DATA 609 - Final Project

Daina Bouquin, Christophe Hunt, Christina Taylor

April 16, 2017

Textbook Projects from:

Giordano, F. R., Fox, W. P., & Horton, S. B. (2014). *A first course in mathematical modeling.* Australia: Brooks/Cole.

Textbook Part I:

[Christina's work]

Textbook part II: Chapter 5.3, Project 1

Construct and perform a Monte Carlo simulation of blackjack as per the following:

- Play 12 games (simulations) where each game lasts two decks. When the two decks are out, the round is completed using two fresh decks (that is the last round of the game). Everything is then reset for the start of the next game.
- The dealer cannot see the players cards and vice versa.
- The player wins 3 dollars with a winning hand.
- The player loses 2 dollars with a losing hand.
- No money is exchanged if there is no winner.
  - There is no winner when neither has gone bust and they stand at the same amount.
  - If the dealer goes bust, the player automatically wins.
- The dealer strategy is to stand at 17 or above.
- The player strategy is open and can be set as desired.

Create a two deck game

The below functions create the game of blackjack from a dealt card, to a hand, to a round, and to a full two-deck game. Game results are stored as a vector: 1 = dealer win; 2 = player win; 0 = no winner. The resulting vector is used to calculate player winnings/losses.

```
# Create a hand for the dealer.
deal_card <- function(d_deck,decks_2){
  flag_last_round <- 0
  if (length(d_deck) == 0) {
    flag_last_round <- 1
    d_deck <- decks_2
```

```
  }
  d_card <- sample(d_deck,1,FALSE)
  d_deck <- d_deck [!d_deck %in% d_card | duplicated(d_deck)]
  return(list(d_card,d_deck,flag_last_round))
}
dealer_hand <- function(dh_deck,decks_2) {
  dh_hand <- 0
  dh_last_round_flag <- 0
  dh_count <- 1
  while (sum(dh_hand) < 17) {
    dh_deal <- deal_card(dh_deck,decks_2)
    dh_hand[dh_count] <- dh_deal[[1]]
    dh_deck <- dh_deal[[2]]
    if (dh_deal[[3]] == 1) {dh_last_round_flag <- dh_deal[[3]]}
    dh_count <- dh_count+1 # ace = 1 if hand > 21
    if(sum(dh_hand) > 21) {
      if(11 %in% dh_hand) {
        dh_hand[match(11,dh_hand)] <- 1
      }
    }
  }
  return(list(dh_hand,dh_deck,dh_last_round_flag))
}

# Create player hand
player_hand <- function(ph_deck,decks_2,threshold_p) {
  ph_hand <- 0
  ph_count <- 1
  ph_last_round_flag <- 0
  while (sum(ph_hand) < threshold_p) {
    ph_deal <- deal_card(ph_deck,decks_2)
    ph_hand[ph_count] <- ph_deal[[1]]
    ph_deck <- ph_deal[[2]]
    if (ph_deal[[3]] == 1) {ph_last_round_flag <- ph_deal[[3]]}
    ph_count <- ph_count+1 # ace = 1 if hand > 21
    if(sum(ph_hand) > 21) {
      if(11 %in% ph_hand) {
        ph_hand[match(11,ph_hand)] <- 1
      }
    }
  }
  return(list(ph_hand,ph_deck,ph_last_round_flag))
}

# Create a game
game <- function(game_deck,decks_2,threshold_p) {
  #dealer hand
  d_result <- dealer_hand(game_deck,decks_2)
  d_hand <- d_result[[1]]
  game_deck <- d_result[[2]]
  last_round_flag_d <- d_result[[3]]
  #player hand
  p_result <- player_hand(game_deck,decks_2,threshold_p)
```

```
  p_hand <- p_result[[1]]
  game_deck <- p_result[[2]]
  last_round_flag_p <- p_result[[3]]
  if (last_round_flag_d == 1 || last_round_flag_p == 1) {
    game_last_round_flag <- 1
  } else {
    game_last_round_flag <- 0
  }
  return(list(d_hand,p_hand,game_deck,game_last_round_flag))
}

# Go through two decks- create dealer and player hand each round
play_decks_2 <- function(p2d_deck,decks_2,threshold_p) {
  round_counter <- 1
  p2d_last_round_flag <- 0
  result_list.names <- c("Dealer_Hand", "Player_Hand")
  result_list <- vector("list", length(result_list.names))
  names(result_list) <- result_list.names
  while (p2d_last_round_flag != 1) {
    round_result <- game(p2d_deck,decks_2,threshold_p)
    result_list$Dealer_Hand[round_counter] <- list(round_result[[1]])
    result_list$Player_Hand[round_counter] <- list(round_result[[2]])
    p2d_deck <- round_result[[3]]
    p2d_last_round_flag <- round_result[[4]]
    round_counter <- round_counter+1
  }
  return(result_list)
}

# Takes in list containing all hands after above running through two decks.
results_vec <- function(results_set) {
  winner <- numeric(length(results_set$Dealer_Hand))
  for (i in 1:length(results_set$Dealer_Hand)) {
    d_Hand <- sum(results_set$Dealer_Hand[[i]])
    p_Hand <- sum(results_set$Player_Hand[[i]])
    if(d_Hand < 22) {
      if(p_Hand < 22) {
        if(d_Hand > p_Hand) {
          winner[i] <- 1
        } else if (d_Hand < p_Hand) {
            winner[i] <- 2
        } else if (d_Hand == p_Hand) {
            winner[i] <- 0
        }
      } else if(p_Hand >= 22) {
          winner[i] <- 1
      }
    } else if(d_Hand >= 22){
            winner[i] <- 2 # if dealer goes bust player wins
      }
  }
  return(winner)
}
```

```r
# Calculates the player winnings with assumptions:
# player bets $2/hand
# player wins = player receives $3
# player loss = player loses $2 bet
# tie = $0 exchanged
# dealer goes bust = win for player
calc_player_win <- function(results) {
  winnings <- numeric(length(results))
  for(i in 1:length(results)) {
    if (results[i] == 0) winnings[i] <- 0
    if (results[i] == 1) winnings[i] <- -2
    if (results[i] == 2) winnings[i] <- 3
  }
  return(winnings)
}

# Play a full two deck game, calculate winnings/losses
play_deck2 <- function(pd2_deck,threshold_p){
  #create list of results per round after running two decks game
  one_set <- play_decks_2(pd2_deck,pd2_deck,threshold_p)
  #create vector for results of each hand
  outcome_vec <- results_vec(one_set)
  player_winnings <- 0
  player_winnings <- calc_player_win(outcome_vec)
  return(sum(player_winnings))
}

# Create function to run the simulation for two deck games
# Takes two decks and number of simulations as params
# Each run uses player strategies (stand at 16 to 20); dealer stands at 17
build_output <- function(out_deck, runs) {
  count_obs <- numeric(runs)
  vec16 <- numeric(runs)
  vec17 <- numeric(runs)
  vec18 <- numeric(runs)
  vec19 <- numeric(runs)
  vec20 <- numeric(runs)
  for (i in 1:runs){
    count_obs[i] <- i
    vec16[i] <- play_deck2(out_deck,16)
    vec17[i] <- play_deck2(out_deck,17)
    vec18[i] <- play_deck2(out_deck,18)
    vec19[i] <- play_deck2(out_deck,19)
    vec20[i] <- play_deck2(out_deck,20)
  }
  output <- data.frame("count_obs"= count_obs,
                       "stand_16"=vec16,"stand_17"=vec17,
                       "stand_18"=vec18,"stand_19"=vec19,
                       "stand_20"=vec20)
  return(output)
}
```

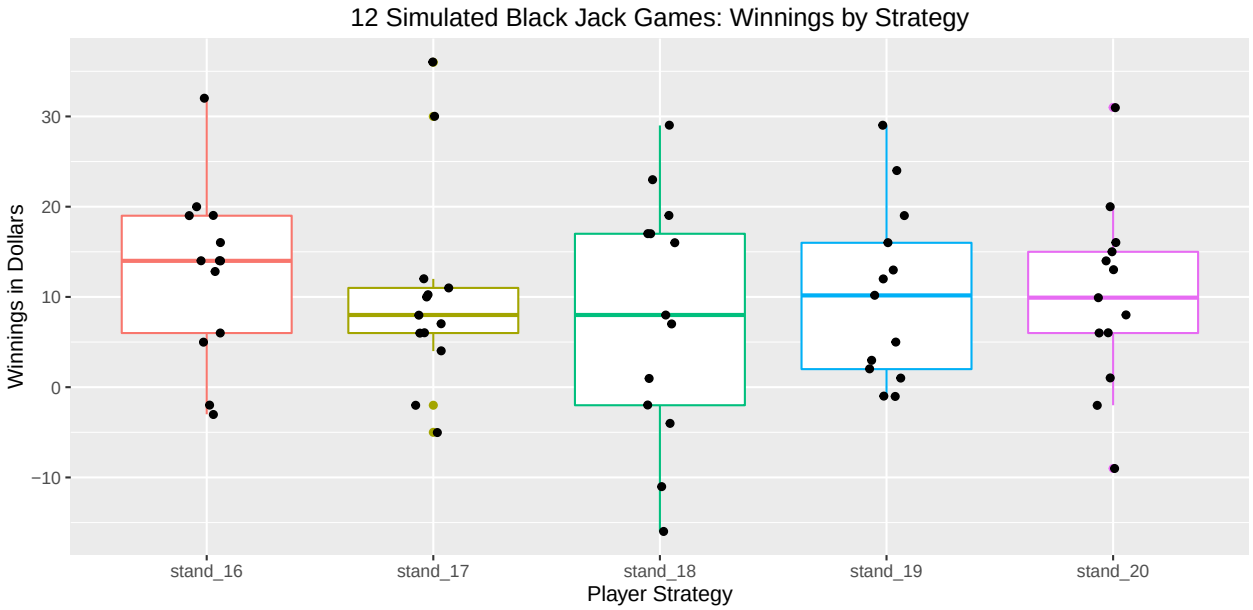
12 Simulations

```
set.seed(2345)
# Create set of two decks, simulate 12 two-deck games; run sim with diff strategies

# vector for two decks
two_deck <- c(2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,6,6,6,6,7,7,7,7,8,8,8,8,
              9,9,9,9,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,
              11,11,11,11,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,6,6,6,6,
              7,7,7,7,8,8,8,8,9,9,9,9,10,10,10,10,10,10,10,10,10,10,10,10,
              10,10,10,10,10,11,11,11,11)

# 12 games per strategy
output12 <- build_output(two_deck,12)
output12_avg <- c("stand_16"=mean(output12$stand_16),
                  "stand_17"=mean(output12$stand_17),
                  "stand_18"=mean(output12$stand_18),
                  "stand_19"=mean(output12$stand_19),
                  "stand_20"=mean(output12$stand_20))
```

Table 1: Winnings by Strategy

stand_16	12.83
stand_17	10.25
stand_18	8.00
stand_19	10.17
stand_20	9.92

If you were to only look at the average winnings across 12 games, it would appear that standing at 16 is the best option when the dealer stands at 17. The below plot though illustrates just how much variation there is in results. Simulations of 100, 500, 1000, and 1,500 rounds are made below to improve decision-making.



Increasing the Number of Simulations

```
set.seed(2345)
# 100 simulations
output100 <- build_output(two_deck,100)

# 500 simulations
output500 <- build_output(two_deck,500)

# 1000 simulations
output1000 <- build_output(two_deck,1000)

# 1500 simulations
output1500 <- build_output(two_deck,1500)
```

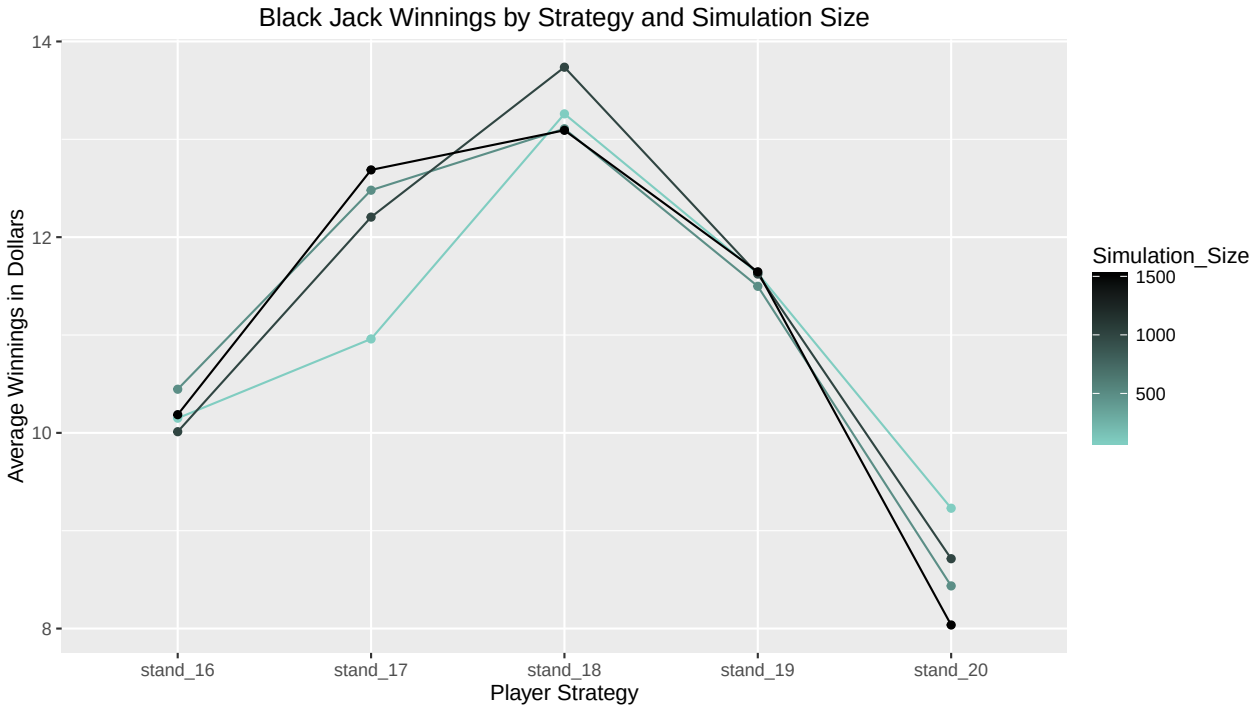


Table 2: Average Winnings by Strategy and Simulation Size

Simulation_Size	stand_16	stand_17	stand_18	stand_19	stand_20
100	10.15	10.96	13.26	11.63	9.23
500	10.45	12.48	13.11	11.50	8.44
1000	10.01	12.21	13.74	11.62	8.71
1500	10.19	12.69	13.09	11.65	8.04

As the number of simulations increases, the results converge. It appears to be that standing at 18 is the best option, as standing at 19 leaves the player with fewer winnings. Similarly, standing at 16, 17, or 20 results in fewer winnings overall. Therefore if the dealer stands at 17, the player should stand at 18.

Textbook Part III:

[Christophe's work]