# Blackjack

Nimit A. Bhanshali

6/6/2022

**Problem: Simulating a game of blackjack where Player A and Player B play against a dealer.**

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# Constructing a table of n decks of playing cards
get_n_decks <- function(n) {
  numbers <- 1:13 # Jack is 11, Queen is 12, King is 13
  shapes <- c("Spades", "Hearts", "Clubs", "Diamonds")
  single_deck <- data.frame(
    number = rep(numbers, 4),
    shape = rep(shapes, each = 13)
  )
  # assign "value" of the hand to be 10 if the card is a face card
  # aces are marked as 1 until the hands are calculated
  single_deck["value"] <- ifelse(single_deck$number > 10, 10, single_deck$number)
  # repeat `single_deck` n times and combine
  n_decks <- do.call(rbind, replicate(n, single_deck, simplify = FALSE))
  players <- c()
  num_rows <- nrow(n_decks)
  for (i in 1:num_rows){
    players <- append(players, 0)
  }
  n_decks["player"] <- players
  return(n_decks)
}

# Shuffling the n decks of playing cards
shuffle_table <- function(tbl) {
```

```
return(tbl[sample(nrow(tbl)), ])
}

# Deal cards to players and dealer
deal_cards <- function(tb1) {
  i <- 1
  while (tb1[i, 4] != 0) {
    i <- i + 1
  }
  tb1[i, 4] <- 'A'
  tb1[i + 1, 4] <- 'A'
  tb1[i + 2, 4] <- 'B'
  tb1[i + 3, 4] <- 'B'
  tb1[i + 4, 4] <- 'D'
  tb1[i + 5, 4] <- 'D'
  return(tb1)
}

# Filter hands from table
filter_hands <- function(tb1, player_name) {
  hand <- tb1$value
  i <- 1
  while (tb1[i, 4] != 0) {
    i <- i + 1
  }
   if (player_name == 'A') {
    return(hand[(i - 6):(i - 5)])
  } else if (player_name == 'B') {
    return(hand[(i - 4):(i - 3)])
  } else if (player_name == 'D') {
    return(hand[(i - 2):(i - 1)])
  }

}

# Computing a player's hand
compute_hand <- function(values) {
  # count the number of aces on hand
  num_aces <- sum(1 %in% values)
  init <- sum(values) # without any 11's
  # you want to change an ace to 11
  # if the sum is <= 11 and (&) there is one or more aces on hand
  if (init <= 11 & num_aces > 0) {
    return(init + 10) # changing an ace from 1 to 11 -> adding 10 to the hand
  } else {
    return(init)
  }
}

# Deal additional cards to dealer
deal_to_dealer <- function(tb1, dealer_hand) {
  i <- 1
  while (tb1[i, 4] != 0) {
```

```
    i <- i + 1
  }
  hand <- dealer_hand
  while (compute_hand(hand) < 17) {
    tb1[i, 4] <- 'D'
    hand <- append(hand, tb1[i, 3])
    i <- i + 1
  }
  return(tb1)
}

# Update the hand of the dealer
update_hand <- function(tb1) {
  i <- 1
  while (tb1[i, 4] != 0) {
    i <- i + 1
  }
  i <- i - 1
  hand <- numeric()
  while (tb1[i,4] != 'B') {
    hand <- append(hand, tb1[i, 3])
    i <- i - 1
  }
  return(hand)
}
```

```
set.seed(178)
m <- 5
n <- 10
N <- 10000 # Number of times the first round is simulated

a_win <- 0 # Number of times Player A wins
a_tie <- 0 # Number of times Player A ties
a_lose <- 0 # Number of times Player A loses

b_win <- 0 # Number of times Player B wins
b_tie <- 0 # Number of times Player B ties
b_lose <- 0 # Number of times Player B loses

aunionb <- 0 # Union of event A1 and B1
aintbnot <- 0 # Intersection of event A1 and B1 Complement
aaunionbb <- 0 # Union of events A1, B1, a1, b1
dealer_earnings <- 0 # Dealer's earnings

playing_cards_table <- get_n_decks(n)

for (i in 1:N) {
  wina <- 0 # Stores whether Player A won the round
  winb <- 0 # Stores whether Player B won the round
  tiea <- 0 # Stores whether Player A ties the round
  tieb <- 0 # Stores whether Player B ties the round

  shuffled_table <- shuffle_table(playing_cards_table)
```

```
deal <- deal_cards(shuffled_table)

player_a_hand <- filter_hands(deal, 'A')
a_value <- compute_hand(player_a_hand)

player_b_hand <- filter_hands(deal, 'B')
b_value <- compute_hand(player_b_hand)

dealer_hand <- filter_hands(deal, 'D')
dealer_value <- compute_hand(dealer_hand)
deal <- deal_to_dealer(deal, dealer_hand)
dealer_hand <- update_hand(deal)
dealer_value <- compute_hand(dealer_hand)

# Player A vs Dealer
if ((a_value == 21) | (a_value > dealer_value) | (dealer_value > 21)){
  a_win <- a_win + 1
  wina <- 1
  dealer_earnings <- dealer_earnings - m
} else if ((a_value == dealer_value) && (dealer_value != 21)) {
  a_tie <- a_tie + 1
  tiea <- 1
} else if ((a_value < dealer_value) && (dealer_value <= 21)) {
  a_lose <- a_lose + 1
  dealer_earnings <- dealer_earnings + m
}

# Player B vs Dealer
if ((b_value == 21) | (b_value > dealer_value) | (dealer_value > 21)){
  b_win <- b_win + 1
  winb <- 1
  dealer_earnings <- dealer_earnings - m
} else if ((b_value == dealer_value) && (dealer_value != 21)) {
  b_tie <- b_tie + 1
  tieb <- 1
} else if ((b_value < dealer_value) && (dealer_value <= 21)) {
  b_lose <- b_lose + 1
  dealer_earnings <- dealer_earnings + m
}

# Player A or Player B win
if ((wina == 1) | (winb == 1)){
  aunionb <- aunionb + 1
}

# Player A wins, Player B doesn't win
if ((wina == 1) && (winb == 0)){
  aintbnot <- aintbnot + 1
}

# Player A or Player B either tie or win
if ((wina == 1) | (winb == 1) | (tiea == 1) | (tieb == 1)){
 aaunionbb <- aaunionbb + 1
```

```
  }
}

print("a. P(A1) = ")
```

```
## [1] "a. P(A1) = "
```

```
(a_win/N)
```

```
## [1] 0.3922
```

```
print("b. P(A1 union B1) = ")
```

```
## [1] "b. P(A1 union B1) = "
```

```
(aunionb/N)
```

```
## [1] 0.4769
```

```
print("c. P[A1 intersection (B1 Complement)] = ")
```

```
## [1] "c. P[A1 intersection (B1 Complement)] = "
```

```
(aintbnot/N)
```

```
## [1] 0.0853
```

```
print("d. P(A1 union B1 union a1 union b1) = ")
```

```
## [1] "d. P(A1 union B1 union a1 union b1) = "
```

```
(aaunionbb/N)
```

```
## [1] 0.5478
```

```
print("e. Expected value of the dealer's earnings after the first round: ")
```

```
## [1] "e. Expected value of the dealer's earnings after the first round: "
```

```
(dealer_earnings/N)
```

```
## [1] 1.718
```

```r
set.seed(178)
m <- 5
n <- 10
rounds <- 5 # Number of rounds played
N <- 10000 # Number of simulations
playing_cards_table <- get_n_decks(n)
round_earnings <- matrix(0, N, 5) # Matrix that stores the cumulative earnings
# of Player A after each round over N simulations


for(i in 1:N) {

  shuffled_table <- shuffle_table(playing_cards_table)
  deal <- shuffled_table
  earnings <- 0 # Player A cumulative winning amount counter per simulation

  for (j in 1:rounds){

    deal <- deal_cards(deal)
    player_a_hand <- filter_hands(deal, 'A')
    a_value <- compute_hand(player_a_hand)

    player_b_hand <- filter_hands(deal, 'B')
    b_value <- compute_hand(player_b_hand)

    dealer_hand <- filter_hands(deal, 'D')
    dealer_value <- compute_hand(dealer_hand)
    deal <- deal_to_dealer(deal, dealer_hand)
    dealer_hand <- update_hand(deal)
    dealer_value <- compute_hand(dealer_hand)

    # Player A vs Dealer
    if ((a_value == 21) | (a_value > dealer_value) | (dealer_value > 21)){
      earnings <- earnings + m
    } else if ((a_value == dealer_value) && (dealer_value != 21)) {
      earnings <- earnings
    } else if ((a_value < dealer_value) && (dealer_value <= 21)) {
      earnings <- earnings - m
    }

    round_earnings[i, j] <- earnings # Stores the cumulative earnings after the
    # jth round into the matrix

  }

}

total_earnings <- c(round_earnings)
round1_earnings <- total_earnings[1:N]
round2_earnings <- total_earnings[(N + 1):(2 * N)]
round3_earnings <- total_earnings[((2 * N) + 1):(3 * N)]
round4_earnings <- total_earnings[((3 * N) + 1):(4 * N)]
round5_earnings <- total_earnings[((4 * N) + 1):(5 * N)]
```

```
print("Expected cumulative winning dollar amount after first round for Player A")
```

```
## [1] "Expected cumulative winning dollar amount after first round for Player A"
```

```
sum(round1_earnings) / N
```

```
## [1] -0.8655
```

```
print("Expected cumulative winning dollar amount after second round for Player A")
```

```
## [1] "Expected cumulative winning dollar amount after second round for Player A"
```

```
sum(round2_earnings) / N
```

```
## [1] -1.86
```

```
print("Expected cumulative winning dollar amount after third round for Player A")
```

```
## [1] "Expected cumulative winning dollar amount after third round for Player A"
```

```
sum(round3_earnings) / N
```

```
## [1] -2.666
```

```
print("Expected cumulative winning dollar amount after fourth round for Player A")
```

```
## [1] "Expected cumulative winning dollar amount after fourth round for Player A"
```

```
sum(round4_earnings) / N
```

```
## [1] -3.5505
```

```
print("Expected cumulative winning dollar amount after fifth round for Player A")
```

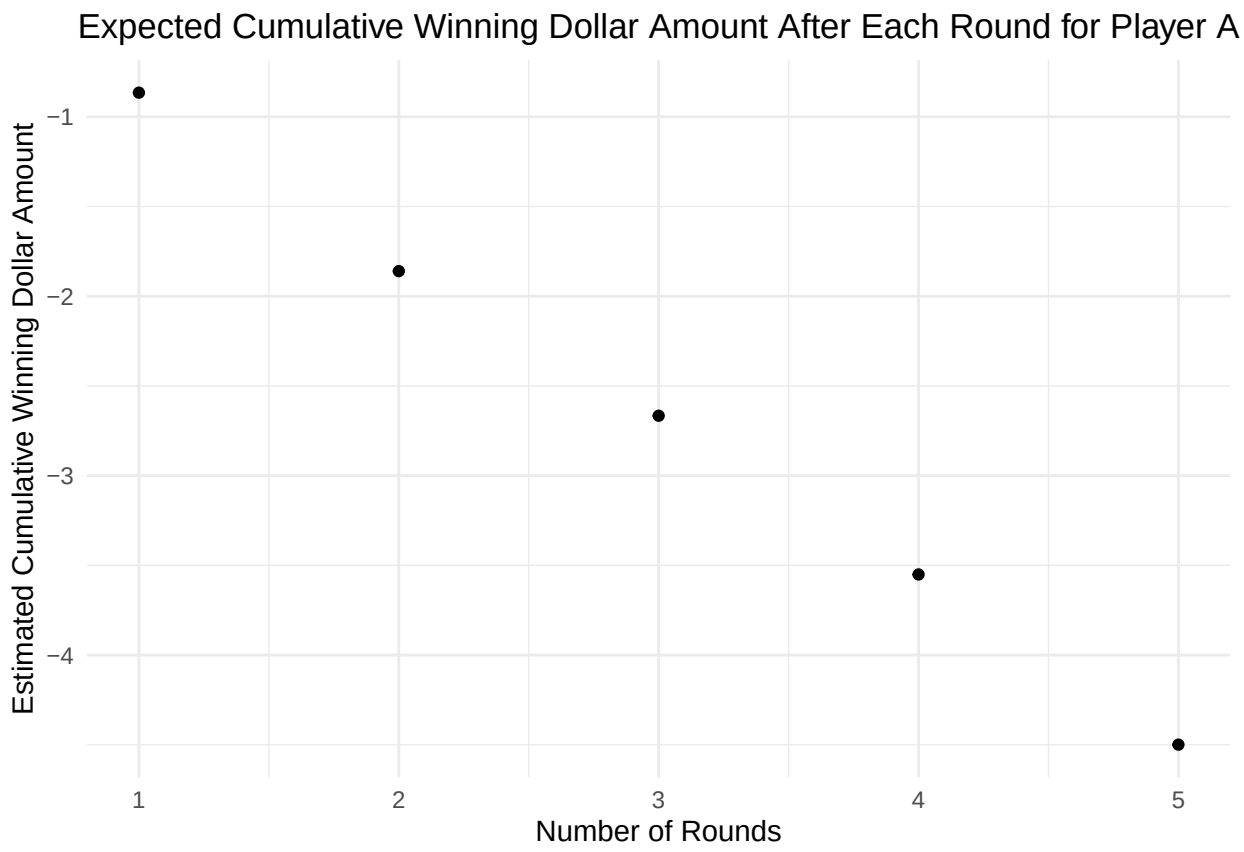
```
## [1] "Expected cumulative winning dollar amount after fifth round for Player A"
```

```
sum(round5_earnings) / N
```

```
## [1] -4.499
```

```r
library(ggplot2)

ggplot(, (aes(x = rounds))) +
  ggtitle("Expected Cumulative Winning Dollar Amount After Each Round for Player A") +
  theme_minimal() +
  # From ggplot2 documentation: https://ggplot2.tidyverse.org/reference/element.html
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_point(aes(x = 1, y = sum(round1_earnings) / N)) +
  geom_point(aes(x = 2, y = sum(round2_earnings) / N)) +
  geom_point(aes(x = 3, y = sum(round3_earnings) / N)) +
  geom_point(aes(x = 4, y = sum(round4_earnings) / N)) +
  geom_point(aes(x = 5, y = sum(round5_earnings) / N)) +
  labs(x = "Number of Rounds", y = "Estimated Cumulative Winning Dollar Amount")
```




```r
set.seed(179)
m <- 5
n <- 10
N <- 10000 # Number of times the first round is simulated

a_win <- 0 # Number of times Player A wins
a_tie <- 0 # Number of times Player A ties
a_lose <- 0 # Number of times Player A loses

earnings <- 0 # Player A's earnings
```

```r
playing_cards_table <- get_n_decks(n)

for (i in 1:N) {
  player_a_hand <- c()

  shuffled_table <- shuffle_table(playing_cards_table)
  player_a_hand[1] <- shuffled_table[1, 3]

  shuffled_table
  player_a_hand

  shuffled_table <- shuffle_table(playing_cards_table)
  player_a_hand[2] <- shuffled_table[1, 3]

  shuffled_table
  player_a_hand

  dealer_hand <- c()
  j <- 1

  while (compute_hand(dealer_hand) < 17) {
    shuffled_table <- shuffle_table(playing_cards_table)
    dealer_hand[j] <- shuffled_table[1, 3]
    j <- j + 1
  }

  dealer_hand

  a_value <- compute_hand(player_a_hand)

  dealer_value <- compute_hand(dealer_hand)

  # Player A vs Dealer
  if ((a_value == 21) | (a_value > dealer_value) | (dealer_value > 21)){
    a_win <- a_win + 1
    earnings <- earnings + m
  } else if ((a_value == dealer_value) && (dealer_value != 21)) {
    a_tie <- a_tie + 1
    earnings <- earnings
  } else if ((a_value < dealer_value) && (dealer_value <= 21)) {
    a_lose <- a_lose + 1
    earnings <- earnings -m
  }
}

print("a. P(X_1 = 0) = ")
```

```
## [1] "a. P(X_1 = 0) = "
```

```r
a_tie / N
```

```
## [1] 0.0474
```

```r
print("b. P(X_1 = 5) = ")
```

```
## [1] "b. P(X_1 = 5) = "
```

```r
a_win / N
```

```
## [1] 0.3856
```