CS112 Assignment 2, Fall 2020

Stevedavies Wambui

09/27/2020

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)

```
#create a set of 1000 observations to mark the systematic
educ <- rnorm(1000, 12, 10)
salary = 1000*educ + 1 + rnorm(1000, 5, 10)
data <- data.frame(educ, salary)
head(data)
```

```
##        educ    salary
## 1 12.146405 12148.067
## 2 20.732887 20715.433
## 3  1.720538  1723.372
## 4 18.856646 18885.430
## 5 16.494370 16492.227
## 6 16.070176 16083.730
```

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?

This is a very simple model that shows the relationship between age and salary. The systematic component (explanatory) is the age which is the x variable here and the salary is the stochastic component is the error term which has a normal distribution as defined by rnorm. Age has a mean of 12 which is the average number of years someone is expected to go to school and a standard deviation of 10 which checks for people

who go for less or more than 12 years. There is also noise introduced to ensure that age does not perfectly predict salary.

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.

```
#create a linear regression model of the data and show the summary
lm_corr <- lm(salary ~ educ, data = data)
summary(lm_corr)
```

```
##
## Call:
## lm(formula = salary ~ educ, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.645  -6.585  -0.175   6.673  34.676
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 5.555e+00  4.986e-01    11.14   <2e-16 ***
## educ        1.000e+03  3.212e-02 31129.24   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.988 on 998 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 9.69e+08 on 1 and 998 DF,  p-value: < 2.2e-16
```

```
#create a new dataset containing 3/4 the education observations
educ_incorr <- 0.75*educ

#create an incorrect linear model of the systematic component and show the summary
lm_incorr <- lm(salary ~ educ_incorr, data = data)
summary(lm_incorr)
```

```
##
## Call:
## lm(formula = salary ~ educ_incorr, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.645  -6.585  -0.175   6.673  34.676
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 5.555e+00  4.986e-01    11.14   <2e-16 ***
## educ_incorr 1.333e+03  4.283e-02 31129.24   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.988 on 998 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 9.69e+08 on 1 and 998 DF,  p-value: < 2.2e-16
```

```
#run 1 simulations for the incorrect regression model
sim_incorr <- sim(lm_incorr, n.sims = 1)
#sim_incorr

#get the simulated coefficients
sim_coef <- data.frame(sim_incorr@coef)
#sim_coef

#generate predictions for the simulated results
incorr_pred <- predict(lm_incorr, newdata=sim_coef)
#incorr_pred

#calculate the rmse of the moel
incorr_rmse <- rmse(data$salary, incorr_pred)
print(incorr_rmse)
```

```
## [1] 1765764
```

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.

```
#create a linear regression model of the data and show the summary
lm_corr <- lm(salary ~ educ, data = data)
summary(lm_corr)
```

```
##
## Call:
## lm(formula = salary ~ educ, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.645  -6.585  -0.175   6.673  34.676
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 5.555e+00  4.986e-01    11.14   <2e-16 ***
## educ        1.000e+03  3.212e-02 31129.24   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.988 on 998 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 9.69e+08 on 1 and 998 DF,  p-value: < 2.2e-16
```

```r
#run 1 simulations for the correct regression model
sim_corr <- sim(lm_corr, n.sims = 1)
#sim_corr

#get the simulated coefficients
sim_coef <- data.frame(sim_corr@coef)
#sim_coef

#generate predictions for the simulated results
corr_pred <- predict(lm_corr, type='response', newdata=sim_coef)
#corr_pred

corr_rmse <- rmse(data$salary, corr_pred)
corr_rmse
```

```
## [1] 988093.2
```

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

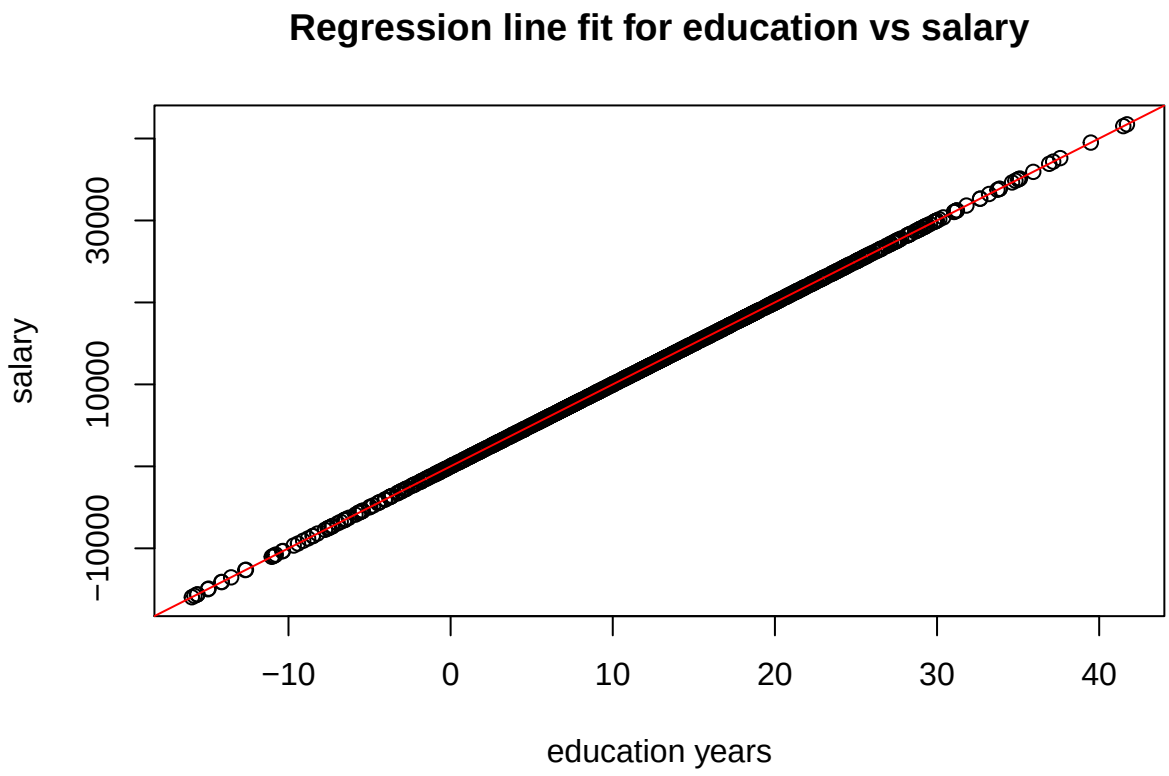
The RMSE of the incorrect model is larger because the model is quite different from the actual model and therefore the difference between true value of the response variable and the predicted value of the variable will be bigger and therefore an higher error.

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
#plot the data and label the x and y axis
plot(data, xlab = "education years", ylab = "salary",
     main = "Regression line fit for education vs salary")

#plot the regression line defined by the model
abline(lm_corr, col= 'red')
```



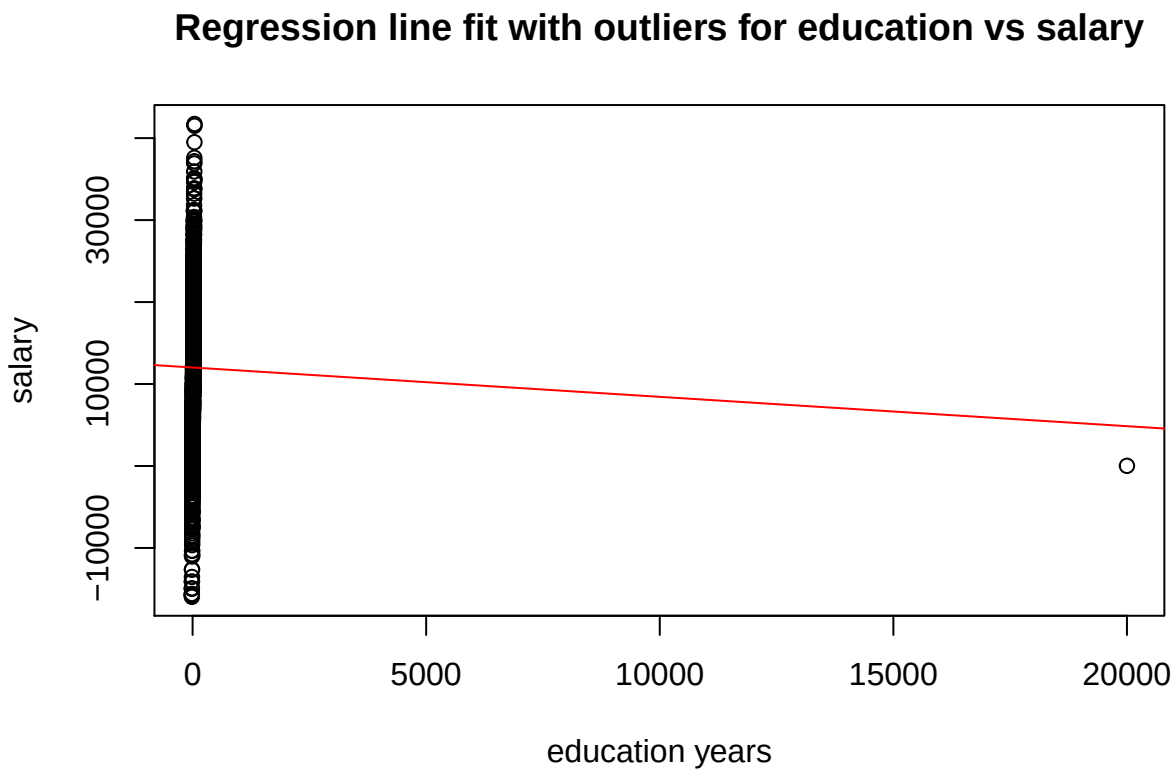
And, another figure that shows a regression line with a negative slope fit to a scatter plot of the same data **plus one additional outlier data point**. This one data point is what changes the sign of the regression line's slope from positive to negative.

```
#add an outlier to the data and create a model with the outlier
data_outl <- rbind(data, c(20000, 20))
lm_corr_outl <- lm(salary ~ educ, data = data_outl)
summary(lm_corr_outl)
```

```
##
## Call:
## lm(formula = salary ~ educ, data = data_outl)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27986.0  -6508.1     98.5   6616.3  29725.6
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12015.1749   311.4528  38.578   <2e-16 ***
## educ           -0.3579     0.4925  -0.727    0.468
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9841 on 999 degrees of freedom
## Multiple R-squared:  0.0005282,  Adjusted R-squared:  -0.0004723
```

```
## F-statistic: 0.5279 on 1 and 999 DF,  p-value: 0.4677
```

```r
#plot the data and label the x and y axis
plot(data_outl, xlab = "education years", ylab = "salary",
     main = "Regression line fit with outliers for education vs salary")

#plot the regression line defined by the model with an outlier
abline(lm_corr_outl, col= 'red')
```



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.

There are 1000 observations obtained from the initial model. I created the dataset using a normal distribution and writing out an equation that relates the response variable to the predictor variable. I created the outlier by picking a point that is very far from the other lines both in the x and y direction which in essence changed the mean and consequently the regression line.

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
#load the lalonde dataset and show first 5 rows
data(lalonde)
head(lalonde)
```

```
##   age educ black hisp married nodegr re74 re75     re78 u74 u75 treat
## 1  37   11     1    0       1      1    0    0  9930.05   1   1     1
## 2  22    9     0    1       0      1    0    0  3595.89   1   1     1
## 3  30   12     1    0       0      0    0    0 24909.50   1   1     1
## 4  27   11     1    0       0      1    0    0  7506.15   1   1     1
## 5  33    8     1    0       0      1    0    0   289.79   1   1     1
## 6  22    9     1    0       0      1    0    0  4056.49   1   1     1
```

```
#create a linear regression model on lalonde dataset
lm_lal <- lm(re78 ~age + educ + re74 + re75 + hisp + black, data = lalonde)
summary(lm_lal)
```

```
##
## Call:
## lm(formula = re78 ~ age + educ + re74 + re75 + hisp + black,
##     data = lalonde)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -9065  -4577  -1775   3186  55037
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.126e+03  2.434e+03   0.463   0.6437
## age          5.928e+01  4.409e+01   1.345   0.1794
## educ         4.293e+02  1.758e+02   2.442   0.0150 *
## re74         7.462e-02  7.699e-02   0.969   0.3330
## re75         6.676e-02  1.314e-01   0.508   0.6116
## hisp        -2.125e+02  1.546e+03  -0.137   0.8907
## black       -2.323e+03  1.162e+03  -1.999   0.0462 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6544 on 438 degrees of freedom
## Multiple R-squared:  0.03942,    Adjusted R-squared:  0.02627
## F-statistic: 2.996 on 6 and 438 DF,  p-value: 0.007035
```

STEP 2 Report coefficients and R-squared.

```
#get the coefficients of the linear model
lmlal_coef <- lm_lal$coef
message("The coefficients of linear model:",lmlal_coef)
```

```
## The coefficients of linear model:1126.4915185927259.2815547244202429.3073957317810.07461871892699310
```

```
#get the r_squared, convert to percentage and round to 2 decimal places
lmlal_rsq <- round(((summary(lm_lal)$r.squared)*100), digits = 2)
message("R-squared is: ", lmlal_rsq, "%")
```

```
## R-squared is: 3.94%
```

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.

The formula for r_squared is as below: R^2 = 1 -(rss/tss) where rss is the residual sum of squares ie the sum of the squared difference between the actual values and the predicted values and rss is the total sum of squares that is the sum of the squared difference between the actual values and the mean of the actual value of the outcome variable.

```
#to calculate the r_squared, get the total sum of squares and the residual sum of squares
#obtain predictions for the outcome variable
preds <- predict(lm_lal, lalonde)

#get the residual sum of squares that is difference between actual and predicted
rss <- sum((preds - lalonde$re78)^2)

#get the total sum of squares that is differnece between actual and mean of actual
tss <- sum((lalonde$re78 - mean(lalonde$re78))^2)

#get the r_squared, convert to percentage and round to 2 decimal places
r_squared <- 1 - (rss/tss)
message("R-squared is: ", round((r_squared* 100),2), "%")
```

```
## R-squared is: 3.94%
```

From this calculation, the value of r_squared is 3.94% just like the value obtained from the summary of the linear regression model.

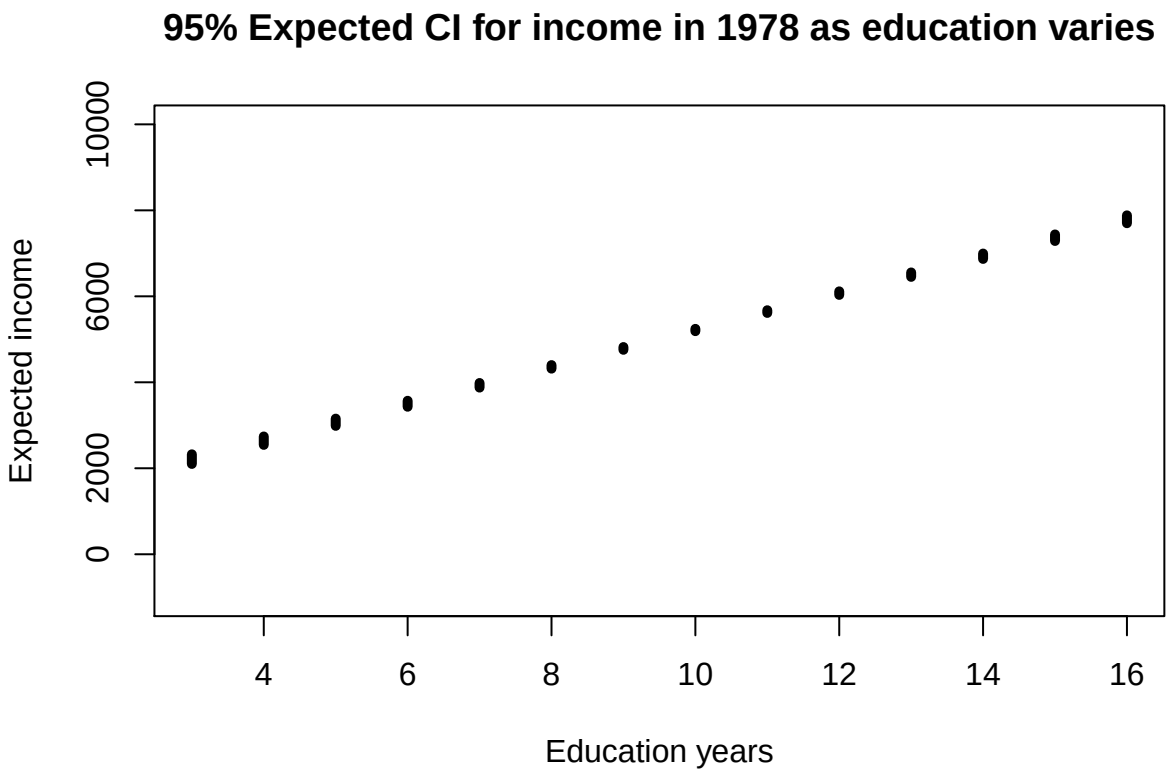
STEP 3 Then, setting all the predictors at their means EXCEPT `education`, create a data visualization that shows the 95% confidence interval of the expected values of `re78` as `education` varies from 3 to 16. Be sure to include axes labels and figure titles.

```
#compute the means of the predictors
age_mean <- mean(lalonde$age)
re74_mean <- mean(lalonde$re74)
re75_mean <- mean(lalonde$re75)
hisp_mean <- mean(lalonde$hisp)
black_mean <- mean(lalonde$black)
```

```
#create a matrix initially containing NAs to store the means
#of the predicted values which is the expected value
means <- matrix(NA, nrow = 500, ncol = 14)

#using simulation, loop through 500 times and create 500
#simulations of the model and then for each person, get the predicted
#values 500 times and store these values and then get the means of these
#predicted values 500 times
for (j in 1:500){
  #run simulations on the model
  sim_lal <- sim(lm_lal, n.sims = 500)

  #create a storage matrix to store the simulated values that
```

```
  #initially contains NAs with 14 columns for educ and 1000 rows
  storage_matrix_educ <- matrix(NA, nrow = 500, ncol = 14)

  #for each value for education years, generate predicted values
  for (educ in (3:16)) {
    for (i in 1:500){
      #the predictors set at their means, 1 is for the intercept and
      #then multiply with the coefficeints
      preds <- c(1, age_mean, educ, re74_mean, re75_mean, hisp_mean, black_mean)
      storage_matrix_educ[i, educ + 1 - min(lalonde$educ)] <- sum(sim_lal@coef[i,]*preds)
    }
  }
  #get the means of the predicted values 500 times for each person 1 to 14
  for (educ in 1:14){
    means[j, educ] <- mean(storage_matrix_educ[,educ])
  }
}
```

```
#compute the confidence intervals for the expected values
expec_confint <- apply(means, 2, quantile, probs = c(0.025, 0.975))

#create a dataframe that stores the confidence interval for education from 3 to 16
mean_df <- data.frame( "Educations"= 3:16,
                       "Lower bound interval" = expec_confint[1, ],
                       "Upper bound interval" = expec_confint[2, ])

#plot the 95% confidence interval of expected values for the probability of receiving treatment
plot(x = c(1:1000), y = c(1:1000), type = "n", xlim = c(3,16), ylim = c(-1000,10000),
     main = "95% Expected CI for income in 1978 as education varies",
     xlab = "Education years",
     ylab = "Expected income")

for (educ in 3:16) {
  segments(
    x0 = educ,
    y0 = mean_df[educ- min(lalonde$educ)+1, 2],
    x1 = educ,
    y1 = mean_df[educ- min(lalonde$educ)+1, 3],
    lwd = 5)
}
```

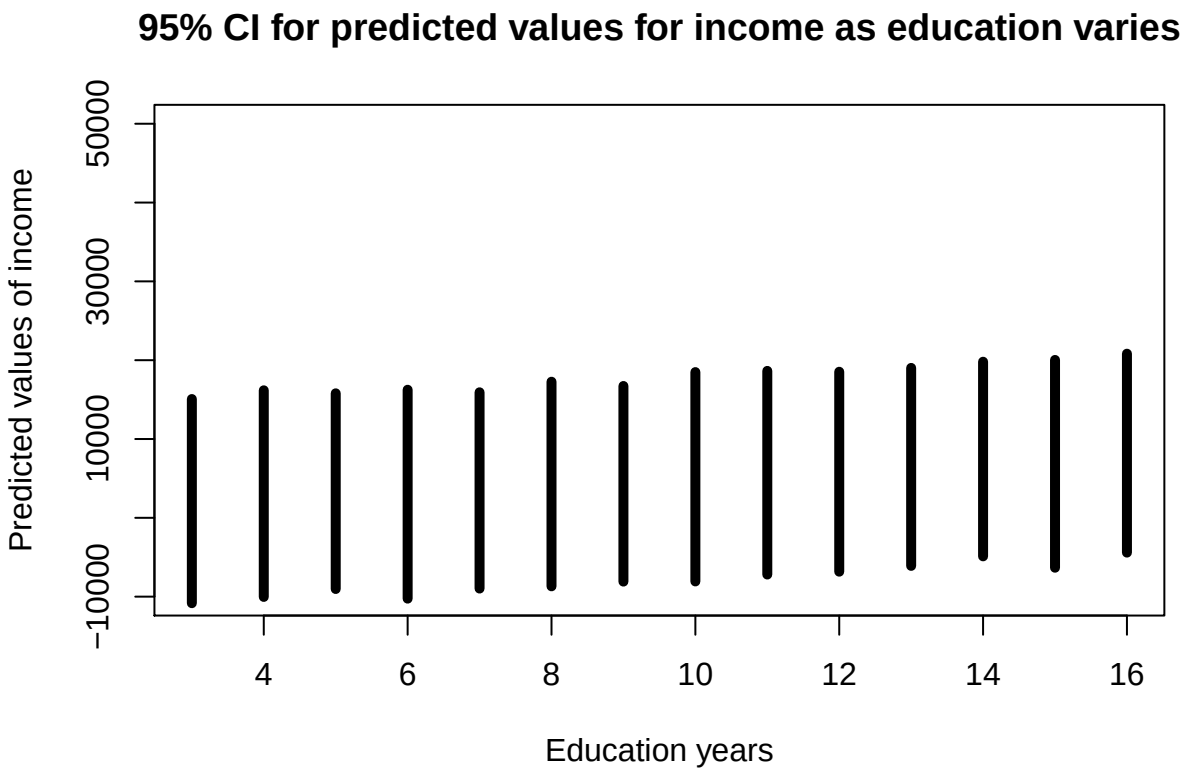


STEP 4 Then, do the same thing, but this time for the predicted values of `re78`. Be sure to include axes labels and figure titles.

```
#create a storage matrix to store the simulated values that initially contains
#NAs with 14 columns for educ and 1000 rows
storage_matrix_educ1 <- matrix(NA, nrow = 1000, ncol = 14)

#for the predicted values, add sigma as an error term in the simulations
#loop through education from 3 to 16 and loop through for 1000 simulations,
#and get the predicted values and append to the storage matrix
#run simulations for the model
sim_lal1 <- sim(lm_lal, n.sims = 1000)
for (educ in 3:16) {
  for (i in 1:1000) {
    preds <- c(1, age_mean, educ, re74_mean, re75_mean, hisp_mean, black_mean)
    storage_matrix_educ1[i, educ + 1 - min(lalonde$educ)] <- sum(preds *sim_lal1@coef[i,])+
    rnorm(1, 0, sim_lal1@sigma[i])
  }
}
```

```
#compute the confidence interval for expected values of re78 from the storage matrix
predict_confint <- apply(storage_matrix_educ1, MARGIN = 2, quantile,
                         probs = c(0.025, 0.975))

#store the education, means calculated above and the lower and upper
```

```r
#bounds of the confidence interval in a dataframe
mean_df1 <- data.frame( "Educations"= 3:16,
                        "Lower bound interval" = predict_confint[1, ],
                        "Upper bound interval" = predict_confint[2, ])

#plot the 95% confidence interval of predicted values for the probability of receiving treatment
plot(x = c(1:10000), y = c(1:10000), type = "n", xlim = c(3,16), ylim = c(-10000, 50000),
     main = "95% CI for predicted values for income as education varies",
     xlab = "Education years",
     ylab = "Predicted values of income")

for (educ in 3:16) {
  segments(
    x0 = educ,
    y0 = mean_df1[educ - min(lalonde$educ)+1, 2],
    x1 = educ,
    y1 = mean_df1[educ - min(lalonde$educ)+1, 3],
    lwd = 5)
}
```



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 length of the intervals of the expected values was much shorter than that of the predicted intervals. This is because the expected values are gotten by averaging the results of the predicted values for the given

number of iterations and therefore the bounds are much tighter. Additionally, the expected values do not include an error term and therefore represent the true model more accurately. For the predicted values, the error term introduces higher variance in the dataset which in turn means that the confidence interval will be much larger.

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`.

```
#run logistic regression on the data set and output a summary
log_lalonde <- glm(treat~age + educ + hisp + re74 + re75, data = lalonde)
summary(log_lalonde)
```

```
##
## Call:
## glm(formula = treat ~ age + educ + hisp + re74 + re75, data = lalonde)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5618  -0.4262  -0.3656   0.5607   0.7686
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.840e-01  1.608e-01   1.144    0.253
## age          2.816e-03  3.314e-03   0.850    0.396
## educ         1.629e-02  1.318e-02   1.236    0.217
## hisp        -1.336e-01  8.405e-02  -1.590    0.113
## re74        -5.235e-06  5.790e-06  -0.904    0.366
## re75         1.226e-05  9.874e-06   1.241    0.215
##
## (Dispersion parameter for gaussian family taken to be 0.2422554)
##
##     Null deviance: 108.09  on 444  degrees of freedom
## Residual deviance: 106.35  on 439  degrees of freedom
## AIC: 639.91
##
## Number of Fisher Scoring iterations: 2
```

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

```
#get the coefficients of the variables
coef(log_lalonde)
```

```
##   (Intercept)           age          educ          hisp          re74
##  1.840208e-01  2.815643e-03  1.629254e-02 -1.336066e-01 -5.234981e-06
##          re75
##  1.225615e-05
```

```r
#compute the confidence intervals for the variables
confint(log_lalonde)
```

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

```
##                      2.5 %       97.5 %
## (Intercept) -1.311371e-01 4.991787e-01
## age         -3.678939e-03 9.310225e-03
## educ        -9.546484e-03 4.213157e-02
## hisp        -2.983413e-01 3.112803e-02
## re74        -1.658372e-05 6.113757e-06
## re75        -7.096908e-06 3.160921e-05
```

The coefficient of age means that each one-unit increase in age increases the log odds of treatment status being 1 by 0.00282 and the coefficient of education means each unit increase in education increases the log odds of treatment status being 1 by 0.0163. The confidence interval of education means that if this model is simulated multiple times, the coefficient of education will between -0.00955 and 0.04213 95% of the time. The coefficient of age means that if this is simulated multiple times, the coefficient of age will fall between -0.00368 and 0.00931 95% of the time.

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
#function to compute the bootstrap on the data
boot_funct <- function(data){
  bootstrap <- sample(1:nrow(data), size=nrow(data), replace = TRUE)
  return(bootstrap)
}

#get the estimates for age and education in 1000 simulations on the
#regression model using the bootsrap function
age_store <- rep(1:1000)
educ_store <- rep(1:1000)
for (i in (1:1000)){
  booted <- lalonde[boot_funct(lalonde),]
  booted_glm <- glm(treat ~ age + educ + hisp + re74 + re75, data = booted)
  age_store[i] <- booted_glm$coefficients[2]
  educ_store[i] <-booted_glm$coefficients[3]
}

#calculate the confidence intervals from the estimates obtained in the 1000 iterations
age_confint <- quantile(age_store, probs=c(0.025, 0.975))
educ_confint <- quantile(educ_store, probs=c(0.025, 0.975))
educ_confint
```

```
##       2.5%      97.5%
## -0.01044704  0.04312333
```

```r
age_confint
```

```
##         2.5%        97.5%
## -0.002989852  0.009543708
```

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

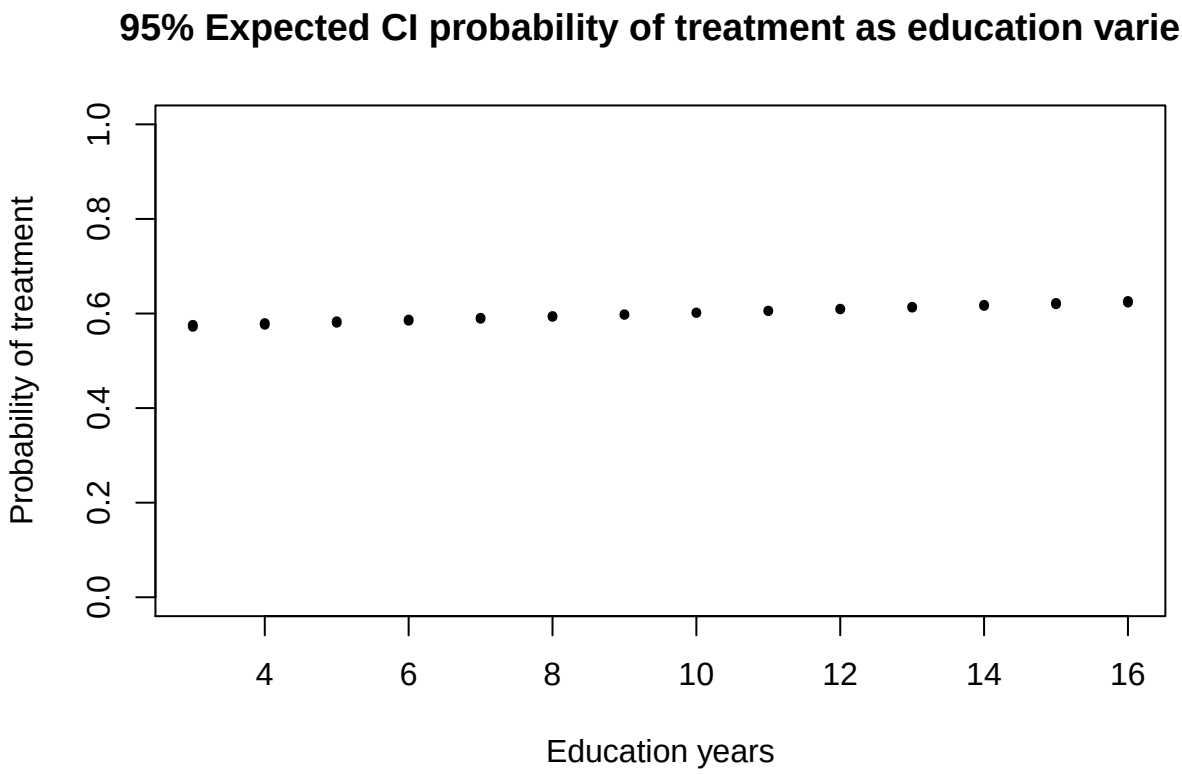
The bootstrapped confidence interval for age is [-0.003222, 0.0091896] and for education is [-0.01039, 0.04342]. These bootstrapped confidence intervals are very close the actual confidence interval from the regression in step 1.

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.

```
#create a matrix initially containing NAs to store the means of the
#predicted values which is the expected value
log_means <- matrix(NA, nrow = 500, ncol = 14)

#using simulation, loop through 500 times and create 500 simulations
#of the model and then for each person, get the predicted values 500 times
#and store these values and then get the means of these predicted values 500 times
for (j in 1:500){
  #run simulations on the model
  sim_log <- sim(log_lalonde, n.sims = 500)

  #create a storage matrix to store the simulated values that initially
  #contains NAs with 14 columns for educ and 1000 rows
  storage_matrix_log <- matrix(NA, nrow = 500, ncol = 14)

  #for each value for education years, generate predicted values
  for (educ in (3:16)) {
    for (i in 1:500){
      #the predictors set at their means, 1 is for the intercept and then
      #multiply with the coefficeints
      preds <- c(1, age_mean, educ, hisp_mean, re74_mean, re75_mean)
      log1 = sum(sim_log@coef[i,]*preds)
      storage_matrix_log[i, educ - min(lalonde$educ)+1] <- exp(log1)/(1+exp(log1))
    }
  }
  #get the means of the predicted values 500 times for each person 1 to 14
  for (educ in 1:14){
    log_means[j, educ] <- mean(storage_matrix_log[,educ])
  }
}
```

```
#compute the confidence intervals for the expected values
expec_confint_log <- apply(log_means, 2, quantile, probs = c(0.025, 0.975))

#create a dataframe that stores the confidence interval for education from 3 to 16
mean_df_log <- data.frame( "Educations"= 3:16,
                           "Lower bound interval" = expec_confint_log[1, ],
                           "Upper bound interval" = expec_confint_log[2, ])

#plot the 95% confidence interval of expected values for the probability of receiving treatment
plot(x = c(1:1000), y = c(1:1000), type = "n", xlim = c(3,16), ylim = c(0,1),
     main = "95% Expected CI probability of treatment as education varies",
     xlab = "Education years",
```

```
     ylab = "Probability of treatment")

for (educ in 3:16) {
  segments(
    x0 = educ,
    y0 = mean_df_log[educ- min(lalonde$educ)+1, 2],
    x1 = educ,
    y1 = mean_df_log[educ- min(lalonde$educ)+1, 3],
    lwd = 5)
}
```

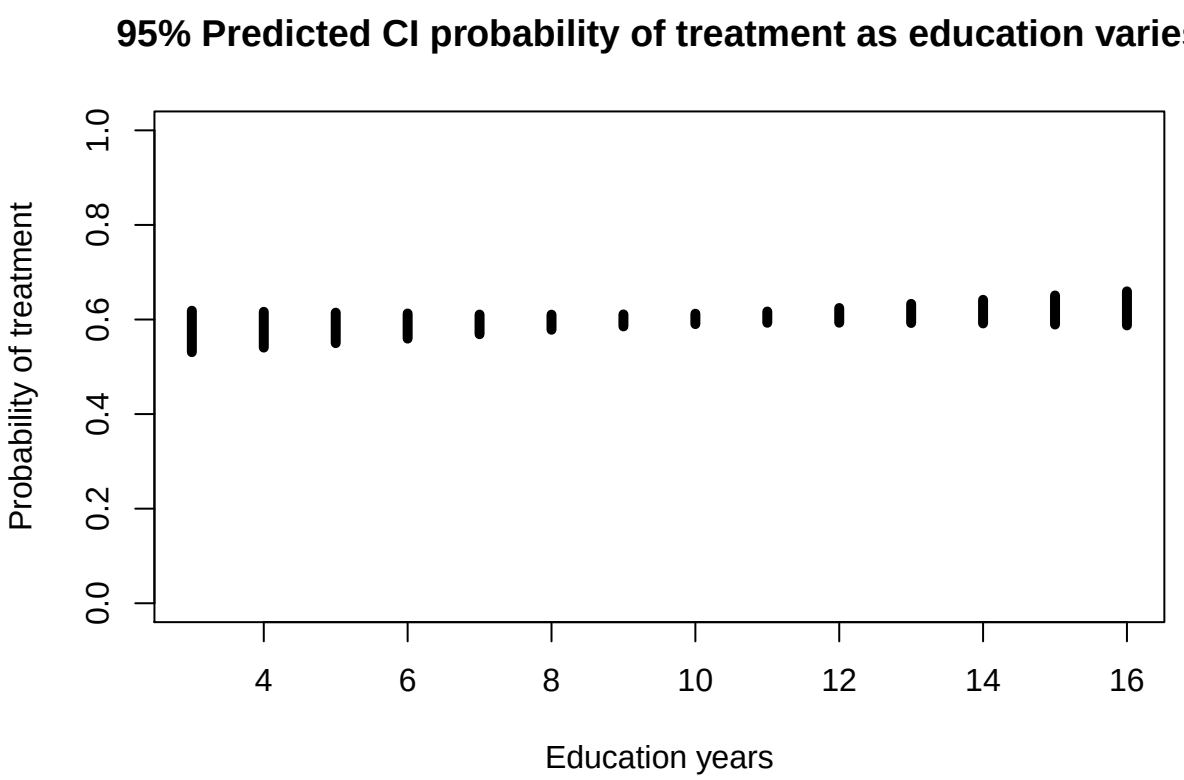


STEP 4 Then, do the same thing, but this time for the predicted values of the probability of receiving treatment as education varies from 3 to 16. Be sure to include axes labels and figure titles.

```
#create a storage matrix to store the simulated values that initially contains
#NAs with 14 columns for educ and 1000 rows
storage_matrix_log1 <- matrix(NA, nrow = 500, ncol = 14)

#using simulation, create 500 simulations of the model and then for each person,
#get the predicted values 500 times and store these values
#run simulations on the model
sim_log1 <- sim(log_lalonde, n.sims = 500)
for (educ in (3:16)) {
  for (i in 1:500){
```

```
    #the predictors set at their means, 1 is for the intercept and then multiply with the coefficeints
    preds <- c(1, age_mean, educ, hisp_mean, re74_mean, re75_mean)
    log2 = sum(sim_log1@coef[i,]*preds)
    storage_matrix_log1[i, educ - min(lalonde$educ)+1] <- exp(log2)/(1+exp(log2))
  }
}
```

```
#compute the confidence interval for expected values of re78 from the storage matrix
predict_confint_log <- apply(storage_matrix_log1, MARGIN = 2, quantile,
                             probs = c(0.025, 0.975))

#store the education, means calculated above and the lower and upper
#bounds of the confidence interval in a dataframe
mean_df_log1 <- data.frame( "Educations"= 3:16,
                            "Lower bound interval" = predict_confint_log[1, ],
                            "Upper bound interval" = predict_confint_log[2, ])

#plot the 95% confidence interval of predicted values for the probability of receiving treatment
plot(x = c(1:10000), y = c(1:10000), type = "n", xlim = c(3,16), ylim = c(0,1),
     main = "95% Predicted CI probability of treatment as education varies",
     xlab = "Education years",
     ylab = "Probability of treatment")
for (educ in 3:16) {
  segments(
    x0 = educ,
    y0 = mean_df_log1[educ-2, 2],
    x1 = educ,
    y1 = mean_df_log1[educ-2, 3],
    lwd = 5)
}
```



STEP 5

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

This exercise helped establish the difference between predicted values and expected values. The confidence intervals for the predicted values are much larger than those of the expected values because expected values are obtained by taking the means of the predicted values for given number of iterations which means that the bounds of the expected values will be tighter.

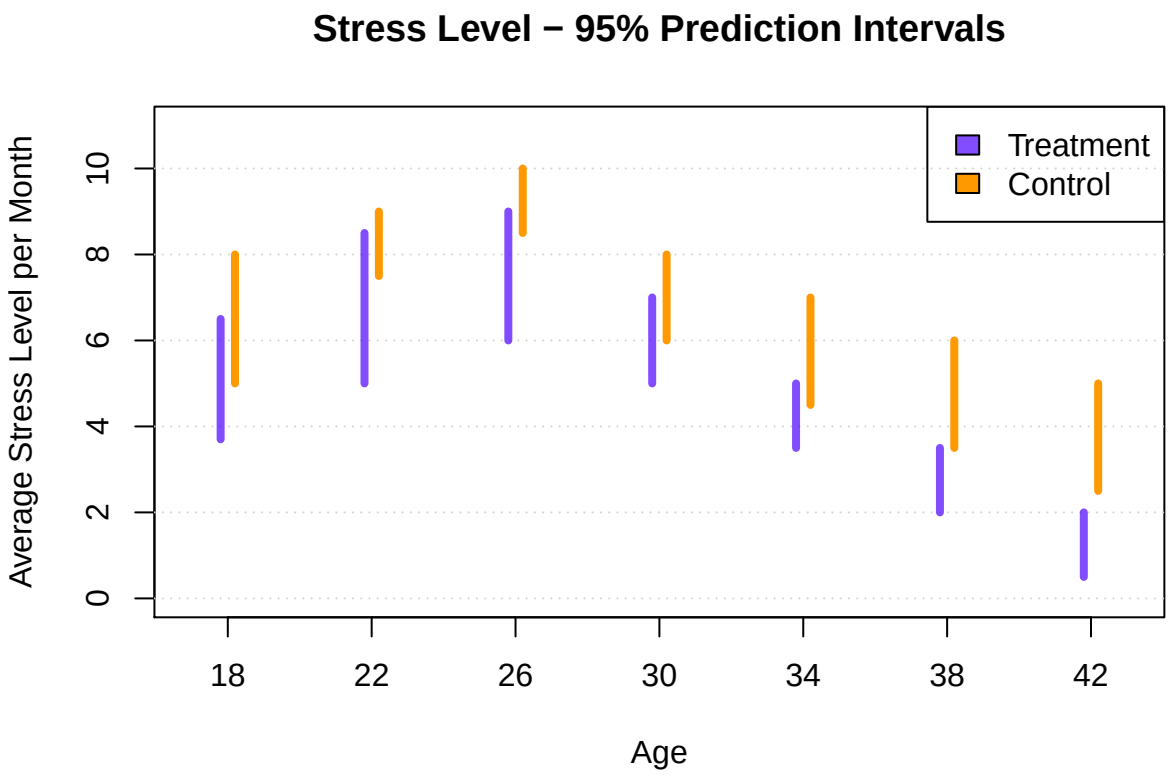
QUESTION 5

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

# 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).

Write your executive summary here.

This study focuses on determining the impact of a stress therapy program for individuals between ages 18 and 42. The stress therapy program is intended to reduce the stress level of the individuals. To determine the effect, the individuals are randomly split into treatment and control groups through randomized control trials where the treatment group undergoes the therapy program and the control group does not. The results obtained are only for one month with an inherent assumption of no attrition. The results tend to show that on average, the people who undergo the therapy program have lower stress levels per month and on average, people at 42 have the lowest stress levels while those at 26 have the highest. The policy implications of these results is that the therapy program reduces the stress levels and therefore people should attend it. For these implications to hold, a number of factors need to be controlled for: are the people in control attending alternative programs? are all the people subject to the same environment such that stress levels are determined by the same factors? do the people have different stress levels for the same environment?

The advantage of RCT is that randomization ensures a fairly even distribution of such factors to ensure the people in both treatment and control have similar factors.

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

```
#import the dataset
kickst_data <- read.csv(file="https://tinyurl.com/KaggleDataCS112")

#print out the first 5 rows
head(kickst_data)
```

```
##           ID                                                 name
## 1 1000002330                      The Songs of Adelaide & Abullah
## 2 1000003930          Greeting From Earth: ZGAC Arts Capsule For ET
## 3 1000004038                                       Where is Hank?
## 4 1000007540       ToshiCapital Rekordz Needs Help to Complete Album
```

```
## 5 1000011046 Community Film Project: The Art of Neighborhood Filmmaking
## 6 1000014025                                        Monarch Espresso Bar
##         category main_category currency   deadline  goal            launched
## 1         Poetry    Publishing      GBP 2015-10-09  1000 2015-08-11 12:12:28
## 2 Narrative Film  Film & Video      USD 2017-11-01 30000 2017-09-02 04:43:57
## 3 Narrative Film  Film & Video      USD 2013-02-26 45000 2013-01-12 00:20:50
## 4          Music         Music      USD 2012-04-16  5000 2012-03-17 03:24:11
## 5   Film & Video  Film & Video      USD 2015-08-29 19500 2015-07-04 08:35:03
## 6    Restaurants          Food      USD 2016-04-01 50000 2016-02-26 13:38:27
##   pledged      state backers country usd.pledged usd_pledged_real usd_goal_real
## 1       0     failed       0      GB           0                0       1533.95
## 2    2421     failed      15      US         100             2421      30000.00
## 3     220     failed       3      US         220              220      45000.00
## 4       1     failed       1      US           1                1       5000.00
## 5    1283   canceled      14      US        1283             1283      19500.00
## 6   52375 successful     224      US       52375            52375      50000.00
```

Remove any rows that include missing values.

```
#Check if there are any null/empty values and print the total in each column
colSums(is.na(kickst_data))
```

```
##               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
##             3797                0                0
```

```
#remove the empty/null values
kickst_data <- na.omit(kickst_data)

#confirm that null values are omitted
colSums(is.na(kickst_data))
```

```
##               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).

```
#codify the outcome variable state
success <- ifelse(kickst_data$state== 'successful', 1, 0)
success[1:5]
```

```
## [1] 0 0 0 0 0
```

```
#add the new column to the dataframe and ptint out first  6 rows
kickst_data1 <- cbind(kickst_data, success)
head(kickst_data1)
```

```
##           ID                                                      name
## 1 1000002330                           The Songs of Adelaide & Abullah
## 2 1000003930              Greeting From Earth: ZGAC Arts Capsule For ET
## 3 1000004038                                             Where is Hank?
## 4 1000007540          ToshiCapital Rekordz Needs Help to Complete Album
## 5 1000011046 Community Film Project: The Art of Neighborhood Filmmaking
## 6 1000014025                                       Monarch Espresso Bar
##         category main_category currency   deadline  goal            launched
## 1         Poetry    Publishing      GBP 2015-10-09  1000 2015-08-11 12:12:28
## 2 Narrative Film  Film & Video      USD 2017-11-01 30000 2017-09-02 04:43:57
## 3 Narrative Film  Film & Video      USD 2013-02-26 45000 2013-01-12 00:20:50
## 4          Music         Music      USD 2012-04-16  5000 2012-03-17 03:24:11
## 5   Film & Video  Film & Video      USD 2015-08-29 19500 2015-07-04 08:35:03
## 6    Restaurants          Food      USD 2016-04-01 50000 2016-02-26 13:38:27
##   pledged      state backers country usd.pledged usd_pledged_real usd_goal_real
## 1       0     failed       0      GB           0                0       1533.95
## 2    2421     failed      15      US         100             2421      30000.00
## 3     220     failed       3      US         220              220      45000.00
## 4       1     failed       1      US           1                1       5000.00
## 5    1283   canceled      14      US        1283             1283      19500.00
## 6   52375 successful     224      US       52375            52375      50000.00
##   success
## 1       0
## 2       0
## 3       0
## 4       0
## 5       0
## 6       1
```

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.

```
#to work with dates, convert the columns to type date
kickst_data1$deadline <- as.Date(kickst_data1$deadline)
kickst_data1$launched <- as.Date(kickst_data1$launched)
head(kickst_data1)
```

```
##           ID                                                      name
## 1 1000002330                           The Songs of Adelaide & Abullah
## 2 1000003930              Greeting From Earth: ZGAC Arts Capsule For ET
## 3 1000004038                                             Where is Hank?
```

```
## 4 1000007540          ToshiCapital Rekordz Needs Help to Complete Album
## 5 1000011046 Community Film Project: The Art of Neighborhood Filmmaking
## 6 1000014025                                        Monarch Espresso Bar
##         category main_category currency   deadline  goal   launched pledged
## 1         Poetry    Publishing      GBP 2015-10-09  1000 2015-08-11       0
## 2 Narrative Film  Film & Video      USD 2017-11-01 30000 2017-09-02    2421
## 3 Narrative Film  Film & Video      USD 2013-02-26 45000 2013-01-12     220
## 4          Music         Music      USD 2012-04-16  5000 2012-03-17       1
## 5   Film & Video  Film & Video      USD 2015-08-29 19500 2015-07-04    1283
## 6    Restaurants          Food      USD 2016-04-01 50000 2016-02-26   52375
##        state backers country usd.pledged usd_pledged_real usd_goal_real success
## 1     failed       0      GB           0               0       1533.95       0
## 2     failed      15      US         100            2421      30000.00       0
## 3     failed       3      US         220             220      45000.00       0
## 4     failed       1      US           1               1       5000.00       0
## 5   canceled      14      US        1283            1283      19500.00       0
## 6 successful     224      US       52375           52375      50000.00       1
```

```
#get the length of the projects by subtracting the launch
#and deadline dates, and add it to the dataframe
length <- kickst_data1 %>%
  transmute(length = deadline - launched)
kickst_data3 <- cbind(kickst_data1, length)

#drop any missing values
kickst_data2 <- na.omit(kickst_data3)

#remove any rows whose duration is more than 60 days
kickst_data2 <- subset(kickst_data2, length<= 60)

head(kickst_data2)
```

```
##           ID                                                    name
## 1 1000002330                        The Songs of Adelaide & Abullah
## 2 1000003930             Greeting From Earth: ZGAC Arts Capsule For ET
## 3 1000004038                                          Where is Hank?
## 4 1000007540          ToshiCapital Rekordz Needs Help to Complete Album
## 5 1000011046 Community Film Project: The Art of Neighborhood Filmmaking
## 6 1000014025                                        Monarch Espresso Bar
##         category main_category currency   deadline  goal   launched pledged
## 1         Poetry    Publishing      GBP 2015-10-09  1000 2015-08-11       0
## 2 Narrative Film  Film & Video      USD 2017-11-01 30000 2017-09-02    2421
## 3 Narrative Film  Film & Video      USD 2013-02-26 45000 2013-01-12     220
## 4          Music         Music      USD 2012-04-16  5000 2012-03-17       1
## 5   Film & Video  Film & Video      USD 2015-08-29 19500 2015-07-04    1283
## 6    Restaurants          Food      USD 2016-04-01 50000 2016-02-26   52375
##        state backers country usd.pledged usd_pledged_real usd_goal_real success
## 1     failed       0      GB           0               0       1533.95       0
## 2     failed      15      US         100            2421      30000.00       0
## 3     failed       3      US         220             220      45000.00       0
## 4     failed       1      US           1               1       5000.00       0
## 5   canceled      14      US        1283            1283      19500.00       0
## 6 successful     224      US       52375           52375      50000.00       1
##    length
```

```
## 1 59 days
## 2 60 days
## 3 45 days
## 4 30 days
## 5 56 days
## 6 35 days
```

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.

```
#create the indices for the data with 80% as training
train_indices = sort(sample(nrow(kickst_data2), nrow(kickst_data2)*.8))

#create a training set
kickst_train <- kickst_data2[train_indices,]

#create a testing set ie the remaining data
kickst_test <- kickst_data2[-train_indices,]
```

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.

```
#convert main category to factor
kickst_data2$main_category <- as.factor(kickst_data2$main_category)
```

```
#fit a logistic regression model to the data
logistic1 <- glm(success ~ backers + length + main_category
                 + goal, data = kickst_train, family=binomial)
```

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

```
logistic1
```

```
##
## Call:  glm(formula = success ~ backers + length + main_category + goal,
##     family = binomial, data = kickst_train)
##
## Coefficients:
##               (Intercept)                    backers
##                -0.0921228                  0.0314960
##                    length        main_categoryComics
##                -0.0188657                 -0.2430604
##       main_categoryCrafts         main_categoryDance
##                -0.6786088                  0.8024240
##       main_categoryDesign       main_categoryFashion
```

```
##                -0.7538177                -0.7000692
## main_categoryFilm & Video          main_categoryFood
##                 0.1251962                -0.5254199
##        main_categoryGames    main_categoryJournalism
##                -1.4588948                -0.6898435
##        main_categoryMusic   main_categoryPhotography
##                 0.2207969                -0.4030949
##   main_categoryPublishing    main_categoryTechnology
##                -0.4863364                -0.9179029
##      main_categoryTheater                      goal
##                 0.7657853                -0.0001146
##
## Degrees of Freedom: 295511 Total (i.e. Null);  295494 Residual
## Null Deviance:       385400
## Residual Deviance: 215000    AIC: 215000
```

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

```
#make a prediction on the logistic regression for the test set
kickst_pred_test <- predict(logistic1, kickst_test)

#make a prediction for training set
kickst_pred_train <- predict(logistic1, kickst_train)

#convert the predicted values to 0 if they are less than 0.5 and 1 if greater than 5
kickst_pred_tebinary <- ifelse(kickst_pred_test < 0.5, 0, 1)

#convert the predicted values to 0 if they are less than 0.5 and 1 if greater than 5
kickst_pred_trbinary <- ifelse(kickst_pred_train < 0.5, 0, 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.

As mentioned in Perusall, since the outcome is binary and the model is logistic, we cannot use the rmse to check the performance of a logistic regression. Instead, we check the misclassification rate that is which of predictions were not correctly classified.

```
#compute the classification table of predictions on training set
kickst_tr_table <- table(kickst_pred_trbinary, kickst_train$success)

#compute the classification accuracy of the model on the predictions of training set
train_accuracy <- sum(diag(kickst_tr_table))/sum(kickst_tr_table)

#compute the misclassification rate for the training set
tr_misclass_rate <- round(((1 - train_accuracy) *100), 2)
message("The misclassifcation error rate on the training set of the data is ",
        tr_misclass_rate, "%")
```

```
## The misclassifcation error rate on the training set of the data is 15.84%
```

```r
#compute the classification table of predictions on test set
kickst_te_table <- table(kickst_pred_tebinary, kickst_test$success)

#compute the classification accuracy of the model on the predictions of test set
test_accuracy <- sum(diag(kickst_te_table))/sum(kickst_te_table)

#compute the misclassification rate for the test set
te_misclass_rate <- round(((1 - test_accuracy) *100), 2)
message("The misclassifcation error rate on the test set of the data is ",
        te_misclass_rate, "%")
```

```
## The misclassifcation error rate on the test set of the data is 15.72%
```

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
#sample 5% of the data for LOOCV
sample_kickst <-sort(sample(nrow(kickst_data2), nrow(kickst_data2)*.05))
sample_data <- kickst_data2[sample_kickst,]

#sample 5% of the sample to perform looocv
sample_kickst1 <-sort(sample(nrow(sample_data), nrow(sample_data)*.05))
looocv_data <- sample_data[sample_kickst1,]
dim(looocv_data)
```

```
## [1] 923  17
```

```r
logistic2 <- glm(success ~ backers + length + main_category + goal,
                 data = looocv_data, family=binomial)

#LOOCV on the training set:
kickst_cv_err <- cv.glm(looocv_data, logistic2)
```

```r
#make a prediction on the logistic regression for the cross_validated data
kickst_pred_looocv <- predict(logistic2, looocv_data)

#convert the predicted values to 0 if they are less than 0.5 and 1 if greater than 5
kickst_pred_lobinary <- ifelse(kickst_pred_looocv < 0.5, 0, 1)

#compute the classification table of predictions on looocv data set
kickst_lo_table <- table(kickst_pred_lobinary, looocv_data$success)

#compute the classification accuracy of the model on the predictions of loocv data set
looocv_accuracy <- sum(diag(kickst_lo_table))/sum(kickst_lo_table)

#compute the misclassification rate for the training set
lo_misclass_rate <- round(((1 - looocv_accuracy) *100), 2)
message("The misclassifcation error rate on the cross-validated set of the data is ",
        lo_misclass_rate, "%")
```

```
## The misclassifcation error rate on the cross-validated set of the data is 11.81%
```

```r
message("The mse is :", kickst_cv_err$delta[1])
```

```
## The mse is :0.0899564160856122
```

Step 9: Explanations Compare the RMSE from the simple method to the LOOCV method?

The misclassification rate from the simple method is 15.76% while that from the LOOCV is 11.92%. It makes sense that the prediction error from LOOCV is lower than from a simple method because looocv uses each single observation as a test while n -1 observations are used as training and then averages all the MSEs which in turn reduces the bias and consequently, the prediction error. The tradeoff however is there is high variance since using just one observation as testing means a lot of similarity between the test and training set.

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!

In practice, data scientists use k-fold cross-validation over LOOCV for several reasons: 1. They work with large datasets and so it is computationally expensive to perform LOOCV. 2. They need to ensure a balance between bias and variance and k-fold gives the flexibility of determining how many subsets of the data will be used for training while the remaining is used for testing. Therefore, the approach in this project differs from the real-world in that in the real-world, data scientists want to use a method that is inexpensive and takes a short time. For example, if you were working with 30 million rows in a dataset as in lending club data, LOOCV will run way too many times and thus too long while k-fold cross-validation will take a much shorter time and ensure low variance and bearable bias.

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.