R for Research

Olamide M. Adu

2024-09-11

Table of contents

Preface

In today's research landscape, the ability to analyze data effectively is crucial. Whether in academia, industry, or governmental research, data-driven decisions shape outcomes. R, a powerful programming language for statistical analysis and data visualization, has emerged as a cornerstone in this field.

This book, R for Research, is designed for researchers, data scientists, and students looking to leverage R for a variety of analytical tasks. From data manipulation to visualization, and advanced modeling, this guide provides the foundational knowledge to transform raw data into actionable insights.

The journey of mastering R may feel daunting at first, but with the right approach, it quickly becomes intuitive and rewarding. Through hands-on examples, practical exercises, and detailed explanations of core concepts, this book will take you from a beginner to an adept user, ready to tackle complex research problems with confidence.

Whether you're aiming to publish research, gain insights from large datasets, or simply deepen your understanding of data, this book equips you with the tools and skills to succeed in your endeavors. Welcome to the world of R—a language that bridges the gap between theory and practice, making data analysis more accessible, efficient, and impactful.

Introduction

R has become a cornerstone in the data science community and academia, largely due to its flexibility, open-source nature, and extensive statistical capabilities. Originally developed in the early 1990s by statisticians Ross Ihaka and Robert Gentleman, R was designed as a language specifically for data analysis and statistical computing. Over time, it evolved into one of the most popular tools for researchers, data scientists, and statisticians alike.

R and Academia

In academia, R holds a uniSque position due to its roots in statistical analysis. Many academic researchers favor R for its comprehensive set of statistical tools, making it ideal for disciplines like economics, psychology, sociology, biostatistics, pharmacolgy, health sciences, biology, genomics, nature and environmental sciences, where complex data analysis is routine. The fact that R is free and open-source has contributed to its popularity in academic settings, where budgets can be tight, and access to proprietary software may be limited.

R is particularly favored in research involving statistical modeling, simulations, and advanced analytics such as machine learning, which is essential for the rapid analysis of large datasets. Universities often integrate R into their curricula in courses related to statistics, bioinformatics, silviculture, biometrics, and even computational social sciences, making it a key part of training the next generation of data scientists.

In academic research, reproducibility is a crucial aspect, and R excels in this area. With some of its packages, users can create dynamic reports that integrate code, data, and narrative in a single document. This allows researchers to ensure that their analyses are transparent and reproducible by others, a key requirement for peer-reviewed publications. Reports, research papers, presentations, and even books can be generated directly from RStudio (R's associated and most popular integrated development environment - IDE), providing a streamlined workflow for sharing results.

The reproducibility features of R make it particularly appealing in academic settings where transparency and rigor in research are paramount. R packages like `quarto` and `RMarkdown` has become a standard for producing research papers, technical reports, and even full theses. It allows researchers to mix prose, statistical results, and plots within a single document, all while maintaining a fully reproducible workflow.

R and Data Science

R use is not limited to academia, it's use is becoming one of the major tool used in a lot of sectors. These sectors usually need some type of data scientist, researcher, data analyst or data visualization specialist. With R and the numerous packages it supports this roles are fulfilled and be automated. This makes R and it's associated IDE RStudio a tool for all. In the data science community, R is known for its versatility in handling data wrangling, visualization, and modeling tasks. It is an excellent tool for managing complex data workflows from data cleaning to reporting

The R Community and Ecosystem

One of R's most significant strengths is its vibrant and growing community. The Comprehensive R Archive Network (CRAN), a central repository for R packages, contains over 21000 packages as at the time of this book was written, with contributions from thousands of developers worldwide. These packages extend R's functionality to virtually every domain of data science and research, from time-series analysis to geospatial data, genomics, and beyond. This ecosystem makes R adaptable to a broad range of applications, enabling users to apply cutting-edge techniques to their data.

The R community is also known for its strong support culture. Forums like Stack Overflow, R-bloggers, and Posit Community provide a wealth of knowledge and resources, ensuring that newcomers and experienced users alike can find help when they encounter challenges. This collaborative environment is a significant factor in R's popularity, as users can quickly find solutions and learn from one another.

Why Learn R

- Data Analysis and Statistics: R was developed specifically for statistical computing, making it ideal for data analysis, modeling, and visualization.
- Rich Ecosystem: With thousands of packages available via CRAN, , R offers tools for everything from machine learning to bioinformatics.
- Reproducible Research: Packages like `RMarkdown` help in creating reproducible reports, crucial for academia and industry.
- Visualization: R excels in producing publication ready visuals.
- Growing Popularity in Data Science: Many industries, including finance, healthcare, and tech, use R for data-driven decision-making.

There's a new IDE that supports Python and R that is still under development by the company Posit PBC. This is called Positron. Positron is still in its early stage of development.

Why Not R?

Despite R's strengths, there are scenarios where it may not be the best fit:

- Performance Limitations: R is often slower than other languages like Python or Julia, especially when dealing with very large datasets or high-performance tasks such as real-time processing. To increase the performance of R, using packages like `data.table`, `parallel`, `duckdb`, `arrow` and `future` can be helpful.
- Learning Curve: While R's syntax is intuitive for statistical tasks, its learning curve can be steep for those unfamiliar with its unique paradigms. Tasks beyond basic data analysis may require more in-depth coding skills.
- Less Versatility: R is heavily focused on data analysis and statistics, making it less suited for general-purpose programming. Its capabilities outside of data science, machine learning, and statistical computing are limited compared to more versatile languages.
- Package Dependency: Although R has a vast library of packages, their quality can vary. Some packages might be poorly maintained or have compatibility issues with newer R versions.
- Integration: R is not always the first choice for web development, app development, or integration with production systems. While solutions exist (such as Shiny for web apps), these use cases are generally more efficient in other languages like Python or JavaScript.

Alternatives to R

- Python: Python is a popular alternative, known for its versatility beyond data science. It has extensive libraries for machine learning (e.g., scikit-learn, TensorFlow), data manipulation (pandas), and visualization (matplotlib, seaborn). Python's general-purpose nature makes it suitable for both data analysis and broader applications like web development, app development and automation.
- Julia: Julia is emerging as a high-performance language designed for numerical and scientific computing. It offers speed advantages over R and Python, particularly in tasks that involve heavy computation.

- SAS: Commonly used in industries such as healthcare, SAS is a robust tool for statistical analysis with a long history in academia and corporate sectors. It provides a stable environment but comes with high licensing costs compared to R's open-source nature.
- SPSS: SPSS is another statistical tool widely used in academic research and business analytics. Like SAS, it's user-friendly for statistical analysis but is expensive and less flexible than R or Python.
- SQL: For tasks that involve managing and querying large databases, SQL is often the better tool. It is not a replacement for R's statistical capabilities but excels at managing, querying, and manipulating relational databases.
- MATLAB: Popular in academia and engineering, MATLAB is strong in matrix computations and simulations. It has high performance but comes with expensive licensing and is more niche compared to R and Python.
- Stata: Commonly used in economics, sociology, and epidemiology, Stata is known for its ease of use and built-in support for statistical and econometric analysis. It has an intuitive interface, making it accessible to non-programmers. However, it lacks the flexibility and vast package ecosystem of R.
- Tableau/Power BI: These tools are highly effective for creating data visualizations and dashboards. While not as powerful as R in terms of statistical computing, they excel in visualizing and presenting data, especially in a business context.
- Microsoft Excel: While very versatile can not typically be considered a direct alternative to R but is sometimes used as a simpler tool for data analysis, especially for smaller datasets. However, compared to R, Excel has limitations in handling larger datasets, performing complex statistical modeling, and automation. While Excel is great for basic data entry, simple calculations, and visualization, R is more suited for advanced statistical analysis, reproducibility, programming, and working with large-scale data. Therefore, Excel may complement R but can't replace its capabilities.

Each of these alternatives has its strengths, and the choice depends on the project requirements, performance needs, and user expertise.

Goal of the book

The primary goal of R for Research is to provide a clear and practical guide to using R for data analysis in research settings. This book aims to bridge the gap between theoretical knowledge and practical application, equipping readers with the skills needed to handle complex datasets, perform statistical analyses, and create insightful visualizations. Whether you are a beginner or have prior programming experience, this book will help you leverage the full power of R to streamline your research process and make data-driven decisions with confidence.

By the end of this book, readers will have:

- A solid understanding of R's core functionalities for data analysis.
- Proficiency in using key R packages for data manipulation, visualization, and statistical modeling.
- Practical experience through real-world examples and exercises that are relevant to various research fields.
- The ability to integrate R into your research workflow, allowing for reproducible and transparent analyses.

Ultimately, R for Research is designed to enhance your analytical skills, helping you transform raw data into meaningful insights that drive impactful research.

Scope, Limitation, and Expectations

Scope

R for Research covers essential techniques for data analysis using R, focusing on the needs of researchers across various disciplines. It introduces readers to R programming, data manipulation, statistical analysis, and visualization, with practical examples and case studies from academic research. The book is structured to help readers at any experience level, from beginner to intermediate, make effective use of R in their research.

Limitation

While the book provides a comprehensive introduction to R, it does not cover advanced topics such as machine learning, deep statistical modeling, or specialized fields like genomics or financial modeling in detail. The focus is primarily on general research applications, so readers looking for highly specialized content may need to supplement their learning with more advanced resources.

Expectations

By the end of this book, readers are expected to:

- Understand R's core functions for data manipulation and visualization.
- Apply R to basic statistical analyses and data preparation.
- Integrate R into their research workflow for reproducible and transparent analysis.
- Have confidence in handling real-world data sets for various research contexts, but be aware that mastering complex, domain-specific analyses may require further study.

This book sets a foundation, equipping readers with skills and knowledge to build upon in more specialized areas of data science and research.

1 R, RStudio, and Posit Cloud

ℹ Note

You can skip all the steps below if you already have R and RStudio installed. Skip to the operating you use.

1.1 Installing R and RStudio

R and RStudio are both free to install and use. To install both software, you need to download and install the right file for your operating system (OS).

1.1.1 Installing R

Windows OS (10/ 11): To download R from CRAN visit this link https://cran.rstudio.com/bin/windows/base/R-4.4.1-win.exe and run the installer. This is the latest version at the time of writing this chapter

Mac OS: There are two versions of R for Mac users based on the processor in your computer.

- For M1-M3 processors visit the link https://cran.r-project.org/bin/macosx/big-sur-arm64/base/R-4.4.1-arm64.pkg and run the installer
- For Intel processors visit the link https://cran.r-project.org/bin/macosx/big-sur-x86_64/base/R-4.4.1-x86_64.pkg and run the installer.

Linux Users R installation on Linux depends on your distribution. Open terminal (`Ctrl` + `Alt` + `T`) and enter the command below based on your distribution.

- Debian/Ubuntu distribution:

```
#| eval: false
$ sudo apt update
$ sudo apt install r-base r-base-dev
```

- Redhat/Fedora distribution:

```
#| eval: false
$ sudo dnf install R
```

1.1.2 Installing RStudio

To install RStudio, visit the link https://posit.co/download/rstudio-desktop/ scroll down and download the package for your OS, and install. Installation of downloaded file is direct. If you prefer a video guide, choose the link based on your OS:

- window
- Mac
- linux-ubuntu

1.1.3 Posit Cloud

There's an option to use R and RStudio online from your browser. This is through Posit Cloud (formerly RStudio Cloud). Posit cloud is owned by the same organization that made RStudio. To use posit cloud, visit https://posit.cloud/ and register. After registering, you get a page similar to Figure 1.1. After registering, click on new workspace on the left sidebar and create a new project.

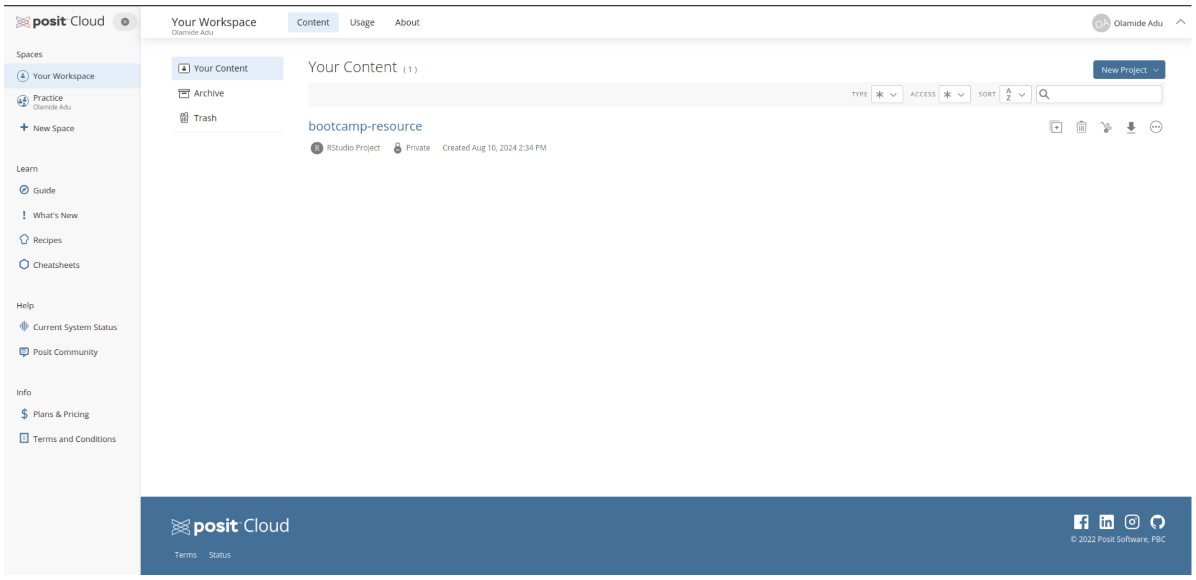
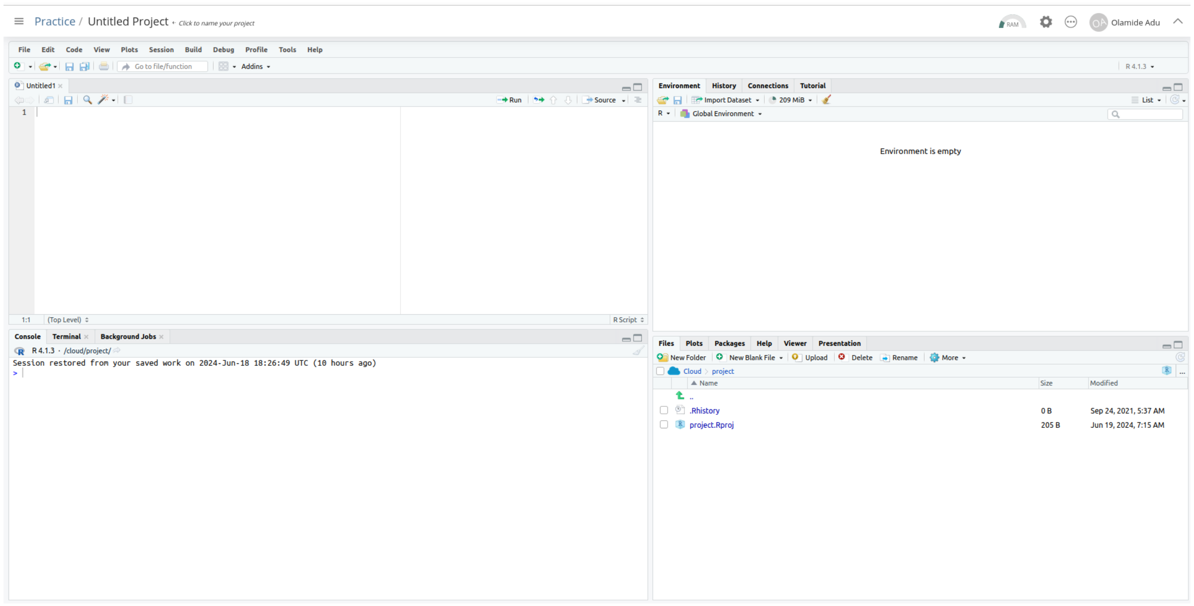


Figure 1.1: posit cloud

The video series in this link is a good resource for starting with Posit Cloud.

1.2 The RStudio Interface

On opening RStudio, you have three panes by default. You can get a fourth pane like what is shown in Figure 1.2. To do this, click the + button at the top-left and select R Script, or use the keyboard shortcut `Ctrl + Shift + N` on Windows/Linux and `Cmd + Shift + N` on Mac. More details on the panes is provided in Section 1.2.1


Practice / Untitled Project
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1
Run
Source
Environment History Connections Tutorial
Import Dataset
Global Environment
Environment is empty
Console Terminal Background Jobs
R 4.1.3 · /cloud/project/
Files Plots Packages Help Viewer Presentation
New Folder New Blank File Upload Delete Rename More
Cloud > project
Name Size Modified
.Rhistory 0 B Sep 24, 2021, 5:37 AM
project.Rproj 205 B Jun 19, 2024, 7:15 AM
Olamide Adu

```
Session restored from your saved work on 2024-Jun-18 18:26:49 UTC (10 hours ago)
>
```

Figure 1.2: R Studio Environment

You can change the arrangement of the panes by clicking `Tools` on the menu bar then navigate to `Global Options` and `Pane Layout` in the popup box.

1.2.1 RStudio Panes/Windows

The four windows/panes as shown in Figure 1.2 are:

1. **Source Pane**: Located in the top-left. This pane shows after adding a R script. The R script is where you will do most of data analysis task. It is technically a plain text file containing a series of R commands that are executed in sequence. The source pane also shows your dataset in spreadsheet-like format.

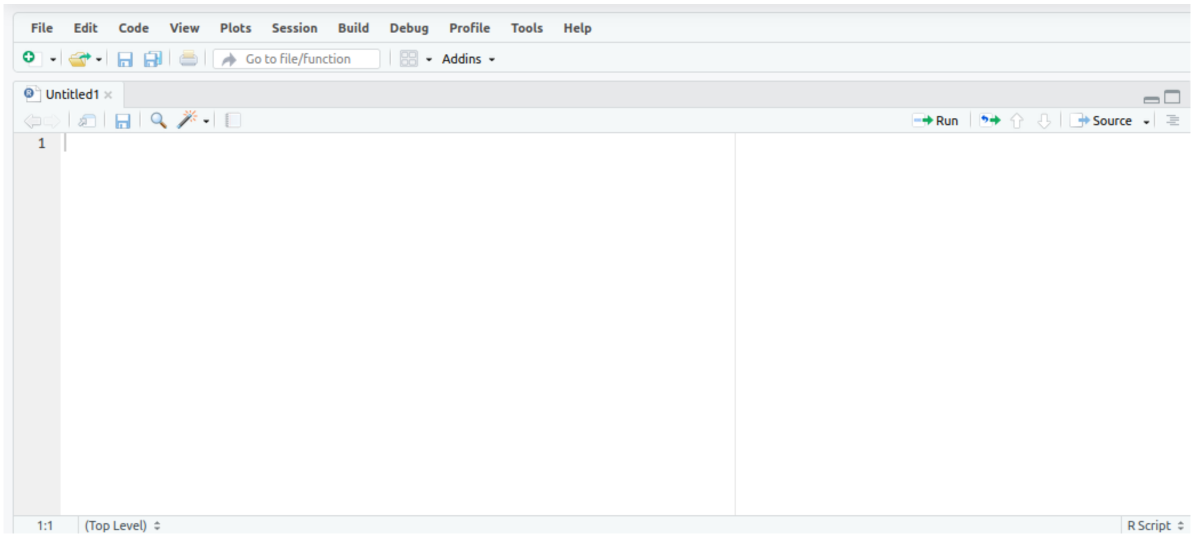
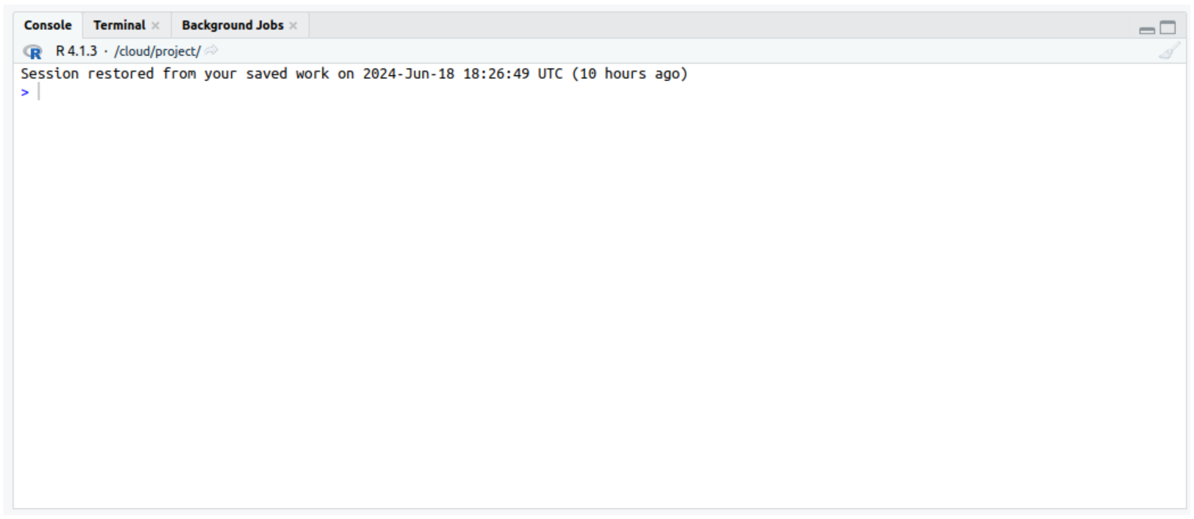


Figure 1.3: Source window

2. **Console/Terminal/Background Jobs pane**: Located in the bottom-left. It include the **Console** pane where outputs and results of your analysis are displayed. The console is also an interactive section where you can write code and carry out some analysis. All the codes written in the script are usually displayed and executed on the console. This pane also include the **Terminal** pane which provides direct access to your computer using command line. Lastly, we have the **Background Jobs** pane which allows you to run R scripts or tasks in the background while you continue working on other tasks in the main R session. This pane is especially useful for long-running operations that might otherwise block your R session, such as data processing, simulations, or model fitting.



```
Session restored from your saved work on 2024-Jun-18 18:26:49 UTC (10 hours ago)
>
```

Figure 1.4: Console window

3. **Environment / History Pane**: Located in the top-right. It includes the **Environment** pane which displays all objects created during a session, such as data and variables. We also have the **History** pane which stores the history of commands executed. This pane allows you to select previous commands and rerun them or add them back to console if you want. Located closely to **Histor** pane is the **Connection** pane which helps you interact with and manage connections to external data sources, such as databases, cloud services, or remote data systems. It provides a user-friendly interface for establishing and managing these connections directly from RStudio, without needing to write extensive code. After this is the **Tutorial** pane provides additional learning materials for R and RStudio. It is designed to be an interactive R tutorial.

An R session refers to the instance of R that is currently running, which includes everything that is active in memory and the state of your environment in RStudio or any R interface.

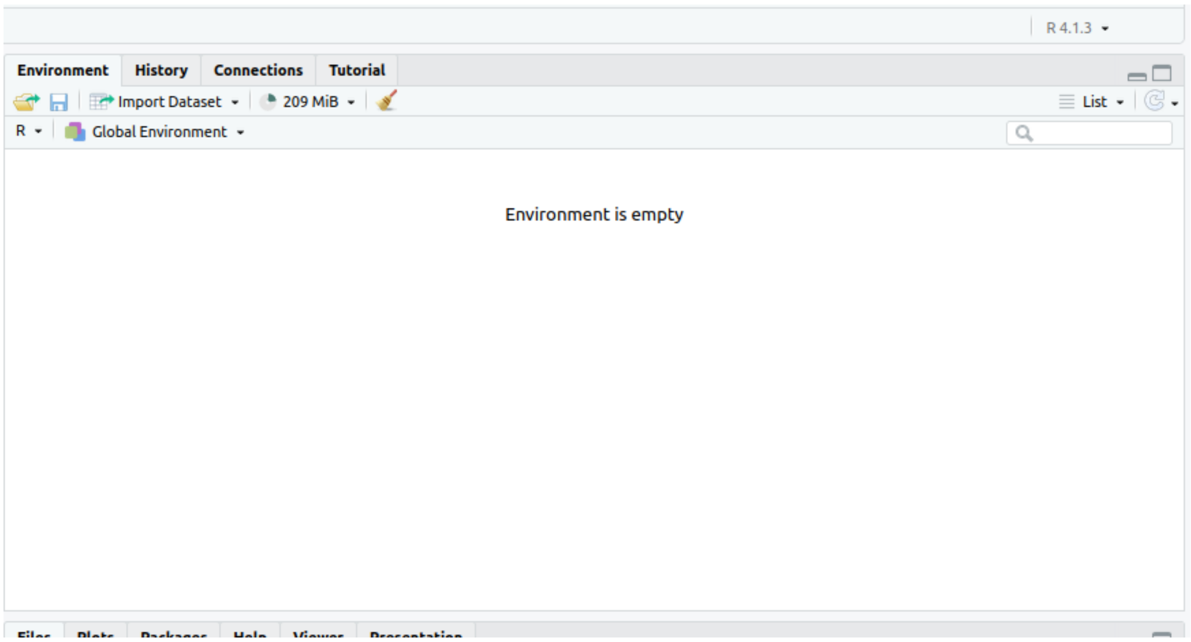
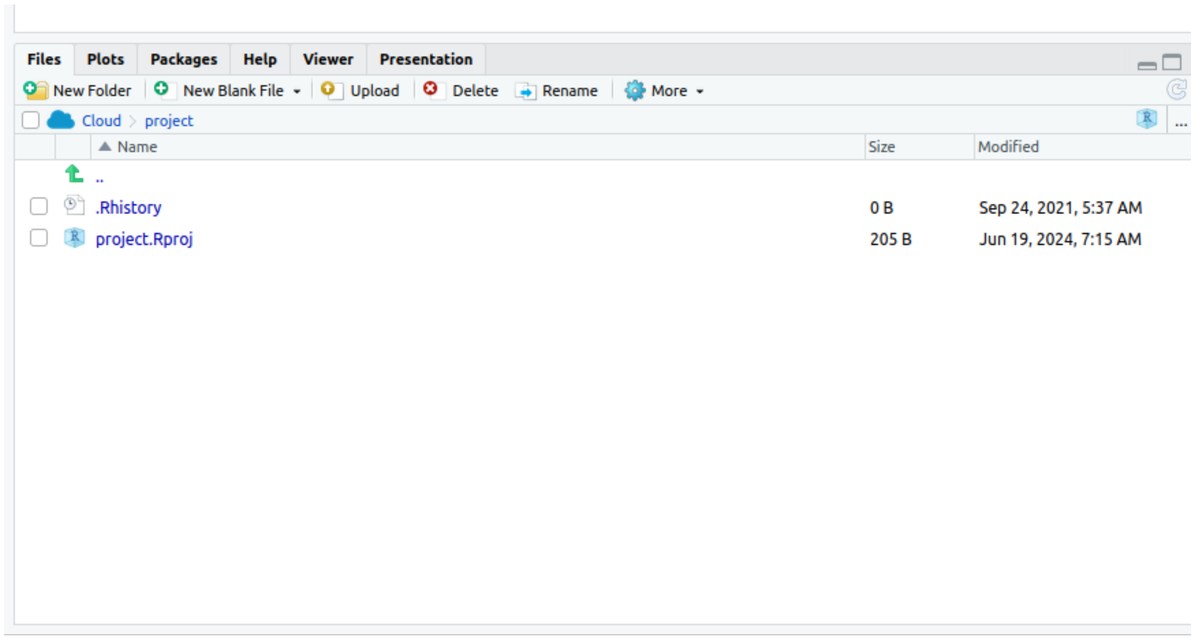


Figure 1.5: Environment window

4 **Files / Plots / Packages / Help / Viewer Window**: Located in the bottom-right. It consists of the **Files** pane which shows files and folders in your working directory. This pane also include tools for renaming, deleting, creating, copying and moving files and folders in your computer. Next is the **Plots** pane which displays visualizations. You can also export, zoom, and copy to clipboard the visuals created in this pane. The **Packages** pane follows closely after and it helps to manage packages you have. It can also be used for updating old packages as well as installing new packages. Next is the **Help** pane which shows the documentation of packages different functions. This documentation include examples which can help understand the usage of the functions. Lastly, we have the **Viewer** pane is used to display a variety of visual outputs including interactive web content, plots, maps, and web-based applications.



Name	Size	Modified
..		
.Rhistory	0 B	Sep 24, 2021, 5:37 AM
project.Rproj	205 B	Jun 19, 2024, 7:15 AM

Figure 1.6: Files window

Script vs Console

It is advised to use a script to ensure reproducibility. It ensures you can edit your work when you make errors, share with others your analysis, and also reproduce all steps in your analysis instantly.

2 R Basics

Understanding the basic syntax of R is essential for performing complex operations. Basic operations in R include arithmetic (e.g., addition, subtraction), comparison operations which involve using conditionals to check how object compares against one another and iterations or looping to solve repetitive tasks.

2.1 Arithmetic Operation

Operations such as addition, subtraction, multiplication and so on, are easy to execute in R. The arithmetic operators are given below:

Table 2.1: Arithmetic operators in R

Operator	Definition
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponent or Power
**	Exponent or Power
%%	Remainder division
%/%	Integer division

Type the following commands in your console and hit enter:

```r
2 + 1
```

```
[1] 3
```

```r
3 - 1
```

```
[1] 2
```

```r
4 * 2
```

```
[1] 8
```

```r
15 / 5
```

```
[1] 3
```

```r
3 ^ 2
```

```
[1] 9
```

```r
3 ** 2
```

```
[1] 9
```

The above operations are straightforward. `%%` operator returns the remainder when two numbers are divided. While `%/%` returns the whole number or integer and discard the remainder.

```r
5 %% 3
```

```
[1] 2
```

```r
5 %/% 2
```

```
[1] 2
```

R performs arithmetic easily and also follows the order of operations.

```r
3 + (5 - 2)
```

```
[1] 6
```

```r
(5 * 4) / (7 - (2 + 4))
```

```
[1] 20
```

Tip

Notice the spaces between numbers in arithmetic operations. While not required, spacing makes the code more readable. Don't be surprised that a lot of times you and people you work with / for will mostly be reading your codes.

2.1.1 Comparison Operation

Comparison operators are used to compare values and returns either TRUE or FALSE. TRUE and FALSE are called logical value. You will get more information on this when we get to data types.

Comparison operator in R include

Table 2.2: Comparison Operators in R

Symbol	Meaning
>	Greater than
<	Less than
==	Equal to
>=	Greater than or equal to
<=	Less than or equal to
!=	Not equal to

Try these commands in R's console:

```r
4 > 5
```

```
[1] FALSE
```

```r
(5^2) < (5 * 2)
```

```
[1] FALSE
```

```r
10 >= 7
```

```
[1] TRUE
```

```r
10 == 12
```

```
[1] FALSE
```

There are other operators which are useful in R. They are used to assign names to a variable or access elements of objects.

Table 2.3: Other Operators in R

Symbol	Meaning
<-	Assignment
=	Assignment
->	Assignment (not common)
$	Access elements

For example, we can assign the number `17` to the variable `c`. `c` here is a container holding values.

```r
c <- 17
c
```

```
[1] 17
```

Above we assigned the value `17` to `c` using `<-`. This is the convention used for assigning objects to variables in R. We can change the value in `c` by assigning another value to it. This time let's assign the value 100 using the `=` operator

```r
c = 100
c
```

```
[1] 100
```

When we assign a different value to the same variable name in R, it gets replaced by the new value, thus a variable can only hold one value at a time. For `->` assignment variable, it comes after the value. For example:

```r
5000 -> d
d
```

```
[1] 5000
```

While `->` is an assignment operator in R, it is rarely used. `<-` is the main operator used for assignment while `=` is common within functions. Following this simple guide ensure an organized code that is easy to follow.

The `$` will be discussed in future chapters and it will be used in this book.

2.2 Summary

In this chapter, you some basic R syntax and the operators. The operators are used for mathematical operations, such as addition, multiplication, and division to mention a few. We also saw other operators such as `<` and `>` for comparing values. Lastly, we saw the assignment operator used to store values into containers called variable. Now that we've got the basics pinned down, we will move on to learning the data types in R.

3 Data Types in R

This section is still under development

3.1 Character

3.2 Factor

3.3 Numerical Data

3.4 Logical

3.5 Objects

3.6 Changing Data Types

4 Functions

This section is still under development

4.1 Some In-built R Functions

4.2 Creating Functions

4.3 Using the custom function

4.4 Getting Help

5 Data Structure

This section is still under development

5.1 Atomic Vectors

5.2 Matrices

5.3 List

5.4 Data frames/tibbles

6 Packages

This section is still under development

6.1 Installing Packages in R

6.2 Making Use of Downloaded Package

6.3 Using the Library Function

6.4 Declaring Packages in Use

7 Workflow

This section is still under development

7.1 The R Script

7.2 Creating a R Script

7.3 The R Project

7.4 Creating an R Project

7.5 Benefits of Using R Projects

8 Introduction to Git and Github

This section is still under development

8.1 What is Git and GitHub

8.2 Why use Git?

8.3 Git Key Concepts

8.4 Git Workflow

8.5 Basic Git Commands

8.6 Working with GitHub

8.7 Git and GitHub in RStudio

9 Getting External Data - importing your data into RStudio

This section is still under development

9.1 Dependent Packages

9.2 CSV

9.3 Base R Approach

9.4 Tidyverse Approach

9.5 Writing CSV File

9.6 Excel Files

9.7 Writing Excel Files

9.8 Google Sheets

10 Data Manipulation

This section is still under development

10.1 Reorder your data with arrange()

10.2 Pick variables with select()

10.3 Subset data with filter()

10.4 Add or change the content of a variable with mutate()

10.5 Collapse columns using summarize()

10.6 Reshaping your data

11 Exploratory Data Analysis

This section is still under development

11.1 Understanding your data

11.1.1 Data preview

11.1.2 Data structure

11.1.3 Data summary / Descriptive Statistics

11.2 Missing Values

11.2.1 Detecting missing values

11.2.2 Removing missing values

11.3 Variable analysis

11.3.1 Univariate analysis

11.3.2 Character / Factor variables

11.3.3 Numerical variables

11.4 Bivariate Analysis

###Two categorical variable analysis ### Two numerical variable analysis ### One numerical and one categorical variable analysis

11.5 Multivariate analysis

12 Inferential Statistics

This section is still under development

12.1 Hypothesis testing with t-test

12.1.1 One-sample t-test

12.1.2 Two Sample t-test

###Limitations and Assumptions of t-test ## ANOVA and Linear regression ### ANOVA ### Linear regression ### Making predictions with linear regression models

13 Introduction to Quarto

This section is still under development

13.1 Why source and not visual editor?

13.2 Markdown Basics

13.3 Including Code Chunks

13.4 Saving Formats

References