Predicting Turbulence Simulations

Enzo Moraes Mescall, Martin Olarte, Charlotte Coudert

2022-10-31

```
data.train = read.csv("data-train.csv")
data.test = read.csv("data-test.csv")
```

Introduction

Methodology

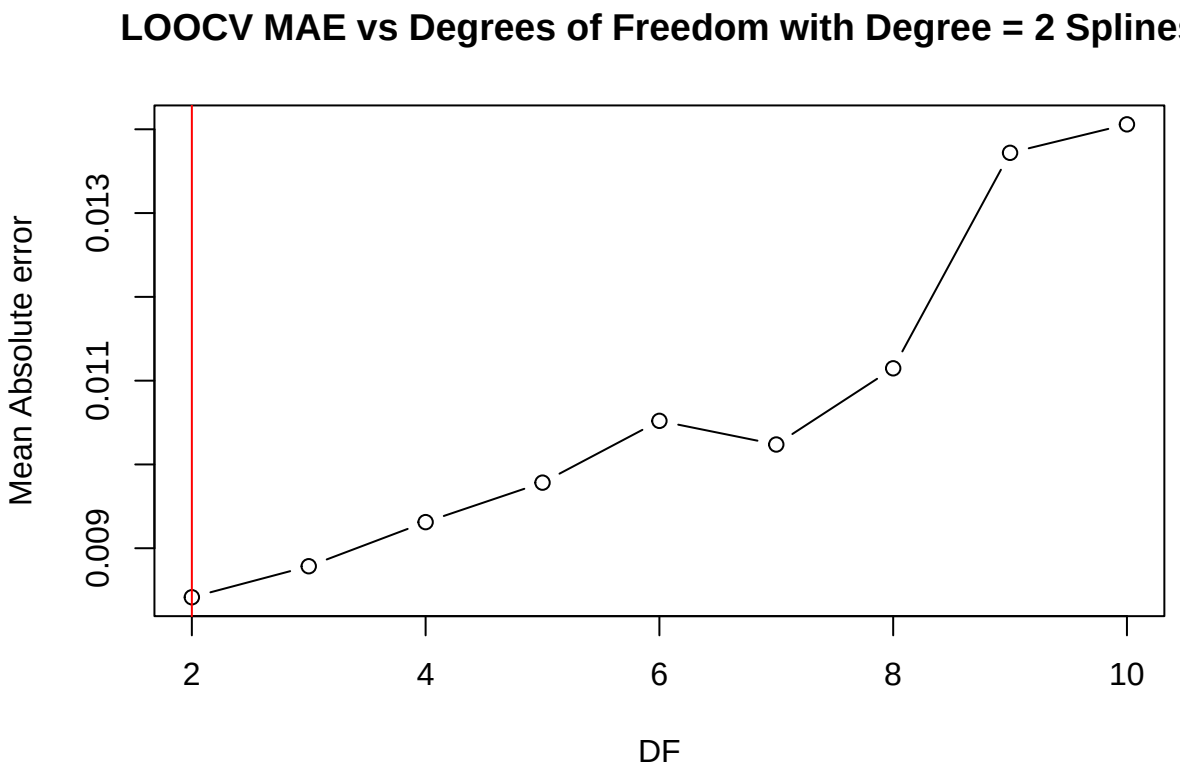
The first noticable thing about the data is that both Fr and Re only have three levels and

```
clean = data.train %>%
  mutate(Fr = as.factor(Fr),
         Re = as.factor(Re))
```

Since we have relatively few data points, n = 89, we decided to rely on LOOCV for our model selection process. We also vetted multiple different cost functions, from MSE to root MSE to root MSlogE to MAE. The main reason for this was that MSE was over-inflating the negative of outliers and making some models seem as if they were performing worse than expected. Root MSE, or rMSE, and MAE helped to taper this by producing more interpretable values that were easier to visualize whilst still preserving the minimum and maximum of the MSE function. One step further was looking at root MSlogE, or RMSLE, which heavily scaled MSE and allowed us to compare the performance of one model to another across differently scaled response variables. For example if the 1st moment's model has an MSE of 0.1 and the 2nd moment's model has an MSE of 0.2, one would be inclined to assume the first model is superior, but since the values of the second moment tend to be orders of magnitude higher the latter model is actually performing a much more impressive estimation. This relative difference is much better captured in RMSLE

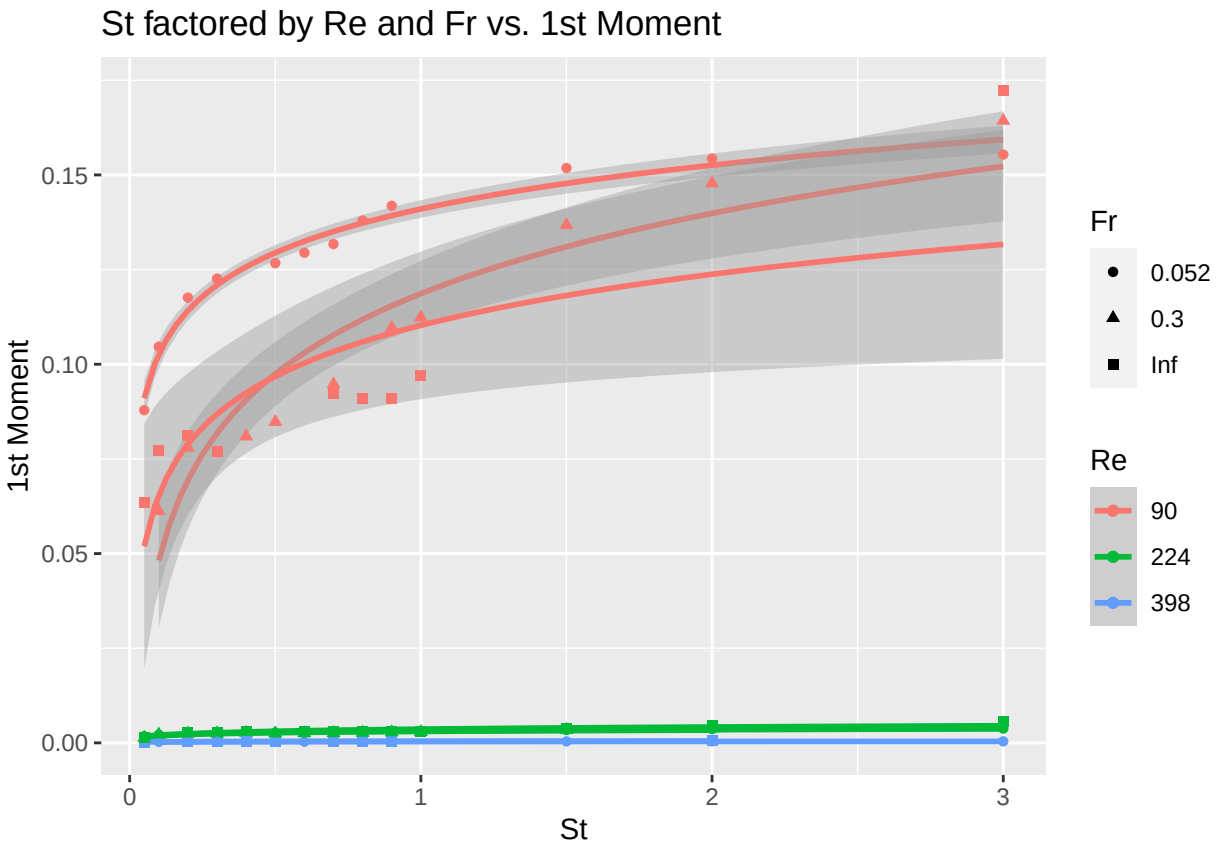
The order we tackled this problem in was to first estimate the 1st moment, and then work our way up. To create a proficient model we estimated the test error performance of various different degrees of polynomials and splines with various combinations of degrees of freedom. An example is shown in the graph below of the process looking at the difference in LOOCV MAE between different numbers of knots for a quadratic spline model.

```
RMSLE = function(y_true, y_pred) sqrt(mean((log(y_true + 1) - log(y_pred + 1))^2))
MAE = function(y_true, y_pred) mean(abs(y_true - y_pred))

errors = rep(NA, 10)
for (i in 2:10) {
  lm = glm(R_moment_1 ~ bs(St, df = i, degree = 2)*Re + Fr, data = clean)
  errors[i] = suppressWarnings(cv.glm(clean, lm, cost = MAE)$delta[1])
}
plot(2:10, errors[-1], type="b", xlab="DF", ylab="Mean Absolute error",
     main = 'LOOCV MAE vs Degrees of Freedom with Degree = 2 Splines')
abline(v = which.min(errors), col = "red")
```

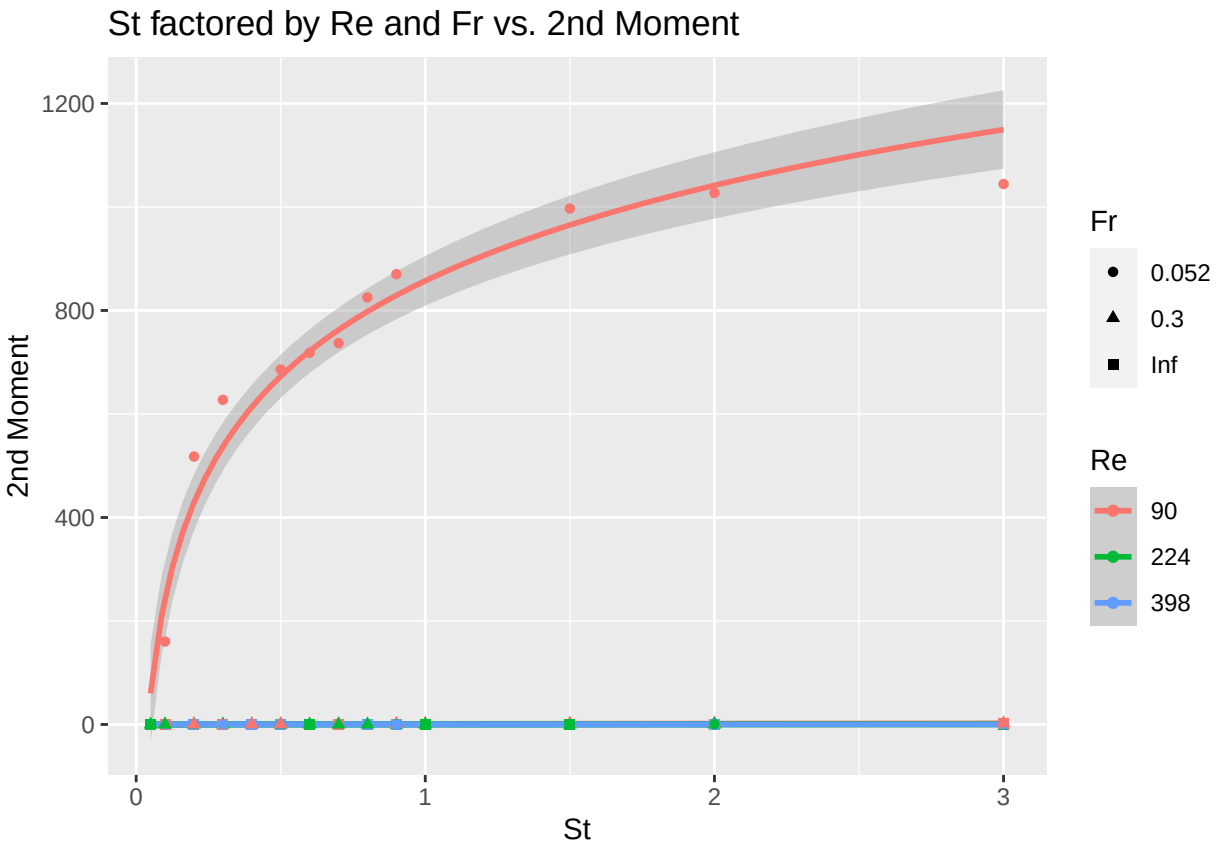


Notably, the simplest model has won out. The next step in our process was to identify possible interaction terms. We started with a very complex model with a complete set of interactions terms, which included triple interaction terms across the three predictors, and included log, square root, and inverse terms. We then performed backwards model selection using LOOCV, MSE, and MAE to select predictors and all metrics agreed on the following models for both the first and second moments.

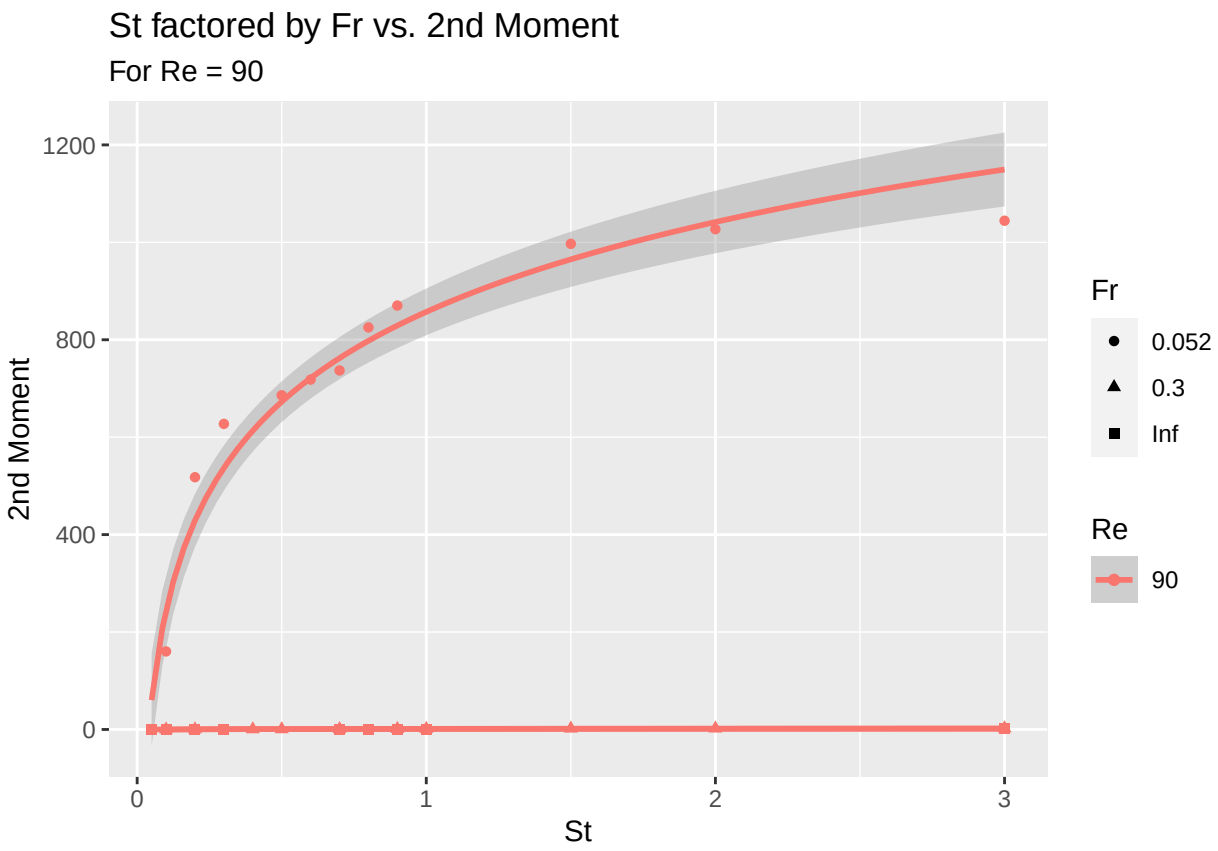
```
#if we keep this i will add labels and such

re_over = data.train %>%
  filter(Re == 90) %>%
  mutate(Fr = as.factor(Fr),
         Re = as.factor(Re))

re_under = data.train %>%
  filter(Re != 90) %>%
  mutate(Fr = as.factor(Fr),
         Re = as.factor(Re))


par(mfrow=c(2,2))


  ggplot(aes(x = St, y = R_moment_1, color = Re, shape = Fr), data = clean) +
  labs(title = "St factored by Re and Fr vs. 1st Moment", x = "St", y = "1st Moment") +
  geom_smooth(method = "lm", formula = "y ~ log(x)") +
  geom_point()
```



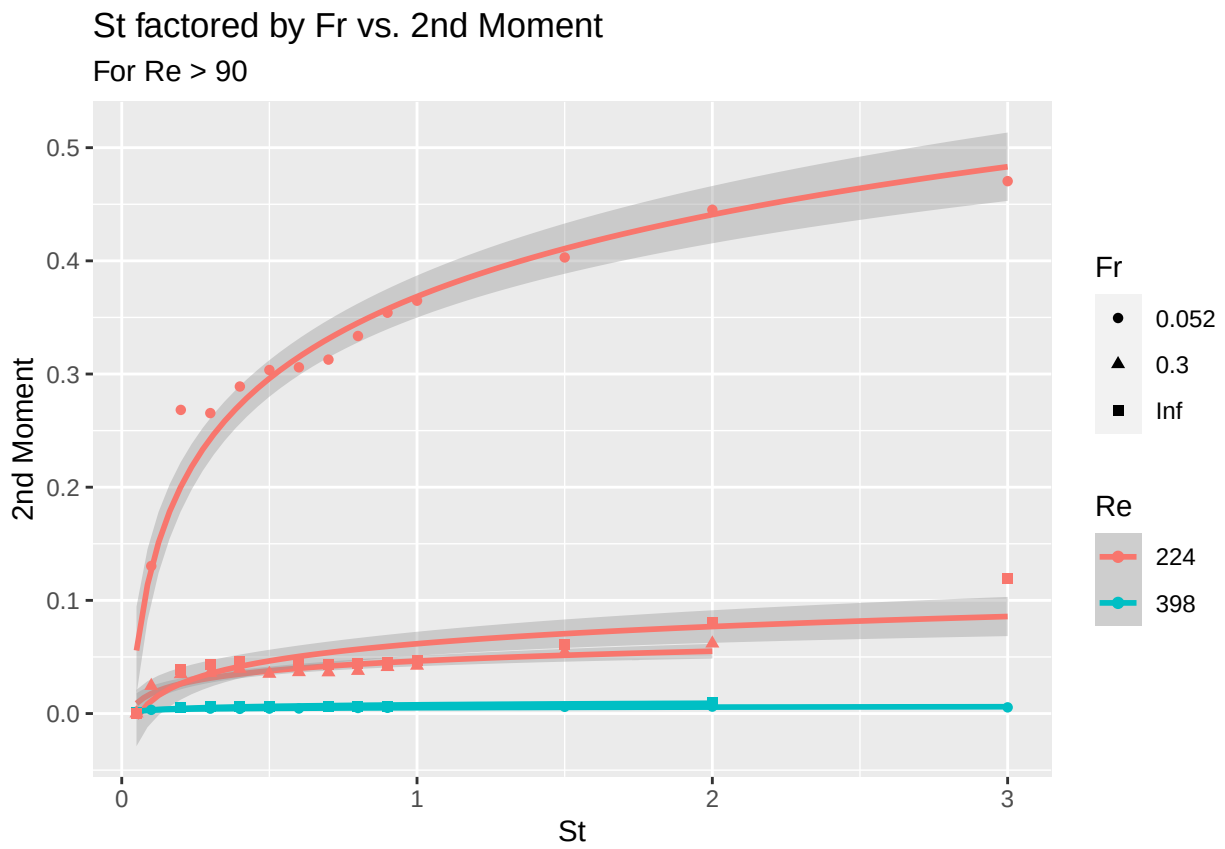

```
ggplot(aes(x = St, y = R_moment_2, color = Re, shape = Fr), data = clean) +
labs(title = "St factored by Re and Fr vs. 2nd Moment", x = "St", y = "2nd Moment") +
geom_smooth(method = "lm", formula = "y ~ log(x)") +
geom_point()
```




```
ggplot(aes(x = St, y = R_moment_2, color = Re, shape = Fr), data = re_over) +
labs(title = "St factored by Fr vs. 2nd Moment", subtitle = "For Re = 90", x = "St", y = "2nd Moment")
geom_smooth(method = "lm", formula = "y ~ log(x)") +
geom_point()
```




```
ggplot(aes(x = St, y = R_moment_2, color = Re, shape = Fr), data = re_under) +
labs(title = "St factored by Fr vs. 2nd Moment", subtitle = "For Re > 90", x = "St", y = "2nd Moment")
  geom_smooth(method = "lm", formula = "y ~ log(x)") +
  geom_point()
```



To look at what kind of interactions we may need, we graphed the data. As we get to further moments, it becomes very clear that the majority of our error came from the much larger values all associated with an Re of 90 and an Fr of 0.052. These looked to follow a logarithmic path of St and when we applied that to all the interaction terms, we found our best model yet.

```
model_m1 = glm(R_moment_1 ~ log(St)*Fr*Re + St*Fr*Re, data = clean)
model_m2 = glm(R_moment_2 ~ log(St)*Fr*Re + St*Fr*Re, data = clean)
```

After this, we created a function that would resemble our final product and take in a tuple of St, Re, and Fr and then return the four predicted moments. We reserved the global variable names `model_mX` for this final function.

```
# temp models
predictive_model = function(test_St, test_Re, test_Fr) {
  newdata = data.frame(list(St = test_St,
                            Re = as.factor(test_Re),
                            Fr = as.factor(test_Fr)))

  m1 = predict(model_m1, newdata =  newdata)
  m2 = predict(model_m2, newdata =  newdata)
  m3 = predict(model_m3, newdata =  newdata)
  m4 = predict(model_m4, newdata =  newdata)

  return(c(m1, m2, m3, m4))
}
```

Results

Conclusion