From Python to R

Learning the Basics of R Programming

2024-02-02

Daniel Woulfin, PhD, MLS

Computational Research Instruction Librarian

Columbia University Libraries

dw3105@columbia.edu

ORCID: [0000-0001-5502-9861](https://orcid.org/0000-0001-5502-9861)

https://github.com/dswoulf/

Objectives

General introduction to R for Python users:

- Lecture based

Basic syntax of R (variables/datasets, operations, data types, functions, conditionals and loops)

- Live coding

R use case: Exploring a dataset from import to data visualizations

- Live coding

Computational Literacy

Computational literacy is the ability to apply computational thinking to a wide range of problems.

Broadly this means to be able to address a problem using steps and concepts (algorithms) that a computer would understand. This means decomposing a problem into its component parts, recognizing and using patterns in data or code, generalizing and abstracting the problem in a way that it can be used elsewhere, and designing an algorithm to solve the problem.

Problems can be solved in different ways and may require different tools and languages. R is really good at some problems. Python is better at others. Every language brings with it it’s history, capabilities, strengths and weaknesses.

Python and R General Information

	Python	R
Owner	Python Software Foundation	R Foundation
Current version	3.12.1	4.3.2
Package/library repository	PyPI (506,951 libraries)	CRAN (20,272 packages)
License	Python Software Foundation License Agreement	GNU General Public License 3.0

Python and R histories

Python

The Python Software Foundation describes Python as “an interpreted, object-oriented, high-level programming language with dynamic semantics.”

- Created in 1989 by Guido van Rossum and released in 1991, named Benevolent Dictator for Life of Python (stepped down in 2018).
- Named after Monty Python’s Flying Circus.
- Designed as a scripting language that would be more easily programmable and readable than shell scripts.

Python and R histories

R

The R Foundation describes R as “a language and environment for statistical computing and graphics”

- First released in 1993 by Ross Ihaka and Robert Gentleman to teach introductory statistics at the University of Auckland. R 1.0 was released in 2000
- Inspired by S programming language and provides an open source route to the capabilities of that language
- Focused on statistical analysis, data manipulation, and data visualization.

Vectors

R is designed around vectors. A vector in R is a series of values of the same type.

This means that when looking at a dataframe (two dimensional data) each column consists of the same type of data (numeric, strings, lists, etc.).

Operations in R can happen on individual values but typically occur on vectors.

Loops in many cases are not the best way to are optional!

Wrapping your head around this is tough but important.

Tips when learning a new programming language

Programming is a skill that requires practice and making mistakes. The best way to learn something new is to do it and do it in form of a project. For this reason I recommend that people:

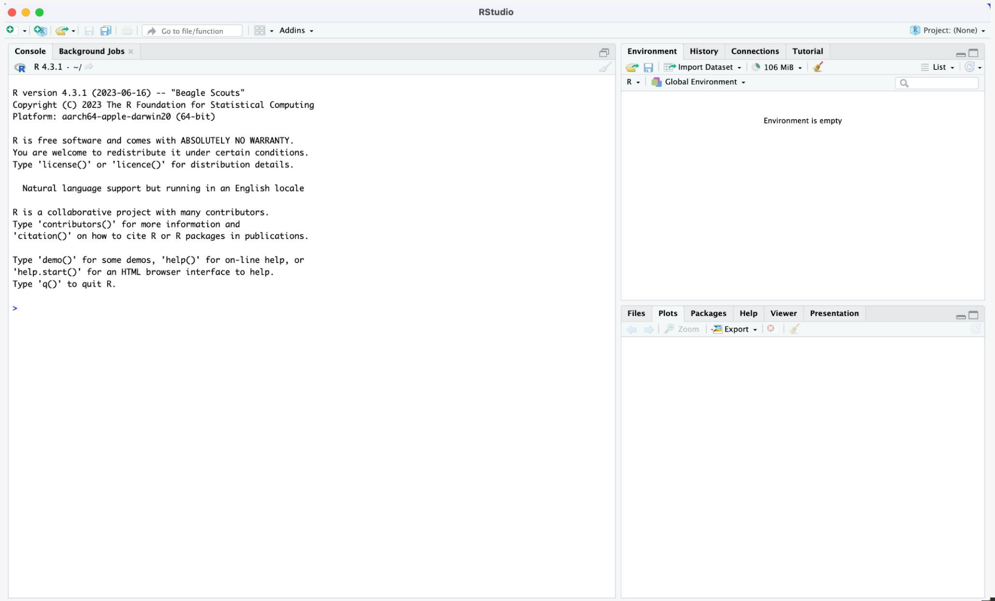
- Type out every example: This build mental muscle memory with syntax and logic.
- Bugs are normal and helpful when learning: By typing out each example you'll create bugs. Finding and fixing them is good practice.
- Figure out how you're most comfortable learning: By this I mean, do you prefer a classroom lecture scenario, to learn on your own through written materials (books and tutorials), pair programming with YouTube videos, etc.?
- Realize you'll never know everything: Relying on cheat sheets, reference materials, ChatGPT/Github Copilot, stackoverflow, etc. is okay as long as you realize their limitations and don't use them as a crutch.

R and IDEs

R natively works in the terminal but typically you would also install RStudio to work as the IDE. R studio is made up of 4 panes (a console, an environment pane, a navigation/help pane, and a scripting pane)

You can also use VSCode, Jupyter, and use R in Google Colab.

I'll primarily be working in RStudio from now on.



```
R version 4.3.1 (2023-06-16) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (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.

>
```

Syntax differences - general differences

	Python	R
Indentation	Has a specific meaning in code	No impact on code. It’s for cosmetic/readability purposes
Indexing	Starts at 0	Starts at 1
Object	Everything is an object, Object oriented language	Primary variable type is a vector and more focus on functional programming

Calling a function	function(argument)
	data.function()
		function(argument)

Syntax differences - Packages, libraries and functions

	Python	R
Strings	pip install name (in terminal/command line)	install.packages(“name”)
Importing additional functionality (packages in R, libraries or modules in Python)	import pandas from sklearn import metrics	library(“name”) “name”::function_name()
Create a function	def function_name(arg1, arg2):	Function_name <- function(arg1, arg2) {…}
lambda	lambda x: x*x	Does not have

Syntax differences - Variables

	Python	R
Variable assignments	= x = 3	->, <-, or = x <- 3, 3 -> x, x = 3 (the first option is the most common)
Data Types	int, float, string, bool	numeric, integer, character, logical, factors
Lists	lists, tuples	vectors (same data type), lists (mixed data types)
Dictionary	key: value pairs	Named lists

Syntax differences - Operators and comments

	Python	R
Modulus Exponentiation Floor division	a % b a ** b a // b	a %% b a ^ b a %/% b
Comparison operators	Mostly the same ==, !=, >, <, >=, <=	Mostly the same ==, !=, >, <, >=, <=, %in%
Logical operators	and, or, not	& (and), | (or), ! (not)
Comments	#	#

Syntax differences - Flow control, loops, and conditionals

	Python	R
Boolean	True, False	TRUE, FALSE, NA To check for na use is.na()
Code blocks	By indentation after a colon	Within curly brackets {}

Loops	for i in range(10):
	while i < 10:
		for(i in 1:10) {…}
		while(i <= 10) {…}

Conditionals	if x > 3:
	elif y < 5:
	else:
		if(x > 3) {..}
		else if(y < 5) {…}
		else {…}

Getting help

Getting help with a function in R is rather simple because CRAN sets a high bar to submit and be archived by the repository. Every function in a package has an explanation, descriptions of arguments, a description of the return value, and examples by the authors (who may be statisticians). So for example, if I forgot an argument of the mean() function all I have to type is:

?mean

In RStudio, the documentation will come up on the right. In addition, packages may have vignettes that go into further detail on how they should be used.

To see all available vignettes type: vignette(), to look at a specific vignette type vignette("vignette_name")

Vector based operations

Let’s assume we have a list of numbers that we wanted to do mathematical operations on :

- x <- c(1, 6, 8, -2, 42, NA_integer_, 100, 9999, - 700)

operation	function
average	mean()
median	median()
minimum	min()
maximum	max()

Pause for questions

Working with datasets

What’s the dataset?

The dataset is from https://github.com/planetsig/ufo-reports.

Important notes:

- Unlicensed: We’re in ethically gray area using this dataset but we’re not publishing so…
- Created in 2014
- Created by scraping the NUFORC website without the assistance of NUFORC
- Pre-cleaned (process outlined in the README on the github pages
- 81,185 records with numeric, categorical, geographic, date-time, and natural language variables

Questions: How trustworthy do you think this data is? What biases are implicit in the dataset? On a scale of 1-10 how certain are you of the accuracy of this data?

Let’s begin…

From now on I’ll be working in RStudio and coming back to this presentation

I’ll be live coding but you can follow along in the Colab however I again recommend typing as much as you can.

We’ll create three steps:

- Create potential research questions
- Examine and wrangle data
- Ask the questions of the data through tables and visualizations

tidyverse

The tidyverse is a group of interrelated packages based around the philosophy that a dataframe should be tidy. This means that every row is an observation and every column is a variable of the same type. It makes R easier to read and chain together commands. These packages are:

- ggplot2: Data visualization (documentation, cheat sheet)
- dplyr: data manipulation and transformation (documentation, cheat sheet)
- tidyr: reshaping data and handling missing values (documentation, cheat sheet)
- readr: reading and parsing different rectangular data files (documentation, cheat sheet)
- tibble: an enhanced dataframe (documentation)
- stringr: string manipulation (documentation, cheat sheet)
- forcats: functions to use with categorical data (documentation, cheat sheet)
- purrr: functions that make it easier to work with vectors (documentation, cheat sheet)
- lubridate: designed to work with date-time data (documentation, cheat sheet)
- magrittr: introduces the pipe %>% operator (documentation)

Potential Research Questions

- How long do UFO sightings last? Has it changed over time?
- What countries are listed as seeing the most UFOs by count and by duration?
- What US states see the most UFOs by count and by duration?
- What is the most seen UFO shape by count and duration?
- Is there a time of day, month or season where UFOs are seen the most? What about day of the week?