0. Getting Started

Bernd Wurth

Table of contents

1 Introduction

1.1 What is R?

R is a powerful, open-source programming language and software environment for statistical computing, data analysis, and graphical visualisation. It provides a wide variety of statistical and graphical techniques, including linear and nonlinear modeling, time-series analysis, classification, clustering, and more.

1.2 Brief History of R

R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team. The project was conceived in 1992, with an initial version released in 1995 and a stable beta version in 2000. R is named partly after its creators (shared first

letter of the authors, Ross and Robert) and partly as a play on the name of the S language, which it was designed to emulate. While S was a commercial software, R was created as a free alternative, gaining wide adoption due to its open-source nature and flexibility.

1.3 Used of R in Research and Data Analysis

R has become increasingly popular in academic fields such as social sciences, economics, entrepreneurship, and business/management as well as in corporate environments. Its appeal stems from being an open-source, free platform, making it globally accessible to researchers and students. R's extensive ecosystem of user-contributed packages significantly expands its functionality, while its scripting capability ensures reproducibility of analyses. Researchers and analysts leverage R's comprehensive statistical and machine learning techniques to explore data trends, build predictive models, and create publication-quality visualizations that inform decision-making. The language is further bolstered by a large, active community that continuously contributes to its development and provides abundant learning resources, making R an invaluable tool for data exploration, analysis, and presentation across various disciplines.

1.4 Key Features and Advantages of R

R's versatility makes it an excellent tool for various aspects of research:

1. **Data wrangling**: R excels at cleaning, transforming, and restructuring data. Packages like `dplyr` and `tidyr` provide intuitive ways to manipulate data.
2. **Statistical analysis**: From basic descriptive statistics to advanced modeling techniques, R covers a wide range of statistical methods. It's particularly strong in areas like regression analysis, ANOVA, time series analysis, and machine learning.
3. **Data visualisation**: The `ggplot2` package, part of the tidyverse, allows for the creation of complex, publication-quality visualisations with a consistent and intuitive syntax.
4. **Large dataset handling**: R can efficiently work with large datasets, especially when using packages optimized for big data, such as `data.table` or `spark`.
5. **Reproducible research**: R Markdown and Quarto (like this document) allow for the integration of code, results, and narrative, facilitating reproducible research practices.
6. **Extensibility**: R's package system allows users to easily extend its functionality, making it adaptable to specific research needs.
7. **Community support**: R has a vibrant community that provides help and feedback and continuously develops and shares packages.

1.5 R's interface with other tools

R can integrate with various other tools and technologies, allowing for flexible workflows across different technologies. Example include:

- Python: The `reticulate` package allows R to interface with Python, combining the strengths of both languages.
- Databases: R can connect to various databases (e.g., SQL, MongoDB) for data retrieval and storage.
- Web technologies: Packages like `shiny` allow for the creation of interactive web applications using R.
- Version control: R projects can be managed with Git and GitHub, facilitating collaboration and version control.

2 Introduction to RStudio

2.1 What is RStudio?

RStudio is an integrated development environment (IDE) specifically designed for R. It provides a user-friendly interface that makes working with R more accessible and efficient, especially for beginners.

2.2 How RStudio enhances the R programming experience

RStudio improves R programming productivity by offering a unified platform that integrates all aspects of the R workflow. Within a single window, users can write, edit, and execute R code, visualize results, and manage

files efficiently. The IDE features a sophisticated code editor with syntax highlighting and auto-completion, streamlining the coding process. It provides seamless access to R documentation and help files, facilitating quick reference and learning. RStudio's integrated plot and data viewers allow for immediate visualisation and inspection of results. The platform also includes robust project management tools to organize work effectively. Furthermore, RStudio's built-in support for version control systems like Git enables smooth collaboration and code versioning, making it an all-encompassing solution for R programmers of all levels.

2.3 Key features of RStudio

RStudio's interface is divided into four main panes (Figure 1):

1. **Source Editor**: Write and save R scripts for easy reproducibility.
2. **Console**: Interact with R directly for quick calculations or testing code snippets.
3. **Environment/History**: Displays your current workspace objects and command history and allows t.
4. **Files/Plots/Packages/Help**: A multi-purpose pane for file management, viewing plots, managing packages, and accessing help documentation.

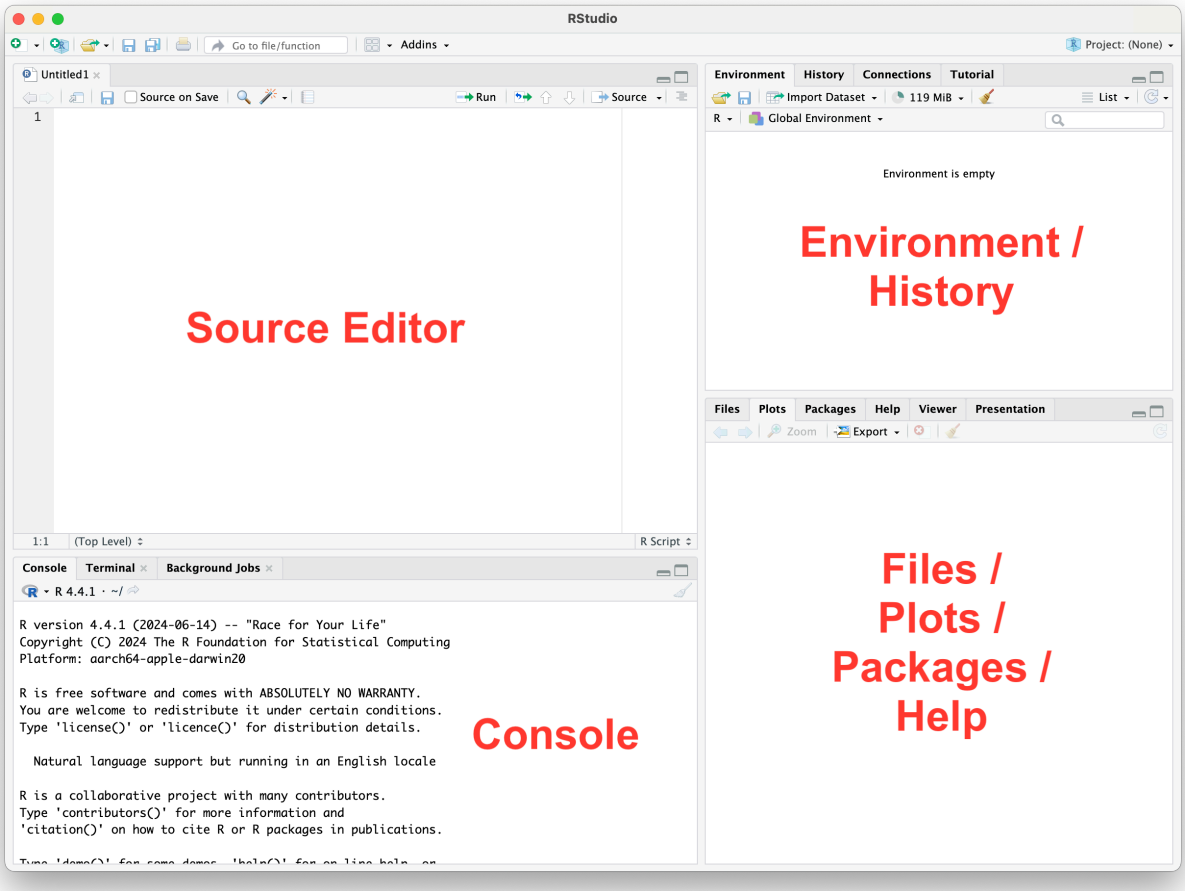

Figure 1: Overview of the RStudio user interface

2.4 RStudio Cloud Option

For students who prefer not to install R locally, RStudio Cloud offers a browser-based alternative. With a free account, you can create projects and run R code in your browser without the need for installation. To set up a free RStudio Cloud account:

1. Visit https://rstudio.cloud/
2. Click "Get Started for Free"
3. Sign up using your email or Google account
4. Once logged in, you can create new projects and start using R immediately in your browser

RStudio Cloud provides a consistent environment across different computers and operating systems, which can be particularly useful for collaborative work or when working on multiple devices.

3 Installation Guide

3.1 System requirements for R and RStudio

Before installing R and RStudio, ensure your system meets these requirements:

- For Windows:
  - Windows 7 or later
  - 32-bit or 64-bit operating system
- For Mac:
  - macOS 10.13 (High Sierra) or later
  - 64-bit operating system

Both R and RStudio are relatively lightweight programs and should run on most modern computers.

3.2 Step-by-step installation process for Windows

1. Download R:

Go to https://cran.r-project.org/

Click on "Download R for Windows" (Figure 2)

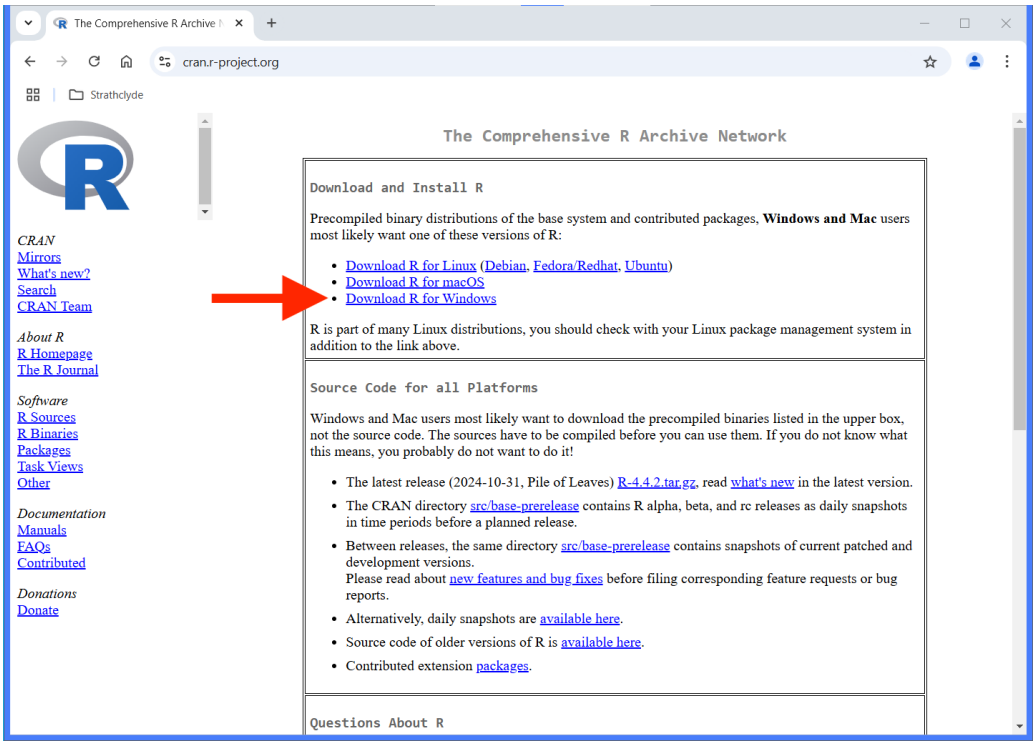

Figure 2: CRAN homepage (red arrow pointing to the link for downloading R for Windows)

Click on "base" (Figure 3)

Click on the download link for the latest version (Figure 4)

2. Install R:

Run the downloaded .exe file

Follow the installation wizard, accepting the default options

3. Download RStudio:

Go to https://www.rstudio.com/products/rstudio/download/

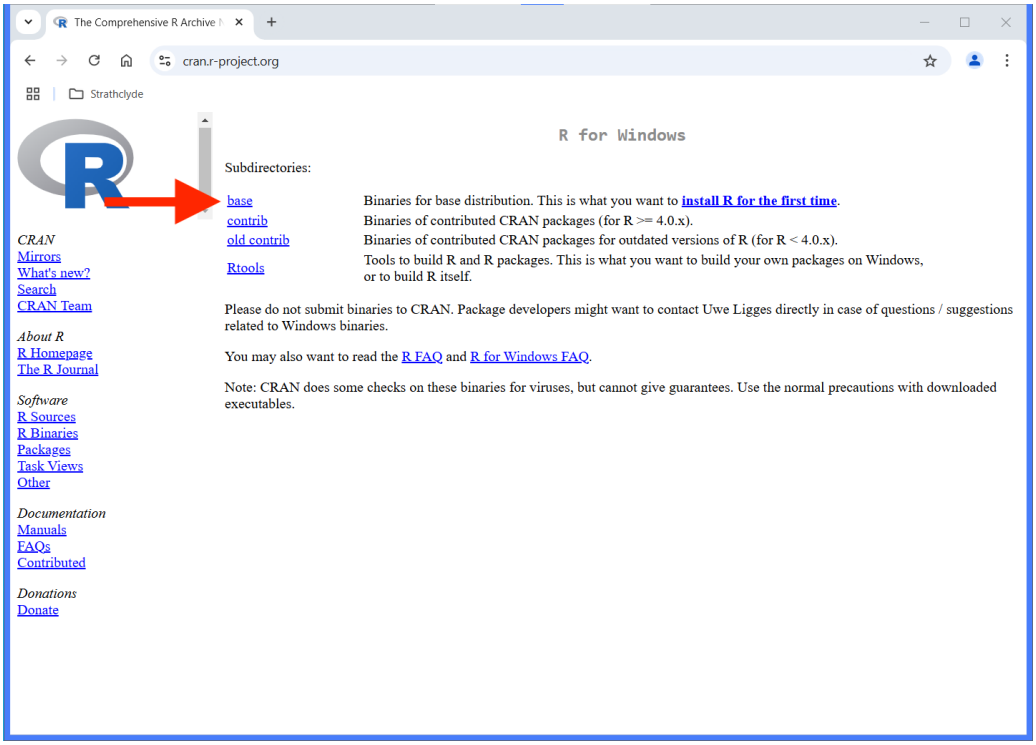

Figure 3: CRAN download page with subdirectories for Windows (red arrow pointing to the link for first time installation of R)

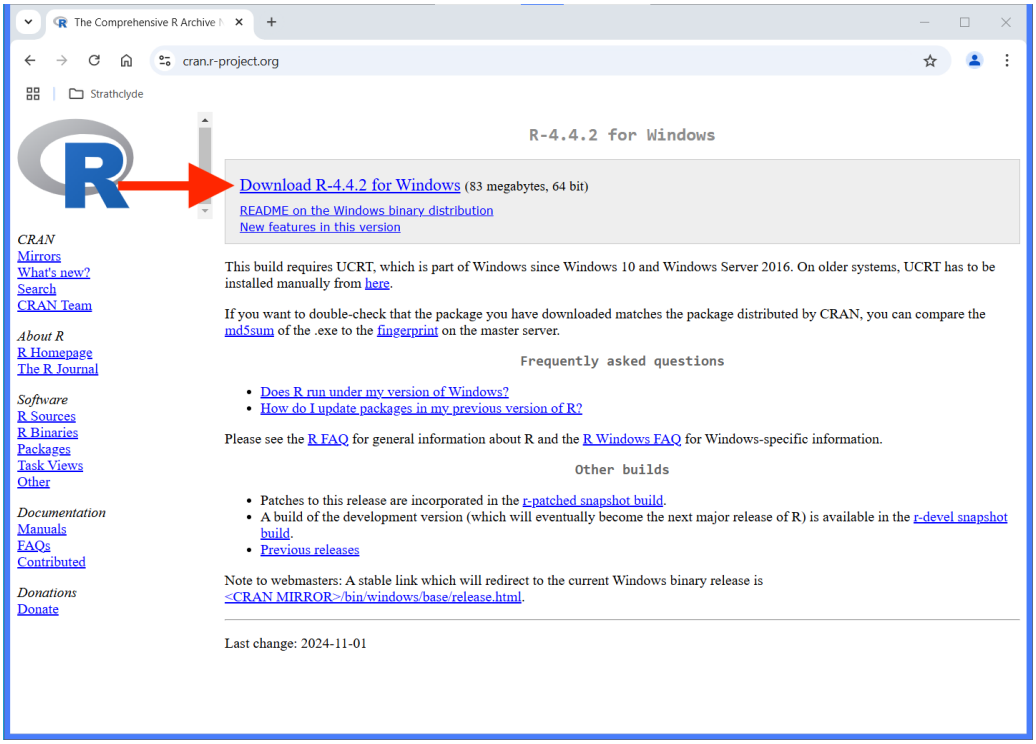

Figure 4: CRAN download page for base version for Windows (red arrow pointing to the download link)

Scroll down to “RStudio Desktop”

Click on “Download RStudio for Windows” (Figure 5)

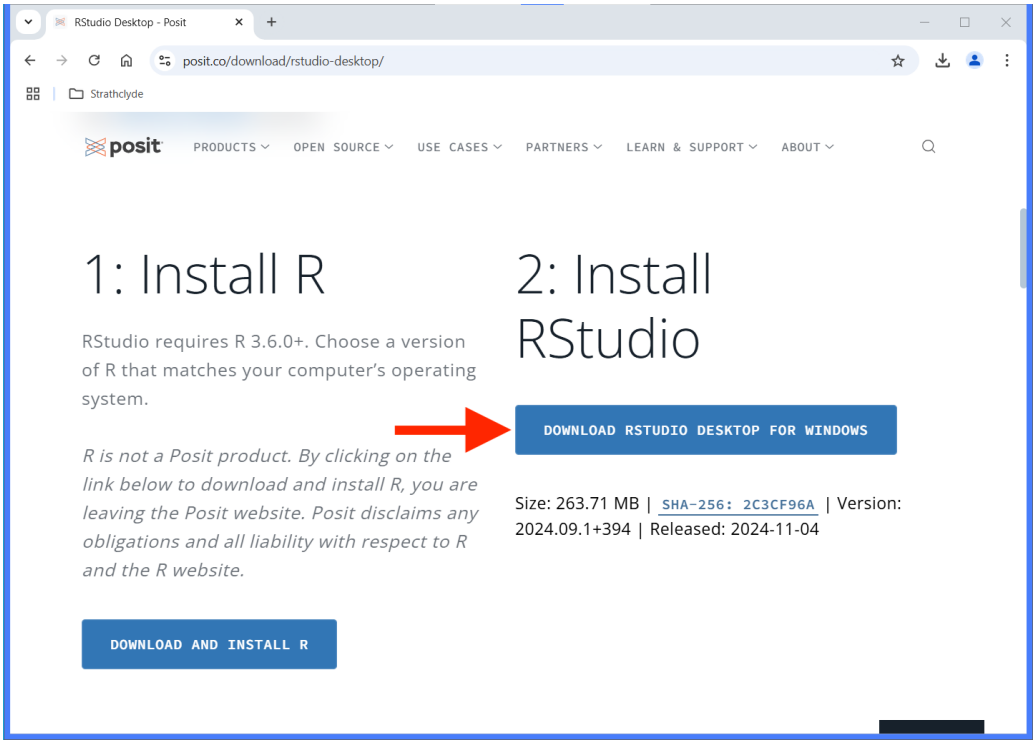

Figure 5: Posit page for RStudio (red arrow pointing to the download link for Windows)

4. Install RStudio:

Run the downloaded .exe file

Follow the installation wizard, accepting the default options (Figure 6)

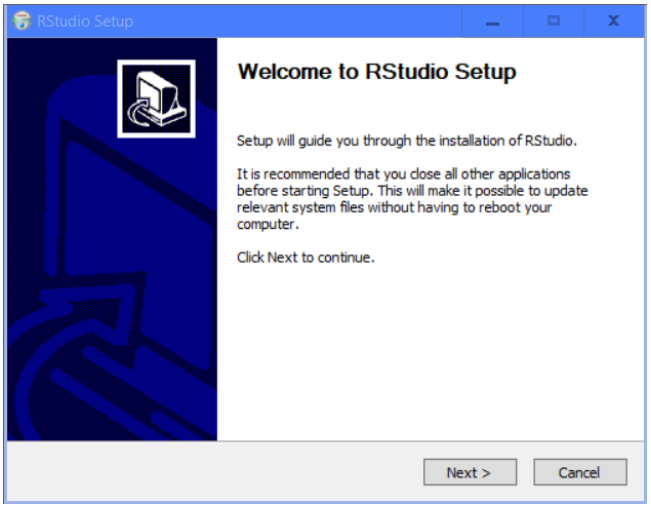

Figure 6: RStudio installation wizard for Windows

3.3 Step-by-step installation process for Mac

1. Download R:

Go to https://cran.r-project.org/

Click on “Download R for macOS” (Figure 7)

Click on the .pkg file appropriate for your macOS version (Figure 8)

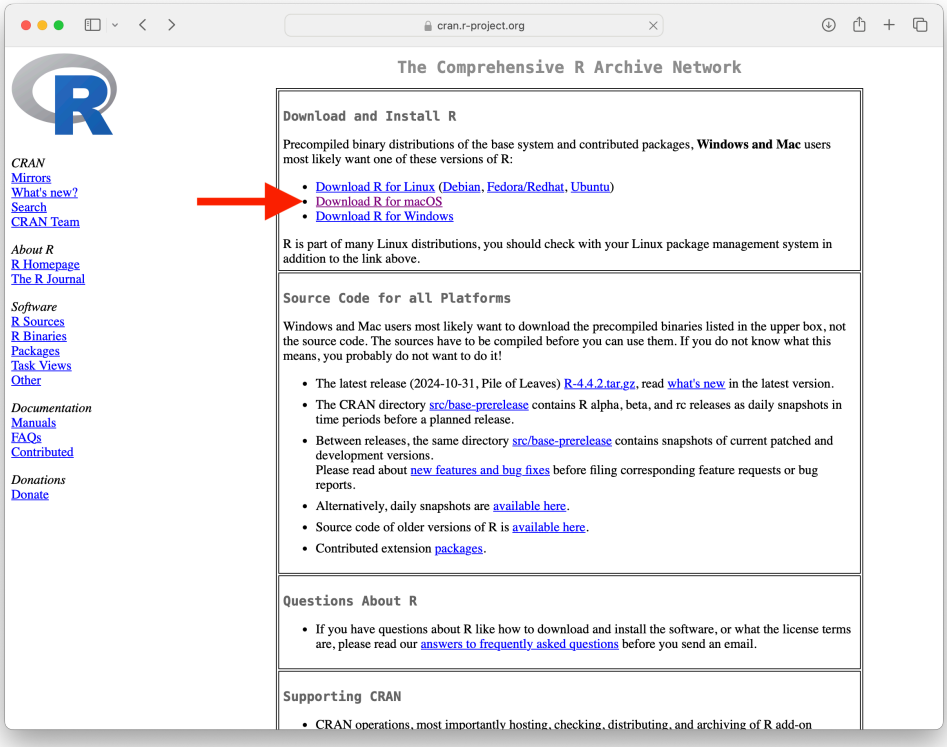

Figure 7: CRAN homepage (red arrow pointing to the link for downloading R for Mac

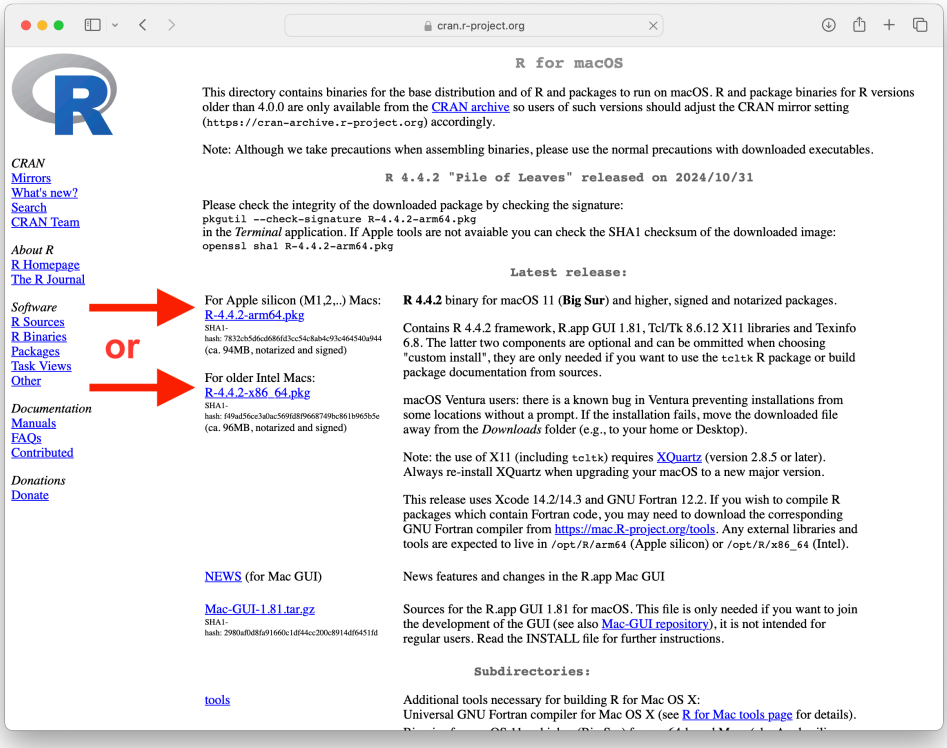

Figure 8: CRAN download page for Mac (red arrows pointing to the download link for respective Mac version, depending on whether you have a newer Apple Silicone Mac or an older Intel Mac)

2. Install R:

Open the downloaded .pkg file

Follow the installation wizard, accepting the default options (Figure 9)

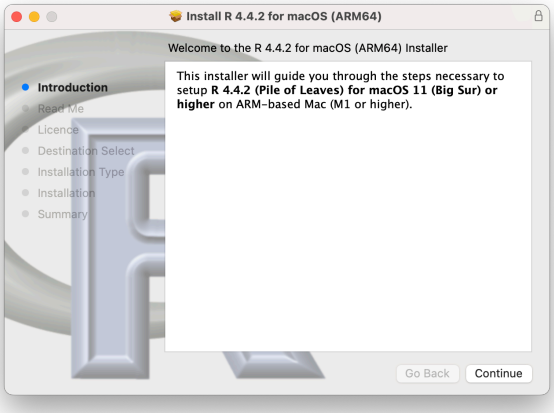

Figure 9: Instalation window for R (Mac)

3. Download RStudio:

Go to https://www.rstudio.com/products/rstudio/download/

Scroll down to "Install RStudio" and click on "Download RStudio Desktop for macOS" (Figure 10)

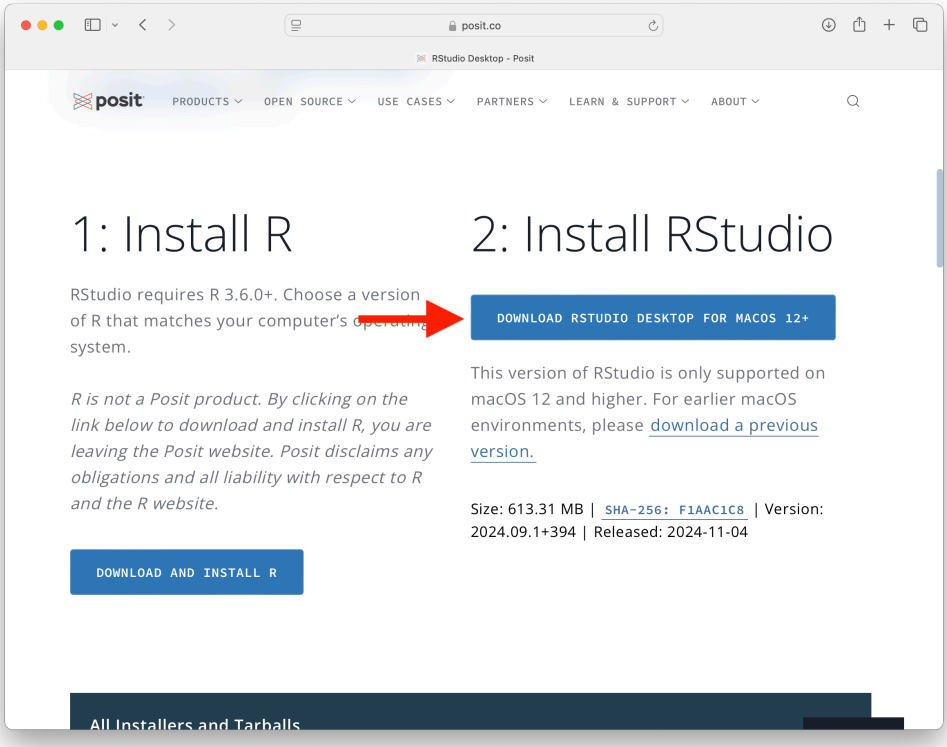

Figure 10: Posit page for RStudio (red arrow pointing to the download link for Mac)

4. Install RStudio:

Open the downloaded .dmg file

Drag the RStudio icon to your Applications folder (Figure 11)

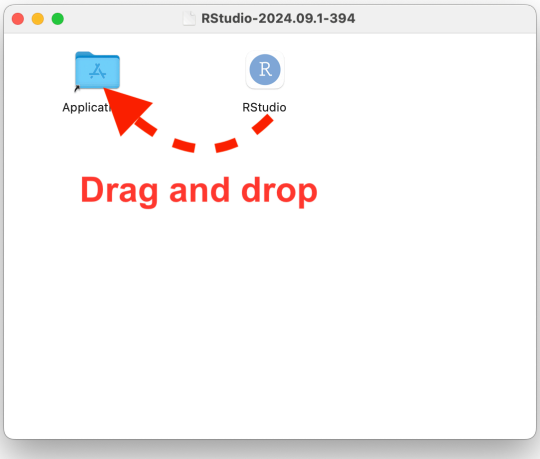

Figure 11: Installation window for RStudio on Mac (drag-and-drop the RStudio icon to the Applications folder)

3.4 Verifying successful installation

After installation, open RStudio. You should see the console and other features of RStudio. In the Console pane (usually bottom-left), you should see information about the R version. Type `1 + 1` in the Console and press Enter. If you get the result `2`, both R and RStudio are working correctly.

3.5 R and RStudio Updates

Keeping R and RStudio up to date is important for accessing the latest features and bug fixes:

- To check for R updates:
  1. Open RStudio
  2. Go to Tools > Check for Updates
  3. If an update is available, it will prompt you to install it
- To check for RStudio updates:
  1. Open RStudio
  2. Go to Help > Check for Updates
  3. If an update is available, it will prompt you to install it

It's generally a good practice to update both R and RStudio every few months or when starting a new project.

4 R Packages: An Overview

4.1 What are R packages?

R packages are collections of R functions, data, and documentation that extend the capabilities of base R. They are the fundamental units of reproducible R code, allowing users to easily share and reuse code.

4.2 The importance of packages in extending R's functionality

R packages provide users with access to specialised, pre-written functions, eliminating the need to code complex operations from scratch. These packages typically undergo rigorous testing and maintenance, ensuring code consistency and reliability. By offering a standardised method for sharing code and methodologies, packages facilitate collaboration among researchers and developers. Moreover, they significantly expand R's capabilities, extending its reach into specific domains, ranging from advanced statistical techniques to interfaces with other software systems. This extensibility through packages makes R a versatile and powerful tool adaptable to a wide array of analytical challenges across various fields.

4.3 Brief introduction to CRAN

The Comprehensive R Archive Network (CRAN) serves as the official repository for R packages, hosting thousands of user-contributed extensions to the R language. CRAN ensures the quality and consistency of its

offerings through a rigorous review process for all submitted packages. Users can easily install these packages directly within R using the install.packages() function, streamlining the process of extending R's capabilities. Furthermore, CRAN provides comprehensive documentation and vignettes for each package, offering users detailed information on functionality, usage, and implementation. This centralized, curated repository plays a crucial role in maintaining R's ecosystem, facilitating easy access to a vast array of tools and functions for R users worldwide. To explore CRAN, visit https://cran.r-project.org/.

5 Best Practices for Getting Started

5.1 Projects and setting up a working directory

5.1.1 Why does this matter?

When working in R, one of the key concepts you'll encounter is managing your **working directory**. The working directory is the folder on your computer where R looks for files (e.g., data) to load and saves files you create. While there are different ways to set your working directory, this lesson will discuss the benefits and drawbacks of two common methods: using `setwd()` and RStudio **Projects**.

5.1.2 What is `setwd()`?

The `setwd()` function in R sets the working directory to a specific folder. Below is the code example for setting your working directory. Alternatively, you can use Session > Set Working Directory > Choose Directory in RStudio.

```
setwd("C:/Users/YourName/Documents/MyProject")
```

Benefits of `setwd()`

- Quick setup: Useful for ad hoc or one-time analyses.
- Simple to understand: Easy to use for beginners doing small, standalone projects.

Drawbacks of `setwd()`

- Not portable: If you share your code with someone else, it may not work because their folder paths are different.
- Error-prone: Forgetting to set the correct working directory can cause errors when loading or saving files.
- Bad practice for larger projects: As projects grow, managing file paths with `setwd()` becomes cumbersome and difficult to maintain.

5.1.3 What Are RStudio Projects?

RStudio Projects provide a structured way to manage your working directory. When you create a Project in RStudio, a special file (`.Rproj`) is created. Opening this file automatically sets the working directory to the folder containing the Project. You can create a Project in RStudio by following these steps:

1. Go to File > New Project
2. Choose a new or existing directory
3. This creates an .Rproj file and sets the working directory automatically

Further information on how to set up Projects in RStudio can be found in the RStudio Projects Guide.

Benefits of RStudio Projects

- Portability: Code and file references work seamlessly on any computer without modification.
- Organization: Keeps all related files (data, scripts, output) in one folder.
- Best practice: Encourages better habits for managing larger or collaborative projects.
- Integration: Works well with version control systems like Git, making collaboration easier.

Drawbacks of RStudio Projects

- Learning curve: May feel complex for students doing very small, simple tasks.
- Overhead for small tasks: Setting up a Project for quick analyses might seem unnecessary.

> **Note**
>
> For most work—-especially as your projects grow in size or complexity—-we recommend using **RStudio Projects**. While it may feel like extra work upfront, it fosters reproducibility and reduces errors, hereby saving time and frustration in the long run. Use `setwd()` sparingly and only for temporary tasks.

5.2 How to organize project files for research

A well-organized project structure might look like this:

```
project/
   data/
       raw/
       processed/
   scripts/
   output/
       figures/
       tables/
   docs/
   project.Rproj
```

- `data/`: Store your data files
- `scripts/`: Keep your R scripts
- `output/`: Save generated figures and tables
- `docs/`: Store documentation and reports

5.3 Importance of commenting and code organization

Good coding practices improve readability and reproducibility:

1. Use clear and concise comments to explain your code
2. Organize your code into logical sections
3. Use meaningful variable and function names
4. Keep your code DRY (Don't Repeat Yourself)

Example of well-commented code:

```r
# Load necessary libraries
library(tidyverse)

# Read in the data
data <- read_csv("data/raw/survey_results.csv")

# Clean the data
clean_data <- data %>%
  filter(!is.na(age)) %>%  # Remove rows with missing age
  mutate(income = as.numeric(income))  # Convert income to numeric

# Calculate summary statistics
summary_stats <- clean_data %>%
  group_by(education) %>%
  summarize(
    mean_income = mean(income, na.rm = TRUE),
    median_age = median(age, na.rm = TRUE)
  )

# Print results
print(summary_stats)
```

5.4 Version control basics

Version control is essential for tracking changes in your code and collaborating with others. Git is a popular version control system, and GitHub is a platform for hosting Git repositories.

Basic Git concepts:

- Repository: A project's folder containing all files and version history
- Commit: A snapshot of your project at a specific point in time
- Branch: A parallel version of your repository
- Pull request: A method to propose changes to a repository

While we won't go into detail here, learning Git can greatly enhance your research workflow.

6 Additional Resources

6.1 Official R Documentation

The official R documentation is a valuable resource for learning about the functions and packages available in R.

- The R Project: https://www.r-project.org/
- R Documentation: https://www.rdocumentation.org/

In addition, the R Journal, a peer-reviewed open-access publication, serves as an invaluable resource for R users, offering in-depth articles on new packages, statistical methods, and applications of R in various fields, thereby providing both support for current users and insights into the evolving capabilities of the R ecosystem.

6.2 Recommended Books and Online Materials

Books

- "R for Data Science (2e)" by Hadley Wickham, Mine Cetinkaya-Rundel, and Garrett Grolemund [book | website | GitHub]
- "The Art of R Programming" by Norman Matloff [book]
- "Advanced R (2e)" by Hadley Wickham [book | website | GitHub]
- "ggplot2: Elegant Graphics for Data Analysis (3e)" by Hadley Wickham, Danielle Navarro, and Thomas Lin Pedersen [website | GitHub]
- "R Graphics Cookbook (2e)" by Winston Chang [book | website]
- "Text Mining with R: A Tidy Approach" by Julia Silge and David Robinson [book | website | GitHub]

Cheatsheets

Cheatsheets provide a handy reference guide for various aspects of working with R and RStudio, including RStudio, data tidying with tidyr, data transformation with dplyr, and data visualisation with ggplot2, among others (see others here).

Posit Recipes

Posit recipes (previously Posit primers) represent a collection of R code snippets and instructions featuring up-to-date best practices for coding in R: https://posit.cloud/learn/recipes.

GitHub Repositories and Online Course Materials

- University of Oregon (EC 607) by Grant McDermott [GitHub]

6.3 Community forums and support channels

There are a variety of other in-person and online resources available, including:

- Stack Overflow (R tag): https://stackoverflow.com/questions/tagged/r
- RStudio community: https://community.rstudio.com/
- R-Ladies (an organization to promote gender diversity in the R community): https://rladies.org
- R-bloggers: https://www.r-bloggers.com/
- #rstats on X (formerly Twitter)

Remember, the R community is known for being helpful and welcoming to newcomers. Don't hesitate to ask questions and engage with other R users as you begin your journey!