Module 1. Foundational Concepts of Statistics and Rstudio

Module 1. Foundational Concepts of Statistics and RStudio

Sports performance is all about how athletes perform, the duty of a sports scientist is to constantly investigate and monitor workload conditions to maximise performance and reducing the risk of injuries. The official term of sports scientist is defined as a role that not only collects, organizes, analyses, and interprets the data, but integrates the data from different disciplines in an effort to generate a holistic perspective of the athlete (Martin, 2019).

Quantifying measures of performance such as the key performance indicators described in the previous modules allow us to capture key insights into minuscule, negligible things to the naked eye.

That is what we seek to do with analytics, find what is off that we can fix, examine if the numbers support the coaches' and scouts experience, or if it can open our eyes to an insight that otherwise would be overlooked.

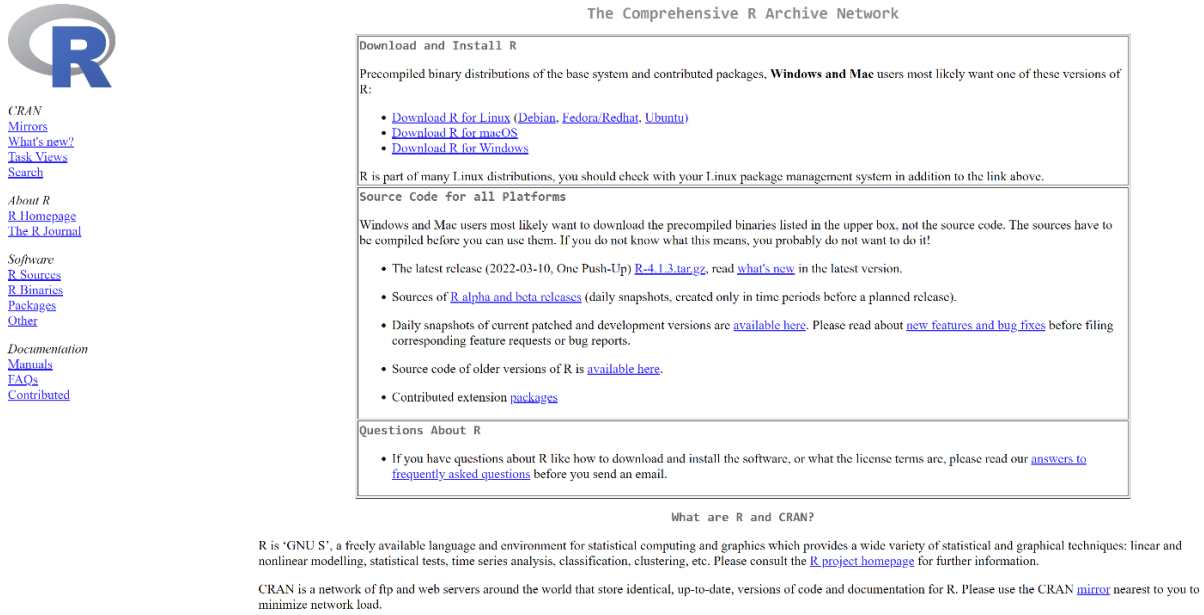
To be able to do this is a science, but there is also an art element to it when visualizing the data correctly. But first, we must learn to implement the tool to be able to import, analyse, and interpret, and extract the actionable insights. To do this, it is ideal to learn a programming language like R and RStudio, which were specifically developed for statistical analyses and data visualization, but at its core, it is critical to have at least a basic understanding of statistics so that they can be applied accurately to the data (Basole and Saupe, 2016).

There are many analytic tools that can be leveraged to analyse data; R and RStudio, Excel, Python, STATA, SAS, and SPSS. R and Python are the most commonly used platforms for analytics and data visualization in the professional sports industry. The reasoning for using R is that its sole purpose for development was for running statistical analysis and, as such, it is a powerful analytic tool. Another reason to start learning R and RStudio is that they are open source and free. With them being open source, there are worldwide contributors that keep improving upon and evolving the platform with new packages and functions. Furthermore, RStudio is an integrated development environment for R that consists of four main panes, although the default opens up with three panes initially (please see Figure 3): the console, syntax-highlighting editor that can execute the code directly; the global environment that handles the workspace management; and the help and plotting window pane. In this course, we will be specifically focusing on learning R and RStudio.

We will guide you through the installation process of R and RStudio. R and RStudio have to be downloaded in the following sequence to function correctly;

1 Download R from the following website: https://cran.r-project.org/ (see Figure 1 below)

Figure 1: Download R

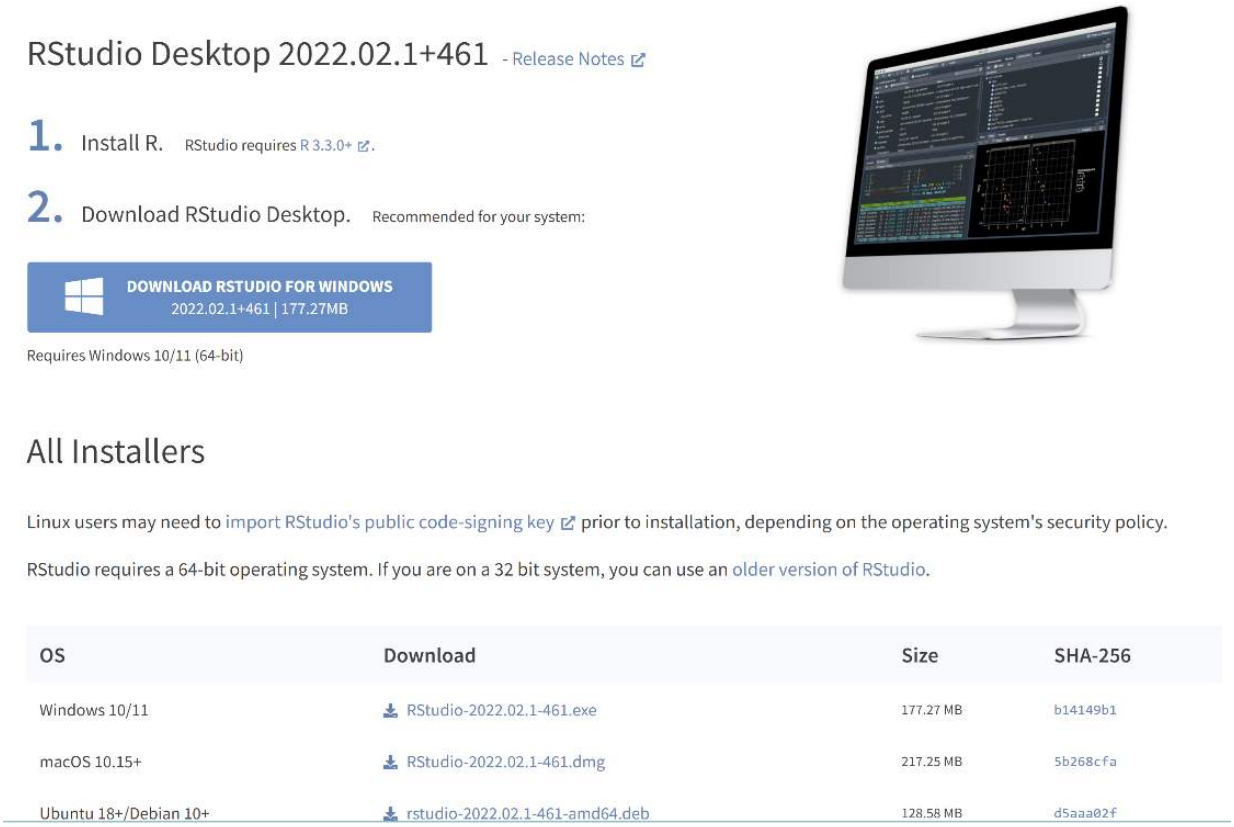


Source: Screenshot by author (https://cran.r-project.org/index.html)

2 Download RStudio from the following website: https://www.RStudio.com/products/RStudio/download/#download (see Figure 2 below).

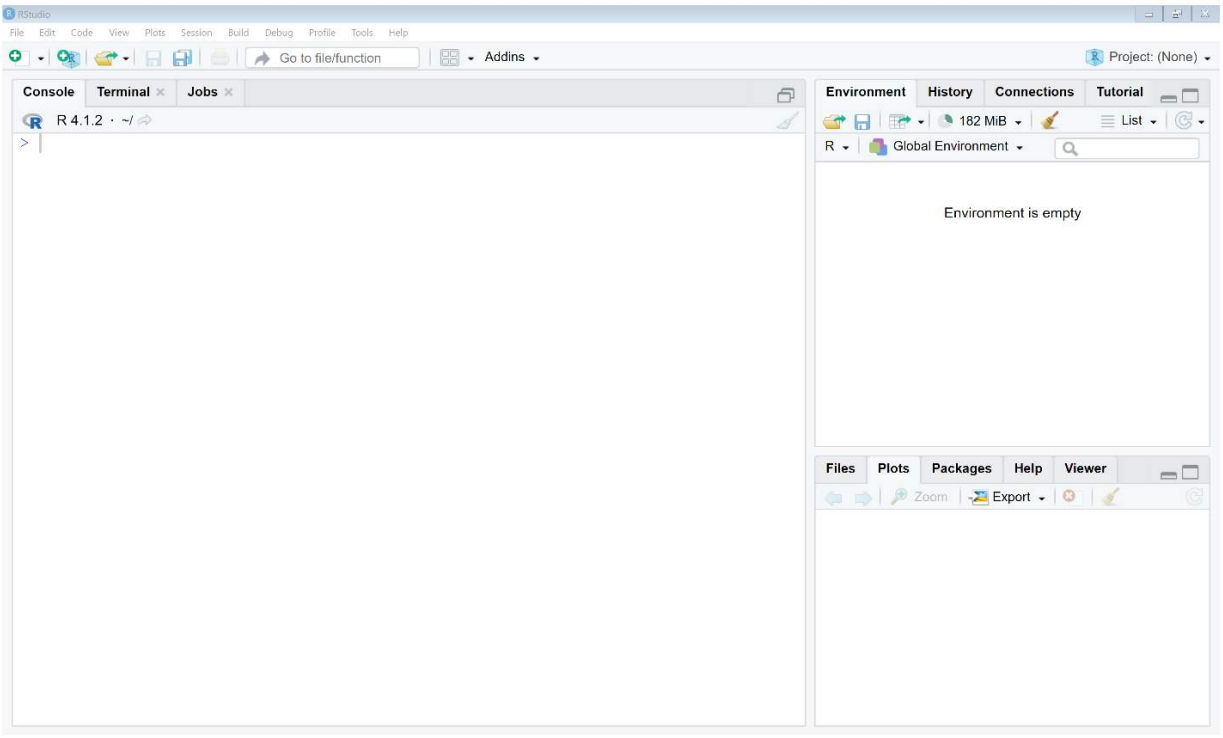
RStudio is available in open-source and commercial editions and runs on the desktop (Windows, Mac, and Linux) or in a browser connected (https://www.RStudio.com/products/RStudio/).

Figure 2: Download RStudio



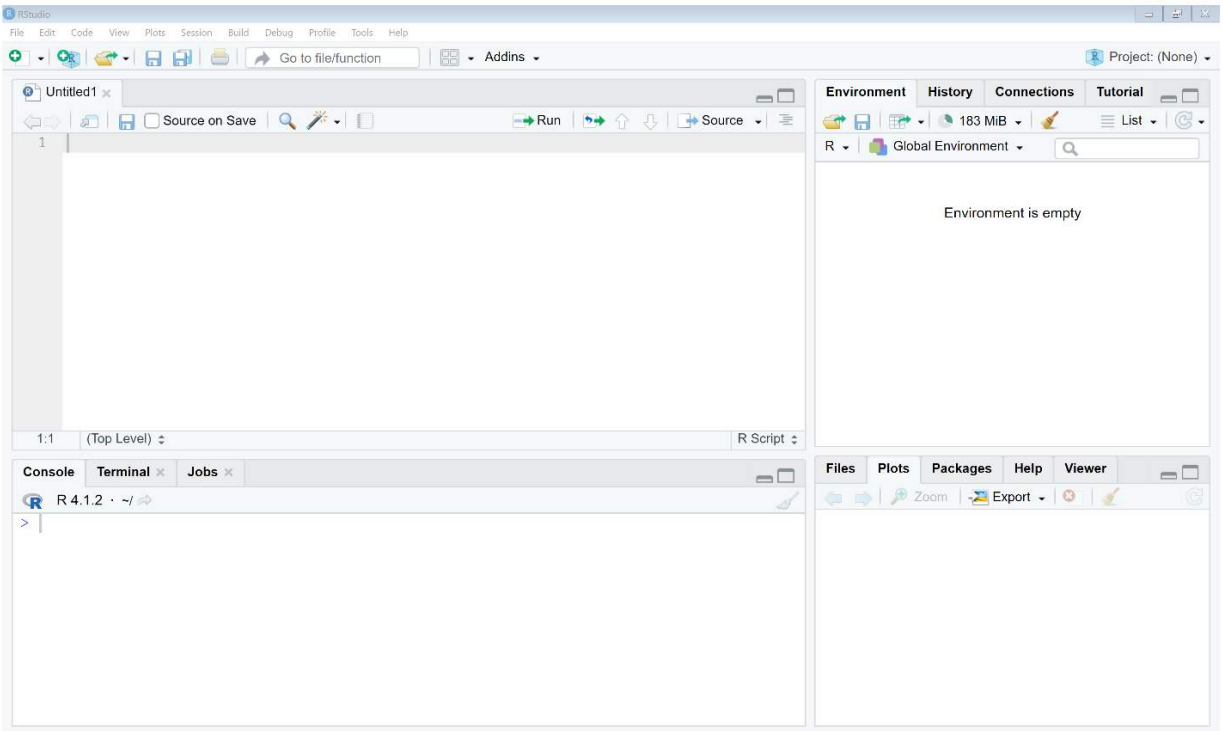
Source: Screenshot by author (https://www.RStudio.com/products/RStudio/download/#download)

Figure 3: An example of the default three pane view



Source: Screenshot by author from RStudio (RStudio, 2022).

Figure 4: An example of the four pane view



Source: Screenshot by author from RStudio (RStudio, 2022).

A description of what each window pane is for

The top left window pane, also commonly referred to as the scripts' pane, is where you typically code and write the syntax. Syntax is a set of legal structures and commands that you use to give instructions to implement certain functions and execute.

The bottom left window pane is called the console, and it is where the output, the results executed from the code on the top left window pane, are displayed, in other words, the messages printed to the user by a program.

The bottom right window pane is called the task pane, and it consists of several tabs that allow you to locate files, plots, packages, and request help. The packages tab shows available R procedures, while the help tab is for any R topic, displayed responses for questions related to the functionality of functions.

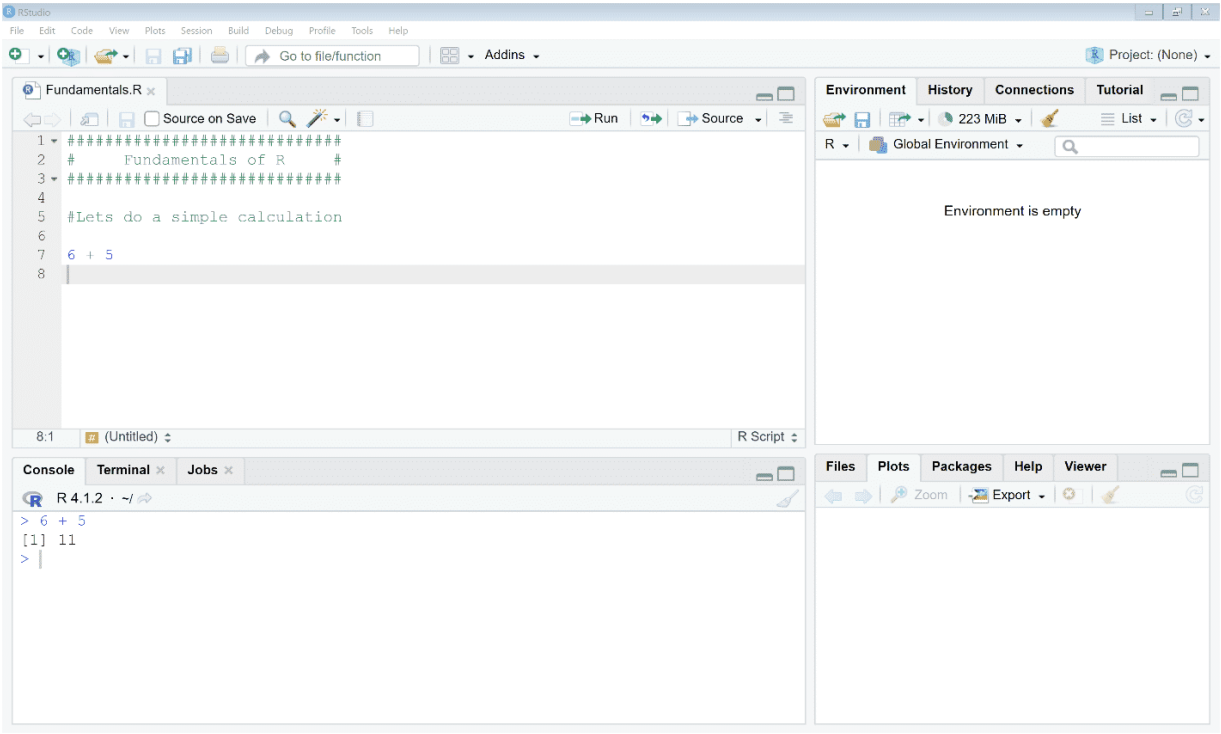
The top right window pane is also referred to as your workspace, and it refers to your R working environment, which includes any objects (vectors, matrices, data frames, lists, functions) that you may have created or imported into R.

Before continuing on with coding, it is important to understand a few terms of the R language such as the object, vectors, functions, arguments, packages, matrices, data frames, and lists (http://www.sthda.com/english/wiki/easy-r-programming-basics).

Below is a bullet point list of general terms and descriptions:

- An **object** is anything that is created in R.
    - Before even beginning to create an object, let us run a simple calculation of the summation of 6 + 5 in the R script (top-left pane).
    - When you press CTRL + Enter, you will see the output result generated on the console (bottom-left pane) as shown on the figure below.
    - Also, go ahead and point towards the file on the top ribbon and choose Save As and name this file Fundamentals, it will now name your tab label Fundamentals.R automatically, as also shown in the figure below (Figure 5).

Figure 5: Calculations in R

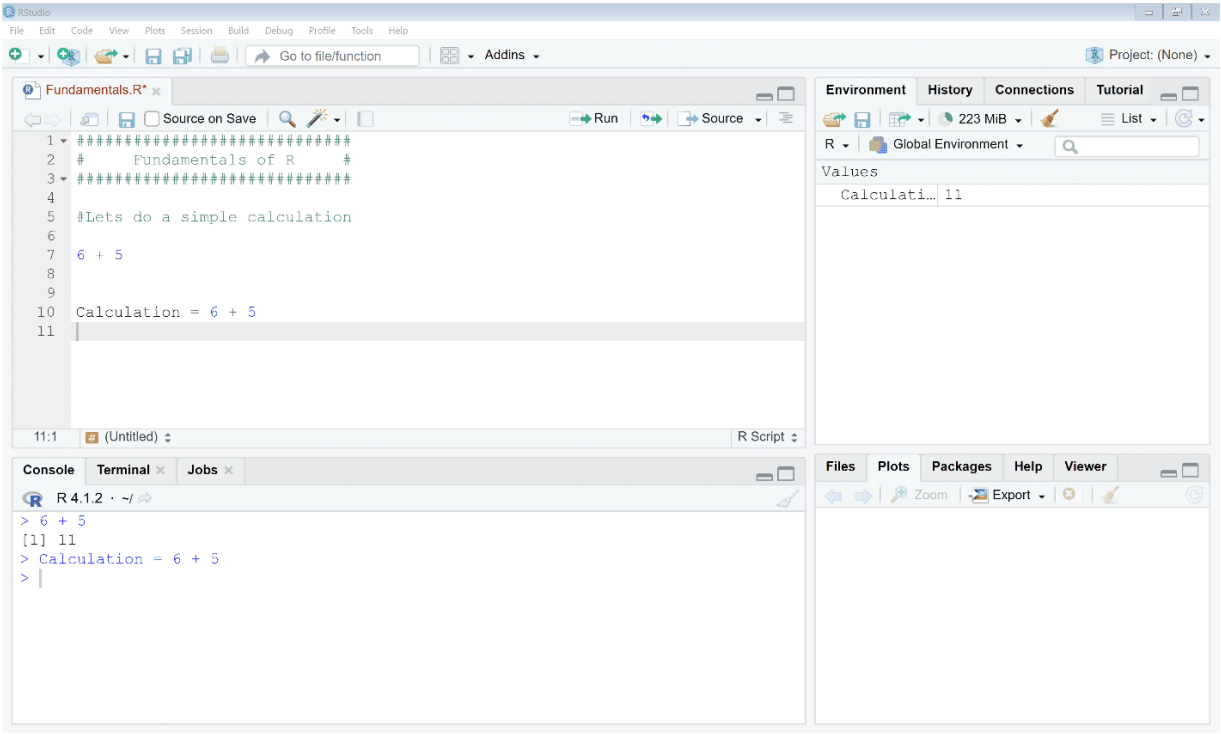


Source: Screenshot by author from RStudio (RStudio, 2022).

- Then, if you want to take it a step further and create an object, we type as follows (assuming you are creating an object called Calculation, also, keep in mind that R is case-sensitive):

    - Calculation = 6 + 5 (Now, notice a few things; if you look at your Global Environment (top right window pane), you will see that your environment no longer states "Environment is empty" as it did before – take a look at the previous screenshots). It now contains an object called Calculation, as shown below in Figure 6.

Figure 6: Create an object in R

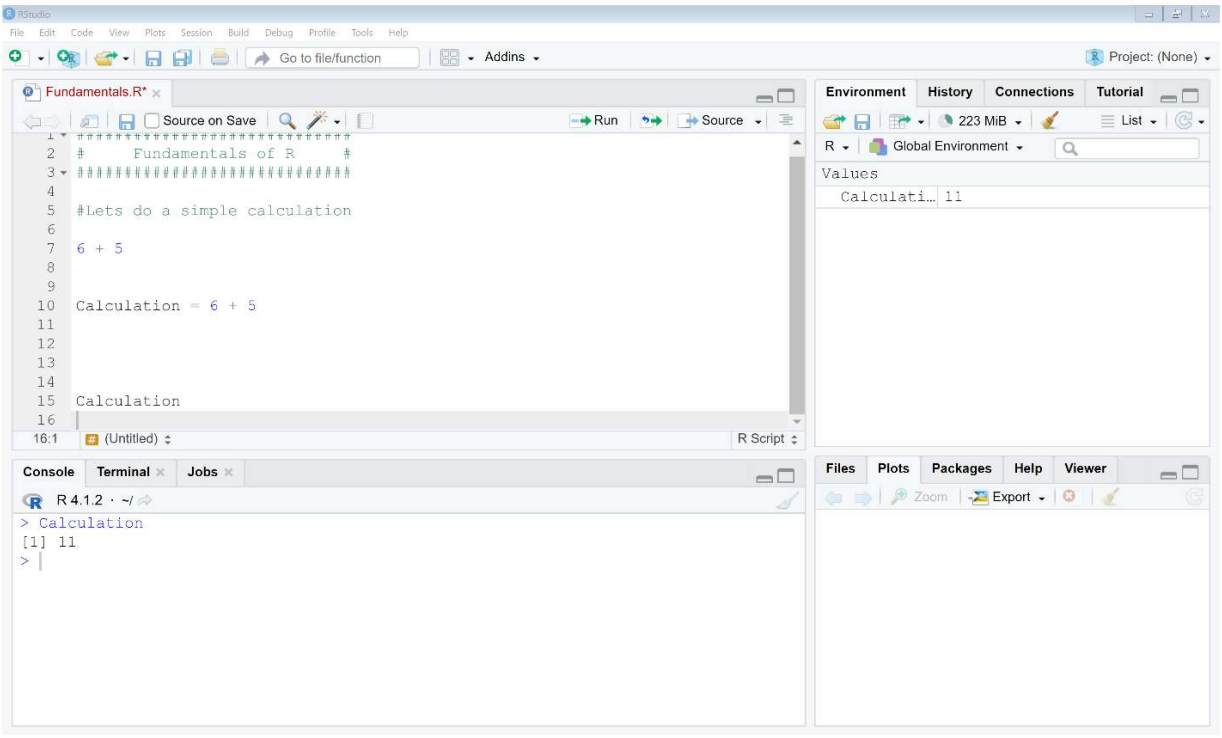


Source: Screenshot by author from RStudio (RStudio, 2022).

- Also, notice that when you executed the code by pressing CTRL + Enter it now shows "Calculation", but it did not print out the result of the calculation. To see the result of the Calculation object, you will have to call Calculation, and it will yield the output in the console.

- What is meant by calling the object, is to simply type either directly into the console as such "Calculation" or type it in the R script and then press CTRL + Enter, and it will yield a similar output to the figure below, Figure 7.

Figure 7: Calling the object

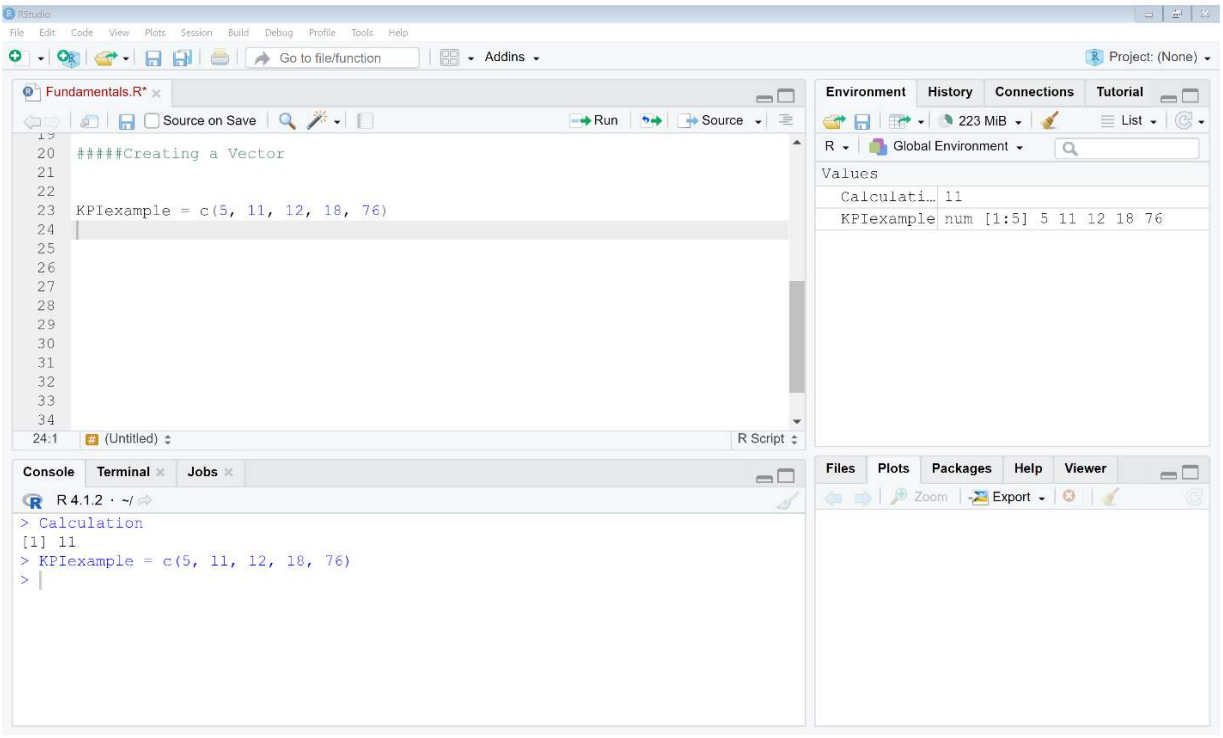


Source: Screenshot by author from RStudio (RStudio, 2022).

- A **vector** is a collection of data that is of the same type. A vector is a combination of multiple values (numeric, character, or logical) in the same object.

  - It is convenient to think of vectors as arrays of data, in simple terms, think and visualize a column from an Excel table and now transpose it horizontally. For example, a vector in R could include five values: 5, 11, 12, 18, 76 into an object called KPIExample by creating a vector in the following manner:
  - KPIExample = c(5, 11, 12, 18, 76)
  - It is important to keep in mind that, to group several numerical values, we are using a function. The function we are using is c(). Key pointer: functions are usually followed by open and closed parentheses. This particular function stands for concatenate, which is analogous with combine.
  - Technically, we are putting together the number values 5, 11, 12, 18, 76 in a column, in R it is called a vector of numbers.
  - If you take a look at the figure below, you will notice that on the Global Environment pane you have an object called KPIexample that, is of num representing numeric type with 1:5 values. As you can see, the Global Environment is your go-to place to get information about your data. See below in Figure 8.

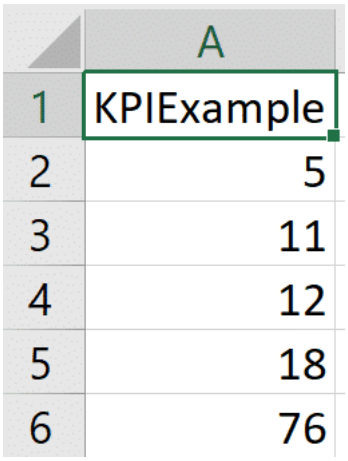
Figure 8: Create a vector



Source: Screenshot by author from RStudio (RStudio, 2022).

- In Excel, it would have been displayed as follows:

Figure 9: Vector in Excel



Source: Screenshot by author from Excel (Microsoft Excel, 2021)

- A **function** is a tool that R uses to complete some kind of operation and is typically identifiable by the pattern that it is typically a lowercase word followed by open and closed parentheses, for instance, summary() which is a function that provides the sports scientist with the six number summary of descriptives (more on that later). Also, keep in mind that, eventually, you can create your own functions that can perform complex operations.

- To show how to use functions in R, it is recommended to execute functions on data to get the experience.

- As such, RStudio has packages that when installed make certain data available to you.

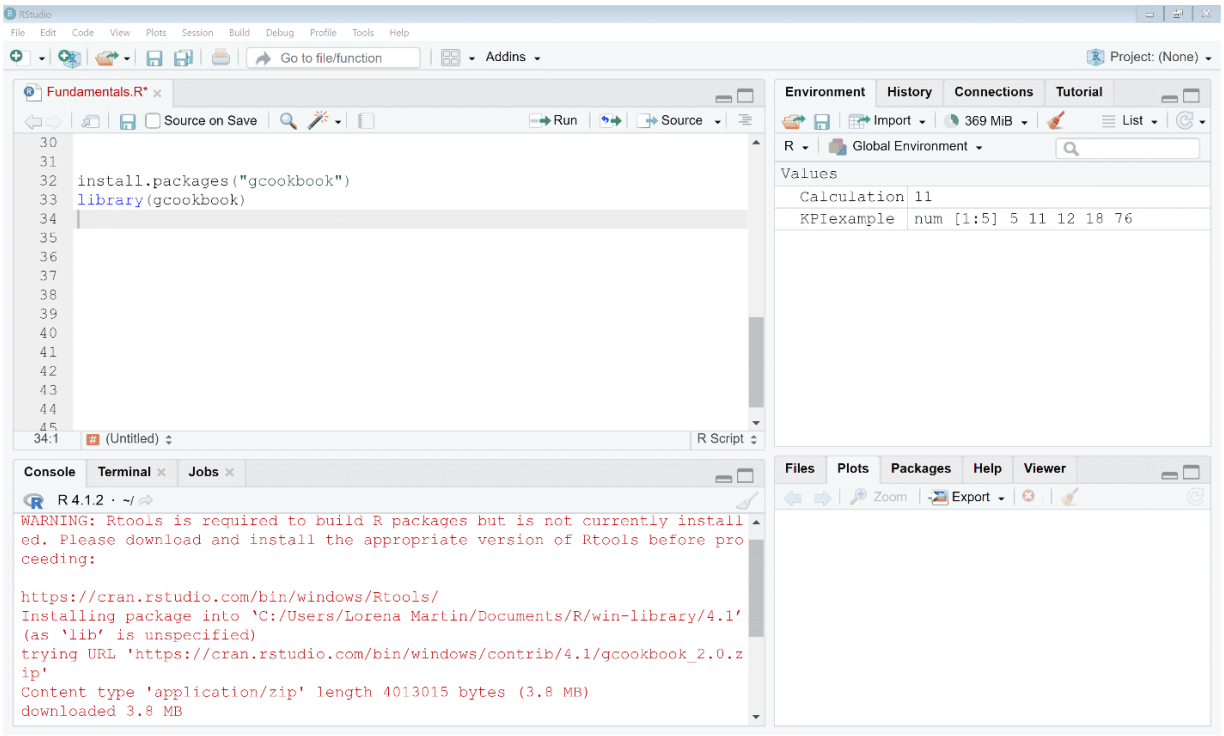
- Please install the package of **gcookbook** as it has a dataset related to the sport of baseball called **tophitters2001** for which we can execute several functions to see how they work in R as shown in Figure 10 below.

- Also, keep in mind that when you type the lines of code (make sure that it is all in lowercase) do not be alarmed by the warnings that may appear in the console.

- *install.packages("gcookbook")*

- *library(gcookbook)*

Figure 10: Functions in R

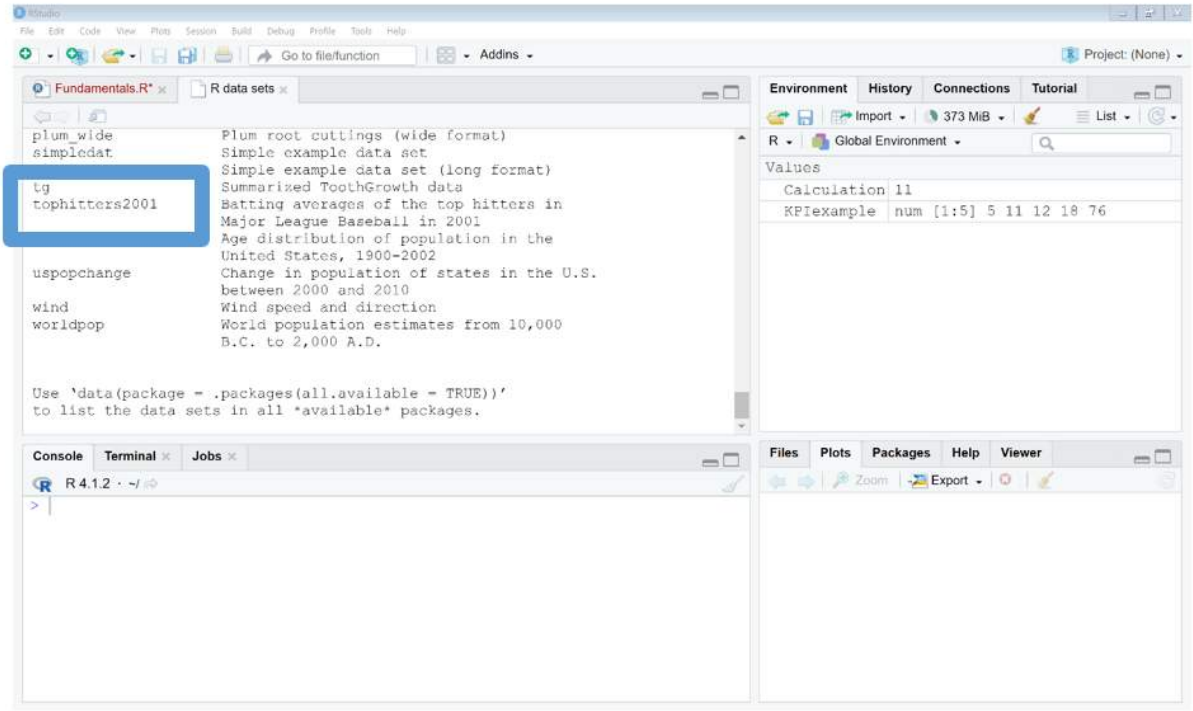


Source: Screenshot by author from RStudio (RStudio, 2022).

Then, once the package of gcookbook has been installed and loaded by calling the library function, please type in a line below: data()

This will display some datasets that are completely available to you for free within R and the package of gcookbook. Being that this is a sports performance analytics course, we strongly recommend that you follow this guide and install gcookbook, call it with the library and identify the tophitters2001 dataset. See the figure below (Figure 11).

Figure 11: Tophitters2001 dataset

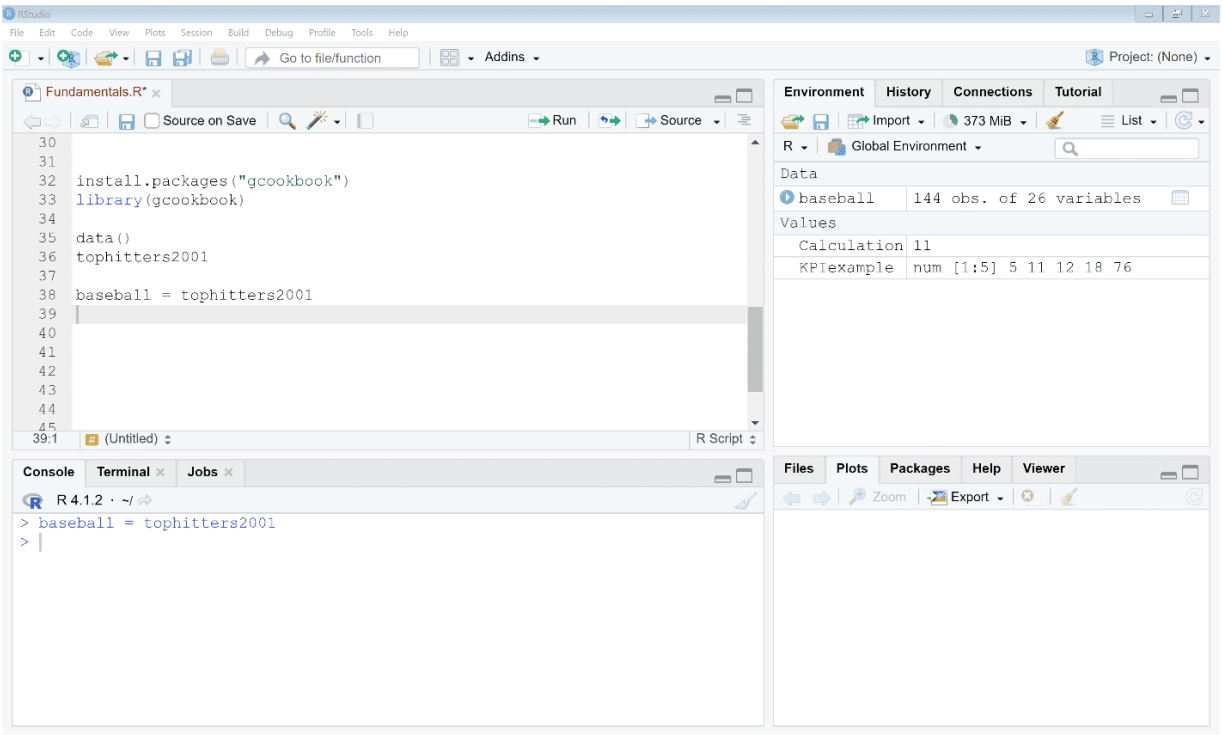


Source: Screenshot by author from RStudio (RStudio, 2022).

Now, let us go ahead and create an object that we will call baseball that contains the data from tophitters2001.

- Notice that, on the Global Environment, you now have an icon that appears as a blue circle. This is to denote that there is an object that is a data frame (more on this later, but for now, just know that the data is organized in rows and columns).

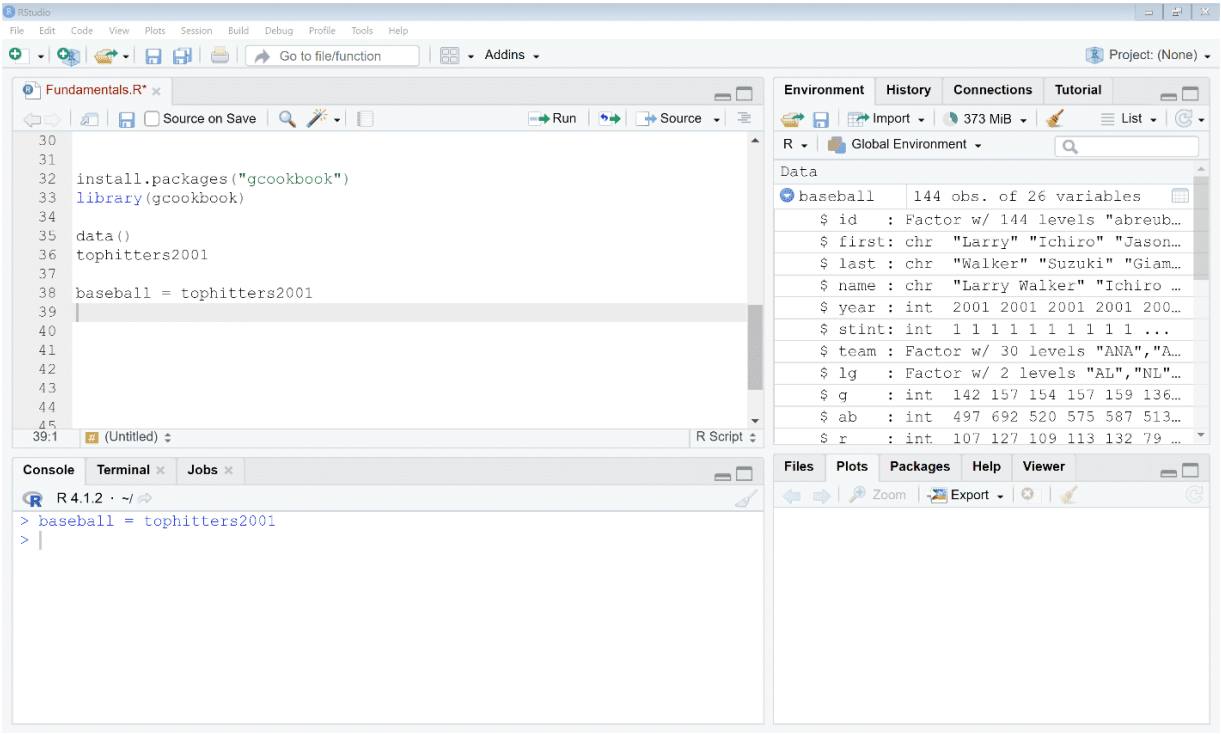
Figure 12: Object containing data from tophitters2001



Source: Screenshot by author from RStudio (RStudio, 2022).

- If you click on the blue circle once, it will drop all the variables and key performance indicators contained within, as shown in figure 13.

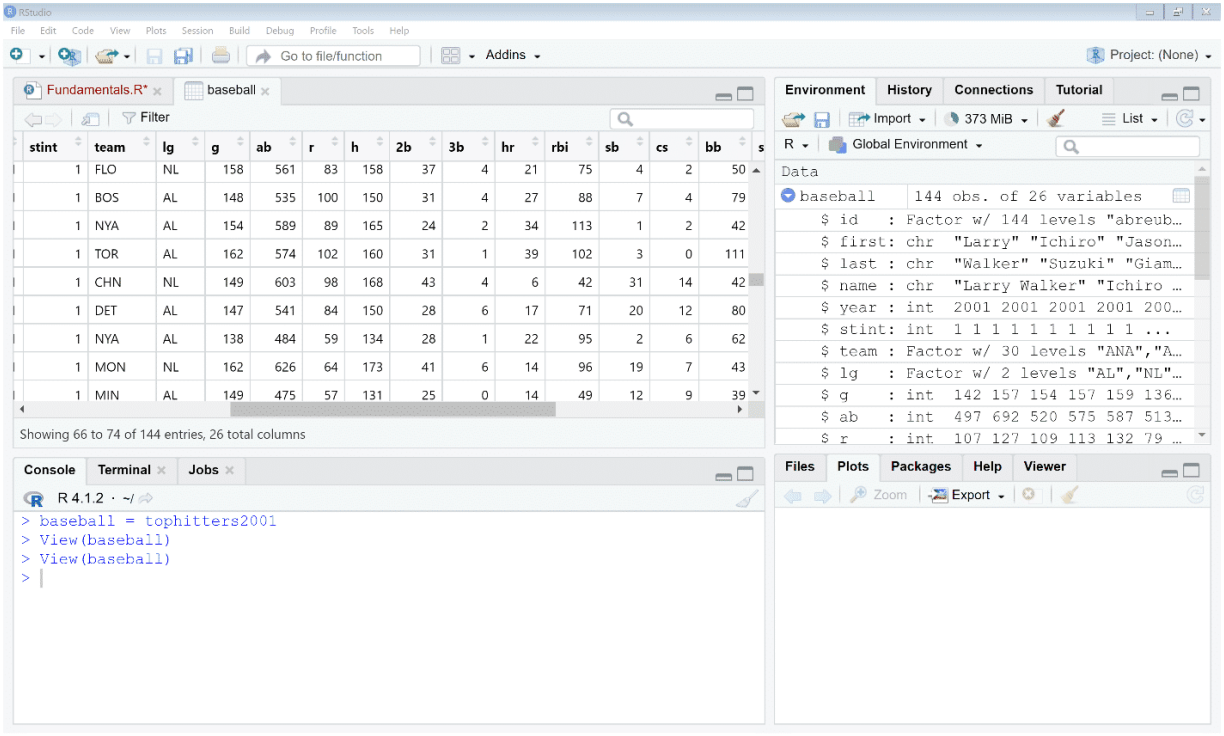
Figure 13: Object variables



Source: Screenshot by author from RStudio (RStudio, 2022).

- If you click on the icon twice, then it will open up a tab such as when you previously typed data(), but this time it will show the data in an Excel format looking table similar to figure 14 below.

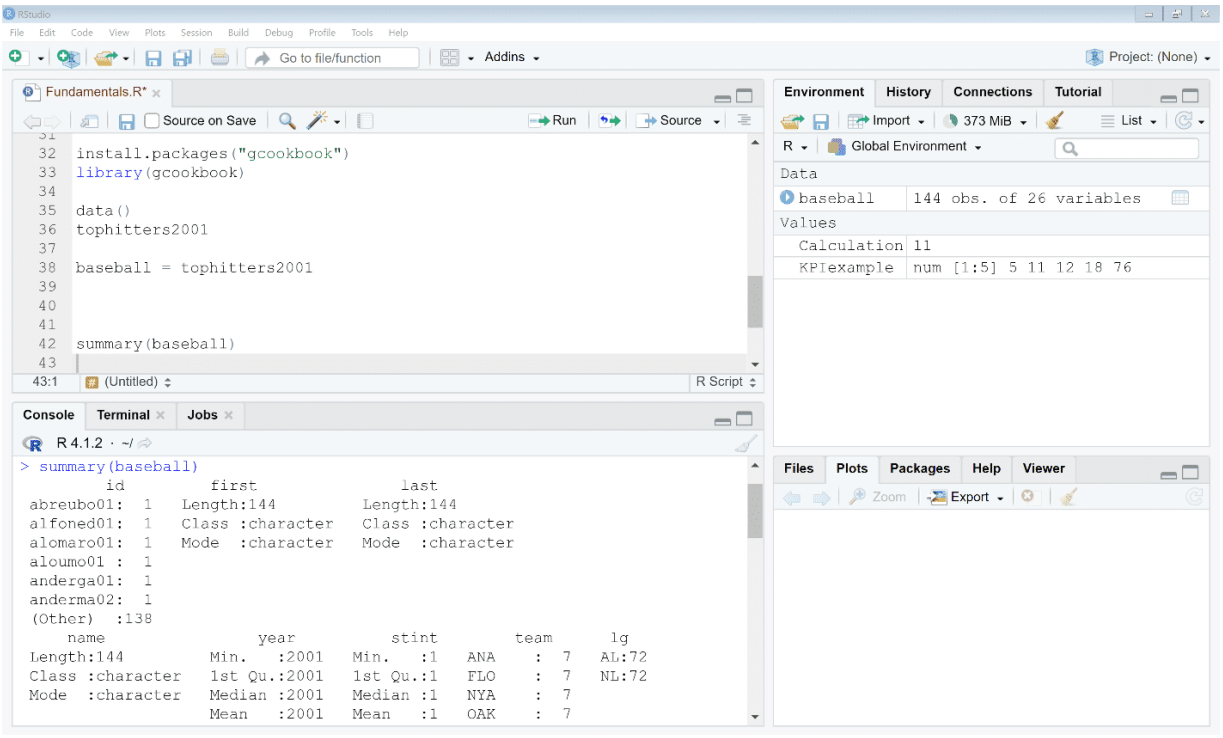
Figure 14: Data shown in table format



Source: Screenshot by author from RStudio (RStudio, 2022).

Going back to functions, let us execute the function summary() on the entire data frame of baseball, as shown on the figure below (Figure 15).

Figure 15: Function summary()



Source: Screenshot by author from RStudio (RStudio, 2022).

When you type summary(baseball), you get the output displayed above in the figure above, whereby it automatically gives you a six number summary for each numerical variable in the dataset. The data types that are of character only yield the following: length, class, and mode (more on data types later). For the numerical data type variables, the minimum, the 1st quartile, the median, the mean, the 3rd quartile, and the maximum are all given.

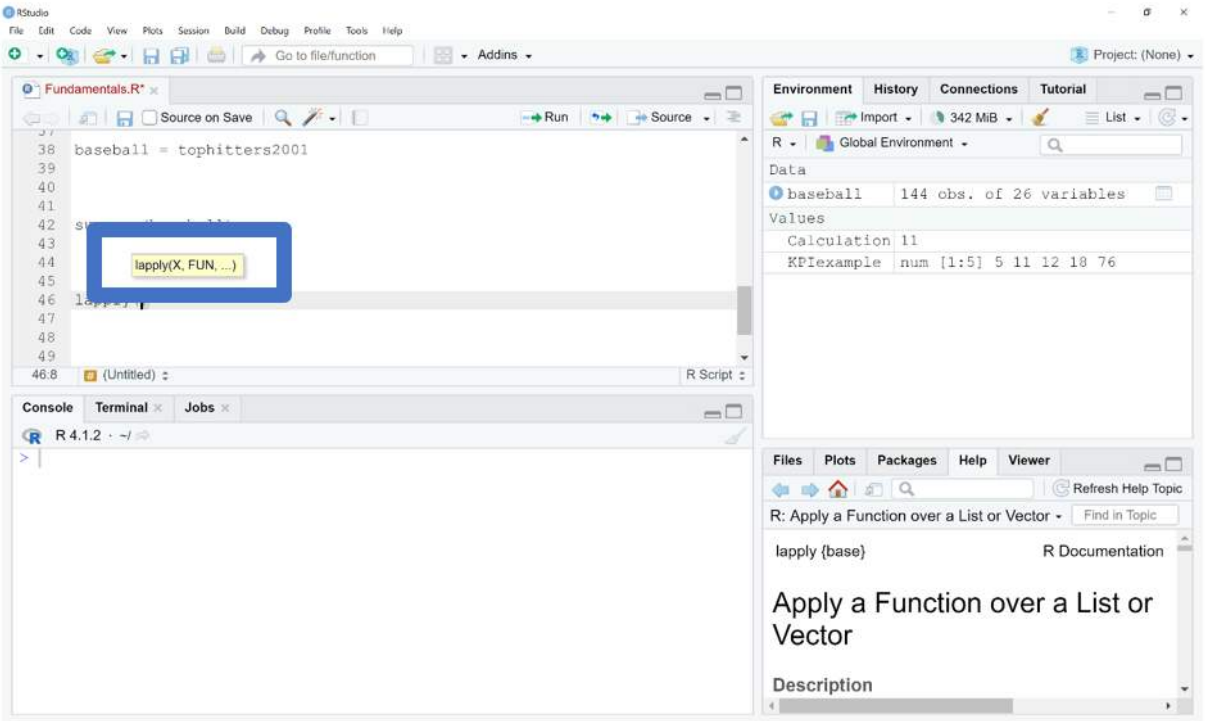
This was a nice, simple way to obtain the descriptives of your entire dataset.

- An **argument** is what a function acts on. For instance, in the previous bullet point regarding functions, we used the example of the summary() function. Now, if we were to implement a new function called lapply which is used to apply a function over a list or vector, and you begin typing lapply() when you open the parentheses, you will see a prompt appear as shown in the figure below, which indicates the arguments that you are recommended to populate so that the code can run correctly. In the case shown in the figure below (Figure 16), we see that it shows an X, followed by FUN.

- The X is to denote your dataset or object.

- FUN represents what function you want to apply to your vector, for instance, mean(). Sidenote, yes, it is a quasi-nested function.

Figure 16: Function lapply()

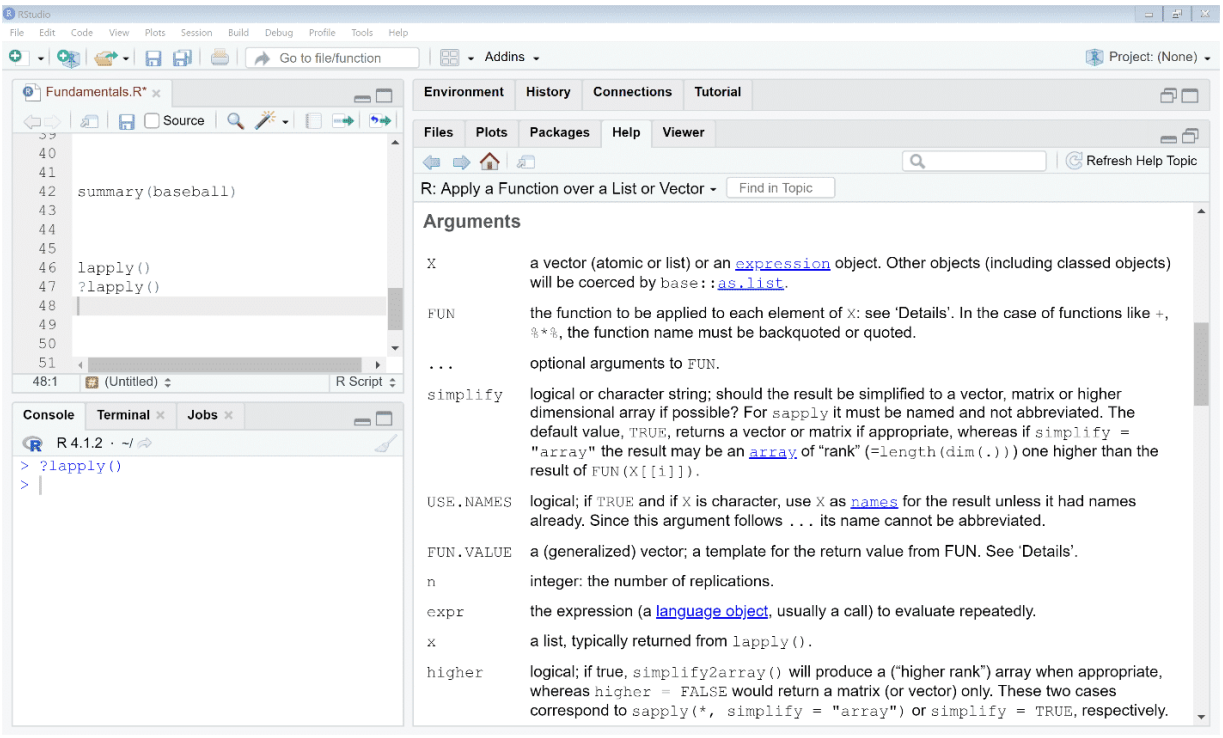


Source: Screenshot by author from RStudio (RStudio, 2022).

Now, let us suppose there is a function you want to use, but are not sure what are the arguments within, you can always ask for help from R by typing a question mark in front of the function and then pressing CTRL + Enter.

As soon as you execute the line of code for help, the bottom right window pane will go to the help tab and open up information on the function as well as the arguments within that function as such, as shown in Figure 17 below.

Figure 17: Help tab



Source: Screenshot by author from RStudio (RStudio, 2022).

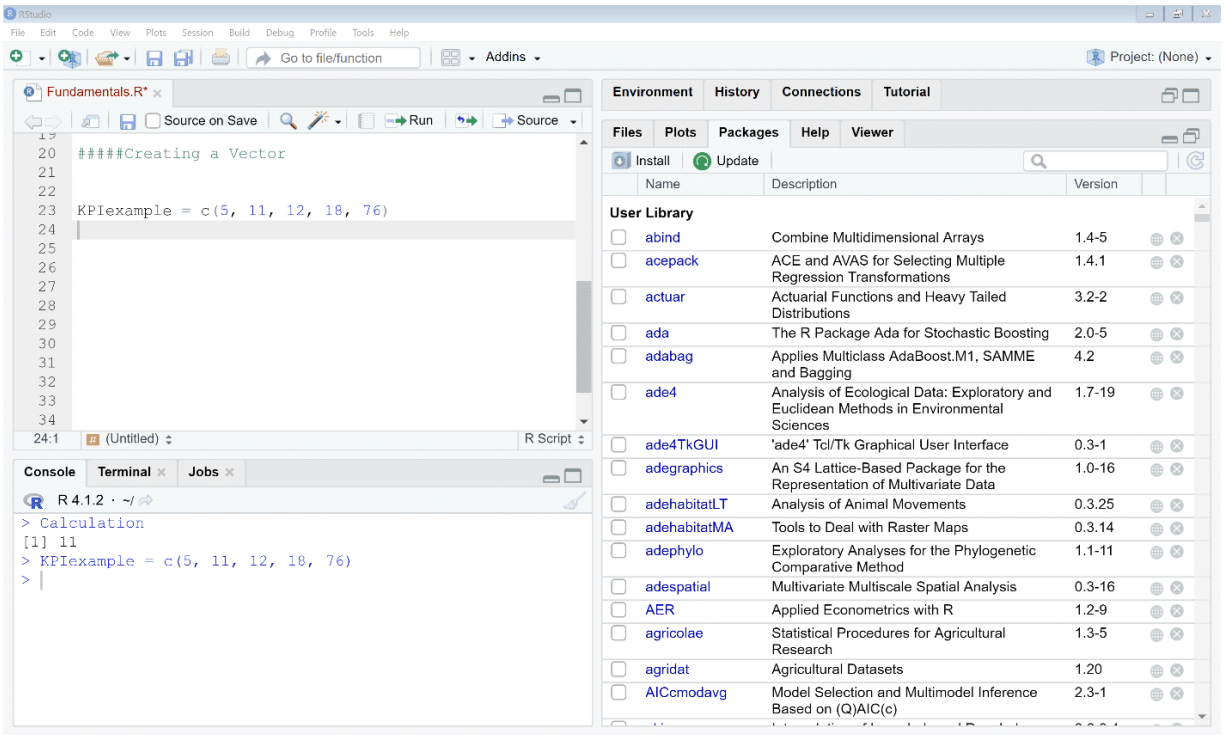
- A **package** is a collection of functions. In R, there are a multitude of packages that consist of numerous functions, in fact, there are currently over 18,375 packages that can help perform the actions you want.

- If you maximize your bottom right window pane; Files/Plots/Packages/Help/Viewer Pane, if you click on the tab for packages, you can see a list of available packages in RStudio that you can install simply by checking the box to the left of the name of the package as shown in Figure 18 below. Otherwise, you could also use the customary lines of codes to install the packages of your choice by using both the following;

- *install.packages("package")*

- *library(package)*

Figure 18: Packages in R



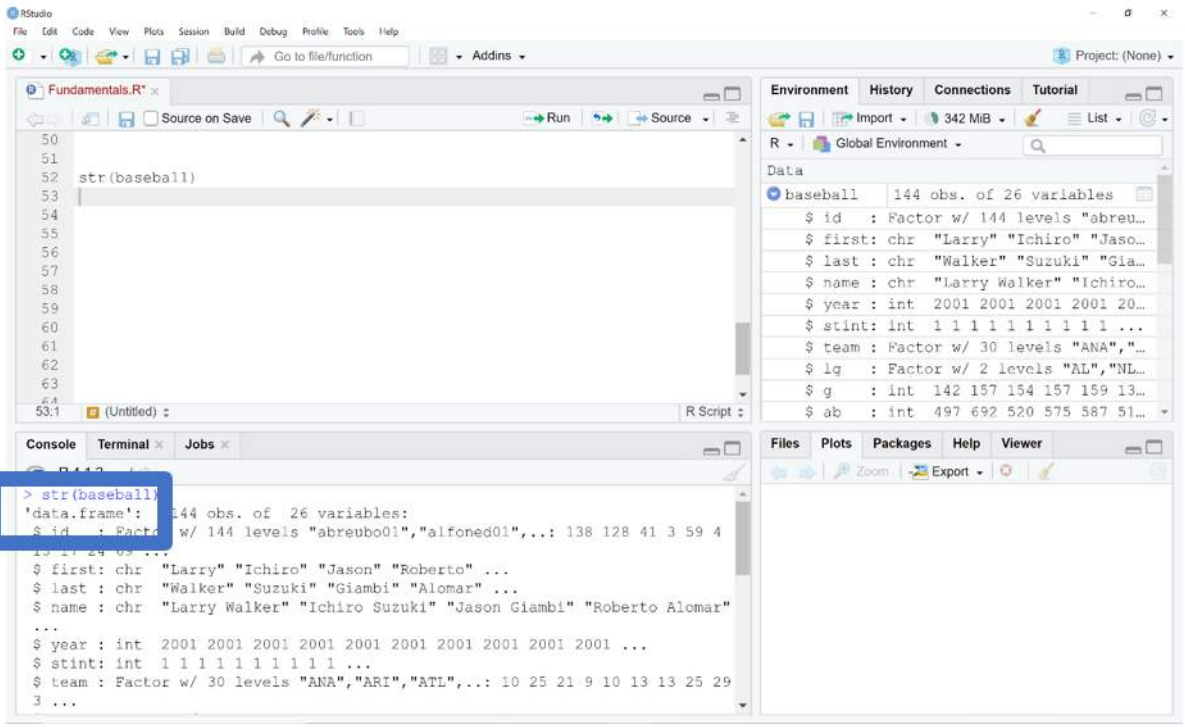
Source: Screenshot by author from RStudio (RStudio, 2022).

- **Data frame**s: a data frame is a table, similar to that of a matrix. The main difference is that the columns can have different data types (numeric, character, logical). It is also important to keep in mind that it is constructed from rows and columns. Finally, rows in R are referred to as observations and columns are known as variables.

- To reiterate, if you examine the Global Environment and you see a blue circled icon, that is indicative that you have an object in R in the form of a data frame. However, it will not explicitly state that it is a data frame at this location.

- To confirm the data structure, it is strongly recommended that you execute the function str() with the name of the object with in it, as shown in Figure 19 below. The first line on the console, the bottom left window pane, will display the data structure along with each of the variables' data type.

Figure 19: Data structure of data frames in R



Source: Screenshot by author from RStudio (RStudio, 2022).

- **Matrices** consist of combined vectors with the same element type, which can be either numeric, character, or logical.
- **Lists** are a collection of objects, the distinguishing characteristic is that it can contain all kinds of R objects. Typically, when we scrape data from the web, it imports it into R as a list since the websites contain numbers, characters, images, and other unstructured items.

For this course, we will concentrate on vectors and data frames, as those are the common data structures that you will be dealing with for the most part as a sports scientist.

Importing data in the form of Excel files and CSV files

Typically, a regular dataset that we receive once we have collected data is characterized by rows and columns, as is customary in Excel and comma-separated values (CSV) files. In R, when we import a CSV or Excel file and save it as an object, we want to make sure that it assumes the structure of a data frame, a table like format that is comparable to an Excel table with rows and columns. In order to this, we import the dataset using the following command:

- read.csv(file.choose())

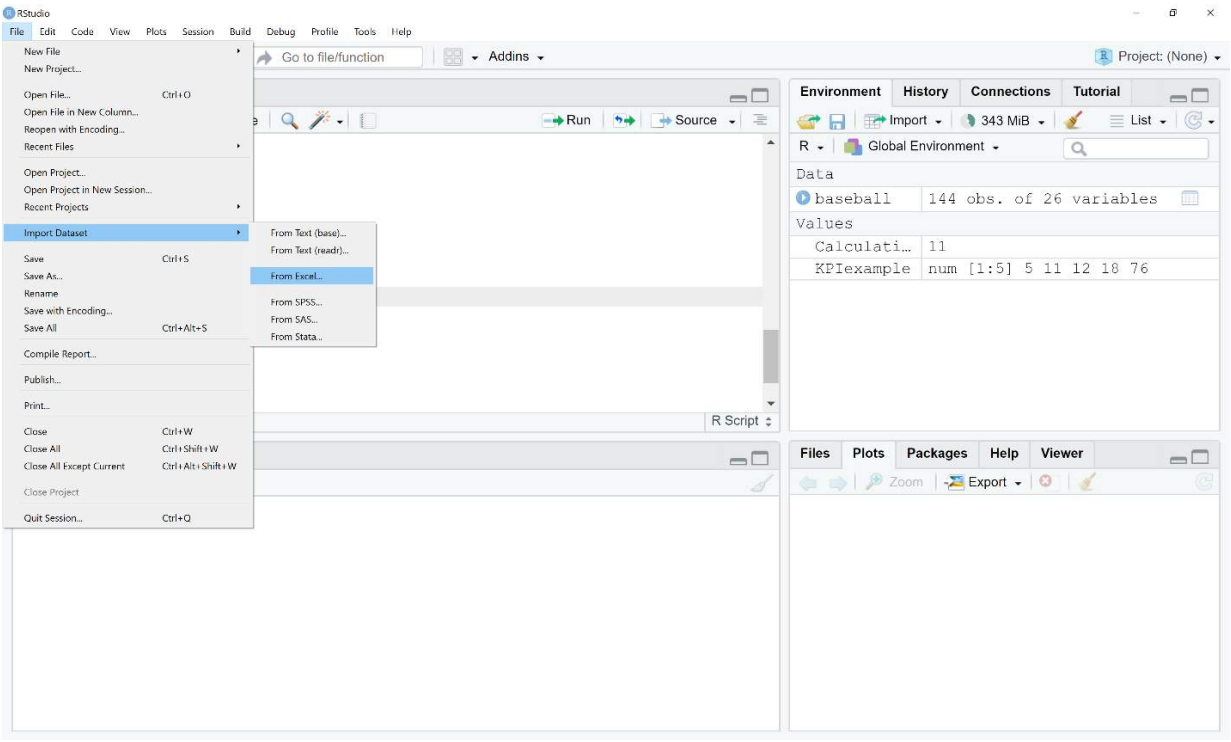
    - These two functions: read.csv and file.choose, which is nested within the read.csv function, will open up a dialogue box for you to choose the appropriate CSV file.

- Once imported, we assign it to an object (let us call it sportsinfo) using the = or <- operators.
- For example, if importing a dataset that you will store and save as an object called sportsinfo, the code would be as follows:
    - sportsinfo = read.csv(file.choose())
- Then, you can check for the structure of the imported file to see whether it was imported as a table or not, using the function str().
- If it was not read into RStudio as a table or data frame, you can transform it to be a data frame by implementing the function data.frame(sportsinfo). This is important because many of the statistical models as well as data visualizations conducted using the package ggplot2 require the data to be of a data frame structure.
- The function data.frame(sportsinfo) will convert the object sportsinfo into a data frame that can later be used to run statistical analyses on as well as create beautiful data visualizations using ggplot2, another package in R specifically designed to generate effective data visualizations.

Then, there is also a point and click alternative way of importing files in to R:

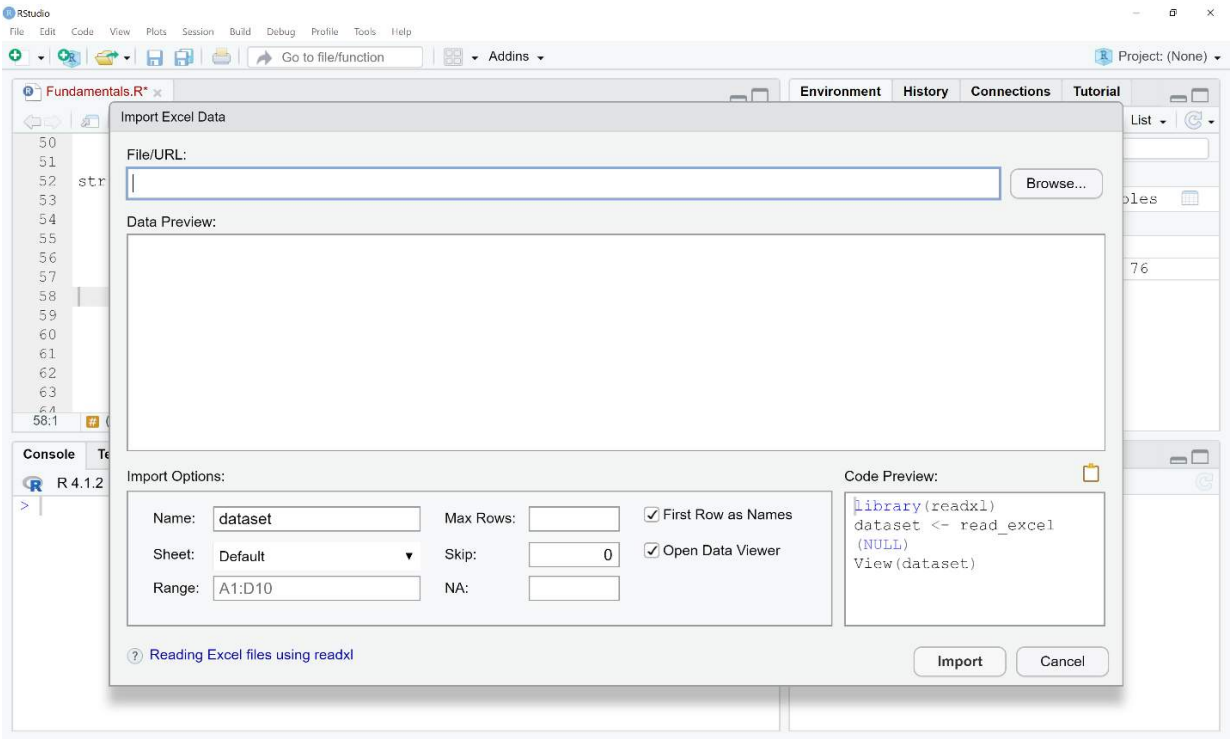
- Go to File > Import Dataset > From Excel in the upper-left corner of the scripts' pane.

Figure 20: Importing a file in R through point and click method



Source: Screenshot by author from RStudio (RStudio, 2022).

Figure 21: Importing a specific sheet within an Excel workbook



Source: Screenshot by author from RStudio (RStudio, 2022).

There is a third way to import data using the directory to where R is pointing at.

To find out where RStudio is pointing at within your system, type in the following code:

- getwd()
    - This function represents getting your working directory and will identify where R is pointing at directly in your computer system.
    - Once you identify where it is pointing to, if it is, in fact, pointing to the CSV file you want to import, then there is no further step to follow, just read in the file using read.csv and save it as an object.
    - sportsinfo <- read.csv("from the file directory")
- If your working directory is not pointing towards the CSV file you wish to import, or you deliberately want to change where R is pointing towards, then you will need to type the following code to redirect the working directory by setting it to point to a different location.
- setwd()
    - This function represents setting your working directory to a new location.
    - If you aim to change your working directory to a different location, you need to identify the new location.
    - One way to do so is to
        - right-click on the file of interest to import into R and scroll down to properties, then, choose Properties, and under the General Tab you will find Location; copy the Location and type that within the setwd("Location of file you want to import") function within double quotations.

These are some of the basics of R and RStudio. Also, in R, if we want to keep track of information and keep notes along the way, for instance, if we created a new variable. In order to this, we can easily include this in the R script pane we must use the hashtag to place comments, the comments can be placed before or after the actual code (Wickham and Grolemund, 2020).

Tip: If you want to hashtag and comment out several lines of code or hundreds of lines of codes, use the following:

CTRL + SHIFT + HASHTAG

This concludes the introduction to R and RStudio installation process and basics. In the next module, we will learn about the basics of statistical methodology and begin implementing descriptive analyses with RStudio.

CONTINUE

References

Allaire, J. J. (2022). R 4.2.1 [Computer Software]. RStudio, Inc. https://cran.r-project.org/index.html

Basole, R. C. and Saupe, D. (2016). Sport Data Visualization [Guest editors' introduction]. IEEE Computer Graphics and Applications, 36, 24-26. https://doi.org/10.1109/MCG.2016.85

Martin, L. (2019). Sports science data protocol. *Sports and Exercise Medicine Open Journal, 5*(2), 36-41. http://dx.doi.org/10.17140/SEMOJ-5-174

Microsoft Corporation. (2018). *Microsoft Excel*. Retrieved from https://office.microsoft.com/excel

Wickham, H. and Grolemund, G. (2020). R for Data Science. O'Reilly.

CONTINUE

Download

PDF

Module 1. Foundational Concepts of Statistics and RStudio.pdf

5.1 MB