DataProject

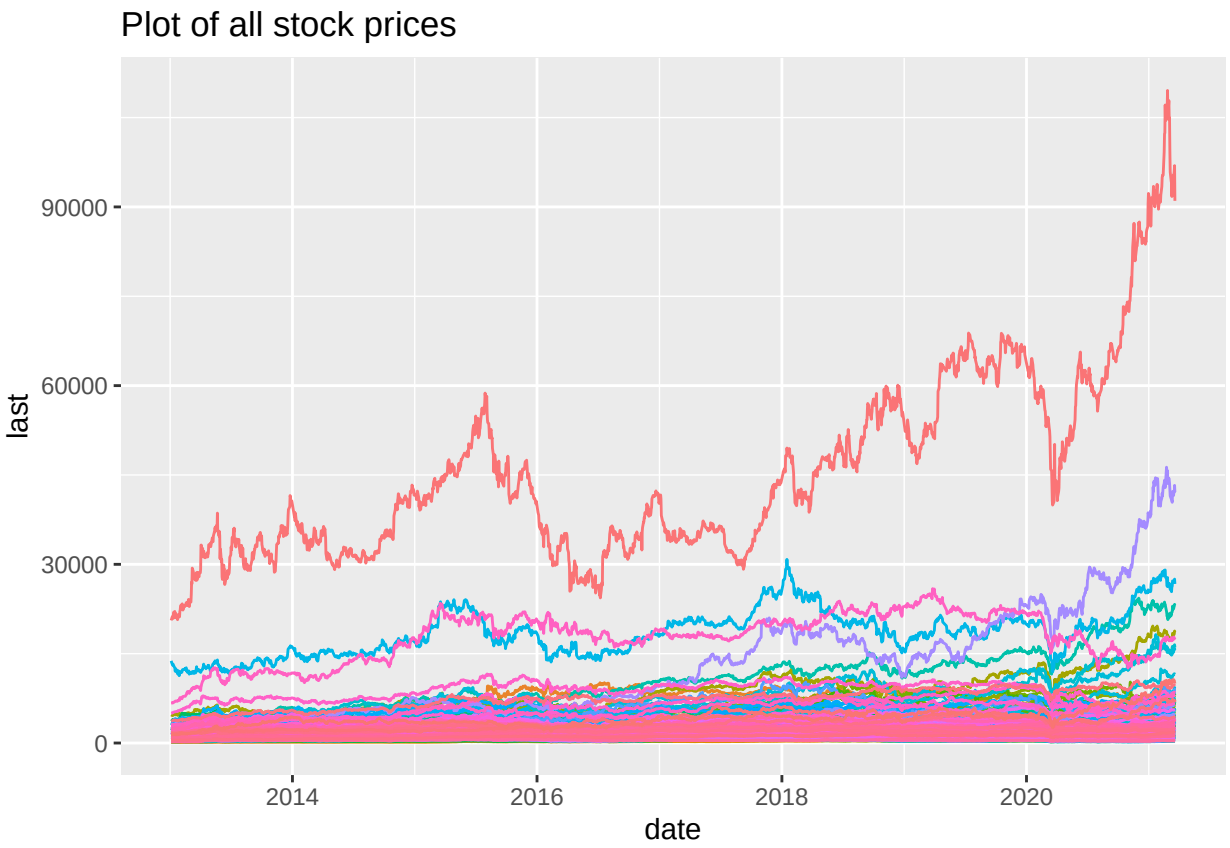
Houjie Wang

2023-11-13

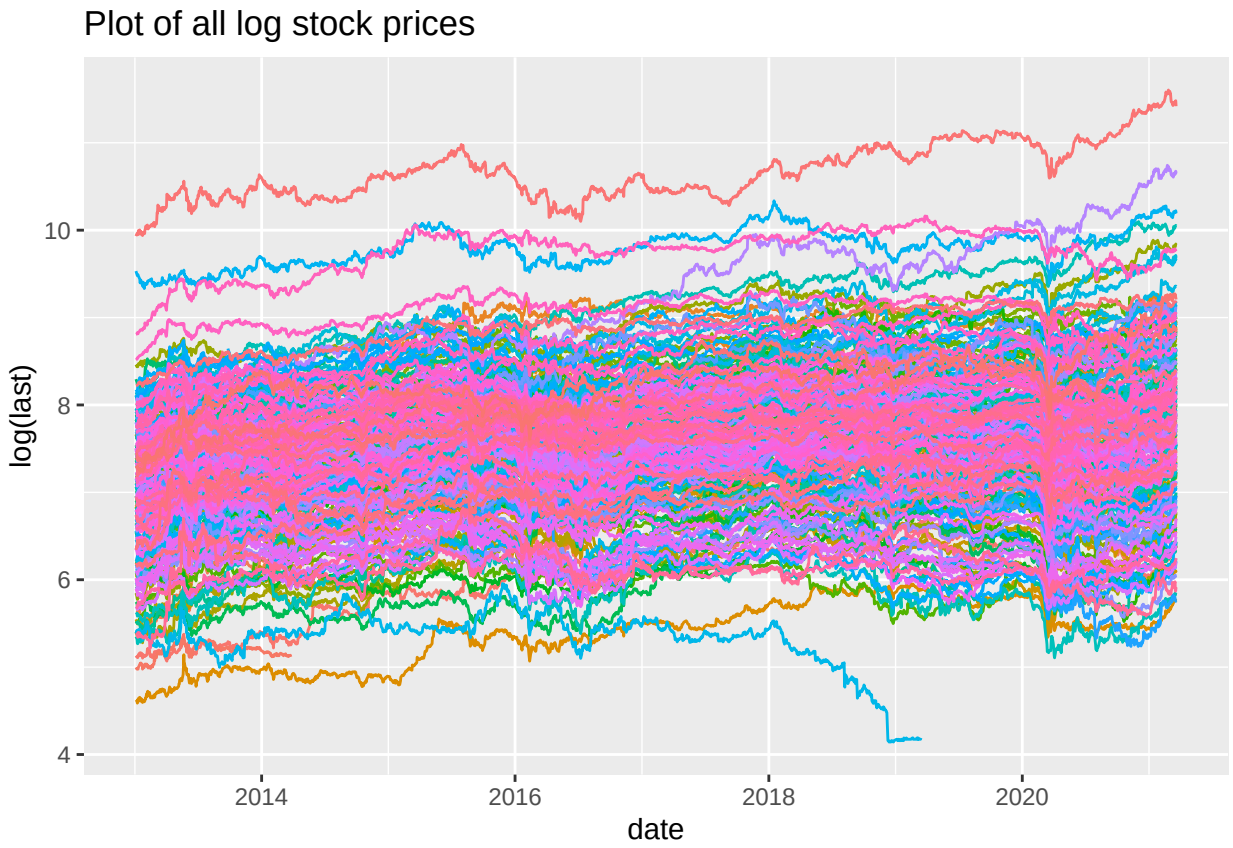
Some EDA Plots

In this section, we simply visualize some line plots of certain values just to obtain a rough sense of the data.

```
# Plot of all prices
ggplot(data = full_df[full_df$ticker %in% unique_stocks[nobs_stock == 2005], ],
       aes(x = date, y = last, color = ticker)) + geom_line() +
  theme(legend.position = "none") + ggtitle("Plot of all stock prices")
```

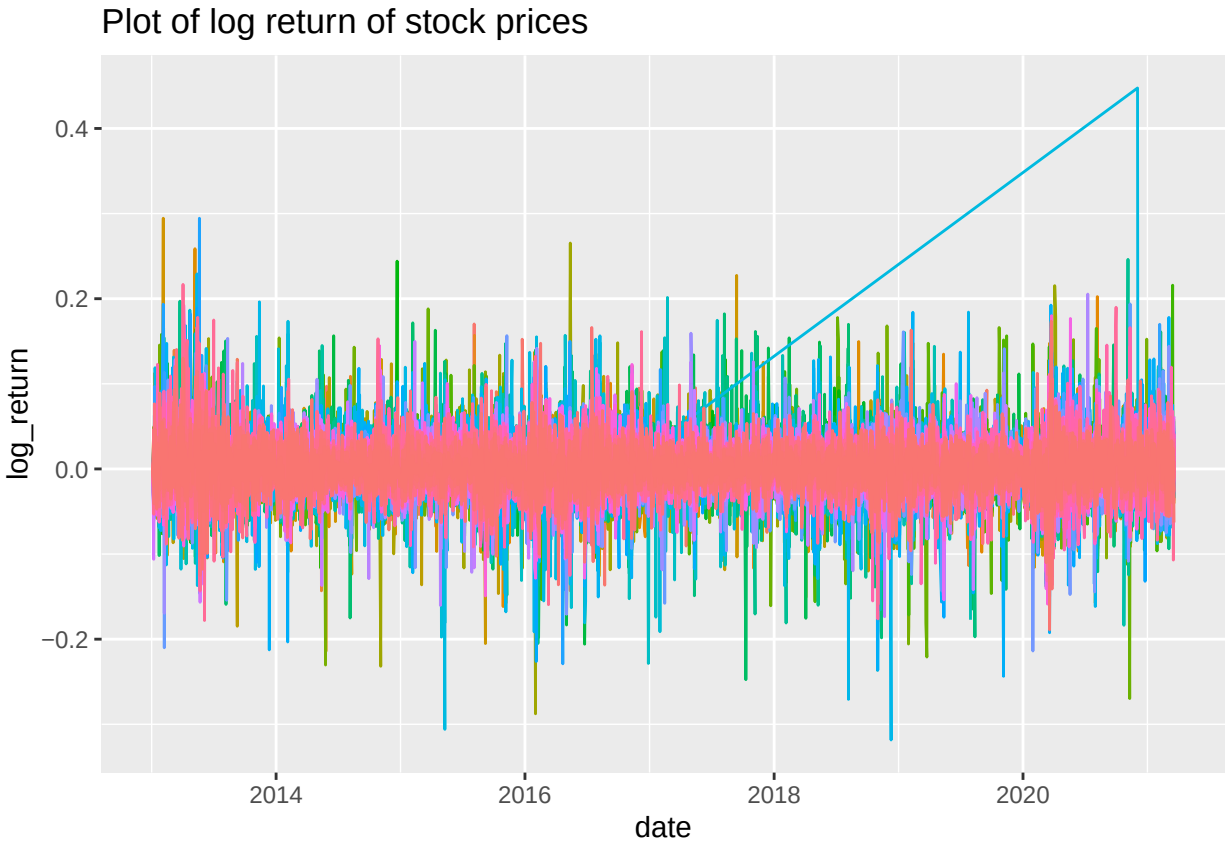



```
# Plot of all log prices
ggplot(data = full_df,
       aes(x = date, y = log(last), color = ticker)) + geom_line() +
  theme(legend.position = "none") + ggtitle("Plot of all log stock prices")
```




```
full_df2 = do.call("rbind", lapply(unique_stocks, function(stock){
  # stock = "1332 JT"
  subdat = full_df[full_df$ticker == stock, ]
  subdat$log_return = c(0, log(subdat$last[-1] / subdat$last[-nrow(subdat)]))
  subdat
}))

ggplot(data = full_df2,
       aes(x = date, y = log_return, color = ticker)) + geom_line() +
  theme(legend.position = "none")+ ggtitle("Plot of log return of stock prices")
```

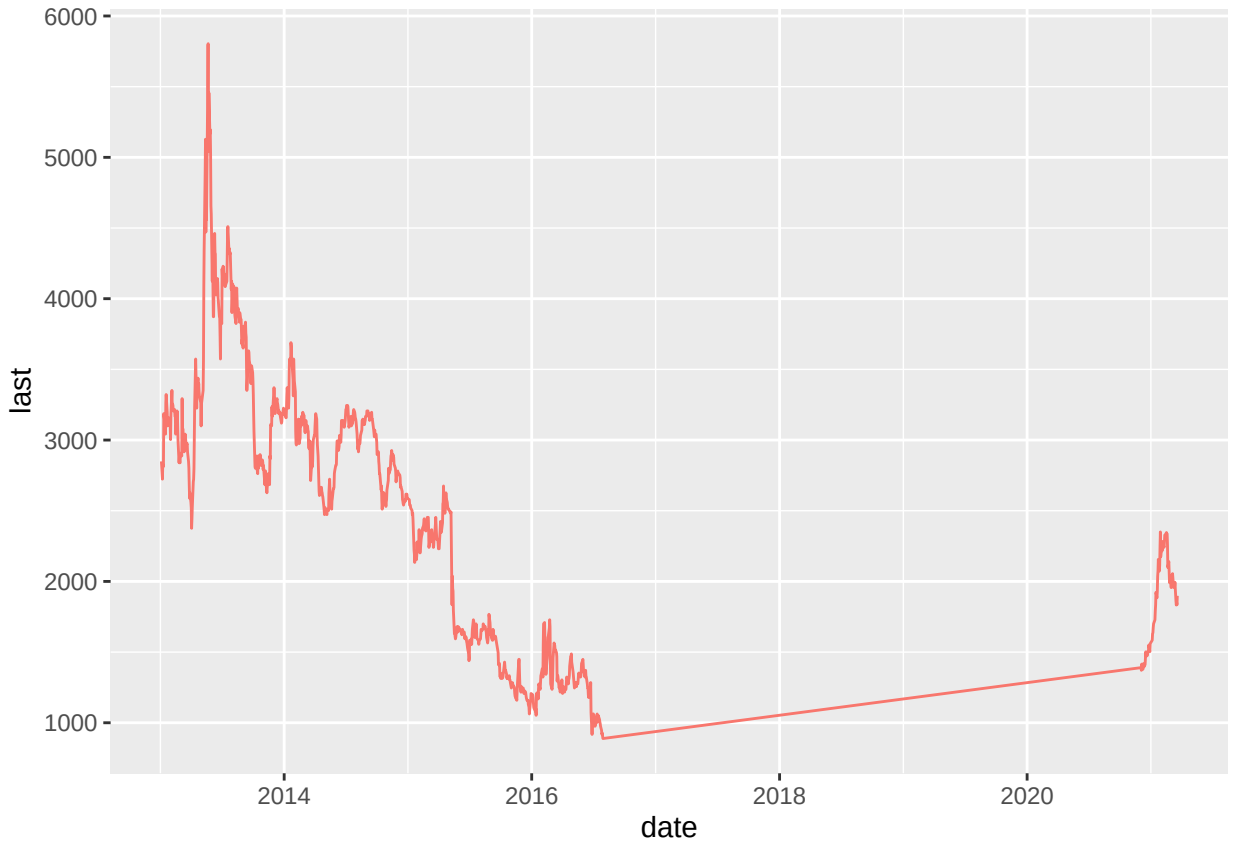


The log return looks stationary as it should be.

Data Processing

Based on the log return plot, it seems that there's a weird data point, which is stock 6753 JT. It has missingness for at most a gap of 1587 days. It's removed because imputation for it seems to be weired and inappropriate otherwise we'll be injecting too much prior information to that stock prices. Given the large amount of stock prices observation we have, we believe it is okay to just remove it.

```
ggplot(data = full_df2[full_df2$ticker == "6753 JT", ],
       aes(x = date, y = last, color = ticker)) + geom_line() +
  theme(legend.position = "none")
```




```
missing_gaps = sapply(unique_stocks, function(stock){
  # stock = "6753 JT"
  subdat = full_df[full_df$ticker == stock, ]
  range(as.integer(subdat$date[-1] - subdat$date[-nrow(subdat)]))
})

# Remove the weird stock
full_df2 = full_df2[full_df2$ticker != "6753 JT", ]
```

```
table(nobs_stock) / sum(table(nobs_stock))
```

```
## nobs_stock
##          55          95         114         157         180         184
## 0.004032258 0.004032258 0.004032258 0.004032258 0.004032258 0.004032258
##         300         356         396         480         486         538
## 0.004032258 0.004032258 0.004032258 0.004032258 0.004032258 0.004032258
##         597         672         791         842         884         894
## 0.004032258 0.008064516 0.004032258 0.008064516 0.004032258 0.004032258
##         917         948         991        1014        1016        1088
## 0.004032258 0.004032258 0.004032258 0.004032258 0.004032258 0.004032258
##        1121        1130        1163        1210        1333        1408
## 0.004032258 0.004032258 0.008064516 0.004032258 0.008064516 0.004032258
##        1467        1519        1525        1609        1649        1701
## 0.004032258 0.004032258 0.004032258 0.004032258 0.004032258 0.004032258
##        1821        1825        1848        1891        1932        1946
## 0.004032258 0.004032258 0.004032258 0.004032258 0.004032258 0.004032258
```

```
##        2005
## 0.814516129
```

```
stock2005 = unique_stocks[nobs_stock == 2005]
sub_df2 = full_df2[full_df2$ticker %in% stock2005, ]
```

We do a sanity check and find that roughly 81% of the stock prices has 2005 observations, which makes it easy for analysis as they all have observations for the same date. To simplify, we will only focus on them in the following analysis.

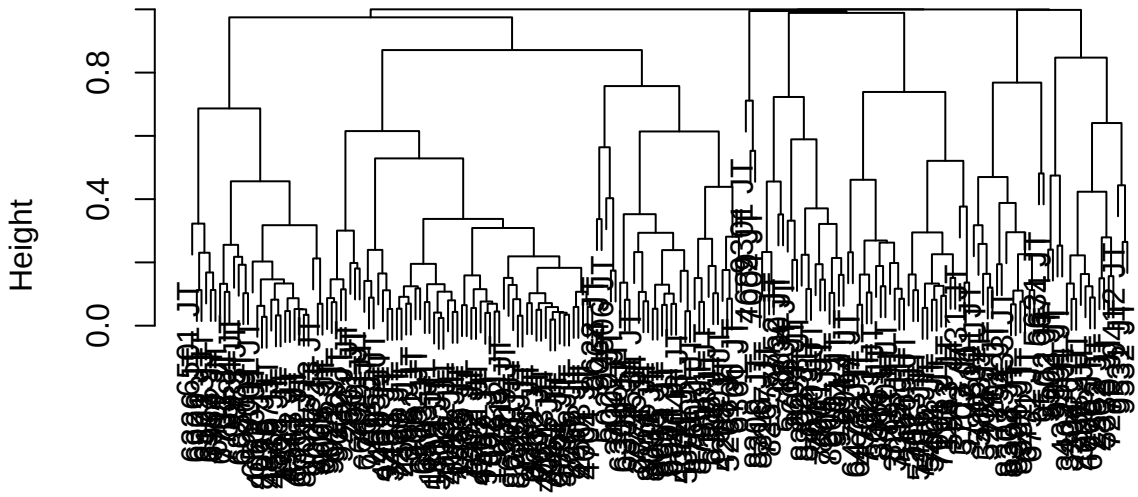
Analysis

We believe that there exists strong correlation among the stock prices given some common characteristics they have (e.g. same type of business, collaboration, and competition etc.). For example, those who are collaborating and working on the same business, it is reasonable to assume that their stock prices are positively correlated and for those competiting (e.g. Intel vs. AMD), they might be negatively correlated as some success of one party might diminish the advantage of other.

However, there are 202 of them and it would be counterproductive to model one stock prices conditioned on all other stock prices. To handle this, we first carry out a hierarchical clustering based on their correlation matrix. Common clustering algorithms are based on distance measures, so here we simply use the $1 - |\rho|$ as the dissimilarity measure between stock prices such that the more correlated the any two stocks, they more likely they will have the same cluster membership.

```
nobs = 2005
prices = sapply(stock2005, function(stock){
  as.numeric(sub_df2$last[sub_df2$ticker == stock])
})
volumes = sapply(stock2005, function(stock){
  as.numeric(sub_df2$volume[sub_df2$ticker == stock])
})
lprices = log(prices)
cor_lprices = cor(lprices)
# We perform a hierarchical clustering on the correlation matrix to obtain clusters of
# stock prices with strong correlation
stock_clusters = hclust(as.dist(1-abs(cor_lprices)))
```

```
plot(stock_clusters)
```

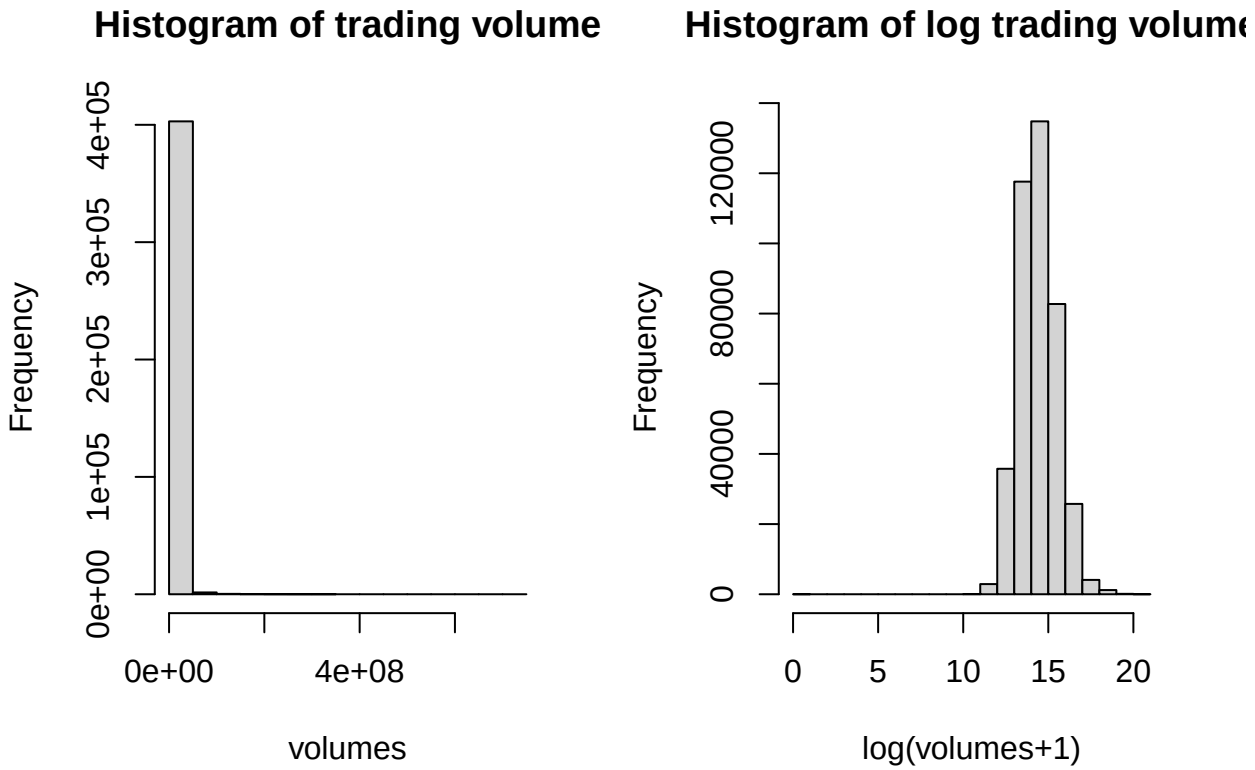



```
height_to_cut <- 0.8 # cut tree at the height where each cluster has at least multiple stock prices
clusters <- cutree(stock_clusters, h = height_to_cut)
nClusters = max(clusters)
table(clusters)
```

```
## clusters
##  1  2  3  4  5  6  7  8  9
## 58 27 29 14 32 16  3 19  4
```

We plot the dendrogram and choose to cut at height 0.8 to ensure that each cluster has at least multiple stock prices. This would ensure the effectiveness of our model since we believe that in such a low-resource settings (e.g. not many endogeneous covaraiets but only trading volume at least to my opinion), the correlation among stock prices would play a key role in determining the future prices and therefore if the clustering structure is too sparse, the model might not be able to learn appropriately. We end up with 9 clusters.

```
# empirical volume dist. if too right skewed, log it to standardize a bit
lvolumes = log(volumes+1)
cor_lvolumes = cor(log(volumes+1))
par(mfrow = c(1, 2))
hist(volumes, main = "Histogram of trading volume")
hist(lvolumes, main = "Histogram of log trading volume", xlab = "log(volumes+1)")
```



We also take the log of the trading volume to reduce its skewness as a more normally distributed covaraite is always preferred at least in linear models (and all kinds of its derivatives as well).

We propose to use cluster-specific elastic net regression to jointly model the stock prices. It is easy to implement and has been showed to have robustness against overfitting and advantage in prediction, which makes it an approprite candidate for our analysis. Moreover, unlike lasso, whose solution set is non-convex and has non-unique solution, the solution set of elastic net is still convex and therefore has unique solution. Unique soution is preferred since coefficient interpretation is also preferred where we would like to identify a small subset of feature works well for the response. Non-unique set of features derived from lasso might be misleading sometimes (although here we don't explore much w.r.t coefficient interpretation). The downside is that it does not easily generate valid variance estimates and makes it hard to carry out protofolio optimization by solving for an optimal combination (usually by Lagrange multipler) of proportions to distribute our capital while minimizing the variance of our future predictions. Bootstrap has been popular to approximate variance but unfortunately our data has temporal dependence where bootstrap won't work. Some better solutions might be to use the Bayesian version of elastic net (i.e. combining double-exponential and normal prior but it requires MCMC so not quite good) or Bayesian dynamic generalized linear models (a state-space model at its core but leaverages the idea of Kalman filter and normal-normal conjugacy for efficient modeling fitting, inference and preiction).

Our model is as follows:

$$\boldsymbol{Y}_t^c = \sum_{k=1}^{n_\delta} \beta_k \boldsymbol{Y}_{t-\delta_k}^c + \sum_{k=1}^{n_\delta} \alpha_k \boldsymbol{X}_{t-\delta_k}^c + \boldsymbol{\epsilon}_t^c$$

$$\text{s.t.}(1-\lambda)\sum_{k=1}^{n_\delta} ||\beta_k|| + ||\alpha_k|| + \lambda \sum_{k=1}^{n_\delta} ||\beta_k||_2 + ||\alpha_k||_2 \leq t,$$

where $\boldsymbol{Y}_t^c$ is the matrix of stock prices in cluster c. $\delta_k \in \{4, 7, 10, 13, 16, 19\}$ is the lagged stock price observation and $\boldsymbol{X}_t^c$ is the lagged trading volume.

We propose a very naive strategy. That is, for each trading day in the testing period where we also have the prediction the stock prices in four trading days, if the predicted stock price in four days is higher than the

current predicted price, then we buy one share and sell in four days regardless of the actual prices, and if the predicted stock price is lower in four days than the current predicted price, we short one share and buy back in four days. We gain profit if our prediction is correct and we lose otherwise.

```
all_lags = seq(4, 20, 3) # lagged observations we would like to use

ntrn = 1600 # take the first 1600 as training data
ntst = nobs - ntrn
lprice_tst = lprices[-c(1: ntrn), ]
lag = 4
fee_rate = 0.001
registerDoMC(cores = 4) # parallel computing for cross-validation in elastic net
stock_specific_profit = list()
prediction_accuracy = list()
for (i in 1: nClusters) {
  # i = 2
  stocks = which(clusters == i)

  # This is the joint response of stock prices for cluster i
  # The first y is the 20-th last price
  Y_reponse = lprices[(1+max(all_lags)): ntrn, stocks]

  # The covariates are the log prices and volume 4, 7, 10, 13, 16, 19 trading days ago
  X = do.call("cbind", lapply(all_lags, function(lag){
    cbind(lprices[seq(1+max(all_lags) - lag, ntrn, length.out = nrow(Y_reponse)), ],
          lvolumes[seq(1+max(all_lags) - lag, ntrn, length.out = nrow(Y_reponse)), ])
  }))
  cv_model <- cv.glmnet(x = X, y = Y_reponse, alpha = 0.05, family = "mgaussian", nfold = 4, parallel =
  lambda_best <- cv_model$lambda.min

  Y_response_tst = lprices[-c(1: ntrn), stocks]
  X_tst = do.call("cbind", lapply(all_lags, function(lag){
    cbind(lprices[seq(1+ntrn - lag, nobs, length.out = ntst), ],
          lvolumes[seq(1+ntrn - lag, nobs, length.out = ntst), ])
  }))
  Y_pred <- predict(cv_model, newx = X_tst, s = lambda_best)[,,1]

  # Now we check the proportion of our model correctly predicting the increase or decrease of
  # stock prices in the next 4-th trading day

  ls_pred = Y_pred[-c(1: lag), ] - Y_pred[1: (ntst-lag), ] > 0

  ls_real = Y_response_tst[-c(1: lag), ] - Y_response_tst[1: (ntst-lag), ] > 0

  prediction_accuracy[[i]] = colMeans(ls_real - ls_pred == 0)
  # print(prediction_accuracy[i])

  cumulative_profit <- c()
  for (j in 1:ncol(Y_pred)) {
    # j = 1
    # Calculate actual returns for the stock
    actual_returns = calculate_returns(Y_response_tst[, j])
    actual_prices = exp(Y_response_tst[, j])  # Converting log prices back to actual prices
```

Table 1: Accuracy of prediction of increase/decrease in four days

cluster1	cluster2	cluster3	cluster4	cluster5	cluster6	cluster7	cluster8	cluster9
57.8	60.2	58.1	57	58.3	61.5	62.5	60.9	60.9

Table 2: Profit of from strategy

cluster1	cluster2	cluster3	cluster4	cluster5	cluster6	cluster7	cluster8	cluster9
10925.45	6060.185	32958.19	5766.581	8459.629	7019.823	5536.328	10334.24	4095.765

```
    # Generate trading signals based on predicted prices four days in advance
    # Comparing predicted prices (from Y_pred) for a day with actual prices four days later

    # 1 means long today and sell in four days; -1 means short today and buy back in four days
    signals <- ifelse(Y_pred[-c(1: lag), j] > Y_pred[1: (ntst-lag), j], 1, -1)

    sum(actual_prices[1: length(signals)][signals == 1])

    # This is the profit generated by only buying or selling one share at a time informed by the model
    profit = sum((actual_prices[-c(1: lag)]*(1 - fee_rate) - actual_prices[1: (ntst-lag)]*(1 + fee_rate)


    # Store the net returns for the stock
    cumulative_profit[j] = profit
  }

  stock_specific_profit[[i]] = cumulative_profit
}
```

```
mean_prediction_accuracy = sapply(prediction_accuracy, mean)
names(mean_prediction_accuracy) = paste0("cluster", 1: 9)
kableExtra::kable(t(round(mean_prediction_accuracy, 3))*100, caption = "Accuracy of prediction of increa
```

Here we present the cluster-specific accuracy of prediction in four days, where all of them are above 55%, indicating its potential to make more profit.

```
mean_profit = sapply(stock_specific_profit, mean)
names(mean_profit) = paste0("cluster", 1: 9)
kableExtra::kable(t(round(mean_profit, 3)), caption = "Profit of from strategy")
```

Here we present the cluster-specific mean profit, where all of them are positive.