

How To Create Your First Graphs In R Using GGPlot

As a biologist, one of the first types of graph which you are likely to want to make from your data is a frequency distribution histogram. This is a graph which shows how records in a data set are distributed across a range of values for a continuous variable. As a result, in this chapter, you will use this type of graph to learn the basics of how to make graphs and other types of data visualisations in R. Simple frequency distribution histograms can be created in R with the `hist` command (see Exercise 2.1 in *An Introduction to Basic Statistics for Biologists using R*). However, such histograms are very basic and they are limited in terms of the ways that data can be divided into bins (or bars) to be plotted on them, the ways that they can be customised, and the different ways that the distributional information can be displayed. As an alternative, many biologists use a more advance graphing package called GGPlot to create not only frequency distribution histograms, but also most other types of graphs and data visualisations they wish to make from their data in R.

The GGPlot package contains an incredibly powerful set of tools for making many different types of graphs and for modifying almost every part of them so that they look exactly the way you wish them to look. This ability to customise almost every possible aspect of the contents and appearance of a graph comes at a cost. This cost is that while the code used to create basic graphs may be relatively simple, and usually consisting of just two commands (see Exercise 1.1 of this workbook), the additions you need to make to this code to fully customise the resulting graph can substantially increase the complexity of the code required to make it (see Exercise 1.2). As a result, using GGPlot can feel quite daunting for those setting out to use it for the first time. However, while using GGPlot can require a very large block of code to produce a graph with a specific appearance (such as that detailed in the instructions to authors from a scientific journal), the underlying structure of the code required to do this is actually quite simple, and with a bit of time and experience, it becomes relatively easy to construct the code required to fully customise any type of graph. The key

to understanding how to structure the code required to do this is to understand that each individual element of the graph that you wish to alter can be assigned either its own command, or its own argument(s) within a command. This means that the long, complex block of code needed to create a final graph with a specific appearance can be built up step by step by adding the code required to modify each individual element that you wish to modify one bit at a time, until you have all the code required to produce a graph with the exact appearance you are looking to achieve.

In order to learn how you can do this, in this chapter, you will work through a series of exercises based around producing frequency distribution graphs using GGPlot. In these exercises, you will learn how to customise many different elements of GGPlot graphs, including how to specify exactly how your data are displayed (as will be done in Exercise 1.1), how the various elements of the graph, such as the axes, the labels and the background, are drawn to produce a publication quality graph (see Exercise 1.2), how the frequency distribution data are plotted (see Exercise 1.3), and how multiple data series can be displayed either on the same graph or on separate graphs that can then be combined into a multi-panel figure (see Exercise 1.4). While you will learn how to do all of these tasks with GGPlot using frequency distribution graphs as examples, the same skills can be applied to almost all the other types of graph and data visualisation that can be made with this package. Thus, by the end of this chapter, you will not only know how to make frequency distribution graphs in a variety of different ways, but you will also have learned how you can customise the various elements of a graph using the tools in the GGPlot package to create high quality and informative data visualisations.

Before you start the exercises in this chapter, you first need to create a **WORKING DIRECTORY** folder on your computer and load the necessary data into it. To do this on a computer with a Windows operating system, open Windows Explorer and navigate to the location where you would like to create the folder (such as your C:\ drive or your DOCUMENTS folder). Next, right click anywhere in this location and select **NEW> FOLDER**. Now call this folder **STATS_FOR_BIOLOGISTS_TWO** by typing this into the folder name section to replace what it is currently called (which will most likely be **NEW FOLDER**). To create a **WORKING DIRECTORY** folder on a computer running a Mac operating system, open Finder and navigate to the location where you would like to create the folder (such as your DOCUMENTS folder or your DESKTOP). Next, click on **FILE>**

NEW FOLDER, and then type the name `STATS_FOR_BIOLOGISTS_TWO` before pressing the ENTER key on your keyboard.

Once you have created your WORKING DIRECTORY folder, you are ready to download the data sets you will use for the exercises in this workbook from www.gisinecology.com/stats-for-biologists-2. After you have downloaded the compressed folder containing the required data by following the instructions provided on that page, you need to extract all the data files from it and copy them into the folder called `STATS_FOR_BIOLOGISTS_TWO` that you have just created.

Next, you need to check that the required data have been extracted to the correct folder. If you are using a computer with a Windows operating system, you can use Windows Explorer to open your newly created WORKING DIRECTORY folder and examine its contents. If all the files from the compressed folder are present in it (there should be a total of 90 of them), you can click on the folder icon at the left hand end of the ADDRESS BAR at the top of the WINDOWS EXPLORER window to reveal its full address. Note this address down somewhere as you will need it to set this folder as your WORKING DIRECTORY during the exercises provided in this workbook (see pages 12 and 13 for details of how to modify folder addresses so they will be recognised by R).

If you are using a computer with a Mac operating system, you can use Finder to open your newly created WORKING DIRECTORY folder and examine its contents. If all the required data files are present in it (there should be a total of 90 of them), press the CMD and I keys on your keyboard at the same time. This will open the GET INFO window where you will find its address (which is also called the pathway). Note this address down somewhere as you will need it to set this folder as your WORKING DIRECTORY during the exercises provided in this workbook (see pages 12 and 13 for details of how to modify folder addresses so they will be recognised by R).

After you have loaded the required data into your WORKING DIRECTORY folder, you can open RGUI or RStudio, depending on which option you wish to use (see Chapter 2 for more details). Once you have opened your preferred R user interface, you need to create a file called `CHAPTER_THREE_EXERCISES` where you will save the results of your analyses from your R CONSOLE window as you work through this chapter. To do this

using RGUI, click on the FILE menu and select SAVE WORKSPACE. To do this in RStudio, click on SESSION and select SAVE WORKSPACE AS. In both cases, save it as a WORKSPACE file with the name CHAPTER_THREE_EXERCISES.RDATA in your WORKING DIRECTORY folder (this will be the one called STATS_FOR_BIOLOGISTS_TWO that you have just created). If you are using RStudio, you will also want to save the contents of your SCRIPT EDITOR window (where you will enter and edit the R code you will use to carry out specific commands). If this window is not already visible, click on the FILE on the main menu bar of RStudio and select NEW FILE> R SCRIPT. To save the contents of your SCRIPT EDITOR window, click on the FILE menu and select SAVE AS. Save your file as an R SCRIPT file with the name CHAPTER_THREE_EXERCISES.R in your WORKING DIRECTORY folder. As you work through the exercises in this chapter, remember to regularly save the contents of your R CONSOLE window (which will contain the R objects you have created up to that point) to your WORKSPACE file and, if you are using RStudio, the contents of your SCRIPT EDITOR window to your R SCRIPT file.

Finally, you need to remove any data that are currently held in R's temporary memory. To do this, enter the following command into R:

```
rm(list=ls())
```

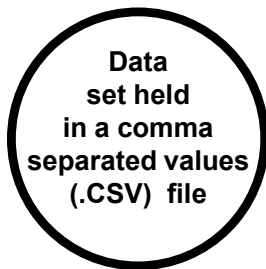
If you are using RGUI, you can simply type this code after the command prompt at the bottom of the R CONSOLE window (it looks like this: >) and then press the ENTER key on your keyboard to run it. If you are using RStudio, you can type this command into the SCRIPT EDITOR window (the upper left hand window). To run this command, select it and then click on the RUN button at the top of this window. This will run it in the R CONSOLE window (the lower left hand one in the main RStudio user interface). You are now ready to start the exercises in this chapter.

EXERCISE 1.1: HOW TO CREATE A BASIC GRAPH IN R USING GGPlot:

The first step in making a graph with the exact appearance that you wish it to have using GGPlot is to create a basic version of it using a short block of code. This code will consist of two commands, one to create a blank GGPlot graph (called `ggplot`) and a second to set the type of graph that will be added to it, such as a histogram, a bar graph, a line graph or a scatter plot, which will be referred to here as the graphing command. These two commands will contain a number of elements and arguments that will not only determine what variable(s) will be used to make the graph, but also exactly how they will be displayed, and these should be modified before you move on to customising any other elements of your graph, such as the colours used for it, the scales used for the axes and the fonts used for the labels (which you will learn how to do in Exercise 1.2). In this exercise, you will learn how to make a basic graph by creating a frequency distribution histogram from a specific data set using a graphing command called `geom_histogram`. Within this command, you will set how the data are divided into bins (or bars) in a number of different ways by modifying the arguments and settings that are used in it. By doing this, you can obtain frequency distribution histograms that look quite different, and as a result, the decisions you make when deciding how to do this will be an important factor for determining exactly how your final graph will look. To create your basic frequency distribution histogram and set how the data will be divided into bins in order to be displayed on it, work through the flow diagram that starts at the top of the next page.

The data that you will use for this exercise, and the others in this chapter, come from a study of the dietary preferences of a range of whale and dolphin species (collectively known as cetaceans). Rather than looking at the species of prey consumed, this study looked at the size distribution of prey and how this varied between different cetacean species. In order to do this, it calculated a predator-prey size ratio (or PPSR for short) for all prey items recovered from the stomachs of each species. This is calculated by dividing the size of the prey found in an individual's stomach contents by its body length. A PPSR of 0.1 means that a specific prey item was 10% of the body length of the individual cetacean that it was recovered from, while a PPSR of 0.01 would mean that it was only 1% of the predator's body length. By comparing the frequency distribution histograms of the PPSR of different cetacean species, their prey size preferences can be identified and compared. This, in turn, allows cetacean species to be divided into different ecological guilds based on the relative

sizes of prey that they consume which are linked to other aspects of cetacean ecology, such as how they capture their prey (see MacLeod *et al.* 2006. *MEPS*. 236: 295-307 for more details).



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

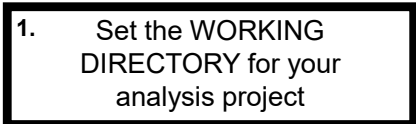
Before you start any analysis in R, you first need to set the WORKING DIRECTORY. To do this, enter the text `setwd("` and then type the address of your WORKING DIRECTORY, using slashes (`/`) as the folder separators, before entering a second quotation mark followed by a closing bracket, like this `")`.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


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

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_TWO` 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 `cetacean_pre_y_sizes.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.





2. Load your data into R using the `read.table` command

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:

```
cetacean_pre_y_sizes <-  
read.table(file="cetacean_pre_y_sizes.csv",  
           sep=";", as.is=FALSE, 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 1 in the document R_CODE_DATA_VISUALISATION_WORKBOOK.DOC and paste it into R.

This command will create a new object in R called `cetacean_pre_y_sizes` 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 `cetacean_pre_y_sizes` 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 column 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(cetacean_pre_y_sizes)
```

This is CODE BLOCK 2 in the document R_CODE_DATA_VISUALISATION_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: `rissos_dolphin`, `pilot_whale`, `common_dolphin`, `white_beaked_dolphin`, `striped_dolphin`, `bottlenose_dolphin`, `atlantic_white_sided_dolphin`, `harbour_porpoise`, `northern_bottlenose_whale`, `pygmy_sperm_whale`, `sowerbys_beaked_whale`, `sperm_whale` and `cuviers_beaked_whale`.



3. Check the data have loaded into R correctly by checking the names of the columns and by viewing it

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

```
View(cetacean_pre_y_sizes)
```

This is CODE BLOCK 3 in the document R_CODE_DATA_VISUALISATION_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.



4. If required, download and install the `ggplot2` package into your version of R


Once your data have been successfully imported into R, you are ready to create your first graph. However, you will only be able to do this if you have the `ggplot2` package installed in your copy of R. To check whether you already have this package installed, enter the following command into R:

```
library()
```

This will open the R PACKAGES AVAILABLE window which will have an alphabetical list of all the packages already installed in your version of R. Scroll down and see if the `ggplot2` package is on this list. If it is, you can go straight to step 5 (where you will load its command library into your analysis project). If it isn't, you can download and install it by entering the following command into R:

```
install.packages("ggplot2")
```

This is CODE BLOCK 4 in the document `R_CODE_DATA_VISUALISATION_WORKBOOK.DOC`. If, when you run it, you are provided with any additional instructions, follow them as outlined on your screen.

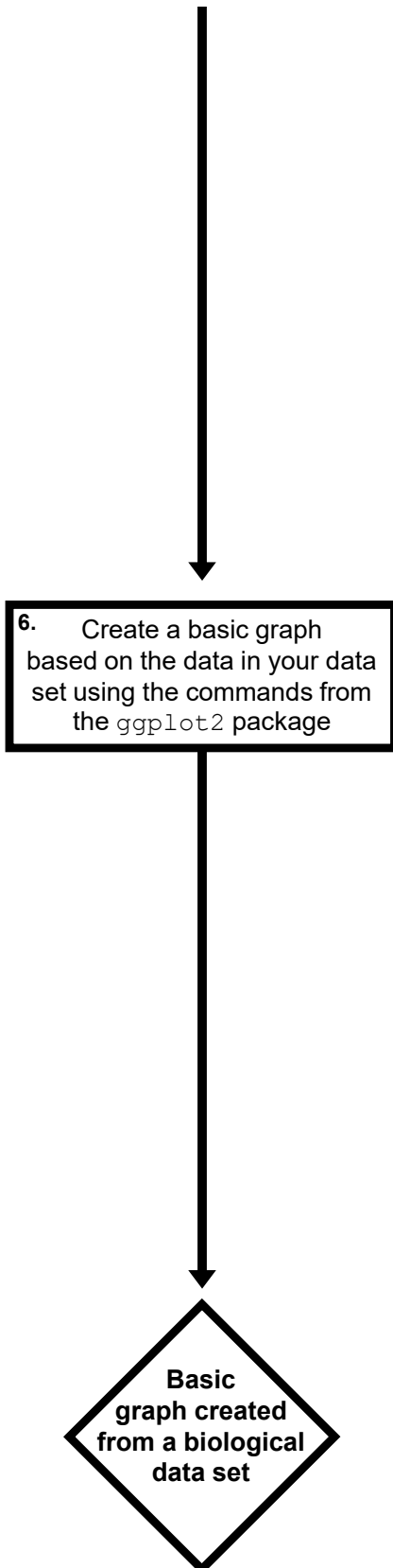


5. Install the `ggplot2` command library into your analysis project

After you have ensured that you have the `ggplot2` package installed in your version of R, you are ready to install the command library it contains into your analysis project. To do this, enter the following command into R:

```
library(ggplot2)
```

This is CODE BLOCK 5 in the document `R_CODE_DATA_VISUALISATION_WORKBOOK.DOC`. This `library` command will load the `ggplot2` command library into your analysis project. **NOTE:** In order to be able to use the commands contained in this library, you will need to run this installation command in each new R project you create. If, when you try to run the R code provided in step 6, you get an error message saying that R cannot find the `ggplot` function, come back to this step and re-run this command.



After you have successfully loaded the `ggplot2` command library into your analysis project, you are ready to use it to create a basic graph from the data set you imported into R in step 2. In this example, this will be a frequency distribution histogram. To do this, enter the following block of code into R:

```
ggplot(data=cetacean_pre_y_sizes,
  aes(x=common_dolphin)) + geom_histogram()
```

This is CODE BLOCK 6 in the document `R_CODE_DATA_VISUALISATION_WORKBOOK.DOC`, and it contains two commands separated by a `+` symbol. These are the `ggplot` command and the `geom_histogram` graphing command. In the `ggplot` command, the `data` argument is used to set the R object containing the data that will be plotted on the graph. In this case, it will be the one called `cetacean_pre_y_sizes` created in step 2 of this exercise. The column of data which will be plotted on the X axis of the resulting graph is set using the `x` argument of the `aes` element of this command. In this case, it is the column called `common_dolphin` in the `cetacean_pre_y_sizes` data set.

The second command in this code block sets the type of graph that will be created from the data specified in the `ggplot` command. In this case, this graphing command will be `geom_histogram`. This means the block of code will create a histogram from the data set and X variable specified in the `ggplot` command. Initially, this `geom_histogram` command will have no elements or arguments in it.

When you run this block of code, you may get a warning message that states: `'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'`. This means that the default number of bins (or bars) for the `geom_histogram` command (which is 30) is not appropriate for your data. For this example, you need to add a `bins` argument to the `geom_histogram` command and then re-run it. To do this, edit the above code so it looks like this (the newly added `bins` argument is highlighted in **bold**):

```
ggplot(data=cetacean_pre_y_sizes,
  aes(x=common_dolphin)) +
  geom_histogram(bins=10)
```

This is CODE BLOCK 7 in the document `R_CODE_DATA_VISUALISATION_WORKBOOK.DOC`, and it adds the argument `bins=10` to the `geom_histogram` command. This means it will create a histogram with approximately 10 bins (or bars) on it. **NOTE:** This argument only provides a target number of bins for your histogram and the actual number will be the closest one that produces an even distribution of bars along the X axis of your graph. Once you have finished editing this code block, you can run it again to create the final version of your first histogram.