EXERCISE 2.1: HOW TO CLASSIFY DATA INTO GROUPS AND CREATE A BAR GRAPH BASED ON THEM:

Bar graphs differ from histograms in one key way. This is that they display information about groups of data rather than for a continuous variable. This means that the data may need to be divided into groups before they can be used to make a bar graph. In this exercise, you will learn how to use R to divide a continuous variable into categories and then use these categories to create a bar graph showing the number of records within a data set which fall into each one. In order to be able to do this, 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 divide into categories and then create a bar graph from.

For this exercise, you will start by dividing data on the prey size preferences of a cetacean species, the common dolphin, into two groups before creating a bar graph based on the counts of data in each one. This will be done using a new data set called `cetacean_prey_sizes_2.csv`. While this contains the same data that were used for the exercises in chapter 3, it differs in its structure. Specifically, all the prey size data are contained in a single column (called `ppsr`), while a second column (called `species`) provides information about which species each individual prey item was recorded from. As a result, before you can start processing the data for common dolphin, you will first need to create a new data set that contains only these data. To process the common dolphin data and create a bar graph based on prey size categories from it, work through the flow diagram that starts at the top of the next page.

NOTE: If you have not already done so for an earlier exercise, you will need to download the `ggplot2` package and install it in your version of R before you start working through the instructions in this flow diagram. If you do not do this, the `library` command used to load the command library from this package into your analysis project in step 6 of this flow diagram will not work. You can check if you already have the `ggplot2` package installed in your version of R using the command `library()`. If you find you need to install the this package, this can be done by entering the following command into R:

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

Data set held in a comma separated values (.CSV) file

For this example, the data set you will use is stored in a file called `cetacean_prey_sizes_2.csv` that is located in the WORKING DIRECTORY folder you created during the introduction to this chapter.

1. Set the WORKING DIRECTORY for your analysis project

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_prey_sizes_2.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_prey_sizes_2 <-
read.table(file="cetacean_prey_sizes_2.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 24 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_prey_sizes_2` 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_prey_sizes_2` 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=";"`.

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

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_prey_sizes_2)
```

This is CODE BLOCK 25 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: `record_no`, `ppsr` and `species`.

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_prey_sizes_2)
```

This is CODE BLOCK 26 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. Create a new R object containing just the subset of data you wish to create a bar graph from

In the data set being used in this example, there are data from a total of four cetacean species (the common dolphin, the harbour porpoise, the pilot whale and the sperm whale). However, for this example, you only wish to create a bar graph from the common dolphin data. This means that before you can do anything else, you will need to create a new R object which contains only the data from common dolphin. In R, this is done using the `subset` command. To do this for the data set being used in this example, enter the following code into R:

```
common_dolphin_ppsr_data <-
subset(cetacean_prey_sizes_2,species==
"common_dolphin")
```

This is CODE BLOCK 27 in the document R_CODE_DATA_VISUALISATION_WORKBOOK.DOC. This command will create a new R object called `common_dolphin_ppsr_data` which will only contain the rows from the `cetacean_prey_sizes_2` data set that have the text `common_dolphin` in the `species` column.

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

```
View(common_dolphin_ppsr_data)
```

This is CODE BLOCK 28 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 it contains the required subset of data. In this example, it should only contain data from common dolphin and not from any of the other species contained in the original data set.

In this example, you will use the `ifelse` command to create your new column containing the categories which will be used for the X axis of your bar graph. This command is found in the `dplyr` package. This means that you will first need to check if you already have this package installed in your version of R by entering the code `library()`. If this package is not listed in the R PACKAGES AVAILABLE window that opens, enter the follow code into R:

```
install.packages("dplyr")
```

This is CODE BLOCK 29 in the document R_CODE_DATA_ VISUALISATION_WORKBOOK.DOC. When you run this command, follow any additional instructions that appear on your screen in order to download and install the specified package. If the `dplyr` package is listed in the R PACKAGES AVAILABLE window, you do not need to re-install it by running the above command.

Once you have ensured that the `dplyr` package is installed in your version of R, you then need to load its command library into your analysis project. This is done by entering following code into R:

```
library(dplyr)
```

This is CODE BLOCK 30 in the document R_CODE_DATA_ VISUALISATION_WORKBOOK.DOC.

5. Create a new column which contains the categories that will be used for the X axis of your bar graph

Now that the required `dplyr` command library has been loaded into your analysis project, you can use the `ifelse` command to create a new column with two categories in it which will then be used as the groups for the X axis of your bar graph. In this command, you provide a logical statement based on a specific column in your data set. You also need to provide two values, one to be entered in the new column for rows where this statement is true, and a second to be used for rows where it is not. To do this for the data set being used in this example, enter the following command into R:

```
common_dolphin_ppsr_data$ppsr_cat=
ifelse(common_dolphin_ppsr_data$ppsr<0.05,
'1','2')
```

This is CODE BLOCK 31 in the document R_CODE_DATA_ VISUALISATION_WORKBOOK.DOC. This command creates a new column called `ppsr_cat` in the data set called `common_dolphin_ppsr_data`. This new column is then filled with values based on the logical statement in the `ifelse` command. In this case, this logical statement is `common_dolphin_ppsr_data$ppsr<0.05`. If this statement is true for any given row in the data set called `common_dolphin_ppsr_data` (i.e. if the value in the `ppsr` column is `<0.05`), it will put a value of `1` (the first value after the `ifelse` logical statement) in the new `ppsr_cat` column. If it is not true (i.e. if the value is equal to or greater than 0.05), the new column for that row will be filled with a value of `2` (the second value after the `ifelse` statement).

6. Load the command library for `ggplot2` package into your analysis project

Before you can make a bar graph based on the categories you have just created, you need to load the `ggplot2` command library into your analysis project. To do this, enter the following command into R:

```
library(ggplot2)
```

This is CODE BLOCK 32 in the document R_CODE_DATA_VISUALISATION_WORKBOOK.DOC.

7. Create your initial bar graph based on the count of records in each category created in step 5

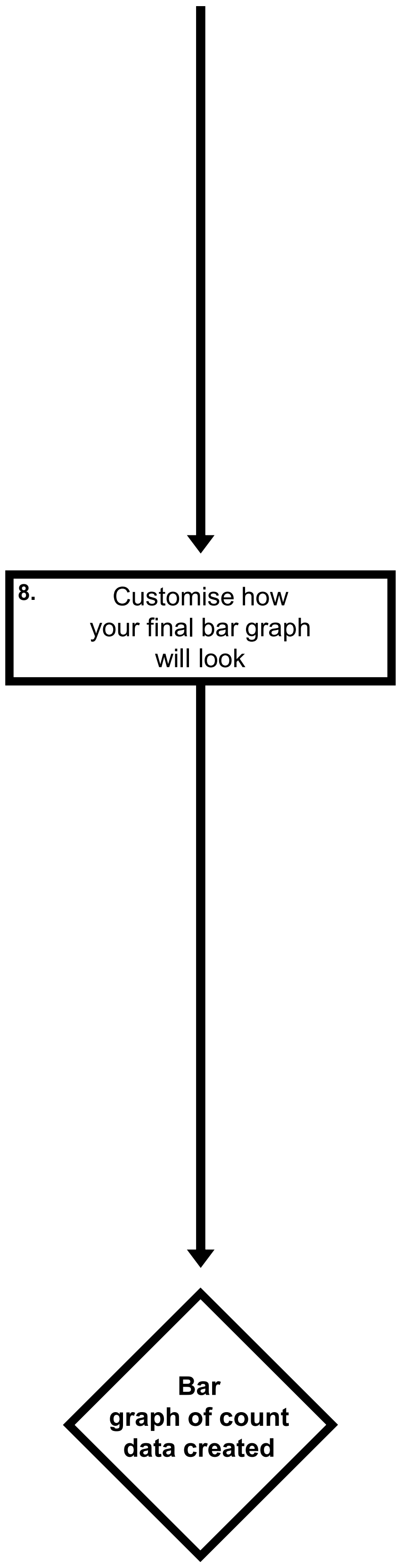
After you have successfully loaded the `ggplot2` command library into your analysis project, you are ready to use it to create your bar graph based on the count of records in each category created in step 5. To do this, enter the following block of code into R:

```
ggplot(data=common_dolphin_ppsr_data,
aes(x=ppsr_cat)) + geom_bar(stat="count") +
scale_x_discrete(labels=c("Small","Large"))
```

This is CODE BLOCK 33 in the document R_CODE_DATA_VISUALISATION_WORKBOOK.DOC, and it contains three commands separated by `+` symbols. These are the `ggplot` command, the `geom_bar` command, and the `scale_x_discrete` command. The `ggplot` command sets the data set which will be used for the graph. This is done using the `data` argument and, in this case, it will be the R object called `common_dolphin_ppsr_data` created in step 4 of this exercise. The column of data that will be plotted on the X axis of the resulting graph is set using the `x` argument of the `aes` element of this `ggplot` command. In this case, it is the column called `ppsr_cat` in the `common_dolphin_ppsr_data` data set created in step 5.

The second command in this code block, `geom_bar`, sets the type of graph that will be created from the data specified in the `ggplot` command. In this case, it will be a bar graph. Within this command, the values to be plotted on the Y axis are set by the `stat` argument. In this case, the argument used is `stat="count"`, meaning that the height of the bar for each category of the X axis will be the count of the number of rows in that category in the data set defined in the `data` argument of the `ggplot` command (in this case, `common_dolphin_ppsr_data`).

The third command is `scale_x_discrete`. This command sets the labels that will be used for each group on the X axis of the resulting bar graph. Without this command, each group would be labelled with the values from the `ppsr_cat` column, in this case, `1` and `2`. However, It is much better to include this command to provide more descriptive labels. In this case, the groups will be labelled as `small` (for the group with a value of `1` in the `ppsr_cat` column) and `large` (for the group with a value of `2` in this column).


8.
Customise how your final bar graph will look
Bar graph of count data created

Once you have created your initial bar graph, and you are happy with how the categories are displayed on it, you can customise how your final graph will look. This can be done by adding a number of new style commands to the block of code used to produce it. To do this, edit the code from step 7 so that it looks like this (the newly added style commands are highlighted in **bold**):

```
ggplot(data=common_dolphin_ppsr_data,
aes(x=ppsr_cat)) + geom_bar(stat="count") +
scale_x_discrete(labels=c("Small",
"Large")) + labs(title="Relative Prey Size
of Common Dolphin",x="Predator-Prey Size
Ratio Category",y="Number of Prey Items") +
ylim(c(0,10000)) + theme_classic()
```

This is CODE BLOCK 34 in the document R_CODE_DATA_VISUALISATION_WORKBOOK.DOC, and it adds three new style commands separated by `+` symbols to the code block used to create your initial bar graph in step 7. These are the `labs` command, the `ylim` command and the `theme_classic` command. The `labs` command sets the labels that will be used tor the title of the graph (using the `title` argument), the X axis (using the `x` argument) and the Y axis (using the `y` argument). The `ylim` command sets the minimum and maximum values that will be displayed on the Y axis of the graph. In this case, these will be `0` for the minimum value and `10,000` for the maximum value. Finally, the `theme_classic` command sets the remaining style elements of the final graph to those of the pre-existing classic theme. Once you have finished editing this code block, you can run it again to create the final version of your bar graph.