STAT 327: Project

Dawson Kinsman and Sabastian Zuhorski

2023-04-25

Functions and Classes

Some functions and classes are imported from a separate file.

```r
source('functions.R')

# function to simulate a random vs. random game and record the outcome in a dataframe
randomSimulation = function(numSims, boardSize = 3, board = NA, df, seed = 0, colName = NA, player = 1)
  # set seed
  set.seed(seed)

  #simulate TTT game numSims times
  winners<- replicate(numSims, randtictactoe(boardSize, board, player = player))
  #table(winners)

  #create a dataframe from the winner results for plotting & tables
  if (is.na(colName)){
    hist(winners,
      main = paste0('Results of 1,000 ',boardSize,'x', boardSize,' Tic-Tac-Toe Games'))
    df2 <- data.frame(Winner = winners,
                 'Board Size' = rep(paste0(boardSize,'x', boardSize), numSims))
  }
  else if (colName == 'First Move' | colName == 'Second Move'){
    if (colName == 'First Move'){
      df2 <- data.frame(Winner = winners, 'Move' = rep(which(board==1), numSims))}
    else{
      df2 <- data.frame(Winner = winners, 'First Move' = rep(which(board==1)),
                        'Second Move' = rep(which(board==-1), numSims))}

  }
  else {
    df2 <- data.frame(Winner = winners, numSims = rep(as.character(colName),numSims))
  }

  # merge and return dataframes
  df <- rbind(df, df2)
  return(df)
}

# function to simulate MCTS vs. Random player game
MCTSvsRandomSim = function(numSims, first = c('MCTS', 'random'), iterations = 500, board = NA, player =
  if (length(which(is.na(board)))==1) board = array(rep(NA, 3^2),
                                            dim = c(3, 3))

  if (first == 'MCTS') winner = replicate(numSims,
                                          MCTSvsRandom(iterations = iterations, board = board, player =
  else winner = replicate(numSims,
                          RandomvsMCTS(iterations = iterations, board = board, player = player))
  return(winner)
}
```

Constant Values

```
n = 1e3 # number of simulations
```

Introductoion

Estimating the probability of a player winning a game can be a complex and difficult problem as the number of legal moves increase. In recent years, use of simulation and the Monte Carlo method have proven useful in the field of game theory and determining the optimal move or choice, given a game board or state. Since the success of the artificial intelligence (AI) gamebots AlphaGo and its successors, the Monte Carlo Tree Search (MCTS) has become an integral tool for developing game AIs. Go has 2.1×10^{170} legal moves which makes the construction of a complete game tree computationally impossible, and this is where MCTS excels. MCTS does not require the construction of a complete game tree to determine the optimal move; instead, MCTS uses repeated simulation at leaf nodes and backpropogation of results to determine the best move. For simplicity and demonstrability of the effectiveness of the Monte Carlo Tree Search method, we implement a MCTS for tic-tac-toe. Our research questions are:

- How does the size of the board effect the probability of winning?
- For each player, what are the probabilities of winning…
  - If each player moves randomly?
  - If one player strategizes and the other plays randomly?
- For which first move does Player 1 have the highest probability of winning?
- Given Player 1's first Move, for which second move does Player 2 have the highest probability of winning?

Monte Carlo Tree Search (MCTS)

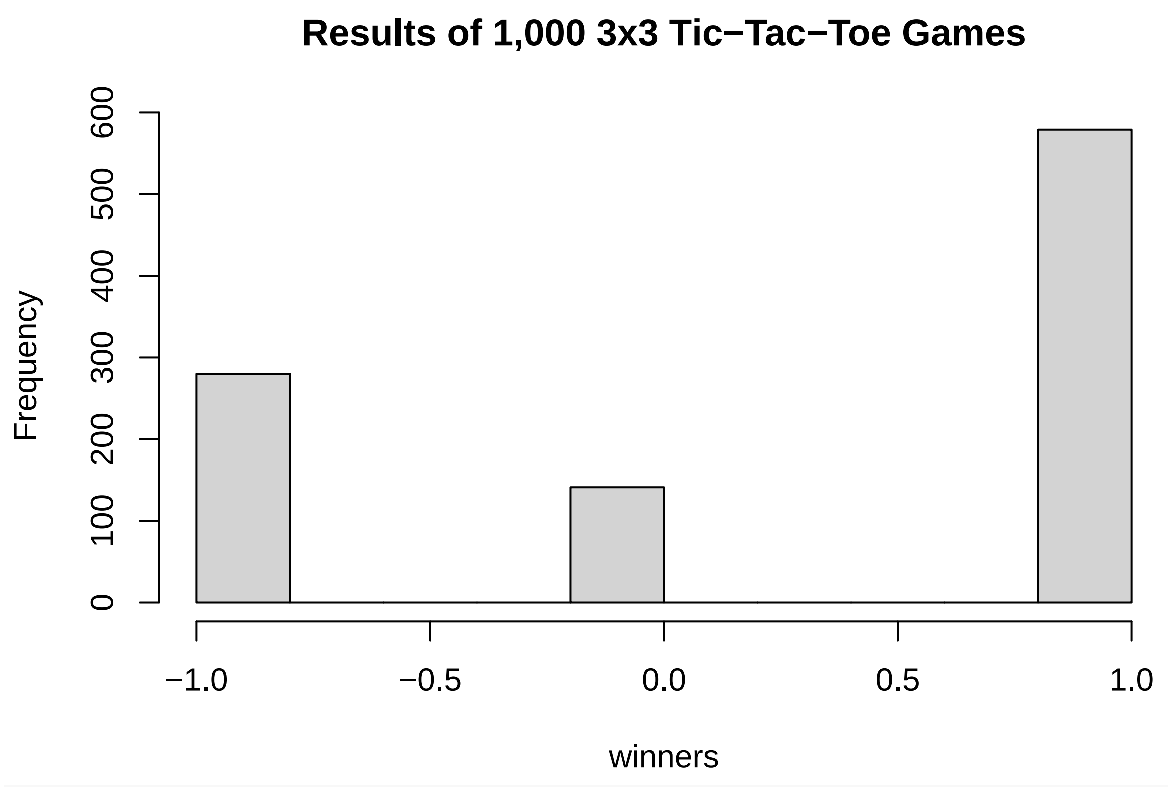
Random Player vs. Random Player:

For these simulations, we play 1,000 tc-tac-toe games and assume that Player 1 moves first.

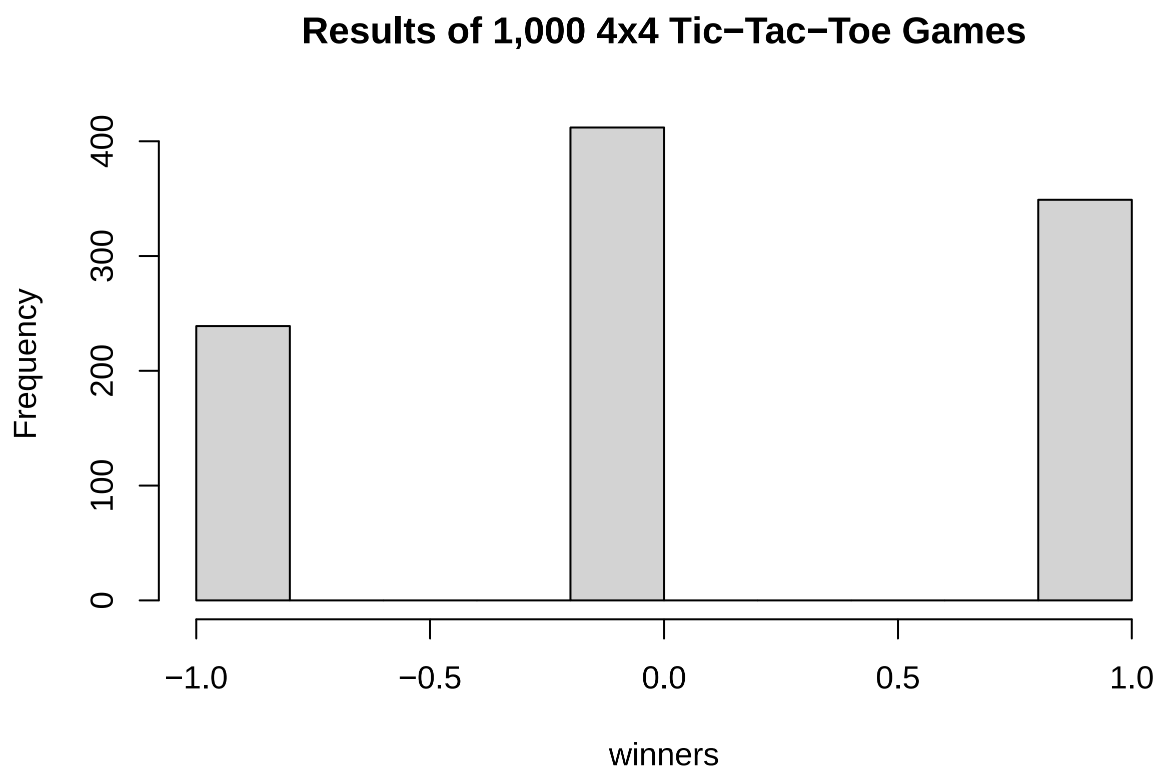
How does the size of the board effect the probability of winning?

First, we simulate the data using the functions we have written. We choose to run 1,000 simulations for boards with dimensions 3×3, 4×4, 5×5, and 10×10.

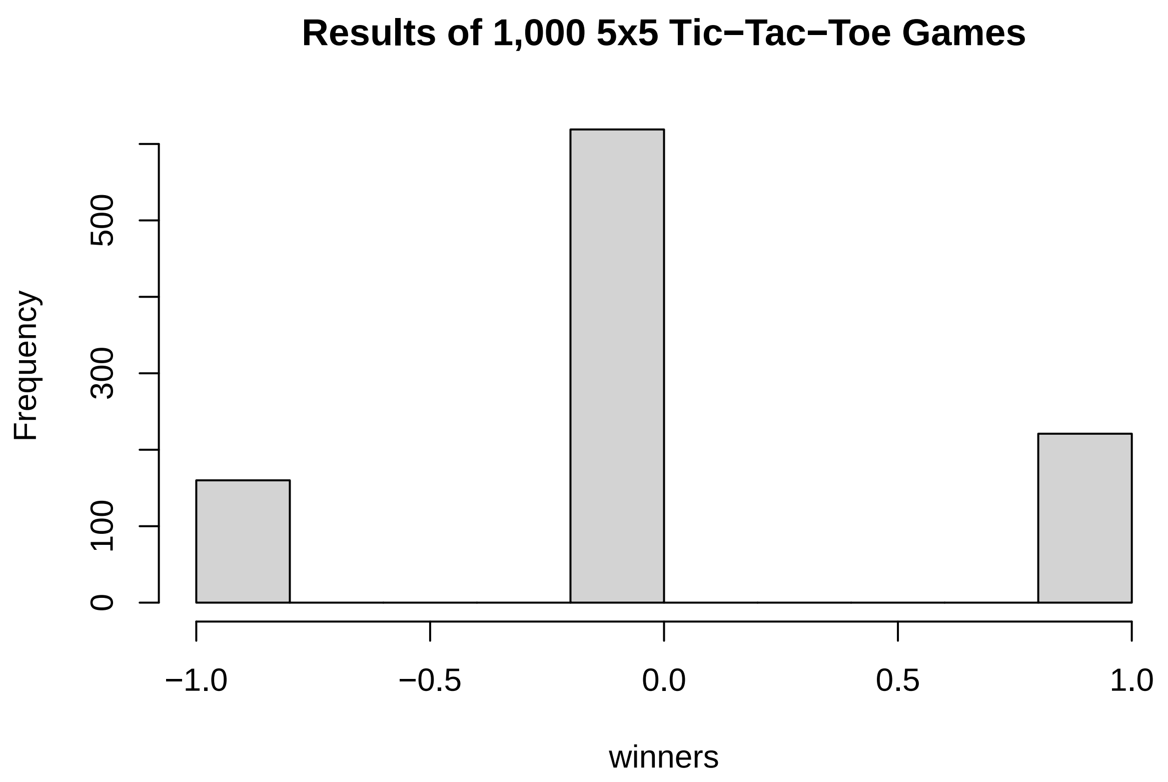
```
#create data frame to store winner results
df <- data.frame(matrix(ncol = 2, nrow = 0))
colnames(df) <- c('Winner', 'Board Size')

# update the the DF with the winner results of 1,000 games on varying board sizes.
df <- randomSimulation(n, 3, df = df)
```



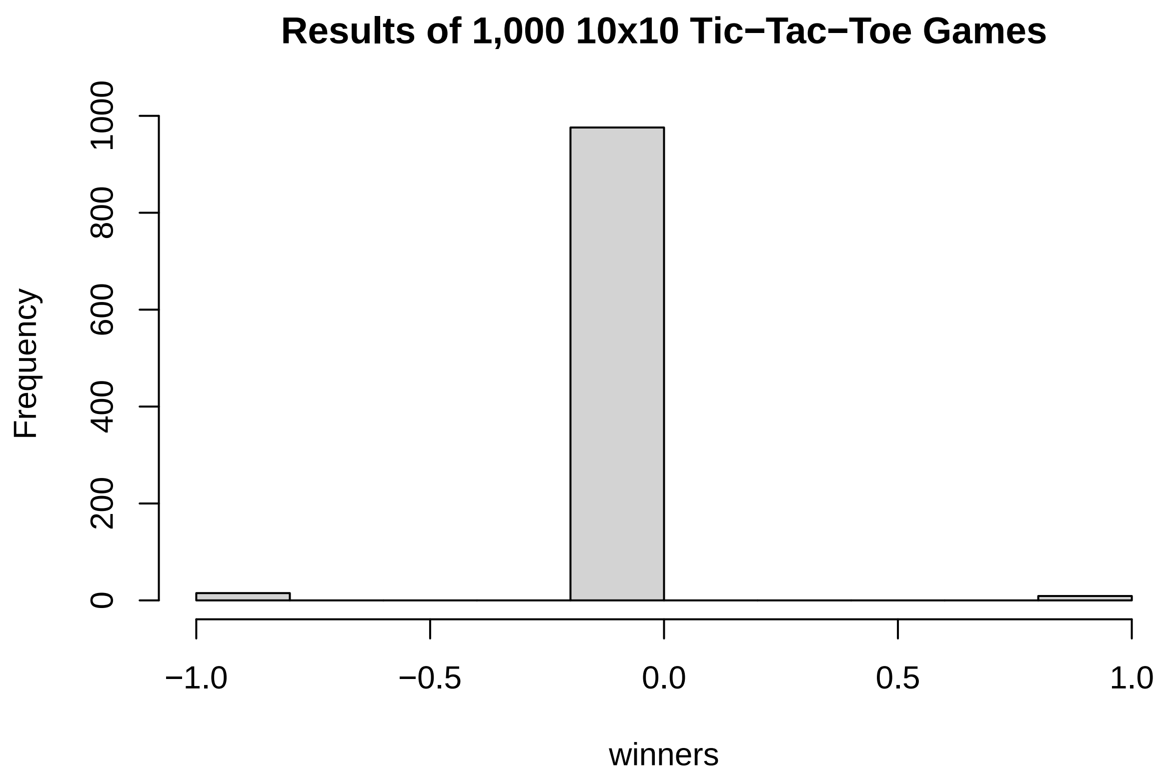
```
df <- randomSimulation(n, 4, df = df, seed = 1)
```



```
df <- randomSimulation(n, 5, df = df, seed = 2)
```



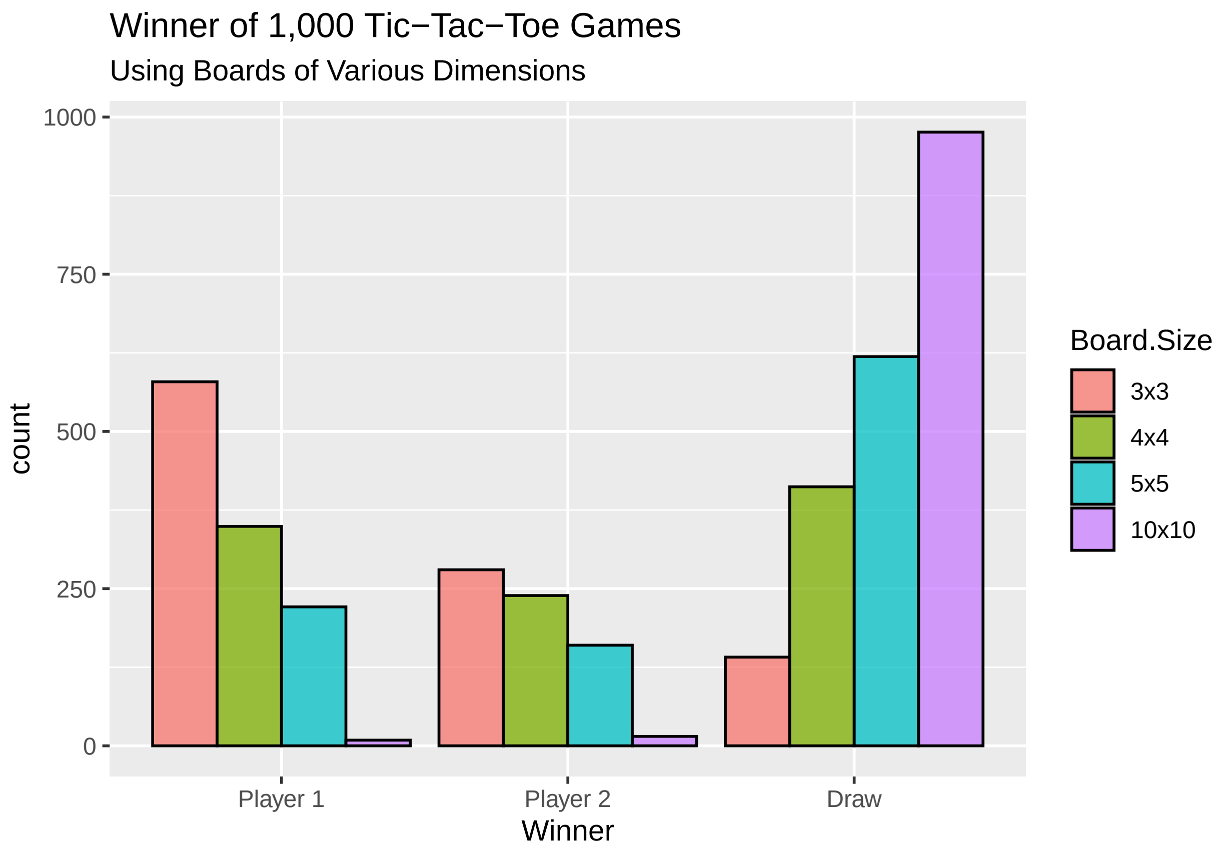
```
df <- randomSimulation(n, 10, df = df, seed = 3)
```



We can visualize the number of times each player wins using a bar chart.

```
# make each of the variables a factor
boards <- c('3x3','4x4','5x5','10x10')
```

```
df$Board.Size <- factor(df$Board.Size, levels = boards)
df$Winner <- factor(df$Winner, levels = c(1, -1, 0),
                    labels = c('Player 1', 'Player 2', 'Draw'))

# plot a bar chart
ggplot(df, aes(Winner)) +
  geom_bar(aes(fill = Board.Size), position = 'dodge', color = 'black',
           alpha = 0.75) +
  ggtitle('Winner of 1,000 Tic-Tac-Toe Games',
          subtitle = 'Using Boards of Various Dimensions')
```



We can also calculate the probabilities of each outcome on each board.

```
# create new DF to store the previous results and their probabilities
results <- data.frame(matrix(ncol = 3, nrow = 0))
colnames(results) <- c('Winner', 'Board Size', 'Probability')

# update the new DF with the probabilities of each player winning on each board size
for (i in boards){
  df_board <- df %>% filter(Board.Size == i)
  p1 <- length(df_board$Winner[df_board$Winner == 'Player 1']) / n; p1
  p2 <- length(df_board$Winner[df_board$Winner == 'Player 2']) / n; p2
  draw <- length(df_board$Winner[df_board$Winner == 'Draw']) / n; draw

  res <- data.frame(Winner = c('Player 1', 'Player 2', 'Draw'),
                 'Board Size' = i, Probability = c(p1,p2, draw))
  results <- rbind(results, res)
```

Table 1: Probabilities of Winning Based on Board Size

Board.Size	Player 1	Player 2	Draw
3x3	0.579	0.280	0.141
4x4	0.349	0.239	0.412
5x5	0.221	0.160	0.619
10x10	0.009	0.015	0.976

```r
}

# create and output a pivot table of the results
pt <- pivot_wider(results, id_cols = Board.Size, names_from = Winner, values_from = Probability);
knitr::kable(pt, align = 'cccc',
             caption = 'Probabilities of Winning Based on Board Size') %>%
  kable_styling(full_width = F)
```

From both the bar chart and the table of probabilities, we can see that on a standard 3×3 tic-tac-toe board, the probability of Player 1 winning is nearly 57.9%, while the probability of Player 2 winning is almost half of that (28%). As the game space increases, the probability of either player winning decreases, while the probability of ending in a draw increases. With a 10×10 board, the probability of either Player winning is less than 3%, that is, draws are the most common outcome; however, Player 2, actually has a slightly larger probability of winning than Player 1.

For which first move does Player 1 have the highest probability of winning?

For the rest of the questions, we only look at the 3×3 tic-tac-toe board, which we implement using a 3×3 array. `R` indexing in a 3×3 array is as follows.

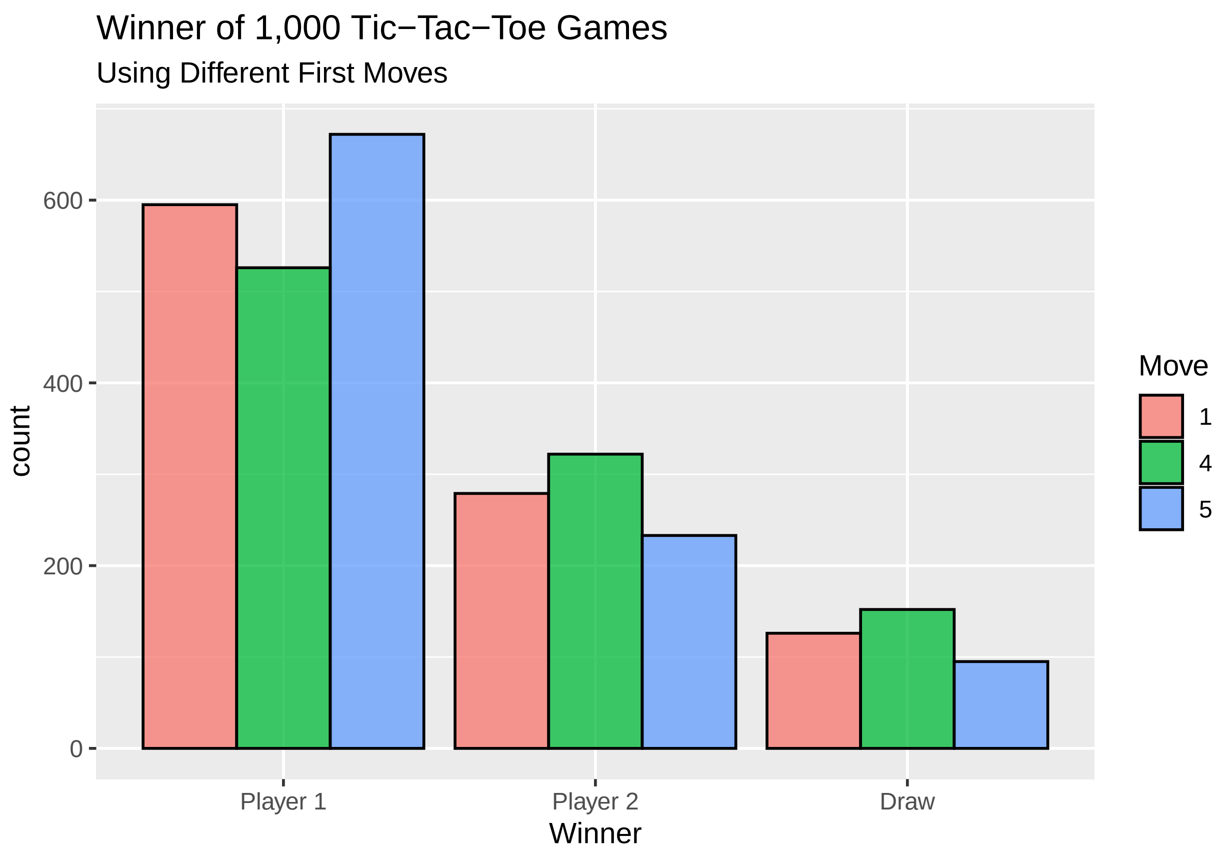
1	4	
2	5	8
3	6	9

There are 3 general first moves, those being, 1, 4, and 5. All of the other first moves can be written as a rotation or reflection of these three (see [5]). Thus, given each of these starting boards, we can simulate the outcomes of 1,000 games for each board and record the outcomes.

```r
# create the boards to be simulated
board1 <- array(c(1, rep(NA, 8)), dim = c(3,3))
board4 <- array(c(rep(NA, 3), 1, rep(NA, 4)), dim = c(3,3))
board5 <- array(c(rep(NA,4),1,rep(NA,4)), dim = c(3,3))

# create new DF to store the results
df1 <- data.frame(matrix(ncol = 2, nrow = 0))
colnames(df1) <- c('Winner', 'Move')

# update the DF with the results
df1 <- randomSimulation(n, 3, board1, df = df1, seed = 0, 'First Move', -1)
df1 <- randomSimulation(n, 3, board4, df = df1, seed = 1, 'First Move', -1)
df1 <- randomSimulation(n, 3, board5, df = df1, seed = 2, 'First Move', -1)
```

```r
# make each variable in the DF a factor
df1$Move <- as.factor(df1$Move)
```

```
df1$Winner <- factor(df1$Winner, levels = c(1, -1, 0),
                    labels = c('Player 1', 'Player 2', 'Draw'))

# plot a barchart
ggplot(df1, aes(Winner)) +
  geom_bar(aes(fill = Move), position = 'dodge', color = 'black',
           alpha = 0.75) +
  ggtitle('Winner of 1,000 Tic-Tac-Toe Games',
          subtitle = 'Using Different First Moves')
```



```
# similar to before create a new DF to store probabilities
results <- data.frame(matrix(ncol = 3, nrow = 0))
colnames(results) <- c('Winner', 'First Move', 'Probability')

# calculate and update the new DF with the probabilities
for (i in c(1,4,5)){
  df_fmove <- df1 %>% filter(Move == i)
  p1 <- length(df_fmove$Winner[df_fmove$Winner == 'Player 1']) / n; p1
  p2 <- length(df_fmove$Winner[df_fmove$Winner == 'Player 2']) / n; p2
  draw <- length(df_fmove$Winner[df_fmove$Winner == 'Draw']) / n; draw

  res <- data.frame(Winner = c('Player 1', 'Player 2', 'Draw'),
                    'First Move' = i, Probability = c(p1,p2, draw))
  results <- rbind(results, res)
}

# create and output a pivot table of the probabilities
```

Table 2: Probabilities of Winning Based on Player 1's First Move

First.Move	Player 1	Player 2	Draw
1	0.595	0.279	0.126
4	0.526	0.322	0.152
5	0.672	0.233	0.095

```r
pt <- pivot_wider(results, id_cols = First.Move, names_from = Winner, values_from = Probability);
knitr::kable(pt, align = 'cccc',
             caption = "Probabilities of Winning Based on Player 1's First Move") %>%
  kable_styling(full_width = F)
```

To answer our research question, box 5, the center box, is the optimal first move for Player 1 to have the highest probability of wining. The probability of Player 1 winning with the first move in box 5 is higher than the probability of Player 1 winning in general on a 3×3 board while the probability of Player 1 winning given their first move is in a corner box (box 1) is approximately equal to the probability of Player 1 winning. These results were expected since picking the center or corner box is a common strategy employed to win tic-tac-toe. Lastly, we note that if Player 1 first picks a middle box that is *not* the center most box, Player 1's probability of winning is the lowest.

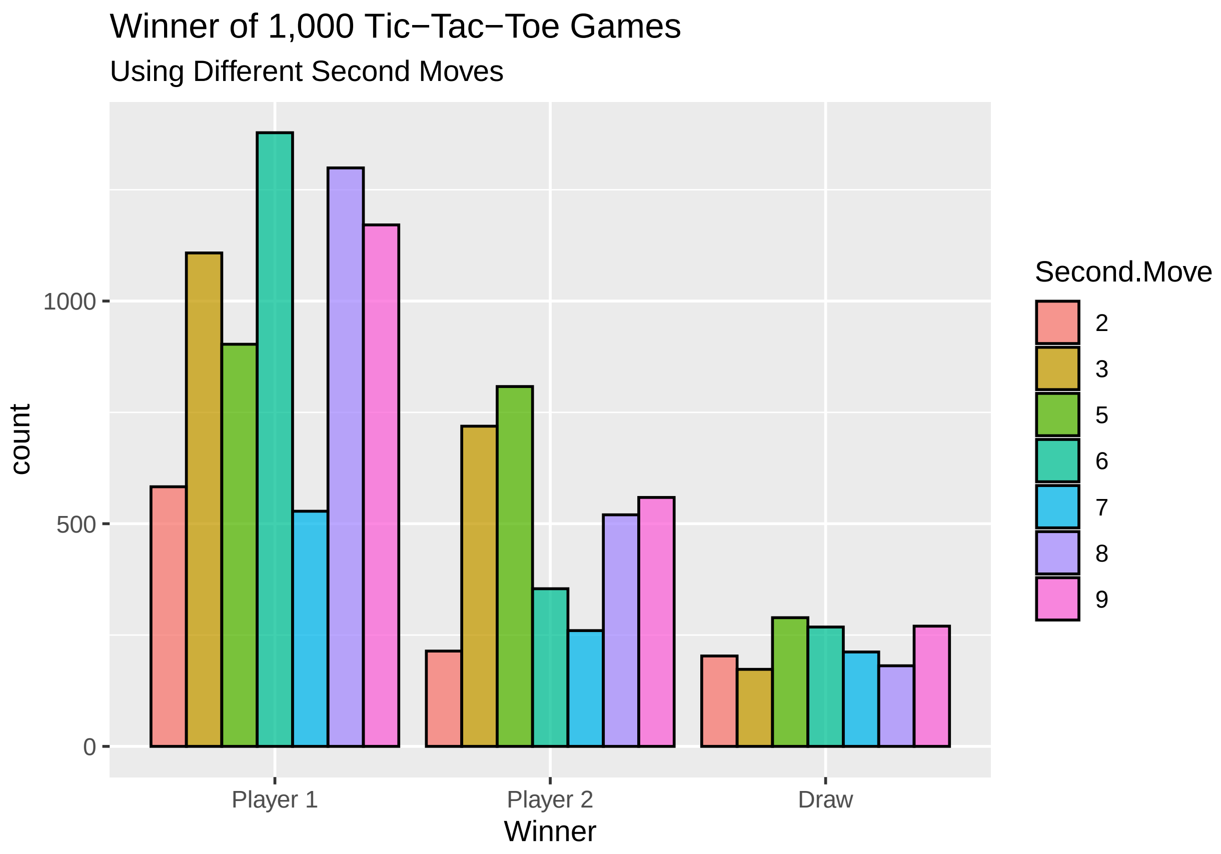
NOTE: Upon writing the report, we noticed an error in our code that was used to generate the figures for the presentation. We have fixed the code and updated the final presentation. Prior, given a board with Player 1's first move, Player one would make an additional second move before alternating to Player 2. We note that either way, box 5 is the optimal first move for Player 1.

Given Player 1's First Move, for which second move does Player 2 have the highest probability of winning?

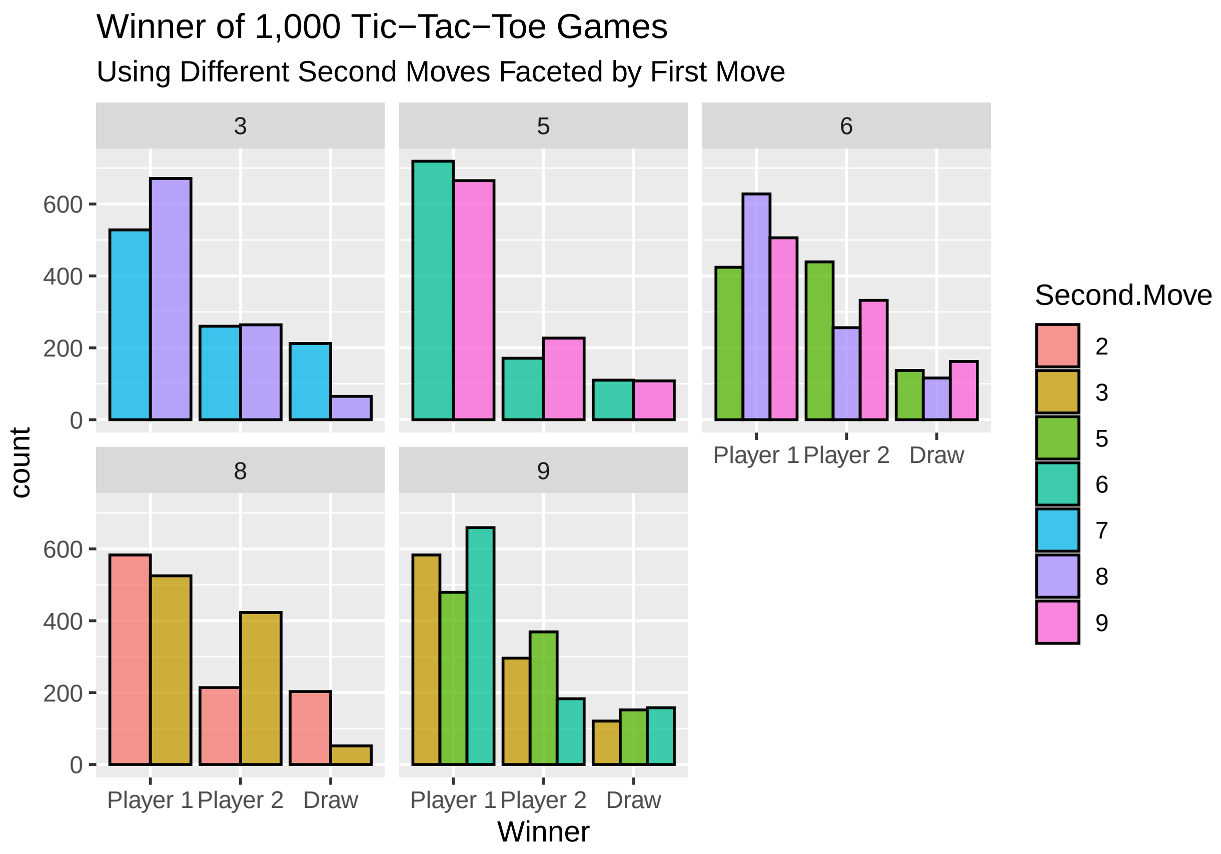
Once more, we can utilize the board's symmetry to reduce the number of boards we have to simulate to 12 different boards (see [5]).

```r
# create boards with different first 2 moves
board1 <- array(c(rep(NA,2), 1, rep(NA, 3), -1, rep(NA, 2)), dim = c(3,3))
board2 <- array(c(rep(NA,2), 1, rep(NA, 4), -1, NA), dim = c(3,3))
board3 <- array(c(rep(NA,2),-1,rep(NA,5), 1), dim = c(3,3))
board4 <- array(c(rep(NA,4),-1,rep(NA,3), 1), dim = c(3,3))
board5 <- array(c(rep(NA,5),-1,rep(NA,2), 1), dim = c(3,3))
board6 <- array(c(rep(NA,2),-1,rep(NA,4), 1, NA), dim = c(3,3))
board7 <- array(c(NA,-1,rep(NA,5), 1, NA), dim = c(3,3))
board8 <- array(c(rep(NA,5),1,NA, -1, NA), dim = c(3,3))
board9 <- array(c(rep(NA,4),-1, 1, rep(NA,3)), dim = c(3,3))
board10 <- array(c(rep(NA,5),1,rep(NA,2), -1), dim = c(3,3))
board11 <- array(c(rep(NA,4),1,rep(NA,3), -1), dim = c(3,3))
board12 <- array(c(rep(NA,4),1, -1, rep(NA, 3)), dim = c(3,3))

# create new DF to store the winning results
df1 <- data.frame(matrix(ncol = 3, nrow = 0))
colnames(df1) <- c('Winner', 'First Move', 'Second Move')

# update the DF with the results of simulations
df1 <- randomSimulation(n, 3, board1, df = df1, seed = 0, 'Second Move')
df1 <- randomSimulation(n, 3, board2, df = df1, seed = 1, 'Second Move')
df1 <- randomSimulation(n, 3, board3, df = df1, seed = 2, 'Second Move')
```

```r
df1 <- randomSimulation(n, 3, board4, df = df1, seed = 3, 'Second Move')
df1 <- randomSimulation(n, 3, board5, df = df1, seed = 4, 'Second Move')
df1 <- randomSimulation(n, 3, board6, df = df1, seed = 5, 'Second Move')
df1 <- randomSimulation(n, 3, board7, df = df1, seed = 6, 'Second Move')
df1 <- randomSimulation(n, 3, board8, df = df1, seed = 7, 'Second Move')
df1 <- randomSimulation(n, 3, board9, df = df1, seed = 8, 'Second Move')
df1 <- randomSimulation(n, 3, board10, df = df1, seed = 9, 'Second Move')
df1 <- randomSimulation(n, 3, board11, df = df1, seed = 10, 'Second Move')
df1 <- randomSimulation(n, 3, board12, df = df1, seed = 11, 'Second Move')

# make each variable factors
df1$First.Move <- as.factor(df1$First.Move)
df1$Second.Move <- as.factor(df1$Second.Move)
df1$Winner <- factor(df1$Winner, levels = c(1, -1, 0),
                     labels = c('Player 1', 'Player 2', 'Draw'))

# plot barplot of winners based on Player 2's second move
ggplot(df1, aes(Winner)) +
  geom_bar(aes(fill = Second.Move), position = 'dodge', color = 'black',
           alpha = 0.75) +
  ggtitle('Winner of 1,000 Tic-Tac-Toe Games',
          subtitle = 'Using Different Second Moves')
```



```r
# plot the barplots faceted by Player 1's First Move
ggplot(df1, aes(Winner)) +
  geom_bar(aes(fill = Second.Move), position = 'dodge', color = 'black',
           alpha = 0.75) + facet_wrap(~First.Move) +
  ggtitle('Winner of 1,000 Tic-Tac-Toe Games',
          subtitle = 'Using Different Second Moves Faceted by First Move')
```



```
# create new DF to store the results
results <- data.frame(matrix(ncol = 4, nrow = 0))
colnames(results) <- c('Winner', 'Second Move', 'Probability')

# calculate probabilities and update the DF
for (i in 1:9){
  df_fmove <- df1 %>% filter(First.Move==i)
  #print(dim(df_fmove))
  for (j in 1:9){
  df_smove <- df_fmove %>% filter(Second.Move == j)
  p1 <- length(df_smove$Winner[df_smove$Winner == 'Player 1']) / dim(df_smove)[1];
  p2 <- length(df_smove$Winner[df_smove$Winner == 'Player 2']) / dim(df_smove)[1];
  draw <- length(df_smove$Winner[df_smove$Winner == 'Draw']) / dim(df_smove)[1];

  res <- data.frame(Winner = c('Player 1', 'Player 2', 'Draw'), 'First Move' = i,
                    'Second Move' = j, Probability = c(p1,p2, draw))
  results <- rbind(results, res)
  }
}

# create and output a pivot table
pt <- pivot_wider(results, id_cols = c(First.Move,Second.Move), names_from = Winner, values_from = Proba
pt <- na.omit(pt);
knitr::kable(pt, align = 'cccc') %>% kable_styling(full_width = F)
```

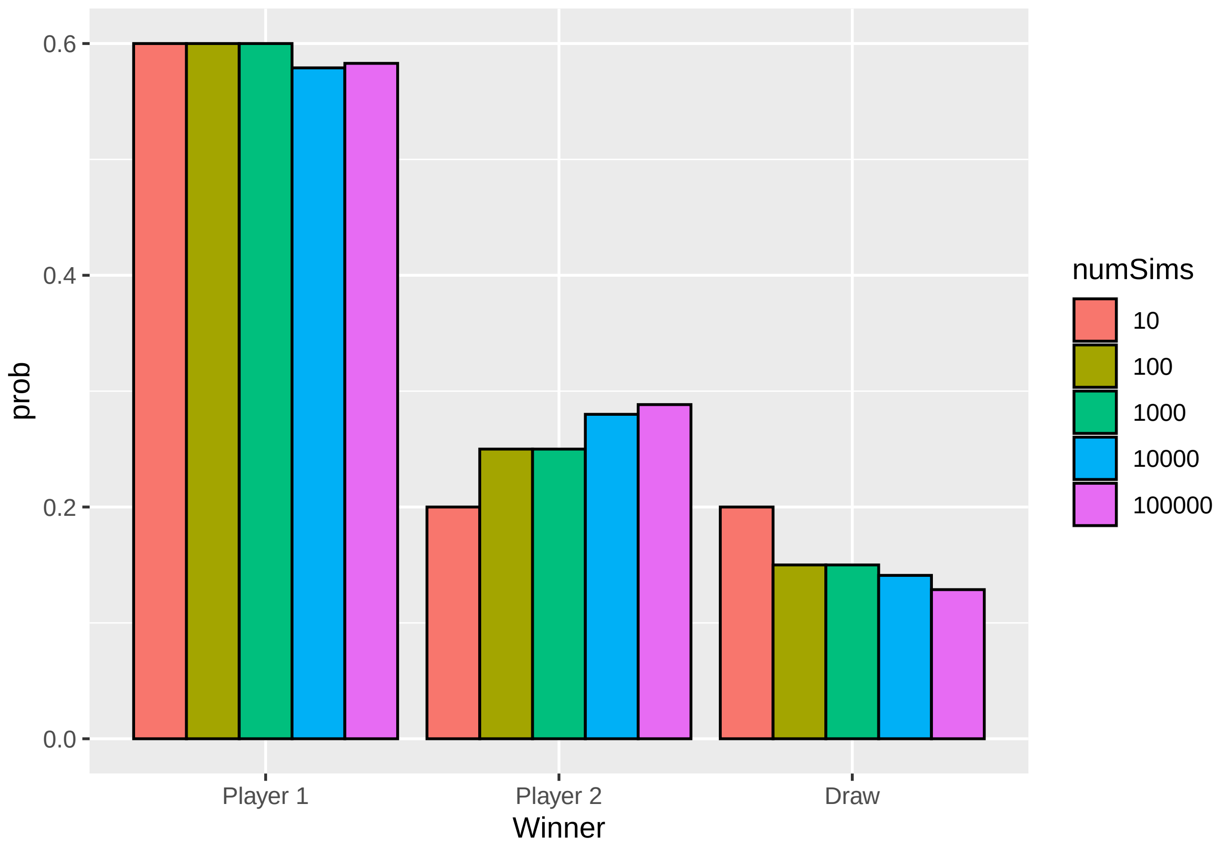
First.Move	Second.Move	Player 1	Player 2	Draw
3	7	0.528	0.260	0.212
3	8	0.671	0.264	0.065
5	6	0.719	0.171	0.110
5	9	0.665	0.227	0.108
6	5	0.424	0.439	0.137
6	8	0.628	0.256	0.116
6	9	0.506	0.332	0.162
8	2	0.583	0.214	0.203
8	3	0.525	0.423	0.052
9	3	0.583	0.296	0.121
9	5	0.479	0.369	0.152
9	6	0.659	0.183	0.158

The faceted bar plot is easiest to read to answer this question; however, we can also select Player 2's optimal move using the pivot table of probabilities of each outcome. We summarize the optimal move for Player 2 given Player 1's first move below

Convergence of Random vs. Random Probabilities:

We only simulated 1,000

```r
set.seed(0)
df <- data.frame(matrix(ncol = 2, nrow = 0))
colnames(df) <- c('Winner', 'numSims')

df <- randomSimulation(10, 3, df = df, colName = '10')
df <- randomSimulation(100, 3, df = df, colName = '100')
df <- randomSimulation(100, 3, df = df, colName = '1000')
df <- randomSimulation(1000, 3, df = df, colName = '10000')
df <- randomSimulation(10000, 3, df = df, colName = '100000')

df$numSims <- as.factor(df$numSims)
df$Winner <- factor(df$Winner, levels = c(1, -1, 0),
                    labels = c('Player 1', 'Player 2', 'Draw'))

df %>% count(Winner, numSims) %>% group_by( numSims) %>%
  mutate(prob = n/sum(n)) %>%
  ggplot(aes(x = Winner, y = prob, fill = numSims)) +
  geom_col(position = position_dodge(), color = 'black')
```



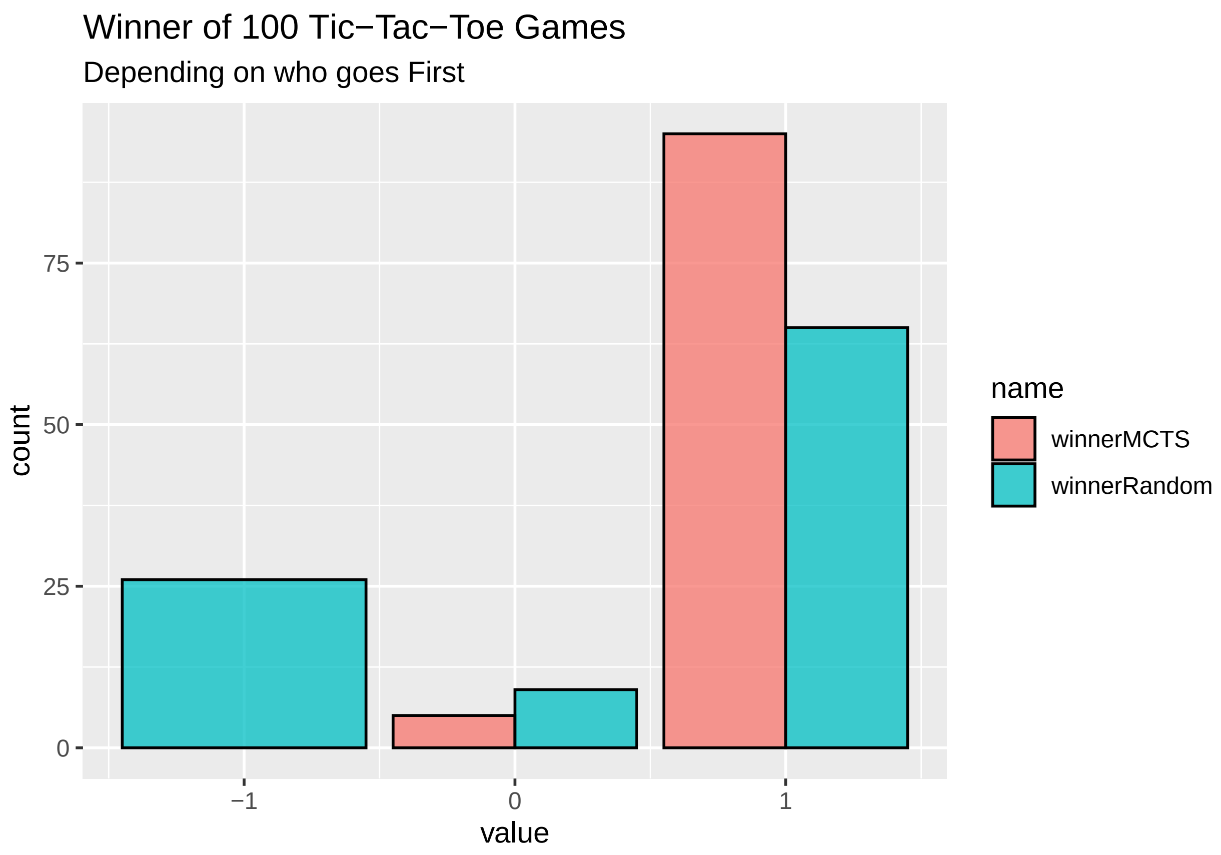
Monte Carlo Search Tree

MCTS vs. Random Simulation:

How does who goes first effect the probability of each outcome?

```
#set seed for reproducibility
set.seed(0)

# simulate 100 games, MCTS plays first
winnerMCTS = MCTSvsRandomSim(numSims =100, first = 'MCTS', iterations = 500)
# simulate 100 games, Random Player goes first
winnerRandom = MCTSvsRandomSim(numSims = 100, first = 'random', iterations = 500)
```

```
# create DF so that each game is a row with a column containing the who goes first
dfWinners <- pivot_longer(data.frame(winnerMCTS, winnerRandom),
                          cols = c('winnerMCTS', 'winnerRandom'))

# create a bar plot of the results
ggplot(dfWinners) +
  geom_bar(aes(value, fill = name), position = 'dodge', color = 'black',
           alpha = 0.75) +
  ggtitle('Winner of 100 Tic-Tac-Toe Games',
          subtitle = 'Depending on who goes First')
```



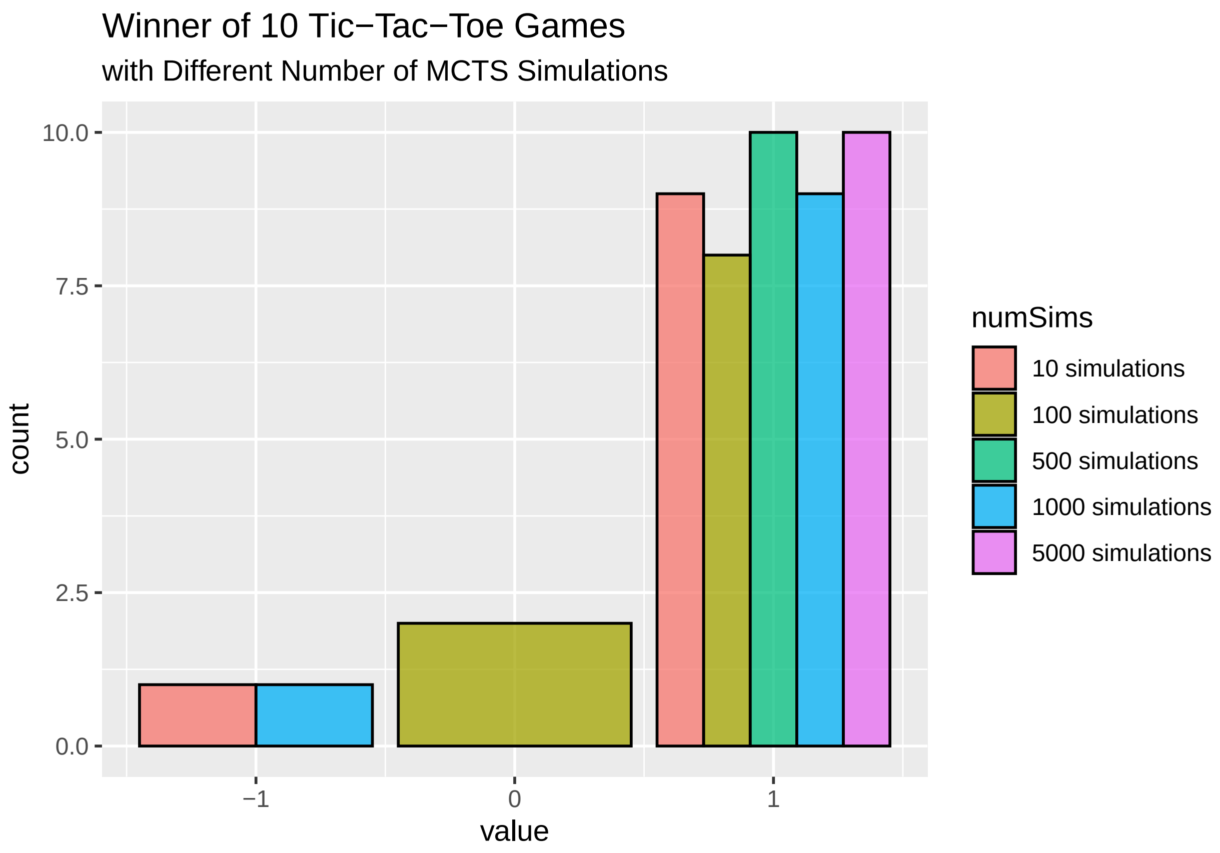
Comparing MCTS with a different number of iterations for each move

```r
#set seed for reproducibility
set.seed(0)

start.time <- Sys.time()
# simulate 10 games for differing numbers of iterations
winner10 = MCTSvsRandomSim(numSims =10, first = 'MCTS', iterations = 10)
winner100 = MCTSvsRandomSim(numSims =10, first = 'MCTS', iterations = 100)
winner500 = MCTSvsRandomSim(numSims =10, first = 'MCTS', iterations = 500)
winner1000= MCTSvsRandomSim(numSims =10, first = 'MCTS', iterations = 1000)
winner5000= MCTSvsRandomSim(numSims =10, first = 'MCTS', iterations = 5000)
#winner200 = MCTSvsRandomSim(numSims =1000, first = 'MCTS', iterations = 200)
end.time <- Sys.time()
time.taken <- end.time - start.time; time.taken
```

```
## Time difference of 1.011086 mins
```

```r
# create DF so that each game is a row with a column containing number of iterations
dfIter = data.frame(winner10, winner100, winner500, winner1000, winner5000)
winnerIterations = pivot_longer(dfIter, cols = colnames(dfIter))
winnerIterations$numSims =
  factor(winnerIterations$name, levels = c('winner10', 'winner100',
                                           'winner500', 'winner1000', 'winner5000'),
                                labels = c('10 simulations', '100 simulations',
                                           '500 simulations', '1000 simulations',
                                           '5000 simulations'))
```

```r
# create a bar plot of the results
ggplot(winnerIterations) +
  geom_bar(aes(value, fill = numSims), position = 'dodge', color = 'black',
           alpha = 0.75) +
  ggtitle('Winner of 10 Tic-Tac-Toe Games',
          subtitle = 'with Different Number of MCTS Simulations')
```

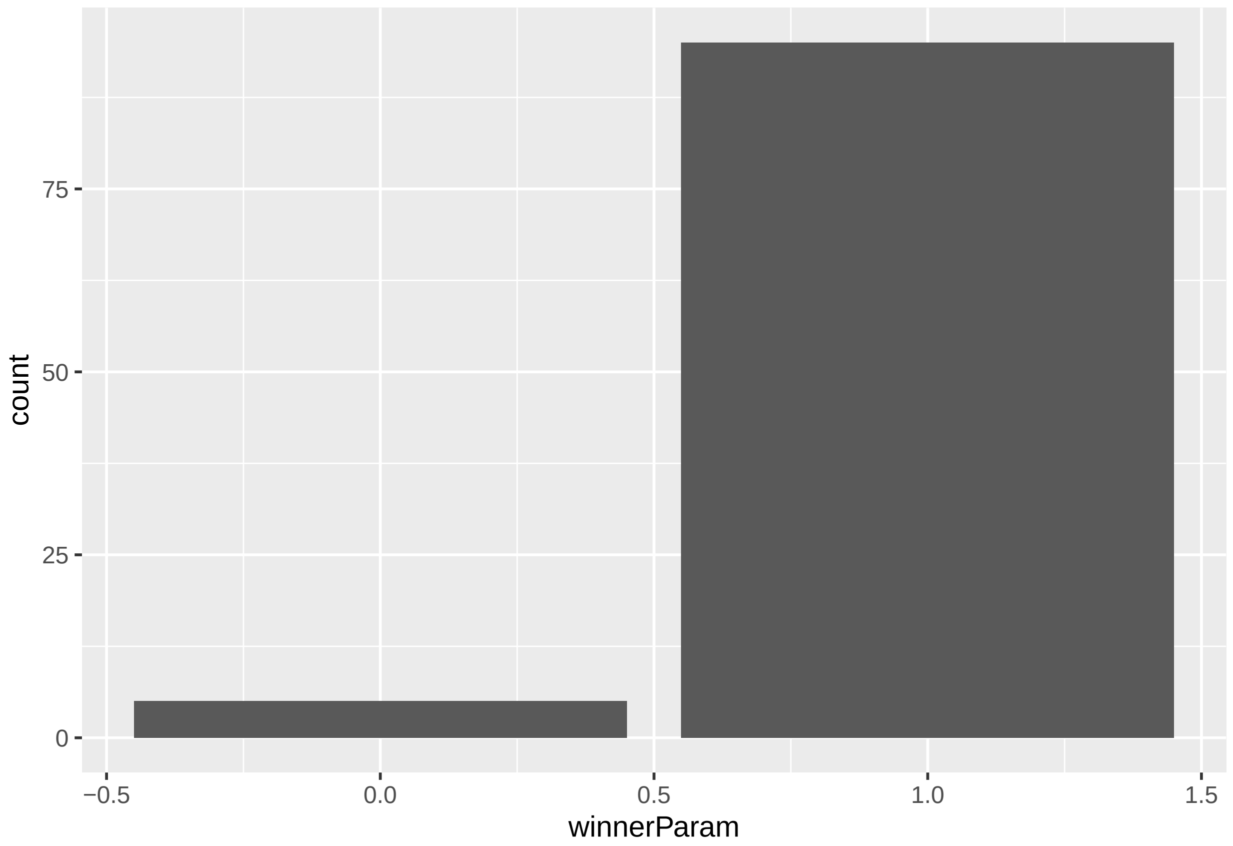


Test how many games and iterations to simulate

```r
set.seed(0)
start.time <- Sys.time()
winnerParam = MCTSvsRandomSim(numSims =100, first = 'MCTS', iterations = 500)
end.time <- Sys.time()
time.taken <- end.time - start.time; time.taken
```

```
## Time difference of 1.029251 mins
```

```r
dftest = data.frame(winnerParam)
ggplot(dftest) + geom_bar(aes(winnerParam))
```

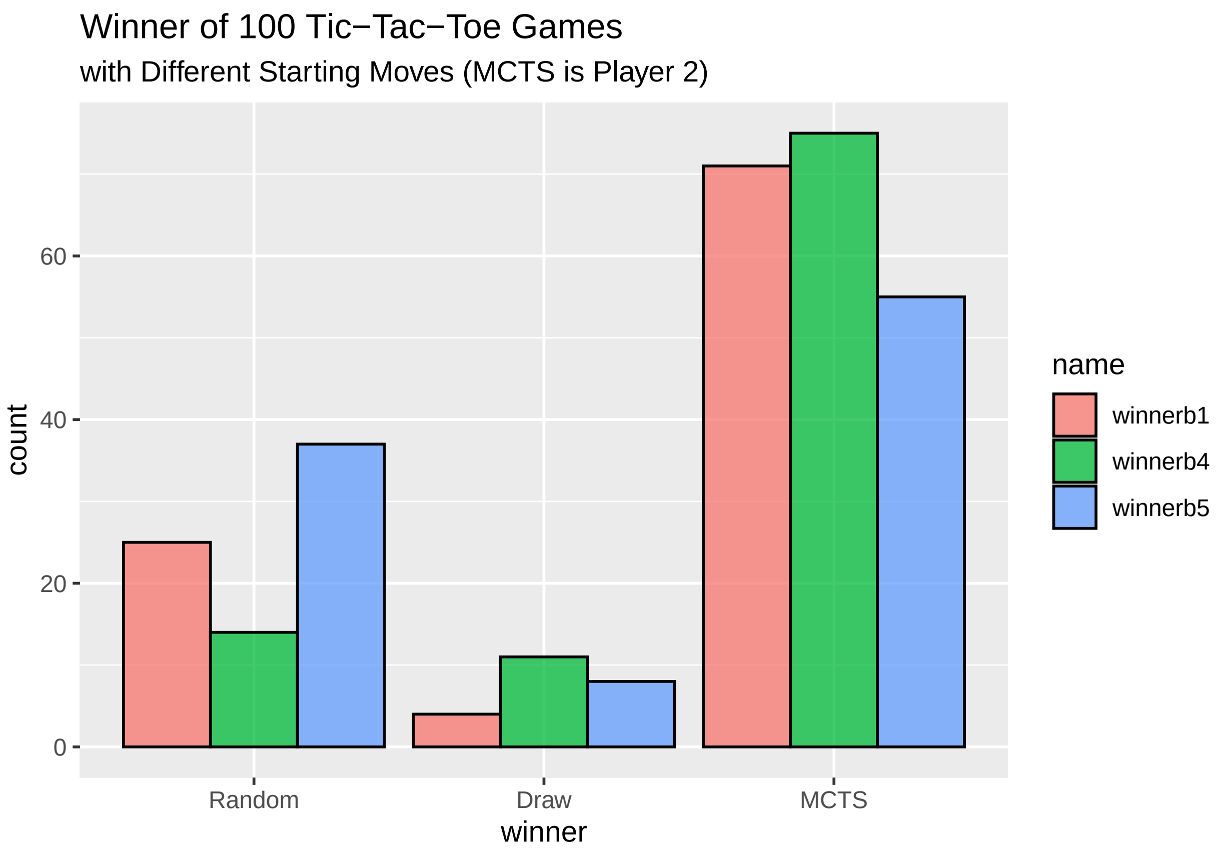


## Given a Random Player's first move, what are the probabilities of each outcome?

```r
# create the boards
board1 <- array(c(-1, rep(NA, 8)), dim = c(3,3))
board4 <- array(c(rep(NA, 3), -1, rep(NA, 4)), dim = c(3,3))
board5 <- array(c(rep(NA,4),-1,rep(NA,4)), dim = c(3,3))

set.seed(0)
start.time <- Sys.time()
# simulate 100 games with 500 iterations per move for each board
winnerb1= MCTSvsRandomSim(numSims =100, first = 'MCTS', iterations = 500, board = board1, player = -1)
winnerb4= MCTSvsRandomSim(numSims =100, first = 'MCTS', iterations = 500, board = board4, player = -1)
winnerb5= MCTSvsRandomSim(numSims =100, first = 'MCTS', iterations = 500, board = board5, player = -1)
end.time <- Sys.time()
time.taken <- end.time - start.time; time.taken
```

```
## Time difference of 2.393873 mins
```

```r
# create a DF to store all of the results and pivot so each row contains a game
winnerFirst <- data.frame(winnerb1, winnerb4, winnerb5)
winnerFirstLong = pivot_longer(winnerFirst, cols =colnames(winnerFirst))
winnerFirstLong$winner = factor(winnerFirstLong$value, levels = c(-1,0,1),
                                labels = c('Random', 'Draw', 'MCTS'))

# plot the barchart
ggplot(winnerFirstLong) +
  geom_bar(aes(winner, fill = name), position = 'dodge', color = 'black',
           alpha = 0.75) +
  ggtitle('Winner of 100 Tic-Tac-Toe Games',
       subtitle = 'with Different Starting Moves (MCTS is Player 2)')
```

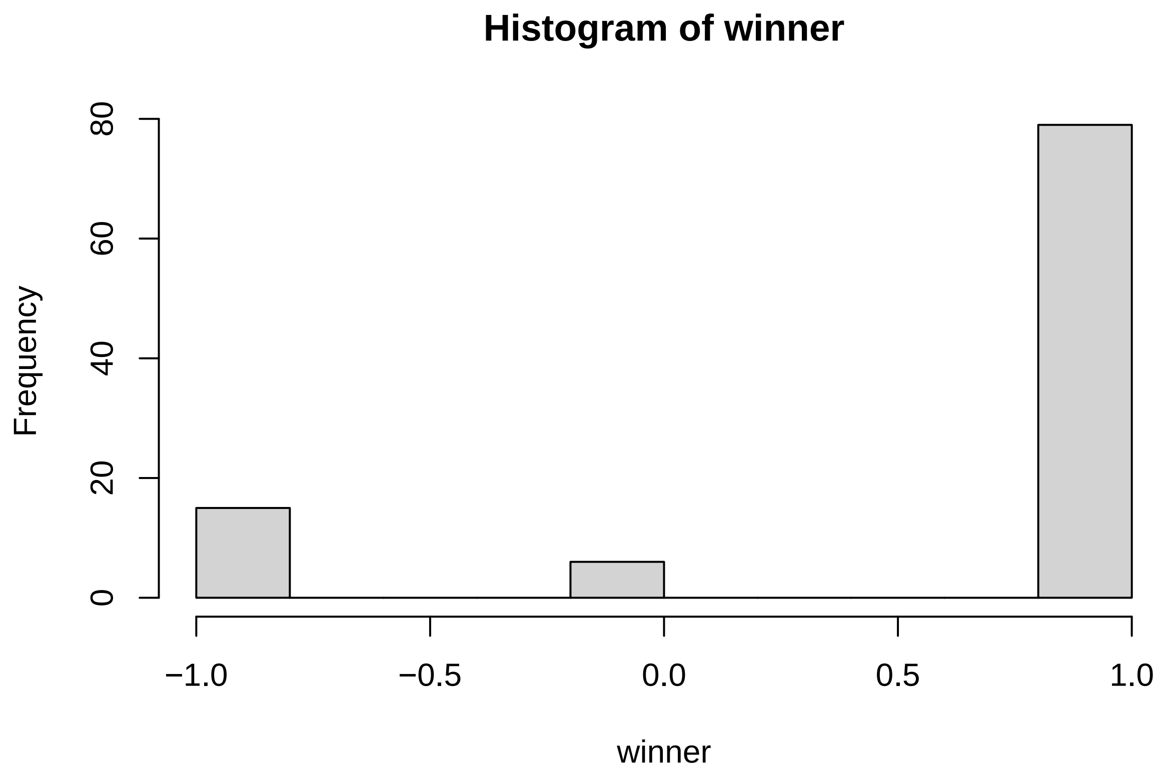


Conclusion

Future Research

MCTS vs. MCTS

```
winner = replicate(100, MCTSvsMCTS(iterations = 500, player = -1))
hist(winner)
```



References

1. C. B. Browne et al., "A Survey of Monte Carlo Tree Search Methods," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, no. 1, pp. 1-43, March 2012, doi: 10.1109/TCIAIG.2012.2186810.
2. https://www.r-bloggers.com/2022/07/programming-a-simple-minimax-chess-engine-in-r/
3. https://towardsdatascience.com/reinforcement-learning-and-deep-reinforcement-learning-with-tic-tac-toe-588d09c41dda
4. https://en.wikipedia.org/wiki/Monte_Carlo_tree_search#Pure_Monte_Carlo_game_search
5. https://martin-ueding.de/posts/tic-tac-toe-with-monte-carlo-tree-search/
6. https://web.eecs.umich.edu/~weimerw/2014-4610/lectures/weimer-game-theory-intro.pdf
7. https://faculty.cc.gatech.edu/~surban6/2019fa-gameAI/lectures/2019-11-20%20Minimax%20MCTS%20and%20CBR.pdf
8. https://nestedsoftware.com/2019/08/07/tic-tac-toe-with-mcts-2h5k.152104.html

GitHub Repository:

https://github.com/dkinsman/stat327project