Homework Assignment 4

Ryan Avery and Meet Gala

June 03, 2018

```
## 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
## 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
## Loading required package: survival
## Loading required package: lattice
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.3
## Loading required package: gplots
##
## Attaching package: 'gplots'
## The following object is masked from 'package:stats':
##
##     lowess
## 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
```

1. Fundamentals of the bootstrap

a)

b) 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.

```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

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

```
##   2.5%  97.5%
## 0.4059 0.6040
```

2. Two Chainz

```r
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(faces_array[,,1])
```

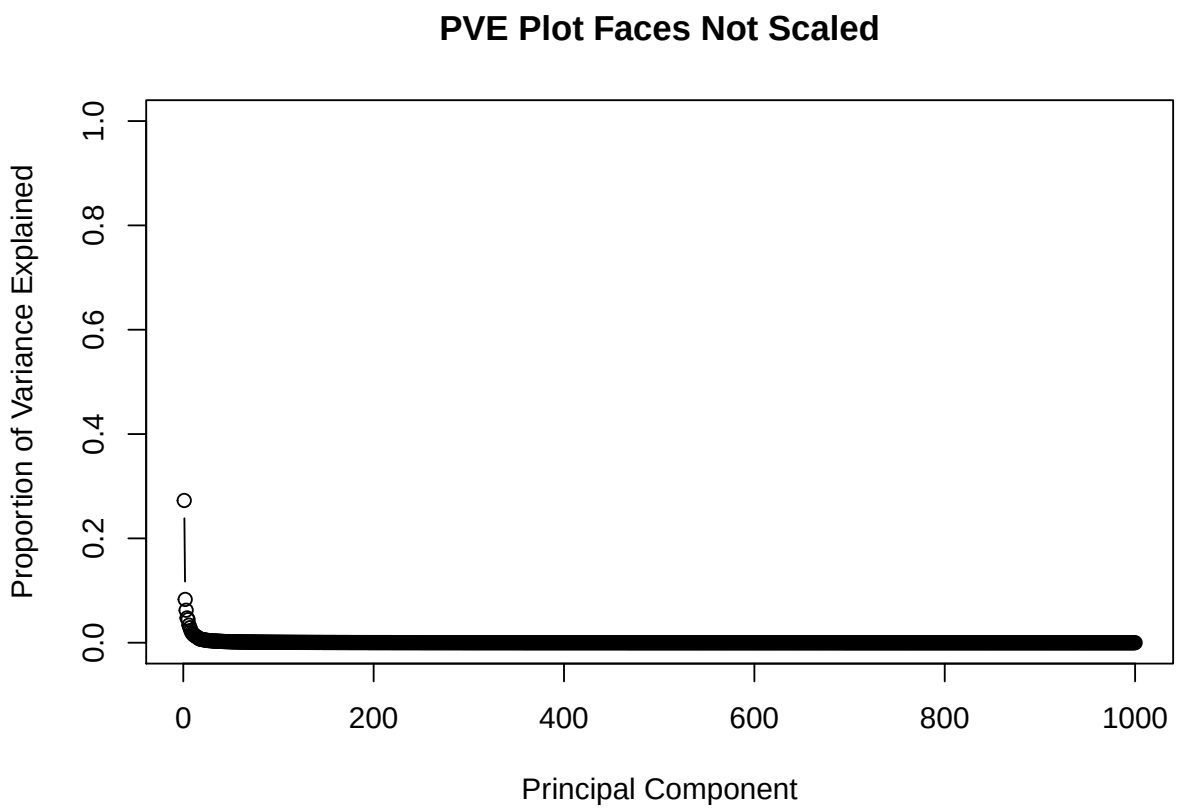
A) Average Face

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

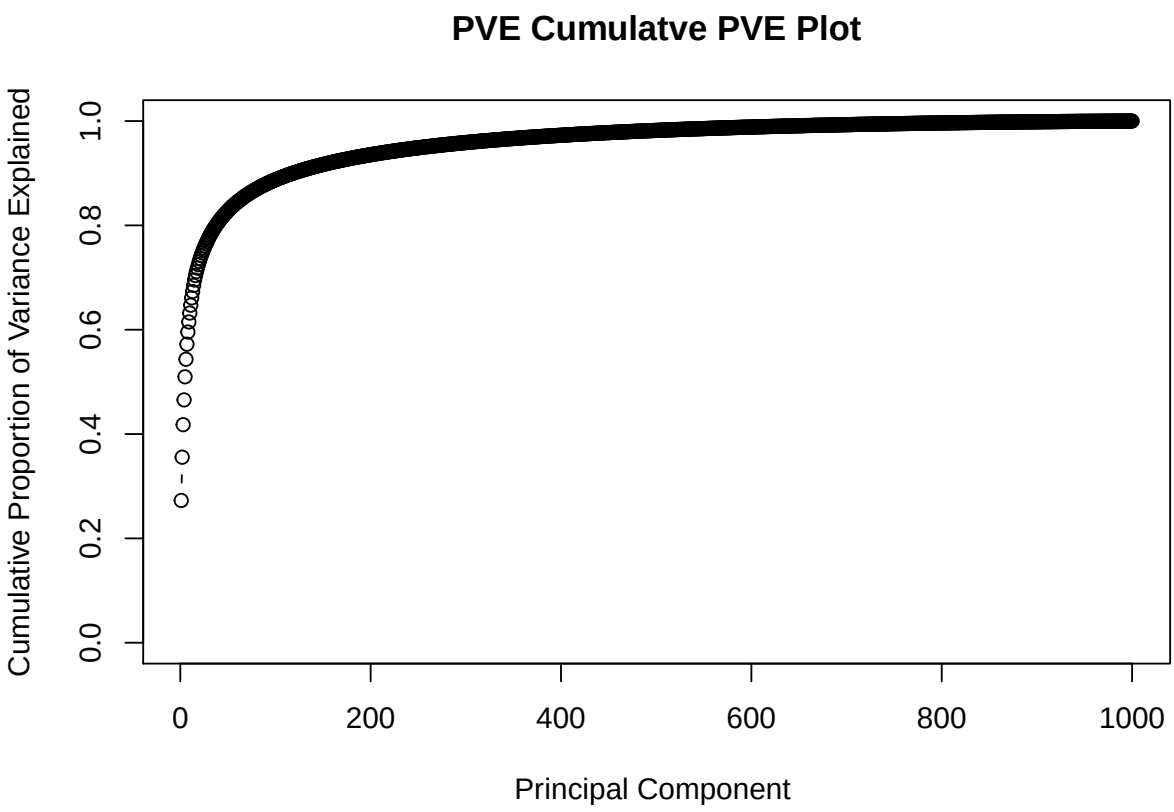
b)S cale comparison and plotting the PVE and cumulative PVE. After the fifth Principal Component, 50 percent of the variance is explaineed. PVE from the PCA.

```
faces.out.scale=prcomp(face_mat, scale=TRUE)
faces.out.noscale = prcomp(face_mat, scale=FALSE)
```

```
faces.var = faces.out.noscale$sdev^2
pve = faces.var/sum(faces.var)
plot(pve, xlab="Principal Component", ylab="Proportion of Variance Explained",ylim= c(0,1), type='b')
title('PVE Plot Faces Not Scaled')
```




```
plot(cumsum(pve), xlab="Principal Component ",
ylab=" Cumulative Proportion of Variance Explained ", ylim=c(0,1), type='b')
title('PVE Cumulatve PVE Plot')
```




```
min(which(cumsum(pve) > .5))
```

```
## [1] 5
```

```
print("after the fifth Principal Component, 50 percent of the variance is explained")
```

```
## [1] "after the fifth Principal Component, 50 percent of the variance is explained"
```

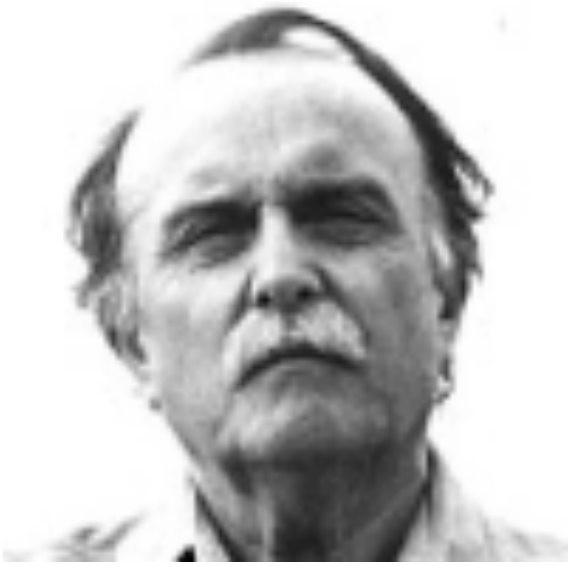
c) Plotting the first 16 principal components

```
for (i in 1:16) {
  plot_face(faces.out.noscale$rotation[,i])
}
```

d) PC 1 primarily describes variation between a face and the background.

```
for (i in head(sort(faces.out.noscale$x[,1]),5)){
  index = which(i==faces.out.noscale$x[,1])
  plot_face(faces_array[,,index])
}
```

```
for (i in tail(sort(faces.out.noscale$x[,1]),5)){
  index = which(i==faces.out.noscale$x[,1])
  plot_face(faces_array[,,index])
}
```



e) PC5 shows variation between features typically associated with male or females. The most apparent one is long hair vs short hair.

```
for (i in head(sort(faces.out.noscale$x[,5]),5)){
  index = which(i==faces.out.noscale$x[,5])
  plot_face(faces_array[,,index])
}
```

```
for (i in tail(sort(faces.out.noscale$x[,5]),5)){
  index = which(i==faces.out.noscale$x[,5])
  plot_face(faces_array[,,index])
}
```

PC5 better resolves differences between individuals, particularly those with long hair and short hair. It would be more useful in a facial recognition model with the goal of detecting differences between people. If the goal of the model was to detect any face in an image with many other objects, PC1 might be better.

f)

```
dim(faces.out.noscale$x)
```

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

```
par(mfrow=c(1, 5))
avg_face <- colMeans(face_mat)
for (i in c(10,50,100,300)){
  compression<-faces.out.noscale$x[,1:i] %*% t(faces.out.noscale$rotation)[1:i,]
  plot_face(compression[5,])+avg_face}
```




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)

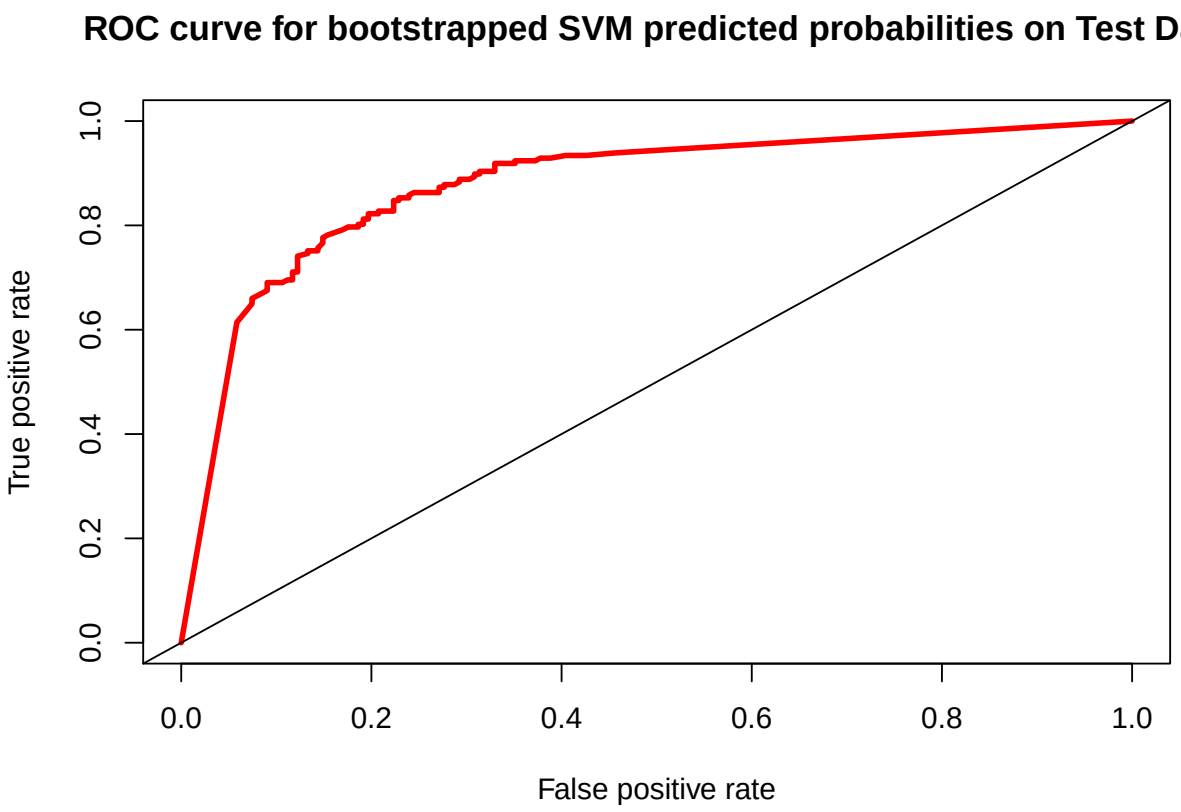
```
class_counts = 0
for (i in 1:200){
ind = sample(nrow(drug_use_train),size=nrow(drug_use_train), replace = TRUE)
train_boot = drug_use_train[ind,]
svmfit=svm(recent_cannabis_use ~ ., data=train_boot, kernel="radial", cost=1,scale=TRUE)
pred = predict(svmfit, drug_use_test)
pred = ifelse(pred=="Yes",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



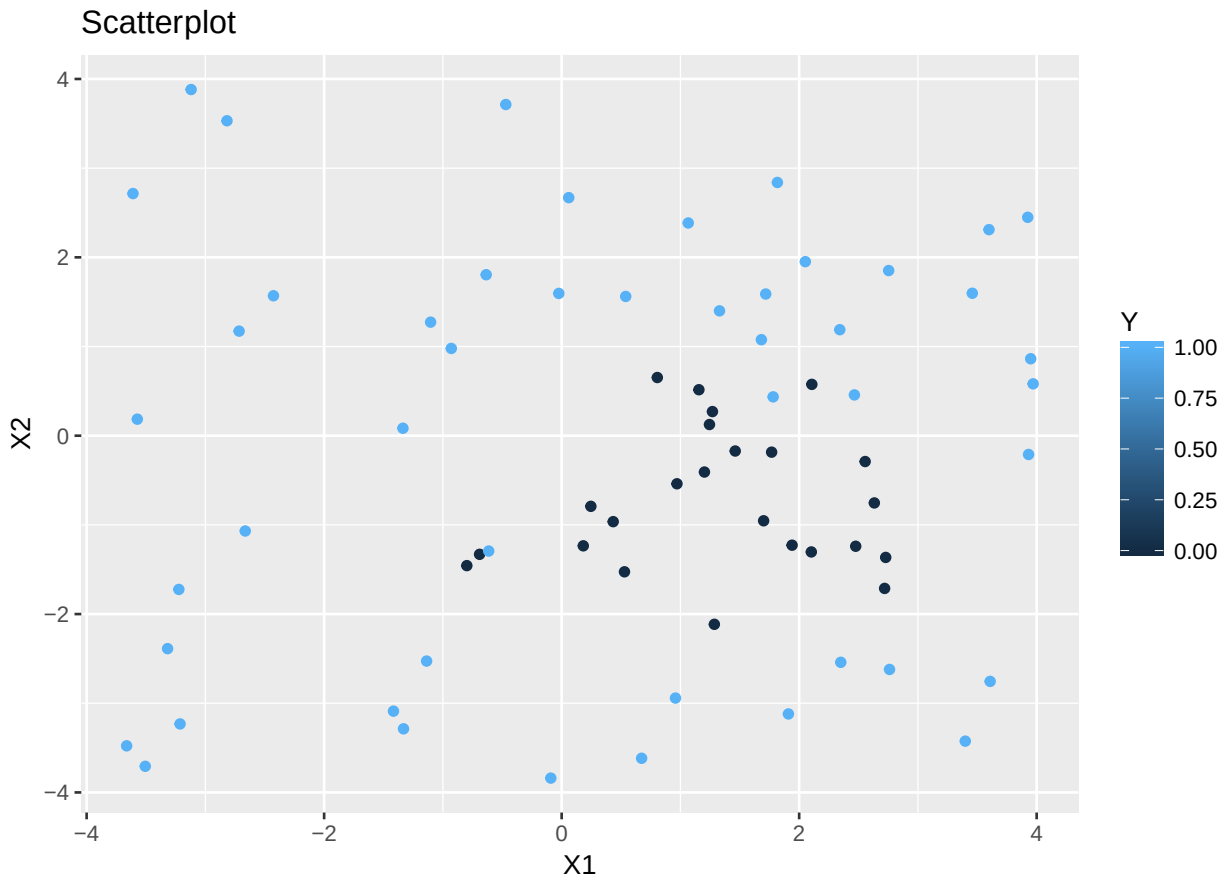
5.

a)

```
nonlinear <-read_csv('nonlinear.csv')
```

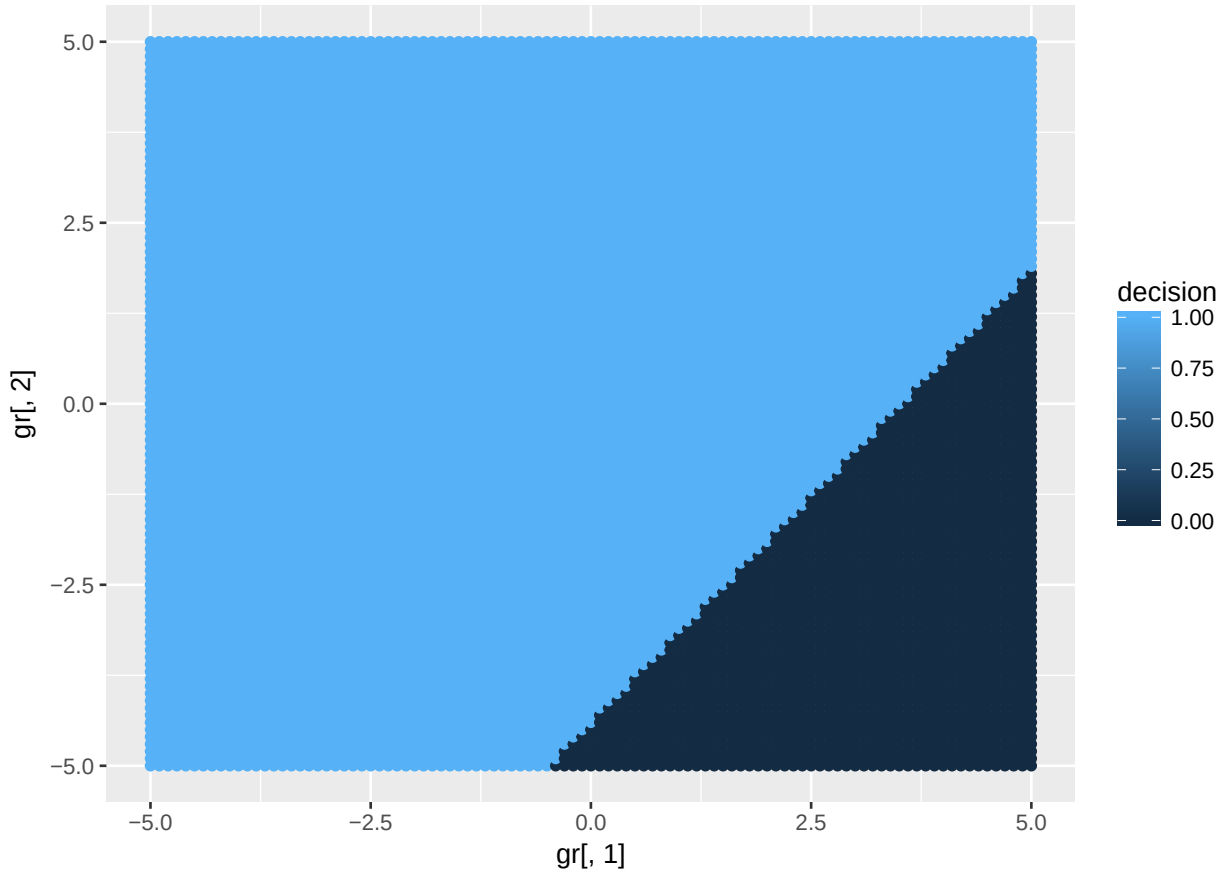
```
## Parsed with column specification:
## cols(
##   Z = col_integer(),
##   X1 = col_double(),
##   X2 = col_double(),
##   Y = col_integer()
## )
```

```
ggplot(data = nonlinear, aes(x=X1,y=X2, color=Y)) +
  geom_point() +
  labs(title='Scatterplot') +
  ylab('X2') +
  xlab('X1')
```



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

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




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

```
##
## Call:
## glm(formula = Y ~ X1 + X2, family = binomial, data = nonlinear)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.594  -1.248   0.626   0.915   1.511
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.022      0.314    3.26   0.0011 **
## X1            -0.289      0.136   -2.13   0.0334 *
## X2             0.232      0.144    1.62   0.1056
## ---
## 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: 90.52
##
## Number of Fisher Scoring iterations: 4
```

c)

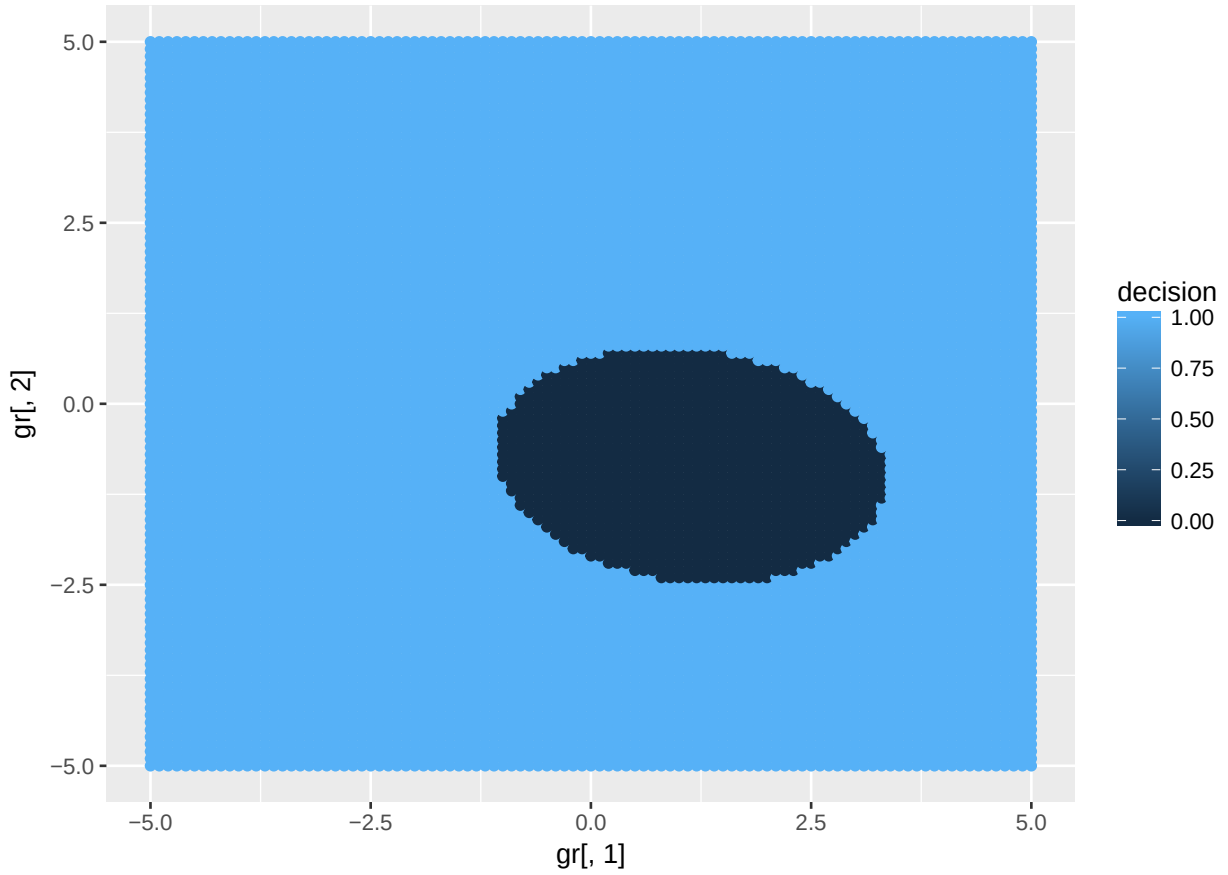
```
X<-as.matrix(nonlinear[,2:3],nrow=36,ncol=2)
nonlinear_poly=as.data.frame(poly(X,2,raw=TRUE))
nonlinear_poly$Y = nonlinear$Y
logistic.poly.fit = glm(Y ~ .,data=nonlinear_poly, family=binomial)
```

```
## 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(gr[,1],gr[,2],color=decision)
```




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

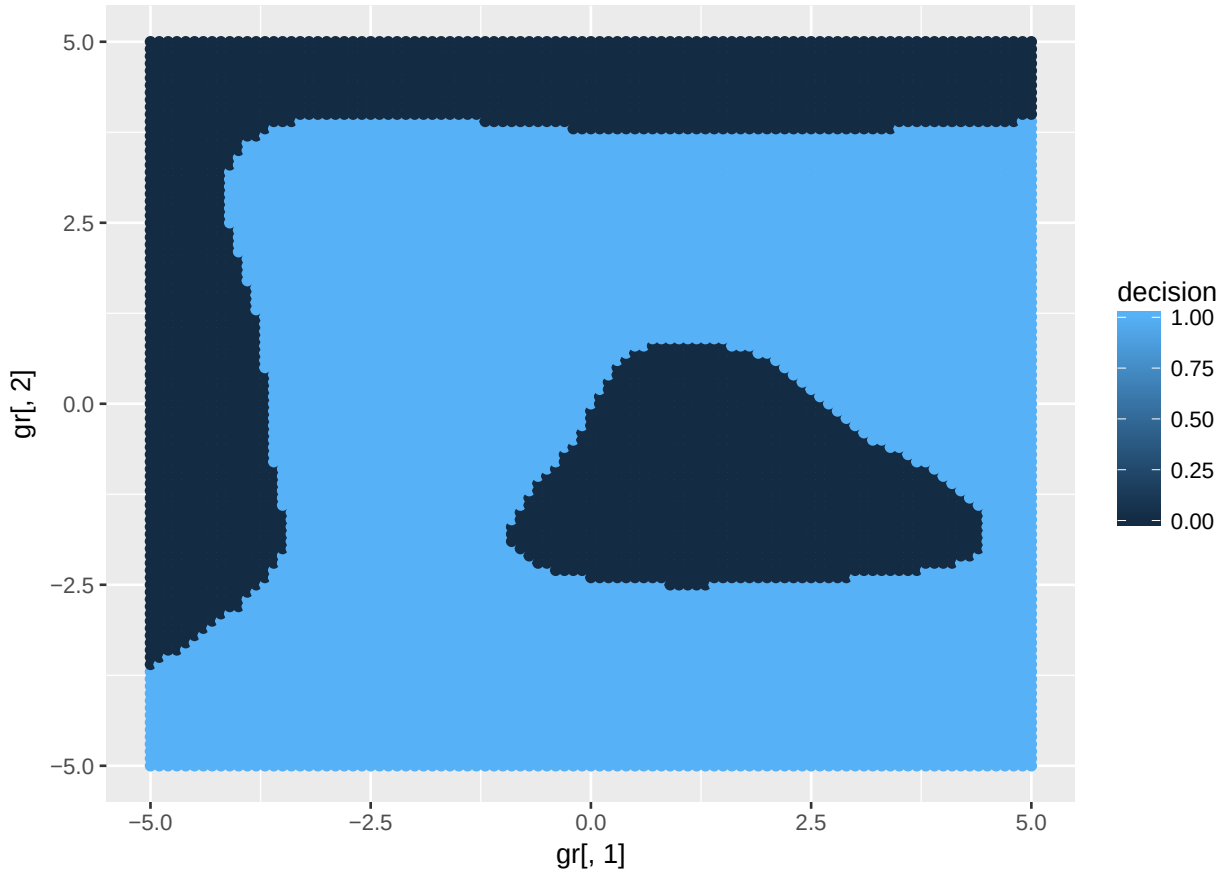
```
##
## Call:
## glm(formula = Y ~ ., family = binomial, data = nonlinear_poly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3908  -0.0827   0.0000   0.0093   1.9007
##
## Coefficients: (4 not defined because of singularities)
##             Estimate Std. Error z value 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
```

```
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
```

```
c(1:6,11,15,18,20)
```

```
##  [1]  1  2  3  4  5  6 11 15 18 20
```

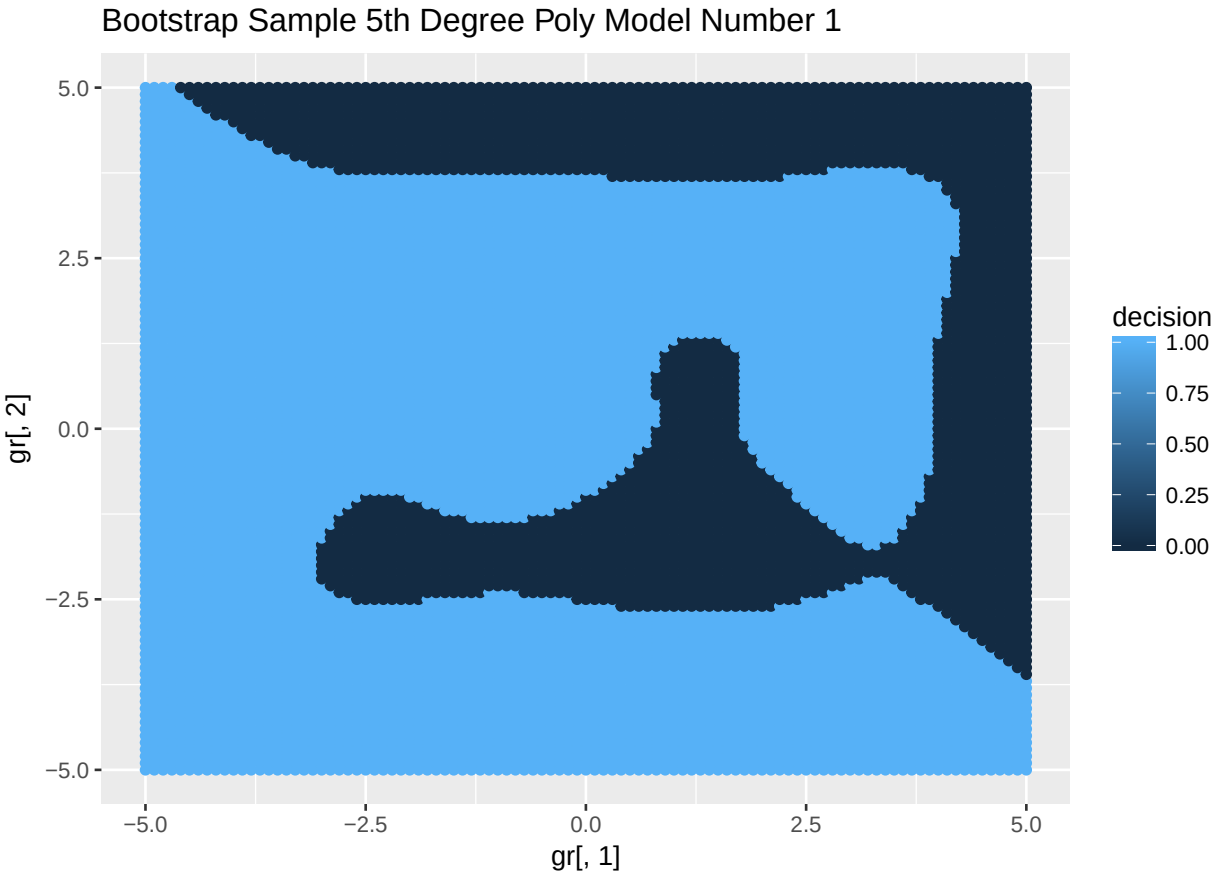
e) The coefficients for the X1^2, X2^2, and X1*X2 terms in the polynomial logistic model are all significant with a p value of less than .05. None of the coefficeints are significant for the 5th order polynomial model due to overfitting. The variance is extremely high for the 5th order polynomial model. 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
}
```

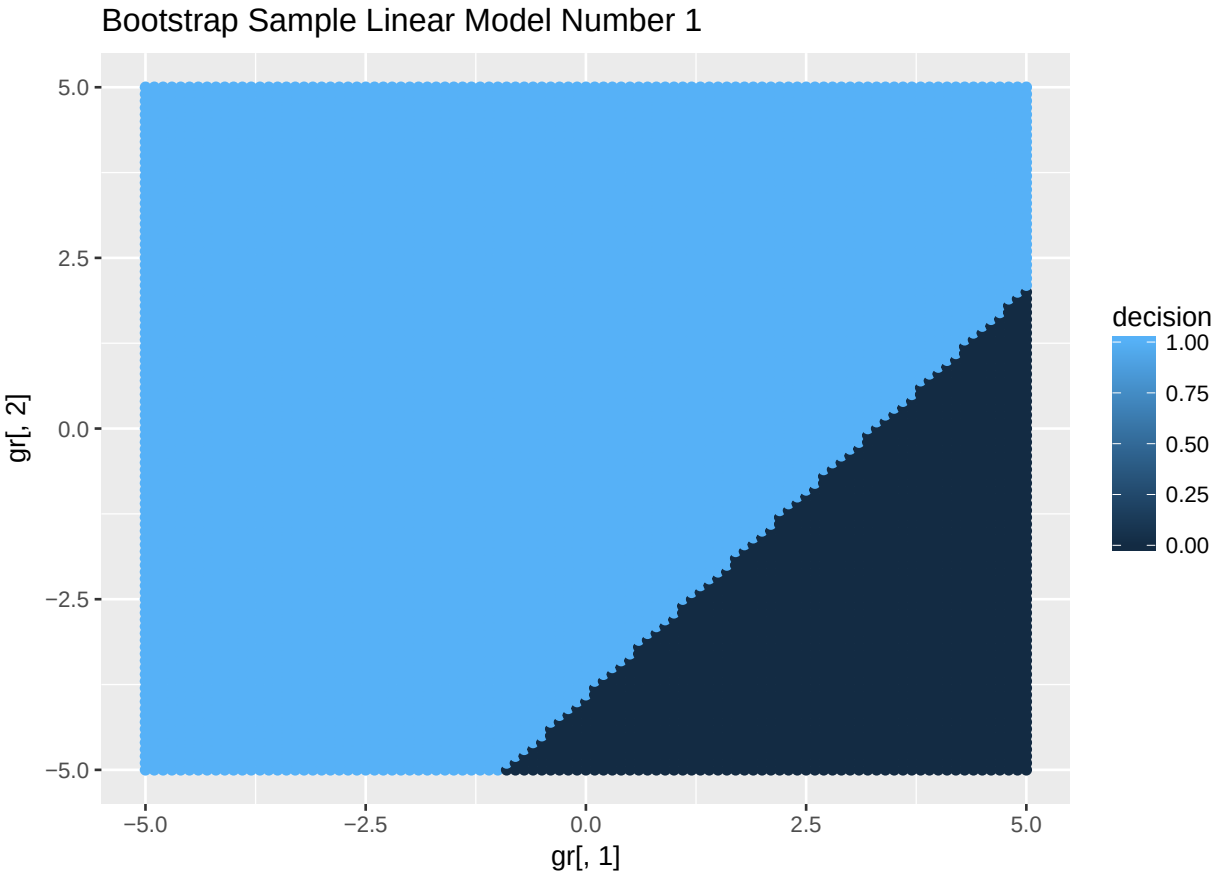
```
## Warning: glm.fit: algorithm did not converge
```

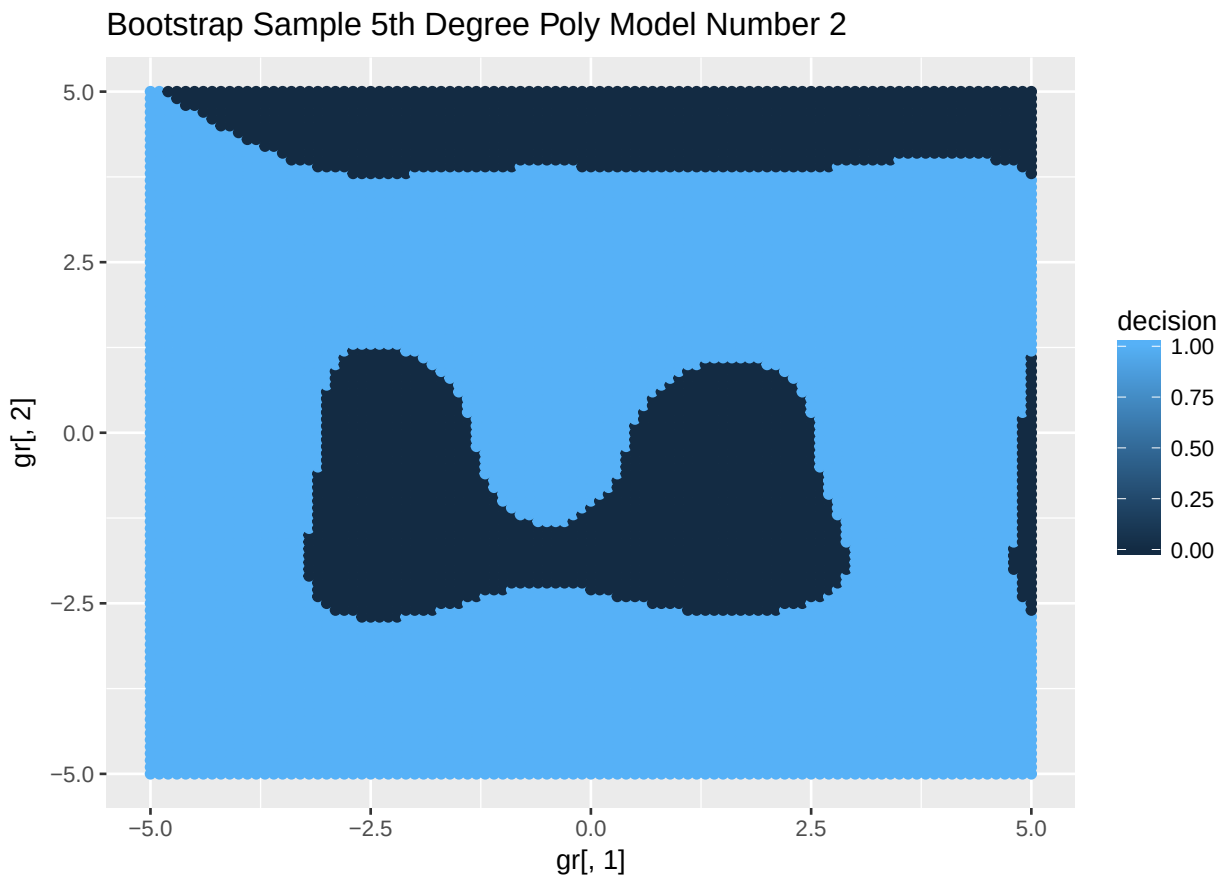
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



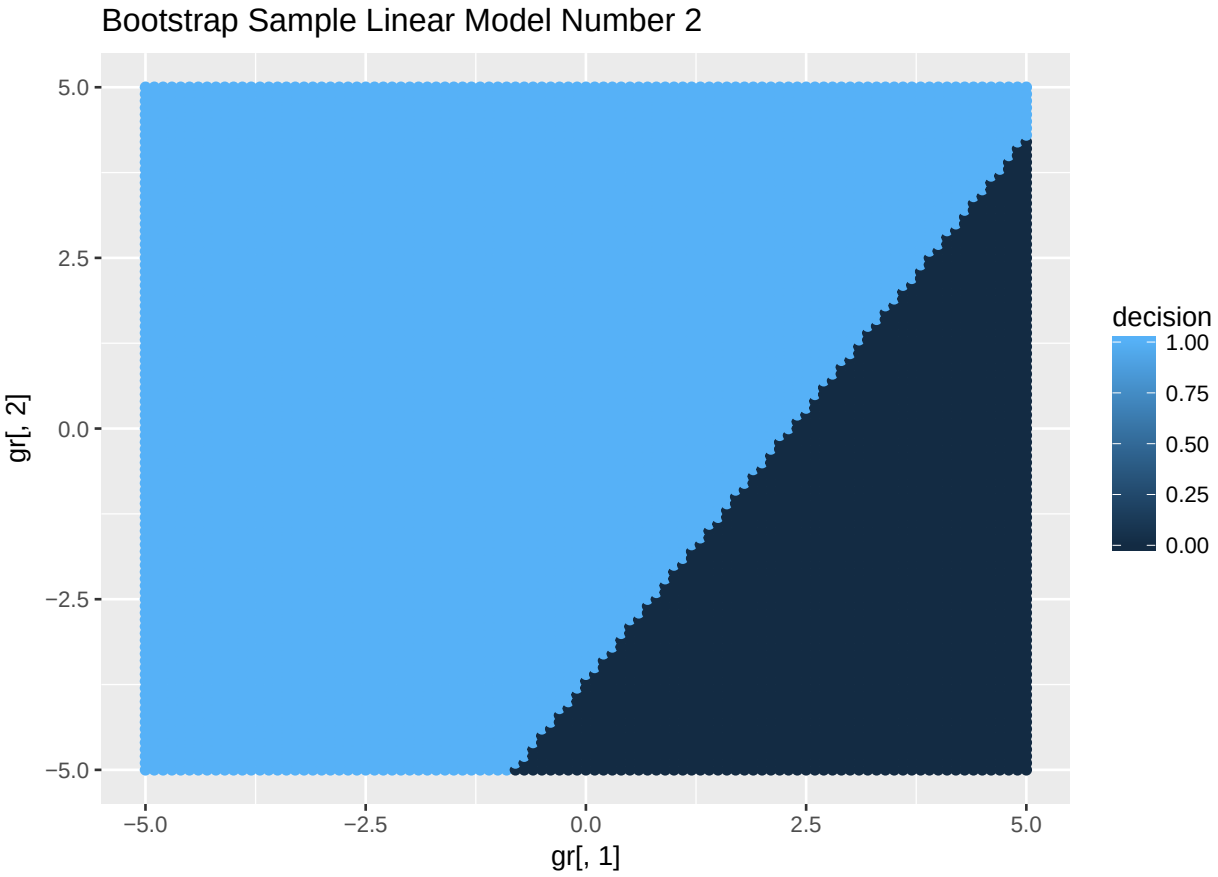

```
## Warning: glm.fit: algorithm did not converge
```

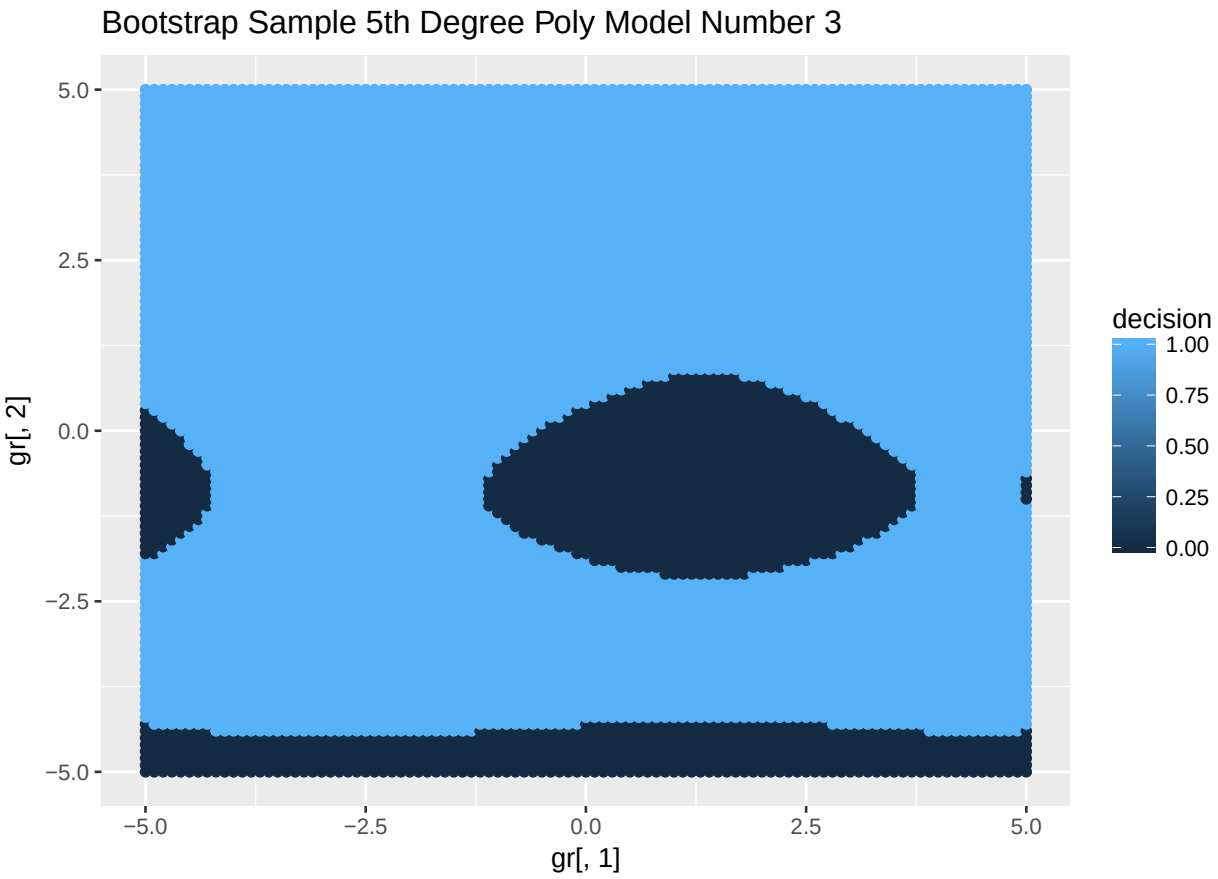
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

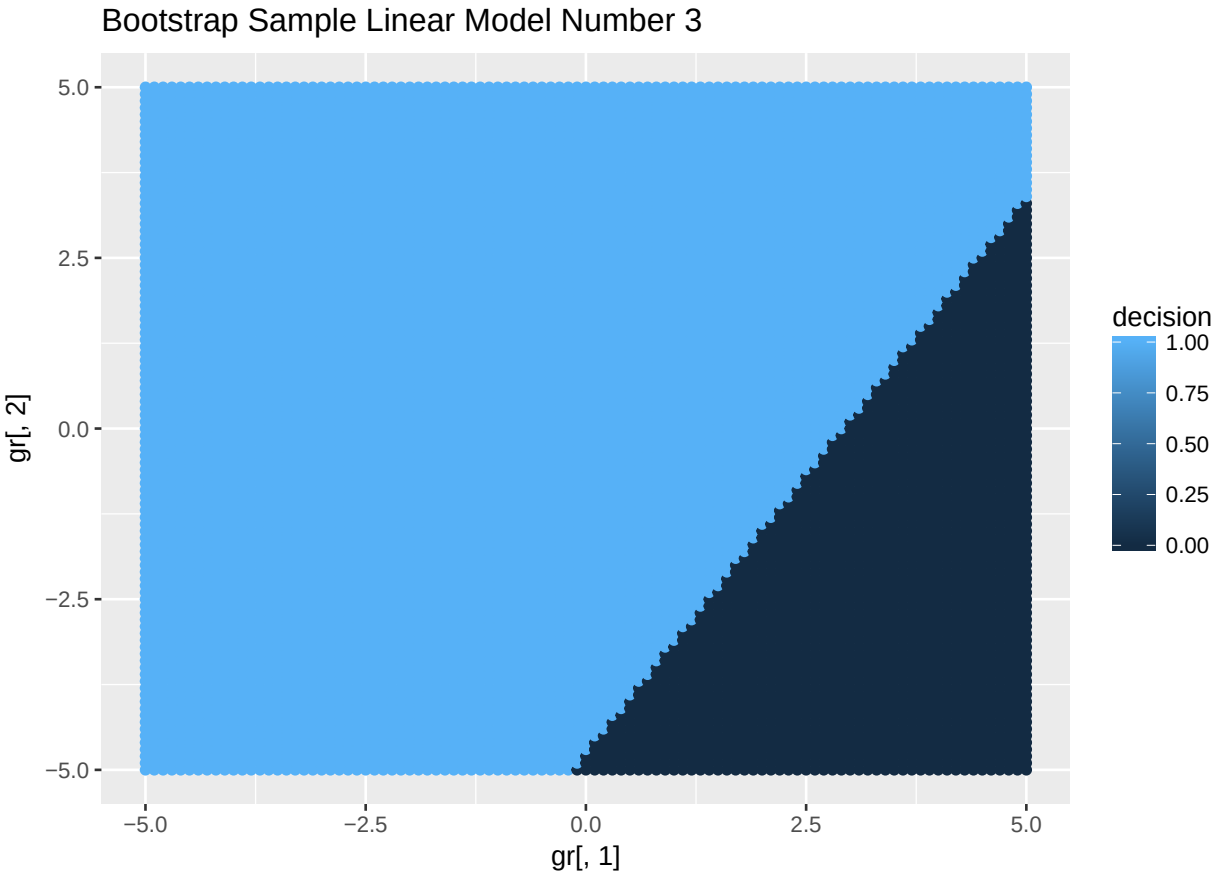

Bootstrap Sample Linear Model Number 1
5.0
2.5
0.0
−2.5
−5.0
gr[, 2]
−5.0
−2.5
0.0
2.5
5.0
gr[, 1]
decision
1.00
0.75
0.50
0.25
0.00




```
## 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.