Intro to R and RStudio

Strauss Health Sciences Library

June 22, 2021

Note: This is material adapted largely from Software Carpentry and Data Carpentry lessons as well as from material from Tobin Magle. We're always on the lookout for more relevant and directly applicable datasets and use cases for the demonstrations, so feel free to contact us if you have any...

Learning Objectives

After completing this tutorial you will be:

- Familiar with the RStudio interface and documentation
- Familiar with core aspects of the base R syntax
- Able to load tabular data into R
- Able to calculate summary statistics for tabular data
- Able to create a publication-quality graph

Preflight

If you haven't already, install R and RStudio on your computer:

Instructions for installing R: https://cran.r-project.org

Instructions for installing RStudio: https://www.rstudio.com/products/rstudio/download/#download

You may also download the toy datasets we'll be working with:

E.Coli strains data: https://osf.io/ga9re/

Inflammation data: https://osf.io/c6r3d/

Note where these two data sets end up in your filesystem

What is R?

R is a programming language originally designed for statistical computing. It has grown into a general scientific computing platform thanks largely to it being Open Source and to the gargantuan number of extensions available for it. Just about any feature you would like in a scientific computing environment likely has been implemented and is continuously being improved. (e.g. the Bioconductor package for genomics, the RISmed package for interacting with NCBI databases, and the RQDA package for Qualitative Data Analysis).

What is RStudio?

RStudio makes programming in R easier. RStudio is an IDE (Integrated Development Environment) for R that includes all relevant documentation and syntax checking at the ready as well as tools for quick data exploration and document preparation.

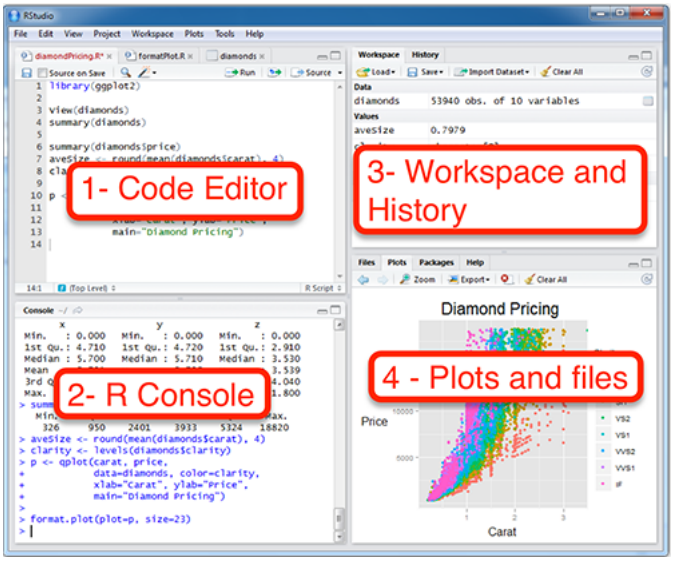

Figure 1: Default RStudio Interface

RStudio can also be a tool for reproducible research. That is, we can use RStudio as a sort of laboratory notebook in case we forget where we put things or how we did things, something that is very helpful given the sprawling capabilities of the R platform.

Getting Started

Set your working directory

To make things easier, it helps to have your code and data files together. First, choose a folder as your working directory:

Option 1: - Go to the **Session** menu - Select **Set Working Directory** - Select **Choose Directory** and navigate to where your files are

Option 2: - Use the **More** pull-down menu in the **Files** pane

Option 3: - Use the `setwd()` and `getwd()` commands in the console

Open a Script file

We will do most of our work today on the R console. However, we could also save our code as a script. To open a new script:

- Go to the **File** menu
- Select **New File**
- Select **R Script**

Now save the script to where you want to work. And remember to Always Be Saving.

Basic Operations

R sees the world in terms of vectors and tables. However, this doesn't preclude us to use it as a simple scientific calculator

chunk 1

```
sqrt(3.5)
print(sqrt(3.5), digits=20)
```

To assign values to variables, we use R's own assignment operator:

chunk 2

```
x <- 2 # Assigns 2 to x; really x is a 1-element vector now
# To find out the value or content of a variable or object, we just input its name
x
```

Fun thing to check: the assignment operator is meant to evoke an arrow. Does it work in both directions? Can you conceive of reasons or cases where it may be preferable to choose one direction versus the other while writing code?

Since R thinks in terms of vectors and tables, it is straightforward to perform vectorized calculations. Loops are also supported (see cheatsheet) but it is good practice to take advantage of vectorization when possible.

chunk 3

```
x <- c(1,2,3,4,5,6) #Creates vector; `c` stands for `concatenate`
y <- x^2
z <- 100:140 #Also easy to create range vectors
jenny <- c(8,6,7,5,3,0,9)
(jenny *3)
```

Quick exercise, can you tell what `b` will be?

chunk 4

```
a <- c(1,2,3,4)
b <- exp(log(a))
```

R is full of more sophisticated functions, but you can also define your own:

chunk 5b

```
middle <- function(a,b) {
  # Dumb function that gives the average of two numbers
  return((a+b)/2)
}

# Let's try it
middle(20,40)
```

Quick exercise: code a function to convert temperatures from Fahrenheit to Kelvin

Disgression: R also has the basic control structures you'd expect from a more "imperative" language. See the "Programming" section of the *Base R* handout sheet.

Accessing help

We can get quick access to the documentation by prefixing command names or concepts with question marks.

Example 1: I forgot the syntax for some commands

chunk 6

```
?mean #gets documentation about the mean function
?plot #gets documentation about the plot function
```

In what follows, feel free to type into the console the commands above and run them instantly or, as intended, type them as part of the script you are saving. Then select the lines you want to run and press the Run button in RStudio. Use `#` to write comments and notes liberally in your script.

Example 2: How do I find documentation on regression in R?

Let's start playing with our datasets

You should have `ecolidata.csv` in your working directory.

Note that we could have downloaded it directly from RStudio with the following command:

chunk 8

```
download.file(url="https://osf.io/ga9re/download", destfile="ecolidata.csv")
```

We could have also imported an actual Excel file with the `readxl` or a SAS file with the `haven` package, we'll talk about that in a bit...

Before we explore our data, we need to import the csv file into an R table or *Data Frame*:

chunk 9

```
straindata <- read.csv("ecolidata.csv")
```

Data frame is a special R table format, e.g. distinct from a matrix: - Each column a variable - Each row an observation - Each row in a column is of the same data type - Columns can be of different data type - Data set is rectangular

Try exploring the imported data from the **Environment** pane (View) or with R commands:

chunk 10

```
str(straindata) # tells you data types
head(straindata)
class(straindata)
```

A data frame is a collection of vectors (columns) of identical lengths

chunk 11

```
dim(straindata)
nrow(straindata)
ncol(straindata)
summary(straindata)
```

Indexing and subsetting

chunk 12

```
straindata[1,2] # First element of second column
straindata[1:3, 7] # First 3 elements of 7th column
straindata[3,] # third element of everycolumn
straindata[,7] # the seventh column
```

Adressing by column name

chunk 13

```
straindata$generation # The "generation column"
```

Creating new objects from existing ones

chunk 14

```
sizematrix <- cbind(straindata[,1],straindata[,7]) # makes a table from columns

rowmatrix <- rbind(straindata[5,],straindata[8,]) # makes a table from rows

sample_size <- data.frame(straindata$sample, straindata$genome_size) # another way of making a table fr

write.csv(sample_size,file="sample_size")
```

Logical Operations

chunk 15

```
sample_check <- read.csv("sample_size")

AreTheyTheSame <- sample_size == sample_check
(AreTheyTheSame)
```

Filtering data

chunk 16

```
FirstFive <- sample_size$straindata.genome_size[1:5]
(FirstFive)

biggergenomes <- subset(straindata, genome_size > 4.7)
```

Basic Plotting

Let's make some plots of this bacterial strains data:

chunk 17

```
plot(straindata$genome_size) #Scatterplot
```

We have some options in dictating presentation, of course:

chunk 18

```
plot(straindata$genome_size, main="Scatter Plot of Genome Sizes")
plot(straindata$genome_size, main="Scatter Plot of Genome Sizes", ylab="Genome Size (Mb)")
```

Some other plots:

chunk 19

```
plot(straindata$generation, straindata$genome_size)

hist(straindata$genome_size) # Histogram

boxplot(straindata$genome_size ~ straindata$cit) # Box Plot
boxplot(genome_size ~ cit, straindata)
boxplot(genome_size ~ cit, straindata, col=c("pink", "purple", "darkgrey"),
        main="Average Expresion Differences", ylab="Genome Size")
```

Let's move now to that Inflammation data

Similar to the previous dataset, you should have `inflammation.csv` in your working directory or:

chunk 20

```
download.file(url = "https://osf.io/c6r3d/download", destfile="inflammation.csv")
```

Quick import and exploration:

chunk 21

```
inflammation <- read.csv("inflammation.csv") # reads data from the csv file

head(inflammation)
str(inflammation)
summary(inflammation)
```

Subsetting Columns

chunk 22

```
day10inf <- inflammation$day10
mean(day10inf)
```

Subsetting Rows

chunk 23

```
males <- subset(inflammation, gender =="M")
females <- subset(inflammation, gender =="F")
```

Basic stats

chunk 24

```
t.test(females$day10, males$day10)
```

Plot a Histogram

chunk 25

```
hist(inflammation$day30, main="Day 30 Inflammation", xlab="Inflammation Level",
     ylab="Frequency", col="blue")
```

Bar Plot

chunk 26b

```
daymeans <- apply(inflammation[,3:32],MARGIN=2,FUN=mean)
barplot(height=daymeans, main="Mean Inflammation Over Time", xlab="Days Post-Vaccine",
        ylab="Mean Inflammation")
```

Quick exercise: how to modify the code above to make a barplot of the median inflammation instead?

Some other Basic Plots

chunk 27

```
boxplot(inflammation$day21 ~ inflammation$gender, main="Day 21 Inflammation by Gender",
        xlab="Gender")

plot(inflammation$day10)

plot(inflammation$day10, xlab="Patient ID", ylab="Inflammation",
     col = inflammation$gender, pch=16)

plot(inflammation$day10, col=inflammation$gender, pch=16)
legend(x=3, y=5, legend=levels(inflammation$gender), col=c(1:2), pch=16)
```

Further Steps

Packages

Packages are a powerful way to expand the functionality of your R install.

This will download the `readxl` package from the archives:

```
install.packages("readxl")
```

Once installed, the functions in `readxl` are available to you provided you declare that you want to use the installed package from a script or from the console with:

```
library(readxl)
```

Helpful Links:

- Swirl http://swirlstats.com/students.html
- Learnr Tutorials (see *Tutorial* tab on upper-right panel of RStudio)
- R base graphics: an Idiot's guide https://rpubs.com/SusanEJohnston/7953

For some more recommended introductory tutorials:

- Kelly Black's R Tutorial http://cyclismo.org/tutorial/R/index.html
- Software Carpentry R Programming Lesson http://swcarpentry.github.io/r-novice-inflammation/

Some of the best books to learn R and RStudio are **free**:

Hands-On Programming with R (https://rstudio-education.github.io/hopr/) "This book will teach you how to program in R, with hands-on examples. I wrote it for non-programmers to provide a friendly introduction to the R language. You'll learn how to load data, assemble and disassemble data objects, navigate R's environment system, write your own functions, and use all of R's programming tools. Throughout the book, you'll use your newfound skills to solve practical data science problems."

Once you are a bit familiar with R and RStudio you can dive into the most useful data science packages by going through this more advanced book:

R for Data Science (https://r4ds.had.co.nz) "This book will teach you how to do data science with R: You'll learn how to get your data into R, get it into the most useful structure, transform it, visualise it and model it. In this book, you will find a practicum of skills for data science."... "You'll learn how to use the grammar of graphics, literate programming, and reproducible research to save time. You'll also learn how to manage cognitive resources to facilitate discoveries when wrangling, visualising, and exploring data."

Other web resources:

- RStudio downloadable cheatsheets: https://www.rstudio.com/resources/cheatsheets/
- Videos from rstudio https://rstudio.com/resources/webinars/
- R blog aggregator: https://www.r-bloggers.com/
- A curated list of the better R packages: https://github.com/qinwf/awesome-R