Kickstarter Data

Columns Explanation

1) ID = internal kickstarter id
2) name = name of project - A project is a finite work with a clear goal that you'd like to bring to life. Think albums, books, or films.
3) category = category
4) main_category = category of campaign
5) currency = currency used to support
6) deadline = deadline for crowdfunding
7) goal = fundraising goal - The funding goal is the amount of money that a creator needs to complete their project.
8) launched = launched date launched
9) pledged = amount pledged by "crowd"
10) state = Current condition the project is in
11) backers = number of backers
12) country = country pledged from
13) usd pledged = amount of money pledged

Packages Needed to Download

```
#install.packages("ggplot2")
library(ggplot2)
library(doBy)
#install.packages("useful")
library(useful)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
#install.packages('rpart')
library(rpart)
#install.packages("randomForest")
library("randomForest")
```

```
## randomForest 4.6-14
```

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

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

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

```
#install.packages("PlotROC")
library(plotROC)
set.seed(1861)
```

```r
#reads in data into KickStar
KickStar <- read.csv("/Users/karlhickel/Desktop//ks-projects-201612.csv")
#KickStar <- read.csv("C:/Users/ericv/Documents/Github/StatisticalModelsProject/ks-projects-201612.csv"
#KickStar <- read.csv("/Users/Cheddar3/Desktop/MGSC310/Group Projects/kickstarter-projects/ks-projects-

#removing a few columns with null values
KickStar <- subset(KickStar, select = -c(14,15,16,17))
#colSums(is.na(KickStar))
KickStar<- KickStar[!is.na(KickStar$name),]
#colSums(is.na(KickStar))
#cleaning up category column
KickStar$category <- factor(KickStar$category, levels = c("Zines","Young Adult", "World Music", "Worksh
#colSums(is.na(KickStar))
KickStar<- KickStar[!is.na(KickStar$category),]
#colSums(is.na(KickStar))
#cleaning main_category column
#table(KickStar$main_category)
KickStar$main_category <- factor(KickStar$main_category, levels = c("Theater", "Technology", "Publishing
#colSums(is.na(KickStar))
#clearning currency column
#table(KickStar$currency)
KickStar$currency <- factor(KickStar$currency, levels = c("USD", "SGD", "SEK", "NZD", "NOK", "MXN", "HK
#colSums(is.na(KickStar))
#cleaning goal column
KickStar$goal <- as.numeric(KickStar$goal)
#colSums(is.na(KickStar))
#cleaning pledge column
KickStar$pledged <- as.numeric(KickStar$pledged)
#cleaning state column
#table(KickStar$state)
KickStar$state <- factor(KickStar$state, levels = c("failed", "successful"))
#colSums(is.na(KickStar))
KickStar<- KickStar[!is.na(KickStar$state),]
#colSums(is.na(KickStar))
#cleaning backers column
KickStar$backers <- as.numeric(KickStar$backers)
#colSums(is.na(KickStar))
#cleaning country column
#table(KickStar$country)
KickStar$country <- factor(KickStar$country, levels = c("US", "SG", "SE", "NZ", "NO", "NL", "MX", "LU",
#colSums(is.na(KickStar))
KickStar<- KickStar[!is.na(KickStar$country),]
#colSums(is.na(KickStar))
#cleaning usd.pledged column
KickStar$usd.pledged <- as.numeric(KickStar$usd.pledged)
#colSums(is.na(KickStar))
#creates deadline and launched variable so its just year/month/day
KickStar$newDeadline <- as.Date(as.POSIXlt(KickStar$deadline, origin = "1582-10-14",tz = "GMT"))
KickStar$newLaunched <- as.Date(as.POSIXlt(KickStar$launched, origin = "1582-10-14",tz = "GMT"))
#difference between launched and deadline in days
#launch kickstarter before meeting the deadline for funds
KickStar$DateDiffDays <- KickStar$newDeadline - KickStar$newLaunched
#if successful, then 1, else 0
```

```
KickStar$binomState <- ifelse(KickStar$state == "successful",1,0)
#Added statistically significant frequency categories
KickStar$SSFCategories <- ifelse(KickStar$main_category == "Film & Video"
                                 | KickStar$main_category == "Publishing"
                                 | KickStar$main_category == "Games"
                                 | KickStar$main_category == "Design"
                                 | KickStar$main_category == "Fashion"
                                 ,1,0)
#added statistically significant currencies
KickStar$SScurrency <- ifelse(KickStar$currency == "NOK"
                                 | KickStar$currency == "GBP"
                                 | KickStar$currency == "EUR"
                                 | KickStar$currency == "CHF"
                                 | KickStar$currency == "CAD"
                                 | KickStar$currency == "AUD"
                                 ,1,0)
#added average pledged
KickStar$AveragePledged <- KickStar$pledged / KickStar$backers
```

Establish Training and Test Set

```
trainSize <- 0.75
trainInd <- sample(1:nrow(KickStar), size = floor(nrow(KickStar) * trainSize))
KickTrain <- KickStar[trainInd, ]
KickTest <- KickStar[-trainInd, ]
#gets summary of data
summary(KickStar)
```

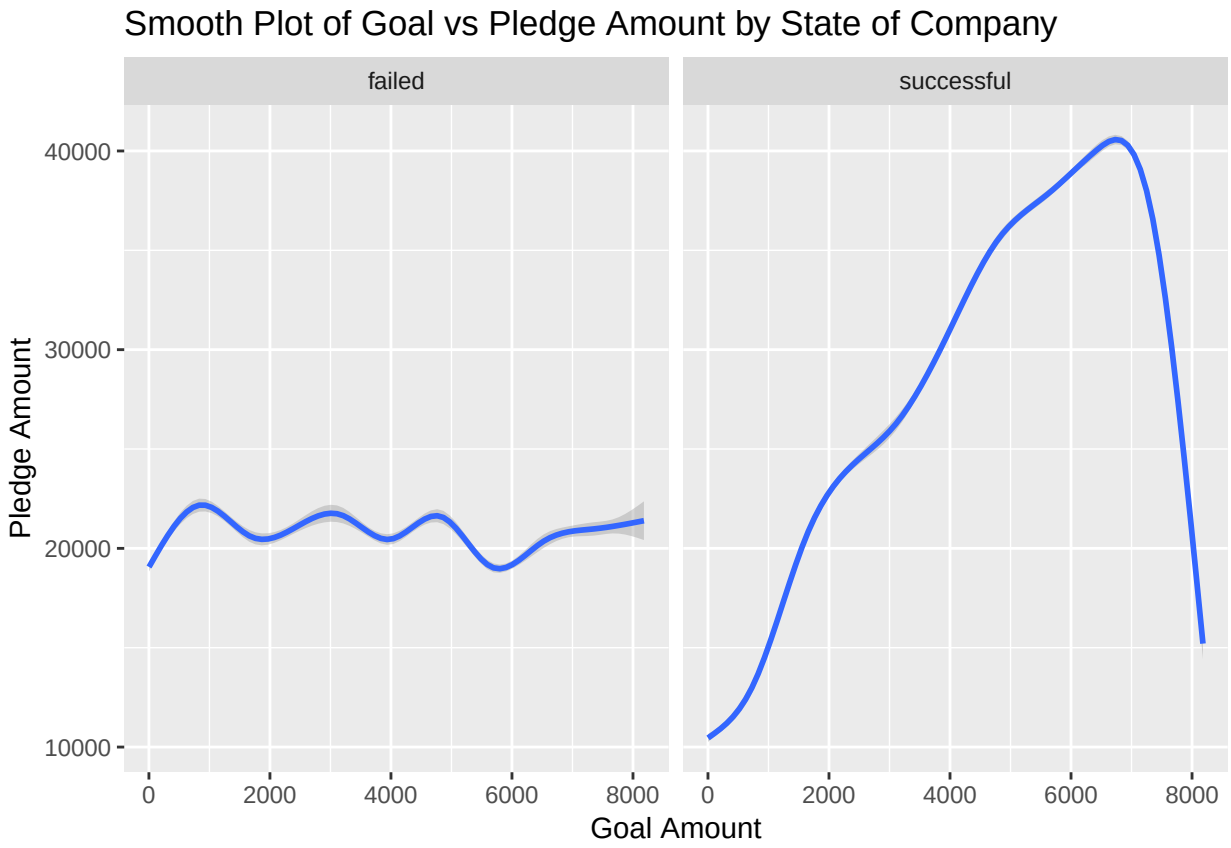
```
##        ID                                     name
##  Min.   :5.971e+03   New EP/Music Development  :    13
##  1st Qu.:5.373e+08   New EP / Music Development:    10
##  Median :1.076e+09   Music Video               :     9
##  Mean   :1.075e+09   Reflections               :     9
##  3rd Qu.:1.611e+09   A Midsummer Night's Dream :     8
##  Max.   :2.147e+09   Pizza                     :     8
##                      (Other)                   :281034
##            category          main_category      currency
##  Product Design: 14515   Film & Video:51034   USD    :229365
##  Documentary   : 13346   Music       :40808   GBP    : 23820
##  Shorts        : 10775   Publishing  :30143   CAD    :  9899
##  Music         : 10724   Games       :22398   EUR    :  9156
##  Food          :  9517   Art         :21745   AUD    :  5183
##  Tabletop Games:  8819   Technology  :21405   SEK    :  1065
##  (Other)       :213395   (Other)     :93558   (Other):  2603
##                 deadline           goal                    launched
##  2012-01-01 05:59:00:    46   Min.   :   1   2009-09-15 05:56:28:     2
##  2014-11-01 04:59:00:    34   1st Qu.:1257   2010-06-30 17:29:43:     2
##  2015-01-01 05:59:00:    34   Median :4145   2011-02-08 04:29:48:     2
##  2012-03-01 05:59:00:    32   Mean   :3693   2011-02-25 09:58:36:     2
##  2010-08-01 05:59:00:    28   3rd Qu.:5820   2011-03-07 17:11:18:     2
##  2010-09-01 05:59:00:    28   Max.   :8181   2011-06-07 06:42:01:     2
```

```
##  (Other)            :280889                  (Other)            :281079
##     pledged             state           backers        country
##  Min.   :    1   failed    :168115   Min.   :   1   US     :229365
##  1st Qu.: 5597   successful:112976   1st Qu.: 156   GB     : 23820
##  Median :23388                       Median :1395   CA     :  9899
##  Mean   :23310                       Mean   :1465   AU     :  5183
##  3rd Qu.:39532                       3rd Qu.:2537   DE     :  2130
##  Max.   :55597                       Max.   :3587   NL     :  1869
##                                                     (Other):  8825
##   usd.pledged     newDeadline          newLaunched
##  Min.   :    2   Min.   :2009-05-03   Min.   :2009-04-21
##  1st Qu.: 9838   1st Qu.:2012-12-26   1st Qu.:2012-11-22
##  Median :39322   Median :2014-08-11   Median :2014-07-11
##  Mean   :39382   Mean   :2014-04-13   Mean   :2014-03-10
##  3rd Qu.:66703   3rd Qu.:2015-08-07   3rd Qu.:2015-07-05
##  Max.   :94361   Max.   :2016-12-06   Max.   :2016-12-01
##
##  DateDiffDays        binomState     SSFCategories      SScurrency
##  Length:281091     Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  Class :difftime   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Mode  :numeric    Median :0.0000   Median :0.0000   Median :0.0000
##                    Mean   :0.4019   Mean   :0.4959   Mean   :0.1739
##                    3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
##                    Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##
##  AveragePledged
##  Min.   :    0.009
##  1st Qu.:    3.913
##  Median :   12.477
##  Mean   :  860.198
##  3rd Qu.:   27.400
##  Max.   :27794.500
##
```

Relevant ggplots

```
#ggplot of Goal Amount and Pledge Amount by state of company
ggplot(data = KickStar, aes(goal, pledged)) + geom_smooth() + facet_wrap(~state) + labs(title = "Smooth
```

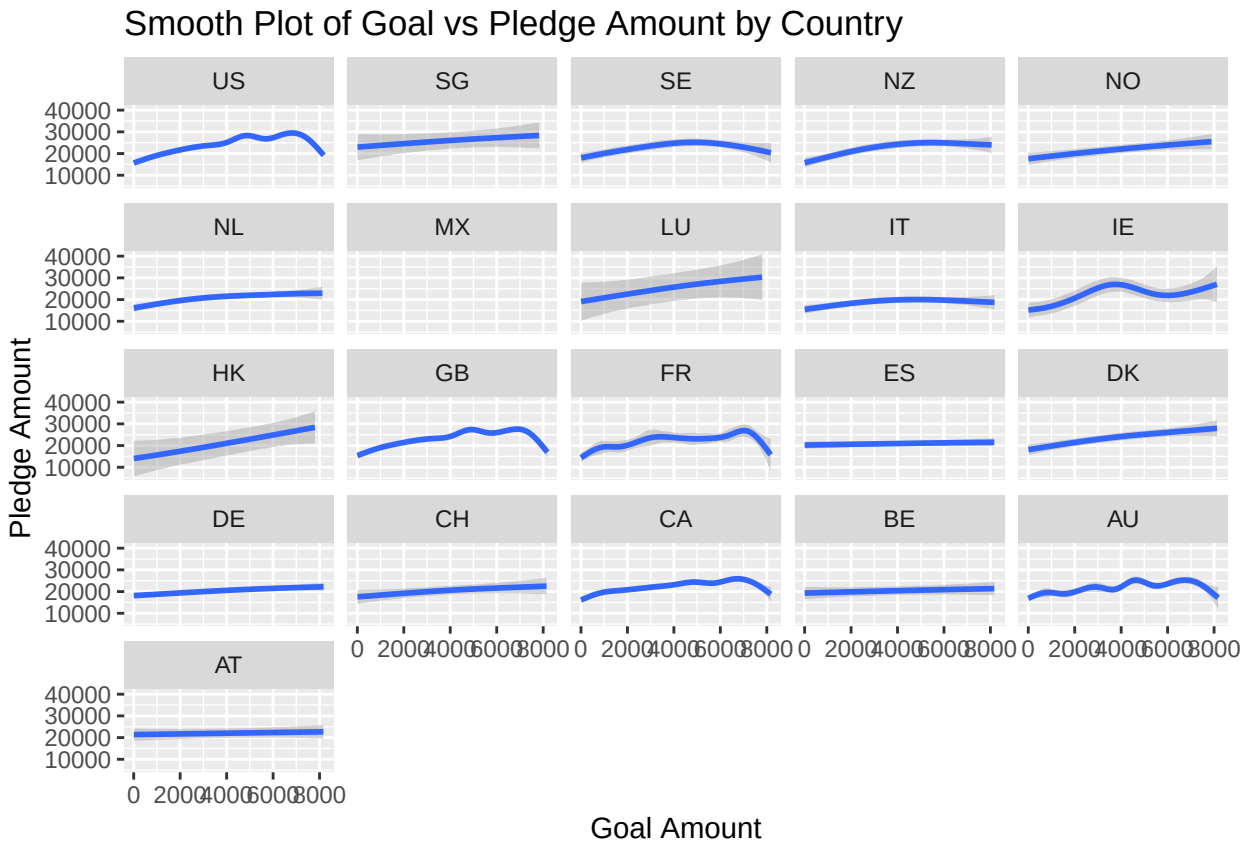
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



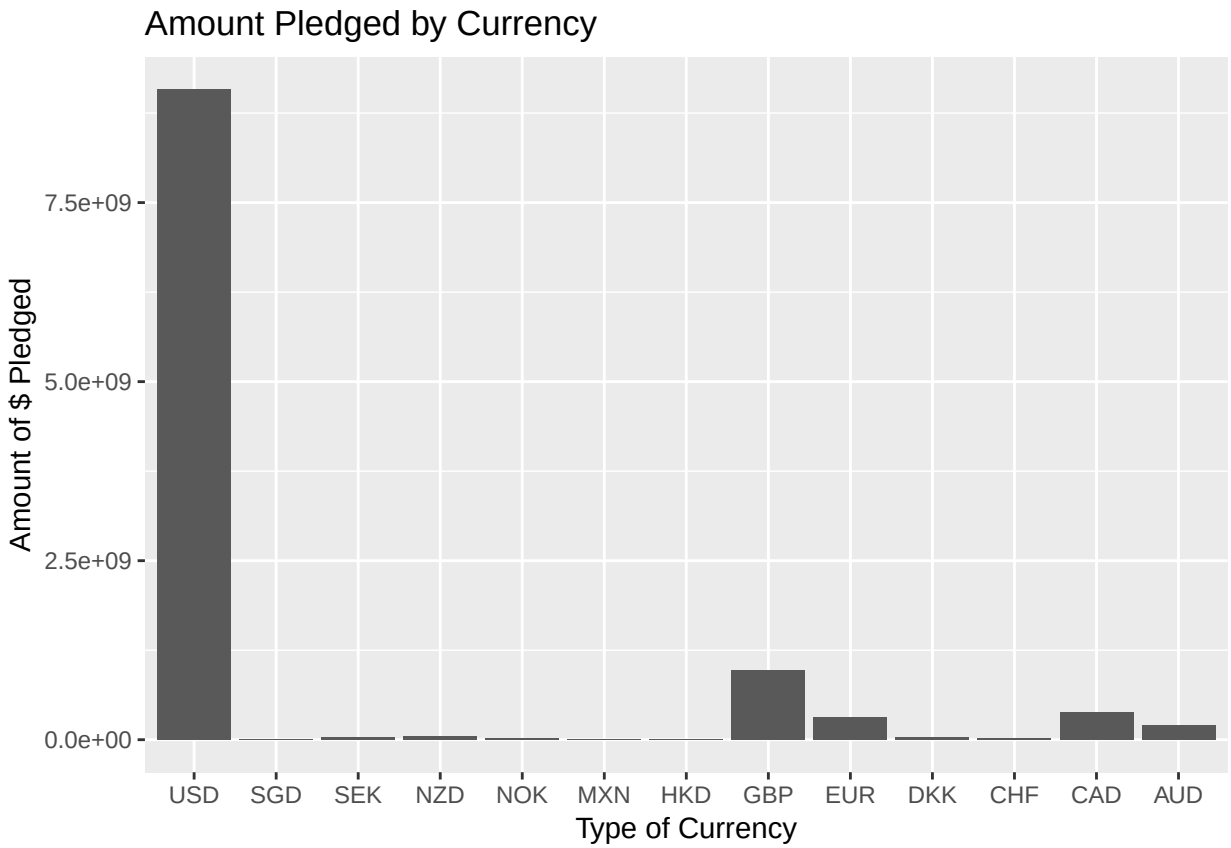

```
#ggplot of Goal Amount and Pledge Aamount by country
ggplot(data = KickStar, aes(goal, pledged)) + geom_smooth() + facet_wrap(~country) + labs(title = "Smoo
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

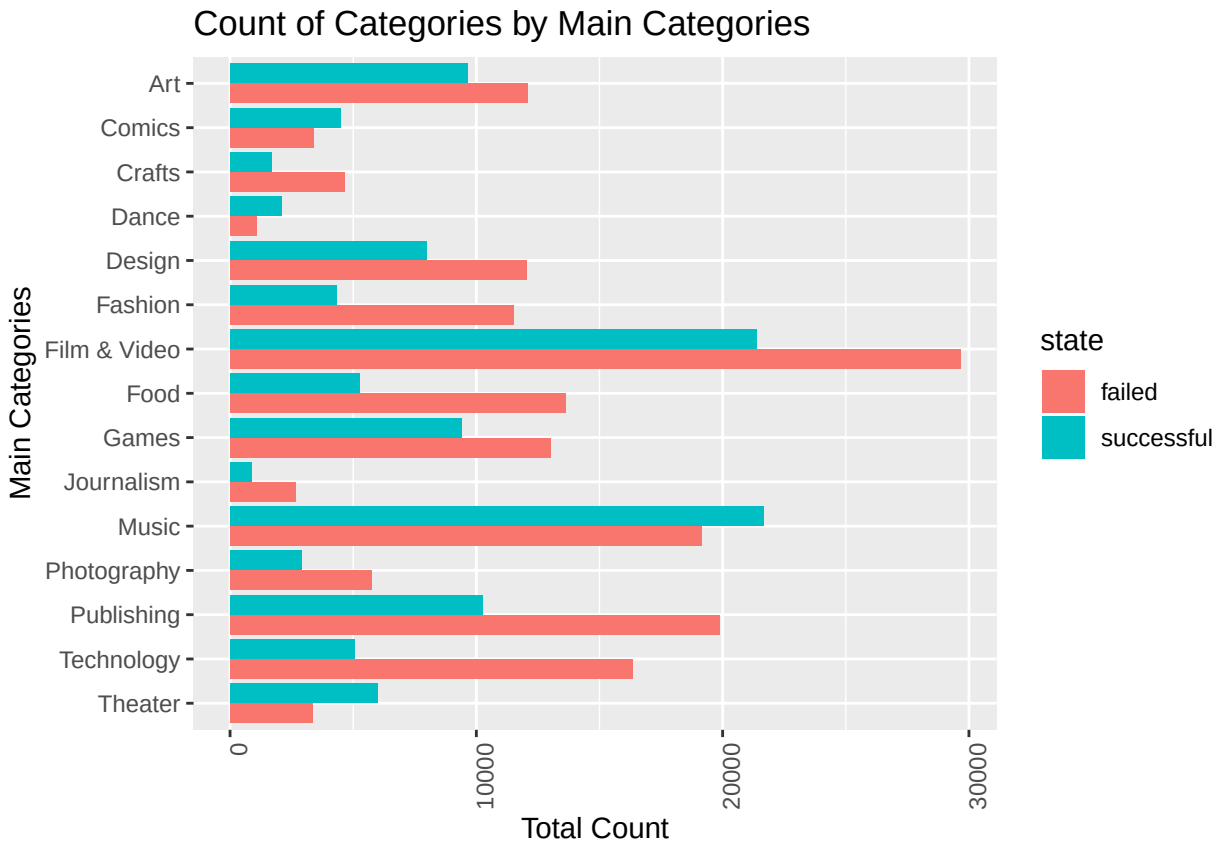
```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```



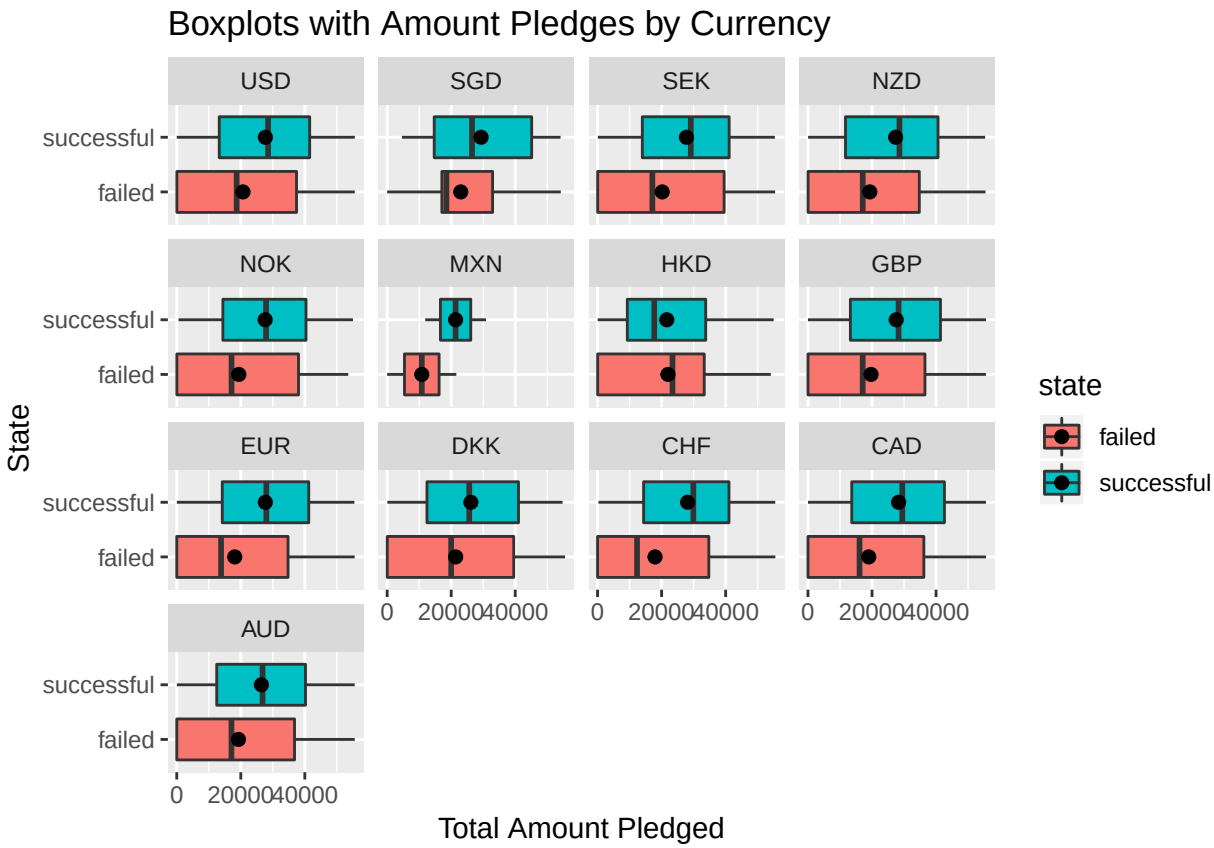

```
#ggplot of bar graph with type of currency and the amount pledged
ggplot(data = KickStar, aes(x = currency, y = usd.pledged)) + geom_bar(stat = "identity") + labs(x = "Ty
```




```
###Graphs to use for powerpoint
##ggplot of bar graph of count of each categories
ggplot(data = KickStar, aes(main_category, fill = state)) + geom_bar(position = "dodge") + theme(axis.t
```




```
##ggplot of bar
ggplot(data = KickStar, aes(x = state, y = pledged, fill = state)) + geom_boxplot() + facet_wrap(~curre
```



Logistic Regression Models

Using the training data to create a model

Find summary statistics when the Kickstarter succeed (1) or did not (0)

```
summaryBy(goal + pledged + backers + currency + DateDiffDays + SScurrency + SSFCategories + AveragePledg
```

```
##   binomState goal.FUN1 pledged.FUN1 backers.FUN1 currency.FUN1
## 1          0  3686.835     20404.68     1196.640      2.714879
## 2          1  3703.354     27685.02     1867.916      2.236159
##   DateDiffDays.FUN1 SScurrency.FUN1 SSFCategories.FUN1 AveragePledged.FUN1
## 1          35.40132       0.1933184          0.5121880           1331.9377
## 2          32.50211       0.1443513          0.4707118            161.9074
```

```
summaryBy(cbind(pledged, backers, DateDiffDays) ~ cbind(binomState,main_category), data = KickTrain)
```

```
##    binomState main_category pledged.mean backers.mean DateDiffDays.mean
## 1           0       Theater     21301.10    1278.9921          36.85975
## 2           0    Technology     19786.99    1208.0694          35.69625
## 3           0    Publishing     19603.52    1101.4016          34.75090
## 4           0   Photography     19410.43    1079.2167          34.53392
## 5           0         Music     19139.33    1080.6714          37.08385
## 6           0    Journalism     16799.25     923.1908          35.16990
## 7           0         Games     23653.89    1483.0382          34.23548
```

```
## 8           0          Food     20177.59    1182.7045          34.85828
## 9           0  Film & Video     19920.35    1171.7924          37.19562
## 10          0       Fashion     18960.08    1066.2426          33.34033
## 11          0        Design     24706.63    1519.1196          34.72235
## 12          0         Dance     21354.68    1296.1973          34.53837
## 13          0        Crafts     19041.64    1047.0944          32.40040
## 14          0        Comics     24747.79    1568.3600          37.34822
## 15          0           Art     19777.08    1124.9704          34.26804
## 16          1       Theater     27637.14    1948.2709          31.78153
## 17          1    Technology     27526.69    1834.9115          34.20544
## 18          1    Publishing     28719.30    1866.5063          32.21177
## 19          1   Photography     27618.23    1839.9835          32.28663
## 20          1         Music     27430.26    1921.0707          34.11977
## 21          1    Journalism     27613.38    1884.1238          32.23824
## 22          1         Games     28020.01    1826.6699          30.73135
## 23          1          Food     26236.91    1760.6612          31.76152
## 24          1  Film & Video     27653.35    1847.8421          32.61243
## 25          1       Fashion     26496.65    1851.4292          31.47640
## 26          1        Design     27655.02    1842.0606          33.75290
## 27          1         Dance     28487.16    1977.9266          31.84946
## 28          1        Crafts     28077.67    1788.6002          29.35771
## 29          1        Comics     28479.24    1851.9060          32.50250
## 30          1           Art     27762.74    1892.4461          30.81564
```

Creating a glm model on the training data and summary

```
glmmodTrain <- glm(binomState ~ goal + pledged + backers + DateDiffDays + SScurrency + SSFCategories + A
                   data = KickTrain, family = binomial)
summary(glmmodTrain)
```

```
##
## Call:
## glm(formula = binomState ~ goal + pledged + backers + DateDiffDays +
##     SScurrency + SSFCategories + AveragePledged, family = binomial,
##     data = KickTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9390  -1.0102  -0.6055   1.1334   2.8635
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -4.078e-01  1.800e-02  -22.66   <2e-16 ***
## goal           -3.696e-05  2.015e-06  -18.34   <2e-16 ***
## pledged         2.145e-05  2.944e-07   72.86   <2e-16 ***
## backers         3.520e-04  4.508e-06   78.08   <2e-16 ***
## DateDiffDays   -2.028e-02  3.918e-04  -51.76   <2e-16 ***
## SScurrency     -3.242e-01  1.290e-02  -25.14   <2e-16 ***
## SSFCategories  -1.992e-01  9.516e-03  -20.93   <2e-16 ***
## AveragePledged -1.358e-04  2.782e-06  -48.83   <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: 284073  on 210817  degrees of freedom
## Residual deviance: 255260  on 210810  degrees of freedom
## AIC: 255276
##
## Number of Fisher Scoring iterations: 5
```

Find odds ratios

```
exp(glmmodTrain$coefficients)
```

```
##    (Intercept)           goal        pledged        backers   DateDiffDays
##      0.6651083      0.9999630      1.0000214      1.0003520      0.9799240
##     SScurrency  SSFCategories AveragePledged
##      0.7231401      0.8193923      0.9998642
```

Creating probabilty of failure (1) or not (0)

```
glmTrainProbs <- predict.glm(glmmodTrain, type = "response")
hist(glmTrainProbs)
```


```
table(KickStar$binomState)
```

```
##
##      0      1
## 168115 112976
```

```
glmTrainPreds <- rep("Failed", nrow(KickTrain))
glmTrainPreds[glmTrainProbs > 0.5] <- "Succeed"
conftabtrain <- table(glmTrainPreds, true = KickTrain$binomState)
print(conftabtrain)
```

```
##              true
## glmTrainPreds     0     1
##       Failed  95351 46924
##       Succeed 30757 37786
```

Using the TESTING data to create a model

Find summary statistics when the Kickstarter failed (1) or did not fail (0)

```
summaryBy(goal + pledged + backers + DateDiffDays + SScurrency + SSFCategories + AveragePledged ~ binomS
```

```
##   binomState goal.FUN1 pledged.FUN1 backers.FUN1 DateDiffDays.FUN1
## 1          0  3694.141     20310.86     1190.185          35.29374
## 2          1  3690.896     27616.60     1867.456          32.34069
##   SScurrency.FUN1 SSFCategories.FUN1 AveragePledged.FUN1
## 1       0.1960626          0.5115576           1315.8813
## 2       0.1427156          0.4750230            171.0363
```

It appears that Kickstarters that had less funding, less pledges, less currency, and less backers failed.

Creating a glm model on the training data and summary

```
glmmodTest <- glm(binomState ~ goal + pledged + backers + DateDiffDays + SScurrency + SSFCategories + Av
summary(glmmodTest)
```

```
##
## Call:
## glm(formula = binomState ~ goal + pledged + backers + DateDiffDays +
##     SScurrency + SSFCategories + AveragePledged, family = binomial,
##     data = KickTest)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9266  -1.0056  -0.5979   1.1310   2.9120
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -3.791e-01  3.116e-02  -12.17   <2e-16 ***
## goal           -4.202e-05  3.498e-06  -12.01   <2e-16 ***
## pledged         2.144e-05  5.101e-07   42.03   <2e-16 ***
## backers         3.614e-04  7.830e-06   46.16   <2e-16 ***
## DateDiffDays   -2.106e-02  6.807e-04  -30.93   <2e-16 ***
## SScurrency     -3.575e-01  2.238e-02  -15.97   <2e-16 ***
## SSFCategories  -1.837e-01  1.650e-02  -11.14   <2e-16 ***
## AveragePledged -1.305e-04  4.687e-06  -27.84   <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: 94715  on 70272  degrees of freedom
## Residual deviance: 84980  on 70265  degrees of freedom
## AIC: 84996
##
## Number of Fisher Scoring iterations: 5
```

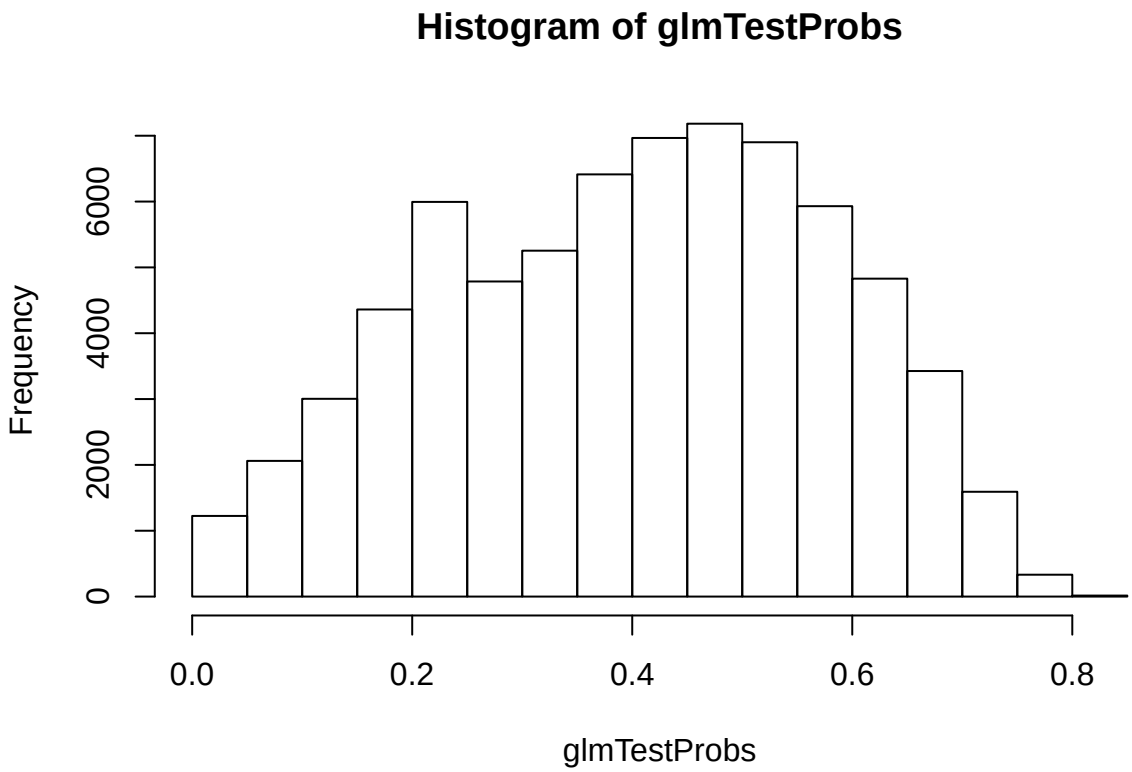
Find odds ratios

```
exp(glmmodTest$coefficients)
```

```
##    (Intercept)           goal        pledged        backers   DateDiffDays
##      0.6844665      0.9999580      1.0000214      1.0003614      0.9791624
##     SScurrency  SSFCategories AveragePledged
##      0.6994044      0.8321849      0.9998695
```

Creating probabilty of failure (1) or not (0)

```
glmTestProbs <- predict.glm(glmmodTest, type = "response")
hist(glmTestProbs)
```




```
glmTestPreds <- rep("Failed", nrow(KickTest))
glmTestPreds[glmTestProbs > 0.5] <- "Succeed"
conftabtest <- table(glmTestPreds, true = KickTest$binomState)
print(conftabtest)
```

```
##             true
## glmTestPreds     0     1
##      Failed  31694 15554
##      Succeed 10313 12712
```

Validating Results

```
TPtrain <- conftabtrain[2, 2]/(conftabtrain[1, 2] + conftabtrain[2, 2])
TNtrain <- conftabtrain[1, 1]/(conftabtrain[1, 1] + conftabtrain[2, 1])
FPtrain <- conftabtrain[2, 1]/(conftabtrain[2, 1] + conftabtrain[1, 1])
FNtrain <- conftabtrain[1, 2]/(conftabtrain[1, 1] + conftabtrain[2, 1])

TPtest <- conftabtest[2, 2]/(conftabtest[1, 2] + conftabtest[2, 2])
TNtest <- conftabtest[1, 1]/(conftabtest[1, 1] + conftabtest[2, 1])
FPtest <- conftabtest[2, 1]/(conftabtest[2, 1] + conftabtest[1, 1])
FNtest <- conftabtest[1, 2]/(conftabtest[1, 1] + conftabtest[2, 1])
#TRAINING RESULTS
print("Training True Positives")
```

```
## [1] "Training True Positives"
```

```
print(TPtrain)
```

```
## [1] 0.446063
```

```
print("Training True Negatives")
```

```
## [1] "Training True Negatives"
```

```
print(TNtrain)
```

```
## [1] 0.7561059
```

```
print("Training False Positives")
```

```
## [1] "Training False Positives"
```

```
print(FPtrain)
```

```
## [1] 0.2438941
```

```
print("Training False Negatives")
```

```
## [1] "Training False Negatives"
```

```
print(FNtrain)
```

```
## [1] 0.3720938
```

```
print("--------")
```

```
## [1] "--------"
```

```
#TESTING RESULTS
print("Testing True Positives")
```

```
## [1] "Testing True Positives"
```

```
print(TPtest)
```

```
## [1] 0.4497276
```

```
print("Testing True Negatives")
```

```
## [1] "Testing True Negatives"
```

```
print(TNtest)
```

```
## [1] 0.7544933
```

```
print("Testing False Positives")
```

```
## [1] "Testing False Positives"
```

```
print(FPtest)
```

```
## [1] 0.2455067
```

```
print("Testing False Negatives")
```

```
## [1] "Testing False Negatives"
```

```
print(FNtest)
```

```
## [1] 0.3702716
```

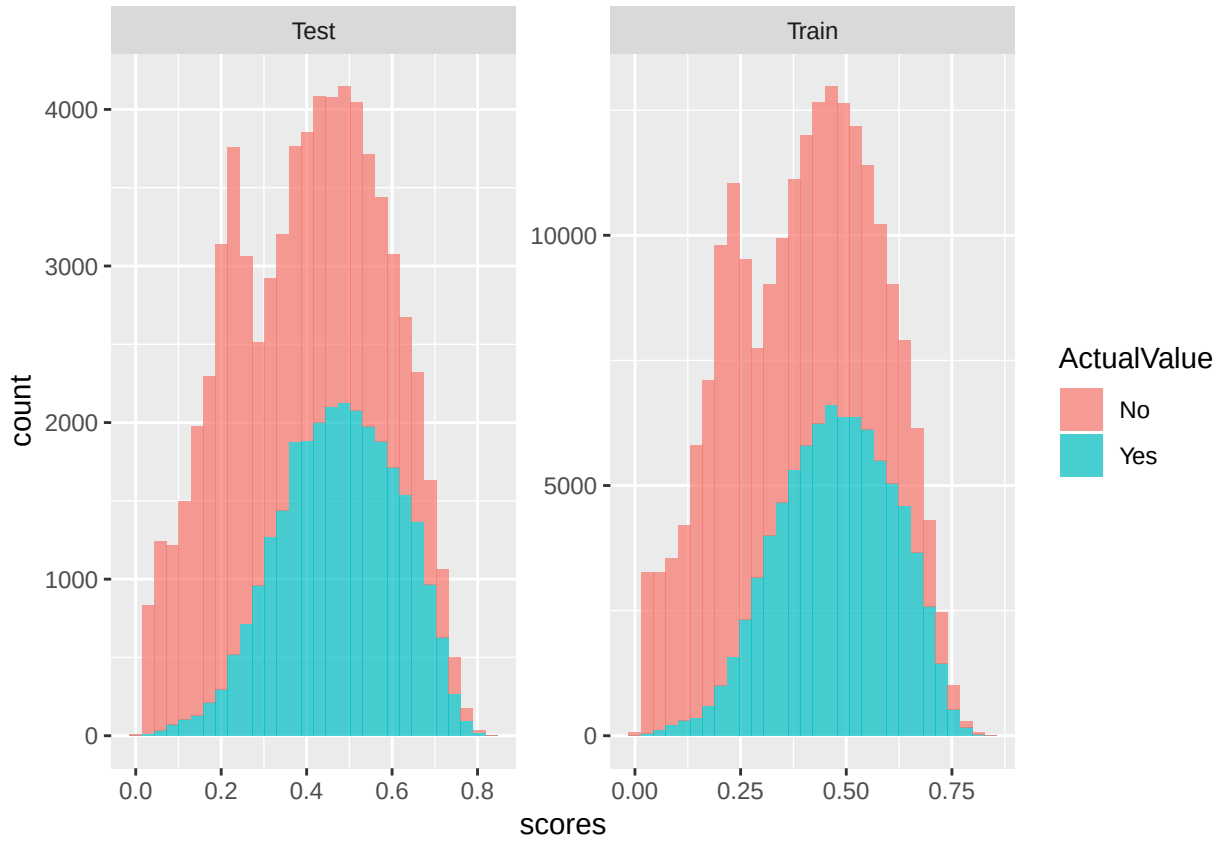
ROC Curve

```
scoresTest <- predict(glmmodTest, newdata = KickTest, type = "response")
scoresTrain <- predict(glmmodTrain, newdata = KickTrain, type = "response")
scoresDF <- data.frame(scores = c(scoresTest, scoresTrain), type = c(rep("Test",
times = length(scoresTest)), rep("Train", times = length(scoresTrain))),
true = c(KickTest$binomState, KickTrain$binomState))
```

Testing and Training Scores

```
scoresDF$ActualValue <- ifelse(scoresDF$true == 1, "Yes", "No")
ggplot(data = scoresDF, aes(x = scores, fill = ActualValue)) + geom_histogram(alpha = .7) + facet_grid(
```

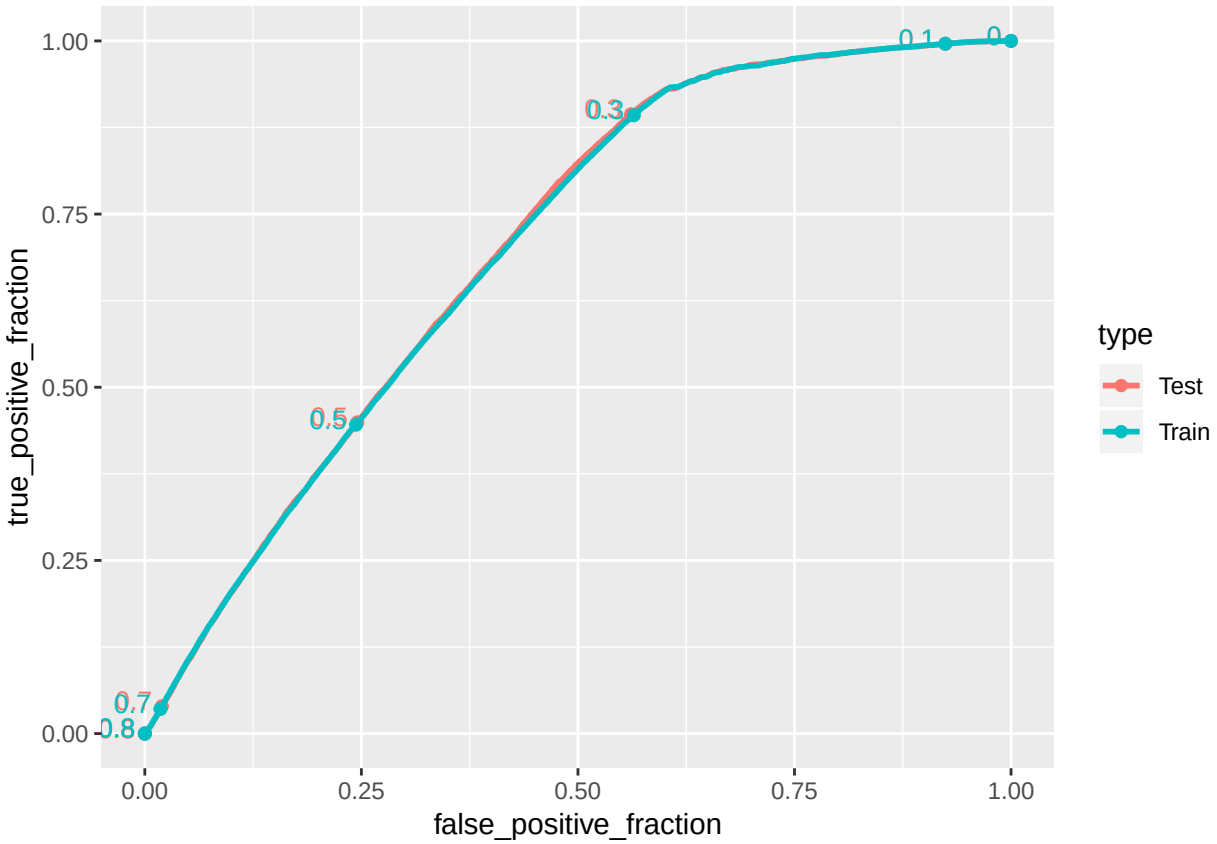
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
doBy::summaryBy(scores ~ type, data = scoresDF)
```

```
##    type scores.mean
## 1  Test   0.4022313
## 2 Train   0.4018158
```

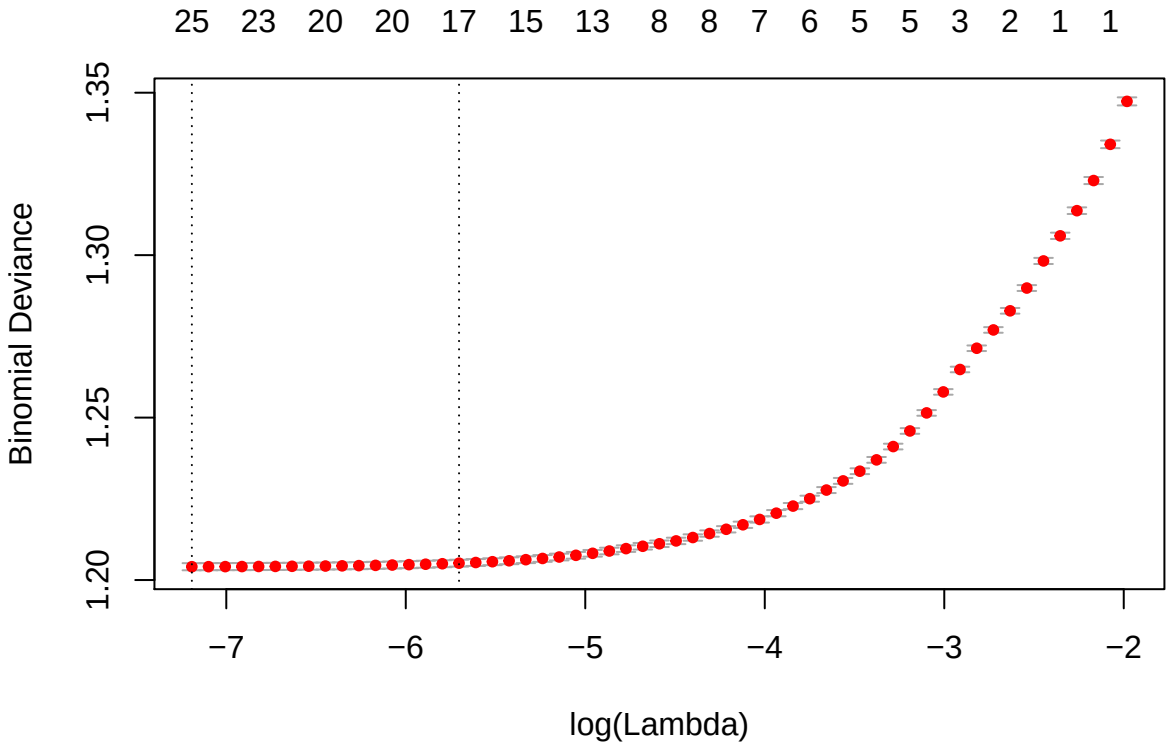
```
ggplot(scoresDF, aes(m = scores, d = true, color = type)) + geom_roc(show.legend = TRUE,
labelsize = 3.5, cutoffs.at = c(0.99, 0.9, 0.7, 0.5, 0.3, 0.1, 0))
```



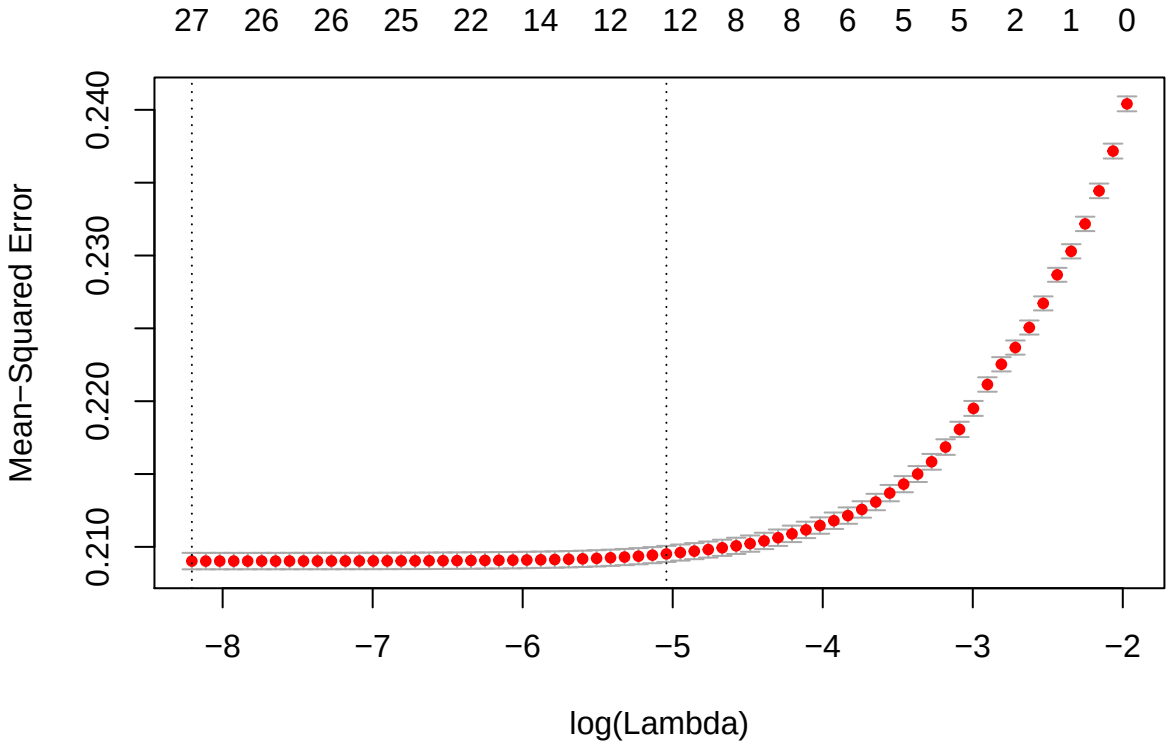
Lasso, Ridge, Elastic Net Models

```
myFormula <- as.formula(binomState ~ goal + pledged + backers + country + usd.pledged + DateDiffDays + S
Xvar <- build.x(myFormula, KickTrain)
Yvar <- build.y(myFormula, KickTrain)
XvarTest <- build.x(myFormula, KickTest)
YvarTest <- build.y(myFormula, KickTest)
LassoMod <- cv.glmnet(x = Xvar, y = Yvar, alpha = 1, nfolds = 10, family = "binomial")
LassoModTest <- cv.glmnet(x = XvarTest, y = YvarTest)

plot(LassoMod)
```




```
plot(LassoModTest)
```




```
LassoMod
```

```
## $lambda
##  [1] 0.1378074883 0.1255650509 0.1144101978 0.1042463112 0.0949853562
##  [6] 0.0865471190 0.0788585115 0.0718529387 0.0654697216 0.0596535720
```

```
## [11] 0.0543541131 0.0495254435 0.0451257396 0.0411168933 0.0374641818
## [16] 0.0341359671 0.0311034219 0.0283402796 0.0258226073 0.0235285980
## [21] 0.0214383822 0.0195338554 0.0177985216 0.0162173500 0.0147766453
## [26] 0.0134639289 0.0122678305 0.0111779903 0.0101849684 0.0092801639
## [31] 0.0084557397 0.0077045551 0.0070201037 0.0063964570 0.0058282134
## [36] 0.0053104510 0.0048386852 0.0044088297 0.0040171614 0.0036602879
## [41] 0.0033351180 0.0030388353 0.0027688735 0.0025228945 0.0022987675
## [46] 0.0020945513 0.0019084771 0.0017389333 0.0015844512 0.0014436930
## [51] 0.0013154393 0.0011985793 0.0010921008 0.0009950816 0.0009066813
## [56] 0.0008261342 0.0007527427
##
## $cvm
##  [1] 1.347328 1.334103 1.322961 1.313688 1.305946 1.298216 1.289876
##  [8] 1.282865 1.276977 1.271344 1.264808 1.257905 1.251439 1.245886
## [15] 1.241082 1.236987 1.233497 1.230515 1.227695 1.225038 1.222766
## [22] 1.220576 1.218647 1.217009 1.215620 1.214311 1.213083 1.212045
## [29] 1.211167 1.210397 1.209672 1.208931 1.208224 1.207609 1.207084
## [36] 1.206642 1.206261 1.205931 1.205642 1.205398 1.205193 1.205020
## [43] 1.204869 1.204734 1.204618 1.204519 1.204436 1.204368 1.204311
## [50] 1.204265 1.204227 1.204194 1.204168 1.204146 1.204128 1.204114
## [57] 1.204102
##
## $cvsd
##  [1] 0.0012373807 0.0011671507 0.0010858220 0.0010203888 0.0009737731
##  [6] 0.0009364055 0.0009010906 0.0008755242 0.0008603351 0.0008550946
## [11] 0.0008576766 0.0008612294 0.0008643797 0.0008684941 0.0008767050
## [16] 0.0008859912 0.0008958821 0.0009060763 0.0009156484 0.0009300144
## [21] 0.0009439401 0.0009602024 0.0009638268 0.0009679311 0.0009721002
## [26] 0.0009771285 0.0009850499 0.0009934400 0.0010017029 0.0010110873
## [31] 0.0010177978 0.0010206944 0.0010261508 0.0010332472 0.0010390734
## [36] 0.0010441999 0.0010498393 0.0010561890 0.0010596165 0.0010631043
## [41] 0.0010663844 0.0010699186 0.0010737389 0.0010762365 0.0010787980
## [46] 0.0010814531 0.0010837087 0.0010856517 0.0010874028 0.0010887406
## [51] 0.0010896863 0.0010905838 0.0010914649 0.0010921419 0.0010925842
## [56] 0.0010930031 0.0010933822
##
## $cvup
##  [1] 1.348566 1.335270 1.324046 1.314708 1.306920 1.299152 1.290777
##  [8] 1.283740 1.277837 1.272199 1.265665 1.258767 1.252303 1.246754
## [15] 1.241959 1.237873 1.234393 1.231421 1.228611 1.225968 1.223710
## [22] 1.221536 1.219611 1.217977 1.216592 1.215288 1.214068 1.213038
## [29] 1.212169 1.211408 1.210690 1.209952 1.209250 1.208643 1.208123
## [36] 1.207686 1.207311 1.206987 1.206702 1.206461 1.206259 1.206090
## [43] 1.205943 1.205810 1.205697 1.205600 1.205520 1.205453 1.205399
## [50] 1.205354 1.205316 1.205285 1.205259 1.205238 1.205221 1.205207
## [57] 1.205196
##
## $cvlo
##  [1] 1.346091 1.332936 1.321875 1.312667 1.304973 1.297279 1.288975
##  [8] 1.281989 1.276116 1.270489 1.263950 1.257044 1.250575 1.245017
## [15] 1.240205 1.236101 1.232601 1.229609 1.226779 1.224108 1.221822
## [22] 1.219616 1.217683 1.216041 1.214648 1.213334 1.212098 1.211051
## [29] 1.210165 1.209386 1.208655 1.207911 1.207198 1.206576 1.206045
## [36] 1.205598 1.205211 1.204875 1.204583 1.204335 1.204127 1.203950
```

```
## [43] 1.203795 1.203658 1.203539 1.203437 1.203352 1.203282 1.203224
## [50] 1.203176 1.203137 1.203104 1.203076 1.203054 1.203036 1.203021
## [57] 1.203009
##
## $nzero
##  s0  s1  s2  s3  s4  s5  s6  s7  s8  s9 s10 s11 s12 s13 s14 s15 s16 s17
##   0   1   1   1   1   2   2   2   2   3   3   4   4   5   5   5   5   5
## s18 s19 s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35
##   6   6   6   7   7   7   7   8   8   8   8   9  11  12  13  14  14  14
## s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53
##  15  16  16  16  17  18  19  20  20  20  20  20  20  20  21  22  23  23
## s54 s55 s56
##  23  23  25
##
## $name
##            deviance
## "Binomial Deviance"
##
## $glmnet.fit
##
## Call:  glmnet(x = Xvar, y = Yvar, alpha = 1, family = "binomial")
##
##       Df      %Dev    Lambda
##  [1,]  0 6.509e-14 0.1378000
##  [2,]  1 9.952e-03 0.1256000
##  [3,]  1 1.822e-02 0.1144000
##  [4,]  1 2.510e-02 0.1042000
##  [5,]  1 3.083e-02 0.0949900
##  [6,]  2 3.657e-02 0.0865500
##  [7,]  2 4.278e-02 0.0788600
##  [8,]  2 4.798e-02 0.0718500
##  [9,]  2 5.234e-02 0.0654700
## [10,]  3 5.651e-02 0.0596500
## [11,]  3 6.137e-02 0.0543500
## [12,]  4 6.651e-02 0.0495300
## [13,]  4 7.131e-02 0.0451300
## [14,]  5 7.544e-02 0.0411200
## [15,]  5 7.900e-02 0.0374600
## [16,]  5 8.204e-02 0.0341400
## [17,]  5 8.463e-02 0.0311000
## [18,]  5 8.684e-02 0.0283400
## [19,]  6 8.894e-02 0.0258200
## [20,]  6 9.092e-02 0.0235300
## [21,]  6 9.260e-02 0.0214400
## [22,]  7 9.424e-02 0.0195300
## [23,]  7 9.567e-02 0.0178000
## [24,]  7 9.689e-02 0.0162200
## [25,]  7 9.792e-02 0.0147800
## [26,]  8 9.890e-02 0.0134600
## [27,]  8 9.981e-02 0.0122700
## [28,]  8 1.006e-01 0.0111800
## [29,]  8 1.012e-01 0.0101800
## [30,]  9 1.018e-01 0.0092800
## [31,] 11 1.024e-01 0.0084560
```

```
## [32,] 12 1.029e-01 0.0077050
## [33,] 13 1.035e-01 0.0070200
## [34,] 14 1.039e-01 0.0063960
## [35,] 14 1.043e-01 0.0058280
## [36,] 14 1.046e-01 0.0053100
## [37,] 15 1.049e-01 0.0048390
## [38,] 16 1.052e-01 0.0044090
## [39,] 16 1.054e-01 0.0040170
## [40,] 16 1.056e-01 0.0036600
## [41,] 17 1.057e-01 0.0033350
## [42,] 18 1.059e-01 0.0030390
## [43,] 19 1.060e-01 0.0027690
## [44,] 20 1.061e-01 0.0025230
## [45,] 20 1.062e-01 0.0022990
## [46,] 20 1.063e-01 0.0020950
## [47,] 20 1.063e-01 0.0019080
## [48,] 20 1.064e-01 0.0017390
## [49,] 20 1.064e-01 0.0015840
## [50,] 20 1.065e-01 0.0014440
## [51,] 21 1.065e-01 0.0013150
## [52,] 22 1.065e-01 0.0011990
## [53,] 23 1.065e-01 0.0010920
## [54,] 23 1.066e-01 0.0009951
## [55,] 23 1.066e-01 0.0009067
## [56,] 23 1.066e-01 0.0008261
## [57,] 25 1.066e-01 0.0007527
## [58,] 25 1.066e-01 0.0006859
##
## $lambda.min
## [1] 0.0007527427
##
## $lambda.1se
## [1] 0.003335118
##
## attr(,"class")
## [1] "cv.glmnet"
```

```
LassoModTest
```

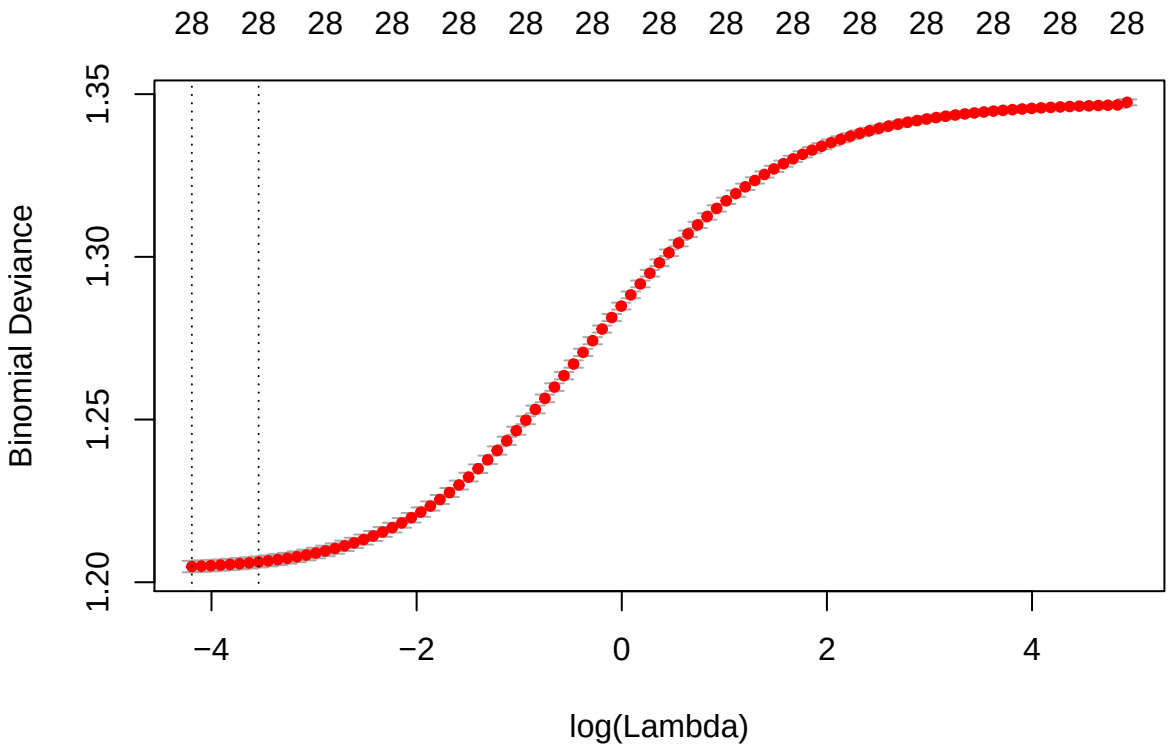
```
## $lambda
##  [1] 0.1391901920 0.1268249189 0.1155581427 0.1052922758 0.0959384002
##  [6] 0.0874154972 0.0796497453 0.0725738815 0.0661266179 0.0602521114
## [11] 0.0548994799 0.0500223614 0.0455785127 0.0415294433 0.0378400820
## [16] 0.0344784734 0.0314155008 0.0286246343 0.0260817007 0.0237646743
## [21] 0.0216534861 0.0197298501 0.0179771046 0.0163800682 0.0149249081
## [26] 0.0135990204 0.0123909209 0.0112901456 0.0102871602 0.0093732773
## [31] 0.0085405811 0.0077818594 0.0070905405 0.0064606364 0.0058866913
## [36] 0.0053637339 0.0048872345 0.0044530661 0.0040574680 0.0036970137
## [41] 0.0033685812 0.0030693257 0.0027966553 0.0025482081 0.0023218323
## [46] 0.0021155672 0.0019276260 0.0017563810 0.0016003490 0.0014581784
## [51] 0.0013286378 0.0012106053 0.0011030585 0.0010050658 0.0009157785
## [56] 0.0008344233 0.0007602954 0.0006927529 0.0006312106 0.0005751356
## [61] 0.0005240421 0.0004774877 0.0004350690 0.0003964187 0.0003612019
## [66] 0.0003291137 0.0002998762 0.0002732360
##
```

```
## $cvm
##  [1] 0.2404112 0.2371690 0.2344380 0.2321707 0.2302882 0.2286750 0.2267141
##  [8] 0.2250574 0.2236818 0.2225341 0.2211453 0.2195033 0.2180624 0.2168498
## [15] 0.2158390 0.2150004 0.2143036 0.2136904 0.2130789 0.2125696 0.2121464
## [22] 0.2117945 0.2114643 0.2111659 0.2108931 0.2106290 0.2104074 0.2102238
## [29] 0.2100707 0.2099369 0.2098155 0.2097069 0.2096105 0.2095163 0.2094310
## [36] 0.2093596 0.2093007 0.2092519 0.2092119 0.2091795 0.2091523 0.2091309
## [43] 0.2091141 0.2091002 0.2090887 0.2090787 0.2090701 0.2090628 0.2090562
## [50] 0.2090504 0.2090452 0.2090410 0.2090376 0.2090351 0.2090330 0.2090311
## [57] 0.2090294 0.2090281 0.2090271 0.2090263 0.2090258 0.2090252 0.2090248
## [64] 0.2090244 0.2090242 0.2090240 0.2090238 0.2090238
##
## $cvsd
##  [1] 0.0005115745 0.0005104942 0.0005013891 0.0004941178 0.0004883746
##  [6] 0.0004786175 0.0004808873 0.0004834279 0.0004866307 0.0004903008
## [11] 0.0004951668 0.0005129025 0.0005250718 0.0005338763 0.0005408288
## [16] 0.0005472547 0.0005531560 0.0005615865 0.0005623677 0.0005628521
## [21] 0.0005631530 0.0005632964 0.0005631596 0.0005632129 0.0005649824
## [26] 0.0005629069 0.0005605788 0.0005586931 0.0005568892 0.0005557495
## [31] 0.0005534573 0.0005522027 0.0005522089 0.0005518428 0.0005533504
## [36] 0.0005547754 0.0005562079 0.0005576921 0.0005590255 0.0005599090
## [41] 0.0005606998 0.0005616898 0.0005625713 0.0005635030 0.0005644721
## [46] 0.0005652394 0.0005657898 0.0005661433 0.0005661739 0.0005660127
## [51] 0.0005658039 0.0005656199 0.0005654576 0.0005653109 0.0005651310
## [56] 0.0005649527 0.0005647788 0.0005646394 0.0005645723 0.0005645130
## [61] 0.0005644596 0.0005644004 0.0005643395 0.0005642816 0.0005642301
## [66] 0.0005641843 0.0005641435 0.0005641205
##
## $cvup
##  [1] 0.2409228 0.2376795 0.2349394 0.2326648 0.2307765 0.2291536 0.2271949
##  [8] 0.2255408 0.2241685 0.2230244 0.2216405 0.2200163 0.2185875 0.2173837
## [15] 0.2163798 0.2155476 0.2148568 0.2142520 0.2136413 0.2131325 0.2127096
## [22] 0.2123578 0.2120274 0.2117291 0.2114581 0.2111919 0.2109680 0.2107825
## [29] 0.2106276 0.2104926 0.2103689 0.2102591 0.2101627 0.2100682 0.2099844
## [36] 0.2099144 0.2098569 0.2098096 0.2097709 0.2097394 0.2097130 0.2096926
## [43] 0.2096766 0.2096637 0.2096532 0.2096439 0.2096359 0.2096290 0.2096223
## [50] 0.2096164 0.2096110 0.2096066 0.2096031 0.2096004 0.2095981 0.2095960
## [57] 0.2095941 0.2095928 0.2095917 0.2095909 0.2095902 0.2095896 0.2095891
## [64] 0.2095887 0.2095884 0.2095882 0.2095880 0.2095879
##
## $cvlo
##  [1] 0.2398996 0.2366585 0.2339366 0.2316766 0.2297998 0.2281963 0.2262332
##  [8] 0.2245739 0.2231952 0.2220438 0.2206502 0.2189904 0.2175374 0.2163159
## [15] 0.2152982 0.2144531 0.2137504 0.2131288 0.2125166 0.2120068 0.2115833
## [22] 0.2112312 0.2109011 0.2106026 0.2103282 0.2100661 0.2098468 0.2096651
## [29] 0.2095138 0.2093811 0.2092620 0.2091547 0.2090583 0.2089645 0.2088777
## [36] 0.2088048 0.2087444 0.2086942 0.2086529 0.2086195 0.2085916 0.2085693
## [43] 0.2085515 0.2085367 0.2085243 0.2085134 0.2085043 0.2084967 0.2084900
## [50] 0.2084844 0.2084794 0.2084754 0.2084722 0.2084698 0.2084679 0.2084661
## [57] 0.2084646 0.2084635 0.2084625 0.2084618 0.2084613 0.2084608 0.2084605
## [64] 0.2084602 0.2084599 0.2084598 0.2084597 0.2084596
##
## $nzero
##  s0  s1  s2  s3  s4  s5  s6  s7  s8  s9 s10 s11 s12 s13 s14 s15 s16 s17
```

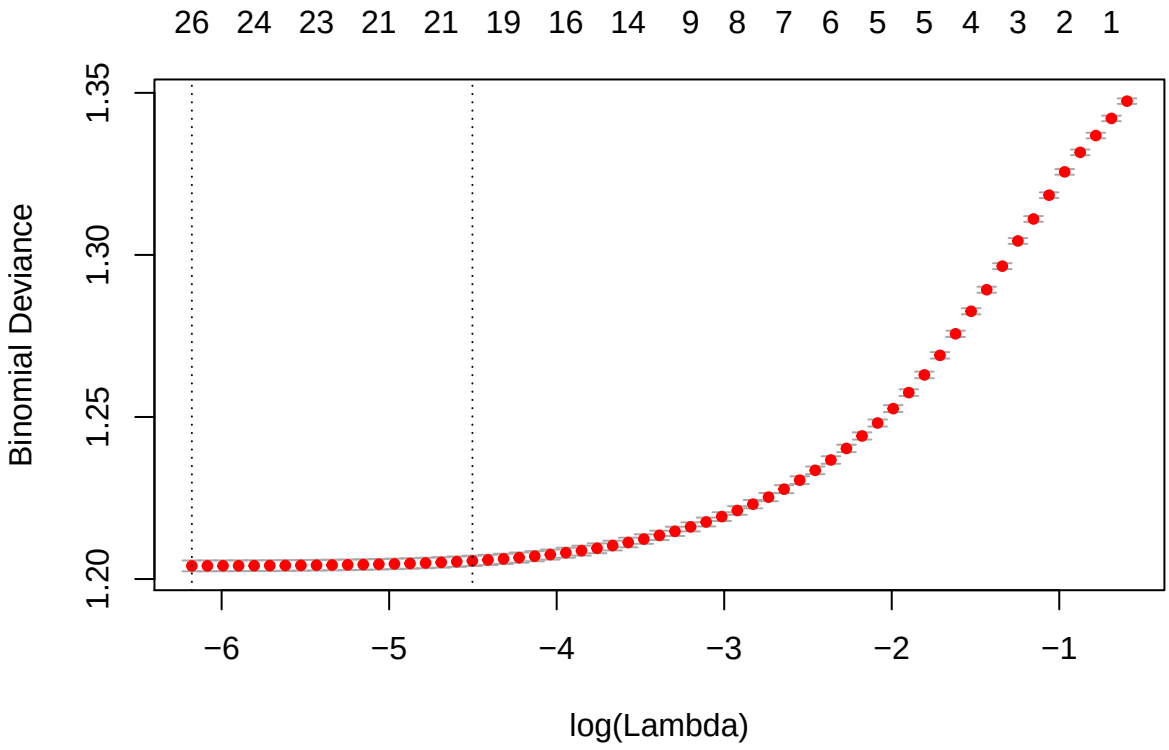
```
##   0   1   1   1   1   2   2   2   2   2   3   4   5   5   5   5   5   6
## s18 s19 s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35
##   6   6   6   6   7   7   8   8   8   8   8  10  11  11  12  12  12  12
## s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53
##  12  12  13  13  13  14  14  18  18  20  21  22  23  24  24  24  25  26
## s54 s55 s56 s57 s58 s59 s60 s61 s62 s63 s64 s65 s66 s67
##  26  26  26  26  26  26  26  26  26  26  26  27  27  27
##
## $name
##                 mse
## "Mean-Squared Error"
##
## $glmnet.fit
##
## Call:  glmnet(x = XvarTest, y = YvarTest)
##
##       Df    %Dev    Lambda
##  [1,]  0 0.00000 0.1392000
##  [2,]  1 0.01368 0.1268000
##  [3,]  1 0.02504 0.1156000
##  [4,]  1 0.03447 0.1053000
##  [5,]  1 0.04230 0.0959400
##  [6,]  2 0.04889 0.0874200
##  [7,]  2 0.05719 0.0796500
##  [8,]  2 0.06408 0.0725700
##  [9,]  2 0.06980 0.0661300
## [10,]  2 0.07455 0.0602500
## [11,]  3 0.08034 0.0549000
## [12,]  4 0.08720 0.0500200
## [13,]  5 0.09318 0.0455800
## [14,]  5 0.09825 0.0415300
## [15,]  5 0.10250 0.0378400
## [16,]  5 0.10590 0.0344800
## [17,]  5 0.10880 0.0314200
## [18,]  6 0.11140 0.0286200
## [19,]  6 0.11390 0.0260800
## [20,]  6 0.11610 0.0237600
## [21,]  6 0.11780 0.0216500
## [22,]  6 0.11930 0.0197300
## [23,]  7 0.12070 0.0179800
## [24,]  7 0.12190 0.0163800
## [25,]  8 0.12310 0.0149200
## [26,]  8 0.12420 0.0136000
## [27,]  8 0.12510 0.0123900
## [28,]  8 0.12590 0.0112900
## [29,]  8 0.12650 0.0102900
## [30,] 10 0.12710 0.0093730
## [31,] 11 0.12760 0.0085410
## [32,] 11 0.12810 0.0077820
## [33,] 12 0.12850 0.0070910
## [34,] 12 0.12890 0.0064610
## [35,] 12 0.12930 0.0058870
## [36,] 12 0.12960 0.0053640
## [37,] 12 0.12980 0.0048870
```

```
## [38,] 12 0.13000 0.0044530
## [39,] 13 0.13020 0.0040570
## [40,] 13 0.13040 0.0036970
## [41,] 13 0.13050 0.0033690
## [42,] 14 0.13060 0.0030690
## [43,] 14 0.13070 0.0027970
## [44,] 18 0.13080 0.0025480
## [45,] 18 0.13080 0.0023220
## [46,] 20 0.13090 0.0021160
## [47,] 21 0.13100 0.0019280
## [48,] 22 0.13100 0.0017560
## [49,] 23 0.13110 0.0016000
## [50,] 24 0.13110 0.0014580
## [51,] 24 0.13110 0.0013290
## [52,] 24 0.13120 0.0012110
## [53,] 25 0.13120 0.0011030
## [54,] 26 0.13120 0.0010050
## [55,] 26 0.13120 0.0009158
## [56,] 26 0.13120 0.0008344
## [57,] 26 0.13120 0.0007603
## [58,] 26 0.13130 0.0006928
## [59,] 26 0.13130 0.0006312
## [60,] 26 0.13130 0.0005751
## [61,] 26 0.13130 0.0005240
## [62,] 26 0.13130 0.0004775
## [63,] 26 0.13130 0.0004351
## [64,] 26 0.13130 0.0003964
## [65,] 26 0.13130 0.0003612
## [66,] 27 0.13130 0.0003291
## [67,] 27 0.13130 0.0002999
## [68,] 27 0.13130 0.0002732
## [69,] 27 0.13130 0.0002490
## [70,] 27 0.13130 0.0002268
##
## $lambda.min
## [1] 0.000273236
##
## $lambda.1se
## [1] 0.006460636
##
## attr(,"class")
## [1] "cv.glmnet"
```

```r
#Ridge Mod.
RidgeMod <- cv.glmnet(x = Xvar, y = Yvar, alpha = 0, nfolds = 10, family = "binomial")
plot(RidgeMod)
```




```r
#Elastic Net
ElasticNet1 <- cv.glmnet(x = Xvar, y = Yvar, alpha = .25, nfolds = 10, family = "binomial")
ElasticNet2 <- cv.glmnet(x = Xvar, y = Yvar, alpha = .50, nfolds = 10, family = "binomial")
ElasticNet3 <- cv.glmnet(x = Xvar, y = Yvar, alpha = .75, nfolds = 10, family = "binomial")
plot(ElasticNet1)
```



```r
#MSE
MSE <- function(true, preds) { mean((true - preds)^2) }
MSE(KickTrain$binomState, predict(LassoMod, newx = Xvar))
```

```
## [1] 1.456998
```

```r
MSE(KickTrain$binomState, predict(RidgeMod, newx = Xvar))
```

```
## [1] 1.339221
```

```r
MSE(KickTrain$binomState, predict(ElasticNet1, newx = Xvar))
```

```
## [1] 1.398907
```

```r
MSE(KickTrain$binomState, predict(ElasticNet2, newx = Xvar))
```

```
## [1] 1.42412
```

```r
MSE(KickTrain$binomState, predict(ElasticNet3, newx = Xvar))
```

```
## [1] 1.434038
```

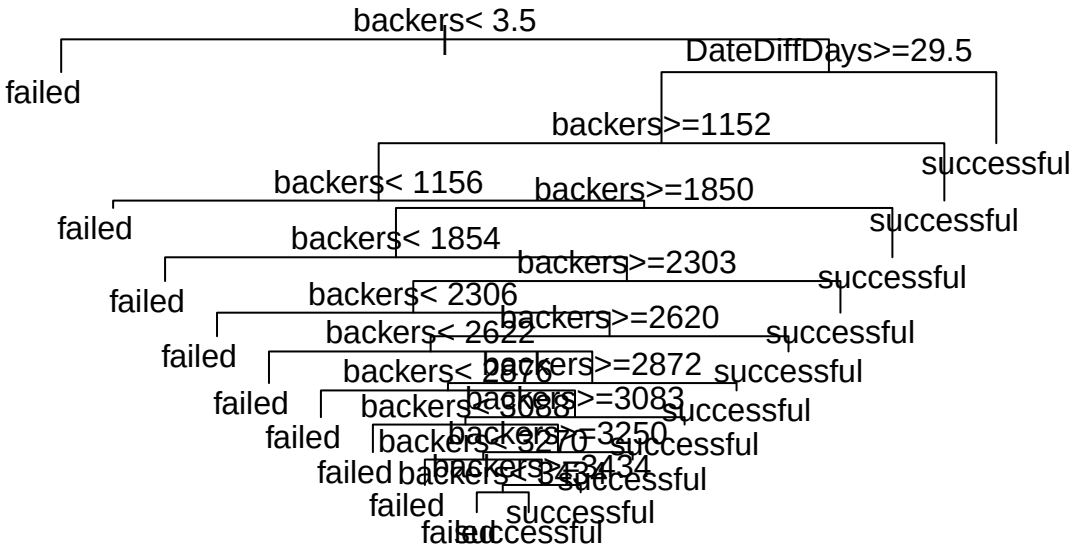
```r
#MSE TEST
MSE(KickTest$binomState, predict(LassoModTest, newx = XvarTest))
```

```
## [1] 0.2094453
```

Tree Based Methods

```r
newKickStar <- subset(KickStar, select = -c(1,2,3,4,5,6,8,12,14,15))
#View(newKickStar)
trainSize <- .05
trainInd <- sample(1:nrow(newKickStar), size = floor(nrow(newKickStar) * trainSize))
newKickTrain <- newKickStar[trainInd, ]
newKickTest <- newKickStar[-trainInd, ]
```

```r
#README!!! make sure you click the green arrow to run this properly
#method = class for classification problem
tree <- rpart(state ~ .-binomState, data = newKickTrain, method = "class", control = rpart.control(minsp
par(xpd=TRUE)
plot(tree); text(tree, pretty = 0)
```



Random Forest

WEBSITE USED: https://www.statmethods.net/advstats/cart.html

```
randTree <- randomForest(state ~ ., data=newKickTrain, mtry=3, method = "class")
#importance(randTree)
#plot(randTree)

randForPredictTrain <- predict(randTree, newdata = newKickTrain, mtry = 3)
#randForTrainMSE = mean((newKickTrain$state - randForPredictTrain)^2)
#randForPredictTest <- predict(randFor, newdata = newKickTest, mtry = 3)
#randForTestMSE = mean((AutoTest$mpg - randForPredictTest)^2)
#print(randForTrainMSE)
#print(randForTestMSE)
```

USE THIS ONE

Pruning Tree

```
# HAVENT TOUCHED THIS YET
# predict using the test data
tree.pred <- predict(tree, newKickTrain,type="class")
# note with() command evaluates an R expression in an environment
# constructed from the data,
with(newKickTrain,table(tree.pred,state))
```

```
##             state
## tree.pred    failed successful
##   failed       5559        188
##   successful   2832       5475
```

```
#install.packages('tree')
require('ISLR')
```

```
## Loading required package: ISLR
```

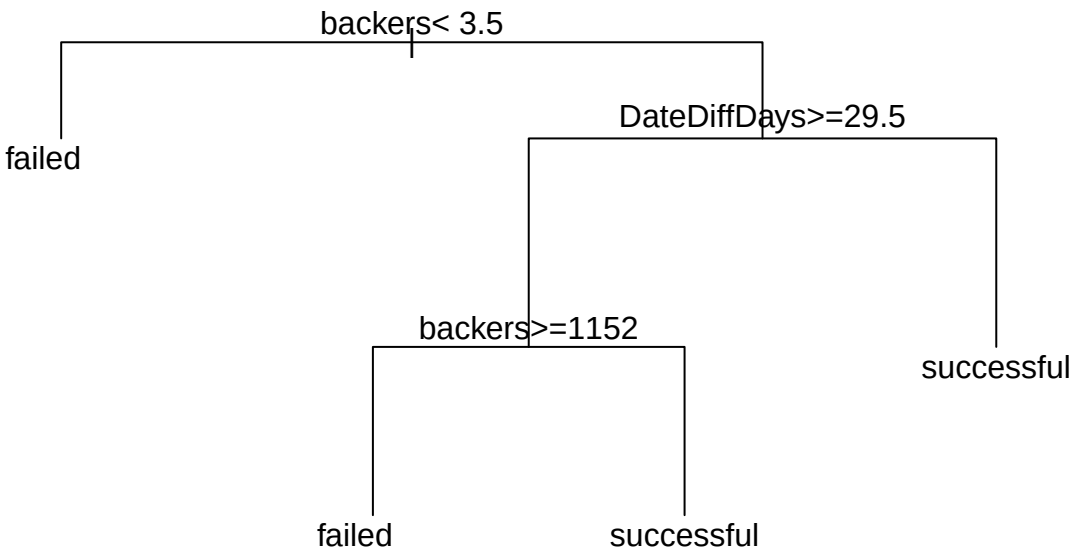
```
require('tree')
```

```
## Loading required package: tree
```

```
# now do some cross-validating
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = state ~ . - binomState, data = newKickTrain,
##     method = "class", control = rpart.control(minsplit = 30,
##         cp = 0.01))
##
## Variables actually used in tree construction:
## [1] backers      DateDiffDays
##
## Root node error: 5663/14054 = 0.40295
##
## n= 14054
##
##         CP nsplit rel error  xerror      xstd
## 1 0.054830      0   1.00000 1.00000 0.0102679
## 2 0.026046      3   0.82977 0.82977 0.0098759
## 3 0.024898      7   0.71923 0.77556 0.0097033
## 4 0.024722      8   0.69433 0.72576 0.0095226
## 5 0.015539     10   0.64489 0.64542 0.0091832
## 6 0.013420     14   0.58114 0.60833 0.0090050
## 7 0.010507     16   0.55430 0.58114 0.0088651
## 8 0.010000     18   0.53329 0.55642 0.0087307
```

```
pruned.tree <- prune(tree, cp = 0.028)
par(xpd=TRUE)
plot(pruned.tree); text(pruned.tree, pretty = 0)
```



USE THIS ONE.

WORK ON THIS LATER

```
MSE <- function(truth, predict) {mean((truth - predict)^2)}
predsTrain <- predict(tree, newdata = newKickTrain)
predsValidate <- predict(tree, newdata = newKickTest)
MSE(newKickTrain$binomState, predsTrain)
```

```
## [1] 0.3575032
```

```
MSE(newKickTest$binomState, predsValidate)
```

```
## [1] 0.3587691
```