Activity 3 - Salvin Chowdhury

Due Date: Febraury 26, 2025

Problem 1: Creating a PMF, CDF, and Calculating Expected Value and Standard Deviation (Discrete Random Variable).

Suppose we randomly ask 250 students about the number of pets they have and find the following:

- 46 have no pets
- 107 have one pet
- 58 have two pet
- 30 have three pets
- 9 have four pets

Consider the random variable, X, that is the number of pets a student has.

Part (a)

Create a PMF "table" using the data above by doing the following:

- Create a vector for the possible values of X and call it "X"
- Create a vector for the probabilities of each of these values and call it "p"
- Bind them into a single object using the rbind() function

```
# setting up a vector for possivle values of X
X <- c(0, 1, 2, 3, 4)

# displaying the vector for possible values of X
X
```

```
## [1] 0 1 2 3 4
```

```
# finding the total number of pets
total_pets = 46 + 107 + 58 + 30 + 9
print(paste("Total Number of Pets: ", total_pets))
```

```
## [1] "Total Number of Pets:  250"
```

```
# calculating all the probabilities
no_pets_probability = 46 / total_pets
one_pet_probability = 107 / total_pets
two_pet_probability = 58 / total_pets
three_pet_proability = 30 / total_pets
four_pet_probability = 9 / total_pets

# setting up a vector for probabilities
p <- c(no_pets_probability, one_pet_probability, two_pet_probability, three_pet_proability,
four_pet_probability)

# displaying the vector for probabilities
p
```

```
## [1] 0.184 0.428 0.232 0.120 0.036
```

```r
# binding both vectors into a single object
PMFPets = rbind(X, p)
print(PMFPets)
```

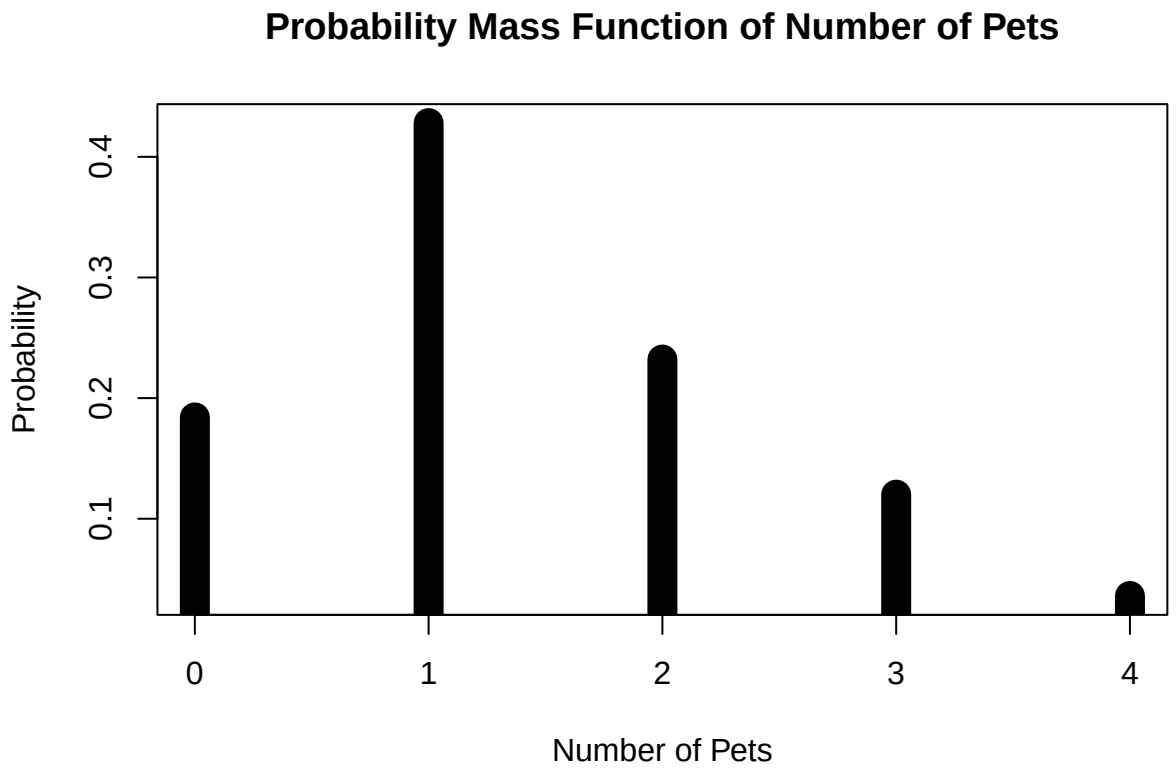
```
##    [,1]  [,2]  [,3] [,4]  [,5]
## X 0.000 1.000 2.000 3.00 4.000
## p 0.184 0.428 0.232 0.12 0.036
```

Part (b)

Choose one (or both) of the following to complete:

- Create a histogram-like line graph for the PMF using the plot() function. Be sure to specify each of the following arguments to get the correct plot: x, y, type. Your graph should also have an appropriate title and axes labels.

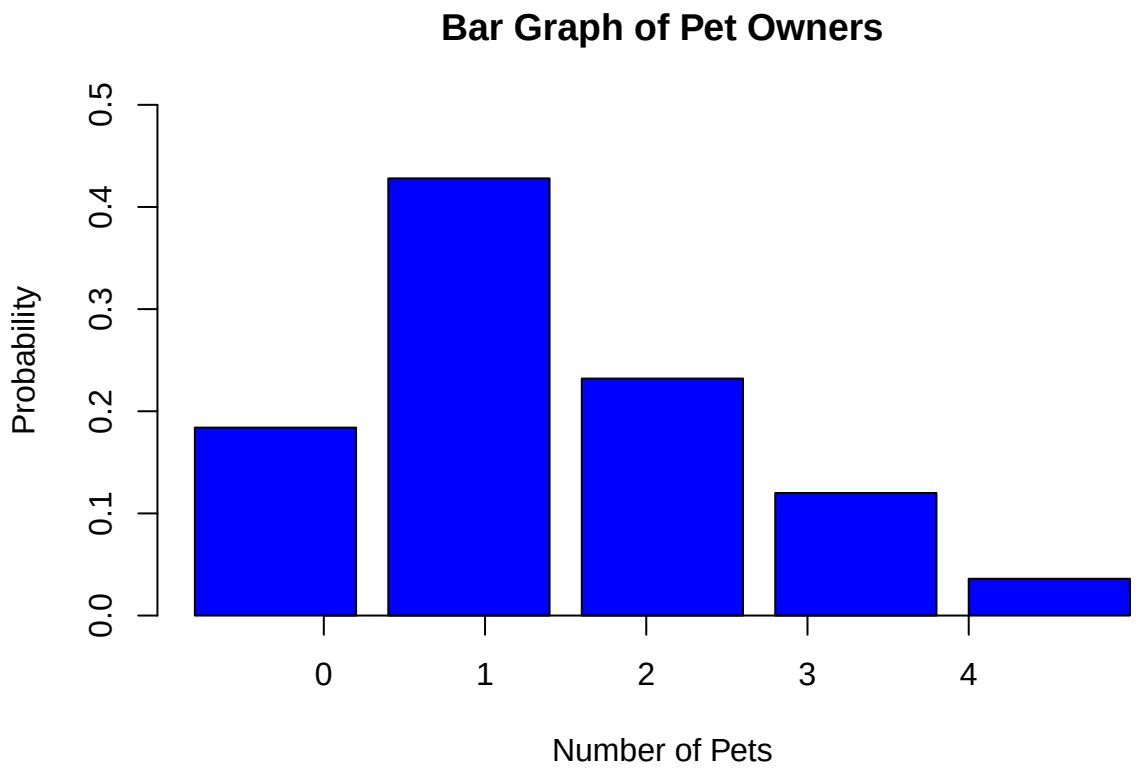
```r
# # saving the plot
# png("C:/GitHub/StatisticsInR/wk_04/plots/PlotPets.png", width=800, height=600)

# the plottng of PMF in terms of numbers of pets vs probability
plot(X, p, type='h', main="Probability Mass Function of Number of Pets",
xlab="Number of Pets",
ylab="Probability", lwd=15)
```




```r
# # saving the plot
# dev.off()
```

- Create a bar graph for the PMF using the barplot() function. To make sure the correct values display on the bottom, you can specify the names.arg
- argument or assign the appropriate vector to names(p) before plotting. You graph should have an appropriate title and axes labels.

```
# # saving the plot
# png("C:/GitHub/StatisticsInR/wk_04/plots/BarPlot.png", width=800, height=600)

# the plotting of a bar plot
barplot(p, col="blue", main="Bar Graph of Pet Owners", xlab="Number of Pets",
ylab="Probability", ylim=c(0, 0.5))

# building a x-axis
axis(1, at = 1:length(X), labels = X)
```




```
# # saving the plot
# dev.off()
```

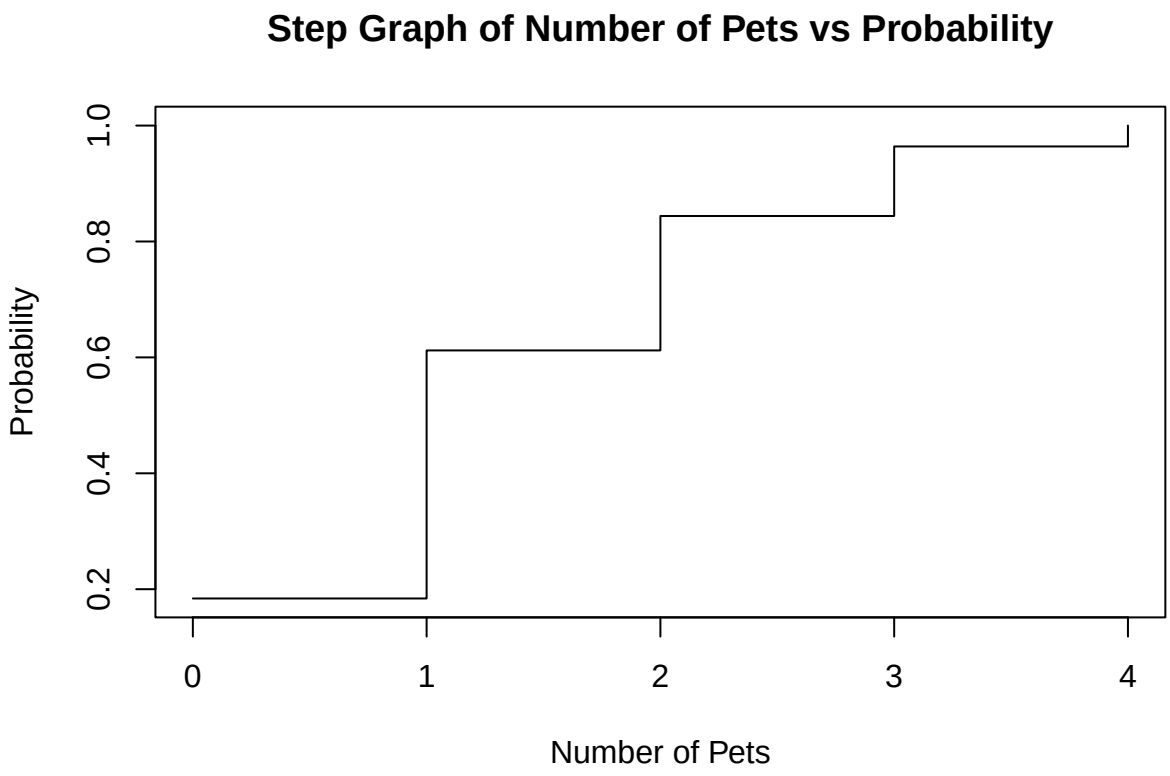
Part (c)

Create a step graph for the CDF using the plot() function. Be sure to specify the x, y, and type arguments. Note, this won't be perfect, but it's about as good as we can do without getting "too fancy" with code. Your graph should have an appropriate title and axes labels.

Hint: the cumulative sum function cumsum() may be helpful here!

```
# # saving the plot
# png("C:/GitHub/StatisticsInR/wk_04/plots/StepPlot.png", width=800, height=600)
```

```r
# the cumulative sum function
sum = cumsum(p)

# the plotting of a step plot
plot(X, sum, type="s", main="Step Graph of Number of Pets vs Probability",
xlab="Number of Pets",
ylab="Probability")
```




```r
# # saving the plot
# dev.off()
```

Part (d)

Find the expected value and standard deviation of X.

Hint: it may be best to just use R as a calculator here, to make sure you get the correct results.

```r
# viewing the probability table
print(PMFPets)
```

```
##    [,1]  [,2]  [,3] [,4]  [,5]
## X 0.000 1.000 2.000 3.00 4.000
## p 0.184 0.428 0.232 0.12 0.036
```

```r
# finding the expected value sing sum()
expected_value_sum = sum(PMFPets[1,] * PMFPets[2, ])
print(paste("Expected Value using sum(): ", expected_value_sum))
```

```
## [1] "Expected Value using sum():  1.396"
```

```
# finding the expected value
expected_value <- (PMFPets[1, 1] * PMFPets[2, 1]) + (PMFPets[1, 2] * PMFPets[2, 2]) +
(PMFPets[1, 3] * PMFPets[2, 3]) + (PMFPets[1, 4] * PMFPets[2, 4]) +
(PMFPets[1, 5] * PMFPets[2, 5])
print(paste("Expected Value Manual: ", expected_value))
```

```
## [1] "Expected Value Manual:  1.396"
```

```
# finding the sum of squares
sum_of_squares <- (PMFPets[1, 1]^2 * PMFPets[2, 1]) + (PMFPets[1, 2]^2 * PMFPets[2, 2]) +
(PMFPets[1, 3]^2 * PMFPets[2, 3]) + (PMFPets[1, 4]^2 * PMFPets[2, 4])
+ (PMFPets[1, 5]^2 * PMFPets[2, 5])
```

```
##     X
## 0.576
```

```
print(paste("Sum of Squares: ", sum_of_squares))
```

```
## [1] "Sum of Squares:  2.436"
```

```
# calculating the standard deviation and variance
variance <- sum_of_squares - (expected_value^2)
standard_deviation <- sqrt(variance)
print(paste("Standard Deviation Manual: ", standard_deviation))
```

```
## [1] "Standard Deviation Manual:  0.697985673205404"
```

Problem 2: Coin Flip Simulation

Part (a)

Suppose we flip a fair coin 20 times, and denote by X the random variable which is the number of heads. List all possible outcomes of X.

Hint: "m:n" outputs a vector of the integers between a and b

```
X <- 0:20
print(X)
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

Part (b)

Use the the rbinom() function to simulate flipping a fair coin 20 times and recording the number of heads. To avoid this result changing if you run the code chunk more than once, first, pick your favorite (whole) number and use the set.seed() function with that number as the input.

```
# setting the seed for reproducibility
set.seed(20)

# calculating the binomial distribution
# parameters: simulations, size, probability
coin_flip <- rbinom(1, 20, 0.5)
print(coin_flip)
```

```
## [1] 13
```

Part(c)

Use the dbinom() function to find the probability of obtaining the exact number of heads you got in part (b).

```
# the binomial probability mass function
# note: I got 13 heads in the previous cell
probability_d <- dbinom(13, 20, 0.5)
print(paste("Probability of getting exactly13 heads: ", probability_d))
```

```
## [1] "Probability of getting exactly13 heads:  0.0739288330078124"
```

Part (d)

Use the pbinom() function to find the probability of obtaining fewer heads than you got in part (b).

```
# the binomial cumulative distribution function
probability_p <- pbinom(12, 20, 0.5)
print(paste("Probability of getting fewer than 13 heads: ", probability_p))
```

```
## [1] "Probability of getting fewer than 13 heads:  0.868412017822266"
```

Part (e)

Find the probability of obtaining more heads than you got in part (b).

```
# probability of more than 13 heads
probability_thirteen <- pbinom(13, 20, 0.5)
probability_greater_than_thirteen <- 1 - probability_thirteen
print(paste("Probability of obtaining more heads: ", probability_greater_than_thirteen))
```

```
## [1] "Probability of obtaining more heads:  0.0576591491699219"
```

Part (f)

Now use the rbinom() function to simulate the process repeating part (b) 100 times. Use the set.seed() function that you used earlier here just to make sure things don't change when running this code multiple times, and I would recommend assigning this simulation a name, such as "sim1."

Note: For clarity, you are simulating the experiment of 20 coin flips 100 times, not flipping 100 coins.

```
# setting the seed for reproducibility
set.seed(20)

# running the first simulation
simulation_one <- rbinom(100, 20, 0.5)
print(simulation_one)
```

```
##   [1] 13 12  9 10 14 15  7  7  9  9 11 12  4 11  8 10  9  7  9 12 10  6 10  7  9
##  [26]  7 13 15  7 11  9 10 12  8 10 10 10 13 11  9 11  6 10 10 12 10 11 12  9 11
##  [51]  7 10  9 14  7 10  5 10  9  6 10  7 14  6 13  6 12 11  8 10  6  9  8 10 13
##  [76]  9 10 10 11  9  8 13  7 13 11 10  9 10  8  6 10  9 12 14 12 13 12 12  9  6
```

Part (g)

Use the table() function to create a frequency table for the result from part (f), then use this to create a PMF table.

```
# the frequency table
frequency_table <- table(simulation_one)
print(frequency_table)
```

```
## simulation_one
##  4  5  6  7  8  9 10 11 12 13 14 15
##  1  1  8 10  6 17 22 10 11  8  4  2
```

```
# the probability mass function table
pmf_table <- frequency_table / sum(frequency_table)
print(pmf_table)
```

```
## simulation_one
##    4    5    6    7    8    9   10   11   12   13   14   15
## 0.01 0.01 0.08 0.10 0.06 0.17 0.22 0.10 0.11 0.08 0.04 0.02
```

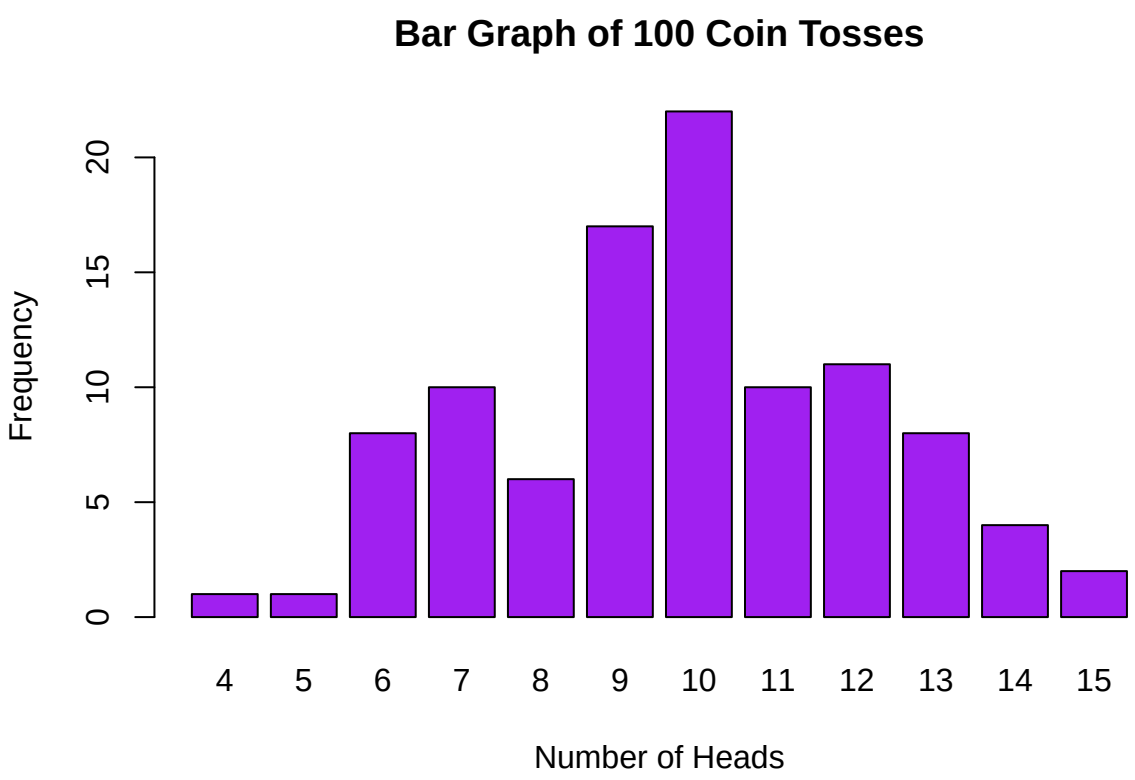
```
# checking if all the probabilities add up to 1
sum_of_probability = sum(pmf_table[])
print(paste("Sum of Probability: ", sum_of_probability))
```

```
## [1] "Sum of Probability:  1"
```

Part (h)

Create a plot (a line graph or bar graph, depending on your preference) based on the table in part (g). Your plot should have an appropriate title and axes labels.

```
# saving the plot
# png("C:/GitHub/StatisticsInR/wk_04/plots/BarPlot.png", width=800, height=600)

# the first row of the pmf table
outcomes <- as.numeric(names(pmf_table))

# making a histogram plot
barplot(frequency_table, col="purple", main="Bar Graph of 100 Coin Tosses",
xlab="Number of Heads", ylab="Frequency")
```




```
# plot(outcomes, pmf_table, type="h",
# main="Histogram of Probability Mass Function of 100 Coin Tosses", xlab="Number of Heads",
# ylab="Probability", lwd=5)

# saving the plot
# dev.off()
```

Part (i)

Create a plot which is based on running this simulation 1000 times instead of 100 times. Do the same for 10000 times. Again, make sure to use the set.seed() function as before to ensure your results don't change if you run the code more than once. These plots should have appropriate title and axes labels.

```
# setting the seed for reproducibility
set.seed(20)

# running the second simulation
simulation_two <- rbinom(1000, 20, 0.5)
print(simulation_one)
```

```
##   [1] 13 12  9 10 14 15  7  7  9  9 11 12  4 11  8 10  9  7  9 12 10  6 10  7  9
##  [26]  7 13 15  7 11  9 10 12  8 10 10 10 13 11  9 11  6 10 10 12 10 11 12  9 11
##  [51]  7 10  9 14  7 10  5 10  9  6 10  7 14  6 13  6 12 11  8 10  6  9  8 10 13
##  [76]  9 10 10 11  9  8 13  7 13 11 10  9 10  8  6 10  9 12 14 12 13 12 12  9  6
```

```
# the frequency table
frequency_table_two <- table(simulation_two)
print(frequency_table_two)
```

```
## simulation_two
##   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  18
##   1   1   7   7  38  76 126 147 182 153 121  76  36  20   8   1
```

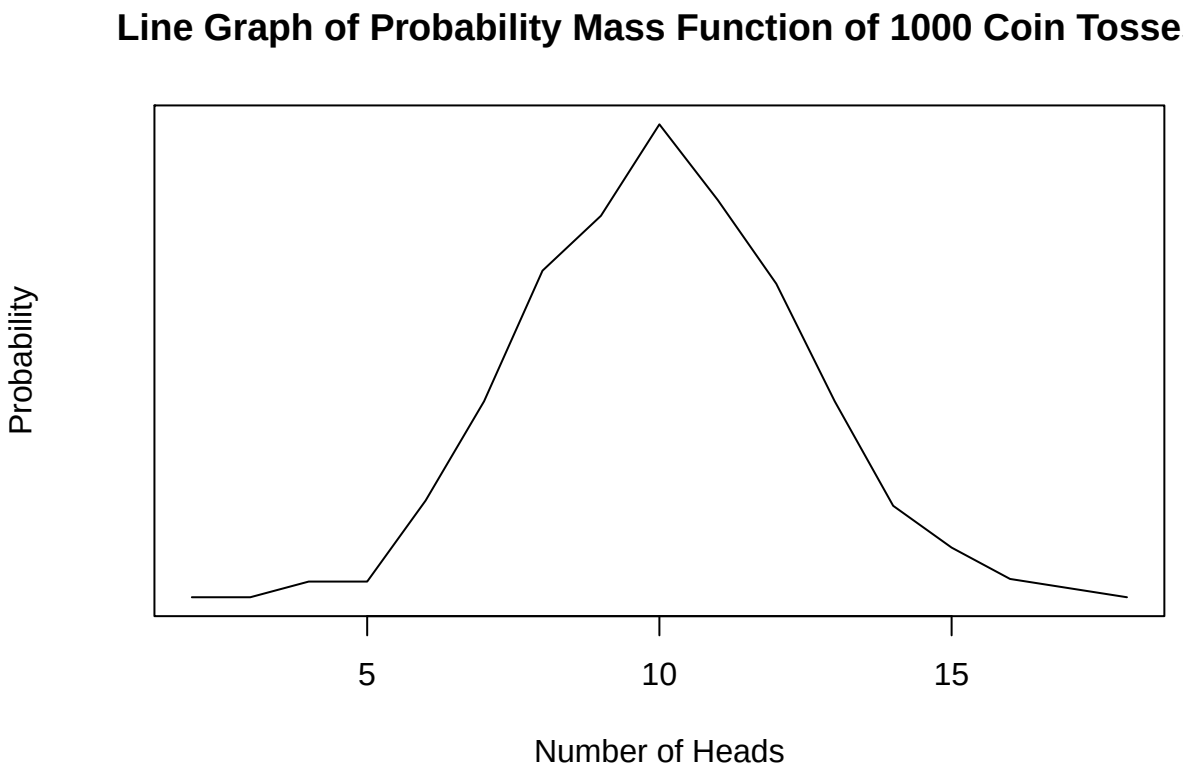
```
# the probability mass function table
pmf_table_two <- frequency_table_two / sum(frequency_table_two)
print(pmf_table_two)
```

```
## simulation_two
##     2     3     4     5     6     7     8     9    10    11    12    13    14
## 0.001 0.001 0.007 0.007 0.038 0.076 0.126 0.147 0.182 0.153 0.121 0.076 0.036
##    15    16    18
## 0.020 0.008 0.001
```

```
# checking if all the probabilities add up to 1
sum_of_probability_two = sum(pmf_table_two[])
print(paste("Sum of Probability: ", sum_of_probability_two))
```

```
## [1] "Sum of Probability:  1"
```

```
# # saving the plot
# png("C:/GitHub/StatisticsInR/wk_04/plots/LinePlot1000.png", width=800, height=600)

# the first row of the pmf table
outcomes_two <- as.numeric(names(pmf_table_two))

# making a line plot
plot(outcomes_two, pmf_table_two, type="l",
main="Line Graph of Probability Mass Function of 1000 Coin Tosses",
xlab="Number of Heads", ylab="Probability")
```




```
# # saving the plot
# dev.off()
```

What do you notice? (Type your answer below)

Answer: We get a nornmal distribution shaped curve.

Problem 3: Poisson Distribution

The probability distribution over the number of earthquakes in a year in a particular area is usually modeled using a Poisson distribution. This is because, although earthquakes occur randomly over a long period, the number of earthquakes over time approaches a constant rate. Suppose, on average, there are 7 earthquakes per month in a certain region.

Part (a)

Use the dpois() function to find the probability that there are 6 earthquakes in the next month.

```
# since average is 7, mean is 7
# we are using the poisson probability mass function
# parameters: x, average
probability_of_six <- dpois(6, 7)
print(paste("Probability of 6 Earthquakes: ", probability_of_six))
```

```
## [1] "Probability of 6 Earthquakes:  0.149002779674338"
```

Part (b)

Use the ppois() function to find the probability that there are 4 or fewer earthquakes in the next month. Confirm this result by summing up results of the dpois() function.

```
# probability that there are 4 or fewer earthquakes
probability_four_or_fewer <- ppois(4, 7)

# cross-checking using dpois()
probability_using_dpois <- dpois(0, 7) + dpois(1, 7) + dpois(2, 7) + dpois(3, 7) +
dpois(4, 7)

print(paste("4 or fewer using ppois(): ", probability_four_or_fewer))
```

```
## [1] "4 or fewer using ppois():  0.172991607882071"
```

```
print(paste("4 or fewer using dpois(): ", probability_using_dpois))
```

```
## [1] "4 or fewer using dpois():  0.172991607882071"
```

Part (c)

Find the probability that there are 9 or more earthquakes in the next month.

```
# 9 or more earthquakes, so less than or equal to 8
probability_eight_or_fewer <- ppois(8, 7)
probability_nine_or_more <- 1 - probability_eight_or_fewer
print(paste("Probability of Nine or More Earthquakes: ", probability_nine_or_more))
```

```
## [1] "Probability of Nine or More Earthquakes:  0.270908732261918"
```

Part (d)

Simulate the number of earthquakes per month over the next 10 **years** using the rpois() function. Be sure to use the set.seed() function as you did earlier and call this something like "sim2" so it is easy to use the exact result you got in the next part.

```
# setting the seed for reproducibility
set.seed(20)

# 12 months per year, so 120 months in 10 years
simulation_three = rpois(120, 7)
```

Part (e)

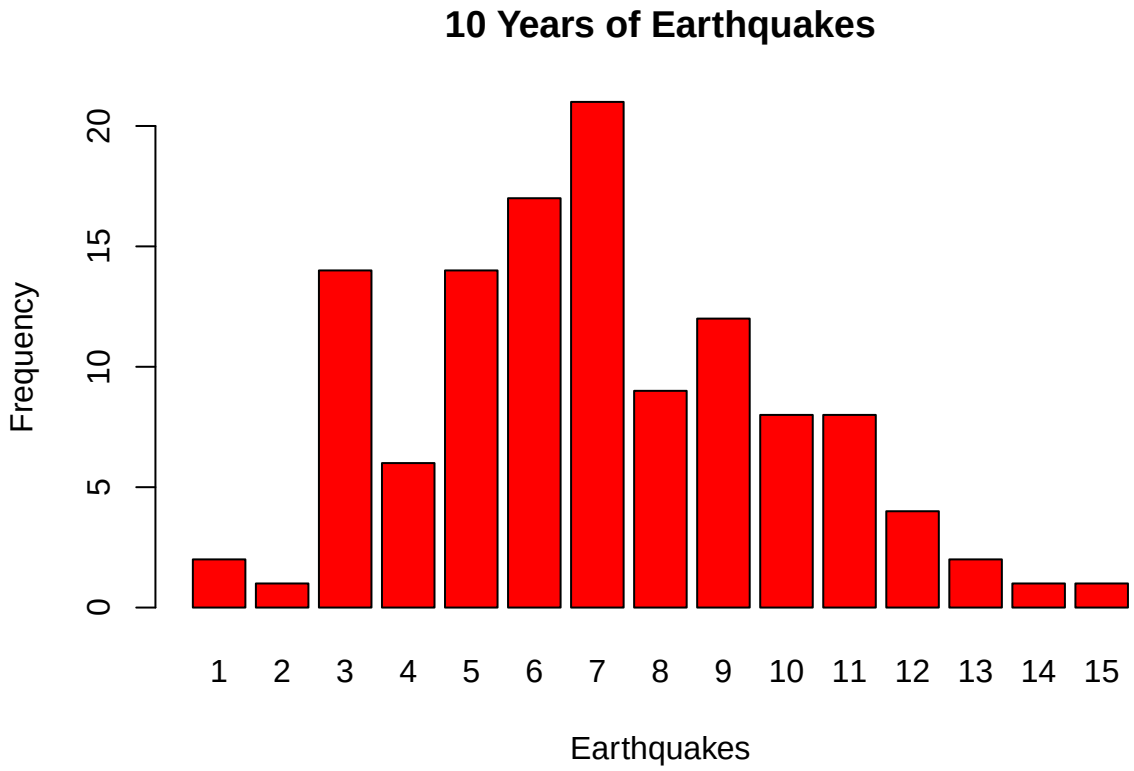
Create a PMF table for your result from part (d) and use it to create a plot. Your plot should have an appropriate title and axes labels.

```
# the frequency table
frequency_table_three  <- table(simulation_three)
print(frequency_table_three)
```

```
## simulation_three
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
##  2  1 14  6 14 17 21  9 12  8  8  4  2  1  1
```

```
# # saving the plot
# png("C:/GitHub/StatisticsInR/wk_04/plots/10YearsOfEarthquakes.png", width=800,
# height=600)
```

```r
# making a line plot
# creating a bar plot
barplot(frequency_table_three, col="red",
        main = "10 Years of Earthquakes",
        xlab = "Earthquakes", ylab = "Frequency")
```




```r
# # saving the plot
# dev.off()
```

Part (f)

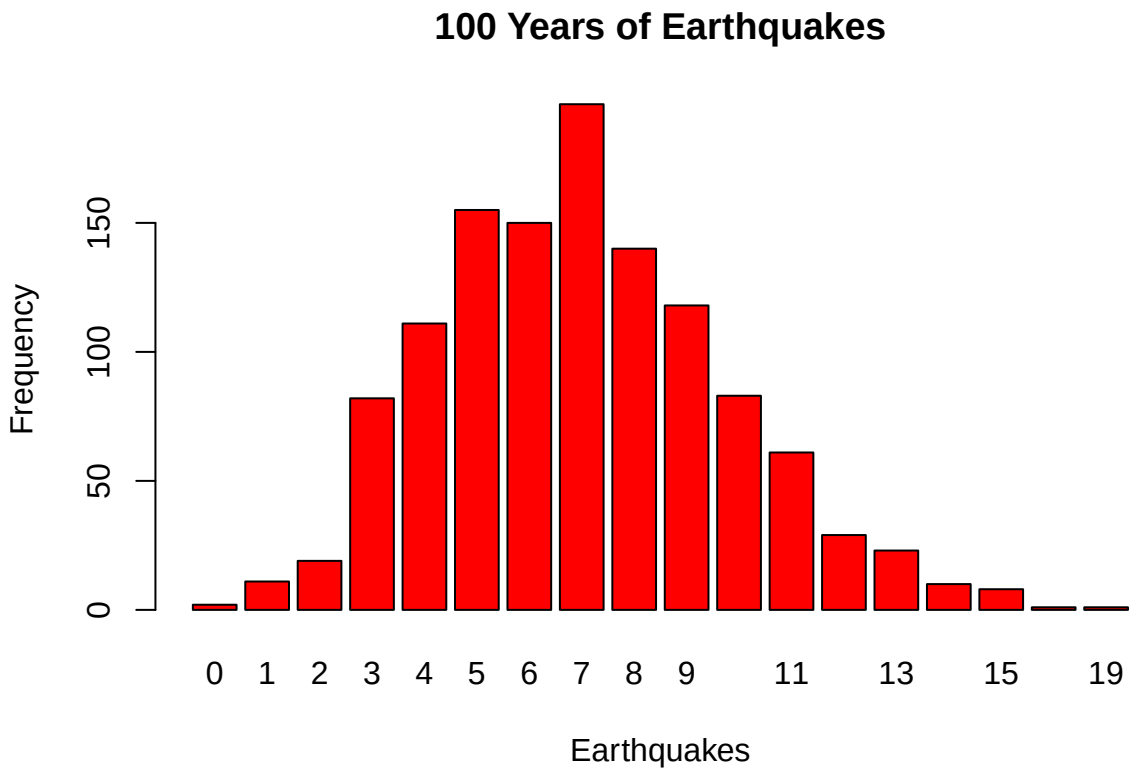
Repeat the simulation from part (d) but over the next 100 years instead. Be sure to use the set.set() function as before and call it something like "sim3." Then use this to create a plot similar to that of part (e). Your plot should have an appropriate title and axes labels.

```r
# setting the seed for reproducibility
set.seed(20)

# 12 months per year, so 1200 months in 100 years
simulation_four = rpois(1200, 7)

# the frequency table
frequency_table_four  <- table(simulation_four)
print(frequency_table_four)
```

```
## simulation_four
```

```
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  19
##   2  11  19  82 111 155 150 196 140 118  83  61  29  23  10   8   1   1
```

```
# # saving the plot
# png("C:/GitHub/StatisticsInR/wk_04/plots/100YearsOfEarthquakes.png", width=800,
# height=600)

# making a line plot
# creating a bar plot
barplot(frequency_table_four, col="red",
        main = "100 Years of Earthquakes",
        xlab = "Earthquakes", ylab = "Frequency")
```




```
# # saving the plot
# dev.off()
```

Part (g)

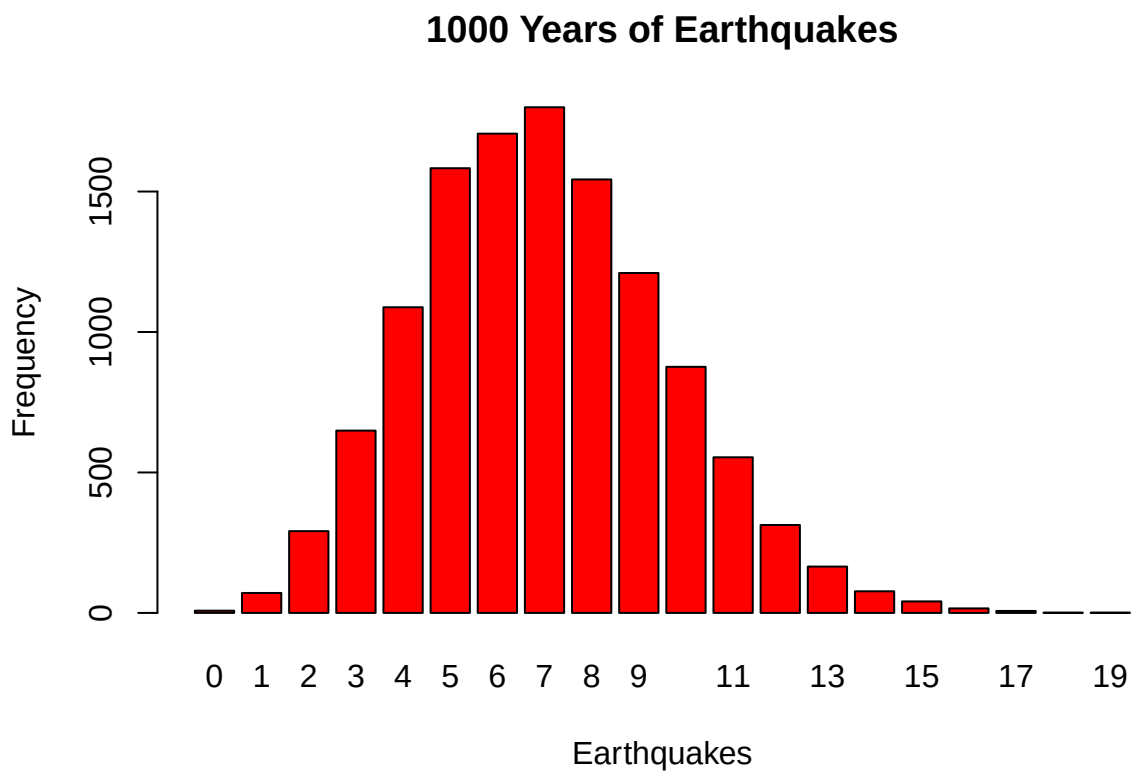
Repeat the simulation once more but over the next 1000 years instead. Again, be sure to use the set.seed() function and give the simulation a name like "sim4." Then, use this to create a plot as before. Your plot should have an appropriate title and axes labels.

```
# setting the seed for reproducibility
set.seed(20)

# 12 months per year, so 12000 months in 1000 years
simulation_five = rpois(12000, 7)
```

```r
# the frequency table
frequency_table_five  <- table(simulation_five)
print(frequency_table_five)
```

```
## simulation_five
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
##    8   71  291  649 1088 1583 1706 1800 1543 1210  876  554  313  165   77   41
##   16   17   18   19
##   16    7    1    1
```

```r
# # saving the plot
# png("C:/GitHub/StatisticsInR/wk_04/plots/1000YearsOfEarthquakes.png", width=800,
# height=600)

# making a line plot
# creating a bar plot
barplot(frequency_table_five, col="red",
        main = "1000 Years of Earthquakes",
        xlab = "Earthquakes", ylab = "Frequency")
```




```r
# # saving the plot
# dev.off()
```

Part (h)

What do you notice about these simulations/plots? Type your answer below.

Answer: As the number of years increases, the shape of the bar plots are of a normal distribution and is uni-modal, however, they become increasingly right skewed.