Homework4

Justin Lau

12/2/2020

Libraries

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.2     ✓ purrr   0.3.4
## ✓ tibble  3.0.4     ✓ dplyr   1.0.2
## ✓ tidyr   1.1.2     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.0
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
library(randomForest)
```

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

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

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

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

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

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(ROCR)
library(e1071)
library(imager)
```

```
## Loading required package: magrittr
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
##
## Attaching package: 'imager'
```

```
## The following object is masked from 'package:magrittr':
##
##     add
```

```
## The following object is masked from 'package:randomForest':
##
##     grow
```

```
## The following object is masked from 'package:stringr':
##
##     boundary
```

```
## The following object is masked from 'package:tidyr':
##
##     fill
```

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

```
## The following object is masked from 'package:graphics':
##
##     frame
```

```
## The following object is masked from 'package:base':
##
##     save.image
```

```
library(dplyr)
```

Question 1a Since there are n observations in the bootstrap sample, and we are trying to pick a sample that isn't j, we can make it n-1 observations. There are n replacement methods therefore the probability of any observation that isn't j is (n-1)^n divided by total observation replacements n^n which is (1-1/n)^n

b

```
n = 1000
bootstrap = (1-1/n)^n
bootstrap
```

```
## [1] 0.3676954
```

c

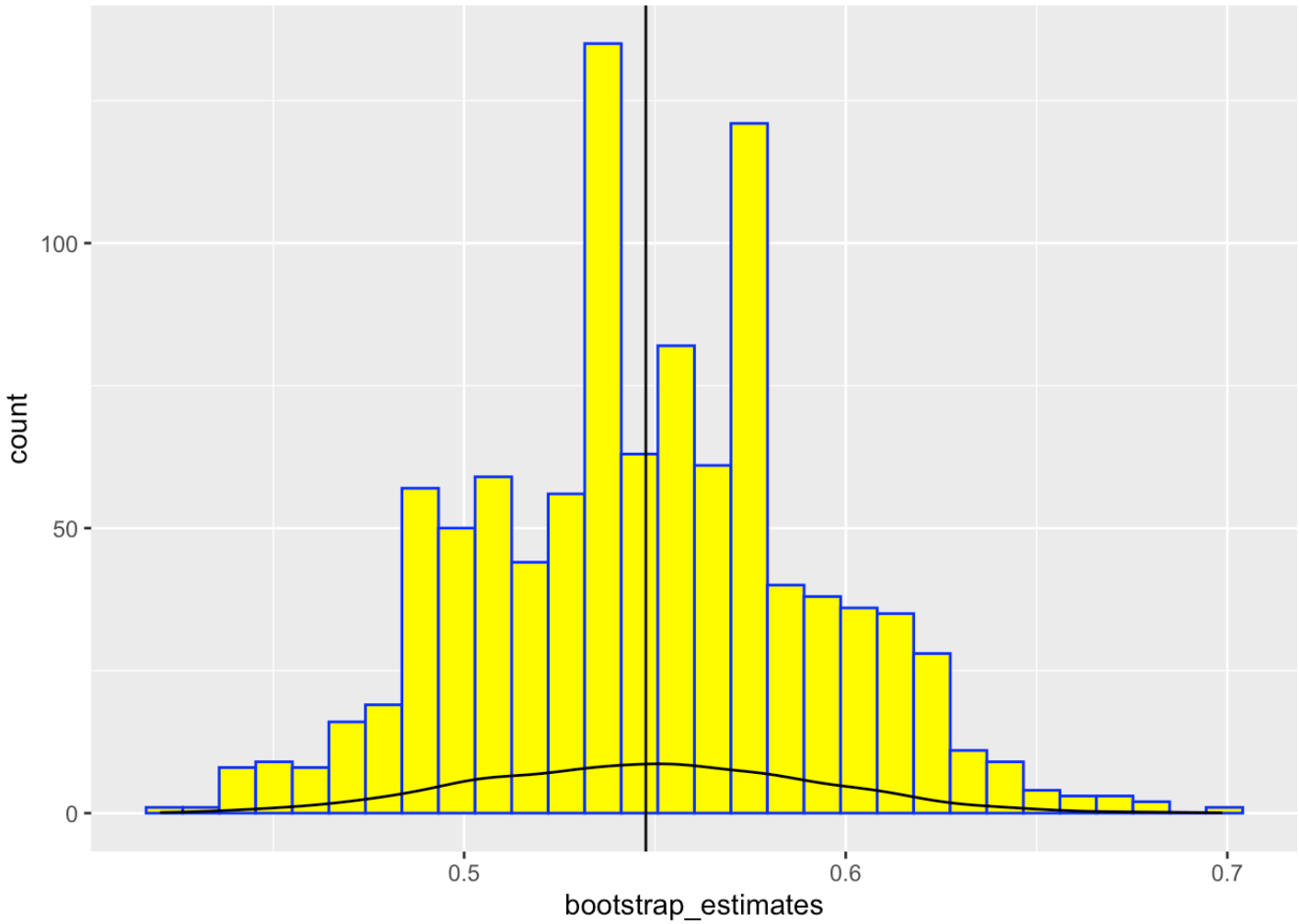
```
numbers <- 1:1000
samples <- list()
for(i in 1:10000){
  samples[[i]] <- sample(numbers, size = 1000, replace = TRUE)
}
boot <- c()
for(i in 1:10000){
  boot[i] <- (25 %in% unique(samples[[i]]))
}
(prob <- (1 - sum(boot)/(length(samples))))
```

```
## [1] 0.3654
```

d

```
B <- 1000
n <- 126
Shots <- rbinom(126,1,p=.492)
phat <- mean(Shots)
sd_hat <- sqrt(phat * (1-phat) / 50 )

bootstrap_estimates <- sapply(1:1000, function(i) mean(Shots[sample(n,replace=TRUE)])
)
bootstrap_estimates <- data.frame(bootstrap_estimates)
ggplot(bootstrap_estimates, aes(x=bootstrap_estimates)) + geom_histogram(color="blue"
, fill="yellow", bins=30) + geom_density(alpha=.2) + geom_vline(data=bootstrap_estima
tes, aes(xintercept=phat))
```




```
quantile(bootstrap_estimates$bootstrap_estimates, probs = c(.025, .975))
```

```
##      2.5%     97.5%
## 0.4603175 0.6349206
```

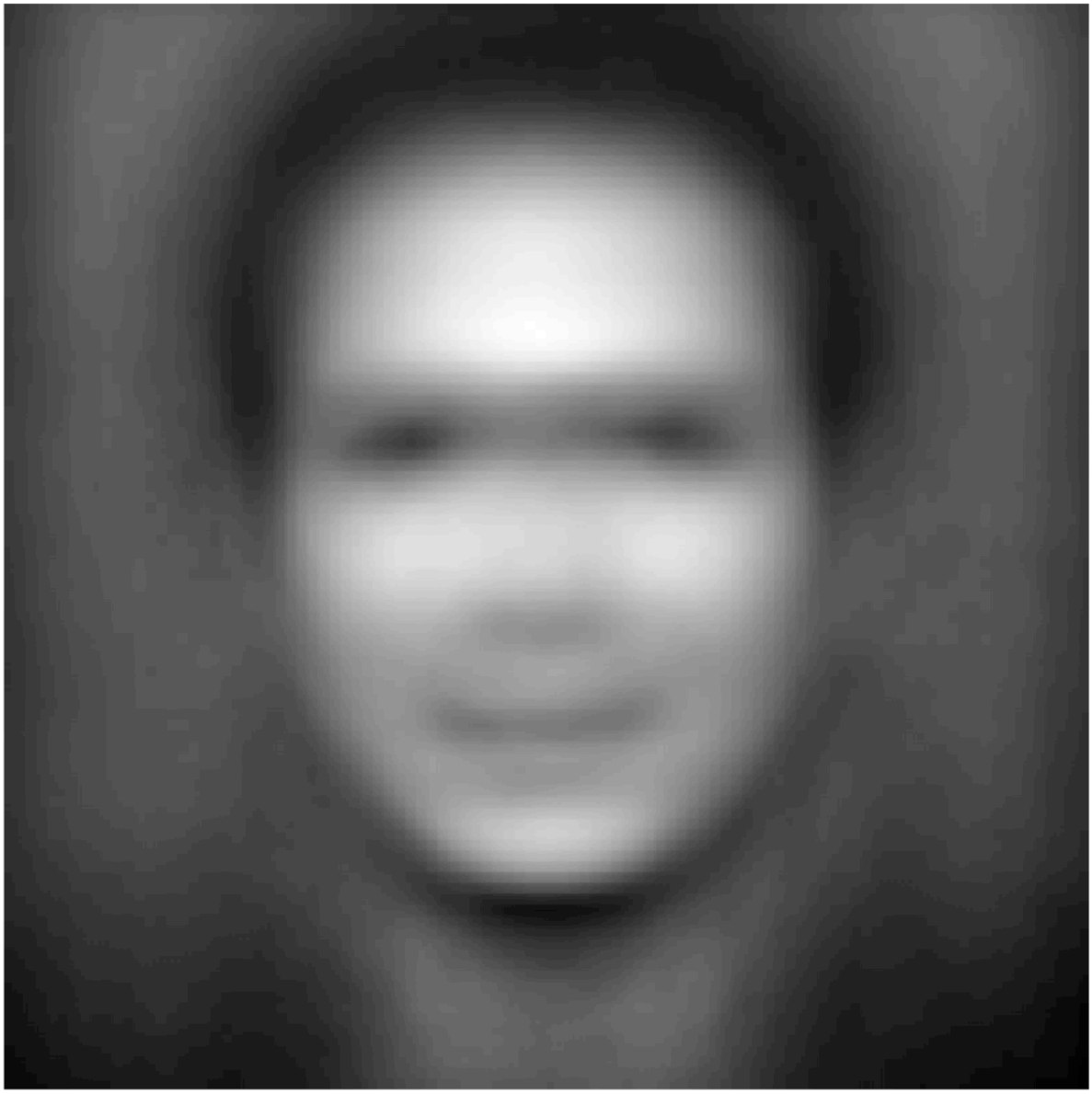
Curry's shooting percentage will go down as the season progresses because of the idea of regression to the mean. His shooting percentage is an outlier to the league's average as well as his own and as time goes on, his stats will regress more towards a mean value instead of an outlier.

Question 2

```
load("faces_array.RData")
```

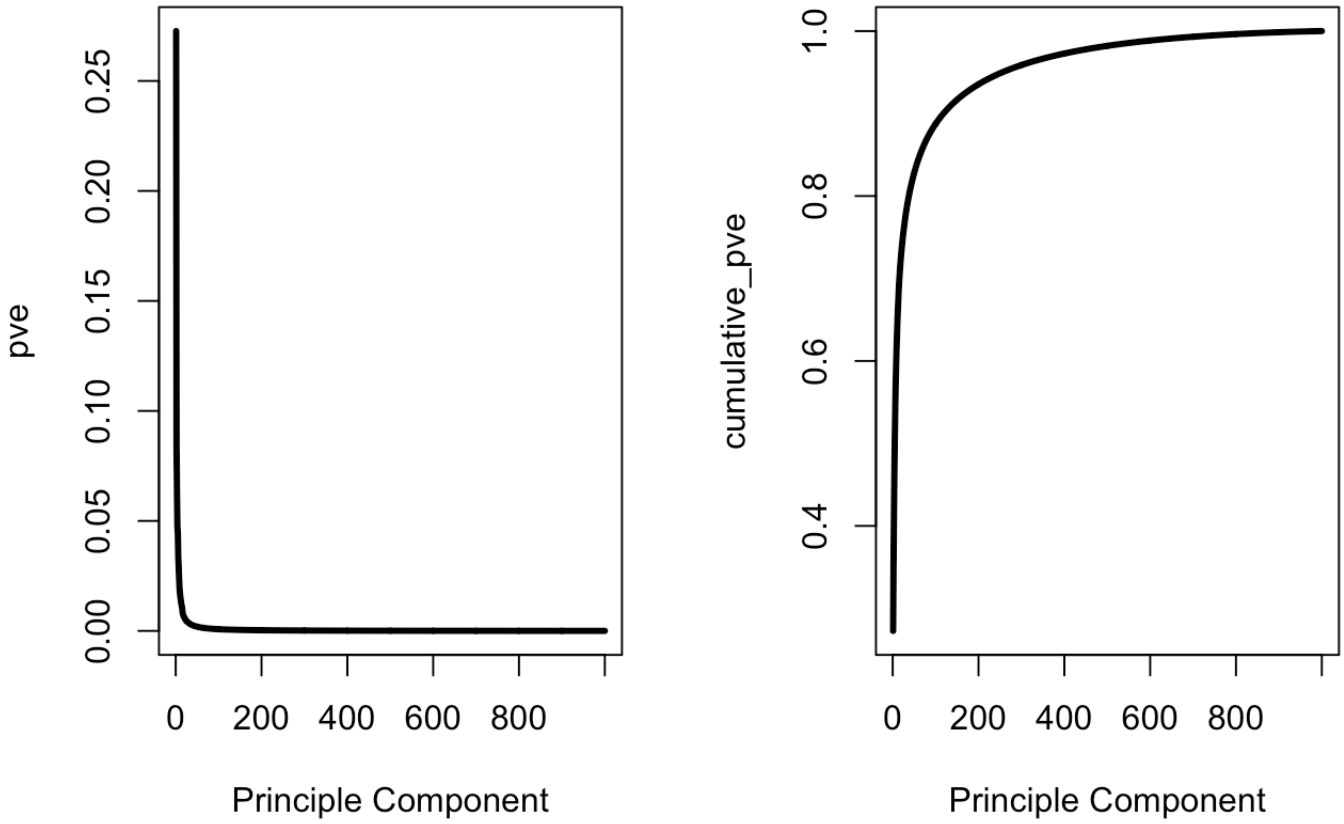
```
face_mat <- sapply(1:1000, function(i) as.numeric(faces_array[, , i])) %>% t
plot_face <- function(image_vector) {
plot(as.cimg(t(matrix(image_vector, ncol=100))), axes=FALSE, asp=1)
}
```

```
plot_face(colMeans(face_mat))
```



2b

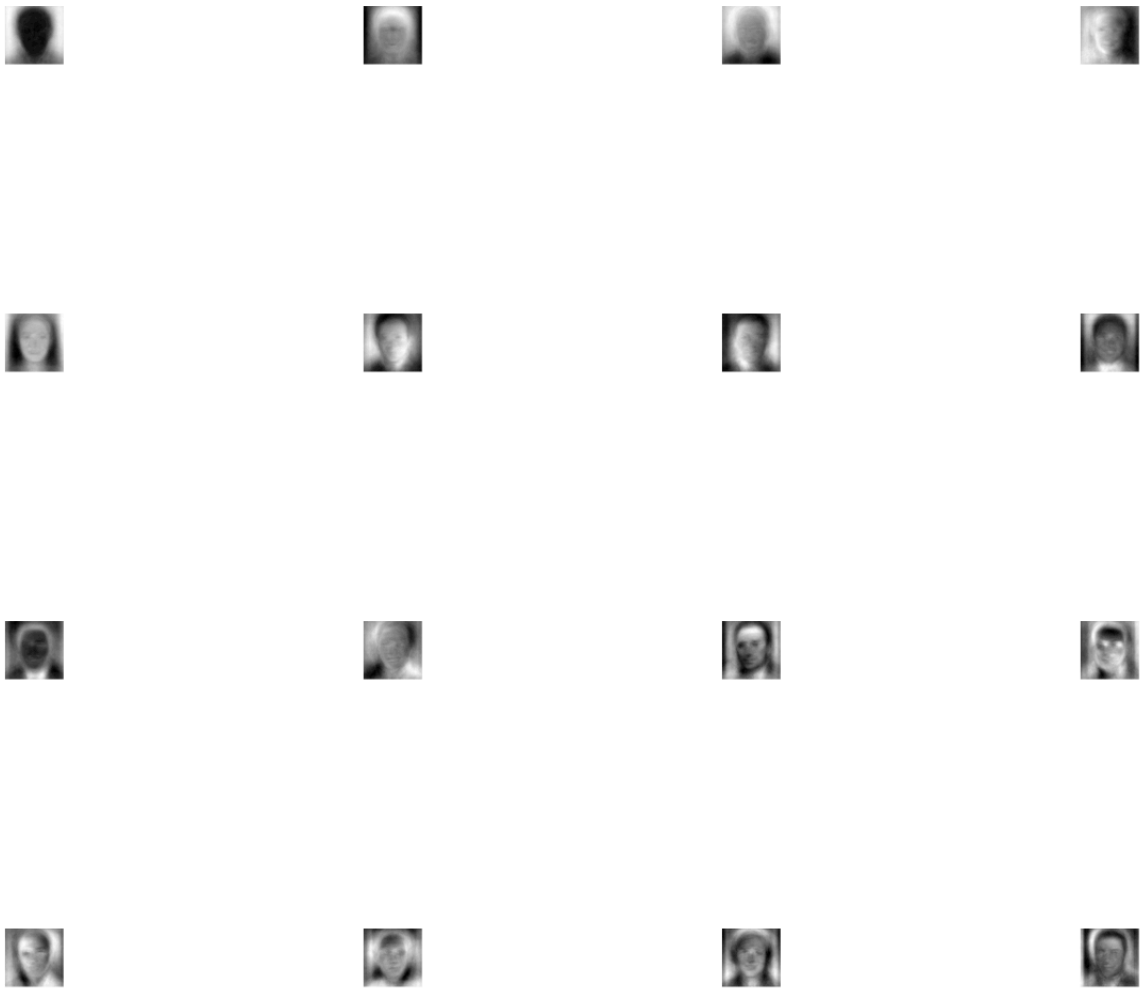
```
pca <- prcomp(face_mat,center=TRUE, scale=FALSE)
sdev <- pca$sdev
pve <- sdev^2 / sum(sdev^2)
cumulative_pve <- cumsum(pve)
par(mfrow=c(1, 2))
plot(pve, type="l", xlab = "Principle Component",lwd=3)
plot(cumulative_pve, type="l", xlab = "Principle Component", lwd=3)
```



We need about 5 PC to explain atleast 50% of the variance.

2c

```
par(mfrow=c(4,4))
for (i in 1:16){
  plot_face(pca$rotation[,i])
}
```



2d

```
par(mfrow=c(2,5))
lowest1 <- order(pca$x[,1], decreasing=FALSE)
highest1 <- order(pca$x[,1], decreasing=TRUE)
for(i in 1:5){
  plot_face(face_mat[lowest1[i],])
}
for (i in 1:5){
  plot_face(face_mat[highest1[i],])
}
```









```
par(mfrow=c(2,5))
lowest5 <- order(pca$x[,5], decreasing=FALSE)
highest5 <- order(pca$x[,5], decreasing=TRUE)
for(i in 1:5){
  plot_face(face_mat[lowest5[i],])
}
for (i in 1:5){
  plot_face(face_mat[highest5[i],])
}
```

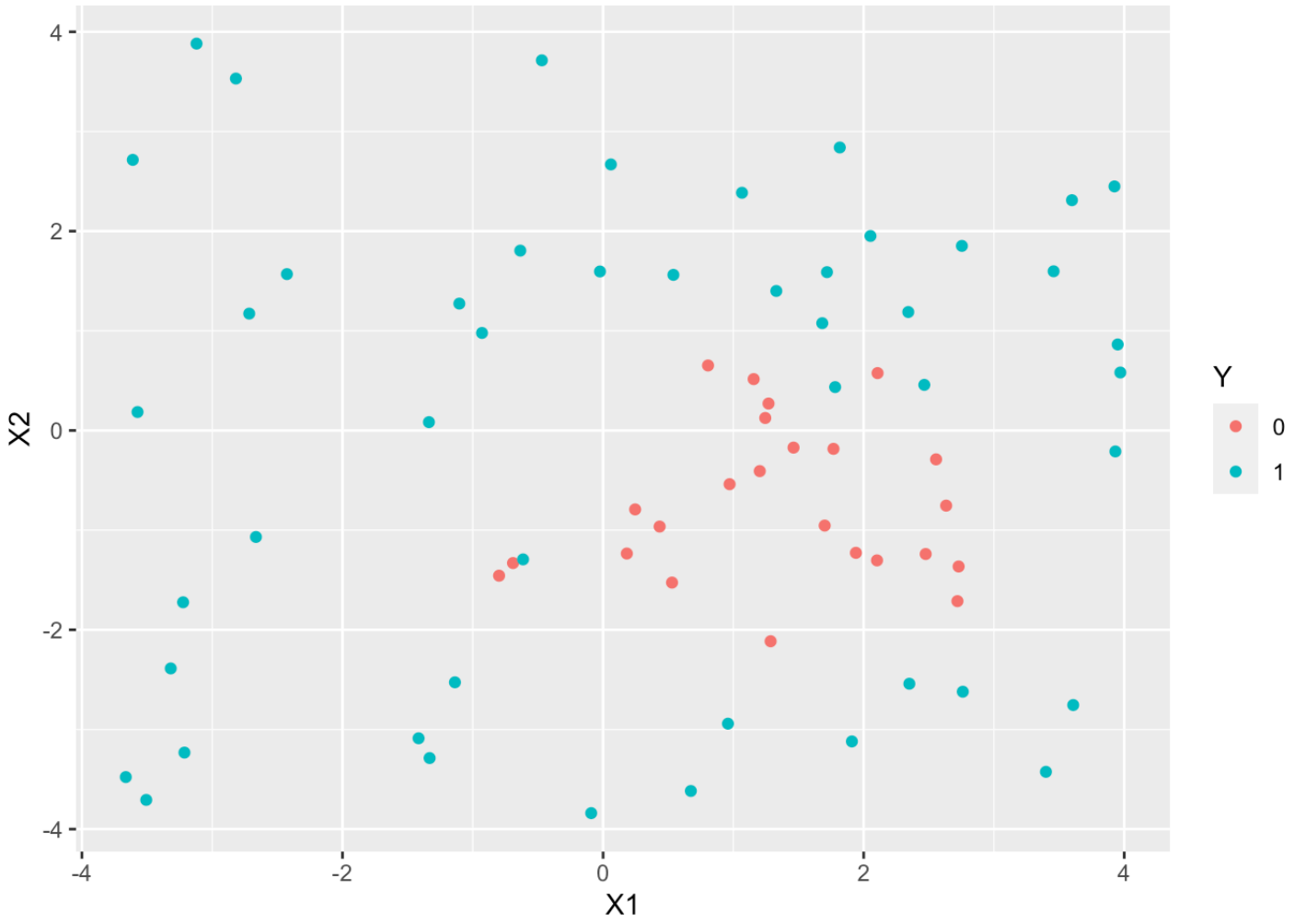
PC1 appears to get a lot more of the general shape of a person's face which might be all that is needed for facial recognition, but PC5 appears to get a lot more of the finer details of a person's face and the more minute details of the photo.

Question 3

```
nonlinear <- read.csv('nonlinear.csv', header=TRUE)
nonlinear$Y <- as.factor(nonlinear$Y)
```

3a

```
ggplot(nonlinear, aes(x=X1, y=X2, color=Y)) + geom_point()
```



3b

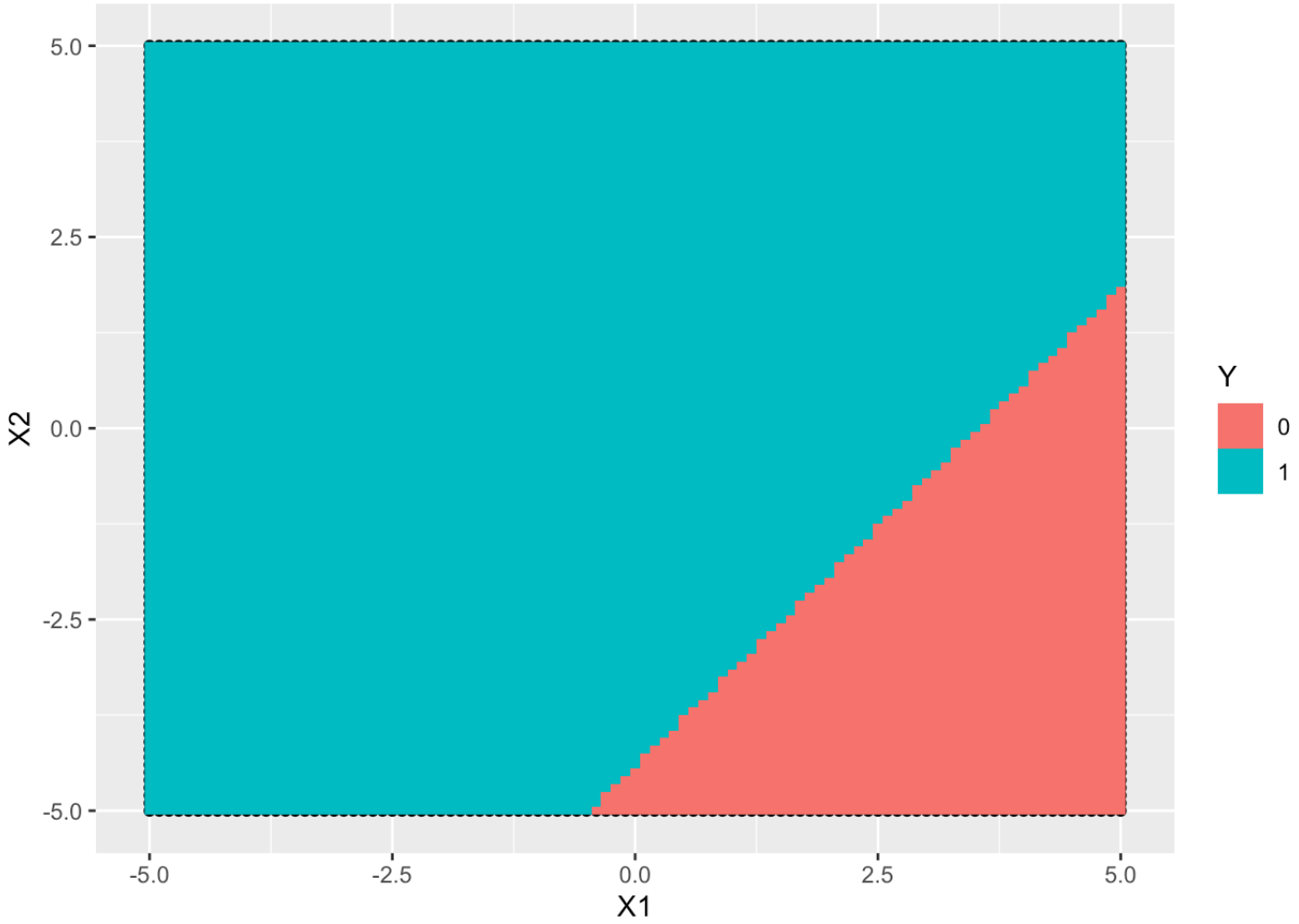
```
glm.fit <- glm(Y~X1+X2, data=nonlinear, family=binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Y ~ X1 + X2, family = binomial, data = nonlinear)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5940  -1.2476   0.6256   0.9155   1.5108
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.0224     0.3137   3.259  0.00112 **
## X1           -0.2893     0.1360  -2.127  0.03341 *
## X2            0.2323     0.1435   1.618  0.10560
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 84.523  on 69  degrees of freedom
## AIC: 90.523
##
## Number of Fisher Scoring iterations: 4
```

```
gr <- expand.grid(X1=seq(-5, 5, by=0.1),
                  X2=seq(-5, 5, by=0.1))

prob.test = round(predict(glm.fit, gr, type="response"),digits=5)
gr  = gr %>% mutate(Probability=prob.test)

gr <- gr %>% mutate(Y=as.factor(ifelse(Probability<=0.5, "0","1")))
ggplot(gr, aes(x=X1, y=X2), alpha=0.5) + geom_point() + geom_raster(aes(fill=Y))
```



3c

```
poly.fit <- glm(Y~poly(X1,2) + poly(X2,2) + X1:X2 , data=nonlinear, family=binomial)
```

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

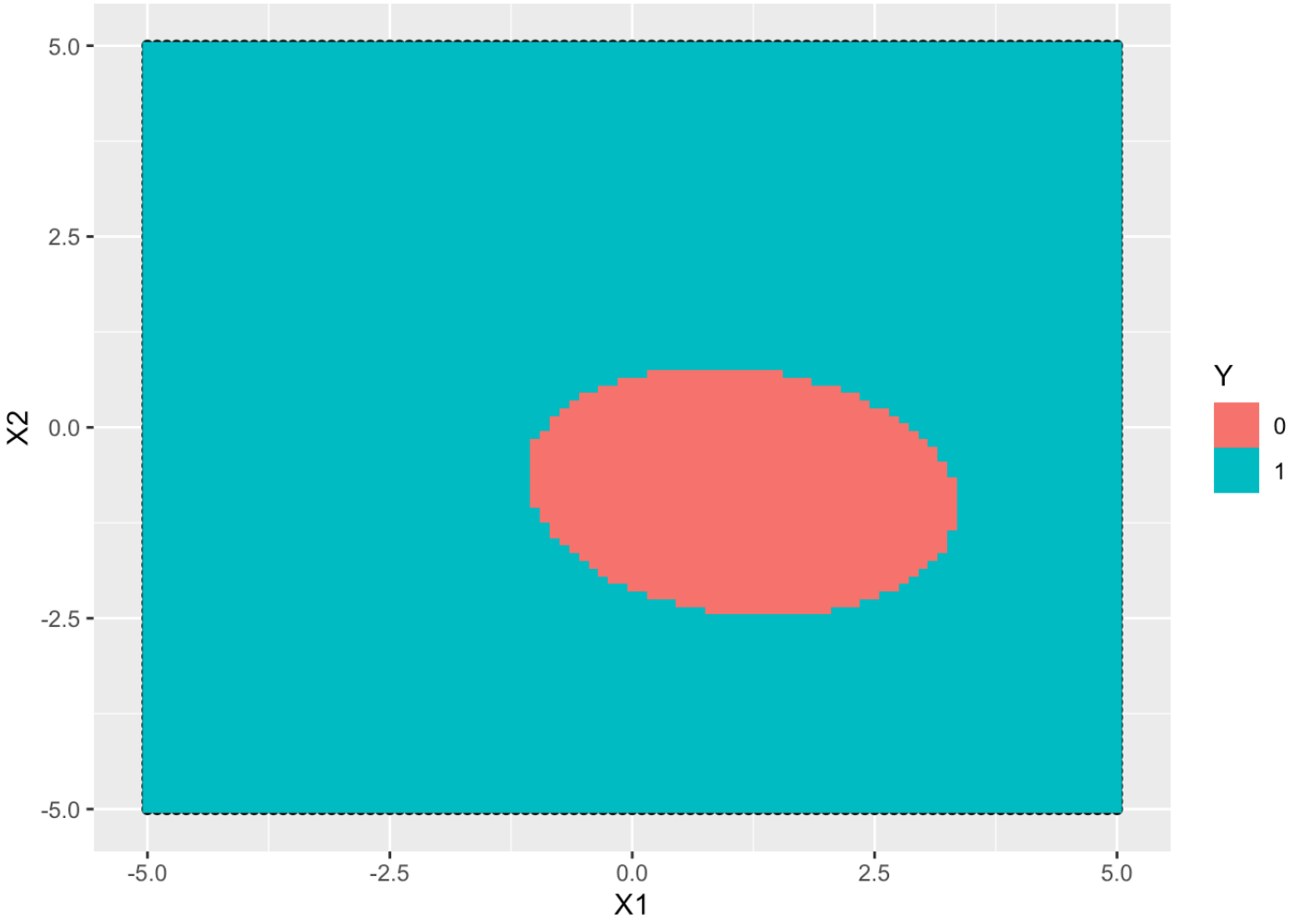
```
summary(poly.fit)
```

```
##
## Call:
## glm(formula = Y ~ poly(X1, 2) + poly(X2, 2) + X1:X2, family = binomial,
##     data = nonlinear)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.39081  -0.08271   0.00000   0.00930   1.90069
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   11.8000     4.8086   2.454   0.0141 *
## poly(X1, 2)1 -47.2697    28.2047  -1.676   0.0937 .
## poly(X1, 2)2  57.7766    29.0429   1.989   0.0467 *
## poly(X2, 2)1  45.0707    26.9112   1.675   0.0940 .
## poly(X2, 2)2  96.3106    39.7327   2.424   0.0154 *
## X1:X2          0.5014     0.7369   0.680   0.4963
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 13.852  on 66  degrees of freedom
## AIC: 25.852
##
## Number of Fisher Scoring iterations: 10
```

```
gr.poly <- expand.grid(X1=seq(-5, 5, by=0.1),
                  X2=seq(-5, 5, by=0.1))

poly.test = round(predict(poly.fit, gr.poly, type="response"),digits=5)
gr.poly  = gr.poly %>% mutate(Probability=poly.test)

gr.poly <- gr.poly %>% mutate(Y=as.factor(ifelse(Probability<=0.5, "0","1")))
ggplot(gr.poly, aes(x=X1, y=X2), alpha=0.5) + geom_point() + geom_raster(aes(fill=Y))
```




```
poly.fifth.fit <- glm(Y~poly(X1,5) + poly(X2,5), data=nonlinear, family=binomial)
```

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

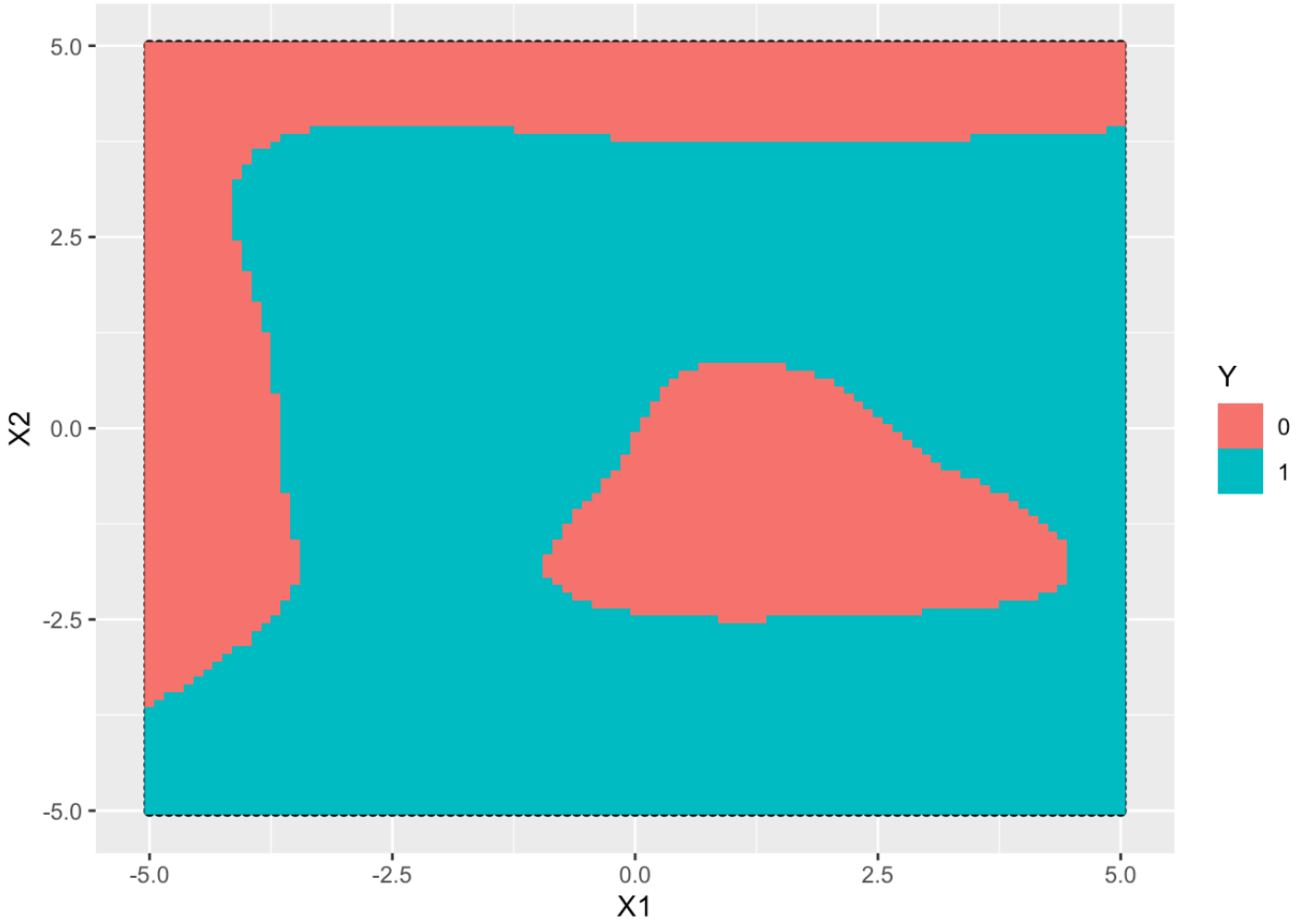
```
summary(poly.fifth.fit)
```

```
##
## Call:
## glm(formula = Y ~ poly(X1, 5) + poly(X2, 5), family = binomial,
##     data = nonlinear)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.24411  -0.02088   0.00000   0.00078   1.85481
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)     25.42      41.06   0.619    0.536
## poly(X1, 5)1   -49.29      88.35  -0.558    0.577
## poly(X1, 5)2    25.89      36.92   0.701    0.483
## poly(X1, 5)3    36.24      60.98   0.594    0.552
## poly(X1, 5)4   -34.71      64.85  -0.535    0.593
## poly(X1, 5)5    12.65      37.72   0.335    0.737
## poly(X2, 5)1  -174.38     386.21  -0.452    0.652
## poly(X2, 5)2   266.09     480.06   0.554    0.579
## poly(X2, 5)3  -228.97     422.75  -0.542    0.588
## poly(X2, 5)4    90.75     219.09   0.414    0.679
## poly(X2, 5)5  -101.31     203.20  -0.499    0.618
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 12.494  on 61  degrees of freedom
## AIC: 34.494
##
## Number of Fisher Scoring iterations: 14
```

```
gr.fifth.poly <- expand.grid(X1=seq(-5, 5, by=0.1),
                  X2=seq(-5, 5, by=0.1))

poly.fifth.test = round(predict(poly.fifth.fit, gr.fifth.poly, type="response"),digit
s=5)
gr.fifth.poly  = gr.fifth.poly %>% mutate(Probability=poly.fifth.test)

gr.fifth.poly <- gr.fifth.poly %>% mutate(Y=as.factor(ifelse(Probability<=0.5, "0","1
")))
ggplot(gr.fifth.poly, aes(x=X1, y=X2), alpha=0.5) + geom_point() + geom_raster(aes(fi
ll=Y))
```



3e The coefficients in the linear equation show that the only variable that is significant with an alpha value of 0.05 is X1, but as with the polynomial models we can see that in the 2 degree that only the ones to the second degree are significant. With the 5th degree polynomial there appears to be no significant variables under a 0.05 alpha value, this could be due to overfitting the data as we can see with the outer portions of the graph.

Question 4

```
library(ISLR)
head(Caravan)
```

```
##   MOSTYPE MAANTHUI MGEMOMV MGEMLEEF MOSHOOFD MGODRK MGODPR MGODOV MGODGE MRELGE
## 1      33        1        3         2        8      0      5      1      3      7
## 2      37        1        2         2        8      1      4      1      4      6
## 3      37        1        2         2        8      0      4      2      4      3
## 4       9        1        3         3        3      2      3      2      4      5
## 5      40        1        4         2       10      1      4      1      4      7
## 6      23        1        2         1        5      0      5      0      5      0
##   MRELSA MRELOV MFALLEEN MFGEKIND MFWEKIND MOPLHOOG MOPLMIDD MOPLLAAG MBERHOOG
## 1      0      2         1         2         6         1         2         7         1
```

```
## 2       2       2         0         4         5         0         5         4         0
## 3       2       4         4         4         2         0         5         4         0
## 4       2       2         2         3         4         3         4         2         4
## 5       1       2         2         4         4         5         4         0         0
## 6       6       3         3         5         2         0         5         4         2
##   MBERZELF MBERBOER MBERMIDD MBERARBG MBERARBO MSKA MSKB1 MSKB2 MSKC MSKD
## 1        0        1        2        5        2    1     1     2    6    1
## 2        0        0        5        0        4    0     2     3    5    0
## 3        0        0        7        0        2    0     5     0    4    0
## 4        0        0        3        1        2    3     2     1    4    0
## 5        5        4        0        0        0    9     0     0    0    0
## 6        0        0        4        2        2    2     2     2    4    2
##   MHHUUR MHKOOP MAUT1 MAUT2 MAUT0 MZFONDS MZPART MINKM30 MINK3045 MINK4575
## 1      1      8     8     0     1       8       1       0        4        5
## 2      2      7     7     1     2       6       3       2        0        5
## 3      7      2     7     0     2       9       0       4        5        0
## 4      5      4     9     0     0       7       2       1        5        3
## 5      4      5     6     2     1       5       4       0        0        9
## 6      9      0     5     3     3       9       0       5        2        3
##   MINK7512 MINK123M MINKGEM MKOOPKLA PWAPART PWABEDR PWALAND PPERSAUT PBESAUT
## 1        0        0       4        3       0       0       0        6       0
## 2        2        0       5        4       2       0       0        0       0
## 3        0        0       3        4       2       0       0        6       0
## 4        0        0       4        4       0       0       0        6       0
## 5        0        0       6        3       0       0       0        0       0
## 6        0        0       3        3       0       0       0        6       0
##   PMOTSCO PVRAAUT PAANHANG PTRACTOR PWERKT PBROM PLEVEN PPERSONG PGEZONG
## 1       0       0        0        0      0     0      0        0       0
## 2       0       0        0        0      0     0      0        0       0
## 3       0       0        0        0      0     0      0        0       0
## 4       0       0        0        0      0     0      0        0       0
## 5       0       0        0        0      0     0      0        0       0
## 6       0       0        0        0      0     0      0        0       0
##   PWAOREG PBRAND PZEILPL PPLEZIER PFIETS PINBOED PBYSTAND AWAPART AWABEDR
## 1       0      5       0        0      0       0        0       0       0
## 2       0      2       0        0      0       0        0       2       0
## 3       0      2       0        0      0       0        0       1       0
## 4       0      2       0        0      0       0        0       0       0
## 5       0      6       0        0      0       0        0       0       0
## 6       0      0       0        0      0       0        0       0       0
##   AWALAND APERSAUT ABESAUT AMOTSCO AVRAAUT AAANHANG ATRACTOR AWERKT ABROM
## 1       0        1       0       0       0        0        0      0     0
## 2       0        0       0       0       0        0        0      0     0
## 3       0        1       0       0       0        0        0      0     0
## 4       0        1       0       0       0        0        0      0     0
## 5       0        0       0       0       0        0        0      0     0
## 6       0        1       0       0       0        0        0      0     0
```

```
##   ALEVEN APERSONG AGEZONG AWAOREG ABRAND AZEILPL APLEZIER AFIETS AINBOED
## 1      0         0       0       0      1       0         0       0        0
## 2      0         0       0       0      1       0         0       0        0
## 3      0         0       0       0      1       0         0       0        0
## 4      0         0       0       0      1       0         0       0        0
## 5      0         0       0       0      1       0         0       0        0
## 6      0         0       0       0      0       0         0       0        0
##   ABYSTAND Purchase
## 1        0       No
## 2        0       No
## 3        0       No
## 4        0       No
## 5        0       No
## 6        0       No
```

```
data("Caravan")
```

4a

```
train <- sample(1:nrow(Caravan), 1000)
train.caravan <- Caravan[train,]
test.caravan <- Caravan[-train,]
```

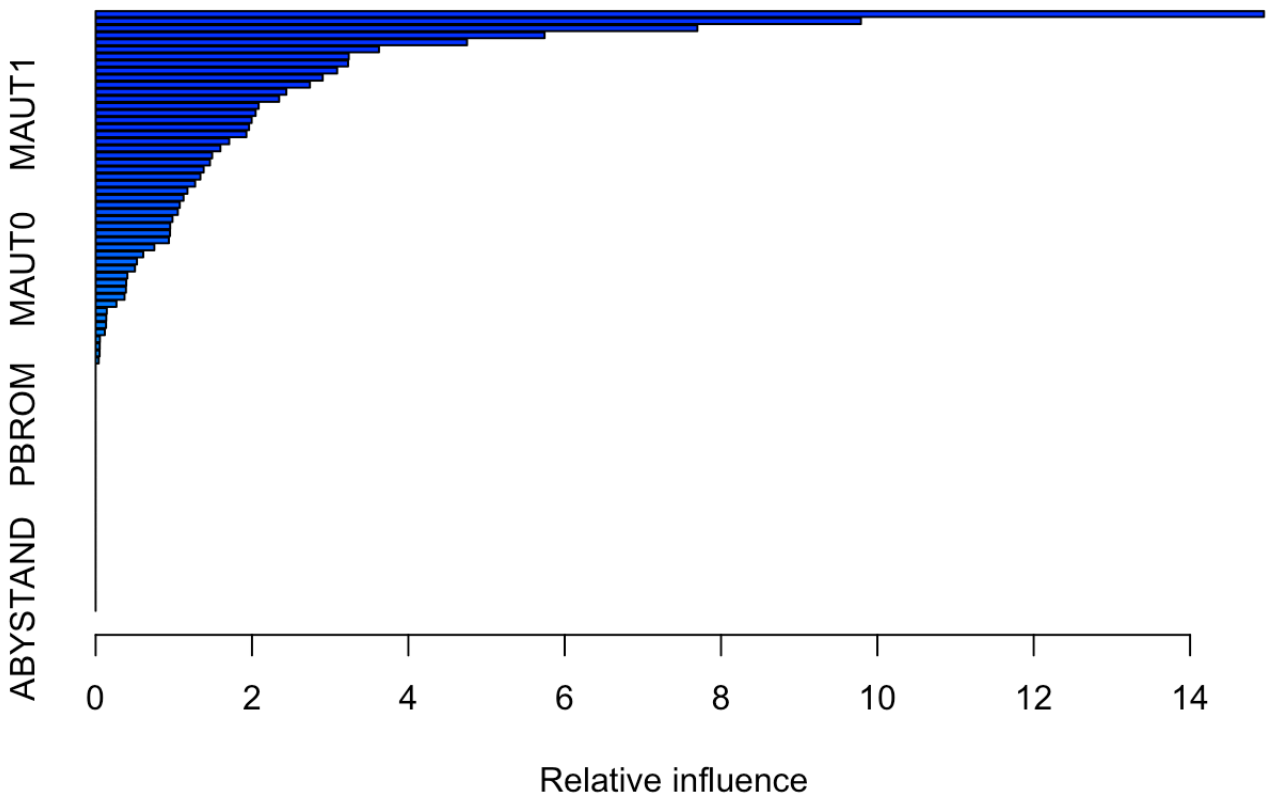
4b

```
set.seed(2)
boost.caravan = gbm(ifelse(Purchase=="Yes",1,0)~., data=train.caravan,
distribution="bernoulli", n.trees=1000, shrinkage = .01)
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
## variable 53: PWERKT has no variation.
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
## variable 74: AWERKT has no variation.
```

```
summary(boost.caravan)
```




```
##                var     rel.inf
## PPERSAUT PPERSAUT 14.94525563
## PBRAND     PBRAND  9.79019947
## MOPLLAAG MOPLLAAG  7.69762044
## MOSTYPE   MOSTYPE  5.74265613
## MBERMIDD MBERMIDD  4.75115866
## MSKC         MSKC  3.62440217
## MOPLHOOG MOPLHOOG  3.23829267
## MSKB1       MSKB1  3.22878392
## MBERARBG MBERARBG  3.08939040
## APERSAUT APERSAUT  2.90479365
## MBERARBO MBERARBO  2.74110843
## MINKGEM   MINKGEM  2.43897281
## MRELGE     MRELGE  2.34738948
## MSKA         MSKA  2.08493520
## MAUT1       MAUT1  2.04664693
## MKOOPKLA MKOOPKLA  1.99627390
## MINK7512 MINK7512  1.96250080
## MBERHOOG MBERHOOG  1.92876930
```

```
## MGODPR     MGODPR  1.70884579
## MINKM30   MINKM30  1.59573035
## PWAPART   PWAPART  1.49076602
## MHKOOP     MHKOOP  1.46282322
## MFGEKIND MFGEKIND  1.38213471
## MOPLMIDD MOPLMIDD  1.34011043
## MINK4575 MINK4575  1.27303270
## MSKD         MSKD  1.17459536
## PLEVEN     PLEVEN  1.12437466
## MHHUUR     MHHUUR  1.07617560
## MGEMLEEF MGEMLEEF  1.05149051
## MZPART     MZPART  0.98321148
## MSKB2       MSKB2  0.95325187
## MZFONDS   MZFONDS  0.95124422
## MINK3045 MINK3045  0.93675030
## MGODOV     MGODOV  0.75102179
## MFWEKIND MFWEKIND  0.60843756
## MOSHOOFD MOSHOOFD  0.52835194
## MAUT0       MAUT0  0.50284777
## MGEMOMV   MGEMOMV  0.40603292
## MBERZELF MBERZELF  0.39115818
## MGODGE     MGODGE  0.38790255
## PMOTSCO   PMOTSCO  0.36983053
## MAUT2       MAUT2  0.26596475
## MRELSA     MRELSA  0.14250669
## MGODRK     MGODRK  0.13579991
## MRELOV     MRELOV  0.13311518
## PBYSTAND PBYSTAND  0.11682771
## PFIETS     PFIETS  0.05458976
## MINK123M MINK123M  0.05206068
## PAANHANG PAANHANG  0.05039934
## MFALLEEN MFALLEEN  0.03946554
## MAANTHUI MAANTHUI  0.00000000
## MBERBOER MBERBOER  0.00000000
## PWABEDR   PWABEDR  0.00000000
## PWALAND   PWALAND  0.00000000
## PBESAUT   PBESAUT  0.00000000
## PVRAAUT   PVRAAUT  0.00000000
## PTRACTOR PTRACTOR  0.00000000
## PWERKT     PWERKT  0.00000000
## PBROM       PBROM  0.00000000
## PPERSONG PPERSONG  0.00000000
## PGEZONG   PGEZONG  0.00000000
## PWAOREG   PWAOREG  0.00000000
## PZEILPL   PZEILPL  0.00000000
## PPLEZIER PPLEZIER  0.00000000
## PINBOED   PINBOED  0.00000000
```

```
## AWAPART   AWAPART  0.00000000
## AWABEDR   AWABEDR  0.00000000
## AWALAND   AWALAND  0.00000000
## ABESAUT   ABESAUT  0.00000000
## AMOTSCO   AMOTSCO  0.00000000
## AVRAAUT   AVRAAUT  0.00000000
## AAANHANG AAANHANG  0.00000000
## ATRACTOR ATRACTOR  0.00000000
## AWERKT     AWERKT  0.00000000
## ABROM       ABROM  0.00000000
## ALEVEN     ALEVEN  0.00000000
## APERSONG APERSONG  0.00000000
## AGEZONG   AGEZONG  0.00000000
## AWAOREG   AWAOREG  0.00000000
## ABRAND     ABRAND  0.00000000
## AZEILPL   AZEILPL  0.00000000
## APLEZIER APLEZIER  0.00000000
## AFIETS     AFIETS  0.00000000
## AINBOED   AINBOED  0.00000000
## ABYSTAND ABYSTAND  0.00000000
```

The most influentual predictors appear to be PPERSAUT and MKOOPKLA.

4c

```
bag.caravan <- randomForest(Purchase~., data=train.caravan, importance=TRUE)
bag.caravan
```

```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = train.caravan, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 7%
## Confusion matrix:
##      No Yes class.error
## No  928   7 0.007486631
## Yes  63   2 0.969230769
```

```
importance(bag.caravan)
```

```
##                   No         Yes MeanDecreaseAccuracy MeanDecreaseGini
## MOSTYPE   2.3421285  0.94021736         2.5744156447      4.070344723
```

```
## MAANTHUI -1.4838561  1.73328096        -0.7790632711      0.379597192
## MGEMOMV   2.7937643  0.66865373         3.0743821964      1.616254127
## MGEMLEEF  2.5904019 -1.58547950         2.0751231504      1.386138568
## MOSHOOFD  1.7891520  2.67058936         2.7772886257      2.369711826
## MGODRK    0.4185954 -0.22259799         0.3431646374      1.319362018
## MGODPR    3.8189705  0.38648686         3.7074373866      2.361634527
## MGODOV    3.1091110  0.37690827         3.1899630421      1.614947673
## MGODGE    2.0933391  2.49618845         2.7932974793      2.260762333
## MRELGE    4.6500271 -0.57393014         4.2520184503      2.094966524
## MRELSA    2.5294556 -1.61310700         1.8380736289      1.253608721
## MRELOV    5.1802824 -2.99947837         4.3412483302      1.651051815
## MFALLEEN  5.2415179 -1.70508785         4.5350108423      1.920710372
## MFGEKIND  0.7225648 -1.14210277         0.4236879966      2.450995528
## MFWEKIND  1.9871325  0.37981898         1.9281631099      2.323515639
## MOPLHOOG  4.3295055  1.19647878         4.7104452676      2.347856247
## MOPLMIDD  2.6653764  0.04474324         2.6873836819      3.085787935
## MOPLLAAG  8.0840089 -0.69004608         8.2018316681      2.771578056
## MBERHOOG  3.3772631 -2.91573732         2.8409400228      1.897806384
## MBERZELF -0.1220701 -0.62196791        -0.3497124530      0.996718453
## MBERBOER -1.6077343 -0.82575647        -1.8746207171      0.740178615
## MBERMIDD  3.8149075  2.40290392         4.2972673253      2.647909531
## MBERARBG  3.7041170  1.48297761         4.2633611004      2.561965684
## MBERARBO  3.4950304 -0.69133495         3.1758378861      2.131918443
## MSKA      2.4773276 -0.83876062         2.2784080756      2.094292344
## MSKB1     2.8103911 -0.78196940         2.4789529838      2.337110943
## MSKB2     3.0662496 -0.85411900         2.8175878973      2.136740193
## MSKC      4.5046266 -3.64819321         3.6117693485      2.515670361
## MSKD      0.5447871  0.37747264         0.6577924001      1.647480101
## MHHUUR    5.1698122 -0.29067746         5.1370680442      2.464008282
## MHKOOP    4.2370336 -0.80454353         4.2704811583      2.458497560
## MAUT1     4.5502729 -0.93283143         4.1986161196      2.011605037
## MAUT2     0.1322512  0.30041963         0.1788141088      1.694954637
## MAUT0     5.0963093  0.09420958         5.1249035615      2.247137160
## MZFONDS   2.9406652 -1.76684013         2.4557014693      1.996604995
## MZPART    3.6832340 -1.38466681         3.1646829415      2.058295417
## MINKM30   3.1541594 -2.16770393         2.5070884264      2.163708387
## MINK3045  3.5645309 -1.06162860         3.2668534546      2.072394245
## MINK4575  2.1691777  0.45340950         2.2777493139      1.985103690
## MINK7512  6.2283023  3.54186940         7.1896471464      2.305397173
## MINK123M  0.1059064 -2.17470053        -0.5053684748      0.612345216
## MINKGEM   2.0816218  0.03040747         2.1212321562      2.006924349
## MKOOPKLA  3.8626463  0.40261529         4.1578874247      2.305600333
## PWAPART   0.9014435  3.84146142         2.1361495086      2.766305111
## PWABEDR   0.0000000  0.00000000         0.0000000000      0.084874254
## PWALAND  -1.1598952 -1.00100150        -1.2914171756      0.128946151
## PPERSAUT  1.3169028  5.58866006         2.8980304884      3.118772907
## PBESAUT   1.0010015  0.00000000         1.0010015025      0.004600000
```

```
## PMOTSCO  -0.6448854  1.42664895        -0.3134657701      0.822599176
## PVRAAUT   0.0000000  0.00000000         0.0000000000      0.000000000
## PAANHANG  2.1768711 -0.23802062         2.0591845982      0.204461224
## PTRACTOR  1.9082270  0.00000000         1.9052161663      0.068745894
## PWERKT    0.0000000  0.00000000         0.0000000000      0.000000000
## PBROM     0.3373802  1.41669934         0.8212070515      0.082955720
## PLEVEN    0.5855650 -1.37962146         0.2898061713      0.388081691
## PPERSONG  0.0000000  0.00000000         0.0000000000      0.067680768
## PGEZONG   1.2820010  1.00100150         1.3908293751      0.277502034
## PWAOREG   0.0000000  0.00000000         0.0000000000      0.022279121
## PBRAND    3.3180972  7.76335994         5.6501578513      3.858450775
## PZEILPL  -1.0189423  0.00000000        -1.0136528089      0.208286669
## PPLEZIER  9.0583103  9.06705677         9.8076151346      1.555898437
## PFIETS    1.9872577 -1.02977837         1.6757965453      0.578395779
## PINBOED   2.2052754 -0.14960742         2.1407604358      0.487806098
## PBYSTAND  2.5868015  1.93844996         2.9945825335      0.876517296
## AWAPART   0.4368320  3.31439078         1.5694300438      1.467116940
## AWABEDR   1.4164094  0.00000000         1.4167755456      0.095988048
## AWALAND  -2.4983232 -1.40383622        -2.7619714563      0.119655808
## APERSAUT  1.0218435  3.21303431         1.9316742460      2.743581473
## ABESAUT   0.0000000  0.00000000         0.0000000000      0.008666667
## AMOTSCO   0.4809330  2.00788540         1.0723226601      0.569052333
## AVRAAUT   0.0000000  0.00000000         0.0000000000      0.006533333
## AAANHANG -0.8315700 -1.41614401        -1.1553164211      0.136921658
## ATRACTOR -1.0010015  0.00000000        -1.0010015025      0.068325536
## AWERKT    0.0000000  0.00000000         0.0000000000      0.000000000
## ABROM    -0.2695665  0.00000000        -0.2718093501      0.080701516
## ALEVEN   -2.9385963 -1.11540459        -2.9880885813      0.594582558
## APERSONG  0.0000000  0.00000000         0.0000000000      0.040192857
## AGEZONG   1.8335098  0.00000000         1.8241493863      0.350745388
## AWAOREG   0.0000000  0.00000000         0.0000000000      0.003866667
## ABRAND    1.0776590  4.36486907         2.2260306701      1.475970585
## AZEILPL   0.3150246 -1.00100150         0.0003440486      0.253273786
## APLEZIER  8.1074043  7.90047757         8.4289358963      1.287734424
## AFIETS    1.5792977 -0.88567858         1.2013882502      0.535590513
## AINBOED   4.0825936 -1.16061083         3.6563523970      0.515978313
## ABYSTAND  3.0326674 -1.31305155         2.7450279926      0.394742564
```

The out-of-bag estimate error is 4.7%. There was 9 variables tried at each split. There were 500 trees used to fit the data.

4d

```
yhat.bag = predict(bag.caravan, newdata = test.caravan, type="prob")
yhat.bag  = data.frame(yhat.bag[,2])
names(yhat.bag)[1] = "Probability"
yhat.bag = yhat.bag %>% mutate(Probability=as.factor(ifelse(Probability<=0.2, "No","Y
es")))
bag.err = table(pred = yhat.bag$Probability, truth = test.caravan$Purchase)
test.bag.err = 1 - sum(diag(bag.err))/sum(bag.err)
test.bag.err
```

```
## [1] 0.1041062
```

```
yhat.boost = predict(boost.caravan, newdata = test.caravan, type="response")
```

```
## Using 1000 trees...
```

```
yhat.boost = data.frame(yhat.boost)
yhat.boost = yhat.boost %>% mutate(Probability=as.factor(ifelse(yhat.boost<=0.2,"No",
"Yes")))
boost.err = table(pred = yhat.boost$Probability, truth = test.caravan$Purchase)
test.boost.err = 1 - sum(diag(boost.err))/sum(boost.err)
test.boost.err
```

```
## [1] 0.07693903
```

```
boost.err
```

```
##      truth
## pred    No  Yes
##   No  4411  243
##   Yes  128   40
```

Out of 304 people who made a purchase 41 were predicted correctly which is 13.486%

Question 5

```
drug_use <- read_csv('drug.csv',
col_names = c('ID','Age','Gender','Education','Country','Ethnicity',
'Nscore','Escore','Oscore','Ascore','Cscore','Impulsive',
'SS','Alcohol','Amphet','Amyl','Benzos','Caff','Cannabis',
'Choc','Coke','Crack','Ecstasy','Heroin','Ketamine','Legalh','LSD',
'Meth', 'Mushrooms', 'Nicotine', 'Semer','VSA'))
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   .default = col_character(),
##   ID = col_double(),
##   Age = col_double(),
##   Gender = col_double(),
##   Education = col_double(),
##   Country = col_double(),
##   Ethnicity = col_double(),
##   Nscore = col_double(),
##   Escore = col_double(),
##   Oscore = col_double(),
##   Ascore = col_double(),
##   Cscore = col_double(),
##   Impulsive = col_double(),
##   SS = col_double()
## )
## ℹ Use `spec()` for the full column specifications.
```

5a

```
drug_use <- drug_use %>% mutate(recent_cannabis_use=as.factor(ifelse(Cannabis>="CL3",
"No", "Yes")))
drug_use_subset <- drug_use %>% select(Age:SS, recent_cannabis_use)
train <- sample(1:nrow(drug_use_subset), 1500)
train.drug <- drug_use[train,]
test.drug <- drug_use[-train,]

svmfit=svm(recent_cannabis_use~Age+SS, data=train.drug, kernel="radial", cost=1,scale
=FALSE)
ypred=predict(svmfit,test.drug)
table(predict=ypred, truth=test.drug$recent_cannabis_use)
```

```
##        truth
## predict  No Yes
##     No  151  49
##     Yes  43 142
```

5b

```
set.seed(1)
tune.out=tune(svm,recent_cannabis_use~Age+SS, data=train.drug, kernel="radial",
ranges=list(cost=c(0.001, 0.01, 0.1, 1, 10, 100)))
summary(tune.out)$"best.model"
```

```
##
## Call:
## best.tune(method = svm, train.x = recent_cannabis_use ~ Age + SS,
##     data = train.drug, ranges = list(cost = c(0.001, 0.01, 0.1, 1,
##         10, 100)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##
## Number of Support Vectors:  1197
```

```
drug.err <- table(true=train.drug$recent_cannabis_use, pred=predict(tune.out$best.mod
el,newdata=train.drug))
train.drug.err = 1 - sum(diag(drug.err))/sum(drug.err)
train.drug.err
```

```
## [1] 0.2486667
```

The best cost for this model is 0.01 and the cross validated training error is 24.933%