# Lab 1-1: Getting started with R and R markdown

PSTAT 131/231, Fall 2021

**Learning Objectives**

- Complete installation of R and RStudio
- First steps using RStudio
- Getting help in R
- Installing packages
- Using R script files
- Using Rmd files
- Get to know markdown syntax
- R code chunk options

---

## R and RStudio

- Make sure everybody has installed **R**
  - R for Mac: https://cran.r-project.org/bin/macosx/
  - R for windows: https://cran.r-project.org/bin/windows/base/
- Make sure everybody has installed **RStudio**
  - RStudio download: https://www.rstudio.com/products/rstudio/download/
- Difference between R-GUI and RStudio
  - R-GUI is a simply graphical user interface
  - RStudio is an *Integrated Development Environment* (IDE)
    - It is much more than a simple GUI
    - It provides a nice working environment and development framework
- We are going to use mainly RStudio
- Keyboard shortcuts will increase your productivity significantly:

  https://support.rstudio.com/hc/en-us/articles/200711853-Keyboard-Shortcuts

---

## More about RStudio

You will be working with RStudio a lot, and you will have time to learn most of the bells and whistles RStudio provides. Think about RStudio as your "workbench". Keep in mind that RStudio is MORE than plain R. RStudio is an environment that makes it easier to work with R, while handling many of the little tasks than can be a hassle.

### A quick tour of RStudio

- Understand the **pane layout** (i.e. windows) of RStudio
  - Source
  - Console
  - Environment, History, etc
  - Files, Plots, Packages, Help, Viewer
- Customize RStudio Appearance of source pane
  - font
  - size
  - background

### Using an R script file

Most of the times you won't be working directly on the console. Instead, you will be typing your commands in some source file. The basic type of source files are known as *R script files*. Open a new script file in the *source* pane, and rewrite the previous commands.

You can copy the commands in your source file and paste them in the console. But that's not very efficient. Find out how to run (execute) the commands (in your source file) and pass them to the console pane.

### Getting help

Because we work with functions all the time, it's important to know certain details about how to use them, what input(s) is required, and what is the returned output.

There are several ways to get help.

If you know the name of a function you are interested in knowing more, you can use the function `help()` and pass it the name of the function you are looking for:

```r
# documentation about the 'abs' function
help(abs)

# documentation about the 'mean' function
help(mean)
```

Alternatively, you can use a shortcut using the question mark `?` followed by the name of the function:

```r
# documentation about the 'abs' function
?abs

# documentation about the 'mean' function
?mean
```

- How to read the manual documentation
  - Title
  - Description
  - Usage of function
  - Arguments
  - Details
  - See Also
  - Examples!!!

`help()` only works if you know the name of the function your are looking for. Sometimes, however, you don't know the name but you may know some keywords. To look for related functions associated to a keyword, use `help.search()` or simply `??`

```r
# search for 'absolute'
help.search("absolute")

# alternatively you can also search like this:
??absolute
```

Notice the use of quotes surrounding the input name inside `help.search()`

### Installing Packages

R comes with a large set of functions and packages. A package is a collection of functions and datasets that have been designed for a specific purpose. One of the great advantages of R is that many analysts, scientists, programmers, and users can create their own packages and make them available for everybody to use them. R packages can be shared in different ways. The most common way to share a package is to submit it to what is known as **CRAN**, the *Comprehensive R Archive Network*.

You can install a package using the `install.packages()` function. Just give it the name of a package, surrounded by quotes, and R will look for it in CRAN, and if it finds it, R will download it to your computer.

```r
# installing
install.packages("knitr")
```

You can also install a bunch of packages at once:

```r
install.packages(c("readr", "ggplot2"))
install.packages(c("hflights", "tidyverse"))
```

Once you installed a package, you can start using its functions by *loading* the package with the function `library()`

```r
library(knitr)
```

You can also install an R package from Rstudio directly, via **Tools** –> **Install package**.

### Your turn 1

```r
# Look for the man (i.e. help) documentation of the function `exp`

# Find out how to look for information about binary operators like `+` or `^`
```

---

## Introduction to Markdown

Besides using R script files to write source code, you will be using other type of source files known as *R markdown* files. These files use a special syntax called **markdown**.

### Get to know the `Rmd` files

In the menu bar of RStudio, click on **File**, then **New File**, and choose **R Markdown**. Select the default option (Document), and click **Ok**.

**Rmd** files are a special type of file, referred to as a *dynamic document*, that allows to combine narrative (text) with R code. Because you will be turning in homework assignments as `Rmd` files, it is important that you quickly become familiar with this resource.

In order to **Knit PDF**, you need to install LaTeX distribution. If you try this option without LaTeX, you will be greeted with the following error:

```
No TeX installation detected (TeX is required to create PDF output).
You should install a recommended TeX distribution for your platform:

  Windows: MiKTeX (Complete) - http://miktex.org/2.9/setup
  (NOTE: Be sure to download the Complete rather than Basic installation)

  Mac OS X: TexLive 2013 (Full) - http://tug.org/mactex/
  (NOTE: Download with Safari rather than Chrome _strongly_ recommended)

  Linux: Use system package manager
```

Locate the button **Knit HTML** (the one with a knitting icon) and click on it so you can see how `Rmd` files are rendered and displayed as HTML documents.

**Knit PDF** is preferred for this class (much more flexible); however, if you do not wish to install LaTeX system, you may **Knit HTML** and print to pdf.

### Yet Another Syntax to Learn

R markdown (`Rmd`) files use markdown as the main syntax to write content. It is a very lightweight type of markup language, and it is relatively easy to learn.

### On your own time

Work through the markdown tutorial: www.markdown-tutorial.com

Your turn: After lab discussion, find some time to go through this additional markdown tutorial www.markdowntutorial.com/

RStudio has a very comprehensive R Markdown tutorial: Rstudio markdown tutorial

---

## R Markdown Code Chunks

The general format of an Rmd code chunk is as shown below

`{r chunk_name, chunk_options}`

### Option 1: `include=FALSE`

Here we just set up some knitr options we want the code to be *evaluated* but **neither code or output to be** ***displayed***:

The code that was hidden in the final knit document was:

```
library(knitr)
```

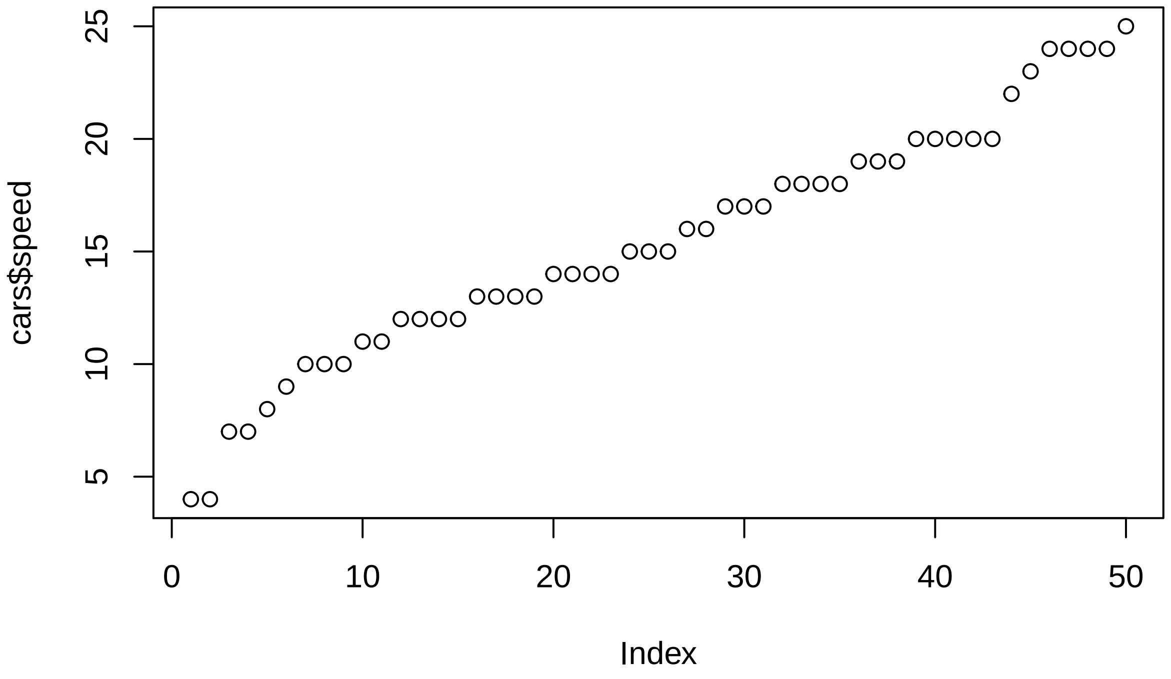
### Option 2: `results='hide'`

Here we just set up some knitr options we want the code to be *evaluated* but **the output not to be** *displayed*:

```r
summary(cars)
```

### Option 3: `echo=FALSE`

Here we just set up some knitr options we want the code to be *evaluated* but **only the output to be** *displayed*:



### Option 4: `eval=FALSE`

Here we just set up some knitr options we want the code to **not be evaluated** but **only the code to be** *displayed*:

```r
min(cars$dist)
```

We didn't get the minimum value of the `distance` variable in `cars` dataframe displayed as expected here, just the R code command

### Option 5: `cache=TRUE`

Here if the code chunk remains unchanged between 'knitting', then it won't be re-run in the next knit i.e. RStudio will cache (save) the output from the previous knit and not re-run.

This is used to **speed up the R Markdown development process** by reducing the time for each subsequent knit process.

If caching is not behaving in a manner you expect, consult this page to troubleshoot: https://yihui.name/knitr/demo/cache/

```r
max(cars$dist)
```

```
## [1] 120
```

#### Option 6: `warning=FALSE, message=FALSE`

Here `warnings` and `messages` of `R` output are suppressed. By default they are not. In general you should not require this but there are some special cases where it is useful.

```r
library(knitr)
```

#### Option 7: `Preferred for PSTAT 131/231`

In `PSTAT 131/231` we (i.e. Instructor and TAs) want to see your code and output displayed in your submissions. So there should be no need to have `echo=FALSE` or `results='hide'` or `eval=FALSE` (using `cache=TRUE` is up to you):

`{r chunk_name}`

or

`{r chunk_name, cache=TRUE}`

#### Other options

There are many other options for other functionality: e.g. plotting. You may use them when appropriate. Here are some resources to help you:

- Karl Borman's Rmd tutorial
- Yihue Xie's Rmd Chunk Options page

#### Your turn 2

```r
# create the following variables
# x = seq(-1, 2, length.out=100)
# y = x^3 - 0.5*x^2 - x + 0.1
```

#### Your turn 3

```r
# plot x and y, and set chunk option to hide the plot command
# plot(x, y, type='o', pch=16, cex=0.5)
```

---