Chutes and Ladder Project

Yuki Kitamura

This is a simulation of the children's board game, Chutes and Ladders. Rules: Players spin a spinner with number 1 through 6. Players move their piece that many spaces. Some spaces have ladders, and the player moves to a new location on the board. If you land at the bottom of a ladder, you move to the top of the ladder. Some spaces have chutes (slides) which causes the player to move backward. If you land at the top of a chute, you move to the bottom of the chute. The start/end of a chute/ladder has a picture on the given space (there is no ambiguity). The first player to land exactly on space 100 wins the game.

```
source("Chutes and Ladder Script.R")  # edit with your file name
```

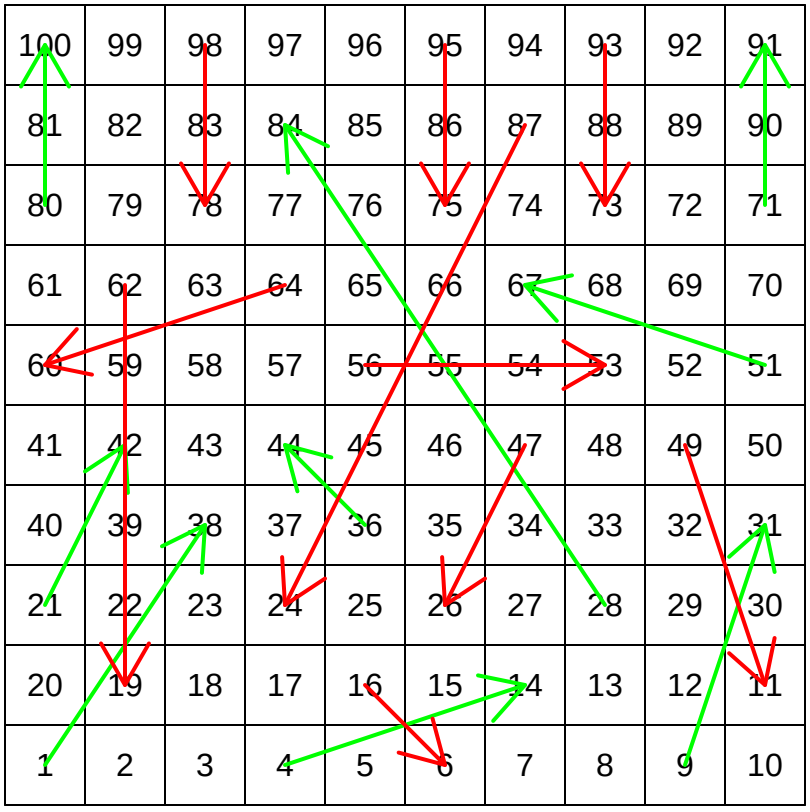
Part 1: Board representation

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

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

Part 2: Plot of Game board

```
# using par() to help the plot be more visible
par(mar = c(0, 0, 0, 0))
show_board(board)
```

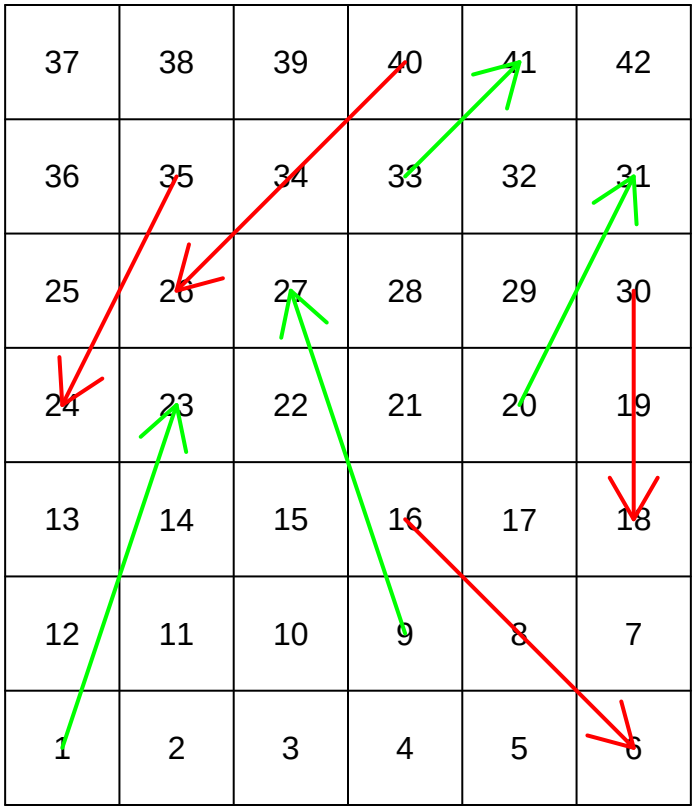



```
# Ladders are shown as green arrows
# Chutes are shown as red arrows
```

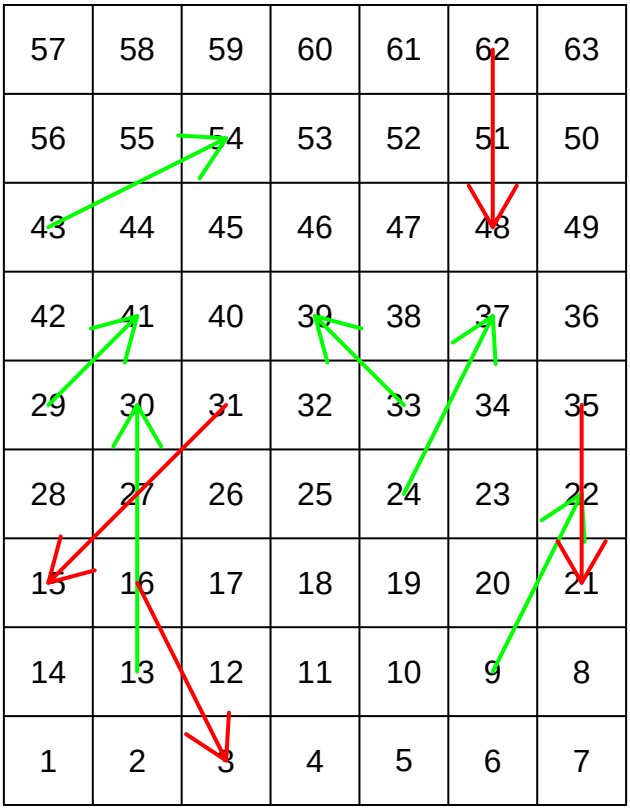
Part 3: Miniboards

Create the `miniboard` objects and plots. # code demonstration for different board sizes

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




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




```
miniboard3 <- list(dimention = c(8, 9),
                   ladders = list(),
                   chutes = list())
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

```
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
```

```
## Landed on: 9
## 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
## Landed on: 47
## Chute!
## 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
## Landed on: 56
## Chute!
## 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
## Landed on: 64
## Chute!
## 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
```

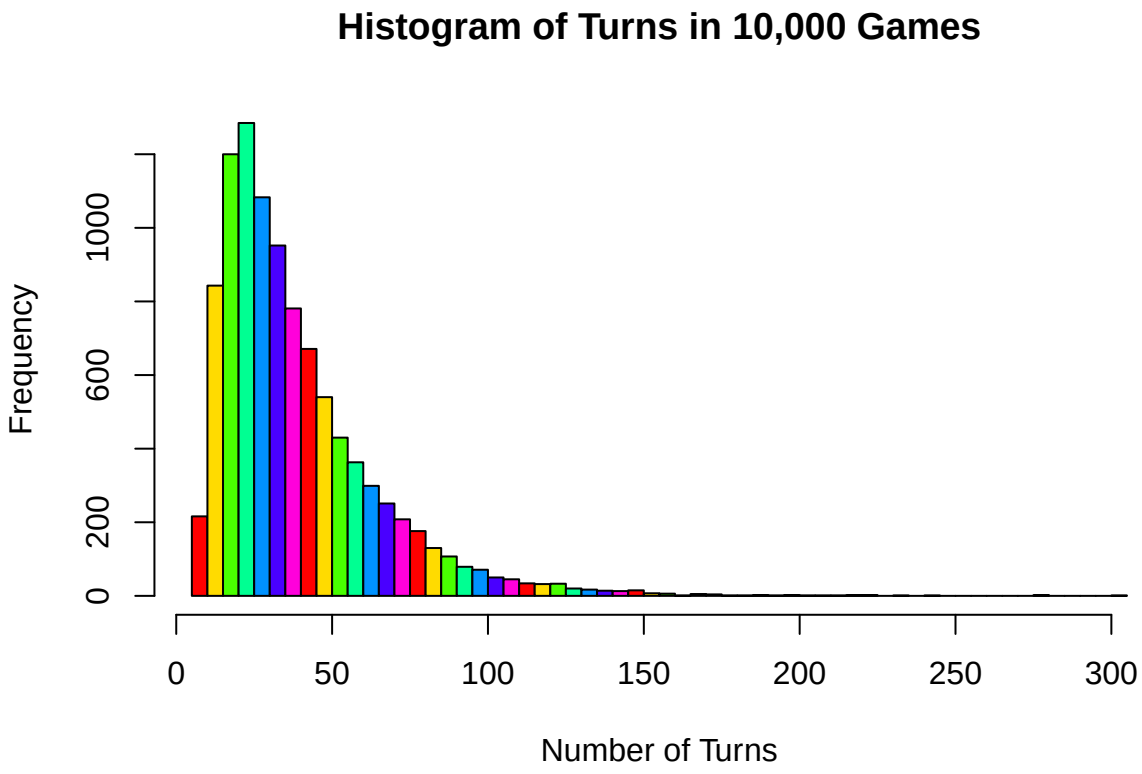
```
## Turn ends at: 72
##
## Turn 25
## Start at 72
## Spinner: 2
## Turn ends at: 74
##
## Turn 26
## Start at 74
## Spinner: 3
## Turn ends at: 77
##
## Turn 27
## Start at 77
## Spinner: 1
## Turn ends at: 78
##
## Turn 28
## Start at 78
## Spinner: 2
## Landed on: 80
## Ladder!
## Turn ends at: 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

```
turn_vec <- numeric(10000)
chute_total <- numeric(length(board[[3]]))
ladder_total <- numeric(length(board[[2]]))

# Simulate a game for 10,000 turns
for (i in 1:10000) {
  x <- play_solo(board)
  turn_vec[i] <- x$turns
  chute_total <- chute_total + x$chute_tally
  ladder_total <- ladder_total + x$ladder_tally
}
```

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

```r
hist(turn_vec, breaks = 50, col = rainbow(7),
     main = "Histogram of Turns in 10,000 Games",
     xlab = "Number of Turns")
```



- 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(turn_vec)
```

```
## [1] 7
```

```r
sum(turn_vec == min(turn_vec))
```

```
## [1] 10
```

- Find the maximum number of turns.

```r
max(turn_vec)
```

```
## [1] 301
```

- What is the median number of turns?

```
median(turn_vec)
```

```
## [1] 32
```

- What is the mean number of turns?

```
mean(turn_vec)
```

```
## [1] 39.3484
```

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

```
sum(turn_vec >= 100) / 10000
```

```
## [1] 0.0329
```

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

```
sum(turn_vec <= 10) / 10000
```

```
## [1] 0.0216
```

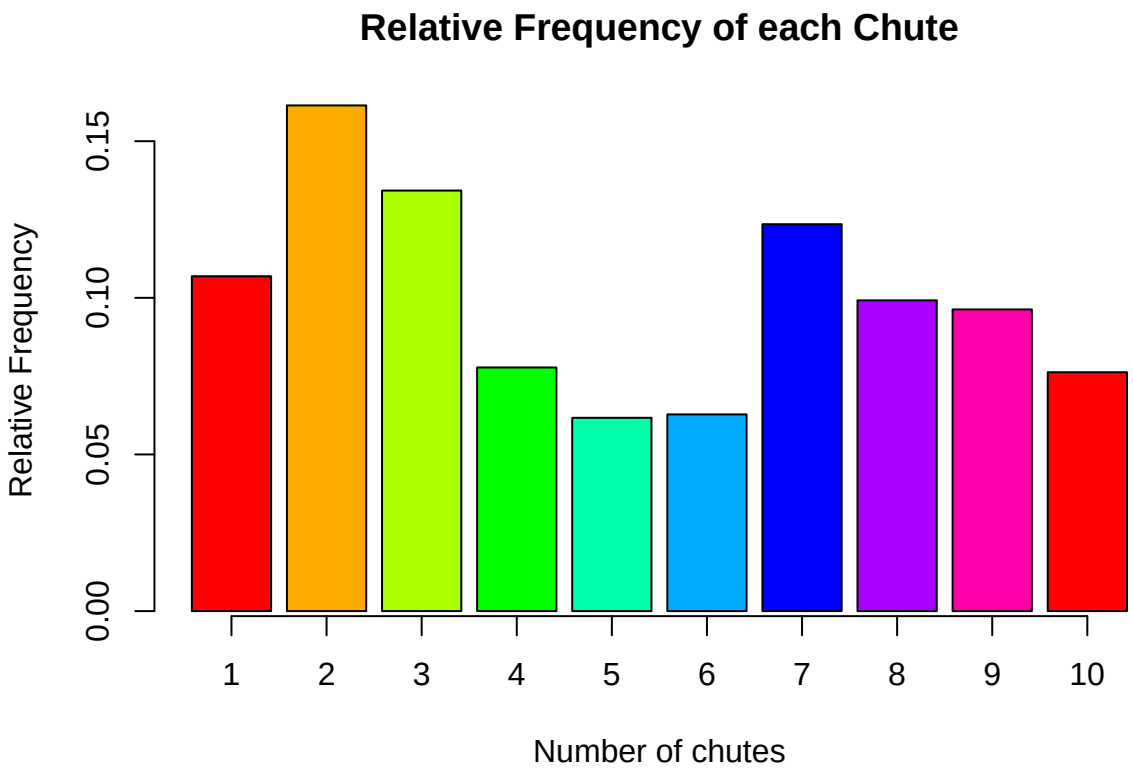
- What proportion of games utilize ladder 9 (the shortcut to win on space 80)?

```
ladder_total[9] / sum(ladder_total)
```

```
## [1] 0.1452143
```

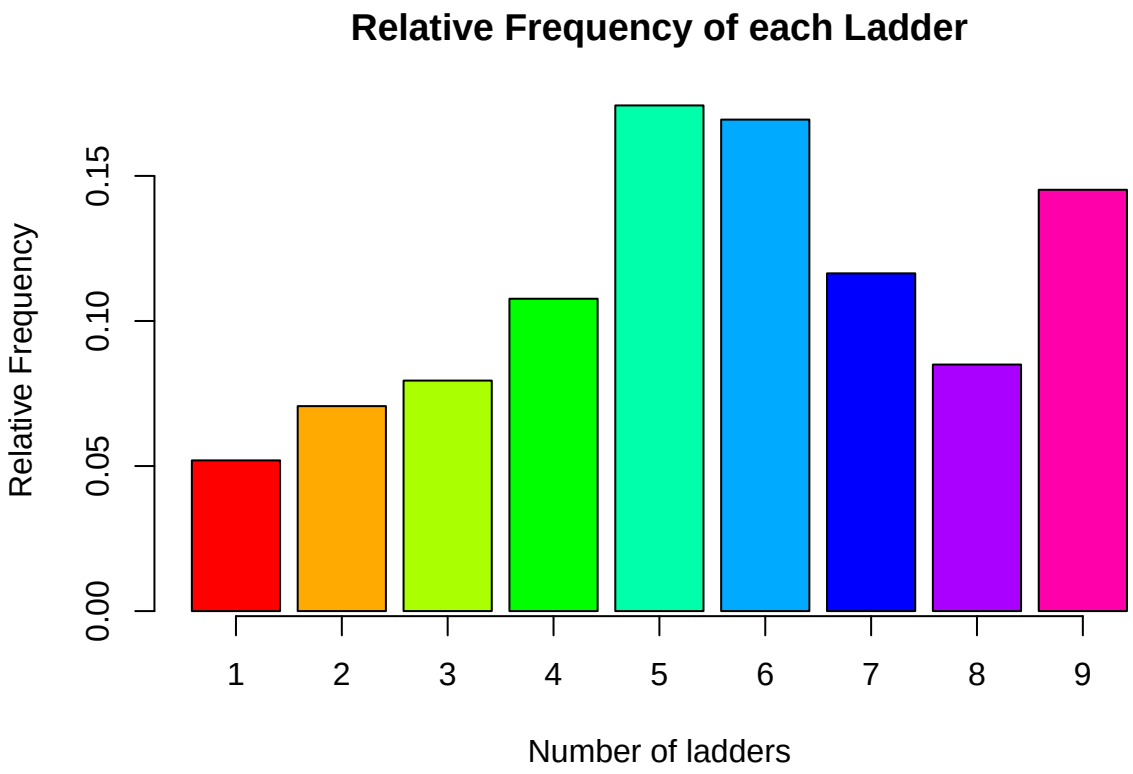
- Create a barplot of the relative frequency of how often each chute is utilized.

```
barplot(chute_total / sum(chute_total), col = rainbow(9),
        xlab = "Number of chutes", ylab = "Relative Frequency")
title("Relative Frequency of each Chute")
axis(1, seq(0.7, 12.3, 1.2), 1:10)
```



- Create a barplot of the relative frequency of how often each ladder is utilized.

```
barplot(ladder_total / sum(ladder_total), col = rainbow(9),
        xlab = "Number of ladders", ylab = "Relative Frequency")
title("Relative Frequency of each Ladder")
axis(1, seq(0.7, 11.1, 1.2), 1:9)
```



Part 6: Heat Maps of the game board for each rolls

On the datagenetics web page, under the section "Transition Matrix" there are plots showing a 'heat map' of which squares are landed on after one turn, after two turns, after three rolls, etc. Recreated this plot in own codes.

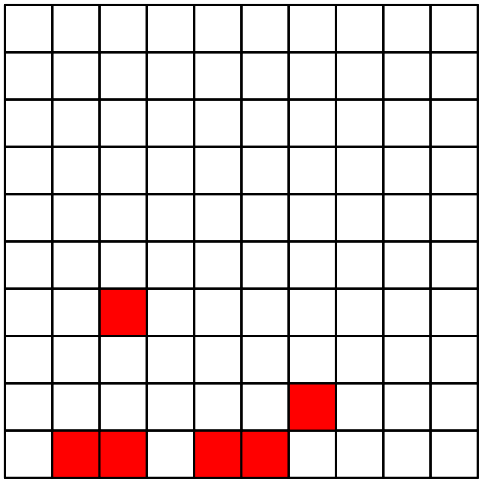
```
## Numbering axis for each box
label <- 1:100
# Reorder numbers
  i <- 1
  while(i < 10) {
    label[seq((i * 10) + 1, (i + 1)* 10, 1)] <- rev(seq((i * 10) + 1, (i + 1)* 10, 1))
    i <- i + 2
  }
# Create a list to store a location for each label
  number <- list(label)
  j = 1
  for (y in 0:9) {
    for (x in 0:9) {
      number[[label[j]]] <- c(x, y)
      j <- j + 1
    }
  }
```

```r
# Input Ladder and Chutes
ladders <- list(c(1, 38), c(4, 14), c(9, 31), c(21, 42), c(28, 84), c(36, 44),
                                  c(51, 67), c(71, 91), c(80, 100))
chutes <- list(c(16, 6), c(47, 26), c(49, 11), c(56, 53), c(62, 19), c(64,60),
                                 c(87, 24), c(93, 73), c(95, 75), c(98, 78))
# Get the start and end number of ladder
ladder_start <- numeric(length(ladders))
ladder_end <- numeric(length(ladders))
for(i in seq_len(length(ladders))) {
  ladder_start[i] <- ladders[[i]][1]
  ladder_end[i] <- ladders[[i]][2]
}
# Get the start and end number of chute
chute_start <- numeric(length(chutes))
chute_end <- numeric(length(chutes))
for(i in seq_len(length(chutes))) {
  chute_start[i] <- chutes[[i]][1]
  chute_end[i] <- chutes[[i]][2]
}

# Function to check ladder or chute
check <- function(position) {
  if(any(ladder_start %in% position)) {
    indice <- which(ladder_start %in% position)
    position <- ladder_end[indice]
  }
  if(any(chute_start %in% position)) {
    indice <- which(chute_start %in% position)
    position <- chute_end[indice]
  }
  else {position <- position}
}


Color_plot <- function(number, turns) {
  # Making a plot
  plot.new()
  plot.window(xlim = c(0,10), ylim = c(0,10), asp = 1)
  for (i in 0:10) {
    segments(x0 = i, y0 = 0, x1 = i, y1 = 10)
    segments (x0 = 0, y0 = i, x1 = 10, y1 = i)
  }

  # Assign probability
  prob_assign <- function(probability) {
    probability / sum(probability)
  }

  move_log <- numeric(100 * turns * 6)

  # What Turns
  a <- 1
  repeat {
```

```
    probability <- numeric(100)
    for (i in 1:6) {
      move_log[i] <- i
      move_log[i] <- check(move_log[i])
      probability[move_log[i]] <- probability[move_log[i]] + 1
    }
    probability <- prob_assign(probability)

    if(turns == 1) {
      break
    }

    move_log <- move_log[move_log > 0]
    next_move_log <- numeric(0)
    temp <- numeric(6)

    for (i in seq_along(move_log)) {
      for (j in 1:6) {
        if (move_log[i] + j == 100) {
          temp[j] <- 100
        }
        else {
          if (move_log[i] + j > 100) {
            temp[j] <- move_log[i]
          }
          else {
            temp[j] <- move_log[i] + j
            temp[j] <- check(temp[j])
          }
          probability[temp[j]] <- probability[temp[j]] + 1
        }
      }
      next_move_log <- c(next_move_log, temp[1:6])
    }

    move_log <- next_move_log[next_move_log > 0]
    probability <- prob_assign(probability)
    a <- a + 1

    if(a == turns) {
      break
    }
  }


    # Color
    for (i in 1:100) {
    rect(number[[i]][1], number[[i]][2], number[[i]][1] + 1, number[[i]][2] + 1,
         col = adjustcolor("red", (probability[i] * 6)))
    }
}
```

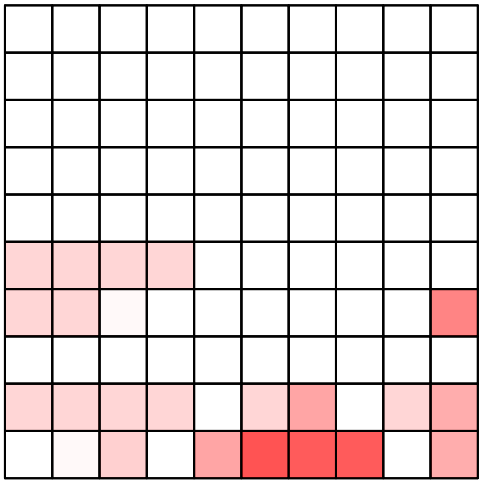
One Roll

```
Color_plot(number, 1)
```



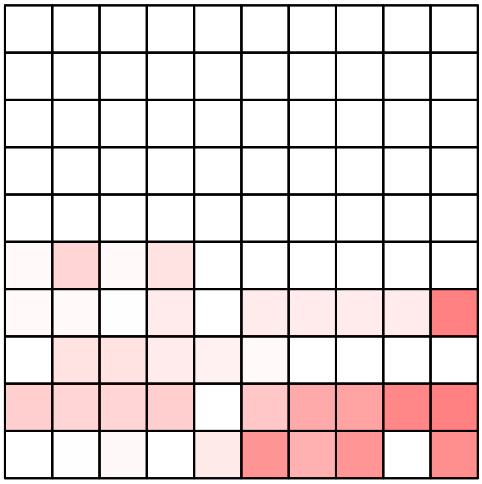
Two Rolls

```
Color_plot(number, 2)
```



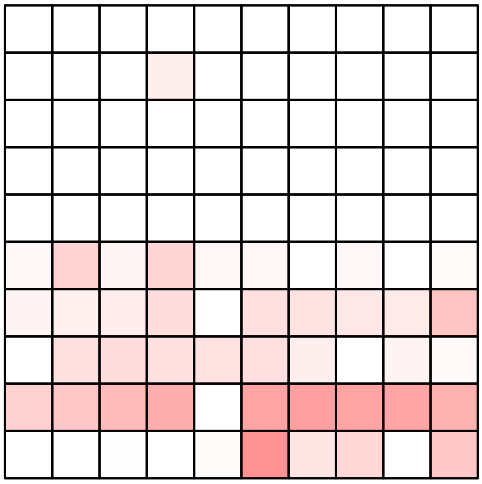
Three Rolls

```
Color_plot(number, 3)
```



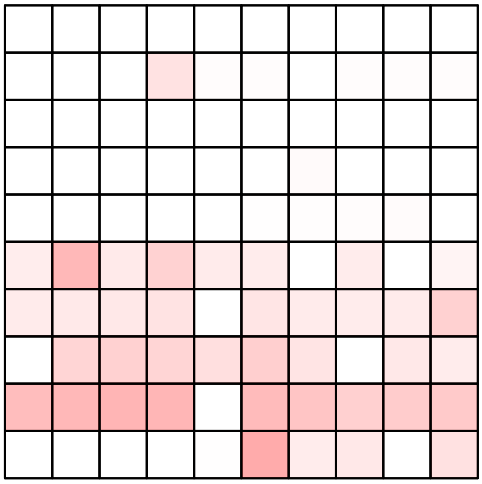
Four Rolls

```
Color_plot(number, 4)
```



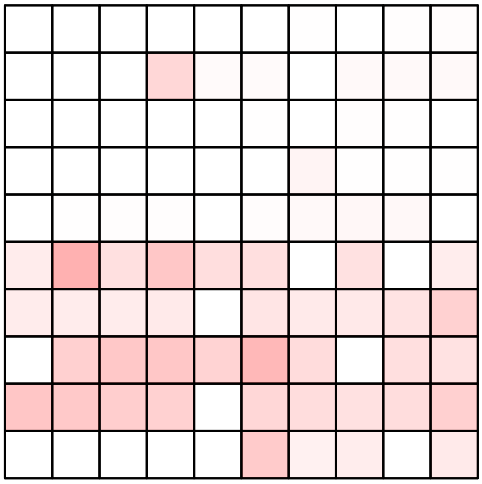
Five Rolls

```
Color_plot(number, 5)
```



Six Rolls

```
Color_plot(number, 6)
```



Seven Rolls

```
Color_plot(number, 7)
```