MA 354: Data Analysis – Fall 2021 – Due 10/8 at 5p
Homework 2: Giancarlo Arcese, Aziz Zafar, Tedi Totojani, Div Chamria

Complete the following opportunities to use what we've talked about in class. These questions will be graded for correctness, communication and succinctness. Ensure you show your work and explain your logic in a legible and refined submission.

The starting jobs will be applied in alphabetical order (last name) for question two.

1. **Solver:** provide a solution, if possible, and reasoning for the solution. **Due to group 10/5 or earlier.**
2. **Code Checker:** provides a first check of the solver's worked solutions and ensures they are correct with a solid interpretation.
3. **Checker** checks the solution for completeness, proposes and implements changes if agreed upon by the group. Provides a short paragraph summarizing the discussion of proposals and their reason for acceptance or non-acceptance.
4. **Double Checker** checks the solution for completeness, communication and polish. The Double Checker ensures that the solution is correct and highly polished for submission.

For subsequent questions student roles will move down one position. The rolls change as follows.

1. **Solver** $\Longrightarrow$ **Code Checker**
2. **Code Checker** $\Longrightarrow$ **Checker**
3. **Checker** $\Longrightarrow$ **Double Checker**
4. **Double Checker** $\Longrightarrow$ **Solver**

While students have assigned jobs for each question I encourage students to help each other complete the homework in collaboration.

1. Select a continuous distribution (Not the uniform or exponential). It does not have to be one that we cover in the notes! To explore the PDF of your distribution, specify two sets of parameter(s) for your distribution.

   (a) **History** Discuss what types of random variables are modeled with your distribution. Be sure to include a discussion about the support and ensure to provide the density function, and CDF. This requires some internet research – what's the history of the distribution, why was it created and named? What are some exciting applications of this distribution?

   Cite all of your sources in LaTeX by adding a BibTeX citation to the .bib file. To help, I've cited R (R Core Team, 2021) in parentheses here. R Core Team (2021) provides helpful tools for the rest of the questions below. BibTeX citations are available through Google Scholar by clicking the cite button below the article of interest and selecting the BibTeX option.

   Solution: **The Gamma distribution models the probability that α events occur in a Poisson process with mean arrival time $\beta = \frac{1}{\theta}$, θ a scale parameter. (Analytica Wiki, 2021). The probability density function, which is a function of the random variable x, and our parameters α and β is given in the equation below: (provided by Gamma Function Wikipedia (2021))**

$$f(x;\alpha,\beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)} \quad \text{for } x > 0 \quad \alpha, \beta > 0, \tag{1}$$

Where $\Gamma(\alpha)$ is the Gamma function. The restrictions on the parameters and random variable comes from the need for the PDF to have positive values only. x, $\beta > 0$ ensures the numerator is positive, and $\alpha > 0$ ensures the denominator is positive, since $\Gamma(x) > 0 \; \forall \; x \in \mathbb{R}^{+}$. $x > 0$ is the support of the distribution - The random variable must be a positive real number. Note some probability distribution function definitions include a location parameter ϵ, which is subtracted from the random variable x and essentially moves the distribution left and right for various values. I have omitted it in the equation above since most sources keep the distribution bounded by 0. The cumulative distribution function is given below: (provided by Gamma Function Wikipedia (2021))

$$F(x;\alpha,\beta) = \int_0^x f(u;\alpha,\beta)\, du = \frac{\gamma(\alpha,\beta x)}{\Gamma(\alpha)}, \tag{2}$$

where $\gamma()$ is the lower incomplete Gamma function, given by: (provided by Incomplete gamma function Wikipedia (2021))

$$\gamma(s,x) = \int_0^x t^{s-1}\, e^{-t}\, \mathrm{d}t. \tag{3}$$

The Gamma distribution is named after the Gamma function, which is principle in its derivation. The Gamma function is a function that extends the idea of the factorial to all real numbers and even complex numbers, provided that any real input is not a non-negative integer. For positive real inputs, (which is the support for the Gamma distribution), the Gamma function draws a continuous line through all points in the (x, y) plane whose x value is an integer and y value equals $(x+1)!$.

The Gamma function is good at modeling any continuous random variable that takes on positive real values, is uni-modal, and is positively skewed. The Gamma distribution is frequently used in climatology to model random weather variables such as

rainfall (Thom, Herbert CS, 1958). It has also been used to model the size of insurance claims (Boland, Philip J, 2007), multi-path fading of of signal power in wireless communication (Gamma Function Wikipedia, 2021), and the age distribution of cancer incidence (Belikov, Aleksey V, 2017), to name just a few of its applications.

(b) Show that you have a valid PDF. You will find the `integrate()` function in `R` helpful.

Solution: **A valid PDF must not take on negative values and must integrate over its support to one. I explained in part A that it is impossible for the Gamma PDF to take on a negative value provided that our random variable x and our shape and rate parameters α and β are all greater than zero. With this established, we can test in R whether the PDF integrated from 0 to ∞ (our support) equals one.**

```
options(scipen=999)                                  #Remove scientific notation
gammaPDF <- function(x, a, b) {                      #define PDF function ourself
  (1/(gamma(a)))*(b^a)*(x^(a-1))*(exp(1)^(-b*x))     #a is shape, b is rate.
}

gammaPDF(1, 1, 0.5)

## [1] 0.3032653

dgamma(1, 1, 0.5)

## [1] 0.3032653

#We see the value for the built in gamma density function dgamma() is the dame as ours
integrate(gammaPDF, 0, Inf, a = 1, b = 0.5)

## 1 with absolute error < 0.000034

integrate(dgamma, 0, Inf, shape = 1, rate = 0.5)

## 1 with absolute error < 0.000034

#Furthermore, when we integrate from 0 to Inf we get 1, which is expected.
#string.of.ones <- NULL
#for (i in 1:100) {
  #for (j in 1:100) {
    #string.of.ones <- c(string.of.ones, integrate(gammaPDF, 0, Inf, a = i, b = j)[1])
  #}
#}
integrate(gammaPDF, 0, Inf, a = 70, b = 1)           #We get a strange error

## Error in integrate(gammaPDF, 0, Inf, a = 70, b = 1):  non-finite function value

integrate(dgamma, 0, Inf, shape = 70, rate = 1)      #works for R function tho

## 1 with absolute error < 0.00000026

string.of.ones.redo <- NULL
for (i in 1:100) {
  for (j in 1:100) {
    string.of.ones.redo <- c(string.of.ones.redo, integrate(dgamma, 0, Inf, shape = i, rate = j
  }
}
sum(string.of.ones.redo > 0.999999)

## [1] 9994
```

```r
sum(string.of.ones.redo < 1.000001)                    #6 Values are not 1.

## [1] 10000

which(string.of.ones.redo < 0.999999)                  #which values

## value value value value value value
##  9602  9701  9702  9801  9802  9901

string.of.ones.redo[9602]                              #It's close to 0 instead.

## $value
## [1] 0.00005067519

integrate(dgamma, 0, Inf, shape = 97, rate = 2)        #Why is this?

## 0.00005067519 with absolute error < 0.000093
```

In the code above, I defined my own Gamma PDF function `gammaPDF()`, so that I could compare that to the built in `dgamma()` function that `R` uses. They return identical values for $\alpha = 1$ and $\beta = 2$ and both integrate to one. I tried running some for loops to set the integral for various values of α and β. This is the commented out code. I got an error when computing the integral for large shape values and low rate values when using my own defined function. I'm not sure why this is – I am sure it has something to do with `R`'s definition of the Gamma PDF being more clean and polished than my own, since I get no error when running that.

I ran the for loop for the built in `R` density function and the outputted values did not all equal 1 because of a strange rounding issue, but we can see that functionally most of them are one, but I assume `R` does some approximating when calculating integrals, so they are bounded below by 0.999999 and above by 1.000001. There were, however, six values that weren't one, and in fact they were all close to 0. You can see on the last line of code that the integral should equate to 1, but doesn't, it equals approximately zero. I have no idea why this – or why it only occurred for six values of α and β. I am going to assume this is an `R` mistake, perhaps something to do with numbers being too small and messing with `R`'s logic. After a conversation with professor Cipolli, we both agree that the strange behavior is most likely a result of `R`'s inability to work with extremely small numbers.

From a theoretical standpoint, the reason the Gamma probability density function integrates to one follows very quickly from some algebra on the Gamma function itself. Essentially, you divide by the gamma function on two sides of an equation, leaving you with one on one side, and a division by the Gamma function on the other, which is exactly what is in the PDF. This YouTube video explains it better at about one minute.

(c) Find the median for your two sets of parameter(s). Conduct some research to find the median based on our PDF to confirm that your numerical approach is correct.

Solution: **My two sets of parameters will be α (shape) equals 1, and β (rate) equals 0.5, and α (shape) equals 3, and β (rate) equals 1. I chose these because they result in qualitatively different density curves. In the code below, I use the `qgamma()` function in `R` to calculate the 50th quantile of the distribution for the two sets of parameters. It is approximately 1.386295 and 2.67406. We can test this is correct by integrating the PDF from 0 to 1. We get approximately 0.5 which affirms our answer.**

```r
qgamma(0.5, 1, 0.5)
```

```
## [1] 1.386294
```

```r
qgamma(0.5, 3, 1)
```

```
## [1] 2.67406
```

```r
integrate(dgamma, 0, 1.386295, shape = 1, rate = 0.5)
```

```
## 0.5000002 with absolute error < 0.0000000000000056
```

```r
integrate(dgamma, 0, 2.67406, shape = 3, rate = 1)
```

```
## 0.4999999 with absolute error < 0.0000000000000056
```

There is no closed-form solution for calculating the median – Only numerical estimates are possible. There are ways to approximate the median using the mean, but still no closed form solution (Gamma Function Wikipedia, 2021). A plethora of interesting properties have been discovered about the median, such as finding an asymptotic formula for the median, connecting the median to the Ramanujan sequence, and many other ways to bound the median, or expand it asymptotically (You, 2017).

(d) Graph the PDF for several values of the parameter(s) including the two sets you specified. What does changing the parameter(s) do to the shape of the PDF?

(e) Graph the CDF for the same values of the parameter(s) as you did in Question 1d. What does changing the parameter(s) do to the shape of the CDF? Comment on the aspects of the CDFs that show that the CDF is valid.

Solution: **I answer both part d and e here. Below, I have graphed the probability distribution function for my two specified parameters, as well as other combination of parameters too.**

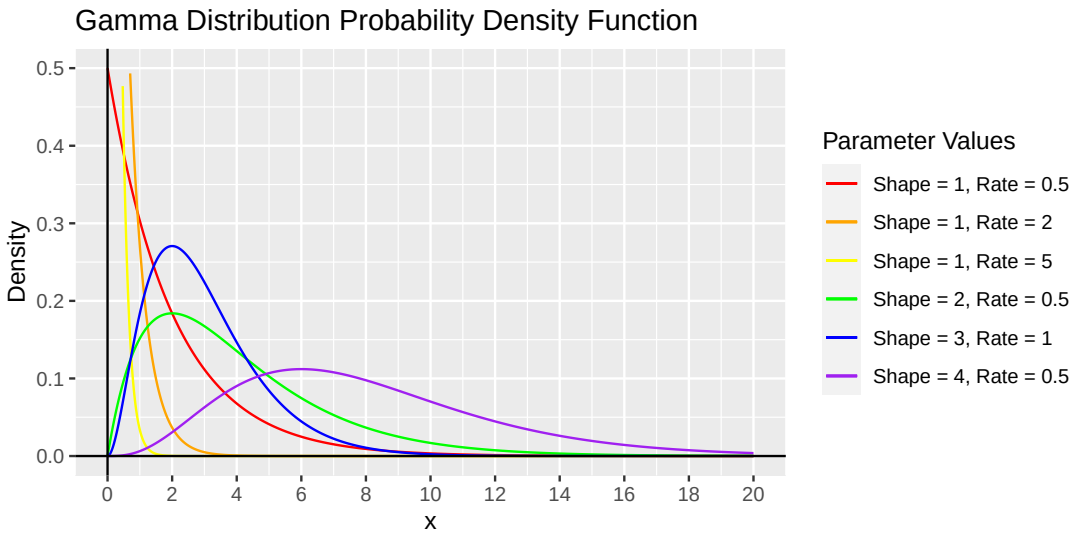
```r
x <- seq(0, 20, 0.01)           #values we evaluate at
dg.main <- dgamma(x, 1, 0.5)
dg.1.2 <- dgamma(x, 1, 2)
dg.1.5 <- dgamma(x, 1, 5)
dg.2.0.5 <- dgamma(x, 2, 0.5)
dg.3.1 <- dgamma(x, 3, 1)
dg.4.0.5 <- dgamma(x, 4, 0.5)
#above is the density curve evaluated at vec x for various parameters
gg1 <- ggplot() +
  geom_line(aes(x, dg.main, colour="Shape = 1, Rate = 0.5")) +
  geom_line(aes(x, dg.1.2, colour="Shape = 1, Rate = 2")) +
  geom_line(aes(x, dg.1.5, colour="Shape = 1, Rate = 5")) +
  geom_line(aes(x, dg.2.0.5, colour="Shape = 2, Rate = 0.5")) +
  geom_line(aes(x, dg.3.1, colour="Shape = 3, Rate = 1")) +
  geom_line(aes(x, dg.4.0.5, colour="Shape = 4, Rate = 0.5")) +
  scale_color_manual(name = "Parameter Values",
                     values = c("Shape = 1, Rate = 0.5" = "red",
                                "Shape = 1, Rate = 2" = "orange",
                                "Shape = 1, Rate = 5" = "yellow",
                                "Shape = 2, Rate = 0.5" = "green",
                                "Shape = 3, Rate = 1" = "blue",
                                "Shape = 4, Rate = 0.5" = "purple"
                               )
```

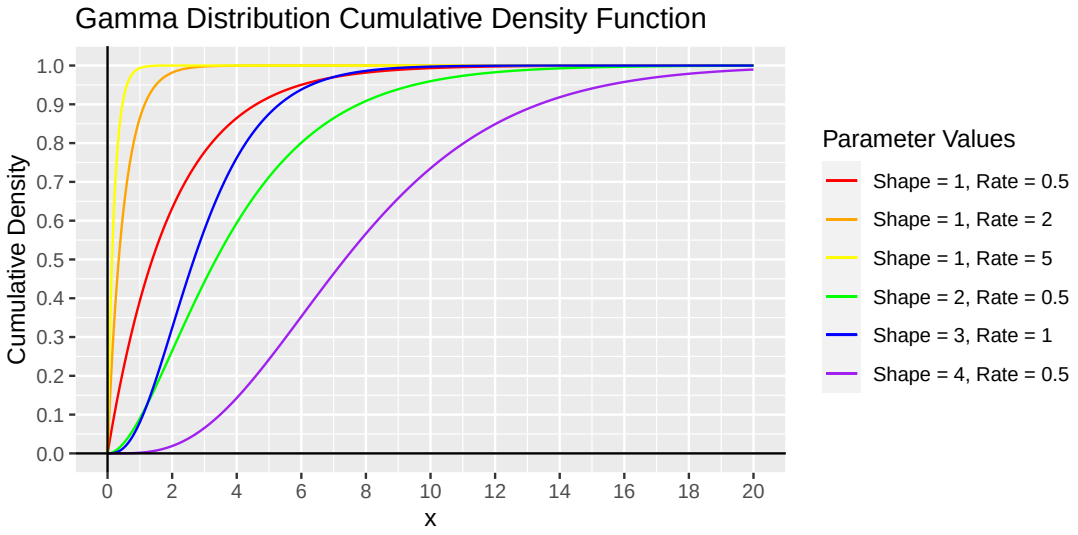
```
                    ) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) + #Fix x intervals
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10)) + #Fix y intervals
  ylim(c(0, 0.5)) +                                            #Limit y we see
  geom_hline(yintercept=0) +                                   #line on x-axis
  geom_vline(xintercept=0) +                                   #Line on y-axis
  ylab("Density") +                                            #y-axis label
  ggtitle("Gamma Distribution Probability Density Function")    #Title
```

```
## Scale for 'y' is already present.  Adding another scale for 'y', which will
## replace the existing scale.
```

```
#The code below calculates the CDF for values in the x vector defined above.
#The code is a bit cumbersome - most likely a cleaner way to do this.
cg.main <- NULL
  for (i in x) {cg.main <- c(cg.main, integrate(dgamma, 0, i, shape = 1, rate = 0.5)[1])}
  cg.main <- unlist(cg.main)
cg.1.2 <- NULL
  for (i in x) {cg.1.2 <- c(cg.1.2, integrate(dgamma, 0, i, shape = 1, rate = 2)[1])}
  cg.1.2 <- unlist(cg.1.2)
cg.1.5 <- NULL
  for (i in x) {cg.1.5 <- c(cg.1.5, integrate(dgamma, 0, i, shape = 1, rate = 5)[1])}
  cg.1.5 <- unlist(cg.1.5)
cg.2.0.5 <- NULL
  for (i in x) {cg.2.0.5 <- c(cg.2.0.5, integrate(dgamma, 0, i, shape = 2, rate = 0.5)[1])}
  cg.2.0.5 <- unlist(cg.2.0.5)
cg.3.1 <- NULL
  for (i in x) {cg.3.1 <- c(cg.3.1, integrate(dgamma, 0, i, shape = 3, rate = 1)[1])}
  cg.3.1 <- unlist(cg.3.1)
cg.4.0.5 <- NULL
  for (i in x) {cg.4.0.5 <- c(cg.4.0.5, integrate(dgamma, 0, i, shape = 4, rate = 0.5)[1])}
  cg.4.0.5 <- unlist(cg.4.0.5)

gg2 <- ggplot() +
  geom_line(aes(x, cg.main, colour="Shape = 1, Rate = 0.5")) +
  geom_line(aes(x, cg.1.2, colour="Shape = 1, Rate = 2")) +
  geom_line(aes(x, cg.1.5, colour="Shape = 1, Rate = 5")) +
  geom_line(aes(x, cg.2.0.5, colour="Shape = 2, Rate = 0.5")) +
  geom_line(aes(x, cg.3.1, colour="Shape = 3, Rate = 1")) +
  geom_line(aes(x, cg.4.0.5, colour="Shape = 4, Rate = 0.5")) +
  scale_color_manual(name = "Parameter Values",
                     values = c("Shape = 1, Rate = 0.5" = "red",
                                "Shape = 1, Rate = 2" = "orange",
                                "Shape = 1, Rate = 5" = "yellow",
                                "Shape = 2, Rate = 0.5" = "green",
                                "Shape = 3, Rate = 1" = "blue",
                                "Shape = 4, Rate = 0.5" = "purple"
                               )
                     ) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) + #Fix x intervals
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10)) + #Fix y intervals
  geom_hline(yintercept=0) +                                   #line on x-axis
  geom_vline(xintercept=0) +                                   #Line on y-axis
  ylab("Cumulative Density") +                                 #Y-axis label
  ggtitle("Gamma Distribution Cumulative Density Function")    #Title
```

```
gg1 / gg2                          #Overlay the graphs nicely on top of eachother
## Warning:  Removed 70 row(s) containing missing values (geom_path).
## Warning:  Removed 47 row(s) containing missing values (geom_path).
```





For the PDF, changing the shape parameter while keeping rate the same has the effect of making the distribution more platykurtic and more symmetrical (from visual inspection only). Increasing the rate while keeping the shape the same seems to shift more of the distribution towards the y-axis, making the PDF steeper and contain more of the probability for lower x-values. Low Shape parameters causes the distribution to look more like the exponential distribution, which makes sense because the exponential distribution is actually a special case of the gamma distribution (Gamma Function Wikipedia, 2021).

Increasing the shape parameter results in the CDF taking on more x values before asymptotically approaching a probability of one. This corresponds with my observation that increasing the shape results in a platykurtic distribution that is shifted outward. Increasing the rate has the opposite effect. A higher rate results in the CDF approaching one over fewer x observations. This, again, aligns with my obser-

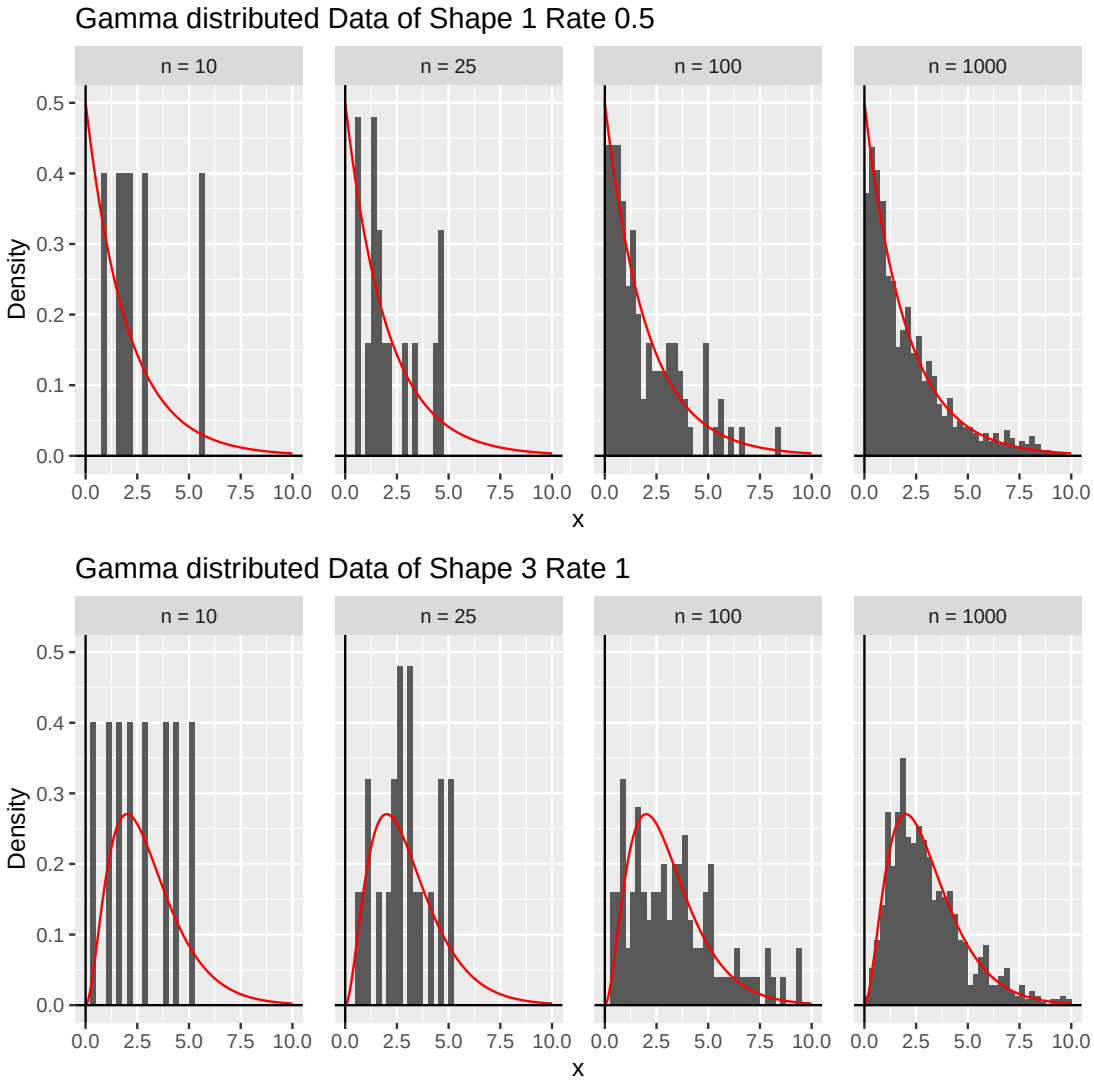
vations of the PDF, since the corresponding PDF was much more steep towards the y-axis for low rate values.
We know the CDF is valid because for the various parameters graphed, all the CDF curves approach one but do not exceed the value of one, which is theoretically correct since the area under PDF curve is one. Furthermore, the CDFs do not intersect, I'm not sure if this is a property that is necessarily true for all CDFs though.

(f) Generate a random sample of size $n = 10, 25, 100$, and 1000 for your two sets of parameter(s). In a 4×2 grid, plot a histogram of each set of data and superimpose the true density function at the specified parameter values. Interpret the results.

Solution: **Below are the desired graphs. We can see that as the number of observations increases, the closer the data fits into the theoretical distribution curve. The higher the n, the less the data varies from the PDF curve.**

```
#Gamma distributed data for first set of parameters
param.1.10 <- rgamma(10, shape = 1, rate = 0.5)
param.1.25 <- rgamma(25, shape = 1, rate = 0.5)
param.1.100 <- rgamma(100, shape = 1, rate = 0.5)
param.1.1000 <- rgamma(1000, shape = 1, rate = 0.5)
#Gamma distributed data for second set of parameters
param.2.10 <- rgamma(10, shape = 3, rate = 1)
param.2.25 <- rgamma(25, shape = 3, rate = 1)
param.2.100 <- rgamma(100, shape = 3, rate = 1)
param.2.1000 <- rgamma(1000, shape = 3, rate = 1)
#Append the vectors together
param.set.1.vec <- c(param.1.10, param.1.25, param.1.100, param.1.1000)
param.set.2.vec <- c(param.2.10, param.2.25, param.2.100, param.2.1000)
#For facet grid/wrapping
a <- replicate(10, "n = 10")
b <- replicate(25, "n = 25")
c <- replicate(100, "n = 100")
d <- replicate(1000, "n = 1000")
for.facetwrap <- c(a,b,c,d)
#Create a dataframe of all the gamma distributed data
param.dat <- data.frame(param.set.1.vec, param.set.2.vec, for.facetwrap)
#Function returns a plot per parameters you set in
facet.function <- function(param.set, p.shape, p.rate)  {
  ggplot(data = param.dat) +
    geom_histogram(                                          #Plot the data
      aes(x=param.set, y=..density..),
      binwidth=0.5,
      breaks=seq(0,10,0.25)
      ) +
    stat_function(                                           #Add in PDF
      fun=dgamma,
      args=list(shape=p.shape, rate=p.rate),
      color="red"
      ) +
    ylim(c(0, 0.5)) +                                        #Limit y we see
    xlim(c(0, 10)) +                                         #Limit y we see
    geom_hline(yintercept=0) +                               #line on x-axis
    geom_vline(xintercept=0) +                               #Line on y-axis
    ylab("Density") +                                           #y-axis label
    xlab("x") +                                            #y-axis label
```

```
    ggtitle(paste(c("Gamma distributed Data of Shape", p.shape, "Rate", p.rate), collapse = " "
    facet_grid(~factor(for.facetwrap, levels=c('n = 10','n = 25','n = 100','n = 1000')))
}
#Plot the graphs for the two sets of parameters. Options removes error messages
options(warn=-1) #disable warnings for a second
a <- facet.function(param.set.1.vec, 1, 0.5) + theme(panel.spacing = unit(1, "lines"))
b <- facet.function(param.set.2.vec, 3, 1) + theme(panel.spacing = unit(1, "lines"))
a / b
```




```
options(warn=0) #Re-enable warnings
```

2. Continue with the continuous distribution you selected for Question 1.

   (a) Provide the mean, standard deviation, skewness, and kurtosis of the PDF. Ensure to interpret each. **Solution: I want to thank Giancarlo for his help with this!**
   I used the moments from Wikipedia. Note that $shape = k$ and $scale = \theta$. $location = \epsilon$, which determines the lower bound for the distribution. We assume $\epsilon = 0$. The moments are:

$$Mean = \epsilon + \theta \cdot k \tag{4}$$

$$Standard\ Deviation = \theta \cdot \sqrt{k} \tag{5}$$

$$Skewness = \frac{2}{\sqrt{k}} \tag{6}$$

$$Kurtosis = 3 + \frac{6}{k} \tag{7}$$

For our parameter sets, which are shape = 1, scale = 2 (rate = 0.5), shape = 3, scale = 1 (rate = 1), the values are calculated in Figure 1 below. (Note: We take the location parameter, ϵ, to be 0 for both):

$Parameters\ (k, \theta)$	$Mean$	$Standard\ Deviation$	$Skewness$	$Kurtosis$
$(1, 2)$	2	2	2	9
$(3, 1)$	3	$\sqrt{3}$	$\frac{2\sqrt{3}}{3}$	5

Figure 1: Mean, Standard deviation, Skewness, and Kurtosis for our parameter sets.

Interpretation:
The mean for the Gamma distribution, calculated using the formula above, informs us about the central tendency of the distribution, depending on shape and scale. Because the gamma distribution is right-skewed for both sets of parameters that we are using, I would consider the median to be a more accurate measure of the central tendency. The standard deviation, calculated by $\sqrt{variance}$, tells us about the variability in the gamma distribution about the center. The skewness refers to the symmetry of the distribution – a high skewness corresponds to a less symmetric distribution. In the case of our parameters, I would expect to see a positive value for skewness because the distributions are right skewed, and have a long tail pointing towards the right (which is true based on our calculation). As for kurtosis, this measure refers to the peak of the distribution. A distribution is leptokurtic, if it has a tall peak and a kurtosis greater than 3 (so excess kurtosis is positive), whereas a distribution is platykurtic if the peak is flat, and the kurtosis is less than 3 (excess kurtosis is negative). Since our calculations show that kurtosis is greater than 3, our distributions will be leptokurtic.

(b) Generate a random sample of size $n = 10, 25, 100$, and 1000 for your two sets of parameter(s). Calculate the sample mean, standard deviation, skewness, and kurtosis. Interpret the results.
Solution:
I used e1071 in this code (Meyer et al., 2021). Note that the kurtosis() function here provides excess kurtosis, which is found by subtracting 3 from the kurtosis.

```
library(e1071)

set.seed(1) #set seed to ensure re-testing

n_10 = rgamma(10, shape = 1, scale =2)
n_25 = rgamma(25, shape = 1, scale =2)
n_100 = rgamma(100, shape =1, scale =2)
```

```r
n_1000 = rgamma(1000, shape =1, scale =2)
#making random samples

mean(n_10)

## [1] 1.721669

mean(n_25)

## [1] 1.634266

mean(n_100)

## [1] 2.024089

mean(n_1000)

## [1] 2.01548

sd(n_10)

## [1] 1.342847

sd(n_25)

## [1] 1.570013

sd(n_100)

## [1] 2.082571

sd(n_1000)

## [1] 2.01114

skewness(n_10)

## [1] 0.2190743

skewness(n_25)

## [1] 2.023939

skewness(n_100)

## [1] 1.585443

skewness(n_1000)

## [1] 1.728012

kurtosis(n_10)

## [1] -1.603052

kurtosis(n_25)

## [1] 5.194188

kurtosis(n_100)

## [1] 2.894753

kurtosis(n_1000)
```

```
## [1] 3.936587
```

```
n_10 = rgamma(10, shape = 3, scale =1)
n_25 = rgamma(25, shape = 3, scale =1)
n_100 = rgamma(100, shape =3, scale =1)
n_1000 = rgamma(1000, shape =3, scale =1)
```

```
mean(n_10)
```

```
## [1] 2.845343
```

```
mean(n_25)
```

```
## [1] 2.886486
```

```
mean(n_100)
```

```
## [1] 3.143448
```

```
mean(n_1000)
```

```
## [1] 3.028033
```

```
sd(n_10)
```

```
## [1] 1.603004
```

```
sd(n_25)
```

```
## [1] 1.610979
```

```
sd(n_100)
```

```
## [1] 1.731908
```

```
sd(n_1000)
```

```
## [1] 1.662081
```

```
skewness(n_10)
```

```
## [1] 0.4862953
```

```
skewness(n_25)
```

```
## [1] 0.4720326
```

```
skewness(n_100)
```

```
## [1] 1.118056
```

```
skewness(n_1000)
```

```
## [1] 1.014066
```

```
kurtosis(n_10)
```

```
## [1] -0.6511291
```

```
kurtosis(n_25)
```

```
## [1] -1.208189
```

```
kurtosis(n_100)
```

```
## [1] 1.284554
```

```
kurtosis(n_1000)
```

```
## [1] 1.248744
```

Interpretation:
For the mean of the first parameter set (shape = 1, scale = 2), we see that the mean and the standard deviation get closer to the true mean and standard deviation of 2 as n increases to a 1000. For the skewness, the value is very low at $n = 10$, and it gets closest to the true skewness at $n = 25$, but then slightly deviates from the true value. The skewness here is positive, which indicates that the distribution is right-skewed (which I had expected based on the visual inspection). The excess kurtosis is negative at $n = 10$, indicating that for this parameter, the distribution is platykurtic. However, as n increases, the excess kurtosis becomes positive, indicating that now, the distribution is leptokurtic. The excess kurtosis at $n = 25$ is closest to the true kurtosis of 9 (note, I added 3 to translate between the two), but then as n increases, the kurtosis deviates and then comes closer to the true value, similar to what we saw for skewness.
For the mean of the second parameter set (shape = 3, scale = 1), we see that the mean and the standard deviation get closer to the true mean of 3 as n approaches 1000 and standard deviation of $\sqrt{3} \approx 1.73$ at $n = 100$. At $n = 1000$, the standard deviation moves slightly away from the true value. For the skewness, the value is low for $n = 10$ and $n = 25$, but gets closest to the true skewness of $\frac{2\sqrt{3}}{3} \approx 1.15$ at $n = 100$, but then slightly deviates from the true value (similar to standard deviation). The skewness here is positive, which indicates that the distribution is right-skewed (which I had expected based on the visual inspection and the calculations). The excess kurtosis is negative at $n = 10$ and $n = 25$, indicating that for this parameter, the distribution is platykurtic for these values of n. However, as n increases, the excess kurtosis becomes positive, indicating that now, the distribution is leptokurtic. The excess kurtosis at $n = 100$ and $n = 1000$ are relatively closer to the true kurtosis of 5 (add three again), but they are not yet close to the true value.
To briefly comment on the above tasks, it seems that for certain parameters, like mean and standard deviation, increasing sample size for a random sample allows us to get closer to the true parameters. But, for skewness and kurtosis, there is a bit more uncertainty and variation – to truly see a pattern for these two values, I would have to simulate the task many times, with n varying over a larger range, and a smaller stepsize (to truly observe the exact details and trend). Then, I would be able to comment with a bit more certainty as to how they change with varying n.

(c) Generate a random sample of size $n = 10$ for your two sets of parameter(s). Calculate the method of moments estimator(s) and maximum likelihood estimator(s). In a 1×2 grid, plot a histogram of each set of data with (1) the method of moments estimated distribution, (2) the maximum likelihood estimated distribution, and superimpose the true distribution in both.
Solution: In this question, I am using ggplot2 (Wickham, 2016), patchwork (Pedersen, 2020), and nleqslv (Hasselman, 2018). I also used the Freedman-Diaconis rule (Freedman and Diaconis, 1981) to determine the bin width for my histograms; it is conventionally used and is the default for base R. I prefer using this rule. Below are the functions I used to produce my plots!

```
library(nleqslv)
library(ggplot2)
library(patchwork)
set.seed(3) #so that the plots do not change between runs
gamma.MOM<-function(par, data){

  #borrowed from Professor Cipolli
  shape <- par[1]
  scale <- par[2]

  EX1 <- scale*shape
  EX2 <- (scale^2)*(shape * (shape +1))
  #moments found online.
```

```r
  xbar1 <- mean(data)
  xbar2 <- mean(data^2)

  c(EX1-xbar1, EX2-xbar2)
}

gamma.LL<-function(par, data, neg=T){
  #Also borrowed from Professor Cipolli
  shape <- par[1]
  scale <- par[2]

  ll <- sum(dgamma(x=data, shape = shape, scale = scale, log =T))
  ifelse(neg, -ll, ll)
}

find.MOM.MLE = function(n, par){

  Density = rgamma(n, shape = par[1], scale =par[2])
  MOM = nleqslv(x = c(1,1),
                fn = gamma.MOM,
                data = Density)
  MLE = optim(par = c(1,1),
              fn = gamma.LL,
              data = Density)
  #finds MOM and MLE for random data

  #print(c(MOM$x, MLE$par))
  #print for testing


  #using Freedman-Diaconis rule which is conventionally used to determine
  #binwidths for histograms (default for base R)


  estimated.MOM = rgamma(n, shape = MOM$x[1], scale = MOM$x[2])
  bw.MOM = 2 * IQR(estimated.MOM, na.rm =T) /
    length(na.omit(estimated.MOM))^(1/3)

  true.dist = dgamma(seq(0,10, 0.1), shape = par[1],
                     scale = par[2])

  plot_MOM = ggplot() +
       geom_histogram(aes(x= estimated.MOM,
                          y= ..density..),
                      binwidth = bw.MOM,
                      color = "mediumpurple", fill ="white")+
       theme_minimal()+ ylab("Density")+xlab("Observations")+
       geom_hline(yintercept = 0)+
    geom_line(aes(x= seq(0,10, 0.1),y = true.dist),
              color ="forestgreen", lwd = 0.7)+
    ggtitle(paste("Methods of Moments Estimator n =", n, sep = " "))+
    theme(plot.title = element_text(hjust = 0.5, size = 9))+
    xlim(c(0,10))+ylim(c(0, max(true.dist)))
```

```
  plot_MOM = plot_MOM +
    annotate("text", x = 7.5, y = 0.9*max(true.dist),
             label = paste("Shape =", round(MOM$x[1],3)))+
    annotate("text", x = 7.5, y = 0.8*max(true.dist),
             label = paste("Scale =", round(MOM$x[2],3)))

  estimated.MLE = rgamma(n, shape = MLE$par[1], scale = MLE$par[2])

  bw.MLE = 2 * IQR(estimated.MLE, na.rm = T) /
    length(na.omit(estimated.MLE))^(1/3)

  plot_MLE = ggplot() +
    geom_histogram(aes(x= estimated.MLE,
                       y= ..density..),
                   binwidth = bw.MLE,
                   color = "mediumpurple", fill ="white")+
    theme_minimal()+ ylab("Density")+xlab("Observations")+
    geom_hline(yintercept = 0)+
    geom_line(aes(x= seq(0,10, 0.1),y = true.dist),
              color ="forestgreen", lwd = 0.7)+
    ggtitle(paste("Maximum Likelihood Estimator n =", n, sep=" ")) +
    theme(plot.title = element_text(hjust = 0.5, size = 9))+
    xlim(c(0,10)) +ylim(c(0, max(true.dist)))

  plot_MLE = plot_MLE+
    annotate("text", x = 7.5, y = 0.9*max(true.dist),
             label = paste("Shape =", round(MLE$par[1],3)))+
    annotate("text", x = 7.5, y = 0.8*max(true.dist),
             label = paste("Scale =", round(MLE$par[2],3)))

  (plot_MOM + plot_MLE)
}
```

Note that I will analyze all these plots in detail in the last part of the question!

```
find.MOM.MLE(10, c(1,2))
```

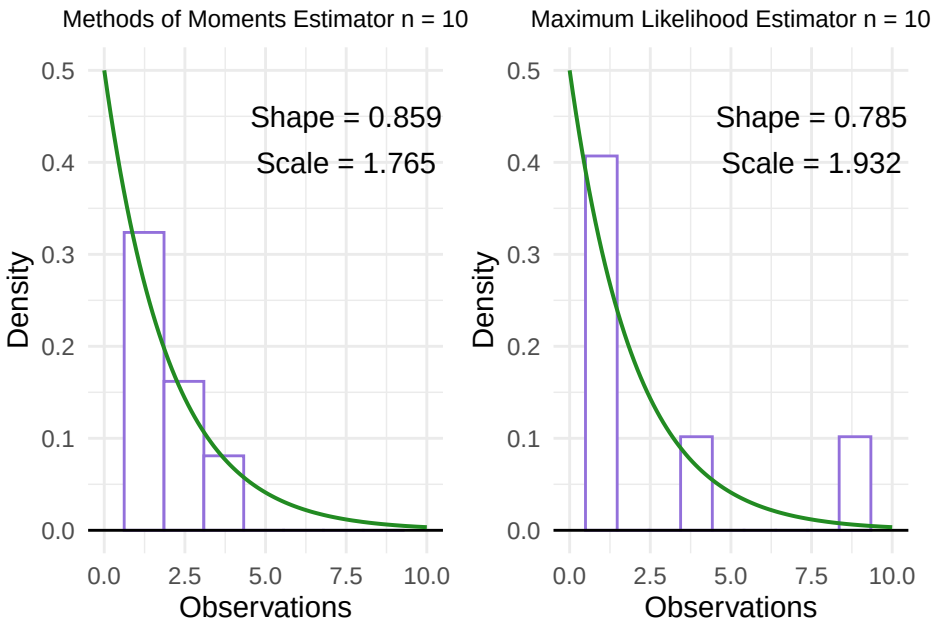


Figure 2: Density histograms using Methods of Moments and Maximum Likelihood Estimator for the Gamma Distribution. The true distribution is superimposed with a forest green color, with a shape of 1 and a scale of 2. The estimated parameters are shown on the plots.

```
find.MOM.MLE(10, c(3,1))
```

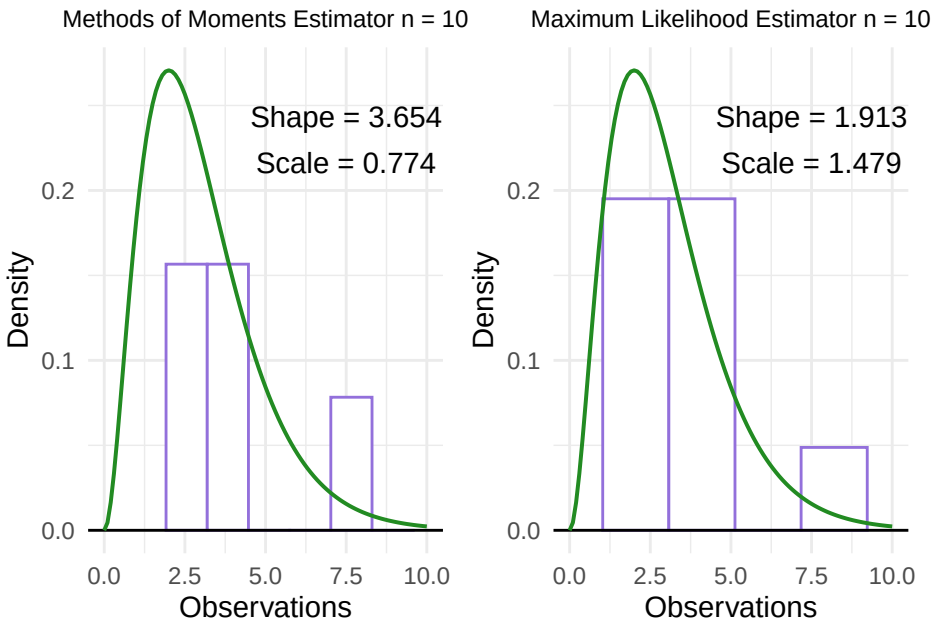


Figure 3: Density histograms using Methods of Moments and Maximum Likelihood Estimator for the Gamma Distribution. The true distribution is superimposed with a forest green color, with a shape of 3 and a scale of 1. The estimated parameters are shown on the plots.

(d) Generate a random sample of size $n = 25$ for your two sets of parameter(s). Calculate the method of moments estimator(s) and maximum likelihood estimator(s). In a 1×2 grid, plot a histogram of each set of data with (1) the method of moments estimated distribution, (2) the maximum likelihood estimated distribution, and superimpose the true distribution in both.
Solution:

```
find.MOM.MLE(25, c(1,2))
```

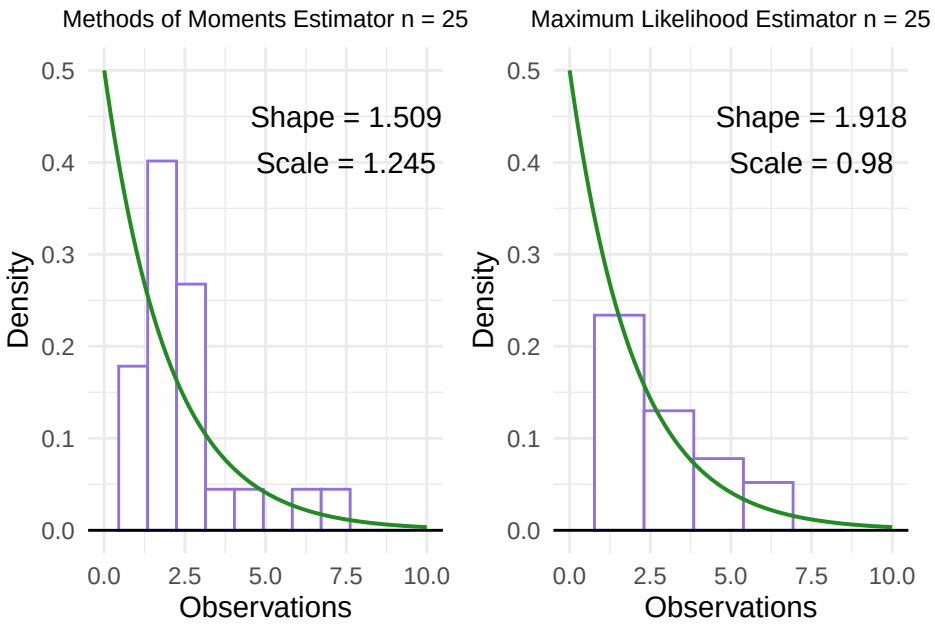


Figure 4: Density histograms using Methods of Moments and Maximum Likelihood Estimator for the Gamma Distribution. The true distribution is superimposed with a forest green color, with a shape of 1 and a scale of 2. The estimated parameters are shown on the plots.

```
find.MOM.MLE(25, c(3,1))
```

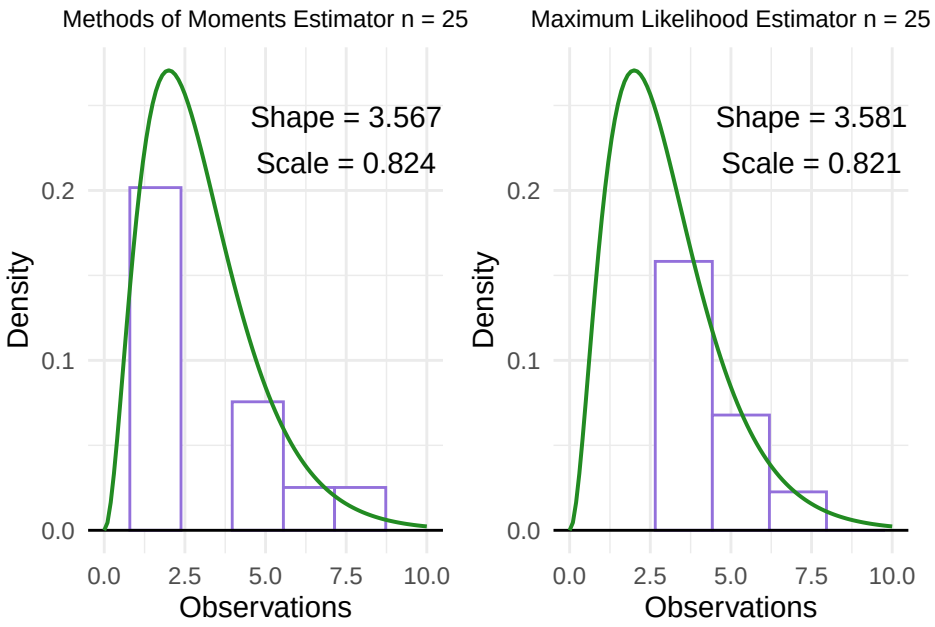


Figure 5: Density histograms using Methods of Moments and Maximum Likelihood Estimator for the Gamma Distribution. The true distribution is superimposed with a forest green color, with a shape of 3 and a scale of 1. The estimated parameters are shown on the plots.

(e) Generate a random sample of size $n = 100$ for your two sets of parameter(s). Calculate the method of moments estimator(s) and maximum likelihood estimator(s). In a 1×2 grid, plot a histogram of each set of data with (1) the method of moments estimated distribution, (2) the maximum likelihood estimated distribution, and superimpose the true distribution in both.
Solution:

```
find.MOM.MLE(100, c(1,2))
```

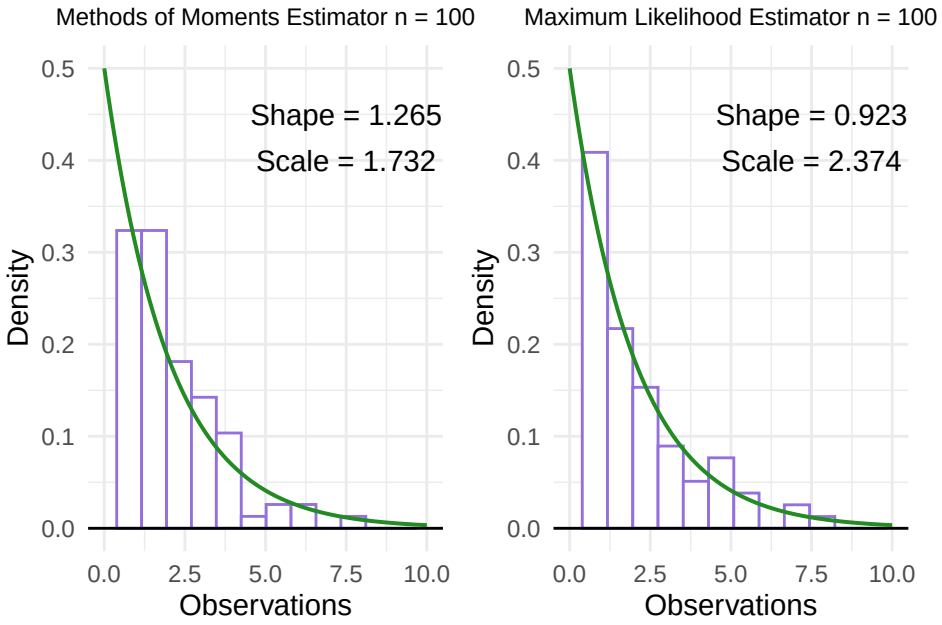


Figure 6: Density histograms using Methods of Moments and Maximum Likelihood Estimator for the Gamma Distribution. The true distribution is superimposed with a forest green color, with a shape of 1 and a scale of 2. The estimated parameters are shown on the plots.

```
find.MOM.MLE(100, c(3,1))
```

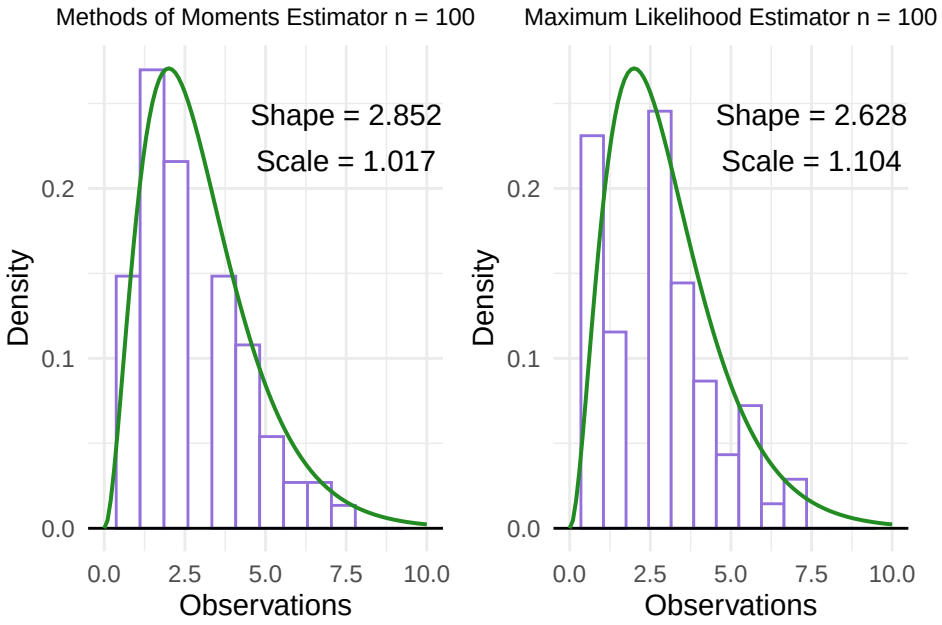


Figure 7: Density histograms using Methods of Moments and Maximum Likelihood Estimator for the Gamma Distribution. The true distribution is superimposed with a forest green color, with a shape of 3 and a scale of 1. The estimated parameters are shown on the plots.

(f) Generate a random sample of size $n = 100$ for your two sets of parameter(s). Calculate the method of moments estimator(s) and maximum likelihood estimator(s). In a 1×2 grid, plot a histogram of each set of data with (1) the method of moments estimated distribution, (2) the maximum likelihood estimated distribution, and superimpose the true distribution in both.
Solution:
I am assuming you mean $n = 1000$!

```
find.MOM.MLE(1000, c(1,2))
```

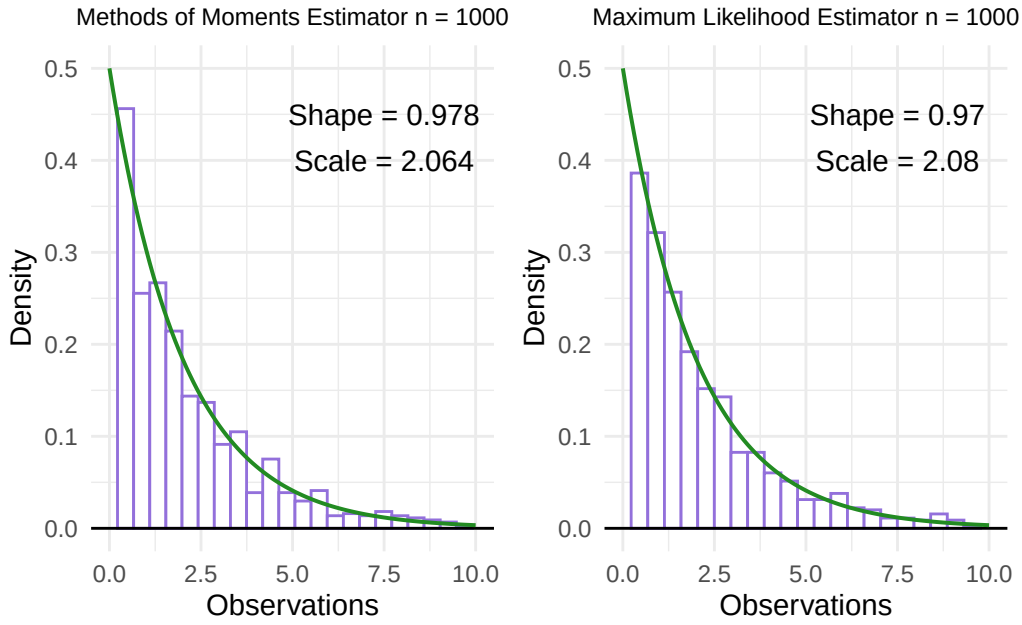


Figure 8: Density histograms using Methods of Moments and Maximum Likelihood Estimator for the Gamma Distribution. The true distribution is superimposed with a forest green color, with a shape of 1 and a scale of 2. The estimated parameters are shown on the plots.

```
find.MOM.MLE(1000, c(3,1))
```

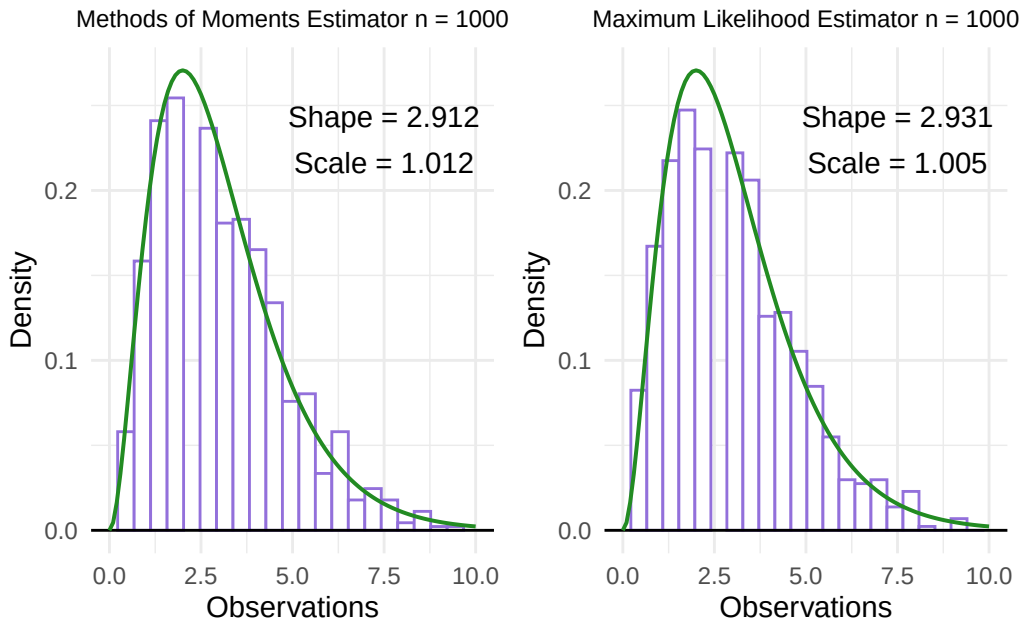


Figure 9: Density histograms using Methods of Moments and Maximum Likelihood Estimator for the Gamma Distribution. The true distribution is superimposed with a forest green color, with a shape of 3 and a scale of 1. The estimated parameters are shown on the plots.

(g) Comment on the results of parts (c)-(f).

Solution:

For $n = 10$ case, for both Figure 2 and Figure 3, we can observe that the estimated parameters for both estimators are different from the true parameters. As a result, by visual inspection, we also observe that the true distribution and the estimated distributions are not close to each other. Interestingly, for $n = 25$, we see a slight improvement in Figure 4 and Figure 5, but there is still a significant (not in the statistical sense) mismatch. To elaborate, the estimated parameters are still quite different from the true parameters, especially for Figure 4: for Maximum Likelihood estimator, we see a shape of 1.918, which is quite different from a true shape parameter of 1. Similarly, a scale of 0.98 is definitely far off from a scale of 2. But, the corresponding histogram does not reflect that mismatch; it is somewhat close to the true distribution! This emphasizes my biggest takeaway: do not rely on either quantitative or qualitative analysis alone, but instead use both simultaneously! This idea is especially obvious with normality tests (a Shapiro-Wilk test might have a very small p-value, but the histogram might look very Gaussian!). Now for $n = 100$, Figure 6 and 7 show that the estimated parameters are getting closer to the true parameters. Similarly, the visuals also indicate that the estimated distribution and true distribution are becoming a better fit (but still not perfect!). Once again, in Figure 6, the scale of 2.374 is not very close to 2, but the visuals suggest that the estimated distribution is a good fit with the true distribution. For Figure 8 and Figure 9, we see the application of the Weak Law of Large Numbers (WLLG): compared to $n = 10$ or $n = 25$, the estimated distributions for $n = 1000$ are a much better fit to the true distribution! The parameters are also quite close to the true parameters. This is because the WLLG says that as our sample size increases, our estimated point estimators from the sample will become arbitrarily closer to the true population parameters.

Note, I would like to mention a few potential sources of bias. Firstly, since the number of bins increase as the sample size increases, we may be seeing a better fit due to an increase in bins. However, in my opinion, using 30 bins for a sample size of $n = 10$ might have made visualizing the

results difficult, and lead to some other bias! Because of this tradeoff, I think using the additional quantitative assessment of the estimated parameters compared to population parameters is very important to form a holistic conclusion about the plots! I wanted to use Kolmogorov-Smirnov test to check how different the estimated distribution and true distribution is, but the KS test's p-value tells us if the distributions are statistically different ($p \leq 0.05$), but if $p > 0.05$, we cannot draw a conclusion without performing some kind of power analysis to find the Type-II error. Our Type-II error must be less than 0.2 for us to say that we can safely not reject the null hypothesis. To avoid these issues, I decided to display the estimated parameters instead!
Secondly, all of these plots come from one random run, so any deviation from the true distribution may even be due to random chance! What if I run this code again, and everything fits really well, or does not fit well at all? For this reason, it is important to simulate such tasks an arbitrarily large number of times, and find the average and standard error of those runs! Then, we could assess, while minimizing bias, whether the increase in the quality of fit is truly due to the increase in n.

3. Select a discrete distribution (not the Poisson). It does not have to be one that we cover in the notes! To explore the PMF of your distribution, specify two sets of parameter(s) for your distribution.

   (a) **History** Discuss what types of random variables are modeled with your distribution. Be sure to include a discussion about the support and ensure to provide the mass function, and CDF. This requires some internet research – what's the history of the distribution, why was it created and named? What are some exciting applications of this distribution? Cite all of your sources.
   Solution:
   Let us consider the binomial distribution. The binomial distribution is a discrete distribution that models the number of successes of n Bernoulli trials. It is used to model any experiment of the sort when we can denote one outcome as "success" or "outcome of interest", provided that this outcome has a set probability of happening. Examples of experiments that can be modeled using the binomial distribution include, flipping a coin (pick heads or tails to be our success), drawing a card from a standard deck (consider some card a success), etc. (Philippou and Antzoulakos, 2011), (Wikipedia, 2021)

   Consider that we perform an experiment a total number of n times. We can have 0 successes or n successes, as well as everything in between. So, the support of the distribution goes from 0 to n, and has $n+1$ elements. Here are the PMF and the CDF for the distribution. (Philippou and Antzoulakos, 2011)
   PMF:
   $$f(x) = P(X = x) = \binom{n}{x} p^x q^{n-x}, x = \{0, 1, 2, \cdots, n\}$$
   CDF:
   $$F(x) = P(X \leq x) = \sum_{i=0}^{x} \binom{n}{i} p^i q^{n-i}$$
   The distribution was derived by Jacob Bernoulli, with special cases being derived by Blaise Pascal. The name of the distribution comes from the binomial theorem. (Philippou and Antzoulakos, 2011), (Wikipedia, 2021)

   (b) Show that you have a valid PMF. You can show this approximately by calculating the series in a repeat loop until probability mass evaluations are infinitesimally small.
   Solution:

```
# Specifying two sets of parameters:

# SET 1 (trials = 40, probability = 0.5)
t1 <- 40
prob1 <- 0.5

# SET 2 (trials = 20, probability = 0.25)
t2 <- 20
prob2 <- 0.25
```

   We show the validity of a PMF by 1 - adding up all the individual probabilities and checking to see if they add up to zero. 2 - checking to see if all the probabilities are nonnegative

```
# Let's check this for the first set:

x <- 0                      # first possible number of successes is zero
v <- NULL                   # creating a null vector to append the probabilities to
repeat{
  a <- dbinom(x,t1,prob1)   # dbinom function takes value and returns probability
  v <- append (v,a)         # adding an element to v with each loop
```

```
  x <- x + 1                # considering the next possible number of successes
  if(sum(v)>=0.999999999){ # mimicking "infinitesimally small"
    break
  }
}
print(sum(v))               # sum of all the probabilities should equal 1
```

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

```
v >= 0                      # the probabilities should be nonnegative
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
# Now let's check it for the second set:

x <- 0                      # first possible nr. of success is zero
v <- NULL                   # creating a null vector to append the probabilities to
repeat{
  a <- dbinom(x,t2,prob2)   # dbinom function takes value and returns probability
  v <- append (v,a)         # adding an element to v with each loop
  x <- x + 1                # considering the next possible number of successes
  if(sum(v)>=0.999999999){ # mimicking "infinitesimally small"
    break
  }
}
print(sum(v))               # sum of all the probabilities should equal 1
```

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

```
v >= 0                      # the probabilities should be nonnegative
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE
```

□

(c) Find the median for your two sets of parameter(s). Conduct some research to find the median based on our PMF to confirm that your numerical approach is correct.
Solution:

```
# Median for the first set (median = 50th percentile)
qbinom(0.5, t1, prob1)
```

```
## [1] 20
```

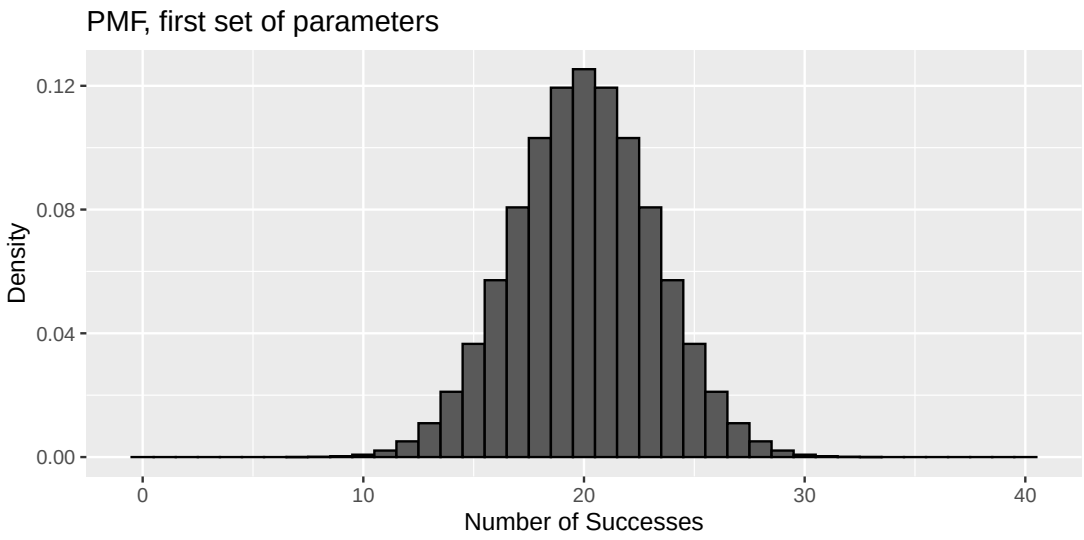
```
# Median for the second set (median = 50th percentile)
qbinom(0.5, t2, prob2)
```

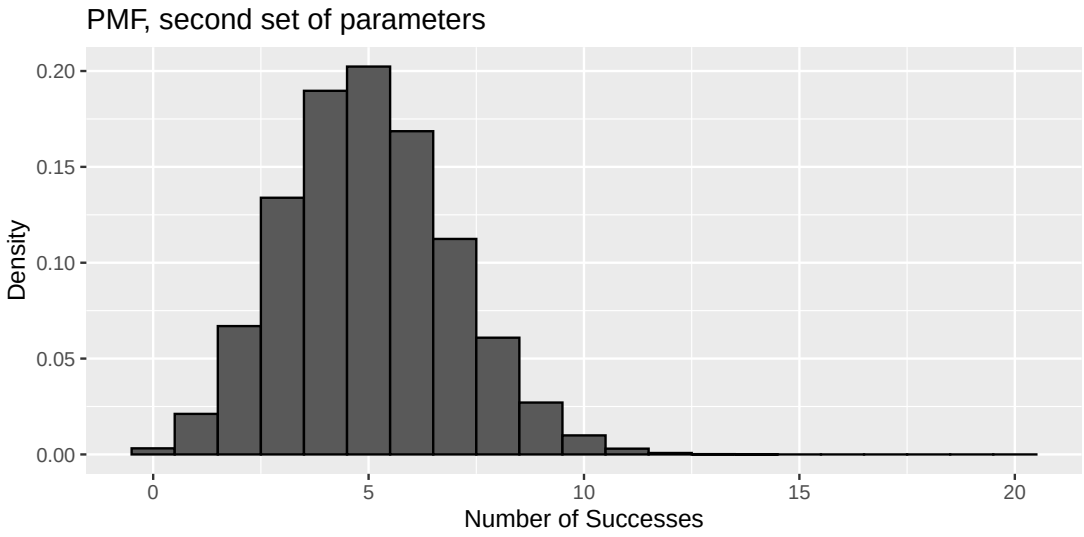
```
## [1] 5
```

Theoretically, if np is an integer, then the median equals mp, which supports our results. (Wikipedia, 2021) □

(d) Graph the PMF for several values of the parameter(s) including the two sets you specified. What does changing the parameter(s) do to the shape of the PMF?
Solution:

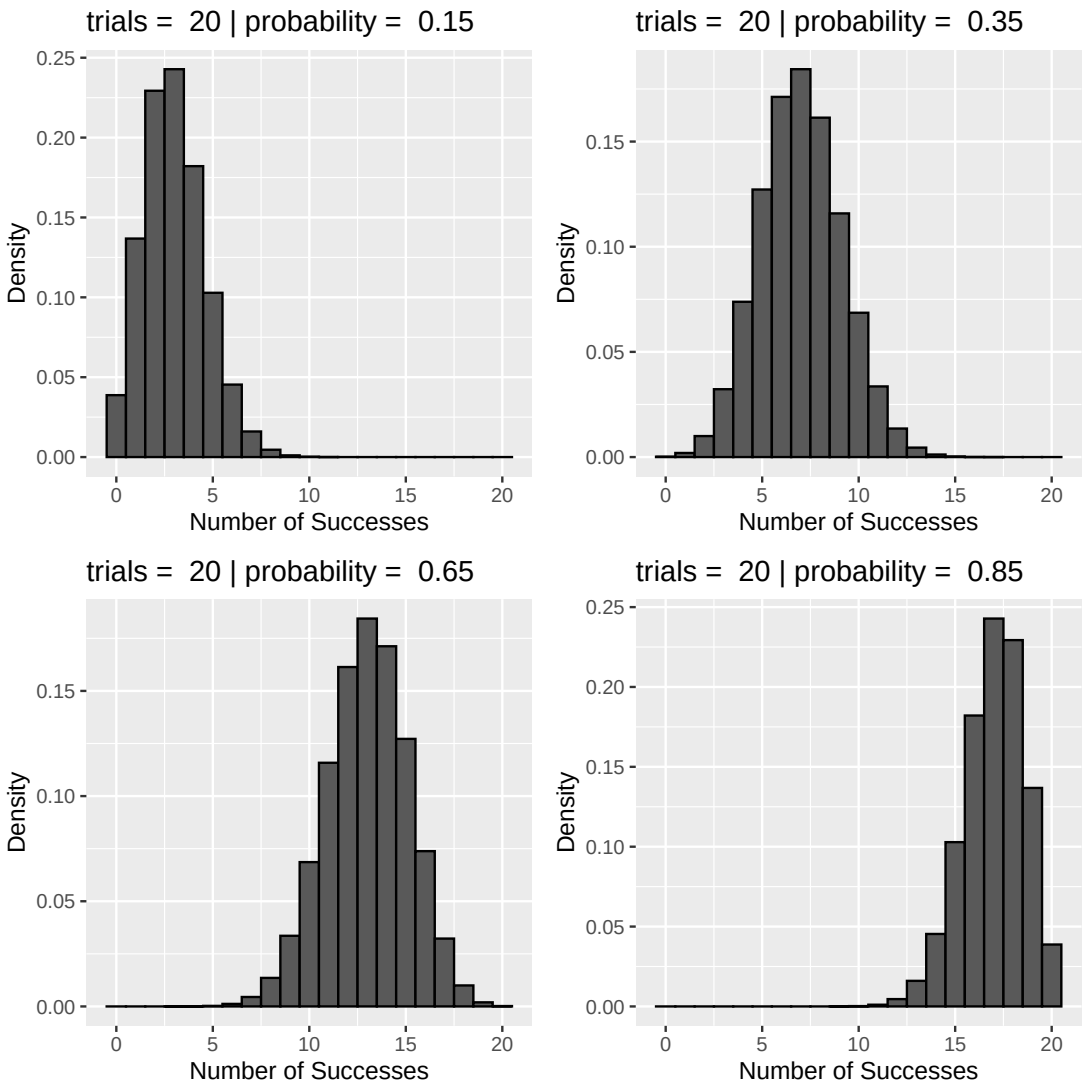
```r
# Graphing the PMF for the first set
x1 <- 0:t1   # for any number of trials t, there are t+1 possible successes.
p1 <- dbinom(x1,t1,prob1) # dbinom function takes value and returns probability

pmf1 <- qplot(x1,                # telling R to plot x
              weight = p1,       # weighted by the probabilities p
              geom="histogram",  # telling R we want a histogram
              bins=t1+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = "PMF, first set of parameters",
              xlab = "Number of Successes",
              ylab = "Density")

# Graphing the PMF for the second set
x2 <- 0:t2   # for any number of trials t, there are t+1 possible successes.
p2 <- dbinom(x2,t2,prob2) # dbinom function takes value and returns probability

pmf2 <- qplot(x2,                # telling R to plot x
              weight = p2,       # weighted by the probabilities p
              geom="histogram",  # telling R we want a histogram
              bins=t2+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = "PMF, second set of parameters",
              xlab = "Number of Successes",
              ylab = "Density")

# Displaying the graphs

pmf1+pmf2 +plot_layout(ncol=1)
```






```r
# Let's keep the number of trials t constant to evaluate how different p's change the PMF.

# a) Parameters: t=20, prob=0.15
t3 <- 20
prob3 <- 0.15

x3 <- 0:t3  # for any number of trials t, there are t+1 possible successes
p3 <- dbinom(x3,t3,prob3) # dbinom function takes value and returns probability

pmf3 <- qplot(x3,                # telling R to plot x
              weight = p3,       # weighted by the probabilities p
              geom="histogram",  # telling R we want a histogram
              bins=t3+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = paste("trials = ",t3, "| probability = ", prob3),
              xlab = "Number of Successes",
              ylab = "Density")
```

```r
# b) Parameters: t = 20, prob=0.35
t4 <- 20
prob4 <- 0.35

x4 <- 0:t4  # for any number of trials t, there are t+1 possible successes
p4 <- dbinom(x4,t4,prob4) # dbinom function takes value and returns probability

pmf4 <- qplot(x4,                # telling R to plot x
              weight = p4,       # weighted by the probabilities p
              geom="histogram",  # telling R we want a histogram
              bins=t4+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = paste("trials = ", t4, "| probability = ", prob4),
              xlab = "Number of Successes",
              ylab = "Density")

# c) Parameters: t=20, prob=0.65
t5 <- 20
prob5 <- 0.65

x5 <- 0:t5  # for any number of trials t there are t+1 possible successes
p5 <- dbinom(x5,t5,prob5) # dbinom function takes value and returns probability

pmf5 <- qplot(x5,                # telling R to plot x
              weight = p5,       # weighted by the probabilities p
              geom="histogram",  # telling R we want a histogram
              bins=t5+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = paste("trials = ", t5, "| probability = ", prob5),
              xlab = "Number of Successes",
              ylab = "Density")


# d) Parameters: t=20, prob=0.85
t6 <- 20
prob6 <-0.85

x6 <- 0:t6  # for any number of trials t, there are t+1 possible successes
p6 <- dbinom(x6,t6,prob6) # dbinom function takes value and returns probability

pmf6 <- qplot(x6,                # telling R to plot x
              weight = p6,       # weighted by the probabilities p
              geom="histogram",  # telling R we want a histogram
              bins=t6+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = paste("trials = ", t6, "| probability = ", prob6),
              xlab = "Number of Successes",
              ylab = "Density")

# Displaying the graphs:
(pmf3+pmf4)/(pmf5+pmf6)
```

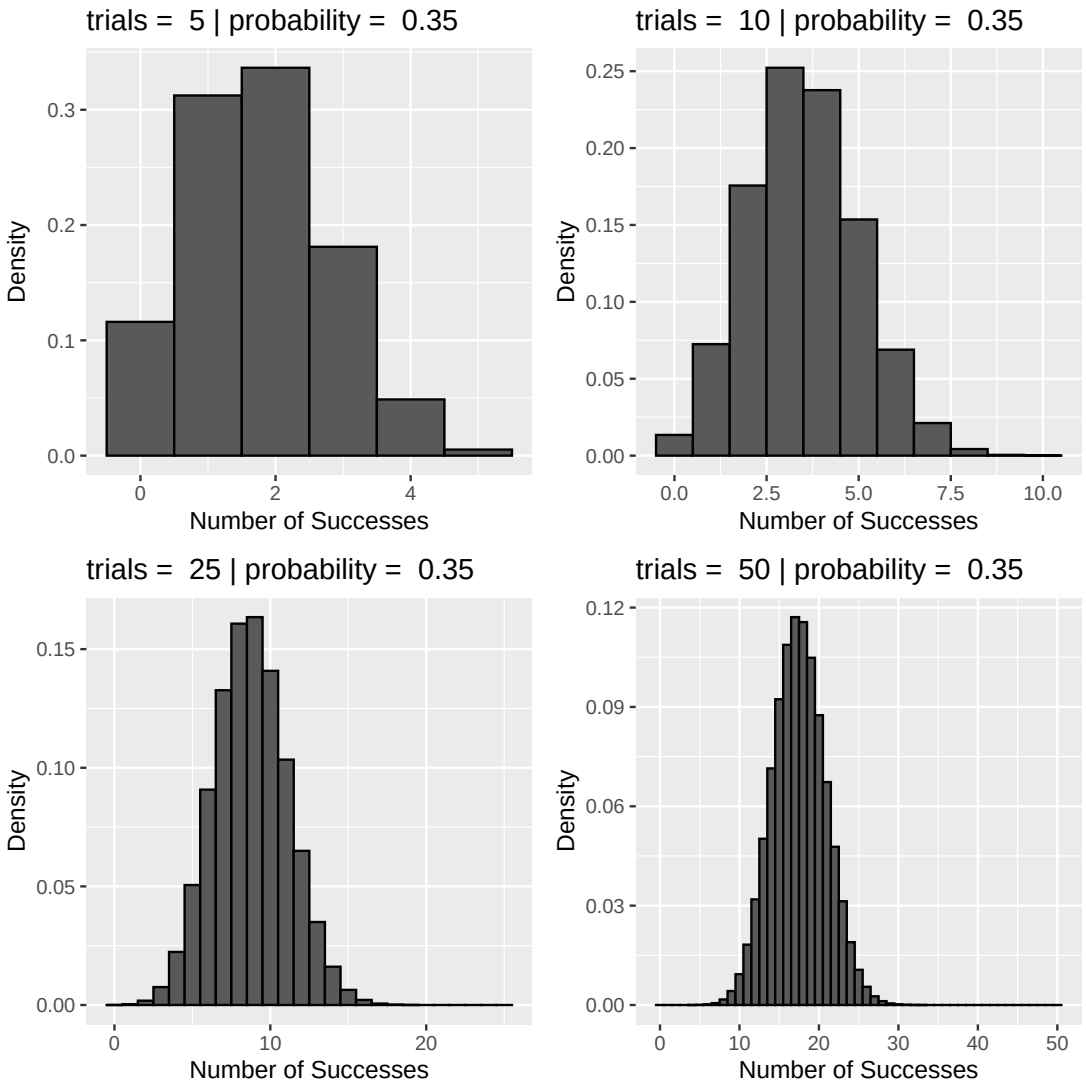


We notice that, as p gets larger, the distribution shifts more and more to the right.

```
# Now, let's keep p constant to evaluate how different t's change the PMF.

# a) Parameters: t=5, prob=0.35
t7 <- 5
prob7 <- 0.35

x7 <- 0:t7  # for any number of trials t, there are t+1 possible successes
p7 <- dbinom(x7,t7,prob7) # dbinom function takes value and returns probability

pmf7 <- qplot(x7,                # telling R to plot x
              weight = p7,       # weighted by the probabilities p
              geom="histogram",  # telling R we want a histogram
              bins=t7+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = paste("trials = ",t7, "| probability = ", prob7),
```

```
              xlab = "Number of Successes",
              ylab = "Density")


# b) Parameters: t=10, prob=0.35
t8 <- 10
prob8 <- 0.35

x8 <- 0:t8  # for any number of trials t, there are t+1 possible successes
p8 <- dbinom(x8,t8,prob8) # dbinom function takes value and returns probability

pmf8 <- qplot(x8,                # telling R to plot x
              weight = p8,       # weighted by the probabilities p
              geom="histogram",  # telling R we want a histogram
              bins=t8+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = paste("trials = ",t8, "| probability = ", prob8),
              xlab = "Number of Successes",
              ylab = "Density")


# c) Parameters: t=25, prob=0.35
t9 <- 25
prob9 <- 0.35

x9 <- 0:t9  # for any number of trials t, there are t+1 possible successes
p9 <- dbinom(x9,t9,prob9) # dbinom function takes value and returns probability

pmf9 <- qplot(x9,                # telling R to plot x
              weight = p9,       # weighted by the probabilities p
              geom="histogram",  # telling R we want a histogram
              bins=t9+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = paste("trials = ",t9, "| probability = ", prob9),
              xlab = "Number of Successes",
              ylab = "Density")

# d) Parameters: t=50, prob=0.35
t10 <- 50
prob10 <- 0.35

x10 <- 0:t10  # for any number of trials t, there are t+1 possible successes
p10 <- dbinom(x10,t10,prob10) # dbinom function takes value and returns probability

pmf10 <- qplot(x10,                # telling R to plot x
              weight = p10,        # weighted by the probabilities p
              geom="histogram",    # telling R we want a histogram
              bins=t10+1,          # we have t+1 possible successes
              col=I("black"),      # histogram border color
              main = paste("trials = ",t10, "| probability = ", prob10),
              xlab = "Number of Successes",
              ylab = "Density")
```

```
# Displaying the graphs
(pmf7+pmf8)/(pmf9+pmf10)
```



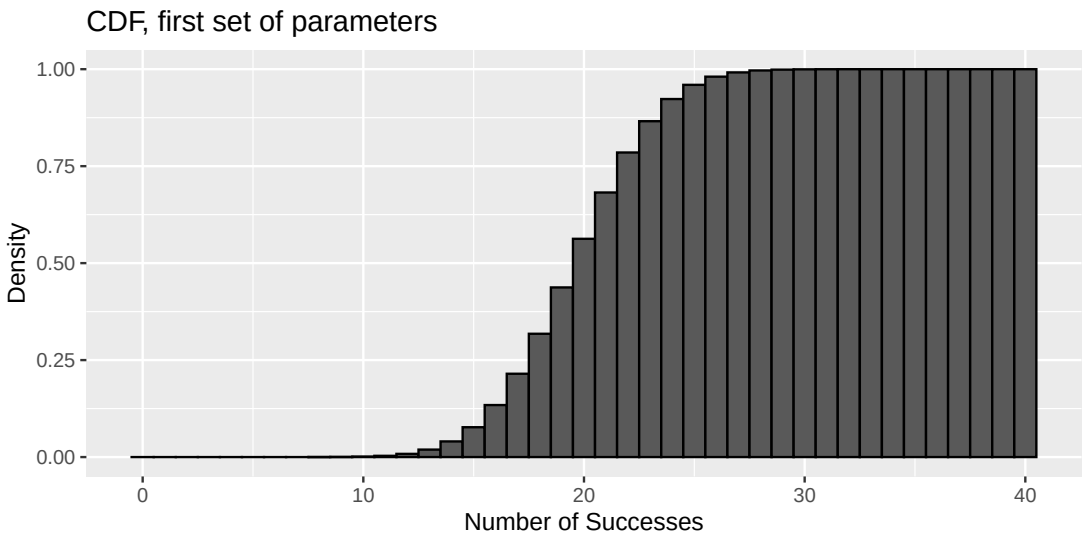
We observe that increasing the number of trials does not change the shape of the distribution, but it does increase the number of bins. □

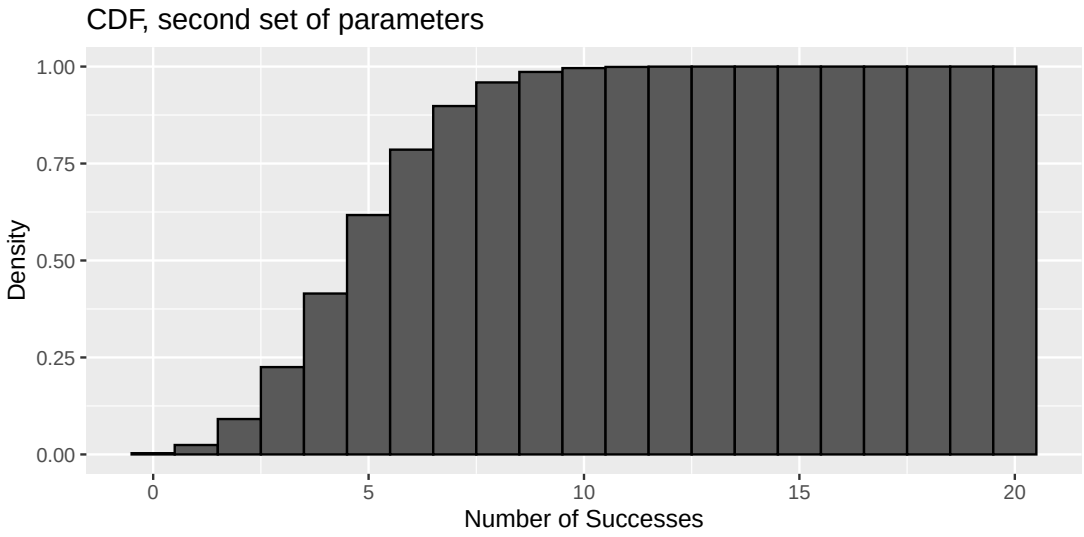
(e) Graph the CDF for the same values of the parameter(s) as you did in Question 3d. What does changing the parameter(s) do to the shape of the CDF? Comment on the aspects of the CDFs that show that the CDF is valid.

Solution:

```
# Graphing the CDF for the first set
x1 <- 0:t1   # for any number of trials t, there are t+1 possible successes
p1 <- dbinom(x1,t1,prob1) # dbinom function takes value and returns probability
cp1 <- cumsum(p1) # finding the cumulative sum

cdf1 <- qplot(x1,                 # telling R to plot x
              weight = cp1,       # weighted by the cumulative probabilities
```

```
              geom="histogram",  # telling R we want a histogram
              bins=t1+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = "CDF, first set of parameters",
              xlab = "Number of Successes",
              ylab = "Density")

# Graphing the CDF for the second set
x2 <- 0:t2   # for any number of trials t, there are t+1 possible successes
p2 <- dbinom(x2,t2,prob2) # dbinom function takes value and returns probability
cp2 <- cumsum(p2) # finding the cumulative sum

cdf2 <- qplot(x2,                # telling R to plot x
              weight = cp2,      # weighted by the cumulative probabilities
              geom="histogram",  # telling R we want a histogram
              bins=t2+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = "CDF, second set of parameters",
              xlab = "Number of Successes",
              ylab = "Density")

# Displaying the graphs

cdf1+cdf2 +plot_layout(ncol=1)
```

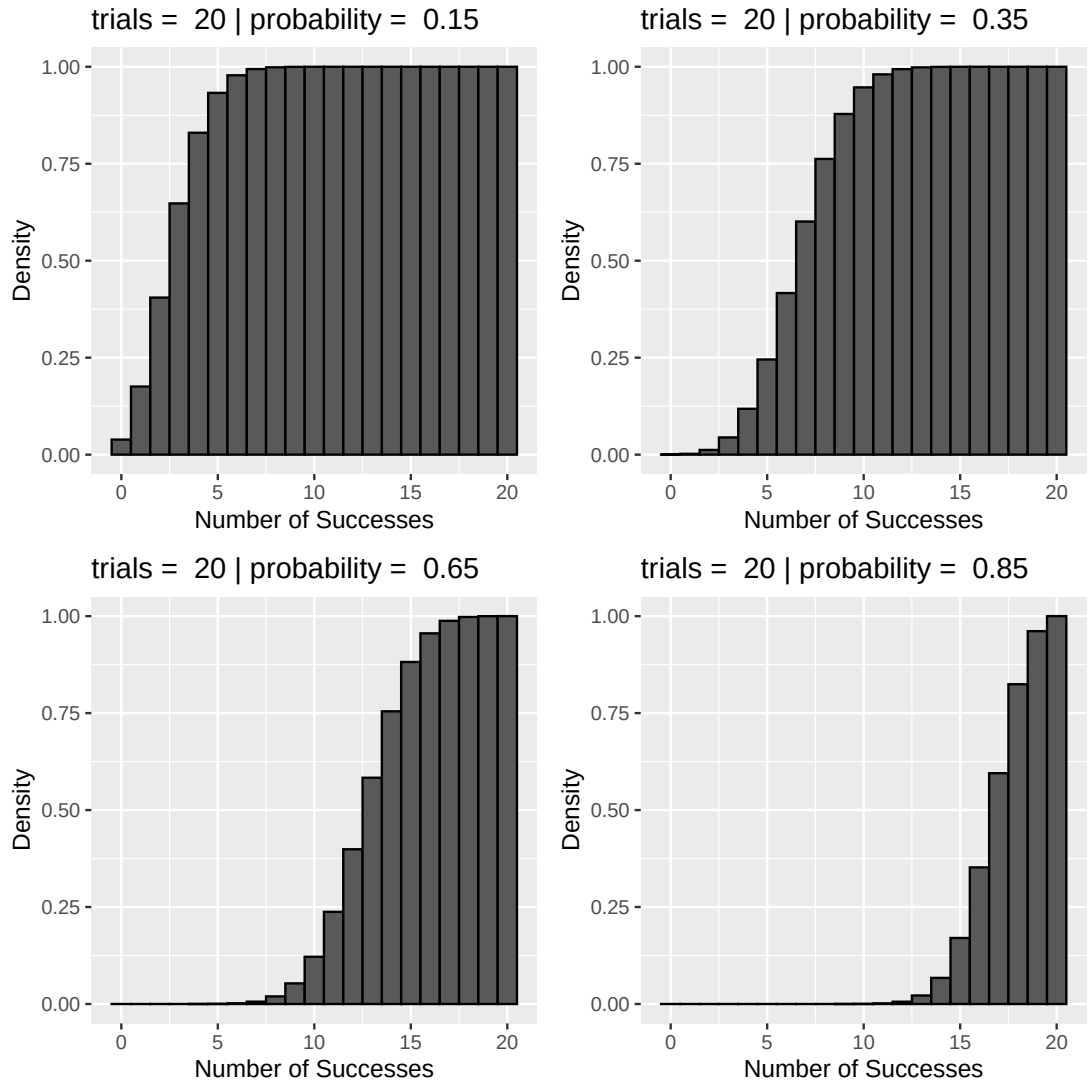





```r
# Let's keep the number of trials t constant to evaluate how different p's change the CDF

# a) Parameters: t=20, prob=0.15
t3 <- 20
prob3 <- 0.15

x3 <- 0:t3  # for any number of trials t, there are t+1 possible successes
p3 <- dbinom(x3,t3,prob3) # dbinom function takes value and returns probability
cp3 <- cumsum(p3) # finding the cumulative sum

cdf3 <- qplot(x3,                 # telling R to plot x
              weight = cp3,       # weighted by the cumulative probabilities
              geom="histogram",   # telling R we want a histogram
              bins=t3+1,          # we have t+1 possible successes
              col=I("black"),     # histogram border color
              main = paste("trials = ",t3, "| probability = ", prob3),
              xlab = "Number of Successes",
              ylab = "Density")
```

```r
# b) Parameters: t=20, prob=0.35
t4 <- 20
prob4 <- 0.35

x4 <- 0:t4  # for any number of trials t, there are t+1 possible successes
p4 <- dbinom(x4,t4,prob4) # dbinom function takes value and returns probability
cp4 <- cumsum(p4) # finding the cumulative sum

cdf4 <- qplot(x4,                # telling R to plot x
              weight = cp4,      # weighted by the cumulative probabilities
              geom="histogram",  # telling R we want a histogram
              bins=t4+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = paste("trials = ", t4, "| probability = ", prob4),
              xlab = "Number of Successes",
              ylab = "Density")

# c) Parameters: t=20, prob=0.65
t5 <- 20
prob5 <- 0.65

x5 <- 0:t5  # for any number of trials t, there are t+1 possible successes
p5 <- dbinom(x5,t5,prob5) # dbinom function takes value and returns probability
cp5 <- cumsum(p5) # finding the cumulative sum

cdf5 <- qplot(x5,                # telling R to plot x
              weight = cp5,      # weighted by the cumulative probabilities
              geom="histogram",  # telling R we want a histogram
              bins=t5+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = paste("trials = ", t5, "| probability = ", prob5),
              xlab = "Number of Successes",
              ylab = "Density")


# d) Parameters: t=20, prob=0.85
t6 <- 20
prob6 <-0.85

x6 <- 0:t6  # for any number of trials t, there are t+1 possible successes
p6 <- dbinom(x6,t6,prob6) # dbinom function takes value and returns probability
cp6 <- cumsum(p6) # finding the cumulative sum

cdf6 <- qplot(x6,                # telling R to plot x
              weight = cp6,      # weighted by the cumulative probabilities
              geom="histogram",  # telling R we want a histogram
              bins=t6+1,         # we have t+1 possible successes
              col=I("black"),    # histogram border color
              main = paste("trials = ", t6, "| probability = ", prob6),
              xlab = "Number of Successes",
              ylab = "Density")

# Displaying the graphs:
```

```
(cdf3+cdf4)/(cdf5+cdf6)
```

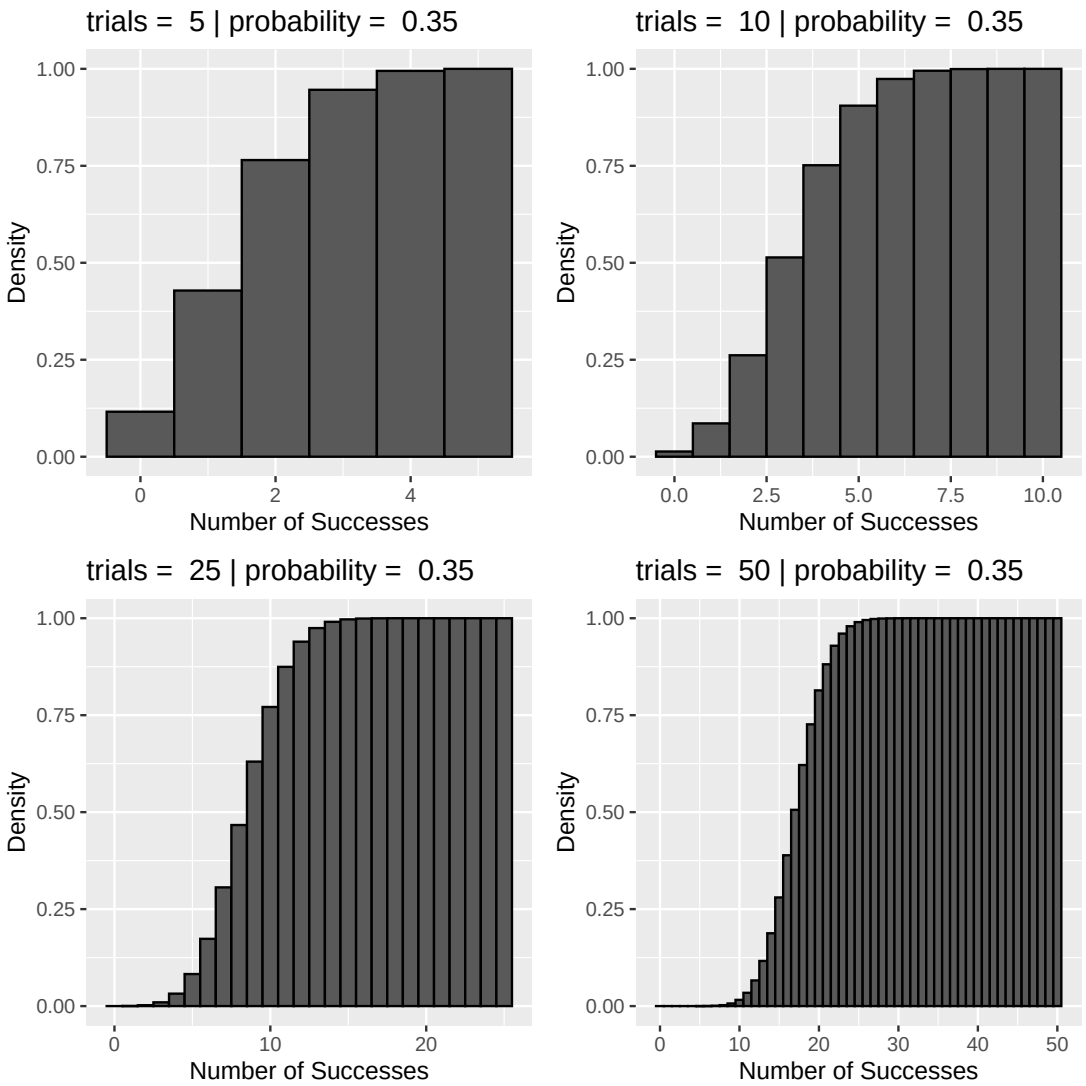


As p gets larger, the CDF rises later in the support.

```
# Now, let's keep p constant to evaluate how different t's change the CDF.

# a) Parameters: t=5, prob=0.35
t7 <- 5
prob7 <- 0.35

x7 <- 0:t7  # for any number of trials t, there are t+1 possible successes
p7 <- dbinom(x7,t7,prob7) # dbinom function takes value and returns probability
cp7 <- cumsum(p7) # finding the cumulative sum

cdf7 <- qplot(x7,                # telling R to plot x
              weight = cp7,      # weighted by the cumulative probabilities
              geom="histogram",  # telling R we want a histogram
```

```
              bins=t7+1,          # we have t+1 possible successes
              col=I("black"),     # histogram border color
              main = paste("trials = ",t7, "| probability = ", prob7),
              xlab = "Number of Successes",
              ylab = "Density")


# b) Parameters: t=10, prob=0.35
t8 <- 10
prob8 <- 0.35

x8 <- 0:t8  # for any number of trials t, there are t+1 possible successes
p8 <- dbinom(x8,t8,prob8) # dbinom function takes value and returns probability
cp8 <- cumsum(p8) # finding the cumulative sum

cdf8 <- qplot(x8,                 # telling R to plot x
              weight = cp8,       # weighted by the cumulative probabilities
              geom="histogram",   # telling R we want a histogram
              bins=t8+1,          # we have t+1 possible successes
              col=I("black"),     # histogram border color
              main = paste("trials = ",t8, "| probability = ", prob8),
              xlab = "Number of Successes",
              ylab = "Density")


# c) Parameters: t=25, prob=0.35
t9 <- 25
prob9 <- 0.35

x9 <- 0:t9  # for any number of trials t, there are t+1 possible successes
p9 <- dbinom(x9,t9,prob9) # dbinom function takes value and returns probability
cp9 <- cumsum(p9) # finding the cumulative sum

cdf9 <- qplot(x9,                 # telling R to plot x
              weight = cp9,       # weighted by the cumulative probabilities
              geom="histogram",   # telling R we want a histogram
              bins=t9+1,          # we have t+1 possible successes
              col=I("black"),     # histogram border color
              main = paste("trials = ",t9, "| probability = ", prob9),
              xlab = "Number of Successes",
              ylab = "Density")

# d) Parameters: t=50, prob=0.35
t10 <- 50
prob10 <- 0.35

x10 <- 0:t10  # for any number of trials t, there are t+1 possible successes
p10 <- dbinom(x10,t10,prob10) # dbinom function takes value and returns probability
cp10 <- cumsum(p10) # finding the cumulative sum

cdf10 <- qplot(x10,                # telling R to plot x
              weight = cp10,       # weighted by the cumulative probabilities
              geom="histogram",    # telling R we want a histogram
```

```
                bins=t10+1,           # we have t+1 possible successes
                col=I("black"),       # histogram border color
                main = paste("trials = ",t10, "| probability = ", prob10),
                xlab = "Number of Successes",
                ylab = "Density")


# Displaying the graphs
(cdf7+cdf8)/(cdf9+cdf10)
```



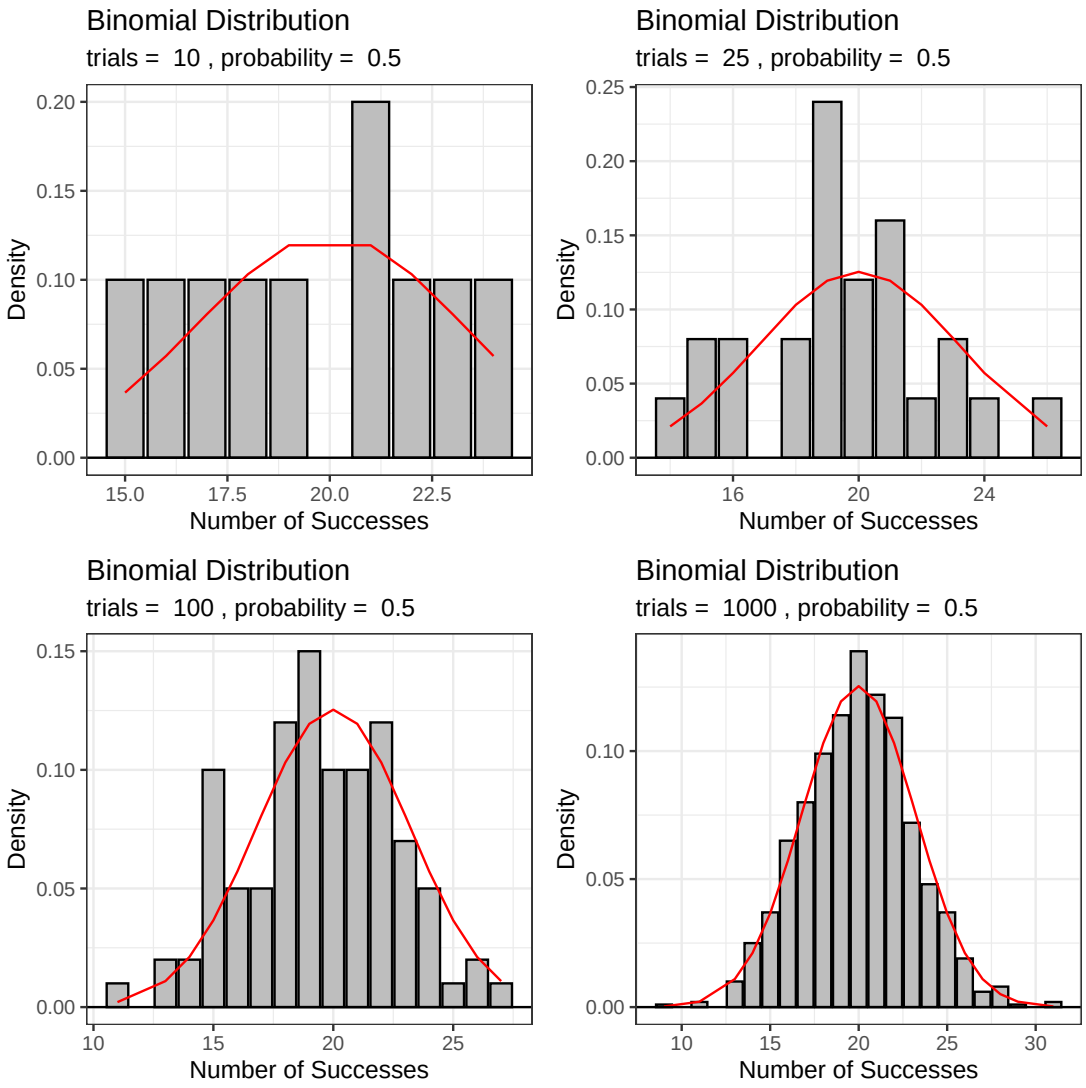
Increasing the number of trials does not change the shape of the CDF, but it does increase the number of bins. Note that the CDF is valid since the values start at 0 and end at 1 for each of the graphs. □

(f) Generate a random sample of size $n = 10, 25, 100$, and 1000 for your two sets of parameter(s). In a 4×2 grid, plot a histogram (with bin size 1) of each set of data and superimpose the true mass function at the specified parameter values. Interpret the results.
Solution:

```r
##########################
# For SET 1
##########################

# Insert parameters:
n <- 10   # number of values to generate
t <- 40   # number of trials
p <- 0.5  # probability of success for each trial

sample <- rbinom(n,t,p)  # store the sample
ggdat <- data.frame(prop.table(table(sample)))  # make a relative frequency table
ggdat$sample <- as.numeric(as.character(ggdat$sample)) # converting factor into numeric
                                                       # while preserving value
ggdat$prob = dbinom(ggdat$sample, t, p)  # calculate the true binomial probability
                                         # of each number of successes


pmf1 <- ggplot(ggdat) +
        geom_bar(aes(x=sample, y=Freq),     # plotting the histogram
                 stat = "identity",
                 color = "black",
                 fill = "grey") +
        geom_line(aes(x=sample, y=prob),    # plotting the true mass function
                  color = "red") +
        xlab("Number of Successes") +       # x axis label
        ylab("Density") +                   # y axis label
        ggtitle("Binomial Distribution",
                subtitle = paste("trials = ", n, ", probability = ", p)) + # adding title to p
        geom_hline(yintercept=0) +          # adding a line for the x-axis
        theme_bw()                          # removes grey background


# Insert parameters:
n <- 25   # number of values to generate
t <- 40   # number of trials
p <- 0.5  # probability of success for each trial

sample <- rbinom(n,t,p)  # store the sample
ggdat <- data.frame(prop.table(table(sample)))  # make a relative frequency table
ggdat$sample <- as.numeric(as.character(ggdat$sample)) # converting factor into numeric
                                                       # while preserving value
ggdat$prob = dbinom(ggdat$sample, t, p)  # calculate the true binomial probability
                                         # of each number of successes


pmf2 <- ggplot(ggdat) +
        geom_bar(aes(x=sample, y=Freq),     # plotting the histogram
                 stat = "identity",
                 color = "black",
                 fill = "grey") +
        geom_line(aes(x=sample, y=prob),    # plotting the true mass function
                  color = "red") +
        xlab("Number of Successes") +       # x axis label
```

```
        ylab("Density") +                    # y axis label
        ggtitle("Binomial Distribution",
                subtitle = paste("trials = ", n, ", probability = ", p)) + # adding title to p
        geom_hline(yintercept=0) +           # adding a line for the x-axis
        theme_bw()                           # removes grey background


# Insert parameters:
n <- 100   # number of values to generate
t <- 40   # number of trials
p <- 0.5  # probability of success for each trial

sample <- rbinom(n,t,p)  # store the sample
ggdat <- data.frame(prop.table(table(sample)))  # make a relative frequency table
ggdat$sample <- as.numeric(as.character(ggdat$sample)) # converting factor into numeric
                                                       # while preserving value
ggdat$prob = dbinom(ggdat$sample, t, p)  # calculate the true binomial probability
                                         # of each number of successes


pmf3 <- ggplot(ggdat) +
        geom_bar(aes(x=sample, y=Freq),     # plotting the histogram
                 stat = "identity",
                 color = "black",
                 fill = "grey") +
        geom_line(aes(x=sample, y=prob),    # plotting the true mass function
                  color = "red") +
        xlab("Number of Successes") +       # x axis label
        ylab("Density") +                   # y axis label
        ggtitle("Binomial Distribution",
                subtitle = paste("trials = ", n, ", probability = ", p)) + # adding title to p
        geom_hline(yintercept=0) +          # adding a line for the x-axis
        theme_bw()                          # removes grey background


# Insert parameters:
n <- 1000   # number of values to generate
t <- 40   # number of trials
p <- 0.5  # probability of success for each trial

sample <- rbinom(n,t,p)  # store the sample
ggdat <- data.frame(prop.table(table(sample)))  # make a relative frequency table
ggdat$sample <- as.numeric(as.character(ggdat$sample)) # converting factor into numeric
                                                       # while preserving value
ggdat$prob = dbinom(ggdat$sample, t, p)  # calculate the true binomial probability
                                         # of each number of successes


pmf4 <- ggplot(ggdat) +
        geom_bar(aes(x=sample, y=Freq),     # plotting the histogram
                 stat = "identity",
                 color = "black",
                 fill = "grey") +
```

```
        geom_line(aes(x=sample, y=prob),     # plotting the true mass function
                  color = "red") +
        xlab("Number of Successes") +        # x axis label
        ylab("Density") +                    # y axis label
        ggtitle("Binomial Distribution",
                subtitle = paste("trials = ", n, ", probability = ", p)) + # adding title to p
        geom_hline(yintercept=0) +           # adding a line for the x-axis
        theme_bw()                           # removes grey background


(pmf1+pmf2)/(pmf3+pmf4)
```

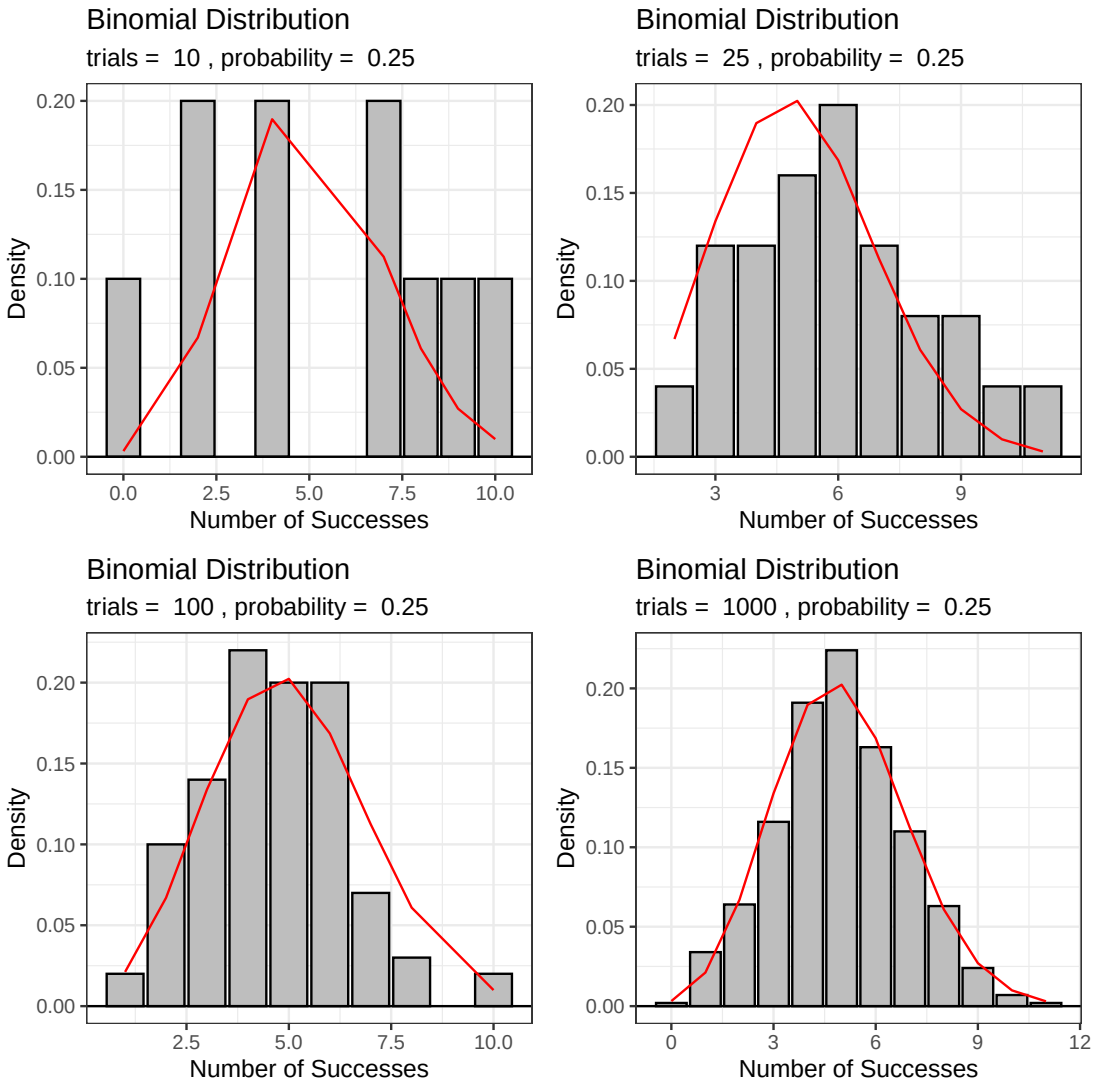



```
###################################
# For SET 2
###################################

# Insert parameters:
n <- 10    # number of values to generate
```

```r
t <- 20    # number of trials
p <- 0.25  # probability of success for each trial

sample <- rbinom(n,t,p)  # store the sample
ggdat <- data.frame(prop.table(table(sample)))  # make a relative frequency table
ggdat$sample <- as.numeric(as.character(ggdat$sample)) # converting factor into numeric
                                                       # while preserving value
ggdat$prob = dbinom(ggdat$sample, t, p)  # calculate the true binomial probability
                                         # of each number of successes


pmf5 <- ggplot(ggdat) +
        geom_bar(aes(x=sample, y=Freq),     # plotting the histogram
                 stat = "identity",
                 color = "black",
                 fill = "grey") +
        geom_line(aes(x=sample, y=prob),    # plotting the true mass function
                  color = "red") +
        xlab("Number of Successes") +       # x axis label
        ylab("Density") +                   # y axis label
        ggtitle("Binomial Distribution",
                subtitle = paste("trials = ", n, ", probability = ", p)) + # adding title to p
        geom_hline(yintercept=0) +          # adding a line for the x-axis
        theme_bw()                          # removes grey background


# Insert parameters:
n <- 25    # number of values to generate
t <- 20    # number of trials
p <- 0.25  # probability of success for each trial

sample <- rbinom(n,t,p)  # store the sample
ggdat <- data.frame(prop.table(table(sample)))  # make a relative frequency table
ggdat$sample <- as.numeric(as.character(ggdat$sample)) # converting factor into numeric
                                                       # while preserving value
ggdat$prob = dbinom(ggdat$sample, t, p)  # calculate the true binomial probability
                                         # of each number of successes


pmf6 <- ggplot(ggdat) +
        geom_bar(aes(x=sample, y=Freq),     # plotting the histogram
                 stat = "identity",
                 color = "black",
                 fill = "grey") +
        geom_line(aes(x=sample, y=prob),    # plotting the true mass function
                  color = "red") +
        xlab("Number of Successes") +       # x axis label
        ylab("Density") +                   # y axis label
        ggtitle("Binomial Distribution",
                subtitle = paste("trials = ", n, ", probability = ", p)) + # adding title to p
        geom_hline(yintercept=0) +          # adding a line for the x-axis
        theme_bw()                          # removes grey background
```

```r
# Insert parameters:
n <- 100    # number of values to generate
t <- 20     # number of trials
p <- 0.25   # probability of success for each trial

sample <- rbinom(n,t,p)  # store the sample
ggdat <- data.frame(prop.table(table(sample)))  # make a relative frequency table
ggdat$sample <- as.numeric(as.character(ggdat$sample)) # converting factor into numeric
                                                       # while preserving value
ggdat$prob = dbinom(ggdat$sample, t, p)  # calculate the true binomial probability
                                         # of each number of successes


pmf7 <- ggplot(ggdat) +
        geom_bar(aes(x=sample, y=Freq),     # plotting the histogram
                 stat = "identity",
                 color = "black",
                 fill = "grey") +
        geom_line(aes(x=sample, y=prob),    # plotting the true mass function
                  color = "red") +
        xlab("Number of Successes") +       # x axis label
        ylab("Density") +                   # y axis label
        ggtitle("Binomial Distribution",
                subtitle = paste("trials = ", n, ", probability = ", p)) + # adding title to p
        geom_hline(yintercept=0) +          # adding a line for the x-axis
        theme_bw()                          # removes grey background

# Insert parameters:
n <- 1000    # number of values to generate
t <- 20      # number of trials
p <- 0.25    # probability of success for each trial

sample <- rbinom(n,t,p)  # store the sample
ggdat <- data.frame(prop.table(table(sample)))  # make a relative frequency table
ggdat$sample <- as.numeric(as.character(ggdat$sample)) # converting factor into numeric
                                                       # while preserving value
ggdat$prob = dbinom(ggdat$sample, t, p)  # calculate the true binomial probability
                                         # of each number of successes


pmf8 <- ggplot(ggdat) +
        geom_bar(aes(x=sample, y=Freq),     # plotting the histogram
                 stat = "identity",
                 color = "black",
                 fill = "grey") +
        geom_line(aes(x=sample, y=prob),    # plotting the true mass function
                  color = "red") +
        xlab("Number of Successes") +       # x axis label
        ylab("Density") +                   # y axis label
        ggtitle("Binomial Distribution",
                subtitle = paste("trials = ", n, ", probability = ", p)) + # adding title to p
        geom_hline(yintercept=0) +          # adding a line for the x-axis
        theme_bw()                          # removes grey background
```

```
(pmf5+pmf6)/(pmf7+pmf8)
```



In both cases, we see that, as we increase the sample size, our stimulations get closer and closer to the true mass function for the binomial distribution. □

4. Continue with the discrete distribution you selected for Question 3.

   (a) Provide the mean, standard deviation, skewness, and kurtosis of the PMF. Ensure to interpret each.

   Solution:

   The binomial distribution helps model discrete sets of data with a certain number of trials with a probability of success. The mean of this distribution is

$$\textbf{Mean} = np$$

   This makes sense, because the average number of successes is given by the above formula. The standard deviation of this distribution is

$$\textbf{SD} = np(1-p)$$

This is just multiplying the probability of failiure with the mean, which then gives the variance. The skewness is

$$\textbf{Skewness} = \frac{1-2p}{\sqrt{np(1-p)}}$$

For a small p and small n, the distribution will be skewed right since there are a low number of successes. For a large p and small n, the distribution is skewed left, since there are a high number of successes. For p = 0.5, the skewness is 0, since the distribution becomes symmetric. The kurtosis of the distribution is

$$\textbf{Kurtosis} = \frac{1-6p(1-p)}{np(1-p)}$$

The kurtosis tells us the rate of outliers in a given set of data.

(b) Generate a random sample of size $n = 10, 25, 100$, and 1000 for your two sets of parameter(s). Calculate the sample mean, standard deviation, skewness, and kurtosis. Interpret the results.
Solution:

```
library(e1071) #allows skewness and kurtosis
#The two sets of parameters are -
#40 coin tosses with the success being a heads
set.seed(345) #helps with testing again
t1 = 40
prob1 = 0.5
#20 rolls of a D4 with success being a 1 rolled
t2 = 20
prob2 = 0.25
#generating each set of data for each parameter
n.10.1 <- rbinom(10, t1, prob1)
n.10.2 <- rbinom(10, t2, prob2)
n.25.1 <- rbinom(25, t1, prob1)
n.25.2 <- rbinom(25, t2, prob2)
n.100.1 <- rbinom(100, t1, prob1)
n.100.2 <- rbinom(100, t2, prob2)
n.1000.1 <- rbinom(1000, t1, prob1)
n.1000.2 <- rbinom(1000, t2, prob2)
#making a dataframe of the samples
stat1 <- data.frame(n.10.1,
                    n.25.1,
                    n.100.1,
                    n.1000.1,
                    n.10.2,
                    n.25.2,
                    n.100.2,
                    n.1000.2)
#applying the R functions and listing the results
#Mean
apply(stat1, 2, mean)
```

```
##   n.10.1   n.25.1  n.100.1 n.1000.1   n.10.2   n.25.2  n.100.2 n.1000.2
##    19.60    20.52    19.84    19.88     5.90     4.28     4.98     5.01
```

```
#Standard Deviation
apply(stat1, 2, sd)
```

```
##   n.10.1   n.25.1  n.100.1 n.1000.1   n.10.2   n.25.2  n.100.2 n.1000.2
## 2.108185 3.603135 3.121561 3.203102 2.344247 1.662638 1.828372 1.886653
```

```
#Skewness
apply(stat1, 2, skewness)
```

```
##      n.10.1      n.25.1     n.100.1    n.1000.1      n.10.2      n.25.2
##  0.27663605 -0.23458002  0.22147552  0.08987464 -0.29900331  0.59578386
##     n.100.2    n.1000.2
##  0.02951240  0.19109951
```

```
#kurtosis
apply(stat1, 2, kurtosis)
```

```
##      n.10.1      n.25.1     n.100.1    n.1000.1      n.10.2      n.25.2
## -0.70875300 -0.49916581  0.01274940 -0.05466433 -1.40963737  0.94517654
##     n.100.2    n.1000.2
## -0.25228810 -0.04125743
```

The mean is usually close to the the actual mean regardless of number of samples. It would make sense for the standard deviation to increase with the sample size, since there are more values. The skewness and kurtosis both tend to decreases with a higher number of samples.

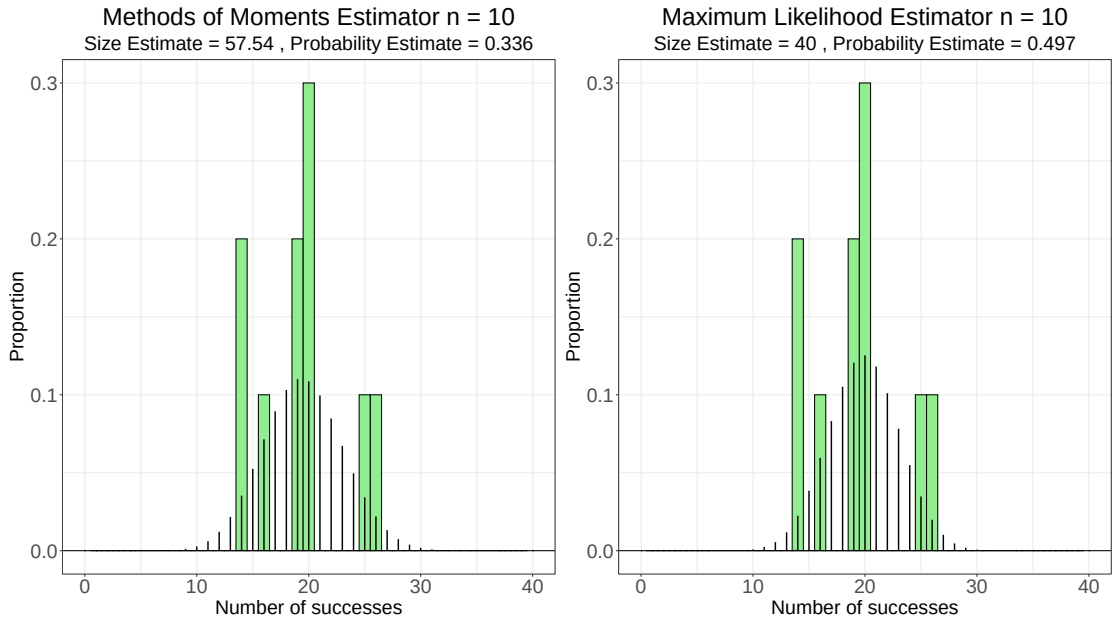
(c) Generate a random sample of size $n = 10$ for your two sets of parameter(s). Calculate the method of moments estimator(s) and maximum likelihood estimator(s). In a 1×2 grid, plot a histogram (with bin size 1) of each set of data with (1) the method of moments estimated distribution, (2) the maximum likelihood estimated distribution, and superimpose the true distribution in both.
Solution: The functions that are used for parts c through f are listed below. The moments for the binomial distribution were obtained via (Weisstein, 2002), and

```
set.seed(32423)
library(tidyverse)
library(patchwork)
binom.mom <- function(par, data){#calculates binomial MOM Estimator
  #adapted from Prof. Cipolli's Chapter 7 notes
  n <- par[1] #par[1] has the size n
  p <- par[2] # par[2] has the probability p
  #Here, k = 2 since estimator is highly variable at k>2
  #First two population moments
  EX1 <- n*p
  EX2 <- n*p*(1-p + n*p) #moments found in notes
  eq1 <- EX1 - mean(data) #sample moment 1
  eq2 <- EX2 - mean(data^2) #sample moment 2
  c(eq1, eq2)
}
# calculates Maximum Likelihood Estimator via negative log likelihood
binom.MLE <- function(par, data, neg = T){
  #adapted from Prof. Cipolli's Chapter 7 notes
  n <- par[1]
  p <- par[2]
  #sums up the probability mass function for the sample
  MLE <- sum(dbinom(x = data, size = n, prob = p, log=TRUE))
  ifelse(!neg,MLE,-MLE) #just in case neg is changed
}

library(nleqslv)# used to calculate MOM

find.binom.mom.mle <-function(n, par){
  #function that calculates and plots MOM and MLE
```

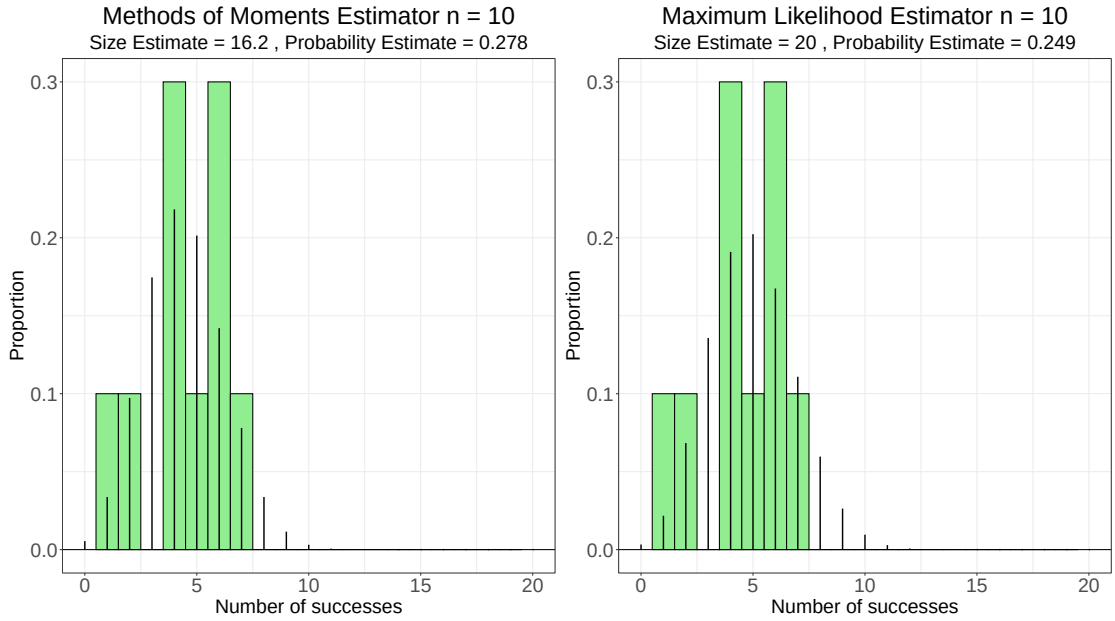
```
Sample = rbinom(n, size = par[1], prob =par[2])
#creating the sample for this set of parameters
moms<-nleqslv(x = c(par[1], par[2]), #passes in size and probability
fn = binom.mom,
data=Sample)
#this essentially minimizes the difference between the sample and population
#moments. It then returns the corresponding n and p
mles <- optim(fn = binom.MLE,
              par = c(par[1], par[2]),
              data = Sample)
#mles optimizes the MLE for the given parameters
#plotting MOM graph
  plot_mom = ggplot() +
    geom_histogram(aes(x = Sample,
                       y = ..density..),
                   binwidth = 1,
                   color = "black",
                   fill ="lightgreen") +
    geom_hline(yintercept = 0) +
    theme_bw() +
    xlim(0, par[1]) +
    geom_linerange(size = 0.7, aes(x=0:par[1],
                       ymin=0,
                       ymax=dbinom(0:par[1],
                                   size = round(moms$x[1]),
                                   prob = moms$x[2]))) +
    xlab("Number of successes") +
    ylab("Proportion") +
    ggtitle(paste("Methods of Moments Estimator n =", n, sep=" "),
            subtitle = paste("Size Estimate =", round(moms$x[1], 3),
                             ", Probability Estimate =", round(moms$x[2], 3),
                             sep =" ")) +
    theme(plot.title = element_text(hjust = 0.5, size = 25),
          plot.subtitle = element_text(hjust = 0.5, size = 20),
          axis.text=element_text(size=20),
          axis.title.x = element_text(size = 20),
          axis.title.y = element_text(size = 20))

#plotting MLE graph
  plot_mle = ggplot() +
    geom_histogram(aes(x = Sample,
                       y = ..density..),
                   binwidth = 1,
                   color = "black",
                   fill ="lightgreen") +
    geom_hline(yintercept = 0) +
    theme_bw() +
    xlim(0, par[1]) +
    geom_linerange(size = 0.7,aes(x=0:par[1],
                       ymin=0,
                       ymax=dbinom(0:par[1],
                                   size = round(mles$par[1]),
                                   prob = mles$par[2]))) +
```

```
    xlab("Number of successes") +
    ylab("Proportion") +
    ggtitle(paste("Maximum Likelihood Estimator n =", n, sep=" "),
            subtitle = paste("Size Estimate =", round(mles$par[1], 3),
                             ", Probability Estimate =", round(mles$par[2], 3),
                             sep =" ")) +
    theme(plot.title = element_text(hjust = 0.5, size = 25),
          plot.subtitle = element_text(hjust = 0.5, size = 20),
          axis.text=element_text(size=20),
          axis.title.x = element_text(size = 20),
          axis.title.y = element_text(size = 20))


      plot_mom+plot_mle
}
```

```
find.binom.mom.mle(10, c(t1, prob1))
```

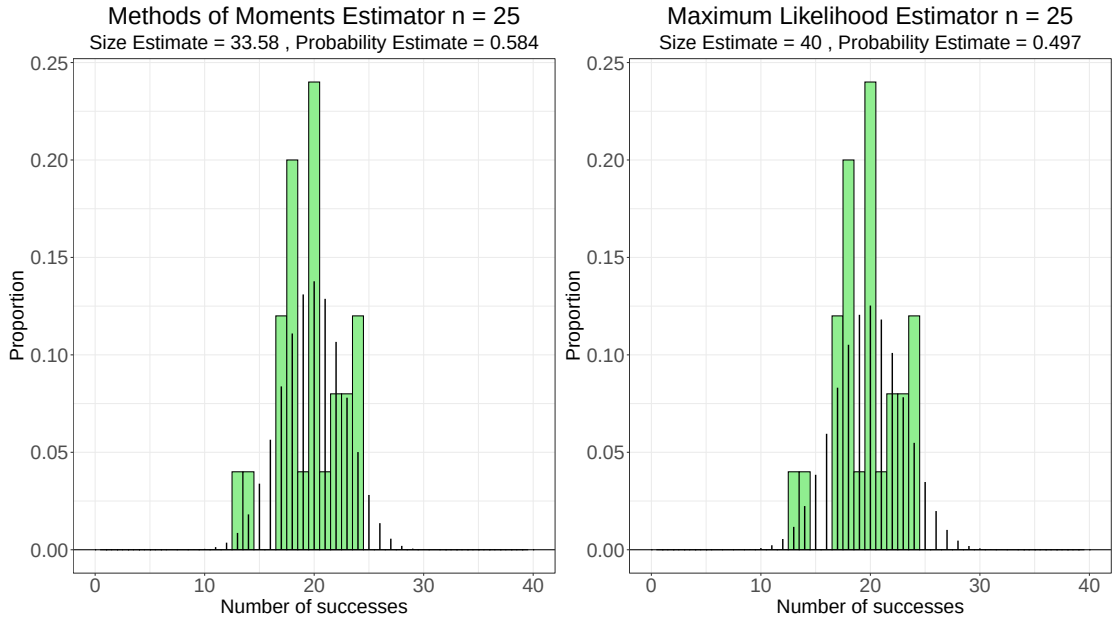



```
find.binom.mom.mle(10, c(t2, prob2))
```

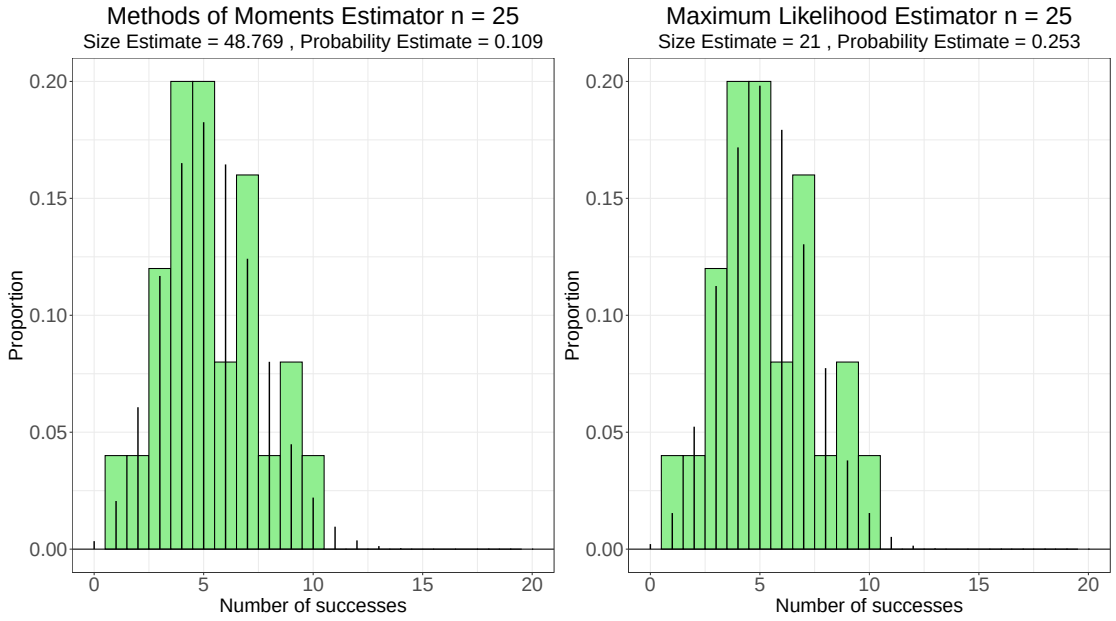


(d) Generate a random sample of size $n = 25$ for your two sets of parameter(s). Calculate the method of moments estimator(s) and maximum likelihood estimator(s). In a 1×2 grid, plot a histogram (with bin size 1) of each set of data with (1) the method of moments estimated distribution, (2) the maximum likelihood estimated distribution, and superimpose the true distribution in both.
Solution:

```
find.binom.mom.mle(25, c(t1, prob1))
```

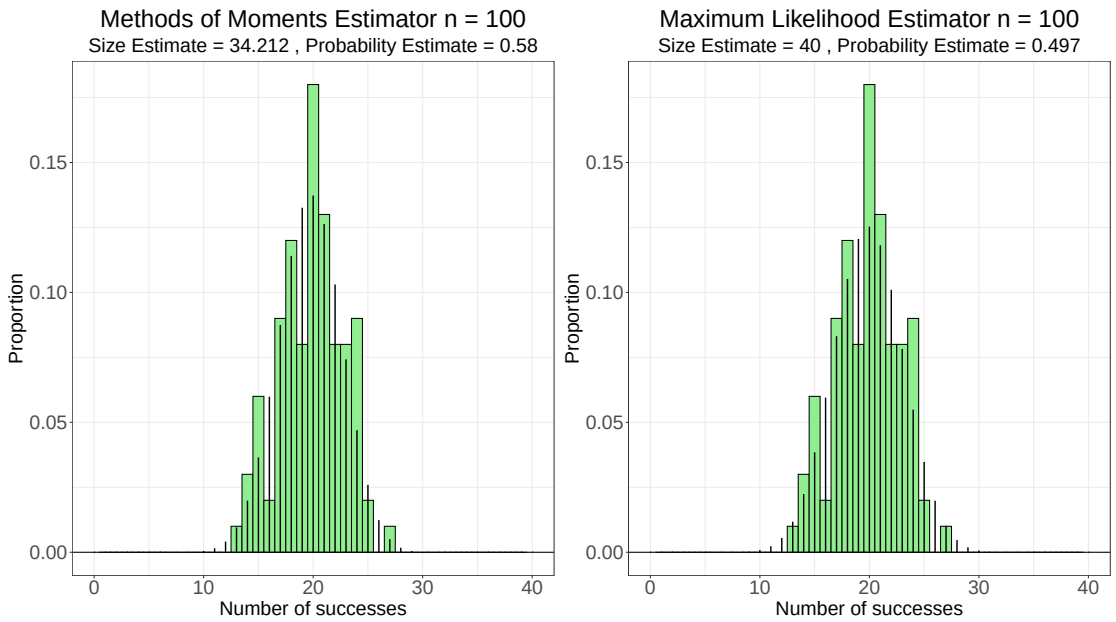


```
find.binom.mom.mle(25, c(t2, prob2))
```

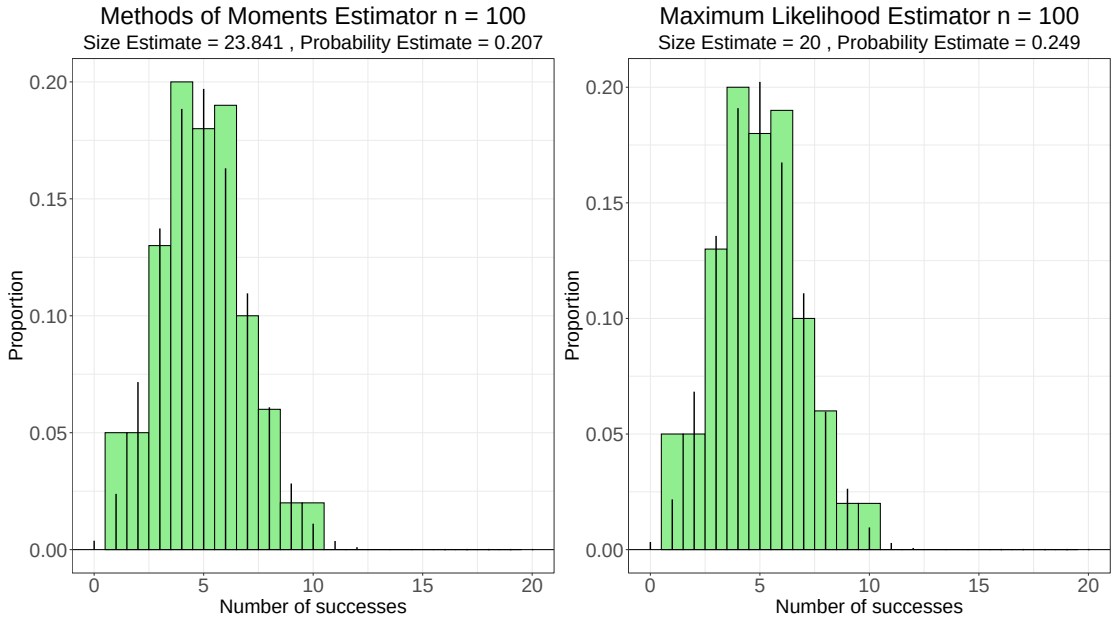


(e) Generate a random sample of size $n = 100$ for your two sets of parameter(s). Calculate the method of moments estimator(s) and maximum likelihood estimator(s). In a 1×2 grid, plot a histogram (with bin size 1) of each set of data with (1) the method of moments estimated distribution, (2) the maximum likelihood estimated distribution, and superimpose the true distribution in both.
Solution:

```
find.binom.mom.mle(100, c(t1, prob1))
```

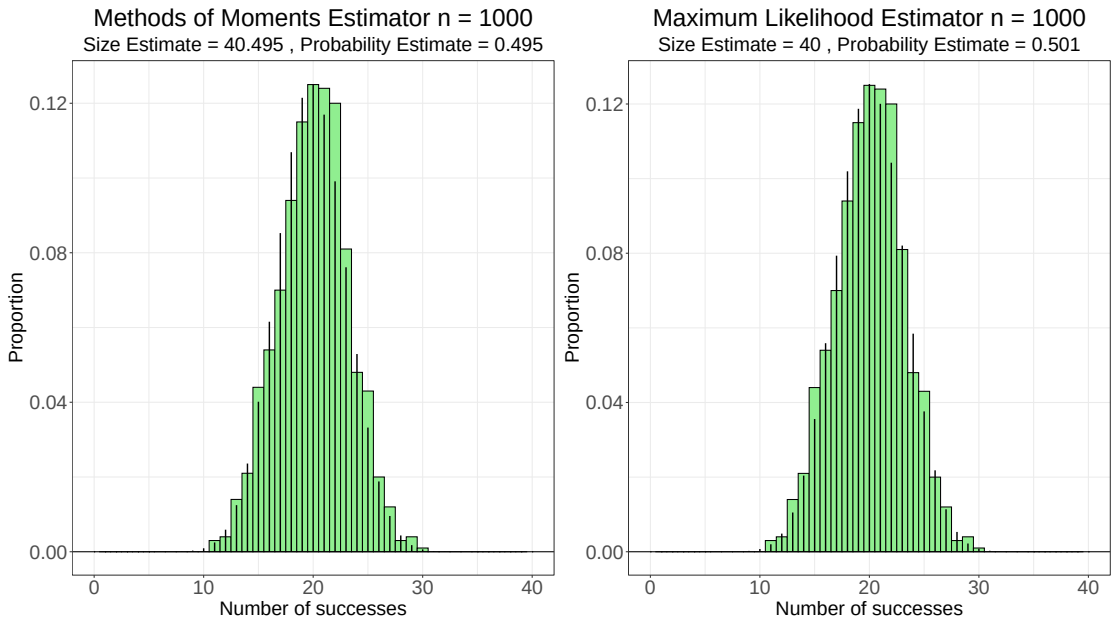



```
find.binom.mom.mle(100, c(t2, prob2))
```

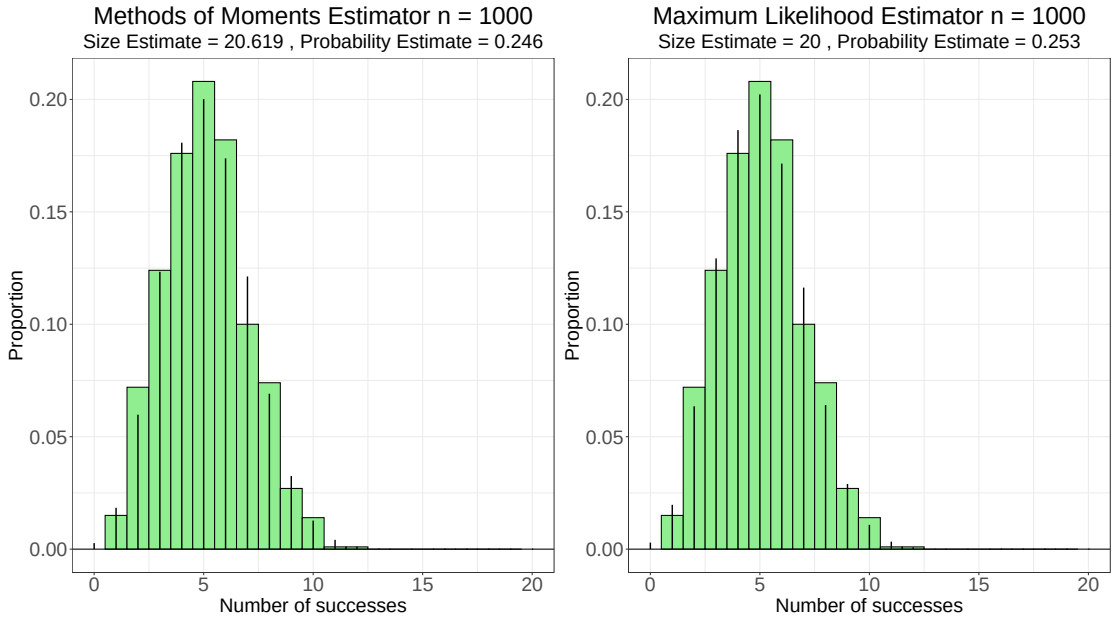


(f) Generate a random sample of size $n = 100$ for your two sets of parameter(s). Calculate the method of moments estimator(s) and maximum likelihood estimator(s). In a 1×2 grid, plot a histogram (with bin size 1) of each set of data with (1) the method of moments estimated distribution, (2) the maximum likelihood estimated distribution, and superimpose the true distribution in both.
Solution:

```
find.binom.mom.mle(1000, c(t1, prob1))
```



```
find.binom.mom.mle(1000, c(t2, prob2))
```



(g) Comment on the results of parts (c)-(f).
Solution: As seen in the plots above, the size estimate for the MOM Estimator is very inaccurate for n = 10 and n = 25, but gets increasingly closer as n increases. This corresponds with the Weak Law of Large Numbers. This pattern is also seen for the MOM Estimator of the probability estimate. The MLE estimator is a different since the size estimate provided by it is almost always completely accurate, and the probability estimate is very close even for n = 10. To find out which Estimator is better, we would need to run this for a larger array of samples, since it is completely possible that the MLE is not as good at providing estimate for some other parameters. Overall, these estimates look very close to the data generates, especially at higher n values.

References

Analytica Wiki (2021). Gamma distribution.

Belikov, Aleksey V (2017). The number of key carcinogenic events can be predicted from cancer incidence. *Scientific reports*, 7(1):1–8.

Boland, Philip J (2007). *Statistical and probabilistic methods in actuarial science*. CRC Press.

Freedman, D. and Diaconis, P. (1981). On the histogram as a density estimator:l2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 57:453–476.

Gamma Function Wikipedia (2021). Gamma function.

Hasselman, B. (2018). *nleqslv: Solve Systems of Nonlinear Equations*. R package version 3.3.2.

Incomplete gamma function Wikipedia (2021). Incomplete gamma function.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2021). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-8.

Pedersen, T. L. (2020). *patchwork: The Composer of Plots*. R package version 1.1.1.

Philippou, A. N. and Antzoulakos, D. L. (2011). Binomial distribution. *International Encyclopedia of Statistical Science*, 1:152–154.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Thom, Herbert CS (1958). A note on the gamma distribution. *Monthly weather review*, 86(4):117–122.

Weisstein, E. W. (2002). Binomial distribution. *https://mathworld. wolfram. com/*.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wikipedia (2021). Binomial distribution — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Binomial%20distribution&oldid=1048287102. [Online; accessed 10-October-2021].

You, X. (2017). Approximation of the median of the gamma distribution. *Journal of Number Theory*, 174:487–493.