C91AR | ADVANCED STATISTICS USING R

Lecture 3: Tidy Code Principles

Dr Pete McKenna

2025-01-24

HERIOT WATT UNIVERSITY
SoSS Doctoral Centre

LET'S CREATE AN R MARKDOWN FOR TODAY

- Call the file "Lecture3_tidy-code.rmd"

PACKAGES FOR TODAY'S SESSION

- Add this package installation to your Markdown file
- Remember to annotate inside your code chunks using #
- Also, you can add text summaries outside the code chunks to add clarity and interpretation.

```
# Let's stick to pacman to keep loading and installation simple
pacman::p_load(tidyverse,
               nycflights13,
               summarytools)
```

HERIOT WATT UNIVERSITY
SoSS Doctoral Centre

READING FOR TODAY'S SESSION

- Chapter 2: Workflow Basics
  - covers basics of writing R code (e.g., commenting and functions)
- Chapter 4 Workflow: code style
  - Covers the basics of code formatting
- Why did I ask you to read this?
  - These chapters provide valuable advice about how to set up your project and format your code.
  - The principles covered here will be the building blocks for your R journey and will make life a lot easier for you in the long-term.

HERIOT WATT UNIVERSITY
SoSS Doctoral Centre

ENCOUNTERING MESSY CODE

- What can you tell me about this bedroom?
- How does this image make you feel?

HERIOT WATT UNIVERSITY
SoSS Doctoral Centre

WHAT ARE WE GOING TO COVER TODAY?

- This is a hybrid session, involving both lecturing and coding
- We are going to use the `nycflights13` data to learn about and apply tidy code principles
- In doing so, we will get our first glimpse of *tidyverse* code
- Outline plans for Tutorial 2

WHAT IS THE *TIDYVERSE*?

- The *tidyverse* is a collection of R packages designed for data science.
- These packages share an underlying design philosophy, grammar, and data structures (e.g., datasets), making it easier to learn and use them together.
- The tidyverse simplifies many common data science tasks, making your code more readable and easier to maintain.

KEY TIDYVERSE VERBS AND SYNTAX

- `mutate()`: adds new variables or modifies existing variables.
- `select()`: picks (and amends) variables based on their names.
- `filter()`: subsets cases based on specified conditional criteria.
- `count()`: counts the number of occurrences of unique values.
- `summarise()`: reduces multiple values down to a single summary.
- `arrange()`: changes the ordering of the vectors.
- `|>`: the **pipe operator** means 'then do this'
- More information about Tidyverse verbs here

WHY ARE YOU TELLING ME THIS NOW?

The examples of tidy code formatting in this lesson use these verbs. We will examine these functions in more detail in future, but it's useful for you to have some grounding before discussion 'tidy code'.

flights METADATA

```
1  ?flights
```

A QUICK TOOL TO EXPLORE THE DATA

```
1  # Use summary tools to get an overview of the data
2  dfSummary(flights)
```

What can you infer from this summary?

SOME EXAMPLES

Naming variables | Spaces | Pipes | Long, tidy command chains

```
1  # Strive for:
2  short_flights <-
3    flights |> filter(air_time < 60)
4
5  # Avoid:
6  SHORTFLIGHTS <- flights |> filter(air_time < 60)
```

Given what we have learned so far, what do you think this code does?

CLEANUP

```
 1  # Clear data
 2  rm(list = ls())   # Removes all objects from environment
 3
 4  # Clear packages
 5  p_unload(all)   # Remove all contributed packages
 6
 7  # Clear plots
 8  graphics.off()   # Clears plots, closes all graphics devices
 9
10  # Clear console
11  cat("\014")   # Mimics ctrl+L
```

RECAP

- You read about tidy code principles
- I introduced you to the *tidyverse*
- We did a little a little exploration and summarising of `nycflights13` dataset
- Showed you how to clean up after yourself.

HERIOT WATT UNIVERSITY
SoSS Doctoral Centre

NEXT SESSION

- We will be applying these skills to an experiment dataset
- Learn how to read in datasets
- Perform more data cleaning using the *tidyverse*

HERIOT WATT UNIVERSITY
SoSS Doctoral Centre


HERIOT
WATT
UNIVERSITY
SoSS Doctoral
Centre