321 project

```r
#if random number is < prob, player A wins point
#else Player B wins the point
play_point <- function(prob) {
  return(ifelse(runif(1) < prob, 1, 0))
}
```

simulates a game

```r
play_game <- function(p_A, p_B) {
  score_A <- 0
  score_B <- 0
  points_A <- 0
  points_B <- 0


  #if number is less than the prob A wins a point
  #if number is greater equal to prob, B wins point
  while (TRUE) {
    if (runif(1) < p_A) {
      score_A <- score_A + 1
      points_A <- points_A + 1
    } else {
      score_B <- score_B + 1
      points_B <- points_B + 1
    }

    #player A wins if they have at least 4 points and lead by 2 points and vice versa
    if (score_A >= 4 && score_A - score_B >= 2) {
      return(list(winner = 1, points_A = points_A, points_B = points_B))
    }
    if (score_B >= 4 && score_B - score_A >= 2) {
      return(list(winner = -1, points_A = points_A, points_B = points_B))
    }
  }
}
```

A player needs to win at least 6 games to win the set.

```r
play_set <- function(p_A, p_B) {
  games_A <- 0
  games_B <- 0
  total_points_A <- 0
  total_points_B <- 0

  while (TRUE) {
    game_result <- play_game(p_A, p_B)

    if (game_result$winner == 1) {
      games_A <- games_A + 1
    } else {
      games_B <- games_B + 1
    }


    #total points added up
    total_points_A <- total_points_A + game_result$points_A
    total_points_B <- total_points_B + game_result$points_B


    #if player wins at least 6 games and lead by 2, they win
    if (games_A >= 6 && games_A - games_B >= 2) {
      return(list(winner = 1, points_A = total_points_A, points_B = total_points_B))
    }
    if (games_B >= 6 && games_B - games_A >= 2) {
      return(list(winner = -1, points_A = total_points_A, points_B = total_points_B))
    }


    #if there is a tie, there is a 50% chance a new game is won from either side

    if (games_A == 6 && games_B == 6) {
      score_A <- 0
      score_B <- 0
      points_A <- 0
      points_B <- 0
      while (TRUE) {
        if (runif(1) < 0.5) {
          score_A <- score_A + 1
          points_A <- points_A + 1
        } else {
          score_B <- score_B + 1
          points_B <- points_B + 1
        }

        #whoever gets that extra point wins
        if (score_A >= 7 && score_A - score_B >= 2) {
          return(list(winner = 1, points_A = points_A, points_B = points_B))
        }
        if (score_B >= 7 && score_B - score_A >= 2) {
          return(list(winner = -1, points_A = points_A, points_B = points_B))
        }
      }
    }
  }
}
```

```
#need to win 3 sets to win the match
play_match <- function(p_A, p_B, sets_to_win = 3) {
```

```r
  sets_A <- 0
  sets_B <- 0
  total_points_A <- 0
  total_points_B <- 0

  while (TRUE) {
    set_result <- play_set(p_A, p_B)

    if (set_result$winner == 1) {
      sets_A <- sets_A + 1
    } else {
      sets_B <- sets_B + 1
    }

    #if they win the set, add another point
    total_points_A <- total_points_A + set_result$points_A
    total_points_B <- total_points_B + set_result$points_B

    #if either won at least 3 sets, they win the match
    if (sets_A >= sets_to_win) {
      return(list(winner = "A", total_points_A = total_points_A, total_points_B = total_points_B))
    }
    if (sets_B >= sets_to_win) {
      return(list(winner = "B", total_points_A = total_points_A, total_points_B = total_points_B))
    }
  }
}
```

```r
simulate_matches <- function(n, p_A, p_B) {
  results <- data.frame(Winner = character(n), Points_Won = numeric(n), Total_Points = numeric(n))

  for (i in 1:n) {
    result <- play_match(p_A, p_B)

    total_points <- result$total_points_A + result$total_points_B

    #whoever wins, assign the according points they won
    points_won_by_winner <- ifelse(result$winner == "A", result$total_points_A, result$total_points_B)

    #what percent of points did the winner win
    percentage_won <- (points_won_by_winner / total_points) * 100

    results$Winner[i] <- result$winner
    results$Points_Won[i] <- percentage_won
    results$Total_Points[i] <- total_points
  }

  return(results)
}

#results if both players have a 50% chance of winning
results <- simulate_matches(10000, 0.5, 0.5)
```

```r
cat("Average percentage of points won by winner across all matches: ", mean(results$Points_Won), "\n")
```

```
## Average percentage of points won by winner across all matches:  53.46824
```

```r
cat("minimum percentage of points won by the winner across all matches", min(results$Points_Won), "\n")
```

```
## minimum percentage of points won by the winner across all matches 45.33333
```

```r
cat("Maximum percentage of points won by the winner across all matches", max(results$Points_Won), "\n")
```

```
## Maximum percentage of points won by the winner across all matches 70.4918
```

```r
results_2 <- simulate_matches(10000, 0.7, 0.3) # adjust proportions for different results

cat("Average percentage of points won by winner: ", mean(results_2$Points_Won), "\n")
```

```
## Average percentage of points won by winner:  70.53527
```

```r
cat("Winner A percentage: ", mean(results_2$Points_Won[results_2$Winner == "A"]), "\n")
```

```
## Winner A percentage:  70.53527
```

```r
cat("Winner B percentage: ", mean(results_2$Points_Won[results_2$Winner == "B"]), "\n")
```

```
## Winner B percentage:  NaN
```

Min points won:

```r
min(results_2$Points_Won)
```

```
## [1] 57.57576
```

Max points won:

```r
max(results_2$Points_Won)
```

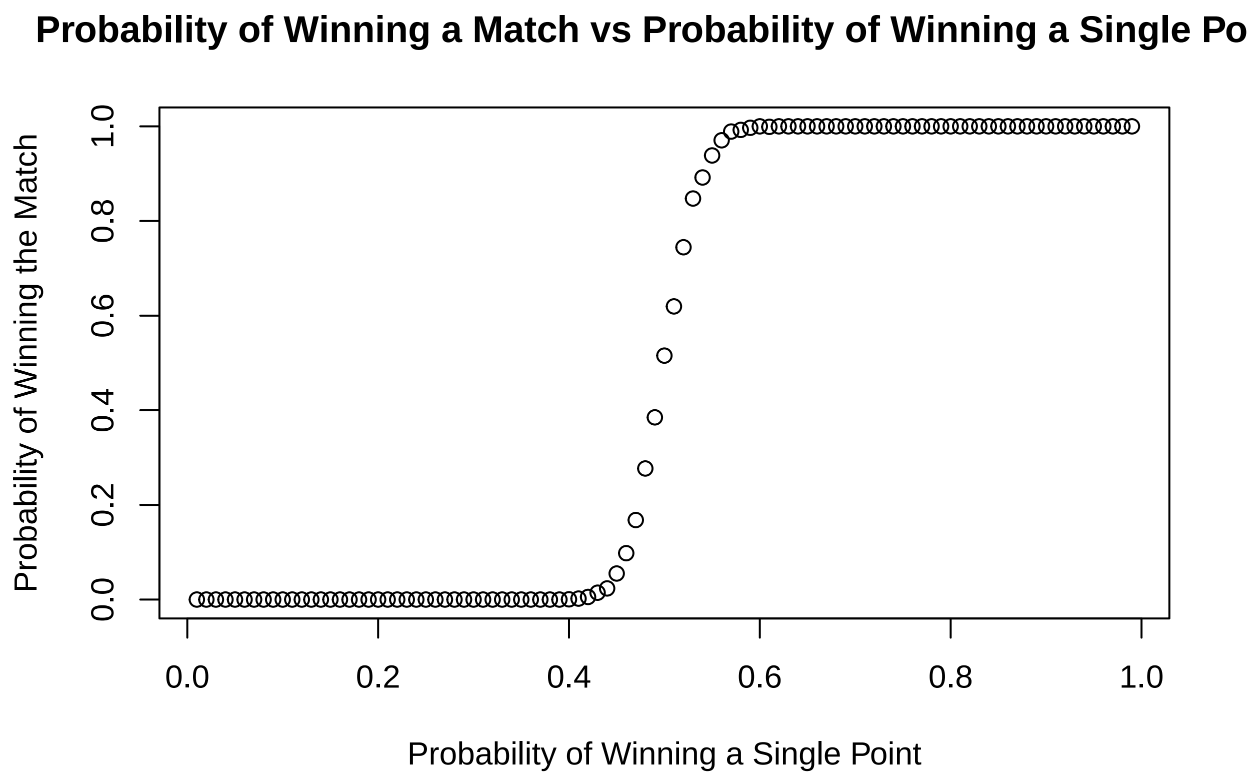
```
## [1] 87.80488
```

```r
winner_less_than_50 <- sum(results$Points_Won < 50)

frequency_less_than_50 <- (winner_less_than_50 / 10000)

cat("Probability of matches where the winner wins less than 50% of the points: ", frequency_less_than_5
```

```
## Probability of matches where the winner wins less than 50% of the points:  0.0711
```

Probability of winning a point vs Probability of Winning a Match

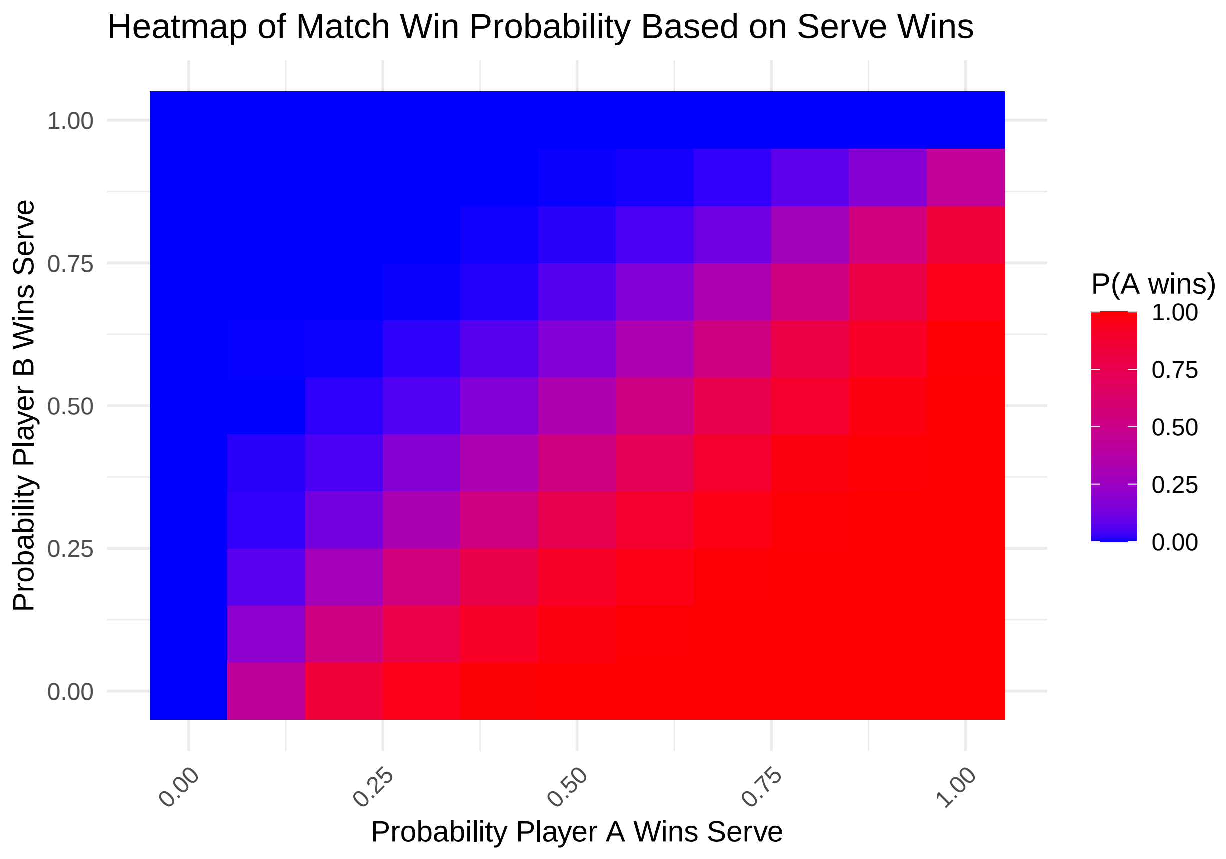
```r
prob_values <- seq(0.01, 0.99, by = 0.01)

match_wins <- c()

for (i in 1:length(prob_values)) {

  prob <- prob_values[i]

  match_outcomes_A <- replicate(1000, play_match(prob, 1 - prob)$winner == "A")
  match_outcomes_B <- replicate(1000, play_match(1 - prob, prob)$winner == "B")

  match_wins[i] <- mean(c(match_outcomes_A, match_outcomes_B))
}

data_probplot <- data.frame(prob_values, match_wins)

plot(data_probplot$prob_values, data_probplot$match_wins,
     xlab = "Probability of Winning a Single Point",
     ylab = "Probability of Winning the Match",
     main = "Probability of Winning a Match vs Probability of Winning a Single Point")
```



```r
## Probability of match win based on serve win, Figure 3
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```r
library(reshape2)

p_A_serve <- 0.6
p_B_serve <- 0.5

sets <- 3
games_per_set <- 6

simulate_match_2 <- function(p_A_serve, p_B_serve, sets, games_per_set) {
  a_sets_won <- 0
  b_sets_won <- 0

  for (i in 1:sets) {
    a_games_won <- 0
    b_games_won <- 0

    for (j in 1:games_per_set) {
      if (runif(1) < p_A_serve) {
        a_games_won <- a_games_won + 1
      }
      if (runif(1) < p_B_serve) {
        b_games_won <- b_games_won + 1
      }
    }

    if (a_games_won > b_games_won) {
      a_sets_won <- a_sets_won + 1
    } else {
      b_sets_won <- b_sets_won + 1
    }

    if (a_sets_won > sets / 2) {
      return(1)
    }
    if (b_sets_won > sets / 2) {
      return(0)
    }
  }
  return(a_sets_won > b_sets_won)
}

serve_outcomes <- expand.grid(A_wins_serve = seq(0, 1, by = 0.1),
                              B_wins_serve = seq(0, 1, by = 0.1))

calculate_probabilities <- function(serve_outcomes, p_A_serve, p_B_serve, sets, games_per_set, simulati
  prob_A_wins <- numeric(nrow(serve_outcomes))

  for (i in 1:nrow(serve_outcomes)) {
    p_A <- serve_outcomes$A_wins_serve[i]
    p_B <- serve_outcomes$B_wins_serve[i]

    results <- replicate(simulations, simulate_match_2(p_A, p_B, sets, games_per_set))
    prob_A_wins[i] <- mean(results == 1)
  }

  serve_outcomes$Prob_A_wins <- prob_A_wins
  return(serve_outcomes)
}

result_data <- calculate_probabilities(serve_outcomes, p_A_serve, p_B_serve, sets, games_per_set)

result_data_melted <- melt(result_data, id.vars = c("A_wins_serve", "B_wins_serve"))

ggplot(result_data_melted, aes(x = A_wins_serve, y = B_wins_serve, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "blue", high = "red", name = "P(A wins)") +
  theme_minimal() +
  labs(
    title = "Heatmap of Match Win Probability Based on Serve Wins",
    x = "Probability Player A Wins Serve",
    y = "Probability Player B Wins Serve",
    fill = "P(A wins)"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Heat map of serve percentage vs opponent rank on prob of winning match:

```r
library(ggplot2)
library(dplyr)
```

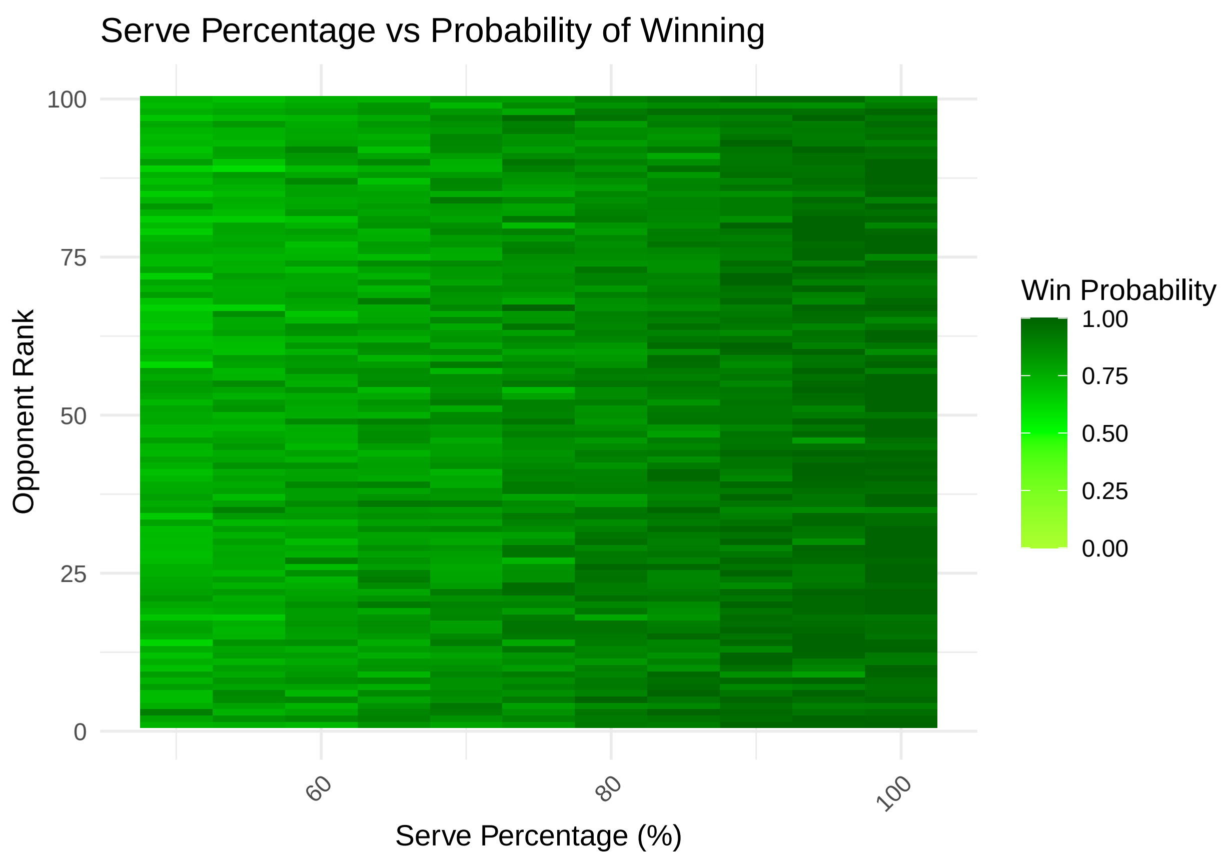
```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
serve_percentages <- seq(50, 100, by = 5)
opponent_ranks <- seq(1, 100, by = 1)

sim_data <- expand.grid(Serve_Percentage = serve_percentages, Opponent_Rank = opponent_ranks)

sim_data$Win_Probability <- with(sim_data,
                                 0.5 + 0.005 * Serve_Percentage - 0.0003 * Opponent_Rank + rnorm(nrow(si

sim_data$Win_Probability <- pmax(pmin(sim_data$Win_Probability, 1), 0)

ggplot(sim_data, aes(x = Serve_Percentage, y = Opponent_Rank, fill = Win_Probability)) +
  geom_tile() +
  scale_fill_gradient2(low = "greenyellow", high = "darkgreen", mid = "green", midpoint = 0.5, limits =
  labs(title = "Serve Percentage vs Probability of Winning",
       x = "Serve Percentage (%)",
       y = "Opponent Rank",
       fill = "Win Probability") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Prob of winning based on which player starts on serve:

```
simulate_point <- function(server_win_probability) {
  runif(1) < server_win_probability
}

simulate_game <- function(server_win_probability) {
  server_points <- 0
  returner_points <- 0
  while (server_points < 4 & returner_points < 4) {
    if (simulate_point(server_win_probability)) {
      server_points <- server_points + 1
    } else {
      returner_points <- returner_points + 1
    }
  }

  while (server_points == 3 & returner_points == 3) {
    if (simulate_point(server_win_probability)) {
      server_points <- server_points + 1
    } else {
      returner_points <- returner_points + 1
    }
  }

  if (server_points > returner_points) {
```

```r
    return("server")
  } else {
    return("returner")
  }
}

simulate_set <- function(server_win_probability) {
  server_games <- 0
  returner_games <- 0
  while (server_games < 6 & returner_games < 6) {
    game_winner <- simulate_game(server_win_probability)
    if (game_winner == "server") {
      server_games <- server_games + 1
    } else {
      returner_games <- returner_games + 1
    }
  }

  return(ifelse(server_games > returner_games, "server", "returner"))
}

simulate_match <- function(server_win_probability, server_first = TRUE) {
  set1_winner <- simulate_set(server_win_probability)
  set2_winner <- simulate_set(server_win_probability)

  if (set1_winner != set2_winner) {
    match_winner <- ifelse(set1_winner == "server", "server", "returner")
  } else {
    set3_winner <- simulate_set(server_win_probability)
    match_winner <- ifelse(set3_winner == "server", "server", "returner")
  }

  return(match_winner)
}

simulate_multiple_matches <- function(n, server_win_probability, server_first = TRUE) {
  results <- replicate(n, simulate_match(server_win_probability, server_first))
  server_wins <- sum(results == "server")
  returner_wins <- sum(results == "returner")

  server_win_proportion <- server_wins / n
  returner_win_proportion <- returner_wins / n

  return(list(server_win_proportion = server_win_proportion,
              returner_win_proportion = returner_win_proportion))
}

server_win_probability <- 0.50

simulation_result_server_first <- simulate_multiple_matches(1000, server_win_probability, server_first
cat("Proportion of Wins (Server First):\n")
```

```
## Proportion of Wins (Server First):
```

```r
cat("Server Win Proportion: ", simulation_result_server_first$server_win_proportion, "\n")
```

```
## Server Win Proportion:  0.498
```

```r
cat("Returner Win Proportion: ", simulation_result_server_first$returner_win_proportion, "\n")
```

```
## Returner Win Proportion:  0.502
```

```r
simulation_result_returner_first <- simulate_multiple_matches(1000, server_win_probability, server_firs
cat("\nProportion of Wins (Returner First):\n")
```

```
##
## Proportion of Wins (Returner First):
```

```r
cat("Server Win Proportion: ", simulation_result_returner_first$server_win_proportion, "\n")
```

```
## Server Win Proportion:  0.488
```

```r
cat("Returner Win Proportion: ", simulation_result_returner_first$returner_win_proportion, "\n")
```

```
## Returner Win Proportion:  0.512
```

```r
server_first_prop <- simulation_result_server_first$server_win_proportion
returner_first_prop <- simulation_result_returner_first$server_win_proportion

pooled_prop <- (sum(simulation_result_server_first$server_win_proportion) +
                sum(simulation_result_returner_first$server_win_proportion)) / 2

se_diff <- sqrt(pooled_prop * (1 - pooled_prop) * (1 / 1000 + 1 / 1000))


z_stat <- (server_first_prop - returner_first_prop) / se_diff


p_value <- 2 * (1 - pnorm(abs(z_stat)))

cat("Server First Win Proportion: ", server_first_prop, "\n")
```

```
## Server First Win Proportion:  0.498
```

```r
cat("Returner First Win Proportion: ", returner_first_prop, "\n")
```

```
## Returner First Win Proportion:  0.488
```

```r
cat("Z-statistic: ", z_stat, "\n")
```

```
## Z-statistic:  0.4472574
```

```r
cat("P-value: ", p_value, "\n")
```

```
## P-value:  0.6546892
```