Data Wrangling

Dr Rafael de Andrade Moral
Associate Professor of Statistics, Maynooth University

rafael.deandrademoral@mu.ie
https://rafamoral.github.io

Outline

- Tidy data
- `dplyr` verbs
- `group_by` and `summarise`
- `merge` and `join`
- pivoting

Data Wrangling

- Cleaning, preparing, transforming
- Going from *messy* data to data ready to be analysed
- Wickham & Grolemund (2016) refer to this as *tidy* data
- Tidy data have 3 common features
  1. every variable has its own column
  2. every observation has its own row
  3. every cell contains one value

Common Problems with Messy Data

- Column headers are values, not variable names
- Multiple variables stored in one column
- Variables stored in both rows and columns
- Multiple types of observational units are stored in the same table
- Single observational unit stored in multiple tables

Base R vs. `dplyr`

- We can do data wrangling in R in different ways
  - ‘base R’ functions and indexing
  - using `data.tables`
  - the `tidyverse` way
- The `tidyverse` style of data wrangling is the most *readable*
- In this course we will use the `tidyverse` but compare with ‘base R’ from time to time

Readability: An Example

```
with(my_data, tapply(response, treatment, mean))
```

```
##          1          2          3          4          5
##  0.01509418  0.13897818  0.46683516 -0.17079731 -0.11251869
```

```
my_data %>%
  group_by(treatment) %>%
  summarise(avg = mean(response))
```

```
## # A tibble: 5 x 2
##   treatment     avg
##   <fct>       <dbl>
## 1 1          0.0151
## 2 2          0.139
## 3 3          0.467
## 4 4         -0.171
## 5 5         -0.113
```

Readability: It Can Get Worse

```
sort(
  with(my_data, tapply(response, treatment, mean))[
  with(my_data, tapply(response, treatment, mean)) > 0]
  )
```

```
##          1          2          3
## 0.01509418 0.13897818 0.46683516
```

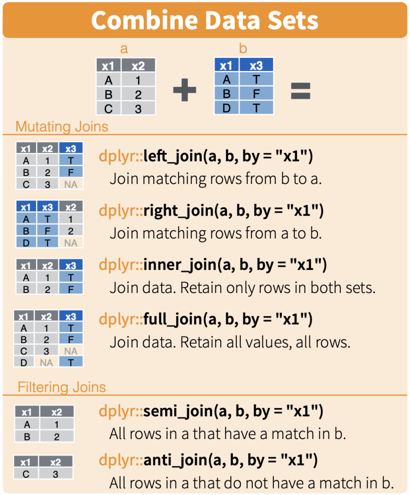
```
my_data %>%
  group_by(treatment) %>%
  summarise(avg = mean(response)) %>%
  filter(avg > 0) %>%
  arrange(avg)
```

```
## # A tibble: 3 x 2
##   treatment    avg
##   <fct>      <dbl>
## 1 1         0.0151
## 2 2         0.139
## 3 3         0.467
```

The `dplyr` verbs

- `select`: subsetting columns
- `relocate`: reordering columns
- `rename`: renaming columns
- `slice`: subsetting rows
- `filter`: subsetting rows according to a condition
- `mutate`: creating new variables/modifying variables
- `transmute`: minor variant of mutate
- `arrange`: sorting rows

join



Combine Data Sets

a

x1	x2
A	1
B	2
C	3

+

b

x1	x3
A	T
B	F
D	T

=

Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

dplyr::**left_join(a, b, by = "x1")**
Join matching rows from b to a.

x1	x3	x2
A	T	1
B	F	2
D	T	NA

dplyr::**right_join(a, b, by = "x1")**
Join matching rows from a to b.

x1	x2	x3
A	1	T
B	2	F

dplyr::**inner_join(a, b, by = "x1")**
Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

dplyr::**full_join(a, b, by = "x1")**
Join data. Retain all values, all rows.

Filtering Joins

x1	x2
A	1
B	2

dplyr::**semi_join(a, b, by = "x1")**
All rows in a that have a match in b.

x1	x2
C	3

dplyr::**anti_join(a, b, by = "x1")**
All rows in a that do not have a match in b.