MSD 2019 Final Project

A replication and extension of Comparing Random Forest with Logistic Regression for Predicting Class-Imbalanced Civil War Onset Data by David Muchlinski, David Siroky, et. al., October 22, 2015

Deniz Ulcay, Vatsala Swaroop, Ongun Uzay Macar (du2147, vs2671, oum2000)

2019-05-12 05:35:49

Contents

Introduction **2**
Motivation 2
Paper Description 2
Replication & Implementation Description 2

Data Exploration **2**
Importing Datasets 2
Dataset Exploration and Comparison 3
Data Visualization 3
Data Insights and Moving Forward 5
Exploration of Class Imbalance in Datasets 6

Model Specifications **7**
Feature Selection Based on Model Specifications 7

Data Processing **8**
Removing Unused Columns 8
Training/Test Split for Accurate Assessment 8
SMOTE 9

Model Training **9**
Training Setup 9
Replicating Author's Training Processes 10
Training Insights 11
Our Own Training Processes 12
Experimenting with Different Models 15

Variable Importance for Random Forest Models **16**
Replicating Author's Variable Importance Visualizations 16
Retraining Random Forest Models 17
Our Own Variable Importance Visualizations 18

Model Testing **21**
Testing Insights 21
Replicating Author's Prediction Processes 21
Our Own Prediction Processes 22
Confusion Matrices 23
Replicating Author's Seperation Plots 25
Our Own Seperation Plots 27
Replicating Author's ROC Curves with AUC Scores 34
ROC Curves with AUC Scores 36
Grouped By Specifications 37

Grouped By Model Types 41

Dependencies Summary **44**

References **46**
Related Papers & Datasets 46
StackOverflow for the Rescue! 46

Introduction

Motivation

Prediction is at the heart of many machine learning and data science applications, and its importance is amplified in the context of political science. Especially for the case of civil war onset, robust models that can make correct predictions has the potential to save millions of lives and guide political agendas for the years to come.

Paper Description

Comparing Random Forest with Logistic Regression for Predicting Class-Imbalanced Civil War Onset Data by David Muchlinski, David Siroky, et. al. is an explaratory research paper that, among several other things, makes the argument that most well-known logistic models acquire relatively lower predictive powers for civil war onsets, and shows that a custom random forest model achieves much higer prediction accuracies.

Replication & Implementation Description

The replication for the paper mentioned will consist of two main parts, which will go parallel to each other as represented in this R Notebook. The first part will deal with extracting code snippets from the original R code (found in `original_code/`), and questioning the reasonability and correctness of the methods and code semantics used, as well as an effort to see if results can be completely replicated. The second part will deal with suggesting improvements, modifications, and eventually corrections to the original R code, where we will try to report summaries of each model using fair metrics and correct methodologies. Comments included with double hashcodes (`##`) corresponds to comments made by authors themselves.

Later, a third part will deal with suggesting new models like neural networks, custom decision trees, and boosting algorithms.

Data Exploration

Importing Datasets

```r
# HS_original: Civil War Data by Hegre and Sambanis (2006), the original version
data_HS_original <- read.dta(file="data/Sambanis (Aug 06).dta")
# HS_cleaned: Civil War Data by Hegre and Sambanis (2006), NAs eliminated version
data_HS_cleaned <- na.omit(data_HS_original)
# HS_imputed: Civil War Data by Hegre and Sambanis (2006), imputed by authors
data_HS_imputed <- read.csv(file="data/SambnisImp.csv") ## data for prediction
# AM_imputed: Amelia dataset imputed by authors
data_AM_imputed <- read.csv(file="data/Amelia.Imp3.csv") ## data for causal machanisms
```

```r
# AF_imputed: Africa dataset imputed by authors
data_AF_imputed <- read.csv(file="data/AfricaImp.csv")
```

Dataset Exploration and Comparison

```r
data_presentation <- matrix(c(ncol(data_HS_original), sum(is.na(data_HS_original)),
                              nrow(data_HS_original),
                              ncol(data_HS_cleaned), sum(is.na(data_HS_cleaned)),
                              nrow(data_HS_cleaned),
                              ncol(data_HS_imputed), sum(is.na(data_HS_imputed)),
                              nrow(data_HS_imputed),
                              ncol(data_AM_imputed), sum(is.na(data_AM_imputed)),
                              nrow(data_AM_imputed),
                              ncol(data_AF_imputed), sum(is.na(data_AF_imputed)),
                              nrow(data_AF_imputed)), ncol=5)
colnames(data_presentation) <- c('HS_original', 'HS_cleaned',
                                 'HS_imputed', 'AM_imputed', 'AF_imputed')
rownames(data_presentation) <- c('No. features', 'No. empty cells', 'No. examples')
as.data.frame(data_presentation)
```

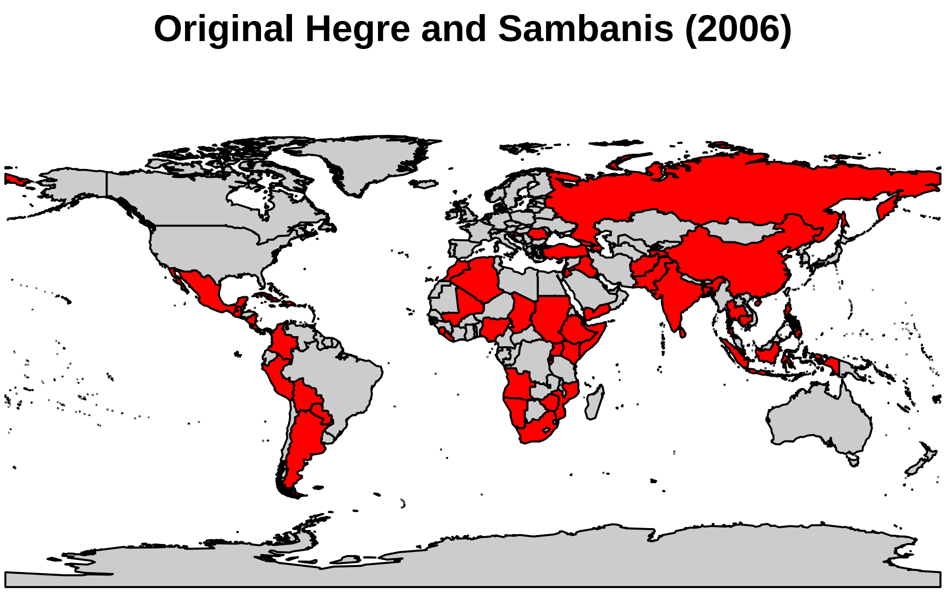
```
##                 HS_original HS_cleaned HS_imputed AM_imputed AF_imputed
## No. features            284        284        286         53         11
## No. empty cells      979981          0          0        778          0
## No. examples           9691          0       7140       7141        737
```

```r
# Check intersection and difference of features on two datasets
# setdiff(colnames(data_HS_imputed), colnames(data_HS_original))
# length(intersect(colnames(data_HS_imputed), colnames(data_HS_original)))
# intersect(colnames(data_HS_imputed), colnames(data_AF_imputed))
```

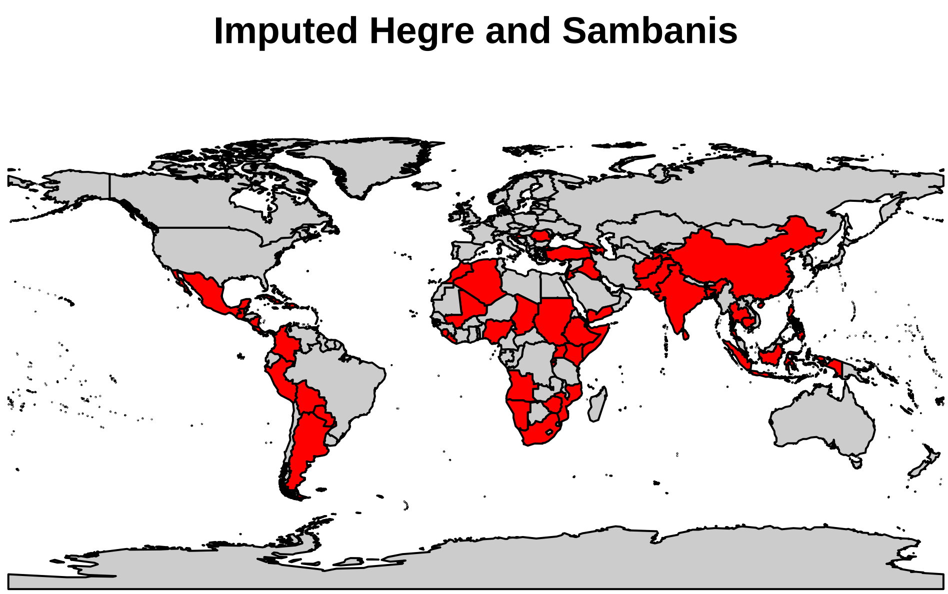
Data Visualization

```r
# Function for converting a specified dependent variable into factor levels
Y_factor <- function(dataset_column) {
  return(factor(dataset_column, levels=c(0,1), labels=c("peace", "war")))
}

# Function for getting dataset for correspondence between cid (Country ID) and country from original HS
# This is needed because some other datasets only include cid
country_correspondence <- data_HS_original[,c("cid", "country")]
get_countries <- function(country_ids) {
  countries <- c()
  for(id in country_ids) {
    countries <- c(countries, country_correspondence[country_correspondence$cid==id, ]$country[1])
  }
  return(countries)
}

# Convert 'warstds' column into 'peace' or 'war' (initially marked 0 or 1)
data_HS_original$warstds <- Y_factor(data_HS_original$warstds)
data_HS_imputed$warstds <- Y_factor(data_HS_imputed$warstds)
```

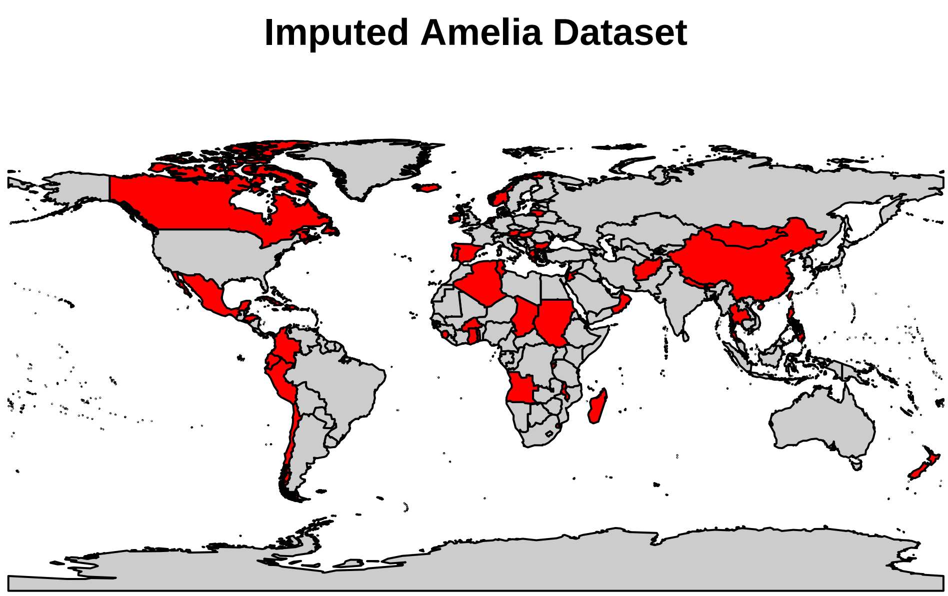
```r
data_AM_imputed$warstds <- Y_factor(data_AM_imputed$warstds)
data_AF_imputed$warstds <- Y_factor(data_AF_imputed$warstds)

# Visualize countries with civil war for each dataset
data(wrld_simpl)
HS_original_countries <- wrld_simpl@data$NAME %in%
  data_HS_original[data_HS_original$warstds=='war',]$country
HS_imputed_countries <- wrld_simpl@data$NAME %in%
  get_countries(data_HS_imputed[data_HS_imputed$warstds=='war',]$cid)
AM_imputed_countries <- wrld_simpl@data$NAME %in%
  data_AM_imputed[data_AM_imputed$warstds=='war',]$country

plot(wrld_simpl, col=c(gray(.80),"red")[HS_original_countries+1],
    main='Original Hegre and Sambanis (2006)')
```




```r
plot(wrld_simpl, col=c(gray(.80),"red")[HS_imputed_countries+1],
    main='Imputed Hegre and Sambanis')
```



```r
plot(wrld_simpl, col=c(gray(.80),"red")[AM_imputed_countries+1],
     main='Imputed Amelia Dataset')
```



Data Insights and Moving Forward

There are a few questions left unanswered after replicating the data importing and processing done by authors. Below we represent the comparisons of various datasets mentioned in the paper and the original R code, and give insights about our reasoning.

- The reported comparison data frame highlights that the original Hegre and Sambanis (2006) dataset, denoted by `HS_original`, has 9691 examples, 284 features, and 979981 cells containing missing (NA) values.
- The dataset that is constructed by ourselves when these cells were omitted, `HS_cleaned`, shows that every single row of the original dataset contains some missing values. Naturally, this dataset is **dropped** from now on as it doesn't contain any entries.

- The dataset imputed by authors on the other hand, denoted `HS_imputed`, has 7140 examples, 286 features, and 0 cells containing missing (NA) values. It is unclear and unmentioned how and why the authors have imputed this dataset, filled all cells with missing values, and deleted ~2500 examples from the original dataset.
- The second dataset imputed by authors, denoted `AM_imputed`, has 7141 examples, 53 features, and 778 cells containing missing (NA) values.
- Lastly, the third dataset imputed by authors, denoted `AF_imputed`, has 737 examples, 11 features, and 0 cells containing missing (NA) values. This dataset is also **dropped**, because
  i) no country information exists whatsoever, and more importantly
  ii) none of the features (columns) in this dataset match with any of the other features in the other datasets (except the dependent variable).

It is also unclear why the paper needs three different imputed datasets. The `AM_imputed` dataset is supposed to be the smaller dataset where features theorized to be most relevant to the onset of civil war are imputed. Although the number of features decrease as claimed, 778 cells with missing (NA) values reappear in this dataset, and the number of examples increase by 1 without any explanation.

For comparison and metrics reporting purposes anyway, it is usually a better idea to select a singular dataset and train & test models on this same dataset through a reasonable splitting of data. Other datasets and the `HS_imputed` dataset without the split will also be utilized in parts where we try to replicate results from the paper. Although we don't think in-sample measures are the correct way to assess the performance of any model, we will do so to see if we can **completely** replicate graphs and reported measures in the original paper.

Before generating the out-of-sample data through a train/test split, let's try to understand the difference between the datasets used by authors through an exploration of the respective numbers of binary classes in each dataset.

Exploration of Class Imbalance in Datasets

```r
# Explore class imbalance
table(data_HS_original$warstds)
```

```
##
## peace   war
##  6247   116
```

```r
table(data_HS_imputed$warstds)
```

```
##
## peace   war
##  7024   116
```

```r
table(data_AM_imputed$warstds)
```

```
##
## peace   war
##  7025   116
```

```r
table(data_AF_imputed$warstds)
```

```
##
## peace   war
##   716    21
```

It can be inferred that authors have either deleted or modified examples from the `HS_original` dataset where the `warstds` variable was NA when they were preparing their imputed dataset `HS_imputed`. Hence, they increased examples of peace by 800 examples in their imputation. In such a class-imbalanced data (as displayed by tables above), one would imagine i) down-sampling from the majority class or ii) up-sampling from the minority class would be the reasonable action, rather than increasing the number of examples of the majority class.

We will implement these two techniques, i) and ii), through a method called SMOTE and see if they yield better results. SMOTE (Synthetic Minority Over-sampling Technique) is designed for problems when one class dominates the other, which usually happens in rare-event occurrences. We can easily make the argument that it is thus appropraite for the civil war onset data at hand. The general idea of this method is to artificially generate new examples of the minority class using the nearest neighbors of these cases. Furthermore, the majority class examples are also under-sampled, leading to a more balanced dataset. (RDocumentation)

Model Specifications

To keep track of model specifications which use different numbers and types of features based on either theory or computations, we propose that we explicitly define features and hence accompanying formulas to be used by models later in this next code block. The 4 specifications the paper have covered are:

1) Fearon and Laitin specification (2003) consisting of 11 variables
2) Collier and Hoeffler specification (2004) consisting of 12 variables
3) Hegre and Sambanis specification (2006) consisting of 20 variables
4) The authors' specifications consisting of 88 variables, selected from Sambanis (2006) index

Feature Selection Based on Model Specifications

```r
# Function to generate a formula given dependent variable label and features
# Ex: get_model_formula('height', c('age', 'weight')) : height ~ age + weight
get_model_formula <- function(label, feature_vector) {
  formula_string <- ""
  for (feature in feature_vector) {
    formula_string <- paste(formula_string, feature, "+")
  }
  formula_string <- substring(formula_string, 1, nchar(formula_string)-1)
  return(as.formula(paste(paste(label, "~"), formula_string)))
}

# Specify the dependent variable that will be predicted in all models
y_var <- "warstds"

# The 88 variables selected by authors from Sambanis (2006) Appendix as spec of their RF model
author_vars <- c("ager", "agexp", "anoc", "army85", "autch98", "auto4",
                 "autonomy", "avgnabo", "centpol3", "coldwar", "decade1", "decade2",
                 "decade3", "decade4", "dem", "dem4", "demch98", "dlang", "drel",
                 "durable", "ef", "ef2", "ehet", "elfo", "elfo2", "etdo4590",
                 "expgdp", "exrec", "fedpol3", "fuelexp", "gdpgrowth", "geo1", "geo2",
                 "geo34", "geo57", "geo69", "geo8", "illiteracy", "incumb", "infant",
                 "inst", "inst3", "life", "lmtnest", "ln_gdpen", "lpopns", "major", "manuexp",
                 "milper", "mirps0", "mirps1", "mirps2", "mirps3", "nat_war", "ncontig",
                 "nmgdp", "nmdp4_alt", "numlang", "nwstate", "oil", "p4mchg",
                 "parcomp", "parreg", "part", "partfree", "plural", "plurrel",
                 "pol4", "pol4m", "pol4sq", "polch98", "polcomp", "popdense",
                 "presi", "pri", "proxregc", "ptime", "reg", "regd4_alt", "relfrac",
```

```r
                 "seceduc", "second", "semipol3", "sip2", "sxpnew", "sxpsq", "tnatwar",
                 "trade", "warhist", "xconst")
author_spec <- get_model_formula(y_var, author_vars)

# The 11 variables selected by Fearon and Laitin (2003) as spec of their LR model
FL_vars <- c("warhist", "ln_gdpen", "lpopns", "lmtnest", "ncontig",
             "oil", "nwstate", "inst3", "pol4", "ef", "relfrac")
FL_spec <- get_model_formula(y_var, FL_vars)

# The 12 variables selected by Collier and Hoeffler (2004) as spec of their LR model
CH_vars <- c("sxpnew", "sxpsq", "ln_gdpen", "gdpgrowth", "warhist", "lmtnest",
             "ef", "popdense", "lpopns", "coldwar", "seceduc", "ptime")
CH_spec <- get_model_formula(y_var, CH_vars)

# The 20 variables selected by Hegre and Sambanis (2006) as spec of their LR model
HS_vars <- c("lpopns", "ln_gdpen", "inst3", "parreg", "geo34", "proxregc", "gdpgrowth",
             "anoc", "partfree", "nat_war", "lmtnest", "decade1", "pol4sq", "nwstate",
             "regd4_alt", "etdo4590", "milper", "geo1", "tnatwar", "presi")
HS_spec <- get_model_formula(y_var, HS_vars)
```

Data Processing

Based on our exploration, we will **drop** every dataset discussed up until now except `HS_imputed`, and do a split on this dataset. The original R code also **mostly** uses this same dataset for training and testing models. This is to prevent any ambiguity, and also to respect the research done by Muchlinski et. al. (2016). Although it is unmentioned how they removed examples and filled missing values, this dataset seems the most workable with when the missing values and variables in other datasets are considered. The specific splitting strategy is the one that was mentioned but not implemented by the authors: Examples spanning years *1945-1990* will construct the training set, whereas examples after *1990* will construct the testing set.

Removing Unused Columns

```r
# Specify extra variables to keep apart from specifications mentioned in the previous segment
extra_vars <- c(y_var, "year", "cid")
all_vars <- unique(c(author_vars, FL_vars, CH_vars, HS_vars, extra_vars))
# Remove unwanted columns and specify chosen dataset
data <- data_HS_imputed[,all_vars] # KEEP: corresponds to data.full in original author's code
ncol(data) # debugging
```

```
## [1] 93
```

Training/Test Split for Accurate Assessment

```r
# Perform the split
split_year <- 1990
data_train <- data[data$year <= split_year,]
data_test <- data[data$year > split_year,]
```

```r
# Observe class-imbalance in training and testing sets
table(data_train$warstds)
```

```
##
## peace   war
##  5357    93
```

```r
table(data_test$warstds)
```

```
##
## peace   war
##  1667    23
```

```r
# Since we are done with visualizations and exploration, refactor the dependent variable
data_train$warstds <- as.factor(data_train$warstds)
data_test$warstds <- as.factor(data_test$warstds)
```

The ratio of war to peace in our training set is approx. 0.013 and in our test set is 0.017. Since these two ratios are comparable, the training and test set generated by our split year share similar characteristics. Hence, we stick to this train-test split suggested.

SMOTE

```r
# SMOTE for artificially creating a more balanced training dataset
data_train_balanced <- SMOTE(warstds ~ ., data_train, perc.over = 600, perc.under=100)
table(data_train_balanced$warstds)
```

```
##
## peace   war
##   558   651
```

Model Training

Training Setup

Below is the setup to be run before training. One thing to look for is that, we will set the seed for each caret training function call in order to present results that are replicable. The original code by the authors only set the seed once, which in turn makes some their results unreplicable. Moreover, the `tc_original` and `tc` training controls showcase the difference in indexing between author's approach and our approach. Setting the index argument makes our models replicable, whereas the lack of it causes author's models to be unreplicable.

```r
set.seed(666) ## the most metal seed for CV

# Specify number of folds (10 by default, suggested by authors)
num_folds <- 10

# Indexing that will be used with our models -> aimed to control randomness so all models are replicabl
cv_index <- createFolds(factor(data_train$warstds), num_folds, returnTrain=T)

# This is the original training control included in author's code that unfortunately makes models NOT r
# This will only be used with author's models (notice that the index argument is NOT set)
tc_original <- trainControl(method="cv",
```

```r
                                number=num_folds, ## creates CV folds - 10 for this data
                                summaryFunction=twoClassSummary, ## provides ROC stats in call to model
                                classProb=T)

# This is the corrected training control with indexing for replicability
tc <- trainControl(method="cv",
                   index=cv_index,
                   number=num_folds, ## creates CV folds - 10 for this data
                   summaryFunction=twoClassSummary, ## provides ROC stats in call to model
                   classProb=T)
```

Replicating Author's Training Processes

```r
# Fearon and Laitin LR Model (2003) Uncorrected
REP_model_FL_uncorrected <- train(FL_spec,
                                  metric="ROC", method="glm", family="binomial",
                                  trControl=tc_original, data=data)
#summary(REP_model_FL_uncorrected) ## provides coefficients & traditional R model output
#REP_model_FL_uncorrected ## provides CV summary stats
#confusionMatrix(REP_model_FL_uncorrected, norm="average") ## confusion matrix for predicted classes

# Fearon and Laitin LR Model (2003) Penalized
REP_model_FL_penalized <- train(FL_spec,
                                metric="ROC", method="plr", # Firth's penalized LR
                                trControl=tc_original, data=data)
#summary(REP_model_FL_penalized)
#REP_model_FL_penalized
#confusionMatrix(REP_model_FL_penalized, norm="average")

# Collier and Hoeffler LR Model (2004) Uncorrected
REP_model_CH_uncorrected <- train(CH_spec,
                                  metric="ROC", method="glm", family="binomial",
                                  trControl=tc_original, data=data)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
#summary(REP_model_CH_uncorrected) ## provides coefficients & traditional R model output
#REP_model_CH_uncorrected ## provides CV summary stats
#confusionMatrix(REP_model_CH_uncorrected, norm="average") ## confusion matrix for predicted classes

# Collier and Hoeffler LR Model (2004) Penalized
REP_model_CH_penalized <- train(CH_spec,
                                metric="ROC", method="plr", # Firth's penalized LR
                                trControl=tc_original, data=data)
```

```
##
## Convergence warning in plr: 2
```

```r
#summary(REP_model_CH_penalized)
#REP_model_CH_penalized
#confusionMatrix(REP_model_CH_penalized, norm="average")

# Hegre and Sambanis LR Model (2006) Uncorrected
```

```r
REP_model_HS_uncorrected <- train(HS_spec,
                                  metric="ROC", method="glm", family="binomial",
                                  trControl=tc_original, data=data)

#summary(REP_model_HS_uncorrected) ## provides coefficients & traditional R model output
#REP_model_HS_uncorrected ## provides CV summary stats
#confusionMatrix(REP_model_HS_uncorrected, norm="average") ## confusion matrix for predicted classes

# Hegre and Sambanis LR Model (2006) Penalized
REP_model_HS_penalized <- train(HS_spec,
                                metric="ROC", method="plr", # Firth's penalized LR
                                trControl=tc_original, data=data)
```

```
##
## Convergence warning in plr: 2
```

```r
#summary(REP_model_HS_penalized)
#REP_model_HS_penalized
#confusionMatrix(REP_model_HS_penalized, norm="average")

# Random Forest Model on author specification (2016)
REP_model_AS_RF <- train(author_spec,
                         metric="ROC", method="rf",
                         sampsize=c(30,90), ## Downsampling the class-imbalanced DV
                         importance=T, ## Variable importance measures retained
                         proximity=F, ntree=1000, ## number of trees grown
                         trControl=tc_original, data=data)
#summary(REP_model_AS_RF)
#REP_model_AS_RF
#confusionMatrix(REP_model_AS_RF, norm="average")
```

Training Insights

A major problem we noticed with the original implementation was that the initially commented out random forest training process included an option for down-sampling, whereas the other logistic regression training processes didn't have this option specified. As they weren't presented on the paper, we chose to not include those random forest models (except for author specification 88 variables) in our replication above. This has the potential to yield unfair comparisons, so we decided to remove down-sampling option in our own implementations.

Perhaps the biggest problem is that the authors have only used an in-sample accuracy metric to report the performance of their models which couldn't be directly inferred from the paper otherwise. Hence, the reported performance measures on the paper don't really give us much insight into how these models might behave with unseen data. In order to assess performance, we have to test our models for **out-of-sample data** `data_test` while training our models on the seperated `data_train`.

The already mentioned problem with index argument of the training controls and uncorrect seed setting unfortunately makes the results of the paper completely unreplicable. One can uncomment the 3 lines of summary reporting below each model in the above code block to observe how ROC values changes (although in miniscule amounts) with every training instance. This could be confirmed by running the same corresponding lines from author's original code included in `data/Comparing+Random+Forest+with+Logistic+Regression+R+Code.R` and observing how ROC values changes (although in miniscule amounts) again. It seems that the author's were unaware of the unreplicability of their results as they included hardcoded values for their AUC measures. In contrast, uncommenting the 3 lines of summary reporting below each model in the below code block should showcase how ROC values are preserved and hence prove the replicability of our implementation.

We will first train the 4 specifications mentioned in the paper with the `data_train` dataset, which was acquired by split (based on year) from the `data_HS_imputed` dataset provided by the authors. Then, finally, we will train a fifth instance with author specification (88 variables), but instead on the `data_train_balanced` dataset this time. As previously mentioned, this dataset was created with the SMOTE algorithm to deal with the class-imbalance in the dataset. For each of the 5 instances of training, we will include i) a logistic regression model (LR) with uncorrected logits, ii) a logistic regression model (LR) with penalized logits using Firth's method, and iii) a random forest model (RF).

All of the models use the corrected version of the cross-validation training control, `tc`, implemented in the above code block.

Our Own Training Processes

1) Fearon and Laitin specification (2003) consisting of 11 variables

```
set.seed(666) ## the most metal seed for CV

# Fearon and Laitin LR Model (2003) Uncorrected
model_FL_uncorrected <- train(FL_spec,
                              metric="ROC", method="glm", family="binomial",
                              trControl=tc, data=data_train)

#summary(model_FL_uncorrected) ## provides coefficients & traditional R model output
#model_FL_uncorrected ## provides CV summary stats
#confusionMatrix(model_FL_uncorrected, norm="average") ## confusion matrix for predicted classes

# Fearon and Laitin LR Model (2003) Penalized
model_FL_penalized <- train(FL_spec,
                            metric="ROC", method="plr", # Firth's penalized LR
                            trControl=tc, data=data_train)
#summary(model_FL_penalized)
#model_FL_penalized
#confusionMatrix(model_FL_penalized, norm="average")

#Random Forest Model on Fearon and Laitin (2003) specification
model_FL_RF <- train(FL_spec,
                     metric="ROC", method="rf",
                     trControl=tc, data=data_train)
#summary(model_FL_RF)
#model_FL_RF
#confusionMatrix(model_FL_RF, norm="average")
```

2) Collier and Hoeffler specification (2004) consisting of 12 variables

```
set.seed(666) ## the most metal seed for CV

# Collier and Hoeffler LR Model (2004) Uncorrected
model_CH_uncorrected <- train(CH_spec,
                              metric="ROC", method="glm", family="binomial",
                              trControl=tc, data=data_train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
#summary(model_CH_uncorrected) ## provides coefficients & traditional R model output
#model_CH_uncorrected ## provides CV summary stats
#confusionMatrix(model_CH_uncorrected, norm="average") ## confusion matrix for predicted classes

# Collier and Hoeffler LR Model (2004) Penalized
model_CH_penalized <- train(CH_spec,
                            metric="ROC", method="plr", # Firth's penalized LR
                            trControl=tc, data=data_train)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.

##
## Convergence warning in plr: 2
```

```r
#summary(model_CH_penalized)
#model_CH_penalized
#confusionMatrix(model_CH_penalized, norm="average")

# Random Forest Model on Collier and Hoeffler (2004) specification
model_CH_RF <- train(CH_spec,
                     metric="ROC", method="rf",
                     trControl=tc, data=data_train)
#summary(model_CH_RF)
#model_CH_RF
#confusionMatrix(model_CH_RF, norm="average")
```

3) Hegre and Sambanis specification (2006) consisting of 20 variables

```r
set.seed(666) ## the most metal seed for CV

# Hegre and Sambanis LR Model (2006) Uncorrected
model_HS_uncorrected <- train(HS_spec,
                              metric="ROC", method="glm", family="binomial",
                              trControl=tc, data=data_train)
#summary(model_HS_uncorrected) ## provides coefficients & traditional R model output
#model_HS_uncorrected ## provides CV summary stats
#confusionMatrix(model_HS_uncorrected, norm="average") ## confusion matrix for predicted classes

# Hegre and Sambanis LR Model (2006) Penalized
model_HS_penalized <- train(HS_spec,
                            metric="ROC", method="plr", # Firth's penalized LR
                            trControl=tc, data=data_train)
```

```
##
## Convergence warning in plr: 2
```

```r
#summary(model_HS_penalized)
#model_HS_penalized
#confusionMatrix(model_HS_penalized, norm="average")

# Random Forest Model on Hegre and Sambanis (2006) specification
model_HS_RF <- train(HS_spec,
                     metric="ROC", method="rf",
                     trControl=tc, data=data_train)
#summary(model_HS_RF)
```

```
#model_HS_RF
#confusionMatrix(model_HS_RF, norm="average")
```

4) The authors' specifications consisting of 88 variables, selected from Sambanis (2006) index

```
set.seed(666) ## the most metal seed for CV

# Author specification (2016) LR Model Uncorrected
model_AS_uncorrected <- train(author_spec,
                              metric="ROC", method="glm", family="binomial",
                              trControl=tc, data=data_train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#summary(model_AS_uncorrected) ## provides coefficients & traditional R model output
#model_AS_uncorrected ## provides CV summary stats
#confusionMatrix(model_AS_uncorrected, norm="average") ## confusion matrix for predicted classes

# Author specification (2016) LR Model Penalized (CAUTION: estimated training time ~30 min)
model_AS_penalized <- train(author_spec,
                            metric="ROC", method="plr", # Firth's penalized LR
                            trControl=tc, data=data_train)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.

##
## Convergence warning in plr: 2
```

```
#summary(model_AS_penalized)
#model_AS_penalized
#confusionMatrix(model_AS_penalized, norm="average")

# Random Forest Model on author specification (2016) (CAUTION: estimated training time ~30 min)
model_AS_RF <- train(author_spec,
                     metric="ROC", method="rf",
                     importance=T, ## Variable importance measures retained
                     trControl=tc, data=data_train)
#summary(model_AS_RF)
#model_AS_RF
#confusionMatrix(model_AS_RF, norm="average")
```

5) The authors' specifications consisting of 88 variables, trained on `data_train_balanced`, which was artificially generated from `data_train` using the SMOTE algorithm discussed previously.

```
set.seed(666) ## the most metal seed for CV

# Author specification (2016) LR Model Uncorrected
model_AS_uncorrected_smoted <- train(author_spec,
                                     metric="ROC", method="glm", family="binomial",
                                     trControl=tc, data=data_train_balanced)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
#summary(model_AS_uncorrected_smoted) ## provides coefficients & traditional R model output
#model_AS_uncorrected_smoted ## provides CV summary stats
#confusionMatrix(model_AS_uncorrected_smoted, norm="average") ## confusion matrix for predicted classes

# Author specification (2016) LR Model Penalized
model_AS_penalized_smoted <- train(author_spec,
                                   metric="ROC", method="plr", # Firth's penalized LR
                                   trControl=tc, data=data_train_balanced)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
##
## Convergence warning in plr: 2
```

```r
#summary(model_AS_penalized_smoted)
#model_AS_penalized_smoted
#confusionMatrix(model_AS_penalized_smoted, norm="average")

# Random Forest Model on author specification (2016)
model_AS_RF_smoted <- train(author_spec,
                            metric="ROC", method="rf",
                            importance=T, ## Variable importance measures retained
                            sampsize=c(30,90), # ADDED
                            proximity=F, ntree=100,  # ADDED
                            trControl=tc, data=data_train_balanced)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```r
#summary(model_AS_RF_smoted)
#model_AS_RF_smoted
#confusionMatrix(model_AS_RF_smoted, norm="average")
```

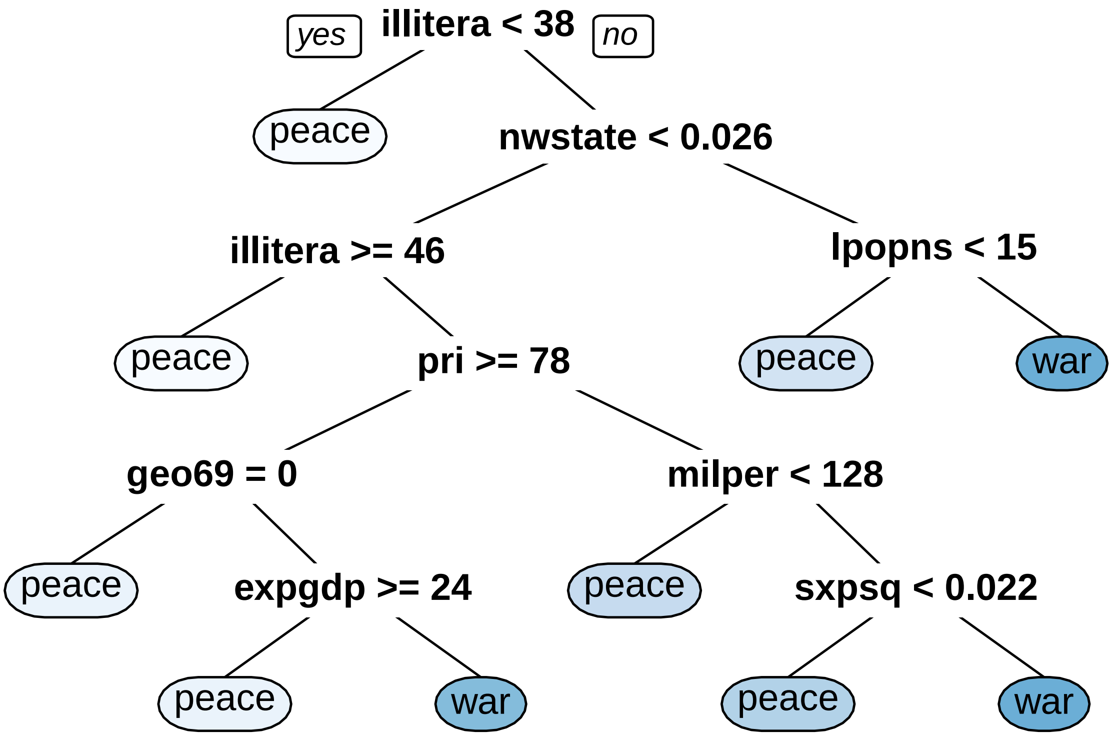
Experimenting with Different Models

In order to assess whether Random Forest is the ideal model to capture the behavior of the data, we decided to experiment with some other Machine Learning models. After running some parameter grid searches, we noticed that the Boosted Classification Trees model with the parameters below can achieve an AUC score of 0.922. The simpler Decision Tree model was not able to capture the nature of the dataset that well or achieve such results.

```r
# This is the Decision Tree model that we implemented
model_AS_DT <- train(author_spec,
                     metric="ROC",
                     method="rpart",
                     trControl=tc,
                     data=data_train)

# This is the Boosted Classifiaction Trees model that we implemented
model_AS_Boost <- train(author_spec,
                        metric="ROC",
                        method="ada",
                        trControl=tc,
```

```
                              tuneGrid=data.frame(.iter=150, .maxdepth=3, .nu=0.01),
                              data=data_train)
```

```
# Visualize the Decision Tree to assess which predictors were used and what the cut-offs were
prp(model_AS_DT$finalModel, box.palette = "Blues", tweak = 1.2)
```



Variable Importance for Random Forest Models

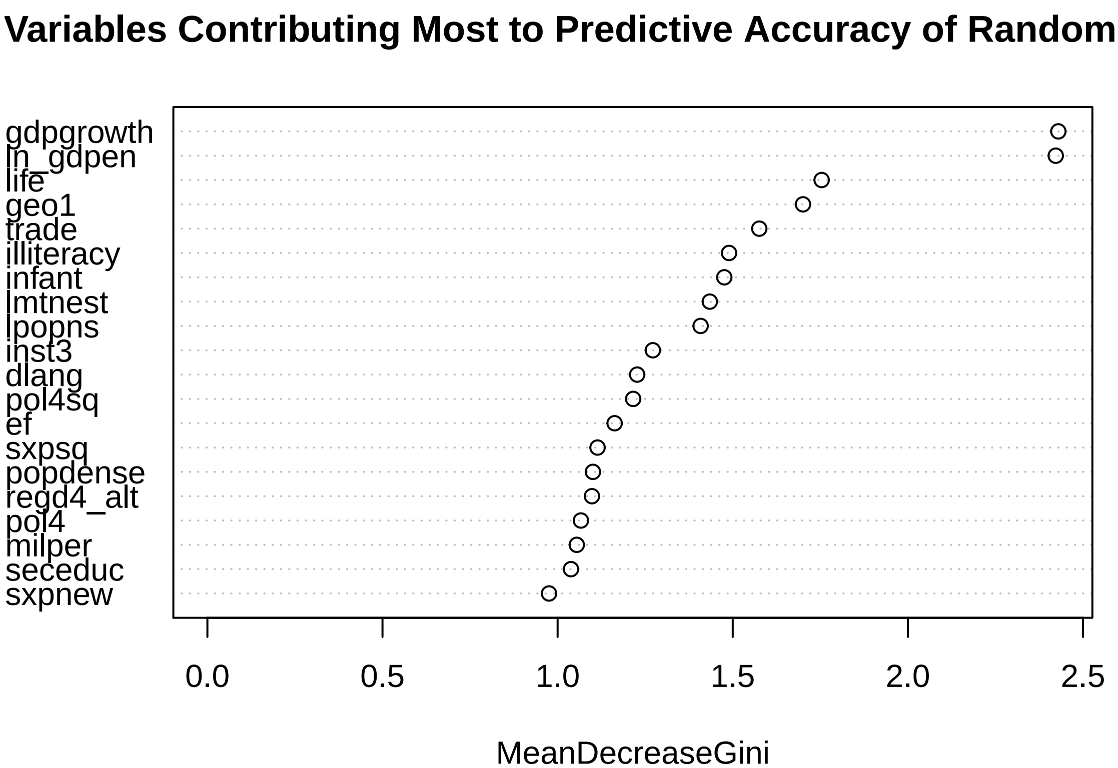
One advantage of the Random Forests algorithm is that it is easy to interpret, robust to outliers and noise, and allows the analyst to infer causal mechanisms. Although the paper has serious methodological flaws, we think that its suggestion of Random Forests is well-grounded. The original R code by authors visualizes the variable importance for a Random Forest Model trained on the `data_AM_imputed` dataset, however we **dropped** that dataset as discussed before. It makes the most sense to us to visualize the variable importance on the 5 different random forest models we trained in the previous segment, 4 on `data_train` and 1 on `data_train_balanced` to be exact. By doing this, we can achieve: i) See if theories suggested by different parties on theoretical variables deciding a civil war onset agree with our findings or not, ii) Observe if any variables are deemed to be `unimportant` by the random forest algorithm, measured by mean decrease in Gini classification measure in the case that they are not considered in training, iii) Observe the difference of variable importance on random forests models trained on original training set and artifically created training set, and essentially see how the SMOTE algorithms affects this measure.

Replicating Author's Variable Importance Visualizations

Before diving into our own versions, let's see if we get the same results with the paper on variable importance of a random forest model trained on `data_AM_imputed`.

```
## Random Forests on Amelia Imputed Data for Variable Importance Plot
## Data Imputed only for Theoretically Important Variables
```

```r
## Done to analyze causal mechanisms

# Author removes unnecessary columns from the dataset
removed_vars <- names(data_AM_imputed) %in% c("X", "country", "year", "atwards")
data_AM_imputed <- data_AM_imputed[!removed_vars]

REP_model_AM_imputed <-randomForest(as.factor(warstds)~., # new spec: all columns in Amelia dataset
                                    sampsize=c(30, 90), # downsampling
                                    importance=T, proximity=F, ntree=1000,
                                    confusion=T, err.rate=T, data=data_AM_imputed)
# NOTE: The downsampling is random at each run due to incorrect seed setting (not replicable)

varImpPlot(REP_model_AM_imputed, sort=T, type=2,
           main="Variables Contributing Most to Predictive Accuracy of Random Forests",
           n.var=20)
```




```r
## Creating dotplot for Variable Importance Plot for RF
#importance(REP_model_AM_imputed) # commented-out because prints a long list
```

One can check the full descriptions of the variables printed above on the y-axis of the Variable Importance table from the included `data/Sambanis Appendix (Aug 06).pdf` PDF file. When compared to the results from the paper, we see that roughly the same variables are included in the plot as the 20 most important variables. However, it could be easily observed that the order of these variables are different than those reported in the original paper. This is parallel to unreplicability issue mentioned previously.

Retraining Random Forest Models

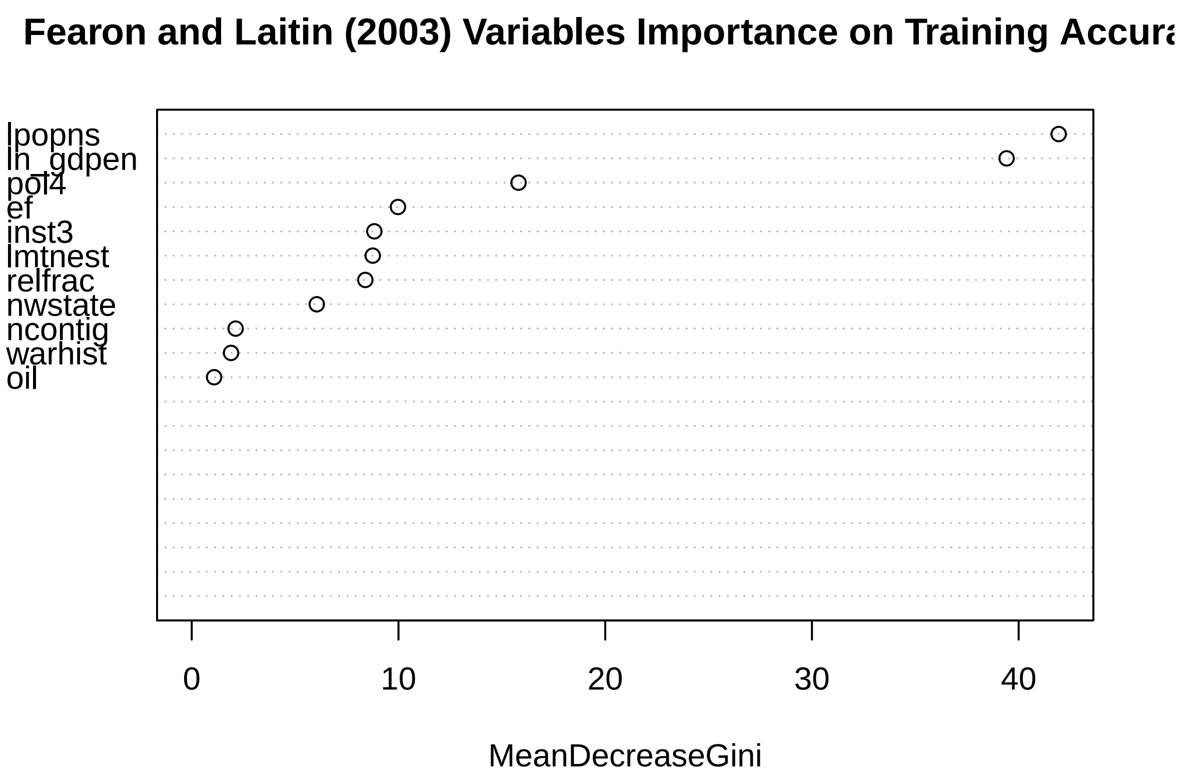
Firstly, we will retrain random forest models with the same specs and data mentioned above, but with `randomForests()` class this same in order to plot importance of variables. Note that these models will only be used to visualize variable importance, and will be dropped later in order to not cause any confusion.

```
num_trees <- 1000
FL_RF <- randomForest(FL_spec, importance=T, proximity=F, ntree=num_trees,
                      confusion=T, err.rate=T, data=data_train)
CH_RF <- randomForest(CH_spec, importance=T, proximity=F, ntree=num_trees,
                      confusion=T, err.rate=T, data=data_train)
HS_RF <- randomForest(HS_spec, importance=T, proximity=F, ntree=num_trees,
                      confusion=T, err.rate=T, data=data_train)
AS_RF <- randomForest(author_spec, importance=T, proximity=F, ntree=num_trees,
                      confusion=T, err.rate=T, data=data_train)
AS_RF_smoted <- randomForest(author_spec, importance=T, proximity=F, ntree=num_trees,
                             confusion=T, err.rate=T, data=data_train_balanced)
```

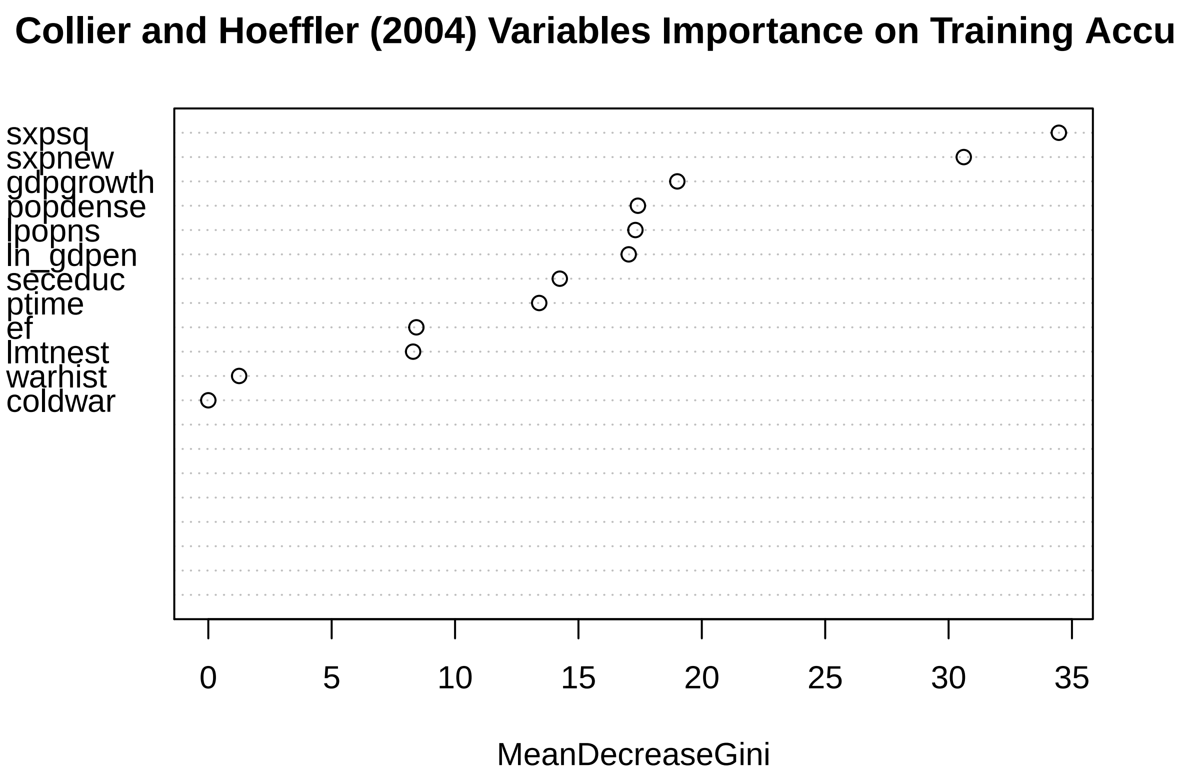
Our Own Variable Importance Visualizations

Now, we can visualize the variable importances for each of the random forest models trained above.

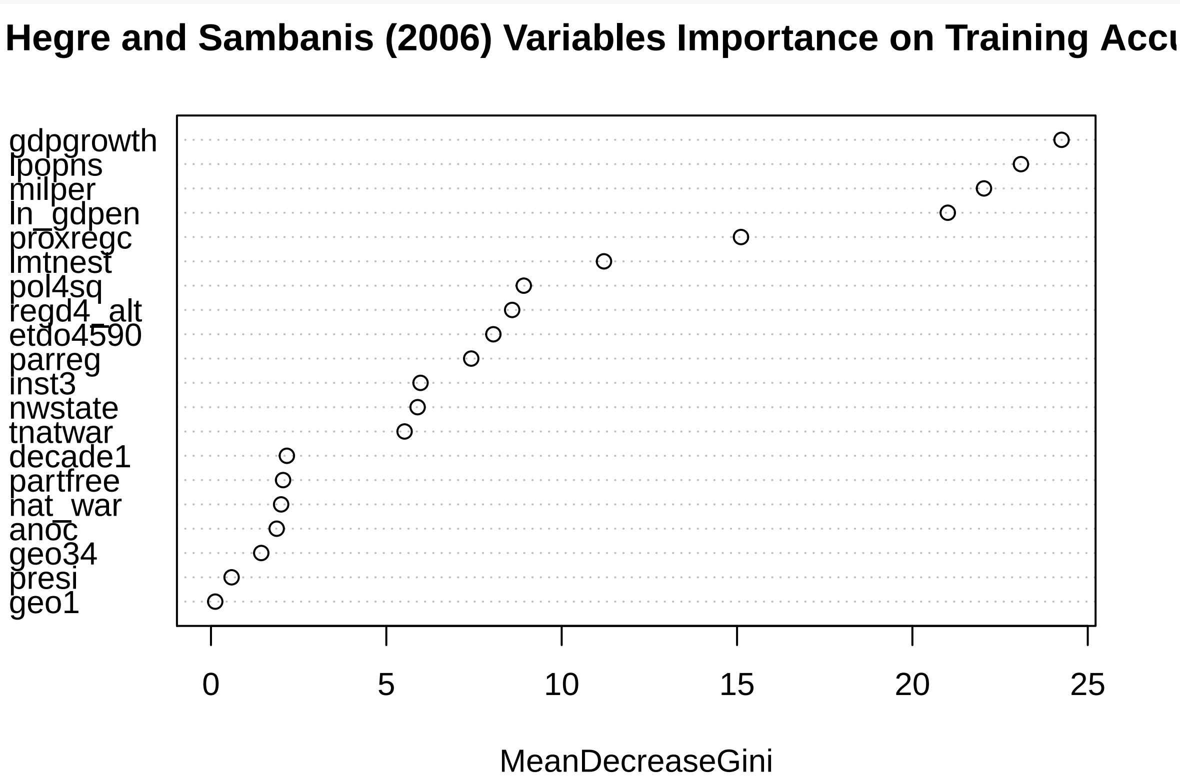
```
# Specify maximum number of variables (features) to be displayed
num_vars <- 20
varImpPlot(FL_RF, sort=T, type=2,
           main="Fearon and Laitin (2003) Variables Importance on Training Accuracy",
           n.var=num_vars)
```




```
varImpPlot(CH_RF, sort=T, type=2,
           main="Collier and Hoeffler (2004) Variables Importance on Training Accuracy",
           n.var=num_vars)
```



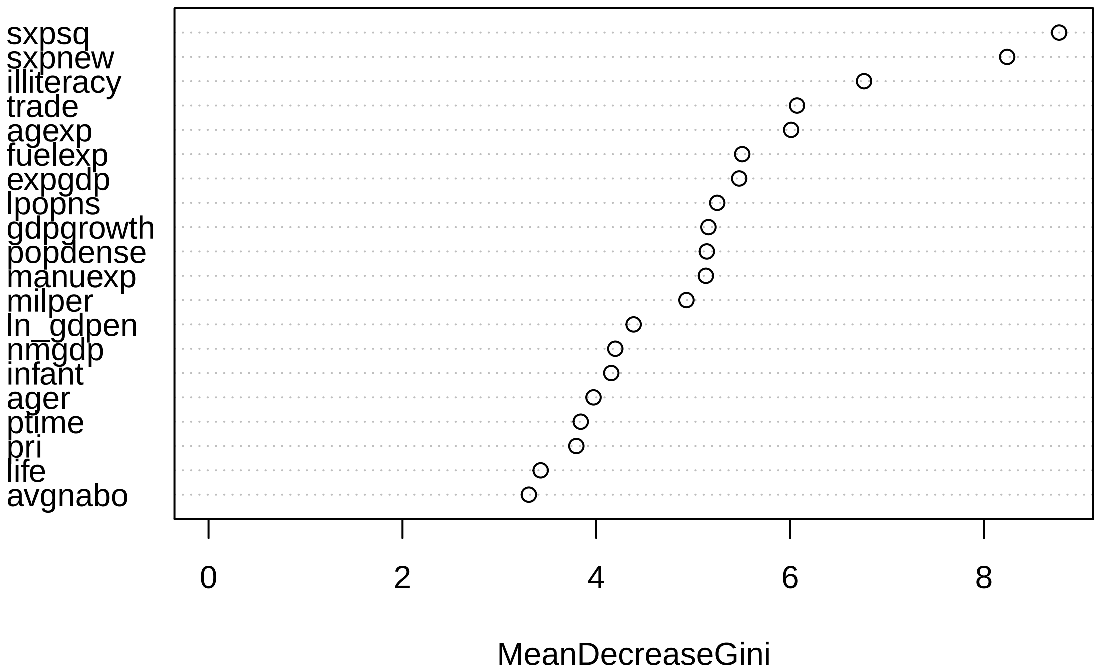

```
varImpPlot(HS_RF, sort=T, type=2,
           main="Hegre and Sambanis (2006) Variables Importance on Training Accuracy",
           n.var=num_vars)
```




```
varImpPlot(AS_RF, sort=T, type=2,
           main="Author Specified (2016) Variables Importance on Training Accuracy",
```

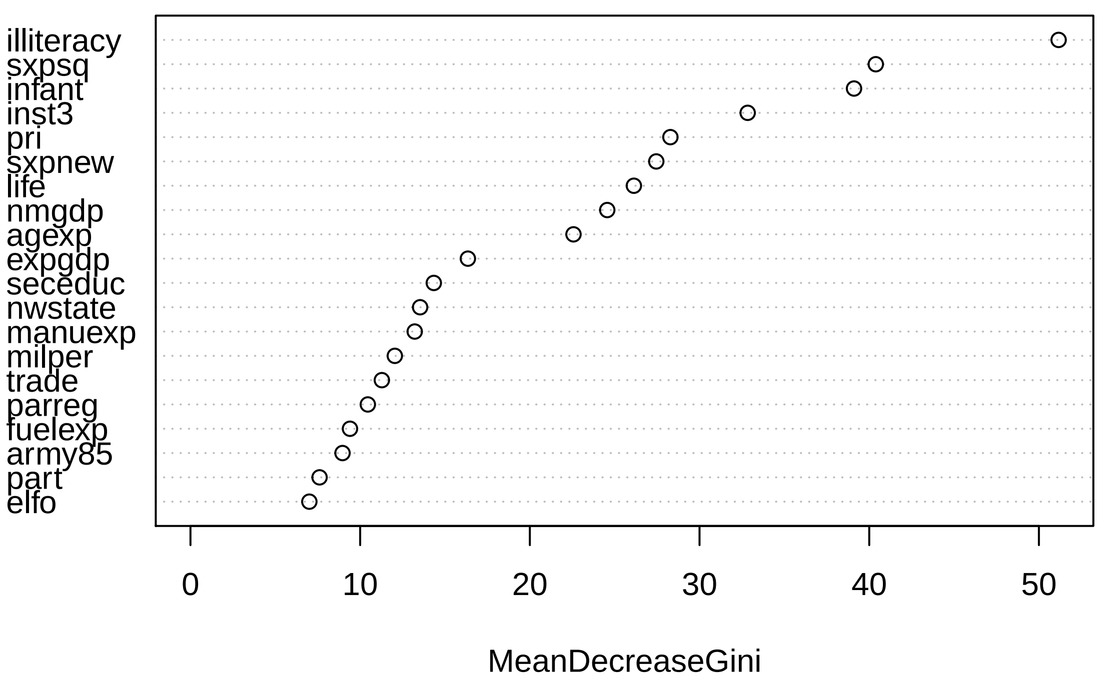
```
           n.var=num_vars)
```

Author Specified (2016) Variables Importance on Training Accura




```
varImpPlot(AS_RF_smoted, sort=T, type=2,
           main="Author Specified (2016) Variables Importance on Training Accuracy w/ SMOTEd data",
           n.var=num_vars)
```

thor Specified (2016) Variables Importance on Training Accuracy w/ SM



Model Testing

Testing Insights

The paper includes a graph for comparing training AUC for uncorrected logistic regression models versus the random forest model, and a seperate graph for comparing the same metric for penalized logistic regression models versus the random forest model. The immediately noticable problem with this approach is that, each of these model have different specifications and hence different numbers of features they are trained on. As discussed before, the random forest model included downsampling which made the comparison even less reliable due to possible overfitting.

Here, we propose that we compare ROC curves and AUC scores for each of the specifications on **out-of-sample** data independently. With this methodology, we will be able to assess the performance of random forest algorithm in comparison to uncorrected & penalized logistic regression models in a more reliable way. We realize that the out-of-sample testing data contains a low number of examples, however this approach is still better than comparing plain training accuracies. Furthermore, this is due to the nature of this problem and the difficulty of data gathering associated with it.

For each of the 3 models used for each of the 5 specifications and, we will use the `predict()` function with: i) type="raw" for number/class of predictions to be used in confusion matrices, and ii) type="prob" for class probabilities to be used with ROC curves and computation of AUC metric.

Replicating Author's Prediction Processes

We have previously discussed that the author's replicated models are trained on the entire `HS_imputed` dataset, predicted/tested on the same, entire dataset as well, and how this hurst the reliability of the performance measures reported in the original paper. Below is the corresponding replication.

```
# Testing/predictions for Fearon and Laitin specification (2003)
REP_FL_uncorrected_pred_raw <- predict(REP_model_FL_uncorrected, newdata=data, type="raw")
REP_FL_uncorrected_pred_prob <- predict(REP_model_FL_uncorrected, newdata=data, type="prob")
REP_FL_penalized_pred_raw <- predict(REP_model_FL_penalized, newdata=data, type="raw")
REP_FL_penalized_pred_prob <- predict(REP_model_FL_penalized, newdata=data, type="prob")

# Testing/predictions for Collier and Hoeffler specification (2004)
REP_CH_uncorrected_pred_raw <- predict(REP_model_CH_uncorrected, newdata=data, type="raw")
REP_CH_uncorrected_pred_prob <- predict(REP_model_CH_uncorrected, newdata=data, type="prob")
REP_CH_penalized_pred_raw <- predict(REP_model_CH_penalized, newdata=data, type="raw")
REP_CH_penalized_pred_prob <- predict(REP_model_CH_penalized, newdata=data, type="prob")

# Testing/predictions for Hegre and Sambanis specification (2006)
REP_HS_uncorrected_pred_raw <- predict(REP_model_HS_uncorrected, newdata=data, type="raw")
REP_HS_uncorrected_pred_prob <- predict(REP_model_HS_uncorrected, newdata=data, type="prob")
REP_HS_penalized_pred_raw <- predict(REP_model_HS_penalized, newdata=data, type="raw")
REP_HS_penalized_pred_prob <- predict(REP_model_HS_penalized, newdata=data, type="prob")

# Testing/predictions for author specification (2016)
REP_AS_RF_pred_raw <- predict(REP_model_AS_RF, newdata=data, type="raw")
REP_AS_RF_pred_prob <- predict(REP_model_AS_RF, newdata=data, type="prob")
```

Our Own Prediction Processes

```r
# Testing/predictions for Fearon and Laitin specification (2003)
FL_uncorrected_pred_raw <- predict(model_FL_uncorrected, newdata=data_test, type="raw")
FL_uncorrected_pred_prob <- predict(model_FL_uncorrected, newdata=data_test, type="prob")
FL_penalized_pred_raw <- predict(model_FL_penalized, newdata=data_test, type="raw")
FL_penalized_pred_prob <- predict(model_FL_penalized, newdata=data_test, type="prob")
FL_RF_pred_raw <- predict(model_FL_RF, newdata=data_test, type="raw")
FL_RF_pred_prob <- predict(model_FL_RF, newdata=data_test, type="prob")

# Testing/predictions for Collier and Hoeffler specification (2004)
CH_uncorrected_pred_raw <- predict(model_CH_uncorrected, newdata=data_test, type="raw")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```r
CH_uncorrected_pred_prob <- predict(model_CH_uncorrected, newdata=data_test, type="prob")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```r
CH_penalized_pred_raw <- predict(model_CH_penalized, newdata=data_test, type="raw")
CH_penalized_pred_prob <- predict(model_CH_penalized, newdata=data_test, type="prob")
CH_RF_pred_raw <- predict(model_CH_RF, newdata=data_test, type="raw")
CH_RF_pred_prob <- predict(model_CH_RF, newdata=data_test, type="prob")

# Testing/predictions for Hegre and Sambanis specification (2006)
HS_uncorrected_pred_raw <- predict(model_HS_uncorrected, newdata=data_test, type="raw")
HS_uncorrected_pred_prob <- predict(model_HS_uncorrected, newdata=data_test, type="prob")
HS_penalized_pred_raw <- predict(model_HS_penalized, newdata=data_test, type="raw")
HS_penalized_pred_prob <- predict(model_HS_penalized, newdata=data_test, type="prob")
HS_RF_pred_raw <- predict(model_HS_RF, newdata=data_test, type="raw")
HS_RF_pred_prob <- predict(model_HS_RF, newdata=data_test, type="prob")

# Testing/predictions for author specification (2016)
AS_uncorrected_pred_raw <- predict(model_AS_uncorrected, newdata=data_test, type="raw")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```r
AS_uncorrected_pred_prob <- predict(model_AS_uncorrected, newdata=data_test, type="prob")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```r
AS_penalized_pred_raw <- predict(model_AS_penalized, newdata=data_test, type="raw")
AS_penalized_pred_prob <- predict(model_AS_penalized, newdata=data_test, type="prob")
AS_RF_pred_raw <- predict(model_AS_RF, newdata=data_test, type="raw")
AS_RF_pred_prob <- predict(model_AS_RF, newdata=data_test, type="prob")

# Testing/prediction for author specification (2016) trained on SMOTEd dataset
AS_uncorrected_smoted_pred_raw <- predict(model_AS_uncorrected_smoted, newdata=data_test, type="raw")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```r
AS_uncorrected_smoted_pred_prob <- predict(model_AS_uncorrected_smoted, newdata=data_test, type="prob")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```r
AS_penalized_smoted_pred_raw <- predict(model_AS_penalized_smoted, newdata=data_test, type="raw")
AS_penalized_smoted_pred_prob <- predict(model_AS_penalized_smoted, newdata=data_test, type="prob")
AS_RF_smoted_pred_raw <- predict(model_AS_RF_smoted, newdata=data_test, type="raw")
AS_RF_smoted_pred_prob <- predict(model_AS_RF_smoted, newdata=data_test, type="prob")

# Testing/prediction for additional Machine Learning models
AS_DT_pred_raw <- predict(model_AS_DT, newdata=data_test, type="raw")
AS_DT_pred_prob <- predict(model_AS_DT, newdata=data_test, type="prob")
AS_Boost_pred_raw <- predict(model_AS_Boost, newdata=data_test, type="raw")
AS_Boost_pred_prob <- predict(model_AS_Boost, newdata=data_test, type="prob")
```

Confusion Matrices

Let's draw the confusion matrices for each instance we trained in our own implementation.

```r
# Confusion matrices for prediction with Fearon and Laitin specification (2003)
table(pred=FL_uncorrected_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1667   23
##   war       0    0
```

```r
table(pred=FL_penalized_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1667   23
##   war       0    0
```

```r
table(pred=FL_RF_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1664   20
##   war       3    3
```

```r
# Confusion matrices for prediction with Collier and Hoeffler specification (2004)
table(pred=CH_uncorrected_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1666   23
##   war       1    0
```

```r
table(pred=CH_penalized_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1555   16
##   war     112    7
```

```r
table(pred=CH_RF_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1660   15
##   war       7    8
```

```r
# Confusion matrices for prediction with Hegre and Sambanis specification (2006)
table(pred=HS_uncorrected_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1664   22
##   war       3    1
```

```r
table(pred=HS_penalized_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1667   22
##   war       0    1
```

```r
table(pred=HS_RF_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1666   23
##   war       1    0
```

```r
# Confusion matrices for prediction with author specification (2016)
table(pred=AS_uncorrected_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1629   20
##   war      38    3
```

```r
# table(pred=AS_penalized_pred_raw, obs=data_test$warstds)
# table(pred=AS_RF_pred_raw, obs=data_test$warstds)

# Confusion matrices for prediction with author specification (2016) trained on SMOTEd dataset
table(pred=AS_uncorrected_smoted_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1404   10
##   war     263   13
```

```r
table(pred=AS_penalized_smoted_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1533   14
##   war     134    9
```

```r
table(pred=AS_RF_smoted_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
```

```
##   peace  1288    2
##   war     379   21
```

```r
# Confusion matrices for prediction with author specification using additional models
table(pred=AS_DT_pred_raw, obs=data_test$warstds)
```

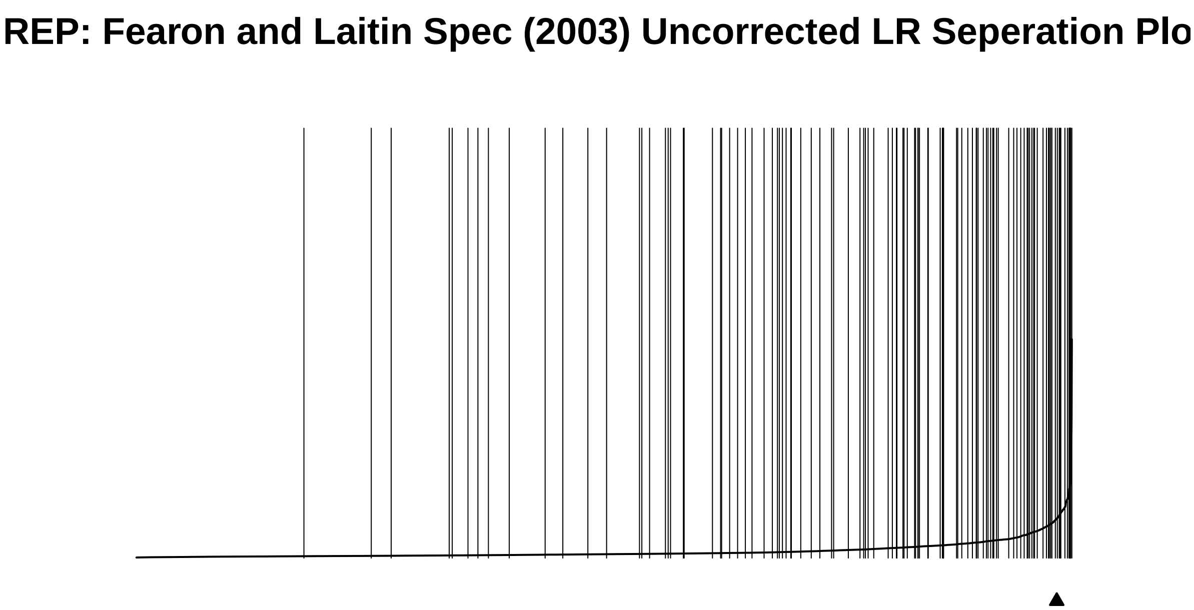
```
##        obs
## pred    peace  war
##   peace  1640   19
##   war      27    4
```

```r
table(pred=AS_Boost_pred_raw, obs=data_test$warstds)
```

```
##        obs
## pred    peace  war
##   peace  1667   22
##   war       0    1
```

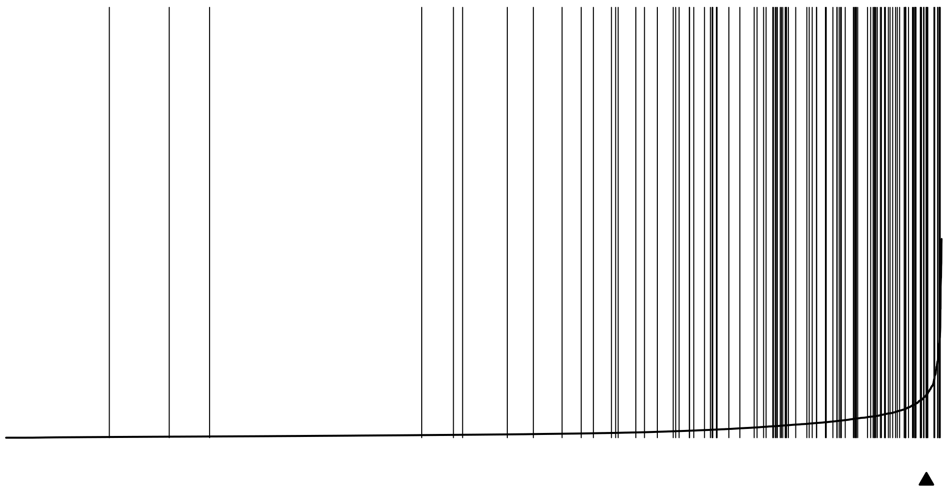
Replicating Author's Seperation Plots

```r
# Seperation plot for prediction with Fearon and Laitin specification (2003) uncorrected LR
separationplot(REP_FL_uncorrected_pred_prob$war,
               as.numeric(data$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="REP: Fearon and Laitin Spec (2003) Uncorrected LR Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```



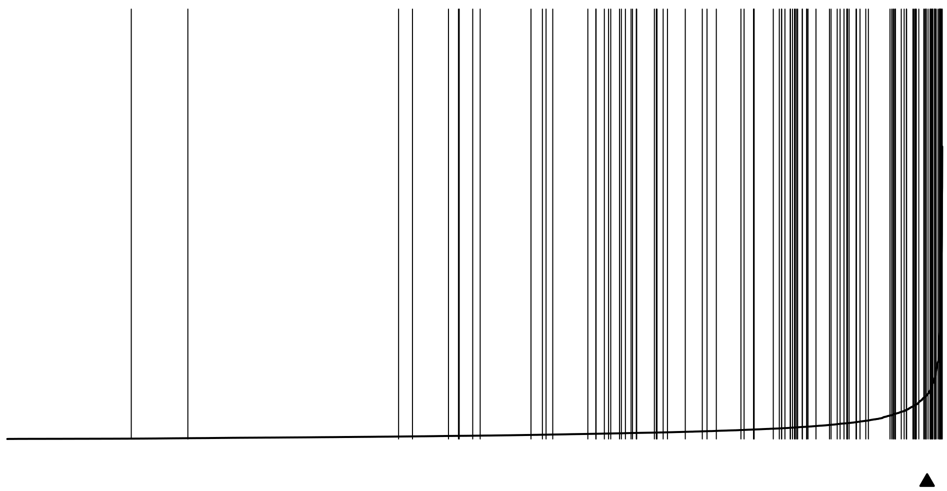

```r
# Seperation plot for prediction with Collier and Hoeffler specification (2004) uncorrected LR
separationplot(REP_CH_uncorrected_pred_prob$war,
               as.numeric(data$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="REP: Collier and Hoeffler Spec (2004) Uncorrected LR Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

REP: Collier and Hoeffler Spec (2004) Uncorrected LR Seperation Pl



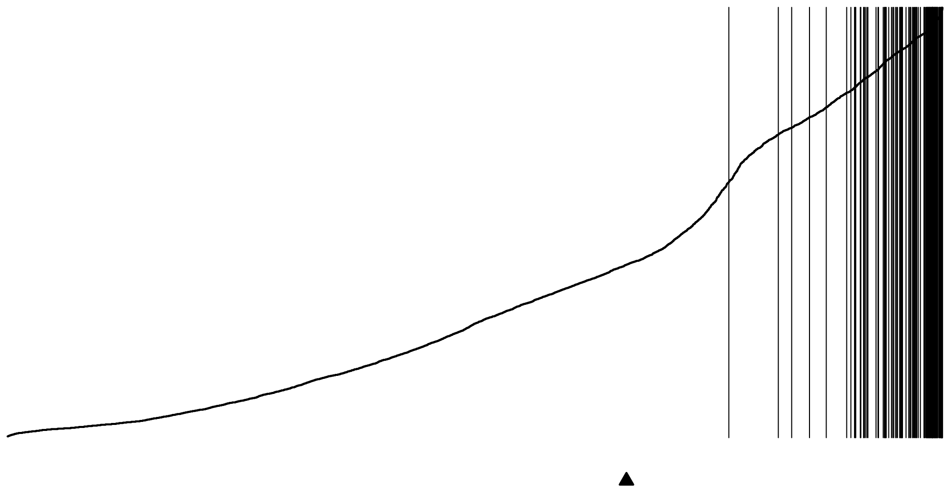
```r
# Seperation plot for prediction with Hegre and Sambanis specification (2006) uncorrected LR
separationplot(REP_HS_uncorrected_pred_prob$war,
               as.numeric(data$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="REP: Hegre and Sambanis Spec (2006) Uncorrected LR Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

REP: Hegre and Sambanis Spec (2006) Uncorrected LR Seperation P



```r
# Seperation plot for prediction with Hegre and Sambanis specification (2006) uncorrected LR
separationplot(REP_AS_RF_pred_prob$war,
               as.numeric(data$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="REP: Author Spec (2016) Random Forests Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

REP: Author Spec (2016) Random Forests Seperation Plot



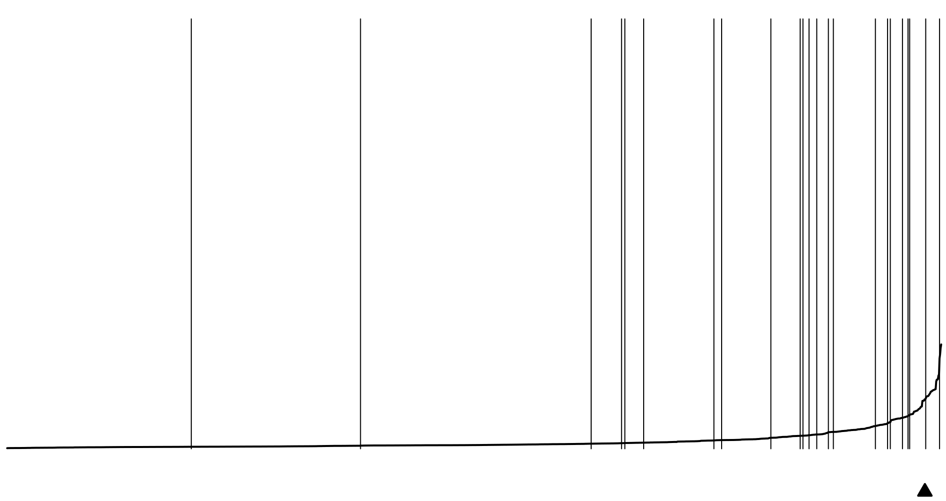
Although we have argued that models are not replicable, the seperation plots we generated for author's replicated models and the ones presented in the original paper are somewhat similar for all of the tree logistic regression models. The Random Forests model's seperation plot differs, and this is expected, as this is the model has the highest randomness with both downsampling and training control.

Our Own Seperation Plots

Let's draw the seperation plots for each instance.

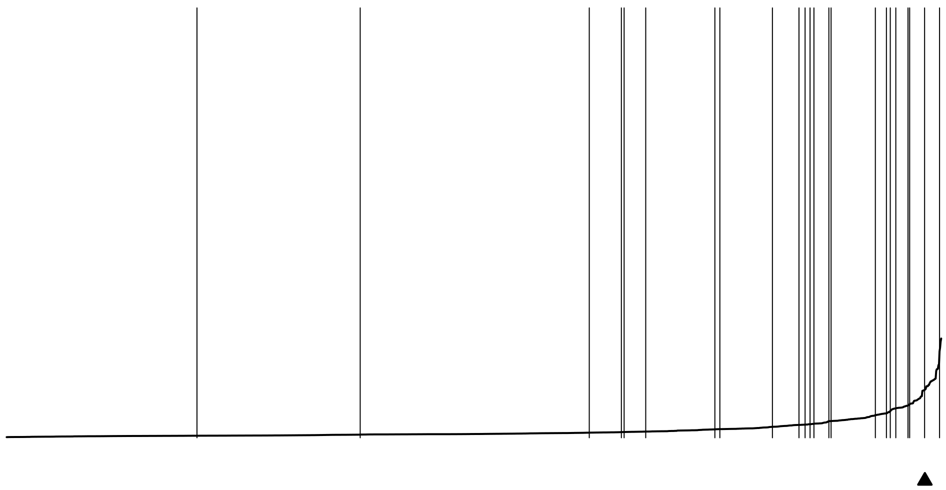
```
# Seperation plots for prediction with Fearon and Laitin specification (2003)
separationplot(FL_uncorrected_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Fearon and Laitin Spec (2003) Uncorrected LR Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

Fearon and Laitin Spec (2003) Uncorrected LR Seperation Plot



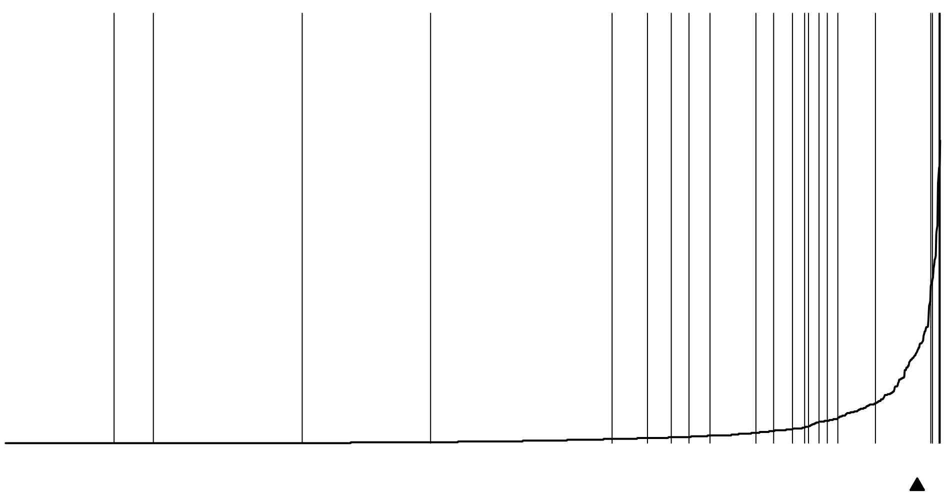
```r
separationplot(FL_penalized_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Fearon and Laitin Spec (2003) Penalized LR Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

Fearon and Laitin Spec (2003) Penalized LR Seperation Plot



```r
separationplot(FL_RF_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Fearon and Laitin Spec (2003) RF Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

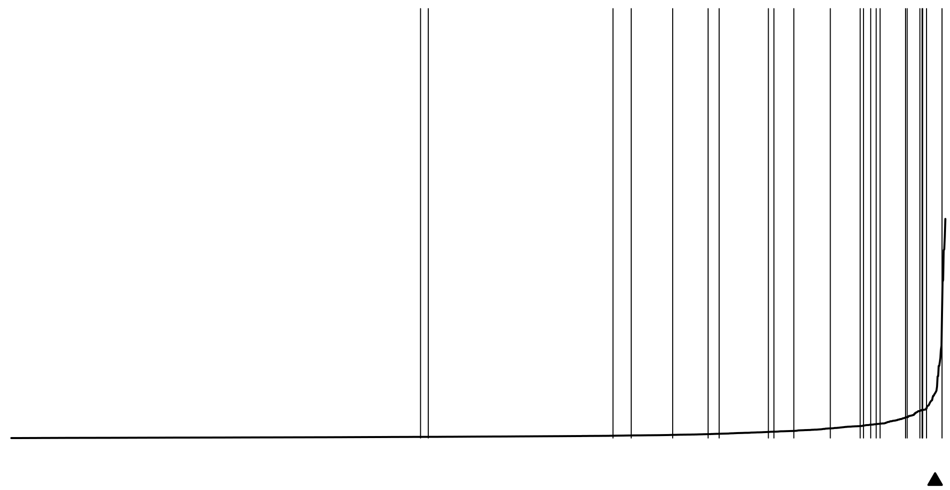
Fearon and Laitin Spec (2003) RF Seperation Plot



```r
# Seperation plots for prediction with Collier and Hoeffler specification (2004)
separationplot(CH_uncorrected_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
```

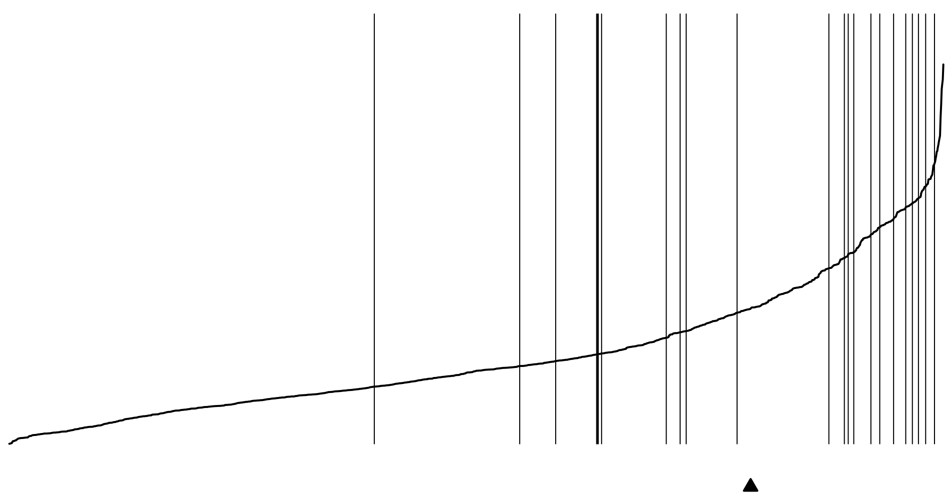
```r
             line=T, lwd2=1, show.expected=T,
             heading="Collier and Hoeffler Spec (2004) Uncorrected LR Seperation Plot",
             height=1.5, col0="white", col1="black", newplot=F)
```

Collier and Hoeffler Spec (2004) Uncorrected LR Seperation Plot



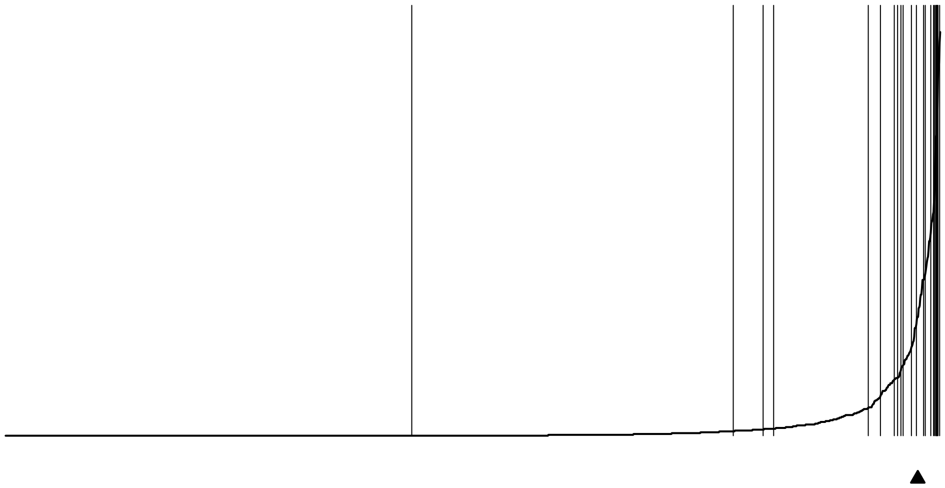
```r
separationplot(CH_penalized_pred_prob$war,
             as.numeric(data_test$warstds)-1, type="line",
             line=T, lwd2=1, show.expected=T,
             heading="Collier and Hoeffler Spec (2004) Penalized LR Seperation Plot",
             height=1.5, col0="white", col1="black", newplot=F)
```

Collier and Hoeffler Spec (2004) Penalized LR Seperation Plot



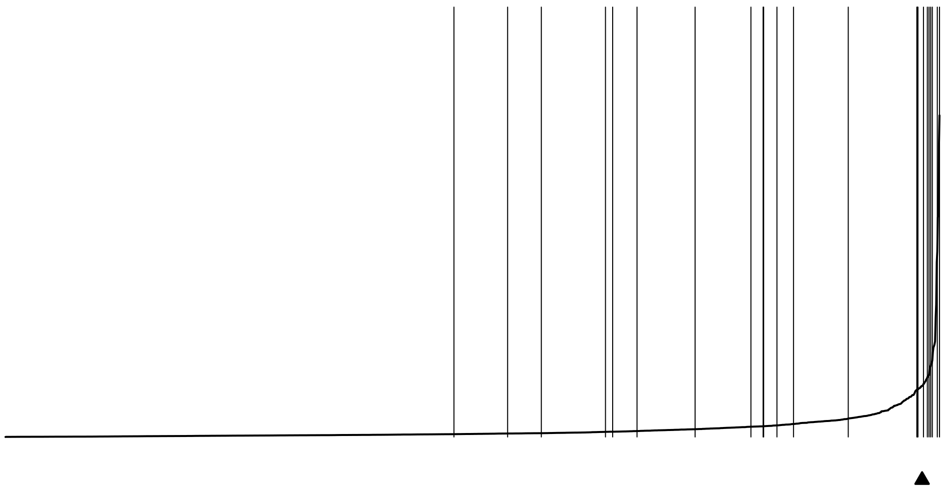
```r
separationplot(CH_RF_pred_prob$war,
             as.numeric(data_test$warstds)-1, type="line",
             line=T, lwd2=1, show.expected=T,
             heading="Collier and Hoeffler Spec (2004) RF Seperation Plot",
             height=1.5, col0="white", col1="black", newplot=F)
```

Collier and Hoeffler Spec (2004) RF Seperation Plot



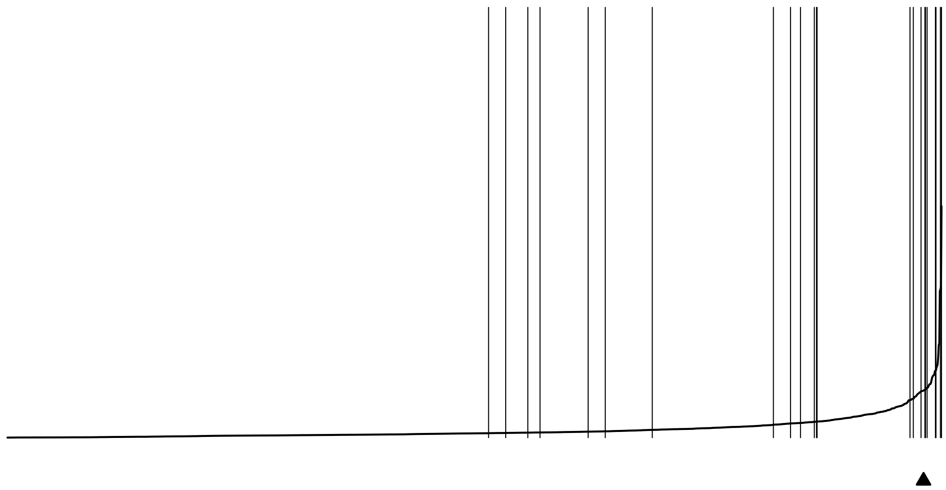
```r
# Seperation plots for prediction with Hegre and Sambanis specification (2006)
separationplot(HS_uncorrected_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Hegre and Sambanis Spec (2006) Uncorrected LR Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

Hegre and Sambanis Spec (2006) Uncorrected LR Seperation Plot



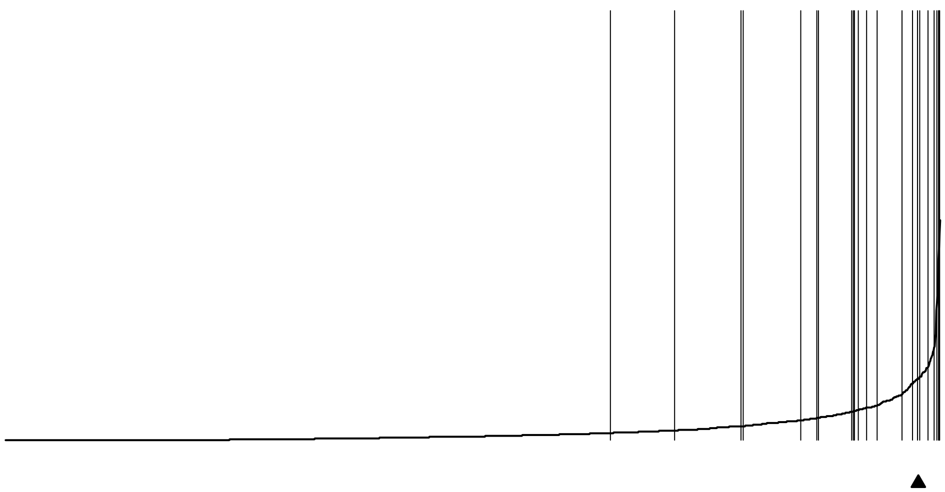
```r
separationplot(HS_penalized_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Hegre and Sambanis Spec (2006) Penalized LR Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

Hegre and Sambanis Spec (2006) Penalized LR Seperation Plot



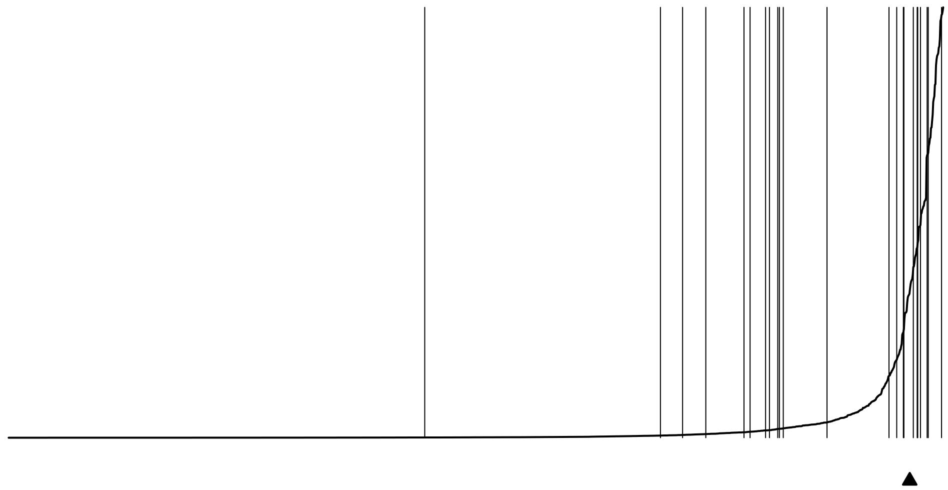
```
separationplot(HS_RF_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Hegre and Sambanis Spec (2006) RF Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

Hegre and Sambanis Spec (2006) RF Seperation Plot



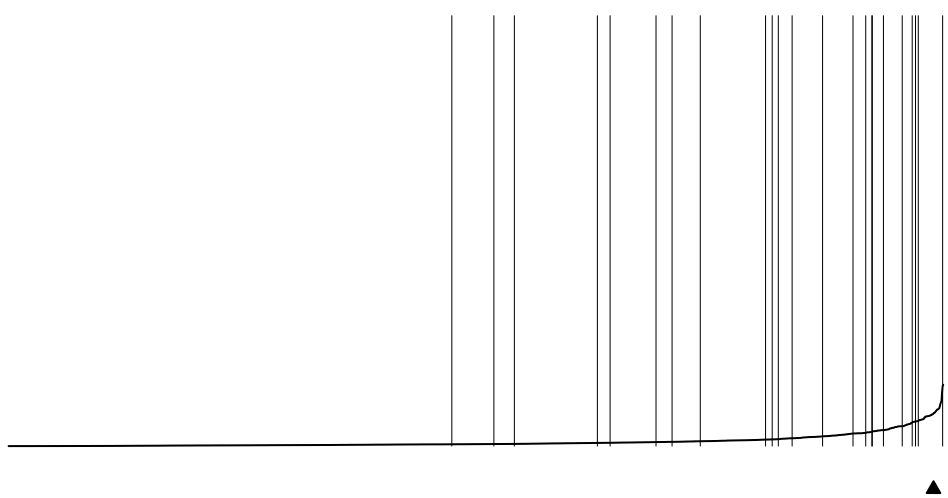
```
# Seperation plots for prediction with author specification (2016)
separationplot(AS_uncorrected_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Author Spec (2016) Uncorrected LR Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

Author Spec (2016) Uncorrected LR Seperation Plot



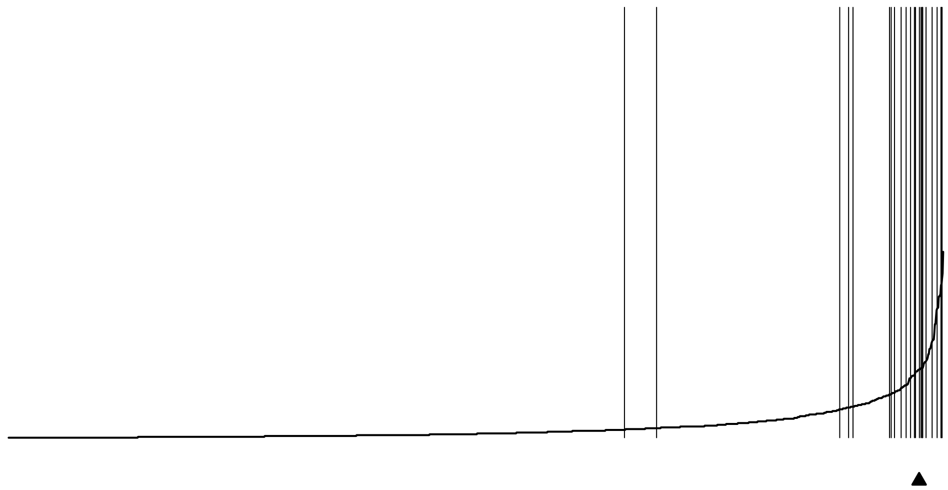
```r
separationplot(AS_penalized_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Author Spec (2016) Penalized LR Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

Author Spec (2016) Penalized LR Seperation Plot



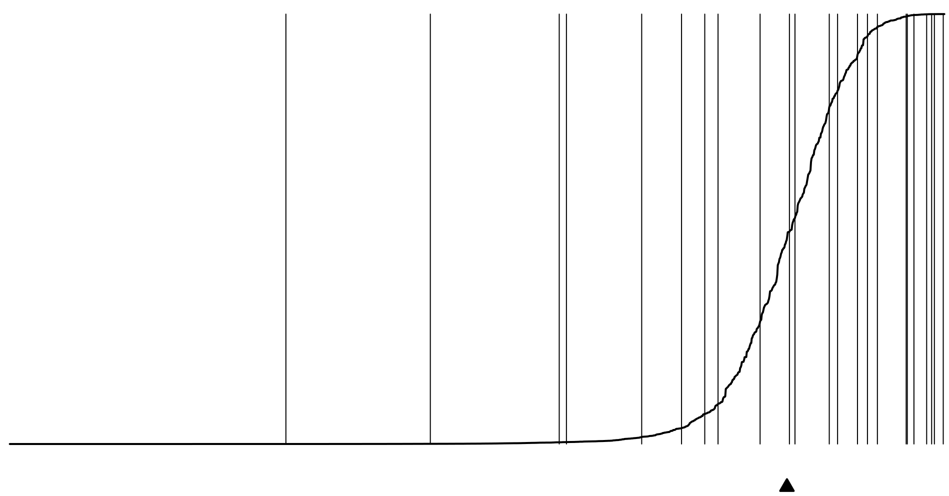
```r
separationplot(AS_RF_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Author Spec (2016) RF Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

Author Spec (2016) RF Seperation Plot



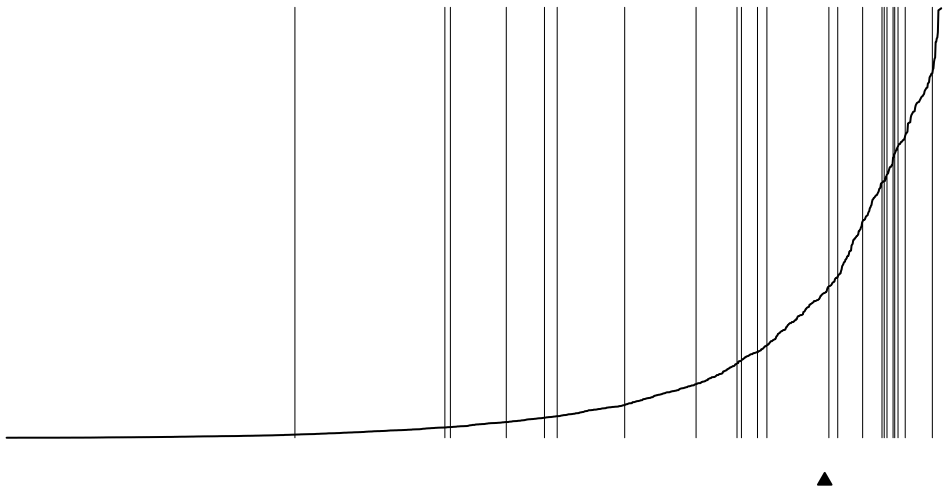
```r
# Seperation plots for prediction with author specification (2016) trained on SMOTEd dataset
separationplot(AS_uncorrected_smoted_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Author Spec (2016) Uncorrected LR (SMOTEd) Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

Author Spec (2016) Uncorrected LR (SMOTEd) Seperation Plot



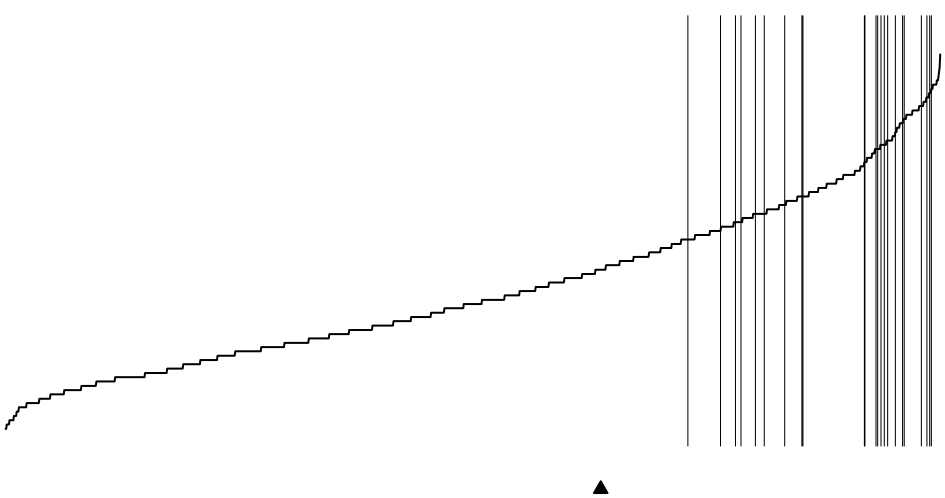
```r
separationplot(AS_penalized_smoted_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Author Spec (2016) Penalized LR (SMOTEd) Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

Author Spec (2016) Penalized LR (SMOTEd) Seperation Plot



```r
separationplot(AS_RF_smoted_pred_prob$war,
               as.numeric(data_test$warstds)-1, type="line",
               line=T, lwd2=1, show.expected=T,
               heading="Author Spec (2016) RF (SMOTEd) Seperation Plot",
               height=1.5, col0="white", col1="black", newplot=F)
```

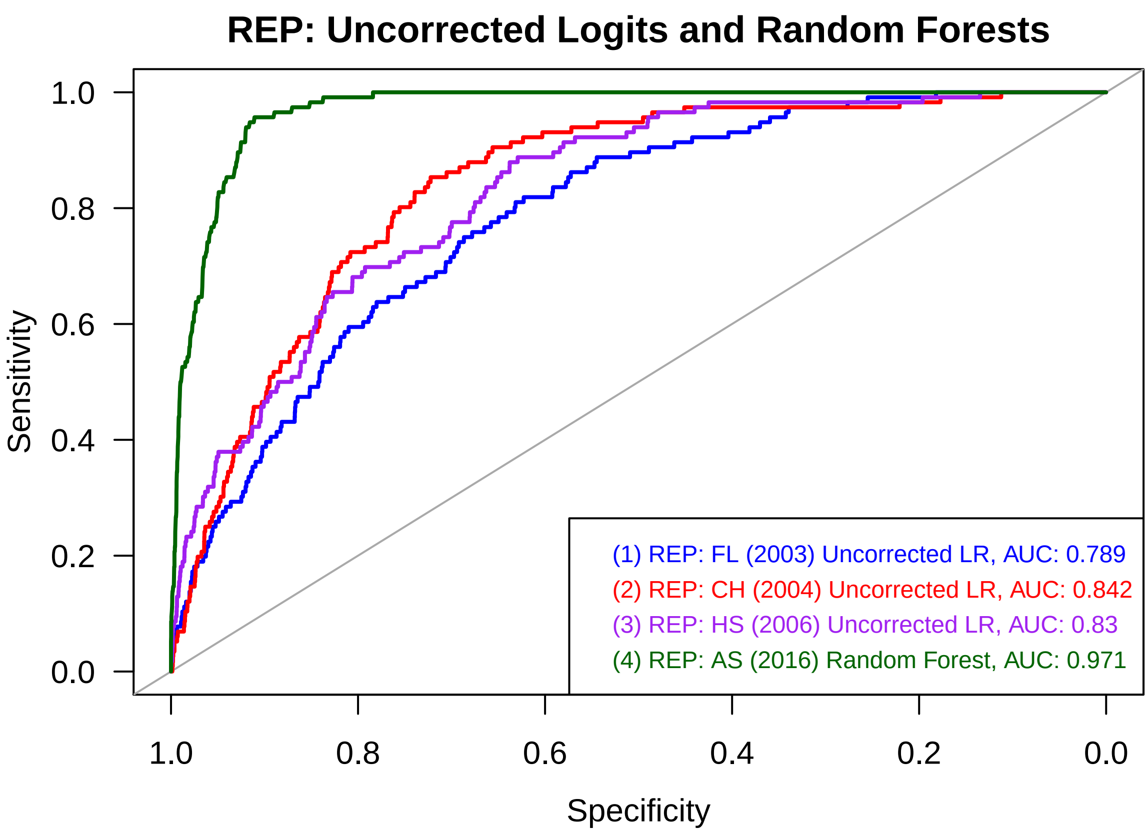
Author Spec (2016) RF (SMOTEd) Seperation Plot



Replicating Author's ROC Curves with AUC Scores

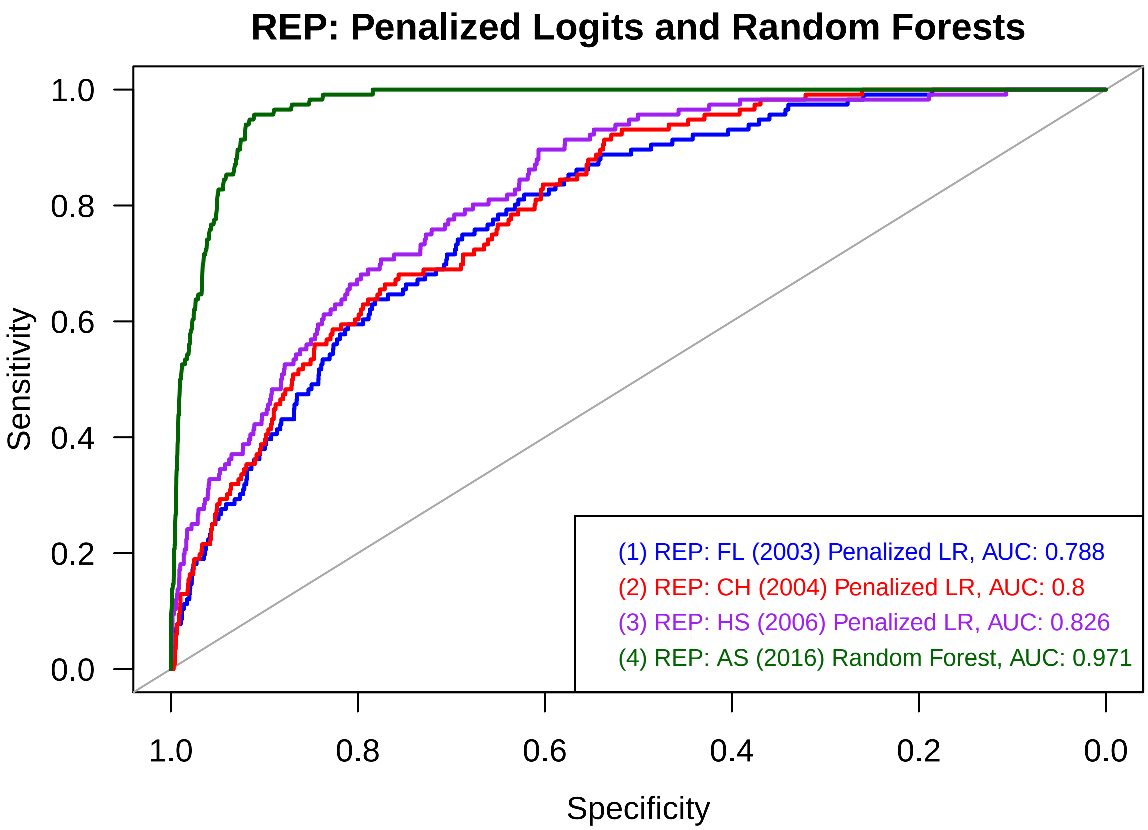
```r
# Plot ROC curves for prediction comparing uncorrected FL (2003), CH (2004),
# and HS (2006) specs, as well as the Random Forests models (author's versions)
plot.roc(data$warstds, REP_FL_uncorrected_pred_prob$war, col="blue",
         xlim=c(1,0), las=1, bty="n", asp=NA,
         main="REP: Uncorrected Logits and Random Forests")
plot.roc(data$warstds, add=T, REP_CH_uncorrected_pred_prob$war, col="red")
```

```
plot.roc(data$warstds, add=T, REP_HS_uncorrected_pred_prob$war, col="purple")
plot.roc(data$warstds, add=T, REP_AS_RF_pred_prob$war, col="darkgreen")
legend("bottomright", c(paste("(1) REP: FL (2003) Uncorrected LR, AUC:",
                              round(as.numeric(roc(data$warstds,
                                                   REP_FL_uncorrected_pred_prob$war)$auc),3)),
                        paste("(2) REP: CH (2004) Uncorrected LR, AUC:",
                              round(as.numeric(roc(data$warstds,
                                                   REP_CH_uncorrected_pred_prob$war)$auc),3)),
                        paste("(3) REP: HS (2006) Uncorrected LR, AUC:",
                              round(as.numeric(roc(data$warstds,
                                                   REP_HS_uncorrected_pred_prob$war)$auc),3)),
                        paste("(4) REP: AS (2016) Random Forest, AUC:",
                              round(as.numeric(roc(data$warstds,
                                                   REP_AS_RF_pred_prob$war)$auc),3))),
       text.col=c("blue","red","purple", "darkgreen"),
       cex = .75)
```




```
# Plot ROC curves for prediction comparing penalized FL (2003), CH (2004),
# and HS (2006) specs, as well as the Random Forests models (author's versions)
plot.roc(data$warstds, REP_FL_penalized_pred_prob$war, col="blue",
         xlim=c(1,0), las=1, bty="n", asp=NA,
         main="REP: Penalized Logits and Random Forests")
plot.roc(data$warstds, add=T, REP_CH_penalized_pred_prob$war, col="red")
plot.roc(data$warstds, add=T, REP_HS_penalized_pred_prob$war, col="purple")
plot.roc(data$warstds, add=T, REP_AS_RF_pred_prob$war, col="darkgreen")
legend("bottomright", c(paste("(1) REP: FL (2003) Penalized LR, AUC:",
                              round(as.numeric(roc(data$warstds,
```

```
                                              REP_FL_penalized_pred_prob$war)$auc),3)),
                        paste("(2) REP: CH (2004) Penalized LR, AUC:",
                              round(as.numeric(roc(data$warstds,
                                                    REP_CH_penalized_pred_prob$war)$auc),3)),
                        paste("(3) REP: HS (2006) Penalized LR, AUC:",
                              round(as.numeric(roc(data$warstds,
                                                    REP_HS_penalized_pred_prob$war)$auc),3)),
                        paste("(4) REP: AS (2016) Random Forest, AUC:",
                              round(as.numeric(roc(data$warstds,
                                                    REP_AS_RF_pred_prob$war)$auc),3))),
       text.col=c("blue","red","purple", "darkgreen"),
       cex = .75)
```



The un-replicability of the results are justified most with the reported ROC curves and accompanying AUC curves as shown above. It should be remembered that the curves and measures presented above are NOT tested with **out-of-sample** data, rather they simply act as a training summary. This explains the high AUC score of the random forest model.

ROC Curves with AUC Scores

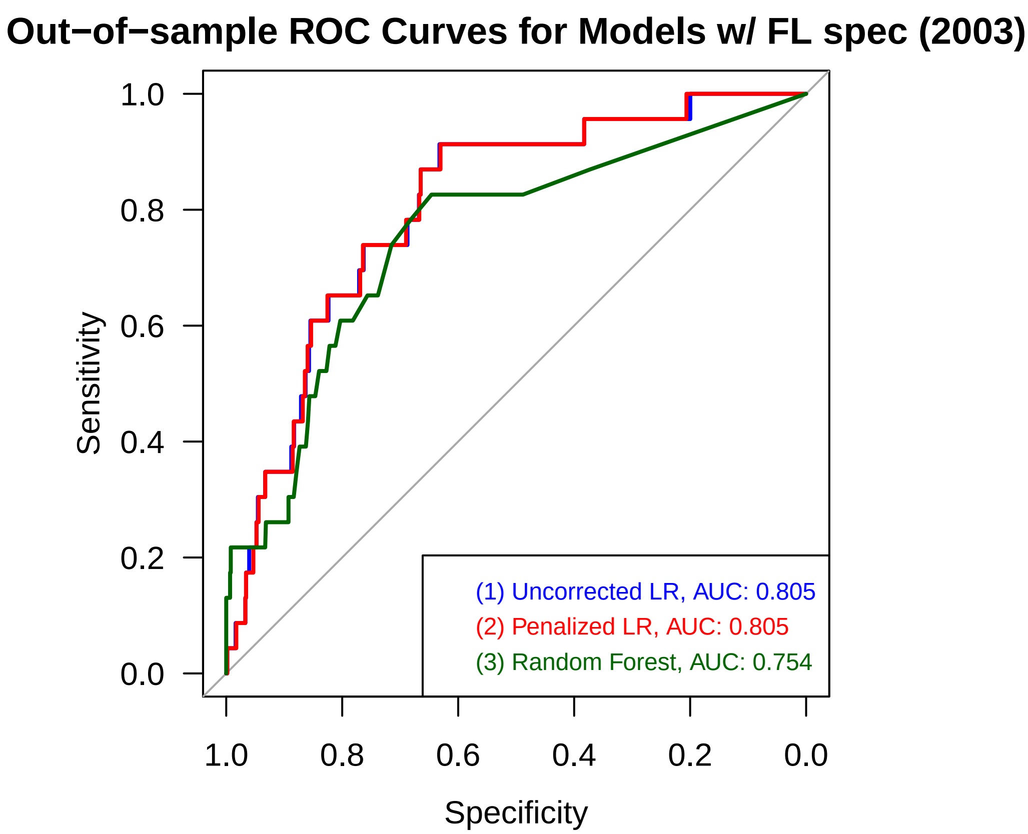
Let's draw ROC Curves with AUC metric for each instance. There are two possible groupings that could be applied:

i) We can group by specifications and assess the performance difference between uncorrected logistic regression, penalized logistic regression, and random forest algorithm for each of the specifications, or
ii) We can group by type of model and assess the performance difference in same models when different specifications with different numbers of variables and circumstances (SMOTE) applied.

Here, we will represent both.

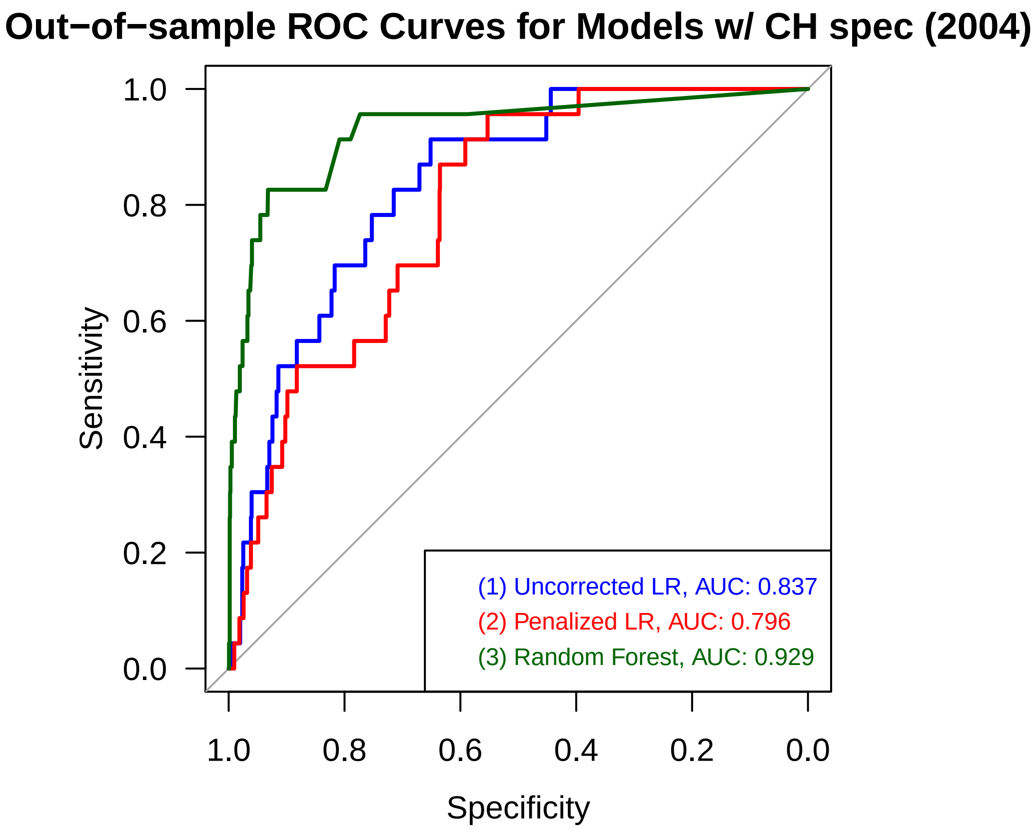
Grouped By Specifications

```
# Set for square-grid ROC plots
par(pty = "s")

# Plot ROC curves for prediction with Fearon and Laitin specification (2003)
plot.roc(data_test$warstds, FL_uncorrected_pred_prob$war, col="blue",
         xlim=c(1,0), las=1, bty="n", asp=NA,
         main="Out-of-sample ROC Curves for Models w/ FL spec (2003)")
plot.roc(data_test$warstds, add=T, FL_penalized_pred_prob$war, col="red")
plot.roc(data_test$warstds, add=T, FL_RF_pred_prob$war, col="darkgreen")
legend("bottomright", c(paste("(1) Uncorrected LR, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   FL_uncorrected_pred_prob$war)$auc),3)),
                        paste("(2) Penalized LR, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   FL_penalized_pred_prob$war)$auc),3)),
                        paste("(3) Random Forest, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   FL_RF_pred_prob$war)$auc),3))),
       text.col=c("blue","red","darkgreen"),
       cex = .75)
```



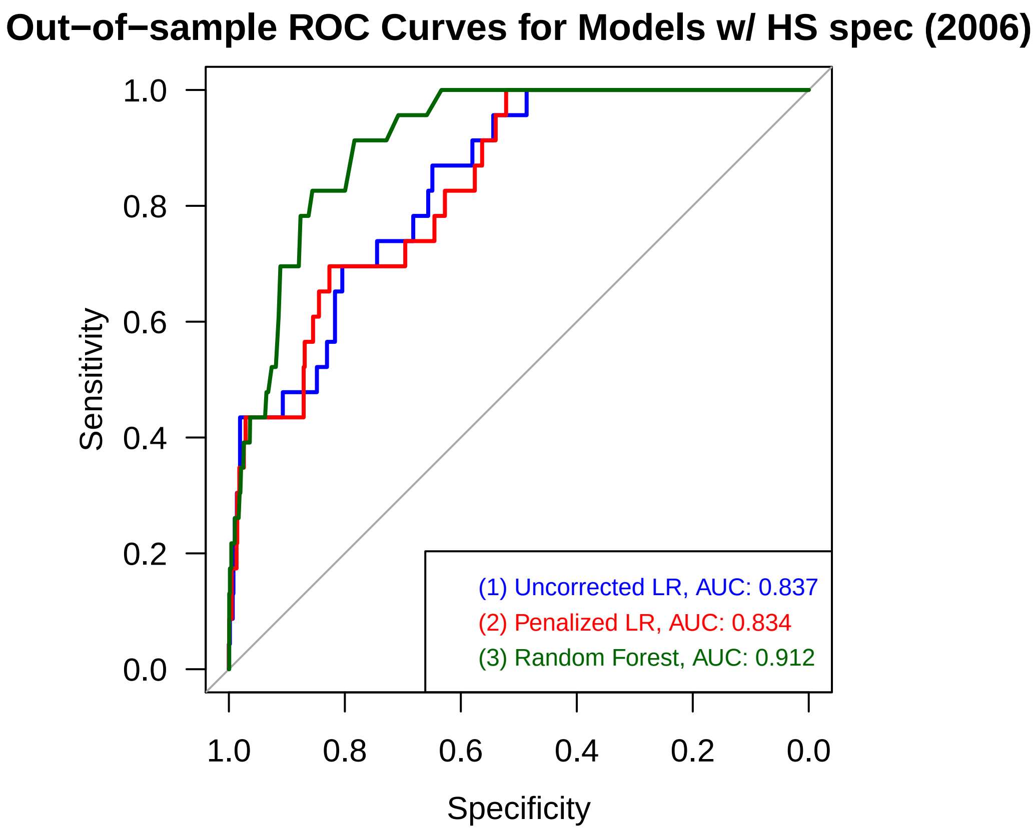

```
# Plot ROC curves for prediction with Collier and Hoeffler specification (2004)
plot.roc(data_test$warstds, CH_uncorrected_pred_prob$war, col="blue",
```

```
         xlim=c(1,0), las=1, bty="n", asp=NA,
         main="Out-of-sample ROC Curves for Models w/ CH spec (2004)")
plot.roc(data_test$warstds, add=T, CH_penalized_pred_prob$war, col="red")
plot.roc(data_test$warstds, add=T, CH_RF_pred_prob$war, col="darkgreen")
legend("bottomright", c(paste("(1) Uncorrected LR, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   CH_uncorrected_pred_prob$war)$auc),3)),
                        paste("(2) Penalized LR, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   CH_penalized_pred_prob$war)$auc),3)),
                        paste("(3) Random Forest, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   CH_RF_pred_prob$war)$auc),3))),
       text.col=c("blue","red","darkgreen"),
       cex = .75)
```

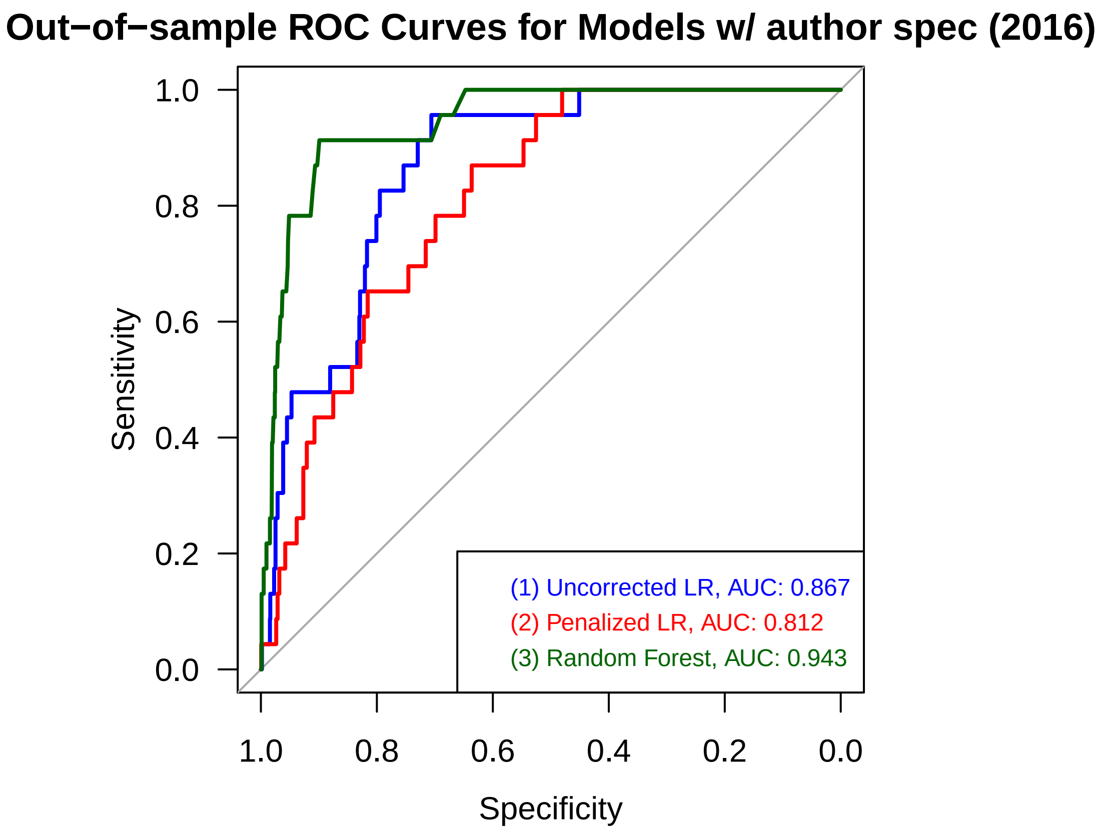



```
# Plot ROC curves for prediction with Hegre and Sambanis specification (2006)
plot.roc(data_test$warstds, HS_uncorrected_pred_prob$war, col="blue",
         xlim=c(1,0), las=1, bty="n", asp=NA,
         main="Out-of-sample ROC Curves for Models w/ HS spec (2006)")
plot.roc(data_test$warstds, add=T, HS_penalized_pred_prob$war, col="red")
plot.roc(data_test$warstds, add=T, HS_RF_pred_prob$war, col="darkgreen")
legend("bottomright", c(paste("(1) Uncorrected LR, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   HS_uncorrected_pred_prob$war)$auc),3)),
                        paste("(2) Penalized LR, AUC:",
                              round(as.numeric(roc(data_test$warstds,
```

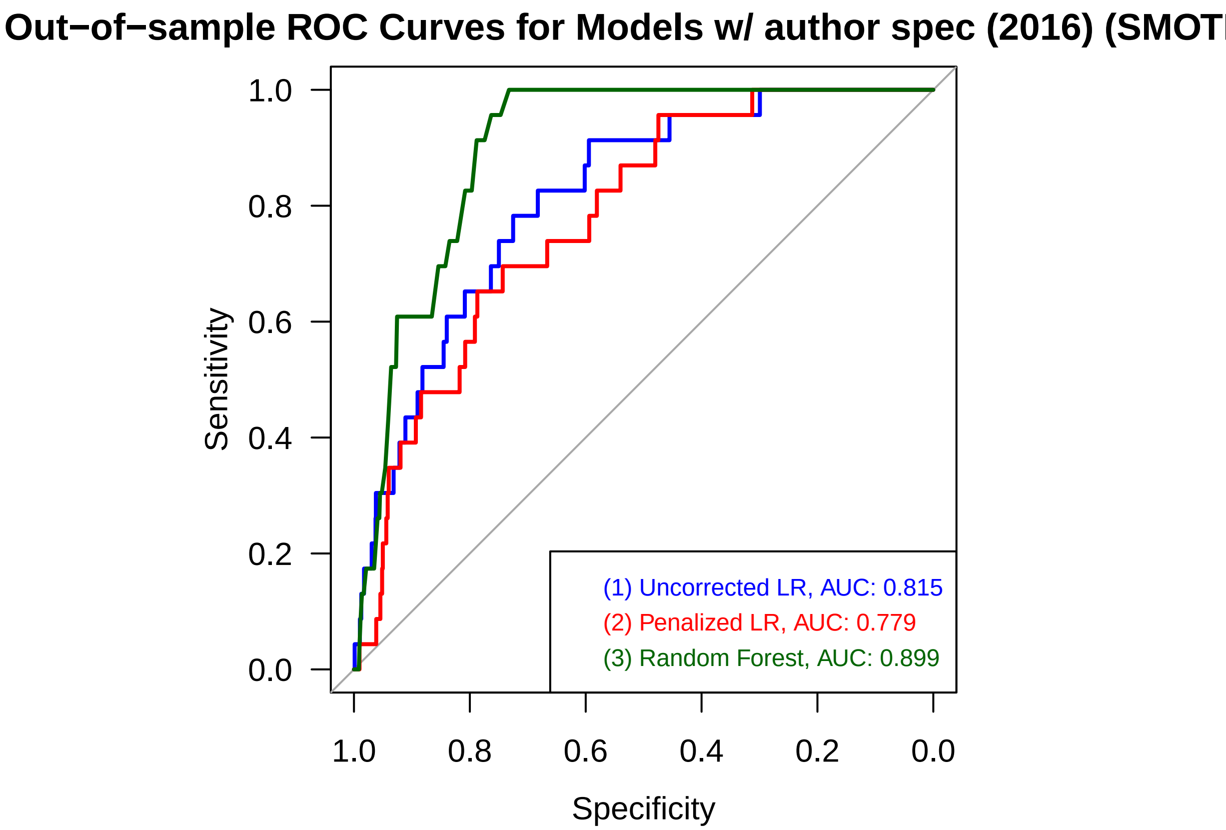
```
                                           HS_penalized_pred_prob$war)$auc),3)),
                        paste("(3) Random Forest, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                           HS_RF_pred_prob$war)$auc),3))),
       text.col=c("blue","red","darkgreen"),
       cex = .75)
```




```
# Plot ROC curves for prediction with author specification (2016)
plot.roc(data_test$warstds, AS_uncorrected_pred_prob$war, col="blue",
         xlim=c(1,0), las=1, bty="n", asp=NA,
         main="Out-of-sample ROC Curves for Models w/ author spec (2016)")
plot.roc(data_test$warstds, add=T, AS_penalized_pred_prob$war, col="red")
plot.roc(data_test$warstds, add=T, AS_RF_pred_prob$war, col="darkgreen")
legend("bottomright", c(paste("(1) Uncorrected LR, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                           AS_uncorrected_pred_prob$war)$auc),3)),
                        paste("(2) Penalized LR, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                           AS_penalized_pred_prob$war)$auc),3)),
                        paste("(3) Random Forest, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                           AS_RF_pred_prob$war)$auc),3))),
       text.col=c("blue","red","darkgreen"),
       cex = .75)
```



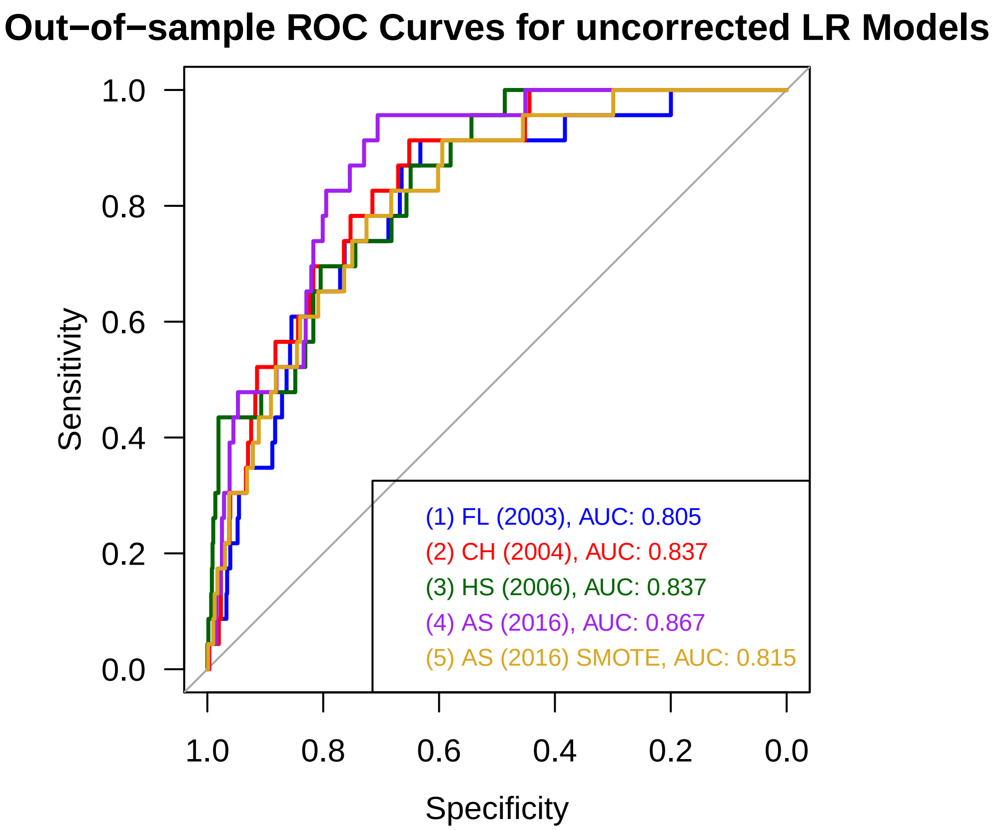

```
# Plot ROC curves for prediction with author specification (2016) trained SMOTEd dataset
plot.roc(data_test$warstds, AS_uncorrected_smoted_pred_prob$war, col="blue",
         xlim=c(1,0), las=1, bty="n", asp=NA,
         main="Out-of-sample ROC Curves for Models w/ author spec (2016) (SMOTEd)")
plot.roc(data_test$warstds, add=T, AS_penalized_smoted_pred_prob$war, col="red")
plot.roc(data_test$warstds, add=T, AS_RF_smoted_pred_prob$war, col="darkgreen")
legend("bottomright", c(paste("(1) Uncorrected LR, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   AS_uncorrected_smoted_pred_prob$war)$auc),3)),
                        paste("(2) Penalized LR, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   AS_penalized_smoted_pred_prob$war)$auc),3)),
                        paste("(3) Random Forest, AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   AS_RF_smoted_pred_prob$war)$auc),3))),
       text.col=c("blue","red","darkgreen"),
       cex = .75)
```



Grouped By Model Types

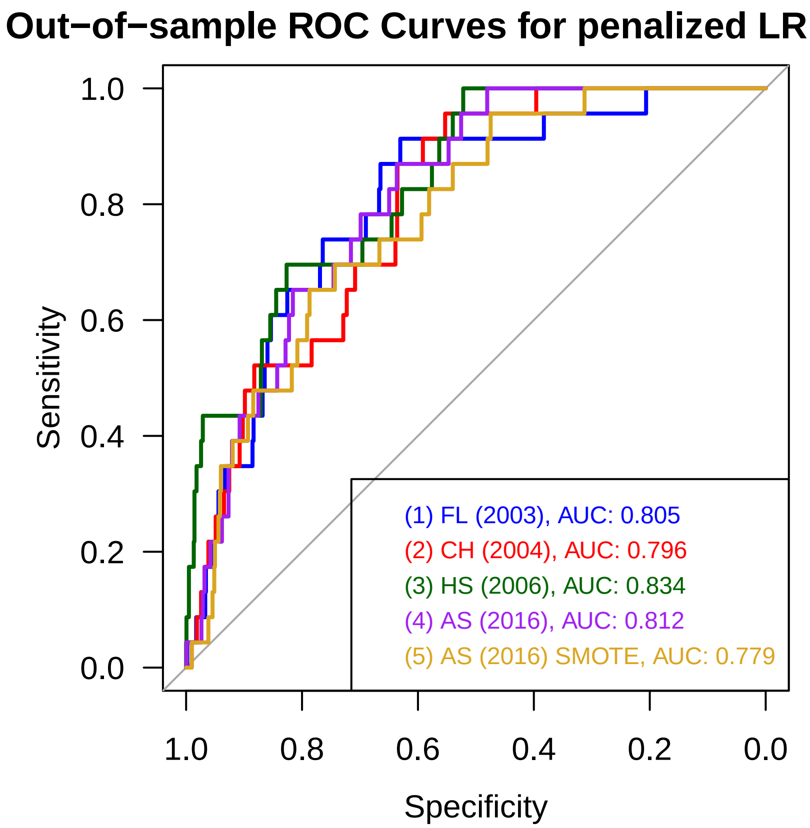
```
# Set for square-grid ROC plots
par(pty = "s")

# Plot ROC curves for prediction with uncorrected logistic regression model
plot.roc(data_test$warstds, FL_uncorrected_pred_prob$war, col="blue",
         xlim=c(1,0), las=1, bty="n", asp=NA,
         main="Out-of-sample ROC Curves for uncorrected LR Models")
plot.roc(data_test$warstds, add=T, CH_uncorrected_pred_prob$war, col="red")
plot.roc(data_test$warstds, add=T, HS_uncorrected_pred_prob$war, col="darkgreen")
plot.roc(data_test$warstds, add=T, AS_uncorrected_pred_prob$war, col="purple")
plot.roc(data_test$warstds, add=T, AS_uncorrected_smoted_pred_prob$war, col="goldenrod")

legend("bottomright", c(paste("(1) FL (2003), AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   FL_uncorrected_pred_prob$war)$auc),3)),
                        paste("(2) CH (2004), AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   CH_uncorrected_pred_prob$war)$auc),3)),
                        paste("(3) HS (2006), AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   HS_uncorrected_pred_prob$war)$auc),3)),
                        paste("(4) AS (2016), AUC:",
                              round(as.numeric(roc(data_test$warstds,
                                                   AS_uncorrected_pred_prob$war)$auc),3)),
```

```r
                      paste("(5) AS (2016) SMOTE, AUC:",
                            round(as.numeric(roc(data_test$warstds,
                                                 AS_uncorrected_smoted_pred_prob$war)$auc),3))),

       text.col=c("blue","red","darkgreen", "purple", "goldenrod"),
       cex = .75)
```



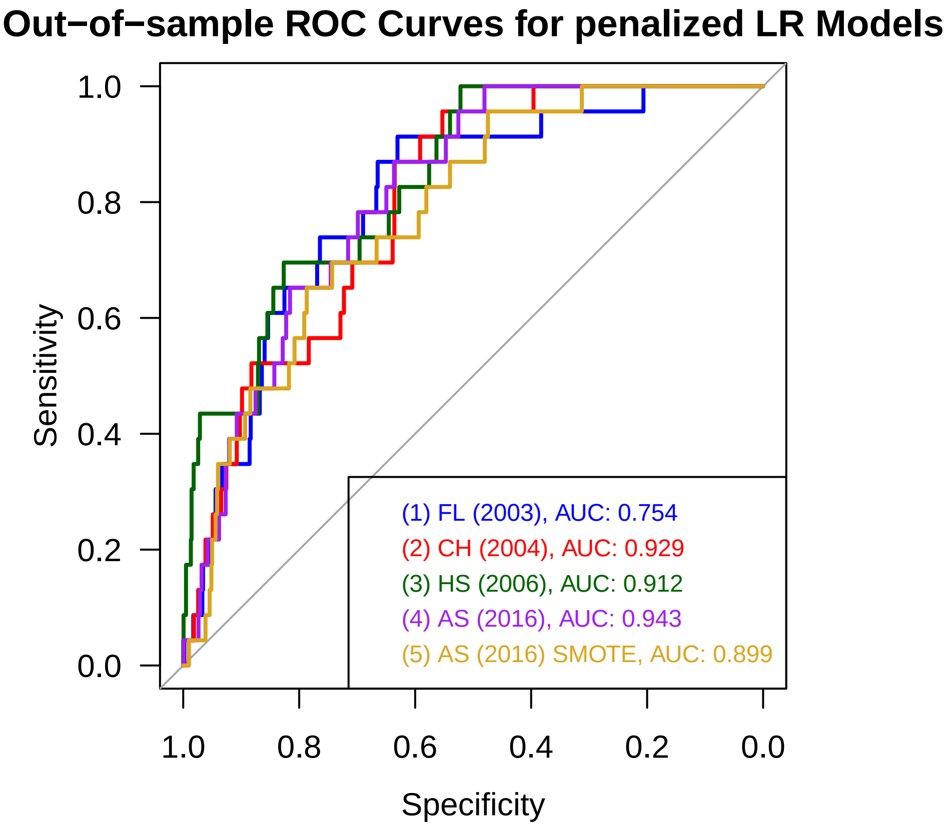

```r
# Plot ROC curves for prediction with penalized logistic regression model
plot.roc(data_test$warstds, FL_penalized_pred_prob$war, col="blue",
         xlim=c(1,0), las=1, bty="n", asp=NA,
         main="Out-of-sample ROC Curves for penalized LR Models")
plot.roc(data_test$warstds, add=T, CH_penalized_pred_prob$war, col="red")
plot.roc(data_test$warstds, add=T, HS_penalized_pred_prob$war, col="darkgreen")
plot.roc(data_test$warstds, add=T, AS_penalized_pred_prob$war, col="purple")
plot.roc(data_test$warstds, add=T, AS_penalized_smoted_pred_prob$war, col="goldenrod")

legend("bottomright", c(paste("(1) FL (2003), AUC:",
                            round(as.numeric(roc(data_test$warstds,
                                                 FL_penalized_pred_prob$war)$auc),3)),
                      paste("(2) CH (2004), AUC:",
                            round(as.numeric(roc(data_test$warstds,
                                                 CH_penalized_pred_prob$war)$auc),3)),
                      paste("(3) HS (2006), AUC:",
                            round(as.numeric(roc(data_test$warstds,
                                                 HS_penalized_pred_prob$war)$auc),3)),
                      paste("(4) AS (2016), AUC:",
                            round(as.numeric(roc(data_test$warstds,
```

```
                                            AS_penalized_pred_prob$war)$auc),3)),
                     paste("(5) AS (2016) SMOTE, AUC:",
                           round(as.numeric(roc(data_test$warstds,
                                            AS_penalized_smoted_pred_prob$war)$auc),3))),

       text.col=c("blue","red","darkgreen", "purple", "goldenrod"),
       cex = .75)
```




```
# Plot ROC curves for prediction with random forest model
plot.roc(data_test$warstds, FL_penalized_pred_prob$war, col="blue",
         xlim=c(1,0), las=1, bty="n", asp=NA,
         main="Out-of-sample ROC Curves for penalized LR Models")
plot.roc(data_test$warstds, add=T, CH_penalized_pred_prob$war, col="red")
plot.roc(data_test$warstds, add=T, HS_penalized_pred_prob$war, col="darkgreen")
plot.roc(data_test$warstds, add=T, AS_penalized_pred_prob$war, col="purple")
plot.roc(data_test$warstds, add=T, AS_penalized_smoted_pred_prob$war, col="goldenrod")

legend("bottomright", c(paste("(1) FL (2003), AUC:",
                           round(as.numeric(roc(data_test$warstds,
                                            FL_RF_pred_prob$war)$auc),3)),
                     paste("(2) CH (2004), AUC:",
                           round(as.numeric(roc(data_test$warstds,
                                            CH_RF_pred_prob$war)$auc),3)),
                     paste("(3) HS (2006), AUC:",
                           round(as.numeric(roc(data_test$warstds,
                                            HS_RF_pred_prob$war)$auc),3)),
                     paste("(4) AS (2016), AUC:",
```

```
                                  round(as.numeric(roc(data_test$warstds,
                                                       AS_RF_pred_prob$war)$auc),3)),
                           paste("(5) AS (2016) SMOTE, AUC:",
                                  round(as.numeric(roc(data_test$warstds,
                                                       AS_RF_smoted_pred_prob$war)$auc),3))),

       text.col=c("blue","red","darkgreen", "purple", "goldenrod"),
       cex = .75)
```



Dependencies Summary

The following is a list of all packages used to generate these results.

```
sessionInfo()
```

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS  10.14.4
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
```

```
## attached base packages:
## [1] grid      parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] separationplot_1.1  DMwR_0.4.1          maptools_0.9-5
##  [4] sp_1.3-1            foreign_0.8-71      forcats_0.3.0
##  [7] stringr_1.3.1       dplyr_0.7.5         purrr_0.2.5
## [10] readr_1.1.1         tidyr_0.8.1         tibble_1.4.2
## [13] tidyverse_1.2.1     doMC_1.3.5          iterators_1.0.10
## [16] foreach_1.4.4       stepPlr_0.93        pROC_1.14.0
## [19] ROCR_1.0-7          gplots_3.0.1.1      caret_6.0-84
## [22] ggplot2_2.2.1       lattice_0.20-35     rpart.plot_3.0.7
## [25] ada_2.0-5           rpart_4.1-13        randomForest_4.6-14
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-137       bitops_1.0-6       xts_0.11-2
##  [4] lubridate_1.7.4    httr_1.3.1         rprojroot_1.3-2
##  [7] tools_3.5.0        backports_1.1.2    R6_2.2.2
## [10] KernSmooth_2.23-15 lazyeval_0.2.1     colorspace_1.3-2
## [13] nnet_7.3-12        withr_2.1.2        tidyselect_0.2.4
## [16] mnormt_1.5-5       curl_3.2           compiler_3.5.0
## [19] cli_1.0.0          rvest_0.3.2        xml2_1.2.0
## [22] caTools_1.17.1     scales_1.0.0       psych_1.8.4
## [25] digest_0.6.15      rmarkdown_1.10     pkgconfig_2.0.1
## [28] htmltools_0.3.6    TTR_0.23-4         rlang_0.3.4
## [31] readxl_1.1.0       quantmod_0.4-14    rstudioapi_0.7
## [34] bindr_0.1.1        generics_0.0.2     zoo_1.8-5
## [37] jsonlite_1.5       gtools_3.8.1       ModelMetrics_1.2.2
## [40] magrittr_1.5       Matrix_1.2-14      Rcpp_0.12.17
## [43] munsell_0.5.0      abind_1.4-5        stringi_1.2.3
## [46] yaml_2.1.19        MASS_7.3-49        plyr_1.8.4
## [49] recipes_0.1.5      gdata_2.18.0       crayon_1.3.4
## [52] haven_1.1.1        splines_3.5.0      hms_0.4.2
## [55] knitr_1.20         pillar_1.2.3       reshape2_1.4.3
## [58] codetools_0.2-15   stats4_3.5.0       glue_1.2.0
## [61] evaluate_0.10.1    data.table_1.12.2  modelr_0.1.2
## [64] cellranger_1.1.0   gtable_0.2.0       assertthat_0.2.0
## [67] gower_0.2.0        prodlim_2018.04.18 broom_0.4.4
## [70] class_7.3-14       survival_2.41-3    timeDate_3043.102
## [73] bindrcpp_0.2.2     lava_1.6.5         ipred_0.9-9
```

#Comments regarding the reproducibility of the original paper eg. F1 score variation plot couldn't be replicated because the author's paper did not make these training test split ratios clear. We do not know the seed they used to shuffle their data. We can definitely split our data as per different ratios but we do not get exactly similar characteristics in the resulting training - test set (basically war - peace ratio samples)

#Summary of changes made to original paper and extensions i) The original paper used different feature variables for each of the four models. To make the comparison fair, we have tested the performance of the four models across all set of features specified in the paper. ii) Instead of measuring the performance only on the training data set, we do a train - test split to test performance on out of sample data iii) Since we are working with an imbalanced dataset, we have used SMOTE to generate a more balanced version of the data. We have trained our 4 models on this data and monitored their performance iv) We have also trained and checked the performance of new models - decision trees, boosting, neural networks

#Comparing our models / findings add detailed results and comparisons

#Final comments on the original paper

References

Related Papers & Datasets

StackOverflow for the Rescue!

- https://stackoverflow.com/questions/11225343/how-to-create-a-world-map-in-r-with-specific-countries-filled-in
- https://stackoverflow.com/questions/20624698/fixing-set-seed-for-an-entire-session
- https://stackoverflow.com/questions/42057979/proc-roc-curves-remove-empty-space
- https://stackoverflow.com/questions/30491213/the-union-of-several-vectors