| |
|---|
| tle: "S&DS 220: Homework 5" |
| btitle: "Due Friday February 16" |
| thor: "Braeden Cullen" |
| tput: |
| pdf_document: default |
| lcolor: blue |

## Instructions

1. Complete the questions below. Upload your knitted PDF solutions to Gradescope by the due date.
2. Your solutions should be a combination of writing and R code. When writing, use complete sentences.
3. Previous homework assignments already had code chunks created for you. Now it is up to you to insert R code chunks within each problem as needed.
4. You should aim for clear and concise communication (in both words and R code).

## Problem set questions

### Question 1: (Exercise 3.1, 3.5, 3.23) PMFs, mean, and variance

Let $X$ be a random variable with probability mass function given by

$$p(x) = \begin{cases} 1/4, & x = 0, \\ 1/2, & x = 1, \\ 1/8, & x = 2, \\ 1/8, & x = 3 \end{cases}.$$

Answer the following (without simulation).

(a) Verify that $p$ is a valid probability mass function.

```
1/4 + 1/2 + 1/8 + 1/8
```

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

(b) Find $P(X \geq 2)$.

$P(X \geq 2)$

```
1/8 + 1/8
```

```
## [1] 0.25
```

(c) Find $P(X \geq 2 | X \geq 1)$.

$P(X \geq 2 | X \geq 1) = P(X \geq 2, X \geq 1)/P(X \geq 1) = P(X \geq 2)/P(X \geq 1) =$

```
((1/8) + (1/8)) / ((1/8) + (1/8) + (1/2))
```

```
## [1] 0.3333333
```

(d) Find $P(X \geq 2 \cup X \geq 1)$.

$P(X \geq 2 \cup X \geq 1) = P(X \geq 1)$

```
1/8 + 1/8 + 1/2
```

```
## [1] 0.75
```

(e) Find the mean of $X$.

```
# the mean of x is solved using the pmf and the x values
# the probs
probs <- c(1/4, 1/2, 1/8, 1/8)
# the x vals
x <- 0:3
# solve for mean
mean_x <- sum (x*probs)
mean_x
```

```
## [1] 1.125
```

(f) Find the variance and standard deviation of $X$.

```
# variance of x is going to be the second moment - mean squared
second_moment_x <- sum((x^2)*probs)
# variance
variance_x <- second_moment_x - mean_x^2
variance_x
```

```
## [1] 0.859375
```

```
standard_deviation_x <- sqrt(variance_x)
standard_deviation_x
```

```
## [1] 0.9270248
```

**Question 2: (Exercise 3.39) Detecting cheating in video games with the binomial distribution**

In October 2020, the YouTuber called "Dream" posted a speedrun of Minecraft and was accused of cheating. Answer the following (without simulation).

In Minecraft, when you trade with a piglin, the piglin gives you an ender pearl 4.7% of the time. Dream got 42 ender pearls after 262 trades with piglin.

Answer the following (without simulation). Recall that `pbinom` is the cdf and `dbinom` is the pmf of the binomial distribution in R.

(a) If you trade 262 times, what is the expected number of ender pearls you receive?

```r
# the expected number of ender pearls is the mean of the binomial distribution, therefore np
number_pearls <- 262
probability <- 0.047
number_pearls*probability
```

```
## [1] 12.314
```

(b) What is the probability of getting 42 or more ender pearls after 262 trades?

($P(X \geq 42)$

```r
# we can calculate the probability binomial to solve for $(P(X \geq 42)$
number_of_trades <- 262
probability <- 0.047
pbinom(41, size = number_of_trades, prob = probability, lower.tail = FALSE)
```

```
## [1] 4.627613e-12
```

When you kill a blaze, you have a 50% chance of getting a blaze rod. Dream got 211 blaze rods after killing 305 blazes.

(c) If you kill 305 blazes, what is the expected number of blaze rods you receive?

```r
# again we can solve for the expected number of blaze rods by solving for np
blazes_killed <- 305
probability <- 0.5
blazes_killed*probability
```

```
## [1] 152.5
```

(d) What is the probability of getting 211 or more blaze rods after killing 305 blazes?

($P(X \geq 211)$

```r
# we can calculate the probability binomial to solve for $(P(X \geq 211)$
# again, using probability binom exceeding 211
pbinom(210, size = blazes_killed, prob = probability, lower.tail = FALSE)
```

```
## [1] 8.791427e-12
```

(e) Do you think Dream was cheating?

The calculated figures suggest that Dream was cheating with almost absolute certainty. The probabilities of these events occouring are extremely low, and the fact that both of these events happened to the same person is even more unlikely.

**Question 3: (Exercise 3.31, with some tweaks) Simulation and Poisson approximation**

Suppose 27 people write their names down on slips of paper and put them in a hat. Each person then draws one name from the hat. Let $N$ be the number of people who draw their own name (assuming no two people have the same name).

(a) Estimate the expected value and standard deviation of $N$.

```r
num_same_name <- replicate(1e4, {
  names <- 1:27
  draws <- sample(names)
  sum(names == draws) # did they draw their name ?
})

# expected value of num_same_name
mean(num_same_name)
```

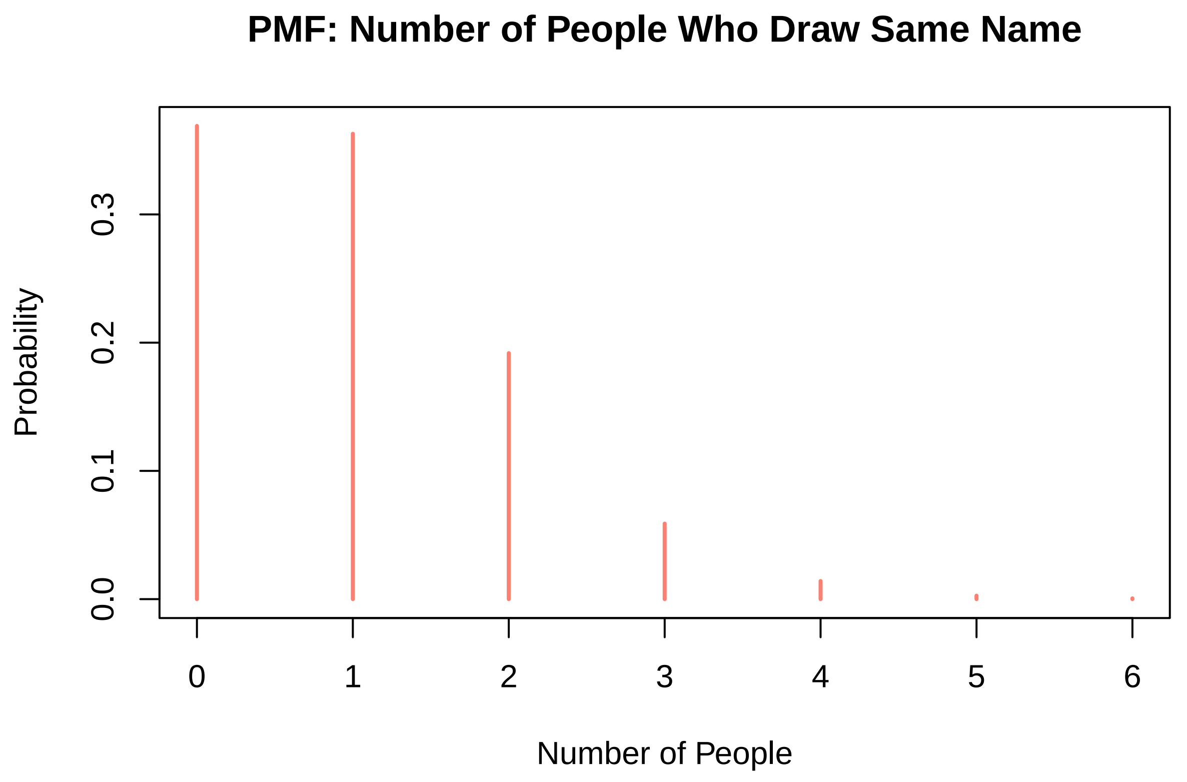
```
## [1] 0.9967
```

```r
# standard deviation
sd(num_same_name)
```

```
## [1] 0.9907509
```

(b) The pmf of $N$ can estimated from your simulation from part (a) using `proportions(table())`. Print this pmf and make a plot of the pmf using `plot()` with the argument `type = "h"`.

```r
pmf_num_same_name <- proportions(table(num_same_name))
plot(pmf_num_same_name,
  type = "h",
  xlab = "Number of People",
  ylab = "Probability",
  main = "PMF: Number of People Who Draw Same Name",
  col = "salmon")
```

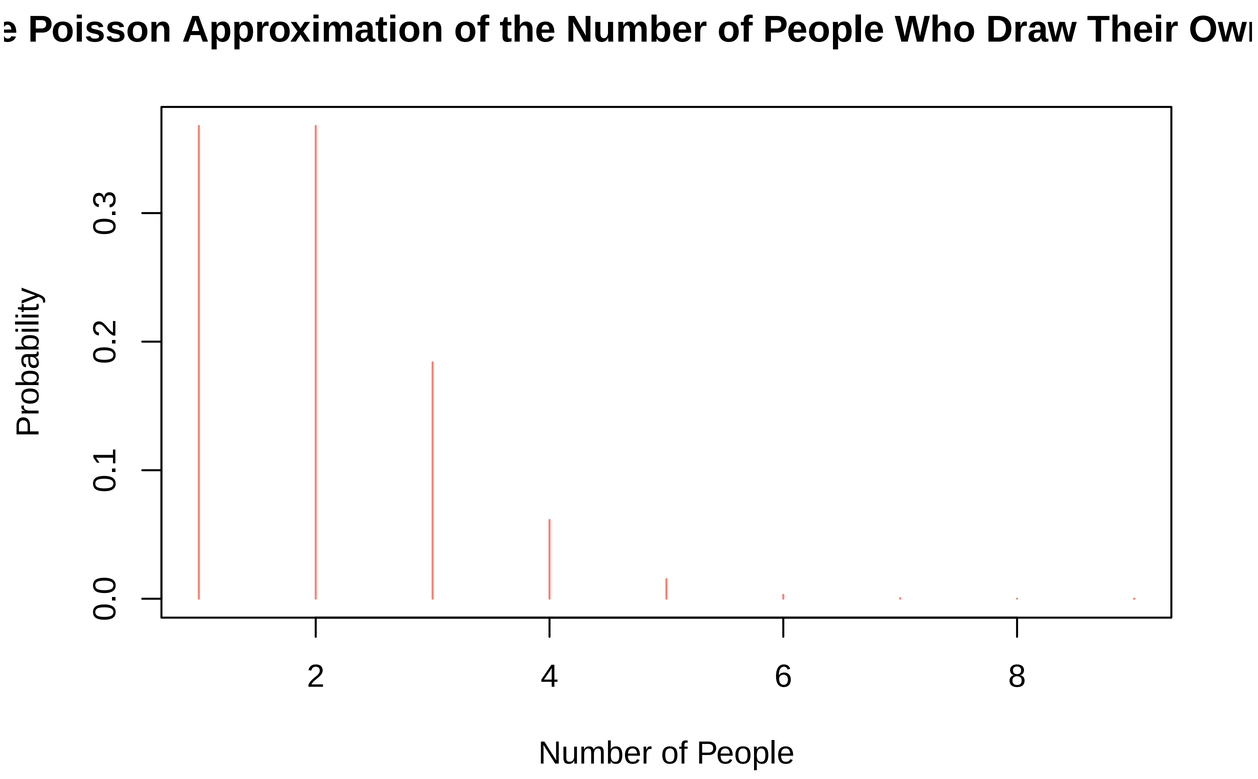


(c) We can also approximate the distribution of $N$ using a Poisson distribution. Since each person has a $1/27$ chance of drawing their name from the hat and all 27 people draw a name, the mean of $N$ is $E(N) = 27 \cdot 1/27 = 1$. And so we can approximate $N$ with a Poisson(1) distribution. Using `dpois`, give the values of the pmf of a Poisson(1) for `x = 0:8` (if the values are given in scientific notation, you can get rid of this by using `round()` with the argument `digits = 4`). Plot this pmf as in (b) using `plot()` with `type = "h"`.

```r
# poisson approximation
dpois(x = 0:8, lambda = 1) |> round(4) # poisson approximation rounded to 4 decimal places
```

```
## [1] 0.3679 0.3679 0.1839 0.0613 0.0153 0.0031 0.0005 0.0001 0.0000
```

```r
plot(dpois(x = 0:8, lambda = 1),
  type = "h",
  xlab = "Number of People",
  ylab = "Probability",
  main = "The Poisson Approximation of the Number of People Who Draw Their Own Name",
  col = "salmon")
```



(d) Compare these values with the estimated pmf from (b). Do you think the Poisson approximation of $N$ is good?

```
# pmf values from above
pmf_num_same_name
```

```
## num_same_name
##      0      1      2      3      4      5      6 
## 0.3690 0.3629 0.1918 0.0589 0.0141 0.0027 0.0006
```

```
# poisson approximation from above
dpois(x = 0:8, lambda = 1) |> round(4)
```

```
## [1] 0.3679 0.3679 0.1839 0.0613 0.0153 0.0031 0.0005 0.0001 0.0000
```

Because the outputs are so close, the Poisson approximation is doing a **great job.**

**Question 4: (Exercise 2.19) Scrabble**

In the game of Scrabble, players make words using letter tiles. The data set `fosdata::scrabble` contains all 100 tiles.

Players begin the game by drawing seven tiles from a bag of 100 tiles. Estimate the probability that a player's first seven tiles contain no vowels. (Vowels are A, E, I, O, and U.)

```r
# create vowels vector
vowels <- c("A", "E", "I", "O", "U")

# run simulation, check if the tiles pulled are in vowels vector using the %in% operator
num_vowels <- replicate(1e4, {
  tiles <- sample(scrabble$piece, size = 7, replace = FALSE) # use fosdata:scrabble
  sum(tiles %in% vowels) # %in% operator simply checks if tiles sample are in vowels
})



mean(num_vowels == 0) # probability of no vowels
```

```
## [1] 0.0176
```

**Question 5: (Exercise 3.39) More Scrabble!**

In the game of Scrabble, players make words using letter tiles, see Exercise 2.19. The tiles consist of 42 vowels and 58 non-vowels (including blanks).

*Hint:* For sampling without replacement, see the Hypergeometric distribution in Section 3.6.3.

(a) If a player draws 7 tiles (without replacement), what is the probability of getting 7 vowels?

```
dhyper(x = 7, m = 42, n = 58, k = 7)
```

```
## [1] 0.001685349
```

(b) If a player draws 7 tiles (without replacement), what is the probability of 2 or fewer vowels?

```
phyper(q = 2, m = 42, n = 58, k = 7)
```

```
## [1] 0.3714395
```

(c) What is the expected number of vowels drawn when drawing 7 tiles?

```
k <- 7

n <- 58

m <- 42

k * m / (m + n)
```

```
## [1] 2.94
```

(d) What is the standard deviation of the number of vowels drawn when drawing 7 tiles?

The standard deviation of the number of vowels drawn when drawing 7 tiles is given by the formula:

$$\sqrt{\frac{k \cdot m}{m+n} \cdot \frac{n}{m+n} \cdot \frac{m+n-k}{m+n-1}}$$

```
sqrt(k * m / (m + n) * n / (m + n) * (m + n - k)/(m + n - 1))
```

```
## [1] 1.265644
```

**Question 6: (Exercise 3.40 with some tweaks) Simulating deathrolling in World of Warcraft**

Deathrolling in World of Warcraft works as follows. Player 1 tosses a 1000-sided die. Say they get $x_1$. Then player 2 tosses a die with $x_1$ sides on it. Say they get $x_2$. Player 1 tosses a die with $x_2$ sides on it. The player who loses is the player who first rolls a 1.

> *Coding hint:* a `while` may be helpful in this exercise.

(a) Estimate the expected total number of rolls before a player loses.

```r
rolls <- replicate(1e4, {
  die_1000 <- 1000
  roll <- 0
  # rule for deathroll: keep rolling until you get a 1, then if you get a 1, you lose
  while(die_1000 > 1)
    {
    roll <- roll + 1
    die_1000 <- sample(1:die_1000, size = 1)
  }
  roll
})

mean(rolls)
```

```
## [1] 8.4784
```

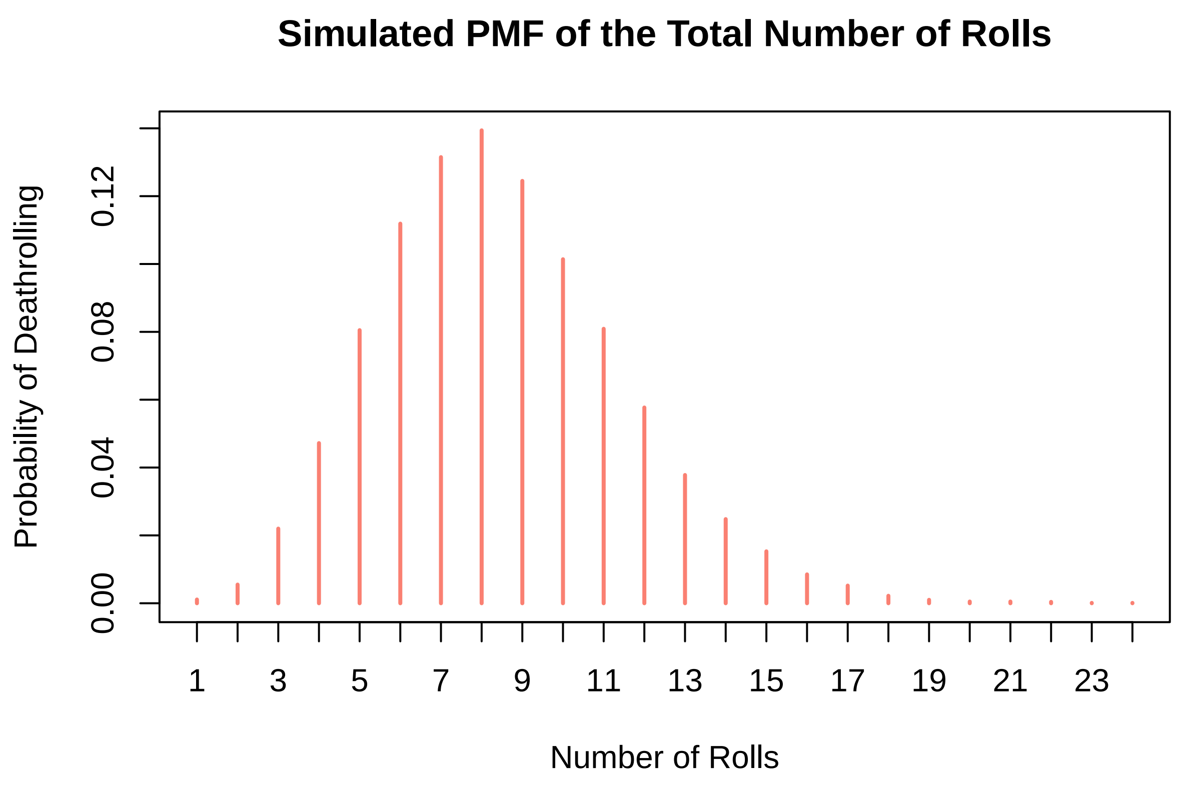
(b) Estimate the probability mass function of the total number of rolls. You can use `proportions(table( ))` to give your answer.

```r
pmf_estimation_rolls <- proportions(table(rolls))
pmf_estimation_rolls
```

```
## rolls
##      1      2      3      4      5      6      7      8      9     10     11
## 0.0011 0.0055 0.0220 0.0472 0.0805 0.1119 0.1315 0.1394 0.1245 0.1014 0.0809
##     12     13     14     15     16     17     18     19     20     21     22
## 0.0577 0.0378 0.0248 0.0153 0.0085 0.0052 0.0022 0.0010 0.0005 0.0005 0.0004
##     23     24
## 0.0001 0.0001
```

(c) Plot the estimated probability mass function from (b), using `plot()` with the argument `type = "h"`.

```r
plot(pmf_estimation_rolls,
     type = "h",
     xlab = "Number of Rolls",
     ylab = "Probability of Deathrolling",
     main = "Simulated PMF of the Total Number of Rolls",
     col = "salmon")
```



(d) Estimate the probability that player 1 wins.

```
mean((rolls %% 2) == 0) # the prob that 1 came up on an even roll, leading to a death roll and a win fo
```

```
## [1] 0.4996
```