STAT 230 CAA #1

Aania Raheem

a4raheem

20870815

Question 1

STAT 230 Computer Assisted Assignment 1

Q1 a) A – Participant passes screening

B – Participant is a suitable partner.

- Sensitivity: P (Passes screening given that they are suitable)

$\therefore$ Sensitivity $= P(A|B)$

- Specificity: P (Doesn't pass screening given that they're not suitable)

$\therefore$ Specificity $= P(\bar{A}|\bar{B})$

b) Probability that a randomly selected Plenty of fisher member passes the screening.

This is $P(A)$.

$P(A) = P(A \cap B) + P(A \cap \bar{B})$

As $P(A|B) = \dfrac{P(A \cap B)}{P(B)} \Rightarrow P(A \cap B) = P(A|B) \times P(B)$.

$= 0.95 \times 1/1000$

$= 0.00095$

As $P(A|\bar{B}) = \dfrac{P(A \cap \bar{B})}{P(\bar{B})} \Rightarrow P(A \cap \bar{B}) = P(A|\bar{B}) \times P(\bar{B})$

$= 0.01 \times (1 - 1/1000)$

$= 0.00999$

$P(A) = 0.00095 + 0.00999 = 0.01094$

c) Probability that a Plenty of fisher member is a suitable partner given that they passed the screening.

This is $P(B|A)$

$P(B|A) = \dfrac{P(B \cap A)}{P(A)} \Rightarrow \dfrac{P(A \cap B)}{P(A)} = \dfrac{0.00095}{0.01094}$

$= 0.08684$

d) i) $P(\text{pass2}|\text{pass1})$

$P(\text{pass2}|\text{pass1}) = \dfrac{P(\text{pass2} \cap \text{pass1})}{P(\text{pass1})}$

$P(\text{pass1}) = P(\text{pass2}) = P(A)$ * as each attemp has the same variables.

$P(\text{pass2}|\text{pass1}) = \dfrac{P(\text{pass2}) \times P(\text{pass1})}{P(\text{pass1})}$ * as they are both independent

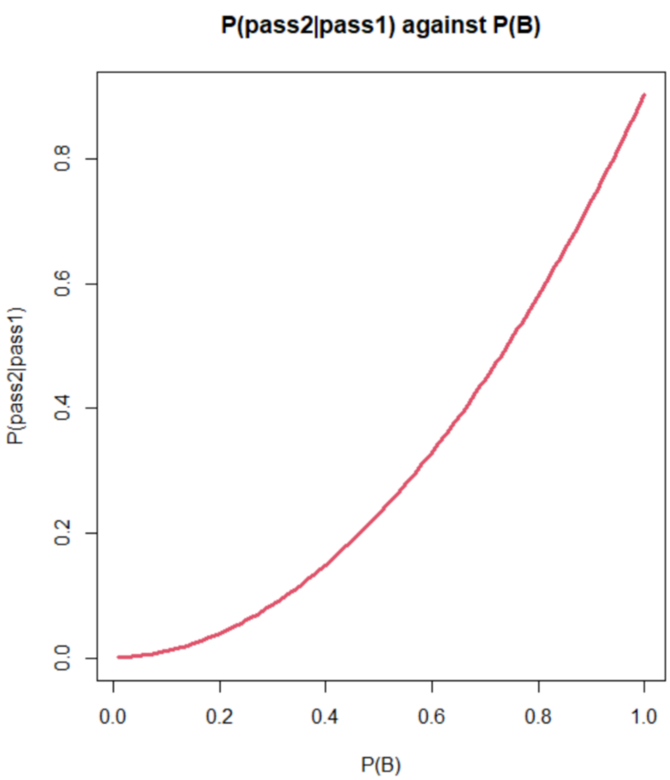
$= P(pass2) = P(A) = 0.01094$

ii) Both probabilities are the same which makes sense given that pass1 and pass2 are independent events of equal probability due to identical conditions.

iii)

```r
## d iii)
prob_p2_given_p1 = function(prob_b) {
  probability <- (0.95 * grid + 0.01 * (1 - grid))^2
  probability
}

grid <- seq(0.01, 1, by = 0.01)
y <-prob_p2_given_p1(prob_b)
plot(grid, y, type='l', xlab="P(B)", ylab="P(pass2|pass1)", col = 2, lwd=3,
     main="P(pass2|pass1) against P(B)")
```



P(pass2|pass1) seems to increase at an increasing rate as P(B)'s value increases.

Question 2

Q2a) What is the size of the sample space?

$10C3 = 120$

b) Probability that a randomly selected ~~arena~~ arena game would have a single ~~class~~ class from each tier?

size of Sample space of having a single class from each tier:

$3C1 \times 3C1 \times 4C1 = 36$

Probability $= \frac{36}{120} = 3/10 = 0.3$

ci)

```
set.seed(1234)
tier_A <- c("Rogue","Druid", "Shaman")
tier_B <- c("Warrior", "Warlock", "Hunter")
tier_C <- c("Mage", "Priest", "Paladin", "Demon Hunter")
arena <- c(tier_A, tier_B, tier_C)
sample(arena, 3)
N <- 10000
all_samples <- t(replicate(N, sample(arena, 3)))
head(all_samples)

     [,1]           [,2]           [,3]
[1,] "Demon Hunter" "Hunter"       "Warlock"
[2,] "Paladin"      "Warlock"      "Hunter"
[3,] "Warrior"      "Druid"        "Mage"
[4,] "Hunter"       "Demon Hunter" "Mage"
[5,] "Warrior"      "Priest"       "Hunter"
[6,] "Warrior"      "Demon Hunter" "Warlock"
```

cii)

Code:

```
#ii)
for (row in 1:6) {
  cat("Warlock" %in% all_samples[row,])
  print(all_samples[row,])
}
```

Output:

```
TRUE[1] "Demon Hunter" "Hunter"        "Warlock"
TRUE[1] "Paladin" "Warlock" "Hunter"
FALSE[1] "Warrior" "Druid"   "Mage"
FALSE[1] "Hunter"       "Demon Hunter" "Mage"
FALSE[1] "Warrior" "Priest"  "Hunter"
TRUE[1] "Warrior"       "Demon Hunter" "Warlock"
```

ciii)

```
#iii)
in_tier_A <- function(game) {
  return (("Shaman" %in% game) | ("Rogue" %in% game) | ("Druid" %in% game))
}

in_tier_B <- function(game) {
  return (("Hunter" %in% game) | ("Warrior" %in% game) | ("Warlock" %in% game))
}

in_tier_C <- function(game) {
  return (("Mage" %in% game) | ("Priest" %in% game) | ("Paladin" %in% game) |
            ("Demon Hunter" %in% game))
}

all_tiers <- function(game) {
  return ((in_tier_A(game)) & (in_tier_B(game)) & (in_tier_C(game)))
}
```

civ)

Code:

```
#iv)
N <- 10000
count <- 0
for (row in 1:N) {
  if (all_tiers(all_samples[row,])) {
    count <- count + 1
  }
}
proportion <- count / N
proportion
```

Output:

```
> #iv)
> N <- 10000
> count <- 0
> for (row in 1:N) {
+   if (all_tiers(all_samples[row,])) {
+     count <- count + 1
+   }
+ }
> proportion <- count / N
> proportion
[1] 0.3017
```

The probability calculated is approximately equal to the one calculated in b) so this result does agree with my own calculation.

di)

Code:

```
# Q2di
N <- 10000
all_samples <- t(replicate(N, sample(arena, 3)))
total_number_in_A <- 0
count_rogue <- 0
for (row in 1:N) {
  if ((all_samples[row,1] %in% tier_A) || (all_samples[row,2] %in% tier_A) ||
      (all_samples[row,3] %in% tier_A)) {
    if ("Rogue" %in% all_samples[row,]) {
      count_rogue <- count_rogue + 1
    }
    total_number_in_A <- total_number_in_A + 1
  }
}
probability_rogue_given_A <- count_rogue / total_number_in_A
probability_rogue_given_A
```

Output:

```
[1] 0.4284091
```

dii)

Code:

```
# Q2dii
total_number_in_B <- 0
count_rogue <- 0
for (row in 1:N) {
  if (all_samples[row,1] %in% tier_B || all_samples[row,2] %in% tier_B ||
      all_samples[row,3] %in% tier_B) {
    if ("Rogue" %in% all_samples[row,]) {
      count_rogue <- count_rogue + 1
    }
    total_number_in_B <- total_number_in_B + 1
  }
}
probability_rogue_given_B <- count_rogue / total_number_in_B
probability_rogue_given_B
```

Output:

```
[1] 0.2487709
```

diii)

The probability in di) is higher than that it dii) since P(Rogue) intersection with P(At least one from Tier A) is much higher as Rogue showing up satisfies the condition of at least one from Tier A showing up, thus the event of Rogue showing up interlaps completely with part of the event of at least 1 from Tier A showing up. Thus, by the formula of conditional probability its numerator will be much higher, making the probability higher than that in dii)

e) 3 classes are picked with replacement from a pool of 10.

i) Find the probability that a class from every tier would appear in an arena game.

Size of sample space of 3 classes picked with replacement from a pool of 10. $= 10 \times 10 \times 10 = 1000$

Size of sample space of a class from every tier appearing.

$= 3 \times 3 \times 4 \times 3!$

$$\text{Probability} = \frac{216}{1000} = 0.216$$

eii)

Code:

```
# Q2eii
set.seed(1234)
N <- 10000
samples_replace <- t(replicate(N, sample(arena, 3, replace = TRUE)))
head(samples_replace)

count_replace <- 0
for (row in 1:N) {
  if (all_tiers(samples_replace[row,])) {
    count_replace <- count_replace + 1
  }
}
proportion_with_replace <- count_replace / N
proportion_with_replace
```

Output:

```
[1] 0.2152
```

Since my answer in ei) is approximately equal to the answer computed in eii), it is correct.

Question 3

Q3. Charlie rolls a fair die 4 times.
If he rolls at least one 6, he wins a dollar from Sheen.
Otherwise, he gives Sheen \$1.
Rolls are independent.

a) Probability that Charlie wins. Is the game fair?
Probability that Charlie wins = Probability of rolling at least one six.

$P(\text{at least one six}) = 1 - P(\text{no sixes})$

$P(\text{n sixes}) = 5/6 \times 5/6 \times 5/6 \times 5/6$

$= 625/1296$

$1 - 625/1296 = 671/1296 = 0.518$

$\therefore P(\text{Charlie wins}) = 0.518$

Sheen wins otherwise, thus $P(\text{sheen wins}) = 1 - 0.518\ldots$

$= 625/1296$

$= 0.482$

$\therefore$ as $P(\text{Charlie wins}) > P(\text{sheen wins})$

The game is in Charlie's favour.
$\therefore$ It is not fair

b)

Code:

```
# Q3b
set.seed(20870815)
N <- 10000
dice_results <- 1:6
die_roll <- t(replicate(N, sample(dice_results, 4, replace = TRUE)))
wins <- 0
for (row in 1:N) {
  if (1 %in% die_roll[row,]) {
    wins <- wins + 1
  }
}
proportion_of_wins = wins / N
proportion_of_wins
```

Output:

```
[1] 0.5232
```

c) If Charlie rolls a double six within the n rolls, he wins \$1, otherwise he loses \$1.

i) Determine the probability (as a function of n) that Charlie wins the game.

$$P(\text{Charlie wins}) = P(\text{at least one double six})$$

$$= 1 - P(\text{no double six})$$

$$= 1 - (35/36)^n$$

cii)

Code

```r
# Q3cii
double_six = function(n) {
  prob_win <- 1 - (35/36)^n
  prob_win
}
```

Output:

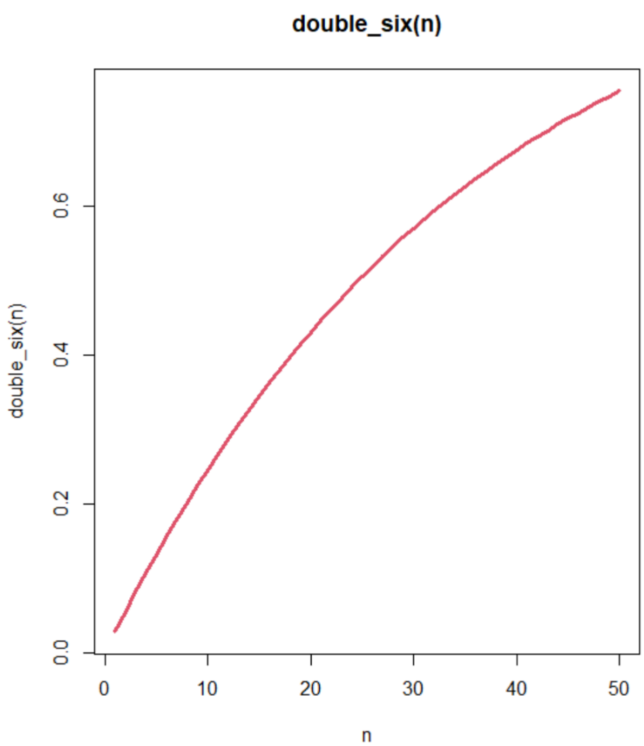
```
> double_six(10)
[1] 0.2455066
> double_six(20)
[1] 0.4307397
> double_six(30)
[1] 0.5704969
```

ciii)

Code:

```
# Q3ciii
x <- 1:50
y <- double_six(x)
plot(x, y, type='l', main="double_six(n)", col=2, lwd=3, xlab="n",
     ylab = "double_six(n)")
```

Output:



This function is monotonic in this particular range as it is continuously increasing (albeit at a decreasing rate).

civ)

Code:

```
# Q3civ
largest_int = function() {
  n <- 1
  while (double_six(n) < 0.5) {
    n <- n + 1
  }
  n <- n - 1
  n
}
```

Output:

```
> largest_int()
[1] 24
> double_six(24)
[1] 0.4914039
> double_six(25)
[1] 0.5055315
```