Using R with RStudio & Tidyverse

Arend M Kuyper

## Introduction

The purpose of this chapter is to provide you with an introduction to the free statistical software R along with RStudio and the tidyverse packages. While R can have a steep learning curve and be intimidating to new users, especially those new to coding, RStudio and the tidyverse packages make R much more accessible. Beyond improving accessibility, these tools are intentionally designed to make users more productive with R and improve the reproducibility of their work. Not surprisingly, this chapter is focused on the “how to” of foundational data work. We will highlight and demonstrate essential best practices for using R with RStudio and the tidyverse for quantitative research. However, don’t let this chapter’s focus on first steps of data work deceive, R and its ecosystem of extension packages allow for the implementation of all the statistical techniques found in this book and much, much more.

## R

R is a powerful and highly flexible statistical analysis tool. It provides a wide array of statistical techniques and methods while also providing highly developed graphics capabilities. R’s ability to create publication-quality graphics has been, and continues to be, one of its greatest strengths. It does all of this free of charge with a dedicated community of developers.

The R project is a GNU project, which means it is a free software, mass collaboration project. Knowing that R is open source and is actively developed and maintained through mass collaboration provides important context for users concerning its basic structure and potential resources. R can be considered as being made up of two 2 parts:

1. the base R system that is downloaded from the Comprehensive R Archive Network, also known as CRAN, and
2. a large ecosystem of extension packages, sometimes called libraries.

The base R system is actively maintained and updated for various operating systems by the R Core Team. Typically there is 1 major update along with 2 minor updates per year. Having an active release schedule like this is critical for the success of open source software. This ensures up-to-date compatibility with operating systems and signals to users that it won’t be abandoned.

While the base R system adequately covers most statistical needs and functionality, it is arguably the large ecosystem of extension packages that has contributed to R’s growth. Packages can provide implementations of methodologies not currently in base R, improve the usability of R, or provide tools that allow you to do non-statistical tasks (i.e. sending emails or building websites). In particular, the tidyverse packages have been very influential and have made working with R significantly more accessible. We will take a closer look at the tidyverse packages later in this chapter.

As of this writing, the CRAN package repository features over 21,000 contributed packages. The CRAN repository checks all packages for compatibility and expects the packages to be maintained, which means users can expect packages from the CRAN repository to work with R. This work is done by a network of volunteers, called the CRAN team, and it is a testament to the size and dedication of the R community that this is possible. Though CRAN is the primary R package repository, users can find packages through:

* CRAN-like repositories such as BioConductor and R-forge;
* GitHub and BitBucket;
* Personal websites.

While packages outside of the CRAN repository aren’t vetted by the CRAN team for compatibility, they can be very useful. They may implement cutting edge statistical techniques or provide tools for more bespoke analyses. Going through the CRAN submission process can be daunting, time intensive, and restrictive so it is not uncommon to find very useful packages not hosted on CRAN.

### Using R

A common roadblock for many new R users is that it requires the users to write code or commands. This can be a significant hurdle for many, but there are several free software options that make working with R much more user-friendly. The most popular being RStudio, which we will discuss in more detail later on in the chapter. The need to write code or commands isn’t removed, but it is made much more intuitive and accessing help is made easier. Using R and having to write code is a net positive for increasing the reproducibility of research, at least for computational and analysis work.

The value of learning to write R code is significantly enhanced by following best practices for coding and setting up workflows. When users are first learning it can seem unnecessary to follow such advice, but it is important to avoid developing bad and inefficient habits. RStudio and the tidyverse are specifically designed to guide users to follow and implement best practices. We will be highlighting and demonstrating some of these best practices in the following sections, but readers wanting more guidance should see the **Suggested further readings**.

## RStudio

RStudio is an integrated development environment (IDE) designed to make working with R more accessible and productive. While R comes with its own graphical user interface (GUI), it simply was not designed with a wide range of users in mind. So, it is common that R be paired with some other open source software such as RStudio, R Commander, Deducer, jupyter notebooks, vscode, or positron. RStudio is by far the most widely used and known. A significant portion of R’s growth in usage can be reasonably attributed to RStudio. It has become synonymous with R. RStudio can be downloaded for free from https://posit.co/download/rstudio-desktop/.

When RStudio is first opened, there are 4 panes as seen in [Figure 1](#fig-rstudio-layout). Sometimes the source pane is missing, but that is easily remedied by opening a new R script (.R file): **File > New File > R Script**.

1. The **Source pane** is where you can edit and save R scripts, which are essentially text files containing R code. This is where most of the data analysis work happens and should be documented.
2. The **Console pane** is used to write short interactive R commands.
3. The **Environment pane** displays temporary R objects as created during that R session. It also contains the useful history tab.
4. The **Output pane** displays the plots, tables, or HTML outputs of executed code along with saved files. This pane also includes the packages and help tabs which are especially useful since the first is for managing and installing packages and the second is setup to help access documentation.

|  |
| --- |
| Figure 1: RStudio’s basic layout |

### Prepare for success

Before starting to work with data, there are 2 best practices that should be discussed:

1. Adjusting a few RStudio default options to improve long-term reproducibility of your data analysis work.
2. Use RStudio projects to improve organization and collaboration, which includes with your future self.

By default, workspaces will load everything that you had been working on previously, from .Rdata files. While this might sound harmless or even desirable, it actually creates a workflow that could easily lead to ghost or zombie objects. That is, objects that are not reproducible because we may have ran code out of order ot altered code and forgot to re-run it to update things. By being automatically saved we might not catch this error until it is way too late. So, to develop R scripts that are complete and self-contained records of the data work we need to make a few adjustments. In RStudio, set this via **Tools > Global Options**, uncheck “Restore .RData into Workspace at Startup” and choose **Never** on the “Save workspace to .RData on exit” as seen in [Figure 2](#fig-rstudio-wrk-options).

|  |
| --- |
| Figure 2: RStudio’s Globol Options: Workspace Best Practice |

The larger and more sprawling research and analysis work gets, the more important it becomes to be organized. Meaning, it is important to have a “home base” of operations where everything for your data analysis or research work is self contained. By using RStudio projects it becomes straightforward to organize your work. You can think of an RStudio project as being a home directory/folder that will ultimately contain everything for your data analysis project. RStudio projects provide a solid workflow that will serve you well in the future:

* When starting a data analysis project (or any work in R), create a new RStudio project,
* Keep all data files there; organized in a data sud-directory.
* Keep all scripts there; edit them, run them in bits or as a whole. Naming them sequentially (e.g. 0-loading-data.R, 1-inspecting-data.R, etc) is recommended.
* Save your outputs (plots and cleaned data) there. Organized into appropriate sub-directories or folders

By using an RStudio project, everything you need is in one place, and cleanly separated from all the other projects that you are working on. The also make collaboration easier. The folders could be maintained on a shared drive so or version control software like git can be integrated into the projects. If used appropriately, then we should be able to simply zip/compress the RStudio project folder and share it with anyone else.

To create a new project in the RStudio, use the **File > New Project**. In the New Project wizard that pops up, select **New Directory**, then **New Project**. Pick a name for the project, for example “cwift-examples” (name of project for examples provided later in this chapter), and then click the **Create Project** button. This will launch a new RStudio Project inside a new folder called “cwift-examples”. The name of the project should appear in the top-right hand corner of Rstudio as seen in FIGURE REFERENCE. A good piece of advice, if it says “Project: (None)”, then don’t do any work. Set up a miscellaneous project for scratch work and musings. Always be working in a project.

## Tidyverse

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures. (*Tidyverse*, 2024)

This generally makes the tidyverse syntax easier to read and implement, especially for those new to coding and data analytic work. It makes chaining together data processing steps intuitive and easier than in other programming languages. The core tidyverse packages are those most likely needed for everyday data analyses and are all loaded by the meta {tidyverse} package. As of {tidyverse} 2.0, the following 9 packages are included in the core tidyverse:

* {ggplot2} is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.
* {dplyr} provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges.
* {tidyr} provides a set of functions that help you get to tidy data. Tidy data is data with a consistent form: in brief, every variable goes in a column, and every column is a variable.
* {readr} provides a fast and friendly way to read rectangular data (like csv, tsv, and fwf).
* {purrr} enhances R’s functional programming toolkit by providing a complete and consistent set of tools for working with functions and vectors.
* {tibble} is a modern re-imagining of the data frame, keeping what time has proven to be effective, and throwing out what it has not.
* {stringr} provides a cohesive set of functions designed to make working with strings as easy as possible.
* {forcats} provides a suite of useful tools that solve common problems with factors (categorical variables).
* {lubridate} provides a set of tools to make it easier to work with dates and times in R, which can be difficult and inconsistent in base R.

To install the tidyverse packages, go to **output pane > Packages > Install**. The package install wizard will pop-up and enter “tidyverse” as the name of the page to download. Ensure that “Install dependencies” is checked, then click **Install**. Alternatively, we could have ran install.packages("tidyverse") in the console. Not only will we get the 9 core tidyverse packages, but all of the packages they are dependent on will also be installed.

Beyond the core tidyverse packages, we will be using several additional packages as we work through a few data explorations in the next section. Packages such as tidycenus (tools to obtain census data in tidy format), {janitor} (tools for cleaning data), {sf} (tools for making maps), and {skimr} (tools to quickly skim data). Packages like {tidycensus} and {janitor} are often described as tidyverse-adjacent because they are built to work with the tidyverse and therefore adhere to the shared design philosophies.

The tidyverse’s influence on the R package ecosystem has been transformative. It has laid a foundation upon which developers in a large open-source community can collaborate and build tools that are much more accessible (i.e. tidyverse-adjacent packages). Developers can build their packages to leverage all existing tools and benefits of the tidyverse ecosystem which helps to alleviate dependency issues and compatibility with current and future features. Ultimately, this means users will be able to pick up new packages much more efficiently because of the shared design philosophies and standards.

## Data Exploration Examples

Pipping

# non-piping  
f(g(h(my\_data)))  
  
# piping  
my\_data |>   
 h() |>   
 g() |>   
 f()

### Dataset

We will be using data from the Education Demographic and Geographic Estimates (EDGE) Program. Specifically we will be using the American Community Survey Comparable Wage Index for Teachers (ACS-CWIFT or CWIFT) (*Comparable Wage Index for Teachers (CWIFT)*, 2024).

The Comparable Wage Index (CWI) is an index that was initially created by the National Center for Education Statistics (NCES) to facilitate comparison of educational expenditures across locales (principally school districts, or local educational agencies—LEAs) or states (state educational agencies— SEAs). The CWI is a measure of the systematic, regional variations in the wages and salaries of college graduates who are not PK-12 educators as determined by reported occupational category. It can be used by researchers to adjust district-level finance data at different levels in order to make better comparisons across geographic areas. (Cornman et al., 2019)

The CWIFT is designed to identify geographic variation in wages for college-educated workers outside of the education field after controlling for job-related and demographic characteristics. The basic premise of any CWI is that all types of workers demand higher wages in areas where the cost of living is high or desirable local amenities (such as good climate, low crime rates, or access to beaches, museums, and fancy restaurants) are lacking. As a result, it should be possible to measure most of the geographic variation in the cost of hiring teachers and other PK-12 educators by observing systematic, regional variations in the wages of comparable workers who are not PK-12 educators.

In theory, if accountants, nurses, and computer programmers, for example, all earn 5 percent more than the national average for their professions in Houston, then it is reasonable to expect that the cost of hiring teachers in Houston would also be 5 percent more than the national average for teachers.

Using the Index to Make Geographic Adjustments

One important reason for the development of the CWIFT is to enable more meaningful comparisons across school districts. To normalize dollar amounts and make them comparable, divide the dollar amounts by the district-level CWIFT, which are already normalized to the national average wage. For example, suppose one wished to make an adjustment to expenditure data from the Elementary and Secondary Information (ELSI) system for the 2013-14 school year. The 2015 CWIFT for Los Angeles Unified School District 1.129. So the $6,137 total current expenditures on salary per pupil in Los Angeles Unified for 2013-14, when normalized, are equal to $5,436 ($6,137 / 1.129). The 2013-14 total current expenditures on salary per pupil by Palm Beach County School District (in Florida) were $5,433.Normalized to reflect the lower cost of hiring in this area, they are the equivalent of $5,677 ($5,433 / 0.957). In other words, even though Los Angeles Unified School District spent more than Palm Beach County School District in nominal terms, once the two dollar figures were adjusted for the difference in purchasing power between the two districts, Palm Beach County School District effectively spent $241 more per pupil than did Los Angeles Unified School District.

Geographic Adjustment Applied to State Aid

Because one of the great virtues of the CWIFT is that it is outside of school district control, another application of the CWIFT is to adjust state aid to a school district for differences in wages. For example, consider a program intended to provide an additional $100 per pupil, but adjusted for geographic variations in the cost of education. The 2015 CWIFT for New Rochelle, NY in 2015 is 1.163, or 16.3 percent higher than the national average; the 2015 CWIFT for Buffalo, NY is 0.902, or approximately 10 percent lower than the national average. Therefore, to receive the same increase in purchasing power as a $100 increase in Buffalo City School District, New Rochelle City School District would need to receive $128.94 ($100\*(1.163 / 0.902))

### dplyr: data wrangling

Demo(s) with comments and a few tips for best practices

### ggplot2: data visualization

Demo(s) with comments and a few tips for best practices

## Comment: AI & R coding

chattr tidychatmodels GitHub copilot ask

## Conclusion

Other than R scripts, Quarto (.qmd) or R Markdown (.Rmd) documents could be used to document the work. While beyond the scope chapter, we encourge you to

version control git & git hub

## Research essentials (250-300 words)

## Questions for further investigation (50-100 words)

How does getting setup compare to other statistical software you may have used? If setup was an issue, have you searched for any free workshops or materials online? Try using R with RStudio and the tidyverse by playing with a data project of your own and/or by downloading the data exploration examples RStudio project from its GitHub webpage (https://github.com/akuyper/cwift-examples).

## Suggested further reading (50-100 words/3 resources & why)

Hadley Wickham Danielle Navarro & Pedersen (2024)

Hadley Wickham Mine Çetinkaya-Rundel & Grolemund (2024)

(Wickham, 2024)

(Wickham et al., 2019)

## References

*Comparable wage index for teachers (CWIFT)*. (2024). NCES Education Demographic; Geographic Estimates (EDGE); <https://nces.ed.gov/programs/edge/Economic/TeacherWage>.Accessed: 2024-08-21

Cornman, S. Q., Nixon, L. C., Spence, M. J., Taylor, L. L., & Geverdt, D. E. (2019). *Education demographic and geographic estimates (EDGE) program: American community survey comparable wage index for teachers (ACS-CWIFT)* (NCES 2018130). U.S. Department of Education; National Center for Education Statistics.Retrieved 2024-08-20 from http://nces.ed.gov/pubsearch/

Hadley Wickham, Danielle Navarro, & Pedersen, T. L. (2024). *ggplot2: Elegant graphics for data analysis (3e)*. <https://ggplot2-book.org/>.Accessed: 2024-08-20

Hadley Wickham, Mine Çetinkaya-Rundel, & Grolemund, G. (2024). *R for data science (2e)*. <https://r4ds.hadley.nz/>.Accessed: 2024-08-20

*Tidyverse*. (2024). <https://www.tidyverse.org/>.Accessed: 2024-08-15

Wickham, H. (2024). *The tidyverse style guide*. <https://style.tidyverse.org/index.html>.Accessed: 2024-09-03

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. <https://doi.org/10.21105/joss.01686>