Duncan McKinnon

West

W 7

QA

```
# Load packages and data
suppressPackageStartupMessages({
  library(tidyverse)
  library(rpart)
  library(partykit)
  library(randomForest)
  library(NHANES)
})
```

1).

```
# set random seed
set.seed(1847)

# get sample size and training proportion
relevant <- c("HardDrugs", "RegularMarij", "Age", "AlcoholYear" , "BMI" , "Gender" , "HHIncomeMid" , "We
m <- dim(NHANES[,relevant])
p <- 0.2

train_indices <- sample.int(m[1], (1-p) * m[1])
train_data <- NHANES[train_indices, relevant]
test_data <- NHANES[-train_indices, relevant]

glimpse(train_data)
```

```
## Observations: 8,000
## Variables: 11
## $ HardDrugs       <fct> No, No, No, No, NA, NA, Yes, No, No, NA, No, N...
## $ RegularMarij    <fct> No, Yes, Yes, No, NA, NA, No, NA, No, NA, No, ...
## $ Age             <int> 26, 55, 22, 18, 26, 80, 37, 68, 59, 16, 38, 70...
## $ AlcoholYear     <int> 52, 0, 12, NA, NA, 0, 12, 200, NA, NA, NA, 0, ...
## $ BMI             <dbl> 22.66, 26.00, 22.80, 26.80, 21.00, 22.33, 31.9...
## $ Gender          <fct> female, male, male, male, female, male, female...
## $ HHIncomeMid     <int> 50000, 50000, 70000, 30000, 100000, 12500, 100...
## $ Weight          <dbl> 67.9, 93.7, 73.5, 78.6, 50.4, 64.7, 89.1, 100....
## $ Height          <dbl> 173.1, 189.7, 179.6, 171.2, 154.9, 170.2, 167....
## $ SexNumPartnLife <int> 0, 7, 90, 23, NA, NA, 2, 40, 3, NA, 5, NA, 6, ...
## $ TotChol         <dbl> 3.72, 3.72, 4.84, 4.19, 3.49, 5.69, 5.30, 4.19...
```

```
# create modeling formula
f <- formula(HardDrugs ~ RegularMarij + Age + AlcoholYear + BMI + Gender + HHIncomeMid + Weight + Heigh
m <- c(m[1], 10)
```

2).

```r
# create bagging model with 100 trees
bagged_trees_100 <- randomForest(f, data = train_data, mtry = m[2], ntree = 100, na.action = na.omit)

bagged_trees_100
```

```
##
## Call:
##  randomForest(formula = f, data = train_data, mtry = m[2], ntree = 100,      na.action = na.omit)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 8.66%
## Confusion matrix:
##       No Yes class.error
## No  2362 103  0.04178499
## Yes  169 506  0.25037037
```

3).

```r
# create bagging model with 500 trees
bagged_trees_500 <- randomForest(f, data = train_data, mtry = m[2], ntree = 500, na.action = na.omit)

bagged_trees_500
```

```
##
## Call:
##  randomForest(formula = f, data = train_data, mtry = m[2], ntree = 500,      na.action = na.omit)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 8.28%
## Confusion matrix:
##       No Yes class.error
## No  2370  95  0.03853955
## Yes  165 510  0.24444444
```

4).

```r
# create bagging model with 1000 trees
bagged_trees_1000 <- randomForest(f, data = train_data, mtry = m[2], ntree = 1000, na.action = na.omit)

bagged_trees_1000
```

```
##
## Call:
##  randomForest(formula = f, data = train_data, mtry = m[2], ntree = 1000,      na.action = na.omit)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 8.22%
## Confusion matrix:
##       No Yes class.error
## No  2372  93  0.03772819
## Yes  165 510  0.24444444
```

5).

The three models shown above used bagging with different numbers of trees. Despite the large variation in the number of trees used, all of the models had relatively similar OOB error rate and performed similarly in cross validation. The models generated ended up being similar because every tree they created had to account for all the predictive variables within the sample set used. Because every tree had the same predictors, increasing the number of trees did little to effect error rate because most of the trees would be expected to come to similar conclusions about the importance of the predictors. By using all the predictors on each tree, we greatly limit the number of possible decision trees that can go into our random forest, and with so many similar training samples, increasing the number of trees only increases duplication.

6).

```
# get importance of predictors for bagged rf with 1000 trees
importance(bagged_trees_1000)
```

```
##                  MeanDecreaseGini
## RegularMarij            224.34124
## Age                     113.57964
## AlcoholYear              82.59351
## BMI                      95.07387
## Gender                   12.05194
## HHIncomeMid              62.33526
## Weight                   95.17922
## Height                  116.28053
## SexNumPartnLife         128.61981
## TotChol                 130.57862
```

7).

From this importance we can see that the predictor with the greatest mean decrease in gini impurity was regular marijuana use. After that both total cholesterol and mean number of partners in life were similarly important, and may even have masked each other in many trees. The same can be said of Height and Age.

8).

```
# predict test set on bagged rf with 1000 trees
test_pred_bagged_1000 <- predict(bagged_trees_1000, newdata=test_data)

# create confusion matrix
conf_matrix <- table(test_pred_bagged_1000, test_data$HardDrugs)

conf_matrix
```

```
##
## test_pred_bagged_1000  No Yes
##                   No  569  55
##                   Yes  33 143
```

9).

```r
# create random forest model with 500 trees and 5 predictors/tree
rf_5 <- randomForest(f, data = train_data, mtry = 5, ntree = 500, na.action = na.omit)

rf_5
```

```
##
## Call:
##  randomForest(formula = f, data = train_data, mtry = 5, ntree = 500,      na.action = na.omit)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 8.15%
## Confusion matrix:
##       No Yes class.error
## No  2374  91  0.03691684
## Yes  165 510  0.24444444
```

10).

```r
# create random forest model with 500 trees and 2 predictors/tree
rf_2 <- randomForest(f, data = train_data, mtry = 2, ntree = 500, na.action = na.omit)

rf_2
```

```
##
## Call:
##  randomForest(formula = f, data = train_data, mtry = 2, ntree = 500,      na.action = na.omit)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 8.06%
## Confusion matrix:
##       No Yes class.error
## No  2392  73   0.0296146
## Yes  180 495   0.2666667
```

11).

```r
# create random forest model with 500 trees and 1 predictors/tree
rf_1 <- randomForest(f, data = train_data, mtry = 1, ntree = 500, na.action = na.omit)
```

```r
rf_1
```

```
##
## Call:
##  randomForest(formula = f, data = train_data, mtry = 1, ntree = 500,      na.action = na.omit)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 11.59%
## Confusion matrix:
##       No Yes class.error
## No  2430  35  0.01419878
## Yes  329 346  0.48740741
```

12).

After testing the random forest on the training data using 1, 2 and 5 predictors per tree, I would use the model that selected 5 predictors per tree. With 1 predictor, there can only be 1 split per tree, and few of the tree will include the most significant predictors, leading to the highest OOB error (11.59%). Using 2 predictors had the lowest OOB error (8.06), and seemed less likely to penalize slight differences in the significance of predictors than using 10 or even 5 predictors, since many of the trees generated would not have similarly significant predictors.

QB

1).

```r
# set random seed
set.seed(1847)

# create sample of data
data_unif <- data.frame( y = sample(c('yes','no'), 500, replace = T), x1 = runif(500, 0, 10), x2 = runi

glimpse(data_unif)
```

```
## Observations: 500
## Variables: 3
## $ y  <fct> yes, no, no, no, no, yes, yes, no, yes, yes, no, no, no, no...
## $ x1 <dbl> 8.3793318, 4.4838668, 2.8321497, 2.5175372, 8.5770707, 7.51...
## $ x2 <dbl> 4.67711044, 9.17958498, 5.94734839, 2.56939501, 7.70956941,...
```

2).

```r
# set random seed
set.seed(1847)
m <- 500
p <- 0.2

train_indices_unif <- sample.int(m, (1-p)*m)
```

```
train_data_unif <- data_unif[train_indices_unif, ]
test_data_unif <- data_unif[-train_indices_unif, ]

glimpse(train_data_unif)
```

```
## Observations: 400
## Variables: 3
## $ y  <fct> no, no, no, no, no, yes, yes, no, yes, no, yes, yes, no, no...
## $ x1 <dbl> 8.59285014, 9.37109700, 6.36942114, 8.83725895, 4.83748983,...
## $ x2 <dbl> 1.2488676, 9.7732755, 2.7639338, 6.9972891, 5.7779546, 9.05...
```

3).

```
# create formula to model
f_unif <- formula(y ~ x1 + x2)

# create bagging model with 100 trees
bagged_unif_100 <- randomForest(f_unif, data = train_data_unif, mtry = 2, ntree = 100, na.action = na.o

bagged_unif_100
```

```
##
## Call:
##  randomForest(formula = f_unif, data = train_data_unif, mtry = 2,      ntree = 100, na.action = na.o
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 53.75%
## Confusion matrix:
##      no yes class.error
## no  101 103   0.5049020
## yes 112  84   0.5714286
```

4).

The out of bag error in this bagging model was a little over 50% (53.75%). This is what we would expect given that the labels and predictors were randomly generated. We wouldn't expect there to be any correlation between x1,x2 and y, and since y has two possible labels we would expect about half the predictions to match the labels just due to random chance.

5).

```
# predict the test set
pred_test_unif_100 <- predict(bagged_unif_100, newdata = test_data_unif)

# create confusion matrix
conf_matrix_unif <- table(pred_test_unif_100, test_data_unif$y)

conf_matrix_unif
```

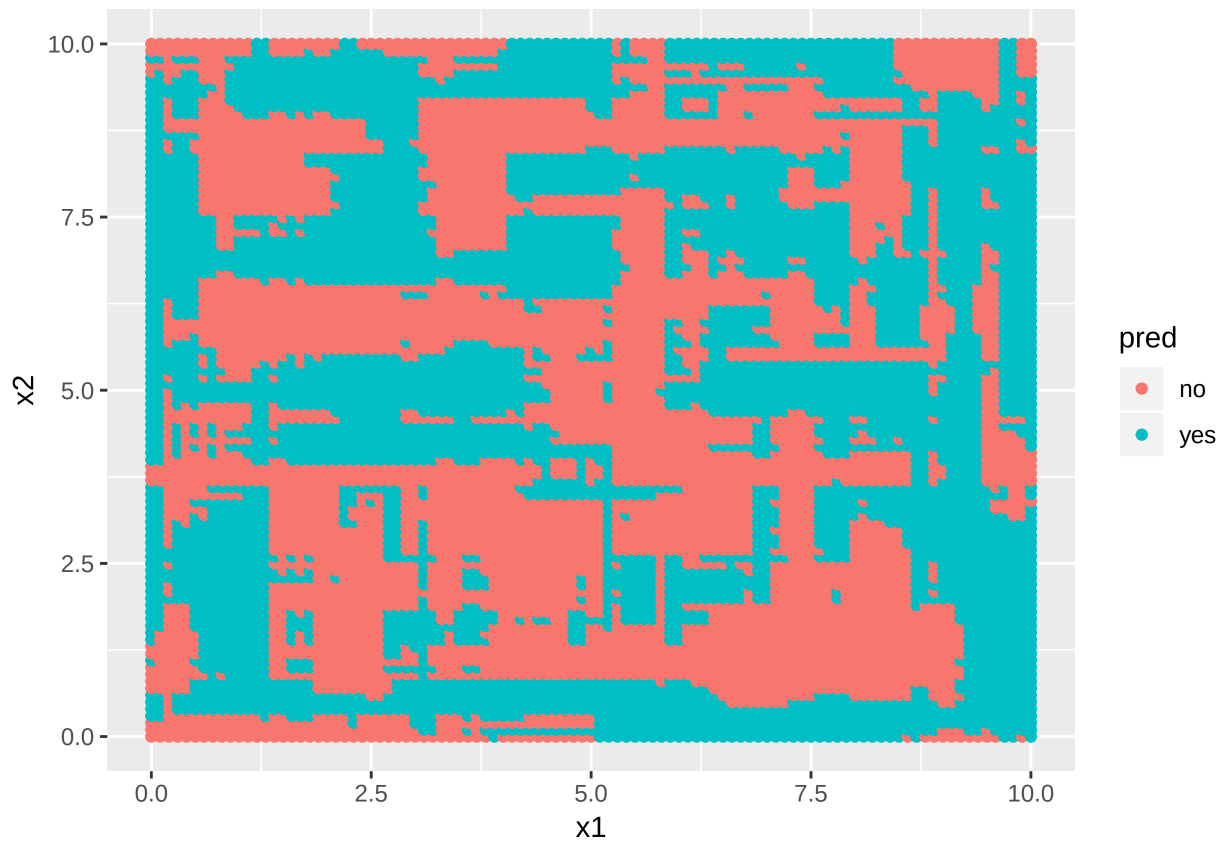
```
##
## pred_test_unif_100 no yes
##                no  27  24
##                yes 26  23
```

6).

```r
# create test grid
test_grid_unif <- expand.grid(x1 = seq(0, 10, 0.1), x2 = seq(0, 10, 0.1))

# predict values of test grid with bagged model
test_grid_unif$pred <- predict(bagged_unif_100, newdata = test_grid_unif, type="class")

# plot predictions
pred_plot <- ggplot(test_grid_unif,aes(x=x1,y=x2,color=pred))+geom_point()
pred_plot
```



QC

```r
# set random seed
set.seed(1847)

# group 1
```

```r
x1 <- rnorm(250, 6, 1)
x2 <- rnorm(250, 6, 1)
y <- rep('yes', 250)
g1 <- cbind.data.frame(x1, x2, y)
glimpse(g1)
```

```
## Observations: 250
## Variables: 3
## $ x1 <dbl> 4.548232, 6.400951, 6.162160, 5.524947, 4.733338, 6.799025,...
## $ x2 <dbl> 6.985999, 5.426683, 7.070074, 7.000385, 7.357278, 4.935165,...
## $ y  <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes,...
```

```r
# set random seed
set.seed(1847)

# group 2
x1 <- rnorm(250, 5, 1)
x2 <- rnorm(250, 5, 1)
y <- rep('no', 250)
g2 <- cbind.data.frame(x1, x2, y)
glimpse(g2)
```

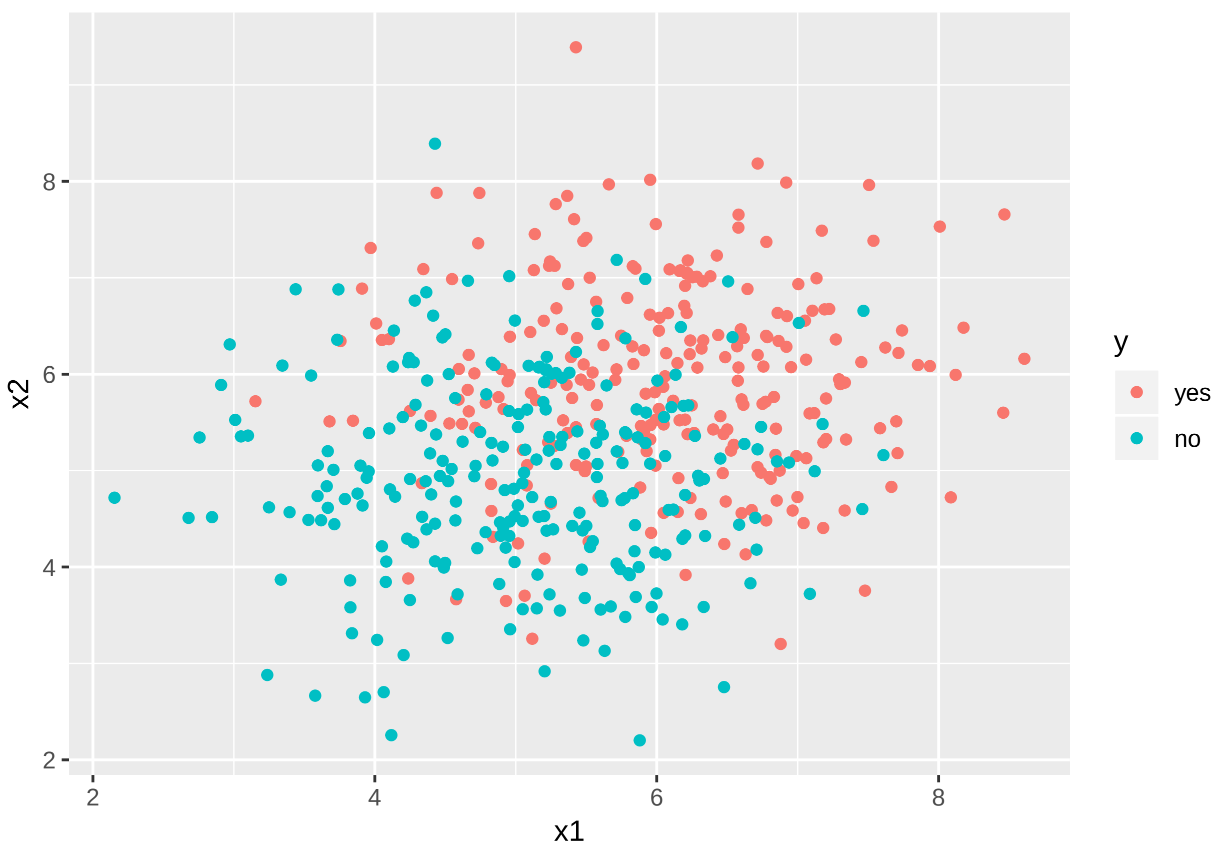
```
## Observations: 250
## Variables: 3
## $ x1 <dbl> 3.548232, 5.400951, 5.162160, 4.524947, 3.733338, 5.799025,...
## $ x2 <dbl> 5.985999, 4.426683, 6.070074, 6.000385, 6.357278, 3.935165,...
## $ y  <fct> no, no, no, no, no, no, no, no, no, no, no, no, no, no, no,...
```

```r
# combine g1 and g2
sim_dat3 <- rbind(g1, g2)
glimpse(sim_dat3)
```

```
## Observations: 500
## Variables: 3
## $ x1 <dbl> 4.548232, 6.400951, 6.162160, 5.524947, 4.733338, 6.799025,...
## $ x2 <dbl> 6.985999, 5.426683, 7.070074, 7.000385, 7.357278, 4.935165,...
## $ y  <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes,...
```

1).

```r
# scatter plot of sim data and labels
ggplot(sim_dat3) +
  geom_point(aes(x = x1, y = x2, color = y))
```



2).

```
# set random seed
set.seed(1847)
m <- 500
p <- 0.2

# create training and testing datasets
train_indices_sim <- sample.int(m, (1-p)*m)
train_data_sim <- sim_dat3[train_indices_sim,]
test_data_sim <- sim_dat3[-train_indices_sim,]

glimpse(train_data_sim)
```

```
## Observations: 400
## Variables: 3
## $ x1 <dbl> 7.708587, 5.775904, 4.951744, 4.516853, 4.835113, 5.715045,...
## $ x2 <dbl> 5.180246, 5.397889, 5.617803, 3.264038, 5.104902, 6.049787,...
## $ y  <fct> yes, no, no, no, no, yes, yes, no, yes, yes, no, no, no, no...
```

3).

```
# create formula to model
f_sim <- formula(y ~ x1 + x2)
```

```
# create bagging model with 100 trees
bagged_sim_100 <- randomForest(f_sim, data = train_data_sim, mtry = 2, ntree = 100, na.action = na.omit

bagged_sim_100
```

```
##
## Call:
##  randomForest(formula = f_sim, data = train_data_sim, mtry = 2,      ntree = 100, na.action = na.omi
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 31.75%
## Confusion matrix:
##     yes  no class.error
## yes 128  64   0.3333333
## no   63 145   0.3028846
```

4).

```
# predict the test set
pred_test_sim_100 <- predict(bagged_sim_100, newdata = test_data_sim)

# create confusion matrix
conf_matrix_sim <- table(pred_test_sim_100, test_data_sim$y)

conf_matrix_sim
```

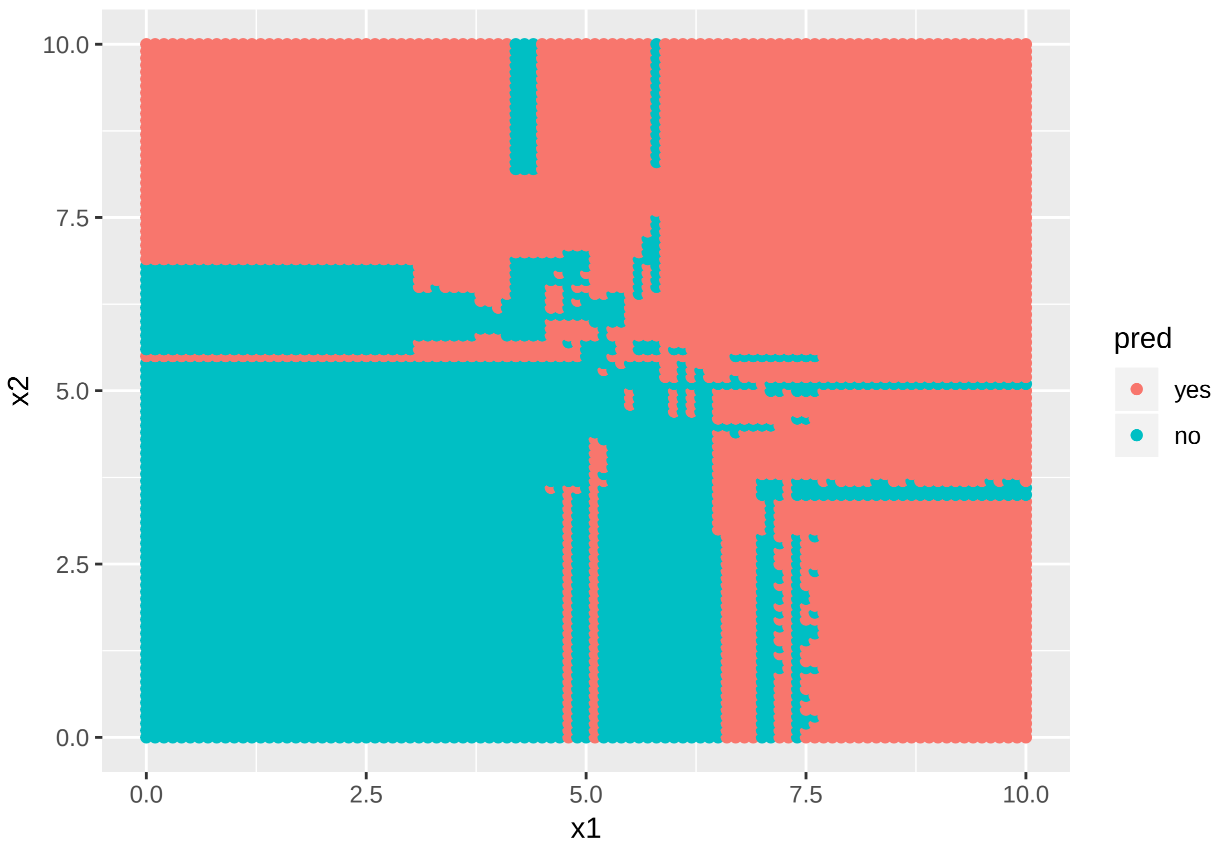
```
##
## pred_test_sim_100 yes no
##               yes  38 12
##               no   20 30
```

5).

```
# create test grid
test_grid_sim <- expand.grid(x1 = seq(0, 10, 0.1), x2 = seq(0, 10, 0.1))

# predict values of test grid with bagged model
test_grid_sim$pred <- predict(bagged_sim_100, newdata = test_grid_sim, type="class")

# plot predictions
pred_plot <- ggplot(test_grid_sim ,aes(x=x1,y=x2,color=pred))+geom_point()
pred_plot
```



6).

It's easy to see that there is a relationship between predictors and labels in the simulated for data in question C. The separations between the two labels include large continuous segments, with only a few overlapping sections near boundaries where the values of the predictors may not allow the labels to easily be identified. In the uniformly distributed and randomly labeled data in question B, there were many small overlapping partitions with no clear boundaries or consistent segments for any values of the predictors. It was clearly just noise.