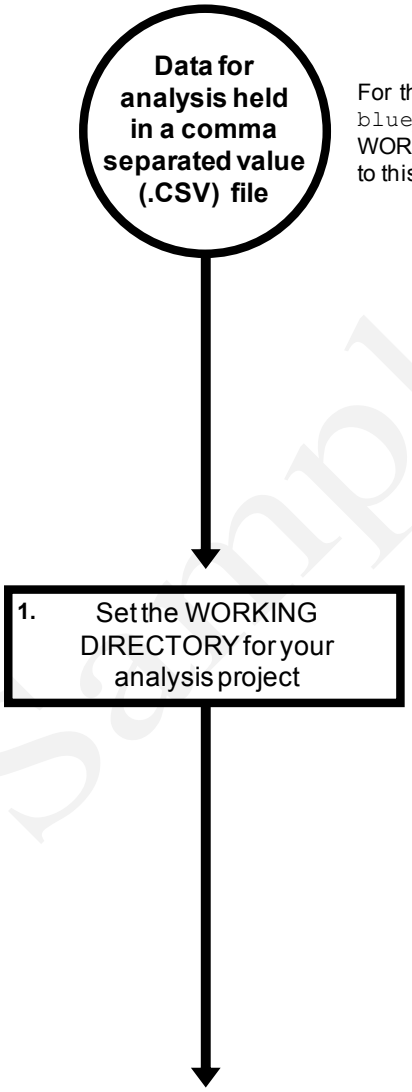
EXERCISE 2.1: HOW TO MAKE A FREQUENCY DISTRIBUTION HISTOGRAM:

One of the first graphs that you are likely to want to make is a frequency distribution histogram. This is a graph that shows the distribution of the data in a data set in relation to one of its variables. In order to make a frequency distribution histogram in R, you need to have your data arranged in a spreadsheet or table where each row contains data from a single record in your data set. In this table, there also needs to be a column which contains the values for the continuous variable you wish to create a frequency distribution histogram for. For this exercise, you will start by creating a histogram that shows the distribution of a set of nest boxes used to study breeding behaviour in hole-nesting birds in relation to elevation. To do this, work through the following flow diagram:



For this example, the data set you will use is stored in a file called `blue_tit_occupancy_data.csv` that is located in the WORKING DIRECTORY folder you created during the introduction to this chapter.

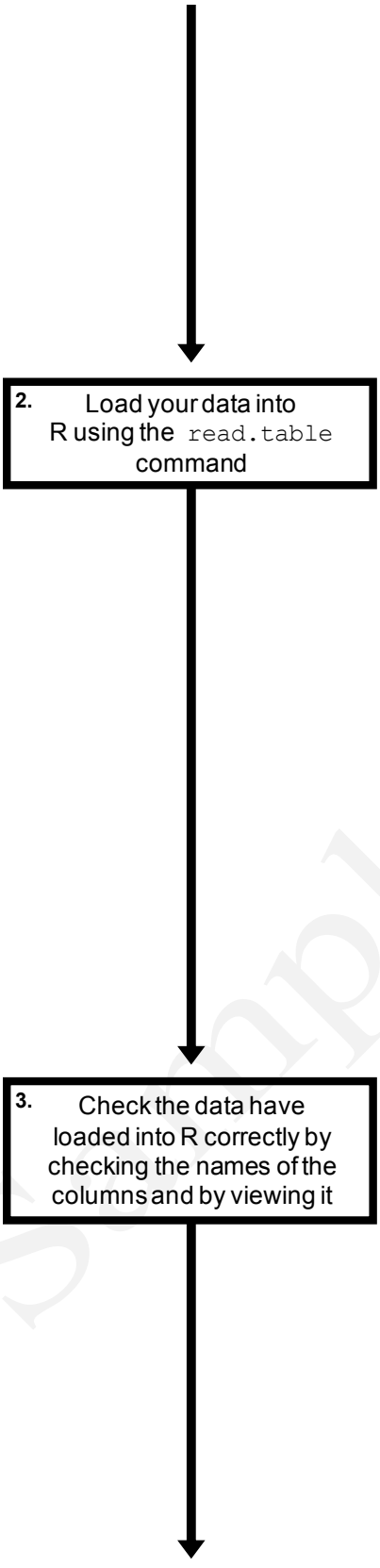
Before you start any analysis in R, you first need to set the WORKING DIRECTORY. To do this, enter the text `setwd("` and then type the address of your WORKING DIRECTORY, using slashes (/) as the folder separators, before entering a second quotation mark followed by a closing bracket, like this `")`. For example, if your WORKING DIRECTORY has the address C:\STATS_FOR_BIOLOGISTS_ONE, your `setwd` command should look like this:

```
setwd("C:/STATS_FOR_BIOLOGISTS_ONE")
```

If you are using RGUI, enter your `setwd` command in the R CONSOLE window (remembering to use the address of your own WORKING DIRECTORY folder in it) and then press the ENTER key on your keyboard. If you are using RStudio, enter your `setwd` command into the SCRIPT EDITOR window. To run it, select it and then click on the RUN button at the top of this window. You will enter all the remaining commands for this exercise in a similar manner, depending on the user interface you are using.

To check that your WORKING DIRECTORY has been set properly, enter the command `getwd()` and carefully check that the address it returns is the same as the one for the STATS_FOR_BIOLOGISTS_ONE folder you created at the start of this chapter.

Before you move on to step 2, make sure that all the data you wish to use in your analysis project are located in this WORKING DIRECTORY folder. In this case, this is a file called `blue_tit_occupancy_data.csv`. **NOTE:** If the data you are going to import into R in step 2 are not located in the WORKING DIRECTORY you set in this step, the import code provided in the next step will not work.



The `read.table` command provides the easiest way to load data held in a .CSV file (and stored in the WORKING DIRECTORY you set in step 1) into R so you can analyse it. To do this for the data set being used in this example, enter the following command into R:

```
frequency_histogram_data <- read.table(file=
   "blue_tit_occupancy_data.csv", sep=",",
                header=TRUE)
```

This code has to be entered exactly as it is written here or it will not work. If you wish to use the copy-and-paste approach for entering this command, copy the text directly below CODE BLOCK 25 in the document R_CODE_BASIC_STATS_WORKBOOK.DOC and paste it into R.

This command will create a new object in R called `frequency_histogram_data` which will contain the data from the specified .CSV file. To import a different .CSV file into R, all you need to do is change the file name in the `file` argument to the name of the one you wish to import. You can also use whatever name you wish for the R object which will be created by this command. To do this, simply replace `frequency_histogram_data` at the start of the first line of the above code with the name you wish to use for it. **NOTE:** If your .CSV data set uses a semicolon as the decimal separator, you would need to replace the `sep=","` argument with `sep=";"`.

Whenever you import any data into R you need to check that they have loaded correctly. First, you need to check that all the required columns are present in the R object you just created. To do this, enter the following command into R:

```
names(frequency_histogram_data)
```

This is CODE BLOCK 26 in the document R_CODE_BASIC_STATS_WORKBOOK.DOC. This command will return the names used for each column in the R object you just created. For this example, the names should be `box_number`, `latitude`, `longitude`, `occupied`, `elevation` and `el_cat`.

Next, you should view the contents of the whole table using the `View` command. This is done by entering following code into R:

```
View(frequency_histogram_data)
```

This is CODE BLOCK 27 in the document R_CODE_BASIC_STATS_WORKBOOK.DOC. This command will open a DATA VIEWER window where you can examine your data set and check that the correct data have been loaded into R.

Once your data have been successfully imported into R, you are ready to create your initial frequency distribution histogram. You will use this initial histogram to check what your final graph will look like. To do this, enter the following command into R:

```
hist(frequency_histogram_data$elevation,
                nclass=11)
```

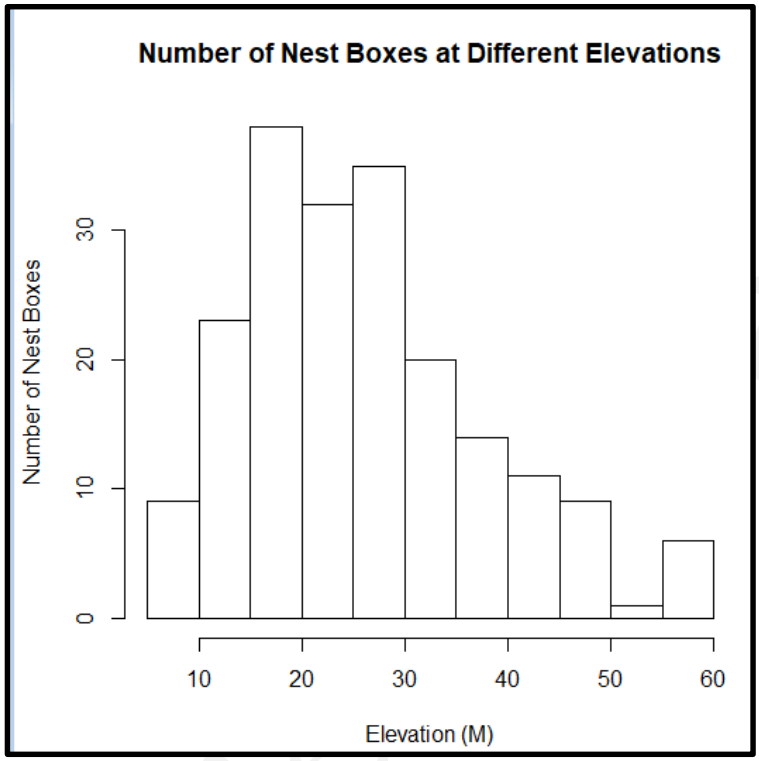
This is CODE BLOCK 28 in the document R_CODE_BASIC_STATS_WORKBOOK.DOC. This `hist` command will create a frequency distribution histogram from the data in the column called `elevation` in the R object called `frequency_histogram_data` created in step 2 of this exercise. The argument `nclass=11` in this command means that the data will be divided into eleven equal-sized classes to make this histogram. If you are not happy with how your frequency distribution histogram looks at this stage, you can try using a different number of classes. This can be done by replacing the `11` in the `nclass` argument with a different number and then re-running it. This can be repeated as often as you need to using different numbers until you are happy with how your frequency distribution histogram looks. **NOTE:** The number of classes in your histogram will not necessarily exactly match the number you have entered in the `nclass` argument. This is because R will adjust the actual number of classes up or down by a small amount from this value to create a histogram with a more appropriate distribution of data between the classes.

Once you have found the settings required to make a frequency distribution histogram that you are happy with, you are ready to create your final histogram. This will use the settings for the `hist` command identified in step 4 and include some additional arguments to add labels to it. To do this, enter the following code into R:

```
hist(frequency_histogram_data$elevation,
  nclass=11, xlab="Elevation (M)", ylab=
"Number of Nest Boxes", main="Number of Nest
      Boxes at Different Elevations")
```

This is CODE BLOCK 29 in the document R_CODE_BASIC_STATS_WORKBOOK.DOC. This code adds three new arguments to the `hist` command. These provide a label for the X axis (specified by the `xlab` argument), a label the Y axis (specified by the `ylab` argument) and a title for the graph (specified by the `main` argument).

At the end of the first part of this exercise, you should have a frequency distribution histogram that looks like this:



Once you have created a histogram, you can export it from R so that you can include it in a manuscript or presentation. If you are using RGUI, you can do this by clicking on the R GRAPHICS window containing your frequency distribution histogram to select it, before clicking on FILE on the main menu bar and selecting SAVE AS. This will allow you to save it in a variety of different formats. If you are using RStudio, you can export your graph by clicking on the EXPORT button at the top of the window displaying your frequency distribution histogram and selecting SAVE AS IMAGE.

When including a frequency distribution histogram in a manuscript, it is important that you provide an appropriate figure legend for it. This legend should provide all the information required for the reader to interpret the contents of the graph. For the above frequency distribution histogram, an appropriate legend would be:

Figure 1: *The distribution of nest boxes used to study breeding behaviour of hole-nesting birds in relation to the local land elevation (measured in metres).*

The `hist` command used to create the frequency distribution histograms in this exercise can be modified using a range of additional arguments to create a graph with the exact characteristics you wish it to have. The additional arguments that biologists most commonly use when creating histograms are provided in the table below, while the full list of additional arguments that can be used with the `hist` command can be found at *www.rdocumentation.org/packages/graphics/versions/3.6.1/topics/hist*).

Additional Argument	How To Use It
`col`	This argument allows you to specify the fill colour of your frequency distribution histogram. The option for this argument can be the name of the desired colour. For example, including the argument `col="blue"` would create a frequency distribution histogram with blue bars on it. Alternatively, you can use a hexadecimal code to specify your desired colour. For example, including the argument `col="#ff3300"` would create a histogram with red bars on it. You can find a full list of hexadecimal codes for different colours at *www.color-hex.com*.
`xlim`	This argument allows you to specify the exact minimum and maximum values for the X axis of your frequency distribution histogram. The options for this argument can be any pair of numbers. For example, including the argument `xlim=c(0,70)` will create a frequency distribution with an X axis that has values ranging from 0 to 70, while including the argument `xlim=c(20,90)` will create a frequency distribution histogram with an X axis that has values ranging from 20 to 90.
`ylim`	This argument allows you to specify the exact minimum and maximum values for the Y axis of your frequency distribution histogram. The options for this argument can be any pair of numbers. For example, including the argument `ylim=c(0,40)` will create a frequency distribution with a Y axis that has values ranging from 0 to 40, while including the argument `ylim=c(20,60)` will create a frequency distribution histogram with a Y axis that has values ranging from 20 to 60.
`main`	This argument allows you to specify the text that will appear as the title of your frequency distribution histogram. The option for this argument is the text you wish to use for your title. For example, including the argument `main="Number of Nest Boxes at Different Elevations"` will create a frequency distribution histogram with this text as the title above it. **NOTE:** This additional argument is optional, and in many cases you will include this information in your figure legend rather than on the graph itself.
`xlab`	This argument allows you to specify the label that will appear alongside the X axis of your frequency distribution histogram. The option for this argument is the text you wish to use as the label for it. For example, including the argument `xlab="Elevation (M)"` will create a frequency distribution histogram with this label alongside its X axis.
`ylab`	This argument allows you to specify the label that will appear alongside the Y axis of your frequency distribution histogram. The option for this argument is the text you wish to use as the label for it. For example, including the argument `ylab="Number of Nest Boxes"` will create a frequency distribution histogram with this label alongside its Y axis.

Additional Argument	How To Use It
`labels`	This argument allows you to show the number of records in each class above the individual bars of your frequency distribution histogram, or add any other type of label to them. The options for this argument can either be either a list of labels or a logical argument. For example, to add labels that show the number of records represented by each bar on your histogram you would include the additional argument `labels=TRUE`.
`nclass`	This argument allows you to set the number of bars that will appear on your frequency distribution histogram. The option for this argument can be any positive whole number. The number of classes in your histogram will not necessarily exactly match the number you have entered in the `nclass` argument. This is because R will adjust the actual number of classes up or down from this value to create a histogram with a more appropriate distribution of data between the classes. For example, including the argument `nclass=10` (rather than `nclass=11`) in the above example will still result in a frequency distribution histogram for this data set with 11 bars on it, while including the argument `nclass=5` will create a frequency distribution histogram for this data set with 6 bars on it.

For the next part of this exercise, you will customise your `hist` command in a number of different ways. Firstly, you will alter the contents of the `nclass` argument to change the number of bars (which are referred to as classes in R) used to make your histogram. To do this, change the `nclass` value from 11 to 6 in the `hist` command in step 5 of the above flow diagram. The modified code should look like this (required modifications are highlighted in **bold**):

```
hist(frequency_histogram_data$elevation, nclass=6,
xlab="Elevation (M)", ylab="Number of Nest Boxes",
main="Number of Nest Boxes at Different Elevations")
```

You can modify this code either by editing it in the R CONSOLE window of RGUI or through the SCRIPT EDITOR window of RStudio (depending on which interface you are using). If you are entering commands directly into the R CONSOLE window, you can use the UP arrow on your keyboard to bring commands you have previously run during the same session back on to the command line of this window, and then use the LEFT and RIGHT arrows to scroll through and edit them. In this case, use the UP arrow to bring the previous version of the `hist` command back onto the command line and edit it so that it looks like the one above. Once you have finished modifying this command, you can run it by pressing the ENTER key on your keyboard. If you are using RStudio, you can copy and paste the original `hist` command in the SCRIPT EDITOR window before editing the new