A QUICK TOUR OF RSTUDIO

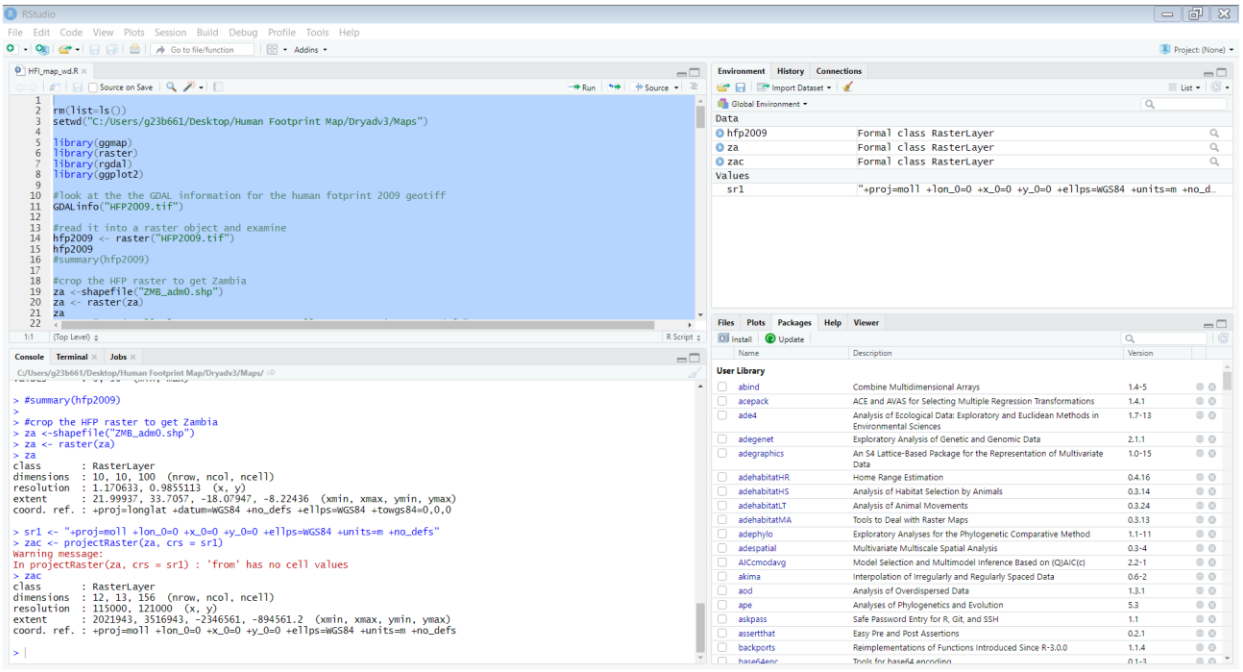
Conservation Biology, BIOE 440R/521

Scott Creel

3 September 2019

There are 4 panels or panes visible when you run RStudio in its default configuration.

Each panel has more than one tab, and clicking the tabs determines what the panel is used for.



The **TOP LEFT** panel is primarily the **SCRIPT EDITOR,** where you write (or paste) lines of code, run them , and save them as an R 'script' or program (with a .R extension on the filename) using the file tab on the menu at the very top. Once you have saved a script as a file with the .R extension, you can re-open it and re-run it, or modify it and save it with a new name. Save often when writing code, especially right after you've written and run a line or set of lines that work correctly.

To run one LINE of code, put the cursor in that line and press CTRL-ENTER. When you are not sure everything is OK or you are getting error messages, running code one line at a time and using the other panels (see below) to understand is a good idea.

To run a BLOCK of code, highlight it by dragging, and then press CTRL-ENTER to run the highlighted block.

To run an entire SCRIPT (all the lines, in order), press CTRL-A (for 'all'), to the entire script is highlighted, and then press CRTL-ENTER.

A line that starts with # is a COMMENT. Running a comment line does not create any OBJECTS (see below) or make anything else happen. Use comments to explain to yourself and others what a line or block of code does. Bothering to write clear comments makes a script MUCH easier to understand, especially when you have put it aside and returned to it.

The top left panel can also be used to examine OBJECTS that your code creates, especially DATA FRAMES. More on this when discussing the top right panel below.

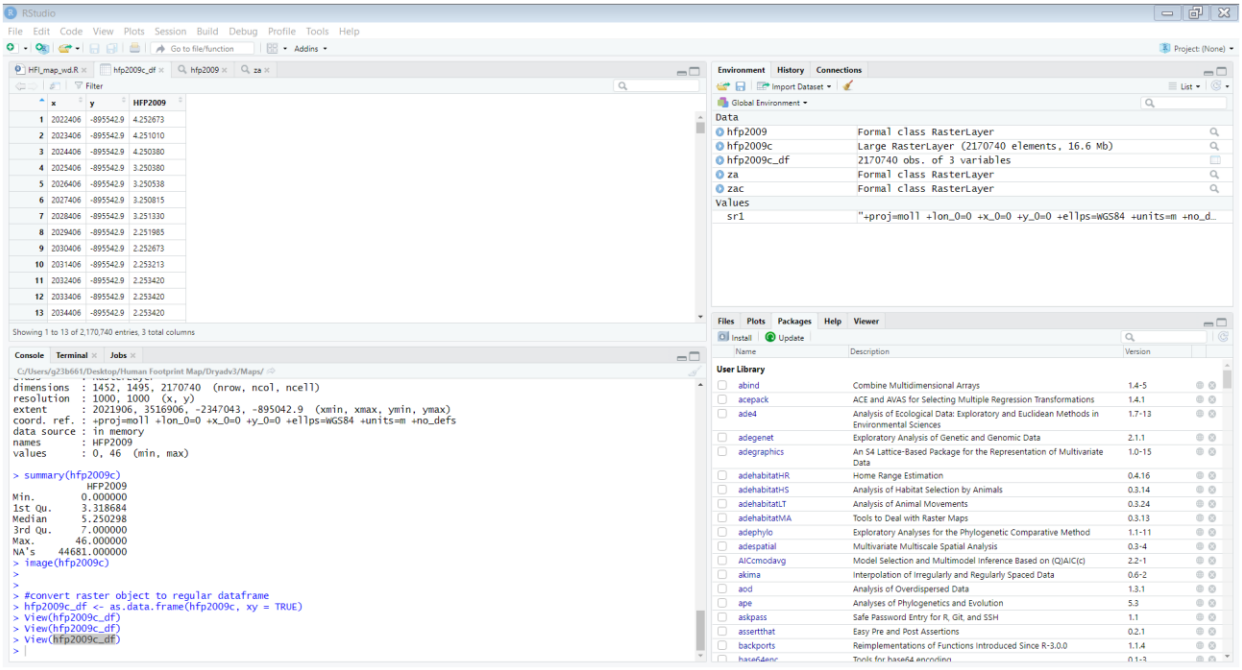
The **BOTTOM LEFT** panel is the **CONSOLE**, where the R code runs, showing each line that is executed and the output of that line, in order. You can type lines of code directly in the console and press enter to run them, *but they will not be part of your script.* This can be useful when error checking or debugging, though I recommend that you just make the habit of writing your code in the script editor and running sections to make sure it is OK using the console (and other panels, see below). Working only in the console basically equivalent to running R without RStudio.

There are two other tabs in this panel, but you will probably never use them.

The **TOP RIGHT** panel is primarily used with the ENVIRONMENT tab. This gives you a list of all the OBJECTS your that the lines of code you have run have created. Just writing a line of code in the script editor does not create an object – you have to run the code. Objects include variables, vectors, matrices, data frames and many other types once you get to more complex R packages.

A DATA FRAME is like an excel spreadsheet. It has variables (columns) and observations (rows). A data frame is also like a matrix (because it is data organized in rows and columns), but a matrix can only hold numeric data, but the variables in a data frame (the columns) can be of different types (character, numeric, factor).

In the screenshot below, some code has been run that created an object named hfp2009c_df. Clicking on that object in the ENVIRONMENT tab in the top right panel causes it to appear in the viewer in the top left panel. You can see that there are multiple tabs open in the **TOP LEFT** panel now – one with the script that I'm working on, and one with the dataframe hfp2009c_df, which is open in this screenshot.
hfp2009c_df



Checking the environment tab to make sure that an object was indeed created, is of the type you intended, and has the dimensions (rows and columns) you expected is critical to checking a script as you write it. Clicking on dataframes in the top right to view them in the top left is a major part of this – makes sure that each stage of a script is OK before moving on to the next step. Save it each time you see that you've made progress.

Remember that the objects in memory (in the environment tab) depend on the lines of code that ave been run. Always starting with the top line of code and running the code in the order it appears in the script is therefore a good idea. If you run code one line at a time but jump up and down the script, code that would otherwise run might not run because an object that is needed to run line 120 does not exist (or has the wrong contents) because you didn't yet run line 119. Even if nothing is wrong with the script, this can create errors.

It also is good practice to have the line

rm(list=ls())

at the top of every script. This removes everything from memory (i.e., the environment tab at top right will have nothing in it) so you are starting fresh and creating the objects you need to do the task you are doing.

It is also good practice when you start a new script to use the 'set working directory option' and select 'to source file location' using the 'session' tab on the top menu. In the console, you see the line of code that sets the working directory to the location where you are saving your script

setwd("C:/Users/g23b661/Desktop/Human Footprint Map/Dryadv3/Maps")

If you then copy this from the console and paste it in the script as the second line, when you save the script it will automatically be written to the same directory location, so you avoid ending up with multiple versions of a script saved in different places.

The **BOTTOM RIGHT** panel has three tabs that you'll jump between often. In the screenshot at the bottom of the previous page, the PACKAGES tab is open. You can use the sub-menu that appears to INSTALL packages (which provide FUNCTIONS that are R code written by others and made available by CRAN). Click the install button, write the name of the package you want to install, make sure the 'install dependencies' box is checked (which will install any other packages you need to run this package) and click install. That package will now appear in the list of installed packages seen in the screenshot above. The R exercises in this class will have you install quite a few useful packages.

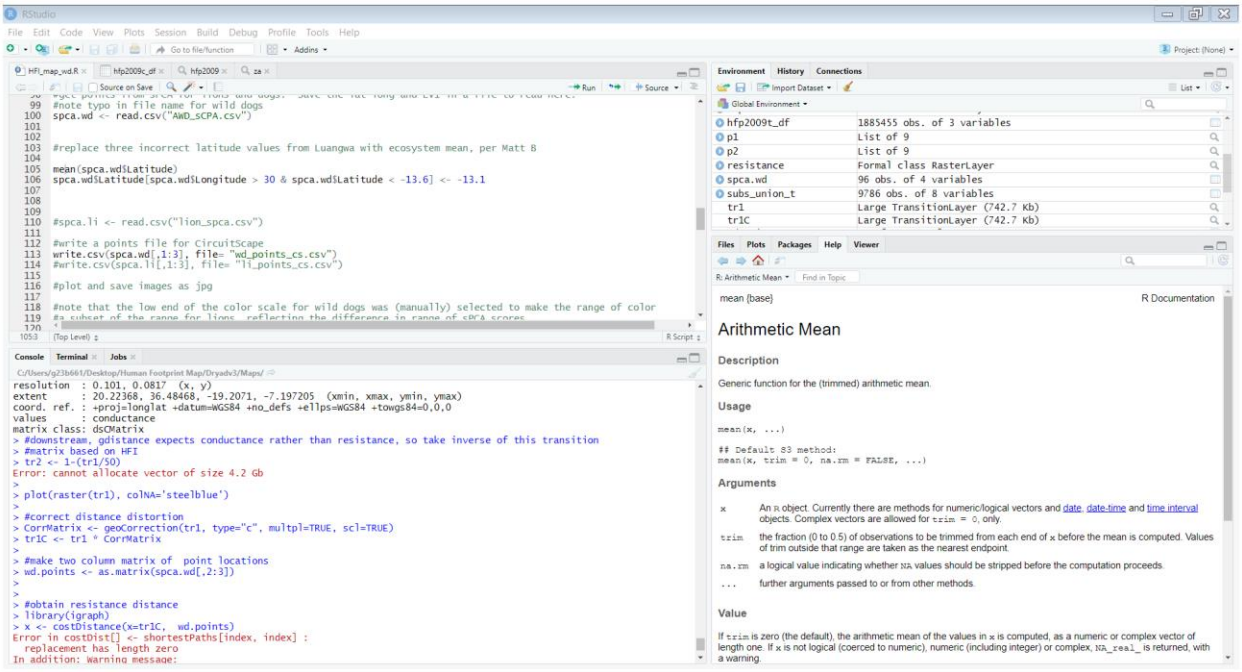
Once a package is installed, you have the functions in that package for future use, but you have to LOAD it in each script that will use it. You load a package by writing a line of code with the library() function and running it:

library(ggplot2)

will load the ggplot2 package, if it has been installed. It will give an error message (in the console) if that package is not installed.

You can also load a package by clicking the check box next to it in the bottom right panel. If you do that, you'll see the library() function run in the console, but that line of code will NOT be in your script. It is therefore good practice to write the line of code into the script, rather than just using the check boxes.

In the screenshot just below, the bottom right panel is set to the HELP tab



This is extremely useful to learn how any function works when you are first using it. A FUNCTION in r is a command that runs hidden, lower-level code that was used to create the function. An example is

mean()

This is a function that will determine the arithmetic mean of the data used as one of the ARGUMENTS of the function. Arguments are the pieces of information that a function needs, or can optionally use but does not necessarily need, to produce its output.

If you put the cursor in a function in the script editor (TOP LEFT) and press F1, the HELP tab in the BOTTOM RIGHT will explain the function, tell you what arguments it requires, and explain any additional arguments that are optional but can be used to tweak how the function works.

I use this all the time.

The screenshot below shows the third main use of the bottom right panel. Some code (visible in the script editor at top left) has been run (in the console, and creating an object named p1 visible in the environment tab at top right) to make a plot. The PLOTS tab shows the output plot, which can be copied, saved or exported using the export tab. (That can also be done using a line of code in the script itself.)