Chapter 2

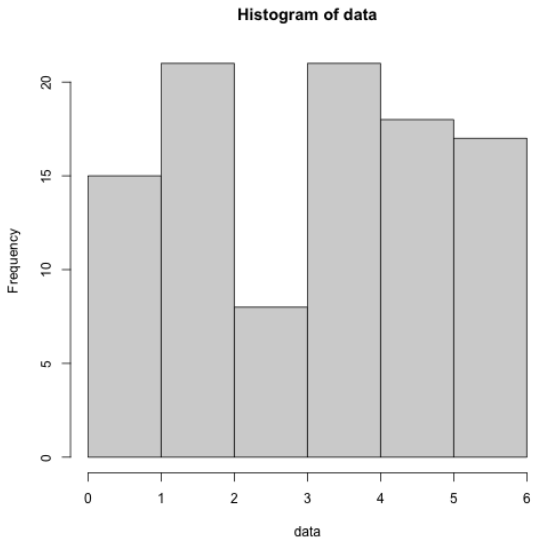
Example 1: Let's simulate a fair six-sided die roll. Recall that we need to use the **set.seed** function so that we all generate the same random data. For consistency, I always set the seed to 2020, i.e., set.seed(2020). The sample function will generate 100 random die rolls with each possible outcome (1,2,3,4,5,6) being equally likely.

```
set.seed(2020)
data = sample(x = 1:6,size = 100,replace = T); data
```

```
##   [1] 4 4 6 1 1 4 2 6 1 5 2 2 6 5 2 3 2 5 4 2 6 6 4 6 4 2 4 5 4 4 3 6
##  [33] 2 2 6 3 5 4 5 5 2 5 1 6 3 5 1 5 3 1 5 3 2 6 2 1 3 2 1 6 5 5 2 5
##  [65] 6 4 3 6 4 4 2 2 1 6 1 4 6 2 2 2 1 4 1 1 4 2 2 5 5 4 4 4 6 5 4 1
##  [97] 6 5 1 4
```

Let's make a histogram of the simulated data. Recall that each outcome (1,2,3,4,5,6) is equally likely in theory but in our simulation, some numbers may come up more than others. To make the histogram look better, I define the boundary breaks to be 0:6 (i.e., 0,1,2,3,4,5,6).

```
hist(data,breaks=0:6)
```



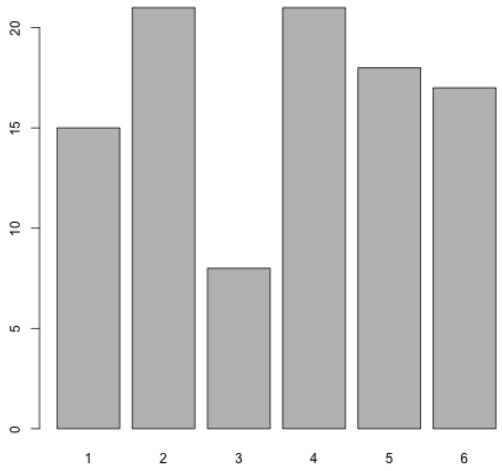
With only 100 random die rolls, the distribution isn't as flat, uniform, and symmetric as we would expect. The number 3 came up less than we would anticipate. This is OK.

We can also generate a frequency table and use this to create a barplot. This looks a little nicer than the standard histogram and doesn't require use to use the **breaks** argument.

```
table(data)

## data
##  1  2  3  4  5  6
## 15 21  8 21 18 17

barplot(table(data))
```

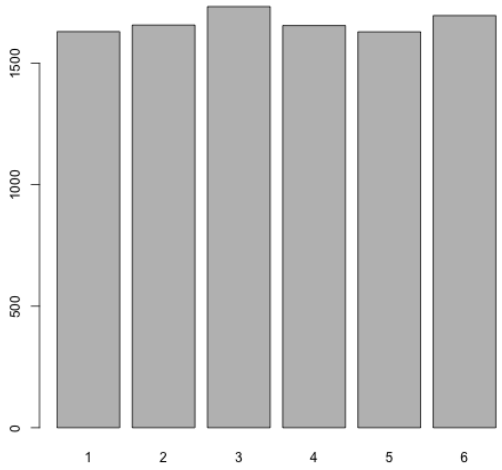


If we want a better representation of the true distribution of the fair six-sided die rolls, we could simulate 10000 die rolls instead of 100.

```
set.seed(2020)
data = sample(x = 1:6,size = 10000,replace = T)
table(data)

## data
##    1    2    3    4    5    6
## 1630 1657 1733 1655 1629 1696
```

```r
barplot(table(data))
```



Simulation can be helpful for us to verify hand calculations and to estimate quantities that are hard to calculate by hand. Our simulation results won't exactly match our hand calculations, but if the sample size is large enough, they should be close.

For instance, we know that P(Roll=1) = 1/6 for the population of all fair six-sided die rolls. If you refer to the table at the bottom of the previous page, you see that 1 came up 1630 times out of 10000 die rolls. In the simulation, P(1) = 1630/10000 = 0.1630 which is pretty close the the theoretical value of P(1) = 1/6 = 0.1667.

Here is an example of how to calculate the probability of a die roll less than 3 in our simulation.

```r
# Calculate P(Roll < 3)
# From theory, we know that P(Roll < 3) = P(1) + P(2) = 1/6 + 1/6 = 1/3

# count up the number of rolls less than 3
num.under3 = sum( data < 3 )

# divide the number of rolls less than 3 by the number of rolls
prob.under3 = num.under3 / 10000
prob.under3

## [1] 0.3287
```

From theory, we expect P(Roll < 3) = 1/3 = 0.3333. In our simulation, we found P(Roll < 3) = 0.3287.

Example 2: Let's simulate rolling two fair six-sided dice 10,000 times and calculate some probabilities. **d1** will represent the first die roll and **d2** will represent the second die roll.

```
set.seed(2020)
d1 = sample(x = 1:6,size = 10000,replace = T)
d2 = sample(x = 1:6,size = 10000,replace = T)
```

From our simulated data, let's calculate P(D1 = 1 and D2 = 2), that is, the probability that first die roll is 1 and the second die roll is 2. From theory, this probability should be 1/36 = 0.0278. (We use the **&** for **AND** in R.)

```
num.events = sum(d1 == 1 & d2 == 2)
prob = num.events / 10000; prob

## [1] 0.0256
```

Next, let's calculate P(D1 < 2 or D2 > 2), that is, the probability that the first die roll is less than 2 or the second die roll is greater than 2. From theory, this probability should be 0.7222. We use the | for the **OR** in R.)

```
num.events = sum(d1 < 2 | d2 > 2)
prob = num.events / 10000; prob

## [1] 0.7183
```

Finally, let's look at the frequency table and distribution of the sum of the two die rolls.

```
table(d1+d2)

##
##    2    3    4    5    6    7    8    9   10   11   12
##  271  540  824 1154 1373 1703 1360 1149  814  525  287

barplot(table(d1+d2))
```