Advanced Data Wrangling in R

Dr. Adrian Correndo

2025-01-29

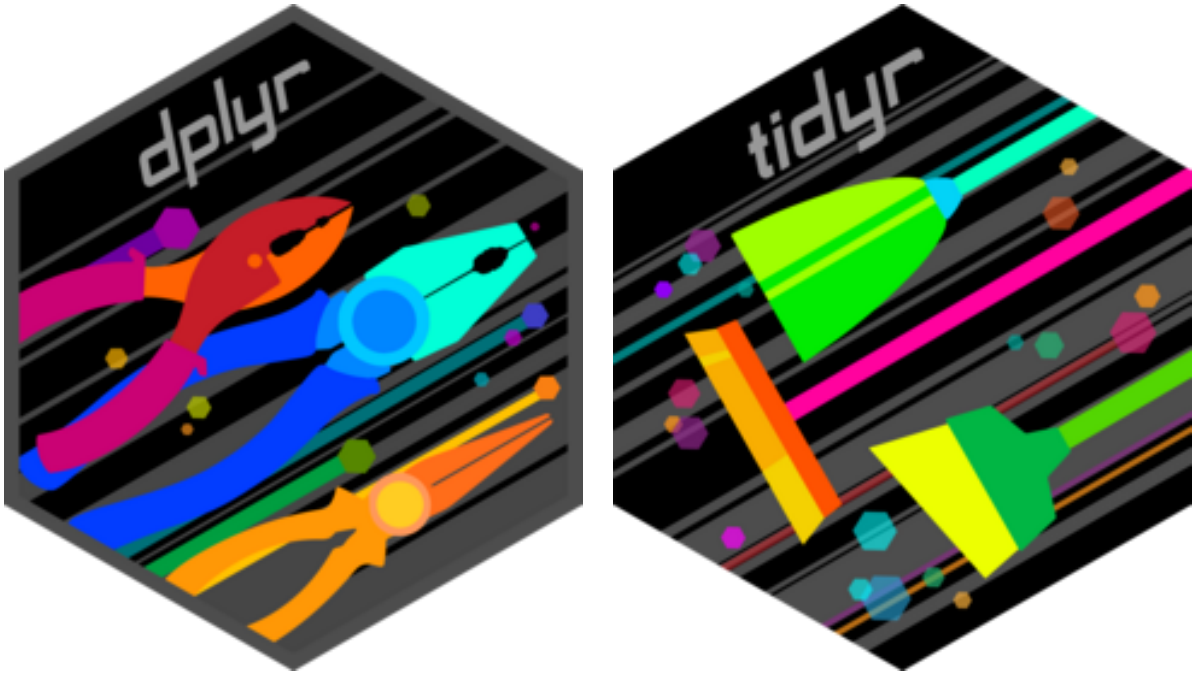
Description

This class dives deeper into the world of R data wrangling, covering advanced techniques and their applications. We'll explore complex functions and join operations essential for real-world data analysis.

Required packages for today

```
library(pacman)
p_load(dplyr, tidyr, stringr, lubridate, janitor)
```

1 Review of Previous Lessons



Quick recap of essential data wrangling functions:

1.1 `dplyr`

- `mutate()`
- `filter()`
- `select()`
- `slice()`, `slice_head()`, `slice_tail()`, `slice_sample()`
- `rename()`
- `arrange()`
- `distinct()`
- `count()`
- `glimpse()`
- `summarise()`
- `group_by`

1.2 `tidyr`

- `unite()`
- `separate()`
- `pivot_longer()`
- `pivot_wider()`
- `case_when()`
- `nest()`
- `unnest()`

1.3 `skimr`

- `skim()`

2 Complex Joins

In data analysis, combining data from different sources is often necessary. Here, we'll use `dplyr`'s join functions to merge datasets based on common keys.

2.1 Example: Merging Weather and Crop Yield Data

- `left_join()`: Includes all records from the left dataset and the matched records from the right dataset. If there is no match, the result is NA in the columns of the right dataset.

```
weather_data <- tibble(
  date = seq(as.Date("2024-04-01"), as.Date("2024-10-01"), by = "month"),
  precipitation = c(20, 40, 60, 80, 50, 30, 2),
  temperature = c(15, 18, 25, 30, 22, 16, 12)
)

forage_data <- tibble(
  date = seq(as.Date("2024-04-01"), as.Date("2024-10-01"), by = "month"),
  forage_yield = c(500, 1200, 3000, 4000, 2800, 1500, 0)
)

# Merge data
left_joined_data <- left_join(weather_data, forage_data, by = "date")
```

- `right_join()`: Includes all records from the right dataset and the matched records from the left dataset. If there is no match, the result is NA in the columns of the left dataset.

```r
# Merge data
right_joined_data <- right_join(weather_data, forage_data, by = "date")
```

- `full_join()`: Includes all records when there is a match in the keys of the left or right datasets. If there is no match, the result is NA in the columns of the dataset that does not have a match.

```r
# Merge data
full_joined_data <- full_join(weather_data, forage_data, by = "date")
```

3 Date Handling with `lubridate`

Using `lubridate`, we can extract various date components for analysis.

```r
weather_data_dates <- weather_data %>%
  mutate(
    year_month_day = format(date, "%Y_%m_%d"),
    day_of_year = yday(date),
    day_of_month = mday(date),
    week_of_year = week(date),
    month_name = month(date, label = TRUE, abbr = FALSE)
  )
glimpse(weather_data_dates)
```

```
Rows: 7
Columns: 8
$ date           <date> 2024-04-01, 2024-05-01, 2024-06-01, 2024-07-01, 2024-0~
$ precipitation  <dbl> 20, 40, 60, 80, 50, 30, 2
```

```
$ temperature    <dbl> 15, 18, 25, 30, 22, 16, 12
$ year_month_day <chr> "2024_04_01", "2024_05_01", "2024_06_01", "2024_07_01",~
$ day_of_year    <dbl> 92, 122, 153, 183, 214, 245, 275
$ day_of_month   <int> 1, 1, 1, 1, 1, 1, 1
$ week_of_year   <dbl> 14, 18, 22, 27, 31, 35, 40
$ month_name     <ord> April, May, June, July, August, September, October
```

4 Data Cleaning with `janitor`

Sometimes, datasets come with inconsistent column names, which can cause issues in analysis.

```
messy_weather_data <- tibble(
  `Date Recorded` = seq(as.Date("2024-04-01"), as.Date("2024-10-01"), by = "month"),
  `Precipitation (mm)` = c(20, 40, 60, 80, 50, 30, 2),
  `Temperature..(C,)` = c(15, 18, 25, 30, 22, 16, 12)
)

clean_weather_data <- messy_weather_data %>% clean_names()
clean_weather_data
```

```
# A tibble: 7 x 3
  date_recorded precipitation_mm temperature_c
  <date>                   <dbl>         <dbl>
1 2024-04-01                  20            15
2 2024-05-01                  40            18
3 2024-06-01                  60            25
4 2024-07-01                  80            30
5 2024-08-01                  50            22
```

```
6 2024-09-01                  30            16
7 2024-10-01                   2            12
```

5 Checking Data Quality

After merging, it's crucial to check for missing values and duplicates:

```
data_quality_summary <- full_joined_data %>%
  summarise(across(everything(), ~ sum(is.na(.)), .names = "missing_{.col}"))

duplicate_rows <- full_joined_data %>% get_dupes()
```

```
No variable names specified - using all columns.
```

```
No duplicate combinations found of: date, precipitation, temperature, forage_yield
```

```
data_quality_summary
```

```
# A tibble: 1 x 4
  missing_date missing_precipitation missing_temperature missing_forage_yield
         <int>                 <int>               <int>                 <int>
1            0                     0                   0                     0
```

```
duplicate_rows
```

```
# A tibble: 0 x 5
# i 5 variables: date <date>, precipitation <dbl>, temperature <dbl>,
#   forage_yield <dbl>, dupe_count <int>
```

6 Final Thoughts and Resources

Data wrangling is a crucial step in data analysis, ensuring datasets are clean, structured, and ready for further exploration. By leveraging `dplyr`, `tidyr`, `janitor`, and `lubridate`, we can efficiently manage and transform our data to extract meaningful insights.

For further reading and practice, consider the following resources:

- R for Data Science by Hadley Wickham & Garrett Grolemund

- tidyverse documentation
- lubridate cheatsheet
- Data Wrangling with R - A comprehensive tutorial

Keep practicing and experimenting with different datasets to solidify your understanding. Happy coding!