Introduction to R, Season 1

September, 2023

Table of Contents

# 1 About this Course

## 1.1 Curriculum

The course covers fundamentals of R, a high-level programming language, and use it to wrangle data for analysis and visualization.

## 1.2 Target Audience

The course is intended for researchers who want to learn coding for the first time with a data science application, or have explored programming and want to focus on fundamentals.

# 2 Intro to Computing

## 2.1 Goals of the course

* Fundamental concepts in high-level programming languages (R, Python, Julia, WDL, etc.) that is transferable: *How do programs run, and how do we solve problems using functions and data structures?*
* Beginning of data science fundamentals: *How do you translate your scientific question to a data wrangling problem and answer it?*
* 
* Figure : Data science workflow
* Find a nice balance between the two throughout the course: we will try to reproduce a figure from a scientific publication using new data.

## 2.2 What is a computer program?

* A sequence of instructions to manipulate data for the computer to execute.
* A series of translations: English <-> Programming Code for Interpreter <-> Machine Code for Central Processing Unit (CPU)

We will focus on English <-> Programming Code for R Interpreter in this class.

More importantly: **How we organize ideas <-> Instructing a computer to do something**.

## 2.3 A programming language has following elements:

* Grammar structure (simple building blocks)
* Means of combination to analyze and create content (examples around genomics provided, and your scientific creativity is strongly encouraged!)
* Means of abstraction for modular and reusable content (data structures, functions)
* Culture (emphasis on open-source, collaborative, reproducible code)

Requires a lot of practice to be fluent!

## 2.4 What is R and why should I use it?

It is a:

* Dynamic programming interpreter
* Highly used for data science, visualization, statistics, bioinformatics
* Open-source and free; easy to create and distribute your content; quirky culture

## 2.5 R vs. Python as a first language

In terms of our goals, recall:

* Fundamental concepts in high-level programming languages
* Beginning of data science fundamentals

There are a lot of nuances and debates, but I argue that Python is a better learning environment for the former and R is better for the latter.

Ultimately, either should be okay! Perhaps more importantly, *consider what your research group and collaborator are more comfortable with*.

## 2.6 Posit Cloud Setup

Posit Cloud/RStudio is an Integrated Development Environment (IDE). Think about it as Microsoft Word to a plain text editor. It provides extra bells and whistles to using R that is easier for the user.

Today, we will pay close attention to:

* Script editor: where sequence of instructions are typed and saved as a text document as a R program. To run the program, the console will execute every single line of code in the document.
* Console (interpreter): Instead of giving a entire program in a text file, you could interact with the R Console line by line. You give it one line of instruction, and the console executes that single line. It is what R looks like without RStudio.
* Environment: Often, code will store information *in memory*, and it is shown in the environment. More on this later.

## 2.7 Using Quarto for your work

Why should we use Quarto for data science work?

* Encourages reproducible workflows
* Code, output from code, and prose combined together
* Extendability to Python, Julia, and more.

More options and guides can be found in [Introduction to Quarto](https://quarto.org/docs/get-started/hello/rstudio.html) .

## 2.8 Grammar Structure 1: Evaluation of Expressions

* **Expressions** are be built out of **operations** or **functions**.
* Operations and functions combine **data types** to return another data type.
* We can combine multiple expressions together to form more complex expressions: an expression can have other expressions nested inside it.

For instance, consider the following expressions entered to the R Console:

18 + 21

## [1] 39

max(18, 21)

## [1] 21

max(18 + 21, 65)

## [1] 65

18 + (21 + 65)

## [1] 104

nchar("ATCG")

## [1] 4

Here, our input **data types** to the operation are **numeric** in lines 1-4 and our input data type to the function is **character** in line 5.

Operations are just functions in hiding. We could have written:

sum(18, 21)

## [1] 39

sum(18, sum(21, 65))

## [1] 104

Remember the function machine from algebra class? We will use this schema to think about expressions.



Figure : Function machine from algebra class.

If an expression is made out of multiple, nested operations, what is the proper way of the R Console interpreting it? Being able to read nested operations and nested functions as a programmer is very important.

3 \* 4 + 2

## [1] 14

3 \* (4 + 2)

## [1] 18

Lastly, a note on the use of functions: a programmer should not need to know how the function is implemented in order to use it - this emphasizes [abstraction and modular thinking](#X4289ef2db5109e8791918a3c7b0cce6f2f89fd2), a foundation in any programming language.

### 2.8.1 Data types

Here are some data types that we will be using in this course:

* **Numeric**: 18, 21, 65, 1.25
* **Character**: “ATCG”, “Whatever”, “948-293-0000”
* **Logical**: TRUE, FALSE

## 2.9 Grammar Structure 2: Storing data types in the global environment

To build up a computer program, we need to store our returned data type from our expression somewhere for downstream use. We can assign a variable to it as follows:

x = 18 + 21

If you enter this in the Console, you will see that in the Environment, the variable x has a value of 39.

### 2.9.1 Execution rule for variable assignment

Evaluate the expression to the right of =.

Bind variable to the left of = to the resulting value.

The variable is stored in the environment.

<- is okay too!

The environment is where all the variables are stored, and can be used for an expression anytime once it is defined. Only one unique variable name can be defined.

The variable is stored in the working memory of your computer, Random Access Memory (RAM). This is temporary memory storage on the computer that can be accessed quickly. Typically a personal computer has 8, 16, 32 Gigabytes of RAM. When we work with large datasets, if you assign a variable to a data type larger than the available RAM, it will not work. More on this later.

Look, now x can be reused downstream:

x - 2

## [1] 37

y = x \* 2

## 2.10 Grammar Structure 3: Evaluation of Functions

A function has a **function name**, **arguments**, and **returns** a data type.

### 2.10.1 Execution rule for functions:

Evaluate the function by its arguments, and if the arguments are functions or contains operations, evaluate those functions or operations first.

The output of functions is called the **returned value**.

sqrt(nchar("hello"))

## [1] 2.236068

(nchar("hello") + 4) \* 2

## [1] 18

## 2.11 Tips on Exercises / Debugging

Common errors:

* Syntax error.
* Changing a variable without realizing you did so.
* The function or operation does not accept the input data type.
* It did something else than I expected!

Solutions:

* Where is the problem?
* What kind of problem is it?
* Explain your problem to someone!

# About the Authors

These credits are based on our [course contributors table guidelines](https://www.ottrproject.org/more_features.html#giving-credits-to-contributors).

| Credits | Names |
| --- | --- |
| **Pedagogy** |  |
| Lead Content Instructor(s) | Chris Lo |
| Lecturer | Chris Lo |
| Content Author(s) (include chapter name/link in parentheses if only for specific chapters) - make new line if more than one chapter involved | If any other authors besides lead instructor |
| Content Contributor(s) (include section name/link in parentheses) - make new line if more than one section involved | Wrote less than a chapter |
| Content Editor(s)/Reviewer(s) | Checked your content |
| Content Director(s) | Helped guide the content direction |
| Content Consultants (include chapter name/link in parentheses or word “General”) - make new line if more than one chapter involved | Gave high level advice on content |
| Acknowledgments | Gave small assistance to content but not to the level of consulting |
| **Production** |  |
| Content Publisher(s) | Helped with publishing platform |
| Content Publishing Reviewer(s) | Reviewed overall content and aesthetics on publishing platform |
| **Technical** |  |
| Course Publishing Engineer(s) | Helped with the code for the technical aspects related to the specific course generation |
| Template Publishing Engineers | [Candace Savonen](https://www.cansavvy.com/), [Carrie Wright](https://carriewright11.github.io/), [Ava Hoffman](https://www.avahoffman.com/) |
| Publishing Maintenance Engineer | [Candace Savonen](https://www.cansavvy.com/) |
| Technical Publishing Stylists | [Carrie Wright](https://carriewright11.github.io/), [Ava Hoffman](https://www.avahoffman.com/), [Candace Savonen](https://www.cansavvy.com/) |
| Package Developers ([ottrpal](https://github.com/jhudsl/ottrpal)) [Candace Savonen](https://www.cansavvy.com/), [John Muschelli](https://johnmuschelli.com/), [Carrie Wright](https://carriewright11.github.io/) |  |
| **Art and Design** |  |
| Illustrator(s) | Created graphics for the course |
| Figure Artist(s) | Created figures/plots for course |
| Videographer(s) | Filmed videos |
| Videography Editor(s) | Edited film |
| Audiographer(s) | Recorded audio |
| Audiography Editor(s) | Edited audio recordings |
| **Funding** |  |
| Funder(s) | Institution/individual who funded course including grant number |
| Funding Staff | Staff members who help with funding |

## ─ Session info ───────────────────────────────────────────────────────────────  
## setting value   
## version R version 4.0.2 (2020-06-22)  
## os Ubuntu 20.04.5 LTS   
## system x86\_64, linux-gnu   
## ui X11   
## language (EN)   
## collate en\_US.UTF-8   
## ctype en\_US.UTF-8   
## tz Etc/UTC   
## date 2023-09-29   
##   
## ─ Packages ───────────────────────────────────────────────────────────────────  
## package \* version date lib source   
## assertthat 0.2.1 2019-03-21 [1] RSPM (R 4.0.5)   
## bookdown 0.24 2023-03-28 [1] Github (rstudio/bookdown@88bc4ea)   
## cachem 1.0.7 2023-02-24 [1] CRAN (R 4.0.2)   
## callr 3.5.0 2020-10-08 [1] RSPM (R 4.0.2)   
## cli 3.6.1 2023-03-23 [1] CRAN (R 4.0.2)   
## crayon 1.3.4 2017-09-16 [1] RSPM (R 4.0.0)   
## desc 1.2.0 2018-05-01 [1] RSPM (R 4.0.3)   
## devtools 2.3.2 2020-09-18 [1] RSPM (R 4.0.3)   
## digest 0.6.25 2020-02-23 [1] RSPM (R 4.0.0)   
## ellipsis 0.3.1 2020-05-15 [1] RSPM (R 4.0.3)   
## evaluate 0.20 2023-01-17 [1] CRAN (R 4.0.2)   
## fastmap 1.1.1 2023-02-24 [1] CRAN (R 4.0.2)   
## fs 1.5.0 2020-07-31 [1] RSPM (R 4.0.3)   
## glue 1.4.2 2020-08-27 [1] RSPM (R 4.0.5)   
## htmltools 0.5.5 2023-03-23 [1] CRAN (R 4.0.2)   
## knitr 1.33 2023-03-28 [1] Github (yihui/knitr@a1052d1)   
## magrittr 2.0.3 2022-03-30 [1] CRAN (R 4.0.2)   
## memoise 2.0.1 2021-11-26 [1] CRAN (R 4.0.2)   
## pkgbuild 1.1.0 2020-07-13 [1] RSPM (R 4.0.2)   
## pkgload 1.1.0 2020-05-29 [1] RSPM (R 4.0.3)   
## prettyunits 1.1.1 2020-01-24 [1] RSPM (R 4.0.3)   
## processx 3.4.4 2020-09-03 [1] RSPM (R 4.0.2)   
## ps 1.4.0 2020-10-07 [1] RSPM (R 4.0.2)   
## R6 2.4.1 2019-11-12 [1] RSPM (R 4.0.0)   
## remotes 2.2.0 2020-07-21 [1] RSPM (R 4.0.3)   
## rlang 1.1.0 2023-03-14 [1] CRAN (R 4.0.2)   
## rmarkdown 2.10 2023-03-28 [1] Github (rstudio/rmarkdown@02d3c25)  
## rprojroot 2.0.3 2022-04-02 [1] CRAN (R 4.0.2)   
## sessioninfo 1.1.1 2018-11-05 [1] RSPM (R 4.0.3)   
## stringi 1.5.3 2020-09-09 [1] RSPM (R 4.0.3)   
## stringr 1.4.0 2019-02-10 [1] RSPM (R 4.0.3)   
## testthat 3.0.1 2023-03-28 [1] Github (R-lib/testthat@e99155a)   
## usethis 1.6.3 2020-09-17 [1] RSPM (R 4.0.2)   
## withr 2.3.0 2020-09-22 [1] RSPM (R 4.0.2)   
## xfun 0.26 2023-03-28 [1] Github (yihui/xfun@74c2a66)   
## yaml 2.2.1 2020-02-01 [1] RSPM (R 4.0.3)   
##   
## [1] /usr/local/lib/R/site-library  
## [2] /usr/local/lib/R/library

# 3 References