R Lab #6b - Binomial Test

Contents

Binomial test: flower example

Chapter 7 covers how to analyze the data set on "right-handed" and "left-handed" flowers using the binomial test.

In the data set the sample size was 27, the number of left-handed flowers was 6, and the expected proportion of left-handed flowers was 27*0.25=6.75.

As covered in the book and lecture, you can use the binomial distribution to calculate the probability of getting 0, 1, 2, 3, etc. left-handed flowers in a sample of 27 given a null proportion of 0.25. That is, calculate the sampling distribution given a null proportion.

Using dbinom to calculate the sampling distribution

We can use a new command in R to calculate the probabilities shown in Table 7.1-1.

First, you will need to install a new module called "binom" using install.packages(). That is install.packages("binom",dependencies=T). Then load binom with the command library():

```
library(binom)
```

Now we can start with calculating the probability of getting *exactly* 0 left-handed flowers in a sample of 27 flowers and a null proportion of 0.25. This is done with the command dbinom(), which takes three arguments: 1) number of "successes", 2) number of observations, 3) probability of a success.

```
dbinom(0, 27, 0.25)
## [1] 0.0004233057
```

We could use this command to get the proability of each possible value of number of successes. . .

```
dbinom(1, 27, 0.25)
## [1] 0.003809751
dbinom(2, 27, 0.25)
## [1] 0.01650892
dbinom(3, 27, 0.25)
## [1] 0.04585812
```

Etc., but that would be tedious. If you really wanted to replicate Table 7.1-1 then you could use a vector of possible success values to make it more efficient.

First, make a vectoer with 0, 1, 2, . . . 27.

```
xsuccesses <- c(0:27)
xsuccesses
##  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
## [26] 25 26 27
```

Now you can run dbinom() using this vector as input for the number of successes and it will create a vector of respective probabilities.

```
probs <- dbinom(xsuccesses, 27, 0.25)
probs
##  [1] 4.233057e-04 3.809751e-03 1.650892e-02 4.585812e-02 9.171623e-02
##  [6] 1.406316e-01 1.718830e-01 1.718830e-01 1.432358e-01 1.007956e-01
## [11] 6.047736e-02 3.115500e-02 1.384667e-02 5.325641e-03 1.775214e-03
## [16] 5.128395e-04 1.282099e-04 2.765311e-05 5.120947e-06 8.085705e-07
## [21] 1.078094e-07 1.197882e-08 1.088984e-09 7.891188e-11 4.383993e-12
## [26] 1.753597e-13 4.496403e-15 5.551115e-17
```

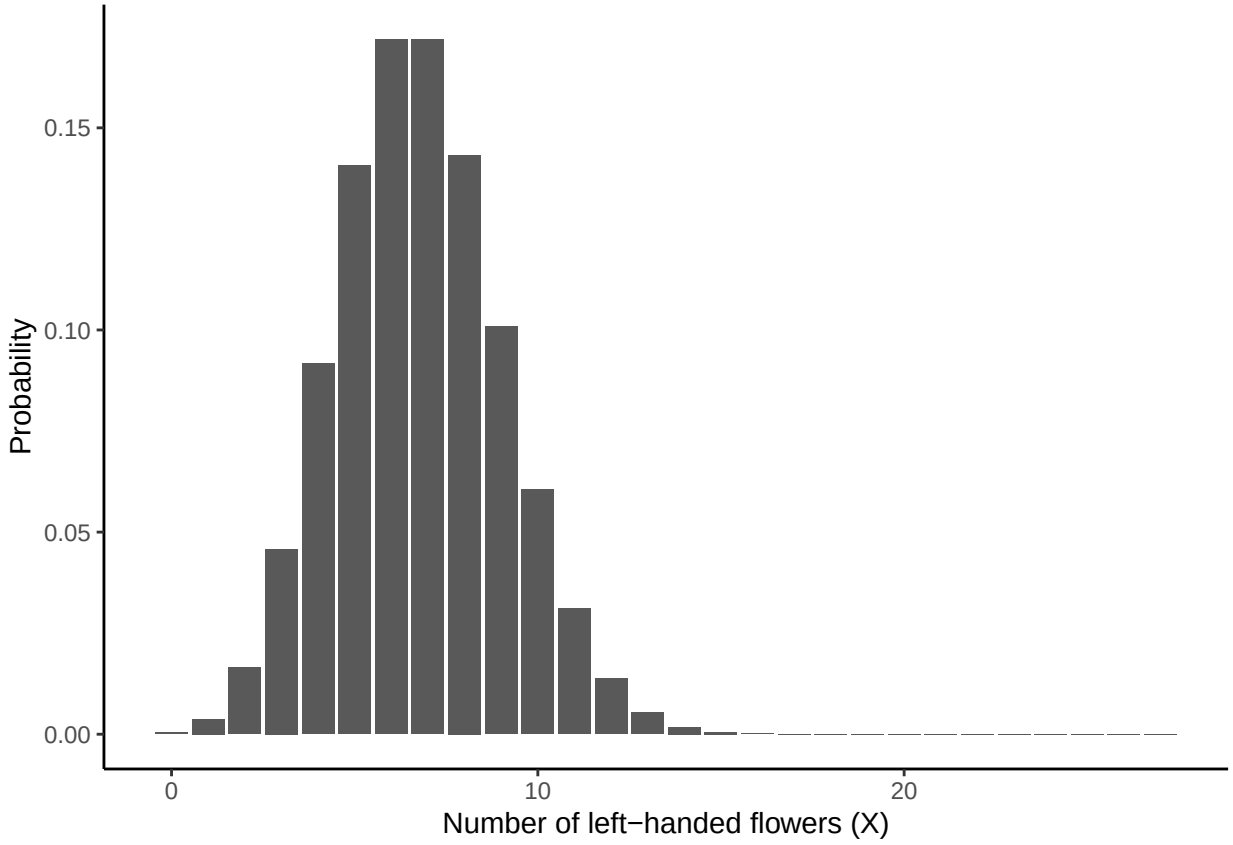
If you wanted to put these two vectors together to make a table then you can use the data.frame() command to join them:

```
probTable <- data.frame(xsuccesses, probs)
probTable
##    xsuccesses        probs
## 1           0 4.233057e-04
## 2           1 3.809751e-03
## 3           2 1.650892e-02
## 4           3 4.585812e-02
## 5           4 9.171623e-02
## 6           5 1.406316e-01
## 7           6 1.718830e-01
## 8           7 1.718830e-01
## 9           8 1.432358e-01
## 10          9 1.007956e-01
## 11         10 6.047736e-02
## 12         11 3.115500e-02
## 13         12 1.384667e-02
## 14         13 5.325641e-03
## 15         14 1.775214e-03
## 16         15 5.128395e-04
## 17         16 1.282099e-04
## 18         17 2.765311e-05
## 19         18 5.120947e-06
## 20         19 8.085705e-07
## 21         20 1.078094e-07
## 22         21 1.197882e-08
## 23         22 1.088984e-09
## 24         23 7.891188e-11
## 25         24 4.383993e-12
## 26         25 1.753597e-13
## 27         26 4.496403e-15
## 28         27 5.551115e-17
```

Then you could graph the sampling distribution to mimic Figure 7.1-1. Since the x-axis variable is not continuous it will look best if we use geom_bar(). Also, instead of the y-axis being "count" (as done previously), we want it to be the vector probTable$probs. We add this into the aes(), but also need to add

stat="identity" to the geom_bar() command. This tells R that you want to use your own values for y.

```
library(ggplot2)
ggplot(probTable, aes(x = xsuccesses, y = probs)) + geom_bar(stat = "identity") +
    xlab("Number of left-handed flowers (X)") + ylab("Probability") +
    theme_classic()
```



Using binom.test() to calcualte the P-value

We can use the command binom.test() to calculate the P-value for our observed number of left-handed flowers. That is, the probability of getting data at least as extreme as the observed data (6/27) given that the null hypothesis (p=0.25) is true.

binom.test() takes the same three arguments as dbinom(): 1) number of successes, 2) number of observations, 3) null proportion.

```
binom.test(6, 27, 0.25)
##
##  Exact binomial test
##
## data:  6 and 27
## number of successes = 6, number of trials = 27, p-value = 1
## alternative hypothesis: true probability of success is not equal to 0.25
## 95 percent confidence interval:
##  0.08621694 0.42258306
## sample estimates:
## probability of success
##              0.2222222
```

Notice that the output has a P-value of 1. Since this is above alpha of 0.05 we fail to reject the null hypothesis that the population proportion of left-handed flowers is 0.25.

Using binom.confint() to calcualte the Agresti-Coull 95% CI

Also notice that the output reports a 95% confidence interval. This is for the population proportion, however, it is **not** calculated with the recommended Agresti-Coull method.

This can be estimated with a different command: binom.confint(). For this command you provide the number of successes, sample size, and method for calculating the 95% confidence interval ("ac"):

```
binom.confint(6, 27, method = "ac")
##          method x  n      mean     lower    upper
## 1 agresti-coull 6 27 0.2222222 0.1026357 0.411006
```