DSC630 Course Project

Moshe Burnstein

2023-05-21

Load in bank data

```
bank_df <- read.csv('bank-additional-full.csv', sep=';', header=TRUE, stringsAsFactors = TRUE)
```

Check balance of dataset in yes/no subscribing, the target.

```
library(vctrs)
subscriber_counts <- vec_count(bank_df$y)
subscriber_counts
```

```
##   key count
## 1  no 36548
## 2 yes  4640
```

```
percent_subscribed <- 4640/36548*100
percent_subscribed
```

```
## [1] 12.69563
```

The target variable is imbalanced. There are 36,548 who declined to subscribe, and 4,640 who subscribed. Only 12.7% subscribed.

Create new df for subscribers and a separate df for non-subscribers in order to compare distibutions of the different groups.

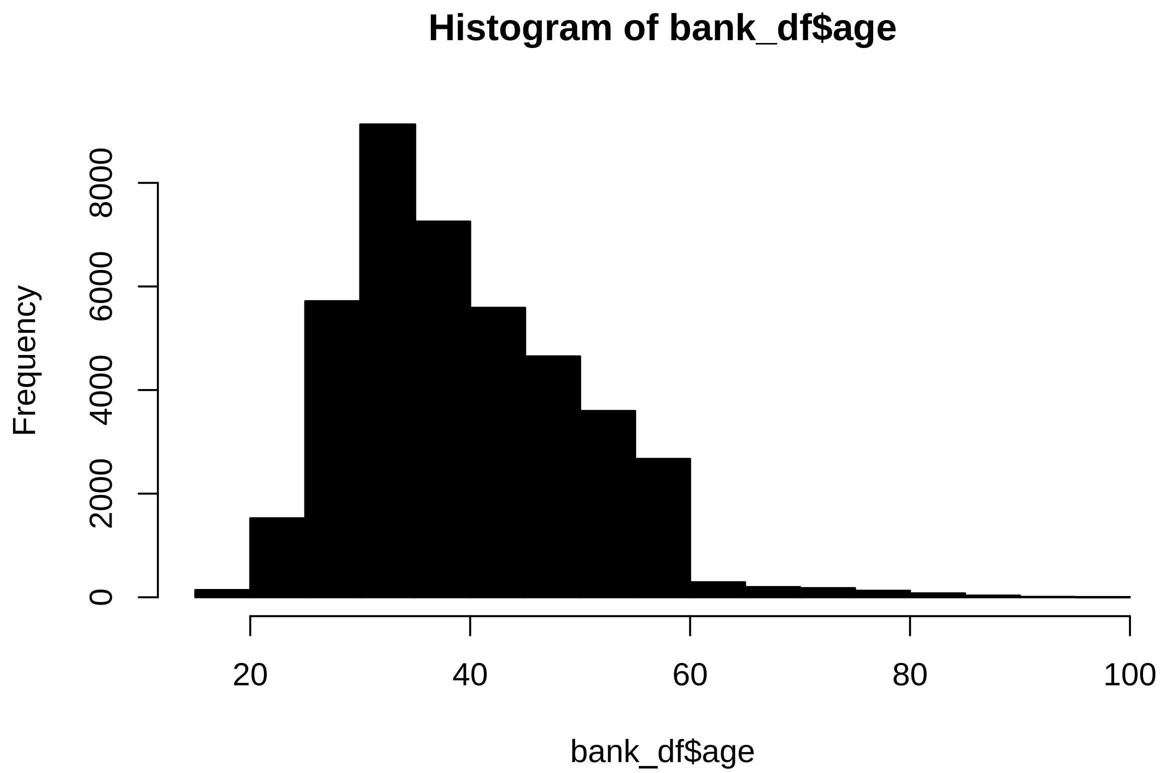
```
yes_subscribe_df <- bank_df[bank_df$y == 'yes',]
```

Create df of non-subscribers

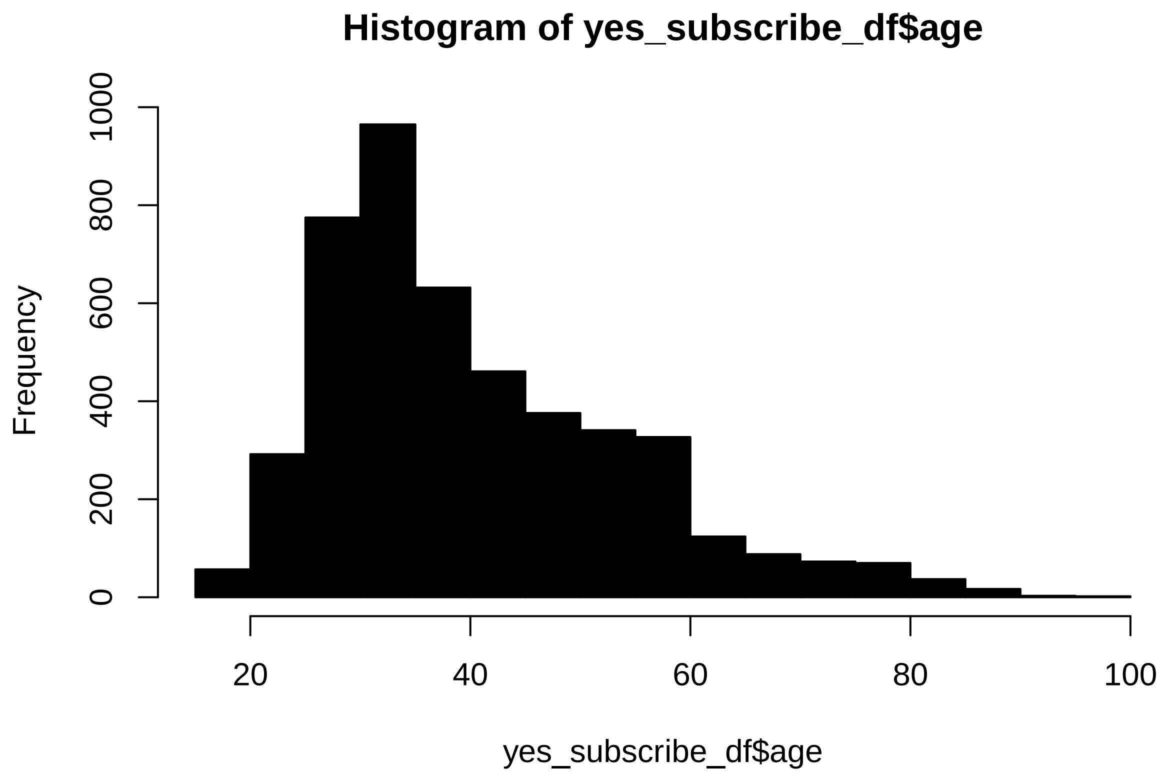
```
non_subscribe_df <- bank_df[bank_df$y != 'yes',]
```

Visualize the distributions of age in the entire df, in the subscribe df, and in the non-subscribe df.

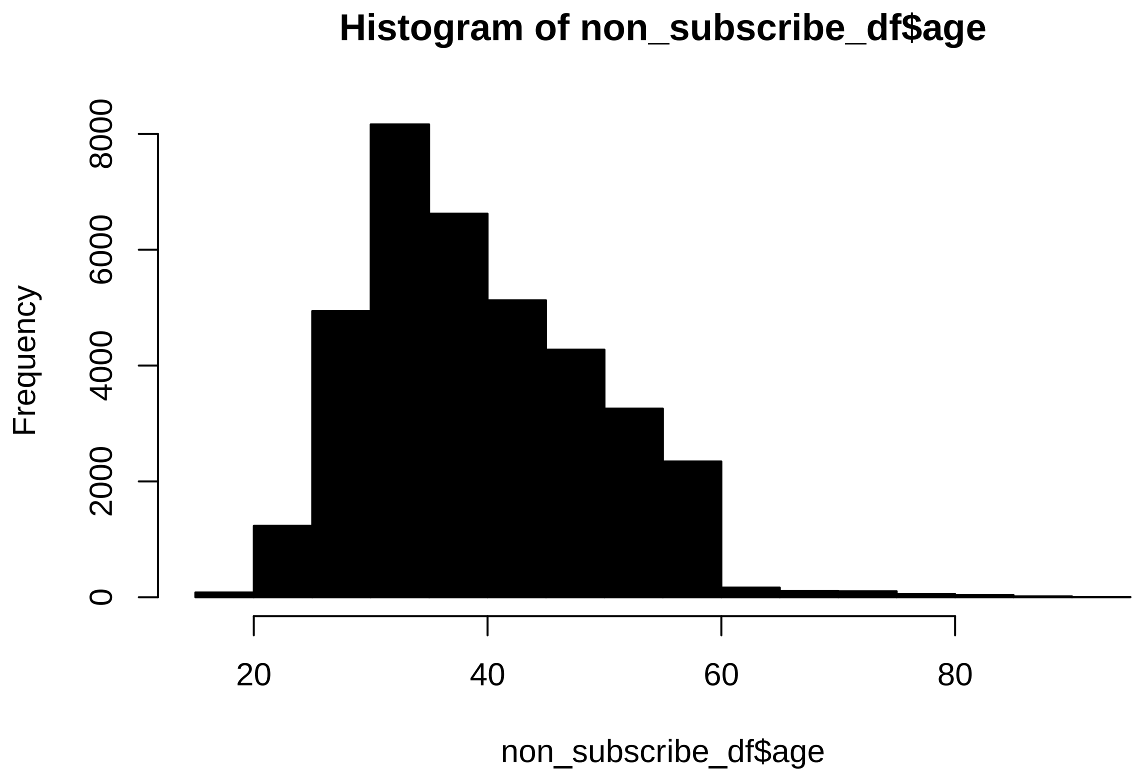
```
hist(bank_df$age, col = 1)
```




```
hist(yes_subscribe_df$age, col = 1)
```




```
hist(non_subscribe_df$age, col = 1)
```



One must take note that while the x-axis is on the same scale throughout, aside from no 100 year-old outliers in the non-subscribe df, the scale of the y-axis frequency is up to 1,000, and not 8,000. This is because there are only a fraction of the number of observations in the subscribe df as there are in the entire df, or in the non-subscribe df. The shape of the distribution differs from subscribers to non-subscribers. The vast majority of non-subscribers are between 20 and 60 years of age, while subscribers are well-represented both in the under 20 age bin and in the over 60 bins.

Check balance of subscribers in people younger or equal to 20 and older or equal to 60.

```
old_young_df <- bank_df[bank_df$age <= 20 | bank_df$age >= 60,]
library(vctrs)
old_young_subscribe_counts <- vec_count(old_young_df$y)
old_young_subscribe_counts
```

```
##   key count
## 1  no   804
## 2 yes   529
```

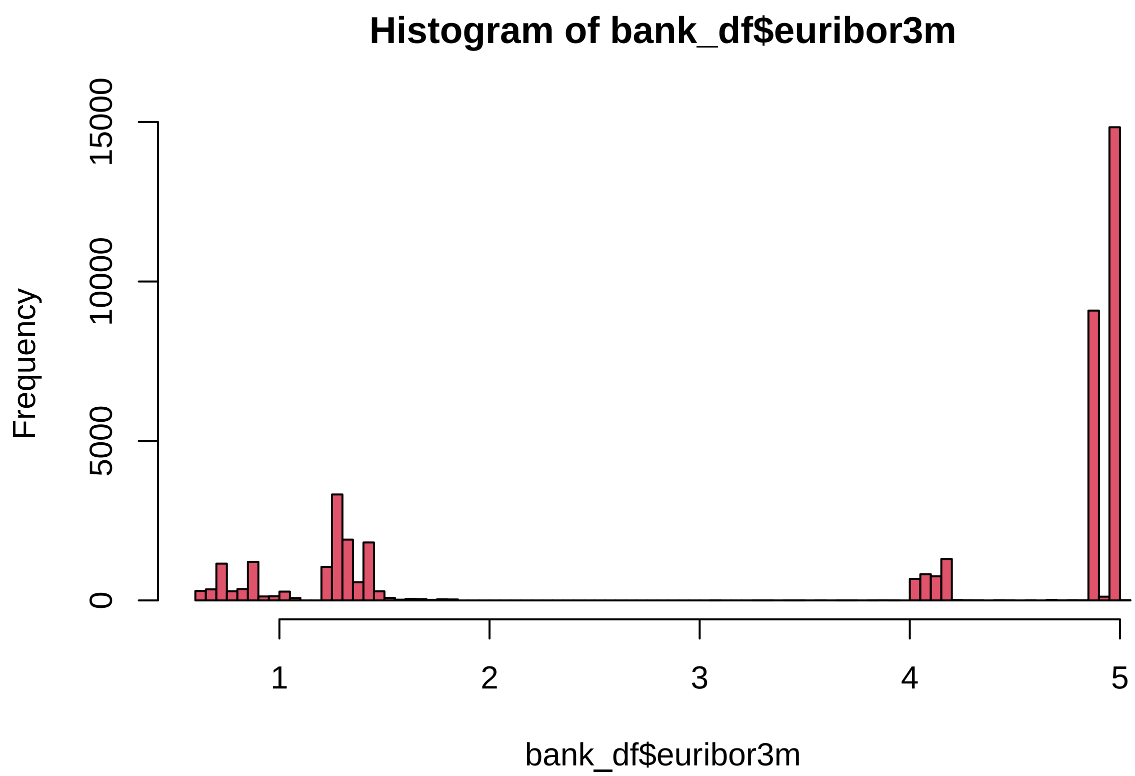
```
subscribe_percent <- 529/804*100
subscribe_percent
```

```
## [1] 65.79602
```

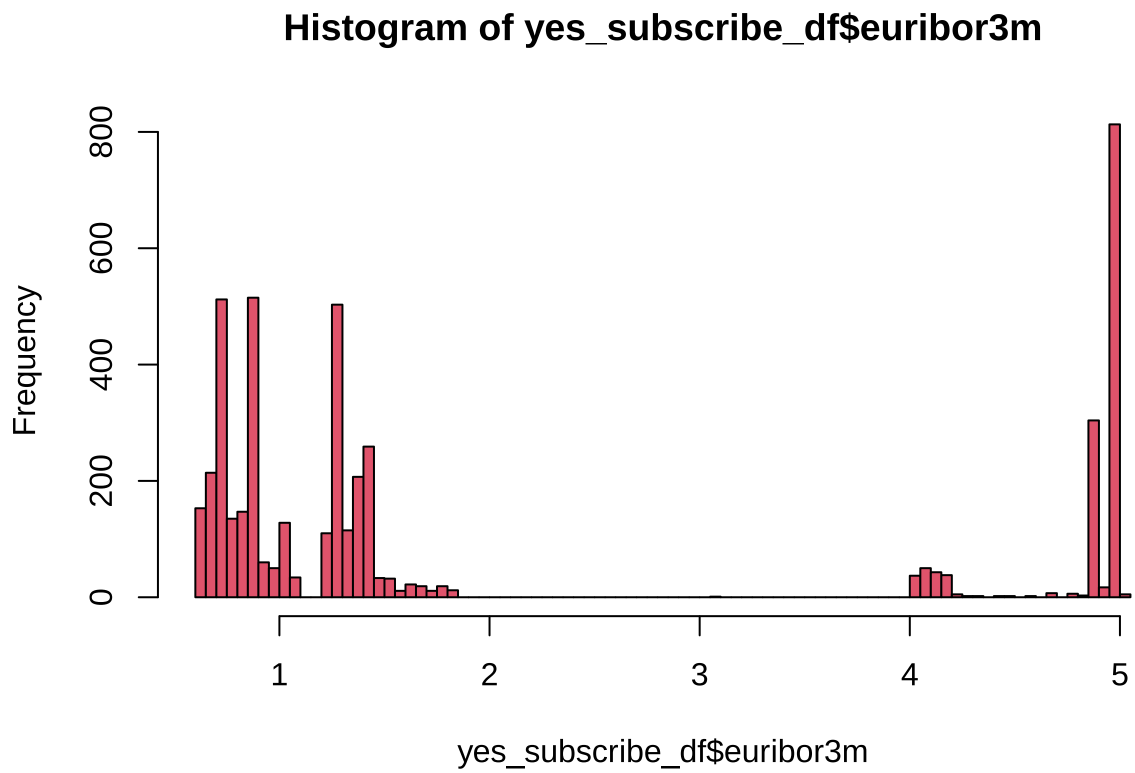
Yes-subscribers make up 65.8% of the population.

Plot euribor3m distributions for entire population, for yes-subscribers, and for non-subscribers.

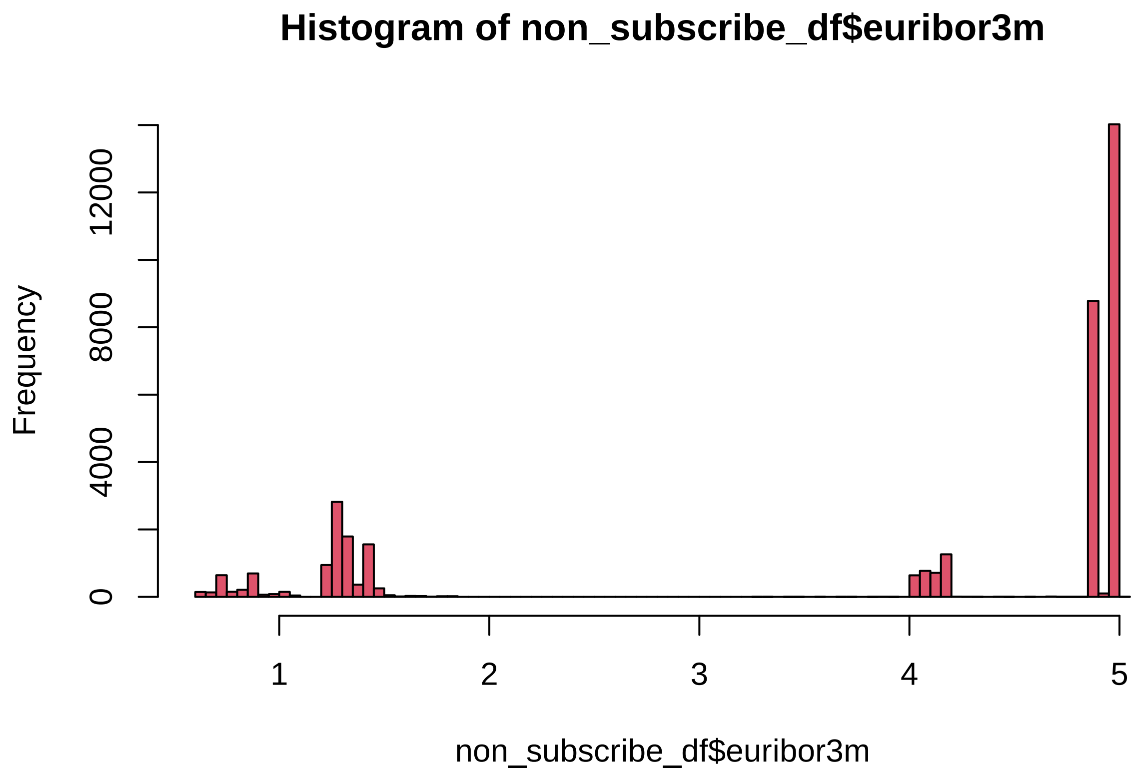
```r
hist(bank_df$euribor3m, breaks = 100, col = 2)
```




```r
hist(yes_subscribe_df$euribor3m, breaks = 100, col = 2)
```



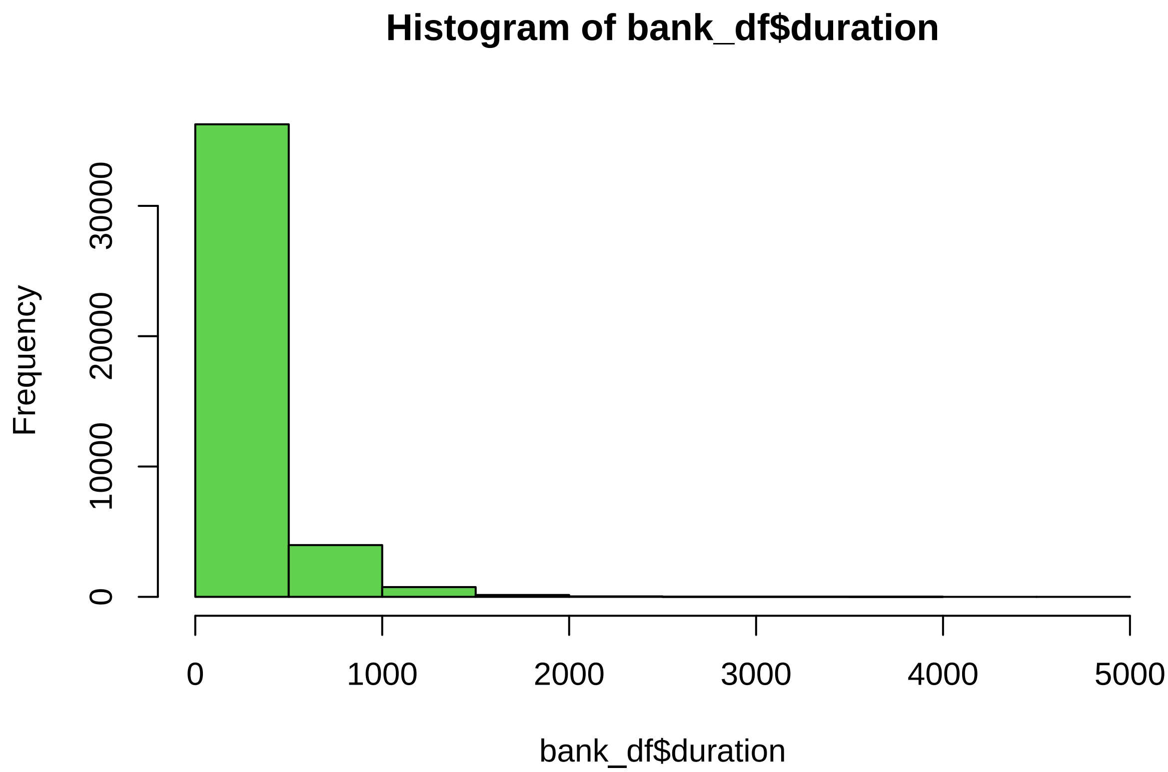

```
hist(non_subscribe_df$euribor3m, breaks = 100, col = 2)
```



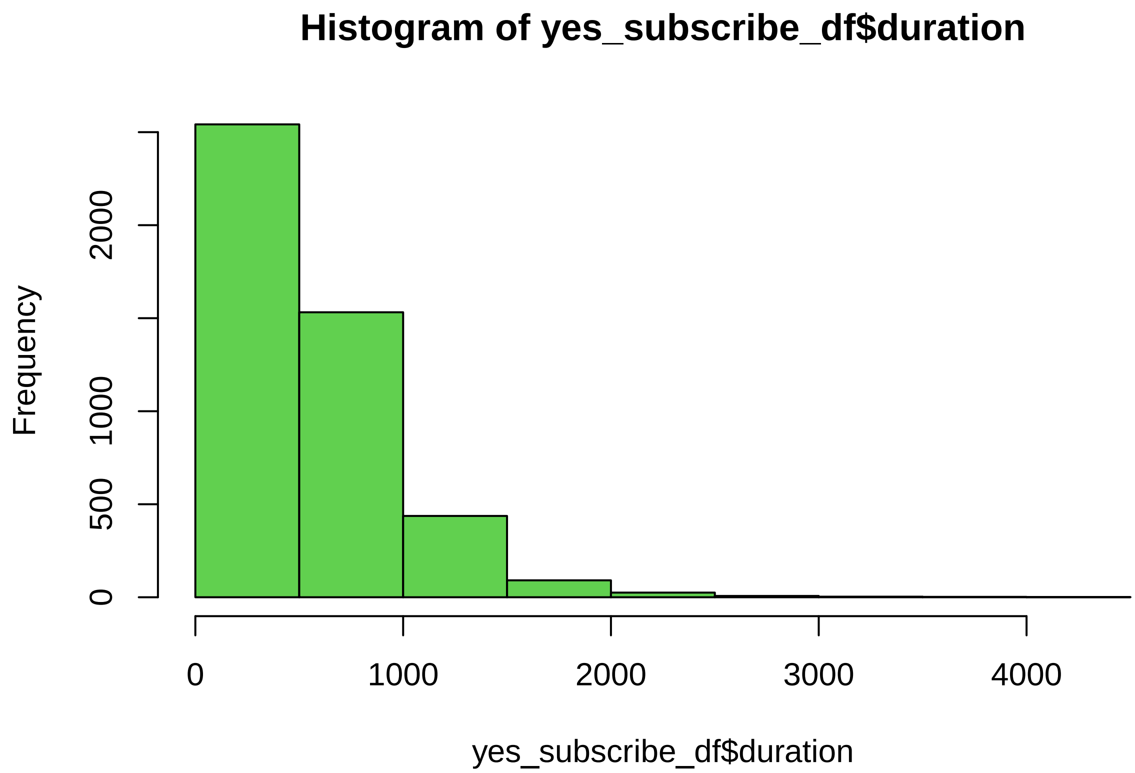
Note that the scales of the three plots are all different ranges because of the differing magnitudes of counts. It is clear, however, that the yes subscibers are right-skewed as opposed to the entire population and the non subscribers. The lower euribor3m rates are positively correlated with subscriptions.

Plot duration

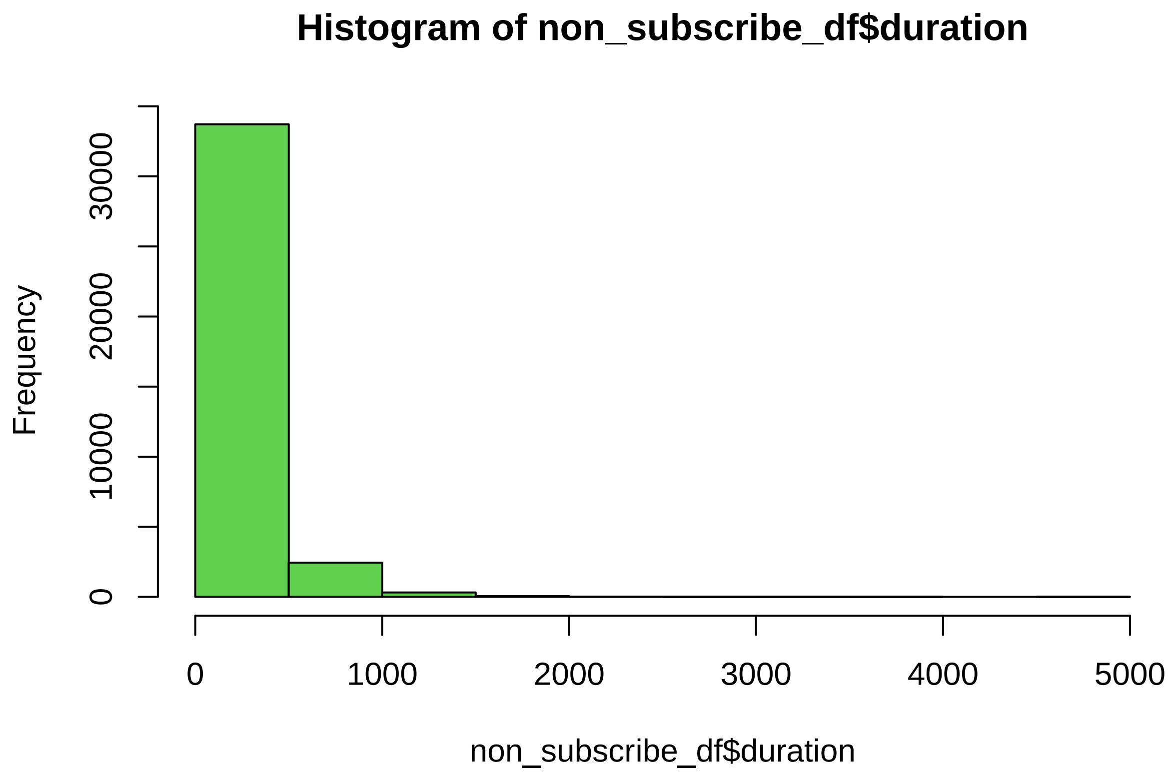
```
hist(bank_df$duration, col = 3)
```




```
hist(yes_subscribe_df$duration, col = 3)
```




```
hist(non_subscribe_df$duration, col = 3)
```



There are clearly more phone calls of longer duration in yes-subscribe.

Prepare data for modeling by scaling, converting to factor

```
library(fastDummies)
bank_df_dummies <- dummy_cols(bank_df[,-21])
bank_df_dummies$y <- bank_df$y
bank_df_dummies$y <- as.numeric(bank_df_dummies$y)
x <- sapply(bank_df_dummies, is.factor)
bank_df_dummies[, x] <- as.data.frame(apply(bank_df_dummies[, x], 2, as.numeric))
bank_df_scaled <- scale(bank_df_dummies[, -74])
bank_df_scaled$y <- bank_df_dummies$y
bank_df_dummies <- bank_df_dummies[, colSums(is.na(bank_df_dummies))==0]
bank_df_dummies$y <- as.factor(bank_df_dummies$y)
```

Create 70/30 train/test split. Use 'sample.split' from the caTools library.

```
library(caTools)
split <- sample.split(bank_df_dummies, SplitRatio = 0.7)
train <- subset(bank_df_dummies, split == "TRUE")
test <- subset(bank_df_dummies, split == "FALSE")
```

Try logistic regression (GLM) on entire dataset

```
set.seed(334)
logit_model <- glm(y ~.,family=binomial(link='logit'),data=train)
anova(logit_model, test = 'Chisq')
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: y
##
## Terms added sequentially (first to last)
##
##
##                               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                          28317      19838
## age                            1     29.5     28316      19808 5.646e-08 ***
## duration                       1   3439.8     28315      16368 < 2.2e-16 ***
## campaign                       1    159.7     28314      16209 < 2.2e-16 ***
## pdays                          1   1733.4     28313      14475 < 2.2e-16 ***
## previous                       1     57.4     28312      14418 3.516e-14 ***
## emp.var.rate                   1   1632.1     28311      12786 < 2.2e-16 ***
## cons.price.idx                 1    285.3     28310      12500 < 2.2e-16 ***
## cons.conf.idx                  1     69.8     28309      12431 < 2.2e-16 ***
## euribor3m                      1      7.7     28308      12423 0.0054117 **
## nr.employed                    1     25.6     28307      12397 4.275e-07 ***
## job_admin.                     1     11.8     28306      12386 0.0005869 ***
## `job_blue-collar`              1     66.7     28305      12319 3.195e-16 ***
## job_entrepreneur               1      5.3     28304      12314 0.0219081 *
## job_housemaid                  1      2.0     28303      12312 0.1570240
## job_management                 1      0.0     28302      12312 0.9864075
## job_retired                    1     16.5     28301      12295 4.894e-05 ***
## `job_self-employed`            1      0.4     28300      12295 0.5359609
## job_services                   1     15.2     28299      12280 9.552e-05 ***
## job_student                    1      0.8     28298      12279 0.3658180
## job_technician                 1      1.1     28297      12278 0.3001426
## job_unemployed                 1      0.6     28296      12277 0.4553975
## job_unknown                    0      0.0     28296      12277
## marital_divorced               1      0.0     28295      12277 0.9922684
## marital_married                1      5.8     28294      12271 0.0162978 *
## marital_single                 1      0.0     28293      12271 0.9209687
## marital_unknown                0      0.0     28293      12271
## education_basic.4y             1      1.0     28292      12270 0.3247065
## education_basic.6y             1      0.0     28291      12270 0.9918182
## education_basic.9y             1      3.8     28290      12267 0.0526065 .
## education_high.school          1     15.6     28289      12251 7.883e-05 ***
## education_illiterate           1      3.8     28288      12247 0.0513542 .
## education_professional.course  1      4.5     28287      12243 0.0342142 *
## education_university.degree    1      1.5     28286      12241 0.2234985
## education_unknown              0      0.0     28286      12241
## default_no                     1     33.3     28285      12208 7.862e-09 ***
## default_unknown                1      0.0     28284      12208 0.8573630
## default_yes                    0      0.0     28284      12208
## housing_no                     2      0.2     28282      12208 0.8960314
## housing_unknown                1      0.8     28281      12207 0.3711310
## housing_yes                    0      0.0     28281      12207
## loan_no                        1      1.5     28280      12205 0.2180363
## loan_unknown                   0      0.0     28280      12205
## loan_yes                       0      0.0     28280      12205
```

```
## contact_cellular              1    150.2     28279      12055 < 2.2e-16 ***
## contact_telephone             0      0.0     28279      12055
## month_apr                     1      3.0     28278      12052 0.0834895 .
## month_aug                     1     55.0     28277      11997 1.191e-13 ***
## month_dec                     1      0.0     28276      11997 0.8266879
## month_jul                     1     36.1     28275      11961 1.864e-09 ***
## month_jun                     1     17.9     28274      11943 2.283e-05 ***
## month_mar                     0    233.0     28274      11710
## month_may                     2      1.5     28272      11708 0.4637919
## month_nov                     1     22.9     28271      11686 1.748e-06 ***
## month_oct                     1      0.0     28270      11686 0.9304640
## month_sep                     0      0.0     28270      11686
## day_of_week_fri               1      0.3     28269      11685 0.5577694
## day_of_week_mon               1     15.3     28268      11670 9.049e-05 ***
## day_of_week_thu               1      0.0     28267      11670 1.0000000
## day_of_week_tue               1      0.1     28266      11670 0.7714634
## day_of_week_wed               0      0.0     28266      11670
## poutcome_failure              1     22.4     28265      11648 2.224e-06 ***
## poutcome_nonexistent          1      5.6     28264      11642 0.0180190 *
## poutcome_success              0      0.0     28264      11642
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
library(pscl)
```

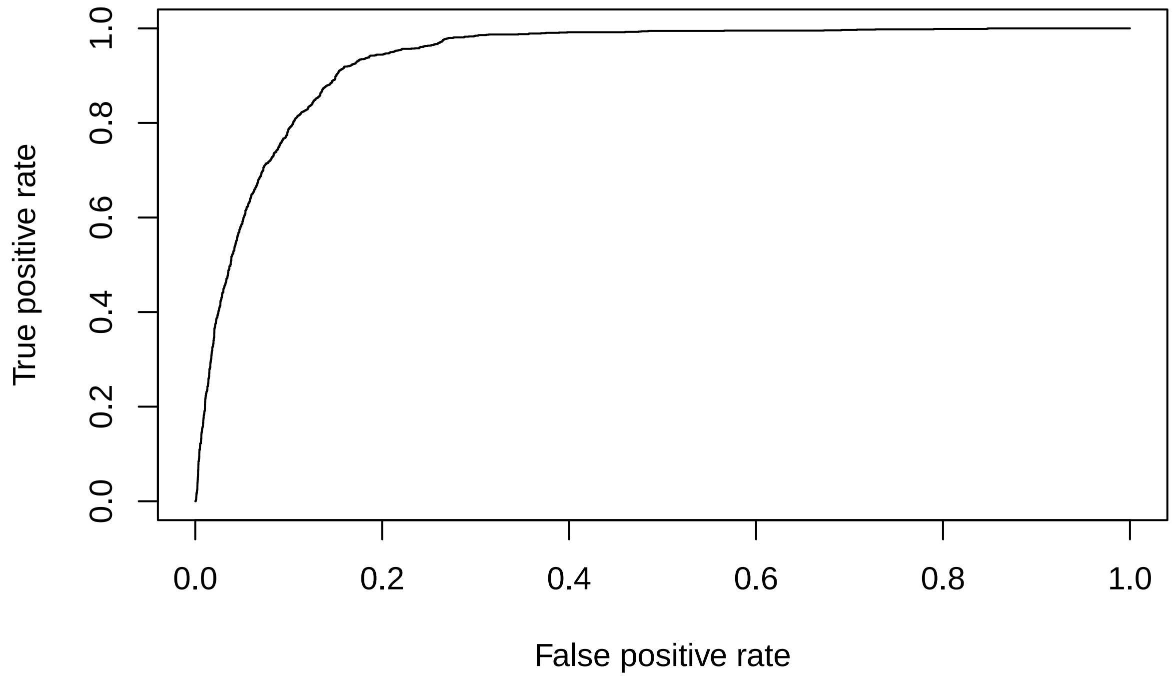
```
## Classes and Methods for R developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis
```

```
pR2(logit_model)
```

```
## fitting null model for pseudo-r2
```

```
##           llh       llhNull            G2      McFadden          r2ML
## -5820.9627595 -9918.8236818  8195.7218446     0.4131398     0.2513004
##          r2CU
##     0.4989301
```

```
library(ROCR)
p <- predict(logit_model,newdata=test,type='response')
pr <- prediction(p, test$y)
prf <- performance(pr, measure = 'tpr', x.measure = 'fpr')
plot(prf)
```




```
auc <- performance(pr, measure = 'auc')
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.9351181
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
head(p, 10)
```

```
##           3           5           8          14          17          24
## 0.011189976 0.015152515 0.006942150 0.015714896 0.016387647 0.010414835
##          26          28          31          36
## 0.007565746 0.007486587 0.023657451 0.021287821
```

```
p.rd <- ifelse(p > 0.5, 1, 0)
head(p.rd, 10)
```

```
##  3  5  8 14 17 24 26 28 31 36
##  0  0  0  0  0  0  0  0  0  0
```

```r
confusionMatrix(table(p.rd,test[,18]))
```

```
## Confusion Matrix and Statistics
##
##
## p.rd     0     1
##    0 10780  1157
##    1   866    67
##
##                Accuracy : 0.8428
##                  95% CI : (0.8364, 0.8491)
##     No Information Rate : 0.9049
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.022
##
##  Mcnemar's Test P-Value : 1.136e-10
##
##             Sensitivity : 0.92564
##             Specificity : 0.05474
##          Pos Pred Value : 0.90307
##          Neg Pred Value : 0.07181
##              Prevalence : 0.90490
##          Detection Rate : 0.83761
##    Detection Prevalence : 0.92751
##       Balanced Accuracy : 0.49019
##
##        'Positive' Class : 0
##
```

Note the utter failure in specificity(0.03686). Not a useful model.

Try glm with only duration and emp.var.rate

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:vctrs':
##
##     data_frame

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
bank_df_subset <- bank_df %>% select(duration, emp.var.rate, y)
bank_df_subset_scaled <- scale(bank_df_subset[, -3])
bank_df_subset_scaled <- as.data.frame(bank_df_subset_scaled)
bank_df_subset_scaled$y <- bank_df_subset$y
```

Splitting data in train and test data

```r
train <- bank_df_subset_scaled[1:30000,]
test <-  bank_df_subset_scaled[30001:41188,]
```

Create GLM

```r
set.seed(221)
logit_model <- glm(y ~.,family=binomial(link='logit'),data=train)
anova(logit_model, test = 'Chisq')
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: y
##
## Terms added sequentially (first to last)
##
##
##              Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                         29999    13151.6
## duration      1   3931.3     29998     9220.3 < 2.2e-16 ***
## emp.var.rate  1    419.3     29997     8801.0 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

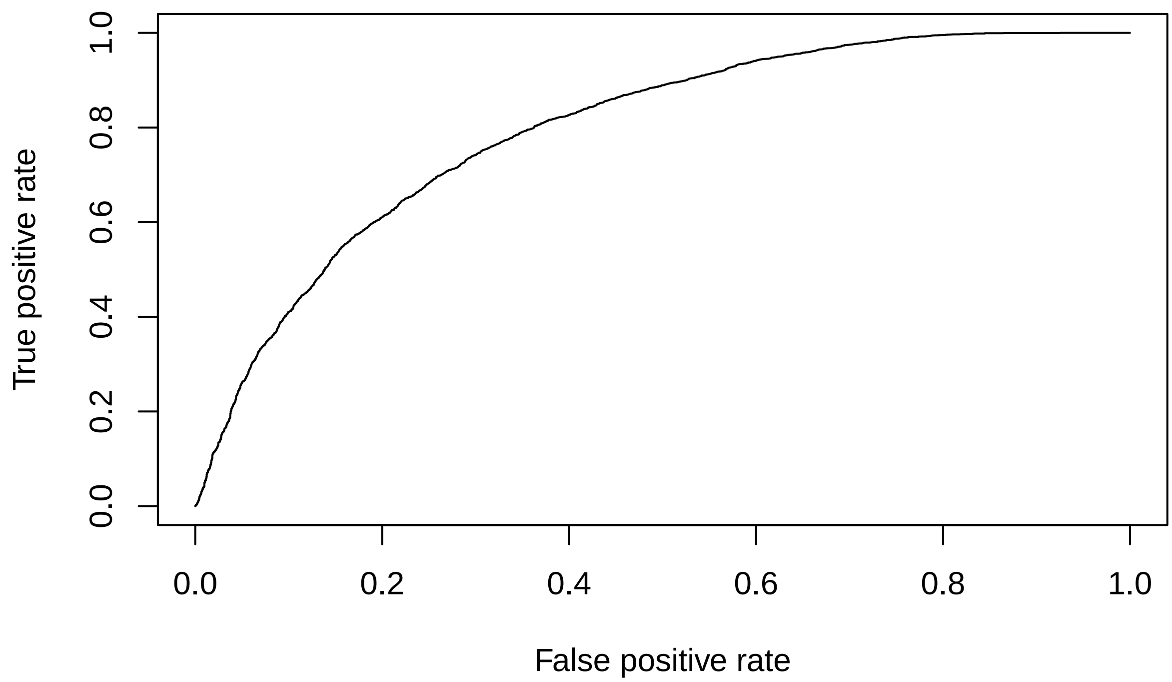
Must resort to pseudo-Rsquared instead of Rsquared because this is a classification problem.

```r
library(pscl)
pR2(logit_model)
```

```
## fitting null model for pseudo-r2
```

```
##           llh       llhNull            G2      McFadden          r2ML
## -4400.4840275 -6575.7805055  4350.5929560     0.3308043     0.1349948
##          r2CU
##     0.3803497
```

```r
library(ROCR)
p <- predict(logit_model,newdata=test,type='response')
pr <- prediction(p, test$y)
prf <- performance(pr, measure = 'tpr', x.measure = 'fpr')
plot(prf)
```




```
auc <- performance(pr, measure = 'auc')
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.7942699
```

The subset model is not as robust as the GLM on the entire dataset

Try RandomForest. No need to scale because it is tree-based Fitting Random Forest to the train dataset

```
bank_df_dummies <- bank_df_dummies[, colSums(is.na(bank_df_dummies))==0]
bank_df_dummies$y <- as.factor(bank_df_dummies$y)
```

Split data train/test

```
library(caTools)
split <- sample.split(bank_df_dummies, SplitRatio = 0.7)
train <- subset(bank_df_dummies, split == "TRUE")
test <- subset(bank_df_dummies, split == "FALSE")
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

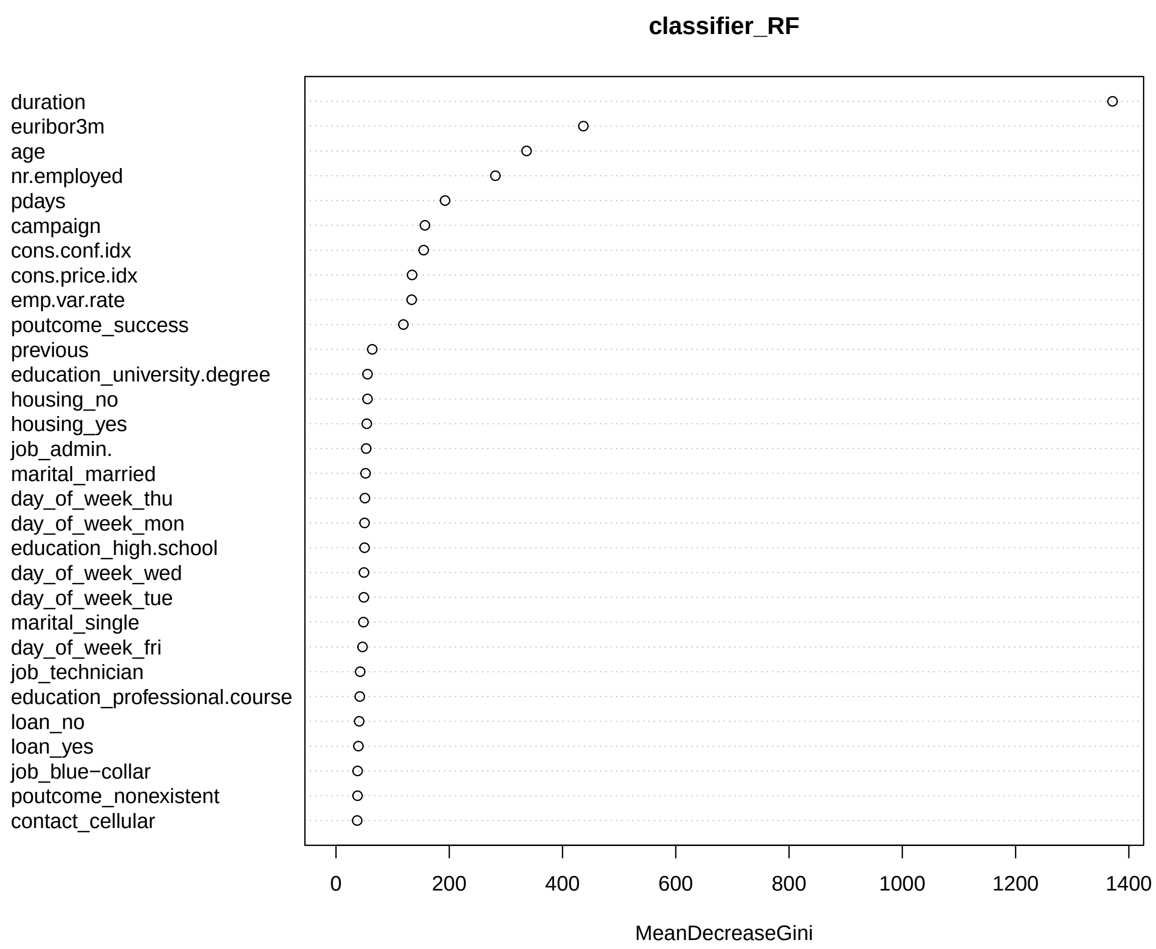
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'

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

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

```
set.seed(126) # Setting seed
classifier_RF = randomForest(x = train[-64],
                             y = train$y,
                             keep.forest = TRUE,
                             ntree = 400)
```

Variable Importance Plot

```
varImpPlot(classifier_RF)
```

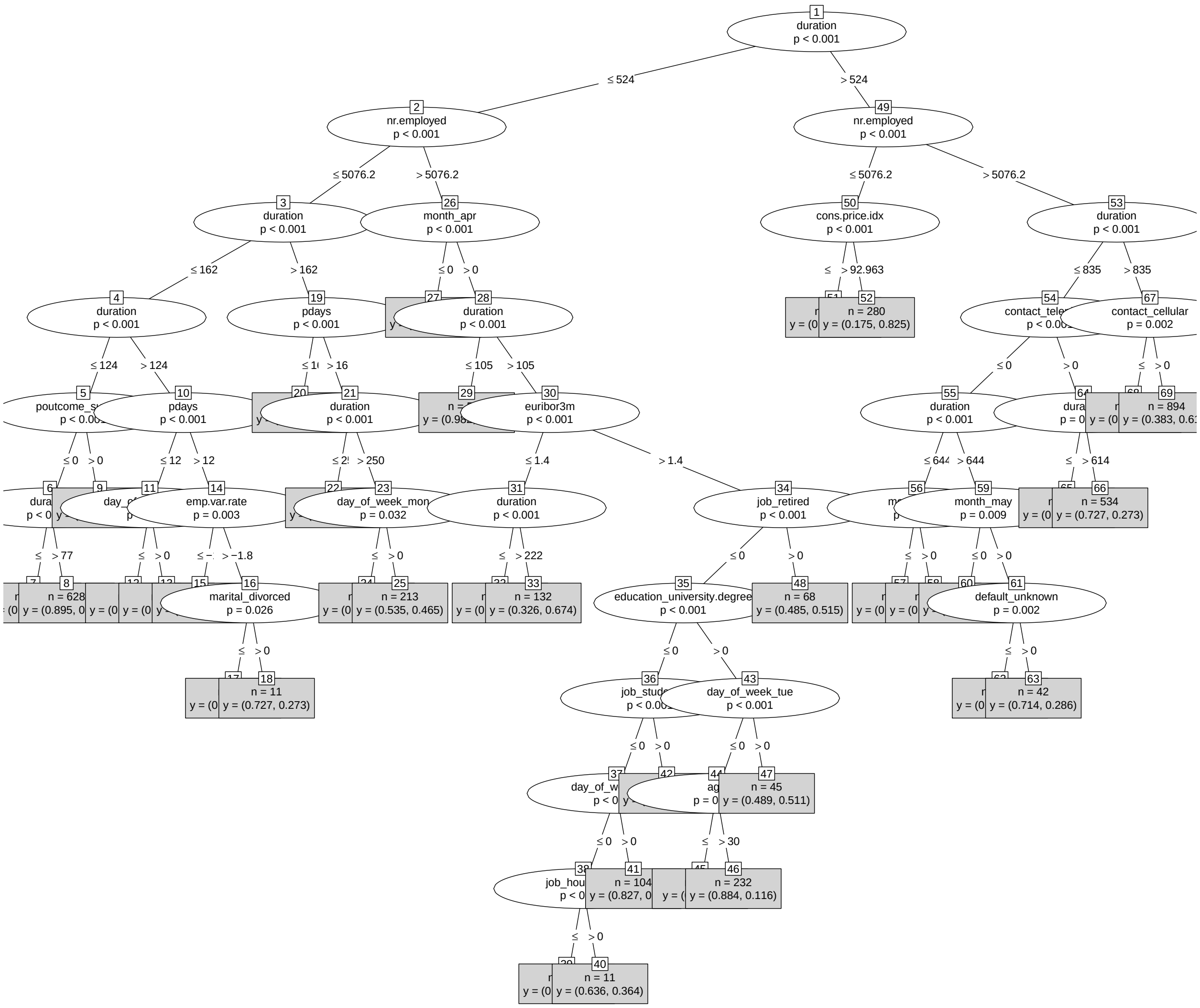


Display the decision trees

```r
library(party)
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

##
## Attaching package: 'party'

## The following object is masked from 'package:dplyr':
##
##     where
```

```r
x <- ctree(y ~ ., data = bank_df_dummies)
plot(x, type = 'simple')
```



Accuracy on test set

```r
accuracy = (24469 + 1342)/(24469 + 1342 + 1836 + 669)
accuracy
```

```
## [1] 0.9115341
```

Score model on test set

```r
library(caret)
y_predict <- predict(classifier_RF, test)
confusionMatrix(y_predict, test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2
##          1 11095   839
```

```
##          2   300   636
##
##                 Accuracy : 0.9115
##                   95% CI : (0.9065, 0.9164)
##      No Information Rate : 0.8854
##      P-Value [Acc > NIR] : < 2.2e-16
##
##                    Kappa : 0.4814
##
##   Mcnemar's Test P-Value : < 2.2e-16
##
##              Sensitivity : 0.9737
##              Specificity : 0.4312
##           Pos Pred Value : 0.9297
##           Neg Pred Value : 0.6795
##               Prevalence : 0.8854
##           Detection Rate : 0.8621
##     Detection Prevalence : 0.9273
##        Balanced Accuracy : 0.7024
##
##         'Positive' Class : 1
##
```

More robust than logistic regression, but specificity well below 50%.

Fit PCR Model to visualize important components. Calculate how much we can minimize features to model.

```
library(ISLR)
library(pls)
```

```
##
## Attaching package: 'pls'
```

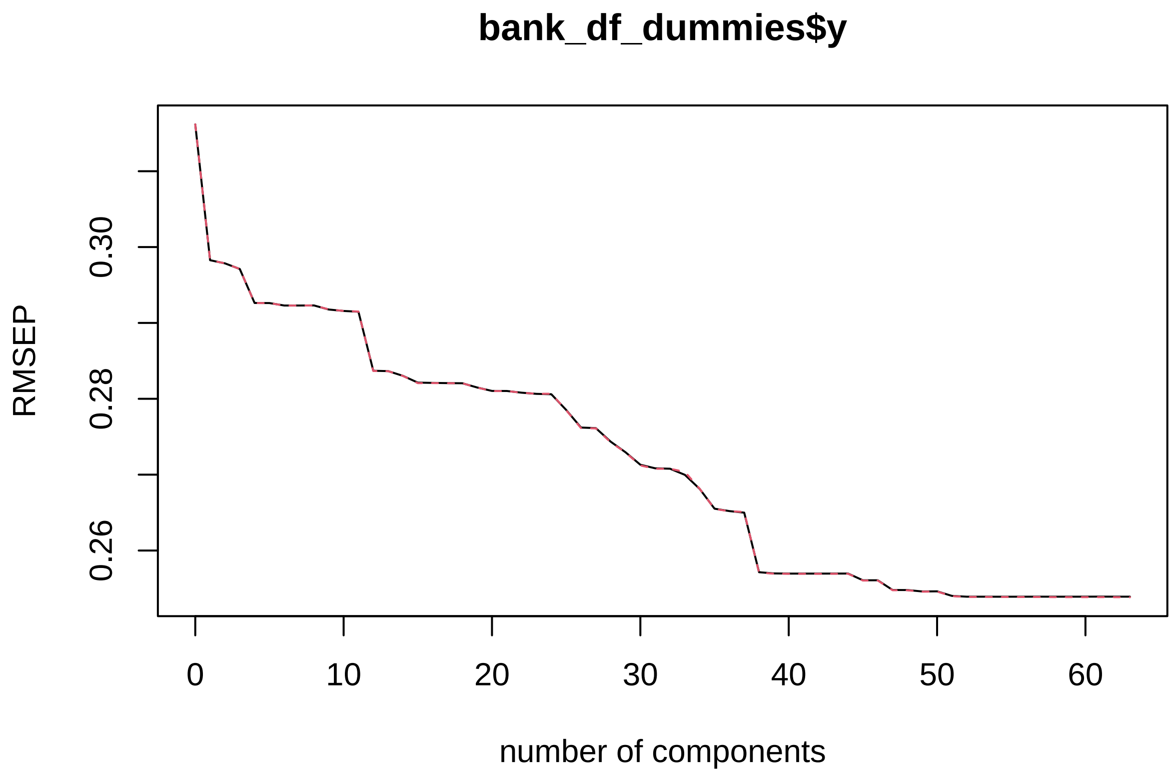
```
## The following object is masked from 'package:caret':
##
##     R2
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

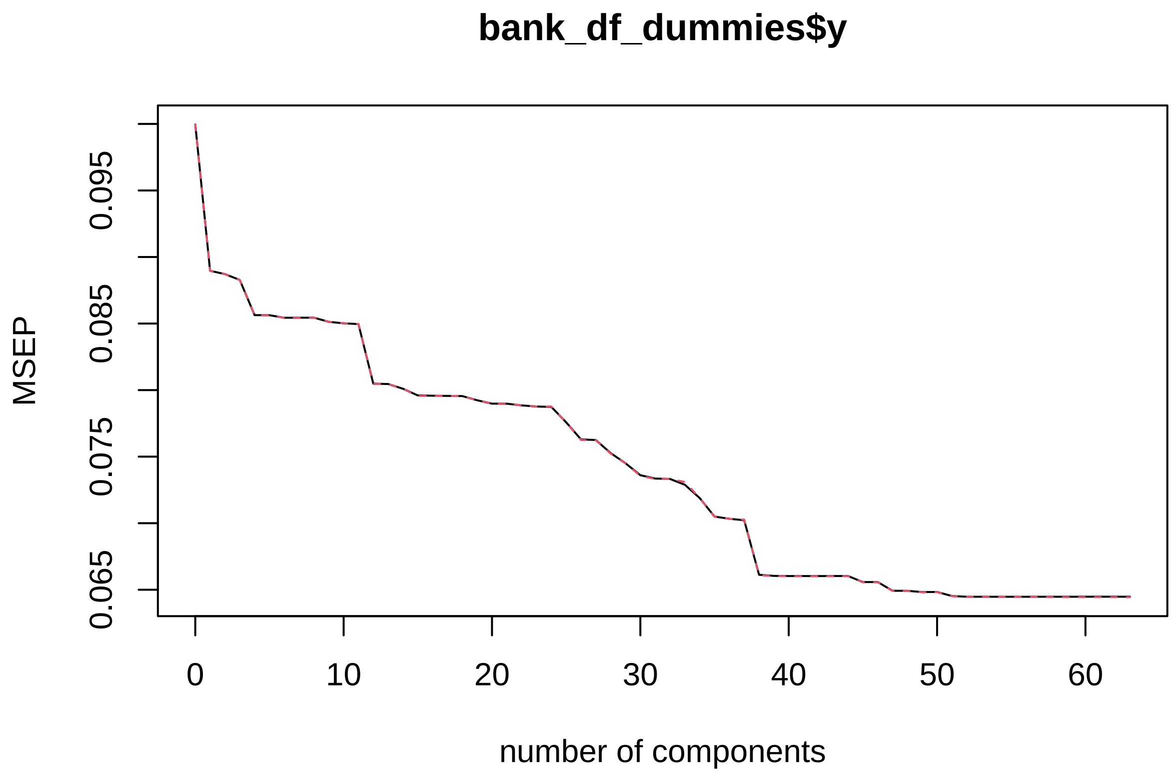
```
set.seed(2)
bank_df_dummies$y <- as.numeric(bank_df_dummies$y)
pcr.fit=pcr(bank_df_dummies$y~., data=bank_df_dummies, scale=TRUE,
            validation ="CV")
```

Visualize cross-validation plots

```
validationplot(pcr.fit)
```




```
validationplot(pcr.fit, val.type="MSEP")
```



One must use upwards of 37 components to create a viable model. Calculate log loss

Try kNN. k = 60

```
library(caret)
library(class)
set.seed(234)
knn_60 <- knn(train=train[-64], test=test[-64], cl=train$y, k=60)
ACC.60 <- 100 * sum(test$y == knn_60)/NROW(test$y)
ACC.60
```

```
## [1] 90.97902
```

```
table(knn_60, test$y)
```

```
##
## knn_60     1     2
##      1 11032   798
##      2   363   677
```

```
confusionMatrix(table(knn_60, test$y))
```

```
## Confusion Matrix and Statistics
##
##
```

```
## knn_60     1     2
##      1 11032   798
##      2   363   677
##
##                Accuracy : 0.9098
##                  95% CI : (0.9047, 0.9147)
##     No Information Rate : 0.8854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.49
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9681
##             Specificity : 0.4590
##          Pos Pred Value : 0.9325
##          Neg Pred Value : 0.6510
##              Prevalence : 0.8854
##          Detection Rate : 0.8572
##    Detection Prevalence : 0.9192
##       Balanced Accuracy : 0.7136
##
##        'Positive' Class : 1
##
```

The best kNN does not produce 50% specificity.

Create SVM Model

```r
# Create the classifier here
# install.packages("e1071")
# you can also use kernlab
library(e1071)
classifier <- svm(formula = y ~ .,
                  data = train,
                  type = "C-classification",
                  kernel = "radial",
                  cost = 0.25,
                  cross = 10,
                  sigma = 1.22723
)
y_pred <-  predict(classifier, newdata=test[-64])
```

```r
library(caret)
confusionMatrix(data = (y_pred),
                reference = test[,64])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2
##          1 11245  1125
##          2   150   350
##
```

```
##                Accuracy : 0.9009
##                  95% CI : (0.8956, 0.906)
##     No Information Rate : 0.8854
##     P-Value [Acc > NIR] : 8.901e-09
##
##                   Kappa : 0.3147
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9868
##             Specificity : 0.2373
##          Pos Pred Value : 0.9091
##          Neg Pred Value : 0.7000
##              Prevalence : 0.8854
##          Detection Rate : 0.8737
##    Detection Prevalence : 0.9611
##       Balanced Accuracy : 0.6121
##
##        'Positive' Class : 1
##
```

This SVM Model failed miserably in predicting subscribers, 1125 false positives, and only 350 true positives.

Try bagged model

```r
library(dplyr)       #for data wrangling
library(e1071)       #for calculating variable importance
library(caret)       #for general model fitting
library(rpart)       #for fitting decision trees
library(ipred)       #for fitting bagged decision trees
set.seed(1)

#fit the bagged model
bag <- bagging(
  formula = y ~ .,
  data = train,
  nbagg = 150,
  coob = TRUE,
  control = rpart.control(minsplit = 2, cp = 0)
)

#display fitted bagged model
bag
```

```
##
## Bagging classification trees with 150 bootstrap replications
##
## Call: bagging.data.frame(formula = y ~ ., data = train, nbagg = 150,
##     coob = TRUE, control = rpart.control(minsplit = 2, cp = 0))
##
## Out-of-bag estimate of misclassification error:  0.0888
```

```r
pred <- predict(object = bag,
                newdata = test,
                type = "class")
```

```r
library(caret)
confusionMatrix(data = as.factor(pred),
                reference = as.factor(test$y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2
##          1 10980   677
##          2   415   798
##
##                Accuracy : 0.9152
##                  95% CI : (0.9102, 0.9199)
##     No Information Rate : 0.8854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5469
##
##  Mcnemar's Test P-Value : 2.829e-15
##
##             Sensitivity : 0.9636
##             Specificity : 0.5410
##          Pos Pred Value : 0.9419
##          Neg Pred Value : 0.6579
##              Prevalence : 0.8854
##          Detection Rate : 0.8531
##    Detection Prevalence : 0.9057
##       Balanced Accuracy : 0.7523
##
##        'Positive' Class : 1
##
```

Try xgboost model

```r
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```r
library(caret)
set.seed(112)
model_gbm = gbm(y ~.,
                data = train,
                distribution = "multinomial",
                cv.folds = 10,
                shrinkage = .01,
                n.minobsinnode = 10,
                n.trees = 500)       # 500 trees to be built
```

```r
pred_test = predict.gbm(object = model_gbm,
                        newdata = test,
                        n.trees = 500,           # 500 trees to be built
                        type = "response")
# Give class names to the highest prediction value.
class_names = colnames(pred_test)[apply(pred_test, 1, which.max)]
result = data.frame(test$y, class_names)
conf_mat = confusionMatrix(test$y, as.factor(class_names))
print(conf_mat)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2
##          1 11213   182
##          2  1018   457
##
##                Accuracy : 0.9068
##                  95% CI : (0.9016, 0.9117)
##     No Information Rate : 0.9503
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3901
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9168
##             Specificity : 0.7152
##          Pos Pred Value : 0.9840
##          Neg Pred Value : 0.3098
##              Prevalence : 0.9503
##          Detection Rate : 0.8713
##    Detection Prevalence : 0.8854
##       Balanced Accuracy : 0.8160
##
##        'Positive' Class : 1
##
```

This is the most robust model with a specificity of 0.7320.