Introduction to R/RStudio

Table of Contents

## Learning Outcomes

At the end of this session you should be able to:

* Perform basic arithmetic using R/RStudio
* Identify the major components of the RStudio interface
* Create an Rmd file and use it to document a sequence of R functions
* Read in a data set and perform basic data summary procedures in R

## Getting started

For this session, you can either work in RStudio on your own computer (R/RStudio install instructions can be found [here](https://www.rstudio.com/products/rstudio/download/#download)) or you can work in an RStudio Cloud session. To start an RStudio Cloud session go to <https://rstudio.cloud/> and used your prefered log in method, or create a new account.

### Difference between R and RStudio

In this workshop, we will be working primarily in RStudio. So what is the difference between R and RStudio? R is a programming language (specifically a statistical analysis programming language), while RStudio is an **Integrated Development Environment**, or **IDE**. You can think of it as a interface with R that aids the analyst. RStudio offers a number of features, mostly related to visual presentation of information, that make writing and working with R code easier.

### Overall layout

There are four panels in the RStudio interface (though you may only have three open when you first start it), each has valuable information.

* Console / Terminal panel (lower-left)
* Environment / History / Git (upper-right)
* Files / Plots / Packages / Help (lower-right)
* Source / Editor (upper-left)

### File management

Good file management is important in data science. We’ll start working on this now.

* Make a DataSci directory / folder
* Make a data directory / folder within DataSci
* Make a scripts directory / folder within DataSci
* In RStudio, set working directory to your new DataSci directory

#### Setting your working directory

Point-and-click method - Use ‘Session’ > ‘Set Working Directory’ > ‘Choose Directory’.

Using the R Console:

setwd("/Users/maiellolammens/DataSci/")

### Making an R Project

Let’s make a new R Project associated with your DataSci directory. To make a new project, got to the upper right-hand side of the RStudio interface, where it says **Project: (None)**. Click the little downward arrow, select “New Project”, then select “Existing Directory” from the window that pops up. Use the graphical user interface (GUI) to navigate to the DataSci directory, then select “Create Project”.

### Getting help

* **Help** panel (lower right corner)
* help.search

help.search("bar plot")

#### Challenge

Use the help.search function to search for something in statistics that you think should be in R? Did you find anything?

I know my function - just give me the details - ?barplot

## R as calculator

We can use R just like any other calculator.

3 + 5

## [1] 8

There’s internal control for order of operations (Please Excuse My Dear Aunt Sally)

(3 \* 5) + 7

## [1] 22

3 \* 5 + 7

## [1] 22

3 \* (5 + 7)

## [1] 36

### Internal functions

There are a ton of internal functions, and a lot of add-ons.

sqrt(4)

## [1] 2

abs(-5)

## [1] 5

## Documenting your work with R Markdown

Using [R Markdown](https://rmarkdown.rstudio.com/), we can integrate descriptive text, R code, and the output from that code, into a seamless document that can be easily **reproduced**. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. While some of the details of using Markdown (and thus R Markdown) can get tricky, the basics are very easy to pickup. RStudio even comes with two very useful tools to learn and use Markdown. First, go to the Help tab and select Markdown Quick Reference. Second, a more detailed reference can be found in the Help -> Cheatsheets section.

### Markdown vs R Markdown

R Markdown is a tool that allows you to make a simple text document using the Markdown formatting syntax **with** R code and associated output embedded.

### Starting with R Markdown

1. Go to the new file button (upper left corner of the RStudio interface, or File -> New File) and choose the R Markdown option.
2. You may be prompted to install new R packages, let’s go ahead and do this if you are prompted.
3. A screen will pop-up requesting information such as Title, Author, and Output Format. Give your new doc a title and select **HTML** as the output format.
4. A new file that is pre-populated with a bunch of text will open in the Source panel.

We will go over each section briefly:

* Header
* Sections
* Code Chunks (The code chunk syntax is *very* specific and important. As a beginner, you may choose to use the GUI button to make empty chunks - it’s the little green box with a **C** in it and a plus sign.)

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. Click **Knit** now to see what happens.

## R script file

Another common file format for documenting your work is an *.R script file. Script files are primarily filled with only code and minimal* comments\* that you can include to explain that code.

**Important:** within an R file, you can use the # sign to add comments. Anything written after the # is **not** interpreted when you run the code. You can print a script file, but it will print as a simple text file without any formatting.

#### Challenge

In the Rmd file we created above, create a new R chunk and add some math operations. Now Knit the new file.

### Basic file managment in R

# What working directory am I in?  
getwd()

## [1] "/Users/maiellolammens/Dropbox/Projects/BEDE-Network-Workshop-Materials/docs"

# Move to a different directory?  
setwd(".")

#### Additional file management points

* Navigating the file path
* Tab completion of file paths
* Tab completion of R commands
* The cheat - file.choose()

#### Challenge

* Try to auto-complete fil, what do you find?
* Use the brief help menu that comes up to find a function that starts with file, and describe what you think it does.

## Variables and objects

There are several basic types of data structures in R.

* **VECTORS**: One-dimensional arrays of numbers, character strings, or logical values (T/F)
* **FACTORS**: One-dimensional arrays of factors (Stop - Let’s discuss factors)
* **DATA FRAMES**: Data tables in which the various columns may be of different type
* **MATRICES**: In R, matrices can only be 2-dimensional, anything higher dimension is called an array (see below). Matrix elements are numerical; some functions, like the transpose function t(), only work on matrices
* **ARRAYS**: higher dimensional matrices are called arrays in R
* **LISTS**: lists can contain any type of object as list elements. You can have a list of numbers, a list of matrices, a list of characters, etc., or any combination of the above.

### Functions that are useful for understanding the different types of data structures

These functions will tell you what *kind* of variable you are dealing with, as well as some additional information which may be useful as you advance in your use of R.

str()  
class()

## Practice with variables

Let’s define a variable.

my\_var <- 8

And another

my\_var2 <- 10

Work with vars

my\_var + my\_var2

## [1] 18

Make a new variable

my\_var\_tot <- my\_var + my\_var2

#### Challenge

Change the value of my\_var2

my\_var2 <- 3

What is the value of my\_var\_tot now?

### Make a vector

Let’s combine multiple values into a vector of length greater than 1. This introduces us to the one of the most prevalent functions in R, c() - i.e., the *combine* function.

# Vector of variables  
my\_vect <- c(my\_var, my\_var2)  
  
# Numeric vector  
v1 <- c(10, 2, 8, 7, 11, 15)  
  
# Char vector  
pets <- c("cat", "dog", "rabbit", "pig")

Making a vector of numbers in sequence

v2 <- 1:10  
v3 <- seq(from = 1, to = 10)

#### Challenge

1. Look up the help for the seq function, and use this to make a vector from 0 to 100, by steps of 5.
2. Come up with a way that you would use the length.out argument.

## Exploring variable elements

You can get specific elements from vectors and other data structures

* Introduction to the square brackets []

pets <- c("cat", "dog", "rabbit", "pig", "snake")  
pets[1]

## [1] "cat"

* Getting a number of elements, in sequence

pets[3:4]

## [1] "rabbit" "pig"

* Getting a number of elements, not in sequence

pets[c(1,4)]

## [1] "cat" "pig"

## Data frames

### Reading in your own data

One of the most basic things you will need to do in R is read in your own data set. You can read in Excel files, simple text files, and even files from Google Sheets (this is a bit more complicated though). But the easiest type of file to read in is a comma separated values (CSV) file. You can save an Excel workbook (or Numbers workbook or Google Sheet) as a CSV file by using the “Save as …” menu item.

Let’s read in the soil data from our Speed Data Science activity using the function read.csv. First, you’ll need to download the data set from the workshop site. Second, move the file to the data sub-folder in your DataSci folder. Then, we’ll read in the file. **NOTE: you may need to adjust the path to the file.**

soil\_data <- read.csv("data/soil\_nutrient\_data.csv")

Let’s have a brief look at these data.

head(soil\_data)

## date condition site replicate pH nitrate\_lbacre phosphorus\_lbacre  
## 1 3/29/18 wet townhouse 1 6.5 10 10.0  
## 2 3/29/18 wet townhouse 2 7.0 10 10.0  
## 3 3/29/18 wet townhouse 3 6.2 10 17.5  
## 4 3/29/18 wet townhouse 4 7.1 10 25.0  
## 5 3/29/18 wet townhouse 5 6.6 10 25.0  
## 6 3/29/18 wet osa\_top 1 7.0 10 25.0

tail(soil\_data)

## date condition site replicate pH nitrate\_lbacre phosphorus\_lbacre  
## 20 3/29/18 wet wetland\_top 5 6.2 10 10  
## 21 3/29/18 wet wetland\_bot 1 6.8 15 10  
## 22 3/29/18 wet wetland\_bot 2 6.3 10 10  
## 23 3/29/18 wet wetland\_bot 3 6.4 5 10  
## 24 3/29/18 wet wetland\_bot 4 6.4 10 10  
## 25 3/29/18 wet wetland\_bot 5 6.7 10 5

As we saw in our previous activity, these data include nutrient measurements at multiple different sites on a suburban campus. The data set also includes information about where the data samples were collected and the conditions of the site during collection.

#### summary function

Let’s begin by using the summary function to examine this data set. summary returns many of the standard statistics. When doing data exploration, a few things you want to look at are:

* How do the mean and median values within a variable compare?
* Do the min and max values suggest there are outliers?
* Which variables (i.e., columns) are quantitative (numeric) versus categorical (factors or characters)

summary(soil\_data)

## date condition site replicate  
## Length:25 Length:25 Length:25 Min. :1   
## Class :character Class :character Class :character 1st Qu.:2   
## Mode :character Mode :character Mode :character Median :3   
## Mean :3   
## 3rd Qu.:4   
## Max. :5   
## pH nitrate\_lbacre phosphorus\_lbacre  
## Min. :6.000 Min. : 5.0 Min. : 5.0   
## 1st Qu.:6.200 1st Qu.: 5.0 1st Qu.: 5.0   
## Median :6.600 Median :10.0 Median :10.0   
## Mean :6.564 Mean : 9.4 Mean :12.3   
## 3rd Qu.:6.800 3rd Qu.:10.0 3rd Qu.:17.5   
## Max. :7.200 Max. :20.0 Max. :25.0

#### A (very) brief introduction to navigating a data.frame

We will be very brief here. Check out [this Data Carpentry lesson](http://www.datacarpentry.org/R-ecology/03-data-frames.html) for more information.

* Looking at specific data.frame elements. Use *row* and *column* notation.

Here is the 5th row, 3rd column (Site). **Note: We are using square brackets to index the data.frame and we *always* use [row, column] notation.**

soil\_data[5, 3]

## [1] "townhouse"

* Looking at an entire column.

Here are two ways to get the pH column.

First, **note: we leave the row part blank, but still add the comma.**

soil\_data[ ,5]

## [1] 6.5 7.0 6.2 7.1 6.6 7.0 6.8 6.8 6.2 6.6 6.2 6.6 6.5 7.0 7.2 6.0 6.8 6.0 6.2  
## [20] 6.2 6.8 6.3 6.4 6.4 6.7

Second, **use only the variable (column) name. Note the use of the $ operator**

soil\_data$pH

## [1] 6.5 7.0 6.2 7.1 6.6 7.0 6.8 6.8 6.2 6.6 6.2 6.6 6.5 7.0 7.2 6.0 6.8 6.0 6.2  
## [20] 6.2 6.8 6.3 6.4 6.4 6.7

* Looking at specific column entry

This is another way to look at the 5th entry in the pH column.

soil\_data$pH[5]

## [1] 6.6

* Looking at all entries for a given row.

Here’s all the entries for the 5th row. **Note: here we leave the column part blank, but still add the comma.**

soil\_data[5, ]

## date condition site replicate pH nitrate\_lbacre phosphorus\_lbacre  
## 5 3/29/18 wet townhouse 5 6.6 10 25

* Looking at a set of rows and/or columns.

Here’s all the entries in the 5th through 10th rows, 5th through 7th columns. **Note: we use the : operator to look at a range of value.**

soil\_data[5:10, 5:7]

## pH nitrate\_lbacre phosphorus\_lbacre  
## 5 6.6 10 25  
## 6 7.0 10 25  
## 7 6.8 5 25  
## 8 6.8 5 25  
## 9 6.2 10 25  
## 10 6.6 5 10

* For data.frames, if you do not use row, column notation, you will get only the columns back.

head(soil\_data[5:7])

## pH nitrate\_lbacre phosphorus\_lbacre  
## 1 6.5 10 10.0  
## 2 7.0 10 10.0  
## 3 6.2 10 17.5  
## 4 7.1 10 25.0  
## 5 6.6 10 25.0  
## 6 7.0 10 25.0

#### Challenge

What am I going to get if I execute the command below?

head(soil\_data[c("site","phosphorus\_lbacre")])

## Basic data visualization

There are many data visualization tools in R, and a very rich ecosystem of add-on [packages](https://cran.r-project.org/web/packages/index.html) that make it easy to create publication ready figures. Here we will learn a few basic visualization tools.

### Visualization using the ggplot2 package

We are going to introduce a data visualization package called **ggplot2**. This package is great for producing publication quality graphics, but the syntax used to create a plot is a little more involved than base R (i.e., the graphics package).

### Aside: Installing and loading packages

First, we need to install the ggplot2 package.

**STOP** did you install the tidyverse package?

# Only need to do this once  
install.packages("ggplot2")

Then load it:

library(ggplot2)

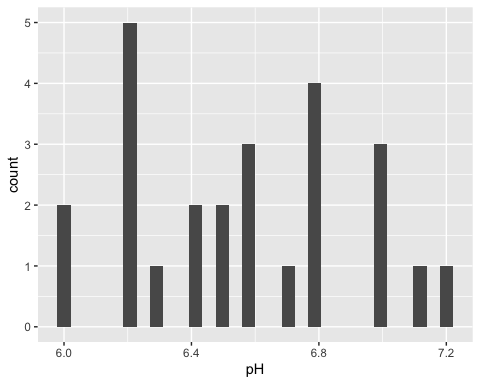
**NOTE: You will only have to install a new package once, but you will need to call the library function at the beginning of every new R session.**

### Historgrams

Let’s make a histogram of some of the nutrient data

ggplot(data = soil\_data, aes(x = pH)) +  
 geom\_histogram()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



Let’s break down this call to introduce a few key things about ggplot

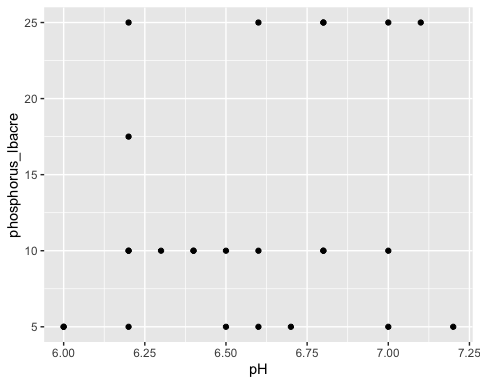
* ggplot: the initial canvas we’re working on
* geom: geometric objects (i.e. the type of plot - histogram, points, line, etc)
* aes: aesthetic mapping

#### Challenge - use ?geom\_histogram to determine how to change the number of bins used

### Scatter plots

Next let’s make a scatter plot, showing pH versus phosphorus.

ggplot(data = soil\_data, aes(x = pH, y = phosphorus\_lbacre)) +  
 geom\_point()

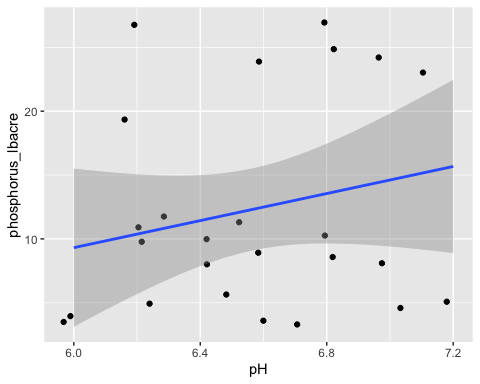


**THAT SEEMS SO COMPLICATED!**

It’s true. The syntax for ggplot can seem pretty complicated. But the power of ggplot lies in the ability to lay several geometries (geoms) over each other. Also, each geometry has a rich set of options. For example, let’s say I want to create the plot we just made, but I want to jitter the points (so they aren’t overlapping) and add a trend line.

ggplot(data = soil\_data, aes(x = pH, y = phosphorus\_lbacre)) +  
 geom\_point(position = "jitter") +  
 geom\_smooth(method = "lm")

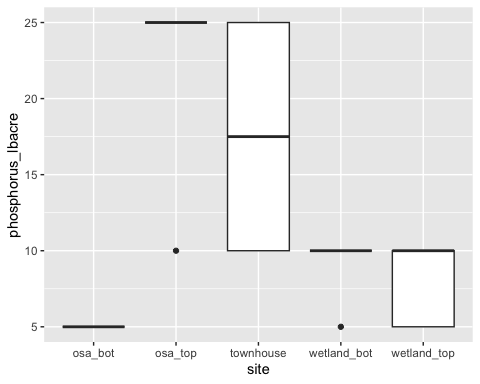
## `geom\_smooth()` using formula = 'y ~ x'



### Boxplots

Lastly, let’s make a box plot.

ggplot(data = soil\_data, aes(x = site, y = phosphorus\_lbacre)) +  
 geom\_boxplot()



#### Challenge

Using R, remake one of the plots your group worked on during the Speed Data Science activity.

## Instructor’s Notes

This lesson utilizes **live coding**. While the learners should be made aware of the existence of this document, both the instructor and the learner should work on all coding as if this document was being created from scratch. Also, the instructor and students should both be working in a new \*.R script file.

### Learning Outcomes

At the end of this session you should be able to:

* Perform basic arithmetic using R/RStudio
* Identify the major components of the RStudio interface
* Create an Rmd file and use it to document a sequence of R functions
* Read in a data set and perform basic data summary procedures in R

### Assessment

Assessment for live coding is based on the learners performance on successfully carrying out **Challenges**. This is primarily a **formative assessment**. A **summative assessment** could be incorporated as an in-class analysis lab or as part of a homework assignment.

### Learning Activities

Live-coding using a data set relevant to the course content. In this case, these data could be associated with a Plant Ecology course or an undergraduate research course.