Data Wrangling with R: Day 2

Relational data wrangling, starring `janitor` and `broom`

Presented by Emi Tanaka

Department of Econometrics and Business Statistics

MONASH University

2nd December 2020 @ Statistical Society of Australia | Zoom

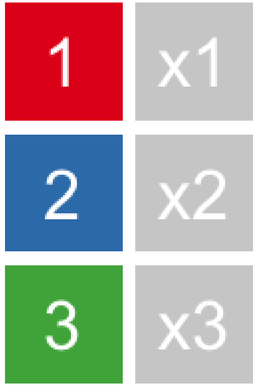
Relational data

Do you rememember what the definition of a tidy data is?

Definition of a tidy data

- Each variable must have its own column
- Each observation must have its own row
- Each value must have its own cell

Observational unit



- In the original definition in Wickham (2014), which was a statistical version of the definition in Codd (1990), the third point originally was actually:

> Each type of observational unit forms a table.

- What is an **observational unit**?
- It's the unit in which measurement is made, e.g. if measure height of a person then the person is the observational unit

Codd EF. 1990. The Relational Model for Database Management: Version 2. Addison-WesleyLongman Publishing, Boston.

Wickham, Hadley. 2014. "Tidy Data." Journal of Statistical Software, Articles 59 (10): 1–23.

Multiple observational units in one table

```
df
```

```
##      year variety height period temper
## 1    1974     G01   81.0     T1
## 2    1974     G01   81.0     T2
## 3    1974     G01   81.0     T3
## 4    1974     G01   81.0     T4
## 5    1974     G01   81.0     T5
## 6    1974     G01   81.0     T6
## 7    1975     G01   67.3     T1
## 8    1975     G01   67.3     T2
## 9    1975     G01   67.3     T3
## 10   1975     G01   67.3     T4
## 11   1975     G01   67.3     T5
## 12   1975     G01   67.3     T6
## 13   1976     G01   71.5     T1
```

- The data on the left shows the height of a barley variety at a given year in Norway experiment
- The temperature at six different time points of the growth barley was recorded
- What's the observational unit here?
- Yes. there are two observational unit here: the barley and the environment at six different time points per year

Related data sets

- Originally the data were in a separate table
- Notice before that the `height` measurements were duplicated
- While it's tidier to have it separated like this, you may need to join the data for downstream analysis

```
##     year period temperature
## 1   1974     T1        8.36
## 2   1974     T2        9.60
## 3   1974     T3       12.11
## 4   1974     T4       12.13
## 5   1974     T5       18.43
## 6   1974     T6       13.75
## 7   1975     T1        7.66
## 8   1975     T2       11.66
## 9   1975     T3       10.06
```

```
##      year variety height
## 1    1974     G01   81.0
## 2    1975     G01   67.3
## 3    1976     G01   71.5
## 4    1977     G01   64.3
## 5    1978     G01   55.8
## 6    1979     G01   84.9
## 7    1980     G01   86.2
## 8    1981     G01   88.0
## 9    1982     G01   72.0
```

So, how do you join two related data?

x

1	x1
2	x2
3	x3

y

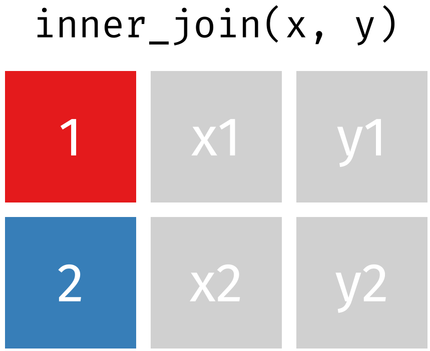
1	y1
2	y2
4	y4

Yes, you can only join the table if each table has columns that you can join by

Joining datasets with `dplyr`

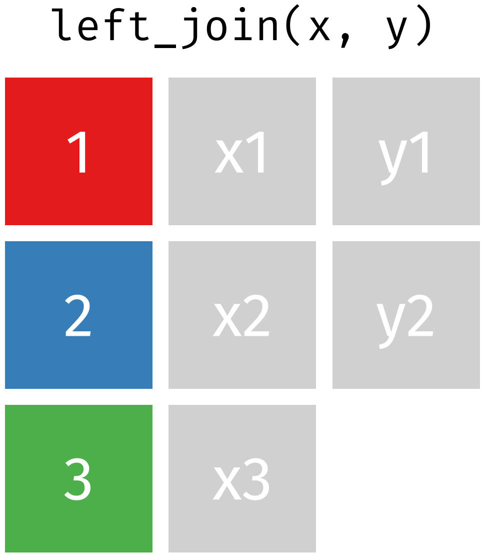
There are are many ways to do so

Inner join



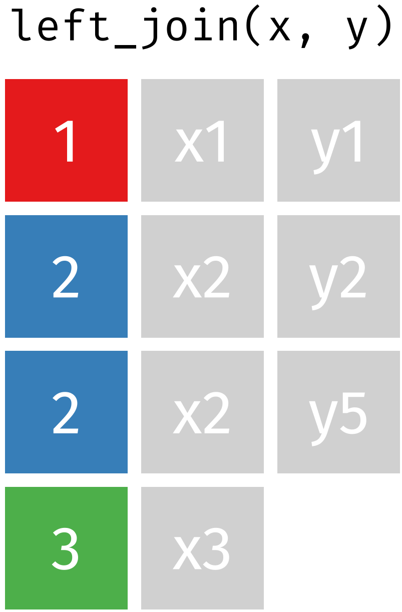
- All rows from `x` where there are matching values in `y`, and all columns from `x` and `y`.

Left join

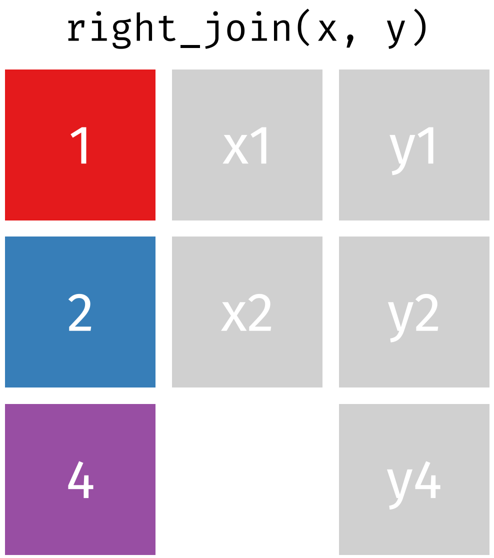


- All rows from x, and all columns from x and y. Rows in x with no match in y will have NA values in the new columns.

- If there are multiple matches between x and y, all combinations of the matches are returned.

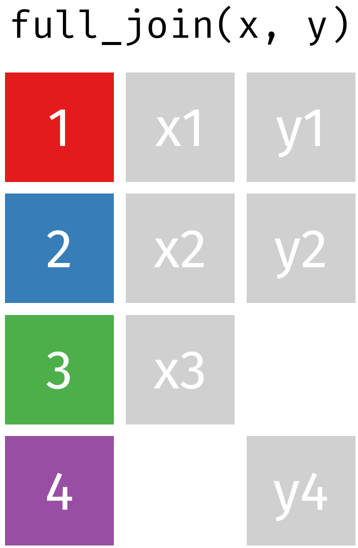


Right join



- All rows from y, and all columns from x and y. Rows in y with no match in x will have NA values in the new columns.

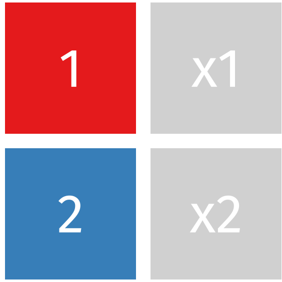
Full join



- All rows and all columns from both x and y. Where there are not matching values, returns NA for the one missing.

Semi join

`semi_join(x, y)`



- All rows from `x` where there are matching values in `y`, keeping just columns from `x`.

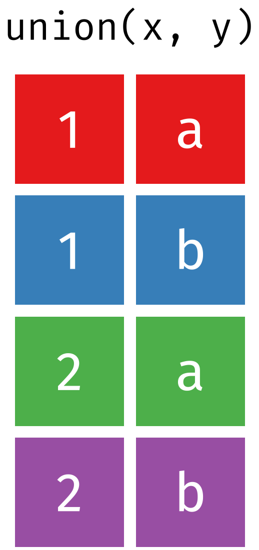
Anti join

```
anti_join(x, y)
```

- All rows from `x` where there are not matching values in `y`, keeping just columns from `x`.

Set Operations with Relational Data

Union



- All unique rows from x and y.

Union all

```
union_all(x, y)
```

1	a
1	b
2	a
1	a
2	b

- All rows from x and y, keeping duplicates.

GIF credit: Garrick Aden-Buie

Intersection

```
intersect(x, y)
```

- Common rows in both x and y, keeping just unique rows.

Set difference

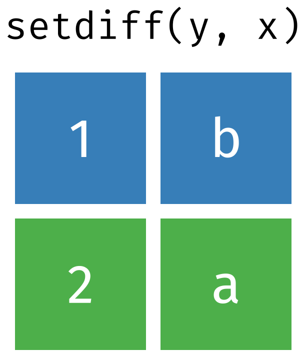
```
setdiff(x, y)
```

1	b
2	a

- All rows from x which are not also rows in y, keeping just unique rows.

GIF credit: Garrick Aden-Buie

Set difference reversed



- All rows from y which are not also rows in x, keeping just unique rows.

Joining by multiple columns

```r
x
```

```
## # A tibble: 3 x 3
##    year site  value
##   <dbl> <chr> <dbl>
## 1  2010 A         1
## 2  2010 B         3
## 3  2011 B         2
```

```r
y
```

```
## # A tibble: 2 x 4
##    year loc   value  resp
##   <dbl> <chr> <dbl> <dbl>
## 1  2010 A         1   5.4
## 2  2010 B         4   3
```

```r
x %>%
  left_join(y,
            by = c("year", "site" = "loc"),
            suffix = c("_x", "_y"))
```

```
## # A tibble: 3 x 5
##    year site  value_x value_y  resp
##   <dbl> <chr>   <dbl>   <dbl> <dbl>
## 1  2010 A           1       1   5.4
## 2  2010 B           3       4   3
## 3  2011 B           2      NA  NA
```

Nesting data with `tidyr`

Nest data

```r
df %>% as_tibble()

## # A tibble: 810 x 5
##       year variety height period temper
##      <int> <fct>    <dbl> <chr>
##  1   1974 G01        81    T1
##  2   1974 G01        81    T2
##  3   1974 G01        81    T3
##  4   1974 G01        81    T4
##  5   1974 G01        81    T5
##  6   1974 G01        81    T6
##  7   1975 G01        67.3 T1
##  8   1975 G01        67.3 T2
##  9   1975 G01        67.3 T3
## 10   1975 G01        67.3 T4
## # … with 800 more rows
```

```r
df %>%
  nest(weather = period:temperature)

# # A tibble: 135 x 4
#       year variety height weather
#      <int> <fct>    <dbl> <list>
#  1   1974 G01        81    <tibble [6 ×
#  2   1975 G01        67.3 <tibble [6 ×
#  3   1976 G01        71.5 <tibble [6 ×
#  4   1977 G01        64.3 <tibble [6 ×
#  5   1978 G01        55.8 <tibble [6 ×
#  6   1979 G01        84.9 <tibble [6 ×
#  7   1980 G01        86.2 <tibble [6 ×
#  8   1981 G01        88    <tibble [6 ×
#  9   1982 G01        72    <tibble [6 ×
# 10   1974 G02        72.3 <tibble [6 ×
```

Using nested data: `rowwise`

```
df %>%
  nest(weather = period:temperature) %>%
  rowwise() %>%
  mutate(avg_temp = mean(weather$temperature))
```

```
# # A tibble: 135 x 5
# # Rowwise:
#      year variety height weather          avg_temp
#     <int> <fct>    <dbl> <list>              <dbl>
#   1  1974 G01       81   <tibble [6 × 2]>     12.4
#   2  1975 G01       67.3 <tibble [6 × 2]>     13.4
#   3  1976 G01       71.5 <tibble [6 × 2]>     14.6
#   4  1977 G01       64.3 <tibble [6 × 2]>     14.7
#   5  1978 G01       55.8 <tibble [6 × 2]>     13.8
#   6  1979 G01       84.9 <tibble [6 × 2]>     13.8
#   7  1980 G01       86.2 <tibble [6 × 2]>     14.5
```

Using nested data: group_by

- `rowwise` is different to using `group_by` even if `group_by` refers to each row

```
df %>%
  nest(weather = period:temperature) %>%
  group_by(year, variety) %>%
  mutate(avg_temp = mean(weather[[1]]$temperature))

# # A tibble: 135 x 5
# # Groups:   year, variety [135]
#      year variety height weather          avg_temp
#     <int> <fct>    <dbl> <list>              <dbl>
#  1  1974 G01       81   <tibble [6 × 2]>     12.4
#  2  1975 G01       67.3 <tibble [6 × 2]>     13.4
#  3  1976 G01       71.5 <tibble [6 × 2]>     14.6
#  4  1977 G01       64.3 <tibble [6 × 2]>     14.7
#  5  1978 G01       55.8 <tibble [6 × 2]>     13.8
```

Tidy model output with `broom`

Model outputs are generally messy

```
fit <- lm(speed ~ dist, data = cars)
summary(fit)

##
## Call:
## lm(formula = speed ~ dist, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5293 -2.1550  0.3615  2.4377  6.4179
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.28391    0.87438   9.474 1.44e-12 ***
## dist         0.16557    0.01749   9.464 1.49e-12 ***
## ---
```

broom::tidy and broom::glance

```
fit <- lm(speed ~ dist, data = cars)
tidy(fit)

## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    8.28     0.874       9.47 1.44e-12
## 2 dist           0.166    0.0175      9.46 1.49e-12

glance(fit)

## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.651         0.644  3.16      89.6 1.49e-12     1  -127.  261.  267.
## # … with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

broom::augment

```
fit %>%
  augment()
```

```
## # A tibble: 50 x 8
##     speed  dist .fitted  .resid .std.resid   .hat .sigma    .cooksd
##     <dbl> <dbl>   <dbl>   <dbl>      <dbl>  <dbl>  <dbl>      <dbl>
##  1      4     2    8.62 -4.62      -1.52   0.0716   3.11 0.0888
##  2      4    10    9.94 -5.94      -1.93   0.0534   3.06 0.106
##  3      7     4    8.95 -1.95      -0.638  0.0667   3.18 0.0146
##  4      7    22   11.9  -4.93      -1.59   0.0335   3.10 0.0437
##  5      8    16   10.9  -2.93      -0.950  0.0424   3.16 0.0200
##  6      9    10    9.94 -0.940     -0.306  0.0534   3.19 0.00264
##  7     10    18   11.3  -1.26      -0.409  0.0392   3.18 0.00340
##  8     10    26   12.6  -2.59      -0.832  0.0289   3.17 0.0103
##  9     10    34   13.9  -3.91      -1.25   0.0225   3.14 0.0181
## 10     11    17   11.1  -0.0986    -0.0319 0.0407   3.19 0.0000216
```

Dealing with non-syntatic names with `janitor`

Non-syntatic variable names

Syntatic names consist of letters, digits, . and *only and begin with letters or . _and* also cannot be in reserved words list (`?Reserved`)

- E.g. `"loc@nsw"`, `"Frog ID"` and `"_name"` are non-syntatic names
- E.g. `"nsw_yield"`, `"var1"` and `".valid"` are syntatic names
- Non-syntatic names must be referred with a backtick

```
tmp <- tibble(`=non-syntatic` = 1:2)
tmp$`=non-syntatic`

## [1] 1 2

`1` <- 2
`1`

## [1] 2
```

janitor::clean_names

- It's usually easier to transform names to syntatic names rather than constantly referring to them by using backticks
- But rename one at a time is a pain

```
ns
```

```
## # A tibble: 3 x 3
##    `Frog id` `weight (kg)` `1980`
##        <int>         <dbl>  <dbl>
## 1          1            40    4.3
## 2          2            23    3
## 3          3             4    1.5
```

- The `clean_names` function in `janitor` is super handy

```
ns %>%
  clean_names()
```

```
## # A tibble: 3 x 3
##   frog_id weight_kg x1980
##     <int>     <dbl> <dbl>
## 1       1        40   4.3
## 2       2        23   3
## 3       3         4   1.5
```

- I use this A LOT

Adorn tables with `janitor`

Tables

- You can make nicer looking tables for publication using the `adorn_*` functions in `janitor`

```
count
```

```
## # A tibble: 3 x 3
##   site  `1980` `1981`
##   <chr>  <dbl>  <dbl>
## 1 A         40     30
## 2 B         20     40
## 3 C         10     10
```

```
count %>%
  adorn_totals(c("row", "col")) %>%
  adorn_percentages() %>%
  adorn_pct_formatting(digits = 0) %>%
  adorn_ns("front")
```

site	1980	1981	Total
A	40 (57%)	30 (43%)	70 (100%)
B	20 (33%)	40 (67%)	60 (100%)
C	10 (50%)	10 (50%)	20 (100%)
Total	70 (47%)	80 (53%)	150 (100%)

</> If you installed the `dwexercise` package, run below in your R console

```
learnr::run_tutorial("day2-exercise-01", package = "dwexercise")
```

🔗 If the above doesn't work for you, go here.

? Questions or issues, let us know!

15:00

Session Information

```
devtools::session_info()

## ─ Session info ───────────────────────────────────────────────
##  setting  value
##  version  R version 4.0.1 (2020-06-06)
##  os       macOS Catalina 10.15.7
##  system   x86_64, darwin17.0
##  ui       X11
##  language (EN)
##  collate  en_AU.UTF-8
##  ctype    en_AU.UTF-8
##  tz       Australia/Melbourne
##  date     2020-12-01
##
## ─ Packages ───────────────────────────────────────────────────
##  package     * version    date       lib source
##  agridat     * 1.17       2020-08-03 [1] CRAN (R 4.0.2)
##  anicon        0.1.0      2020-06-21 [1] Github (emitanaka/anicon@0b756df)
```