3221444 | -0.0338763 | -0.0543786 |
| PerCapInc | -0.1877151 | 0.2356102 | -0.0316484 | 0.0185907 | 0.3161346 | 0.1948082 | 0.1175391 | 0.5229961 | 0.6953619 | -0.6017250 | -0.3221444 | 1.0000000 | 0.3476816 | -0.2224937 |
| TotInc | 0.1270743 | 0.9867476 | 0.0711615 | -0.0227332 | 0.9481106 | 0.9020615 | 0.8430980 | 0.0433557 | 0.2222301 | -0.0387393 | -0.0338763 | 0.3476816 | 1.0000000 | 0.0376855 |
| Region | 0.3628682 | 0.0694371 | 0.0524141 | -0.1732916 | 0.0246461 | -0.0031069 | 0.0913073 | -0.0100551 | 0.0202990 | 0.2709848 | -0.0543786 | -0.2224937 | 0.0376855 | 1.0000000 |

```
# C

act_phys_reg <- lm(nActPhys ~ TotPop + LandArea + TotInc, data = cdi)

act_phys_reg %>% summary()
```

```
##
## Call:
## lm(formula = nActPhys ~ TotPop + LandArea + TotInc, data = cdi)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1855.6  -215.2   -74.6    79.0  3689.0
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.332e+01  3.537e+01  -0.377 0.706719
## TotPop       8.366e-04  2.867e-04   2.918 0.003701 **
## LandArea    -6.552e-02  1.821e-02  -3.597 0.000358 ***
## TotInc       9.413e-02  1.330e-02   7.078 5.89e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 560.4 on 436 degrees of freedom
## Multiple R-squared:  0.9026, Adjusted R-squared:  0.902
## F-statistic:  1347 on 3 and 436 DF,  p-value: < 2.2e-16
```

The equation of this line is:

$$Y = (-1.332 * 10) + (8.336 \cdot 10^{-4} X_1) + (-6.552 \cdot 10^{-2} X_2) + (9.413 \cdot 10^{-2})$$

```
# D1
a <- 8.336 * 10^(-4) * 400000
b <- -6.552 * 10^(-2) * 1000
c <- 9.413 * 10^(-2) * 8000

-1.332 * 10 + a + b + c
```

```
## [1] 1007.64
```

```
# D2

predict(
    act_phys_reg,
    newdata = data.frame(
        TotPop = 400000,
        LandArea = 1000,
        TotInc = 8000
    )
)
```

```
##        1
## 1008.864
```

e)

The number of active physicians increases by $8.336 \cdot 10^{-4}$ for every additional personal added to the population.

f)

The number of active physicians increases by $9.413 \cdot 10^{-2}$ for every additional 1.0 million dollars in total personal income.

---

**Exercise 11**

```
cdi <- cdi %>%
    dplyr::mutate(PopDens = TotPop / LandArea)
```

```
act_phys_dens_reg <- lm(nActPhys ~ PopDens + PctPop65 + PerCapInc, data = cdi)
```

```
act_phys_dens_reg %>% summary()
```

```
##
## Call:
## lm(formula = nActPhys ~ PopDens + PctPop65 + PerCapInc, data = cdi)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4998.0  -522.8  -293.8    64.2 22067.6
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.088e+03  4.240e+02  -2.566   0.0106 *
## PopDens      2.873e-01  3.554e-02   8.083 6.27e-15 ***
## PctPop65    -7.964e+00  1.900e+01  -0.419   0.6753
## PerCapInc    1.033e-01  1.921e-02   5.377 1.24e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1589 on 436 degrees of freedom
## Multiple R-squared:  0.2173, Adjusted R-squared:  0.2119
## F-statistic: 40.35 on 3 and 436 DF,  p-value: < 2.2e-16
```

The equation of this plane is:

$$Y = -1087.8142 + 0.2873X_1 - 7.9640X_2 + 0.1033X_3$$

```
# B1

- 1087.8142 + (0.2873*900) - (7.9640*15) + (0.1033*20000)
```

```
## [1] 1117.296
```

```
# B2

predict(
    act_phys_dens_reg,
    newdata = data.frame(
        PopDens = 900,
        PctPop65 = 15,
        PerCapInc = 20000
    )
)
```

```
##        1
## 1117.385
```

---

**Exercise 12**

a)

The residual standard error of the model using total population has a residual standard error of **560.4 on 436 degrees of freedom**. The residual standard error of the other model has an RSE of **1589 on 436 degrees of freedom**. This shows that the first model we used was probably the better model to use.

b)

Total Population Model:

$$R^2 = .9026$$

Population Density Model:

$$R^2 = .2173$$

This mirrors our results from part a, this also shows the total population model as being a better predictor for the number of active physicians.

c)

Total population is a better predictor for the number of active physicians. Population density does not seem to be a good predictor of this information at all.

---

**9.7: Logistic Regression: Dichotomous (0 or 1) Response Variable.**