Appendix 4: Documentation of Code

Max Harleman, Chris Moloney, Minbae Lee, Dan Kardish, Andrew Morgan

April 8, 2019; Updated July - August 2021 by Andrew Morgan

Brief Introduction and Summary:

This file is a compilation of various statistical learning methods we (Max Harleman, Chris Moloney, Minbae Lee, Dan Kardish, Andrew Morgan) performed as a group on the CATO Human Freedom Index that measures human freedom throughout the world on a country-level. This analysis was completed for Statistical Learning course at the University of Pittsburgh in the Spring of 2019. The project had several parts including finding the dataset, performing various methods (as shown in this markdown file), the conclusions of findings (this markdown file and report we wrote as individuals), and a group presentation. We recommend only reading through (or downloading the R-Markdown file) if you are interested in the details of the methods performed or the code we wrote.

The **primary** task of the project was to use real data and various statistical methods learned in class to demonstrate the pros and cons each method possesses. The class had a heavy focus on the concept of the bias-variance tradeoff. Meaning some models may capture less variance by being more biased than others (like multiple linear regression). Or models may capture a lot of variance, but have very little bias (random forests or neural networks). High bias tends to allow for more inferencing to analyze the effects of specific predictors. High variance tends to fit the data better (guaranteed on training set), however the accuracy may falter if the model is overfit (big issue with high-variance models like neural networks). The methods used are multiple linear regression (MLR), bootstrapping (primarily cross-validation), polynomial regression, splines, GAMs (Generalized Additive Models), KNN regression, decision trees, random forest (bagging), boosting. Some other models not seen here are related to classification (our dataset is a regression problem) like linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA).

The **secondary** task of the project is to determine the most significant predictors of human freedom. The dataset is large (39 predictors, and 1305 observations after data tidying). By determining the most significant predictors, we should have a better grasp of what determines human freedom. Then, we will create a smaller model (~10 predictors) to try and capture the greatest amount of variance possible within the dataset. If this smaller model is relatively close to the most accurate models, we can assert that the human freedom index could be simplified. An important note here is that this task is largely based around **inferenceing** which some models perform better than others. The models that have better inferencing tend to have biased models to allow for these inferences (again like multiple linear regression). Hence, some models could more accurate, but due to limited inferencing to determine those significant predictors, we may decide to not focus on that model (unless the accuracy is significantly better than the preferred model).

Another Note: We decided on using a random seed to get consistent results for our analysis. Obviously this will bias the dataset, so we chose the number '111' as the seed value at the begining of the analysis before using any methods. This number was randomly chosen, and will constantly appear with the comman 'set.seed(111)' at the start of many code chunks.

Andrew's Website Changes

In the Summer of 2021, I (Andrew), have modified the project in certain ways including the Markdown file and the Rendered PDF and HTML files.

Changelog

The Goal: Get the project into a state I am proud of and displays some of my statistical learning skills.

Overall the changes are made as an improvement over the existing code. Most of the code was initially written by Max and Myself (Andrew), as we had the most programming experience of the group. **I believe Max Harleman's work on the project was incredible and much credit for this code needs to go to him!**

- Primary reason for the alterations is to reacquaint myself with the project, dataset, and packages used.
- Secondary reason for the changes are to improve the visualizations for my written summary about this project's findings.
- Tertiary reason for the changes is some large errors were found and refactoring was desired due to inconsistent naming, syntax, spacing.
- **Fixed the major Error.** The test set not being held out from training some models. Often we seemed to use the combined (train + test) for cross-validation to find the better estimate of the MSE. Then, we take the best cross-validated model and test it against the test set (but it was trained on that data as well), making some conclusions very wrong. Luckily, the findings are not really affected by this potentially catastrophic error.
- Improvements to graphs and visualizations. I improved the descriptions (labels and points) within the graphs and added new graphs as visualizations for my written post on my website.
- Many of my changes result in more time, memory, and resource consumption. This is unfortunate, but I believe the results are now more robust and less biased.
- Refactoring: I revised the code to have better style, syntax, and flexibility. Additionally, some errors were discovered that we missed over 2 years ago.
- Restructure and Reorganized: I reorganized the code to be more concise and explicit about tasks and methods used. Additionally, some code was rearranged to be in a more logical spot in terms of part of the analysis it should/had been performed.
- Areas Expanded: Non-linear section with polynomial regression, splines, and GAMS were improved to test more models than there were initially. The Findings do not seem to change the final conclusions much (the models are significantly more accurate, but still not preferable to the MLR models).
- Areas with Major Refactoring: I improved the multiple linear regression section (more analysis of best models found), tree-based methods (decision trees, random forests, boosting; focus on the most important predictors within models),
- A larger focus in conclusion about our "Final" model(s). I create a final models using the predictors we isolated as most significant (I added some additional as I believe we missed some. The models used in this section were MLR, MLR with polynomial terms, Random Forest, and Boosting.

1. Data Cleaning and Setup:

1.1 Load the Libraries

1.2

The following Code aims to load, tidy, and format the data and resulting dataframe into a state that can be easily manipulated, altered, and used for the following methods. The goals are as follows:

- Many variables are related and explain the same thing as aggregates for more specific metrics already captured. For example, there is a predictor that measures the amount of personal freedom with respect to the rule-of-law of the land that is an average (or aggregate) of more specific predictors like security, protection of private property, safety for citizens from murder or kidnappings. We remove the aggregate, while including the specific predictors.
- We remove columns with a large number of NAs. This is far from ideal, but we have no good way of estimating these values. These columns that are removed, frankly are not terribly important, and were only added to the dataset in more recent years.
- We tested the dataset for leverage points and outliers. We found and removed these observations using code found here: https://stats.stackexchange.com/questions/164099/removing-outliers-based-on-cooks-distance-in-r-language/345040.
- Now we need to convert the data to a dataframe ready for analysis. This dataframe then needs to be split into a train and test set (80:20). The train and test sets are used for getting a more accurate and less biased estimate of the MSE (mean-squared error) from our models.

```
################################   GET DATA READY   ############################

#######  THE FILE LOCATION MAY NOT BE RIGHT, YOU NEED TO CORRECT IT  ##########
# Read in the dataset
hfi_data = read.csv(file = "C:/GitHub/Private-Projects/School/2019-Spring/STAT - Statistical Learning a
###############################################################################


## Creating a county by year row names as the new 'identifier' or primary key
country= hfi_data$countries
year= hfi_data$year
co_year= paste(country, year, sep = " ", collapse = NULL)
hfi_data= data.frame(co_year, hfi_data)
hfi_data$ISO_code <- NULL
hfi_data$countries <- NULL
hfi_data$region <- NULL
hfi_data$year <- NULL
rownames(hfi_data)=hfi_data[,1]
hfi_data$co_year <- NULL

# Get the # columns and # rows before removing the columns and rows with N/A's
# nrow(hfi_data)
# ncol(hfi_data)

# Start adding columns/variables to remove.
# These vars are aggragates of other variables listed, and therefore we remove.
cols_to_remove = c("pf_rol", "pf_ss", "pf_movement", "pf_ss_women", "pf_religion",
                   "pf_association", "pf_expression", "pf_identity", "pf_score", "pf_rank",
                   "ef_government", "ef_legal", "ef_money", "ef_trade", "ef_regulation",
```

```
                    "ef_score", "ef_rank", "pf_identity_sex", "ef_trade_movement",
                    "pf_ss_disappearances", "ef_regulation_labor", "ef_regulation_business")
hfi_data = hfi_data %>% dplyr::select(-cols_to_remove)

# Second batch of columns to remove
# This finds columns with less consistent values for a given country
# Remove if there are many (100+) N/A's for a given variable/predictor
cols_to_remove = list()
list_counter = 1
for (i in 1:ncol(hfi_data)){
  # checking if the number of N/A's is 100 or more
  if(sum(is.na(hfi_data[,i])) >= 100 ){
    # append the column name that has 100+ N/A's
    cols_to_remove[[list_counter]] = colnames(hfi_data[i])
    list_counter = list_counter + 1     # update the counter
  }
}
cols_to_remove = unlist(cols_to_remove)  # creates a vector to use with dyplr
hfi_data = hfi_data %>% dplyr::select(-cols_to_remove)

# After removing the columns producing significant number of N/A's
# Remove all observations that still have miscellaneous N/A's
hfi_data = hfi_data %>% na.omit()

# Get the # columns and # rows after removing dimensions and observations
# nrow(hfi_data)
# ncol(hfi_data)

# the list of columns that relate to human freedom score (the response) to ignore
responses = c("hf_rank", "hf_quartile")
hfi_data_features = hfi_data %>% dplyr::select(-(c(responses, "hf_score")))
hfi_data_combined = hfi_data %>% dplyr::select(-responses)

# Get the # columns and # rows after removing dimensions and observations
# nrow(hfi_data)
# ncol(hfi_data)

# simply used to get same results regardless of run. Methods should be robust to the seed changing, tho
set.seed(111)

# create the Training and Testing Set
hfi_data_combined_train = hfi_data_combined %>% sample_frac(size = .8)
hfi_data_combined_test = hfi_data_combined %>% setdiff(hfi_data_combined_train)

# get the Mean-TSS of the Training and Testing Sets; Use for judging effectiveness
mean_tss_combined = mean((mean(hfi_data_combined$hf_score) - hfi_data_combined$hf_score)^2)
mean_tss_train = mean((mean(hfi_data_combined_train$hf_score) - hfi_data_combined_train$hf_score)^2)
mean_tss_test = mean((mean(hfi_data_combined_test$hf_score) - hfi_data_combined_test$hf_score)^2)
```

The most important takeaway is the estimates for Mean Total Sums Squared (mean TSS). These values represent the expected variance using just the average human freedom score (the response) instead of a more complex model (it is simply an intercept = mean(hf_score)). The point is to see how much more variance is explained from other models using the resulting Mean Squared Error (MSE). However, the MSE is averaged, so we take the average of the TSS to result in the mean TSS in order to have comparable values. The Mean

TSS is primarily used to judge a model's test MSE (the MSE from a model on NEW data the model has never seen).

- Mean TSS for all the observations (combined test and train dataset): 0.9762658
- Mean TSS for all the *training* observations: 0.9995088
- Mean TSS for all the *testing* observations: 0.882886 (the most important one to compare models' MSEs)

2. Exploratory Analysis

Create a Correlation Matrix with Permutation Testing

The first exploratory analysis will be to create a correlation matrix to analyze how linearly correlated the features of the dataset are. The primary focus will be on the Human Freedom Score (hf_score) that measures human freedom within a given country. The hf_score will be located on the very first row of the matrix (first element) and the far right column (last element in columns). The matrix is split into 2 different measurements:

- Top-Left of the matrix is the permutation testing results. Permutation testing uses bootstrap-like techniques to get a better estimate of the correlation. If there is a black dot for a row, column pair, that means the pair of features are significantly correlated (most likely not a correlation of 0).
- Bottom-Right of the matrix is the correlation values in different shades of red/orange and blue. The more red a row, column feature pair is, the more positively, linearly correlated the features are; the more blue indicates that there is a negatively, linearly correlated feature pair.
- The diagonal from top-right to bottom-left is the correlation of the same predictors (so the correlation of hf_score and hf_score, or pf_ss_homicide and pf_ss_homicide). These correlation values will always be +1 (dark red). These values should be ignored.

```
##############  CORRELATION WITH 'hf_score' RESPONSE   ####################
# Get all correlations (will use the rest later for a correlation matrix)
actual_correlation_matrix = cor(x = hfi_data_combined)
hf_score_correlations = actual_correlation_matrix['hf_score', ]

# This sorts the ACTUAL and MAGNITUDE of correlations
actual_sorted_hf_score_correlations = sort(hf_score_correlations, decreasing = TRUE)
abs_sorted_hf_score_correlations = sort(abs(hf_score_correlations), decreasing = TRUE)

head(actual_sorted_hf_score_correlations, 10)
```

```
##                      hf_score          pf_expression_control
##                     1.0000000                      0.7586429
##        pf_expression_influence              ef_legal_military
##                     0.7457798                      0.7308047
##           ef_trade_regulatory ef_trade_regulatory_compliance
##                     0.7073884                      0.6632339
##               ef_legal_gender          ef_trade_tariffs_mean
##                     0.6104228                      0.6017655
##          pf_movement_domestic                    ef_money_sd
##                     0.5561843                      0.5466009
```

```
tail(actual_sorted_hf_score_correlations, 4)
```

```
##    pf_religion_harassment       ef_trade_tariffs_sd ef_regulation_labor_hours
##                0.10589262                0.06847924                0.06347336
## ef_government_consumption
##               -0.32431183
```

```
head(abs_sorted_hf_score_correlations, 10)
```

```
##                      hf_score          pf_expression_control
##                     1.0000000                      0.7586429
##        pf_expression_influence              ef_legal_military
##                     0.7457798                      0.7308047
##           ef_trade_regulatory ef_trade_regulatory_compliance
##                     0.7073884                      0.6632339
```

```
##                ef_legal_gender          ef_trade_tariffs_mean
##                      0.6104228                      0.6017655
##           pf_movement_domestic                    ef_money_sd
##                      0.5561843                      0.5466009
```

```
###############    Correlation Matrix Testing    ####################
# Using Bootstrap Testing/Estimating to estimate Correlation throughout dataset
# Useful for finding insights to how correlated data is with a more robust
#     correlation estimate
nperms = 100 # smaller number of permutations to speed up processing
significance_level = 0.02  # better with the smaller nperms

nperms = 1000  # a larger number, slow process
significance_level = 0.005  # better with the larger nperms
permutation_upper_estimate = data.frame(matrix(NA ,
                                               ncol = ncol(hfi_data_combined),
                                               nrow = ncol(hfi_data_combined)),
                                        row.names = rownames(actual_correlation_matrix))
colnames(permutation_upper_estimate) = colnames(actual_correlation_matrix)

permutation_lower_estimate = data.frame(matrix(NA ,
                                               ncol = ncol(hfi_data_combined),
                                               nrow = ncol(hfi_data_combined)),
                                        row.names = rownames(actual_correlation_matrix))
colnames(permutation_lower_estimate) = colnames(actual_correlation_matrix)


for ( i in 1:ncol(actual_correlation_matrix)){
  for( j in 1:nrow(actual_correlation_matrix)){
    if( i > j){
      i_name = rownames(actual_correlation_matrix)[i]
      j_name = rownames(actual_correlation_matrix)[i]
      # ONLY do half the matrix
      permutation_vector = rep(-10, nperms)
      for (k in 1:nperms) {
        suffle_i = sample(x = hfi_data_combined[, i_name],
                          size = nrow(hfi_data_combined),
                          replace = FALSE)
        permutation_vector[k] = cor(suffle_i, hfi_data_combined[, j_name])
      }
      permutation_lower_estimate[i,j] = quantile(permutation_vector, significance_level)
      permutation_upper_estimate[i,j] = quantile(permutation_vector, 1-significance_level)
    } else if (i == j){
      # correlation of X an X is ALWAYS == 1
      permutation_lower_estimate[i,j] = 1
      permutation_upper_estimate[i,j] = 1
    }
  }
}

# These functions get the Upper and Lower 'Triangles' of a correlation matrix
get_upper_tri <- function(cormat){
  cormat[lower.tri(cormat)]<- NA
  return(cormat)
```

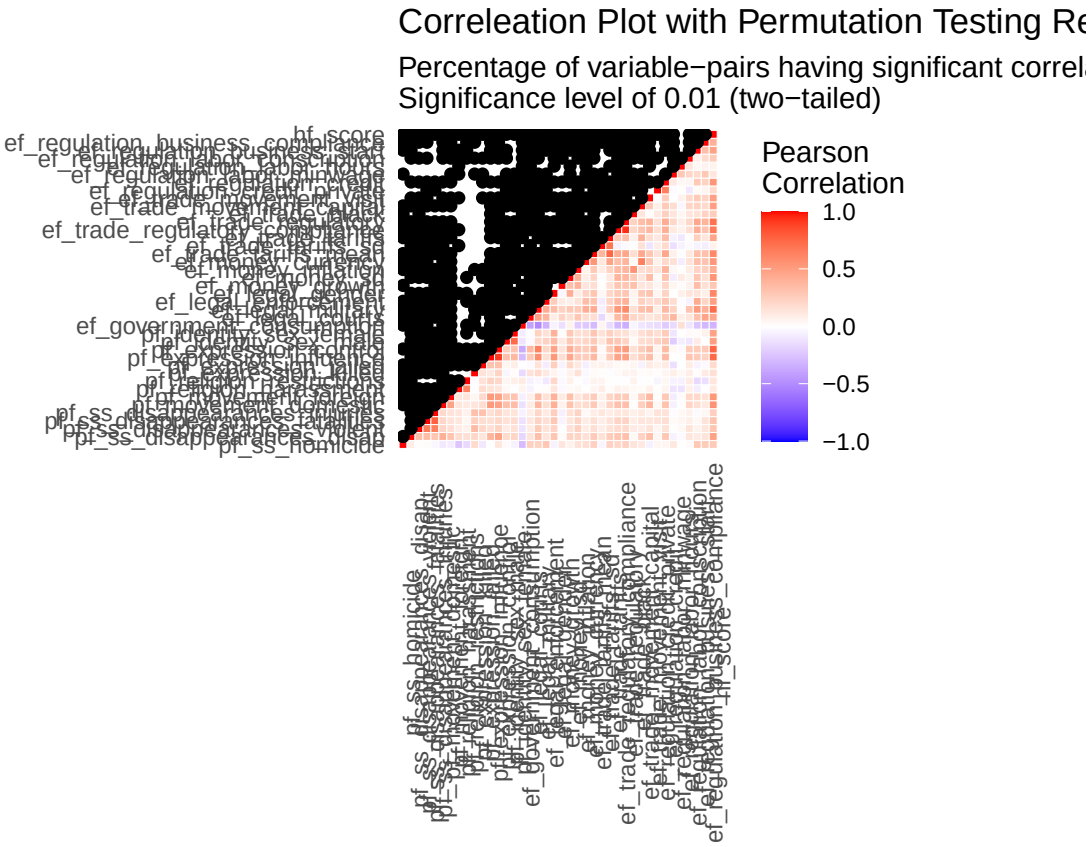
```
}

get_lower_tri<-function(cormat){
  cormat[upper.tri(cormat)] <- NA
  return(cormat)
}

# Get All pairs of Vars that are significant via permutation testing
is_significant = matrix(NA , ncol = ncol(hfi_data_combined),
                        nrow = ncol(hfi_data_combined),
                        dimnames = list(c(colnames(hfi_data_combined)),
                                        c(colnames(hfi_data_combined))))
total = 0
count = 0
col_names = colnames(hfi_data_combined)
for ( i in 1:ncol(hfi_data_combined)){
  for( j in 1:ncol(hfi_data_combined)){
    if( i > j){
      total = total + 1
      if( permutation_lower_estimate[i,j] <= actual_correlation_matrix[i,j] &
          actual_correlation_matrix[i,j] <= permutation_upper_estimate[i,j] ){
        is_significant[i, j] = NA  # this will remove the 'FALSE' values later
      } else{
        is_significant[i,j] = TRUE
        count = count + 1
      }
    }
  }
}

rownames(is_significant)
```

```
##  [1] "pf_ss_homicide"                  "pf_ss_disappearances_disap"
##  [3] "pf_ss_disappearances_violent"    "pf_ss_disappearances_fatalities"
##  [5] "pf_ss_disappearances_injuries"   "pf_movement_domestic"
##  [7] "pf_movement_foreign"             "pf_religion_harassment"
##  [9] "pf_religion_restrictions"        "pf_expression_killed"
## [11] "pf_expression_jailed"            "pf_expression_influence"
## [13] "pf_expression_control"           "pf_identity_sex_male"
## [15] "pf_identity_sex_female"          "ef_government_consumption"
## [17] "ef_legal_courts"                 "ef_legal_military"
## [19] "ef_legal_enforcement"            "ef_legal_gender"
## [21] "ef_money_growth"                 "ef_money_sd"
## [23] "ef_money_inflation"              "ef_money_currency"
## [25] "ef_trade_tariffs_mean"           "ef_trade_tariffs_sd"
## [27] "ef_trade_tariffs"                "ef_trade_regulatory_compliance"
## [29] "ef_trade_regulatory"             "ef_trade_black"
## [31] "ef_trade_movement_capital"       "ef_trade_movement_visit"
## [33] "ef_regulation_credit_private"    "ef_regulation_credit"
## [35] "ef_regulation_labor_minwage"     "ef_regulation_labor_hours"
## [37] "ef_regulation_labor_conscription" "ef_regulation_business_start"
## [39] "ef_regulation_business_compliance" "hf_score"
```

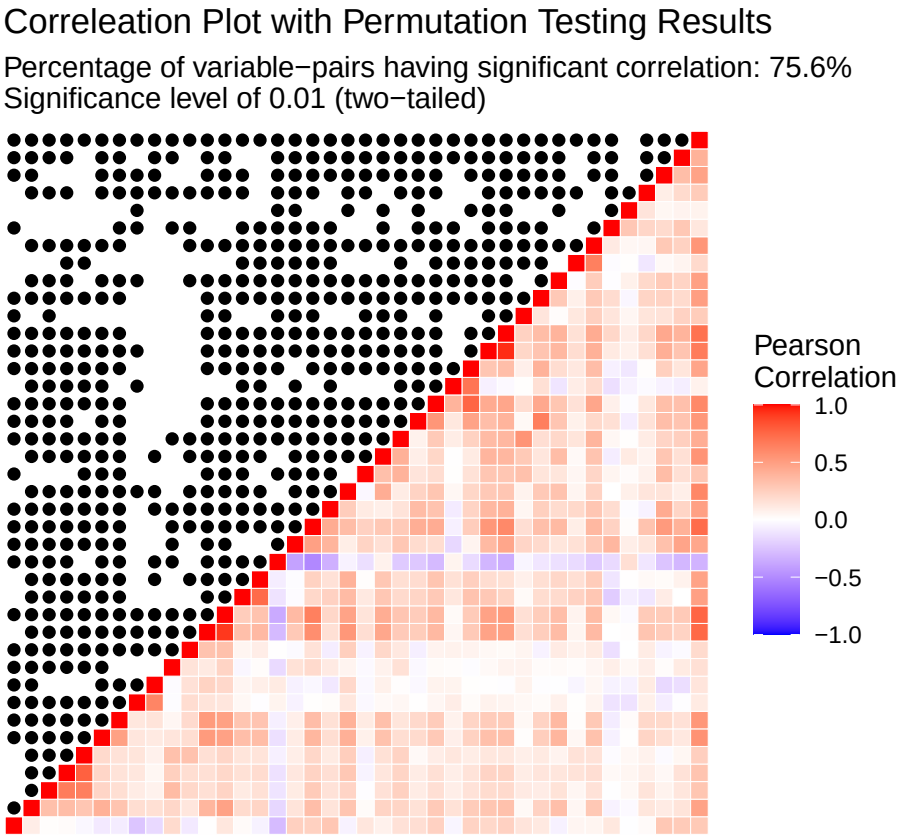
```r
# Now to Combine into 1 dataframe
upper_triangle = get_upper_tri(actual_correlation_matrix)
lower_triangle = get_lower_tri(is_significant)

melted_upper = melt(upper_triangle, na.rm = TRUE)
melted_lower = melt(lower_triangle, na.rm = TRUE)
colnames(melted_lower)[3] = "IsSignificant"

graph_df = melted_upper %>%full_join(melted_lower)
```

```
## Joining, by = c("Var1", "Var2")
```

```r
graph_df$IsSignificant = as.factor(as.character(graph_df$IsSignificant))
graph_df_final = graph_df %>% mutate(value = ifelse(is.na(value), 0, value))

# Graph the Dataframe!
ggplot(data = graph_df_final, aes(Var2, Var1, fill = value)) +
  geom_tile(color = "white") +
  geom_point(aes(shape = IsSignificant, fill=value), size=2 ) +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal() +
  scale_shape(guide = 'none') +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.major = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank(),
    axis.text.x = element_text(angle = 90),
  ) +
  coord_fixed() +
  ggtitle("Correleation Plot with Permutation Testing Results",
          subtitle = paste0("Percentage of variable-pairs having significant correlation: ",
                            ((round((count/total),3)) * 100),
                            "%\nSignificance level of ", significance_level*2, " (two-tailed)"))
```




```
# Graph the Dataframe! with no VARNAMES!
ggplot(data = graph_df_final, aes(Var2, Var1, fill = value)) +
  geom_tile(color = "white") +
  geom_point(aes(shape = IsSignificant, fill=value), size=2 ) +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal() +
  scale_shape(guide = 'none') +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.major = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank()
  ) +
  coord_fixed() +
  ggtitle("Correleation Plot with Permutation Testing Results",
          subtitle = paste0("Percentage of variable-pairs having significant correlation: ",
                            ((round((count/total),3)) * 100),
                            "%\nSignificance level of ", significance_level*2, " (two-tailed)"))
```




```
# This is SUPER HIGH Correlation
actual_correlation_matrix['ef_trade_regulatory',
                          'ef_trade_regulatory_compliance']
```

```
## [1] 0.9452313
```

At a glance, the color scheme indicates a lot of red, meaning many predictors are positively, linearly correlated with each other. Additionally, there are many black dots meaning almost all feature pairs are correlated with a 99% accuracy based on permutation testing. The hf_score (measuring the human freedom variable) is found on the first row (very top of matrix) and last column (far right of matrix). All but 1 variable seems to have correlation with hf_score. This finding indicates the multiple linear regression technique may yield very useful and accurate predictions.

The variables ef_trade_regulatory and ef_trade_regulatory_compliance are heavily correlated predictors. This does not surprise us as one deals with how harsh regulations are to businesses and the other is how strictly the country follows through. The countries with lenient policies will tend to have lacking enforcement, while heavily regulated economies will have their governments cracking down on companies breaking regulations.

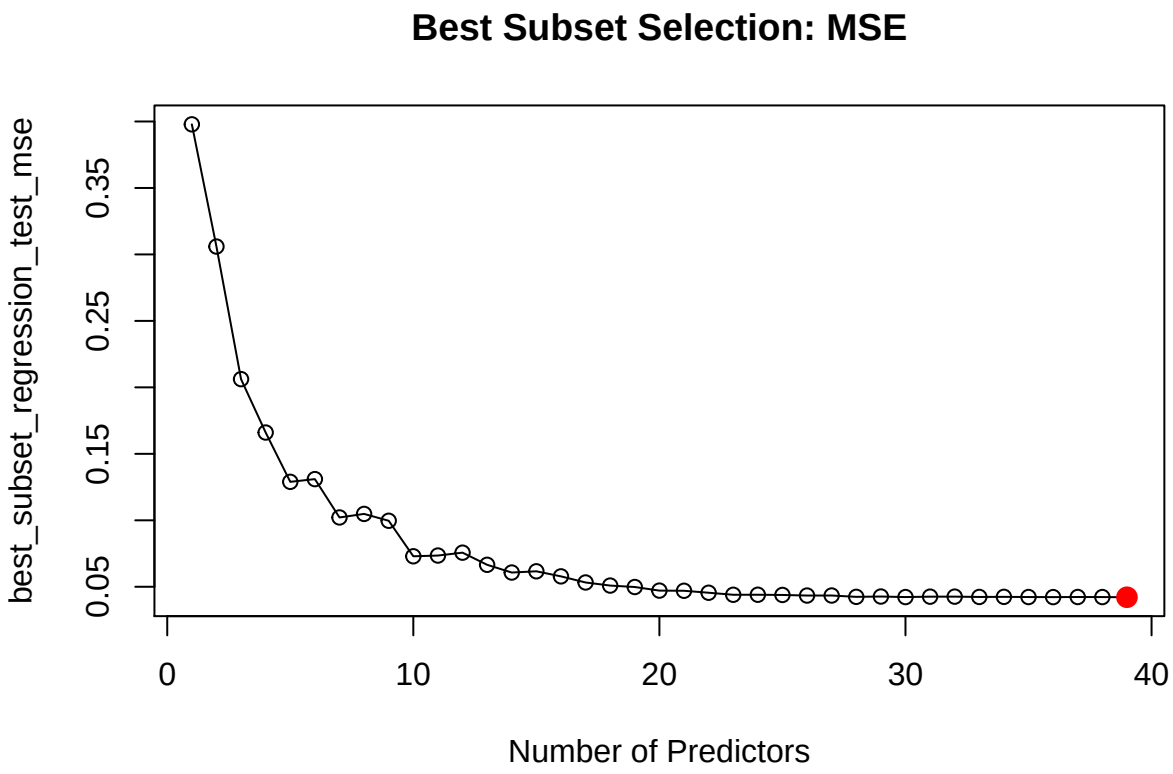
This is an example of the significant correlation between the predictors within the dataset. This finding indicates the human freedom index can probably be measured will minimal loss in accuracy with fewer predictors. Additionally, the correlation may indicate confounding predictors. Confounding predictors are predictors that capture the same variance as the predictors themselves are heavily correlated (like ef_trade_regulatory and ef_trade_regulatory_compliance). These confounding variables are known to not improve model accuracy significantly if 1 exists within the model already.

3. Linear Regression and Model Selection Methods:

3.1 Best-Subset Model Selection

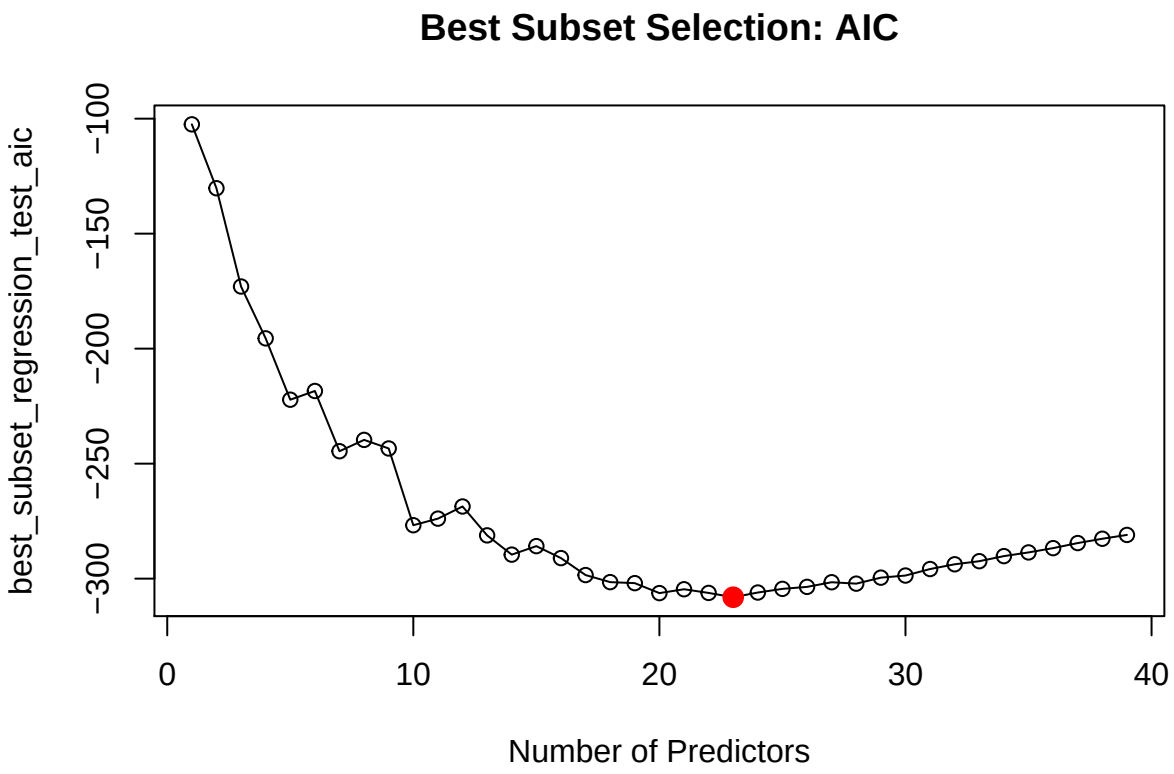
The MLR selection technique to select the best model. We exhaustively look through all the permutations of the predictors with best-subset, and select the best model for each x-predictor subset. **This section takes time.**

```
# Best-Subset
set.seed(111)
max_predictors_best_subset = 39   # set to max; can alter if takes too long
# This trains *all* possible permutations for MLR of the data
best_subset_regression = regsubsets(hf_score ~ ., data = hfi_data_combined_train,
                                    method = "exhaustive",
                                    nvmax = max_predictors_best_subset, intercept = TRUE)
result = summary(best_subset_regression)

# Now to find the Test Errors to judge which model is best!
# Doing Generic MSE and AIC to penalize extra terms/predictors more
best_subset_regression_test_mse = rep(-1, max_predictors_best_subset)
best_subset_regression_test_aic = rep(-1, max_predictors_best_subset)

for (i in 1:max_predictors_best_subset){
  coef_i = coef(best_subset_regression, id = i)
  yhat_temp = as.matrix(
    hfi_data_combined_test[,colnames(hfi_data_combined_test) %in% names(coef_i)] ) %*%
    coef_i[names(coef_i) %in% colnames(hfi_data_combined_test)]
  yhat = as.vector(yhat_temp) + coef_i["(Intercept)"]
  # MSE = RSS / N (Average of Residuals)
  best_subset_regression_test_mse[i] = mean((yhat- hfi_data_combined_test$hf_score )^2)

  # AIC = 2*P + N * LOG(RSS/N) === 2*P + N * LOG(MSE)
  # P = # predictors; N = number of obs in test set
  best_subset_regression_test_aic[i] = 2*i +
    nrow(hfi_data_combined_test) * log(best_subset_regression_test_mse[i], base = 10)
}

# plot the regular MSE With highlighted min. point
plot(best_subset_regression_test_mse, main = "Best Subset Selection: MSE",
     xlab = 'Number of Predictors') +
  lines(x = 1:max_predictors_best_subset, y=best_subset_regression_test_mse) +
  points(x=which.min(best_subset_regression_test_mse),
         y=best_subset_regression_test_mse[which.min(best_subset_regression_test_mse)],
         col = "red", pch = 16, cex = 1.5)
```




```
## integer(0)
```

```
# plot the AIC results With highlighted min. point
plot(best_subset_regression_test_aic, main = "Best Subset Selection: AIC",
     xlab = 'Number of Predictors') +
  lines(x = 1:max_predictors_best_subset, y=best_subset_regression_test_aic) +
  points(x=which.min(best_subset_regression_test_aic),
         y=best_subset_regression_test_aic[which.min(best_subset_regression_test_aic)],
         col = "red", pch = 16, cex = 1.5)
```




```
## integer(0)
```

```
best_subset_regression_test_mse[which.min(best_subset_regression_test_mse)]
```

```
## [1] 0.04212529
```

```
best_subset_regression_test_aic[which.min(best_subset_regression_test_aic)]
```

```
## [1] -308.0917
```

```
which.min(best_subset_regression_test_mse)  # Number of Predictors with Best MSE
```

```
## [1] 39
```

```
which.min(best_subset_regression_test_aic)  # Number of Predictors with Best AIC
```

```
## [1] 23
```

The best-subset model results in 2 different findings. We did a default best-subset selection and an AIC (penalizing larger, more complex models).

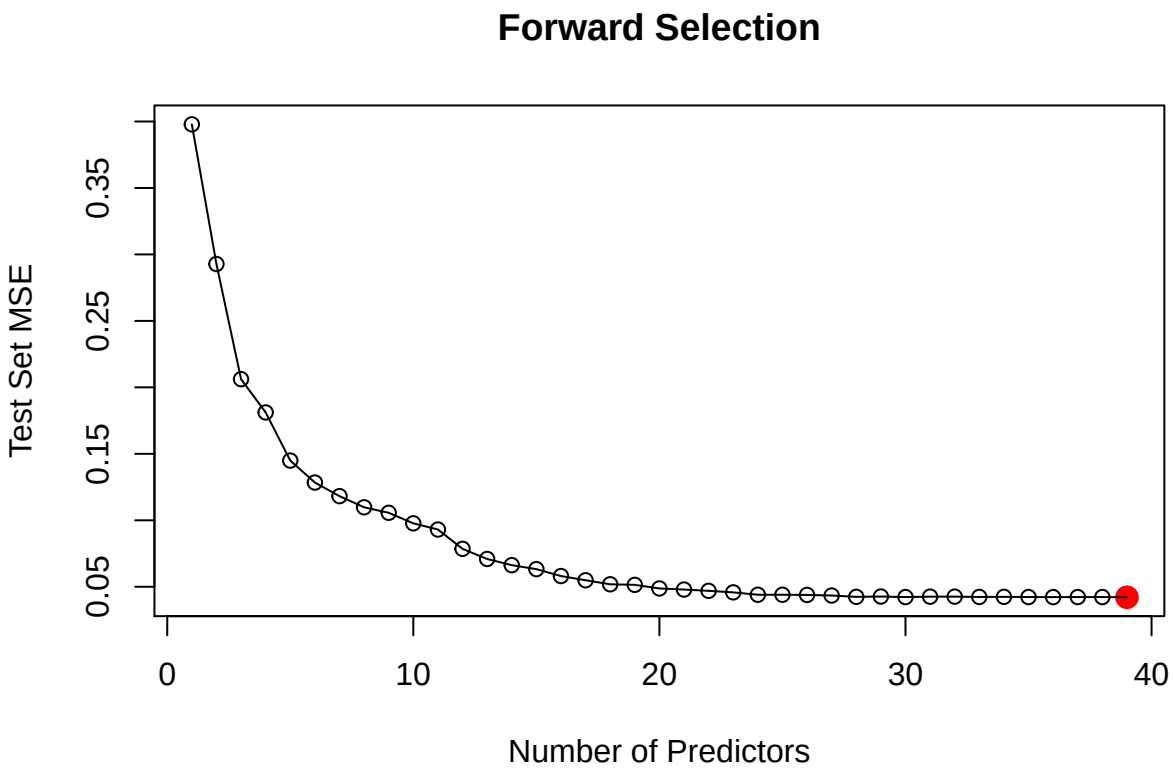
- the Default model returned an MSE of 0.042125 and had 39 number of predictors.
- the AIC (penalized) model returned an MSE of -308.091744 and had 23 number of predictors.

The default best-subset result indicates that all 39 predictors are needed to maximize accuracy. However, one needs to be cautious as the benefit of adding more than ~15 predictors seems to decline. The AIC model selection penalizes these larger models that inherently can predict more variance with the higher degrees of freedom. The AIC model is probably the better model as it is 23 number of predictors (less than the default).

3.2 Backward Deletion and Forward Selection

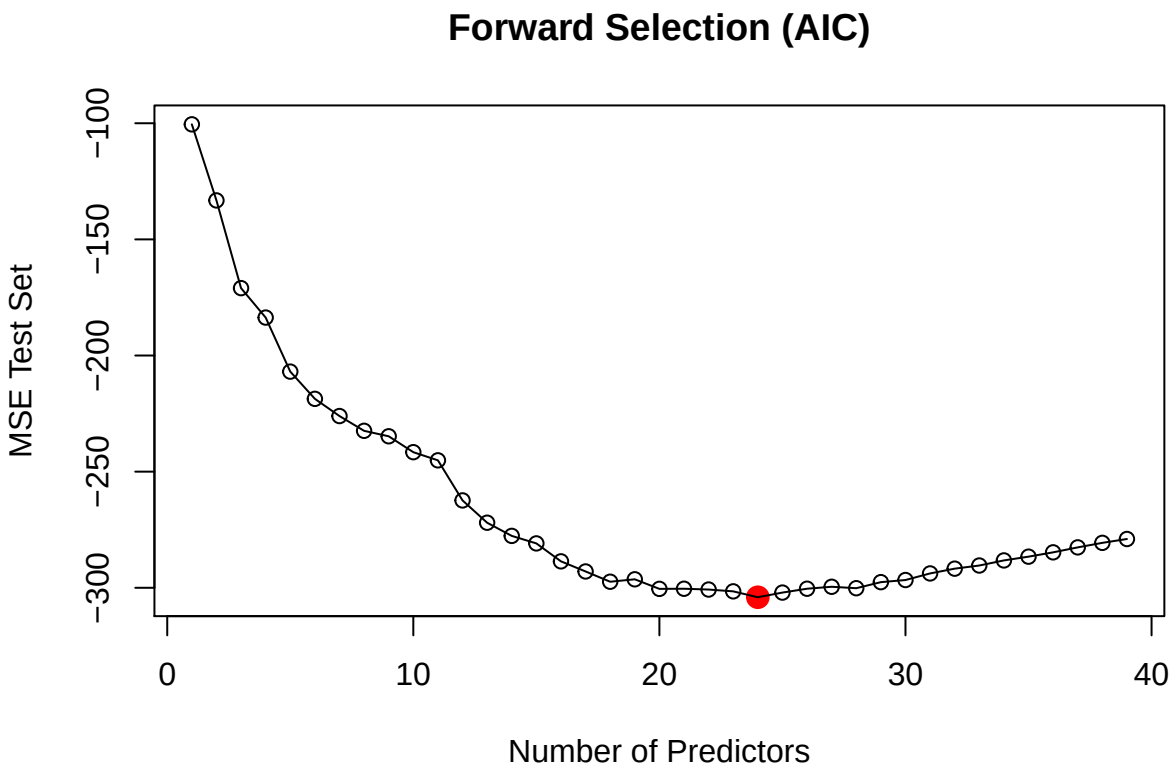
To be honest, after the best-subset, these selection methods are more for our interest and intrigue. We completed these before best-subset initially (as best-subset takes a lot of time to run exhaustively). Therefore, we left these models in. We use the AIC to again penalize models that have many predictors.

```
set.seed(111)


# Forward Selection MLR Method
forward_regression = regsubsets(hf_score ~ ., method = "forward",
                                nvmax = (ncol(hfi_data_combined_train)-1),
                                data = hfi_data_combined_train)
# create test model matrix to easily create the predictions
forward_regression_model_matrix = model.matrix(hf_score ~ ., data = hfi_data_combined_test)
# store MSE and AIC results for the test set
forward_regression_test_mse = rep(-1, (ncol(hfi_data_combined_train)-1))
forward_regression_test_aic = rep(-1, (ncol(hfi_data_combined_train)-1))

for( i in 1:(ncol(hfi_data_combined_train)-1)){
  # AGAIN, using pipes to efficiently create predictions
  coef_i = coef(forward_regression, id = i)
  preds = forward_regression_model_matrix[,names(coef_i)] %*% coef_i
  # create the MSE and then AIC
  forward_regression_test_mse[i] = mean((preds - hfi_data_combined_test$hf_score)^2)
  forward_regression_test_aic[i] = 2*length(coef_i) +
    (nrow(hfi_data_combined_test))*log(forward_regression_test_mse[i], base = 10)
}

# Plot the MSE and AIC results with the minimum point highlighted
plot(x = 1:(ncol(hfi_data_combined_train)-1), y = forward_regression_test_mse,
     main = "Forward Selection", ylab = "Test Set MSE",
     xlab = "Number of Predictors") +
  points(x = which.min(forward_regression_test_mse),
         y = forward_regression_test_mse[which.min(forward_regression_test_mse)],
         col="red", pch=19, cex = 1.5) +
  lines(x = 1:(ncol(hfi_data_combined_train)-1), y = forward_regression_test_mse)
```




```
## integer(0)
```

```
plot(x = 1:(ncol(hfi_data_combined_train)-1), y = forward_regression_test_aic,
     main = "Forward Selection (AIC)", ylab = "MSE Test Set",
     xlab = "Number of Predictors") +
  points(x = which.min(forward_regression_test_aic),
         y = forward_regression_test_aic[which.min(forward_regression_test_aic)],
         col="red", pch=19, cex = 1.5) +
  lines(x = 1:(ncol(hfi_data_combined_train)-1), y = forward_regression_test_aic)
```




```
## integer(0)
```

```
forward_regression_test_mse[which.min(forward_regression_test_mse)]
```

```
## [1] 0.04212529
```

```
forward_regression_test_aic[which.min(forward_regression_test_aic)]
```

```
## [1] -304.0694
```

```
which.min(forward_regression_test_mse)
```

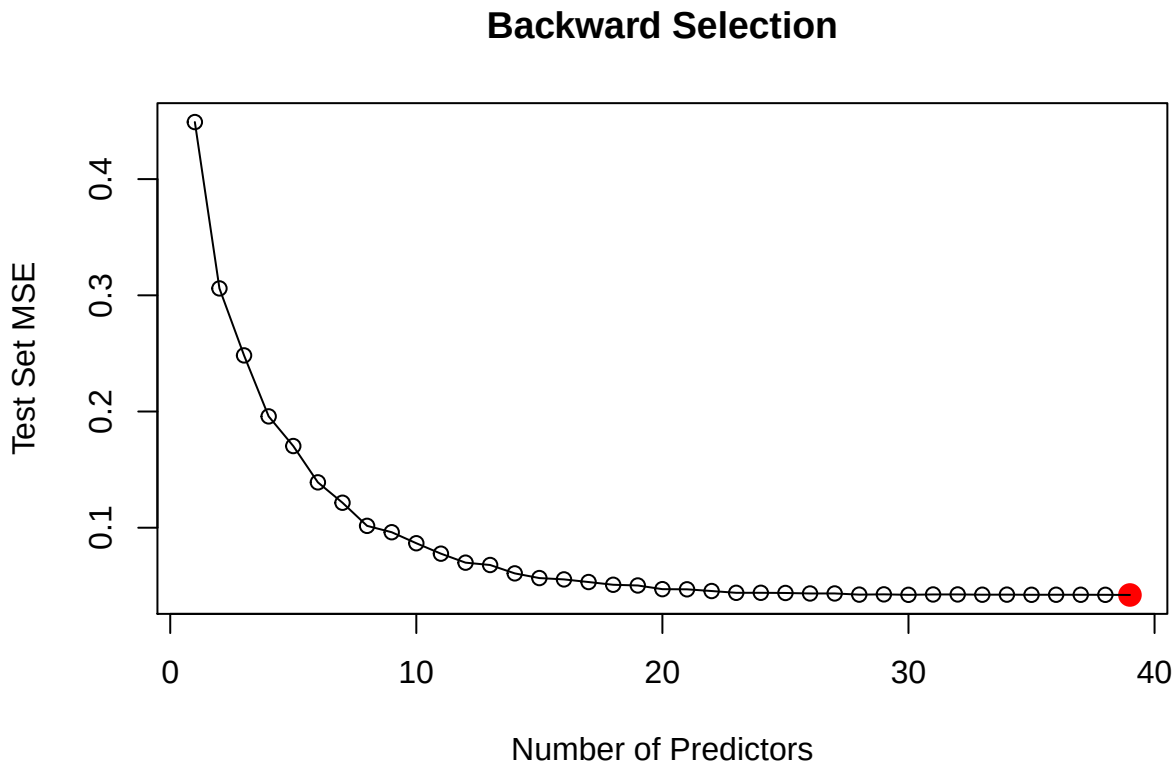
```
## [1] 39
```

```
which.min(forward_regression_test_aic)
```

```
## [1] 24
```

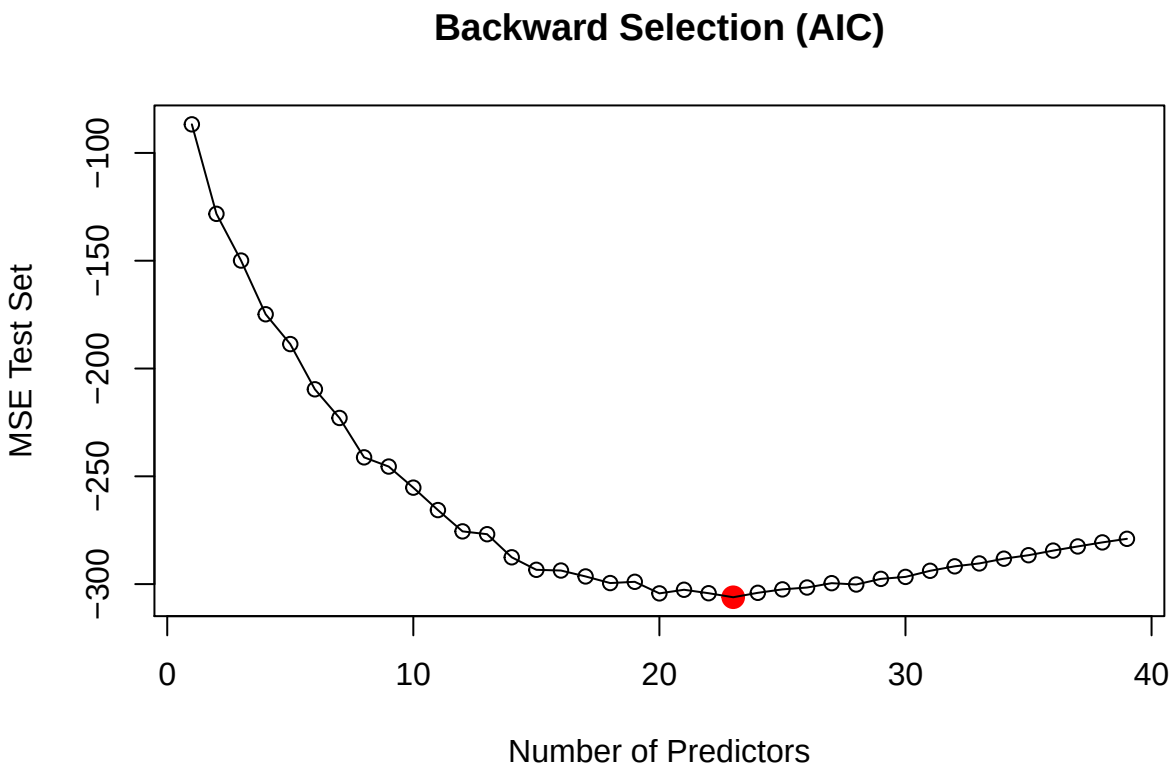
```
# Backward Selection MLR Method
backward_regression = regsubsets(hf_score ~ ., method = "backward",
                                 nvmax = (ncol(hfi_data_combined_train)-1),
                                 data = hfi_data_combined_train)
# create test model matrix to easily create the predictions
backward_regression_model_matrix = model.matrix(hf_score ~ ., data = hfi_data_combined_test)
# store MSE and AIC results for the test set
backward_regression_test_mse = rep(-1, (ncol(hfi_data_combined_train)-1))
backward_regression_test_aic = rep(-1, (ncol(hfi_data_combined_train)-1))

for( i in 1:(ncol(hfi_data_combined_train)-1)){
  # AGAIN, using pipes to efficiently create predictions
```

```
  coef_i = coef(backward_regression, id = i)
  preds = backward_regression_model_matrix[,names(coef_i)] %*% coef_i
  # create the MSE and then AIC
  backward_regression_test_mse[i] = mean((preds - hfi_data_combined_test$hf_score)^2)
  backward_regression_test_aic[i] = 2*length(coef_i) +
    (nrow(hfi_data_combined_test))*log(backward_regression_test_mse[i], base = 10)
}

# Plot the MSE and AIC results with the minimum point highlighted
plot(x = 1:(ncol(hfi_data_combined_train)-1), y = backward_regression_test_mse,
     main = "Backward Selection", ylab = "Test Set MSE",
     xlab = "Number of Predictors") +
  points(x = which.min(backward_regression_test_mse),
         y = backward_regression_test_mse[which.min(backward_regression_test_mse)],
         col="red", pch=19, cex = 1.5) +
  lines(x = 1:(ncol(hfi_data_combined_train)-1), y = backward_regression_test_mse)
```




```
## integer(0)
```

```
plot(x = 1:(ncol(hfi_data_combined_train)-1), y = backward_regression_test_aic,
     main = "Backward Selection (AIC)", ylab = "MSE Test Set",
     xlab = "Number of Predictors") +
  points(x = which.min(backward_regression_test_aic),
         y = backward_regression_test_aic[which.min(backward_regression_test_aic)],
         col="red", pch=19, cex = 1.5) +
  lines(x = 1:(ncol(hfi_data_combined_train)-1), y = backward_regression_test_aic)
```




```
## integer(0)
```

```
backward_regression_test_mse[which.min(backward_regression_test_mse)]
```

```
## [1] 0.04212529
```

```
backward_regression_test_aic[which.min(backward_regression_test_aic)]
```

```
## [1] -306.0917
```

```
which.min(backward_regression_test_mse)
```

```
## [1] 39
```

```
which.min(backward_regression_test_aic)
```

```
## [1] 23
```

The results from forward selection are similar to best-subset selection. The MSE of the default model (no penalizing for more complex models) is 0.042125 with 39 predictors. The AIC model (penalizing for more complexity) resulting in an MSE of -304.069427 with 24 predictors. Overall, forward selection comes close to the findings of best-subset.

The results from the backward selection are similar to the best-subset selection. The MSE of the default model (no penalizing for more complex models) is 0.042125 with 39 predictors. The AIC model (penalizing for more complexity) resulting in an MSE of -306.091744 with 23 predictors. Overall, backward selection comes close to the findings of best-subset.

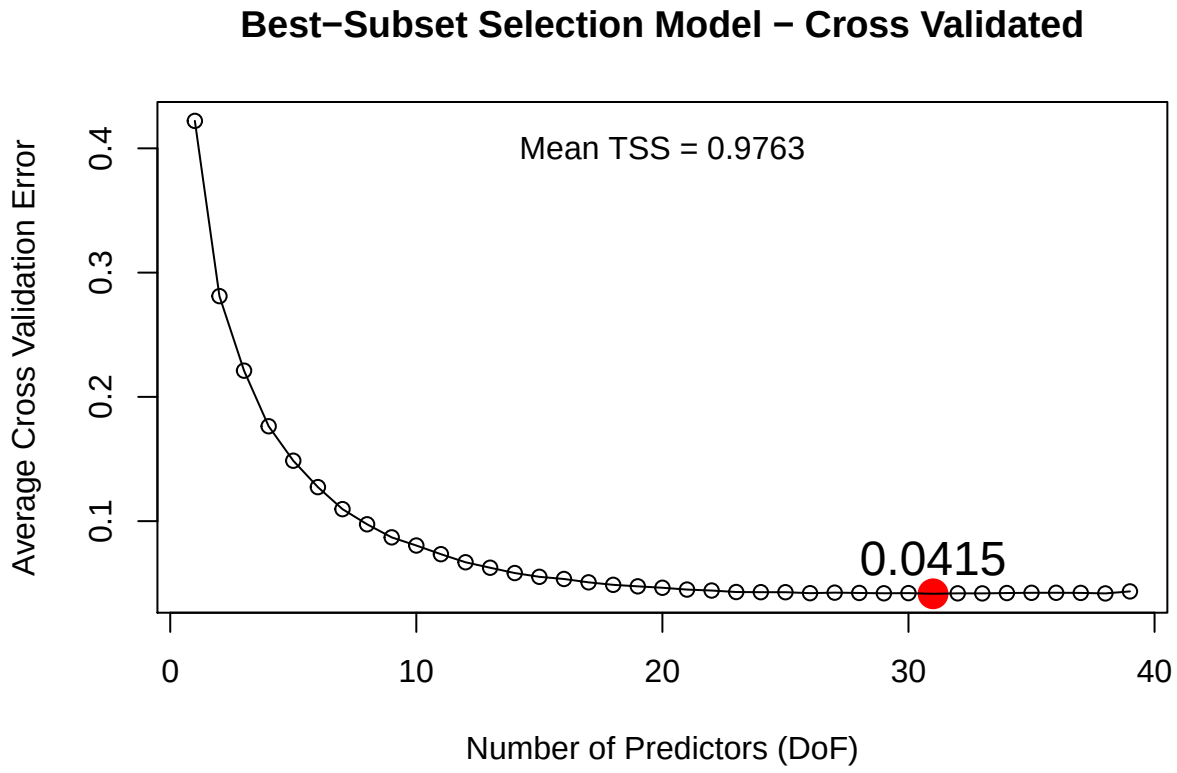
3.3 Cross-Validation and Our Preliminary Findings

3.3.1 Implement Cross-Validation on the MLR Models from Above

Cross-Validation utilizes bootstrapping to estimate the MSE in a more robust way to help avoid overfitting a model. We perform cross-validation using the best-subset, forward selection, and backwards deletion models generated above. We get a more robust error (MSE) estimate and take the best from each set of models before testing against the test set as "new" data.

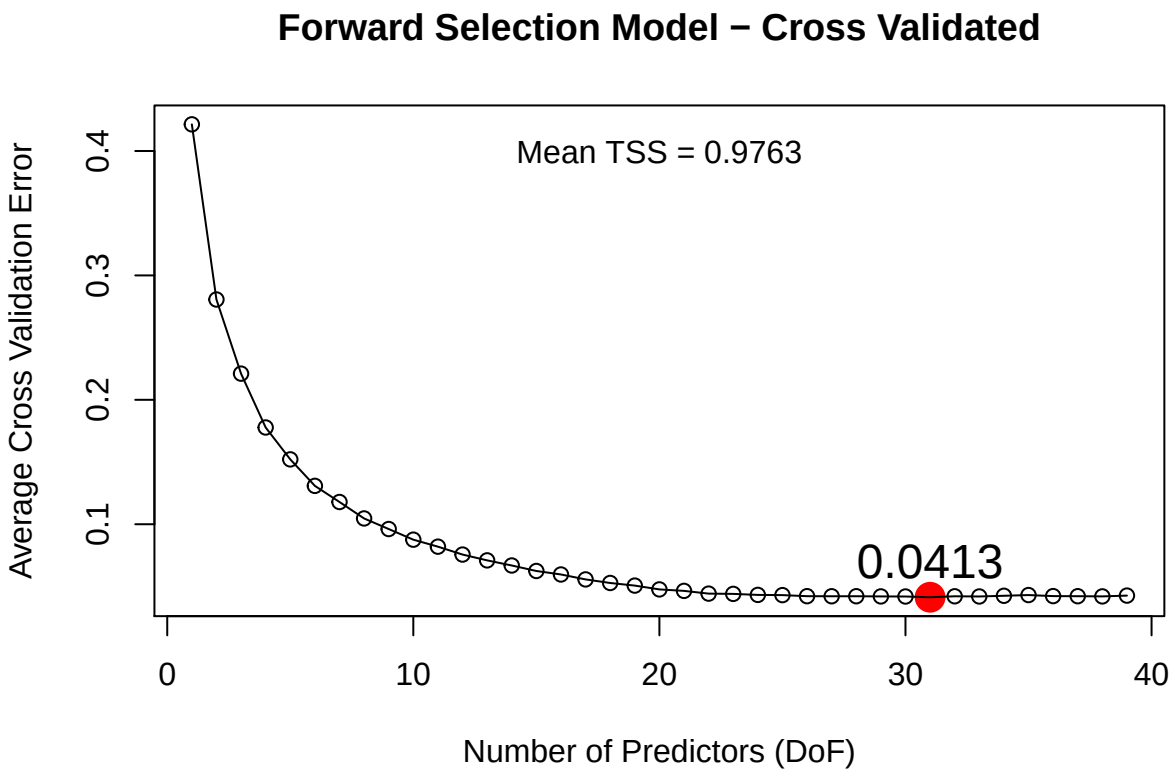
```
set.seed(111)

############################   CROSS-VALIDATION on MLR   ##################
k_folds = 10
par(mfrow=c(1,1))

cv_best_subset_regression_mse = rep(NA, max_predictors_best_subset)
cv_forward_regression_mse = rep(NA, (ncol(hfi_data_combined)-1))
cv_backward_regression_mse = rep(NA, (ncol(hfi_data_combined)-1))

# iterate through all 39 dof for forward, backward and best-subset
for (i in 1:(ncol(hfi_data_combined)-1)){
  # Take the formula for given degree of freedom and set it up for CV
  # Best-Subset Formula; if there is limit on @ predictors, do not calculate
  if( i <= max_predictors_best_subset){
    formula_best_subset_regression = paste( names(coef(best_subset_regression, id = i))[-1],
                                            collapse = " + " )
    formula_best_subset_regression = as.formula( paste("hf_score ~ ", formula_best_subset_regression, s
    cv_best_subset_regression = cv.glm(data = hfi_data_combined_train, K = k_folds,
                                       glmfit = glm(formula = formula_best_subset_regression,
                                                    data = hfi_data_combined_train))
    cv_best_subset_regression_mse[i] = cv_best_subset_regression$delta[1]
  }

  # Forward Selection Formula
  formula_forward_regression = paste(names(coef(forward_regression, id = i))[-1],
                                     collapse = " + " )
  formula_forward_regression = as.formula( paste("hf_score ~ ", formula_forward_regression, sep = ""))
  cv_forward_regression = cv.glm(data = hfi_data_combined_train, K = k_folds,
                                 glmfit = glm(formula = formula_forward_regression,
                                              data = hfi_data_combined_train))
  cv_forward_regression_mse[i] = cv_forward_regression$delta[1]

  # Backward Selection Formula
  formula_backward_regression = paste(names(coef(backward_regression, id = i))[-1],
                                      collapse = " + " )
  formula_backward_regression = as.formula( paste("hf_score ~ ", formula_backward_regression, sep = ""))
  cv_backward_regression = cv.glm(data = hfi_data_combined_train, K = k_folds,
                                  glmfit = glm(formula = formula_backward_regression,
                                               data = hfi_data_combined_train))
  cv_backward_regression_mse[i] = cv_backward_regression$delta[1]
}

# 1 by 3 graph plot
par(mfrow=c(1,1))
# Plots of the CV Errors for all 3 Methods
```

```
plot(cv_best_subset_regression_mse, main="Best-Subset Selection Model - Cross Validated",
     xlab = "Number of Predictors (DoF)", ylab = "Average Cross Validation Error") +
  points(which.min(cv_best_subset_regression_mse), cv_best_subset_regression_mse[which.min(cv_best_subs
       col ="red", cex=2, pch =19) +
  lines(x = 1:(ncol(hfi_data_combined)-1), y = cv_best_subset_regression_mse) +
  text(x=which.min(cv_best_subset_regression_mse), y=min(cv_best_subset_regression_mse),
       labels = paste(round(min(cv_best_subset_regression_mse), 4)), pos=3, cex=1.5) +
  text(x=20, y=max(cv_best_subset_regression_mse),
       labels = paste("Mean TSS =", round(mean_tss_combined, 4)), cex=1, pos=1)
```



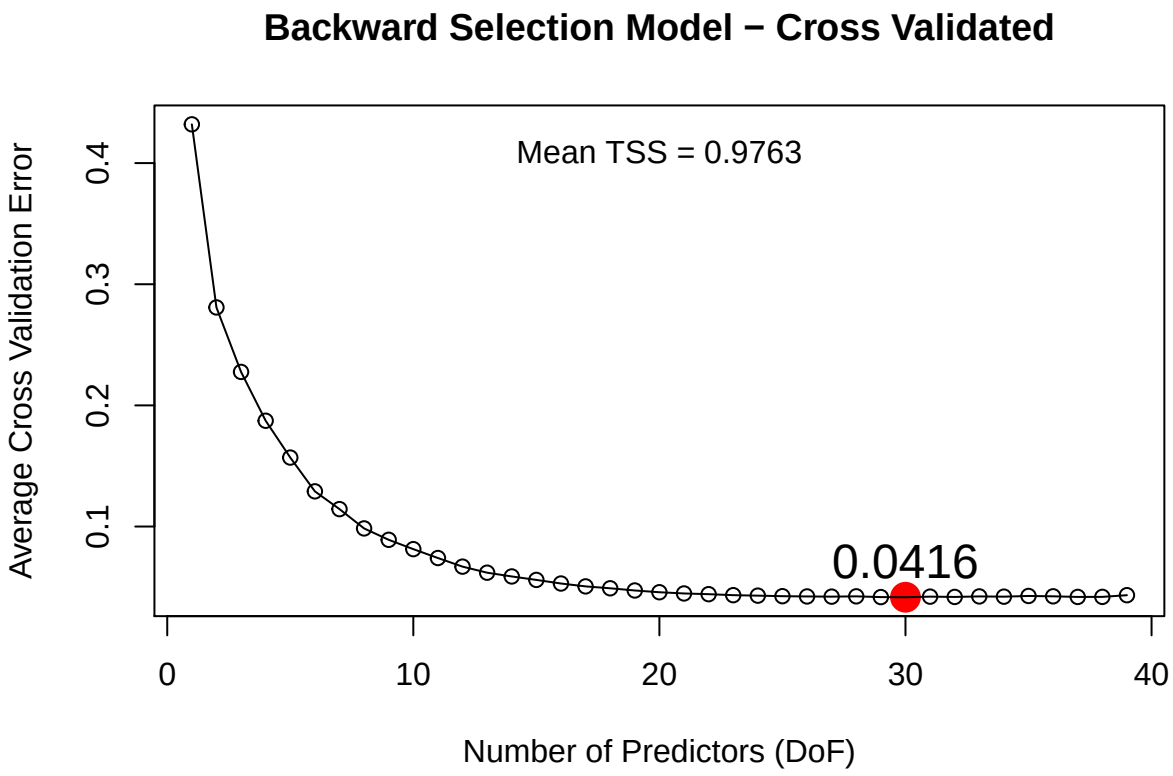

```
## integer(0)
```

```
plot(cv_forward_regression_mse, main = "Forward Selection Model - Cross Validated",
     xlab = "Number of Predictors (DoF)", ylab = "Average Cross Validation Error") +
  points(which.min(cv_forward_regression_mse), cv_forward_regression_mse[which.min(cv_forward_regressio
       col ="red", cex=2, pch =19) +
  lines(x = 1:(ncol(hfi_data_combined)-1), y = cv_forward_regression_mse) +
  text(x=which.min(cv_forward_regression_mse), y=min(cv_forward_regression_mse),
       labels = paste(round(min(cv_forward_regression_mse), 4)), pos=3, cex=1.5) +
  text(x=20, y=max(cv_forward_regression_mse),
       labels = paste("Mean TSS =", round(mean_tss_combined, 4)), cex=1, pos=1)
```




```
## integer(0)
```

```
plot(cv_backward_regression_mse, main = "Backward Selection Model - Cross Validated",
     xlab = "Number of Predictors (DoF)", ylab = "Average Cross Validation Error") +
  points(which.min(cv_backward_regression_mse), cv_backward_regression_mse[which.min(cv_backward_regress
       col ="red", cex=2, pch =19) +
  lines(x = 1:(ncol(hfi_data_combined)-1), y = cv_backward_regression_mse) +
  text(x=which.min(cv_backward_regression_mse), y=min(cv_backward_regression_mse),
       labels = paste(round(min(cv_backward_regression_mse), 4)), pos=3, cex=1.5) +
  text(x=20, y=max(cv_backward_regression_mse),
       labels = paste("Mean TSS =", round(mean_tss_combined, 4)), cex=1, pos=1)
```




```
## integer(0)
```

```
paste("Best-Subset MSE: ", cv_best_subset_regression_mse[which.min(cv_best_subset_regression_mse)], "; N
```

```
## [1] "Best-Subset MSE:  0.0415338167308513 ; Number of predictors:  31"
```

```
paste("Forward Selection MSE: ", cv_forward_regression_mse[which.min(cv_forward_regression_mse)], "; Nu
```

```
## [1] "Forward Selection MSE:  0.0413132950377038 ; Number of predictors:  31"
```

```
paste("Backward Deletion MSE: ", cv_backward_regression_mse[which.min(cv_backward_regression_mse)], "; N
```

```
## [1] "Backward Deletion MSE:  0.0416274461279086 ; Number of predictors:  30"
```

All 3 models have very low cross-validated MSEs around 0.0415 compared to the Mean TSS of 0.999509. Roughly 95.85 % of variance is explained by all 3 best models. That result is incredibly good. This is odd that the results of multiple linear regression deliver such good results with out adding polynomial terms or interaction terms. This model is already useful enough to stop trying the other methods as MLR delivers high accuracy and very good inferencing for discovering the most significant predictors in terms of affecting the human freedom score.

Now we will test the best model for each method (best-subset, backwards, forwards) against new data (the test set) and see which one performs the best. Whichever model performs the best, we will use it (we will prefer a smaller, simpler model if it does nearly as well).

```
set.seed(111)

# recreate the best-subset model with the coefs given from Cross-Validation
# USE THE TRAINING DATA
best_subset_best_regression = lm(as.formula(paste('hf_score ~', paste(names(coef(best_subset_regression
```

```
  cv_best_subset_regression_mse ))[2: which.min(cv_best_subset_regression_mse)],
  collapse = ' + '))), data=hfi_data_combined_train )
yhat = predict(best_subset_best_regression, newdata = hfi_data_combined_test)
final_best_subset_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)



# recreate the forward selection model with the coefs given from Cross-Validation
# USE THE TRAINING DATA
best_forward_regression = lm(as.formula(paste('hf_score ~', paste(names(coef(forward_regression, id = wh
  cv_forward_regression_mse)))[2: which.min(cv_forward_regression_mse)],
  collapse = ' + '))), data=hfi_data_combined_train )
yhat = predict(best_forward_regression, newdata = hfi_data_combined_test)
final_forward_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)



# recreate the best-subset model with the coefs given from Cross-Validation
# USE THE TRAINING DATA
best_backward_regression = lm(as.formula(paste('hf_score ~', paste(names(coef(backward_regression, id =
  cv_backward_regression_mse)))[2: which.min(cv_backward_regression_mse)],
  collapse = ' + '))), data=hfi_data_combined_train )
yhat = predict(best_backward_regression, newdata = hfi_data_combined_test)
final_backward_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)


paste("Mean TSS of Test Set:", round(mean_tss_test, 6))
```

```
## [1] "Mean TSS of Test Set: 0.882886"
```

```
paste("MSE of Best-Subset Model with Test Set:", round(final_best_subset_mse, 6))
```

```
## [1] "MSE of Best-Subset Model with Test Set: 0.042611"
```

```
paste("MSE of Forward Selection Model with Test Set:", round(final_forward_mse, 6))
```

```
## [1] "MSE of Forward Selection Model with Test Set: 0.042611"
```

```
paste("MSE of Backward Deletion Model with Test Set:", round(final_backward_mse, 6))
```

```
## [1] "MSE of Backward Deletion Model with Test Set: 0.042272"
```

```
# I found that the Best-Subset and Forward Selection maybe the same model
# this does depend on the random seed most likely
a = sort(names(coef(forward_regression, id = which.min(cv_forward_regression_mse)))[2: which.min(cv_forw
b = sort(names(coef(forward_regression, id = which.min(cv_best_subset_regression_mse)))[2: which.min(cv_
paste("Are Best-Subset and Forward Selection the Same Models? ....", all(a == b))
```

```
## [1] "Are Best-Subset and Forward Selection the Same Models? .... TRUE"
```

We performed cross validation on each model for a given number of predictors with the results found previously for best-subset, backward deletion, and forward selection. The cross-validation helps to add robustness to the estimated MSE, preventing overfitting of the training dataset. It appears that the forward selection and best-subset have the same exact models, hence the identical test MSEs. Ironically, the backward deletion method derives the best model of the three, not best-subset. Granted, the difference is minuscule (like less than 1%). However, we will use the best-subset and forward selection models (as they are the same, assuming the random seed stays the same). This choice is made because the best-subset is exhaustive and the fact that

2 methods returned the same exact linear models gives us more reason to believe this model is better (though both are incredibly accurate and practically identical in R-Squared or correlation).

3.3.2 Our Preliminary Analysis So Far

Here are the coefficients of the 3 models (2 are identical).

```
paste("Backwards Deletion Method Coefficients: ")
```

```
## [1] "Backwards Deletion Method Coefficients: "
```

```
paste(names(coef(backward_regression, id=which.min(cv_backward_regression_mse))), collapse = ', ' )
```

```
## [1] "(Intercept), pf_ss_homicide, pf_ss_disappearances_disap, pf_ss_disappearances_violent, pf_moveme
```

```
paste("")
```

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

```
paste("Best-Subset and Forward Selection Methods Coefficients (identical): ")
```

```
## [1] "Best-Subset and Forward Selection Methods Coefficients (identical): "
```

```
paste(names(coef(forward_regression, id = which.min(cv_forward_regression_mse))), collapse = ', ' )
```

```
## [1] "(Intercept), pf_ss_homicide, pf_ss_disappearances_disap, pf_ss_disappearances_violent, pf_moveme
```

```
paste("")
```

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

```
a = sort(names(coef(forward_regression, id = which.min(cv_forward_regression_mse))))
b = sort(names(coef(backward_regression, id=which.min(cv_backward_regression_mse))))

same_predictors = 0
diff_predictors = 0
different_preds = vector()
for (i in 1:length(a)){
  if (a[i] %in% b ){
    same_predictors = same_predictors + 1
  } else{
    diff_predictors = diff_predictors + 1
    different_preds = append(different_preds, a[i])
  }
}

paste("Number of Same Predictors in 2 Models:", same_predictors )
```

```
## [1] "Number of Same Predictors in 2 Models: 31"
```

```
paste("Number of Different Predictors in 2 Models:", diff_predictors )
```

```
## [1] "Number of Different Predictors in 2 Models: 1"
```

```
paste("The different predictors are:", different_preds)
```

```
## [1] "The different predictors are: ef_trade_regulatory_compliance"
```

The coefficients in the 3 different models are nearly identical, meaning all 3 models are nearly the same (2 actually are). There is only 1 different predictors between the models, which may be due to the different sizes of the models. The forward selection's model has 31 predictors within the model, while backwards deletion's

model has 30. Therefore, the near identical models means any of the three can be chosen. We will choose best-subset/forward selection.

Now, we will analyze the significance of the predictors within the model using T-Tests as MLR gives us the ability to easily infer the importance or significance of a predictor.

```
summary(best_forward_regression)
```

```
##
## Call:
## lm(formula = as.formula(paste("hf_score ~", paste(names(coef(forward_regression,
##     id = which.min(cv_forward_regression_mse)))[2:which.min(cv_forward_regression_mse)],
##     collapse = " + "))), data = hfi_data_combined_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6707 -0.1308  0.0113  0.1326  0.8244
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      0.331258   0.122962   2.694 0.007177 **
## pf_ss_homicide                   0.054883   0.002787  19.690  < 2e-16 ***
## pf_ss_disappearances_disap       0.007723   0.002589   2.983 0.002923 **
## pf_ss_disappearances_violent     0.014433   0.004464   3.233 0.001264 **
## pf_movement_domestic             0.018801   0.002422   7.761 2.05e-14 ***
## pf_movement_foreign              0.016882   0.002260   7.468 1.75e-13 ***
## pf_religion_harassment           0.024979   0.009287   2.690 0.007269 **
## pf_religion_restrictions         0.023538   0.005089   4.625 4.23e-06 ***
## pf_expression_influence          0.043360   0.007257   5.975 3.19e-09 ***
## pf_expression_control            0.049226   0.008606   5.720 1.40e-08 ***
## pf_identity_sex_male             0.021200   0.002327   9.110  < 2e-16 ***
## pf_identity_sex_female           0.016234   0.002574   6.307 4.25e-10 ***
## ef_government_consumption        0.039983   0.003854  10.375  < 2e-16 ***
## ef_legal_courts                  0.075238   0.005579  13.486  < 2e-16 ***
## ef_legal_military                0.046731   0.004002  11.677  < 2e-16 ***
## ef_legal_enforcement             0.052020   0.005071  10.259  < 2e-16 ***
## ef_legal_gender                  1.034573   0.066013  15.672  < 2e-16 ***
## ef_money_growth                  0.033803   0.007150   4.728 2.59e-06 ***
## ef_money_sd                      0.035833   0.005329   6.724 2.95e-11 ***
## ef_money_inflation               0.030944   0.006656   4.649 3.77e-06 ***
## ef_money_currency                0.026736   0.002373  11.268  < 2e-16 ***
## ef_trade_tariffs_mean            0.057275   0.009305   6.155 1.08e-09 ***
## ef_trade_regulatory_compliance   0.011359   0.008203   1.385 0.166453
## ef_trade_regulatory              0.044053   0.013172   3.344 0.000855 ***
## ef_trade_black                   0.010797   0.006479   1.667 0.095919 .
## ef_trade_movement_capital        0.015497   0.003179   4.874 1.27e-06 ***
## ef_trade_movement_visit          0.018729   0.002293   8.168 9.30e-16 ***
## ef_regulation_credit_private     0.006343   0.004115   1.542 0.123486
## ef_regulation_credit             0.037836   0.007988   4.737 2.48e-06 ***
## ef_regulation_labor_hours        0.013880   0.003554   3.905 0.000100 ***
## ef_regulation_labor_conscription 0.005521   0.001745   3.164 0.001604 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2009 on 1013 degrees of freedom
```

```
## Multiple R-squared:  0.9608, Adjusted R-squared:  0.9597
## F-statistic:   828 on 30 and 1013 DF,  p-value: < 2.2e-16
```

The predictors that seem most significant (3 asterisks) are:

- pf_ss_homicide: a number based on the number of homicides within the country. More homicides means less human freedom as life is being taken away unqillingly.
- pf_movement_domestic: The freedom of citizens to move around the country at their will. Countries that prevent citizens from easily moving from city to city will tend to have less human freedom.
- pf_movement_foreign: the same as the above, but for foreigners' movement within the country.
- pf_religion_restrictions: the freedom to practice one's religion. The more lenient countries are with religion, the greater the human freedom tends to be.
- pf_expression_influence, pf_expression_control: the degree that one's expression (like content of meetings or protests and influence over media) is controlled by the country. Country that allows basically any form of expression will tend to have higher human freedom overall.
- pf_identity_sex_male, pf_identity_sex_female: the freedom to identify and have equal legal protections based on sex. The more equal a country treats men and women, the human freedom tends to be greater.
- ef_government_consumption: the amount of money the government spends. The more the government spends, the more free the countries are. This runs counter to the libertarian ideal that CATO follows.
- ef_legal_courts, ef_legal_military, ef_legal_enforcement, ef_legal_gender: the equality within the judicial system for courts (the amount, availability, and impartiality), the fairness towards the military legal issues, the fairness that laws are enforced, and equality of laws between genders. The more fair and equal and just the judicial system is, the greater the human freedom tends to be.
- ef_money_growth, ef_money_sd, ef_money_inflation, ef_money_currency: The degree the monetary system follows libertarian ideals. AKA, no manipulation of currency or of the money supply for artificial money generation. These variables are quite murky, so beware. I am not sure how they calculate this and how accurate their data is. Additionally, the USA has artificially created *a lot* of money and therefore they should have poor scores, but the USA does not. However, China does as they publicly (kinda) manipulate currency. Basically, I would take these predictors with a ground of salt. The less the government messes with the money supply, interest rates, and monetary policy, the greater the human freedom tends to be.
- ef_trade_tariffs_mean, ef_trade_regulatory: the mean of the trade tariffs and regulation on trade (often affecting trade out-of-country). The greater the tariffs, the less economic freedom according to libertarian views. If the tariffs are small, the human freedom tends to be greater.
- ef_trade_movement_capital, ef_trade_movement_visit: how easily capital can be moved around the country for citizens and visitors (foreigners). The easier money/capital can be moved, then the human freedom tends to be greater.
- ef_regulation_credit, ef_regulation_labor_hours: how well regulated the credit sector is and how much time is spent on enforcing regulation. The CATO institute believes that the credit industry needs to have some oversight to prevent bad actors from taking advantage from no regulation. However, they also will there was very little time spend on enforcing regulation. Therefore, this combo of predictors is again wierd. It is hard to draw many conclusions from it.

The predictors we are most interested in are: pf_ss_homicide, pf_movement_domestic, pf_movement_foreign, pf_religion_restrictions, pf_expression_influence, pf_expression_control, ef_legal_courts, ef_legal_military, ef_legal_enforcement, ef_legal_gender, ef_money_inflation (how well they manage inflation), ef_trade_regulatory, and ef_trade_movement_visit for a total of 12 predictors. The resulting MLR model is:

```
set.seed(111)
simple_mlr_model = lm(hf_score ~ pf_ss_homicide + pf_movement_domestic + pf_movement_foreign + pf_relig
                        pf_expression_control + ef_legal_courts + ef_legal_military + ef_legal_enforceme
                        ef_trade_regulatory + ef_trade_movement_visit, data = hfi_data_combined_train)
yhat = predict(simple_mlr_model, newdata = hfi_data_combined_test)
simple_mlr_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

paste("The Mean TSS of the Test Set:", round(mean_tss_test, 6))
```

```
## [1] "The Mean TSS of the Test Set: 0.882886"
```

```
paste("The Test MSE of the Simple MLR Model:", round(simple_mlr_test_mse, 6))
```

```
## [1] "The Test MSE of the Simple MLR Model: 0.114887"
```

```
paste("The R-Squared of the Simple MLR Model:", round(1 - simple_mlr_test_mse / mean_tss_test, 6))
```

```
## [1] "The R-Squared of the Simple MLR Model: 0.869874"
```

The resulting Test MSE is 0.114887 which is worse than the 30+ predictor models above, but still explains roughly 86.99% of the variance within the dataset. The results are preliminary, but they do seem to indicate the human freedom index could be estimated with substantially fewer predictors than the 39 in the entire dataset. We wil return to this at the end of the analysis.

3.4 Dimension Reduction Methods:

We performed additional methods to try and reduce the number of predictors within the linear models. The techniques utilized were Ridge, Lasso, principle component analysis (PCA), and partial least squares (PLS). Due to the high accuracy of the previous models, we will only briefly analyze these findings. We strongly believe the model should not become overly complex as we will lose some inferencing ability with some of the following techniques (PCA, PLS).

3.4.1 Ridge

```
# simply used to get same results regardless of run. Methods should be robust to the seed changing, tho
set.seed(111)

#  RIDGE REGRESSION METHOD
# grid holds the range of LAMBDA values we use

grid = 10^seq(10,-2, length = 1000)
regression_model_train = model.matrix(hf_score ~ ., data = hfi_data_combined_train)[,-1]
regression_model_test = model.matrix(hf_score ~ ., data = hfi_data_combined_test)[,-1]

# Perform Cross-Validation and test on the test set to get MSE's
# Get the best lambda from CV
cv_ridge_regression = cv.glmnet(regression_model_train, hfi_data_combined_train$hf_score,
                                alpha = 0, lambda = grid, thresh = 1e-12)
# Get the single, best model with CV best lambda
best_ridge_model = glmnet(regression_model_train, hfi_data_combined_train$hf_score,
                          alpha = 0, lambda = cv_ridge_regression$lambda.min, thresh = 1e-12)
# Store the best Lambda, and get MSE by predicting with test set
best_lambda_ridge <- cv_ridge_regression$lambda.min
ridge_model_predictions = predict(best_ridge_model,
                                  newx = regression_model_test,
```

```
                                  s = cv_ridge_regression$lambda.min)
best_ridge_model_mse = mean((hfi_data_combined_test$hf_score - ridge_model_predictions)^2)

best_lambda_ridge
```

```
## [1] 0.03669142
```

```
best_ridge_model$beta
```

```
## 39 x 1 sparse Matrix of class "dgCMatrix"
##                                          s0
## pf_ss_homicide                    0.048143239
## pf_ss_disappearances_disap        0.009107639
## pf_ss_disappearances_violent      0.010751955
## pf_ss_disappearances_fatalities   0.012108566
## pf_ss_disappearances_injuries    -0.008094916
## pf_movement_domestic              0.018365280
## pf_movement_foreign               0.016600531
## pf_religion_harassment            0.022107300
## pf_religion_restrictions          0.026782092
## pf_expression_killed              0.004940367
## pf_expression_jailed              0.005380067
## pf_expression_influence           0.040909831
## pf_expression_control             0.050735317
## pf_identity_sex_male              0.020631078
## pf_identity_sex_female            0.017276297
## ef_government_consumption         0.034299065
## ef_legal_courts                   0.067955677
## ef_legal_military                 0.040888903
## ef_legal_enforcement              0.049533817
## ef_legal_gender                   0.980300442
## ef_money_growth                   0.029226572
## ef_money_sd                       0.039288578
## ef_money_inflation                0.027941977
## ef_money_currency                 0.023809489
## ef_trade_tariffs_mean             0.047530346
## ef_trade_tariffs_sd              -0.007878708
## ef_trade_tariffs                  0.023060196
## ef_trade_regulatory_compliance    0.014764128
## ef_trade_regulatory               0.037602317
## ef_trade_black                    0.012040169
## ef_trade_movement_capital         0.017686057
## ef_trade_movement_visit           0.016960368
## ef_regulation_credit_private      0.008702898
## ef_regulation_credit              0.033870605
## ef_regulation_labor_minwage       0.004212930
## ef_regulation_labor_hours         0.013373527
## ef_regulation_labor_conscription  0.005972801
## ef_regulation_business_start      0.024265085
## ef_regulation_business_compliance 0.004767761
```

```
mean_tss_test
```

```
## [1] 0.882886
```

```
best_ridge_model_mse
```

```
## [1] 0.04268541
```

Using the 10-fold cross validated best lamdba of about 0.036691, in the Ridge model all 39 variables remain in the model.The test MSE rises to 0.042685. The result delivers a model with no predictor removed no improvement to the test MSE of the MLR (the test MSE was 0.042611). The Ridge model fails to simplify the existing model.

3.4.2 Lasso

```
set.seed(111)

# LASSO REGRESSION
# perform same thing as RIDGE but with alpha==1; get MSE's
cv_lasso_regression = cv.glmnet(regression_model_train, hfi_data_combined_train$hf_score,
                                alpha = 1, lambda = grid, thresh = 1e-12)
best_lasso_regression = glmnet(regression_model_train, hfi_data_combined_train$hf_score,
                               alpha = 1, lambda = cv_lasso_regression$lambda.min, thresh = 1e-12)
best_lambda_lasso <- cv_lasso_regression$lambda.min
lasso_model_predictions = predict(best_lasso_regression,
                                  newx = regression_model_test,
                                  s = cv_lasso_regression$lambda.min)
best_lasso_model_mse = mean((hfi_data_combined_test$hf_score - lasso_model_predictions)^2)

# this code counts number of parameters that were set to 0
# This is the difference between RIDGE and LASSO
#     RIDGE: lowers the Betas to reduce degree of Beta terms
#     LASSO: lowers the Betas to 0 to remove completely from model
sum_betas_0s = 0
for( i in 1:length(best_lasso_regression$beta[,1])){
  if(best_lasso_regression$beta[i,1]==0 ){
    sum_betas_0s = sum_betas_0s + 1
  }
}

sum_betas_0s
```

```
## [1] 4
```

```
best_lambda_lasso
```

```
## [1] 0.01
```

```
mean_tss_test
```

```
## [1] 0.882886
```

```
best_lasso_model_mse
```

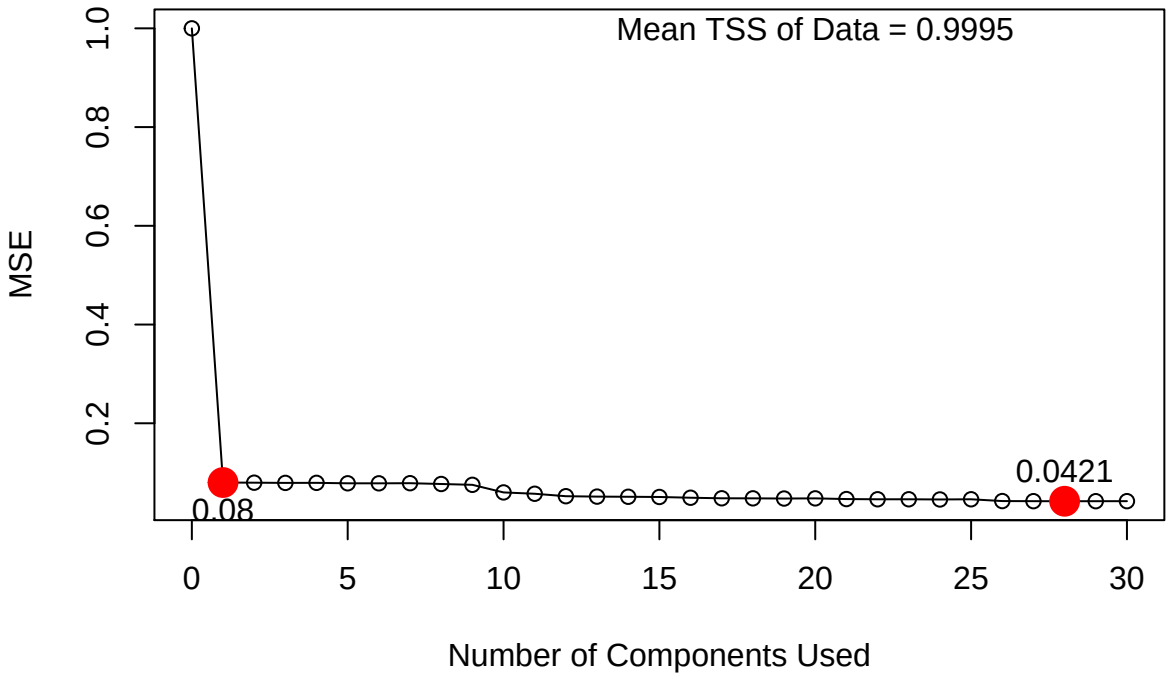
```
## [1] 0.04306138
```

Using the 10-fold cross validated best lambda of 0.01, the resulting Lasso is a better model than Ridge because it removes several predictors. This Lasso model removes 4 predictors, which is somewhat underwhelming considering we have 39 predictors. The test MSE of 0.043061 is slightly worse than Ridge (0.042685). Therefore, both the Ridge and Lasso models do not help us significantly as both models are still more complex

than the forward selection model, backward deletion model, and best-subset model, yet fail to deliver better results.

3.4.3 PCR: Principle Component Reduction

```
set.seed(111)

### DIMENSION REDUCTION METHODS ###
# Principle Component Reduction (PCR)
pcr_model = pcr(as.formula(paste("hf_score ~", paste(names(coef(forward_regression, id = which.min(cv_f
                                 collapse = " + "))), data = hfi_data_combined_train, scale = TRUE, vali
plot(x=0:pcr_model$ncomp,
     y=c(pcr_model$validation$PRESS0, pcr_model$validation$PRESS )/pcr_model$validation$PRESS0,
     main = "Validation Plot for Principle Component Reduction (PCR) Analysis",
     ylab = "MSE", xlab = "Number of Components Used") +
  lines(x=0:pcr_model$ncomp,
        y=c(pcr_model$validation$PRESS0, pcr_model$validation$PRESS )/pcr_model$validation$PRESS0) +
  points(x = which.min(pcr_model$validation$PRESS),
         y = min(pcr_model$validation$PRESS)/pcr_model$validation$PRESS0,
         col ="red", cex=2, pch =19) +
  text(x = which.min(pcr_model$validation$PRESS),
       y = min(pcr_model$validation$PRESS)/pcr_model$validation$PRESS0,
       labels=round(min(pcr_model$validation$PRESS)/pcr_model$validation$PRESS0, 4), pos = 3) +
  points(x = 1, y = pcr_model$validation$PRESS[1]/pcr_model$validation$PRESS0,
         col ="red", cex=2, pch =19) +
  text(x = 1, y = pcr_model$validation$PRESS[1]/pcr_model$validation$PRESS0,
       labels = round(pcr_model$validation$PRESS[1]/pcr_model$validation$PRESS0, 4), pos = 1) +
  text(x=20, y=1, labels = paste("Mean TSS of Data =", round(mean_tss_train,4)))
```




```
## integer(0)
```

```
yhat = predict(pcr_model, newdata = hfi_data_combined_test, ncomp = which.min(pcr_model$validation$PRESS
best_pcr_model_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

yhat = predict(pcr_model, newdata = hfi_data_combined_test, ncomp = 1)
pcr_model_1_comp_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

yhat = predict(pcr_model, newdata = hfi_data_combined_test, ncomp = 9)
pcr_model_9_comp_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

yhat = predict(pcr_model, newdata = hfi_data_combined_test, ncomp = 10)
pcr_model_10_comp_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)



paste("The Mean TSS for the Test Set:", round(mean_tss_test, 6))
```

```
## [1] "The Mean TSS for the Test Set: 0.882886"
```

```
paste("The Test MSE for the Forward Selection Model:", round(final_forward_mse, 6))
```

```
## [1] "The Test MSE for the Forward Selection Model: 0.042611"
```

```
paste("The Test MSE for the Best PCR Model with",which.min(pcr_model$validation$PRESS), "Components:", 
```

```
## [1] "The Test MSE for the Best PCR Model with 28 Components: 0.042747"
```

```
paste("The Test MSE for the Best PCR Model with 1 Component:", round(pcr_model_1_comp_test_mse, 6))
```

```
## [1] "The Test MSE for the Best PCR Model with 1 Component: 0.087674"
```

```
paste("The Test MSE for the Best PCR Model with 11 Components:", round(pcr_model_9_comp_test_mse, 6))
```

```
## [1] "The Test MSE for the Best PCR Model with 11 Components: 0.080525"
```

```
paste("The Test MSE for the Best PCR Model with 10 Components:", round(pcr_model_10_comp_test_mse, 6))
```

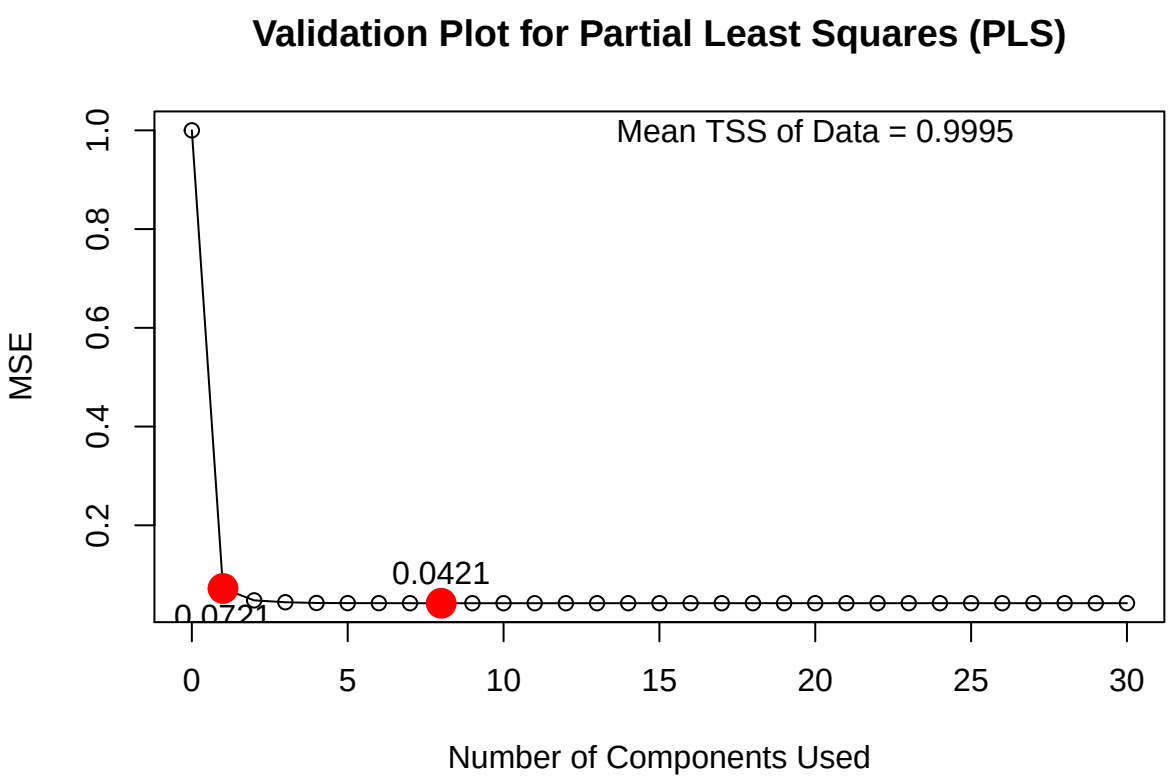
```
## [1] "The Test MSE for the Best PCR Model with 10 Components: 0.063762"
```

Using 10-fold cross validation, the lowest average root MSE is the one corresponding to M=28 components. The resulting PCR gives a test MSE of 0.042747, and is slightly worse than the linear models shown earlier. Notably, the validation plot shows that after the first component is added, only marginal benefits are gained from adding more components. After running the PCR with 1 component results in a test MSE of 0.087674. The test MSE is still very accurate, but not improved compared to the forward selection MLR test MSE of 0.042611.

A large issue with PCR, is the predictors are transformed into a combination of each other to reduce the dimensionality. However, this does not mean the model is simpler. It just means there is 1 predictor for a pair of predictors. Additionally, the transformation of the predictors distort our understanding of the significance and importance of specific predictors on human freedom. Therefore, we prefer the forward selection and best-subset models from the MLR section above. More accuracy and more inferencing. Literally, the MLR is strictly better.

3.4.4 PLS

```
set.seed(111)

# Partial Least Squares (PLS) Model
pls_model = plsr(as.formula(paste("hf_score ~", paste(names(coef(forward_regression, id = which.min(cv_
                                  collapse = " + "))),  data = hfi_data_combined_train, scale = TRUE,
                 validation = "CV")
plot(x=0:pls_model$ncomp,
     y=c(pls_model$validation$PRESS0, pls_model$validation$PRESS )/pls_model$validation$PRESS0,
     main = "Validation Plot for Partial Least Squares (PLS)",
     ylab = "MSE", xlab = "Number of Components Used") +
  lines(x=0:pls_model$ncomp,
        y=c(pls_model$validation$PRESS0, pls_model$validation$PRESS )/pls_model$validation$PRESS0) +
  points(x = which.min(pls_model$validation$PRESS),
         y = min(pls_model$validation$PRESS)/pls_model$validation$PRESS0,
         col ="red", cex=2, pch =19) +
  text(x = which.min(pls_model$validation$PRESS),
       y = min(pls_model$validation$PRESS)/pls_model$validation$PRESS0,
       labels=round(min(pls_model$validation$PRESS)/pls_model$validation$PRESS0, 4), pos = 3) +
  points(x = 1, y = pls_model$validation$PRESS[1]/pls_model$validation$PRESS0,
         col ="red", cex=2, pch =19) +
  text(x = 1, y = pls_model$validation$PRESS[1]/pls_model$validation$PRESS0,
       labels = round(pls_model$validation$PRESS[1]/pls_model$validation$PRESS0, 4), pos = 1) +
  text(x=20, y=1, labels = paste("Mean TSS of Data =", round(mean_tss_train,4)))
```




```
## integer(0)
```

```
yhat = predict(pls_model, newdata = hfi_data_combined_test, ncomp = which.min(pls_model$validation$PRESS
best_pls_model_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

yhat = predict(pls_model, newdata = hfi_data_combined_test, ncomp = 1)
pls_model_1_comp_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

yhat = predict(pls_model, newdata = hfi_data_combined_test, ncomp = 2)
pls_model_2_comp_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

paste("The Mean TSS for the Test Set:", round(mean_tss_test, 6))
```

```
## [1] "The Mean TSS for the Test Set: 0.882886"
```

```
paste("The Test MSE for the Forward Selection Model:", round(final_forward_mse, 6))
```

```
## [1] "The Test MSE for the Forward Selection Model: 0.042611"
```

```
paste("The Test MSE for the Best PLS Model with",which.min(pls_model$validation$PRESS), "Components:", r
```

```
## [1] "The Test MSE for the Best PLS Model with 8 Components: 0.042749"
```

```
paste("The Test MSE for the Best PLS Model with 1 Component:", round(pls_model_1_comp_test_mse, 6))
```

```
## [1] "The Test MSE for the Best PLS Model with 1 Component: 0.078617"
```

```
paste("The Test MSE for the Best PLS Model with 2 Components:", round(pls_model_2_comp_test_mse, 6))
```

```
## [1] "The Test MSE for the Best PLS Model with 2 Components: 0.048627"
```

The PLS method delivers very accurate MSE after the first component, then only marginal benefits afterwards. It seems after the ninth component, the models no longer improve the test MSE results. Again, the transformations of the predictors to form these components leads to less inferencing power (especially compared to the equally if not superior linear models). Therefore, we decide that the linear models are still the best models. The results are practically identical to the PCR, not useful enough to use nor better than the existing MLR models from previous sections.

3.4.5 Dimension Reduction Findings

These methods failed to simplify the model in a meaningful way. Again, it is important to highlight that the accuracy of the MLR models from forward, backward, best-subset, and cross-validated were all highly accurate. These models explained over 90% (and often 95%) of variance, therefore the reduction is not really necessary as the model benefits from having all the variables included. If the MLR was not as effective as a predictor and inferencing tool, we would look into a more stringent analysis using these dimension reduction techniques.

While some methods deliver results on par with linear models, the linear models are superior for inferencing. We are very interested in the inferencing capabilities of the linear models to draw conclusions about which predictors are the most significant and affect human freedom score the most.

We create a smaller MLR model at the end of the project using hand-picked predictors based on all the methods for inferencing.

We have already done this partially already, but there maybe changes to the variables chosen by the end of the analysis.

4. Non-Linear Methods

We have focused on linear methods of analyzing trends in data. Now we will pivot to some non-linear methods like polynomial regression, splines, Generalized Additive Models (GAMs), and K-Nearest Neighbors (KNN).

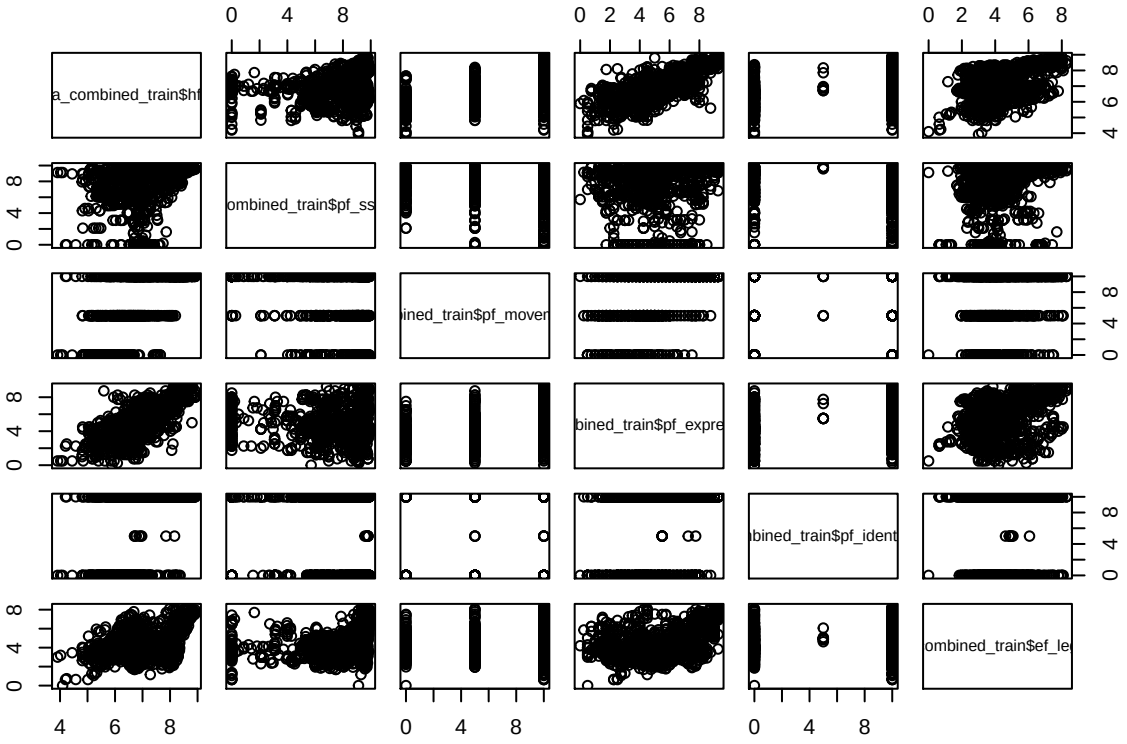
4.1 Polynomial Regression

This regression using polynomial terms is not necessary. We do not really expect polynomial trends within the data and these polynomial curves tend to add variance and less interpretability to the end results. However, we still try the polynomial curves to see if they do offer insights to the data.

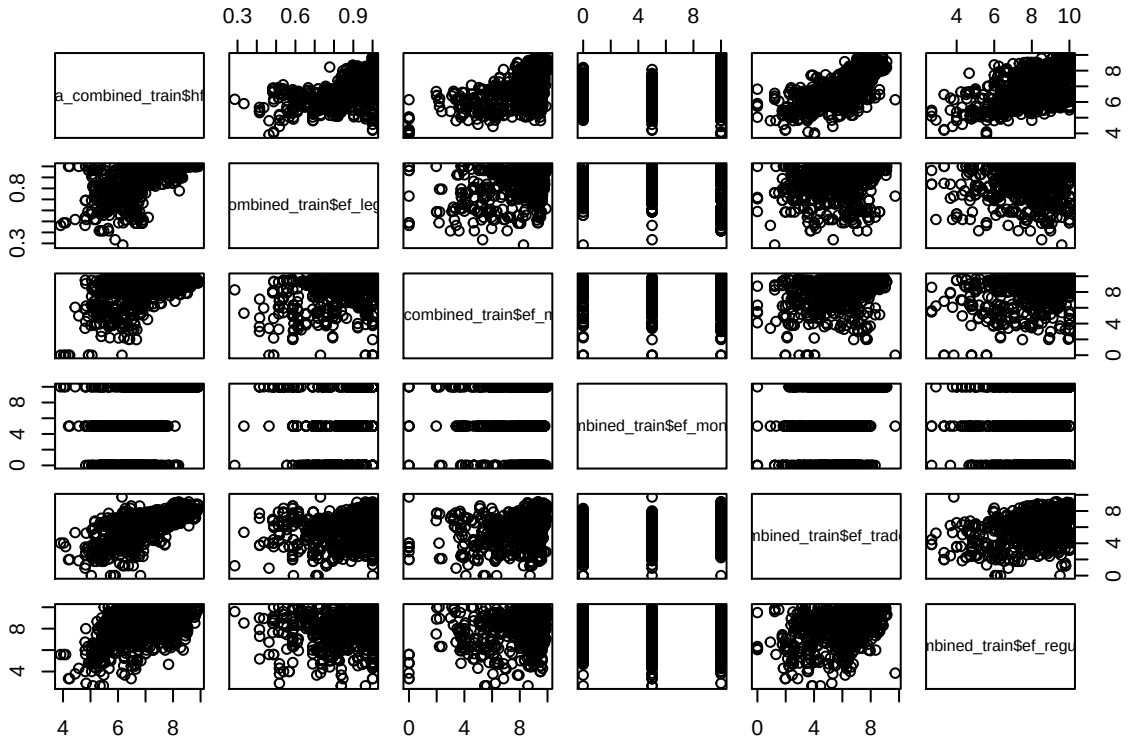
Initially, we need to analyze the pair-plots that showcase the correlation between variables. The focus is hf_score with the most significant variables found from the best-subset regression of size 10 predictors. This model was chosen as these will probably be the most significant variables we know so far. This isolates which predictors we should analyze for non-linear trends to the primary ones.

The Pair-Plots:

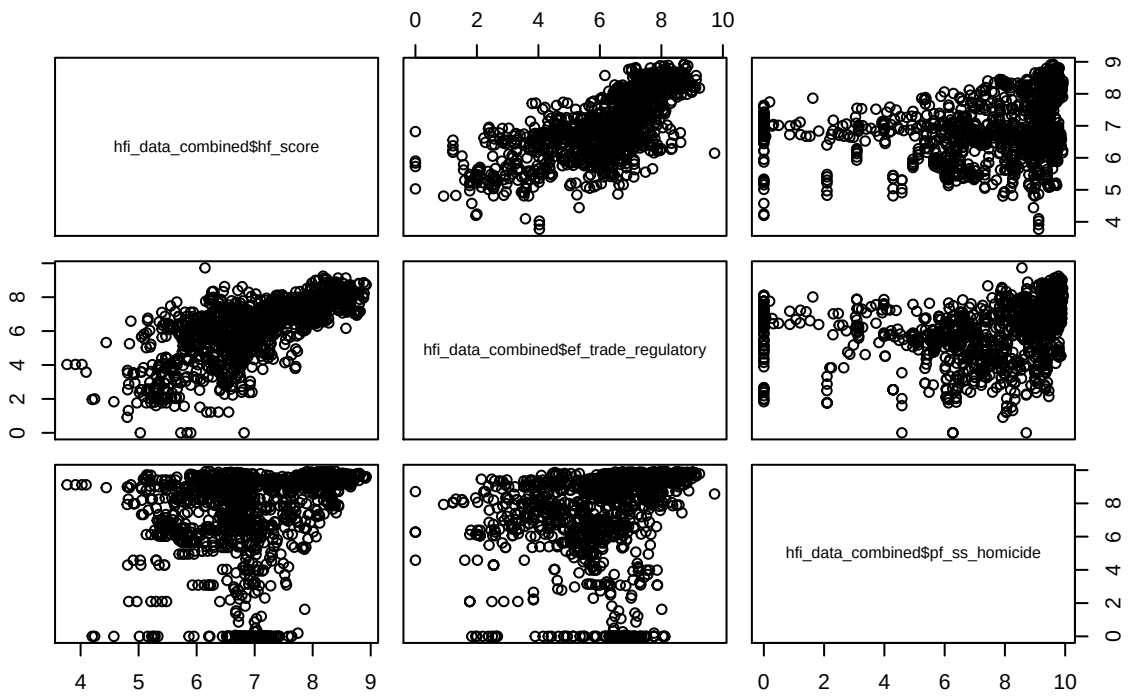
```
pairs(hfi_data_combined_train$hf_score ~ hfi_data_combined_train$pf_ss_homicide + hfi_data_combined_tra
        hfi_data_combined_train$pf_expression_control + hfi_data_combined_train$pf_identity_sex_male + h
```




```
pairs(hfi_data_combined_train$hf_score ~ hfi_data_combined_train$ef_legal_gender + hfi_data_combined_tra
        hfi_data_combined_train$ef_money_currency + hfi_data_combined_train$ef_trade_regulatory + hfi_da
```



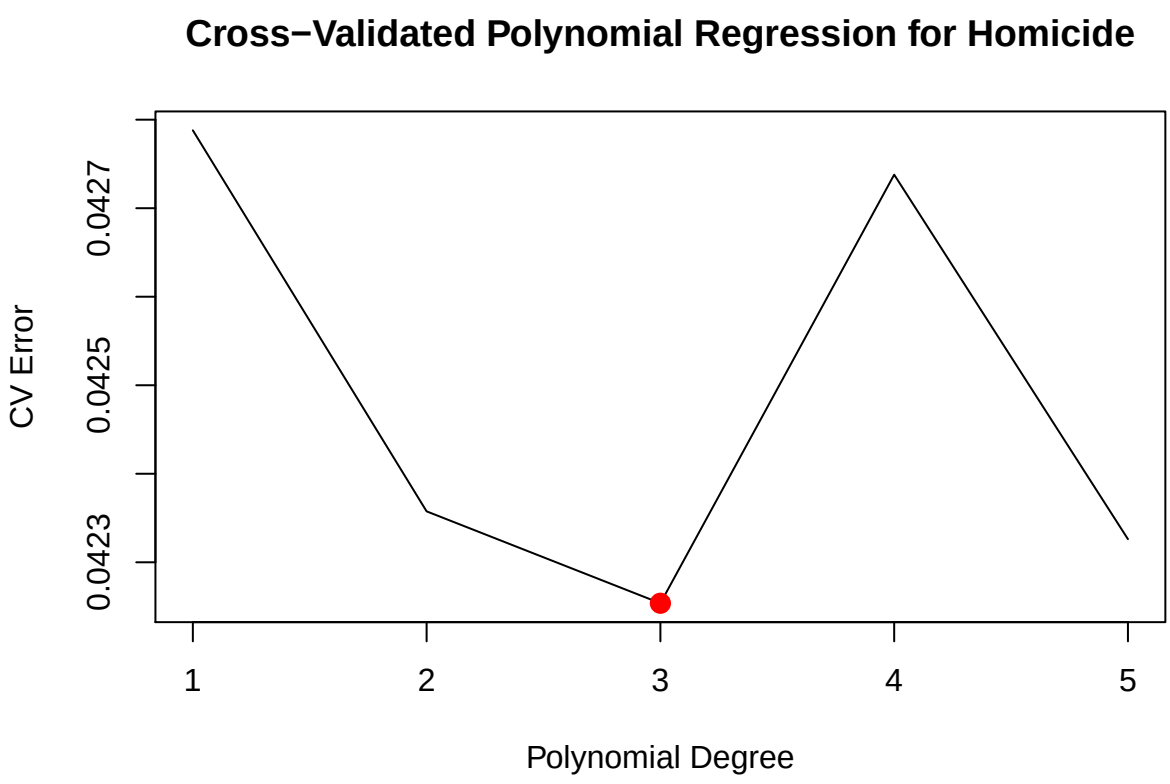

```
pairs(hfi_data_combined$hf_score ~ hfi_data_combined$ef_trade_regulatory + hfi_data_combined$pf_ss_homi
      main="The Homicide and Trade Regulation Pairs with HF_Score")
```



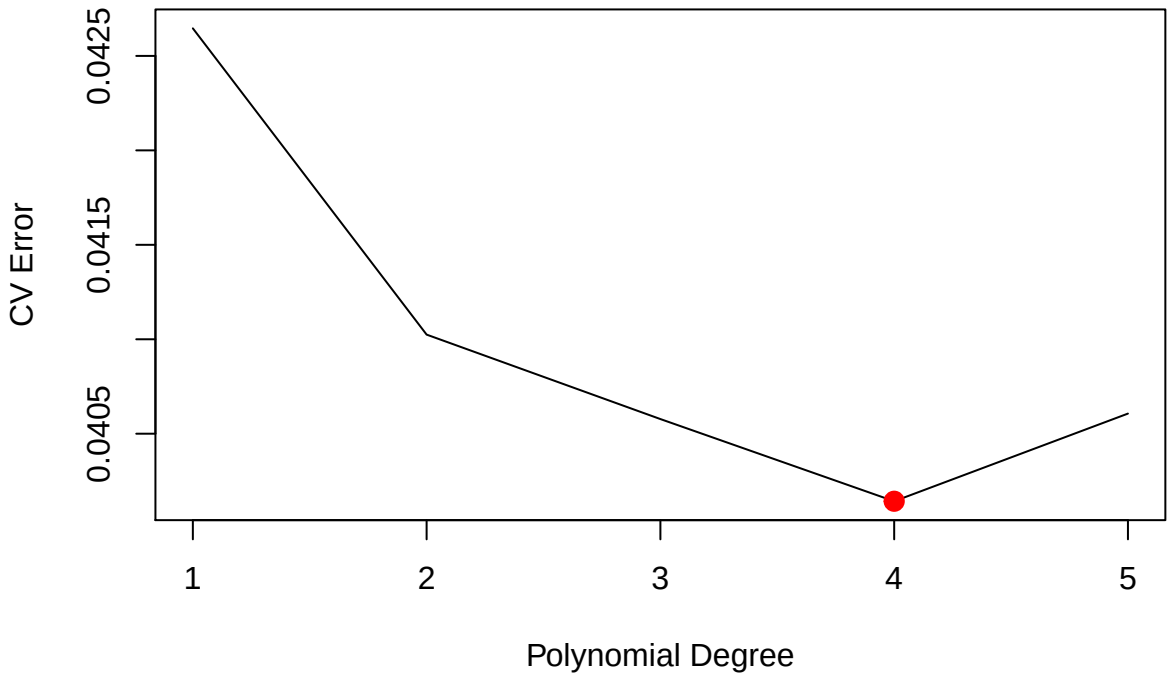
Overall, the pair plots do not indicate a large amount of polynomial trends, but more that certain values are more common (often skewed or multi-modal). We isolated that the clearest non-linear relationships with hf_score: pf_ss_homicide and ef_trade_regulatory.

Let's first test this with polynomial regression:

```
set.seed(111)
max_poly_term = 5
par(mfrow=c(1,1))

CVerror_homocide = rep(0, max_poly_term)
for (i in 1:max_poly_term) {
    fit = glm(hf_score ~ poly(pf_ss_homicide, i) + . - pf_ss_homicide, data = hfi_data_combined_train)
    CVerror_homocide[i] = cv.glm(hfi_data_combined_train, fit, K = 10)$delta[1]
}
plot(CVerror_homocide, main = "Cross-Validated Polynomial Regression for Homicide",
     xlab = "Polynomial Degree", ylab = "CV Error", type = "l")
points(which.min(CVerror_homocide), CVerror_homocide[which.min(CVerror_homocide)], col = "red",cex=2,pch
```

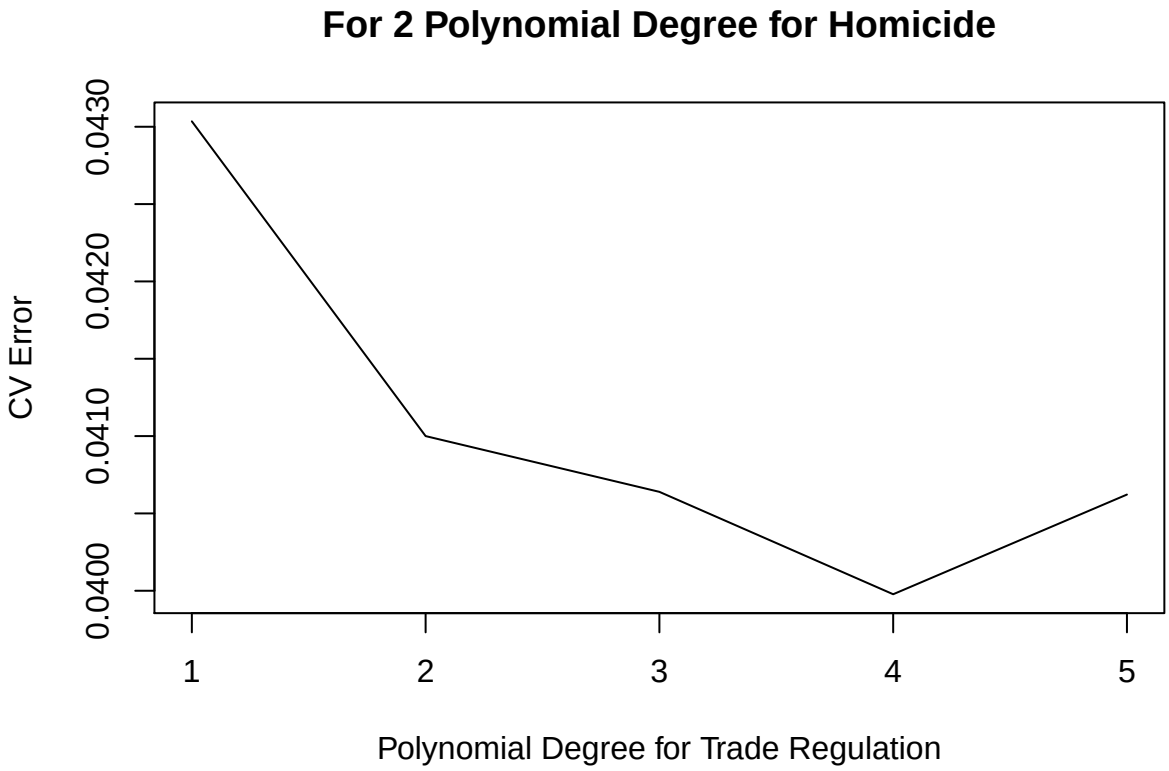


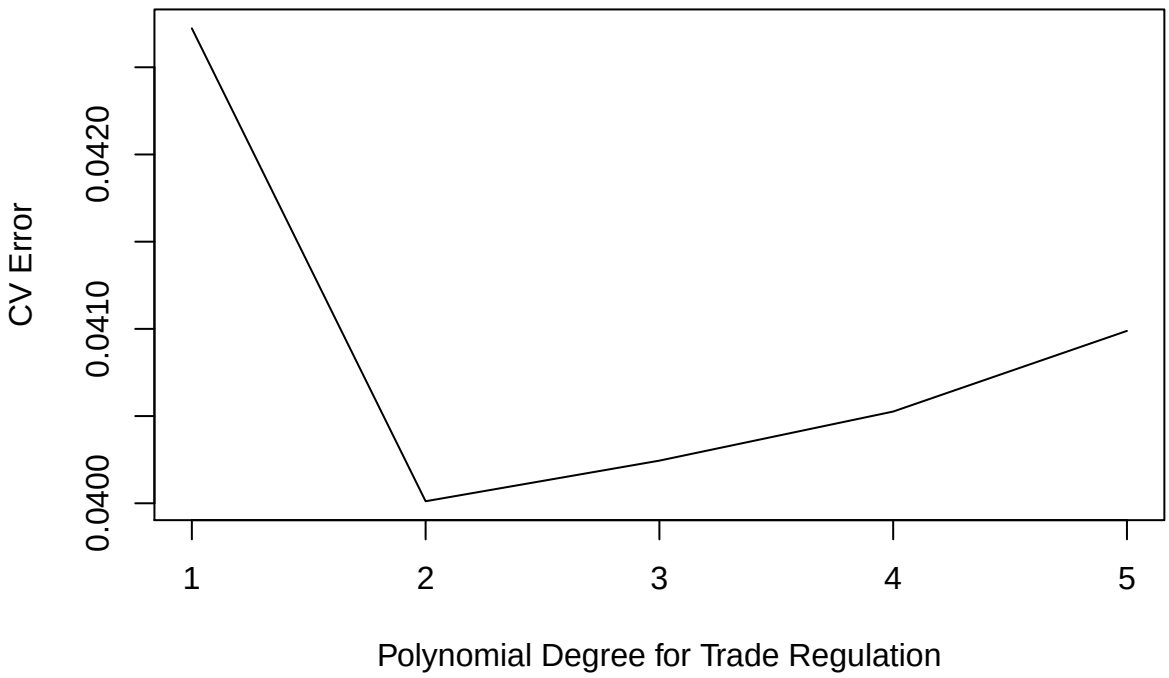

```
CVerror_trade_reg = rep(0, max_poly_term)
for (i in 1:max_poly_term) {
    fit = glm(hf_score~ poly(ef_trade_regulatory, i) + . - ef_trade_regulatory, data = hfi_data_combine
    CVerror_trade_reg[i] = cv.glm(hfi_data_combined_train, fit, K = 10)$delta[1]
}
plot(CVerror_trade_reg, main = "Cross-Validated Polynomial Regression for Trade Regulation",
     xlab = "Polynomial Degree", ylab = "CV Error", type = "l")
points(which.min(CVerror_trade_reg), CVerror_trade_reg[which.min(CVerror_trade_reg)], col = "red",cex=2
```

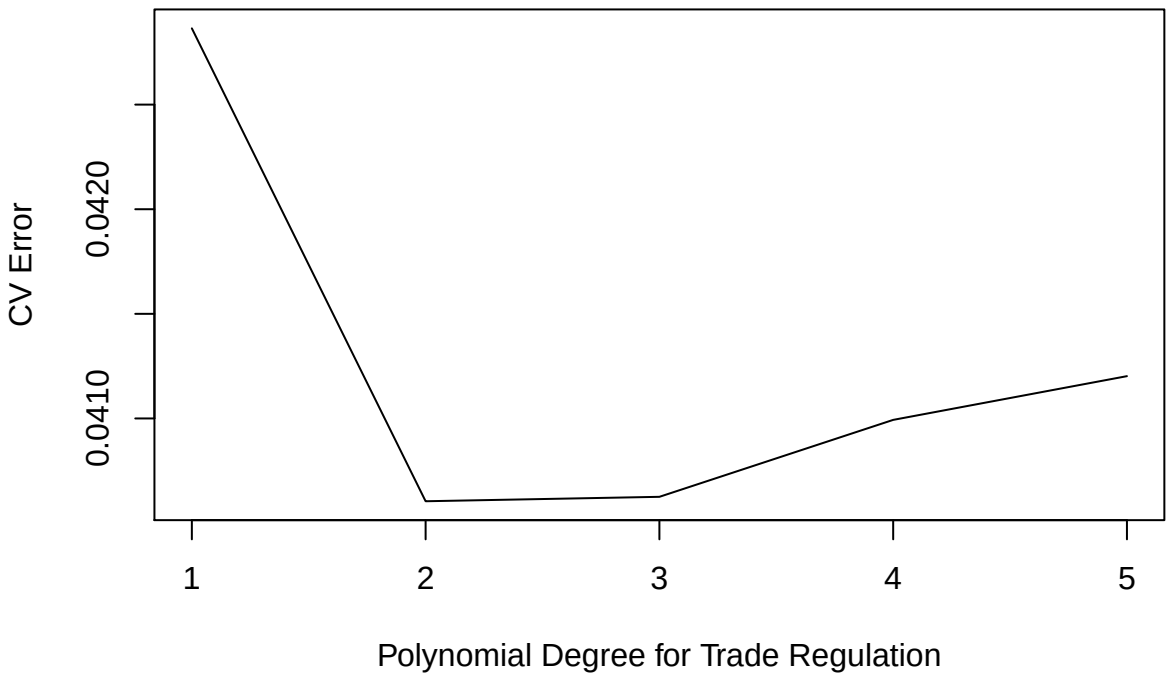


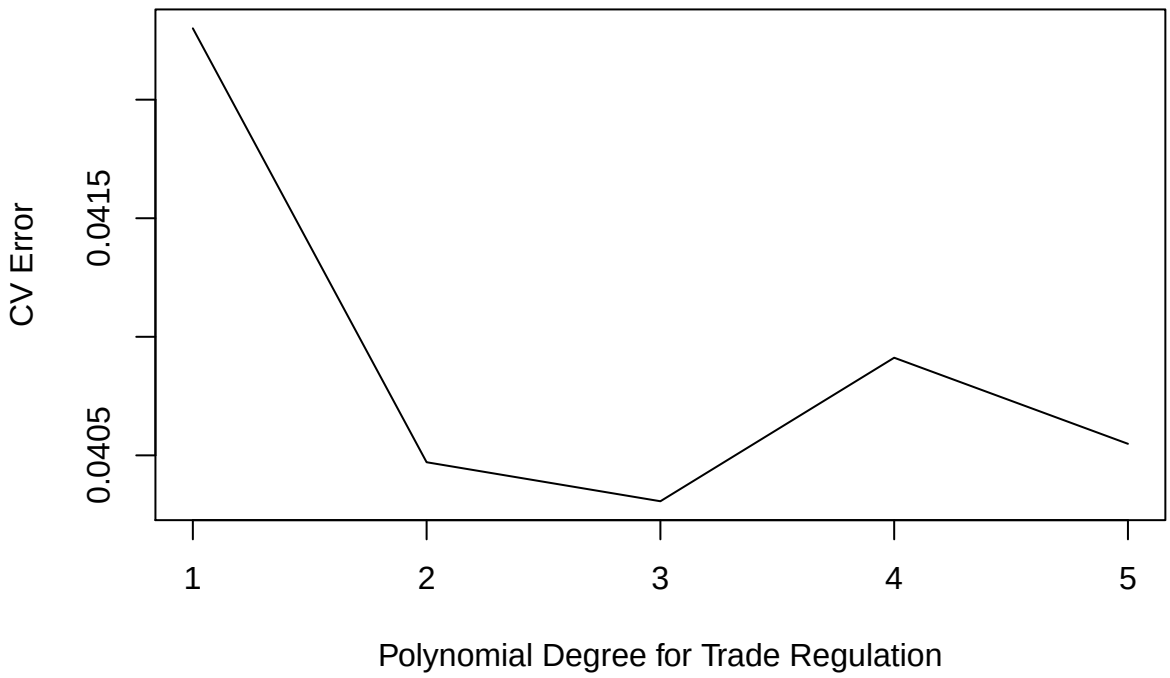

```
CVerror_combined = matrix(data=rep(rep(0, max_poly_term), max_poly_term), nrow=max_poly_term, ncol = ma
for (i in 1:max_poly_term) {
  for (j in 1:max_poly_term){
    fit = glm(hf_score~ poly(pf_ss_homicide, i) + poly(ef_trade_regulatory, j) + . -
                ef_trade_regulatory - pf_ss_homicide, data = hfi_data_combined_train)
    CVerror_combined[i, j] = cv.glm(hfi_data_combined_train, fit, K = 10)$delta[1]
  }
  plot(CVerror_combined[i,1:max_poly_term],
       main = paste("For", i, "Polynomial Degree for Homicide"),
       xlab = "Polynomial Degree for Trade Regulation", ylab = "CV Error", type = "l")
}
```


For 1 Polynomial Degree for Homicide
CV Error
0.0400
0.0410
0.0420
1
2
3
4
5
Polynomial Degree for Trade Regulation


For 2 Polynomial Degree for Homicide
CV Error
0.0400
0.0410
0.0420
0.0430
1
2
3
4
5
Polynomial Degree for Trade Regulation


For 3 Polynomial Degree for Homicide
CV Error
0.0400
0.0410
0.0420
1
2
3
4
5
Polynomial Degree for Trade Regulation


For 4 Polynomial Degree for Homicide
CV Error
0.0420
0.0410
1
2
3
4
5
Polynomial Degree for Trade Regulation




```
polyfit_homicide = lm(hf_score ~ poly(pf_ss_homicide, which.min(CVerror_homocide)) + . -
                        pf_ss_homicide, data = hfi_data_combined_train)
polyfit_trade_reg = lm(hf_score ~ poly(ef_trade_regulatory, which.min(CVerror_trade_reg)) + . -
                         ef_trade_regulatory, data = hfi_data_combined_train)

polyfit_combined = lm(hf_score ~ poly(ef_trade_regulatory,
                                      arrayInd(which.min(CVerror_combined),
                                               dim(CVerror_combined))[1,2]) +
                        poly(pf_ss_homicide, arrayInd(which.min(CVerror_combined),
                                                      dim(CVerror_combined))[1,1]) +
                        . - ef_trade_regulatory - pf_ss_homicide, data = hfi_data_combined_train)

summary(polyfit_homicide)
```

```
##
## Call:
## lm(formula = hf_score ~ poly(pf_ss_homicide, which.min(CVerror_homocide)) +
##     . - pf_ss_homicide, data = hfi_data_combined_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.64870 -0.12449  0.01113  0.12901  0.83843
##
## Coefficients:
##                                                  Estimate Std. Error t value
## (Intercept)                                      0.547950   0.140994   3.886
```

```
## poly(pf_ss_homicide, which.min(CVerror_homocide))1  4.790312   0.279480  17.140
## poly(pf_ss_homicide, which.min(CVerror_homocide))2  0.463717   0.256543   1.808
## poly(pf_ss_homicide, which.min(CVerror_homocide))3 -0.236718   0.223218  -1.060
## pf_ss_disappearances_disap                           0.007781   0.002656   2.929
## pf_ss_disappearances_violent                         0.011086   0.005830   1.902
## pf_ss_disappearances_fatalities                      0.013376   0.008553   1.564
## pf_ss_disappearances_injuries                       -0.011095   0.009675  -1.147
## pf_movement_domestic                                 0.018729   0.002435   7.690
## pf_movement_foreign                                  0.016525   0.002279   7.250
## pf_religion_harassment                               0.026265   0.009583   2.741
## pf_religion_restrictions                             0.025011   0.005324   4.698
## pf_expression_killed                                 0.004797   0.003186   1.506
## pf_expression_jailed                                 0.004502   0.005823   0.773
## pf_expression_influence                              0.041976   0.007328   5.728
## pf_expression_control                                0.052902   0.008885   5.954
## pf_identity_sex_male                                 0.021768   0.002384   9.130
## pf_identity_sex_female                               0.016418   0.002630   6.242
## ef_government_consumption                            0.040139   0.003991  10.057
## ef_legal_courts                                      0.071404   0.006019  11.863
## ef_legal_military                                    0.042406   0.004240  10.002
## ef_legal_enforcement                                 0.050728   0.005331   9.516
## ef_legal_gender                                      1.005031   0.067199  14.956
## ef_money_growth                                      0.030642   0.007296   4.200
## ef_money_sd                                          0.035872   0.005597   6.409
## ef_money_inflation                                   0.030666   0.006784   4.521
## ef_money_currency                                    0.024999   0.002471  10.116
## ef_trade_tariffs_mean                                0.054934   0.012607   4.357
## ef_trade_tariffs_sd                                 -0.002009   0.005940  -0.338
## ef_trade_tariffs                                     0.005524   0.013546   0.408
## ef_trade_regulatory_compliance                       0.012988   0.008570   1.516
## ef_trade_regulatory                                  0.037723   0.013725   2.748
## ef_trade_black                                       0.012721   0.006671   1.907
## ef_trade_movement_capital                            0.017284   0.003282   5.267
## ef_trade_movement_visit                              0.017898   0.002433   7.357
## ef_regulation_credit_private                         0.008571   0.004236   2.023
## ef_regulation_credit                                 0.034967   0.008174   4.278
## ef_regulation_labor_minwage                          0.003811   0.002879   1.324
## ef_regulation_labor_hours                            0.013015   0.003706   3.512
## ef_regulation_labor_conscription                     0.005602   0.001771   3.163
## ef_regulation_business_start                         0.018529   0.007470   2.480
## ef_regulation_business_compliance                    0.001934   0.004166   0.464
##                                                     Pr(>|t|)
## (Intercept)                                         0.000108 ***
## poly(pf_ss_homicide, which.min(CVerror_homocide))1  < 2e-16 ***
## poly(pf_ss_homicide, which.min(CVerror_homocide))2 0.070975 .
## poly(pf_ss_homicide, which.min(CVerror_homocide))3 0.289183
## pf_ss_disappearances_disap                          0.003474 **
## pf_ss_disappearances_violent                        0.057520 .
## pf_ss_disappearances_fatalities                     0.118146
## pf_ss_disappearances_injuries                       0.251712
## pf_movement_domestic                                3.50e-14 ***
## pf_movement_foreign                                 8.35e-13 ***
## pf_religion_harassment                              0.006239 **
## pf_religion_restrictions                            2.99e-06 ***
```

```
## pf_expression_killed                                0.132418
## pf_expression_jailed                                0.439683
## pf_expression_influence                             1.34e-08 ***
## pf_expression_control                               3.61e-09 ***
## pf_identity_sex_male                                 < 2e-16 ***
## pf_identity_sex_female                              6.38e-10 ***
## ef_government_consumption                            < 2e-16 ***
## ef_legal_courts                                      < 2e-16 ***
## ef_legal_military                                    < 2e-16 ***
## ef_legal_enforcement                                 < 2e-16 ***
## ef_legal_gender                                      < 2e-16 ***
## ef_money_growth                                     2.91e-05 ***
## ef_money_sd                                         2.25e-10 ***
## ef_money_inflation                                  6.91e-06 ***
## ef_money_currency                                    < 2e-16 ***
## ef_trade_tariffs_mean                               1.45e-05 ***
## ef_trade_tariffs_sd                                 0.735313
## ef_trade_tariffs                                    0.683499
## ef_trade_regulatory_compliance                      0.129954
## ef_trade_regulatory                                 0.006095 **
## ef_trade_black                                      0.056819 .
## ef_trade_movement_capital                           1.70e-07 ***
## ef_trade_movement_visit                             3.91e-13 ***
## ef_regulation_credit_private                        0.043301 *
## ef_regulation_credit                                2.07e-05 ***
## ef_regulation_labor_minwage                         0.185907
## ef_regulation_labor_hours                           0.000465 ***
## ef_regulation_labor_conscription                    0.001610 **
## ef_regulation_business_start                        0.013286 *
## ef_regulation_business_compliance                   0.642669
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1998 on 1002 degrees of freedom
## Multiple R-squared:  0.9617, Adjusted R-squared:  0.9601
## F-statistic: 613.2 on 41 and 1002 DF,  p-value: < 2.2e-16
```

```
yhat = predict(polyfit_homicide, newdata = hfi_data_combined_test)
polyfit_homicide_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

summary(polyfit_trade_reg)
```

```
##
## Call:
## lm(formula = hf_score ~ poly(ef_trade_regulatory, which.min(CVerror_trade_reg)) +
##     . - ef_trade_regulatory, data = hfi_data_combined_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.61567 -0.12767  0.01549  0.12815  0.71910
##
## Coefficients:
##                                                      Estimate Std. Error
## (Intercept)                                          0.402484   0.158341
## poly(ef_trade_regulatory, which.min(CVerror_trade_reg))1  2.595332   0.789196
```

```
## poly(ef_trade_regulatory, which.min(CVerror_trade_reg))2  1.689967   0.227994
## poly(ef_trade_regulatory, which.min(CVerror_trade_reg))3 -0.183688   0.231637
## poly(ef_trade_regulatory, which.min(CVerror_trade_reg))4 -0.400560   0.203097
## pf_ss_homicide                                            0.049379   0.002971
## pf_ss_disappearances_disap                                0.009489   0.002596
## pf_ss_disappearances_violent                              0.013291   0.005681
## pf_ss_disappearances_fatalities                           0.019476   0.008342
## pf_ss_disappearances_injuries                            -0.015004   0.009375
## pf_movement_domestic                                      0.019771   0.002389
## pf_movement_foreign                                       0.016242   0.002224
## pf_religion_harassment                                    0.022005   0.009298
## pf_religion_restrictions                                  0.028101   0.005181
## pf_expression_killed                                      0.003012   0.003103
## pf_expression_jailed                                      0.002384   0.005681
## pf_expression_influence                                   0.039409   0.007143
## pf_expression_control                                     0.050027   0.008622
## pf_identity_sex_male                                      0.020290   0.002329
## pf_identity_sex_female                                    0.016821   0.002562
## ef_government_consumption                                 0.038246   0.003840
## ef_legal_courts                                           0.061593   0.005973
## ef_legal_military                                         0.041979   0.004115
## ef_legal_enforcement                                      0.049645   0.005157
## ef_legal_gender                                           0.992303   0.065340
## ef_money_growth                                           0.030342   0.007116
## ef_money_sd                                               0.040560   0.005470
## ef_money_inflation                                        0.022199   0.006685
## ef_money_currency                                         0.024158   0.002413
## ef_trade_tariffs_mean                                     0.052231   0.012050
## ef_trade_tariffs_sd                                      -0.007870   0.005726
## ef_trade_tariffs                                          0.012539   0.012801
## ef_trade_regulatory_compliance                            0.011849   0.009096
## ef_trade_black                                            0.019928   0.006517
## ef_trade_movement_capital                                 0.015850   0.003163
## ef_trade_movement_visit                                   0.016760   0.002378
## ef_regulation_credit_private                              0.006763   0.004131
## ef_regulation_credit                                      0.038848   0.007977
## ef_regulation_labor_minwage                               0.004321   0.002801
## ef_regulation_labor_hours                                 0.016924   0.003632
## ef_regulation_labor_conscription                          0.006064   0.001738
## ef_regulation_business_start                              0.024062   0.007159
## ef_regulation_business_compliance                         0.002418   0.004044
##                                                          t value Pr(>|t|)
## (Intercept)                                                2.542 0.011175 *
## poly(ef_trade_regulatory, which.min(CVerror_trade_reg))1   3.289 0.001042 **
## poly(ef_trade_regulatory, which.min(CVerror_trade_reg))2   7.412 2.64e-13 ***
## poly(ef_trade_regulatory, which.min(CVerror_trade_reg))3  -0.793 0.427967
## poly(ef_trade_regulatory, which.min(CVerror_trade_reg))4  -1.972 0.048855 *
## pf_ss_homicide                                            16.621  < 2e-16 ***
## pf_ss_disappearances_disap                                 3.655 0.000270 ***
## pf_ss_disappearances_violent                               2.340 0.019497 *
## pf_ss_disappearances_fatalities                            2.335 0.019758 *
## pf_ss_disappearances_injuries                             -1.600 0.109839
## pf_movement_domestic                                       8.277 3.99e-16 ***
## pf_movement_foreign                                        7.304 5.68e-13 ***
```

```
## pf_religion_harassment                                  2.367 0.018137 *
## pf_religion_restrictions                                5.424 7.29e-08 ***
## pf_expression_killed                                    0.970 0.332039
## pf_expression_jailed                                    0.420 0.674864
## pf_expression_influence                                 5.517 4.39e-08 ***
## pf_expression_control                                   5.803 8.76e-09 ***
## pf_identity_sex_male                                    8.711  < 2e-16 ***
## pf_identity_sex_female                                  6.565 8.34e-11 ***
## ef_government_consumption                               9.960  < 2e-16 ***
## ef_legal_courts                                        10.311  < 2e-16 ***
## ef_legal_military                                      10.201  < 2e-16 ***
## ef_legal_enforcement                                    9.627  < 2e-16 ***
## ef_legal_gender                                        15.187  < 2e-16 ***
## ef_money_growth                                         4.264 2.20e-05 ***
## ef_money_sd                                             7.416 2.58e-13 ***
## ef_money_inflation                                      3.321 0.000930 ***
## ef_money_currency                                      10.013  < 2e-16 ***
## ef_trade_tariffs_mean                                   4.335 1.61e-05 ***
## ef_trade_tariffs_sd                                    -1.375 0.169555
## ef_trade_tariffs                                        0.980 0.327534
## ef_trade_regulatory_compliance                          1.303 0.193007
## ef_trade_black                                          3.058 0.002287 **
## ef_trade_movement_capital                               5.011 6.40e-07 ***
## ef_trade_movement_visit                                 7.048 3.38e-12 ***
## ef_regulation_credit_private                            1.637 0.101925
## ef_regulation_credit                                    4.870 1.30e-06 ***
## ef_regulation_labor_minwage                             1.543 0.123254
## ef_regulation_labor_hours                               4.660 3.58e-06 ***
## ef_regulation_labor_conscription                        3.489 0.000505 ***
## ef_regulation_business_start                            3.361 0.000805 ***
## ef_regulation_business_compliance                       0.598 0.550055
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1945 on 1001 degrees of freedom
## Multiple R-squared:  0.9637, Adjusted R-squared:  0.9622
## F-statistic: 632.9 on 42 and 1001 DF,  p-value: < 2.2e-16
```

```
yhat = predict(polyfit_trade_reg, newdata = hfi_data_combined_test)
polyfit_trade_reg_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

summary(polyfit_combined)
```

```
##
## Call:
## lm(formula = hf_score ~ poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined),
##     dim(CVerror_combined))[1, 2]) + poly(pf_ss_homicide, arrayInd(which.min(CVerror_combined),
##     dim(CVerror_combined))[1, 1]) + . - ef_trade_regulatory -
##     pf_ss_homicide, data = hfi_data_combined_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.61567 -0.12767  0.01549  0.12815  0.71910
##
## Coefficients:
```

```
##                                                                                                  Estim
## (Intercept)                                                                                       0.768
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])1  2.595
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])2  1.689
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])3 -0.183
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])4 -0.400
## poly(pf_ss_homicide, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 1])        4.479
## pf_ss_disappearances_disap                                                                       0.009
## pf_ss_disappearances_violent                                                                     0.013
## pf_ss_disappearances_fatalities                                                                  0.019
## pf_ss_disappearances_injuries                                                                   -0.015
## pf_movement_domestic                                                                             0.019
## pf_movement_foreign                                                                              0.010
## pf_religion_harassment                                                                           0.022
## pf_religion_restrictions                                                                         0.028
## pf_expression_killed                                                                             0.003
## pf_expression_jailed                                                                             0.001
## pf_expression_influence                                                                          0.039
## pf_expression_control                                                                            0.050
## pf_identity_sex_male                                                                             0.020
## pf_identity_sex_female                                                                           0.010
## ef_government_consumption                                                                        0.038
## ef_legal_courts                                                                                  0.061
## ef_legal_military                                                                                0.041
## ef_legal_enforcement                                                                             0.049
## ef_legal_gender                                                                                  0.991
## ef_money_growth                                                                                  0.030
## ef_money_sd                                                                                      0.040
## ef_money_inflation                                                                               0.022
## ef_money_currency                                                                                0.024
## ef_trade_tariffs_mean                                                                            0.051
## ef_trade_tariffs_sd                                                                             -0.007
## ef_trade_tariffs                                                                                 0.011
## ef_trade_regulatory_compliance                                                                   0.011
## ef_trade_black                                                                                   0.019
## ef_trade_movement_capital                                                                        0.015
## ef_trade_movement_visit                                                                          0.010
## ef_regulation_credit_private                                                                     0.000
## ef_regulation_credit                                                                             0.038
## ef_regulation_labor_minwage                                                                      0.004
## ef_regulation_labor_hours                                                                        0.010
## ef_regulation_labor_conscription                                                                 0.000
## ef_regulation_business_start                                                                     0.024
## ef_regulation_business_compliance                                                                0.001
##                                                                                                 Std. E
## (Intercept)                                                                                       0.15
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])1   0.78
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])2   0.21
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])3   0.23
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])4   0.20
## poly(pf_ss_homicide, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 1])         0.20
## pf_ss_disappearances_disap                                                                       0.00
## pf_ss_disappearances_violent                                                                     0.00
## pf_ss_disappearances_fatalities                                                                  0.00
```

```
## pf_ss_disappearances_injuries                                                                  0.0
## pf_movement_domestic                                                                           0.0
## pf_movement_foreign                                                                            0.0
## pf_religion_harassment                                                                         0.0
## pf_religion_restrictions                                                                       0.0
## pf_expression_killed                                                                           0.0
## pf_expression_jailed                                                                           0.0
## pf_expression_influence                                                                        0.0
## pf_expression_control                                                                          0.0
## pf_identity_sex_male                                                                           0.0
## pf_identity_sex_female                                                                         0.0
## ef_government_consumption                                                                      0.0
## ef_legal_courts                                                                                0.0
## ef_legal_military                                                                              0.0
## ef_legal_enforcement                                                                           0.0
## ef_legal_gender                                                                                0.0
## ef_money_growth                                                                                0.0
## ef_money_sd                                                                                    0.0
## ef_money_inflation                                                                             0.0
## ef_money_currency                                                                              0.0
## ef_trade_tariffs_mean                                                                          0.0
## ef_trade_tariffs_sd                                                                            0.0
## ef_trade_tariffs                                                                               0.0
## ef_trade_regulatory_compliance                                                                 0.0
## ef_trade_black                                                                                 0.0
## ef_trade_movement_capital                                                                      0.0
## ef_trade_movement_visit                                                                        0.0
## ef_regulation_credit_private                                                                   0.0
## ef_regulation_credit                                                                           0.0
## ef_regulation_labor_minwage                                                                    0.0
## ef_regulation_labor_hours                                                                      0.0
## ef_regulation_labor_conscription                                                               0.0
## ef_regulation_business_start                                                                   0.0
## ef_regulation_business_compliance                                                              0.0
##                                                                                              t valu
## (Intercept)                                                                                    4.84
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])1   3.28
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])2   7.41
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])3  -0.79
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])4  -1.97
## poly(pf_ss_homicide, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 1])       16.61
## pf_ss_disappearances_disap                                                                     3.65
## pf_ss_disappearances_violent                                                                   2.34
## pf_ss_disappearances_fatalities                                                                2.33
## pf_ss_disappearances_injuries                                                                 -1.60
## pf_movement_domestic                                                                           8.27
## pf_movement_foreign                                                                            7.30
## pf_religion_harassment                                                                         2.30
## pf_religion_restrictions                                                                       5.41
## pf_expression_killed                                                                           0.97
## pf_expression_jailed                                                                           0.41
## pf_expression_influence                                                                        5.51
## pf_expression_control                                                                          5.80
## pf_identity_sex_male                                                                           8.71
```

```
## pf_identity_sex_female                                                                                  6.5
## ef_government_consumption                                                                               9.9
## ef_legal_courts                                                                                        10.3
## ef_legal_military                                                                                      10.2
## ef_legal_enforcement                                                                                    9.6
## ef_legal_gender                                                                                        15.1
## ef_money_growth                                                                                         4.2
## ef_money_sd                                                                                             7.4
## ef_money_inflation                                                                                      3.3
## ef_money_currency                                                                                      10.0
## ef_trade_tariffs_mean                                                                                   4.3
## ef_trade_tariffs_sd                                                                                    -1.3
## ef_trade_tariffs                                                                                        0.9
## ef_trade_regulatory_compliance                                                                          1.3
## ef_trade_black                                                                                          3.0
## ef_trade_movement_capital                                                                               5.0
## ef_trade_movement_visit                                                                                 7.0
## ef_regulation_credit_private                                                                            1.6
## ef_regulation_credit                                                                                    4.8
## ef_regulation_labor_minwage                                                                             1.5
## ef_regulation_labor_hours                                                                               4.6
## ef_regulation_labor_conscription                                                                        3.4
## ef_regulation_business_start                                                                            3.3
## ef_regulation_business_compliance                                                                       0.5
##                                                                                                       Pr(>|
## (Intercept)                                                                                           1.49e
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])1 0.001
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])2 2.64e
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])3 0.427
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])4 0.048
## poly(pf_ss_homicide, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 1])        < 2e
## pf_ss_disappearances_disap                                                                            0.000
## pf_ss_disappearances_violent                                                                          0.019
## pf_ss_disappearances_fatalities                                                                       0.019
## pf_ss_disappearances_injuries                                                                         0.109
## pf_movement_domestic                                                                                  3.99e
## pf_movement_foreign                                                                                   5.68e
## pf_religion_harassment                                                                                0.018
## pf_religion_restrictions                                                                              7.29e
## pf_expression_killed                                                                                  0.332
## pf_expression_jailed                                                                                  0.674
## pf_expression_influence                                                                               4.39e
## pf_expression_control                                                                                 8.76e
## pf_identity_sex_male                                                                                   < 2e
## pf_identity_sex_female                                                                                8.34e
## ef_government_consumption                                                                              < 2e
## ef_legal_courts                                                                                        < 2e
## ef_legal_military                                                                                      < 2e
## ef_legal_enforcement                                                                                   < 2e
## ef_legal_gender                                                                                        < 2e
## ef_money_growth                                                                                       2.20e
## ef_money_sd                                                                                           2.58e
## ef_money_inflation                                                                                    0.000
## ef_money_currency                                                                                      < 2e
```

```
## ef_trade_tariffs_mean                                                                    1.61e-
## ef_trade_tariffs_sd                                                                      0.169
## ef_trade_tariffs                                                                         0.327
## ef_trade_regulatory_compliance                                                           0.193
## ef_trade_black                                                                           0.002
## ef_trade_movement_capital                                                                6.40e-
## ef_trade_movement_visit                                                                  3.38e-
## ef_regulation_credit_private                                                             0.101
## ef_regulation_credit                                                                     1.30e-
## ef_regulation_labor_minwage                                                              0.123
## ef_regulation_labor_hours                                                                3.58e-
## ef_regulation_labor_conscription                                                         0.000
## ef_regulation_business_start                                                             0.000
## ef_regulation_business_compliance                                                        0.550
##
## (Intercept)                                                                              ***
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])1 **
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])2 ***
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])3
## poly(ef_trade_regulatory, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 2])4 *
## poly(pf_ss_homicide, arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1, 1])       ***
## pf_ss_disappearances_disap                                                               ***
## pf_ss_disappearances_violent                                                             *
## pf_ss_disappearances_fatalities                                                          *
## pf_ss_disappearances_injuries
## pf_movement_domestic                                                                     ***
## pf_movement_foreign                                                                      ***
## pf_religion_harassment                                                                   *
## pf_religion_restrictions                                                                 ***
## pf_expression_killed
## pf_expression_jailed
## pf_expression_influence                                                                  ***
## pf_expression_control                                                                    ***
## pf_identity_sex_male                                                                     ***
## pf_identity_sex_female                                                                   ***
## ef_government_consumption                                                                ***
## ef_legal_courts                                                                          ***
## ef_legal_military                                                                        ***
## ef_legal_enforcement                                                                     ***
## ef_legal_gender                                                                          ***
## ef_money_growth                                                                          ***
## ef_money_sd                                                                              ***
## ef_money_inflation                                                                       ***
## ef_money_currency                                                                        ***
## ef_trade_tariffs_mean                                                                    ***
## ef_trade_tariffs_sd
## ef_trade_tariffs
## ef_trade_regulatory_compliance
## ef_trade_black                                                                           **
## ef_trade_movement_capital                                                                ***
## ef_trade_movement_visit                                                                  ***
## ef_regulation_credit_private
## ef_regulation_credit                                                                     ***
## ef_regulation_labor_minwage
```

```
## ef_regulation_labor_hours                                                    ***
## ef_regulation_labor_conscription                                             ***
## ef_regulation_business_start                                                 ***
## ef_regulation_business_compliance
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1945 on 1001 degrees of freedom
## Multiple R-squared:  0.9637, Adjusted R-squared:  0.9622
## F-statistic: 632.9 on 42 and 1001 DF,  p-value: < 2.2e-16
```

```
yhat = predict(polyfit_combined, newdata = hfi_data_combined_test)
polyfit_combined_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

paste("Minimum Cross-Validate MSE of Combined Polynomial Terms is:",
      (CVerror_combined[arrayInd(which.min(CVerror_combined), dim(CVerror_combined))]))
```

```
## [1] "Minimum Cross-Validate MSE of Combined Polynomial Terms is: 0.0399753403154818"
```

```
paste("With the Homicide Polynomial Term at",
        arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1,1])
```

```
## [1] "With the Homicide Polynomial Term at 1"
```

```
paste("With the Trade Regulation Polynomial Term at",
        arrayInd(which.min(CVerror_combined), dim(CVerror_combined))[1,2])
```

```
## [1] "With the Trade Regulation Polynomial Term at 4"
```

```
paste("MLR Forward Regression with No Polynomial Terms Test MSE: ", round(final_forward_mse, 6))
```

```
## [1] "MLR Forward Regression with No Polynomial Terms Test MSE:  0.042611"
```

```
paste("MLR with Polynomial Term Homicide Test MSE: ", round(polyfit_homicide_mse, 6))
```

```
## [1] "MLR with Polynomial Term Homicide Test MSE:  0.042126"
```

```
paste("MLR with Polynomial Term Trade Regulation Test MSE:", round(polyfit_trade_reg_mse, 6))
```

```
## [1] "MLR with Polynomial Term Trade Regulation Test MSE: 0.040142"
```

```
paste("MLR with Polynomial Term Trade Regulation + Homicide Test MSE:", round(polyfit_combined_mse, 6))
```

```
## [1] "MLR with Polynomial Term Trade Regulation + Homicide Test MSE: 0.040142"
```

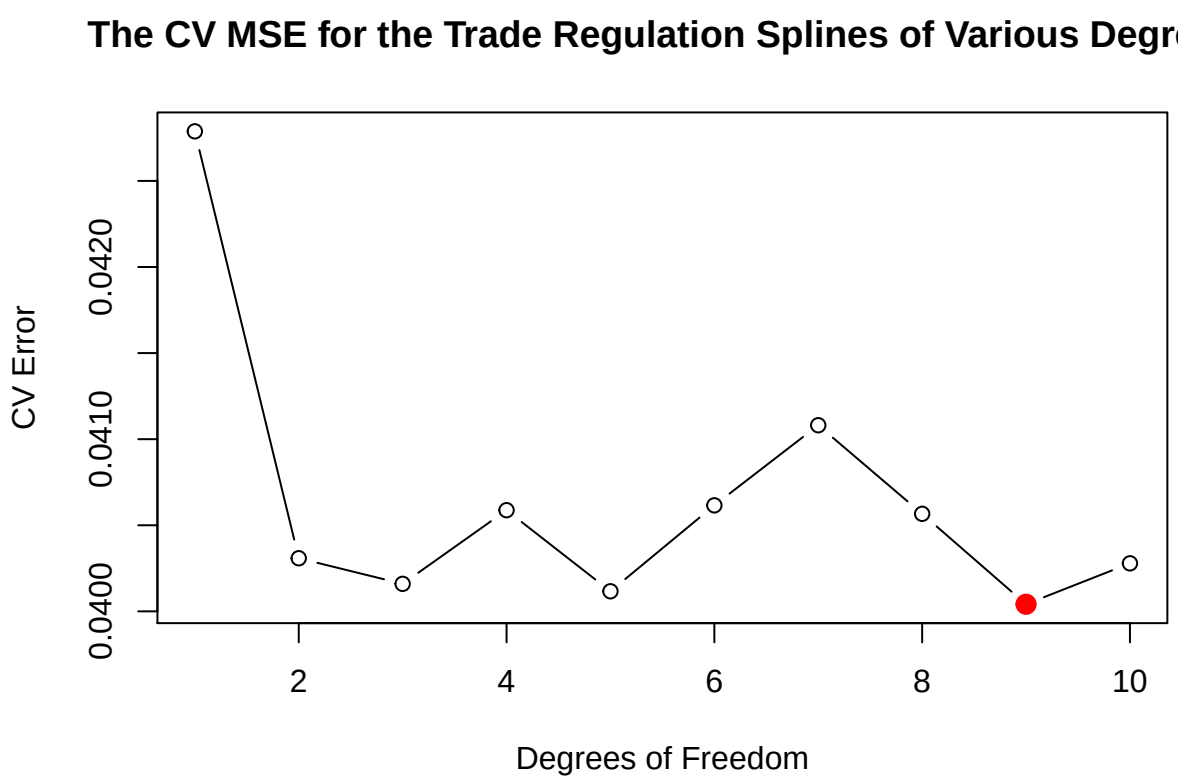
After performing the polynomial regression, the results seem to indicate that the test MSE of the cross-validate polynomial models is improved slightly. The combined model and the trade regulation polynomial models are identical as the combined model sets the homicide polynomial to degree 1 (which is identical to a normal, non-polynomial model). The Test MSE of the combined polynomial model (and trade regulation model as they are identical) is 0.040142 compared to the forward selection or best-subset model's Test MSE of 0.042611. There is a slight improvement of roughly 5% in the MSE, however, relative to the Mean TSS, the change is marginal at best. The combined polynomial regression explains less than 1% more variance than the forward selection MLR with no polynomial terms.

Therefore, we will not use polynomial regression as it adds unnecessary complexity and the relation does not seem obviously quartic (refer back to the pair-plots above). Since there is no obvious reason that a 4th power term would help other than to increase the variance within the model, we strongly believe leaving trade regulation as a linear term (1-power).

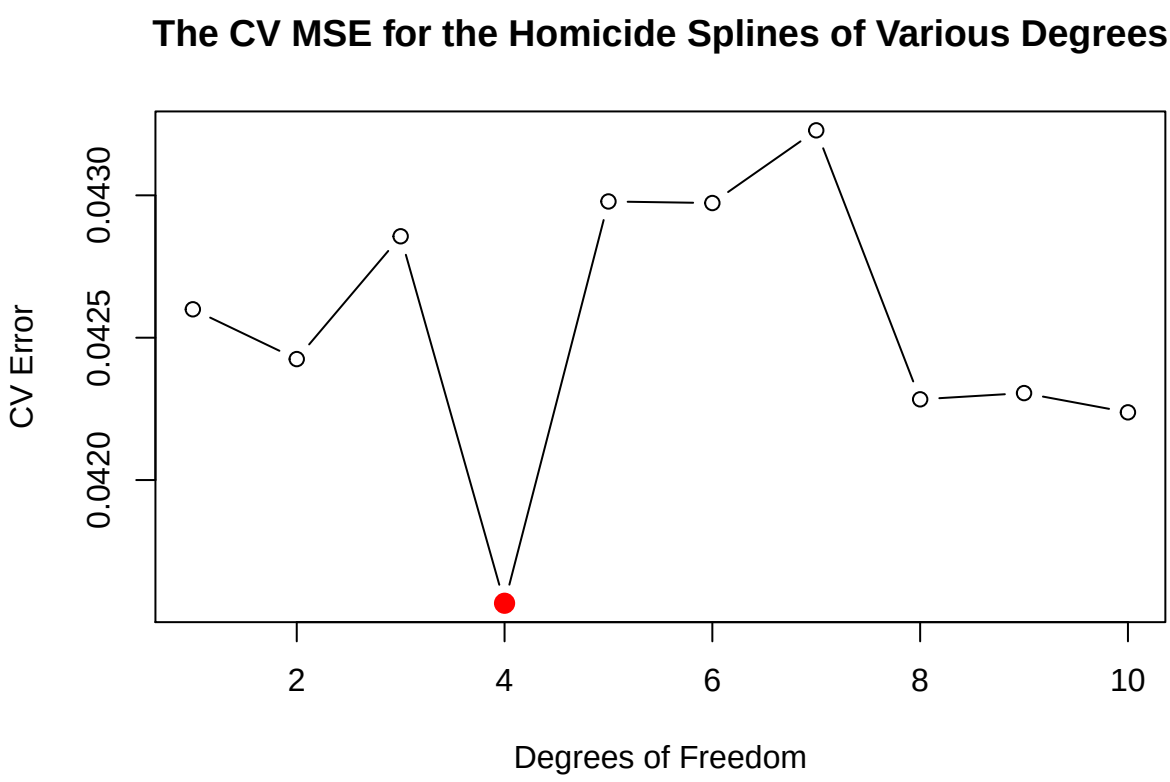
4.2 Splines

Here we will perform the natural spline using ef_trade_regulatory (trade regulation), which had the most non-linear relationship with Human Freedom. The goal is to see if a spline for the predictor trade regulation will have a positive effect on model accuracy. The worry here is that the spline will significantly increase the deviation of the model near the splitting points.

```
set.seed(111)
max_spline_term = 10
CVerrorSpline_trade_reg = rep(0, max_spline_term)
for (i in 1:max_spline_term) {
  fit = glm(hf_score ~ ns(ef_trade_regulatory, df=i) + . - ef_trade_regulatory, data = hfi_data_combine
  CVerrorSpline_trade_reg[i] <- cv.glm(hfi_data_combined_train, fit, K = 10)$delta[1]
}
plot(CVerrorSpline_trade_reg, main="The CV MSE for the Trade Regulation Splines of Various Degrees",
     xlab = "Degrees of Freedom", ylab = "CV Error", type = "b")
points(which.min(CVerrorSpline_trade_reg), CVerrorSpline_trade_reg[which.min(CVerrorSpline_trade_reg)],
```




```
CVerrorSpline_homicide = rep(0, max_spline_term)
for (i in 1:max_spline_term) {
  fit = glm(hf_score ~ ns(pf_ss_homicide, df=i) + . - pf_ss_homicide, data = hfi_data_combined_train)
  CVerrorSpline_homicide[i] <- cv.glm(hfi_data_combined_train, fit, K = 10)$delta[1]
}
plot(CVerrorSpline_homicide, main="The CV MSE for the Homicide Splines of Various Degrees",
     xlab = "Degrees of Freedom", ylab = "CV Error", type = "b")
points(which.min(CVerrorSpline_homicide), CVerrorSpline_homicide[which.min(CVerrorSpline_homicide)], co
```




```
trade_reg_spline_model = glm(hf_score ~ ns(ef_trade_regulatory, df=which.min(CVerrorSpline_trade_reg)) +
                             data = hfi_data_combined_train)
yhat = predict(trade_reg_spline_model, newdata = hfi_data_combined_test)
trade_reg_spline_model_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

homicide_spline_model = glm(hf_score ~ ns(ef_trade_regulatory, df=which.min(CVerrorSpline_homicide)) +
                         data = hfi_data_combined_train)
yhat = predict(homicide_spline_model, newdata = hfi_data_combined_test)
homicide_spline_model_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)


paste("The Mean TSS of the Test Set: ", round(mean_tss_test, 6))
```

```
## [1] "The Mean TSS of the Test Set:  0.882886"
```

```
paste("MLR Forward Regression with No Polynomial Terms Test MSE: ", round(final_forward_mse, 6))
```

```
## [1] "MLR Forward Regression with No Polynomial Terms Test MSE:  0.042611"
```

```
paste("MLR with Spline of Term Trade Regulation Test MSE: ", round(trade_reg_spline_model_mse, 6))
```

```
## [1] "MLR with Spline of Term Trade Regulation Test MSE:  0.04055"
```

```
paste("MLR with Spline of Term Homicide Test MSE: ", round(homicide_spline_model_mse, 6))
```

```
## [1] "MLR with Spline of Term Homicide Test MSE:  0.039919"
```

The splines with 9 degrees of freedom for trade regulation predictor seems to result in a very accurate model. The test MSE is 0.04055 compared to the overall TSS for the test dataset: 0.882886. Additionally, we tested

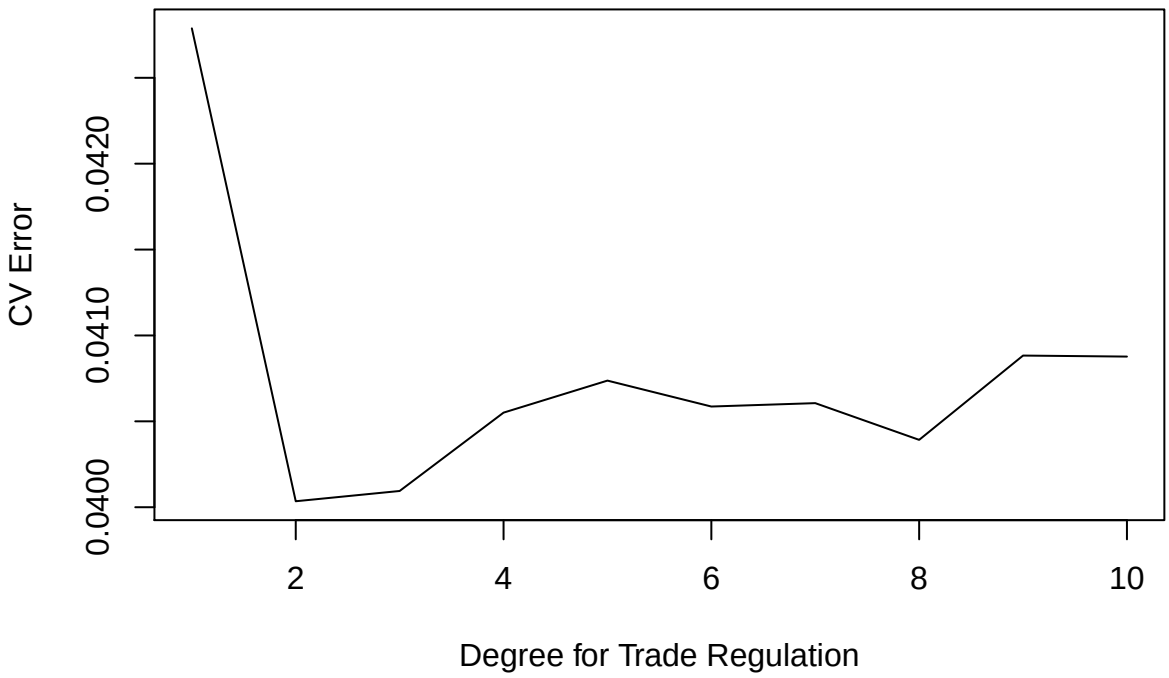
the the spline functions on homicide and resulted in a more accurate model than the trade regulation model (possibly the trade regulation spline model is overfitting). The resulting test MSE is 0.039919 versus the TSS of 0.882886. Overall, this indicates a non-linear relationship similar to the finding with polynomial regression. However, there is added complexity due to the added splines, and with only a small change in explanation of variance in the data (<1% change). Therefore, we prefer the best-subset model over the polynomial and spline models.

4.3 Generalized Additive Models (GAMs)

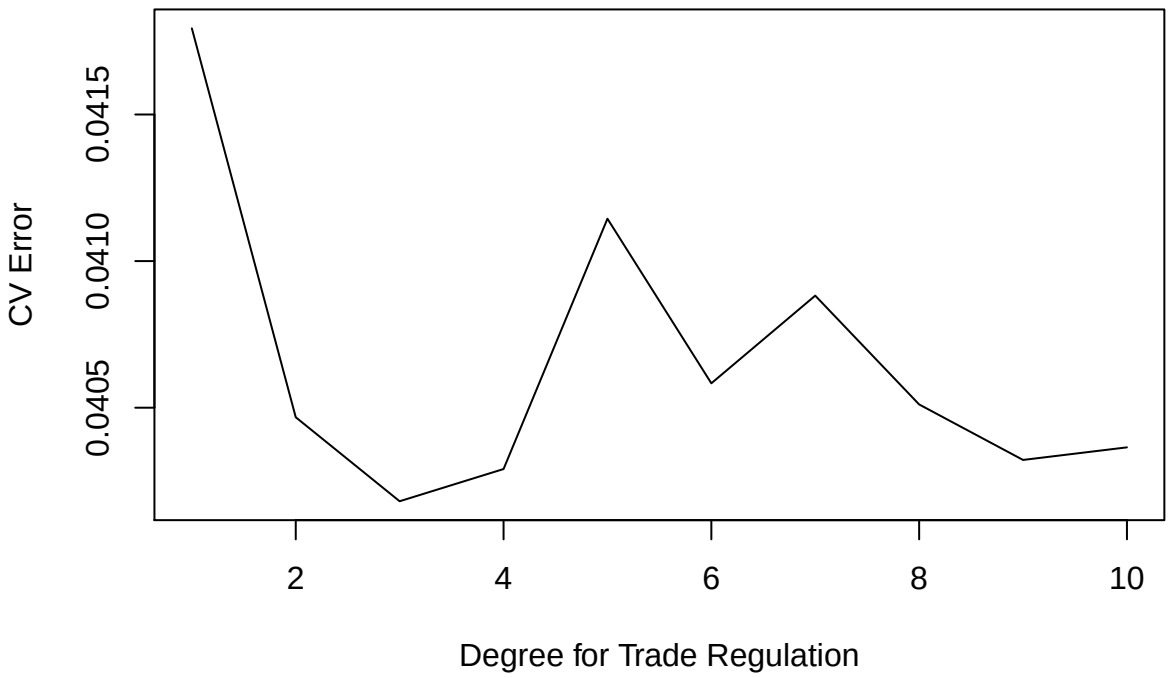
We are going to combine the two splines above with a GAM (Generalized Additive Model) to see if improvements can be made.

```
set.seed(111)

CVerrorGam = matrix(data=rep(rep(0, max_spline_term), max_spline_term), nrow=max_spline_term,
                          ncol = max_spline_term)
for (i in 1:max_spline_term) {
  for (j in 1:max_spline_term){
    fit = gam(hf_score ~ ns(pf_ss_homicide, i) + ns(ef_trade_regulatory, j) + . -
                ef_trade_regulatory - pf_ss_homicide, data = hfi_data_combined_train)
    CVerrorGam[i, j] = cv.glm(hfi_data_combined_train, fit, K = 10)$delta[1]
  }
  plot(CVerrorGam[i, 1:max_spline_term],
       main = paste("For ", i, " Spline Degrees of Freedom for Homicide (Held Constant)"),
       xlab = "Degree for Trade Regulation", ylab = "CV Error", type = "l")
}
```


For 2 Spline Degrees of Freedom for Homicide (Held Constant)
CV Error
0.0400
0.0410
0.0420
2
4
6
8
10
Degree for Trade Regulation


For 3 Spline Degrees of Freedom for Homicide (Held Constant)
CV Error
0.0400
0.0410
0.0420
2
4
6
8
10
Degree for Trade Regulation

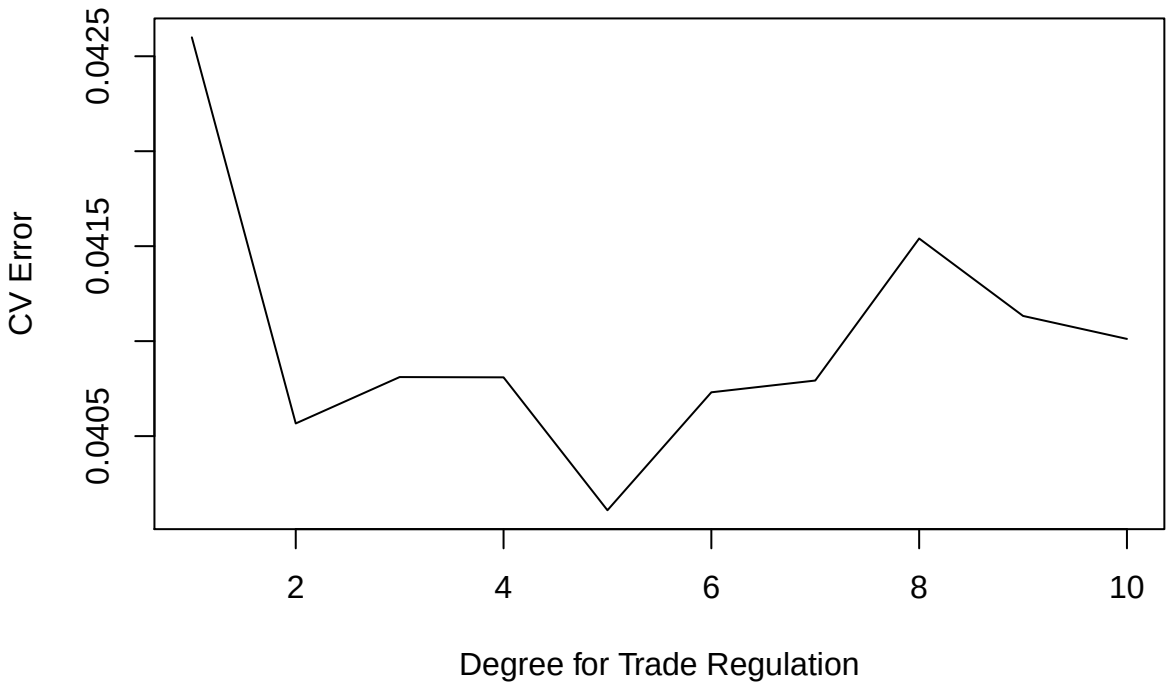

For 4 Spline Degrees of Freedom for Homicide (Held Constant)
CV Error
0.0395
0.0405
0.0415
2
4
6
8
10
Degree for Trade Regulation

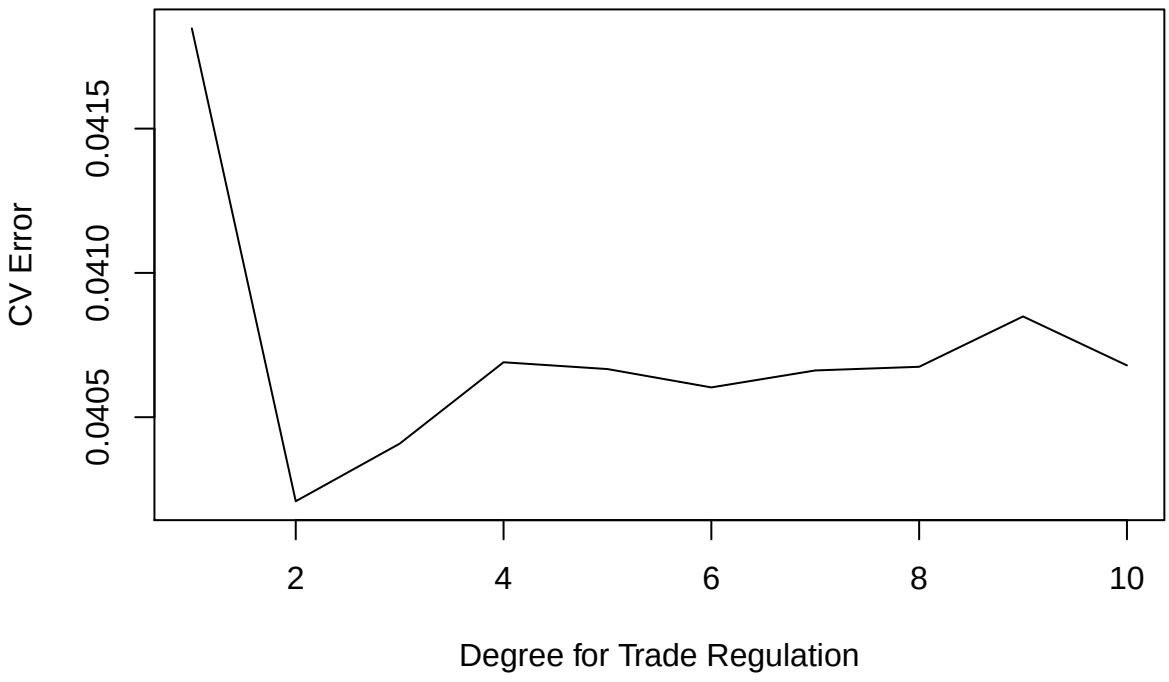

For 5 Spline Degrees of Freedom for Homicide (Held Constant)
CV Error
0.0405
0.0410
0.0415
2
4
6
8
10
Degree for Trade Regulation


For 6 Spline Degrees of Freedom for Homicide (Held Constant)
CV Error
0.0400
0.0410
0.0420
2
4
6
8
10
Degree for Trade Regulation


For 7 Spline Degrees of Freedom for Homicide (Held Constant)
CV Error
0.0405
0.0415
2
4
6
8
10
Degree for Trade Regulation


For 8 Spline Degrees of Freedom for Homicide (Held Constant)
CV Error
0.0405
0.0410
0.0415
0.0420
2
4
6
8
10
Degree for Trade Regulation


For 9 Spline Degrees of Freedom for Homicide (Held Constant)
CV Error
0.0405
0.0415
0.0425
2
4
6
8
10
Degree for Trade Regulation




```
gam_final_best = glm(hf_score ~
                       ns(ef_trade_regulatory, arrayInd(which.min(CVerrorGam), dim(CVerrorGam))[1,2]) +
                       ns(pf_ss_homicide, arrayInd(which.min(CVerrorGam), dim(CVerrorGam))[1,1]) + . - c
                     data = hfi_data_combined_train)

yhat = predict(gam_final_best, newdata = hfi_data_combined_test)
gam_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

paste("The degrees of freedom of spline for Homicide:", arrayInd(which.min(CVerrorGam), dim(CVerrorGam))
```

```
## [1] "The degrees of freedom of spline for Homicide: 4"
```

```
paste("The degrees of freedom of spline for Trade Regulation:", arrayInd(which.min(CVerrorGam), dim(CVe
```

```
## [1] "The degrees of freedom of spline for Trade Regulation: 6"
```

```
paste("The Cross-Validated MSE for the GAMs tested:", round(min(CVerrorGam)), 6)
```

```
## [1] "The Cross-Validated MSE for the GAMs tested: 0 6"
```

```
paste("GAM Test MSE:", round(gam_test_mse, 6))
```

```
## [1] "GAM Test MSE: 0.039921"
```

```
paste("Mean TSS of Test Set:", round(mean_tss_test, 6))
```

```
## [1] "Mean TSS of Test Set: 0.882886"
```

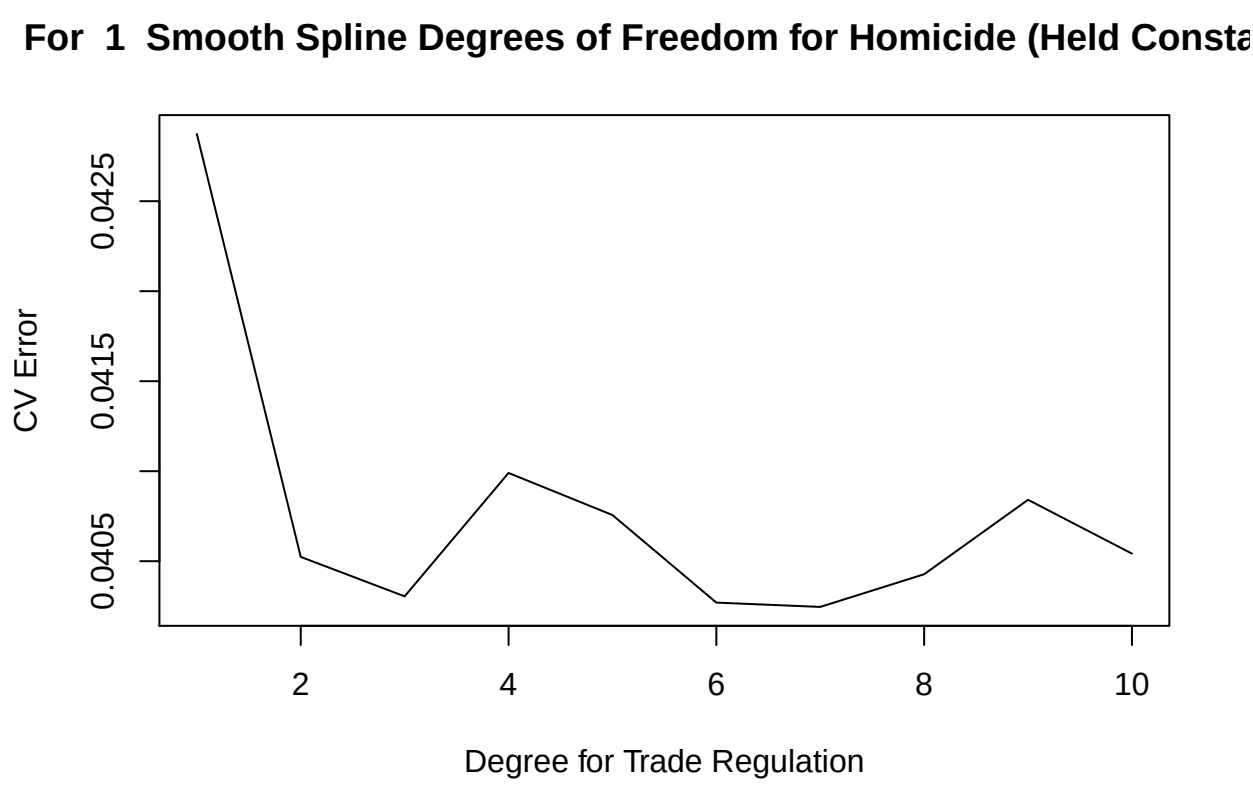
```
paste("MSE of forward selection MLR model:", round(final_forward_mse, 6))
```

```
## [1] "MSE of forward selection MLR model: 0.042611"
```

Using cross-validation to find the best model that does not overfit with the training data with GAM of variable spline DoF for Homicide and Trade Regulation. The test MSE of the lowest CV MSE for the GAMs tested resulted in a very accurate MSE of 0.039921 compared to the mean TSS of the test set of 0.882886.

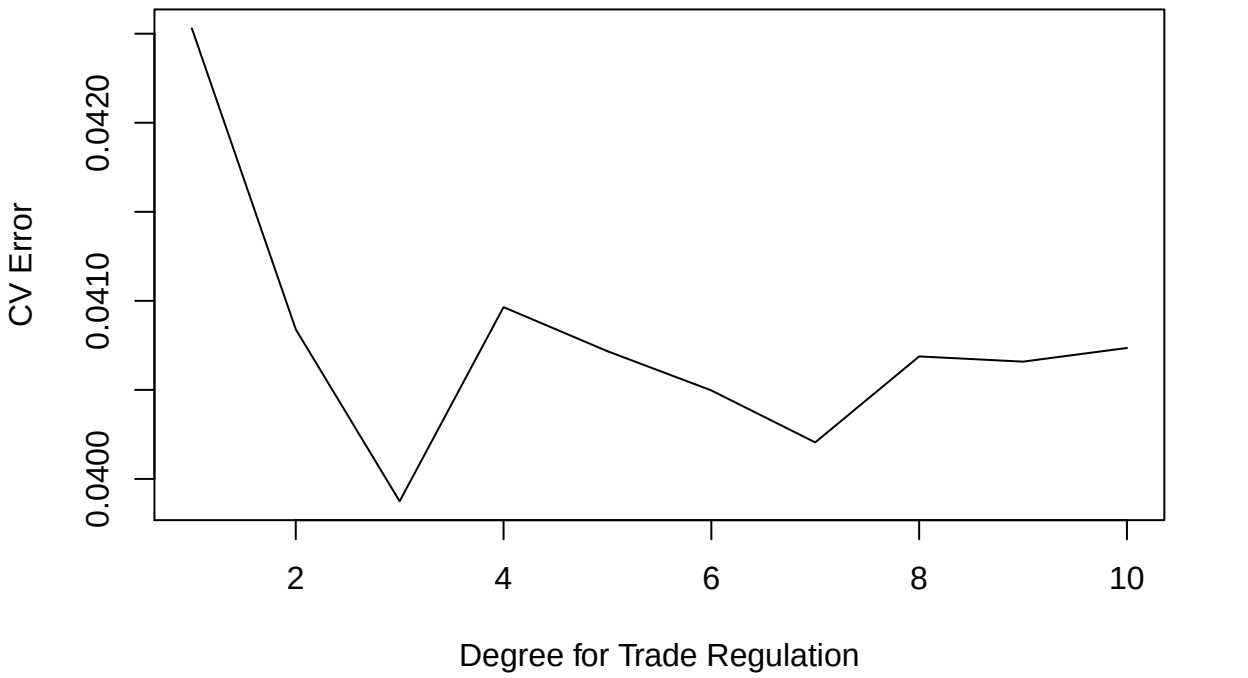
We will now perform the GAMs with SMOOTHING splines to try and reduce the variance where the splines split:

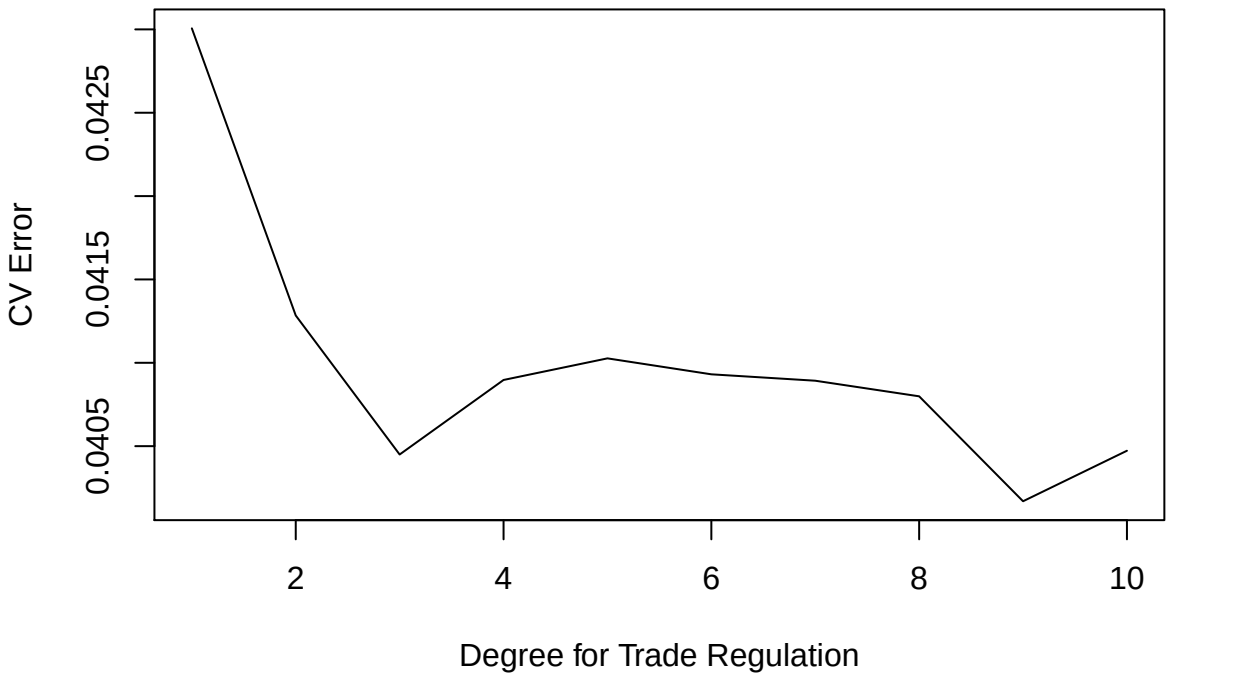
```
CVerrorGamSmooth = matrix(data=rep(rep(0, max_spline_term), max_spline_term), nrow=max_spline_term,
                          ncol = max_spline_term)
for (i in 1:max_spline_term) {
  for (j in 1:max_spline_term){
    fit = gam(hf_score ~ s(pf_ss_homicide, i) + s(ef_trade_regulatory, j) + . -
                ef_trade_regulatory - pf_ss_homicide, data = hfi_data_combined_train)
    CVerrorGamSmooth[i, j] = cv.glm(hfi_data_combined_train, fit, K = 10)$delta[1]
  }
  plot(CVerrorGamSmooth[i, 1:max_spline_term],
       main = paste("For ", i, " Smooth Spline Degrees of Freedom for Homicide (Held Constant)"),
       xlab = "Degree for Trade Regulation", ylab = "CV Error", type = "l")
}
```

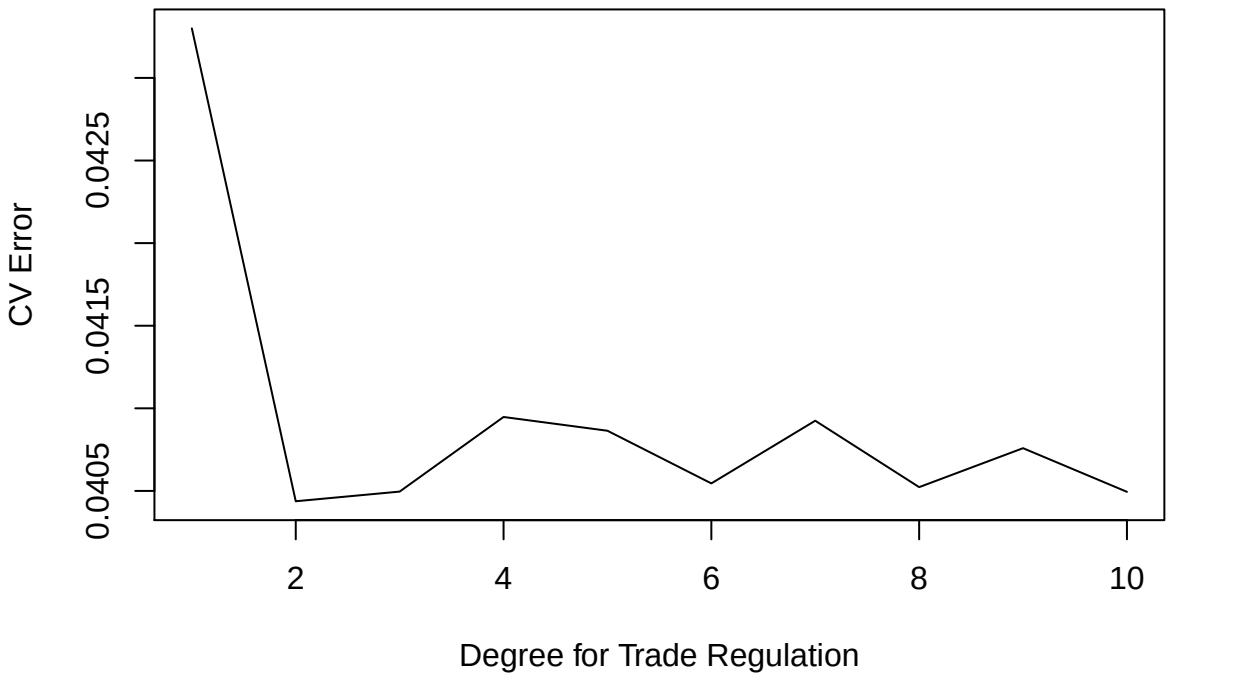


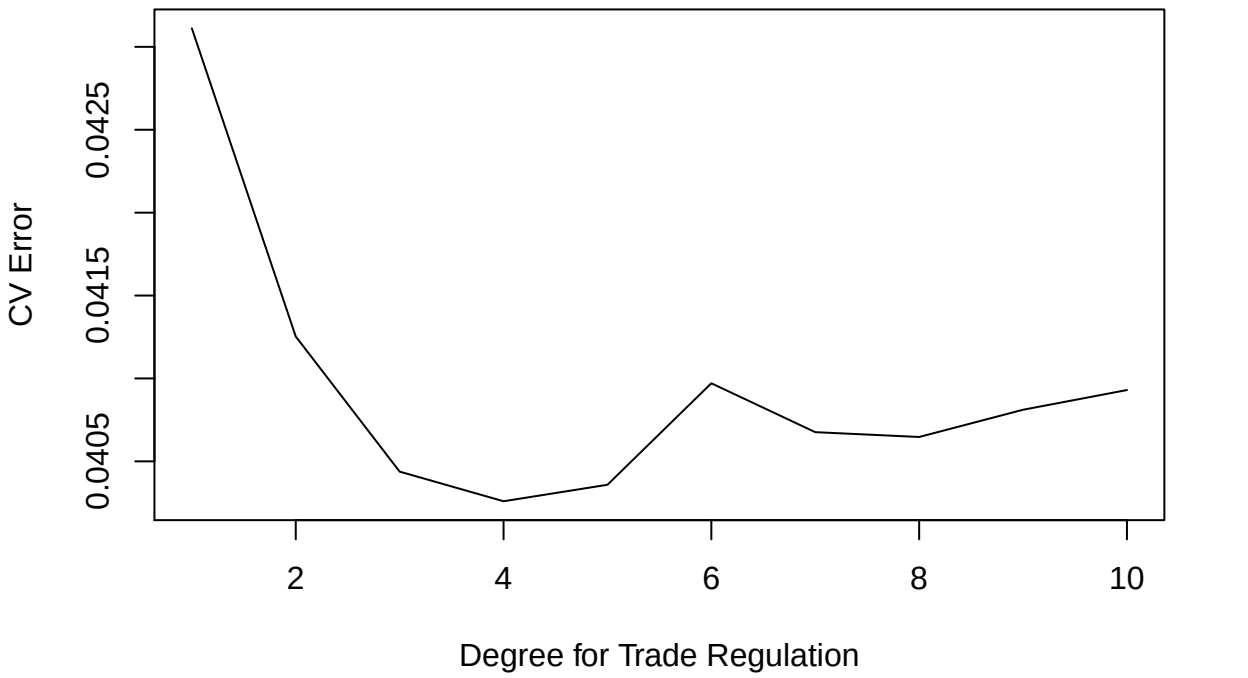

For 2 Smooth Spline Degrees of Freedom for Homicide (Held Consta
CV Error
0.0420
0.0410
0.0400
2
4
6
8
10
Degree for Trade Regulation

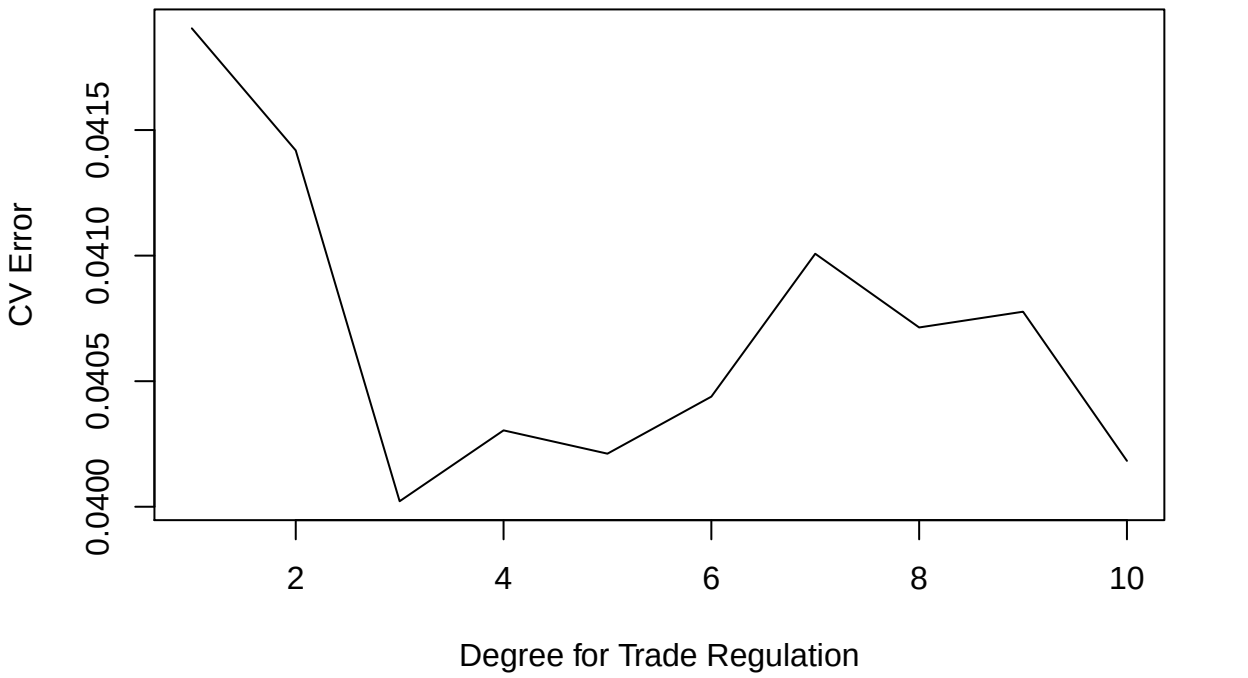

For 3 Smooth Spline Degrees of Freedom for Homicide (Held Consta
CV Error
0.0425
0.0415
0.0405
2
4
6
8
10
Degree for Trade Regulation

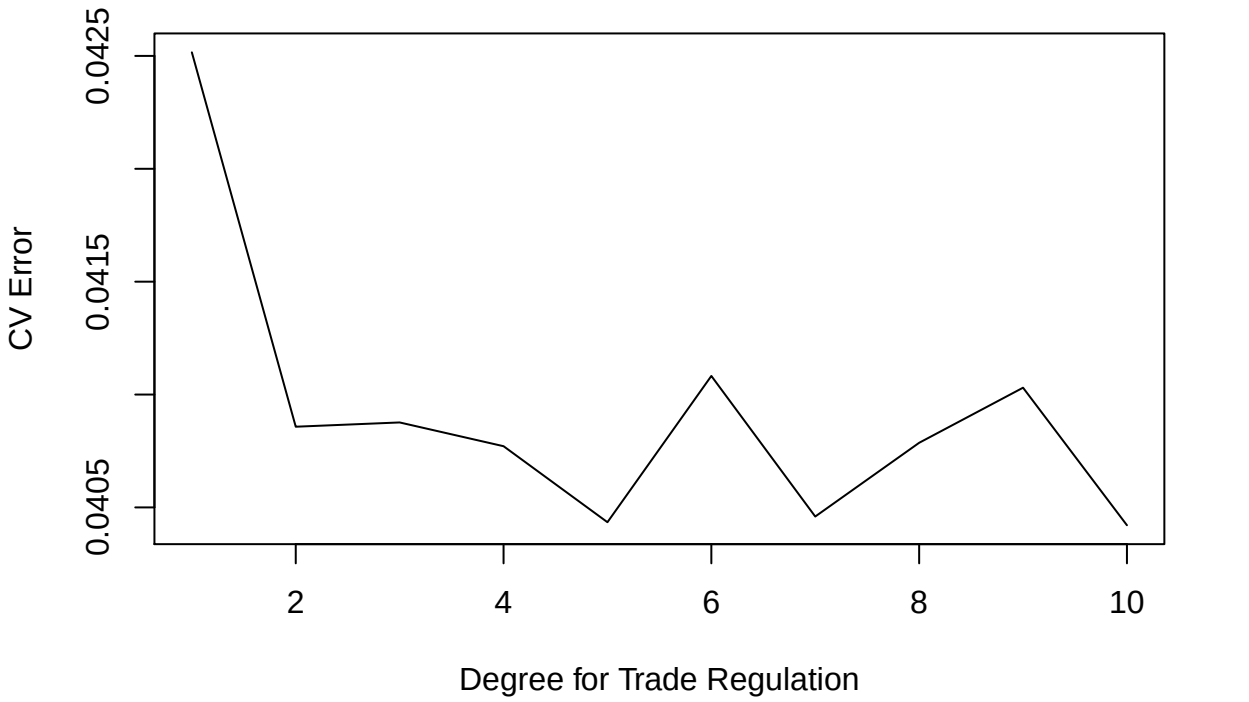

For 4 Smooth Spline Degrees of Freedom for Homicide (Held Consta
CV Error
0.0400
0.0410
0.0420
2
4
6
8
10
Degree for Trade Regulation

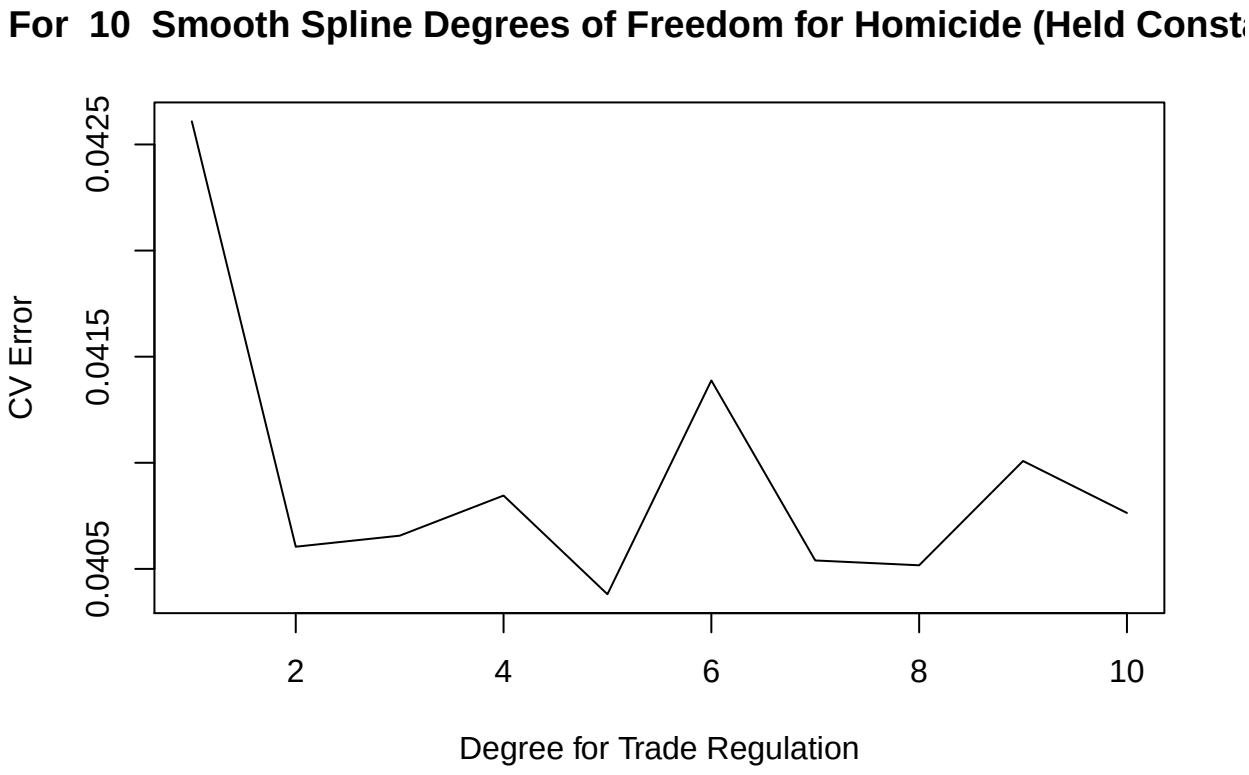

For 5 Smooth Spline Degrees of Freedom for Homicide (Held Consta
CV Error
0.0405
0.0415
0.0425
2
4
6
8
10
Degree for Trade Regulation


For 6 Smooth Spline Degrees of Freedom for Homicide (Held Consta
CV Error
0.0405
0.0415
0.0425
2
4
6
8
10
Degree for Trade Regulation


For 7 Smooth Spline Degrees of Freedom for Homicide (Held Consta
CV Error
0.0405
0.0415
0.0425
2
4
6
8
10
Degree for Trade Regulation


For 8 Smooth Spline Degrees of Freedom for Homicide (Held Consta
CV Error
0.0400
0.0405
0.0410
0.0415
2
4
6
8
10
Degree for Trade Regulation


For 9 Smooth Spline Degrees of Freedom for Homicide (Held Consta
CV Error
0.0425
0.0415
0.0405
2
4
6
8
10
Degree for Trade Regulation




```
gam_smooth_final_best = glm(hf_score ~
                            s(ef_trade_regulatory, arrayInd(which.min(CVerrorGamSmooth), dim(CVerrorGa
                            s(pf_ss_homicide, arrayInd(which.min(CVerrorGamSmooth), dim(CVerrorGamSmoo
                            . - ef_trade_regulatory - pf_ss_homicide, data = hfi_data_combined_train)
yhat = predict(gam_smooth_final_best, newdata = hfi_data_combined_test)
gam_smooth_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)

paste("The Results for SMOOTHING Splines")
```

```
## [1] "The Results for SMOOTHING Splines"
```

```
paste("The degrees of freedom of spline (smoothing) for Homicide:", arrayInd(which.min(CVerrorGamSmooth
```

```
## [1] "The degrees of freedom of spline (smoothing) for Homicide: 2"
```

```
paste("The degrees of freedom of spline (smoothing) for Trade Regulation:", arrayInd(which.min(CVerrorGa
```

```
## [1] "The degrees of freedom of spline (smoothing) for Trade Regulation: 3"
```

```
paste("The Cross-Validated MSE for the GAMs Smoothing Splines tested:", round(min(CVerrorGamSmooth), 6)
```

```
## [1] "The Cross-Validated MSE for the GAMs Smoothing Splines tested: 0.039753"
```

```
paste("GAM (smoothing splines) Test MSE:", round(gam_smooth_test_mse, 6))
```

```
## [1] "GAM (smoothing splines) Test MSE: 0.042125"
```

```
paste("Mean TSS of Test Set:", round(mean_tss_test, 6))
```

```
## [1] "Mean TSS of Test Set: 0.882886"
```

```r
paste("MSE of forward selection MLR model:", round(final_forward_mse, 6))
```

```
## [1] "MSE of forward selection MLR model: 0.042611"
```

Smoothing the splines seem to result in a worse model than the non-smoothed splines. This finding indicates we are biasing the model in a negative way by forcing the smoothing of splines.

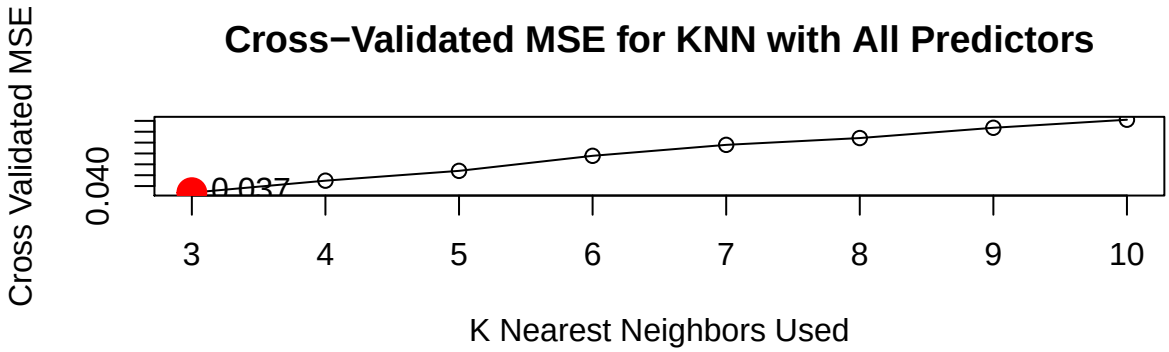
Overall, the resulting GAMs are very accurate and minimize the MSE either equal or marginally better than the best-subset of MLR model. However, the added complexity and variance of the model seems unnecessary for less than a 1% improvement of the test MSE. We prefer the polynomial regression to the GAMs because the splines and GAMs tend to have large degrees of freedom to capture more variance, but it fails to deliver better results (granted the results of plain MLR are already very good).

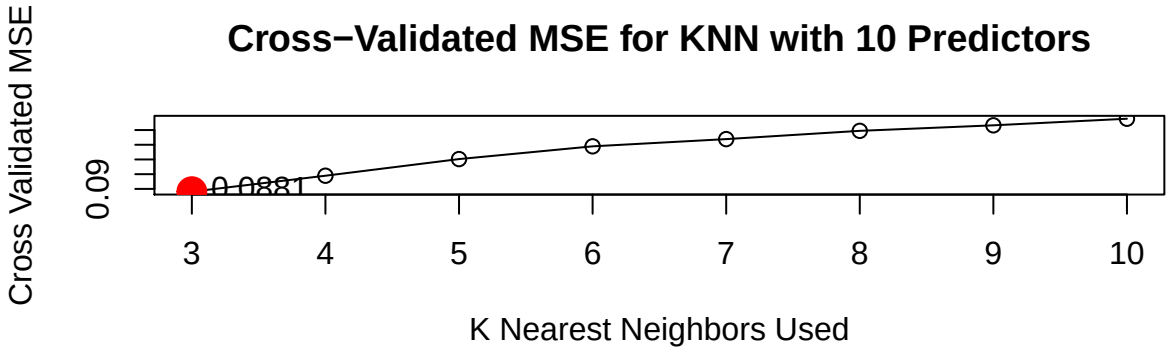
In conclusion, we prefer the best-subset or forward selection models as they tend to be very close in accuracy as the polynomial regression, splines, and GAMs while being simpler.

4.4 KNN Regression

```r
#########################    KNN REGRESSION   ##############################
# simply used to get same results regardless of run. Methods should be robust to the seed changing, tho
set.seed(111)

# Lowest and Highest value of k allowed
k_folds = 10
top_correlated = 11 # really is 10 as hf_score is included; so 11 - 1 = 10 most correlated predictors
lowest_k = 3
highest_k = 10 # could do higher, but the lower the 'K' is, the better the results

# 2 by 1 graphs
par(mfrow=c(2,1))

do_knn = function(title_string, is_top_correlated){
  # the cross-validated MSE of all K
  knn_regression_mse = rep(-1,highest_k - lowest_k + 1)
  for(k in lowest_k:highest_k){
    if(is_top_correlated){
      knn_reg = knn.reg(train = (hfi_data_combined_train %>%
                                   select(c(names(abs_sorted_hf_score_correlations %>% head(top_correla
                        y = hfi_data_combined_train$hf_score, k = k)
    }else{
      knn_reg = knn.reg(train = (hfi_data_combined_train %>% select(-hf_score)),
                        y = hfi_data_combined_train$hf_score, k = k)
    }

    knn_regression_mse[k - lowest_k + 1] = mean((knn_reg$pred - hfi_data_combined_train$hf_score)^2)
  }

  plot(main = title_string, x=lowest_k:highest_k, y = knn_regression_mse,
       xlab = "K Nearest Neighbors Used", ylab = "Cross Validated MSE") +
    lines(x=lowest_k:highest_k, y = knn_regression_mse) +
    points(x = (which.min(knn_regression_mse) + lowest_k - 1),
           y = min(knn_regression_mse), col ="red", cex=2, pch =19) +
    text(x = (which.min(knn_regression_mse) + lowest_k - 1), y = min(knn_regression_mse),
         labels = paste(round(min(knn_regression_mse), 4)), pos = 4)
```

```
  knn_regression_mse
}

# do KNN regression with 1ll 39 predictors
knn_regression_mse_default = do_knn(title_string = "Cross-Validated MSE for KNN with All Predictors",
                                    is_top_correlated = FALSE)



# Try with only limited variables
# this will select the top_correlated highest correlated vars with hf_score
# then attempt the KNN regression on this smaller dataset
knn_regression_mse_diff = do_knn(title_string = paste("Cross-Validated MSE for KNN with", top_correlate
                                 is_top_correlated = TRUE)
```






```
# test the 3-k models
knn_default_best = knn.reg(train = (hfi_data_combined_train %>% select(-hf_score)),
                           test = (hfi_data_combined_test %>% select(-hf_score)),
                           y = hfi_data_combined_train$hf_score,
                           k = (lowest_k - 1 + which.min(knn_regression_mse_default)))
knn_default_test_mse = mean((knn_default_best$pred - hfi_data_combined_test$hf_score)^2)


knn_limit_pred_best = knn.reg(train = (hfi_data_combined_train %>%
                                       select(c(names(abs_sorted_hf_score_correlations %>% head(top_corr
                           test = (hfi_data_combined_test %>%
```

```
                                    select(c(names(abs_sorted_hf_score_correlations %>% head(top_correl
                           y = hfi_data_combined_train$hf_score,
                           k = (lowest_k - 1 + which.min(knn_regression_mse_diff)))
knn_limit_pred_best_mse = mean((knn_limit_pred_best$pred - hfi_data_combined_test$hf_score)^2)


paste("The Predictors Used in the", top_correlated - 1, "Predictor Model:",
      paste(c(names(abs_sorted_hf_score_correlations %>% head(top_correlated))), collapse = ", "))
```

```
## [1] "The Predictors Used in the 10 Predictor Model: hf_score, pf_expression_control, pf_expression_in
```

```
paste("The Mean TSS of the Test Set:", round(mean_tss_test, 6))
```

```
## [1] "The Mean TSS of the Test Set: 0.882886"
```

```
paste("The Test MSE of Forward Selection MLR:", round(final_forward_mse, 6))
```

```
## [1] "The Test MSE of Forward Selection MLR: 0.042611"
```

```
paste("The Test MSE of ALL 39 Predictor Model:", round(knn_default_test_mse, 6))
```

```
## [1] "The Test MSE of ALL 39 Predictor Model: 0.031429"
```

```
paste("The Test MSE of", top_correlated, "Predictor Model:", round(knn_limit_pred_best_mse, 6))
```

```
## [1] "The Test MSE of 11 Predictor Model: 0.108217"
```

```
# 1 by 1 graphs
par(mfrow=c(1,1))
```

There are 2 different KNN regressions. Both regressions are tested on the training set using cross validation (Leave one out cross-validation). We take the best cross-validated K, and create those models. The first KNN model uses all 39 predictors. The second KNN model uses the top 10 correlated predictors only using KNN regression. For both models, the value for 'K' is 3. It seems that the lower the K, the more accurate the model becomes.

- The 39 Predictor model returns an impressive Test MSE of 0.031429. The Test MSE is superior to the MLR Test MSE from forward selection (0.042611).
- The top 10 KNN model returns a less impressive Test MSE of 0.108217 compared to the forward selection model (using ~30 predictors) of 0.042611. Even though the MSE is not as good as the full 39 predictor model nor the MLR model, it does surprisingly well with fewer predictors.

Overall, the KNN regression results in the lowest test MSE seen so far. The model seems to be very accurate, especially with all predictors being used to train the model. However, KNN is unsupervised, and therefore we have very little ability to judge which predictors are most important. Additionally, the test MSE is lower than the MLR, but it only explains ~1-2% more variance within the data. This small difference in model variance explained, is not enough to merit this model over the MLR methods performed beforehand. Thus, we will not focus on KNN as it fails to deliver meaningful insights regarding predictor significance or importance.

An important note is that K=3 is the most variant model we can create. Meaning the data does not seem to overfit with additional variance within a model. This finding indicates the data is heavily correlated and linked together, which it is.

4.5 Conclusions on Non-Linear Methods

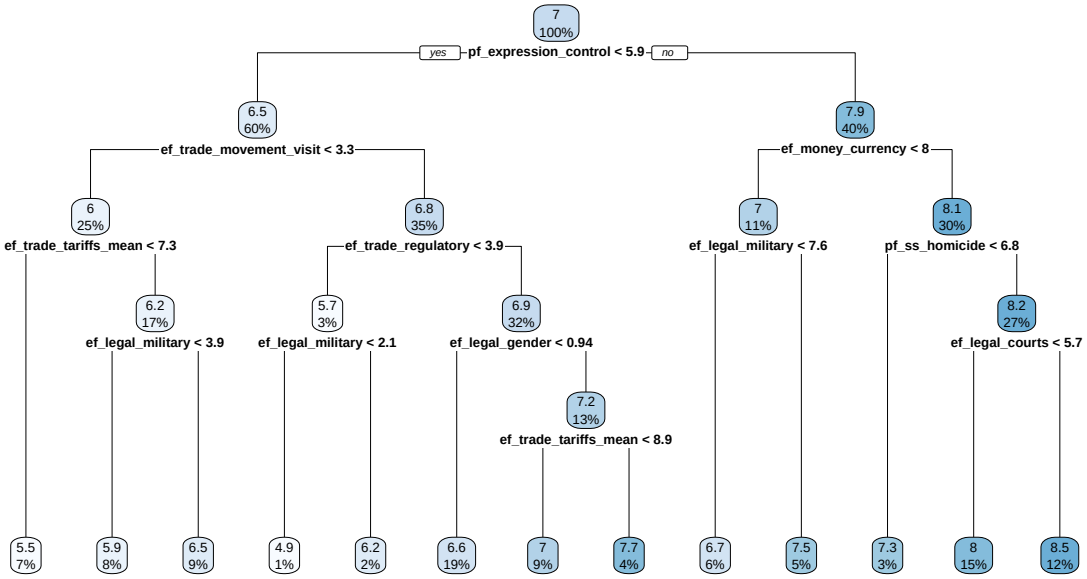
Overall, the non-linear methods due capture more variance than the MLR models, however, the gains in prediction ability (lower MSE) come at the cost of inferencing ability. This inferencing ability for MLR allows us to determine with ease the most significant predictors using T-tests and F-Tests. The polynomial terms

for Homicide and Trade Regulation is something to keep in mind for the final part of our analysis. The polynomial terms in a smaller model may help increase accuracy. The splines, GAMs, and smoothed spline GAMs offered some better models (GAMs with non-smoothed splines), but failed to significantly improve accuracy compared to the forward regression model. In conclusion, we decided to sill prefer the best-subset model over these non-linear models as an improvement of accuracy equal to **less** than 1% is not worth the less inferencing ability and added complexity over the basic MLR.

5. Tree-Based Methods

5.1 Standard Regression Tree

```
# simply used to get same results regardless of run. Methods should be robust to the seed changing, tho
set.seed(111)

par(mfrow=c(1,1))

# Standard Regression Tree
tree_regression = rpart(hf_score ~ ., data=hfi_data_combined_train)
rpart.plot(tree_regression)
```




```
# the tree is the same as above, but rpart has nice plotting
cv_tree_base = tree(hf_score ~ ., data=hfi_data_combined_train)
cv_tree_regression = cv.tree(cv_tree_base, K = k_folds)
plot(cv_tree_regression$size, cv_tree_regression$dev, type="b",
     main = "Tree Regression Errors (cv.tree())",
     ylab = "Cross-Validated Deviation", xlab = "Tree Regression Size") +
  points(cv_tree_regression$size[which.min(cv_tree_regression$dev)],
         min(cv_tree_regression$dev),
         col="red", cex = 2, pch = 19)
```


```
## integer(0)
```

Eight variables are used in constructing the standard tree on the full data set, which has 12 terminal nodes. They are pf_expression_control, ef_trade_movement_visit, ef_trade_tariffs_mean, ef_legal_military, ef_trade_regulatory, ef_legal_gender, ef_trade_tariffs_sd, and ef_legal_courts.

We obtain 10-fold cross validated errors for trees of different sizes, which shows that the most complicated tree (13 terminal nodes) has the lowest 10-fold CV deviance. This tree was grown by selecting splits that maximize the reduction in training MSE, and splitting continued until 12 terminal nodes, at which point the terminal nodes are too small or too few to be split.

No pruning is needed as the cross-validated tree shows it performs best with the most complex model.

A further look into the tree:

```
## No pruning is needed. Most complex model is best model.
summary(cv_tree_base)
```

```
##
## Regression tree:
## tree(formula = hf_score ~ ., data = hfi_data_combined_train)
## Variables actually used in tree construction:
## [1] "pf_expression_control"   "ef_trade_movement_visit"
## [3] "ef_trade_tariffs_mean"   "ef_legal_military"
## [5] "ef_trade_regulatory"     "ef_legal_gender"
## [7] "ef_money_currency"       "pf_ss_homicide"
## [9] "ef_legal_courts"
## Number of terminal nodes:  13
```

```
## Residual mean deviance:  0.1466 = 151.1 / 1031
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.557000 -0.211900  0.003695  0.000000  0.215800  1.482000
```

```
# Test on the test set
tree_predictions = predict(cv_tree_base, newdata = hfi_data_combined_test)
pruned_tree_test_mse = mean((tree_predictions - hfi_data_combined_test$hf_score) ^ 2)
```

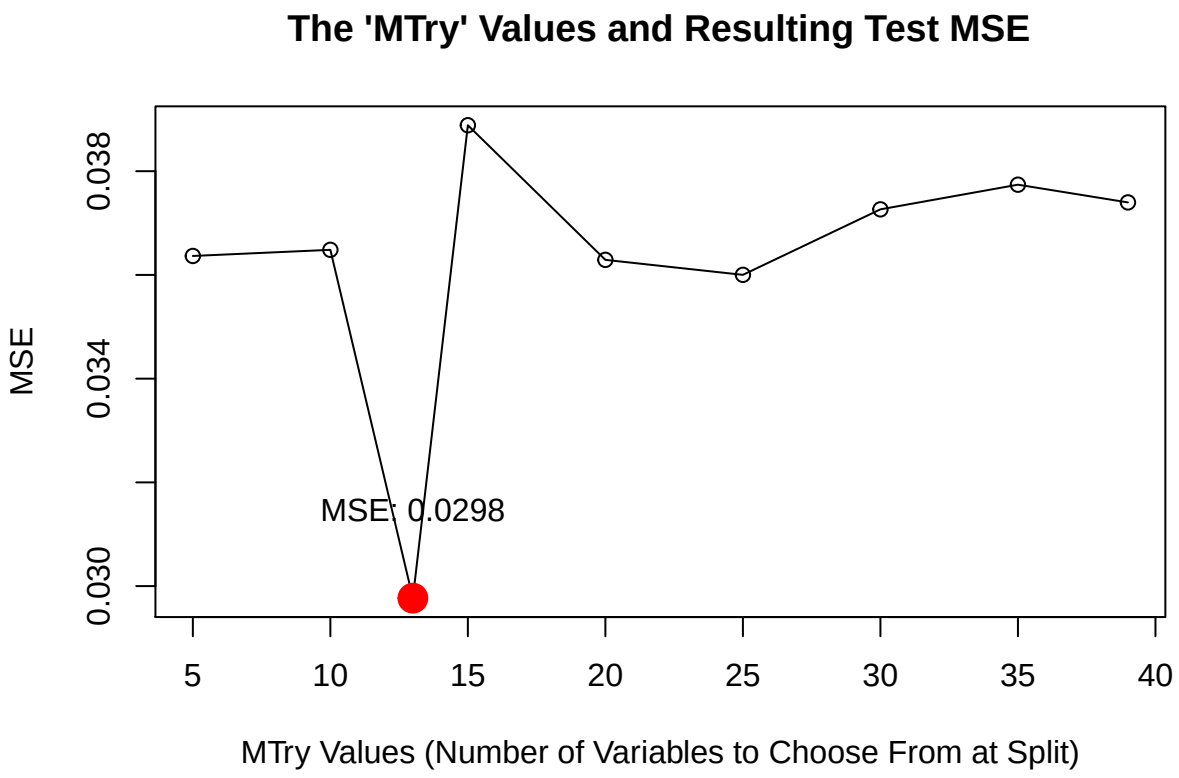
The "Residual Mean Deviance" is the MSE and it does significantly worse than the linear models and dimension reduction models (0.14 versus < 0.05; smaller the better). The test MSE is 0.168124 which again makes the tree look like a much worse model than the linear models and non-linear models above.

5.2 Bagging and Random Forests

```
# RANDOM FOREST AND BAGGING
set.seed(111)
default_ntrees = 500


# Store the TEST MSE for all MTrys we are testing
random_forests = data.frame(MTry = numeric(9), Test_MSE = numeric(9))
forest_i = 1
for (current_mtry in seq(from = 5, to = 35, by = 5)){
  random_forest = randomForest(hf_score ~ ., data = hfi_data_combined_train,
                        ntree = default_ntrees, mtry = ncol(hfi_data_combined_train) - 1)
  yhat = predict(random_forest, newdata = hfi_data_combined_test)
  random_forests$MTry[forest_i] = current_mtry
  random_forests$Test_MSE[forest_i] = mean((yhat - hfi_data_combined_test$hf_score)^2)
  forest_i = forest_i + 1
}


# BAGGING (MTY == 39 or all possible predictors)
random_forest = randomForest(hf_score ~ ., data = hfi_data_combined_train,
                        ntree = default_ntrees, mtry = ncol(hfi_data_combined_train) - 1)
# bagging; Subtract 1 for the hf_score response in the dataframe
random_forests$MTry[forest_i] = ncol(hfi_data_combined_train) - 1
yhat = predict(random_forest, newdata = hfi_data_combined_test)
random_forests$Test_MSE[forest_i] = mean((yhat - hfi_data_combined_test$hf_score)^2)
forest_i = forest_i + 1


# p/3; this is the recommended value of 'MTry'
recommended_forest = randomForest(hf_score ~ ., data = hfi_data_combined_train,
                        ntree = default_ntrees, mtry = round((ncol(hfi_data_combined_train) - 1)/ 3),
                        importance = TRUE)

random_forests$MTry[forest_i] = round((ncol(hfi_data_combined_train) - 1)/ 3)
yhat = predict(recommended_forest, newdata = hfi_data_combined_test)
random_forests$Test_MSE[forest_i] = mean((yhat - hfi_data_combined_test$hf_score)^2)
forest_i = forest_i + 1
```

```r
random_forests = random_forests %>% arrange(MTry)

plot(x = random_forests$MTry, y = random_forests$Test_MSE, type = "line",
     main = "The 'MTry' Values and Resulting Test MSE", ylab = 'MSE',
     xlab = "MTry Values (Number of Variables to Choose From at Split)") +
  points(x = random_forests$MTry, y = random_forests$Test_MSE) +
  points(x = random_forests$MTry[which.min(random_forests$Test_MSE)],
         y = min(random_forests$Test_MSE), col="red", cex = 2, pch = 19) +
  text(x = random_forests$MTry[which.min(random_forests$Test_MSE)],
       y = min(random_forests$Test_MSE),
       labels = paste("MSE:", round(min(random_forests$Test_MSE),4) ),
       pos = 3, offset = 2)
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to first
## character
```




```
## integer(0)
```

```r
# Importance from Recommended MTry of p/3

paste("The Mean TSS for the Test Set:", round(mean_tss_test, 6))
```

```
## [1] "The Mean TSS for the Test Set: 0.882886"
```

```r
paste("The Test MSE for the Forward Selection MLR Model:", round(final_forward_mse, 6))
```

```
## [1] "The Test MSE for the Forward Selection MLR Model: 0.042611"
```

```
paste("The following Test MSE for Random Forest of Varying MTrys:")
```

```
## [1] "The following Test MSE for Random Forest of Varying MTrys:"
```

```
random_forests
```

```
##   MTry   Test_MSE
## 1    5 0.03636563
## 2   10 0.03648366
## 3   13 0.02976794
## 4   15 0.03888583
## 5   20 0.03629059
## 6   25 0.03600169
## 7   30 0.03726412
## 8   35 0.03773972
## 9   39 0.03739804
```

```
a = as.data.frame(recommended_forest$importance)
colnames(a)[1] = "IncMSE"
a %>% arrange(desc(IncMSE)) %>% head(20)
```

```
##                                      IncMSE IncNodePurity
## pf_expression_control             0.16925608    192.417524
## pf_expression_influence           0.13618930    153.811308
## ef_trade_tariffs_mean             0.11450204    105.314494
## ef_legal_gender                   0.10613598     61.401821
## ef_trade_regulatory               0.07248463     77.953543
## ef_trade_regulatory_compliance    0.06975010     75.675454
## ef_legal_military                 0.06780429     62.870452
## ef_trade_movement_visit           0.05412679     32.715250
## pf_ss_homicide                    0.04440094     23.590177
## ef_regulation_credit              0.03503951     28.255045
## ef_trade_tariffs                  0.03469401     22.007846
## ef_money_currency                 0.03413209     22.284553
## ef_trade_movement_capital         0.03400717     24.703913
## ef_legal_enforcement              0.03025537     16.237776
## ef_money_sd                       0.02537540     19.280493
## ef_regulation_business_compliance 0.02228364     11.541355
## ef_legal_courts                   0.02167767     13.055406
## pf_identity_sex_female            0.02011653     10.061894
## pf_identity_sex_male              0.01926254      7.123270
## ef_trade_tariffs_sd               0.01462112      8.294682
```

Bagging is a random forest with the MTry of all possible predictors. This model often can result in good results as it is high in variance or can lead to an overfit of a model. My guess, since the MSE is worse than all other MTry values, there was an overfit issue. Additionally, the random forest models tend to stagnate between 10, 13, and 15 predictors. This finding is in line with the typical thinking of the ideal Mtry is roughly 1/3 the predictors (13 predictors in this case).

The bagged model results in an MSE of 0.037398. The best model with the lowest MSE was 0.029768.

Like KNN, this model results in a very low test MSE, meaning that it is very good for prediction. But because the random forest is complex, there is limited inferencing available. However, random forest does have a proxy for significant predictors within the model by measuring importance. The importance is measured by how much of the model variance is explained by a singular predictor. This measurement is calculated by removing the predictor from the model and seeing how much worse the resulting MSE would be. This importance is not nearly as mathematical and robust as the T-tests or F-tests for MLR, unfortunately.

The most important predictors are:

- pf_expression_control (the control over citizen personal expression like speech)
- pf_expression_influence (the amount of government influence on expression like ownership of media)
- ef_legal_gender (how equal the economic system is for men versus women)
- ef_trade_tariffs_mean (how large the average tariffs are)
- ef_trade_regulatory (how much economic and trade regulation there is within country)

We will keep these predictors in mind for our final analysis.

5.3 Boosting

```
# BOOSTING
set.seed(111)

exponents = seq(-3, 0, by = 0.1)
lambda = 0.02  # from preliminary tests, this seems consistently good; reduces time significantly
default_ntrees = 3000
max_inter_depth = 3  # this is more reasonable, typically, one does not need many leaf nodes per tree
max_inter_depth = 5  # this is excessive. Will take a while. Only used to get overfitting in model.

boosting_test_mse = rep(NA, 4)

paste("The Mean TSS for the Test Set:", round(mean_tss_test, 6))
```

```
## [1] "The Mean TSS for the Test Set: 0.882886"
```

```
paste("The MSE for the Forward Selection MLR Model:", round(final_forward_mse, 6))
```

```
## [1] "The MSE for the Forward Selection MLR Model: 0.042611"
```

```
for( inter_depth in 1:max_inter_depth){
  boosting_gbm = gbm(hf_score ~ ., data = hfi_data_combined_train,
                     distribution="gaussian", interaction.depth = inter_depth,
                     n.trees=default_ntrees, shrinkage = lambda,
                     cv.folds = 10)

  yhat = predict(boosting_gbm, newdata = hfi_data_combined_test, n.trees = default_ntrees)
  boosting_test_mse[inter_depth] = mean((yhat - hfi_data_combined_test$hf_score)^2)

  print(paste("At Maximum Inter-Depth: ", inter_depth))
  print(paste("10-Fold Cross-Validation Error Estimate is",
              round(boosting_gbm$cv.error[default_ntrees], 6)))
  print(paste("Test MSE:", round(boosting_test_mse[inter_depth], 6)))
  print(paste("Mean TSS of Test Set:", round(mean_tss_test, 6)))
}
```

```
## [1] "At Maximum Inter-Depth:  1"
## [1] "10-Fold Cross-Validation Error Estimate is 0.037806"
## [1] "Test MSE: 0.038161"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  2"
## [1] "10-Fold Cross-Validation Error Estimate is 0.026671"
## [1] "Test MSE: 0.024575"
```

```
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  3"
## [1] "10-Fold Cross-Validation Error Estimate is 0.023438"
## [1] "Test MSE: 0.021805"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  4"
## [1] "10-Fold Cross-Validation Error Estimate is 0.020807"
## [1] "Test MSE: 0.019383"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  5"
## [1] "10-Fold Cross-Validation Error Estimate is 0.020529"
## [1] "Test MSE: 0.0198"
## [1] "Mean TSS of Test Set: 0.882886"
```

```
boosting_depth_4 = gbm(hf_score ~ ., data = hfi_data_combined_train,
                       distribution="gaussian", interaction.depth = 4,
                       n.trees=default_ntrees, shrinkage = lambda,
                       cv.folds = 10)

boosting_depth_1 = gbm(hf_score ~ ., data = hfi_data_combined_train,
                       distribution="gaussian", interaction.depth = 1,
                       n.trees=default_ntrees, shrinkage = lambda,
                       cv.folds = 10)

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

Here we perform boosting on the training set with `default_ntrees` trees with an initial round for each model selection. Since boosting takes a significant amount of time to run, we did some testing that seemed to indicate 0.02 is a good lambda value to use. The initial lambda value we used was 0.01, so very close to the standard. The shrinkage (lambda) parameter dictates how 'fast' the boosting model 'learns.' Lower the shrinkage, the slower the model learns; larger, the faster. Obviously faster learning sounds strictly better, but that is not the case. If the learning is too fast, then the model tends to overfit. If the model learns too slow, the model does not have enough iterations of tree creation to capture the variance.

We perform boosting with different tree depths allowed. The default boosting is with trees generated with a singular leaf node (inter-depth = 1). As the inter-depth increases, the amount of variance the model can capture increases as well. Again, overfit becomes an issue. We limit the max leaf nodes at 1 (default), 2, 3, and 4. We then find the MSE of the Test Set to see if there is overfitting concerns.

The test MSE of different inter_depths can be found in the output above (focus on the Test MSE as that is testing the model on new data). The important takeaways:

- At about inter-depth = 4 (aka, 4 leaf nodes per tree), the Test MSE is minimized the most. After the inter-depth > 4, model overfitting seems to occur or stagnation in the Test MSE.
- We decide on the inter-depth of 4 as it has a very low test MSE compared to the other 9 models (not the lowest, but nearly). The other models that perform better on the test set, have minimal improvements.
- Boosting does not offer much insight into the most significant predictors, but we can use the importance metric (same as Random Forest's importance metric) to gauge roughly the significance. These importance components of the model are the best insights we have regarding the black box boosting is.

Overall, boosting does better than the MLR forward selection model's Test MSE of 0.042611 compared to boosting models' Test MSEs of around ~0.02. Still, linear models have more use to us because of T-tests and F-tests, but these tree models (random forests and boosting) do deliver some impressive results and some insights. The insights on predictor importance are below:

5.4 Predictor Importance from Random Forests and Boosting

We ignore the simple decision tree regression model as the MSE was disappointing. All other models above outperformed the tree model.

We choose the recommended MTry size of 13 predictors (1/3 number of predictors) as the results were the best (or very marginally off of the best). ### The following output is the **most Important** predictors within the random forest model with MTry = 13 (recommended value): S

```
top_n_important_preds = 20

# show the summary() of the RF (Mtry = 13)
rf_imp = as.data.frame(recommended_forest$importance)
rf_imp %>% dplyr::arrange(desc(rf_imp$`%IncMSE`)) %>% head(top_n_important_preds)
```

```
##                                    %IncMSE IncNodePurity
## pf_expression_control           0.16925608    192.417524
## pf_expression_influence         0.13618930    153.811308
## ef_trade_tariffs_mean           0.11450204    105.314494
## ef_legal_gender                 0.10613598     61.401821
## ef_trade_regulatory             0.07248463     77.953543
## ef_trade_regulatory_compliance  0.06975010     75.675454
## ef_legal_military               0.06780429     62.870452
## ef_trade_movement_visit         0.05412679     32.715250
## pf_ss_homicide                  0.04440094     23.590177
## ef_regulation_credit            0.03503951     28.255045
## ef_trade_tariffs                0.03469401     22.007846
## ef_money_currency               0.03413209     22.284553
## ef_trade_movement_capital       0.03400717     24.703913
## ef_legal_enforcement            0.03025537     16.237776
## ef_money_sd                     0.02537540     19.280493
## ef_regulation_business_compliance 0.02228364   11.541355
## ef_legal_courts                 0.02167767     13.055406
## pf_identity_sex_female          0.02011653     10.061894
## pf_identity_sex_male            0.01926254      7.123270
## ef_trade_tariffs_sd             0.01462112      8.294682
```

We found boosting to yield very accurate models. We use the inter-depth (number of leaf nodes) of value = 4 as the results were marginally off of the best boosting model with inter-depth 8 (significantly more complexity, which is unnecessary).

The following output is the most Influential (or Important) predictors within the boosting model with inter-depth = 4:

```
# the plots for boosting summary do not work well with our labels
summary(boosting_depth_4, plotit=FALSE) %>% select(rel.inf) %>% head(top_n_important_preds)
```

```
##                                  rel.inf
## pf_expression_control          20.3484115
## ef_trade_tariffs_mean          12.2486287
## ef_trade_regulatory_compliance  9.4910075
## pf_expression_influence         7.4997698
## ef_trade_regulatory             7.4691922
## ef_legal_gender                 7.1773834
## ef_legal_military               4.1598740
## ef_regulation_credit            4.0093249
```

```
## ef_trade_movement_visit         3.1976280
## ef_money_currency               2.8441064
## ef_money_sd                     2.3933488
## pf_ss_homicide                  2.2872051
## ef_trade_movement_capital       2.1987477
## pf_movement_foreign             1.8033758
## ef_legal_courts                 1.5460861
## pf_identity_sex_female          1.4883787
## ef_legal_enforcement            1.3751853
## pf_movement_domestic            0.9258235
## pf_identity_sex_male            0.9207515
## ef_money_inflation              0.6822446
```

We additionally use the default boosting model of inter-depth 1 (simple stub trees with only 1 leaf node created). This hyper-parameter for inter-depth is commonly used. We also got very promising results from this basic boosting model.

The following output is the most Influential (or Important) predictors within the boosting model with inter-depth = 1:

```
summary(boosting_depth_1, plotit=FALSE) %>% select(rel.inf) %>% head(top_n_important_preds)
```

```
##                                   rel.inf
## pf_expression_control          16.6045671
## ef_trade_tariffs_mean          11.6735087
## ef_trade_regulatory_compliance 10.2401482
## ef_legal_gender                 9.4644370
## pf_expression_influence         8.6794647
## ef_trade_regulatory             7.4036403
## ef_legal_military               4.8357756
## ef_money_sd                     4.7248777
## ef_trade_movement_capital       3.2960017
## ef_regulation_credit            2.6468564
## ef_trade_movement_visit         2.4547669
## ef_money_currency               1.8048650
## pf_identity_sex_male            1.7628074
## ef_legal_enforcement            1.6710180
## pf_movement_foreign             1.5850447
## ef_legal_courts                 1.4862318
## pf_ss_homicide                  1.4284188
## pf_movement_domestic            1.2903832
## pf_identity_sex_female          1.2670191
## ef_money_inflation              0.6961354
```

There seem to be some overlap in predictor relevance (significance or importance; all mean roughly the same) between the 3 models above. We will further analyze these results combined with the predictors found to be significant in other models (namely the MLR model found using forward selection and best-subset selection).

6. Conclusions from Analysis for Simplified Model

This section will look at the overall results and findings from the completed methods and analyses above. This section will primarily target task 2 from our goals for the project (identify the most significant or relevant or influential predictors regarding human freedom). Then, we will attempt to simplify our models with the knowledge that certain predictors seem to be important, while other predictors less so.

6.1 Initial Conclusions

Overall, many methods we used seemed to yield very accurate results (consistently more than 90% of the variance was explained). These findings help confirm that the human freedom score we were predicting is made up of these more specific predictors/variables. Potentially, the human freedom score is some weighted average of all 39 (maybe more predictors that we removed at the beginning during data tidying). The CATO Institute has a 400+ page manual and description of their Human Freedom Index. It details how certain variables (predictors within our model) are calculated, but we could not derive their method of calculating the final human freedom score (hf_score).

Our preliminary conclusions are:

1. The Human Freedom Index has many predictors, many of which may be unnecessary. We tested various models with only 10 or 15 predictors, and yielded very accurate models (roughly equivalent to 85% R-Squared). Thus, we will later craft a simplified model that uses 10 - 20 predictors to minimize the MSE (maximize variance explained) and allow an easier model to comprehend as a human.
2. Certain predictors tend to be used in all models. Such as: pf_ss_homicide, pf_expression_control, ef_trade_regulatory / ef_trade_regulatory_compliance, pf_expression_influence, ef_legal_gender. Many of these predictors make sense for the high correlation with human freedom. If people are likely to be murdered, the human freedom of that country is probably lower. On the other hand, if media and expression (like art, discussions, public meetings) are not controlled by the government (like no censorship), then human freedom tends to be greater.

6.2 The Predictor Analysis

We will highlight the various models that performed very well in an effort to highlight some of the most important / influential / significant predictors for human freedom. Then, we can use these findings to create as smaller, simplified model using a subset of the 39 total predictors.

Boosting Models' Findings Regarding Predictors:

The best model we found was the boosting model of inter-depth = 4 (inter-depth of 1 was also very accurate) . The boosting model performed with an MSE around 0.02 (which equates to an R-Squared = ~97%). This accuracy is impressive and the predictors it viewed most influential on the model were pf_expression_control, ef_trade_tariffs_mean, ef_trade_regulatory_compliance, pf_expression_influence, ef_trade_regulatory, ef_legal_gender, ef_legal_military, ef_regulation_credit, ef_trade_movement_visit, ef_money_currency. We used the output from the above models (5.4) regarding rel.inf (relative influence on human freedom).

Random Forest Models' Findings Regarding Predictors:

Another very accurate model was the Random Forest model with Mtry of 13. The predictors that the model gave the greatest importance level were: pf_expression_control, pf_expression_influence, ef_legal_gender, ef_trade_tariffs_mean, ef_trade_regulatory, ef_trade_regulatory_compliance, ef_legal_military, ef_trade_movement_visit, pf_ss_homicide, ef_regulation_credit. We used the output from above models (5.4) regarding the importance metric for estimating the importance of a predictor on human freedom.

Multiple Linear Regression Models' Findings Regarding Predictors:

However, the basic Multiple Linear Regression model from forward selection and best-subset selection performed near these other 2 models with an MSE of roughly 0.042 for an R-Squared of roughly 95%. Very accurate model and very good for predictor inference. The predictors the MLR model found to be most significant were: pf_ss_homicide, ef_legal_gender, ef_legal_courts, ef_legal_military, ef_money_currency, ef_government_consumption, ef_legal_enforcement, pf_identity_sex_male, ef_trade_movement_visit, pf_movement_domestic, pf_movement_foreign, ef_money_sd, pf_identity_sex_female, ef_trade_tariffs_mean, pf_expression_influence, pf_expression_control. We used the output from above models (3.3.2) focusing on the T-values associated with predictors (cutoff was t-value > 5).

Correlation Matrix from Exploratory Analysis Findings:

```
temp_df = as.data.frame(actual_correlation_matrix)
temp_df %>% select(hf_score) %>% arrange(desc(abs(hf_score)))
```

```
##                                    hf_score
## hf_score                         1.00000000
## pf_expression_control            0.75864286
## pf_expression_influence          0.74577981
## ef_legal_military                0.73080473
## ef_trade_regulatory              0.70738841
## ef_trade_regulatory_compliance   0.66323390
## ef_legal_gender                  0.61042279
## ef_trade_tariffs_mean            0.60176553
## pf_movement_domestic             0.55618432
## ef_money_sd                      0.54660088
## ef_regulation_credit             0.53899435
## pf_movement_foreign              0.53299140
## ef_money_currency                0.52859876
## ef_trade_movement_capital        0.49873491
## ef_trade_movement_visit          0.48978923
## ef_legal_enforcement             0.48870597
## pf_identity_sex_male             0.48803096
## pf_ss_disappearances_disap       0.47633648
## ef_regulation_business_start     0.47545342
## pf_identity_sex_female           0.47473262
## ef_trade_tariffs                 0.46449890
## ef_legal_courts                  0.45508684
## ef_money_inflation               0.42271581
## ef_regulation_business_compliance 0.40082096
## ef_government_consumption       -0.32431183
## pf_ss_disappearances_fatalities  0.30751674
## pf_ss_disappearances_violent     0.29877319
## ef_money_growth                  0.29192385
## pf_ss_homicide                   0.27909561
## ef_regulation_labor_conscription 0.26714762
## ef_trade_black                   0.26337282
## pf_ss_disappearances_injuries    0.24888615
## ef_regulation_credit_private     0.19452600
## pf_expression_jailed             0.18793156
## pf_expression_killed             0.16200392
## pf_religion_restrictions         0.13621943
## ef_regulation_labor_minwage      0.12691352
```

```
## pf_religion_harassment            0.10589262
## ef_trade_tariffs_sd               0.06847924
## ef_regulation_labor_hours         0.06347336
```

The correlation matrix from the beginning of the analysis highlights many predictors are significantly correlated with human freedom score. Interestingly, the pf_expression_control and pf_expression_influence are both highly correlated with hf_score, but the linear model does not value these predictors as much. They may possibly be confounding, therefore reducing each others importance relative to the overall model. Overall, the correlation matches fairly strongly to the predictor importance findings with MLR, random forest, and boosting.

Other Models' Findings

We are not going to analyze the polynomial regression as the default multiple linear regression model does well without the added complexity. We do not use GAMs or Splines as the inferencing for those methods are not straight forward, nor are they easy to use. The Lasso and Ridge analyses failed to deliver a better model than the MLR using forward selection. PCR and PLS are unnecessary as the predictors are transformed and therefore lose much of the interpretation of influence and significance. The KNN regression has near 0 inferencing ability, and thus we will not use it to analyze the predictor importance. The default, singular decision tree model is poor, and thus we will not analyze it, either.

Key Takeaways:

We have a few primary takeaways. In the MLR, random forest, boosting models have the same predictors with greater influence on the model. Here are a list of predictors (with simple definitions) that appear in all 3 models as significant:

- pf_expression_control or pf_expression_influence. This metric measures the censorship, control over media, and limitations on freedom of speech/assembly. This finding is intuitive. The more free people are to say what they want to say without persecution, the greater the human freedom tends to be.
- ef_legal_military, ef_legal_gender. How fair and just the judicial system is to military (does the military escape punishments or are they held accountable) and gender (is there legal burdens to a specific gender; typically women). Again, an intuitive finding. It makes sense that the more fair and just a nation's court system is, the greater the human freedom tends to be.
- ef_trade_tariffs_mean and ef_trade_movement_visit. The economic freedom to trade without tariffs (free-trade) and the ability for foreign trade goods to easily move through the country. Basically, these measures estimate the ability to have good, easy, free trade. The more free the trade is, the greater the human freedom tends to be.

Other predictors we find shared between the boosting and random forest models:

- ef_trade_regulatory, ef_trade_regulatory_compliance, ef_regulation_credit. These three predictors are oriented toward measuring human freedom surrounding regulation of various parts of the economy including trade (foreign), compliance to regulations, and the credit sector. Again, the more free from stringent regulations and bureaucracy overhead, the greater the human freedom tends to be.

Some other predictors that are important in the three models (less so in random forest and boosting, but very significant in MLR):

- pf_ss_homicide. This predictor gauges how free people are to avoid being murdered. If a country has high murder rates, the human freedom tends to be lower.

These variables help us to understand the major components of human freedom. Many variables within the dataset do not seem to affect human freedom as much has others. There is also a confounding issue, as many predictors have significant correlation with the human freedom, but also with other variables. These confounding relationships lead some models to ignore certain variables as another variable is capturing largely the 'same' variance.

```
pf_expression_control_lm = lm(hf_score ~ pf_expression_control, data = hfi_data_combined_train)
yhat = predict(pf_expression_control_lm, newdata = hfi_data_combined_test)
pf_expression_control_lm_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)
pf_expression_control_lm_test_r2 = round(1 - pf_expression_control_lm_test_mse / mean_tss_test, 6)

pf_expression_tot_lm = lm(hf_score ~ pf_expression_control + pf_expression_influence , data = hfi_data_
yhat = predict(pf_expression_tot_lm, newdata = hfi_data_combined_test)
pf_expression_tot_lm_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)
pf_expression_tot_lm_test_r2 = round(1 - pf_expression_tot_lm_test_mse / mean_tss_test, 6)

paste("The Test R-Squared for MLR with only pf_expression_control is:", pf_expression_control_lm_test_r
```

```
## [1] "The Test R-Squared for MLR with only pf_expression_control is: 0.549374"
```

```
paste("The Test R-Squared for MLR with pf_expression_control and pf_expression_influence is:", pf_expres
```

```
## [1] "The Test R-Squared for MLR with pf_expression_control and pf_expression_influence is: 0.549399"
```

Another piece of evidence for the confounding relationships is the correlation matrix from section 2 (Exploratory Analysis). The correlation matrix indicates huge correlation of predictors, yet the MLR has more than 30 predictors within it. However, using just pf_expression_control would result in roughly an R-Squared of 0.549374. Meaning a simple linear model with only 1 predictor results in a fairly accurate prediction. However if we add pf_expression_influence (the second most correlated variable to human freedom; first is pf_expression_control), the estimated test R-Squared is 0.549399. As we can see, the improvement to the model is negligible. This indicates that the 2 variables are confounded and explain the same variance within the human freedom metric.

These results will allow us to build a simpler, basic model with the variables shared between the most accurate models to derive an accurate final model with less complexity and predictors. Ideally, this smaller model(s) will allow easy understanding of the main components of human freedom.

6.3 Predictors to Analyze for Final Models

We will use the variables highlighted above (end of section 6.2, with details on what the predictor represents). These predictors and the models that showed they were significant or important are:

1. pf_expression_control (found in MLR, random forest, boosting)
2. pf_expression_influence (found in MLR, random forest, boosting)
3. ef_legal_gender (found in MLR, random forest, boosting)
4. ef_legal_military (found in MLR, random forest, boosting)
5. ef_trade_tariffs_mean (found in MLR, random forest, boosting)
6. ef_trade_movement_visit (found in MLR, random forest, boosting)
7. ef_trade_regulatory (found in random forest and boosting)
8. ef_trade_regulatory_compliance (found in random forest and boosting)
9. ef_regulation_credit (found in random forest and boosting)
10. pf_ss_homicide (found in MLR and random forest)

6.4 Multiple Linear Regression Model, Simplified

First we will perform a basic multiple linear regression model with the 10 predictors from above.

```
cor(hfi_data_combined$ef_trade_regulatory, hfi_data_combined$ef_trade_regulatory_compliance)
```

```
## [1] 0.9452313
```

```
cor(hfi_data_combined$pf_expression_influence, hfi_data_combined$pf_expression_control)
```

```
## [1] 0.9152187
```

The above correlations indicate the pairs of predictors (ef_trade_regulatory and ef_trade_regulatory_compliance; pf_expression_influence and pf_expression_control) are confounders. They are highly correlated, and thus capture the same variance within human variance. We tested this assumption with the MLR model with and without the pairs. We find in our analysis that the MLR needs only 8 of the 10 predictors. We removed the 2 predictors that were least significant: ef_trade_regulatory_compliance and pf_expression_control.

```
names_of_preds = c("pf_expression_control", "pf_expression_influence", "ef_legal_gender",
                   "ef_legal_military", "ef_trade_tariffs_mean", "ef_trade_movement_visit",
                   "ef_trade_regulatory", "ef_trade_regulatory_compliance", "ef_regulation_credit",
                   "pf_ss_homicide")
final_lm_model = lm(paste("hf_score ~", paste(names_of_preds, collapse = " + "), "- ef_trade_regulatory_
yhat = predict(final_lm_model, newdata = hfi_data_combined_test)
final_lm_model_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)
final_lm_model_r2 = round((1 - final_lm_model_mse / mean_tss_test) * 100, 2)

paste("The Mean TSS of the Test Set:", round(mean_tss_test ,6))
```

```
## [1] "The Mean TSS of the Test Set: 0.882886"
```

```
paste("The Test MSE of the Final MLR Model:", round(final_lm_model_mse ,6))
```

```
## [1] "The Test MSE of the Final MLR Model: 0.134352"
```

```
paste("The Test R-Sqared of the Final MLR Model:", round(final_lm_model_r2 ,6))
```

```
## [1] "The Test R-Sqared of the Final MLR Model: 84.78"
```

We get a MLR model with a resulting Test MSE of 0.134352 compared to the mean TSS of 0.882886. The models chosen by forward selection and best-subset perform much better with a Test MSE of ~0.042 - 0.043. The Test R-Squared of the simplified model is 84.78%, meaning a substantial amount of variance is explained using the simplified model. This result indicates that the simplified and biased model (we chose the predictors ourselves), lead to a still highly accurate and predictive model.

```
summary(final_lm_model)
```

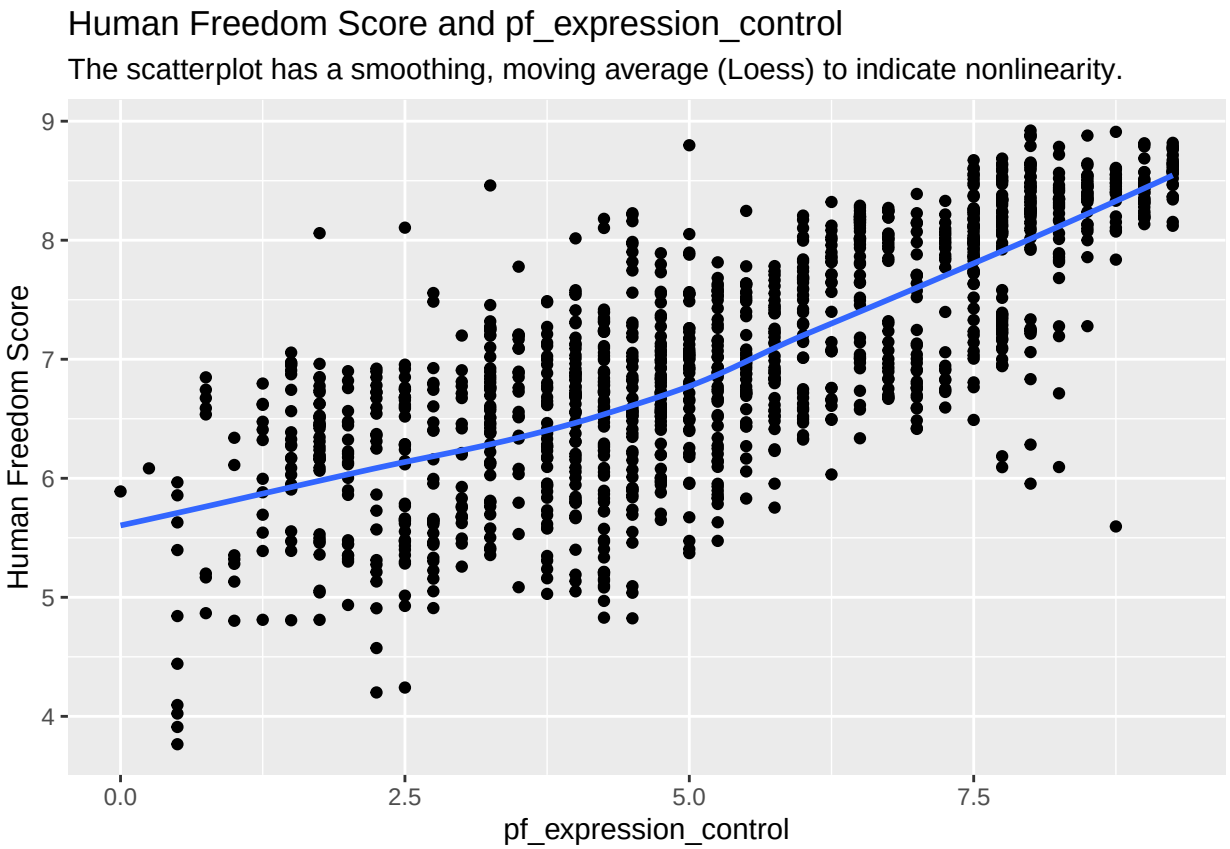
```
##
## Call:
## lm(formula = paste("hf_score ~", paste(names_of_preds, collapse = " + "),
##     "- ef_trade_regulatory_compliance - pf_expression_control"),
##     data = hfi_data_combined_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.27226 -0.20451 -0.00112  0.23439  1.20918
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.281089   0.113197   11.32   <2e-16 ***
```

```
## pf_expression_influence 0.104201   0.005817   17.91   <2e-16 ***
## ef_legal_gender         1.783596   0.096041   18.57   <2e-16 ***
## ef_legal_military       0.069966   0.005527   12.66   <2e-16 ***
## ef_trade_tariffs_mean   0.121069   0.013438    9.01   <2e-16 ***
## ef_trade_movement_visit 0.035683   0.003555   10.04   <2e-16 ***
## ef_trade_regulatory     0.103140   0.008603   11.99   <2e-16 ***
## ef_regulation_credit    0.104088   0.008922   11.67   <2e-16 ***
## pf_ss_homicide          0.061853   0.004118   15.02   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3404 on 1035 degrees of freedom
## Multiple R-squared:  0.885,  Adjusted R-squared:  0.8842
## F-statistic: 996.1 on 8 and 1035 DF,  p-value: < 2.2e-16
```

6.5 Multiple Linear Regression Model with Polynomial Terms, Simplified

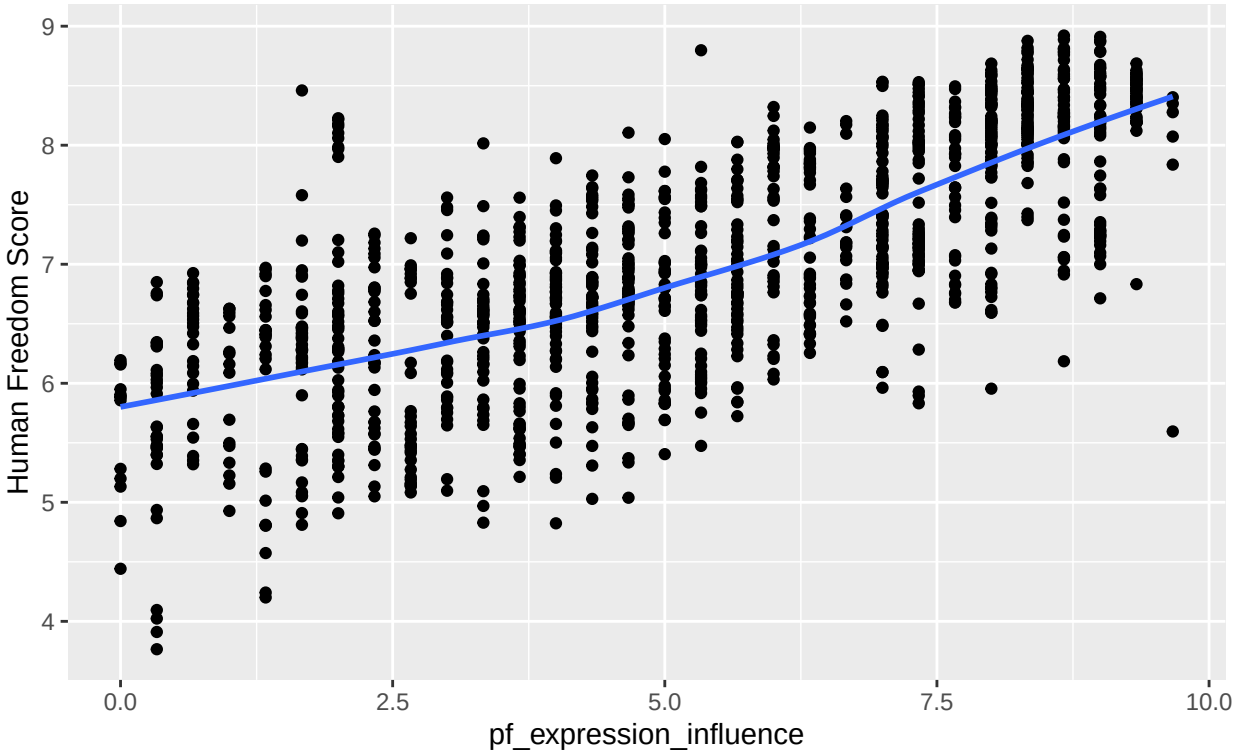
Now we will take the above linear model and create a polynomial regression model. First, we will analyze the pair plots of the predictors with human freedom.

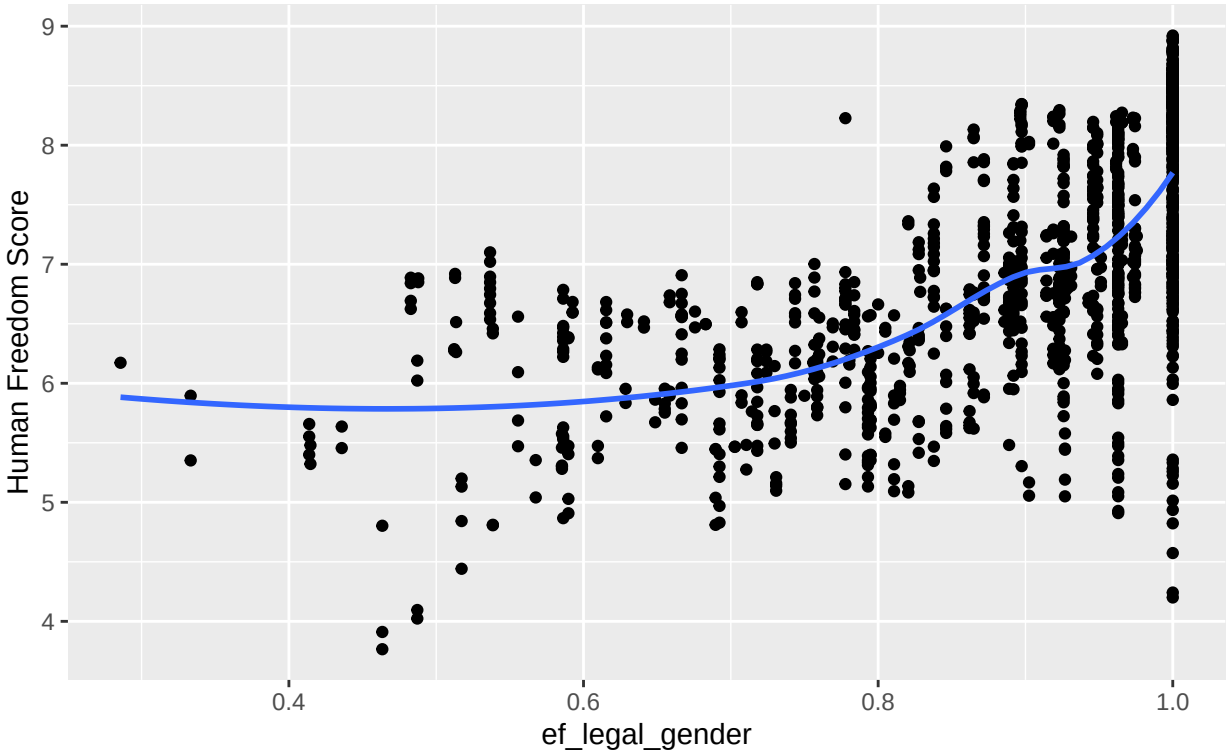
```
par(mfrow=c(1,1))

for ( i in 1:length(names_of_preds)){
  plot = ggplot(data = hfi_data_combined,
                aes(x = hfi_data_combined[[names_of_preds[i]]], y = hf_score)) +
    geom_point() + geom_smooth(method = loess, formula = y ~ x, se=FALSE) +
    xlab(names_of_preds[i]) + ylab("Human Freedom Score") +
    ggtitle(paste(title = "Human Freedom Score and", names_of_preds[i]),
            subtitle = "The scatterplot has a smoothing, moving average (Loess) to indicate nonlinearity

  print(plot)
}
```

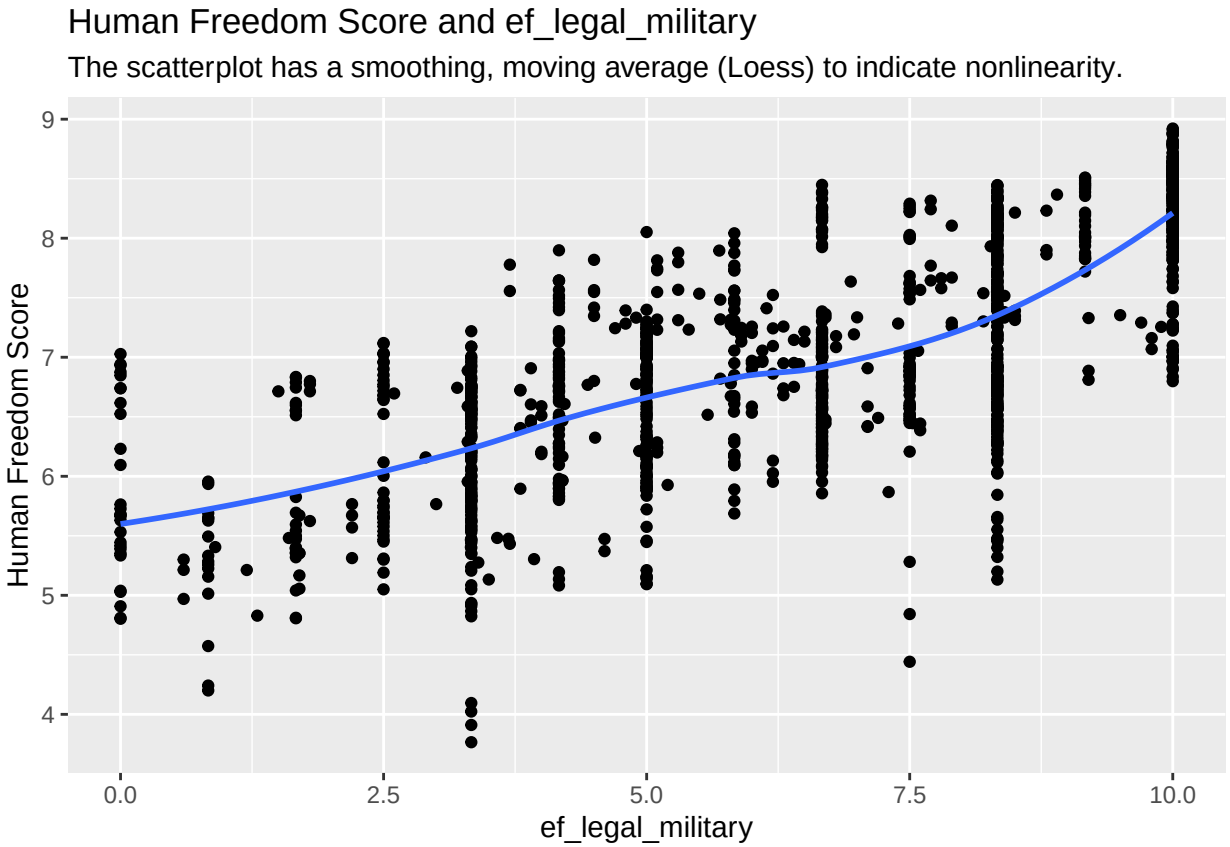

Human Freedom Score and pf_expression_control
The scatterplot has a smoothing, moving average (Loess) to indicate nonlinearity.
Human Freedom Score
9
8
7
6
5
4
0.0
2.5
5.0
7.5
pf_expression_control

Human Freedom Score and pf_expression_influence

The scatterplot has a smoothing, moving average (Loess) to indicate nonlinearity.

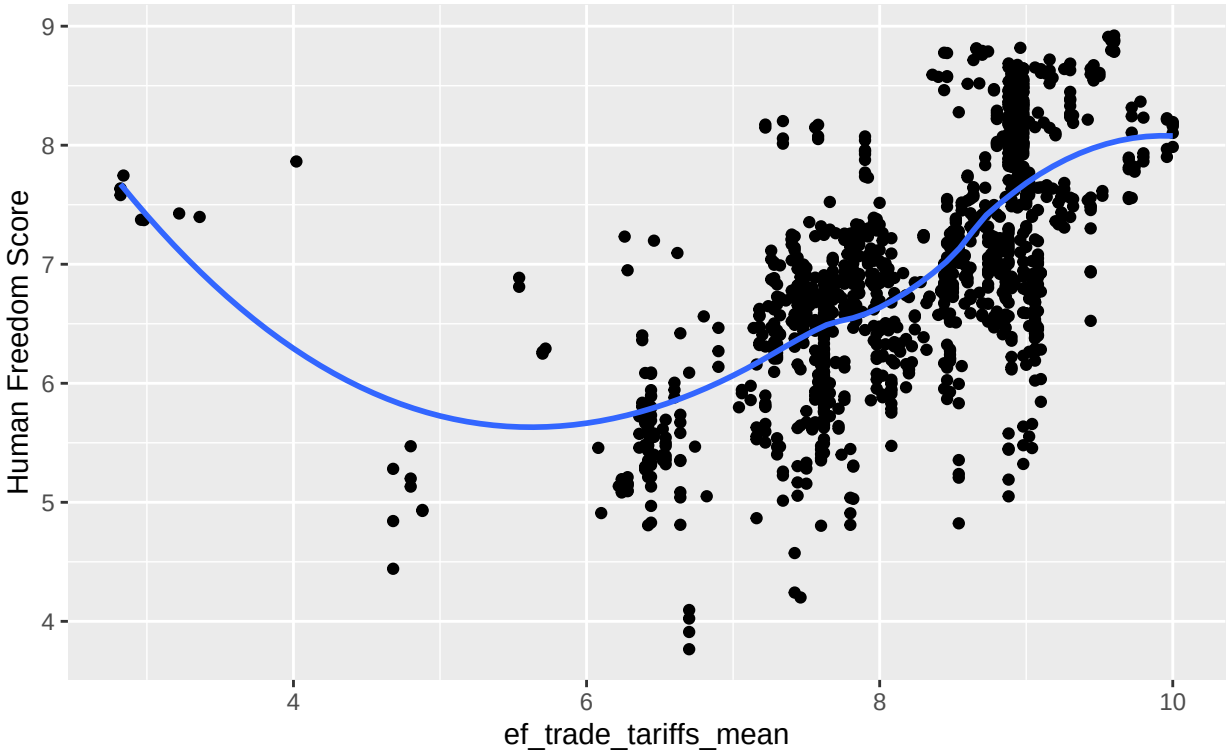


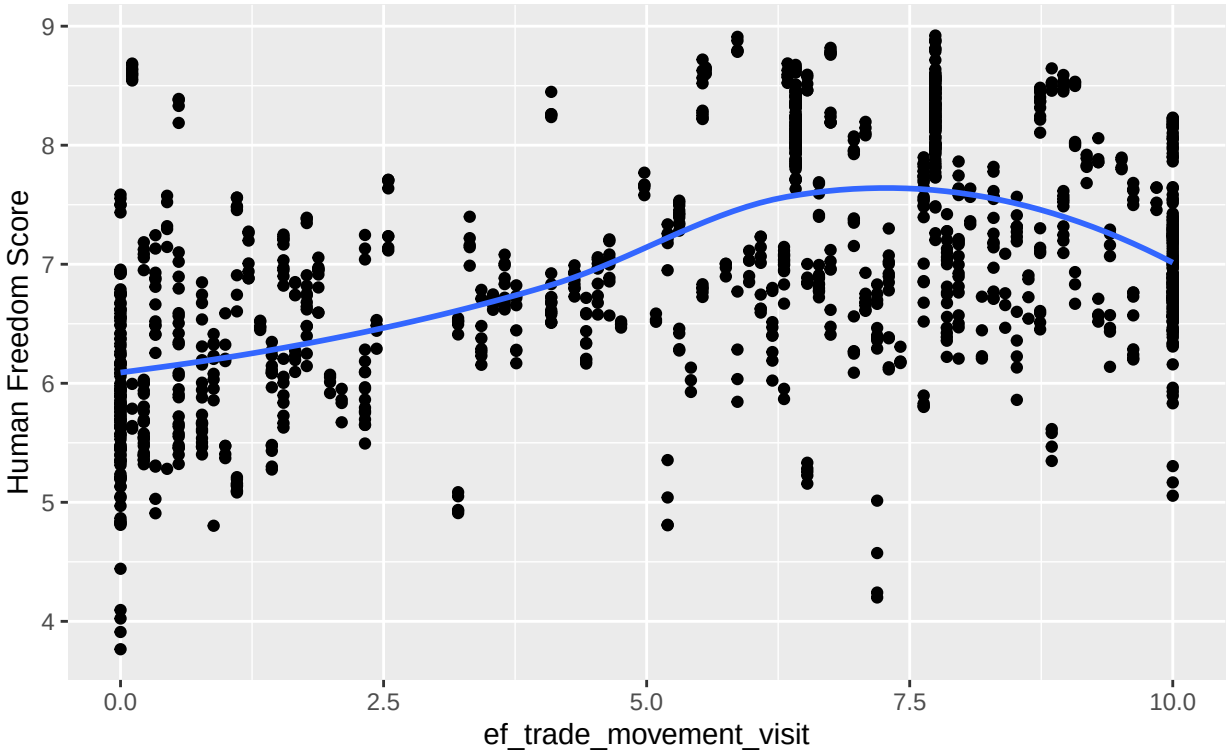

Human Freedom Score and ef_legal_gender
The scatterplot has a smoothing, moving average (Loess) to indicate nonlinearity.
Human Freedom Score
9
8
7
6
5
4
0.4
0.6
0.8
1.0
ef_legal_gender

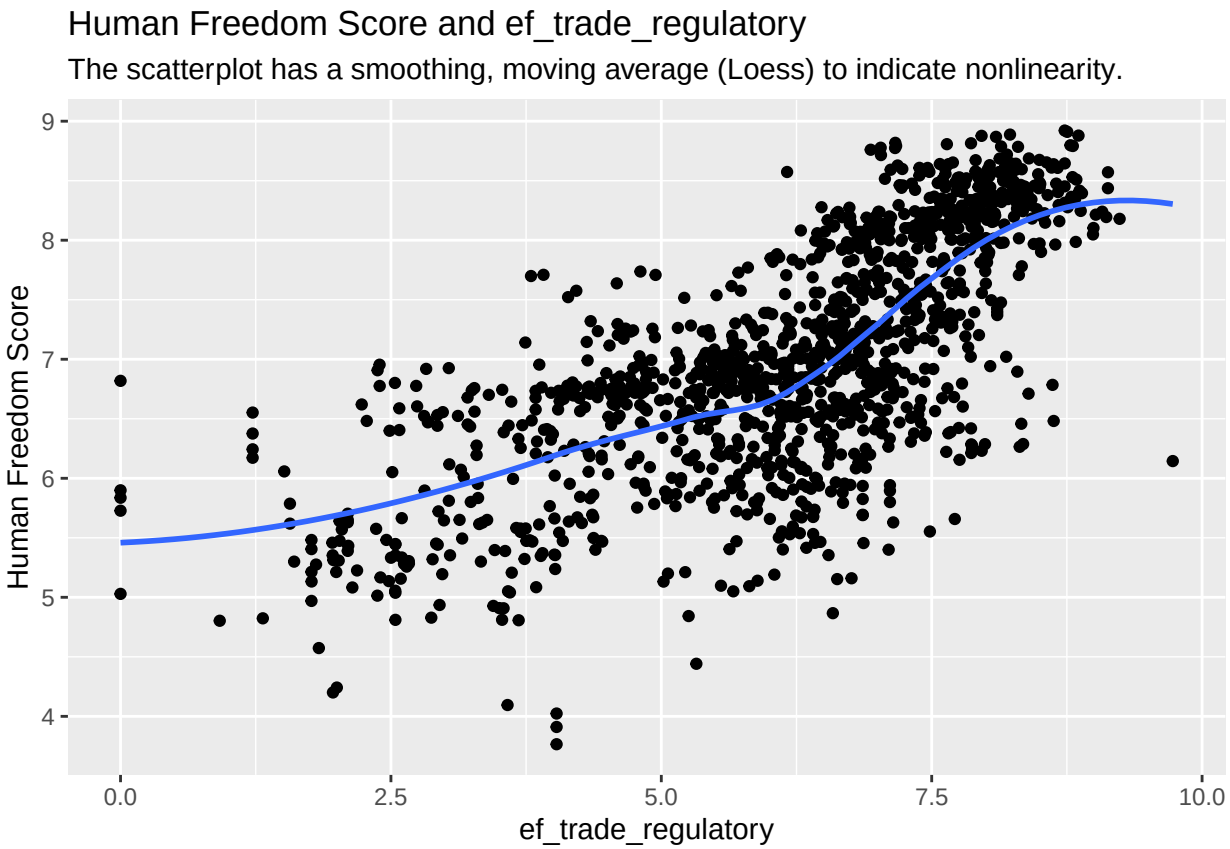

Human Freedom Score and ef_legal_military
The scatterplot has a smoothing, moving average (Loess) to indicate nonlinearity.
Human Freedom Score
9
8
7
6
5
4
0.0
2.5
5.0
7.5
10.0
ef_legal_military

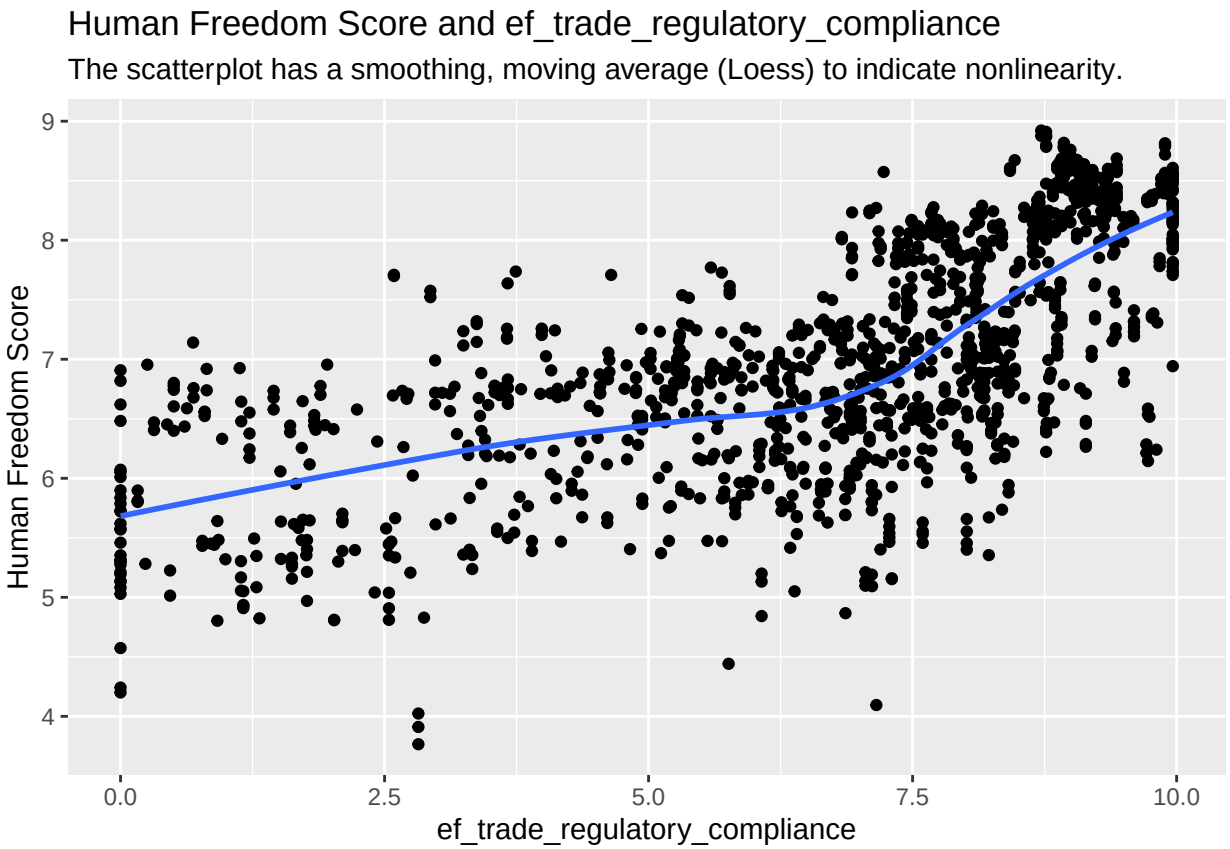
Human Freedom Score and ef_trade_tariffs_mean

The scatterplot has a smoothing, moving average (Loess) to indicate nonlinearity.



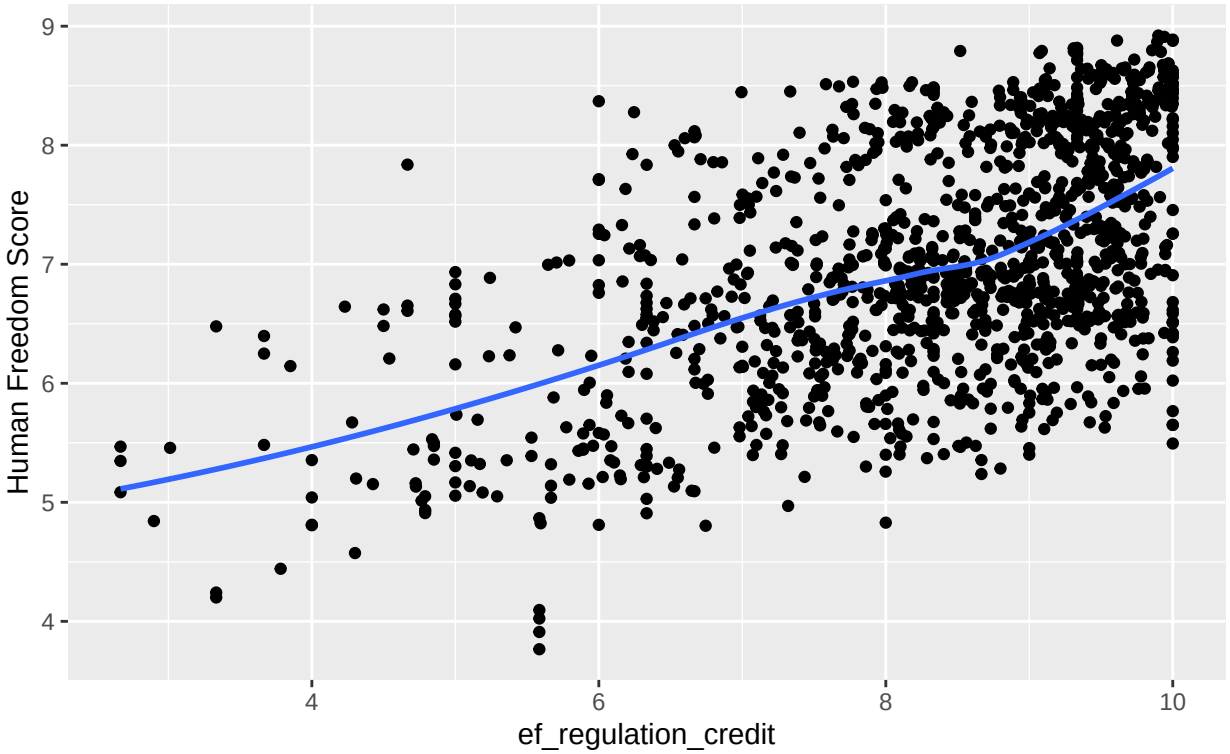

Human Freedom Score and ef_trade_movement_visit
The scatterplot has a smoothing, moving average (Loess) to indicate nonlinearity.
Human Freedom Score
9
8
7
6
5
4
0.0
2.5
5.0
7.5
10.0
ef_trade_movement_visit

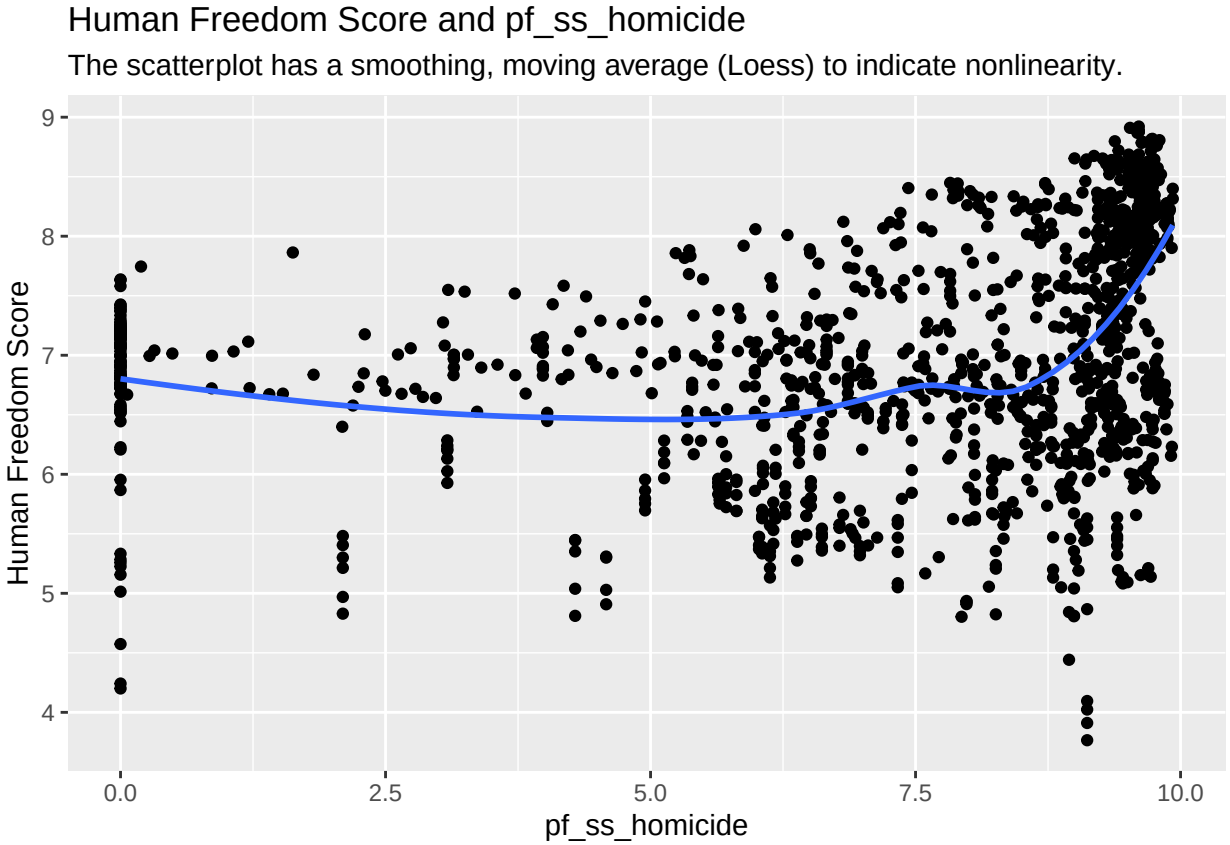

Human Freedom Score and ef_trade_regulatory
The scatterplot has a smoothing, moving average (Loess) to indicate nonlinearity.
Human Freedom Score
9
8
7
6
5
4
0.0
2.5
5.0
7.5
10.0
ef_trade_regulatory


Human Freedom Score and ef_trade_regulatory_compliance
The scatterplot has a smoothing, moving average (Loess) to indicate nonlinearity.
Human Freedom Score
9
8
7
6
5
4
0.0
2.5
5.0
7.5
10.0
ef_trade_regulatory_compliance

Human Freedom Score and ef_regulation_credit

The scatterplot has a smoothing, moving average (Loess) to indicate nonlinearity.





These various scatterplots highlight that nearly all predictors have a somewhat linear relationship (even the predictors found in boosting and random forest, like ef_trade_regulatory, ef_trade_regulatory_compliance, ef_regulation_credit). The predictors we will try to see if polynomial regression can improve our simplified MLR model are:

- ef_trade_regulatory
- ef_trade_tariffs_mean
- ef_regulation_credit
- ef_legal_gender
- pf_ss_homicide

```
# Commentted out for sake of speed. This part takes a long time ~5+ mins. The results are not very inte

# set.seed(111)
#
# poly_vars = c("ef_trade_regulatory", "ef_trade_tariffs_mean", "ef_regulation_credit", "ef_legal_gende
# max_poly_term = 5
# coef_matrix = matrix(data = rep(c(0, 0, 0, 0, 0), 5^5), ncol = 5)
# final_poly_cv_mse = rep(100, 5^5)
# i = 1
# for( a in 1:max_poly_term){
#   for( b in 1:max_poly_term){
#     for( c in 1:max_poly_term){
#       for( d in 1:max_poly_term){
#         for( e in 1:max_poly_term){
```

```
#           fit = glm(as.formula(paste("hf_score ~", paste(names_of_preds, collapse = " + "),
#                                      "+ poly(ef_trade_regulatory,", a, ") + poly(ef_trade_tariffs_mean,",
#                                      b, ") + poly(ef_regulation_credit,", c, ") + poly(ef_legal_gender,", d
#                                      , ") + poly(pf_ss_homicide,", e, ") -", paste(poly_vars, collapse = " - "))),
#                     data = hfi_data_combined_train)
#           coef_matrix[i, ] = c(a, b, c, d, e)
#           final_poly_cv_mse[i] = cv.glm(hfi_data_combined_train, fit, K = 5)$delta[1]
#           i = i+1
#         }
#       }
#     }
#   }
# }
#
# paste("Minimum Coefs:")
# paste(coef_matrix[which.min(final_poly_cv_mse), ])
# paste("Minimum CV MSE:", round(min(final_poly_cv_mse), 6))
# paste("Minimum CV MSE of Default MLR:", round(final_poly_cv_mse[1], 6))
# paste("Mean TSS of Train Set:", round(mean_tss_train, 6))
```

After running, the above regression, the benefits from the polynomial terms were minimal. The best cross-validated MSE we returned was 0.106 compared to the default model's 0.11164 cross-validated MSE with **no** polynomial terms. Therefore, we will not add any terms to the MLR model. The improvement is too insignificant, especially as the additional terms were 1 quartic (4th power), 1 cubic, 2 quadratic, and 6 linear terms. That means there are 7 additional degrees of freedom for the model. This model adds complexity, yet fails to deliver more than a 2% increase in R-Squared value.

6.6 Random Forest Model, Simplified

Run random forest analysis again; with subset of predictors identified at the beginning of section 6. The MLR and MLR with

```
# RANDOM FOREST AND BAGGING
set.seed(111)
default_ntrees = 500
simple_random_forests = data.frame(MTry =  seq(from = 1, to = 10, by = 1), Test_MSE = rep(100, 10))

forest_i = 1
for (current_mtry in simple_random_forests$MTry) {
  rf_simplified = randomForest(as.formula(paste("hf_score ~", paste(names_of_preds, collapse = " + "))),
                               data = hfi_data_combined_train,
                               ntree = default_ntrees, mtry = current_mtry)

  yhat = predict(rf_simplified, newdata = hfi_data_combined_test)

  simple_random_forests$MTry[forest_i] = current_mtry
  simple_random_forests$Test_MSE[forest_i] = mean((yhat - hfi_data_combined_test$hf_score)^2)
  forest_i = forest_i + 1
}

paste("The Mean TSS for the Test Set:", round(mean_tss_test, 6))
```

```
## [1] "The Mean TSS for the Test Set: 0.882886"
```

```r
paste("The Test MSE for the Final Simplified MLR Model:", round(final_lm_model_mse, 6))
```

```
## [1] "The Test MSE for the Final Simplified MLR Model: 0.134352"
```

```r
simple_random_forests %>% arrange(Test_MSE)
```

```
##    MTry   Test_MSE
## 1     4 0.04770324
## 2     3 0.04821449
## 3     5 0.04854603
## 4     2 0.04899809
## 5     6 0.04935124
## 6     8 0.05047776
## 7     7 0.05068233
## 8     9 0.05081980
## 9     1 0.05115384
## 10   10 0.05278814
```

```r
temp = simple_random_forests %>% arrange(Test_MSE)
best_mtry_10pred = temp$MTry[1]
best_mse_10pred = round(temp$Test_MSE[1], 5)
```

The 10-predictor subset random forest model is accurate. The Test MSE is consistently around 0.05 for all MTry (Number of variables to consider at each split in tree generation). The best model is MTry = 4 with a Test MSE of 0.0477. This result is significantly improved over the final MLR model Test MSE of 0.134352.

We will now run the same process but with the 2 confounding predictors from MLR removed (ef_trade_regulatory_compliance and pf_expression_influence).

```r
# RANDOM FOREST AND BAGGING
set.seed(111)

simple_random_forests = data.frame(MTry =  seq(from = 1, to = 8, by = 1), Test_MSE = rep(100, 8))
forest_i = 1

for (current_mtry in simple_random_forests$MTry) {
  rf_simplified = randomForest(as.formula(paste("hf_score ~",
                                                paste(names_of_preds, collapse = " + "),
                                                "- ef_trade_regulatory_compliance - pf_expression_influ
                               data = hfi_data_combined_train,
                               ntree = default_ntrees, mtry = current_mtry)

  yhat = predict(rf_simplified, newdata = hfi_data_combined_test)
  simple_random_forests$Test_MSE[forest_i] = mean((yhat - hfi_data_combined_test$hf_score)^2)
  forest_i = forest_i + 1
}

paste("The Mean TSS for the Test Set:", round(mean_tss_test, 6))
```

```
## [1] "The Mean TSS for the Test Set: 0.882886"
```

```r
paste("The Test MSE for the Final Simplified MLR Model:", round(final_lm_model_mse, 6))
```

```
## [1] "The Test MSE for the Final Simplified MLR Model: 0.134352"
```

```r
simple_random_forests %>% arrange(Test_MSE)
```

```
##   MTry   Test_MSE
```

```
## 1    2 0.04647152
## 2    3 0.04663756
## 3    4 0.04735718
## 4    5 0.04900835
## 5    1 0.04970858
## 6    6 0.05053962
## 7    7 0.05275419
## 8    8 0.05340238
```

```
temp = simple_random_forests %>% arrange(Test_MSE)
best_mtry_8pred = temp$MTry[1]
best_mse_8pred = round(temp$Test_MSE[1], 5)
```

The 8-predictor subset random forest model is equally as accurate. The lowest Test MSE is 0.04647 with MTry set at 2. These random forest models are very accurate and the 8-predictor subset results in a model that has an R-Squared of roughly 94%. Random forest models seem to capture most of the variance within the HFI with minimal worries about overfit. Most likely this accuracy indicates the HFI's variance **does** have underlying connections as a model with increasing degrees of freedom can continually deliver high accuracy models without much overfit. As stated before, these simpler models are useful to estimate while having more insight into what affects overall human freedom.

The final model we are using is the 8-predictor subset with MTRY of 2. The results of the model and importance analysis of predictors:

```
default_ntrees = 500
rf_final = randomForest(as.formula(paste("hf_score ~",
                                         paste(names_of_preds, collapse = " + "),
                                         "- ef_trade_regulatory_compliance - pf_expression_influen
                        data = hfi_data_combined_train, ntree = default_ntrees,
                        mtry = best_mtry_8pred, importance = T)

yhat = predict(rf_final, newdata = hfi_data_combined_test)
rf_final_mse = round(mean((yhat - hfi_data_combined_test$hf_score)^2), 6)
rf_final$importance
```

```
##                            %IncMSE IncNodePurity
## pf_expression_control   0.24791986     218.44974
## ef_legal_gender         0.21253098     136.34933
## ef_legal_military       0.17727805     146.84084
## ef_trade_tariffs_mean   0.20443612     169.85846
## ef_trade_movement_visit 0.12115843      82.87914
## ef_trade_regulatory     0.18540051     168.35430
## ef_regulation_credit    0.07256042      55.98973
## pf_ss_homicide          0.09342418      55.92985
```

```
rf_final_mse
```

```
## [1] 0.04494
```

```
print( paste("The Test MSE for 8-Predictor Subset Random Forest: ", round(rf_final_mse, 4), sep = "" ))
```

```
## [1] "The Test MSE for 8-Predictor Subset Random Forest: 0.0449"
```

```
print( paste("The Estimated R-Squared for 8-Predictor Subset Random Forest: ",
             round((1 - (rf_final_mse / mean_tss_test)) * 100, 2), "%", sep = "" ))
```

```
## [1] "The Estimated R-Squared for 8-Predictor Subset Random Forest: 94.91%"
```

```
print( paste("The Test MSE for 8-Predictor MLR Model: ", round(final_lm_model_mse, 4), sep = "" ))
```

```
## [1] "The Test MSE for 8-Predictor MLR Model: 0.1344"
```

```
print( paste("The Estimated R-Squared for 8-Predictor MLR Model: ",
             round((1 - (final_lm_model_mse / mean_tss_test)) * 100, 2), "%", sep = "" ))
```

```
## [1] "The Estimated R-Squared for 8-Predictor MLR Model: 84.78%"
```

Overall, the random forests had exceptional predictive accuracy with the subset of predictors. The R-Squared is roughly 95%. That is an improvement compared to the simplified MLR (R-Squared = 85%). The importance of the random forest indicates nearly all predictors have a degree of relevance (especially compared to the random forest with all 39 predictors).

We strongly believe the 8-predictor subset random forest model is better than the complete 39-predictor model. The 8-predictor model simplifies the complexity of explaining the components of human freedom by removing over 75% of the predictors. The resulting model is nearly as accurate and is easier to see the predictors that are strongly connected to human freedom. The importance highlights that each of the 8 predictors are critical to lowering the MSE.

Additionally, the final random forest model (8-predictor subset) is superior to the final MLR model created earlier. The final MLR model captures roughly 85% of the variance of human freedom compared to the random forest model's 95%. However, the MLR is more useful for determining the influence each predictor has on the estimated human freedom; so both models are useful.

6.7 Boosting Model, Simplified

```
# BOOSTING
set.seed(111)
par(mfrow=c(1,1))

lambda = 0.01  # the recommended value. we could test more thoroughly, but the amount of time would be
default_ntrees = 3000
max_inter_depth = 10  # this is more reasonable, typically, one does not need many leaf nodes per tree

boosting_simplified_test_mse = rep(NA, max_inter_depth)

for( inter_depth in 1:max_inter_depth){
  boosting_gbm = gbm(as.formula(paste("hf_score ~", paste(names_of_preds, collapse = " + " ))),
                     data = hfi_data_combined_train, distribution="gaussian",
                     interaction.depth = inter_depth, n.trees=default_ntrees,
                     shrinkage = lambda, cv.folds = 10)

  yhat = predict(boosting_gbm, newdata = hfi_data_combined_test, n.trees = default_ntrees)
  boosting_simplified_test_mse[inter_depth] = mean((yhat - hfi_data_combined_test$hf_score)^2)

  print(paste("At Maximum Inter-Depth: ", inter_depth))
  print(paste("10-Fold Cross-Validation Error Estimate is",
              round(boosting_gbm$cv.error[default_ntrees], 6)))
  print(paste("Test MSE:", round(boosting_simplified_test_mse[inter_depth], 6)))
  print(paste("Mean TSS of Test Set:", round(mean_tss_test, 6)))
}
```

```
## [1] "At Maximum Inter-Depth:  1"
## [1] "10-Fold Cross-Validation Error Estimate is 0.093557"
```

```
## [1] "Test MSE: 0.10245"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  2"
## [1] "10-Fold Cross-Validation Error Estimate is 0.067987"
## [1] "Test MSE: 0.070459"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  3"
## [1] "10-Fold Cross-Validation Error Estimate is 0.056298"
## [1] "Test MSE: 0.057646"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  4"
## [1] "10-Fold Cross-Validation Error Estimate is 0.049961"
## [1] "Test MSE: 0.055207"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  5"
## [1] "10-Fold Cross-Validation Error Estimate is 0.046808"
## [1] "Test MSE: 0.051231"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  6"
## [1] "10-Fold Cross-Validation Error Estimate is 0.045773"
## [1] "Test MSE: 0.04958"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  7"
## [1] "10-Fold Cross-Validation Error Estimate is 0.042923"
## [1] "Test MSE: 0.047684"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  8"
## [1] "10-Fold Cross-Validation Error Estimate is 0.040045"
## [1] "Test MSE: 0.046754"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  9"
## [1] "10-Fold Cross-Validation Error Estimate is 0.039292"
## [1] "Test MSE: 0.046229"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  10"
## [1] "10-Fold Cross-Validation Error Estimate is 0.039185"
## [1] "Test MSE: 0.046509"
## [1] "Mean TSS of Test Set: 0.882886"
```

```
best_boosting_inter_depth = 8  # chosen manually looking at results.
final_boosting_test_mse = boosting_simplified_test_mse[8]
```

This boosting model is very accurate with the smaller 10-predictor subset. With more complex individual trees (dictated by the inter_depth hyper-parameter), indicate the model does not suffer much risk of overfit. MEaning, the variance within human freedom can be captured by these 10-predictors quite well. The smallest Test MSE is 0.047 which is comparable to the final 8-predictor subset random forest model above.

We will run the boosting models with an 8-predictor subset removing the potentially confounding predictors ef_trade_regulatory_compliance and pf_expression_influence.

```
# BOOSTING
set.seed(111)

lambda = 0.01  # the recommended value. we could test more thoroughly, but the amount of time would be
default_ntrees = 3000
max_inter_depth = 8  # this is more reasonable, typically, one does not need many leaf nodes per tree
```

```
boosting_simplified_test_mse = rep(NA, max_inter_depth)

for( inter_depth in 1:max_inter_depth){
  boosting_gbm = gbm(as.formula(paste("hf_score ~", paste(names_of_preds, collapse = " + " ),
                                      "- ef_trade_regulatory_compliance - pf_expression_influence" )),
                     data = hfi_data_combined_train, distribution="gaussian",
                     interaction.depth = inter_depth, n.trees=default_ntrees,
                     shrinkage = lambda, cv.folds = 10)

  yhat = predict(boosting_gbm, newdata = hfi_data_combined_test, n.trees = default_ntrees)
  boosting_simplified_test_mse[inter_depth] = mean((yhat - hfi_data_combined_test$hf_score)^2)

  print(paste("At Maximum Inter-Depth: ", inter_depth))
  print(paste("10-Fold Cross-Validation Error Estimate is",
              round(boosting_gbm$cv.error[default_ntrees], 6)))
  print(paste("Test MSE:", round(boosting_simplified_test_mse[inter_depth], 6)))
  print(paste("Mean TSS of Test Set:", round(mean_tss_test, 6)))
}
```

```
## [1] "At Maximum Inter-Depth:  1"
## [1] "10-Fold Cross-Validation Error Estimate is 0.098747"
## [1] "Test MSE: 0.10434"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  2"
## [1] "10-Fold Cross-Validation Error Estimate is 0.070641"
## [1] "Test MSE: 0.073336"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  3"
## [1] "10-Fold Cross-Validation Error Estimate is 0.058842"
## [1] "Test MSE: 0.061472"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  4"
## [1] "10-Fold Cross-Validation Error Estimate is 0.052378"
## [1] "Test MSE: 0.059604"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  5"
## [1] "10-Fold Cross-Validation Error Estimate is 0.048486"
## [1] "Test MSE: 0.056403"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  6"
## [1] "10-Fold Cross-Validation Error Estimate is 0.046524"
## [1] "Test MSE: 0.052344"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  7"
## [1] "10-Fold Cross-Validation Error Estimate is 0.04461"
## [1] "Test MSE: 0.052067"
## [1] "Mean TSS of Test Set: 0.882886"
## [1] "At Maximum Inter-Depth:  8"
## [1] "10-Fold Cross-Validation Error Estimate is 0.041987"
## [1] "Test MSE: 0.052326"
## [1] "Mean TSS of Test Set: 0.882886"
```

The 8-predictor subset (removing 2 predictors for being confounding in the MLR; ef_trade_regulatory_compliance and pf_expression_influence) results in a worse model, but only slightly. With only somewhat worse Test

MSE results 0.052 vs 0.047, we will choose the less complex (by number of predictors perspective). The 8-predictor subset boosting model performs best with high inter-depth level; meaning that the individual trees aggregating within the boosting model capture a lot of variance without much overfit concerns. Again, this finding has been found in all other models previously. The fact that models can have high degrees of freedom and flexibility to capture any potential trend, yet not overfit, means the predictors are most likely very strongly (near 100%) correlated with human freedom.

Here is our final 8-predictor subset boosting model with inter-depth 6 (after 6, the model does not seem to benefit much from the additional model variance).

```
default_ntrees = 3000
lambda = 0.01

final_boosting_gbm = gbm(as.formula(paste("hf_score ~", paste(names_of_preds, collapse = " + " ),
                                    "- ef_trade_regulatory_compliance - pf_expression_influence" )),
                         data = hfi_data_combined_train, distribution="gaussian",
                         interaction.depth = 6, n.trees=default_ntrees,
                         shrinkage = lambda, cv.folds = 10)

summary(final_boosting_gbm, plot=F)
```

```
##                                           var   rel.inf
## pf_expression_control     pf_expression_control 31.560413
## ef_trade_regulatory         ef_trade_regulatory 16.709604
## ef_trade_tariffs_mean     ef_trade_tariffs_mean 16.658093
## ef_legal_gender                ef_legal_gender 10.862381
## ef_legal_military            ef_legal_military  6.742109
## ef_regulation_credit       ef_regulation_credit  6.184339
## ef_trade_movement_visit ef_trade_movement_visit  6.122543
## pf_ss_homicide                   pf_ss_homicide  5.160518
```

```
yhat = predict(boosting_gbm, newdata = hfi_data_combined_test, n.trees = default_ntrees)
final_boosting_test_mse = mean((yhat - hfi_data_combined_test$hf_score)^2)
final_boosting_test_r2 = round((1 - (final_boosting_test_mse / mean_tss_test)) * 100, 2)

print( paste("The Test MSE for 8-Predictor Subset Boosting Model: ", round(final_boosting_test_mse, 4),
```

```
## [1] "The Test MSE for 8-Predictor Subset Boosting Model: 0.0523"
```

```
print( paste("The Estimated R-Squared for 8-Predictor Subset Boosting Model: ",
             round((1 - (final_boosting_test_mse / mean_tss_test)) * 100, 2), "%", sep = "" ))
```

```
## [1] "The Estimated R-Squared for 8-Predictor Subset Boosting Model: 94.07%"
```

```
print( paste("The Test MSE for 8-Predictor Subset Random Forest: ", round(rf_final_mse, 4), sep = "" ))
```

```
## [1] "The Test MSE for 8-Predictor Subset Random Forest: 0.0449"
```

```
print( paste("The Estimated R-Squared for 8-Predictor Subset Random Forest: ",
             round((1 - (rf_final_mse / mean_tss_test)) * 100, 2), "%", sep = "" ))
```

```
## [1] "The Estimated R-Squared for 8-Predictor Subset Random Forest: 94.91%"
```

```
print( paste("The Test MSE for 8-Predictor MLR Model: ", round(final_lm_model_mse, 4), sep = "" ))
```

```
## [1] "The Test MSE for 8-Predictor MLR Model: 0.1344"
```

```
print( paste("The Estimated R-Squared for 8-Predictor MLR Model: ",
             round((1 - (final_lm_model_mse / mean_tss_test)) * 100, 2), "%", sep = "" ))
```

```
## [1] "The Estimated R-Squared for 8-Predictor MLR Model: 84.78%"
```

Our final boosting model is comparable to the final random forest model (also 8-predictor subset); while both tree-based models outperform the final MLR model considerably (R-Squared is different by roughly 10%; 85% vs 95% where 100% is perfect). However, the boosting model is not better than the random forest model as it is more complex and has a slightly worse error. We prefer the random forest model for use with predictions and MLR for inferencing on the predictors to gage the effect predictors have on human freedom.

6.8 Comparision of Model Predictive Accuracy, Complexity, and Predictor Inferencing

```
final_results_matrix = matrix(
  data=c(round(final_lm_model_mse, 4), round(rf_final_mse, 4),
         round(final_boosting_test_mse, 4),
         round((1 - final_lm_model_mse / mean_tss_test) * 100, 2),
         round((1 - rf_final_mse / mean_tss_test) * 100, 2),
         round((1 - final_boosting_test_mse / mean_tss_test) * 100, 2)), nrow=3, ncol=2)
colnames(final_results_matrix) = c("Test MSE", "Estimate R-Squared (%)")
rownames(final_results_matrix) = c("Multiple Linear Regression with 8 Predictors",
                                   "Random Forest with 8 Predictors and Mtry of 2",
                                   "Boosting with 8 Predictors and Inter-Depth of 6")

print(as.data.frame(final_results_matrix))
```

```
##                                                Test MSE Estimate R-Squared (%)
## Multiple Linear Regression with 8 Predictors      0.1344                  84.78
## Random Forest with 8 Predictors and Mtry of 2     0.0449                  94.91
## Boosting with 8 Predictors and Inter-Depth of 6   0.0523                  94.07
```

These final results show that biasing the dataset significantly (using only 20-25% of the total predictors available) still yields powerful, accurate models. The MLR model is clearly inferior to the ensemble methods in accuracy. However, the MLR does allow for inferencing, which shows all 8 predictors are significantly influencing the estimate for human freedom.

The random forest and boosting models both are the models for accuracy. We actually prefer these boosting and random forest models compared to the 39-predictor versions as the subset of predictors removes data needed to estimate. However, it is important to note that fewer predictors does not mean simpler, per-se. The random forest and boosting models are complex still as they build many trees (500 and 3000, respectively, for these specific models though could be more tailored to potentially improve accuracy), which drives the non-linearity of the models and the high amount of variance captured by these models. Yet, the amount of data needed to run these random forest and boosting models is less stringent (fewer predictors == fewer observations needed).

The more complex the boosting and random forest model became, the better the accuracy (especially boosting and inter-depth hyper-parameter). This finding shows that the variance within human freedom can be captured with the predictors with little risk of overfitting. Normally, these more complex and flexible models can capture increasingly non-linear and subtle trends, but with this increased sensitivity to patterns, the models often find patterns where they do not exist. But, the HFI does not seem to have this issue. The patterns found by the flexible boosting model are actually patterns within the entire dataset. Most likely because the dataset is an aggregate of the variables, so human freedom is directly (or largely) calculated by the predictors we are utilizing to estimate human freedom.

In short: The MLR model for analyzing predictor's affects on human freedom and random forest model for accurately estimating what humna freedom is with the 8-predictor subset.

The 8-predictor subset is:

pf_expression_control, ef_legal_gender, ef_legal_military, ef_trade_tariffs_mean, ef_trade_movement_visit, ef_trade_regulatory, ef_regulation_credit, pf_ss_homicide,

7. Our Final Thoughts on Complete Analysis

Overall, the dataset has variables that are all correlated and inter-connected. The strong relationships between the variables result in very accurate models, even the simple ones like MLR that tend to have weaker predictive ability. Hence, there is not much to takeaway from the prediction side of the various models, but to focus on what predictors are the strongest markers for greater human freedom (or lesser).

We find that MLR, random forest, and boosting models all tend to indicate that these predictors tend to be important / significant / relevant. We list some of them out under section 6.3. Here is the list again:

1. pf_expression_control (found in MLR, random forest, boosting)
2. pf_expression_influence (found in MLR, random forest, boosting)
3. ef_legal_gender (found in MLR, random forest, boosting)
4. ef_legal_military (found in MLR, random forest, boosting)
5. ef_trade_tariffs_mean (found in MLR, random forest, boosting)
6. ef_trade_movement_visit (found in MLR, random forest, boosting)
7. ef_trade_regulatory (found in random forest and boosting)
8. ef_trade_regulatory_compliance (found in random forest and boosting)
9. ef_regulation_credit (found in random forest and boosting)
10. pf_ss_homicide (found in MLR and random forest)

Some notes of confounders within the MLR:

We found 2 pairs of confounding predictors Confounding predictors indicate that the predictors are capturing the same variance and are correlated themselves.

1. pf_expression_control and pf_expression_influence
2. ef_trade_regulatory_compliance and ef_trade_regulatory

Which Model is Best:

We recommend using the MLR if all 31 predictors required for the MLR model given by best-subset or forward selection are present with valid observations. Otherwise, the simpler, 8-predictor random forest or boosting models are preferable. IF computation time is a concern, random forest is better as it can be generated quicker than the boosting. Boosting is slightly less accurate and is more resource intensive to generate the model.

Final Conclusions and Takeaways

Overall, this dataset clearly has inter-connected variables. Most likely the human freedom score is an aggregate of the other 39+ predictors (the additional predictors we ignored due to numerous N/As). Thus, nearly all methods should deliver good results as the dataset is formulaically related among the variables. Regardless, the ensemble methods' accuracy (random forest and boosting) was impressive. These models can often predict human values very well (like rating human freedom metrics). With such high accuracy of the random forest and boosting models, we believe there is a possibility that the dataset variables are not completely objectively selected.

In the beginning of the CATO institute manual for the Human Freedom Index, they bash and attack the lack of freedom within China. This biased stance from the intro of a paper, combined with calculating human freedom as a measure based on **western ideals** of freedom and **libertarian ideals** distorts human freedom. They are not comparing their human freedom metric to surveys of individuals' own perception of freedom within their home country. Obviously, the survey in Russia or China would have a high likelihood of people

lying to ensure they are not hurt, which helps to further CATO's own opinions on human freedom. On the other hand, they may find citizens in Norway feel more liberated and free than Americans, despite their measurement.

OVerall, the Human Freedom Index is an attempt to measure and quantify human freedom by the CATO Institute. Our findings and models indicate that the index is overly complex and could be simplified by focusing on 10 main indicators for human freedom (listed above under section 7 Conclusion). The dataset is also using the western and libertarian views on freedom; often focusing on individual freedom versus collective well-being like some of the eastern world focuses on (like South Korea, Japan).