**Intro to R Tutorial**

We will be going through the “Getting Started with R and R Studio Tutorial” provided by *The Coding Club.* Online link to tutorial here: <https://ourcodingclub.github.io/tutorials/intro-to-r/>

**Tutorial aims:**

1. Understand what are R and R Studio
2. Develop the good habit of working with scripts
3. Learn to import data in R
4. Learn to manipulate R objects like vectors and data frames
5. Make a simple plot

**Step 1: Download R and R Studio**

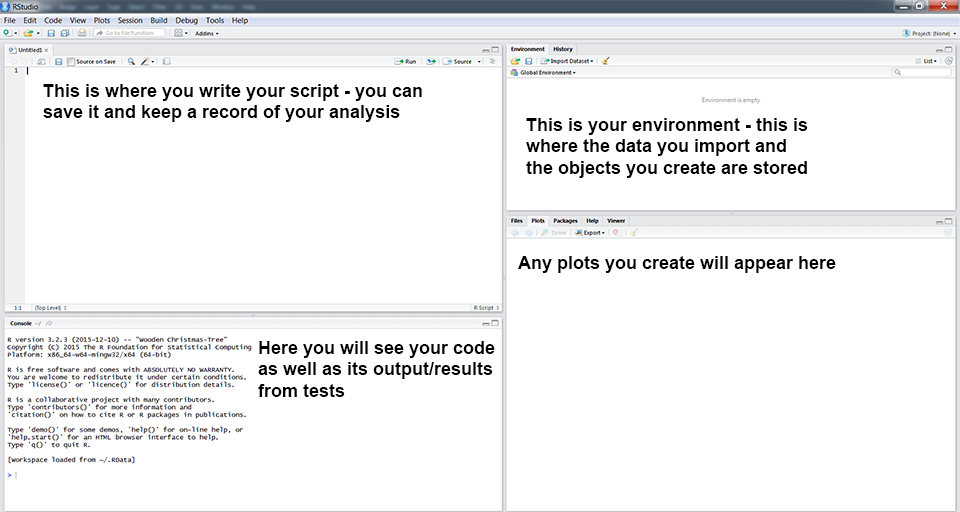
R itself does not have a graphical interface, but most people interact with R through graphical platforms that provide extra functionality. We will be using a program called RStudio as a graphical front-end to R, so that we can access our scripts and data, find help, and preview plots and outputs all in one place.

You can download R from [CRAN (The Comprehensive R Archive Network)](https://cran.r-project.org/). Select the link appropriate for your operating system.

Then, download RStudio from  [the RStudio website](https://www.rstudio.com/products/RStudio/) (select the free open source desktop version).

If you are using a Mac, in addition to R and RStudio, you need to download XQuartz ([available here](https://www.xquartz.org/)).

Open RStudio. Click on “File/New File/R script”.



**Step 2: Begin to Write Script**

For now, start by recording who is writing, the date, and the main goal - in our case, determining how many species from different taxa have been recorded in Edinburgh. Here’s an example, which you can copy, paste and edit into your new script:

*# Introduction to R Tutorial - R Basics*

*# Learning how to import and explore data, and make graphs about Edinburgh's biodiversity*

*# Written by Gergana Daskalova 06/11/2016 University of Edinburgh*

The next few lines of code usually load the packages you will be needing for your analysis. A **package** is a bundle of commands that can be loaded into R to provide extra functionality. To install a package, type install.packages("package-name"). You only need to install packages once, so in this case you can type directly in the console box, rather than saving the line in your script and re-installing the package every time.

Once installed, you just need to load the packages using library(package-name).

install.packages("dplyr")

library(dplyr)

*# There are quotation marks when installing a package, but not when loading it*

*# Hashtags let you add useful notes to your code!*

The next lines of code should define your **working directory**. This is a folder on your computer where R will look for data, save your plots, etc. To make your workflow easier, it is good practice to save everything related to one project in the same place.

Save your script and all the data for this tutorial in a folder called “Intro\_to\_R”. (It is good practice to avoid spaces in file names as it can sometimes confuse R.)

To find out where your working directory is now, run the code getwd(). If you want to change it, you can use setwd().

setwd("C:/User/CC-1-RBasics-master")

*# This is an example filepath, alter to your own filepath*

**Step 3: Import and Check Data**

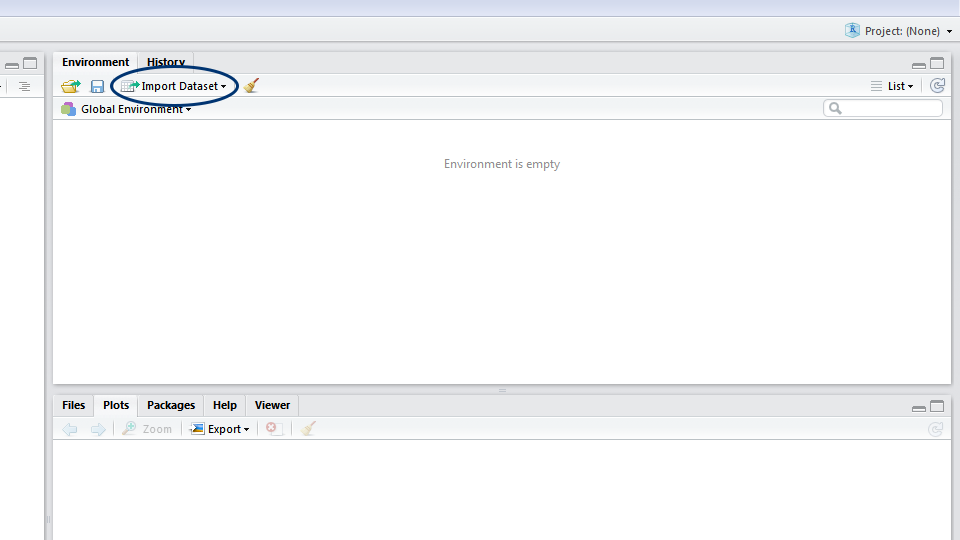
This tutorial uses a publicly available dataset of occurrence records for many animal, plant and fungi species. First, you will need to download the data. Go to your blackboard course site 🡪 Lab modules 🡪 Intro to R 🡪 download all files in the folder titled “data”

In RStudio, you can either click on the *Import dataset* button and navigate to where you have saved your file, or use the read.csv() command. If you use the button, a window will pop up previewing your data. Make sure that next to *Heading* you have selected *Yes* (this tells R to treat the first row of your data as the column names) and click *Import*. In the console, you will see the code for your import, which includes the file path - it’s a good idea to copy this code into your script, so that for future reference you know where your dataset came from. R works best with .csv (comma separated values) files. If you entered your data in Excel, you would need to click on *Save as* and select csv as the file extension. When entering data in Excel, don’t put any spaces in your row names, as they will confuse R later (e.g. go for something like height\_meters rather than height (m)

didiv <- read.csv("C:/Users/user/Desktop/Intro\_to\_R/edidiv.csv")

*# This is the file path based on where I saved the data, your*

*filepath will be different*



A really important step is to check that your data was imported without any mistakes. If you go straight into analysis, you risk later finding out that R didn’t read your data correctly and having to re-do it, or worse, analysing wrong data without noticing. str(object.name) is a great command that shows the structure of your data

head(edidiv) *# Displays the first few rows*

tail(edidiv) *# Displays the last rows*

str(edidiv) *# Tells you whether the variables are*

*continuous, integers, categorical or*

*characters*

The  taxonGroup variable shows as a character variable, but it should be a factor (categorical variable), so we’ll force it to be one. When you want to access just one column of a data frame, you combine the variable name to the object name with a dollar $sign. This syntax lets you see, modify, and/or reassign this variable.

head(edidiv$taxonGroup) *# Displays the first few rows of this*

*column only*

class(edidiv$taxonGroup) *# Tells you what type of variable we're dealing with: it's character now but we want it to be a factor*

edidiv$taxonGroup <- as.factor(edidiv$taxonGroup) *#* Transforms the character values in the taxonGroup column from the edidiv object to factors.

**Step 4: Calculate Species Richness**

Species richness is simply the total number of different species in a given place or group. To know how many bird, plant, mammal, etc. species we have in Edinburgh, we first need to split edidiv into multiple objects, each containing rows for only one taxonomic group. We do this with the useful filter()function from the dplyr package.

Beetle <- filter(edidiv, taxonGroup == "Beetle")

*# The first argument of the function is the data frame, the second argument is the condition you want to filter on. Because we only want the beetles here, we say: the variable taxonGroup MUST BE EXACTLY (==) Beetle - drop everything else from the dataset. (R is case-sensitive so it's important to watch your spelling! "beetle" or "Beetles" would not have worked here.)*

Bird <- filter(edidiv, taxonGroup == "Bird") *# We do the same with birds.*

*# You can create the objects for the remaining taxa. If you need to remind yourself of the names and spellings, type summary(edidiv$taxonGroup)*

**You need to do these steps for ALL of the taxa in the data, here we have given examples for the first two.**

Once you have created objects for each taxon, we can calculate species richness, i.e. the number of *different* species in each group. For this, we will nest two functions together: unique(), which identifies different species, and length(), which counts them.

a <- length(unique(Beetle$taxonName))

b <- length(unique(Bird$taxonName))

*# You can choose whatever names you want for your objects, here I*

*used a, b, c, d...*

**Again, calculate species richness for the other taxa in the dataset.**

**Step 5: Create a vector and plot**

Combine all species richness values into a  **vector**. A vector is another type of R object that stores values. As opposed to a data frame, which has two dimensions (rows and columns), a vector only has one. When you call a column of a data frame like we did earlier with edidiv$taxonGroup, you are essentially producing a vector - but you can also create them from scratch.

We do this using the c() function (c stands for chain(). We can also add labels with the names()function, so that the values are not coming out of the blue.

biodiv <- c(a,b,c,d,e,f,g,h,i,j,k) *# We are chaining together all the values; pay attention to the object names you have calculated and their order*

names(biodiv) <- c("Beetle",

"Bird",

"Butterfly",

"Dragonfly",

"Flowering.Plants",

"Fungus",

"Hymenopteran",

"Lichen",

"Liverwort",

"Mammal",

"Mollusc")

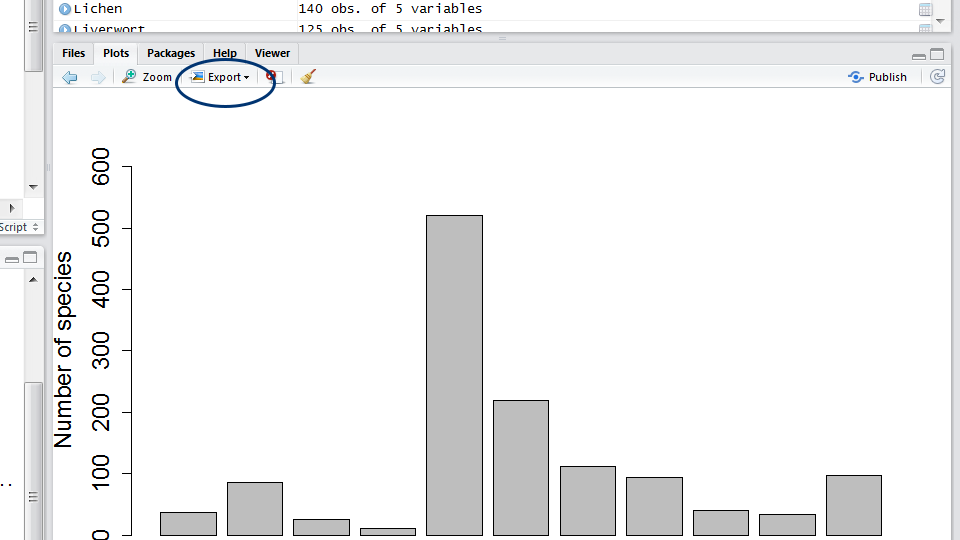
**We can now visualise species richness with the barplot() function.** Plots appear in the bottom right window in RStudio.

barplot(biodiv, xlab="Taxa", ylab="Number of species",

ylim=c(0,600), cex.names= 1.0, cex.axis=1.0, cex.lab=1.0)

*# The cex code increases the font size when greater than one (and decreases it when less than one).*

To save plot: click *Export* in the Plots window. If you don’t change the directory, the file will be saved in your working directory. You can adjust the dimensions to get the bar chart to look how you like it, and you should also add in a meaningful file name



**Step 6: Create a Data Frame and plot it**

**Data frames are tables of values: they have a two-dimensional structure with rows and columns, where each column can have a different data type.** For instance, a column called “Wingspan” would have numeric values measured on different birds (21.3, 182.1, 25.1, 8.9), and a column “Species” would have character values of with the names of the species (“House sparrow”, “Golden eagle”, “Eurasian kingfisher”, “Ruby-throated hummingbird”) .

We will use the data.frame() function, but first we will create an object that contains the names of all the taxa (one column) and another object with all the values for the species richness of each taxon (another column).

*# Creating an object called "taxa" that contains all the taxa names*

taxa <- c("Beetle",

"Bird",

"Butterfly",

"Dragonfly",

"Flowering.Plants",

"Fungus",

"Hymenopteran",

"Lichen",

"Liverwort",

"Mammal",

"Mollusc")

*# Turning this object into a factor, i.e. a categorical variable*

taxa\_f <- factor(taxa)

*# Combining all the values for the number of species in an object called richness*

richness <- c(a,b,c,d,e,f,g,h,i,j,k)

*# Creating the data frame from the two vectors*

biodata <- data.frame(taxa\_f, richness)

*# Saving the file*

write.csv(biodata, file="biodata.csv") *# it will be saved in your working directory*

barplot(biodata$richness, names.arg=c("Beetle",

"Bird",

"Butterfly",

"Dragonfly",

"Flowering.Plants",

"Fungus",

"Hymenopteran",

"Lichen",

"Liverwort",

"Mammal",

"Mollusc"),

xlab="Taxa", ylab="Number of species", ylim=c(0,600))