Markov Transition Modeling

This investigation builds and analyzes two Markov chains: 1) a reservoir supply system, and 2) a Monopoly board movement model.

```
set.seed(1234)
```

Reservoir supply Markov chain

We model weekly water storage as a Markov chain where the state is the volume of water available at the beginning of each week. The dam capacity is 10 units. Weekly inflow is random with probabilities: $P(I_t = 0) = 0.3$, $P(I_t = 1) = 0.1$, $P(I_t = 2) = 0.3$, $P(I_t = 3) = 0.2$, $P(I_t = 4) = 0.1$. Demand is 2 units each week and must be supplied at the start of the week. If demand exceeds available water, the shortage is purchased at cost 5 per unit. The dam also has a fixed depreciation cost of 2 per week.

State space: $S = 0, 1, 2, ..., 10$where state i means i units available.

State update rule: after supplying demand, the remaining amount is $max(0, i - 2)$, then inflow arrives and the new state is $X_{t+1} = min(10, max(0, X_t - 2) + I_t)$.

We build the transition matrix from this rule, compute the steady-state distribution, and estimate long-run cost behavior through simulation.

```
states <- 0:10
inflow_probs <- c(0.3, 0.1, 0.3, 0.2, 0.1)
names(inflow_probs) <- 0:4

P_res <- matrix(0, nrow = 11, ncol = 11, dimnames = list(states, states))
for (i in states) {
  base <- max(0, i - 2)
  for (k in 0:4) {
    j <- min(10, base + k)
    P_res[i + 1, j + 1] <- P_res[i + 1, j + 1] + inflow_probs[as.character(k)]
  }
}
P_res
```

```
##      0   1   2   3   4   5   6   7   8   9  10
## 0  0.3 0.1 0.3 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.0
## 1  0.3 0.1 0.3 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.0
## 2  0.3 0.1 0.3 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.0
## 3  0.0 0.3 0.1 0.3 0.2 0.1 0.0 0.0 0.0 0.0 0.0
## 4  0.0 0.0 0.3 0.1 0.3 0.2 0.1 0.0 0.0 0.0 0.0
## 5  0.0 0.0 0.0 0.3 0.1 0.3 0.2 0.1 0.0 0.0 0.0
## 6  0.0 0.0 0.0 0.0 0.3 0.1 0.3 0.2 0.1 0.0 0.0
## 7  0.0 0.0 0.0 0.0 0.0 0.3 0.1 0.3 0.2 0.1 0.0
## 8  0.0 0.0 0.0 0.0 0.0 0.0 0.3 0.1 0.3 0.2 0.1
## 9  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.3 0.1 0.3 0.3
## 10 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.3 0.1 0.6
```

```
A <- t(P_res) - diag(11)
A <- rbind(A, rep(1, 11))
```

```
RHS <- c(rep(0, 11), 1)
pi_res <- qr.solve(A, RHS)
pi_res
```

```
##          0          1          2          3          4          5          6
## 0.11218209 0.08838623 0.17337199 0.16997399 0.14730832 0.09820967 0.07302009
##          7          8          9         10
## 0.04995233 0.03915354 0.02208765 0.02635412
```

```
cost_vec <- ifelse(states == 0, 12, ifelse(states == 1, 7, 2))
long_run_cost <- sum(cost_vec * pi_res)
long_run_cost
```

```
## [1] 3.563752
```

```
n_weeks <- 100000
X <- numeric(n_weeks)
X[1] <- 1
for (t in 1:(n_weeks - 1)) {
  X[t + 1] <- sample(states, size = 1, prob = P_res[X[t] + 1, ])
}

weekly_costs <- cost_vec[X + 1]
annual_costs <- sapply(seq(1, n_weeks - 51, by = 52), function(k) {
  sum(weekly_costs[k:(k + 51)])
})
prob_annual_cost_gt_250 <- mean(annual_costs > 250)
prob_annual_cost_gt_250
```

```
## [1] 0.06448258
```

The long-run annual-cost exceeding probability is small, indicating that high annual costs are relatively rare.

Monopoly board Markov chain

We model player position as a Markov chain on a 40-space board. The state space is $1, ..., 40$ where space 1 is "Go", space 11 is "Jail", and space 31 is "Go to Jail". Movement is determined by the sum of two dice, with probabilities given by the standard 2–12 distribution.

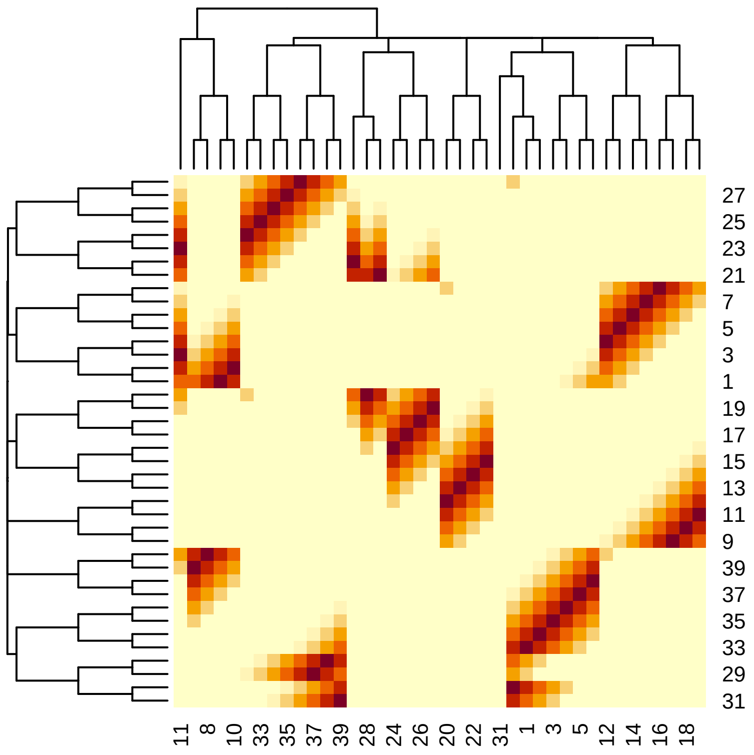
Rules used in this simplified model:

- If a player lands on space 31, they move to space 11 on the next move.
- Jail is a one-turn delay: the player always leaves jail on their next move.
- Movement wraps around the board: next $= (i + roll - 1 mod 40) + 1$.

We build the 40x40 transition matrix, compute steady-state probabilities, and evaluate long-run revenue and cost under two hotel placement scenarios.

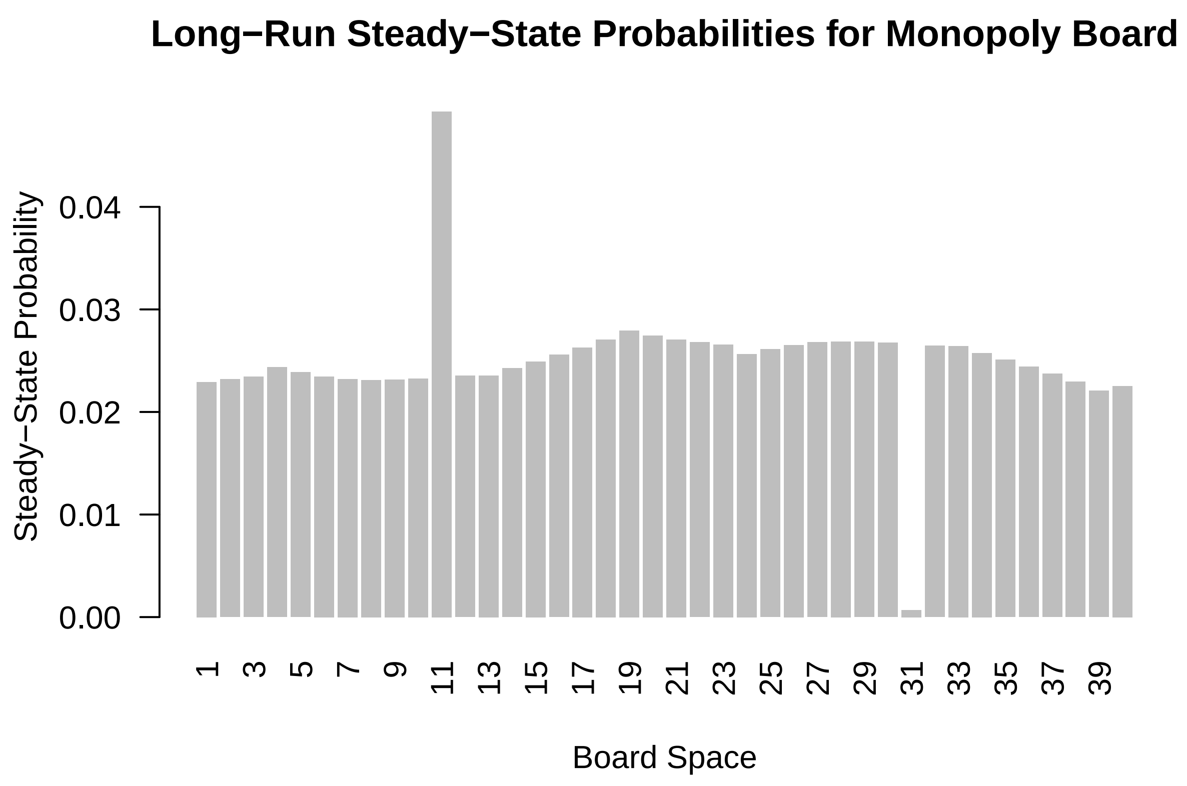
```
dice_prob <- c(0, 0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1) / 36
P_board <- matrix(0, 40, 40)

for (i in 1:40) {
  for (roll in 2:12) {
    prob <- dice_prob[roll]
    j <- (i + roll - 1) %% 40 + 1
    if (j == 31) j <- 11
    if (i == 11) {
      j <- (11 + roll - 1) %% 40 + 1
      if (j == 31) j <- 11
    }
    P_board[i, j] <- P_board[i, j] + prob
  }
}
heatmap(P_board)
```



```
A <- t(P_board) - diag(40)
A <- rbind(A, rep(1, 40))
RHS <- c(rep(0, 40), 1)
PI <- qr.solve(A, RHS)
names(PI) <- 1:40

barplot(
  PI,
  names.arg = 1:40,
  main = "Long-Run Steady-State Probabilities for Monopoly Board",
  xlab = "Board Space",
  ylab = "Steady-State Probability",
  border = NA,
  las = 2
)
```



```r
print("Example: Probability of landing on Space 40")
```

```
## [1] "Example: Probability of landing on Space 40"
```

```r
PI["40"]
```

```
##         40
## 0.02254028
```

```r
print("Most visited spaces:")
```

```
## [1] "Most visited spaces:"
```

```r
sort(PI, decreasing = TRUE)[1:5]
```

```
##         11         19         20         21         18
## 0.04930460 0.02793460 0.02745875 0.02707429 0.02706007
```

Scenario A: Hotels on 17, 19, 20 vs opponent on 38, 40

```r
revenue_vec <- numeric(40)
revenue_vec[c(17, 19)] <- 950
revenue_vec[20] <- 1000

cost_vec <- numeric(40)
cost_vec[38] <- 1500
cost_vec[40] <- 2000
```

```
E_rev <- sum(revenue_vec * PI)
E_cost <- sum(cost_vec * PI)
profit <- E_rev - E_cost
E_rev
```

```
## [1] 78.96147
```

```
E_cost
```

```
## [1] 79.48901
```

```
profit
```

```
## [1] -0.5275473
```

```
Var_rev <- sum(PI * revenue_vec^2) - E_rev^2
Var_cost <- sum(PI * cost_vec^2) - E_cost^2
Var_profit <- Var_rev + Var_cost
SD_profit <- sqrt(Var_profit)
Var_rev
```

```
## [1] 70151.42
```

```
Var_cost
```

```
## [1] 135455.3
```

```
Var_profit
```

```
## [1] 205606.7
```

```
SD_profit
```

```
## [1] 453.4388
```

Variance is large because gains and losses are infrequent but very large when they happen, producing highly volatile turn-by-turn profit.

Scenario B: Additional hotels on 22, 24, 25 vs opponent on 32, 33, 35

```
revenue_vec2 <- numeric(40)
revenue_vec2[c(17, 19)] <- 950
revenue_vec2[20] <- 1000
revenue_vec2[22] <- 1050
revenue_vec2[24] <- 1050
revenue_vec2[25] <- 1100

cost_vec2 <- numeric(40)
cost_vec2[38] <- 1500
cost_vec2[40] <- 2000
cost_vec2[32] <- 1275
cost_vec2[33] <- 1275
cost_vec2[35] <- 1400

E_rev_2 <- sum(revenue_vec2 * PI)
E_cost_2 <- sum(cost_vec2 * PI)
profit_2 <- E_rev_2 - E_cost_2

Var_rev2 <- sum(PI * revenue_vec2^2) - E_rev_2^2
```

```
Var_cost2 <- sum(PI * cost_vec2^2) - E_cost_2^2
Var_profit2 <- Var_rev2 + Var_cost2
SD_profit2 <- sqrt(Var_profit2)

E_rev_2
```

```
## [1] 162.7698
```

```
E_cost_2
```

```
## [1] 182.053
```

```
profit_2
```

```
## [1] -19.28314
```

```
Var_rev2
```

```
## [1] 139328.1
```

```
Var_cost2
```

```
## [1] 243788.3
```

```
Var_profit2
```

```
## [1] 383116.4
```

```
SD_profit2
```

```
## [1] 618.9639
```

With additional hotels, expected revenue rises but expected cost rises more, so expected profit becomes negative and variability remains high.

Game simulation (win probability)

We simulate concurrent chains for both players to estimate win probability.

```
next_state <- function(state, P) {
  sample(1:40, size = 1, prob = P[state, ])
}

simulate_game <- function(P, rev_vec, cost_vec, cash0 = 5000) {
  pos_me <- 1
  pos_opp <- 1
  cash_me <- cash0
  cash_opp <- cash0
  repeat {
    pos_me <- next_state(pos_me, P)
    cash_me <- cash_me + rev_vec[pos_opp] - cost_vec[pos_me]
    if (cash_me <= 0) return(0)
    pos_opp <- next_state(pos_opp, P)
    cash_opp <- cash_opp + rev_vec[pos_me] - cost_vec[pos_opp]
    if (cash_opp <= 0) return(1)
  }
}
```

```
N <- 1000
wins <- numeric(N)
for (i in 1:N) {
```

```r
  wins[i] <- simulate_game(P_board, revenue_vec, cost_vec)
}
prob_win_a <- mean(wins)
prob_win_a
```

```
## [1] 0.489
```

```r
wins <- numeric(N)
for (i in 1:N) {
  wins[i] <- simulate_game(P_board, revenue_vec2, cost_vec2)
}
prob_win_b <- mean(wins)
prob_win_b
```

```
## [1] 0.469
```

Win rates are roughly even in Scenario A (48.9%) and lower in Scenario B (around 46.9%).