```r
# The state of a game is captured in a numeric vector of length 5:
# the number of red, green, blue and yellow fruits left, and the number of
# steps the raven has taken.
# We allow for two game parameters:
# - fruit_count: the number of fruit of each type at the start of the game.
#   Original game has fruit_count = 4.
# - max_raven_steps: the number of steps the raven must take in order for it
#   to win. Original game has max_raven_steps = 5.
#
# The starting game state is
#     (fruit_count, fruit_count, fruit_count, fruit_count, 0).
# The game ends when all of the first 4 entries are 0, or when the 5th entry
# is equal to max_raven_steps.

library(tidyverse)

SimulateTurn <- function(state, verbose = FALSE) {
  # roll dice
  roll <- sample(6, size = 1)

  # (1-4: red, green, blue, yellow, 5: raven, 6: fruit basket)
  if (roll <= 4) {
    state[roll] <- max(state[roll] - 1, 0)
  } else if (roll == 5) {
    state[5] <- state[5] + 1
  } else {
    # remove the color that has the most fruit remaining
    idx <- which.max(state[1:4])
    state[idx] <- max(state[idx] - 1, 0)
  }

  if (verbose)
    cat(paste("Roll:", roll, ", State:", paste(state, collapse = ",")),
        fill = TRUE)

  return(state)
}

CheckGameState <- function(state, max_raven_steps) {
  if (sum(state[1:4]) == 0) {
    return(1)  # players win
  } else if (state[5] >= max_raven_steps) {
    return(2)  # raven wins
  } else {
    return(3)  # game ongoing
  }
}

SimulateGame <- function(fruit_count, max_raven_steps, verbose = FALSE) {
  # set beginning state
  state <- rep(fruit_count, length.out = 4)
  state <- c(state, 0)

  num_turns <- 0  # total number of turns for the game
  outcome <- CheckGameState(state, max_raven_steps)
  while (outcome == 3) {
    num_turns <- num_turns + 1
    state <- SimulateTurn(state, verbose = verbose)
    outcome <- CheckGameState(state, max_raven_steps)
  }

  # print game info
  if (verbose) {
    cat(switch(outcome, "Players win", "Raven wins", "Game ongoing"),
        fill = TRUE)
    cat(paste("# of turns:", num_turns), fill = TRUE)
    cat(paste("# of steps raven took:", state[5]), fill = TRUE)
    cat(paste("# fruit left:", sum(state[1:4])), fill = TRUE)
  }

  # return 4 interesting game stats:
  # outcome, # of turns taken, # steps raven took, # fruit left
  return(c(outcome, num_turns, state[5], sum(state[1:4])))
}

#####
# ONE SAMPLE RUN
#####
set.seed(1)
results <- SimulateGame(fruit_count = 2, max_raven_steps = 3, verbose = TRUE)
# Roll: 1 , State: 1,2,2,2,0
# Roll: 4 , State: 1,2,2,1,0
# Roll: 1 , State: 0,2,2,1,0
# Roll: 2 , State: 0,1,2,1,0
# Roll: 5 , State: 0,1,2,1,1
# Roll: 3 , State: 0,1,1,1,1
# Roll: 6 , State: 0,0,1,1,1
# Roll: 2 , State: 0,0,1,1,1
# Roll: 3 , State: 0,0,0,1,1
# Roll: 3 , State: 0,0,0,1,1
# Roll: 1 , State: 0,0,0,1,1
# Roll: 5 , State: 0,0,0,1,2
# Roll: 5 , State: 0,0,0,1,3
# Raven wins
# # of turns: 13
# # of steps raven took: 3
# # fruit left: 1

#####
# SIMULATION: probability of winning the game
# We allow number of fruit of each color to vary from 1 to 8,
```

```r
# And the the number of steps the raven must take from 1 to 8.
#####
nsim <- 10000 # no. of simulation runs for each param setting

sim_df <- expand.grid(fruit_count = 1:8, max_raven_steps = 1:8)
sim_df$win_prob <- NA
set.seed(1)
system.time({
  for (row_idx in 1:nrow(sim_df)) {
    # take just the win outcome of the simulation
    results <- replicate(
      nsim,
      SimulateGame(fruit_count = sim_df$fruit_count[row_idx],
                   max_raven_steps = sim_df$max_raven_steps[row_idx])[1])
    sim_df$win_prob[row_idx] <- sum(results == 1) / nsim
  }
})
# saveRDS(sim_df, "sim_df1.rds")

ggplot(sim_df, aes(x = fruit_count, y = max_raven_steps)) +
  geom_tile(aes(fill = win_prob)) +
  geom_text(aes(label = round(win_prob, 3))) +
  labs(title = "Win probability", x = "# Fruit of each color",
       y = "# Steps for raven to win") +
  scale_fill_gradient2(low = "red", high = "blue", midpoint = 0.5) +
  theme_bw() +
  theme(legend.position = "null")

#####
# SIMULATION: # of steps in the game
#####
fruit_count <- 4
max_raven_steps <- 5
nsim <- 1e5

set.seed(2)
system.time({
  results <- replicate(nsim, SimulateGame(fruit_count, max_raven_steps))
})
sim_df <- data.frame(t(results))
names(sim_df) <- c("outcome", "num_turns", "raven_steps", "fruit_left")
# saveRDS(sim_df, "sim_df2.rds")

# max & min of turns
range(sim_df$num_turns)
table(sim_df$num_turns)

# overall histogram for num_turns
ggplot(sim_df) +
  geom_histogram(aes(x = num_turns), binwidth = 1, center = 0,
                 fill = "white", col = "black") +
  labs(title = "Histogram for # of turns", x = "# of turns",
       y = "Frequency") +
  theme_bw()

# histogram for num_turns by outcome
sim_df %>%
  mutate(outcome = fct_recode(as.character(outcome),
                              `Players win` = "1", `Raven wins` = "2")) %>%
  ggplot() +
  geom_histogram(aes(x = num_turns), binwidth = 1, center = 0,
                 fill = "white", col = "black") +
  labs(title = "Histogram for # of turns", x = "# of turns",
       y = "Frequency") +
  facet_wrap(~ outcome) +
  theme_bw()

# mean & median by outcome
sim_df %>%
  mutate(outcome = fct_recode(as.character(outcome),
                              `Players win` = "1", `Raven wins` = "2")) %>%
  group_by(outcome) %>%
  summarize(mean = mean(num_turns), median = median(num_turns))

#####
# SIMULATION: # of steps raven takes in losing games
# We can use sim_df from the previous section
#####
raven_lose_df <- filter(sim_df, outcome == 1)
ggplot(raven_lose_df) +
  geom_histogram(aes(x = raven_steps, y = stat(count / sum(count))),
                 binwidth = 1, center = 0, fill = "white", col = "black") +
  geom_vline(xintercept = mean(raven_lose_df$raven_steps),
             col = "red", size = 1.5) +
  labs(title = "Histogram for # of raven steps (when raven loses)",
       x = "# of raven steps", y = "Proportion") +
  theme_bw()

#####
# SIMULATION: # of fruit left in raven winning games
# We can use sim_df from the previous section
#####
raven_win_df <- filter(sim_df, outcome == 2)
summary_df <- raven_win_df %>%
  summarize(mean = mean(fruit_left), median = median(fruit_left)) %>%
  pivot_longer(mean:median)
ggplot(raven_win_df) +
  geom_histogram(aes(x = fruit_left, y = stat(count / sum(count))),
                 binwidth = 1, center = 0, fill = "white", col = "black") +
  geom_vline(data = summary_df, aes(xintercept = value, col = name),
             size = 1.5) +
  labs(title = "Histogram for # of fruit left (when raven wins)",
       x = "# of fruit left", y = "Proportion") +
  theme_bw() +
  theme(legend.position = "bottom", legend.title = element_blank())

#####
# SIMULATION: # of steps to harvest all fruit
#####
fruit_count <- 4
max_raven_steps <- Inf
nsim <- 1e5

set.seed(2)
system.time({
  results <- replicate(nsim, SimulateGame(fruit_count, max_raven_steps))
})
sim_df <- data.frame(t(results))
names(sim_df) <- c("outcome", "num_turns", "raven_steps", "fruit_left")

# max & min of turns
range(sim_df$num_turns)
sort(table(sim_df$num_turns))
mean(sim_df$num_turns)

ggplot(sim_df) +
  geom_histogram(aes(x = num_turns), binwidth = 1, center = 0,
               fill = "white", col = "black") +
  labs(title = "Histogram for # of turns (inf steps for raven)", x = "# of turns",
       y = "Frequency") +
  theme_bw()
```