# Predic422-CharityProject Part 3

*Artur Mrozowski*

*May 28, 2017*

Load packages required for this code.

```r
# Load packages required for this code.
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 3.2.3
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(lift)
```

```
## Warning: package 'lift' was built under R version 3.2.5
```

```r
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.2.2
```

```r
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.2.3
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.3
```

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

```
## Warning: package 'lattice' was built under R version 3.2.2
```

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

```
## Warning: package 'ggplot2' was built under R version 3.2.3
```

Exercise 1 Read Data from CSV File

```
charityData = read.csv(file.choose(),na.strings=c("NA"," "))
```

Convert categorical variables to factors

```
charityData$DONR = as.factor(charityData$DONR)
charityData$HOME = as.factor(charityData$HOME)
charityData$HINC = as.factor(charityData$HINC)
```

Rename the dataset to classData for clarity. Remove charityData from R session environment

```
classData = charityData
rm(charityData)
```

```
## Check for Missing Values
which(sapply(classData,anyNA))
```

```
##   HOME   HINC GENDER
##      5      6      7
```

```
# HOME - Make a level 0 and code missing values as 0
levels(classData$HOME) = c(levels(classData$HOME),"0")
classData$HOME[is.na(classData$HOME)] = "0"
table(classData$HOME,useNA="ifany")
```

```
##
##     0     1
## 23899 46972
```

```
# HINC - Make a level 0 and code missing values as 0
levels(classData$HINC) = c(levels(classData$HINC),"0")
classData$HINC[is.na(classData$HINC)] = "0"
table(classData$HINC,useNA="ifany")
```

```
##
##     1     2     3     4     5     6     7     0
##  7084 10616  7189 10983 13454  6770  6657  8118
```

```
# GENDER - Assign A, J, and NA to category U
idxMF = classData$GENDER %in% c("M","F")
classData$GENDER[!idxMF] = "U"
classData$GENDER = factor(classData$GENDER)
table(classData$GENDER)
```

```
##
##     F     M     U
## 38183 30494  2194
```

Part B - Derived or Transformed Variables(Optional)

Part C - Re-categorize Variables

```r
# Separate RFA Values (R = recency, F = frequency, A = amount)
separateRFA = function(xData,varName)
{
  bytes = c("R","F","A")
  newVarNames = paste(varName,bytes, sep="_")

  for (ii in 1:length(bytes)) # Loop over 1 to 3 (corresponding to R, F, and A)
  {
    # Find the unique values for current byte
    byteVals = unique(substr(levels(xData[,varName]),ii,ii))

    for (jj in 1:length(byteVals)) # Loop over unique byte values
    {
      rowIdx = substr(xData[,varName],ii,ii) == byteVals[jj]
      xData[rowIdx,newVarNames[ii]] = byteVals[jj]
    }

    xData[,newVarNames[ii]] = factor(xData[,newVarNames[ii]])
  }

  return(xData)
}
```

```r
# Apply separateRFA to the variables RFA_96 and check results.

classData = separateRFA(classData,"RFA_96")
#table(classData$RFA_96,classData$RFA_96_R)
#table(classData$RFA_96,classData$RFA_96_F)
#table(classData$RFA_96,classData$RFA_96_A)
```

Part D - Drop Variables

```r
dropIdx = which(names(classData) %in% c("DAMT","RFA_96"))

# Drop the variables indicated by dropIdx.
classData2 = classData[,-dropIdx]
names(classData2)   # check that the result is as expected
```

```
##  [1] "ID"       "DONR"     "AGE"      "HOME"     "HINC"     "GENDER"
##  [7] "MEDAGE"   "MEDPPH"   "MEDHVAL"  "MEDINC"   "MEDEDUC"  "NUMPROM"
## [13] "NUMPRM12" "RAMNTALL" "NGIFTALL" "MAXRAMNT" "LASTGIFT" "TDON"
## [19] "RFA_96_R" "RFA_96_F" "RFA_96_A"
```

Exercise 3 Dataset Partitioning

```r
# Specify the fraction of data to use in the hold-out test.
testFraction = 0.25
set.seed(123)
```

```r
# Sample training subset indices.

trainIdx = sample(nrow(classData2),size=(1-testFraction)*nrow(classData2),
                  replace=FALSE)
```

Exercise 4 Model Fitting

```r
glm.fit=glm(DONR~ AGE+MEDAGE+MEDHVAL+MEDINC+MEDEDUC+NUMPROM+MAXRAMNT +MEDINC+MEDEDUC+ NUMPROM+NUMPRM12+
backwards=step(glm.fit,trace=0)
formula(backwards)
```

```
## DONR ~ AGE + MEDAGE + MEDHVAL + MEDINC + NUMPROM + NUMPRM12 +
##     RAMNTALL + TDON + RFA_96_F + RFA_96_A
```

```r
summary(backwards)
```

```
##
## Call:
## glm(formula = DONR ~ AGE + MEDAGE + MEDHVAL + MEDINC + NUMPROM +
##     NUMPRM12 + RAMNTALL + TDON + RFA_96_F + RFA_96_A, family = binomial,
##     data = classData2, subset = trainIdx)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8107  -0.3530  -0.3018  -0.2631   2.8164
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.054e+01  8.444e+01  -0.125  0.90065
## AGE         -2.944e-03  1.287e-03  -2.288  0.02213 *
## MEDAGE       5.556e-03  2.587e-03   2.147  0.03176 *
## MEDHVAL      1.125e-04  2.696e-05   4.172 3.01e-05 ***
## MEDINC       2.724e-04  1.585e-04   1.719  0.08564 .
## NUMPROM      4.338e-03  1.344e-03   3.227  0.00125 **
## NUMPRM12    -1.507e-02  5.820e-03  -2.590  0.00959 **
## RAMNTALL     4.148e-04  2.156e-04   1.924  0.05437 .
## TDON        -3.755e-02  6.083e-03  -6.172 6.73e-10 ***
## RFA_96_F2    2.108e-01  5.299e-02   3.977 6.97e-05 ***
## RFA_96_F3    2.863e-01  6.283e-02   4.556 5.21e-06 ***
## RFA_96_F4    3.846e-01  7.097e-02   5.419 6.00e-08 ***
## RFA_96_AC    7.983e+00  8.445e+01   0.095  0.92469
## RFA_96_AD    8.146e+00  8.444e+01   0.096  0.92315
## RFA_96_AE    7.960e+00  8.444e+01   0.094  0.92490
## RFA_96_AF    7.764e+00  8.444e+01   0.092  0.92674
## RFA_96_AG    7.408e+00  8.444e+01   0.088  0.93009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 21276  on 53152  degrees of freedom
## Residual deviance: 20869  on 53136  degrees of freedom
## AIC: 20903
##
## Number of Fisher Scoring iterations: 9
```

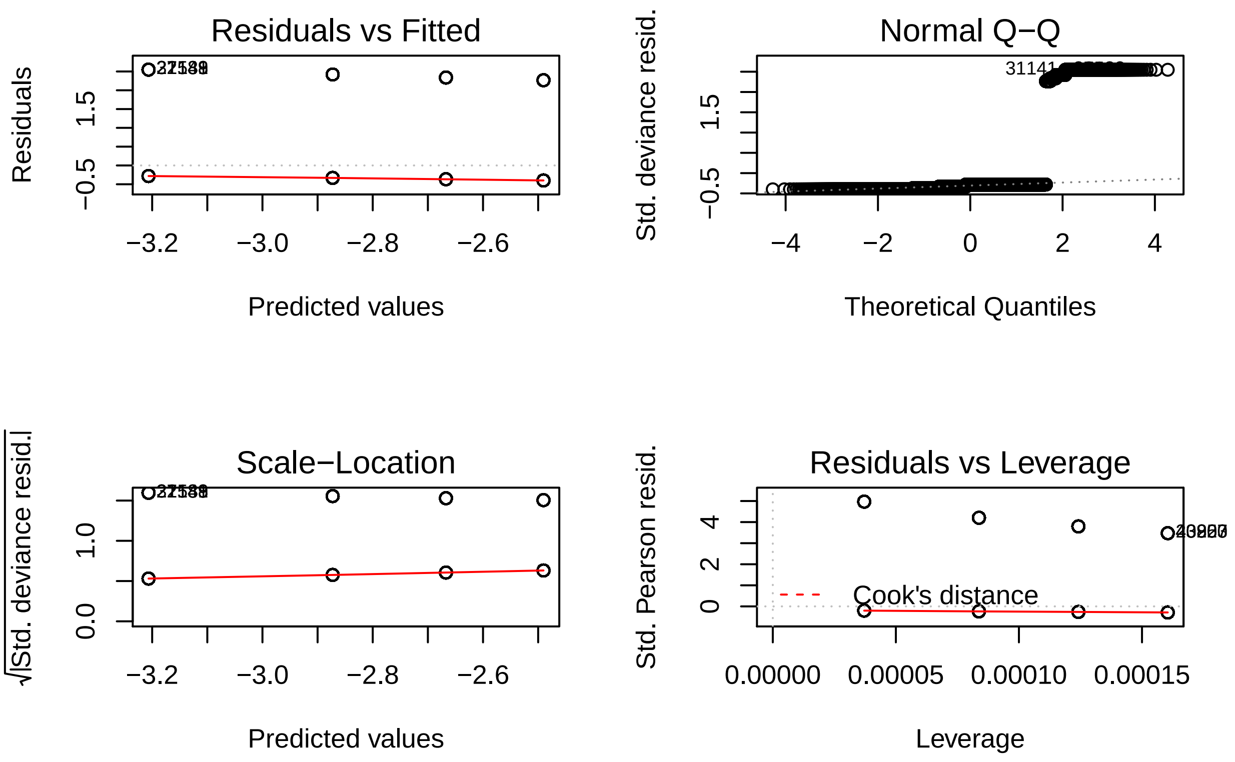
Part A - Simple Logistic Regression

One of the variables with considerable significance is RFA_96_R. I will now fit logistic regression using that variable.

```r
#modelA1 = glm(DONR ~ MAXRAMNT,data=classData2,subset=trainIdx,family=binomial)
modelA1 = glm(DONR ~ RFA_96_F,data=classData2,subset=trainIdx,family=binomial)
summary(modelA1)
```

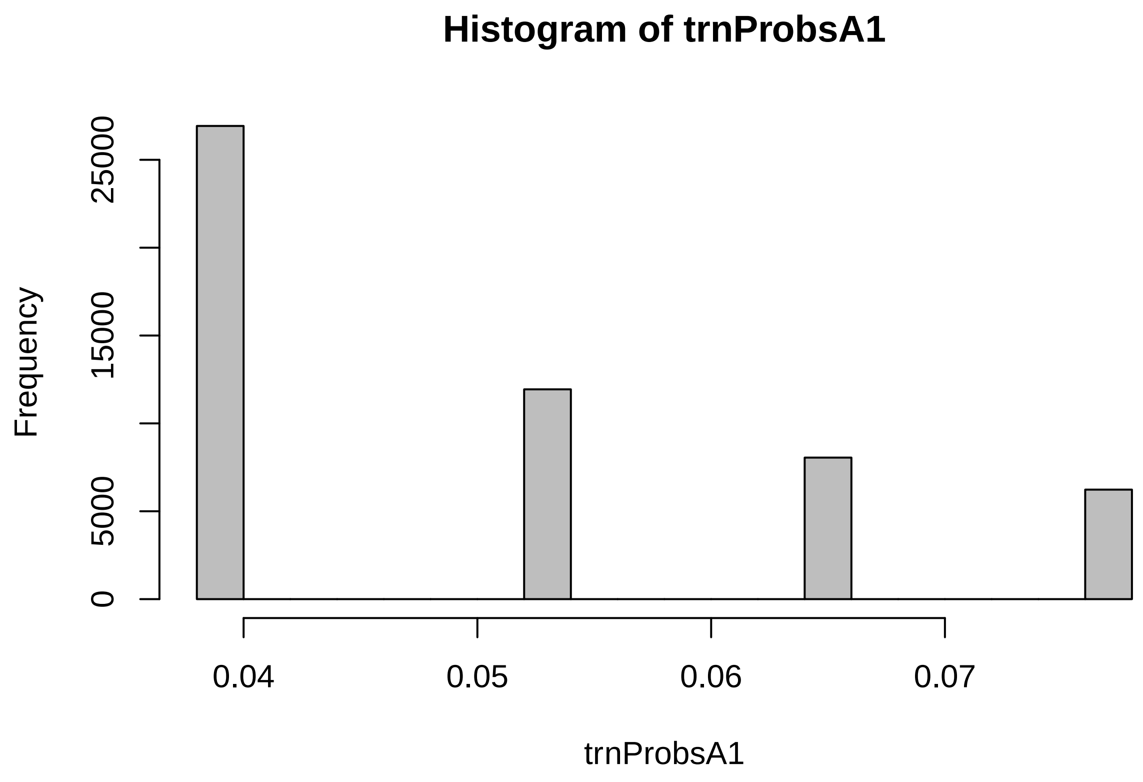
```
##
## Call:
## glm(formula = DONR ~ RFA_96_F, family = binomial, data = classData2,
##     subset = trainIdx)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.3991  -0.3317  -0.2818  -0.2818   2.5480
##
## Coefficients:
##             Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -3.20659    0.03151 -101.766  < 2e-16 ***
## RFA_96_F2    0.33393    0.05144    6.491  8.5e-11 ***
## RFA_96_F3    0.53939    0.05512    9.787  < 2e-16 ***
## RFA_96_F4    0.71628    0.05712   12.539  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 21276  on 53152  degrees of freedom
## Residual deviance: 21083  on 53149  degrees of freedom
## AIC: 21091
##
## Number of Fisher Scoring iterations: 6
```

```r
par(mfrow=c(2,2))
plot(modelA1)
```

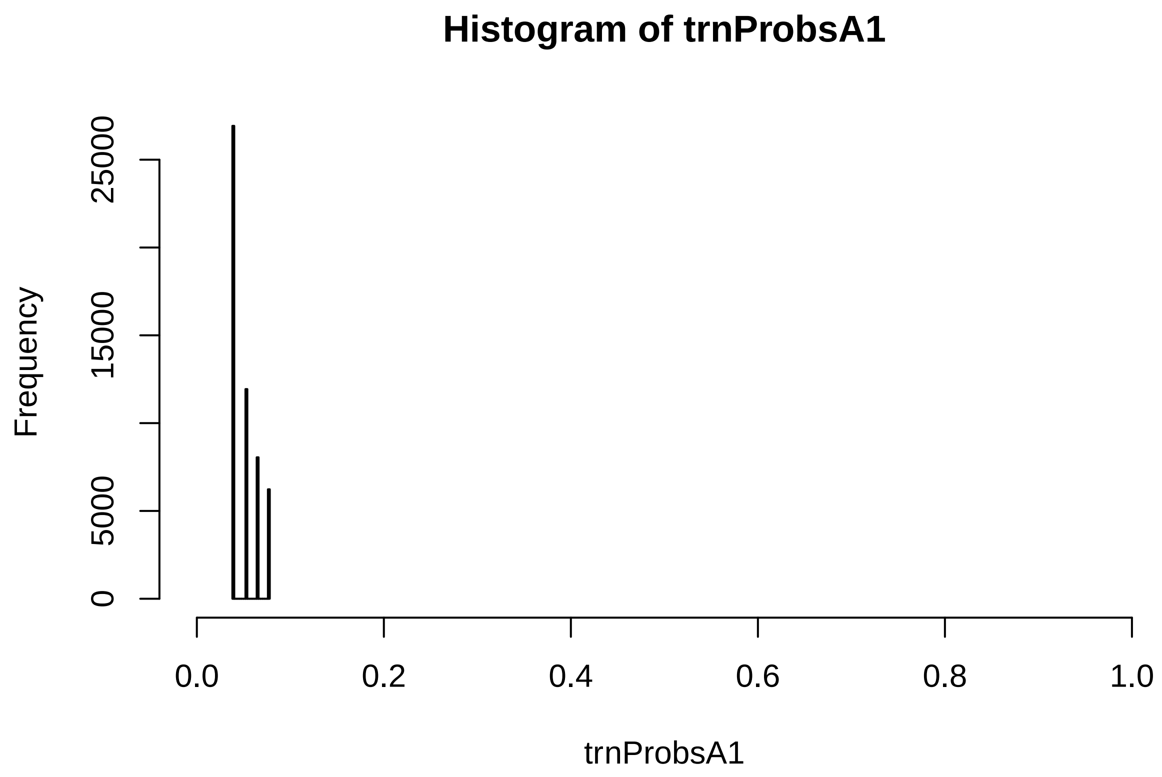


```r
par(mfrow=c(1,1))
```

```r
trnProbsA1 = predict(modelA1,type="response")
hist(trnProbsA1,col="gray")    # Note that scores are distributed around 0.05.
```



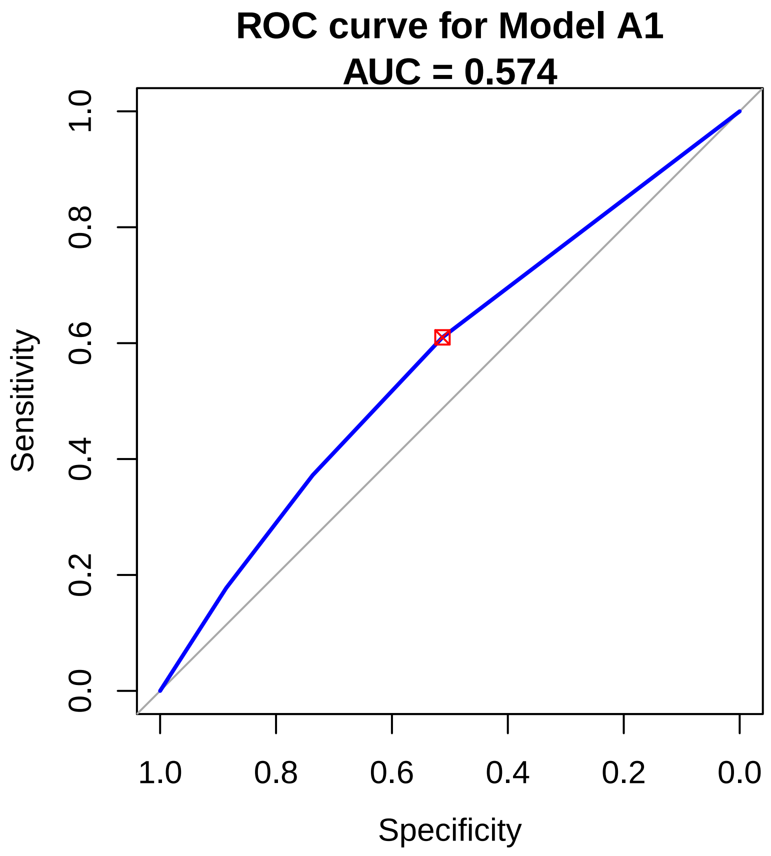
```r
hist(trnProbsA1,col="gray",xlim=c(0,1))    # Rescale to make obvious.
```



```r
# Classification: ROC Curve for Model A1 - Use methods from pROC package.
rocA1 = roc(response=classData2$DONR[trainIdx],predictor=trnProbsA1)
par(pty="s")  # sets plotting region to a square, useful for ROC curves
# Use par(pty="m") to return to default of rectangular plotting region.
plot(rocA1,col="blue",
     main=paste("ROC curve for Model A1\nAUC = ",round(rocA1$auc,digits=3),sep=""))
```

```
##
## Call:
## roc.default(response = classData2$DONR[trainIdx], predictor = trnProbsA1)
##
## Data: trnProbsA1 in 50466 controls (classData2$DONR[trainIdx] 0) < 2687 cases (classData2$DONR[trainI
## Area under the curve: 0.5739
```

```r
par(pty="m")
# Classification: Determine "optimal" threshold.
dist01 = sqrt((rocA1$specificities-1)^2 + (rocA1$sensitivities-1)^2)
optIdxA1 = which.min(dist01)  # index corresponding to minimum distance
threshA1 = rocA1$thresholds[optIdxA1]  # threshold corresponding to min. distance
points(rocA1$specificities[optIdxA1],rocA1$sensitivities[optIdxA1],col="red",pch=7)
```



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.