sara-experimental

Sara Shao

10/7/2021

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.0.6     v purrr   0.3.4
## v tidyr   1.1.2     v dplyr   1.0.4
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x readr::guess_encoding()  masks rvest::guess_encoding()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x purrr::pluck()           masks rvest::pluck()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

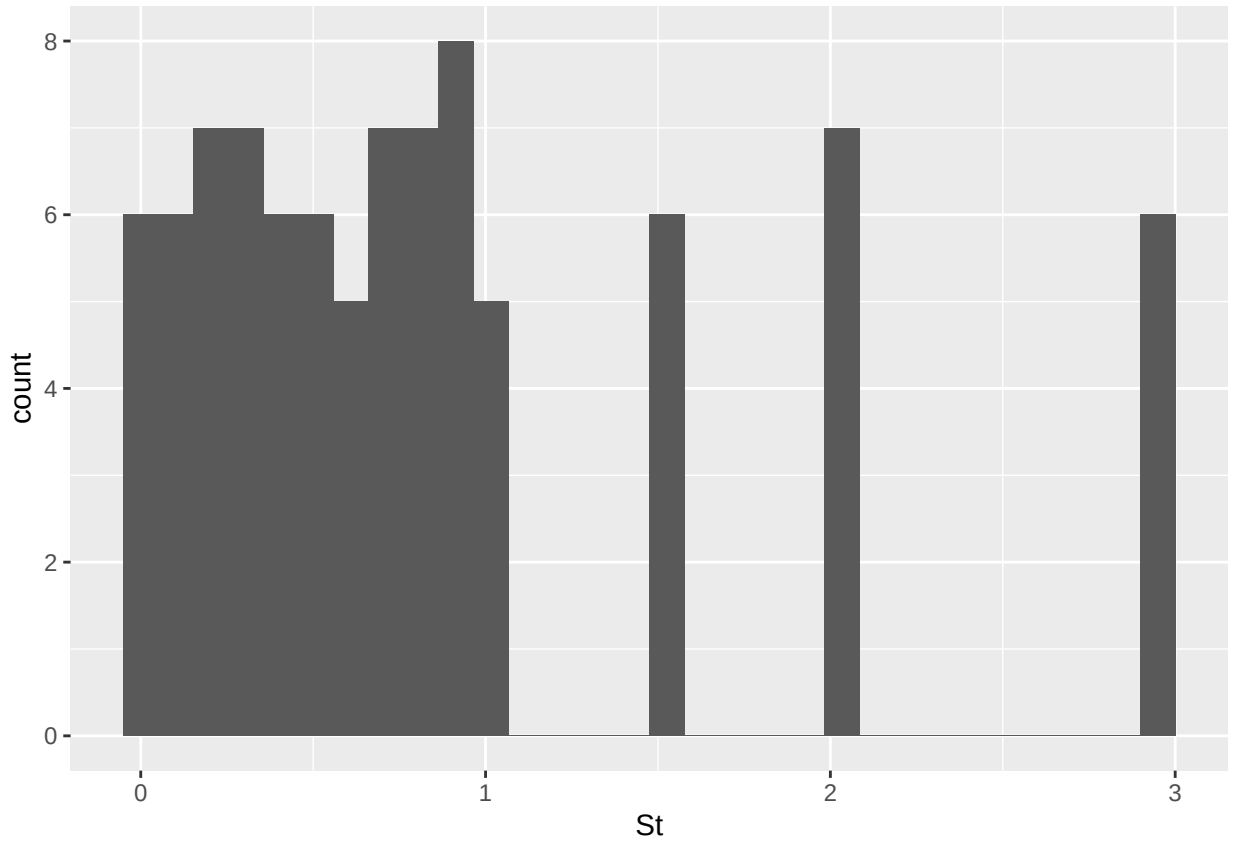
```
train <- read.csv('data-train.csv')
```

```
head(train)
```

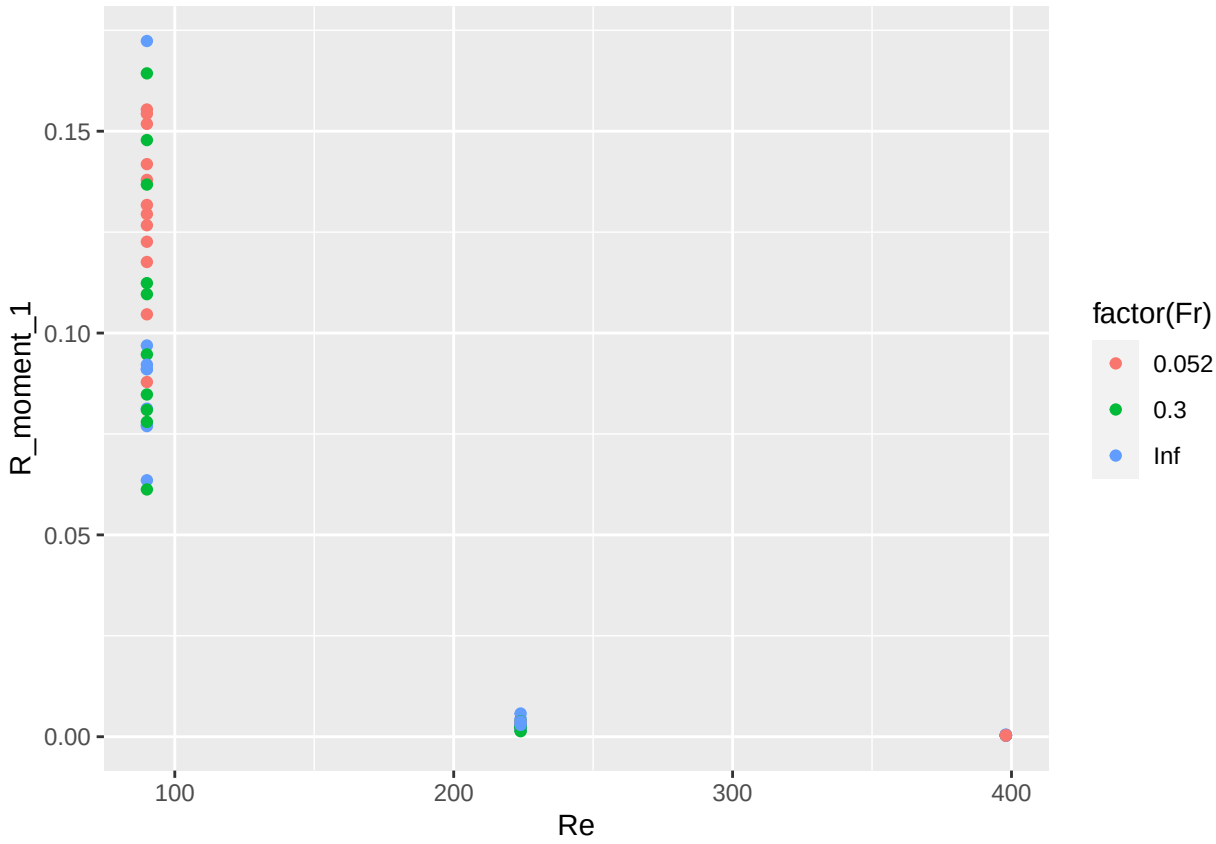
```
##     St  Re    Fr R_moment_1 R_moment_2 R_moment_3 R_moment_4
## 1 0.10 224 0.052 0.00215700  0.1303500   14.37400  1586.5000
## 2 3.00 224 0.052 0.00379030  0.4704200   69.94000 10404.0000
## 3 0.70 224   Inf 0.00290540  0.0434990    0.82200    15.5510
## 4 0.05  90   Inf 0.06352800  0.0906530    0.46746     3.2696
## 5 0.70 398   Inf 0.00036945  0.0062242    0.12649     2.5714
## 6 2.00  90 0.300 0.14780000  2.0068000   36.24900   671.6700
```

```
ggplot(data = train, mapping = aes(x = St)) +
  geom_histogram()
```

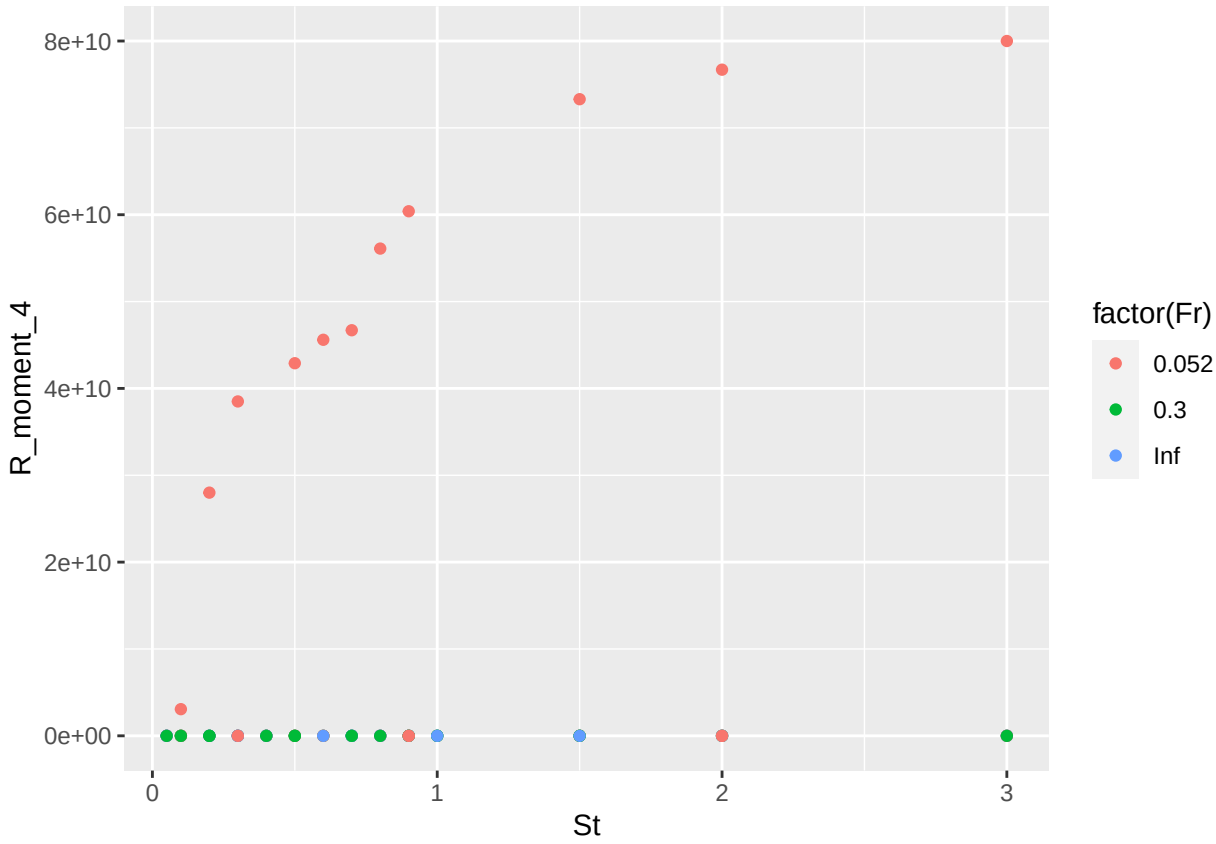
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(data = train, mapping = aes(x = Re, y = R_moment_1, color = factor(Fr))) +
  geom_point()
```

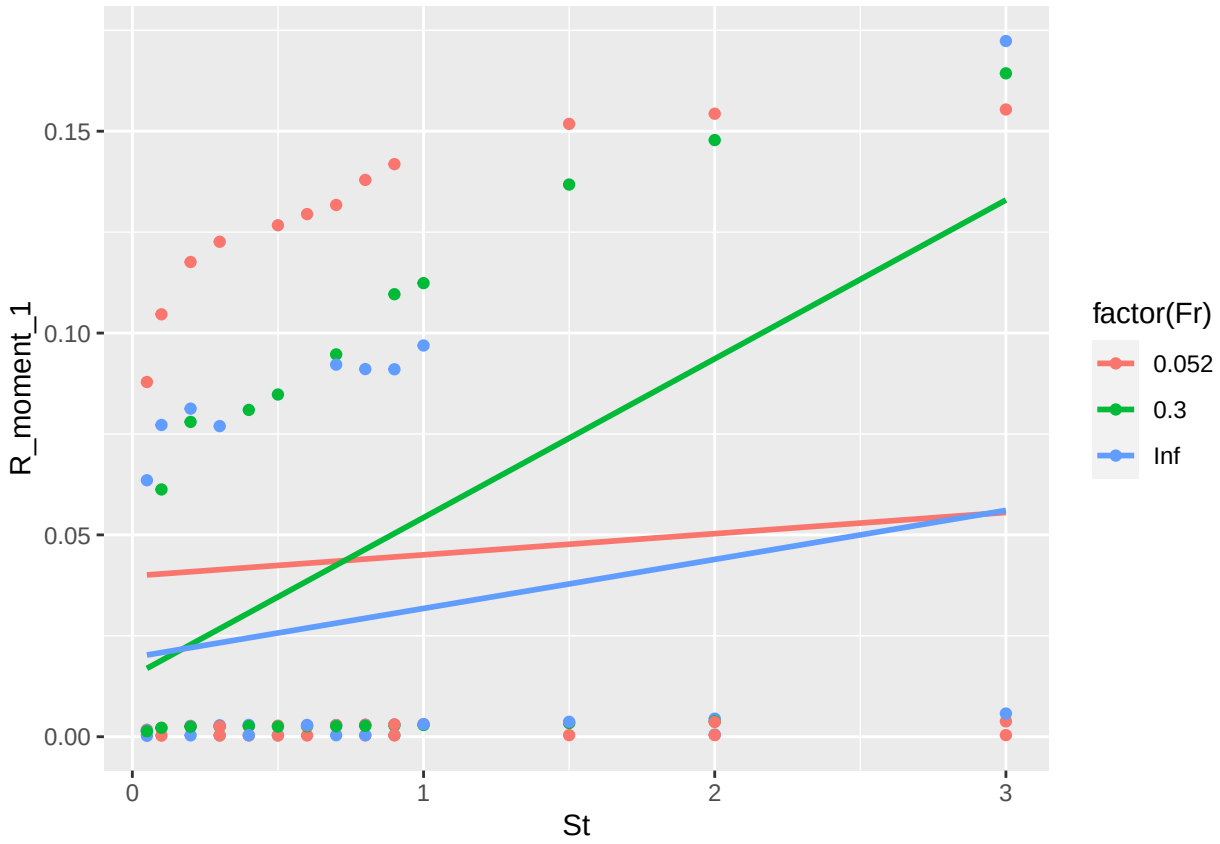



```
ggplot(data = train, mapping = aes(x = St, y = R_moment_4, color = factor(Fr))) +
  geom_point()
```



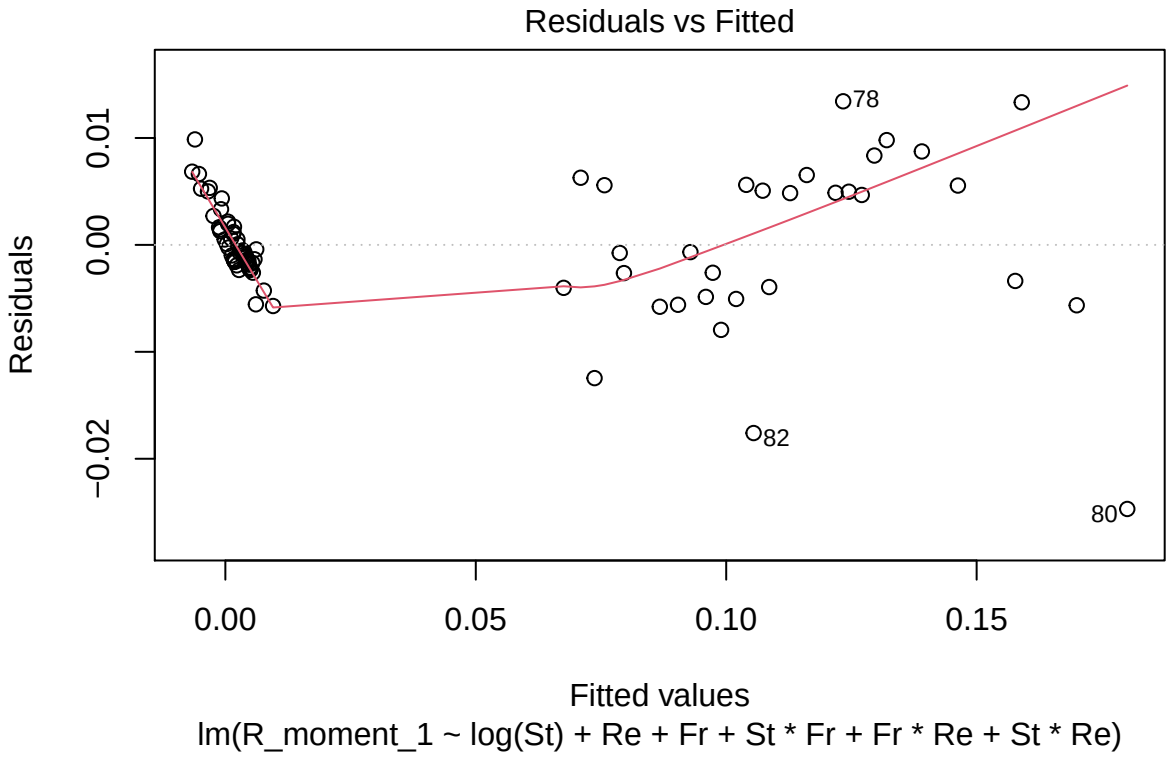

```
ggplot(data = train, mapping = aes(x = St, y = R_moment_1, color = factor(Fr))) +
  geom_point() +
  geom_smooth(method = lm, se = F)
```

```
## `geom_smooth()` using formula 'y ~ x'
```

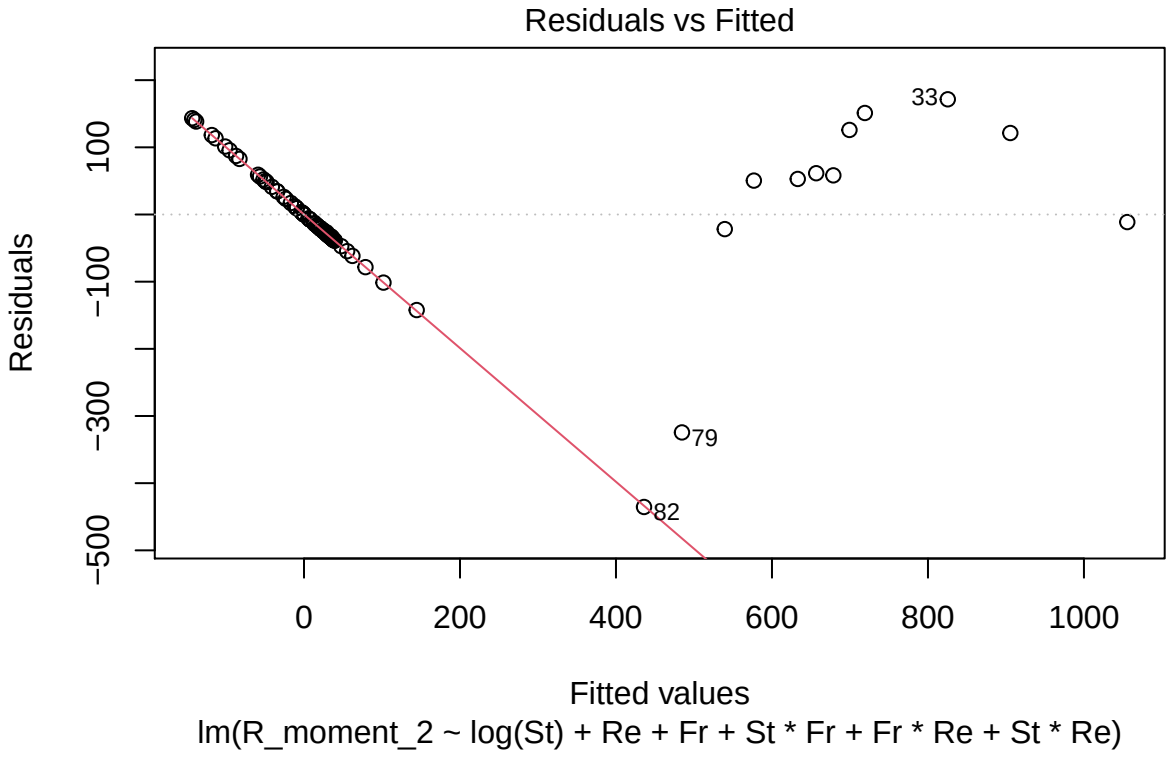



```
train_data <- train %>%
  mutate(Fr = as.ordered(Fr)) %>%
  mutate(Re = as.ordered(Re))
```

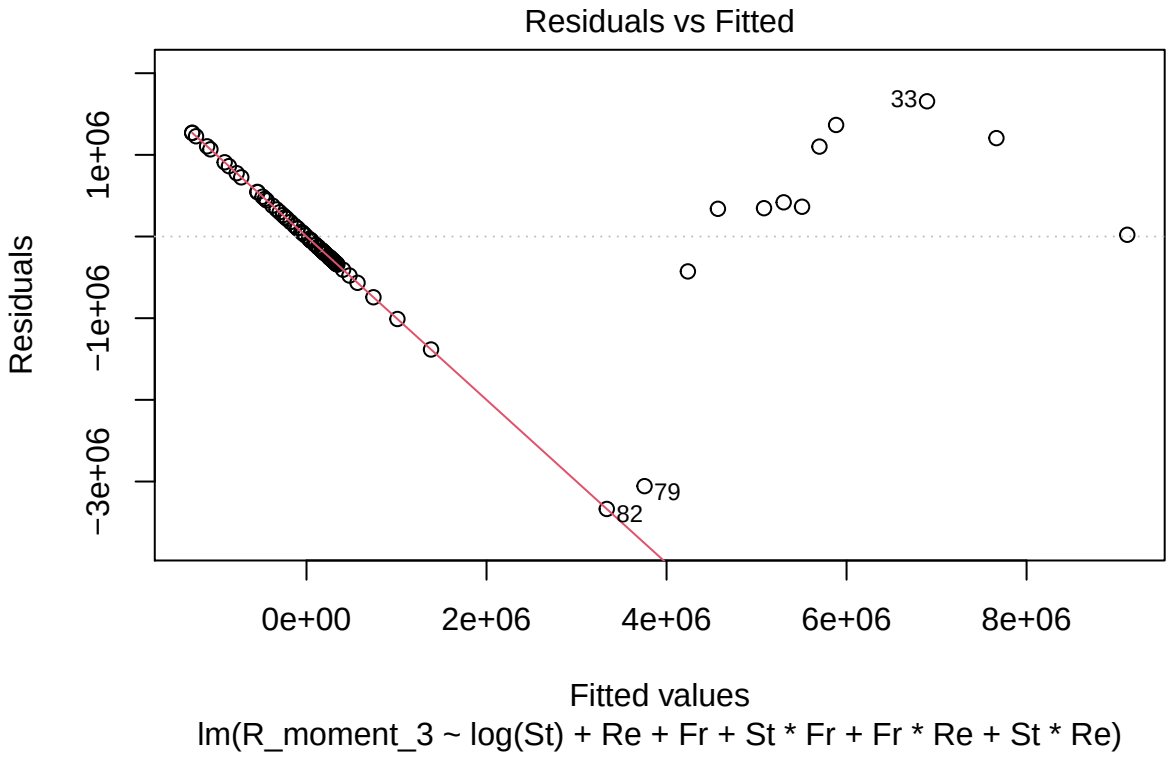
```
lm_R1 <- lm(R_moment_1 ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
lm_R2 <- lm(R_moment_2 ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
lm_R3 <- lm(R_moment_3 ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
lm_R4 <- lm(R_moment_4 ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
plot(lm_R1, 1)
```



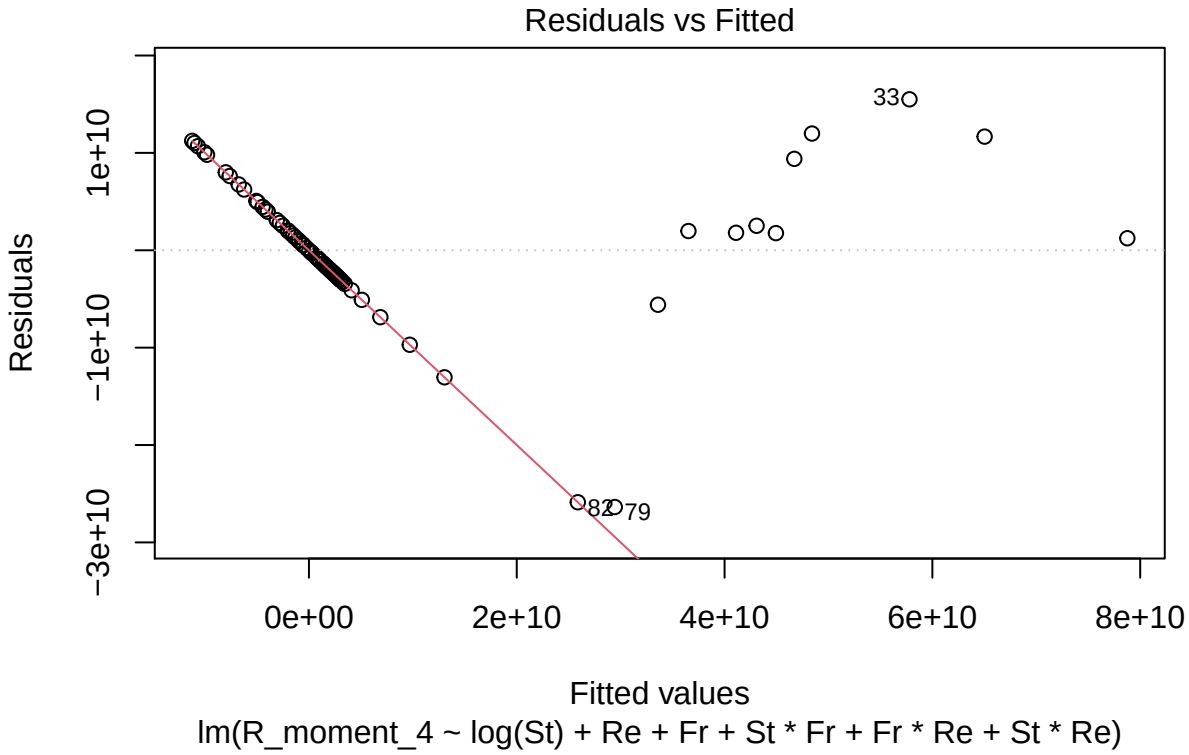

```
plot(lm_R2, 1)
```




```
plot(lm_R3, 1)
```




```
plot(lm_R4, 1)
```




```
lm1 <- lm(log(R_moment_1) ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
summary(lm1)
```

```
##
```

```
## Call:
## lm(formula = log(R_moment_1) ~ log(St) + Re + Fr + St * Fr +
##     Fr * Re + St * Re, data = train_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.211809 -0.042926 -0.006391  0.038831  0.171243
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -5.349488   0.027649 -193.480  < 2e-16 ***
## log(St)      0.145668   0.014562   10.003 1.89e-15 ***
## Re.L        -4.028476   0.027949 -144.139  < 2e-16 ***
## Re.Q         0.644476   0.022457   28.698  < 2e-16 ***
## Fr.L        -0.102135   0.019793   -5.160 1.95e-06 ***
## Fr.Q         0.109493   0.026874    4.074 0.000113 ***
## St           0.095896   0.021136    4.537 2.13e-05 ***
## Fr.L:St      0.095376   0.017271    5.522 4.60e-07 ***
## Fr.Q:St     -0.064244   0.022042   -2.915 0.004692 **
## Re.L:Fr.L    0.242947   0.024770    9.808 4.39e-15 ***
## Re.Q:Fr.L   -0.076314   0.022588   -3.379 0.001159 **
## Re.L:Fr.Q   -0.077781   0.046306   -1.680 0.097169 .
## Re.Q:Fr.Q          NA         NA       NA       NA
## Re.L:St     -0.008897   0.021672   -0.411 0.682581
## Re.Q:St     -0.025325   0.018376   -1.378 0.172252
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07645 on 75 degrees of freedom
## Multiple R-squared:  0.999,  Adjusted R-squared:  0.9988
## F-statistic:  5797 on 13 and 75 DF,  p-value: < 2.2e-16
```

```
lm2 <- lm(log(R_moment_2) ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
summary(lm2)
```

```
##
## Call:
## lm(formula = log(R_moment_2) ~ log(St) + Re + Fr + St * Fr +
##     Fr * Re + St * Re, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6442 -0.2697 -0.0561  0.3429  1.8016
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.300240   0.307283  -0.977  0.33167
## log(St)      1.500864   0.161841   9.274 4.50e-14 ***
## Re.L        -4.269814   0.310614 -13.746  < 2e-16 ***
## Re.Q         1.098368   0.249584   4.401 3.52e-05 ***
## Fr.L        -1.927717   0.219976  -8.763 4.20e-13 ***
## Fr.Q         0.946857   0.298671   3.170  0.00221 **
## St          -0.997463   0.234905  -4.246 6.16e-05 ***
## Fr.L:St     -0.007867   0.191944  -0.041  0.96742
## Fr.Q:St     -0.043393   0.244968  -0.177  0.85988
```

```
## Re.L:Fr.L    3.399373   0.275290  12.348  < 2e-16 ***
## Re.Q:Fr.L   -0.646249   0.251039  -2.574  0.01202 *
## Re.L:Fr.Q   -2.588037   0.514630  -5.029 3.27e-06 ***
## Re.Q:Fr.Q          NA         NA      NA       NA
## Re.L:St     -0.470624   0.240856  -1.954  0.05443 .
## Re.Q:St     -0.128587   0.204226  -0.630  0.53085
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8497 on 75 degrees of freedom
## Multiple R-squared:  0.9553, Adjusted R-squared:  0.9476
## F-statistic: 123.4 on 13 and 75 DF,  p-value: < 2.2e-16
```

```
lm3 <- lm(log(R_moment_3) ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
summary(lm3)
```

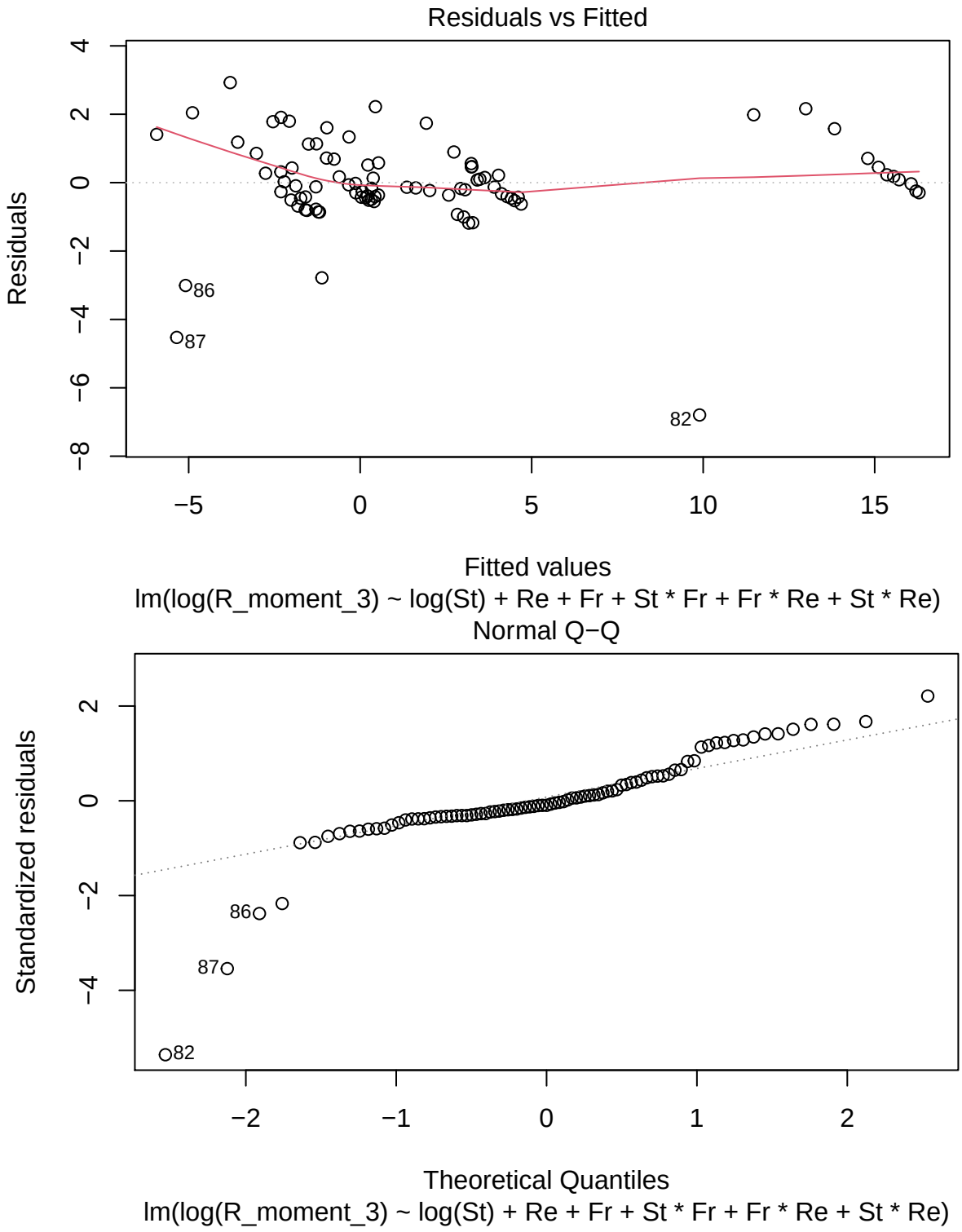
```
##
## Call:
## lm(formula = log(R_moment_3) ~ log(St) + Re + Fr + St * Fr +
##     Fr * Re + St * Re, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7949 -0.4431 -0.1224  0.5575  2.9257
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.54701    0.51756   8.785 3.81e-13 ***
## log(St)      2.36189    0.27259   8.665 6.47e-13 ***
## Re.L        -4.89792    0.52317  -9.362 3.06e-14 ***
## Re.Q         1.52364    0.42038   3.624 0.000525 ***
## Fr.L        -3.75114    0.37051 -10.124 1.12e-15 ***
## Fr.Q         1.83513    0.50306   3.648 0.000486 ***
## St          -1.72871    0.39565  -4.369 3.95e-05 ***
## Fr.L:St     -0.09194    0.32329  -0.284 0.776889
## Fr.Q:St     -0.03580    0.41260  -0.087 0.931098
## Re.L:Fr.L    6.45803    0.46368  13.928  < 2e-16 ***
## Re.Q:Fr.L   -1.15825    0.42283  -2.739 0.007689 **
## Re.L:Fr.Q   -4.96308    0.86680  -5.726 2.01e-07 ***
## Re.Q:Fr.Q         NA         NA      NA       NA
## Re.L:St     -0.79750    0.40568  -1.966 0.053018 .
## Re.Q:St     -0.17827    0.34398  -0.518 0.605816
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.431 on 75 degrees of freedom
## Multiple R-squared:  0.9459, Adjusted R-squared:  0.9365
## F-statistic: 100.8 on 13 and 75 DF,  p-value: < 2.2e-16
```

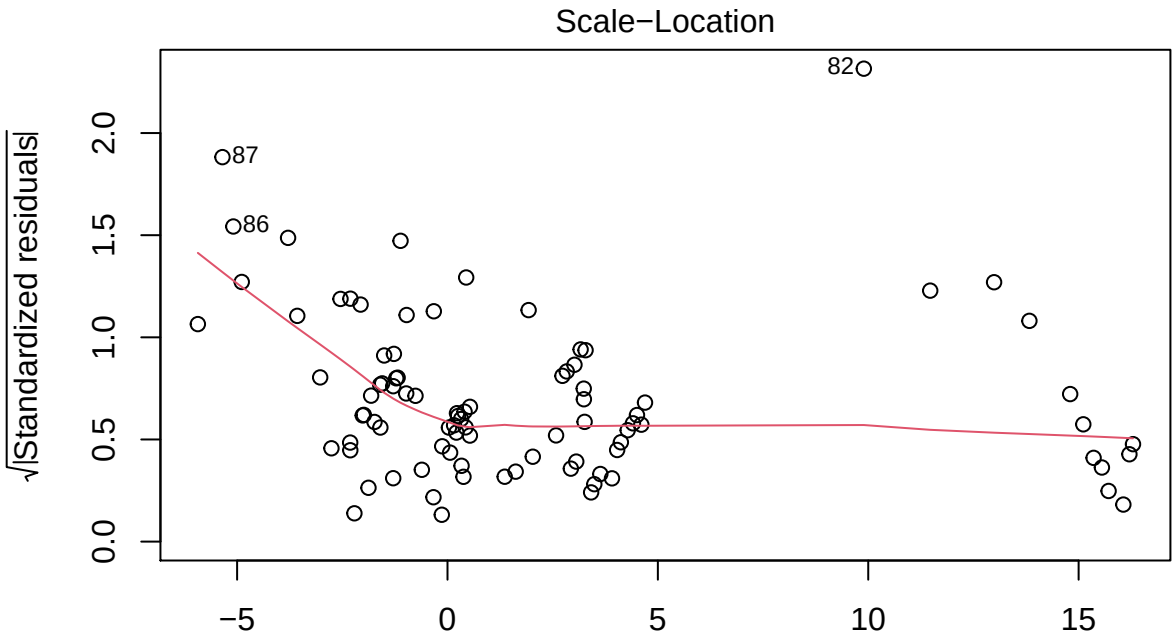
```
lm4 <- lm(log(R_moment_4) ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
summary(lm4)
```

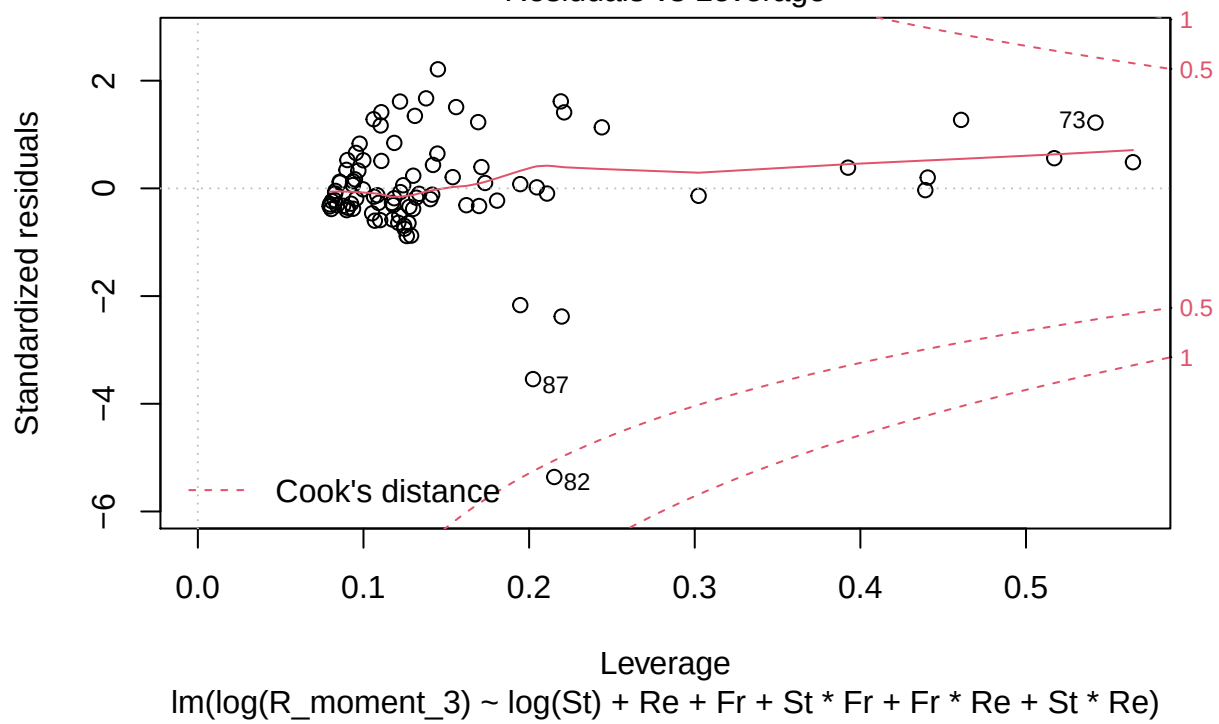
```
##
## Call:
## lm(formula = log(R_moment_4) ~ log(St) + Re + Fr + St * Fr +
```

```
##     Fr * Re + St * Re, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6675 -0.6183 -0.1392  0.7410  3.8875
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.28287    0.70476  13.172  < 2e-16 ***
## log(St)      3.10948    0.37119   8.377 2.28e-12 ***
## Re.L        -5.63270    0.71240  -7.907 1.80e-11 ***
## Re.Q         1.94835    0.57242   3.404 0.001070 **
## Fr.L        -5.55550    0.50452 -11.012  < 2e-16 ***
## Fr.Q         2.71540    0.68500   3.964 0.000167 ***
## St          -2.36662    0.53876  -4.393 3.62e-05 ***
## Fr.L:St     -0.17635    0.44023  -0.401 0.689864
## Fr.Q:St     -0.01987    0.56184  -0.035 0.971880
## Re.L:Fr.L    9.47956    0.63138  15.014  < 2e-16 ***
## Re.Q:Fr.L   -1.66019    0.57576  -2.883 0.005130 **
## Re.L:Fr.Q   -7.28081    1.18031  -6.169 3.21e-08 ***
## Re.Q:Fr.Q         NA         NA      NA       NA
## Re.L:St     -1.08269    0.55241  -1.960 0.053716 .
## Re.Q:St     -0.22245    0.46840  -0.475 0.636226
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.949 on 75 degrees of freedom
## Multiple R-squared:  0.9457, Adjusted R-squared:  0.9363
## F-statistic: 100.6 on 13 and 75 DF,  p-value: < 2.2e-16
```

```
plot(lm3)
```


Residuals vs Fitted
Residuals
Fitted values
lm(log(R_moment_3) ~ log(St) + Re + Fr + St * Fr + Fr * Re + St * Re)
Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(log(R_moment_3) ~ log(St) + Re + Fr + St * Fr + Fr * Re + St * Re)
86
87
82






```
set.seed(21)
shuffled_train <- train_data[sample(nrow(train_data)),]
folds <- cut(seq(1,nrow(train_data)),breaks=10,labels=FALSE)

# error
rmse.cv.lm <- rep(0, 10)
```

```r
# Cross validation
for(i in 1:10){
    #Segment your data by fold using the which() function
    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData <- shuffled_train[testIndexes, ]
    y.test <- testData$R_moment_1
    trainData <- shuffled_train[-testIndexes, ]

    #Use the test and train data
    lm_cv <- lm(log(R_moment_1) ~ log(St) + Re + Fr + St*Fr + Re*Fr + St*Re, data = trainData)
    pred_lm <- exp(predict(lm_cv, testData, type='response'))
    rmse.cv.lm[i] = mean((pred_lm - y.test)^2)
}

mean(rmse.cv.lm)
```

```
## [1] 2.864351e-05
```

```r
# error
rmse.cv.lm <- rep(0, 10)

# Cross validation
for(i in 1:10){
    #Segment your data by fold using the which() function
    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData <- shuffled_train[testIndexes, ]
    y.test <- testData$R_moment_2
    trainData <- shuffled_train[-testIndexes, ]

    #Use the test and train data
    lm_cv <- lm(log(R_moment_2) ~ log(St) + Re + Fr + St*Fr + Re*Fr + St*Re, data = trainData)
    pred_lm <- exp(predict(lm_cv, testData, type='response'))
    rmse.cv.lm[i] = mean((pred_lm - y.test)^2)
}

mean(rmse.cv.lm)
```

```
## [1] 4621.783
```

```r
# error
rmse.cv.lm <- rep(0, 10)

# Cross validation
for(i in 1:10){
    #Segment your data by fold using the which() function
    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData <- shuffled_train[testIndexes, ]
    y.test <- testData$R_moment_3
    trainData <- shuffled_train[-testIndexes, ]

    #Use the test and train data
    lm_cv <- lm(log(R_moment_3) ~ log(St) + Re + Fr + St*Fr + Re*Fr + St*Re, data = trainData)
    pred_lm <- exp(predict(lm_cv, testData, type='response'))
    rmse.cv.lm[i] = mean((pred_lm - y.test)^2)
}
```

```r
mean(rmse.cv.lm)
```

```
## [1] 578718160228
```

```r
# error
rmse.cv.lm <- rep(0, 10)

# Cross validation
for(i in 1:10){
    #Segment your data by fold using the which() function
    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData <- shuffled_train[testIndexes, ]
    y.test <- testData$R_moment_4
    trainData <- shuffled_train[-testIndexes, ]

    #Use the test and train data
    lm_cv <- lm(log(R_moment_4) ~ log(St) + Re + Fr + St*Fr + Re*Fr + St*Re, data = trainData)
    pred_lm <- exp(predict(lm_cv, testData, type='response'))
    rmse.cv.lm[i] = mean((pred_lm - y.test)^2)
}

mean(rmse.cv.lm)
```

```
## [1] 5.488601e+19
```