# Case Study Report

31 October, 2022

```r
data_train<-read_csv("data/data-train.csv")
data_test<-read_csv("data/data-test.csv")
```

## Introduction

## Methodology

### EDA

After loading the data, we performed exploratory data analysis on all three predictors and four moments.

We first noted that the predictor variables `Re` is clustered at fixed values, with `Re` clustering at 90, 224 and 398 (3 levels).

```r
summary(data_train$St)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0500  0.3000  0.7000  0.8596  1.0000  3.0000
```

```r
summary(data_train$Re)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    90.0    90.0   224.0   214.5   224.0   398.0
```

```r
summary(data_train$Fr)
```

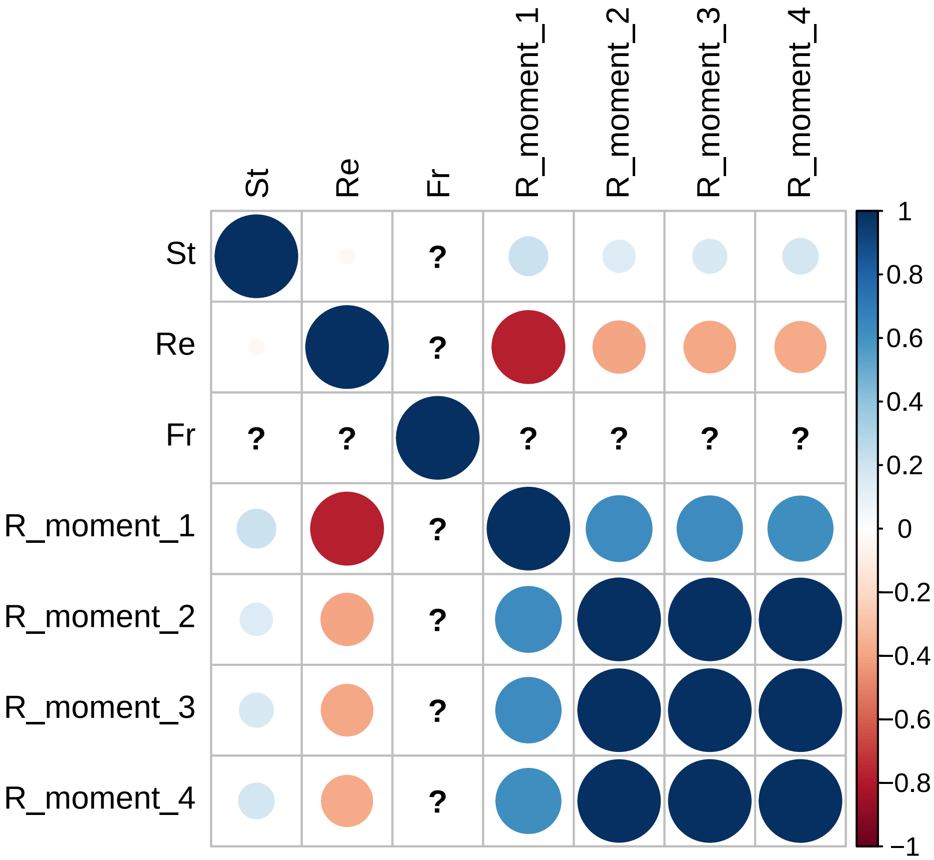
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.052   0.052   0.300     Inf     Inf     Inf
```

We found that the moments are not only highly correlated,

```r
res<-cor(data_train)
res
```

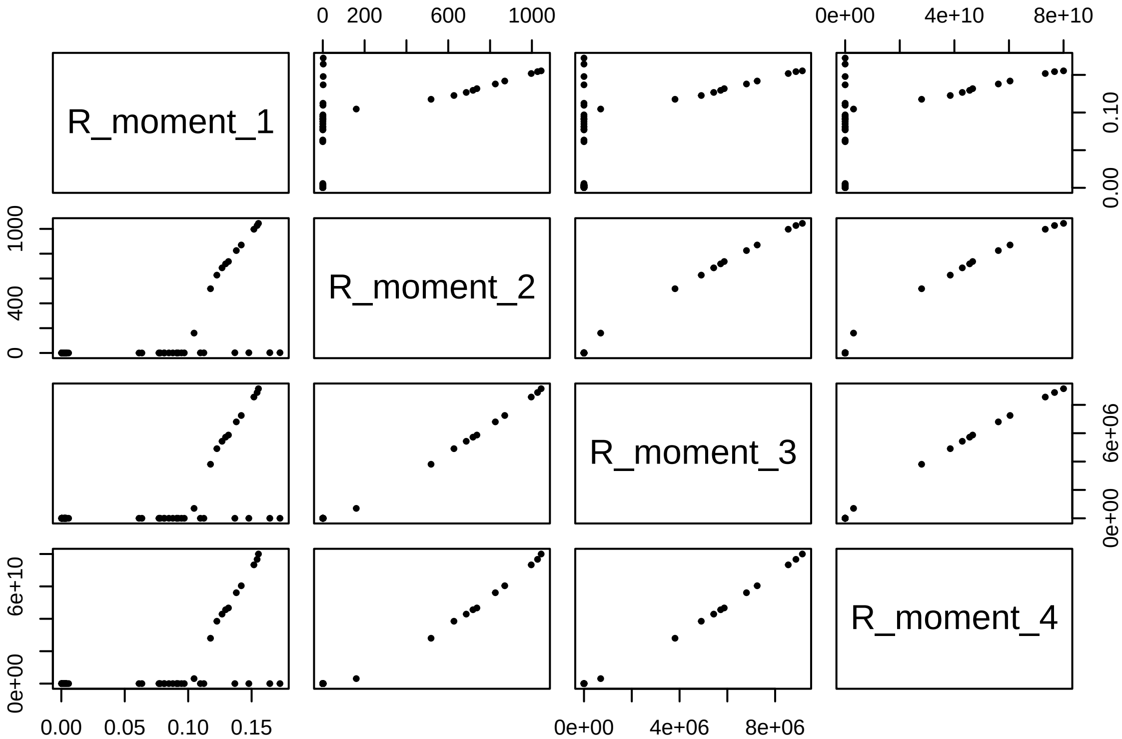
```
##                      St          Re  Fr R_moment_1 R_moment_2 R_moment_3
## St           1.00000000 -0.03169871 NaN  0.2147681  0.1479257  0.1647465
## Re          -0.03169871  1.00000000 NaN -0.7747206 -0.3932344 -0.3844289
## Fr                  NaN         NaN   1        NaN        NaN        NaN
## R_moment_1  0.21476813 -0.77472058 NaN  1.0000000  0.6298829  0.6217326
## R_moment_2  0.14792571 -0.39323445 NaN  0.6298829  1.0000000  0.9984335
## R_moment_3  0.16474648 -0.38442895 NaN  0.6217326  0.9984335  1.0000000
## R_moment_4  0.18004537 -0.37741773 NaN  0.6150484  0.9946671  0.9988414
##             R_moment_4
## St           0.1800454
## Re          -0.3774177
## Fr                 NaN
## R_moment_1   0.6150484
## R_moment_2   0.9946671
## R_moment_3   0.9988414
## R_moment_4   1.0000000
```

```
corrplot(res, tl.col ="black")
```




but that they are linearly correlated:

```
pairs(data_train[4:7], cex = 0.5, pch = 19)
```




Therefore, we decided to fit a model on `R_moment_1`, which will give us the relationship of the predictor variables on the other moments due to the high linear correlation between the moments.

Moreover, we noticed that the gravitational acceleration has infinite values, which is problematic. Therefore, we used inverse logit transform on `Fr` to transform the infinity value into a finite value (Inf transformed to 1):

```
data_train <- data_train %>%
  mutate(Re_category = case_when(Re == 90 ~ "Low", Re==224 ~ "Medium", Re == 398 ~ "High"))%>%
  mutate(Fr_transformed = invlogit(Fr))
```

We then explored shrinkage methods such as ridge regression and lasso. However, since we know that the three predictors are all active so that we do not need predictor selection, so we attempted to fit a ridge regression model:

### Ridge Model

```
y <- data_train$R_moment_1
x <- data.matrix(data_train[, c('Re', 'St', 'Fr_transformed')])
```

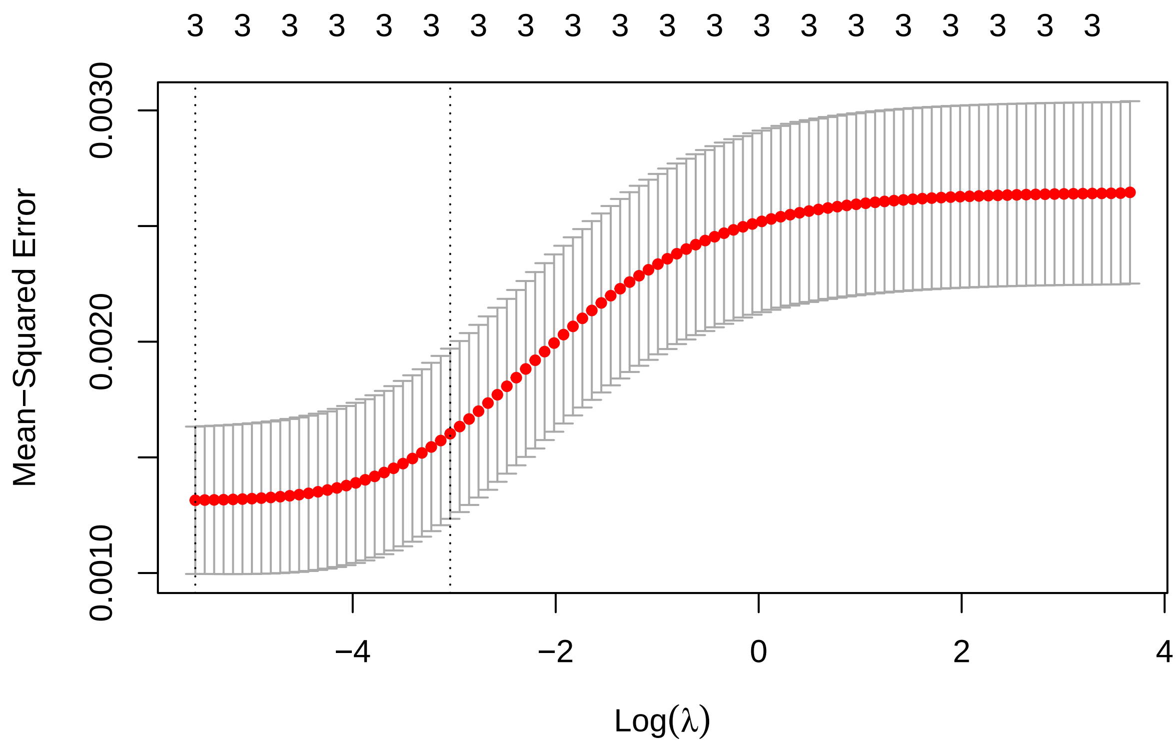
```
set.seed(123)
sample <- sample(c(TRUE, FALSE), nrow(data_train), replace=TRUE)
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]
```

```
lambda_seq = 10^seq(10, -2, length = 100)

ridgemodel <- glmnet(x[train,], y[train], alpha = 0, lambda = lambda_seq)

summary(ridgemodel)
```

```
##           Length Class     Mode
## a0        100    -none-    numeric
```

```
## beta      300    dgCMatrix S4
## df        100    -none-    numeric
## dim         2    -none-    numeric
## lambda    100    -none-    numeric
## dev.ratio 100    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        5    -none-    call
## nobs        1    -none-    numeric
```

```r
set.seed(123)
#perform k-fold cross-validation to find optimal lambda value
cv_ridgemodel <- cv.glmnet(x[train,], y[train], alpha = 0)
plot(cv_ridgemodel)
```




```r
set.seed(123)
#find optimal lambda value that minimizes test MSE
best_lambda <- cv_ridgemodel$lambda.min
best_lambda
```

```
## [1] 0.003882444
```

```r
ridge.pred <- predict(ridgemodel, s = best_lambda, newx = x[test,])
mse <- mean((ridge.pred - y.test)^2)
mse
```

```
## [1] 0.001498519
```

```r
out <- glmnet(x, y, alpha = 0)
predict(out, type = 'coefficients', s = best_lambda)
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                           s1
## (Intercept)     0.112719153
## Re             -0.000350419
## St              0.012535950
## Fr_transformed -0.011465294
```

The ridge regression we fitted through cross-validation gives us an MSE of 0.001498519

```
lm.fit <- lm(R_moment_1 ~ St+ Re_category+Fr_transformed+Fr_transformed*Re_category+St*Re_category, data
lm_summary<-summary(lm.fit)
lm_mse <-mean(lm_summary$residual^2)
final_model<-lm.fit
lm_mse
```

```
## [1] 7.656106e-05
```

By fitting a simple linear regression and adding interaction terms, we obtained a model that has a MSE of 7.656106e-05, which is much smaller than the ridge regression MSE. Moreover, the model fits the data closely with $R^2$ value of 0.9727. Therefore, we decided to use this model as our final model:

```
library(jtools) # Load jtools
```

```
## Warning: package 'jtools' was built under R version 4.1.2
```

```
summ(lm.fit)
```

| | |
|---|---|
| Observations | 89 |
| Dependent variable | R_moment_1 |
| Type | OLS linear regression |

| | |
|---|---|
| F(8,80) | 392.74 |
| $R^2$ | 0.98 |
| Adj. $R^2$ | 0.97 |

| | Est. | S.E. | t val. | p |
|---|---|---|---|---|
| (Intercept) | 0.00 | 0.01 | 0.03 | 0.98 |
| St | 0.00 | 0.00 | 0.02 | 0.98 |
| Re_categoryLow | 0.12 | 0.01 | 12.27 | 0.00 |
| Re_categoryMedium | 0.00 | 0.01 | 0.16 | 0.87 |
| Fr_transformed | 0.00 | 0.01 | 0.01 | 0.99 |
| Re_categoryLow:Fr_transformed | -0.05 | 0.01 | -4.39 | 0.00 |
| Re_categoryMedium:Fr_transformed | 0.00 | 0.01 | 0.05 | 0.96 |
| St:Re_categoryLow | 0.03 | 0.00 | 7.99 | 0.00 |
| St:Re_categoryMedium | 0.00 | 0.00 | 0.23 | 0.82 |

Standard errors: OLS

## Results

## Conclusion