STAT 6340 Mini Project 3

Matthew Lynn

March 11, 2019

Section 1

Section 2 is coded in Section 1

```
# library and function for plotting decision boundaries later
library(caret)
decisionplot <- function(model, data, class = NULL, predict_type = "class",
                         resolution = 125, showgrid = TRUE, ...) {

  if(!is.null(class)) cl <- data[,class] else cl <- 1
  data <- data[,1:2]
  k <- length(unique(cl))

  plot(data, col = as.integer(cl)+1L, pch = as.integer(cl)+1L, ...)

  # make grid
  r <- sapply(data, range, na.rm = TRUE)
  xs <- seq(r[1,1], r[2,1], length.out = resolution)
  ys <- seq(r[1,2], r[2,2], length.out = resolution)
  g <- cbind(rep(xs, each=resolution), rep(ys, time = resolution))
  colnames(g) <- colnames(r)
  g <- as.data.frame(g)

  ### guess how to get class labels from predict
  ### (unfortunately not very consistent between models)
  p <- predict(model, g, type = predict_type)
  if(is.list(p)) p <- p$class
  p <- as.factor(p)

  if(showgrid) points(g, col = as.integer(p)+1L, pch = ".")

  z <- matrix(as.integer(p), nrow = resolution, byrow = TRUE)
  # contour(xs, ys, z, add = TRUE, drawlabels = FALSE,
  #         lwd = 1, levels = (1:(k-1))+.5)

  invisible(z)
}
```

Question 1(a)

First we load in our data and explore what the data is about and see if we can pick predictors for modeling

```r
set.seed(1)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
admission = read.csv("admission.csv", header = T)
# these x columns appear but have no use so we delete them
admission$X = admission$X.1 = admission$X.2 = admission$X.3 <- NULL
head(admission)
```

```
##    GPA GMAT Group
## 1 2.96  596     1
## 2 3.14  473     1
## 3 3.22  482     1
## 4 3.29  527     1
## 5 3.69  505     1
## 6 3.46  693     1
```

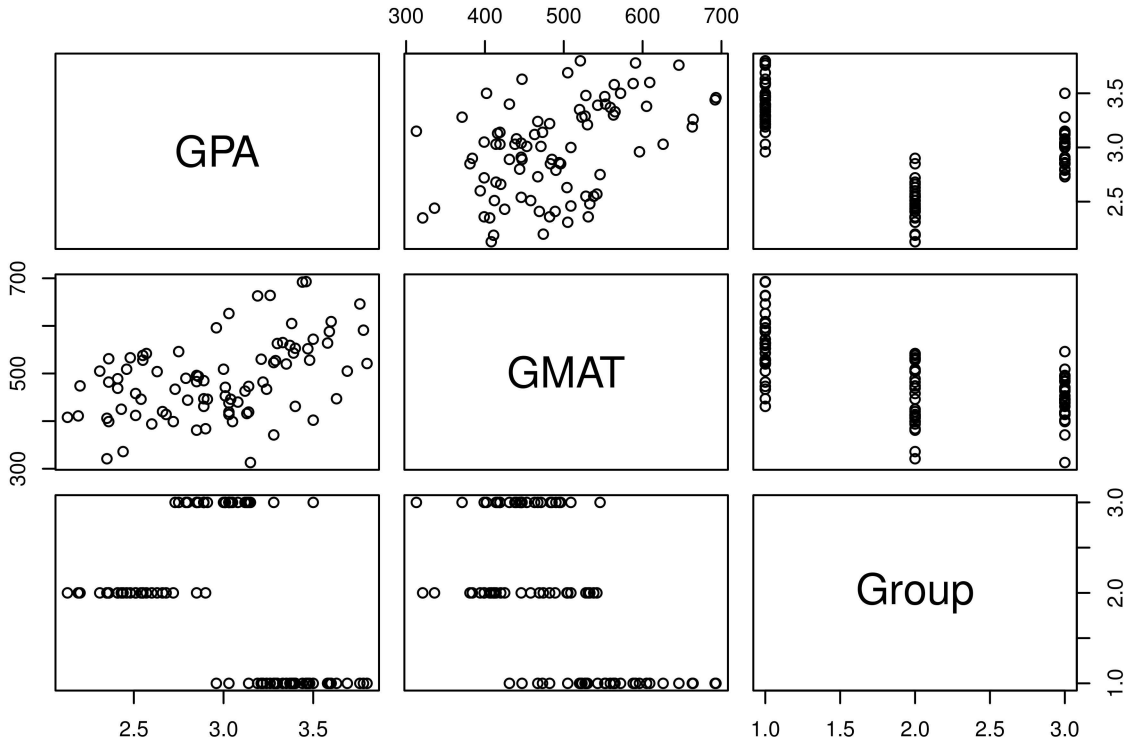
```r
str(admission)
```

```
## 'data.frame':    85 obs. of  3 variables:
##  $ GPA  : num  2.96 3.14 3.22 3.29 3.69 3.46 3.03 3.19 3.63 3.59 ...
##  $ GMAT : int  596 473 482 527 505 693 626 663 447 588 ...
##  $ Group: int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
summary(admission)
```

```
##       GPA             GMAT           Group      
##  Min.   :2.130   Min.   :313.0   Min.   :1.000  
##  1st Qu.:2.600   1st Qu.:425.0   1st Qu.:1.000  
##  Median :3.010   Median :482.0   Median :2.000  
##  Mean   :2.975   Mean   :488.4   Mean   :1.941  
##  3rd Qu.:3.300   3rd Qu.:538.0   3rd Qu.:3.000  
##  Max.   :3.800   Max.   :693.0   Max.   :3.000
```

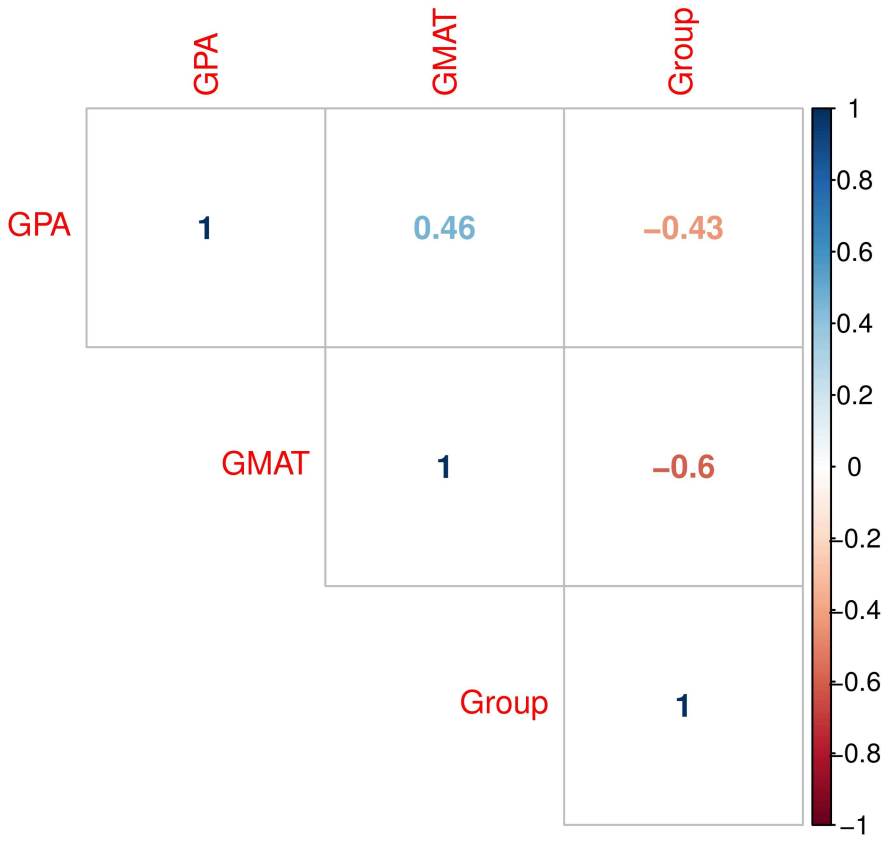
```r
pairs(subset(admission))
```




```
round(cor(subset(admission)), 2)
```

```
##          GPA  GMAT Group
## GPA     1.00  0.46 -0.43
## GMAT    0.46  1.00 -0.60
## Group -0.43 -0.60  1.00
```

```
corrplot(cor(subset(admission)), method = "number", type = "upper")
```



We see that GPA and GMAT have a nice relationship amongst each other. Group is categorical and would make for a great response variable.

1(b)

Here our goal is to use LDA, plot it with a decision boundary and compute the confusion matrix. Notes are in the comments!

```
set.seed(1)
# question 1 part b ----
# apply lda, make a decision boundary, and compute confusion matrix
library(MASS)
attach(admission)
library(dplyr)
```

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

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
```

```
##
##     filter, lag

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

```
# first we make a logical vector to sort out our test and train data
# since we want the last 5 of each Group category we must filter by category
admission_test = dplyr::bind_rows(tail(dplyr::filter(admission, Group==1), 5),
                                  tail(dplyr::filter(admission, Group==2), 5),
                                  tail(admission, 5))
# now we make a new vector, a logic vector and
# set all values in the test set to TRUE
admission_test$logic = TRUE
# merge the logic vector into admission and call it merger,
# this now sets the test obs to true in admission
merger = dplyr::left_join(admission, admission_test)
```

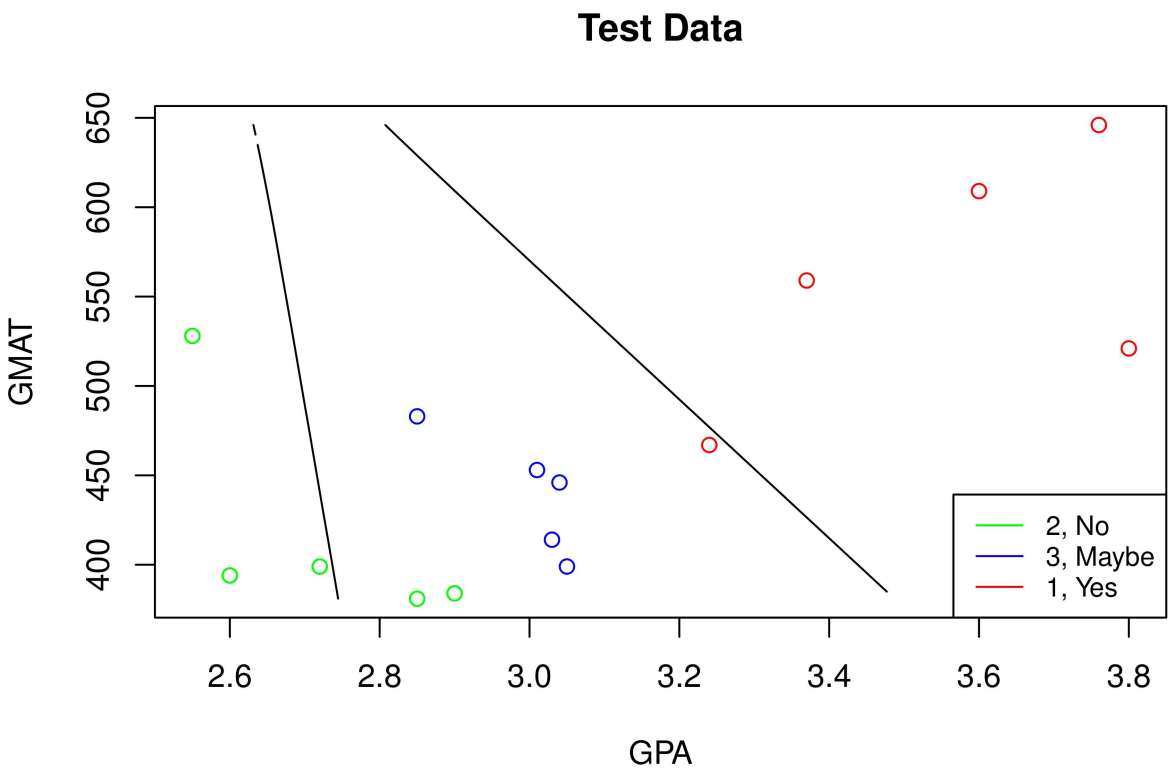
```
## Joining, by = c("GPA", "GMAT", "Group")
```

```
# flip the values of NA to true and flip the test true values to false.
# T = train, F = test
merger$logic = is.na(merger$logic)
# now we construct our train/test sets from the logic vector
train = cbind(GPA, GMAT, Group)[merger$logic, ]
test = cbind(GPA, GMAT, Group)[!merger$logic, ]
train = as.data.frame(train)
test = as.data.frame(test)

# now we can perform LDA and superimpose the decision boundary!
lda.fit <- lda(Group ~ GPA + GMAT, data = train)
lda.pred <- predict(lda.fit, test)
n.grid <- 50
x1.grid <- seq(f = min(test[, 1]), t = max(test[, 1]), l = n.grid)
x2.grid <- seq(f = min(test[, 2]), t = max(test[, 2]), l = n.grid)
grid <- expand.grid(x1.grid, x2.grid)
colnames(grid) <- colnames(test[,1:2])
pred.grid <- predict(lda.fit, grid)

# put equation for decision boundary here
lda.fit
```

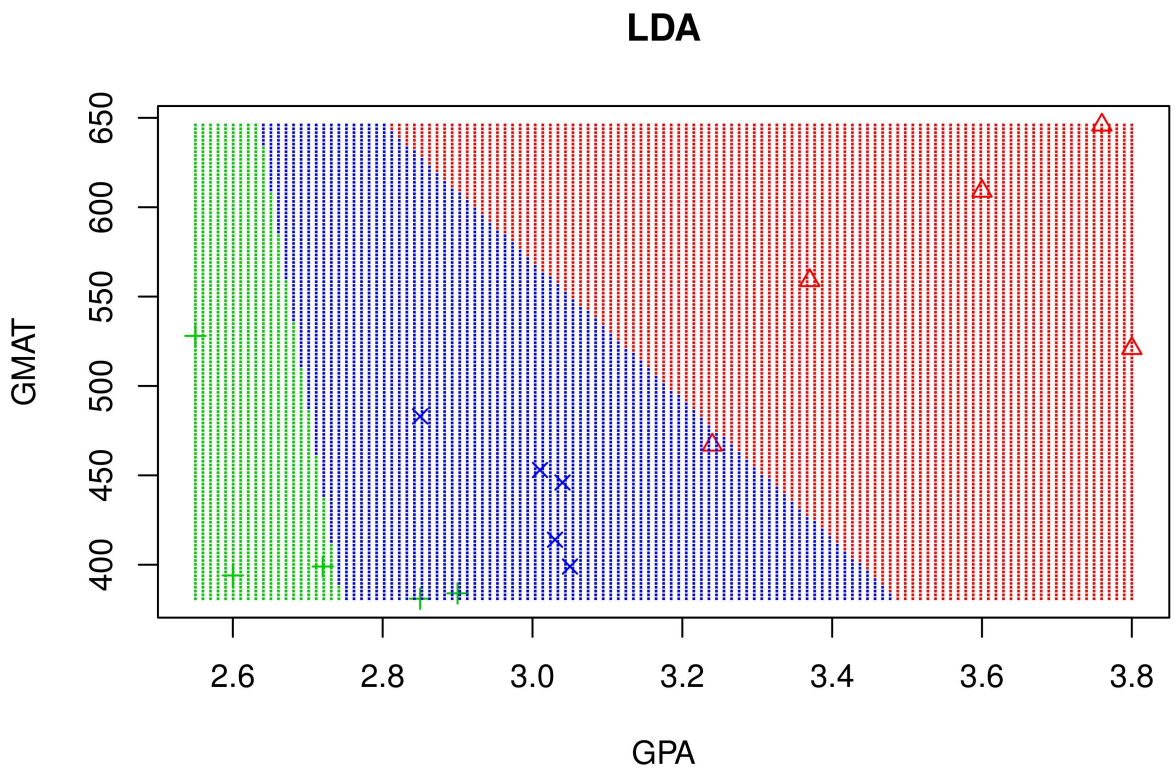
```
## Call:
## lda(Group ~ GPA + GMAT, data = train)
##
## Prior probabilities of groups:
##         1         2         3
## 0.3714286 0.3285714 0.3000000
##
## Group means:
##        GPA     GMAT
## 1 3.375000 561.3846
```

```
## 2 2.430000 453.5652
## 3 2.991905 447.9524
##
## Coefficients of linear discriminants:
##               LD1         LD2
## GPA  -5.458111330  1.70413416
## GMAT -0.007521159 -0.01466313
##
## Proportion of trace:
##    LD1    LD2
## 0.9657 0.0343
```

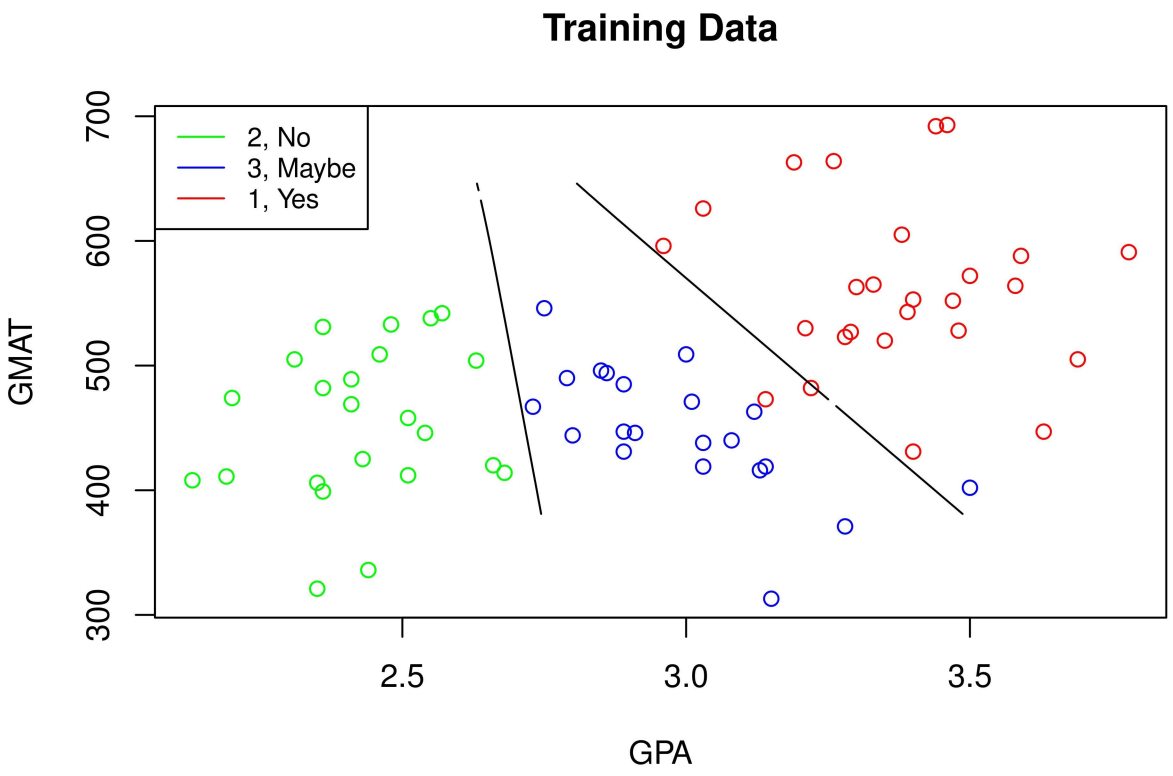
```r
# -5.458*GPA - 0.008*GMAT

par(mfrow = c(1,1))
model <- lda(Group ~ GPA+GMAT, data=train)
# simple and pretty version for graphing
# plot on test set with decision boundaries
prob1 <- matrix(pred.grid$posterior[, 1],
                nrow = n.grid, ncol = n.grid, byrow = F)
prob2 <- matrix(pred.grid$posterior[, 2],
                nrow = n.grid, ncol = n.grid, byrow = F)
plot(test[,1:2], col = ifelse(test$Group != 1,
                              ifelse(test$Group == 2, "green", "blue"), "red"))
contour(x1.grid, x2.grid, prob1, levels = 0.5,
        labels = "", xlab = "", ylab = "", main = "", add = T)
contour(x1.grid, x2.grid, prob2, levels = 0.5,
        labels = "", xlab = "", ylab = "", main = "", add = T)
legend("bottomright", legend=c("2, No", "3, Maybe", "1, Yes"),
       col=c("green", "blue", "red"), lty=1, cex=0.8, bg="transparent")
title("Test Data")
```



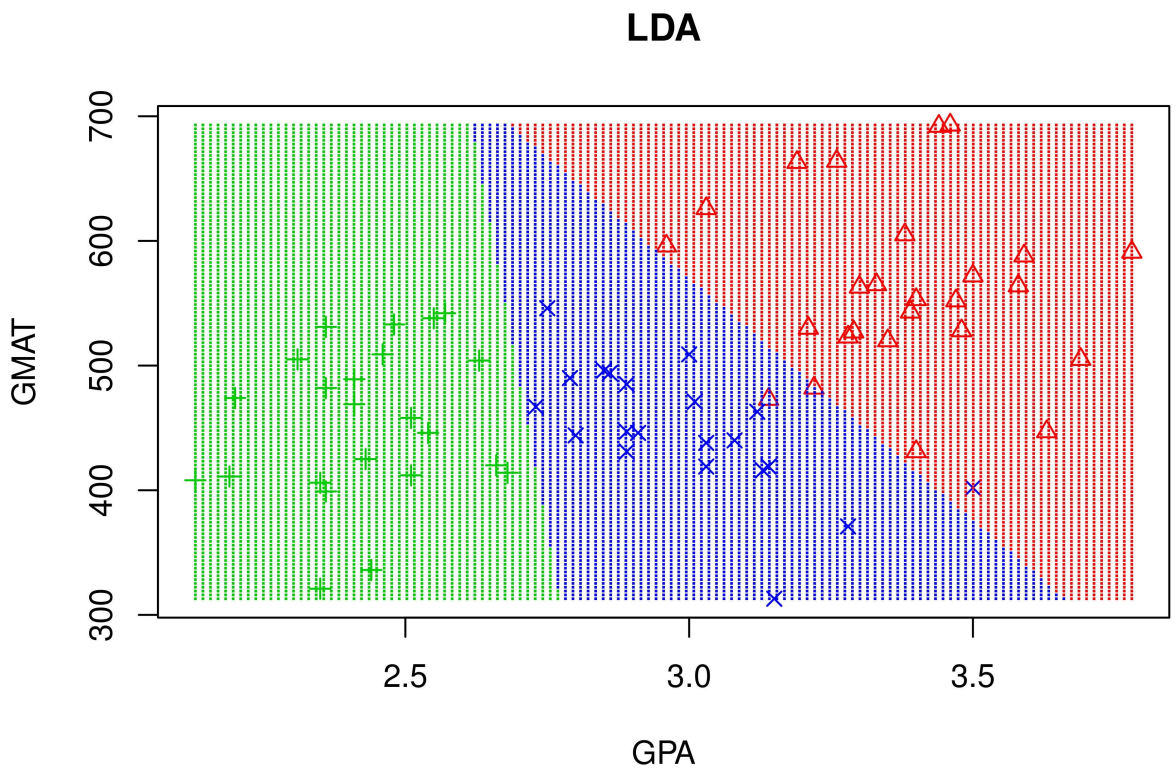

```
decisionplot(model, test, class = "Group", main = "LDA")
```




```
# now let us see what the plot looks like on the training set
plot(train[,1:2], col = ifelse(train$Group != 1,
                               ifelse(train$Group == 2, "green", "blue"), "red"))
contour(x1.grid, x2.grid, prob1, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
contour(x1.grid, x2.grid, prob2, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
legend("topleft", legend=c("2, No", "3, Maybe", "1, Yes"),
       col=c("green", "blue", "red"), lty=1, cex=0.8, bg="transparent")
title("Training Data")
```




```
decisionplot(model, train, class = "Group", main = "LDA")
```




```
# Cool it matches the handout in elearning :D
# now let us see on the full data
plot(admission[,1:2], col = ifelse(admission$Group != 1,
                        ifelse(admission$Group == 2, "green", "blue"), "red"))
contour(x1.grid, x2.grid, prob1, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
contour(x1.grid, x2.grid, prob2, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
legend("topleft", legend=c("2, No", "3, Maybe", "1, Yes"),
       col=c("green", "blue", "red"), lty=1, cex=0.8, bg="transparent")
title("Admission Data")
```