case_study_report

Connie Wu, Jason McEachin, Joe Choo, Scott Heng

10/10/2020

Introduction

Background Information

Turbulence is highly versatile motion that is often times difficult to predict and understanding fluid motion and its effect on natural problems poses a great challenge. Interpreting turbulence data is an incredibly important task in the engineering from understanding the cosmos to the cosmic cycle.

Parametric modeling is effective when we want to compactly represent features as model parameters. Unlike certain "black box" machine learning techniques, it offers a higher level of interpretability, which is especially useful for a practical setting.Rather than simply outputting a classification result, a parametric model allows us to investigate in more detail which aspects of turbulence differ between high and low particle cluster volumes.

In our project, we use linear and (...) and apply it to interpret the difference in model parameters in order to achieve the the following objectives:

1. Build a model that predict its particle cluster volume distribution in terms of the moments.
2. Investigate and interpret how model parameters affects the probability distribution for particle cluster volumes

Data

The data set procured for this case study consists of information about cluster volumes. In total, it contains 89 observations with 7 variables. Details of each variable is specified in Table 1.1. The original response variable, a probability distribution for particle cluster volumes, is difficult to interpret, and therefore is summarized into its first 4 raw moments.

Table 1: Table 1.1 Description Table of Data set variables

Metric	Value	Description
St	$0 < \mathrm{St} < 3$	Particle property: effect on inertia (e.g. size, density)
Re	90, 224, 398	Reynolds Number: turbulent flow property
Fr	Infinity, 0, 3	Particle propety: gravitational acceleration
R moment 1	Continuous response variable	First raw moment of probability distribution
R moment 2	Continuous response variable	Second raw moment of volume probability distribution
R moment 3	Continuous response variable	Third raw moment of volume probability distribution
R moment 4	Continuous response variable	Fourth raw moment of volume probability distribution

Performing some exploratory data analysis, we can observe from Figure 1 that St, with the exception of the 0 values, follow a linear relationship with the first moment, while for Re, there is high variability for the first moment when $Re = 90$, and low variability otherwise. There is high variability across all Fr values with respect to the first moment. Some of these insights influenced our decisions for modelling approaches which will be further discussed in the Methods section.

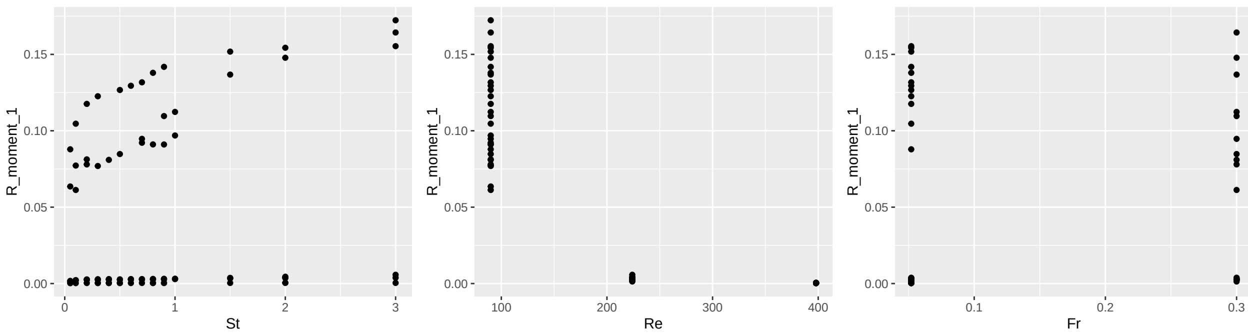


Figure 1: Exploratory data analysis plots on predictors vs first moment

Methodology

We fit a linear regression model with interaction effects, taking Re and Fr as factor variables and St as a bounded, continuous variable. Other models were also explored, but they had larger limitations in comparison with the linear model which is discussed further in the report. We implement a linear regression model for each moment, with a total of 4 models for 4 moments. The baseline for each of the models references $Moment_i$ derived by St=1, Re=224 and Fr=0.052.

For our chosen linear regression model, we can write this in statistical notation below:

$$Moment_x = \beta_0 + \beta_1 * St + \sum_{i=2}^{4} \beta_i * Re_i + \sum_{j=5}^{7} \beta_j * Fr_j$$

$$+ \sum_{k=8}^{10} \beta_k * (\text{all two-way interactions between St, Re and Fr})$$

$$\text{For } 1 \leq x \leq 4,$$

Considering the nature of our response variables, since each moment is derived from the same probability distribution, we decided to perform log transformations for the higher-order moments ($Moment_2$, $Moment_3$ and $Moment_4$). Furthmore, our design decision to include interaction effects stem from the exploratory data plot showing certain correlations between the predictor variables, so as to allow our model to be more interpretable to the collinearity between pair of predictors.

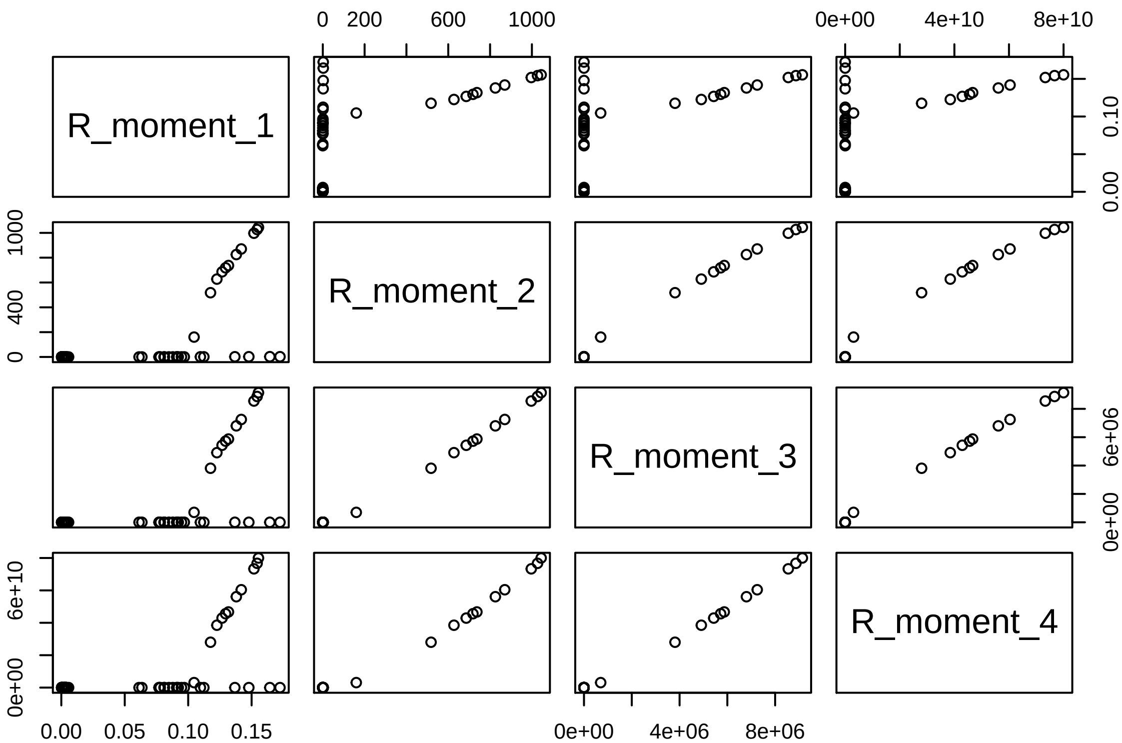
Results

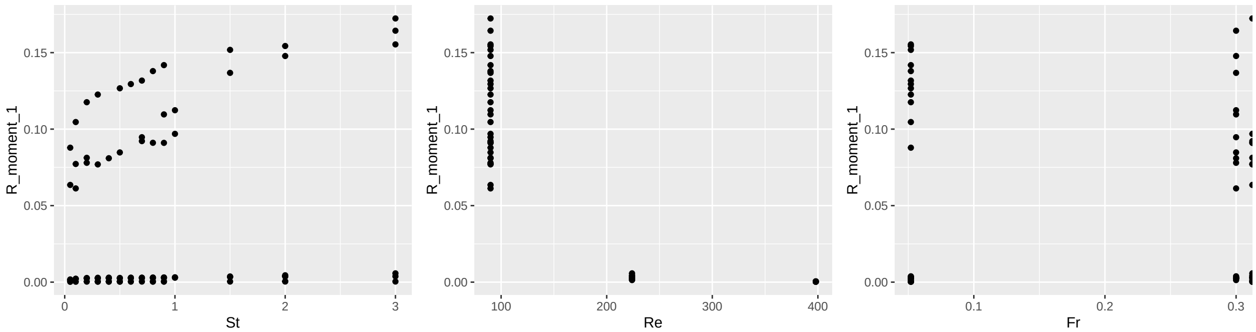
Conclusion

Appendix

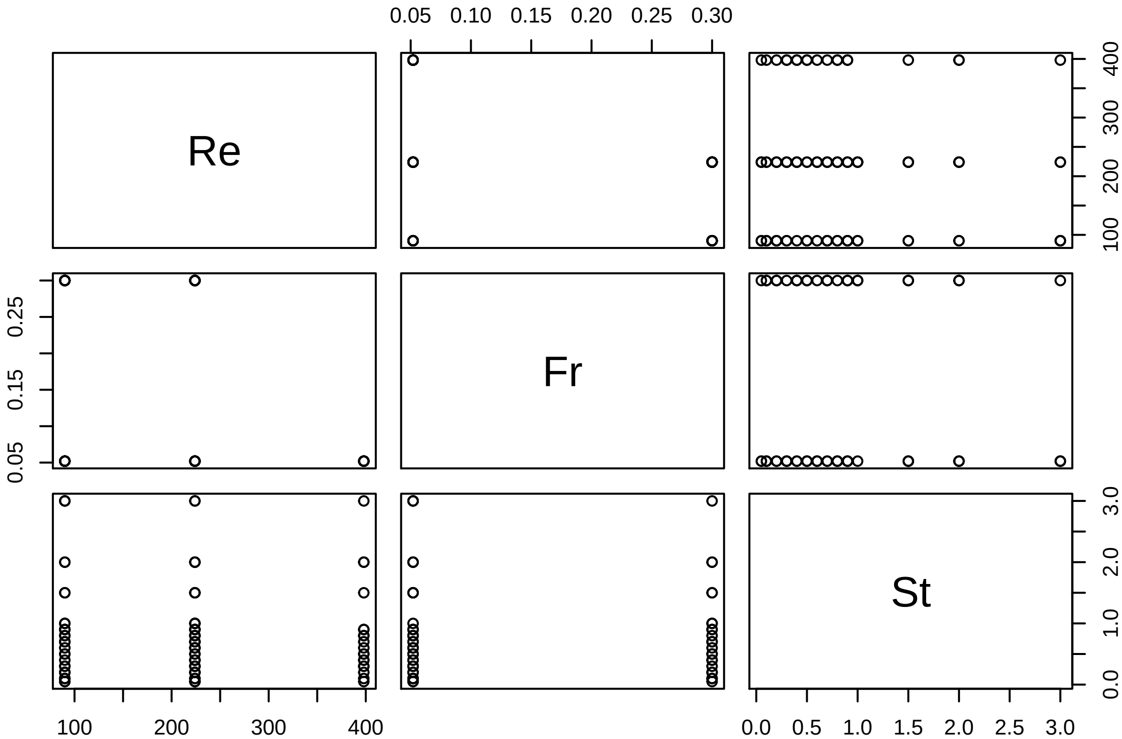
EDA

```r
#pairs plot to show correlation between response variables
train_response <- data.frame(train[,c("R_moment_1","R_moment_2","R_moment_3","R_moment_4")])
pairs(train_response)
```






```r
#correlation between predictor variables (in case we need to include interaction effects)
train_predictors <- data.frame(train[,c("Re","Fr","St")])
pairs(train_predictors)
```



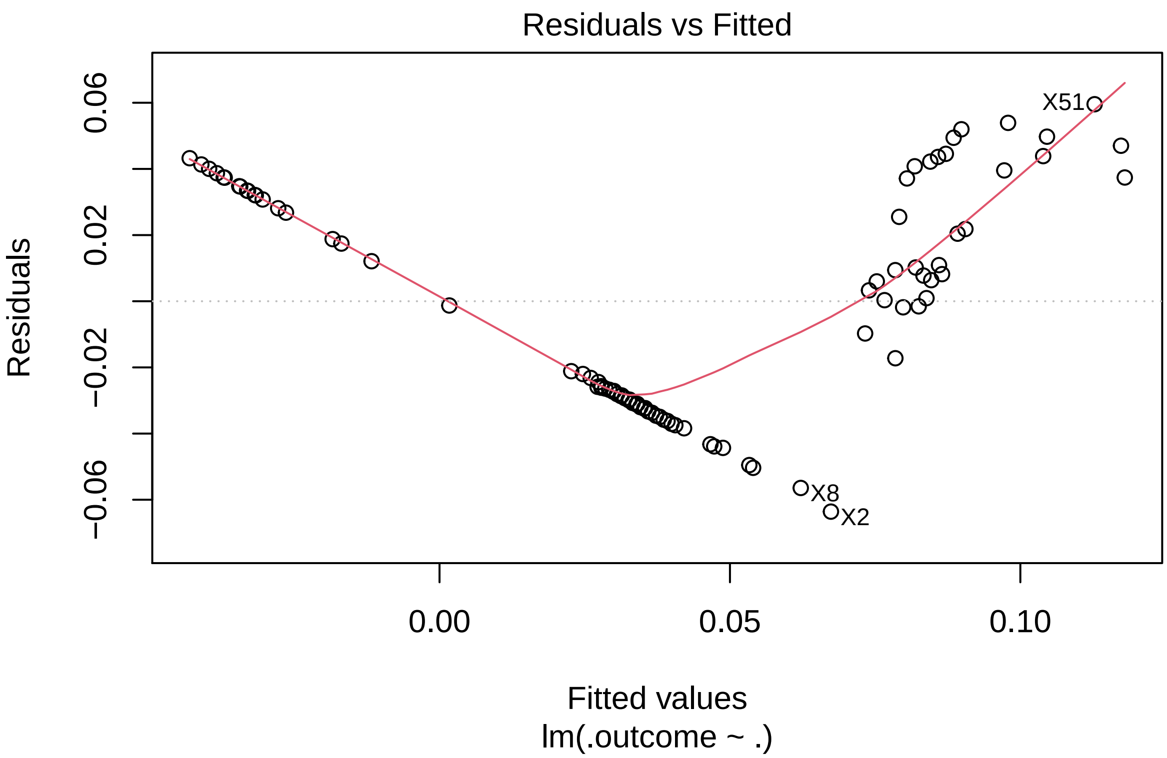
Modeling

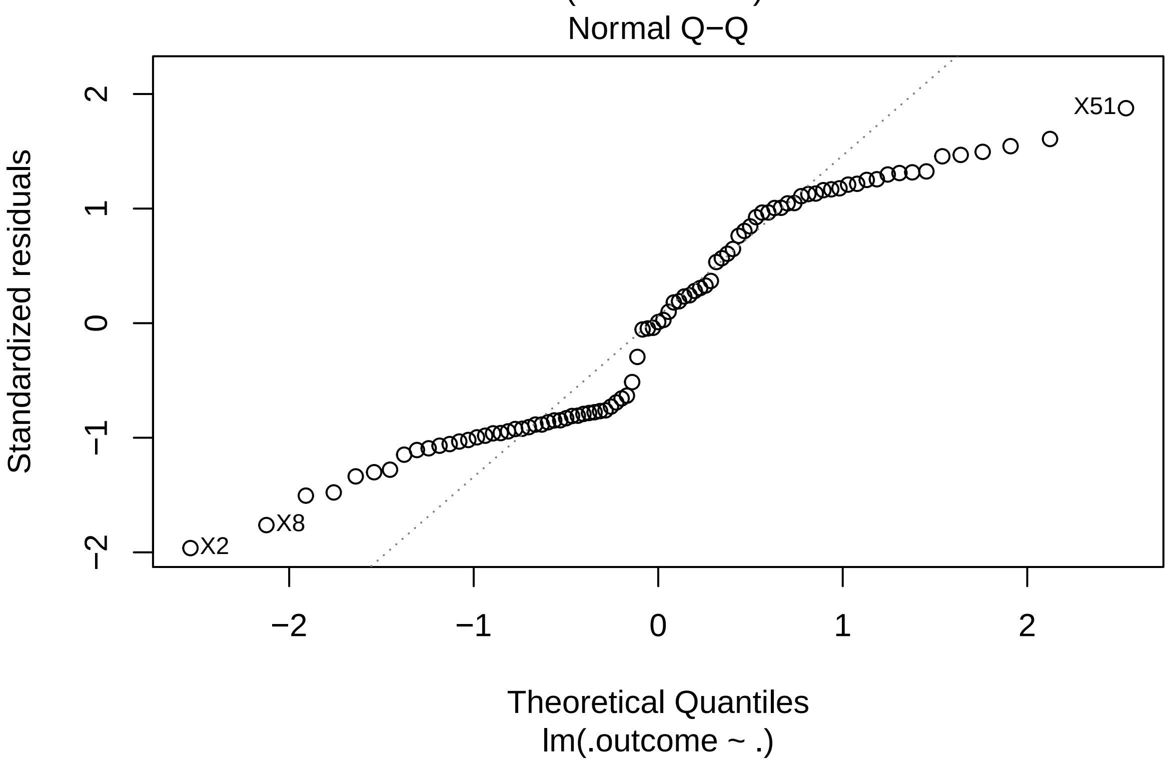
```
data_ctrl <- trainControl(method = "cv", number = 5)
model_caret <- train(R_moment_1 ~ St + Fr.logit + Re, data=train,
                     trControl = data_ctrl,              # folds
                     method = "lm",                       # specifying regression model
                     na.action = na.pass)

model_caret
```

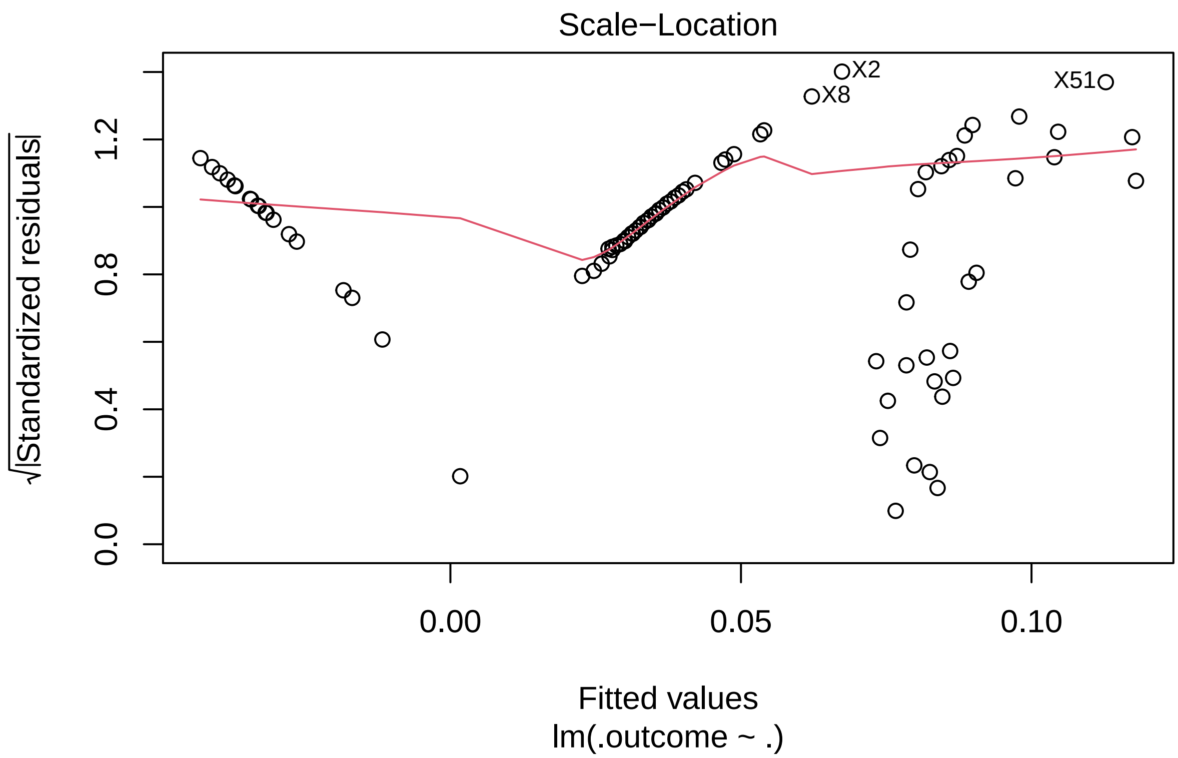
Simple linear modeling

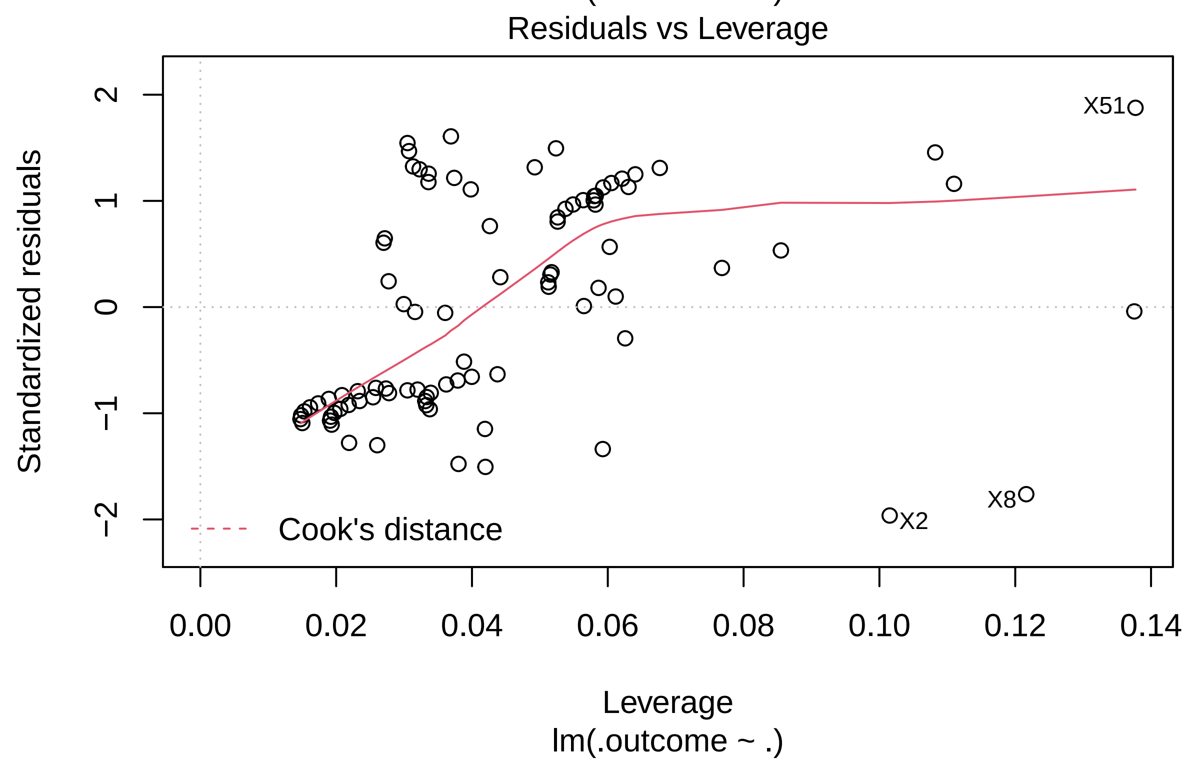
```
## Linear Regression
##
## 89 samples
##  3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 70, 70, 71, 72, 73
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   0.03533856  0.6222672  0.03178162
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
plot(model_caret$finalModel)
```


Residuals vs Fitted
Residuals
0.06
0.02
−0.02
−0.06
0.00
0.05
0.10
X51
X8
X2
Fitted values
lm(.outcome ~ .)


Normal Q−Q
Standardized residuals
2
1
0
−1
−2
X51
X8
X2
−2
−1
0
1
2
Theoretical Quantiles
lm(.outcome ~ .)






```
x <- model.matrix(R_moment_1~St + Fr.logit + Re,data=train)[,-1]

set.seed(17)
train.samp <- sample(1:nrow(train), 4 * nrow(train)/5)
```

```
test <- (-train.samp)
y.test <- train$R_moment_1[test]

preds <- predict(lm_m1_int, newdata = as.data.frame(train[test,]))
mean((preds - y.test)^2)
```

Interaction effects

```
## [1] 0.0009855535
```

```
data_ctrl <- trainControl(method = "cv", number = 5)
model_caret <- train(R_moment_1 ~ St + Fr.logit + Re + St*Fr.logit + St*Re + Fr.logit*Re, data=train,
                     trControl = data_ctrl,              # folds
                     method = "lm",                      # specifying regression model
                     na.action = na.pass)

model_caret
```

```
## Linear Regression
##
## 89 samples
##  3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 70, 71, 72, 72, 71
## Resampling results:
##
##   RMSE        Rsquared  MAE
##   0.03466167  0.621514  0.03131898
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#linear regression with factored Re and Fr.logit
train1 <- train
train1$Re <- as.factor(train$Re)
train1$Fr.logit <- as.factor(train$Fr.logit)
lm1_m1 <- lm(R_moment_1 ~ St + Fr.logit + Re, data=train1)
lm2_m1 <- lm(R_moment_2 ~ St + Fr.logit + Re, data=train1)
lm3_m1 <- lm(R_moment_3 ~ St + Fr.logit + Re, data=train1)
lm4_m1 <- lm(R_moment_4 ~ St + Fr.logit + Re, data=train1)
summary(lm1_m1)
```

Predictors as factors

```
##
## Call:
## lm(formula = R_moment_1 ~ St + Fr.logit + Re, data = train1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.038834 -0.008614  0.001702  0.009854  0.039423
##
## Coefficients:
```

```
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 0.106488   0.003971  26.818  < 2e-16 ***
## St                          0.012213   0.002078   5.877 8.42e-08 ***
## Fr.logit0.574442516811659 -0.007623   0.004245  -1.796  0.07618 .
## Fr.logit1                  -0.010210   0.003787  -2.696  0.00849 **
## Re224                      -0.108091   0.003682 -29.353  < 2e-16 ***
## Re398                      -0.111553   0.004632 -24.081  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01529 on 83 degrees of freedom
## Multiple R-squared:  0.9293, Adjusted R-squared:  0.9251
## F-statistic: 218.2 on 5 and 83 DF,  p-value: < 2.2e-16
```

```
set.seed(17)
train.samp <- sample(1:nrow(train), 4 * nrow(train)/5)
test <- (-train.samp)
y.test <- train$R_moment_1[test]

preds <- predict(lm1_m1, newdata = as.data.frame(train1[test,]))
mean((preds - y.test)^2)
```

```
## [1] 0.0001623039
```

```
set.seed(1)
data_ctrl <- trainControl(method = "cv", number = 5)
model_caret <- train(R_moment_1 ~ St + Fr.logit + Re + St*Fr.logit + St*Re + Fr.logit*Re, data=train1,
                     trControl = data_ctrl,              # folds
                     method = "lm",                      # specifying regression model
                     na.action = na.pass)
```

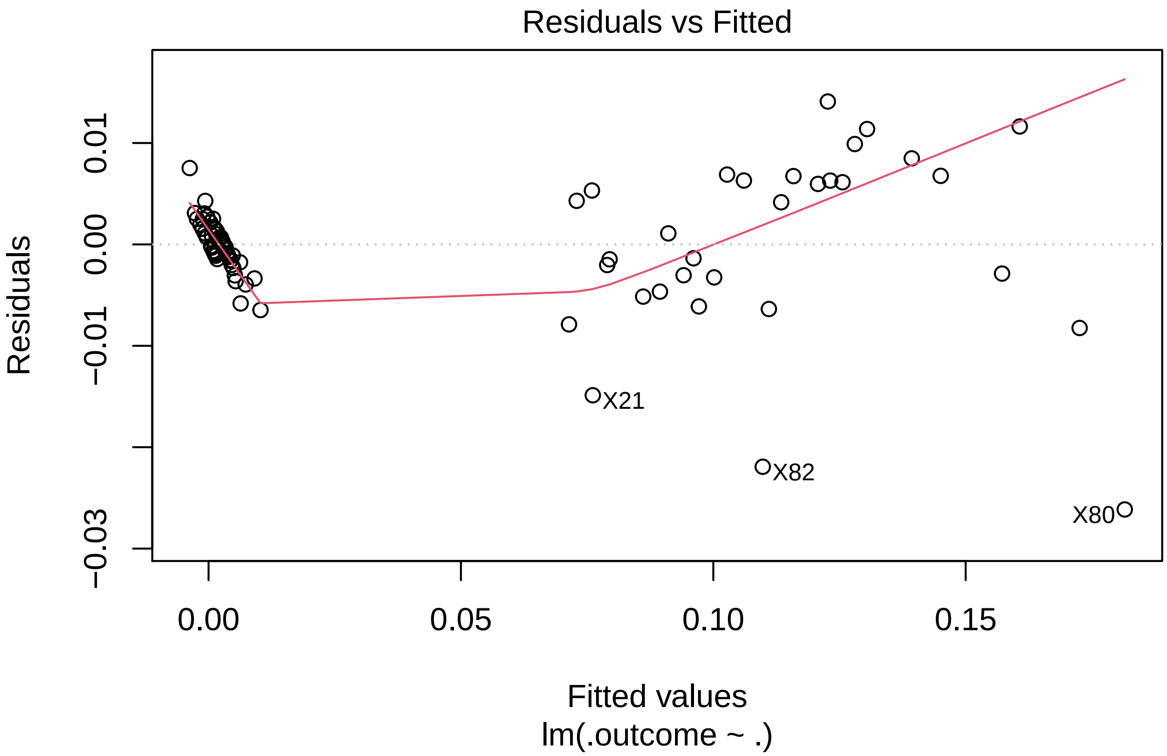
```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

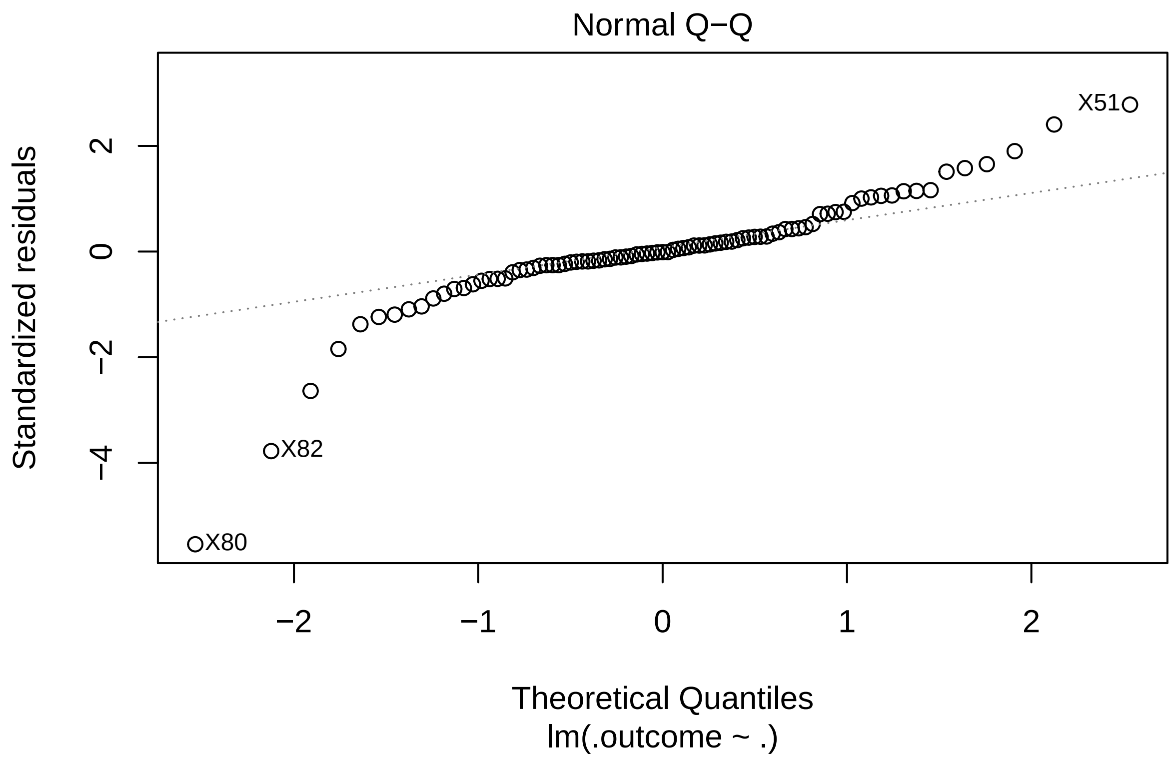
```
model_caret
```

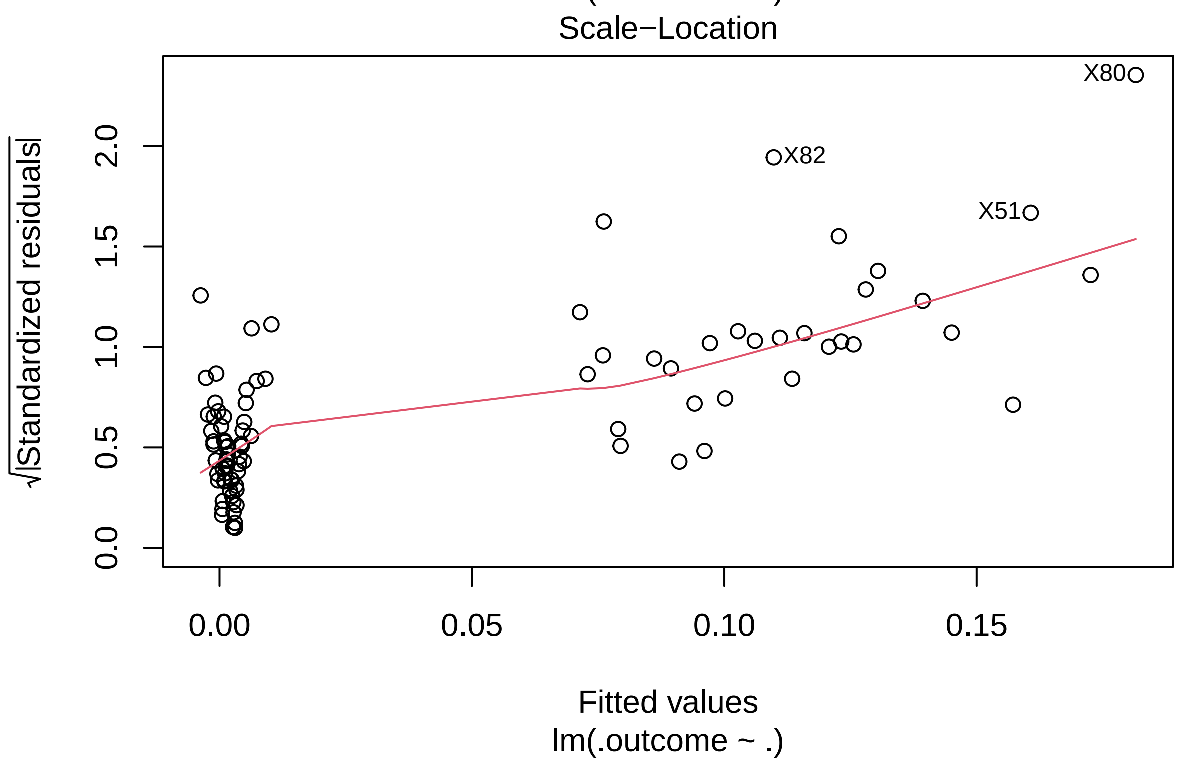
```
## Linear Regression
##
## 89 samples
##  3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 71, 72, 71, 70, 72
```

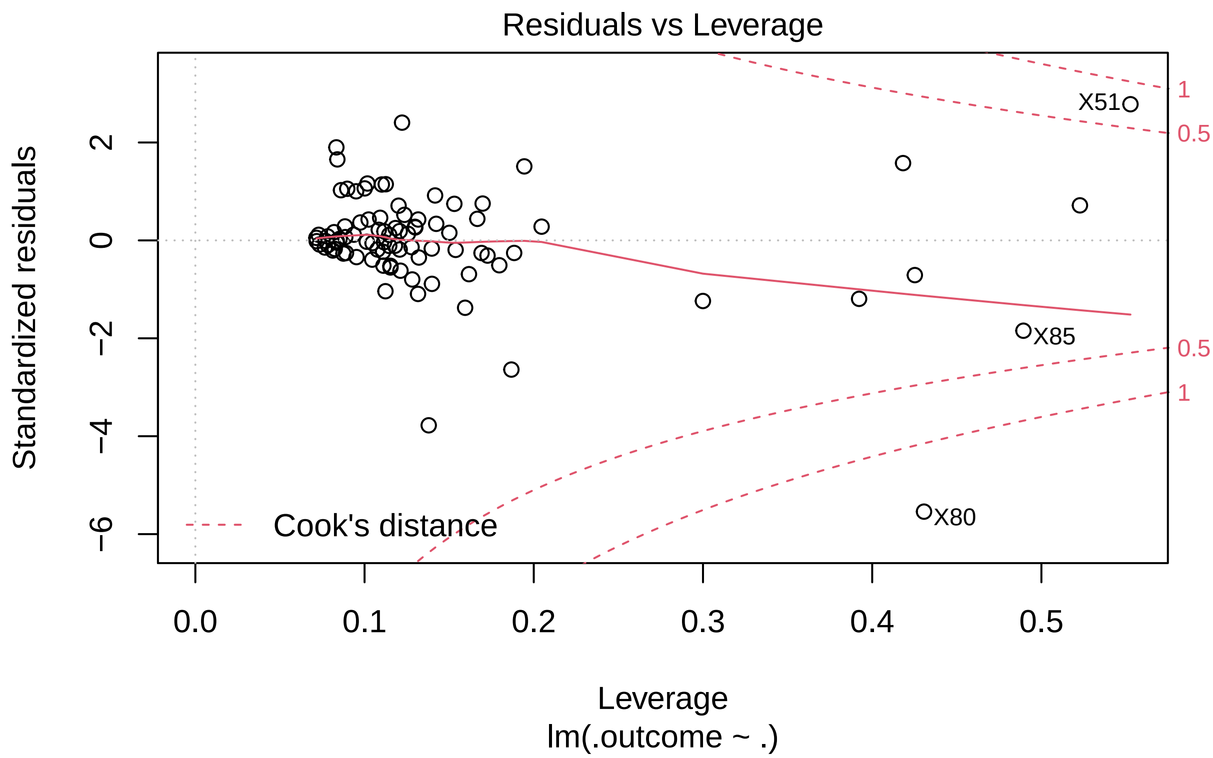
```
## Resampling results:
##
##   RMSE         Rsquared   MAE
##   0.008673306  0.9820262  0.004945823
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
plot(model_caret$finalModel)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(.outcome ~ .)
X51
X82
X80
−4
−2
0
2
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
Fitted values
lm(.outcome ~ .)
X80
X82
X51
0.0
0.5
1.0
1.5
2.0
0.00
0.05
0.10
0.15




```r
summary(model_caret$finalModel)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0261390 -0.0016048 -0.0000646  0.0024878  0.0140925
##
## Coefficients: (1 not defined because of singularities)
##                                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)                           0.108581   0.002378  45.662  < 2e-16 ***
## St                                    0.024313   0.001744  13.939  < 2e-16 ***
## Fr.logit0.574442516811659            -0.035778   0.003540 -10.105 1.05e-15 ***
## Fr.logit1                            -0.038679   0.003219 -12.015  < 2e-16 ***
## Re224                                -0.103060   0.002987 -34.498  < 2e-16 ***
## Re398                                -0.106710   0.003603 -29.616  < 2e-16 ***
## `St:Fr.logit0.574442516811659`        0.008944   0.002380   3.759 0.000333 ***
## `St:Fr.logit1`                        0.005956   0.001995   2.985 0.003812 **
## `St:Re224`                           -0.027400   0.001938 -14.136  < 2e-16 ***
## `St:Re398`                           -0.025834   0.002506 -10.309 4.34e-16 ***
## `Fr.logit0.574442516811659:Re224`     0.028831   0.003657   7.884 1.84e-11 ***
## `Fr.logit1:Re224`                     0.033657   0.003705   9.084 9.21e-14 ***
## `Fr.logit0.574442516811659:Re398`           NA         NA      NA       NA
## `Fr.logit1:Re398`                     0.034294   0.004051   8.465 1.41e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.006253 on 76 degrees of freedom
```

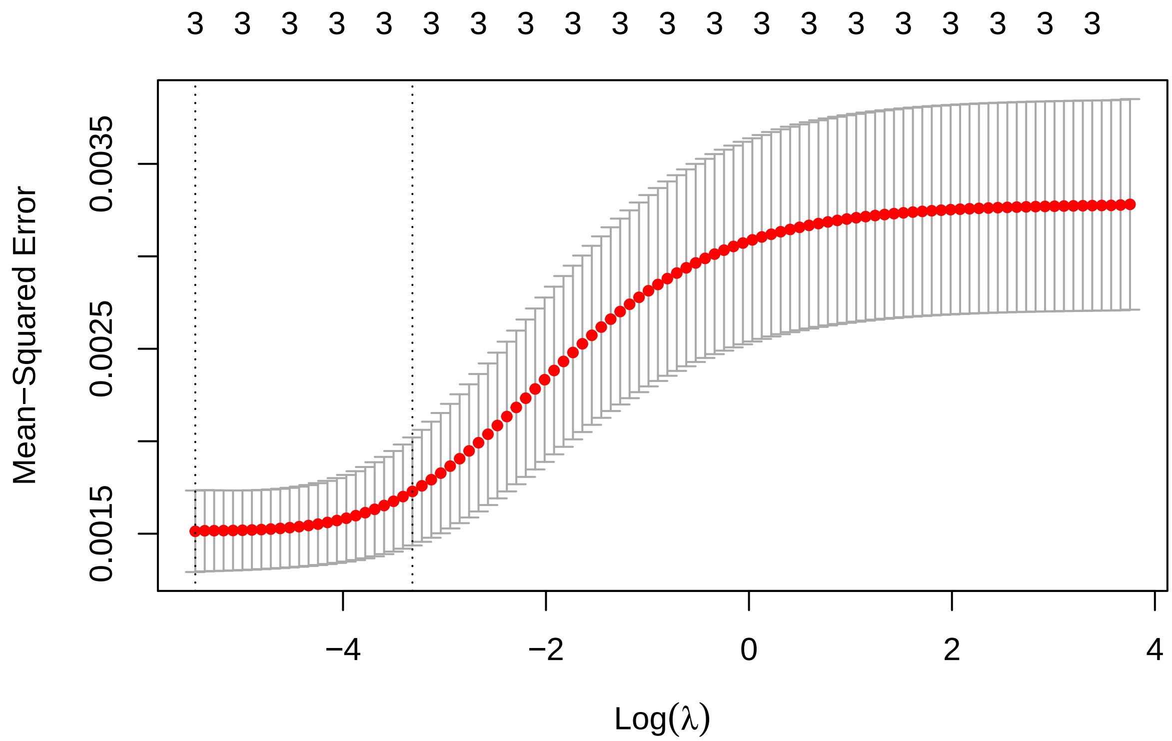
```
## Multiple R-squared:  0.9892, Adjusted R-squared:  0.9875
## F-statistic: 578.6 on 12 and 76 DF,  p-value: < 2.2e-16
```

```
x <- model.matrix(R_moment_1~St + Fr.logit + Re,data=train)[,-1]
y <- train$R_moment_1

set.seed(17)
train.samp <- sample(1:nrow(x), nrow(x)/2)
test.samp <- (-train.samp)
y.test <- y[test.samp]

grid <- 10^seq(10, -2, length = 100) # grid of values for lambda param

ridge.mod <- glmnet(x[train.samp,], y[train.samp], alpha = 0, lambda = grid, thresh = 1e-12)
ridge.pred <- predict(ridge.mod, s=0, x = x[train.samp,], y = y[train.samp],
                      newx = x[test.samp,], exact = T)
mean((ridge.pred - y.test)^2) ## calculate MSE
```

Ridge Regression

```
## [1] 0.001078549
```

```
set.seed(1)
cv.out <- cv.glmnet(x[train.samp,], y[train.samp], alpha = 0)
plot(cv.out)
```




```
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 0.004272015
```

```
ridge.pred <- predict(ridge.mod, s = bestlam, newx = x[test.samp,])
mean((ridge.pred - y.test)^2)
```

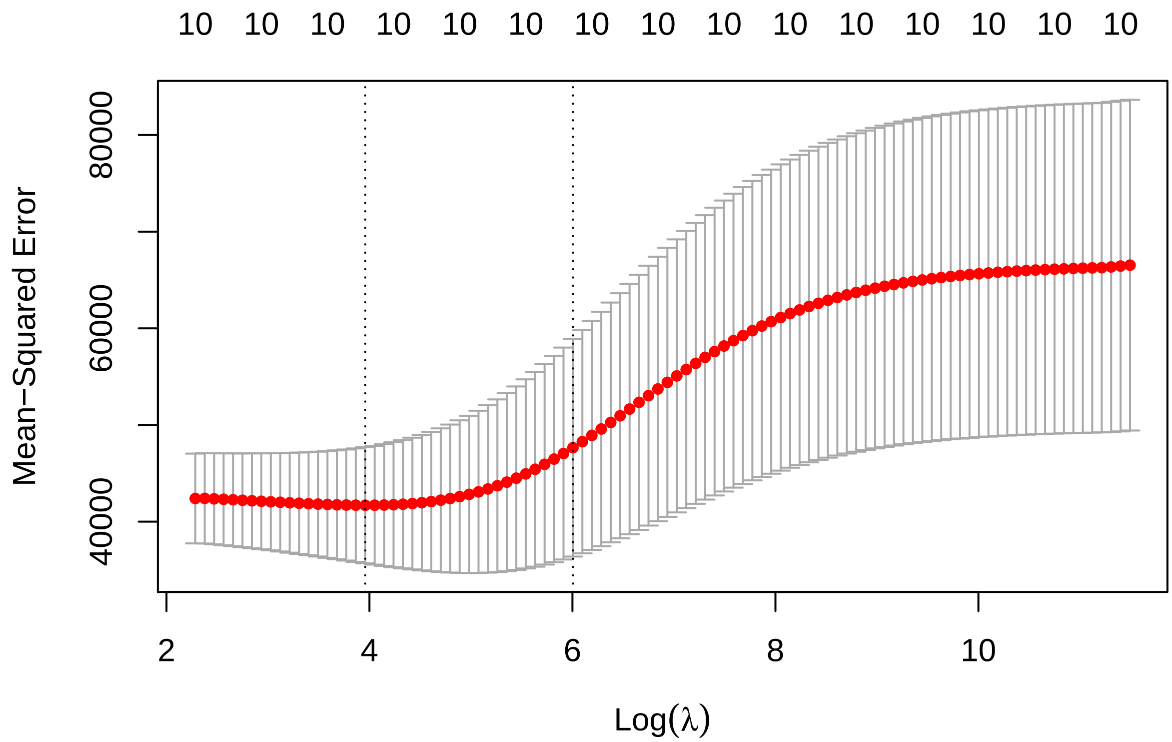
```
## [1] 0.001091239
```

MSE stays basically the same

```
x2 <- model.matrix(R_moment_2~ St + Fr.logit + Re + poly(St,2) + poly(Fr.logit,2) + poly(Re,2) + St:Re,
y2 <- train$R_moment_2

set.seed(17)
train2 <- sample(1:nrow(x2), 4 * nrow(x2)/5)
test2 <- (-train2)
y.test2 <- y2[test2]

grid <- 10^seq(10, -2, length = 100) # grid of values for lambda param

ridge.mod2 <- glmnet(x2[train2,], y2[train2], alpha = 0, lambda = grid, thresh = 1e-12)
ridge.pred2 <- predict(ridge.mod2, s=0, x = x2[train2,], y = y2[train2],
                       newx = x2[test2,], exact = T)
mean((ridge.pred2 - y.test2)^2) ## calculate MSE
```

```
## [1] 43396.09
```

```
set.seed(1)
cv.out2 <- cv.glmnet(x2[train2,], y2[train2], alpha = 0)
plot(cv.out2)
```




```
bestlam2 <- cv.out2$lambda.min
bestlam2
```

```
## [1] 52.36596
```

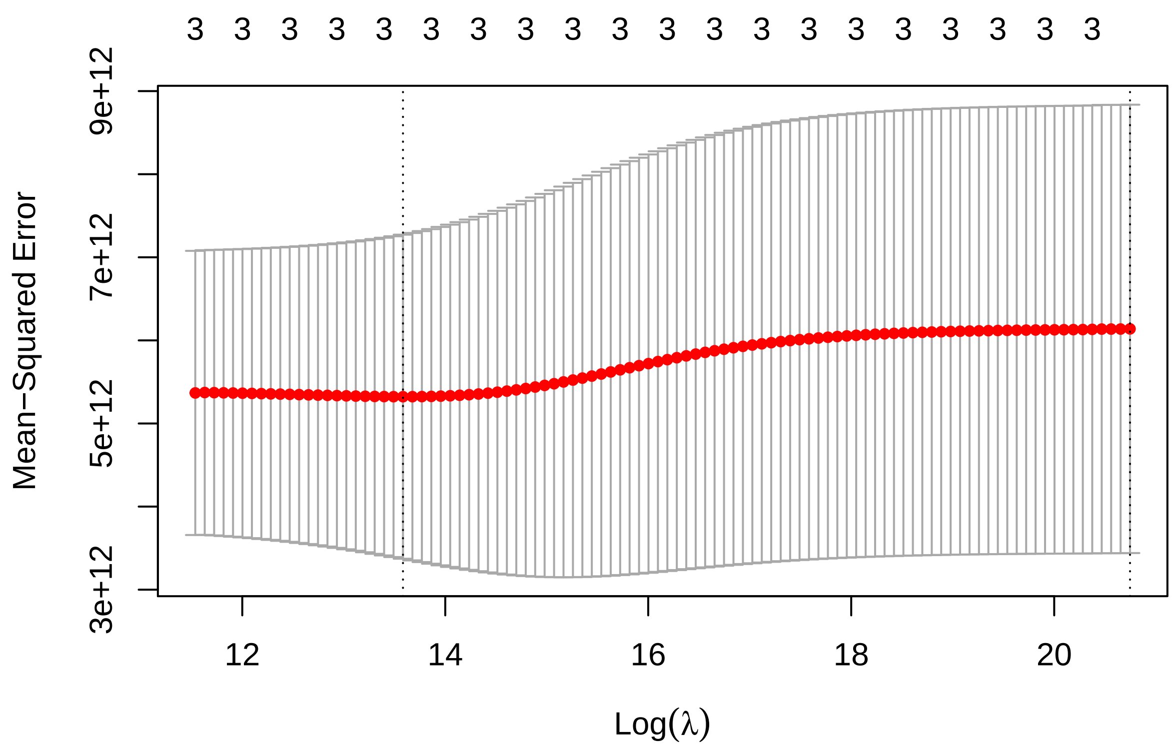
```
ridge.pred2 <- predict(ridge.mod2, s = bestlam2, newx = x2[test2,])
mean((ridge.pred2 - y.test2)^2)
```

```
## [1] 46196.89
```

```
x3 <- model.matrix(R_moment_3~St + Fr.logit + Re,data=train)[,-1]
y3 <- train$R_moment_3

set.seed(17)
train3 <- sample(1:nrow(x3), nrow(x3)/2)
test3 <- (-train3)
y.test3 <- y3[test3]

ridge.mod3 <- glmnet(x3[train3,], y3[train3], alpha = 0, lambda = grid, thresh = 1e-12)
ridge.pred3 <- predict(ridge.mod3, s = 0, newx = x3[test3,])
mean((ridge.pred3 - y.test3)^2) ## calculate MSE
```

```
## [1] 3.166732e+12
```

```
set.seed(1)
cv.out3 <- cv.glmnet(x3[train3,], y3[train3], alpha = 0)
plot(cv.out3)
```




```
bestlam3 <- cv.out3$lambda.min
bestlam3
```

```
## [1] 792973.2
```

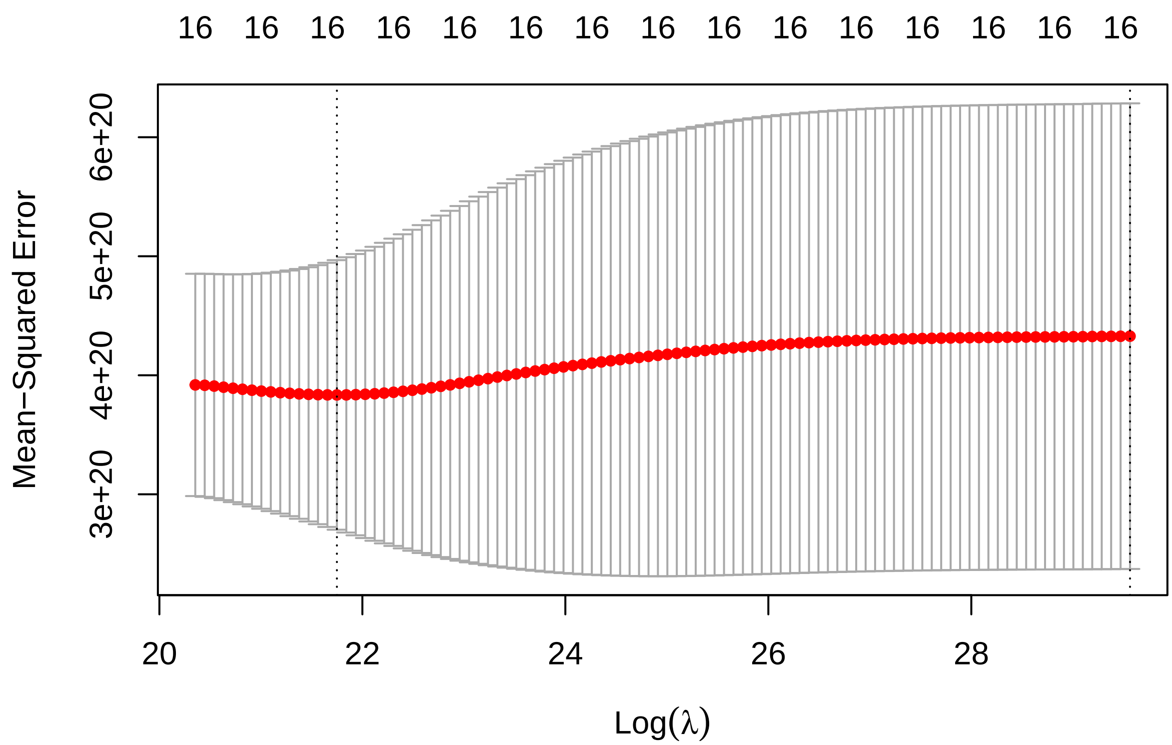
```
ridge.pred3 <- predict(ridge.mod3, s = bestlam3, newx = x3[test3,])
mean((ridge.pred3 - y.test3)^2)
```

```
## [1] 3.007604e+12
```

```r
x4 <- model.matrix(R_moment_4~as.factor(St) + as.factor(Fr.logit) + as.factor(Re),data=train)[,-1]
y4 <- train$R_moment_4

set.seed(17)
train4 <- sample(1:nrow(x4), nrow(x4)/2)
test4 <- (-train4)
y.test4 <- y[test4]

ridge.mod4 <- glmnet(x4[train4,], y4[train4], alpha = 0, lambda = grid, thresh = 1e-12)
ridge.pred4 <- predict(ridge.mod4, s = 0, newx = x4[test4,])
mean((ridge.pred4 - y.test4)^2) ## calculate MSE
```

```
## [1] 3.509278e+20
```

```r
set.seed(1)
cv.out4 <- cv.glmnet(x4[train4,], y4[train4], alpha = 0)
plot(cv.out4)
```




```r
bestlam4 <- cv.out4$lambda.min
bestlam4
```

```
## [1] 2789083228
```

```r
ridge.pred4 <- predict(ridge.mod4, s = bestlam4, newx = x4[test4,])
mean((ridge.pred4 - y.test4)^2)
```

```
## [1] 2.039584e+20
```

improvement? still large

```r
x <- model.matrix(R_moment_1~St + Fr.logit + Re,data=train)[,-1]
```

```
set.seed(17)
train.samp <- sample(1:nrow(train), 4 * nrow(train)/5)
test <- (-train.samp)
y.test <- train$R_moment_1[test]

gam1 <- lm(R_moment_1 ~ ns(St, 1) + ns(Re, 1) + ns(Fr.logit, 1), data = train, subset = train.samp)
preds <- predict(gam1, newdata = as.data.frame(x[test,]))
mean((preds - y.test)^2)
```

GAMS

```
## [1] 0.001226049
```

```
x <- model.matrix(R_moment_2 ~ St + Fr.logit +Re,data=train)[,-1]

set.seed(17)
train.samp <- sample(1:nrow(train), 4 * nrow(train)/5)
test <- (-train.samp)
y.test <- train$R_moment_2[test]

gam2 <- lm(R_moment_2 ~ ns(St, 2) + ns(Re, 2) + ns(Fr.logit, 2), data = train, subset = train.samp)
preds <- predict(gam2, newdata = as.data.frame(x[test,]))
mean((preds - y.test)^2)
```

```
## [1] 48573.71
```