EPFL

Fundamentals in Ecology Practicals

Week 4

Introduction to R

–

Using R

Christoph Bachofen

First steps in R and Rstudio

Topic overview

- Setting up your working environment in RStudio
- Creating projects
- Get to know RStudio: objects and functions
- Installing packages
- Getting help in R and online
- Get to know the syntax for data manipulation

Difference between R and Rstudio

R

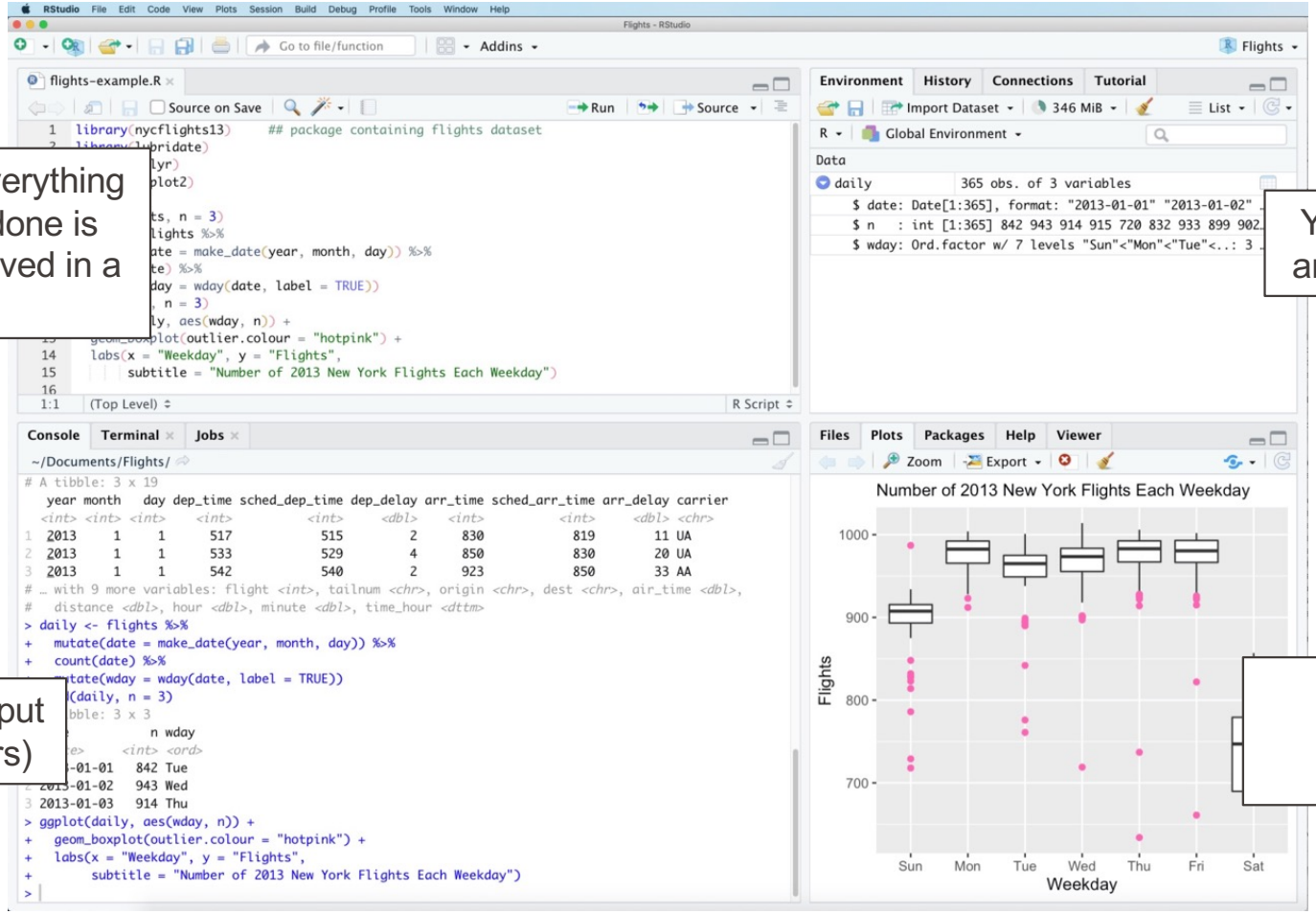
- Command-line tool
- Few utilities
- Not very well arranged
- Therefore there is a GUI (graphical user interface)

RStudio

- Most important R GUI
- Controls R
- R is running in the background!

Rstudio functionalities

Script language: Everything that needs to be done is written here and saved in a R script



Your objects and functions

Here's the output (and the errors)

Figures
Help
Packages

Rstudio functionalities

A) Setting up your working environment

1. Go to Tools > Global Options > General and select following options:
   - 'Restore .RData into workspace at startup' (should NOT be selected)
   - 'Save workspace to .RData on exit' (select 'Never')
2. All coding in RStudio should be done in a script file (or source file). You will soon have multiple script files and several datafiles, and to organise them you should start by creating a project for our R course.
   → create new project
3. Projects are great to keep everything tidy, and it also allows you to run two R sessions at the same time (e.g. if a calculation takes some time you might want to work on another piece of code)
4. Now, create a new folder for the scripts, and one folder for the data.
   → create new folders
5. All the course lectures will get their separate script file that you save in the same script folder.
   → create new script file

Rstudio functionalities

B) Start coding

1. To execute a command, write the code, select the line, then either click on "run" or press cmd+ENTER or ctrl+ENTER → type 2+2
2. Everything in R is either an object or a function. To create an object: assign the value 2 + 2 to a, and see it appearing as an object (numeric) in your environment. → type a <- 2+2
3. Avoid basic mistakes: don't override the existing objects or functions "pi", "mean" or "matrix" or "dataframe" and so forth. Instead use: "treeheight_mean" or "river_data". → type mean
4. Objects need to start with a letter (not a number) and cannot contain spaces. Instead use "_" or "." or use CamelCase. → type 1a <- 2+2
5. R always overwrites variables if you execute the command → type a <- 100
6. Case sensitivity: A is not a → type A to see the error
7. If your command does not work, R returns an error → type 2 +"two" and 2 + two
8. If your command is incomplete R will expect more input → type 2+ and run. There is now a + at the beginning of the line that indicates that R is waiting for more input. Complete it, or escape it with "esc".
   → escape with "esc"
9. Structure your code with headers and descriptions so that you can share code and others understand what you wanted to do. Use the hash key to "outcomment" everything that follows it. → type ### this is my first calculation exercise above the first line of code. Also create a header for the script → type ### Fundamentals in Ecology, Introduction to R, Date

Rstudio functionalities

C) Plots, Help, Packages

1. Next, we are going to use built-in functions that we apply to our objects.
2. When you create figures, they will appear in a new window. → type plot(1)
3. Figures will be kept, so you can scroll backwards through them with the green arrow. → create another figure and scroll between them
4. You can click on zoom to get a larger figure. → click zoom
5. If figures are too large, or the plot window too small, then you get an error. → make figure window too small to plot
6. If you want to get built-in help and example code for a function type ? and the name of the function → type ?sum
7. If you want to install a new package either click on "packages" and then "install" or write "install.packages("your_package_name")" in your source file and execute. → do both