Homework 1 - EEB590C

Devin Molnau, Holly Loper, Elizabeth McMurchie

February 18, 2021

EEB590C -Homework 1

Homework 1: Lectures 1-4

This assignment is due prior to class in week 6. You are to self-select and work in groups: 2-3 in a group. For the assignment below submit one R-script. Annotations via comments are highly encouraged. The script should run!

Assignment Instructions:

1: Select some form of linear model containing a single dependent variable (continuous) and at least 1 independent variable. Next, simulate two datasets: the first with no relationship between X & Y, and the second with some positive association between X & Y. Perform 100 simulations under each condition. Run the linear models on all datasets to confirm that on average, the patterns for condition 1 (no relationship) and condition 2 (some relationship) are met. (HINT: this requires determining an appropriate summary measure extracted from the linear model).

2: Devise a permutation procedure to evaluate the above linear model. Write code for this permutation procedure. Next, devise a SECOND implementation of the same permutation procedure (ie, code the procedure in a different manner). For a single dataset compare the two implementations for their computational performance. Summarize your findings via comments in the code (e.g., which approach was faster? Which components of the slower approach could be improved, etc.).

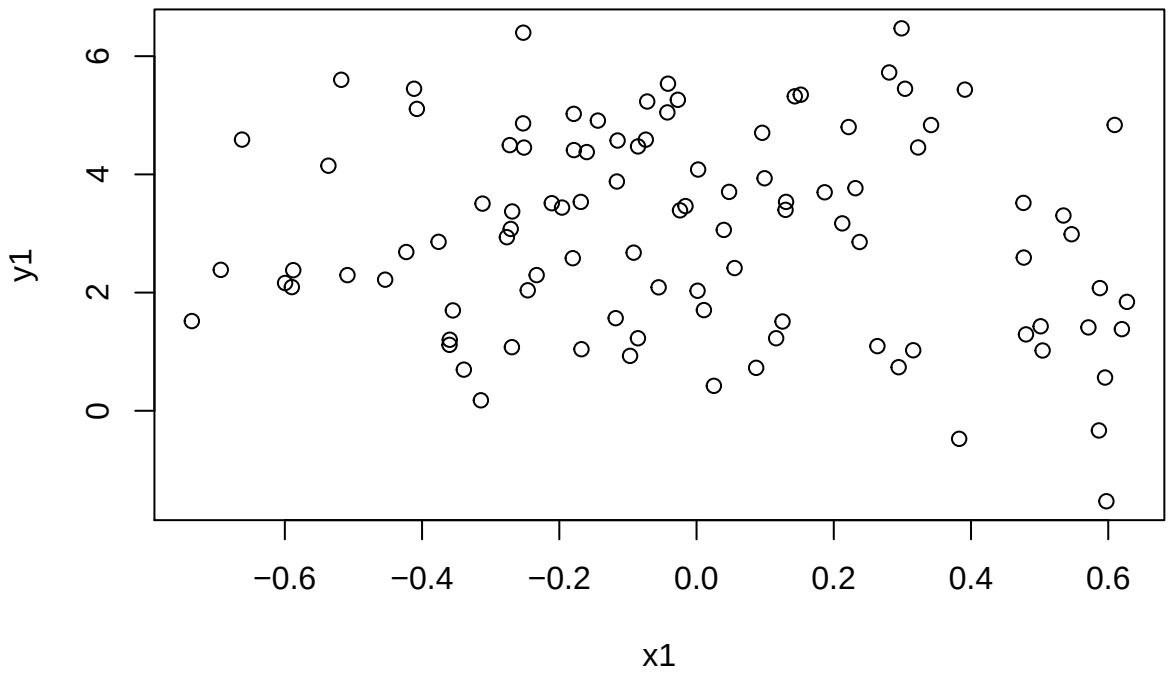
Homework Breakdown:

Select some form of linear model containing a single dependent variable (continuous) and at least 1 independent variable. Next, simulate two datasets: the first with no relationship between X & Y, and the second with some positive association between X & Y.

```
# data set with no relationship between x & Y

random_corr_gen <- function(elements) {
    set.seed(2)
    x1 <- rnorm(elements, mean = 0, sd = 0.3)  # this could also be simply 1:100
    set.seed(3)
    y1 <- rnorm(elements, mean = 3, sd = 2)  #this could be pulled from any distribution (ie poisson, u
    uncor_dataframe <- data.frame(x1, y1)
}

uncor_data <- random_corr_gen(100)
plot(uncor_data)
```




```
# TODO ADD LABELS AND TITLE TO PLOT
```

```
# data set with positive association between x & Y
library(mvtnorm)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.6     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
corr.val <- 0.9
pos_cor_gen <- function(elements) {
    cor_data <- data.frame(rmvnorm(n = elements, mean = c(0,
        0), sigma = matrix(c(1, corr.val, corr.val, 1), 2, 2)))
    names(cor_data)[1] <- "x1"
    names(cor_data)[2] <- "y1"
    return(cor_data)
}

pos_cor_data <- pos_cor_gen(100)
```

Perform 100 simulations under each condition.

First we simulate 100 noncorrelated x and y dataframes and put it a list called uncor_data_sim.

```
# Random Uncorrelated Data List
uncor_data_sim <- replicate(100, random_corr_gen(100), simplify = FALSE)
```

Then we simulate 100 lists of the correlated data, called cor_data_sim.

```
# Random Correlated Data List
cor_data_sim <- replicate(100, pos_cor_gen(100), simplify = FALSE)

# cor_data_sim cor_data_sim[[1]] #shows the first list of 100
# elements cor_data_sim[[1]][,1] #shows column x of the first
# list of 100 elements
```

Run the linear models on all datasets to confirm that on average, the patterns for condition 1 (no relationship) and condition 2 (some relationship) are met. (HINT: this requires determining an appropriate summary measure extracted from the linear model).

Below is the linear model lm function applied to each list of x and y in the uncorrelated dataset. The summary of the linear model is saved tp summary_lm_uncor_tests.

```
# model 1 - uncorrelated
lm_uncor_tests <- lapply(uncor_data_sim, lm, formula = y1 ~ x1)
summary_lm_uncor_tests <- lapply(lm_uncor_tests, summary)
# TODO -> sum up the pvalues that were less than 0.05
```

Next is the linar model lm applied to each list of the positively correlated dataset.

```
# model 2 - positively correlated model
lm_cor_tests <- lapply(cor_data_sim, lm, formula = y1 ~ x1)
summary_lm_cor_tests <- lapply(lm_cor_tests, summary)
# TODO -> sum up the pvalues that were less than 0.05
```

There is a significant linear relationship between x and y for the positively correlated model (model2) with a p value of 2.2e-16. #### Question 2:

Devise a permutation procedure to evaluate the above linear model. Write code for this permutation procedure.

```
# install.packages('RRPP')
library(RRPP)
ourdata <- rrpp.data.frame(pos_cor_data[, 2], pos_cor_data[,
    1])
model3 <- lm.rrpp(pos_cor_data[, 2] ~ pos_cor_data[, 1], print.progress = FALSE,
    data = ourdata)
anova(model3)
```

```
##
## Analysis of Variance, using Residual Randomization
## Permutation procedure: Randomization of null model residuals
## Number of permutations: 1000
## Estimation method: Ordinary Least Squares
## Sums of Squares and Cross-products: Type I
## Effect sizes (Z) based on F distributions
##
##                   Df     SS     MS     Rsq      F     Z Pr(>F)
## pos_cor_data[, 1]  1 71.481 71.481 0.74547 287.02 7.599  0.001 **
## Residuals         98 24.407  0.249 0.25453
## Total             99 95.888
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Call: lm.rrpp(f1 = pos_cor_data[, 2] ~ pos_cor_data[, 1], data = ourdata,
##     print.progress = FALSE)
```

Next, devise a SECOND implementation of the same permutation procedure (ie, code the procedure in a different manner). For a single dataset compare the two implementations for their computational performance.

```
# F.obs<-anova(summary_lm_cor_tests)$F[[1]] #Find Test value
# and save permute<-1999 F.rand.vec<-array(NA,(permute+1))
# F.rand.vec[permute+1]<-F.obs Y = pos_cor_data[,2] X1 =
# pos_cor_data[,1] for(i in 1:permute){ ###Shuffle Data
# y.rand<-sample(Y) #Resample vector
# F.rand.vec[i]<-anova(lm(y.rand~X1))$F[[1]] } F.obs
# P.Ftest<-rank(F.rand.vec[permute+1])/(permute+1) P.Ftest
# ####Plot hist(F.rand.vec,40,freq=T,col='gray')
# segments(F.obs, 0, F.obs, 50) ##Plot Observed value

# TODO --> make this a nice replcate or apply statement
```

Summarize your findings via comments in the code (e.g., which approach was faster? Which components of the slower approach could be improved, etc.).

```
# TODO time the two methods of permutations
# https://www.rdocumentation.org/packages/microbenchmark/versions/1.4-7/topics/microbenchmark
# https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/system.time
```