Hands-on introduction to R

Author	**A Pardhanani**
Date	**2021-10-05T23:14:10**
Project	**f589b585-2079-40e9-a404-1ea0aa8fb62b**
Location	**introduction_to_r_ds.sagews**
Original file	**introduction_to_r_ds.sagews**

Hands-on introduction to R

R is a powerful, comprehensive, open-source software framework for doing a variety of modern computational tasks including those needed in statistics and data science. It is possible to download and install the software on compute to use it through a website-interface without downloading anything.

We will start by using R through a website-interface in the form of Sage worksheets. In fact, what you are reading he a Sage worsheet that will guide you through the first steps of getting familiar with R.

Let us begin by learning how to input (small) datasets into R.

How to input simple datasets by hand

Prelude: To use R through Sage worksheets (and Sage cells), the first line in each new cell must always be "%r" (wi the quotes).

Another alternative is to choose "R" from the modes menu at the top of the worksheet.

R allows setting up your data through keyboard input, or by reading the data through an input file. It is extremely use know how to do keyboard input for simple and small datasets.

Example: Find the mean, standard deviation and 5-number summary for the set of values: 1, 2, 3, 4, 8

The commands below show how to do this.
Note that all the information following any "#" sign is just to explain what is going on. R ignores anything that follows sign.

```
%r
# Example showing calculations done in the simplest way for
# a dataset consisting of the 5 numbers: 1, 2, 3, 4, 8
a = c(1, 2, 3, 4, 8)  # define your dataset and give it some name, say, a
mean(a)               # find its mean
sd(a)                 # find its standard deviation
var(a)                # find its variance
summary(a)            # find its 5-number summary & mean
```

3.6

2.70185121722126

7.3

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0     2.0     3.0     3.6     4.0     8.0
```

Now, let's define a 2nd variable that is categorical.

For example, suppose it contains the 5 values: Yes, No, Yes, Yes, Maybe

```
%r
# A dataset consisting of the 5 values: Yes, No, Yes, Yes, Maybe
acat = c("Yes", "No", "Yes", "Yes", "Maybe")
         # notice that you must use quotes to enclose categorical values
table(acat)           # make a frequency table

```

```
acat
Maybe    No   Yes
    1     1     3
```

Exercise 1: Create an R variable for each of the following datasets

a. P={blue, pink, blue, green, green, blue, pink, blue}
b. Q={3.9, 0, -4.6, -3.3, 2.2, 3.6, -2.9, -0.4, 0.9, 1.5}
c. R={0, 1, 2, 3, a, b, c}

Compute summary stats for each quantitative variable, and make a frequency table for each categorical variable.

A very useful thing to know about R is how to access the builtin help utility that is available for every function: simply the "?" symbol, followed by the command-name or function for which you want help.

For example: `?table`
`?var`
`?sd`

How to input a spreadsheet of data

A spreadsheet or table of raw data can be created manually via keyboard input, or by reading in data files written in various standard formats. The structure used in R to represent such tables is called a "dataframe."

Consider, for example, the following dataset

Age	Sex	Class year	SAT score	Financial aid?
18	F	1	1014	N
20	F	3	1222	Y

17	M	1	1141	Y
17	F	1	1082	N
19	M	2	1261	Y
18	F	2	1288	N
20	F	1	1002	N
21	M	3	1078	N

We will now learn how to create a dataframe like this. The first step is to input each column of data as a separate variable. After that we will organize the variables into a dataframe. The dataframe can be given any convenient nam e.g., "mydata"

```
%r
# First create each column as a separate variable: I'll use the names
# "age", "sex", etc., for the names of my variables
age = c(18, 20, 17, 17, 19, 18, 20, 21)
sex = c("f", "f", "m", "f", "m", "f", "f", "m")
year = c(1, 3, 1, 1, 2, 2, 1, 3)
sat_score = c(1014, 1222, 1141, 1082, 1261, 1288, 1002, 1078)
f_aid = c("n", "y", "y", "n", "y", "n", "n", "n")

# Next, I'll combine the variables into a dataframe that
# I will call "mydata"
mydata = data.frame(age, sex, year, sat_score, f_aid)

# Let's print out the dataframe and see if it is what I expect
mydata

# Now we can compute summary stats, make histograms, boxplots,
# piecharts, etc.
```

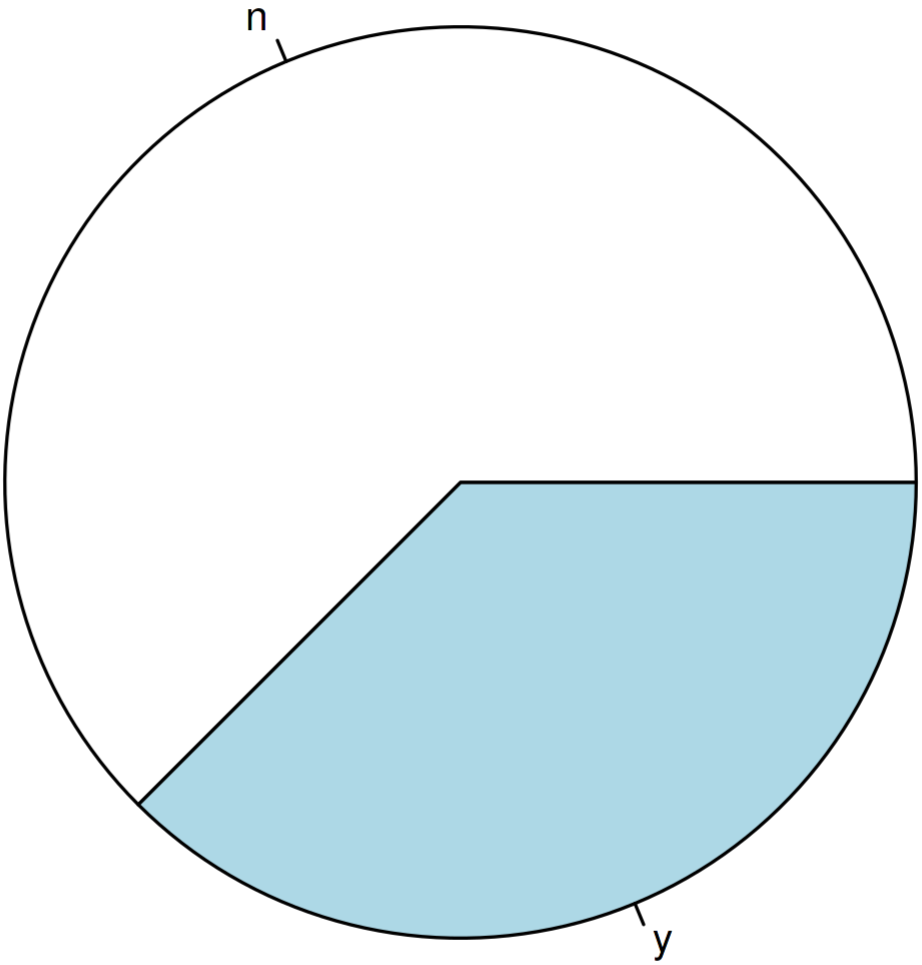
```
  age sex year sat_score f_aid
1 18  f   1    1014      n
2 20  f   3    1222      y
3 17  m   1    1141      y
4 17  f   1    1082      n
5 19  m   2    1261      y
6 18  f   2    1288      n
7 20  f   1    1002      n
8 21  m   3    1078      n
```

Once a dataframe is created, it is easy to make various displays, and to compute summary statistics for variables in dataframe. The following examples show how to do this for variables in the dataframe created above.

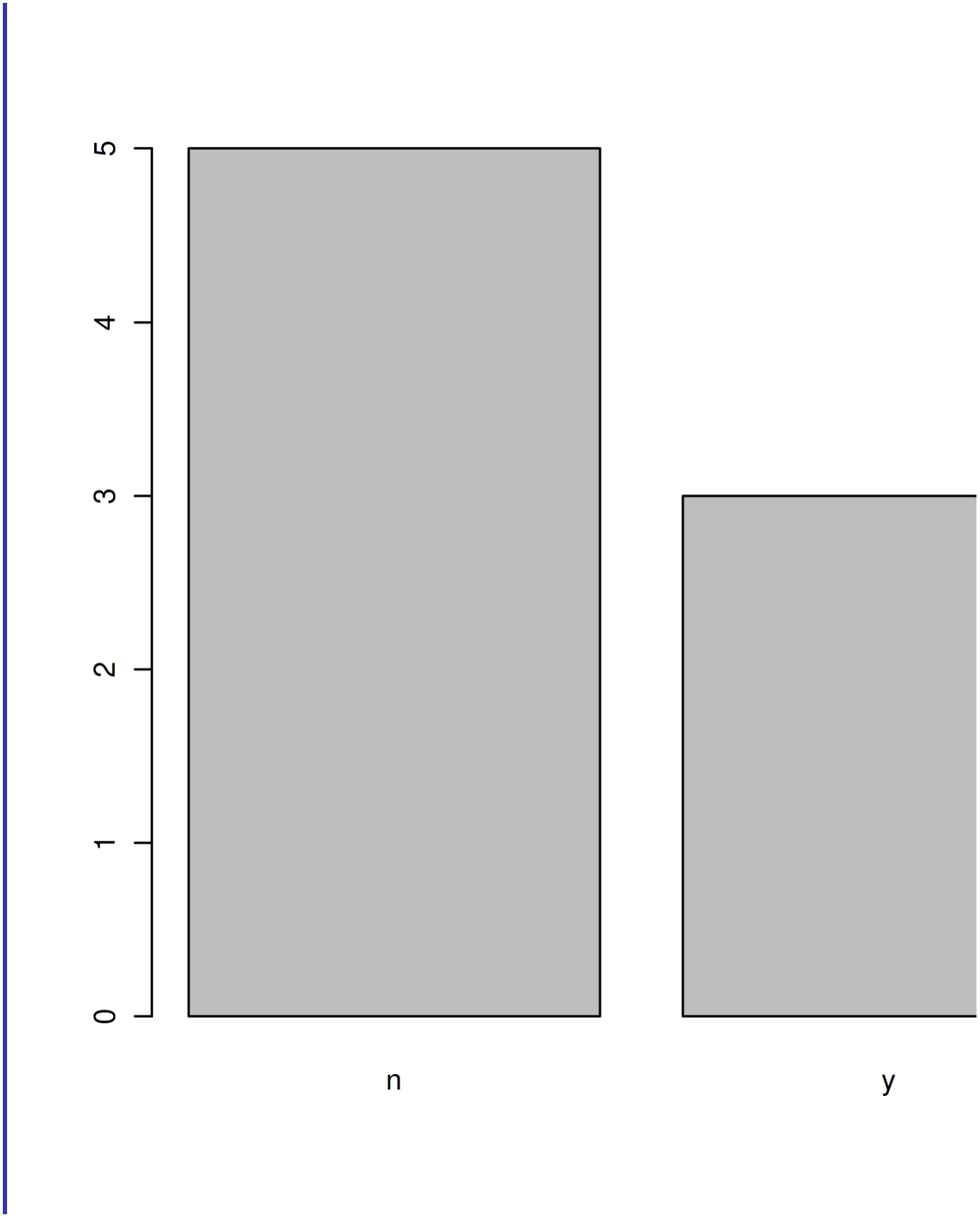
```
%r
table(mydata$f_aid)             # creates frequency table using "f_aid" from "mydata
```

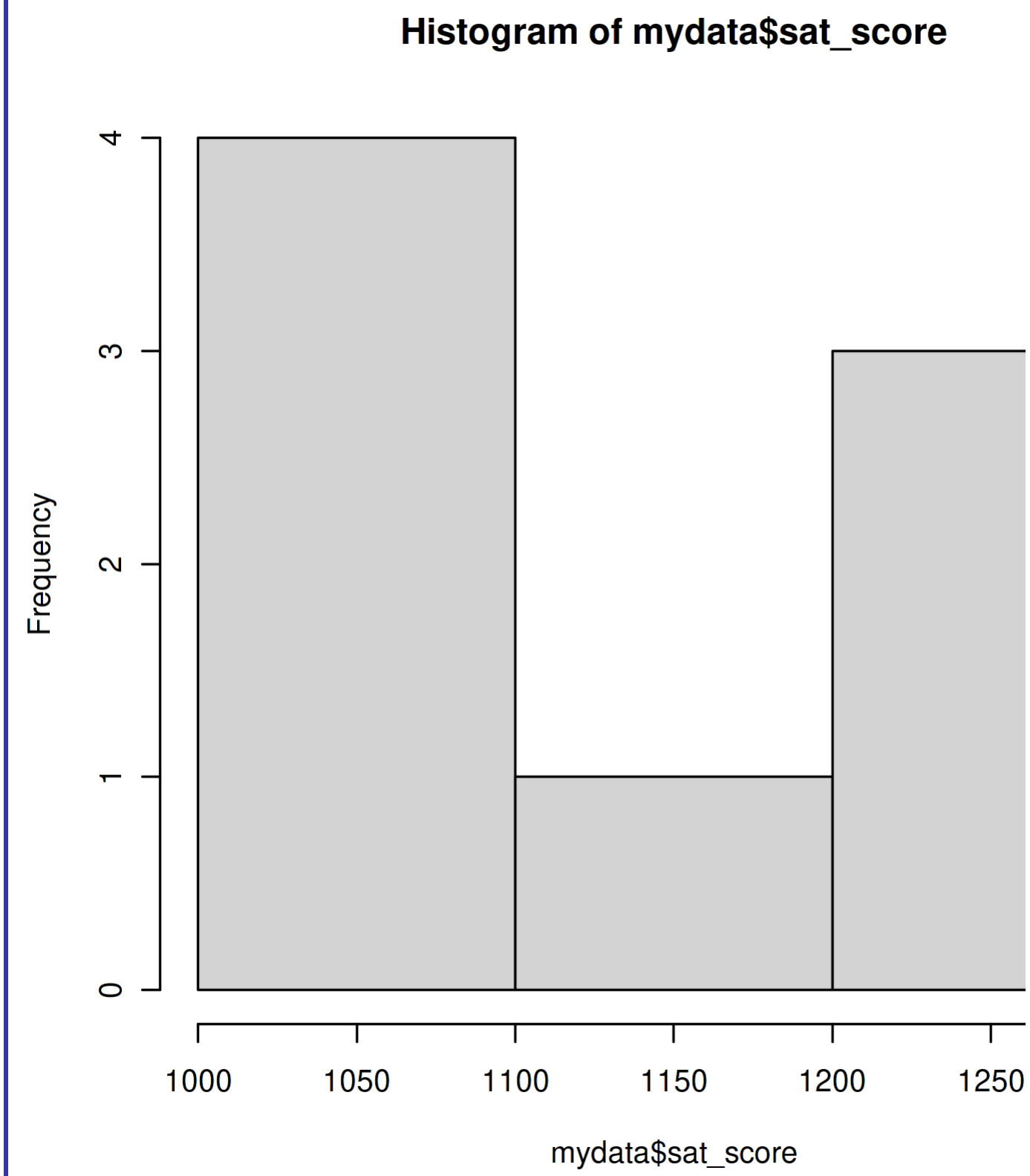
```
pie(table(mydata$f_aid))         # pie chart of "f_aid"
barplot(table(mydata$f_aid))     # bar graph of "f_aid"
summary(mydata$sat_score)        # 5-number summary & mean of "sat_score"
hist(mydata$sat_score)           # plot histogram
boxplot(mydata$sat_score)

# Note that mydata$f_aid, mydata$sat_score, etc., is one way to
# access a specific variable in the dataframe "mydata". R also offers
# other ways to access these same variables, and you may run into them
# when you look at R code from other sources.

# A simple way to set the histogram scale is to specify
# the number of bars to use, like this
hist(mydata$sat_score, breaks=6)   # histogram with 6 equal-width bars

# It is also easy to customize the plot title, axes labels, etc., like this
hist(mydata$sat_score, breaks=6, xlab="SAT scores", main="A title test")

# Try the "?hist" command to see more features of R's histogram function.
```

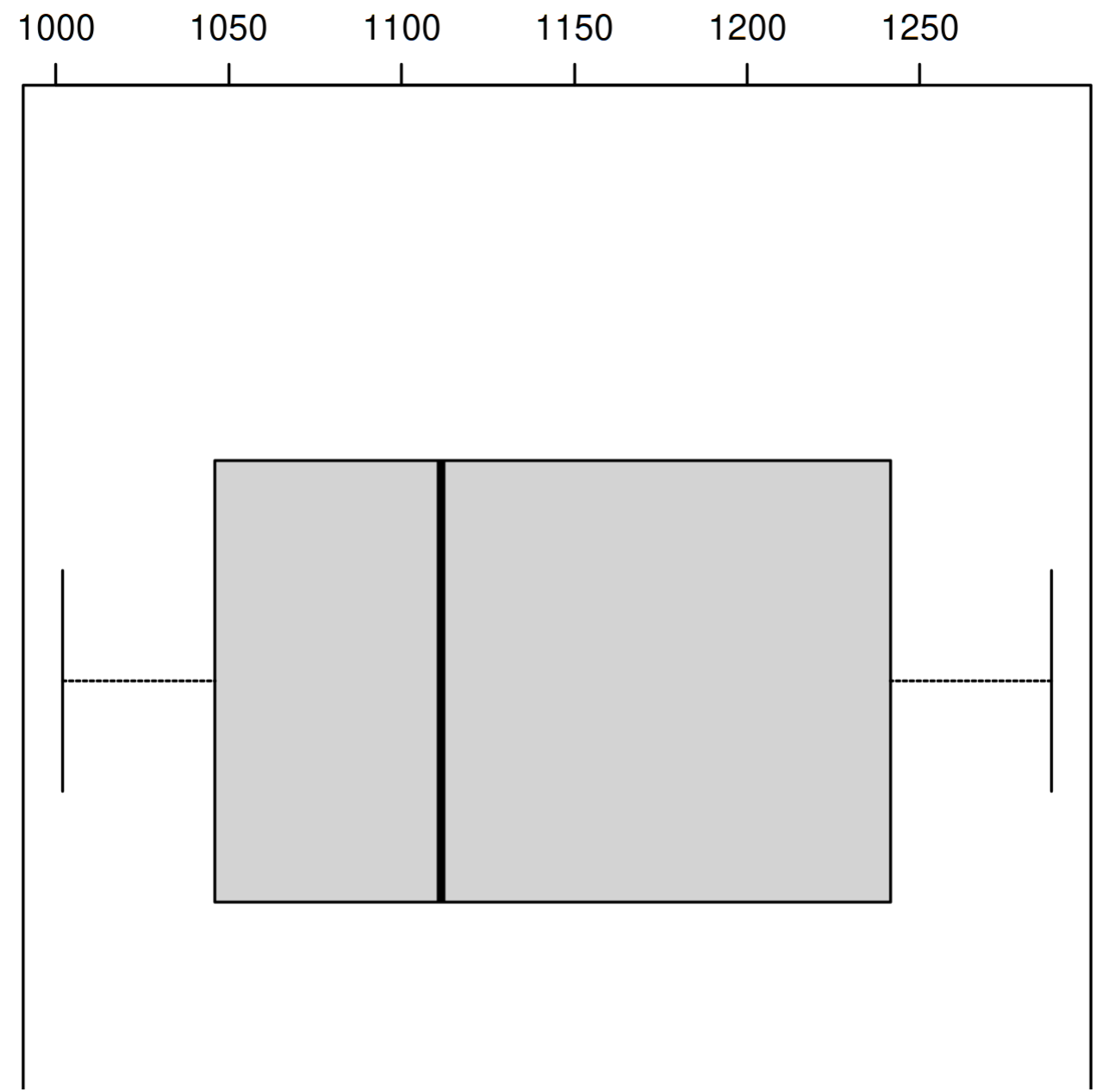
```
n y
5 3
```

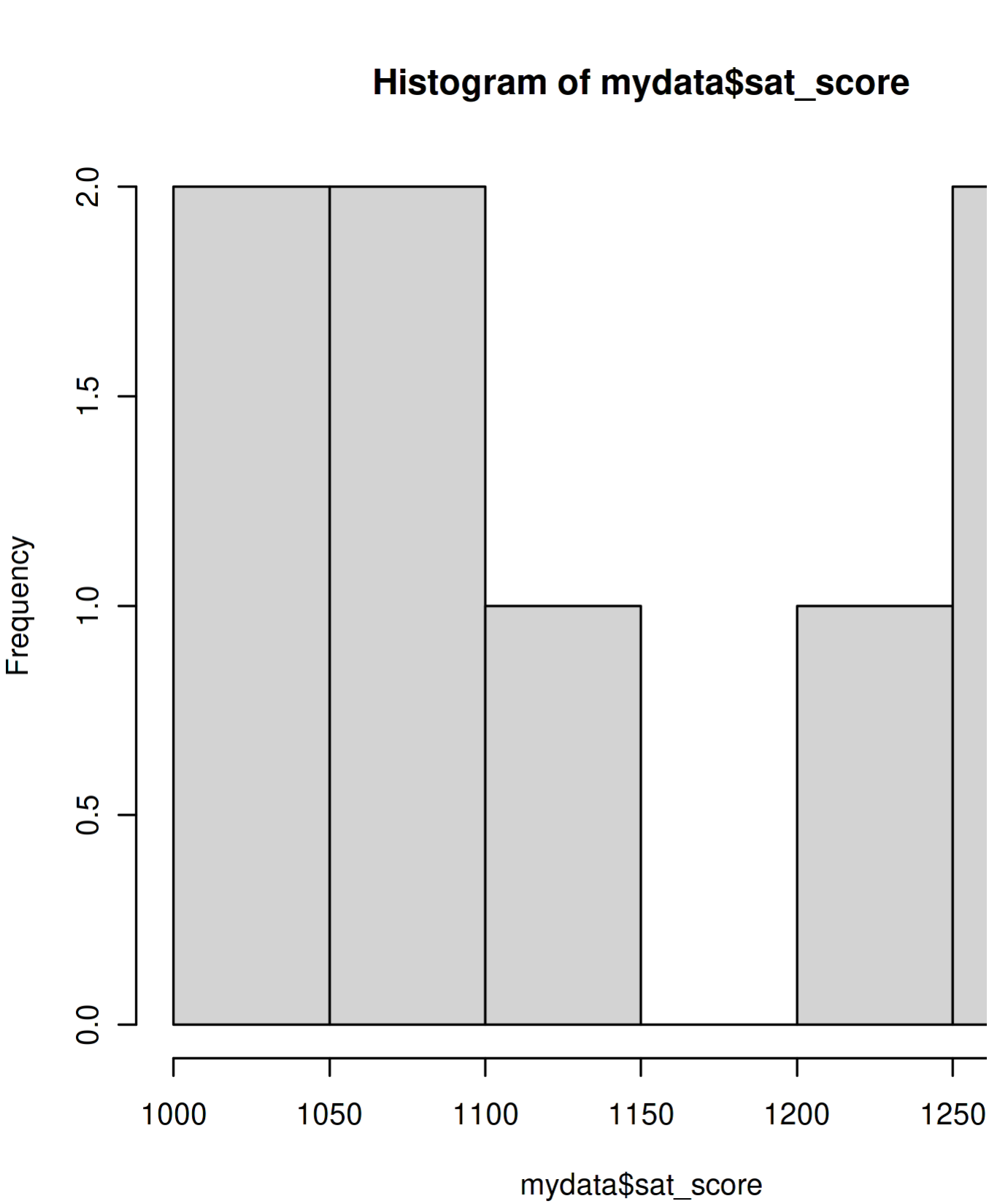


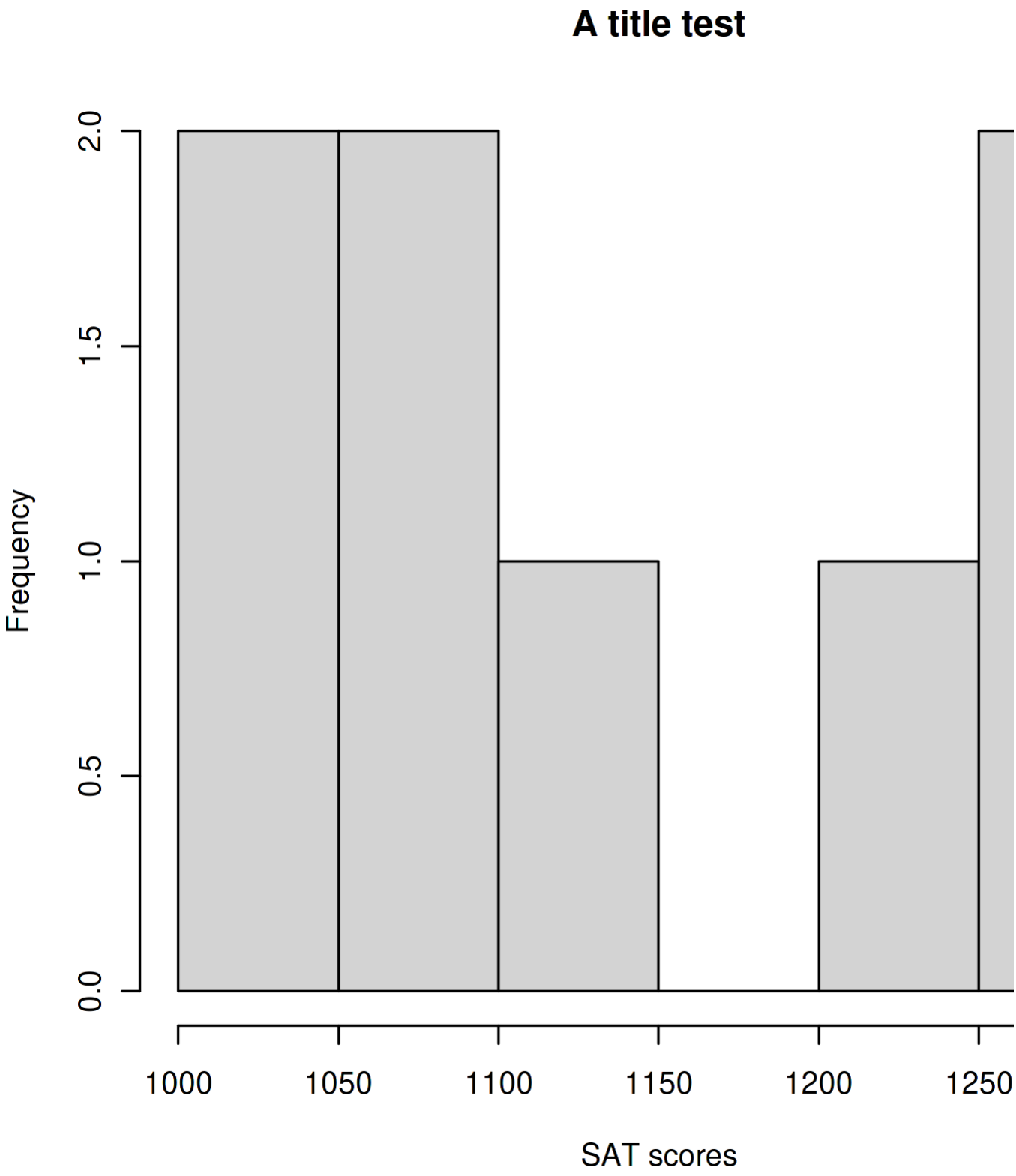

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1002    1062    1112    1136    1232    1288
```


0
1
2
3
4
5
n
y


Histogram of mydata$sat_score
Frequency
0
1
2
3
4
1000
1050
1100
1150
1200
1250
mydata$sat_score


1000
1050
1100
1150
1200
1250


Histogram of mydata$sat_score
Frequency
0.0
0.5
1.0
1.5
2.0
1000
1050
1100
1150
1200
1250
mydata$sat_score



Exercise 2.1: The following table contains data on the employment status of a sample of college students

Age	Major	Employment	Work hours

19	Business	Part time	35
19	English	Part time	30
34	Business	Unemployed	0
20	Psychology	Part time	19
20	Psychology	Part time	32
21	History	Unemployed	0
21	Business	Part time	20
21	History	Part time	15
23	Psychology	Full time	36
41	Business	Full time	50
30	Physics	Unemployed	0

Create a dataframe via keyboard input to represent these data. Print your dataframe and verify that it is correct.

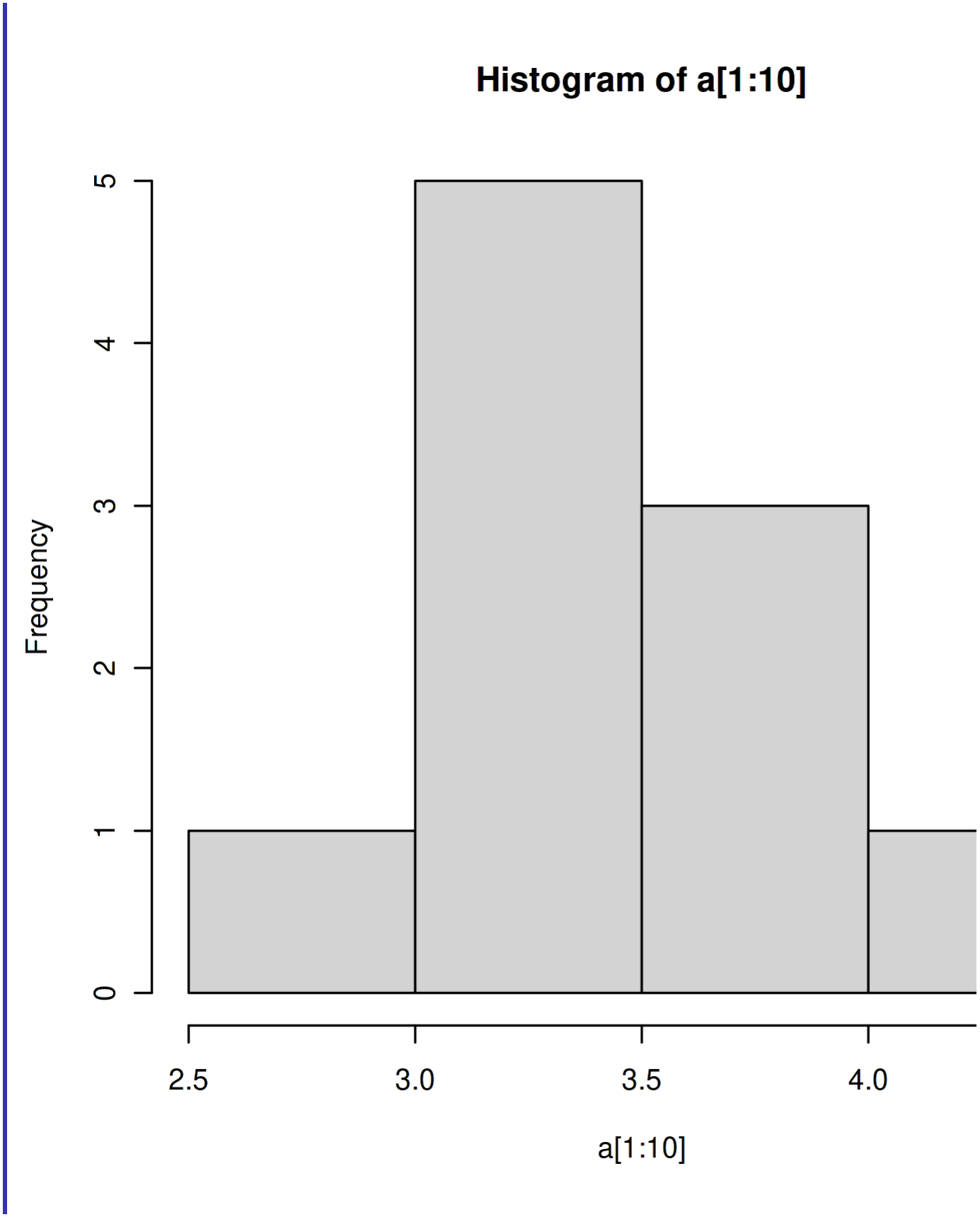
Exercise 2.2: For each variable in the dataset above, make a display (or two!) and compute summary statistics.

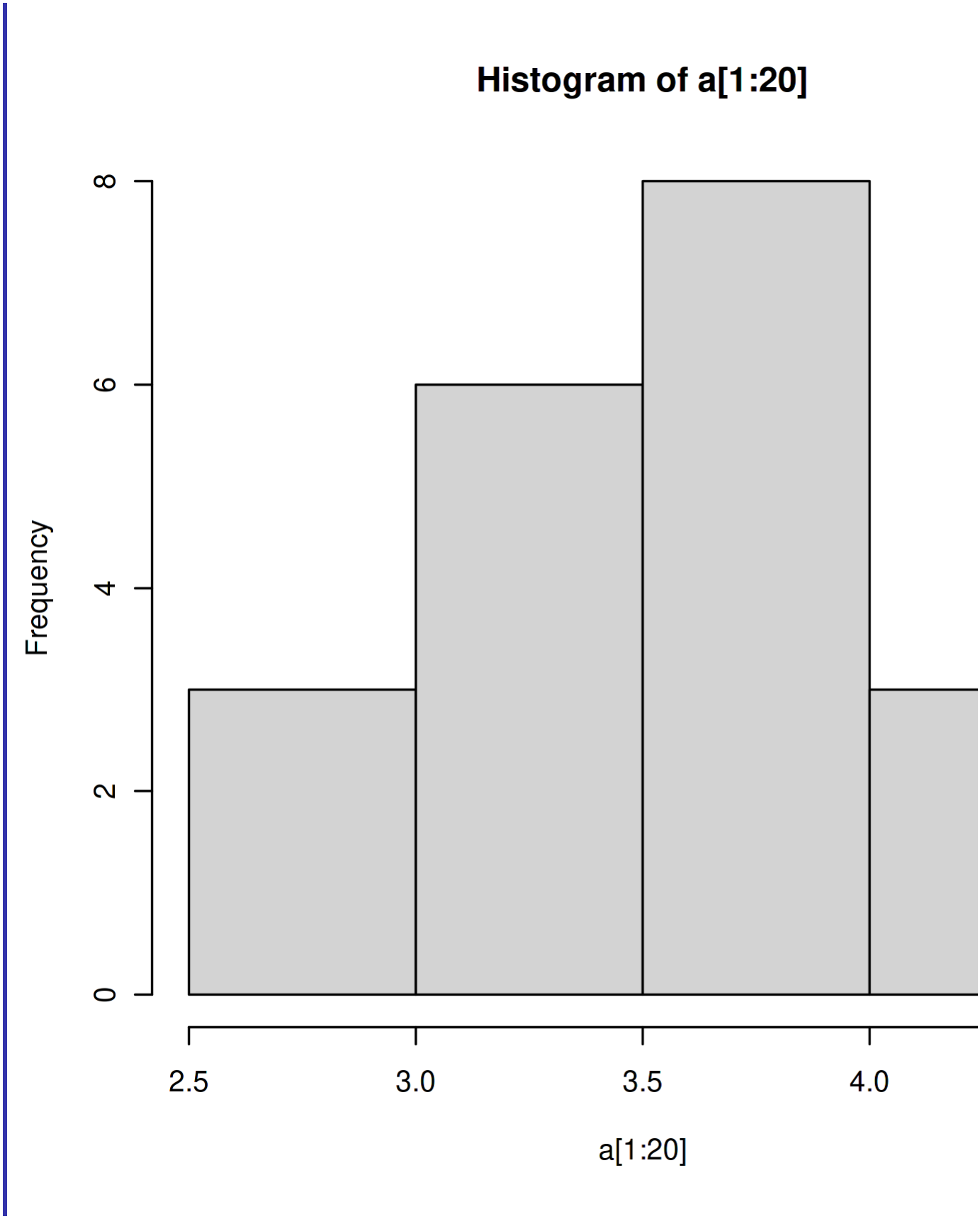
Use the builtin help feature to discover at least a couple of new ways to customize your displays and/or computation
For example, try to figure out how to make a 2-way table for your two categorical variables.

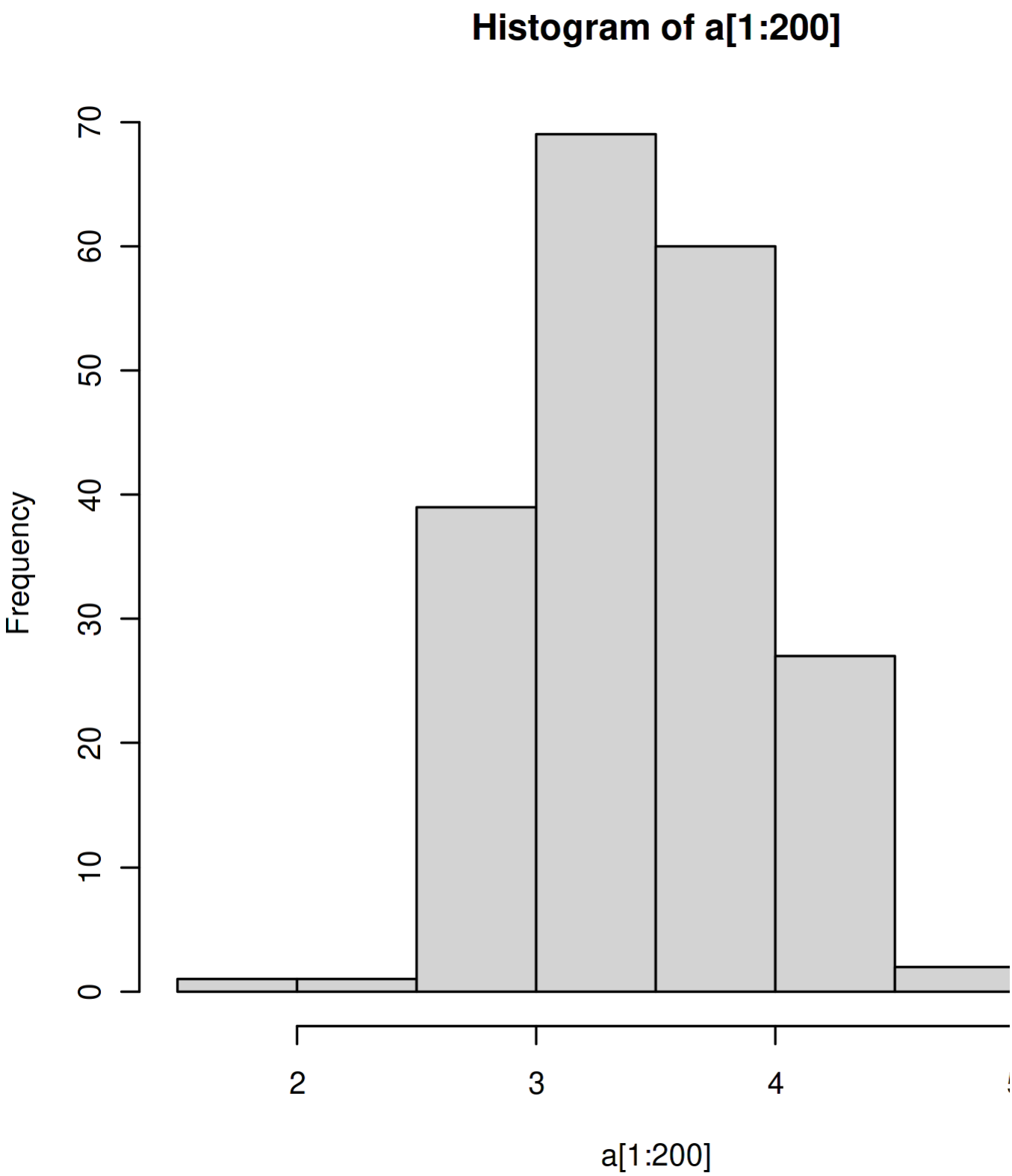
Experiment with the central limit theorem

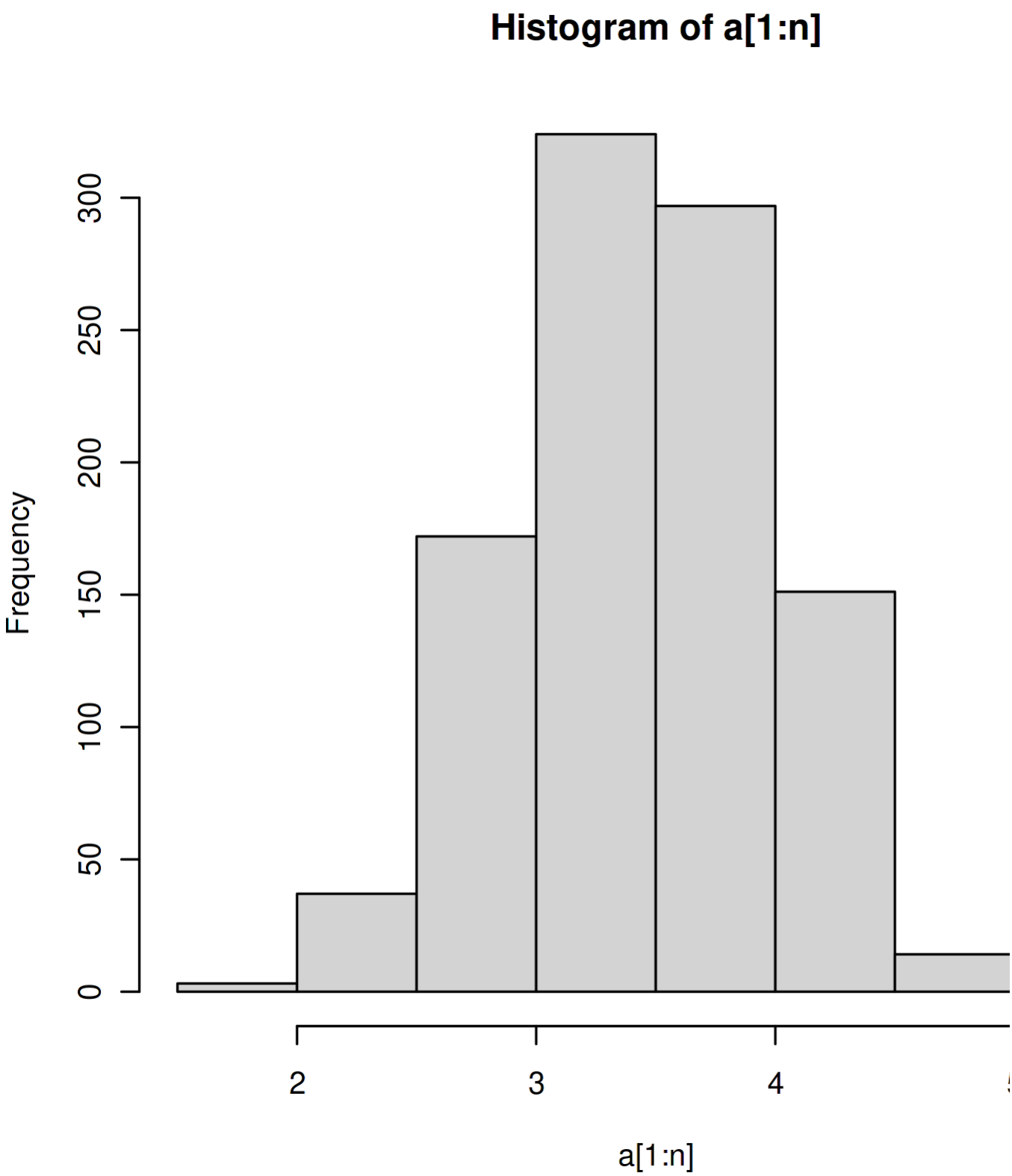
Let us simulate an experiment that rolls a fair 6-sided die 10 times and records the mean. We will run several trials o
experiment and plot a histogram of mean values.

```
%r
a = 0      # initialize vector in which we will store mean values
n = 1000    # number of times to repeat the experiment
for ( i in 1:n ){
    a[i] = mean( sample( 1:6, 10, replace=TRUE ) )
}

 hist( a[1:10] ) ; hist( a[1:20] ) ; hist( a[1:200] ) ; hist( a[1:n] )
```


Histogram of a[1:10]
Frequency
0
1
2
3
4
5
2.5
3.0
3.5
4.0
a[1:10]


Histogram of a[1:20]
Frequency
0
2
4
6
8
2.5
3.0
3.5
4.0
a[1:20]


Histogram of a[1:200]
Frequency
0
10
20
30
40
50
60
70
2
3
4
a[1:200]



Exercise 2.3: Extend the above experiment to rolling a fair 12-sided die 10 times and recording the mean. Perform several trials of the experiment and plot histgrams showing the distribution of mean values.