# Predictive Modeling for Particle Cluster Distributions

Particle Clustering Proj Group

2025-10-20

## Overview

**Goal**: Predict 4 summary statistics of particle cluster volume distributions

- **Mean** ($\mu$): Average cluster volume
- **Variance** ($\sigma^2$): Variability in cluster sizes

- **Skewness** ($\gamma$): Asymmetry of distribution
- **Kurtosis** ($\kappa$): Tail heaviness

**Modeling Pipeline** for each response:

1. Apply log transformations to handle right skew
2. Fit full model with 2-way interactions
3. **Best subset selection** → Choose which variables to include
4. **OLS vs Ridge** → Choose how to estimate coefficients

5. Select final model with best CV performance (10 fold)

---

## 1. Setup

### 1.1 Load Packages

```r
library(tidyverse)
library(leaps)      # best subset selection
library(glmnet)     # ridge regression
set.seed(325)       # reproducibility
```

### 1.2 Load Data

```r
train <- read_csv("data-train-processed.csv")
cat("# observations:", nrow(train))
```

```
## # observations: 89
```

## 1.3 Transform Predictors

```
train <- train %>%
  mutate(
    # Re: Categorical (3 levels: 90, 224, 398)
    Re = as.factor(Re),

    # Fr: Inverse logit (handles Inf values)
    Fr_invlogit = 1 / (1 + exp(-as.numeric(Fr))),

    # St: Log transform (right-skewed)
    log_St = log(St)
  )
```

## 1.4 Transform Responses

```
#log
train <- train %>%
  mutate(
    log_mean = log(mean),
    log_variance = log(variance),
    log_skewness = log(skewness),
    log_kurtosis = log(kurtosis)
  )
```

## 1.5 Helper Functions

```
# 10-fold cross-validation RMSE
cv_rmse <- function(data, formula, k = 10) {
  # Create equal-sized folds
  n <- nrow(data)
  fold_ids <- sample(rep(1:k, length.out = n))
  data$fold <- fold_ids

  cv_errors <- numeric(k)

  for(i in 1:k) {
    train_fold <- data[data$fold != i, ]
    test_fold <- data[data$fold == i, ]

    model <- lm(formula, data = train_fold)
    preds <- predict(model, newdata = test_fold)

    response <- all.vars(formula)[1]
    cv_errors[i] <- mean((test_fold[[response]] - preds)^2)
  }

  return(sqrt(mean(cv_errors)))
}
```

```r
# top models from regsubsets
show_top_models <- function(regfit, n = 5) {
  summ <- summary(regfit)
  data.frame(
    n_vars = 1:length(summ$bic),
    BIC = summ$bic,
    adj_R2 = summ$adjr2
  ) %>%
    arrange(BIC) %>%
    head(n)
}

# clean up by extracting variable names from regsubsets, handling factors properly
get_var_names <- function(regfit, nvars) {
  coef_names <- names(coef(regfit, nvars))[-1]  # remove intercept

  # replace factor dummy variables (Re90, Re224, Re398) with original factor name
  coef_names <- gsub("Re90|Re224|Re398", "Re", coef_names)

  # Remove duplicates (in case Re appeared multiple times)
  coef_names <- unique(coef_names)

  return(coef_names)
}
```

---

# 2. Model 1: MEAN

## 2.1 Fit Full Model

```r
cat("=== MODELING MEAN ===\n\n")
```

```
## === MODELING MEAN ===
```

```r
#full model
formula_mean_full <- log_mean ~ log_St + Re + Fr_invlogit + log_St:Re + log_St:Fr_invlogit + Re:Fr_invlo

lm_mean_full <- lm(formula_mean_full, data = train)
cat("Full model R²:", summary(lm_mean_full)$r.squared, "\n")
```

```
## Full model R²: 0.9975587
```

```r
cat("Full model variables:", length(coef(lm_mean_full)) - 1)
```

```
## Full model variables: 9
```

A full multiple linear regression model was fitted to predict log_mean using the three predictors (log_St, Re, and Fr_invlogit) and all pairwise interaction terms. The resulting model achieved an $R^2$ of 99.75% with 9 model variables, indicating an excellent fit to the training data and suggesting that the predictors capture almost all variation in the mean cluster volume. The inclusion of interaction terms is physically meaningful, as particle inertia effects may depend on turbulence and gravity. However, such a high $R^2$ could suggest overfitting, so subsequent steps will use cross-validation and subset selection to ensure generalizability.

## 2.2 Best Subset Selection

```r
# subset selection
regfit_mean <- regsubsets(formula_mean_full, data = train, nvmax = 10, method = "exhaustive")

# top 5 models
cat("Top 5 models by BIC:\n")
```

```
## Top 5 models by BIC:
```

```r
print(show_top_models(regfit_mean, n = 5))
```

```
##   n_vars       BIC    adj_R2
## 1      6 -496.0621 0.9971378
## 2      7 -495.3559 0.9972230
## 3      8 -494.3390 0.9972959
## 4      9 -490.4698 0.9972806
## 5      5 -486.5179 0.9966893
```

```r
# Extract best model (lowest BIC)
best_models_mean <- show_top_models(regfit_mean)
best_size <- best_models_mean$n_vars[1]
best_vars <- get_var_names(regfit_mean, best_size)

# Note: regsubsets reports n_vars including dummy variables (e.g., Re224, Re398)
# After collapsing dummies back to factors, we have fewer distinct terms
cat("Best model has", length(best_vars), "variables (", best_size, "including dummies)\n")
```

```
## Best model has 4 variables ( 6 including dummies)
```

```r
cat("Variables:", paste(best_vars, collapse = ", "))
```

```
## Variables: log_St, Re, Fr_invlogit, Re:Fr_invlogit
```

Best subset selection identified a 6-variable model as optimal based on the BIC score, achieving an adjusted $R^2$ of 99.71%, which is nearly identical to the full model's performance but with fewer terms. This indicates that some interaction terms are unnecessary and that the simplified model provides a better balance between accuracy and interpretability, reducing the risk of overfitting.

## 2.3 Build Best Model Formula

```r
#create formula to load
formula_mean_best <- as.formula(paste("log_mean ~", paste(best_vars, collapse = " + ")))
cat("best model formula:\n")
```

```
## best model formula:
```

```r
print(formula_mean_best)
```

```
## log_mean ~ log_St + Re + Fr_invlogit + Re:Fr_invlogit
```

The 6 variables reported by best subset selection include dummy variables automatically created for the categorical predictor `Re`. Once these are collapsed back into the original factor, the final model consists of 4 unique predictors (`log_St`, `Re`, `Fr_invlogit`, and `Re:Fr_invlogit`).

### 2.4 Compare OLS vs Ridge

```r
# fit OLS
lm_mean_best <- lm(formula_mean_best, data = train)
cv_ols_mean <- cv_rmse(train, formula_mean_best, k = 10)
cat("OLS CV RMSE:", cv_ols_mean, "\n")
```

```
## OLS CV RMSE: 0.1247262
```

```r
# Fit Ridge on best model
X_mean <- model.matrix(formula_mean_best, data = train)[, -1]
y_mean <- train$log_mean
ridge_mean <- cv.glmnet(X_mean, y_mean, alpha = 0, nfolds = 10)
cv_ridge_mean <- min(sqrt(ridge_mean$cvm))
cat("Ridge CV RMSE:", cv_ridge_mean, "\n")
```

```
## Ridge CV RMSE: 0.3657629
```

```r
cat("Best lambda:", ridge_mean$lambda.min, "\n")
```

```
## Best lambda: 0.1521061
```

### 2.5 Select Final Model

```r
if(cv_ols_mean < cv_ridge_mean) {
  cat("final model OLS (lower CV RMSE)\n")
  final_model_mean <- lm_mean_best
  use_ridge_mean <- FALSE
} else {
  cat("final model ridge (lower CV RMSE)\n")
  final_model_mean <- ridge_mean
  use_ridge_mean <- TRUE
}
```

```
## final model OLS (lower CV RMSE)
```

**Comparing OLS and ridge regression showed that OLS achieved a lower cross-validated RMSE (~0.125) than ridge (~0.366), indicating better predictive performance and no evidence of overfitting in the simpler best-subset model. Ridge regularization did not improve results, likely because the model is already low in complexity and multicollinearity is minimal.**

# 3. Model 2: VARIANCE

## 3.1 Fit Full Model

```
cat("=== MODELING VARIANCE ===\n\n")
```

```
## === MODELING VARIANCE ===
```

```
#full model
formula_var_full <- log_variance ~ log_St + Re + Fr_invlogit + log_St:Re + log_St:Fr_invlogit + Re:Fr_i

lm_var_full <- lm(formula_var_full, data = train)
cat("Full model R²:", summary(lm_var_full)$r.squared, "\n")
```

```
## Full model R²: 0.7648598
```

```
cat("Full model variables:", length(coef(lm_var_full)) - 1)
```

```
## Full model variables: 9
```

## 3.2 Best Subset Selection

```
# subset selection
regfit_var <- regsubsets(formula_var_full, data = train, nvmax = 10, method = "exhaustive")

# top 5 models
cat("Top 5 models by BIC:\n")
```

```
## Top 5 models by BIC:
```

```
print(show_top_models(regfit_var, n = 5))
```

```
##   n_vars       BIC    adj_R2
## 1      6 -95.00596 0.7407348
## 2      7 -92.26035 0.7426243
## 3      5 -89.58117 0.7136781
## 4      8 -88.43433 0.7413401
## 5      4 -84.72355 0.6857607
```

```r
best_models_var <- show_top_models(regfit_var)
best_size <- best_models_var$n_vars[1]
best_vars <- get_var_names(regfit_var, best_size)

# Note: regsubsets reports n_vars including dummy variables (e.g., Re224, Re398)
# After collapsing dummies back to factors, we have fewer distinct terms
cat("Best model has", length(best_vars), "variables (", best_size, "including dummies)\n")
```

```
## Best model has 4 variables ( 6 including dummies)
```

```r
cat("Variables:", paste(best_vars, collapse = ", "))
```

```
## Variables: log_St, Re, Fr_invlogit, Re:Fr_invlogit
```

### 3.3 Build Best Model Formula

```r
#formula
formula_var_best <- as.formula(paste("log_variance ~", paste(best_vars, collapse = " + ")))
cat("best model formula:\n")
```

```
## best model formula:
```

```r
print(formula_var_best)
```

```
## log_variance ~ log_St + Re + Fr_invlogit + Re:Fr_invlogit
```

### 3.4 Compare OLS vs Ridge

```r
# fit OLS
lm_var_best <- lm(formula_var_best, data = train)
cv_ols_var <- cv_rmse(train, formula_var_best, k = 10)
cat("OLS CV RMSE:", round(cv_ols_var, 4), "\n")
```

```
## OLS CV RMSE: 1.9255
```

```r
X_var <- model.matrix(formula_var_best, data = train)[, -1]
y_var <- train$log_variance
ridge_var <- cv.glmnet(X_var, y_var, alpha = 0, nfolds = 10)
cv_ridge_var <- min(sqrt(ridge_var$cvm))
cat("Ridge CV RMSE:", cv_ridge_var, "\n")
```

```
## Ridge CV RMSE: 2.120725
```

```r
cat("Best lambda:", ridge_var$lambda.min, "\n")
```

```
## Best lambda: 0.187422
```

### 3.5 Select Final Model

```r
if(cv_ols_var < cv_ridge_var) {
  cat("final model OLS (lower CV RMSE)\n")
  final_model_var <- lm_var_best
  use_ridge_var <- FALSE
} else {
  cat("final model ridge (lower CV RMSE)\n")
  final_model_var <- ridge_var
  use_ridge_var <- TRUE
}
```

```
## final model OLS (lower CV RMSE)
```

---

## 4. Model 3: SKEWNESS

### 4.1 Fit Full Model

```r
cat("=== MODELING SKEWNESS ===\n\n")
```

```
## === MODELING SKEWNESS ===
```

```r
# Define full model with all 2-way interactions
formula_skew_full <- log_skewness ~ log_St + Re + Fr_invlogit + log_St:Re + log_St:Fr_invlogit + Re:Fr_i

# Fit full model
lm_skew_full <- lm(formula_skew_full, data = train)
cat("Full model R²:", round(summary(lm_skew_full)$r.squared, 4), "\n")
```

```
## Full model R²: 0.5755
```

```r
cat("Full model variables:", length(coef(lm_skew_full)) - 1)
```

```
## Full model variables: 9
```

### 4.2 Best Subset Selection

```r
# subset selection
regfit_skew <- regsubsets(formula_skew_full, data = train, nvmax = 10, method = "exhaustive")

# top 5 models
cat("Top 5 models by BIC:\n")
```

```
## Top 5 models by BIC:
```

```r
print(show_top_models(regfit_skew, n = 5))
```

```
##   n_vars       BIC    adj_R2
## 1      3 -55.75313 0.5477366
## 2      4 -52.02324 0.5462375
## 3      5 -48.32191 0.5448150
## 4      6 -44.55641 0.5429923
## 5      7 -40.34076 0.5387671
```

```r
best_models_skew <- show_top_models(regfit_skew)
best_size <- best_models_skew$n_vars[1]
best_vars <- get_var_names(regfit_skew, best_size)

# Note: regsubsets reports n_vars including dummy variables (e.g., Re224, Re398)
# After collapsing dummies back to factors, we have fewer distinct terms
cat("Best model has", length(best_vars), "variables (", best_size, "including dummies)\n")
```

```
## Best model has 2 variables ( 3 including dummies)
```

```r
cat("Variables:", paste(best_vars, collapse = ", "))
```

```
## Variables: Fr_invlogit, Re:Fr_invlogit
```

### 4.3 Build Best Model Formula

```r
# Create formula for best model
formula_skew_best <- as.formula(paste("log_skewness ~", paste(best_vars, collapse = " + ")))
cat("best model formula:\n")
```

```
## best model formula:
```

```r
print(formula_skew_best)
```

```
## log_skewness ~ Fr_invlogit + Re:Fr_invlogit
```

### 4.4 Compare OLS vs Ridge

```r
# fit OLS
lm_skew_best <- lm(formula_skew_best, data = train)
cv_ols_skew <- cv_rmse(train, formula_skew_best, k = 10)
cat("OLS CV RMSE:", round(cv_ols_skew, 4), "\n")
```

```
## OLS CV RMSE: 0.763
```

```r
X_skew <- model.matrix(formula_skew_best, data = train)[, -1]
y_skew <- train$log_skewness
ridge_skew <- cv.glmnet(X_skew, y_skew, alpha = 0, nfolds = 10)
cv_ridge_skew <- min(sqrt(ridge_skew$cvm))
cat("Ridge CV RMSE:", cv_ridge_skew, "\n")
```

```
## Ridge CV RMSE: 0.7636996
```

```r
cat("Best lambda:", ridge_skew$lambda.min, "\n")
```

```
## Best lambda: 0.04552038
```

### 4.5 Select Final Model

```r
if(cv_ols_skew < cv_ridge_skew) {
  cat("final model OLS (lower CV RMSE)\n")
  final_model_skew <- lm_skew_best
  use_ridge_skew <- FALSE
} else {
  cat("final model ridge (lower CV RMSE)\n")
  final_model_skew <- ridge_skew
  use_ridge_skew <- TRUE
}
```

```
## final model OLS (lower CV RMSE)
```

---

## 5. Model 4: KURTOSIS

### 5.1 Fit Full Model

```r
cat("=== MODELING KURTOSIS ===\n\n")
```

```
## === MODELING KURTOSIS ===
```

```r
#full
formula_kurt_full <- log_kurtosis ~ log_St + Re + Fr_invlogit + log_St:Re + log_St:Fr_invlogit + Re:Fr_i

lm_kurt_full <- lm(formula_kurt_full, data = train)
cat("Full model R²:", round(summary(lm_kurt_full)$r.squared, 4), "\n")
```

```
## Full model R²: 0.5766
```

```r
cat("Full model variables:", length(coef(lm_kurt_full)) - 1)
```

```
## Full model variables: 9
```

### 5.2 Best Subset Selection

```r
# subset selection
regfit_kurt <- regsubsets(formula_kurt_full, data = train, nvmax = 10, method = "exhaustive")

# top 5 models
cat("Top 5 models by BIC:\n")
```

```
## Top 5 models by BIC:
```

```r
print(show_top_models(regfit_kurt, n = 5))
```

```
##   n_vars       BIC    adj_R2
## 1      3 -55.23086 0.5450748
## 2      4 -52.16813 0.5469756
## 3      5 -48.35386 0.5449784
## 4      6 -44.87171 0.5446085
## 5      7 -40.55728 0.5398879
```

```r
best_models_kurt <- show_top_models(regfit_kurt)
best_size <- best_models_kurt$n_vars[1]
best_vars <- get_var_names(regfit_kurt, best_size)

# Note: regsubsets reports n_vars including dummy variables (e.g., Re224, Re398)
# After collapsing dummies back to factors, we have fewer distinct terms
cat("Best model has", length(best_vars), "variables (", best_size, "including dummies)\n")
```

```
## Best model has 2 variables ( 3 including dummies)
```

```r
cat("Variables:", paste(best_vars, collapse = ", "))
```

```
## Variables: Fr_invlogit, Re:Fr_invlogit
```

### 5.3 Build Best Model Formula

```r
# formula
formula_kurt_best <- as.formula(paste("log_kurtosis ~", paste(best_vars, collapse = " + ")))
cat("best model formula:\n")
```

```
## best model formula:
```

```r
print(formula_kurt_best)
```

```
## log_kurtosis ~ Fr_invlogit + Re:Fr_invlogit
```

### 5.4 Compare OLS vs Ridge

```r
# fit OLS
lm_kurt_best <- lm(formula_kurt_best, data = train)
cv_ols_kurt <- cv_rmse(train, formula_kurt_best, k = 10)
cat("OLS CV RMSE:", round(cv_ols_kurt, 4), "\n")
```

```
## OLS CV RMSE: 1.5375
```

```r
X_kurt <- model.matrix(formula_kurt_best, data = train)[, -1]
y_kurt <- train$log_kurtosis
ridge_kurt <- cv.glmnet(X_kurt, y_kurt, alpha = 0, nfolds = 10)
cv_ridge_kurt <- min(sqrt(ridge_kurt$cvm))
cat("Ridge CV RMSE:", round(cv_ridge_kurt, 4), "\n")
```

```
## Ridge CV RMSE: 1.5408
```

```r
cat("Best lambda:", round(ridge_kurt$lambda.min, 6), "\n")
```

```
## Best lambda: 0.088804
```

### 5.5 Select Final Model

```r
if(cv_ols_kurt < cv_ridge_kurt) {
  cat("final model OLS (lower CV RMSE)\n")
  final_model_kurt <- lm_kurt_best
  use_ridge_kurt <- FALSE
} else {
  cat("final model ridge (lower CV RMSE)\n")
  final_model_kurt <- ridge_kurt
  use_ridge_kurt <- TRUE
}
```

```
## final model OLS (lower CV RMSE)
```

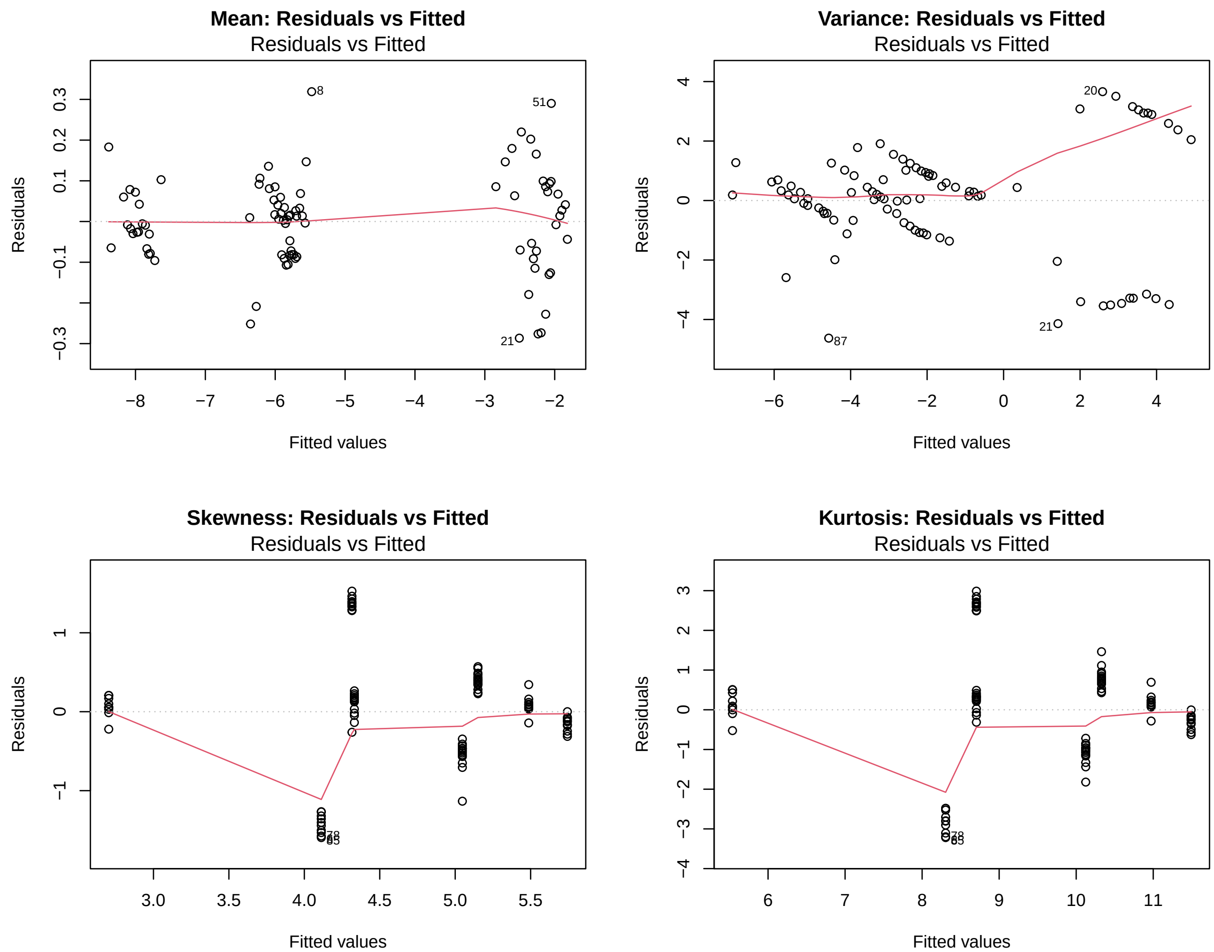
---

## 6. Model Diagnostics

### 6.1 Residual Plots

```r
par(mfrow = c(2, 2))

plot(lm_mean_best, which = 1, main = "Mean: Residuals vs Fitted")
plot(lm_var_best, which = 1, main = "Variance: Residuals vs Fitted")
plot(lm_skew_best, which = 1, main = "Skewness: Residuals vs Fitted")
plot(lm_kurt_best, which = 1, main = "Kurtosis: Residuals vs Fitted")
```



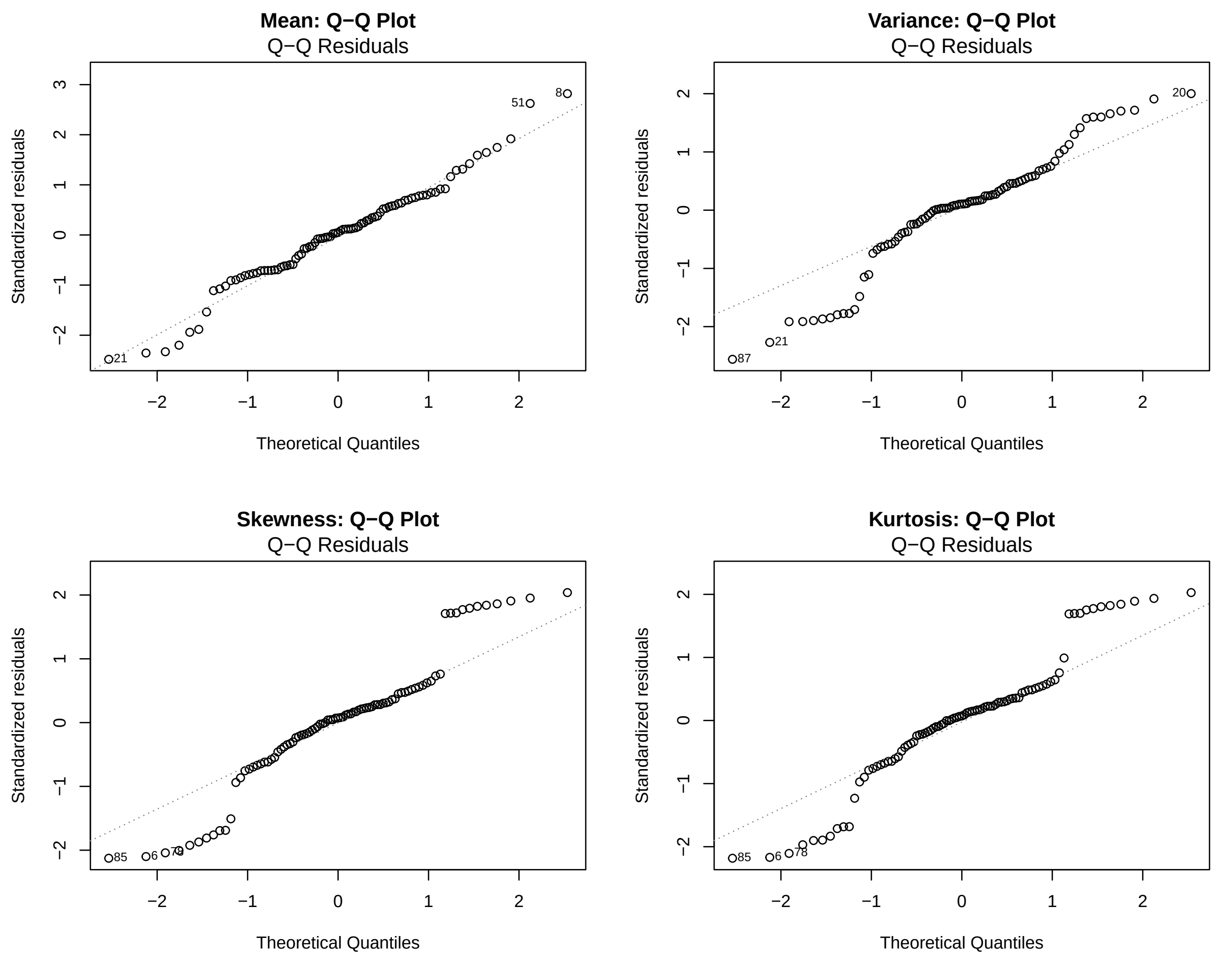

The residual plot shows that the mean model satisfies key assumptions: residuals are centered around zero with no strong patterns, indicating a good linear fit. In contrast, the residual plots for variance, skewness, and kurtosis show clear curvature and changing spread, suggesting nonlinear relationships and heteroskedasticity. These patterns indicate that the current linear models may be misspecified and that additional transformations or nonlinear modeling approaches may be needed to improve performance for higher-order distribution statistics.

## 6.2 Q-Q Plots

```r
par(mfrow = c(2, 2))

plot(lm_mean_best, which = 2, main = "Mean: Q-Q Plot")
```

```r
plot(lm_var_best, which = 2, main = "Variance: Q-Q Plot")
plot(lm_skew_best, which = 2, main = "Skewness: Q-Q Plot")
plot(lm_kurt_best, which = 2, main = "Kurtosis: Q-Q Plot")
```




The Q–Q plot for the mean model shows residuals close to normal, indicating the linear regression assumptions are well met. In contrast, the variance, skewness, and kurtosis models show substantial departures from normality, particularly in the tails, suggesting that these models may require more complex approaches to better capture the data patterns at the tails. (Same analysis as with residual plots)

---

# 7. Test Set Predictions

## 7.1 Load and Transform Test Data

```r
test <- read_csv("data-test.csv")
```

```r
# Apply same transformations as training data
test <- test %>%
  mutate(
    Re = as.factor(Re),
    Fr_invlogit = 1 / (1 + exp(-as.numeric(Fr))),
    log_St = log(St)
  )
```

### 7.2 Generate Predictions

```r
# predictions using the selected final models (OLS or Ridge)

# Mean predictions
if(use_ridge_mean) {
  # For Ridge, need design matrix matching training data
  test_temp <- test %>% mutate(log_mean = 0)  # dummy response for model.matrix
  X_test_mean <- model.matrix(formula_mean_best, data = test_temp)[, -1]
  pred_mean_log <- predict(ridge_mean, newx = X_test_mean, s = "lambda.min")[,1]
} else {
  pred_mean_log <- predict(lm_mean_best, newdata = test)
}

# Variance predictions
if(use_ridge_var) {
  test_temp <- test %>% mutate(log_variance = 0)  # dummy response
  X_test_var <- model.matrix(formula_var_best, data = test_temp)[, -1]
  pred_var_log <- predict(ridge_var, newx = X_test_var, s = "lambda.min")[,1]
} else {
  pred_var_log <- predict(lm_var_best, newdata = test)
}

# Skewness predictions
if(use_ridge_skew) {
  test_temp <- test %>% mutate(log_skewness = 0)  # dummy response
  X_test_skew <- model.matrix(formula_skew_best, data = test_temp)[, -1]
  pred_skew_log <- predict(ridge_skew, newx = X_test_skew, s = "lambda.min")[,1]
} else {
  pred_skew_log <- predict(lm_skew_best, newdata = test)
}

# Kurtosis predictions
if(use_ridge_kurt) {
  test_temp <- test %>% mutate(log_kurtosis = 0)  # dummy response
  X_test_kurt <- model.matrix(formula_kurt_best, data = test_temp)[, -1]
  pred_kurt_log <- predict(ridge_kurt, newx = X_test_kurt, s = "lambda.min")[,1]
} else {
  pred_kurt_log <- predict(lm_kurt_best, newdata = test)
}

# Combine and back-transform from log scale
test_predictions <- test %>%
  mutate(
```

```r
    pred_mean = exp(pred_mean_log),
    pred_variance = exp(pred_var_log),
    pred_skewness = exp(pred_skew_log),
    pred_kurtosis = exp(pred_kurt_log)
  ) %>%
  select(St, Re, Fr, pred_mean, pred_variance, pred_skewness, pred_kurtosis)

# display head
cat("First 10 predictions:\n")
```

```
## First 10 predictions:
```

```r
as.data.frame(test_predictions[1:10, ])
```

```
##      St  Re    Fr    pred_mean pred_variance pred_skewness pred_kurtosis
## 1  0.05 398 0.052 0.0002000637  0.0005030365      312.2236      97974.45
## 2  0.20 398 0.052 0.0002617158  0.0016459850      312.2236      97974.45
## 3  0.70 398 0.052 0.0003336218  0.0048046627      312.2236      97974.45
## 4  1.00 398 0.052 0.0003574949  0.0065180973      312.2236      97974.45
## 5  0.10 398   Inf 0.0002709088  0.0015091919      241.5953      58283.71
## 6  0.60 398   Inf 0.0003833593  0.0069847034      241.5953      58283.71
## 7  1.00 398   Inf 0.0004232469  0.0108106709      241.5953      58283.71
## 8  1.50 398   Inf 0.0004578415  0.0152907829      241.5953      58283.71
## 9  3.00 398   Inf 0.0005236558  0.0276594111      241.5953      58283.71
## 10 3.00 224 0.300 0.0038498315  0.3425589285      155.6283      24929.27
```

### 7.3 Save Predictions

```r
write_csv(test_predictions, "predictions.csv")
```

---

## 8. Model Summaries

### 8.1 Mean Model

```r
cat("Mean model summary")
```

```
## Mean model summary
```

```r
summary(lm_mean_best)
```

```
##
## Call:
## lm(formula = formula_mean_best, data = train)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.28651 -0.07836  0.00584  0.07205  0.31902
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -1.78836    0.07237 -24.711  < 2e-16 ***
## log_St             0.19377    0.01194  16.229  < 2e-16 ***
## Re224             -4.09451    0.09732 -42.075  < 2e-16 ***
## Re398             -6.32588    0.11428 -55.356  < 2e-16 ***
## Fr_invlogit       -0.47301    0.10272  -4.605 1.49e-05 ***
## Re224:Fr_invlogit  0.66495    0.13706   4.852 5.75e-06 ***
## Re398:Fr_invlogit  0.81969    0.15254   5.373 7.09e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1198 on 82 degrees of freedom
## Multiple R-squared:  0.9973, Adjusted R-squared:  0.9971
## F-statistic:  5111 on 6 and 82 DF,  p-value: < 2.2e-16
```

### 8.2 Variance Model

```r
cat("Variance model summary")
```

```
## Variance model summary
```

```r
summary(lm_var_best)
```

```
##
## Call:
## lm(formula = formula_var_best, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6260 -0.7448  0.1866  0.9427  3.6603
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         8.7591     1.1422   7.669 3.15e-11 ***
## log_St              0.8551     0.1884   4.538 1.92e-05 ***
## Re224              -9.2642     1.5358  -6.032 4.47e-08 ***
## Re398             -14.3253     1.8035  -7.943 9.06e-12 ***
## Fr_invlogit        -9.3445     1.6211  -5.764 1.40e-07 ***
## Re224:Fr_invlogit   6.7234     2.1630   3.108  0.00259 **
## Re398:Fr_invlogit  10.3834     2.4075   4.313 4.46e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.89 on 82 degrees of freedom
## Multiple R-squared:  0.7584, Adjusted R-squared:  0.7407
## F-statistic:  42.9 on 6 and 82 DF,  p-value: < 2.2e-16
```

## 8.3 Skewness Model

```r
cat("Skewness Model summary")
```

```
## Skewness Model summary
```

```r
summary(lm_skew_best)
```

```
##
## Call:
## lm(formula = formula_skew_best, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.59484 -0.34593  0.05311  0.33817  1.52756
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           6.0139     0.2686  22.393  < 2e-16 ***
## Fr_invlogit          -3.3103     0.4114  -8.046 4.58e-12 ***
## Fr_invlogit:Re224     1.6279     0.2577   6.318 1.17e-08 ***
## Fr_invlogit:Re398     2.7837     0.2975   9.357 1.02e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7605 on 85 degrees of freedom
## Multiple R-squared:  0.5632, Adjusted R-squared:  0.5477
## F-statistic: 36.53 on 3 and 85 DF,  p-value: 2.907e-15
```

## 8.4 Kurtosis Model

```r
cat("Kurtosis model summary")
```

```
## Kurtosis model summary
```

```r
summary(lm_kurt_best)
```

```
##
## Call:
## lm(formula = formula_kurt_best, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2194 -0.7180  0.1026  0.6495  2.9882
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          12.0396     0.5280  22.801  < 2e-16 ***
```

```
## Fr_invlogit        -6.5010     0.8089  -8.037 4.78e-12 ***
## Fr_invlogit:Re224   3.1660     0.5066   6.249 1.58e-08 ***
## Fr_invlogit:Re398   5.4345     0.5849   9.291 1.39e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.495 on 85 degrees of freedom
## Multiple R-squared:  0.5606, Adjusted R-squared:  0.5451
## F-statistic: 36.15 on 3 and 85 DF,  p-value: 3.723e-15
```

## 8.5 Cross-Validation RMSE Comparison

To compare predictive performance across all four models, we summarize the 10-fold CV RMSE below:

```
cv_results <- data.frame(
  Response = c("Mean", "Variance", "Skewness", "Kurtosis"),
  OLS_CV_RMSE = c(cv_ols_mean, cv_ols_var, cv_ols_skew, cv_ols_kurt),
  Ridge_CV_RMSE = c(cv_ridge_mean, cv_ridge_var, cv_ridge_skew, cv_ridge_kurt)
)

cv_results %>%
  mutate(
    Best_Method = ifelse(OLS_CV_RMSE < Ridge_CV_RMSE, "OLS", "Ridge"),
    Best_RMSE = pmin(OLS_CV_RMSE, Ridge_CV_RMSE)
  ) %>%
  arrange(Best_RMSE)
```

```
##   Response OLS_CV_RMSE Ridge_CV_RMSE Best_Method Best_RMSE
## 1     Mean   0.1247262     0.3657629         OLS 0.1247262
## 2 Skewness   0.7629519     0.7636996         OLS 0.7629519
## 3 Kurtosis   1.5374593     1.5407637         OLS 1.5374593
## 4 Variance   1.9254890     2.1207252         OLS 1.9254890
```

---

# 9. Key Findings

**Model Performance**: - All models use log transformations to handle right skew - Best subset selection reduced variable count from full model - OLS vs Ridge comparison determined final estimation method

**What I think we should do next** 1. Examine coefficient signs and magnitudes 2. Identify which predictors are most important for each response 3. Look for consistent patterns across all 4 models 4. Interpret interaction effects in physical terms 5. Compare CV RMSE across models to see which responses are easiest/hardest to predict