Stat 1361 Final Project

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

April 8, 2019

1. Data Cleaning and Setup:

1.1 Removing Heavily Correlated Features

Many variables (like pf_rol or pf_ss) are aggregates of more specific Rule-of-Law (rol) and security and safety (ss) features. They seem to do a simple average, which causes the more specific variables to have a lower impact on hf_score. The following code removes them:

```
## Creating a county by year row names
  country= human.freedom.index$countries
  year= human.freedom.index$year
  co_year= paste(country, year, sep = " ", collapse = NULL)
  human.freedom.index= data.frame(co_year, human.freedom.index)
  human.freedom.index$ISO_code <- NULL
  human.freedom.index$countries <- NULL
  human.freedom.index$region <- NULL
  #human.freedom.index$year <- NULL
  rownames(human.freedom.index)=human.freedom.index[,1]
  human.freedom.index$co_year <- NULL
```

1.2 Removing Columns with Many NAs

The following code reduces the dataset to `ncol(human.freedom.index)` columns, which is more than a 50% reduction in features. Of the columns, 1 is the response **hf_score**, and two are alternative categorical responses (hf_rank (rank of the hf_scores), hf_quartile (the quartile of hf_scores)). The remaining 39 are predictors.

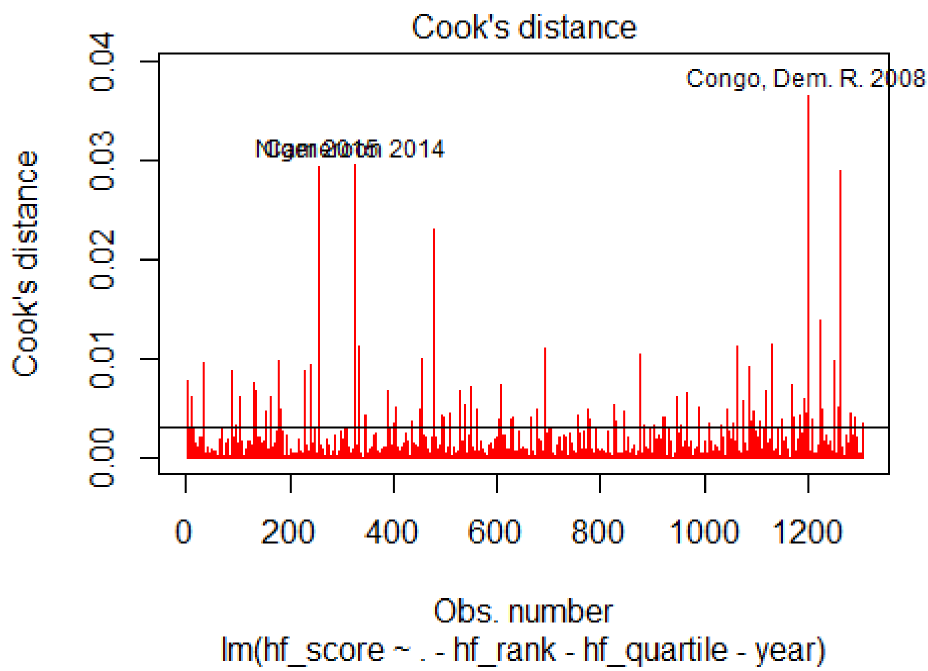
```
cols.to.drop2 = list()    # will create list of column names to drop from
dataset
list_counter = 1          # index for the list() object
for (i in 1:ncol(human.freedom.index)){
  # checking if the number of N/A's is 100 or more
  #     (seemed good with preliminary inspection)
  if(sum(is.na(human.freedom.index[,i])) >= 100 ){
    # append the column name that has 100+ N/A's
    cols.to.drop2[[list_counter]] = colnames(human.freedom.index[i])
    list_counter = list_counter + 1     # update the counter
  }
```

```
}
# now to 'vectorize' the list by unlist()-ing
cols.to.drop2 = unlist(cols.to.drop2)
ncol(human.freedom.index)   # number of features BEFORE dropping

## [1] 98

human.freedom.index = human.freedom.index %>%
  dplyr::select(-cols.to.drop2)
ncol(human.freedom.index)   # number of features AFTER dropping due to NAs

## [1] 43

nrow(human.freedom.index)   # number of rows BEFORE dropping

## [1] 1458

# drop rows with ANY N/A's (this will bias the dataset)
human.freedom.index = human.freedom.index %>% na.omit()
nrow(human.freedom.index)   # number of rows AFTER dropping due to NAs

## [1] 1305
```

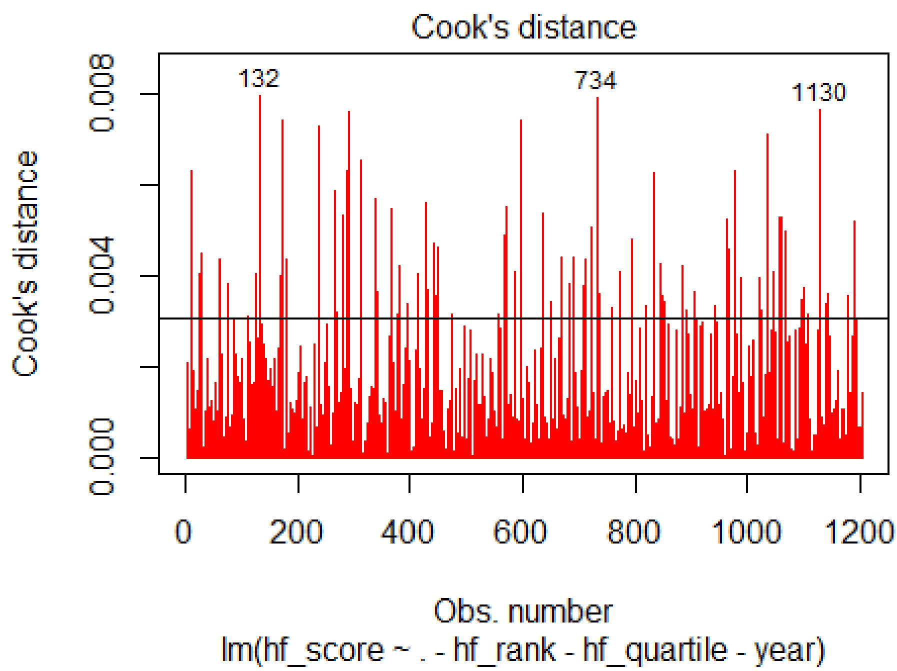
1.3 Removing Outliers and Leverage Points

We have some outliers and leverage points, we remove them with code found here: https://stats.stackexchange.com/questions/164099/removing-outliers-based-on-cooks-distance-in-r-language/345040

```
lm.fit =lm(hf_score~ . -hf_rank -hf_quartile -year ,
data=human.freedom.index)
plot(lm.fit, pch=18, col="red", which=c(4))
abline(h = 4/nrow(human.freedom.index), col="black")
```




```
cooksd <- cooks.distance(lm.fit)
human.freedom.index$cooksd = cooksd
human.freedom.index$outlier = ifelse(cooksd > 4/(nrow(human.freedom.index)),
"Y", "N")
human.freedom.index.removed.outliers = human.freedom.index %>%
  filter(outlier != 'Y') %>% dplyr::select(-c(outlier, cooksd))

lm.fit =lm(hf_score~ . -hf_rank -hf_quartile -year,
data=human.freedom.index.removed.outliers)
plot(lm.fit, pch=18, col="red", which=c(4))
abline(h = 4/nrow(human.freedom.index), col="black")
```




```
human.freedom.index = human.freedom.index %>% dplyr::select(-c(outlier,
cooksd))
nrow(human.freedom.index.removed.outliers)   # number of rows AFTER dropping
due to outliers/leverage points

## [1] 1204

nrow(human.freedom.index.removed.outliers)/nrow(human.freedom.index)

## [1] 0.9226054
```

1.4 Creating Dataframes for Analysis:

```
# vector of all column names that are responses (or forms of the responses)
# leave rank/quartile in case
responses = c("hf_score", "hf_rank", "hf_quartile")
# vector of main response
main.response = responses[1]
# vector of all non-features MINUS hf_score (primary response is retained)
other.response = responses[-1]

# 2 functions to take dataset and filter for features or response
filter.HFI.features = function(data){
  newdata = data %>% dplyr::select(-responses)
  return(newdata)
}
```

```
filter.HFI.response = function(data){
  newdata = data %>% dplyr::select(main.response)
  return(newdata)
}

# these dfs contain ALL observations without NAs
hfi.response = filter.HFI.response(human.freedom.index)
hfi.features = filter.HFI.features(human.freedom.index)
hfi.combined = cbind(hfi.features, hfi.response)

# these dfs remove influential observations
hfi.response.no.outlier =
filter.HFI.response(human.freedom.index.removed.outliers)
hfi.features.no.outlier =
filter.HFI.features(human.freedom.index.removed.outliers)
hfi.combined.no.outlier = cbind(hfi.features.no.outlier,
hfi.response.no.outlier)
```

1.5 Creating Train and Test Sets

```
# set the seed to
set.seed(111)

generate.train.test = function(data){
  combined.train = data %>% sample_frac(size=0.8)
  combined.test = data %>% setdiff(combined.train)

  tss.hf_score = mean((mean(data$hf_score) -
                      data$hf_score)^2)
  tss.test.hf_score = mean((mean(combined.test$hf_score) -
                        combined.test$hf_score)^2)
  # Total Sum of Squares AVERAGED (since we are dealing with MSE)
  tss.hf_score
  tss.test.hf_score
  ret.list = list("train" = combined.train, "test" = combined.test,
                  "tss" = tss.hf_score, "tss.test" = tss.test.hf_score)
}

hfi.combined.list = generate.train.test(hfi.combined)
```

This seperates into a 80-20 split between train-test sets. Then, we get the TSS for all residuals of the test set. This can then be used to get an R^2 with the test set later on. The test set tss TSS `hfi.combined.list$tss`.

2. Linear Regression and Model Selection Methods:

2.1 Train Test Split Estimates of Error

2.1.1 Linear Regression with all 39 Predictors

```
lm.fit = lm(hf_score ~ . -year, data = hfi.combined.list[["train"]])
lm.preds = predict(lm.fit, newdata = hfi.combined.list[["test"]])
# get MSE and R^2 (just 1 - MSE/MEAN(TSS) === 1 - RSS/TSS)
mse.lm = mean((lm.preds - hfi.combined.list[["test"]]$hf_score)^2)
lm.r2 = 1 - (mse.lm/hfi.combined.list[["tss.test"]])
mse.lm

## [1] 0.04620466

lm.r2

## [1] 0.9554123
```

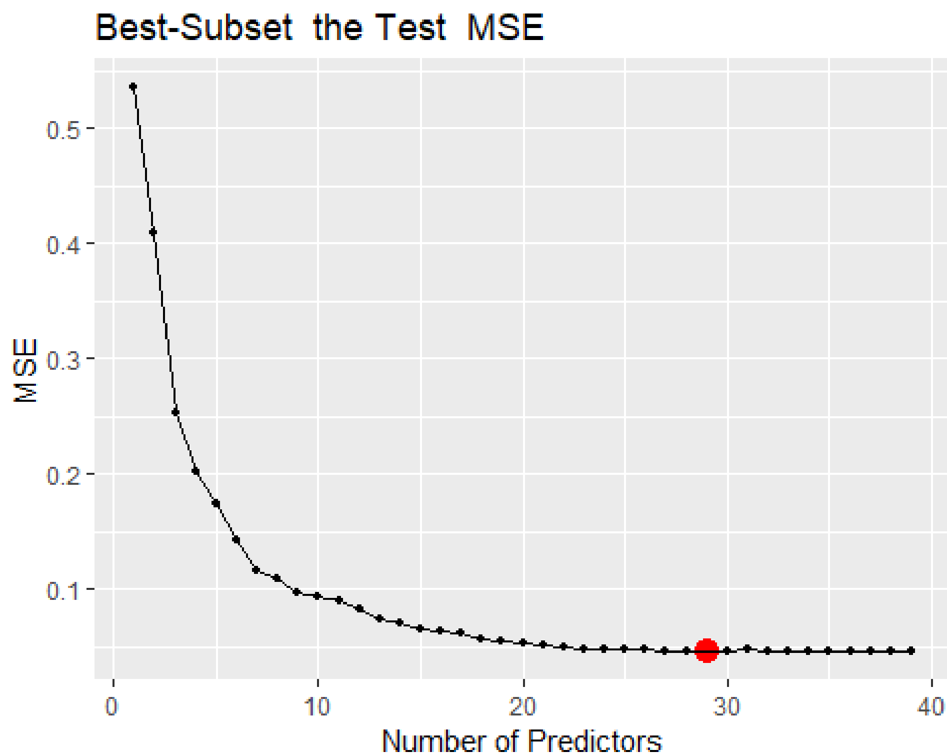
The R^2 of the FULL model using Linear Regression is: `lm.r2`. This is very high, and shows there is alot of potential trying to predict with this dataset.

2.1.2 Best Subset Selection

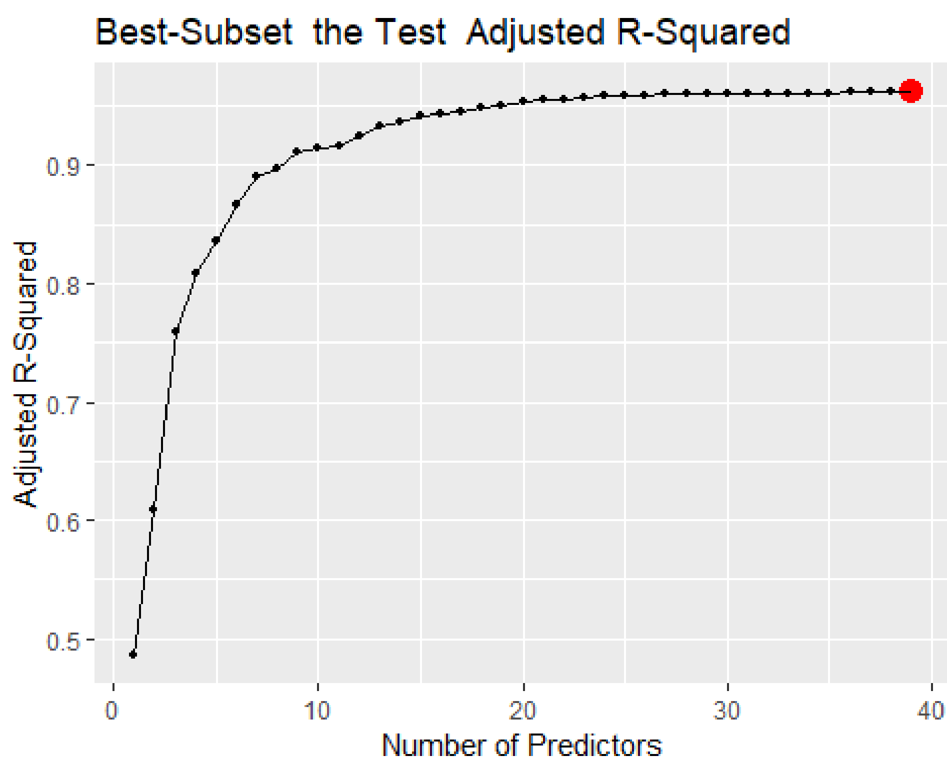
```
set.seed(111)
get.diff.errors = function(rss, N, mtss, p, harsh.penalty = 5){
# These are SIMPLIFIED Formulas:
# AIC = LL + 2 * DF.LL
# BIC = LL + 2 * DF.LL
# ADJ = 1 - [(MSE/MEAN(TSS)) * ((N-P-1)/(N-1))]
# P = # predictors; N = number of obs in test set

  base.part = N * (log(2*pi) + 1 + log(rss/N))
  bic    = base.part + (log(N)*(p+1))
  aic    = base.part + (harsh.penalty *(p+1))
  adjr2 = 1 - ((rss/N)/mtss) * ((N - p - 1 )/ (N-1))
  return(list("bic" = bic,
              "aic" = aic,
              "adjr2" = adjr2))
}

# AIC with larger constant than 2 (BIC is roughly == 2 aswell,
# so larger constant needed for harsher penalty)
harsh.penalty = 10

get.best.subset = function(data.train, data.test, mtss, max.vars=20){
  best.subset = regsubsets(hf_score ~ . -year,
                           data = data.train,
                           nvmax = max.vars,
                           intercept = T,
```

```r
                            method = "exhaustive")

  best.subset.mse = rep(NA, max.vars)
  best.subset.aic = rep(NA, max.vars)
  best.subset.bic = rep(NA, max.vars)
  best.subset.adj = rep(NA, max.vars)

  for (i in 1:max.vars){
    coef.i = coef(best.subset, id = i)
    temp.pred = as.matrix(
      data.test[,colnames(data.test) %in% names(coef.i)] ) %*%
      coef.i[names(coef.i) %in% colnames(data.test)]
    temp.pred = as.vector(temp.pred) + coef.i["(Intercept)"]
    # MSE = RSS / N (Average of Residuals)
    rss = sum((temp.pred- data.test$hf_score )^2)
    best.subset.mse[i] = rss/nrow(data.test)

    other.errors = get.diff.errors(rss, nrow(data.test),mtss,
                                   p = i, harsh.penalty = harsh.penalty)
    best.subset.bic[i] = other.errors$bic
    best.subset.aic[i] = other.errors$aic
    best.subset.adj[i] = other.errors$adjr2
  }
  return(list("model" = best.subset,
              "best.subset.mse"   = best.subset.mse,
              "best.subset.adjr2" = best.subset.adj,
              "best.subset.bic"   = best.subset.bic,
              "best.subset.aic"   = best.subset.aic
              ))
}

# store the results of the best-subset regressions
max.variables.to.run = ncol(hfi.combined.list$train) - 2
best.subset.result = get.best.subset(hfi.combined.list$train,
hfi.combined.list$test,
                         hfi.combined.list$tss.test, max.variables.to.run)

plot.test.errors = function(title, ylab.val, errors, max=FALSE){
  xbound = length(errors)
  df = data.frame(matrix(nrow=xbound, ncol = 0))
  df$pred = 1:xbound
  df$errors = errors
  df$isOpt = rep(FALSE, xbound)

  if(max){
    opt.val = which.max(errors)
    df$isOpt[opt.val] = TRUE
  } else{
    opt.val = which.min(errors)
    df$isOpt[opt.val] = TRUE
```

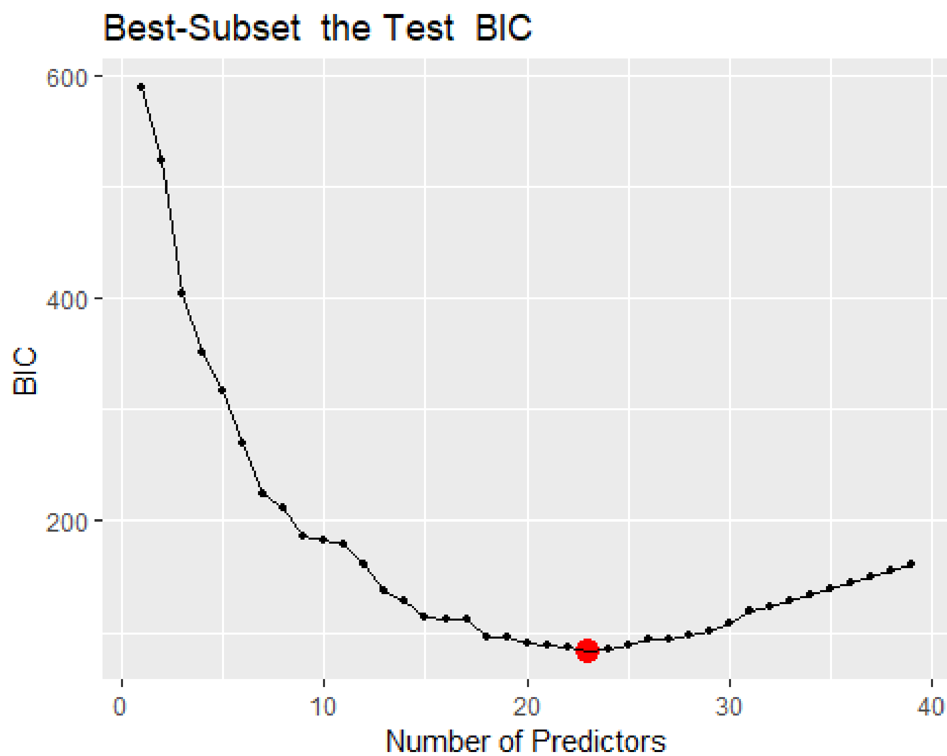
```r
  }

  g = ggplot(data = df, aes(x = pred, y = errors)) +
    geom_point(aes(color=isOpt, size=isOpt)) +
    geom_line() + ggtitle(title) + ylab(ylab.val) +
    xlab("Number of Predictors") +
    scale_color_manual(values = c("black", "red")) +
    scale_size_manual(values = c(1, 4)) +
    guides(color=FALSE, size=FALSE)
  print(g)
}

print.out.graphs.lm = function(model.list, harsh.penalty, title.name){
  list.keys = names(model.list)[-1]
  title.keys = c("MSE", "Adjusted R-Squared", "BIC",
                 paste("AIC (", harsh.penalty, ")",sep=""))
  for( i in 1:length(list.keys)){
    if(title.keys[i] == "Adjusted R-Squared"){
      plot.test.errors(title = paste(title.name, " the Test ",
title.keys[i]),
                       ylab=title.keys[i],
                       errors = unlist(model.list[[list.keys[i]]]),
                       max=TRUE)
    } else{
      plot.test.errors(title = paste(title.name , " the Test ",
title.keys[i]),
                       ylab=title.keys[i],
                       errors = unlist(model.list[[list.keys[i]]]))
    }
    cat("Current Metric: Test ", title.keys[i],"\n")
    if( title.keys[i]=="Adjusted R-Squared"){
      cat("Maximum Value: ", which.max(model.list$best.subset.adjr2),
          "\n Number of Predictors: ",

model.list$best.subset.adjr2[which.max(model.list$best.subset.adjr2)])
    } else{
      cat("Minimum Value: ",
model.list[[list.keys[i]]][which.min(model.list[[list.keys[i]]])],
          "\n Number of Predictors: ",which.min(model.list[[list.keys[i]]]) )

    }
    cat("\n\n")
  }
}

print.out.graphs.lm(best.subset.result, harsh.penalty = harsh.penalty,
title.name = "Best-Subset")
```



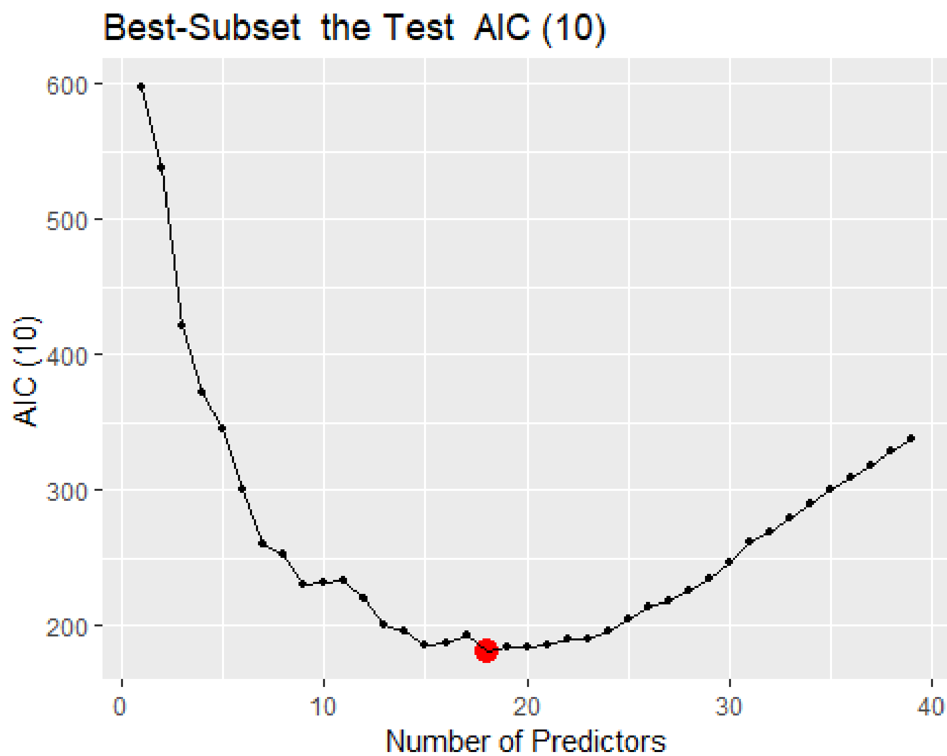

```
## Current Metric: Test  MSE
## Minimum Value:  0.04562551
##  Number of Predictors:  29
```



```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value:  39
##  Number of Predictors:  0.9621004
```




```
## Current Metric: Test  BIC
## Minimum Value:  83.0116
##  Number of Predictors:  23
```




```
## Current Metric: Test  AIC (10)
## Minimum Value:  180.5004
##  Number of Predictors:  18
```

We utilize a best subset model selection method that uses training RSS to define the best model of each size. After reaching about 10 predictors, the test MSE started flattening, and decreasing only slightly. Overall, the best subset is relatively good at showing most important factors in hf_score.

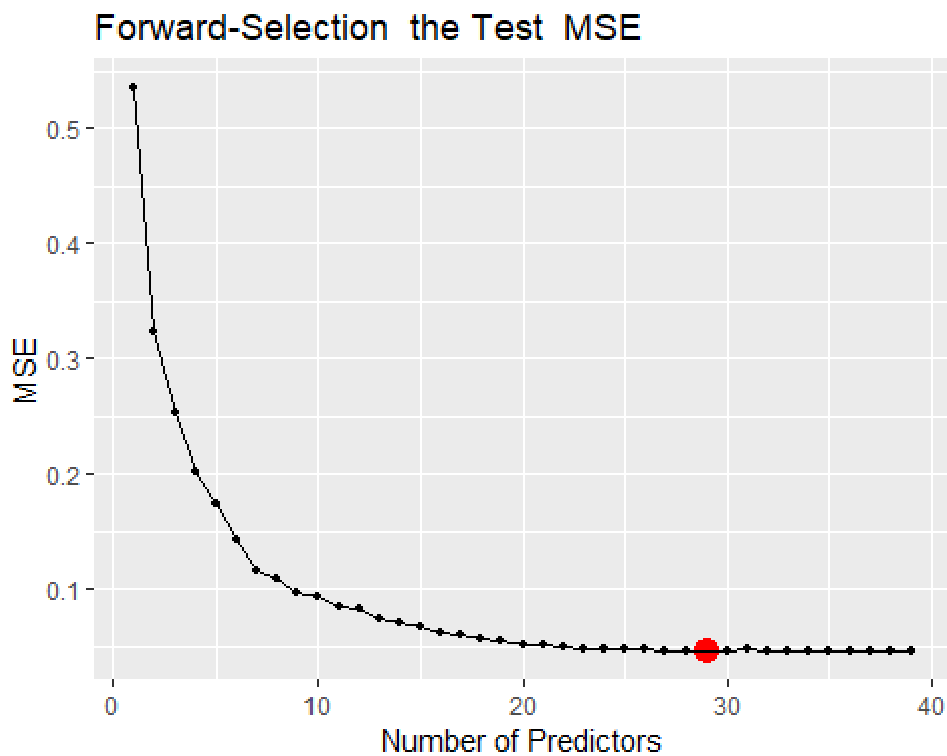
NOTES: What to ADD: - utilize cross-validation to pick best subset (this will take ALONG time) using k=4 or k=5 for number of folds (too many and it will take LONG TIME) Note: Max 3/21 (I'm not sure if we need to do this. It would be ideal, but probably not necessary for the purpose of the course. That being said, if someone wants to try to code it up, go for it!) - Utilizing BIC/*AIC*/Mallow's CP/R^2 Adjusted to penalize the mse on test sets for more predictors present in model

2.1.3 Forward Model Selection

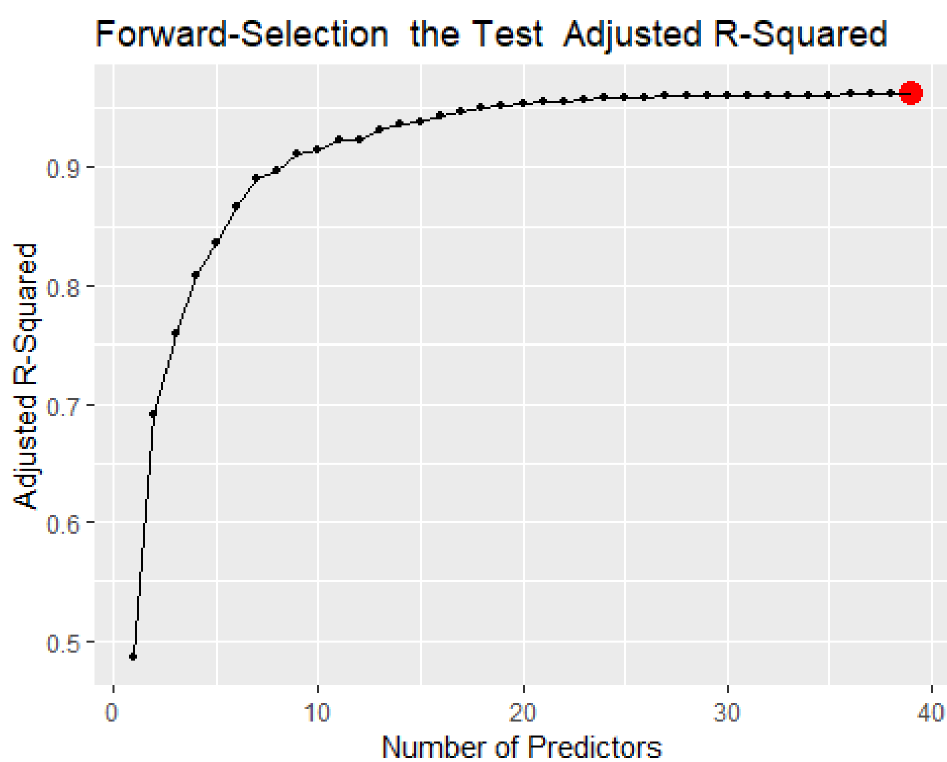
Probably better approach than backwards, since we want to eliminate the most variables in order to find a sparse list of features to predict HFI score.

```
# set to 'forward' selection, allow max variables to be added
do.selection.methods = function(data.train, data.test, mtss, max.vars, type){
  if(type == "forward"){
     fit = regsubsets(hf_score ~ . -year, method = "forward",
```

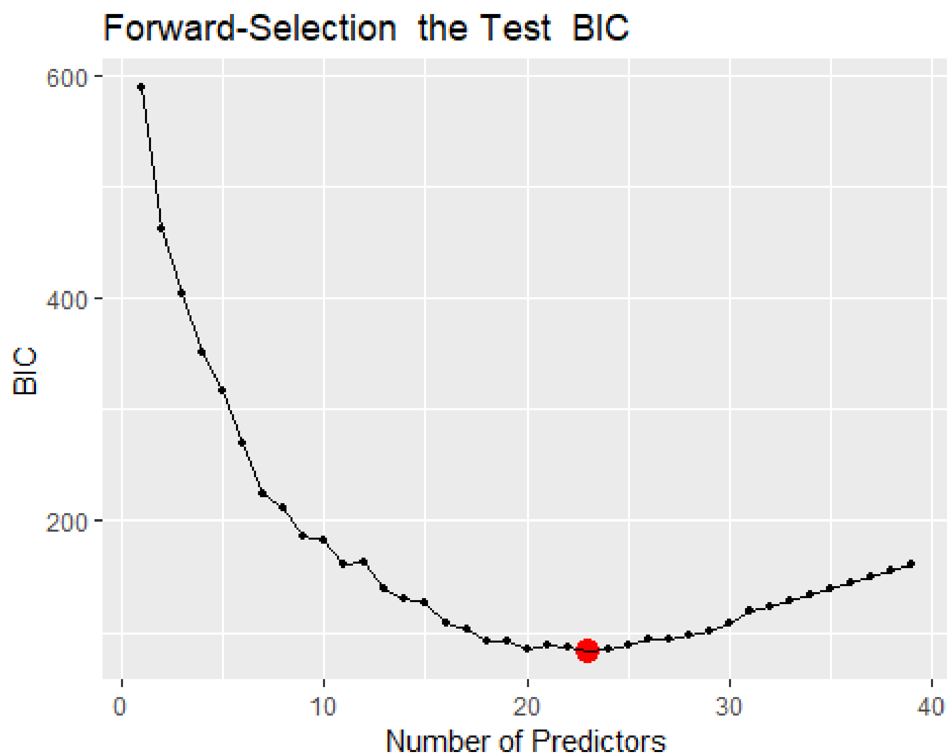
```r
                             nvmax = max.vars, data = data.train)
  } else{
    fit = regsubsets(hf_score ~ . -year, method = "backward",
                             nvmax = max.vars, data = data.train)
  }
  test.model.matrix = model.matrix(hf_score ~ . -year, data = data.test)
  method.mse    = rep(NA, max.vars)
  method.aic    = rep(NA, max.vars)
  method.bic   = rep(NA, max.vars)
  method.adjr2 = rep(NA, max.vars)

  for( i in 1:max.vars){
    # AGAIN, using pipes to efficiently create predictions
    coef.i = coef(fit, id = i)
    preds = test.model.matrix[,names(coef.i)] %*% coef.i
    # create the MSE and then AIC
    rss = sum((preds - data.test$hf_score)^2)
    method.mse[i] = rss/nrow(data.test)

    other.errors = get.diff.errors(rss, nrow(data.test), mtss,
                                       p = i, harsh.penalty = harsh.penalty)

    method.aic[i]   = other.errors$aic
    method.bic[i]   = other.errors$bic
    method.adjr2[i] = other.errors$adjr2
  }
  return(list("model" = fit,
              "forward.mse"   = method.mse,
              "forward.adjr2" = method.adjr2,
              "forward.bic"   = method.bic,
              "forward.aic"   = method.aic
              ))
}

forward.results = do.selection.methods(hfi.combined.list$train,
hfi.combined.list$test,
                         hfi.combined.list$tss.test,
ncol(hfi.combined.list$test)-2, type = "forward")
print.out.graphs.lm(forward.results, harsh.penalty = harsh.penalty,
title.name = "Forward-Selection")
```



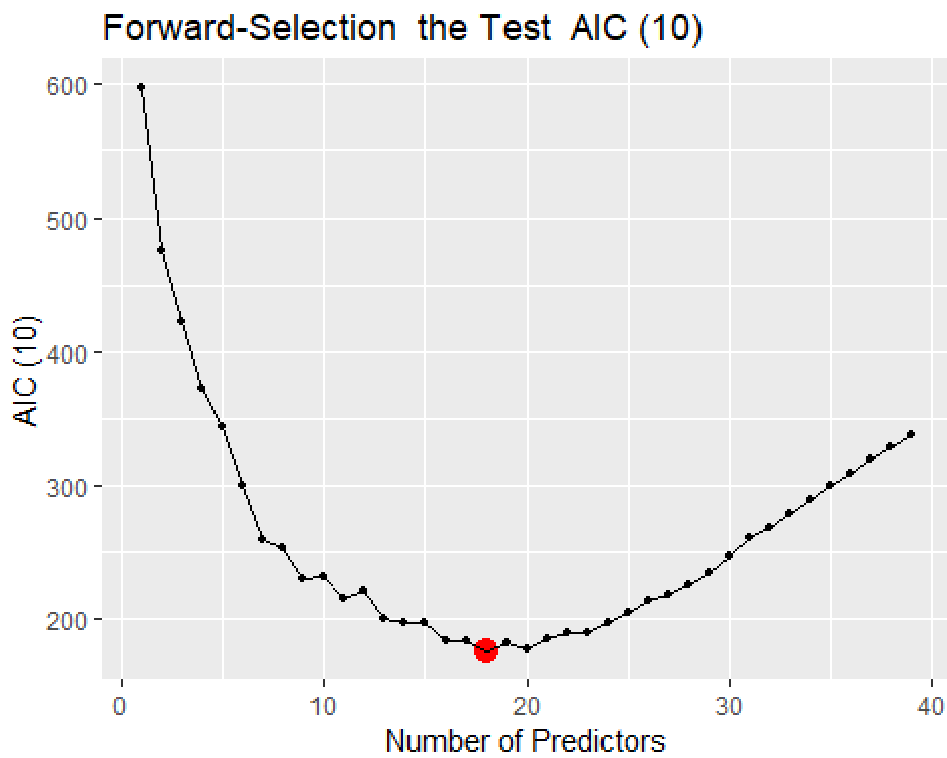

```
## Current Metric: Test  MSE
## Minimum Value:  0.04562551
##  Number of Predictors:  29
```



```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value:
##  Number of Predictors:
```




```
## Current Metric: Test  BIC
## Minimum Value:  83.0116
##  Number of Predictors:  23
```

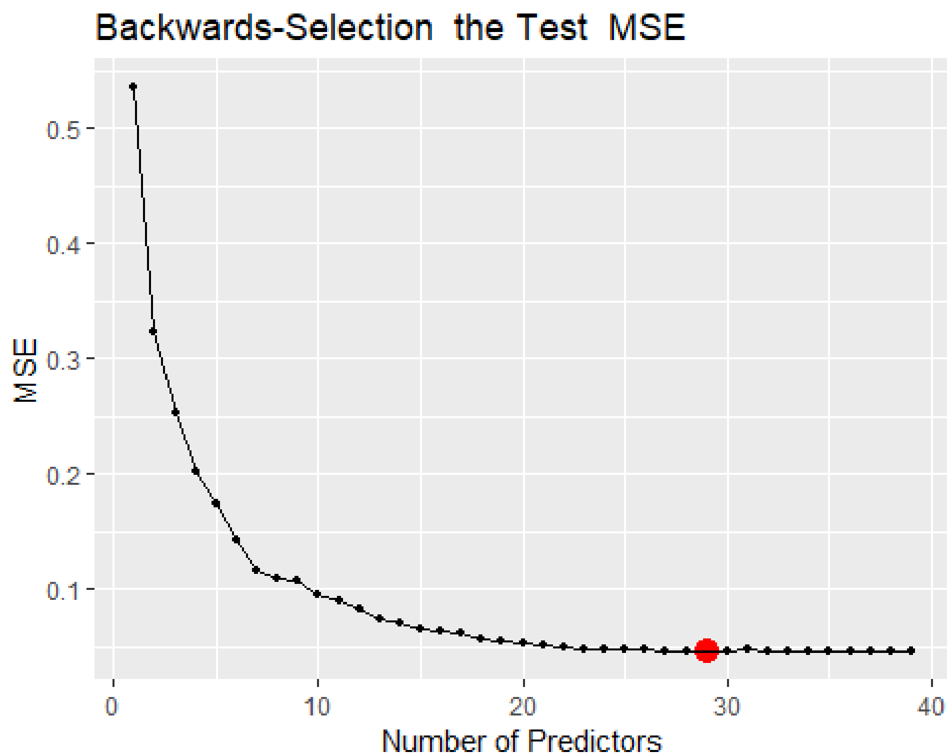



```
## Current Metric: Test  AIC (10)
## Minimum Value:  175.7513
##  Number of Predictors:  18
```

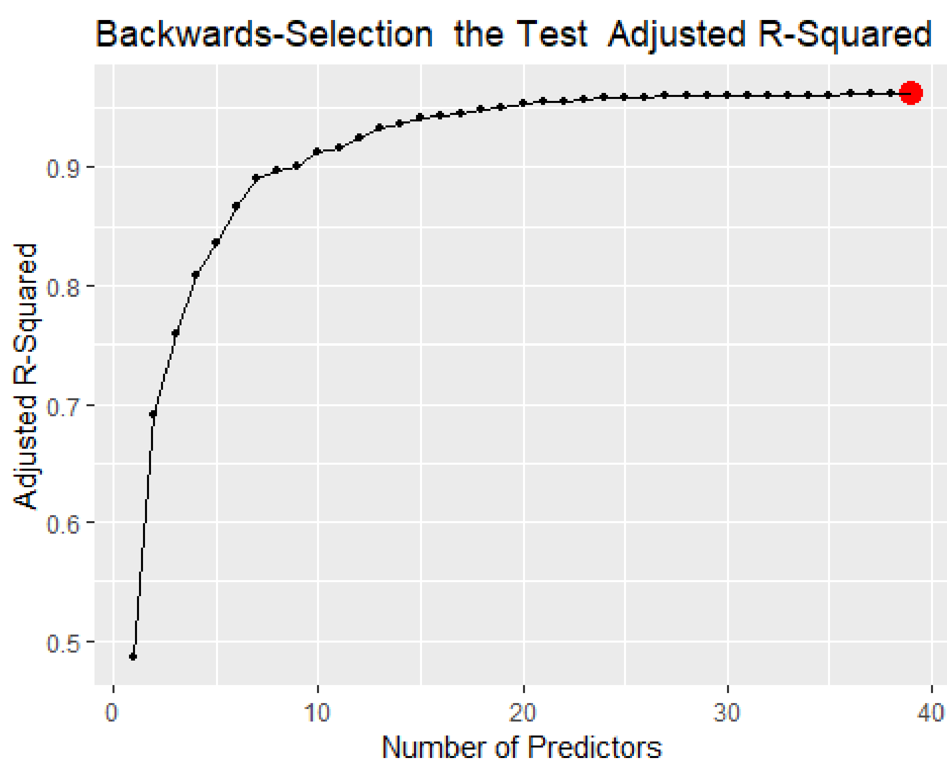
We utilize a forward model selection method that uses training RSS to define the best model of each size. The model with the lowest test MSE of `mse.forward[which.min(mse.forward)]`includes `which.min(mse.forward)` predictors. Like best subset, after reaching about 10 predictors, the test MSE starts flattening as additional predictors are added. Overall, the best subset is relatively good at showing most important factors in hf_score.

2.1.4 Forward Model Selection

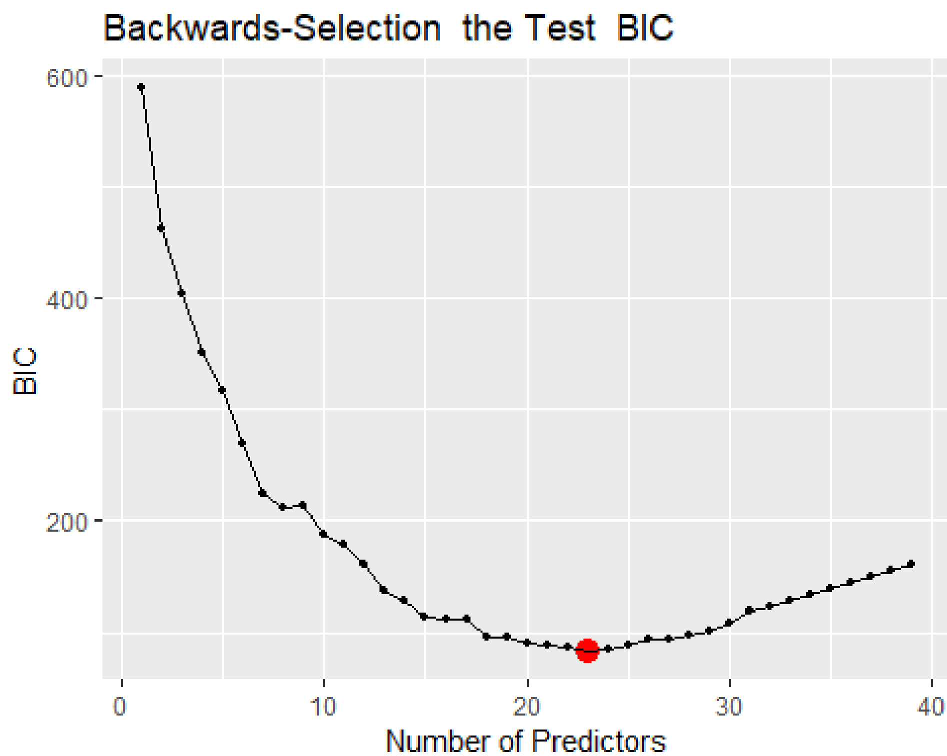
```
backwards.results = do.selection.methods(hfi.combined.list$train,
hfi.combined.list$test,
                         hfi.combined.list$tss.test,
ncol(hfi.combined.list$test)-2, type = "backwards")
print.out.graphs.lm(backwards.results, harsh.penalty = harsh.penalty,
title.name = "Backwards-Selection")
```



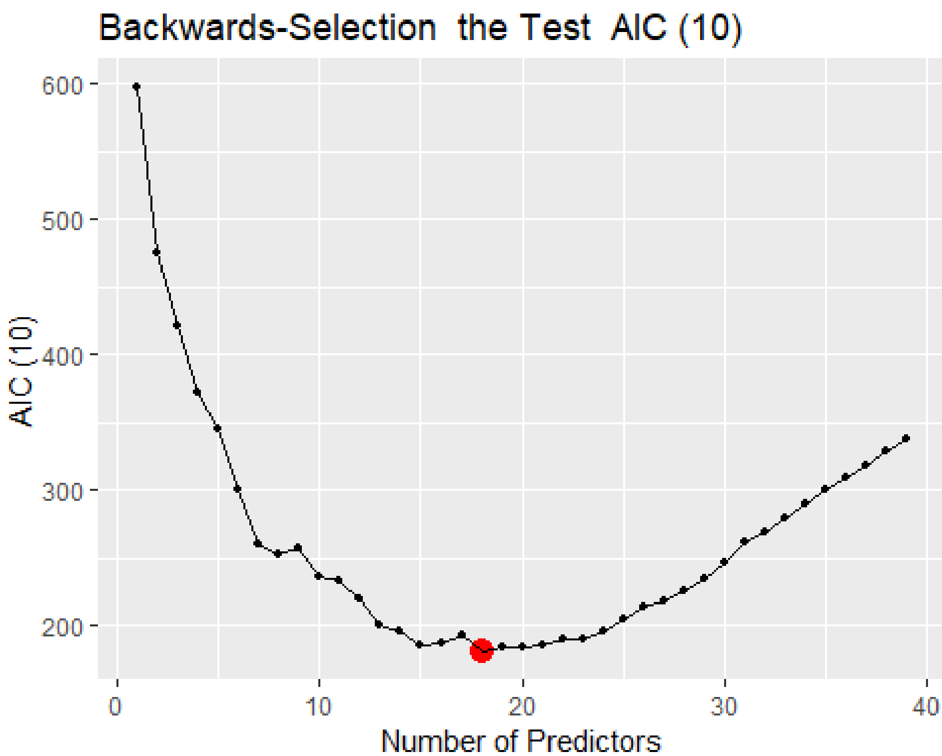

```
## Current Metric: Test  MSE
## Minimum Value:  0.04562551
##  Number of Predictors:  29
```



```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value:
##  Number of Predictors:
```




```
## Current Metric: Test  BIC
## Minimum Value:  83.0116
##  Number of Predictors:  23
```




```
## Current Metric: Test  AIC (10)
## Minimum Value:  180.5004
##  Number of Predictors:  18
```

We utilized a backwards selection method that develops models

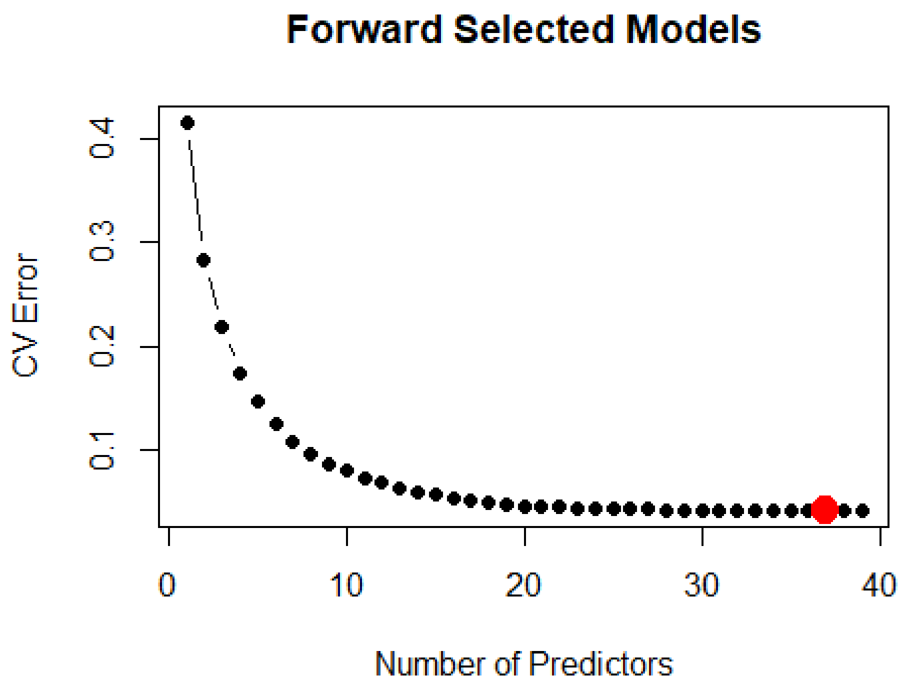
We utilize a forward model selection method that uses training RSS to define the best model of each size. The model with the lowest test MSE of `mse.forward[which.min(mse.forward)]`includes `which.min(mse.forward)` predictors. Like best subset, after reaching about 10 predictors, the test MSE starts flattening as additional predictors are added. Overall, the best subset is relatively good at showing most important factors in hf_score.

2.2 Cross Validation Estimates of Error

Note: Max 3/21 (Andrew, the following code (making the models formulas so we can run CV) is awesome. Great thinking! I added to it so we can also compare models of each size on the CV Error with plots!

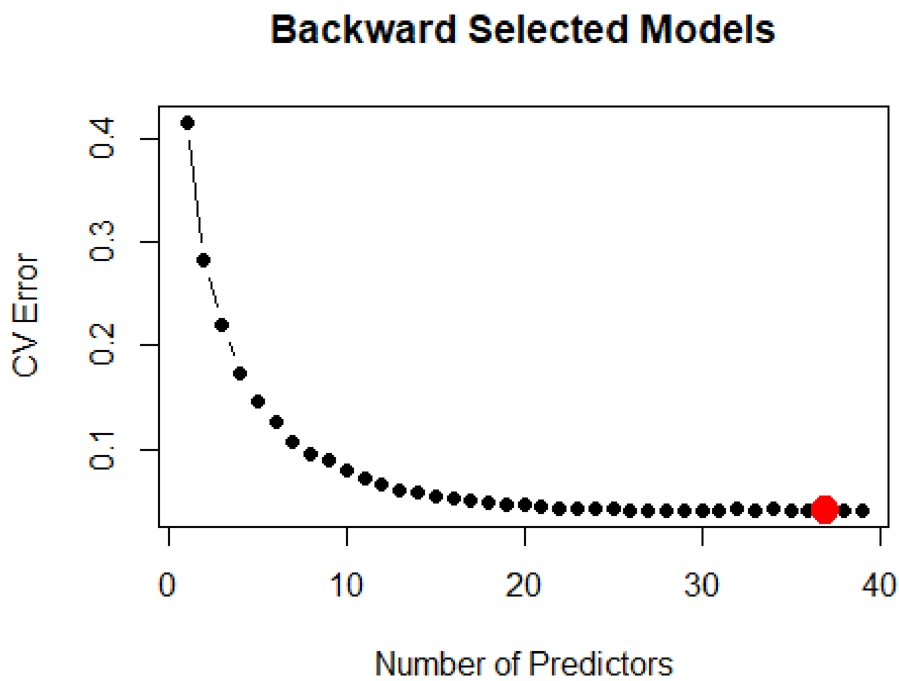
```
set.seed(111)
# Set the number of folds for CV
# This code collects the models as formulas that from forward and best-subset selection
# The goal is to then take the formulas and run cross-validation
```

```
get.cv.errors = function(data.hfi, regression.results, number.of.folds=10){
  cv.err = rep(NA, ncol(data.hfi)-2)
  for(i in 1:(ncol(data.hfi)-2)){
    form = paste(names(coef(regression.results[["model"]], id = i))[-1],
collapse = " + " )
    form = as.formula(paste("hf_score ~ ", form, sep = ""))
    cv = cv.glm(data = data.hfi, K = number.of.folds, glmfit = glm(formula =
form,
                                                                    data =
data.hfi))
    cv.err[i] = cv$delta[1]
  }
  return(cv.err)
}

plot.cv.errors = function(cv.errors, title.string){
  p1 = plot(cv.errors, main=paste(title.string), xlab = "Number of
Predictors",
       ylab = "CV Error", pch = 19, type = "b")
  points (which.min(cv.errors), cv.errors[which.min(cv.errors)],
          col ="red",cex=2,pch =19)
  print(p1)
}

dataset.hfi = hfi.combined.list$train %>% full_join(hfi.combined.list$test)

cv.err.results = list()
cv.err.results[["forward"]] = get.cv.errors(data.hfi = dataset.hfi,
                                          regression.results =
forward.results)
cv.err.results[["backward"]] = get.cv.errors(data.hfi = dataset.hfi,
                                           regression.results =
backwards.results)
cv.err.results[["best.subset"]] = get.cv.errors(data.hfi = dataset.hfi,
                                              regression.results =
best.subset.result)

cv.full.lm = cv.glm(data = dataset.hfi, K = 10, glmfit = glm(formula =
hf_score ~ . -year, data = dataset.hfi))

cv.full.lm.err = cv.full.lm$delta[1]
cv.err.results[["full.lm"]] = cv.full.lm.err


plot.cv.errors(cv.err.results$forward, title.string = "Forward Selected
Models")
```



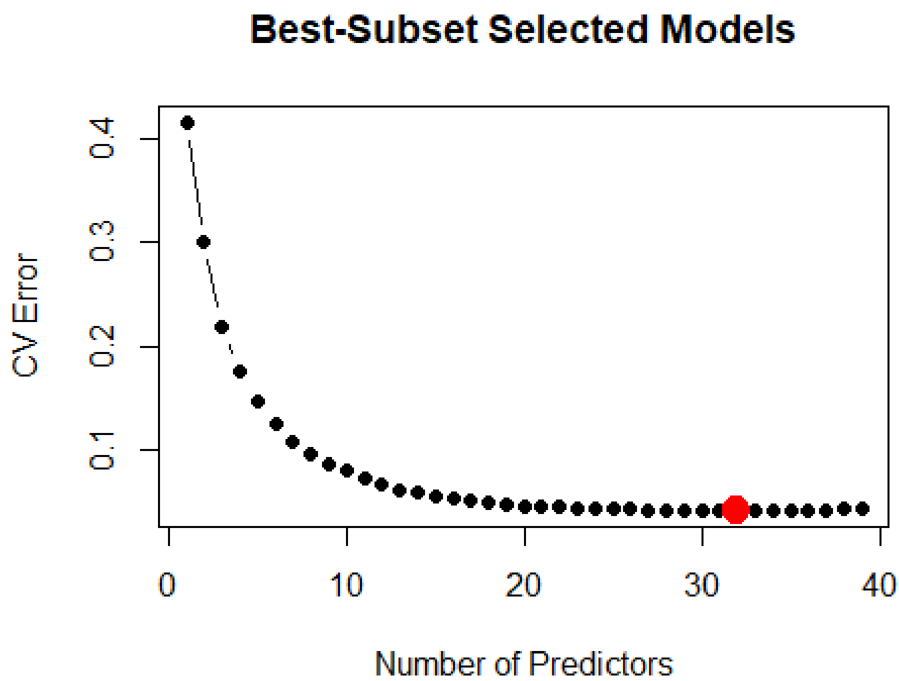

```
## NULL
```

```
plot.cv.errors(cv.err.results$backward, title.string = "Backward Selected
Models")
```




```
## NULL
```

```r
plot.cv.errors(cv.err.results$best.subset, title.string = "Best-Subset
Selected Models")
```




```
## NULL

# These are the errors from the best subset models with the lowest errors:
which.min(cv.err.results$forward)

## [1] 37

cv.err.results$forward[which.min(cv.err.results$forward)]

## [1] 0.04147359

fit4=cv.err.results$forward[which.min(cv.err.results$forward)]

which.min(cv.err.results$best.subset)

## [1] 32

cv.err.results$best.subset[which.min(cv.err.results$best.subset)]

## [1] 0.04148839

fit6= cv.err.results$best.subset[which.min(cv.err.results$best.subset)]

# total preds (-1 because 1 is year that is unused)
ncol(hfi.features)-1

## [1] 39

cv.err.results$full.lm
```

```
## [1] 0.04230932
```

```r
fit2= cv.err.results$full.lm
hfi.combined.list$tss
```

```
## [1] 0.9762658
```

The cross-validated errors are very similar. The full model does very well, but the smaller models capture almost the same exact amount of correlation.

This cross-validation basically shows that the data is highly correlated, and a almost perfect model can be discovered, as the average TSS is `hfi.combined.list$tss`, which means that all of the models explain over 95% of the variance in the outcome variable

Here are the `which.min(cv.err.results$best.subset)` coefficients in the model with the lowest 10-fold CV Error, selected using best subset selection:

```r
coef(forward.results$model,  which.min(cv.err.results$forward))
```

```
##                      (Intercept)                   pf_ss_homicide
##                      0.202748688                      0.049066578
##          pf_ss_disappearances_disap     pf_ss_disappearances_violent
##                      0.009123559                      0.009689860
##    pf_ss_disappearances_fatalities    pf_ss_disappearances_injuries
##                      0.027135610                     -0.021643655
##              pf_movement_domestic              pf_movement_foreign
##                      0.018815180                      0.017211770
##            pf_religion_harassment          pf_religion_restrictions
##                      0.027146387                      0.021488102
##              pf_expression_killed           pf_expression_influence
##                      0.006575982                      0.043641600
##             pf_expression_control              pf_identity_sex_male
##                      0.059045099                      0.021376272
##            pf_identity_sex_female        ef_government_consumption
##                      0.014956662                      0.040247916
##                   ef_legal_courts                ef_legal_military
##                      0.069192838                      0.043231528
##              ef_legal_enforcement                  ef_legal_gender
##                      0.046582639                      1.023472235
##                   ef_money_growth                      ef_money_sd
##                      0.037608105                      0.036380271
##                ef_money_inflation                ef_money_currency
##                      0.026921243                      0.029094871
##             ef_trade_tariffs_mean              ef_trade_tariffs_sd
##                      0.045064991                     -0.007711629
##                  ef_trade_tariffs              ef_trade_regulatory
##                      0.015494440                      0.049194770
##                    ef_trade_black        ef_trade_movement_capital
##                      0.012476095                      0.012898757
##          ef_trade_movement_visit      ef_regulation_credit_private
##                      0.015016463                      0.007107562
```

```
##                ef_regulation_credit        ef_regulation_labor_minwage 
##                         0.042479205                        0.001081308 
##           ef_regulation_labor_hours   ef_regulation_labor_conscription 
##                         0.011788144                        0.004385670 
##        ef_regulation_business_start ef_regulation_business_compliance 
##                         0.017563458                        0.001330272
```

Additionally, here are the `which.min(cv.err.results$best.subset)` coefficients in the model with the lowest 10-fold CV Error, selected using forward selection:

```
coef(best.subset.result$model, which.min(cv.err.results$best.subset))
```

```
##                       (Intercept)                    pf_ss_homicide 
##                       0.228307654                       0.050554992 
##         pf_ss_disappearances_disap      pf_ss_disappearances_violent 
##                       0.008292140                       0.009419556 
##   pf_ss_disappearances_fatalities     pf_ss_disappearances_injuries 
##                       0.028635669                      -0.021079775 
##               pf_movement_domestic               pf_movement_foreign 
##                       0.018433804                       0.016998936 
##             pf_religion_harassment          pf_religion_restrictions 
##                       0.026559616                       0.020799832 
##               pf_expression_killed           pf_expression_influence 
##                       0.006112810                       0.042943167 
##              pf_expression_control              pf_identity_sex_male 
##                       0.060182275                       0.022168447 
##             pf_identity_sex_female        ef_government_consumption 
##                       0.014688203                       0.040595054 
##                    ef_legal_courts                 ef_legal_military 
##                       0.071481222                       0.042783543 
##               ef_legal_enforcement                   ef_legal_gender 
##                       0.047089020                       1.031191271 
##                    ef_money_growth                       ef_money_sd 
##                       0.037831918                       0.033942793 
##                 ef_money_inflation                 ef_money_currency 
##                       0.026068433                       0.030172954 
##              ef_trade_tariffs_mean               ef_trade_regulatory 
##                       0.047617315                       0.050582254 
##                     ef_trade_black         ef_trade_movement_capital 
##                       0.013795525                       0.012302093 
##            ef_trade_movement_visit              ef_regulation_credit 
##                       0.015695580                       0.051214853 
##          ef_regulation_labor_hours ef_regulation_labor_conscription 
##                       0.011601095                       0.004240068 
##       ef_regulation_business_start 
##                       0.016957732
```

Finally, here are the `which.min(cv.err.results$backward)` coefficients in the model with the lowest 10-fold CV Error, selected using forward selection:

```
coef(backwards.results$model, which.min(cv.err.results$backward))
```

```
##                       (Intercept)                         pf_ss_homicide
##                       0.202748688                            0.049066578
##          pf_ss_disappearances_disap      pf_ss_disappearances_violent
##                       0.009123559                            0.009689860
##   pf_ss_disappearances_fatalities     pf_ss_disappearances_injuries
##                       0.027135610                           -0.021643655
##              pf_movement_domestic                   pf_movement_foreign
##                       0.018815180                            0.017211770
##            pf_religion_harassment           pf_religion_restrictions
##                       0.027146387                            0.021488102
##              pf_expression_killed             pf_expression_influence
##                       0.006575982                            0.043641600
##             pf_expression_control                 pf_identity_sex_male
##                       0.059045099                            0.021376272
##            pf_identity_sex_female            ef_government_consumption
##                       0.014956662                            0.040247916
##                   ef_legal_courts                     ef_legal_military
##                       0.069192838                            0.043231528
##              ef_legal_enforcement                       ef_legal_gender
##                       0.046582639                            1.023472235
##                   ef_money_growth                           ef_money_sd
##                       0.037608105                            0.036380271
##                ef_money_inflation                     ef_money_currency
##                       0.026921243                            0.029094871
##             ef_trade_tariffs_mean                   ef_trade_tariffs_sd
##                       0.045064991                           -0.007711629
##                  ef_trade_tariffs                   ef_trade_regulatory
##                       0.015494440                            0.049194770
##                    ef_trade_black             ef_trade_movement_capital
##                       0.012476095                            0.012898757
##           ef_trade_movement_visit      ef_regulation_credit_private
##                       0.015016463                            0.007107562
##              ef_regulation_credit        ef_regulation_labor_minwage
##                       0.042479205                            0.001081308
##         ef_regulation_labor_hours  ef_regulation_labor_conscription
##                       0.011788144                            0.004385670
##      ef_regulation_business_start ef_regulation_business_compliance
##                       0.017563458                            0.001330272
```

It is encouraging that the models with the lowest 10-fold CV Error in forward and best subset selection selection include mostly the same predictors. They also show similar curves, in which reductions in 10-fold CV Error level off after adding about 10 predictors.

3 Additional Linear Model Selection and Regularization Methods:

3.1 Shrinkage Methods

3.1.1 Ridge Regression

```
set.seed(111)
do.ridge.lasso = function(data.train, data.test, alpha){
  grid = 10^seq(10,-2, length = 1000)
  train.model = model.matrix(hf_score ~ . - year, data = data.train)[,-1]
  test.model = model.matrix(hf_score ~ . - year, data = data.test)[,-1]

  cv.model =  cv.glmnet(train.model, data.train$hf_score, alpha = alpha,
                       lambda = grid, thresh = 1e-12)
  final.model = glmnet(train.model, data.train$hf_score, alpha = alpha,
                       lambda = cv.model$lambda.min, thresh = 1e-12)
  bestlam <- cv.model$lambda.min

  pred = predict(final.model, newx = test.model, s = cv.model$lambda.min)
  mse.model = mean((data.test$hf_score - pred)^2)

  if( alpha==1 ){
    return(list("model" = final.model,
                "best.lambda" = bestlam,
                "mse" = mse.model))
  } else{
    return(list("model" = final.model,
                "best.lambda" = bestlam,
                "mse" = mse.model))
  }
}

ridge = do.ridge.lasso(hfi.combined.list$train, hfi.combined.list$test,
alpha=0)

ridge$model$beta

## 39 x 1 sparse Matrix of class "dgCMatrix"
##                                             s0
## pf_ss_homicide                     0.046531218
## pf_ss_disappearances_disap         0.009849628
## pf_ss_disappearances_violent       0.009146326
## pf_ss_disappearances_fatalities    0.024619539
## pf_ss_disappearances_injuries     -0.018342371
## pf_movement_domestic               0.018694526
## pf_movement_foreign                0.017245345
## pf_religion_harassment             0.024866416
## pf_religion_restrictions           0.022185352
## pf_expression_killed               0.006793376
## pf_expression_jailed               0.001885365
```

```
## pf_expression_influence            0.043207157
## pf_expression_control              0.057207109
## pf_identity_sex_male               0.020328535
## pf_identity_sex_female             0.015598259
## ef_government_consumption          0.035991954
## ef_legal_courts                    0.066958001
## ef_legal_military                  0.041682317
## ef_legal_enforcement               0.047388385
## ef_legal_gender                    0.996224039
## ef_money_growth                    0.036260742
## ef_money_sd                        0.038737524
## ef_money_inflation                 0.024949702
## ef_money_currency                  0.027790975
## ef_trade_tariffs_mean              0.041782449
## ef_trade_tariffs_sd               -0.009072961
## ef_trade_tariffs                   0.022689473
## ef_trade_regulatory_compliance     0.007387900
## ef_trade_regulatory                0.038947996
## ef_trade_black                     0.013754099
## ef_trade_movement_capital          0.014341714
## ef_trade_movement_visit            0.014750163
## ef_regulation_credit_private       0.007236377
## ef_regulation_credit               0.041155272
## ef_regulation_labor_minwage        0.001708228
## ef_regulation_labor_hours          0.012130553
## ef_regulation_labor_conscription   0.004559164
## ef_regulation_business_start       0.019718906
## ef_regulation_business_compliance  0.003323178
```

```
ridge$best.lambda
```

```
## [1] 0.02706652
```

```
ridge$mse
```

```
## [1] 0.04612929
```

Using the 10-fold cross validated best lamba of about `ridge$best.lambda`, in the Ridge model all 39 variables remain in the model.The test MSE rises to `ridge$mse` which is just slightly lower than the test MSE of OLS of `fit1`.

3.1.2 LASSO

```
lasso = do.ridge.lasso(hfi.combined.list$train, hfi.combined.list$test,
alpha=1)

rownames.no.na = c()
for (i in 1:length(rownames(lasso$model$beta))){
  if( lasso$model$beta[i] != 0){
    rownames.no.na = c(rownames.no.na, rownames(lasso$model$beta)[i])
```

```r
  }
}
lasso[["rownames.left"]] = rownames.no.na

lasso$model$beta
```

```
## 39 x 1 sparse Matrix of class "dgCMatrix"
##                                             s0
## pf_ss_homicide                    4.589250e-02
## pf_ss_disappearances_disap        8.103406e-03
## pf_ss_disappearances_violent      6.385708e-03
## pf_ss_disappearances_fatalities   1.296471e-02
## pf_ss_disappearances_injuries     .
## pf_movement_domestic              1.800746e-02
## pf_movement_foreign               1.679575e-02
## pf_religion_harassment            1.592288e-02
## pf_religion_restrictions          1.806421e-02
## pf_expression_killed              1.506820e-03
## pf_expression_jailed              .
## pf_expression_influence           4.131387e-02
## pf_expression_control             6.180304e-02
## pf_identity_sex_male              2.034571e-02
## pf_identity_sex_female            1.402770e-02
## ef_government_consumption         2.836518e-02
## ef_legal_courts                   6.443285e-02
## ef_legal_military                 4.124885e-02
## ef_legal_enforcement              4.591465e-02
## ef_legal_gender                   1.014985e+00
## ef_money_growth                   3.222880e-02
## ef_money_sd                       3.820108e-02
## ef_money_inflation                2.298265e-02
## ef_money_currency                 2.872070e-02
## ef_trade_tariffs_mean             4.779292e-02
## ef_trade_tariffs_sd               .
## ef_trade_tariffs                  .
## ef_trade_regulatory_compliance    .
## ef_trade_regulatory               5.362143e-02
## ef_trade_black                    1.276203e-02
## ef_trade_movement_capital         1.310275e-02
## ef_trade_movement_visit           1.499927e-02
## ef_regulation_credit_private      .
## ef_regulation_credit              4.883160e-02
## ef_regulation_labor_minwage       .
## ef_regulation_labor_hours         8.436679e-03
## ef_regulation_labor_conscription  2.986232e-03
## ef_regulation_business_start      1.500773e-02
## ef_regulation_business_compliance 2.086335e-06
```

```r
lasso$best.lambda
```

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

```r
lasso$mse
```

```
## [1] 0.0488749
```

```r
lasso$rownames.left
```

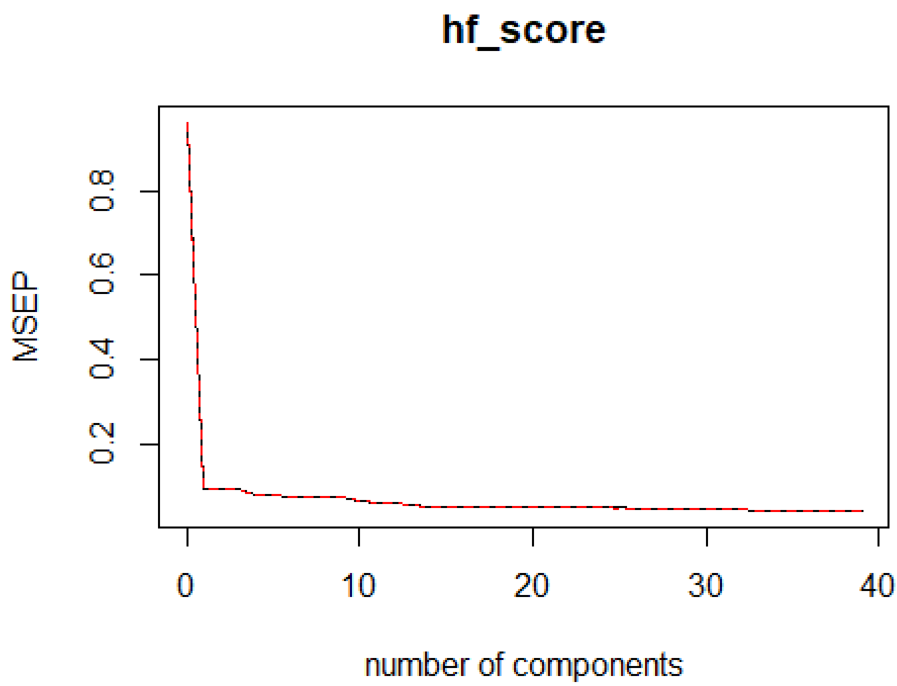
```
##  [1] "pf_ss_homicide"
##  [2] "pf_ss_disappearances_disap"
##  [3] "pf_ss_disappearances_violent"
##  [4] "pf_ss_disappearances_fatalities"
##  [5] "pf_movement_domestic"
##  [6] "pf_movement_foreign"
##  [7] "pf_religion_harassment"
##  [8] "pf_religion_restrictions"
##  [9] "pf_expression_killed"
## [10] "pf_expression_influence"
## [11] "pf_expression_control"
## [12] "pf_identity_sex_male"
## [13] "pf_identity_sex_female"
## [14] "ef_government_consumption"
## [15] "ef_legal_courts"
## [16] "ef_legal_military"
## [17] "ef_legal_enforcement"
## [18] "ef_legal_gender"
## [19] "ef_money_growth"
## [20] "ef_money_sd"
## [21] "ef_money_inflation"
## [22] "ef_money_currency"
## [23] "ef_trade_tariffs_mean"
## [24] "ef_trade_regulatory"
## [25] "ef_trade_black"
## [26] "ef_trade_movement_capital"
## [27] "ef_trade_movement_visit"
## [28] "ef_regulation_credit"
## [29] "ef_regulation_labor_hours"
## [30] "ef_regulation_labor_conscription"
## [31] "ef_regulation_business_start"
## [32] "ef_regulation_business_compliance"
```

Using the 10-fold cross validated best lamba of `bestlam.lasso`, the resulting Lasso is a better model than Ridge because it removes several predictors. The test MSE of `mse.lasso` is slightly better than Ridge (`mse.ridge`), and is lower than the test MSE of the full OLS model (`fit1`).

3.2 Dimension Reduction Methods

3.2.1 Principle Components Regression (PCR)

```
pcr.model = pcr(hf_score ~ . -year , data = hfi.combined.list$train, scale =
TRUE,
                validation = "CV")
print(validationplot(pcr.model, val.type = "MSEP"))
```



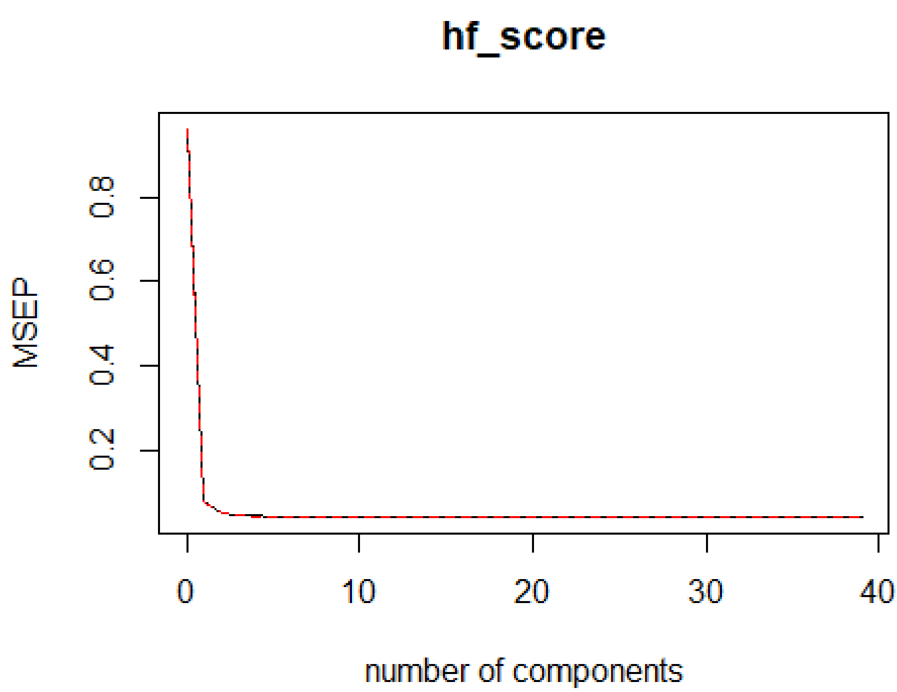

```
## NULL

find.best.mse.pcr.pls = function(model, data.test){
  min.val = 0
  lowest.mse = 1000000
  for( i in 1:model$ncomp){
    preds = predict(model, newdata = data.test, ncomp = i)
    mse = mean((preds - data.test$hf_score)^2)
    if(mse <lowest.mse){
      lowest.mse = mse
      min.val = i
    }
  }
  return(list("min.val" = min.val,
              "best.mse" = lowest.mse))
}
```

```
pcr.best = find.best.mse.pcr.pls(pcr.model, hfi.combined.list$test)
pcr.best$min.val

## [1] 37

pcr.best$best.mse

## [1] 0.0461994

fit9 = pcr.best$best.mse
```

Using 10-fold cross validation, the lowest average root MSE is the one corresponding to M=37 components.The resulting PCR gives a test MSE of `pcr.best$best.mse`, and is slightly higher than the test MSE of the full OLS model (`fit1`). Notably, the validation plot shows that after the first component is added, only marginal benefits are gained from adding more components.

3.2.2 Partial Least Squares (PLS)

```
set.seed(111)
# create pls model and plot validation; report the best mse
pls.model = plsr(hf_score ~ . -year, data = hfi.combined.list$train, scale =
TRUE,
                 validation = "CV")
validationplot(pls.model, val.type = "MSEP")
```



```r
pls.best = find.best.mse.pcr.pls(pls.model, hfi.combined.list$test)
pls.best$min.val
```

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

```r
pls.best$best.mse
```

```
## [1] 0.04569458
```

```r
fit10 = pls.best$best.mse
```

Using 10-fold cross validation, the lowest average root MSE is the one corresponding to M=9. The resulting PCR, the resulting test MSE is `pls.mse`, and is slightly higher than the test MSE of the full OLS model (`fit1`). The validation and the varience explained seem to go completely constant after 9 components. Again, the validation plot shows that after the first component is added, only marginal benefits are gained from adding more components.

```r
x = matrix(data=
           c("Full OLS, 39 Predictors (test/train)",
             "Full OLS, 39 Predictors (10-fold CV)",
             paste0("Forward Select, ",
which.min(forward.results$forward.mse), " Predictors (test/train)"),
             paste0("Forward Select, ", which.min(cv.err.results$forward), "
Predictors (10-fold CV)"),
             paste0("Backward Select, ",
which.min(backwards.results$forward.mse)," Predictors (test/train)"),
             paste0("Backward Select, ", which.min(cv.err.results$backward),
" Predictors (10-fold CV)"),
             paste0("Best Subset, ",
which.min(best.subset.result$best.subset.mse)," Predictors (test/train)"),
             paste0("Best Subset, ", which.min(cv.err.results$best.subset), "
Predictors (10-fold CV)"),
             "Ridge, 39 Predictors (test/train)",
             paste0("Lasso, ", length(lasso$rownames.left), " Predictors
(test/train)"),
             paste0("PCR, ", pcr.best$min.val, " Components, (test/train)"),
             paste0("PLSR, ", pls.best$min.val, " Components, (test/train)"),
             "Mean TSS",
mse.lm,
cv.err.results$full.lm,
min(forward.results$forward.mse), # fit 3
min(cv.err.results$forward),
min(backwards.results$forward.mse),
min(cv.err.results$backward),
min(best.subset.result$best.subset.mse),
min(cv.err.results$best.subset),
ridge$mse,
lasso$mse,
pcr.best$best.mse,
pls.best$best.mse,
hfi.combined.list$tss
```

```r
), nrow=13, ncol=2)
colnames(x) <- c("Method","test MSE")
list.x.axis = c()
for( i in 1:13){
  list.x.axis = c(list.x.axis, paste0(i))
}

df.x.all = as.data.frame(x, row.names = list.x.axis)
df.x.all

##                                          Method           test MSE
## 1         Full OLS, 39 Predictors (test/train) 0.0462046621661508
## 2         Full OLS, 39 Predictors (10-fold CV) 0.0423093203573601
## 3   Forward Select, 29 Predictors (test/train) 0.0456255104812149
## 4   Forward Select, 37 Predictors (10-fold CV) 0.0414735885073518
## 5  Backward Select, 29 Predictors (test/train) 0.0456255104812149
## 6  Backward Select, 37 Predictors (10-fold CV) 0.0413961736107952
## 7      Best Subset, 29 Predictors (test/train) 0.0456255104674457
## 8      Best Subset, 32 Predictors (10-fold CV) 0.0414883885236741
## 9            Ridge, 39 Predictors (test/train) 0.0461292892342989
## 10           Lasso, 32 Predictors (test/train) 0.0488748958474245
## 11            PCR, 37 Components, (test/train) 0.0461993970464234
## 12            PLSR, 4 Components, (test/train) 0.0456945841622288
## 13                                    Mean TSS  0.976265782714137

df.x.all$`test MSE` = round(as.numeric(as.character(df.x.all[["test MSE"]])),
5)
df.x.all

##                                          Method test MSE
## 1         Full OLS, 39 Predictors (test/train)  0.04620
## 2         Full OLS, 39 Predictors (10-fold CV)  0.04231
## 3   Forward Select, 29 Predictors (test/train)  0.04563
## 4   Forward Select, 37 Predictors (10-fold CV)  0.04147
## 5  Backward Select, 29 Predictors (test/train)  0.04563
## 6  Backward Select, 37 Predictors (10-fold CV)  0.04140
## 7      Best Subset, 29 Predictors (test/train)  0.04563
## 8      Best Subset, 32 Predictors (10-fold CV)  0.04149
## 9            Ridge, 39 Predictors (test/train)  0.04613
## 10           Lasso, 32 Predictors (test/train)  0.04887
## 11            PCR, 37 Components, (test/train)  0.04620
## 12            PLSR, 4 Components, (test/train)  0.04569
## 13                                    Mean TSS  0.97627

cv.err.results$forward

##  [1] 0.41613643 0.28343031 0.21874538 0.17428366 0.14758522 0.12599088
##  [7] 0.10786133 0.09638614 0.08662898 0.08020920 0.07317202 0.06844981
## [13] 0.06353480 0.05964648 0.05667014 0.05389986 0.05124762 0.04962155
## [19] 0.04781248 0.04604182 0.04487193 0.04468439 0.04363800 0.04352358
## [25] 0.04253519 0.04278864 0.04247816 0.04189727 0.04195096 0.04214403
```

```
## [31] 0.04161402 0.04192341 0.04179765 0.04200031 0.04175911 0.04188367
## [37] 0.04147359 0.04211854 0.04196733
```

Now Without Outliers

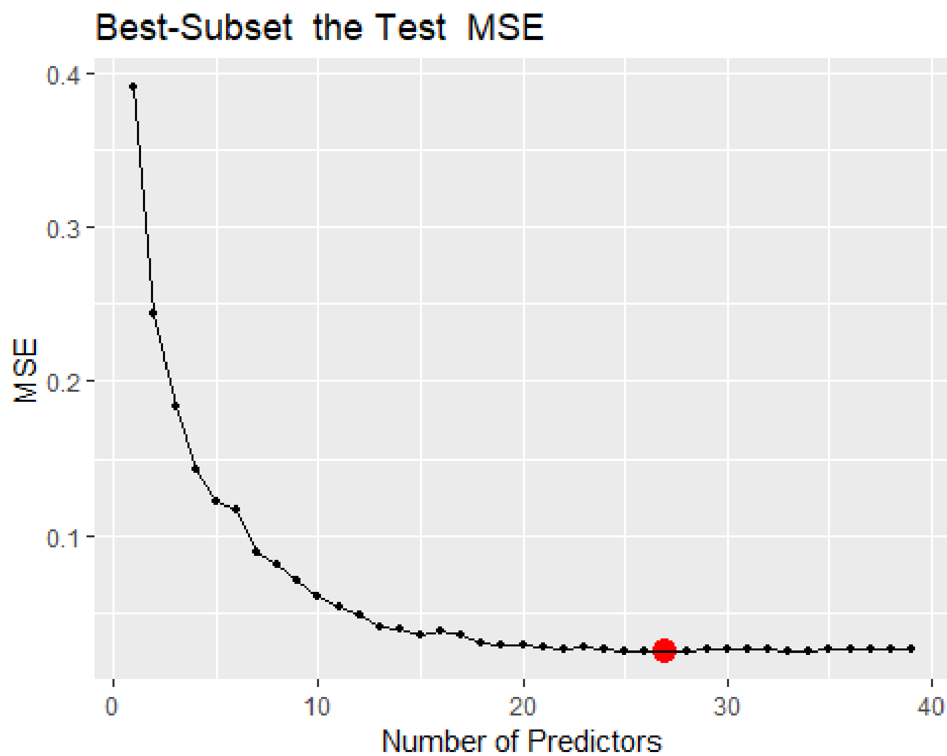
Full-Model

```
hfi.combined.no.outliers.list = generate.train.test(hfi.combined.no.outlier)
lm.fit.out = lm(hf_score ~ . -year, data =
hfi.combined.no.outliers.list[["train"]])
lm.preds = predict(lm.fit.out, newdata =
hfi.combined.no.outliers.list[["test"]])
# get MSE and R^2 (just 1 - MSE/MEAN(TSS) === 1 - RSS/TSS)
mse.lm.no.outlier = mean((lm.preds -
hfi.combined.no.outliers.list[["test"]]$hf_score)^2)
lm.r2 = 1 - (mse.lm.no.outlier/hfi.combined.no.outliers.list[["tss.test"]])
mse.lm.no.outlier

## [1] 0.02620537

lm.r2

## [1] 0.9703953
```

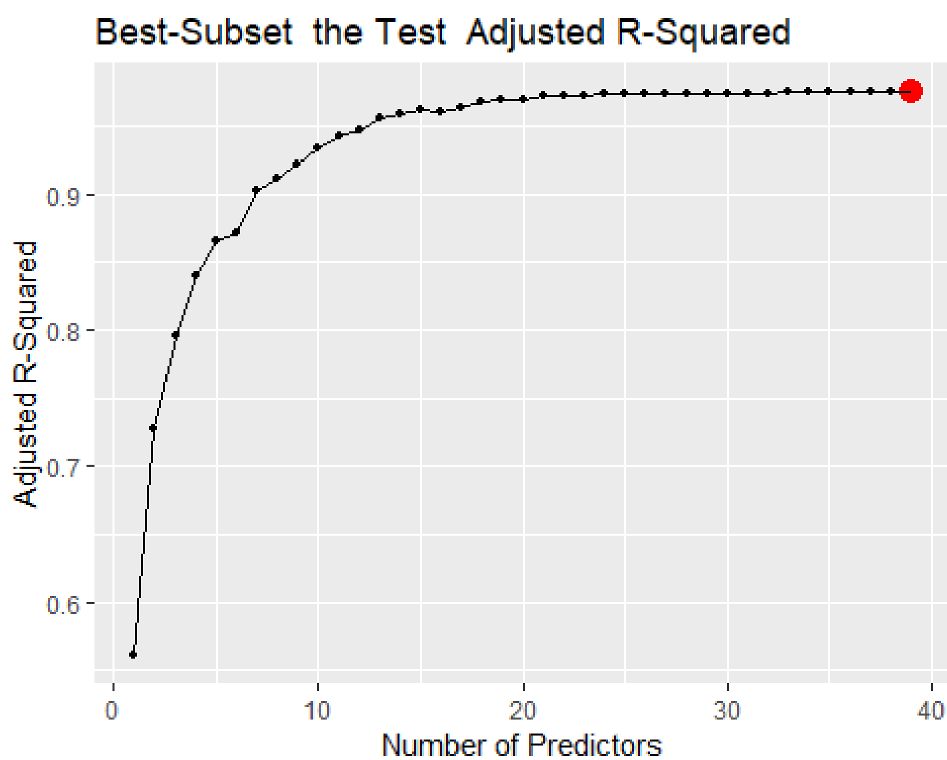
Best-Subset, Forward Selection, Backwards Selection

Best-Subset

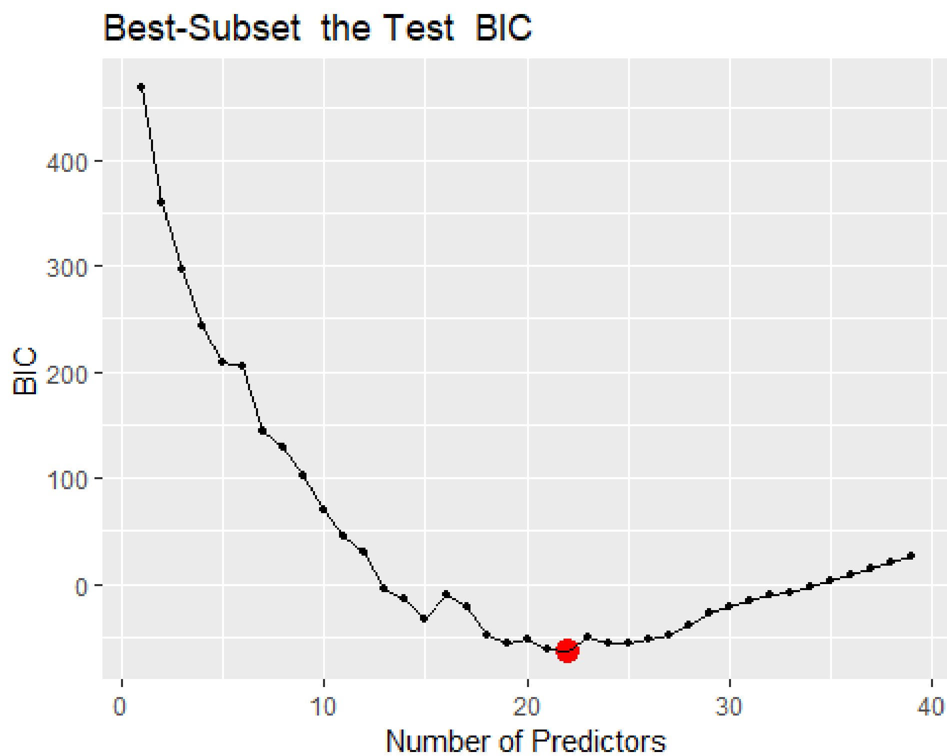
```
# store the results of the best-subset regressions
max.variables.to.run = ncol(hfi.combined.no.outliers.list$train) - 2
best.subset.no.outlier.result =
get.best.subset(hfi.combined.no.outliers.list$train,

hfi.combined.no.outliers.list$test,

hfi.combined.no.outliers.list$tss.test,
                                                max.variables.to.run)

print.out.graphs.lm(best.subset.no.outlier.result,
                    harsh.penalty = harsh.penalty,
                    title.name = "Best-Subset")
```



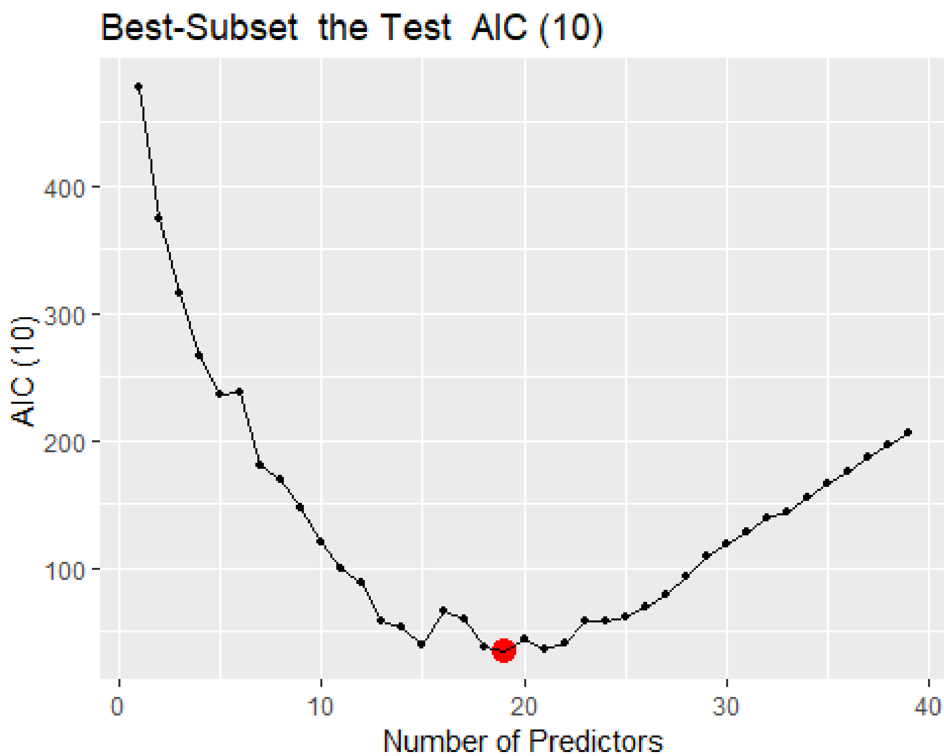

```
## Current Metric: Test  MSE
## Minimum Value:  0.02538499
##  Number of Predictors:  27
```



```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value:  39
##  Number of Predictors:  0.9752061
```



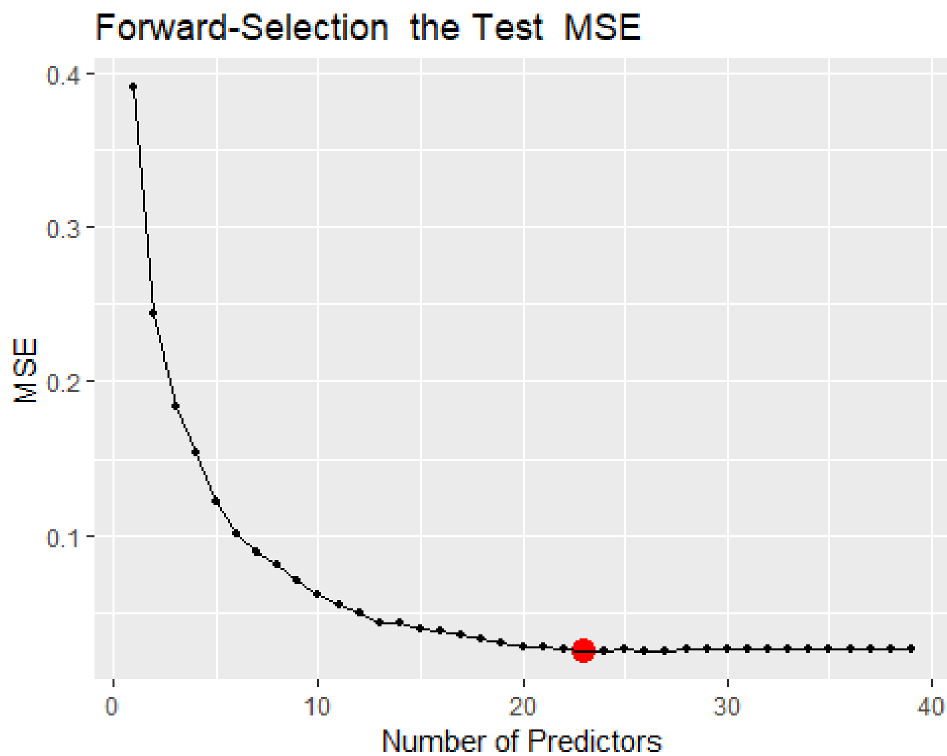

```
## Current Metric: Test  BIC
## Minimum Value:  -63.62389
##  Number of Predictors:  22
```



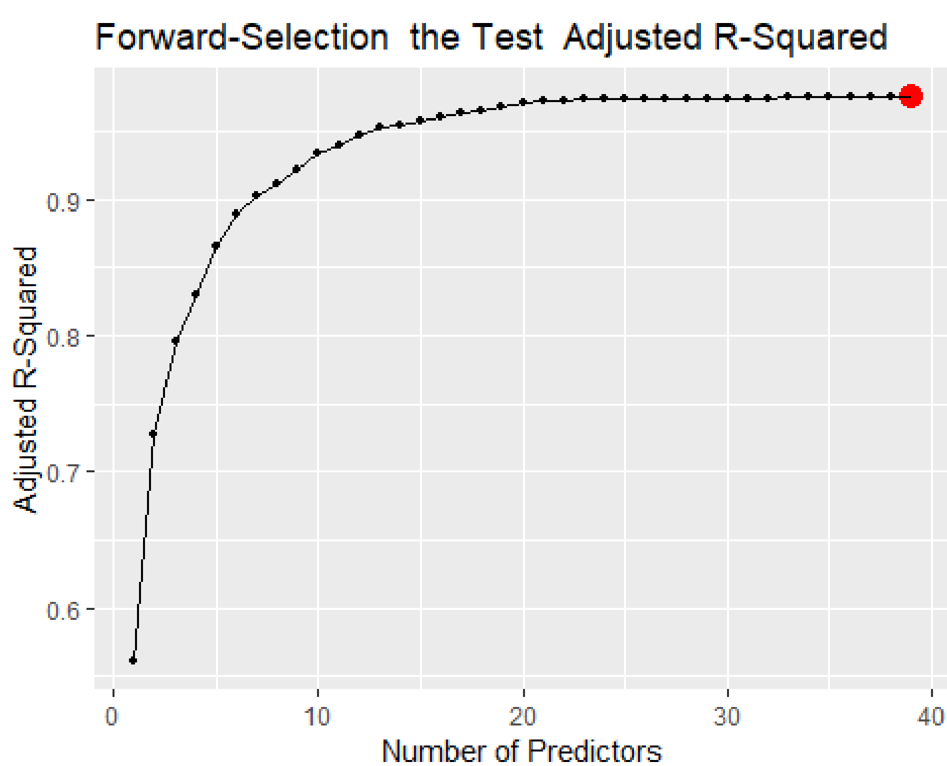

```
## Current Metric: Test  AIC (10)
## Minimum Value:  34.54663
##  Number of Predictors:  19
```

Forward-Selection

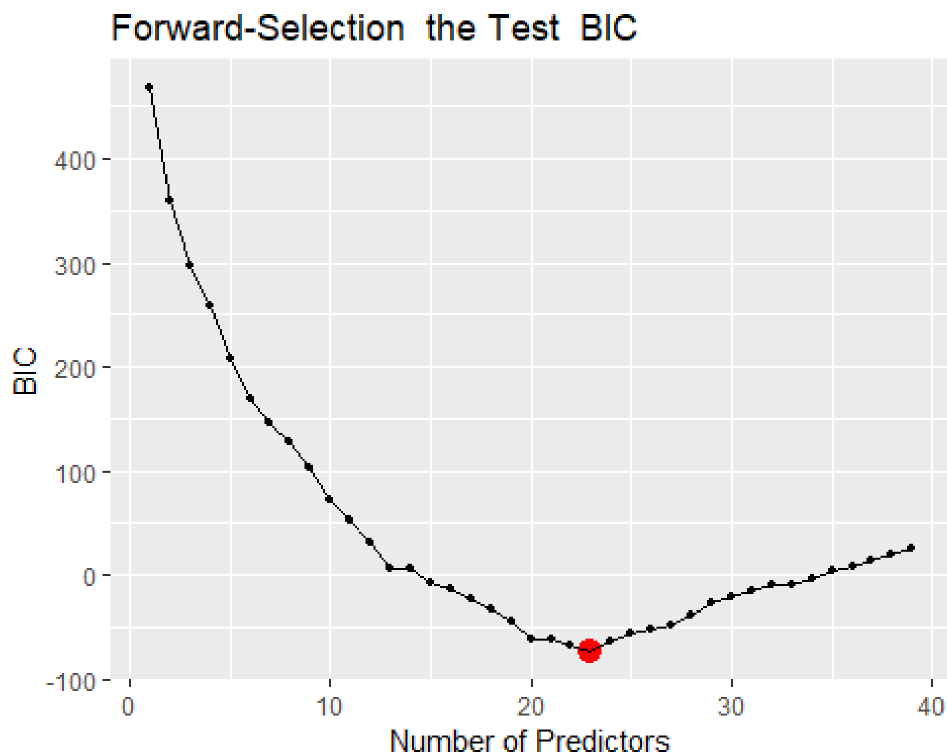
```
forward.no.outlier.result =
do.selection.methods(hfi.combined.no.outliers.list$train,

hfi.combined.no.outliers.list$test,

hfi.combined.no.outliers.list$tss.test,

ncol(hfi.combined.no.outliers.list$test)-2,
                                                  type = "forward")

print.out.graphs.lm(forward.no.outlier.result,
                    harsh.penalty = harsh.penalty,
                    title.name = "Forward-Selection")
```



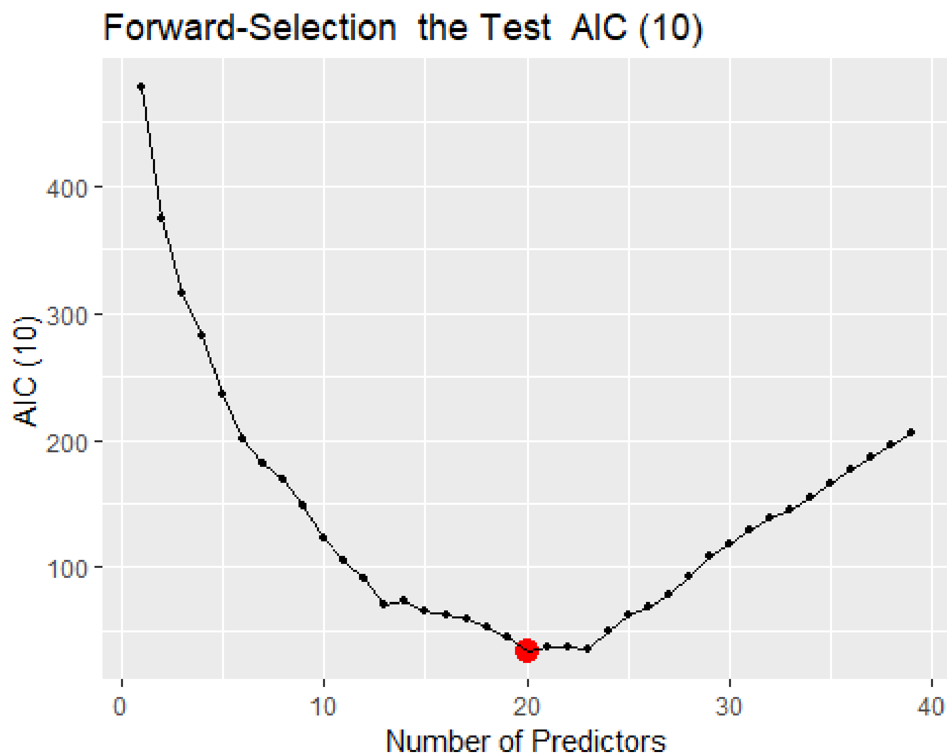

```
## Current Metric: Test  MSE
## Minimum Value:  0.02501721
##  Number of Predictors:  23
```



```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value:
##  Number of Predictors:
```



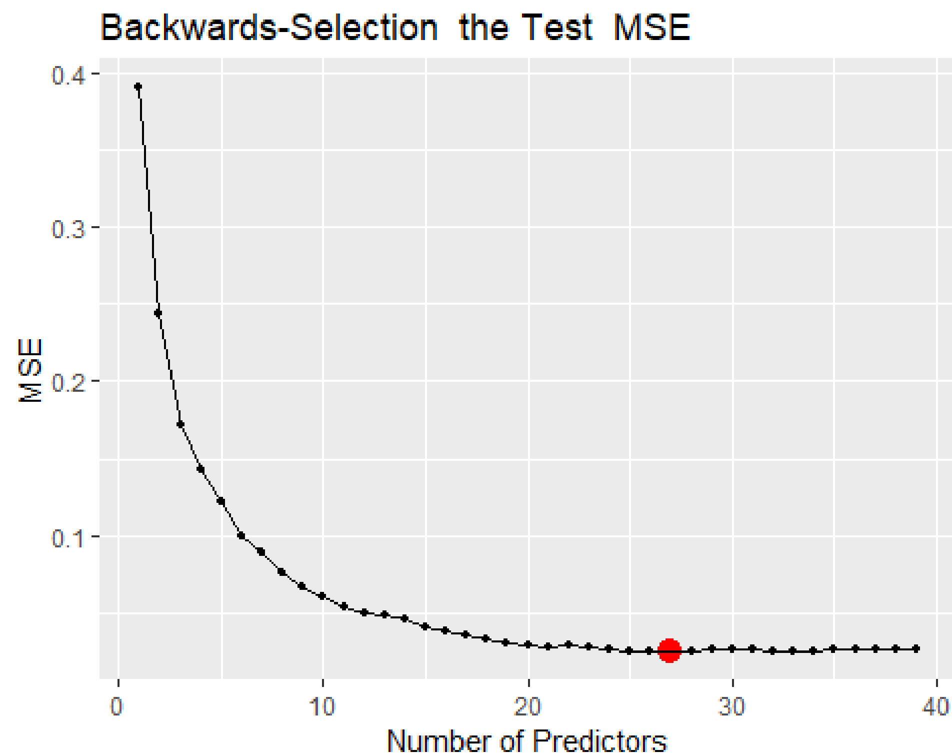

```
## Current Metric: Test  BIC
## Minimum Value:  -73.29063
##  Number of Predictors:  23
```



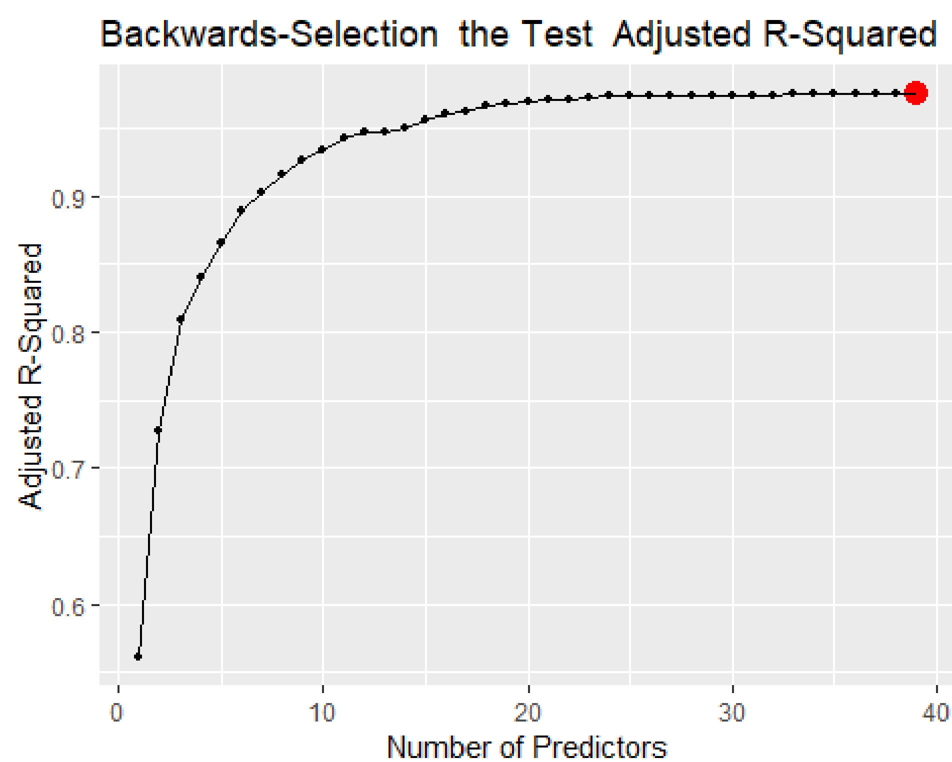

```
## Current Metric: Test  AIC (10)
## Minimum Value:  33.96825
##  Number of Predictors:  20
```

Backward Selection

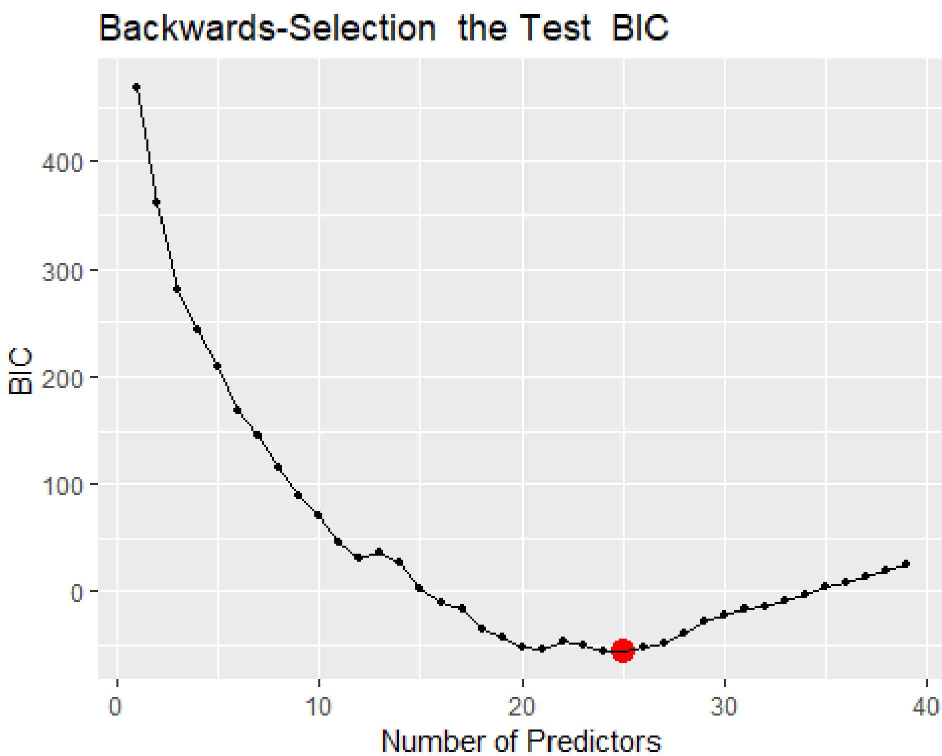
```
backwards.no.outlier.result =
do.selection.methods(hfi.combined.no.outliers.list$train,

hfi.combined.no.outliers.list$test,

hfi.combined.no.outliers.list$tss.test,

ncol(hfi.combined.no.outliers.list$test)-2,
                                                   type = "backwards")

print.out.graphs.lm(backwards.no.outlier.result,
                    harsh.penalty = harsh.penalty,
                    title.name = "Backwards-Selection")
```



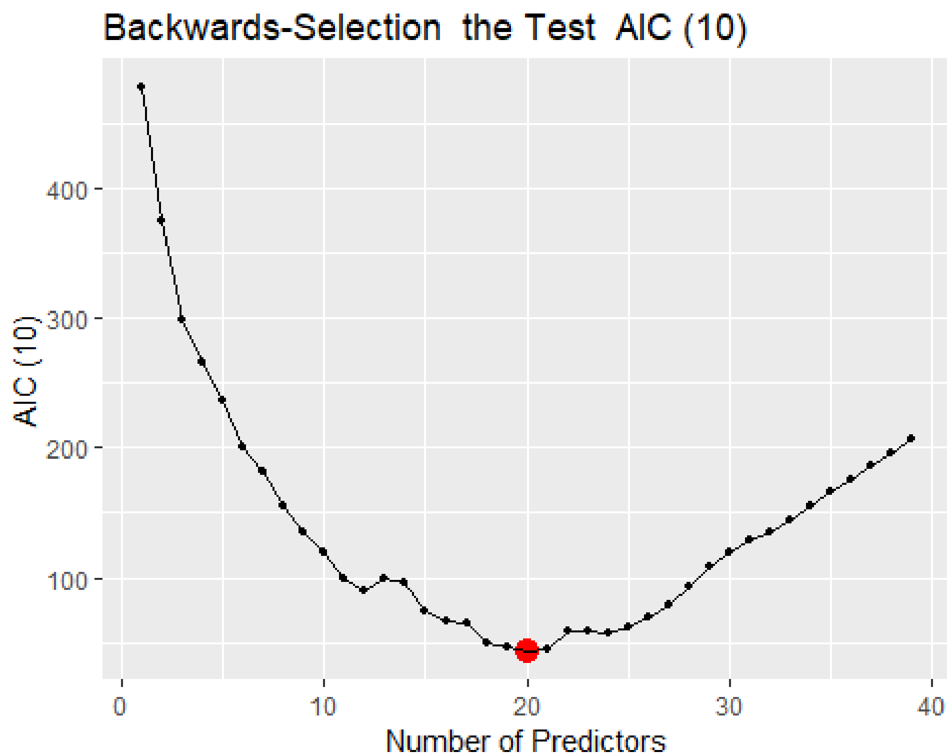

```
## Current Metric: Test  MSE
## Minimum Value:  0.02538499
##  Number of Predictors:  27
```



```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value: 
##  Number of Predictors: 
```




```
## Current Metric: Test  BIC
## Minimum Value:  -55.48842
##  Number of Predictors:  25
```

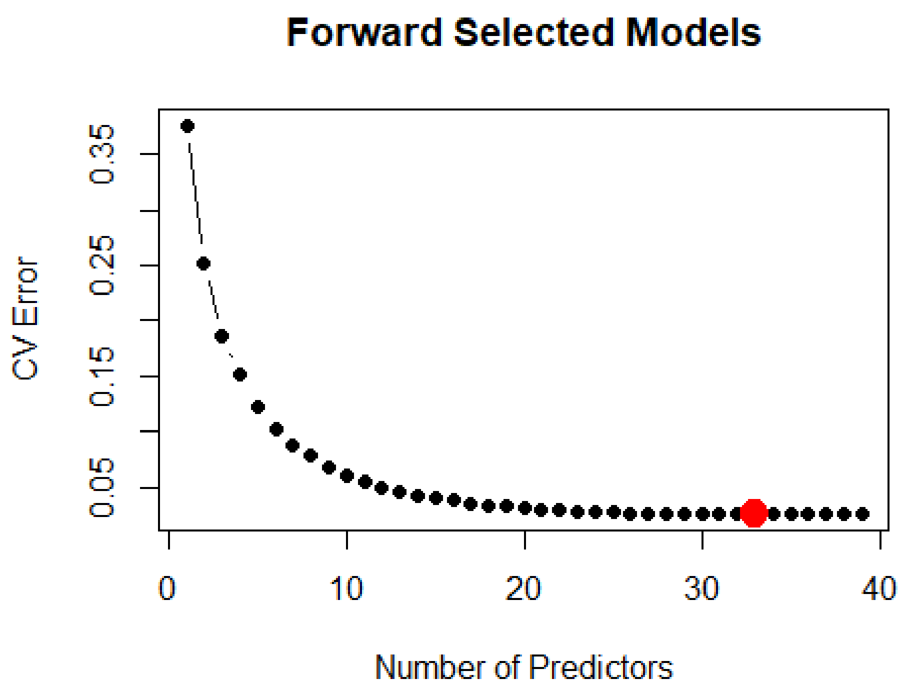



```
## Current Metric: Test  AIC (10)
## Minimum Value:  43.35234
##  Number of Predictors:  20
```

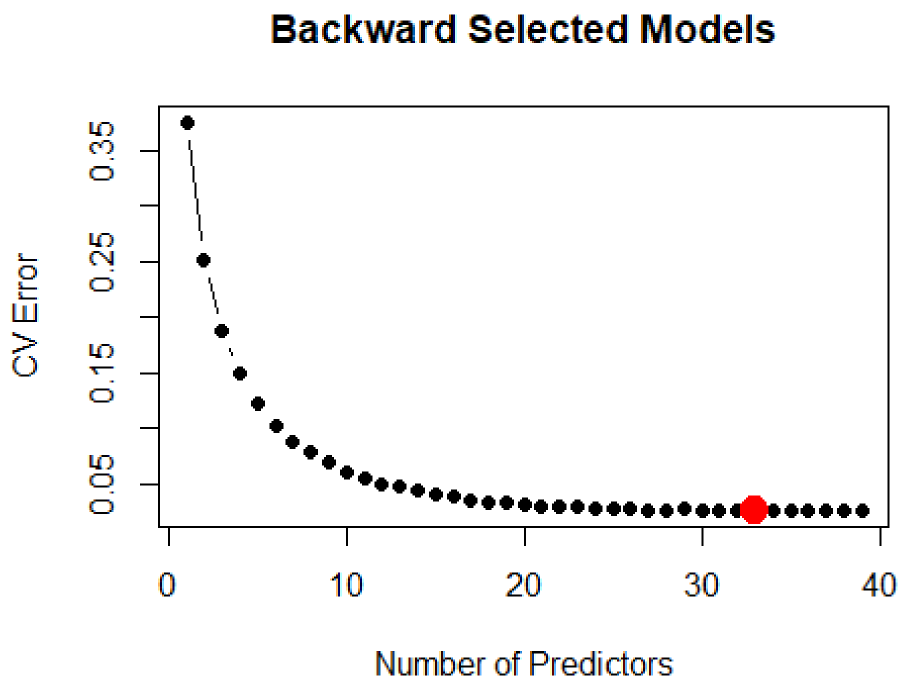
Get the CV Errors

```
dataset.hfi = hfi.combined.no.outliers.list$train %>%
full_join(hfi.combined.no.outliers.list$test)

cv.err.no.outlier.results = list()
cv.err.no.outlier.results[["forward"]] = get.cv.errors(data.hfi =
dataset.hfi,
                                                       regression.results =
forward.no.outlier.result)
cv.err.no.outlier.results[["backward"]] = get.cv.errors(data.hfi =
dataset.hfi,
                                                        regression.results =
backwards.no.outlier.result)
cv.err.no.outlier.results[["best.subset"]] = get.cv.errors(data.hfi =
dataset.hfi,
                                                           regression.results
= best.subset.no.outlier.result)

cv.full.lm = cv.glm(data = dataset.hfi, K = 10, glmfit = glm(formula =
hf_score ~ . - year, data = dataset.hfi))

cv.full.lm.err = cv.full.lm$delta[1]
```

```
cv.err.no.outlier.results[["full.lm"]] = cv.full.lm.err


plot.cv.errors(cv.err.no.outlier.results$forward, title.string = "Forward
Selected Models")
```

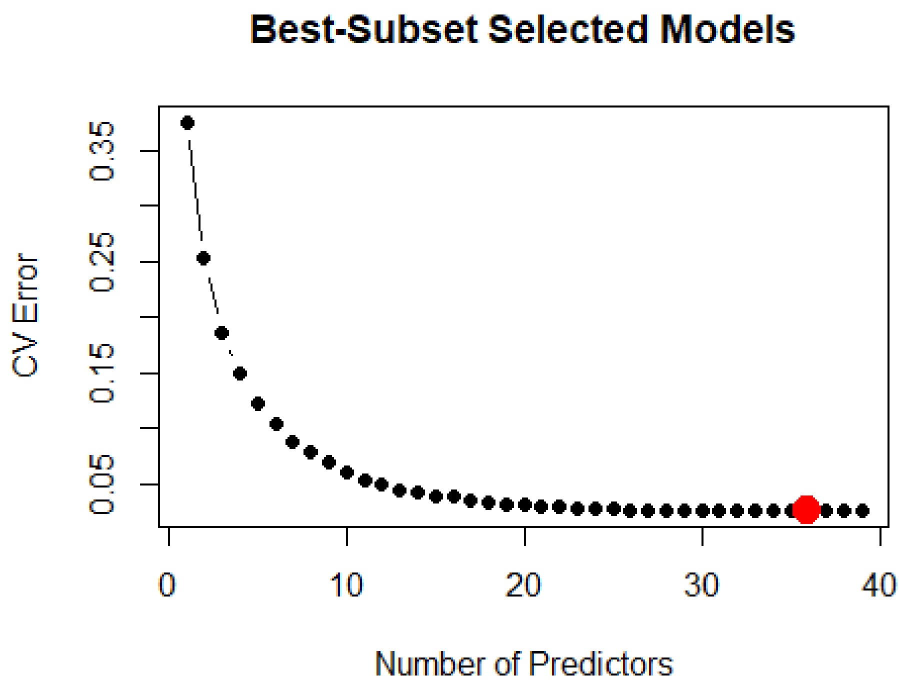



```
## NULL

plot.cv.errors(cv.err.no.outlier.results$backward, title.string = "Backward
Selected Models")
```




```
## NULL
```

```r
plot.cv.errors(cv.err.no.outlier.results$best.subset, title.string = "Best-Subset Selected Models")
```




```
## NULL

# These are the errors from the best subset models with the lowest errors:
which.min(cv.err.no.outlier.results$forward)

## [1] 33

cv.err.no.outlier.results$forward[which.min(cv.err.no.outlier.results$forward
)]

## [1] 0.02632879

which.min(cv.err.no.outlier.results$best.subset)

## [1] 36

cv.err.no.outlier.results$best.subset[which.min(cv.err.no.outlier.results$bes
t.subset)]

## [1] 0.02627389

# total preds (-1 because 1 is year that is unused)
ncol(hfi.features)-1

## [1] 39

cv.err.no.outlier.results$full.lm

## [1] 0.02671547
```

```r
hfi.combined.no.outliers.list$tss
```

```
## [1] 0.9224155
```

Coefs of Forward

```r
coef(forward.no.outlier.result$model,
which.min(cv.err.no.outlier.results$forward))
```

```
##                         (Intercept)                      pf_ss_homicide
##                         0.132484852                         0.053726909
##         pf_ss_disappearances_disap       pf_ss_disappearances_violent
##                         0.010173250                         0.009243880
##   pf_ss_disappearances_fatalities     pf_ss_disappearances_injuries
##                         0.030948193                        -0.029348381
##               pf_movement_domestic                 pf_movement_foreign
##                         0.018865162                         0.017372246
##             pf_religion_harassment            pf_religion_restrictions
##                         0.033772477                         0.020681099
##              pf_expression_killed            pf_expression_influence
##                         0.004809408                         0.042648188
##             pf_expression_control                pf_identity_sex_male
##                         0.055611640                         0.021928267
##             pf_identity_sex_female          ef_government_consumption
##                         0.019606496                         0.044064478
##                    ef_legal_courts                   ef_legal_military
##                         0.073288135                         0.040472756
##               ef_legal_enforcement                     ef_legal_gender
##                         0.046376944                         1.028001181
##                    ef_money_growth                         ef_money_sd
##                         0.035230944                         0.039633795
##                 ef_money_inflation                   ef_money_currency
##                         0.031585391                         0.026881822
##             ef_trade_tariffs_mean                ef_trade_regulatory
##                         0.052795427                         0.067216830
##          ef_trade_movement_capital            ef_trade_movement_visit
##                         0.015510109                         0.011989411
##       ef_regulation_credit_private                ef_regulation_credit
##                         0.010475319                         0.031138321
##        ef_regulation_labor_minwage          ef_regulation_labor_hours
##                         0.006962059                         0.011786108
## ef_regulation_labor_conscription      ef_regulation_business_start
##                         0.002328449                         0.021994039
```

Coefs of Best-Subset

```r
coef(best.subset.no.outlier.result$model,
which.min(cv.err.no.outlier.results$best.subset))
```

```
##                         (Intercept)                      pf_ss_homicide
##                         0.069793855                         0.053276719
##         pf_ss_disappearances_disap       pf_ss_disappearances_violent
```

```
##                             0.010778803                             0.009531775
##   pf_ss_disappearances_fatalities     pf_ss_disappearances_injuries
##                             0.031376690                            -0.030669801
##                  pf_movement_domestic                pf_movement_foreign
##                             0.018976611                             0.017353659
##                pf_religion_harassment           pf_religion_restrictions
##                             0.033322328                             0.020354234
##                  pf_expression_killed              pf_expression_jailed
##                             0.004673759                             0.003920378
##               pf_expression_influence             pf_expression_control
##                             0.041691890                             0.055950956
##                  pf_identity_sex_male            pf_identity_sex_female
##                             0.022164464                             0.019485362
##            ef_government_consumption                   ef_legal_courts
##                             0.043395818                             0.072436714
##                     ef_legal_military              ef_legal_enforcement
##                             0.039542440                             0.045802728
##                       ef_legal_gender                   ef_money_growth
##                             1.022228800                             0.034388588
##                           ef_money_sd                 ef_money_inflation
##                             0.039074326                             0.028127336
##                     ef_money_currency             ef_trade_tariffs_mean
##                             0.027131235                             0.057025671
##                   ef_trade_tariffs_sd               ef_trade_regulatory
##                            -0.004581583                             0.067126373
##                        ef_trade_black         ef_trade_movement_capital
##                             0.009700011                             0.015835453
##              ef_trade_movement_visit      ef_regulation_credit_private
##                             0.011519846                             0.009446584
##                  ef_regulation_credit       ef_regulation_labor_minwage
##                             0.032539351                             0.006530700
##           ef_regulation_labor_hours ef_regulation_labor_conscription
##                             0.011769401                             0.002112424
##         ef_regulation_business_start
##                             0.022030700
```

Coefs of Backwards

```
coef(backwards.no.outlier.result$model,
which.min(cv.err.no.outlier.results$backward))
```

```
##                           (Intercept)                     pf_ss_homicide
##                           0.099851494                        0.053909071
##         pf_ss_disappearances_disap     pf_ss_disappearances_violent
##                           0.010124477                        0.010053208
## pf_ss_disappearances_fatalities    pf_ss_disappearances_injuries
##                           0.030059097                       -0.029269220
##                pf_movement_domestic               pf_movement_foreign
##                           0.019347414                        0.017589403
##              pf_religion_harassment          pf_religion_restrictions
##                           0.034205050                        0.020598283
```

```
##              pf_expression_killed          pf_expression_influence
##                       0.004616189                      0.043647136
##             pf_expression_control             pf_identity_sex_male
##                       0.054619518                      0.021699048
##            pf_identity_sex_female       ef_government_consumption
##                       0.019666906                      0.043211739
##                   ef_legal_courts                ef_legal_military
##                       0.073277760                      0.040977960
##              ef_legal_enforcement                  ef_legal_gender
##                       0.044674778                      1.029485923
##                   ef_money_growth                      ef_money_sd
##                       0.034116943                      0.038445018
##                ef_money_inflation                ef_money_currency
##                       0.027874930                      0.026942606
##             ef_trade_tariffs_mean              ef_trade_regulatory
##                       0.051978996                      0.067517866
##                    ef_trade_black        ef_trade_movement_capital
##                       0.010978777                      0.015744157
##           ef_trade_movement_visit      ef_regulation_credit_private
##                       0.012527472                      0.009607697
##              ef_regulation_credit       ef_regulation_labor_minwage
##                       0.031537747                      0.007078404
##         ef_regulation_labor_hours     ef_regulation_business_start
##                       0.012070959                      0.021473332
```

Ridge and Lasso

Ridge

```
ridge.no.outlier = do.ridge.lasso(hfi.combined.no.outliers.list$train,
                                  hfi.combined.no.outliers.list$test,
                                  alpha=0)

ridge.no.outlier$model$beta
```

```
## 39 x 1 sparse Matrix of class "dgCMatrix"
##                                           s0
## pf_ss_homicide                   0.051764811
## pf_ss_disappearances_disap       0.010807319
## pf_ss_disappearances_violent     0.008627004
## pf_ss_disappearances_fatalities  0.029721701
## pf_ss_disappearances_injuries   -0.027852385
## pf_movement_domestic             0.018883478
## pf_movement_foreign              0.017405784
## pf_religion_harassment           0.032394392
## pf_religion_restrictions         0.021086525
## pf_expression_killed             0.005165011
## pf_expression_jailed             0.003872554
## pf_expression_influence          0.041153485
```

```
## pf_expression_control                0.054800491
## pf_identity_sex_male                 0.021417895
## pf_identity_sex_female               0.020007372
## ef_government_consumption            0.040970765
## ef_legal_courts                      0.071785263
## ef_legal_military                    0.039370798
## ef_legal_enforcement                 0.045621137
## ef_legal_gender                      1.019279330
## ef_money_growth                      0.033573089
## ef_money_sd                          0.040788920
## ef_money_inflation                   0.026964968
## ef_money_currency                    0.026430612
## ef_trade_tariffs_mean                0.050234194
## ef_trade_tariffs_sd                 -0.007546875
## ef_trade_tariffs                     0.012332307
## ef_trade_regulatory_compliance       0.007588157
## ef_trade_regulatory                  0.055214886
## ef_trade_black                       0.010528739
## ef_trade_movement_capital            0.016326658
## ef_trade_movement_visit              0.011224284
## ef_regulation_credit_private         0.009456953
## ef_regulation_credit                 0.032429985
## ef_regulation_labor_minwage          0.006990044
## ef_regulation_labor_hours            0.012118869
## ef_regulation_labor_conscription     0.002265317
## ef_regulation_business_start         0.022458575
## ef_regulation_business_compliance    0.001625332
```

```
ridge.no.outlier$best.lambda
```

```
## [1] 0.01056876
```

```
ridge.no.outlier$mse
```

```
## [1] 0.02624579
```

Lasso

```
lasso.no.outlier = do.ridge.lasso(hfi.combined.no.outliers.list$train,
                                  hfi.combined.no.outliers.list$test,
                                  alpha=1)

rownames.no.na = c()
for (i in 1:length(rownames(lasso.no.outlier$model$beta))){
  if( lasso.no.outlier$model$beta[i] != 0){
    rownames.no.na = c(rownames.no.na,
rownames(lasso.no.outlier$model$beta)[i])
  }
}
lasso.no.outlier[["rownames.left"]] = rownames.no.na
```

```
lasso.no.outlier$model$beta
```

```
## 39 x 1 sparse Matrix of class "dgCMatrix"
##                                              s0
## pf_ss_homicide                     0.0492338681
## pf_ss_disappearances_disap         0.0095709352
## pf_ss_disappearances_violent       0.0011095415
## pf_ss_disappearances_fatalities    0.0126983496
## pf_ss_disappearances_injuries      .
## pf_movement_domestic               0.0178605809
## pf_movement_foreign                0.0168774450
## pf_religion_harassment             0.0223433206
## pf_religion_restrictions           0.0182914544
## pf_expression_killed               0.0001634535
## pf_expression_jailed               .
## pf_expression_influence            0.0402447780
## pf_expression_control              0.0560962081
## pf_identity_sex_male               0.0205159749
## pf_identity_sex_female             0.0191365505
## ef_government_consumption          0.0291913378
## ef_legal_courts                    0.0682650044
## ef_legal_military                  0.0383814095
## ef_legal_enforcement               0.0433732011
## ef_legal_gender                    1.0318353923
## ef_money_growth                    0.0264117970
## ef_money_sd                        0.0430217985
## ef_money_inflation                 0.0215352592
## ef_money_currency                  0.0263215303
## ef_trade_tariffs_mean              0.0498416487
## ef_trade_tariffs_sd                .
## ef_trade_tariffs                   .
## ef_trade_regulatory_compliance     0.0010500719
## ef_trade_regulatory                0.0703818229
## ef_trade_black                     0.0141696755
## ef_trade_movement_capital          0.0157514424
## ef_trade_movement_visit            0.0102203745
## ef_regulation_credit_private       0.0014478714
## ef_regulation_credit               0.0415886036
## ef_regulation_labor_minwage        0.0025229380
## ef_regulation_labor_hours          0.0100653692
## ef_regulation_labor_conscription   0.0005773131
## ef_regulation_business_start       0.0172883072
## ef_regulation_business_compliance  0.0008091463
```

```
lasso.no.outlier$best.lambda
```

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

```
lasso.no.outlier$mse
```

```
## [1] 0.02625248
```

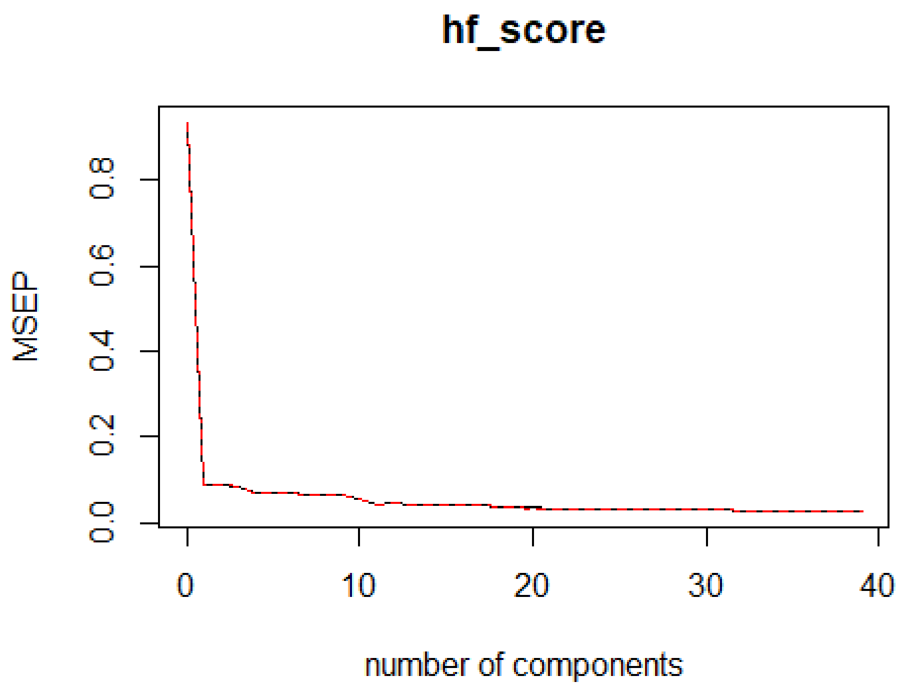
```r
lasso.no.outlier$rownames.left
```

```
##  [1] "pf_ss_homicide"
##  [2] "pf_ss_disappearances_disap"
##  [3] "pf_ss_disappearances_violent"
##  [4] "pf_ss_disappearances_fatalities"
##  [5] "pf_movement_domestic"
##  [6] "pf_movement_foreign"
##  [7] "pf_religion_harassment"
##  [8] "pf_religion_restrictions"
##  [9] "pf_expression_killed"
## [10] "pf_expression_influence"
## [11] "pf_expression_control"
## [12] "pf_identity_sex_male"
## [13] "pf_identity_sex_female"
## [14] "ef_government_consumption"
## [15] "ef_legal_courts"
## [16] "ef_legal_military"
## [17] "ef_legal_enforcement"
## [18] "ef_legal_gender"
## [19] "ef_money_growth"
## [20] "ef_money_sd"
## [21] "ef_money_inflation"
## [22] "ef_money_currency"
## [23] "ef_trade_tariffs_mean"
## [24] "ef_trade_regulatory_compliance"
## [25] "ef_trade_regulatory"
## [26] "ef_trade_black"
## [27] "ef_trade_movement_capital"
## [28] "ef_trade_movement_visit"
## [29] "ef_regulation_credit_private"
## [30] "ef_regulation_credit"
## [31] "ef_regulation_labor_minwage"
## [32] "ef_regulation_labor_hours"
## [33] "ef_regulation_labor_conscription"
## [34] "ef_regulation_business_start"
## [35] "ef_regulation_business_compliance"
```

PCR and PLSR

PCR

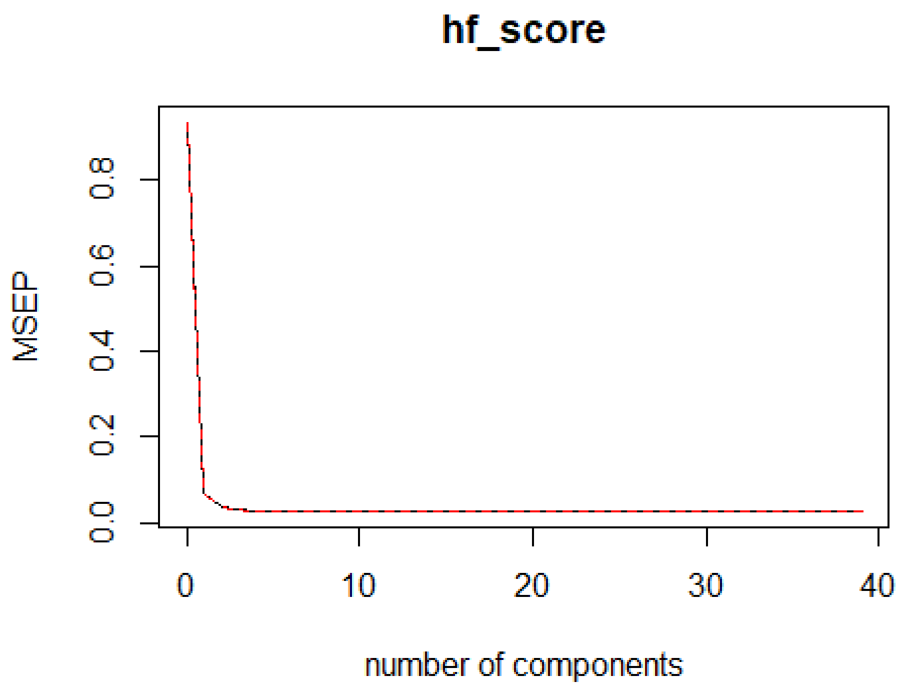
```r
pcr.model.no.outlier = pcr(hf_score ~ . -year , scale= TRUE,
                           data = hfi.combined.no.outliers.list$train,
                           validation = "CV")
print(validationplot(pcr.model.no.outlier, val.type = "MSEP"))
```




```
## NULL

pcr.no.outlier.best = find.best.mse.pcr.pls(pcr.model.no.outlier,
hfi.combined.no.outliers.list$test)
pcr.no.outlier.best$min.val

## [1] 37

pcr.no.outlier.best$best.mse

## [1] 0.02616796
```

PLSR

```
pls.no.outlier.model = plsr(hf_score ~ . -year, scale = TRUE,
                 data = hfi.combined.no.outliers.list$train,
                 validation = "CV")
validationplot(pls.no.outlier.model, val.type = "MSEP")
```




```
pls.no.outlier.best = find.best.mse.pcr.pls(pls.no.outlier.model,
                                            hfi.combined.no.outliers.list$test)
pls.no.outlier.best$min.val

## [1] 13

pls.no.outlier.best$best.mse

## [1] 0.02614839

x = matrix(data=
           c("Full OLS, 39 Predictors (test/train)",
             "Full OLS, 39 Predictors (10-fold CV)",
             paste0("Forward Select, ",
which.min(forward.no.outlier.result$forward.mse), " Predictors
(test/train)"),
             paste0("Forward Select, ",
which.min(cv.err.no.outlier.results$forward), " Predictors (10-fold CV)"),
             paste0("Backward Select, ",
which.min(backwards.no.outlier.result$forward.mse)," Predictors
(test/train)"),
             paste0("Backward Select, ",
which.min(cv.err.no.outlier.results$backward), " Predictors (10-fold CV)"),
             paste0("Best Subset, ",
which.min(best.subset.no.outlier.result$best.subset.mse)," Predictors
(test/train)"),
             paste0("Best Subset, ",
```

```r
which.min(cv.err.no.outlier.results$best.subset), " Predictors (10-fold CV)"),
             paste0("Ridge, ", length(ridge.no.outlier$model$beta), " Predictors (test/train)"),
             paste0("Lasso, ", length(lasso.no.outlier$rownames.left), " Predictors (test/train)"),
             paste0("PCR, ", pcr.no.outlier.best$min.val ," Components, (test/train)"),
             paste0("PLSR, ", pls.no.outlier.best$min.val, " Components, (test/train)"),
             "Mean TSS",
mse.lm.no.outlier,
cv.err.no.outlier.results$full.lm,
min(forward.no.outlier.result$forward.mse), # fit 3
min(cv.err.no.outlier.results$forward),
min(backwards.no.outlier.result$forward.mse),
min(cv.err.no.outlier.results$backward),
min(best.subset.no.outlier.result$best.subset.mse),
min(cv.err.no.outlier.results$best.subset),
ridge.no.outlier$mse,
lasso.no.outlier$mse,
pcr.no.outlier.best$best.mse,
pls.no.outlier.best$best.mse,
hfi.combined.no.outliers.list$tss
), nrow=13, ncol=2)
colnames(x) <- c("Method","test MSE")

df.x.all.no.outlier = as.data.frame(x, row.names = list.x.axis)
df.x.all.no.outlier$`test MSE` = round(as.numeric(as.character(df.x.all.no.outlier[["test MSE"]])), 5)
df.x.all.no.outlier

##                                          Method test MSE
## 1         Full OLS, 39 Predictors (test/train)  0.02621
## 2         Full OLS, 39 Predictors (10-fold CV)  0.02672
## 3   Forward Select, 23 Predictors (test/train)  0.02502
## 4   Forward Select, 33 Predictors (10-fold CV)  0.02633
## 5  Backward Select, 27 Predictors (test/train)  0.02538
## 6  Backward Select, 33 Predictors (10-fold CV)  0.02616
## 7      Best Subset, 27 Predictors (test/train)  0.02538
## 8      Best Subset, 36 Predictors (10-fold CV)  0.02627
## 9            Ridge, 39 Predictors (test/train)  0.02625
## 10           Lasso, 35 Predictors (test/train)  0.02625
## 11            PCR, 37 Components, (test/train)  0.02617
## 12           PLSR, 13 Components, (test/train)  0.02615
## 13                                    Mean TSS  0.92242
```

NOW, 2008 vs 2016

THE DATASET SIZE IS SMALLER – KEEP IN MIND FOR AIC, BIC

split and filter by year (2008 vs 2016)

```
min.year = min(hfi.combined$year)
max.year = max(hfi.combined$year)
hfi.2008 = hfi.combined %>% filter(year == min.year)
hfi.2016 = hfi.combined %>% filter(year == max.year)
hfi.2008.list = generate.train.test(hfi.2008)
hfi.2016.list = generate.train.test(hfi.2016)
```

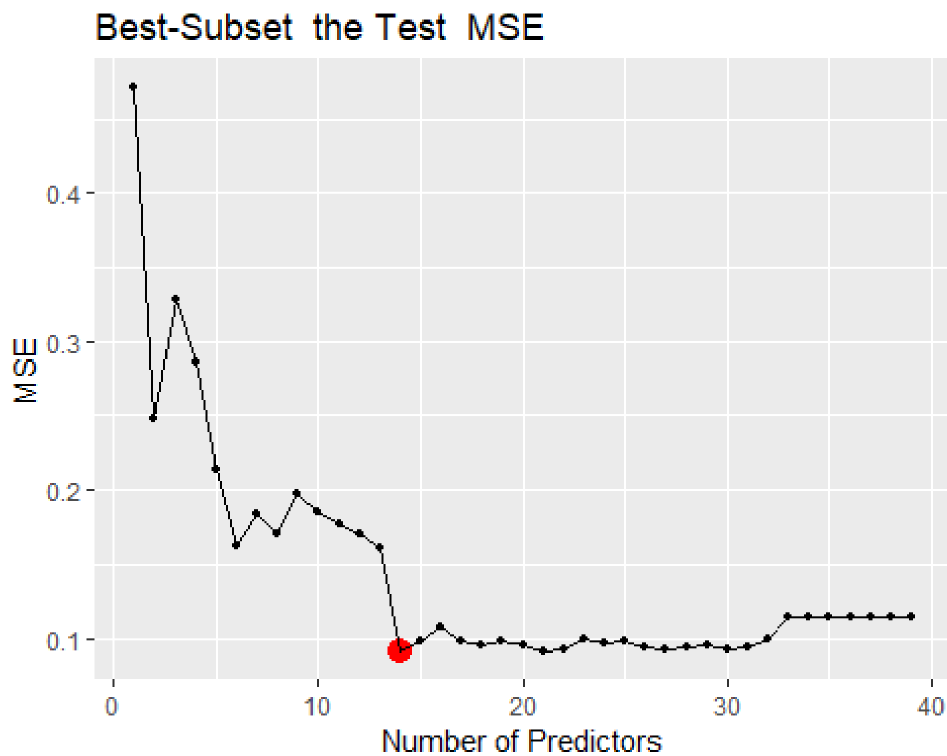
First do 2008

Full Model

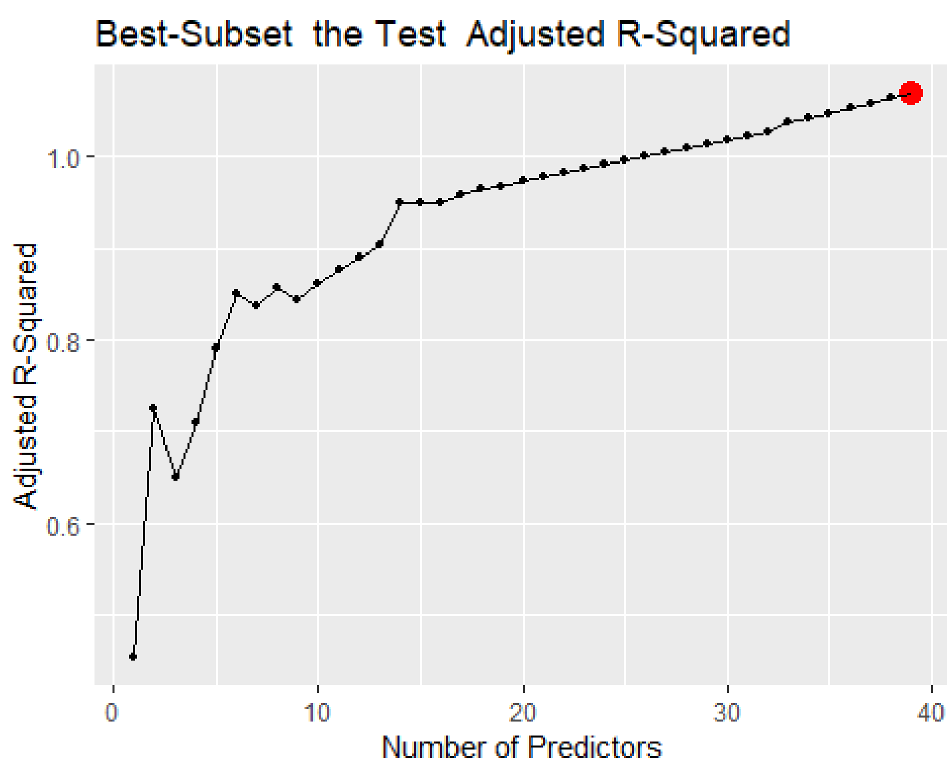
```
lm.fit.2008 = lm(hf_score ~ . -year, data = hfi.2008.list[["train"]])
lm.preds = predict(lm.fit.2008, newdata = hfi.2008.list[["test"]])
# get MSE and R^2 (just 1 - MSE/MEAN(TSS) === 1 - RSS/TSS)
mse.lm.2008 = mean((lm.preds - hfi.2008.list[["test"]]$hf_score)^2)
lm.r2 = 1 - (mse.lm.2008/hfi.2008.list[["tss.test"]])
mse.lm.2008

## [1] 0.1138852

lm.r2

## [1] 0.8627179
```

Best-Subset

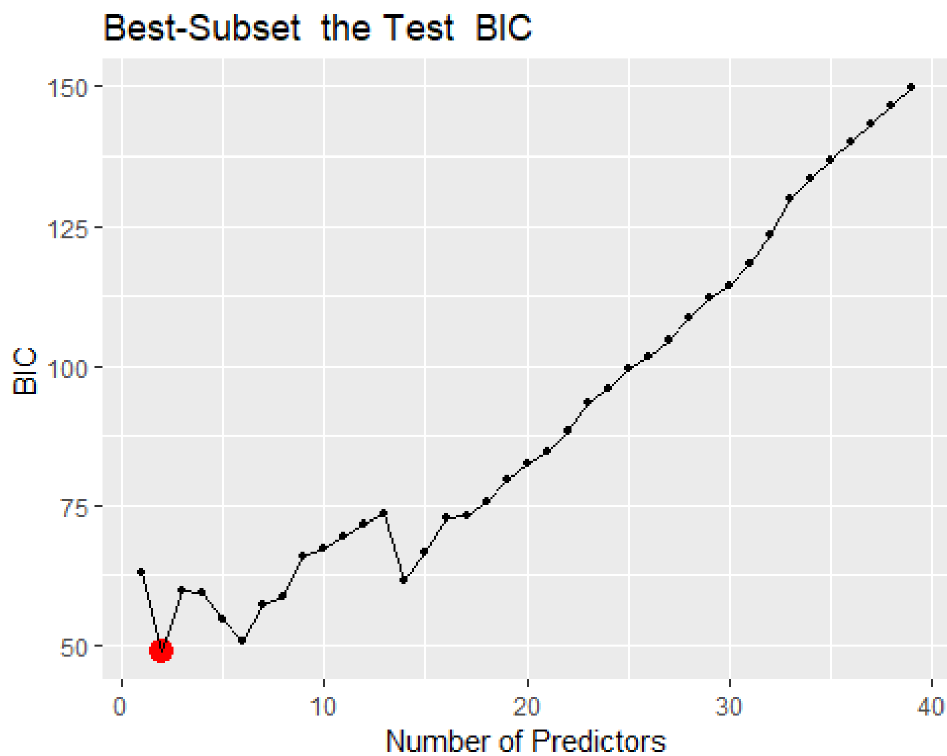
```
max.variables.to.run = ncol(hfi.2008.list$train) - 2
best.subset.2008.result = get.best.subset(hfi.2008.list$train,
hfi.2008.list$test,
                                     hfi.2008.list$tss.test,
                                     max.variables.to.run)

print.out.graphs.lm(best.subset.2008.result,
                    harsh.penalty = harsh.penalty,
                    title.name = "Best-Subset")
```



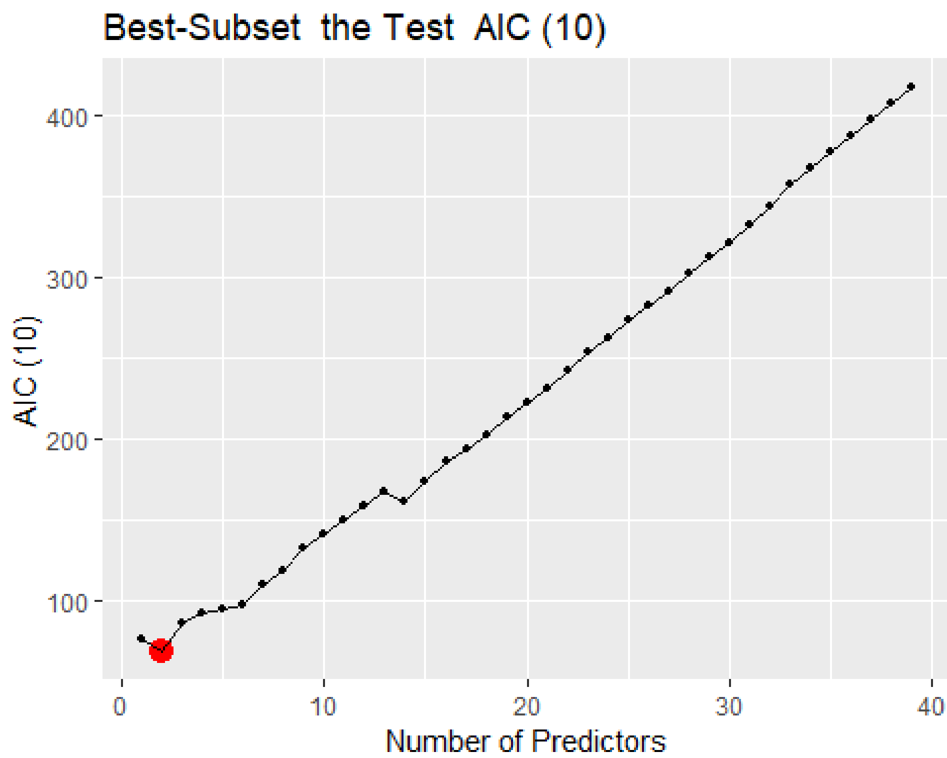

```
## Current Metric: Test  MSE
## Minimum Value:  0.09121692
##  Number of Predictors:  14
```



```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value:  39
##  Number of Predictors:  1.068641
```



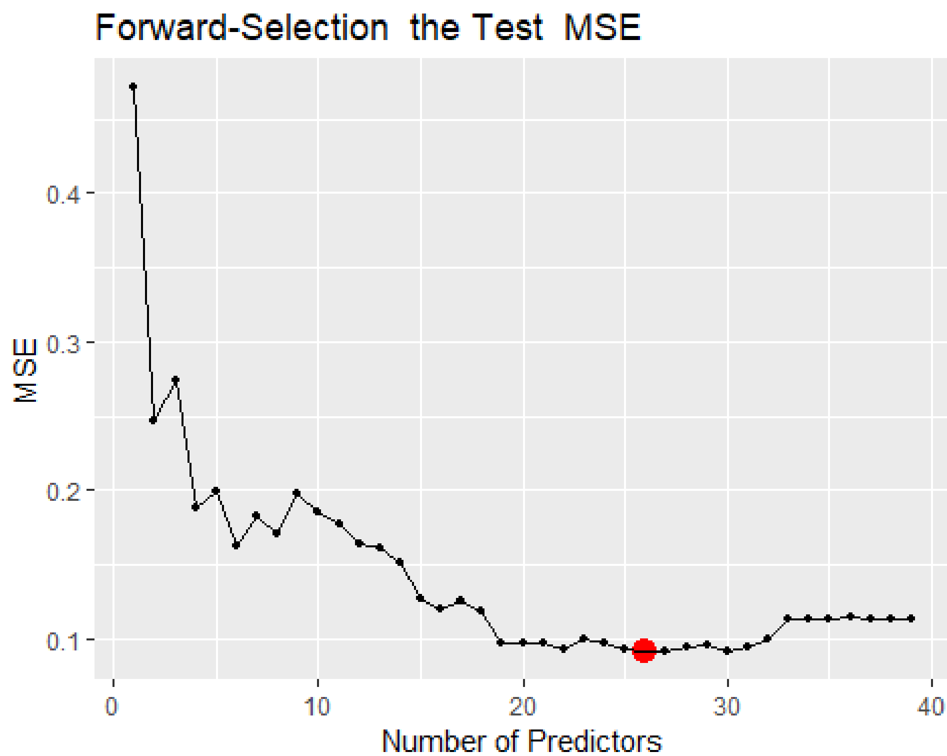

```
## Current Metric: Test  BIC
## Minimum Value:  48.79753
##  Number of Predictors:  2
```



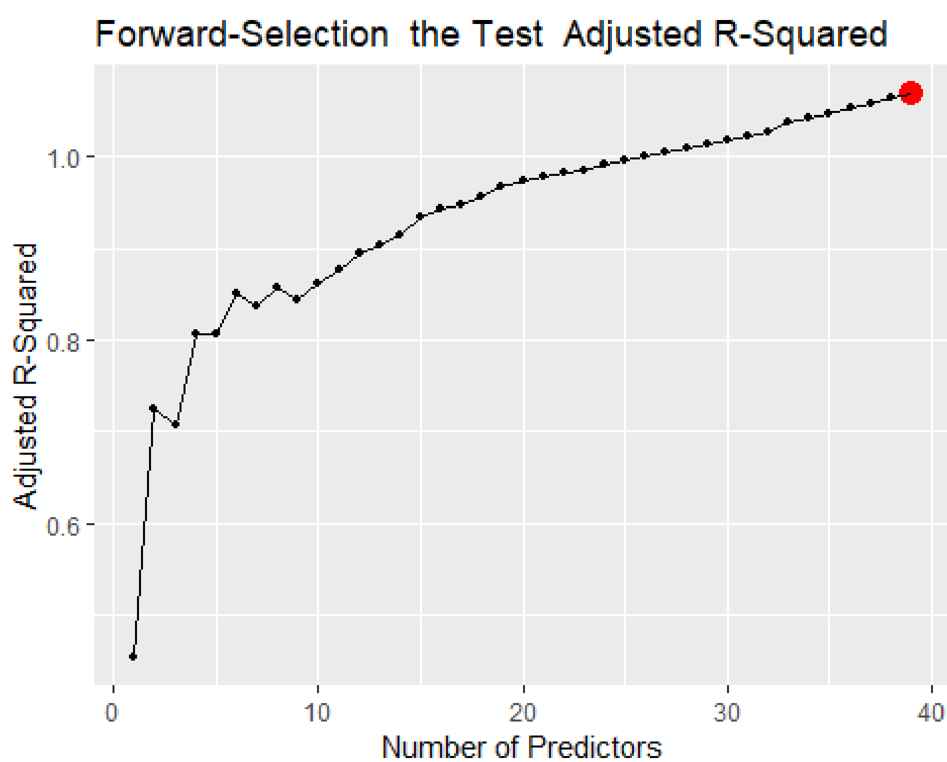

```
## Current Metric: Test  AIC (10)
## Minimum Value:  68.91002
##  Number of Predictors:  2
```

Forward Selection

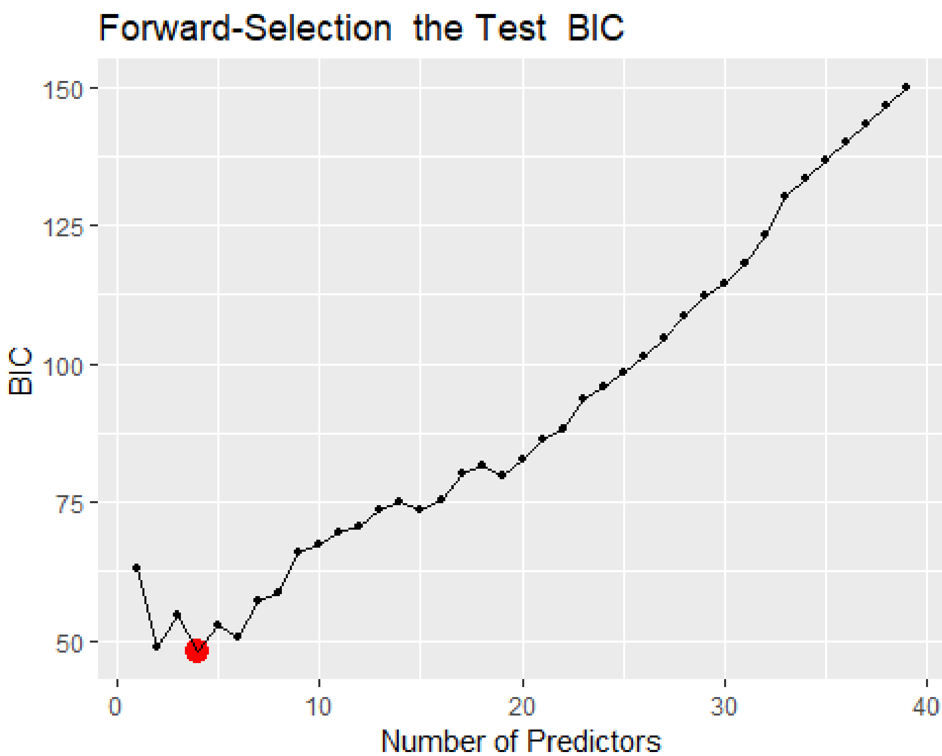
```
forward.2008.results = do.selection.methods(hfi.2008.list$train,
hfi.2008.list$test,
                                            hfi.2008.list$tss.test,
ncol(hfi.2008.list$test)-2,
                                            type = "forward")

print.out.graphs.lm(forward.2008.results,
                    harsh.penalty = harsh.penalty,
                    title.name = "Forward-Selection")
```



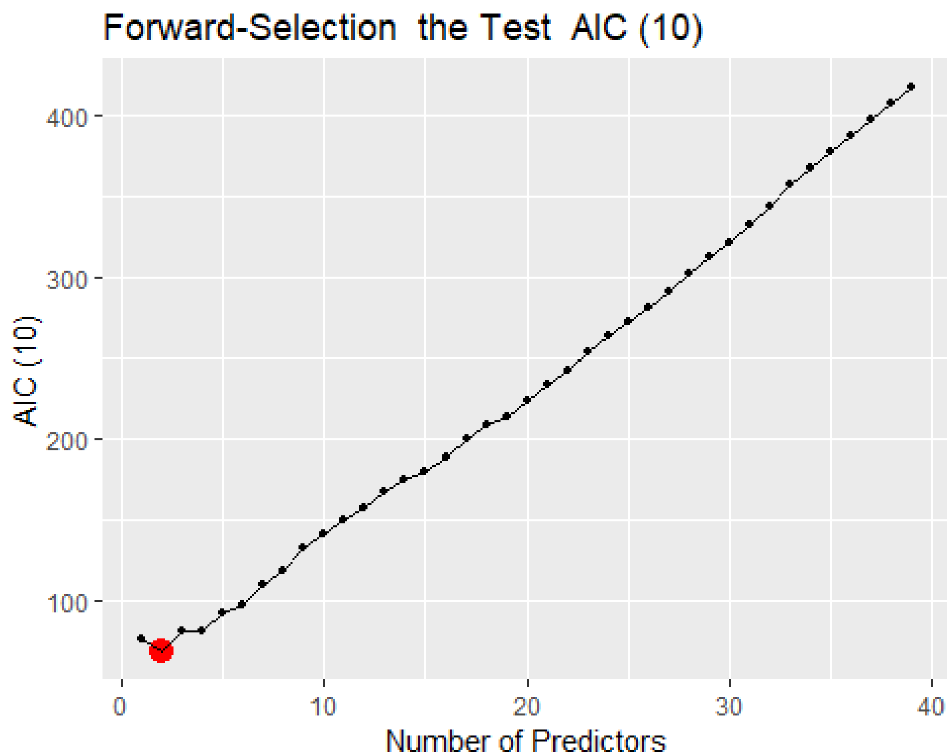

```
## Current Metric: Test  MSE
## Minimum Value:  0.09184443
##  Number of Predictors:  26
```



```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value:
##  Number of Predictors:
```



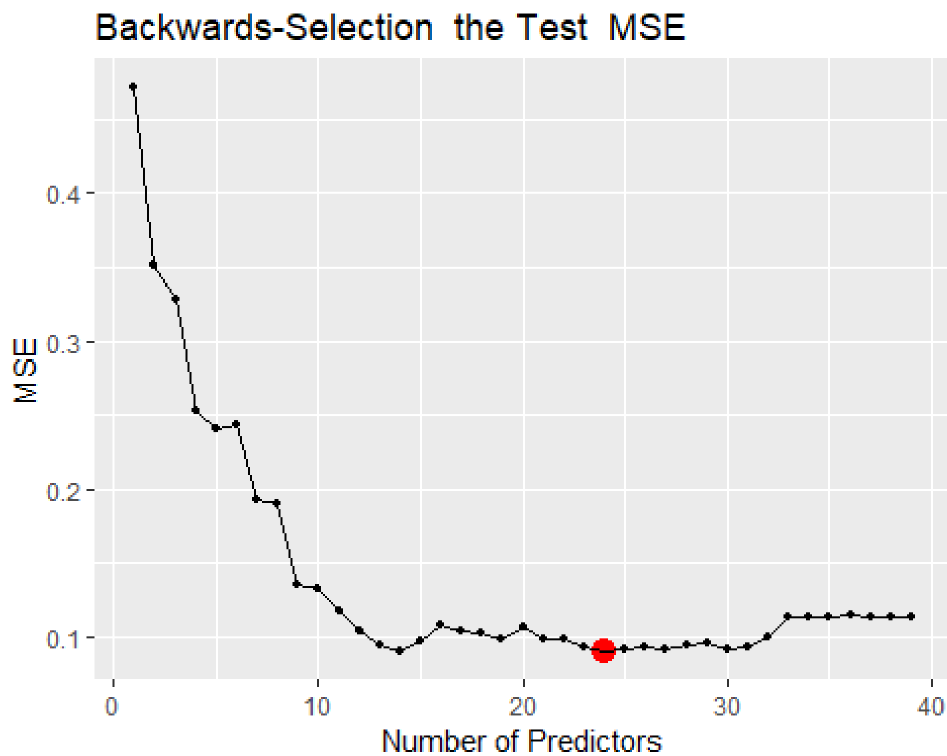

```
## Current Metric: Test  BIC
## Minimum Value:  48.07233
##  Number of Predictors:  4
```



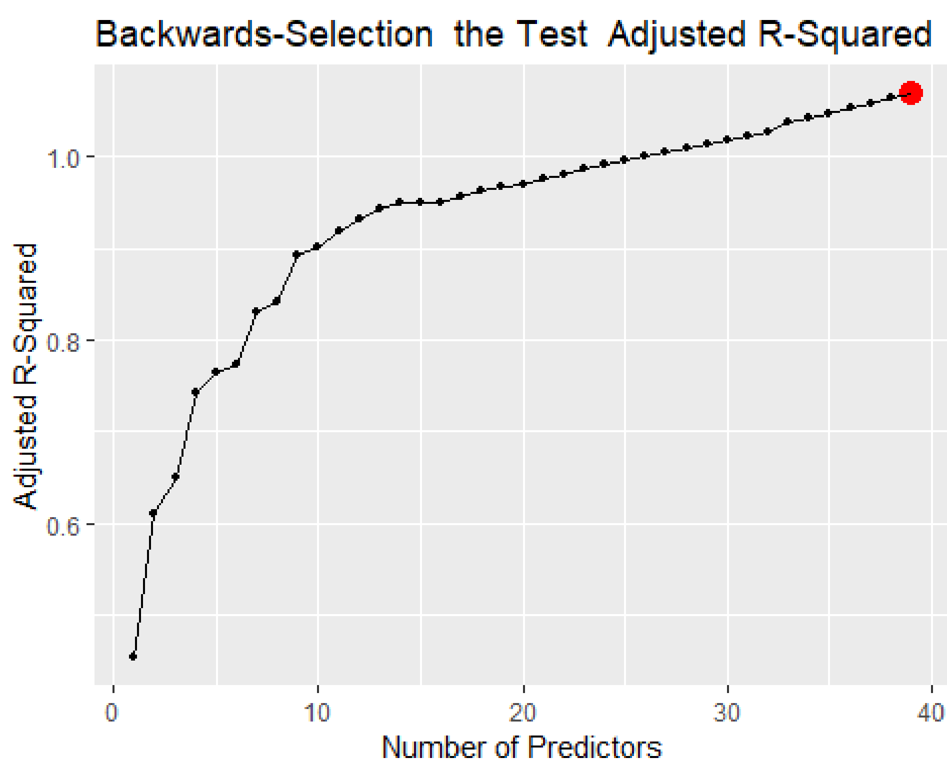

```
## Current Metric: Test  AIC (10)
## Minimum Value:  68.91002
##  Number of Predictors:  2
```

Backward Selection

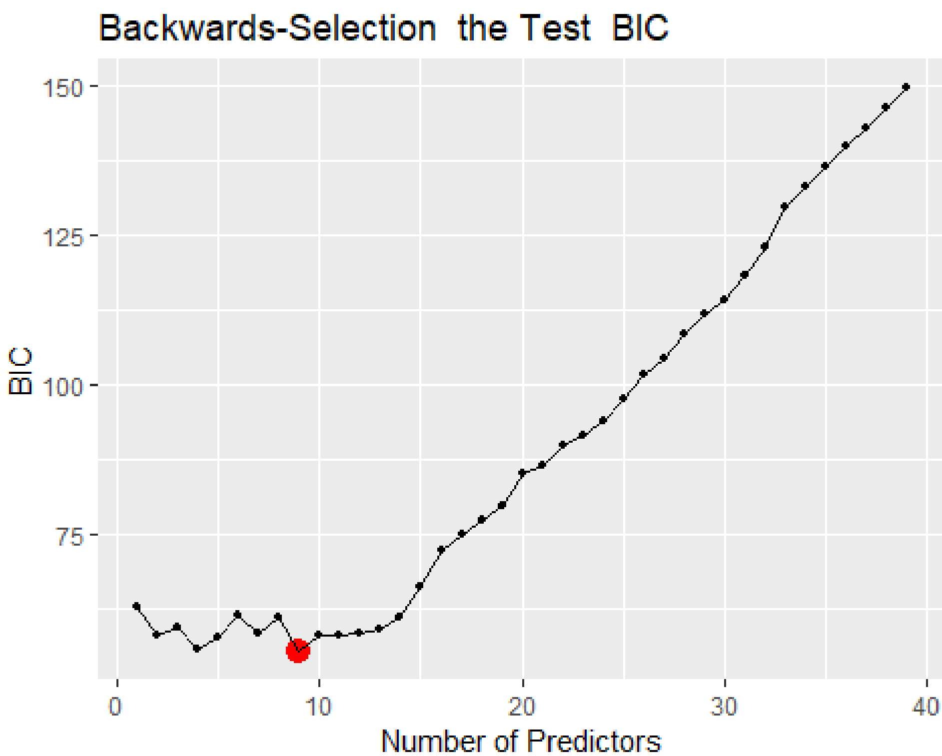
```
backwards.2008.results = do.selection.methods(hfi.2008.list$train,
hfi.2008.list$test,
                                              hfi.2008.list$tss.test,
ncol(hfi.2008.list$test)-2,
                                              type = "backwards")

print.out.graphs.lm(backwards.2008.results,
                    harsh.penalty = harsh.penalty,
                    title.name = "Backwards-Selection")
```



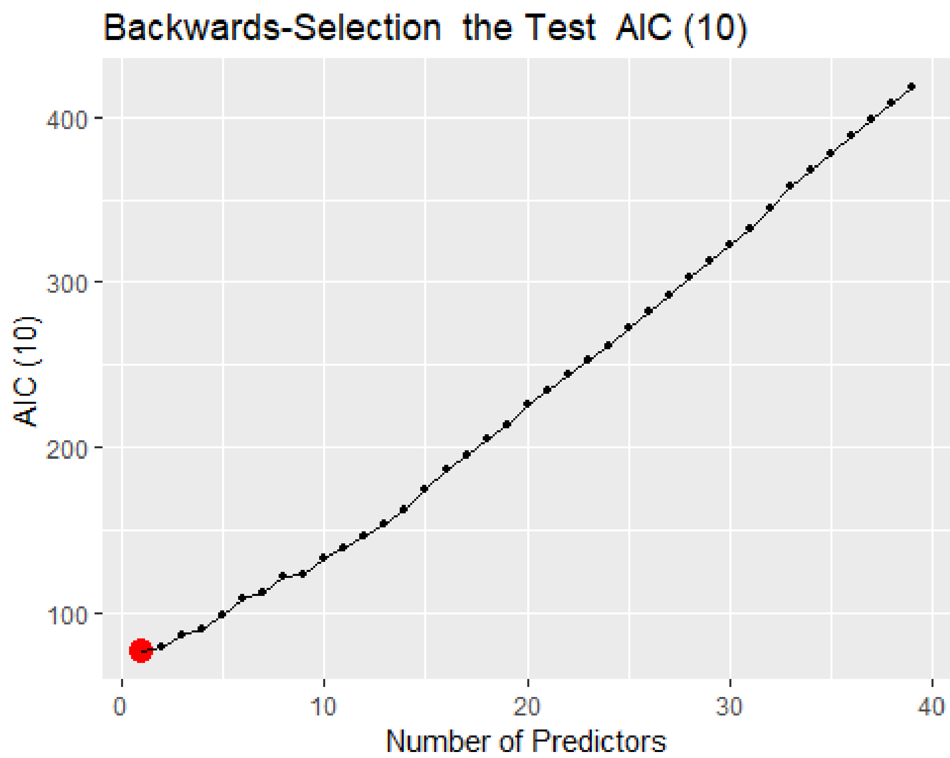

```
## Current Metric: Test  MSE
## Minimum Value:  0.09058843
##  Number of Predictors:  24
```



```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value:
##  Number of Predictors:
```




```
## Current Metric: Test  BIC
## Minimum Value:  55.55221
##  Number of Predictors:  9
```




```
## Current Metric: Test  AIC (10)
## Minimum Value:  76.30445
##  Number of Predictors:  1
```

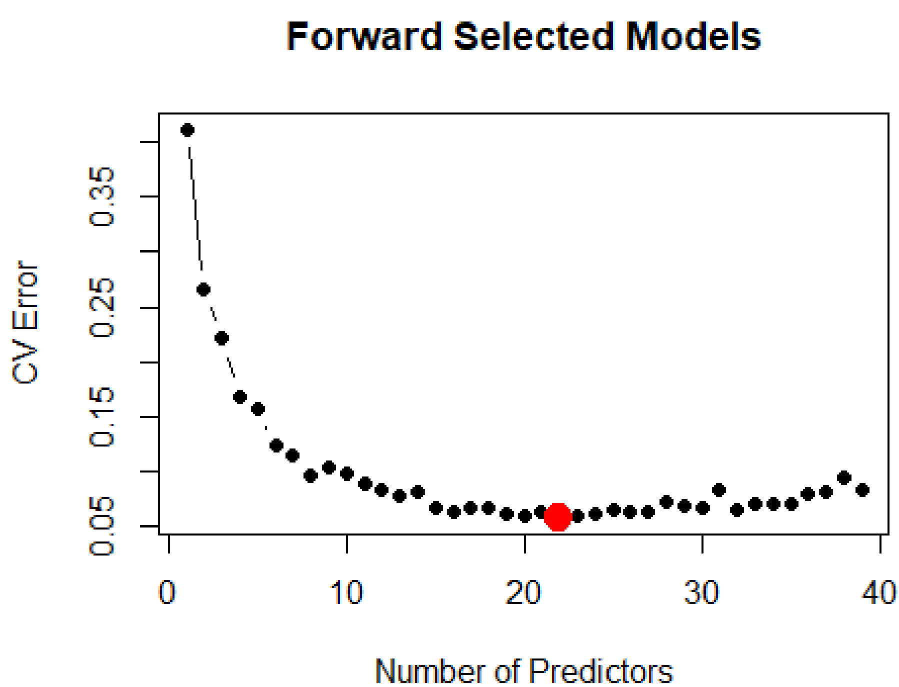
CV Errors:

This is the metric that should be used for sure!!

```
dataset.hfi = hfi.2008.list$train %>% full_join(hfi.2008.list$test)
```

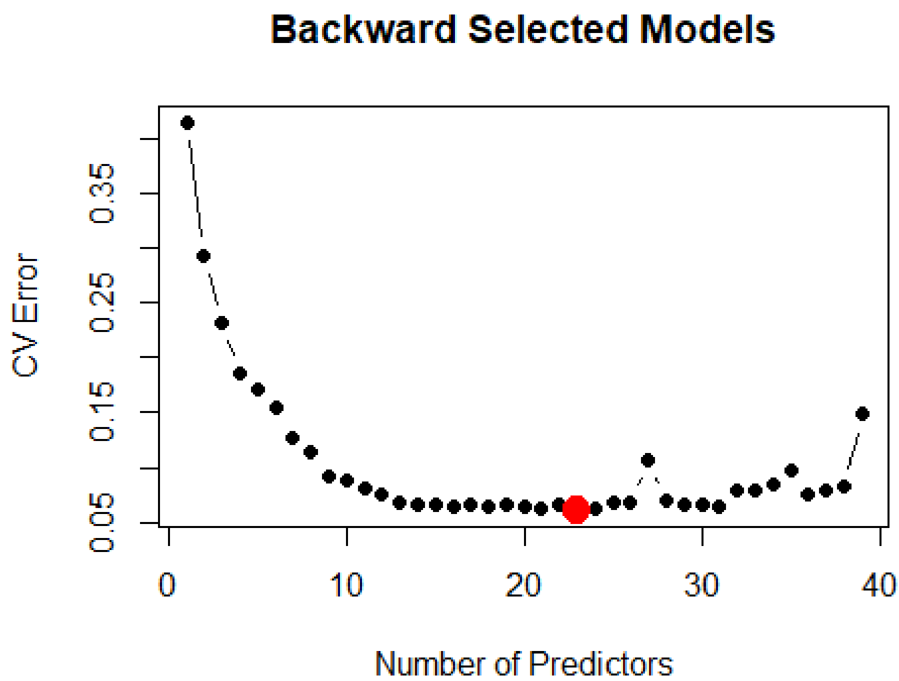
```
## Joining, by = c("year", "pf_ss_homicide", "pf_ss_disappearances_disap",
"pf_ss_disappearances_violent", "pf_ss_disappearances_fatalities",
"pf_ss_disappearances_injuries", "pf_movement_domestic",
"pf_movement_foreign", "pf_religion_harassment", "pf_religion_restrictions",
"pf_expression_killed", "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_regulatory", "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",
"hf_score")
```

```
cv.err.2008.results = list()
cv.err.2008.results[["forward"]] = get.cv.errors(data.hfi = dataset.hfi,
                                              regression.results =
forward.2008.results)
cv.err.2008.results[["backward"]] = get.cv.errors(data.hfi = dataset.hfi,
                                              regression.results =
backwards.2008.results)
cv.err.2008.results[["best.subset"]] = get.cv.errors(data.hfi = dataset.hfi,
                                              regression.results =
best.subset.2008.result)

cv.full.lm = cv.glm(data = dataset.hfi, K = 10, glmfit = glm(formula =
hf_score ~ . -year, data = dataset.hfi))

cv.full.lm.err = cv.full.lm$delta[1]
cv.err.2008.results[["full.lm"]] = cv.full.lm.err


plot.cv.errors(cv.err.2008.results$forward, title.string = "Forward Selected
Models")
```

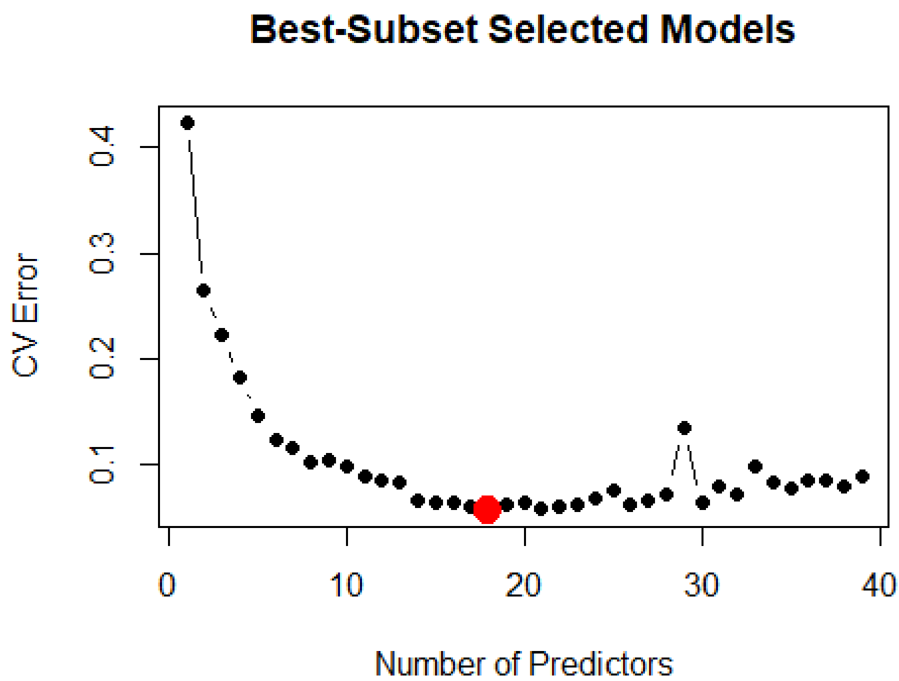



```
## NULL
```

```
plot.cv.errors(cv.err.2008.results$backward, title.string = "Backward
Selected Models")
```




```
## NULL

plot.cv.errors(cv.err.2008.results$best.subset, title.string = "Best-Subset
Selected Models")
```




```
## NULL

# These are the errors from the best subset models with the lowest errors:
which.min(cv.err.2008.results$forward)

## [1] 22

cv.err.2008.results$forward[which.min(cv.err.2008.results$forward)]

## [1] 0.05839676

which.min(cv.err.2008.results$best.subset)

## [1] 18

cv.err.2008.results$best.subset[which.min(cv.err.2008.results$best.subset)]

## [1] 0.0564797

# total preds (-1 because 1 is year that is unused)
ncol(hfi.features)-1

## [1] 39

cv.err.2008.results$full.lm

## [1] 0.08430769

hfi.2008.list$tss
```

```
## [1] 0.9887557
```

```r
coef(forward.2008.results$model,  which.min(cv.err.2008.results$forward))
```

```
##                  (Intercept)                   pf_ss_homicide
##                   0.58202859                       0.05978641
##         pf_movement_domestic         pf_religion_restrictions
##                   0.01819984                       0.04357334
##         pf_expression_killed          pf_expression_influence
##                   0.01409210                       0.08726758
##         pf_identity_sex_male           pf_identity_sex_female
##                   0.02566442                       0.02442359
##     ef_government_consumption                  ef_legal_courts
##                   0.07224351                       0.07417869
##            ef_legal_military             ef_legal_enforcement
##                   0.05440078                       0.06971836
##              ef_legal_gender                  ef_money_growth
##                   0.62102879                       0.05377001
##                  ef_money_sd               ef_money_inflation
##                   0.04160981                       0.04709256
##            ef_money_currency            ef_trade_tariffs_mean
##                   0.03013181                       0.03620479
## ef_trade_regulatory_compliance        ef_trade_movement_capital
##                   0.01880272                       0.02520815
##       ef_trade_movement_visit   ef_regulation_credit_private
##                   0.01818030                       0.04909222
##   ef_regulation_business_start
##                   0.02684369
```

```r
coef(backwards.2008.results$model, which.min(cv.err.2008.results$backward))
```

```
##                  (Intercept)                   pf_ss_homicide
##                   0.64303482                       0.06392195
##         pf_movement_domestic         pf_religion_restrictions
##                   0.02094384                       0.04035257
##      pf_expression_influence            pf_expression_control
##                   0.05249520                       0.04361201
##         pf_identity_sex_male           pf_identity_sex_female
##                   0.03560970                       0.01344629
##     ef_government_consumption                  ef_legal_courts
##                   0.07588022                       0.06260777
##            ef_legal_military             ef_legal_enforcement
##                   0.05655546                       0.06848500
##              ef_legal_gender                  ef_money_growth
##                   0.76652069                       0.07790905
##           ef_money_inflation                ef_money_currency
##                   0.05819944                       0.04168133
##        ef_trade_tariffs_mean              ef_trade_tariffs_sd
##                   0.06577977                       0.02807053
##             ef_trade_tariffs ef_trade_regulatory_compliance
##                  -0.08236920                       0.02734332
```

```
##      ef_trade_movement_capital        ef_trade_movement_visit
##                     0.02164087                     0.02166967
##   ef_regulation_credit_private      ef_regulation_labor_hours
##                     0.06275343                     0.02053494
```

```
coef(best.subset.2008.result$model,
which.min(cv.err.2008.results$best.subset))
```

```
##                  (Intercept)               pf_ss_homicide
##                   0.87163515                   0.06506737
##          pf_movement_domestic     pf_religion_restrictions
##                   0.02128970                   0.04333349
##      pf_expression_influence          pf_identity_sex_male
##                   0.09336189                   0.02575193
##        pf_identity_sex_female    ef_government_consumption
##                   0.01964080                   0.07294603
##              ef_legal_courts              ef_legal_military
##                   0.07751929                   0.06119214
##          ef_legal_enforcement              ef_legal_gender
##                   0.07613082                   0.66834955
##              ef_money_growth                  ef_money_sd
##                   0.06085312                   0.05128388
##           ef_money_inflation            ef_money_currency
##                   0.05120621                   0.03433595
##    ef_trade_movement_capital      ef_trade_movement_visit
##                   0.02901782                   0.02004611
## ef_regulation_credit_private
##                   0.05757708
```

Ridge

```
ridge.2008 = do.ridge.lasso(hfi.2008.list$train,
                            hfi.2008.list$test,
                            alpha=0)

ridge.2008$model$beta
```

```
## 39 x 1 sparse Matrix of class "dgCMatrix"
##                                          s0
## pf_ss_homicide                  0.042384545
## pf_ss_disappearances_disap     -0.004656054
## pf_ss_disappearances_violent    0.002938750
## pf_ss_disappearances_fatalities 0.019486686
## pf_ss_disappearances_injuries   0.004709564
## pf_movement_domestic            0.015914970
## pf_movement_foreign             0.010870032
## pf_religion_harassment          0.010521449
## pf_religion_restrictions        0.035177684
## pf_expression_killed            0.006781064
## pf_expression_jailed           -0.029734975
## pf_expression_influence         0.043579043
```

```
## pf_expression_control                   0.042646640
## pf_identity_sex_male                    0.025878307
## pf_identity_sex_female                  0.023808199
## ef_government_consumption               0.034660770
## ef_legal_courts                         0.045436476
## ef_legal_military                       0.035576682
## ef_legal_enforcement                    0.055984795
## ef_legal_gender                         0.543648235
## ef_money_growth                         0.039519883
## ef_money_sd                             0.047684203
## ef_money_inflation                      0.038690648
## ef_money_currency                       0.022043509
## ef_trade_tariffs_mean                   0.041345785
## ef_trade_tariffs_sd                    -0.009610650
## ef_trade_tariffs                        0.011347258
## ef_trade_regulatory_compliance          0.017359074
## ef_trade_regulatory                     0.025367771
## ef_trade_black                          0.028195236
## ef_trade_movement_capital               0.025802825
## ef_trade_movement_visit                 0.015078237
## ef_regulation_credit_private            0.047803318
## ef_regulation_credit                    0.010204022
## ef_regulation_labor_minwage             0.005076600
## ef_regulation_labor_hours               0.014619672
## ef_regulation_labor_conscription        0.001742414
## ef_regulation_business_start            0.032598805
## ef_regulation_business_compliance       0.009832968
```

```
ridge.2008$best.lambda
```

```
## [1] 0.1928792
```

```
ridge.2008$mse
```

```
## [1] 0.08140788
```

Lasso

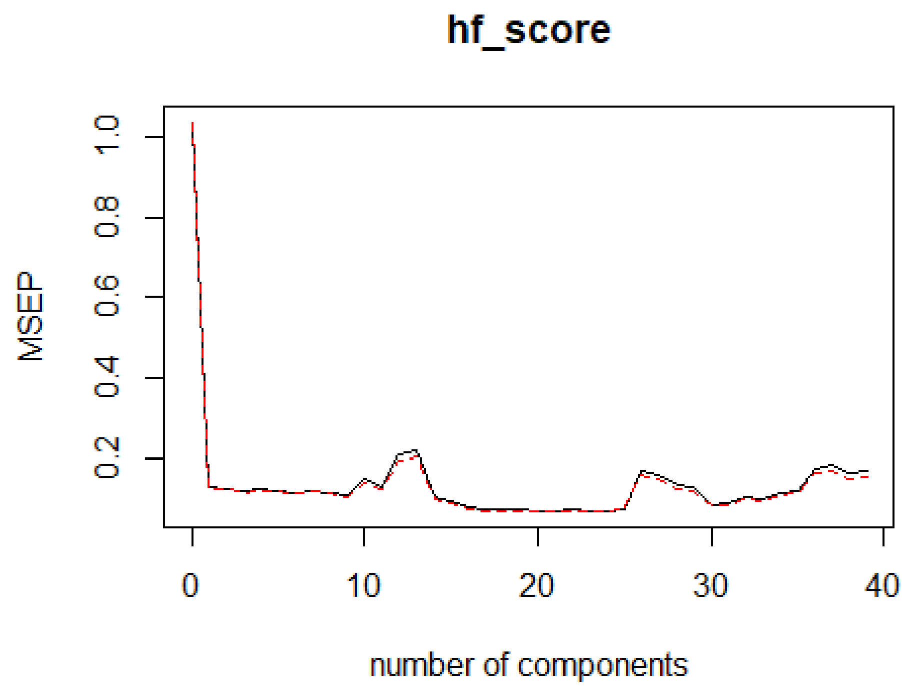
```
lasso.2008 = do.ridge.lasso(hfi.2008.list$train,
                             hfi.2008.list$test,
                             alpha=1)

rownames.no.na = c()
for (i in 1:length(rownames(lasso.2008$model$beta))){
  if( lasso.2008$model$beta[i] != 0){
    rownames.no.na = c(rownames.no.na, rownames(lasso.2008$model$beta)[i])
  }
}
lasso.2008[["rownames.left"]] = rownames.no.na

lasso.2008$model$beta
```

```
## 39 x 1 sparse Matrix of class "dgCMatrix"
##                                              s0
## pf_ss_homicide                      0.051925494
## pf_ss_disappearances_disap          .
## pf_ss_disappearances_violent        .
## pf_ss_disappearances_fatalities     .
## pf_ss_disappearances_injuries       .
## pf_movement_domestic                0.016333939
## pf_movement_foreign                 0.005507928
## pf_religion_harassment              .
## pf_religion_restrictions            0.038350280
## pf_expression_killed                0.005425426
## pf_expression_jailed                .
## pf_expression_influence             0.067609862
## pf_expression_control               0.025856705
## pf_identity_sex_male                0.028926556
## pf_identity_sex_female              0.020726154
## ef_government_consumption           0.050832937
## ef_legal_courts                     0.054483497
## ef_legal_military                   0.049930910
## ef_legal_enforcement                0.067300524
## ef_legal_gender                     0.559965651
## ef_money_growth                     0.043568824
## ef_money_sd                         0.042870239
## ef_money_inflation                  0.040185798
## ef_money_currency                   0.026497147
## ef_trade_tariffs_mean               0.025891408
## ef_trade_tariffs_sd                 .
## ef_trade_tariffs                    .
## ef_trade_regulatory_compliance      0.007836152
## ef_trade_regulatory                 0.023231171
## ef_trade_black                      0.018854713
## ef_trade_movement_capital           0.028204793
## ef_trade_movement_visit             0.015459187
## ef_regulation_credit_private        0.050369003
## ef_regulation_credit                .
## ef_regulation_labor_minwage         .
## ef_regulation_labor_hours           0.012387191
## ef_regulation_labor_conscription    .
## ef_regulation_business_start        0.024381128
## ef_regulation_business_compliance 0.004687157

lasso.2008$best.lambda

## [1] 0.01028045

lasso.2008$mse

## [1] 0.08406437

lasso.2008$rownames.left
```

```
##  [1] "pf_ss_homicide"
##  [2] "pf_movement_domestic"
##  [3] "pf_movement_foreign"
##  [4] "pf_religion_restrictions"
##  [5] "pf_expression_killed"
##  [6] "pf_expression_influence"
##  [7] "pf_expression_control"
##  [8] "pf_identity_sex_male"
##  [9] "pf_identity_sex_female"
## [10] "ef_government_consumption"
## [11] "ef_legal_courts"
## [12] "ef_legal_military"
## [13] "ef_legal_enforcement"
## [14] "ef_legal_gender"
## [15] "ef_money_growth"
## [16] "ef_money_sd"
## [17] "ef_money_inflation"
## [18] "ef_money_currency"
## [19] "ef_trade_tariffs_mean"
## [20] "ef_trade_regulatory_compliance"
## [21] "ef_trade_regulatory"
## [22] "ef_trade_black"
## [23] "ef_trade_movement_capital"
## [24] "ef_trade_movement_visit"
## [25] "ef_regulation_credit_private"
## [26] "ef_regulation_labor_hours"
## [27] "ef_regulation_business_start"
## [28] "ef_regulation_business_compliance"
```

PCR

```
pcr.2008.model = pcr(hf_score ~ . -year ,  scale = TRUE,
                     data = hfi.2008.list$train, validation = "CV")
print(validationplot(pcr.2008.model, val.type = "MSEP"))
```




```
## NULL
```

```r
pcr.2008.best = find.best.mse.pcr.pls(pcr.2008.model, hfi.2008.list$test)
pcr.2008.best$min.val
```

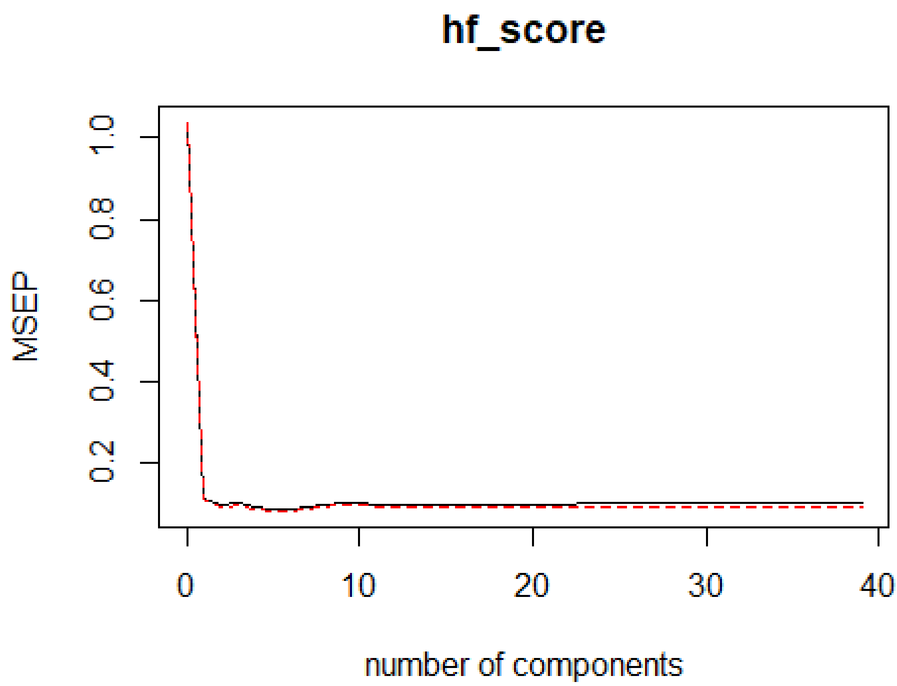
```
## [1] 4
```

```r
pcr.2008.best$best.mse
```

```
## [1] 0.05994815
```

PLSR

```r
# create pls model and plot validation; report the best mse
pls.2008.model = plsr(hf_score ~ . -year, scale = TRUE,
                 data = hfi.2008.list$train, validation = "CV")
validationplot(pls.2008.model, val.type = "MSEP")
```




```
pls.2008.best = find.best.mse.pcr.pls(pls.2008.model, hfi.2008.list$test)
pls.2008.best$min.val

## [1] 1

pls.2008.best$best.mse

## [1] 0.06413841
```

Matrix of Results 2008

```
x = matrix(data=
           c("Full OLS, 39 Predictors (test/train)",
             "Full OLS, 39 Predictors (10-fold CV)",
             paste0("Forward Select, ",
which.min(forward.2008.results$forward.mse), " Predictors (test/train)"),
             paste0("Forward Select, ",
which.min(cv.err.2008.results$forward), " Predictors (10-fold CV)"),
             paste0("Backward Select, ",
which.min(backwards.2008.results$forward.mse)," Predictors (test/train)"),
             paste0("Backward Select, ",
which.min(cv.err.2008.results$backward), " Predictors (10-fold CV)"),
             paste0("Best Subset, ",
which.min(best.subset.2008.result$best.subset.mse)," Predictors
(test/train)"),
             paste0("Best Subset, ",
which.min(cv.err.2008.results$best.subset), " Predictors (10-fold CV)"),
             paste0("Ridge, ", length(ridge.2008$model$beta), " Predictors
```

```
(test/train)"),
             paste0("Lasso, ", length(lasso.2008$rownames.left), " Predictors
(test/train)"),
             paste0("PCR, ", pcr.2008.best$min.val ," Components,
(test/train)"),
             paste0("PLSR, ", pls.2008.best$min.val, " Components,
(test/train)"),
             "Mean TSS",
mse.lm.2008,
cv.err.2008.results$full.lm,
min(forward.2008.results$forward.mse), # fit 3
min(cv.err.2008.results$forward),
min(backwards.2008.results$forward.mse),
min(cv.err.2008.results$backward),
min(best.subset.2008.result$best.subset.mse),
min(cv.err.2008.results$best.subset),
ridge.2008$mse,
lasso.2008$mse,
pcr.2008.best$best.mse,
pls.2008.best$best.mse,
hfi.2008.list$tss
), nrow=13, ncol=2)
colnames(x) <- c("Method","test MSE")

df.x.all.2008 = as.data.frame(x, row.names = list.x.axis)
df.x.all.2008$`test MSE` = round(as.numeric(as.character(df.x.all.2008[["test
MSE"]])), 5)
df.x.all.2008

##                                            Method test MSE
## 1          Full OLS, 39 Predictors (test/train)  0.11389
## 2          Full OLS, 39 Predictors (10-fold CV)  0.08431
## 3    Forward Select, 26 Predictors (test/train)  0.09184
## 4    Forward Select, 22 Predictors (10-fold CV)  0.05840
## 5   Backward Select, 24 Predictors (test/train)  0.09059
## 6   Backward Select, 23 Predictors (10-fold CV)  0.06047
## 7       Best Subset, 14 Predictors (test/train)  0.09122
## 8       Best Subset, 18 Predictors (10-fold CV)  0.05648
## 9             Ridge, 39 Predictors (test/train)  0.08141
## 10            Lasso, 28 Predictors (test/train)  0.08406
## 11             PCR, 4 Components, (test/train)  0.05995
## 12            PLSR, 1 Components, (test/train)  0.06414
## 13                                     Mean TSS  0.98876
```

2016

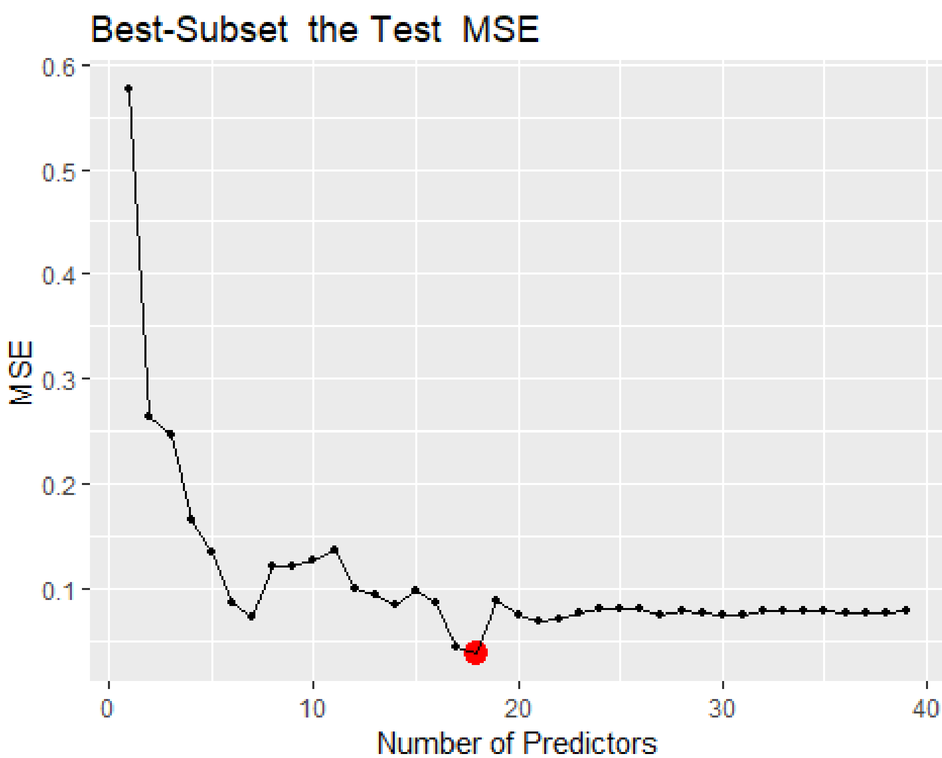
Full Model

```
lm.fit.2016 = lm(hf_score ~ . -year, data = hfi.2016.list[["train"]])
lm.preds = predict(lm.fit.2016, newdata = hfi.2016.list[["test"]])
# get MSE and R^2 (just 1 - MSE/MEAN(TSS) === 1 - RSS/TSS)
```

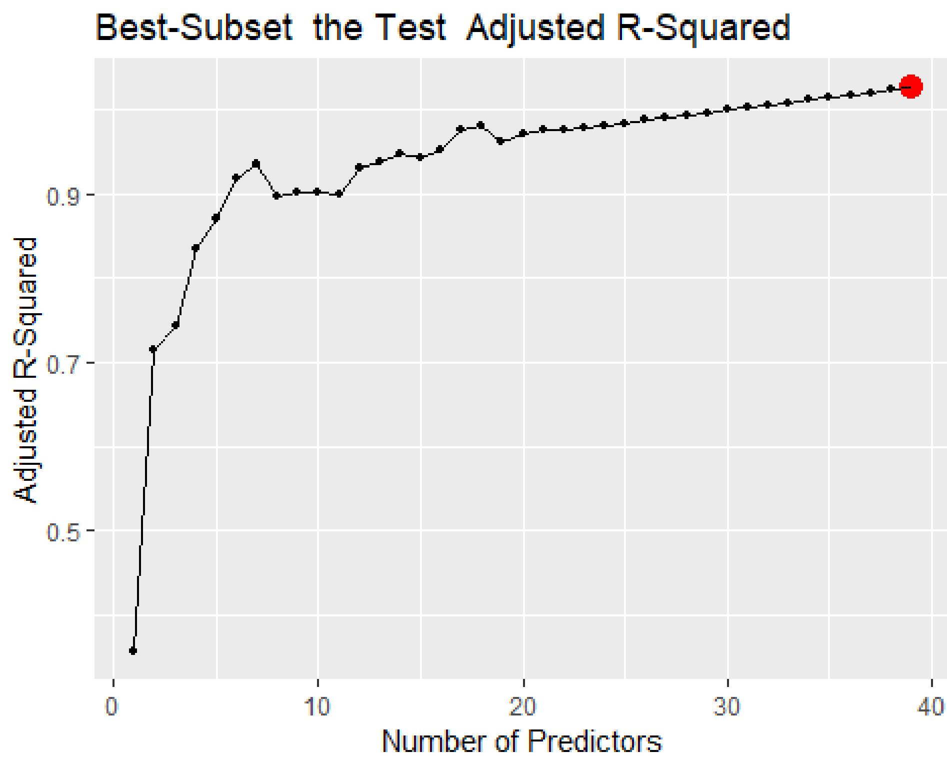
```
mse.lm.2016 = mean((lm.preds - hfi.2016.list[["test"]]$hf_score)^2)
lm.r2 = 1 - (mse.lm.2016/hfi.2016.list[["tss.test"]])
mse.lm.2016

## [1] 0.07813294

lm.r2

## [1] 0.9096844
```

Best-Subset

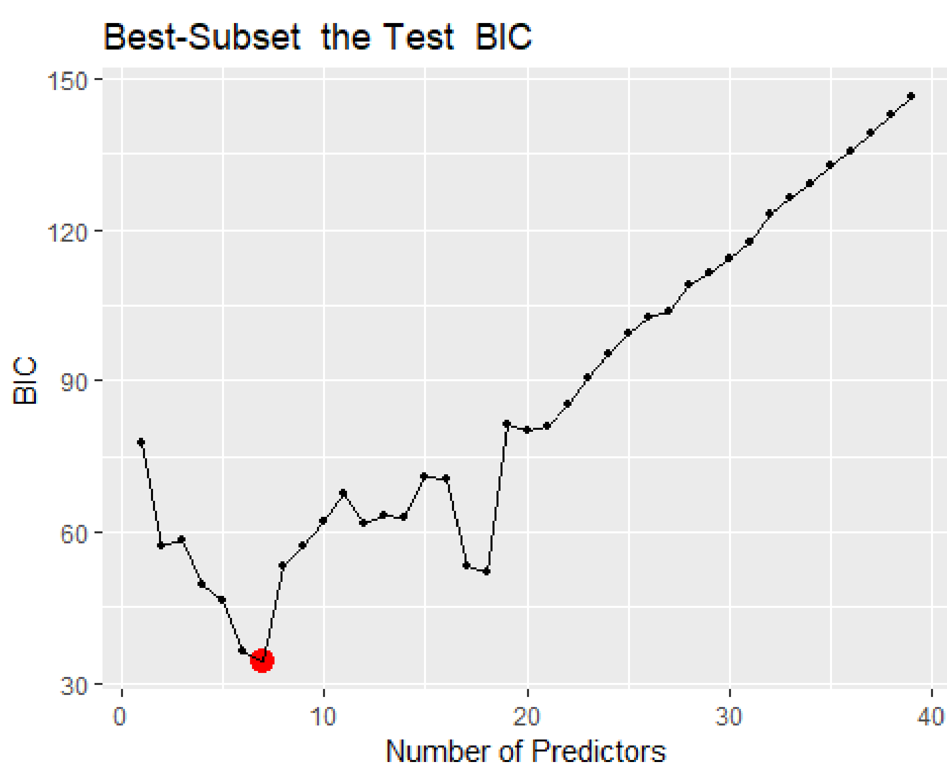
```
max.variables.to.run = ncol(hfi.2016.list$train) - 2
best.subset.2016.result = get.best.subset(hfi.2016.list$train,
hfi.2016.list$test,
                                          hfi.2016.list$tss.test,
max.variables.to.run)

print.out.graphs.lm(best.subset.2016.result,
                    harsh.penalty = harsh.penalty,
                    title.name = "Best-Subset")
```



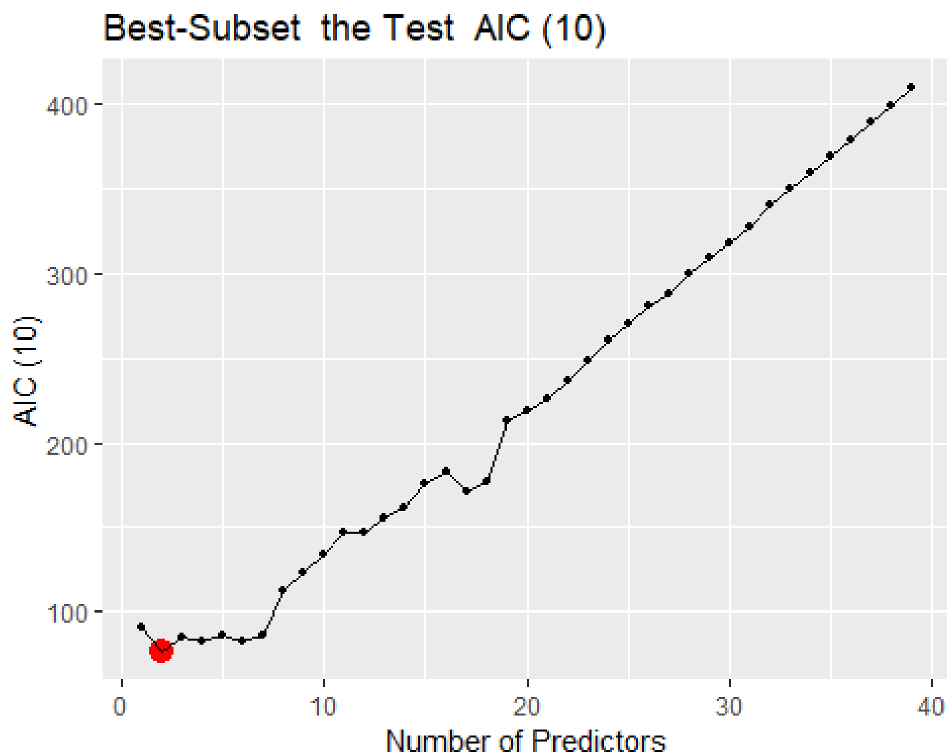

```
## Current Metric: Test  MSE
## Minimum Value:  0.03838836
##  Number of Predictors:  18
```




```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value:  39
##  Number of Predictors:  1.027095
```



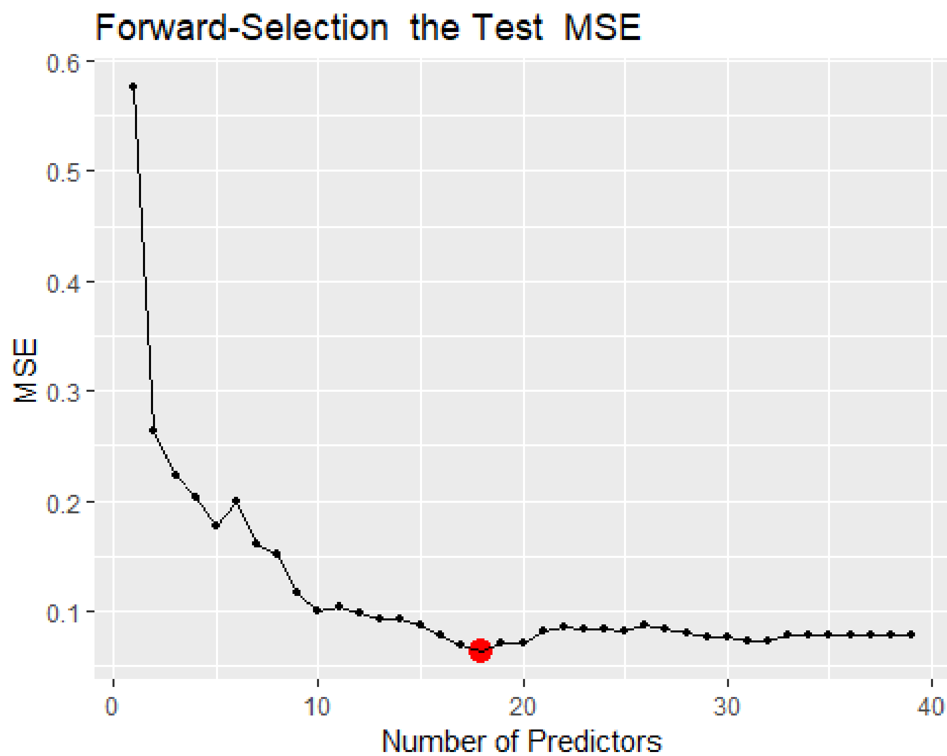
```
## Current Metric: Test  BIC
## Minimum Value:  34.28636
##  Number of Predictors:  7
```



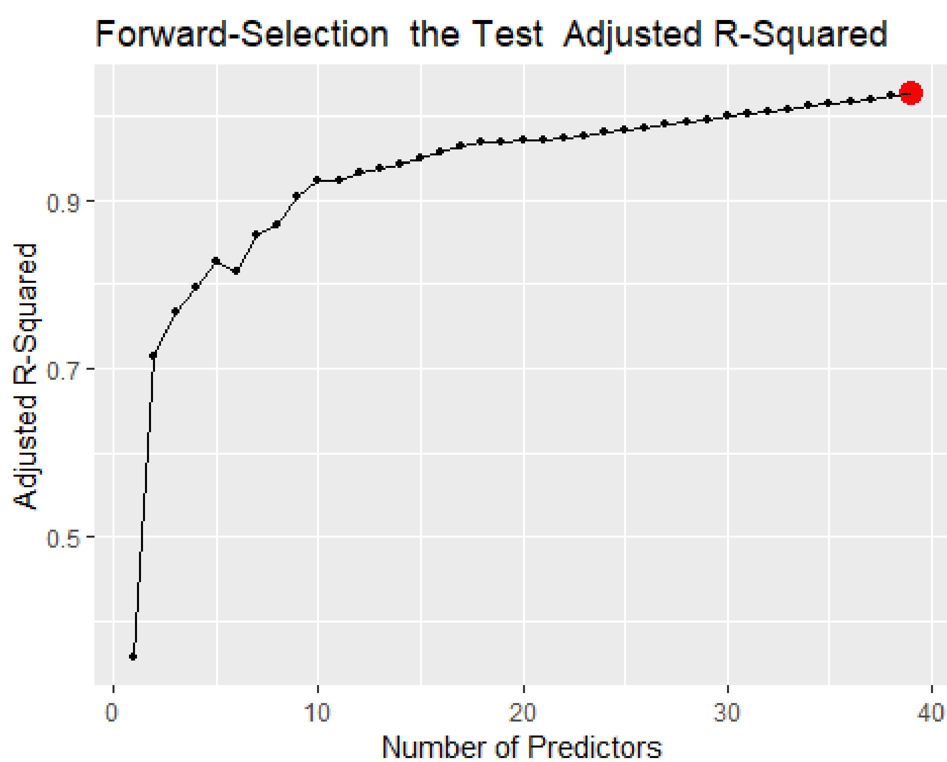

```
## Current Metric: Test  AIC (10)
## Minimum Value:  76.73236
##  Number of Predictors:  2
```

Forward Selection

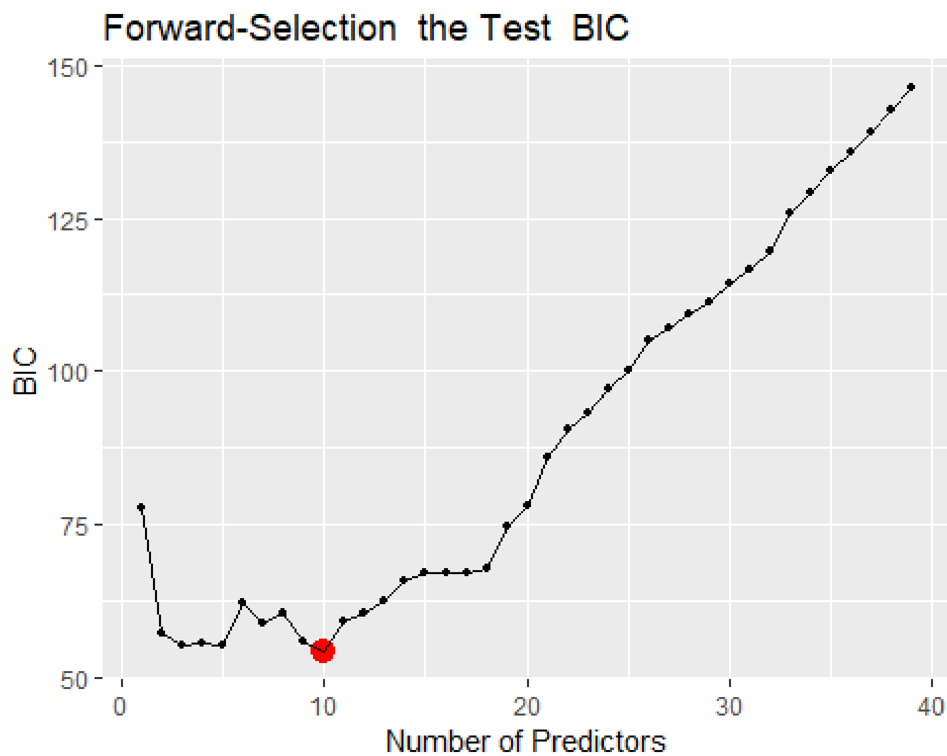
```
forward.2016.results = do.selection.methods(hfi.2016.list$train,
hfi.2016.list$test,
                                            hfi.2016.list$tss.test,
ncol(hfi.2016.list$test)-2,
                                            type = "forward")

print.out.graphs.lm(forward.2016.results,
                    harsh.penalty = harsh.penalty,
                    title.name = "Forward-Selection")
```



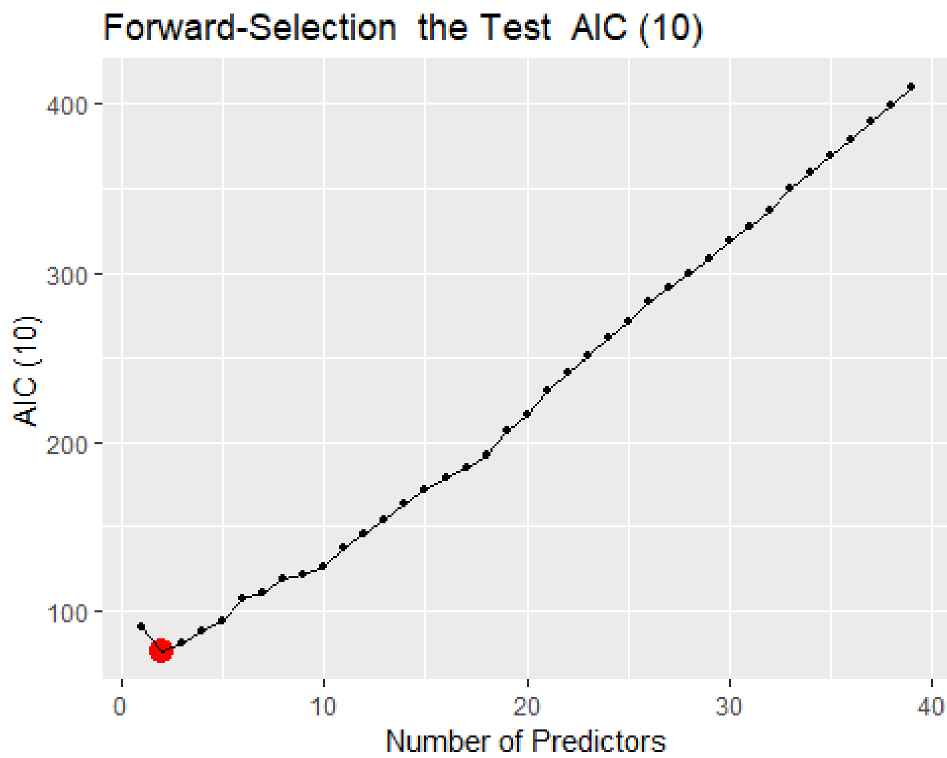

```
## Current Metric: Test  MSE
## Minimum Value:  0.06334112
##  Number of Predictors:  18
```



```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value:
##  Number of Predictors:
```



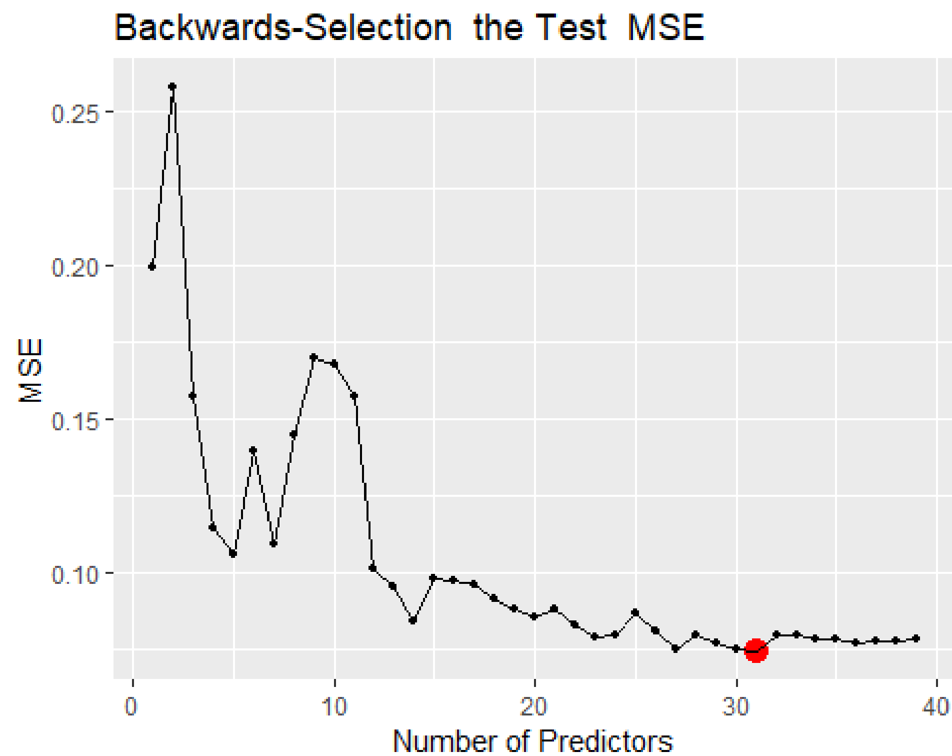

```
## Current Metric: Test  BIC
## Minimum Value:  54.15174
##  Number of Predictors:  10
```



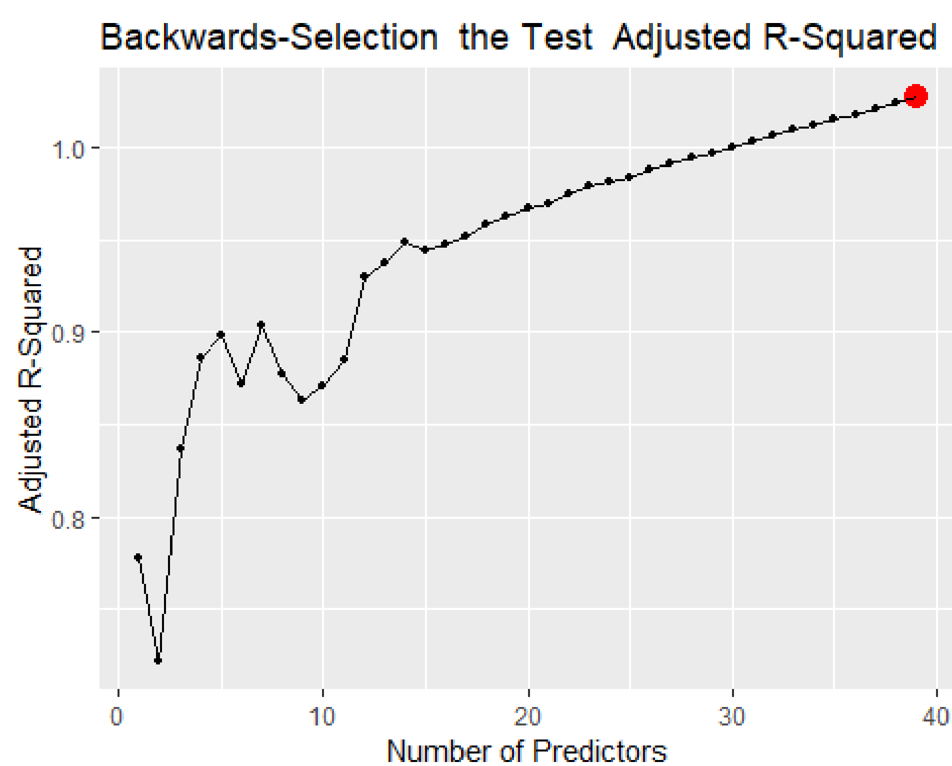

```
## Current Metric: Test  AIC (10)
## Minimum Value:  76.73236
##  Number of Predictors:  2
```

Backward Selection

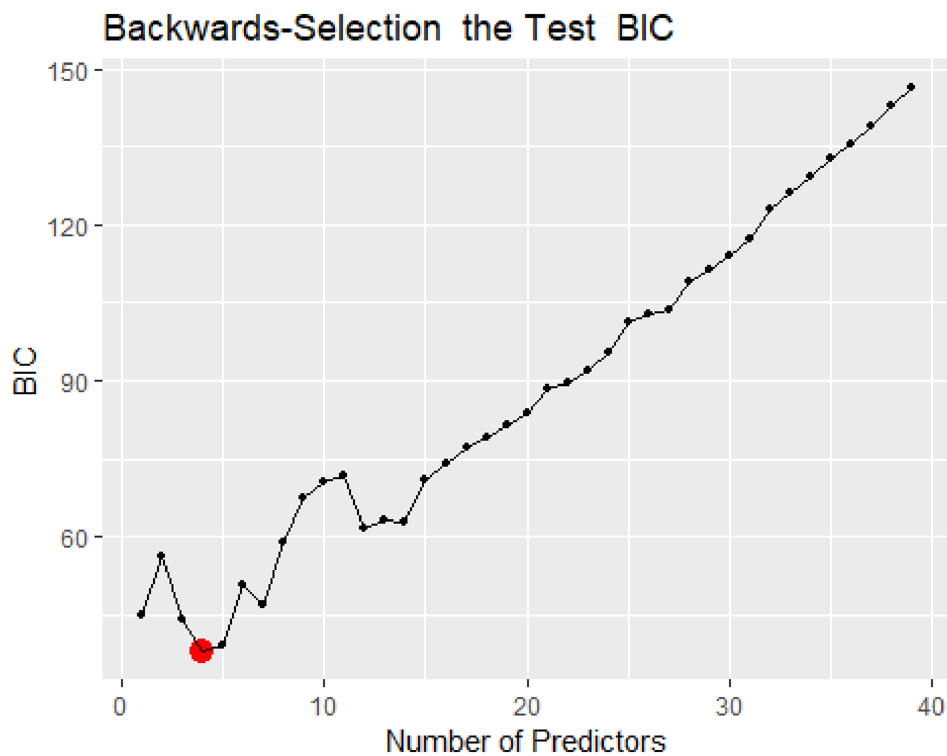
```
backwards.2016.results = do.selection.methods(hfi.2016.list$train,
hfi.2016.list$test,
                                              hfi.2016.list$tss.test,
ncol(hfi.2016.list$test)-2,
                                              type = "backwards")

print.out.graphs.lm(backwards.2016.results,
                    harsh.penalty = harsh.penalty,
                    title.name = "Backwards-Selection")
```



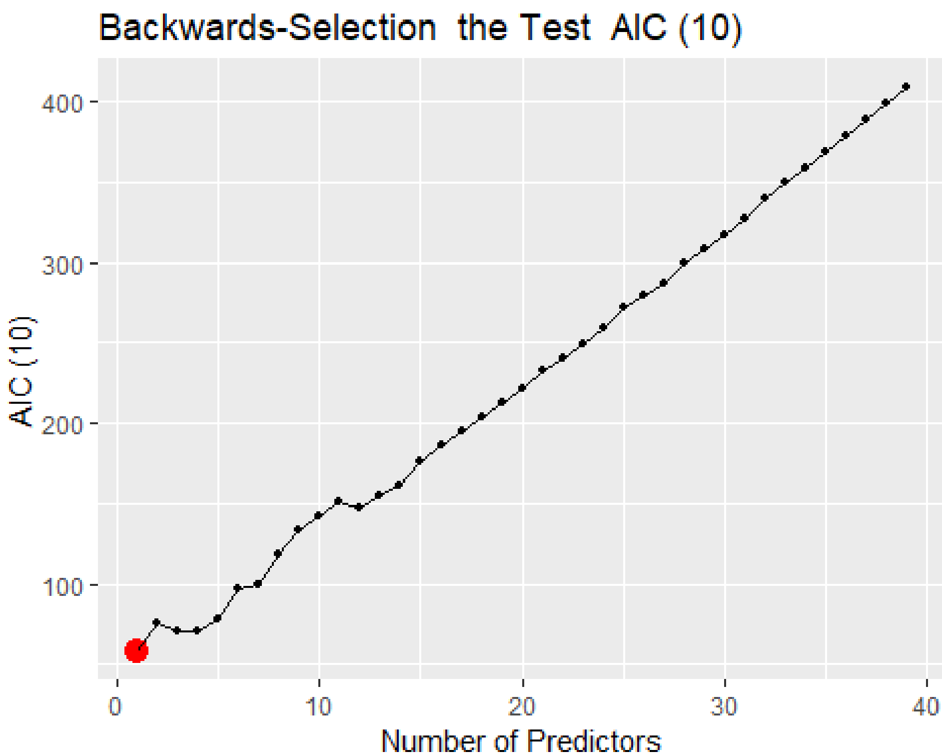

```
## Current Metric: Test  MSE
## Minimum Value:  0.07425556
##  Number of Predictors:  31
```



```
## Current Metric: Test  Adjusted R-Squared
## Maximum Value:
##  Number of Predictors:
```




```
## Current Metric: Test  BIC
## Minimum Value:  37.9121
##  Number of Predictors:  4
```



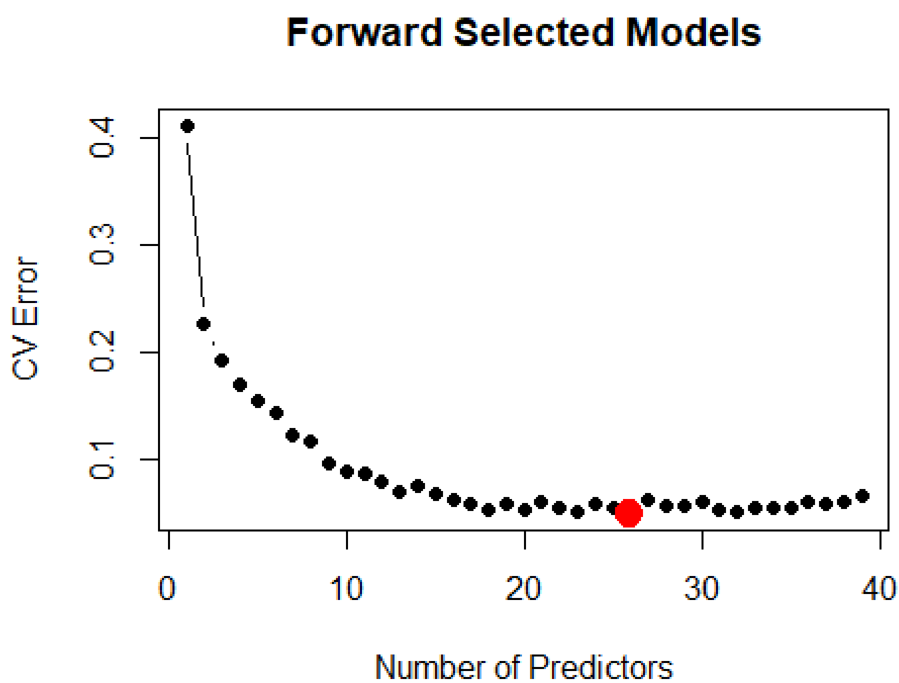

```
## Current Metric: Test  AIC (10)
## Minimum Value:  57.97469
##  Number of Predictors:  1
```

CV Errors:

This is the metric that should be used for sure!!

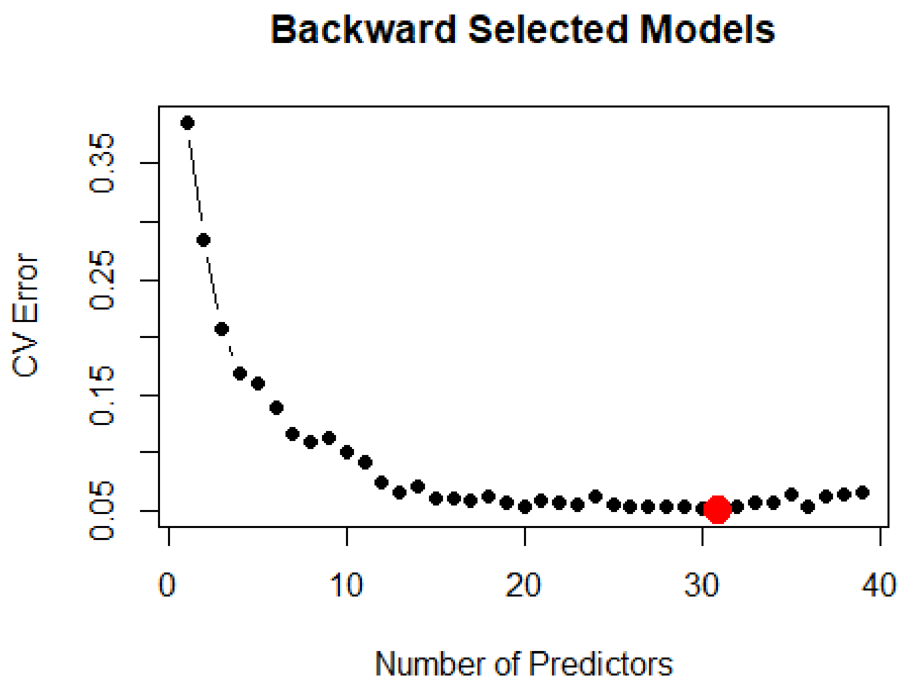
```
dataset.hfi = hfi.2016.list$train %>% full_join(hfi.2016.list$test)

cv.err.2016.results = list()
cv.err.2016.results[["forward"]] = get.cv.errors(data.hfi = dataset.hfi,
                                               regression.results =
forward.2016.results)
cv.err.2016.results[["backward"]] = get.cv.errors(data.hfi = dataset.hfi,
                                         regression.results =
backwards.2016.results)
cv.err.2016.results[["best.subset"]] = get.cv.errors(data.hfi = dataset.hfi,
                                            regression.results =
best.subset.2016.result)

cv.full.lm = cv.glm(data = dataset.hfi, K = 10, glmfit = glm(formula =
hf_score ~ . -year, data = dataset.hfi))

cv.full.lm.err = cv.full.lm$delta[1]
cv.err.2016.results[["full.lm"]] = cv.full.lm.err
```

```r
plot.cv.errors(cv.err.2016.results$forward, title.string = "Forward Selected
Models")
```

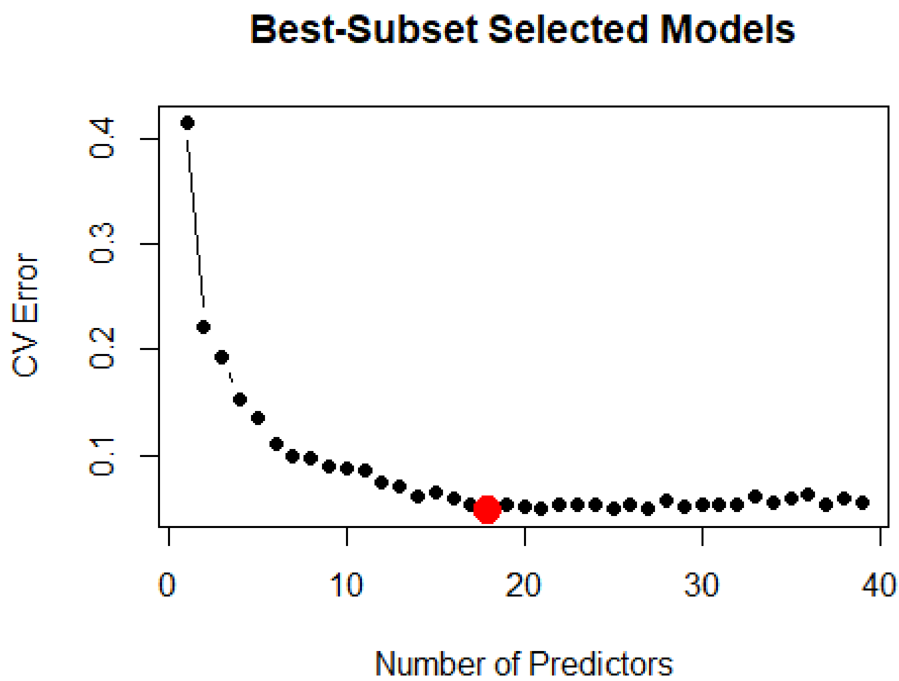



```
## NULL
```

```r
plot.cv.errors(cv.err.2016.results$backward, title.string = "Backward
Selected Models")
```




```
## NULL

plot.cv.errors(cv.err.2016.results$best.subset, title.string = "Best-Subset
Selected Models")
```




```
## NULL
```

```r
# These are the errors from the best subset models with the lowest errors:
which.min(cv.err.2016.results$forward)
```

```
## [1] 26
```

```r
cv.err.2016.results$forward[which.min(cv.err.2016.results$forward)]
```

```
## [1] 0.0495289
```

```r
which.min(cv.err.2016.results$best.subset)
```

```
## [1] 18
```

```r
cv.err.2016.results$best.subset[which.min(cv.err.2016.results$best.subset)]
```

```
## [1] 0.047935
```

```r
# total preds (-1 because 1 is year that is unused)
ncol(hfi.features)-1
```

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

```r
cv.err.2016.results$full.lm
```

```
## [1] 0.05532838
```

```r
hfi.2016.list$tss
```

```
## [1] 1.017658
```

```r
coef(forward.2016.results$model,  which.min(cv.err.2016.results$forward))
```

```
##                      (Intercept)                    pf_ss_homicide
##                      0.237785902                       0.035152858
##     pf_ss_disappearances_violent              pf_movement_domestic
##                      0.036114461                       0.029514810
##         pf_religion_restrictions             pf_expression_killed
##                      0.069764390                       0.027413113
##             pf_expression_jailed          pf_expression_influence
##                     -0.020951067                       0.027091933
##            pf_expression_control            pf_identity_sex_female
##                      0.087252444                       0.024568682
##       ef_government_consumption                   ef_legal_courts
##                      0.050805656                       0.056751039
##                ef_legal_military              ef_legal_enforcement
##                      0.061771736                       0.046826409
##                  ef_legal_gender                   ef_money_growth
##                      0.489575400                       0.059429458
##                      ef_money_sd                 ef_money_currency
##                      0.060179818                       0.020223547
##               ef_trade_tariffs_sd                 ef_trade_tariffs
##                     -0.048002977                       0.146913520
##               ef_trade_regulatory                   ef_trade_black
##                      0.074408939                      -0.023269754
##       ef_trade_movement_capital          ef_trade_movement_visit
##                      0.020903329                       0.010941155
##     ef_regulation_credit_private              ef_regulation_credit
##                      0.027162894                       0.053171218
## ef_regulation_labor_conscription
##                      0.008605033
```

```r
coef(backwards.2016.results$model, which.min(cv.err.2016.results$backward))
```

```
##                      (Intercept)                    pf_ss_homicide
##                      -0.06615198                        0.03439379
##     pf_ss_disappearances_violent              pf_movement_domestic
##                       0.04684146                        0.02868502
##              pf_movement_foreign          pf_religion_restrictions
##                       0.01014147                        0.06032529
##             pf_expression_killed             pf_expression_jailed
##                       0.01609829                       -0.02266957
##            pf_expression_control              pf_identity_sex_male
##                       0.11541950                        0.01008522
##           pf_identity_sex_female         ef_government_consumption
##                       0.01963255                        0.05319554
##                  ef_legal_courts                 ef_legal_military
##                       0.06756626                        0.05137188
##             ef_legal_enforcement                   ef_legal_gender
##                       0.03018399                        0.66724586
```

```
##                    ef_money_growth                          ef_money_sd
##                         0.04590314                           0.02938690
##                 ef_money_inflation                    ef_money_currency
##                         0.02701155                           0.02400962
##                ef_trade_tariffs_sd                     ef_trade_tariffs
##                        -0.05012341                           0.14627672
##    ef_trade_regulatory_compliance                        ef_trade_black
##                         0.03701143                          -0.02499725
##          ef_trade_movement_capital            ef_trade_movement_visit
##                         0.01457518                           0.01064147
##      ef_regulation_credit_private                 ef_regulation_credit
##                         0.02838766                           0.05137302
##       ef_regulation_labor_minwage           ef_regulation_labor_hours
##                         0.01445258                           0.01344164
## ef_regulation_labor_conscription       ef_regulation_business_start
##                         0.01236002                           0.04049172
```

```
coef(best.subset.2016.result$model,
which.min(cv.err.2016.results$best.subset))
```

```
##                        (Intercept)                       pf_ss_homicide
##                        -0.11305107                           0.04617284
##   pf_ss_disappearances_fatalities                 pf_movement_domestic
##                         0.05979903                           0.02962151
##           pf_religion_restrictions               pf_expression_control
##                         0.04554102                           0.12017279
##               pf_identity_sex_male           ef_government_consumption
##                         0.02790397                           0.05165267
##                    ef_legal_courts                    ef_legal_military
##                         0.05981440                           0.05790676
##               ef_legal_enforcement                      ef_legal_gender
##                         0.04609592                           0.85832531
##                    ef_money_growth                          ef_money_sd
##                         0.05616211                           0.05893255
##                  ef_money_currency               ef_trade_tariffs_mean
##                         0.03197152                           0.09269114
##                ef_trade_regulatory                 ef_regulation_credit
##                         0.05381921                           0.07000908
## ef_regulation_labor_conscription
##                         0.01325530
```

Ridge

```
ridge.2016 = do.ridge.lasso(hfi.2016.list$train,
                            hfi.2016.list$test,
                            alpha=0)

ridge.2016$model$beta
```

```
## 39 x 1 sparse Matrix of class "dgCMatrix"
##                                                 s0
```

```
## pf_ss_homicide                          0.035734748
## pf_ss_disappearances_disap              0.004725281
## pf_ss_disappearances_violent            0.029911748
## pf_ss_disappearances_fatalities         0.024858409
## pf_ss_disappearances_injuries          -0.006667508
## pf_movement_domestic                    0.022792489
## pf_movement_foreign                     0.012716102
## pf_religion_harassment                 -0.002868853
## pf_religion_restrictions                0.043219569
## pf_expression_killed                    0.011863627
## pf_expression_jailed                   -0.005631656
## pf_expression_influence                 0.041820631
## pf_expression_control                   0.062898920
## pf_identity_sex_male                    0.013541045
## pf_identity_sex_female                  0.015905050
## ef_government_consumption               0.030503990
## ef_legal_courts                         0.058168704
## ef_legal_military                       0.041267078
## ef_legal_enforcement                    0.039609134
## ef_legal_gender                         0.618162195
## ef_money_growth                         0.036702779
## ef_money_sd                             0.037252796
## ef_money_inflation                      0.020746559
## ef_money_currency                       0.019269927
## ef_trade_tariffs_mean                   0.032399543
## ef_trade_tariffs_sd                    -0.016363561
## ef_trade_tariffs                        0.064590360
## ef_trade_regulatory_compliance          0.020352165
## ef_trade_regulatory                     0.036538566
## ef_trade_black                         -0.002297200
## ef_trade_movement_capital               0.021269331
## ef_trade_movement_visit                 0.011715098
## ef_regulation_credit_private            0.022131942
## ef_regulation_credit                    0.033341906
## ef_regulation_labor_minwage             0.011092283
## ef_regulation_labor_hours               0.013501216
## ef_regulation_labor_conscription        0.008963149
## ef_regulation_business_start            0.025065147
## ef_regulation_business_compliance       0.005198406
```

```
ridge.2016$best.lambda
```

```
## [1] 0.1824993
```

```
ridge.2016$mse
```

```
## [1] 0.04762405
```

Lasso

```
lasso.2016 = do.ridge.lasso(hfi.2016.list$train,
                            hfi.2016.list$test,
```

```r
                            alpha=1)

rownames.no.na = c()
for (i in 1:length(rownames(lasso.2016$model$beta))){
  if( lasso.2016$model$beta[i] != 0){
    rownames.no.na = c(rownames.no.na, rownames(lasso.2016$model$beta)[i])
  }
}
lasso.2016[["rownames.left"]] = rownames.no.na

lasso.2016$model$beta
```

```
## 39 x 1 sparse Matrix of class "dgCMatrix"
##                                          s0
## pf_ss_homicide                  0.039296049
## pf_ss_disappearances_disap      .
## pf_ss_disappearances_violent    0.015573037
## pf_ss_disappearances_fatalities 0.028481889
## pf_ss_disappearances_injuries   .
## pf_movement_domestic            0.025828017
## pf_movement_foreign             0.001954904
## pf_religion_harassment          .
## pf_religion_restrictions        0.045998916
## pf_expression_killed            0.003754460
## pf_expression_jailed           -0.003289929
## pf_expression_influence         0.028811509
## pf_expression_control           0.090804032
## pf_identity_sex_male            0.013722979
## pf_identity_sex_female          0.016111395
## ef_government_consumption       0.032423338
## ef_legal_courts                 0.059824950
## ef_legal_military               0.049833060
## ef_legal_enforcement            0.038622209
## ef_legal_gender                 0.654462255
## ef_money_growth                 0.036769194
## ef_money_sd                     0.047200540
## ef_money_inflation              0.015287118
## ef_money_currency               0.023329299
## ef_trade_tariffs_mean           0.024122634
## ef_trade_tariffs_sd            -0.008620049
## ef_trade_tariffs                0.062613716
## ef_trade_regulatory_compliance  0.018049906
## ef_trade_regulatory             0.037150729
## ef_trade_black                  .
## ef_trade_movement_capital       0.018361139
## ef_trade_movement_visit         0.010775460
## ef_regulation_credit_private    0.016124082
## ef_regulation_credit            0.040102326
## ef_regulation_labor_minwage     0.007794979
## ef_regulation_labor_hours       0.009511282
```

```
## ef_regulation_labor_conscription   0.007714924
## ef_regulation_business_start       0.008494456
## ef_regulation_business_compliance  .
```

```
lasso.2016$best.lambda
```

```
## [1] 0.01028045
```

```
lasso.2016$mse
```

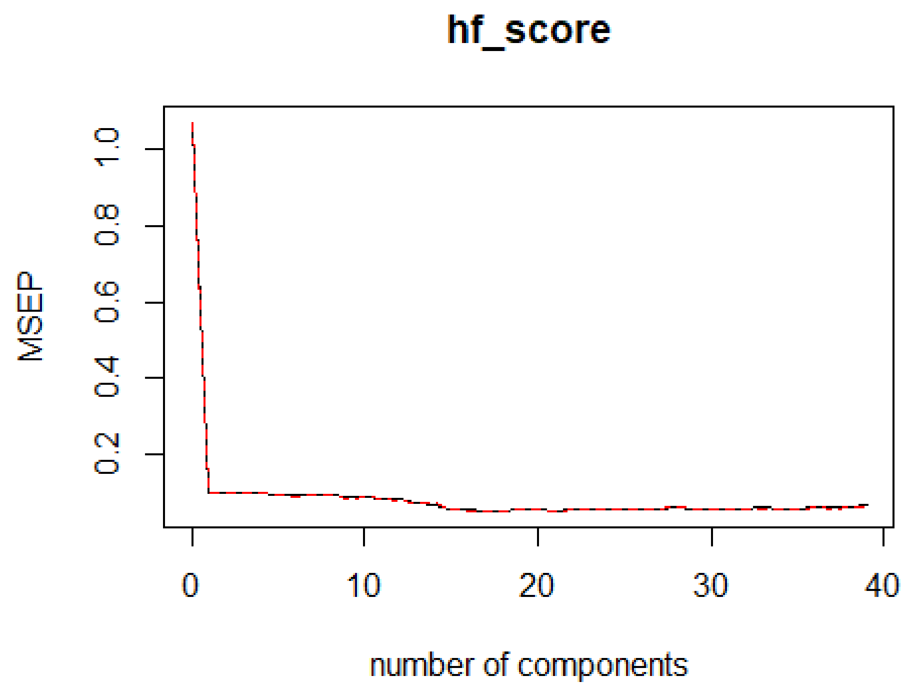
```
## [1] 0.04936473
```

```
lasso.2016$rownames.left
```

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

PCR

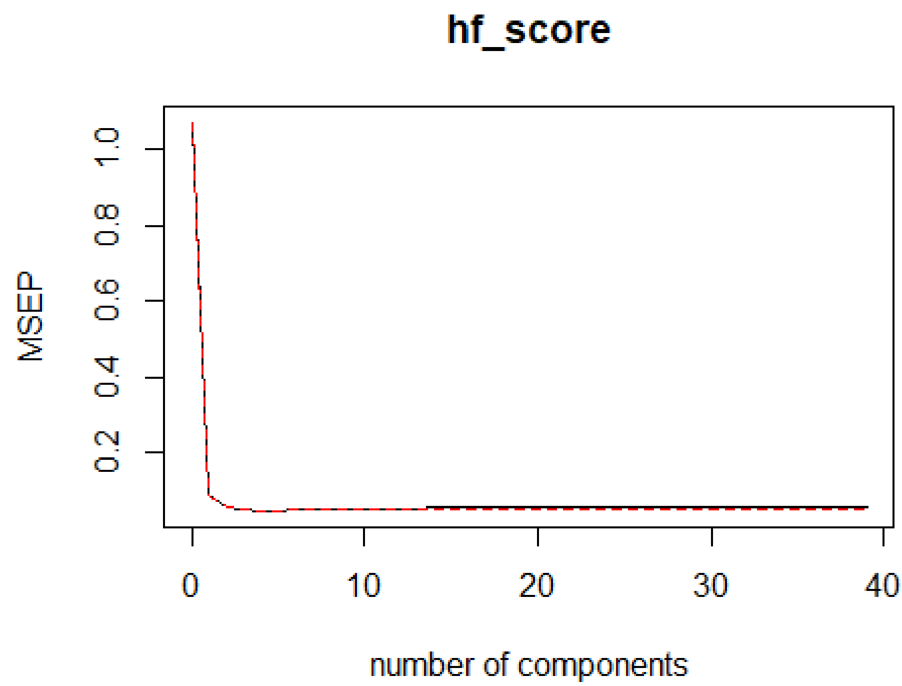
```
pcr.2016.model = pcr(hf_score ~ . -year ,  scale = TRUE,
                     data = hfi.2016.list$train, validation = "CV")
print(validationplot(pcr.2016.model, val.type = "MSEP"))
```




```
## NULL

pcr.2016.best = find.best.mse.pcr.pls(pcr.2016.model, hfi.2008.list$test)
pcr.2016.best$min.val

## [1] 28

pcr.2016.best$best.mse

## [1] 0.0598378
```

PLSR

```
# create pls model and plot validation; report the best mse
pls.2016.model = plsr(hf_score ~ . -year, scale = TRUE,
                 data = hfi.2016.list$train, validation = "CV")
validationplot(pls.2016.model, val.type = "MSEP")
```




```
pls.2016.best = find.best.mse.pcr.pls(pls.2016.model, hfi.2016.list$test)
pls.2016.best$min.val

## [1] 4

pls.2016.best$best.mse

## [1] 0.04046943
```

Matrix of Results 2008

```
x = matrix(data=
           c("Full OLS, 39 Predictors (test/train)",
             "Full OLS, 39 Predictors (10-fold CV)",
             paste0("Forward Select, ",
which.min(forward.2016.results$forward.mse), " Predictors (test/train)"),
             paste0("Forward Select, ",
which.min(cv.err.2016.results$forward), " Predictors (10-fold CV)"),
             paste0("Backward Select, ",
which.min(backwards.2016.results$forward.mse)," Predictors (test/train)"),
             paste0("Backward Select, ",
which.min(cv.err.2016.results$backward), " Predictors (10-fold CV)"),
             paste0("Best Subset, ",
which.min(best.subset.2016.result$best.subset.mse)," Predictors
(test/train)"),
             paste0("Best Subset, ",
which.min(cv.err.2016.results$best.subset), " Predictors (10-fold CV)"),
             paste0("Ridge, ", length(ridge.2016$model$beta), " Predictors
```

```r
(test/train)"),
             paste0("Lasso, ", length(lasso.2016$rownames.left), " Predictors
(test/train)"),
             paste0("PCR, ", pcr.2016.best$min.val ," Components,
(test/train)"),
             paste0("PLS, ", pls.2016.best$min.val, " Components,
(test/train)"),
             "Mean TSS",
mse.lm.2016,
cv.err.2016.results$full.lm,
min(forward.2016.results$forward.mse), # fit 3
min(cv.err.2016.results$forward),
min(backwards.2016.results$forward.mse),
min(cv.err.2016.results$backward),
min(best.subset.2016.result$best.subset.mse),
min(cv.err.2016.results$best.subset),
ridge.2016$mse,
lasso.2016$mse,
pcr.2016.best$best.mse,
pls.2016.best$best.mse,
hfi.2016.list$tss
), nrow=13, ncol=2)
colnames(x) <- c("Method","test MSE")
x
```

```
##       Method                                         test MSE
##  [1,] "Full OLS, 39 Predictors (test/train)"        "0.0781329447403669"
##  [2,] "Full OLS, 39 Predictors (10-fold CV)"        "0.0553283825978335"
##  [3,] "Forward Select, 18 Predictors (test/train)"  "0.0633411248684673"
##  [4,] "Forward Select, 26 Predictors (10-fold CV)"  "0.049528902434771"
##  [5,] "Backward Select, 31 Predictors (test/train)" "0.0742555612644516"
##  [6,] "Backward Select, 31 Predictors (10-fold CV)" "0.0503690551625668"
##  [7,] "Best Subset, 18 Predictors (test/train)"     "0.0383883618869735"
##  [8,] "Best Subset, 18 Predictors (10-fold CV)"     "0.0479349952900372"
##  [9,] "Ridge, 39 Predictors (test/train)"           "0.0476240470946796"
## [10,] "Lasso, 34 Predictors (test/train)"           "0.0493647261784953"
## [11,] "PCR, 28 Components, (test/train)"            "0.0598378004465714"
## [12,] "PLS, 4 Components, (test/train)"             "0.0404694328548725"
## [13,] "Mean TSS"                                    "1.01765845058487"
```

```r
df.x.all.2016 = as.data.frame(x, row.names = list.x.axis)
df.x.all.2016$`test MSE` = round(as.numeric(as.character(df.x.all.2016[["test
MSE"]])), 5)
df.x.all.2016
```

```
##                                          Method test MSE
## 1         Full OLS, 39 Predictors (test/train)  0.07813
## 2         Full OLS, 39 Predictors (10-fold CV)  0.05533
## 3   Forward Select, 18 Predictors (test/train)  0.06334
## 4   Forward Select, 26 Predictors (10-fold CV)  0.04953
```

```
## 5  Backward Select, 31 Predictors (test/train)  0.07426
## 6  Backward Select, 31 Predictors (10-fold CV)  0.05037
## 7      Best Subset, 18 Predictors (test/train)  0.03839
## 8      Best Subset, 18 Predictors (10-fold CV)  0.04793
## 9            Ridge, 39 Predictors (test/train)  0.04762
## 10           Lasso, 34 Predictors (test/train)  0.04936
## 11            PCR, 28 Components, (test/train)  0.05984
## 12             PLS, 4 Components, (test/train)  0.04047
## 13                                    Mean TSS  1.01766
```

```r
print("The 2008 Results:")
```

```
## [1] "The 2008 Results:"
```

```r
names(coef(forward.2008.results$model,
which.min(cv.err.2008.results$forward)))
```

```
##  [1] "(Intercept)"                        "pf_ss_homicide"
##  [3] "pf_movement_domestic"               "pf_religion_restrictions"
##  [5] "pf_expression_killed"               "pf_expression_influence"
##  [7] "pf_identity_sex_male"               "pf_identity_sex_female"
##  [9] "ef_government_consumption"          "ef_legal_courts"
## [11] "ef_legal_military"                  "ef_legal_enforcement"
## [13] "ef_legal_gender"                    "ef_money_growth"
## [15] "ef_money_sd"                        "ef_money_inflation"
## [17] "ef_money_currency"                  "ef_trade_tariffs_mean"
## [19] "ef_trade_regulatory_compliance" "ef_trade_movement_capital"
## [21] "ef_trade_movement_visit"            "ef_regulation_credit_private"
## [23] "ef_regulation_business_start"
```

```r
print("The 2016 Results:")
```

```
## [1] "The 2016 Results:"
```

```r
names(coef(forward.2016.results$model,
which.min(cv.err.2016.results$forward)))
```

```
##  [1] "(Intercept)"                        "pf_ss_homicide"
##  [3] "pf_ss_disappearances_violent"       "pf_movement_domestic"
##  [5] "pf_religion_restrictions"           "pf_expression_killed"
##  [7] "pf_expression_jailed"               "pf_expression_influence"
##  [9] "pf_expression_control"              "pf_identity_sex_female"
## [11] "ef_government_consumption"          "ef_legal_courts"
## [13] "ef_legal_military"                  "ef_legal_enforcement"
## [15] "ef_legal_gender"                    "ef_money_growth"
## [17] "ef_money_sd"                        "ef_money_currency"
## [19] "ef_trade_tariffs_sd"                "ef_trade_tariffs"
## [21] "ef_trade_regulatory"                "ef_trade_black"
## [23] "ef_trade_movement_capital"          "ef_trade_movement_visit"
## [25] "ef_regulation_credit_private"       "ef_regulation_credit"
## [27] "ef_regulation_labor_conscription"
```

```r
print("The Outlier Results:")
```

```
## [1] "The Outlier Results:"
```

```r
names(coef(forward.no.outlier.result$model,
which.min(cv.err.no.outlier.results$forward)))
```

```
##  [1] "(Intercept)"                         "pf_ss_homicide"
##  [3] "pf_ss_disappearances_disap"          "pf_ss_disappearances_violent"
##  [5] "pf_ss_disappearances_fatalities"     "pf_ss_disappearances_injuries"
##  [7] "pf_movement_domestic"                "pf_movement_foreign"
##  [9] "pf_religion_harassment"              "pf_religion_restrictions"
## [11] "pf_expression_killed"                "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_regulatory"
## [27] "ef_trade_movement_capital"           "ef_trade_movement_visit"
## [29] "ef_regulation_credit_private"        "ef_regulation_credit"
## [31] "ef_regulation_labor_minwage"         "ef_regulation_labor_hours"
## [33] "ef_regulation_labor_conscription" "ef_regulation_business_start"
```

```r
print("The No Outlier Results:")
```

```
## [1] "The No Outlier Results:"
```

```r
names(coef(forward.results$model, which.min(cv.err.results$forward)))
```

```
##  [1] "(Intercept)"
##  [2] "pf_ss_homicide"
##  [3] "pf_ss_disappearances_disap"
##  [4] "pf_ss_disappearances_violent"
##  [5] "pf_ss_disappearances_fatalities"
##  [6] "pf_ss_disappearances_injuries"
##  [7] "pf_movement_domestic"
##  [8] "pf_movement_foreign"
##  [9] "pf_religion_harassment"
## [10] "pf_religion_restrictions"
## [11] "pf_expression_killed"
## [12] "pf_expression_influence"
## [13] "pf_expression_control"
## [14] "pf_identity_sex_male"
## [15] "pf_identity_sex_female"
## [16] "ef_government_consumption"
## [17] "ef_legal_courts"
## [18] "ef_legal_military"
## [19] "ef_legal_enforcement"
## [20] "ef_legal_gender"
```

```
## [21] "ef_money_growth"
## [22] "ef_money_sd"
## [23] "ef_money_inflation"
## [24] "ef_money_currency"
## [25] "ef_trade_tariffs_mean"
## [26] "ef_trade_tariffs_sd"
## [27] "ef_trade_tariffs"
## [28] "ef_trade_regulatory"
## [29] "ef_trade_black"
## [30] "ef_trade_movement_capital"
## [31] "ef_trade_movement_visit"
## [32] "ef_regulation_credit_private"
## [33] "ef_regulation_credit"
## [34] "ef_regulation_labor_minwage"
## [35] "ef_regulation_labor_hours"
## [36] "ef_regulation_labor_conscription"
## [37] "ef_regulation_business_start"
## [38] "ef_regulation_business_compliance"
```