Homework Assignment

Ted Tinker and Blake Shaw

December 03, 2017

Exercise 1

Part A

If we randomly sample n observations from a set of n observations with replacement, what is the probability that an observation j is not included in the sample? This would imply that each of n individual samples was one of the $n-1$ observations which was not j. So, the probability j is not included should be $\left(\frac{n-1}{n}\right)^n$.

Part B

```
(999/1000)^1000
```

```
## [1] 0.3677
```

Part C

```
numbers <- c(1:1000)       # Integers 1-1000 for bootstrapping
set.seed(999)              # For reproducibility
unlist(lapply(1:10, function(x) {bootstrap <- sample(numbers,replace=TRUE)
                           # Make ten bootstraps
                           (length(bootstrap) - length(unique(bootstrap)))/length(bootstrap)}))
```

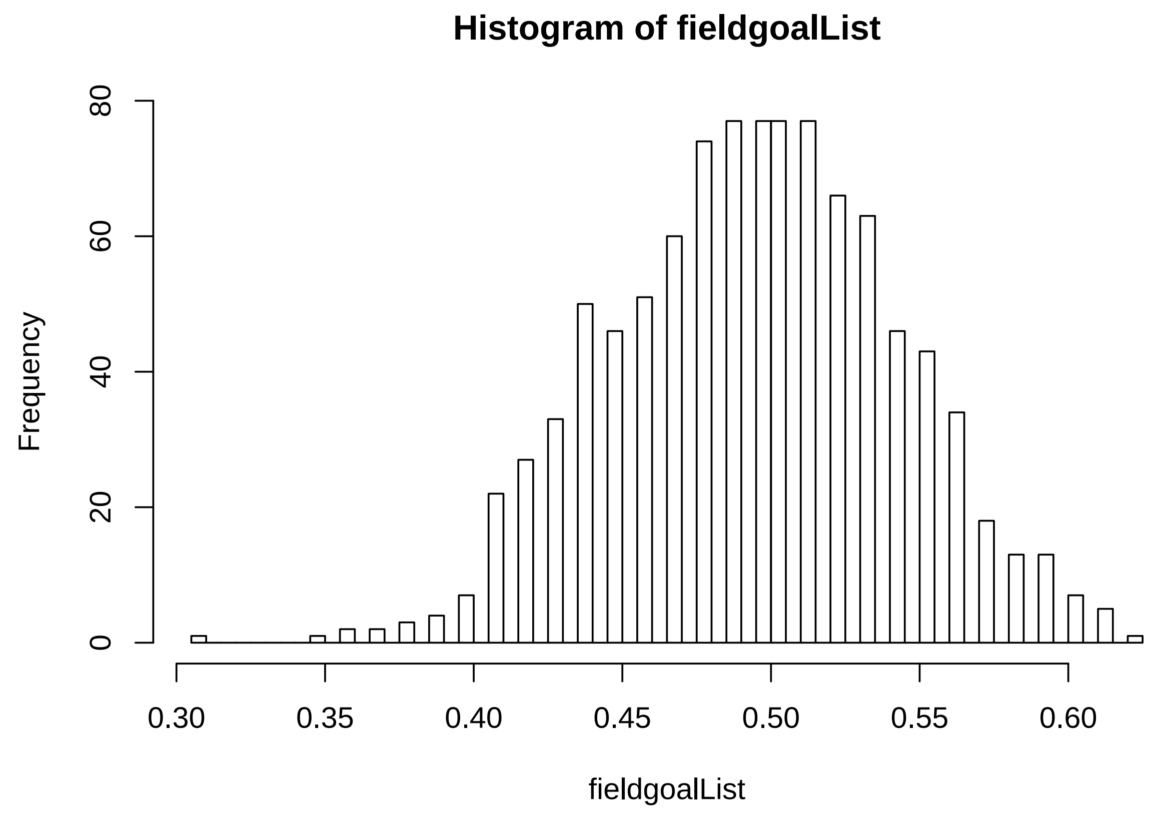
```
##  [1] 0.362 0.377 0.366 0.371 0.364 0.372 0.337 0.380 0.375 0.387
```

```
                           # Print % missing
```

These samples line up very well with the expression produced above.

Part D

```
shots <- c(rep(0,51),rep(1,50))    # 51 misses, 50 baskets
set.seed(999)                      # For reproducibility
fieldgoalList <- unlist(lapply(1:1000, function(x) {fieldgoals <- sample(shots,replace=TRUE)
                                                    sum(fieldgoals)/length(fieldgoals)}))
fieldHist <- hist(fieldgoalList,breaks=100) # Plot field goal percentages as histogram
```

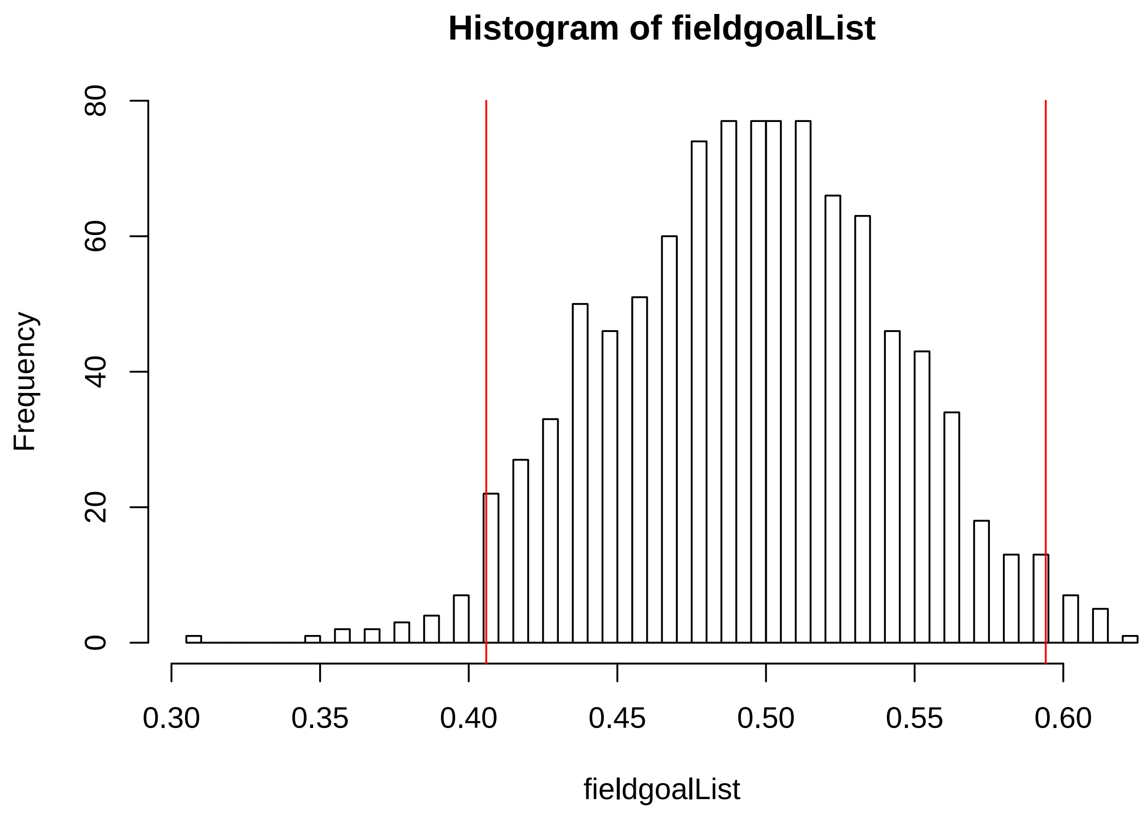


I notice the data is slightly skewed left.

```
(range <- quantile(fieldgoalList, probs=c(.025,.975)))   # 95% confidence interval
```

```
##   2.5%  97.5%
## 0.4059 0.5941
```

```
plot(fieldHist)
abline(v = .4059,col="Red")
abline(v = .5941,col="Red")      # Histogram plus confidence interval bounds
```



The long tail on the lest of the histogram provides statistical reason to suspect that Covington's true field goal percentage is lower than .495. The law of large numbers would suggest his abnormally high field goal percentage this quarter is an anomaly.

Exercise 2

```
setwd("C:\\Users\\Theodore\\Desktop\\R_Studio_Stuff\\data")
# setwd("C:\\Users\\Blake\\Documents\\PSTAT 131\\drug.csv")
     # Blake: switch this to your directory, then we can just trade off which is commented
load("faces_array.RData")
face_mat <- sapply(1:1000, function(i) as.numeric(faces_array[ , , i])) %>% t
     # Load and prepare data

plot_face <- function(image_vector)    # Function to display one picture
  {plot(as.cimg(t(matrix(image_vector, ncol=100))), axes=FALSE, asp=1)}

set.seed(999)
plot_face(face_mat[sample(1000, 1),])  # Display a random photo as a test
```

Part A

```r
plot_face(sapply(1:10000, function (x) {mean(face_mat[,x])}))
```

```
    # For each of 10,000 pixels, find the average
```

An eerie human face is visible as if through fog. It resembles the killer's mask in Halloween.

Part B

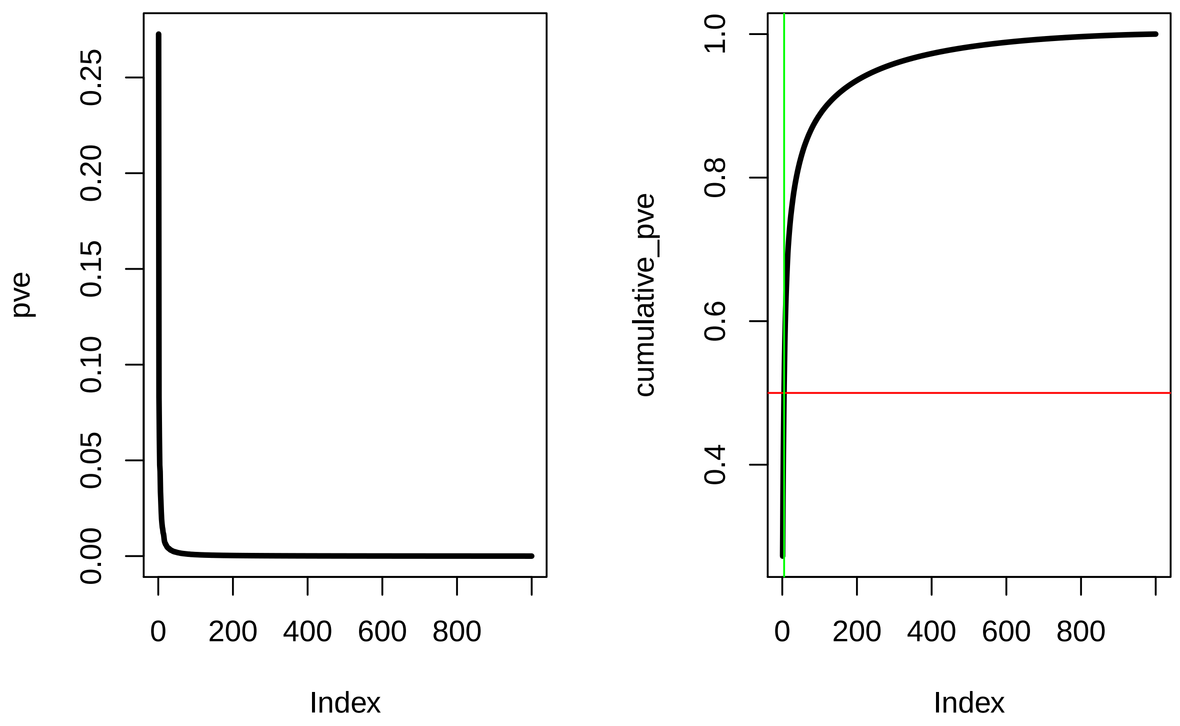
```
face.PCA <- prcomp(face_mat[,1:10000],center=TRUE,scale=FALSE)
face.PCA      # This chunk takes ages; cache it to save time
```

```
pve <- (face.PCA$sdev^2) / sum(face.PCA$sdev^2)                    # Calculate pves
cumulative_pve <- cumsum(face.PCA$sdev^2) / sum(face.PCA$sdev^2)   # Cumulative pves

cumulative_pve[1:5]        # Only 5 columns are needed to account for 50%
```

```
## [1] 0.2727 0.3556 0.4179 0.4655 0.5097
```

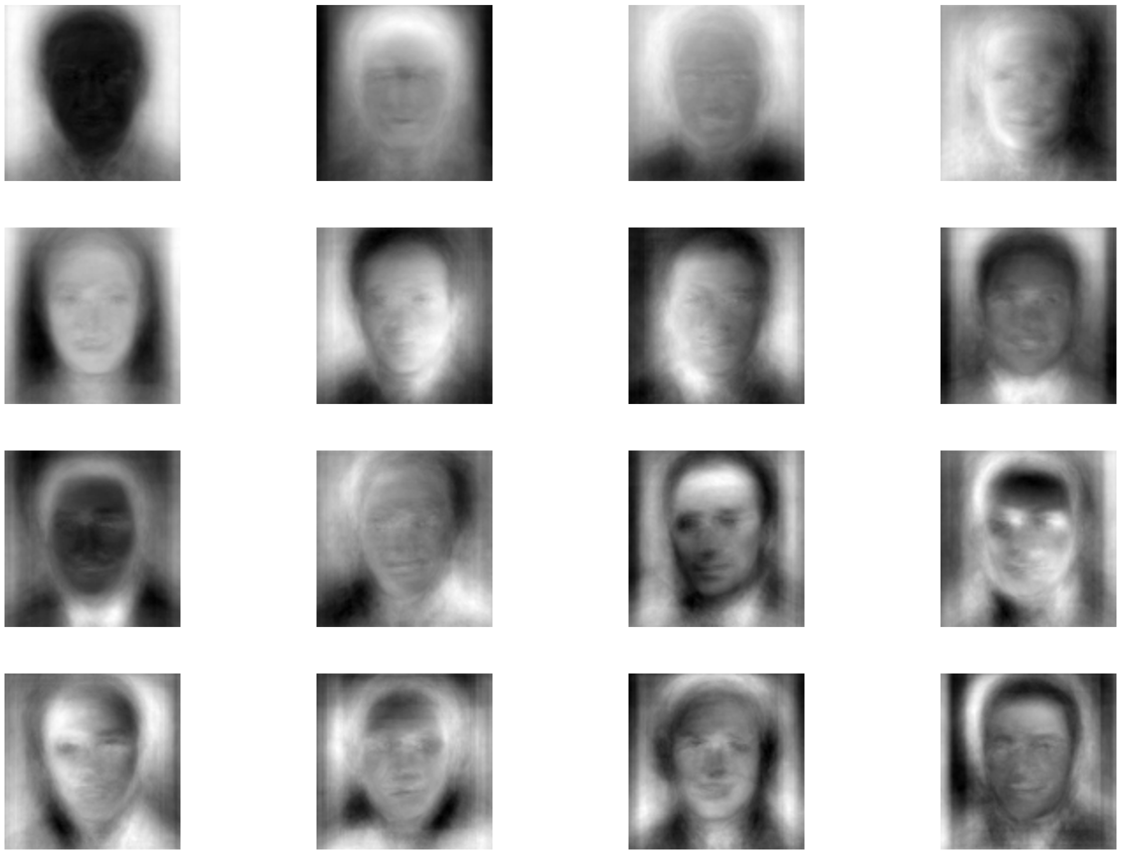
```
par(mfrow=c(1, 2))         # Side-by-side plots
plot(pve, type="l", lwd=3)
plot(cumulative_pve, type="l", lwd=3)
abline(h=.5,col="Red")     # Add line at 50% variance explained
abline(v=5,col="Green")    # Add line at 5 columns
```



Only 5 principle components are necessary to account for 50% of the data's variance.

Part C

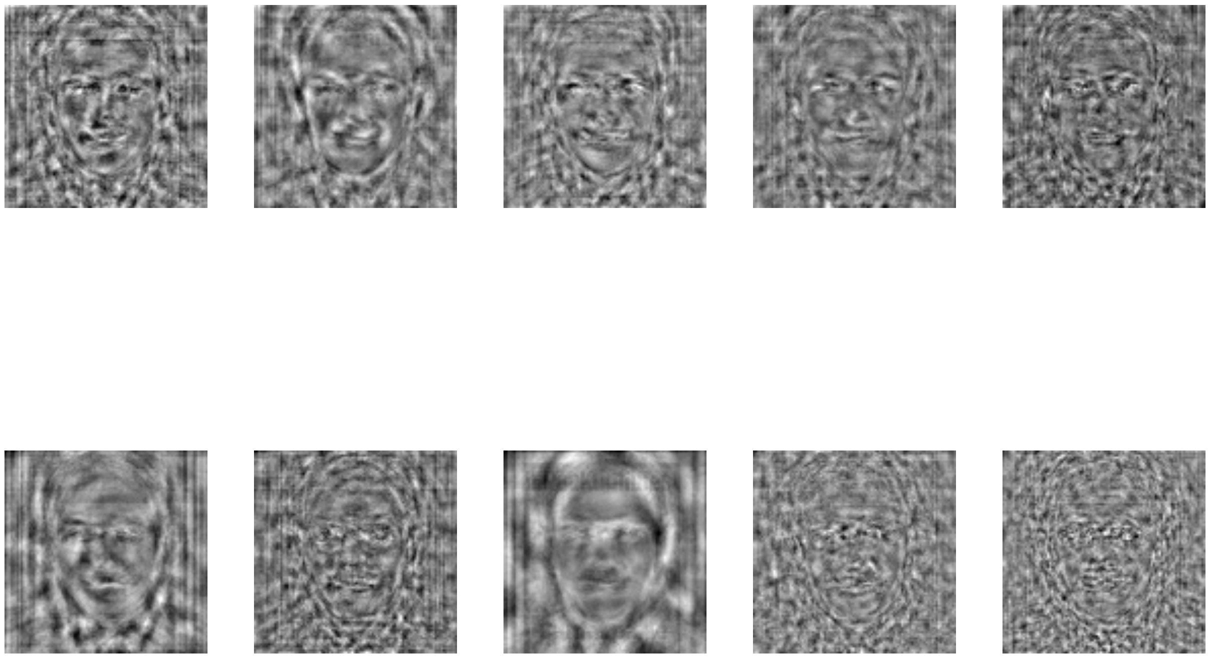
```r
par(mar=c(1,1,1,1))
par(mfrow=c(4,4))  # Code provided for plotting 16 faces
for (i in c(1:16)) {plot_face(face.PCA$rotation[,i])}
```



I find it interesting that the principle components become more 'precise' as the index increases, in that they seem to specify a particular person instead of a vaguely face-like blob. Much of the contrast seems to arise from differences in hair-style, skin-color, and lighting.

Part D

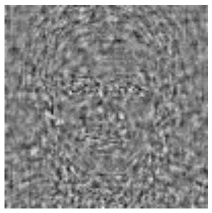
```
par(mar=c(1,1,1,1))
par(mfrow=c(2,5))    # Prepare plots for 2*5 image output
highList <- face.PCA$rotation[,head(order(face.PCA$rotation[1,],decreasing=TRUE),5)]
lowList <- face.PCA$rotation[,head(order(face.PCA$rotation[1,],decreasing=FALSE),5)]
     # Make list of primary components with top/bottom 5 values in first entry
for(i in c(1:5)) {plot_face(highList[,i])}
for(i in c(1:5)) {plot_face(lowList[,i])}
```

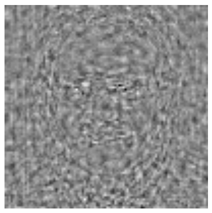


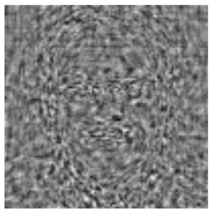
It seems the first value in each primary component corresponds to particular areas of contrast around the facial features. The area outside an ovular face-region is noisy with no clear pattern.

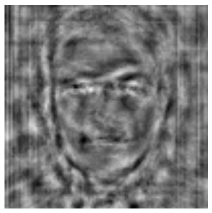
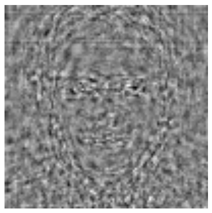
Part E

```
par(mar=c(1,1,1,1))
par(mfrow=c(2,5))    # Prepare plots for 2*5 image output
highList <- face.PCA$rotation[,head(order(face.PCA$rotation[5,],decreasing=TRUE),5)]
lowList <- face.PCA$rotation[,head(order(face.PCA$rotation[5,],decreasing=FALSE),5)]
     # Make list of primary components with top/bottom 5 values in fifth entry
for(i in c(1:5)) {plot_face(highList[,i])}
for(i in c(1:5)) {plot_face(lowList[,i])}
```







These images are much noisier than the previous examples. Interestingly a few faces appear amidst the noise (the clearest being on the right of the bottom row), perhaps justifying the explanatory purpose of this fifth value. However, the first value would seem more useful in reliably and consistantly identifying faces.

Exercise 3

Part A

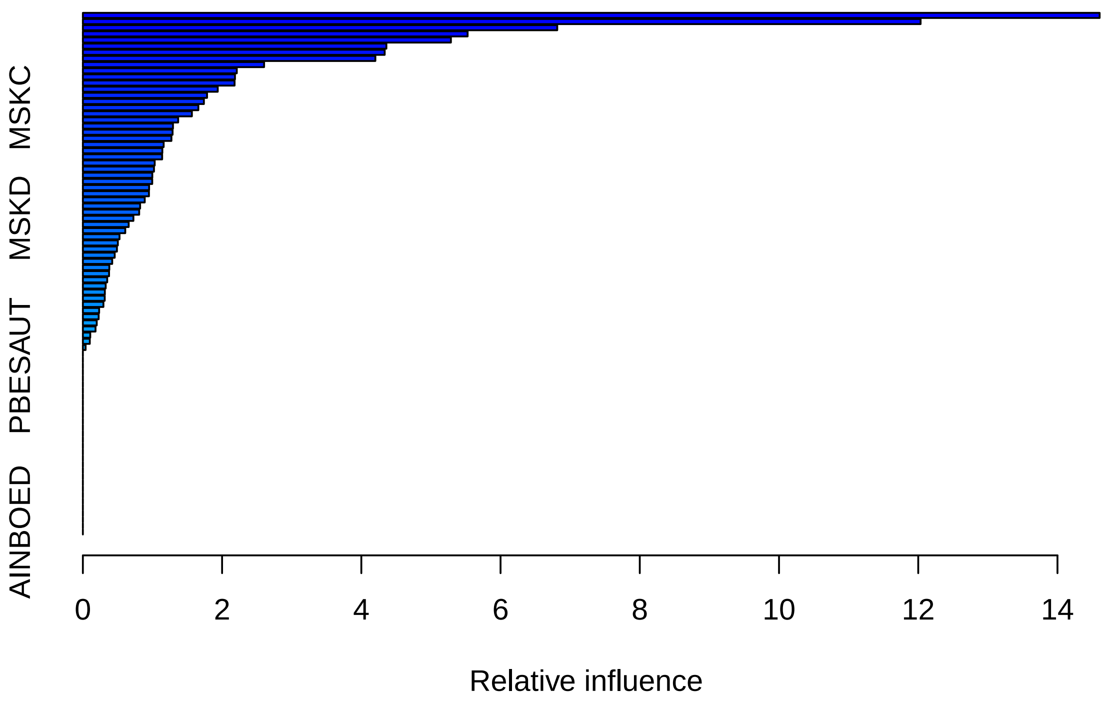
```r
library(ISLR)
train = 1:1000
Caravan$Purchase = ifelse(Caravan$Purchase == "Yes", 1, 0)
Caravan.train = Caravan[train,]
Caravan.test = Caravan[-train,]    # Load Caravan set, divide into test and training
```

Part B

```r
set.seed(999)
boosting.model = gbm(Purchase ~ ., data = Caravan.train, n.trees = 1000, shrinkage = 0.01,
                     distribution = "bernoulli")  # Make boosted model
```

```
## Warning in gbm.fit(x, y, offset = offset, distribution = distribution, w =
## w, : variable 50: PVRAAUT has no variation.
## Warning in gbm.fit(x, y, offset = offset, distribution = distribution, w =
## w, : variable 71: AVRAAUT has no variation.
```

```r
summary(boosting.model)
```



```
##                var  rel.inf
## PPERSAUT PPERSAUT 14.60577
## MKOOPKLA MKOOPKLA 12.03377
## MOPLHOOG MOPLHOOG  6.81441
## MBERMIDD MBERMIDD  5.52684
## PBRAND     PBRAND  5.28693
## MGODGE     MGODGE  4.35971
## MINK3045 MINK3045  4.33585
## ABRAND     ABRAND  4.20173
## MBERARBG MBERARBG  2.60325
## MAUT2       MAUT2  2.21091
## MAUT1       MAUT1  2.18492
## PWAPART   PWAPART  2.17940
## MOSTYPE   MOSTYPE  1.93764
## MGODPR     MGODPR  1.78382
## MBERHOOG MBERHOOG  1.73855
## MSKC         MSKC  1.65907
## MSKA         MSKA  1.56630
## MINK4575 MINK4575  1.36912
## MRELGE     MRELGE  1.29458
## MINKGEM   MINKGEM  1.28868
## MSKB1       MSKB1  1.27185
## MAUT0       MAUT0  1.16080
## MHKOOP     MHKOOP  1.14149
## MGODOV     MGODOV  1.13888
## MGODRK     MGODRK  1.03186
## MFGEKIND MFGEKIND  1.02167
## MRELOV     MRELOV  0.99727
## PBYSTAND PBYSTAND  0.99641
## MINKM30   MINKM30  0.95162
## MOPLMIDD MOPLMIDD  0.94922
## MGEMLEEF MGEMLEEF  0.88903
## MFWEKIND MFWEKIND  0.82253
## ABYSTAND ABYSTAND  0.80888
## MSKD         MSKD  0.72603
## MINK7512 MINK7512  0.65828
## APERSAUT APERSAUT  0.60913
## MBERBOER MBERBOER  0.52653
## MHHUUR     MHHUUR  0.50246
## MBERARBO MBERARBO  0.48946
## MINK123M MINK123M  0.45756
## AMOTSCO   AMOTSCO  0.42141
## MZPART     MZPART  0.37980
## MGEMOMV   MGEMOMV  0.37617
## MRELSA     MRELSA  0.34974
## PMOTSCO   PMOTSCO  0.32794
## AWAPART   AWAPART  0.31597
## PLEVEN     PLEVEN  0.31420
## MSKB2       MSKB2  0.29534
## ALEVEN     ALEVEN  0.23354
## MOSHOOFD MOSHOOFD  0.22711
## MZFONDS   MZFONDS  0.19924
## MOPLLAAG MOPLLAAG  0.18187
## MFALLEEN MFALLEEN  0.10617
## MBERZELF MBERZELF  0.09973
## MAANTHUI MAANTHUI  0.03952
## PWABEDR   PWABEDR  0.00000
## PWALAND   PWALAND  0.00000
## PBESAUT   PBESAUT  0.00000
## PVRAAUT   PVRAAUT  0.00000
## PAANHANG PAANHANG  0.00000
## PTRACTOR PTRACTOR  0.00000
## PWERKT     PWERKT  0.00000
## PBROM       PBROM  0.00000
## PPERSONG PPERSONG  0.00000
## PGEZONG   PGEZONG  0.00000
## PWAOREG   PWAOREG  0.00000
## PZEILPL   PZEILPL  0.00000
## PPLEZIER PPLEZIER  0.00000
## PFIETS     PFIETS  0.00000
## PINBOED   PINBOED  0.00000
## AWABEDR   AWABEDR  0.00000
## AWALAND   AWALAND  0.00000
## ABESAUT   ABESAUT  0.00000
## AVRAAUT   AVRAAUT  0.00000
## AAANHANG AAANHANG  0.00000
## ATRACTOR ATRACTOR  0.00000
## AWERKT     AWERKT  0.00000
## ABROM       ABROM  0.00000
## APERSONG APERSONG  0.00000
## AGEZONG   AGEZONG  0.00000
## AWAOREG   AWAOREG  0.00000
## AZEILPL   AZEILPL  0.00000
## APLEZIER APLEZIER  0.00000
## AFIETS     AFIETS  0.00000
## AINBOED   AINBOED  0.00000
```

PPERSAUT, MKOOPLA, and MOPLHOOG have the three highest relative influences so they are the most important variables.

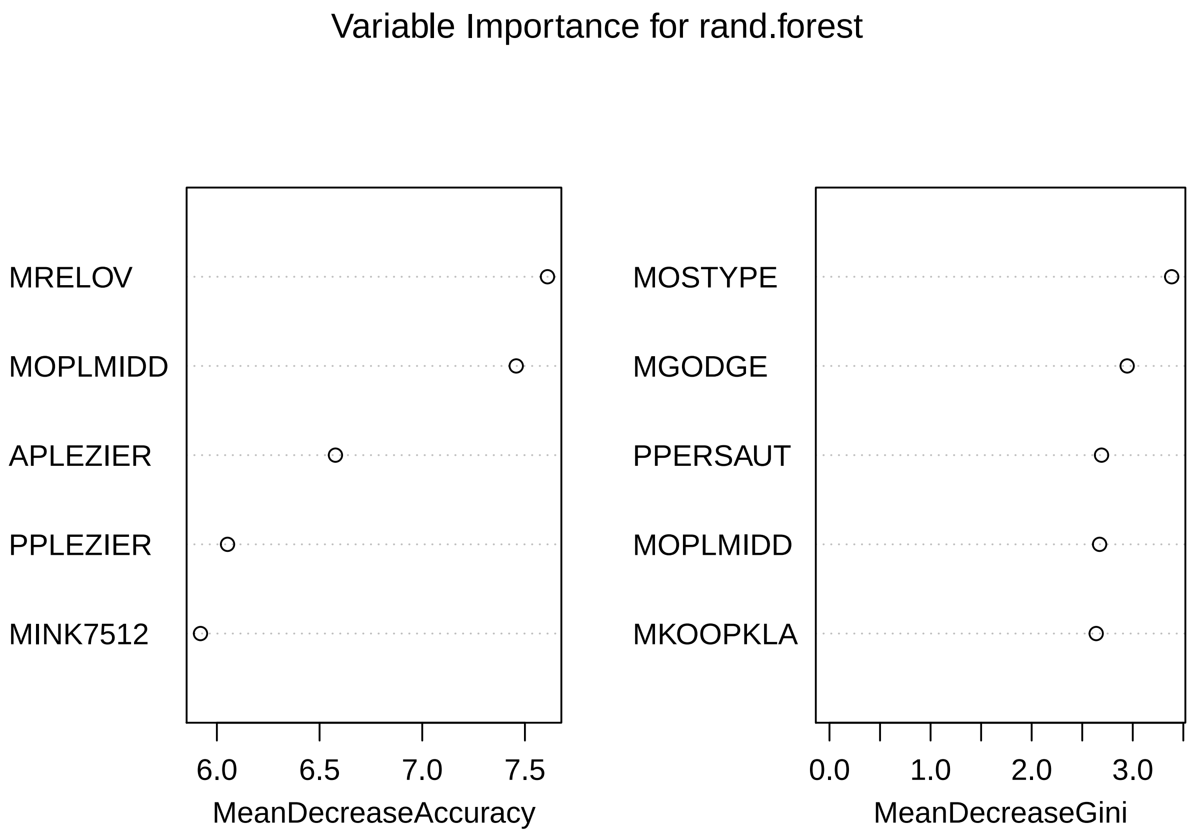
Part C

```r
set.seed(999)
rand.forest <- randomForest(factor(Purchase) ~ ., data = Caravan.train, importance = T)
rand.forest     # Make and display random forest
```

```
##
## Call:
##  randomForest(formula = factor(Purchase) ~ ., data = Caravan.train,      importance = T)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 6.2%
## Confusion matrix:
##     0 1 class.error
## 0 936 5    0.005313
## 1  57 2    0.966102
```

OOB estimate of error rate: 5.9%
No. of variables tried at each split: 9 Number of trees: 500

```r
varImpPlot(rand.forest, sort=T, main="Variable Importance for rand.forest", n.var=5)
```



The order of important variables similar for both boosting and random forest models. MOPLHOOG is the highest variable of impotance in terms of model accuracy for the random forest model and is the second highest variable in terms of relative influence for the boosting model. PPERSAUT and MKOOPKLA are both listed high as gini values for the random forest model and are listed high as variables in terms of relative influence for the the boosting model.

Part D

```
boost.prob = predict(boosting.model, Caravan.test, n.trees = 1000, type = "response")
boost.pred = ifelse(boost.prob > 0.2, 1, 0)
table(Caravan.test$Purchase, boost.pred)     # Confusion matrix for boosting
```

```
##    boost.pred
##        0    1
##   0 4408  125
##   1  257   32
```

```
rand.forest.pred= predict(rand.forest, Caravan.test, type = "class")
table(Caravan.test$Purchase, rand.forest.pred)      # Conf matrix for forest
```

```
##    rand.forest.pred
##        0    1
##   0 4499   34
##   1  280    9
```

```
TPR.rand.forest <- 9 / (9 + 34)     # Calculate TPR
TPR.rand.forest
```

```
## [1] 0.2093
```

Exercise 4

```
setwd("C:\\Users\\Theodore\\Desktop\\R_Studio_Stuff\\data")
# setwd("C:\\Users\\Blake\\Documents\\PSTAT 131")

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'))
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   ID = col_integer(),
##   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()
## )

## See spec(...) for full column specifications.
```

```
# After reading in drug_use, format the data as we did on the midterm:

drug_use <- drug_use %>% mutate_at(as.ordered, .vars=vars(Alcohol:VSA))
drug_use <- drug_use %>%
mutate(Gender = factor(Gender, labels=c("Male", "Female"))) %>%
mutate(Ethnicity = factor(Ethnicity, labels=c("Black", "Asian", "White",
"Mixed:White/Black", "Other",
"Mixed:White/Asian",
"Mixed:Black/Asian"))) %>%
mutate(Country = factor(Country, labels=c("Australia", "Canada", "New Zealand",
"Other", "Ireland", "UK", "USA")))

# Then add recent cannabis use column:

drug_use <- drug_use %>%
mutate(recent_cannabis_use = ifelse(Cannabis=="CL0" | Cannabis=="CL1" | Cannabis=="CL2", "No", "Yes"))
mutate(recent_cannabis_use = factor(recent_cannabis_use, labels = c("No", "Yes")))
```

```
drug_use <- select(drug_use,Age:SS, recent_cannabis_use)    # Choose columns
```

Part A

```
smp_size <- floor((1500 / 1885) * nrow(drug_use))
set.seed(999)
train_ind <- sample(seq_len(nrow(drug_use)), size = smp_size)

train.drug <- drug_use[train_ind, ]
test.drug <- drug_use[-train_ind, ]    # Set training and test data

svm.model <- svm(recent_cannabis_use ~ ., data = train.drug, kernel = "radial", cost = 1)
     # Make support vector machine
table(test.drug$recent_cannabis_use, predict(svm.model,test.drug[,-13]))
```

```
##
##          No Yes
##    No   141  40
##    Yes   39 165
```

```
     # SVM conf matrix
```

Part B

```
tune.out <- tune(svm, recent_cannabis_use ~ ., data = train.drug, kernel = "radial", ranges =          li
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1847
##
## - Detailed performance results:
##    cost  error dispersion
## 1 1e-03 0.4700    0.04568
## 2 1e-02 0.2073    0.02095
## 3 1e-01 0.1920    0.01565
## 4 1e+00 0.1847    0.03556
## 5 1e+01 0.1973    0.03600
## 6 1e+02 0.2167    0.02378
```

The error seems to be minimized when the cost equals 1. Since this was the cost of the first Support Vector Machine we generated, the best model's confusion matrix will be identical to the previous one:

```
table(test.drug$recent_cannabis_use, predict(tune.out$best.model,test.drug[,-13]))
```

```
##
##        No Yes
##   No  141  40
##   Yes  39 165
```

```
     # Best SVM conf matrix
```

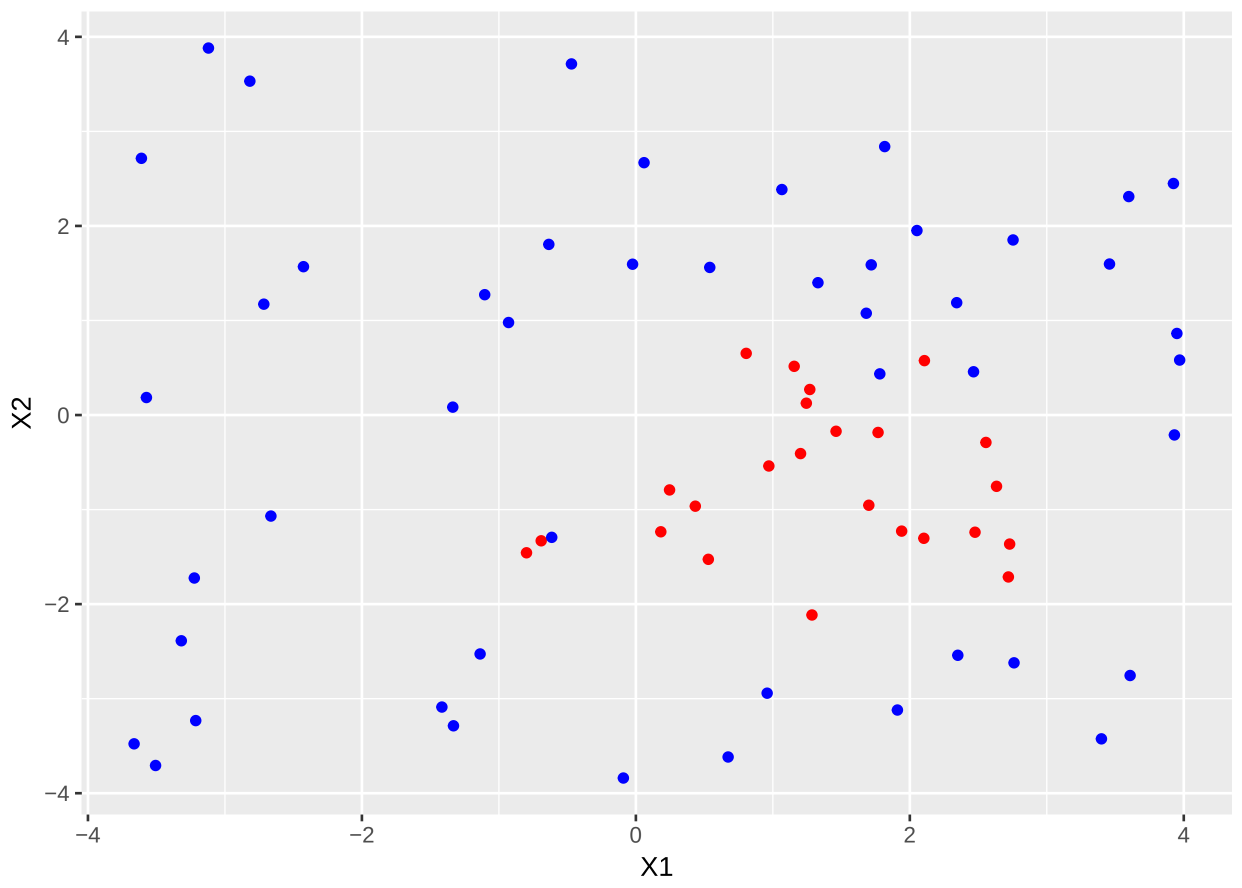
Exercise 5

Part A

```
setwd("C:\\Users\\Theodore\\Desktop\\R_Studio_Stuff\\data")
# setwd("C:\\Users\\Blake\\Documents\\PSTAT 131")
dat <- read_csv("nonlinear.csv")
```

```
## Parsed with column specification:
## cols(
##   Index = col_integer(),
##   X1 = col_double(),
##   X2 = col_double(),
##   Y = col_integer()
## )
```

```
ggplot() +
     geom_point(data=dat[1:24,],aes(x=X1,y=X2),color="red") +
     geom_point(data=dat[25:72,],aes(x=X1,y=X2),color="blue")
```



```
    # Plot points where Y=0 in red, and points where Y=1 in blue
```

Observations of type $Y = 0$ seem to be clustered more closely than observations of type $Y = 1$.

Part B

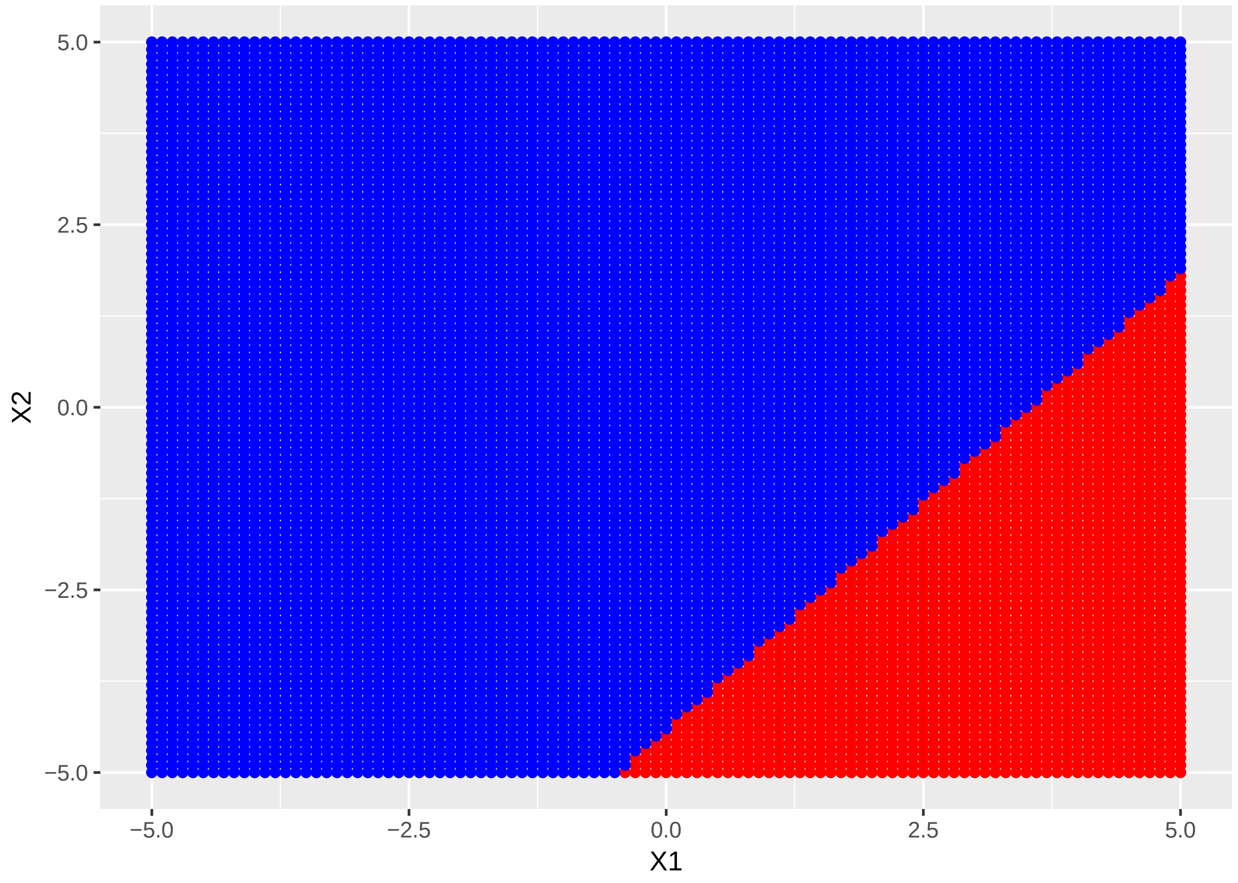
```
# grid of points over sample space
gr <- expand.grid(X1=seq(-5, 5, by=0.1), # sample points in X1
X2=seq(-5, 5, by=0.1)) # sample points in X2

model <- glm(Y~X1+X2,family=binomial(link='logit'),data=dat)

predictions <- predict(model,gr,type="response")    # Predictions in logit form
predictions <- sapply(1:length(predictions), function (x) {
     if(predictions[[x]]>=.5) {predictions[[x]]=1}
     else {predictions[[x]]=0}})
     # Threshold: when the odds are 1-1 or better, assume Y=1. Otherwise 0.

gr["predictions"] <- predictions
gr.0 <- gr[predictions==0,]
gr.1 <- gr[predictions==1,]

ggplot() +
     geom_point(data=gr.0,aes(x=X1,y=X2),color="red") +
    geom_point(data=gr.1,aes(x=X1,y=X2),color="blue")
```



```
    # Plot the prediction of each grid point
```

As a linear model, a straight line cannot totally capture the clustering we observed in part a.

Part C

```
model2 <- glm(Y~poly(X2,5)+poly(X1,2),family=binomial(link='logit'),data=dat)
```

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

```
summary(model2)
```

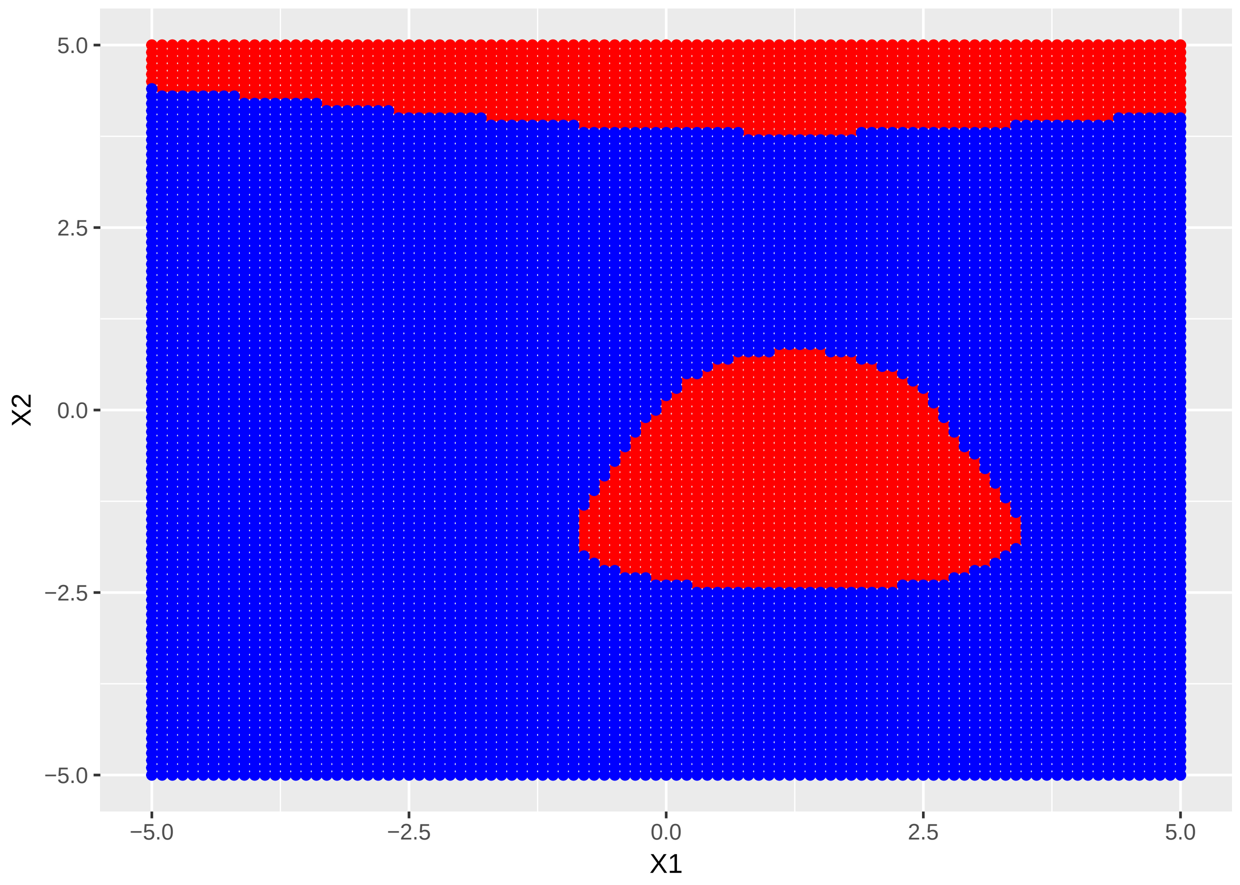
```
##
## Call:
## glm(formula = Y ~ poly(X2, 5) + poly(X1, 2), family = binomial(link = "logit"),
##     data = dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1401  -0.0698   0.0000   0.0006   1.9228
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)      22.1       26.4    0.84     0.40
## poly(X2, 5)1   -100.2      404.7   -0.25     0.80
## poly(X2, 5)2    176.9      378.7    0.47     0.64
## poly(X2, 5)3   -146.3      396.6   -0.37     0.71
## poly(X2, 5)4     48.4      143.5    0.34     0.74
## poly(X2, 5)5    -60.8      129.2   -0.47     0.64
## poly(X1, 2)1    -88.9       72.4   -1.23     0.22
## poly(X1, 2)2     82.6       58.4    1.41     0.16
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 91.658  on 71  degrees of freedom
## Residual deviance: 12.920  on 64  degrees of freedom
## AIC: 28.92
##
## Number of Fisher Scoring iterations: 14
```

```r
gr2 <- expand.grid(X1=seq(-5, 5, by=0.1), # sample points in X1
X2=seq(-5, 5, by=0.1)) # sample points in X2

predictions2 <- predict(model2,gr2,type="response")
predictions2 <- sapply(1:length(predictions2), function (x) {
     if(predictions2[[x]]>=.5) {predictions2[[x]]=1}
     else {predictions2[[x]]=0}})
     # Threshold: when the odds are 1-1 or better, assume Y=1. Otherwise 0.

gr2["predictions2"] <- predictions2
gr2.0 <- gr2[predictions2==0,]
gr2.1 <- gr2[predictions2==1,]

ggplot() +
     geom_point(data=gr2.0,aes(x=X1,y=X2),color="red") +
     geom_point(data=gr2.1,aes(x=X1,y=X2),color="blue")
```



A red ovular region covers the clump observed in part B. There is a red region along the top where data was less dense.

Part D

```
model5 <- glm(Y~poly(X2,5)+poly(X1,5),family=binomial(link='logit'),data=dat)
```

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

```
summary(model5)
```

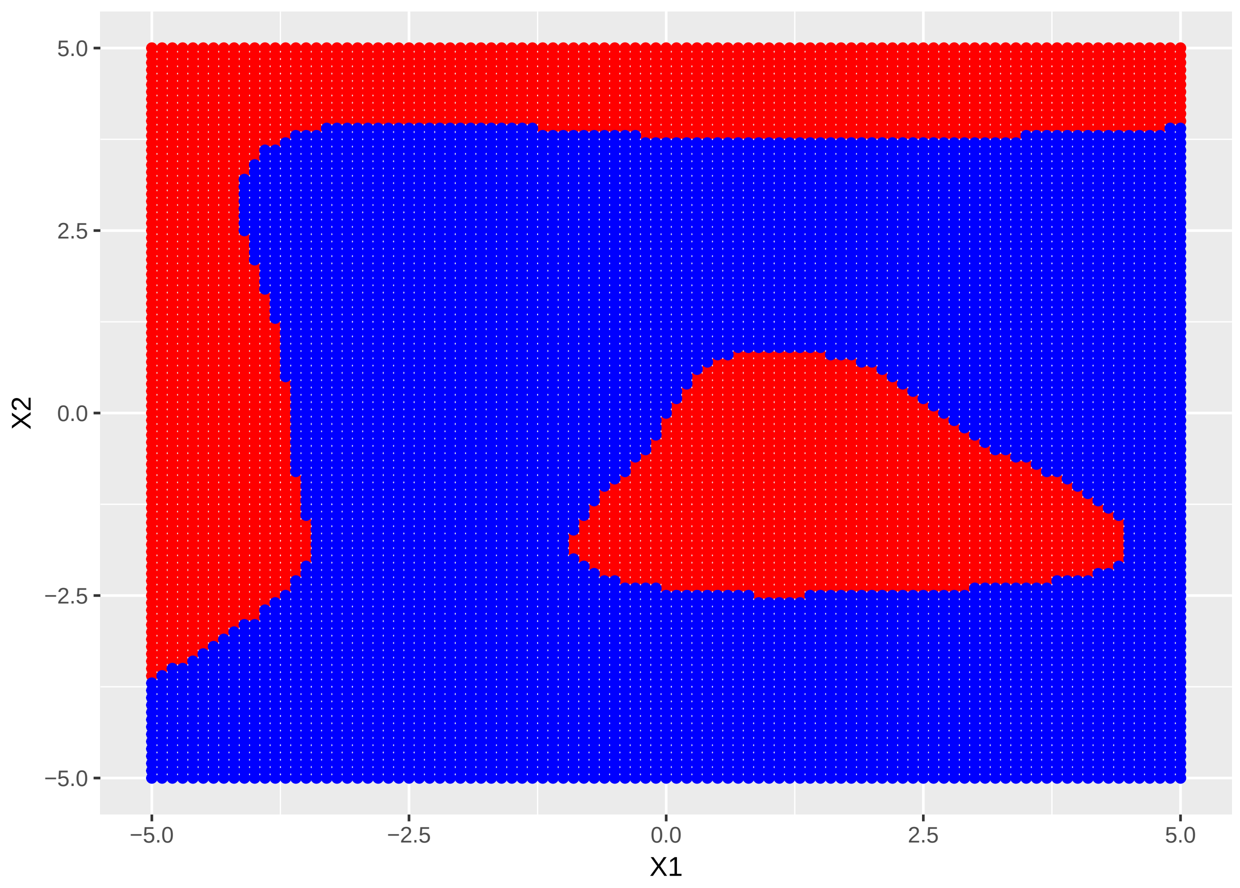
```
##
## Call:
## glm(formula = Y ~ poly(X2, 5) + poly(X1, 5), family = binomial(link = "logit"),
##     data = dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2441  -0.0209   0.0000   0.0008   1.8548
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)      25.4       41.1    0.62     0.54
## poly(X2, 5)1   -174.4      386.2   -0.45     0.65
## poly(X2, 5)2    266.1      480.1    0.55     0.58
## poly(X2, 5)3   -229.0      422.7   -0.54     0.59
## poly(X2, 5)4     90.7      219.1    0.41     0.68
## poly(X2, 5)5   -101.3      203.2   -0.50     0.62
## poly(X1, 5)1    -49.3       88.4   -0.56     0.58
## poly(X1, 5)2     25.9       36.9    0.70     0.48
## poly(X1, 5)3     36.2       61.0    0.59     0.55
## poly(X1, 5)4    -34.7       64.8   -0.54     0.59
## poly(X1, 5)5     12.7       37.7    0.34     0.74
##
## (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: 34.49
##
## Number of Fisher Scoring iterations: 14
```

```
gr5 <- expand.grid(X1=seq(-5, 5, by=0.1), # sample points in X1
X2=seq(-5, 5, by=0.1)) # sample points in X2

predictions5 <- predict(model5,gr5,type="response")
predictions5 <- sapply(1:length(predictions5), function (x) {
     if(predictions5[[x]]>=.5) {predictions5[[x]]=1}
     else{predictions5[[x]]=0}})
     # Threshold: when the odds are 1-1 or better, assume Y=1. Otherwise 0.

gr5["predictions5"] <- predictions5
gr5.0 <- gr5[predictions5==0,]
gr5.1 <- gr5[predictions5==1,]

ggplot() +
     geom_point(data=gr5.0,aes(x=X1,y=X2),color="red") +
     geom_point(data=gr5.1,aes(x=X1,y=X2),color="blue")
```



The region covering the clump is not much more accurate, but the red area along the sides has grown. Perhaps this is the result of overfitting.

Part E

The linear model cannot capture the nonseperable ovular clumping of the data. The fifth polynomial case has too much overfitting causing areas of error along the edges. The second degree polynomial model has probably the best balance of attributes, representing the clumping but with a smaller region of error.