# 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.).

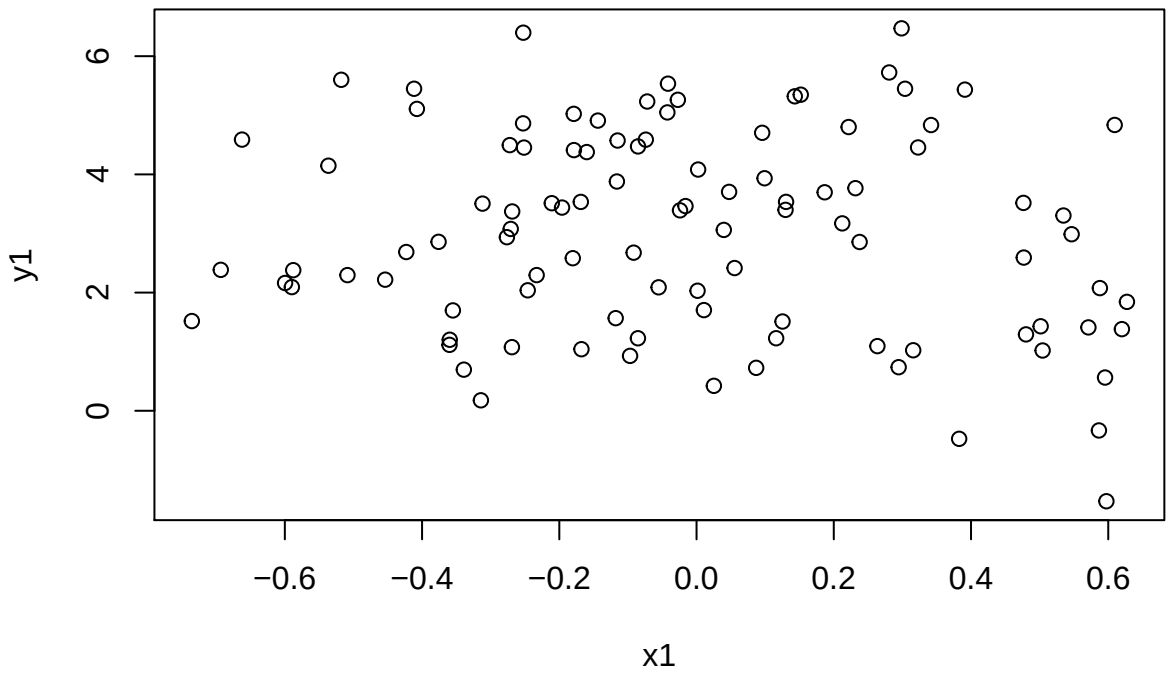
#### Homework Breakdown:

Select some form of linear model containing a single dependent variable (continuous) and at least 1 independent variable.

```
# TODO DO WE NEED TO DECLARE A LINEAR MODEL HERE? SHOULD WE
# REMOVE THIS BREAK? wHAT IS CONSIDERED A LINEAR MODEL?
```

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
set.seed(2)
x1 <- rnorm(100, mean = 0, sd = 0.3)
set.seed(3)
y1 <- rnorm(100, mean = 3, sd = 2)  #this could be pulled from any distribution (ie poisson, uniform...
plot(x1, y1)
```

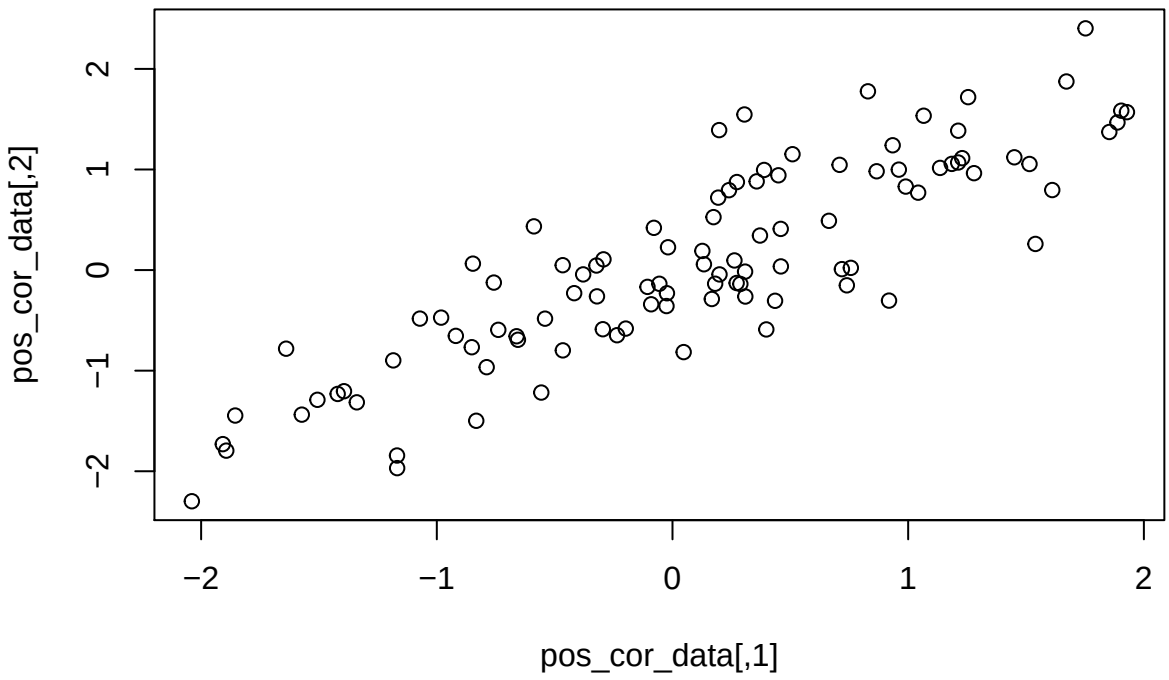



```
# TODO ADD LABELS AND TITLE
```

```
# data set with positive association between x & Y
# install.packages('mvtnorm')
library(mvtnorm)
corr.val <- 0.9
pos_cor_data <- rmvnorm(n = 100, mean = c(0, 0), sigma = matrix(c(1,
    corr.val, corr.val, 1), 2, 2))
cor(pos_cor_data)
```

```
##           [,1]      [,2]
## [1,] 1.0000000 0.8634035
## [2,] 0.8634035 1.0000000
```

```
plot(pos_cor_data)
```




Perform 100 simulations under each condition.

```
# TODO DOES THIS MEAN RESAMPLING OR DOES THIS MEAN INSTEAD OF
# 500 datapoints in each, there should be only 100? MAKE
# DATAFRAME OF 100 of 100 FILL THE DATAFRAME ROW BY ROW WITH
# LIST OF 100


# load in libraries Random Uncorrelated Data List
uncor_data <- list()
sum_slopes_uncor_data <- 0

for (i in 1:100) {
    tmp <- list(rnorm(100, mean = 0, sd = 0.3))  #generate 100 uncorrelated points
    name <- paste("simulatation ", i, sep = "")  #make name for indexing later
    uncor_data[[name]] <- tmp  # add simulation to list

    tmp_model <- lm(1:100 ~ tmp[[1]])  #make linear model
    uncor_data[[name]][["linear_model"]] <- tmp_model  #add linear model to list

    uncor_data[[name]][["summary"]] <- summary(tmp_model)  #summarize the fit of the linear model

    uncor_data[[name]][["anova"]] <- anova(tmp_model)  #get model term tests

    sum_slopes_uncor_data <- sum_slopes_uncor_data + uncor_data[[name]][["linear_model"]][["coefficients
}
```

```
# unlisted <- unlist(uncor_data[['simulatation 1']])

# ununlisted <- unlist(unlisted)

# sum_slope_uncor_data_new <-
# lapply(uncor_data[[]][['linear_model']][['coefficients']][['tmp[[1]]']],
# FUN = sum)

# sum_slope_uncor_data <- tapply(unlist(uncor_data[[]]),
# unlisted[['linear_model.coefficients.tmp[[1]]']],
# lengths(uncor_data[[]])), FUN = sum)

# Random Correlated Data List
cor_data <- list()
corr.val <- 0.9
sum_slopes_cor_data <- 0

for (i in 1:100) {

    tmp <- list(rmvnorm(100, mean = c(0, 0), sigma = matrix(c(1,
        corr.val, corr.val, 1), 2, 2)))  #generate 100 correlated points

    name <- paste("simulatation ", i, sep = "")  #make name for indexing later
    cor_data[[name]] <- tmp  # add simulation to list

    tmp_model <- lm(tmp[[1]][, 1] ~ tmp[[1]][, 2])  #make linear model
    cor_data[[name]][["linear_model"]] <- tmp_model  #add linear model to list

    cor_data[[name]][["summary"]] <- summary(tmp_model)  #summarize the fit of the linear model

    cor_data[[name]][["anova"]] <- anova(tmp_model)  #get model term tests

    sum_slopes_cor_data <- sum_slopes_cor_data + cor_data[[name]][["linear_model"]][["coefficients"]][[
}
```

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).

```
# model 1 - uncorrelated
lm(y1 ~ x1)
```

```
##
## Call:
## lm(formula = y1 ~ x1)
##
## Coefficients:
## (Intercept)           x1
##      3.0156      -0.6975
```

```
model1 <- lm(y1 ~ x1)
summary(model1)  #pvalue of =  0.1593 --> 0.1593 > 0.05 significance threshold
```

```
##
## Call:
```

```
## lm(formula = y1 ~ x1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1298 -1.3120  0.1208  1.3168  3.6638
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0156     0.1704  17.697   <2e-16 ***
## x1           -0.6975     0.4919  -1.418    0.159
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.703 on 98 degrees of freedom
## Multiple R-squared:  0.02011,    Adjusted R-squared:  0.01011
## F-statistic: 2.011 on 1 and 98 DF,  p-value: 0.1593
```

```
# APPLY STATEMENT ON EACH LIST Of 100 ELEMENTS
```

The pvalue of 0.1593 indicating that there is no significant linear relationship between x1 and y1 of model1.

```
# model 2 - positively correlated model

lm(pos_cor_data[, 2] ~ pos_cor_data[, 1])
```

```
##
## Call:
## lm(formula = pos_cor_data[, 2] ~ pos_cor_data[, 1])
##
## Coefficients:
##       (Intercept)  pos_cor_data[, 1]
##        -0.0001028          0.8746253
```

```
model2 <- lm(pos_cor_data[, 2] ~ pos_cor_data[, 1])
summary(model2)
```

```
##
## Call:
## lm(formula = pos_cor_data[, 2] ~ pos_cor_data[, 1])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10684 -0.28693 -0.02869  0.36580  1.28001
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -0.0001028  0.0501526  -0.002    0.998
## pos_cor_data[, 1]  0.8746253  0.0516260  16.942   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.499 on 98 degrees of freedom
## Multiple R-squared:  0.7455, Adjusted R-squared:  0.7429
## F-statistic:   287 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
# APPLY STATEMENT ON EACH LIST Of 100 ELEMENTS
```

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.

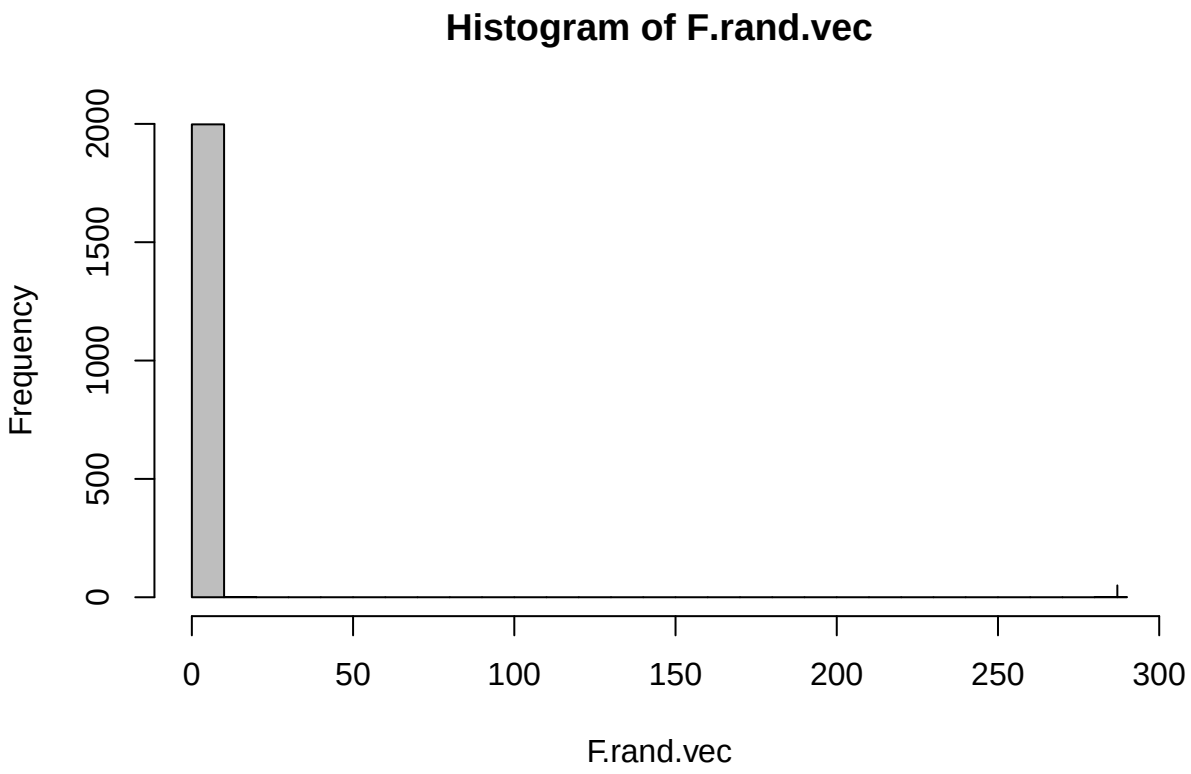
```
F.obs <- anova(model2)$F[[1]]  #Find Test value and save
permute <- 1999
F.rand.vec <- array(NA, (permute + 1))
F.rand.vec[permute + 1] <- F.obs
Y = pos_cor_data[, 2]
X1 = pos_cor_data[, 1]
for (i in 1:permute) {
    ### Shuffle Data
    y.rand <- sample(Y)  #Resample vector
    F.rand.vec[i] <- anova(lm(y.rand ~ X1))$F[[1]]
}
F.obs
```

```
## [1] 287.0168
```

```
P.Ftest <- rank(F.rand.vec[permute + 1])/(permute + 1)
P.Ftest
```

```
## [1] 5e-04
```

```
#### Plot
hist(F.rand.vec, 40, freq = T, col = "gray")
segments(F.obs, 0, F.obs, 50)  ##Plot Observed value
```




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