Prediction

```r
library(tidyverse)
library(readr)
library(vip)
library(tidymodels)
library(mgcv)
library(rsample)
library(yardstick)
library(broom)
library(gratia)
library(GGally)
library(patchwork)
```

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

```
## Rows: 89 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (7): St, Re, Fr, R_moment_1, R_moment_2, R_moment_3, R_moment_4
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
train <- train %>%
  mutate(
    mu    = R_moment_1,
    sigma = sqrt(pmax(R_moment_2 - R_moment_1^2, 1e-10)),
    skew  = (R_moment_3 - 3 * R_moment_2 * R_moment_1 + 2 * R_moment_1^3) / sigma^3,
    kurt  = (R_moment_4 - 4 * R_moment_3 * R_moment_1 + 6 * R_moment_2 * R_moment_1^2 - 3 * R_moment_1^4
  ) %>%
  filter(
    is.finite(mu), is.finite(sigma), is.finite(skew), is.finite(kurt)
  ) %>%
  drop_na(mu, sigma, skew, kurt, Re, Fr, St)
```

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

train <- train %>%
  mutate(
    # Derived target variables
    mu    = R_moment_1,
    sigma = sqrt(pmax(R_moment_2 - R_moment_1^2, 1e-10)),   # avoid sigma = 0
    skew  = (R_moment_3 - 3 * R_moment_2 * R_moment_1 + 2 * R_moment_1^3) / sigma^3,
    kurt  = (R_moment_4 - 4 * R_moment_3 * R_moment_1 + 6 * R_moment_2 * R_moment_1^2 - 3 * R_moment_1^4
  ) %>%
  mutate(
    log_sigma = log(pmax(sigma, 1e-10)),
    log_skew  = log(pmax(skew, 1e-10)),
    log_kurt  = log(pmax(kurt, 1e-10)),
```

```
    # Safe log-transform for Re
    logRe = log(pmax(Re, 1e-8)),

    #  Clamp Fr and St to (0, 1) before logit transform
    Fr = pmin(pmax(Fr, 1e-6), 1 - 1e-6),
    St = pmin(pmax(St, 1e-6), 1 - 1e-6),

    logitFr = log(Fr / (1 - Fr)),
    logitSt = log(St / (1 - St))
  ) %>%
  # Drop bad rows and ensure all numeric columns are finite
  filter(
    is.finite(mu), is.finite(sigma), is.finite(skew), is.finite(kurt),
    is.finite(logRe), is.finite(logitFr), is.finite(logitSt)
  ) %>%
  drop_na(mu, sigma, skew, kurt, logRe, logitFr, logitSt)
```

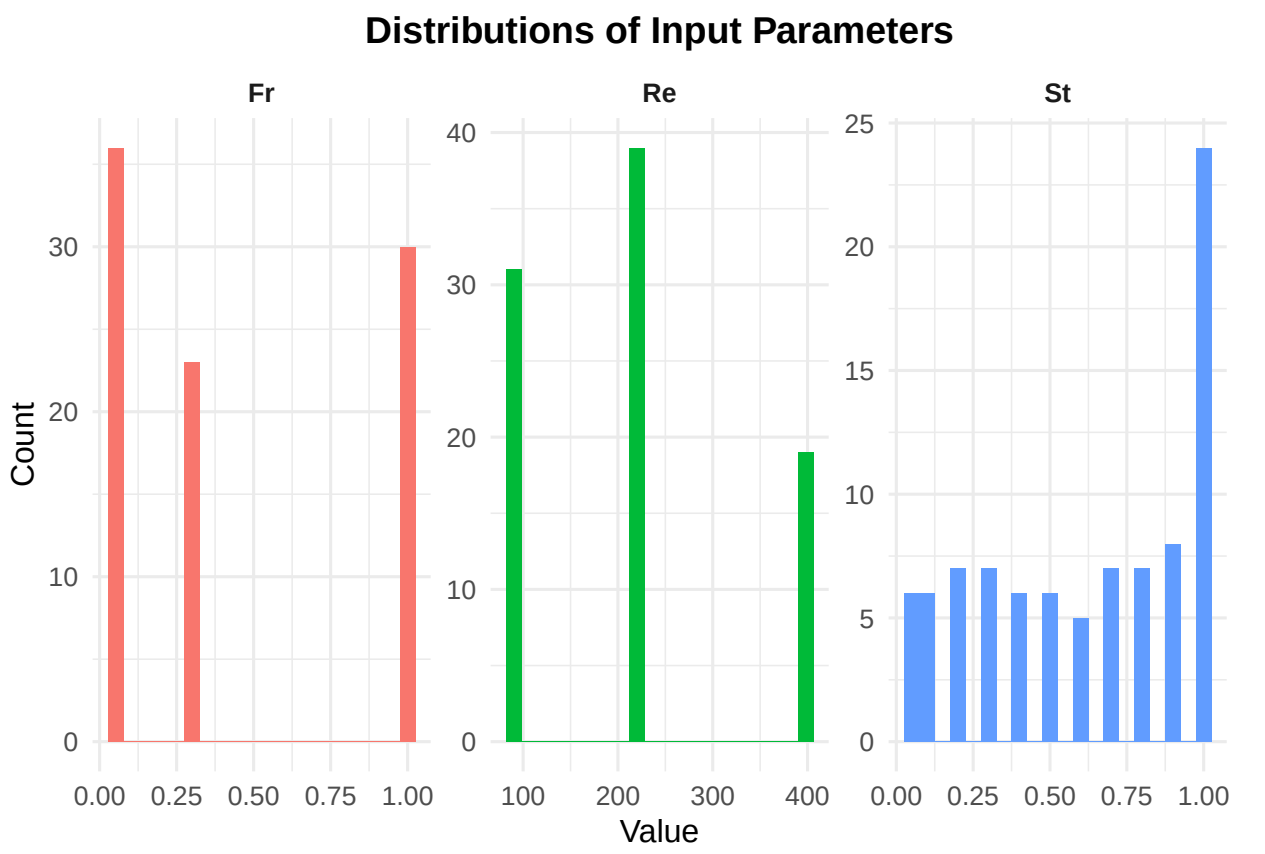
EDA

```
# Summary of predictors and responses
train %>%
  select(Re, Fr, St, mu, sigma, skew, kurt) %>%
  summary()
```

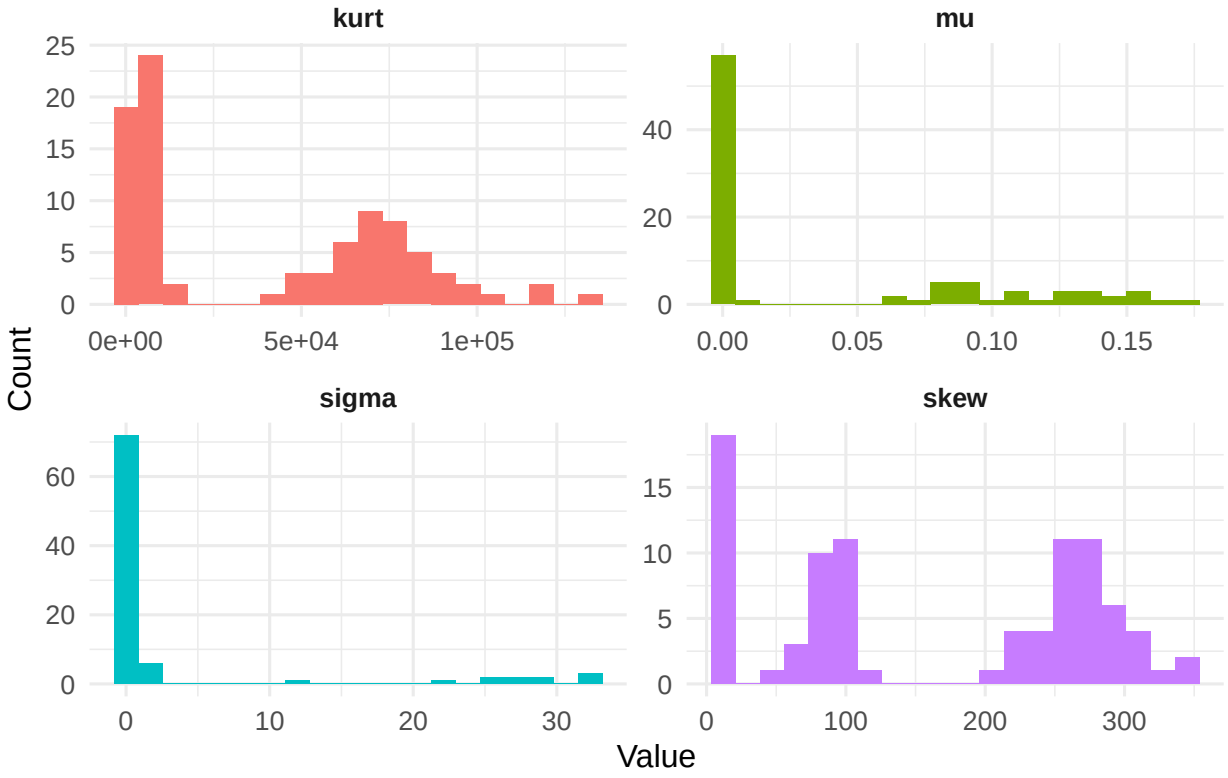
```
##        Re              Fr               St               mu
##  Min.   : 90.0   Min.   :0.0520   Min.   :0.0500   Min.   :0.000222
##  1st Qu.: 90.0   1st Qu.:0.0520   1st Qu.:0.3000   1st Qu.:0.002157
##  Median :224.0   Median :0.3000   Median :0.7000   Median :0.002958
##  Mean   :214.5   Mean   :0.4356   Mean   :0.6124   Mean   :0.040394
##  3rd Qu.:224.0   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.087868
##  Max.   :398.0   Max.   :1.0000   Max.   :1.0000   Max.   :0.172340
##      sigma               skew             kurt
##  Min.   : 0.01006   Min.   : 11.97   Min.   :   150.5
##  1st Qu.: 0.15643   1st Qu.: 72.55   1st Qu.:  5622.3
##  Median : 0.28423   Median :110.12   Median : 12158.7
##  Mean   : 3.65112   Mean   :162.81   Mean   : 39749.6
##  3rd Qu.: 0.72579   3rd Qu.:269.54   3rd Qu.: 72732.4
##  Max.   :32.31526   Max.   :344.91   Max.   :132136.7
```

```
#input histograms
p_inputs <- train %>%
  select(Re, Fr, St) %>%
  pivot_longer(everything(), names_to = "Variable", values_to = "Value") %>%
  ggplot(aes(x = Value, fill = Variable)) +
  geom_histogram(bins = 20) +
  facet_wrap(~ Variable, scales = "free", nrow = 1) +
  labs(
    title = "Distributions of Input Parameters",
    x = "Value",
    y = "Count"
  ) +
  theme_minimal(base_size = 12) +
  theme(
```

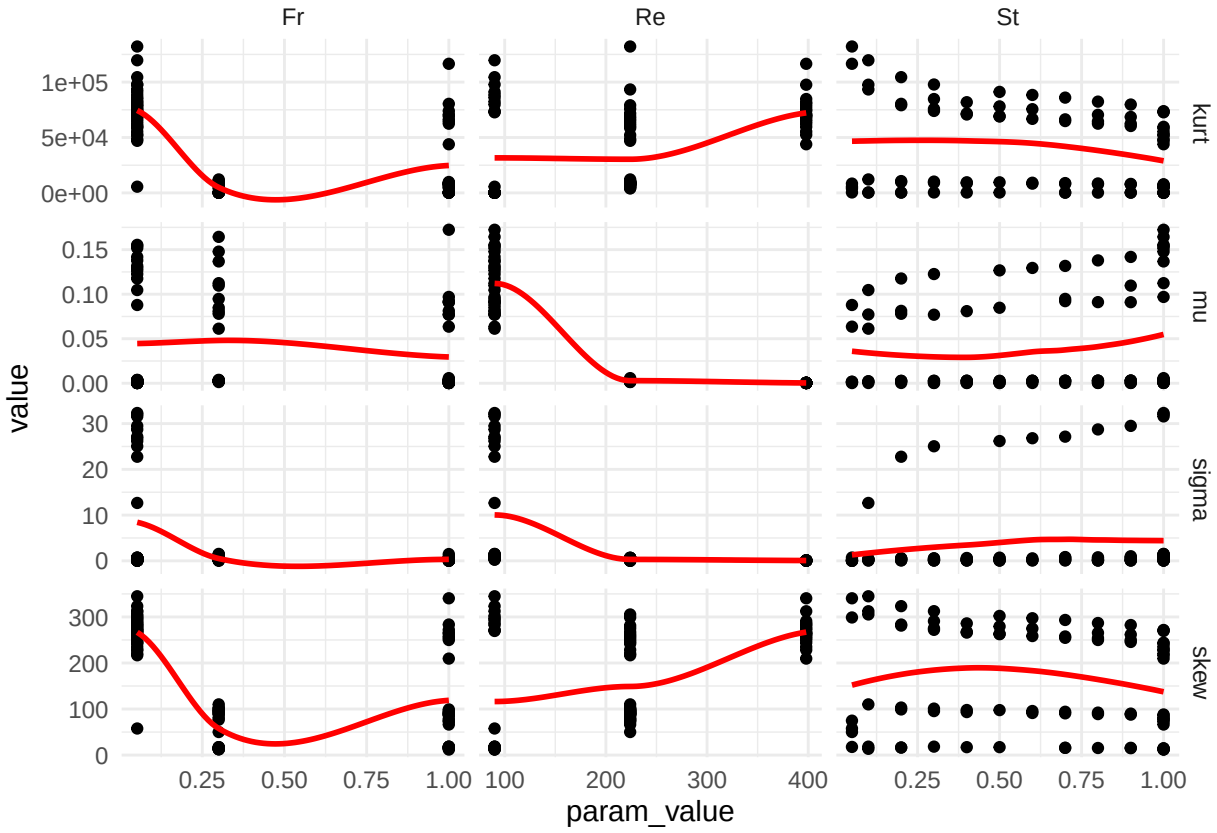
```
    legend.position = "none",
    plot.title = element_text(face = "bold", hjust = 0.5),
    strip.text = element_text(face = "bold")
  )

p_inputs
```



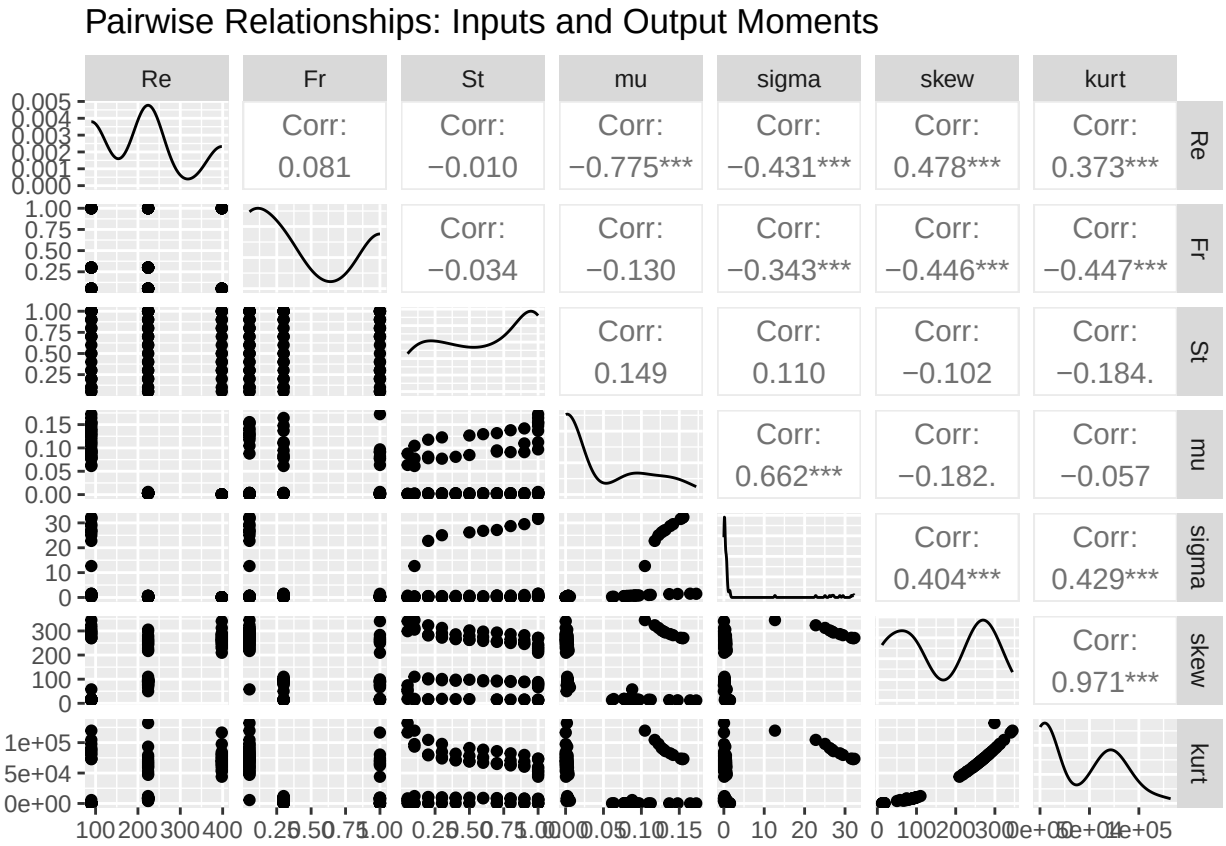

```
#target histograms
p_targets <- train %>%
  select(mu, sigma, skew, kurt) %>%
  pivot_longer(everything(), names_to = "Statistic", values_to = "Value") %>%
  ggplot(aes(x = Value, fill = Statistic)) +
  geom_histogram(bins = 20) +
  facet_wrap(~ Statistic, scales = "free", nrow = 2) +
  labs(
    title = "Distribution of Target Statistics:  ,  ,  ,  ",
    x = "Value",
    y = "Count"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    legend.position = "none",
    plot.title = element_text(face = "bold", hjust = 0.5),
    strip.text = element_text(face = "bold")
  )

p_targets
```



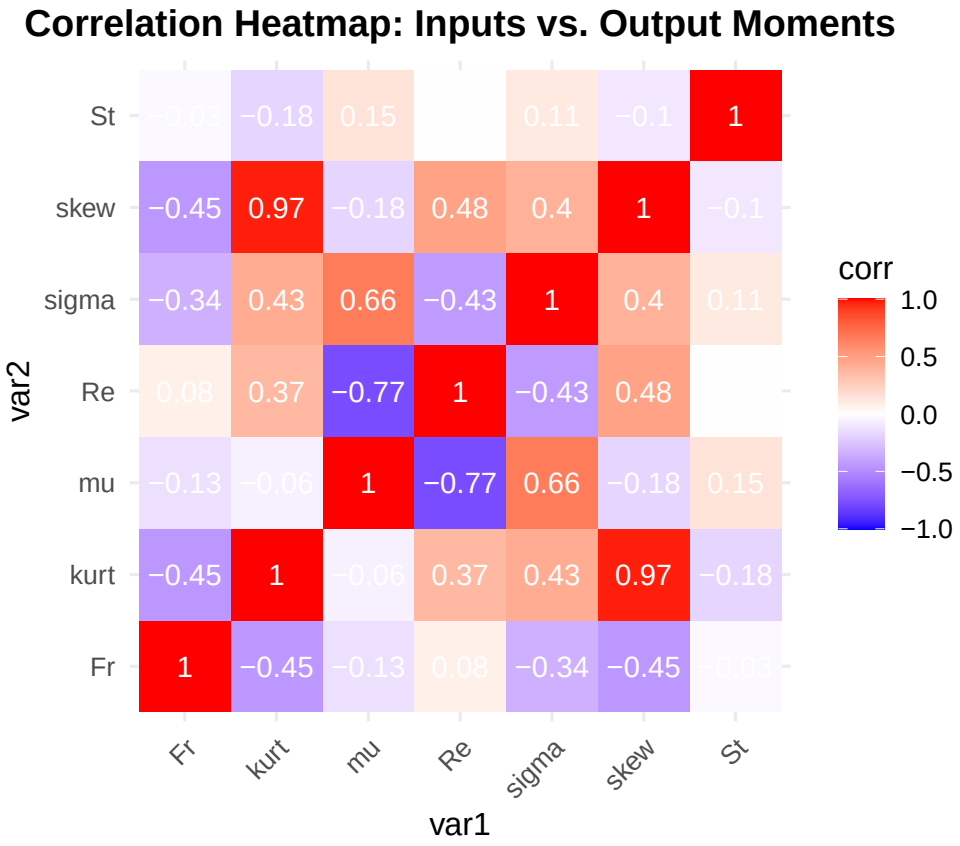

```
#scatterplots
train %>%
  select(Re, Fr, St, mu, sigma, skew, kurt) %>%
  pivot_longer(mu:kurt, names_to="moment", values_to="value") %>%
  pivot_longer(Re:St, names_to="param", values_to="param_value") %>%
  ggplot(aes(param_value, value)) +
  geom_point() +
  geom_smooth(se=FALSE, color="red") +
  facet_grid(moment ~ param, scales="free") +
  theme_minimal()
```




```
options(ggally.progress = FALSE)

# Pairplot of all predictors and target moments
train %>%
  select(Re, Fr, St, mu, sigma, skew, kurt) %>%
  ggpairs(title = "Pairwise Relationships: Inputs and Output Moments")
```




```r
cors <- train %>%
  select(Re, Fr, St, mu, sigma, skew, kurt) %>%
  cor(use = "complete.obs") %>%
  round(2)

p_heat <- as_tibble(cors, rownames="var1") %>%
  pivot_longer(-var1, names_to="var2", values_to="corr") %>%
  ggplot(aes(var1, var2, fill=corr)) +
  geom_tile() +
  geom_text(aes(label=corr), color="white", size=4) +
  scale_fill_gradient2(limits=c(-1,1), low="blue", mid="white", high="red") +
  coord_equal() +
  labs(title="Correlation Heatmap: Inputs vs. Output Moments") +
  theme_minimal(base_size=12) +
  theme(
    plot.title = element_text(face="bold", hjust=0.5),
    axis.text.x = element_text(angle=45, hjust=1)
  )

p_heat
```



GAM

```
sapply(train[, c("Re", "Fr", "St")], function(x) length(unique(x)))
```

```
## Re Fr St
##  3  3 11
```

```
train_gam <- train %>%
  mutate(
    logRe = log(pmax(Re, 1e-8)),

    Fr = pmin(pmax(Fr, 1e-6), 1 - 1e-6),
    St = pmin(pmax(St, 1e-6), 1 - 1e-6),

    logitFr = log(Fr / (1 - Fr)),
    logitSt = log(St / (1 - St))
  ) %>%
  filter(
    is.finite(mu), is.finite(sigma), is.finite(skew), is.finite(kurt),
    is.finite(logRe), is.finite(logitFr), is.finite(logitSt)
  )
```

```
#for reproducibability
set.seed(1)

folds <- vfold_cv(train_gam, v = 10)
```

```r
cv_metrics <- function(formula, data, fit_fun) {
  map_dfr(folds$splits, function(split) {
    tr <- analysis(split)
    va <- assessment(split)

    fit <- fit_fun(formula, data = tr)

    preds <- predict(fit, newdata = va)
    truth <- va[[all.vars(formula)[1]]]

    tibble(
      RMSE = rmse_vec(truth = truth, estimate = preds),
      MAE  = mae_vec(truth = truth, estimate = preds),
      R2   = rsq_vec(truth = truth, estimate = preds)
    )
  }) %>%
    summarize(across(everything(), mean))
}

# Define model formulas
forms <- list(
  mu    = mu    ~ factor(Re) + logitFr + logitSt,
  sigma = sigma ~ factor(Re) + logitFr + logitSt,
  skew  = skew  ~ factor(Re) + logitFr + logitSt,
  kurt  = kurt  ~ factor(Re) + logitFr + logitSt
)

gam_forms <- list(
  mu = mu ~ factor(Re) + s(logitFr, k=3) + s(logitSt, k=5),
  sigma = sigma ~ factor(Re) + s(logitFr, k=3) + s(logitSt, k=5),
  skew = skew ~ factor(Re) + s(logitFr, k=3) + s(logitSt, k=5),
  kurt = kurt ~ factor(Re) + s(logitFr, k=3) + s(logitSt, k=5)
)

# Compute CV metrics for both LM and GAM
cv_results <- map2_dfr(names(forms), forms, function(name, fml) {
  lm_metrics <- cv_metrics(fml, train_gam, lm)
  gam_metrics <- cv_metrics(gam_forms[[name]], train_gam, function(formula, data)
    gam(formula, data = data, method = "REML"))

  tibble(
    Target = name,
    RMSE_Linear = lm_metrics$RMSE,
    RMSE_GAM = gam_metrics$RMSE,
    MAE_Linear = lm_metrics$MAE,
    MAE_GAM = gam_metrics$MAE,
    R2_Linear = lm_metrics$R2,
    R2_GAM = gam_metrics$R2
  )
})

cv_results <- cv_results %>%
  mutate(across(-Target, round, 3))
```

```r
print(cv_results)
```

```
## # A tibble: 4 x 7
##   Target RMSE_Linear  RMSE_GAM MAE_Linear   MAE_GAM R2_Linear R2_GAM
##   <chr>        <dbl>     <dbl>      <dbl>     <dbl>     <dbl>  <dbl>
## 1 mu           0.017     0.017      0.013     0.013     0.843  0.841
## 2 sigma        7.46      6.73       5.61      5.83      0.548  0.589
## 3 skew        87.7      55.1       75.6      44.8       0.377  0.729
## 4 kurt     29858.    19804.     25328.    15520.        0.348  0.71
```

```r
# Fit a GAM for each target
gam_mu <- gam(mu ~ factor(Re) + s(logitFr, k=3) + s(logitSt, k=5), data=train_gam, method="REML")
gam_sigma <- gam(sigma ~ factor(Re) + s(logitFr, k=3) + s(logitSt, k=5), data=train_gam, method="REML")
gam_skew <- gam(skew~ factor(Re) + s(logitFr, k=3) + s(logitSt, k=5), data=train_gam, method="REML")
gam_kurt <- gam(kurt~ factor(Re) + s(logitFr, k=3) + s(logitSt, k=5), data=train_gam, method="REML")

summary(gam_mu)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## mu ~ factor(Re) + s(logitFr, k = 3) + s(logitSt, k = 5)
##
## Parametric coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.111838   0.002895   38.62   <2e-16 ***
## factor(Re)224 -0.108926   0.003865  -28.18   <2e-16 ***
## factor(Re)398 -0.111076   0.004825  -23.02   <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(logitFr) 1.670  1.891  2.785   0.0432 *
## s(logitSt) 1.629  1.921 13.314 5.72e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.917   Deviance explained = 92.2%
## -REML = -217.35  Scale est. = 0.00025782  n = 89
```

```r
summary(gam_sigma)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## sigma ~ factor(Re) + s(logitFr, k = 3) + s(logitSt, k = 5)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     10.285      1.171   8.783 1.86e-13 ***
```

```
## factor(Re)224   -9.511      1.562  -6.089 3.45e-08 ***
## factor(Re)398  -11.552      1.967  -5.873 8.69e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df      F  p-value
## s(logitFr) 1.959  1.998 17.353 6.23e-07 ***
## s(logitSt) 1.556  1.844  0.997     0.46
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.474   Deviance explained = 50.7%
## -REML = 287.76  Scale est. = 42.103    n = 89
```

```
summary(gam_skew)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## skew ~ factor(Re) + s(logitFr, k = 3) + s(logitSt, k = 5)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    121.292      9.933  12.211  < 2e-16 ***
## factor(Re)224   37.114     13.252   2.801  0.00634 **
## factor(Re)398  118.308     16.684   7.091 4.06e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df      F p-value
## s(logitFr) 1.992  2.000 94.542  <2e-16 ***
## s(logitSt) 1.000  1.001  5.771  0.0185 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.763   Deviance explained = 77.7%
## -REML =  467.8  Scale est. = 3030.9    n = 89
```

```
summary(gam_kurt)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## kurt ~ factor(Re) + s(logitFr, k = 3) + s(logitSt, k = 5)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    33380.0     3618.1   9.226 2.31e-14 ***
## factor(Re)224    204.1     4826.9   0.042    0.966
```

```
## factor(Re)398  29417.3     6077.0   4.841 5.90e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df     F p-value
## s(logitFr) 1.99  2.000 80.10 < 2e-16 ***
## s(logitSt) 1.00  1.001 10.46 0.00175 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.723   Deviance explained = 73.9%
## -REML = 963.13  Scale est. = 4.0214e+08  n = 89
```

```
#linear models
lm_mu    <- lm(mu~ factor(Re) + logitFr + logitSt, data = train_gam)
lm_sigma <- lm(sigma~ factor(Re) + logitFr + logitSt, data = train_gam)
lm_skew  <- lm(skew~ factor(Re) + logitFr + logitSt, data = train_gam)
lm_kurt  <- lm(kurt~ factor(Re) + logitFr + logitSt, data = train_gam)

summary(lm_mu)
```

```
##
## Call:
## lm(formula = mu ~ factor(Re) + logitFr + logitSt, data = train_gam)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.045089 -0.013328  0.001382  0.009175  0.053417
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.1086631  0.0031897  34.066  < 2e-16 ***
## factor(Re)224 -0.1089322  0.0039440 -27.619  < 2e-16 ***
## factor(Re)398 -0.1093540  0.0048124 -22.723  < 2e-16 ***
## logitFr       -0.0005112  0.0002315  -2.208     0.03 *
## logitSt        0.0012538  0.0002738   4.580  1.6e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01639 on 84 degrees of freedom
## Multiple R-squared:  0.9178, Adjusted R-squared:  0.9139
## F-statistic: 234.5 on 4 and 84 DF,  p-value: < 2.2e-16
```

```
summary(lm_sigma)
```

```
##
## Call:
## lm(formula = sigma ~ factor(Re) + logitFr + logitSt, data = train_gam)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.798  -3.294  -1.341   3.013  19.807
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   10.60021    1.43512   7.386 1.01e-10 ***
## factor(Re)224 -9.61688    1.77449  -5.420 5.61e-07 ***
## factor(Re)398 -9.13594    2.16517  -4.219 6.16e-05 ***
## logitFr       -0.31831    0.10416  -3.056  0.00301 **
## logitSt        0.07119    0.12317   0.578  0.56482
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.373 on 84 degrees of freedom
## Multiple R-squared:  0.3518, Adjusted R-squared:  0.3209
## F-statistic:  11.4 on 4 and 84 DF,  p-value: 1.953e-07
```

```
summary(lm_skew)
```

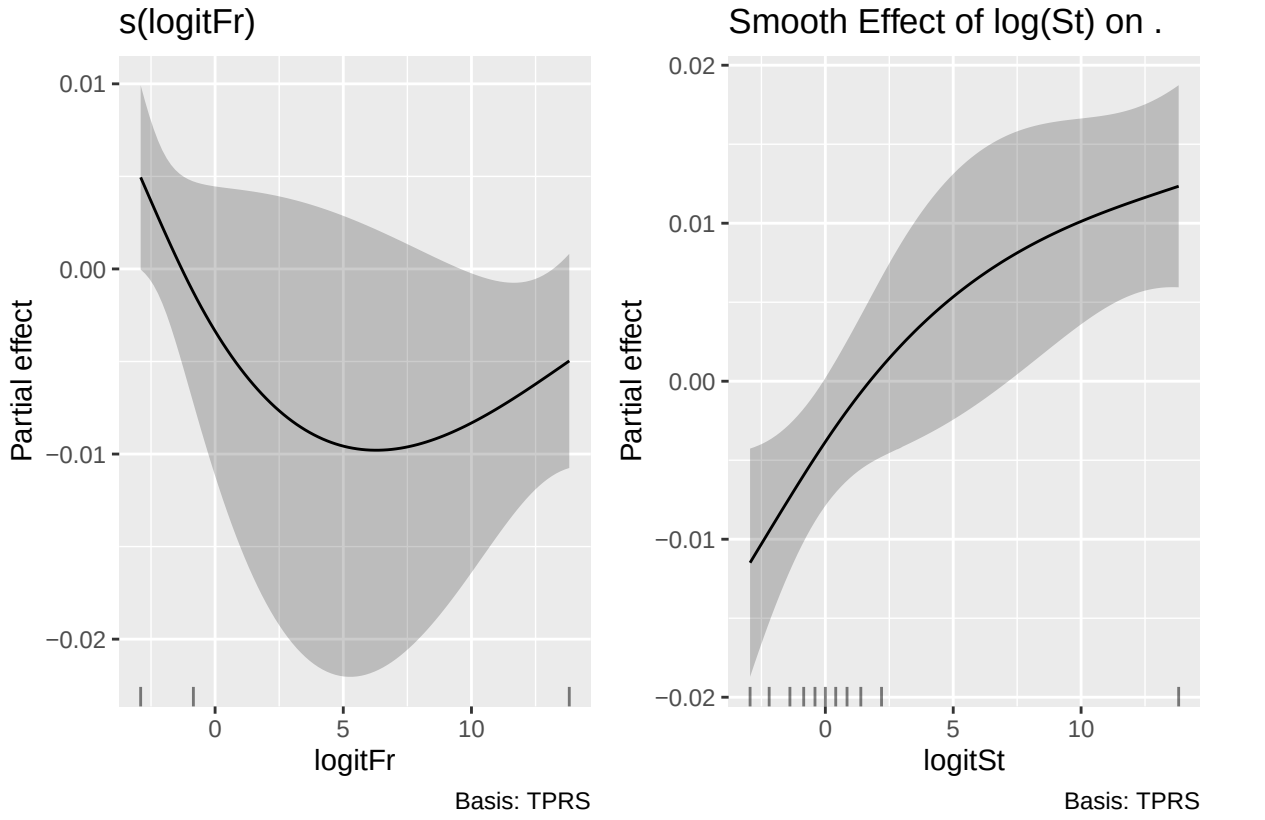
```
##
## Call:
## lm(formula = skew ~ factor(Re) + logitFr + logitSt, data = train_gam)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -138.87  -68.24    4.05   61.64  179.32
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    141.897     16.804   8.444 7.79e-13 ***
## factor(Re)224   34.520     20.777   1.661    0.100
## factor(Re)398  163.965     25.352   6.468 6.26e-09 ***
## logitFr         -6.387      1.220  -5.237 1.19e-06 ***
## logitSt         -2.343      1.442  -1.625    0.108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 86.33 on 84 degrees of freedom
## Multiple R-squared:  0.4438, Adjusted R-squared:  0.4173
## F-statistic: 16.76 on 4 and 84 DF,  p-value: 3.907e-10
```

```
summary(lm_kurt)
```

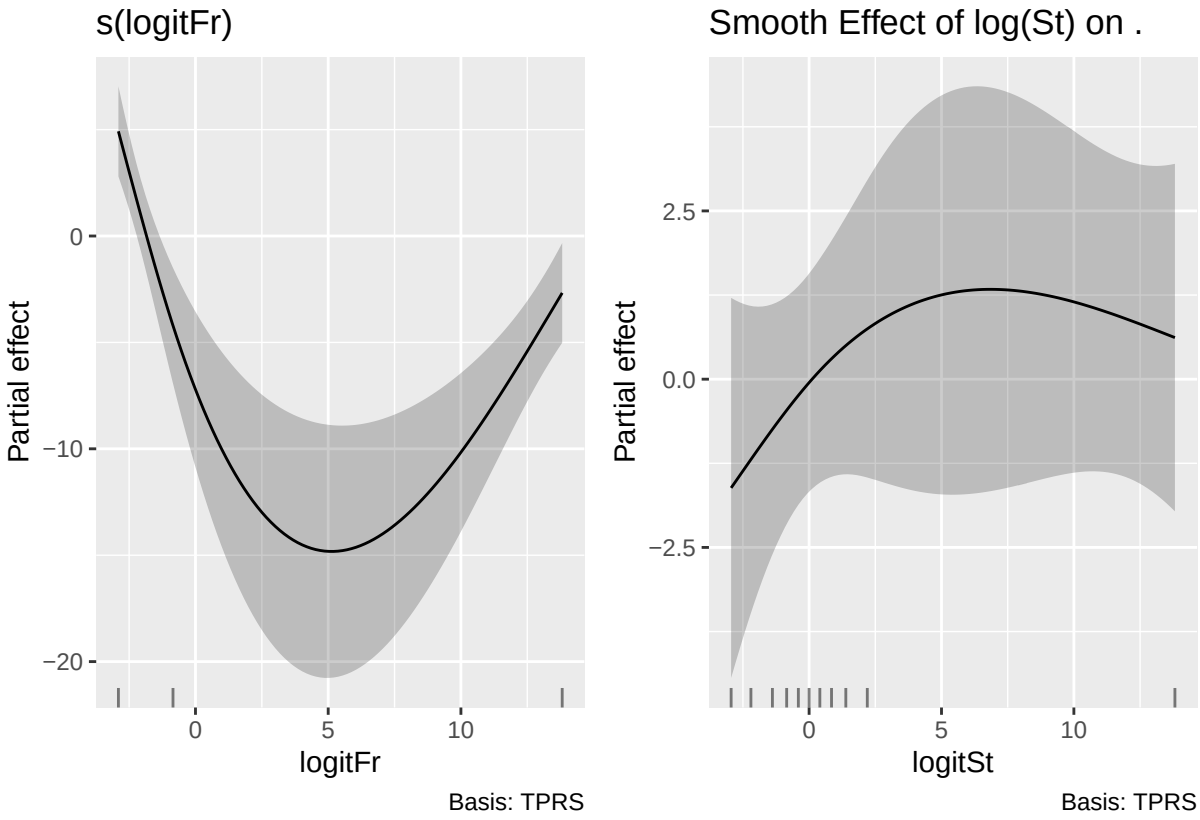
```
##
## Call:
## lm(formula = kurt ~ factor(Re) + logitFr + logitSt, data = train_gam)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -45567 -21604  -2422  18728  81739
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    41519.1     5837.5   7.112 3.50e-10 ***
## factor(Re)224   -665.1     7218.0  -0.092   0.9268
## factor(Re)398  44717.5     8807.1   5.077 2.27e-06 ***
## logitFr        -2143.3      423.7  -5.058 2.45e-06 ***
## logitSt        -1128.2      501.0  -2.252   0.0269 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29990 on 84 degrees of freedom
## Multiple R-squared:  0.4085, Adjusted R-squared:  0.3803
## F-statistic:  14.5 on 4 and 84 DF,  p-value: 4.795e-09
```

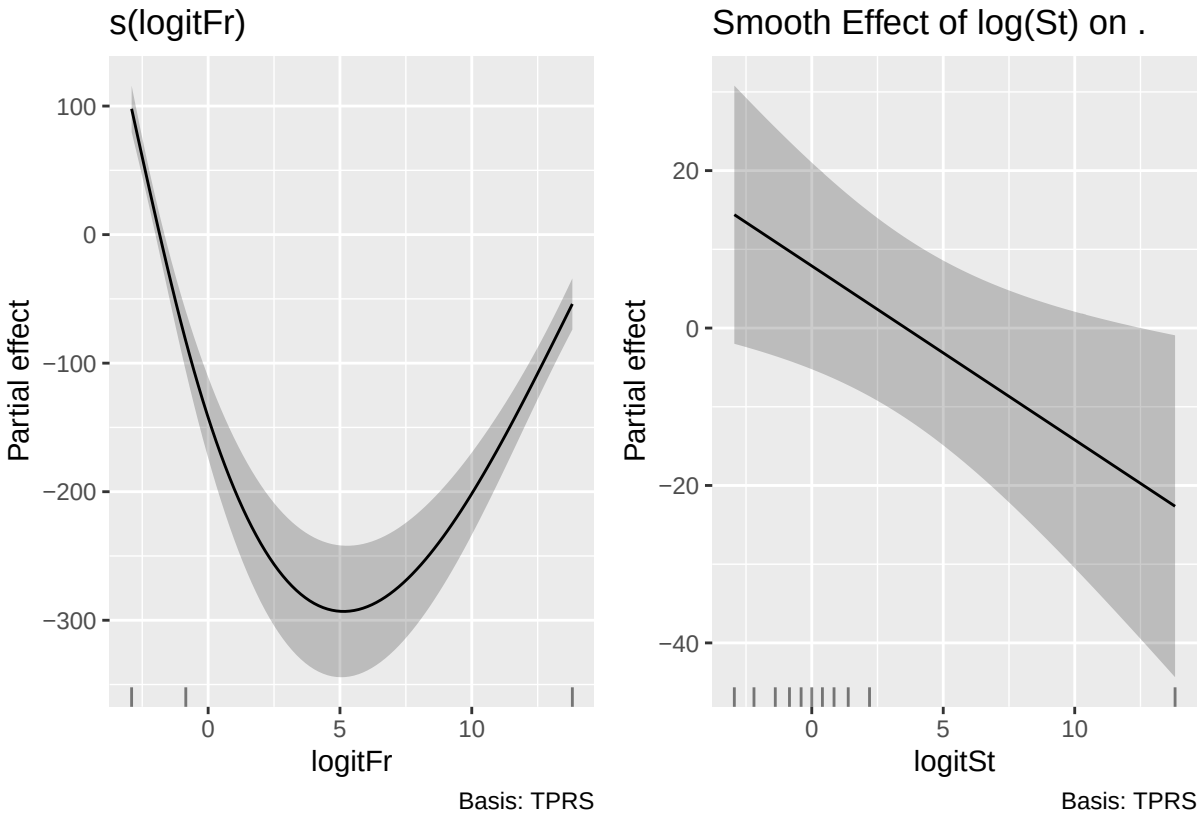
```
#plot smooth plots
draw(gam_mu)    + ggtitle("Smooth Effect of log(St) on  ")
```



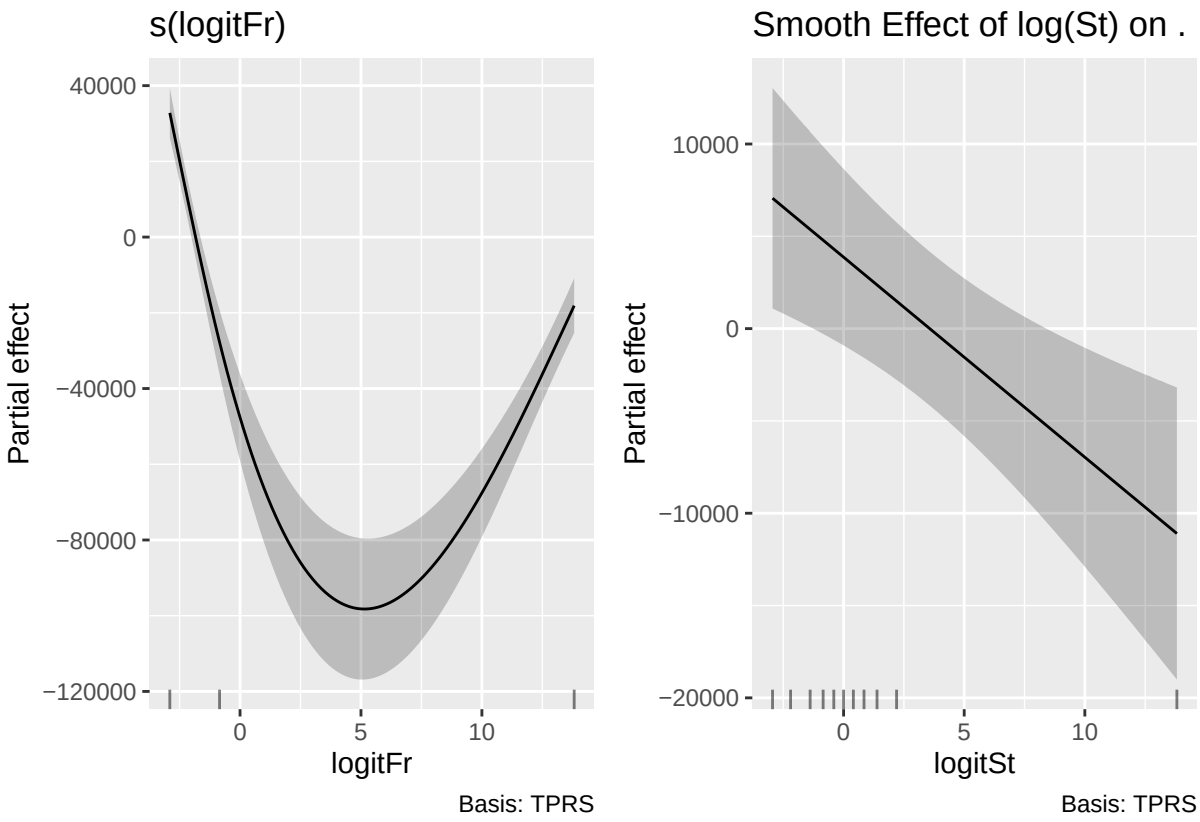

```
draw(gam_sigma) + ggtitle("Smooth Effect of log(St) on  ")
```



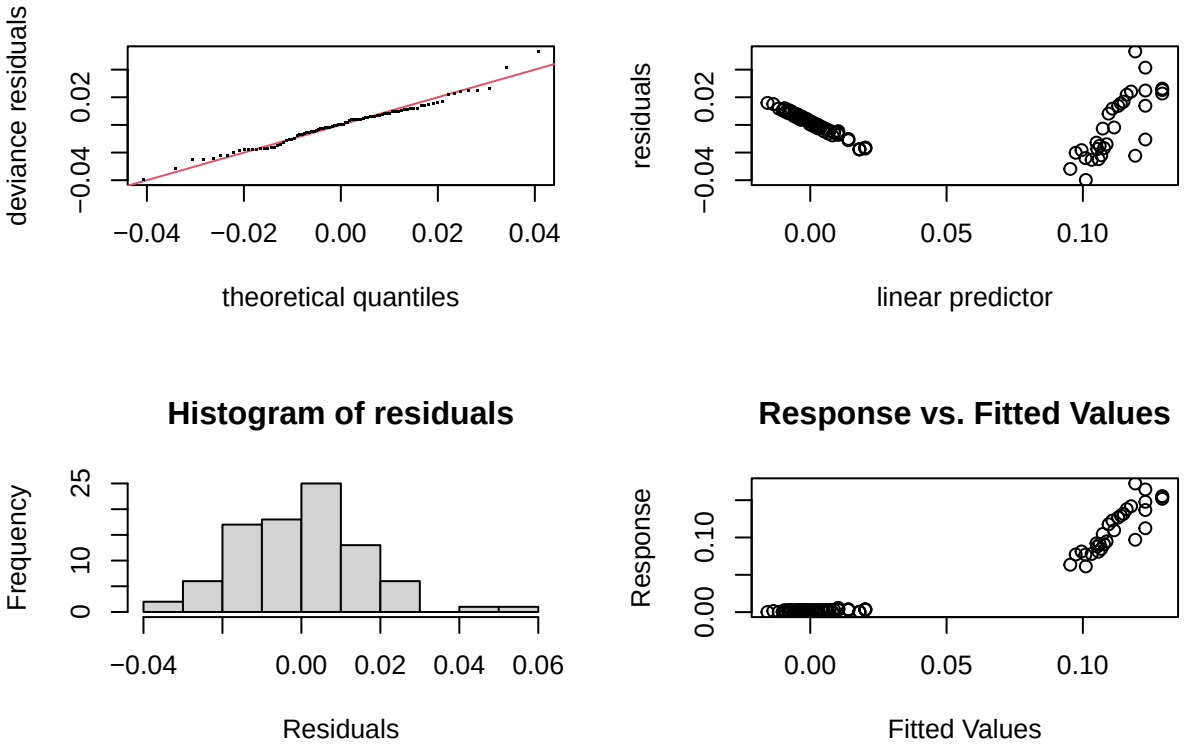

```
draw(gam_skew)  + ggtitle("Smooth Effect of log(St) on  ")
```



```
draw(gam_kurt)  + ggtitle("Smooth Effect of log(St) on  ")
```

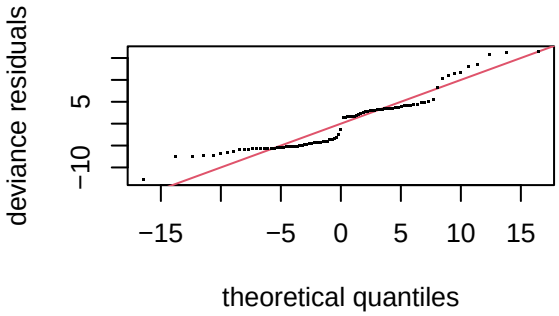


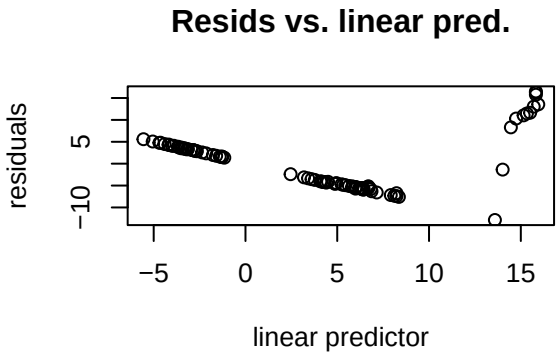

```
# Residual plots
par(mfrow=c(2,2))
gam.check(gam_mu)
```

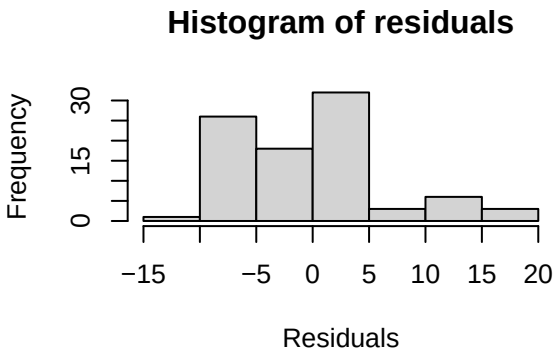


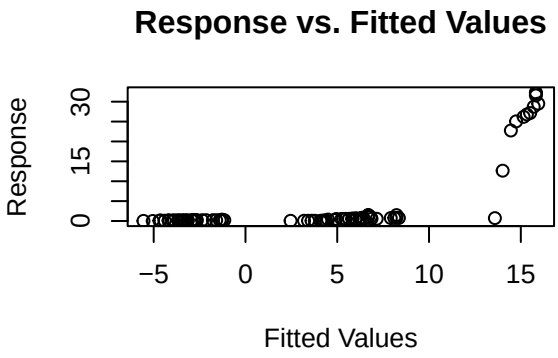

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 6 iterations.
## Gradient range [-5.990245e-07,-2.530419e-09]
## (score -217.3541 & scale 0.0002578228).
## Hessian positive definite, eigenvalue range [0.1888805,42.00505].
## Model rank =  9 / 9
##
## Basis dimension (k) checking 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(logitFr) 2.00 1.67    1.19    0.96
## s(logitSt) 4.00 1.63    1.12    0.91
```

```
gam.check(gam_sigma)
```



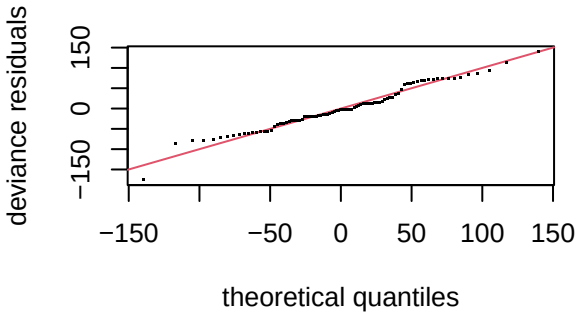


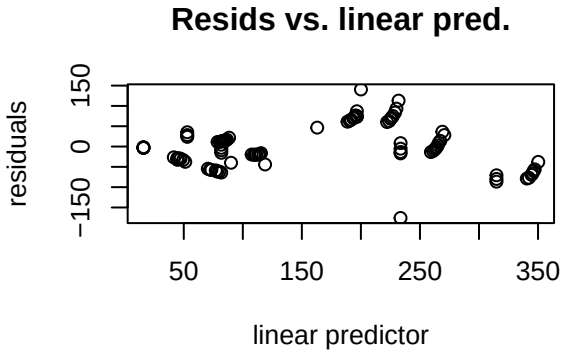


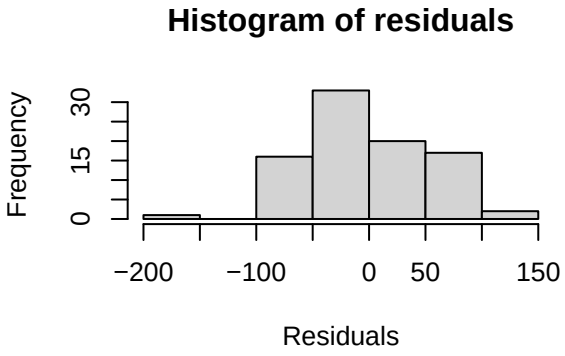


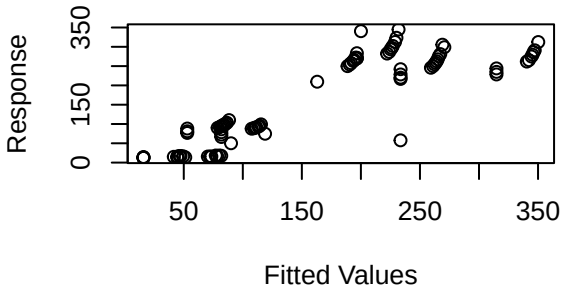

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 9 iterations.
## Gradient range [-2.732822e-06,2.826938e-07]
## (score 287.7556 & scale 42.10276).
## Hessian positive definite, eigenvalue range [0.1416346,42.00738].
## Model rank =  9 / 9
##
## Basis dimension (k) checking 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(logitFr) 2.00 1.96    1.05    0.69
## s(logitSt) 4.00 1.56    1.14    0.89
```

```
gam.check(gam_skew)
```



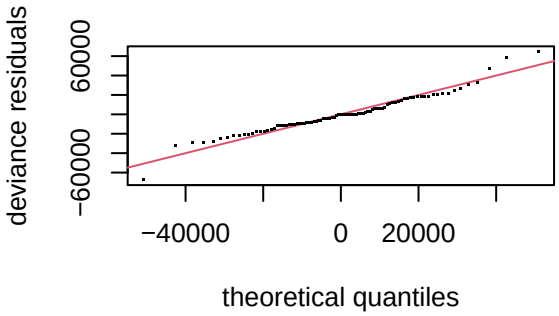


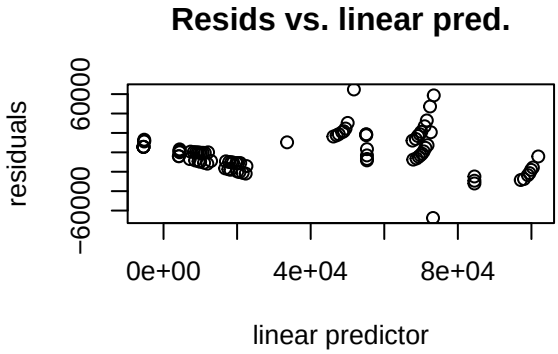


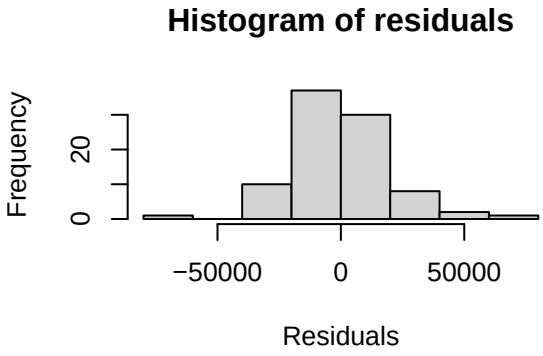


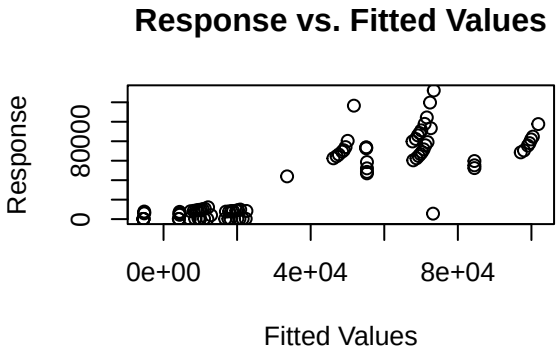

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 8 iterations.
## Gradient range [-0.0001741637,1.450729e-05]
## (score 467.7968 & scale 3030.928).
## Hessian positive definite, eigenvalue range [0.0001741025,42.00591].
## Model rank =  9 / 9
##
## Basis dimension (k) checking 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(logitFr) 2.00 1.99    1.03    0.57
## s(logitSt) 4.00 1.00    1.10    0.81
```

```
gam.check(gam_kurt)
```










```
##
## Method: REML   Optimizer: outer newton
## full convergence after 8 iterations.
## Gradient range [-6.304553e-05,6.120441e-05]
## (score 963.1334 & scale 402135257).
## eigenvalue range [-3.894008e-05,42.00597].
## Model rank =  9 / 9
##
## 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(logitFr) 2.00 1.99    0.99    0.37
## s(logitSt) 4.00 1.00    1.08    0.74
```

```
par(mfrow=c(1,1))
```

Results

```
test <- read_csv("data-test.csv") %>%
  mutate(
    logRe = log(pmax(Re, 1e-8)),
    Fr = pmin(pmax(Fr, 1e-6), 1 - 1e-6),
    St = pmin(pmax(St, 1e-6), 1 - 1e-6),
    logitFr = log(Fr / (1 - Fr)),
    logitSt = log(St / (1 - St))
  )

pred_mu    <- predict(gam_mu,    newdata=test, se.fit=TRUE)
```

```r
pred_sigma <- predict(gam_sigma, newdata=test, se.fit=TRUE)
pred_skew  <- predict(gam_skew,  newdata=test, se.fit=TRUE)
pred_kurt  <- predict(gam_kurt,  newdata=test, se.fit=TRUE)

test_predictions <- tibble(
  Re = test$Re,
  Fr = test$Fr,
  St = test$St,
  mu_hat    = as.numeric(pred_mu$fit),
  mu_se     = as.numeric(pred_mu$se.fit),
  sigma_hat = as.numeric(pred_sigma$fit),
  sigma_se  = as.numeric(pred_sigma$se.fit),
  skew_hat  = as.numeric(pred_skew$fit),
  skew_se   = as.numeric(pred_skew$se.fit),
  kurt_hat  = as.numeric(pred_kurt$fit),
  kurt_se   = as.numeric(pred_kurt$se.fit)
) %>%
  mutate(
    mu_lower    = mu_hat - 1.96 * mu_se,
    mu_upper    = mu_hat + 1.96 * mu_se,
    sigma_lower = sigma_hat - 1.96 * sigma_se,
    sigma_upper = sigma_hat + 1.96 * sigma_se,
    skew_lower  = skew_hat - 1.96 * skew_se,
    skew_upper  = skew_hat + 1.96 * skew_se,
    kurt_lower  = kurt_hat - 1.96 * kurt_se,
    kurt_upper  = kurt_hat + 1.96 * kurt_se
  )

summary(test_predictions)
```

```
##        Re            Fr               St             mu_hat
##  Min.   : 90   Min.   :0.0520   Min.   :0.0500   Min.   :-0.013697
##  1st Qu.: 90   1st Qu.:0.1760   1st Qu.:0.3500   1st Qu.: 0.001102
##  Median :224   Median :1.0000   Median :0.6000   Median : 0.018049
##  Mean   :228   Mean   :0.6005   Mean   :0.6043   Mean   : 0.053239
##  3rd Qu.:398   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 0.106166
##  Max.   :398   Max.   :1.0000   Max.   :1.0000   Max.   : 0.129125
##      mu_se            sigma_hat         sigma_se        skew_hat
##  Min.   :0.003692   Min.   :-5.134   Min.   :1.527   Min.   : 43.33
##  1st Qu.:0.004068   1st Qu.:-2.956   1st Qu.:1.696   1st Qu.: 52.94
##  Median :0.004358   Median : 4.269   Median :1.795   Median :117.20
##  Mean   :0.004490   Mean   : 3.441   Mean   :1.834   Mean   :150.03
##  3rd Qu.:0.004891   3rd Qu.: 7.465   3rd Qu.:1.994   3rd Qu.:197.45
##  Max.   :0.005328   Max.   :15.821   Max.   :2.118   Max.   :351.86
##     skew_se         kurt_hat         kurt_se        mu_lower
##  Min.   :12.77   Min.   : -5247   Min.   :4653   Min.   :-0.023186
##  1st Qu.:13.83   1st Qu.: 11574   1st Qu.:5037   1st Qu.:-0.007488
##  Median :14.79   Median : 21725   Median :5386   Median : 0.008144
##  Mean   :15.07   Mean   : 37128   Mean   :5488   Mean   : 0.044439
##  3rd Qu.:16.06   3rd Qu.: 53014   3rd Qu.:5850   3rd Qu.: 0.098415
##  Max.   :17.84   Max.   :102670   Max.   :6498   Max.   : 0.120558
##     mu_upper          sigma_lower       sigma_upper        skew_lower
##  Min.   :-0.004208   Min.   :-8.9432   Min.   :-1.3255   Min.   : 12.45
##  1st Qu.: 0.009692   1st Qu.:-6.4515   1st Qu.: 0.8223   1st Qu.: 22.75
```

```
##  Median : 0.027954   Median : 0.4458   Median : 8.2767   Median : 90.95
##  Mean   : 0.062040   Mean   :-0.1547   Mean   : 7.0360   Mean   :120.49
##  3rd Qu.: 0.113916   3rd Qu.: 4.2611   3rd Qu.:10.6686   3rd Qu.:167.55
##  Max.   : 0.137692   Max.   :12.2760   Max.   :19.3670   Max.   :321.95
##    skew_upper       kurt_lower       kurt_upper
##  Min.   : 70.59   Min.   :-16437   Min.   :  5942
##  1st Qu.: 83.13   1st Qu.:  1032   1st Qu.: 22116
##  Median :143.45   Median : 12164   Median : 31287
##  Mean   :179.56   Mean   : 26371   Mean   : 47885
##  3rd Qu.:227.35   3rd Qu.: 42126   3rd Qu.: 63903
##  Max.   :381.76   Max.   : 91778   Max.   :113562
```

```
colSums(!is.finite(as.matrix(test_predictions)))
```

```
##          Re          Fr          St      mu_hat       mu_se   sigma_hat
##           0           0           0           0           0           0
##    sigma_se    skew_hat     skew_se    kurt_hat     kurt_se    mu_lower
##           0           0           0           0           0           0
##    mu_upper sigma_lower sigma_upper  skew_lower  skew_upper  kurt_lower
##           0           0           0           0           0           0
##  kurt_upper
##           0
```

```
submission <- test_predictions %>%
  select(mu_hat, sigma_hat, skew_hat, kurt_hat)

# Save to CSV (submission-ready)
write_csv(submission, "predictions.csv")

# Optional: preview to confirm format
print(head(submission))
```

```
## # A tibble: 6 x 4
##     mu_hat sigma_hat skew_hat kurt_hat
##      <dbl>     <dbl>    <dbl>    <dbl>
## 1 -0.00578      2.04     352.  102670.
## 2 -0.00163      2.91     348.  100981.
## 3  0.00384      3.96     343.   98559.
## 4  0.0180       4.27     315.   84508.
## 5 -0.0137      -5.13     198.   50939.
## 6 -0.00708     -3.82     193.   48117.
```