BA810 Team 3 - World Economic Freedom

Huiying Ba, He Chen, Zhaoying Chen, Tenisa Lee, Yiying Wang, Qifan Yang

For this project, we applied multiple model methods on our dataset "Economic Freedom of the World", in our dataset it includes worlds premier measurement ofstructure and security of property rights, access to sound money, freedom to tradeinternationally, and regulation of credit, labour and business economic freedom, ranking countries.

Set up

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------ tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts --------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(ggthemes)
library(scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```r
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loaded glmnet 2.0-18
```

```r
library(rpart)
library(rpart.plot)
theme_set(theme_bw())
```

Load and clean economic freedom dataset

```r
# load datasets
ef <- read.csv("economic_freedom.csv")

# Remove NA
cv<-colnames(ef)
cv1<-cv[c(4:36)]
for (i in 1:length(cv1)){
  Total_median<-median(purrr::flatten_dbl(ef[cv1[i]]), na.rm = TRUE)
  ef[cv1[i]][is.na(ef[cv1[i]])] <- Total_median
}

# selected columns
ef<-ef[,c(4, 7:10, 12:21, 23:26, 28:31, 33:35)]

# train ans test datasets
set.seed(1234)
ef$train <- sample(c(0,1), nrow(ef), replace = T, prob = c(.3, .7))
ef_test <- ef %>% filter(train == 0)
ef_train <- ef %>% filter(train == 1)

# data preparation
f1 <- as.formula(Economic_Freedom ~ government_consumption + transfers +
         gov_enterprises + top_marg_tax_rate + judicial_independence +
           impartial_courts + protection_property_rights +
```

```r
           military_interference + integrity_legal_system +
           legal_enforcement_contracts + restrictions_sale_real_property +
           reliability_police + business_costs_crime + gender_adjustment +
           money_growth + std_inflation + inflation +
           freedom_own_foreign_currency + tariffs + regulatory_trade_barriers +
           black_market + control_movement_capital_ppl + credit_market_reg +
           labor_market_reg + business_reg)
x_train <- model.matrix(f1, ef_train)[, -1]
y_train <- ef_train$Economic_Freedom
x_test <- model.matrix(f1, ef_test)[, -1]
y_test <- ef_test$Economic_Freedom
```

Model #1: Linear Regression

```r
# compute MSEs
fit_lm <- lm(f1, ef_train)
# MSE Train
yhat_train_lm <- predict(fit_lm)
mse_train_lm <- mean((y_train - yhat_train_lm)^2)
paste("Linear Regression Train MSE", mse_train_lm)
```

```
## [1] "Linear Regression Train MSE 0.087062753320789"
```

```r
# MSE Test
yhat_test_lm <- predict(fit_lm, ef_test)
mse_test_lm <- mean((y_test - yhat_test_lm)^2)
paste("Linear Regression Test MSE", mse_test_lm)
```

```
## [1] "Linear Regression Test MSE 0.0791379164238839"
```

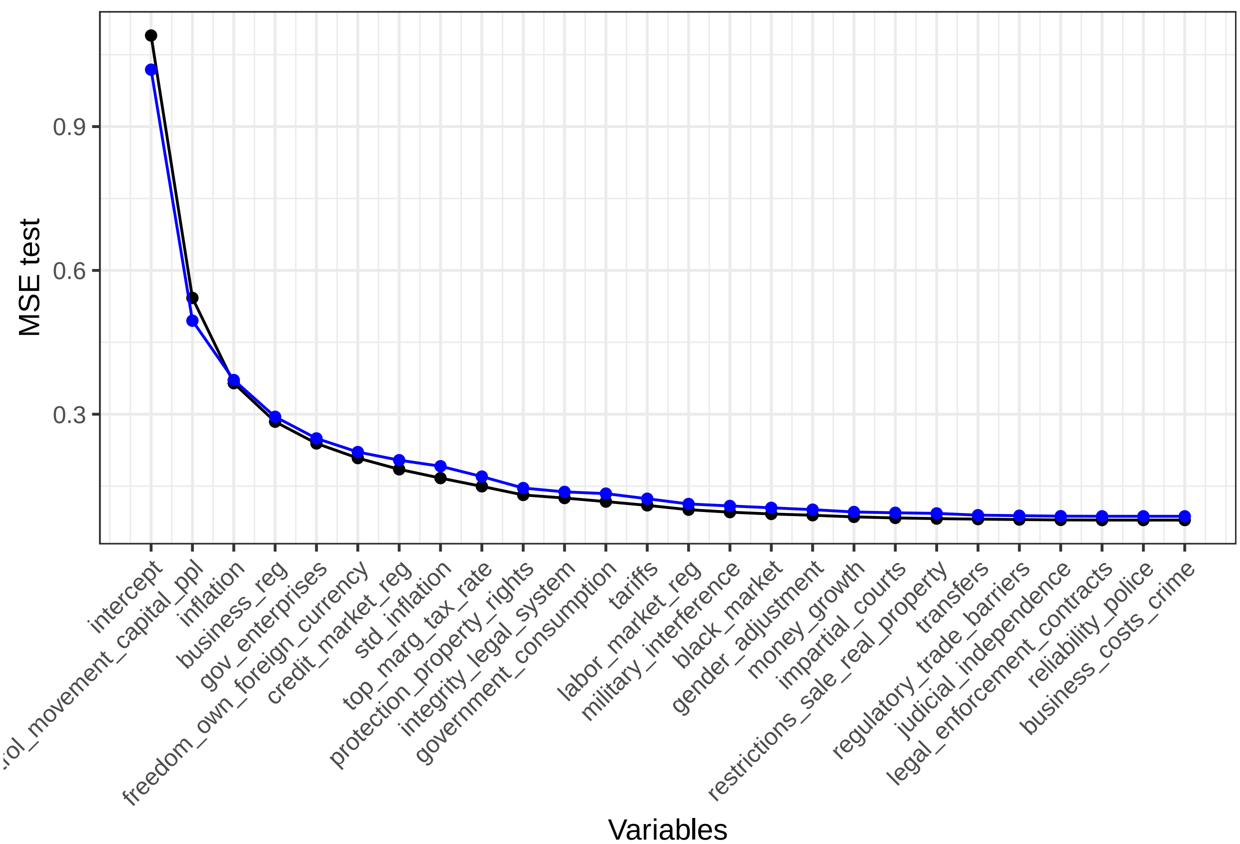
Model #2: Forward Selection

```r
xnames <- colnames(ef_train)
xnames <- xnames[!xnames %in% c("train", "Economic_Freedom","year","ISO_code","countries","rank")]
fit_fw <- lm(Economic_Freedom ~ 1, data = ef_train)
yhat_train <- predict(fit_fw, ef_train)
yhat_test <- predict(fit_fw, ef_test)
mse_train <- mean((ef_train$Economic_Freedom - yhat_train) ^ 2)
mse_test <- mean((ef_test$Economic_Freedom - yhat_test) ^ 2)
xname <- "intercept"
log_fw <-
  tibble(
    xname = xname,
    model = paste0(deparse(fit_fw$call), collapse = ""),
    mse_train = mse_train,
    mse_test = mse_test
  )
###
while (length(xnames) > 0) {
  best_mse_train <- NA
  best_mse_test <- NA
  best_fit_fw <- NA
```

```r
  best_xname <- NA
  # select the next best predictor
  for (xname in xnames) {
    # take a moment to examine and understand the following line
    fit_fw_tmp <- update(fit_fw, as.formula(paste0(". ~ . + ", xname)))
    # compute MSE train
    yhat_train_tmp <- predict(fit_fw_tmp, ef_train)
    mse_train_tmp <- mean((ef_train$Economic_Freedom - yhat_train_tmp) ^ 2)
    # compute MSE test
    yhat_test_tmp <- predict(fit_fw_tmp, ef_test)
    mse_test_tmp <- mean((ef_test$Economic_Freedom - yhat_test_tmp) ^ 2)
    # if this is the first predictor to be examined,
    # or if this predictors yields a lower MSE that the current
    # best, then store this predictor as the current best predictor
    if (is.na(best_mse_test) | mse_test_tmp < best_mse_test) {
      best_xname <- xname
      best_fit_fw <- fit_fw_tmp
      best_mse_train <- mse_train_tmp
      best_mse_test <- mse_test_tmp
    }
  }
  log_fw <-
    log_fw %>% add_row(
      xname = best_xname,
      model = paste0(deparse(best_fit_fw$call), collapse = ""),
      mse_train = best_mse_train,
      mse_test = best_mse_test
    )
  # adopt the best model for the next iteration
  fit_fw <- best_fit_fw

  # remove the current best predictor from the list of predictors
  xnames <- xnames[xnames!=best_xname]
}

ggplot(log_fw, aes(seq_along(xname), mse_test)) +
  geom_point() +
  geom_line() +
  geom_point(aes(y=mse_train), color="blue") +
  geom_line(aes(y=mse_train), color="blue") +
  scale_x_continuous("Variables", labels = log_fw$xname, breaks = seq_along(log_fw$xname)) +
  scale_y_continuous("MSE test") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
print(log_fw)
```

```
## # A tibble: 26 x 4
##    xname             model                                  mse_train mse_test
##    <chr>             <chr>                                      <dbl>    <dbl>
##  1 intercept         lm(formula = Economic_Freedom ~ 1, ~       1.02     1.09
##  2 control_movemen~ lm(formula = Economic_Freedom ~ con~       0.495    0.542
##  3 inflation         lm(formula = Economic_Freedom ~ con~       0.371    0.365
##  4 business_reg      lm(formula = Economic_Freedom ~ con~       0.295    0.284
##  5 gov_enterprises   lm(formula = Economic_Freedom ~ con~       0.250    0.239
##  6 freedom_own_for~ lm(formula = Economic_Freedom ~ con~       0.221    0.208
##  7 credit_market_r~ lm(formula = Economic_Freedom ~ con~       0.204    0.185
##  8 std_inflation     lm(formula = Economic_Freedom ~ con~       0.192    0.167
##  9 top_marg_tax_ra~ lm(formula = Economic_Freedom ~ con~       0.170    0.150
## 10 protection_prop~ lm(formula = Economic_Freedom ~ con~       0.146    0.132
## # ... with 16 more rows
```

Model #3: Backward Selection

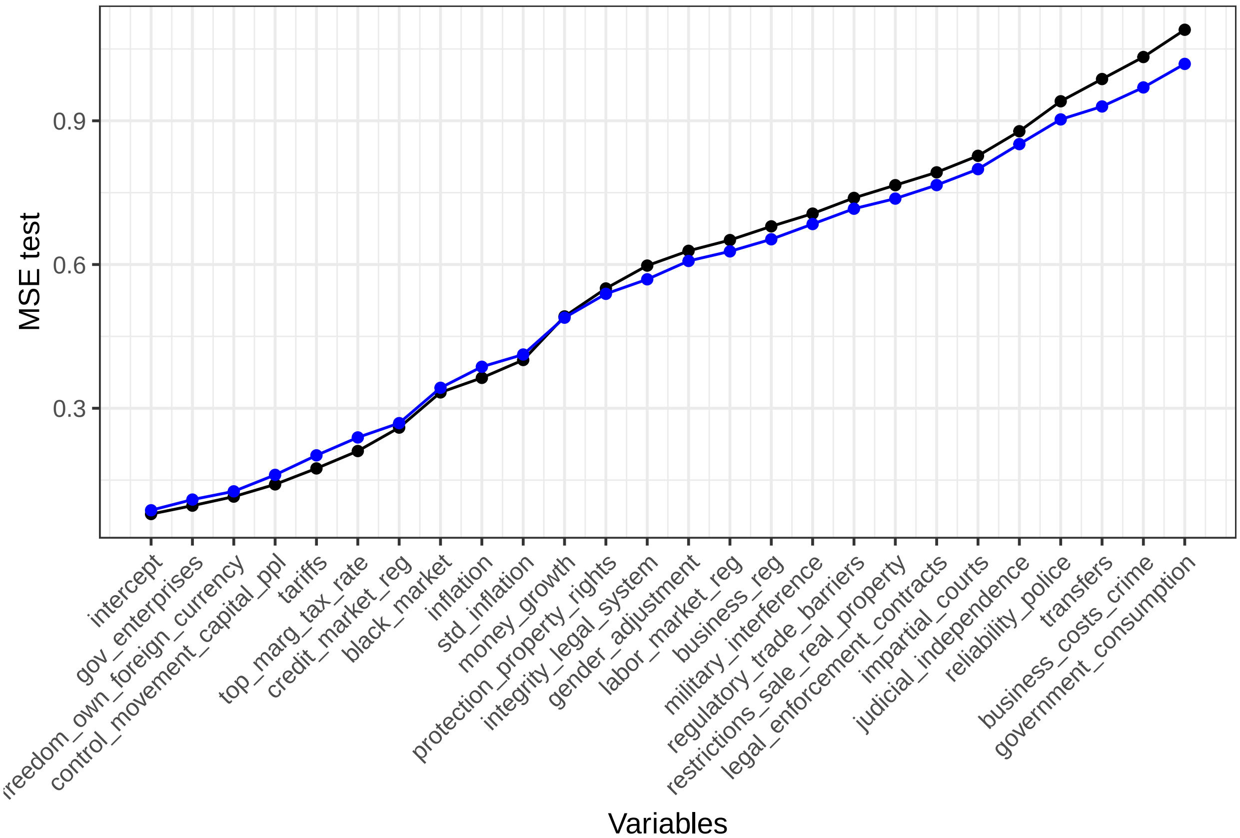
```r
xnames <- colnames(ef_train)
xnames <- xnames[!xnames %in% c("train", "Economic_Freedom","year","ISO_code","countries","rank")]
fit_bk <- lm(Economic_Freedom ~ ., data = ef_train)
yhat_train_bk <- predict(fit_bk, ef_train)
yhat_test_bk <- predict(fit_bk, ef_test)
mse_train_bk <- mean((ef_train$Economic_Freedom - yhat_train_bk) ^ 2)
mse_test_bk <- mean((ef_test$Economic_Freedom - yhat_test_bk) ^ 2)
```

```r
xname <- "intercept"
log_bk <-
  tibble(
    xname = xname,
    model = paste0(deparse(fit_bk$call), collapse = ""),
    mse_train_bk = mse_train_bk,
    mse_test_bk = mse_test_bk
  )
###
while (length(xnames) > 0) {
  best_mse_train_bk <- NA
  best_mse_test_bk <- NA
  best_fit_bk <- NA
  best_xname_bk <- NA
  # select the next best predictor
  for (xname in xnames) {
    # take a moment to examine and understand the following line
    fit_bk_tmp <- update(fit_bk, as.formula(paste0(". ~ . - ", xname)))
    # compute MSE train
    yhat_train_tmp_bk <- predict(fit_bk_tmp, ef_train)
    mse_train_tmp_bk <- mean((ef_train$Economic_Freedom - yhat_train_tmp_bk) ^ 2)
    # compute MSE test
    yhat_test_tmp_bk <- predict(fit_bk_tmp, ef_test)
    mse_test_tmp_bk <- mean((ef_test$Economic_Freedom - yhat_test_tmp_bk) ^ 2)
    # if this is the first predictor to be examined,
    # or if this predictors yields a lower MSE that the current
    # best, then store this predictor as the current best predictor
    if (is.na(best_mse_test_bk) | mse_test_tmp_bk > best_mse_test_bk) {
      best_xname_bk <- xname
      best_fit_bk <- fit_bk_tmp
      best_mse_train_bk <- mse_train_tmp_bk
      best_mse_test_bk <- mse_test_tmp_bk
    }
  }
  log_bk <-
    log_bk %>% add_row(
      xname = best_xname_bk,
      model = paste0(deparse(best_fit_bk$call), collapse = ""),
      mse_train_bk = best_mse_train_bk,
      mse_test_bk = best_mse_test_bk
    )
  # adopt the best model for the next iteration
  fit_bk <- best_fit_bk

  # remove the current best predictor from the list of predictors
  xnames <- xnames[xnames!=best_xname_bk]
}

ggplot(log_bk, aes(seq_along(xname), mse_test_bk)) +
  geom_point() +
  geom_line() +
  geom_point(aes(y=mse_train_bk), color="blue") +
  geom_line(aes(y=mse_train_bk), color="blue") +
```

```r
  scale_x_continuous("Variables", labels = log_bk$xname, breaks = seq_along(log_bk$xname)) +
  scale_y_continuous("MSE test") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
print(log_bk)
```

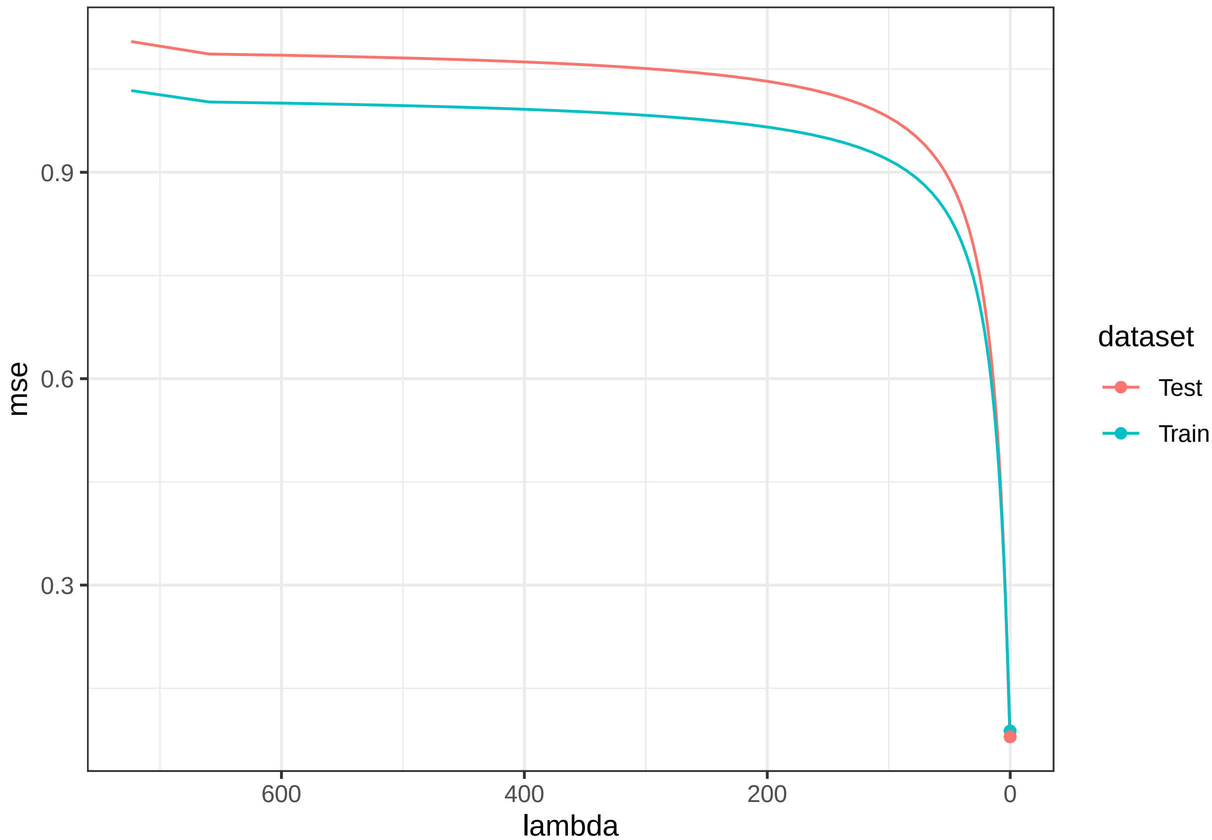
```
## # A tibble: 26 x 4
##    xname           model                               mse_train_bk mse_test_bk
##    <chr>           <chr>                                      <dbl>       <dbl>
##  1 intercept       lm(formula = Economic_Freedom ~           0.0871      0.0791
##  2 gov_enterprises lm(formula = Economic_Freedom ~           0.109       0.0967
##  3 freedom_own_fo~ lm(formula = Economic_Freedom ~           0.127       0.116
##  4 control_moveme~ lm(formula = Economic_Freedom ~           0.161       0.141
##  5 tariffs         lm(formula = Economic_Freedom ~           0.202       0.174
##  6 top_marg_tax_r~ lm(formula = Economic_Freedom ~           0.239       0.211
##  7 credit_market_~ lm(formula = Economic_Freedom ~           0.269       0.260
##  8 black_market    lm(formula = Economic_Freedom ~           0.343       0.333
##  9 inflation       lm(formula = Economic_Freedom ~           0.387       0.363
## 10 std_inflation   lm(formula = Economic_Freedom ~           0.412       0.401
## # ... with 16 more rows
```

Model #4: Ridge Regression

```r
##lambda
est_r <- glmnet(x_train, y_train, alpha = 0, nlambda = 100)
y_train_hat_r <- predict(est_r,x_train)
```

```r
#y_train_hat
y_test_hat_r <- predict(est_r, x_test)
#y_test_hat
lm<-function(x,y) {
  mean((x-y)^2)
}
# write code to create a vector that contains 100 MSE estimates for the train data
#use apply() to use the funciton created
mse_train_r<-apply(y_train_hat_r,2,lm,y=y_train)
# write code to create a vector that contains 100 MSE estimates for the test data
#use apply() to use the funciton created
mse_test_r<-apply(y_test_hat_r,2,lm,y=y_test)
lambda_min_mse_train_r <- mse_train_r[which.min(mse_train_r)]
lambda_min_mse_test_r <- mse_test_r[which.min(mse_test_r)]
# create a tibble of train MSEs and lambdas
ef_mse_r <- tibble(
  lambda = est_r$lambda,
  mse = mse_train_r,
  dataset = "Train"
)
ef_mse_r<- rbind(ef_mse_r, tibble(
  lambda = est_r$lambda,
  mse = mse_test_r,
  dataset = "Test"
))
# Use the rbind command to combine dd_mse_train
# and dd_mse_test into a single data frame

ef_mse_r %>%
  ggplot(aes(x=lambda,y=mse,color=dataset))+
  geom_line()+
  #reverse x scale
  scale_x_reverse()+
  geom_point(data =filter(ef_mse_r,dataset=="Train"),aes(x=lambda[which.min(mse)],y=min(mse)))+
  geom_point(data =filter(ef_mse_r,dataset=="Test"),aes(x=lambda[which.min(mse)],y=min(mse)))
```




```r
paste("Ridge Regression Train MSE",lambda_min_mse_train_r)
```

```
## [1] "Ridge Regression Train MSE 0.0879899894085094"
```

```r
paste("Ridge Regression Test MSE",lambda_min_mse_test_r)
```

```
## [1] "Ridge Regression Test MSE 0.0795354198496345"
```

```r
coef(est_r,lambda_min_mse_test_r)
```

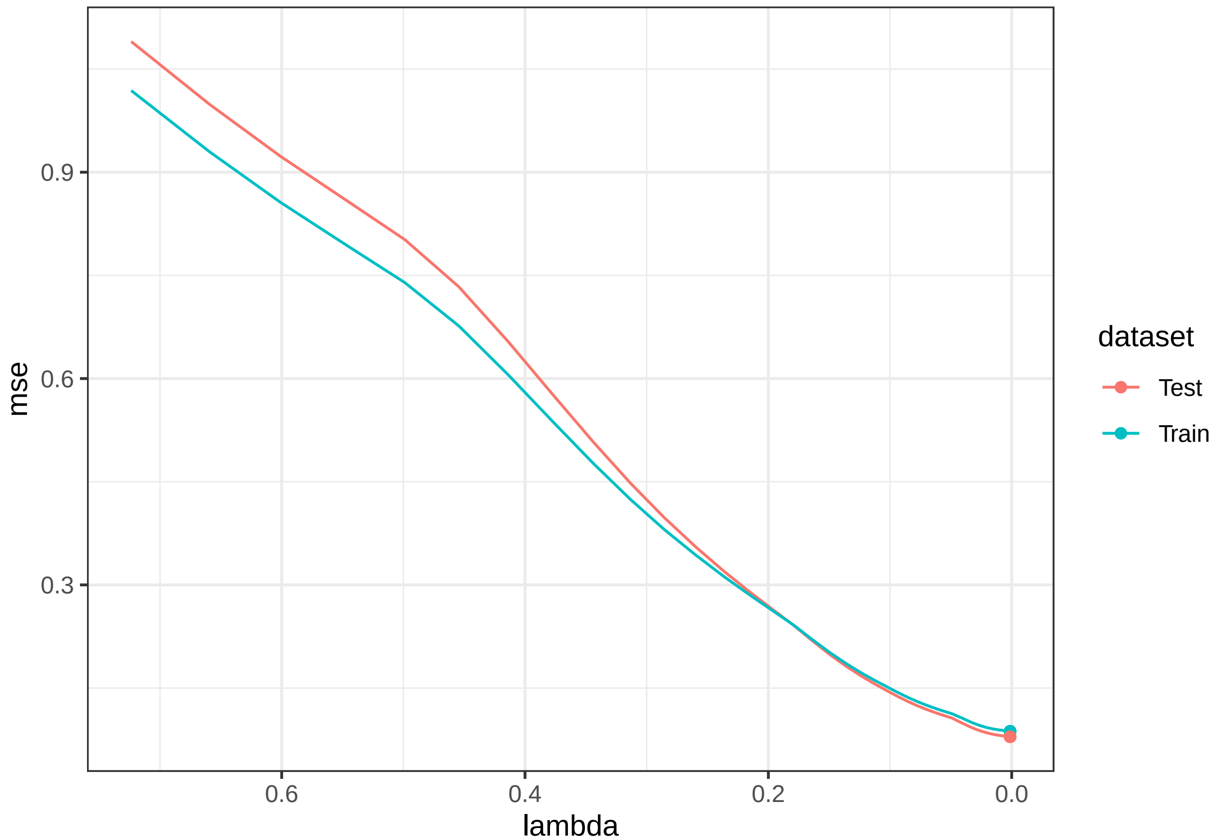
```
## 26 x 1 sparse Matrix of class "dgCMatrix"
##                                          1
## (Intercept)                  -0.4767707505
## government_consumption        0.0387165241
## transfers                     0.0413675869
## gov_enterprises               0.0550088282
## top_marg_tax_rate             0.0591281859
## judicial_independence        -0.0182410007
## impartial_courts              0.0471120047
## protection_property_rights    0.0728791890
## military_interference         0.0307395449
## integrity_legal_system        0.0480500529
## legal_enforcement_contracts   0.0176487526
## restrictions_sale_real_property 0.0199843659
## reliability_police            0.0019232212
## business_costs_crime          0.0001644325
## gender_adjustment             0.4412616118
## money_growth                  0.0571091355
## std_inflation                 0.0465256359
```

```
## inflation                          0.0672758717
## freedom_own_foreign_currency       0.0420462271
## tariffs                            0.0842888833
## regulatory_trade_barriers          0.0332767909
## black_market                       0.0390793442
## control_movement_capital_ppl       0.0617214107
## credit_market_reg                  0.0349152447
## labor_market_reg                   0.0718718254
## business_reg                       0.0498227561
```

Model #5: Lasso Regression

```r
##lambda
est_l <- glmnet(x_train, y_train, alpha = 1, nlambda = 100)
y_train_hat_l <- predict(est_l,x_train)
#y_train_hat
y_test_hat_l <- predict(est_l, x_test)
#y_test_hat
lm<-function(x,y) {
  mean((x-y)^2)
}
# write code to create a vector that contains 100 MSE estimates for the train data
#use apply() to use the funciton created
mse_train_l<-apply(y_train_hat_l,2,lm,y=y_train)
# write code to create a vector that contains 100 MSE estimates for the test data
#use apply() to use the funciton created
mse_test_l<-apply(y_test_hat_l,2,lm,y=y_test)
lambda_min_mse_train_l <- mse_train_l[which.min(mse_train_l)]
lambda_min_mse_test_l <- mse_test_l[which.min(mse_test_l)]
# create a tibble of train MSEs and lambdas
ef_mse_l <- tibble(
  lambda = est_l$lambda,
  mse = mse_train_l,
  dataset = "Train"
)
ef_mse_l<- rbind(ef_mse_l, tibble(
  lambda = est_l$lambda,
  mse = mse_test_l,
  dataset = "Test"
))
# Use the rbind command to combine dd_mse_train
# and dd_mse_test into a single data frame

ef_mse_l %>%
  ggplot(aes(x=lambda,y=mse,color=dataset))+
  geom_line()+
  #reverse x scale
  scale_x_reverse()+
  geom_point(data =filter(ef_mse_l,dataset=="Train"),aes(x=lambda[which.min(mse)],y=min(mse)))+
  geom_point(data =filter(ef_mse_l,dataset=="Test"),aes(x=lambda[which.min(mse)],y=min(mse)))
```




```r
paste("Lasso Regression Train MSE",lambda_min_mse_train_l)
```

```
## [1] "Lasso Regression Train MSE 0.0871182804528582"
```

```r
paste("Lasso Regression Test MSE",lambda_min_mse_test_l)
```

```
## [1] "Lasso Regression Test MSE 0.0791935605686631"
```

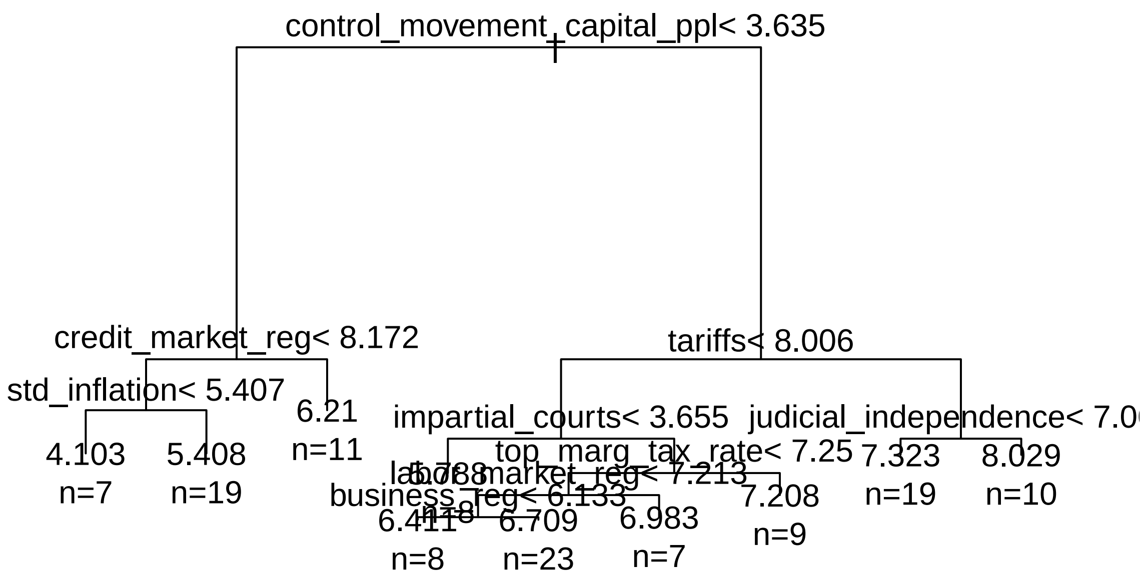
```r
coef(est_l,lambda_min_mse_test_l)
```

```
## 26 x 1 sparse Matrix of class "dgCMatrix"
##                                         1
## (Intercept)                   1.734312722
## government_consumption        .
## transfers                     .
## gov_enterprises               0.045957841
## top_marg_tax_rate             0.043816147
## judicial_independence         .
## impartial_courts              0.010788906
## protection_property_rights    0.062166248
## military_interference         0.013544384
## integrity_legal_system        0.024606953
## legal_enforcement_contracts   .
## restrictions_sale_real_property .
## reliability_police            .
## business_costs_crime          .
## gender_adjustment             0.096543386
## money_growth                  0.038467526
## std_inflation                 0.043583380
```

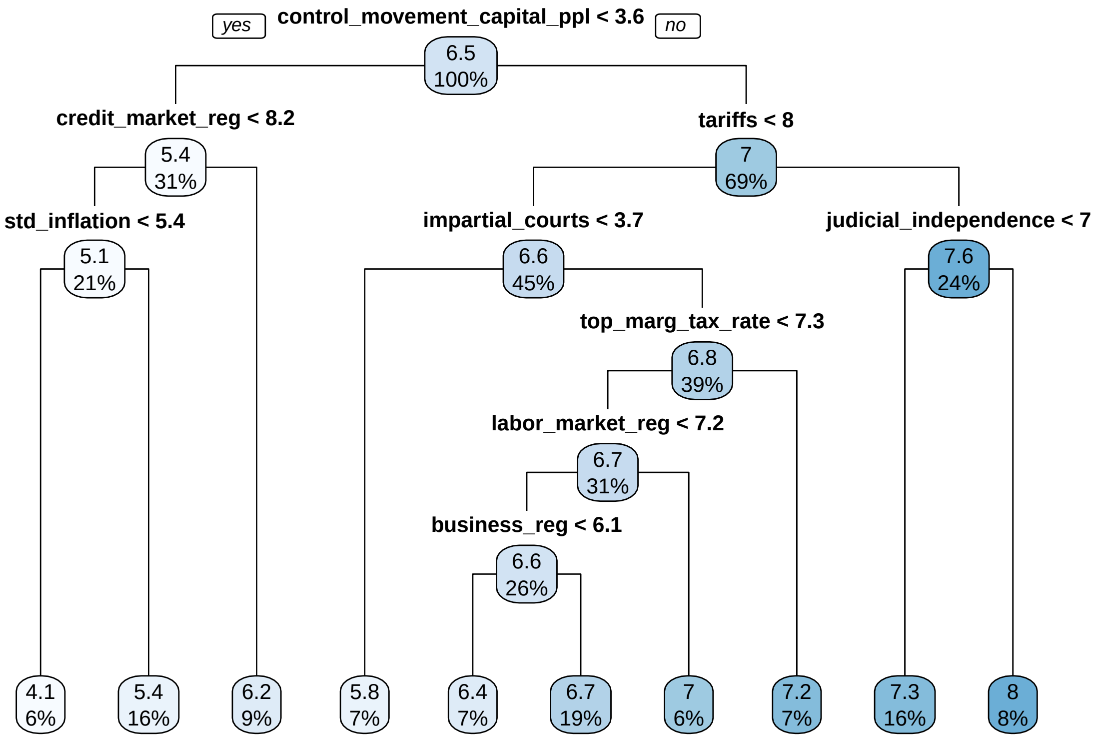
```
## inflation                          0.058280222
## freedom_own_foreign_currency       0.034724277
## tariffs                            0.063780927
## regulatory_trade_barriers          0.008677913
## black_market                       0.024852400
## control_movement_capital_ppl       0.081403888
## credit_market_reg                  0.037222950
## labor_market_reg                   0.050240637
## business_reg                       0.079433172
```

Model #6: Regression Tree

```r
ef_test_rt <- ef_test %>%
  filter(sample(c(0,1),nrow(ef_test),replace=TRUE,prob=c(0.95,0.05))==1)
ef_train_rt <- ef_train %>%
  filter(sample(c(0,1),nrow(ef_train),replace=TRUE,prob=c(0.95,0.05))==1)
fit.tree <- rpart(f1,
                  ef_train_rt,
                  control = rpart.control(cp = 0.001))
par(xpd = TRUE)
plot(fit.tree, compress=TRUE)
text(fit.tree, use.n=TRUE)
```




```r
rpart.plot(fit.tree, type = 1)
```




```r
yhat.train.tree <- predict(fit.tree, ef_train_rt)
mse.train.tree <- mean((ef_train_rt$Economic_Freedom - yhat.train.tree)^2)
yhat.test.tree <- predict(fit.tree, ef_test_rt)
mse.test.tree <- mean((ef_test_rt$ Economic_Freedom- yhat.test.tree)^2)

paste("Regression Tree Train MSE",mse.train.tree)
```

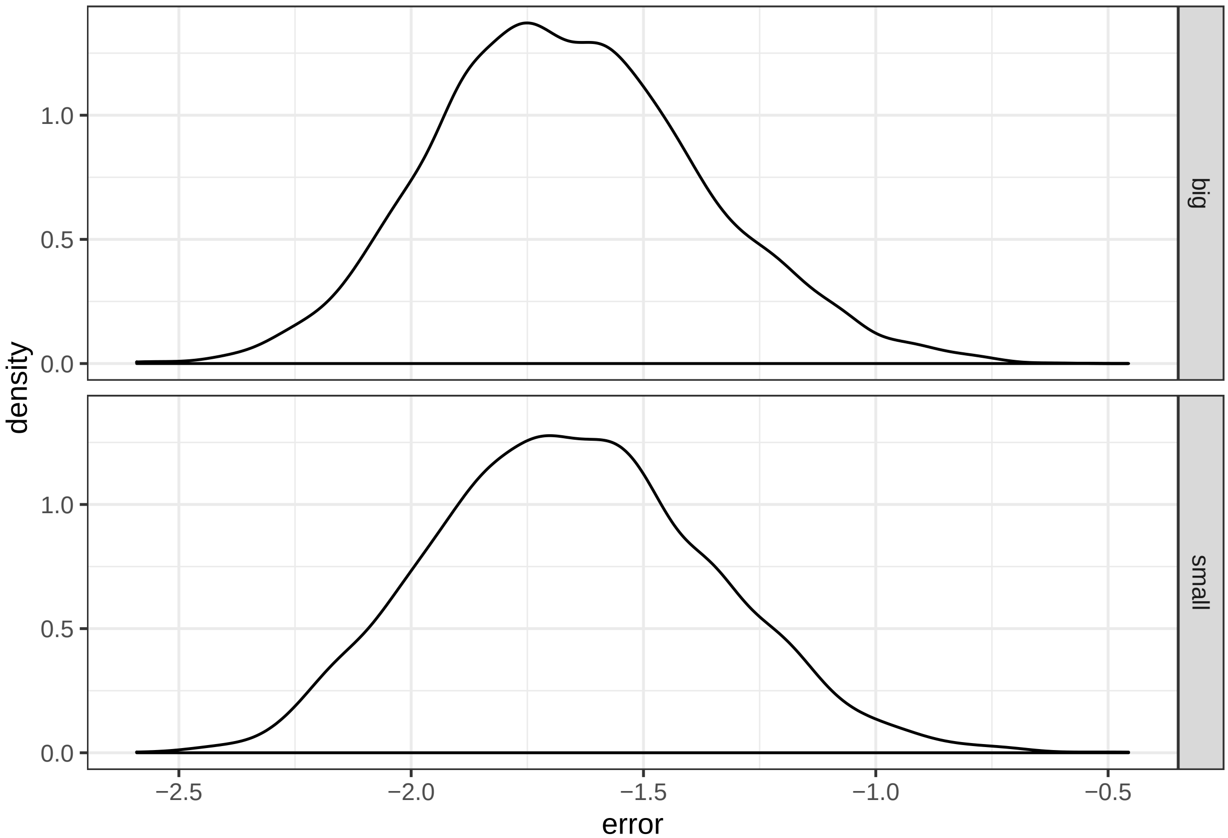
```
## [1] "Regression Tree Train MSE 0.213845395019018"
```

```r
paste("Regression Tree Test MSE",mse.test.tree)
```

```
## [1] "Regression Tree Test MSE 0.456252647730941"
```

```r
#bias and variance trade-off with trees
x0 <- ef_train_rt[1,]
ef_train_1 <- ef_train_rt[-1,]
yhat_small_tree <- c()
for (i in seq(3726)) {
  fit_tree <- rpart(f1,
                    ef_train_1 %>% sample_frac(size = .1),
                    control = rpart.control(cp = 0.001))
  yhat <- predict(fit_tree, x0)
  yhat_small_tree <- c(yhat_small_tree, yhat)
}
yhat_big_tree <- c()
for (i in seq(3726)) {
  fit_tree <- rpart(f1,
                    ef_train_1 %>% sample_frac(size = .1),
                    control = rpart.control(cp = 0.0001))
  yhat <- predict(fit_tree, x0)
  yhat_big_tree <- c(yhat_big_tree, yhat)
}
```

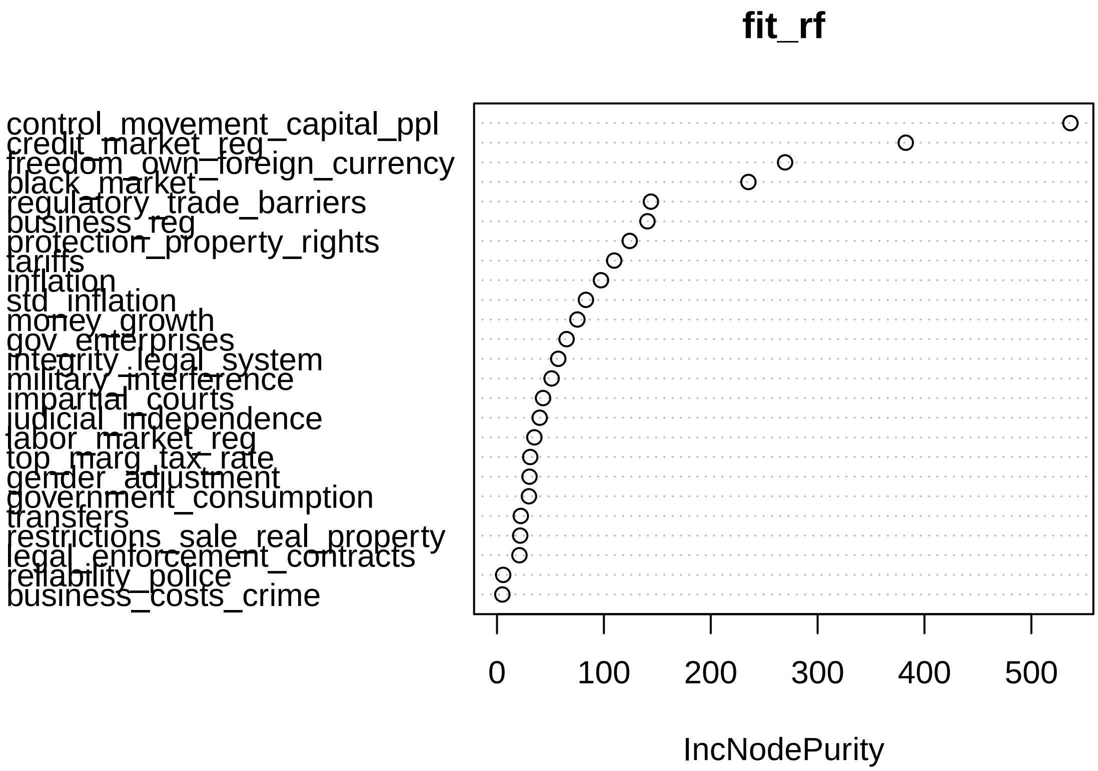
```r
errors <- data.frame(
  "error"= (x0$Economic_Freedom - c(yhat_small_tree, yhat_big_tree)),
  "flexibility"= c(rep("small", length(yhat_small_tree)), rep("big",length(yhat_big_tree))))
ggplot(errors,aes(error)) +geom_density()+facet_grid(flexibility~.)
```



Model #7: Random Forests

```r
fit_rf <- randomForest(f1, ef_train, ntree = 500, do.trace=F)
yhat_rf_train <- predict(fit_rf, ef_train)
mse_rf_train <- mean((yhat_rf_train - y_train) ^2)
yhat_rf_test <- predict(fit_rf, ef_test)
mse_rf_test <- mean((yhat_rf_test - y_test) ^2)

varImpPlot(fit_rf)
```




```r
paste("Random Forest Train MSE",mse_rf_train)
```

```
## [1] "Random Forest Train MSE 0.0147101501821982"
```

```r
paste("Random Forest Test MSE",mse_rf_test)
```

```
## [1] "Random Forest Test MSE 0.0773669077844378"
```

Model #8: Boosted Trees

```r
fit_btree <- gbm(f1, data = ef_train, distribution = "gaussian",
                 n.trees = 100, interaction.depth = 2, shrinkage = 0.001)
relative.influence(fit_btree)
```

```
## n.trees not given. Using 100 trees.

##          government_consumption                        transfers
##                          0.0000                           0.0000
##                 gov_enterprises                  top_marg_tax_rate
##                          0.0000                           0.0000
##            judicial_independence                  impartial_courts
##                          0.0000                           0.0000
##       protection_property_rights            military_interference
##                        461.4666                           0.0000
##           integrity_legal_system      legal_enforcement_contracts
##                          0.0000                           0.0000
## restrictions_sale_real_property               reliability_police
##                          0.0000                           0.0000
##             business_costs_crime                gender_adjustment
##                          0.0000                           0.0000
```

```
##                    money_growth                    std_inflation
##                          0.0000                           0.0000
##                       inflation    freedom_own_foreign_currency
##                          0.0000                       13343.7084
##                         tariffs       regulatory_trade_barriers
##                          0.0000                        1474.9568
##                    black_market    control_movement_capital_ppl
##                          0.0000                       43541.6519
##               credit_market_reg                 labor_market_reg
##                       3971.4072                           0.0000
##                    business_reg
##                          0.0000
```

```r
yhat_btree_train <- predict(fit_btree, ef_train, n.trees = 100)
mse_btree_train <- mean((yhat_btree_train - y_train) ^ 2)
yhat_btree_test <- predict(fit_btree, ef_test, n.trees = 100)
mse_btree_test <- mean((yhat_btree_test - y_test) ^ 2)

paste("Boosted Trees Train MSE",mse_btree_train)
```

```
## [1] "Boosted Trees Train MSE 0.924382854466046"
```

```r
paste("Boosted Trees Test MSE",mse_btree_test)
```

```
## [1] "Boosted Trees Test MSE 0.990916266247947"
```