Lab 08

Question 1:

The probability of a perfect bracket is $(0.5)^{63}$.

Warren Buffet's expected payout is $(0.5)^{63} \cdot 300000000 \cdot 1000000000 = 0.0325$

Question 2:

Load the data sets

```
#Import data
GameStats <-
read.csv('/Data/bonifontea/da350/MM/MRegularSeasonDetailedResults.csv')
TourneyResults <-
read.csv('/Data/bonifontea/da350/MM/MNCAATourneyDetailedResults.csv')
Slots <- read.csv('/Data/bonifontea/da350/MM/MNCAATourneySlots.csv')
Seeds <- read.csv('/Data/bonifontea/da350/MM/MNCAATourneySeeds.csv')
Team <- read.csv('/Data/bonifontea/da350/MM/MTeams.csv')
```

Change the format of the data

```
#Create summary table of average stats per team per year
Winners = GameStats[,c(1,3,9:21)]
colnames(Winners)[2:15] = c("Team","Fgm","Fga","Tpm","Tpa",
                            "Ftm","Fta","OR","DR","Ast",
                            "TO","St","Bl","PF")
Winners$Win = 1

Losers = GameStats[,c(1,5,22:34)]
colnames(Losers)[2:15] = c("Team","Fgm","Fga","Tpm","Tpa",
                           "Ftm","Fta","OR","DR","Ast",
                           "TO","St","Bl","PF")
Losers$Win = 0

FullGame = rbind(Winners,Losers)
rm(Winners,Losers)
```

We will use the difference between each team's average stats for that year as the predictors of each game winner.

```
AnnualSummaries = FullGame %>%
                    group_by(Team,Season) %>%
                    summarise(WinPct = mean(Win),
                              Fgm = mean(Fgm), Fga = mean(Fga),
                              Tpm = mean(Tpm), Tpa = mean(Tpa),
                              Ftm = mean(Ftm), Fta = mean(Fta),
                              OR = mean(OR), DR = mean(DR),
```

```
                                  Ast = mean(Ast), TO = mean(TO),
                                  St = mean(St), Bl = mean(Bl),
                                  PF = mean(PF))
```

We then tie these annual summaries with the tournament outcomes

```
#Construct Training Data merging games played with stats above
TrainingResults = filter(TourneyResults, Season <= 2017)
WinningTraining = as.data.frame(matrix(0, ncol = 31, nrow =
nrow(TrainingResults)))
WinningTraining[,1:3] = TrainingResults[,c(1,3,5)]

for (i in 1:nrow(TrainingResults)){
  WinningTraining[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==TrainingResults[i,"Season"]
                      &AnnualSummaries$Team
==TrainingResults[i,"WTeamID"]),3:16]

  WinningTraining[i,18:31] =
    AnnualSummaries[which(AnnualSummaries$Season==TrainingResults[i,"Season"]
                      &AnnualSummaries$Team
==TrainingResults[i,"LTeamID"]),3:16]
}

colnames(WinningTraining)[1:3] =
  c("Season","Team1","Team2")

colnames(WinningTraining)[4:17] =
  c("T1WinPct","T1Fgm","T1Fga","T1Tpm","T1Tpa","T1Ftm",
    "T1Fta","T1OR","T1DR","T1Ast","T1TO","T1St","T1Bl","T1PF")

colnames(WinningTraining)[18:31] =
  c("T2WinPct","T2Fgm","T2Fga","T2Tpm","T2Tpa","T2Ftm",
    "T2Fta","T2OR","T2DR","T2Ast","T2TO","T2St","T2Bl","T2PF")

WinningTraining$Win = 1

#Now use a similar logic as above, changing indices as necessary, to create
#a dataframe 'LoseTraining' with the losing team of each game as Team 1.

#Fill in Here
LoseTraining = as.data.frame(matrix(0, ncol = 31, nrow =
nrow(TrainingResults)))
LoseTraining[,1:3] = TrainingResults[,c(1,5,3)]

for (i in 1:nrow(TrainingResults)){
  LoseTraining[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==TrainingResults[i,"Season"]
                      &AnnualSummaries$Team
```

```
==TrainingResults[i,"LTeamID"]),3:16]

  LoseTraining[i,18:31] =
    AnnualSummaries[which(AnnualSummaries$Season==TrainingResults[i,"Season"]
                     &AnnualSummaries$Team
==TrainingResults[i,"WTeamID"]),3:16]
}

colnames(LoseTraining)[1:3] =
  c("Season","Team1","Team2")

colnames(LoseTraining)[4:17] =
  c("T1WinPct","T1Fgm","T1Fga","T1Tpm","T1Tpa","T1Ftm",
    "T1Fta","T1OR","T1DR","T1Ast","T1TO","T1St","T1Bl","T1PF")

colnames(LoseTraining)[18:31] =
  c("T2WinPct","T2Fgm","T2Fga","T2Tpm","T2Tpa","T2Ftm",
    "T2Fta","T2OR","T2DR","T2Ast","T2TO","T2St","T2Bl","T2PF")

LoseTraining$Win = 0

Merged = rbind(WinningTraining,LoseTraining)

TrainingData = Merged[,4:17] - Merged[,18:31]
TrainingData[,15] = Merged[,32]

colnames(TrainingData)[1:15] =

c("WinPct","Fgm","Fga","Tpm","Tpa","Ftm","Fta","OR","DR","Ast","TO","St","Bl"
,"PF","Win")
```

Question 3

```
TrainingData$Win <- as.factor(TrainingData$Win)
set.seed(1)
ctrl <- trainControl(method = 'repeatedcv', number = 10, repeats = 5)
logreg.fit <- train(Win ~., data = TrainingData, method = 'glm', family =
'binomial', trControl = ctrl)
logreg.fit$results
```

Description: df [1 × 5]

	parameter <chr>	Accuracy <dbl>	Kappa <dbl>	AccuracySD <dbl>	KappaSD <dbl>
1	none	0.6819741	0.363929	0.02826117	0.05655048

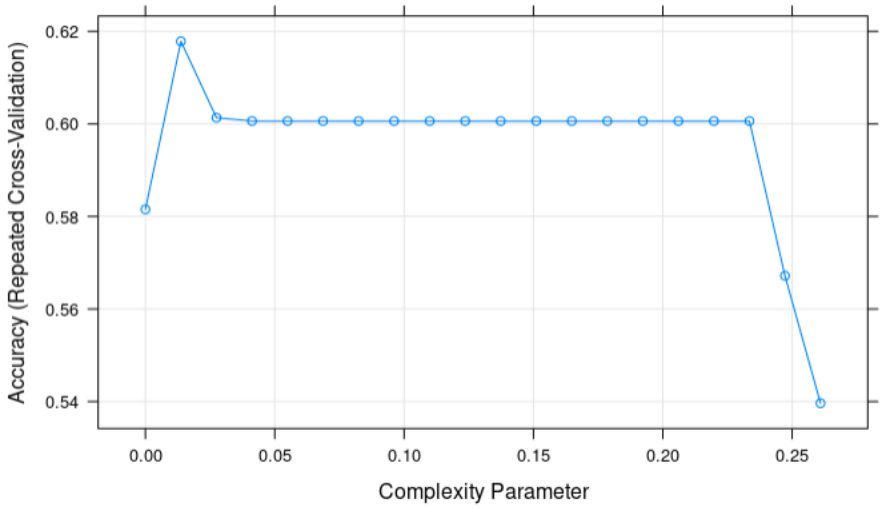
1 row

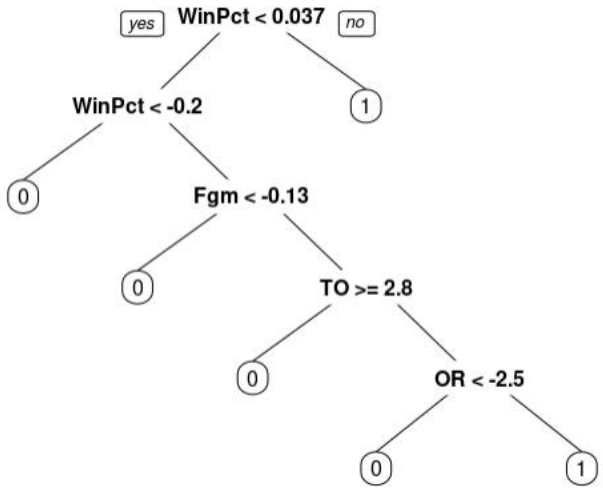
Using repeated 10-fold cross validation, we attain an accuracy score of 0.6819.

Question 4

Build a simple decision tree

```
dtree.fit <- train(Win ~., data = TrainingData, method = 'rpart', trControl =
ctrl, tuneGrid = expand.grid(cp = 0))

start = Sys.time()
dtree.fit_tune <- train(Win ~., data = TrainingData, method = 'rpart',
trControl = ctrl, tuneLength = 20)
end = Sys.time()
end-start

plot(dtree.fit_tune)
dtree.fit_tune$bestTune

dtree.fit_pruning <- train(Win ~., data = TrainingData, method = 'rpart',
trControl = ctrl, tuneGrid = expand.grid(cp = 0.01373464))

prp(dtree.fit_pruning$finalModel)
dtree.fit_pruning$results
```





Using accuracy is acceptable here because of two reasons. First, there is no class imbalance here. We have the same number of observations for both classes thanks to the way we transform our data. If there is class imbalance, the model may tend to predict one class over another so we may want to change our threshold to get more balanced predictions. However, if there is no class imbalance, the model has the probability to predict both classes equally. Second, in some other situations, false positive may be worse than false negative or vice versa so we want to choose a different threshold (cutoff point) that maximize our benefit. However, in this situation, false positive and false negative or equally bad so there is no need to change the threshold.

When pruning the decision tree with cp = 0.01373, we have a model with an accuracy score of 0.613 and a Kappa score of 0.226 which is not very good.

The plot of the pruned decision tree tells me that WinPct, Fgm, TO and OR are important predictors. If WinPct >= 0.037, the decision tree predicts a 1 without the need to consider any other variables. If WinPct < -0.2, it predicts a 0. Following the branches, we have if -0.2 <= WinPct < 0.037 then it predicts a 0 if Fgm < -0.13 and 1 otherwise. It then considers TO and OR. Overall, the decision tree only considers 4 variables. This is not very good because there are 14 other variables that may be useful and considering only 4 variables may lead to overfitting. We may also need to see if these 4 variables are correlated. The multicollinearity worsens the high variance problem.

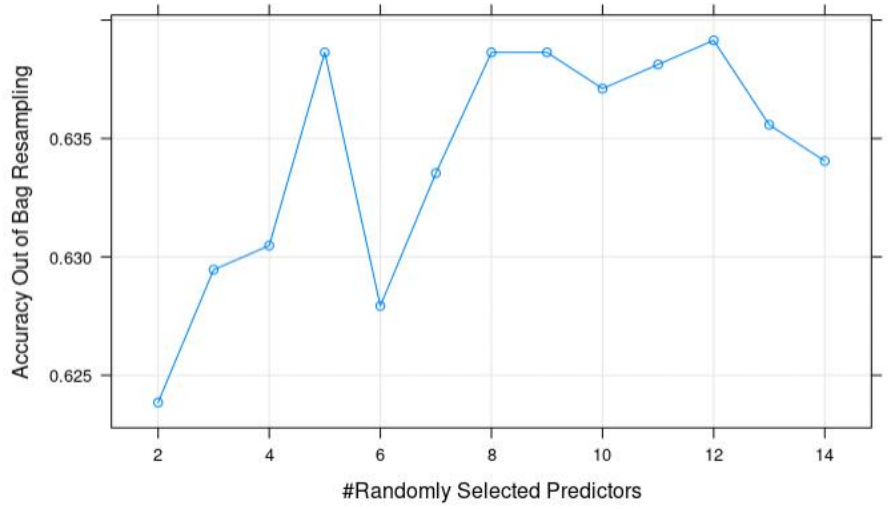
Question 5

Train a random forest model

```
ctrl = trainControl(method = 'oob')
start = Sys.time()
rfr.fit_tune = train(Win ~ ., data = TrainingData, method = 'rf',trControl =
ctrl, tuneLength = 13, importance = TRUE)
end = Sys.time()
end-start

plot(rfr.fit_tune$results$mtry, rfr.fit_tune$results$Accuracy)
rfr.fit_tune$bestTune

rfr.fit <- train(Win ~ ., data = TrainingData, method = 'rf',trControl =
ctrl, tuneGrid = expand.grid(mtry = 7), importance = TRUE)

rfr.fit_tune2 = train(Win ~ ., data = TrainingData, method = 'rf',trControl =
ctrl, tuneLength = 13, importance = TRUE)
plot(rfr.fit_tune2$results$mtry, rfr.fit_tune2$results$Accuracy)
rfr.fit_tune2$bestTune
plot(rfr.fit_tune2)

rfr.fit_tune3 = train(Win ~ ., data = TrainingData, method = 'rf',trControl =
ctrl, tuneLength = 13, importance = TRUE)
plot(rfr.fit_tune3$results$mtry, rfr.fit_tune3$results$Accuracy)
rfr.fit_tune3$bestTune
```

We keep getting different mtry for the parameters of best fit. Therefore, we increase ntree to see if mtry converges.

```
rfr.fit_tune4 = train(Win ~ ., data = TrainingData, method = 'rf',trControl =
ctrl, tuneLength = 13, importance = TRUE, ntree = 1250)
plot(rfr.fit_tune4)
rfr.fit_tune4$bestTune$mtry
```



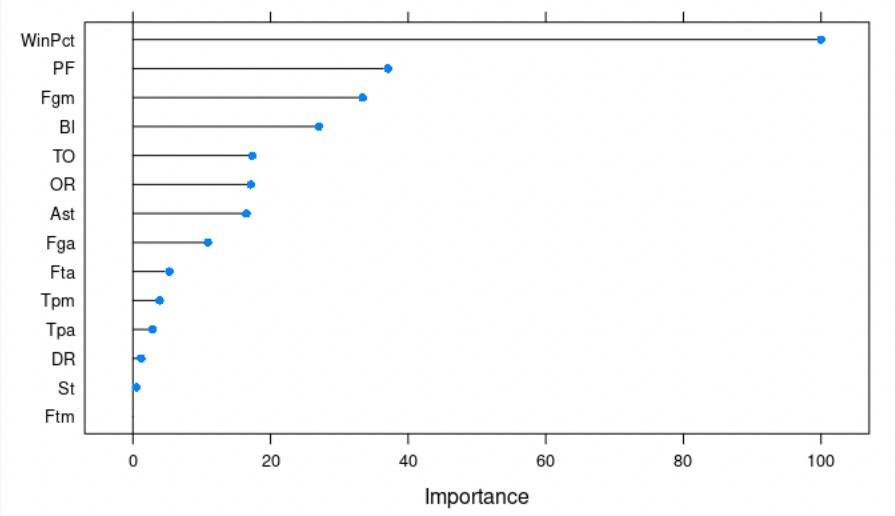
We finally decide on ntree = 1250 and mtry = 12 as it doesn't worth tuning both ntree and mtry at the same time when the improvement is just in the hundredths place.

```
rfr.fit2 = train(Win ~ ., data = TrainingData, method = 'rf',trControl =
ctrl, tuneGrid = expand.grid(mtry = rfr.fit_tune4$bestTune$mtry), importance
= TRUE, ntree = 1250)
rfr.fit2$results
```

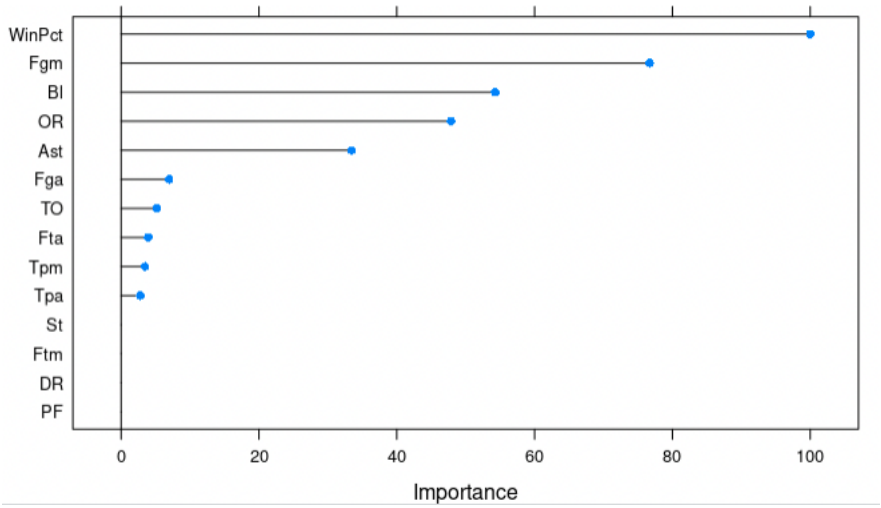
	Accuracy <dbl>	Kappa <dbl>	mtry <dbl>
1	0.6391437	0.2782875	12

The 'oob' estimate of accuracy is 0.6391. It's quite better than the model using decision tree.

```
rf_imp = varImp(rfr.fit2)
plot(rf_imp)
```




```
dtree_imp = varImp(dtree.fit_pruning)
plot(dtree_imp)
```



WinPct appears to be the most important predictor with a score of 100, while Ftm appears to be the least important predictor with a score of 0. In the single decision tree, WinPct is also the most important predictor with a score of 100, and Fgm is among the three least important predictors with a score of 0. There are some other slight changes in the order of importance but in general, the single decision tree is quite consistent with the random forest. The biggest difference occurs in PF. PF is the second most important predictor in the random forest, but it is among the three least important in the single decision tree.

Question 6

```
PredictWinners = function(thisModel,Year){

mycolnames =
c("Season","Team1","Team2","WinPct","Fgm","Fga","Tpm","Tpa","Ftm","Fta","OR",
"DR","Ast","TO","St","Bl","PF")
```

```
TheseSlots = filter(Slots,Season==Year)
TheseSeeds = filter(Seeds,Season==Year)

TheseSlots$Prediction = 0 #Initiate to store predictions

#Round 1
Round1Games = as.data.frame(matrix(0, ncol = 17, nrow = 32))
colnames(Round1Games) = mycolnames
Round1Games$Season = Year

for (i in 1:32){
  Round1Games[i,"Team1"] =
    TheseSeeds[which(TheseSeeds$Seed ==
as.character(TheseSlots$StrongSeed[i])),3]

  Round1Games[i,"Team2"] =
    TheseSeeds[which(TheseSeeds$Seed ==
as.character(TheseSlots$WeakSeed[i])),3]

  Round1Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round1Games[i,"Season"]
                          & AnnualSummaries$Team
==Round1Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round1Games[i,"Season"]
                          & AnnualSummaries$Team
==Round1Games[i,"Team2"]),3:16]
}

#Create predictions on round 1
pred = predict(thisModel, Round1Games)
Round1Pred = data.frame(Slot = Slots[1:32,"Slot"],PredictedWinner = 0)
for (i in 1:32){
  if (pred[i] == 1){
    Round1Pred[i,"PredictedWinner"] = Round1Games[i,"Team1"]
  }
  else{
    Round1Pred[i,"PredictedWinner"] = Round1Games[i,"Team2"]
  }
}
TheseSlots$Prediction[1:32] = Round1Pred[,"PredictedWinner"]



## Round 2
#Use the predicted classes to construct round 2
Round2Games = as.data.frame(matrix(0, ncol = 17, nrow = 16))
colnames(Round2Games) = mycolnames
Round2Games$Season = Year
```

```
for (i in 1:16){
  Round2Games[i,"Team1"] =

Round1Pred[which(Round1Pred$Slot==as.character(TheseSlots$StrongSeed[i+32])),
"PredictedWinner"]

  Round2Games[i,"Team2"] =
    Round1Pred[which(Round1Pred$Slot ==
as.character(TheseSlots$WeakSeed[i+32])),"PredictedWinner"]

  Round2Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round2Games[i,"Season"]
                          & AnnualSummaries$Team
==Round2Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round2Games[i,"Season"]
                         & AnnualSummaries$Team ==Round2Games[i,"Team2"]),3:16]
}

#Create predictions on round 2
pred = predict(thisModel, Round2Games)
Round2Pred = data.frame(Slot = Slots[33:48,"Slot"],PredictedWinner = 0)
for (i in 1:16){
  if (pred[i] == 1){
    Round2Pred[i,"PredictedWinner"] = Round2Games[i,"Team1"]
  }
  else{
    Round2Pred[i,"PredictedWinner"] = Round2Games[i,"Team2"]
  }
}
TheseSlots$Prediction[33:48] = Round2Pred[,"PredictedWinner"]



## Round 3
Round3Games = as.data.frame(matrix(0, ncol = 17, nrow = 8))
colnames(Round3Games) = mycolnames
Round3Games$Season = Year

for (i in 1:8){
  Round3Games[i,"Team1"] =

Round2Pred[which(Round2Pred$Slot==as.character(TheseSlots$StrongSeed[i+48])),
"PredictedWinner"]

  Round3Games[i,"Team2"] =

Round2Pred[which(Round2Pred$Slot==as.character(TheseSlots$WeakSeed[i+48])),"P
redictedWinner"]
```

```
  Round3Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round3Games[i,"Season"]
                          & AnnualSummaries$Team
==Round3Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round3Games[i,"Season"]
                          & AnnualSummaries$Team
==Round3Games[i,"Team2"]),3:16]
}

#Create predictions on round 3
pred = predict(thisModel, Round3Games)
Round3Pred = data.frame(Slot = Slots[49:56,"Slot"],PredictedWinner = 0)
for (i in 1:8){
  if (pred[i] == 1){
    Round3Pred[i,"PredictedWinner"] = Round3Games[i,"Team1"]
  }
  else{
    Round3Pred[i,"PredictedWinner"] = Round3Games[i,"Team2"]
  }
}
TheseSlots$Prediction[49:56] = Round3Pred[,"PredictedWinner"]


## Round 4
Round4Games = as.data.frame(matrix(0, ncol = 17, nrow = 4))
colnames(Round4Games) = mycolnames
Round4Games$Season = Year

for (i in 1:4){
  Round4Games[i,"Team1"] =

Round3Pred[which(Round3Pred$Slot==as.character(TheseSlots$StrongSeed[i+56])),
"PredictedWinner"]

  Round4Games[i,"Team2"] =

Round3Pred[which(Round3Pred$Slot==as.character(TheseSlots$WeakSeed[i+56])),"P
redictedWinner"]

  Round4Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
                          & AnnualSummaries$Team
==Round4Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
                          & AnnualSummaries$Team
==Round4Games[i,"Team2"]),3:16]
}
```

```
#Create predictions on round 4
pred = predict(thisModel, Round4Games)
Round4Pred = data.frame(Slot = Slots[57:60,"Slot"],PredictedWinner = 0)
for (i in 1:4){
  if (pred[i] == 1){
    Round4Pred[i,"PredictedWinner"] = Round4Games[i,"Team1"]
  }
  else{
    Round4Pred[i,"PredictedWinner"] = Round4Games[i,"Team2"]
  }
}
TheseSlots$Prediction[57:60] = Round4Pred[,"PredictedWinner"]


## Round 5
Round5Games = as.data.frame(matrix(0, ncol = 17, nrow = 2))
colnames(Round5Games) = mycolnames
Round5Games$Season = Year

for (i in 1:2){
  Round5Games[i,"Team1"] =

Round4Pred[which(Round4Pred$Slot==as.character(TheseSlots$StrongSeed[i+60])),
"PredictedWinner"]

  Round5Games[i,"Team2"] =

Round4Pred[which(Round4Pred$Slot==as.character(TheseSlots$WeakSeed[i+60])),"P
redictedWinner"]

  Round5Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
                          & AnnualSummaries$Team
==Round4Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
                          & AnnualSummaries$Team
==Round4Games[i,"Team2"]),3:16]
}

#Create predictions on round 5
pred = predict(thisModel, Round5Games)
Round5Pred = data.frame(Slot = Slots[61:62,"Slot"],PredictedWinner = 0)
for (i in 1:2){
  if (pred[i] == 1){
    Round5Pred[i,"PredictedWinner"] = Round5Games[i,"Team1"]
  }
  else{
    Round5Pred[i,"PredictedWinner"] = Round5Games[i,"Team2"]
  }
```

```
}
TheseSlots$Prediction[61:62] = Round5Pred[,"PredictedWinner"]


## Round 6
Round6Games = as.data.frame(matrix(0, ncol = 17, nrow = 1))
colnames(Round6Games) = mycolnames
Round6Games$Season = Year

for (i in 1:1){
  Round6Games[i,"Team1"] =

Round5Pred[which(Round5Pred$Slot==as.character(TheseSlots$StrongSeed[i+62])),
"PredictedWinner"]

  Round6Games[i,"Team2"] =

Round5Pred[which(Round5Pred$Slot==as.character(TheseSlots$WeakSeed[i+62])),"P
redictedWinner"]

  Round6Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round6Games[i,"Season"]
                          & AnnualSummaries$Team
==Round6Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round6Games[i,"Season"]
                          & AnnualSummaries$Team
==Round6Games[i,"Team2"]),3:16]
}

#Create predictions on round 6
pred = predict(thisModel, Round6Games)
Round6Pred = data.frame(Slot = Slots[63,"Slot"],PredictedWinner = 0)
for (i in 1:1){
  if (pred[i] == 1){
    Round6Pred[i,"PredictedWinner"] = Round6Games[i,"Team1"]
  }
  else{
    Round6Pred[i,"PredictedWinner"] = Round6Games[i,"Team2"]
  }
}
TheseSlots$Prediction[63] = Round6Pred[,"PredictedWinner"]

TheseResults = filter(TourneyResults, Season==Year)
 TheseSlots$Actual = 0

for (i in 1:63){
  TheseSlots[i,"Actual"] =
TheseResults[which(as.character(TheseSlots[i,"Slot"])==TheseResults$Slot),"WT
eamID"]
```

```
}

Rounds = 0
for (i in 1:32){
  Rounds[i] = 1
}
for (i in 33:48){
  Rounds[i] = 2
}
for (i in 49:56){
  Rounds[i] = 3
}
for (i in 57:60){
  Rounds[i] = 4
}
for (i in 61:62){
  Rounds[i] = 5
}
for (i in 63){
  Rounds[i] = 6
}

Results = data.frame(Round = Rounds, Predicted = TheseSlots$Prediction,
Winner = TheseSlots$Actual)

return(Results)
}

PredictWinners(rfr.fit,2019)
```

For each round, the code first tries to get the information of which team compete against which team using the `Seeds` and `Slots` data sets. It then get all the annual summaries of the team performance. Next, it creates predictions and saves the predicted values. After having done with predicting, it tries to get the actual winners from the `TourneyResults` data set. Finally, it creates a data frame of the results of all rounds with both the predicted and actual values. It takes so many lines of code instead of just one line using the predict function because there are 6 rounds and the data for the next round depend on the predictions of the previous round. Therefore, there are many predictions that need to be made instead of just one prediction which leads to many lines of code.

Question 7

```
report_accuracy <- function(model, year) {
  predict_df = PredictWinners(model, year)
  right = predict_df[predict_df$Predicted == predict_df$Winner,]
  accuracy = nrow(right)/nrow(predict_df)
  return (accuracy);
}

report_accuracy(logreg.fit,2018)
```

```
report_accuracy(logreg.fit,2019)
report_accuracy(logreg.fit,2021)

report_accuracy(dtree.fit_pruning,2018)
report_accuracy(dtree.fit_pruning,2019)
report_accuracy(dtree.fit_pruning,2021)

report_accuracy(rfr.fit2,2018)
report_accuracy(rfr.fit2,2019)
report_accuracy(rfr.fit2,2021)
```

```
[1] 0.5396825
[1] 0.5555556
[1] 0.4603175
[1] 0.4761905
[1] 0.5396825
[1] 0.4285714
[1] 0.6190476
[1] 0.5555556
[1] 0.4126984
```

Question 8

The expectation of how many points would be made in a bracket that flips a fair coin to make each prediction:

Expected points for round 1: $0.5 * 1 * 32 = 16$

Expected points for round 2: $0.5 * 0.5 * 2 * 16 = 8$

Expected points for round 3: $0.5 * 0.5 * 0.5 * 4 * 8 = 4$

Expected points for round 4: $0.5 * 0.5 * 0.5 * 0.5 * 8 * 4 = 2$

Expected points for round 5: $0.5 * 0.5 * 0.5 * 0.5 * 0.5 * 16 * 2 = 1$

Expected points for round 6: $ 0.5*0.5*0.5*0.5*0.5*32*1 = 0.5$

=> Expected points: $16 + 8 + 4 + 2 + 1 + 0.5 = 31.5$

Question 9

```
point_per_round <- function(num, df) {
  round <- df[df$Round == num,]
  point <- nrow(round[round$Predicted == round$Winner,])*2^(num-1)
  return (point)
}

calculate_point <- function(model, year) {
  predict_df = PredictWinners(model, year)
```

```
  points = sapply(1:6, function (x) point_per_round(x, predict_df))
  return (sum(points))
}

dt = data.frame(Year = c(2018, 2019, 2021), Expected = c(31.5,31.5,31.5),
Log_Reg = c(calculate_point(logreg.fit,2018),
calculate_point(logreg.fit,2019), calculate_point(logreg.fit,2021)), Dtree =
c(calculate_point(dtree.fit_pruning,2018),
calculate_point(dtree.fit_pruning,2019),
calculate_point(dtree.fit_pruning,2021)), Rfr =
c(calculate_point(rfr.fit2,2018), calculate_point(rfr.fit2,2019),
calculate_point(rfr.fit2,2021)))

kable(dt)
```

Year	Expected	Log_Reg	Dtree	Rfr
2018	31.5	72	94	95
2019	31.5	62	73	54
2021	31.5	68	59	62

The method appears to be working the best is the single decision tree with an average of 75.3. The random forest did worse than the average bracket score for the 2019 bracket with only 54. The random forest did a fantastic job for the 2018 data with 95. In general, the predictions of the function get higher points than the points made by just randomly guessing (31.5).

Question 10

Under normal circumstances, it takes the majority of all classes to decide on which class the testing observation fits into. I believe for the random forest, it generates probabilities based on how many times the observations are predicted as a specific class. For example, if we generate 500 single decision trees and get Team 1 to win using 400 decision trees. Then the probability that Team 1 wins will be $\frac{400}{500} = 0.8$.

```
PredictWinners_modified = function(thisModel,Year){

mycolnames =
c("Season","Team1","Team2","WinPct","Fgm","Fga","Tpm","Tpa","Ftm","Fta","OR",
"DR","Ast","TO","St","Bl","PF")

TheseSlots = filter(Slots,Season==Year)
TheseSeeds = filter(Seeds,Season==Year)

TheseSlots$Prediction = 0 #Initiate to store predictions

#function to generate predictions based on the predicted probability
```

```
generate_pred <- function(predict_prob) {
  colnames(predict_prob) <- c('Team 2','Team 1')
  predict_prob <- predict_prob %>%
  mutate(predicted = ifelse(runif(1)<`Team 1`,1,0))
  return (predict_prob$predicted)
}

#Round 1
Round1Games = as.data.frame(matrix(0, ncol = 17, nrow = 32))
colnames(Round1Games) = mycolnames
Round1Games$Season = Year

for (i in 1:32){
  Round1Games[i,"Team1"] =
    TheseSeeds[which(TheseSeeds$Seed ==
as.character(TheseSlots$StrongSeed[i])),3]

  Round1Games[i,"Team2"] =
    TheseSeeds[which(TheseSeeds$Seed ==
as.character(TheseSlots$WeakSeed[i])),3]

  Round1Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round1Games[i,"Season"]
                          & AnnualSummaries$Team
==Round1Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round1Games[i,"Season"]
                          & AnnualSummaries$Team
==Round1Games[i,"Team2"]),3:16]
}

#Create predictions on round 1
pred = predict(thisModel, Round1Games, type = 'prob')
pred = generate_pred(pred)

Round1Pred = data.frame(Slot = Slots[1:32,"Slot"],PredictedWinner = 0)
for (i in 1:32){
  #random = runif(1)
  #if random < pred[1]
  if (pred[i] == 1){
    Round1Pred[i,"PredictedWinner"] = Round1Games[i,"Team1"]
  }
  else{
    Round1Pred[i,"PredictedWinner"] = Round1Games[i,"Team2"]
  }
}
TheseSlots$Prediction[1:32] = Round1Pred[,"PredictedWinner"]

## Round 2
#Use the predicted classes to construct round 2
```

```
Round2Games = as.data.frame(matrix(0, ncol = 17, nrow = 16))
colnames(Round2Games) = mycolnames
Round2Games$Season = Year

for (i in 1:16){
  Round2Games[i,"Team1"] =

Round1Pred[which(Round1Pred$Slot==as.character(TheseSlots$StrongSeed[i+32])),
"PredictedWinner"]

  Round2Games[i,"Team2"] =
    Round1Pred[which(Round1Pred$Slot ==
as.character(TheseSlots$WeakSeed[i+32])),"PredictedWinner"]

  Round2Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round2Games[i,"Season"]
                          & AnnualSummaries$Team
==Round2Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round2Games[i,"Season"]
                         & AnnualSummaries$Team ==Round2Games[i,"Team2"]),3:16]
}

#Create predictions on round 2
pred = predict(thisModel, Round2Games, type = 'prob')
pred = generate_pred(pred)
Round2Pred = data.frame(Slot = Slots[33:48,"Slot"],PredictedWinner = 0)
for (i in 1:16){
  if (pred[i] == 1){
    Round2Pred[i,"PredictedWinner"] = Round2Games[i,"Team1"]
  }
  else{
    Round2Pred[i,"PredictedWinner"] = Round2Games[i,"Team2"]
  }
}
TheseSlots$Prediction[33:48] = Round2Pred[,"PredictedWinner"]

## Round 3
Round3Games = as.data.frame(matrix(0, ncol = 17, nrow = 8))
colnames(Round3Games) = mycolnames
Round3Games$Season = Year

for (i in 1:8){
  Round3Games[i,"Team1"] =

Round2Pred[which(Round2Pred$Slot==as.character(TheseSlots$StrongSeed[i+48])),
"PredictedWinner"]

  Round3Games[i,"Team2"] =
```

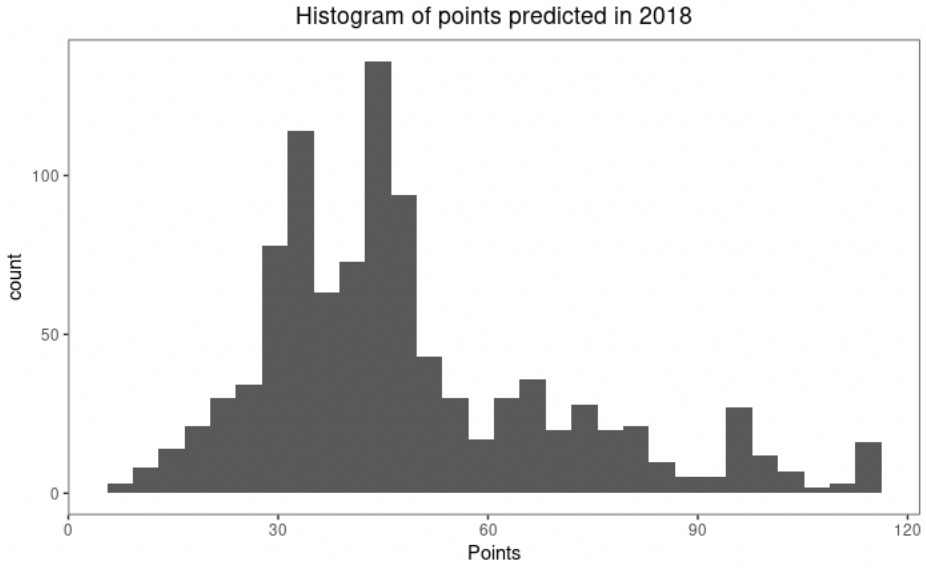
```
Round2Pred[which(Round2Pred$Slot==as.character(TheseSlots$WeakSeed[i+48])),"P
redictedWinner"]

  Round3Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round3Games[i,"Season"]
                          & AnnualSummaries$Team
==Round3Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round3Games[i,"Season"]
                          & AnnualSummaries$Team
==Round3Games[i,"Team2"]),3:16]
}

#Create predictions on round 3
pred = predict(thisModel, Round3Games, type = 'prob')
pred = generate_pred(pred)

Round3Pred = data.frame(Slot = Slots[49:56,"Slot"],PredictedWinner = 0)
for (i in 1:8){
  if (pred[i] == 1){
    Round3Pred[i,"PredictedWinner"] = Round3Games[i,"Team1"]
  }
  else{
    Round3Pred[i,"PredictedWinner"] = Round3Games[i,"Team2"]
  }
}
TheseSlots$Prediction[49:56] = Round3Pred[,"PredictedWinner"]


## Round 4
Round4Games = as.data.frame(matrix(0, ncol = 17, nrow = 4))
colnames(Round4Games) = mycolnames
Round4Games$Season = Year

for (i in 1:4){
  Round4Games[i,"Team1"] =

Round3Pred[which(Round3Pred$Slot==as.character(TheseSlots$StrongSeed[i+56])),
"PredictedWinner"]

  Round4Games[i,"Team2"] =

Round3Pred[which(Round3Pred$Slot==as.character(TheseSlots$WeakSeed[i+56])),"P
redictedWinner"]

  Round4Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
                          & AnnualSummaries$Team
==Round4Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
```

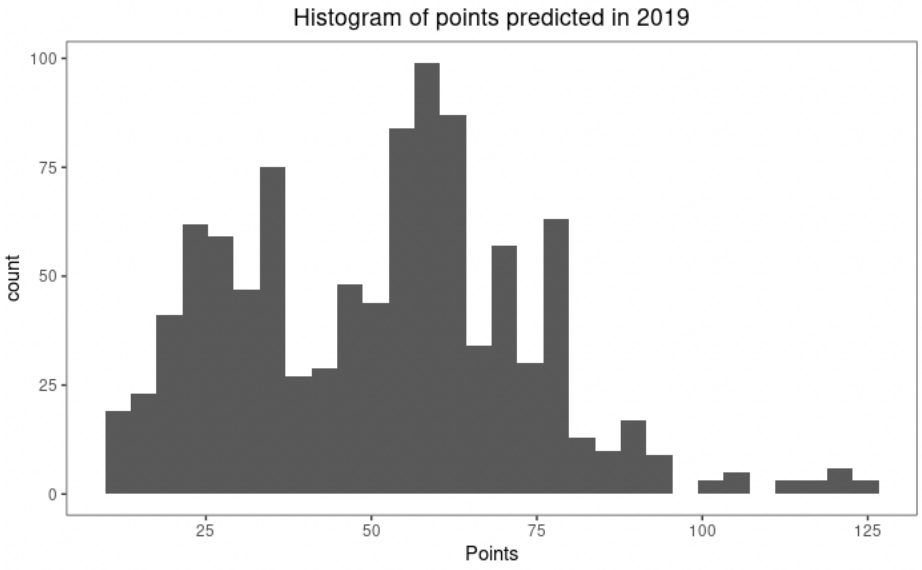
```
                        & AnnualSummaries$Team
==Round4Games[i,"Team2"]),3:16]
}

#Create predictions on round 4
pred = predict(thisModel, Round4Games, type = 'prob')
pred = generate_pred(pred)

Round4Pred = data.frame(Slot = Slots[57:60,"Slot"],PredictedWinner = 0)
for (i in 1:4){
  if (pred[i] == 1){
    Round4Pred[i,"PredictedWinner"] = Round4Games[i,"Team1"]
  }
  else{
    Round4Pred[i,"PredictedWinner"] = Round4Games[i,"Team2"]
  }
}
TheseSlots$Prediction[57:60] = Round4Pred[,"PredictedWinner"]


## Round 5
Round5Games = as.data.frame(matrix(0, ncol = 17, nrow = 2))
colnames(Round5Games) = mycolnames
Round5Games$Season = Year

for (i in 1:2){
  Round5Games[i,"Team1"] =

Round4Pred[which(Round4Pred$Slot==as.character(TheseSlots$StrongSeed[i+60])),
"PredictedWinner"]

  Round5Games[i,"Team2"] =

Round4Pred[which(Round4Pred$Slot==as.character(TheseSlots$WeakSeed[i+60])),"P
redictedWinner"]

  Round5Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
                        & AnnualSummaries$Team
==Round4Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
                        & AnnualSummaries$Team
==Round4Games[i,"Team2"]),3:16]
}

#Create predictions on round 5
pred = predict(thisModel, Round5Games, type = 'prob')
pred = generate_pred(pred)
```

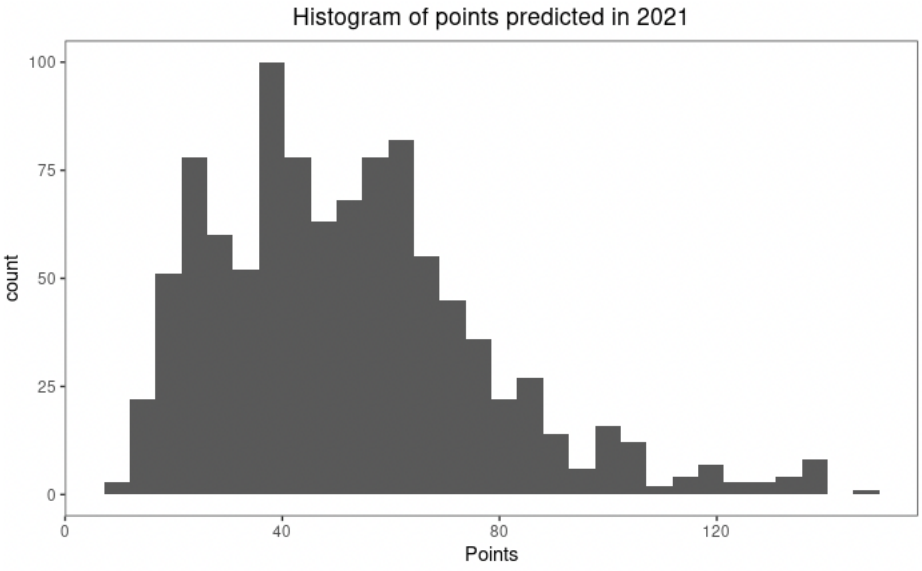
```
Round5Pred = data.frame(Slot = Slots[61:62,"Slot"],PredictedWinner = 0)
for (i in 1:2){
  if (pred[i] == 1){
    Round5Pred[i,"PredictedWinner"] = Round5Games[i,"Team1"]
  }
  else{
    Round5Pred[i,"PredictedWinner"] = Round5Games[i,"Team2"]
  }
}
TheseSlots$Prediction[61:62] = Round5Pred[,"PredictedWinner"]


## Round 6
Round6Games = as.data.frame(matrix(0, ncol = 17, nrow = 1))
colnames(Round6Games) = mycolnames
Round6Games$Season = Year

for (i in 1:1){
  Round6Games[i,"Team1"] =

Round5Pred[which(Round5Pred$Slot==as.character(TheseSlots$StrongSeed[i+62])),
"PredictedWinner"]

  Round6Games[i,"Team2"] =

Round5Pred[which(Round5Pred$Slot==as.character(TheseSlots$WeakSeed[i+62])),"P
redictedWinner"]

  Round6Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round6Games[i,"Season"]
                          & AnnualSummaries$Team
==Round6Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round6Games[i,"Season"]
                          & AnnualSummaries$Team
==Round6Games[i,"Team2"]),3:16]
}

#Create predictions on round 6
pred = predict(thisModel, Round6Games, type = 'prob')
pred = generate_pred(pred)

Round6Pred = data.frame(Slot = Slots[63,"Slot"],PredictedWinner = 0)
for (i in 1:1){
  if (pred[i] == 1){
    Round6Pred[i,"PredictedWinner"] = Round6Games[i,"Team1"]
  }
  else{
    Round6Pred[i,"PredictedWinner"] = Round6Games[i,"Team2"]
  }
```

```
}
TheseSlots$Prediction[63] = Round6Pred[,"PredictedWinner"]

TheseResults = filter(TourneyResults, Season==Year)
 TheseSlots$Actual = 0

for (i in 1:63){
  TheseSlots[i,"Actual"] =
TheseResults[which(as.character(TheseSlots[i,"Slot"])==TheseResults$Slot),"WT
eamID"]
}

Rounds = 0
for (i in 1:32){
  Rounds[i] = 1
}
for (i in 33:48){
  Rounds[i] = 2
}
for (i in 49:56){
  Rounds[i] = 3
}
for (i in 57:60){
  Rounds[i] = 4
}
for (i in 61:62){
  Rounds[i] = 5
}
for (i in 63){
  Rounds[i] = 6
}

Results = data.frame(Round = Rounds, Predicted = TheseSlots$Prediction,
Winner = TheseSlots$Actual)

return(Results)
}

PredictWinners_modified(rfr.fit2,2019)

calculate_point_modified <- function(model, year) {
  predict_df = PredictWinners_modified(model, year)
  points = sapply(1:6, function (x) point_per_round(x, predict_df))
  return (sum(points))
}

rfr_pred2018 <- lapply(1:1000, function(x)
calculate_point_modified(rfr.fit2,2018))
rfr_pred2019 <- lapply(1:1000, function(x)
```

```
calculate_point_modified(rfr.fit2,2019))
rfr_pred2021 <- lapply(1:1000, function(x)
calculate_point_modified(rfr.fit2,2021))

rfr_pred2018 <- data.frame(Points = c(unlist(rfr_pred2018)))
rfr_pred2019 <- data.frame(Points = c(unlist(rfr_pred2019)))
rfr_pred2021 <- data.frame(Points = c(unlist(rfr_pred2021)))

ggplot(aes(Points),data = rfr_pred2018) +
  geom_histogram() +
  labs(title = 'Histogram of points predicted in 2018') +
  theme_bw() +
  theme_test() +
  theme(plot.title = element_text(hjust = 0.5))

ggplot(aes(Points),data = rfr_pred2019) +
  geom_histogram() +
  labs(title = 'Histogram of points predicted in 2019') +
  theme_bw() +
  theme_test() +
  theme(plot.title = element_text(hjust = 0.5))

ggplot(aes(Points),data = rfr_pred2021) +
  geom_histogram() +
  labs(title = 'Histogram of points predicted in 2021') +
  theme_bw() +
  theme_test() +
  theme(plot.title = element_text(hjust = 0.5))
```







On average, the points seem to get worse. In 2018, the most frequent point is under 60, lower than all the points using the deterministic procedure. In 2019, the most frequent point is at around 58. This is better than the deterministic random forest but still worse than other methods. In 2021, the most frequent point is at around 40, which is worse than all the points using the deterministic procedure. Overall, this procedure generates worse results compared to the deterministic procedure.

Question 11

```
PredictWinners2022 = function(thisModel){

mycolnames =
c("Season","Team1","Team2","WinPct","Fgm","Fga","Tpm","Tpa","Ftm","Fta","OR",
"DR","Ast","TO","St","Bl","PF")
```

```
TheseSlots = filter(Slots,Season==2022)
TheseSeeds = filter(Seeds,Season==2022)

TheseSlots$Prediction = 0 #Initiate to store predictions

#Round 1
Round1Games = as.data.frame(matrix(0, ncol = 17, nrow = 32))
colnames(Round1Games) = mycolnames
Round1Games$Season = 2022

for (i in 1:32){
  Round1Games[i,"Team1"] =
    TheseSeeds[which(TheseSeeds$Seed ==
as.character(TheseSlots$StrongSeed[i])),3]

  Round1Games[i,"Team2"] =
    TheseSeeds[which(TheseSeeds$Seed ==
as.character(TheseSlots$WeakSeed[i])),3]

  Round1Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round1Games[i,"Season"]
                          & AnnualSummaries$Team
==Round1Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round1Games[i,"Season"]
                          & AnnualSummaries$Team
==Round1Games[i,"Team2"]),3:16]
}

#Create predictions on round 1
pred = predict(thisModel, Round1Games)
Round1Pred = data.frame(Slot = Slots[1:32,"Slot"],PredictedWinner = 0)
for (i in 1:32){
  if (pred[i] == 1){
    Round1Pred[i,"PredictedWinner"] = Round1Games[i,"Team1"]
  }
  else{
    Round1Pred[i,"PredictedWinner"] = Round1Games[i,"Team2"]
  }
}
TheseSlots$Prediction[1:32] = Round1Pred[,"PredictedWinner"]



## Round 2
#Use the predicted classes to construct round 2
Round2Games = as.data.frame(matrix(0, ncol = 17, nrow = 16))
colnames(Round2Games) = mycolnames
Round2Games$Season = 2022
```

```
for (i in 1:16){
  Round2Games[i,"Team1"] =

Round1Pred[which(Round1Pred$Slot==as.character(TheseSlots$StrongSeed[i+32])),
"PredictedWinner"]

  Round2Games[i,"Team2"] =
    Round1Pred[which(Round1Pred$Slot ==
as.character(TheseSlots$WeakSeed[i+32])),"PredictedWinner"]

  Round2Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round2Games[i,"Season"]
                          & AnnualSummaries$Team
==Round2Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round2Games[i,"Season"]
                         & AnnualSummaries$Team ==Round2Games[i,"Team2"]),3:16]
}

#Create predictions on round 2
pred = predict(thisModel, Round2Games)
Round2Pred = data.frame(Slot = Slots[33:48,"Slot"],PredictedWinner = 0)
for (i in 1:16){
  if (pred[i] == 1){
    Round2Pred[i,"PredictedWinner"] = Round2Games[i,"Team1"]
  }
  else{
    Round2Pred[i,"PredictedWinner"] = Round2Games[i,"Team2"]
  }
}
TheseSlots$Prediction[33:48] = Round2Pred[,"PredictedWinner"]



## Round 3
Round3Games = as.data.frame(matrix(0, ncol = 17, nrow = 8))
colnames(Round3Games) = mycolnames
Round3Games$Season = 2022

for (i in 1:8){
  Round3Games[i,"Team1"] =

Round2Pred[which(Round2Pred$Slot==as.character(TheseSlots$StrongSeed[i+48])),
"PredictedWinner"]

  Round3Games[i,"Team2"] =

Round2Pred[which(Round2Pred$Slot==as.character(TheseSlots$WeakSeed[i+48])),"P
redictedWinner"]
```

```
  Round3Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round3Games[i,"Season"]
                          & AnnualSummaries$Team
==Round3Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round3Games[i,"Season"]
                          & AnnualSummaries$Team
==Round3Games[i,"Team2"]),3:16]
}

#Create predictions on round 3
pred = predict(thisModel, Round3Games)
Round3Pred = data.frame(Slot = Slots[49:56,"Slot"],PredictedWinner = 0)
for (i in 1:8){
  if (pred[i] == 1){
    Round3Pred[i,"PredictedWinner"] = Round3Games[i,"Team1"]
  }
  else{
    Round3Pred[i,"PredictedWinner"] = Round3Games[i,"Team2"]
  }
}
TheseSlots$Prediction[49:56] = Round3Pred[,"PredictedWinner"]


## Round 4
Round4Games = as.data.frame(matrix(0, ncol = 17, nrow = 4))
colnames(Round4Games) = mycolnames
Round4Games$Season = 2022

for (i in 1:4){
  Round4Games[i,"Team1"] =

Round3Pred[which(Round3Pred$Slot==as.character(TheseSlots$StrongSeed[i+56])),
"PredictedWinner"]

  Round4Games[i,"Team2"] =

Round3Pred[which(Round3Pred$Slot==as.character(TheseSlots$WeakSeed[i+56])),"P
redictedWinner"]

  Round4Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
                          & AnnualSummaries$Team
==Round4Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
                          & AnnualSummaries$Team
==Round4Games[i,"Team2"]),3:16]
}
```

```
#Create predictions on round 4
pred = predict(thisModel, Round4Games)
Round4Pred = data.frame(Slot = Slots[57:60,"Slot"],PredictedWinner = 0)
for (i in 1:4){
  if (pred[i] == 1){
    Round4Pred[i,"PredictedWinner"] = Round4Games[i,"Team1"]
  }
  else{
    Round4Pred[i,"PredictedWinner"] = Round4Games[i,"Team2"]
  }
}
TheseSlots$Prediction[57:60] = Round4Pred[,"PredictedWinner"]


## Round 5
Round5Games = as.data.frame(matrix(0, ncol = 17, nrow = 2))
colnames(Round5Games) = mycolnames
Round5Games$Season = 2022

for (i in 1:2){
  Round5Games[i,"Team1"] =

Round4Pred[which(Round4Pred$Slot==as.character(TheseSlots$StrongSeed[i+60])),
"PredictedWinner"]

  Round5Games[i,"Team2"] =

Round4Pred[which(Round4Pred$Slot==as.character(TheseSlots$WeakSeed[i+60])),"P
redictedWinner"]

  Round5Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
                          & AnnualSummaries$Team
==Round4Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round4Games[i,"Season"]
                          & AnnualSummaries$Team
==Round4Games[i,"Team2"]),3:16]
}

#Create predictions on round 5
pred = predict(thisModel, Round5Games)
Round5Pred = data.frame(Slot = Slots[61:62,"Slot"],PredictedWinner = 0)
for (i in 1:2){
  if (pred[i] == 1){
    Round5Pred[i,"PredictedWinner"] = Round5Games[i,"Team1"]
  }
  else{
    Round5Pred[i,"PredictedWinner"] = Round5Games[i,"Team2"]
  }
```

```
}
TheseSlots$Prediction[61:62] = Round5Pred[,"PredictedWinner"]


## Round 6
Round6Games = as.data.frame(matrix(0, ncol = 17, nrow = 1))
colnames(Round6Games) = mycolnames
Round6Games$Season = 2022

for (i in 1:1){
  Round6Games[i,"Team1"] =

Round5Pred[which(Round5Pred$Slot==as.character(TheseSlots$StrongSeed[i+62])),
"PredictedWinner"]

  Round6Games[i,"Team2"] =

Round5Pred[which(Round5Pred$Slot==as.character(TheseSlots$WeakSeed[i+62])),"P
redictedWinner"]

  Round6Games[i,4:17] =
    AnnualSummaries[which(AnnualSummaries$Season==Round6Games[i,"Season"]
                          & AnnualSummaries$Team
==Round6Games[i,"Team1"]),3:16]-
    AnnualSummaries[which(AnnualSummaries$Season==Round6Games[i,"Season"]
                          & AnnualSummaries$Team
==Round6Games[i,"Team2"]),3:16]
}

#Create predictions on round 6
pred = predict(thisModel, Round6Games)
Round6Pred = data.frame(Slot = Slots[63,"Slot"],PredictedWinner = 0)
for (i in 1:1){
  if (pred[i] == 1){
    Round6Pred[i,"PredictedWinner"] = Round6Games[i,"Team1"]
  }
  else{
    Round6Pred[i,"PredictedWinner"] = Round6Games[i,"Team2"]
  }
}
TheseSlots$Prediction[63] = Round6Pred[,"PredictedWinner"]

Rounds = 0
for (i in 1:32){
  Rounds[i] = 1
}
for (i in 33:48){
  Rounds[i] = 2
}
```

```
for (i in 49:56){
  Rounds[i] = 3
}
for (i in 57:60){
  Rounds[i] = 4
}
for (i in 61:62){
  Rounds[i] = 5
}
for (i in 63){
  Rounds[i] = 6
}

Results = data.frame(Round = Rounds, Predicted = TheseSlots$Prediction)

return(Results)
}


PredictWinners2022(rfr.fit2)
```

Round <dbl>	Predicted <dbl>
1	1124
1	1246
1	1345
1	1417
1	1388
1	1439
1	1293
1	1314
1	1211
1	1181

Round <dbl>	Predicted <dbl>
1	1403
1	1436
1	1308
1	1323
1	1172
1	1129
1	1242
1	1120
1	1159
1	1355

Round <dbl>	Predicted <dbl>
1	1234
1	1261
1	1425
1	1166
1	1112
1	1437
1	1397
1	1151
1	1222
1	1161

Round <dbl>	Predicted <dbl>
1	1260
1	1395
2	1124
2	1293
2	1345
2	1417
2	1211
2	1181
2	1323
2	1436

Round <dbl>	Predicted <dbl>
2	1242
2	1120
2	1159
2	1355
2	1112
2	1437
2	1397
2	1222
3	1124
3	1293

Round <dbl>	Predicted <dbl>
3	1211
3	1181
3	1242
3	1120
3	1112
3	1397
4	1293
4	1211
4	1242
4	1112

Round <dbl>	Predicted <dbl>
5	1211
5	1242
6	1211