

# Lab Session 3 (Week 11)

## 1.36 What we have learnt so far

In the previous two labs (Week 9 and Week 10) we learnt about variables, vectors, and operators. Furthermore, we were introduced to the uses of functions and their arguments. These were the fundamental steps that we had to take before we can start carrying out statistical tasks. In the last lab session, we began exploring some basic statistical tasks such as creating grouped frequency tables and histograms. These steps, even though simple, allowed us to develop a sense of our data. Creating visual representations of our data is one of R's strongest aspects. For the time being we have only practiced with **hist()** (last week's stats lab) and **pie()** (last week's 'taking it further' tasks), however as we become more confident with R we will also explore some additional features that R **packages** can give us.

R in its core installation has a number of useful functions. However, these functions do not cover all the possible tasks we would wish to perform. In order to overcome this weakness, R allows us to install and use additional **packages**. These packages are more specialised than the core R and can provide us with numerous new statistical tools. Today, we will learn how to install and use such packages and see how they can provide us with useful new features. We will also see a number of examples of how to use R to acquire the **mean**, **median**, and **mode** of a sample. **Finally, in order to make things more interesting we will use some experimental data that R offers by default. So today, we will not be generating datasets comprised of random data.** Instead we will be using some of the built-in datasets.

## 1.37 Learning objectives

- Installing R packages
- Calling R packages
- Use core R and packages to find measures of central tendency

## 1.38 Let's get started

## If you are working on RStudio Cloud skip to step 2 below

1. For this lab session you will need to create a new project in RStudio. Run RStudio and create a new project by clicking on File >> New Project ... .Then select New Directory and then New Project. Enter a Directory name. As mentioned in Week 9, we recommend naming the folder by week, so “Week 11” would be ideal. Click Browse and make sure you place that folder in your “PSYC3000” folder that you created on your Desktop. Click “Create Project” to finalise creating your new project.
2. Now create a new script from File >> New File >> R Script. Save your new script under the name **Week 11**.

Also, please remember that code we want you to type, will appear within a grey box in our instructions. **However, lines that begin with ‘[1]’ are *output***, this means that you **do not** need to type these lines of text in your script.

One of the built-in datasets that R offers for free is called **ToothGrowth**. The ToothGrowth data set contains the result from an experiment studying the effect of vitamin C on tooth growth in 60 Guinea pigs. Each animal received one of three dose levels of vitamin C (0.5, 1, and 2 mg/day) by one of two delivery methods, (orange juice (OJ) or ascorbic acid (a form of vitamin C and coded as VC). Even though this dataset is not the result of Psychological research it does have the structure of a data file from Psychological research where we administer an intervention in three different dosages and two different methods. Let’s see how we can explore this dataset.

In your new script type and run the following:

```
data("ToothGrowth")
```

This informs R that we want to work with the build-in dataset called “ToothGrowth”. If you have a look at your **Environment** you will now see under the section called **Data** an object called **ToothGrowth**. You will also see some additional information, it has 60 observations and 3 variables. If you also move your mouse pointer on top of **ToothGrowth** and leave it there you will get a tooltip window informing you that this is a **data.frame**. For now you can consider a data.frame as a table that contains information for more than one variables. If you click once on **ToothGrowth** then RStudio will open and display this data.frame for you. You can see it in a new tab that opened next to your script. You can see it has three columns, one per variable. The top row includes the variable names. **len** is the variable that contains information on the tooth growth, **supp** is the variable that contains information on the vitamin supplement used, and **dose** is the variable that describes the dosage that was administered. Now return to your script.

If you type and run each the following commands one at a time you will see each variable displayed in your console:

```
ToothGrowth$len
```

```
ToothGrowth$supp
```

```
ToothGrowth$dose
```

So if I want to work with one of the variables in a dataset I can call it by typing the name of the dataset, then type the **\$** symbol, and then type the name of the variable.

**Exercise 1.** Try to create a histogram for the variable **len** of the dataset **ToothGrowth**. *If you are not certain how to do this you can check the solution at the end of this pdf*

## 1.39 Measures of Central Tendency

Let us now see how we can use R functions to calculate the mean, median, and mode for a variable. If we want to find the mean of a variable we can use the function **mean()**. Similarly, if we want to find the median, we can use the **median()** function. Let's see them in practice for the variable **len** of the **Toothgrowth** dataset.

```
# the following command will calculate the mean of len
```

```
mean(ToothGrowth$len)
```

```
# the following command will find the median value of len
```

```
median(ToothGrowth$len)
```

You can see now how with one line of code we can instantly calculate a figure that would have taken us much more time and energy to calculate by hand.

Now let's see if we can also find the **mode** in a similar way. Try the following command and see what happens.

```
mode(ToothGrowth$len)
```

You probably got the answer “numeric”. This is because the function `mode()` corresponds to a different task than what we hoped it would perform. Actually, the core installation of R does not have a function that can find the mode value of a variable. However, as we said earlier this is not a problem as we can install and run external packages that can add additional functionalities.

## 1.40 Adding a new package

In order to have a function that can find the mode of our sample we need to install a new package called **DescTools**. We do that by clicking **Tools** in the menu and then **Install Packages....** Then where it says packages, type in **DescTools** and click **Install**. If you are prompted with any questions then click on **Yes**. RStudio will download and install everything automatically and you should be ready to go once you can see the **>** symbol in the console. Once you install a package you can use it in all future projects and scripts without the need to re-install it. Now we can tell R that we want to use this package in our script by adding and running the following command:

```
library(DescTools)
```

As a good practice we tend to declare the packages we want to use at the top of our scripts. We also need to include the command `library(DescTools)` only once per script, if we need it at all. Now we have the necessary function to find the **mode** of **len** by calling the function `Mode()` that comes with the package **DescTools**. Add the following command and run it to get the mode (**note** that the command in the **DescTools** package - '`Mode()`' - has a capital letter!).

```
library(DescTools)
Mode(ToothGrowth$len)
```

You should get the following output:

```
## Warning: package 'DescTools' was built under R version 4.1.3

## [1] 26.4
## attr(,"freq")
## [1] 4
```

This means that your **mode** is 26.4 and it has a frequency of 4.

## 1.41 To sum up

As we progress through our course we will be using specific packages and functions. You do not need to memorise the packages names. We will be providing those for you. You will also not need to memorise the names of the functions for this term. We will be providing those too.

## 1.42 Additional Exercises (spoiler alert: if you get stuck the answers are at the end of the pdf)

Another dataset that comes build-in with R is called **iris**. The iris dataset gives the measurements in centimeters of the variables sepal length, sepal width, petal length and petal width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica. The following exercises should be completed in one script: they are one exercise but have been broken into small steps to make your work easier, as well as making it easier to refer to the solutions in the end.

**Exercise 5.1** Create a new script and name it **week 11 iris**. Then type in and execute the right command in order to load the dataset **iris**

**Exercise 5.2** Explore the data.frame called **iris** (you can find it in your **Environment**). You should be able to see that some of the included variables are **Sepal.Length**, **Sepal.Width**, **Petal.Length**, and **Petal.Width**. Write code to create one histogram for *each* of the above variables. Spend some time to observe each histogram and think about how the scores are distributed and whether we have **skewed** distributions. (CAUTION: In R, upper case and lower case letter do matter, make sure to type them exactly as you see them)

**Exercise 5.3** Type the right code in order to get the **mean** and the **median** for all the above variables.

**Exercise 5.4** Type the right code in order to get the **mode** for all the above variables. (Hint: do not forget the **library** and remember the importance of upper case letters.)

.

.

.

- .
- .
- .
- .
- .
- .

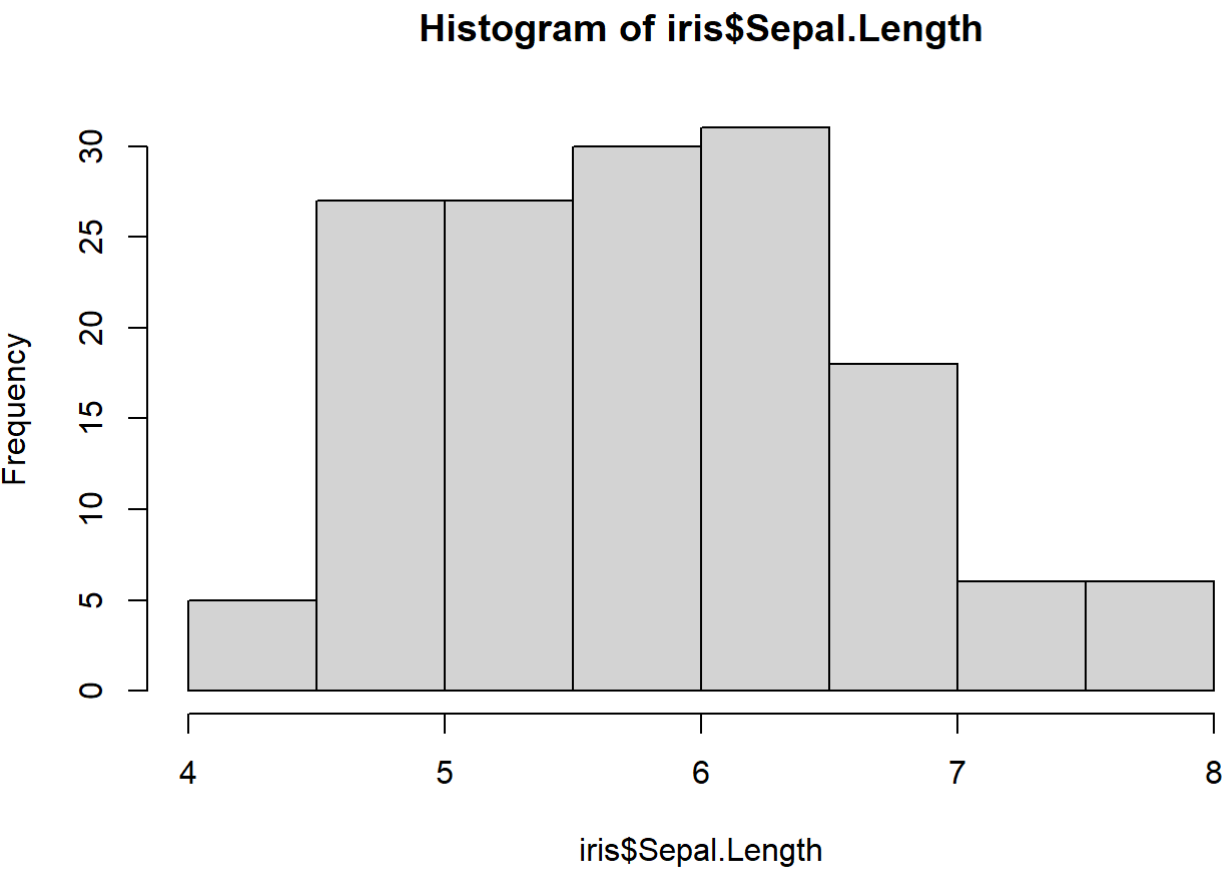
## 1.43 Solutions: SPOILER ALERT

### Exercise 5.1

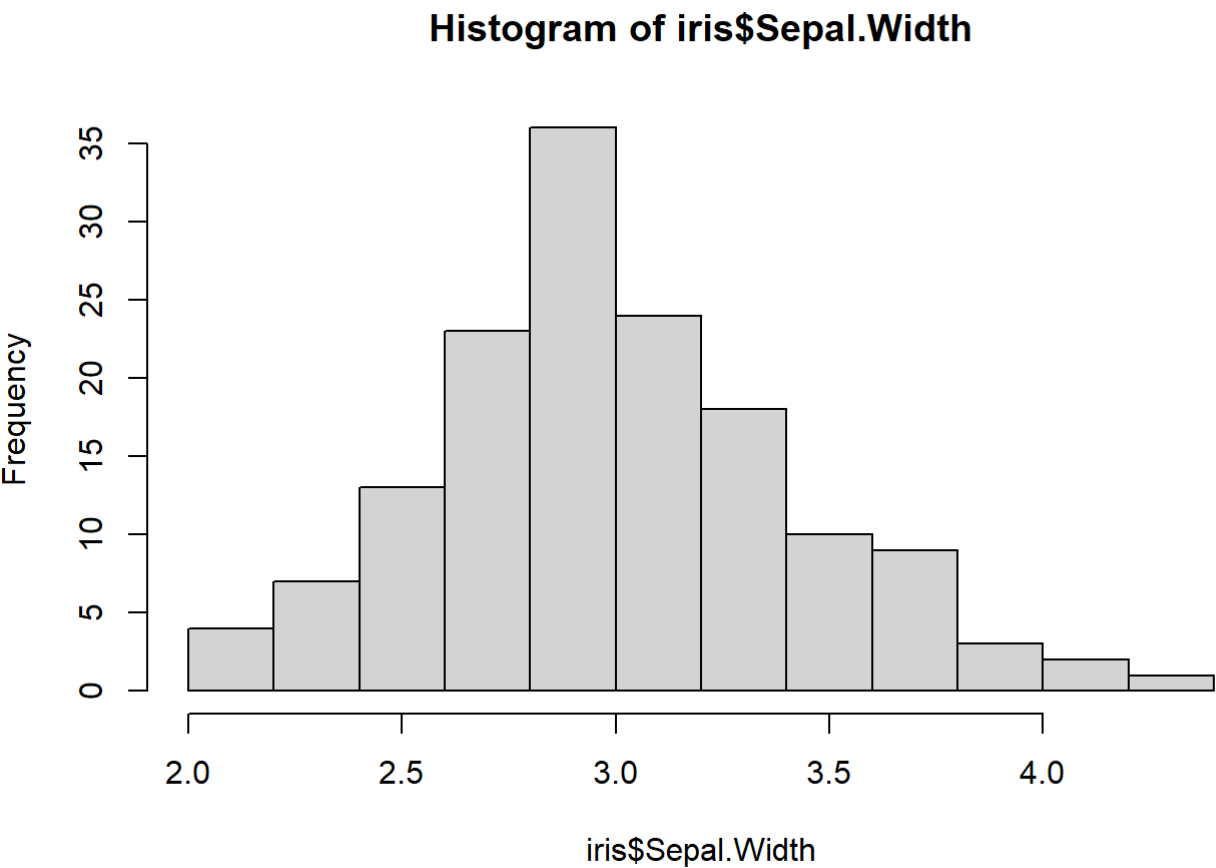
```
data(iris)
```

### Exercise 5.2

```
hist(iris$Sepal.Length)
```



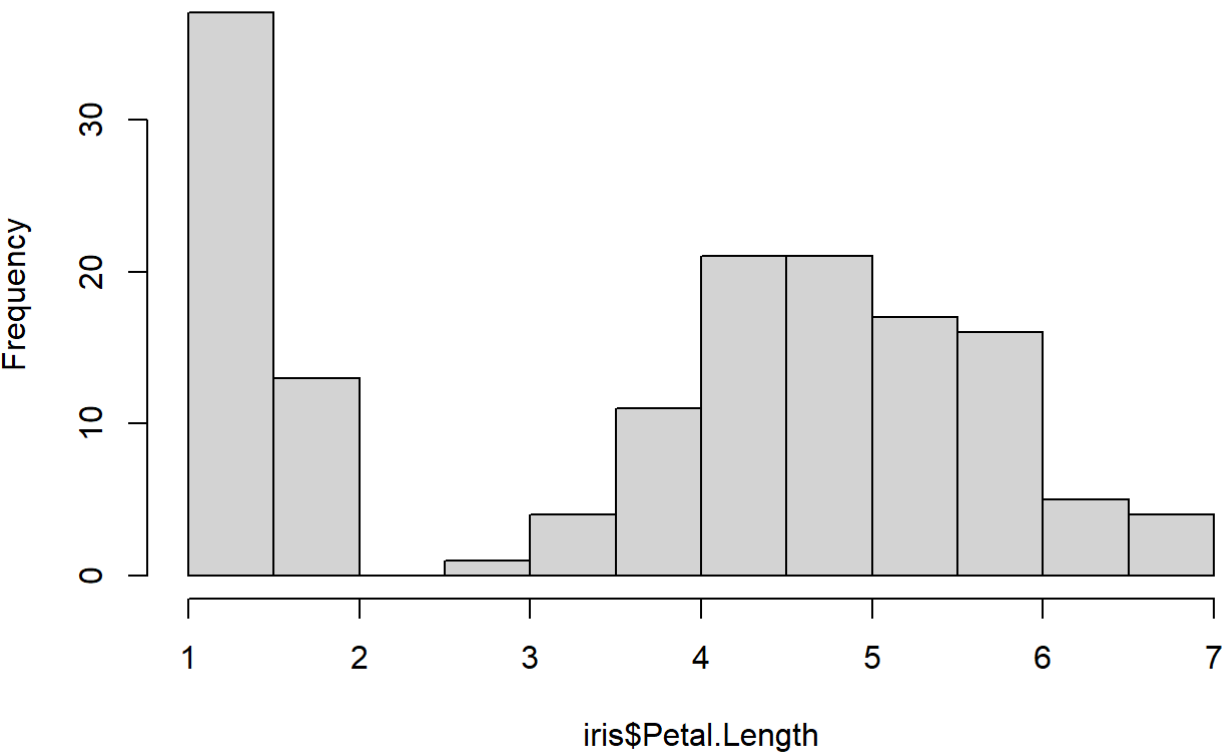
```
hist(iris$Sepal.Width)
```



```
hist(iris$Petal.Length)
```

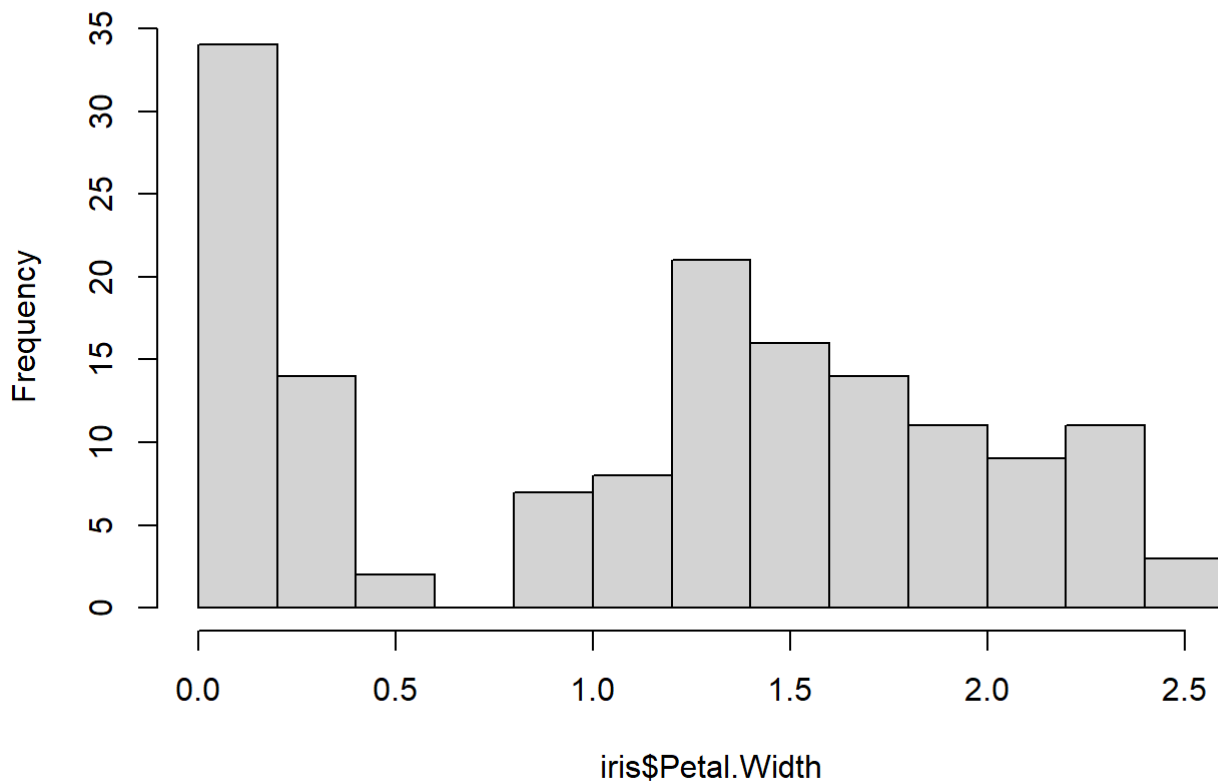


### Histogram of iris\$Petal.Length



```
hist(iris$Petal.Width)
```

## Histogram of iris\$Petal.Width



### Exercise 5.3

```
mean(iris$Sepal.Length)
## [1] 5.843333
mean(iris$Sepal.Width)
## [1] 3.057333
mean(iris$Petal.Length)
## [1] 3.758
mean(iris$Petal.Width)
## [1] 1.199333
median(iris$Sepal.Length)
## [1] 5.8
median(iris$Sepal.Width)
## [1] 3
median(iris$Petal.Length)
## [1] 4.35
median(iris$Petal.Width)
## [1] 1.3
```

## Exercise 5.4

```
library(DescTools)

Mode(iris$Sepal.Length)

## [1] 5
## attr(,"freq")
## [1] 10

Mode(iris$Sepal.Width)

## [1] 3
## attr(,"freq")
## [1] 26

Mode(iris$Petal.Length)

## [1] 1.4 1.5
## attr(,"freq")
## [1] 13

Mode(iris$Petal.Width)

## [1] 0.2
## attr(,"freq")
## [1] 29
```