Case Study: Turbulence

2022-10-31

Introduction

Load Data

```
data_train <- read.csv("data-train.csv")
data_test <- read.csv("data-test.csv")
```

Goals (include in writeup)

(1) Prediction: For a new parameter setting of (Re, F r, St), predict its particle cluster volume distribution in terms of its four raw moments.

(2) Inference: Investigate and interpret how each parameter affects the probability distribution for particle cluster volumes.

Exploratory Data Analysis (dont include in writeup, see methodology)

To begin, we will explore the data to ensure it is fit for modelling, determine inital transformations needed of the data, and determine which model we see would best fit the data.

```
names(data_train)
```

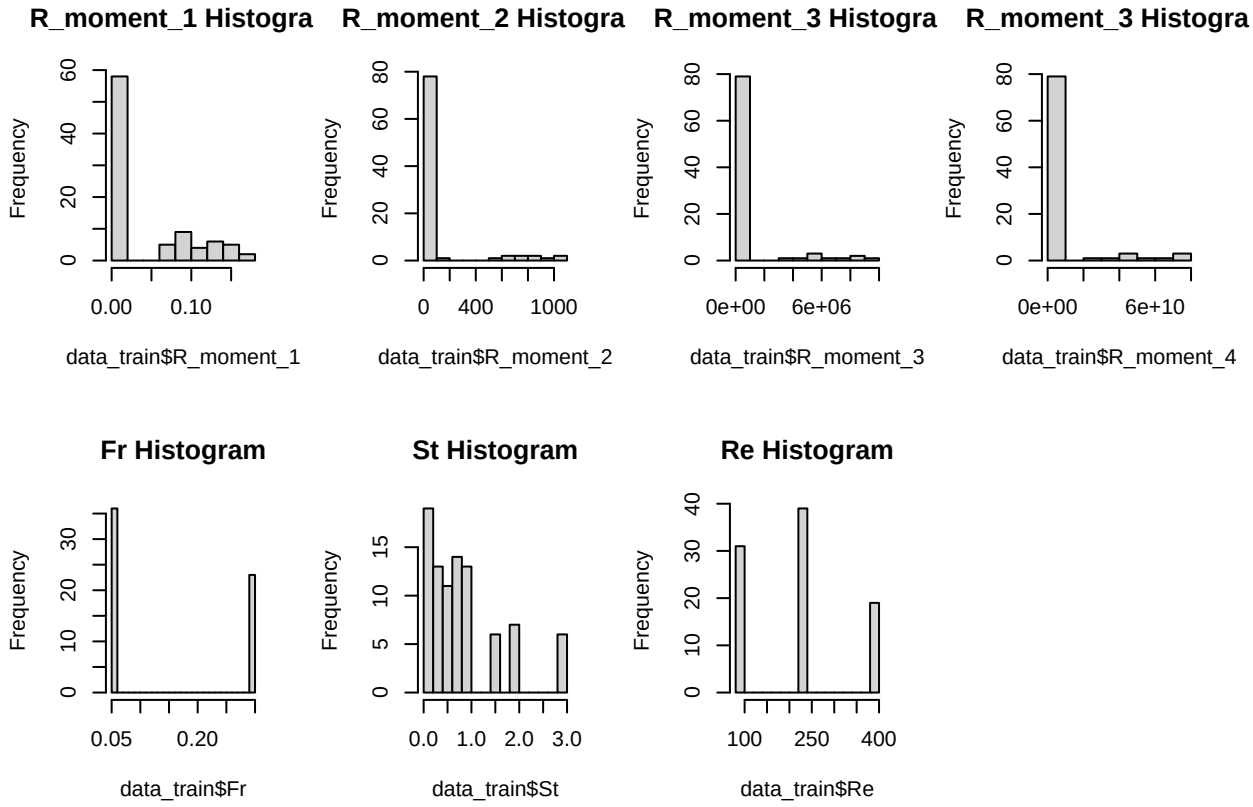
```
## [1] "St"         "Re"         "Fr"         "R_moment_1" "R_moment_2"
## [6] "R_moment_3" "R_moment_4"
```

```
summary(data_train)
```

```
##        St               Re              Fr          R_moment_1
##  Min.   :0.0500   Min.   : 90.0   Min.   :0.052   Min.   :0.000222
##  1st Qu.:0.3000   1st Qu.: 90.0   1st Qu.:0.052   1st Qu.:0.002157
##  Median :0.7000   Median :224.0   Median :0.300   Median :0.002958
##  Mean   :0.8596   Mean   :214.5   Mean   :  Inf   Mean   :0.040394
##  3rd Qu.:1.0000   3rd Qu.:224.0   3rd Qu.:  Inf   3rd Qu.:0.087868
##  Max.   :3.0000   Max.   :398.0   Max.   :  Inf   Max.   :0.172340
##    R_moment_2          R_moment_3        R_moment_4
##  Min.   :   0.0001   Min.   :      0   Min.   :0.000e+00
##  1st Qu.:   0.0245   1st Qu.:      0   1st Qu.:3.000e+00
##  Median :   0.0808   Median :      1   Median :2.100e+01
##  Mean   :  92.4902   Mean   : 753370   Mean   :6.194e+09
##  3rd Qu.:   0.5345   3rd Qu.:     40   3rd Qu.:5.345e+03
##  Max.   :1044.3000   Max.   :9140000   Max.   :8.000e+10
```

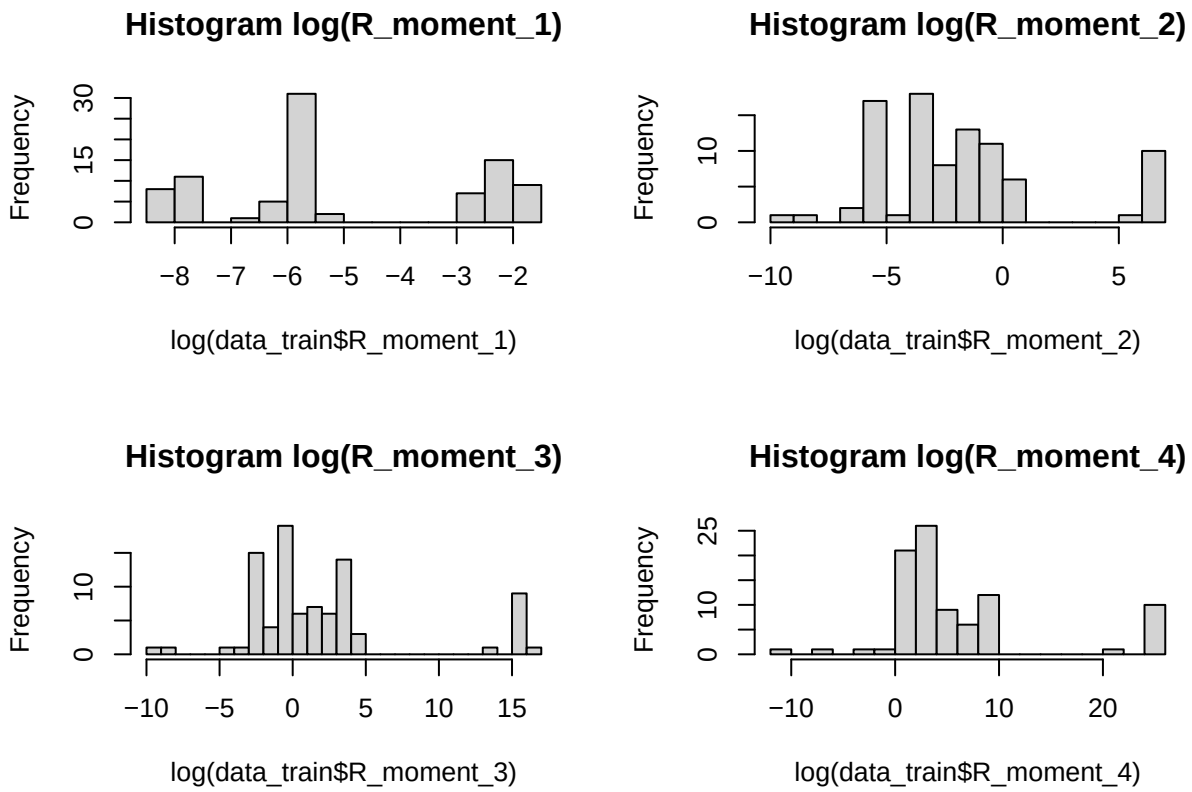
Histograms

```
par(mfrow = c(2, 4))
hist(data_train$R_moment_1, main = paste("R_moment_1 Histogram"))
hist(data_train$R_moment_2, main = paste("R_moment_2 Histogram"))
hist(data_train$R_moment_3, main = paste("R_moment_3 Histogram"))
hist(data_train$R_moment_4, main = paste("R_moment_3 Histogram"))
hist(data_train$Fr, breaks=20, main = paste("Fr Histogram"))
hist(data_train$St, breaks=20, main = paste("St Histogram"))
hist(data_train$Re, breaks=20, main = paste("Re Histogram"))
```



These histograms indicate that each R_moment has distributions that are heavily skewed to the right, because the data has many rows of 0. In R_moment_3 and R_moment_4, we notice that the maximum values are extremely high, while the medians are much smaller in comparison. This could pose a problem to our analysis, thus we believe it is best then to apply a transformation to these variables in order to obtain more accurate analysis.

```
par(mfrow = c(2, 2))
hist(log(data_train$R_moment_1), breaks=20, main = paste("Histogram log(R_moment_1)"))
hist(log(data_train$R_moment_2), breaks=20, main = paste("Histogram log(R_moment_2)"))
hist(log(data_train$R_moment_3), breaks=20, main = paste("Histogram log(R_moment_3)"))
hist(log(data_train$R_moment_4), breaks=20, main = paste("Histogram log(R_moment_4)"))
```

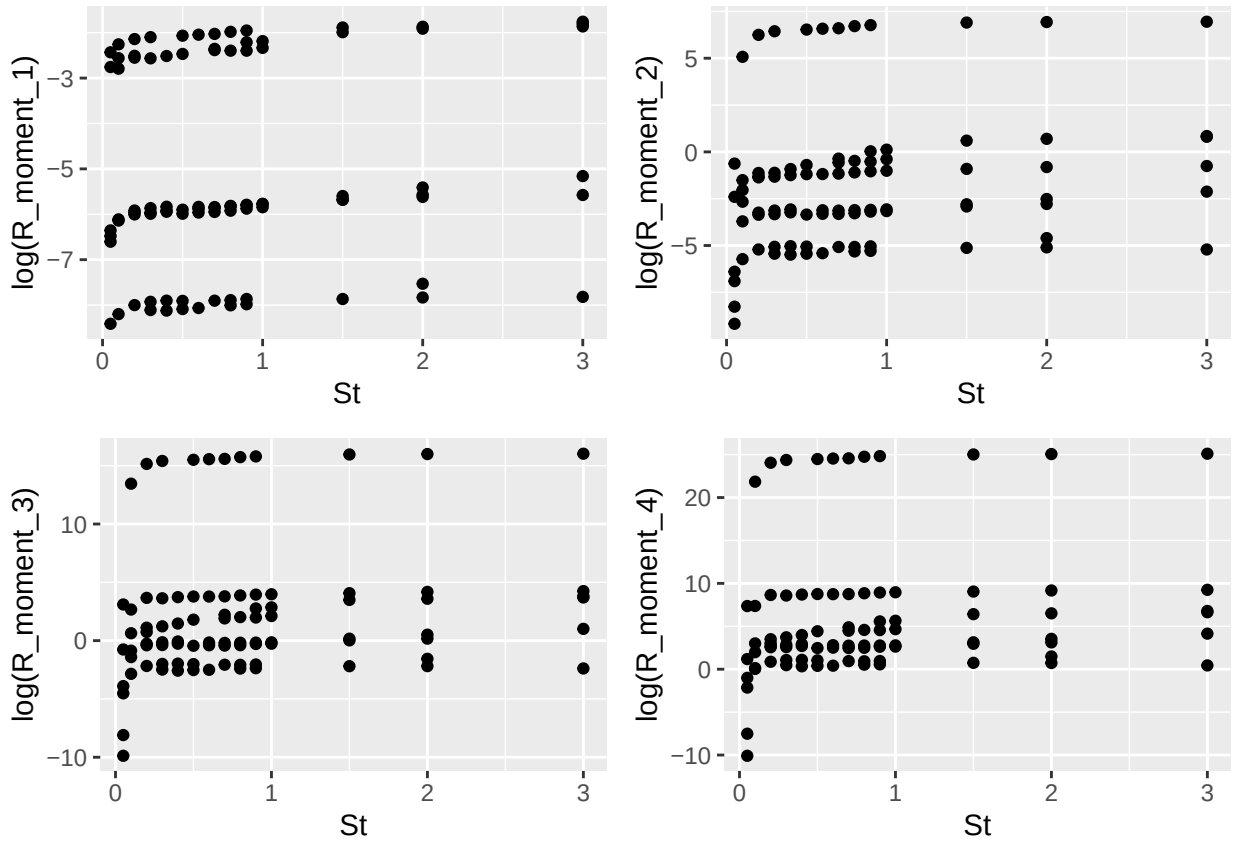


Performing a log transformation on these variables created more normally distributed variables. While not perfectly normal, this is a big improvement to the non-transformed variables. We will use the log version of variables and will reflect these transformed variables as such in our interpretations and analysis.

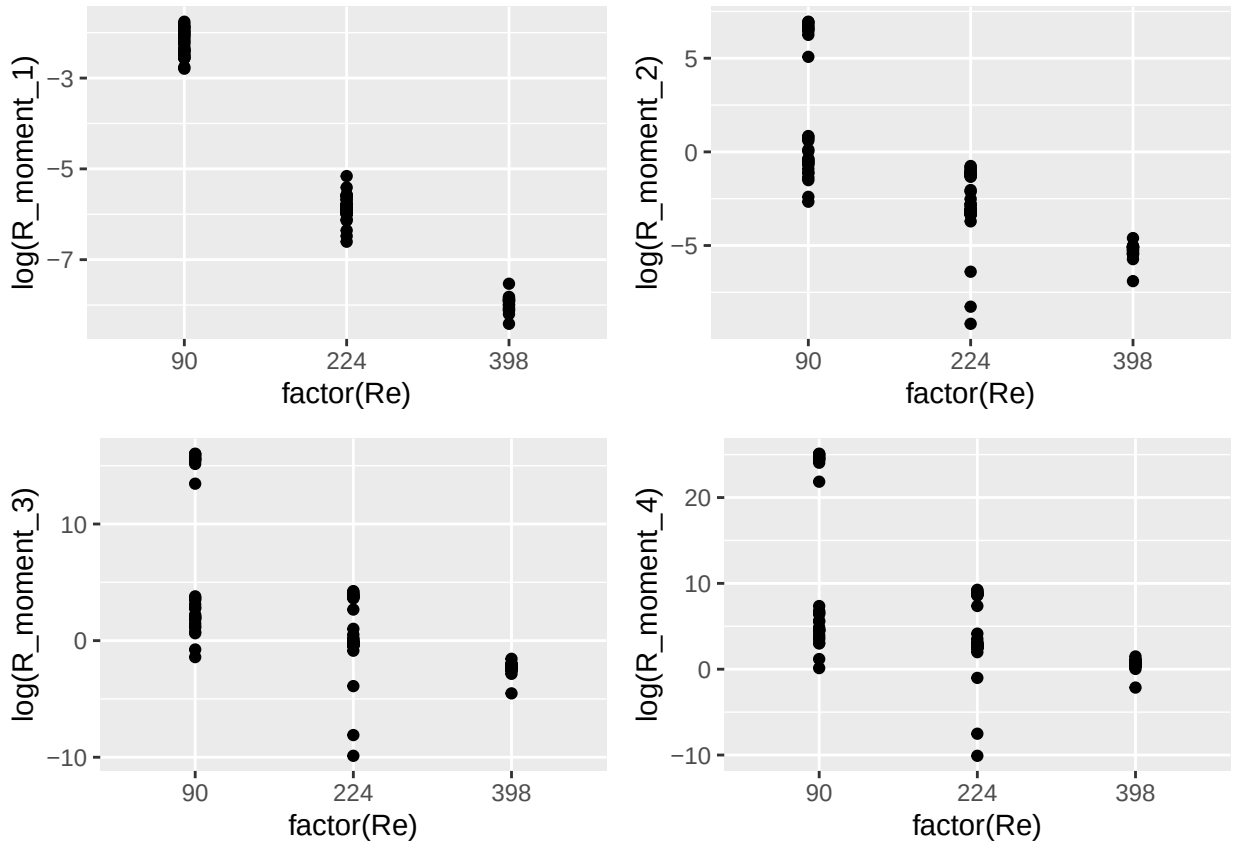
Another transformation to consider is to turn Fr and Re into ordered, categorical variables, since they each only have 2 or 3 unique values.

(TODO: Figure out how to handle infinite for Fr)

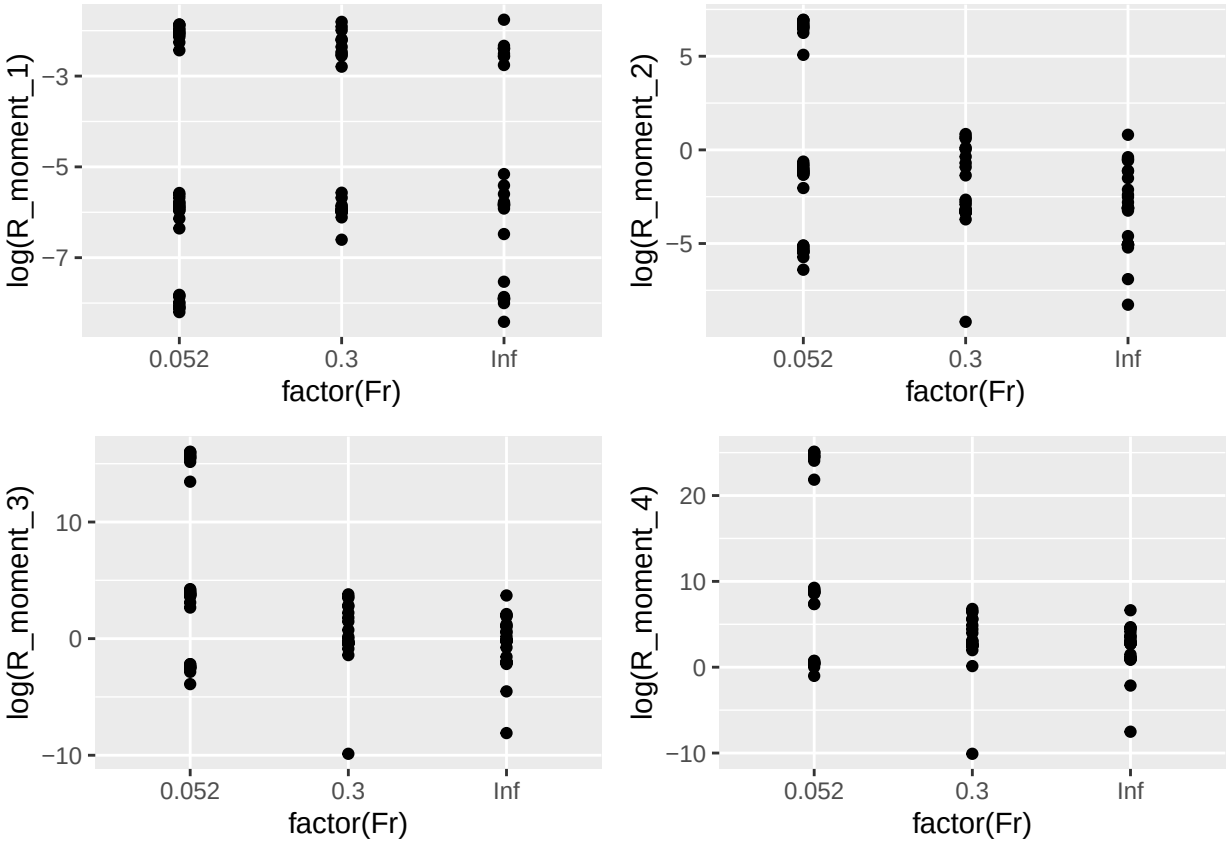
```
p1 <- ggplot(data=data_train, aes(x = St, y=log(R_moment_1))) +
  geom_point()

p2 <- ggplot(data=data_train, aes(x = St, y=log(R_moment_2))) +
  geom_point()

p3 <- ggplot(data=data_train, aes(x = St, y=log(R_moment_3))) +
  geom_point()

p4 <- ggplot(data=data_train, aes(x = St, y=log(R_moment_4))) +
  geom_point()

ggarrange(p1, p2, p3, p4)
```



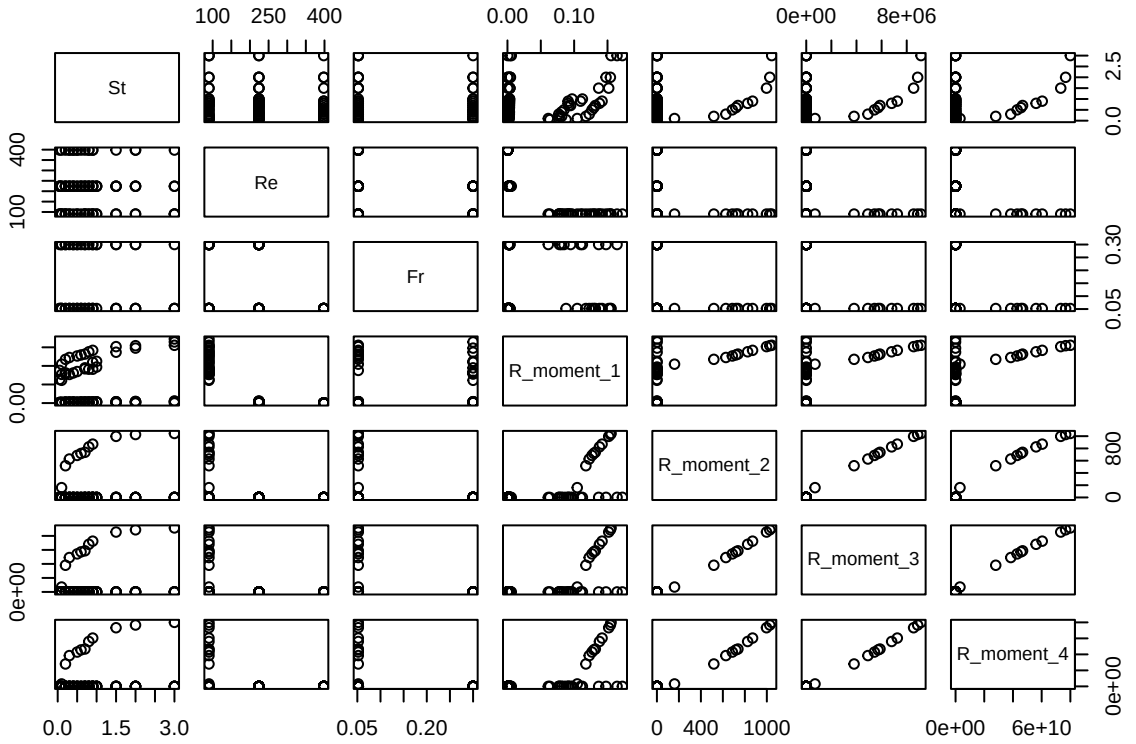

```
p5 <- ggplot(data=data_train, aes(x = factor(Re), y=log(R_moment_1))) +
  geom_point()

p6 <- ggplot(data=data_train, aes(x = factor(Re), y=log(R_moment_2))) +
  geom_point()

p7 <- ggplot(data=data_train, aes(x = factor(Re), y=log(R_moment_3))) +
  geom_point()

p8 <- ggplot(data=data_train, aes(x = factor(Re), y=log(R_moment_4))) +
  geom_point()

ggarrange(p5, p6, p7, p8)
```




```
p9 <- ggplot(data=data_train, aes(x = factor(Fr), y=log(R_moment_1))) +
  geom_point()

p10 <- ggplot(data=data_train, aes(x = factor(Fr), y=log(R_moment_2))) +
  geom_point()

p11 <- ggplot(data=data_train, aes(x = factor(Fr), y=log(R_moment_3))) +
  geom_point()

p12 <- ggplot(data=data_train, aes(x = factor(Fr), y=log(R_moment_4))) +
  geom_point()

ggarrange(p9, p10, p11, p12)
```




```
pairs(data_train)
```



It appears that each R_moment variable has somewhat of a linear relationship with St. Thus, we may want to begin our search for a best model by fitting a linear model.

Modeling

We will fit a basic linear model onto each log-transformed response variable.

```
model1 <- lm(log(R_moment_1) ~ St + factor(Re) + factor(Fr), data=data_train)
summary(model1)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ St + factor(Re) + factor(Fr),
##     data = data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.47532 -0.07168  0.02101  0.10237  0.23554
##
## Coefficients:
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   -2.40825    0.04137  -58.218   <2e-16 ***
## St             0.24652    0.02165   11.386   <2e-16 ***
## factor(Re)224 -3.62590    0.03836  -94.517   <2e-16 ***
## factor(Re)398 -5.75678    0.04826 -119.287   <2e-16 ***
```

```
## factor(Fr)0.3 -0.10770    0.04422   -2.435    0.017 *
## factor(Fr)Inf -0.02584    0.03945   -0.655    0.514
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1593 on 83 degrees of freedom
## Multiple R-squared:  0.9952, Adjusted R-squared:  0.9949
## F-statistic:  3460 on 5 and 83 DF,  p-value: < 2.2e-16
```

```
model2 <- lm(log(R_moment_2) ~ St + factor(Re) + factor(Fr), data=data_train)
summary(model2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ St + factor(Re) + factor(Fr),
##     data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0075 -1.2112 -0.1009  1.1631  3.0215
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.2049     0.4690   6.833 1.29e-09 ***
## St              0.7167     0.2455   2.920  0.00451 **
## factor(Re)224  -4.6321     0.4350 -10.650  < 2e-16 ***
## factor(Re)398  -7.7930     0.5472 -14.242  < 2e-16 ***
## factor(Fr)0.3  -3.4422     0.5014  -6.865 1.12e-09 ***
## factor(Fr)Inf  -2.7650     0.4473  -6.182 2.27e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.806 on 83 degrees of freedom
## Multiple R-squared:  0.7768, Adjusted R-squared:  0.7633
## F-statistic: 57.76 on 5 and 83 DF,  p-value: < 2.2e-16
```

```
model3 <- lm(log(R_moment_3) ~ St + factor(Re) + factor(Fr), data=data_train)
summary(model3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ St + factor(Re) + factor(Fr),
##     data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.7282 -2.3839 -0.4306  2.1123  5.4634
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     9.6598     0.8341  11.581  < 2e-16 ***
## St              0.9452     0.4366   2.165   0.0332 *
## factor(Re)224  -5.8796     0.7735  -7.601 4.03e-11 ***
```

```
## factor(Re)398 -10.2176     0.9731 -10.500  < 2e-16 ***
## factor(Fr)0.3  -6.8055     0.8917  -7.632 3.50e-11 ***
## factor(Fr)Inf  -5.4848     0.7955  -6.895 9.77e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.211 on 83 degrees of freedom
## Multiple R-squared:  0.6983, Adjusted R-squared:  0.6802
## F-statistic: 38.43 on 5 and 83 DF,  p-value: < 2.2e-16
```

```
model4 <- lm(log(R_moment_4) ~ St + factor(Re) + factor(Fr), data=data_train)
summary(model4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ St + factor(Re) + factor(Fr),
##     data = data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.1076  -3.5768  -0.7964   3.0052   7.8067
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    16.2281     1.1836  13.711  < 2e-16 ***
## St              1.1304     0.6195   1.825   0.0716 .
## factor(Re)224  -7.1866     1.0977  -6.547 4.58e-09 ***
## factor(Re)398 -12.7305     1.3808  -9.219 2.38e-14 ***
## factor(Fr)0.3 -10.1437     1.2654  -8.016 6.04e-12 ***
## factor(Fr)Inf  -8.1791     1.1288  -7.246 2.02e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.557 on 83 degrees of freedom
## Multiple R-squared:  0.6716, Adjusted R-squared:  0.6518
## F-statistic: 33.95 on 5 and 83 DF,  p-value: < 2.2e-16
```

Exploring collinearity:

```
vif(model1)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## St         1.004871  1        1.002433
## factor(Re) 1.107716  2        1.025905
## factor(Fr) 1.109374  2        1.026289
```

```
vif(model2)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## St         1.004871  1        1.002433
## factor(Re) 1.107716  2        1.025905
## factor(Fr) 1.109374  2        1.026289
```

```
vif(model3)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## St         1.004871  1        1.002433
## factor(Re) 1.107716  2        1.025905
## factor(Fr) 1.109374  2        1.026289
```

```
vif(model4)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## St         1.004871  1        1.002433
## factor(Re) 1.107716  2        1.025905
## factor(Fr) 1.109374  2        1.026289
```

Including all interaction terms:

```
glm.full <- lm(cbind(log(R_moment_1), log(R_moment_2), log(R_moment_3), log(R_moment_4)) ~  (St + factor
summary(glm.full)
```

```
## Response log(R_moment_1) :
##
## Call:
## lm(formula = `log(R_moment_1)` ~ (St + factor(Re) + factor(Fr))^2,
##     data = data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.41510 -0.01331  0.01761  0.05940  0.13973
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -2.186457   0.044127 -49.549  < 2e-16 ***
## St                           0.152307   0.032368   4.705 1.11e-05 ***
## factor(Re)224               -3.854073   0.055437 -69.522  < 2e-16 ***
## factor(Re)398               -5.970943   0.066863 -89.301  < 2e-16 ***
## factor(Fr)0.3               -0.419887   0.065701  -6.391 1.20e-08 ***
## factor(Fr)Inf               -0.454654   0.059741  -7.610 6.11e-11 ***
## St:factor(Re)224             0.041342   0.035969   1.149 0.254002
## St:factor(Re)398            -0.005585   0.046504  -0.120 0.904722
## St:factor(Fr)0.3             0.165845   0.044159   3.756 0.000337 ***
## St:factor(Fr)Inf             0.146870   0.037025   3.967 0.000164 ***
## factor(Re)224:factor(Fr)0.3  0.252705   0.067863   3.724 0.000375 ***
## factor(Re)398:factor(Fr)0.3        NA         NA      NA       NA
## factor(Re)224:factor(Fr)Inf  0.392182   0.068754   5.704 2.13e-07 ***
## factor(Re)398:factor(Fr)Inf  0.494113   0.075178   6.573 5.54e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.116 on 76 degrees of freedom
## Multiple R-squared:  0.9977, Adjusted R-squared:  0.9973
## F-statistic:  2723 on 12 and 76 DF,  p-value: < 2.2e-16
##
```

```
##
## Response log(R_moment_2) :
##
## Call:
## lm(formula = `log(R_moment_2)` ~ (St + factor(Re) + factor(Fr))^2,
##     data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8344 -0.0069  0.2296  0.5224  1.0188
##
## Coefficients: (1 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   5.164989   0.470307  10.982  < 2e-16 ***
## St                            0.858695   0.344985   2.489    0.015 *
## factor(Re)224                -7.434512   0.590851 -12.583  < 2e-16 ***
## factor(Re)398               -10.787379   0.712633 -15.137  < 2e-16 ***
## factor(Fr)0.3                -6.678147   0.700244  -9.537 1.26e-14 ***
## factor(Fr)Inf                -6.737794   0.636727 -10.582  < 2e-16 ***
## St:factor(Re)224             -0.004091   0.383357  -0.011    0.992
## St:factor(Re)398             -0.593466   0.495640  -1.197    0.235
## St:factor(Fr)0.3              0.250783   0.470653   0.533    0.596
## St:factor(Fr)Inf              0.112392   0.394615   0.285    0.777
## factor(Re)224:factor(Fr)0.3   4.477795   0.723295   6.191 2.81e-08 ***
## factor(Re)398:factor(Fr)0.3         NA         NA      NA       NA
## factor(Re)224:factor(Fr)Inf   4.694433   0.732788   6.406 1.13e-08 ***
## factor(Re)398:factor(Fr)Inf   6.883436   0.801251   8.591 8.12e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.237 on 76 degrees of freedom
## Multiple R-squared:  0.9041, Adjusted R-squared:  0.889
## F-statistic: 59.73 on 12 and 76 DF,  p-value: < 2.2e-16
##
##
## Response log(R_moment_3) :
##
## Call:
## lm(formula = `log(R_moment_3)` ~ (St + factor(Re) + factor(Fr))^2,
##     data = data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.2415   0.0321   0.3599   0.8096   1.7370
##
## Coefficients: (1 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   13.27626    0.76479  17.359  < 2e-16 ***
## St                             1.31188    0.56100   2.338    0.022 *
## factor(Re)224                -11.09434    0.96082 -11.547  < 2e-16 ***
## factor(Re)398                -16.02257    1.15885 -13.826  < 2e-16 ***
## factor(Fr)0.3                -12.80536    1.13871 -11.246  < 2e-16 ***
## factor(Fr)Inf                -12.80794    1.03542 -12.370  < 2e-16 ***
## St:factor(Re)224              -0.07617    0.62340  -0.122    0.903
```

```
## St:factor(Re)398             -1.01438    0.80599  -1.259    0.212
## St:factor(Fr)0.3              0.29860    0.76536   0.390    0.698
## St:factor(Fr)Inf              0.06435    0.64171   0.100    0.920
## factor(Re)224:factor(Fr)0.3   8.49426    1.17619   7.222 3.35e-10 ***
## factor(Re)398:factor(Fr)0.3        NA         NA      NA       NA
## factor(Re)224:factor(Fr)Inf   8.74071    1.19163   7.335 2.04e-10 ***
## factor(Re)398:factor(Fr)Inf  13.04934    1.30296  10.015 1.55e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.011 on 76 degrees of freedom
## Multiple R-squared:  0.8917, Adjusted R-squared:  0.8746
## F-statistic: 52.14 on 12 and 76 DF,  p-value: < 2.2e-16
##
##
## Response log(R_moment_4) :
##
## Call:
## lm(formula = `log(R_moment_4)` ~ (St + factor(Re) + factor(Fr))^2,
##     data = data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.2051   0.0767   0.4981   1.0761   2.3826
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 21.476884   1.024275  20.968  < 2e-16 ***
## St                           1.714788   0.751338   2.282   0.0253 *
## factor(Re)224              -14.780292   1.286808 -11.486  < 2e-16 ***
## factor(Re)398              -21.349343   1.552033 -13.756  < 2e-16 ***
## factor(Fr)0.3              -18.836080   1.525053 -12.351  < 2e-16 ***
## factor(Fr)Inf              -18.790108   1.386720 -13.550  < 2e-16 ***
## St:factor(Re)224            -0.138437   0.834909  -0.166   0.8687
## St:factor(Re)398            -1.381788   1.079449  -1.280   0.2044
## St:factor(Fr)0.3             0.320628   1.025029   0.313   0.7553
## St:factor(Fr)Inf             0.006509   0.859426   0.008   0.9940
## factor(Re)224:factor(Fr)0.3 12.435539   1.575254   7.894 1.75e-11 ***
## factor(Re)398:factor(Fr)0.3        NA         NA      NA       NA
## factor(Re)224:factor(Fr)Inf 12.719148   1.595930   7.970 1.26e-11 ***
## factor(Re)398:factor(Fr)Inf 19.134575   1.745034  10.965  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.693 on 76 degrees of freedom
## Multiple R-squared:  0.895,  Adjusted R-squared:  0.8784
## F-statistic: 53.96 on 12 and 76 DF,  p-value: < 2.2e-16
```

Re and Fr seem to have significant interaction for all moments, while St and Re only have significant interaction for the first moment.

A model with the interaction term for Re and Fr:

```
glm.interaction <- lm(cbind(log(R_moment_1), log(R_moment_2), log(R_moment_3), log(R_moment_4)) ~  (St +
# summary(glm.interaction)
glm.interaction
```

```
##
## Call:
## lm(formula = cbind(log(R_moment_1), log(R_moment_2), log(R_moment_3),
##     log(R_moment_4)) ~ (St + factor(Re) + factor(Fr) + factor(Re) *
##     factor(Fr)), data = data_train)
##
## Coefficients:
##                               [,1]      [,2]      [,3]      [,4]
## (Intercept)                 -2.2731    5.1869   13.3986   21.6950
## St                           0.2499    0.8340    1.1740    1.4690
## factor(Re)224               -3.8159   -7.4387  -11.1636  -14.9060
## factor(Re)398               -5.9885  -11.3837  -17.0302  -22.7148
## factor(Fr)0.3               -0.2630   -6.4163  -12.4781  -18.4708
## factor(Fr)Inf               -0.3294   -6.6523  -12.7719  -18.8106
## factor(Re)224:factor(Fr)0.3  0.2205    4.3872    8.3648   12.2758
## factor(Re)398:factor(Fr)0.3      NA        NA        NA        NA
## factor(Re)224:factor(Fr)Inf  0.4019    4.7181    8.7718   12.7559
## factor(Re)398:factor(Fr)Inf  0.5015    7.0758   13.3707   19.5683
```

Adding the interaction term between Re and Fr improved the fit of the model according to the adjusted R^2 values.

```
# forward.model <- regsubsets(log(R_moment_1) ~ St + factor(Re) + factor(Fr),
#                             data = data_train)
#
# summary(forward.model)
# names(forward.model)
# forward.model$rss
```

```
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:car':
##
##     logit
```

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'lattice'

## The following object is masked from 'package:boot':
##
##     melanoma
```

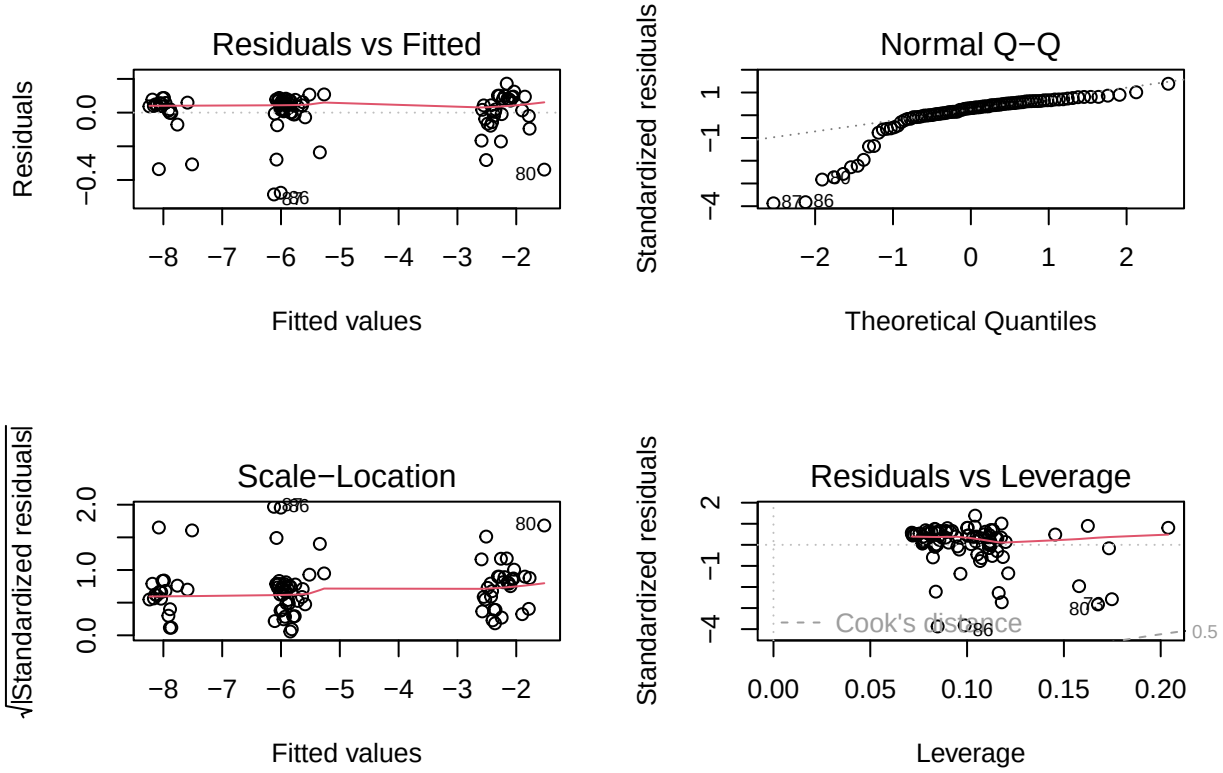
```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
#  + factor(Fr)*St + St*factor(Re)

glm.interaction1 <- lm(log(R_moment_1) ~  St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr), data=da
glm.interaction2 <- lm(log(R_moment_2) ~  St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr), data=da
glm.interaction3 <- lm(log(R_moment_3) ~  St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr), data=da
glm.interaction4 <- lm(log(R_moment_4) ~  St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr), data=da
#
# set.seed(325)
#
# # define training control which
# # generates parameters that further
# # control how models are created
# train_control <- trainControl(method = "cv",
#                               number = 10)
#
# model <- train(log(R_moment_1)~., data = data_train,
#                trControl = train_control,
#                method = "lm")

confint(glm.interaction1)
```

```
##                                   2.5 %     97.5 %
## (Intercept)                 -2.35486531 -2.1912624
## St                           0.21402088  0.2857638
## factor(Re)224               -3.91856830 -3.7131839
## factor(Re)398               -6.10038960 -5.8766862
## factor(Fr)0.3               -0.37485503 -0.1510907
## factor(Fr)Inf               -0.44460368 -0.2142777
## factor(Re)224:factor(Fr)0.3  0.06976511  0.3712336
## factor(Re)398:factor(Fr)0.3          NA         NA
## factor(Re)224:factor(Fr)Inf  0.24743728  0.5562646
## factor(Re)398:factor(Fr)Inf  0.33502256  0.6679905
```

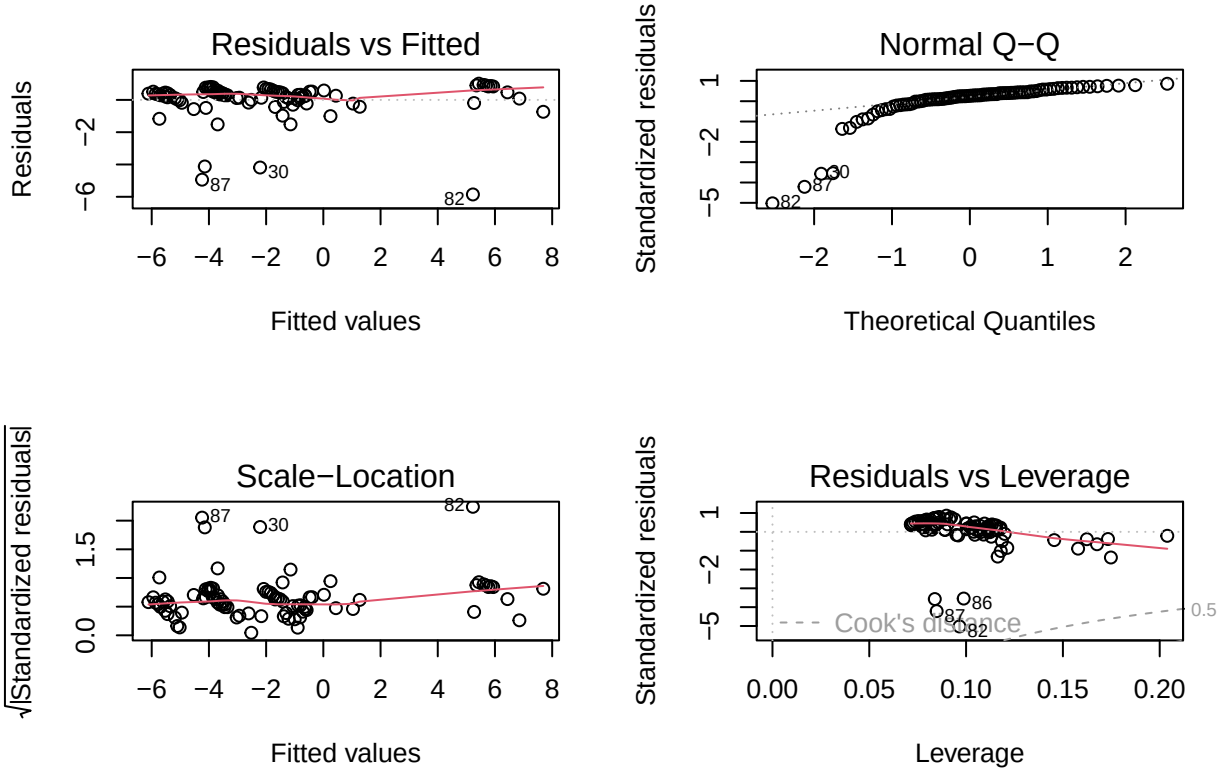
```
par(mfrow = c(2, 2))
plot(glm.interaction1)
```




```
confint(glm.interaction2)
```

```
##                                    2.5 %     97.5 %
## (Intercept)                    4.4222117   5.951662
## St                             0.4986196   1.169312
## factor(Re)224                 -8.3987188  -6.478672
## factor(Re)398                -12.4294010 -10.338098
## factor(Fr)0.3                 -7.4622526  -5.370380
## factor(Fr)Inf                 -7.7289370  -5.575723
## factor(Re)224:factor(Fr)0.3    2.9780047   5.796299
## factor(Re)398:factor(Fr)0.3           NA         NA
## factor(Re)224:factor(Fr)Inf    3.2745944   6.161682
## factor(Re)398:factor(Fr)Inf    5.5193875   8.632156
```

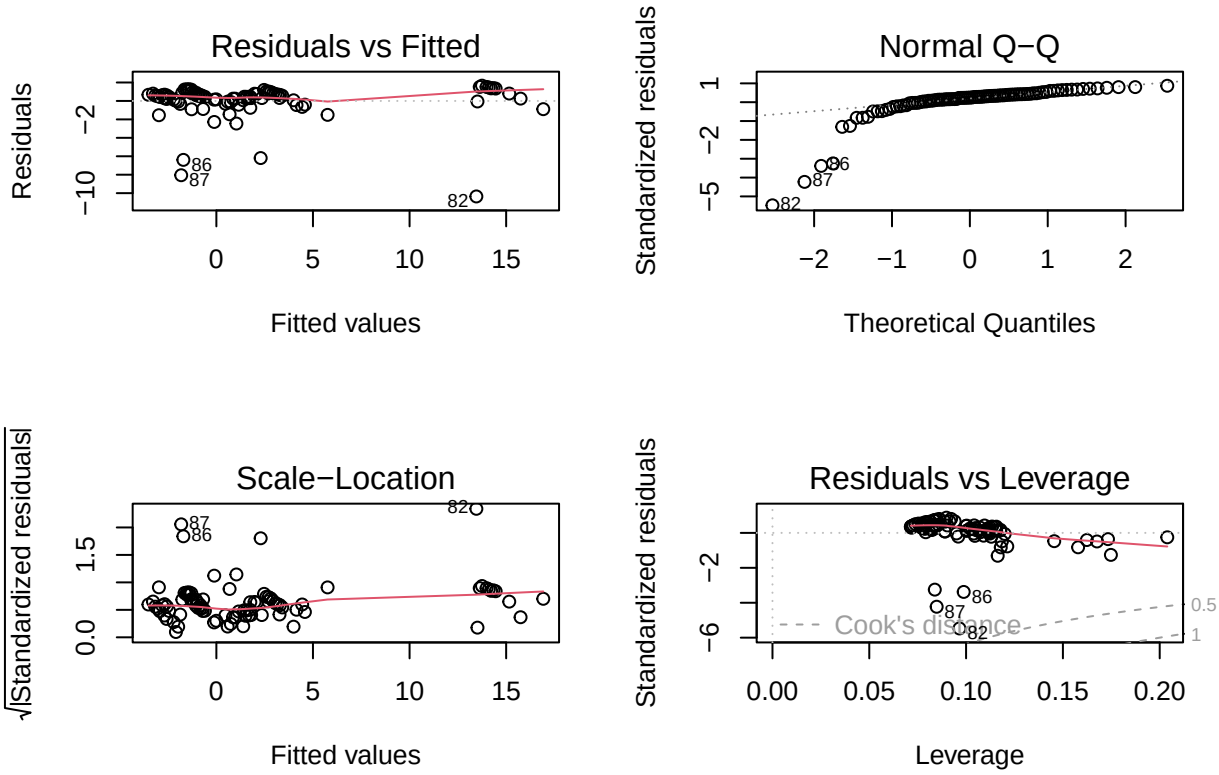
```
par(mfrow = c(2, 2))
plot(glm.interaction2)
```




```
confint(glm.interaction3)
```

```
##                                    2.5 %     97.5 %
## (Intercept)                   12.156659  14.640638
## St                             0.629346   1.718617
## factor(Re)224                -12.722766  -9.604419
## factor(Re)398                -18.728444 -15.331960
## factor(Fr)0.3                -14.176855 -10.779445
## factor(Fr)Inf                -14.520411 -11.023376
## factor(Re)224:factor(Fr)0.3    6.076172  10.653362
## factor(Re)398:factor(Fr)0.3          NA         NA
## factor(Re)224:factor(Fr)Inf    6.427375  11.116294
## factor(Re)398:factor(Fr)Inf   10.842930  15.898376
```

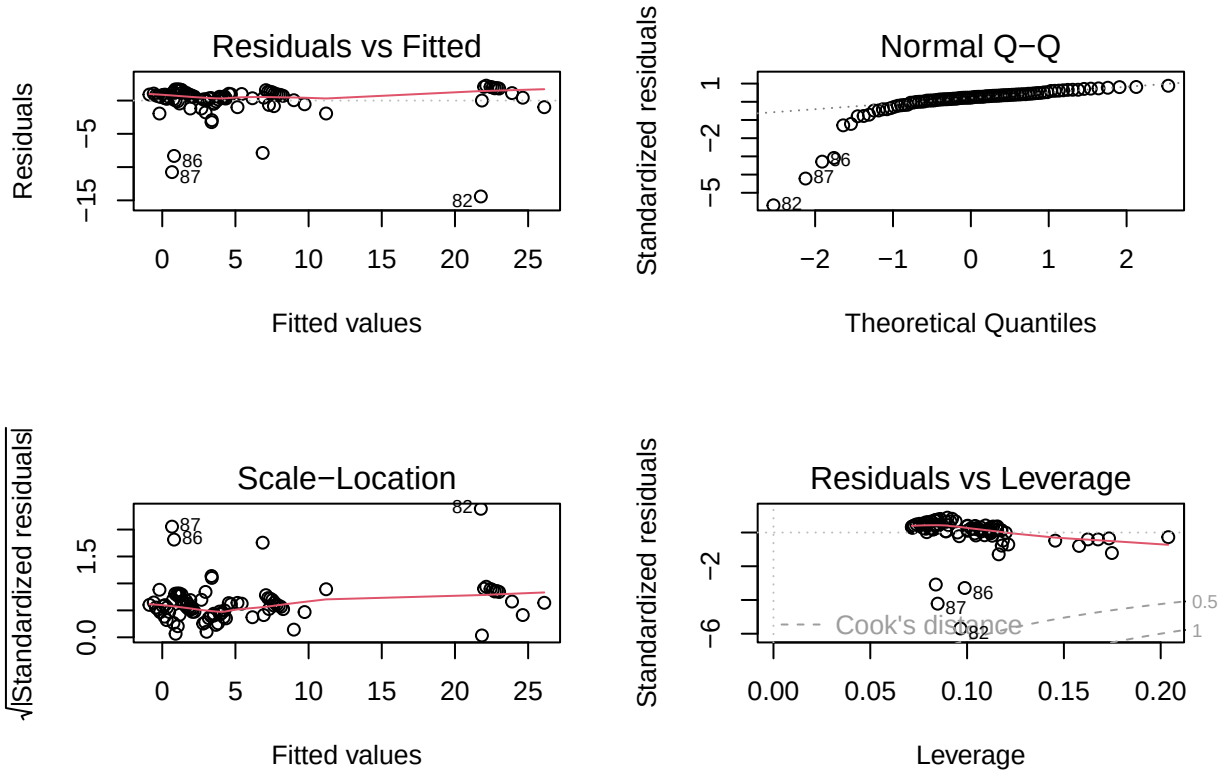
```
par(mfrow = c(2, 2))
plot(glm.interaction3)
```




```
confint(glm.interaction4)
```

```
##                                    2.5 %     97.5 %
## (Intercept)                   20.032580  23.357400
## St                             0.740037   2.198032
## factor(Re)224                -16.992992 -12.819067
## factor(Re)398                -24.987951 -20.441738
## factor(Fr)0.3                -20.744539 -16.197088
## factor(Fr)Inf                -21.151008 -16.470208
## factor(Re)224:factor(Fr)0.3    9.212522  15.339116
## factor(Re)398:factor(Fr)0.3          NA         NA
## factor(Re)224:factor(Fr)Inf    9.617818  15.893960
## factor(Re)398:factor(Fr)Inf   16.184908  22.951649
```

```
par(mfrow = c(2, 2))
plot(glm.interaction4)
```



Split data into training and test sets

```
attach(data_train)
set.seed(3)
train_ind <- sample(x = nrow(data_train), size = 0.8 * nrow(data_train))
test_ind_neg <- -train_ind
training <- data_train[train_ind, ]
testing <- data_train[test_ind_neg, ]
```

Linear model using least squares & no interaction term

```
fit.lm1 <- lm(log(R_moment_1) ~ (St + factor(Re) + factor(Fr)), data = training)
pred.lm1 <- predict(fit.lm1, testing)
mse_test1 <- mean((pred.lm1 - log(testing$R_moment_1))^2)
fit.lm2 <- lm(log(R_moment_2) ~ (St + factor(Re) + factor(Fr)), data = training)
pred.lm2 <- predict(fit.lm2, testing)
mse_test2 <- mean((pred.lm2 - log(testing$R_moment_2))^2)
fit.lm3 <- lm(log(R_moment_3) ~ (St + factor(Re) + factor(Fr)), data = training)
pred.lm3 <- predict(fit.lm3, testing)
mse_test3 <- mean((pred.lm3 - log(testing$R_moment_3))^2)
```

```
fit.lm4 <- lm(log(R_moment_4) ~ (St + factor(Re) + factor(Fr)), data = training)
pred.lm4 <- predict(fit.lm4, testing)
mse_test4 <- mean((pred.lm4 - log(testing$R_moment_4))^2)
mse_test1
```

```
## [1] 0.01787931
```

```
mse_test2
```

```
## [1] 3.4922
```

```
mse_test3
```

```
## [1] 10.6892
```

```
mse_test4
```

```
## [1] 21.32186
```

Linear model using least squares & interaction term

```
fit.lm1 <- lm(log(R_moment_1) ~ (St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr)), data = training
pred.lm1 <- predict(fit.lm1, testing)
```

```
## Warning in predict.lm(fit.lm1, testing): prediction from a rank-deficient fit
## may be misleading
```

```
mse_test1 <- mean((pred.lm1 - log(testing$R_moment_1))^2)
fit.lm2 <- lm(log(R_moment_2) ~ (St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr)), data = training
pred.lm2 <- predict(fit.lm2, testing)
```

```
## Warning in predict.lm(fit.lm2, testing): prediction from a rank-deficient fit
## may be misleading
```

```
mse_test2 <- mean((pred.lm2 - log(testing$R_moment_2))^2)
fit.lm3 <- lm(log(R_moment_3) ~ (St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr)), data = training
pred.lm3 <- predict(fit.lm3, testing)
```

```
## Warning in predict.lm(fit.lm3, testing): prediction from a rank-deficient fit
## may be misleading
```

```
mse_test3 <- mean((pred.lm3 - log(testing$R_moment_3))^2)
fit.lm4 <- lm(log(R_moment_4) ~ (St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr)), data = training
pred.lm4 <- predict(fit.lm4, testing)
```

```
## Warning in predict.lm(fit.lm4, testing): prediction from a rank-deficient fit
## may be misleading
```

```
mse_test4 <- mean((pred.lm4 - log(testing$R_moment_4))^2)
mse_test1
```

```
## [1] 0.008822464
```

```
mse_test2
```

```
## [1] 1.396723
```

```
mse_test3
```

```
## [1] 3.184988
```

```
mse_test4
```

```
## [1] 5.272393
```

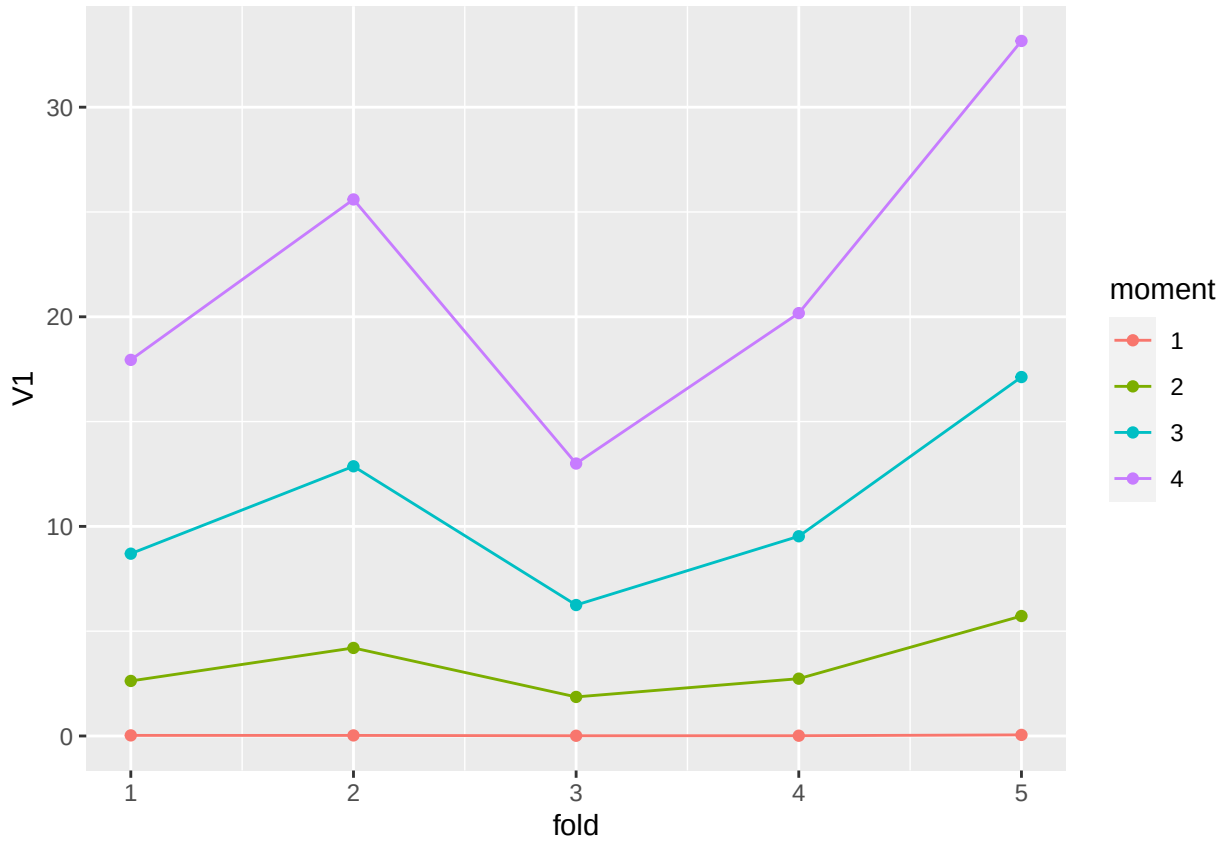
Having an interaction term significantly improved the test MSEs of the linear model.

```
#Create 5 equally size folds
set.seed(325)
folds <- cut(seq(1,nrow(data_train)),breaks=5,labels=FALSE)

test_mses_noint <- list()
#Perform 5 fold cross validation
for(i in 1:5){
    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData <- data_train[testIndexes, ]
    trainData <- data_train[-testIndexes, ]

    fit.lm1 <- lm(log(R_moment_1) ~ (St + factor(Re) + factor(Fr)), data = trainData)
    pred.lm1 <- predict(fit.lm1, testData)
    mse_test1 <- mean((pred.lm1 - log(testData$R_moment_1))^2)
    fit.lm2 <- lm(log(R_moment_2) ~ (St + factor(Re) + factor(Fr)), data = trainData)
    pred.lm2 <- predict(fit.lm2, testData)
    mse_test2 <- mean((pred.lm2 - log(testData$R_moment_2))^2)
    fit.lm3 <- lm(log(R_moment_3) ~ (St + factor(Re) + factor(Fr)), data = trainData)
    pred.lm3 <- predict(fit.lm3, testData)
    mse_test3 <- mean((pred.lm3 - log(testData$R_moment_3))^2)
    fit.lm4 <- lm(log(R_moment_4) ~ (St + factor(Re) + factor(Fr)), data = trainData)
    pred.lm4 <- predict(fit.lm4, testData)
    mse_test4 <- mean((pred.lm4 - log(testData$R_moment_4))^2)
    mses = list(mse_test1, mse_test2, mse_test3, mse_test4)
    test_mses_noint <- append(test_mses_noint, mses)
}
```

```
df <- as.data.frame(do.call(rbind, test_mses_noint))
df$moment <- as.factor(c(1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4))
df$fold <- c(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5)
df
```

```
##             V1 moment fold
## 1   0.02837571      1    1
## 2   2.62630196      2    1
## 3   8.69862029      3    1
## 4  17.94533166      4    1
## 5   0.02718505      1    2
## 6   4.20105845      2    2
## 7  12.86688080      3    2
## 8  25.59910498      4    2
## 9   0.01151457      1    3
## 10  1.86361447      2    3
## 11  6.24776790      3    3
## 12 12.99797132      4    3
## 13  0.01332755      1    4
## 14  2.73487897      2    4
## 15  9.52874477      3    4
## 16 20.17487144      4    4
## 17  0.05207977      1    5
## 18  5.72342365      2    5
## 19 17.12609853      3    5
## 20 33.16084024      4    5
```

```
ggplot(df, aes(x = fold, y = V1, color = moment, group = moment)) +
  geom_point() +
  geom_line()
```



```r
#Create 5 equally size folds
set.seed(325)
folds <- cut(seq(1,nrow(data_train)),breaks=5,labels=FALSE)

test_mses <- list()
#Perform 5 fold cross validation
for(i in 1:5){
    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData <- data_train[testIndexes, ]
    trainData <- data_train[-testIndexes, ]

    fit.lm1 <- lm(log(R_moment_1) ~ (St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr)), data = trai
    pred.lm1 <- predict(fit.lm1, testData)
    mse_test1 <- mean((pred.lm1 - log(testData$R_moment_1))^2)
    fit.lm2 <- lm(log(R_moment_2) ~ (St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr)), data = trai
    pred.lm2 <- predict(fit.lm2, testData)
    mse_test2 <- mean((pred.lm2 - log(testData$R_moment_2))^2)
    fit.lm3 <- lm(log(R_moment_3) ~ (St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr)), data = trai
    pred.lm3 <- predict(fit.lm3, testData)
    mse_test3 <- mean((pred.lm3 - log(testData$R_moment_3))^2)
    fit.lm4 <- lm(log(R_moment_4) ~ (St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr)), data = trai
    pred.lm4 <- predict(fit.lm4, testData)
    mse_test4 <- mean((pred.lm4 - log(testData$R_moment_4))^2)
    mses = list(mse_test1, mse_test2, mse_test3, mse_test4)
    test_mses <- append(test_mses, mses)
}
```

```
## Warning in predict.lm(fit.lm1, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm2, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm3, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm4, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm1, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm2, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm3, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm4, testData): prediction from a rank-deficient fit
## may be misleading
```

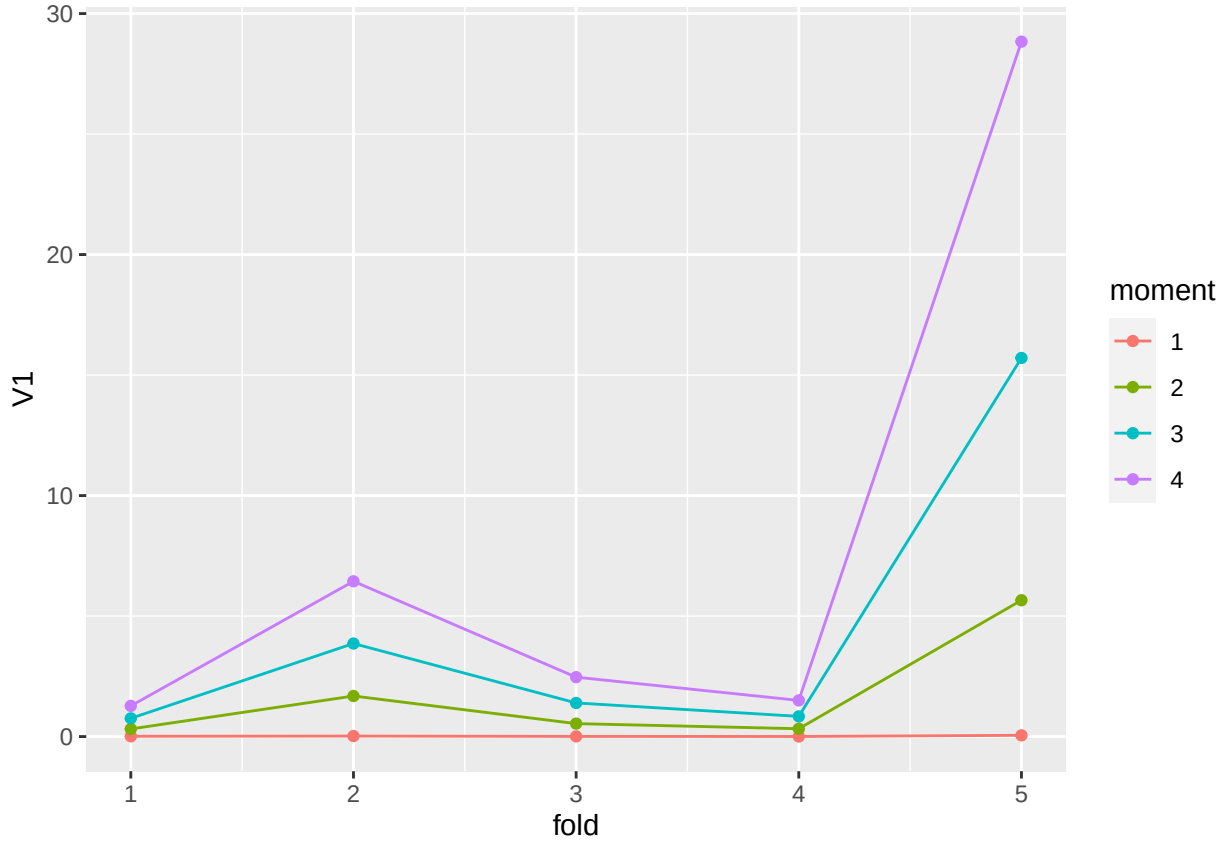
```
## Warning in predict.lm(fit.lm1, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm2, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm3, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm4, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm1, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm2, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm3, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm4, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm1, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm2, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm3, testData): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(fit.lm4, testData): prediction from a rank-deficient fit
## may be misleading
```

```
df <- as.data.frame(do.call(rbind, test_mses))
df$moment <- as.factor(c(1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4))
df$fold <- c(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5)
df
```

```
##              V1 moment fold
## 1   0.012048464      1    1
## 2   0.317081416      2    1
## 3   0.756757577      3    1
## 4   1.272977077      4    1
## 5   0.021399623      1    2
## 6   1.683630868      2    2
## 7   3.861472463      3    2
## 8   6.444438050      4    2
## 9   0.007774591      1    3
```

```
## 10  0.538208184      2    3
## 11  1.394950000      3    3
## 12  2.465178473      4    3
## 13  0.004395870      1    4
## 14  0.325188200      2    4
## 15  0.838788055      3    4
## 16  1.501354352      4    4
## 17  0.051681422      1    5
## 18  5.657287934      2    5
## 19 15.709810170      3    5
## 20 28.834876841      4    5
```

```
ggplot(df, aes(x = fold, y = V1, color = moment, group = moment)) +
  geom_point() +
  geom_line()
```




```
data_ctrl <- trainControl(method = "cv", number = 5)

model_caret <- train(log(R_moment_1) ~ (St + factor(Re) + factor(Fr) + factor(Re)*factor(Fr)),   # mode
                     data = trainData,
                     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
##
## 71 samples
##  3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 59, 55, 57, 55, 58
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   0.09788092  0.9978884  0.07180903
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Polynomial Regression

For each of the four moments, we try to fit a polynomial model based on the degree of the numerical variable, St. We also include the other two factored variables in each model.

First moment:

```
polym1 <- lm(log(R_moment_1) ~ poly(St, 2) + factor(Re) + factor(Fr), data = training)
summary(polym1)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ poly(St, 2) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33265 -0.06610  0.00707  0.07555  0.34096
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
```

```
## (Intercept)   -2.16637    0.03629  -59.702  < 2e-16 ***
## poly(St, 2)1   1.58953    0.14605   10.884 3.38e-16 ***
## poly(St, 2)2  -0.68385    0.14764   -4.632 1.83e-05 ***
## factor(Re)224 -3.64684    0.03907  -93.344  < 2e-16 ***
## factor(Re)398 -5.79385    0.05059 -114.522  < 2e-16 ***
## factor(Fr)0.3 -0.15091    0.04656   -3.241  0.00189 **
## factor(Fr)Inf -0.02023    0.03967   -0.510  0.61171
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1448 on 64 degrees of freedom
## Multiple R-squared:  0.996,  Adjusted R-squared:  0.9957
## F-statistic:  2677 on 6 and 64 DF,  p-value: < 2.2e-16
```

```
poly2m1 <- lm(log(R_moment_1) ~ poly(St, 3) + factor(Re) + factor(Fr), data = training)
summary(poly2m1)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ poly(St, 3) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.31477 -0.07409 -0.00261  0.09929  0.30537
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   -2.16241    0.03558  -60.781  < 2e-16 ***
## poly(St, 3)1   1.58984    0.14296   11.121  < 2e-16 ***
## poly(St, 3)2  -0.68415    0.14452   -4.734 1.29e-05 ***
## poly(St, 3)3   0.28014    0.14378    1.948  0.05583 .
## factor(Re)224 -3.65344    0.03839  -95.161  < 2e-16 ***
## factor(Re)398 -5.80718    0.04999 -116.163  < 2e-16 ***
## factor(Fr)0.3 -0.14556    0.04565   -3.188  0.00223 **
## factor(Fr)Inf -0.01874    0.03883   -0.482  0.63117
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1418 on 63 degrees of freedom
## Multiple R-squared:  0.9963, Adjusted R-squared:  0.9958
## F-statistic:  2395 on 7 and 63 DF,  p-value: < 2.2e-16
```

```
poly3m1 <- lm(log(R_moment_1) ~ poly(St, 4) + factor(Re) + factor(Fr), data = training)
summary(poly3m1)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ poly(St, 4) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -0.29739 -0.06934 -0.01242  0.08023  0.31329
##
## Coefficients:
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   -2.157680   0.032765  -65.852  < 2e-16 ***
## poly(St, 4)1   1.594005   0.131556   12.117  < 2e-16 ***
## poly(St, 4)2  -0.693114   0.133013   -5.211 2.28e-06 ***
## poly(St, 4)3   0.294630   0.132369    2.226 0.029671 *
## poly(St, 4)4  -0.470535   0.133620   -3.521 0.000811 ***
## factor(Re)224 -3.667027   0.035538 -103.185  < 2e-16 ***
## factor(Re)398 -5.836580   0.046754 -124.837  < 2e-16 ***
## factor(Fr)0.3 -0.137911   0.042067   -3.278 0.001714 **
## factor(Fr)Inf -0.002662   0.036025   -0.074 0.941342
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1304 on 62 degrees of freedom
## Multiple R-squared:  0.9969, Adjusted R-squared:  0.9965
## F-statistic:  2476 on 8 and 62 DF,  p-value: < 2.2e-16
```

```
poly4m1 <- lm(log(R_moment_1) ~ poly(St, 5) + factor(Re) + factor(Fr), data = training)
summary(poly4m1)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ poly(St, 5) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.315418 -0.067990 -0.005822  0.075227  0.307888
##
## Coefficients:
##                 Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   -2.1666151  0.0312280  -69.380  < 2e-16 ***
## poly(St, 5)1   1.5983470  0.1247464   12.813  < 2e-16 ***
## poly(St, 5)2  -0.6871726  0.1261360   -5.448 9.67e-07 ***
## poly(St, 5)3   0.2913410  0.1255131    2.321 0.023634 *
## poly(St, 5)4  -0.4660782  0.1267042   -3.678 0.000498 ***
## poly(St, 5)5   0.3524325  0.1248844    2.822 0.006433 **
## factor(Re)224 -3.6617252  0.0337487 -108.500  < 2e-16 ***
## factor(Re)398 -5.8223510  0.0446161 -130.499  < 2e-16 ***
## factor(Fr)0.3 -0.1266630  0.0400856   -3.160 0.002457 **
## factor(Fr)Inf  0.0005497  0.0341770    0.016 0.987219
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1237 on 61 degrees of freedom
## Multiple R-squared:  0.9972, Adjusted R-squared:  0.9968
## F-statistic:  2449 on 9 and 61 DF,  p-value: < 2.2e-16
```

```
poly5m1 <- lm(log(R_moment_1) ~ poly(St, 6) + factor(Re) + factor(Fr), data = training)
summary(poly5m1)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ poly(St, 6) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32045 -0.06636 -0.00751  0.07647  0.30827
##
## Coefficients:
##                 Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   -2.1675450  0.0314175  -68.992  < 2e-16 ***
## poly(St, 6)1   1.5979933  0.1253666   12.747  < 2e-16 ***
## poly(St, 6)2  -0.6862440  0.1267703   -5.413 1.14e-06 ***
## poly(St, 6)3   0.2895551  0.1261676    2.295  0.02525 *
## poly(St, 6)4  -0.4636290  0.1273919   -3.639  0.00057 ***
## poly(St, 6)5   0.3531189  0.1255088    2.814  0.00662 **
## poly(St, 6)6  -0.0793434  0.1255883   -0.632  0.52993
## factor(Re)224 -3.6596637  0.0340727 -107.407  < 2e-16 ***
## factor(Re)398 -5.8189474  0.0451600 -128.852  < 2e-16 ***
## factor(Fr)0.3 -0.1276059  0.0403121   -3.165  0.00243 **
## factor(Fr)Inf -0.0008333  0.0344162   -0.024  0.98076
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1243 on 60 degrees of freedom
## Multiple R-squared:  0.9973, Adjusted R-squared:  0.9968
## F-statistic:  2183 on 10 and 60 DF,  p-value: < 2.2e-16
```

```
poly6m1 <- lm(log(R_moment_1) ~ poly(St, 7) + factor(Re) + factor(Fr), data = training)
summary(poly6m1)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ poly(St, 7) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.314045 -0.057152 -0.008371  0.071402  0.306723
##
## Coefficients:
##                 Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   -2.169020   0.031474  -68.915  < 2e-16 ***
## poly(St, 7)1   1.599136   0.125448   12.747  < 2e-16 ***
## poly(St, 7)2  -0.687297   0.126852   -5.418 1.17e-06 ***
## poly(St, 7)3   0.288807   0.126246    2.288 0.025759 *
## poly(St, 7)4  -0.463958   0.127469   -3.640 0.000576 ***
## poly(St, 7)5   0.353746   0.125586    2.817 0.006588 **
## poly(St, 7)6  -0.079363   0.125664   -0.632 0.530121
## poly(St, 7)7   0.120345   0.124960    0.963 0.339446
## factor(Re)224 -3.659404   0.034094 -107.332  < 2e-16 ***
## factor(Re)398 -5.817566   0.045210 -128.679  < 2e-16 ***
## factor(Fr)0.3 -0.127109   0.040340   -3.151 0.002556 **
```

```
## factor(Fr)Inf  0.001922   0.034556    0.056 0.955824
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1244 on 59 degrees of freedom
## Multiple R-squared:  0.9973, Adjusted R-squared:  0.9968
## F-statistic:  1982 on 11 and 59 DF,  p-value: < 2.2e-16
```

```
poly7m1 <- lm(log(R_moment_1) ~ poly(St, 8) + factor(Re) + factor(Fr), data = training)
summary(poly7m1)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ poly(St, 8) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32396 -0.06248 -0.00834  0.07358  0.30616
##
## Coefficients:
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   -2.168615   0.031700  -68.411  < 2e-16 ***
## poly(St, 8)1   1.599062   0.126300   12.661  < 2e-16 ***
## poly(St, 8)2  -0.686162   0.127737   -5.372 1.44e-06 ***
## poly(St, 8)3   0.291206   0.127213    2.289 0.025731 *
## poly(St, 8)4  -0.466124   0.128423   -3.630 0.000601 ***
## poly(St, 8)5   0.353233   0.126444    2.794 0.007052 **
## poly(St, 8)6  -0.077942   0.126556   -0.616 0.540390
## poly(St, 8)7   0.119490   0.125823    0.950 0.346220
## poly(St, 8)8   0.059253   0.130219    0.455 0.650787
## factor(Re)224 -3.659571   0.034328 -106.607  < 2e-16 ***
## factor(Re)398 -5.821425   0.046300 -125.733  < 2e-16 ***
## factor(Fr)0.3 -0.124911   0.040900   -3.054 0.003406 **
## factor(Fr)Inf  0.001746   0.034792    0.050 0.960145
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1252 on 58 degrees of freedom
## Multiple R-squared:  0.9973, Adjusted R-squared:  0.9968
## F-statistic:  1793 on 12 and 58 DF,  p-value: < 2.2e-16
```

```
anova(fit.lm1, polym1, poly2m1, poly3m1, poly4m1, poly5m1, poly6m1, poly7m1)
```

```
## Analysis of Variance Table
##
## Model 1: log(R_moment_1) ~ (St + factor(Re) + factor(Fr) + factor(Re) *
##     factor(Fr))
## Model 2: log(R_moment_1) ~ poly(St, 2) + factor(Re) + factor(Fr)
## Model 3: log(R_moment_1) ~ poly(St, 3) + factor(Re) + factor(Fr)
## Model 4: log(R_moment_1) ~ poly(St, 4) + factor(Re) + factor(Fr)
## Model 5: log(R_moment_1) ~ poly(St, 5) + factor(Re) + factor(Fr)
## Model 6: log(R_moment_1) ~ poly(St, 6) + factor(Re) + factor(Fr)
```

```
## Model 7: log(R_moment_1) ~ poly(St, 7) + factor(Re) + factor(Fr)
## Model 8: log(R_moment_1) ~ poly(St, 8) + factor(Re) + factor(Fr)
##   Res.Df     RSS Df Sum of Sq       F    Pr(>F)
## 1     62 0.53580
## 2     64 1.34226 -2  -0.80646 25.7178 1.009e-08 ***
## 3     63 1.26598  1   0.07628  4.8654 0.0313772 *
## 4     62 1.05498  1   0.21100 13.4576 0.0005319 ***
## 5     61 0.93315  1   0.12183  7.7702 0.0071691 **
## 6     60 0.92698  1   0.00617  0.3933 0.5330320
## 7     59 0.91263  1   0.01435  0.9150 0.3427539
## 8     58 0.90939  1   0.00325  0.2071 0.6507871
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pred.polym1 <- predict(polym1, testing)
pred.poly2m1 <- predict(poly2m1, testing)
pred.poly3m1 <- predict(poly3m1, testing)
pred.poly4m1 <- predict(poly4m1, testing)
pred.poly5m1 <- predict(poly5m1, testing)
pred.poly6m1 <- predict(poly6m1, testing)
pred.poly7m1 <- predict(poly7m1, testing)
mse_polym1 <- mean((pred.polym1 - log(testing$R_moment_1))^2)
mse_poly2m1 <- mean((pred.poly2m1 - log(testing$R_moment_1))^2)
mse_poly3m1 <- mean((pred.poly3m1 - log(testing$R_moment_1))^2)
mse_poly4m1 <- mean((pred.poly4m1 - log(testing$R_moment_1))^2)
mse_poly5m1 <- mean((pred.poly5m1 - log(testing$R_moment_1))^2)
mse_poly6m1 <- mean((pred.poly6m1 - log(testing$R_moment_1))^2)
mse_poly7m1 <- mean((pred.poly7m1 - log(testing$R_moment_1))^2)
mse_polym1
```

```
## [1] 0.02244211
```

```
mse_poly2m1
```

```
## [1] 0.0250986
```

```
mse_poly3m1
```

```
## [1] 0.027301
```

```
mse_poly4m1
```

```
## [1] 0.02443886
```

```
mse_poly5m1
```

```
## [1] 0.02302476
```

```
mse_poly6m1
```

```
## [1] 0.02203137
```

```
mse_poly7m1
```

```
## [1] 0.02338871
```

Similar to least squares.

Second moment:

```
polym2 <- lm(log(R_moment_2) ~ poly(St, 2) + factor(Re) + factor(Fr), data = training)
summary(polym2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ poly(St, 2) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2423 -1.0935 -0.1559  1.2966  3.1722
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.1076     0.4209   9.758 2.74e-14 ***
## poly(St, 2)1    4.6783     1.6942   2.761  0.00750 **
## poly(St, 2)2   -5.4282     1.7128  -3.169  0.00234 **
## factor(Re)224  -4.8151     0.4532 -10.624 9.20e-16 ***
## factor(Re)398  -7.9846     0.5869 -13.605  < 2e-16 ***
## factor(Fr)0.3  -3.9073     0.5401  -7.235 7.20e-10 ***
## factor(Fr)Inf  -2.8244     0.4601  -6.138 5.87e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.68 on 64 degrees of freedom
## Multiple R-squared:  0.8171, Adjusted R-squared:    0.8
## F-statistic: 47.65 on 6 and 64 DF,  p-value: < 2.2e-16
```

```
poly2m2 <- lm(log(R_moment_2) ~ poly(St, 3) + factor(Re) + factor(Fr), data = training)
summary(poly2m2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ poly(St, 3) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7756 -1.1822 -0.1902  1.1267  3.3059
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.1625     0.4073  10.219 5.34e-15 ***
## poly(St, 3)1    4.6826     1.6368   2.861  0.00573 **
## poly(St, 3)2   -5.4325     1.6547  -3.283  0.00168 **
## poly(St, 3)3    3.8863     1.6462   2.361  0.02134 *
## factor(Re)224  -4.9066     0.4396 -11.163  < 2e-16 ***
## factor(Re)398  -8.1695     0.5724 -14.273  < 2e-16 ***
## factor(Fr)0.3  -3.8331     0.5227  -7.333 5.24e-10 ***
## factor(Fr)Inf  -2.8036     0.4446  -6.306 3.19e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.623 on 63 degrees of freedom
## Multiple R-squared:  0.832,  Adjusted R-squared:  0.8133
## F-statistic: 44.56 on 7 and 63 DF,  p-value: < 2.2e-16
```

```
poly3m2 <- lm(log(R_moment_2) ~ poly(St, 4) + factor(Re) + factor(Fr), data = training)
summary(poly3m2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ poly(St, 4) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5999 -1.0636 -0.1464  1.0962  3.0934
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.2103     0.3833  10.984 3.51e-16 ***
## poly(St, 4)1    4.7247     1.5391   3.070 0.003176 **
## poly(St, 4)2   -5.5230     1.5561  -3.549 0.000743 ***
## poly(St, 4)3    4.0328     1.5486   2.604 0.011510 *
## poly(St, 4)4   -4.7570     1.5632  -3.043 0.003432 **
## factor(Re)224  -5.0440     0.4158 -12.132  < 2e-16 ***
## factor(Re)398  -8.4668     0.5470 -15.480  < 2e-16 ***
## factor(Fr)0.3  -3.7558     0.4921  -7.632 1.73e-10 ***
## factor(Fr)Inf  -2.6411     0.4215  -6.267 3.92e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.526 on 62 degrees of freedom
## Multiple R-squared:  0.8538, Adjusted R-squared:  0.8349
## F-statistic: 45.26 on 8 and 62 DF,  p-value: < 2.2e-16
```

```
poly4m2 <- lm(log(R_moment_2) ~ poly(St, 5) + factor(Re) + factor(Fr), data = training)
summary(poly4m2)
```

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

```
## lm(formula = log(R_moment_2) ~ poly(St, 5) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8136 -1.0494 -0.3377  1.1077  3.4162
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.1380     0.3775  10.960 4.76e-16 ***
## poly(St, 5)1    4.7599     1.5082   3.156  0.00248 **
## poly(St, 5)2   -5.4749     1.5250  -3.590  0.00066 ***
## poly(St, 5)3    4.0062     1.5174   2.640  0.01051 *
## poly(St, 5)4   -4.7209     1.5318  -3.082  0.00309 **
## poly(St, 5)5    2.8550     1.5098   1.891  0.06339 .
## factor(Re)224  -5.0010     0.4080 -12.257  < 2e-16 ***
## factor(Re)398  -8.3515     0.5394 -15.483  < 2e-16 ***
## factor(Fr)0.3  -3.6647     0.4846  -7.562 2.50e-10 ***
## factor(Fr)Inf  -2.6151     0.4132  -6.329 3.25e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.495 on 61 degrees of freedom
## Multiple R-squared:  0.8619, Adjusted R-squared:  0.8415
## F-statistic:  42.3 on 9 and 61 DF,  p-value: < 2.2e-16
```

```
poly5m2 <- lm(log(R_moment_2) ~ poly(St, 6) + factor(Re) + factor(Fr), data = training)
summary(poly5m2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ poly(St, 6) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8324 -1.0196 -0.2869  1.0730  3.4527
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.1345     0.3810  10.852 8.82e-16 ***
## poly(St, 6)1    4.7586     1.5202   3.130 0.002698 **
## poly(St, 6)2   -5.4714     1.5372  -3.559 0.000734 ***
## poly(St, 6)3    3.9995     1.5299   2.614 0.011292 *
## poly(St, 6)4   -4.7117     1.5448  -3.050 0.003403 **
## poly(St, 6)5    2.8575     1.5219   1.878 0.065305 .
## poly(St, 6)6   -0.2969     1.5229  -0.195 0.846073
## factor(Re)224  -4.9933     0.4132 -12.085  < 2e-16 ***
## factor(Re)398  -8.3388     0.5476 -15.227  < 2e-16 ***
## factor(Fr)0.3  -3.6682     0.4888  -7.504 3.45e-10 ***
## factor(Fr)Inf  -2.6202     0.4173  -6.279 4.19e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1.507 on 60 degrees of freedom
## Multiple R-squared:  0.862,  Adjusted R-squared:  0.839
## F-statistic: 37.47 on 10 and 60 DF,  p-value: < 2.2e-16
```

```
poly6m2 <- lm(log(R_moment_2) ~ poly(St, 7) + factor(Re) + factor(Fr), data = training)
summary(poly6m2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ poly(St, 7) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7372 -0.9985 -0.2034  0.9639  3.6305
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.1126     0.3801  10.818 1.24e-15 ***
## poly(St, 7)1    4.7756     1.5152   3.152 0.002550 **
## poly(St, 7)2   -5.4871     1.5321  -3.581 0.000692 ***
## poly(St, 7)3    3.9884     1.5248   2.616 0.011291 *
## poly(St, 7)4   -4.7166     1.5396  -3.064 0.003294 **
## poly(St, 7)5    2.8668     1.5169   1.890 0.063675 .
## poly(St, 7)6   -0.2972     1.5178  -0.196 0.845423
## poly(St, 7)7    1.7885     1.5093   1.185 0.240782
## factor(Re)224  -4.9895     0.4118 -12.116  < 2e-16 ***
## factor(Re)398  -8.3182     0.5461 -15.233  < 2e-16 ***
## factor(Fr)0.3  -3.6608     0.4872  -7.514 3.65e-10 ***
## factor(Fr)Inf  -2.5793     0.4174  -6.180 6.49e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.502 on 59 degrees of freedom
## Multiple R-squared:  0.8652, Adjusted R-squared:  0.8401
## F-statistic: 34.42 on 11 and 59 DF,  p-value: < 2.2e-16
```

```
poly7m2 <- lm(log(R_moment_2) ~ poly(St, 8) + factor(Re) + factor(Fr), data = training)
summary(poly7m2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ poly(St, 8) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7747 -0.9913 -0.2090  0.9555  3.6222
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.1141     0.3835  10.728 2.15e-15 ***
## poly(St, 8)1    4.7753     1.5279   3.125 0.002773 **
```

```
## poly(St, 8)2   -5.4828     1.5453  -3.548 0.000777 ***
## poly(St, 8)3    3.9974     1.5390   2.597 0.011883 *
## poly(St, 8)4   -4.7248     1.5536  -3.041 0.003534 **
## poly(St, 8)5    2.8649     1.5297   1.873 0.066127 .
## poly(St, 8)6   -0.2918     1.5310  -0.191 0.849490
## poly(St, 8)7    1.7852     1.5222   1.173 0.245658
## poly(St, 8)8    0.2241     1.5753   0.142 0.887361
## factor(Re)224  -4.9901     0.4153 -12.016  < 2e-16 ***
## factor(Re)398  -8.3328     0.5601 -14.877  < 2e-16 ***
## factor(Fr)0.3  -3.6525     0.4948  -7.382 6.69e-10 ***
## factor(Fr)Inf  -2.5800     0.4209  -6.130 8.33e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.515 on 58 degrees of freedom
## Multiple R-squared:  0.8652, Adjusted R-squared:  0.8374
## F-statistic: 31.03 on 12 and 58 DF,  p-value: < 2.2e-16
```

```
anova(fit.lm2, polym2, poly2m2, poly3m2, poly4m2, poly5m2, poly6m2, poly7m2)
```

```
## Analysis of Variance Table
##
## Model 1: log(R_moment_2) ~ (St + factor(Re) + factor(Fr) + factor(Re) *
##     factor(Fr))
## Model 2: log(R_moment_2) ~ poly(St, 2) + factor(Re) + factor(Fr)
## Model 3: log(R_moment_2) ~ poly(St, 3) + factor(Re) + factor(Fr)
## Model 4: log(R_moment_2) ~ poly(St, 4) + factor(Re) + factor(Fr)
## Model 5: log(R_moment_2) ~ poly(St, 5) + factor(Re) + factor(Fr)
## Model 6: log(R_moment_2) ~ poly(St, 6) + factor(Re) + factor(Fr)
## Model 7: log(R_moment_2) ~ poly(St, 7) + factor(Re) + factor(Fr)
## Model 8: log(R_moment_2) ~ poly(St, 8) + factor(Re) + factor(Fr)
##   Res.Df     RSS Df Sum of Sq       F    Pr(>F)
## 1     62  30.517
## 2     64 180.634 -2  -150.117 32.7100 3.085e-10 ***
## 3     63 165.953  1    14.681  6.3980  0.014165 *
## 4     62 144.387  1    21.566  9.3984  0.003295 **
## 5     61 136.392  1     7.995  3.4840  0.067022 .
## 6     60 136.306  1     0.086  0.0376  0.846856
## 7     59 133.137  1     3.169  1.3808  0.244762
## 8     58 133.091  1     0.046  0.0202  0.887361
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pred.polym2 <- predict(polym2, testing)
pred.poly2m2 <- predict(poly2m2, testing)
pred.poly3m2 <- predict(poly3m2, testing)
pred.poly4m2 <- predict(poly4m2, testing)
pred.poly5m2 <- predict(poly5m2, testing)
pred.poly6m2 <- predict(poly6m2, testing)
pred.poly7m2 <- predict(poly7m2, testing)
mse_polym2 <- mean((pred.polym2 - log(testing$R_moment_2))^2)
mse_poly2m2 <- mean((pred.poly2m2 - log(testing$R_moment_2))^2)
mse_poly3m2 <- mean((pred.poly3m2 - log(testing$R_moment_2))^2)
```

```
mse_poly4m2 <- mean((pred.poly4m2 - log(testing$R_moment_2))^2)
mse_poly5m2 <- mean((pred.poly5m2 - log(testing$R_moment_2))^2)
mse_poly6m2 <- mean((pred.poly6m2 - log(testing$R_moment_2))^2)
mse_poly7m2 <- mean((pred.poly7m2 - log(testing$R_moment_2))^2)
mse_test2
```

```
## [1] 5.657288
```

```
mse_polym2
```

```
## [1] 3.698693
```

```
mse_poly2m2
```

```
## [1] 3.954142
```

```
mse_poly3m2
```

```
## [1] 3.92246
```

```
mse_poly4m2
```

```
## [1] 3.614817
```

```
mse_poly5m2
```

```
## [1] 3.542806
```

```
mse_poly6m2
```

```
## [1] 3.324489
```

```
mse_poly7m2
```

```
## [1] 3.373734
```

Same as linear regression? Polynomial model with degree 7 has lowest MSE, but degree 5 or LSR may be better based on ANOVA.

Third moment:

```
polym3 <- lm(log(R_moment_3) ~ poly(St, 2) + factor(Re) + factor(Fr), data = training)
summary(polym3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ poly(St, 2) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.5859 -2.1191 -0.3035  2.2601  5.7511
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.9306     0.7634  14.318  < 2e-16 ***
## poly(St, 2)1    6.3124     3.0727   2.054  0.04403 *
## poly(St, 2)2   -8.6934     3.1063  -2.799  0.00677 **
## factor(Re)224  -6.2233     0.8220  -7.571 1.84e-10 ***
## factor(Re)398 -10.5048     1.0644  -9.869 1.77e-14 ***
## factor(Fr)0.3  -7.5429     0.9795  -7.701 1.09e-10 ***
## factor(Fr)Inf  -5.5606     0.8345  -6.663 7.23e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.047 on 64 degrees of freedom
## Multiple R-squared:  0.7481, Adjusted R-squared:  0.7245
## F-statistic: 31.68 on 6 and 64 DF,  p-value: < 2.2e-16
```

```
poly2m3 <- lm(log(R_moment_3) ~ poly(St, 3) + factor(Re) + factor(Fr), data = training)
summary(poly2m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ poly(St, 3) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9637 -2.2255 -0.3323  2.0431  5.9714
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    11.0211     0.7444  14.805  < 2e-16 ***
## poly(St, 3)1    6.3195     2.9913   2.113  0.03861 *
## poly(St, 3)2   -8.7005     3.0240  -2.877  0.00547 **
## poly(St, 3)3    6.4023     3.0085   2.128  0.03725 *
## factor(Re)224  -6.3740     0.8033  -7.934 4.65e-11 ***
## factor(Re)398 -10.8095     1.0461 -10.334 3.43e-15 ***
## factor(Fr)0.3  -7.4206     0.9553  -7.768 9.09e-11 ***
## factor(Fr)Inf  -5.5263     0.8126  -6.801 4.44e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.966 on 63 degrees of freedom
## Multiple R-squared:  0.765,  Adjusted R-squared:  0.7389
## F-statistic: 29.29 on 7 and 63 DF,  p-value: < 2.2e-16
```

```r
poly3m3 <- lm(log(R_moment_3) ~ poly(St, 4) + factor(Re) + factor(Fr), data = training)
summary(poly3m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ poly(St, 4) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.673 -1.985 -0.297  2.055  5.774
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    11.1002     0.7099  15.636  < 2e-16 ***
## poly(St, 4)1    6.3891     2.8503   2.242  0.02858 *
## poly(St, 4)2   -8.8504     2.8819  -3.071  0.00317 **
## poly(St, 4)3    6.6448     2.8679   2.317  0.02382 *
## poly(St, 4)4   -7.8726     2.8950  -2.719  0.00847 **
## factor(Re)224  -6.6014     0.7700  -8.573 4.03e-12 ***
## factor(Re)398 -11.3013     1.0130 -11.157  < 2e-16 ***
## factor(Fr)0.3  -7.2927     0.9114  -8.001 3.94e-11 ***
## factor(Fr)Inf  -5.2574     0.7805  -6.736 6.15e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.826 on 62 degrees of freedom
## Multiple R-squared:   0.79,  Adjusted R-squared:  0.7629
## F-statistic: 29.16 on 8 and 62 DF,  p-value: < 2.2e-16
```

```r
poly4m3 <- lm(log(R_moment_3) ~ poly(St, 5) + factor(Re) + factor(Fr), data = training)
summary(poly4m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ poly(St, 5) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.940 -1.835 -0.610  1.995  6.333
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.9867     0.7050  15.584  < 2e-16 ***
## poly(St, 5)1    6.4443     2.8162   2.288  0.02560 *
## poly(St, 5)2   -8.7749     2.8475  -3.082  0.00309 **
## poly(St, 5)3    6.6031     2.8335   2.330  0.02311 *
## poly(St, 5)4   -7.8160     2.8604  -2.733  0.00821 **
## poly(St, 5)5    4.4768     2.8193   1.588  0.11747
## factor(Re)224  -6.5340     0.7619  -8.576 4.51e-12 ***
## factor(Re)398 -11.1206     1.0072 -11.041 3.53e-16 ***
```

```
## factor(Fr)0.3  -7.1498     0.9049  -7.901 6.51e-11 ***
## factor(Fr)Inf  -5.2166     0.7716  -6.761 5.95e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.792 on 61 degrees of freedom
## Multiple R-squared:  0.7984, Adjusted R-squared:  0.7686
## F-statistic: 26.83 on 9 and 61 DF,  p-value: < 2.2e-16
```

```
poly5m3 <- lm(log(R_moment_3) ~ poly(St, 6) + factor(Re) + factor(Fr), data = training)
summary(poly5m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ poly(St, 6) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9366 -1.8405 -0.6204  1.9943  6.3257
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.9874     0.7116  15.440  < 2e-16 ***
## poly(St, 6)1    6.4446     2.8396   2.270  0.02684 *
## poly(St, 6)2   -8.7756     2.8714  -3.056  0.00334 **
## poly(St, 6)3    6.6044     2.8577   2.311  0.02428 *
## poly(St, 6)4   -7.8179     2.8854  -2.709  0.00877 **
## poly(St, 6)5    4.4763     2.8428   1.575  0.12060
## poly(St, 6)6    0.0608     2.8446   0.021  0.98302
## factor(Re)224  -6.5356     0.7718  -8.469 7.81e-12 ***
## factor(Re)398 -11.1232     1.0229 -10.874 8.13e-16 ***
## factor(Fr)0.3  -7.1491     0.9131  -7.830 9.57e-11 ***
## factor(Fr)Inf  -5.2155     0.7795  -6.691 8.41e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.815 on 60 degrees of freedom
## Multiple R-squared:  0.7984, Adjusted R-squared:  0.7648
## F-statistic: 23.76 on 10 and 60 DF,  p-value: < 2.2e-16
```

```
poly6m3 <- lm(log(R_moment_3) ~ poly(St, 7) + factor(Re) + factor(Fr), data = training)
summary(poly6m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ poly(St, 7) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7756 -1.9581 -0.5718  1.8527  6.6265
##
```

```
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.9504     0.7116  15.389  < 2e-16 ***
## poly(St, 7)1    6.4733     2.8362   2.282  0.02609 *
## poly(St, 7)2   -8.8021     2.8680  -3.069  0.00324 **
## poly(St, 7)3    6.5856     2.8543   2.307  0.02457 *
## poly(St, 7)4   -7.8261     2.8819  -2.716  0.00867 **
## poly(St, 7)5    4.4920     2.8394   1.582  0.11898
## poly(St, 7)6    0.0603     2.8411   0.021  0.98314
## poly(St, 7)7    3.0242     2.8252   1.070  0.28879
## factor(Re)224  -6.5291     0.7708  -8.470 8.81e-12 ***
## factor(Re)398 -11.0885     1.0221 -10.848 1.12e-15 ***
## factor(Fr)0.3  -7.1366     0.9120  -7.825 1.08e-10 ***
## factor(Fr)Inf  -5.1463     0.7813  -6.587 1.35e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.812 on 59 degrees of freedom
## Multiple R-squared:  0.8022, Adjusted R-squared:  0.7653
## F-statistic: 21.75 on 11 and 59 DF,  p-value: < 2.2e-16
```

```
poly7m3 <- lm(log(R_moment_3) ~ poly(St, 8) + factor(Re) + factor(Fr), data = training)
summary(poly7m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ poly(St, 8) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8894 -1.8271 -0.4686  1.8300  6.6013
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.95501    0.71765  15.265  < 2e-16 ***
## poly(St, 8)1    6.47244    2.85928   2.264  0.02735 *
## poly(St, 8)2   -8.78903    2.89182  -3.039  0.00355 **
## poly(St, 8)3    6.61317    2.87995   2.296  0.02529 *
## poly(St, 8)4   -7.85100    2.90735  -2.700  0.00906 **
## poly(St, 8)5    4.48616    2.86254   1.567  0.12251
## poly(St, 8)6    0.07662    2.86508   0.027  0.97876
## poly(St, 8)7    3.01437    2.84848   1.058  0.29433
## poly(St, 8)8    0.68044    2.94801   0.231  0.81827
## factor(Re)224  -6.53100    0.77714  -8.404 1.29e-11 ***
## factor(Re)398 -11.13280    1.04818 -10.621 3.17e-15 ***
## factor(Fr)0.3  -7.11139    0.92593  -7.680 2.11e-10 ***
## factor(Fr)Inf  -5.14829    0.78766  -6.536 1.75e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.835 on 58 degrees of freedom
## Multiple R-squared:  0.8024, Adjusted R-squared:  0.7615
## F-statistic: 19.62 on 12 and 58 DF,  p-value: 3.66e-16
```

```
poly8m3 <- lm(log(R_moment_3) ~ poly(St, 9) + factor(Re) + factor(Fr), data = training)
summary(poly8m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ poly(St, 9) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9652 -1.7446 -0.5884  1.8966  6.5355
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.97039    0.72562  15.119  < 2e-16 ***
## poly(St, 9)1    6.46980    2.88237   2.245  0.02869 *
## poly(St, 9)2   -8.78986    2.91517  -3.015  0.00383 **
## poly(St, 9)3    6.61846    2.90326   2.280  0.02639 *
## poly(St, 9)4   -7.85103    2.93082  -2.679  0.00964 **
## poly(St, 9)5    4.48454    2.88566   1.554  0.12570
## poly(St, 9)6    0.08032    2.88824   0.028  0.97791
## poly(St, 9)7    3.01047    2.87151   1.048  0.29888
## poly(St, 9)8    0.67828    2.97181   0.228  0.82028
## poly(St, 9)9   -0.78979    2.88552  -0.274  0.78530
## factor(Re)224  -6.55547    0.78850  -8.314 2.07e-11 ***
## factor(Re)398 -11.13557    1.05669 -10.538 5.32e-15 ***
## factor(Fr)0.3  -7.10851    0.93346  -7.615 3.00e-10 ***
## factor(Fr)Inf  -5.15994    0.79516  -6.489 2.25e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.858 on 57 degrees of freedom
## Multiple R-squared:  0.8026, Adjusted R-squared:  0.7576
## F-statistic: 17.83 on 13 and 57 DF,  p-value: 1.612e-15
```

```
anova(fit.lm3, polym3, poly2m3, poly3m3, poly4m3, poly5m3, poly6m3, poly7m3, poly8m3)
```

```
## Analysis of Variance Table
##
## Model 1: log(R_moment_3) ~ (St + factor(Re) + factor(Fr) + factor(Re) *
##     factor(Fr))
## Model 2: log(R_moment_3) ~ poly(St, 2) + factor(Re) + factor(Fr)
## Model 3: log(R_moment_3) ~ poly(St, 3) + factor(Re) + factor(Fr)
## Model 4: log(R_moment_3) ~ poly(St, 4) + factor(Re) + factor(Fr)
## Model 5: log(R_moment_3) ~ poly(St, 5) + factor(Re) + factor(Fr)
## Model 6: log(R_moment_3) ~ poly(St, 6) + factor(Re) + factor(Fr)
## Model 7: log(R_moment_3) ~ poly(St, 7) + factor(Re) + factor(Fr)
## Model 8: log(R_moment_3) ~ poly(St, 8) + factor(Re) + factor(Fr)
## Model 9: log(R_moment_3) ~ poly(St, 9) + factor(Re) + factor(Fr)
##   Res.Df    RSS Df Sum of Sq       F    Pr(>F)
## 1     62  68.18
## 2     64 594.14 -2   -525.96 32.2040 4.379e-10 ***
```

```
## 3     63 554.29  1     39.85  4.8794   0.03121 *
## 4     62 495.23  1     59.07  7.2332   0.00937 **
## 5     61 475.57  1     19.66  2.4073   0.12630
## 6     60 475.56  1      0.00  0.0004   0.98327
## 7     59 466.50  1      9.06  1.1094   0.29665
## 8     58 466.08  1      0.43  0.0524   0.81972
## 9     57 465.46  1      0.61  0.0749   0.78530
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pred.polym3 <- predict(polym3, testing)
pred.poly2m3 <- predict(poly2m3, testing)
pred.poly3m3 <- predict(poly3m3, testing)
pred.poly4m3 <- predict(poly4m3, testing)
pred.poly5m3 <- predict(poly5m3, testing)
pred.poly6m3 <- predict(poly6m3, testing)
pred.poly7m3 <- predict(poly7m3, testing)
pred.poly8m3 <- predict(poly8m3, testing)
mse_polym3 <- mean((pred.polym3 - log(testing$R_moment_3))^2)
mse_poly2m3 <- mean((pred.poly2m3 - log(testing$R_moment_3))^2)
mse_poly3m3 <- mean((pred.poly3m3 - log(testing$R_moment_3))^2)
mse_poly4m3 <- mean((pred.poly4m3 - log(testing$R_moment_3))^2)
mse_poly5m3 <- mean((pred.poly5m3 - log(testing$R_moment_3))^2)
mse_poly6m3 <- mean((pred.poly6m3 - log(testing$R_moment_3))^2)
mse_poly7m3 <- mean((pred.poly7m3 - log(testing$R_moment_3))^2)
mse_poly8m3 <- mean((pred.poly8m3 - log(testing$R_moment_3))^2)
mse_test3
```

```
## [1] 15.70981
```

```
mse_polym3
```

```
## [1] 11.47799
```

```
mse_poly2m3
```

```
## [1] 12.44993
```

```
mse_poly3m3
```

```
## [1] 12.44521
```

```
mse_poly4m3
```

```
## [1] 11.6731
```

```
mse_poly5m3
```

```
## [1] 11.69897
```

```
mse_poly6m3
```

```
## [1] 11.09748
```

```
mse_poly7m3
```

```
## [1] 11.3553
```

```
mse_poly8m3
```

```
## [1] 11.70741
```

Seem to be slightly worse than linear regression. Optimal model in terms of MSE still seems to be Least Squares.

Fourth moment:

```
polym4 <- lm(log(R_moment_4) ~ poly(St, 2) + factor(Re) + factor(Fr), data = training)
summary(polym4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ poly(St, 2) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5706 -3.1427 -0.5878  3.3212  8.1907
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     17.823      1.093  16.312  < 2e-16 ***
## poly(St, 2)1     7.648      4.398   1.739   0.0868 .
## poly(St, 2)2   -11.584      4.446  -2.606   0.0114 *
## factor(Re)224   -7.693      1.176  -6.539 1.19e-08 ***
## factor(Re)398  -13.109      1.523  -8.605 2.78e-12 ***
## factor(Fr)0.3  -11.123      1.402  -7.935 4.20e-11 ***
## factor(Fr)Inf   -8.270      1.194  -6.924 2.53e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.361 on 64 degrees of freedom
## Multiple R-squared:  0.7228, Adjusted R-squared:  0.6968
## F-statistic: 27.81 on 6 and 64 DF,  p-value: 4.449e-16
```

```
poly2m4 <- lm(log(R_moment_4) ~ poly(St, 3) + factor(Re) + factor(Fr), data = training)
summary(poly2m4)
```

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

```
## lm(formula = log(R_moment_4) ~ poly(St, 3) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.1120 -3.1859 -0.6285  3.0255  8.4875
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     17.945      1.070  16.774  < 2e-16 ***
## poly(St, 3)1     7.657      4.299   1.781   0.0797 .
## poly(St, 3)2   -11.593      4.346  -2.668   0.0097 **
## poly(St, 3)3     8.623      4.323   1.995   0.0504 .
## factor(Re)224   -7.896      1.154  -6.839 3.80e-09 ***
## factor(Re)398  -13.519      1.503  -8.993 6.65e-13 ***
## factor(Fr)0.3  -10.959      1.373  -7.983 3.83e-11 ***
## factor(Fr)Inf   -8.224      1.168  -7.043 1.68e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.263 on 63 degrees of freedom
## Multiple R-squared:  0.7392, Adjusted R-squared:  0.7103
## F-statistic: 25.51 on 7 and 63 DF,  p-value: 3.788e-16
```

```
poly3m4 <- lm(log(R_moment_4) ~ poly(St, 4) + factor(Re) + factor(Fr), data = training)
summary(poly3m4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ poly(St, 4) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7076 -2.7785 -0.4152  2.8563  8.2559
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     18.052      1.027  17.580  < 2e-16 ***
## poly(St, 4)1     7.752      4.123   1.880  0.06478 .
## poly(St, 4)2   -11.796      4.168  -2.830  0.00627 **
## poly(St, 4)3     8.952      4.148   2.158  0.03480 *
## poly(St, 4)4   -10.672      4.188  -2.549  0.01331 *
## factor(Re)224   -8.204      1.114  -7.366 4.99e-10 ***
## factor(Re)398  -14.186      1.465  -9.682 5.13e-14 ***
## factor(Fr)0.3  -10.785      1.318  -8.181 1.92e-11 ***
## factor(Fr)Inf   -7.859      1.129  -6.961 2.51e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.088 on 62 degrees of freedom
## Multiple R-squared:  0.764,  Adjusted R-squared:  0.7335
## F-statistic: 25.08 on 8 and 62 DF,  p-value: < 2.2e-16
```

```r
poly4m4 <- lm(log(R_moment_4) ~ poly(St, 5) + factor(Re) + factor(Fr), data = training)
summary(poly4m4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ poly(St, 5) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9842 -2.5057 -0.8637  2.9498  8.9878
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     17.904      1.024  17.492  < 2e-16 ***
## poly(St, 5)1     7.824      4.089   1.914   0.0604 .
## poly(St, 5)2   -11.698      4.134  -2.829   0.0063 **
## poly(St, 5)3     8.898      4.114   2.163   0.0345 *
## poly(St, 5)4   -10.598      4.153  -2.552   0.0132 *
## poly(St, 5)5     5.856      4.093   1.431   0.1577
## factor(Re)224   -8.116      1.106  -7.337 6.10e-10 ***
## factor(Re)398  -13.949      1.462  -9.539 1.05e-13 ***
## factor(Fr)0.3  -10.598      1.314  -8.066 3.38e-11 ***
## factor(Fr)Inf   -7.806      1.120  -6.968 2.63e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.054 on 61 degrees of freedom
## Multiple R-squared:  0.7716, Adjusted R-squared:  0.7379
## F-statistic:  22.9 on 9 and 61 DF,  p-value: < 2.2e-16
```

```r
poly5m4 <- lm(log(R_moment_4) ~ poly(St, 6) + factor(Re) + factor(Fr), data = training)
summary(poly5m4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ poly(St, 6) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9466 -2.5461 -0.8998  2.9398  8.9150
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    17.9105     1.0330  17.338  < 2e-16 ***
## poly(St, 6)1    7.8266     4.1220   1.899  0.06241 .
## poly(St, 6)2  -11.7046     4.1682  -2.808  0.00672 **
## poly(St, 6)3    8.9109     4.1484   2.148  0.03576 *
## poly(St, 6)4  -10.6163     4.1886  -2.535  0.01389 *
## poly(St, 6)5    5.8505     4.1267   1.418  0.16145
## poly(St, 6)6    0.5934     4.1293   0.144  0.88622
```

```
## factor(Re)224  -8.1312     1.1203  -7.258 9.08e-10 ***
## factor(Re)398 -13.9749     1.4849  -9.412 2.02e-13 ***
## factor(Fr)0.3 -10.5913     1.3255  -7.991 5.08e-11 ***
## factor(Fr)Inf  -7.7955     1.1316  -6.889 3.87e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.087 on 60 degrees of freedom
## Multiple R-squared:  0.7717, Adjusted R-squared:  0.7336
## F-statistic: 20.28 on 10 and 60 DF,  p-value: 9.727e-16
```

```
poly6m4 <- lm(log(R_moment_4) ~ poly(St, 7) + factor(Re) + factor(Fr), data = training)
summary(poly6m4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ poly(St, 7) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7261 -2.8475 -0.8583  2.7493  9.3269
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    17.8597     1.0341  17.271  < 2e-16 ***
## poly(St, 7)1    7.8659     4.1216   1.908  0.06120 .
## poly(St, 7)2  -11.7408     4.1677  -2.817  0.00658 **
## poly(St, 7)3    8.8852     4.1478   2.142  0.03632 *
## poly(St, 7)4  -10.6276     4.1880  -2.538  0.01382 *
## poly(St, 7)5    5.8720     4.1262   1.423  0.15997
## poly(St, 7)6    0.5927     4.1287   0.144  0.88634
## poly(St, 7)7    4.1414     4.1056   1.009  0.31723
## factor(Re)224  -8.1222     1.1202  -7.251 1.02e-09 ***
## factor(Re)398 -13.9274     1.4854  -9.376 2.71e-13 ***
## factor(Fr)0.3 -10.5742     1.3254  -7.978 5.95e-11 ***
## factor(Fr)Inf  -7.7007     1.1353  -6.783 6.30e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.086 on 59 degrees of freedom
## Multiple R-squared:  0.7756, Adjusted R-squared:  0.7337
## F-statistic: 18.54 on 11 and 59 DF,  p-value: 2.798e-15
```

```
poly7m4 <- lm(log(R_moment_4) ~ poly(St, 8) + factor(Re) + factor(Fr), data = training)
summary(poly7m4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ poly(St, 8) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -6.9316 -2.6638 -0.7832  2.7083  9.2814
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    17.8681     1.0426  17.138  < 2e-16 ***
## poly(St, 8)1    7.8644     4.1541   1.893  0.06333 .
## poly(St, 8)2  -11.7173     4.2013  -2.789  0.00714 **
## poly(St, 8)3    8.9349     4.1841   2.135  0.03696 *
## poly(St, 8)4  -10.6725     4.2239  -2.527  0.01426 *
## poly(St, 8)5    5.8614     4.1588   1.409  0.16406
## poly(St, 8)6    0.6222     4.1625   0.149  0.88170
## poly(St, 8)7    4.1237     4.1384   0.996  0.32317
## poly(St, 8)8    1.2285     4.2830   0.287  0.77525
## factor(Re)224  -8.1257     1.1291  -7.197 1.37e-09 ***
## factor(Re)398 -14.0074     1.5228  -9.198 6.25e-13 ***
## factor(Fr)0.3 -10.5286     1.3452  -7.827 1.20e-10 ***
## factor(Fr)Inf  -7.7043     1.1443  -6.733 8.23e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.118 on 58 degrees of freedom
## Multiple R-squared:  0.7759, Adjusted R-squared:  0.7295
## F-statistic: 16.73 on 12 and 58 DF,  p-value: 1.197e-14
```

```
poly8m4 <- lm(log(R_moment_4) ~ poly(St, 8) + factor(Re) + factor(Fr), data = training)
summary(poly8m4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ poly(St, 8) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9316 -2.6638 -0.7832  2.7083  9.2814
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    17.8681     1.0426  17.138  < 2e-16 ***
## poly(St, 8)1    7.8644     4.1541   1.893  0.06333 .
## poly(St, 8)2  -11.7173     4.2013  -2.789  0.00714 **
## poly(St, 8)3    8.9349     4.1841   2.135  0.03696 *
## poly(St, 8)4  -10.6725     4.2239  -2.527  0.01426 *
## poly(St, 8)5    5.8614     4.1588   1.409  0.16406
## poly(St, 8)6    0.6222     4.1625   0.149  0.88170
## poly(St, 8)7    4.1237     4.1384   0.996  0.32317
## poly(St, 8)8    1.2285     4.2830   0.287  0.77525
## factor(Re)224  -8.1257     1.1291  -7.197 1.37e-09 ***
## factor(Re)398 -14.0074     1.5228  -9.198 6.25e-13 ***
## factor(Fr)0.3 -10.5286     1.3452  -7.827 1.20e-10 ***
## factor(Fr)Inf  -7.7043     1.1443  -6.733 8.23e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 4.118 on 58 degrees of freedom
## Multiple R-squared:  0.7759, Adjusted R-squared:  0.7295
## F-statistic: 16.73 on 12 and 58 DF,  p-value: 1.197e-14
```

```
anova(fit.lm4, polym4, poly2m4, poly3m4, poly4m4, poly5m4, poly6m4, poly7m4, poly8m4)
```

```
## Analysis of Variance Table
##
## Model 1: log(R_moment_4) ~ (St + factor(Re) + factor(Fr) + factor(Re) *
##     factor(Fr))
## Model 2: log(R_moment_4) ~ poly(St, 2) + factor(Re) + factor(Fr)
## Model 3: log(R_moment_4) ~ poly(St, 3) + factor(Re) + factor(Fr)
## Model 4: log(R_moment_4) ~ poly(St, 4) + factor(Re) + factor(Fr)
## Model 5: log(R_moment_4) ~ poly(St, 5) + factor(Re) + factor(Fr)
## Model 6: log(R_moment_4) ~ poly(St, 6) + factor(Re) + factor(Fr)
## Model 7: log(R_moment_4) ~ poly(St, 7) + factor(Re) + factor(Fr)
## Model 8: log(R_moment_4) ~ poly(St, 8) + factor(Re) + factor(Fr)
## Model 9: log(R_moment_4) ~ poly(St, 8) + factor(Re) + factor(Fr)
##   Res.Df     RSS Df Sum of Sq       F   Pr(>F)
## 1     62  111.55
## 2     64 1216.95 -2  -1105.41 32.5860 3.27e-10 ***
## 3     63 1144.66  1     72.29  4.2619  0.04346 *
## 4     62 1036.12  1    108.54  6.3994  0.01415 *
## 5     61 1002.49  1     33.63  1.9828  0.16443
## 6     60 1002.14  1      0.34  0.0203  0.88710
## 7     59  985.15  1     16.99  1.0017  0.32106
## 8     58  983.76  1      1.40  0.0823  0.77525
## 9     58  983.76  0      0.00
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pred.polym4 <- predict(polym4, testing)
pred.poly2m4 <- predict(poly2m4, testing)
pred.poly3m4 <- predict(poly3m4, testing)
pred.poly4m4 <- predict(poly4m4, testing)
pred.poly5m4 <- predict(poly5m4, testing)
pred.poly6m4 <- predict(poly6m4, testing)
pred.poly7m4 <- predict(poly7m4, testing)
pred.poly8m4 <- predict(poly8m4, testing)
mse_polym4 <- mean((pred.polym4 - log(testing$R_moment_4))^2)
mse_poly2m4 <- mean((pred.poly2m4 - log(testing$R_moment_4))^2)
mse_poly3m4 <- mean((pred.poly3m4 - log(testing$R_moment_4))^2)
mse_poly4m4 <- mean((pred.poly4m4 - log(testing$R_moment_4))^2)
mse_poly5m4 <- mean((pred.poly5m4 - log(testing$R_moment_4))^2)
mse_poly6m4 <- mean((pred.poly6m4 - log(testing$R_moment_4))^2)
mse_poly7m4 <- mean((pred.poly7m4 - log(testing$R_moment_4))^2)
mse_poly8m4 <- mean((pred.poly8m4 - log(testing$R_moment_4))^2)
mse_test4
```

```
## [1] 28.83488
```

```
mse_polym4
```

```
## [1] 22.95277
```

```
mse_poly2m4
```

```
## [1] 25.02578
```

```
mse_poly3m4
```

```
## [1] 25.05991
```

```
mse_poly4m4
```

```
## [1] 23.69646
```

```
mse_poly5m4
```

```
## [1] 24.05208
```

```
mse_poly6m4
```

```
## [1] 22.95743
```

```
mse_poly7m4
```

```
## [1] 23.60226
```

```
mse_poly8m4
```

```
## [1] 23.60226
```

The linear regression fit seems to have the minimal MSE for the fourth order.

Splines

First moment:

```
spline1 <- lm(log(R_moment_1) ~ bs(log(St)) + factor(Re) + factor(Fr), data = training)
summary(spline1)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ bs(log(St)) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.29700 -0.06613 -0.00928  0.09227  0.27652
##
## Coefficients:
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   -2.695985   0.062097  -43.416  < 2e-16 ***
## bs(log(St))1   0.504229   0.151056    3.338 0.001420 **
## bs(log(St))2   0.418550   0.106273    3.938 0.000208 ***
## bs(log(St))3   0.922160   0.076158   12.108  < 2e-16 ***
## factor(Re)224 -3.657326   0.033996 -107.582  < 2e-16 ***
## factor(Re)398 -5.814895   0.044247 -131.418  < 2e-16 ***
## factor(Fr)0.3 -0.131149   0.040266   -3.257 0.001815 **
## factor(Fr)Inf -0.004388   0.034483   -0.127 0.899143
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1253 on 63 degrees of freedom
## Multiple R-squared:  0.9971, Adjusted R-squared:  0.9968
## F-statistic:  3069 on 7 and 63 DF,  p-value: < 2.2e-16
```

```
pred.spline1 <- predict(spline1, testing)
mse_spline1 <- mean((pred.spline1 - log(testing$R_moment_1))^2)
spline2 <- lm(log(R_moment_1) ~ bs(log(St), df=4) + factor(Re) + factor(Fr), data = training)
summary(spline2)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ bs(log(St), df = 4) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29645 -0.06942 -0.01147  0.08196  0.28926
##
## Coefficients:
##                      Estimate Std. Error  t value Pr(>|t|)
## (Intercept)          -2.70253    0.06363  -42.476  < 2e-16 ***
## bs(log(St), df = 4)1  0.37947    0.14088    2.694  0.00908 **
## bs(log(St), df = 4)2  0.39201    0.12706    3.085  0.00304 **
## bs(log(St), df = 4)3  0.78764    0.10080    7.814 8.32e-11 ***
## bs(log(St), df = 4)4  0.91380    0.07815   11.693  < 2e-16 ***
## factor(Re)224        -3.65536    0.03438 -106.310  < 2e-16 ***
## factor(Re)398        -5.81008    0.04539 -127.998  < 2e-16 ***
## factor(Fr)0.3        -0.13334    0.04070   -3.276  0.00173 **
## factor(Fr)Inf        -0.00420    0.03468   -0.121  0.90400
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.126 on 62 degrees of freedom
## Multiple R-squared:  0.9971, Adjusted R-squared:  0.9967
## F-statistic:  2655 on 8 and 62 DF,  p-value: < 2.2e-16
```

```
pred.spline2 <- predict(spline2, testing)
mse_spline2 <- mean((pred.spline2 - log(testing$R_moment_1))^2)
spline3 <- lm(log(R_moment_1) ~ bs(log(St), df=5) + factor(Re) + factor(Fr), data = training)
summary(spline3)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ bs(log(St), df = 5) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32045 -0.06353 -0.00529  0.07553  0.30925
##
## Coefficients:
##                       Estimate Std. Error  t value Pr(>|t|)
## (Intercept)          -2.687214   0.062790  -42.797  < 2e-16 ***
## bs(log(St), df = 5)1  0.094829   0.161492    0.587  0.55923
## bs(log(St), df = 5)2  0.549311   0.109090    5.035 4.51e-06 ***
## bs(log(St), df = 5)3  0.491589   0.088726    5.541 6.81e-07 ***
## bs(log(St), df = 5)4  0.920682   0.105136    8.757 2.22e-12 ***
## bs(log(St), df = 5)5  0.879542   0.078427   11.215  < 2e-16 ***
## factor(Re)224        -3.659384   0.033675 -108.668  < 2e-16 ***
## factor(Re)398        -5.818879   0.044594 -130.486  < 2e-16 ***
## factor(Fr)0.3        -0.128906   0.039884   -3.232  0.00198 **
## factor(Fr)Inf        -0.001217   0.033956   -0.036  0.97154
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1232 on 61 degrees of freedom
## Multiple R-squared:  0.9973, Adjusted R-squared:  0.9969
## F-statistic:  2467 on 9 and 61 DF,  p-value: < 2.2e-16
```

```
pred.spline3 <- predict(spline3, testing)
mse_spline3 <- mean((pred.spline3 - log(testing$R_moment_1))^2)
spline4 <- lm(log(R_moment_1) ~ bs(log(St), df=6) + factor(Re) + factor(Fr), data = training)
summary(spline4)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ bs(log(St), df = 6) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32354 -0.06220 -0.00659  0.07268  0.31040
##
## Coefficients:
##                       Estimate Std. Error  t value Pr(>|t|)
## (Intercept)          -2.685613   0.063433  -42.338  < 2e-16 ***
## bs(log(St), df = 6)1  0.047771   0.176694    0.270 0.787809
## bs(log(St), df = 6)2  0.479295   0.131253    3.652 0.000548 ***
```

```
## bs(log(St), df = 6)3  0.505514   0.088456    5.715 3.66e-07 ***
## bs(log(St), df = 6)4  0.599654   0.104721    5.726 3.50e-07 ***
## bs(log(St), df = 6)5  0.934388   0.124194    7.524 3.19e-10 ***
## bs(log(St), df = 6)6  0.877153   0.079039   11.098 3.58e-16 ***
## factor(Re)224        -3.659994   0.034006 -107.629  < 2e-16 ***
## factor(Re)398        -5.819792   0.045046 -129.196  < 2e-16 ***
## factor(Fr)0.3        -0.128687   0.040149   -3.205 0.002163 **
## factor(Fr)Inf        -0.000969   0.034402   -0.028 0.977622
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.124 on 60 degrees of freedom
## Multiple R-squared:  0.9973, Adjusted R-squared:  0.9968
## F-statistic:  2194 on 10 and 60 DF,  p-value: < 2.2e-16
```

```
pred.spline4 <- predict(spline4, testing)
mse_spline4 <- mean((pred.spline4 - log(testing$R_moment_1))^2)
spline5 <- lm(log(R_moment_1) ~ bs(log(St), df=7) + factor(Re) + factor(Fr), data = training)
summary(spline5)
```

```
##
## Call:
## lm(formula = log(R_moment_1) ~ bs(log(St), df = 7) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32523 -0.06066 -0.00016  0.07307  0.30946
##
## Coefficients:
##                       Estimate Std. Error  t value Pr(>|t|)
## (Intercept)          -2.684805   0.064194  -41.823  < 2e-16 ***
## bs(log(St), df = 7)1  0.032301   0.193549    0.167  0.86803
## bs(log(St), df = 7)2  0.437879   0.134680    3.251  0.00190 **
## bs(log(St), df = 7)3  0.496630   0.089607    5.542 7.33e-07 ***
## bs(log(St), df = 7)4  0.528333   0.092052    5.739 3.48e-07 ***
## bs(log(St), df = 7)5  0.720740   0.166629    4.325 5.96e-05 ***
## bs(log(St), df = 7)6  0.920564   0.158934    5.792 2.85e-07 ***
## bs(log(St), df = 7)7  0.877690   0.080121   10.955 7.56e-16 ***
## factor(Re)224        -3.659271   0.034375 -106.451  < 2e-16 ***
## factor(Re)398        -5.819057   0.045823 -126.990  < 2e-16 ***
## factor(Fr)0.3        -0.128912   0.040597   -3.175  0.00238 **
## factor(Fr)Inf        -0.002093   0.034711   -0.060  0.95212
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1253 on 59 degrees of freedom
## Multiple R-squared:  0.9973, Adjusted R-squared:  0.9968
## F-statistic:  1954 on 11 and 59 DF,  p-value: < 2.2e-16
```

```
pred.spline5 <- predict(spline5, testing)
mse_spline5 <- mean((pred.spline5 - log(testing$R_moment_1))^2)
mse_spline1
```

```
## [1] 0.02050534
```

```
mse_spline2
```

```
## [1] 0.01902618
```

```
mse_spline3
```

```
## [1] 0.02246655
```

```
mse_spline4
```

```
## [1] 0.02302992
```

```
mse_spline5
```

```
## [1] 0.02305703
```

Second moment:

```
spline1m2 <- lm(log(R_moment_2) ~ bs(log(St)) + factor(Re) + factor(Fr), data = training)
summary(spline1m2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ bs(log(St)) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5892 -0.9503 -0.1786  0.9750  3.6233
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.3434     0.7347   0.467 0.641799
## bs(log(St))1    4.8146     1.7872   2.694 0.009040 **
## bs(log(St))2    4.3767     1.2574   3.481 0.000913 ***
## bs(log(St))3    4.3560     0.9011   4.834 8.95e-06 ***
## factor(Re)224  -4.9356     0.4022 -12.271  < 2e-16 ***
## factor(Re)398  -8.2172     0.5235 -15.696  < 2e-16 ***
## factor(Fr)0.3  -3.7197     0.4764  -7.808 7.74e-11 ***
## factor(Fr)Inf  -2.6518     0.4080  -6.500 1.48e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.482 on 63 degrees of freedom
## Multiple R-squared:  0.8598, Adjusted R-squared:  0.8443
## F-statistic: 55.21 on 7 and 63 DF,  p-value: < 2.2e-16
```

```
pred.spline1m2 <- predict(spline1m2, testing)
mse_spline1m2 <- mean((pred.spline1m2 - log(testing$R_moment_2))^2)
spline2m2 <- lm(log(R_moment_2) ~ bs(log(St), df=4) + factor(Re) + factor(Fr), data = training)
summary(spline2m2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ bs(log(St), df = 4) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6030 -0.9729 -0.1833  0.9724  3.6160
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.3747     0.7543   0.497 0.621127
## bs(log(St), df = 4)1    2.8433     1.6701   1.703 0.093668 .
## bs(log(St), df = 4)2    4.8044     1.5062   3.190 0.002234 **
## bs(log(St), df = 4)3    4.1510     1.1949   3.474 0.000941 ***
## bs(log(St), df = 4)4    4.3959     0.9265   4.745 1.27e-05 ***
## factor(Re)224          -4.9450     0.4076 -12.132  < 2e-16 ***
## factor(Re)398          -8.2402     0.5381 -15.313  < 2e-16 ***
## factor(Fr)0.3          -3.7092     0.4825  -7.688 1.38e-10 ***
## factor(Fr)Inf          -2.6527     0.4111  -6.452 1.89e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.494 on 62 degrees of freedom
## Multiple R-squared:  0.8599, Adjusted R-squared:  0.8419
## F-statistic: 47.58 on 8 and 62 DF,  p-value: < 2.2e-16
```

```
pred.spline2m2 <- predict(spline2m2, testing)
mse_spline2m2 <- mean((pred.spline2m2 - log(testing$R_moment_2))^2)
spline3m2 <- lm(log(R_moment_2) ~ bs(log(St), df=5) + factor(Re) + factor(Fr), data = training)
summary(spline3m2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ bs(log(St), df = 5) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8236 -0.9019 -0.1415  0.9843  3.5600
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.4618     0.7617   0.606 0.546608
## bs(log(St), df = 5)1    1.0563     1.9591   0.539 0.591722
## bs(log(St), df = 5)2    4.9700     1.3234   3.755 0.000389 ***
## bs(log(St), df = 5)3    3.6529     1.0764   3.394 0.001216 **
```

```
## bs(log(St), df = 5)4   4.8500     1.2754   3.803 0.000334 ***
## bs(log(St), df = 5)5   4.1883     0.9514   4.402 4.39e-05 ***
## factor(Re)224         -4.9664     0.4085 -12.157  < 2e-16 ***
## factor(Re)398         -8.2879     0.5410 -15.320  < 2e-16 ***
## factor(Fr)0.3         -3.6842     0.4839  -7.614 2.03e-10 ***
## factor(Fr)Inf         -2.6362     0.4119  -6.400 2.47e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.495 on 61 degrees of freedom
## Multiple R-squared:  0.8619, Adjusted R-squared:  0.8416
## F-statistic: 42.31 on 9 and 61 DF,  p-value: < 2.2e-16
```

```
pred.spline3m2 <- predict(spline3m2, testing)
mse_spline3m2 <- mean((pred.spline3m2 - log(testing$R_moment_2))^2)
spline4m2 <- lm(log(R_moment_2) ~ bs(log(St), df=6) + factor(Re) + factor(Fr), data = training)
summary(spline4m2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ bs(log(St), df = 6) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7853 -0.9687 -0.2396  1.0630  3.5767
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.4504     0.7688   0.586 0.560172
## bs(log(St), df = 6)1    1.1751     2.1416   0.549 0.585256
## bs(log(St), df = 6)2    3.7787     1.5909   2.375 0.020749 *
## bs(log(St), df = 6)3    4.5589     1.0721   4.252 7.52e-05 ***
## bs(log(St), df = 6)4    3.4480     1.2693   2.717 0.008607 **
## bs(log(St), df = 6)5    5.3150     1.5053   3.531 0.000803 ***
## bs(log(St), df = 6)6    4.1677     0.9580   4.350 5.36e-05 ***
## factor(Re)224          -4.9869     0.4122 -12.099  < 2e-16 ***
## factor(Re)398          -8.3178     0.5460 -15.234  < 2e-16 ***
## factor(Fr)0.3          -3.6887     0.4866  -7.580 2.55e-10 ***
## factor(Fr)Inf          -2.6116     0.4170  -6.263 4.45e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.503 on 60 degrees of freedom
## Multiple R-squared:  0.8628, Adjusted R-squared:  0.8399
## F-statistic: 37.73 on 10 and 60 DF,  p-value: < 2.2e-16
```

```
pred.spline4m2 <- predict(spline4m2, testing)
mse_spline4m2 <- mean((pred.spline4m2 - log(testing$R_moment_2))^2)
spline5m2 <- lm(log(R_moment_2) ~ bs(log(St), df=7) + factor(Re) + factor(Fr), data = training)
summary(spline5m2)
```

```
##
```

```
## Call:
## lm(formula = log(R_moment_2) ~ bs(log(St), df = 7) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8207 -0.8912 -0.1957  1.0151  3.5749
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.4674     0.7783   0.601 0.550432
## bs(log(St), df = 7)1    0.8766     2.3465   0.374 0.710053
## bs(log(St), df = 7)2    3.7224     1.6328   2.280 0.026250 *
## bs(log(St), df = 7)3    4.2751     1.0863   3.935 0.000222 ***
## bs(log(St), df = 7)4    4.1639     1.1160   3.731 0.000430 ***
## bs(log(St), df = 7)5    3.7003     2.0201   1.832 0.072040 .
## bs(log(St), df = 7)6    5.2197     1.9268   2.709 0.008820 **
## bs(log(St), df = 7)7    4.1633     0.9713   4.286 6.82e-05 ***
## factor(Re)224          -4.9803     0.4167 -11.950  < 2e-16 ***
## factor(Re)398          -8.3183     0.5555 -14.973  < 2e-16 ***
## factor(Fr)0.3          -3.6941     0.4922  -7.506 3.76e-10 ***
## factor(Fr)Inf          -2.6262     0.4208  -6.241 5.14e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.519 on 59 degrees of freedom
## Multiple R-squared:  0.8622, Adjusted R-squared:  0.8365
## F-statistic: 33.57 on 11 and 59 DF,  p-value: < 2.2e-16
```

```
pred.spline5m2 <- predict(spline5m2, testing)
mse_spline5m2 <- mean((pred.spline5m2 - log(testing$R_moment_2))^2)
mse_spline1m2
```

```
## [1] 2.977825
```

```
mse_spline2m2
```

```
## [1] 3.063367
```

```
mse_spline3m2
```

```
## [1] 3.290766
```

```
mse_spline4m2
```

```
## [1] 3.288646
```

```
mse_spline5m2
```

```
## [1] 3.29419
```

Third moment:

```
spline1m3 <- lm(log(R_moment_3) ~ bs(log(St)) + factor(Re) + factor(Fr), data = training)
summary(spline1m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ bs(log(St)) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6115 -1.8244 -0.3367  1.8706  6.5840
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5.0661     1.3745   3.686 0.000476 ***
## bs(log(St))1    7.4719     3.3436   2.235 0.028995 *
## bs(log(St))2    7.2474     2.3523   3.081 0.003058 **
## bs(log(St))3    6.4098     1.6858   3.802 0.000326 ***
## factor(Re)224  -6.4233     0.7525  -8.536 4.14e-12 ***
## factor(Re)398 -10.8893     0.9794 -11.118  < 2e-16 ***
## factor(Fr)0.3  -7.2476     0.8913  -8.132 2.10e-11 ***
## factor(Fr)Inf  -5.2838     0.7633  -6.923 2.73e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.773 on 63 degrees of freedom
## Multiple R-squared:  0.7946, Adjusted R-squared:  0.7717
## F-statistic: 34.81 on 7 and 63 DF,  p-value: < 2.2e-16
```

```
pred.spline1m3 <- predict(spline1m3, testing)
mse_spline1m3 <- mean((pred.spline1m3 - log(testing$R_moment_3))^2)
spline2m3 <- lm(log(R_moment_3) ~ bs(log(St), df=4) + factor(Re) + factor(Fr), data = training)
summary(spline2m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ bs(log(St), df = 4) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6192 -1.8419 -0.3788  1.9047  6.5627
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             5.1568     1.4103   3.656 0.000529 ***
## bs(log(St), df = 4)1    4.0616     3.1228   1.301 0.198196
## bs(log(St), df = 4)2    8.1167     2.8164   2.882 0.005424 **
## bs(log(St), df = 4)3    6.0914     2.2343   2.726 0.008317 **
## bs(log(St), df = 4)4    6.5256     1.7323   3.767 0.000370 ***
## factor(Re)224          -6.4505     0.7622  -8.463 6.24e-12 ***
```

```
## factor(Re)398        -10.9560     1.0062 -10.889 5.01e-16 ***
## factor(Fr)0.3         -7.2173     0.9022  -8.000 3.96e-11 ***
## factor(Fr)Inf         -5.2864     0.7687  -6.877 3.51e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.793 on 62 degrees of freedom
## Multiple R-squared:  0.7949, Adjusted R-squared:  0.7685
## F-statistic: 30.04 on 8 and 62 DF,  p-value: < 2.2e-16
```

```
pred.spline2m3 <- predict(spline2m3, testing)
mse_spline2m3 <- mean((pred.spline2m3 - log(testing$R_moment_3))^2)
spline3m3 <- lm(log(R_moment_3) ~ bs(log(St), df=5) + factor(Re) + factor(Fr), data = training)
summary(spline3m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ bs(log(St), df = 5) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9780 -1.6688 -0.3055  1.8169  6.4613
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             5.3141     1.4252   3.729 0.000424 ***
## bs(log(St), df = 5)1    1.0225     3.6654   0.279 0.781215
## bs(log(St), df = 5)2    8.1947     2.4760   3.310 0.001571 **
## bs(log(St), df = 5)3    5.5471     2.0138   2.755 0.007736 **
## bs(log(St), df = 5)4    7.3421     2.3863   3.077 0.003131 **
## bs(log(St), df = 5)5    6.1503     1.7801   3.455 0.001007 **
## factor(Re)224          -6.4885     0.7643  -8.489 6.36e-12 ***
## factor(Re)398         -11.0411     1.0121 -10.909 5.76e-16 ***
## factor(Fr)0.3          -7.1720     0.9053  -7.923 5.98e-11 ***
## factor(Fr)Inf          -5.2573     0.7707  -6.821 4.69e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.797 on 61 degrees of freedom
## Multiple R-squared:  0.7976, Adjusted R-squared:  0.7678
## F-statistic: 26.71 on 9 and 61 DF,  p-value: < 2.2e-16
```

```
pred.spline3m3 <- predict(spline3m3, testing)
mse_spline3m3 <- mean((pred.spline3m3 - log(testing$R_moment_3))^2)
spline4m3 <- lm(log(R_moment_3) ~ bs(log(St), df=6) + factor(Re) + factor(Fr), data = training)
summary(spline4m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ bs(log(St), df = 6) + factor(Re) +
##     factor(Fr), data = training)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9031 -1.7160 -0.5931  2.0268  6.4936
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             5.2917     1.4383   3.679 0.000502 ***
## bs(log(St), df = 6)1    1.4354     4.0064   0.358 0.721397
## bs(log(St), df = 6)2    5.9311     2.9760   1.993 0.050822 .
## bs(log(St), df = 6)3    7.3568     2.0057   3.668 0.000521 ***
## bs(log(St), df = 6)4    5.0078     2.3744   2.109 0.039124 *
## bs(log(St), df = 6)5    8.2355     2.8160   2.925 0.004861 **
## bs(log(St), df = 6)6    6.1113     1.7921   3.410 0.001167 **
## factor(Re)224          -6.5278     0.7710  -8.466 7.88e-12 ***
## factor(Re)398         -11.0984     1.0214 -10.866 8.38e-16 ***
## factor(Fr)0.3          -7.1808     0.9103  -7.888 7.61e-11 ***
## factor(Fr)Inf          -5.2098     0.7800  -6.679 8.80e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.811 on 60 degrees of freedom
## Multiple R-squared:  0.7989, Adjusted R-squared:  0.7654
## F-statistic: 23.84 on 10 and 60 DF,  p-value: < 2.2e-16
```

```
pred.spline4m3 <- predict(spline4m3, testing)
mse_spline4m3 <- mean((pred.spline4m3 - log(testing$R_moment_3))^2)
spline5m3 <- lm(log(R_moment_3) ~ bs(log(St), df=7) + factor(Re) + factor(Fr), data = training)
summary(spline5m3)
```

```
##
## Call:
## lm(formula = log(R_moment_3) ~ bs(log(St), df = 7) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9734 -1.6798 -0.4138  1.8793  6.4911
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             5.3255     1.4557   3.658 0.000543 ***
## bs(log(St), df = 7)1    0.8432     4.3891   0.192 0.848316
## bs(log(St), df = 7)2    5.9099     3.0541   1.935 0.057784 .
## bs(log(St), df = 7)3    6.8045     2.0320   3.349 0.001419 **
## bs(log(St), df = 7)4    6.5568     2.0875   3.141 0.002631 **
## bs(log(St), df = 7)5    5.2793     3.7787   1.397 0.167602
## bs(log(St), df = 7)6    8.1363     3.6042   2.257 0.027694 *
## bs(log(St), df = 7)7    6.0994     1.8169   3.357 0.001384 **
## factor(Re)224          -6.5166     0.7795  -8.360 1.35e-11 ***
## factor(Re)398         -11.1024     1.0391 -10.684 2.03e-15 ***
## factor(Fr)0.3          -7.1917     0.9206  -7.812 1.14e-10 ***
## factor(Fr)Inf          -5.2371     0.7872  -6.653 1.04e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 2.841 on 59 degrees of freedom
## Multiple R-squared:  0.7981, Adjusted R-squared:  0.7605
## F-statistic: 21.21 on 11 and 59 DF,  p-value: < 2.2e-16
```

```
pred.spline5m3 <- predict(spline5m3, testing)
mse_spline5m3 <- mean((pred.spline5m3 - log(testing$R_moment_3))^2)
mse_spline1m3
```

```
## [1] 9.972745
```

```
mse_spline2m3
```

```
## [1] 10.4218
```

```
mse_spline3m3
```

```
## [1] 11.13734
```

```
mse_spline4m3
```

```
## [1] 11.12919
```

```
mse_spline5m3
```

```
## [1] 11.14938
```

Fourth moment:

```
spline1m4 <- lm(log(R_moment_4) ~ bs(log(St)) + factor(Re) + factor(Fr), data = training)
summary(spline1m4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ bs(log(St)) + factor(Re) + factor(Fr),
##     data = training)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.648 -2.759 -0.479  2.731  9.242
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     10.152      1.997   5.083 3.57e-06 ***
## bs(log(St))1     9.668      4.859   1.990  0.05096 .
## bs(log(St))2     9.837      3.418   2.878  0.00546 **
## bs(log(St))3     8.127      2.450   3.318  0.00151 **
## factor(Re)224   -7.964      1.093  -7.283 6.41e-10 ***
## factor(Re)398  -13.629      1.423  -9.576 6.60e-14 ***
```

```
## factor(Fr)0.3  -10.737      1.295  -8.290 1.11e-11 ***
## factor(Fr)Inf   -7.903      1.109  -7.126 1.21e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.03 on 63 degrees of freedom
## Multiple R-squared:  0.7669, Adjusted R-squared:  0.741
## F-statistic: 29.61 on 7 and 63 DF,  p-value: < 2.2e-16
```

```
pred.spline1m4 <- predict(spline1m4, testing)
mse_spline1m4 <- mean((pred.spline1m4 - log(testing$R_moment_4))^2)
spline2m4 <- lm(log(R_moment_4) ~ bs(log(St), df=4) + factor(Re) + factor(Fr), data = training)
summary(spline2m4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ bs(log(St), df = 4) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6611 -2.7934 -0.5599  2.8229  9.2050
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             10.308      2.049   5.032 4.44e-06 ***
## bs(log(St), df = 4)1     4.933      4.536   1.087  0.28109
## bs(log(St), df = 4)2    11.136      4.091   2.722  0.00841 **
## bs(log(St), df = 4)3     7.674      3.246   2.365  0.02120 *
## bs(log(St), df = 4)4     8.327      2.516   3.309  0.00156 **
## factor(Re)224           -8.011      1.107  -7.236 8.40e-10 ***
## factor(Re)398          -13.744      1.462  -9.404 1.52e-13 ***
## factor(Fr)0.3          -10.685      1.310  -8.153 2.15e-11 ***
## factor(Fr)Inf           -7.908      1.117  -7.082 1.55e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.057 on 62 degrees of freedom
## Multiple R-squared:  0.7675, Adjusted R-squared:  0.7375
## F-statistic: 25.58 on 8 and 62 DF,  p-value: < 2.2e-16
```

```
pred.spline2m4 <- predict(spline2m4, testing)
mse_spline2m4 <- mean((pred.spline2m4 - log(testing$R_moment_4))^2)
spline3m4 <- lm(log(R_moment_4) ~ bs(log(St), df=5) + factor(Re) + factor(Fr), data = training)
summary(spline3m4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ bs(log(St), df = 5) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -7.0554 -2.4879 -0.4701  2.6401  9.0583
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            10.5359     2.0704   5.089 3.70e-06 ***
## bs(log(St), df = 5)1    0.7063     5.3250   0.133  0.89491
## bs(log(St), df = 5)2   11.0977     3.5971   3.085  0.00306 **
## bs(log(St), df = 5)3    7.1141     2.9256   2.432  0.01798 *
## bs(log(St), df = 5)4    9.4775     3.4667   2.734  0.00818 **
## bs(log(St), df = 5)5    7.7845     2.5860   3.010  0.00379 **
## factor(Re)224          -8.0653     1.1104  -7.263 8.16e-10 ***
## factor(Re)398         -13.8665     1.4704  -9.430 1.60e-13 ***
## factor(Fr)0.3         -10.6192     1.3151  -8.075 3.27e-11 ***
## factor(Fr)Inf          -7.8663     1.1197  -7.026 2.09e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.064 on 61 degrees of freedom
## Multiple R-squared:  0.7705, Adjusted R-squared:  0.7366
## F-statistic: 22.76 on 9 and 61 DF,  p-value: 2.284e-16
```

```
pred.spline3m4 <- predict(spline3m4, testing)
mse_spline3m4 <- mean((pred.spline3m4 - log(testing$R_moment_4))^2)
spline4m4 <- lm(log(R_moment_4) ~ bs(log(St), df=6) + factor(Re) + factor(Fr), data = training)
summary(spline4m4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ bs(log(St), df = 6) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9436 -2.4784 -0.8562  2.9676  9.1064
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             10.502      2.089   5.027 4.80e-06 ***
## bs(log(St), df = 6)1     1.463      5.820   0.251   0.8024
## bs(log(St), df = 6)2     7.760      4.323   1.795   0.0777 .
## bs(log(St), df = 6)3     9.832      2.913   3.375   0.0013 **
## bs(log(St), df = 6)4     6.231      3.449   1.806   0.0759 .
## bs(log(St), df = 6)5    10.802      4.091   2.641   0.0105 *
## bs(log(St), df = 6)6     7.727      2.603   2.968   0.0043 **
## factor(Re)224           -8.123      1.120  -7.253 9.27e-10 ***
## factor(Re)398          -13.951      1.484  -9.403 2.09e-13 ***
## factor(Fr)0.3          -10.632      1.322  -8.040 4.18e-11 ***
## factor(Fr)Inf           -7.796      1.133  -6.880 4.00e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.084 on 60 degrees of freedom
## Multiple R-squared:  0.772,  Adjusted R-squared:  0.7341
## F-statistic: 20.32 on 10 and 60 DF,  p-value: 9.309e-16
```

```
pred.spline4m4 <- predict(spline4m4, testing)
mse_spline4m4 <- mean((pred.spline4m4 - log(testing$R_moment_4))^2)
spline5m4 <- lm(log(R_moment_4) ~ bs(log(St), df=7) + factor(Re) + factor(Fr), data = training)
summary(spline5m4)
```

```
##
## Call:
## lm(formula = log(R_moment_4) ~ bs(log(St), df = 7) + factor(Re) +
##     factor(Fr), data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.0487 -2.4752 -0.6338  2.7261  9.1034
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            10.5528     2.1146   4.990 5.64e-06 ***
## bs(log(St), df = 7)1    0.5787     6.3757   0.091  0.92799
## bs(log(St), df = 7)2    7.7837     4.4365   1.754  0.08454 .
## bs(log(St), df = 7)3    9.0110     2.9517   3.053  0.00340 **
## bs(log(St), df = 7)4    8.6249     3.0323   2.844  0.00611 **
## bs(log(St), df = 7)5    6.5057     5.4889   1.185  0.24067
## bs(log(St), df = 7)6   10.7158     5.2354   2.047  0.04514 *
## bs(log(St), df = 7)7    7.7074     2.6392   2.920  0.00495 **
## factor(Re)224          -8.1077     1.1323  -7.160 1.45e-09 ***
## factor(Re)398         -13.9591     1.5095  -9.248 4.43e-13 ***
## factor(Fr)0.3         -10.6487     1.3373  -7.963 6.32e-11 ***
## factor(Fr)Inf          -7.8358     1.1434  -6.853 4.80e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.126 on 59 degrees of freedom
## Multiple R-squared:  0.7712, Adjusted R-squared:  0.7285
## F-statistic: 18.07 on 11 and 59 DF,  p-value: 4.848e-15
```

```
pred.spline5m4 <- predict(spline5m4, testing)
mse_spline5m4 <- mean((pred.spline5m4 - log(testing$R_moment_4))^2)
mse_spline1m4
```

```
## [1] 20.67205
```

```
mse_spline2m4
```

```
## [1] 21.77453
```

```
mse_spline3m4
```

```
## [1] 23.19761
```

```
mse_spline4m4
```

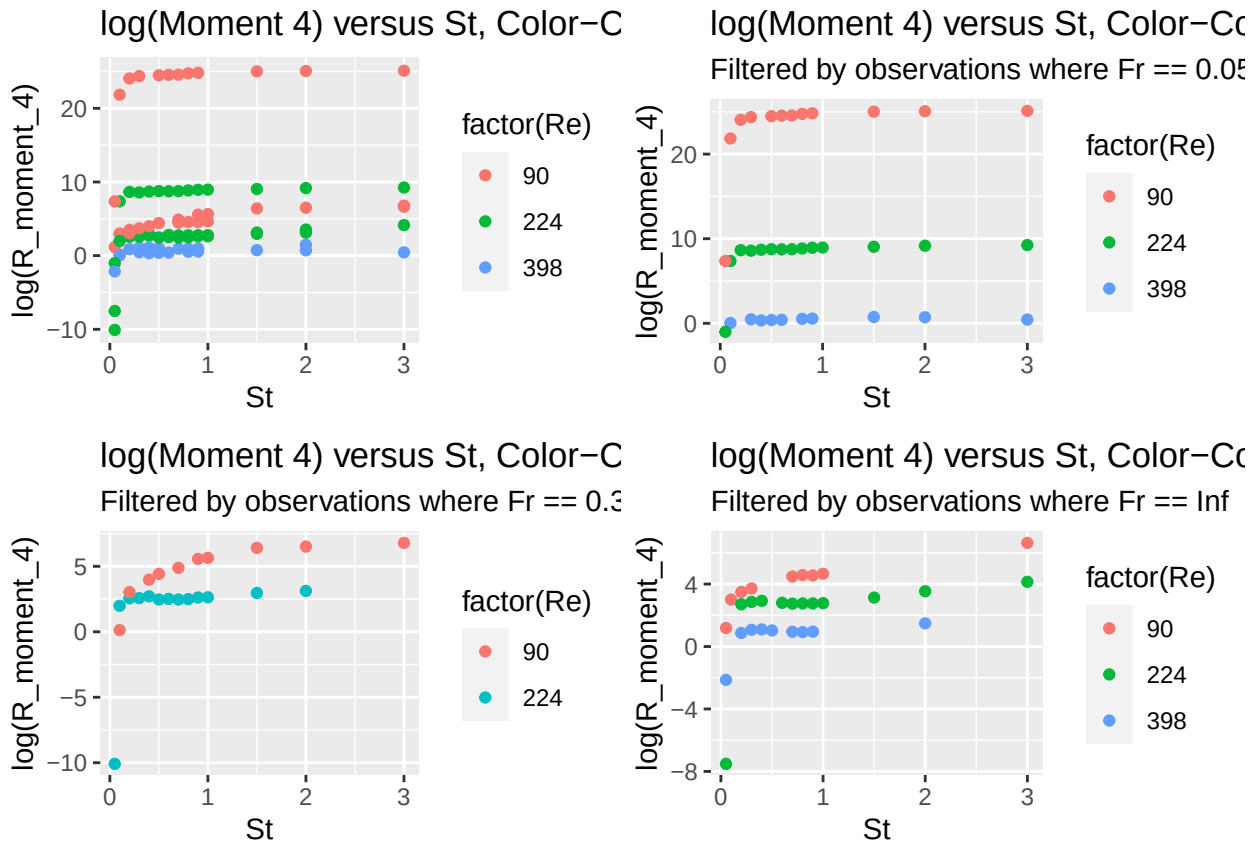
```
## [1] 23.18499
```

```
mse_spline5m4
```

```
## [1] 23.22217
```

Model for Predicting New Inputs

```
p1 = ggplot(data_train, aes(x=St, y=log(R_moment_4), color = factor(Re))) +
  geom_point() +
  labs(title = "log(Moment 4) versus St, Color-Coded by Re")
p2 = ggplot(filter(data_train, Fr==0.052), aes(x=St, y=log(R_moment_4), color = factor(Re))) +
  geom_point() +
  labs(title = "log(Moment 4) versus St, Color-Coded by Re", subtitle = "Filtered by observations where
p3 = ggplot(filter(data_train, Fr==0.3), aes(x=St, y=log(R_moment_4), color = factor(Re))) +
  geom_point() +
  labs(title = "log(Moment 4) versus St, Color-Coded by Re", subtitle = "Filtered by observations where
p4 = ggplot(filter(data_train, Fr==Inf), aes(x=St, y=log(R_moment_4), color = factor(Re))) +
  geom_point() +
  labs(title = "log(Moment 4) versus St, Color-Coded by Re", subtitle = "Filtered by observations where
eda.plots = ggarrange(p1, p2, p3, p4, ncol=2, nrow=2)
eda.plots
```



```
# perform transformation on Fr to get rid of infinity
g <- function(x) {
  return (1 - 2^(-x))
}

printModels <- function(models) {
  for (i in 1:length(models)) {
    print(paste0("R Moment ", i))
    print(knitr::kable(tidy(models[[i]]),
                       digits=3))
    stats = matrix(c(summary(models[[i]])$adj.r.squared, summary(models[[i]])$sigma),
                   ncol=2)
    colnames(stats) = c("Adj R^2", "Std Err")
    print(knitr::kable(stats, digits=3))
    cat('\n\n<!-- -->\n\n')
  }
}
```

```
moments = list(data_train$R_moment_1, data_train$R_moment_2, data_train$R_moment_3, data_train$R_moment
```

Try base linear model:

```
lm.models = vector("list",4)
for (i in 1:4) {
  model = lm(log(moments[[i]]) ~ St + Re + g(Fr), data=data_train)
  lm.models[[i]] = model
}
printModels(lm.models)
```

[1] "R Moment 1"

term	estimate	std.error	statistic	p.value
(Intercept)	-1.232	0.163	-7.578	0.000
St	0.268	0.081	3.309	0.001
Re	-0.019	0.001	-33.518	0.000
g(Fr)	0.058	0.147	0.395	0.694

Adj R^2	Std Err
0.929	0.597

[1] "R Moment 2"

term	estimate	std.error	statistic	p.value
(Intercept)	3.361	0.634	5.304	0.000
St	0.796	0.316	2.519	0.014
Re	-0.023	0.002	-10.443	0.000
g(Fr)	-1.989	0.574	-3.463	0.001

Adj R^2	Std Err
0.607	2.326

[1] “R Moment 3”

term	estimate	std.error	statistic	p.value
(Intercept)	8.862	1.134	7.817	0.000
St	1.083	0.565	1.917	0.059
Re	-0.028	0.004	-7.184	0.000
g(Fr)	-4.004	1.027	-3.897	0.000

Adj R^2	Std Err
0.463	4.162

[1] “R Moment 4”

term	estimate	std.error	statistic	p.value
(Intercept)	14.504	1.625	8.925	0.000
St	1.328	0.810	1.639	0.105
Re	-0.034	0.006	-6.007	0.000
g(Fr)	-5.997	1.473	-4.071	0.000

Adj R^2	Std Err
0.403	5.966

St doesn't even seem to be a significant predictor in some of these models. Linear regression might not work super well here.

Including all interaction terms:

```
glm.full.models = vector("list",4)
for (i in 1:4) {
  model = lm(log(moments[[i]]) ~ (St + Re + g(Fr))^2, data=data_train)
  glm.full.models[[i]] = model
}
printModels(glm.full.models)
```

[1] “R Moment 1”

term	estimate	std.error	statistic	p.value
(Intercept)	-1.056	0.252	-4.182	0.000
St	0.325	0.182	1.787	0.078
Re	-0.020	0.001	-18.850	0.000
g(Fr)	-0.442	0.363	-1.217	0.227

term	estimate	std.error	statistic	p.value
St:Re	0.000	0.001	-0.183	0.856
St:g(Fr)	-0.062	0.187	-0.333	0.740
Re:g(Fr)	0.002	0.001	1.917	0.059

Adj R^2	Std Err
0.93	0.593

[1] “R Moment 2”

term	estimate	std.error	statistic	p.value
(Intercept)	4.889	0.911	5.369	0.000
St	1.058	0.657	1.611	0.111
Re	-0.030	0.004	-7.977	0.000
g(Fr)	-6.246	1.309	-4.770	0.000
St:Re	-0.001	0.003	-0.347	0.729
St:g(Fr)	-0.068	0.673	-0.101	0.920
Re:g(Fr)	0.019	0.005	4.152	0.000

Adj R^2	Std Err
0.668	2.14

[1] “R Moment 3”

term	estimate	std.error	statistic	p.value
(Intercept)	11.753	1.621	7.250	0.000
St	1.468	1.170	1.255	0.213
Re	-0.042	0.007	-6.232	0.000
g(Fr)	-11.830	2.331	-5.075	0.000
St:Re	-0.001	0.005	-0.258	0.797
St:g(Fr)	-0.114	1.198	-0.095	0.924
Re:g(Fr)	0.035	0.008	4.285	0.000

Adj R^2	Std Err
0.55	3.809

[1] “R Moment 4”

term	estimate	std.error	statistic	p.value
(Intercept)	18.769	2.320	8.092	0.000
St	1.803	1.674	1.077	0.285
Re	-0.055	0.010	-5.620	0.000
g(Fr)	-17.335	3.335	-5.198	0.000

term	estimate	std.error	statistic	p.value
St:Re	-0.001	0.007	-0.197	0.844
St:g(Fr)	-0.172	1.714	-0.100	0.920
Re:g(Fr)	0.050	0.012	4.343	0.000

Adj R^2	Std Err
0.502	5.45

Re and g(Fr) appear to be significant, but St doesn't have any significant interactions.

A model with the interaction term for Re and Fr:

```
glm.interaction.models = vector("list",4)
for (i in 1:4) {
  model = lm(log(moments[[i]]) ~ St + Re*g(Fr), data=data_train)
  glm.interaction.models[[i]] = model
}
printModels(glm.interaction.models)
```

[1] "R Moment 1"

term	estimate	std.error	statistic	p.value
(Intercept)	-1.009	0.194	-5.191	0.000
St	0.275	0.080	3.455	0.001
Re	-0.020	0.001	-26.121	0.000
g(Fr)	-0.505	0.315	-1.603	0.113
Re:g(Fr)	0.002	0.001	2.013	0.047

Adj R^2	Std Err
0.931	0.587

[1] "R Moment 2"

term	estimate	std.error	statistic	p.value
(Intercept)	5.091	0.701	7.260	0.000
St	0.852	0.288	2.961	0.004
Re	-0.031	0.003	-11.307	0.000
g(Fr)	-6.358	1.136	-5.597	0.000
Re:g(Fr)	0.019	0.004	4.331	0.000

Adj R^2	Std Err
0.675	2.116

[1] "R Moment 3"

term	estimate	std.error	statistic	p.value
(Intercept)	12.029	1.248	9.638	0.000
St	1.186	0.512	2.316	0.023
Re	-0.043	0.005	-8.824	0.000
g(Fr)	-11.999	2.022	-5.935	0.000
Re:g(Fr)	0.035	0.008	4.453	0.000

Adj R^2	Std Err
0.56	3.766

[1] "R Moment 4"

term	estimate	std.error	statistic	p.value
(Intercept)	19.086	1.785	10.690	0.000
St	1.476	0.732	2.015	0.047
Re	-0.056	0.007	-7.925	0.000
g(Fr)	-17.563	2.892	-6.073	0.000
Re:g(Fr)	0.051	0.011	4.503	0.000

Adj R^2	Std Err
0.514	5.387

None of these models are performing well at all, let's try other methods.

Linear model using least squares & no interaction term

```
model1 <- lm(log(R_moment_1) ~ St + Re + g(Fr), data=training)
pred.lm1 <- predict(model1, testing)
mse_test1 <- mean((pred.lm1 - log(testing$R_moment_1))^2)
model2 <- lm(log(R_moment_2) ~ St + Re + g(Fr), data=training)
pred.lm2 <- predict(model2, testing)
mse_test2 <- mean((pred.lm2 - log(testing$R_moment_2))^2)
model3 <- lm(log(R_moment_3) ~ St + Re + g(Fr), data=training)
pred.lm3 <- predict(model3, testing)
mse_test3 <- mean((pred.lm3 - log(testing$R_moment_3))^2)
model4 <- lm(log(R_moment_4) ~ St + Re + g(Fr), data=training)
pred.lm4 <- predict(model4, testing)
mse_test4 <- mean((pred.lm4 - log(testing$R_moment_4))^2)
mse_test = matrix(c(mse_test1, mse_test2, mse_test3, mse_test4), ncol=4)
colnames(mse_test) = c("MSE 1", "MSE 2", "MSE 3", "MSE 4")
kable(mse_test, digits=3)
```

MSE 1	MSE 2	MSE 3	MSE 4
0.344	5.274	16.683	34.389

Linear model using least squares & interaction term

```
model1 <- lm(log(R_moment_1) ~ St + Re*g(Fr), data=training)
pred.lm1 <- predict(model1, testing)
mse_test1 <- mean((pred.lm1 - log(testing$R_moment_1))^2)
model2 <- lm(log(R_moment_2) ~ St + Re*g(Fr), data=training)
pred.lm2 <- predict(model2, testing)
mse_test2 <- mean((pred.lm2 - log(testing$R_moment_2))^2)
model3 <- lm(log(R_moment_3) ~ St + Re*g(Fr), data=training)
pred.lm3 <- predict(model3, testing)
mse_test3 <- mean((pred.lm3 - log(testing$R_moment_3))^2)
model4 <- lm(log(R_moment_4) ~ St + Re*g(Fr), data=training)
pred.lm4 <- predict(model4, testing)
mse_test4 <- mean((pred.lm4 - log(testing$R_moment_4))^2)
mse_test = matrix(c(mse_test1, mse_test2, mse_test3, mse_test4), ncol=4)
colnames(mse_test) = c("MSE 1", "MSE 2", "MSE 3", "MSE 4")
kable(mse_test, digits=3)
```

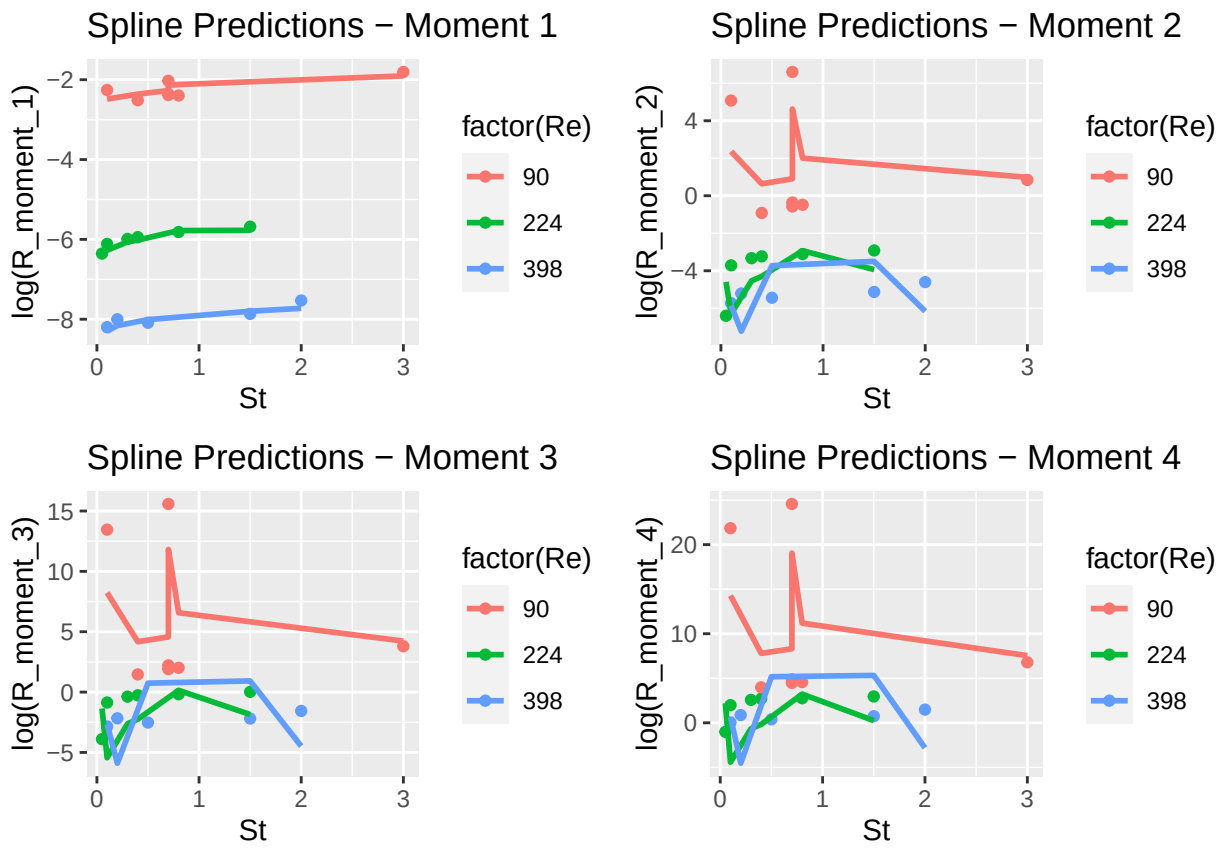
MSE 1	MSE 2	MSE 3	MSE 4
0.345	4.897	15.246	31.211

Really bad results, interaction terms aren't helping much. Let's try some other methods other than simple linear regression.

Splines

```
spline1 <- lm(log(R_moment_1) ~ bs(log(St)) + bs(Re) + bs(g(Fr)), data = training)
pred.spline1 <- predict(spline1, testing)
mse_spline1 <- mean((pred.spline1 - log(testing$R_moment_1))^2)
r1=summary(spline1)$adj.r.squared

spline2 <- lm(log(R_moment_2) ~ bs(log(St)) + bs(Re) + bs(g(Fr)), data = training)
pred.spline2 <- predict(spline2, testing)
mse_spline2 <- mean((pred.spline2 - log(testing$R_moment_2))^2)
r2=summary(spline2)$adj.r.squared

spline3 <- lm(log(R_moment_3) ~ bs(log(St)) + bs(Re) + bs(g(Fr)), data = training)
pred.spline3 <- predict(spline3, testing)
mse_spline3 <- mean((pred.spline3 - log(testing$R_moment_3))^2)
r3=summary(spline3)$adj.r.squared

spline4 <- lm(log(R_moment_4) ~ bs(log(St)) + bs(Re) + bs(g(Fr)), data = training)
pred.spline4 <- predict(spline4, testing)
```

```
mse_spline4 <- mean((pred.spline4 - log(testing$R_moment_4))^2)
r4=summary(spline4)$adj.r.squared

p1 = ggplot(testing, aes(x = St, y = log(R_moment_1), color = factor(Re))) +
  geom_point() +
  geom_line(aes(x = St, y = pred.spline1), size = 1) +
  labs(title = "Spline Predictions - Moment 1")
p2 = ggplot(testing, aes(x = St, y = log(R_moment_2), color = factor(Re))) +
  geom_point() +
  geom_line(aes(x = St, y = pred.spline2), size = 1) +
  labs(title = "Spline Predictions - Moment 2")
p3 = ggplot(testing, aes(x = St, y = log(R_moment_3), color = factor(Re))) +
  geom_point() +
  geom_line(aes(x = St, y = pred.spline3), size = 1) +
  labs(title = "Spline Predictions - Moment 3")
p4 = ggplot(testing, aes(x = St, y = log(R_moment_4), color = factor(Re))) +
  geom_point() +
  geom_line(aes(x = St, y = pred.spline4), size = 1) +
  labs(title = "Spline Predictions - Moment 4")
spline.graphs = ggarrange(p1, p2, p3, p4, ncol=2, nrow=2)
spline.graphs
```




```
mse_spline = rbind(c(mse_spline1, mse_spline2, mse_spline3, mse_spline4), c(r1, r2, r3, r4))
colnames(mse_spline) = c("Moment 1", "Moment 2", "Moment 3", "Moment 4")
rownames(mse_spline) = c("MSE", "Ajd R^2")
```

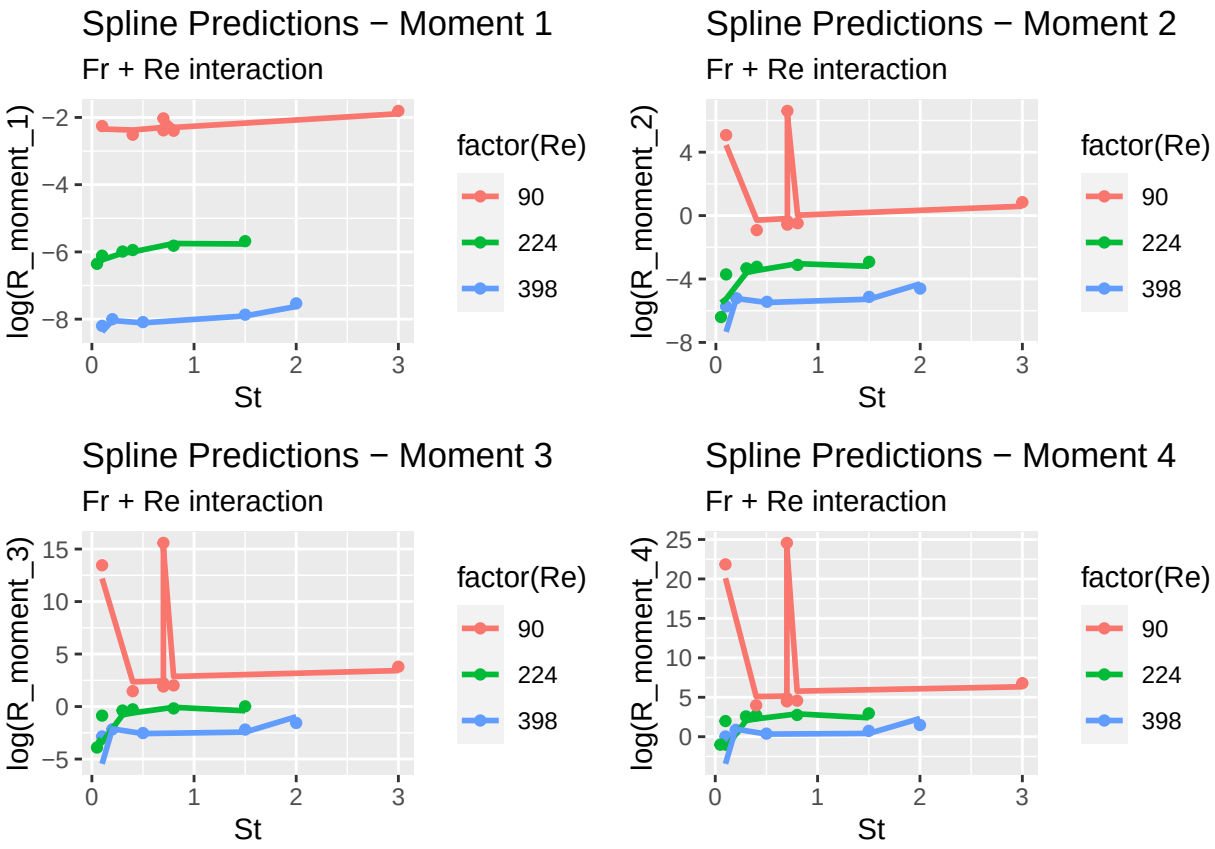
```
kable(mse_spline, digits=3)
```

	Moment 1	Moment 2	Moment 3	Moment 4
MSE	0.021	2.978	9.973	20.672
Ajd R^2	0.997	0.844	0.772	0.741

Splines definitely perform better than linear regression, but still room for improvement.

Splines with interaction between Fr and Re

```
spline1 <- lm(log(R_moment_1) ~ bs(log(St)) + bs(Re)*bs(g(Fr)), data = training)
pred.spline1 <- predict(spline1, testing)
mse_spline1 <- mean((pred.spline1 - log(testing$R_moment_1))^2)
r1=summary(spline1)$adj.r.squared

spline2 <- lm(log(R_moment_2) ~ bs(log(St)) + bs(Re)*bs(g(Fr)), data = training)
pred.spline2 <- predict(spline2, testing)
mse_spline2 <- mean((pred.spline2 - log(testing$R_moment_2))^2)
r2=summary(spline2)$adj.r.squared

spline3 <- lm(log(R_moment_3) ~ bs(log(St)) + bs(Re)*bs(g(Fr)), data = training)
pred.spline3 <- predict(spline3, testing)
mse_spline3 <- mean((pred.spline3 - log(testing$R_moment_3))^2)
r3=summary(spline3)$adj.r.squared

spline4 <- lm(log(R_moment_4) ~ bs(log(St)) + bs(Re)*bs(g(Fr)), data = training)
pred.spline4 <- predict(spline4, testing)
mse_spline4 <- mean((pred.spline4 - log(testing$R_moment_4))^2)
r4=summary(spline4)$adj.r.squared

p1 = ggplot(testing, aes(x = St, y = log(R_moment_1), color = factor(Re))) +
  geom_point() +
  geom_line(aes(x = St, y = pred.spline1), size = 1) +
  labs(title = "Spline Predictions - Moment 1", subtitle = "Fr + Re interaction")
p2 = ggplot(testing, aes(x = St, y = log(R_moment_2), color = factor(Re))) +
  geom_point() +
  geom_line(aes(x = St, y = pred.spline2), size = 1) +
  labs(title = "Spline Predictions - Moment 2", subtitle = "Fr + Re interaction")
p3 = ggplot(testing, aes(x = St, y = log(R_moment_3), color = factor(Re))) +
  geom_point() +
  geom_line(aes(x = St, y = pred.spline3), size = 1) +
  labs(title = "Spline Predictions - Moment 3", subtitle = "Fr + Re interaction")
p4 = ggplot(testing, aes(x = St, y = log(R_moment_4), color = factor(Re))) +
  geom_point() +
  geom_line(aes(x = St, y = pred.spline4), size = 1) +
  labs(title = "Spline Predictions - Moment 4", subtitle = "Fr + Re interaction")
interaction.spline.graphs = ggarrange(p1, p2, p3, p4, ncol=2, nrow=2)
interaction.spline.graphs
```




```
mse_spline = rbind(c(mse_spline1, mse_spline2, mse_spline3, mse_spline4), c(r1, r2, r3, r4))
colnames(mse_spline) = c("Moment 1", "Moment 2", "Moment 3", "Moment 4")
rownames(mse_spline) = c("MSE", "Ajd R^2")
kable(mse_spline, digits=3)
```

	Moment 1	Moment 2	Moment 3	Moment 4
MSE	0.007	0.424	1.077	1.895
Ajd R^2	0.998	0.970	0.962	0.961

Adding the interaction term to the splines makes the performance of this model superior to anything we've used previously (when considering numerical datapoints) based on Adj R^2 and MSE in test validation.

Methodology (to include in writeup)