sara-experimental

Sara Shao

10/7/2021

```r
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()
```

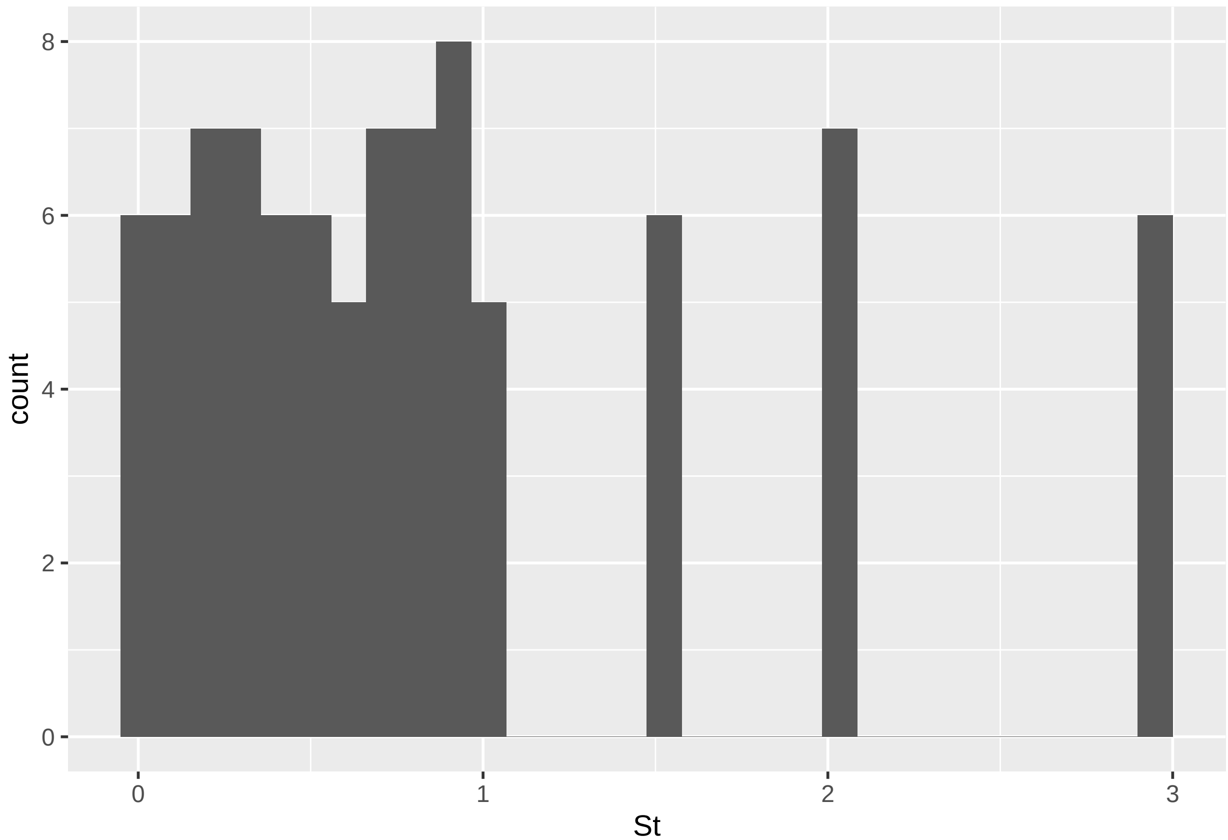
```r
train <- read.csv('data-train.csv')
```

```r
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
```

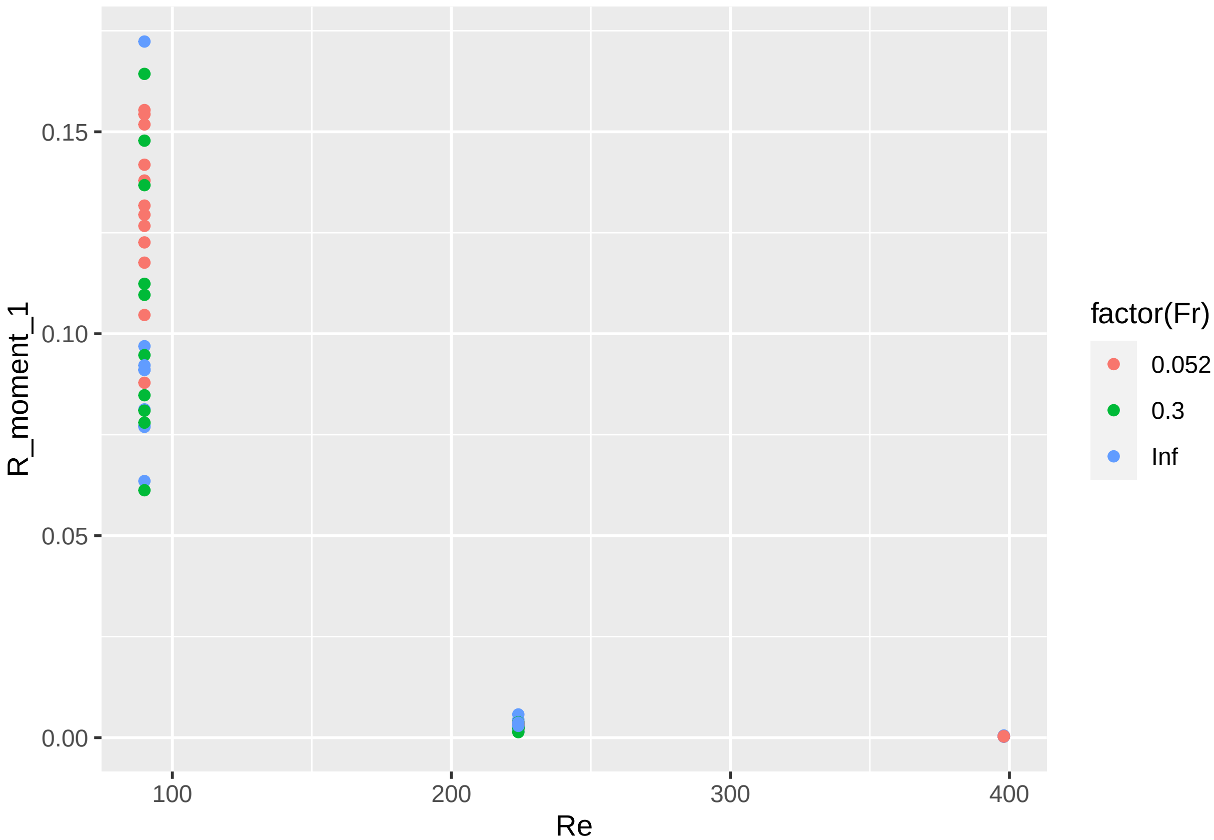
```r
ggplot(data = train, mapping = aes(x = St)) +
  geom_histogram()
```

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

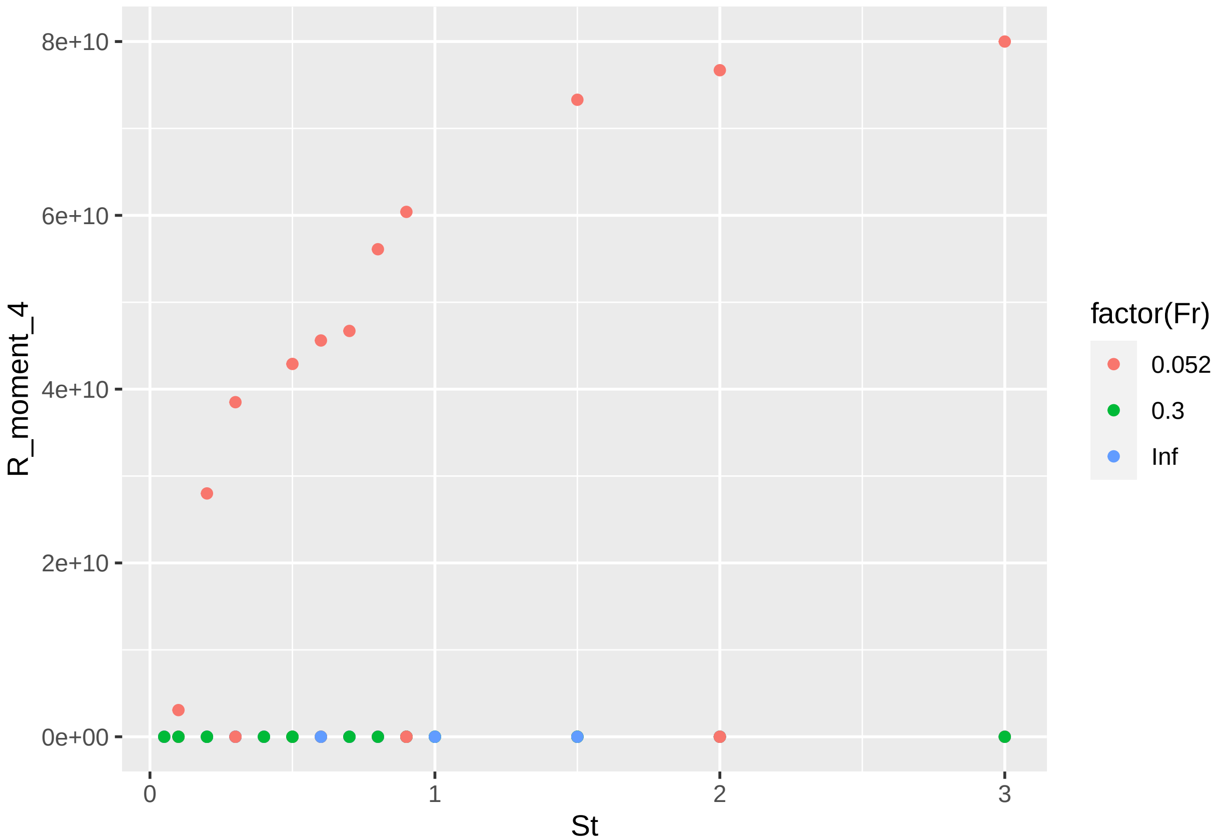


We will try using a log transform on the St variable since the distribution for the St variable is not normally distributed.

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

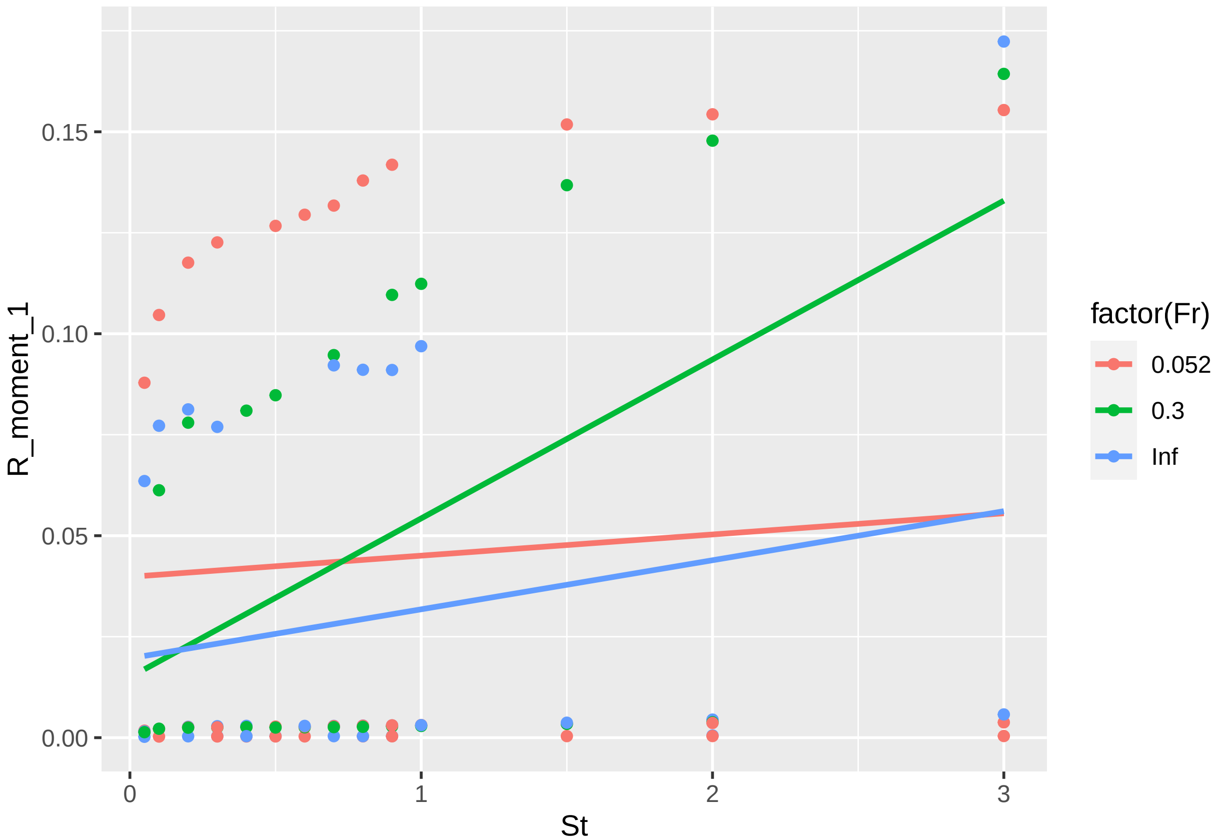



```r
ggplot(data = train, mapping = aes(x = St, y = R_moment_4, color = factor(Fr))) +
  geom_point()
```




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

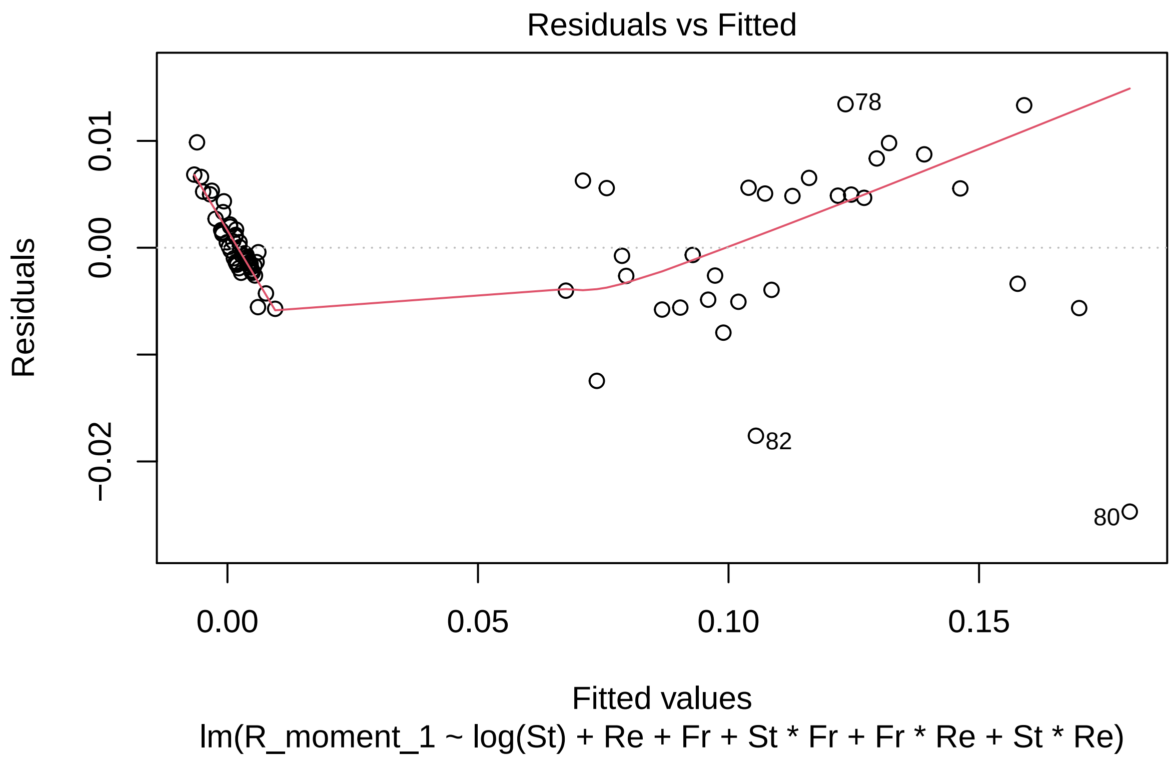
```
## `geom_smooth()` using formula 'y ~ x'
```



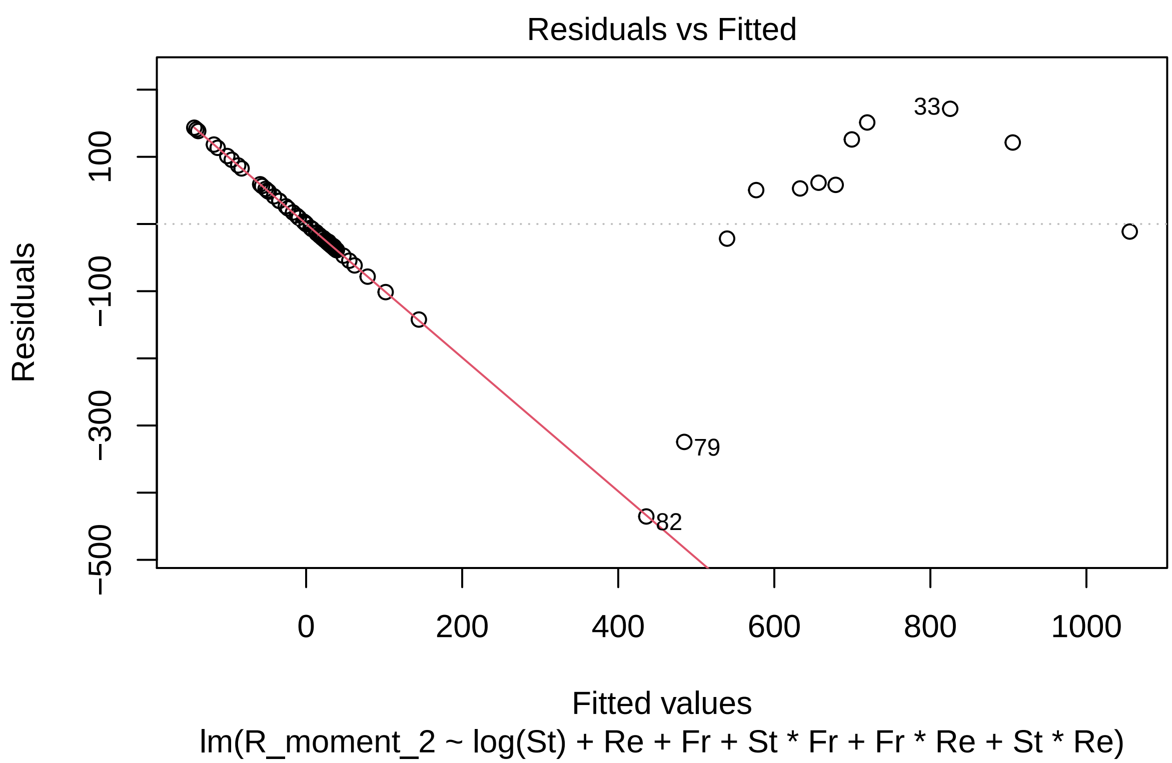
The graphs above show some evidence of interactions, so we will explore interaction terms in our model.

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

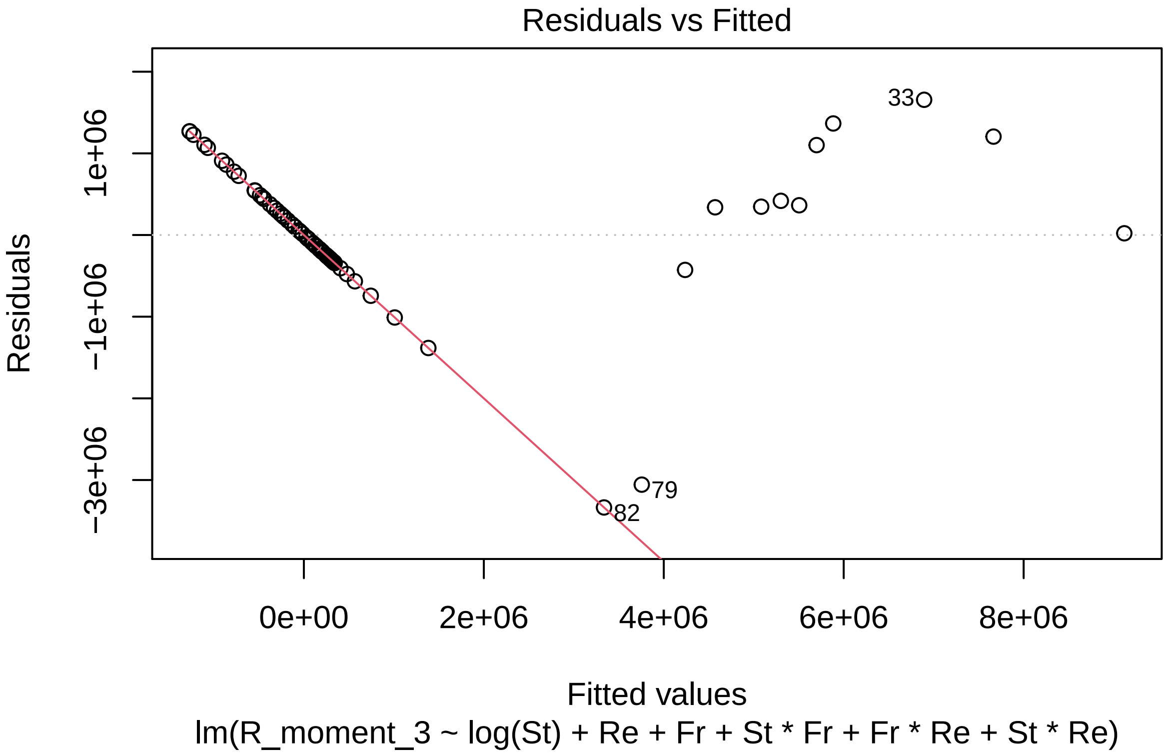
```
lm_R1 <- lm(R_moment_1 ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
lm_R2 <- lm(R_moment_2 ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
lm_R3 <- lm(R_moment_3 ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
lm_R4 <- lm(R_moment_4 ~ log(St) + Re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



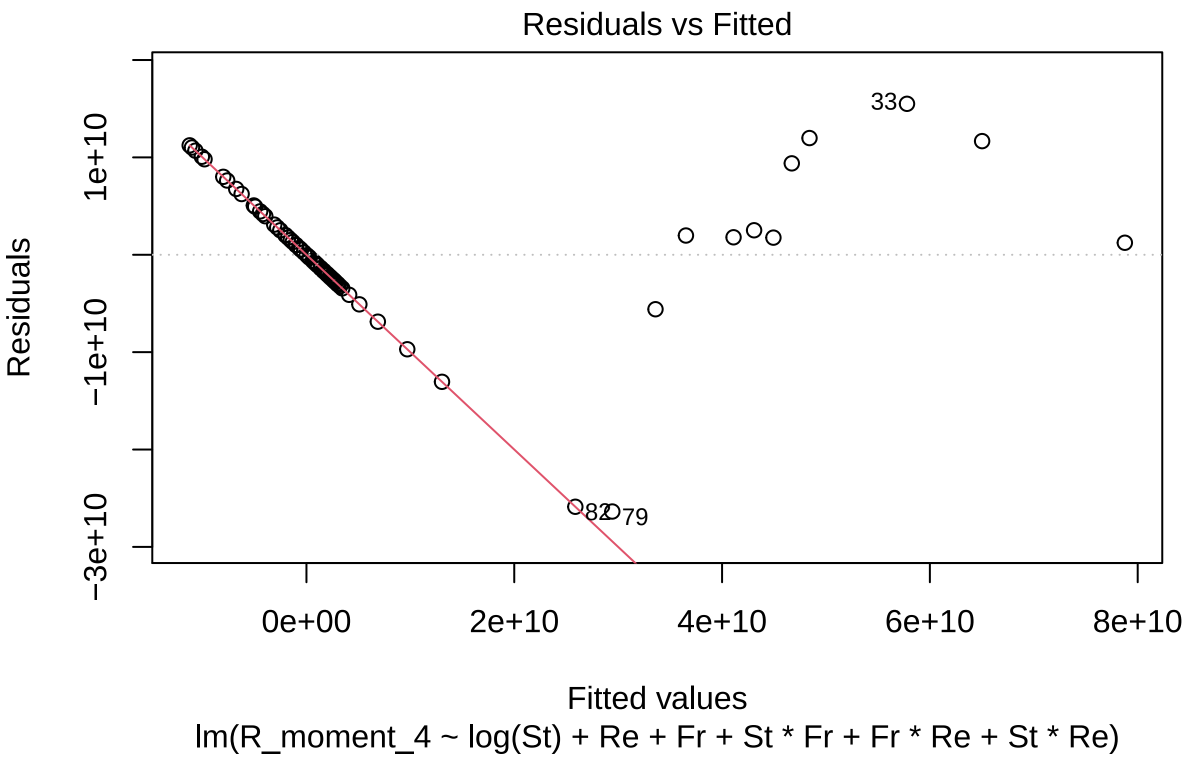

```r
plot(lm_R2, 1)
```




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




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



Because the linearity condition is not fulfilled in the above Residuals vs. Fitted plots, we will consider performing a log transformation on our response variables (R moments 1-4).

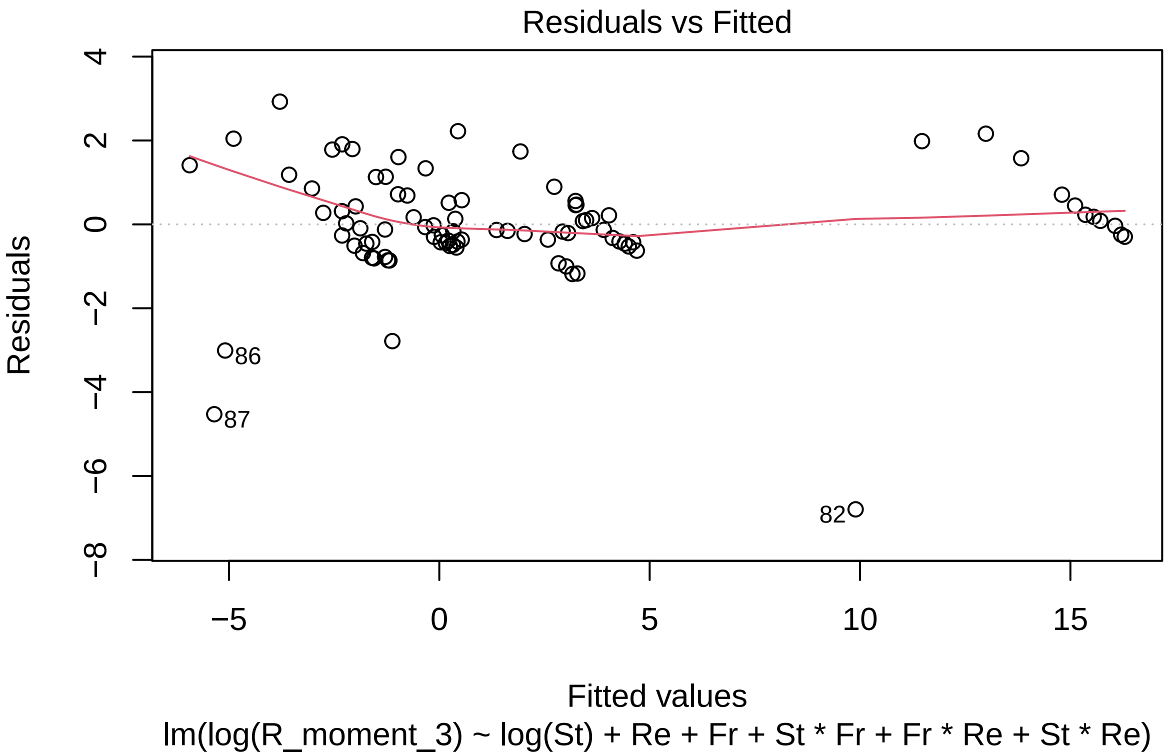
```r
lm1 <- lm(log(R_moment_1) ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
summary(lm1)
```

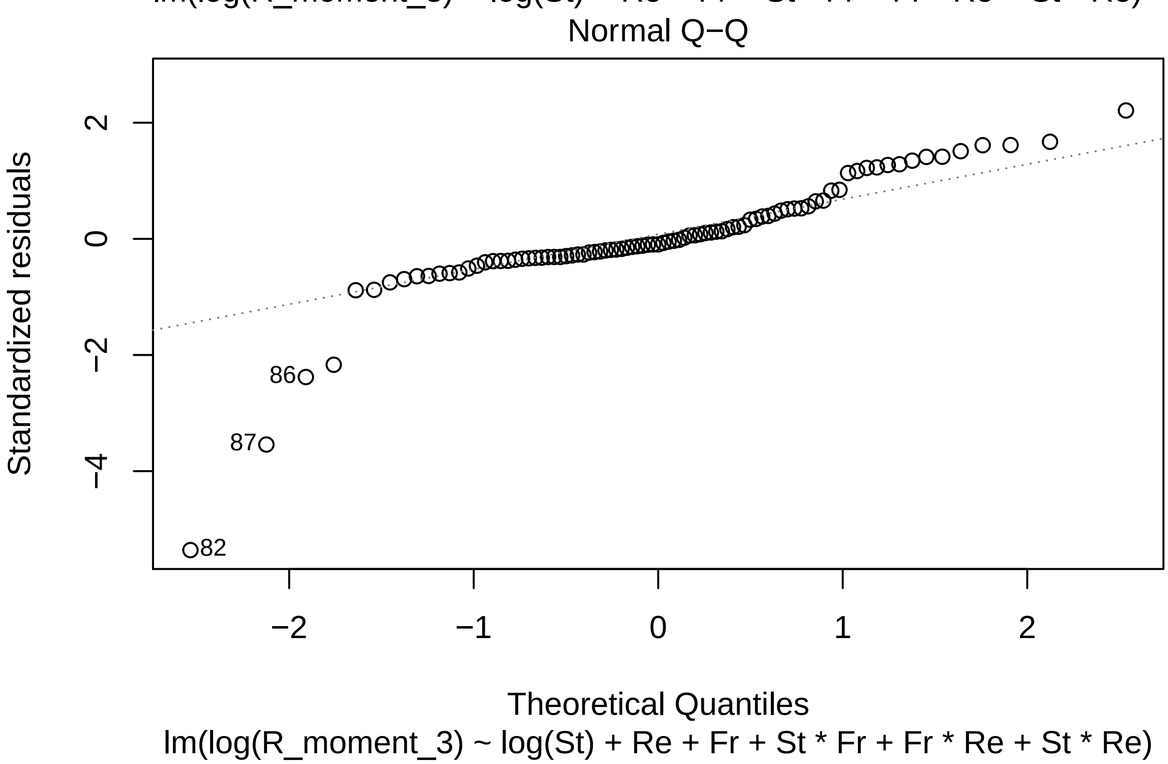
```
##
## Call:
## lm(formula = log(R_moment_1) ~ log(St) + Re + Fr + St * Fr +
##     Fr * Re + St * Re, data = train_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.211809 -0.042926 -0.006391  0.038831  0.171243
##
## Coefficients: (1 not defined because of singularities)
##              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
```

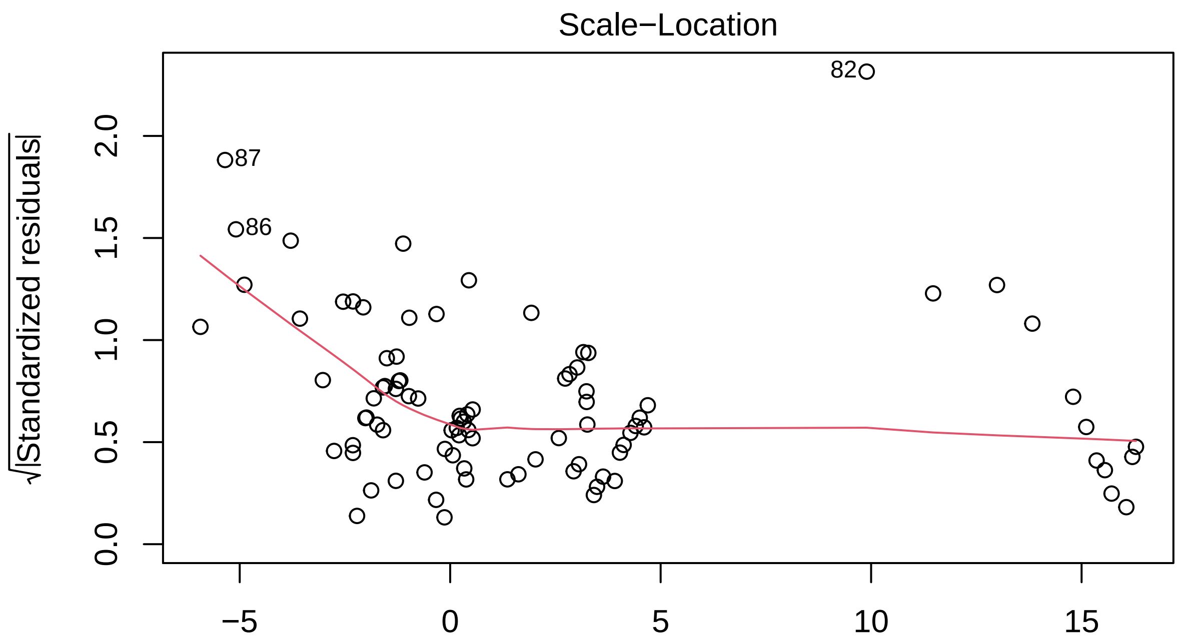
```r
lm2 <- lm(log(R_moment_2) ~ log(St) + Re + Fr + St*Fr + Fr*Re + St*Re, data = train_data)
#summary(lm2)

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

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

A one percent increase in Stokes number is associated with 0.146% increase in R moment 1, holding all other predictors constant. When the Reynolds number is 224, the R moment 1 is expected to decrease by 403% from when the Reynolds number is 90, holding all other predictors constant. When the Reynolds number is 398, the R moment 1 is expected to increase by 64% compared to when the Reynolds number is 90. When the Reynolds number is 224 and the Froud number is 0.3, the R moment 1 is expected to be an additional 24% lower compared to when both of those conditions are not met.

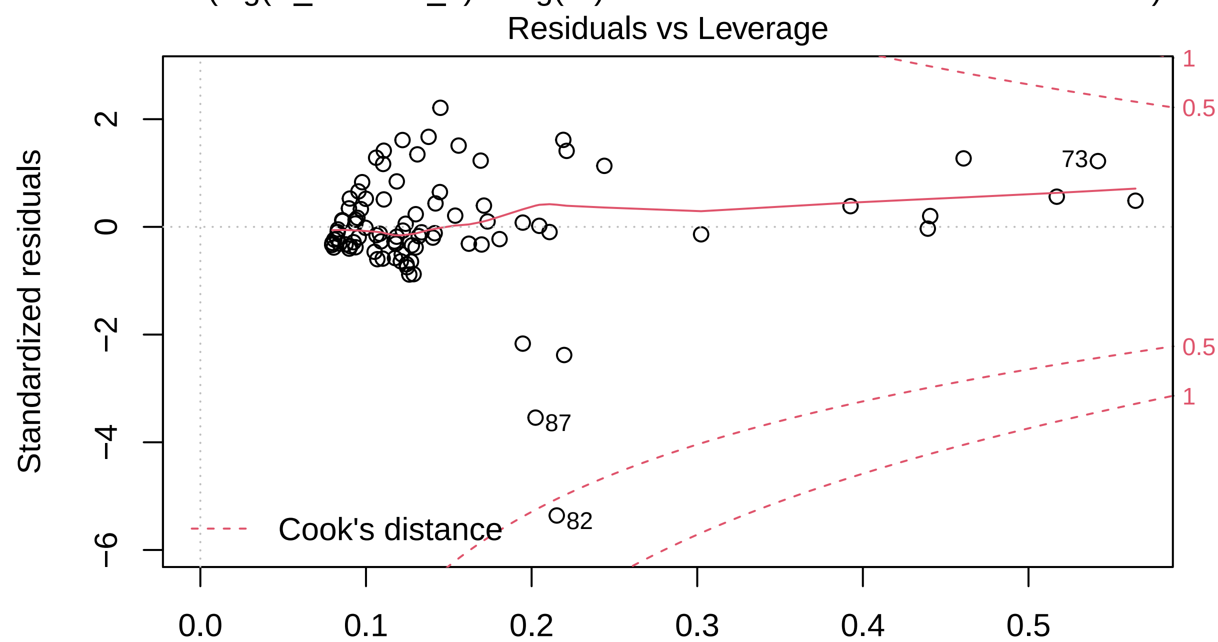
```r
plot(lm3)
```


Residuals vs Fitted
Residuals
4
2
0
−2
−4
−6
−8
86
87
82
−5
0
5
10
15
Fitted values
lm(log(R_moment_3) ~ log(St) + Re + Fr + St * Fr + Fr * Re + St * Re)


Normal Q−Q
Standardized residuals
2
0
−2
−4
86
87
82
−2
−1
0
1
2
Theoretical Quantiles
lm(log(R_moment_3) ~ log(St) + Re + Fr + St * Fr + Fr * Re + St * Re)



lm(log(R_moment_3) ~ log(St) + Re + Fr + St * Fr + Fr * Re + St * Re)



lm(log(R_moment_3) ~ log(St) + Re + Fr + St * Fr + Fr * Re + St * Re)

```
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
```