Homework 1
MSBA 400: Statistical Foundations for Data Analytics
Professor Rossi

Question 1

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

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

   Solution:

   $\sum_{i=1}^{N}(Y_i - \bar{Y}) = \sum_{i=1}^{N} Y_i - \sum_{i=1}^{N} \bar{Y} = (Y_1 + Y_2 + ... + Y_N) - N\bar{Y} = N\frac{(Y_1+Y_2+...+Y_N)}{N} - N\bar{Y} = N\bar{Y} - N\bar{Y} = 0.$

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

   Solution:

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

Question 2

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

Solution:

The expectation of a random variable is a measure of the most likely value of the variable and is computed by summing all possible values of the variable multiplied by their weights (represented by an integral for continuous variables and by a summation for discrete variables). It is the mean of the distribution of that variable.

The sample average is the mean of N samples chosen from that distribution.

The difference is that expectation is the mean of the entire distribution, while the sample average is the mean of a subset of samples taken from the distribution. Thus, they can (and often do) take on different values.

Question 3

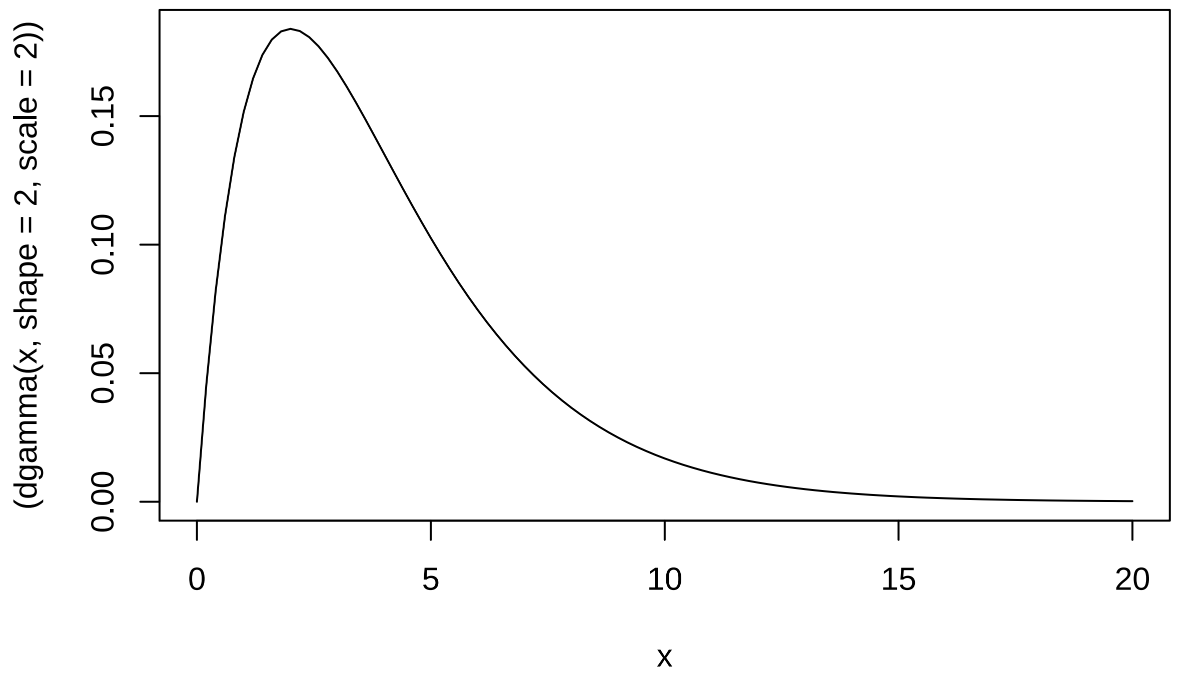
a. Describe the Central Limit Theorem as simply as you can.

   Solution: Put simply, the Central Limit Theorem states that the sample mean of N iid observations approaches a normal distribution as N increases.

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 $\mathbb{E}[X] = \alpha\beta$. Plot the density of X and describe what you see. You may find the functions `dgamma()` or `curve()` to be helpful.

   Solution:

```r
# generate distribution, plot curve
curve((dgamma(x, shape=2, scale=2)), from=0, to=20)
```



In this gamma distribution, we see a peak (mode) around $x = 2.5$, and the tail gets thin as $n \to \infty$. Unlike the normal distribution, we can see that the gamma distribution is not symmetric.

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 $\bar{X}_n^{(1)}, \dots, \bar{X}_n^{(r)}$. Plot a histogram of these r values and describe what you see. This is the sampling distribution of $\bar{X}_{(n)}$.

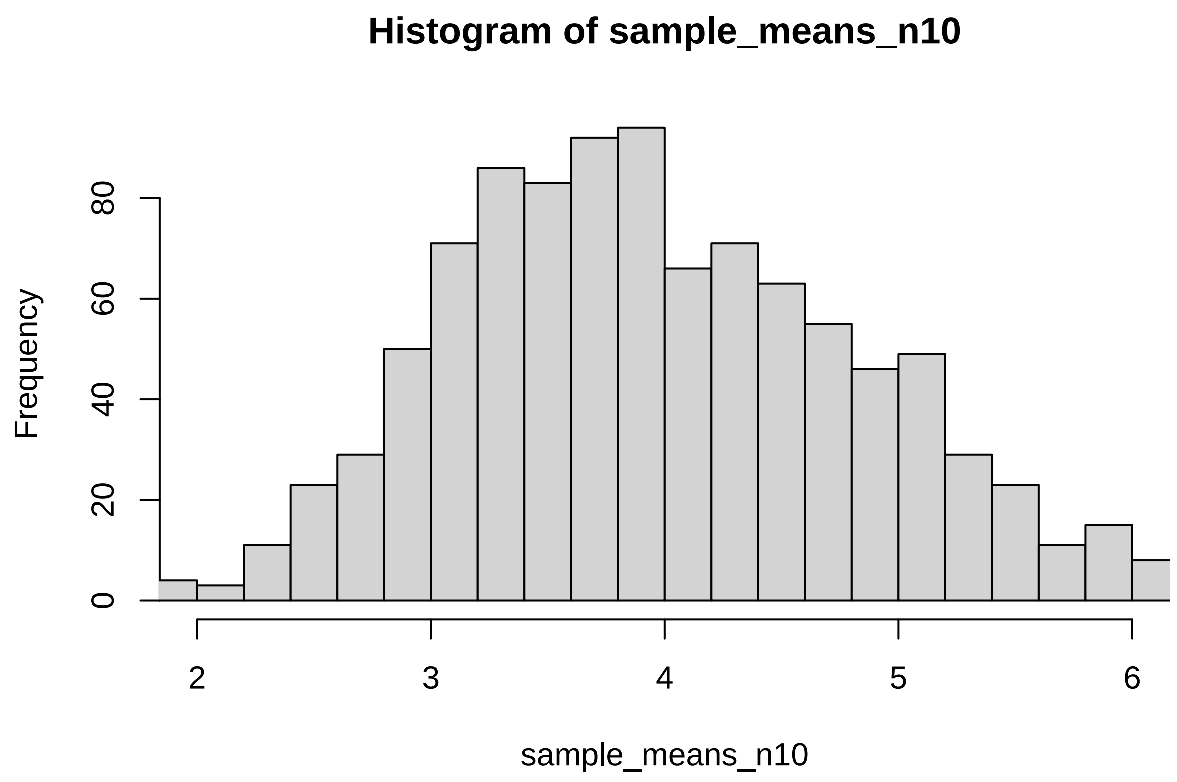
Solution:

```r
# create gamma distribution with n = 10
n = 10
rvals = rgamma(n, shape=2, scale=2)
# calculate first sample average
x1 = mean(rvals)
cat('First sample average is', x1)
```

```
## First sample average is 4.927188
```

```r
# generate 1000 sample means with n = 10 and plot histogram
r = 1000
sample_means_n10 = c(rep(0,r))
for (i in 1:r){
  rvals = rgamma(n, shape=2, scale=2)
  sample_means_n10[i] = mean(rvals)
}
```

```r
hist(sample_means_n10, breaks = 25, xlim = c(2,6))
```

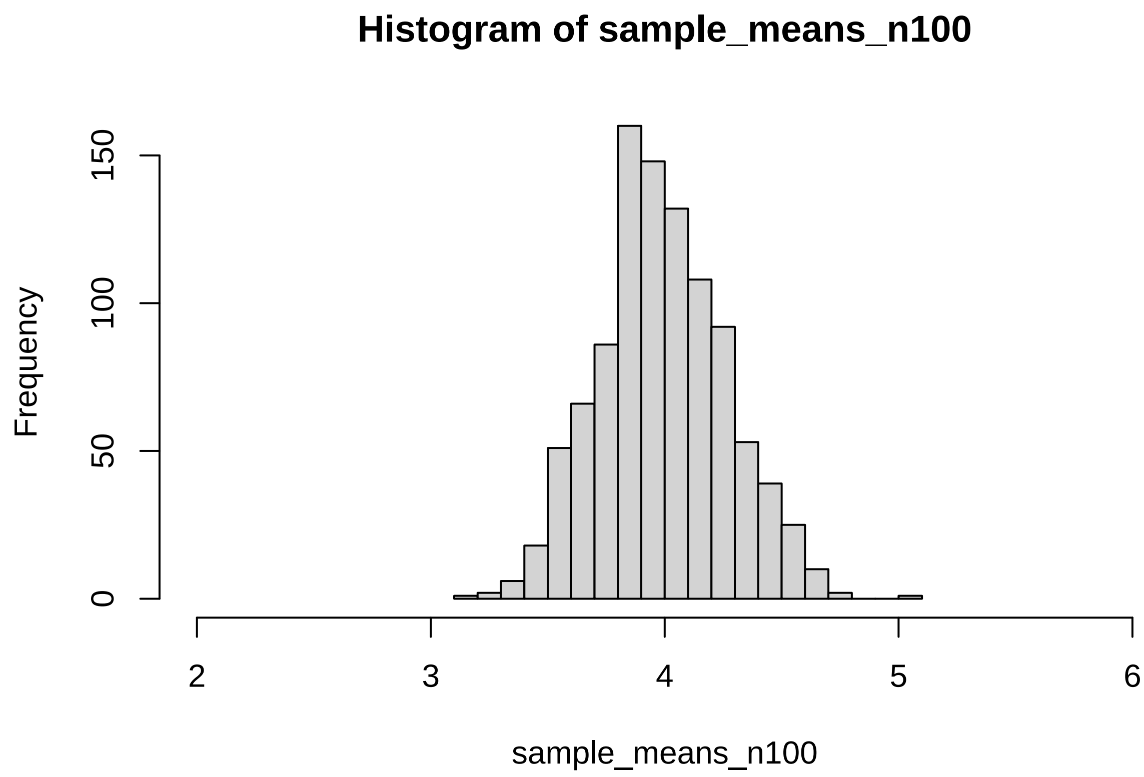


From this histogram, we can see that the distribution of sample means with $n = 10$ roughly resembles a normal distribution. However, it's certainly not perfect; the tails are not very symmetric nor thin.

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:

```r
# generate 1000 sample means with n = 100 and plot histogram
n = 100
r = 1000
sample_means_n100 = c(rep(0,r))
for (i in 1:r){
  rvals = rgamma(n, shape=2, scale=2)
  sample_means_n100[i] = mean(rvals)
}
hist(sample_means_n100, breaks = 25, xlim = c(2,6))
```



From this histogram, we can see that the distribution of sample means with $n = 100$ resembles a normal distribution to a better degree than with $n = 10$. This demonstrates the CLT at work because the increase in sample size led to a better representation of a normal distribution, upholding the notion that the distribution of sample means approaches a normal distribution as n increases.

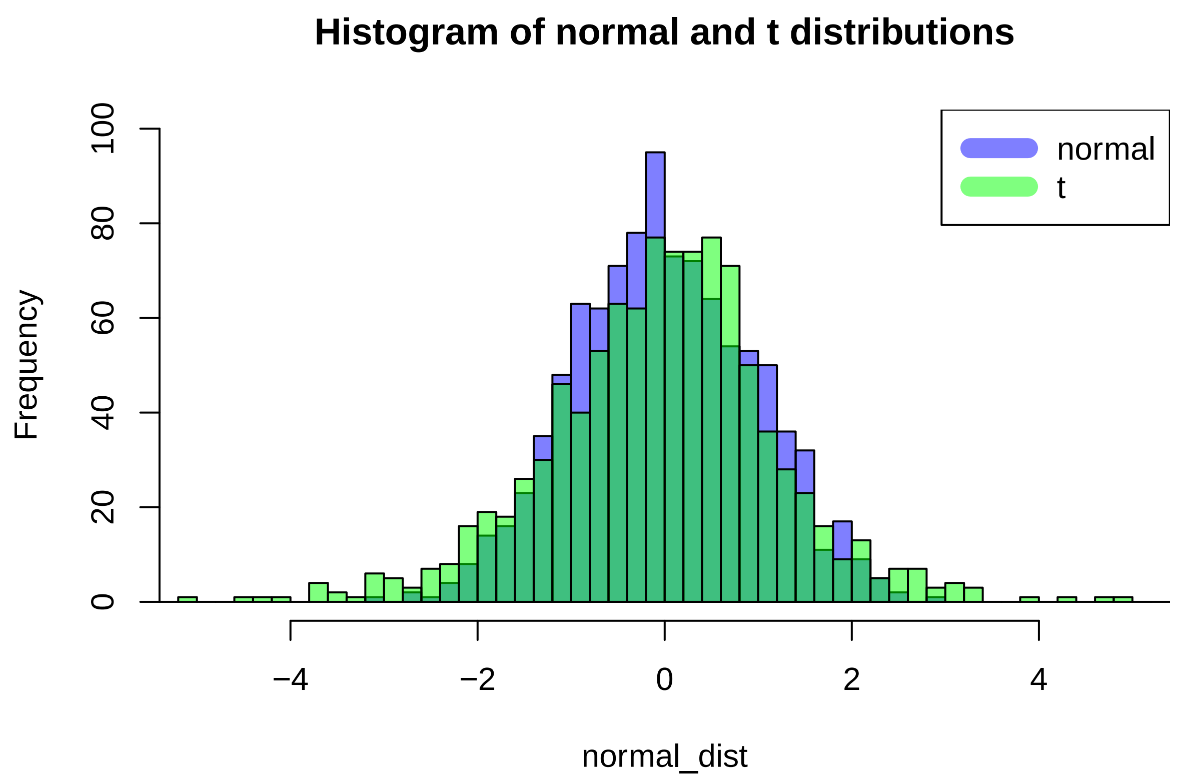
Question 4

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:

```r
# draw from N(0,1) distribution
n = 1000
normal_dist = rnorm(n, mean = 0, sd = 1)
# draw from t-distribution with 5 dof
t_dist = rt(n, df=5)
# plot and overlay histograms
hist(normal_dist, col=scales::alpha("blue",0.5),
     xlim = c(-5,5), ylim = c(0,100), breaks=25,
     main = "Histogram of normal and t distributions")
hist(t_dist, col=scales::alpha("green",0.5), breaks=75, xlim = c(-5,5), add=TRUE)
legend("topright", c("normal", "t"),
       col=c(scales::alpha("blue",0.5),scales::alpha("green",0.5)),
       lwd=10)
```



We can see from the overlayed histograms that the normal distribution has thinner tails than the t-distribution with 5 degrees of freedom, as expected.

Question 5

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

   Solution:

```
# import package and data
library(DataAnalytics)
data(Vanguard)
# subset data to contain only the VFIAX fund
# note that we could also use dcast() to reshape the data and get a row for each fund
vfiax = Vanguard[Vanguard$ticker == 'VFIAX',]
# compute standard error of the return
vfiax_return_std_error = sd(vfiax$mret) / sqrt(length(vfiax$mret))
print(vfiax_return_std_error)
```

```
## [1] 0.003670128
```

```r
# verify with std.error from plotrix package
library(plotrix)
verify_std_error = std.error(vfiax$mret)
print(verify_std_error)
```

```
## [1] 0.003670128
```

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?

   Solution:

```r
# verify near equality of standard error of the mean and mean return
print(mean(vfiax$mret))
```

```
## [1] 0.003959993
```

It is problematic that the mean and standard error of the mean are almost exactly the same because, when computing confidence intervals, this will cause the lower bound to be negative. For the purpose of assessing the performance of this fund, this is problematic because it becomes unclear whether the fund will have a positive return or a negative return.

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:

We know that

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

where n is the size of the sample. To reduce the standard error of the mean to 1/10th the size of the mean return, we thus need to increase $\sqrt{n}$ by a factor of 10 and thus we need to increase n by a factor of 100 (this will reduce the standard error of the mean to 1/10th of the size of the mean return because we know that the standard error of the mean is roughly equal to the mean return, as mentioned in part b). We retrieve the current sample size and multiply by 100 to get our answer:

```r
n = length(vfiax$mret)
n_new = n * 100
print(n_new)
```

```
## [1] 15100
```

So, we need a sample size of roughly 15100 data points to reduce the standard error of the mean to 1/10th the size of the mean return.

Question 6 : Subsetting Observations

We have seen that R has very powerful capabilities to find observations with specific characteristics. The `[<select rows>,<select columns>]` notation is used to subset a data frame. For example, `cars[1:100,]` selects the first 100 observations (rows) of the `cars` data frame.

The most powerful subsetting is done by using what are called logical expressions. A logical expression is an expression that is either TRUE or FALSE. A very simple example would be

```r
var = "Ford"     # 1
var == "Ford"    # 2
```

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

```
var != "Ford"     # 3
```

```
## [1] FALSE
```

In statement #1, we are assigning the value of "Ford" to the R object called `var`. This makes `var` a character vector with only one element.
In statement #2, we are comparing what is stored in `var` to the string "Ford", `==` is the "equals" comparison operator. If the contents of `var` is equal to "Ford", then R will produce the result `TRUE`.
In statement #3, we use the "not equals" comparison operator. #3 will return `FALSE`.

Logical Comparison Operators

Operator	Meaning
`==`	equals
`!=`	not equal
`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to

These ideas can be extended to the vectors.

```
vec=c("BMW","Cadillac","Audi")
vec == "BMW"
```

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

```
vec != "BMW"
```

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

What does this have to do with selecting rows in a data frame (or subsetting our data)? We can use a vector that only contains "TRUE" or "FALSE" to select rows. Let's look at a simple example.

```
library(DataAnalytics)
data(mvehicles)
cars=mvehicles[mvehicles$bodytype != "Truck",]
cars_f4 = head(cars,n=4)
cars_f4[,1:3] # show only the first three columns to save space in the output
```

```
##   make year    model
## 1  BMW 2011 5 Series
## 2  BMW 2011 5 Series
## 3  BMW 2011 5 Series
## 4  BMW 2011 5 Series
```

```
cars_f4[c(FALSE,TRUE,FALSE,TRUE),1:3]
```

```
##   make year    model
## 2  BMW 2011 5 Series
## 4  BMW 2011 5 Series
```

The last statement in the code block above has selected the second and fourth rows. This looks like a much more cumbersome way to do it than just `cars_f4[c(2,4),]`. However, the power comes when you try to select on the values of some of the variables in the data frame which describe which observation is which.

As we saw, if we want to select all of the Fords in the data, we can simply write

```r
fords = cars[cars$make == "Ford",]
```

We now know how this works. `cars$make == "Ford"` creates a vector with TRUE values marking any observation where `make == "Ford"` and FALSE when not.

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.

   Solution:

```r
# import package and data
library(DataAnalytics)
data(mvehicles)
# initialize data frame and print logical vector
cars = mvehicles[mvehicles$bodytype != "Truck",]
ford_logicals = cars$make == "Ford"
ford_logicals[1:50]
```

```
##  [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.

   Solution:

```r
# create row selection logical vector
selection = cars$make == "Ford"
# select observations
fords = cars[selection,]
# confirm selections
fords[1:5, 1:3]
```

```
##    make year model
## 6  Ford 2011  Edge
## 7  Ford 2011  Edge
## 8  Ford 2011  Edge
## 9  Ford 2011  Edge
## 10 Ford 2011  Edge
```

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

   Solution:

```r
# count number of kias
num_kias = nrow(cars[cars$make == "Kia",])
print(num_kias)
```

```
## [1] 43
```

So, there are 43 Kias in the dataset.

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

Solution:

```r
# count number of cars that cost over 100k
num_over_100k = nrow(cars[cars$emv > 100000,])
print(num_over_100k)
```

```
## [1] 37
```

So, there are 37 cars with a price over $100,000.

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?'

   Solution:

```r
# create selections
selection_eur = cars$origin == "Europe"
selection_over_75k = cars$emv > 75000
# subset data
eur_over_75k = cars[selection_eur & selection_over_75k,]
# calculate mean
avg_sales_eur_over_75k = mean(eur_over_75k$sales)
print(avg_sales_eur_over_75k)
```

```
## [1] 626.6957
```

So, the average sales are roughly 627.

In many data sets, there are long text fields which 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.

Q 6, part C

1. How many four door vehicles are in cars?

   Solution:

```r
# select four door cars, count
four_drs = cars[grepl("4dr",cars$style, ignore.case=TRUE),]
```

```r
num_four_drs = nrow(four_drs)
print(num_four_drs)
```

```
## [1] 1105
```

So, there are 1105 four door cars.

2. How many four door sedans are in cars?

   Solution:

```r
# select sedans from four door selection
four_dr_sed = four_drs[grepl("sedan",cars$style,ignore.case=TRUE),]
num_four_dr_sed = nrow(four_dr_sed)
print(num_four_dr_sed)
```

```
## [1] 433
```

So, there are 433 four door sedans.

Question 7 : Sales and Price relationships

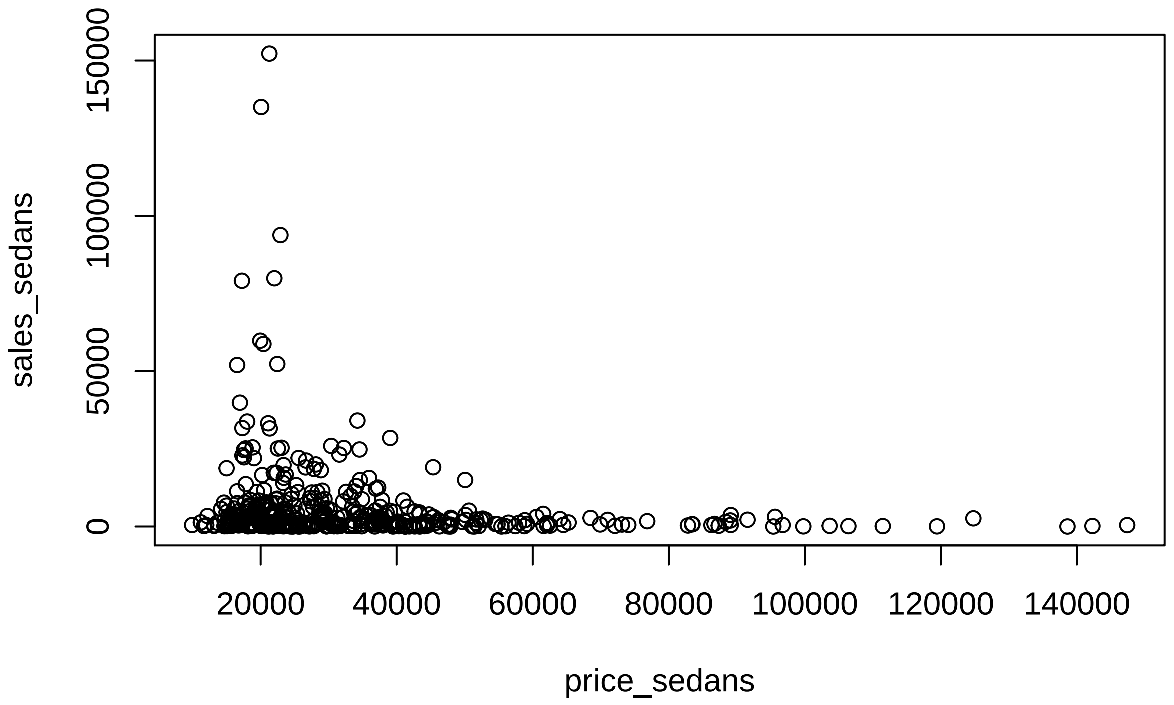
In this question, use `cars` only.

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?

Solution:

```r
sedans = cars[cars$bodytype == "Sedan",]
price_sedans = sedans$emv
sales_sedans = sedans$sales
plot(price_sedans, sales_sedans)
```



The problem with displaying the data in this manner is that there are some data points with very high price and some with very high sales values. This offsets the scale of the data, making it very hard to interpret the data points at lower prices and sales values.

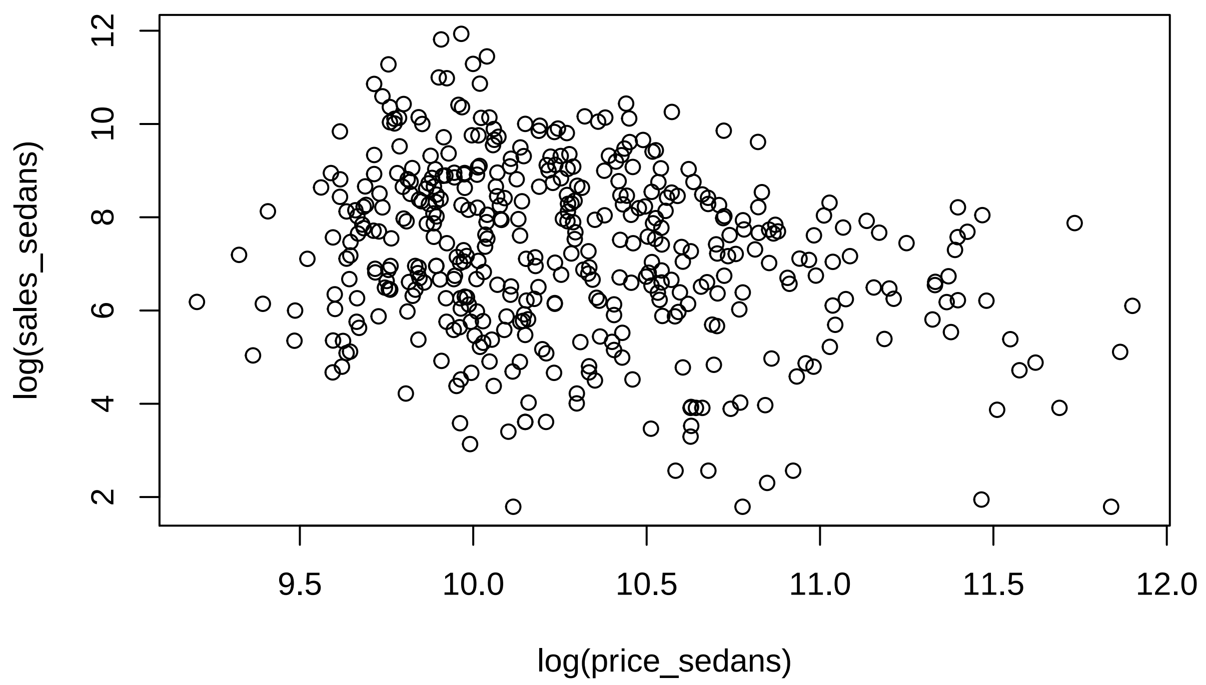
Q7, part B

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?

Solution:

The log transformation:

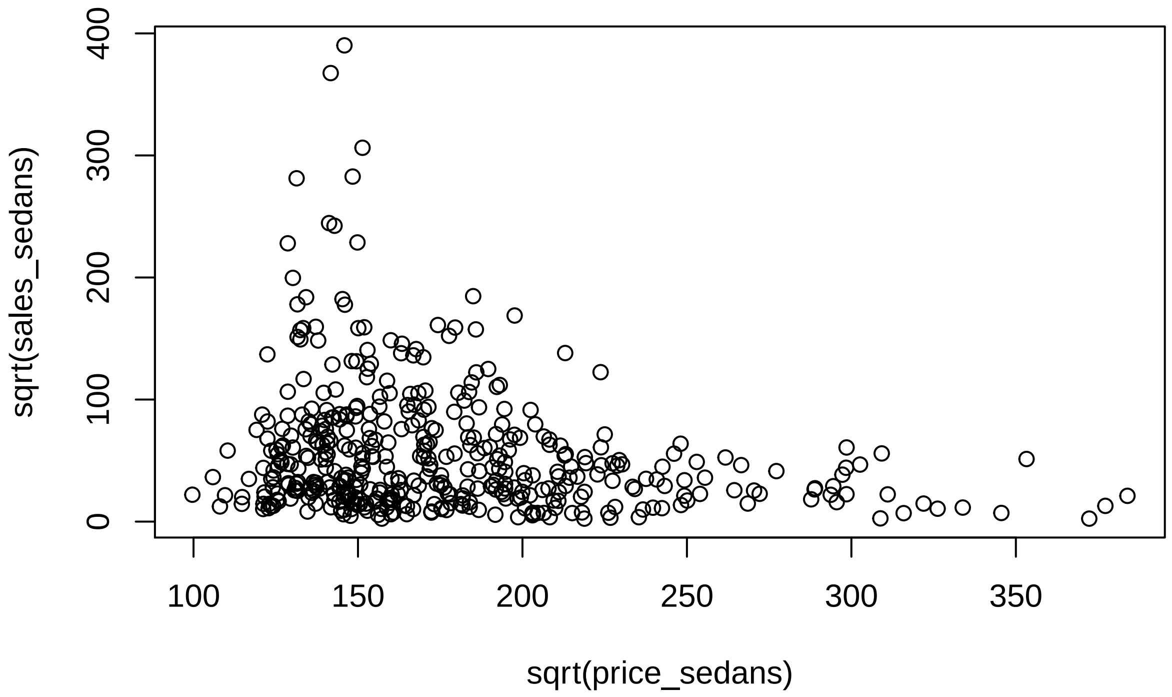
```r
plot(log(price_sedans), log(sales_sedans))
```



This has improved the visualization of the data in the sense that most of the data points are now on a more similar scale, allowing them to more clearly distinguished. Some disadvantages of this transformation are that values less than 1 will become negative (irrelevant here since no cars cost less than a dollar) and a negative value for something like a price makes no sense. Additionally, this transformation can skew or obscure a relationship between variables (e.g. uniform data becomes left-skewed).

Now, the sqrt transformation:

```r
plot(sqrt(price_sedans), sqrt(sales_sedans))
```



This transformation is also useful, but it doesn't reconcile the volatile scale of the data as well as the log transformation. Because of this, there are still some data points with very high sqrt(price) and sqrt(sales) values, making the plot still somewhat difficult to interpret visually.

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?

Solution: The log-transformed plot supports the idea that sales decrease with price. This trend is less obvious on the sqrt-transformed plot.

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`.

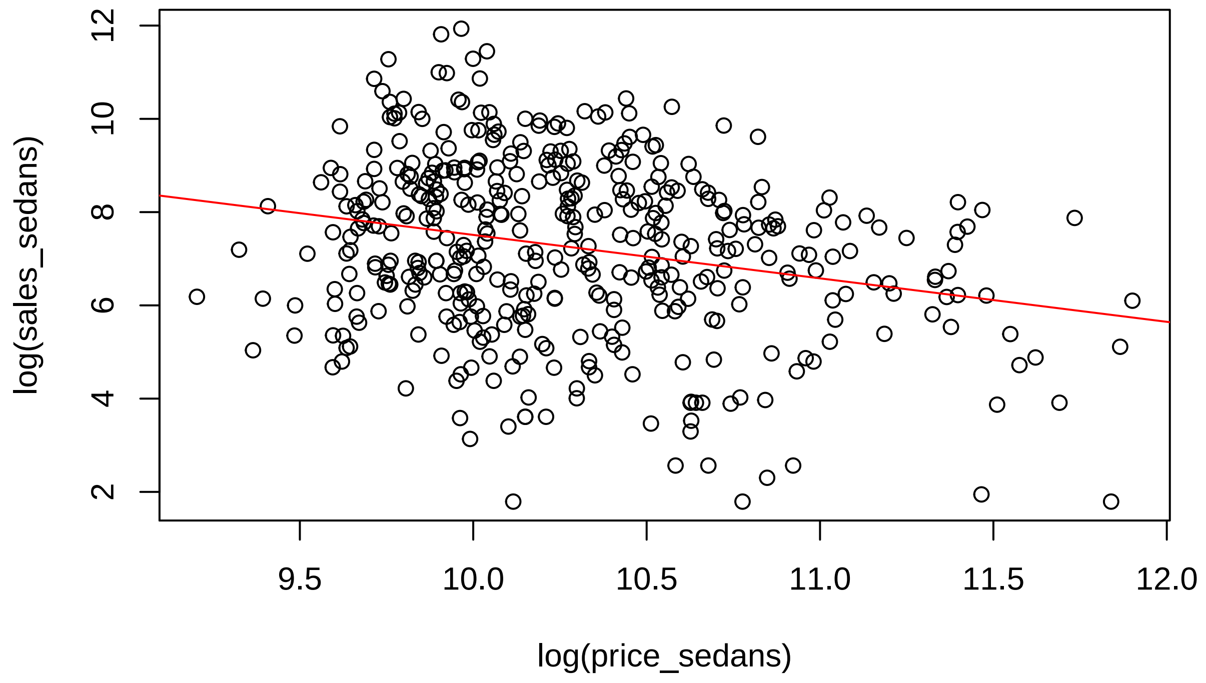
Solution:

```r
# run regression and output summary
outlm = lm(log(sales_sedans)~log(price_sedans))
summary(outlm)
```

```
##
## Call:
## lm(formula = log(sales_sedans) ~ log(price_sedans))
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6121 -1.1720  0.2205  1.3554  4.3897
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         16.8321     1.8196   9.250  < 2e-16 ***
## log(price_sedans)  -0.9321     0.1768  -5.271 2.14e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.822 on 431 degrees of freedom
## Multiple R-squared:  0.06057,    Adjusted R-squared:  0.05839
## F-statistic: 27.79 on 1 and 431 DF,  p-value: 2.144e-07
```

```
# plot regression line on top of log plot
plot(log(price_sedans), log(sales_sedans))
abline(outlm$coef, col = "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.

Solution:

```r
pred = predict(outlm, new=data.frame(price_sedans=45000))
pred_transf = exp(pred)
print(pred_transf)
```

```
##        1
## 939.5983
```

So, we predict sales of roughly 940 for a car with a price of $45,000.