Homework Assignment 4

Ryan Avery, Meet Gala

6/3/2018

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
library(tree)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## 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)
```

```
## Loading required package: survival
```

```
## Loading required package: lattice
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

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

```r
library(e1071)
library(ISLR)
library(imager)
```

```
## Loading required package: plyr
## ------------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------------
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
## The following object is masked from 'package:purrr':
##
##     compact
## 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:plyr':
##
##     liply
## The following object is masked from 'package:randomForest':
##
##     grow
## 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
```

Question 1 a)

$P(j \notin S)$ *where* S *is a sample of size* n *(with replacement) and* $j \in P$ *where* P *is the Population.*

This probability can be interpreted as the number of ways N spots can be filled with anything else but element j. Or :

$$P(j \notin S) = \left(\frac{n-1}{n}\right)^n$$

b) For n = 1000, this probability is:

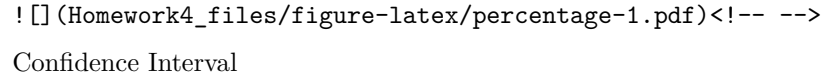
$$P(j \notin S) = \left(\frac{999}{1000}\right)^{1000} = 0.3676954.$$

c)

```
x <- c(1:1000)
length(unique(sample(x,1000,replace=TRUE)))
```

```
## [1] 637
```

Here We see that a little over 600 unique values are sampled. Which is very close to the expected 63.2%

d) ```r
```
population = c(rep(0,50),rep(1,51))
means <- vector()
for (i in 1:1000){
xboot <- sample(population, replace = TRUE)
means[i] <- mean(xboot)
}

hist(means)
```
```



Confidence Interval

```
print(quantile(means, probs = c(0.025,0.975)))
```

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

Regression to the mean is the reason you'd expect his end of season percentage to be lower than start of season, as his start of season percentage was abnormally good. As time goes on, it is mor elikely that covington will shoot at the average rate rather than the abnormally good rate he has been keeping up. Below we use bootstrap resampling to get the condifdence interval of the true percentage.
```

Question 2

```
# Transforming the data in 1 matrix of 10,000 features.
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)
}
```

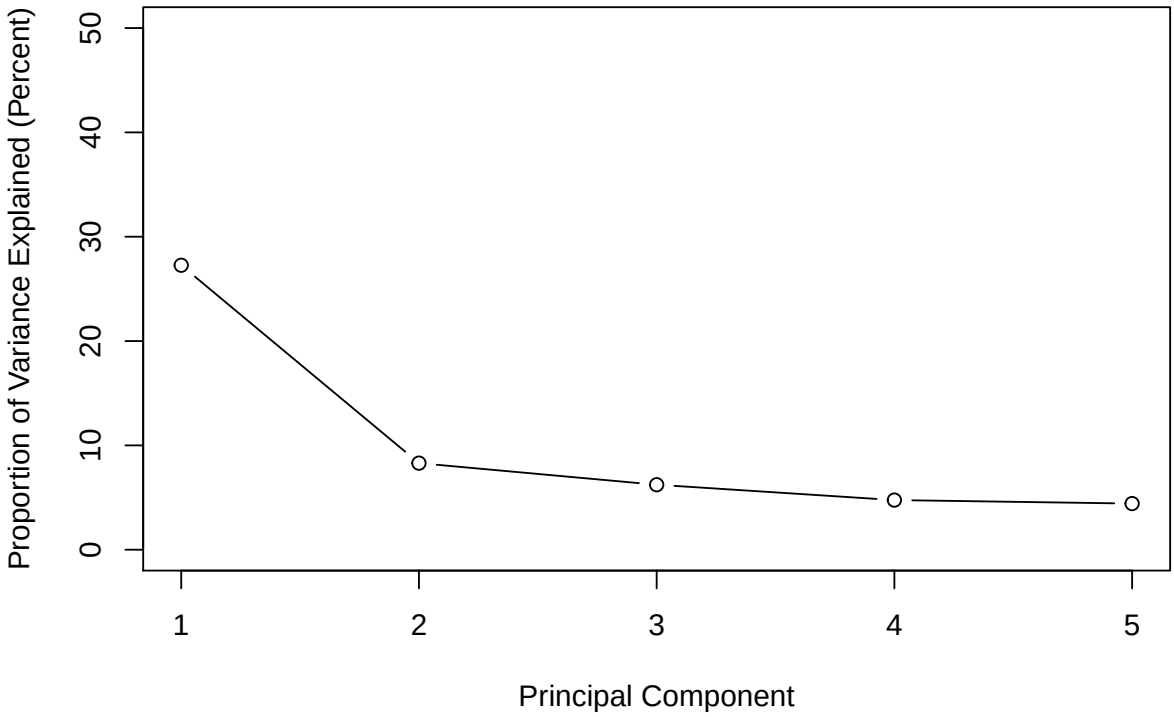
a) Below is the "Average" face

```
avg_face=colMeans(face_mat, na.rm = FALSE, dims = 1)
plot_face(avg_face)
```

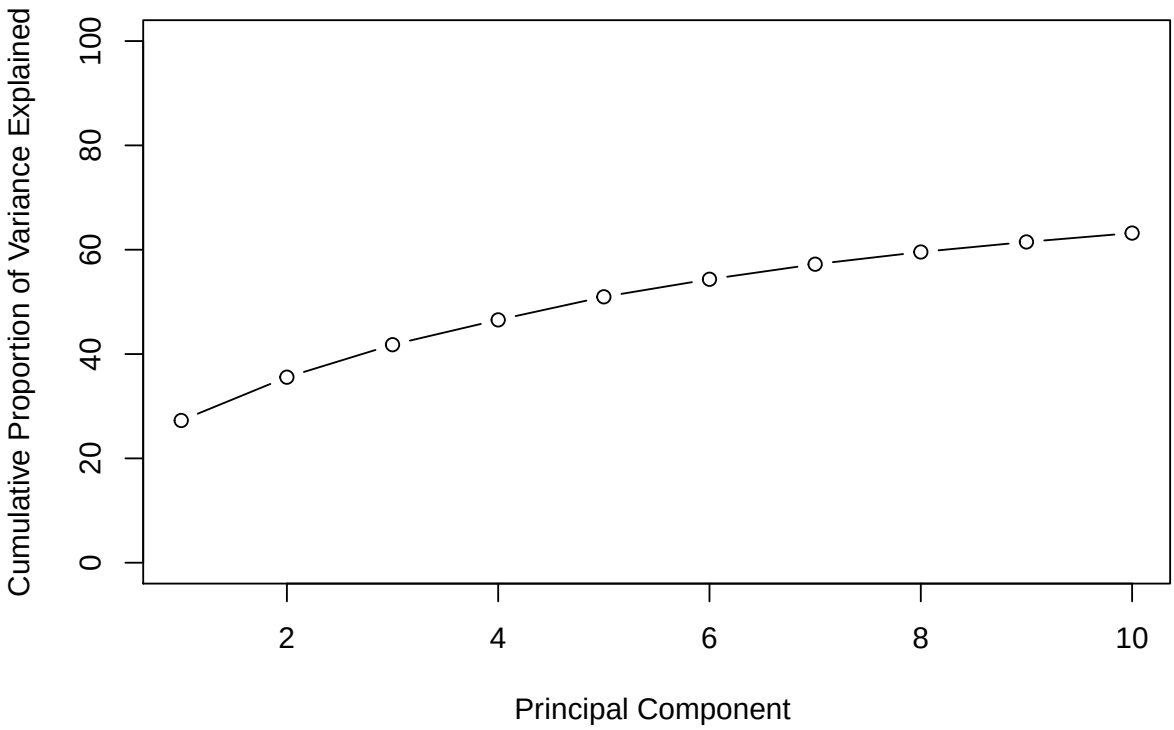
b)

```
pr.out=prcomp((face_mat),center = TRUE, scale = FALSE)
```

```
pr.var=pr.out$sdev ^2
pve=pr.var/sum(pr.var)
plot(pve[1:5]*100, xlab="Principal Component",
ylab="Proportion of Variance Explained (Percent)", ylim=c(0,50),type='b')
```




```
plot(cumsum(pve[0:10])*100, xlab="Principal Component ",
ylab=" Cumulative Proportion of Variance Explained ", ylim=c(0,100), type='b')
```




```
cumsum(pve[1:5])[5]*100
```

```
## [1] 50.97
```

We need 5 Principal components to explain 50% of the variance.

c) Below are the 16 main prinicipal component representations.

```
par(mar=rep(0.5,4), mfrow = c(4,4))
for (i in 1:16)
  plot_face(pr.out$rotation[,i])
```

d)

```
par(mar=rep(0.5,4), mfrow = c(4,4))
faces_pc1<-face_mat[order(pr.out$x[,1]),]
#par(mar=rep(0.5,4), mfrow = c(4,4))
for (i in 1:5)
  plot_face(faces_pc1[i,])
```

```
par(mar=rep(0.5,4), mfrow = c(4,4))
for (i in 996:1000)
  plot_face(faces_pc1[i,])
```

The First 5 images (low PC1 values), represent a large difference between the face (center or the image) and the background. The last 5 images, (High PC1 values) represent a smaller difference between the face and the background.

e)

```
faces_pc5<-face_mat[order(pr.out$x[,5]),]
#par(mar=rep(0.5,4), mfrow = c(4,4))
par(mar=rep(0.5,4), mfrow = c(4,4))
for (i in 996:1000)
  plot_face(faces_pc5[i,])
```

```
par(mar=rep(0.5,4), mfrow = c(4,4))
for (i in 1:5)
  plot_face(faces_pc5[i,])
```

The First 5 images (low PC5 values), represent a people with long hair. The last 5 images, (High PC5 values) represent a people with short hair.

f)

Below are umages retirved after compression

```
dim(pr.out$x)
```

```
## [1] 1000 1000
```

```
par(mfrow=c(1, 5))


k=sample(1000,1)  #taking a random image to plot using compression
  for (i in c(10,50,100,300)){
    compression<-pr.out$x[,1:i] %*% t(pr.out$rotation)[1:i,]
    plot_face(compression[k,])+avg_face}
```




Question 3

a)

```
Caravan.train <- Caravan[1:1000,]
Caravan.test <- Caravan[1001:5822,]
```

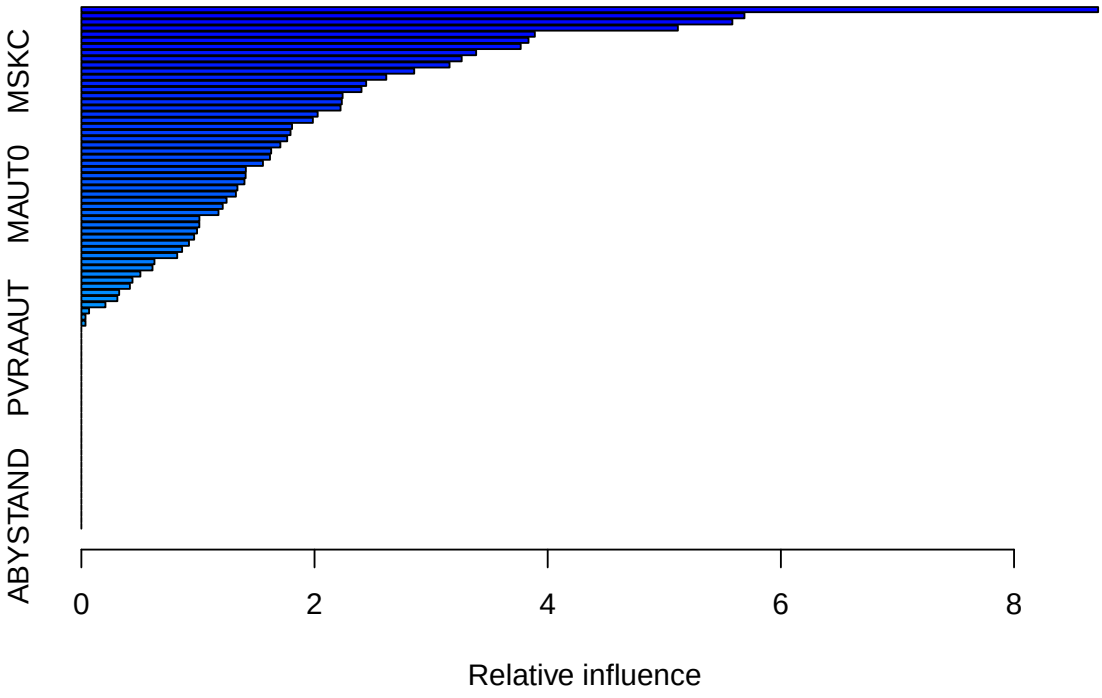
b)

```
set.seed(1)
boost.caravan  = gbm(ifelse(Purchase=="Yes",1,0)~., data=Caravan.train,
                     distribution="bernoulli", n.trees=500, interaction.depth=4, shrinkage = 0.01 )
```

```
## 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.
```

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




```
##                var rel.inf
## PPERSAUT PPERSAUT 8.72133
## MOPLHOOG MOPLHOOG 5.68747
## MGODGE     MGODGE 5.58387
## MKOOPKLA MKOOPKLA 5.11650
## PBRAND     PBRAND 3.88960
## MBERMIDD MBERMIDD 3.83544
## MGODPR     MGODPR 3.76837
## MOSTYPE   MOSTYPE 3.38615
## MBERARBG MBERARBG 3.26229
## MINK3045 MINK3045 3.15809
## MSKC         MSKC 2.85513
## MAUT2       MAUT2 2.61540
## PWAPART   PWAPART 2.44302
## MSKA         MSKA 2.40288
## MSKB1       MSKB1 2.24082
## MINK7512 MINK7512 2.23402
## MAUT1       MAUT1 2.22323
## MFWEKIND MFWEKIND 2.02686
## MRELOV     MRELOV 1.98568
## MOPLMIDD MOPLMIDD 1.80715
## MBERHOOG MBERHOOG 1.79406
## MRELGE     MRELGE 1.76570
## MGODOV     MGODOV 1.70713
## MINKM30   MINKM30 1.62814
```

```
## MFGEKIND MFGEKIND 1.61774
## MBERARBO MBERARBO 1.55761
## ABRAND     ABRAND 1.41088
## MINK4575 MINK4575 1.40899
## MHHUUR     MHHUUR 1.39907
## MZFONDS   MZFONDS 1.33874
## MAUT0       MAUT0 1.32559
## MFALLEEN MFALLEEN 1.24576
## MGEMLEEF MGEMLEEF 1.21251
## MGODRK     MGODRK 1.17620
## MHKOOP     MHKOOP 1.01250
## MINKGEM   MINKGEM 1.01187
## MRELSA     MRELSA 0.99223
## MSKB2       MSKB2 0.96751
## MGEMOMV   MGEMOMV 0.92311
## MZPART     MZPART 0.86420
## MSKD         MSKD 0.82221
## MOSHOOFD MOSHOOFD 0.62702
## APERSAUT APERSAUT 0.61015
## MOPLLAAG MOPLLAAG 0.50629
## PMOTSCO   PMOTSCO 0.43854
## PBYSTAND PBYSTAND 0.41620
## PLEVEN     PLEVEN 0.32440
## MBERZELF MBERZELF 0.30874
## MBERBOER MBERBOER 0.20640
## MINK123M MINK123M 0.06625
## MAANTHUI MAANTHUI 0.03549
## ALEVEN     ALEVEN 0.03545
## 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
## AWAPART   AWAPART 0.00000
## AWABEDR   AWABEDR 0.00000
## AWALAND   AWALAND 0.00000
## ABESAUT   ABESAUT 0.00000
## AMOTSCO   AMOTSCO 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
## ABYSTAND ABYSTAND 0.00000
```

The Variables PPERSAUT and MOPHLOOG appear to be the most important.

c)

```
rf.caravan = randomForest(Purchase ~ ., data=Caravan.train, mtry=3, ntree=500, importance=TRUE)
rf.caravan
```

```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = Caravan.train, mtry = 3,      ntree = 500, importance =
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 5.8%
## Confusion matrix:
##      No Yes class.error
## No  940   1    0.001063
## Yes  57   2    0.966102
```

The Out of bag error rate is 5.8%. 3 Variables vere subsampled at each split. 500 trees were used to develop this forest.

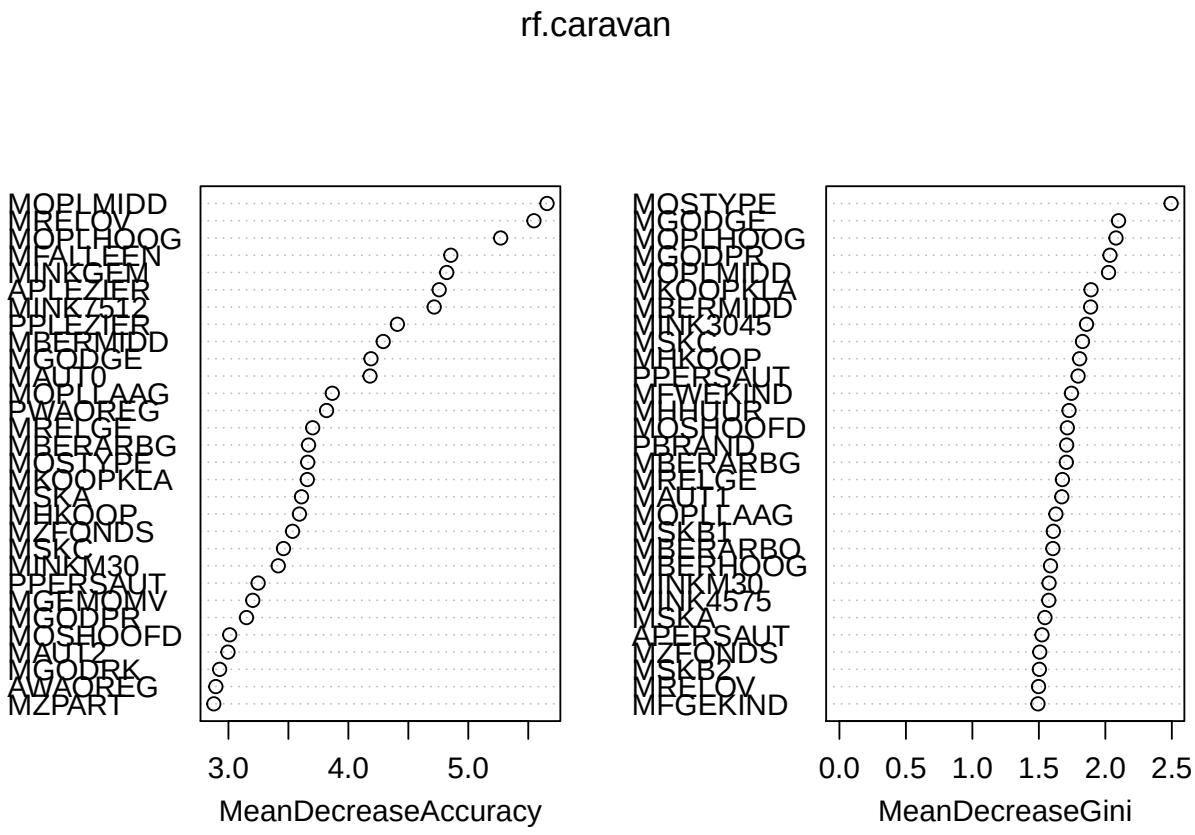
```
k<-(importance(rf.caravan))
k[order(k[,3]),]
```

```
##                 No      Yes MeanDecreaseAccuracy MeanDecreaseGini
## AMOTSCO  -3.380769 -0.86753             -3.53864        6.616e-01
## PBRAND   -4.281369  1.77373             -3.47492        1.709e+00
## PAANHANG -1.396304 -0.60075             -1.41311        2.315e-01
## PFIETS   -1.103369 -1.00100             -1.18604        1.546e-01
## ABESAUT  -1.001002  0.00000             -1.00100        1.418e-02
## ABRAND   -1.030508  0.80097             -0.78203        1.183e+00
## PINBOED  -0.653646 -1.00100             -0.72805        1.065e-01
## AAANHANG -0.426600 -1.41670             -0.72179        1.649e-01
## PWAPART  -2.277921  4.96277             -0.52035        1.472e+00
## PWABEDR  -0.280469 -1.00100             -0.49660        2.004e-01
## AFIETS   -0.494533  0.00000             -0.48067        1.326e-01
## PMOTSCO  -0.891047  1.43477             -0.30036        4.983e-01
## ALEVEN    0.113469 -1.55683             -0.28635        2.593e-01
## AWAPART  -1.844509  4.54655             -0.26511        9.754e-01
## PWALAND   0.005188  0.00000             -0.01027        6.426e-02
## PVRAAUT   0.000000  0.00000              0.00000        0.000e+00
## PWERKT    0.000000  0.00000              0.00000        1.543e-05
## PZEILPL   0.000000  0.00000              0.00000        2.145e-01
## AVRAAUT   0.000000  0.00000              0.00000        0.000e+00
## AWERKT    0.000000  0.00000              0.00000        4.324e-04
## AZEILPL   0.000000  0.00000              0.00000        3.128e-01
## AWALAND   0.166767 -0.09843              0.19014        9.313e-02
```

```
## MSKB1     0.185031 -0.11382            0.26095       1.609e+00
## MBERHOOG -0.572271  2.48708            0.28335       1.588e+00
## MBERBOER  0.252393  0.16504            0.30048       5.700e-01
## AINBOED   0.316972  0.00000            0.31718       1.063e-01
## PLEVEN    0.376914  0.15618            0.35264       4.460e-01
## ABROM     0.551228  1.35009            0.72105       2.270e-01
## APERSONG  1.001002  0.00000            1.00100       4.887e-03
## PBESAUT   1.001002  0.00000            1.00100       6.110e-03
## PPERSONG  1.001002  0.00000            1.00100       8.522e-03
## MAANTHUI  1.425485 -1.14928            1.00201       4.525e-01
## AWABEDR   1.473069 -1.38943            1.09230       9.716e-02
## MSKB2     1.423198 -0.70961            1.14322       1.504e+00
## MRELSA    0.784616  1.52281            1.22641       1.244e+00
## MINK123M  1.216427 -0.19105            1.23820       3.429e-01
## PBYSTAND  1.122694  0.60593            1.28862       4.953e-01
## MHHUUR    0.484250  3.05423            1.33115       1.727e+00
## PTRACTOR  2.166110 -1.71489            1.34413       2.836e-01
## ABYSTAND  1.349217  0.20556            1.38444       4.757e-01
## PGEZONG   1.519819  0.00000            1.52044       4.611e-01
## MINK4575  0.367087  3.85755            1.56990       1.575e+00
## MGODOV    1.429363  0.54329            1.59933       1.379e+00
## AGEZONG   1.401789  1.23170            1.67459       3.290e-01
## MFGEKIND  2.409204 -2.72747            1.69714       1.493e+00
## APERSAUT  1.105249  2.36072            1.77087       1.523e+00
## PBROM     2.097770 -1.20084            1.84920       2.362e-01
## MGEMLEEF  1.838018  0.91998            2.00158       1.016e+00
## MSKD      1.636555  1.52880            2.06404       9.435e-01
## MINK3045  1.794087  1.39364            2.13005       1.858e+00
## ATRACTOR  2.635226  0.00000            2.43448       8.857e-02
## MFWEKIND  1.795421  2.64501            2.45075       1.745e+00
## MBERZELF  2.390327  0.91146            2.64268       7.275e-01
## MBERARBO  1.987812  2.50740            2.73927       1.605e+00
## MAUT1     2.086549  2.46753            2.74008       1.672e+00
## MZPART    3.056478 -0.65880            2.87826       1.440e+00
## AWAOREG   3.159334  0.96250            2.89498       5.480e-01
## MGODRK    2.624204  1.14937            2.92652       9.790e-01
## MAUT2     2.920167  0.49305            2.99607       1.258e+00
## MOSHOOFD  1.787184  4.21760            3.01080       1.714e+00
## MGODPR    2.370717  2.38789            3.15056       2.035e+00
## MGEMOMV   2.841383  1.08198            3.20287       8.748e-01
## PPERSAUT  2.139119  4.29206            3.24759       1.795e+00
## MINKM30   2.884541  2.31370            3.41516       1.577e+00
## MSKC      2.819874  2.50185            3.45910       1.828e+00
## MZFONDS   3.505401  0.04818            3.53463       1.507e+00
## MHKOOP    2.377097  3.94393            3.59202       1.806e+00
## MSKA      3.171041  1.66094            3.60941       1.546e+00
## MKOOPKLA  2.452009  4.31217            3.65754       1.892e+00
## MOSTYPE   2.614588  3.60324            3.66097       2.494e+00
## MBERARBG  3.578998  0.39262            3.66785       1.706e+00
## MRELGE    3.758470 -0.66742            3.70240       1.677e+00
## PWAOREG   3.439446  2.83829            3.81810       5.332e-01
## MOPLLAAG  4.053901 -0.83103            3.86607       1.628e+00
## MAUT0     3.882622  1.53118            4.18040       1.300e+00
## MGODGE    2.819479  4.49343            4.18867       2.099e+00
```

```
## MBERMIDD  3.316404  3.22572              4.29058        1.889e+00
## PPLEZIER  2.844576  4.28206              4.40875        1.375e+00
## MINK7512  4.268596  1.87393              4.71560        1.331e+00
## APLEZIER  2.330737  5.47171              4.75738        1.029e+00
## MINKGEM   4.403930  1.84144              4.82016        1.352e+00
## MFALLEEN  4.687745  0.15458              4.85422        1.374e+00
## MOPLHOOG  3.423589  5.29497              5.27006        2.080e+00
## MRELOV    5.336745  0.74255              5.54742        1.498e+00
## MOPLMIDD  5.927381 -1.02803              5.65644        2.024e+00
```

```
varImpPlot(rf.caravan)
```



Above is the list of the varibale imporance (sorted by Decrease in Accuracy) in the Random forset model and it is very different from our boosting model. MOPLHOOG is the second most important for boosting and 3rd most important for Random Forests. Apart from that there is not much similarity in variable importance between the two models.

d)

```
yhat.boost = as.factor(ifelse(predict(boost.caravan, newdata = Caravan.test, n.trees=500, type = "respo
boost.err = table(pred = yhat.boost, truth = Caravan.test$Purchase)

boost.err
```

```
##       truth
## pred     No  Yes
##   No   4387  262
```

```
##   Yes  146   27
```

```
yhat.rf = predict (rf.caravan, newdata = Caravan.test)
rf.err = table(pred = yhat.rf, truth = Caravan.test$Purchase)
rf.err
```

```
##      truth
## pred    No  Yes
##   No  4528  288
##   Yes    5    1
```

```
rf.err[2,2]/sum(rf.err[2,])
```

```
## [1] 0.1667
```

16.67% of the people predicted to make a purchase actually do.

Question 4

```
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.
```

```
cannabis_levels <- c("No","Yes")
drug_use <- drug_use %>%
  mutate(recent_cannabis_use = factor(ifelse(Cannabis>='CL3',"Yes","No")))
drug_use_subset <- drug_use %>% select(Age:SS, recent_cannabis_use)
set.seed(1)
train.indices = sample(1:nrow(drug_use_subset), 1500)
drug_use_train=drug_use_subset[train.indices,]
drug_use_test=drug_use_subset[-train.indices,]
```

A) Confusion matrix for predictions against test data with radial kernel cost = 1

```
svmfit=svm(recent_cannabis_use ~ ., data=drug_use_train, kernel="radial", cost=1,scale=TRUE)
cannabis_pred=predict(svmfit,drug_use_test)
```

```
summary(svmfit)
```

```
##
## Call:
## svm(formula = recent_cannabis_use ~ ., data = drug_use_train,
##     kernel = "radial", cost = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.08333
##
## Number of Support Vectors:  744
##
##  ( 360 384 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
table(predict=cannabis_pred, truth=drug_use_test$recent_cannabis_use)
```

```
##        truth
## predict  No Yes
##     No  145  30
##     Yes  43 167
```

B) The optimal cost is .1. Optimal cost, training and test error and confusion matrix for the trianing and test data is below:

```
tune.out=tune(svm,recent_cannabis_use~.,data=drug_use_train,kernel="radial",
ranges=list(cost=c(.001,.01,.1,1,10,100)), scale=TRUE)
bestmod=tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = recent_cannabis_use ~ ., data = drug_use_train,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10, 100)), kernel = "radial",
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.1
##       gamma:  0.08333
##
## Number of Support Vectors:  882
##
##  ( 440 442 )
##
```

```
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
print("training error")
```

```
## [1] "training error"
```

```
cannabis_pred_train=predict(bestmod,drug_use_train)
train_table = table(predict=cannabis_pred_train, truth=drug_use_train$recent_cannabis_use)
print(train_table)
```

```
##        truth
## predict  No Yes
##     No  579 135
##     Yes 119 667
```

```
train.err = 1 - sum(diag(train_table))/sum(train_table)
print(train.err)
```

```
## [1] 0.1693
```

```
print("test error")
```

```
## [1] "test error"
```

```
cannabis_pred_test=predict(bestmod,drug_use_test)
test_table = table(predict=cannabis_pred_test, truth=drug_use_test$recent_cannabis_use)
print(test_table)
```

```
##        truth
## predict  No Yes
##     No  148  35
##     Yes  40 162
```

```
test.err = 1 - sum(diag(test_table))/sum(test_table)
print(test.err)
```

```
## [1] 0.1948
```

C)

```
class_counts = 0
for (i in 1:200){
ind = sample(nrow(drug_use_train),size=nrow(drug_use_train), replace = TRUE)
train_boot = drug_use_train[ind,]
svmfit=svm(recent_cannabis_use ~ ., data=train_boot, kernel="radial", cost=1,scale=TRUE)
pred = predict(svmfit, drug_use_test)
pred = ifelse(pred=="Yes", 1, 0)
class_counts = class_counts + pred
}
```

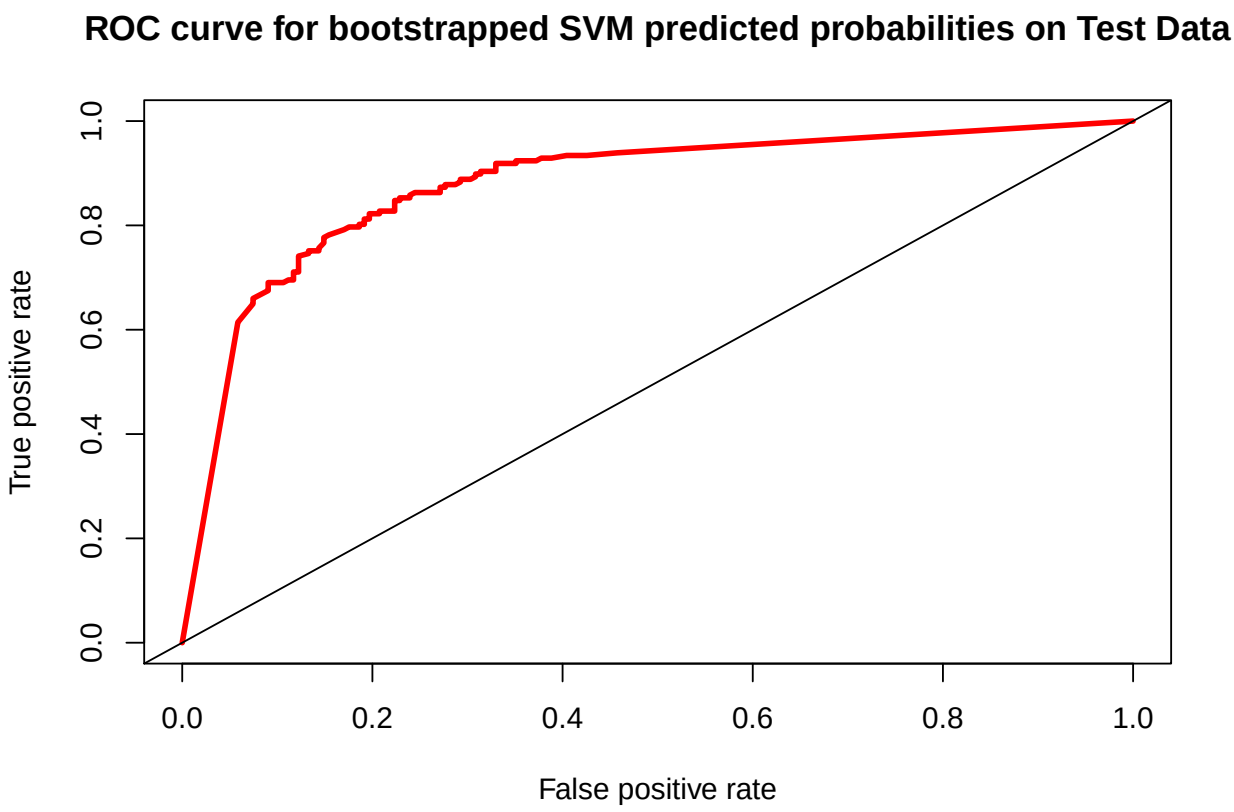
Below are the bootstrapped svm class probabilities for each observation.

```
predicted_svm_probs = class_counts/200
predicted_svm_probs
```

```
##   [1] 0.060 0.190 0.000 0.545 0.610 0.305 0.000 0.000 0.000 0.985 0.000
```

```
##  [12] 0.345 0.025 1.000 0.000 0.965 0.000 0.000 0.000 0.050 0.000 0.010
##  [23] 0.045 0.150 0.000 0.000 0.000 0.000 0.000 0.040 0.000 0.000 0.000
##  [34] 0.980 0.000 0.000 0.135 0.000 0.000 0.690 0.255 0.000 0.000 0.300
##  [45] 0.770 0.140 0.000 0.000 0.000 0.000 0.000 0.405 0.005 0.085 0.000
##  [56] 0.000 0.000 0.025 0.000 0.770 0.000 0.000 0.000 0.000 0.000 0.045
##  [67] 0.190 0.560 0.000 0.000 0.000 0.070 0.735 1.000 1.000 1.000 0.815
##  [78] 0.000 0.000 0.000 0.000 0.970 0.000 0.000 0.000 0.000 1.000 0.760
##  [89] 0.000 0.000 0.195 0.315 0.000 0.000 0.010 0.275 0.280 0.570 0.000
## [100] 0.925 0.745 0.000 1.000 0.410 1.000 1.000 0.000 1.000 0.000 0.000
## [111] 1.000 0.005 0.560 0.000 0.000 0.000 0.320 0.960 0.995 0.000 0.975
## [122] 0.000 0.000 0.000 0.010 0.405 1.000 1.000 1.000 0.145 0.000 0.115
## [133] 0.000 0.000 0.185 1.000 0.000 0.935 0.000 0.000 0.000 1.000 0.760
## [144] 0.580 0.780 0.295 0.000 0.990 1.000 0.050 0.000 0.000 0.865 1.000
## [155] 0.960 1.000 1.000 0.000 0.000 0.005 1.000 1.000 1.000 0.780 0.630
## [166] 1.000 0.980 0.875 1.000 0.775 1.000 1.000 0.540 1.000 1.000 1.000
## [177] 0.995 1.000 0.780 0.225 1.000 1.000 1.000 1.000 1.000 1.000 1.000
## [188] 1.000 1.000 1.000 1.000 0.215 0.875 1.000 0.510 1.000 1.000 0.645
## [199] 1.000 1.000 1.000 1.000 1.000 1.000 0.025 0.995 1.000 0.995 1.000
## [210] 0.925 1.000 1.000 1.000 1.000 0.585 1.000 0.995 0.065 1.000 0.570
## [221] 0.950 0.670 1.000 0.060 1.000 0.000 0.000 1.000 1.000 0.655 1.000
## [232] 1.000 0.745 1.000 1.000 1.000 1.000 0.000 0.985 0.995 1.000 0.760
## [243] 1.000 1.000 1.000 1.000 0.985 1.000 1.000 1.000 1.000 1.000 0.980
## [254] 1.000 1.000 0.005 1.000 1.000 1.000 0.195 0.000 0.975 0.005 0.000
## [265] 0.000 1.000 1.000 1.000 1.000 0.760 1.000 0.985 1.000 0.945 0.500
## [276] 1.000 1.000 1.000 1.000 0.000 1.000 1.000 0.020 0.000 1.000 0.000
## [287] 0.000 0.095 0.000 0.225 1.000 1.000 1.000 0.000 0.000 0.875 0.000
## [298] 0.000 0.000 0.090 1.000 0.995 0.985 0.805 0.795 0.930 0.985 1.000
## [309] 1.000 0.990 1.000 1.000 0.995 1.000 0.625 1.000 1.000 0.975 0.000
## [320] 0.000 0.030 0.000 0.000 0.000 0.000 0.240 0.025 0.000 0.000 0.000
## [331] 0.425 0.000 0.005 0.705 1.000 0.000 0.940 0.000 1.000 0.000 0.005
## [342] 0.995 0.000 1.000 1.000 1.000 1.000 0.390 0.000 0.125 0.000 0.070
## [353] 0.000 0.000 0.605 0.000 1.000 0.110 1.000 1.000 0.000 0.460 0.760
## [364] 1.000 1.000 1.000 1.000 1.000 0.995 0.940 0.000 1.000 1.000 0.775
## [375] 1.000 0.970 1.000 1.000 1.000 0.695 1.000 1.000 1.000 1.000 1.000
```

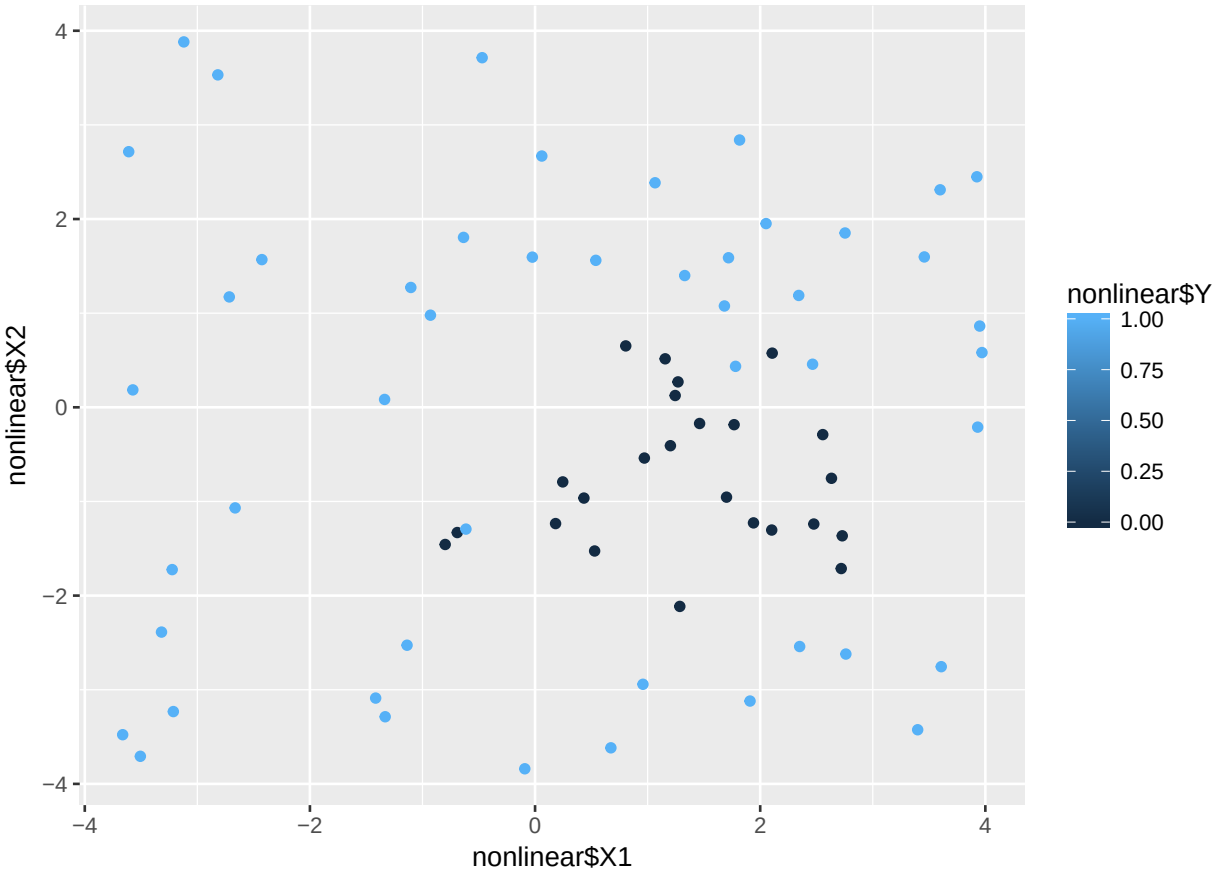
```
pred = prediction(predicted_svm_probs, drug_use_test$recent_cannabis_use)
perf = performance(pred, measure="tpr", x.measure="fpr")
plot(perf, col=2, lwd=3, main="ROC curve for bootstrapped SVM predicted probabilities on Test Data")
abline(0,1)
```



Question 5

a)

```
nonlinear = read.csv("nonlinear.csv")
qplot(nonlinear$X1,nonlinear$X2,color=nonlinear$Y)
```

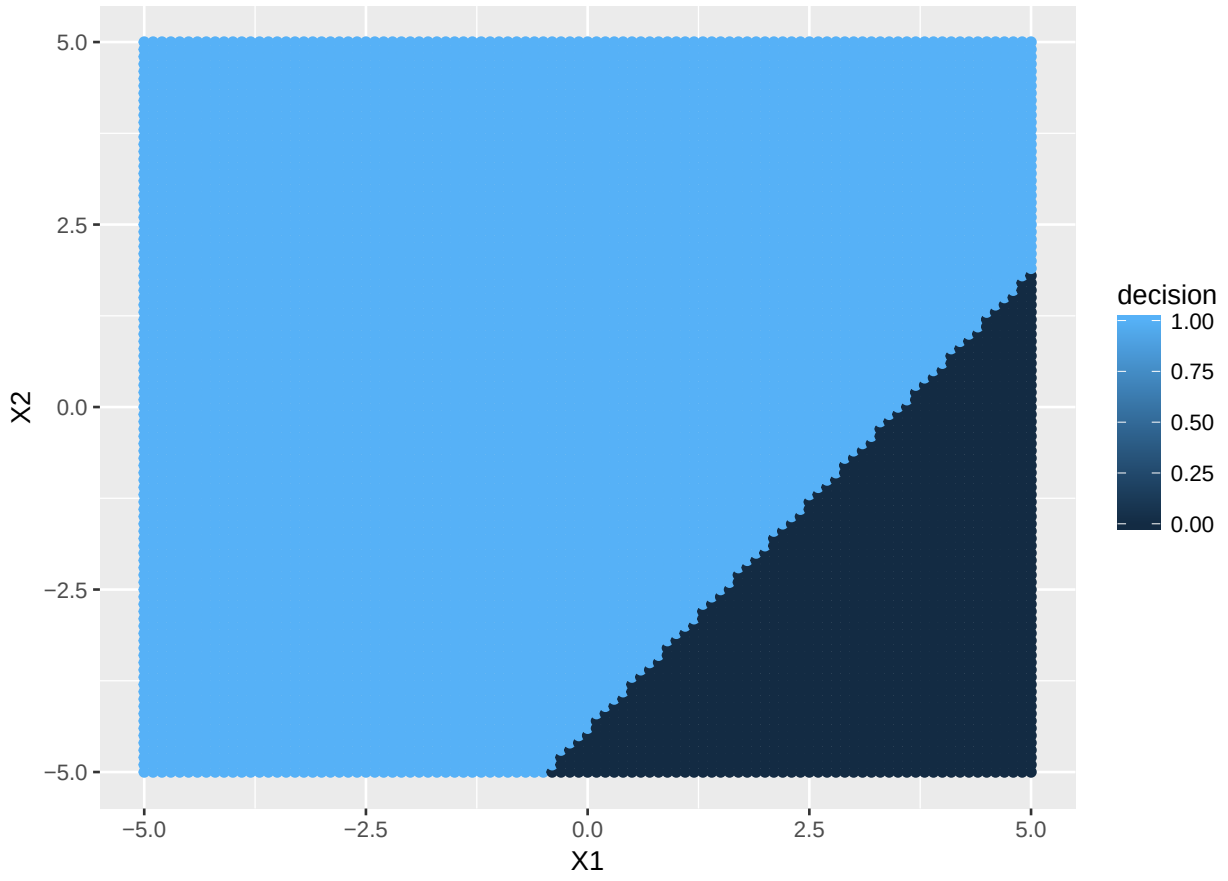


b)

```
# 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
```

Below is the linear logistic regression decision boundry

```
logistic.fit = glm(Y ~ X1 + X2 ,data=nonlinear, family=binomial)
decision<-ifelse(predict(logistic.fit, gr, type="response")>0.5,1,0)
qplot(x=gr[,1],y=gr[,2],color=decision,xlab = "X1",ylab = "X2")
```



c)

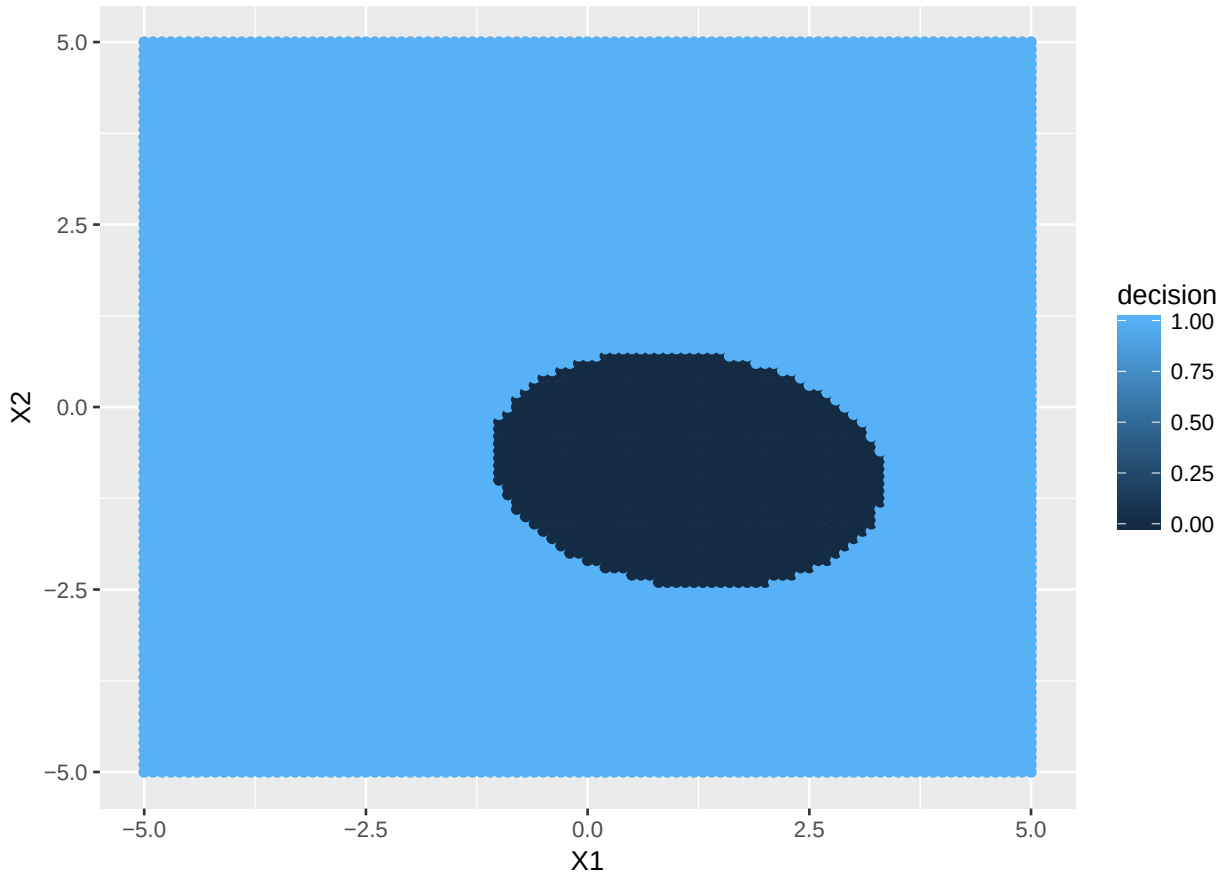
```
X<-as.matrix(nonlinear[,2:3],nrow=36,ncol=2)
nonlinear_poly=as.data.frame(poly(X,2,raw=TRUE))
nonlinear_poly$Y=nonlinear$Y
logistic.poly.fit = glm(Y ~. ,data=nonlinear_poly, family=binomial)
```

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

```
decision<-ifelse(predict(logistic.poly.fit, as.data.frame(poly(as.matrix(gr),2,raw=TRUE)), type="respons
```

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

```
qplot(x=gr[,1],y=gr[,2],color=decision,xlab = "X1",ylab = "X2")
```




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

```
##
## Call:
## glm(formula = Y ~ ., family = binomial, data = nonlinear_poly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3908  -0.0827   0.0000   0.0093   1.9007
##
## Coefficients: (4 not defined because of singularities)
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -3.681      2.011   -1.83    0.067 .
## `1.0.0`       -2.812      1.632   -1.72    0.085 .
## `2.0.0`        1.427      0.718    1.99    0.047 *
## `0.1.0`        3.935      1.911    2.06    0.039 *
## `1.1.0`        0.501      0.737    0.68    0.496
## `0.2.0`        2.678      1.105    2.42    0.015 *
## `0.0.1`           NA         NA      NA       NA
## `1.0.1`           NA         NA      NA       NA
## `0.1.1`           NA         NA      NA       NA
## `0.0.2`           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: 91.658  on 71  degrees of freedom
## Residual deviance: 13.852  on 66  degrees of freedom
## AIC: 25.85
##
## Number of Fisher Scoring iterations: 10
```

Here we see an elliptical decision boundry. Which does not seem very un-resonable. However it is a bit variant on the data. A new point outside the ellipse would change the size of the decision boundry.

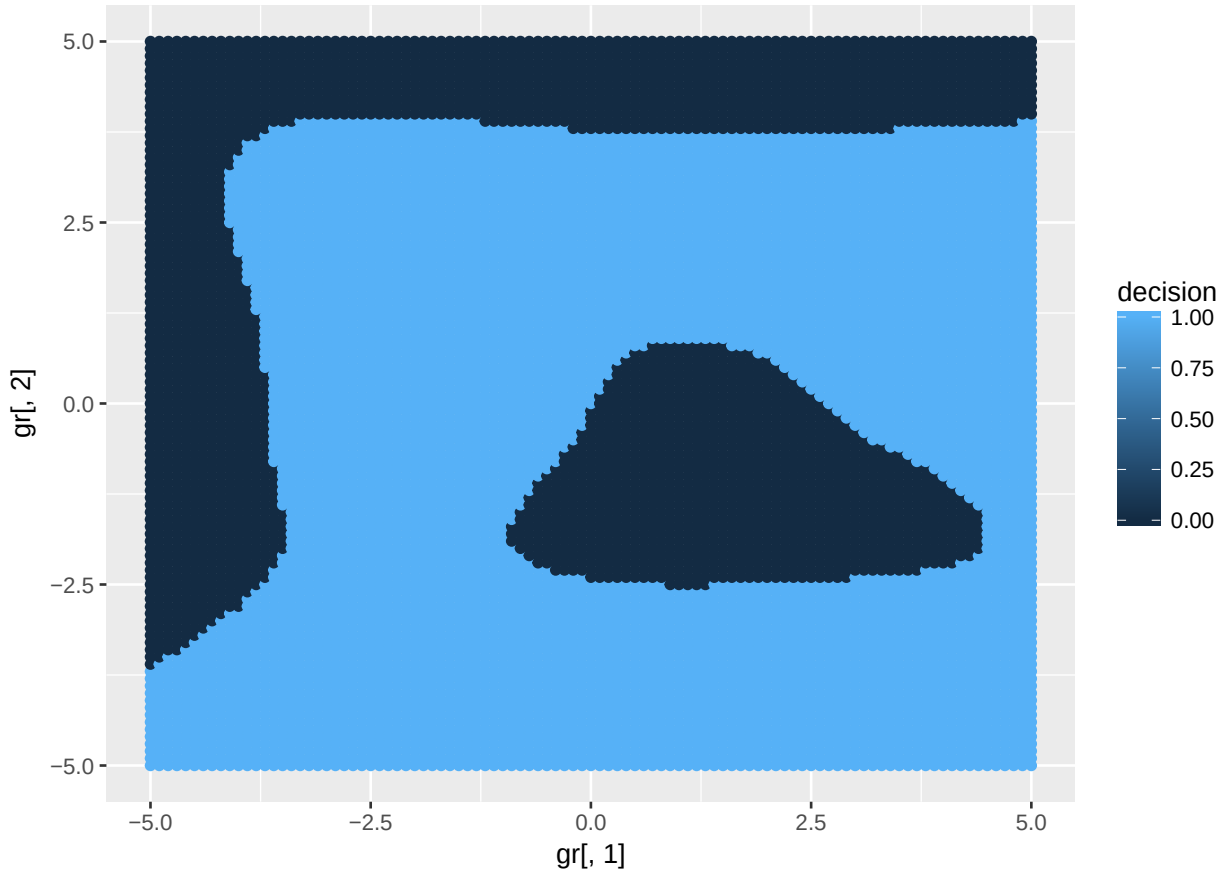
We see three parameters that are outside the 5% confidence level which means our model is probably biased.

d)

```
nonlinear_poly=as.data.frame(poly(X,5,raw=TRUE))[c(1:6,11,15,18,20)]
nonlinear_poly$Y=nonlinear$Y
logistic.poly.fit = glm(Y ~. ,data=nonlinear_poly, family=binomial)
```

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

```
decision<-ifelse(predict(logistic.poly.fit, as.data.frame(poly(as.matrix(gr),5,raw=TRUE))[c(1:6,11,15,18
qplot(gr[,1],gr[,2],color=decision)
```




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

```
##
## Call:
## glm(formula = Y ~ ., family = binomial, data = nonlinear_poly)
```

```
##
## 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)   0.2233     7.7978    0.03     0.98
## `1.0.0`      -7.7007    11.3922   -0.68     0.50
## `2.0.0`       4.0598     6.9647    0.58     0.56
## `3.0.0`       0.0198     1.3872    0.01     0.99
## `4.0.0`      -0.2911     0.5795   -0.50     0.62
## `5.0.0`       0.0436     0.1299    0.34     0.74
## `0.1.0`       2.7517     3.3885    0.81     0.42
## `0.2.0`      -0.8467     7.2604   -0.12     0.91
## `0.3.0`       2.7750     6.2580    0.44     0.66
## `0.4.0`       0.6199     1.5100    0.41     0.68
## `0.5.0`      -0.3566     0.7153   -0.50     0.62
##
## (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
```

Here we see an interesting and somewhat doubtful result. The Dark blue spots that are not in the cetral blob (where the existing ellipse was) would be classified as a "NO", but we know that there is no point there. Here any new data would change that boundry by a lot and hence we night be have high variability in our model. The coefficeients all are signifincant and lie within our confidence bounds, so we can trust this model to not be very biased.

e) The coefficients for the X1^2, X2^2, and X1*X2 terms in the polynomial logistic model are all significant with a p value of less than .05. None of the coefficeints are significant for the 5th order polynomial model due to overfitting. The variance is extremely high for the 5th order polynomial model. On the converse, the bias is very high for the linear model, as the model is too simple and does not identify the circular clustering of the data. The 2nd order polynomial model strikes the best balance between complexity and simplicity, and thus the best balance between bias and variance.

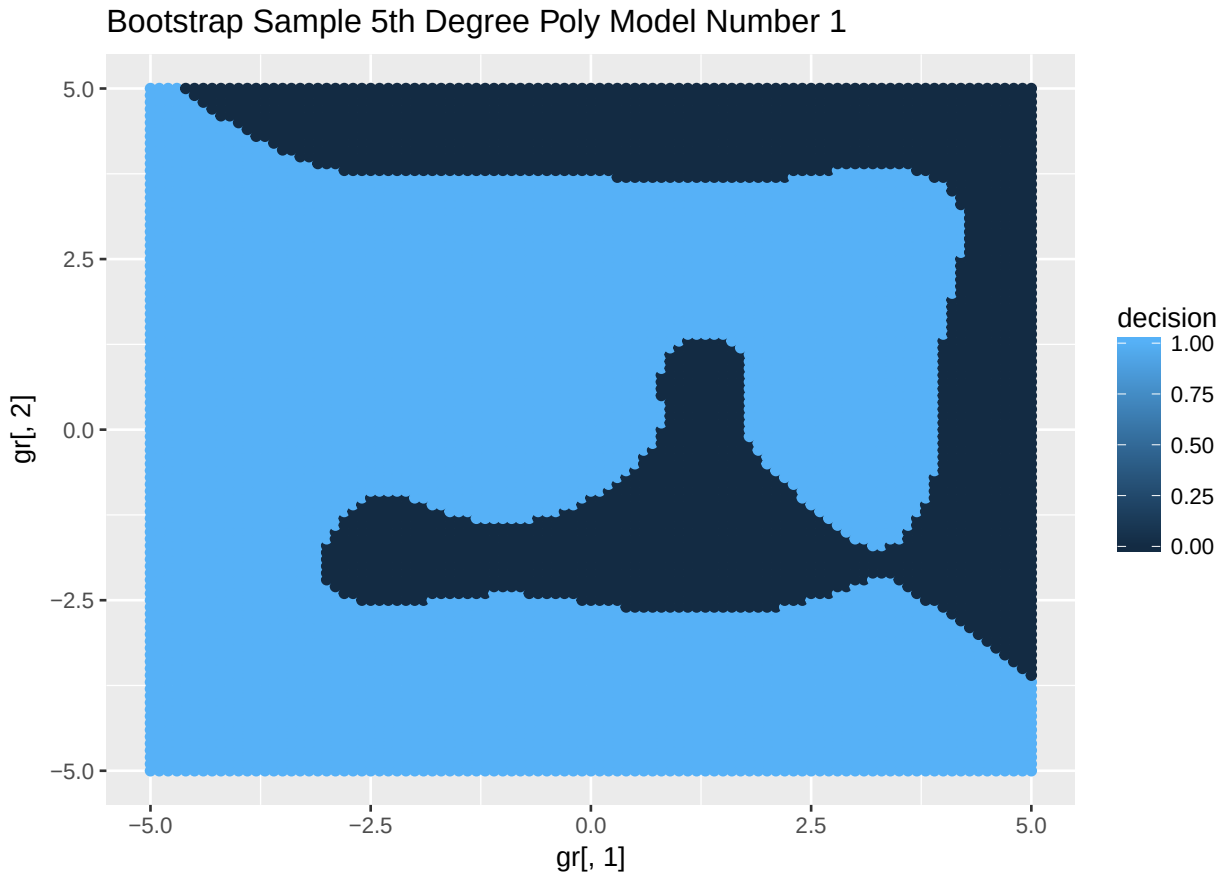
f)

```
for (i in 1:3){

ind = sample(nrow(nonlinear_poly),size=nrow(nonlinear_poly), replace = TRUE)
train_poly_boot = nonlinear_poly[ind,]

X<-as.matrix(nonlinear[2:4],nrow=36,ncol=2)
nonlinear_copy = as.data.frame(X)
ind_lin = sample(nrow(nonlinear_copy),size=nrow(nonlinear_copy), replace = TRUE)
train_lin_boot = nonlinear_copy[ind_lin,]

logistic.poly.fit = glm(Y ~. ,data=train_poly_boot, family=binomial)
logistic.fit = glm(Y ~ X1 + X2 ,data=train_lin_boot, family=binomial)

decision<-ifelse(predict(logistic.poly.fit, as.data.frame(poly(as.matrix(gr),5,raw=TRUE))[c(1:6,11,15,18
```

```
print(qplot(gr[,1],gr[,2],color=decision, main=paste(c("Bootstrap Sample 5th Degree Poly Model Number",

decision<-ifelse(predict(logistic.fit, gr, type="response")>0.5,1,0)
print(qplot(gr[,1],gr[,2],color=decision, main=paste(c("Bootstrap Sample Linear Model Number", i), colla
}
```

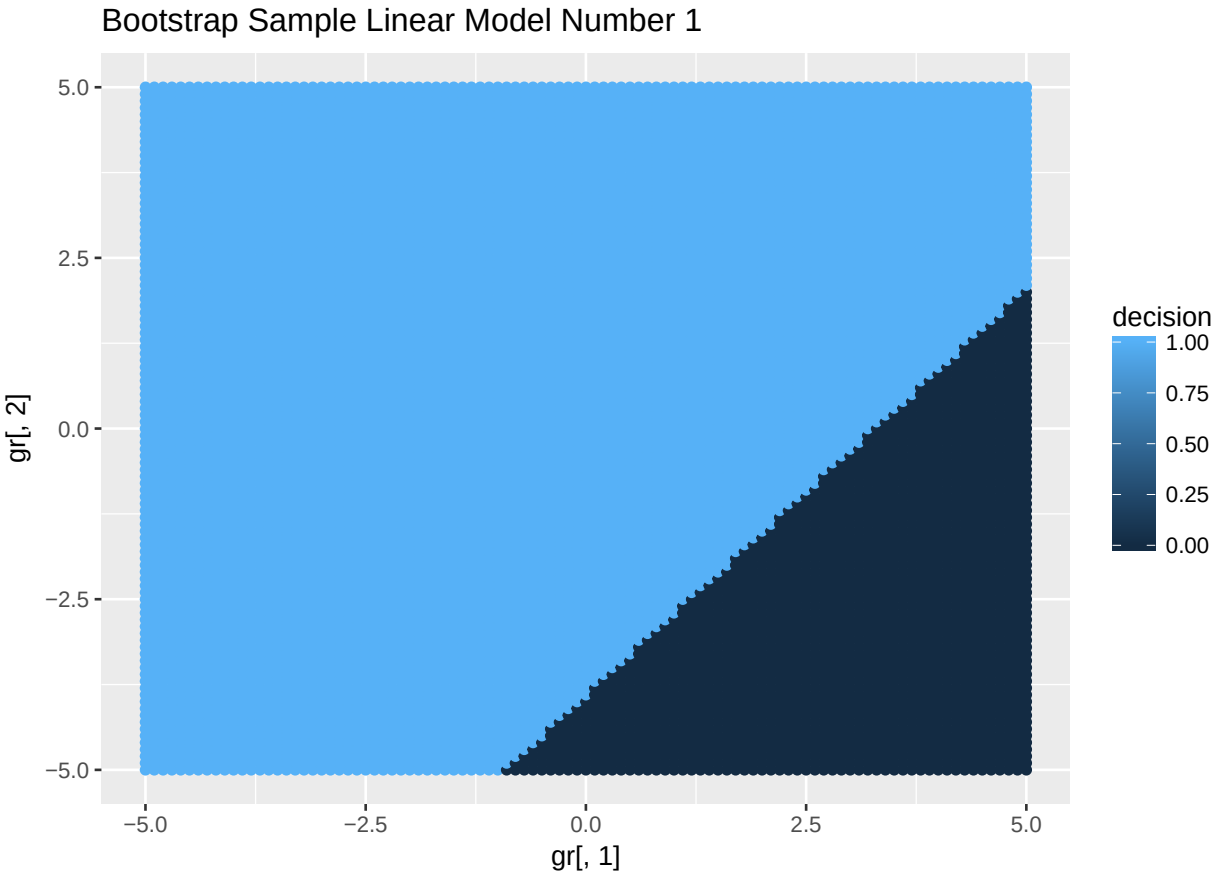
```
## Warning: glm.fit: algorithm did not converge
```

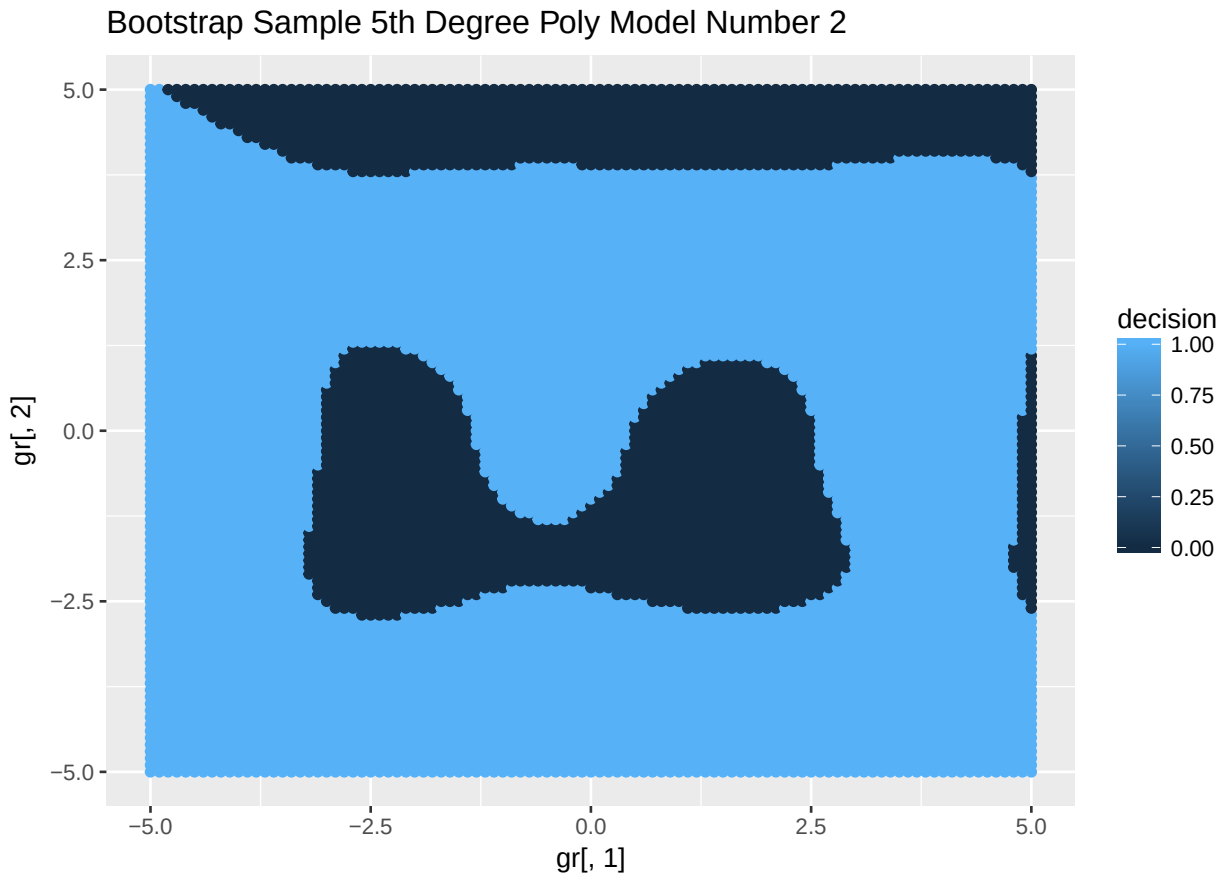
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



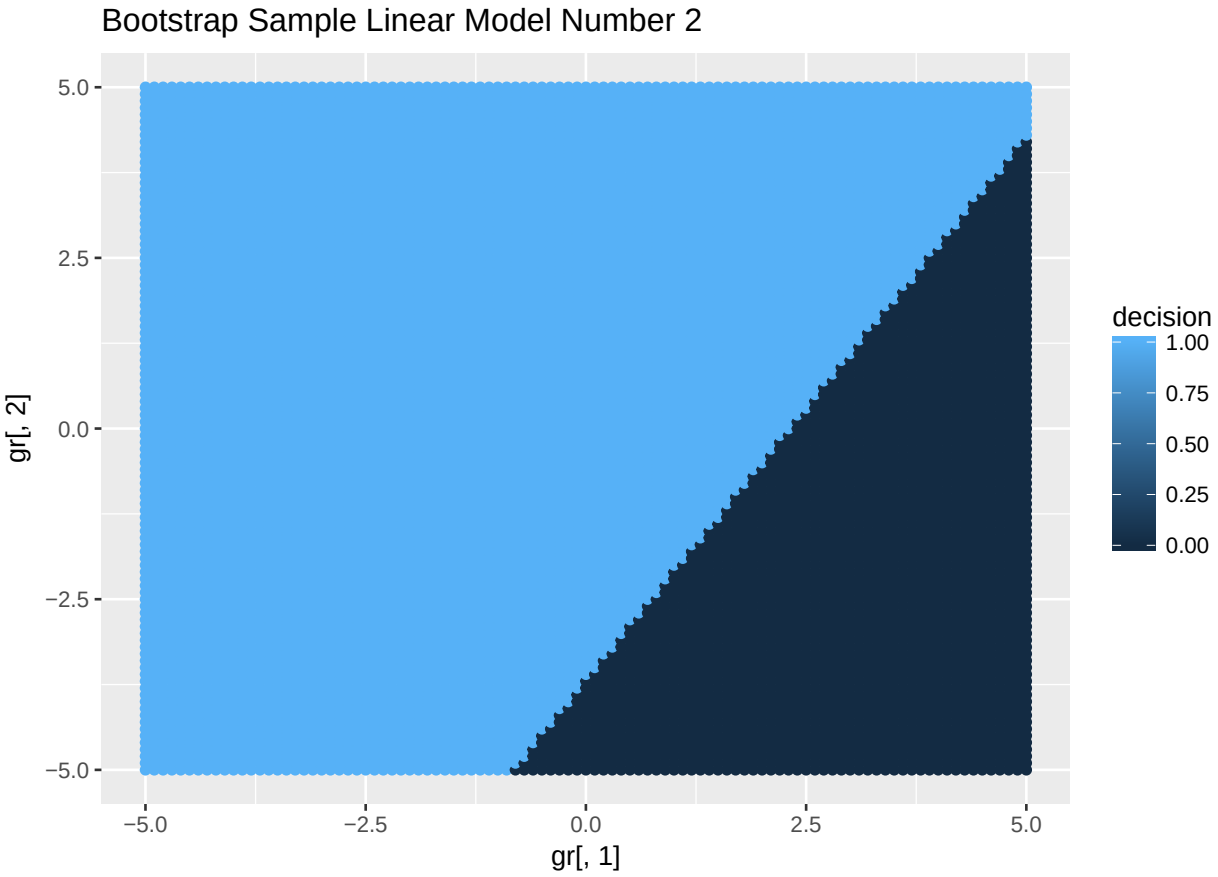

```
## Warning: glm.fit: algorithm did not converge
```

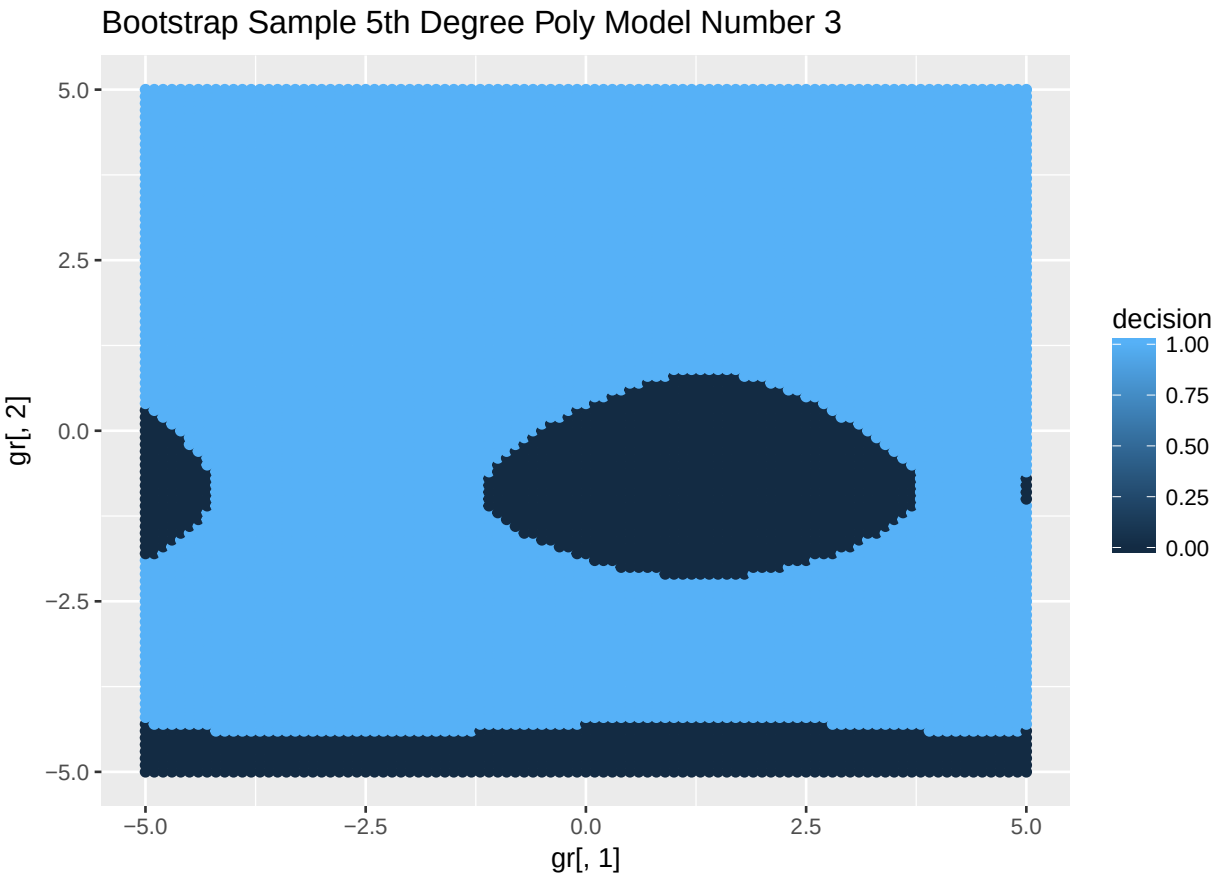
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

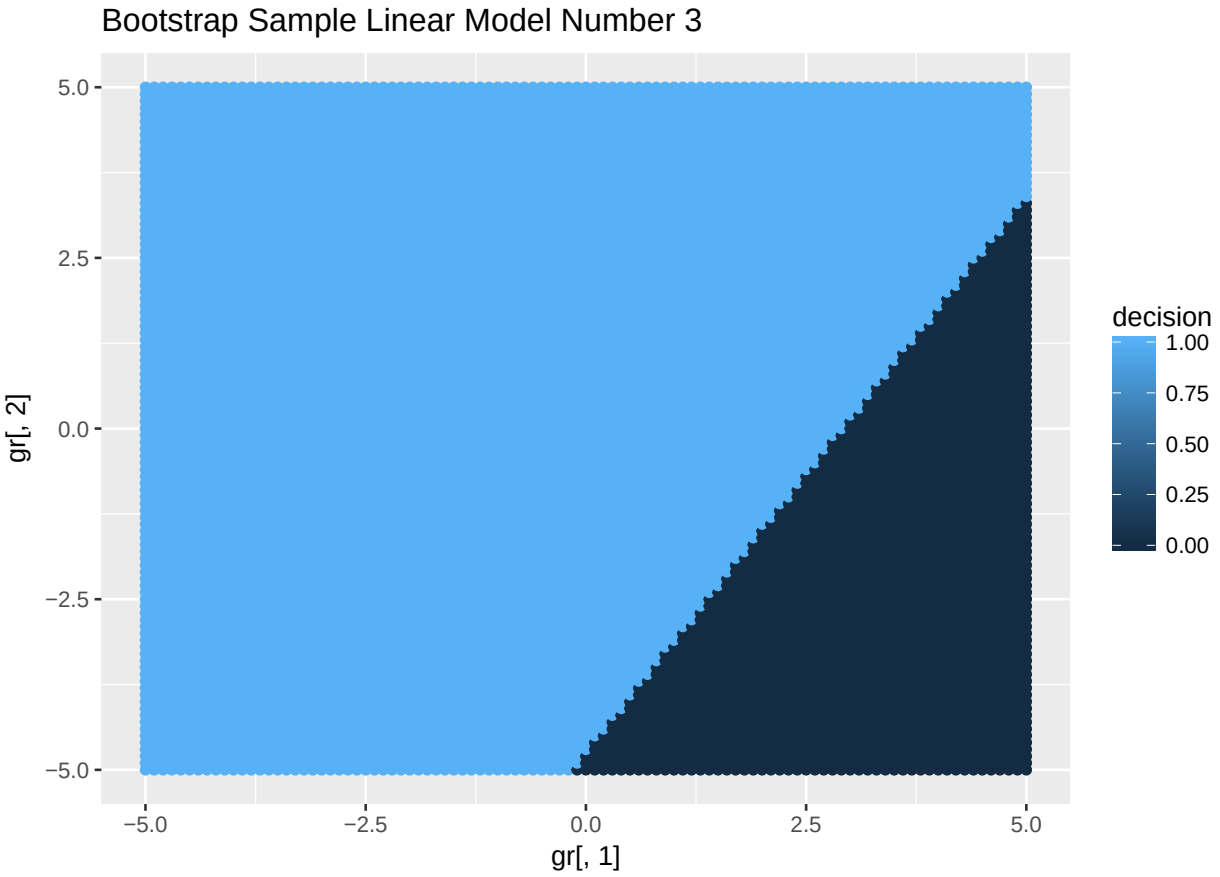

Bootstrap Sample Linear Model Number 1
5.0
2.5
0.0
−2.5
−5.0
gr[, 2]
−5.0
−2.5
0.0
2.5
5.0
gr[, 1]
decision
1.00
0.75
0.50
0.25
0.00




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


Bootstrap Sample Linear Model Number 2
5.0
2.5
0.0
−2.5
−5.0
gr[, 2]
−5.0
−2.5
0.0
2.5
5.0
gr[, 1]
decision
1.00
0.75
0.50
0.25
0.00


Bootstrap Sample 5th Degree Poly Model Number 3
5.0
2.5
0.0
−2.5
−5.0
gr[, 2]
−5.0
−2.5
0.0
2.5
5.0
gr[, 1]
decision
1.00
0.75
0.50
0.25
0.00



We see that for the 5th degree polynomial models there is a very high degree of variation in the curved boundaries, with especially bad predictions in the left hand side of the plot near side, top, and bottom edges for more than one plot. For the linear model, we see high bias and also high variance between samples, in two cases the decison boundary line is closer to vertical than diagonal. These plot clearly show the extrmely high variance of the 5th degree polynomial model and the high bias and considerable variance of the linear model.