Introduction to R, R Studio and R Markdown

R

R can be obtained for free from CRAN

Important thing to know about R

R is case-sensitive! So a variable named `x` is not the same as a variable named `X`.

Getting help

You can get help and learn more about R through mailing lists and websites such as Stack Overflow, or on the RStudio community. There are hundreds of blogs about R and these are all compiled in one place at R bloggers. Many people make code available on-line. You can modifying existing code to suit your purpose but if you do so, be sure to cite or acknowledge where you found the code. You may not think that I would know that you copied code from online, but I do and it is not ok to try to pass someone else's code off as your own.

One of the most 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.

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

```
?barplot
```

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 and rseek.org is a search engine that is specific to R.

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.

R packages

There are literally thousands of packages for R. In order to use a package for the first time, you need to download and install it, either from CRAN, Bioconductor, or GitHub. One way to learn about the various packages that are available is to look at the CRAN Task Views.

The following will install packages that are available on CRAN.

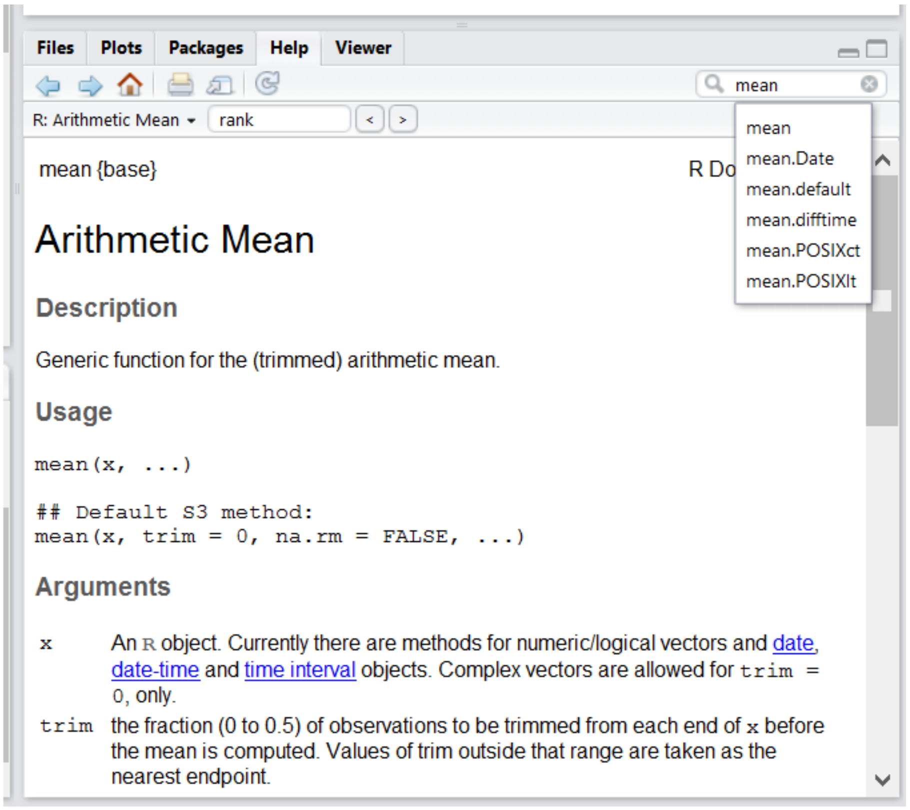
```
install.packages("mosaicData")
```

The above step only needs to be done the first time. The step below must be done each time you want to use the package in a new R session.

```
library(mosaicData)
```

This is a package that includes some data sets. If you have loaded a package containing data files, then you can access that data by it's name. For example, there is a data file called `HELPfull` in the `mosaicData` package.

More likely, data will be on your computer and you will want to read it into R. .csv files can be read into R using `read.csv()` and ascii (i.e., .txt files) can be read into R using `read.table()`



mean {base}   R Do

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

`x`	An `R` object. Currently there are methods for numeric/logical vectors and date, date-time and time interval objects. Complex vectors are allowed for `trim = 0`, only.
`trim`	the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.

Figure 1: RStudio help interface

```r
csv <- read.csv("help.csv")
```

Creating objects in R

To create an object, we need to give it a name followed by the assignment operator, `<-`, and the value we want to give it:

```r
weight_kg <- 55
```

`<-` assigns values on the right to objects on the left. So, after executing `weight_kg <- 55`, the value of `weight_kg` is `55`. The arrow can be read as 55 **goes into** `weight_kg`. For historical reasons, you can also use `=` for assignments, but not in every context. Because of the slight differences in syntax, it is good practice to always use `<-` for assignments.

Object names cannot start with a number (`2x` is not valid, but `x2` is). There are some names that cannot be used because they are the names of fundamental functions in R (e.g., `if`, `else`, `for`, see here for a complete list).

Comments

The comment character in R is `#`, anything to the right of a `#` in a script will be ignored by R. It is useful to leave notes, and explanations in your scripts so that other people, including your future self, can understand your code.

R Studio

RStudio is an Integrated Development Environment (IDE) for working with R. Renewed as "Posit" in November 2023, but still uses R Studio.

RStudio is divided into 4 "Panes": the **Source** for your scripts and documents (top-left, in the default layout), the R **Console/Terminal/R Markdown** (bottom-left), your **Environment/History/Connections** (top-right), and your **Files/Plots/Packages/Help/Viewer** (bottom-right). The placement of these panes and their content can be customized (see menu, Tools -> Global Options -> Pane Layout). One of the advantages of using RStudio is that with many keyboard shortcuts, autocompletion, and highlighting, RStudio will make typing R code easier and less error-prone.

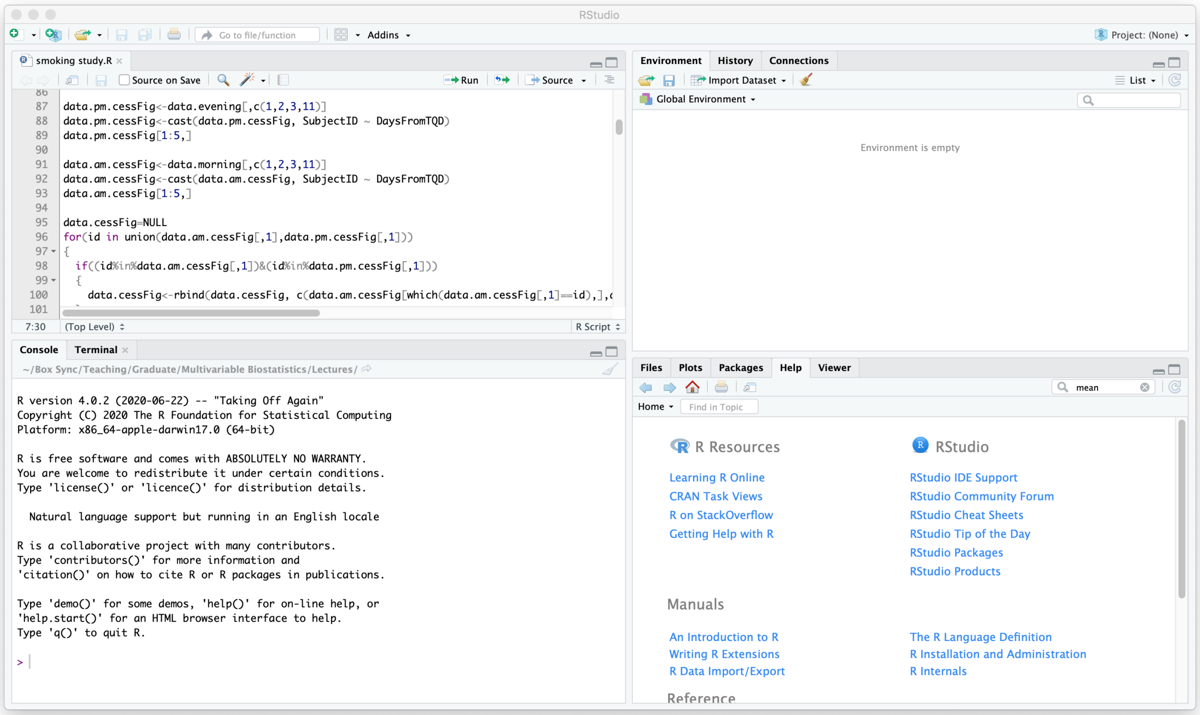
R Markdown

R Markdown allows you to write a report with R code and the output from running that code. To start a new R Markdown document, open it in R Studio and save as an .Rmd file. To "knit" the document and run all the R code embedded in it, click the button that says **Knit**. The R code chunks are then extracted, executed, and replaced by their output, be it text, tables, or figures.

You can specify things like title, author and date in the **header** of your R Markdown file. This goes at the very beginning of the file, preceded and followed by lines containing three dashes. Thus the beginning of this file looks like so:

```
---
title: "Introduction to R, RStudio, and R Markdown"
author: "Name"
date: "8/28/2023"
---
```

You can also use the header to tell R Markdown whether you want it to knit to HTML (the default), pdf, or Word. Note that you need to have LaTeX installed to knit to a pdf.



```
data.pm.cessFig<-data.evening[,c(1,2,3,11)]
data.pm.cessFig<-cast(data.pm.cessFig, SubjectID ~ DaysFromTQD)
data.pm.cessFig[1:5,]

data.am.cessFig<-data.morning[,c(1,2,3,11)]
data.am.cessFig<-cast(data.am.cessFig, SubjectID ~ DaysFromTQD)
data.am.cessFig[1:5,]

data.cessFig=NULL
for(id in union(data.am.cessFig[,1],data.pm.cessFig[,1]))
{
  if((id%in%data.am.cessFig[,1])&(id%in%data.pm.cessFig[,1]))
  {
    data.cessFig<-rbind(data.cessFig, c(data.am.cessFig[which(data.am.cessFig[,1]==id),],
```

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Figure 2: Figure: R Studio interface screenshot

Code Chunks

A code **chunk** is simply an off-set piece of code by itself. It is preceded by ` ```{r} ` on a line by itself, and ended by a line which just says ` ``` `. The code itself goes in between. The first code chunk is code that sets up some options. You do not need to change it from the template.

After the setup chunk, you insert chunks to include code in the appropriate places within your reports. Here, for instance, is some code which loads a data set that is included with R and generates some summary statistics.

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

To prevent printing of the R code, you can add `echo = FALSE` to the code chunk.

To not include the output of a code chunk and the code itself, you can add `include = FALSE` as in the setup code block. To not include warning messages, you can add `warning=FALSE` as in the following block of code that loads packages.

```
library(car)
```

```
## Loading required package: carData
```

```
library(ggplot2)
```

If you are emailing me for help, **always include the output of `sessionInfo()`** as it provides critical information about your platform, the versions of R and the packages that you are using, and other information that can be very helpful to understand your problem. Insert it as a code chunk at the end of your R Markdown file and include it in every homework submission.

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur ... 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggplot2_3.5.1     car_3.1-2         carData_3.0-5     mosaicData_0.20.3
##
## loaded via a namespace (and not attached):
##  [1] rstudioapi_0.14  knitr_1.43       magrittr_2.0.3   tidyselect_1.2.0
##  [5] munsell_0.5.0    colorspace_2.1-0 R6_2.5.1         rlang_1.1.1
##  [9] fastmap_1.1.1    fansi_1.0.4      dplyr_1.1.2      tools_4.2.2
## [13] grid_4.2.2       gtable_0.3.3     xfun_0.39        utf8_1.2.3
## [17] cli_3.6.1        withr_2.5.0      htmltools_0.5.5  abind_1.4-5
## [21] yaml_2.3.7       digest_0.6.31    tibble_3.2.1     lifecycle_1.0.3
## [25] vctrs_0.6.2      glue_1.6.2       evaluate_0.21    rmarkdown_2.22
## [29] compiler_4.2.2   pillar_1.9.0     generics_0.1.3   scales_1.3.0
## [33] pkgconfig_2.0.3
```

Section Headers in R Markdown

The character `#` at the beginning of a line means that the rest of the line is interpreted as a section header. The number of `#`s at the beginning of the line indicates whether it is treated as a section, sub-section, etc. I draw your attention to this because within R code chunks in R Markdown, `#` is the comment character as described in the introduction to R.

Environments

Finally, and this is important, R Studio keeps *two* environments or workspaces which it uses to evaluate R code. One is the "usual" global environment of the console, which builds cumulatively from the start of your session. Every time you knit, however, it re-runs your code in clean workspace, as though you had just started R from scratch. This means knitted code evaluates *only* code in the .Rmd file. This can lead to a situation in which your code worked in the console but it won't knit because you left something important out of your code in the .Rmd file.

Another important tip is to keep everything in the same folder (e.g., keep data files in the same folder as the .Rmd file). If you are having problems knitting, try completely closing RStudio and then restart it and try to knit again.