Capstone - Establishment Success of Alien Mammals

John Hopkins

November 27th, 2019

Abstract

This document uses the data presented in a 2015 American Naturalist article 'Intraspecific Trait Variation Is Correlated with Establishment Success of Alien Mammals'[1]. This study focused on the relocation of mammals into a new environment and the features that most contributed to survival. In this paper, several strategies are explored to determine the best predictive bivariate model and corresponding important predictors.

Contents

Summary

Data collected on the introduction of mammals into a new environment supports the idea that certain factors play a key role in predicting subsequent survival.

The dataset contained information on 96 species spanning over 200 years and containing 57 features relating to the success or failure of a particular introduction.

After elimination of features and instances containing large numbers of missing data and balancing the dataset, a working set of 27 features was used to build a number of predictive models and resulting lists of most important features.

Reports were run against this data, illustrating the factors which most effect the success of the introduction of a species into a new environment.

```r
library(mlbench)
library(caret)
library(randomForest)
library(knitr)
library(kableExtra)
library(dplyr)
library(tidyr)
library(missCompare)
library(mice)
library(ROSE)
library(xgboost)
library(Matrix)
library(ggplot2)
library(gridExtra)
library(scales)
library(e1071)
library(MLmetrics)
```

Setup

The original dataset contained species introduction instances from pre-19th century dates, which were eliminated.

The impact of categorical features was low with regard to prediction. Categorical features, with the exception of species name, were converted into factors, backed up and removed. This information will be used later on for reporting purposes.

All features containing coefficients of variation, which showed a high level of missing data, were also removed.

```r
df.orig <- read.csv("../data/Gonzalez-Suarez_et_al_Dataset.csv", header=T, stringsAsFactors=F)

## rearrange / reclassify char columns
df.orig <- df.orig %>%
  select(everything()) %>%
  filter(Year_introduction >= 1800)

df.orig <- df.orig %>% select(-Establishment_success,Establishment_success)

df.orig$Year_introduction <-  as.character(df.orig$Year_introduction)

for( col in names(df.orig[, sapply(df.orig, class) == 'character'])) {
  colnbr <- which(colnames(df.orig) == col )
  df.orig[,colnbr] <- (factor(df.orig[,colnbr] ))
  print(paste( col))
  df.orig <- df.orig %>% select(-col,col)
}
```

```
## [1] "Species_name"
## [1] "Taxonomic_order"
## [1] "Taxonomic_family"
## [1] "Taxonomic_genus"
## [1] "Year_introduction"
## [1] "Biogeographic_region"
## [1] "Nation_introduction"
```

```r
df.orig.dvar <- which( colnames(df.orig)=="Establishment_success" )

df <- df.orig

remove.named.cols <- function(dt, nme.tok) {
  for (i in sort(1:ncol(dt), decreasing = T)) {
    #print(paste(nme.tok, i, names(df)[i]))
    if (grepl(nme.tok, names(dt)[i])) {
      dt[,i] <- NULL
    }
  }
  return(dt)
}

df  <- remove.named.cols(df, 'CV_')
dvar <- which( colnames(df)=="Establishment_success" )

for (i in (dvar + 1): (length(df))) {
  df[,length(df)] <- NULL
}
```

Remove Outliers

The data was examined to see if removal of outliers was necesary. As proved to be true, steps were taken to mark outliers as NA. The algorythm used to do this employed Tukey's method to identify the outliers ranged above and below the 1.5 * IQR. [2]

```
post.ol.stats <- function(nme, ol.nbr, ol.prop,  ol.mean, ol.mean.before, ol.mean.after ) {
  tmp <- data.frame( Name     = nme,
                     Count = ol.nbr,
                     Prop   = ol.prop,
                     Mean   = ol.mean,
                     MeanPrior = ol.mean.before,
                     MeanAfter = ol.mean.after )
  rownames(tmp) <- ''
  ol.stats <-rbind(ol.stats, tmp)
  return (ol.stats)
}

outlierKD <- function(dt, nme) {
  var_name <- dt[[nme]]
  na1 <- sum(is.na(var_name))
  m1 <- mean(var_name, na.rm = T)
  outlier <- boxplot(var_name, plot=FALSE)$out
  mo <- mean(outlier)
  var_name <- ifelse(var_name %in% outlier, NA, var_name)
  na2 <- sum(is.na(var_name))
  m2 <- mean(var_name, na.rm = T)
  dt[[nme]] <- invisible(var_name)
  assign(as.character(as.list(match.call())$dt), dt, envir = .GlobalEnv)
  ol.nbr  <- (na2 - na1)
  ol.prop <- round((na2 - na1) / sum(!is.na(var_name))*100, 1)
  ol.mean <- round(mo, 2)
  ol.mean.before <- round(m1, 2)
  ol.mean.after <- round(m2, 2)
  ol.tmp <- post.ol.stats(nme, ol.nbr , ol.prop,  ol.mean, ol.mean.before, ol.mean.after )
  return(ol.tmp)
}
ol.stats <- data.frame()

for (i in 1:26) {
  ol.stats <- outlierKD(df, names(df)[i])
}
```

Table 1: Outliers

	Name	Count	Prop	Mean	MeanPrior	MeanAfter
	Mean_adult_body_mass_g	50	11.5	349801.24	61647.80	28450.40
1	N_adult_body_mass_g	60	14.2	32.25	12.99	10.27
2	Mean_neonate_body_neonate_body_mass_g	45	11.5	14387.45	2612.20	1260.45
3	N_neonate_body_mass_g	36	9.0	27.14	9.59	8.02
4	Mean_home_range_size_km2	38	14.0	48.82	8.19	2.49
5	N_home_range_size_km2	33	12.0	23.64	5.66	3.51
6	Mean_population_density_ind_km2	57	28.9	1495.97	366.37	39.54
7	N_population_density_ind_km2	22	9.5	23.73	6.52	4.88
8	Mean_group_size	16	14.3	163673.55	20470.05	12.41
9	N_group_size	0	0.0	NaN	4.38	4.38
10	Mean_litter_per_year	28	17.5	4.50	2.27	1.88
11	N_litter_per_year	39	26.2	43.00	14.23	6.70
12	Mean_interbirth_interval	2	0.8	906.04	294.45	289.79
13	N_interbirth_interval	28	11.9	7.11	2.99	2.50
14	Mean_weaning_age_d	31	8.2	314.91	95.45	77.35
15	N_weaning_age_d	11	2.8	22.91	7.97	7.55
16	Mean_sexual_maturity_age_d	28	6.8	1327.05	491.38	434.86
17	N_sexual_maturity_age_d	2	0.5	32.00	10.79	10.69
18	Mean_litter_size	0	0.0	NaN	3.21	3.21
19	N_litter_size	0	0.0	NaN	15.41	15.41
20	Mean_gestation_length_d	0	0.0	NaN	119.92	119.92
21	N_gestation_length_d	21	4.7	44.00	14.32	12.94
22	Number_records	0	0.0	NaN	81.41	81.41
23	Native_geographic_range_area_km2	6	1.3	53398716.45	7823035.31	7244908.27
24	Habitat_breadth	0	0.0	NaN	8.22	8.22
25	Propagule_pressure	68	16.0	1347.04	209.56	27.56

```r
kable(ol.stats, caption = "Outliers")
```

Clean Impute Scale

The data was next filtered to remove any rows and columns which contained missing data exceeding a given thresthold.

After that, the data was imputed using the mice utility [3]. This replaced all missing data with a calculated value, based on the mice algorithm.

Lastly, the data was centered and scaled, in prepatation for subsequent modeling and checked for near-Zero variance.

```r
df.clean <- missCompare::clean(df,
                               var_removal_threshold = 0.5,
                               ind_removal_threshold = 0.8,
                               missingness_coding = -9)
```

```
## Variable(s) Mean_population_density_ind_km2, N_population_density_ind_km2, Mean_group_size, N_group_
```

```r
kable(sapply(df.clean, function(x) sum(is.na(x))), caption = "Missing Data")
```

```r
dvar <- which( colnames(df)=="Establishment_success" )
```

Table 2: Missing Data

	x
Mean_adult_body_mass_g	59
N_adult_body_mass_g	69
Mean_neonate_body_neonate_body_mass_g	101
N_neonate_body_mass_g	92
Mean_home_range_size_km2	222
N_home_range_size_km2	217
Mean_interbirth_interval	231
Mean_weaning_age_d	117
N_weaning_age_d	97
Mean_sexual_maturity_age_d	79
N_sexual_maturity_age_d	53
Mean_litter_size	9
N_litter_size	9
Mean_gestation_length_d	21
N_gestation_length_d	42
Number_records	0
Native_geographic_range_area_km2	20
Habitat_breadth	29
Propagule_pressure	68
Establishment_success	0

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

## impute missing values

df.mice <- mice(df.clean, m=5, maxit = 20, method = 'cart', seed = 500, print=F)
df <- mice::complete(df.mice ,1)

# center / scale data
df.pp <- preProcess(df[, -dvar],
                        method = c("center", "scale", "YeoJohnson", "nzv"))
df.pp
```

```
## Created from 493 samples and 20 variables
##
## Pre-processing:
##   - centered (20)
##   - ignored (0)
##   - scaled (20)
##   - Yeo-Johnson transformation (19)
##
## Lambda estimates for Yeo-Johnson transformation:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.58588 -0.01749  0.20073  0.15516  0.46749  1.01515
```

```r
df.pp <- predict(df.pp, newdata = df[, -dvar])

df.pp$Establishment_success = factor(df$Establishment_success)
df <- df.pp
```

```
## check for near zero predictors

nzv <- nearZeroVar(df)
nzv
```

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

Balance Dataset

Upon examination, the data proved to be unbalanced. Using the ROSE Package [4], several balancing methods were applied in an effort to find the one which would best maximize sensitivity/specificity and accuracy / balanced accuracy.

The over method produced the best overall results and so was used to drive model selection.

```
## data is unbalanced and requires balancing
dvar <- which( colnames(df)=="Establishment_success" )
cbind(freq=table(df[,dvar]), percentage=prop.table(table(df[,dvar]))*100)
```

```
##   freq percentage
## 0  116   23.52941
## 1  377   76.47059
```

```
set.seed(777)
train.no.cnt <- summary(df$Establishment_success)['0']
train.yes.cnt <- summary(df$Establishment_success)['1']
train.both.cnt <- train.no.cnt + train.yes.cnt

## create training samples to deal with unbalanced class

train.full <- list(
  train = df,
  over  = ovun.sample(Establishment_success ~ ., data = df,
                      method = "over", N = train.yes.cnt * 2)$data,
  under = ovun.sample(Establishment_success ~ ., data = df,
                      method = "under", N = train.no.cnt * 2)$data,
  both  = ovun.sample(Establishment_success ~ ., data = df, p = 0.5,
                      seed = 777, method = "both", N = train.both.cnt)$data,
  rose  = ROSE(Establishment_success ~ ., data = df, N = train.no.cnt * 2,
               seed=111)$data
)

# determine the best balancing method
cm <- data.frame()
for( i in 1:length(train.full)) {
  # print(paste(i))
  dt <- train.full[[i]]
  learn_rf <- randomForest(dt$Establishment_success ~ .,
                           data=dt, ntree=500,
                           proximity=T, importance=T, na.action=na.roughfix)
  pre_rf <- predict(learn_rf, df[,-dvar])
  cfm <- confusionMatrix(pre_rf, df$Establishment_success)
  stats <- data.frame( name = names(train.full)[i],
                       Accuracy = cfm$overall['Accuracy'],
                       AccuracyNull = cfm$overall['AccuracyNull'],
```

Table 3: Balancing Methods

name	Accuracy	AccuracyNull	Sensitivity	Specificity	Balanced_Accuracy
train	0.9290061	0.7647059	0.7413793	0.9867374	0.8640584
over	0.9148073	0.7647059	0.8620690	0.9310345	0.8965517
under	0.7849899	0.7647059	0.8620690	0.7612732	0.8116711
both	0.8782961	0.7647059	0.8275862	0.8938992	0.8607427
rose	0.6328600	0.7647059	0.7672414	0.5915119	0.6793767

```
                       Sensitivity = cfm$byClass['Sensitivity'],
                       Specificity = cfm$byClass['Specificity'],
                       Balanced_Accuracy = cfm$byClass['Balanced Accuracy'],
                       stringsAsFactors = FALSE)
  rownames(stats) <- c()
  cm <- rbind(cm, stats)
}

kable(cm, caption = "Balancing Methods")
```

```
## over method maximizes sensitivity/specificity and balanced accuracy
df <- train.full[['over']]
```

Test Models

Several models were run against the data. Information from their corresponding confusionMatrixes was captured for comparison upon their completion.

- Random Forest
- GLM
- XGBoost

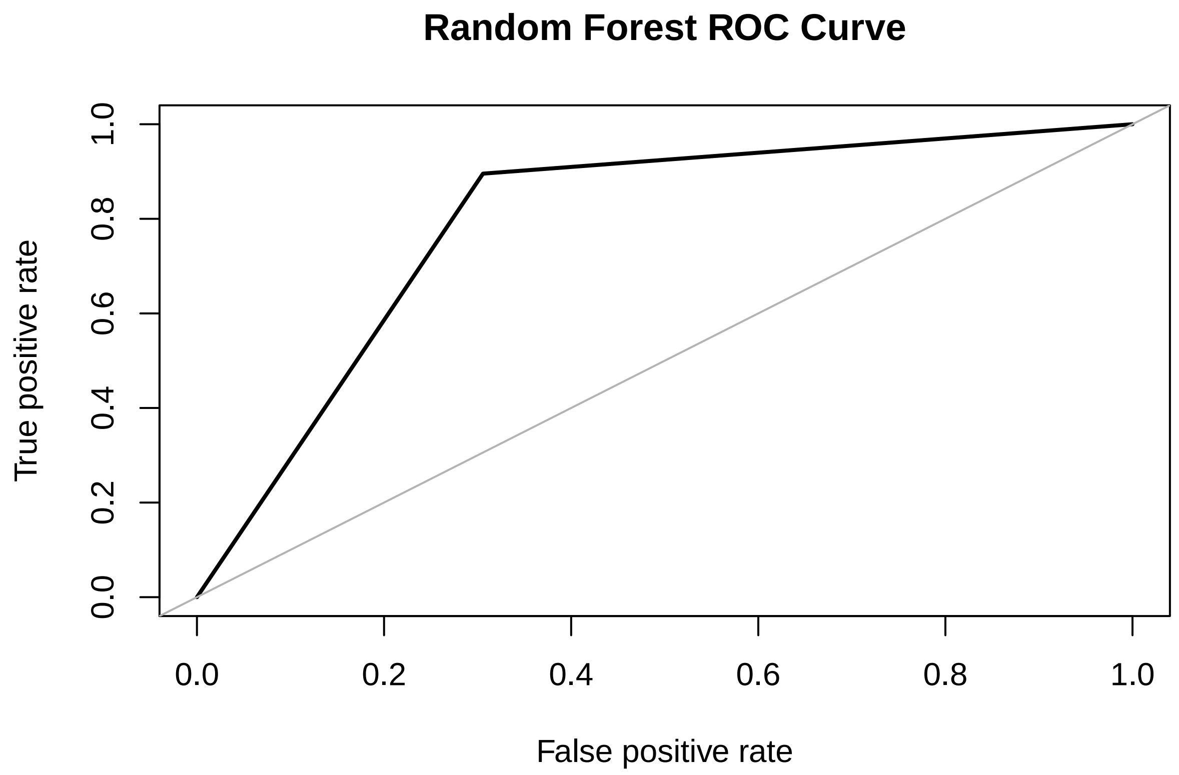
```
cm.stats <- data.frame()

post.cm.stats <- function(meth,cmtx,auc) {
  tmp <- data.frame( Method = meth,
                     Accuracy = cmtx$overall[1],
                     Sensitivity = cmtx$byClass[1],
                     Specificity = cmtx$byClass[2],
                     BalAccuracy = cmtx$byClass[6],
                     AUC = auc
  )
  rownames(tmp) <- ''
  cm.stats <-rbind(cm.stats, tmp)
  return (cm.stats)
}
```

Random Forest

A random forest model was contructed against the data and the results further explored to identify which features stood out as most important. [4]

```
set.seed(222)
ind <- sample(2, nrow(df), replace = T, prob = c(0.6, 0.4))
train <- df[ind==1,]
```

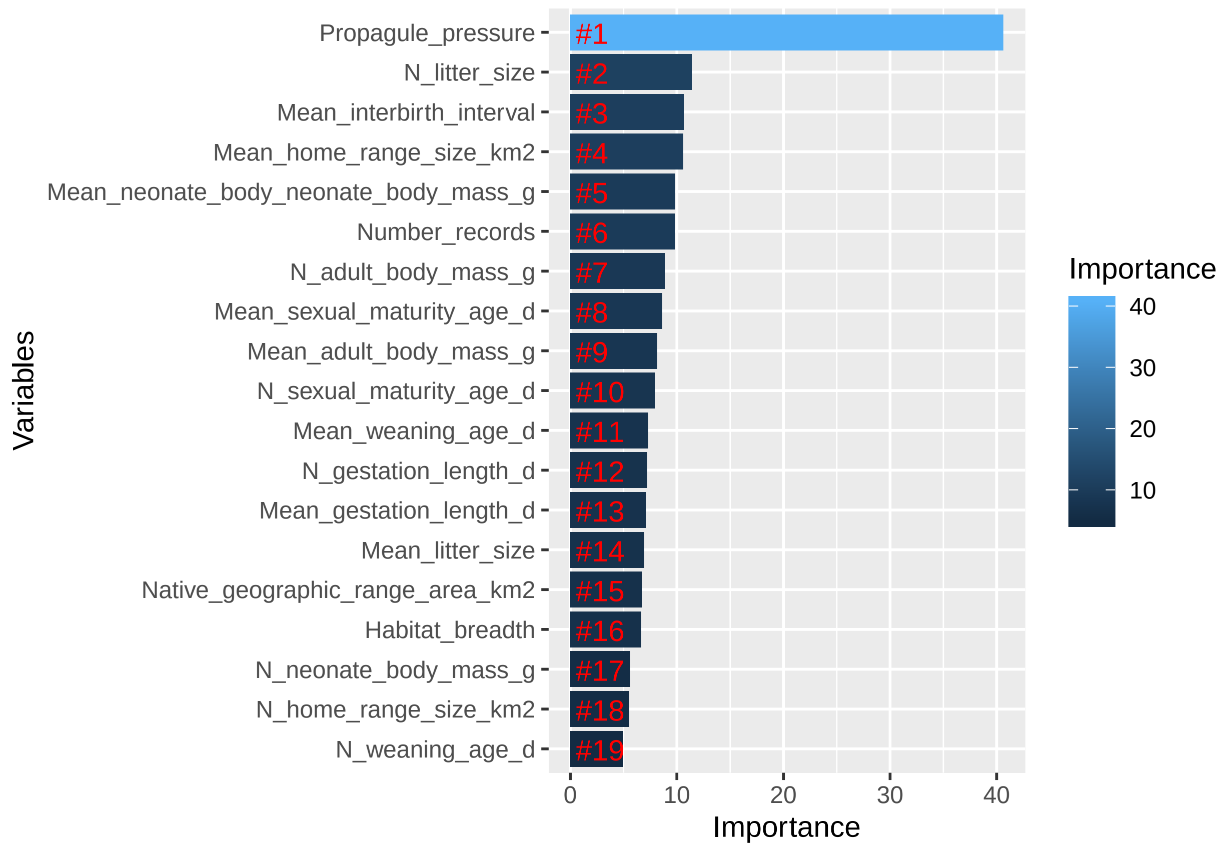
```
test <- df[ind==2,]

rf01 <- randomForest(train$Establishment_success ~ ., data=train,
                     ntree=500, proximity=T, importance=T, na.action=na.roughfix)
rf01.pred <- predict(rf01, test[,-dvar])
rf01.roc <- roc.curve( test[,dvar], rf01.pred)
rect(0, 1.1, 1, 1.7, xpd=TRUE, col="white", border="white")
title("Random Forest ROC Curve")
```



```
print(rf01.roc$auc)
```

```
## [1] 0.794895
```

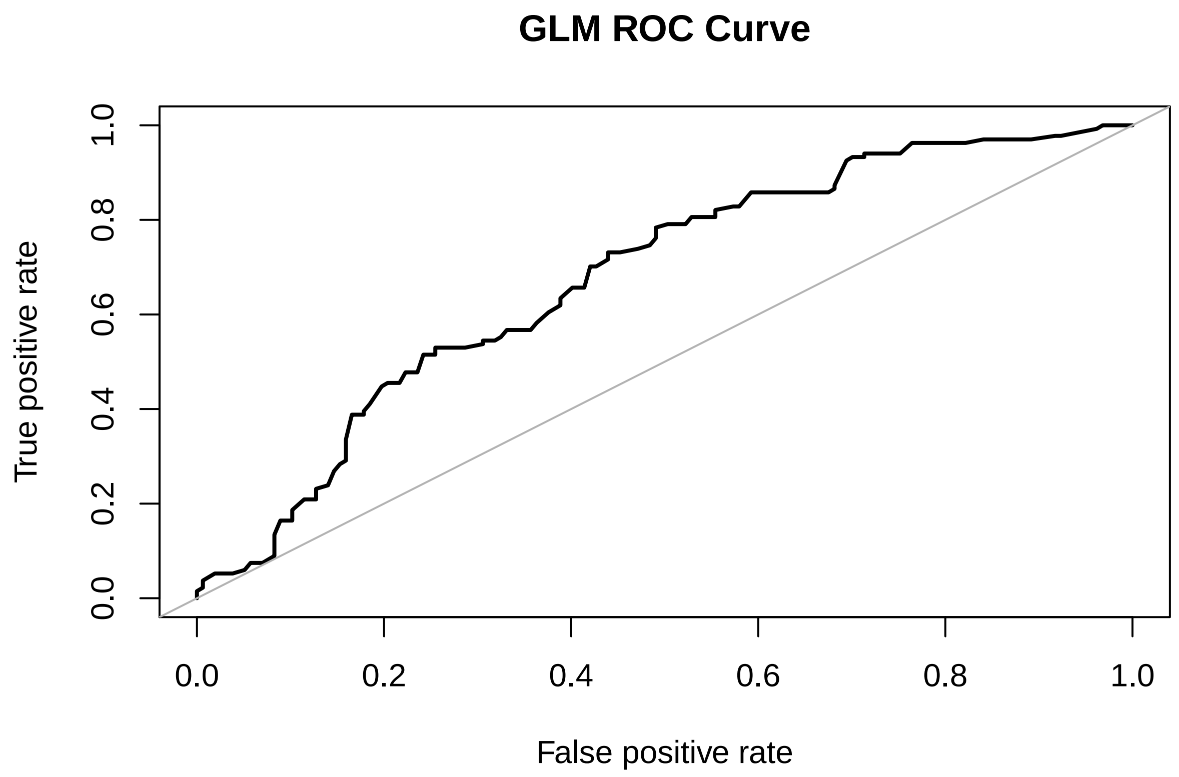
```
# Prediction & Confusion Matrix - Test
rf01.cm <-confusionMatrix(rf01.pred, test[, dvar])

cm.stats <- post.cm.stats('randomForest',rf01.cm, rf01.roc$auc)

importance    <- importance(rf01)
varImportance <- data.frame(Variables = row.names(importance),
                            Importance = round(importance[ ,'MeanDecreaseGini'],2))

rankImportance <- varImportance %>%
  mutate(Rank = paste0('#',dense_rank(desc(Importance))))

ggplot(rankImportance, aes(x = reorder(Variables, Importance),
                           y = Importance, fill = Importance)) +
  geom_bar(stat='identity') +
  geom_text(aes(x = Variables, y = 0.5, label = Rank),
            hjust=0, vjust=0.55, size = 4, colour = 'red') +
  labs(x = 'Variables') +
  coord_flip()
```



GLM

Generalized Linear Models (GLM) is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution. [6]

```
glm.model <- glm(Establishment_success ~ ., data=train, family='binomial')
glm.pred <- predict(glm.model, test, type="response")
glm.tbl <- table(test$Establishment_success, glm.pred > 0.5)
colnames(glm.tbl) <- c('TRUE', 'FALSE')
rownames(glm.tbl) <- c('TRUE', 'FALSE')
glm.cm <- confusionMatrix(glm.tbl)

glm.roc <-roc.curve( test[,dvar], glm.pred)
rect(0, 1.1, 1, 1.7, xpd=TRUE, col="white", border="white")
title("GLM ROC Curve")
```



```
print(glm.roc$auc)
```

```
## [1] 0.676918
```

```
cm.stats <- post.cm.stats('glm', glm.cm, glm.roc$auc)
```

XGBoost

Extreme Gradient Boosting (xgboost) includes efficient linear model solver and tree learning algorithms. It supports various objective functions, including regression, classification and ranking. [7]

A best fit model is determined by computing a minimum log loss based on an iterative process.[8]

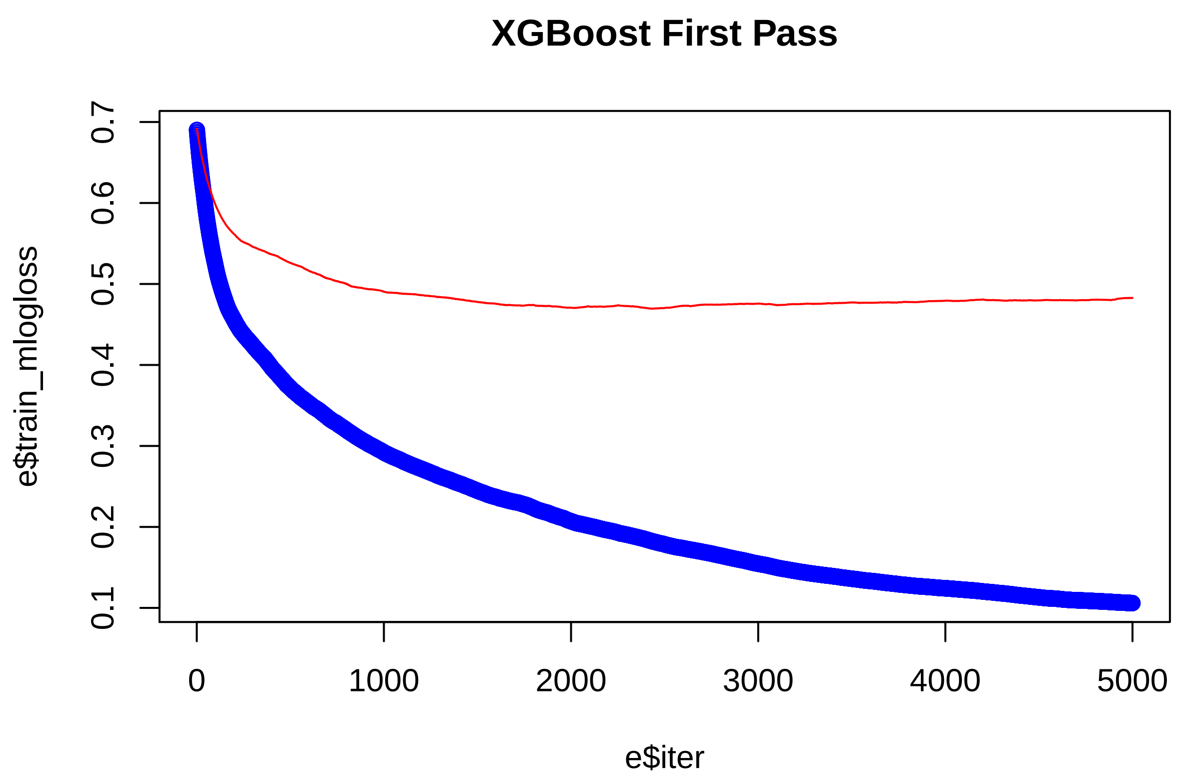
Most important feaures were derived based on the best fit.

```
train$Establishment_success <- as.numeric(train$Establishment_success)-1
test$Establishment_success <- as.numeric(test$Establishment_success)-1

trainm <- sparse.model.matrix(Establishment_success~., data = train)
train.label <- train[,"Establishment_success"]
train.matrix <- xgb.DMatrix(data = as.matrix(trainm), label = train.label)

testm <- sparse.model.matrix(Establishment_success~., data = test)
test.label <- test[,"Establishment_success"]
test.matrix <- xgb.DMatrix(data = as.matrix(testm), label = test.label)

nc <- length(unique(train.label))
xgb_params <- list("objective" = "multi:softprob",
```

```r
                   "eval_metric" = "mlogloss",
                   "num_class" = nc)

watchlist <- list(train = train.matrix, test = test.matrix)

best.xgboost.model <- function(nbr_rounds, params) {
     best <- xgb.train(params = params,
                       data = train.matrix,
                       nrounds = nbr_rounds,
                       watchlist = watchlist,
                       eta = 0.01,
                       max.depth = 3,
                       gamma = 0,
                       subsample = 1,
                       colsample_bytree = 1,
                       missing = NA,
                       seed = 333,
                       verbose = 0)
# Training & test error plot
    e <- data.frame(best$evaluation_log)
    plot(e$iter, e$train_mlogloss, col = 'blue')
    lines(e$iter, e$test_mlogloss, col = 'red')

    print(e[e$test_mlogloss == min(e$test_mlogloss),])
    return(list(best = best, iter = e[e$test_mlogloss == min(e$test_mlogloss),]))
}

best.iter <- 5000
res <- best.xgboost.model(best.iter, xgb_params)
```

```
##      iter train_mlogloss test_mlogloss
## 2436 2436        0.18208      0.469486
```

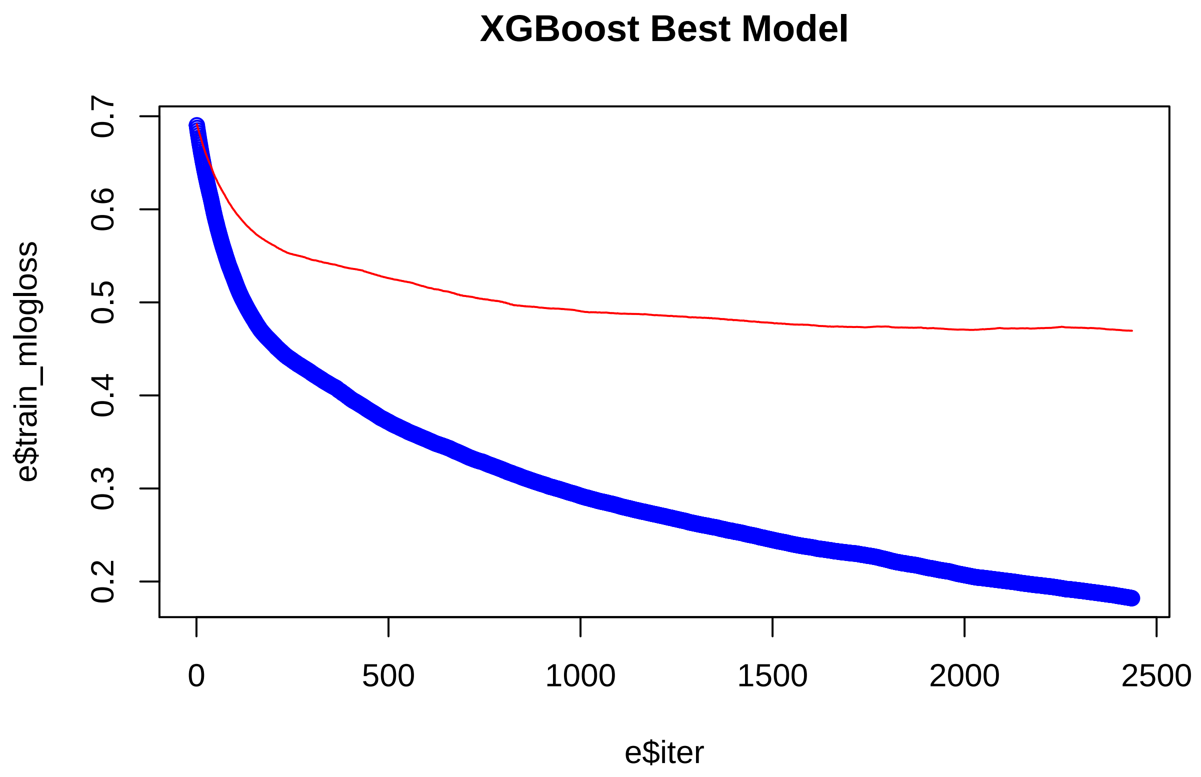
```r
rect(0, 1.1, 1, 1.7, xpd=TRUE, col="white", border="white")
title("XGBoost First Pass")
```



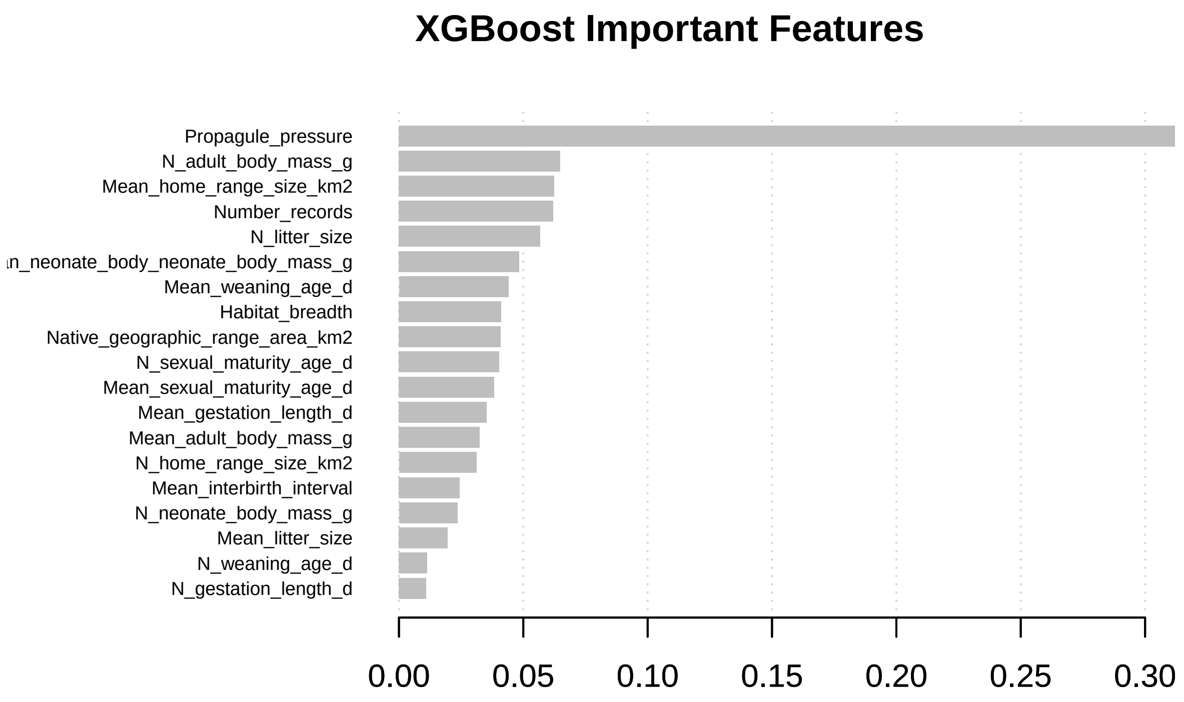
```r
if (best.iter != res$iter[[1]]) {
    res <- best.xgboost.model(res$iter[[1]], xgb_params)
}
```

```
##      iter train_mlogloss test_mlogloss
## 2436 2436        0.18208      0.469486
```

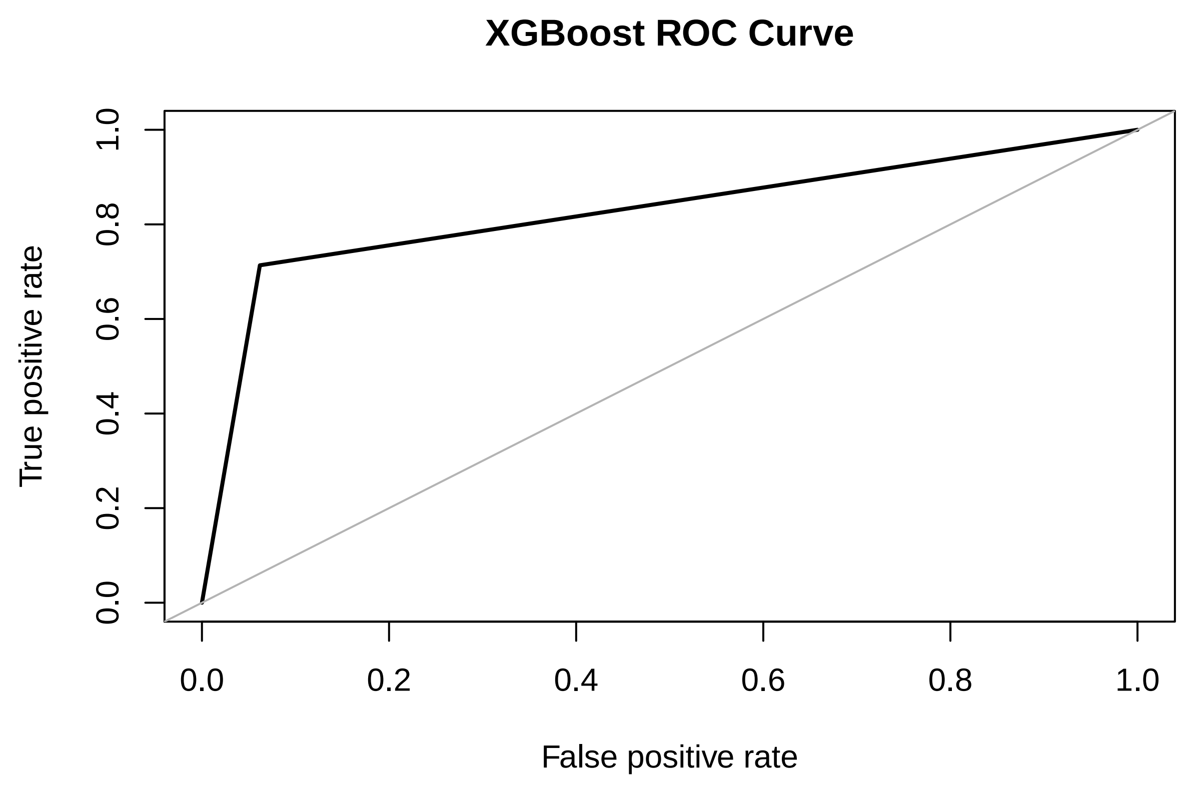
```r
rect(0, 1.1, 1, 1.7, xpd=TRUE, col="white", border="white")
title("XGBoost Best Model")
```



```
best.model <- res$best

imp <- xgb.importance(colnames(train.matrix), model = best.model)
xgb.plot.importance(imp)
rect(0, 1.1, 1, 1.7, xpd=TRUE, col="white", border="white")
title("XGBoost Important Features")
```



```
# Prediction & confusion matrix - test data
xg01.pred <- predict(best.model, newdata = test.matrix)
pred <- matrix(xg01.pred, nrow = nc, ncol = length(xg01.pred)/nc) %>%
  t() %>%
  data.frame() %>%
  mutate(label = test.label, max_prob = max.col(., "last")-1)

xg01.roc <-roc.curve( pred$max_prob, pred$label)
rect(0, 1.1, 1, 1.7, xpd=TRUE, col="white", border="white")
title("XGBoost ROC Curve")
```



```r
print(xg01.roc$auc)
```

```
## [1] 0.8257681
```

```r
xgboost.cm <- confusionMatrix(factor(pred$max_prob), factor(pred$label))

cm.stats <- post.cm.stats('xgboost', xgboost.cm, xg01.roc$auc)
```

Compare Models

Results from random forest and xgboost were very close, although xgboost did report a better Accuracy and larger AUC value.

```r
cm.stats
```

```
##         Method  Accuracy Sensitivity Specificity BalAccuracy       AUC
##   randomForest 0.7869416   0.6942675   0.8955224   0.6942675 0.7948950
## 1          glm 0.6357388   0.6946565   0.5875000   0.6946565 0.6769180
## 2      xgboost 0.8006873   0.6751592   0.9477612   0.6751592 0.8257681
```

Identify Important Features

In addition to the results from Random Forest and XGBoost, xgboost can be iteratively run against the data, identifying those features which make the most contribution for most successful and unseccessful introductions. [9]

```r
observation_level_variable_importance = function(train_data, live_data,
                                                 outcome_name,
                                                 eta = 0.2,
                                                 max_depth=4, max_rounds=3000,
                                                 number_of_factors=2) {
  set.seed(1234)
  split <- sample(nrow(train_data), floor(0.9 * nrow(train_data)))
  train_data_tmp <- train_data[split,]
  val_data_tmp <- train_data[-split,]

  feature_names <- setdiff(names(train_data_tmp), outcome_name)
  dtrain <- xgb.DMatrix(data.matrix(train_data_tmp[,feature_names]),
                        label=train_data_tmp[,outcome_name], missing=NaN)
  dval <- xgb.DMatrix(data.matrix(val_data_tmp[,feature_names]),
                      label=val_data_tmp[,outcome_name], missing=NaN)
  watchlist <- list(eval = dval, train = dtrain)
  param <- list(  objective = "binary:logistic",
                  eta = eta,
                  max_depth = max_depth,
                  subsample= 0.9,
                  colsample_bytree= 0.9
  )

  xgb_model <- xgb.train ( params = param,
                           data = dtrain,
                           eval_metric = "auc",
                           nrounds = max_rounds,
                           missing=NaN,
                           verbose = 0,
                           print_every_n = 10,
                           early_stop_round = 20,
                           watchlist = watchlist,
                           maximize = TRUE)

  original_predictions <- predict(xgb_model,
                                  data.matrix(live_data[,feature_names]),
                                  outputmargin=FALSE, missing=NaN)

  # strongest factors
  new_preds <- c()
  for (feature in feature_names) {
    live_data_trsf <- live_data
    # neutralize feature to population mean
    if (sum(is.na(train_data[,feature])) > (nrow(train_data) / 2)) {
      live_data_trsf[,feature] <- NA
    } else {
      live_data_trsf[,feature] <- mean(train_data[,feature], na.rm = TRUE)
    }
    predictions <- predict(object=xgb_model, data.matrix(live_data_trsf[,feature_names]),
                           outputmargin=FALSE, missing=NaN)
    new_preds <- cbind(new_preds, original_predictions - predictions)
  }

  positive_features <- c()
  negative_features <- c()
```

```
  feature_effect_df <- data.frame(new_preds)
  names(feature_effect_df) <- c(feature_names)

  for (pred_id in seq(nrow(feature_effect_df))) {
    vector_vals <- feature_effect_df[pred_id,]
    vector_vals <- vector_vals[,!is.na(vector_vals)]
    positive_features <- rbind(positive_features,
                               c(colnames(vector_vals)[order(vector_vals,
                                                            decreasing=TRUE)][1:number_of_factors]))
    negative_features <- rbind(negative_features,
                               c(colnames(vector_vals)[order(vector_vals,
  }

  positive_features <- data.frame(positive_features)
  names(positive_features) <- paste0('Pos_', names(positive_features))
  negative_features <- data.frame(negative_features)
  names(negative_features) <- paste0('Neg_', names(negative_features))

  return(data.frame(original_predictions, positive_features, negative_features))
}

xg.train <- train
xg.test <- test

xg.train$Establishment_success <-ifelse(xg.train$Establishment_success=='1', 1,0)
xg.test$Establishment_success <-ifelse(xg.test$Establishment_success=='1', 1,0)

outcome_name <- 'Establishment_success'

preds <- observation_level_variable_importance(train_data = xg.train,
                                               live_data = xg.test,
                                               outcome_name = outcome_name,
                                               number_of_factors=2)
preds <- preds[order(-preds$original_predictions),]
```

Survival Predictive Strength

Based upon the previous step, the following features show up as having the most positive impact among successful survival:

- Mean_neonate_body_neonate_body_mass_g
- Mean_interbirth_interval
- Propagule_pressure
- Mean_weaning_age_d

The most negative impact among successful survival:

- Mean_adult_body_mass_g
- N_neonate_body_mass_g
- N_home_range_size_km2
- N_litter_size

```
kable(preds[1:5,2:3], caption = "Survivors - Most Positive Impact")
```

Table 4: Survivors - Most Positive Impact

	Pos_X1	Pos_X2
226	Mean_neonate_body_neonate_body_mass_g	Mean_interbirth_interval
251	Mean_neonate_body_neonate_body_mass_g	Mean_interbirth_interval
142	Mean_neonate_body_neonate_body_mass_g	Propagule_pressure
177	Propagule_pressure	Mean_interbirth_interval
188	Propagule_pressure	Mean_weaning_age_d

Table 5: Survivors - Most Negative Impact

	Neg_X1	Neg_X2
226	Mean_adult_body_mass_g	N_neonate_body_mass_g
251	Mean_adult_body_mass_g	N_neonate_body_mass_g
142	N_home_range_size_km2	N_litter_size
177	Mean_sexual_maturity_age_d	Mean_litter_size
188	N_neonate_body_mass_g	N_adult_body_mass_g

```r
kable(preds[1:5,4:5], caption = "Survivors - Most Negative Impact")
```

NonSurvival Predictive Strength

The following features show up as having the most positive impact among unsuccessful survival:

- N_home_range_size_km2
- Mean_adult_body_mass_g
- Habitat_breadth
- Mean_interbirth_interval

The most negative impact among unsuccessful survival:

- Propagule_pressure
- N_adult_body_mass_g
- N_litter_size
- Propagule_pressure

```r
nbrr <- nrow(preds)
kable(preds[(nbrr-5):nbrr,2:3], caption = "NonSurvivors - Most Positive Impact")
```

```r
kable(preds[(nbrr-5):nbrr,4:5], caption = "NonSurvivors - Most Negative Impact")
```

Table 6: NonSurvivors - Most Positive Impact

	Pos_X1	Pos_X2
123	Habitat_breadth	N_weaning_age_d
3	N_home_range_size_km2	Mean_adult_body_mass_g
127	Habitat_breadth	Mean_interbirth_interval
2	Mean_adult_body_mass_g	Mean_interbirth_interval
124	Habitat_breadth	N_weaning_age_d
129	Habitat_breadth	Mean_neonate_body_neonate_body_mass_g

Table 7: NonSurvivors - Most Negative Impact

	Neg_X1	Neg_X2
123	N_litter_size	N_adult_body_mass_g
3	Propagule_pressure	N_adult_body_mass_g
127	N_litter_size	Propagule_pressure
2	Propagule_pressure	N_adult_body_mass_g
124	N_litter_size	N_adult_body_mass_g
129	N_litter_size	Propagule_pressure

```r
i <- sapply(preds, is.factor)
preds[i] <- lapply(preds[i], as.character)
```

Reports

This section contains reports and charts that visualize the most important features found in this study.

First, the categorical data is restored to the dataset and used in the reporting.

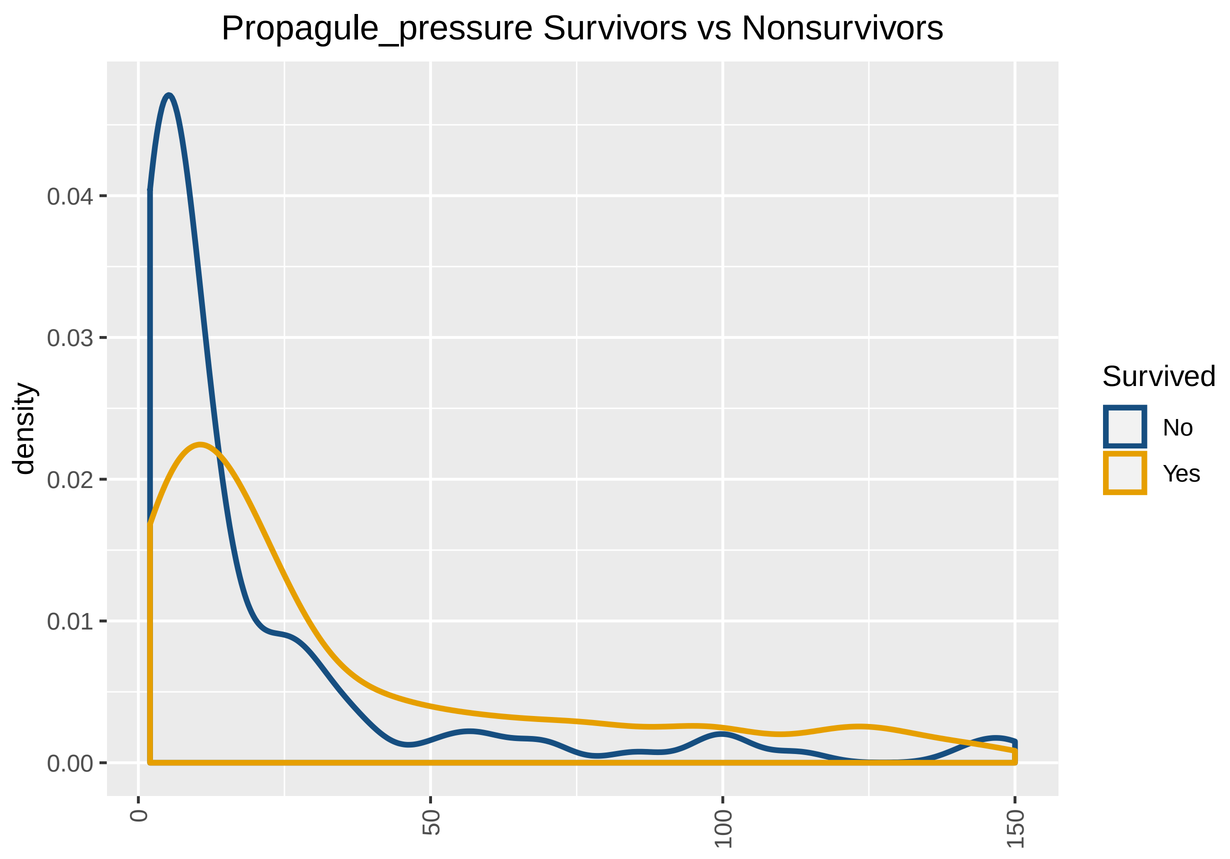
Propagule_pressure ranks the most important feature by all measures. It is defined as:

'Number of introduced individuals in the event as described by Long 2003. For multiple releases of the same species to the same location within time intervals <20 years, the total number of individuals released was pooled and considered as one single release.' [10]

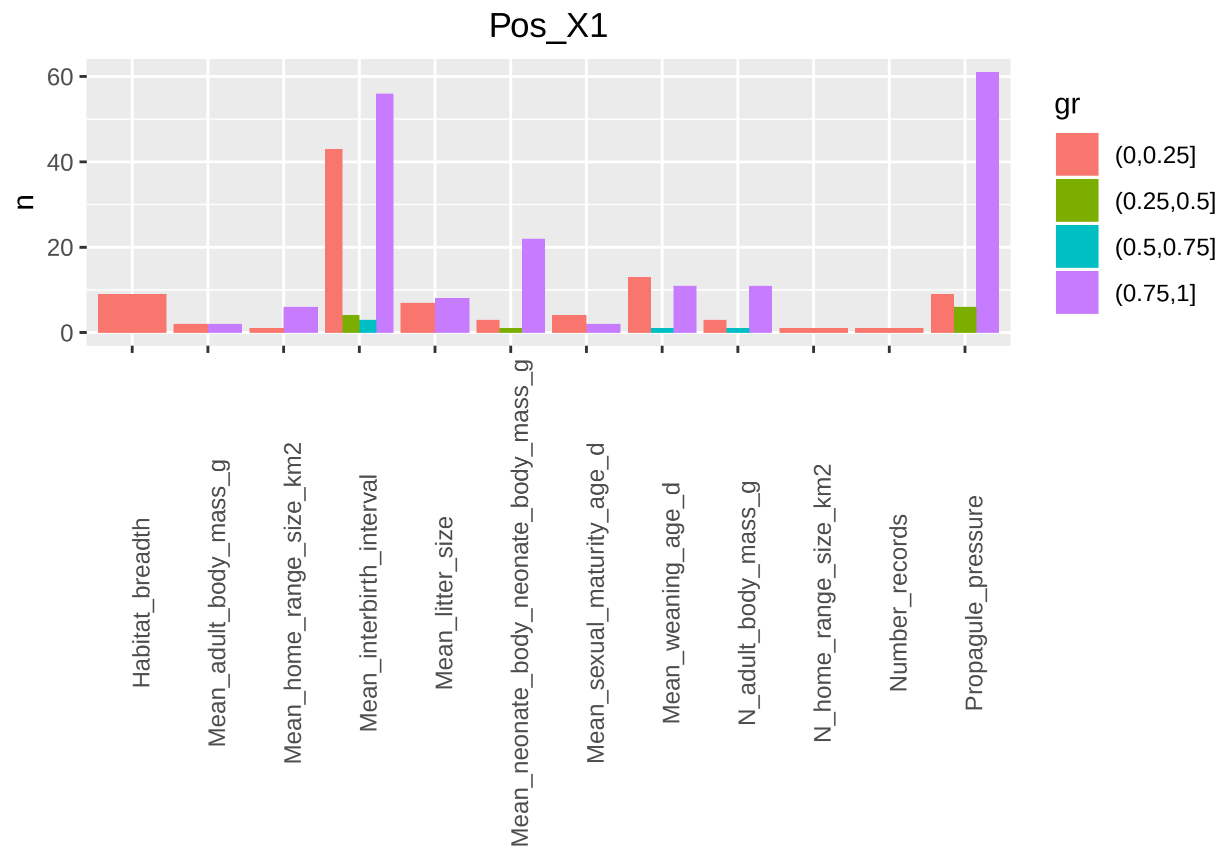
This section contains the following Reports and charts:

- Density plot - Survived vs Non-Survived
- Important Features as per the xgbost iterative approach [9]
- Top Survivors by Propagule Pressure
- Top Survivors by Mean Neonate Body Mass
- Top Survivors by Mean Interbirth Interval
- Top Survivors by Mean Home Range Size
- Introductions by Year
- Propagule Pressure by Year
- Top Survivors
- 100% Non Survivors
- Survivors - Propagule Pressure By Introductions
- NonSurvivors - Propagule Pressure By Introductions

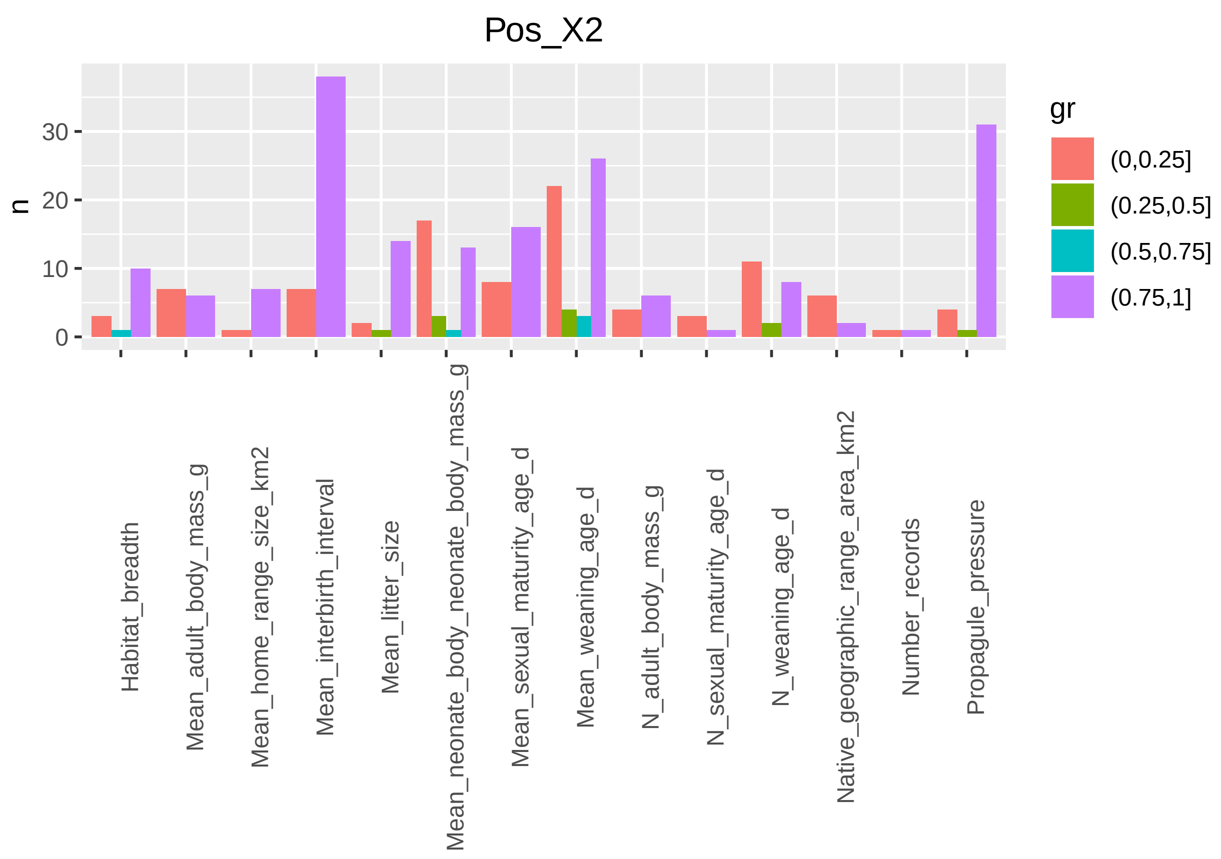
```r
## restore categorical data
df.cmpl <- mice::complete(df.mice ,1)
df.cmpl[, dvar] <- factor(df.orig[, df.orig.dvar])

for(i in (df.orig.dvar +1) : length(df.orig)) {
  df.cmpl[, (length(df.cmpl) + 1)] <- factor(df.orig[, i])
  names(df.cmpl)[length(df.cmpl)] <- names(df.orig)[i]
}

# density plot
ggplot(df.cmpl, aes(x=Propagule_pressure,  color=Establishment_success)) +
  geom_density(size=1.0) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        axis.title.x=element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  labs(color = 'Survived') +
  scale_color_manual(labels = c("No", "Yes"), values = c("#164E80", "#E69F00")) +
  ggtitle('Propagule_pressure Survivors vs Nonsurvivors')
```



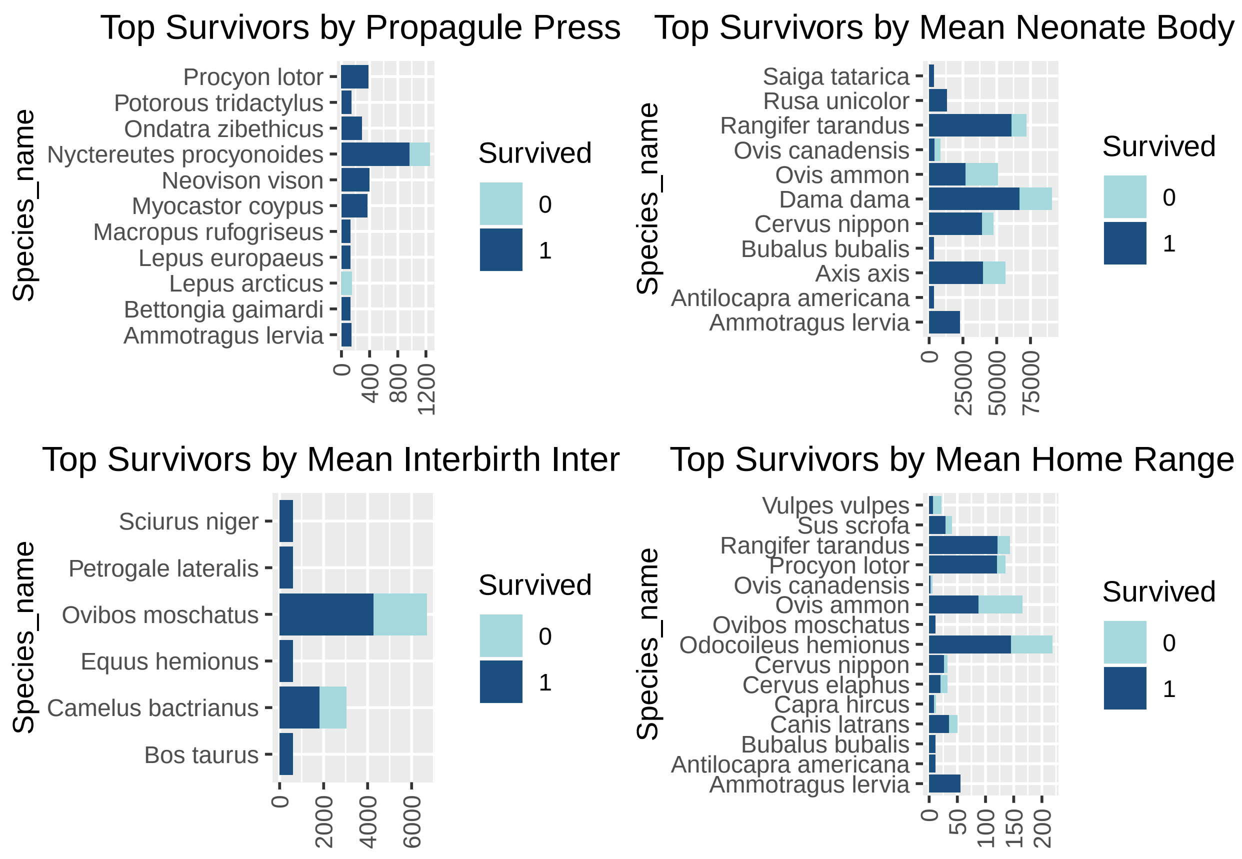
```
pos_x1_cnt <- preds %>%
  group_by(gr=cut(original_predictions, breaks= seq(0, 1, by = 0.25)) ) %>%
  dplyr::count(Pos_X1) %>%
  arrange(desc(gr),-n)

pos_x2_cnt <- preds %>%
  group_by(gr=cut(original_predictions, breaks= seq(0, 1, by = 0.25)) ) %>%
  dplyr::count(Pos_X2) %>%
  arrange(desc(gr),-n)

par(mfrow=c(1, 1))

ggplot(pos_x1_cnt,aes(Pos_X1,n,fill=gr)) +
  geom_bar(stat="identity",position='dodge') +
  theme(axis.text.x = element_text(angle = 90),
        axis.title.x=element_blank(),
        plot.title = element_text(hjust = 0.5))  +
  ggtitle('Pos_X1')
```



```r
ggplot(pos_x2_cnt,aes(Pos_X2,n,fill=gr)) +
  geom_bar(stat="identity",position='dodge') +
  theme(axis.text.x = element_text(angle = 90),
        axis.title.x=element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  ggtitle('Pos_X2')
```

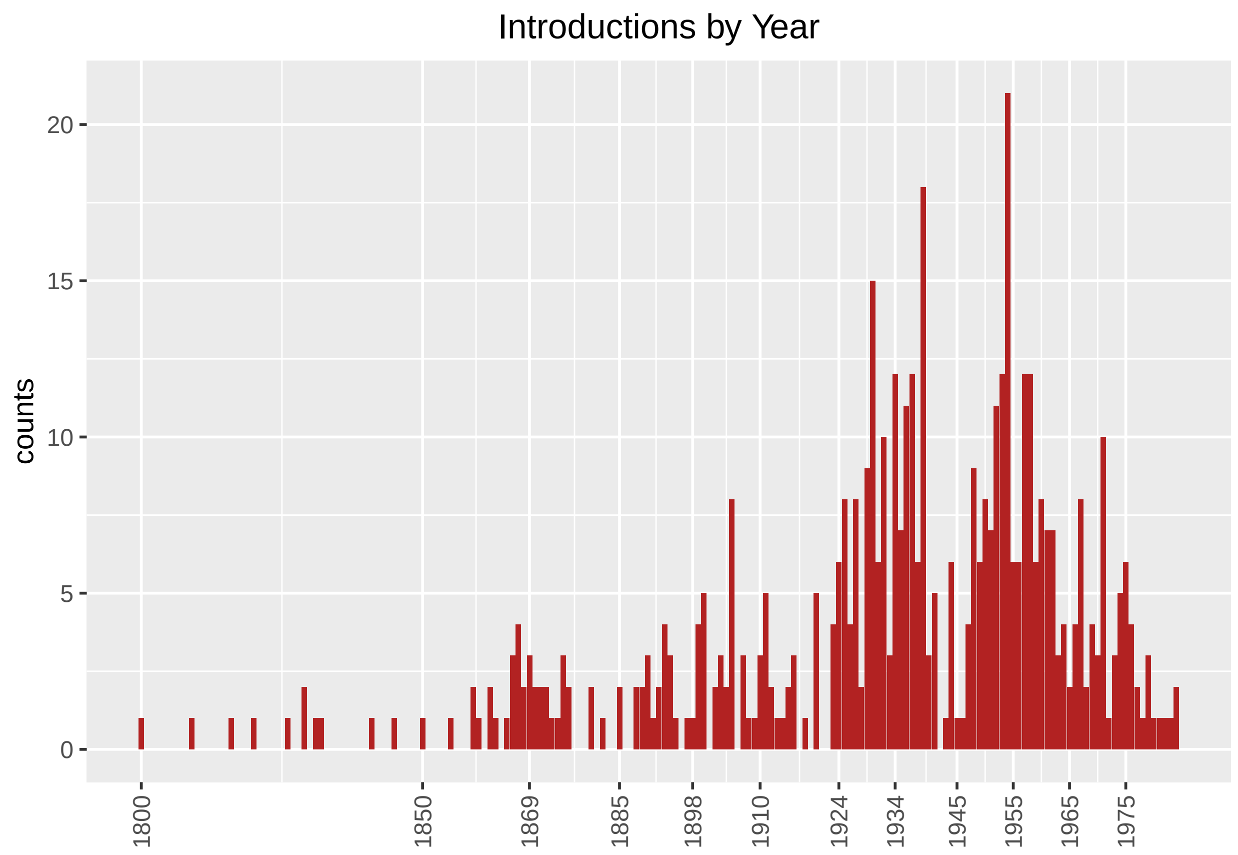


```
TSur01 <- ggplot(arrange(top_n(df.cmpl, 20, Propagule_pressure), -Propagule_pressure),
       aes( Species_name, Propagule_pressure, fill=Establishment_success)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        axis.title.x=element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c('#A5D8DD', '#1C4E80'), name = "Survived") +
  scale_color_manual(labels = c("No", "Yes"), values = c('#A5D8DD', '#1C4E80')) +
  geom_bar(stat="identity") +
  geom_text(aes(y =ave( Propagule_pressure, Species_name, FUN = mean), label='')) +
  coord_flip() +
  ggtitle('Top Survivors by Propagule Pressure')

TSur02 <- ggplot(arrange(top_n(df.cmpl, 80, Mean_neonate_body_neonate_body_mass_g), -Mean_neonate_body_n
       aes( Species_name, Mean_neonate_body_neonate_body_mass_g, fill=Establishment_success)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        axis.title.x=element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c('#A5D8DD', '#1C4E80'), name = "Survived") +
  scale_color_manual(labels = c("No", "Yes"), values = c('#A5D8DD', '#1C4E80')) +
  geom_bar(stat="identity") +
  geom_text(aes(y =ave( Mean_neonate_body_neonate_body_mass_g, Species_name, FUN = mean), label='')) +
  coord_flip() +
  ggtitle('Top Survivors by Mean Neonate Body Mass')

TSur03 <- ggplot(arrange(top_n(df.cmpl, 5, Mean_interbirth_interval), -Mean_interbirth_interval),
       aes( Species_name, Mean_interbirth_interval, fill=Establishment_success)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        axis.title.x=element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c('#A5D8DD', '#1C4E80'), name = "Survived") +
  scale_color_manual(labels = c("No", "Yes"), values = c('#A5D8DD', '#1C4E80')) +
  geom_bar(stat="identity") +
  geom_text(aes(y =ave( Mean_interbirth_interval, Species_name, FUN = mean), label='')) +
  coord_flip() +
  ggtitle('Top Survivors by Mean Interbirth Interval')

TSur04 <- ggplot(arrange(top_n(df.cmpl, 100, Mean_home_range_size_km2), -Mean_home_range_size_km2),
       aes( Species_name, Mean_home_range_size_km2, fill=Establishment_success)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        axis.title.x=element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c('#A5D8DD', '#1C4E80'), name = "Survived") +
  scale_color_manual(labels = c("No", "Yes"), values = c('#A5D8DD', '#1C4E80')) +
  geom_bar(stat="identity") +
  geom_text(aes(y =ave( Mean_home_range_size_km2, Species_name, FUN = mean), label='')) +
  coord_flip() +
  ggtitle('Top Survivors by Mean Home Range Size')

grid.arrange(TSur01, TSur02, TSur03, TSur04,ncol=2, nrow=2)
```

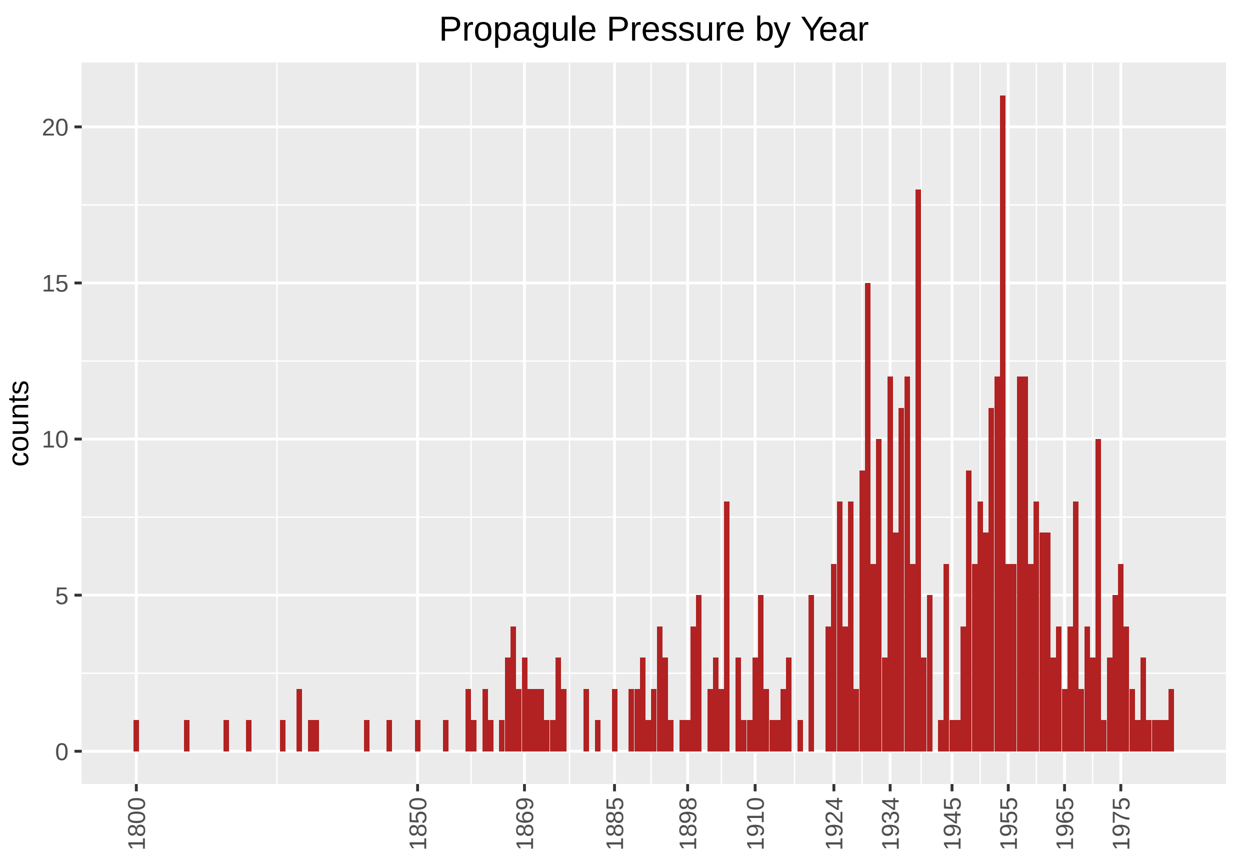


```
## split into successful/unsuccessful
df0 <- subset(df.cmpl, Establishment_success == 0)
```

```r
df1 <- subset(df.cmpl, Establishment_success == 1)

year.tbl <- df.cmpl[,c(1:dvar, 25)] %>%
  group_by(Year_introduction) %>%
  summarise(counts = n())

intro.date <- as.Date(lubridate::ymd(year.tbl$Year_introduction, truncated = 2L))

ggplot(year.tbl, aes(x=intro.date, y=counts)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        axis.title.x=element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  geom_bar(stat = "identity", fill="firebrick") +
  scale_x_date(breaks =intro.date[seq(1, length(intro.date), by = 10)],
               labels = date_format("%Y")) +
  ggtitle('Introductions by Year')
```



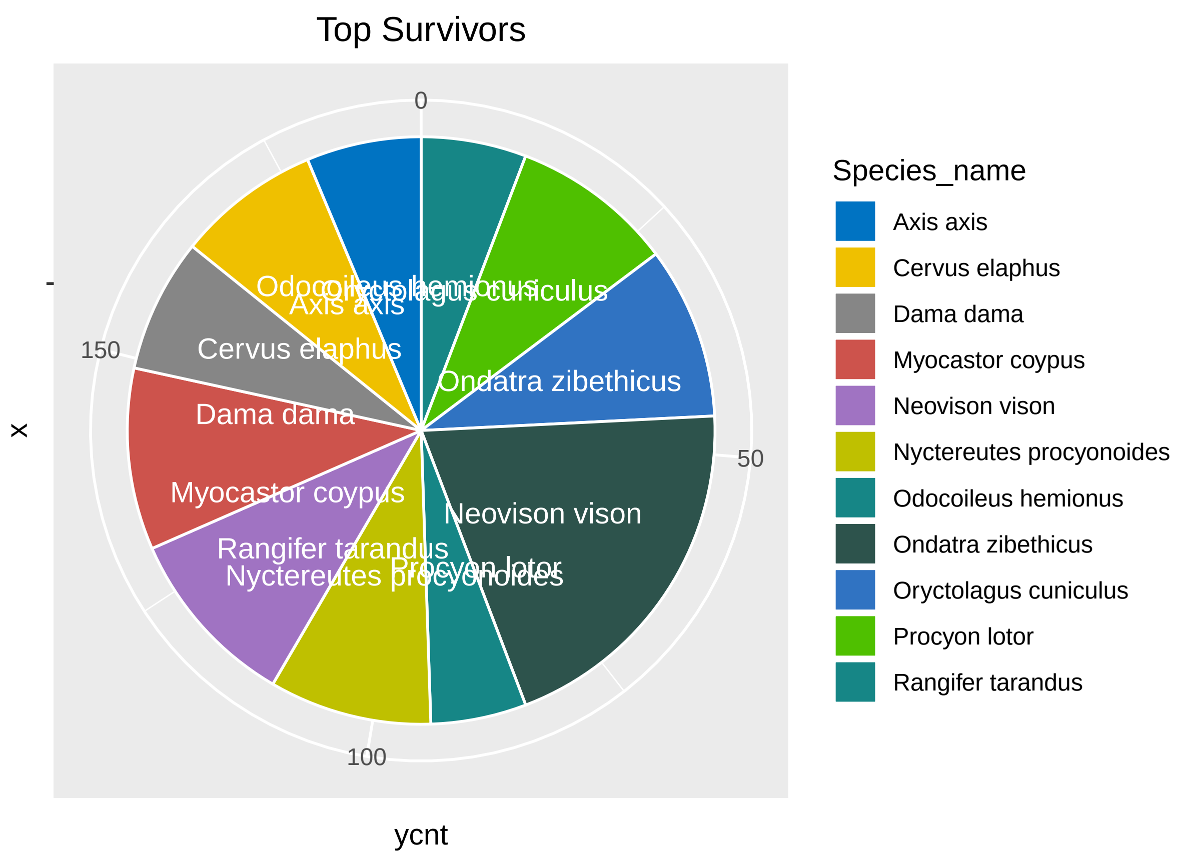
```r
ppress.tbl <- df.cmpl[,c(1:dvar, 25)] %>%
  group_by(Year_introduction) %>%
  summarise(counts = mean(Propagule_pressure))

ppress.date <- as.Date(lubridate::ymd(ppress.tbl$Year_introduction, truncated = 2L))

ggplot(year.tbl, aes(x=ppress.date, y=counts)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
        axis.title.x = element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  geom_bar(stat="identity", fill="firebrick") +
  scale_x_date(breaks = ppress.date[seq(1, length(ppress.date), by = 10)],
               labels = date_format("%Y")) +
  ggtitle('Propagule Pressure by Year')
```



```
## species comparison table
spec.no <- df.cmpl %>%
  filter(Establishment_success == 0) %>%
  group_by(Species_name  ) %>%
  summarise(ncnt = n(), nppress= mean(Propagule_pressure))

spec.yes <- df.cmpl %>%
  filter(Establishment_success == 1) %>%
  group_by(Species_name  ) %>%
  summarise(ycnt = n(), yppress= mean(Propagule_pressure))

spec.yn <- full_join(spec.yes,spec.no)
```

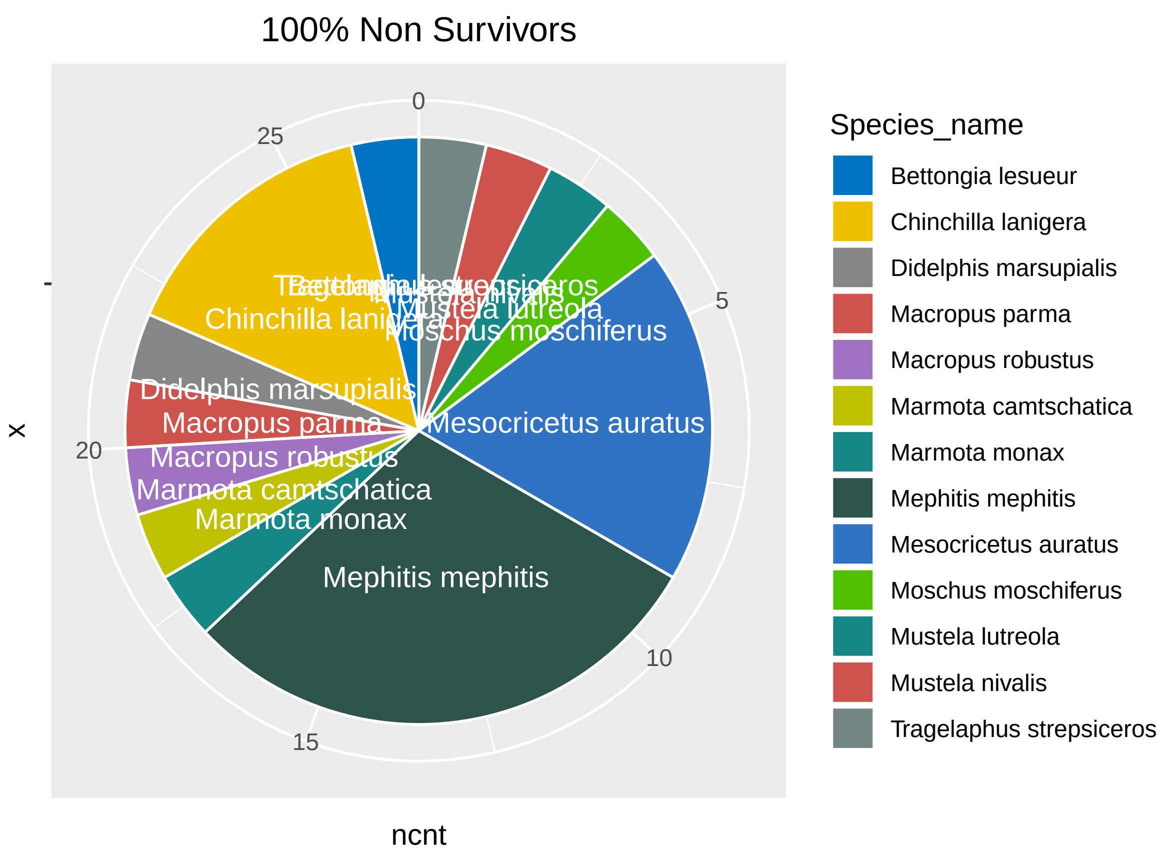
```
## Joining, by = "Species_name"
```

```
spec.yn[is.na(spec.yn)] <- 0
spec.yn <- spec.yn %>%
  mutate( pctSur = ycnt / (ycnt + ncnt))

spec.ynTop <- spec.yn %>%
  filter(ycnt >= 10) %>%
  mutate( pctSur = ycnt / (ycnt + ncnt))

spec.ynTop<- spec.ynTop %>%
  arrange(desc(pctSur)) %>%
  mutate(lab.ypos = cumsum(ycnt) - 0.5*ycnt)

mycols <- c("#0073C2FF", "#EFC000FF", "#868686FF", "#CD534CFF",
            "#A073C2FF", "#BFC000FF", "#168686FF", "#2D534CFF",
            "#3073C2FF", "#4FC000FF", "#168686FF", "#CD534CFF", "#758686FF")

ggplot(spec.ynTop, aes(x = "", y = ycnt, fill = Species_name)) +
  geom_bar(width = 1, stat = "identity", color = "white") +
  coord_polar("y", start = 0)+
  geom_text(aes(y = lab.ypos, label = Species_name), color = "white")+
  scale_fill_manual(values = mycols) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ggtitle('Top Survivors')
```

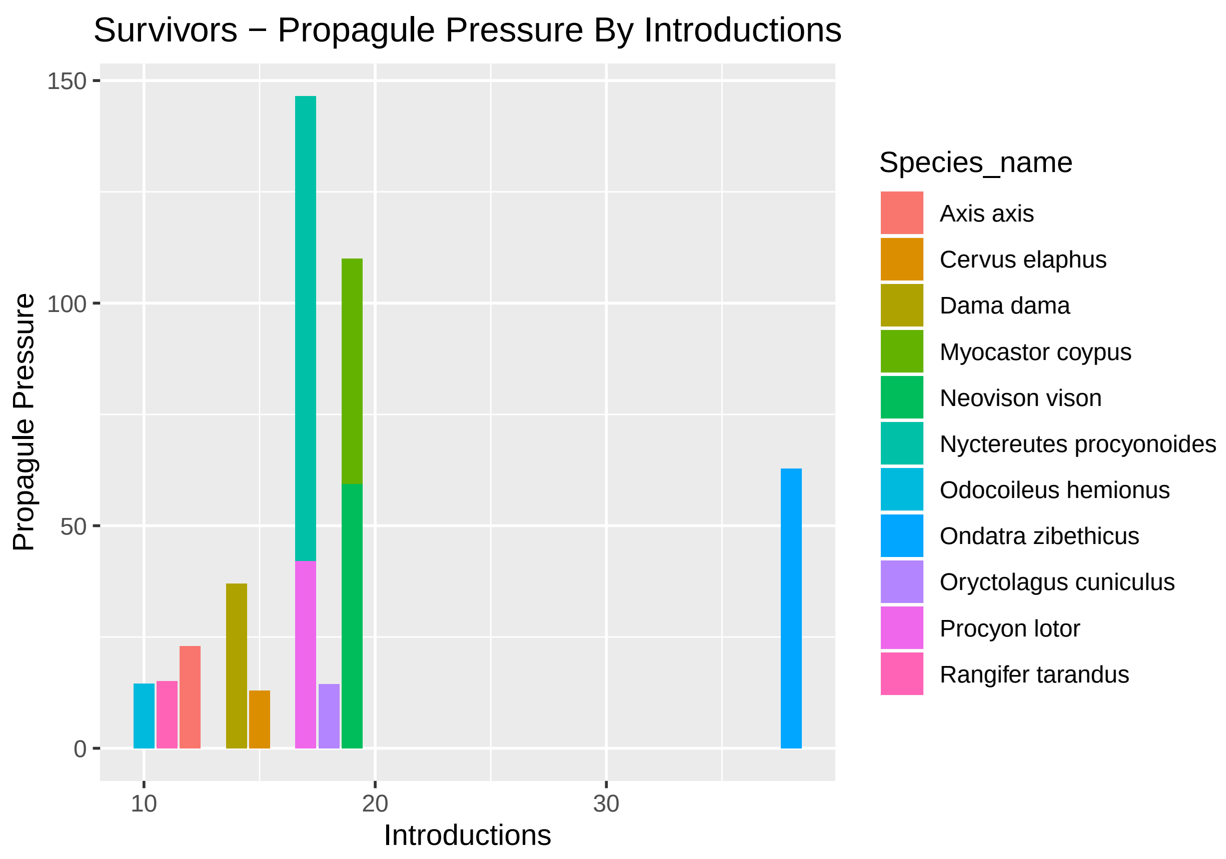


```
spec.ynBot <- spec.yn %>%
  filter(ycnt == 0) %>%
  mutate( pctSur = ycnt / (ycnt + ncnt))

spec.ynBot<- spec.ynBot %>%
  arrange(desc(Species_name)) %>%
  mutate(lab.ypos = cumsum(ncnt) - 0.5*ncnt)
```

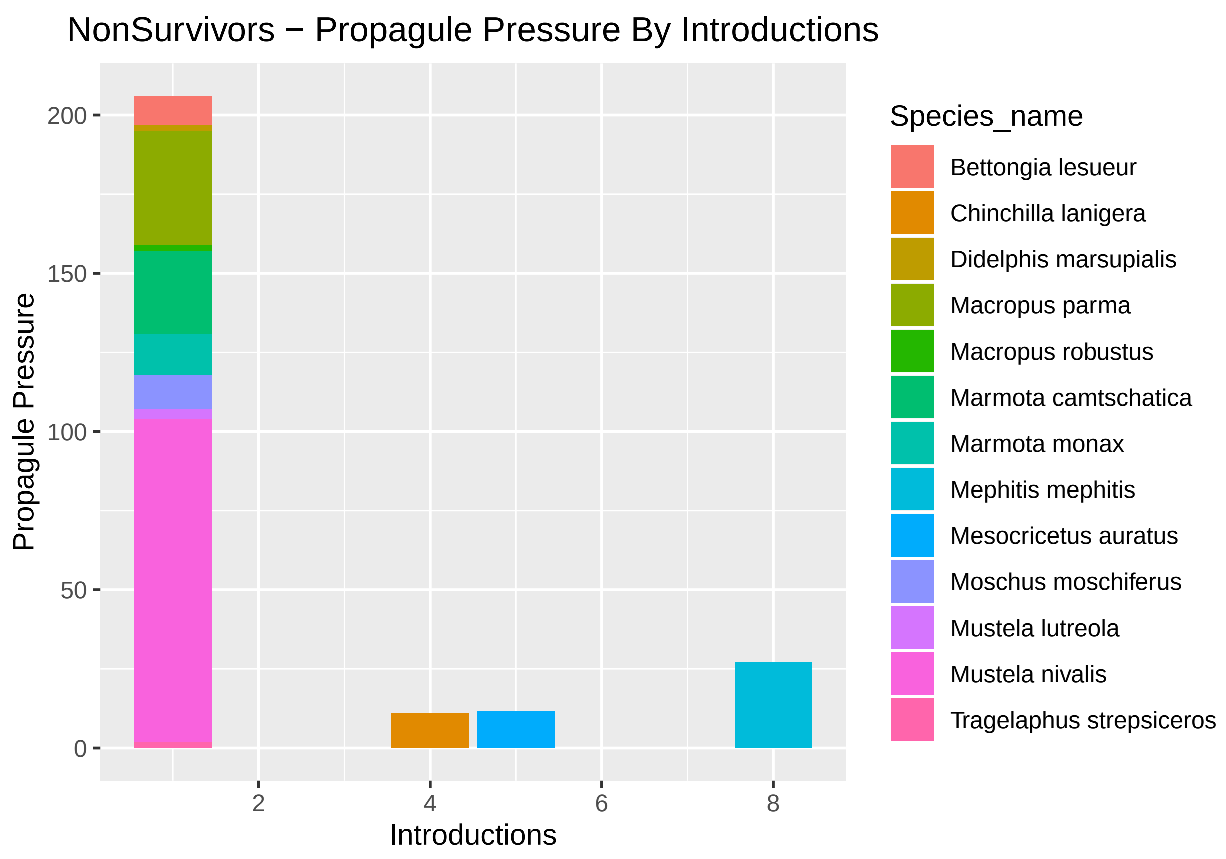
```r
ggplot(spec.ynBot, aes(x = "", y = ncnt, fill = Species_name)) +
  geom_bar(width = 1, stat = "identity", color = "white") +
  coord_polar("y", start = 0)+
  geom_text(aes(y = lab.ypos, label = Species_name), color = "white")+
  scale_fill_manual(values = mycols) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ggtitle('100% Non Survivors')
```



```r
ggplot(data=spec.ynTop, aes(x=ycnt, y=yppress, fill=Species_name)) +
  geom_bar(stat="identity")+
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x = "Introductions", y = "Propagule Pressure") +
  ggtitle('Survivors - Propagule Pressure By Introductions')
```



```r
ggplot(data=spec.ynBot, aes(x=ncnt, y=nppress, fill=Species_name)) +
  geom_bar(stat="identity")+
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x = "Introductions", y = "Propagule Pressure") +
  ggtitle('NonSurvivors - Propagule Pressure By Introductions')
```



## Sources

[1] The American Naturalist Vol 185 Number 6 June 2015 https://www.journals.uchicago.edu/doi/10.1086/681105

Manuela Gonzalez-Suarez,1,* Sven Bacher,2,and Jonathan M. Jeschke3, 1. Department of Conservation Biology, Estacion Biológica de Donana-Consejo Superior de Investigaciones Científicas, Calle Américo Vespucio s/n, 41092 Sevilla, Spain

2. Department of Biology, Unit Ecology and Evolution, University of Fribourg, Chemin du Musee 10, 1700 Fribourg, Switzerland

3. Department of Ecology and Ecosystem Management, Restoration Ecology, Technische Universitat München, 85350 Freising-Weihenstephan, Germany; Leibniz-Institute of Freshwater Ecology and Inland Fisheries (IGB), Muggelseedamm 310, 12587 Berlin, Germany; and Department of Biology, Chemistry, Pharmacy, Institute of Biology, Freie Universitat Berlin, Konigin-Luise-Strasse 1-3, 14195 Berlin, Germany

[2] Identify, describe, plot, and remove the outliers from the dataset Klodian Dhana https://www.r-bloggers.com/identify-describe-plot-and-remove-the-outliers-from-the-dataset/

[3] mice: Multivariate Imputation by Chained Equations Van Buuren and Groothuis-Oudshoorn (2011) https://cran.r-project.org/web/packages/mice/index.html

[4] ROSE: Random Over-Sampling Examples Menardi and Torelli, 2013 https://cran.r-project.org/web/packages/ROSE/index.html

[5] How to do feature selection in R using Random Forest for classification and regression? sauravkaushik8 https://discuss.analyticsvidhya.com/t/how-to-do-feature-selection-in-r-using-random-forest-for-classification-and-regressio
8420/9

[6] glm Fitting Generalized Linear Models https://www.rdocumentation.org/packages/stats/versions/3.6.1/topics/glm

[7] xgboost Extreme Gradient Boosting https://cran.r-project.org/web/packages/xgboost/xgboost.pdf

[8] eXtreme Gradient Boosting XGBoost Algorithm with R - Example in Easy Steps with One-Hot Encoding Dr. Bharatendra Rai https://www.youtube.com/watch?v=woVTNwRrFHE

[9] Actionable Insights: Getting Variable Importance at the Prediction Level in R Manuel Amunategui https://www.youtube.com/watch?v=gs74y1R-uEw

[10] Metadata file for González-Suárez et al. (2015) American Naturalist https://datadryad.org/stash/data_paper/doi:10.5061/dryad.sp963?