# CS112 Assignment 2, Fall 2020

Thiago Silva

10/24/2020

**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)

```r
#independent variables
found <- rnorm(1000, mean=5, sd=3)
hours <- rnorm(1000, mean=3, sd=1)

#error
alertness <- rnorm(1000, mean=0, sd=2)

#outcome variable
grades <- 0.4*found + 0.6*hours + 0.2*found*hours + alertness

#Creates a data frame using the independent and dependent variables
df <- data.frame(found, hours, grades)
```

**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?

In the independent variables are how much foundational knowledge the student had on the subject and how many hours they studied for a given test. Each one individually contributes to the grades, but there's also an interaction. The stochastic component comes from alertness during the test, which can't be measured. The outcome variable is the final grade.

**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.

```
#Imports the library used for simulations
library(arm)
```

```
## Loading required package: Matrix

## Loading required package: lme4

##
## arm (Version 1.11-2, built: 2020-7-27)

## Working directory is C:/Users/thiag/OneDrive/Área de Trabalho/Desktop/Folders & Files/Minerva Stuff/

##
## Attaching package: 'arm'

## The following object is masked from 'package:boot':
##
##     logit
```

```
#performs a simple linear regression on the dataframe
lm1 <- lm(grades ~ found + hours, data=df)

#Makes one simulation of the coefficients using the sim function
s1 <- sim(lm1, n.sims=1)

#Extracts the coefficients from the simulation into easier to use variables
s1.intercept <- coef(s1)[1]
s1.found <- coef(s1)[2]
s1.hours <- coef(s1)[3]

#Calculate the predictions for the dataset based on the linear regression
s1.pred <- s1.intercept + s1.found*found + s1.hours*hours

calc_rmse <- function(actual, pred){
#Calculates the Root Mean Squared Error between two vectors of the same size

  rmse <- (mean((actual-pred)**2))**(1/2)
  return (rmse)
}

#Outputs RMSE of the calculated predictions relative to the actual grades
cat("The RMSE of the incorrect model is: ", calc_rmse(grades,s1.pred))
```

```
## The RMSE of the incorrect model is:  2.068344
```

**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.

```r
#Performs a linear regression on the dataframe, including the interaction
#between the independent variables
lm2 <- lm(grades ~ found + hours + found:hours, data=df)

#simulates one set of coefficients using the linear regression
s2 <- sim(lm2, n.sims=1)

#Extracts the coefficients from the simulation into easier to use variables
s2.intercept <- coef(s2)[1]
s2.found <- coef(s2)[2]
s2.hours <- coef(s2)[3]
s2.interaction <- coef(s2)[4]

#Calculate the predictions for the dataset based on the linear regression
s2.pred <- s2.intercept + s2.found*found + s2.hours*hours + s2.interaction*hours*found

cat("The RMSE of the correct model is: ", calc_rmse(grades,s2.pred))
```

```
## The RMSE of the correct model is:  2.002471
```

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

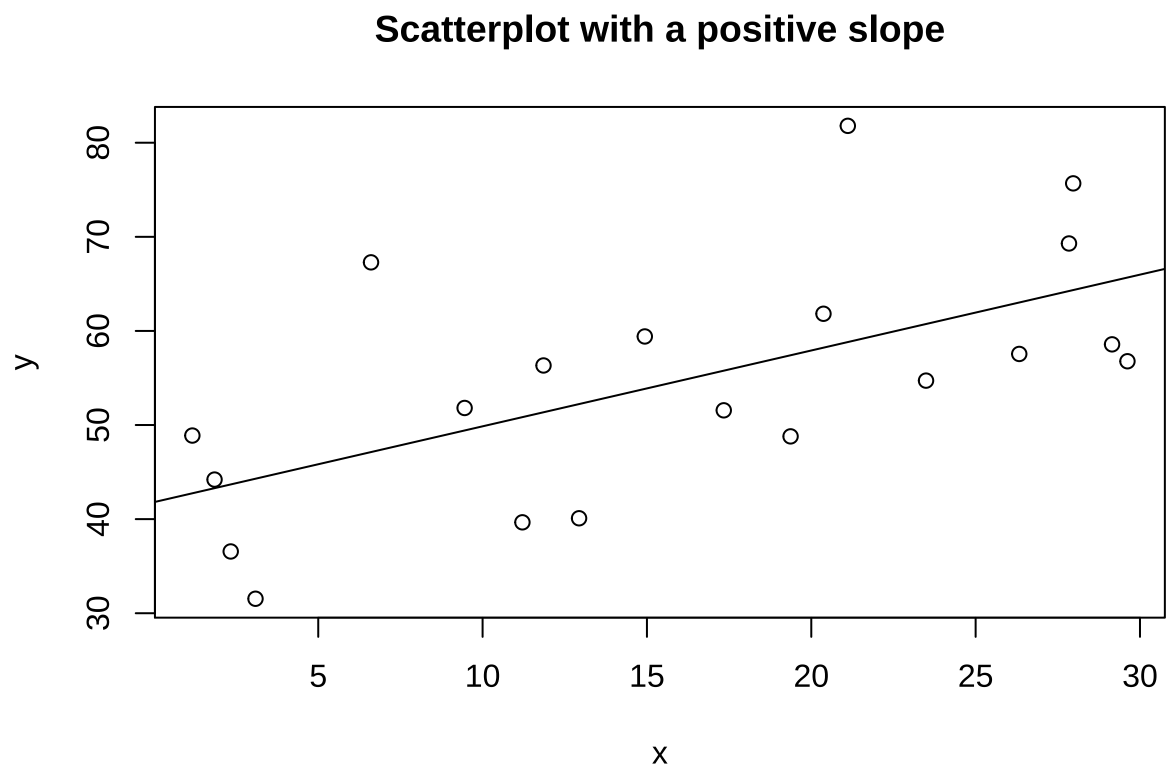
The RSME of the incorrect model is larger. Due to being a better representation of how the data was generated, the correct model is expected to yield a better fit, and therefore have a lower RMSE. In this case, the correction of the model was achieved by considering the interaction between the variables, which wasn't present in the incorrect model but had a significant influence on the outcome variable

## 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.

```r
#Generates 20 random numbers, uniformly distributed between 0 and 30
x = runif(20, min=0, max=30)

#Creates a dependent variable with a linear relationship with x, with an
#intercept of 35 and normally distributed error
y = 1*x + 35 + rnorm(10, mean=0, sd=10)

#Creates a simple scatterplot of the data, with the regression line
plot(x, y, main="Scatterplot with a positive slope")
regression <- lm(y~x)
abline(regression)
```

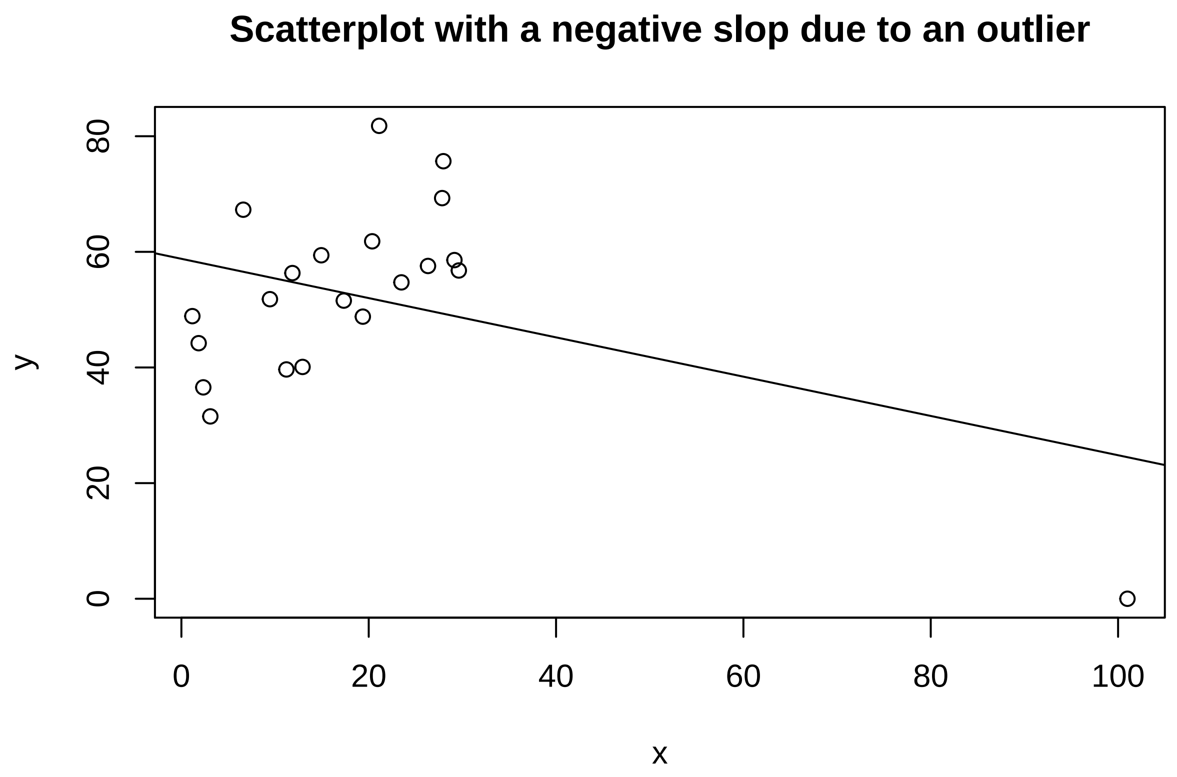


```r
#Prints the slope of the line (Positive)
cat("The slope of the regression line is: ", coef(regression)[2])
```

```
## The slope of the regression line is:  0.8059633
```

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.

```r
#Creates new vectors that are the same as the previous with the addition of an outlier
x <- c(x,101)
y <- c(y,0)

#Creates a simple scatterplot of the data, with the regression line
plot(x, y, main="Scatterplot with a negative slop due to an outlier")
regression <- lm(y~x)
abline(regression)
```



```r
#Prints the slope of the line (Negative)
cat("The slope of the regression line is: ", coef(regression)[2])
```

```
## The slope of the regression line is:  -0.3397702
```

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.

To create the data, I created X as a vector with 20 uniformly distributed samples between 0 and 30. Then, I calculated a variable Y with a linear relationship to the independent variable, with the addition of an intercept of 35 and a normally distributed error. I then added an outlier with X significantly larger than the any other point (100), and Y of 0, which was sufficient to make the regression negative and show the sensitivity of the data to an outlier.

## 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`.

```r
#Loads the lalonde data
data(lalonde)

#Creates a linear regression for the lalonde dataset
lalonde.lm <- lm(re78 ~ age + educ + re74 + re75 + hisp + black, data=lalonde)
```

**STEP 2** Report coefficients and R-squared.

```r
#Source: https://stat.ethz.ch/pipermail/r-help/2010-January/225345.html

cat("The Coefficients are: \n")
```

```
## The Coefficients are:
```

```r
coef(lalonde.lm)
```

```
##   (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
cat("\n The Rsquared is: ", summary(lalonde.lm)$r.squared)
```

```
##
##  The Rsquared is:  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.

```r
#Predicts the outcome of the lalonde dataset using the linear regression
lalonde.pred <- predict(lalonde.lm, newdata=lalonde)

#Calculates the unexplained variance (Sum of the Squares of the Predictor errors)
lalonde.varun <- sum((lalonde$re78-lalonde.pred)**2)

#Calculates the total variance (Sum of the squares of the differences between
#each outcome and the mean outcome)
lalonde.totalvar <- sum((lalonde$re78-mean(lalonde$re78))**2)

#Calculates the R-squared as 1-(Unex. Var./Total Var.)
rsquared <- 1 - lalonde.varun/lalonde.totalvar

cat("The R-squared is:", rsquared)
```

```
## The R-squared is: 0.03942359
```

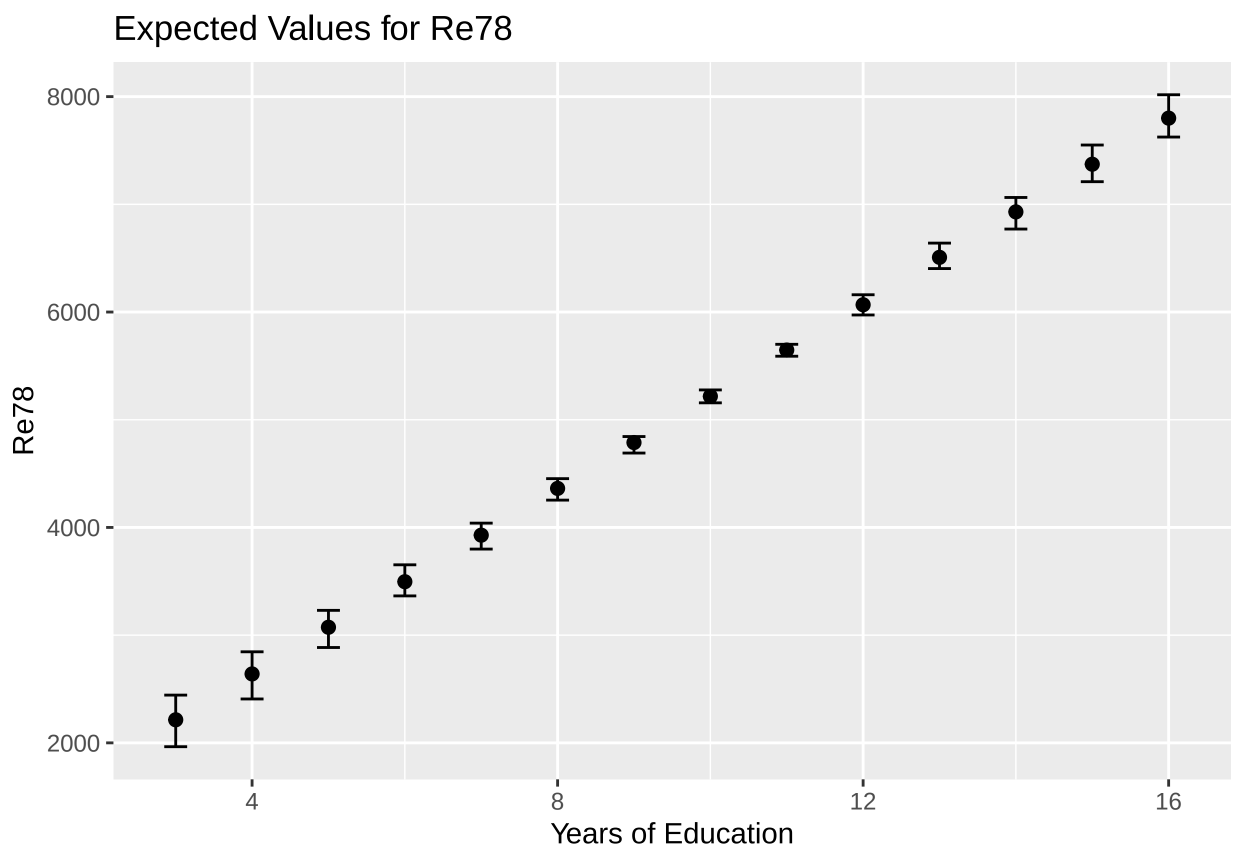
**STEP 3** Then, setting all the predictors at their means EXCEPT `education`, create a data visualization that shows the 95% confidence interval of the expected values of `re78` as `education` varies from 3 to 16. Be sure to include axes labels and figure titles.

```r
#Source: http://www.cookbook-r.com/Graphs/Plotting_means_and_error_bars_(ggplot2)/

#Imports the ggplot2 library, to be used in the dataviz
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin
```

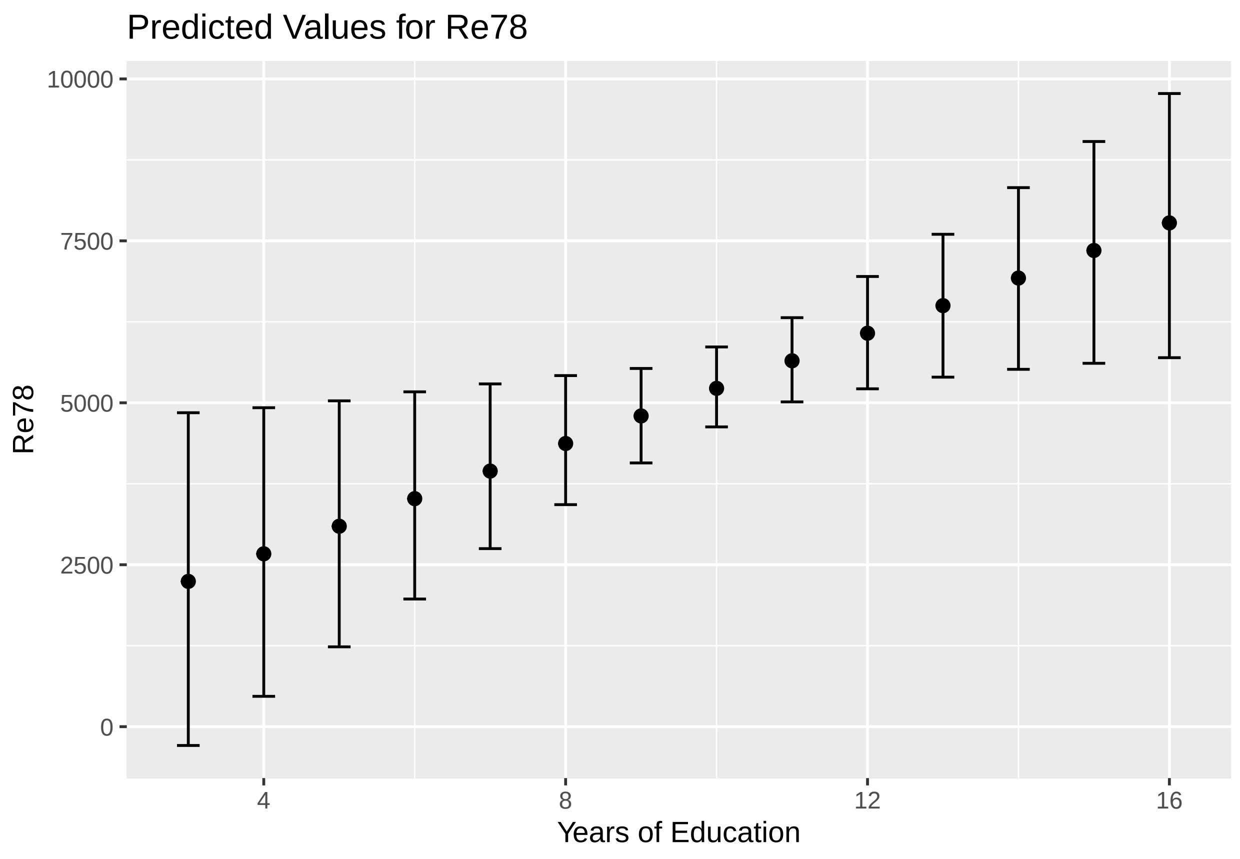
```r
#Calculate the means for all predictors except education
mean_age <- mean(lalonde$age)
mean_re74 <- mean(lalonde$re74)
mean_re75 <- mean(lalonde$re75)
mean_hisp <- mean(lalonde$hisp)
mean_black <- mean(lalonde$black)

#Creates the vector for the years of education
educs <- c(3:16)

#Creates empty vector for the average result and confidence interval obtained
#as education varies from 3 to 16 years
mean <- c()
lower <- c()
upper <- c()

#For each choice of education years
for (i in 1:length(educs)){
  simResults <- c()

  for (j in 1:100){
    #Simulates the coefficients using the linear regression
    simCoef <- sim(lalonde.lm, n.sims=100)

    #Calculates the results using the coefficients obtained from the
    #simulation, averages the results
    simResults[j] <- mean(
              coef(simCoef)[,1] +
              coef(simCoef)[,2]*mean_age +
              coef(simCoef)[,3]*educs[i] +
              coef(simCoef)[,4]*mean_re74 +
              coef(simCoef)[,5]*mean_re75 +
              coef(simCoef)[,6]*mean_hisp +
              coef(simCoef)[,7]*mean_black
              )
  }

  #Calculates the mean and the confidence interval of the data
  lower[i] <- quantile(simResults, 0.025)
  upper[i] <- quantile(simResults, 0.975)
  mean[i] <- mean(simResults)
}

#Puts all variables relevant to the data visualization in a data frame
education.df <- data.frame(educs, mean, lower, upper)

#Uses ggplot to plot the confidence intervals
```

```r
ggplot(education.df, aes(x = educs, y = mean)) +
        geom_point(size=2)+
        geom_errorbar(aes(ymax = upper, ymin = lower, width=0.3)) +
        labs(x="Years of Education", y = "Re78", title = "Expected Values for Re78")
```



**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.

```r
#Simulates the coefficients a 1000 times
simCoef <- sim(lalonde.lm, n.sims=1000)

#Creates the vector for the years of education
educs <- c(3:16)

#Creates empty vector for the average result and confidence interval obtained
#as education varies from 3 to 16 years
mean <- c()
lower <- c()
upper <- c()

#For each choice of education years
for (i in 1:length(educs)){

  #Calculates the results using the coefficients obtained from the simulation
  simResults <- coef(simCoef)[,1] +
              coef(simCoef)[,2]*mean_age +
              coef(simCoef)[,3]*educs[i] +
              coef(simCoef)[,4]*mean_re74 +
              coef(simCoef)[,5]*mean_re75 +
              coef(simCoef)[,6]*mean_hisp +
              coef(simCoef)[,7]*mean_black

  #Calculates the mean and the confidence interval of the data
  lower[i] <- quantile(simResults, 0.025)
  upper[i] <- quantile(simResults, 0.975)
  mean[i] <- mean(simResults)
}

#Puts all variables relevant to the data visualization in a data frame
education.df <- data.frame(educs, mean, lower, upper)

#Uses ggplot to plot the confidence intervals
ggplot(education.df, aes(x = educs, y = mean)) +
        geom_point(size=2)+
        geom_errorbar(aes(ymax = upper, ymin = lower, width=0.3)) +
        labs(x="Years of Education", y = "Re78", title = "Predicted Values for Re78")
```



**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.

Given that the confidence interval for the expected values are calculated using the mean from a hundred simulations instead of the simulation's results, there is much less deviation and therefore the confidence interval is much smaller. The means, however, of the predicted and expected values are very close to each other.

## 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`.

```
#Performs the Logistic Regression
lalonde.logit <- glm(treat ~ age + educ + re74 + re75 + hisp, data=lalonde,
                     family = "binomial")

#Outputs the coefficients
cat("The regression coefficient of age is: ", coef(lalonde.logit)[2], "\n")
```

```
## The regression coefficient of age is:  0.01164838
```

```
cat("The regression coefficient of education is: ", coef(lalonde.logit)[3], "\n")
```

```
## The regression coefficient of education is:  0.06922206
```

```
#Outputs the confidence interval
cat("The confidence interval is given by: \n")
```

```
## The confidence interval is given by:
```

```
confint(lalonde.logit)[c("age", "educ"),]
```

```
## Waiting for profiling to be done...
```

```
##             2.5 %     97.5 %
## age  -0.01538200 0.03852192
## educ -0.03853866 0.17958880
```

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

The coefficients in the linear regression indicate the relationship between the natural log of odds ratio of the outcome variable and the associated predictor. In this case, the coefficient for age is 0.0116, so it is expected that log of odds will increase by 0.0116 for each additional year. Due to the uncertainty of the data, the "true coefficient" is not known, but calculated to be between -0.0154 and 0.0385. Similarly, the expected proportion between log of odds ratio and education is 0.0692, with 95% chance that the true value is between -0.0386 and 0.1796.

**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.

```r
#Creates the vector to store bootstraped data for age and education
ages <- c()
educations <- c()

#Defines there will be 1000 simulations in the bootstrap
for(i in 1:1000){
  #Samples indexes for this iteration of the bootstrap, with replacement
  index <- sample(1:nrow(lalonde), nrow(lalonde), replace=TRUE)

  #Creates a new dataset with the sampled indexes
  newLalonde <- lalonde[index, ]

  #Runs a logistical regression on the dataset
  newLalonde.logit <- glm(treat ~ age + educ + re74 + re75 + hisp,
                          data=newLalonde, family = "binomial")

  #Adds the coefficients to the vectors that will store them
  ages[i] <- coef(newLalonde.logit)[2]
  educations[i] <- coef(newLalonde.logit)[3]
}
cat("Confidence interval for age coefficient: \n")
```

```
## Confidence interval for age coefficient:
```

```r
quantile(ages,c(0.025,0.975))
```

```
##        2.5%       97.5%
## -0.01740007  0.04075321
```

```r
cat("Confidence interval for the education coefficient: \n")
```

```
## Confidence interval for the education coefficient:
```

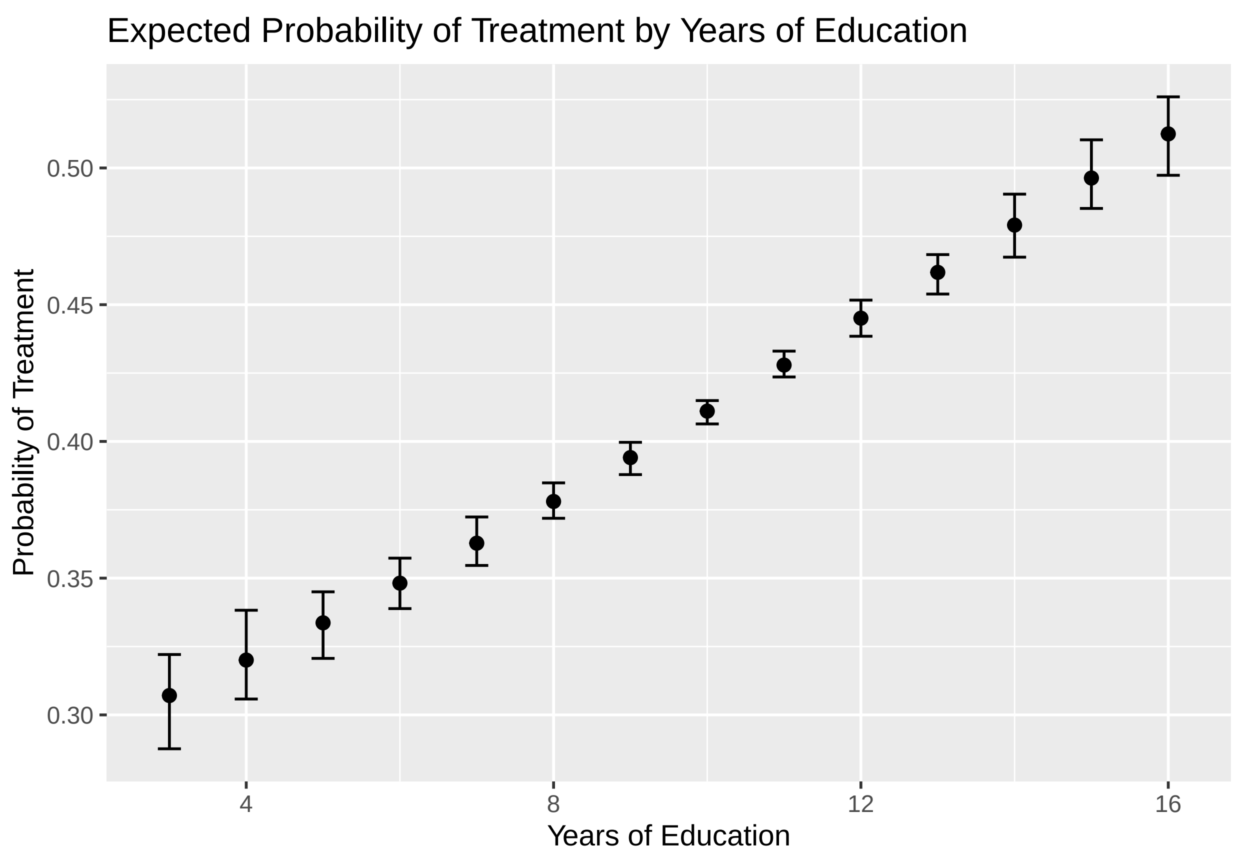
```r
quantile(educations,c(0.025,0.975))
```

```
##        2.5%       97.5%
## -0.04481417  0.19158729
```

Report bootstrapped confidence intervals for `age` and `education` here. Bootstraped confidence interval for the age coefficient: {-0.0174,0.0407} Bootstraped confidence interval for the education coefficient: {-0.0448,0.1916}

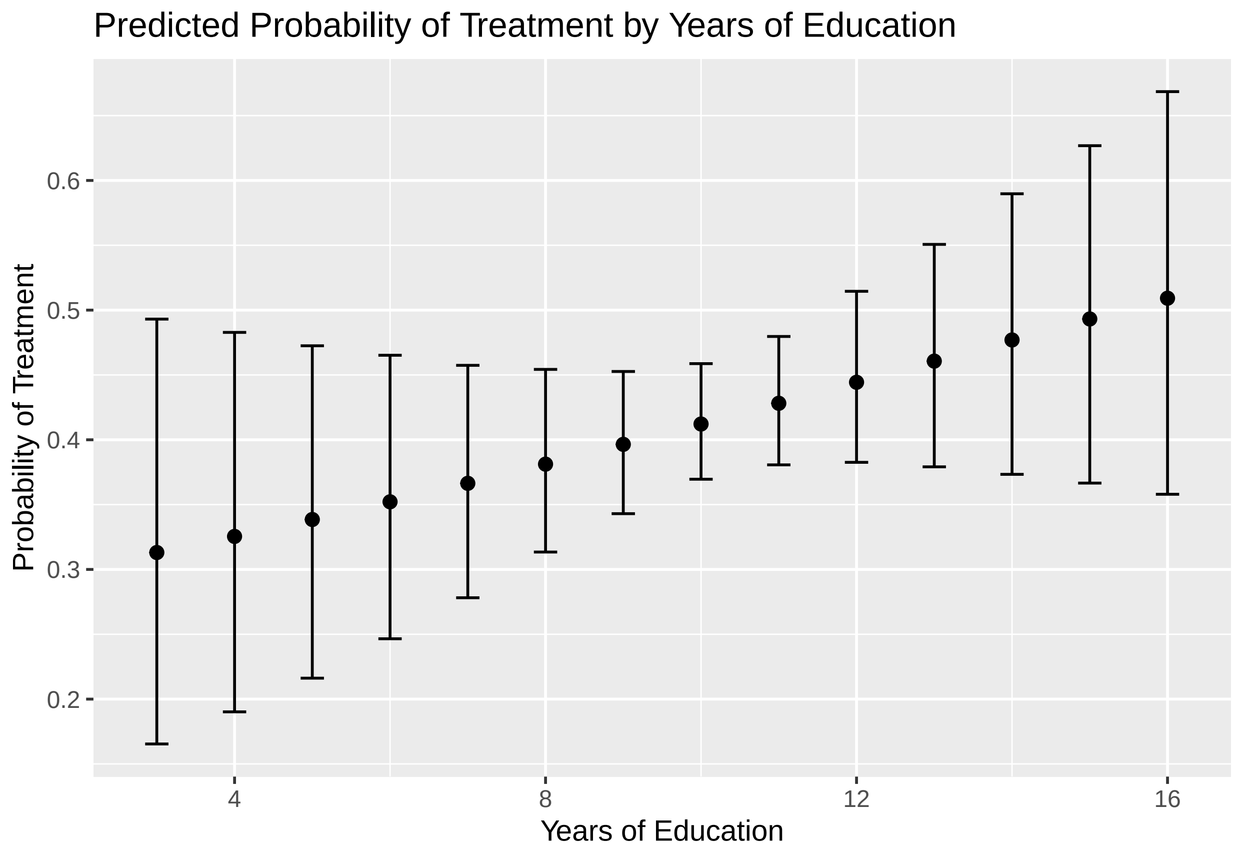
**STEP 3** Then, using the simulation-based approach and the `arm` library, set all the predictors at their means EXCEPT `education`, create a data visualization that shows the 95% confidence interval of the expected values of the probability of receiving treatment as education varies from 3 to 16. Be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

```r
#Function that transforms logit in probability
#source("https://sebastiansauer.github.io/Rcode/logit2prob.R")
logit2prob <- function(logit){
  odds <- exp(logit)
  prob <- odds / (1 + odds)
  return(prob)
}

#Creates empty vectors for the average result and confidence interval obtained
#as education varies from 3 to 16 years
mean <- c()
lower <- c()
upper <- c()

#For each choice of education years
for (i in 1:length(educs)){
  simProb <- c()

  for (j in 1:100){
    #Simulates the coefficients using the linear regression
    simCoef <- sim(lalonde.logit, n.sims=100)

    #Calculates the results using the coefficients obtained from the
    #simulation, averages the results
    simLogit <- coef(simCoef)[,1] +
                coef(simCoef)[,2]*mean_age +
                coef(simCoef)[,3]*educs[i] +
                coef(simCoef)[,4]*mean_re74 +
                coef(simCoef)[,5]*mean_re75 +
                coef(simCoef)[,6]*mean_hisp

    simProb[j] <- mean(logit2prob(simLogit))
  }
  #Calculates the mean and the confidence interval of the data
  lower[i] <- quantile(simProb, 0.025)
  upper[i] <- quantile(simProb, 0.975)
  mean[i] <- mean(simProb)

}

#Puts all variables relevant to the data visualization in a data frame
education.df <- data.frame(educs, mean, lower, upper)

#Uses ggplot to plot the confidence intervals
ggplot(education.df, aes(x = educs, y = mean)) +
        geom_point(size=2)+
        geom_errorbar(aes(ymax = upper, ymin = lower, width=0.3)) +
        labs(x="Years of Education", y = "Probability of Treatment",
             title = "Expected Probability of Treatment by Years of Education")
```



**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.

```r
#Simulates the coefficients a 1000 times
simCoef <- sim(lalonde.logit, n.sims=1000)

#Creates the vector for the years of education
educs <- c(3:16)

#Creates empty vectors for the average result and confidence interval obtained
#as education varies from 3 to 16 years
mean <- c()
lower <- c()
upper <- c()

#For each choice of education years
for (i in 1:length(educs)){

  #Calculates the results using the coefficients obtained from the simulation
  simLogit <- coef(simCoef)[,1] +
              coef(simCoef)[,2]*mean_age +
              coef(simCoef)[,3]*educs[i] +
              coef(simCoef)[,4]*mean_re74 +
              coef(simCoef)[,5]*mean_re75 +
              coef(simCoef)[,6]*mean_hisp
  simProb <- logit2prob(simLogit)


  #Calculates the mean and the confidence interval of the data
  lower[i] <- quantile(simProb, 0.025)
  upper[i] <- quantile(simProb, 0.975)
  mean[i] <- mean(simProb)
}

#Puts all variables relevant to the data visualization in a data frame
education.df <- data.frame(educs, mean, lower, upper)

#Uses ggplot to plot the confidence intervals
ggplot(education.df, aes(x = educs, y = mean)) +
        geom_point(size=2)+
        geom_errorbar(aes(ymax = upper, ymin = lower, width=0.3)) +
        labs(x="Years of Education", y = "Probability of Treatment",
             title = "Predicted Probability of Treatment by Years of Education")
```



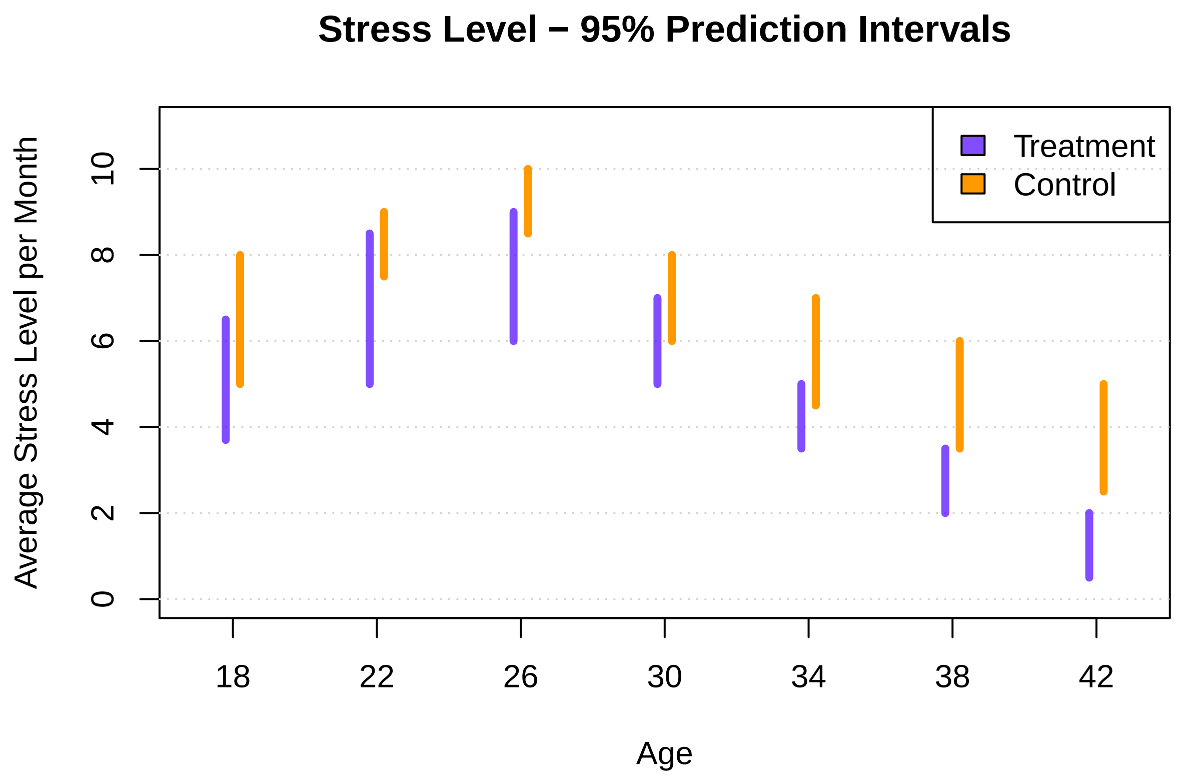
**STEP 5** Lastly, write a short paragraph with your reflections on this exercise and the results you obtained.

Similarly to Question 3, in Question 4 it was possible to see that due to being calculated using means, the expected values show a much smaller confidence interval relative to the predicted values. It also helped me deepen my understanding on what is the log to odds ratio and how to use it. The correlation between chance of receiving treatment and years of education indicates that either the treatment influences the amount of education a person gets or that there was selection bias in the treatment assignment.

## 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:

```r
# 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

# 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 study aims to assess the effect of a stress therapy program in the average stress level of subjects in a given month. The subjects were stratified by age, which was suspected to be a major confounding variable in the analysis. The results indicate that the program has a statistically significant stress reduction effect in adults with 38 and 42 years old, possibly extending to 38+ population in general. While there is indication that the therapy may be effective in younger people, the results were not statistically significant. The suggestion would be to administrate the therapy to adults in the ages where the effect was proven. To other demographics, treat it as an experimental program while developing a study with a higher sample size that may be able to prove the effect for a broader population.

## 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 uodated 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.

```r
#Imports the dataset, setting empty string as NAs to help in the next step
kickDataRaw <- read.csv("https://tinyurl.com/KaggleDataCS112", na.strings=c("", "NA"))

#Removes rows with NAs
kickDataBig <- na.omit(kickDataRaw)

#Reduces the size of the dataset by randomly sampling 1% of the entries, for
#more reasonable computing times
kickData <- kickDataBig[sample(1:nrow(kickDataBig), nrow(kickDataBig)*0.01), ]
```

**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).

```r
#Defines every row with the state "successful" as having success equal to 1,
#while all other states have success set to 0

#PS: Ideally live projects shouldn't be counted in this definition, but the
#prompt clearly asks to add them as unsuccessful

kickData$success <- as.numeric(kickData$state=="successful")
```

**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.

```
#Creates the length variable by calculating the time difference between the
#deadline and the date the project was launched, measured in days
kickData$length <- difftime(kickData$deadline,kickData$launched,units="days")

#Excludes all projects with length over 60 days
kickData <- subset(kickData, 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.

```
#Calculates the length of the dataset
total <- nrow(kickData)

#Samples 80% of the dataset's indexes to be in the training dataset
trainID <- sample(1:total, total*0.8)

#Creates a training dataset with the previously sampled indexes
kickTrain <- kickData[trainID, ]

#Creates a testing dataset with all the entries that are not in the training dataset
kickTest <- kickData[-trainID, ]
```

**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.

```
#Creates a logistic regression of the dataset
kickLogit <- glm(success ~ backers + length + as.factor(main_category) +
                   usd_goal_real, data=kickTrain, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

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

```
#Predicts the probability of success in the testing dataset based on the
#logistic regression
testProbs <- predict(kickLogit, newdata=kickTest, type="response")
```

```r
#Creates a vector for the binary predictions based on the probabilities. Sets all to 0
testPred <- rep(0, length(testProbs))

#Considers 1 as the outcome of the entries with probability over 50%
testPred[testProbs>0.5] <- 1
```

**STEP 7: How well did it do?** Report the Root Mean Squared Error (RMSE) of the predictions for the training and the test sets.

```r
#Predicts the result of the training dataset using the logistic regression and
#the same process outlined in the previous step
trainProbs <- predict(kickLogit, newdata=kickTrain, type="response")
trainPred <- rep(0, length(trainProbs))
trainPred[trainProbs>0.5] <- 1

#Calculates the RMSE of the Training Dataset
cat("The RMSE of the Training Dataset is: ", calc_rmse(kickTrain$success, trainPred), "\n")
```

```
## The RMSE of the Training Dataset is:  0.3087899
```

```r
#Calculates the RMSE of the Testing Dataset using the function created in
#Question 1 Step 3
cat("The RMSE of the Testing Dataset is: ", calc_rmse(kickTest$success, testPred))
```

```
## The RMSE of the Testing Dataset is:  0.3168707
```

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

```r
library(boot)

#Creates a cost function so that delta calculates the MSE of the LOOCV
cost <- function(r, pi) mean(abs(r-pi) > 0.5)

#Performs a Cross Validation with K equal to the length of the data,
#transforming it in a LOOCV
cv.train <- suppressWarnings(cv.glm(kickTrain, kickLogit, cost=cost, K=nrow(kickTrain)))

#Gets the Mean Squared Error from the
mse.train <- cv.train$delta[1]
rmse.train <- mse.train**(1/2)

#Prints the RMSE of the Training Dataset
cat("The RMSE of the Training Dataset is: ", rmse.train, "\n")
```

```
## The RMSE of the Training Dataset is:  0.3126124
```

```r
#The RMSE from the testing dataset is still the same
cat("The RMSE of the Testing Dataset is still: ",
    calc_rmse(kickTest$success, testPred), "\n")
```

```
## The RMSE of the Testing Dataset is still:  0.3168707
```

```r
#Calculates the difference in RMSE from LOOCV to the previous training RMSE
cat("The increase in the RMSE of the training data was: ",
    (rmse.train/calc_rmse(kickTrain$success, trainPred)-1)*100, "%" )
```

```
## The increase in the RMSE of the training data was:  1.23789 %
```

**Step 9: Explanations** Compare the RMSE 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!

The RMSE from the LOOCV method is slightly higher than in the simple method, but not by much difference. The reason is that the by not using the observation to predict itself, the LOOCV will have a slightly poorer fit to the data comparing to using the complete data (the complete data is more likely to overfit). However, given that the change is only one observation in over 2000, the difference in prediction is almost negligible. Combined, this effects point to a very slight increase in the error, 1.24% in this case.

In real-world cases, LOOCV will hardly be used. Imagine a training set of 2.000.000 entries. Making a LOOCV would take an unreasonable amount of computational power, with over 2 million regressions. It would also not be very useful because 1 entry wouldn't make a real difference in a regression line with over a million observations, so the results would be almost indistinguishable from simply predicting using the entire dataset. The solution tends to be the K-fold cross-validation, that is much less computationally intensive and provides a more meaningful cross-validation.

Furthermore, the point of cross-validation is to be able to use the entire dataset, so in a real-world application the CV would be on the full dataset rather than the sampled training data

## 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`.

```r
# 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`).

```r
# 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.