author: "Ava Exelbirt" This is my work for the prelininary steps (Step 1-8) on the document.

1. Data loading and initial setup

```
#install.packages("pdp")
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ----------------------- tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ----------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(broom)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(pdp)
```

```
##
## Attaching package: 'pdp'
##
## The following object is masked from 'package:purrr':
##
##     partial
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

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

head(data_test)
```

```
##     St  Re    Fr
## 1 0.05 398 0.052
## 2 0.20 398 0.052
## 3 0.70 398 0.052
## 4 1.00 398 0.052
## 5 0.10 398   Inf
## 6 0.60 398   Inf
```

```
head(data_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
```

2. Data Preparation and Transformation I cleaned and transformed the raw moments to central moments to summary statistics

```
# get the central moments from raw moments
raw2central <- function(m1, m2, m3, m4) {
  mu <- m1
  cm2 <- m2 - mu^2
  cm3 <- m3 - 3*m2*mu + 2*mu^3
  cm4 <- m4 - 4*m3*mu + 6*m2*mu^2 - 3*mu^4
  tibble(mu = mu, cm2 = cm2, cm3 = cm3, cm4 = cm4)
}


# apply to the abovec func to training
df_train <- data_train |>
  mutate(
    R_moment_1 = as.numeric(R_moment_1),
    R_moment_2 = as.numeric(R_moment_2),
    R_moment_3 = as.numeric(R_moment_3),
    R_moment_4 = as.numeric(R_moment_4),
    Fr = suppressWarnings(as.numeric(Fr)),
    Re = as.numeric(Re),
    St = as.numeric(St)
  ) |>
  rowwise() |>
  mutate( #now get the central moments and summary stats
    .temp = list(raw2central(R_moment_1, R_moment_2, R_moment_3, R_moment_4)),
    mean = .temp$mu,
    var = .temp$cm2,
    sd = sqrt(if_else(.temp$cm2 > 0, .temp$cm2, NA_real_)),
    skew = if_else(!is.na(.temp$cm2) & .temp$cm2 > 0, .temp$cm3 / (.temp$cm2^(3/2)), NA_real_),
    kurt = if_else(!is.na(.temp$cm2) & .temp$cm2 > 0, .temp$cm4 / (.temp$cm2^2), NA_real_)
  ) |>
  ungroup() |>
  select(-.temp)
```

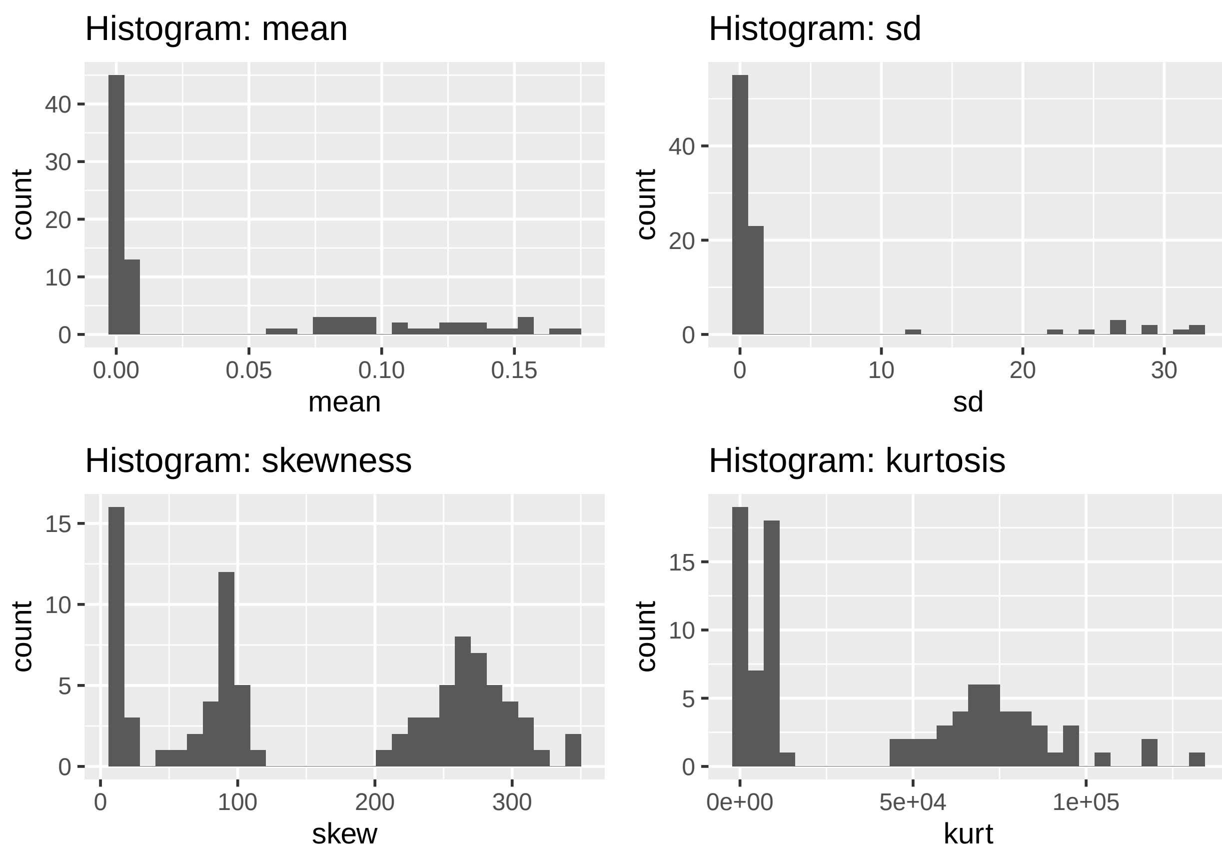
```r
# Basic checks for NA/Inf in mean/var (chat added this when debugged)
stopifnot(nrow(df_train) > 0)

# for the test set, check types & detect Inf in Fr
df_test <- data_test |>
  mutate(
    Fr = suppressWarnings(as.numeric(Fr)),
    Re = as.numeric(Re),
    St = as.numeric(St)
  )
```

3. EDA

```r
summary_stats <- summary(df_train)
print(summary_stats)
```

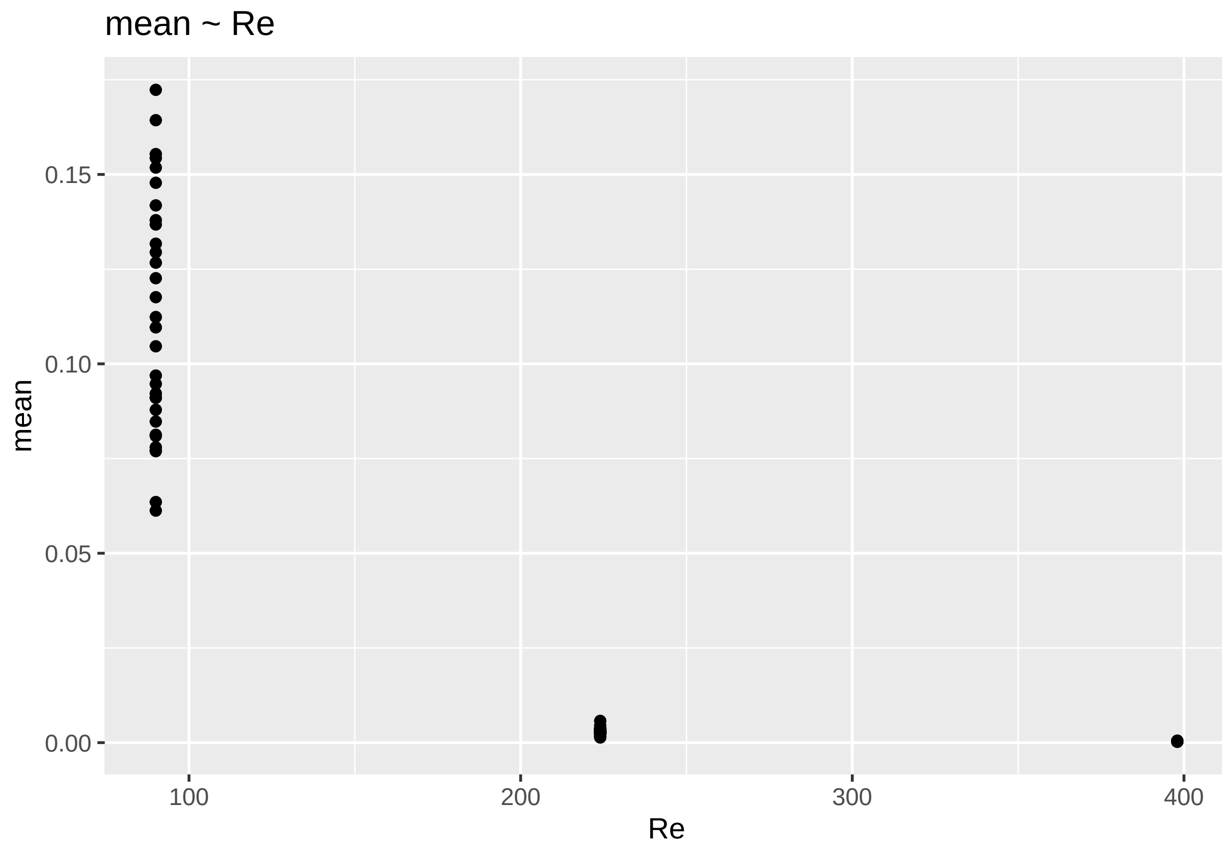
```
##        St               Re              Fr           R_moment_1
##  Min.   :0.0500   Min.   : 90.0   Min.   :0.052   Min.   :0.000222
##  1st Qu.:0.3000   1st Qu.: 90.0   1st Qu.:0.052   1st Qu.:0.002157
##  Median :0.7000   Median :224.0   Median :0.300   Median :0.002958
##  Mean   :0.8596   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             mean
##  Min.   :   0.0001   Min.   :      0   Min.   :0.000e+00   Min.   :0.000222
##  1st Qu.:   0.0245   1st Qu.:      0   1st Qu.:3.000e+00   1st Qu.:0.002157
##  Median :   0.0808   Median :      1   Median :2.100e+01   Median :0.002958
##  Mean   :  92.4902   Mean   : 753370   Mean   :6.194e+09   Mean   :0.040394
##  3rd Qu.:   0.5345   3rd Qu.:     40   3rd Qu.:5.345e+03   3rd Qu.:0.087868
##  Max.   :1044.3000   Max.   :9140000   Max.   :8.000e+10   Max.   :0.172340
##       var                  sd                 skew             kurt
##  Min.   :   0.0001   Min.   : 0.01006   Min.   : 11.97   Min.   :   150.5
##  1st Qu.:   0.0245   1st Qu.: 0.15643   1st Qu.: 72.55   1st Qu.:  5622.3
##  Median :   0.0808   Median : 0.28423   Median :110.12   Median : 12158.7
##  Mean   :  92.4855   Mean   : 3.65112   Mean   :162.81   Mean   : 39749.6
##  3rd Qu.:   0.5268   3rd Qu.: 0.72579   3rd Qu.:269.54   3rd Qu.: 72732.4
##  Max.   :1044.2759   Max.   :32.31526   Max.   :344.91   Max.   :132136.7
```

```r
#histograms of responses (raw)
p1 <- ggplot(df_train, aes(x = mean)) + geom_histogram(bins=30) + ggtitle("Histogram: mean")
p2 <- ggplot(df_train, aes(x = sd)) + geom_histogram(bins=30) + ggtitle("Histogram: sd")
p3 <- ggplot(df_train, aes(x = skew)) + geom_histogram(bins=30) + ggtitle("Histogram: skewness")
p4 <- ggplot(df_train, aes(x = kurt)) + geom_histogram(bins=30) + ggtitle("Histogram: kurtosis")
grid.arrange(p1,p2,p3,p4, ncol=2)
```

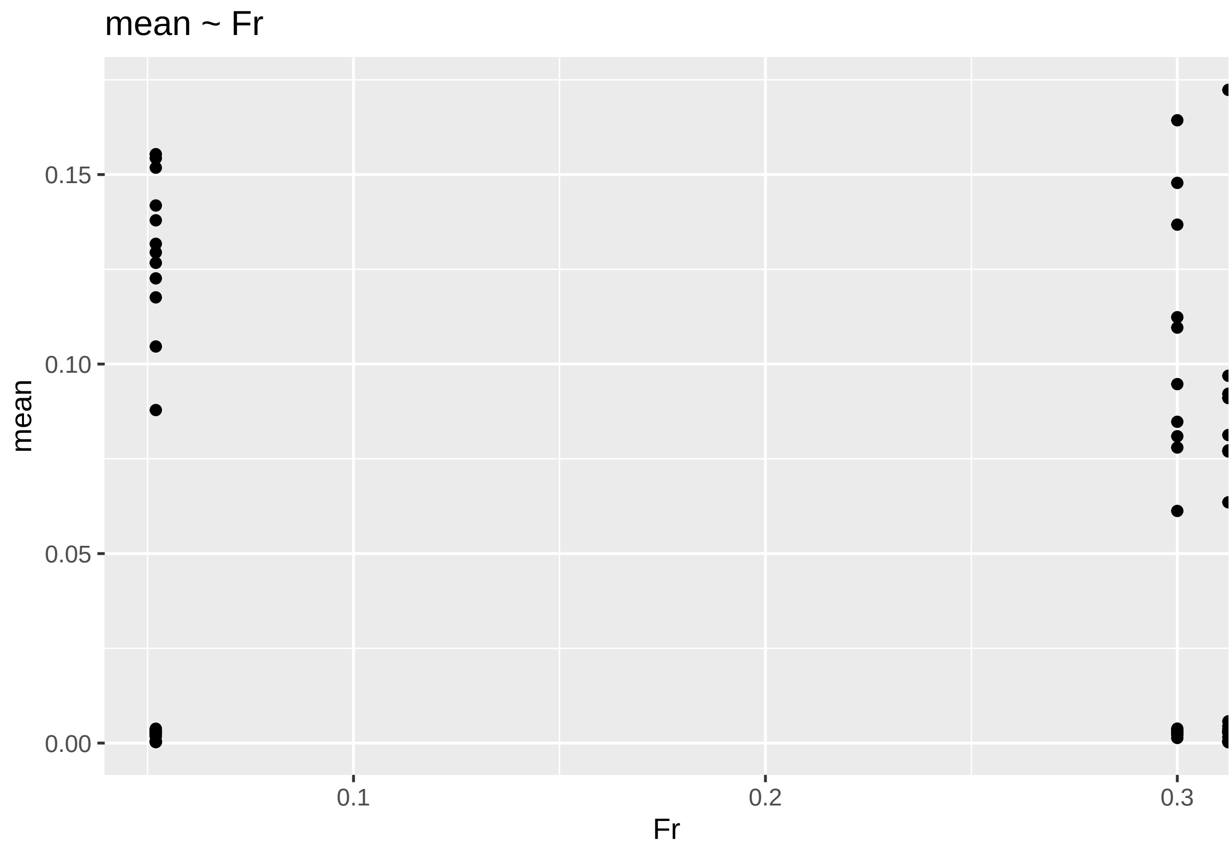



```r
# plots of inputs vs outputs
plot_grid_inputs <- list(
  ggplot(df_train, aes(x=Re, y=mean)) + geom_point() + ggtitle("mean ~ Re"),
  ggplot(df_train, aes(x=Fr, y=mean)) + geom_point() + ggtitle("mean ~ Fr"),
  ggplot(df_train, aes(x=St, y=mean)) + geom_point() + ggtitle("mean ~ St"),
  ggplot(df_train, aes(x=Re, y=sd)) + geom_point() + ggtitle("sd ~ Re"),
  ggplot(df_train, aes(x=Fr, y=sd)) + geom_point() + ggtitle("sd ~ Fr"),
  ggplot(df_train, aes(x=St, y=sd)) + geom_point() + ggtitle("sd ~ St"),
  ggplot(df_train, aes(x=Re, y=skew)) + geom_point() + ggtitle("skew ~ Re"),
  ggplot(df_train, aes(x=Fr, y=skew)) + geom_point() + ggtitle("skew ~ Fr"),
  ggplot(df_train, aes(x=St, y=skew)) + geom_point() + ggtitle("skew ~ St"),
  ggplot(df_train, aes(x=Re, y=kurt)) + geom_point() + ggtitle("kurt ~ Re"),
  ggplot(df_train, aes(x=Fr, y=kurt)) + geom_point() + ggtitle("kurt ~ Fr"),
  ggplot(df_train, aes(x=St, y=kurt)) + geom_point() + ggtitle("kurt ~ St")
)
plot_grid_inputs
```

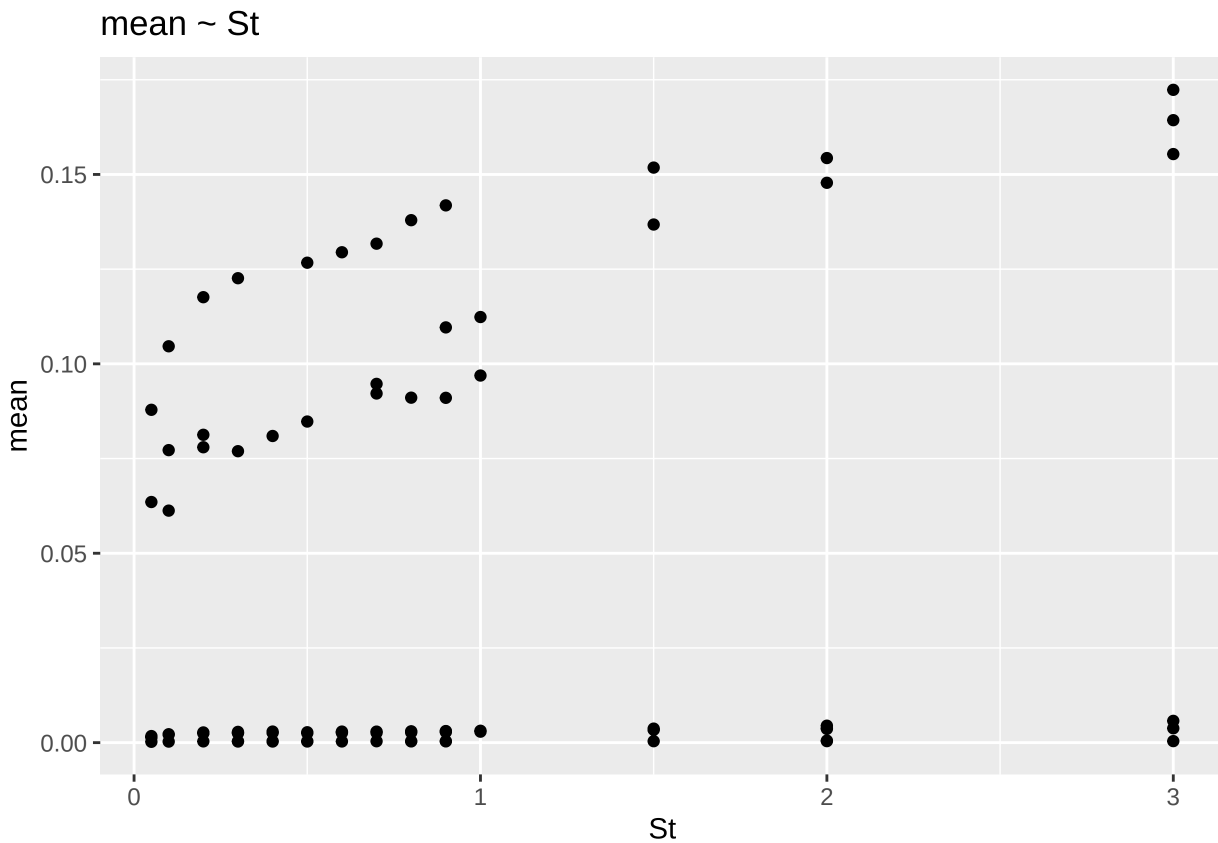
```
## [[1]]
```



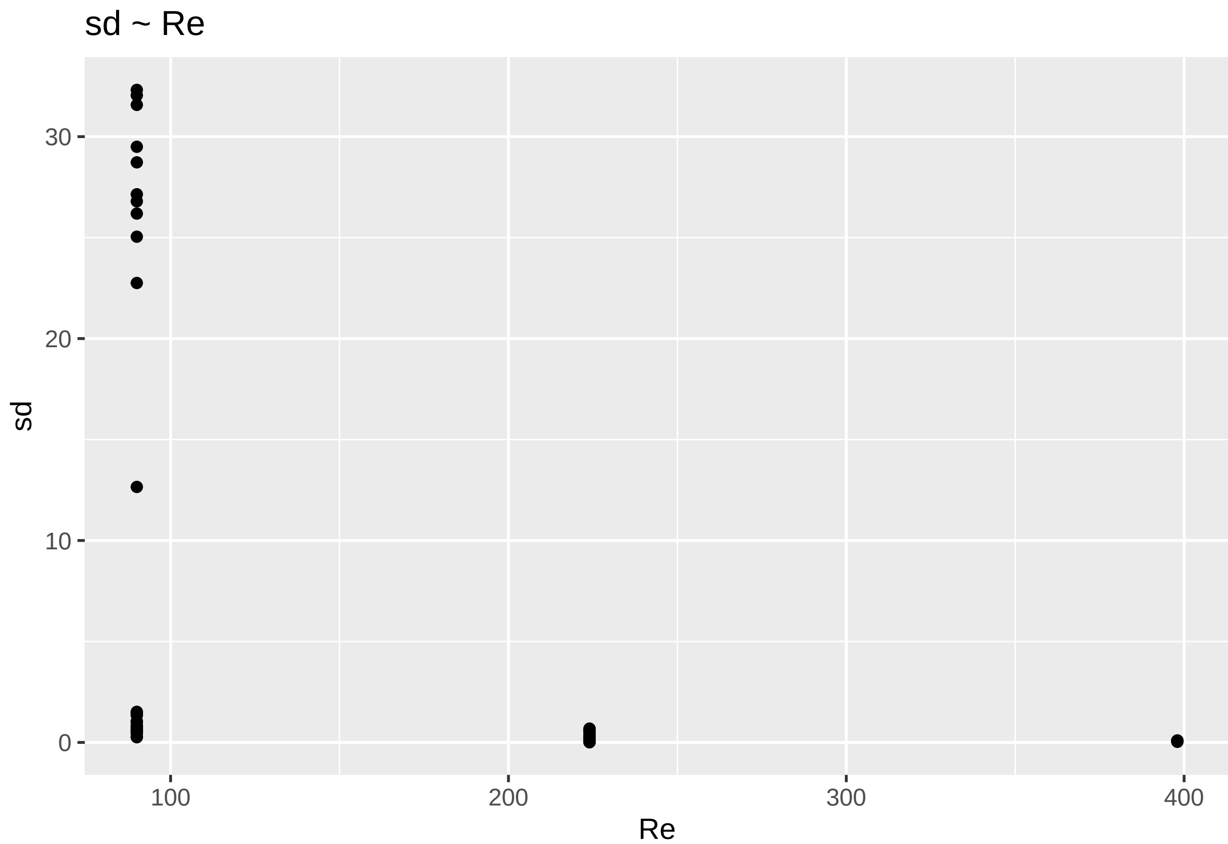

```
##
## [[2]]
```



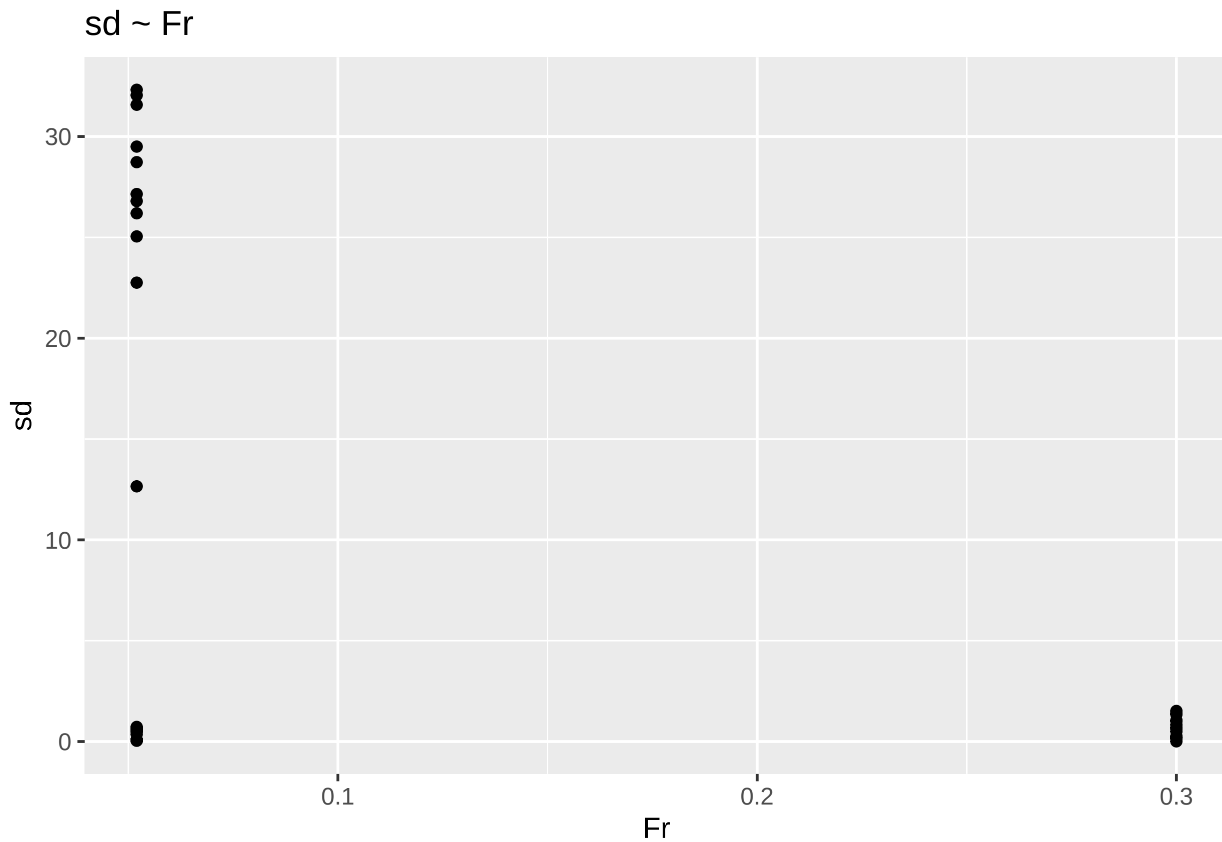

```
##
## [[3]]
```



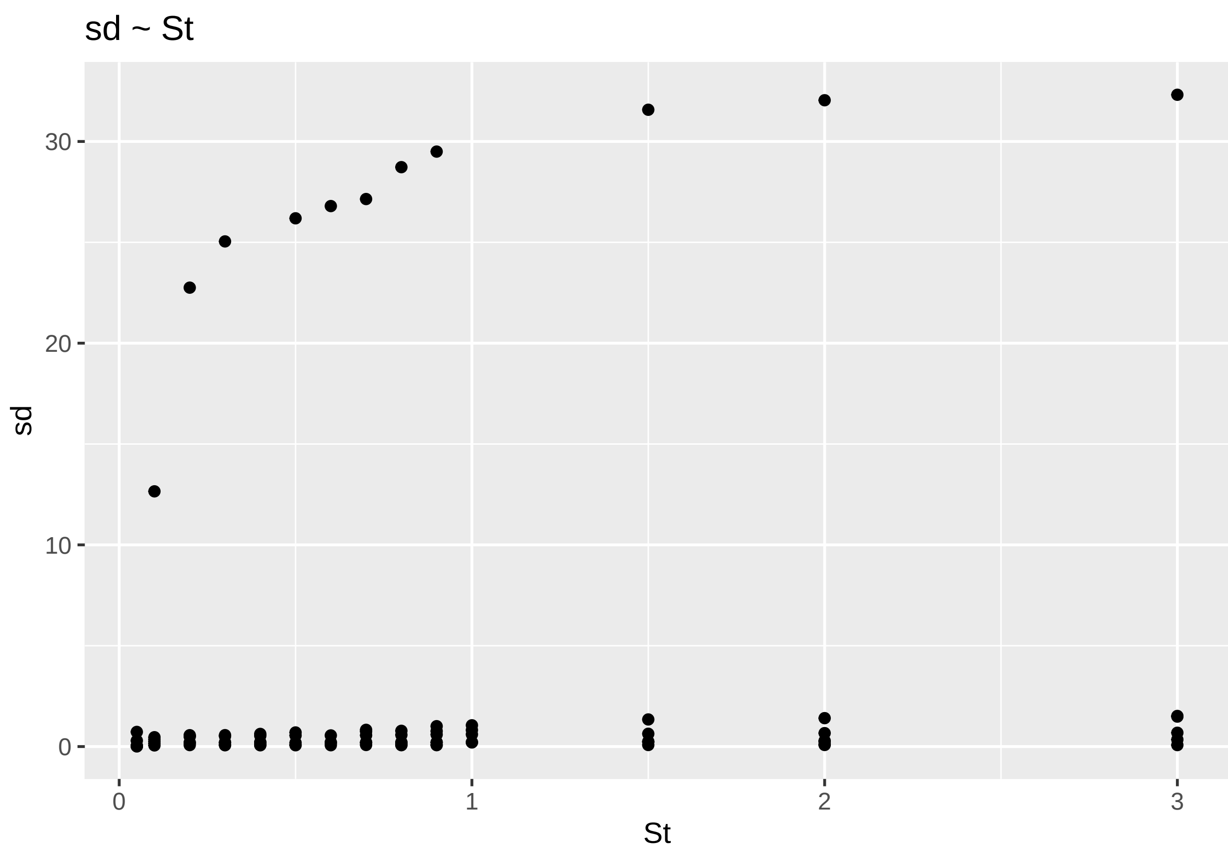

```
##
## [[4]]
```



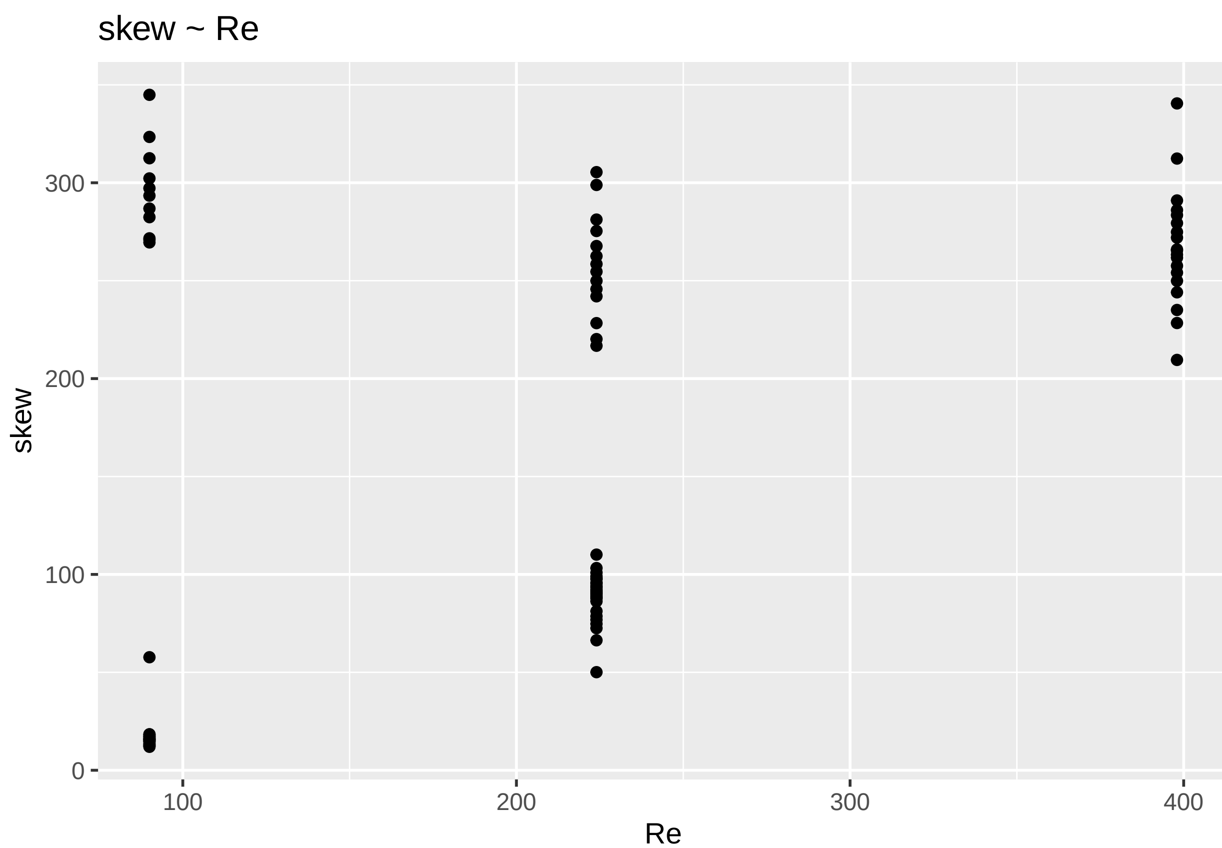

```
##
## [[5]]
```



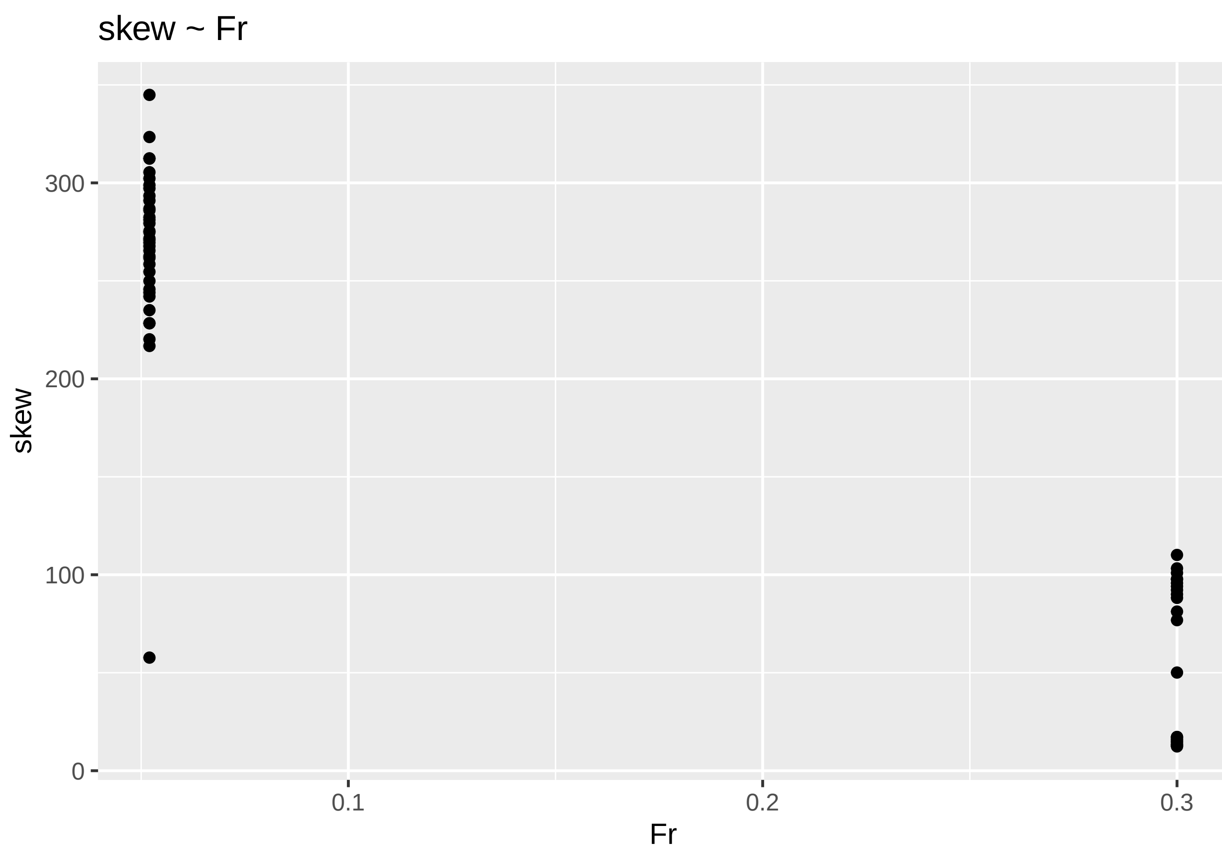

```
##
## [[6]]
```



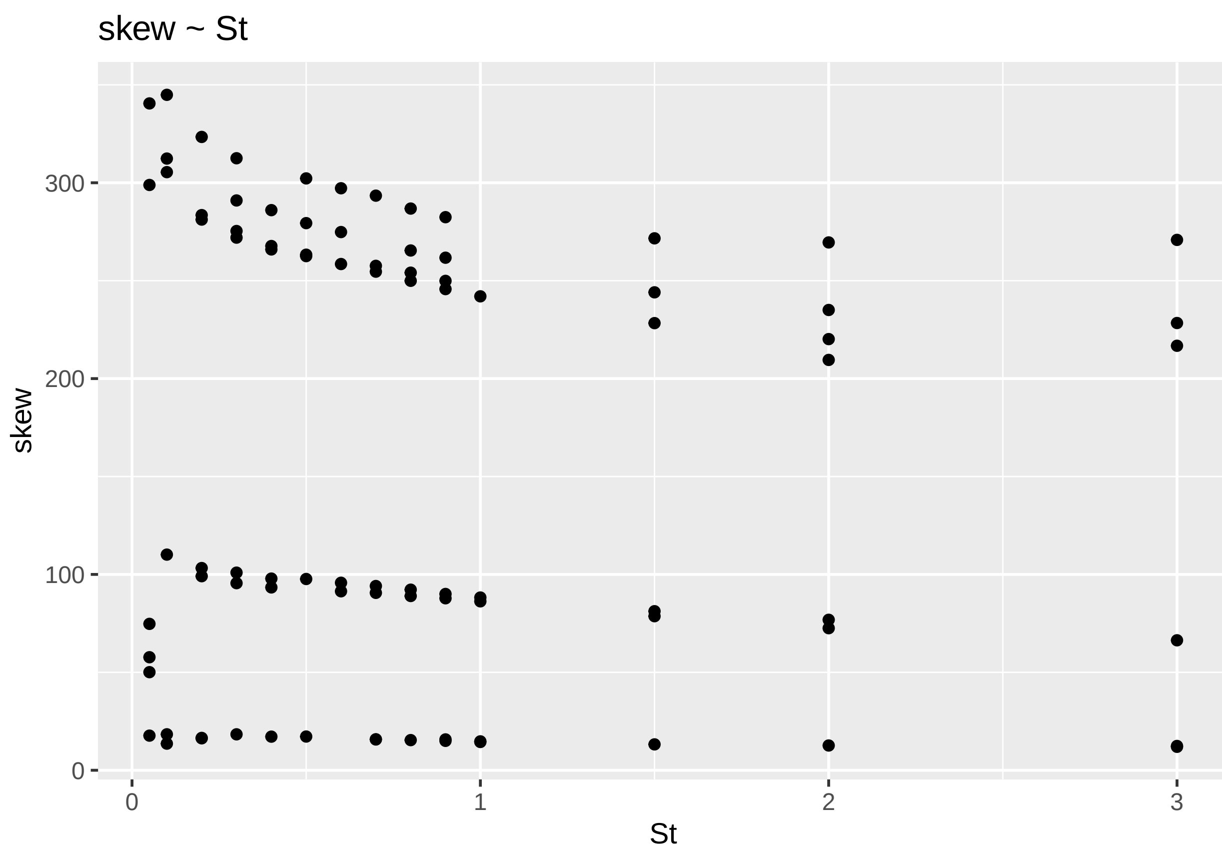

```
##
## [[7]]
```



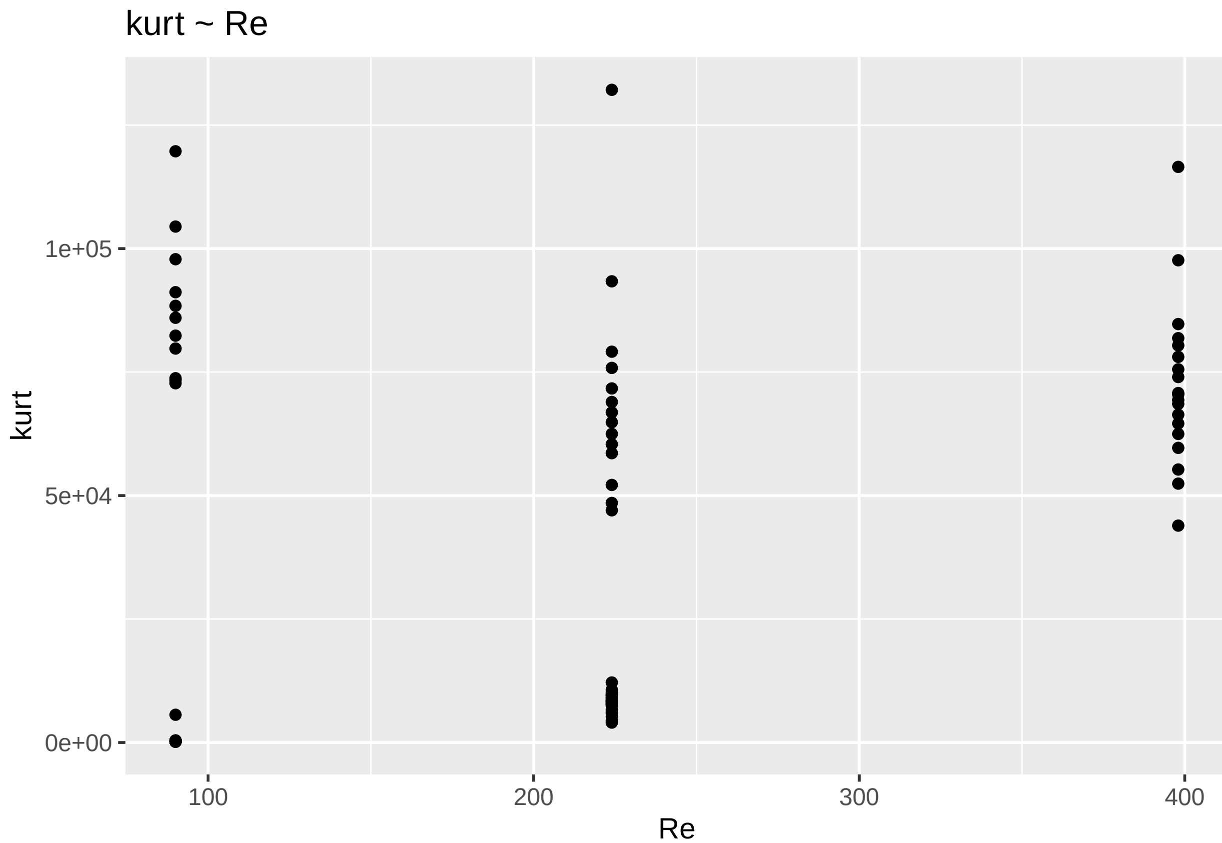

```
##
## [[8]]
```



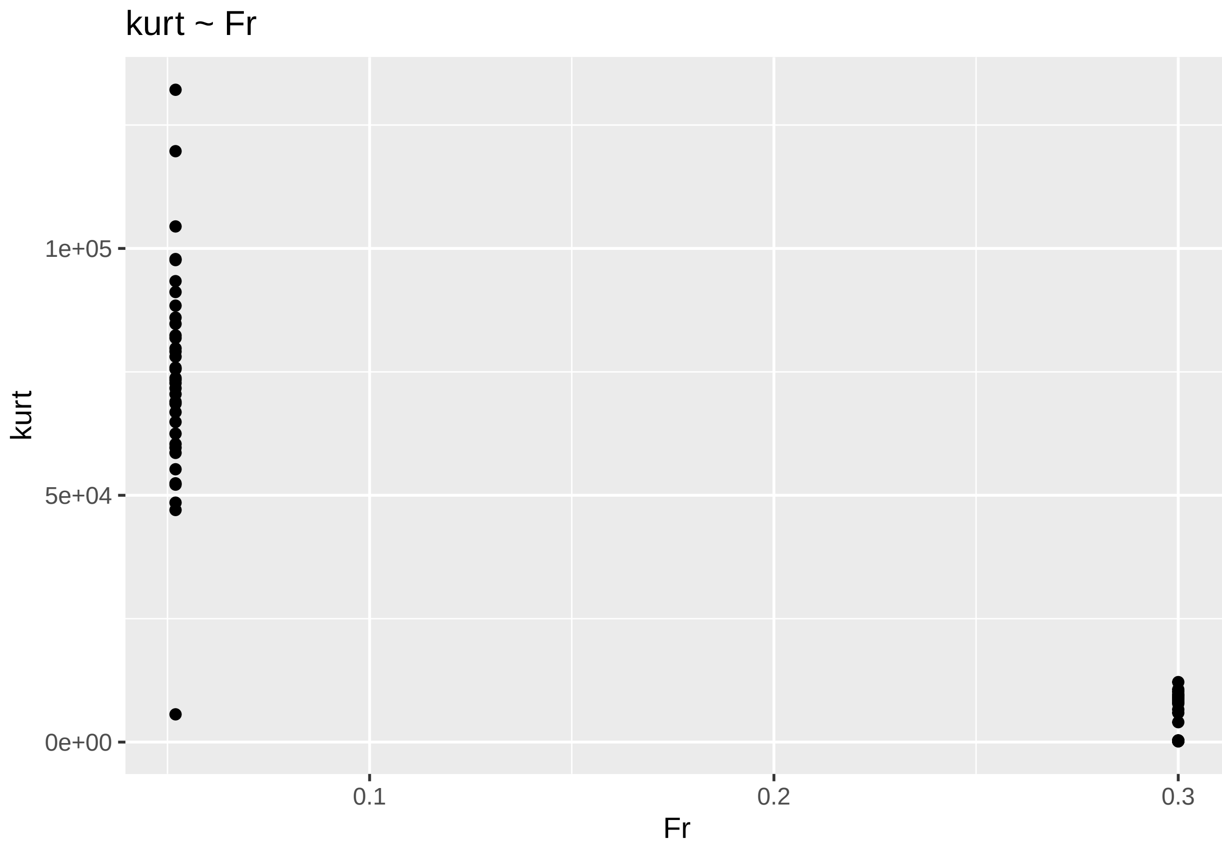

```
##
## [[9]]
```



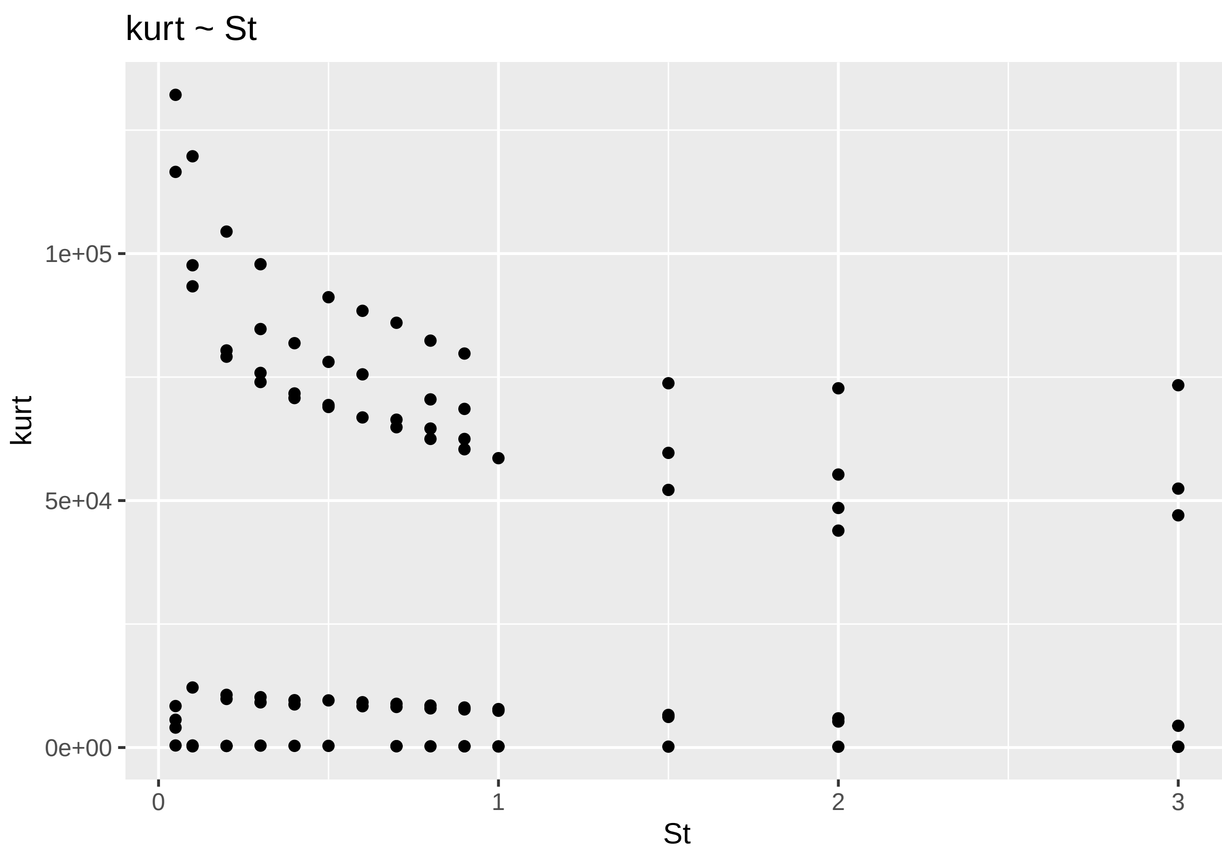

```
##
## [[10]]
```




```
##
## [[11]]
```




```
##
## [[12]]
```




```r
#when i asked chat to check my above stuff and expand on it, it gave me this so not sure if this is ful
# Many moment-derived responses are highly skewed -> log-transform candidates
# Inspect positive/zero entries
df_train %>% summarize(
  mean_min = min(mean, na.rm=TRUE),
  sd_min = min(sd, na.rm=TRUE),
  skew_min = min(skew, na.rm=TRUE)
) %>% print()
```

```
## # A tibble: 1 x 3
##   mean_min sd_min skew_min
##      <dbl>  <dbl>    <dbl>
## 1 0.000222 0.0101     12.0
```

```r
# We'll create transformed responses:
df_train <- df_train %>%
  mutate(
    log_mean = if_else(mean > 0, log(mean), NA_real_),
    log_sd   = if_else(sd > 0, log(sd), NA_real_),
    # skew and kurtosis can be negative/positive; use raw skew but consider robust transforms for model
    sign_skew = sign(skew),
    abs_log_skew = if_else(!is.na(skew) & skew != 0, sign_skew * log(abs(skew)), NA_real_),
    log_kurt = if_else(kurt > 0, log(kurt), NA_real_)
  )
```

5. Train-Test Split for model validation

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.2.0 --

## v dials        1.3.0     v rsample      1.2.1
## v infer        1.0.7     v tune         1.2.1
## v modeldata    1.4.0     v workflows    1.1.4
## v parsnip      1.2.1     v workflowsets 1.1.0
## v recipes      1.1.0     v yardstick    1.3.1

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x gridExtra::combine()     masks dplyr::combine()
## x scales::discard()        masks purrr::discard()
## x dplyr::filter()          masks stats::filter()
## x recipes::fixed()         masks stringr::fixed()
## x dplyr::lag()             masks stats::lag()
## x caret::lift()            masks purrr::lift()
## x pdp::partial()           masks purrr::partial()
## x yardstick::precision()   masks caret::precision()
## x yardstick::recall()      masks caret::recall()
## x yardstick::sensitivity() masks caret::sensitivity()
## x yardstick::spec()        masks readr::spec()
## x yardstick::specificity() masks caret::specificity()
## x recipes::step()          masks stats::step()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
set.seed(325)

split <- initial_split(df_train, prop = 0.8)
train_from_split <- training(split)
test_from_split  <- testing(split)
```

6. Baseline linear model

```
train_from_split <- train_from_split |>
  mutate(across(c(Re, Fr, St), ~ ifelse(is.infinite(.x), max(.x[is.finite(.x)], na.rm=TRUE), .x)))

test_from_split <- test_from_split |>
  mutate(across(c(Re, Fr, St), ~ ifelse(is.infinite(.x), max(.x[is.finite(.x)], na.rm=TRUE), .x)))

f_base <- as.formula("log_mean ~ Re + Fr + St")

lm_base <- lm(f_base, data = train_from_split)
summary(lm_base)
```

```
##
## Call:
## lm(formula = f_base, data = train_from_split)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -1.0237 -0.5029  0.1557  0.5189  1.0691
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.8518958  0.2138744  -3.983  0.00017 ***
## Re          -0.0199188  0.0006851 -29.076  < 2e-16 ***
## Fr          -0.9271696  0.5730538  -1.618  0.11037
## St           0.2150644  0.0906528   2.372  0.02055 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5929 on 67 degrees of freedom
## Multiple R-squared:  0.9282, Adjusted R-squared:  0.925
## F-statistic: 288.8 on 3 and 67 DF,  p-value: < 2.2e-16
```

```r
pred_base <- predict(lm_base, newdata = test_from_split)

rmse_base <- sqrt(mean((test_from_split$log_mean - pred_base)^2, na.rm=TRUE))
r2_base <- cor(test_from_split$log_mean, pred_base, use="complete.obs")^2

cat("Baseline RMSE (log_mean):", rmse_base, " R^2:", r2_base, "\n")
```

```
## Baseline RMSE (log_mean): 0.6559594  R^2: 0.9617176
```

7. Nonlinearity assessment: Add polynomial terms (squared, cubic) # Fit models with polynomial terms # Compare with linear model (AIC, BIC, R^2) # Residual plots to check for patterns # Partial dependence plots

```r
#polynomial models for Re, St and numeric Fr
f_poly2 <- as.formula("log_mean ~ poly(Re,3, raw=TRUE) + poly(St,3, raw=TRUE)  + poly(Fr,3, raw=TRUE)")
lm_poly2 <- lm(f_poly2, data = train_from_split)
summary(lm_poly2)
```

```
##
## Call:
## lm(formula = f_poly2, data = train_from_split)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.36962 -0.08059 -0.00243  0.08267  0.36608
##
## Coefficients: (3 not defined because of singularities)
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             9.490e-01  8.918e-02  10.642 8.59e-16 ***
## poly(Re, 3, raw = TRUE)1 -4.284e-02  7.204e-04 -59.462  < 2e-16 ***
## poly(Re, 3, raw = TRUE)2  4.924e-05  1.511e-06  32.591  < 2e-16 ***
## poly(Re, 3, raw = TRUE)3         NA         NA      NA       NA
## poly(St, 3, raw = TRUE)1  7.517e-01  1.650e-01   4.556 2.40e-05 ***
## poly(St, 3, raw = TRUE)2 -2.989e-01  1.436e-01  -2.081   0.0414 *
## poly(St, 3, raw = TRUE)3  4.377e-02  3.261e-02   1.342   0.1842
## poly(Fr, 3, raw = TRUE)1 -3.184e-01  1.401e-01  -2.273   0.0264 *
## poly(Fr, 3, raw = TRUE)2         NA         NA      NA       NA
```

```
## poly(Fr, 3, raw = TRUE)3         NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1432 on 64 degrees of freedom
## Multiple R-squared:  0.996,  Adjusted R-squared:  0.9956
## F-statistic:  2658 on 6 and 64 DF,  p-value: < 2.2e-16
```

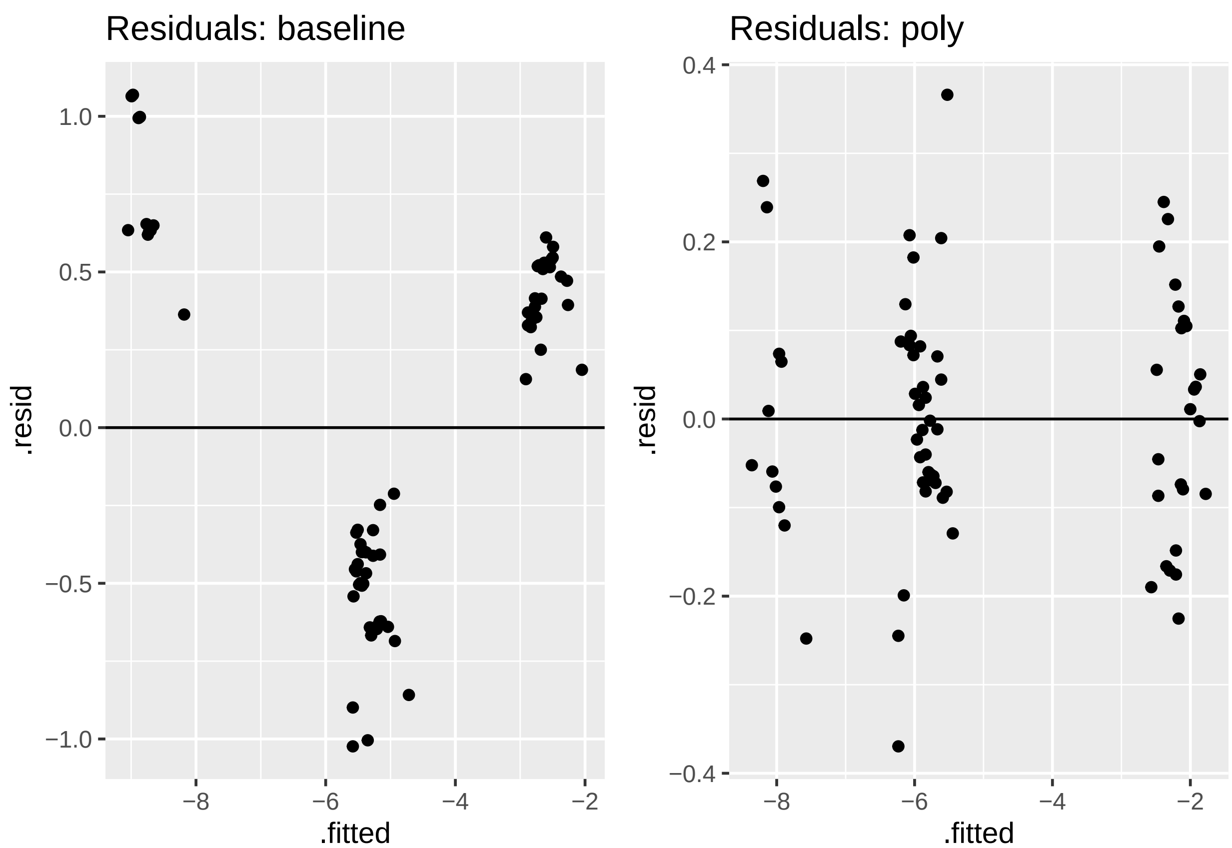
```
# look at AIC/BIC and R2 on validation
models <- list(base = lm_base, poly = lm_poly2)
model_comp <- tibble(
  model = names(models),
  AIC = map_dbl(models, AIC),
  BIC = map_dbl(models, BIC),
  adjR2 = map_dbl(models, ~summary(.x)$adj.r.squared)
)
print(model_comp)
```

```
## # A tibble: 2 x 4
##   model   AIC   BIC adjR2
##   <chr> <dbl> <dbl> <dbl>
## 1 base  133.  144.  0.925
## 2 poly  -65.9 -47.8 0.996
```

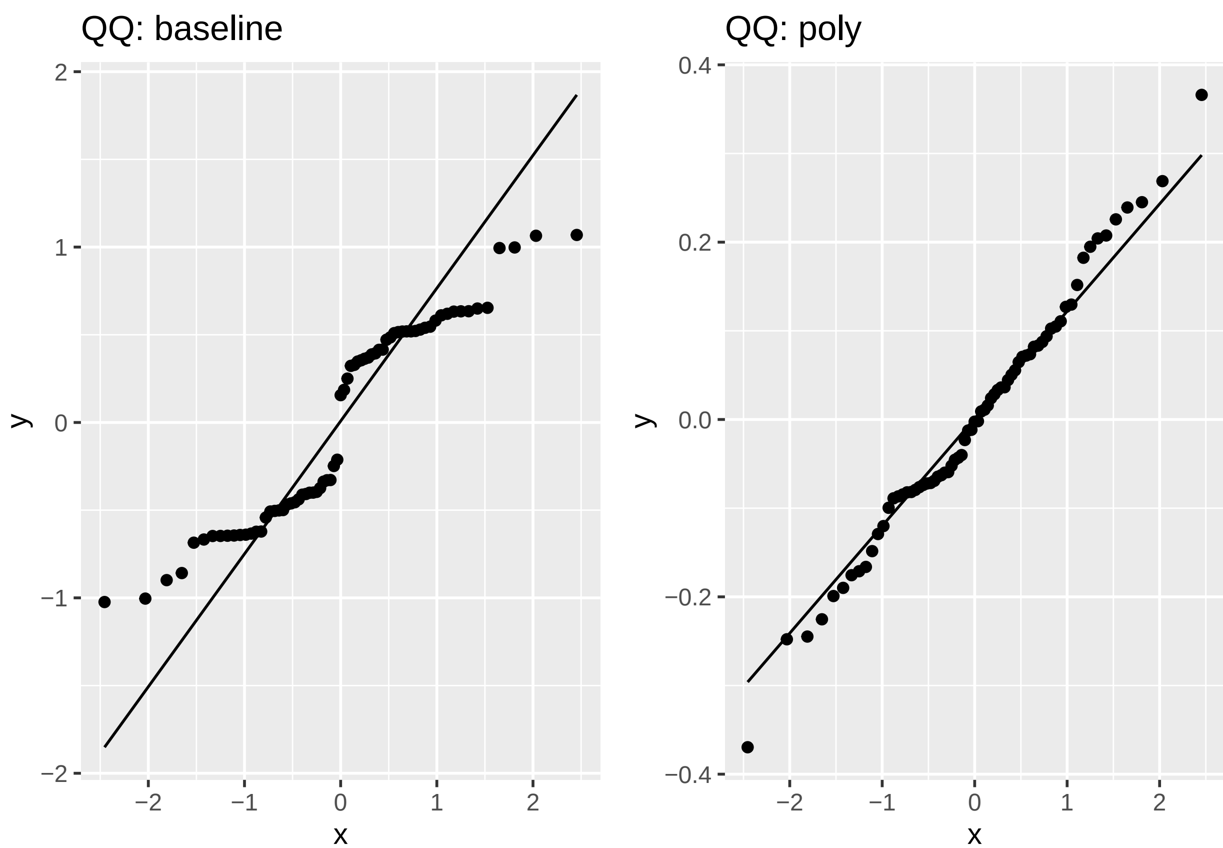
```
# validate
pred_poly <- predict(lm_poly2, newdata = test_from_split)
rmse_poly <- sqrt(mean((test_from_split$log_mean - pred_poly)^2, na.rm=TRUE))
r2_poly <- cor(test_from_split$log_mean, pred_poly, use="complete.obs")^2
cat("Poly RMSE (log_mean):", rmse_poly, " R^2:", r2_poly, "\n")
```

```
## Poly RMSE (log_mean): 0.1668979  R^2: 0.9965375
```

```
# Residual plots
resid_plot_base <- ggplot(lm_base, aes(.fitted, .resid)) + geom_point() + geom_hline(yintercept = 0) + g
resid_plot_poly <- ggplot(lm_poly2, aes(.fitted, .resid)) + geom_point() + geom_hline(yintercept = 0) +
grid.arrange(resid_plot_base, resid_plot_poly, ncol=2)
```




```r
# Check QQ
qq_base <- ggplot(lm_base, aes(sample = .resid)) + stat_qq() + stat_qq_line() + ggtitle("QQ: baseline")
qq_poly <- ggplot(lm_poly2, aes(sample = .resid)) + stat_qq() + stat_qq_line() + ggtitle("QQ: poly")
grid.arrange(qq_base, qq_poly, ncol=2)
```



can see that polynomial one way better

8. Interaction Effects: Test two-way interactions (Re:Fr, Re:St, Fr:St) # Test three-way interactions if appropriate # Fit models with interaction terms # Use ANOVA or model comparison to assess significance # Visualize interaction effects

```
f_int_2way <- as.formula("log_mean ~ poly(Re,2,raw=TRUE)*poly(St,2,raw=TRUE)  + poly(Fr,2,raw=TRUE)")
lm_int_2way <- lm(f_int_2way, data = train_from_split)

#  pairwise interaction models
f_re_fr <- as.formula("log_mean ~ Re * Fr + St")
f_re_st <- as.formula("log_mean ~ Re * St + Fr")
f_fr_st <- as.formula("log_mean ~ Fr * St + Re")

mod_re_fr <- lm(f_re_fr, data = train_from_split)
mod_re_st <- lm(f_re_st, data = train_from_split)
mod_fr_st <- lm(f_fr_st, data = train_from_split)

# vcompare models
anova(mod_re_fr, lm_base)   # Re:Fr
```

```
## Analysis of Variance Table
##
## Model 1: log_mean ~ Re * Fr + St
## Model 2: log_mean ~ Re + Fr + St
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
```

```
## 1     66 23.489
## 2     67 23.550 -1 -0.060948 0.1713 0.6803
```

```
anova(mod_re_st, lm_base)   # Re:St
```

```
## Analysis of Variance Table
##
## Model 1: log_mean ~ Re * St + Fr
## Model 2: log_mean ~ Re + Fr + St
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     66 23.141
## 2     67 23.550 -1  -0.40924 1.1672 0.2839
```

```
anova(mod_fr_st, lm_base)   # Fr:St
```

```
## Analysis of Variance Table
##
## Model 1: log_mean ~ Fr * St + Re
## Model 2: log_mean ~ Re + Fr + St
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     66 23.365
## 2     67 23.550 -1  -0.18502 0.5226 0.4723
```

```
# AIC comparisons
mods <- list(base = lm_base, re_fr = mod_re_fr, re_st = mod_re_st, fr_st = mod_fr_st, all2way = lm_int_
tibble(name = names(mods),
       AIC = map_dbl(mods, AIC),
       BIC = map_dbl(mods, BIC)) |> arrange(AIC) |> print()
```

```
## # A tibble: 5 x 3
##   name      AIC   BIC
##   <chr>   <dbl> <dbl>
## 1 all2way -63.3 -38.4
## 2 base    133.  144.
## 3 re_st   134.  147.
## 4 fr_st   135.  148.
## 5 re_fr   135.  149.
```

```
#need help debugging whole section below
#Visualize an interaction: Re * St effect on predicted log_mean
#new_grid <- expand.grid(
#  Re = seq(min(df_train$Re, na.rm=TRUE), max(df_train$Re, na.rm=TRUE), length.out = 40),
#  St = seq(min(df_train$St, na.rm=TRUE), max(df_train$St, na.rm=TRUE), length.out = 40),
#  Fr = seq(min(df_train$Fr, na.rm=TRUE), max(df_train$Fr, na.rm=TRUE), length.out = 40),
#)
#NEED HELP DeBUGGING LINE ABOVE BC OF FR. DO WE MAKE FR INF AND NUM?

#predict (doesnt load yet bc need to figure out how to debug above)
#new_grid$pred <- predict(lm_int_2way, newdata = new_grid, se.fit = FALSE)
#ggplot(new_grid, aes(x=Re, y=St, fill=pred)) + geom_tile() + labs(title = "Predicted log_mean (Re #x S
```

Model Selection Summaries

```r
cat("\nModel comparison summary (AIC/BIC/adjR2):\n")
```

```
##
## Model comparison summary (AIC/BIC/adjR2):
```

```r
print(model_comp)
```

```
## # A tibble: 2 x 4
##   model   AIC   BIC adjR2
##   <chr> <dbl> <dbl> <dbl>
## 1 base  133.  144.  0.925
## 2 poly  -65.9 -47.8 0.996
```

```r
cat("\nValidation RMSE base vs poly:\n")
```

```
##
## Validation RMSE base vs poly:
```

```r
cat("Base RMSE:", rmse_base, " Poly RMSE:", rmse_poly, "\n")
```

```
## Base RMSE: 0.6559594  Poly RMSE: 0.1668979
```

All AIC and BIC are pretty similar for linear model and model with interaction. Really only changes a lot with poly model. Should we be concerned it is negative?