Final Writeup

Kadriye Sude Almus, Cheyenne Kim, Martin Lim, Carrie Wang, Michael Li

10/25/2020

The Data

```
data <- read.csv("data/data-train.csv")
```

Exploratory Data Analysis

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

```
names(data)
```

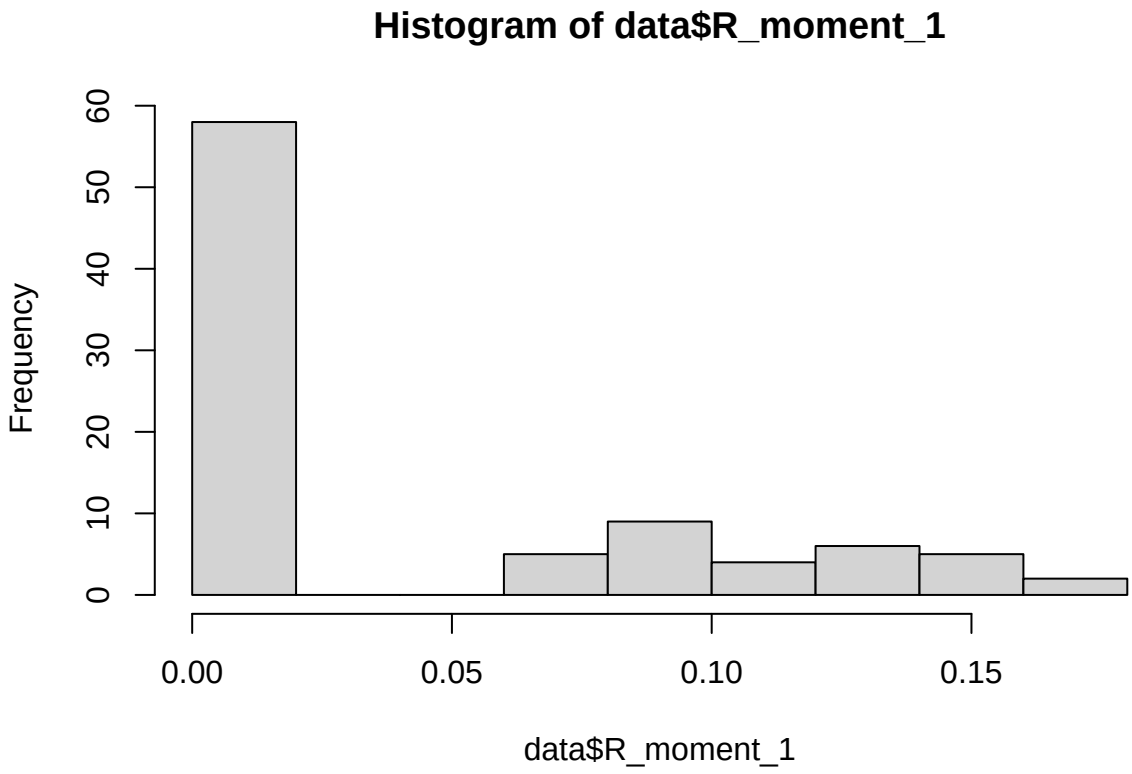
```
## [1] "St"         "Re"         "Fr"         "R_moment_1" "R_moment_2"
## [6] "R_moment_3" "R_moment_4"
```

```
summary(data)
```

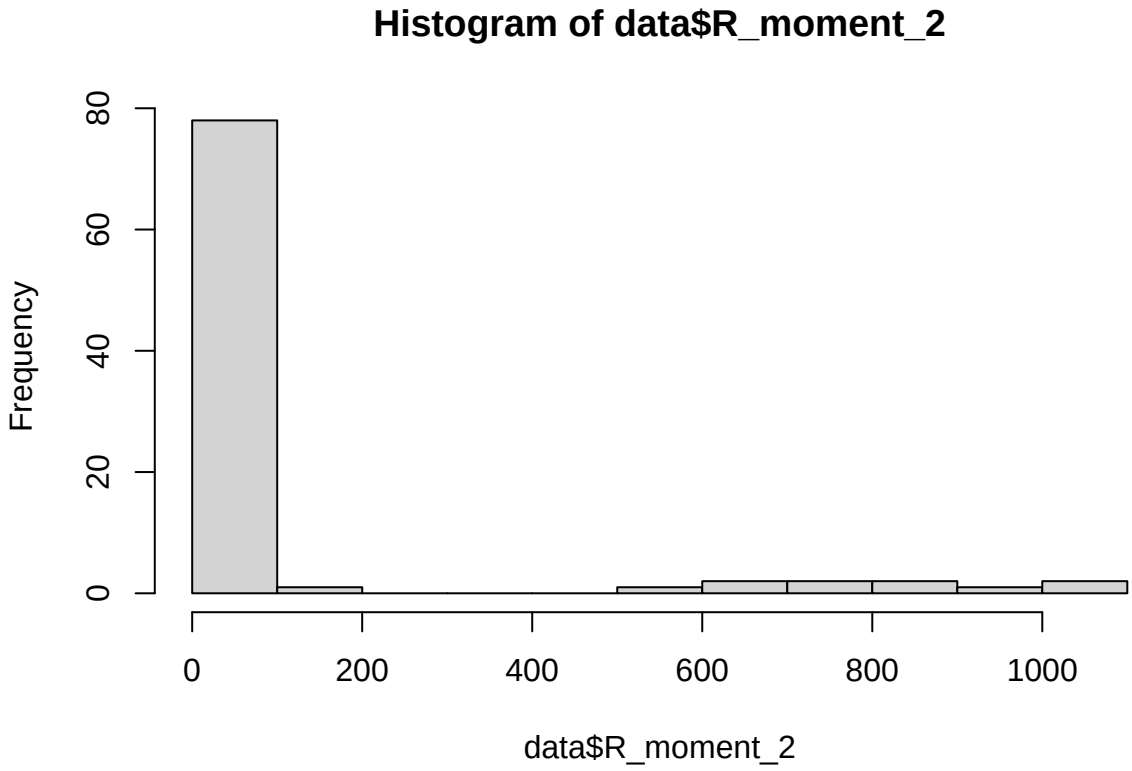
```
##        St                Re               Fr          R_moment_1
##  Min.   :0.0500   Min.   : 90.0   Min.   :0.052   Min.   :0.000222
##  1st Qu.:0.3000   1st Qu.: 90.0   1st Qu.:0.052   1st Qu.:0.002157
##  Median :0.7000   Median :224.0   Median :0.300   Median :0.002958
##  Mean   :0.8596   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

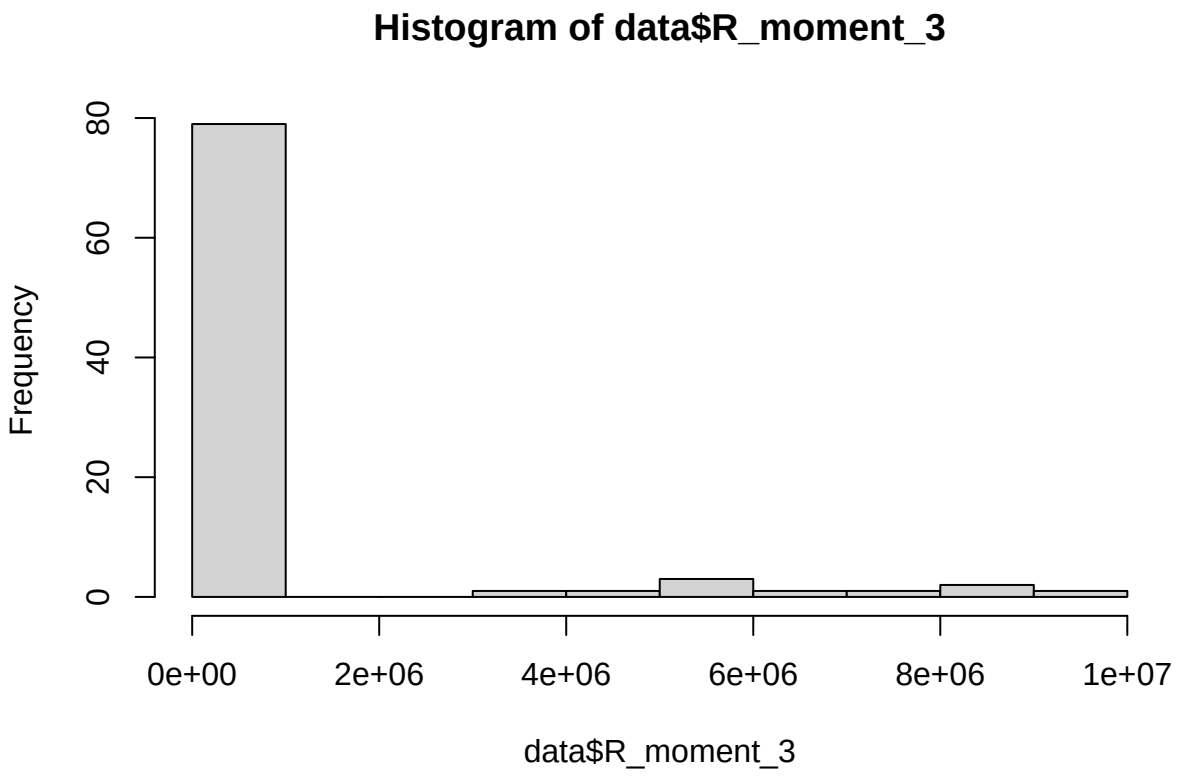
```
hist(data$R_moment_1)
```



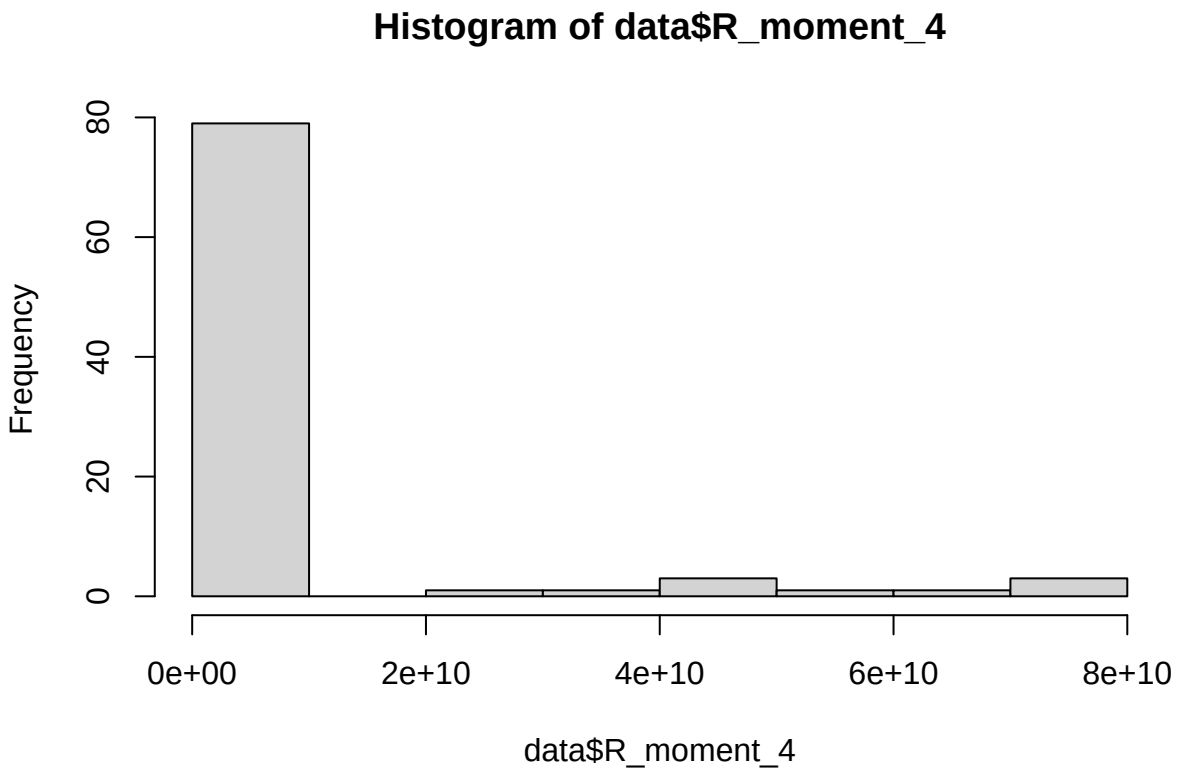

```
hist(data$R_moment_2)
```



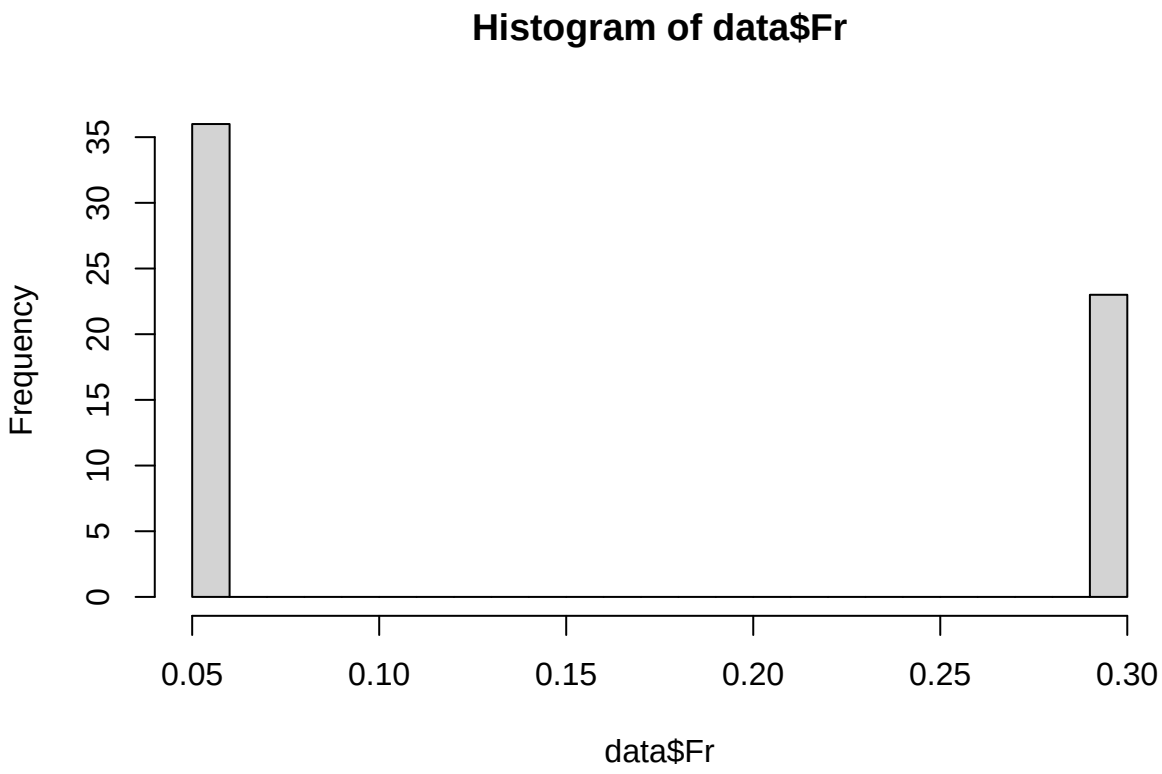

```
hist(data$R_moment_3)
```



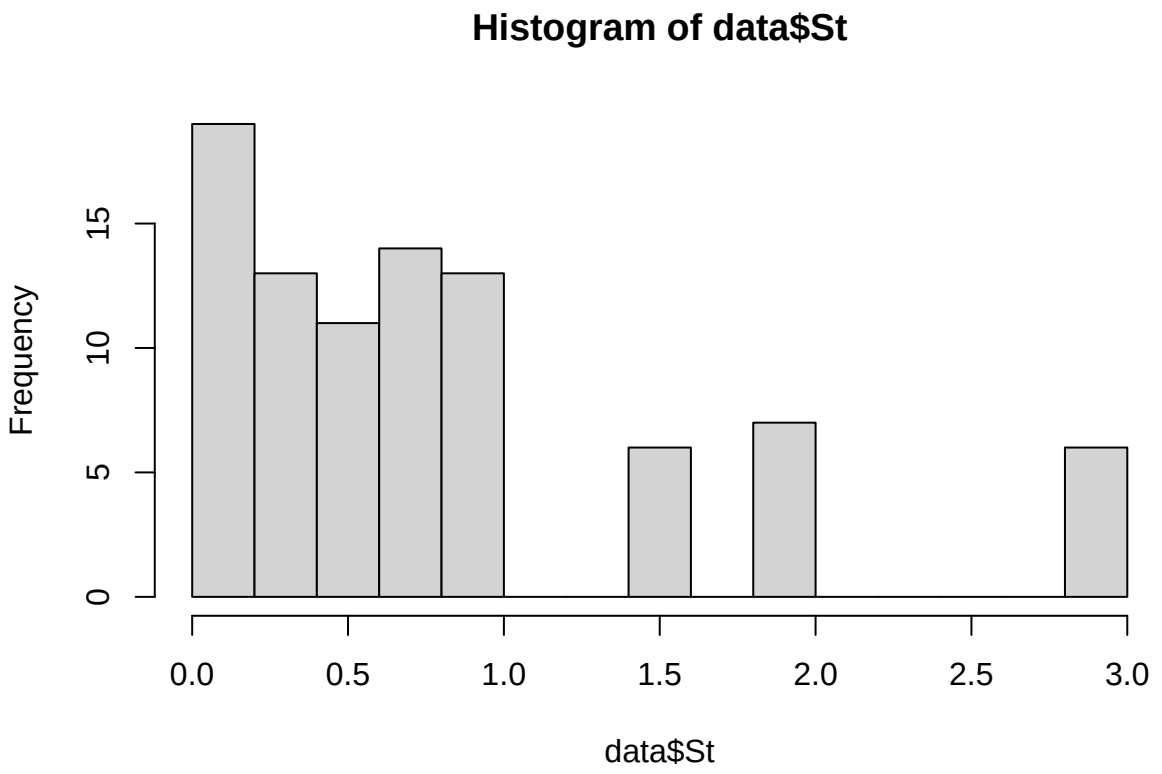

```
hist(data$R_moment_4)
```



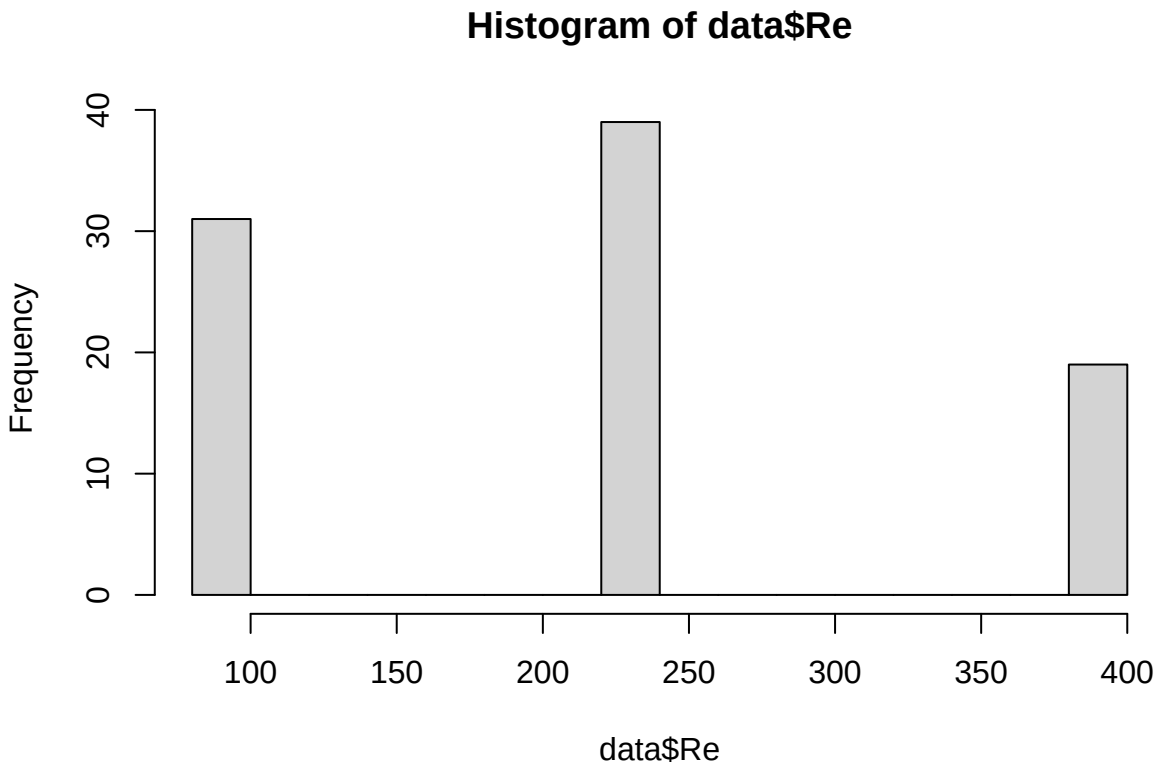

```
hist(data$Fr, breaks=20)
```




```
hist(data$St, breaks=20)
```

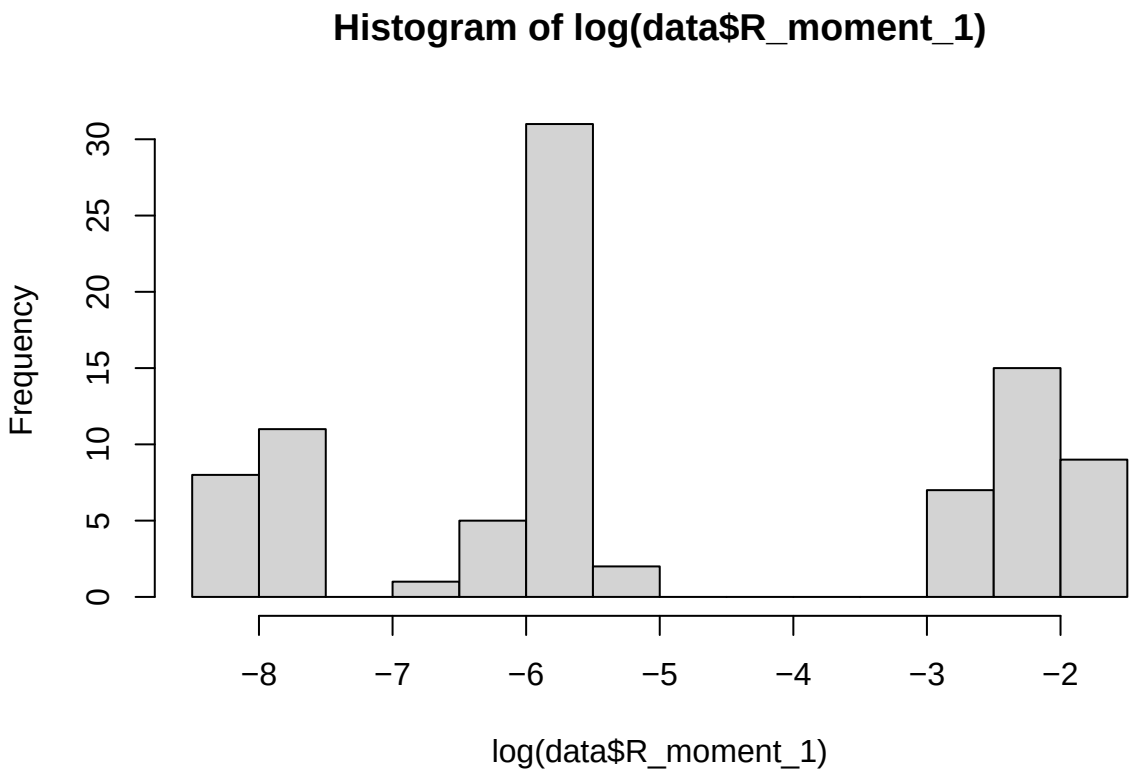



```
hist(data$Re, breaks=20)
```

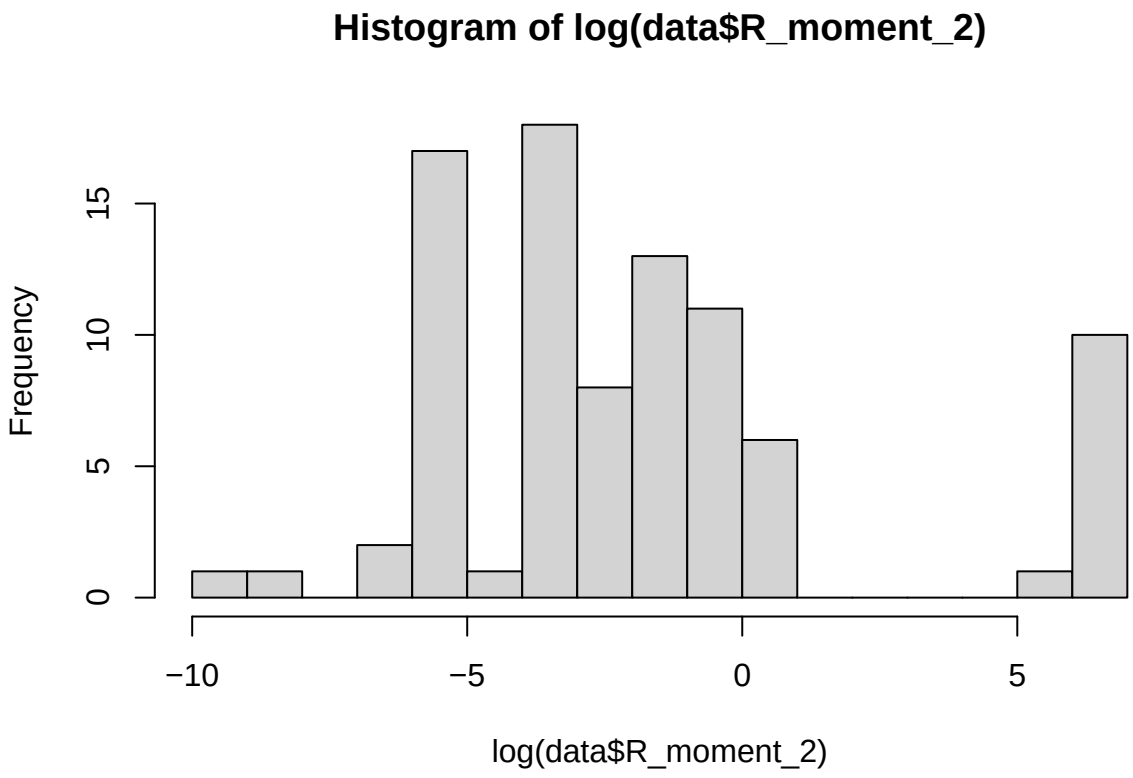


With these histograms its clear to see that each R_moment is heavily right skewed, since there are many rows of 0 in the data. In R_moment_3 and R_moment_4, the maximum values are extremely high whereas the medians are much smaller in comparison, which poses a problem to the analysis. We believe it is best then to apply a transformation to these variables in order to obtain more accurate analysis.

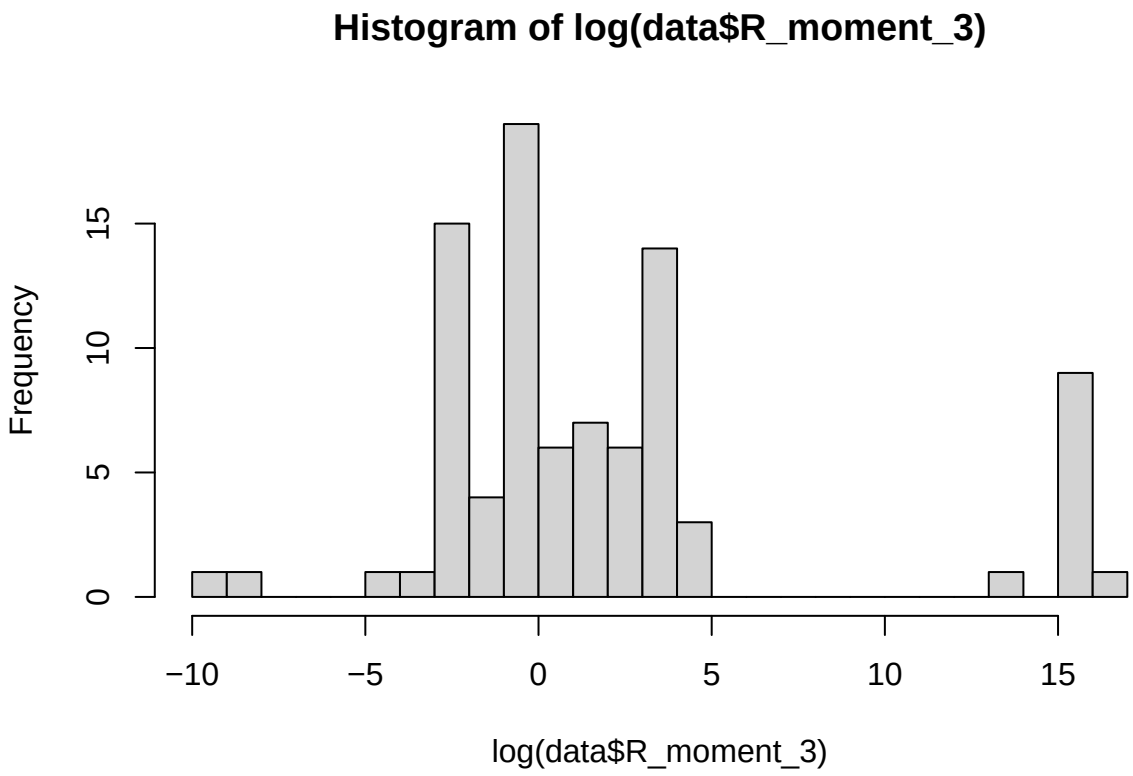
```
hist(log(data$R_moment_1), breaks=20)
```



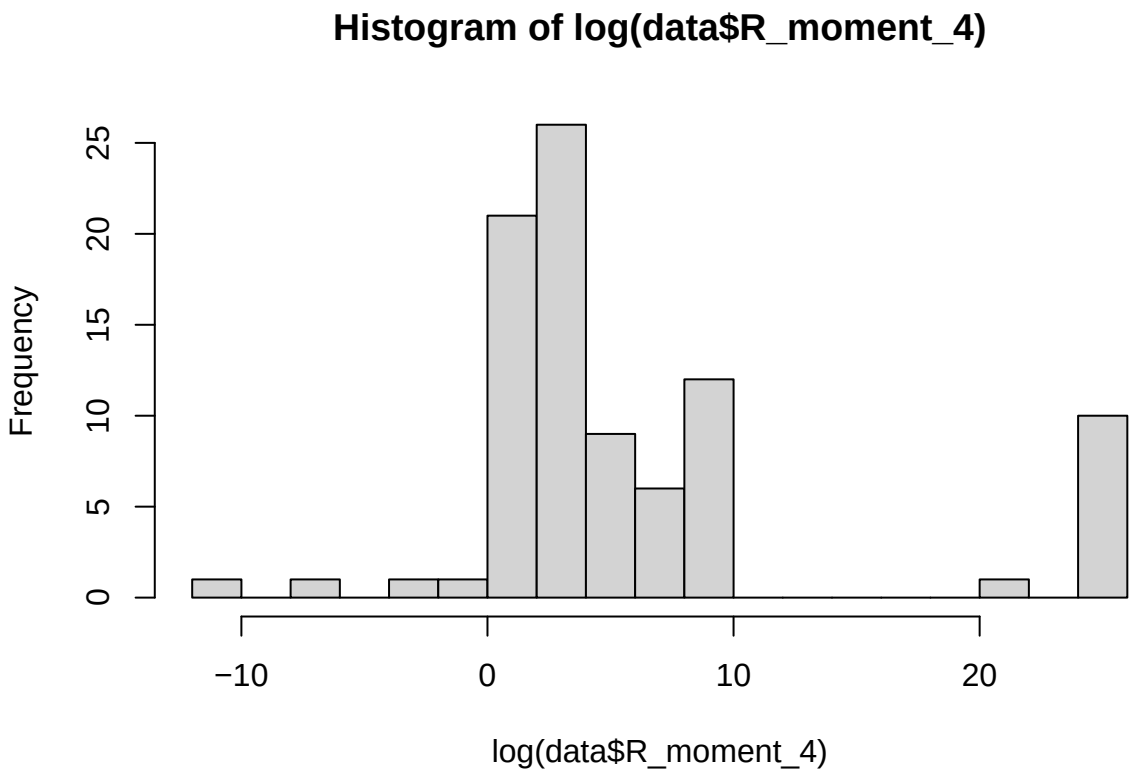

```
hist(log(data$R_moment_2), breaks=20)
```




```
hist(log(data$R_moment_3), breaks=20)
```



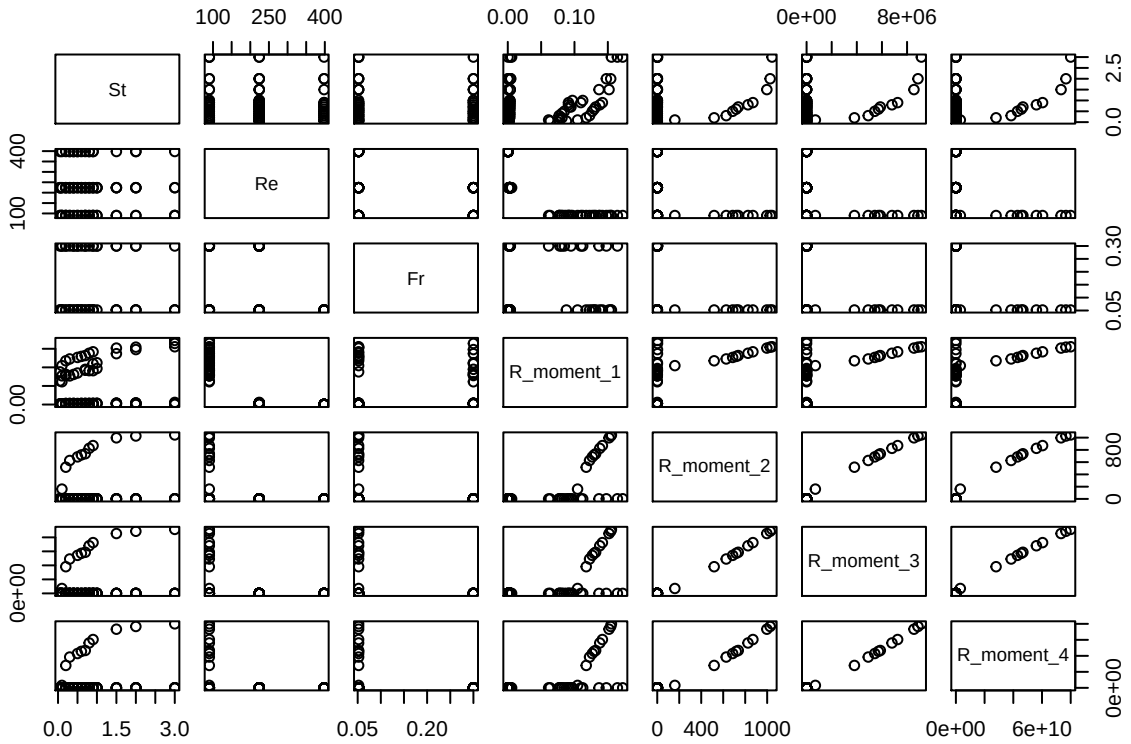

```
hist(log(data$R_moment_4), breaks=20)
```



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. From here on out, the log version of variables will be used and will be reflected as such in our interpretations and analysis.

One thing that we should do is turn Fr and Re into ordered, categorical variables, because they only have 2 or 3 unique values each.

```
pairs(data)
```



It appears that each R_moment variable has somewhat of a linear relationship with St.

Initial Modelling

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)

summary(model1)
```

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

summary(model2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ St + factor(Re) + factor(Fr),
##     data = data)
##
## 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)

summary(model3)
```

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

summary(model4)
```

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

```r
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
```

When all interaction terms are included:

```r
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)
##
## 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)
##
## 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)
##
## 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)
##
## 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. We will attempt to only include the interaction term for Re and Fr.

A model with the interaction term for Re and Fr:

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

```
## Response log(R_moment_1) :
##
## Call:
## lm(formula = `log(R_moment_1)` ~ (St + factor(Re) + factor(Fr) +
##     factor(Re) * factor(Fr)), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.48592 -0.00915  0.03880  0.07277  0.17182
##
```

```
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                 -2.27306    0.04110  -55.299  < 2e-16 ***
## St                           0.24989    0.01803   13.863  < 2e-16 ***
## factor(Re)224               -3.81588    0.05160  -73.948  < 2e-16 ***
## factor(Re)398               -5.98854    0.05621 -106.548  < 2e-16 ***
## factor(Fr)0.3               -0.26297    0.05622   -4.678 1.16e-05 ***
## factor(Fr)Inf               -0.32944    0.05787   -5.693 1.99e-07 ***
## factor(Re)224:factor(Fr)0.3  0.22050    0.07574    2.911  0.00466 **
## factor(Re)398:factor(Fr)0.3       NA         NA       NA       NA
## factor(Re)224:factor(Fr)Inf  0.40185    0.07759    5.179 1.63e-06 ***
## factor(Re)398:factor(Fr)Inf  0.50151    0.08366    5.995 5.58e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1312 on 80 degrees of freedom
## Multiple R-squared:  0.9969, Adjusted R-squared:  0.9966
## F-statistic:  3193 on 8 and 80 DF,  p-value: < 2.2e-16
##
##
## Response log(R_moment_2) :
##
## Call:
## lm(formula = `log(R_moment_2)` ~ (St + factor(Re) + factor(Fr) +
##     factor(Re) * factor(Fr)), data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8551 -0.0206  0.3104  0.5102  1.0043
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    5.1869     0.3843  13.498  < 2e-16 ***
## St                             0.8340     0.1685   4.949 4.06e-06 ***
## factor(Re)224                 -7.4387     0.4824 -15.420  < 2e-16 ***
## factor(Re)398                -11.3837     0.5254 -21.665  < 2e-16 ***
## factor(Fr)0.3                 -6.4163     0.5256 -12.208  < 2e-16 ***
## factor(Fr)Inf                 -6.6523     0.5410 -12.297  < 2e-16 ***
## factor(Re)224:factor(Fr)0.3    4.3872     0.7081   6.196 2.37e-08 ***
## factor(Re)398:factor(Fr)0.3       NA         NA      NA       NA
## factor(Re)224:factor(Fr)Inf    4.7181     0.7254   6.504 6.25e-09 ***
## factor(Re)398:factor(Fr)Inf    7.0758     0.7821   9.047 7.09e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.226 on 80 degrees of freedom
## Multiple R-squared:  0.9008, Adjusted R-squared:  0.8909
## F-statistic: 90.79 on 8 and 80 DF,  p-value: < 2.2e-16
##
##
## Response log(R_moment_3) :
##
## Call:
## lm(formula = `log(R_moment_3)` ~ (St + factor(Re) + factor(Fr) +
```

```
##     factor(Re) * factor(Fr)), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.3570  -0.0586   0.4564   0.8018   1.6559
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   13.3986     0.6241  21.469  < 2e-16 ***
## St                             1.1740     0.2737   4.290 4.97e-05 ***
## factor(Re)224                -11.1636     0.7835 -14.249  < 2e-16 ***
## factor(Re)398                -17.0302     0.8534 -19.957  < 2e-16 ***
## factor(Fr)0.3                -12.4781     0.8536 -14.618  < 2e-16 ***
## factor(Fr)Inf                -12.7719     0.8786 -14.536  < 2e-16 ***
## factor(Re)224:factor(Fr)0.3   8.3648     1.1500   7.274 2.10e-10 ***
## factor(Re)398:factor(Fr)0.3       NA         NA      NA       NA
## factor(Re)224:factor(Fr)Inf   8.7718     1.1781   7.446 9.76e-11 ***
## factor(Re)398:factor(Fr)Inf  13.3707     1.2702  10.527  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.991 on 80 degrees of freedom
## Multiple R-squared:  0.8882, Adjusted R-squared:  0.877
## F-statistic: 79.44 on 8 and 80 DF,  p-value: < 2.2e-16
##
##
## Response log(R_moment_4) :
##
## Call:
## lm(formula = `log(R_moment_4)` ~ (St + factor(Re) + factor(Fr) +
##     factor(Re) * factor(Fr)), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.4109   0.0031   0.5741   1.0506   2.2382
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   21.6950     0.8354  25.971  < 2e-16 ***
## St                             1.4690     0.3663   4.010 0.000135 ***
## factor(Re)224                -14.9060     1.0487 -14.214  < 2e-16 ***
## factor(Re)398                -22.7148     1.1422 -19.886  < 2e-16 ***
## factor(Fr)0.3                -18.4708     1.1425 -16.166  < 2e-16 ***
## factor(Fr)Inf                -18.8106     1.1760 -15.995  < 2e-16 ***
## factor(Re)224:factor(Fr)0.3  12.2758     1.5393   7.975 9.06e-12 ***
## factor(Re)398:factor(Fr)0.3       NA         NA      NA       NA
## factor(Re)224:factor(Fr)Inf  12.7559     1.5769   8.089 5.40e-12 ***
## factor(Re)398:factor(Fr)Inf  19.5683     1.7001  11.510  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.666 on 80 degrees of freedom
## Multiple R-squared:  0.8917, Adjusted R-squared:  0.8809
## F-statistic: 82.34 on 8 and 80 DF,  p-value: < 2.2e-16
```

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

Split data into training and test sets

```
attach(data)
set.seed(3)
train_ind <- sample(x = nrow(data), size = 0.8 * nrow(data))
test_ind_neg <- -train_ind
training <- data[train_ind, ]
testing <- data[test_ind_neg, ]
ftraining <- training
ftesting <- testing
ftraining$Fr <- factor(ftraining$Fr, levels = c(0.052, 0.300, Inf))
ftraining$Re <- factor(ftraining$Re, levels = c(90, 224, 398))
ftesting$Fr <- factor(ftesting$Fr, levels = c(0.052, 0.300, Inf))
ftesting$Re <- factor(ftesting$Re, levels = c(90, 224, 398))
```

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
```

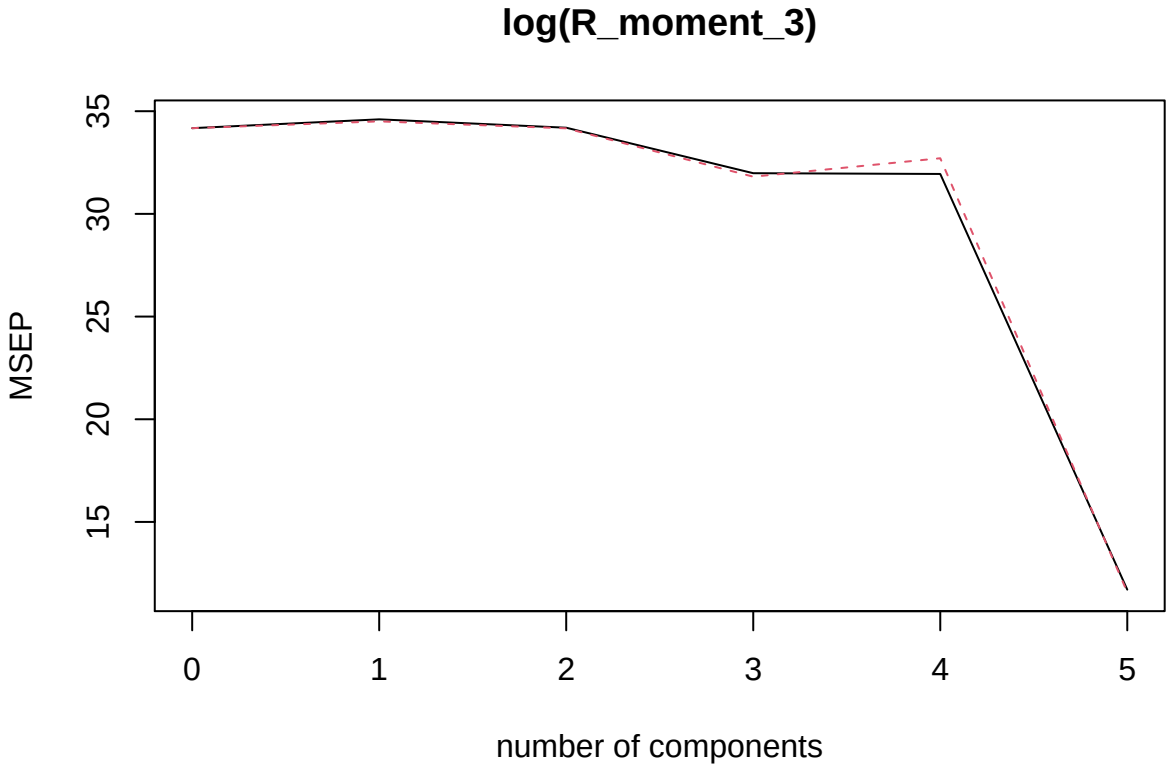
Having an interaction term significantly improved the test MSEs of the linear model.

Other Linear Regularization Techniques

We would now like to explore linear model regularization techniques on the higher moments to see if any produce a better adj. R^2 or test MSE value than the least squares with an interaction term.

Trying PCR model on the 3rd moment:

```
fit3.pcr <- pcr(log(R_moment_3) ~ (St + factor(Re) + factor(Fr)), data = training, scale = TRUE, validation
validationplot(fit3.pcr, val.type = "MSEP")
```



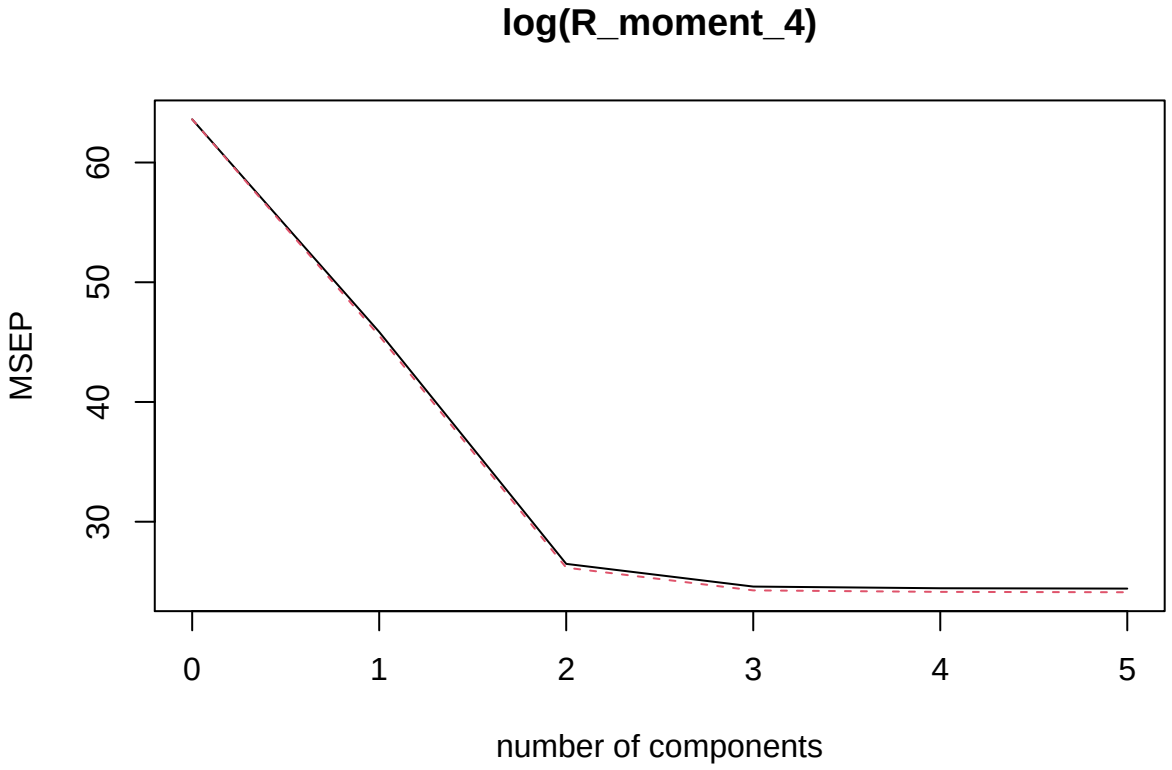

```
pred3.pcr <- predict(fit3.pcr, testing, ncomp = 5)
mean((pred3.pcr - log(testing$R_moment_3))^2)
```

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

The test MSE is the same as the least squares linear model with no interaction term. It has a higher test MSE and lower adj. R^2 than the linear model with the interaction term added.

Trying PLS on the 4th moment:

```
fit4.pls <- plsr(log(R_moment_4) ~ (St + factor(Re) + factor(Fr)), data = training, validation = "CV")
validationplot(fit4.pls, val.type = "MSEP")
```




```
predict4.pls<-predict(fit4.pls,testing,ncomp=5)
mean((predict4.pls - log(testing$R_moment_4))^2)
```

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

The test MSE is the same as the least squares linear model with no interaction term. It has a higher test MSE and lower adj. R^2 than the linear model with the interaction term added.

Nonlinear Techniques

For ease of perusal, we have inserted our unused/ineffective model selection techniques below as plain text instead of R code.

Regression Tree

library(tree) tree1 <- tree(R_moment_1 ~ St + factor(Re) + factor(Fr), data = training)

cv1 <- cv.tree(tree1) plot(cv1$size, cv1$dev, type = "b") abline(h = min(cv1$dev) + 1 * sd(cv1$dev), col = "red", lty = 2)

y_hat <- predict(tree1, newdata = testing) moment1_test <- testing[,"R_moment_1"] plot(y_hat, moment1_test) abline(0,1) test_error <- mean((y_hat-moment1_test)^2) test_error tss <- mean((testing$R_moment_1 - mean(testing$R_moment_1))^2) (rss <- 1 - test_error / tss) ### Regrssion tree does not work well on higher moments tree2 <- tree(R_moment_2 ~ St + factor(Re) + factor(Fr), data = training) summary(tree2) plot(tree2) text(tree2, pretty = 0)

cv2 <- cv.tree(tree2) plot(cv2$size, cv2$dev, type = "b") abline(h = min(cv2$dev) + 1 * sd(cv2$dev), col = "red", lty = 2)

y_hat <- predict(tree2, newdata = testing) moment2_test <- testing[,"R_moment_2"] plot(y_hat, moment2_test) abline(0,1) (test_error <- mean((y_hat-moment2_test)^2)) test_error tss <- mean((testing$R_moment_2 - mean(testing$R_moment_2))^2) (rss <- 1 - test_error / tss)

Random Forest

library(randomForest) set.seed(120) rf_mom2 <- randomForest(R_moment_2 ~ St + Re, data = training, ntree = 25, importance = TRUE) summary(rf_mom2)

yhat_rf <- predict(rf_mom2, newdata = testing) plot(yhat_rf, moment2_test) abline(0,1) mean((yhat_rf - moment2_test)^2)

importance(rf_mom2) varImpPlot(rf_mom2)

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)

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

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

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

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

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

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

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

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 mse_poly2m1 mse_poly3m1 mse_poly4m1 mse_poly5m1 mse_poly6m1 mse_poly7m1

Similar to least squares.

Second moment: polym2 <- lm(log(R_moment_2) ~ poly(St, 2) + factor(Re) + factor(Fr), data = training) poly2m2 <- lm(log(R_moment_2) ~ poly(St, 3) + factor(Re) + factor(Fr), data = training)

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

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

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

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

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

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

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 mse_polym2 mse_poly2m2 mse_poly3m2 mse_poly4m2 mse_poly5m2 mse_poly6m2 mse_poly7m2 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)

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

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

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

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

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

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

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

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

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 mse_polym3 mse_poly2m3 mse_poly3m3 mse_poly4m3 mse_poly5m3 mse_poly6m3 mse_poly7m3 mse_poly8m3

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)

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

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

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

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

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

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

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

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

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 mse_polym4 mse_poly2m4 mse_poly3m4 mse_poly4m4 mse_poly5m4 mse_poly6m4 mse_poly7m4 mse_poly8m4 The linear regression fit seems to have the minimal MSE for the fourth order.

Attempting splines:

library(splines)

First moment: spline1 <- lm(log(R_moment_1) ~ bs(log(St)) + factor(Re) + factor(Fr), data = training) 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) 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) 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) 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) pred.spline5 <- predict(spline5, testing) mse_spline5 <- mean((pred.spline5 - log(testing$R_moment_1))^2)

mse_spline1 mse_spline2 mse_spline3 mse_spline4 mse_spline5

Second moment:

spline1m2 <- lm(log(R_moment_2) ~ bs(log(St)) + factor(Re) + factor(Fr), data = training) summary(spline1m2) 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) 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) 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) 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) pred.spline5m2 <- predict(spline5m2, testing) mse_spline5m2 <- mean((pred.spline5m2 - log(testing$R_moment_2))^2)

mse_spline1m2 mse_spline2m2 mse_spline3m2 mse_spline4m2 mse_spline5m2

Third moment:

spline1m3 <- lm(log(R_moment_3) ~ bs(log(St)) + factor(Re) + factor(Fr), data = training) summary(spline1m3) 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) 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) 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) 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) pred.spline5m3 <- predict(spline5m3, testing) mse_spline5m3 <- mean((pred.spline5m3 - log(testing$R_moment_3))^2)

mse_spline1m3 mse_spline2m3 mse_spline3m3 mse_spline4m3 mse_spline5m3

Fourth moment: spline1m4 <- lm(log(R_moment_4) ~ bs(log(St)) + factor(Re) + factor(Fr), data = training) 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) 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) 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) 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) pred.spline5m4 <- predict(spline5m4, testing) mse_spline5m4 <- mean((pred.spline5m4 - log(testing$R_moment_4))^2)

mse_spline1m4 mse_spline2m4 mse_spline3m4 mse_spline4m4 mse_spline5m4

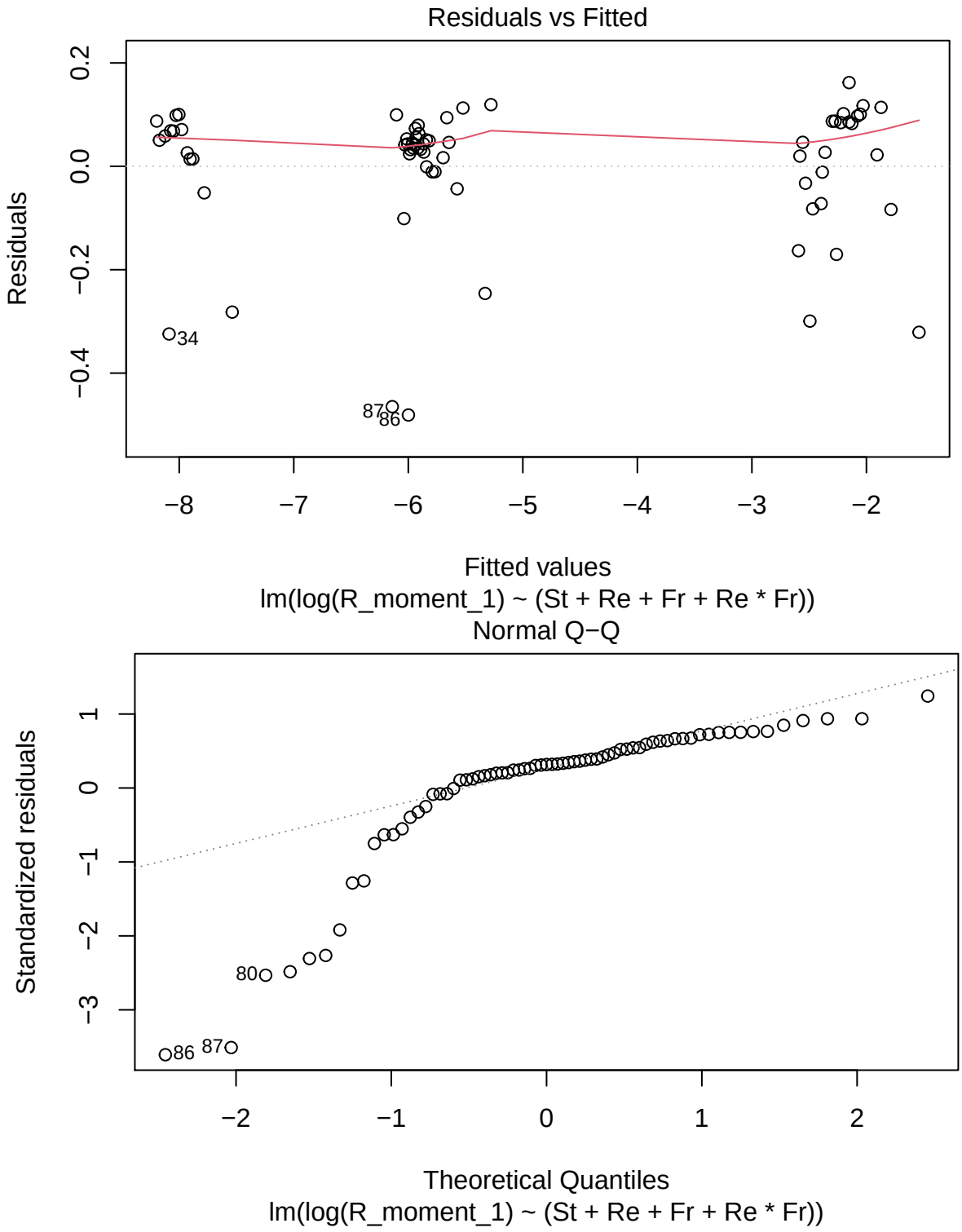
The Generalized Additive Model with a spline on St is shown below.

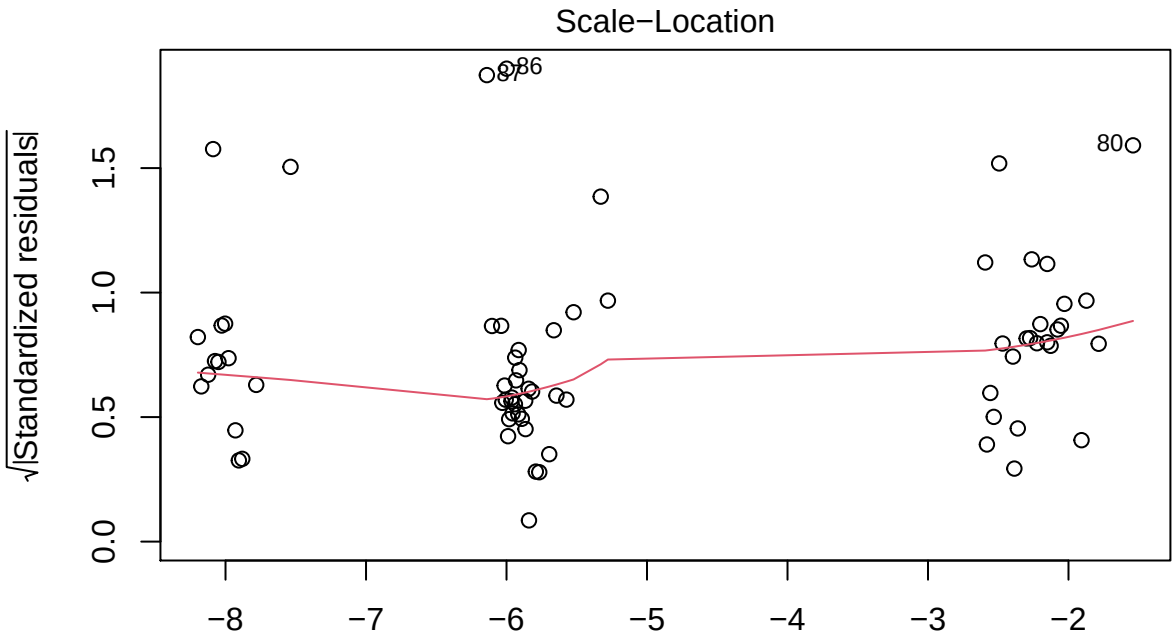
Final Model and Predictions

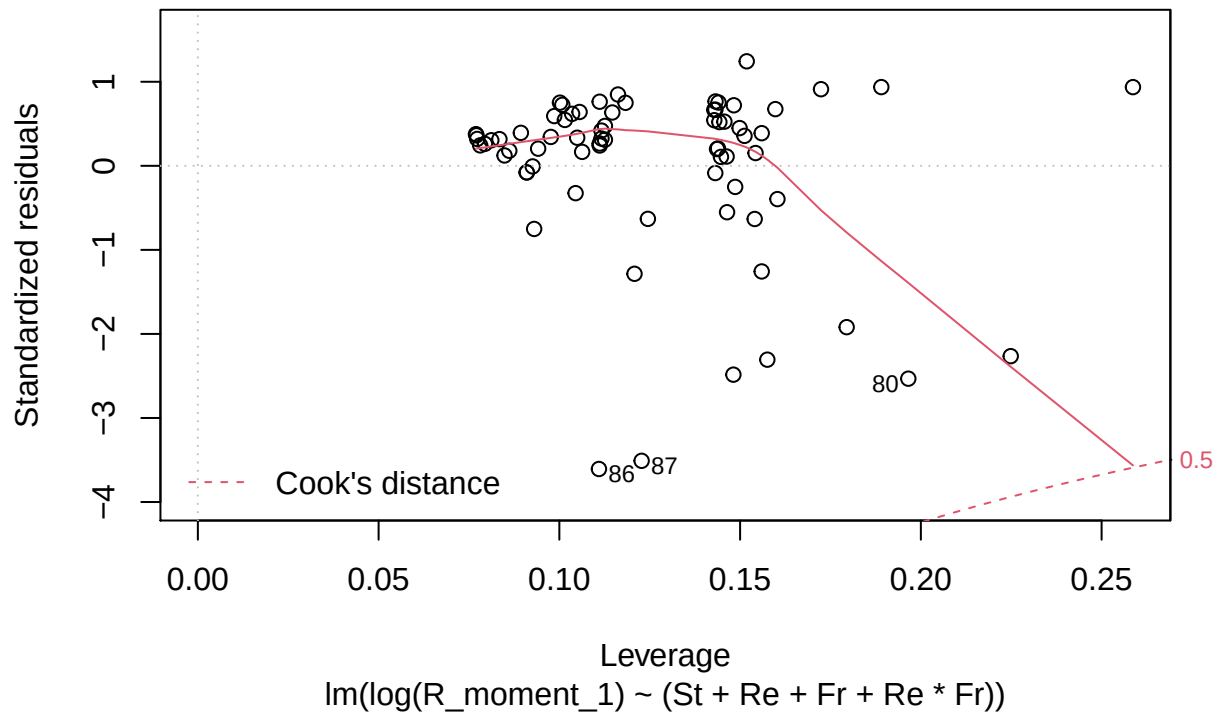
```
fit.lm1 <- lm(log(R_moment_1) ~ (St + Re + Fr + Re*Fr), data = ftraining)
pred.lm1 <- predict(fit.lm1, ftesting)
```

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

```
plot(fit.lm1)
```


Residuals vs Fitted
Residuals
0.2
0.0
−0.2
−0.4
34
87
86
−8
−7
−6
−5
−4
−3
−2
Fitted values
lm(log(R_moment_1) ~ (St + Re + Fr + Re * Fr))
Normal Q−Q
Standardized residuals
1
0
−1
−2
−3
80
86
87
−2
−1
0
1
2
Theoretical Quantiles
lm(log(R_moment_1) ~ (St + Re + Fr + Re * Fr))






```
summary(fit.lm1)
```

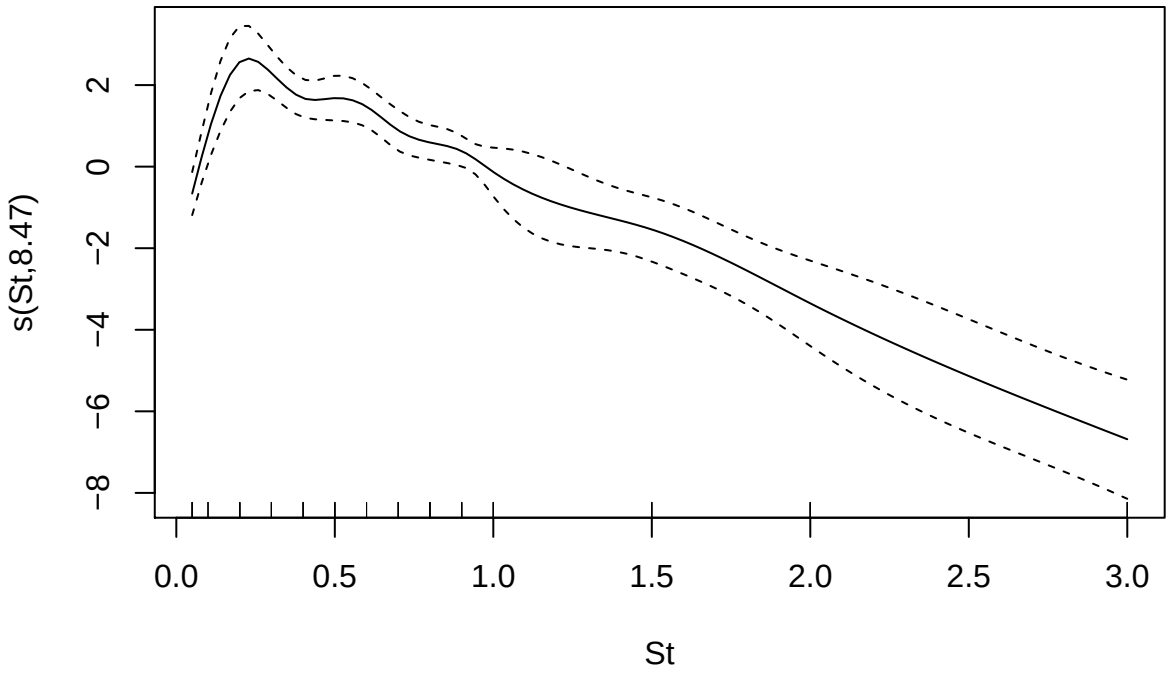
```
##
## Call:
## lm(formula = log(R_moment_1) ~ (St + Re + Fr + Re * Fr), data = ftraining)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.48100 -0.01056  0.04300  0.08114  0.16195
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.27376    0.04961 -45.828  < 2e-16 ***
## St           0.24431    0.02181  11.203  < 2e-16 ***
## Re224       -3.78857    0.05950 -63.669  < 2e-16 ***
## Re398       -5.99652    0.06978 -85.932  < 2e-16 ***
## Fr0.3       -0.24403    0.06973  -3.500 0.000869 ***
## FrInf       -0.33146    0.06982  -4.747 1.26e-05 ***
## Re224:Fr0.3  0.15420    0.09285   1.661 0.101811
## Re398:Fr0.3       NA         NA      NA       NA
## Re224:FrInf  0.38257    0.09078   4.214 8.28e-05 ***
## Re398:FrInf  0.50112    0.10325   4.853 8.55e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1414 on 62 degrees of freedom
## Multiple R-squared:  0.9963, Adjusted R-squared:  0.9959
## F-statistic:  2106 on 8 and 62 DF,  p-value: < 2.2e-16
```

```
mse_test1 <- mean((pred.lm1 - log(testing$R_moment_1))^2)
mse_test1
```

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

```
gam.m2 = gam(log(R_moment_2) ~ s(St) + Re + Fr + St:Re + St:Fr + Re:Fr, data = ftraining)
plot(gam.m2)
```

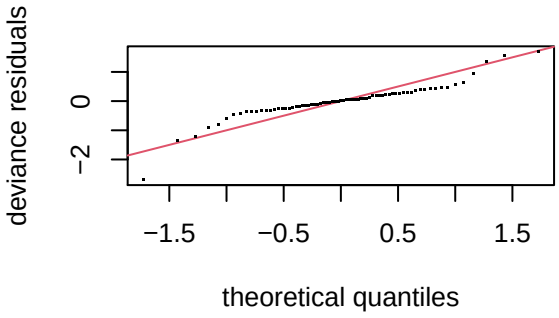


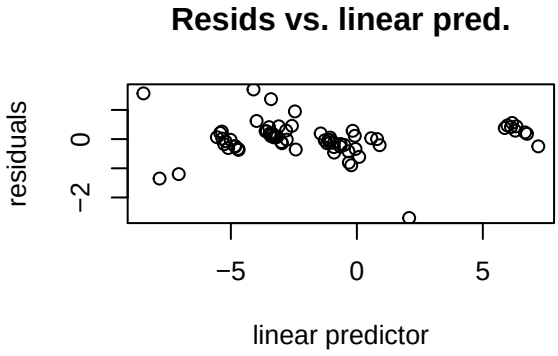

```
summary(gam.m2)
```

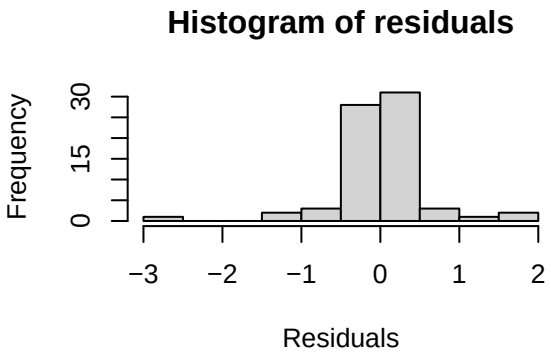
```
##
```

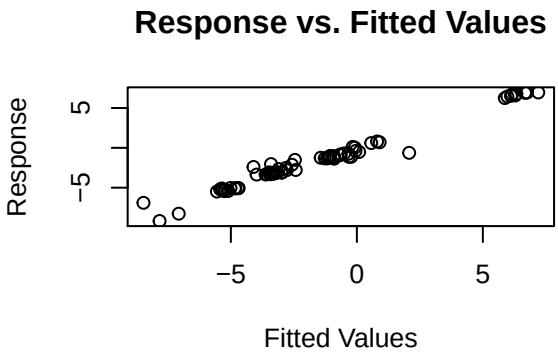
```
## Family: gaussian
## Link function: identity
##
## Formula:
## log(R_moment_2) ~ s(St) + Re + Fr + St:Re + St:Fr + Re:Fr
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.54657    0.37518   6.788 1.24e-08 ***
## Re224        -7.11591    0.37585 -18.933  < 2e-16 ***
## Re398       -11.08527    0.52216 -21.230  < 2e-16 ***
## Fr0.3        -6.87397    0.49181 -13.977  < 2e-16 ***
## FrInf        -6.17375    0.42687 -14.463  < 2e-16 ***
## Re90:St       3.77928    0.35437  10.665 1.60e-14 ***
## Re224:St      3.63936    0.36826   9.883 2.19e-13 ***
## Re398:St      3.26669    0.38455   8.495 2.71e-11 ***
## Fr0.3:St      0.51261    0.38567   1.329    0.190
## FrInf:St     -0.07419    0.25302  -0.293    0.771
## Re224:Fr0.3   4.06642    0.47277   8.601 1.86e-11 ***
## Re398:Fr0.3   0.00000    0.00000      NA       NA
## Re224:FrInf   4.15555    0.46108   9.013 4.39e-12 ***
## Re398:FrInf   6.74397    0.55523  12.146  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df     F p-value
## s(St) 8.467  8.799 17.03  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 21/23
## R-sq.(adj) =  0.965   Deviance explained = 97.5%
## GCV = 0.69646  Scale est. = 0.49431   n = 71
```

```
gam.check(gam.m2)
```



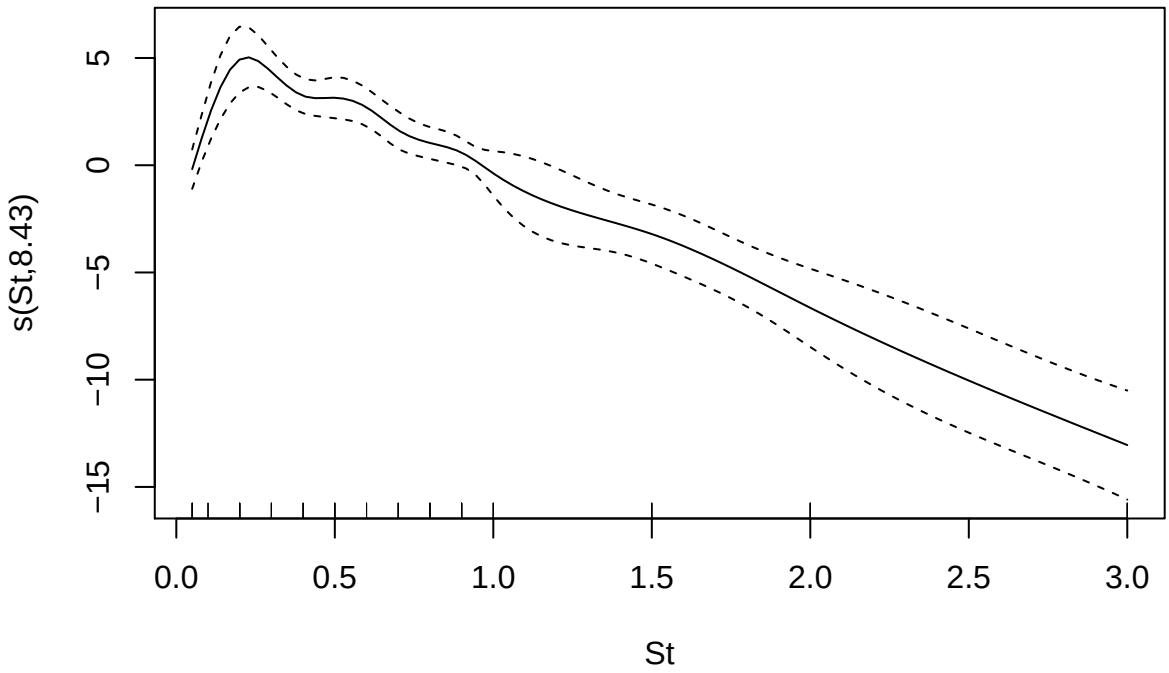







```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 8 iterations.
## The RMS GCV score gradient at convergence was 3.715048e-07 .
## The Hessian was positive definite.
## Model rank =  21 / 23
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##         k'  edf k-index p-value
## s(St) 9.00 8.47    1.08    0.71
```

```
gam.m3 = gam(log(R_moment_3) ~ s(St) + Re + Fr + St:Re + St:Fr + Re:Fr, data = ftraining)
plot(gam.m3)
```

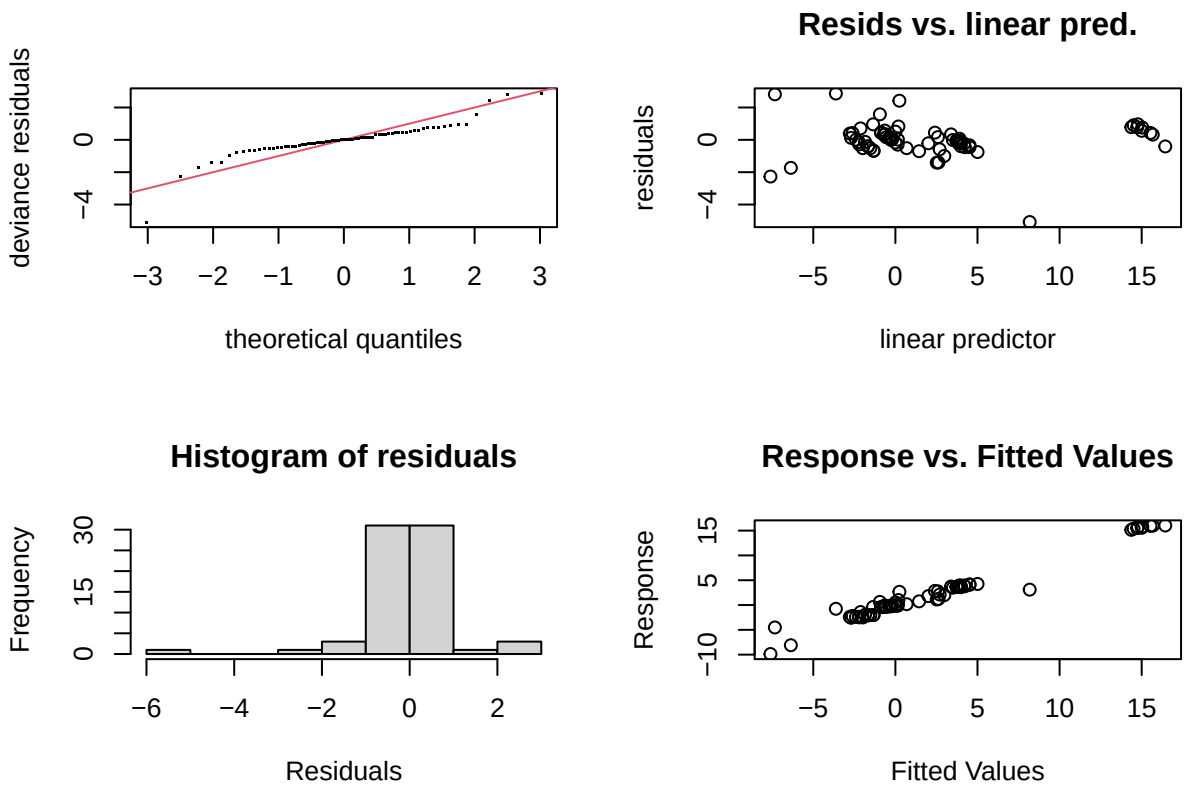



```
summary(gam.m3)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(R_moment_3) ~ s(St) + Re + Fr + St:Re + St:Fr + Re:Fr
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.0050     0.6534  12.251  < 2e-16 ***
## Re224       -10.6005     0.6575 -16.124  < 2e-16 ***
## Re398       -16.4306     0.9132 -17.993  < 2e-16 ***
## Fr0.3       -13.0638     0.8601 -15.188  < 2e-16 ***
## FrInf       -11.7892     0.7467 -15.788  < 2e-16 ***
## Re90:St       7.1596     0.6164  11.614 7.29e-16 ***
## Re224:St      6.8820     0.6412  10.734 1.26e-14 ***
## Re398:St      6.2334     0.6705   9.296 1.63e-12 ***
## Fr0.3:St      0.6865     0.6746   1.018    0.314
## FrInf:St     -0.3003     0.4426  -0.678    0.501
## Re224:Fr0.3   7.8560     0.8270   9.499 8.08e-13 ***
## Re398:Fr0.3   0.0000     0.0000      NA       NA
## Re224:FrInf   7.8729     0.8066   9.761 3.30e-13 ***
## Re398:FrInf  12.7630     0.9712  13.141  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df     F p-value
## s(St) 8.435  8.788 19.87  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

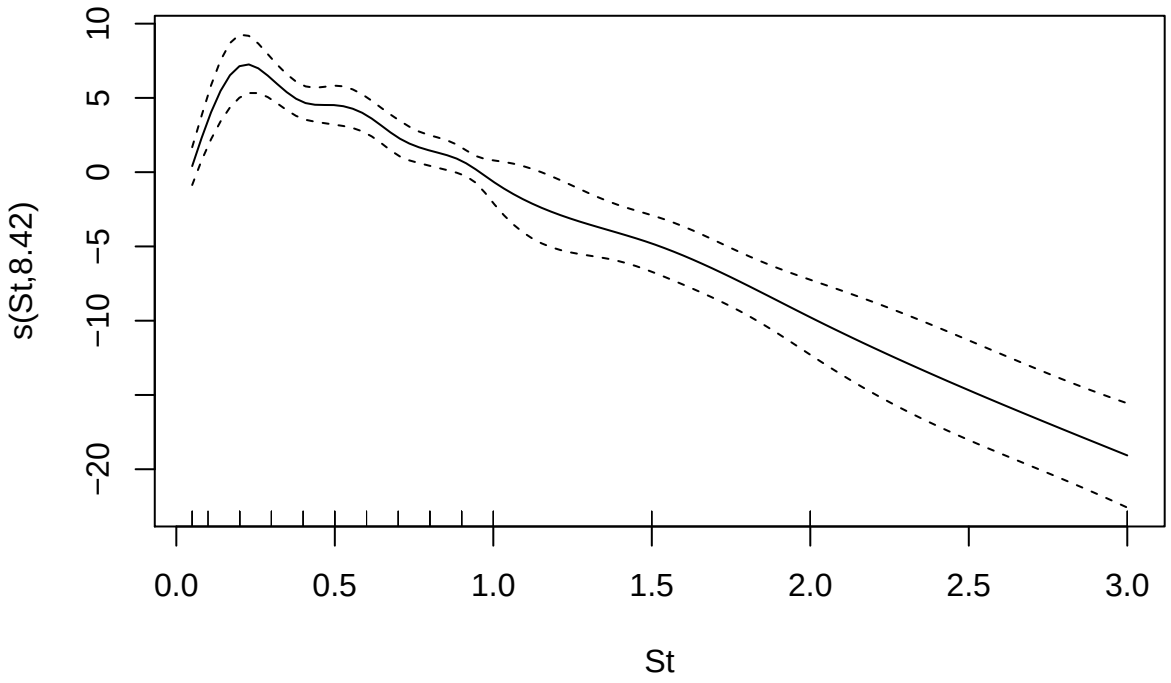
```
##
## Rank: 21/23
## R-sq.(adj) =  0.955   Deviance explained = 96.8%
## GCV =   2.13  Scale est. = 1.5127    n = 71
```

```
gam.check(gam.m3)
```




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 7 iterations.
## The RMS GCV score gradient at convergence was 8.319095e-05 .
## The Hessian was positive definite.
## Model rank =  21 / 23
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##         k'  edf k-index p-value
## s(St) 9.00 8.43    1.08    0.75
```

```
gam.m4 = gam(log(R_moment_4) ~ s(St) + Re + Fr + St:Re + St:Fr + Re:Fr, data = ftraining)
plot(gam.m4)
```

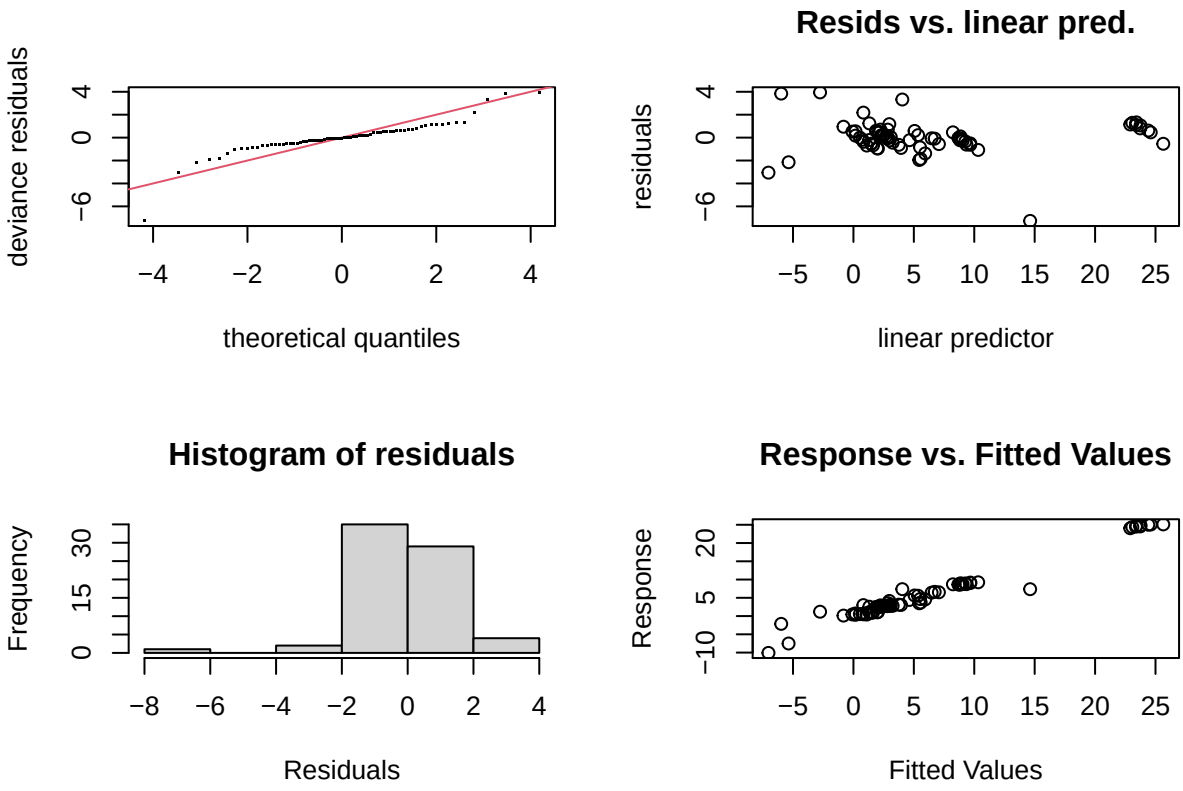



```
summary(gam.m4)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(R_moment_4) ~ s(St) + Re + Fr + St:Re + St:Fr + Re:Fr
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  13.6971     0.9028  15.171  < 2e-16 ***
## Re224       -14.1498     0.9106 -15.538  < 2e-16 ***
## Re398       -21.8583     1.2647 -17.284  < 2e-16 ***
## Fr0.3       -19.1353     1.1913 -16.063  < 2e-16 ***
## FrInf       -17.3592     1.0342 -16.784  < 2e-16 ***
## Re90:St      10.3389     0.8511  12.148  < 2e-16 ***
## Re224:St      9.9495     0.8857  11.234 2.46e-15 ***
## Re398:St      9.0499     0.9271   9.762 3.27e-13 ***
## Fr0.3:St      0.8019     0.9343   0.858    0.395
## FrInf:St     -0.5199     0.6131  -0.848    0.400
## Re224:Fr0.3  11.6078     1.1455  10.134 9.28e-14 ***
## Re398:Fr0.3   0.0000     0.0000      NA       NA
## Re224:FrInf  11.5630     1.1172  10.350 4.49e-14 ***
## Re398:FrInf  18.7047     1.3453  13.904  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df     F p-value
## s(St) 8.416  8.782 22.52  <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 21/23
## R-sq.(adj) =  0.954   Deviance explained = 96.7%
## GCV = 4.0851  Scale est. = 2.9023    n = 71
```

```
gam.check(gam.m4)
```




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 7 iterations.
## The RMS GCV score gradient at convergence was 0.0001068624 .
## The Hessian was positive definite.
## Model rank =  21 / 23
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##         k'  edf k-index p-value
## s(St) 9.00 8.42    1.08    0.73
```

```
pred.gam2 <- predict(gam.m2, ftesting)
pred.gam3 <- predict(gam.m3, ftesting)
pred.gam4 <- predict(gam.m4, ftesting)

mse_gam2 <- mean((pred.gam2 - log(ftesting$R_moment_2))^2)
mse_gam3 <- mean((pred.gam3 - log(ftesting$R_moment_3))^2)
mse_gam4 <- mean((pred.gam4 - log(ftesting$R_moment_4))^2)

mse_gam2
```

```
## [1] 0.8611115
```

```
mse_gam3
```

```
## [1] 2.292613
```

```
mse_gam4
```

```
## [1] 4.100509
```

Predictions on the Test Data

```
datat <- read.csv("data/data-test.csv")
datat$Fr <- factor(datat$Fr, levels = c(0.052, 0.300, Inf))
datat$Re <- factor(datat$Re, levels = c(90, 224, 398))
```

```
predt.lm1 <- predict(fit.lm1, datat)
```

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

```
predt.gam2 <- predict(gam.m2, datat)
predt.gam3 <- predict(gam.m3, datat)
predt.gam4 <- predict(gam.m4, datat)
```

Prediction Consolidation

Creating a csv of predictions:

```
dataframe_all <- bind_rows(predt.lm1, predt.gam2, predt.gam3, predt.gam4)
dataframe_all <- exp(dataframe_all)
dataframe_all
```

```
## # A tibble: 4 x 23
##       `1`     `2`     `3`     `4`     `5`     `6`     `7`     `8`     `9`
##     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 2.59e-4 2.69e-4 3.04e-4 3.27e-4 3.11e-4 3.51e-4 3.87e-4 4.38e-4 6.31e-4
## 2 1.19e-4 4.91e-3 4.70e-3 4.51e-3 1.06e-3 1.03e-2 7.40e-3 8.91e-3 6.25e-3
## 3 2.49e-4 1.06e-1 8.81e-2 7.64e-2 9.05e-3 3.04e-1 1.50e-1 1.72e-1 6.72e-2
## 4 6.77e-4 2.22e+0 1.64e+0 1.28e+0 8.55e-2 8.48e+0 2.92e+0 3.24e+0 7.44e-1
## # ... with 14 more variables: `10` <dbl>, `11` <dbl>, `12` <dbl>, `13` <dbl>,
## #   `14` <dbl>, `15` <dbl>, `16` <dbl>, `17` <dbl>, `18` <dbl>, `19` <dbl>,
## #   `20` <dbl>, `21` <dbl>, `22` <dbl>, `23` <dbl>
```

```
write.csv(dataframe_all,"data/predictions.csv", row.names = FALSE)
```