Lesson 02 - Getting set up

Last Updated 06-27-2019

Learning Objectives

After completing this lesson learners will be able to:

- Use R and R Studio on their personal computer.
- Describe the purpose of the RStudio Script, Console, Environment, and Plots panes.
- Organize files and directories for a set of analyses as an R Project, and understand the purpose of the working directory.
- Execute simple commands in the console
- Use the built-in RStudio help interface to search for more information on R functions.
- Demonstrate how to provide sufficient information for troubleshooting with the R user community.

Installing programs

> If you are using an iPad or Chromebook or otherwise do not have a computer that you can install programs on, head over to http://rstudio.cloud.com, make an account and start a new project. Then go watch the video below on how to navigate R studio.

While the cloud is easier to initially setup, I want you to only use it if you absolutely have to. Having your own installation on your computer ensures that

- you likely want to customize your programs
- you will be able to put your files under version control
- you always have access to your code even with unstable or no internet
- Honestly, it may not always be free. This is an alpha testing stage for the company.

R is the programming language that we will be using. R Studio is the program we will *use* R through. You need to install both.

1. Download and install R

- Download R (v3.5+) from https://cran.r-project.org/
  - Direct link for Mac OS X El Capitan or higher: https://cran.r-project.org/bin/macosx/R-3.6.0.pkg
  - Direct link for Windows 10 https://cran.r-project.org/bin/windows/base/R-3.6.0-win.exe
- Install R by double clicking on the downloaded file and following the prompts. Default settings are OK.
  - Delete any desktop shortcut that was created (looks like the icon above.)

2. Download and install R Studio

- Download R Studio (v 1.2+) from https://www.rstudio.com/products/rstudio/download/#download
  - Choose the download link that corresponds to your operating system.
- Install R Studio
  - Windows: Double click on the downloaded file to run the installer program.
  - Mac: Double click on the downloaded file, then drag the R Studio Icon into your Applications folder.
    - After you are done, eject the "Drive" that you downloaded by dragging it to your trash.
- Watch the following short video to learn how to navigate R Studio.

3. Setting preferences in R Studio to retain sanity while debugging

- Open R Studio and go to the file menu go to Tools then Global Options.
- Uncheck "Restore .RData into workspace at startup"
- Where it says "Save workspace to .RData on exit:" Select "Never""
- Click apply then ok to close that window.

This will ensure that when you restart R you do not "carry forward" objects such as data sets that you were working on in a prior assignment.

You should make a habit to completely shut down R studio when you are done working. Your open tabs will be saved, but your environment will be cleared. *This is a good thing.*

4. Organizing your working directory

Using a consistent folder structure across your projects will help keep things organized, and will also make it easy to find/file things in the future. This can be especially helpful when you have multiple projects. In general, you may create directories (folders) for **scripts**, **data**, and **documents**.

You need to choose a naming convention for your class folder and stick with it. Recommended options are:

- ALL CAPS (MATH130)
- no caps (math130)
- snake_case (math_130)
- CamelCase (Math130)

Call this working directory `math130`, and create the three subfolders: `data`, `homework`, `documents`.

You will put all files related to this class in here. For example lecture notes go in the document folder, assignments in the `homework` folder, and data in, you guessed it, the `data` folder.

This means when you download a file, right click and "Save as" or "Save target as" and **actively choose** where to download this file. Do not let files live in your downloads folder.

Do not open any files from your browser window after downloading.

Your working directory should now look like this:

Right click [this link] to download and save Assignment 1 into your `homework` folder now.

Interacting with R

The basis of programming is that we write down instructions for the computer to follow, and then we tell the computer to follow those instructions. We write, or *code*, instructions in R because it is a common language that both the computer and we can understand. We call the instructions *commands* and we tell the computer to follow the instructions by *executing* (also called *running*) those commands.

The console pane is the place where commands written in the R language can be typed and executed immediately by the computer. It is also where the results will be shown for commands that have been executed. You can type commands directly into the console and press `Enter` to execute those commands, but they will be forgotten when you close the session.

Text written like this are instructions for you to do it.

In the console type the following code.

```
2+2
```

You can do basic arithmetic this way. R is basically an overgrown calculator. Try a more complicated equation next.

```
2 + 5*(8^3)- 3*log10)
```

Uh oh, we got an Error. Let's try to fix it.

```
2 + 5*(8^3)- 3*log(10
```

Notice the console shows a `+` prompt. This means that you haven't finished entering a complete command. This is because you have not 'closed' a parenthesis or quotation, i.e. you don't have the same number of left-parentheses as right-parentheses, or the same number of opening and closing quotation marks. When this happens, and you thought you finished typing your command, click inside the console window and press `Esc`; this will cancel the incomplete command and return you to the `>` prompt.

Fix the math expression above by closing the parenthesis.

We'll get into how R sees and uses data this week.

Installing Packages

R is considered an **Open Source** software program. That means many (thousands) of people contribute to the software. They do this by writing commands (called functions) to make a particular analysis easier, or to make a graphic prettier.

When you download R, you get access to a lot of functions that we will use. However these other *user-written* packages add so much good stuff that it really is the backbone of the customizability and functionality that makes R so powerful of a language.

For example we will be creating graphics using functions like `boxplot()` and `hist()` that exist in base R. But we will quickly move on to creating graphics using functions contained in the `ggplot2` package. We will be managing data using functions in `dplyr` and reading in Excel files using `readxl`. Installing packages will become your favorite past-time.

To get started type in the console the following command to install the `rmarkdown` package.

```r
install.packages("ggplot2")
```

When the console returns to showing a `>` (and the stop sign in the top right corner of the Console window goes away), R is done doing what you asked it to do and ready for you to give it another command. You should get a message that looks similar to the message below.

```
The downloaded binary packages are in
    C:\Users\Robin\AppData\Local\Temp\Rtmpi8NAym\downloaded_packages
```

Some packages use functions that live in other packages. These are called dependencies and will be automatically installed when you tell R to install a new package. 99% of the time you will not have to install them manually on your own.

You only have to install packages once. To access commands within the functions you must load the package first using the code `library(package name)`. We will start to use various packages next week.

Now that you're a package installing pro, go ahead and install the following packages that we will be using in the next few weeks. Type these commands into the console one at a time and wait for it to finish before entering the next command. **R is case sensitive**.

```r
install.packages("rmarkdown")
install.packages("knitr")
install.packages("forcats")
install.packages("readxl")
install.packages("dplyr")
```

Note in this last package that is a lower case letter L at the end. For read EXCELL. It is not the number 1 (one).

If you see a message such as the one below, check for a typo.

```
Warning in install.packages :
  package ‘ggplot’ is not available (for R version 3.5.1)
```

The correct package name is `ggplot2`, not `ggplot`.

Alternative Method of installing Packages: Use the Package tab in the lower right corner.

Seeking help

Use the built-in RStudio help interface to search for more information on R functions

One of the fastest ways to get help, is to use the RStudio help interface. This panel by default can be found at the lower right hand panel of RStudio. As seen in the screenshot, by typing the word "Mean", RStudio tries to also give a number of suggestions that you might be interested in. The description is then shown in the display window.

I know the name of the function I want to use, but I'm not sure how to use it

If you need help with a specific function, let's say `barplot()`, you can type:

```r
?barplot
```

If you just need to remind yourself of the names of the arguments, you can use:

```r
args(lm)
```

I want to use a function that does X, there must be a function for it but I don't know which one...

If you are looking for a function to do a particular task, you can use the `help.search()` function, which is called by the double question mark `??`. However, this only looks through the installed packages for help pages with a match to your search request

```r
??kruskal
```

If you can't find what you are looking for, you can use the rdocumentation.org website that searches through the help files across all packages available.

Finally, a generic Google or internet search "R <task>" will often either send you to the appropriate package documentation or a helpful forum where someone else has already asked your question.

I am stuck... I get an error message that I don't understand

Start by googling the error message. However, this doesn't always work very well because often, package developers rely on the error catching provided by R. You end up with general error messages that might not be very helpful to diagnose a problem (e.g. "subscript out of bounds"). If the message is very generic, you might also include the name of the function or package you're using in your query.

However, you should check Stack Overflow. Search using the `[r]` tag. Most questions have already been answered, but the challenge is to use the right words in the search to find the answers: http://stackoverflow.com/questions/tagged/r

---

Where to get help

The key to receiving help from someone is for them to rapidly grasp your problem. You should make it as easy as possible to pinpoint where the issue might be.

Try to use the correct words to describe your problem. For instance, a package is not the same thing as a library. Most people will understand what you meant, but others have really strong feelings about the difference in meaning. The key point is that it can make things confusing for people trying to help you. Be as precise as possible when describing your problem.

In Person

- The person sitting next to you. Don't hesitate to talk to your neighbor, compare your answers, and ask for help.
- Your friendly classmates: if you know someone with more experience than you, they might be able and willing to help you.
- Data Science Initiative
  - http://datascience.csuchico.edu What the Data Science community at Chico is up to.
  - Community Coding.
    - Drop in work session & dedicated space to work on coding projects.
    - Collaborate with your peers and learn from experts.
    - Need 1 unit? Enroll in Math 290-02 and attend at least 10 times.
    - You DON'T need to be enrolled to attend!
  - Weekly seminars and workshops in a variety of languages and topics.
- The R Users Meetup group useful if you want to stay connected to the community and learn about upcoming events.
  - https://www.meetup.com/Chico-R-Users-Group/

Sometimes a second pair of eyeballs is all you need

Online

- In RStudio go to `Help` –> `Cheatsheets`

- Stack Overflow: if your question hasn't been answered before and is well crafted, chances are you will get an answer in less than 5 min. Remember to follow their guidelines on how to ask a good question.
- Chico R Users Google Group.
- The R-Studio Community: it is read by a lot of people and is more welcoming to new users than the R list-serv.
- If your question is about a specific package, see if there is a mailing list for it. Usually it's included in the DESCRIPTION file of the package that can be accessed using `packageDescription("name-of-package")`. You may also want to try to email the author of the package directly, or open an issue on the code repository (e.g., GitHub).
- Twitter: #rstats @Rstudio

Written

If you're a book kinda person, there is plenty of help available as well. Many have online versions or free PDF's.

- R Markdown, the Definitive Guide: https://bookdown.org/yihui/rmarkdown/
- R for Data Science https://r4ds.had.co.nz/
- Cookbook for R http://www.cookbook-r.com/
- R Graphics Cookbook (I use this all the time) – Chapter 8 in the above link
- The Art of R Programming https://nostarch.com/artofr.htm
- R for… http://r4stats.com/
  - Excel Users https://www.rforexcelusers.com/
  - SAS and SPSS Users http://r4stats.com/books/r4sas-spss/
  - STATA Users http://r4stats.com/books/r4stata/

More resources

- The Posting Guide for the R mailing lists.
- How to ask for R help useful guidelines
- This blog post by Jon Skeet has quite comprehensive advice on how to ask programming questions.
