CS112 Assignment 2, Fall 2020

Nikita Kotsehub

10/20/2020

Note: *This is an RMarkdown document. Did you know you can open this document in RStudio, edit it by adding your answers and code, and then knit it to a pdf? Then you can submit both the .rmd file (the edited file) and the pdf file as a zip file on Forum. This method is actually preferred. To learn more about RMarkdown, watch the videos from session 1 and session 2 of the CS112B optional class. This is also a cheat sheet for using Rmarkdown. If you have questions about RMarkdown, please post them on Perusall.*

Note: *If you are not comfortable with RMarkdown, you can use any text editor (google doc or word) and type your answers and then save everything as a pdf and submit both the pdf file AND the link to your code (on github or jupyter) on Forum.*

Note: *Try knitting this document in your RStudio. You should be able to get a pdf file. At any step, you can try knitting the document and recreate a pdf. If you get error, you might have incomplete code.*

Note: *If you are submitting your assignment as an RMarkdown file, make sure you add your name to the top of this document.*

QUESTION 1

STEP 1 Create a set of 1000 outcome observations using a data-generating process (DGP) that incorporates a systematic component and a stochastic component (of your choice)

```
# YOUR CODE HERE

temperature <- rtruncnorm(n=1000, a=34, b=45, mean=36.6, sd=1.4)
sex <- rbinom(1000, 1, 0.5)
age <- as.integer(rtruncnorm(n=1000, a=18, b=80, mean=38, sd=25))
noise <- rnorm(1000)

dd0 <- data.frame(cbind(temperature, sex, age, noise)) %>%
  mutate(chebs = 3*temperature + 1.5*sex + 4*age + 8*noise)

head(dd0)
```

```
##   temperature sex age       noise    chebs
## 1    36.62050   1  38  0.76440734 269.4767
## 2    37.82260   1  74 -0.14674361 409.7939
## 3    35.16088   1  70 -1.87357930 371.9940
## 4    37.55993   1  60 -0.95080461 346.5734
## 5    37.22921   0  47 -0.63006181 294.6471
## 6    37.16982   1  29  0.05923826 229.4834
```

STEP 2 Tell a 2-3 sentence story about the data generating process you coded up above. What is it about and what each component mean?

The independent variable is chebs - the number of special blood cells in one's blood (the variable name is made up). Each observation contains measurement of chebs, temperature in Celsius, sex (male = 1, female = 0), and age in the range of [18, 80]. The stochastic element is a random number sampled from a normal distribution with mean=0 and SD=1, depicting natural variation in the number of chebs.

STEP 3 Using an incorrect model of the systematic component (of your choice), and using the simulation-based approach covered in class (the `arm` library, etc.), generate 1 prediction for every observation and calculate and report RMSE. Make sure you write out your model and report your RMSE.

Each prediction should be conditional on the predictor values of the corresponding observation. E.g., when predicting for observation #1, use the predictor values of observation #1.

```
lm1 <- lm(chebs ~ temperature + age, data=dd0)
sim_results <- sim(lm1, n.sims=100)

# get aggregated coefficients from 100 models
intr <- mean(coef(sim_results)[, 1])
tmp <- mean(coef(sim_results)[, 2])
age1 <- mean(coef(sim_results)[, 3])

# create predictions for every observation
prets <- c()
for (i in 1:nrow(dd0)){
  prets[i] <- sum(c(intr, tmp, age1)*c(1, dd0$temperature[i], dd0$age[i]))
}

# function for r^2
my_r <- function(preds){
  prd_mean = mean(preds)

  fitted_diff <- c()
  for (i in 1:length(preds)){
    fitted_diff[i] = (preds[i] - prd_mean)**2
  }
  sum_fit <- sum(fitted_diff)

  act_values <- c()
  for (i in 1:length(preds)){
    act_values[i] = (dd0$chebs[i] - prd_mean)**2
  }
  sum_act <- sum(act_values)

  sum_fit/sum_act
}


my_rmse <- function(preds, actual){
  sqrt(mean((preds-actual)**2))
}



sprintf("The RMSE of the incorrect model is %f, R^2 is %f", my_rmse(prets, dd0$chebs), my_r(prets))
```

```
## [1] "The RMSE of the incorrect model is 7.814628, R^2 is 0.987087"
```

STEP 4 Using the correct model (correct systematic and stochastic components), and using the simulation-based approach covered in class (the `arm` library, etc.), generate 1 prediction for every observation and calculate & report your RMSE. Once again, write out your model and report your RMSE.

Each prediction should be conditional on the predictor values of the corresponding observation. E.g., when predicting for observation #1, use the predictor values of observation #1.

```
# YOUR CODE HERE
lm2 <- lm(chebs ~ temperature + sex + age + noise, data=dd0)

sim_results2 <- sim(lm2, n.sims=100)

# get aggregated coefficients
intr2 <- mean(coef(sim_results2)[, 1])
tmp2 <- mean(coef(sim_results2)[, 2])
sex2 <- mean(coef(sim_results2)[, 3])
age2 <- mean(coef(sim_results2)[, 4])
noise2 <- mean(coef(sim_results2)[, 5])

# predict the value for each observation
prets2 <- c()
for (i in 1:nrow(dd0)){
  prets2[i] <- sum(c(intr2, tmp2, sex2, age2, noise2)*c(1, dd0$temperature[i], dd0$sex[i], dd0$age[i], c
}

my_rmse <- function(preds, actual){
  sqrt(mean((preds-actual)**2))
}


sprintf("The RMSE of the correct model is %f, R^2 is %f", my_rmse(prets2, dd0$chebs), my_r(prets2))
```

```
## [1] "The RMSE of the correct model is 0.000000, R^2 is 1.000000"
```

STEP 5 Which RMSE is larger: The one from the correct model or the one from the incorrect model? Why?

The incorrect model has a larger RMSE and a smaller R^2. This makes sense because the incorrect model does not use all the variables and does not account for the noise, resulting in a worse fit. Worse fit results in larger residuals. Therefore, we get a larger RMSE. Meanwhile, the correct model has all the information and even accounts for the inherent variance by making use of the noise. This results in an almost perfect fit, leading to a small RMSE (0.00) and perfect R^2 (1.00).

QUESTION 2

Imagine that you want to create a data viz that illustrates the sensitivity of regression to outlier data points. So, you want to create two figures:

One figure that shows a regression line fit to a 2-dimensional (x and y) scatterplot, such that the regression line clearly has a positive slope.

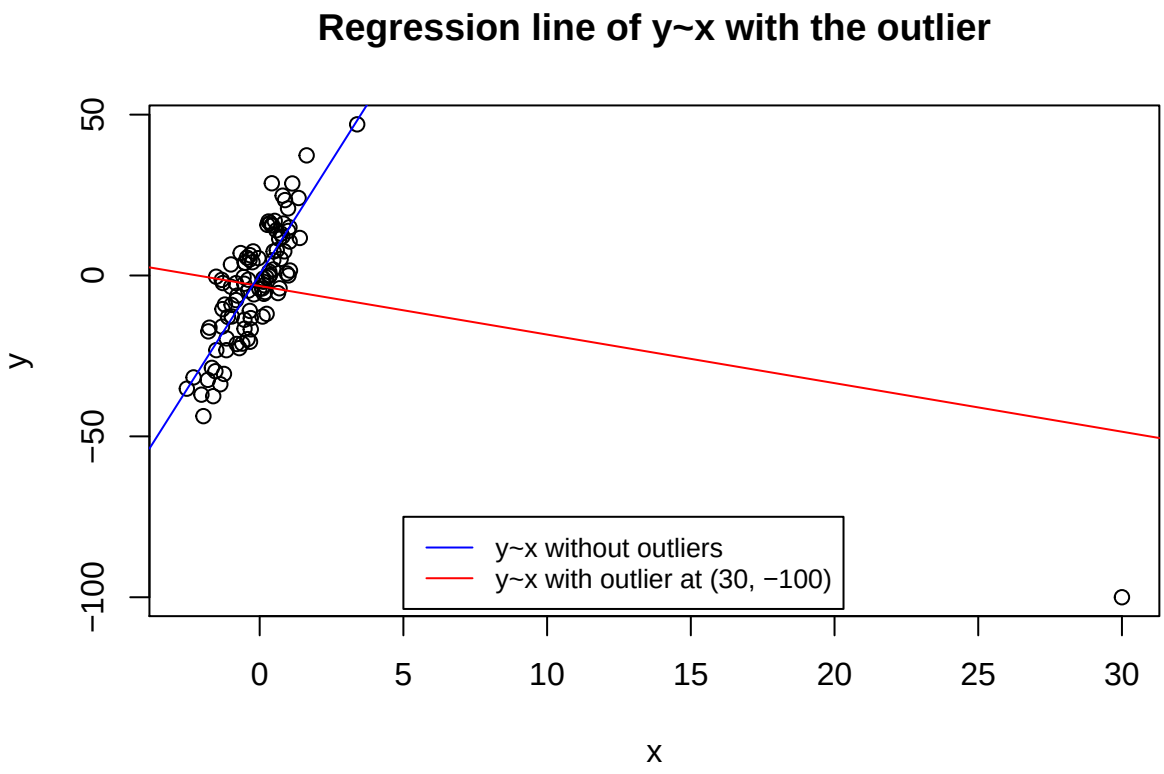
```
# YOUR CODE HERE
x <- rnorm(100)
y <- 15*x + 3 + 10*rnorm(50)
df001 <- data.frame(cbind(x, y))
```

```
plot(df001, main="Regression line of y~x without outliers") + abline(lm(y~x, data=df001), col="blue")
```


```
## integer(0)
```

And, another figure that shows a regression line with a negative slope fit to a scatter plot of the same data **plus one additional outlier data point**. This one data point is what changes the sign of the regression line's slope from positive to negative.

```
# YOUR CODE HERE
df002 <- rbind(df001, c(30, -100))

{
plot(df002, main="Regression line of y~x with the outlier") + abline(lm(y~x, data=df002), col="red") +
  abline(lm(y~x, data=df001), col="blue")
legend(5, -75, legend=c("y~x without outliers", "y~x with outlier at (30, -100)"),
       col=c("blue", "red"), lty=1:1, cex=0.8)
}
```



Be sure to label the axes and the title the figures appropriately. Include a brief paragraph that explains the number of observations and how you created the data set and the outlier.

We created the independent variable x by running the rnorm function which generated 100 observations from a normal distribution with mean 0 and standard deviation 1. The dependent variable y was created as a function of x with some random noise, which represents the inherent variance in the data. To create the outlier, we appended additional point to our dataframe with drastically different values - (30, -100).

QUESTION 3

STEP 1 Using the `laLonde` data set, run a linear regression that models `re78` as a function of `age`, `education`, `re74`, `re75`, `hisp`, and `black`. Note that the `lalonde` data set comes with the package `Matching`.

```
# YOUR CODE HERE
data(lalonde)
lm31 <- lm(re78 ~ age+educ+re74+re75+hisp+black, data=lalonde)
```

STEP 2 Report coefficients and R-squared.

```
list("Coefficients" = lm31$coef, "R-squared" = summary(lm31)$r.squared)
```

```
## $Coefficients
##   (Intercept)           age          educ          re74          re75
##  1.126492e+03  5.928155e+01  4.293074e+02  7.461872e-02  6.676314e-02
##          hisp         black
## -2.125053e+02 -2.323282e+03
##
## $`R-squared`
```

```
## [1] 0.03942359
```

Then calculate R-squared by hand and confirm / report that you get the same or nearly the same answer as the summary (`lm`) command.

Write out hand calculations here.

```
# generate the pridictions
preds <- predict(lm31, newdata=lalonde)


my_r <- function(preds){
  prd_mean = mean(preds)

  fitted_diff <- c()
  for (i in 1:length(preds)){
    fitted_diff[i] = (preds[i] - prd_mean)**2
  }
  sum_fit <- sum(fitted_diff)

  act_values <- c()
  for (i in 1:length(preds)){
    act_values[i] = (lalonde$re78[i] - prd_mean)**2
  }
  sum_act <- sum(act_values)

  sum_fit/sum_act
}


sprintf("Hand-calculated R-squared is equal to %f. R-squared from the model's summary is equal to %f", r
```

```
## [1] "Hand-calculated R-squared is equal to 0.039424. R-squared from the model's summary is equal to
```

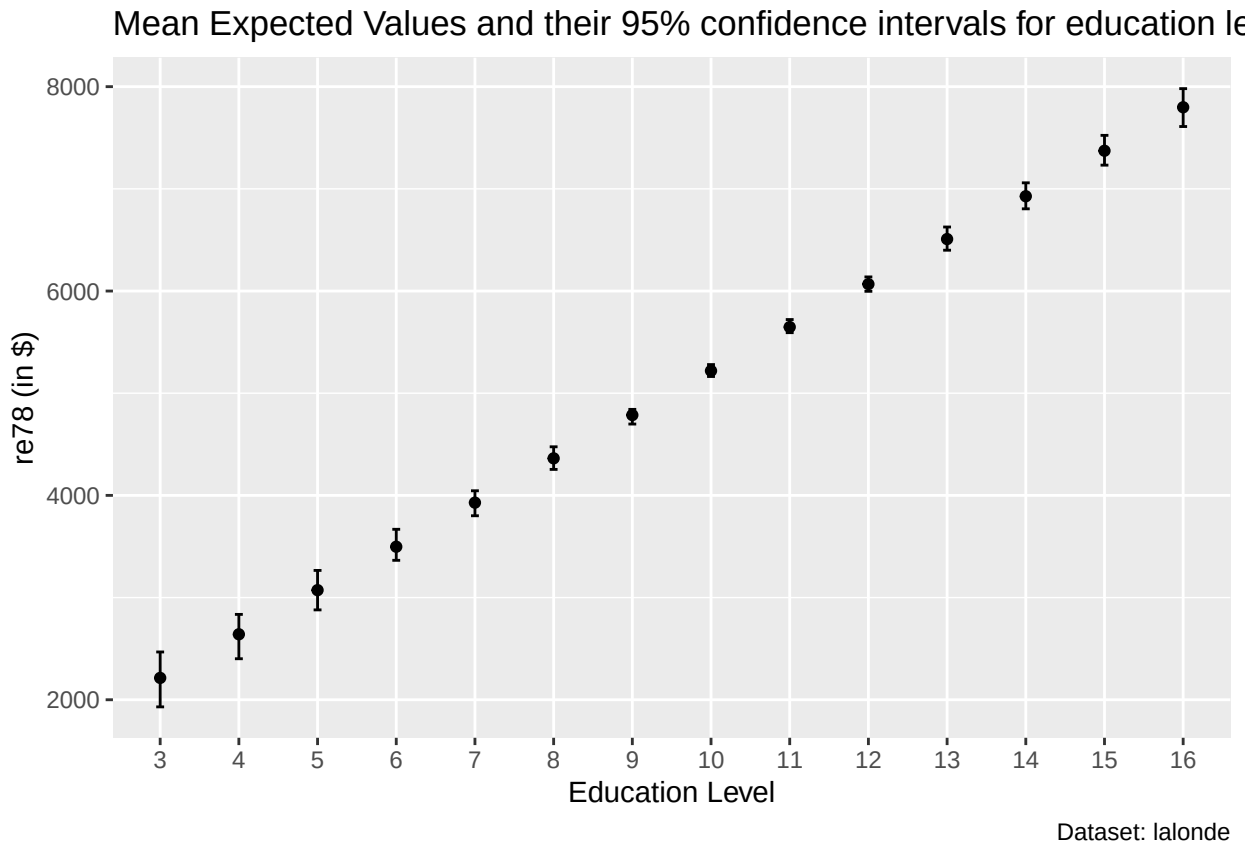
STEP 3 Then, setting all the predictors at their means EXCEPT `education`, create a data visualization that shows the 95% confidence interval of the expected values of `re78` as `education` varies from 3 to 16. Be sure to include axes labels and figure titles.

Here, for each education level k, we want to obtain 100 Expected Values j. For each j, we create a new set of simulated models i, we use each model to create a prediction, and average the predictions to get the Expected Value. Then, we store that expected value in a storage.

After we obtain j expected values for each education level, we find the mean and CI of each level and plot them.

```
# YOUR CODE HERE

exp_vals <- c()

for (k in 3:16){
  temp_exp_vals <- c()
  for (j in 1:100)
  {
    sim31 <- sim(lm31, n=100)
    vsc <- c()
    for (i in 1:100)
      {
```

```
      vsc[i] <- sum(coef(sim31)[i, ]*c(1, mean(lalonde$age), k, mean(lalonde$re74), mean(lalonde$re75),
    }
    temp_exp_vals[j] <- mean(vsc)
  }
  exp_vals[[k]] <- temp_exp_vals
}


# obtain the 2.5% quantile
quants_exp_low <- c()
for (i in 1:14) {
  quants_exp_low[i] = quantile(exp_vals[[i+2]], probs = c(0.025))
}

# 97.5 quantile
quants_exp_high <- c()
for (i in 1:14) {
  quants_exp_high[i] = quantile(exp_vals[[i+2]], probs = c(0.975))
}

# mean of the expected values
mean_exp <- c()
for (i in 1:14){
  mean_exp[i] <- mean(exp_vals[[i+2]])
}

df32 <- data.frame(cbind(c(3:16), mean_exp, quants_exp_low, quants_exp_high))

# plot
ggplot(data=df32, aes(x=as.factor(V1), y=mean_exp)) +
  geom_point() +
  geom_errorbar(aes(ymin=quants_exp_low, ymax=quants_exp_high), width=.1) +
  labs(title="Mean Expected Values and their 95% confidence intervals for education levels", x="Educati
```

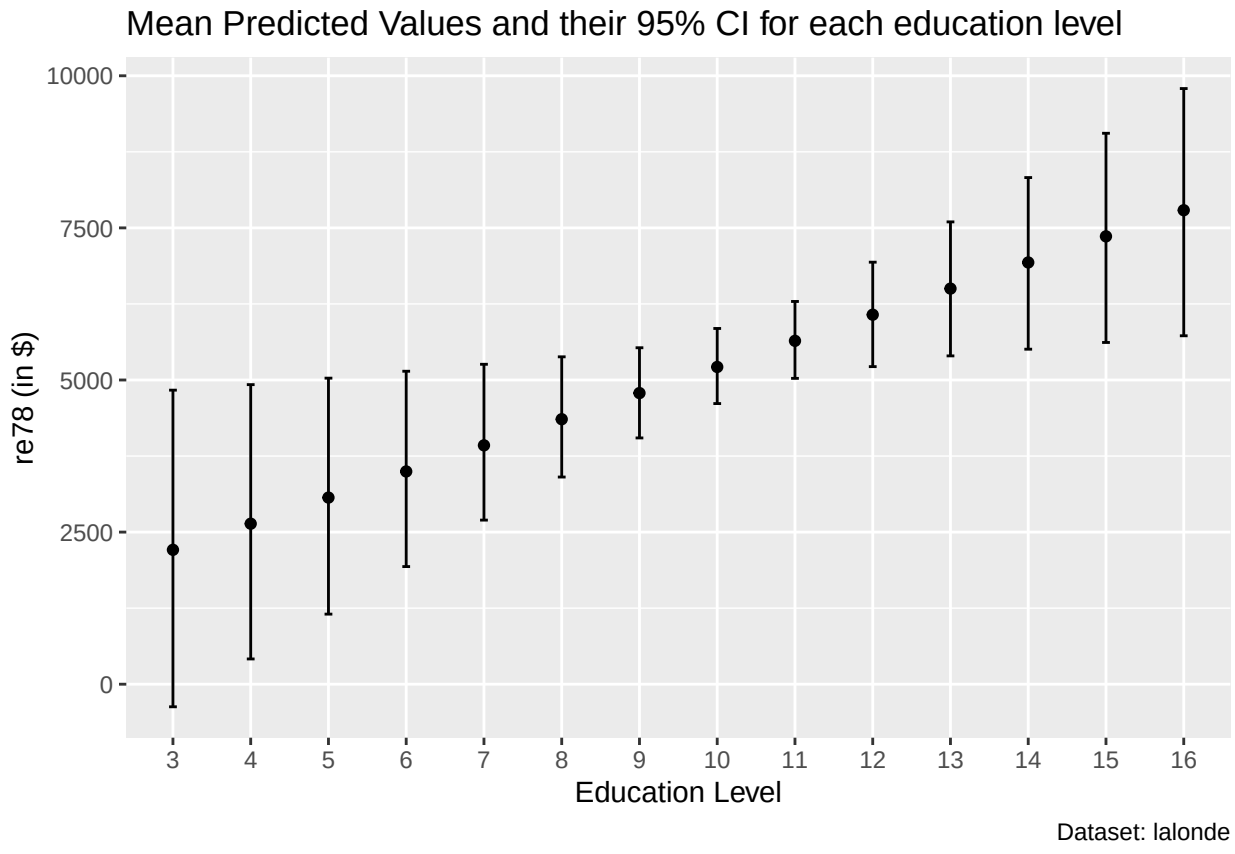


STEP 4 Then, do the same thing, but this time for the predicted values of `re78`. Be sure to include axes labels and figure titles.

Here, for each education level k, we want to obtain 100 predicted values i. We use 1000 different models to obtain 1000 different predictions for each level. After we obtain i predicted valeus for each education level, we find the mean and CI of each level and plot them.

```
# YOUR CODE HERE

vsfin <- list()

sim31 <- sim(lm31, n=1000)

for (k in 3:16){
  vsc <- c()
  for (i in 1:1000){
    vsc[i] <- sum(coef(sim31)[i, ]*c(1, mean(lalonde$age), k, mean(lalonde$re74), mean(lalonde$re75), m
  }
  vsfin[[k]] <- vsc
}


quants <- c()
for (i in 1:14) {
  quants[i] = quantile(vsfin[[i+2]], probs = c(0.025))
}
```

```r
quants2 <- c()
for (i in 1:14) {
  quants2[i] = quantile(vsfin[[i+2]], probs = c(0.975))
}

mean_vsfin <- c()
for (i in 1:14){
  mean_vsfin[i] <- mean(vsfin[[i+2]])
}


df01 <- data.frame(cbind(c(3:16), mean_vsfin, quants, quants2))


ggplot(data=df01, aes(x=as.factor(V1), y=mean_vsfin)) +
  geom_point() +
  geom_errorbar(aes(ymin=quants, ymax=quants2), width=.1) +
  labs(title="Mean Predicted Values and their 95% CI for each education level", x="Education Level", y=
```



STEP 5 Lastly, write a short paragraph with your reflections on this exercise (specifically, the length of intervals for given expected vs. predicted values) and the results you obtained.

The 95% Confidence Intervals (CI) for expected values are significantly narrower than for predicted values. This makes sense because each expected value is a mean of all the predicted values. With a large sample size of predicted values, the expected values would be within a narrow range and would not vary much. We

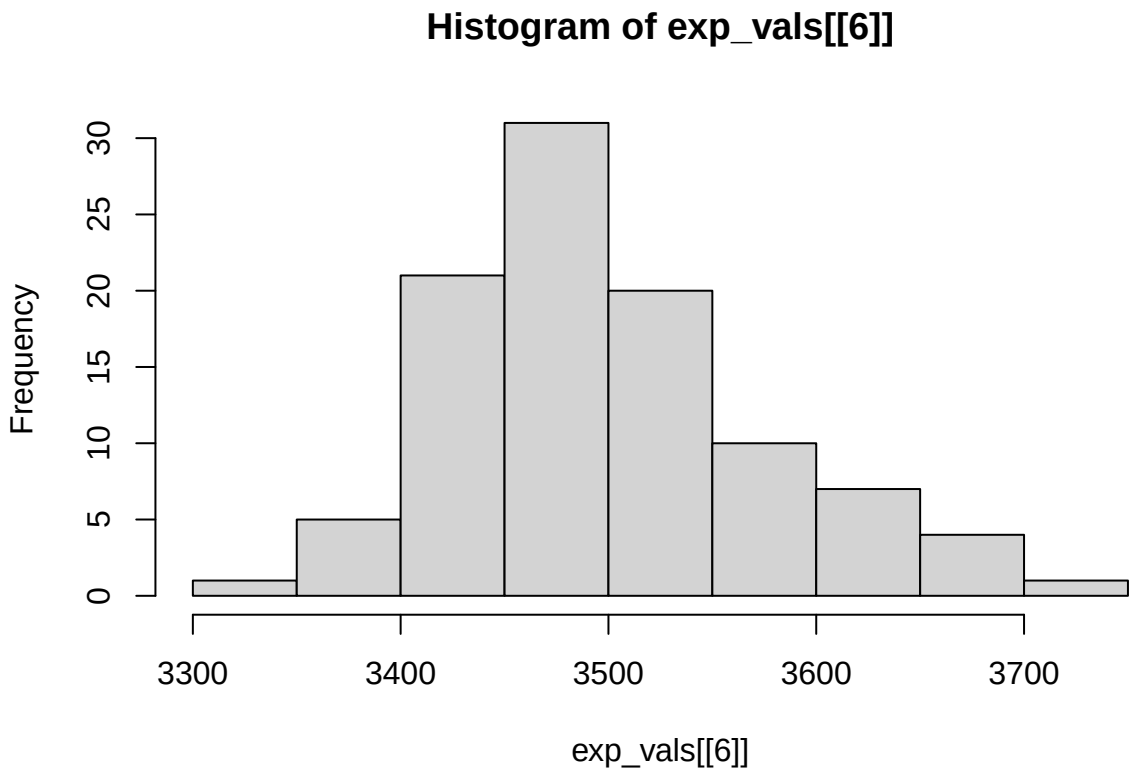
can confirm this by plotting the distribution of expected and predicted values (we will use education level 6). From the first plot, we can see that the range of values for the expected values of education level 6 is approximately from 3300 to 3700. The second histogram tells us that the range of predicted values for education level 6 is from 1000 to 5000+. If we look at the standard deviations, we can also see that SD for Expected values is 10 times smaller, so we would expect smaller variation in Expected values, and thus narrower confidence intervals.

From a decision-making standpoint, we can see a trend: people who spent more years in education tend to have higher sallaries, on average (in 1978).

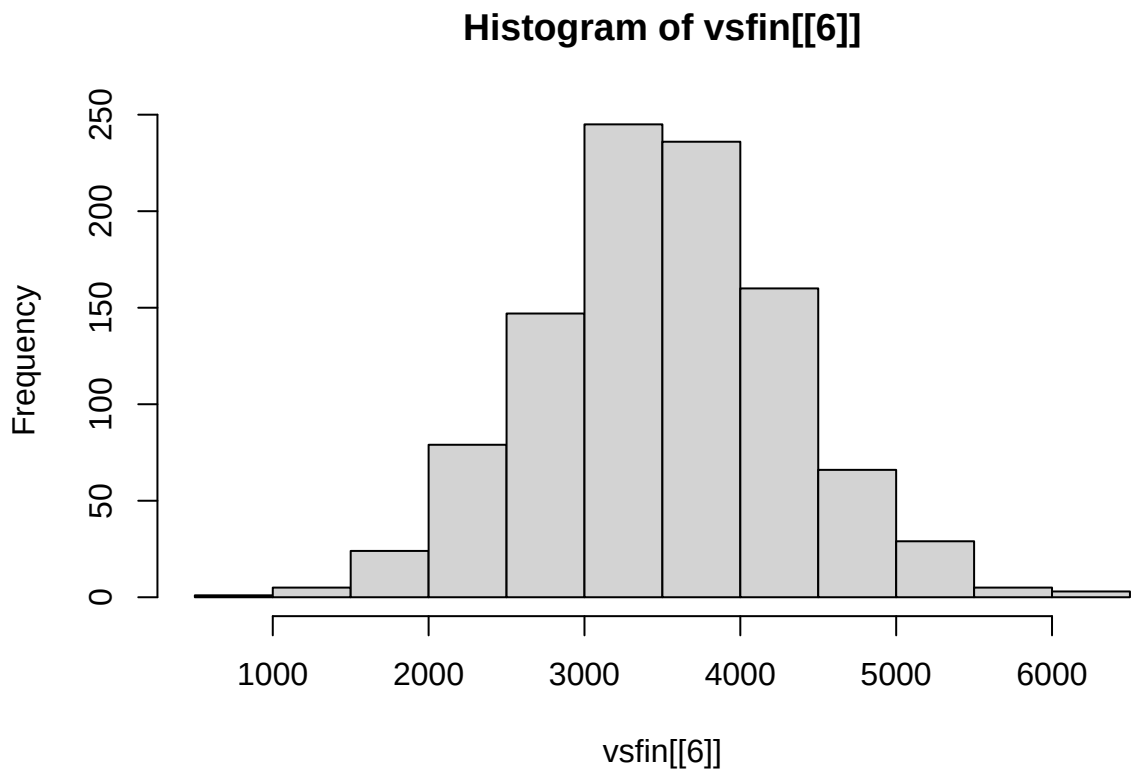
```
sprintf("SD of Expected Values is %f. SD of Predicted valeus is %f", sd(exp_vals[[6]]), sd(vsfin[[6]]))
```

```
## [1] "SD of Expected Values is 75.686537. SD of Predicted valeus is 810.157123"
```

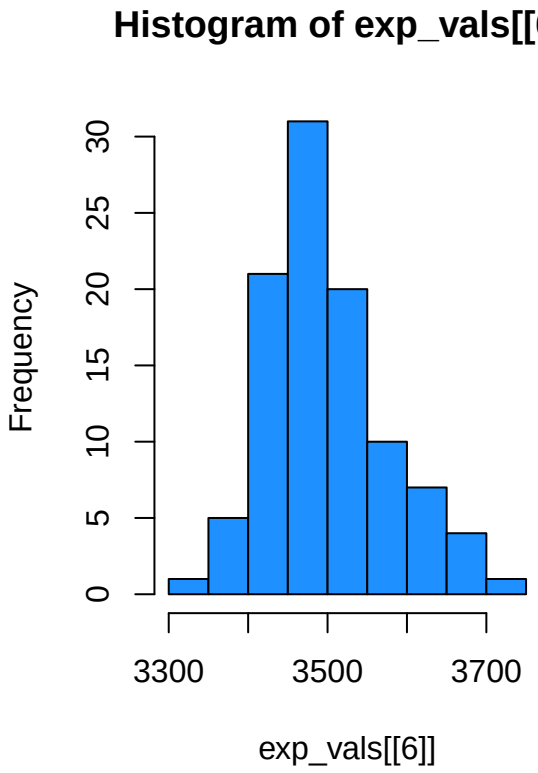
```
hgA = hist(exp_vals[[6]])
```

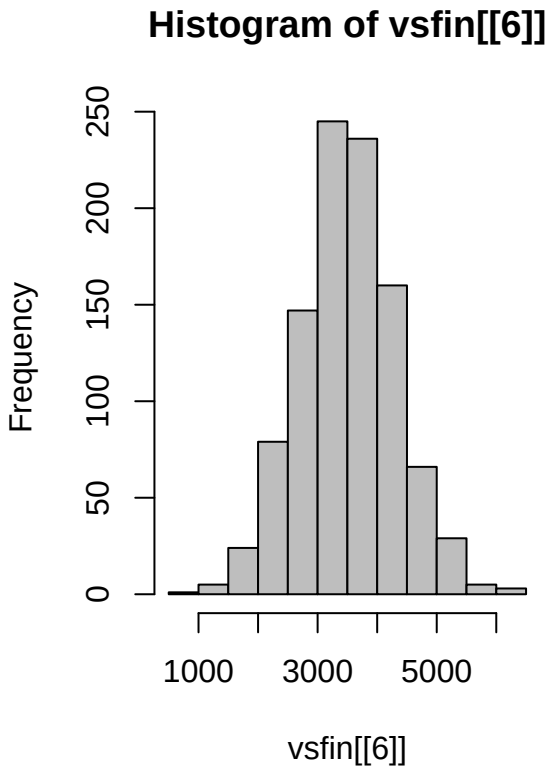



```
hgB = hist(vsfin[[6]])
```




```
{
par(mfrow = c(1, 2))
plot(hgA, col='dodgerblue1', freq=T)
plot(hgB, col = 'gray', freq=T)
}
```





QUESTION 4

STEP 1 Using the `lalonde` data set, run a logistic regression, modeling treatment status as a function of `age`, `education`, `hisp`, `re74` and `re75`. Report and interpret the regression coefficient and 95% confidence intervals for `age` and `education`.

```
# YOUR CODE HERE

lg41 <- glm(treat~age + educ + hisp + re74 + re75, data=lalonde, family=binomial)
summary(lg41)
```

```
##
## Call:
## glm(formula = treat ~ age + educ + hisp + re74 + re75, family = binomial,
##     data = lalonde)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2970  -1.0519  -0.9521   1.2830   1.6924
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.320e+00  6.750e-01  -1.956   0.0505 .
## age          1.165e-02  1.370e-02   0.850   0.3953
## educ         6.922e-02  5.545e-02   1.248   0.2119
## hisp        -6.024e-01  3.777e-01  -1.595   0.1108
## re74        -2.278e-05  2.511e-05  -0.907   0.3643
```

```
## re75         5.221e-05  4.180e-05   1.249   0.2116
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 604.20  on 444  degrees of freedom
## Residual deviance: 596.82  on 439  degrees of freedom
## AIC: 608.82
##
## Number of Fisher Scoring iterations: 4
```

```
# age
ag41 <- c('2.5%' = lg41$coefficients[2] - 1.96*0.0137, '97.5%' = lg41$coefficients[2] + 1.96*0.0137)

# educ
educ41 <- c('2.5%' = lg41$coefficients[3] - 1.96*0.05545, '97.5%'= lg41$coefficients[3] + 1.96*0.05545)

# output
list("CI for Age" = ag41,"CI for Education" = educ41)
```

```
## $`CI for Age`
##    2.5%.age   97.5%.age
## -0.01520362  0.03850038
##
## $`CI for Education`
##   2.5%.educ  97.5%.educ
## -0.03945994  0.17790406
```

Report and interpret regression coefficient and 95% confidence intervals for `age` and `education` here.

The 95% CI for age is -0.015 to 0.0385. This is not good because 0 is included in the confidence interval, meaning that age might have also no effect at all on the outcome. The same goes for Education, whose 95% is [-0.0394, 0.1779] and also includes 0. However, we ran only one model, and to conclude whether variables age and education are significant, we would need to run a simulation on 1000 bootstrapped samples, giving us more accurate CIs. We will do so in the next step.

STEP 2 Use a simple bootstrap to estimate (and report) bootstrapped confidence intervals for `age` and `education` given the logistic regression above. Code the bootstrap algorithm yourself.

Out bootstrapping algorithm works as follows: We want to have 1000 bootstrapped samples. For each sample, we bootstrap a sample using sampling with replacement, then fit the model using that sample, and store the age and education coefficients of the model in a vector.

Once we have 1000 estimates for each variable (age and education), we use quantile function to obtain the confidence intervals.

```
# YOUR CODE HERE
boot_age <- c()
boot_educ <- c()

for (i in 1:1000){
  # create a bootstrapped sample
  new_data_index <- sample(nrow(lalonde), nrow(lalonde), replace=TRUE)
  boot_lalonde <- lalonde[new_data_index, ]

  # fit the model
```

```
  lg42 <- glm(treat~age + educ + hisp + re74 + re75, data=boot_lalonde, family=binomial)

  # store the coefficients in vectors
  boot_age[i] <- coef(lg42)[2]
  boot_educ[i] <- coef(lg42)[3]
}

# age
ag42 <- quantile(boot_age, c(0.025, 0.975))

# educ
educ42 <- quantile(boot_educ, c(0.025, 0.975))

list("CI for Age" = ag42, "CI for Education"=educ42)
```

```
## $`CI for Age`
##        2.5%       97.5%
## -0.01702296  0.04001726
##
## $`CI for Education`
##        2.5%       97.5%
## -0.04077733  0.19960608
```

Report bootstrapped confidence intervals for `age` and `education` here.

After simulating the coefficients a thousand times on slightly different samples, we can confirm that variables Age and Education are not statistically significant for prediction because 0 is included in both intervals for age ([-0.018, 0.039]) and education ([-0.045, 0.188]). Although the predictors are not statistically significant, they may still be valuable in prediction, they just might not add as much predictive power. We will see how education may still contribute to the increase of probability of observation getting a treatment in the steps below.

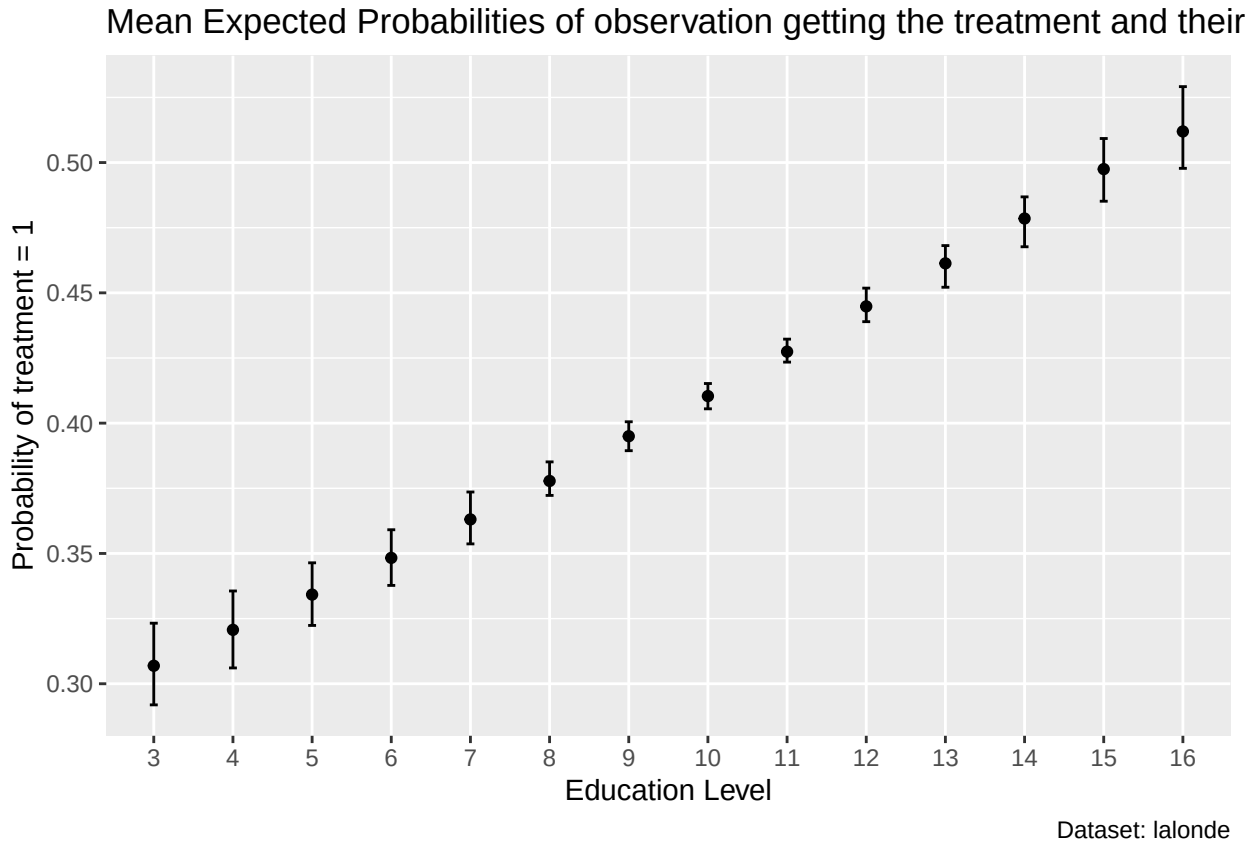
STEP 3 Then, using the simulation-based approach and the `arm` library, set all the predictors at their means EXCEPT `education`, create a data visualization that shows the 95% confidence interval of the expected values of the probability of receiving treatment as education varies from 3 to 16. Be sure to include axes labels and figure titles.

Here, for each education level k, we want to obtain 100 Expected Values j. For each j, we create a new set of simulated models i, we use each model to create a prediction (which we convert to a probability), and average the predictions to get the Expected Value. Then, we store that expected value in a storage.

After we obtain j expected values for each education level, we find the mean and CI of each level and plot them.

```
# YOUR CODE HERE

lg41 <- glm(treat~age + educ + hisp + re74 + re75, data=lalonde, family=binomial)

exp_vals2 <- c()

for (k in 3:16){
  temp_exp_vals2 <- c()
  for (j in 1:100)
  {
    sim42 <- sim(lg41, n=100)
    vsc2 <- c()
```

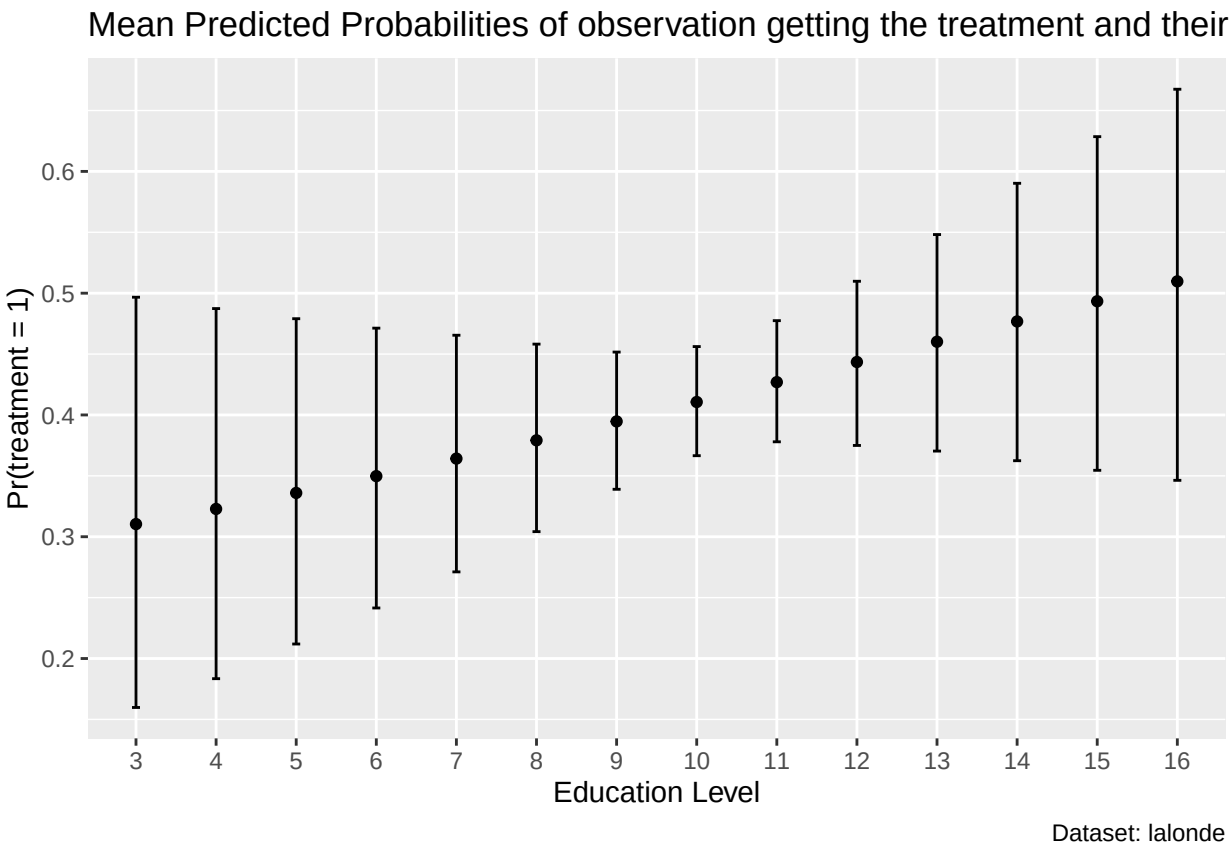
```
    for (i in 1:100)
      {
      lg_model <- sum(coef(sim42)[i, ]*c(1, mean(lalonde$age), k, mean(lalonde$hisp), mean(lalonde$re74)
      probability = 1/(1 + 2.718**(-lg_model))
      vsc2[i] <- probability
    }
    temp_exp_vals2[j] <- mean(vsc2)
  }
  exp_vals2[[k]] <- temp_exp_vals2
}


quants_exp_low2 <- c()
for (i in 1:14) {
  quants_exp_low2[i] = quantile(exp_vals2[[i+2]], probs = c(0.025))
}

quants_exp_high2 <- c()
for (i in 1:14) {
  quants_exp_high2[i] = quantile(exp_vals2[[i+2]], probs = c(0.975))
}

mean_exp2 <- c()
for (i in 1:14){
  mean_exp2[i] <- mean(exp_vals2[[i+2]])
}

df42 <- data.frame(cbind(c(3:16), mean_exp2, quants_exp_low2, quants_exp_high2))

ggplot(data=df42, aes(x=as.factor(V1), y=mean_exp2)) +
  geom_point() +
  geom_errorbar(aes(ymin=quants_exp_low2, ymax=quants_exp_high2), width=.1) +
  labs(title="Mean Expected Probabilities of observation getting the treatment and their 95% CI", x="Edu
```



STEP 4 Then, do the same thing, but this time for the predicted values of the probability of receiving treatment as education varies from 3 to 16. Be sure to include axes labels and figure titles.

```
# YOUR CODE HERE
vsfin4 <- list()

lg41 <- glm(treat~age + educ + hisp + re74 + re75, data=lalonde, family=binomial)

sim42 <- sim(lg41, n=1000)

for (k in 3:16){
  vsc4 <- c()
  for (i in 1:1000){
    linear_logit <- sum(coef(sim42)[i, ]*c(1, mean(lalonde$age), k, mean(lalonde$hisp), mean(lalonde$re
    probability = 1/(1 + 2.718**(-linear_logit))
    vsc4[i] <- probability
  }
  vsfin4[[k]] <- vsc4
}

quants4_low <- c()
for (i in 1:14) {
  quants4_low[i] = quantile(vsfin4[[i+2]], probs = c(0.025))
}

quants4_high <- c()
```

```r
for (i in 1:14) {
  quants4_high[i] = quantile(vsfin4[[i+2]], probs = c(0.975))
}

mean_vsfin4 <- c()
for (i in 1:14){
  mean_vsfin4[i] <- mean(vsfin4[[i+2]])
}


df42 <- data.frame(cbind(c(3:16), mean_vsfin4, quants4_low, quants4_high))


ggplot(data=df42, aes(x=as.factor(V1), y=mean_vsfin4)) +
  geom_point() +
  geom_errorbar(aes(ymin=quants4_low, ymax=quants4_high), width=.1) +
  labs(title="Mean Predicted Probabilities of observation getting the treatment and their 95% CI", x="E
```



STEP 5 Lastly, write a short paragraph with your reflections on this exercise and the results you obtained.

Similarly to the reflection in Qustion 4, we can see that the confidence interval for expected values is much narrower. We can confirm with smaller standard deviation (10x smaller than for predicted values) and narrower range the distribution in the figures below.

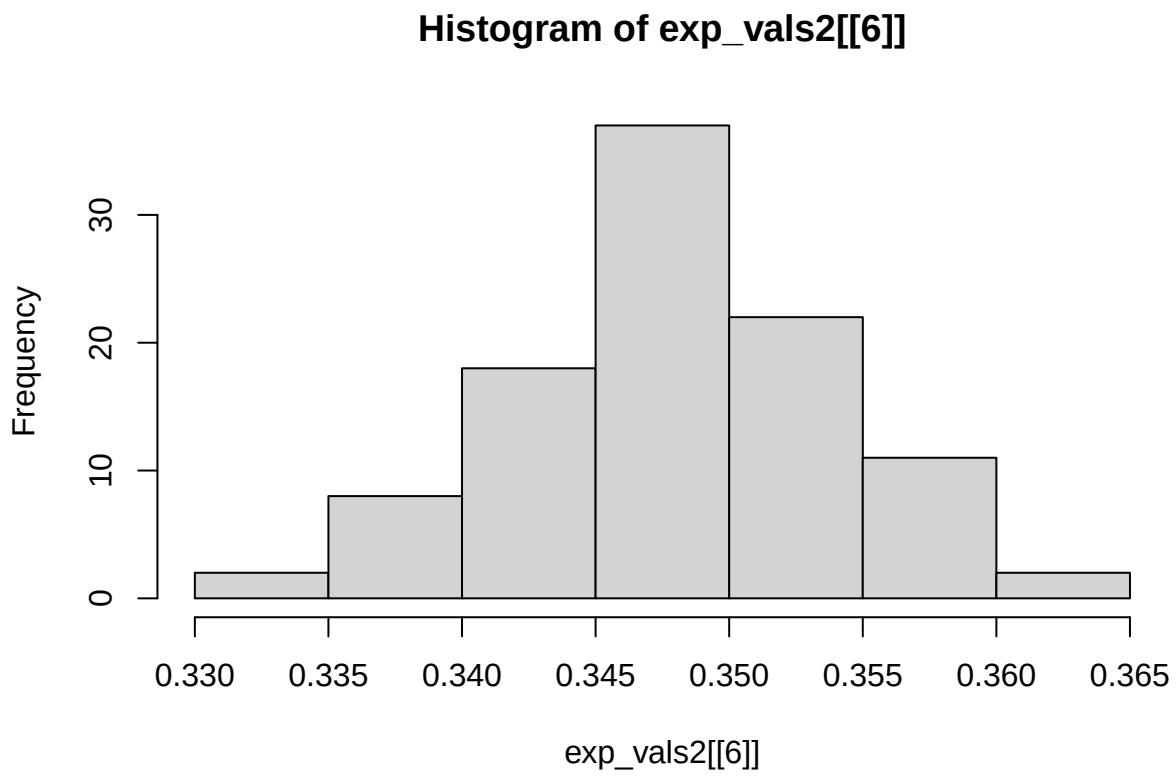
For both Question 4 and Question 5, we can conclude that presenting results from a single model is unreliable because the coefficients may be unluckily drawn from the edges of the distributions of possible coefficients.

This would give us a wrong understanding of the relationships between the variables. False coefficients would also skew our predicted values. Therefore, a better way is to run many simulations of a model, and use each simulation to predict a value for one observation. Then, we can average all the predictions and get a more accurate prediction value with a confidence interval.

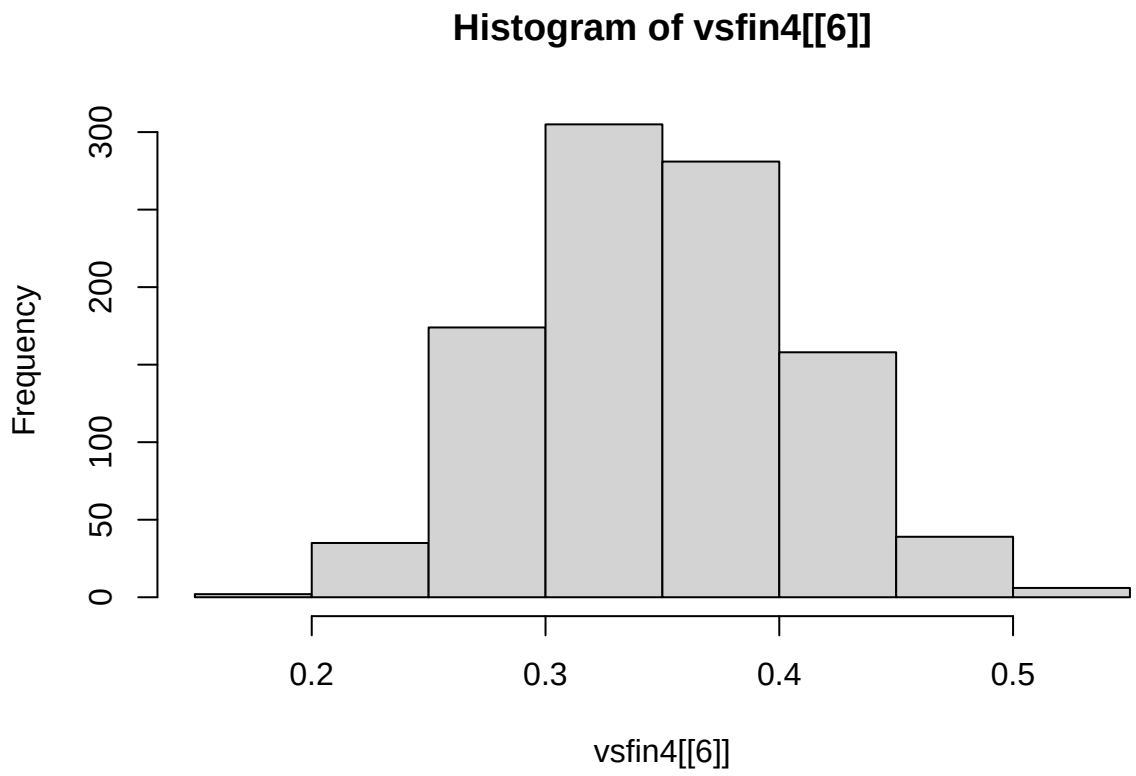
```
sprintf("SD of Expected Values is %f. SD of Predicted valeus is %f", sd(exp_vals2[[6]]), sd(vsfin4[[6]]))
```

```
## [1] "SD of Expected Values is 0.005980. SD of Predicted valeus is 0.058344"
```

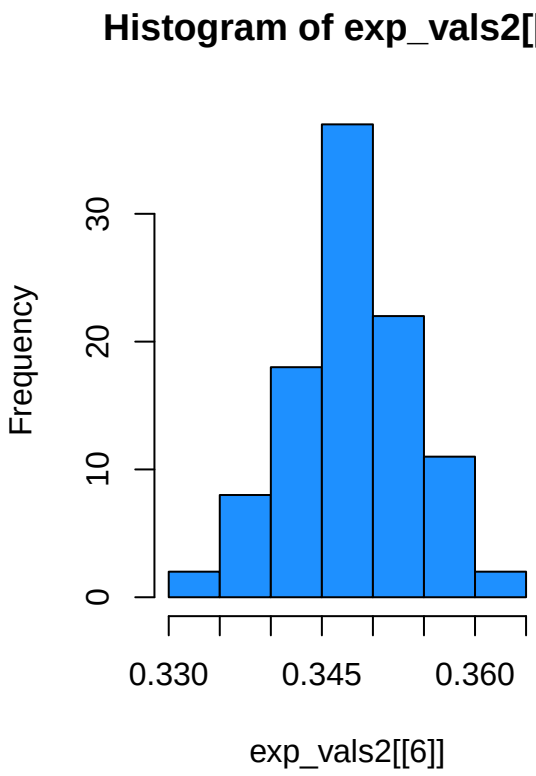
```
hgC = hist(exp_vals2[[6]])
```

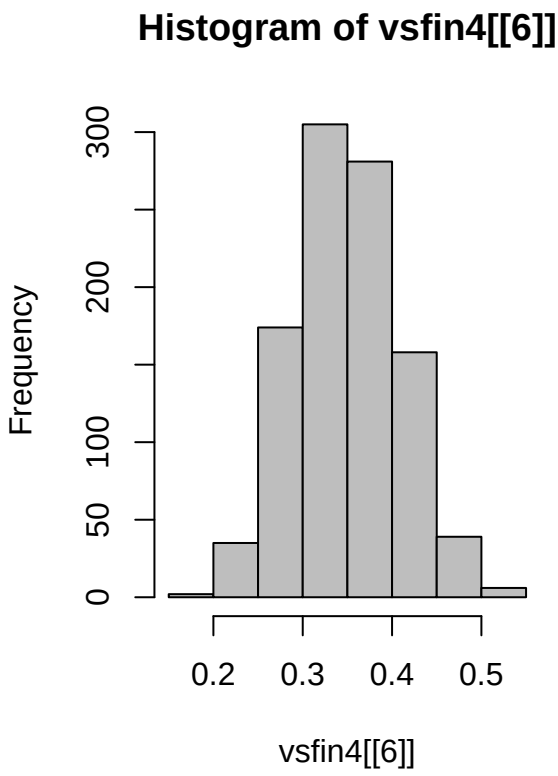



```
hgD = hist(vsfin4[[6]])
```




```
{
par(mfrow = c(1, 2))
plot(hgC, col='dodgerblue1', freq=T)
plot(hgD, col = 'gray', freq=T)
}
```



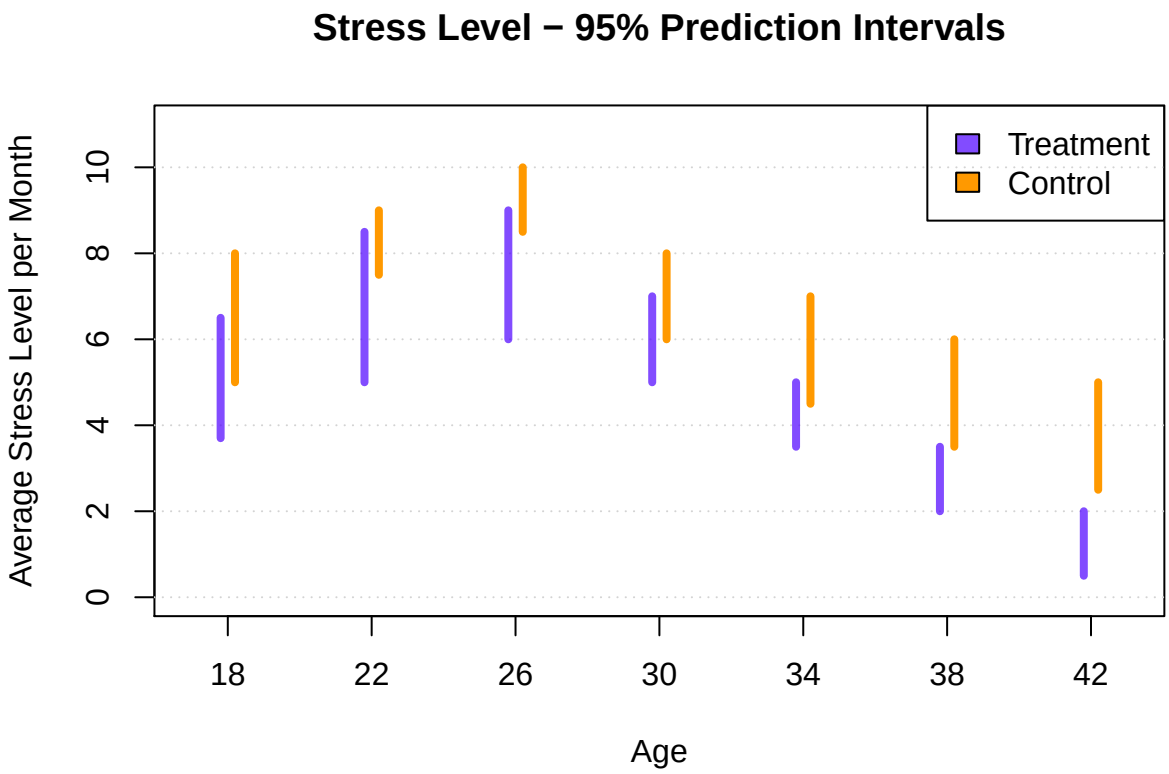


QUESTION 5

Write the executive summary for a decision brief about the impact of a stress therapy program, targeted at individuals age 18-42, intended to reduce average monthly stress. The program was tested via RCT, and the results are summarized by the figure that you get if you run this code chunk:

```
# Note that if you knit this document, this part of the code won't
# show up in the final pdf which is OK. We don't need to see the code
# we wrote.

# How effective is a therapy method against stress

# Participants in the study record their stress level for a month.
# Every day, participants assign a value from 1 to 10 for their stress level.
# At the end of the month, we average the results for each participant.

#adds the confidence interval (first row of the matrix is lower
# bound, second row is the upper bound)
trt1 = matrix(NA,nrow=2,ncol=7)
ctrl = matrix(NA,nrow=2,ncol=7)

trt1[,1]=c(3.7, 6.5) #18
ctrl[,1]=c(5, 8)

trt1[,2]=c(5, 8.5) #22
ctrl[,2]=c(7.5, 9)
```

```
trt1[,3]=c(6, 9) #26
ctrl[,3]=c(8.5, 10)

trt1[,4]=c(5, 7) #30
ctrl[,4]=c(6, 8)

trt1[,5]=c(3.5, 5) #34
ctrl[,5]=c(4.5, 7)

trt1[,6]=c(2, 3.5) #38
ctrl[,6]=c(3.5, 6)

trt1[,7]=c(0.5, 2) #42
ctrl[,7]=c(2.5, 5)

# colors to each group
c1 = rgb(red = 0.3, green = 0, blue = 1, alpha = 0.7) #trt1
c2 = rgb(red = 1, green = 0.6, blue = 0, alpha = 1) #trt2
c3 = rgb(red = 0, green = 0.5, blue = 0, alpha = 0.7) #ctrl

plot.new()
# creates the background of the graph
{plot(x = c(1:100), y = c(1:100),
     type = "n",
     xlim = c(17,43),
     ylim = c(0,11),
     cex.lab=1,
     main = "Stress Level - 95% Prediction Intervals",
     xlab = "Age",
     ylab = "Average Stress Level per Month",
     xaxt = "n")

axis(1, at=seq(18,42,by=4), seq(18, 42, by=4))

grid(nx = NA, ny = NULL, col = "lightgray", lty = "dotted",
     lwd=par("lwd"), equilogs = TRUE)

# adds the legend
legend('topright',legend=c('Treatment','Control'),fill=c(c1,c2))

# iterates to add stuff to plot
for (age in seq(from=18,to=42,by=4)) {
  #treatment
  segments(x0=age-0.2, y0=trt1[1, (age-18)/4+1],
           x1=age-0.2, y1=trt1[2, (age-18)/4+1], lwd=4, col=c1)

  #control
  segments(x0=age+0.2, y0=ctrl[1, (age-18)/4+1],
           x1=age+0.2, y1=ctrl[2, (age-18)/4+1], lwd=4, col=c2)
}
}
```



(Not that it matters, really, but you can imagine that these results were obtained via simulation, just like the results you have hopefully obtained for question 2 above).

Your executive summary should be between about 4 and 10 sentences long, it should briefly describe the the purpose of the study, the methodology, and the policy implications/prescription. (Feel free to imaginatively but realistically embellish/fill-in-the-blanks with respect to any of the above, since I am not giving you backstory here).

The RCT aims at measuring the impact of a stress therapy program by analyzing the average monthly stress. The units were grouped by different age categories in range 18-42. We sampled 100 people from each age category, resulting in a stratified sample. Then, we grouped by age and randomly assigned people to either treatment or control groups. Over the next 12 month, we surveyed the participants of both groups weekly, asking about their life, stressful events, and levels of stress. We averaged the stress levels for each month, and obtained the graph above.

From the graph, we can see that for categories 18, 22, 26, 30 and 34, the 95% confidence intervals intersect. This is not good because CI tells us that we are 95% confident that the true mean of the group is within that range. If we look at age 18, we can see that the true estimate of the Control population could be 6 (at the lower bound), but the same goes for treatment - the true estimate for Treatment population could be 6 (at the upper bound). So, if the true parameter for both groups is 6, then there is not effect of treatment on the potential outcome. Therefore, for categories 18 - 34, we **do not reject** our null hypothesis - treatment and control groups have the same average stress level.

If we look at categories 38 and 42, the CIs do not intersect, and hence we can be 95% confident that there is a statistically significant difference. We conclude that the stress therapy program significantly reduces average monthly stress of its participants.

Our final decision is to (1) implement the stress therapy program for people of age 38-42, (2) Revise the program for categories 18 - 34 and repeat the experiment in another settings.

QUESTION 6

Can we predict what projects end up being successful on Kickstarter?

We have data from the Kickstarter company.

From Wikipedia: Kickstarter is an American public-benefit corporation based in Brooklyn, New York, that maintains a global crowdfunding platform focused on creativity and merchandising. The company's stated mission is to "help bring creative projects to life". As of May 2019, Kickstarter has received more than $4 billion in pledges from 16.3 million backers to fund 445,000 projects, such as films, music, stage shows, comics, journalism, video games, technology, publishing, and food-related projects.

The data is collected by Mickaël Mouillé and is last updated in 2018. Columns are self explanatory. Note that `usd_pledged` is the column `pledged` in US dollars (conversion done by kickstarter) and `usd_pledge_real` is the `pledged` column in real US dollars of the pledged column. Finally, `usd_goal_real` is the column `goal` in real US dollars. You should use the real columns.

So what makes a project successful? Undoubtedly, there are many factors, but perhaps we could set up a prediction problem here, similar to the one from the bonus part of the last assignment where we used GDP to predict personnel contributions.

We have columns representing the the number of backers, project length, the main category, and the real project goal in USD for each project.

Let's explore the relationship between those predictors and the dependent variable of interest — the success of a project.

Instead of running a simple linear regression and calling it a day, let's use cross-validation to make our prediction a little more sophisticated.

Our general plan is the following:

1. Build the model on a training data set
2. Apply the model on a new test data set to make predictions based on the inferred model parameters.
3. Compute and track the prediction errors to check performance using the mean squared difference between the observed and the predicted outcome values in the test set.

Let's get to it, step, by step. Make sure you have loaded the necessary packages for this project.

STEP 1: Import & Clean the Data Import the dataset from this link: https://tinyurl.com/KaggleDataCS112

Remove any rows that include missing values.

```
# YOUR CODE HERE
df60 <- read.csv("https://tinyurl.com/KaggleDataCS112")

df60 <- na.omit(df60)

# check for mising values across each column
sapply(df60, function(x) sum(is.na(x)))
```

```
##               ID             name         category    main_category
##                0                0                0                0
##         currency         deadline             goal         launched
##                0                0                0                0
##          pledged            state          backers          country
##                0                0                0                0
##      usd.pledged usd_pledged_real    usd_goal_real
##                0                0                0
```

STEP 2: Codify outcome variable Create a new variable that is either successful or NOT successful and call it `success` and save it in your dataframe. It should take values of 1 (successful) or 0 (unsuccessful).

```
# YOUR CODE HERE
df61 <- df60 %>%
  mutate(success = as.factor(ifelse(state=="successful", 1, 0)))
```

STEP 3: Getting the project length variable Projects on Kickstarter can last anywhere from 1 - 60 days. Kickstarter claims that projects lasting any longer are rarely successful and campaigns with shorter durations have higher success rates, and create a helpful sense of urgency around your project. Using the package `lubridate` or any other package in R you come across by Googling, create a new column that shows the length of the project by taking the difference between the variable `deadline` and the variable `launched`. Call the new column `length` and save it in your dataframe.

Remove any project length that is higher than 60.

```
# YOUR CODE HERE
df62 <- df61 %>%
  mutate(deadline = as.Date(df61$deadline),launched = as.Date(df61$launched)) %>%
  mutate(length = deadline - launched) %>%
  filter(length <= 60)
```

STEP 4: Splitting the data into a training and a testing set While there are several variations of the k-fold cross-validation method, let's stick with the simplest one where we just split randomly the dataset into two (k = 2) and split our available data from the link above into a training and a testing (aka validation) set.

Randomly select 80% of the data to be put in a training set and leave the rest for a test set.

```
# YOUR CODE HERE
train_index = sample(nrow(df62), 0.8*nrow(df62), replace=F)
df62_train <- df62[train_index, ]

df62_test <- df62[-train_index, ]
```

STEP 5: Fitting a model Use a logistic regression to find what factors determine the chances a project is successful. Use the variable indicating whether a project is successful or not as the dependent variables (Y) and number of backers, project length, main category of the project, and the real project goal as independent variables. Make sure to use the main category as factor.

```
# YOUR CODE HERE
lg60 <- glm(success ~ backers + length + as.factor(main_category) + usd_goal_real, data=df62_train, fam
```

STEP 6: Predictions Use the model you've inferred from the previous step to predict the success outcomes in the test set.

```
# YOUR CODE HERE
predicted_vals <- predict(lg60, newdata=df62_test, type='response')

predicted_vals_binary <- ifelse(predicted_vals > 0.5, 1, 0)

conf_matrix <- table(df62_test$success, predicted_vals_binary)
conf_matrix
```

```
##    predicted_vals_binary
##         0     1
##   0 45692  1707
```

```
##   1  6077 20402
```

STEP 7: How well did it do? Report the ""misclassification rate** of the predictions for the training and the test sets.

```
# YOUR CODE HERE
misclass_test <- (conf_matrix[[2]] + conf_matrix[[3]])/sum(conf_matrix)


# misclassification rate for train
predicted_vals_train <- predict(lg60, newdata=df62_train, type='response')

predicted_vals_binary_train <- ifelse(predicted_vals_train > 0.5, 1, 0)

conf_matrix_train <- table(df62_train$success, predicted_vals_binary_train)

misclass_train <- (conf_matrix_train[[2]] + conf_matrix_train[[3]])/sum(conf_matrix_train)

sprintf("Test error rate: %f, Train error rate: %f", misclass_test, misclass_train)
```

```
## [1] "Test error rate: 0.105363, Train error rate: 0.105938"
```

Step 8: LOOCV method Apply the leave-one-out cross validation (LOOCV) method to the training set. What is the misclassification rate of the training and test sets. How similar are the misclassification rates?

```
# YOUR CODE HERE
# getting 0.5% of the data
df63 <- df62[sample(nrow(df62), 0.005*nrow(df62), replace=F), ]

lg61 <- glm(success ~ backers + length + main_category + usd_goal_real, data=df63, family=binomial)

glm_obj <- cv.glm(df63, lg61)

#  The first component is the raw cross-validation estimate of prediction error.
sprintf("The missclassification rate with LOOCV method is %f", glm_obj$delta[1])
```

```
## [1] "The missclassification rate with LOOCV method is 0.081916"
```

Step 9: Explanations Compare the misclassification rates from the simple method to the LOOCV method? How do data scientists really use cross-validation? How is the approach in this project differ from real-world cases? Give an example to make your point!

As we would expect, LOOCV would give us a smaller misclassification rate (8% vs. 10%) because (1) LOOCV uses more data (n-1) to train each model, which makes the models more accurate. (2) LOOCV looks for the best model out of all n models, which would produce the best results. Meanwhile, with the simple method, we get a suboptimal model.

However, we should also consider that each of the LOOCV models is almost identical because removing 1 observation from such a huge dataset would have little impact. This leads to higher variance and lower generalizability when using the model in-real-life.

Additionally, the variable 'backers' indicates how many people have donated to the project. At the moment of creating the project and predicting its success, one cannot possibly know how many people would donate to it. The information comes from the future, leading to data leakage. Therefore, including this variable is not reasonable for prediction purposes.

To optimize our model for predicting whether the project will succeed we can: (1) Use K-fold cross-validation to decrease variation and increase generalizability. In K-fold, the training datasets are more diverse, resulting in more diverse models and thus higher generalizability. (2) Include more variables from the present, such as sentiment analysis of the project's name and description, and analysis of the photos and videos (3) Try different types of models, such non-parametric models which do not assume the S-shaped relationship between independent variables and the probability. We have a lot of data (350k) to justify our use of non-parametric models such as Random Forests or SVC.

Additional notes:

- For some of the questions, I collaborated with Ha Tran.

Extra Credit: Least Absolute Deviation Estimator

STEP 1 Figure out how to use rgenoud to run a regression that maximizes the least absolute deviation instead of the traditional **sum of the squared residuals**. Show that this works by running that regression on the `lalonde` data set with outcome being `re78` and independent variables being `age`, `education`, `hisp`, `re74`, `re75`, and `treat`.

```
# YOUR CODE HERE
```

STEP 2 How different is this coef on treat from the coef on treat that you get from the corresponding traditional least squares regression?

STEP 3 Now figure out how to do the same by using rgenoud to run the logistic regression (modeling treatment status as a function of `age`, `education`, `hisp`, `re74` and `re75`).

```
# YOUR CODE HERE
```

END OF Assignment!!!

Final Steps

Add Markdown Text to .Rmd

Before finalizing your project you'll want be sure there are **comments in your code chunks** and **text outside of your code chunks** to explain what you're doing in each code chunk. These explanations are incredibly helpful for someone who doesn't code or someone unfamiliar to your project. You have two options for submission:

1. You can complete this .rmd file, knit it to pdf and submit both the .rmd file and the .pdf file on Forum as one .zip file.
2. You can submit your assignment as a separate pdf using your favorite text editor and submit the pdf file along with a lint to your github code. Note that links to Google Docs are not accepted.

Knitting your R Markdown Document

Last but not least, you'll want to **Knit your .Rmd document into an HTML document**. If you get an error, take a look at what the error says and edit your .Rmd document. Then, try to Knit again! Troubleshooting these error messages will teach you a lot about coding in R. If you get any error that doesn't make sense to you, post it on Perusall.

A Few Final Checks

If you are submitting an .rmd file, a complete project should have:

- Completed code chunks throughout the .Rmd document (your RMarkdown document should Knit without any error)

- Comments in your code chunks
- Answered all questions throughout this exercise.

If you are NOT submitting an .rmd file, a complete project should have:

- A pdf that includes all the answers and their questions.
- A link to Github (gist or repository) that contais all the code used to answer the questions. Each part of you code should say which question it's referring to.