Chapter 1

Descriptive statistics

Please read Chapter 1 (excl. section 1.3), Chapter 2, and Section 5.1-5.6 from *Learning Statistics with R* before starting these assignments.

Content *Page*

Chapter 1: Descriptive statistics

Learning objectives of this chapter:

- Calculating the mean, mode, median, quartiles and range by hand
- Get started with R: set the working directory, inspect and load data
- Calculating the mean, mode, median and range in R
- Demonstrating the Central Limit Theorem in R

Assignment 1.1: Descriptive statistics by hand

Calculate by hand (or by the use of a pocket calculator) the descriptive statistics that are listed below:

- **mean**
- **mode**
- **median**
- **range**
- **lower and upper quartiles**
- **interquartile range**

1.1 a) 2 7 4 5 8 10 10 7 9 2 8 8 9 4 6

Answer 1.1a:

Mean: _____ Range: _____

Mode: _____ Lower quartile: _____

Median: _____ Upper quartile: _____

Interquartile range: _____

1.1 b) 7 7 6 5 2 1 3 7 5 9 9 10

Answer 1.1b:

Mean: _____ Range: _____

Mode: _____ Lower quartile: _____

Median: _____ Upper quartile: _____

Interquartile range: _____

1.1 c) Describe which question (1.1a or 1.1b) was more difficult to calculate and why.

Answer 1.1c:

1.1 d) Are these data sets **positively skewed** or **negatively skewed** ? Circle the correct answer and explain why you chose this answer using the relation between the **mean** , the **median** , and the **mode** .

Answer 1.1d:

These data sets are **positively** / **negatively** skewed.

Explanation:

Assignment 1.2: Descriptive statistics of small data sets in R

This assignment assumes you opened a new script in the code editor in RStudio. Write and run your own code to answer the following questions.

In R, a one-dimensional row of numbers is represented by a **vector** .

1.2 a) Use the **c()** function to enter the numbers from assignment 1.1a in a new vector called **dataset1** . Next, run the following code in R and explain what you see.

```
View(dataset1)
```

Hint 1.1: Check Part I of the R help on page 150 for more information on how to make a **vector** *.*

R code 1.2a:

Answer 1.2a:

1.2 b) Write and run your own code in R to find the **mean** , **mode** , **median** , and **range** for the vector **dataset1** . Compare your answers with those of assignment 1.1a.

Hint 1.2: Check part IV of the R help on page 156 for descriptive statistics functions.

Hint 1.3: There is no **mode** *function in R, but you can find the mode in a frequency* **table** *.*

R code 1.2b:

Answer 1.2b:

Mean: _____ Median: _____

Mode: _____ Range: _____

1.2 c) Run the following code in R and explain what you see.

```
quantile(dataset1, type = 6)
```

Answer 1.2c:

_____

_____

1.2 d) Use the `c()` function to create a vector `dataset2` with the data from assignment 1.1b and find the `mean`, `mode`, `median`, `range`, and `quartiles` for these data. Compare your answers with the answers of assignment 1.1b.

R code 1.2d:

```

```

Answer 1.2d:

Mean: _____ Range: _____

Mode: _____ Lower quartile: _____

Median: _____ Upper quartile: _____

Assignment 1.3: Descriptive statistics of larger data sets in R

For this assignment, you need the **`bloodPressure.csv`** data set that you can download from the online resources. This data set contains measurements of the age, blood pressure, cholesterol level, gender, and description of a random selection of people. It is normally used to look for relationships between these variables. Note that this is fake data and does not contain actual measurements.

Let's start with importing the data set (which is available in the online resources) into R.

1.3 a) Inspect and run the following code in R to import the blood pressure data and store it in the object **`dataset3`** . Explain how the code works and describe what the **`bloodPressure.csv`** file contains.

```
dataset3 <- read.csv(file.choose())
```

Answer 1.3a:

This method of importing data can be a lot of work if there are many files or if the script will be run many times. Faster methods exist though, for example by providing the full file path.

1.3 b) Describe and test ways this code can be improved to make importing a file easier.

Hint 1.4: Look at Part I of the R help at page 150 to find out more functions for importing data.

Answer 1.3b:

R code 1.3b:

The functions you used for descriptive statistics on the small data sets in assignment 1.1 can also be applied to the data set that is currently stored in **dataset3** .

1.3 c) Find the **mean** , **mode** , **median** , **range** , and **quartiles** for the column **Age** in **dataset3** . Describe this variable in running text using these statistics.

Hint 1.5: First find out how to extract (index) a specific column in a data frame using the **$** *sign.*

R code 1.3c:

Answer 1.3c:

For large data sets, it becomes a lot of work finding the **mode** in a frequency table each time. It is possible to import a package into the R session that contains a function for calculating the **mode** automatically. However, it is also possible to create a function that calculates the **mode** ourselves.

Run the following lines of R code together:

```
getMode <- function(x){
   uniqx <- unique(x)
   uniqx[which.max(tabulate(match(x, uniqx)))]
}
```

You have now created your first R function and you will see it displayed separately in the R environment. This function will give you the **mode** for any **numeric** vector or column. It works by first extracting all unique values, counting their frequency, and then selecting the value with the highest frequency. Note that you can use this function, but will not be required to understand or make functions like this. However, for the interested reader, part III of the R help contains more information on how to create your own functions.

1.3 d) Use the new **getMode()** function to determine the **mode** for column **Age** in **dataset3** and check if it is consistent with your answer for assignment 1.3c.

R code 1.3d:

```
```

Answer 1.3d:

Mode: _____

1.3 e) Find the **mean**, **mode**, **median**, **range**, and **quartiles** for the column **BloodPressure** in **dataset3**. Also use the new **getMode()** function.

R code 1.3e:

```
```

Answer 1.3e:

Mean: _____ Range: _____

Mode: _____ Lower quartile: _____

Median: _____ Upper quartile: _____

1.3 f) Is the distribution of the values in the **BloodPressure** column **positively skewed** or **negatively skewed**? Explain your answer using the relation between the **mean**, **median**, and **mode**.

Answer 1.3f:

These data sets are **positively** / **negatively** / **not** skewed.

Explanation:

_____

_____

_____

1.3 g) Determine the **variance** and **standard deviation** for the column **Cholesterol** in **dataset3** .

R code 1.3g:

Answer 1.3g:

Variance: _____

Standard deviation: _____

1.3 h) Validate the relation between the **variance** and the **standard deviation** by performing a calculation in R.

R code 1.3h:

Assignment 1.4: Demonstrating the Central Limit Theorem in R

The **normal distribution** is important in statistics because many natural phenomena are normally distributed. For example; height, weight, and IQ scores are all distributed according to this bell shaped curve. But it is famous for another reason: the distributions of all **sample means** follows a **normal distribution** . This is useful knowledge because it means that the values in the original population can be distributed in any way, and still the **means** of its samples will be normally distributed. This is called the **Central Limit Theorem** .

To gain an understanding of how the **Central Limit Theorem** works, suppose that you have a single die. Each time you roll that die, you can expect to see a value between 1 and 6. The **population mean** μ (i.e., the sum of the values multiplied by the probability of seeing each value) is calculated as:

$$\mu = \frac{1}{6} \times 1 + \frac{1}{6} \times 2 + \frac{1}{6} \times 3 + \frac{1}{6} \times 4 + \frac{1}{6} \times 5 + \frac{1}{6} \times 6 = 3.5$$

Run the following code in R to confirm the population mean:

```
meanValue <- sum(1:6 * (1 / 6))
print(meanValue)
```

Now let's roll two dice at the same time. If you roll a 3 on the first die and a 1 on the second die, then the **mean** is $\frac{1}{2} \times 1 + \frac{1}{2} \times 3 = 2$. With that knowledge you can create a matrix which will give you the **mean** for any combination of dice rolls (the values in the square brackets, **[ ]** , refer to the values obtained from each die). This is done for you below, but you can try to figure out how to recreate this matrix for yourself.

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,] 1.0  1.5  2.0  2.5  3.0  3.5
## [2,] 1.5  2.0  2.5  3.0  3.5  4.0
## [3,] 2.0  2.5  3.0  3.5  4.0  4.5
## [4,] 2.5  3.0  3.5  4.0  4.5  5.0
## [5,] 3.0  3.5  4.0  4.5  5.0  5.5
## [6,] 3.5  4.0  4.5  5.0  5.5  6.0
```

The **sample mean** of the two dice ranges from 1 to 6. Note that you only see the values 1 and 6 once in the matrix. This is because the only way to get an **sample mean** of 1 is to roll a 1 on our first die and then roll a 1 on our second die as well. In a similar manner, getting an **sample mean** of 6 means you have to roll a 6 on both dice. Compare these **sample means** of 1 and 6 with a **sample mean** of 1.5. There are two ways to obtain a **sample mean** of 1.5. You could roll a 1 on your first die and a 2 on our second die, or you could roll a 2 on our first die and a 1 on our second die. So, there are more ways to get a 1.5 than to get a 1. In fact, the probability of getting a certain value tends to get larger as you get closer to 3.5 (which is the true **mean** of the population). Look at the diagonal entries of the matrix: 3.5 appears six times in the matrix.

From the above, you know that when you roll a die, the **mean** score over the long run will be 3.5. Even though 3.5 isn't an actual value that appears on the die, over the long run if you take the **mean** of the values from multiple rolls, you'd get very close to 3.5. This is the **Central Limit Theorem** in action.

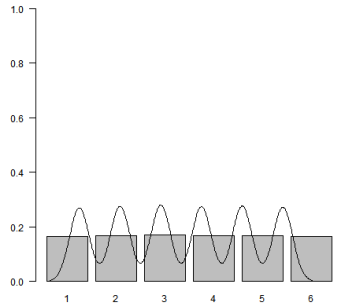
Run the following R code to try this for yourself:

```
rolls <- 1000000
mean(sample(x = 1:6, size = rolls, replace = TRUE))
```

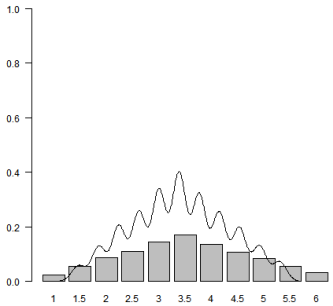
As you can see, the mean of 1000000 rolls is very close to 3.5. However, this trend continues as you add more dice. What's interesting is that the underlying distribution of the rolls is not normal, it is uniform: there's a $\frac{1}{6}$ chance of seeing each face of a particular die. But the **sample means** are normally distributed. Run the following R code to simulate multiple **rolls** of multiple **dice**, calculate their **sample means**, and make a figure of their distribution. If you want to fully understand this code, check out the R help on page 150.

```
roll <- function(rolls, dice){
   means <- NULL
   for(i in 1:rolls){
     means <- c(means, mean(x = sample(1:6, size = dice, replace = TRUE)))
   }
   return(means)
}

rolls <- 10000 # Number of rolls with each die
dice <- 1      # Number of dice

data <- roll(rolls, dice)
barplot(prop.table(table(data)),
        ylim = c(0,1), width = 6/length(table(data)), las = 1)
lines(density(data))
```

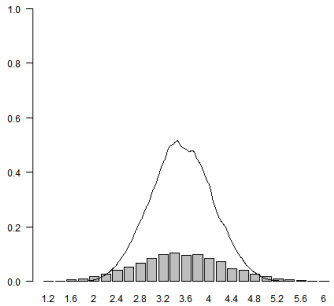
Now gradually increase the number of **dice** and run the code again several times. Watch what happens to the distribution of the **sample means** as it becomes more tight around 3.5.



dice = 1



dice = 2



dice = 5

Saying that the sample distribution becomes more tight around 3.5 as a result of increasing the sample size (the number of **dice**) can be expressed in a mathematical way:

$$\sigma_{\text{sampling distribution}} = \frac{\sigma_{\text{population}}}{\sqrt{n}},$$

where σ is the **standard deviation** and n is the size of your sample. So if the sample size is very large, you can expect the **standard deviation** of the **sampling distribution** to be rather small. Important is that the **mean** of the **sampling distribution** is the same as the **population mean**, but the **standard deviation** is not.