Question 1

Review the basics of summation notation and covariance formulas. Show that:

a. $\sum_{i=1}^{N}(Y_i - \bar{Y}) = 0$
b. $\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y}) = \sum_{i=1}^{N}(X_i - \bar{X})Y_i$

Solution 1:

a. $\sum_{i=1}^{N}(Y_i - \bar{Y}) = 0$

a.) To show $\sum_{i=1}^{N}(Y_i - \bar{Y}) = 0$

Expanding the equation, we can see the following

$= \sum_{i=1}^{N}(Y_i - \bar{Y}) = (Y_1 - \bar{Y}) + (Y_2 - \bar{Y}) + (Y_3 - \bar{Y}) + \dots (Y_N - \bar{Y})$ (1)

We know that $\bar{Y} = \frac{\sum_{i=1}^{N} Y_i}{N}$ (2)

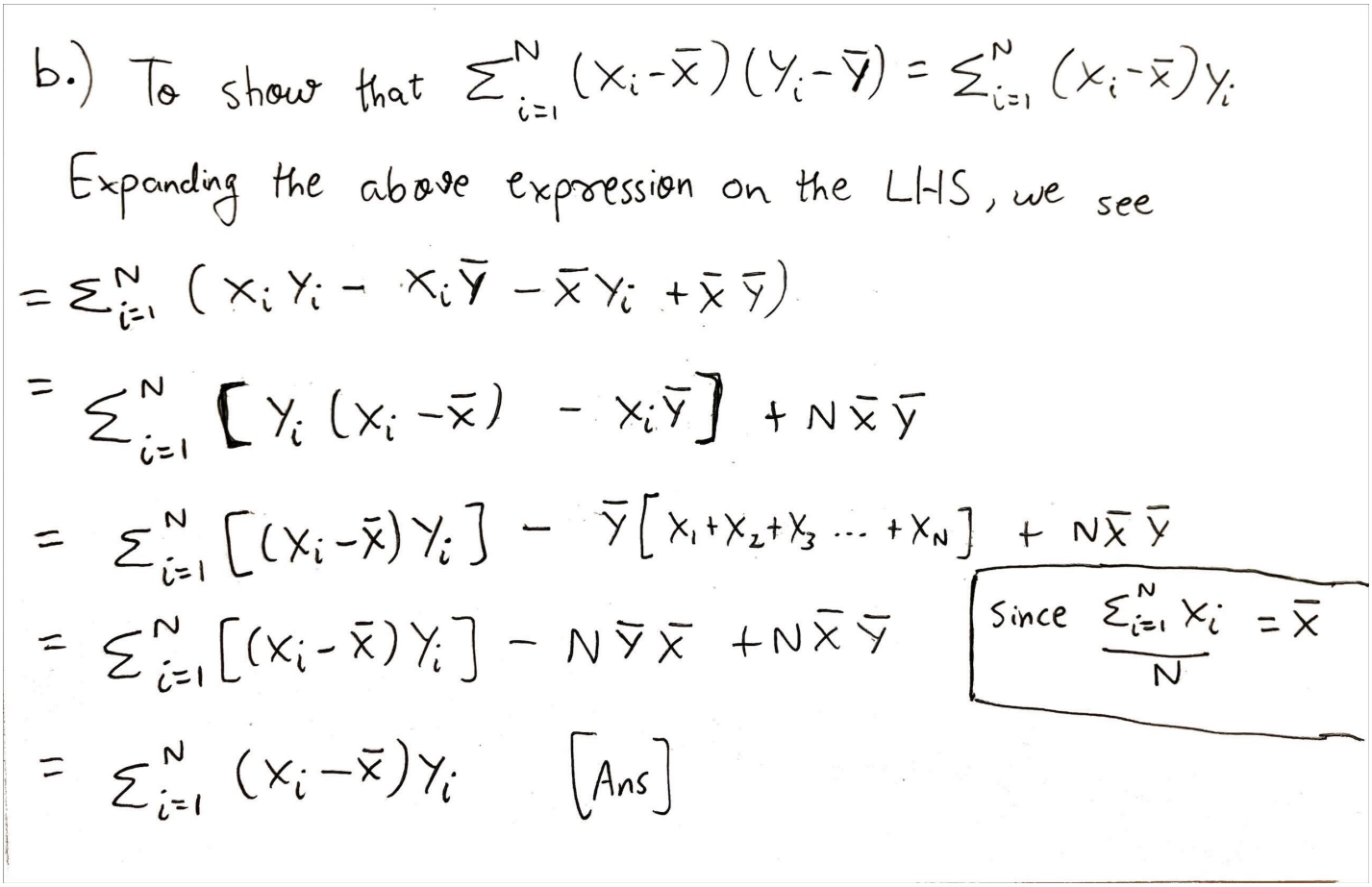
Hence, by rearranging the above expression,

$= (Y_1 + Y_2 + Y_3 + Y_4 + \dots Y_N) - N\bar{Y}$ from (1)

$= N\bar{Y} - N\bar{Y} = 0$

from (2)

b. $\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y}) = \sum_{i=1}^{N}(X_i - \bar{X})Y_i$



Question 2

Define both and explain the difference between (a) the expectation of a random variable and (b) the sample average?

Solution 2:

(a) The expectation of a random variable is defined as the weighted average of values of a random variable, which is weighted by the probability of its occurance

(b) The sample average is defined as the simple mean of a sample from a probability distribution

Difference between Expected Value of a random variable and the Sample Average?
Let's consider a Normal Distribution N(0,1) with mean equals 0 and standard deviation equals 1. Furthermore, lets take a sample of 10 from the above distribution for this example.

Sample of 10 observations from a normal distribution

```
rnorm(10, mean=0, sd=1)
```

```
[-0.24, 0.58, -0.76, 0.59, -2.28, -0.73, 2.47, 0.89, 1.90, -1.61]
```

The difference between Expected Value and Sample Average in the context of this example would be-

1) The sample average depends on the sample taken from the population. Every time we take a sample from a population, the sample average will change. In the above sample, the sample average is 0.082

2) The expected value of a random variable from the above distribution N(0,1) would essentially be equal to the mean, which is 0. It is evaluated based on the weighted average of all values the random variable can take and its probability. Since a random variable X ~ N(0,1) has the distribution centered around 0 and is symmetric on both sides, the Expected Value E(X) is equal to the mean, which is 0. It is a fixed value for a given distribution and doesn't depend on sampling

---

Question 3

a. Describe the Central Limit Theorem as simply as you can.

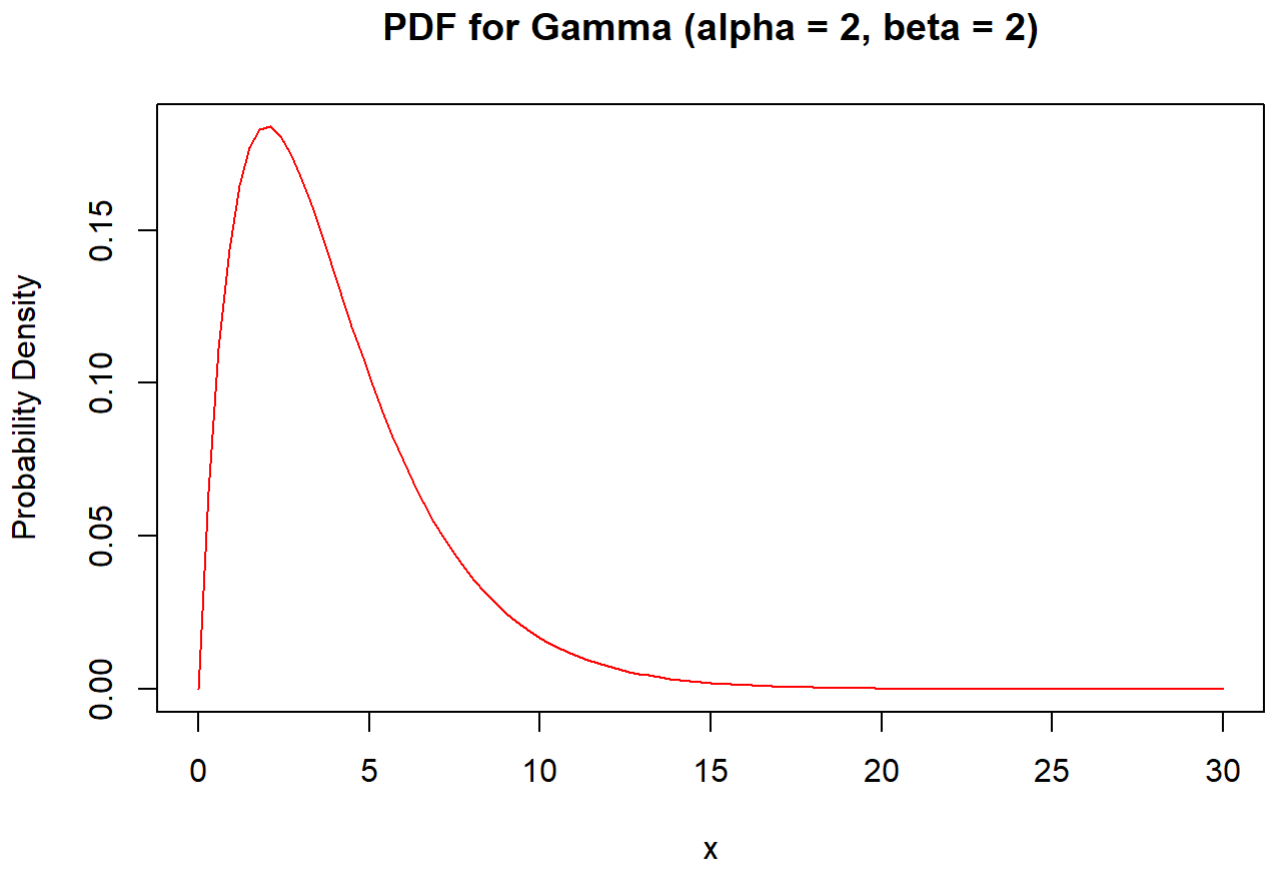
Solution 3.a.)

The Central Limit Theorem states that regardless of the distribution from which we take a sample, the distribution of the sample means will tend towards an approximately normal distribution as we increase the sample size. Furthermore, the mean of all sampled variables from the same distribution will be approximately equal to the population mean and the variances will approximately be equal to the variance of the population as the sample size gets larger, according to the law of large numbers.

b. Let $X \sim \text{Gamma}(\alpha = 2,\ \beta = 2)$. For the Gamma distribution, α is often called the "shape" parameter, β is often called the "scale" parameter, and the $\mathbb{E}[X] = \alpha\beta$. Plot the density of X and describe what you see. You may find the functions `dgamma()` or `curve()` to be helpful.

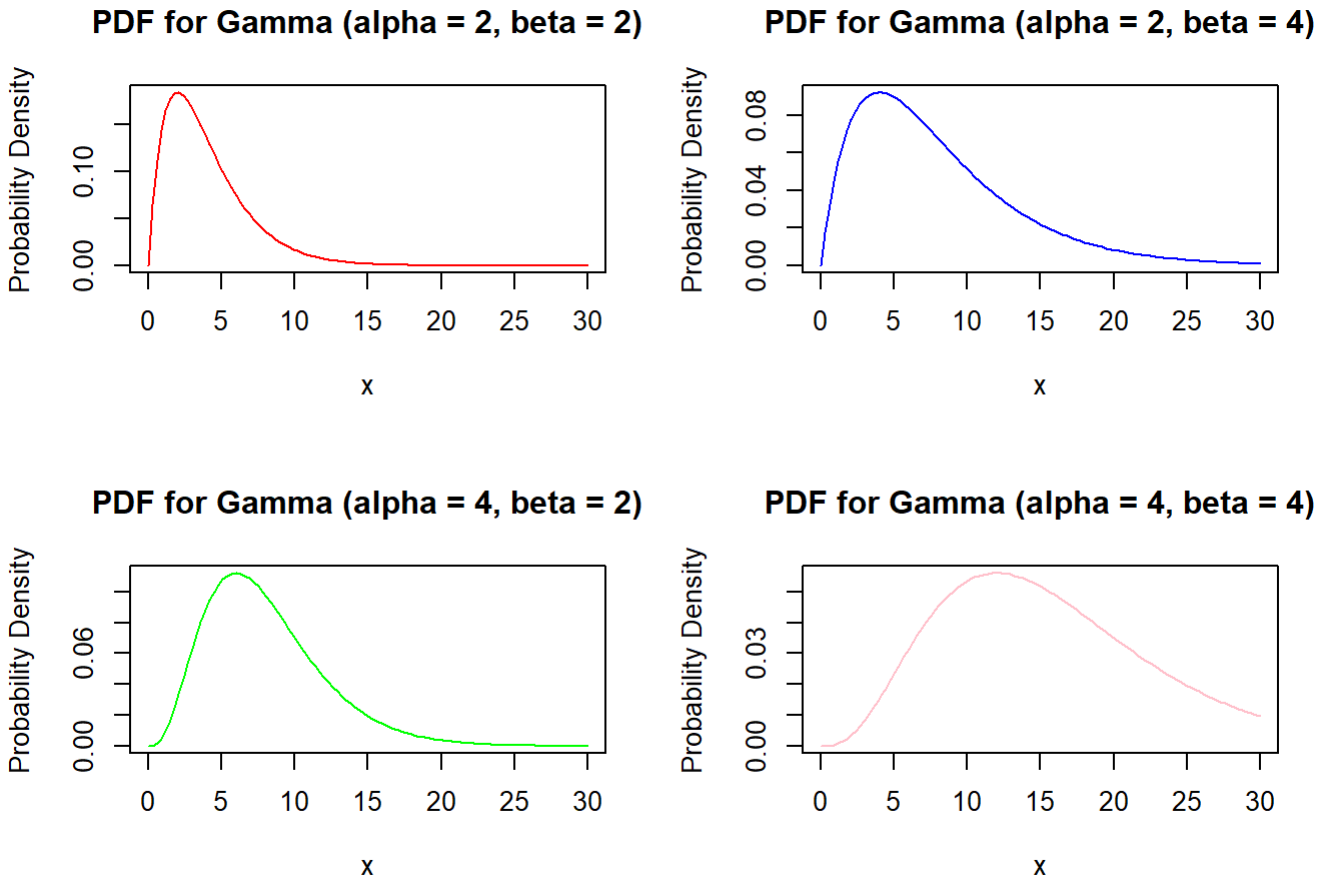
Solution 3.b.)

```
# Answer:
library(repr)
options(repr.plot.width = 10, repr.plot.height = 6)

# Plotting the density of gamma function with shape parameter = 2
# and rate (which is 1/scale) = 0.5
curve(dgamma(x, shape = 2, rate = 0.5), xlim = c(0,30), type = 'l', col = 'red',
      main = "PDF for Gamma (alpha = 2, beta = 2)", ylab = "Probability Density")
```



If we change the shape and scale parameters of the gamma distribution, then we observe the following

```
pl_par = par(mfrow=c(2, 2))
curve(dgamma(x, shape = 2, scale = 2), xlim = c(0,30), type = 'l', col = 'red',
      main = "PDF for Gamma (alpha = 2, beta = 2)", ylab = "Probability Density")

curve(dgamma(x, shape = 2, scale = 4), xlim = c(0,30), type = 'l', col = 'blue',
      main = "PDF for Gamma (alpha = 2, beta = 4)", ylab = "Probability Density")

curve(dgamma(x, shape = 4, scale = 2), xlim = c(0,30), type = 'l', col = 'green',
      main = "PDF for Gamma (alpha = 4, beta = 2)", ylab = "Probability Density")

curve(dgamma(x, shape = 4, scale = 4), xlim = c(0,30), type = 'l', col = 'pink',
      main = "PDF for Gamma (alpha = 4, beta = 4)", ylab = "Probability Density")
```



Notes: In the above PDF plot for Gamma Distribution, we can observe that the curve is skewed and has a long right-tail. The curve rapidly rises to its peak value and decreases with a long right-tail.

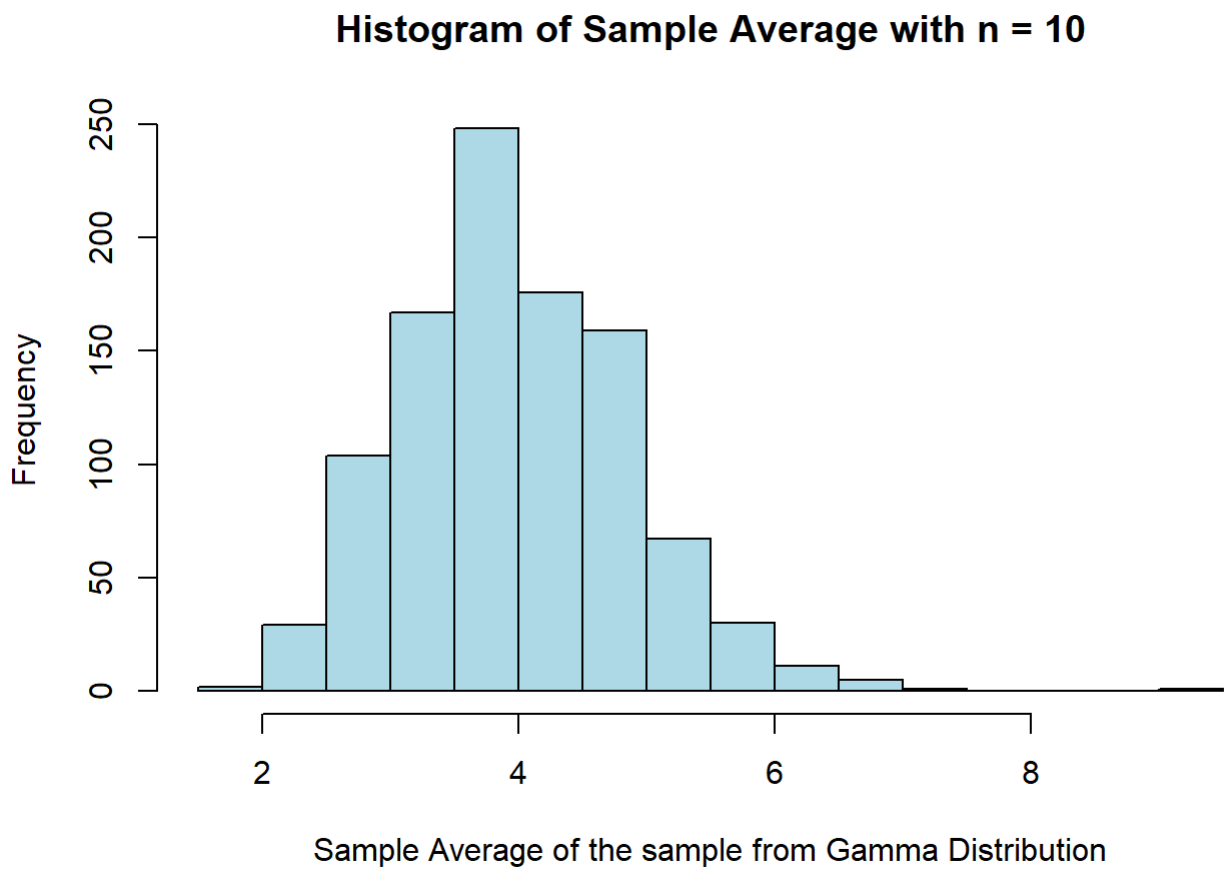
The influence of shape and scale parameter:

- Increasing the value of shape parameter shifts the peak of the distribution. As shape parameter increases, the peak shifts to the left as the expected value of the distribution increases (assuming scale parameter remains constant)
- Increasing the value of scale parameter reduces the peakedness of the distribution and the peak widens. Essentially, the distribution spreads as scale parameter increases

c. Let n be the number of draws from that distribution in one sample and r be the number of times we repeat the process of sampling from that distribution. Draw an iid sample of size $n = 10$ from the Gamma(2,2) distribution and calculate the sample average; call this $\bar{X}_n^{(1)}$. Repeat this process r times where $r = 1000$ so that you have $\bar{X}_n^{(1)}, \ldots, \bar{X}_n^{(r)}$. Plot a histogram of these r values and describe what you see. This is the sampling distribution of $\bar{X}_{(n)}$.

Solution 3.c.)

```
# Answer:

# Creating an empty list to store the sample averages
sample_avg_list <- rep(NA, 1000)

# Running a loop x1000 times to generate a sample of n=10 from a
# gamma distribution and calculate it's sample average
for (i in 1:1000) {
    y_val = rgamma(10, shape = 2, rate = 0.5)
    sample_avg = sum(y_val)/length(y_val)
    sample_avg_list[i] = sample_avg
}

# Plotting the histogram of sample averages from the above
# gamma distribution
options(repr.plot.width = 10, repr.plot.height = 6)
hist(sample_avg_list, col = 'light blue',
     main = "Histogram of Sample Average with n = 10",
     xlab = 'Sample Average of the sample from Gamma Distribution')
```

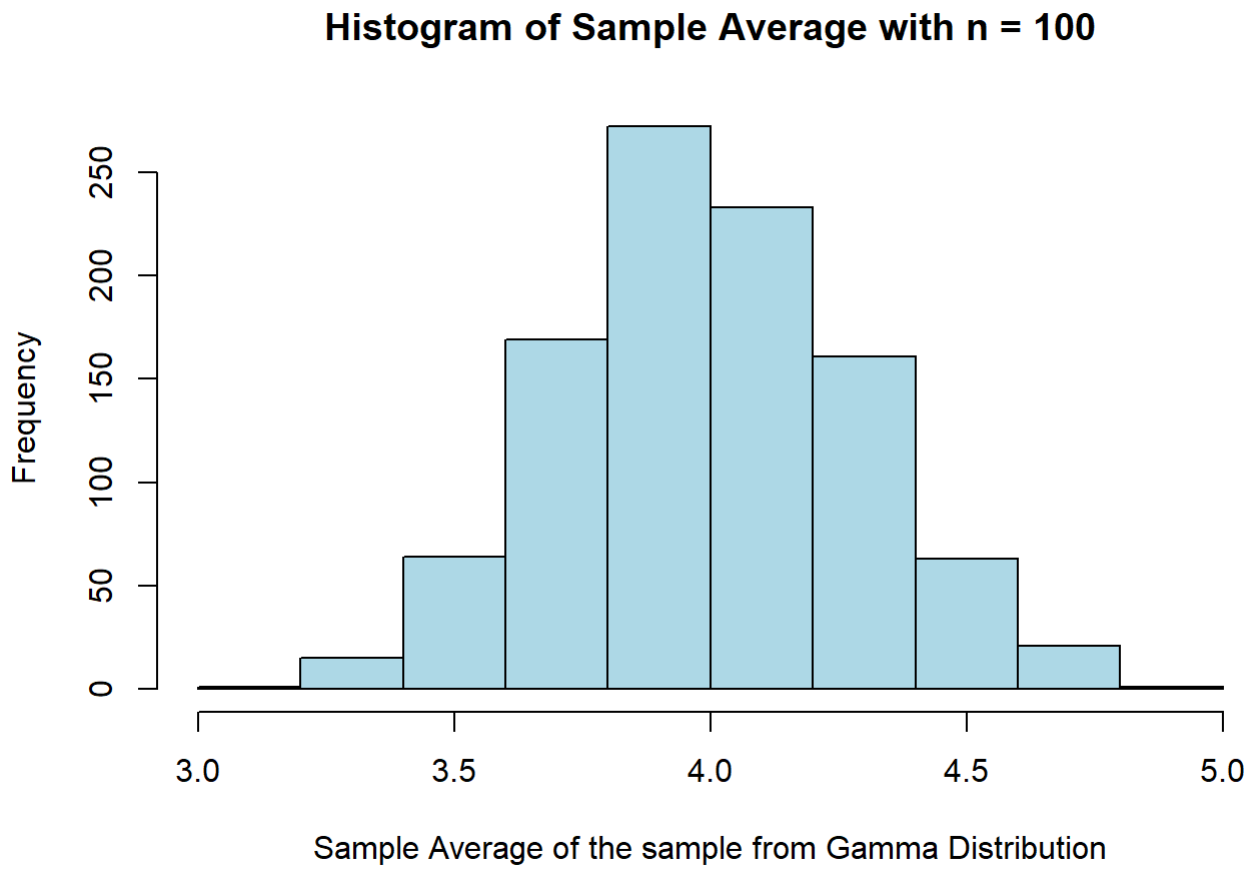


Notes: We observe that the distribution of sample means is approximately resembling a normal distribution. Even though the sampling was done from a gamma distribution, the sample means distribution resembles a normal distribution. As we increase the sample size, the distribution of sample means will tend towards a normal distribution (*as stated by Central Limit Theorem*)

d. Repeat part (c) but with $n = 100$. Be sure to produce and describe the histogram. Explain how this illustrates the CLT at work.

Solution 3.d.)

```
# Answer:

# Creating an empty list to store the sample averages
sample_avg_list <- rep(NA, 1000)

# Running a loop x1000 times to generate a sample of n=100
# from a gamma distribution and calculate it's sample average
for (i in 1:1000) {
    y_val = rgamma(100, shape = 2, rate = 0.5)
    sample_avg = sum(y_val)/length(y_val)
    sample_avg_list[i] = sample_avg
}

# Plotting the histogram of sample averages from the above
# gamma distribution
options(repr.plot.width = 10, repr.plot.height = 6)
hist(sample_avg_list, col = 'light blue',
     main = "Histogram of Sample Average with n = 100",
     xlab = 'Sample Average of the sample from Gamma Distribution')
```



Notes: As we increase the sample size from 10 to 100, we reduce the standard error of mean by a factor of the square root of 10. When we look at distribution of the new sample means (with n = 100), the distribution closely resembles a normal distribution. This exercise shows CLT at work. CLT states that as we increase the sample size of random variables from any distribution, the distribution of the sample means will tend towards a normal distribution.

Hence, we can see that the distribution of sample means (n=100) is a better approximation of normal distribution as compared to that of sample means (n=10)

Question 4

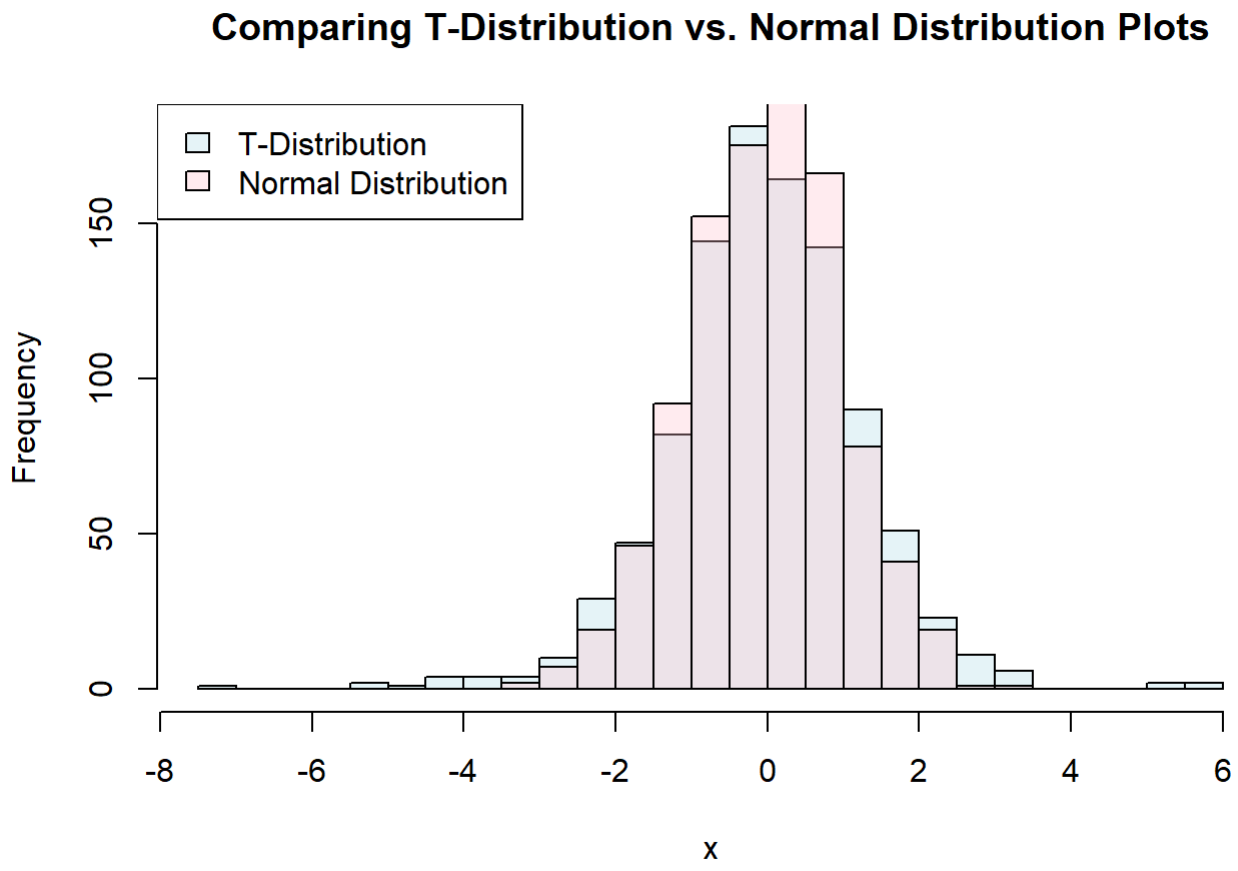
The normal distribution is often said to have “thin tails” relative to other distributions like the t-distribution. Use random number generation in R to illustrate that a $\mathcal{N}(0,1)$ distribution has much thinner tails than a t-distribution with 5 degrees of freedom.

A few coding hints: `rnorm()` and `rt()` are the functions in R to draw from a normal distribution and a t-distribution. The option `add=TRUE` for the `hist()` command can be used to overlay a second histogram on top of another histogram, and after installing the `scales` package, you can make a blue histogram 50% transparent with the option `col=scales::alpha("blue",0.5)`. Alternatively, you can put two plots side-by-side by first setting the plotting parameter with the code `par(mfrow=c(1,2))`. You can set the range of the x-axis to go from -5 to 5 with the plotting option `xlim=c(-5,5)`.

Solution 4:

```
# Sampling from T-distribution and from Normal Distribution
t_dist_values = rt(n= 1000,df = 5)
norm_dist_values = rnorm(1000,0,1)
```

```
options(repr.plot.width = 12, repr.plot.height = 8)
c1 = rgb(173,216,230,max = 255, alpha = 80, names = "lt.blue")
c2 = rgb(255,192,203, max = 255, alpha = 80, names = "lt.pink")

min_val = min(c(t_dist_values, norm_dist_values))
max_val = max(c(t_dist_values, norm_dist_values))
ax = pretty(min_val:max_val, n = 30)


t_plot = hist(t_dist_values,
              plot = FALSE, breaks = ax)

norm_plot = hist(norm_dist_values,
                 plot = FALSE,
                 breaks = ax)

plot(t_plot,
     main = "Comparing T-Distribution vs. Normal Distribution Plots",
     col = c1,
     xlab = "x",
     ylab = "Frequency")

plot(norm_plot, col = c2, add = TRUE)

legend(x = "topleft",
       legend=c("T-Distribution", "Normal Distribution"),
       fill = c(c1,c2))
```



Notes: In the above plot, we observe that the normal distribution has thinner tails relative to the t-distribution!

Question 5

a. From the Vanguard dataset, compute the standard error of the mean for the `VFIAX` index fund return.
b. For this fund, the mean and the standard error of the mean are almost exactly the same. Why is this a problem for a financial analyst who wants to assess the performance of this fund?
c. Calculate the size of the sample which would be required to reduce the standard error of the mean to 1/10th of the size of the mean return.

Solution 5:

```
# Loading the required libraries
library(DataAnalytics)
library(reshape2)

# Loading the Vanguard data
data(Vanguard)
head(Vanguard, n = 5)
```

	date <date>	ticker <chr>	crsp_fundno <int>	mtna <dbl>	mret <dbl>
1	1988-04-29	VEIPX	31217	*NA*	0.010215
2	1988-05-31	VEIPX	31217	*NA*	0.027300
3	1988-06-30	VEIPX	31217	16.763	0.030535
4	1988-07-29	VEIPX	31217	*NA*	0.005797
5	1988-08-31	VEIPX	31217	*NA*	-0.013449

5 rows

```
# Filtering the rows in the table to only retain ticker = 'VFAIX'
Van_vfiax = Vanguard[Vanguard$ticker == 'VFIAX',]
Van_vfiax = Van_vfiax[ , c("date","ticker","mret")]
head(Van_vfiax, n=5)
```

	date <date>	ticker <chr>	mret <dbl>
304	2000-12-29	VFIAX	0.005223
305	2001-01-31	VFIAX	0.035448
306	2001-02-28	VFIAX	-0.091370
307	2001-03-30	VFIAX	-0.063425
308	2001-04-30	VFIAX	0.077606

5 rows

```
# Reshaping the table to shift ticker to columns and
# fill with monthly returns
Van_vfiax_reshaped = dcast(Van_vfiax, date~ticker,
                           value.var="mret")
head(Van_vfiax_reshaped, n=5)
```

	date <date>	VFIAX <dbl>
1	2000-12-29	0.005223
2	2001-01-31	0.035448
3	2001-02-28	-0.091370
4	2001-03-30	-0.063425
5	2001-04-30	0.077606

5 rows

```
# Descriptive statistics for the VFAIX ticker monthly data
descStat(Van_vfiax_reshaped)
```

```
##        Mean Median    SD  IQR SE Mean 95% CI-L 95% CI-U NMissing
## VFIAX 0.004  0.011 0.045 0.05   0.004   -0.003    0.011        0
## Number of Observations =  151
```

a. From the Vanguard dataset, compute the standard error of the mean for the `VFIAX` index fund return.
Ans. The standard error of the mean is 0.004

b. For this fund, the mean and the standard error of the mean are almost exactly the same. Why is this a problem for a financial analyst who wants to assess the performance of this fund?
Ans. The standard error of the mean is of the same magnitude as that of the mean of monthly returns. This implies that the true mean monthly return for the VFIAX fund can significantly deviate from the estimated mean. Therefore, a financial analyst cannot confidently trust the mean monthly return for this fund. The 95% confidence interval varies from -0.003 to 0.011. Hence, investors can also lose money by investing in this fund.

c. Calculate the size of the sample which would be required to reduce the standard error of the mean to 1/10th of the size of the mean return.
Ans.

The standard error of the mean is defined as follows-

$$S_{\bar{Y}} = \frac{S_Y}{\sqrt{N}}$$

So, if we want to reduce the standard error of the mean to 1/10th of the size of the mean return, then the number of observations required would be

```
# Storing the summary statistics for the dataset
data_stats = descStat(Van_vfiax_reshaped)
```

```
##        Mean Median    SD  IQR SE Mean 95% CI-L 95% CI-U NMissing
## VFIAX 0.004  0.011 0.045 0.05   0.004   -0.003    0.011        0
## Number of Observations =  151
```

```
# Required Standard Error of the Sample Mean is-
required_se_sample_mean = data_stats["VFIAX","Mean"]/10

# The Sample Standard Deviation is-
sample_sd = data_stats["VFIAX","SD"]

# As per the relationship between Sample Mean SE and Sample SD, the number of observations neede
d would be-
number_obs = ((sample_sd/required_se_sample_mean)^2)

print(paste0("The number of observations needed would be: ", sprintf(number_obs, fmt = '%#.0f'
)))
```

```
## [1] "The number of observations needed would be: 12656."
```

The key takeaway here is that to reduce the error to it's 1/10th value, we would need approximately 100x more observations (which is very high).

Question 6 : Subsetting Observations

Q6, Part A

1. Display the contents of the first 50 elements of the vector, `cars$make == "Ford"` , to verify that it is a logical vector.

```
# Q.6 Part A (1)
# Loading mvehicles dataset
data(mvehicles)

# Filtering only cars from the mvehicles dataset
cars = mvehicles[mvehicles$bodytype != "Truck",]

# Printing the logical vector for cars$make == "Ford"
head(cars$make == "Ford", n=50)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## [37]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE
```

2. Subset the `cars` data frame by a two step process to only the "Ford" make. That is, create the row selection logical vector in one statement and select observations from the `cars` data frame in the second.

```
# Q.6 Part A (2)
# Creating logical vector where 'make' == 'Ford'
ford_logical = (cars$make == "Ford")

# Filtering using the above logical vector
ford = cars[ford_logical,]
head(ford, n = 3)
```

	…	y…	mo…	style	origin	bodyty…	emv	se…	rc
	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	<dbl>	<int>	
6	Ford	2011	Edge	Limited 4dr SUV (3.5L 6cyl 6A)	America	SUV	35441.33	5	0
7	Ford	2011	Edge	Limited 4dr SUV AWD (3.5L 6cyl 6A)	America	SUV	38441.82	5	0
8	Ford	2011	Edge	SE 4dr SUV (3.5L 6cyl 6A)	America	SUV	26999.12	5	0

3 rows | 1-10 of 18 columns

3. How many Kia observations are there in the `cars` data frame? hint: `nrow()` tells you how many rows are in a data frame.

```
# Q.6 Part A (3)
print(paste0("Number of rows in 'cars' dataset where the",
             " 'make' = Kia is ",
            nrow(cars[cars$make == "Kia",])))
```

```
## [1] "Number of rows in 'cars' dataset where the 'make' = Kia is 43"
```

4. How many cars are have a price (emv) that is greater than $100,000?

```
# Q.6 Part A (4)
print(paste0("Number of cars with expected market value (emv)",
             " greater than $100,000 is ",
            nrow(cars[cars$emv > 100000,])))
```

```
## [1] "Number of cars with expected market value (emv) greater than $100,000 is 37"
```

We can also couple two logical expressions together using AND `&` or OR `|`. For example, if we want to select all rows with either Kia or Hyundai; we would say `cars[cars$make == "Kia" | cars$make == "Hyundai",]`.

Q6, part B

1. What is the average sales for all cars made in Europe with price above $75,000?

In many data sets, there are long text fields whcih describe an observation. These fields are not formatted in any way and so it is difficult to use simple comparison methods to fetch observations. However, we can use the power of something called regular expressions to find any observations for which a given variable contains some character pattern. Regular expressions are very complicated to use in generality but we can get a lot of use out of a very simple expression.

The `style` variable in `cars` is a general text description variable, We can find the rows for each `style` contains any string by using the command `grepl("string",column,ignore.case=TRUE)` . For example, `grepl("hybrid",cars$style,ignore.case=TRUE)` creates a logical vector (TRUE or FALSE) to help select rows corresponding to hybrids. `cars[grepl("hybrid",cars$style,ignore.case=TRUE),]` will fetch only hybrids.

```
print(paste0("Average sales of all cars made in Europe with price above $75,000: ",
             sprintf(mean(cars[(grepl("Europe",cars$origin,ignore.case = TRUE) &
                        (cars$emv > 75000)),c("sales")]), fmt = '%#.0f')))
```

```
## [1] "Average sales of all cars made in Europe with price above $75,000: 627."
```

Q 6, part C

1. How many four door vehicles are in cars?
2. How many four door sedans are in cars?

```
# To identify cars with 4 doors by searching for '4dr' in the 'style' column of the dataset
print(paste0("Number of cars with four doors: ",
             nrow(cars[grepl("4dr",cars$style,ignore.case = FALSE),])))
```

```
## [1] "Number of cars with four doors: 1105"
```

```
# To identify Sedan with 4 doors
print(paste0("Number of Sedans with four doors: ",
             nrow(cars[(grepl("4dr",cars$style,ignore.case = FALSE) &
                        grepl("Sedan",cars$bodytype,ignore.case=TRUE)),])))
```

```
## [1] "Number of Sedans with four doors: 432"
```

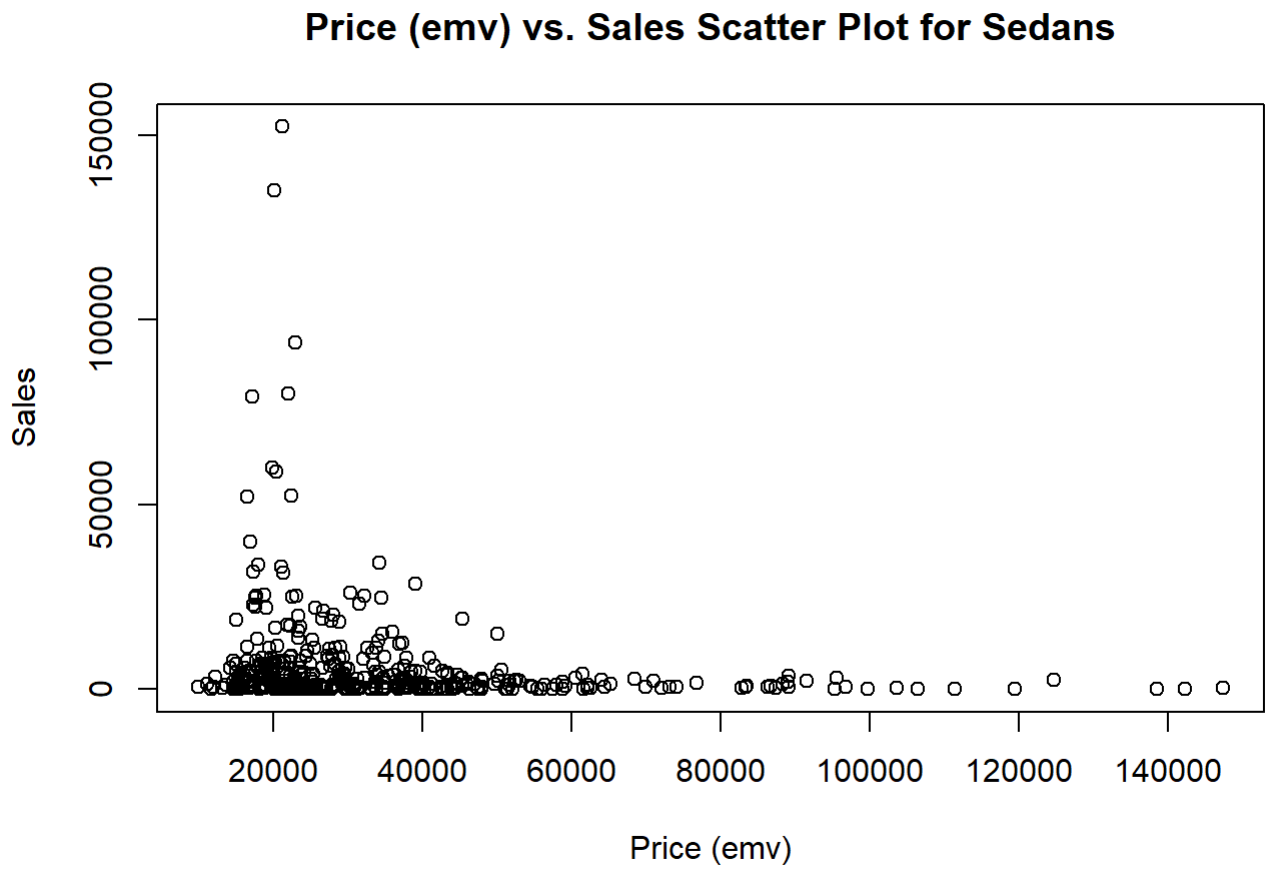
Question 7 : Sales and Price relationships

In this question, use `cars` only.

Q7, part A

Plot price (horizontal axis) vs. sales (vertical axis) for cars with bodytype == “Sedan”. What is the problem with displaying the data in this manner?

```
# Filtering the cars dataset to only retain "Sedan"
car_dat = cars[grepl("Sedan",cars$bodytype,ignore.case=TRUE),
               c("emv", "sales")]

# Plotting Price (emv) vs. Sales
plot(x = car_dat$emv, y = car_dat$sales,
     main = "Price (emv) vs. Sales Scatter Plot for Sedans",
     ylab = "Sales", xlab = "Price (emv)")
```

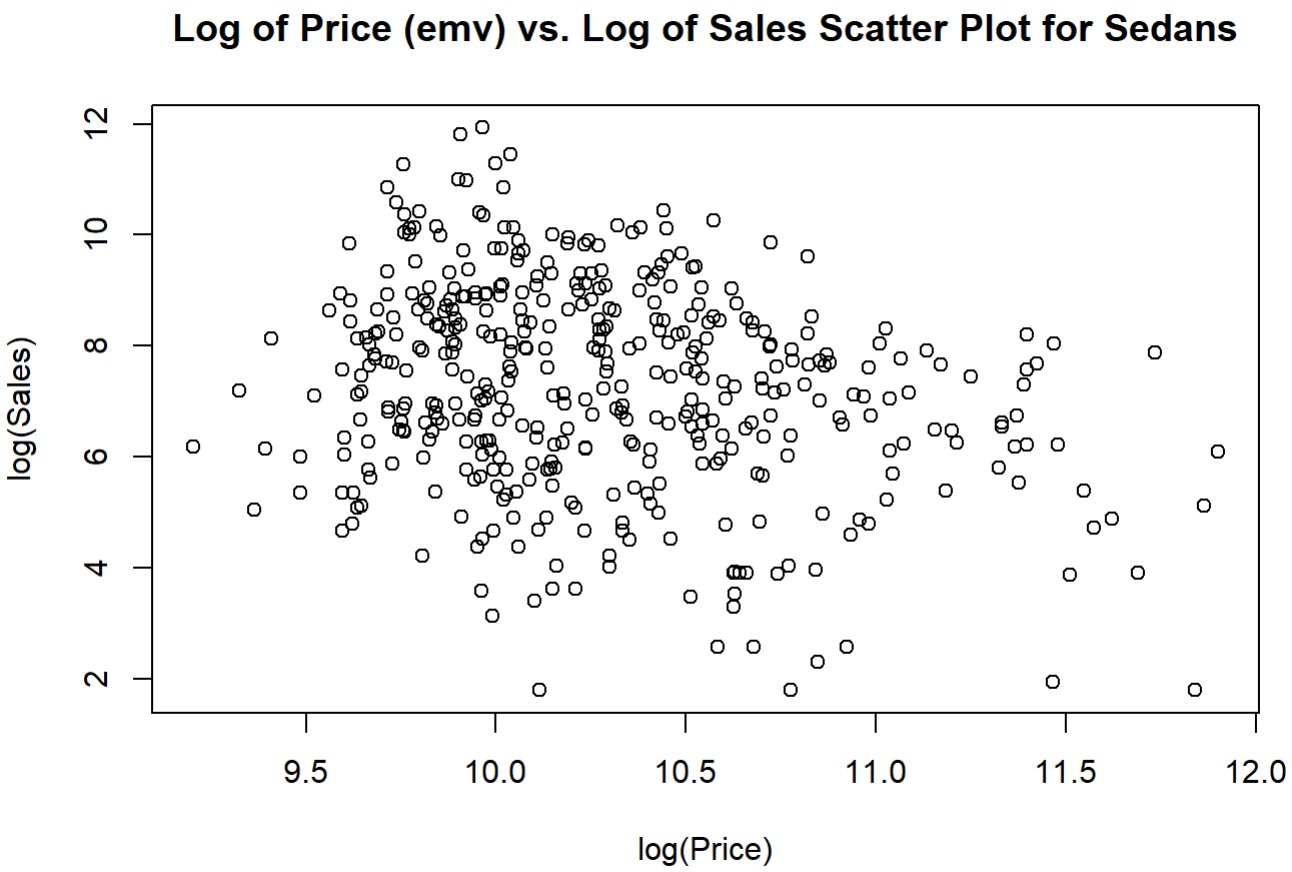


Notes: The problem with displaying data in this manner is that we cannot extract any meaningful insights from the plot. Both “Sales” and “Price” have a large range of values, where most of the data points lie on the lower end. Hence, when we plot data in this manner, we cannot extract any meaningful insigths regarding the relationship between Sales and Price.

Q7, part B

Plot log(price) vs. log(sales) for the same subset of observations as in part 1. How has this improved the visualization of this data? Are there any disadvantages of taking the log transformation? A very similar but less "violent" tranformation is the sqrt transformation. Try the sqrt transformation. Is this useful?

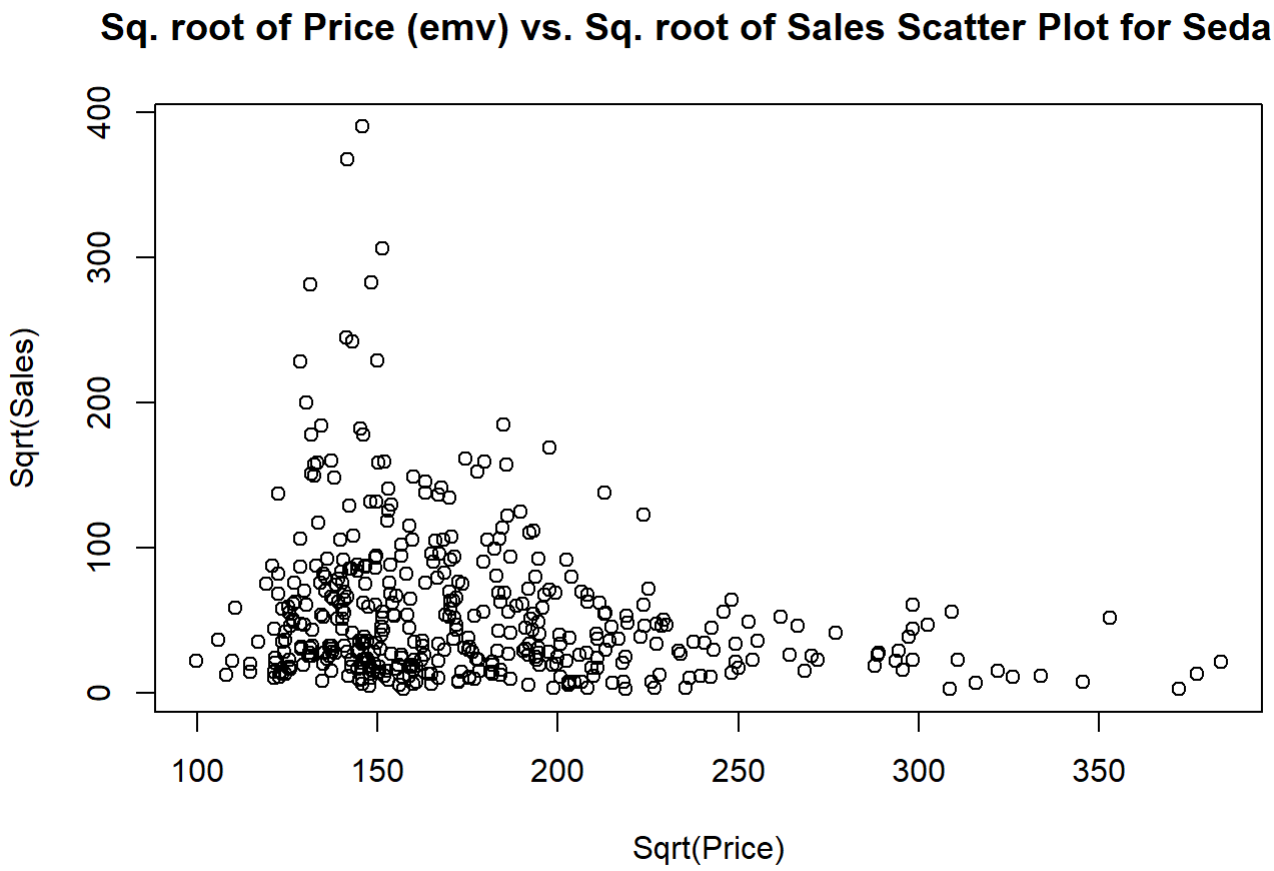
```
# Plotting Log(Price) vs. Log(Sales)
plot(x = log(car_dat$emv), y = log(car_dat$sales),
     main = "Log of Price (emv) vs. Log of Sales Scatter Plot for Sedans",
     ylab = "log(Sales)", xlab = "log(Price)")
```



Notes: By taking a log transformation of Sales and Price, we can observe all data points in a much better manner. We can also extract some meaningful insights from this plot. We observe that as price increases, the sales decrease in general. Log transformation has helped scale down outliers in the data and bring them together with the larger population of data points.

The disadvantage of doing the log transformation is that the numbers won't be intuitive. So we cannot properly interpret log(Sales) and log(Price). E.g. It is difficult to compare `log(Price) = 12` vs. `log(Price) = 9` and how that impacts sales

```
# Plotting Sq. Root(Price) vs. Sq. Root(Sales)
plot(x = sqrt(car_dat$emv), y = sqrt(car_dat$sales),
     main = "Sq. root of Price (emv) vs. Sq. root of Sales Scatter Plot for Sedans",
     ylab = "Sqrt(Sales)", xlab = "Sqrt(Price)")
```



Notes: The square root transformation helps by bringing the data points with large Price or Sales closer to the remaining data points on the lower end of the scale. It also closely resembles the distribution pattern observed in the original scatter plot. However, it still suffers from interpretability. The units of the values (e.g. $) don't make sense when we take the square root.

Q7, part C

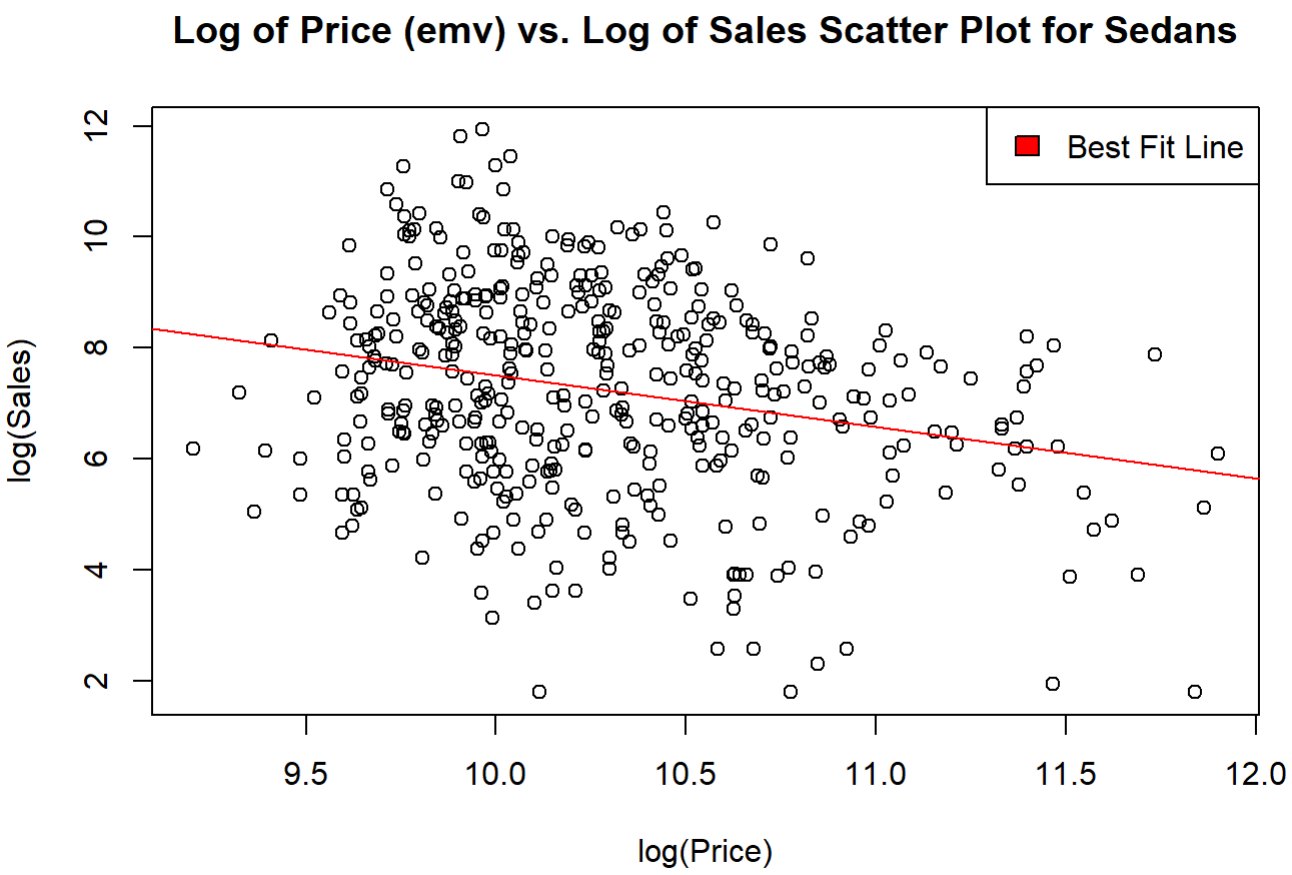
Economists will tell you that as price increase sales will decreases, all other things being equal. Does this plot support this conclusion?

Ans) In the above plots we can observe that in general as the price increases the Sedan sales decrease

Q7, part D

Fit a regression model to this data. That is, "regress" log(sales) on log(price) (log(sales) is Y or the dependent variable). Plot the fitted line on top of the scatterplot using `abline` .

```
# Plotting the best fit line
plot(x = log(car_dat$emv), y = log(car_dat$sales),
     main = "Log of Price (emv) vs. Log of Sales Scatter Plot for Sedans",
     ylab = "log(Sales)", xlab = "log(Price)")
abline(lm(log(sales) ~ log(emv), data = car_dat), col = "red")
legend(x = "topright", legend=c("Best Fit Line"), fill = c("red"))
```



Q7, part E

Predict sales for price = $45,000 using the model fit in part D). Don't forget to transform back to unit sales by using the `exp()` function.

```
# Fitting a line using linear model
regress_sales_price = lm(log(sales) ~ log(emv), data = car_dat)
regress_sales_price
```

```
##
## Call:
## lm(formula = log(sales) ~ log(emv), data = car_dat)
##
## Coefficients:
## (Intercept)     log(emv)
##     16.8321      -0.9321
```

```
# Predicting the sales of a Sedan which is priced at $45,000
print(paste0("The predicted sales of a Sedan which is priced at $45,000 is ",
             sprintf(exp(predict(regress_sales_price, new = data.frame(emv = 45000))),
                     fmt = '%#.0f')))
```

```
## [1] "The predicted sales of a Sedan which is priced at $45,000 is 940."
```