Capstone Data Logistic Regression - Predict Spruce and Fir

Tom Thorpe

July 25, 2018

Objective

Use Logistic regression to predict tree coverage.

```
# Include required libraries.

library(gsubfn)
```

```
## Loading required package: proto
```

```
library(dplyr)
```

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

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

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

```
library(ggplot2)
library(ggridges) # for easier viewing of sub-group distributions
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

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

```
suppressMessages(library(latticeExtra, warn.conflicts = FALSE, quietly=TRUE))
#library(latticeExtra)

  curTime=Sys.time()
  print(paste("Forest Cover Logistic script started at",curTime))
```

```
## [1] "Forest Cover Logistic script started at 2018-08-12 21:48:04"
```

```
#Point to data. The forestcover_clean_full.csv is the cleaned data to be graphed.

calcROC <- 1
saveFileName="ForestCoverLogisticStats.csv"

infile="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestcover_clean_full.csv"
```

```
#infile="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestcover_clean.csv"
#infile="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestcoversmall_clean_full.csv"
#infile="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestcoversmall_clean.csv"
out2file="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestcover_graph.csv"
#out1file="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestcoversmall_clean_full.csv"
#out2file="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestcoversmall_clean.csv"

alphaVal<-0.05 # large data
#alphaVal<-0.1  # small data
```

```
forestcover <- read.csv(infile,header=TRUE,sep=",") %>% tbl_df()
  curTime=Sys.time()
  print(paste("Forest Cover data load completed at",curTime))
```

```
## [1] "Forest Cover data load completed at 2018-08-12 21:48:45"
```

```
forestcover$SoilType<-as.factor(forestcover$SoilType)
forestcover$ClimateZone<-as.factor(forestcover$ClimateZone)
forestcover$GeoZone<-as.factor(forestcover$GeoZone)

# glimpse(forestcover)

# table(forestcover$Sed_mix)
#knitr::knit_exit()

# Coverage binary outcome Vars:
# Aspen
# Cottonwood_Willow
# DouglasFir
# Krummholz
# LodgepolePine
# PonderosaPine
# Spruce_Fir
```

A table showing the number of occurrences for each tree type is shown below.

```
covCount<-data.frame(table(forestcover$CovName))
totCount<-nrow(forestcover)
covCount <- mutate(covCount,Percent = as.integer(covCount$Freq*1000/totCount)/10)
LodgePct<-covCount$Percent[covCount$Var1=="Lodgepole"]
SpruceAndFirPct<-covCount$Percent[covCount$Var1=="Spruce&Fir"]
LodgeAndSpruceAndFir<-LodgePct+SpruceAndFirPct
#```
#```{r echo=TRUE}
covCount
```

```
##             Var1   Freq Percent
## 1          Aspen   9493     1.6
## 2 Cotton&Willow   2747     0.4
## 3    DouglasFir  17367     2.9
## 4     Krummholz  20510     3.5
## 5     Lodgepole 283301    48.7
## 6     Ponderosa  35754     6.1
## 7    Spruce&Fir 211840    36.4
```

Lodge pole Pine represents 48.7 percent of the sample. So always guessing “Lodge pole” would provide success rate of 48.7 percent and can be used as a baseline for comparing our predictions. Spruce & Fir represent the next largest number of trees. The two together represent 85.1 percent.

Logistic Model Accuracy Function

A function to help determine threshold for best accuracy and testing is shown below.

```r
#load("logisticAccuracy.Rdata")

calcLogisticModelAccuracy <- function(actualValues, predictedValues,
                          thresholdStart, thresholdEnd, thresholdParts,
                          positiveLabel, negativeLabel, printLevel,
                          findThreshold=0,
                          saveFile="", desc="LogisticStats", AIC=NA, AUC=NA, Append=TRUE) {
  # Description
  #   -Calculate accuracy of logistic regression model
  #   -depending on print level option:
  #      print accuracy of logistic model and baseline model
  #      print confusion matrix
  #      print sensitivity and specificty
  #
  # Input Values
  #   -actualValues = actual values of outcome variables, a vector of 0's and 1's
  #   -predictedValues = logistic model predicted probalilities between 0 and 1
  #   -thresholdStart = threshold initial value for applying to predicted values
  #      to determine predicted outcome
  #   -thresholdEnd = end value for incrementing the threshold
  #   -thresholdParts = number of partitions to apply threshold values between
  #      thresholdStart and thresholdEnd
  #   -positiveLabel = text to label true outcomes. This will be displayed
  #      on the confusion matrix when the print level is greater than 1.
  #   -negativeLable = text to label false outcomes. This will be displayed
  #      on the confusion matrix when the print level is greater than 1.
  #   -printLevel = level of detail printed by calcLogisticModelAccuracy
  #      0 - no printed output unless and error is encountered
  #      1 - print threshold, logistic model accuracy and baseline accuracy
  #      2 - Print level 1 and confusion matrix and sensitivity and specificity values
  #      3 - Print level 2 and details of sensitivity and specificity calculations
  #      4 - Print level 3 and debug information
  # -findThreshold
  #      1 - search for threshold producing best sensitivity and specificity combination
  #      2 - search for threshold producing best accuracy
  #
  # Return Values
  #   -function status:
  #      - "OK":function completed without errors
  #      - "ERROR": function did not complete, and error information
  #         See other variables for possible additional error information
  #   -logistic model accuracy based on last threshold value tested
  #   -baseline model accuracy based on last threshold value tested
  #   -confusion matrix values in following order: TN, FN, FP, TP
  #   -sensitivity
  #   -specificity
```

```r
# x <- data.frame('1', 'ab', 'username', '<some.sentence>', '2017-05-04T00:51:35Z', '24')
# write.table(x, file = "Tweets.csv", sep = ",", append = TRUE, quote = FALSE,
#  col.names = FALSE, row.names = FALSE)
if (saveFile != "") {
  if (!file.exists(saveFile) | Append == FALSE){
     if (file.exists(saveFile)) { file.remove(saveFile) }
     x <- data.frame('Description', 'TrueLabel', 'FalseLabel', 'BaselineLabel',
      'BaselineAcc', 'Accuracy', 'Sensitivity', 'Specificity',
      'AIC', 'AUC%', 'TP', 'FN', 'FP','TN', 'Count', 'Threshold' )
     write.table(x, file=saveFile, sep=",", quote=FALSE,
       col.names = FALSE, row.names = FALSE)
  }
}

# set default values in case of errors
accuracy=baseline=retVal="ERROR"
more=1
bestLabel="Sensitivity_Specificity"
SensSpec = -1

bestAccuracy=bestThreshold=NA # set default values for bestThreshold calcs

if (findThreshold) {
  thresholdStart=0.0
  thresholdEnd=1.0
  thresholdParts=10
  more=3 # allows calculation to find threshold to nearest 0.001
  bestAccuracy=bestThreshold=-1.0
  if (findThreshold == 2) { bestLabel = "Accuracy" }
  print (paste("Searching for threshold producing best",bestLabel))
}

# Calculate increment value to iterate through the threshold values
if ( thresholdParts ==0) { thresholdParts = 1 }
if ( thresholdParts < 0) { thresholdParts = - thresholdParts }
thresholdInc = (thresholdEnd - thresholdStart) / thresholdParts
if (thresholdStart==thresholdEnd | thresholdParts < 2) {
  thresholdEnd=thresholdStart
  thresholdInc=1
}
threshold=thresholdStart

if (findThreshold) {
  print(paste("start=",thresholdStart,"end=",thresholdEnd,"inc=",thresholdInc))
}

funcStat="OK"

workPerformance = table(actualValues, predictedValues > threshold)

for (row in rownames(workPerformance)) {
  if(row != "0" & row != "1") {
    funcStat=paste("ERROR:Bad row name:",row,",must be '0' or '1'")}
```

```
}
for (col in colnames(workPerformance)) {
  if(col != "TRUE" & col != "FALSE") {
    funcStat=paste("ERROR:Bad column name:",col,", must be 'TRUE' or 'FALSE'")}
}

if (funcStat=="OK") {
  while (more) {
  repeat {

    if (thresholdParts>1 & printLevel > 1) { print("----------")}

    workPerformance = table(actualValues, predictedValues > threshold)

    # create a modelPerformance table and set all the values to zero.
    # This ensures a 2x2 matrix in case the threshold causes all values predicted
    # to be TRUE or FALSE values and produces a 2x1 vector.
    # The table of actual and predicted values with be copied into the
    # modelPerformance table later.
    Actual = c(0, 1)
    Predicted = c(FALSE, TRUE )
    modelPerformance = table(Actual,Predicted)
    modelPerformance["0","TRUE"]=0
    modelPerformance["0","FALSE"]=0
    modelPerformance["1","FALSE"]=0
    modelPerformance["1","TRUE"]=0

    # Descriptions          | Predict Good Care (0) | Predict Poor Care (1)
    # ---------------------|-----------------------|----------------------
    # Actual Good Care (0) |     TN (true neg)     |   FP (false pos)
    # Actual Poor Care (1) |     FN (false neg)    |   TP (true pos)

    # Remember: 0 means negative which means Good care,
    #           1 means positive which means Poor care
    #   (Opposite of intuition)

    # Sensitivity = TP / (TP + FN) = percent of true positives identified

    # Specificity = TN / (TN + FP) = percent of true negatives identified

    # transfer the workPerformance table to the final performance table
    for (row in rownames(workPerformance)) {
      for (col in colnames(workPerformance)) {
        modelPerformance[row,col]=workPerformance[row,col]
        if (printLevel > 3) { print(paste("workPerformance[",row,",",col,"]=",
                                    workPerformance[row,col]))}
      }
    }

    if (printLevel > 3) {print(modelPerformance) }

    #                  Actual,Prediction
    TP = modelPerformance["1","TRUE"]  # Predicted True (1),  and actually TRUE (1) = True Positive
```

```
FN = modelPerformance["1","FALSE"] # Predicted False (0), but actually TRUE (1) = False 0/Negativ

TN = modelPerformance["0","FALSE"] # Predicted False (0), and actually False (0) = True Negative
FP = modelPerformance["0","TRUE"] # Predicted True (1), but actually False (0) = False 1/Positive

# Prevent and report divide by zero error
if (TP+FN == 0) {
  sensitivity="ERROR:TP+FN=0"
  funcStat=sensitivity
} else { sensitivity = TP / (TP + FN ) }

# Prevent and report divide by zero error
if (TN+FP == 0) {
  specificity="ERROR:TN+FP=0"
  funcStat=specificity
} else { specificity = TN / (TN + FP) }

if (funcStat == "OK") { # calc SensSpec
  if (sensitivity > 0.0 & sensitivity < 1.0) {
      SensSpec=sensitivity^2 + specificity^2
  } else { SensSpec = -2.0 }
      printSensSpec=as.integer(SensSpec*1000)/1000
}

retVal = c(modelPerformance, sensitivity,specificity) # TN, FN, FP, TP, sens, spec

if (printLevel > 1) {
  modelPerformance["1","TRUE"]  = paste("   ",modelPerformance["1","TRUE"], "(TP)")
  modelPerformance["1","FALSE"] = paste("   ",modelPerformance["1","FALSE"],"(FN)")
  modelPerformance["0","FALSE"] = paste("   ",modelPerformance["0","FALSE"],"(TN)")
  modelPerformance["0","TRUE"]  = paste("   ",modelPerformance["0","TRUE"], "(FP)")
}

c1=paste("FALSE=Predict:",negativeLabel,sep="")
c2=paste("TRUE=Predict:",positiveLabel,sep="")
r1=paste("0=Actual:",negativeLabel,sep="")
r2=paste("1=Actual:",positiveLabel,sep="")
colnames(modelPerformance) <- c(c1,c2)
rownames(modelPerformance) <- c(r1,r2)

if (printLevel > 1) {
  print(paste("Model Performance for threshold=", threshold))
  print("predicted performance=")
  print(modelPerformance)

  sensPrint=paste("Sensitivity=",sensitivity,"(True positive rate of",positiveLabel)

  specPrint=paste("Specificity=",specificity,"(True negative rate of",negativeLabel)

  if (printLevel > 2 ) {
    sensPrint=paste(sensPrint,"= TP/(TP+FN) =",TP,"/(",TP,"+",FN,"))")
    specPrint=paste(specPrint,"= TN/(TN+FP) =",TN,"/(",TN,"+",FP,"))")
  }
```

```
    print(sensPrint)
    print(specPrint)
  }

  # Calculate actual true and actual false totals to calculate baseline accuracy
  # and logistic model accuracy
  totSamples=TP+FN+TN+FP
  actTrue=TP+FN
  actFalse=TN+FP

  # double check there were actually some non-zero values
  if (totSamples>0) {
    if (actTrue > actFalse) {
      baseline = actTrue / totSamples
      baseModel= positiveLabel
    } else {
      baseline = actFalse / totSamples
      baseModel=negativeLabel
    }

    # the accuracy is the number of TRUE positives and True negatives
    # divided by the number of samples
    accuracy=(TP+TN)/totSamples
    if (findThreshold)  {
      if (findThreshold == 2) {
        if (accuracy > bestAccuracy) {
          bestAccuracy=accuracy
          bestThreshold=threshold
        }
      } else {
        if (SensSpec > bestAccuracy) {
          bestAccuracy=SensSpec
          bestThreshold=threshold
        }
      }
    }
  } else {
    baseModel="ERROR:0 samples"
    baseline="ERROR:0 samples"
    accuracy="ERROR:0 samples"
    funcStat=accuracy
  }

  if (printLevel > 0) {

    printAcc=(as.integer(accuracy*1000000))/1000000
    printbaseline=(as.integer(baseline*1000000))/1000000

    if (printLevel > 1) {
      print(paste("Sens^2+Spec^2=",printSensSpec,sep=""))
      print(paste("Baseline (",baseModel,") Accuracy=",printbaseline,sep=""))
      print(paste("Logistic Accuracy=",printAcc,sep=""))
    } else {
```

```r
        print(paste("Thresh=",threshold,
              ", Accuracy=",as.integer(accuracy*1000)/10,
              "%, BaseAcc(",baseModel,")=",as.integer(baseline*1000)/10,
              "%, Sens=",as.integer(sensitivity*1000)/10,
              "%, Spec=",as.integer(specificity*1000)/10,
              "%, Sens^2+Spec^2=",printSensSpec,
              sep=""))
      }
    }

    # c(funcStat,accuracy,baseline,retVal)

    #print(paste("threshold=",threshold,",End=",thresholdEnd,",Inc=",thresholdInc))

    threshold=threshold+thresholdInc
    if(thresholdEnd < thresholdStart) {
      if (threshold < thresholdEnd) { break}
    } else { if (threshold > thresholdEnd) { break} }

  } # end repeat

    more=more-1

    if (findThreshold & more) {
      print(paste("Best",bestLabel,"threshold=",bestThreshold,"inc=",thresholdInc))
      thresholdStart = bestThreshold - thresholdInc
      if (thresholdStart < 0.0) { thresholdStart = 0.0 }
      thresholdEnd = bestThreshold + thresholdInc
      if (thresholdEnd > 1.0) { thresholdEnd = 1.0 }
      thresholdInc = (thresholdEnd - thresholdStart) / 20.0
      threshold=thresholdStart
      print("========================================")
      print(paste("start=",thresholdStart,"end=",thresholdEnd,"inc=",thresholdInc))
    }
  } # end while

  if (findThreshold) {
    print("========================================")
    print(paste("Best Threshold=",bestThreshold,sep=""))
    print(paste("Best ",bestLabel,"=",bestAccuracy,sep=""))
  }
} else {
  # Had an error, just return the error information
  print(funcStat)
}

# x <- data.frame('Description', 'TrueLabel', 'FalseLabel', 'BaselineLabel',
#       'BaseLineAcc', 'Accuracy', 'Sensitivity', 'Specificity',
#       'AIC', 'AUC', 'TP', 'FN', 'TP','TN', 'Count' )
if (saveFile != "" & funcStat == "OK") {
  threshold=threshold-thresholdInc
  x <-data.frame(desc, positiveLabel,negativeLabel, baseModel,
    baseline, accuracy, sensitivity, specificity,
```

```
      AIC, AUC, TP, FN, FP,TN, totSamples, threshold)
    write.table(x, file=saveFile, sep=",", append=TRUE, quote=FALSE,
      col.names = FALSE, row.names = FALSE)
  }

  c(funcStat,accuracy,baseline,retVal,bestAccuracy,bestThreshold)
}
bestThreshIndex=11
#save("calcLogisticModelAccuracy", file="logisticAccuracy.Rdata")
# OR
# dump("add2", file="myFunction.R")
dump("calcLogisticModelAccuracy", file="logisticAccuracy.R")

## Then in a subsequent R session
# source("logisticAccuracy.R")
```

Create Training and Testing Sets

Split data into training and testing data for logistic regression. The split is based on cover type so that the different coverage types will be split proportionately for all cover types in the training and test sets.

```
library(caTools)
set.seed(127)
split = sample.split(forestcover$CovType, 0.70) # we want 65% in the training set
forestTrain = subset(forestcover, split == TRUE)
forestTest  = subset(forestcover, split == FALSE)
```

Check training set coverage percentages and compare with test set to ensure there is a representative amount of data in each set for each coverage type.

View Training Set Coverage Percentages

Check training set coverage percentages.

```
covCount<-data.frame(table(forestTrain$CovName))
totCount<-nrow(forestTrain)
covCount <- mutate(covCount,Percent = as.integer(covCount$Freq*1000/totCount)/10)
covCount
```

```
##            Var1   Freq Percent
## 1         Aspen   6645     1.6
## 2 Cotton&Willow   1923     0.4
## 3    DouglasFir  12157     2.9
## 4     Krummholz  14357     3.5
## 5     Lodgepole 198311    48.7
## 6     Ponderosa  25028     6.1
## 7    Spruce&Fir 148288    36.4
```

View Test Set Coverage Percentages

Check test set coverage percentages.

```r
covCount<-data.frame(table(forestTest$CovName))
totCount<-nrow(forestTest)
covCount <- mutate(covCount,Percent = as.integer(covCount$Freq*1000/totCount)/10)
covCount
```

```
##            Var1  Freq Percent
## 1         Aspen  2848     1.6
## 2 Cotton&Willow   824     0.4
## 3    DouglasFir  5210     2.9
## 4     Krummholz  6153     3.5
## 5     Lodgepole 84990    48.7
## 6     Ponderosa 10726     6.1
## 7    Spruce&Fir 63552    36.4
```

```r
# knitr::knit_exit() # exit early

#glimpse(forestTrain)
#glimpse(forestTest)
#summary(forestTrain)
#summary(forestTest)
#table(forestTrain$Sed_mix)
#table(forestTrain$GeoName)
#table(forestTrain$Spruce_Fir)
#table(forestTest$Spruce_Fir)

# the above all work without error.

#table(forestTest$Rock_Land)
# Get the following error with above code:
#  Error in table(SpfFir_test$Rock_Land) : object 'SpfFir_test' not found
#    Calls: <Anonymous> ... withCallingHandlers -> withVisible -> eval -> eval -> table


#table(forestTrain$Rock_Land)
#table(forestTest$Rock_Land)
#table(forestTrain$Rubbly)
#table(forestTest$Rubbly)

#table(forestTrain$Sed_mix)
#table(forestTrain$Gateview)
#table(forestTrain$Rubbly)
#table(forestTest$Sed_mix)
#table(forestTest$Gateview)
#table(forestTest$Rubbly)

############# Start Start Start Start Start Start Start Start ##################
```

Spruce and Fir Logistic Regression

Logistic regression models are created and compared for the Spruce and Fir coverage type. The outcome is based on the binary 'Spruce_Fir' variable.

Spruce and Fir Logistic Regression - All Variables

Create Spruce and Fir Logistic Model - All Vars

Create the Spruce and Fir logistic model for the Aggregated Soil data using all independent variables.

Spruce and Fir All Aggregated Soil Types

The original project used aggregated Soil Types. Compute a logistic regression model using the aggregated soil types to see how the dis-aggregated / individuated variables compare.

```
# You can remove the levels of the factor variables using the option exclude:
#   lm(dependent ~ factor(independent1, exclude=c('b','d')) + independent2)
#   This way the factors b, d will not be included in the regression.

curTime=Sys.time()
print(paste("Spruce_Fir aggregated Logistic Model Calculation started at",curTime))
```

```
## [1] "Spruce_Fir aggregated Logistic Model Calculation started at 2018-08-12 21:48:48"
```

```
SprFir_Agg_LogMod =
  glm(Spruce_Fir ~
        Elev +      # Elevation in meters of data cell
        Aspect +    # Direction in degrees slope faces
        Slope +     # Slope / steepness of hill in degrees (0 to 90)
        H2OHD +     # Horizontal distance in meters to nearest water
        H2OVD +     # Vertical distance in meters to nearest water
        RoadHD +    # Horizontal distance in meters to nearest road
        FirePtHD + # Horizontal distance in meters to nearest fire point
        Shade9AM + Shade12PM + Shade3PM + # Amount of shade at 9am, 12pm and 3pm
        # Wilderness areas:
          RWwild + NEwild + CMwild + CPwild +
        # Aggregated Soil type:
          ST01 + ST02 + ST03 + ST04 + ST05 + ST06 + ST07 + ST08 + ST09 + ST10 +
          ST11 + ST12 + ST13 + ST14 + ST15 + ST16 + ST17 + ST18 + ST19 + ST20 +
          ST21 + ST22 + ST23 + ST24 + ST25 + ST26 + ST27 + ST28 + ST29 + ST30 +
          ST31 + ST32 + ST33 + ST34 + ST35 + ST36 + ST37 + ST38 + ST39 + ST40 ,
        data=forestTrain, family=binomial)
```

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

```
SprFir_Agg_All_LogMod = SprFir_Agg_LogMod
save("SprFir_Agg_All_LogMod", file="SprFir_Agg_All_LogMod.Rdata")

SprFir_Agg_All_aic<-as.integer(SprFir_Agg_LogMod$aic)
SprFir_Agg_All_aic
```

```
## [1] 379526
```

```
curTime=Sys.time()
print(paste("Spruce_Fir aggregated Logistic Model Calculation completed at",curTime))
```

```
## [1] "Spruce_Fir aggregated Logistic Model Calculation completed at 2018-08-12 21:52:01"
```

Check the coefficients for the Spruce and Fir model using all aggregated data.

```
summary(SprFir_Agg_LogMod)
```

```
##
## Call:
## glm(formula = Spruce_Fir ~ Elev + Aspect + Slope + H2OHD + H2OVD +
##     RoadHD + FirePtHD + Shade9AM + Shade12PM + Shade3PM + RWwild +
##     NEwild + CMwild + CPwild + ST01 + ST02 + ST03 + ST04 + ST05 +
##     ST06 + ST07 + ST08 + ST09 + ST10 + ST11 + ST12 + ST13 + ST14 +
##     ST15 + ST16 + ST17 + ST18 + ST19 + ST20 + ST21 + ST22 + ST23 +
##     ST24 + ST25 + ST26 + ST27 + ST28 + ST29 + ST30 + ST31 + ST32 +
##     ST33 + ST34 + ST35 + ST36 + ST37 + ST38 + ST39 + ST40, family = binomial,
##     data = forestTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4516  -0.7610  -0.2834   0.8360   3.0828
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.115e+10  4.814e+11  -0.106   0.9154
## Elev         7.018e-03  3.405e-05 206.133  < 2e-16 ***
## Aspect      -3.687e-04  4.855e-05  -7.594 3.09e-14 ***
## Slope       -6.987e-03  1.570e-03  -4.451 8.54e-06 ***
## H2OHD       -1.406e-03  2.506e-05 -56.110  < 2e-16 ***
## H2OVD       -1.582e-03  9.384e-05 -16.855  < 2e-16 ***
## RoadHD      -9.385e-05  3.191e-06 -29.414  < 2e-16 ***
## FirePtHD    -8.921e-06  3.753e-06  -2.377   0.0174 *
## Shade9AM     1.053e-02  1.803e-03   5.840 5.23e-09 ***
## Shade12PM   -3.433e-02  1.483e-03 -23.145  < 2e-16 ***
## Shade3PM     1.620e-02  1.475e-03  10.984  < 2e-16 ***
## RWwild       5.115e+10  4.814e+11   0.106   0.9154
## NEwild       5.115e+10  4.814e+11   0.106   0.9154
## CMwild       5.115e+10  4.814e+11   0.106   0.9154
## CPwild       5.115e+10  4.814e+11   0.106   0.9154
## ST01         2.647e+00  2.681e+03   0.001   0.9992
## ST02        -1.854e+01  1.604e+03  -0.012   0.9908
## ST03        -1.786e+01  1.942e+03  -0.009   0.9927
## ST04         9.402e-01  9.950e-02   9.449  < 2e-16 ***
## ST05         2.396e+00  3.606e+03   0.001   0.9995
## ST06         1.306e+00  1.971e+03   0.001   0.9995
## ST07        -2.164e+01  1.541e+04  -0.001   0.9989
## ST08         1.950e+00  1.984e-01   9.827  < 2e-16 ***
## ST09         2.842e+00  1.082e-01  26.275  < 2e-16 ***
## ST10         1.736e+00  5.427e-02  31.994  < 2e-16 ***
## ST11         1.570e+00  5.845e-02  26.866  < 2e-16 ***
## ST12         9.568e-01  4.222e-02  22.664  < 2e-16 ***
## ST13         1.432e+00  4.473e-02  32.005  < 2e-16 ***
## ST14        -1.673e+01  5.243e+03  -0.003   0.9975
## ST15         2.765e+00  7.490e+04   0.000   1.0000
## ST16         2.107e+00  6.858e-02  30.723  < 2e-16 ***
## ST17         1.993e+00  9.301e-02  21.424  < 2e-16 ***
## ST18         1.253e+00  1.714e-01   7.308 2.71e-13 ***
## ST19         2.260e+00  5.186e-02  43.574  < 2e-16 ***
## ST20         2.450e+00  4.366e-02  56.117  < 2e-16 ***
## ST21         5.103e+00  2.059e-01  24.778  < 2e-16 ***
## ST22         2.688e+00  3.466e-02  77.544  < 2e-16 ***
```

```
## ST23          2.394e+00  3.328e-02  71.928  < 2e-16 ***
## ST24          2.151e+00  3.610e-02  59.593  < 2e-16 ***
## ST25          6.722e-01  1.355e-01   4.960 7.05e-07 ***
## ST26          1.422e+00  8.557e-02  16.614  < 2e-16 ***
## ST27          2.484e+00  8.189e-02  30.339  < 2e-16 ***
## ST28          8.428e-01  2.133e-01   3.951 7.78e-05 ***
## ST29          1.385e+00  3.257e-02  42.520  < 2e-16 ***
## ST30          1.228e+00  3.652e-02  33.639  < 2e-16 ***
## ST31          2.110e+00  3.593e-02  58.738  < 2e-16 ***
## ST32          1.652e+00  3.353e-02  49.267  < 2e-16 ***
## ST33          2.073e+00  3.430e-02  60.429  < 2e-16 ***
## ST34         -9.247e-02  1.383e-01  -0.669   0.5037
## ST35         -1.828e-02  6.398e-02  -0.286   0.7751
## ST36         -4.699e-01  3.248e-01  -1.447   0.1480
## ST37         -2.553e+01  8.744e+03  -0.003   0.9977
## ST38          5.601e-01  3.539e-02  15.828  < 2e-16 ***
## ST39          5.249e-01  3.623e-02  14.486  < 2e-16 ***
## ST40                 NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 533620  on 406708  degrees of freedom
## Residual deviance: 379418  on 406655  degrees of freedom
## AIC: 379526
##
## Number of Fisher Scoring iterations: 23
```

WOW! The intercept is huge and listed as not significant. Wilderness area and several soil types are not significant and can be removed in the next iteration.

Spruce and Fir All Individuated Soil Types

Create a logistic model using the Individuated variables that were derived from the Soil Types. The Soil Type was the intersection of climate zone, geology zone, soil families, and rock content. These variables are used instead of the Soil types.

```
curTime=Sys.time()
print(paste("Spruce_Fir Individual Logistic Model Calculation started at",curTime))
```

```
## [1] "Spruce_Fir Individual Logistic Model Calculation started at 2018-08-12 21:52:01"
```

```
SprFir_Ind_LogMod =
  glm(Spruce_Fir ~
        Elev +     # Elevation in meters of cell
        Aspect +   # Direction in degrees slope faces
        Slope +    # Slope / steepness of hill in degrees (0 to 90)
        H2OHD +    # Horizontal distance in meters to nearest water
        H2OVD +    # Vertical distance in meters to nearest water
        RoadHD +   # Horizontal distance in meters to nearest road
        FirePtHD + # Horizontal distance in meters to nearest fire point
        Shade9AM + Shade12PM + Shade3PM + # Amount of shade at 9am, 12pm and 3pm
        # Wilderness areas:
          RWwild + NEwild + CMwild + CPwild +
```

```r
          # Climate Zone:
          # ClimateName +
            Montane_low + Montane + SubAlpine + Alpine + Dry + Non_Dry +
          # Geology Zone:
          # GeoName +
            Alluvium + Glacial + Sed_mix + Ign_Meta +
          # Soil Family:
            Aquolis_cmplx + Argiborolis_Pachic + Borohemists_cmplx + Bross +
            Bullwark + Bullwark_Cmplx + Catamount + Catamount_cmplx +
            Cathedral + Como + Cryaquepts_cmplx + Cryaquepts_Typic + Cryaquolls +
            Cryaquolls_cmplx + Cryaquolls_Typic + Cryaquolls_Typic_cmplx +
            Cryoborolis_cmplx + Cryorthents + Cryorthents_cmplx + Cryumbrepts +
            Cryumbrepts_cmplx + Gateview + Gothic + Granile + Haploborolis +
            Legault + Legault_cmplx + Leighcan + Leighcan_cmplx + Leighcan_warm +
            Moran + Ratake + Ratake_cmplx + Rogert + Supervisor_Limber_cmplx +
            Troutville + Unspecified + Vanet + Wetmore +
          # Soil Rock composition:
            Bouldery_ext + Rock_Land + Rock_Land_cmplx + Rock_Outcrop +
            Rock_Outcrop_cmplx + Rubbly + Stony + Stony_extreme + Stony_very +
            Till_Substratum ,
          data=forestTrain, family=binomial)
```

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

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

```r
SprFir_Ind_All_LogMod = SprFir_Ind_LogMod
save("SprFir_Ind_All_LogMod", file="SprFir_Ind_All_LogMod.Rdata")

#table(forestTrain$GeoName)
#table(forestTrain$Sed_mix)
#table(forestTrain$Gateview)
# above: Error in table(SpfFir_test$Gateview) : object 'SpfFir_train' not found <-------

SprFir_Ind_All_aic<-as.integer(SprFir_Ind_LogMod$aic)
SprFir_Ind_All_aic
```

```
## [1] 379731
```

```r
summary(SprFir_Ind_LogMod)
```

```
##
## Call:
## glm(formula = Spruce_Fir ~ Elev + Aspect + Slope + H2OHD + H2OVD +
##     RoadHD + FirePtHD + Shade9AM + Shade12PM + Shade3PM + RWwild +
##     NEwild + CMwild + CPwild + Montane_low + Montane + SubAlpine +
##     Alpine + Dry + Non_Dry + Alluvium + Glacial + Sed_mix + Ign_Meta +
##     Aquolis_cmplx + Argiborolis_Pachic + Borohemists_cmplx +
##     Bross + Bullwark + Bullwark_Cmplx + Catamount + Catamount_cmplx +
##     Cathedral + Como + Cryaquepts_cmplx + Cryaquepts_Typic +
##     Cryaquolls + Cryaquolls_cmplx + Cryaquolls_Typic + Cryaquolls_Typic_cmplx +
##     Cryoborolis_cmplx + Cryorthents + Cryorthents_cmplx + Cryumbrepts +
##     Cryumbrepts_cmplx + Gateview + Gothic + Granile + Haploborolis +
##     Legault + Legault_cmplx + Leighcan + Leighcan_cmplx + Leighcan_warm +
##     Moran + Ratake + Ratake_cmplx + Rogert + Supervisor_Limber_cmplx +
##     Troutville + Unspecified + Vanet + Wetmore + Bouldery_ext +
```

```
##     Rock_Land + Rock_Land_cmplx + Rock_Outcrop + Rock_Outcrop_cmplx +
##     Rubbly + Stony + Stony_extreme + Stony_very + Till_Substratum,
##     family = binomial, data = forestTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4789  -0.7616  -0.2849   0.8364   3.0845
##
## Coefficients: (19 not defined because of singularities)
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              1.126e+08  2.176e+10   0.005 0.995872
## Elev                     7.034e-03  3.758e-05 187.206  < 2e-16 ***
## Aspect                  -3.631e-04  4.853e-05  -7.482 7.33e-14 ***
## Slope                   -6.830e-03  1.579e-03  -4.326 1.52e-05 ***
## H2OHD                   -1.414e-03  2.491e-05 -56.775  < 2e-16 ***
## H2OVD                   -1.549e-03  9.356e-05 -16.558  < 2e-16 ***
## RoadHD                  -9.302e-05  3.195e-06 -29.114  < 2e-16 ***
## FirePtHD                -9.945e-06  3.535e-06  -2.813 0.004903 **
## Shade9AM                 1.130e-02  1.806e-03   6.255 3.99e-10 ***
## Shade12PM               -3.504e-02  1.485e-03 -23.595  < 2e-16 ***
## Shade3PM                 1.682e-02  1.479e-03  11.373  < 2e-16 ***
## RWwild                  -1.126e+08  2.176e+10  -0.005 0.995872
## NEwild                  -1.126e+08  2.176e+10  -0.005 0.995872
## CMwild                  -1.126e+08  2.176e+10  -0.005 0.995872
## CPwild                  -1.126e+08  2.176e+10  -0.005 0.995872
## Montane_low             -1.816e+11  8.138e+12  -0.022 0.982198
## Montane                 -6.897e+10  2.720e+11  -0.254 0.799855
## SubAlpine               -1.021e+00  1.972e-01  -5.178 2.24e-07 ***
## Alpine                  -9.278e-01  2.412e-01  -3.846 0.000120 ***
## Dry                      6.897e+10  2.720e+11   0.254 0.799855
## Non_Dry                  6.897e+10  2.720e+11   0.254 0.799855
## Alluvium                 2.393e+00  2.346e-01  10.201  < 2e-16 ***
## Glacial                  5.624e-01  2.571e-02  21.878  < 2e-16 ***
## Sed_mix                         NA         NA      NA       NA
## Ign_Meta                        NA         NA      NA       NA
## Aquolis_cmplx           -6.897e+10  2.720e+11  -0.254 0.799855
## Argiborolis_Pachic              NA         NA      NA       NA
## Borohemists_cmplx       -2.057e+00  2.200e-01  -9.351  < 2e-16 ***
## Bross                   -1.760e+00  3.628e-01  -4.852 1.22e-06 ***
## Bullwark                -4.952e-01  1.112e-01  -4.453 8.47e-06 ***
## Bullwark_Cmplx          -3.275e-01  1.412e-01  -2.319 0.020392 *
## Catamount               -3.717e-01  8.927e-02  -4.164 3.12e-05 ***
## Catamount_cmplx         -1.968e-02  2.497e-02  -0.788 0.430466
## Cathedral                1.126e+11  8.152e+12   0.014 0.988978
## Como                     4.005e-02  7.678e-02   0.522 0.601963
## Cryaquepts_cmplx        -1.460e+00  1.720e-01  -8.489  < 2e-16 ***
## Cryaquepts_Typic        -9.920e-01  2.540e-01  -3.906 9.38e-05 ***
## Cryaquolls              -1.549e+00  1.750e-01  -8.850  < 2e-16 ***
## Cryaquolls_cmplx        -1.665e+00  1.407e-01 -11.834  < 2e-16 ***
## Cryaquolls_Typic         5.798e-01  3.121e-01   1.857 0.063255 .
## Cryaquolls_Typic_cmplx  -2.939e-01  2.035e-02 -14.444  < 2e-16 ***
## Cryoborolis_cmplx               NA         NA      NA       NA
## Cryorthents            -1.594e+00  1.658e-01  -9.613  < 2e-16 ***
## Cryorthents_cmplx       -2.898e+01  2.379e+04  -0.001 0.999028
```

```
## Cryumbrepts                    NA          NA       NA        NA
## Cryumbrepts_cmplx              NA          NA       NA        NA
## Gateview                       NA          NA       NA        NA
## Gothic                 -2.557e+01  4.161e+04   -0.001 0.999510
## Granile                 9.682e-02  1.091e-01    0.888 0.374736
## Haploborolis            1.126e+11  8.152e+12    0.014 0.988978
## Legault                -1.478e+00  1.517e-01   -9.743  < 2e-16 ***
## Legault_cmplx                  NA          NA       NA        NA
## Leighcan                7.815e-01  7.327e-02   10.667  < 2e-16 ***
## Leighcan_cmplx          6.758e-01  1.138e-01    5.937 2.90e-09 ***
## Leighcan_warm           4.043e-01  1.152e-01    3.510 0.000449 ***
## Moran                          NA          NA       NA        NA
## Ratake                  1.126e+11  8.152e+12    0.014 0.988978
## Ratake_cmplx           -2.258e+01  4.824e+04    0.000 0.999627
## Rogert                         NA          NA       NA        NA
## Supervisor_Limber_cmplx        NA          NA       NA        NA
## Troutville                     NA          NA       NA        NA
## Unspecified            -6.897e+10  2.720e+11   -0.254 0.799855
## Vanet                   1.126e+11  8.152e+12    0.014 0.988978
## Wetmore                -1.097e+00  5.175e+04    0.000 0.999983
## Bouldery_ext                   NA          NA       NA        NA
## Rock_Land              -1.552e-01  1.987e-02   -7.811 5.68e-15 ***
## Rock_Land_cmplx         1.506e-01  1.079e-01    1.396 0.162618
## Rock_Outcrop                   NA          NA       NA        NA
## Rock_Outcrop_cmplx      3.170e-01  8.889e-02    3.567 0.000362 ***
## Rubbly                         NA          NA       NA        NA
## Stony                          NA          NA       NA        NA
## Stony_extreme                  NA          NA       NA        NA
## Stony_very                     NA          NA       NA        NA
## Till_Substratum                NA          NA       NA        NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 533620  on 406708  degrees of freedom
## Residual deviance: 379622  on 406654  degrees of freedom
## AIC: 379732
##
## Number of Fisher Scoring iterations: 25
```

```
curTime=Sys.time()
print(paste("Spruce_Fir Individual Logistic Model Calculation completed at",curTime))
```

```
## [1] "Spruce_Fir Individual Logistic Model Calculation completed at 2018-08-12 21:57:42"
```

```
#table(forestTest$Rock_Land)
# Get the following error with above code:
#  Error in table(SpfFir_test$Rock_Land) : object 'SpfFir_test' not found
#    Calls: <Anonymous> ... withCallingHandlers -> withVisible -> eval -> eval -> table
```

Predict Spruce and Fir Logistic Model Probabilities - All Aggregated Vars

Spruce and Fir Probabilities - All Aggregated Data

Predict the probability of Spruce and Fir for aggregated Data - all variables.

```
# Predict Spruce and Fir Agg Data - all variables

  SprFir_Agg_Train_predict= predict(SprFir_Agg_LogMod, type="response")
  SprFir_Agg_Train_Logit= predict(SprFir_Agg_LogMod)
  summary(SprFir_Agg_Train_predict)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1091  0.3330  0.3646  0.5991  0.9974
```

```
  str(SprFir_Agg_Train_predict)
```

```
##  Named num [1:406709] 0.0377 0.0371 0.0508 0.0422 0.0264 ...
##  - attr(*, "names")= chr [1:406709] "1" "2" "3" "4" ...
```

```
  #plot(table(SprFir_Agg_Train_predict))
  #plot(table(SprFir_Agg_Train_Logit))
  dens<-data.frame(table(SprFir_Agg_Train_predict))
# str(dens)

  SprFir_Agg_Test_predict= predict(SprFir_Agg_LogMod, type="response",newdata=forestTest)
```

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

```
  summary(SprFir_Agg_Test_predict)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1090  0.3349  0.3655  0.6003  0.9961
```

```
   str(SprFir_Agg_Test_predict)
```

```
##  Named num [1:174303] 0.0445 0.0451 0.0278 0.0178 0.0533 ...
##  - attr(*, "names")= chr [1:174303] "1" "2" "3" "4" ...
```

Spruce and Fir Probabilities - All Individuated Data

Predict the probability of Spruce and Fir for Individual Data - all variables.

```
  SprFir_Ind_Train_predict= predict(SprFir_Ind_LogMod, type="response")
  summary(SprFir_Ind_Train_predict)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1097  0.3332  0.3646  0.5986  0.9976
```

```
  SprFir_Ind_Test_predict= predict(SprFir_Ind_LogMod, type="response",newdata=forestTest)
```

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

```
  summary(SprFir_Ind_Test_predict)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1095  0.3351  0.3656  0.5995  0.9964
```

Spruce and Fir Receiver Operating Characteristic (ROC) - All Vars

Spruce and Fir Receiver ROC - All Aggregated Data

Next, look at the True Positive and False Positive rates based on threshold value for the aggregated data.

```
if (calcROC) {
  curTime=Sys.time()
  print(paste("ROC graph 1 started at",curTime))

  ROCpred_SprFir_Agg = prediction(SprFir_Agg_Train_predict, forestTrain$Spruce_Fir)
  summary(ROCpred_SprFir_Agg)
  ROCperf_SprFir_Agg = performance(ROCpred_SprFir_Agg, "tpr", "fpr")
  summary(ROCperf_SprFir_Agg)

  SprFir_Agg_All_ROC_AUC = as.numeric(performance(ROCpred_SprFir_Agg, "auc")@y.values)
  SprFir_Agg_All_ROC_AUC=as.integer(as.numeric(SprFir_Agg_All_ROC_AUC)*1000)/10
  print(paste("SprFir_Agg_All_ROC_AUC=",SprFir_Agg_All_ROC_AUC))

  jpeg(filename="Fig-ROCR_perf_SprFir_Agg.jpg")
  plot(ROCperf_SprFir_Agg, colorize=TRUE, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-0.2,1.7))
  dev.off()
} else {
  SprFir_Agg_All_ROC_AUC = 84.2
}
```

```
## [1] "ROC graph 1 started at 2018-08-12 21:57:49"
## [1] "SprFir_Agg_All_ROC_AUC= 84.2"
```

```
## pdf
##   2
```

Spruce and Fir Receiver ROC - All Individuated Data

The Response Operating Curve for the individuated data is shown below.

```
if (calcROC) {
  curTime=Sys.time()
  print(paste("ROCR graph 2 started at",curTime))

  ROCpred_SprFir_Ind = prediction(SprFir_Ind_Train_predict, forestTrain$Spruce_Fir)
  summary(ROCpred_SprFir_Ind)
  ROCperf_SprFir_Ind = performance(ROCpred_SprFir_Ind, "tpr", "fpr")
  summary(ROCperf_SprFir_Ind)

  SprFir_Ind_All_ROC_AUC = as.numeric(performance(ROCpred_SprFir_Ind, "auc")@y.values)
  SprFir_Ind_All_ROC_AUC=as.integer(as.numeric(SprFir_Ind_All_ROC_AUC)*1000)/10
  print(paste("SprFir_Ind_All_ROC_AUC=",SprFir_Ind_All_ROC_AUC))

  jpeg(filename="Fig-ROCR_perf_SprFir_Ind.jpg")
  plot(ROCperf_SprFir_Ind, colorize=TRUE, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-0.2,1.7))
  dev.off()
} else {
  SprFir_Ind_All_ROC_AUC = 84.2
}
```

```
## [1] "ROCR graph 2 started at 2018-08-12 22:01:18"
## [1] "SprFir_Ind_All_ROC_AUC= 84.2"
```

```
## pdf
```

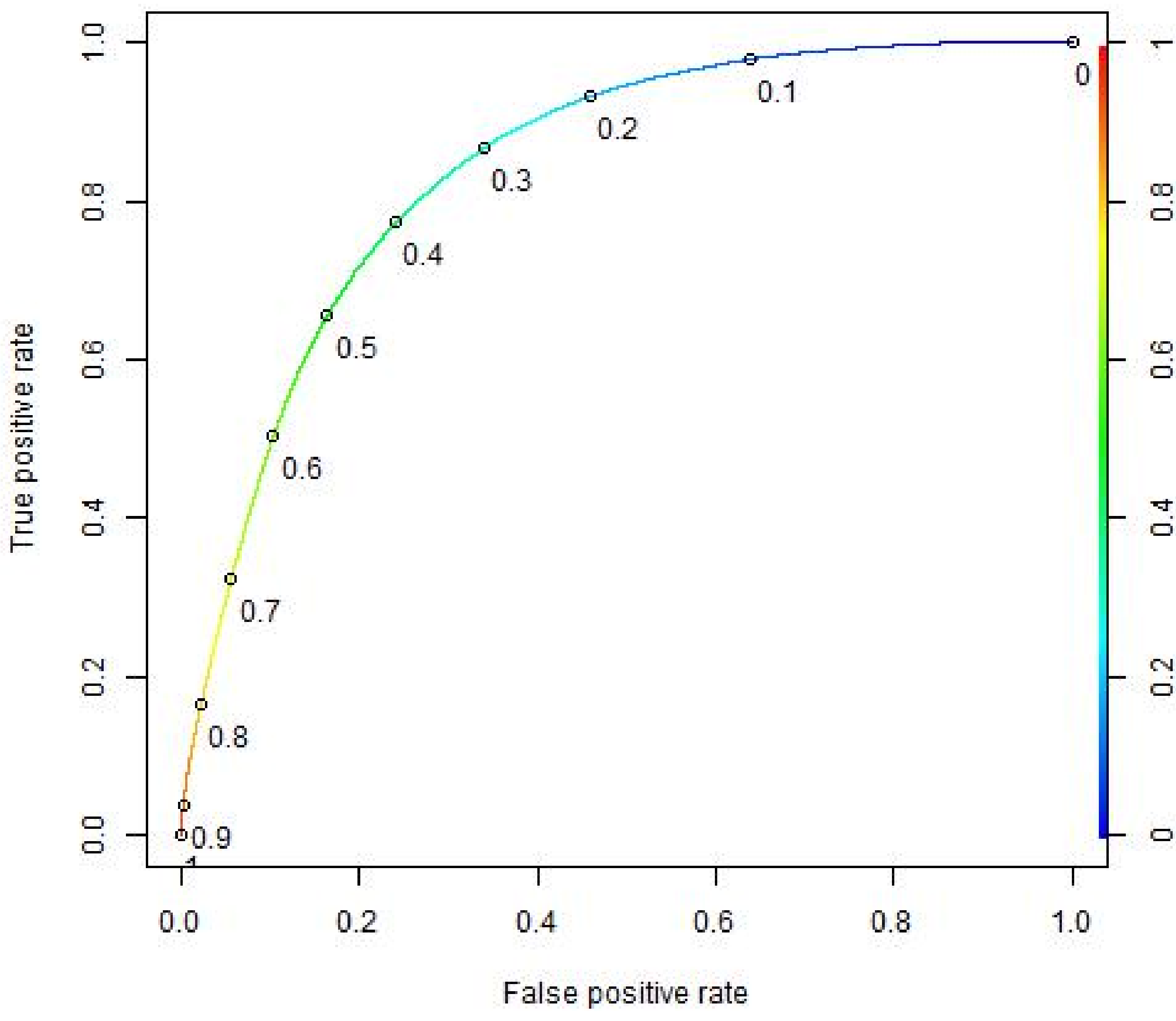

Figure 1: Spruce and Fir ROC for All Aggregated Data

```
## 2
```

The threshold graphs are essentially identical. This is making me think that there is not much difference between the two models. The AIC score for the Soil Type model is AIC: 351676 and for the individuated variables is: AIC: 351839. The Soil type model AIC score is 0.046% better than the individuated model.

```
curTime=Sys.time()
print(paste("ROCR graph 2 completed at",curTime))
```

```
## [1] "ROCR graph 2 completed at 2018-08-12 22:04:19"
```

Calculate Accuracy of Spruce and Fir Logisitic Models - All Vars

Calculate Spruce and Fir Aggregated Data Logisitic Model Accuracy - All Vars

Find best threshold for Spruce and Fir using all aggregated data.

```
result = calcLogisticModelAccuracy (forestTrain$Spruce_Fir, SprFir_Agg_Train_predict,
                       0.0, 1, 10, "Spruce_Fir", "Other", 1,1)
```

```
## [1] "Searching for threshold producing best Sensitivity_Specificity"
## [1] "start= 0 end= 1 inc= 0.1"
## [1] "Thresh=0, Accuracy=36.4%, BaseAcc(Other)=63.5%, Sens=100%, Spec=0%, Sens^2+Spec^2=-2"
## [1] "Thresh=0.1, Accuracy=58.6%, BaseAcc(Other)=63.5%, Sens=97.9%, Spec=36.1%, Sens^2+Spec^2=1.09"
## [1] "Thresh=0.2, Accuracy=68.3%, BaseAcc(Other)=63.5%, Sens=93.2%, Spec=54%, Sens^2+Spec^2=1.162"
## [1] "Thresh=0.3, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=86.6%, Spec=66%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.4, Accuracy=76.4%, BaseAcc(Other)=63.5%, Sens=77.3%, Spec=75.9%, Sens^2+Spec^2=1.174"
## [1] "Thresh=0.5, Accuracy=77%, BaseAcc(Other)=63.5%, Sens=65.6%, Spec=83.5%, Sens^2+Spec^2=1.129"
## [1] "Thresh=0.6, Accuracy=75.3%, BaseAcc(Other)=63.5%, Sens=50.3%, Spec=89.6%, Sens^2+Spec^2=1.058"
## [1] "Thresh=0.7, Accuracy=71.7%, BaseAcc(Other)=63.5%, Sens=32.2%, Spec=94.4%, Sens^2+Spec^2=0.995"
## [1] "Thresh=0.8, Accuracy=68.1%, BaseAcc(Other)=63.5%, Sens=16.4%, Spec=97.8%, Sens^2+Spec^2=0.983"
## [1] "Thresh=0.9, Accuracy=64.7%, BaseAcc(Other)=63.5%, Sens=3.8%, Spec=99.7%, Sens^2+Spec^2=0.996"
## [1] "Thresh=1, Accuracy=63.5%, BaseAcc(Other)=63.5%, Sens=0%, Spec=100%, Sens^2+Spec^2=-2"
## [1] "Best Sensitivity_Specificity threshold= 0.3 inc= 0.1"
## [1] "========================================"
## [1] "start= 0.2 end= 0.4 inc= 0.01"
## [1] "Thresh=0.2, Accuracy=68.3%, BaseAcc(Other)=63.5%, Sens=93.2%, Spec=54%, Sens^2+Spec^2=1.162"
## [1] "Thresh=0.21, Accuracy=69%, BaseAcc(Other)=63.5%, Sens=92.7%, Spec=55.4%, Sens^2+Spec^2=1.166"
## [1] "Thresh=0.22, Accuracy=69.6%, BaseAcc(Other)=63.5%, Sens=92.1%, Spec=56.7%, Sens^2+Spec^2=1.17"
## [1] "Thresh=0.23, Accuracy=70.2%, BaseAcc(Other)=63.5%, Sens=91.5%, Spec=57.9%, Sens^2+Spec^2=1.174"
## [1] "Thresh=0.24, Accuracy=70.7%, BaseAcc(Other)=63.5%, Sens=90.9%, Spec=59.2%, Sens^2+Spec^2=1.177"
## [1] "Thresh=0.25, Accuracy=71.3%, BaseAcc(Other)=63.5%, Sens=90.2%, Spec=60.4%, Sens^2+Spec^2=1.18"
## [1] "Thresh=0.26, Accuracy=71.8%, BaseAcc(Other)=63.5%, Sens=89.5%, Spec=61.6%, Sens^2+Spec^2=1.182"
## [1] "Thresh=0.27, Accuracy=72.3%, BaseAcc(Other)=63.5%, Sens=88.8%, Spec=62.8%, Sens^2+Spec^2=1.184"
## [1] "Thresh=0.28, Accuracy=72.8%, BaseAcc(Other)=63.5%, Sens=88.1%, Spec=63.9%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.29, Accuracy=73.2%, BaseAcc(Other)=63.5%, Sens=87.4%, Spec=65%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.3, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=86.6%, Spec=66%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.31, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=85.8%, Spec=67.1%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.32, Accuracy=74.3%, BaseAcc(Other)=63.5%, Sens=85%, Spec=68.1%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.33, Accuracy=74.6%, BaseAcc(Other)=63.5%, Sens=84.1%, Spec=69.1%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.34, Accuracy=74.9%, BaseAcc(Other)=63.5%, Sens=83.2%, Spec=70.1%, Sens^2+Spec^2=1.185"
## [1] "Thresh=0.35, Accuracy=75.2%, BaseAcc(Other)=63.5%, Sens=82.3%, Spec=71.2%, Sens^2+Spec^2=1.184"
## [1] "Thresh=0.36, Accuracy=75.5%, BaseAcc(Other)=63.5%, Sens=81.3%, Spec=72.1%, Sens^2+Spec^2=1.182"
## [1] "Thresh=0.37, Accuracy=75.7%, BaseAcc(Other)=63.5%, Sens=80.3%, Spec=73.1%, Sens^2+Spec^2=1.18"
## [1] "Thresh=0.38, Accuracy=76%, BaseAcc(Other)=63.5%, Sens=79.2%, Spec=74.1%, Sens^2+Spec^2=1.178"
## [1] "Thresh=0.39, Accuracy=76.2%, BaseAcc(Other)=63.5%, Sens=78.3%, Spec=75%, Sens^2+Spec^2=1.176"
```

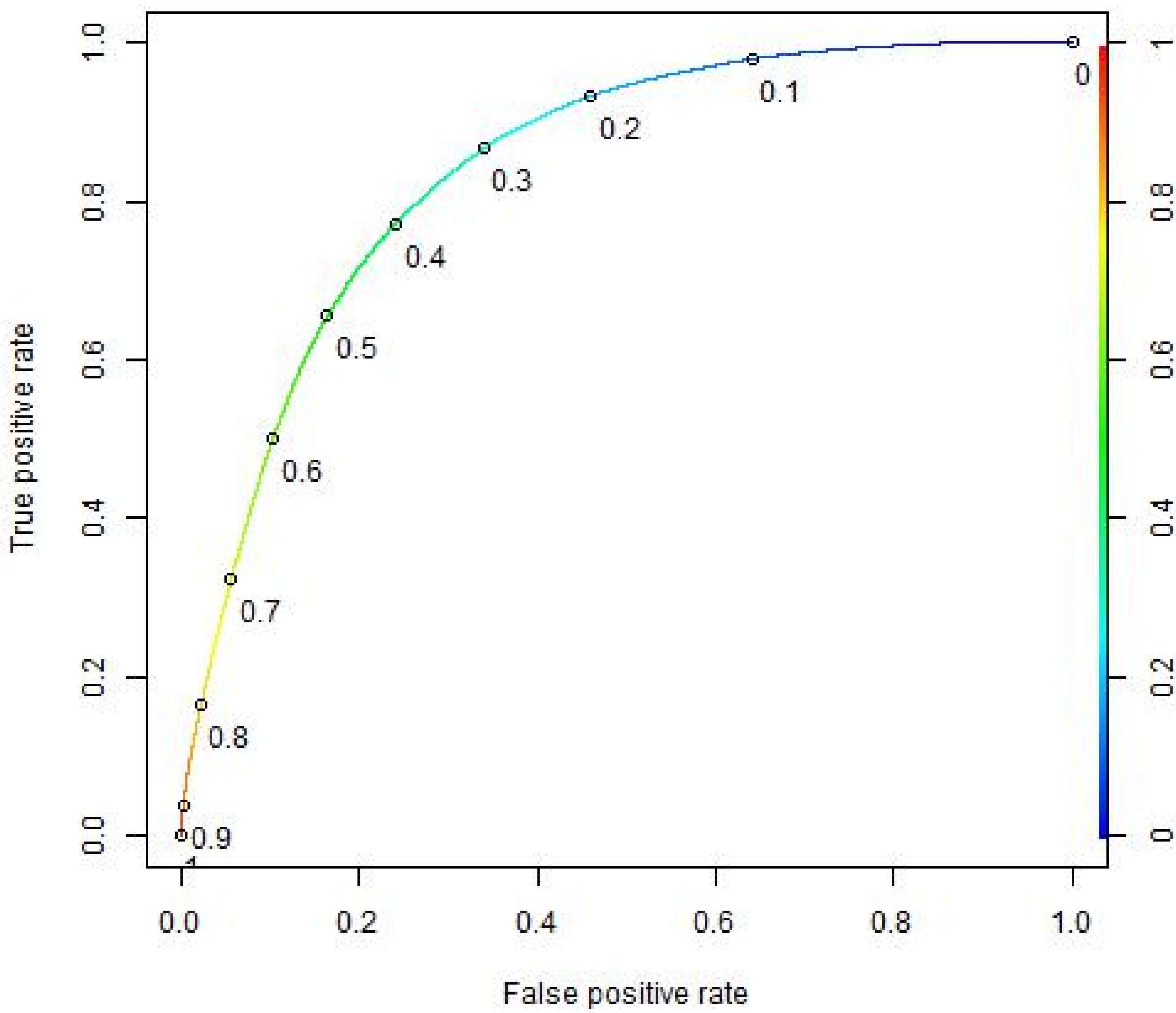

Figure 2: Spruce and Fir ROC for All Individuated Data

```
## [1] "Best Sensitivity_Specificity threshold= 0.3 inc= 0.01"
## [1] "========================================"
## [1] "start= 0.29 end= 0.31 inc= 0.001"
## [1] "Thresh=0.29, Accuracy=73.2%, BaseAcc(Other)=63.5%, Sens=87.4%, Spec=65%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.291, Accuracy=73.2%, BaseAcc(Other)=63.5%, Sens=87.3%, Spec=65.1%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.292, Accuracy=73.3%, BaseAcc(Other)=63.5%, Sens=87.3%, Spec=65.2%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.293, Accuracy=73.3%, BaseAcc(Other)=63.5%, Sens=87.2%, Spec=65.3%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.294, Accuracy=73.3%, BaseAcc(Other)=63.5%, Sens=87.1%, Spec=65.4%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.295, Accuracy=73.4%, BaseAcc(Other)=63.5%, Sens=87%, Spec=65.5%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.296, Accuracy=73.4%, BaseAcc(Other)=63.5%, Sens=87%, Spec=65.7%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.297, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=86.9%, Spec=65.8%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.298, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=86.8%, Spec=65.9%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.299, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=86.7%, Spec=66%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.3, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=86.6%, Spec=66%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.301, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=86.6%, Spec=66.1%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.302, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=86.5%, Spec=66.3%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.303, Accuracy=73.7%, BaseAcc(Other)=63.5%, Sens=86.4%, Spec=66.4%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.304, Accuracy=73.7%, BaseAcc(Other)=63.5%, Sens=86.3%, Spec=66.5%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.305, Accuracy=73.7%, BaseAcc(Other)=63.5%, Sens=86.2%, Spec=66.6%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.306, Accuracy=73.8%, BaseAcc(Other)=63.5%, Sens=86.1%, Spec=66.7%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.307, Accuracy=73.8%, BaseAcc(Other)=63.5%, Sens=86%, Spec=66.8%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.308, Accuracy=73.8%, BaseAcc(Other)=63.5%, Sens=86%, Spec=66.9%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.309, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=85.9%, Spec=67%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.31, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=85.8%, Spec=67.1%, Sens^2+Spec^2=1.187"
## [1] "========================================"
## [1] "Best Threshold=0.297"
## [1] "Best Sensitivity_Specificity=1.18880759315759"
```

```
curThresh = as.numeric(result[bestThreshIndex])
SprFir_Agg_All_threshold = curThresh
```

The accuracy for the best threshold on the training set for Spruce and Fir using all aggregated data is shown below.

```
result = calcLogisticModelAccuracy (forestTrain$Spruce_Fir, SprFir_Agg_Train_predict,
                       curThresh, curThresh, 1, "Spruce_Fir", "Other", 3)
```

```
## [1] "Model Performance for threshold= 0.297"
## [1] "predicted performance="
##                        Predicted
## Actual                  FALSE=Predict:Other TRUE=Predict:Spruce_Fir
##   0=Actual:Other            170079 (TN)         88342 (FP)
##   1=Actual:Spruce_Fir       19384 (FN)          128904 (TP)
## [1] "Sensitivity= 0.869281398359948 (True positive rate of Spruce_Fir = TP/(TP+FN) = 128904 /( 12890
## [1] "Specificity= 0.658146977219344 (True negative rate of Other = TN/(TN+FP) = 170079 /( 170079 + 88
## [1] "Sens^2+Spec^2=1.188"
## [1] "Baseline (Other) Accuracy=0.635395"
## [1] "Logistic Accuracy=0.735127"
```

The accuracy for the best threshold on the testing set for Spruce and Fir using all aggregated data is shown below.

```
result = calcLogisticModelAccuracy (forestTest$Spruce_Fir, SprFir_Agg_Test_predict,
                       curThresh, curThresh, 1, "Spruce_Fir", "Other", 3,
                       saveFile=saveFileName, desc="Spruce/Fir All Aggregate Vars",
                       AIC=SprFir_Agg_All_aic, AUC=SprFir_Agg_All_ROC_AUC)
```

```
## [1] "Model Performance for threshold= 0.297"
## [1] "predicted performance="
##                      Predicted
## Actual                FALSE=Predict:Other TRUE=Predict:Spruce_Fir
##   0=Actual:Other          72912 (TN)          37839 (FP)
##   1=Actual:Spruce_Fir     7991 (FN)           55561 (TP)
## [1] "Sensitivity= 0.874260448136959 (True positive rate of Spruce_Fir = TP/(TP+FN) = 55561 /( 55561 +
## [1] "Specificity= 0.658341685402389 (True negative rate of Other = TN/(TN+FP) = 72912 /( 72912 + 3783
## [1] "Sens^2+Spec^2=1.197"
## [1] "Baseline (Other) Accuracy=0.635393"
## [1] "Logistic Accuracy=0.737067"
```

```
# retVal = c(modelPerformance, sensitivity,specificity) # TN, FN, FP, TP, sens, spec
# c(funcStat,accuracy,baseline,retVal)
list[RC, SprFir_Agg_All_model_acc, SprFir_Agg_All_baseline_acc,
    TN, FN, FP, TP, SprFir_Agg_All_sens, SprFir_Agg_All_spec] <- result
if (RC != "OK") {
  print(paste("Error - terminating:",RC))
  knitr:knit_exit()
}
SprFir_Agg_All_model_acc = as.integer(as.numeric(SprFir_Agg_All_model_acc)*1000)/10
SprFir_Agg_All_baseline_acc = as.integer(as.numeric(SprFir_Agg_All_baseline_acc)*1000)/10
SprFir_Agg_All_sens = as.integer(as.numeric(SprFir_Agg_All_sens)*1000)/10
SprFir_Agg_All_spec = as.integer(as.numeric(SprFir_Agg_All_spec)*1000)/10
```

Calculate Spruce and Fir Individuated Data Logisitic Model Accuracy - All Vars

Find best threshold for Spruce and Fir using all individuated data.

```
result = calcLogisticModelAccuracy (forestTrain$Spruce_Fir, SprFir_Ind_Train_predict,
                        0.0, 1, 10, "Spruce_Fir", "Other", 1,1)
```

```
## [1] "Searching for threshold producing best Sensitivity_Specificity"
## [1] "start= 0 end= 1 inc= 0.1"
## [1] "Thresh=0, Accuracy=36.4%, BaseAcc(Other)=63.5%, Sens=100%, Spec=0%, Sens^2+Spec^2=-2"
## [1] "Thresh=0.1, Accuracy=58.5%, BaseAcc(Other)=63.5%, Sens=97.9%, Spec=36%, Sens^2+Spec^2=1.089"
## [1] "Thresh=0.2, Accuracy=68.3%, BaseAcc(Other)=63.5%, Sens=93.3%, Spec=54%, Sens^2+Spec^2=1.162"
## [1] "Thresh=0.3, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=86.7%, Spec=66%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.4, Accuracy=76.4%, BaseAcc(Other)=63.5%, Sens=77.2%, Spec=75.9%, Sens^2+Spec^2=1.173"
## [1] "Thresh=0.5, Accuracy=77%, BaseAcc(Other)=63.5%, Sens=65.5%, Spec=83.5%, Sens^2+Spec^2=1.128"
## [1] "Thresh=0.6, Accuracy=75.3%, BaseAcc(Other)=63.5%, Sens=50.2%, Spec=89.7%, Sens^2+Spec^2=1.057"
## [1] "Thresh=0.7, Accuracy=71.7%, BaseAcc(Other)=63.5%, Sens=32.2%, Spec=94.4%, Sens^2+Spec^2=0.995"
## [1] "Thresh=0.8, Accuracy=68.1%, BaseAcc(Other)=63.5%, Sens=16.5%, Spec=97.8%, Sens^2+Spec^2=0.983"
## [1] "Thresh=0.9, Accuracy=64.8%, BaseAcc(Other)=63.5%, Sens=3.9%, Spec=99.7%, Sens^2+Spec^2=0.996"
## [1] "Thresh=1, Accuracy=63.5%, BaseAcc(Other)=63.5%, Sens=0%, Spec=100%, Sens^2+Spec^2=-2"
## [1] "Best Sensitivity_Specificity threshold= 0.3 inc= 0.1"
## [1] "========================================"
## [1] "start= 0.2 end= 0.4 inc= 0.01"
## [1] "Thresh=0.2, Accuracy=68.3%, BaseAcc(Other)=63.5%, Sens=93.3%, Spec=54%, Sens^2+Spec^2=1.162"
## [1] "Thresh=0.21, Accuracy=68.9%, BaseAcc(Other)=63.5%, Sens=92.7%, Spec=55.3%, Sens^2+Spec^2=1.166"
## [1] "Thresh=0.22, Accuracy=69.6%, BaseAcc(Other)=63.5%, Sens=92.1%, Spec=56.6%, Sens^2+Spec^2=1.17"
## [1] "Thresh=0.23, Accuracy=70.1%, BaseAcc(Other)=63.5%, Sens=91.5%, Spec=57.9%, Sens^2+Spec^2=1.174"
## [1] "Thresh=0.24, Accuracy=70.7%, BaseAcc(Other)=63.5%, Sens=90.9%, Spec=59.2%, Sens^2+Spec^2=1.177"
## [1] "Thresh=0.25, Accuracy=71.3%, BaseAcc(Other)=63.5%, Sens=90.2%, Spec=60.4%, Sens^2+Spec^2=1.179"
## [1] "Thresh=0.26, Accuracy=71.8%, BaseAcc(Other)=63.5%, Sens=89.5%, Spec=61.6%, Sens^2+Spec^2=1.182"
```

```
## [1] "Thresh=0.27, Accuracy=72.3%, BaseAcc(Other)=63.5%, Sens=88.8%, Spec=62.8%, Sens^2+Spec^2=1.184"
## [1] "Thresh=0.28, Accuracy=72.7%, BaseAcc(Other)=63.5%, Sens=88.2%, Spec=63.9%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.29, Accuracy=73.2%, BaseAcc(Other)=63.5%, Sens=87.4%, Spec=65%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.3, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=86.7%, Spec=66%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.31, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=85.8%, Spec=67.1%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.32, Accuracy=74.2%, BaseAcc(Other)=63.5%, Sens=85%, Spec=68.1%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.33, Accuracy=74.6%, BaseAcc(Other)=63.5%, Sens=84.1%, Spec=69.1%, Sens^2+Spec^2=1.186"
## [1] "Thresh=0.34, Accuracy=74.9%, BaseAcc(Other)=63.5%, Sens=83.2%, Spec=70.1%, Sens^2+Spec^2=1.184"
## [1] "Thresh=0.35, Accuracy=75.2%, BaseAcc(Other)=63.5%, Sens=82.2%, Spec=71.1%, Sens^2+Spec^2=1.183"
## [1] "Thresh=0.36, Accuracy=75.4%, BaseAcc(Other)=63.5%, Sens=81.2%, Spec=72.1%, Sens^2+Spec^2=1.181"
## [1] "Thresh=0.37, Accuracy=75.7%, BaseAcc(Other)=63.5%, Sens=80.2%, Spec=73.1%, Sens^2+Spec^2=1.178"
## [1] "Thresh=0.38, Accuracy=75.9%, BaseAcc(Other)=63.5%, Sens=79.2%, Spec=74%, Sens^2+Spec^2=1.176"
## [1] "Thresh=0.39, Accuracy=76.2%, BaseAcc(Other)=63.5%, Sens=78.2%, Spec=75%, Sens^2+Spec^2=1.175"
## [1] "Best Sensitivity_Specificity threshold= 0.3 inc= 0.01"
## [1] "========================================"
## [1] "start= 0.29 end= 0.31 inc= 0.001"
## [1] "Thresh=0.29, Accuracy=73.2%, BaseAcc(Other)=63.5%, Sens=87.4%, Spec=65%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.291, Accuracy=73.2%, BaseAcc(Other)=63.5%, Sens=87.4%, Spec=65.1%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.292, Accuracy=73.2%, BaseAcc(Other)=63.5%, Sens=87.3%, Spec=65.2%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.293, Accuracy=73.3%, BaseAcc(Other)=63.5%, Sens=87.2%, Spec=65.3%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.294, Accuracy=73.3%, BaseAcc(Other)=63.5%, Sens=87.1%, Spec=65.4%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.295, Accuracy=73.4%, BaseAcc(Other)=63.5%, Sens=87%, Spec=65.5%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.296, Accuracy=73.4%, BaseAcc(Other)=63.5%, Sens=87%, Spec=65.6%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.297, Accuracy=73.4%, BaseAcc(Other)=63.5%, Sens=86.9%, Spec=65.7%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.298, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=86.8%, Spec=65.8%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.299, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=86.7%, Spec=65.9%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.3, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=86.7%, Spec=66%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.301, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=86.6%, Spec=66.1%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.302, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=86.5%, Spec=66.2%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.303, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=86.4%, Spec=66.3%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.304, Accuracy=73.7%, BaseAcc(Other)=63.5%, Sens=86.3%, Spec=66.4%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.305, Accuracy=73.7%, BaseAcc(Other)=63.5%, Sens=86.2%, Spec=66.6%, Sens^2+Spec^2=1.188
## [1] "Thresh=0.306, Accuracy=73.8%, BaseAcc(Other)=63.5%, Sens=86.2%, Spec=66.6%, Sens^2+Spec^2=1.187
## [1] "Thresh=0.307, Accuracy=73.8%, BaseAcc(Other)=63.5%, Sens=86.1%, Spec=66.8%, Sens^2+Spec^2=1.187
## [1] "Thresh=0.308, Accuracy=73.8%, BaseAcc(Other)=63.5%, Sens=86%, Spec=66.9%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.309, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=85.9%, Spec=67%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.31, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=85.8%, Spec=67.1%, Sens^2+Spec^2=1.187"
## [1] "========================================"
## [1] "Best Threshold=0.297"
## [1] "Best Sensitivity_Specificity=1.18842689723771"
```

```r
curThresh = as.numeric(result[bestThreshIndex])
SprFir_Ind_All_threshold = curThresh
```

The accuracy for the best threshold on the training set for Spruce and Fir using all individuated data is shown below.

```r
result = calcLogisticModelAccuracy (forestTrain$Spruce_Fir, SprFir_Ind_Train_predict,
                        curThresh, curThresh, 1, "Spruce_Fir", "Other", 3)
```

```
## [1] "Model Performance for threshold= 0.297"
## [1] "predicted performance="
##                      Predicted
## Actual                FALSE=Predict:Other TRUE=Predict:Spruce_Fir
##   0=Actual:Other          169972 (TN)         88449 (FP)
```

```
##   1=Actual:Spruce_Fir     19370 (FN)          128918 (TP)
## [1] "Sensitivity= 0.869375809236081 (True positive rate of Spruce_Fir = TP/(TP+FN) = 128918 /( 12891
## [1] "Specificity= 0.657732924181858 (True negative rate of Other = TN/(TN+FP) = 169972 /( 169972 + 8
## [1] "Sens^2+Spec^2=1.188"
## [1] "Baseline (Other) Accuracy=0.635395"
## [1] "Logistic Accuracy=0.734898"
```

The accuracy for the best threshold on the testing set for Spruce and Fir using all individuated data is shown below.

```
result = calcLogisticModelAccuracy (forestTest$Spruce_Fir, SprFir_Ind_Test_predict,
                       curThresh, curThresh, 1, "Spruce_Fir", "Other", 3,
                       saveFile=saveFileName, desc="Spruce/Fir All Individualized Vars",
                       AIC=SprFir_Ind_All_aic, AUC=SprFir_Ind_All_ROC_AUC)
```

```
## [1] "Model Performance for threshold= 0.297"
## [1] "predicted performance="
##                       Predicted
## Actual                 FALSE=Predict:Other TRUE=Predict:Spruce_Fir
##   0=Actual:Other          72876 (TN)          37875 (FP)
##   1=Actual:Spruce_Fir     7974 (FN)           55578 (TP)
## [1] "Sensitivity= 0.874527945619335 (True positive rate of Spruce_Fir = TP/(TP+FN) = 55578 /( 55578 
## [1] "Specificity= 0.658016631904001 (True negative rate of Other = TN/(TN+FP) = 72876 /( 72876 + 378
## [1] "Sens^2+Spec^2=1.197"
## [1] "Baseline (Other) Accuracy=0.635393"
## [1] "Logistic Accuracy=0.736958"
```

```
list[RC, SprFir_Ind_All_model_acc, SprFir_Ind_All_baseline_acc,
      TN, FN, FP, TP, SprFir_Ind_All_sens, SprFir_Ind_All_spec] <- result
  if (RC != "OK") {
    print(paste("Error - terminating:",RC))
    knitr:knit_exit()
  }
  SprFir_Ind_All_model_acc = as.integer(as.numeric(SprFir_Ind_All_model_acc)*1000)/10
  SprFir_Ind_All_baseline_acc = as.integer(as.numeric(SprFir_Ind_All_baseline_acc)*1000)/10
  SprFir_Ind_All_sens = as.integer(as.numeric(SprFir_Ind_All_sens)*1000)/10
  SprFir_Ind_All_spec = as.integer(as.numeric(SprFir_Ind_All_spec)*1000)/10
```

The Spruce and Fir aggregated model accuracy on the test data is 77.15% compared to 77.12% for the individuated data model, essentially identical. Both are ~ 14% better than the baseline model.

Spruce and Fir Logistic Regression - Significant Variables

Create Spruce and Fir Logistic Model - Sig Vars

Now create the logistic model for the Aggregated Soil data using just the significant variables and compare to the previous models.

Spruce and Fir Logistic Model using Significant Aggregated Data

Variables that have been removed are commented out in the code below.

```
  SprFir_Agg_LogMod =
    glm(Spruce_Fir ~
          Elev +      # Elevation in meters of cell
```

```
          Aspect +   # Direction in degrees slope faces
          Slope +    # Slope / steepness of hill in degrees (0 to 90)
          H2OHD +    # Horizontal distance in meters to nearest water
          H2OVD +    # Vertical distance in meters to nearest water
          RoadHD +   # Horizontal distance in meters to nearest road
          FirePtHD + # Horizontal distance in meters to nearest fire point
          Shade9AM + Shade12PM + Shade3PM + # Amount of shade at 9am, 12pm and 3pm
          # Wilderness areas:
            # RWwild + NEwild + CMwild + CPwild +
          # Aggregated Soil type:
            # ST01 + ST02 + ST03 +
            ST04 +
            # ST05 + ST06 + ST07 +
            ST08 + ST09 + ST10 + ST11 + ST12 +
            # ST13 + ST14 + ST15 +
            ST16 + ST17 + ST18 + ST19 + ST20 +
            ST21 + ST22 + ST23 + ST24 + ST25 + ST26 + ST27 + ST28 + ST29 + ST30 +
            ST31 + ST32 + ST33 +
            # ST34 + ST35 +
            ST36 +
            # ST37 +
            ST38 + ST39 ,
            # + ST40 ,
          data=forestTrain, family=binomial)

SprFir_Agg_Sig_LogMod = SprFir_Agg_LogMod
save("SprFir_Agg_Sig_LogMod", file="SprFir_Agg_Sig_LogMod.Rdata")

SprFir_Agg_Sig_aic<-as.integer(SprFir_Agg_LogMod$aic)
SprFir_Agg_Sig_aic
```

```
## [1] 384126
```

Check the coefficients of the Spruce and Fir model using significant aggregated data.

```
summary(SprFir_Agg_LogMod)
```

```
##
## Call:
## glm(formula = Spruce_Fir ~ Elev + Aspect + Slope + H2OHD + H2OVD +
##     RoadHD + FirePtHD + Shade9AM + Shade12PM + Shade3PM + ST04 +
##     ST08 + ST09 + ST10 + ST11 + ST12 + ST16 + ST17 + ST18 + ST19 +
##     ST20 + ST21 + ST22 + ST23 + ST24 + ST25 + ST26 + ST27 + ST28 +
##     ST29 + ST30 + ST31 + ST32 + ST33 + ST36 + ST38 + ST39, family = binomial,
##     data = forestTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3322  -0.7740  -0.2732   0.8484   3.0533
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.714e+01  2.905e-01 -59.017  < 2e-16 ***
## Elev         6.465e-03  3.030e-05 213.401  < 2e-16 ***
## Aspect      -5.464e-04  4.825e-05 -11.324  < 2e-16 ***
```

```
## Slope       -8.028e-03  1.488e-03  -5.396 6.81e-08 ***
## H2OHD       -1.419e-03  2.468e-05 -57.489  < 2e-16 ***
## H2OVD       -1.417e-03  9.354e-05 -15.150  < 2e-16 ***
## RoadHD      -3.439e-05  2.907e-06 -11.831  < 2e-16 ***
## FirePtHD    -9.349e-06  3.435e-06  -2.722 0.006494 **
## Shade9AM     6.014e-03  1.675e-03   3.591 0.000329 ***
## Shade12PM   -2.901e-02  1.378e-03 -21.055  < 2e-16 ***
## Shade3PM     1.196e-02  1.369e-03   8.734  < 2e-16 ***
## ST04         1.120e-01  9.529e-02   1.176 0.239768
## ST08         1.747e+00  1.970e-01   8.867  < 2e-16 ***
## ST09         2.680e+00  1.046e-01  25.614  < 2e-16 ***
## ST10         7.718e-01  4.726e-02  16.333  < 2e-16 ***
## ST11         8.323e-01  5.205e-02  15.991  < 2e-16 ***
## ST12         8.084e-01  3.473e-02  23.277  < 2e-16 ***
## ST16         1.828e+00  6.470e-02  28.254  < 2e-16 ***
## ST17         1.216e+00  8.861e-02  13.723  < 2e-16 ***
## ST18         8.469e-01  1.795e-01   4.717 2.39e-06 ***
## ST19         1.932e+00  4.648e-02  41.561  < 2e-16 ***
## ST20         2.166e+00  3.658e-02  59.219  < 2e-16 ***
## ST21         4.485e+00  2.038e-01  22.009  < 2e-16 ***
## ST22         2.353e+00  2.655e-02  88.629  < 2e-16 ***
## ST23         2.019e+00  2.444e-02  82.645  < 2e-16 ***
## ST24         1.587e+00  2.787e-02  56.956  < 2e-16 ***
## ST25        -1.922e-01  1.329e-01  -1.446 0.148311
## ST26         6.971e-01  8.128e-02   8.577  < 2e-16 ***
## ST27         1.865e+00  7.793e-02  23.936  < 2e-16 ***
## ST28         1.324e-01  2.116e-01   0.626 0.531570
## ST29         1.319e+00  2.313e-02  57.041  < 2e-16 ***
## ST30         1.153e+00  2.794e-02  41.270  < 2e-16 ***
## ST31         1.504e+00  2.665e-02  56.418  < 2e-16 ***
## ST32         1.050e+00  2.382e-02  44.085  < 2e-16 ***
## ST33         1.423e+00  2.436e-02  58.421  < 2e-16 ***
## ST36        -1.018e+00  3.239e-01  -3.144 0.001669 **
## ST38         2.847e-01  2.878e-02   9.892  < 2e-16 ***
## ST39         2.910e-01  2.972e-02   9.791  < 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: 533620  on 406708  degrees of freedom
## Residual deviance: 384051  on 406671  degrees of freedom
## AIC: 384127
##
## Number of Fisher Scoring iterations: 6
```

The intercept looks much more reasonable. Some soil types that were significant previously are no longer significant.

Spruce and Fir Logistic Model using Significant Individuated Data

Create a logistic model for the significant individuated variables.

Again, the non-significant variables have been commented out.

```r
SprFir_Ind_LogMod =
  glm(Spruce_Fir ~
        Elev +      # Elevation in meters of cell
        Aspect +    # Direction in degrees slope faces
        Slope +     # Slope / steepness of hill in degrees (0 to 90)
        H2OHD +     # Horizontal distance in meters to nearest water
        H2OVD +     # Vertical distance in meters to nearest water
        RoadHD +    # Horizontal distance in meters to nearest road
        FirePtHD + # Horizontal distance in meters to nearest fire point
        Shade9AM + Shade12PM + Shade3PM + # Amount of shade at 9am, 12pm and 3pm
        # Wilderness areas:
          # RWwild + NEwild + CMwild + CPwild +
        # Climate Zone:
        # ClimateName +
          # Montane_low + Montane +
          SubAlpine + Alpine +
          # Dry + Non_Dry +
        # Geology Zone:
        # GeoName +
          Alluvium + Glacial +
          # Sed_mix + Ign_Meta +
        # Soil Family:
          Aquolis_cmplx +
          # Argiborolis_Pachic +
          Borohemists_cmplx + Bross +
          Bullwark + Bullwark_Cmplx + Catamount + Catamount_cmplx +
          # Cathedral + Como +
          Cryaquepts_cmplx + Cryaquepts_Typic + Cryaquolls +
          Cryaquolls_cmplx + Cryaquolls_Typic + Cryaquolls_Typic_cmplx +
          # Cryoborolis_cmplx +
          Cryorthents +
          # Cryorthents_cmplx + Cryumbrepts + Cryumbrepts_cmplx + Gateview +
          # Gothic + Granile + Haploborolis +
          Legault +
          # Legault_cmplx +
          Leighcan + Leighcan_cmplx + Leighcan_warm +
          # Moran + Ratake + Ratake_cmplx + Rogert + Supervisor_Limber_cmplx +
          # Troutville + Unspecified + Vanet + Wetmore +
        # Soil Rock composition:
          # Bouldery_ext +
          Rock_Land +
          # Rock_Land_cmplx + Rock_Outcrop +
          Rock_Outcrop_cmplx ,
          # Rubbly + Stony + Stony_extreme + Stony_very + Till_Substratum ,
        data=forestTrain, family=binomial)

SprFir_Ind_Sig_LogMod = SprFir_Ind_LogMod
save("SprFir_Ind_Sig_LogMod", file="SprFir_Ind_Sig_LogMod.Rdata")

SprFir_Ind_Sig_aic<-as.integer(SprFir_Ind_LogMod$aic)
SprFir_Ind_Sig_aic
```

```
## [1] 384376
```

```r
summary(SprFir_Ind_LogMod)
```

```
##
## Call:
## glm(formula = Spruce_Fir ~ Elev + Aspect + Slope + H2OHD + H2OVD +
##     RoadHD + FirePtHD + Shade9AM + Shade12PM + Shade3PM + SubAlpine +
##     Alpine + Alluvium + Glacial + Aquolis_cmplx + Borohemists_cmplx +
##     Bross + Bullwark + Bullwark_Cmplx + Catamount + Catamount_cmplx +
##     Cryaquepts_cmplx + Cryaquepts_Typic + Cryaquolls + Cryaquolls_cmplx +
##     Cryaquolls_Typic + Cryaquolls_Typic_cmplx + Cryorthents +
##     Legault + Leighcan + Leighcan_cmplx + Leighcan_warm + Rock_Land +
##     Rock_Outcrop_cmplx, family = binomial, data = forestTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4732  -0.7688  -0.2853   0.8441   3.0002
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -1.768e+01  2.994e-01 -59.061  < 2e-16 ***
## Elev                    6.674e-03  3.204e-05 208.285  < 2e-16 ***
## Aspect                 -5.010e-04  4.824e-05 -10.386  < 2e-16 ***
## Slope                  -1.054e-02  1.512e-03  -6.973 3.09e-12 ***
## H2OHD                  -1.458e-03  2.477e-05 -58.855  < 2e-16 ***
## H2OVD                  -1.398e-03  9.282e-05 -15.061  < 2e-16 ***
## RoadHD                 -2.898e-05  2.906e-06  -9.970  < 2e-16 ***
## FirePtHD               -4.146e-06  3.442e-06  -1.204  0.22841
## Shade9AM                8.457e-03  1.705e-03   4.961 7.03e-07 ***
## Shade12PM              -3.267e-02  1.405e-03 -23.259  < 2e-16 ***
## Shade3PM                1.388e-02  1.395e-03   9.947  < 2e-16 ***
## SubAlpine               1.219e+00  6.783e-02  17.976  < 2e-16 ***
## Alpine                  9.605e-02  1.024e-01   0.938  0.34843
## Alluvium                2.633e-01  1.186e-01   2.220  0.02643 *
## Glacial                 7.511e-01  2.419e-02  31.046  < 2e-16 ***
## Aquolis_cmplx          -7.585e+00  2.507e+01  -0.303  0.76227
## Borohemists_cmplx      -2.256e+00  2.077e-01 -10.859  < 2e-16 ***
## Bross                  -1.060e+00  3.333e-01  -3.179  0.00148 **
## Bullwark                1.163e+00  7.621e-02  15.258  < 2e-16 ***
## Bullwark_Cmplx          1.478e+00  8.892e-02  16.622  < 2e-16 ***
## Catamount              -6.155e-01  3.137e-02 -19.618  < 2e-16 ***
## Catamount_cmplx        -1.329e-01  2.307e-02  -5.758 8.54e-09 ***
## Cryaquepts_cmplx       -6.044e-01  9.684e-02  -6.242 4.33e-10 ***
## Cryaquepts_Typic        9.800e-01  1.231e-01   7.962 1.69e-15 ***
## Cryaquolls              2.972e-01  1.325e-01   2.243  0.02493 *
## Cryaquolls_cmplx       -2.685e-01  8.212e-02  -3.269  0.00108 **
## Cryaquolls_Typic        2.620e+00  2.368e-01  11.064  < 2e-16 ***
## Cryaquolls_Typic_cmplx -3.532e-01  2.019e-02 -17.493  < 2e-16 ***
## Cryorthents            -4.157e-01  7.584e-02  -5.481 4.22e-08 ***
## Legault                 7.610e-01  7.114e-02  10.697  < 2e-16 ***
## Leighcan                3.352e-01  1.912e-02  17.534  < 2e-16 ***
## Leighcan_cmplx          4.070e-01  2.943e-02  13.830  < 2e-16 ***
## Leighcan_warm          -2.688e-01  6.856e-02  -3.921 8.84e-05 ***
## Rock_Land              -1.191e-01  1.964e-02  -6.066 1.31e-09 ***
## Rock_Outcrop_cmplx      4.091e-01  3.357e-02  12.189  < 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: 533620  on 406708  degrees of freedom
## Residual deviance: 384307  on 406674  degrees of freedom
## AIC: 384377
##
## Number of Fisher Scoring iterations: 12
```

Again the intercept looks much better. Also a few variables have become non-significant.

Predict Spruce and Fir Logistic Model Probabilities - Sig Vars

Spruce and Fir Probabilities using Significant Aggregated Data

Predict the probability of Spruce and Fir for aggregated Data - significant variables.

```r
# Predict Spruce and Fir Agg Data - significant variables

  SprFir_Agg_Train_predict= predict(SprFir_Agg_LogMod, type="response")
  summary(SprFir_Agg_Train_predict)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000311 0.1067035 0.3369070 0.3646047 0.5954624 0.9961201
```

```r
  SprFir_Agg_Test_predict= predict(SprFir_Agg_LogMod, type="response",newdata=forestTest)
  summary(SprFir_Agg_Test_predict)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000221 0.1061770 0.3389619 0.3654279 0.5962905 0.9950011
```

Spruce and Fir Probabilities using Significant Individuated Data

Predict the probability of Spruce_Fir using significant Individuated Data.

```r
  SprFir_Ind_Train_predict= predict(SprFir_Ind_LogMod, type="response")
  summary(SprFir_Ind_Train_predict)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000001 0.1103242 0.3337354 0.3646047 0.5942093 0.9975981
```

```r
  SprFir_Ind_Test_predict= predict(SprFir_Ind_LogMod, type="response",newdata=forestTest)
  summary(SprFir_Ind_Test_predict)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000001 0.1099858 0.3355733 0.3654766 0.5947334 0.9965586
```

```r
  print(paste("ROCR graph 2 completed at",curTime))
```

```
## [1] "ROCR graph 2 completed at 2018-08-12 22:04:19"
```

Spruce and Fir Receiver Operating Characteristic (ROC) - Sig Vars

Look at the True Positive and False Positive rates based on threshold value.

```r
if (calcROC) {
  ROCpred_SprFir_Agg = prediction(SprFir_Agg_Train_predict, forestTrain$Spruce_Fir)
  summary(ROCpred_SprFir_Agg)

  ROCperf_SprFir_Agg = performance(ROCpred_SprFir_Agg, "tpr", "fpr")
  summary(ROCperf_SprFir_Agg)

  SprFir_Agg_Sig_ROC_AUC = as.numeric(performance(ROCpred_SprFir_Agg, "auc")@y.values)
  SprFir_Agg_Sig_ROC_AUC=as.integer(as.numeric(SprFir_Agg_Sig_ROC_AUC)*1000)/10
  SprFir_Agg_Sig_ROC_AUC

  jpeg(filename="Fig-ROCR_perf_SprFir_Agg_Sig.jpg")
  plot(ROCperf_SprFir_Agg, colorize=TRUE, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-0.2,1.7))
  dev.off()
} else {
  SprFir_Agg_Sig_ROC_AUC = 83.7
}
```

```
## pdf
##   2
```

```r
if (calcROC) {
  curTime=Sys.time()
  print(paste("ROCR graph 2 started at",curTime))

  ROCpred_SprFir_Ind = prediction(SprFir_Ind_Train_predict, forestTrain$Spruce_Fir)
  summary(ROCpred_SprFir_Ind)

  ROCperf_SprFir_Ind = performance(ROCpred_SprFir_Ind, "tpr", "fpr")
  summary(ROCperf_SprFir_Ind)

  SprFir_Ind_Sig_ROC_AUC = as.numeric(performance(ROCpred_SprFir_Ind, "auc")@y.values)
  SprFir_Ind_Sig_ROC_AUC=as.integer(as.numeric(SprFir_Ind_Sig_ROC_AUC)*1000)/10
  SprFir_Ind_Sig_ROC_AUC

  jpeg(filename="Fig-ROC_perf_SprFir_Ind_Sig.jpg")
  plot(ROCperf_SprFir_Ind, colorize=TRUE, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-0.2,1.7))
  dev.off()
} else {
  SprFir_Ind_Sig_ROC_AUC = 83.8
}
```

```
## [1] "ROCR graph 2 started at 2018-08-12 22:09:44"
```

```
## pdf
##   2
```

The threshold graphs are essentially identical. This is making me think that there is not much difference between the two models. The AIC score for the Soil Type model is AIC: 351676 and for the individuated variables is: AIC: 351839. The Soil type model AIC score is 0.046% better than the individuated model.

Calculate Accuracy of Spruce and Fir Logisitic Model - Sig Vars

Calculate Spruce and Fir Aggregated Data Logisitic Model Accuracy - Significant Vars

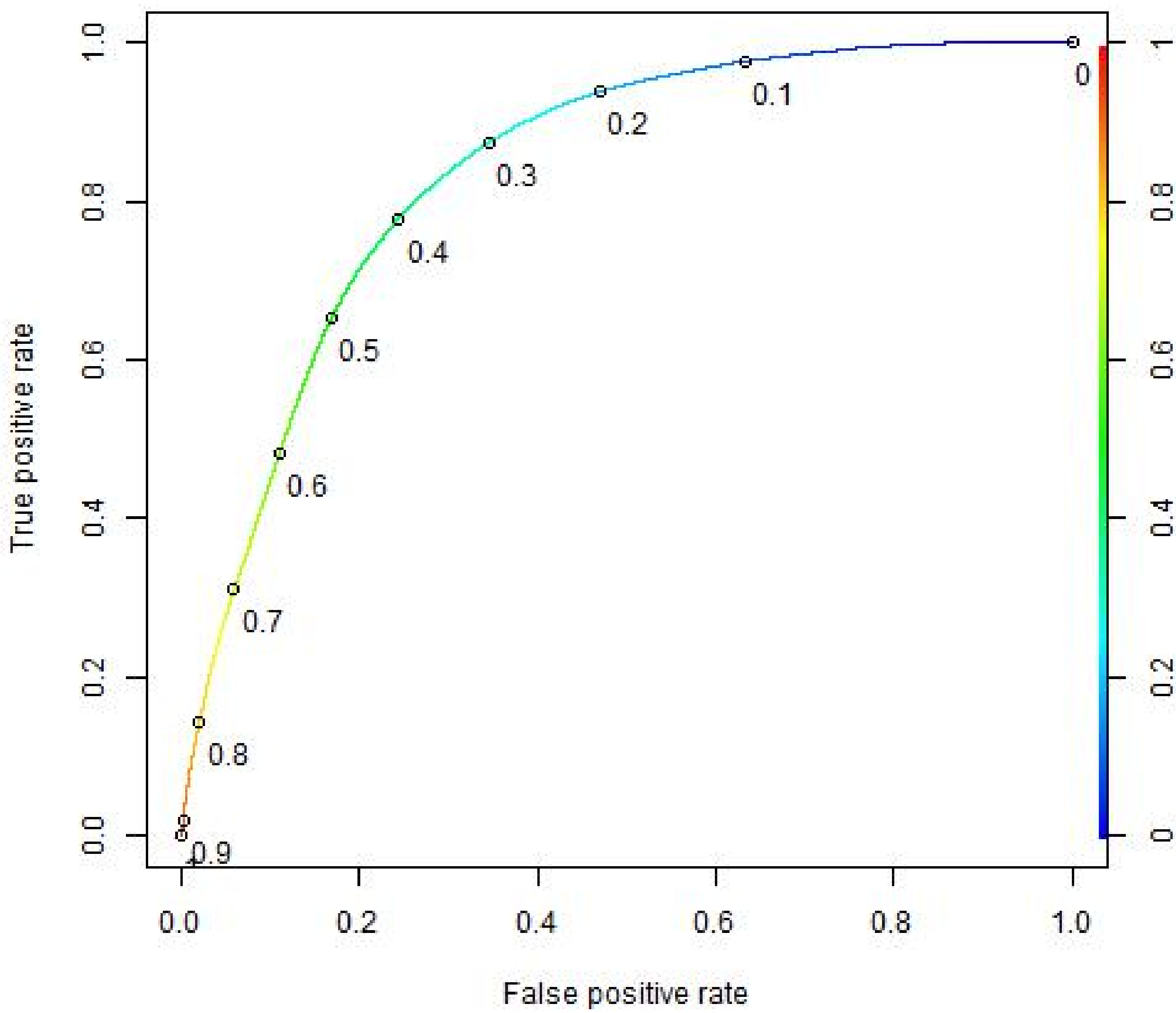

Figure 3: Spruce and Fir ROC for Aggregated Significant Data

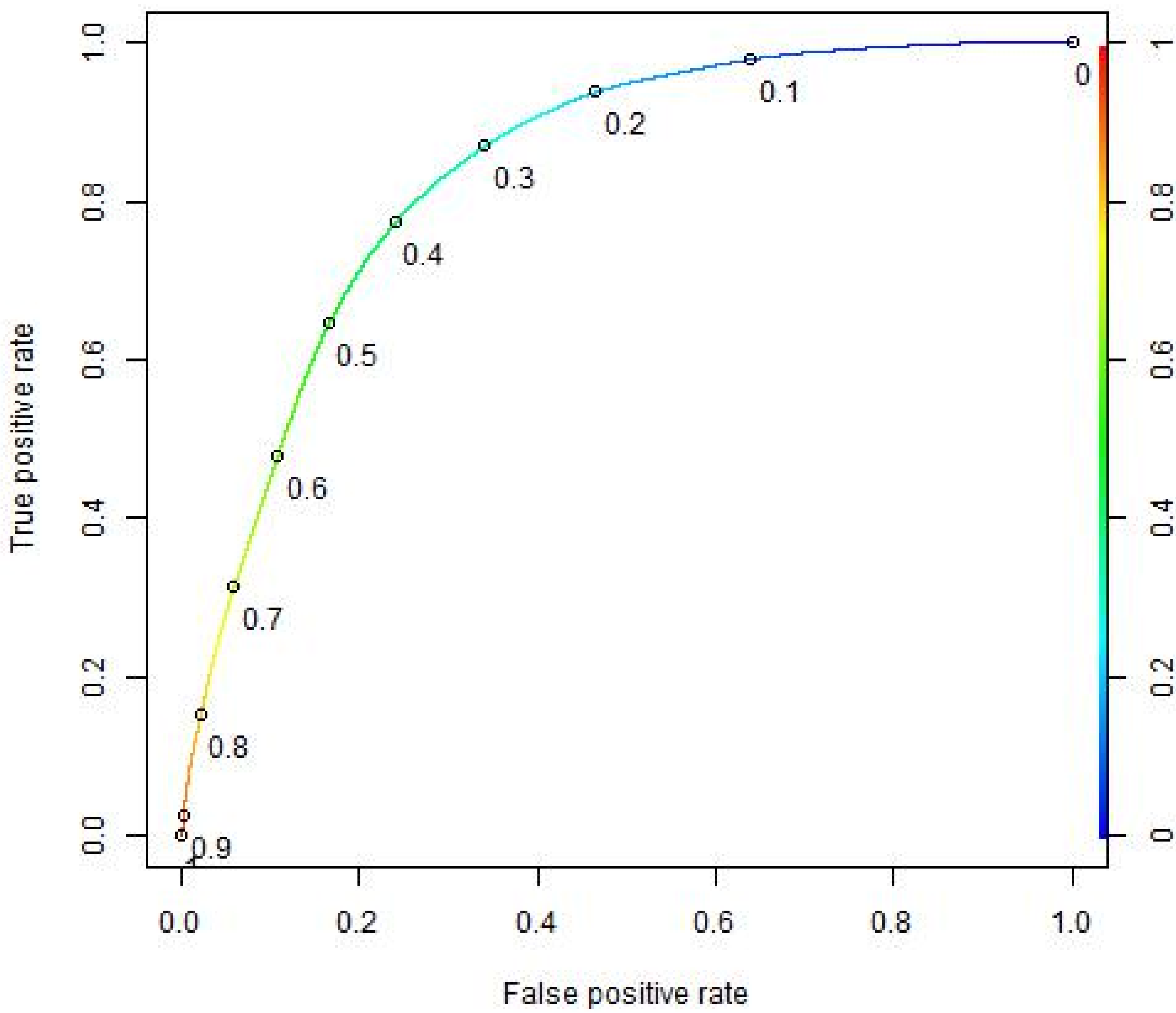

Figure 4: Spruce and Fir ROC for Individuated Significant Data

Find best Spruce and Fir threshold for Aggregated Data using significant variables.

```
result = calcLogisticModelAccuracy (forestTrain$Spruce_Fir, SprFir_Agg_Train_predict,
                       0.0, 1, 10, "Spruce_Fir", "Other", 1,1)
```

```
## [1] "Searching for threshold producing best Sensitivity_Specificity"
## [1] "start= 0 end= 1 inc= 0.1"
## [1] "Thresh=0, Accuracy=36.4%, BaseAcc(Other)=63.5%, Sens=100%, Spec=0%, Sens^2+Spec^2=-2"
## [1] "Thresh=0.1, Accuracy=59%, BaseAcc(Other)=63.5%, Sens=97.6%, Spec=36.8%, Sens^2+Spec^2=1.089"
## [1] "Thresh=0.2, Accuracy=67.9%, BaseAcc(Other)=63.5%, Sens=93.8%, Spec=53%, Sens^2+Spec^2=1.162"
## [1] "Thresh=0.3, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=87.4%, Spec=65.5%, Sens^2+Spec^2=1.194"
## [1] "Thresh=0.4, Accuracy=76.4%, BaseAcc(Other)=63.5%, Sens=77.8%, Spec=75.6%, Sens^2+Spec^2=1.177"
## [1] "Thresh=0.5, Accuracy=76.6%, BaseAcc(Other)=63.5%, Sens=65.1%, Spec=83.1%, Sens^2+Spec^2=1.116"
## [1] "Thresh=0.6, Accuracy=74.1%, BaseAcc(Other)=63.5%, Sens=48.1%, Spec=88.9%, Sens^2+Spec^2=1.023"
## [1] "Thresh=0.7, Accuracy=71.1%, BaseAcc(Other)=63.5%, Sens=31%, Spec=94%, Sens^2+Spec^2=0.981"
## [1] "Thresh=0.8, Accuracy=67.4%, BaseAcc(Other)=63.5%, Sens=14.4%, Spec=97.9%, Sens^2+Spec^2=0.98"
## [1] "Thresh=0.9, Accuracy=64.1%, BaseAcc(Other)=63.5%, Sens=2%, Spec=99.7%, Sens^2+Spec^2=0.995"
## [1] "Thresh=1, Accuracy=63.5%, BaseAcc(Other)=63.5%, Sens=0%, Spec=100%, Sens^2+Spec^2=-2"
## [1] "Best Sensitivity_Specificity threshold= 0.3 inc= 0.1"
## [1] "========================================"
## [1] "start= 0.2 end= 0.4 inc= 0.01"
## [1] "Thresh=0.2, Accuracy=67.9%, BaseAcc(Other)=63.5%, Sens=93.8%, Spec=53%, Sens^2+Spec^2=1.162"
## [1] "Thresh=0.21, Accuracy=68.6%, BaseAcc(Other)=63.5%, Sens=93.3%, Spec=54.4%, Sens^2+Spec^2=1.168"
## [1] "Thresh=0.22, Accuracy=69.2%, BaseAcc(Other)=63.5%, Sens=92.8%, Spec=55.7%, Sens^2+Spec^2=1.172"
## [1] "Thresh=0.23, Accuracy=69.8%, BaseAcc(Other)=63.5%, Sens=92.3%, Spec=57%, Sens^2+Spec^2=1.177"
## [1] "Thresh=0.24, Accuracy=70.4%, BaseAcc(Other)=63.5%, Sens=91.7%, Spec=58.2%, Sens^2+Spec^2=1.18"
## [1] "Thresh=0.25, Accuracy=71%, BaseAcc(Other)=63.5%, Sens=91%, Spec=59.5%, Sens^2+Spec^2=1.184"
## [1] "Thresh=0.26, Accuracy=71.5%, BaseAcc(Other)=63.5%, Sens=90.3%, Spec=60.7%, Sens^2+Spec^2=1.186"
## [1] "Thresh=0.27, Accuracy=72.1%, BaseAcc(Other)=63.5%, Sens=89.7%, Spec=62%, Sens^2+Spec^2=1.189"
## [1] "Thresh=0.28, Accuracy=72.6%, BaseAcc(Other)=63.5%, Sens=88.9%, Spec=63.2%, Sens^2+Spec^2=1.191"
## [1] "Thresh=0.29, Accuracy=73%, BaseAcc(Other)=63.5%, Sens=88.2%, Spec=64.3%, Sens^2+Spec^2=1.193"
## [1] "Thresh=0.3, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=87.4%, Spec=65.5%, Sens^2+Spec^2=1.194"
## [1] "Thresh=0.31, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=86.5%, Spec=66.6%, Sens^2+Spec^2=1.194"
## [1] "Thresh=0.32, Accuracy=74.3%, BaseAcc(Other)=63.5%, Sens=85.6%, Spec=67.7%, Sens^2+Spec^2=1.193"
## [1] "Thresh=0.33, Accuracy=74.6%, BaseAcc(Other)=63.5%, Sens=84.7%, Spec=68.8%, Sens^2+Spec^2=1.192"
## [1] "Thresh=0.34, Accuracy=74.9%, BaseAcc(Other)=63.5%, Sens=83.8%, Spec=69.8%, Sens^2+Spec^2=1.191"
## [1] "Thresh=0.35, Accuracy=75.2%, BaseAcc(Other)=63.5%, Sens=82.9%, Spec=70.9%, Sens^2+Spec^2=1.19"
## [1] "Thresh=0.36, Accuracy=75.5%, BaseAcc(Other)=63.5%, Sens=81.9%, Spec=71.8%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.37, Accuracy=75.8%, BaseAcc(Other)=63.5%, Sens=80.9%, Spec=72.9%, Sens^2+Spec^2=1.186"
## [1] "Thresh=0.38, Accuracy=76%, BaseAcc(Other)=63.5%, Sens=79.9%, Spec=73.8%, Sens^2+Spec^2=1.184"
## [1] "Thresh=0.39, Accuracy=76.2%, BaseAcc(Other)=63.5%, Sens=78.8%, Spec=74.7%, Sens^2+Spec^2=1.181"
## [1] "Best Sensitivity_Specificity threshold= 0.3 inc= 0.01"
## [1] "========================================"
## [1] "start= 0.29 end= 0.31 inc= 0.001"
## [1] "Thresh=0.29, Accuracy=73%, BaseAcc(Other)=63.5%, Sens=88.2%, Spec=64.3%, Sens^2+Spec^2=1.193"
## [1] "Thresh=0.291, Accuracy=73.1%, BaseAcc(Other)=63.5%, Sens=88.1%, Spec=64.5%, Sens^2+Spec^2=1.193
## [1] "Thresh=0.292, Accuracy=73.1%, BaseAcc(Other)=63.5%, Sens=88%, Spec=64.6%, Sens^2+Spec^2=1.193"
## [1] "Thresh=0.293, Accuracy=73.2%, BaseAcc(Other)=63.5%, Sens=88%, Spec=64.7%, Sens^2+Spec^2=1.193"
## [1] "Thresh=0.294, Accuracy=73.2%, BaseAcc(Other)=63.5%, Sens=87.9%, Spec=64.8%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.295, Accuracy=73.3%, BaseAcc(Other)=63.5%, Sens=87.8%, Spec=64.9%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.296, Accuracy=73.3%, BaseAcc(Other)=63.5%, Sens=87.7%, Spec=65.1%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.297, Accuracy=73.4%, BaseAcc(Other)=63.5%, Sens=87.6%, Spec=65.2%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.298, Accuracy=73.4%, BaseAcc(Other)=63.5%, Sens=87.6%, Spec=65.3%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.299, Accuracy=73.4%, BaseAcc(Other)=63.5%, Sens=87.5%, Spec=65.4%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.3, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=87.4%, Spec=65.5%, Sens^2+Spec^2=1.194"
```

```
## [1] "Thresh=0.301, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=87.3%, Spec=65.6%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.302, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=87.2%, Spec=65.7%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.303, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=87.1%, Spec=65.9%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.304, Accuracy=73.7%, BaseAcc(Other)=63.5%, Sens=87.1%, Spec=66%, Sens^2+Spec^2=1.194"
## [1] "Thresh=0.305, Accuracy=73.7%, BaseAcc(Other)=63.5%, Sens=87%, Spec=66.1%, Sens^2+Spec^2=1.194"
## [1] "Thresh=0.306, Accuracy=73.7%, BaseAcc(Other)=63.5%, Sens=86.9%, Spec=66.2%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.307, Accuracy=73.8%, BaseAcc(Other)=63.5%, Sens=86.8%, Spec=66.3%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.308, Accuracy=73.8%, BaseAcc(Other)=63.5%, Sens=86.7%, Spec=66.4%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.309, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=86.6%, Spec=66.5%, Sens^2+Spec^2=1.194
## [1] "Thresh=0.31, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=86.5%, Spec=66.6%, Sens^2+Spec^2=1.194"
## [1] "========================================"
## [1] "Best Threshold=0.307"
## [1] "Best Sensitivity_Specificity=1.19476151496147"
```

```
curThresh = as.numeric(result[bestThreshIndex])
SprFir_Agg_Sig_threshold = curThresh
```

The accuracy for the best threshold on the training set for Spruce and Fir using significant aggregated data is shown below.

```
result = calcLogisticModelAccuracy (forestTrain$Spruce_Fir, SprFir_Agg_Train_predict,
                        curThresh, curThresh, 1, "Spruce_Fir", "Other", 3)
```

```
## [1] "Model Performance for threshold= 0.307"
## [1] "predicted performance="
##                        Predicted
## Actual                  FALSE=Predict:Other TRUE=Predict:Spruce_Fir
##   0=Actual:Other          171470 (TN)         86951 (FP)
##   1=Actual:Spruce_Fir     19483 (FN)          128805 (TP)
## [1] "Sensitivity= 0.868613778593008 (True positive rate of Spruce_Fir = TP/(TP+FN) = 128805 /( 12880
## [1] "Specificity= 0.663529666706653 (True negative rate of Other = TN/(TN+FP) = 171470 /( 171470 + 8
## [1] "Sens^2+Spec^2=1.194"
## [1] "Baseline (Other) Accuracy=0.635395"
## [1] "Logistic Accuracy=0.738304"
```

The accuracy for the best threshold on the testing set for Spruce and Fir using significant aggregated data is shown below.

```
result = calcLogisticModelAccuracy (forestTest$Spruce_Fir, SprFir_Agg_Test_predict,
                        curThresh, curThresh, 1, "Spruce_Fir", "Other", 3,
                        saveFile=saveFileName, desc="Spruce/Fir Sig Aggregate Vars",
                        AIC=SprFir_Agg_Sig_aic, AUC=SprFir_Agg_Sig_ROC_AUC)
```

```
## [1] "Model Performance for threshold= 0.307"
## [1] "predicted performance="
##                        Predicted
## Actual                  FALSE=Predict:Other TRUE=Predict:Spruce_Fir
##   0=Actual:Other          73436 (TN)          37315 (FP)
##   1=Actual:Spruce_Fir     8051 (FN)           55501 (TP)
## [1] "Sensitivity= 0.873316339375629 (True positive rate of Spruce_Fir = TP/(TP+FN) = 55501 /( 55501
## [1] "Specificity= 0.663073019656707 (True negative rate of Other = TN/(TN+FP) = 73436 /( 73436 + 373
## [1] "Sens^2+Spec^2=1.202"
## [1] "Baseline (Other) Accuracy=0.635393"
## [1] "Logistic Accuracy=0.739729"
```

```
list[RC, SprFir_Agg_Sig_model_acc, SprFir_Agg_Sig_baseline_acc,
      TN, FN, FP, TP, SprFir_Agg_Sig_sens, SprFir_Agg_Sig_spec] <- result
```

```r
if (RC != "OK") {
  print(paste("Error - terminating:",RC))
  knitr:knit_exit()
}
SprFir_Agg_Sig_model_acc = as.integer(as.numeric(SprFir_Agg_Sig_model_acc)*1000)/10
SprFir_Agg_Sig_baseline_acc = as.integer(as.numeric(SprFir_Agg_Sig_baseline_acc)*1000)/10
SprFir_Agg_Sig_sens = as.integer(as.numeric(SprFir_Agg_Sig_sens)*1000)/10
SprFir_Agg_Sig_spec = as.integer(as.numeric(SprFir_Agg_Sig_spec)*1000)/10
```

Calculate Spruce and Fir Individuated Data Logisitic Model Accuracy - Significant Vars

Find best Spruce and Fir threshold for Inividuated Data using significant variables.

```r
result = calcLogisticModelAccuracy (forestTrain$Spruce_Fir, SprFir_Ind_Train_predict,
                        0.0, 1, 10, "Spruce_Fir", "Other", 1,1)
```

```
## [1] "Searching for threshold producing best Sensitivity_Specificity"
## [1] "start= 0 end= 1 inc= 0.1"
## [1] "Thresh=0, Accuracy=36.4%, BaseAcc(Other)=63.5%, Sens=100%, Spec=0%, Sens^2+Spec^2=-2"
## [1] "Thresh=0.1, Accuracy=58.6%, BaseAcc(Other)=63.5%, Sens=97.9%, Spec=36%, Sens^2+Spec^2=1.088"
## [1] "Thresh=0.2, Accuracy=68.1%, BaseAcc(Other)=63.5%, Sens=93.7%, Spec=53.5%, Sens^2+Spec^2=1.165"
## [1] "Thresh=0.3, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=87%, Spec=65.9%, Sens^2+Spec^2=1.191"
## [1] "Thresh=0.4, Accuracy=76.4%, BaseAcc(Other)=63.5%, Sens=77.3%, Spec=75.9%, Sens^2+Spec^2=1.175"
## [1] "Thresh=0.5, Accuracy=76.5%, BaseAcc(Other)=63.5%, Sens=64.6%, Spec=83.4%, Sens^2+Spec^2=1.113"
## [1] "Thresh=0.6, Accuracy=74.1%, BaseAcc(Other)=63.5%, Sens=47.9%, Spec=89.1%, Sens^2+Spec^2=1.024"
## [1] "Thresh=0.7, Accuracy=71.2%, BaseAcc(Other)=63.5%, Sens=31.4%, Spec=94%, Sens^2+Spec^2=0.983"
## [1] "Thresh=0.8, Accuracy=67.7%, BaseAcc(Other)=63.5%, Sens=15.3%, Spec=97.7%, Sens^2+Spec^2=0.979"
## [1] "Thresh=0.9, Accuracy=64.2%, BaseAcc(Other)=63.5%, Sens=2.5%, Spec=99.7%, Sens^2+Spec^2=0.995"
## [1] "Thresh=1, Accuracy=63.5%, BaseAcc(Other)=63.5%, Sens=0%, Spec=100%, Sens^2+Spec^2=-2"
## [1] "Best Sensitivity_Specificity threshold= 0.3 inc= 0.1"
## [1] "========================================"
## [1] "start= 0.2 end= 0.4 inc= 0.01"
## [1] "Thresh=0.2, Accuracy=68.1%, BaseAcc(Other)=63.5%, Sens=93.7%, Spec=53.5%, Sens^2+Spec^2=1.165"
## [1] "Thresh=0.21, Accuracy=68.8%, BaseAcc(Other)=63.5%, Sens=93.2%, Spec=54.8%, Sens^2+Spec^2=1.17"
## [1] "Thresh=0.22, Accuracy=69.4%, BaseAcc(Other)=63.5%, Sens=92.6%, Spec=56.1%, Sens^2+Spec^2=1.174"
## [1] "Thresh=0.23, Accuracy=70%, BaseAcc(Other)=63.5%, Sens=92%, Spec=57.4%, Sens^2+Spec^2=1.177"
## [1] "Thresh=0.24, Accuracy=70.6%, BaseAcc(Other)=63.5%, Sens=91.4%, Spec=58.6%, Sens^2+Spec^2=1.18"
## [1] "Thresh=0.25, Accuracy=71.1%, BaseAcc(Other)=63.5%, Sens=90.7%, Spec=59.8%, Sens^2+Spec^2=1.182"
## [1] "Thresh=0.26, Accuracy=71.7%, BaseAcc(Other)=63.5%, Sens=90%, Spec=61.1%, Sens^2+Spec^2=1.185"
## [1] "Thresh=0.27, Accuracy=72.2%, BaseAcc(Other)=63.5%, Sens=89.3%, Spec=62.3%, Sens^2+Spec^2=1.187"
## [1] "Thresh=0.28, Accuracy=72.7%, BaseAcc(Other)=63.5%, Sens=88.5%, Spec=63.6%, Sens^2+Spec^2=1.189"
## [1] "Thresh=0.29, Accuracy=73.1%, BaseAcc(Other)=63.5%, Sens=87.8%, Spec=64.7%, Sens^2+Spec^2=1.191"
## [1] "Thresh=0.3, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=87%, Spec=65.9%, Sens^2+Spec^2=1.191"
## [1] "Thresh=0.31, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=86.1%, Spec=67%, Sens^2+Spec^2=1.191"
## [1] "Thresh=0.32, Accuracy=74.3%, BaseAcc(Other)=63.5%, Sens=85.2%, Spec=68.1%, Sens^2+Spec^2=1.19"
## [1] "Thresh=0.33, Accuracy=74.7%, BaseAcc(Other)=63.5%, Sens=84.3%, Spec=69.1%, Sens^2+Spec^2=1.19"
## [1] "Thresh=0.34, Accuracy=75%, BaseAcc(Other)=63.5%, Sens=83.4%, Spec=70.1%, Sens^2+Spec^2=1.189"
## [1] "Thresh=0.35, Accuracy=75.3%, BaseAcc(Other)=63.5%, Sens=82.5%, Spec=71.2%, Sens^2+Spec^2=1.188"
## [1] "Thresh=0.36, Accuracy=75.6%, BaseAcc(Other)=63.5%, Sens=81.5%, Spec=72.2%, Sens^2+Spec^2=1.186"
## [1] "Thresh=0.37, Accuracy=75.8%, BaseAcc(Other)=63.5%, Sens=80.5%, Spec=73.2%, Sens^2+Spec^2=1.184"
## [1] "Thresh=0.38, Accuracy=76.1%, BaseAcc(Other)=63.5%, Sens=79.5%, Spec=74.1%, Sens^2+Spec^2=1.182"
## [1] "Thresh=0.39, Accuracy=76.3%, BaseAcc(Other)=63.5%, Sens=78.4%, Spec=75%, Sens^2+Spec^2=1.179"
## [1] "Best Sensitivity_Specificity threshold= 0.3 inc= 0.01"
## [1] "========================================"
```

```
## [1] "start= 0.29 end= 0.31 inc= 0.001"
## [1] "Thresh=0.29, Accuracy=73.1%, BaseAcc(Other)=63.5%, Sens=87.8%, Spec=64.7%, Sens^2+Spec^2=1.191"
## [1] "Thresh=0.291, Accuracy=73.2%, BaseAcc(Other)=63.5%, Sens=87.7%, Spec=64.8%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.292, Accuracy=73.2%, BaseAcc(Other)=63.5%, Sens=87.6%, Spec=64.9%, Sens^2+Spec^2=1.19"
## [1] "Thresh=0.293, Accuracy=73.3%, BaseAcc(Other)=63.5%, Sens=87.5%, Spec=65.1%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.294, Accuracy=73.3%, BaseAcc(Other)=63.5%, Sens=87.5%, Spec=65.2%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.295, Accuracy=73.3%, BaseAcc(Other)=63.5%, Sens=87.4%, Spec=65.3%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.296, Accuracy=73.4%, BaseAcc(Other)=63.5%, Sens=87.3%, Spec=65.4%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.297, Accuracy=73.4%, BaseAcc(Other)=63.5%, Sens=87.2%, Spec=65.5%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.298, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=87.1%, Spec=65.6%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.299, Accuracy=73.5%, BaseAcc(Other)=63.5%, Sens=87.1%, Spec=65.7%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.3, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=87%, Spec=65.9%, Sens^2+Spec^2=1.191"
## [1] "Thresh=0.301, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=86.9%, Spec=66%, Sens^2+Spec^2=1.191"
## [1] "Thresh=0.302, Accuracy=73.6%, BaseAcc(Other)=63.5%, Sens=86.8%, Spec=66.1%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.303, Accuracy=73.7%, BaseAcc(Other)=63.5%, Sens=86.7%, Spec=66.2%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.304, Accuracy=73.7%, BaseAcc(Other)=63.5%, Sens=86.6%, Spec=66.3%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.305, Accuracy=73.8%, BaseAcc(Other)=63.5%, Sens=86.5%, Spec=66.4%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.306, Accuracy=73.8%, BaseAcc(Other)=63.5%, Sens=86.4%, Spec=66.5%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.307, Accuracy=73.8%, BaseAcc(Other)=63.5%, Sens=86.4%, Spec=66.6%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.308, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=86.3%, Spec=66.8%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.309, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=86.2%, Spec=66.9%, Sens^2+Spec^2=1.191
## [1] "Thresh=0.31, Accuracy=73.9%, BaseAcc(Other)=63.5%, Sens=86.1%, Spec=67%, Sens^2+Spec^2=1.191"
## [1] "========================================"
## [1] "Best Threshold=0.298"
## [1] "Best Sensitivity_Specificity=1.19176650170647"
```

```r
curThresh = as.numeric(result[bestThreshIndex])
SprFir_Ind_Sig_threshold = curThresh
```

The accuracy for the best threshold on the training set for Spruce and Fir using significant individuated data is shown below.

```r
result = calcLogisticModelAccuracy (forestTrain$Spruce_Fir, SprFir_Ind_Train_predict,
                        curThresh, curThresh, 1, "Spruce_Fir", "Other", 3)
```

```
## [1] "Model Performance for threshold= 0.298"
## [1] "predicted performance="
##                        Predicted
## Actual                  FALSE=Predict:Other TRUE=Predict:Spruce_Fir
##   0=Actual:Other          169749 (TN)         88672 (FP)
##   1=Actual:Spruce_Fir     18989 (FN)          129299 (TP)
## [1] "Sensitivity= 0.871945133793699 (True positive rate of Spruce_Fir = TP/(TP+FN) = 129299 /( 12929
## [1] "Specificity= 0.656869991215884 (True negative rate of Other = TN/(TN+FP) = 169749 /( 169749 + 8
## [1] "Sens^2+Spec^2=1.191"
## [1] "Baseline (Other) Accuracy=0.635395"
## [1] "Logistic Accuracy=0.735287"
```

The accuracy for the best threshold on the testing set for Spruce and Fir using significant individuated data is shown below.

```r
result = calcLogisticModelAccuracy (forestTest$Spruce_Fir, SprFir_Ind_Test_predict,
                        curThresh, curThresh, 1, "Spruce_Fir", "Other", 3,
                        saveFile=saveFileName, desc="Spruce/Fir Sig Individualized Vars",
                        AIC=SprFir_Ind_Sig_aic, AUC=SprFir_Ind_All_ROC_AUC)
```

```
## [1] "Model Performance for threshold= 0.298"
## [1] "predicted performance="
```

```
##                         Predicted
## Actual                   FALSE=Predict:Other TRUE=Predict:Spruce_Fir
##   0=Actual:Other             72756 (TN)          37995 (FP)
##   1=Actual:Spruce_Fir        7805 (FN)           55747 (TP)
## [1] "Sensitivity= 0.87718718529708 (True positive rate of Spruce_Fir = TP/(TP+FN) = 55747 /( 55747 +
## [1] "Specificity= 0.656933120242707 (True negative rate of Other = TN/(TN+FP) = 72756 /( 72756 + 379
## [1] "Sens^2+Spec^2=1.201"
## [1] "Baseline (Other) Accuracy=0.635393"
## [1] "Logistic Accuracy=0.737239"
```

```
list[RC, SprFir_Ind_Sig_model_acc, SprFir_Ind_Sig_baseline_acc,
      TN, FN, FP, TP, SprFir_Ind_Sig_sens, SprFir_Ind_Sig_spec] <- result
  if (RC != "OK") {
    print(paste("Error - terminating:",RC))
    knitr:knit_exit()
  }
  SprFir_Ind_Sig_model_acc = as.integer(as.numeric(SprFir_Ind_Sig_model_acc)*1000)/10
  SprFir_Ind_Sig_baseline_acc = as.integer(as.numeric(SprFir_Ind_Sig_baseline_acc)*1000)/10
  SprFir_Ind_Sig_sens = as.integer(as.numeric(SprFir_Ind_Sig_sens)*1000)/10
  SprFir_Ind_Sig_spec = as.integer(as.numeric(SprFir_Ind_Sig_spec)*1000)/10

############# End End End End End End End End End End End End #################
```

The accuracy of the models is shown below:

Logistic Model	Accuracy	Sens	Spec	AIC	AUC	Threshold
Spruce and Fir Aggregate All Vars	73.7%	87.4%	65.8%	379526	84.2%	0.297
Spruce and Fir Individual All Vars	73.6%	87.4%	65.8%	379731	84.2%	0.297
Spruce and Fir Aggregate Sig Vars	73.9%	87.3%	66.3%	384126	83.7%	0.307
Spruce and Fir Individual Sig Vars	73.7%	87.7%	65.6%	384376	83.8%	0.298

There is a slight degradation in the accuracy with insignificant variables eliminated, but not by much.

Conclusion

It is beginning to look like there is no advantage to dis-aggregating the Soil Type variables into their component parts. I was hoping there would be some improvement by allowing the individual variables to be "more finely" tuned. There is probably a mathematical explanation that proves there is no advantage of breaking out aggregated variables. I have to think about that more.

The logistic regression results for Spruce and Fir are 7% better than the original paper this project was modeled after. These tests need to be done for the remaining 6 forest cover types to see how regression does overall.

```
  curTime=Sys.time()
  print(paste("Forest Cover Logistic script ended at",curTime))
```

```
## [1] "Forest Cover Logistic script ended at 2018-08-13 03:12:57"
```