INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

ROBERT GORDON UNIVERSITY ABERDEEN

BSc. Artificial Intelligence & Data Science

CM2605

Simulations and Modelling Techniques

Antoinette Bonifacia Duweeja de Lima.

IIT ID: 20210522

RGU ID: 2117517

ROBERT GORDON UNIVERSITY ABERDEEN TEF Gold

INFORMATICS INSTITUTE OF TECHNOLOGY

Table of Contents

ROBERT GORDON UNIVERSITY ABERDEEN

INFORMATICS INSTITUTE OF TECHNOLOGY

Question 1:

BMI Distribution

20210522 - Antoinette Bonifacia Duweeja de Lima.

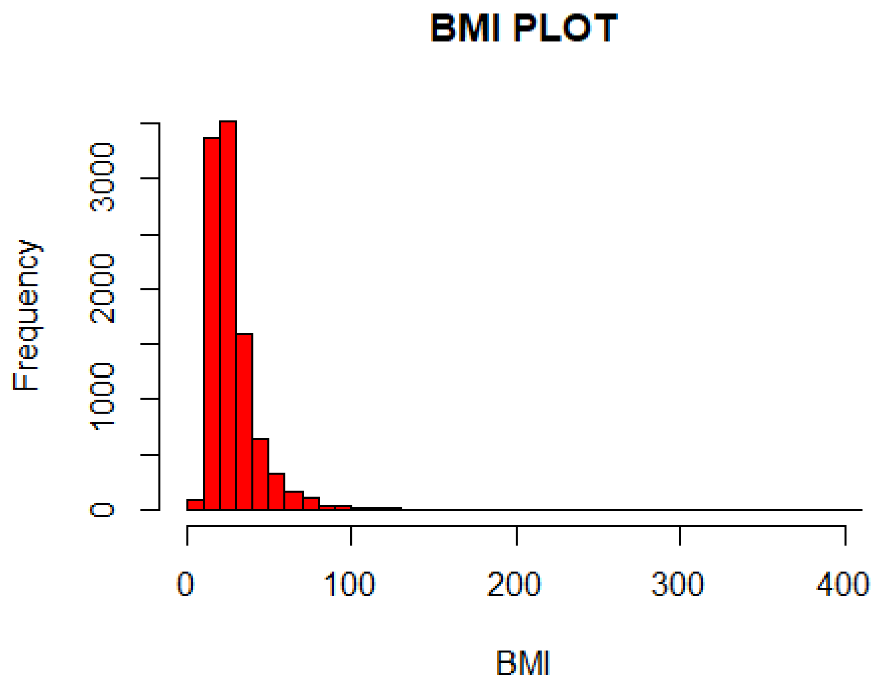
4/15/2023

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this: ated the plot.

```
#Question 1)
#a)
#set seed for reproducibility.
set.seed(123)
#generating 10000 weight values from N(60, 32).
w <- rnorm(10000, mean = 60, sd = 4)
#generate 10000 height values from N(1.6, 0.12).
h <- rnorm(10000, mean = 1.6, sd = 0.3464)
#calculating the BMI values.
bmi <- w / h^2

#b)
#plotting the histogram of BMI values.
hist(bmi, breaks = 50, col = "red", xlab = "BMI", main = "BMI PLOT")
```




```
#c)
#calculating the mean and the variance of BMI.
mean(bmi)

## [1] 28.00382

var(bmi)

## [1] 331.5486

#d)
#estimating P(BMI >= 25).
mean(bmi >= 25)

## [1] 0.4471
```

Question 2:

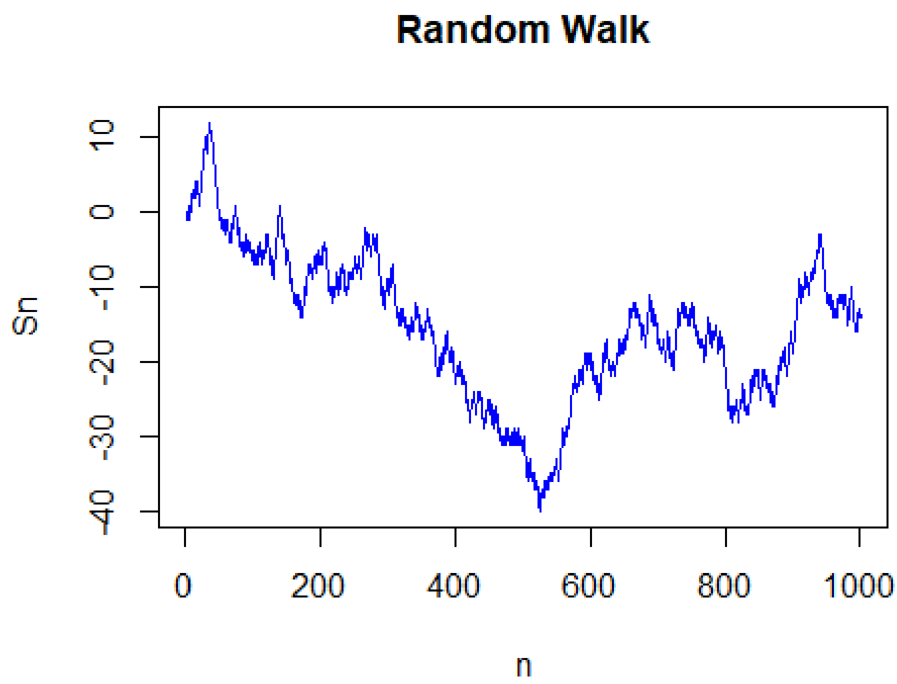
```
#Question2)
#a)
#Set seed for reproducibility.
set.seed(123)

#Defining probabilities for winning a point under each service rule.
p_win_A <- 0.55
p_win_B <- 0.40

#Function to simulate a game.
simulate_game <- function(p_win_serve_A, p_win_serve_B) {
  score_player1 <- 0
  score_player2 <- 0

  #Simulating game until one player wins 2 points.
  while (score_player1 < 2 & score_player2 < 2) {
    if (sample(c(TRUE, FALSE), 1, prob = c(p_win_serve_A, 1 - p_win_serve_A))
) {
      score_player1 <- score_player1 + 1
    } else {
      score_player2 <- score_player2 + 1
    }
  }

  return(score_player1 > score_player2)
}

#Simulating 1000 games under service rule A.
n <- 1000
winners_A <- replicate(n, simulate_game(p_win_serve_A = p_win_A, p_win_serve_
B = p_win_A))

#Simulating 1000 games under service rule B.
winners_B <- replicate(n, simulate_game(p_win_serve_A = p_win_B, p_win_serve_
B = p_win_B))

#Calculating the winning probabilities under each service rule.
winning_prob_A <- mean(winners_A)
winning_prob_B <- mean(winners_B)

#Printing the results.
cat("a) Winning probability of Player 1 under service rule A:", winning_prob_
A, "\n")
```

```
## a) Winning probability of Player 1 under service rule A: 0.588

cat("a) Winning probability of Player 1 under service rule B:", winning_prob_
B, "\n")

## a) Winning probability of Player 1 under service rule B: 0.355

#b)
#Function to simulate a game and return the number of points played.
simulate_game_length <- function(p_win_serve_A, p_win_serve_B) {
  score_player1 <- 0
  score_player2 <- 0
  num_points <- 0

  #Simulating game until one player wins 2 points.
  while (score_player1 < 2 & score_player2 < 2) {
    num_points <- num_points + 1
    if (sample(c(TRUE, FALSE), 1, prob = c(p_win_serve_A, 1 - p_win_serve_A))
) {
      score_player1 <- score_player1 + 1
    } else {
      score_player2 <- score_player2 + 1
    }
  }

  return(num_points)
}

#Simulating 1000 games and calculate the expected length of a game under serv
ice rule A.
#Number of games to simulate.
n <- 1000
#Probability of winning a point under service rule A.
p_win_A <- 0.55

game_lengths_A <- replicate(n, simulate_game_length(p_win_serve_A = p_win_A,
p_win_serve_B = p_win_A))
expected_length_A <- mean(game_lengths_A)

#Simulating 1000 games and calculate the expected length of a game under serv
ice rule B.
#Probability of winning a point under service rule B.
p_win_B <- 0.40

game_lengths_B <- replicate(n, simulate_game_length(p_win_serve_A = p_win_B,
p_win_serve_B = p_win_B))
expected_length_B <- mean(game_lengths_B)

#Printing the results.
```

```r
cat("b) Expected length of a game under service rule A:", expected_length_A,
"points\n")

## b) Expected length of a game under service rule A: 2.511 points

cat("b) Expected length of a game under service rule B:", expected_length_B,
"points\n")

## b) Expected length of a game under service rule B: 2.514 points

#c)
#Based on the simulation results, the following is the estimated length of a
game for each service rule:
# - The expected length of a game under service rule A is projected to be exp
ected_length_A points (where the server is the winner of the preceding point)
. This means that when service rule A is followed, it takes roughly expected_
length_A points for one player to win 2 points and the game.
# - The expected length of a game under service rule B is calculated to be ex
pected_length_B points (where the server is the loser of the preceding point)
. This suggests that when service rule B is followed, it takes about expected
_length_B points for one player to win 2 points and win the game.

#d)
#1)Player performance consistency: The simulation assumes that the players' p
erformance is consistent throughout the game and that the probability of earn
ing a point under each service rule appropriately represent their actual perf
ormance. If player performance varies over time or due to external circumstan
ces such as fatigue or injury, the simulation findings may be invalid.
#2)Players with equal skill levels: The simulation assumes that both players
have equal skill levels, with the only difference being the service rule bein
g followed. If the participants have major skill discrepancies, the results m
ay not correctly represent the game's outcomes. In actuality, player skill le
vels might fluctuate, influencing game dynamics and outcomes.
#3)Point independence: The simulation assumes that the outcome of each point
is independent of prior points, which means that the chances of winning a poi
nt do not alter based on the outcome of previous points. This may not always
be the case in real-world settings, as players may be swayed by momentum, psy
chological variables, or strategy alterations based on recent outcomes.
```

Question 3:

```
#Question3)
#a)
#Setting the parameters.
#number of steps.
n <- 1000
#probability of +1 step.
p <- 0.5
#probability of -1 step.
q <- 0.5

#Initializing the random walk.
#vector to store the random walk.
S <- rep(0, n+1)
#set seed for reproducibility.
set.seed(123)

#Simulating the random walk.
for (i in 1:n) {
  #Generating a random step (+1 or -1).
  step <- sample(c(1, -1), size = 1, prob = c(p, q))

  #Updating the random walk.
  S[i+1] <- S[i] + step
}

#Ploting the random walk.
plot(1:(n+1), S, type = "l", xlab = "n", ylab = "Sn", main = "Random Walk",co
l = "blue")
```




```
#b)
#As n goes to infinity, the random variable Sn will display the following pro
perties:

#Random walk behavior: The random walk Sn, n 1 may display particular tendenc
ies as n goes to infinity, such as becoming more symmetric and centered aroun
d 0, with the absolute values of the steps becoming less relevant relative to
the overall behavior of the random walk. Depending on the precise probabiliti
es of the steps and their relationship to one another, the random walk may al
so display qualities such as recurrence or transience.

#Distribution: Due to the central limit theorem, when n goes to infinity, the
distribution of Sn may converge to a Gaussian distribution (also known as a n
ormal distribution).The central limit theorem says that the sum of a large nu
mber of independent and identically distributed random variables tends to be
regularly distributed, regardless of the form of the initial distribution.

#Mean: Sn's mean can be computed by multiplying the expected value of each in
dividual random variable by n, given that the expected value exists. Because
Xn has a mean of 0 (due to the equal odds of +1 and -1 steps), the mean of Sn
will be 0 for all values of n.

#Variance: Sn's variance can be computed by adding the variances of each indi
vidual random variable and multiplying by n, given that the variances exist.
In this situation, because Xn has variance 1 (owing to equal probabilities of
+1 and -1 steps), Sn has variance n for all values of n.
```

Question 4:

```
#Question4)
#a)
#Initializing transition probability matrix.
P <- matrix(c(0.1, 0.2, 0.7, 0.2, 0.4, 0.4, 0.1, 0.3, 0.6), nrow = 3)

#Initialize state space and current state.
states <- c("A", "B", "C")
#1 represents state A.
current_state <- 1

#Simulate 5 possible beer purchases for 10 weeks.
for (i in 1:5) {
  cat(paste0("Simulation ", i, ":\n"))
  for (j in 1:10) {
    cat(paste0("Week ", j, ": ", states[current_state], "\n"))
    current_state <- sample(1:3, size = 1, prob = P[current_state, ])
  }
  cat("\n")
}

## Simulation 1:
## Week 1: A
## Week 2: B
## Week 3: C
## Week 4: A
## Week 5: A
## Week 6: A
## Week 7: B
## Week 8: C
## Week 9: A
## Week 10: B
##
## Simulation 2:
## Week 1: B
## Week 2: C
## Week 3: A
## Week 4: B
## Week 5: C
## Week 6: A
## Week 7: A
## Week 8: C
## Week 9: C
## Week 10: C
##
## Simulation 3:
```

```
## Week 1: A
## Week 2: B
## Week 3: A
## Week 4: C
## Week 5: A
## Week 6: A
## Week 7: B
## Week 8: C
## Week 9: C
## Week 10: A
##
## Simulation 4:
## Week 1: C
## Week 2: A
## Week 3: C
## Week 4: C
## Week 5: A
## Week 6: C
## Week 7: C
## Week 8: A
## Week 9: C
## Week 10: C
##
## Simulation 5:
## Week 1: C
## Week 2: B
## Week 3: C
## Week 4: C
## Week 5: C
## Week 6: A
## Week 7: B
## Week 8: B
## Week 9: B
## Week 10: A

#b)
#Initializing transition probability matrix.
P <- matrix(c(0.1, 0.2, 0.7, 0.2, 0.4, 0.4, 0.1, 0.3, 0.6), nrow = 3)

#Identifying starting state as A.
#1 represents state A.
start_state <- 1

#Calculating probability of transitioning from A to A in four steps.
prob_A_to_A_4steps <- as.numeric(t(P[start_state, ])) %% P %% P %*% P[start_s
tate, ]

#Multiplying initial probability of being in state A with probability of tran
sitioning from A to A in four steps.
prob_A_A_5weeks <- 1 * prob_A_to_A_4steps
```

```
#Print the results.
cat(prob_A_A_5weeks)

## 0.02 0.06 0.04
```

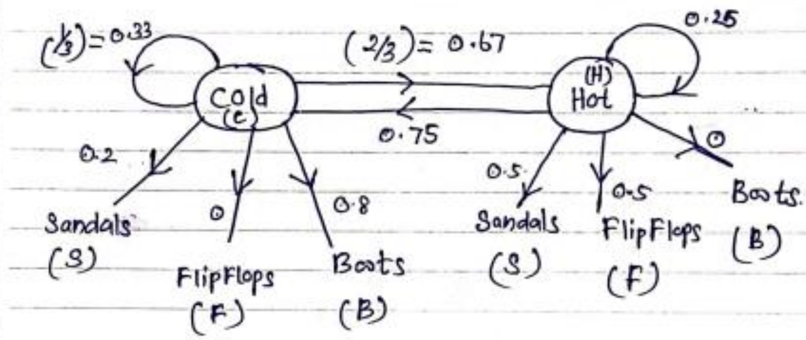
Question 5:

```r
#Question5)
#a)The state space of this HMM would be {hot, cold} representing the two poss
ible weather states of this Model because those 2 are the hidden states in th
is HMM these are the 2 states of the weather and the emission which is not hi
dden will be E={Sandals,Flipflops,Boots}

#b)
#Initializing transition probability matrix P.
P <- matrix(c(0.25, 0.75, 0.25, 0.67, 0.33, 0.33), nrow = 2, byrow = TRUE)

#Printing the transition probability matrix P.
print(P)

##      [,1] [,2] [,3]
## [1,] 0.25 0.75 0.25
## [2,] 0.67 0.33 0.33

#c)
#Initializing transition probability matrix P.
P <- matrix(c(0.25, 0.75, 0.25, 0.67, 0.33, 0.33), nrow = 2, byrow = TRUE)

#Probability of wearing sandals today (state 2, choice 1).
P_sandals_today <- P[2, 1]

#Probability of transitioning from state 2 to state 2 (cold to cold).
P_cold_to_cold <- P[2, 2]

#Probability of wearing sandals tomorrow (state 2, choice 1).
P_sandals_tomorrow <- P_sandals_today * P_cold_to_cold

#Printing the result.
cat("Probability of wearing sandals both today and tomorrow:", P_sandals_tomo
rrow)

## Probability of wearing sandals both today and tomorrow: 0.2211
```

Date   No

Question 5)

$P(A \cap B) = P(A) \cdot P(B)$



~~X = { "Cold",~~

$X = \{ \text{Hot}, \text{Cold} \}$

$S = \{ \text{Sandals}, \text{FlipFlops}, \text{Boots} \}$

Cold

$P(S|C) = 0.2$
$P(B|C) = 0.8$
$P(F|C) = 0$

Hot

$P(S|H) = 0.5$
$P(B|H) = 0$
$P(F|H) = 0.5$

$$P = \begin{array}{c|cc} & H & C \\ \hline H & 0.25 & 0.75 \\ C & 0.67 & 0.33 \end{array}$$