Stats 102A - Homework 2 - Output File

Miles Chen (example)

To receive full credit the functions you write must pass all tests. We may conduct further tests that are not included on this page as well.

Academic Integrity Statement

By including this statement, I, **Yuer Tang**, declare that all of the work in this assignment is my own original work.

At no time did I use an AI tool to generate or debug my solutions.

At no time did I look at the code of other students nor did I search for code solutions.

I understand that forbidden AI usage or plagiarism on any single part of this assignment will result in a 0 for the entire assignment.

```r
source("102a_hw_02_script_Yuer_Tang.R")  # edit with your file name
```

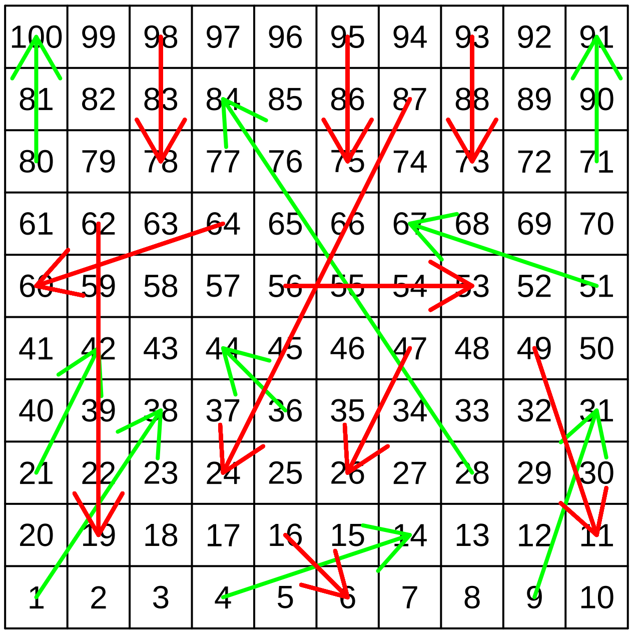
Part 1: Board representation

Create a single list object called `board` where you store the features of the game board in R.

```
board <- list(
  n_row = 10,
  n_col = 10,
  ladders = data.frame(
    start = c(1, 4, 9, 21, 28, 36, 51, 71, 80),
    end   = c(38, 14, 31, 42, 84, 44, 67, 91, 100)
  ),
  chutes = data.frame(
    start = c(16, 47, 49, 56, 62, 64, 87, 93, 95, 98),
    end   = c(6, 26, 11, 53, 19, 60, 24, 73, 75, 78)
  )
)
```

Part 2: Plot of Game board

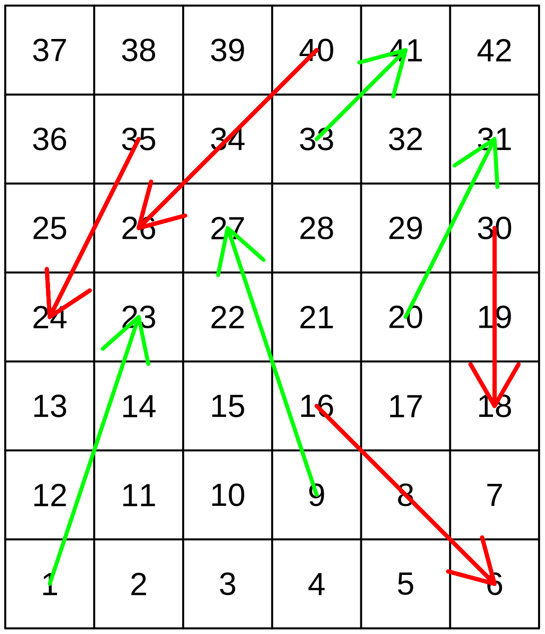
```r
# par() should help the plot be more visible. you can adjust this as necessary
par(mar = c(0, 0, 0, 0))
show_board(board)
```



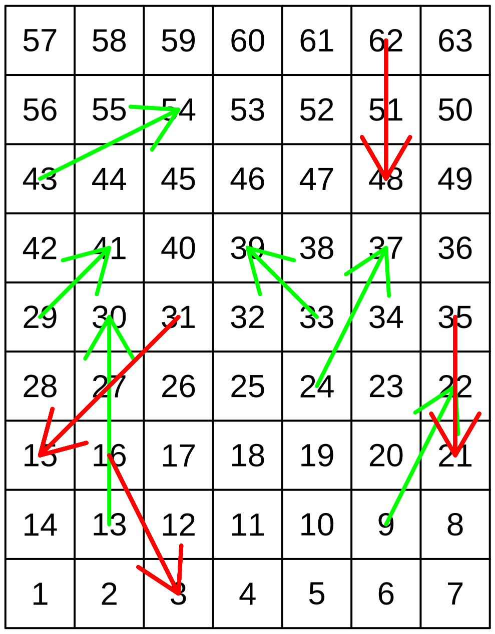
Part 3: Miniboards

Create the `miniboard` objects and plots.

```
miniboard1 <- list(
  n_row = 7,
  n_col = 6,
  ladders = data.frame(
    start = c(1,9,20,33),
    end   = c(23,27,31,41)
  ),
  chutes = data.frame(
    start = c(16,30,35,40),
    end   = c(6,18,24,26)
  ))
par(mar = c(0, 0, 0, 0))
show_board(miniboard1)
```



```
miniboard2 <- list(
  n_row = 9,
  n_col = 7,
  ladders = data.frame(
    start = c(9,13,24,29,33,43),
    end   = c(22,30,37,41,39,54)
  ),
  chutes = data.frame(
    start = c(16,31,35,62),
    end   = c(3,15,21,48)
  ))
par(mar = c(0, 0, 0, 0))
show_board(miniboard2)
```



```
miniboard3 <- list(
  n_row = 9,
  n_col = 8,
  ladders = data.frame(
    start = c(),
    end   = c()
  ),
  chutes = data.frame(
    start = c(),
    end   = c()
  ))
par(mar = c(0, 0, 0, 0))
show_board(miniboard3)
```

65	66	67	68	69	70	71	72
64	63	62	61	60	59	58	57
49	50	51	52	53	54	55	56
48	47	46	45	44	43	42	41
33	34	35	36	37	38	39	40
32	31	30	29	28	27	26	25
17	18	19	20	21	22	23	24
16	15	14	13	12	11	10	9
1	2	3	4	5	6	7	8

Part 4: Verbose output of one single player game

```r
set.seed(5)
play_solo(board, verbose = TRUE)
```

```
Turn 1
Start at 0
Spinner: 2
Turn ends at: 2
Turn 2
Start at 2
Spinner: 3
Turn ends at: 5
Turn 3
Start at 5
Spinner: 1
Turn ends at: 6
Turn 4
Start at 6
Spinner: 3
Ladder!Turn ends at: 31
Turn 5
Start at 31
Spinner: 1
Turn ends at: 32
Turn 6
Start at 32
Spinner: 1
Turn ends at: 33
Turn 7
Start at 33
Spinner: 5
Turn ends at: 38
Turn 8
Start at 38
Spinner: 6
Turn ends at: 44
Turn 9
Start at 44
Spinner: 3
Chutes!Turn ends at: 26
```

```
Turn 10
Start at 26
Spinner: 3
Turn ends at: 29
Turn 11
Start at 29
Spinner: 6
Turn ends at: 35
Turn 12
Start at 35
Spinner: 2
Turn ends at: 37
Turn 13
Start at 37
Spinner: 5
Turn ends at: 42
Turn 14
Start at 42
Spinner: 4
Turn ends at: 46
Turn 15
Start at 46
Spinner: 2
Turn ends at: 48
Turn 16
Start at 48
Spinner: 5
Turn ends at: 53
Turn 17
Start at 53
Spinner: 3
Chutes!Turn ends at: 53
Turn 18
Start at 53
Spinner: 1
Turn ends at: 54
Turn 19
Start at 54
Spinner: 6
Turn ends at: 60
Turn 20
Start at 60
Spinner: 4
```

```
Chutes!Turn ends at: 60
Turn 21
Start at 60
Spinner: 3
Turn ends at: 63
Turn 22
Start at 63
Spinner: 2
Turn ends at: 65
Turn 23
Start at 65
Spinner: 5
Turn ends at: 70
Turn 24
Start at 70
Spinner: 2
Turn ends at: 72
Turn 25
Start at 72
Spinner: 2
Turn ends at: 74
Turn 26
Start at 74
Spinner: 3
Turn ends at: 77
Turn 27
Start at 77
Spinner: 1
Turn ends at: 78
Turn 28
Start at 78
Spinner: 2
Ladder!Turn ends at: 100

$turns
[1] 28

$chute_tally
 [1] 0 1 0 1 0 1 0 0 0 0

$ladder_tally
[1] 0 0 1 0 0 0 0 0 1
```

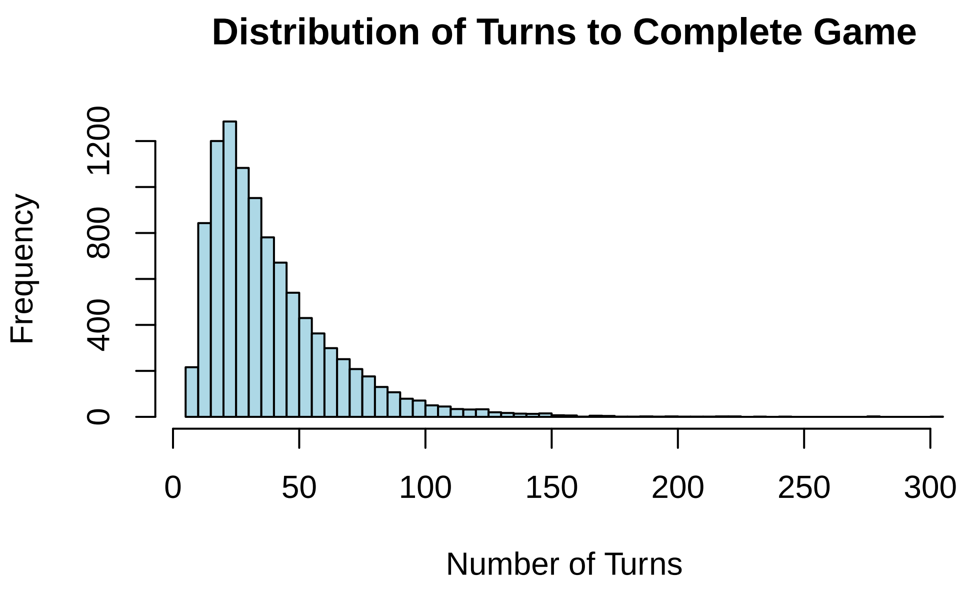
```
$move_log
 [1]   2   5   6  31  32  33  38  44  26  29  35  37  42  46  48  53  53  54  60
[20]  60  63  65  70  72  74  77  78 100
```

Part 5: Monte Carlo Simulation Study

```r
# Run 10,000 simulations
n_sims <- 10000
# To record the 10,000 in one matrix, each row is 1 simulation and column would be pairs of
all_turns <- numeric(n_sims)
all_chute_tallies <- matrix(0, nrow = n_sims, ncol = nrow(board$chutes))
all_ladder_tallies <- matrix(0, nrow = n_sims, ncol = nrow(board$ladders))

for (i in 1:n_sims) {
  result <- play_solo(board, verbose = FALSE)
  all_turns[i] <- result$turns
  all_chute_tallies[i, ] <- result$chute_tally
  all_ladder_tallies[i, ] <- result$ladder_tally
}
```

- Create a histogram (breaks = 50) of the turns.

```r
hist(all_turns, breaks = 50,
     main = "Distribution of Turns to Complete Game",
     xlab = "Number of Turns",
     col = "lightblue")
```



- Find the minimum number of turns. How many times out of 10,000 did a game finish with the minimum number of turns?

```r
min_turns <- min(all_turns)
min_turns
```

```
[1] 7
```

```r
# How many games finished with minimum turns?
sum(all_turns == min_turns)
```

```
[1] 10
```

- Find the maximum number of turns.

```r
max(all_turns)
```

```
[1] 301
```

- What is the median number of turns?

```r
median(all_turns)
```

```
[1] 32
```

- What is the mean number of turns?

```r
mean(all_turns)
```

```
[1] 39.3484
```

- What proportion of games take 100 or more turns to complete?

```r
sum(all_turns >= 100) / n_sims
```

```
[1] 0.0329
```

- What proportion of games take 10 or fewer turns to complete?

```r
sum(all_turns <= 10) /n_sims
```

```
[1] 0.0216
```

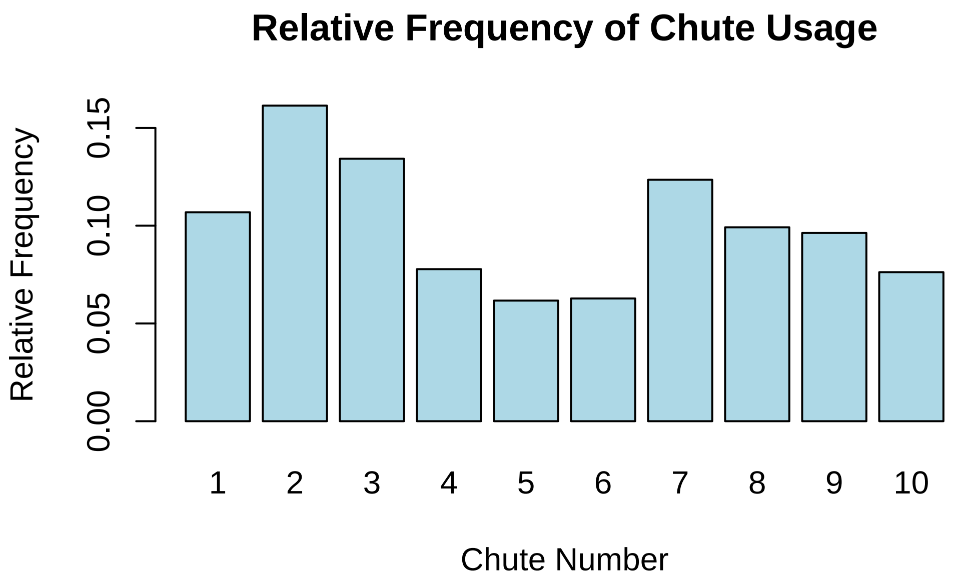
- What proportion of games utilize ladder 9 (the shortcut to win on space 80)?

```r
sum(all_ladder_tallies[, 9] > 0)/n_sims
```

```
[1] 0.4814
```

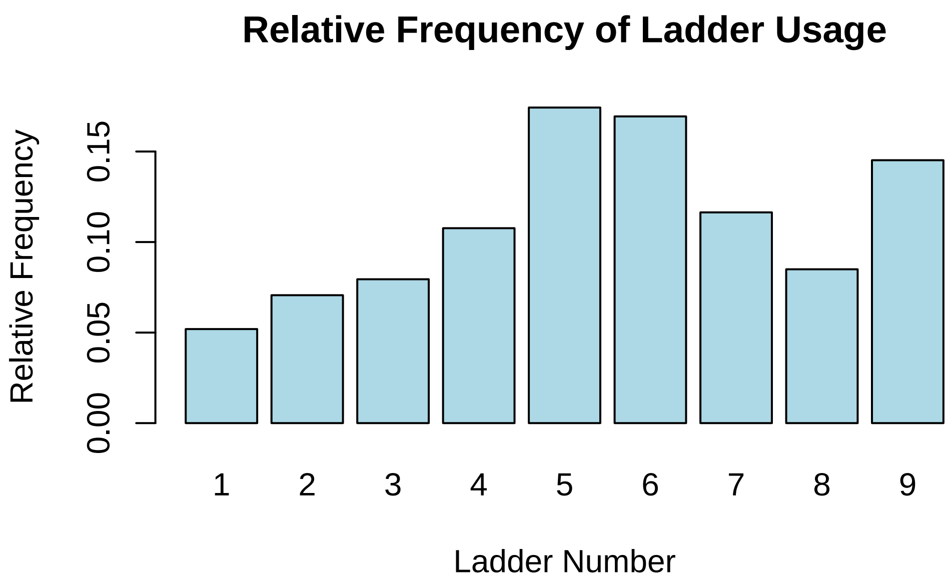
- Create a barplot of the relative frequency of how often each chute is utilized. (Number the chutes in order based on their starting square. The chute with lowest starting number, 16 to 6, is chute 1. The chute going from 98 to 78 is chute 10.)

```r
chute_totals <- colSums(all_chute_tallies)
chute_relative_freq <- chute_totals / sum(chute_totals)

barplot(chute_relative_freq,
        names.arg = 1:10,
        main = "Relative Frequency of Chute Usage",
        xlab = "Chute Number",
        ylab = "Relative Frequency",
        col = "lightblue")
```



- Create a barplot of the relative frequency of how often each ladder is utilized. (Number the ladders in order based on their starting square. The ladder with lowest starting number, 1 to 38, is ladder 1. The Ladder going from 80 to 100 is ladder 9.)

```r
ladder_totals <- colSums(all_ladder_tallies)
ladder_relative_freq <- ladder_totals / sum(ladder_totals)
barplot(ladder_relative_freq,
        names.arg = 1:9,
        main = "Relative Frequency of Ladder Usage",
        xlab = "Ladder Number",
        ylab = "Relative Frequency",
        col = "lightblue")
```



Optional Extra Credit

1. Make a function that input current positions and board and will give me the set of next possible positions
2. Then get move1 set, move2 set, and move3 set positions
3. Plot heatmap

```r
get_next_positions <- function(current_positions, board) {

  winning_number <- board$n_row * board$n_col
  next_positions <- c()

  for (pawn in current_positions) {
    # we want to know all possibilities, so use the for function at here
    for (spin in 1:6) {
      new_pawn <- pawn + spin
      if (new_pawn > winning_number) {
        new_pawn <- pawn
      }

      ladder_idx <- which(board$ladders$start == new_pawn)
      if (length(ladder_idx) > 0) {
        new_pawn <- board$ladders$end[ladder_idx]
      }

      chute_idx <- which(board$chutes$start == new_pawn)
      if (length(chute_idx) > 0) {
        new_pawn <- board$chutes$end[chute_idx]
      }

      next_positions <- c(next_positions, new_pawn)
    }
  }

  return(next_positions)
}
```

```r
move_1 <- get_next_positions(c(0), board)
cat("move1:",move_1, "\n")
```

```
move1: 38 2 3 14 5 6
```

```r
move_2 <- get_next_positions(move_1, board)
cat("move2:",move_2,"\n")
```

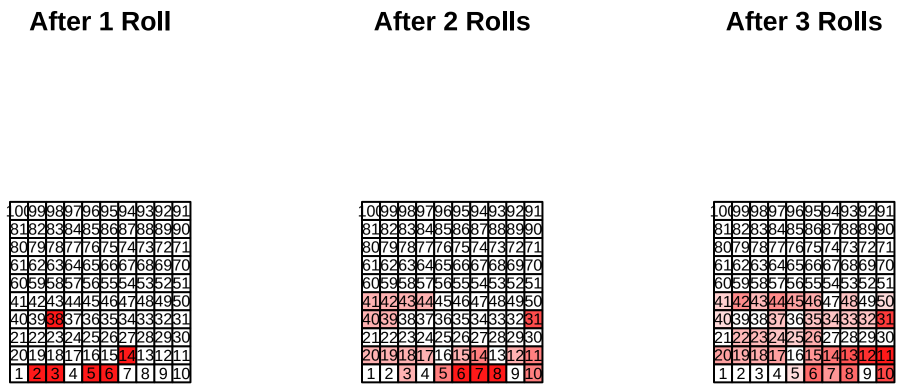
```
move2: 39 40 41 42 43 44 3 14 5 6 7 8 14 5 6 7 8 31 15 6 17 18 19 20 6 7 8 31 10 11 7 8 31 1
```

```r
move_3 <- get_next_positions(move_2, board)
cat("move3",move_3, "\n")
```

```
move3 40 41 42 43 44 45 41 42 43 44 45 46 42 43 44 45 46 26 43 44 45 46 26 48 44 45 46 26 48
```

```r
plot_heatmap <- function(positions, board, title) {

  n_row <- board$n_row
  n_col <- board$n_col
  winning_number <- n_row * n_col
  # Still, build the board
  grid <- matrix(NA, nrow = n_row, ncol = n_col)
  for (row in 1:n_row) {
    if (row %% 2 == 1) {
      start <- (row - 1) * n_col + 1
      grid[row, ] <- start:(start + n_col - 1)
    } else {
      start <- row * n_col
      grid[row, ] <- start:(start - n_col + 1)
    }
  }
  # Logic is to records the vector of appearances for each move and turn it to frequence
   counts <- rep(0, winning_number + 1)
    for (pos in positions) {
      counts[pos + 1] <- counts[pos + 1] + 1
    }

    frequencies <- counts / length(positions)
  # Then put that frequency to the corresponding value on to a matrix
    prob_matrix <- matrix(0, nrow = n_row, ncol = n_col)
    for (row in 1:n_row) {
      for (col in 1:n_col) {
        number <- grid[row, col]
        prob_matrix[row, col] <- frequencies[number+ 1]
      }
    }


  plot.new()
  plot.window(xlim = c(0, n_col), ylim = c(0, n_row), asp = 1)
  title(main = title)

  for (row in 1:n_row) {
    for (col in 1:n_col) {
      prob <- prob_matrix[row, col]

      if (prob > 0) {
        color <- rgb(1, 0, 0, alpha = prob / max(prob_matrix) * 0.8 + 0.1)
      } else {
        color <- "white"
      }

      rect(col - 1, row - 1, col, row, col = color, border = "black")

      text(col - 0.5, row - 0.5, labels = grid[row, col], cex = 0.7)
    }
  }
}
```

```r
par(mfrow = c(1, 3))
plot_heatmap(move_1, board, "After 1 Roll")
plot_heatmap(move_2, board, "After 2 Rolls")
plot_heatmap(move_3, board, "After 3 Rolls")
```


After 1 Roll
After 2 Rolls
After 3 Rolls