```
library(readxl)
library(dplyr)

March_Madness_Data <- read_excel("March Madness Data.xlsx")
teams <- March_Madness_Data

# Load and clean data
teams <- teams %>%
  filter(!is.na(NetRtg), !is.na(ORtg), !is.na(DRtg), !is.na(Luck)) %>%
  mutate(Score = (NetRtg * 0.4) + (ORtg * 0.3) - (DRtg * 0.2) + (Luck * 100 * 0.1)) %>%
  select(Team = `Team`, Seed, Region, NetRtg, ORtg, DRtg, Luck, Score)

# --- Create pairwise training data ---
matchups <- data.frame()

for (i in 1:(nrow(teams) - 1)) {
  for (j in (i + 1):nrow(teams)) {
    t1 <- teams[i, ]
    t2 <- teams[j, ]

    matchups <- bind_rows(matchups, data.frame(
      NetRtg_diff = t1$NetRtg - t2$NetRtg,
      ORtg_diff = t1$ORtg - t2$ORtg,
      DRtg_diff = t1$DRtg - t2$DRtg,
      Luck_diff = t1$Luck - t2$Luck,
      Winner = ifelse(t1$Score > t2$Score, 1, 0)
    ))
  }
}

# --- Train MLR (logistic) model ---
model <- glm(Winner ~ NetRtg_diff + ORtg_diff + DRtg_diff + Luck_diff,
             data = matchups, family = "binomial")

# --- Prediction function using model ---
# --- AI used to help determine how the winner was to be predicted ---
predict_winner <- function(t1, t2) {
  input <- data.frame(
    NetRtg_diff = t1$NetRtg - t2$NetRtg,
    ORtg_diff = t1$ORtg - t2$ORtg,
    DRtg_diff = t1$DRtg - t2$DRtg,
    Luck_diff = t1$Luck - t2$Luck
  )
  prob <- predict(model, input, type = "response")
  winner <- if (prob >= 0.5) t1 else t2
  return(winner)
}

# --- Fixed seed-based Round of 64 bracket logic ---
matchup_order <- list(
  c(1, 16), c(8, 9), c(5, 12), c(4, 13),
  c(6, 11), c(3, 14), c(7, 10), c(2, 15)
)

# --- Simulate region with model-based predictions ---
simulate_region <- function(region_name) {
  region_teams <- teams %>% filter(Region == region_name)

  simulate_round <- function(tlist, round_name) {
    winners <- data.frame()
    cat("\n", round_name, " -", region_name, "\n")

    if (round_name == "Round of 64") {
      for (pair in matchup_order) {
```

```
        t1 <- tlist %>% filter(Seed == pair[1]) %>% slice(1)
        t2 <- tlist %>% filter(Seed == pair[2]) %>% slice(1)
        if (nrow(t1) == 0 | nrow(t2) == 0) next
        win <- predict_winner(t1, t2)
        cat(t1$Team, "vs", t2$Team, "→", win$Team, "\n")
        winners <- bind_rows(winners, win)
      }
    } else {
      for (i in seq(1, nrow(tlist), by = 2)) {
        t1 <- tlist[i, ]
        t2 <- tlist[i + 1, ]
        win <- predict_winner(t1, t2)
        cat(t1$Team, "vs", t2$Team, "→", win$Team, "\n")
        winners <- bind_rows(winners, win)
      }
    }
    return(winners)
  }

  r64 <- simulate_round(region_teams, "Round of 64")
  r32 <- simulate_round(r64, "Round of 32")
  s16 <- simulate_round(r32, "Sweet 16")
  e8  <- simulate_round(s16, "Elite 8")
  return(e8)
}

# --- Simulate all regions ---
south   <- simulate_region("South")
east    <- simulate_region("East")
midwest <- simulate_region("Midwest")
west    <- simulate_region("West")

# --- Final Four + Championship ---
ff <- bind_rows(south, east, midwest, west)

cat("\n Final Four:\n")
print(ff[, c("Team", "Region", "Seed")])

sf1 <- predict_winner(south, west)
sf2 <- predict_winner(midwest, east)

cat("\n Semifinals:\n")
cat(south$Team, "vs", west$Team, "→", sf1$Team, "\n")
cat(midwest$Team, "vs", east$Team, "→", sf2$Team, "\n")

champion <- predict_winner(sf1, sf2)

cat("\n National Champion:", champion$Team, "(Region:", champion$Region, ", Seed:",
champion$Seed, ")\n")
```