Computer Practical 1: Introduction to R and plotting with ggplot2

Dr Alice Risely [a.risely@salford.ac.uk](mailto:a.risely@salford.ac.uk)

Table of Contents

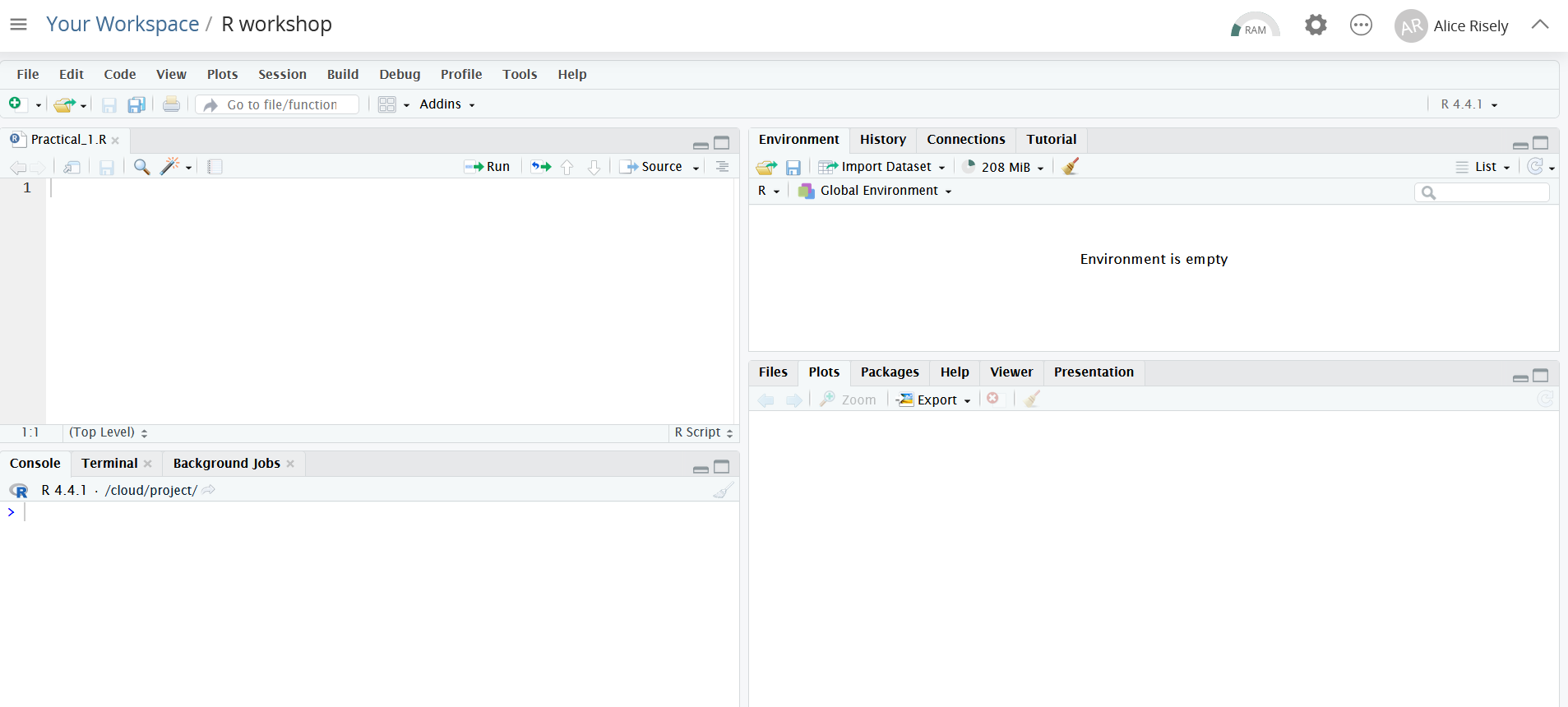
## Learning outcomes

What you will learn in this practical:

* Basic R functionality, including running code, creating data objects, and understanding different data types.
* How to summarise categorical and numerical data.
* How to plot histograms, barplots, boxplots, and scatterplots.
* You have a homework task at the end of the worksheet.

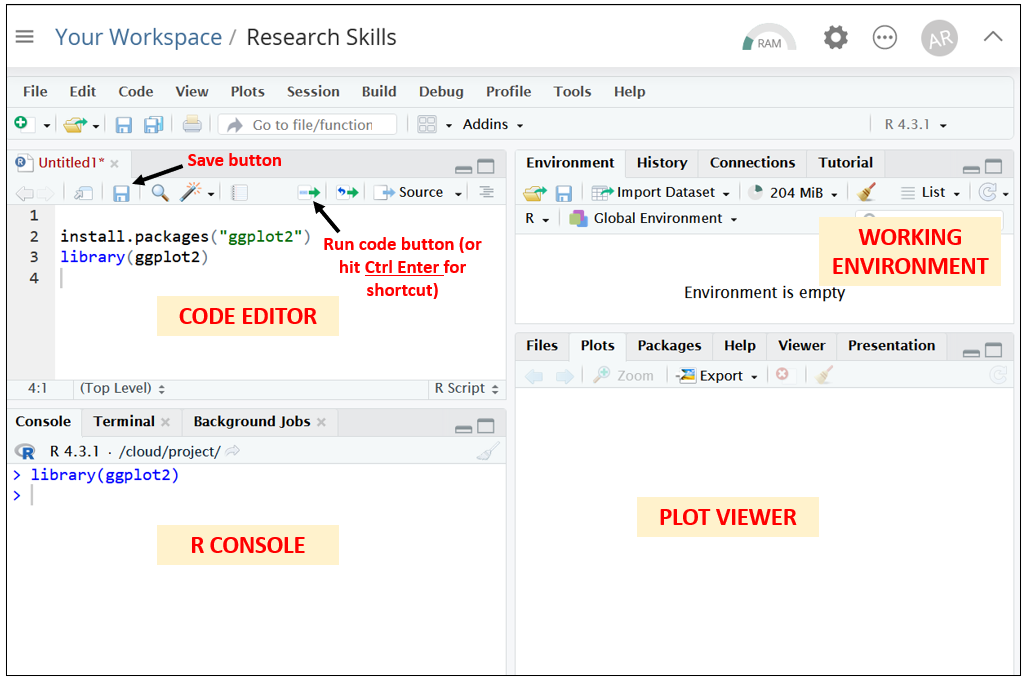
## Preliminary tasks

1. DOWNLOAD this word file from Blackboard, do not just view it. To *download* it, click the “…” next to the file and click ‘Download original file’. **Save this file in an appropriately named folder on your OneDrive (e.g., “MODULE NAME/WEEK X”) so you can access it at home.**
2. READ THROUGH THE WORKSHEET CAREFULLY. If you skip steps, you will not be able to complete the exercises. If you are stuck at an exercise, raise your hand.
3. **Navigate on your web browser to** [**https://posit.cloud/**](https://posit.cloud/)**.** If you have not registered, then register with an email address and password. If you have already registered, then sign in.
4. **Next, if prompted, click “Posit Cloud”, then click the blue “New Project” button on the top right.** Choose “New R Studio Project” on the available options.
5. **Once your project has loaded (it takes a few seconds), click ‘File’ then ‘New file’, then ‘R Script’,** to open a new R script file.
6. **Change the name of your project (currently ‘Untitled Project’) to “R workshop”.** In the following practicals, open this project instead of creating a new one every time.
7. **Press the Save button and save your script as “Practical\_1.R”.** This will save all your code you write today. You should now have a screen that looks like this :



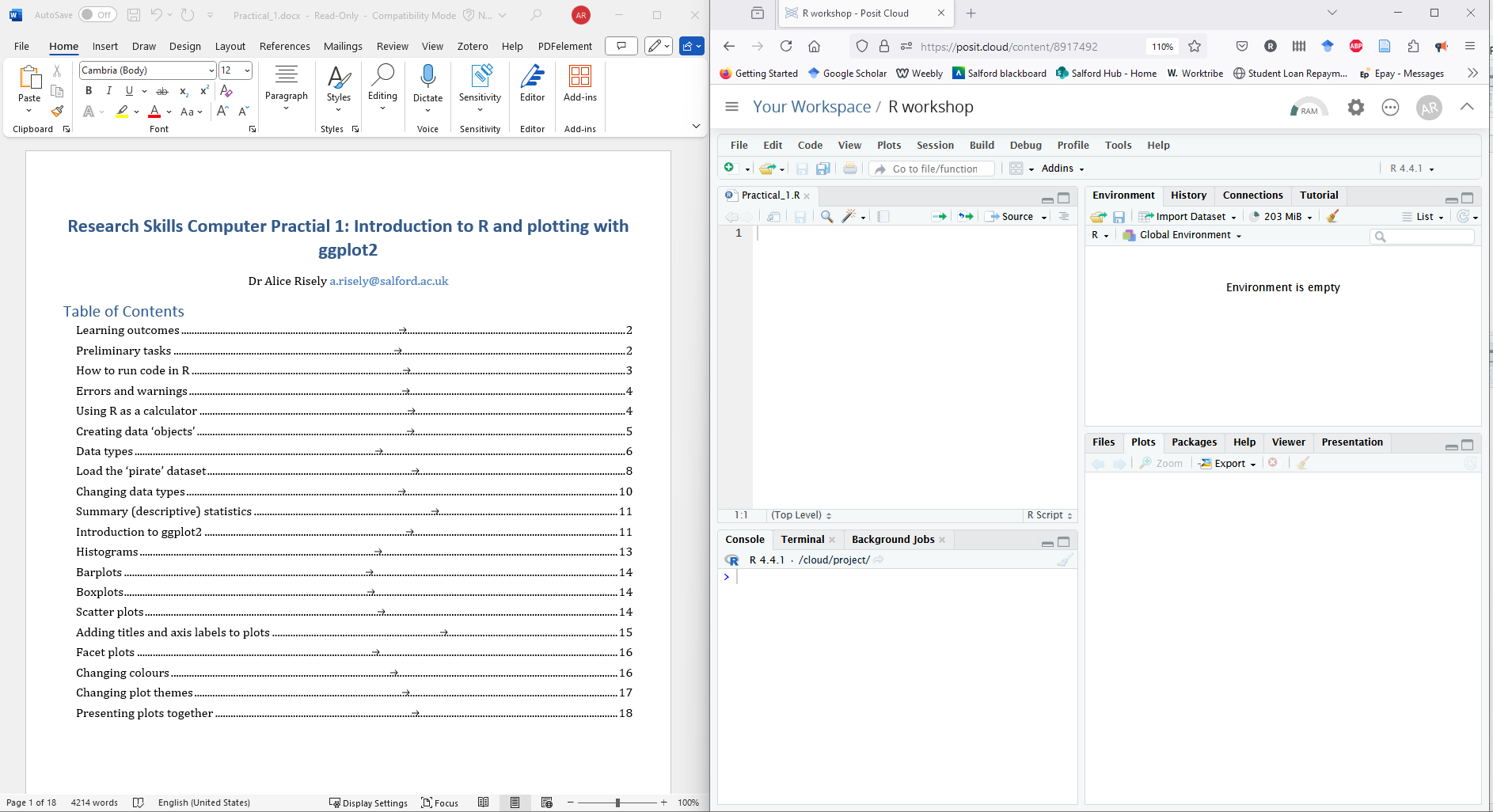
Please note the names of the four R studio windows (**code editor, R console, working environment,** and the **plot viewer**), as these will be referred to in the tutorial.

**Also note where the ‘Run’ button is,** as well as the ‘Save’ button, as you will need to **save your R script** at the end of the tutorial so you can access it again.



**For this tutorial you will need to copy the code contained in the greyed out code blocks into your code editor section in R, and then press the “Run” button with the mouse curser on the line you want to run.**

**TIP:** The easiest way to go about this practical is to have your Word worksheet and R open side by side on a large university computer screen, so that you can read the instructions in Word and work in R at the same time:



## How to run code in R

Copy the code in the greyed out code box below into your **code editor** in R (top left window) and press the ‘Run’ button. This will install the R package ‘ggplot2’ and load it into your environment.

R considers each *numbered line* as a stand-alone line of code. To run a line of code in R, put your mouse cursor on the line you want to run and press **Control and Enter** (this is the most common method). Alternatively, press the ‘Run’ button above the code editor. You can run whole chunks of code by highlighting multiple lines of code and pressing ‘run’. However, for this tutorial please only run one line at a time so you understand what each line is doing.

**Copy the following code into your code editor and then run each line of code by making sure the mouse cursor is on the line you want to run, then press the ‘Run’ button (or Ctrl Enter)**

install.packages("ggplot2") # install ggplot2 if you have not done so already  
  
  
library(ggplot2) # this will load ggplot2 if you havent done so already  
  
  
# The hash lets R know that this is a note, not code. You can run it, but it won't do anything. R just ignores it. You do not need to copy non-coding notes into your R console, they are just there for your information.

You only ever need to install a package once, but you will need to load the package every time you open R.

**Anything after a # key is NOT code,** it is comments/notes. R will ignore any writing after the # key. You do not need to copy out the comments into R.

When you run the code, R will process your command and send it to the **R console** (bottom left window), where you will see both the code and the output.

**NOTE:** Instead of copying your code into the code editor window, you could copy the code directly in the **console** and press Enter. **This will run the code, *but won’t save it*.** Having your code in the code editor section allows you to save your code and edit it.

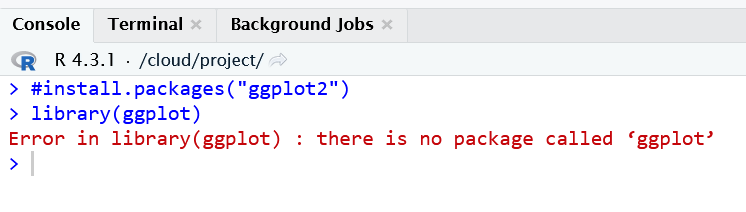
## Errors and warnings

R is fussy. If you make a typo or write code in a format it doesn’t recognise, it will give you an a big red error message. **For example, run the following code:**

library(ggplot)

You should receive an error in the console that says:

Error in library(ggplot) : there is no package called ‘ggplot’



Do not be scared of errors - they are a crucial part of the learning process. They are usually relatively informative. Read the error and try to troubleshoot. In this case, the warning is telling us that we have spelled ‘ggplot’ wrong, so R did not recognise it, it should be *ggplot2*. There is no package called ‘ggplot’.

## Using R as a calculator

R’s most basic functionality is as a calculator.

**Run the following code in your code editor (or directly in the console) to do some maths.**

2+2 # addition  
  
5\*2 # subtraction  
  
10/2 # division  
  
10^3 # to the power of  
  
325 / 13.2  
  
sqrt(9) # use sqrt() function (just like Excel) to calculate the square root of 9

#### **Exercise 1**

Use R to calculate 349 multiplied by 7. What is the answer? **Write all answers to exercises below.**

\*\*\* COPY YOUR ANSWERS HERE \*\*\*  
  
  
  
---

#### **Exercise 2**

What is the square root of 516?

\*\*\* COPY YOUR ANSWERS HERE \*\*\*  
  
  
  
---

## Creating data ‘objects’

You will need to save all sorts of data in R. You can save single numbers or letters, lists of numbers, data frames or matrices.

**To save data, you have to use assignment operators, <- or =.** These instruct R to save your data as an **‘object’** to your working environment. Some data can be extremely simple.

**Run the following code:**

a<- 100 # save the number 100 as an object called 'a'  
a  
  
# You should now have an object called 'a' in your work environment (top right window)  
  
b<- 2 # save the number 2 as an object called 'b'  
b # You should now have an object called 'b' in your work environment  
  
a/b # divide a by b. What's the answer?

Lets say I take a register of my tutor group and I want to record which of my tutees turned up. I want to create a data object that contains the names of my tutees who attended. **Run**

**Run the following code:**

tutees <- c("Jessica", "Chris", "Emma", "Dave", "Jessica", "Dave", "Ahmed") # I am saving a list of names and calling that list 'tutees'  
  
tutees  
  
table(tutees) # I have two Jessica's and two Dave's, and only one Ahmed, Chris, and Emma.

In your work environment on the right, you should now have three objects: ‘a’, ‘b’ and ‘tutees’.

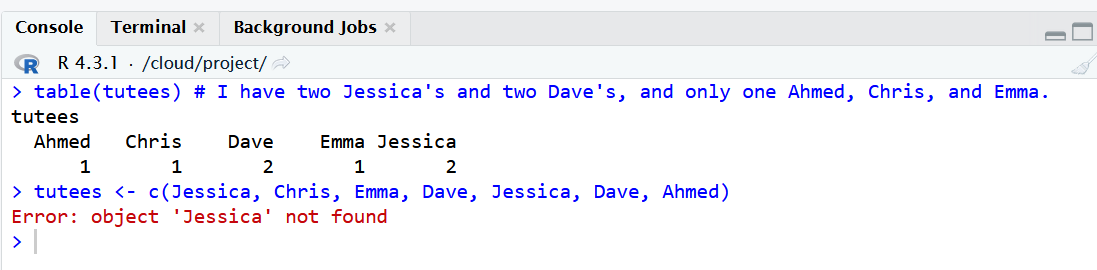
**NOTE 1:** We used two commonly used R **functions** here. We used **c()** to *combine* our data together. We then used **table()** to count how many of each name we have.

**NOTE 2:** Any *free text* in R code must *always* go in quotation marks. This is why we write “Jessica”, and not just Jessica. This tells R that “Jessica” is not a piece of code that it should recognise. Numbers do not need to go in quotation marks, because R knows what numbers are.

**For example, repeat the code but exclude the quotation marks. What happens?**

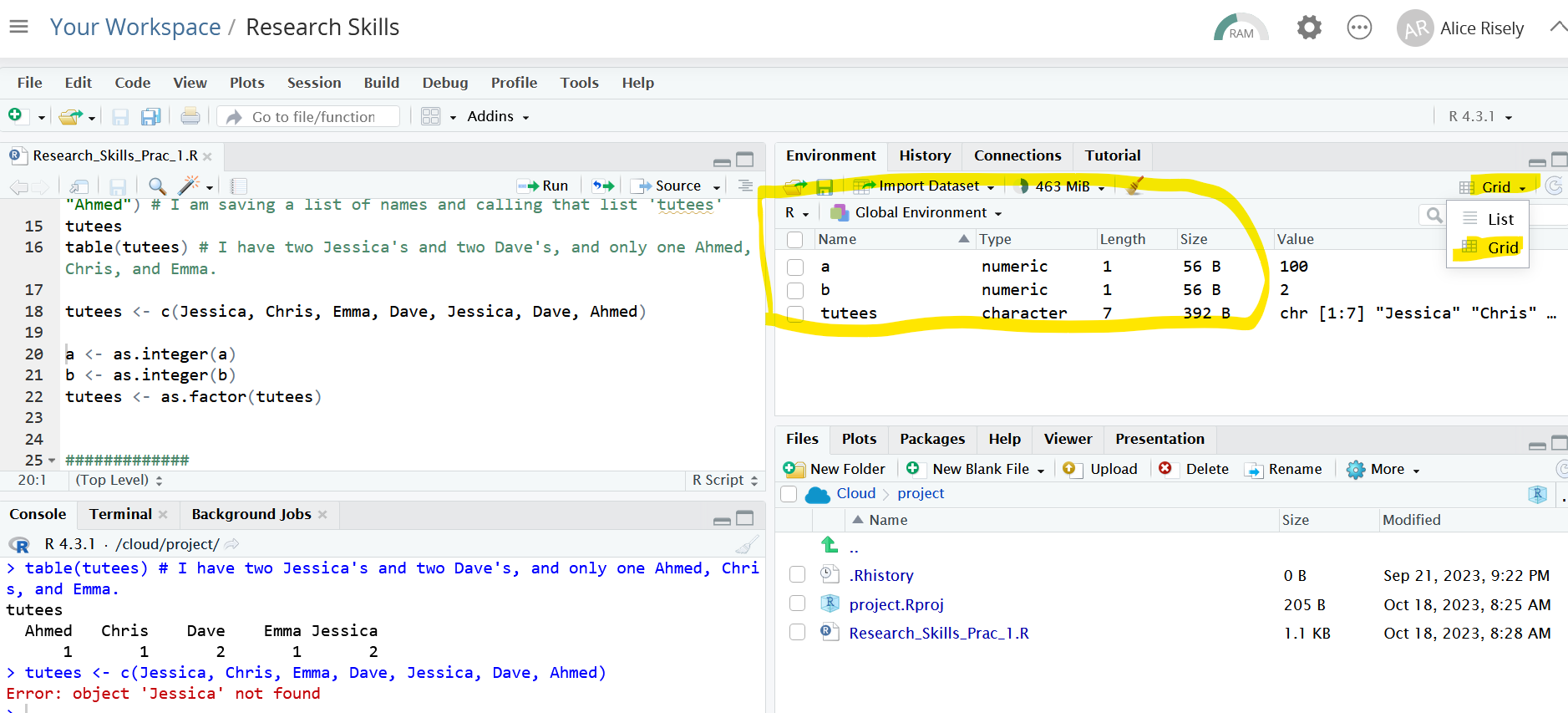
tutees <- c(Jessica, Chris, Emma, Dave, Jessica, Dave, Ahmed)

You should have got an error saying that it couldn’t find ‘Jessica’. R is trying to look for a known function or data object called Jessica, and not finding it. It’s therefore unhappy with you.



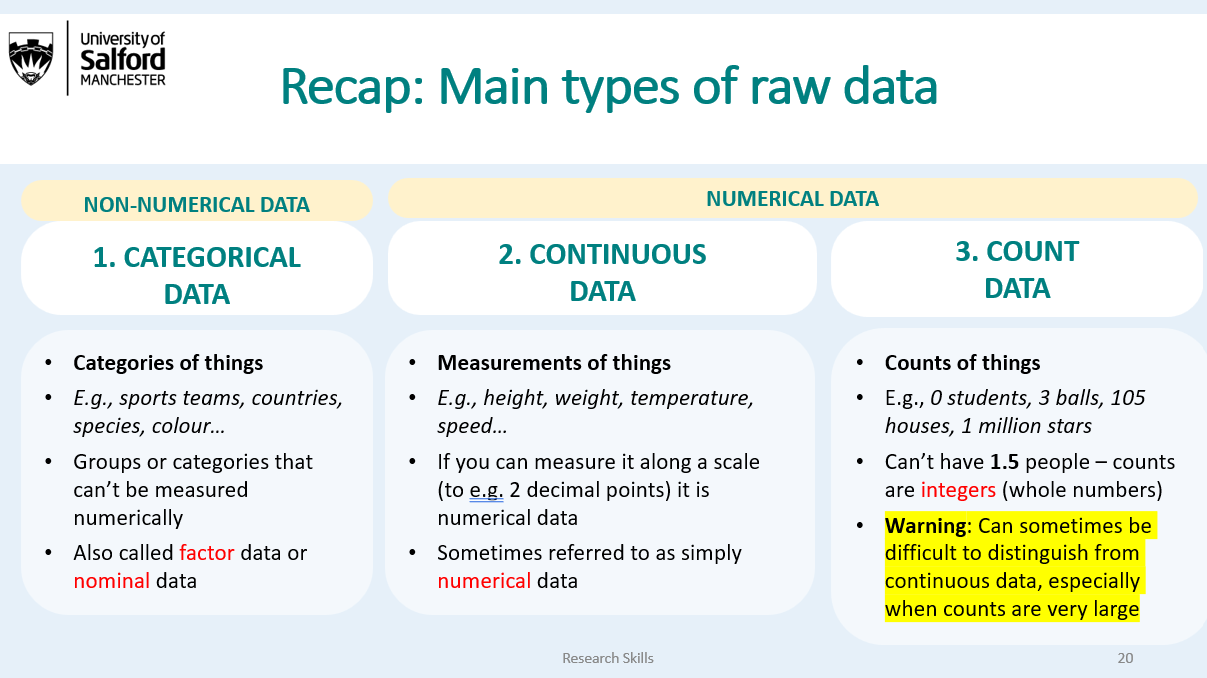
## Data types

In your working environment, you should now see this:



If it doesn’t look like this, make sure the work environment is in ‘grid’ format by clicking on the ‘list’ button and clicking ‘grid’ in the drop down menu.

The work environment gives you a summary of the data you have saved to your environment. The most important information is ‘Type’, which stats for ‘Data type’. This tells you what kind of data R thinks it is. In the lecture, I mentioned that there are different types of data - **numerical** and **non-numerical (categorical),** and numerical data can be further split into **continuous data** and **count data.**



In R (and indeed any other statistical software), it is VERY important to always be explicit about what type of data you have. The three main types of data and how they are called in R are:

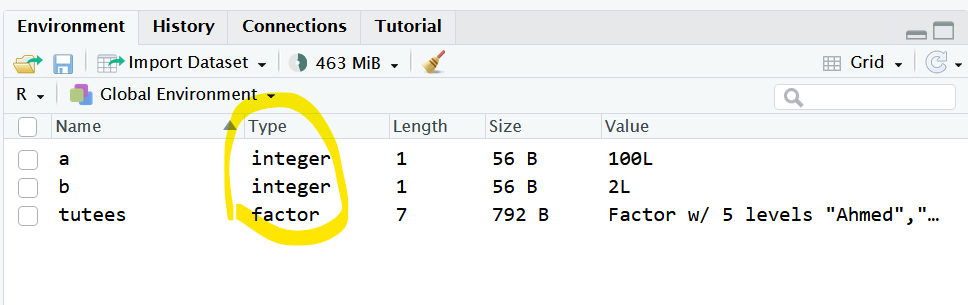
|  |  |  |
| --- | --- | --- |
| Data type | R terminology | R function to specify data type |
| Categorical | FACTOR (“factor”) or CHARACTER (“chr”) | as.factor() or as.character() |
| Continuous | NUMERIC (“num”) | as.numeric() |
| Count | INTEGER (“int”) | as.integer() |

**NOTE 1:** Categorical data in R can be treated as either FACTOR data or CHARACTER data. Often, it doesn’t matter. However, for analyses you will usually want your categorical data treated as a FACTOR.

Look at your work environment again. R has assumed *a* and *b* are continuous numeric data, and *tutees* is character (categorical) data. Lets specify that we want *a* and *b* to in count data, and *tutees* to be categorical data.

a <- as.integer(a) # assign a into count data  
b <- as.integer(b) # assign b as count data  
tutees <- as.factor(tutees) # assign tutees as categorical data

You should now see that R has changed the data type of your three objects in your work environment.



## Load the ‘pirate’ dataset

Now we have covered some of the basics of R, time to dig into some data!

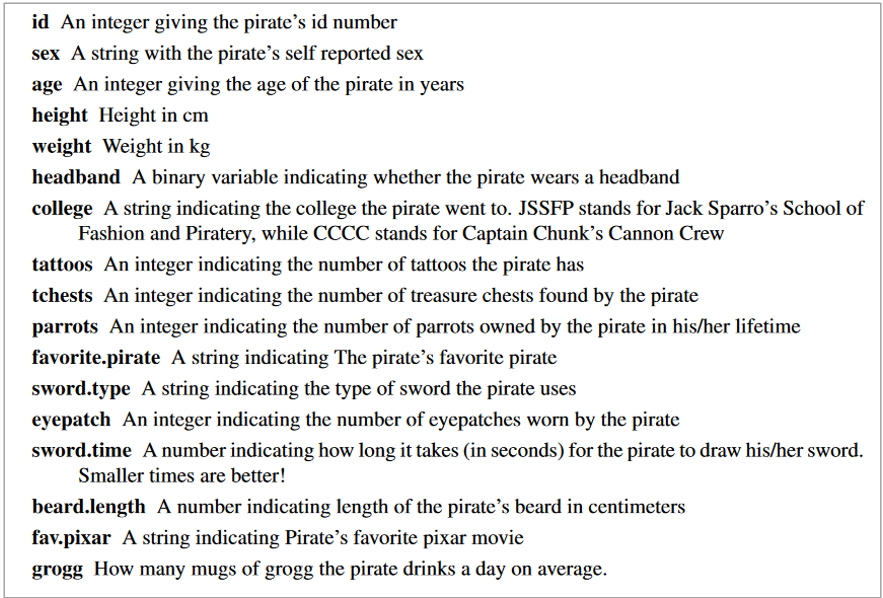
First, lets install and load an example dataset about pirates, which is available from the R package “yarrr”. This dataset is called ‘*pirates*’ and contains the results of a survey of 1000 pirates, including info on their sex, age, height, how many tattoos they have, number of parrots they own, whether they have an eye patch, who their favourite pirate is, etc.

From now on, make sure all your code is copied into the code editor window, and not directly into the console. This is so you can save it and return to it next week.

install.packages("yarrr") # install package

library(yarrr) # load package  
  
data("pirates") # extract the data frame 'pirates' and save into working env  
  
head(pirates) # returns the first six lines of the dataset  
  
dim(pirates) # dimensions of the dataset. It has 1000 rows and 17 columns  
  
names(pirates) # column names. These are important for plotting!

The dataset contains the following data:



To view your dataset in a separate window, you use the function **View()**. This will add a new tab on your code editor window and you will need to click back on your R script to continue coding. In View pane, you can click on the column names to sort the rows, just as you can in Excel.

View(pirates) # View the dataset in a seperate window

**NOTE 1:** Have you noticed that none of the column names contain spaces? **When naming columns for your own data in Excel, never put spaces in the column names**, as this will confuse all statistical software at the data analysis stage.

In this dataset, all column names are lower case and words are separated by full stop. T*his will be important to remember when plotting.*

**To check the names of your columns (“variables”) you can either look at your dataset using the View() function, or use the names() function.**

names(pirates)

You may want to look at data from a specific column. To call a specific column, you use the the **$** sign using the structure *data$column.*

pirates$id # Pirate ID  
  
pirates$age # Shows all data from the Age column  
  
pirates$favorite.pirate # Calls all data from the favourite.pirate column  
  
head(pirates$fav.pixar) # first six rows of the fav.pixar column.

## Changing data types

The first thing you should do when you import a new dataset is to check how R has assigned the data type of each column. To do this you use the function str() (which stands for *structure*).

str(pirates) # show the data types R has assigned for each column

Do you think these data assignments are correct? You will need to change the data types for some of the columns.

At least one column is clearly wrong in its assigned data types. This is pirate ID. PirateID is down as numeric data, but it is actually catagorical (a pirate’s identity cannot be treated as continuous or count data). However - many other columns are also incorrectly categorised (e.g., is number of parrots numerical or count data?).

The code for converting pirate ID into categorical data and converting number of tattoos into count data has been provided (run this code).

pirates$id <- as.factor(pirates$id) # assign pirate ID as categorical data  
  
pirates$tattoos <- as.integer(pirates$tattoos) # assign number of tattoos as count data  
  
str(pirates) # check the data type for these variables has changed

#### **Exercise 3**

Convert the columns for number of parrots and treasure chests (*tchests*) into count data, and convert the column for sword type into a factor (i.e., categorical data).

You will need the functions **as.integer()** and **as.factor(). Once you have worked it out, copy the correct code below:**

\*\*\* COPY YOUR CODE HERE \*\*\*  
  
  
  
---

## Summary (descriptive) statistics

Lets have a look at some summary statistics. To get summary statistics for numerical data, you use the function summary(). To get summary statistics for categorical data, you use the function table(). This is because categorical data cannot have a mean or maximum.

Correct functions to summarise different data types:

**Numerical data = *summary()***

**Categorical data = *table()***

I’m curious what the mean (average) number of tattoos pirates have. I also want to know the most number of tattoos recorded was!

mean(pirates$tattoos) # What is the mean number of tattoos?  
  
summary(pirates$tattoos)# what is the maximum number of tattoos?

I’m also curious about how many pirates really wear eyepatches. I can get a summary how many answered yes and no using the function *table().*

table(pirates$eyepatch ) # how many pirates say they wear eyepatches compared to those who don't?

#### **Exercise 4**

Revise the code above to answer the following questions:

**1) What is the average, minimum, and maximum ages of the surveyed pirates?**

\*\*\* COPY YOUR ANSWERS HERE \*\*\*  
  
  
  
---

**2) Out of all the favourite pirates, who is the most popular? (Hint: use the table() function).**

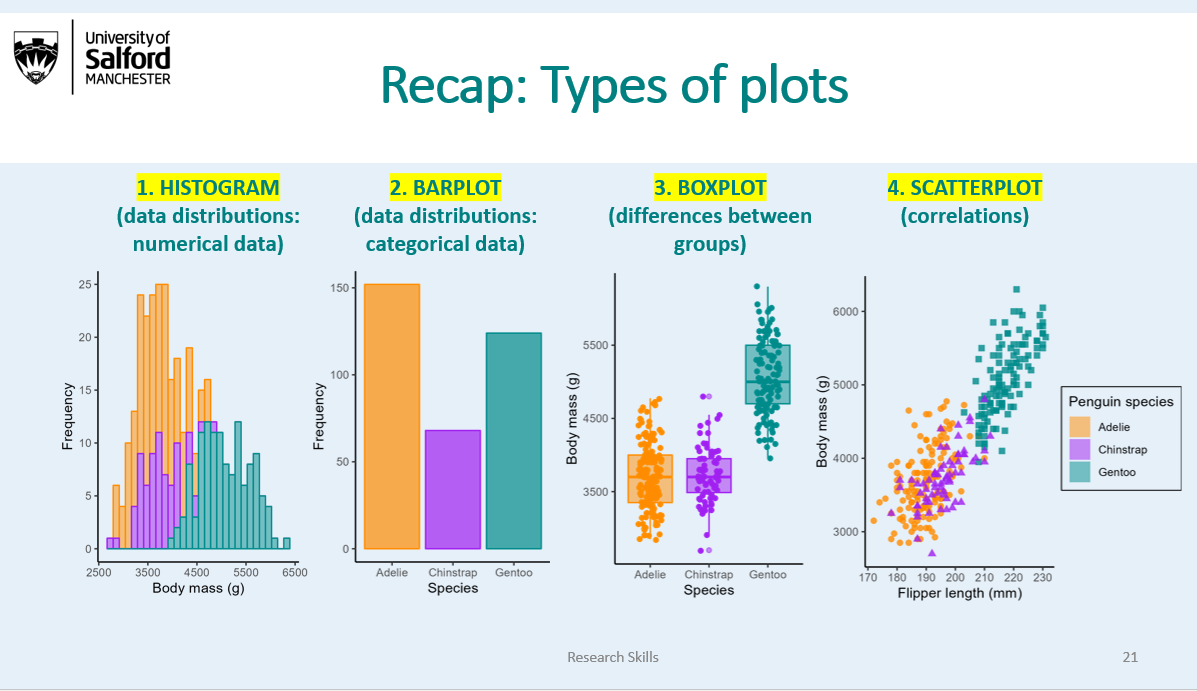
\*\*\* COPY YOUR ANSWERS HERE \*\*\*  
  
  
  
---

**3) What are the four kinds of ‘swords’ that the pirates use? Which is the most common?**

\*\*\* COPY YOUR ANSWERS HERE \*\*\*  
  
  
  
---

## Introduction to plotting with ggplot2

We have gone over the R basics and learned to summarise data. Now it is time for the fun bit: plotting! Remember, there are four main types of plots:



By far the most popular package for plotting in R is by using the *ggplot2* package. *ggplot2* uses a layering system to add layers of aesthetics to your plot. **Each layer is added using a + sign.**

Lets jump straight in and make these four plots, and I will then explain them in more detail. Run the code below.

library(ggplot2) # make sure ggplot2 is loaded int your environment  
  
  
ggplot(data = pirates, aes(x = height)) + geom\_histogram() # histogram  
  
ggplot(data = pirates, aes(x = sex))+ geom\_bar() # barplot  
  
ggplot(data = pirates, aes(x = sex, y = weight)) + geom\_boxplot() # boxplot  
  
ggplot(data = pirates, aes(x = height, y = weight)) + geom\_point() # scatterplot

Lets break this code down a little bit. First, lets run the first part of the code (the first ‘layer’ without the bit after the + sign (the second ‘layer’).

ggplot(data = pirates, aes(x = height))

That looks weird right? There is no data shown. You have told ggplot2 to make a plot with height on the x axis, using the dataset pirates, but you havent told it what kind of plot you want. In the next line, you need to add one of the following layers depending on what type of plot you want.

|  |  |  |  |
| --- | --- | --- | --- |
| Plot type | ggplot layer syntax | Data x axis | Data y axis |
| Histogram | **+geom\_histogram()** | Continuous/Count | None |
| Bar plot | **+geom\_bar()** | Categorical | None |
| Boxplot | **+geom\_boxplot()** | Categorical | Continuous/Count |
| Scatter plot | **+geom\_point()** | Continuous/Count | Continuous/Count |

Now lets add the layer to tell R that we want to plot a histogram by adding “+ geom\_histogram()”.

ggplot(data = pirates, aes(x = height)) + geom\_histogram()

## Histograms

Histograms are probably the most important plot to know how to make, as they give you important information on the distribution of your *numerical* data. This will become very important later in the course when we are choosing statistical tests.

When making histograms, you only need to specify the x axis. This is because the y axis is not a column in your dataset, but represents a summary of your x axis variable.

Lets have a look at the distribution of parrot ownership in pirates. Note, you dont have to write ’data = pirates“, but just simply put”pirates".

ggplot(data = pirates, aes(x = parrots))+geom\_histogram()

As you can see from the histogram, most pirates have between 0 and 3 parrots. However, there are a small number of pirates who love parrots so much that they have over 10 parrots. One even has over 25 parrots! This distribution is definitely not a normal (bell-curve) distribution, it is heavily right skewed! In fact, it is a perfect example of count data, which often tends to be right skewed.

#### **Exercise 5**

Make a histogram showing the distribution of tattoos across pirates. Copy in your plot below. To do this, click ‘Export’ and ‘Copy to clipboard’ in the menu above your plot.

\*\*\* COPY YOUR PLOT HERE \*\*\*  
  
  
  
---

**NOTE 1:** *ggplot2* syntax involves connecting multiple plotting layers with a + sign. You could put all layers on the same line, but most R users put each layer on a separate line to keep it tidy. However, because R treats every line as a stand alone piece of code, the + sign tells R that you’re not finished yet and it should move on to the next line. ***Therefore, make sure the + goes at the end of the line, not at the beginning of the line below.***Otherwise, you will get an error.

ggplot(pirates, aes(x = parrots))  
   
 + geom\_histogram()

## Barplots

Barplots are the equivalent of histograms but for categorical data: they show frequencies of categorical variables (i.e., how many observations do you have in your data per category?). As with histograms, you also don’t need a y variable. For example, they are good for showing how many pirates wear headbands compared to those who don’t:

ggplot(pirates, aes(x = headband))+geom\_bar()

#### **Exercise 6**

Make a barplot showing the distribution of college attendance by pirates.

\*\*\* COPY YOUR PLOT HERE \*\*\*  
  
  
  
---

## Boxplots

Boxplots show you how a continuous variable differs between categories. They are probably the most common plot in scientific research. Lets have a look at beard length (*continuous data*) and self-reported sex (*categorical data*) in pirates.

ggplot(pirates, aes( x = sex, y = beard.length))+  
 geom\_boxplot()

#### **Exercise 7**

Make a boxplot that compares how long it takes to draw ones sword (*sword.time -* a continuous variable) across different sword types (*sword.type -* a categorical variable).

\*\*\* COPY PLOT HERE \*\*\*  
  
  
  
---

## Scatter plots

Scatterplots are for looking at the relationship between two numerical variables. Because height and weight are numerical, the appropriate plot is a scatter plot. In general you want the variable that is being influenced on the y axis. In this case, height influences weight, so weight should go on the y axis. Weight does not influence height!

ggplot(data = pirates, aes(x = height, y = weight))+ geom\_point() # scatter plot

#### **Exercise 8**

Make scatter plot showing the relationship between age and number of treasure chests (*tchests*) found by pirates. Because age influences number of tchests found, tchests should go on the y axis.

\*\*\* COPY PLOT HERE \*\*\*  
  
  
  
---

## Adding titles and axis labels to plots

You can add titles and axis labels to the plot by additional additional layers to your code. Titles and labels are free text, which as mentioned above always needs to go in quotation marks.

Below is a table indicating the code you need to do this.

|  |  |  |
| --- | --- | --- |
| What do you want to do? | ggplot2 layer | Example |
| Add a title | +ggtitle() | **+ggtitle(“My title”)** |
| Change the x axis label | +xlab() | **+xlab(“My x label”)** |
| Change the y axis label | +ylab() | **+ylab(“My y label”)** |

Now lets make some plots with titles and axis labels.

# add a title  
  
ggplot(data = pirates, aes(x = height, y = weight))+ # data and axes  
 geom\_point()+ # plot type  
 ggtitle("Height and weight of pirates") # add title  
  
  
# add an x-axis label  
ggplot(data = pirates, aes(x = height, y = weight))+ # data and axes  
 geom\_point()+ # plot type  
 xlab("Height (cm)") # add x axis label  
  
  
# add a y-axis label  
ggplot(data = pirates, aes(x = height, y = weight))+ # data and axes  
 geom\_point()+ # plot type  
 ylab("Weight (kg)") # add y axis label  
  
  
# put them all together  
ggplot(data = pirates, aes(x = height, y = weight))+ # data and axes  
 geom\_point()+ # plot type  
 ggtitle("Height and weight of pirates")+ # add title  
 xlab("Height (cm)")+ # add x axis label  
 ylab("Weight (kg)") # add y axis label

#### **Exercise 9**

Make a boxplot comparing pirate height across different sexes. Add a title (“Comparison of height across pirate sex”) and an x axis label (“Self-reported sex”) and a y axis label (“Height (cm)”).

\*\* COPY YOUR PLOT BELOW\*\*  
  
  
---

## Changing colours

To change the colours of plots, you need to use the intructions “colour” or “fill”, depending what type of plot you have. Colours always have to be in quotation marks, and R recognises the names of most standard colours. Lets make some colourful histograms.

# histogram  
  
# change the colour inside of the bars  
  
ggplot(pirates, aes(x = height))+  
 geom\_histogram(fill = "yellow")  
  
# change the border colour  
  
ggplot(pirates, aes(x = height))+  
 geom\_histogram(colour = "yellow")  
  
# change border colour and fill colour  
  
ggplot(pirates, aes(x = height))+  
 geom\_histogram(fill = "yellow", colour = "black")

Now lets make some colourful barplots.

ggplot(pirates, aes(x = sex, y = height))+  
 geom\_boxplot(fill = "lightblue", colour = "navyblue")  
  
ggplot(pirates, aes(x = sex, y = height))+  
 geom\_boxplot(fill = "deeppink", colour = "pink")

Now lets make some colourful scatterplots. However, note you cant use ‘fill’ on scatterplots - only color!

ggplot(data = pirates, aes(x = height, y = weight))+ # data and axes  
 geom\_point(colour = "red") # plot type  
  
  
# instead of 'colour', you can also use just 'col' for short  
  
ggplot(data = pirates, aes(x = height, y = weight))+ # data and axes  
 geom\_point(col = "gold") # plot type

A non-exhaustive list of colours you can use can be found here: <https://r-graph-gallery.com/42-colors-names.html>.

## Save your scripts

Before you go home, make sure you save your R scripts by pressing the ‘save’ button. Go to the instructions at the start of the worksheet. If everything is saved, you can access R and the work you have done so far from your laptop.

## Homework

For your homework, you need to finish off this practical in your own time, and in addition do Exercises 10 and 11 below. This will ensure you understand how to make basic plots prior to next week’s computer practical. **You can access your R studio cloud profile from your laptop.**

For your homework, we will use another example dataset, this one on penguin morphology in the Palmer Archipeligo in Antarctica. This dataset is contained within the package palmerpenguins, which you will download and load in order to access the dataset.

install.packages("palmerpenguins") # install package  
  
library(palmerpenguins) # load package  
  
?palmerpenguins # The ? before a package or function will bring up a description of the the package or function. For functions, the help page provides examples on how the function works.   
  
data(penguins) # load the dataset into your environment  
  
View(penguins) # check out the dataset  
  
str(penguins) # check out variables contained in the dataset

#### Exercise 10

1. Summarise the number of observations the data contains per species (i.e., how many penguins in the dataset are Adelie, how many are Chinstrap, and how many are Gentoo?), and b) calculate the mean body weight across all penguins.

* \*\*\* COPY YOUR ANSWERS HERE \*\*\*

#### Exercise 11

Make a plot that shows the difference in body mass between different penguin species. Change the y axis label to “Body mass (g)” and the x axis label to “Penguin species”. Add the title “Body weight across penguin species”. Colour the boxes dark turquoise (“darkturquoise”).

\*\*\* COPY YOUR PLOT HERE \*\*\*  
  
  
  
---

## Optional: Splitting plots by group (facetting)

If you want to split a plot by a third variable, you can use the layer ***facet\_wrap()***. Below is an example:

ggplot(data = pirates, aes(x = height, y = weight))+   
 geom\_point() + # scatter plot  
 facet\_wrap(~sex) # Seperate the plots by the variable 'sex'

## Optional: Changing plot themes

Personally, I don’t love the grey backgound of default ggplot plots. But you can change the theme to change the overall look of the plot. You can do this by addition one of the following layers

* theme\_bw()
* theme\_minimal()
* theme\_dark()
* theme\_classic()

Lets try them out. Which do you like best?

ggplot(data = pirates, aes(x = height, y = weight))+   
 geom\_point(col = "gold") +  
 theme\_bw()  
  
ggplot(data = pirates, aes(x = height, y = weight))+   
 geom\_point(col = "gold") +  
 theme\_minimal()  
  
ggplot(data = pirates, aes(x = height, y = weight))+   
 geom\_point(col = "gold") +  
 theme\_dark()  
  
ggplot(data = pirates, aes(x = height, y = weight))+   
 geom\_point(col = "gold") +  
 theme\_classic()

## Optional: Presenting plots together

There are a number of R packages that are designed to make composite plots. One of the most common ones is *ggpubr.*

To make composite plots, you need to save each plot you want to present together as an object. Below I have made four plots (a bar plot, a histogram, a scatter plot, and a boxplot) and presented them together using the function ggarrange().

install.packages("ggpubr") # install package  
library(ggpubr) # load package  
  
# create the four plots we have been practicing above  
  
barplot <- ggplot(pirates, aes(x = sex))+  
 geom\_bar(fill = "lightblue")+  
 theme\_dark()  
  
histogram <- ggplot(pirates, aes(x = height))+  
 geom\_histogram(fill = "lightblue", col = "black")+  
 theme\_dark()  
  
scatterplot<- ggplot(pirates, aes(y = height, x = weight))+  
 geom\_point(col = "lightblue")+  
 theme\_dark()  
  
boxplot<- ggplot(pirates, aes(x = sex, y = height))+  
 geom\_boxplot(fill = "lightblue")+  
 theme\_dark()  
  
# make a composite plot and label a-d.  
  
ggpubr::ggarrange(barplot, histogram, scatterplot, boxplot, labels = c("a)", "b)", "c)", "d)"))