Lecture_1_Introduction_to_R

2025-02-21

What is R Programming?

R is a functional programming language appropriate for data wrangling, cleaning, and visualization, statistical analysis, modeling, and other computational tasks. It was developed at the University of Auckland in 1993 by Ross Ihaka and Robert Gentleman to teach introductory statistics and is widely used in bioinformatics, epidemiology, finance, and social sciences due to its extensive statistical modeling capabilities and rich ecosystem of packages. R is also an excellent choice of language for publication ready figure production.

R supports built in base functions such as `print()` as well as higher level functions such as `lapply()` and `Reduce()`. R also supports functions provided by specific packages and allows for users to define their own functions. Whether a function is built into R, called from a specialized package, or generated by the user, functions can be assigned to a variable, passed to arguments, or returned from other functions. You can write R code in many text editors, but throughout this class we will be using R Studio to write and execute our code.

Coding in R Studio

A Brief Anatomy of R Studio

Prior to class you should have received an email with instructions for installing R and R Studio. Navigate to your R Studio install on your computer if you have not already done so. When you open R studio, you will see several windows: code editor, console, workspace and history, and plots and files.

Code Editor: The window in the upper left is referring to as the code editor window where you will write and execute your code.

Console: The console in the bottom left window shows the pieces of code that have been run. Importantly, error messages and warning will appear in this window after you have executed code.

Workspace and History: The workspace and history window in the upper right will display the names of objects as well as sizes of objects loaded into the environment.

Plots and Files: The bottom right window will display plots generated by your code. This window will also display any documentation related to functions if you run `?name_of_your_function()`. For example, try typing `print()` in your console and hitting enter. A description of the print function should appear in the plots and files window.

For comfort of use, R Studio also has an "Appearance" option under Tools > Global Options > Appearance. This option allows the user to specify a background theme and color. You might find it easier on your eyes to have a dark background.

File Generation in R Studio

Using R Studio we can generate several types of documents. Here we will discuss the file types with which you will be interacting over the course of our class: R Script: An R Script is a plain text file that contains lines of R codem usually written in sequetial order, It is the most basic file type for writing and running R code. R Scripts are great for writing and executing scripts, developing functions, running analyses, and automating tasks.

R Markdown: An R Markdown file combines R code with formatted text using Markdown syntax. It allows users to create dynamic documents, reports, presentation, and even websites that include both text and executable code. These sorts of files are great for creating reproducible research documents, technical reports, and presentations that combine code, analysis, and visualizations. In fact, this very file is an R Markdown file and we will be using them as a learning tool throughout the course of this class.

R Notebook: An R Notebook is a type of R Markdown file that provides an interactive interface for executing code chunks and immediately viewing results inline. R Notebooks are great for interactive data exploration and analysis with a mix of code and narrative, where results are displayed immediately without needing to knit the document.

R for Basic Calculations

To get our feet wet with R, let's execute some simple arithmetic operations. R leverages the following operators to perform simple calculations

```
library(knitr)

data <- data.frame(
  Function = c("Addition", "Subtraction", "Multiplication", "Division","Exponentiation"),
  Operator = c("+", "-", "*","/","^")
)

knitr::kable(data)
```

Function	Operator
Addition	+
Subtraction	-
Multiplication	*
Division	/
Exponentiation	^

Operators in R

Logicals in R

Data Structures in R

Vectors

Lists

Matrices

Data Frames

Loops in R

If/Else Statements in R

Saving Objects and Other Files in R