

A Short Introduction to Working With Data in R

Jonathan Whiteley

2023-09-03

Prerequisites

- Access to a copy of the  software
 - ▶ i.e., a “binary executable”
 - ▶ Go to www.r-project.org to get a copy, or ask your system administrator.
- Tidyverse packages installed on the same system as R
 - ▶ Please run this command in R *before* the workshop:

```
install.packages("tidyverse")
```

- Knowledge of common mathematical operations: arithmetic, logarithms, etc.
- Knowledge of basic R concepts, such as *variables*, *objects*, *operators*, *functions*, *packages*, etc.
 - ▶ This is covered in the first workshop: “A Gentle Introduction to R”

Learning Objectives

- Load tabular data into R
- Explore data to check that it was loaded correctly
- Export data from R to external files
- Data frames
- Clean data
 - ▶ Add & change columns
 - ▶ Edit values systematically
 - ▶ Change data types
- Tidy data
 - ▶ Change the *shape* of a data frame
- Re-use code, reproducible results, automated reports
 - ▶ Scripts
 - ▶ R Markdown, R Notebooks

Section 1

Welcome

Pop Quiz

We will review these *at the end*, so you can see how much you have learned.

- If multiple packages have functions with the same name, how can you specify which one to use?
- Does R store data in memory or temporary files?
- What is the limit to the size of objects and datasets that can be loaded into R?
- TRUE or FALSE: R has rules and conventions for naming functions
- TRUE or FALSE: if you use one package from the tidyverse, you have to use all of them.

Answer in the chat:

What is your favourite emoji? Why do you like to use it so much?

Introductions

- Name
- Pronouns
- Job title, role
- *optional*: a favourite childhood treat or candy?
- What are you hoping to learn most in today's workshop?

Section 2

Loading data into R

Section 3

Exploring your data

Data frames

Section 4

Saving data outside R

Saving data outside R

Section 5

Re-using your code: scripts and other files

Re-using your code: scripts and other files

Section 6

The tidyverse collection of packages

The tidyverse

```
install.packages("tidyverse")  
help(package="tidyverse")
```

- The **tidyverse** is an “opinionated” **collection of packages** that are designed to work together.
- All packages share an underlying design philosophy, grammar, and data structures.

Today, we will focus on a few of the core tidyverse packages for loading, cleaning, and manipulating data:

- **readr**, **readxl** for **loading** data
- **dplyr** for **manipulating** data (values)
- **tidyr** for **rearranging** data
- **stringr** for working with **strings**

dplyr: grammar of data manipulation

- dplyr provides many functions, within a coherent framework or *grammar*
- They are intended to help you focus on *what* you want to do, and translate your thoughts into code.
- High-level functions have active names and called “verbs” — they describe what they do.
- dplyr and tidyr provide many supporting functions that work *inside* verbs to make many common tasks easier to translate into code.
 - ▶ These functions may not work on their own, outside of dplyr verbs and tidyr functions.

dplyr verbs

Verbs can be grouped based on the component of the dataset that they work with¹:

- Rows:
 - ▶ `filter()` chooses rows based on column values.
 - ▶ `slice()` chooses rows based on location.
 - ▶ `arrange()` changes the order of the rows.
- Columns:
 - ▶ `select()` changes whether or not a column is included.
 - ▶ `rename()` changes the name of columns.
 - ▶ `mutate()` changes the values of columns and creates new columns.
 - ▶ `relocate()` changes the order of the columns.
- Groups of rows:
 - ▶ `group_by()` defines groups of rows.
 - ▶ `summarise()` collapses a group into a single row.

¹<https://dplyr.tidyverse.org/articles/dplyr.html#single-table-verbs>

A 'pipe' operator



Figure 1: “La Trahison des Images” (“The Treachery of Images”) or “Ceci n'est pas une pipe” (“This is not a pipe”) by René Magritte.



- The `magrittr` package (included with `tidyverse`) provides a “forward-pipe operator”:

```
%>% # ?magrittr::`%>%`
```

- The `magrittr` package is automatically loaded when loading most `tidyverse` packages (e.g., `tidyr`, `dplyr`, `ggplot2`), as these packages all use this operator extensively.
 - ▶ It is often unnecessary to load `magrittr` separately, unless you are **not** using these other packages.

magrittr's 'forward-pipe' operator

- `%>%` allows you to pass results from an expression on the left-hand side (LHS) as an argument (usually the first) to a *function call* on the right-hand side (RHS).

This expression ...	is equivalent to:
<code>x %>% f()</code>	<code>f(x)</code>
<code>x %>% f(y)</code>	<code>f(x, y)</code>
<code>x %>% f(y, z = .)</code>	<code>f(y, z = x)</code>
<code>x %>% f %>% g %>% h</code>	<code>h(g(f(x)))</code>

- This can make code easier to read, as expressions are written and evaluated from *left to right*, rather than from *inside to outside* nested parentheses.

R now has a ‘native’ pipe operator

- A pipe operator was introduced in base R in v4.1 (May 2021)²:

```
|>      # ?pipeOp
```

- It was inspired by the “forward pipe operator” introduced by `magrittr`, but is more streamlined. See these links for details:
 - ▶ Differences between the base R and `magrittr` pipes
 - ▶ “Understanding the native R pipe `|>`”
- Because it is so new, most code examples online still use ‘`%>%`’ from `magrittr`.
- This document will use ‘`%>%`’ in the examples, for consistency and because it was designed to work with other tidyverse functions.
- But ‘`|>`’ might work well for you in simple cases, without having to load any additional packages.

²<https://cran.r-project.org/bin/windows/base/old/4.1.0/NEWS.R-4.1.0.html>

Pipes: exercise

Section 7

Clean data

Clean data

Section 8

Tidy data

Tidy datasets

Happy families are all alike; every unhappy family is unhappy in its own way

— *Leo Tolstoy*

Like families, tidy datasets are all alike but every messy dataset is messy in its own way.

— *Hadley Wickham* (doi: [10.18637/jss.v059.i10](https://doi.org/10.18637/jss.v059.i10))

- Tidy datasets provide a standardized way to link the *structure* of a dataset (its physical layout) with its *semantics* (its meaning).

▶ [tidyr vignette](#)

Section 9

Review

Exercise

Quiz Review

Section 10

Backmatter

Other packages to look at

- `data.table`: a high-performance version of `data.frame` with few dependencies.
- `purrr` (part of the `tidyverse`): functional programming (FP) tools for working with functions and vectors.
 - ▶ Replace `for` loops with code that is more efficient and easier to read.

Writing to Microsoft Excel™ files

Packages that can write to Excel files:

- **xlsx**: read, write, format Excel 2007 (.xlsx) and Excel 97/2000/XP/2003 (.xls) files.
 - ▶ Depends on Java and the rJava package
- **XLConnect**: comprehensive and cross-platform R package for manipulating Microsoft Excel files (.xlsx & .xls) from within R.
 - ▶ Requires a Java Runtime Environment (JRE)
- **openxlsx**: simplified creation of Excel .xlsx files (**not** .xls).
 - ▶ No dependency on Java
- **writexl**: portable, light-weight data frame to **xlsx** exporter.
 - ▶ No Java or Excel required

!

I recommend *avoiding* exporting data to Excel files if possible. csv files are easier to read to & write from, and can be read by a wider variety of software (they are more portable).

Automated reports can be produced with R Markdown and output to a variety of more portable formats (pdf, HTML, etc.) instead.

References

Cheatsheets:

- [readr/readxl](#)
- [Data transformation with dplyr](#)
- [Data tidying with tidyr](#)