Golf course simulator code

Aayush

2024-02-24

```
#load latex for rendering
library(tinytex)
# Load the readxl package
library(readxl)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# Define the Excel file path
excel_file_path <- "C:\\Users\\Aayush\\Documents\\Golf project\\SustainableGolfSimulator.xlsx"

# Get sheet names from the Excel file
sheet_names <- excel_sheets(excel_file_path)

# Initialize an empty list for data frames
all_data <- list()

# Read each sheet into a data frame and store in the list
for (sheet in sheet_names) {
  sheet_data <- read_excel(excel_file_path, sheet = sheet)
  all_data[[sheet]] <- sheet_data
}
```

```
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## * `` -> `...8`
```

```r
# Access data frames using sheet names, e.g., all_data$Sheet1
```

```r
# Creating the Parameters data frame
Parameters <- data.frame(
  `Tee Shot Distance` = 250,
  `SD Angle` = 8,
  `Hole Length` = 380,
  `Center to FW Edge` = 20,
  `Center to Hazard Edge` = 40
)

# Print the Parameters data frame
print(Parameters)
```

```
##   Tee.Shot.Distance SD.Angle Hole.Length Center.to.FW.Edge
## 1               250        8         380                20
##   Center.to.Hazard.Edge
## 1                    40
```

```r
# Create the data frame
Outcomes <- data.frame(
  Approach_Dist = 130:240,
  F_ES = c(2.85, 2.855, 2.86, 2.865, 2.87, 2.875, 2.88, 2.885, 2.89,
           2.895, 2.9, rep(NA, 100)),
  R_ES = c(3.05, 3.055, 3.06, 3.065, 3.07, 3.075, 3.08, 3.085, 3.09,
           3.095, rep(NA, 101)),
  H_ES = c(4.05, 4.055, 4.06, 4.065, 4.07, 4.075, 4.08, 4.085, 4.09,
           4.095, rep(NA, 101))
)

# Print the data frame
head(Outcomes)
```

```
##   Approach_Dist  F_ES  R_ES  H_ES
## 1           130 2.850 3.050 4.050
## 2           131 2.855 3.055 4.055
## 3           132 2.860 3.060 4.060
## 4           133 2.865 3.065 4.065
## 5           134 2.870 3.070 4.070
## 6           135 2.875 3.075 4.075
```

```r
# Update row 11 based on F_ES
Outcomes$R_ES[11] <- Outcomes$F_ES[11] + 0.2
Outcomes$H_ES[11] <- Outcomes$R_ES[11] + 1

# Print the updated data frame
head(Outcomes)
```

```
##   Approach_Dist  F_ES  R_ES  H_ES
## 1           130 2.850 3.050 4.050
## 2           131 2.855 3.055 4.055
## 3           132 2.860 3.060 4.060
```

```
## 4           133 2.865 3.065 4.065
## 5           134 2.870 3.070 4.070
## 6           135 2.875 3.075 4.075
```

```
# Fill in values starting from row 12
for (i in 12:nrow(Outcomes)) {
  Outcomes$F_ES[i] <- Outcomes$F_ES[i - 1] + 0.01
  Outcomes$R_ES[i] <- Outcomes$F_ES[i] + 0.2
  Outcomes$H_ES[i] <- Outcomes$R_ES[i] + 1
}

# Print the final data frame
head(Outcomes)
```

```
##   Approach_Dist  F_ES  R_ES  H_ES
## 1           130 2.850 3.050 4.050
## 2           131 2.855 3.055 4.055
## 3           132 2.860 3.060 4.060
## 4           133 2.865 3.065 4.065
## 5           134 2.870 3.070 4.070
## 6           135 2.875 3.075 4.075
```

```
# Define the function to run the golf simulator
runGolfSimulatorPar4 <- function(num_trials,
                                 Tee.Shot.Distance = 250,
                                 SD.Angle = 8,
                                 Hole.Length = 380,
                                 Center.to.FW.Edge = 20,
                                 Center.to.Hazard.Edge = 40) {
  # Create the Simulator data frame
  Simulator <- data.frame(
    Trial = 1:num_trials,
    Tee_Degrees = rep(NA, num_trials),
    Tee_Length = rep(NA, num_trials),
    Tee_Width = rep(NA, num_trials),
    Outcome = rep(NA, num_trials),
    Approach_Dist = rep(NA, num_trials),
    Score = rep(NA, num_trials)
  )

  # Run the simulation for each trial
  for (i in 1:nrow(Simulator)) {
    Simulator$Tee_Degrees_no_abs[i] <- SD.Angle *
      (rnorm(1, mean = 0, sd = 1))

    Simulator$Tee_Width_no_abs[i] <- Tee.Shot.Distance *
      sin(Simulator$Tee_Degrees_no_abs[i] * (pi/180))

    Simulator$Tee_Degrees[i] <- SD.Angle *
      abs(rnorm(1, mean = 0, sd = 1))

    Simulator$Tee_Length[i] <- Tee.Shot.Distance *
      cos(Simulator$Tee_Degrees[i] * (pi/180))
```

```
  Simulator$Tee_Length_no_abs[i] <- Tee.Shot.Distance *
    cos(Simulator$Tee_Degrees_no_abs[i] * (pi/180))

  Simulator$Tee_Width[i] <- Tee.Shot.Distance *
    sin(Simulator$Tee_Degrees[i] * (pi/180))

  Simulator$Outcome[i] <- ifelse(Simulator$Tee_Width[i] <
                                Center.to.FW.Edge, "F",
                                ifelse(Simulator$Tee_Width[i] <
                                       Center.to.Hazard.Edge,
                                       "R", "H"))

  Simulator$Approach_Dist[i] <- sqrt((Hole.Length -
                                     Simulator$Tee_Length[i])^2 +
                                    Simulator$Tee_Width[i]^2)

  Simulator$Score[i] <- ifelse(Simulator$Outcome[i] == 'F',
                              Outcomes[which(Outcomes$Approach_Dist ==
                              floor(Simulator$Approach_Dist[i])),
                              "F_ES"] + 1,
                              ifelse(Simulator$Outcome[i] == 'R',
                                     Outcomes[which(Outcomes$Approach_Dist ==
                                     floor(Simulator$Approach_Dist[i])),
                                     "R_ES"] + 1,
                                     Outcomes[which(Outcomes$Approach_Dist ==
                                     floor(Simulator$Approach_Dist[i])),
                                     "H_ES"] + 1))
}

# Calculate outcome distribution
distribution <- Simulator %>%
  group_by(Outcome) %>%
  count() %>%
  mutate(percent = n / num_trials * 100)

# Print the results
print(Simulator)
print(mean(Simulator$Score))
print(distribution)

# Plot the original setup
rects <- data.frame(xstart = c(-80, -40, -20, 20, 40),
                    xend = c(-40, -20, 20, 40, 80),
                    col = c("red", "darkgreen", "green",
                            "darkgreen", "red"))

custom_labels <- c("Hazard" = "red", "Rough" = "darkgreen",
                   "Fairway" = "green")

plot_1 <- ggplot(data = Simulator) +
  geom_point(aes(x = Tee_Width_no_abs, y = Tee_Length_no_abs),
             alpha = 0.15, size = 3) +
  geom_rect(data = rects,
```

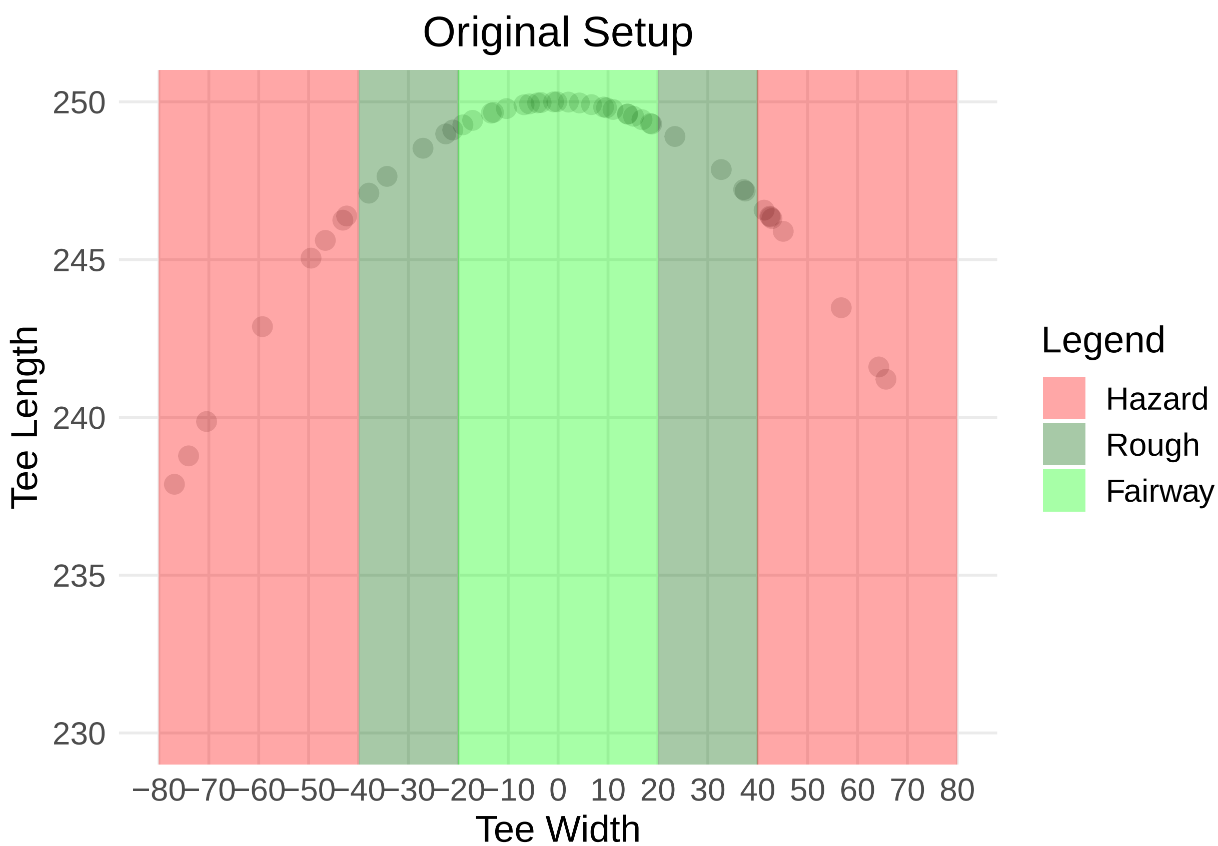
```r
            aes(xmin = xstart, xmax = xend, ymin = -Inf, ymax = Inf,
                fill = col), alpha = 0.35) +
  scale_fill_manual(values = rects$col, breaks = unique(rects$col),
                    labels = c("Hazard", "Rough", "Fairway")) +
  guides(fill = guide_legend(title = "Legend")) +
  scale_x_continuous(breaks = seq(-80, 80, by = 10),
                     limits = c(-80, 80), minor_breaks = NULL) +
  scale_y_continuous(limits = c(230, 250), minor_breaks = NULL) +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14),
    axis.text.x = element_text(size = 12),
    axis.text.y = element_text(size = 12),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12),
    plot.title = element_text(size = 16, hjust = 0.5)
  ) +
  labs(x = "Tee Width", y = "Tee Length", title = "Original Setup")

print(plot_1)

ggsave("plot_1.png", plot = plot_1, width = 21, height = 7, dpi = 300)

# Plot the modified setup
rects2 <- data.frame(xstart = c(-80, -30, -10, 10, 30),
                     xend = c(-30, -10, 10, 30, 80),
                     col = c("red", "darkgreen", "green",
                             "darkgreen", "red"))

custom_labels2 <- c("Hazard" = "red", "Rough" = "darkgreen",
                    "Fairway" = "green")

plot_2 <- ggplot(data = Simulator) +
  geom_point(aes(x = Tee_Width_no_abs, y = Tee_Length_no_abs),
             alpha = 0.15, size = 3) +
  geom_rect(data = rects2,
            aes(xmin = xstart, xmax = xend, ymin = -Inf, ymax = Inf,
                fill = col), alpha = 0.35) +
  scale_fill_manual(values = rects2$col, breaks = unique(rects$col),
                    labels = c("Hazard", "Rough", "Fairway")) +
  guides(fill = guide_legend(title = "Legend")) +
  scale_x_continuous(breaks = seq(-80, 80, by = 10),
                     limits = c(-80, 80), minor_breaks = NULL) +
  scale_y_continuous(limits = c(230, 250), minor_breaks = NULL) +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14),
    axis.text.x = element_text(size = 12),
    axis.text.y = element_text(size = 12),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12),
```

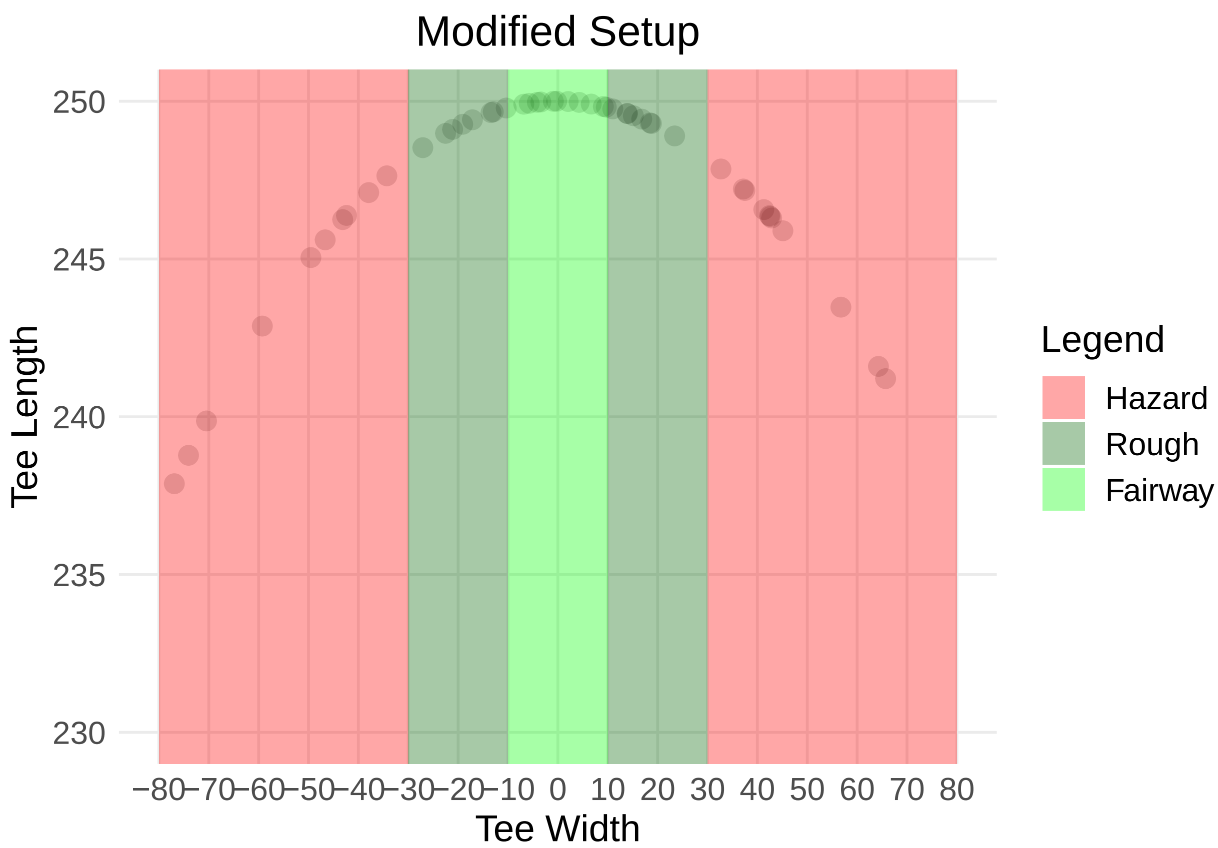
```r
      plot.title = element_text(size = 16, hjust = 0.5)
    ) +
    labs(x = "Tee Width", y = "Tee Length", title = "Modified Setup")

  print(plot_2)

  ggsave("plot_2.png", plot = plot_2, width = 21, height = 7, dpi = 300)
}

# Run the golf simulator with 500 trials
runGolfSimulatorPar4(50)
```

```
##    Trial Tee_Degrees Tee_Length  Tee_Width Outcome Approach_Dist Score
## 1      1  11.9674529   244.5664 51.8390041       H      145.0157 5.150
## 2      2   8.9980790   246.9234 39.1003373       R      138.7019 4.090
## 3      3   6.0973597   248.5857 26.5545624       R      134.0704 4.070
## 4      4   2.1188820   249.8291  9.2432597       F      130.4987 3.850
## 5      5   7.1114660   248.0768 30.9500148       R      135.5051 4.075
## 6      6   3.8023723   249.4497 16.5788032       F      131.5988 3.855
## 7      7   6.5071656   248.3894 28.3318682       R      134.6256 4.070
## 8      8  17.9750328   237.7978 77.1506334       H      161.7829 5.310
## 9      9   1.2227840   249.9431  5.3349966       F      130.1663 3.850
## 10    10  12.8370510   243.7515 55.5447609       H      147.1356 5.170
## 11    11   2.7182954   249.7187 11.8563520       F      130.8197 3.850
## 12    12   5.3695494   248.9030 23.3947987       R      133.1681 4.065
## 13    13   1.4267811   249.9225  6.2248635       F      130.2264 3.850
## 14    14   0.1242195   249.9994  0.5420094       F      130.0017 3.850
## 15    15   6.3987356   248.4426 27.8617503       R      134.4754 4.070
## 16    16   0.6269402   249.9850  2.7354881       F      130.0437 3.850
## 17    17  14.1616145   242.4024 61.1644621       H      150.5795 5.200
## 18    18   3.8764623   249.4280 16.9013565       F      131.6613 3.855
## 19    19   9.4107469   246.6354 40.8777524       H      139.4888 5.095
## 20    20  11.4754583   245.0025 49.7370455       H      143.8683 5.130
## 21    21  12.7958636   243.7913 55.3695242       H      147.0326 5.170
## 22    22   5.6898125   248.7683 24.7857056       R      133.5518 4.065
## 23    23   7.7676880   247.7061 33.7892045       R      136.5408 4.080
## 24    24   8.3261633   247.3649 36.2020096       R      137.4869 4.085
## 25    25   0.3277069   249.9959  1.4298835       F      130.0120 3.850
## 26    26  12.7479303   243.8376 55.1655507       H      146.9131 5.160
## 27    27   7.0052984   248.1337 30.4902818       R      135.3454 4.075
## 28    28   3.0174961   249.6534 13.1602250       F      131.0093 3.855
## 29    29   0.5422303   249.9888  2.3658909       F      130.0327 3.850
## 30    30   6.5955036   248.3454 28.7147984       R      134.7496 4.070
## 31    31   4.5284088   249.2196 19.7383463       F      132.2616 3.860
## 32    32   1.1759533   249.9473  5.1307039       F      130.1538 3.850
## 33    33   1.7729455   249.8803  7.7346996       F      130.3494 3.850
## 34    34   2.2936585   249.7997 10.0053005       F      130.5842 3.850
## 35    35   1.5625625   249.9070  6.8171198       F      130.2715 3.850
## 36    36  19.0144327   236.3591 81.4515796       H      165.1274 5.350
## 37    37  13.4445314   243.1489 58.1259735       H      148.6838 5.180
## 38    38   6.7724744   248.2556 29.4817303       R      135.0028 4.075
## 39    39  11.3597808   245.1024 49.2422958       H      143.6042 5.130
## 40    40  12.8486898   243.7402 55.5942740       H      147.1647 5.170
```

```
## 41    41   4.1505093   249.3443 18.0941783       F      131.9026 3.855
## 42    42   2.0462096   249.8406  8.9263761       F      130.4651 3.850
## 43    43   6.6223799   248.3319 28.8312892       R      134.7877 4.070
## 44    44   0.8901299   249.9698  3.8837680       F      130.0882 3.850
## 45    45   1.5483204   249.9087  6.7550001       F      130.2665 3.850
## 46    46   4.5457787   249.2136 19.8138993       F      132.2788 3.860
## 47    47   6.7487591   248.2678 29.3789724       R      134.9686 4.070
## 48    48  14.1278552   242.4384 61.0216254       H      150.4887 5.200
## 49    49   1.3973301   249.9257  6.0963985       F      130.2171 3.850
## 50    50   2.6665480   249.7293 11.6308107       F      130.7889 3.850
##    Tee_Degrees_no_abs Tee_Width_no_abs Tee_Length_no_abs
## 1          9.80566080       42.5767141          246.3478
## 2          3.18197372       13.8768437          249.6146
## 3          7.51412528       32.6926529          247.8532
## 4          9.76870866       42.4178266          246.3752
## 5          3.85688626       16.8161343          249.4338
## 6          3.20120821       13.9606398          249.6099
## 7         -8.72614698      -37.9279761          247.1062
## 8          0.97749035        4.2648994          249.9636
## 9        -10.75511613      -46.6529404          245.6084
## 10        10.39593765       45.1123520          245.8961
## 11        -1.56660397       -6.8347476          249.9066
## 12         9.87091137       42.8572364          246.2991
## 13        -2.97054275      -12.9556320          249.6641
## 14       -25.02905037     -105.7694319          226.5233
## 15        -9.93648602      -43.1390960          246.2499
## 16        -4.84197082      -21.1019452          249.1078
## 17       -11.42439367      -49.5186682          245.0467
## 18        -6.21811268      -27.0784068          248.5292
## 19         8.55353833       37.1833757          247.2193
## 20        -3.07304723      -13.4022702          249.6405
## 21       -13.71113350      -59.2567323          242.8758
## 22        -2.37384182      -10.3548759          249.7855
## 23        -4.37751040      -19.0819154          249.2707
## 24        -9.75872674      -42.3749031          246.3826
## 25       -21.33242049      -90.9445911          232.8714
## 26        13.11857896       56.7417802          243.4756
## 27         9.50568035       41.2863466          246.5673
## 28        -0.05462649       -0.2383530          249.9999
## 29        -5.16791519      -22.5187213          248.9837
## 30         2.52412848       11.0100260          249.7574
## 31       -17.23020667      -74.0529093          238.7806
## 32        14.89643463       64.2681644          241.5980
## 33         1.53217926        6.6845964          249.9106
## 34         8.62476419       37.4906716          247.1729
## 35         4.24428036       18.5022345          249.3144
## 36       -17.91380629      -76.8964775          237.8801
## 37        -3.92914359      -17.1306889          249.4124
## 38         0.47177284        2.0584741          249.9915
## 39        -0.20848861       -0.9097012          249.9983
## 40        15.23884491       65.7108430          241.2096
## 41        -0.78224640       -3.4130878          249.9767
## 42         2.22483864        9.7052505          249.8115
## 43        -0.94345174       -4.1163988          249.9661
```

```
## 44          4.30296628       18.7575881          249.2953
## 45          3.46977971       15.1305178          249.5417
## 46        -16.36714944      -70.4478468          239.8689
## 47          2.08857858        9.1111251          249.8339
## 48          5.37305226       23.4100159          248.9015
## 49         -7.88322362      -34.2886290          247.6374
## 50         -1.31677909       -5.7450269          249.9340
## [1] 4.2607
## # A tibble: 3 x 3
## # Groups:   Outcome [3]
##   Outcome     n percent
##   <chr>   <int>   <dbl>
## 1 F          23      46
## 2 H          13      26
## 3 R          14      28
```






```
library(tidyverse)
Par3Out <- all_data$Par3Out

Par3Out <- Par3Out %>%
  rename(Approach_Dist = 'Approach Dist', G_ES = 'G E(S)',
         R_ES = 'R E(S)', H_ES = 'H E(S)')
```

```
runGolfSimulator2 <- function(num_trials,
                              Tee.Shot.Distance = 120,
                              SD.Angle = 8,
                              Semicircle.Green.Radius = 25,
                              Center.to.Hazard.Edge = 30) {
  # Create Simulator data frame
  Simulator <- data.frame(
    Trial = 1:num_trials,
    Tee_Degrees = rep(NA, num_trials),
    Tee_Length = rep(NA, num_trials),
    Tee_Width = rep(NA, num_trials),
    Outcome = rep(NA, num_trials),
    Approach_Dist = rep(NA, num_trials),
    Score = rep(NA, num_trials)
  )

  for (i in 1:nrow(Simulator)) {
    Simulator$Tee_Degrees_no_abs[i] <- SD.Angle *
      (rnorm(1, mean = 0, sd = 1))
```

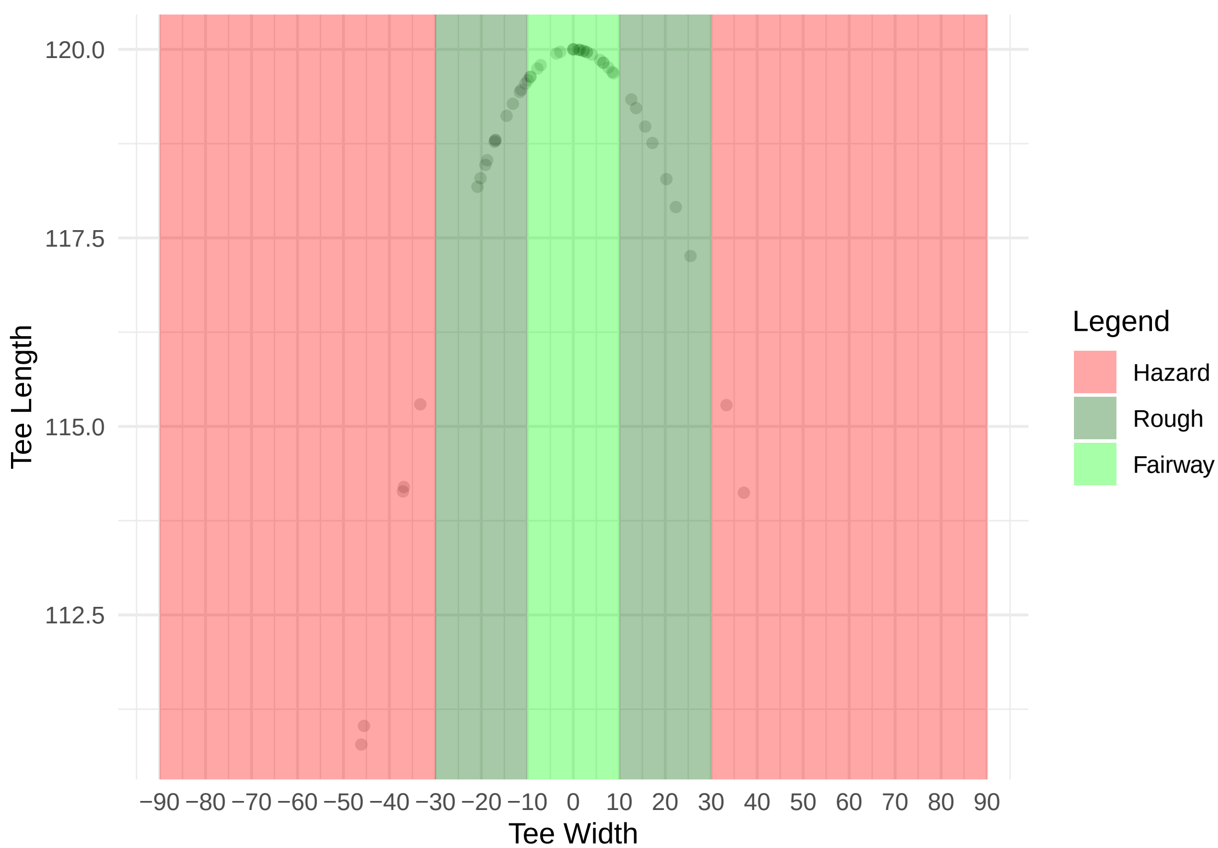
```r
  Simulator$Tee_Width_no_abs[i] <- Tee.Shot.Distance *
    sin(Simulator$Tee_Degrees_no_abs[i] * (pi/180))

  Simulator$Tee_Degrees[i] <- SD.Angle *
    abs(rnorm(1, mean = 0, sd = 1))

  Simulator$Tee_Length[i] <- Tee.Shot.Distance *
    cos(Simulator$Tee_Degrees[i] * (pi/180))

  Simulator$Tee_Length_no_abs[i] <- Tee.Shot.Distance *
    cos(Simulator$Tee_Degrees_no_abs[i] * (pi/180))

  Simulator$Tee_Width[i] <- Tee.Shot.Distance *
    sin(Simulator$Tee_Degrees[i] * (pi/180))

  Simulator$Approach_Dist[i] <- sqrt((Tee.Shot.Distance -
    Simulator$Tee_Length[i])^2 + Simulator$Tee_Width[i]^2)

  Simulator$Outcome[i] <- ifelse(Simulator$Approach_Dist[i] <
    Semicircle.Green.Radius, "G", ifelse(Simulator$Tee_Width[i] <
    Center.to.Hazard.Edge, "R", "H"))

  # Calculate Score based on Outcome
  temp_score <- ifelse(Simulator$Outcome[i] == 'G',
    Par3Out[which(Par3Out$Approach_Dist == floor(
    Simulator$Approach_Dist[i])), "G_ES"] + 1,
    ifelse(Simulator$Outcome[i] == 'R',
    Par3Out[which(Par3Out$Approach_Dist == floor(
    Simulator$Approach_Dist[i])), "R_ES"] + 1,
    Par3Out[which(Par3Out$Approach_Dist == floor(
    Simulator$Approach_Dist[i])), "H_ES"] + 1))

  # Unlist to convert to numeric vector
  temp_score <- unlist(temp_score)

  # Assign the value to Score
  Simulator$Score[i] <- temp_score[1]
}

distribution <- Simulator %>%
  group_by(Outcome) %>%
  count() %>%
  mutate(percent = n/num_trials * 100)

# Print the updated Simulator data frame
print(Simulator)
cat("Average:", mean(Simulator$Score))
print(distribution)

# Define rectangles data
rects <- data.frame(xstart = c(-90, -30, -10, 10, 30),
                    xend = c(-30, -10, 10, 30, 90),
                    col = c("red", "darkgreen", "green", "darkgreen", "red"))
```

```
  custom_labels <- c("Hazard" = "red", "Rough" = "darkgreen", "Fairway" = "green")

  # Create the plot
  plot_2 <- ggplot(data = Simulator) +
    geom_point(aes(x = Tee_Width_no_abs, y = Tee_Length_no_abs), alpha = 0.15) +
    geom_rect(data = rects, aes(xmin = xstart, xmax = xend, ymin = -Inf, ymax = Inf,
                                fill = col), alpha = 0.35) +
    scale_fill_manual(values = rects$col, breaks = unique(rects$col),
                      labels = c("Hazard", "Rough", "Fairway")) +
    guides(fill = guide_legend(title = "Legend")) +
    scale_x_continuous(breaks = seq(-100, 100, by = 10)) +
    theme_minimal() +
    labs(x = "Tee Width", y = "Tee Length", t)

  print(plot_2)

  ggsave("plot_3.png", plot = plot_2, width = 10, height = 8, dpi = 300)
}

# Call the function with the desired number of trials
runGolfSimulator2(50)  # You can replace 1100 with any number of trials you want
```

```
##    Trial Tee_Degrees Tee_Length  Tee_Width Outcome Approach_Dist Score
## 1      1   2.8411623   119.8525  5.9480780       G     5.9499067 2.810
## 2      2  12.8057989   117.0152 26.5976630       R    26.7646132 3.650
## 3      3   5.9592375   119.3515 12.4585074       G    12.4753730 3.110
## 4      4   6.8440560   119.1449 14.3000935       G    14.3256368 3.190
## 5      5   6.3244680   119.2697 13.2190523       G    13.2392111 3.150
## 6      6   4.5315721   119.6249  9.4810106       G     9.4884288 2.990
## 7      7   0.7271541   119.9903  1.5229072       G     1.5229379 2.070
## 8      8  19.9612994   112.7908 40.9662417       H    41.5957373 4.795
## 9      9  10.6261042   117.9422 22.1278993       G    22.2233792 3.350
## 10    10   5.0765940   119.5293 10.6184874       G    10.6289160 3.030
## 11    11   1.1101094   119.9775  2.3248622       G     2.3249713 2.450
## 12    12  12.3006072   117.2452 25.5648888       R    25.7128853 3.640
## 13    13   9.0622537   118.5021 18.9009032       G    18.9601620 3.270
## 14    14   4.0596620   119.6989  8.4954238       G     8.5007578 2.940
## 15    15   0.8581197   119.9865  1.7971744       G     1.7972248 2.070
## 16    16  14.6147985   116.1173 30.2783150       H    30.5262484 4.690
## 17    17   7.7841129   118.8943 16.2529021       G    16.2904729 3.230
## 18    18   2.2410794   119.9082  4.6925090       G     4.6934065 2.730
## 19    19   5.1567086   119.5143 10.7856104       G    10.7965404 3.030
## 20    20   8.5254520   118.6740 17.7898486       G    17.8391972 3.250
## 21    21   0.8325942   119.9873  1.7437199       G     1.7437659 2.070
## 22    22   3.8294496   119.7321  8.0144104       G     8.0188876 2.940
## 23    23   0.6605887   119.9920  1.3835032       G     1.3835261 2.070
## 24    24   0.3197337   119.9981  0.6696453       G     0.6696479 2.000
## 25    25   3.0328939   119.8319  6.3491122       G     6.3513366 2.880
## 26    26  12.3176247   117.2376 25.5997109       R    25.7483214 3.640
## 27    27   0.3431395   119.9978  0.7186655       G     0.7186687 2.000
## 28    28   4.9087087   119.5599 10.2682034       G    10.2776315 3.030
## 29    29   0.6137203   119.9931  1.2853481       G     1.2853666 2.070
## 30    30  10.4286964   118.0177 21.7214089       G    21.8116728 3.330
```

```
## 31    31  12.4105351   117.1959 25.7897888       R    25.9417809 3.640
## 32    32   0.9259927   119.9843  1.9393101       G     1.9393735 2.070
## 33    33   7.9556420   118.8451 16.6087682       G    16.6488757 3.230
## 34    34   7.8439190   118.8772 16.3769965       G    16.4154392 3.230
## 35    35   6.9031745   119.1301 14.4230211       G    14.4492317 3.190
## 36    36   9.9226546   118.2050 20.6782316       G    20.7559982 3.310
## 37    37  11.8213958   117.4549 24.5833888       G    24.7147823 3.390
## 38    38   0.7035909   119.9910  1.4735603       G     1.4735881 2.070
## 39    39   4.6076343   119.6122  9.6398086       G     9.6476066 2.990
## 40    40   5.9640643   119.3505 12.4685620       G    12.4854686 3.110
## 41    41  11.1621882   117.7300 23.2304324       G    23.3410797 3.370
## 42    42   7.2543761   119.0394 15.1529683       G    15.1833834 3.210
## 43    43   0.4938114   119.9955  1.0342234       G     1.0342330 2.070
## 44    44   0.8627117   119.9864  1.8067910       G     1.8068422 2.070
## 45    45   8.5615040   118.6628 17.8645179       G    17.9144946 3.250
## 46    46   8.6187555   118.6449 17.9830800       G    18.0340650 3.270
## 47    47  12.9046684   116.9692 26.7995445       R    26.9703833 3.650
## 48    48   5.4499313   119.4575 11.3971050       G    11.4100068 3.070
## 49    49   5.9005815   119.3642 12.3363159       G    12.3526886 3.110
## 50    50   2.2696576   119.9059  4.7523166       G     4.7532489 2.730
##    Tee_Degrees_no_abs Tee_Width_no_abs Tee_Length_no_abs
## 1        12.268253399      25.49867874          117.2596
## 2         9.713432378      20.24645550          118.2797
## 3       -17.892264258     -36.86737635          114.1963
## 4         0.693355900       1.45212576          119.9912
## 5        -0.025683143      -0.05379065          120.0000
## 6         0.512659307       1.07369682          119.9952
## 7        -8.976022419     -18.72253391          118.5304
## 8        -9.164798092     -19.11296060          118.4681
## 9         0.043318680       0.09072642          120.0000
## 10       -6.948933197     -14.51815857          119.1185
## 11        0.005143289       0.01077208          120.0000
## 12       -9.679054624     -20.17548346          118.2918
## 13        0.678531414       1.42107965          119.9916
## 14       -1.747078251      -3.65850514          119.9442
## 15       -8.114998954     -16.93924760          118.7984
## 16       -5.410461987     -11.31481170          119.4654
## 17        3.636938540       7.61207201          119.7583
## 18       -3.384937547      -7.08527338          119.7906
## 19       -5.561683474     -11.63007842          119.4351
## 20        2.771647571       5.80266136          119.8596
## 21       -4.989408351     -10.43659027          119.5453
## 22       -8.200112410     -17.11570507          118.7731
## 23        4.175621566       8.73766191          119.6815
## 24        1.936272975       4.05454878          119.9315
## 25       -1.357357787      -2.84257759          119.9663
## 26        1.404711505       2.94172618          119.9639
## 27        4.047139346       8.46926192          119.7008
## 28        3.111219586       6.51292129          119.8231
## 29       16.118602641      33.31518983          115.2827
## 30      -17.982791333     -37.04775991          114.1379
## 31       10.712842755      22.30642341          117.9085
## 32        1.054489667       2.20839332          119.9797
## 33      -16.100785499     -33.27933901          115.2930
```

```
## 34      -6.285275463    -13.13746414        119.2787
## 35       3.127565379      6.54710512        119.8213
## 36       1.076025165      2.25348936        119.9788
## 37       1.414745530      2.96273503        119.9634
## 38      -3.733020248     -7.81288899        119.7454
## 39       6.032998963     12.61214777        119.3354
## 40      -9.999510465    -20.83677162        118.1771
## 41     -22.295675993    -45.52636015        111.0286
## 42       6.533849009     13.65482077        119.2206
## 43      -8.129786616    -16.96990811        118.7940
## 44       7.489923967     15.64222017        118.9761
## 45      18.008321524     37.09861448        114.1214
## 46       8.248802580     17.21663272        118.7585
## 47      -4.469825509     -9.35208769        119.6350
## 48     -22.604954786    -46.12501896        110.7812
## 49      -4.451642408     -9.31412045        119.6380
## 50      -4.738113905     -9.91217601        119.5899
## Average: 2.9901# A tibble: 3 x 3
## # Groups:   Outcome [3]
##   Outcome     n percent
##   <chr>   <int>   <dbl>
## 1 G          43      86
## 2 H           2       4
## 3 R           5      10
```



```r
# Function to calculate the maintenance cost of a golf course
calculate_maintenance_cost <- function(length_manicured_land,
                                       width_manicured_land,
                                       radius_green,
                                       length_fairway,
                                       width_fairway,
                                       length_tee_box,
                                       width_tee_box,
                                       length_bunker,
                                       width_bunker,
                                       cost_per_unit_square_green = 25.07,
                                       cost_per_unit_square_fairway = 0.36,
                                       cost_per_unit_square_tee_box = 3.03,
                                       cost_per_unit_square_bunker = 20.23,
                                       cost_per_unit_square_rough = 0.04) {

  # Calculate the total area of the manicured land
  area_manicured_land <- length_manicured_land * width_manicured_land

  # Calculate the area of the green (semicircle)
  area_green <- (pi * radius_green^2) / 2

  # Calculate the area of the fairway
  area_fairway <- length_fairway * width_fairway

  # Calculate the area of the tee box
  area_tee_box <- length_tee_box * width_tee_box

  # Calculate the area of the bunker
  area_bunker <- length_bunker * width_bunker

  # Calculate the area of the rough (remaining area after other features)
  area_rough <- area_manicured_land - area_green - area_fairway -
                area_tee_box - area_bunker

  # Calculate maintenance cost for the green
  total_cost_green <- area_green * cost_per_unit_square_green

  # Calculate maintenance cost for the fairway
  total_cost_fairway <- area_fairway * cost_per_unit_square_fairway

  # Calculate maintenance cost for the tee box
  total_cost_tee_box <- area_tee_box * cost_per_unit_square_tee_box

  # Calculate maintenance cost for the bunker
  total_cost_bunker <- area_bunker * cost_per_unit_square_bunker

  # Calculate maintenance cost for the rough
  total_cost_rough <- area_rough * cost_per_unit_square_rough

  # Calculate the total maintenance cost of the golf course
  total_cost <- total_cost_green + total_cost_fairway + total_cost_tee_box +
                total_cost_bunker + total_cost_rough
```

```
  # Return the total maintenance cost
  return(total_cost)
}

# Example usage of the function with specified dimensions
cost <- calculate_maintenance_cost(length_manicured_land = 100,
                                   width_manicured_land = 80,
                                   radius_green = 20,
                                   length_fairway = 60,
                                   width_fairway = 20,
                                   length_tee_box = 10,
                                   width_tee_box = 10,
                                   length_bunker = 20,
                                   width_bunker = 10)

# Print the calculated maintenance cost
cat("Cost of Maintenance:", cost, "\n")
```

```
## Cost of Maintenance: 20767.81
```

```
# Par 4 Simulator and Cost Difference Analysis
library(tidyverse)

# Define the Par 4 golf simulator function
runGolfSimulatorPar4 <- function(num_trials,
                                 Tee.Shot.Distance = 250,
                                 SD.Angle = 8,
                                 Hole.Length = 380,
                                 Center.to.FW.Edge = 20,
                                 Center.to.Hazard.Edge = 40) {
  # Initialize the simulator data frame
  Simulator <- data.frame(
    Trial = 1:num_trials,
    Tee_Degrees = rep(NA, num_trials),
    Tee_Length = rep(NA, num_trials),
    Tee_Width = rep(NA, num_trials),
    Outcome = rep(NA, num_trials),
    Approach_Dist = rep(NA, num_trials),
    Score = rep(NA, num_trials)
  )

  for (i in 1:nrow(Simulator)) {
    # Simulate the angle and width of the tee shot without taking absolute values
    Simulator$Tee_Degrees_no_abs[i] <- SD.Angle * (rnorm(1, mean = 0, sd = 1))
    Simulator$Tee_Width_no_abs[i] <- Tee.Shot.Distance *
                                     sin(Simulator$Tee_Degrees_no_abs[i] * (pi/180))

    # Simulate the angle and width of the tee shot (absolute values)
    Simulator$Tee_Degrees[i] <- SD.Angle * abs(rnorm(1, mean = 0, sd = 1))
    Simulator$Tee_Length[i] <- Tee.Shot.Distance *
                               cos(Simulator$Tee_Degrees[i] * (pi/180))
    Simulator$Tee_Length_no_abs[i] <- Tee.Shot.Distance *
                                      cos(Simulator$Tee_Degrees_no_abs[i] * (pi/180))
```

```
    Simulator$Tee_Width[i] <- Tee.Shot.Distance *
                              sin(Simulator$Tee_Degrees[i] * (pi/180))

    # Determine outcome based on shot width
    Simulator$Outcome[i] <- ifelse(Simulator$Tee_Width[i] < Center.to.FW.Edge,
                                   "F", ifelse(Simulator$Tee_Width[i] <
                                   Center.to.Hazard.Edge, "R", "H"))

    # Calculate approach distance to the hole
    Simulator$Approach_Dist[i] <- sqrt((Hole.Length -
                                        Simulator$Tee_Length[i])^2 +
                                        Simulator$Tee_Width[i]^2)

    # Assign score based on outcome and approach distance
    Simulator$Score[i] <- ifelse(Simulator$Outcome[i] == 'F',
                                 Outcomes[which(Outcomes$Approach_Dist ==
                                 floor(Simulator$Approach_Dist[i])),
                                 "F_ES"] + 1, ifelse(Simulator$Outcome[i] == 'R',
                                 Outcomes[which(Outcomes$Approach_Dist ==
                                 floor(Simulator$Approach_Dist[i])),
                                 "R_ES"] + 1, Outcomes[which(Outcomes$Approach_Dist ==
                                 floor(Simulator$Approach_Dist[i])),
                                 "H_ES"] + 1))
  }

  # Return the average score from the simulation
  mean_score <- mean(Simulator$Score)
  return(mean_score)
}

# Define the cost calculation function for golf course maintenance
calculate_maintenance_cost <- function(length_manicured_land,
                                       width_manicured_land,
                                       radius_green,
                                       length_fairway,
                                       width_fairway,
                                       length_tee_box,
                                       width_tee_box,
                                       length_bunker,
                                       width_bunker,
                                       cost_per_unit_square_green = 25.07,
                                       cost_per_unit_square_fairway = 0.36,
                                       cost_per_unit_square_tee_box = 3.03,
                                       cost_per_unit_square_bunker = 20.23,
                                       cost_per_unit_square_rough = 0.04) {

  # Calculate the area of the manicured land
  area_manicured_land <- length_manicured_land * width_manicured_land
  area_green <- (pi * radius_green^2) / 2
  area_fairway <- length_fairway * width_fairway
  area_tee_box <- length_tee_box * width_tee_box
  area_bunker <- length_bunker * width_bunker
  area_rough <- area_manicured_land - area_green - area_fairway -
```

```
                area_tee_box - area_bunker

  # Calculate the total maintenance cost for each area
  total_cost_green <- area_green * cost_per_unit_square_green
  total_cost_fairway <- area_fairway * cost_per_unit_square_fairway
  total_cost_tee_box <- area_tee_box * cost_per_unit_square_tee_box
  total_cost_bunker <- area_bunker * cost_per_unit_square_bunker
  total_cost_rough <- area_rough * cost_per_unit_square_rough

  # Return the total maintenance cost for the golf course
  total_cost <- total_cost_green + total_cost_fairway +
                total_cost_tee_box + total_cost_bunker +
                total_cost_rough
  return(total_cost)
}

# Set parameters for each golf course
course1_params <- list(
  num_trials = 1000,
  Tee.Shot.Distance = 250,
  SD.Angle = 8,
  Hole.Length = 400,
  Center.to.FW.Edge = 20,
  Center.to.Hazard.Edge = 40
)

course2_params <- list(
  num_trials = 1000,
  Tee.Shot.Distance = 250,
  SD.Angle = 8,
  Hole.Length = 420,
  Center.to.FW.Edge = 20,
  Center.to.Hazard.Edge = 40
)

# Calculate stroke averages for each course
course1_stroke_avg <- do.call(runGolfSimulatorPar4, course1_params)
course2_stroke_avg <- do.call(runGolfSimulatorPar4, course2_params)

# Set maintenance cost parameters for each course
cost_params_course1 <- list(
  length_manicured_land = 400, width_manicured_land = 80,
  radius_green = 20, length_fairway = 400, width_fairway = 40,
  length_tee_box = 10, width_tee_box = 10, length_bunker = 20,
  width_bunker = 10, cost_per_unit_square_green = 2,
  cost_per_unit_square_fairway = 1.5, cost_per_unit_square_tee_box = 1.2,
  cost_per_unit_square_bunker = 1.8, cost_per_unit_square_rough = 1
)

cost_params_course2 <- list(
  length_manicured_land = 420, width_manicured_land = 80,
  radius_green = 20, length_fairway = 420, width_fairway = 40,
  length_tee_box = 10, width_tee_box = 10, length_bunker = 20,
```

```r
  width_bunker = 10, cost_per_unit_square_green = 2,
  cost_per_unit_square_fairway = 1.5, cost_per_unit_square_tee_box = 1.2,
  cost_per_unit_square_bunker = 1.8, cost_per_unit_square_rough = 1
)

# Calculate maintenance costs for each course
course1_cost <- do.call(calculate_maintenance_cost, cost_params_course1)
course2_cost <- do.call(calculate_maintenance_cost, cost_params_course2)

# Calculate manicured land area for each course
length_manicured_land_course1 <- course1_params$Hole.Length
width_of_manicured_land_course1 <- 2 * course1_params$Center.to.Hazard.Edge
course1_manicured_land <- length_manicured_land_course1 * width_of_manicured_land_course1

length_manicured_land_course2 <- course2_params$Hole.Length
width_of_manicured_land_course2 <- 2 * course2_params$Center.to.Hazard.Edge
course2_manicured_land <- length_manicured_land_course2 * width_of_manicured_land_course2

# Compare stroke averages, maintenance costs, and manicured land areas
stroke_diff <- course2_stroke_avg - course1_stroke_avg
cost_diff <- course2_cost - course1_cost
manicured_land_diff <- course2_manicured_land - course1_manicured_land

# Calculate percentage changes
stroke_percent_change <- (stroke_diff / course1_stroke_avg) * 100
cost_percent_change <- (cost_diff / course1_cost) * 100
manicured_land_percent_change <- (manicured_land_diff / course1_manicured_land) * 100

# Print the results of the comparison
cat("Course 1 Stroke Average:", course1_stroke_avg, "\n")
```

```
## Course 1 Stroke Average: 4.38574
```

```r
cat("Course 2 Stroke Average:", course2_stroke_avg, "\n")
```

```
## Course 2 Stroke Average: NA
```

```r
cat("Difference in Stroke Averages:", stroke_diff, "\n")
```

```
## Difference in Stroke Averages: NA
```

```r
cat("Percentage Change in Stroke Averages:", stroke_percent_change, "%\n\n")
```

```
## Percentage Change in Stroke Averages: NA %
```

```r
cat("Course 1 Maintenance Cost:", course1_cost, "\n")
```

```
## Course 1 Maintenance Cost: 40808.32
```

```
cat("Course 2 Maintenance Cost:", course2_cost, "\n")
```

```
## Course 2 Maintenance Cost: 42808.32
```

```
cat("Difference in Maintenance Costs:", cost_diff, "\n")
```

```
## Difference in Maintenance Costs: 2000
```

```
cat("Percentage Change in Maintenance Costs:", cost_percent_change, "\n")
```

```
## Percentage Change in Maintenance Costs: 4.900962
```

```
#par3 simulator and cost analysis

library(tidyverse)

# Define the new golf simulator function
runGolfSimulator2 <- function(num_trials, Tee.Shot.Distance = 120, SD.Angle = 8,
                              Semicircle.Green.Radius = 25,
                              Center.to.Hazard.Edge = 30) {
  # Create Simulator data frame
  Simulator <- data.frame(
    Trial = 1:num_trials,
    Tee_Degrees = rep(NA, num_trials),
    Tee_Length = rep(NA, num_trials),
    Tee_Width = rep(NA, num_trials),
    Outcome = rep(NA, num_trials),
    Approach_Dist = rep(NA, num_trials),
    Score = rep(NA, num_trials)
  )

  for (i in 1:nrow(Simulator)) {
    Simulator$Tee_Degrees_no_abs[i] <- SD.Angle *
                                       (rnorm(1, mean = 0, sd = 1))
    Simulator$Tee_Width_no_abs[i] <- Tee.Shot.Distance *
                                     sin(Simulator$Tee_Degrees_no_abs[i] *
                                     (pi/180))
    Simulator$Tee_Degrees[i] <- SD.Angle * abs(rnorm(1, mean = 0, sd = 1))
    Simulator$Tee_Length[i] <- Tee.Shot.Distance *
                               cos(Simulator$Tee_Degrees[i] * (pi/180))
    Simulator$Tee_Length_no_abs[i] <- Tee.Shot.Distance *
                                      cos(Simulator$Tee_Degrees_no_abs[i] *
                                      (pi/180))
    Simulator$Tee_Width[i] <- Tee.Shot.Distance *
                              sin(Simulator$Tee_Degrees[i] * (pi/180))
    Simulator$Approach_Dist[i] <- sqrt((Tee.Shot.Distance -
                                       Simulator$Tee_Length[i])^2 +
                                       Simulator$Tee_Width[i]^2)
    Simulator$Outcome[i] <- ifelse(Simulator$Approach_Dist[i] <
                                   Semicircle.Green.Radius, "G",
                                   ifelse(Simulator$Tee_Width[i] <
```

```r
                                   Center.to.Hazard.Edge, "R", "H"))

    temp_score <- ifelse(Simulator$Outcome[i] == 'G',
                         Par3Out[which(Par3Out$Approach_Dist ==
                         floor(Simulator$Approach_Dist[i])), "G_ES"] + 1,
                         ifelse(Simulator$Outcome[i] == 'R',
                         Par3Out[which(Par3Out$Approach_Dist ==
                         floor(Simulator$Approach_Dist[i])), "R_ES"] + 1,
                         Par3Out[which(Par3Out$Approach_Dist ==
                         floor(Simulator$Approach_Dist[i])), "H_ES"] + 1))
    temp_score <- unlist(temp_score)
    Simulator$Score[i] <- temp_score[1]
  }

  mean_score <- mean(Simulator$Score)
  return(mean_score)
}

# Define the cost calculation function
calculate_maintenance_cost <- function(length_manicured_land,
                                       width_manicured_land, radius_green,
                                       length_fairway, width_fairway,
                                       length_tee_box, width_tee_box,
                                       length_bunker, width_bunker,
                                       cost_per_unit_square_green = 25.07,
                                       cost_per_unit_square_fairway = 0.36,
                                       cost_per_unit_square_tee_box = 3.03,
                                       cost_per_unit_square_bunker = 20.23,
                                       cost_per_unit_square_rough = 0.04) {
  area_manicured_land <- length_manicured_land * width_manicured_land
  area_green <- (pi * radius_green^2) / 2
  area_fairway <- length_fairway * width_fairway
  area_tee_box <- length_tee_box * width_tee_box
  area_bunker <- length_bunker * width_bunker
  area_rough <- area_manicured_land - area_green - area_fairway -
              area_tee_box - area_bunker

  total_cost_green <- area_green * cost_per_unit_square_green
  total_cost_fairway <- area_fairway * cost_per_unit_square_fairway
  total_cost_tee_box <- area_tee_box * cost_per_unit_square_tee_box
  total_cost_bunker <- area_bunker * cost_per_unit_square_bunker
  total_cost_rough <- area_rough * cost_per_unit_square_rough

  total_cost <- total_cost_green + total_cost_fairway +
              total_cost_tee_box + total_cost_bunker + total_cost_rough

  return(total_cost)
}

# Set parameters for each golf course
course1_params <- list(
  num_trials = 1000,
  Tee.Shot.Distance = 150,
```

```
  SD.Angle = 8,
  Semicircle.Green.Radius = 25,
  Center.to.Hazard.Edge = 30
)

course2_params <- list(
  num_trials = 1000,
  Tee.Shot.Distance = 170,
  SD.Angle = 8,
  Semicircle.Green.Radius = 25,
  Center.to.Hazard.Edge = 30
)

# Calculate stroke averages using the new function
course1_stroke_avg <- do.call(runGolfSimulator2, course1_params)
course2_stroke_avg <- do.call(runGolfSimulator2, course2_params)

# Set maintenance cost parameters for each course
cost_params_course1 <- list(
  length_manicured_land = 150, width_manicured_land = 60,
  radius_green = 25, length_fairway = 120, width_fairway = 40,
  length_tee_box = 10, width_tee_box = 10, length_bunker = 20,
  width_bunker = 10, cost_per_unit_square_green = 2,
  cost_per_unit_square_fairway = 1.5, cost_per_unit_square_tee_box = 1.2,
  cost_per_unit_square_bunker = 1.8, cost_per_unit_square_rough = 1
)

cost_params_course2 <- list(
  length_manicured_land = 170, width_manicured_land = 60,
  radius_green = 25, length_fairway = 120, width_fairway = 40,
  length_tee_box = 10, width_tee_box = 10, length_bunker = 20,
  width_bunker = 10, cost_per_unit_square_green = 2,
  cost_per_unit_square_fairway = 1.5, cost_per_unit_square_tee_box = 1.2,
  cost_per_unit_square_bunker = 1.8, cost_per_unit_square_rough = 1
)

# Calculate maintenance costs using the function
course1_cost <- do.call(calculate_maintenance_cost, cost_params_course1)
course2_cost <- do.call(calculate_maintenance_cost, cost_params_course2)

# Extract parameters for manicured land area calculation
length_manicured_land_course1 <- course1_params$Tee.Shot.Distance
width_of_manicured_land_course1 <- 2 * course1_params$Center.to.Hazard.Edge
course1_manicured_land <- length_manicured_land_course1 *
                          width_of_manicured_land_course1

length_manicured_land_course2 <- course2_params$Tee.Shot.Distance
width_of_manicured_land_course2 <- 2 * course2_params$Center.to.Hazard.Edge
course2_manicured_land <- length_manicured_land_course2 *
                          width_of_manicured_land_course2

# Compare stroke averages and costs
stroke_diff <- course2_stroke_avg - course1_stroke_avg
```

```r
cost_diff <- course2_cost - course1_cost
manicured_land_diff <- course2_manicured_land - course1_manicured_land

# Calculate percentage changes
stroke_percent_change <- (stroke_diff / course1_stroke_avg) * 100
cost_percent_change <- (cost_diff / course1_cost) * 100
manicured_land_percent_change <- (manicured_land_diff /
                                  course1_manicured_land) * 100

# Print results
cat("Course 1 Stroke Average:", course1_stroke_avg, "\n")
```

```
## Course 1 Stroke Average: 3.292255
```

```r
cat("Course 2 Stroke Average:", course2_stroke_avg, "\n")
```

```
## Course 2 Stroke Average: 3.397255
```

```r
cat("Difference in Stroke Averages:", stroke_diff, "\n")
```

```
## Difference in Stroke Averages: 0.105
```

```r
cat("Percentage Change in Stroke Averages:", stroke_percent_change, "%\n\n")
```

```
## Percentage Change in Stroke Averages: 3.189303 %
```

```r
cat("Course 1 Maintenance Cost:", course1_cost, "\n")
```

```
## Course 1 Maintenance Cost: 12561.75
```

```r
cat("Course 2 Maintenance Cost:", course2_cost, "\n")
```

```
## Course 2 Maintenance Cost: 13761.75
```

```r
cat("Difference in Maintenance Costs:", cost_diff, "\n")
```

```
## Difference in Maintenance Costs: 1200
```

```r
cat("Percentage Change in Maintenance Costs:", cost_percent_change, "%\n\n")
```

```
## Percentage Change in Maintenance Costs: 9.552811 %
```

```r
cat("Course 1 Manicured Land Area:", course1_manicured_land, "\n")
```

```
## Course 1 Manicured Land Area: 9000
```

```r
cat("Course 2 Manicured Land Area:", course2_manicured_land, "\n")
```

```
## Course 2 Manicured Land Area: 10200
```

```r
cat("Difference in Manicured Land Area:", manicured_land_diff, "\n")
```

```
## Difference in Manicured Land Area: 1200
```

```r
cat("Percentage Change in Manicured Land Area:",
    manicured_land_percent_change, "%\n")
```

```
## Percentage Change in Manicured Land Area: 13.33333 %
```