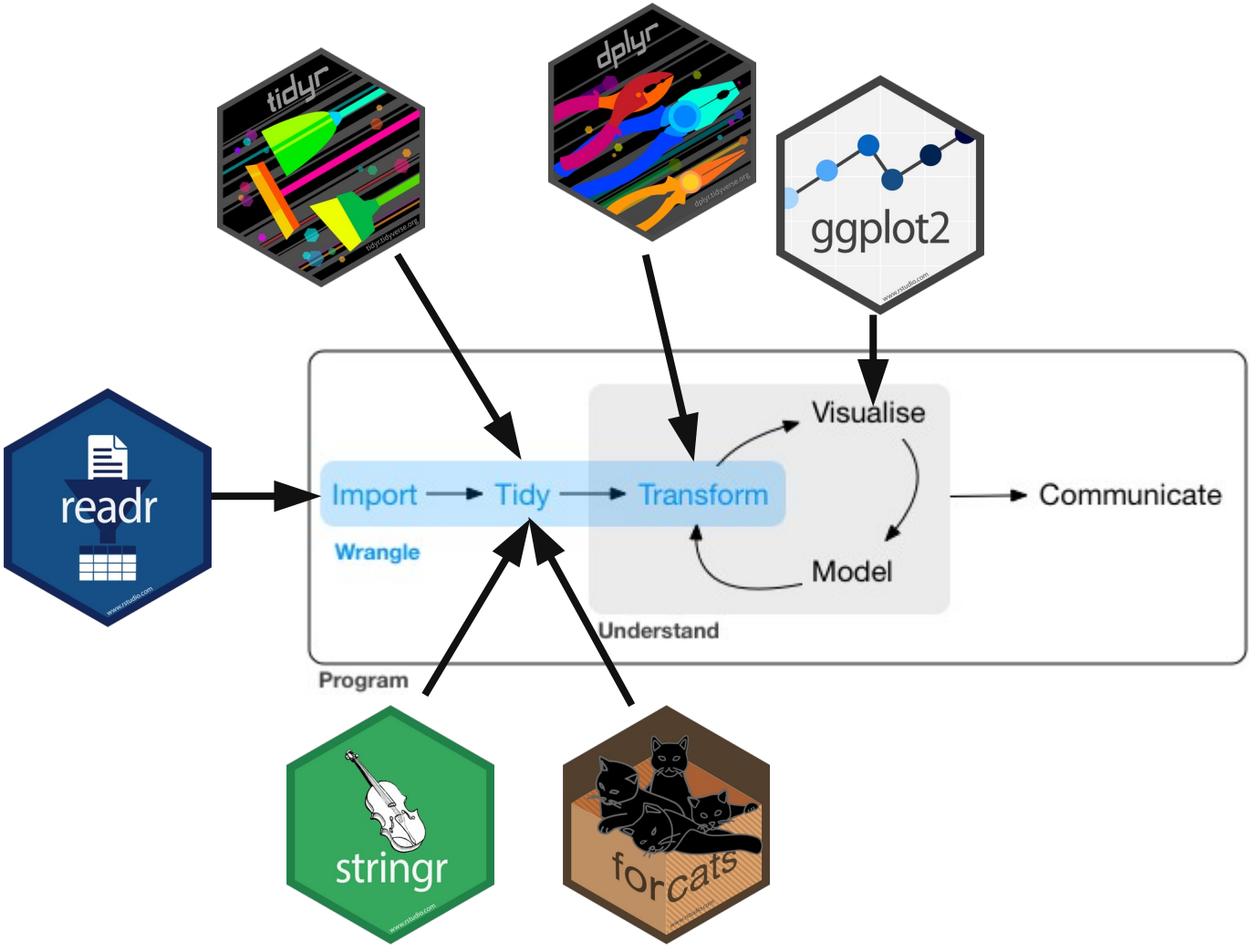
GG606

(More) Tidying data, data forms & formats

Homework

- Combine `nycflights13::flights` & `nycflights13::weather`
  - Identify primary keys
  - Put R-Script and figure in discord
- Work through example to tidy the entire `tidyr::who` dataset in Chapter 12
  - https://r4ds.had.co.nz/tidy-data.html
  - Put a plot with at least 6 countries in discord

Homework


tidyr
dplyr
ggplot2
readr
Import
Tidy
Transform
Visualise
Model
Communicate
Wrangle
Understand
Program
stringr
forcats

Relational Data

- Mutating joins: new variables to one data frame from matching obs in another
- Filtering joins: filter observations from one data frame based on whether or not they match an obs in another
- Set operations, treat obs as if they were set elements

nycfligts13

- Compare airlines, airports, planes, weather

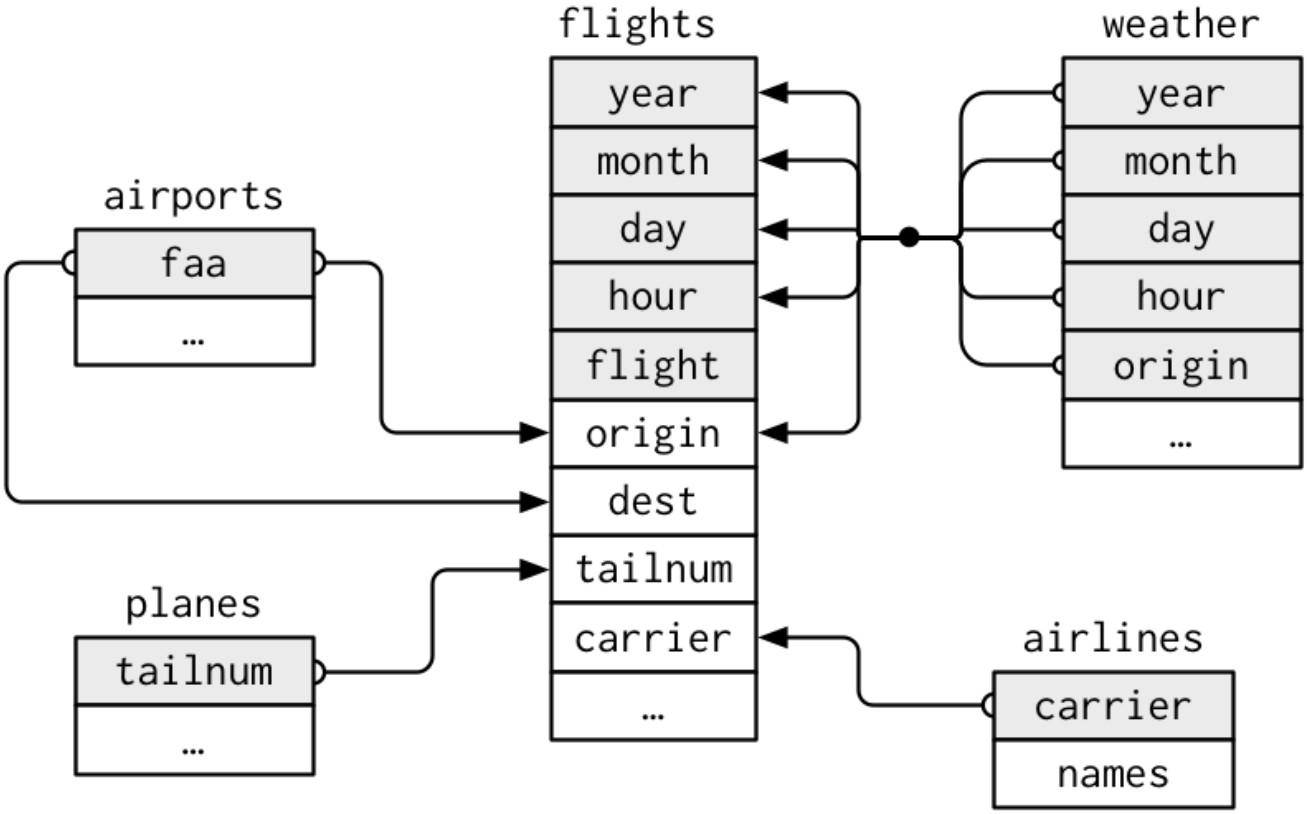
```
airlines
#> # A tibble: 16 x 2
#>   carrier name
#>   <chr>   <chr>
#> 1 9E      Endeavor Air Inc.
#> 2 AA      American Airlines Inc.
#> 3 AS      Alaska Airlines Inc.
#> 4 B6      JetBlue Airways
#> 5 DL      Delta Air Lines Inc.
#> 6 EV      ExpressJet Airlines Inc.
#> # … with 10 more rows
```

```
airports
#> # A tibble: 1,458 x 8
#>   faa   name                          lat   lon   alt
#>   <chr> <chr>                       <dbl> <dbl> <dbl>
#> 1 04G   Lansdowne Airport            41.1 -80.6  1044
#> 2 06A   Moton Field Municipal Airp…  32.5 -85.7   264
#> 3 06C   Schaumburg Regional          42.0 -88.1   801
#> 4 06N   Randall Airport              41.4 -74.4   523
#> 5 09J   Jekyll Isl
#> 6 0A9   Elizabetht
#> # … with 1,452 mor
```

```
planes
#> # A tibble: 3,322 x 9
#>   tailnum  year type           manufacturer   model
#>   <chr>   <int> <chr>          <chr>          <chr>
#> 1 N10156   2004 Fixed wing mu… EMBRAER        EMB-1…
#> 2 N102UW   1998 Fixed wing mu… AIRBUS INDUST… A320-…
#> 3 N103US   1999 Fixed wing mu… AIRBUS INDUST… A320-…
#> 4 N104UW   1999 Fixed wing mu… AIRBUS INDUST… A320-…
#> 5 N10575   2002 Fixed wing mu… EMBRAER        EMB-1…
#> 6 N105UW   1999 Fixed wing mu… AIRBUS INDUST… A320-…
#> # … with 3,316 more rows
```

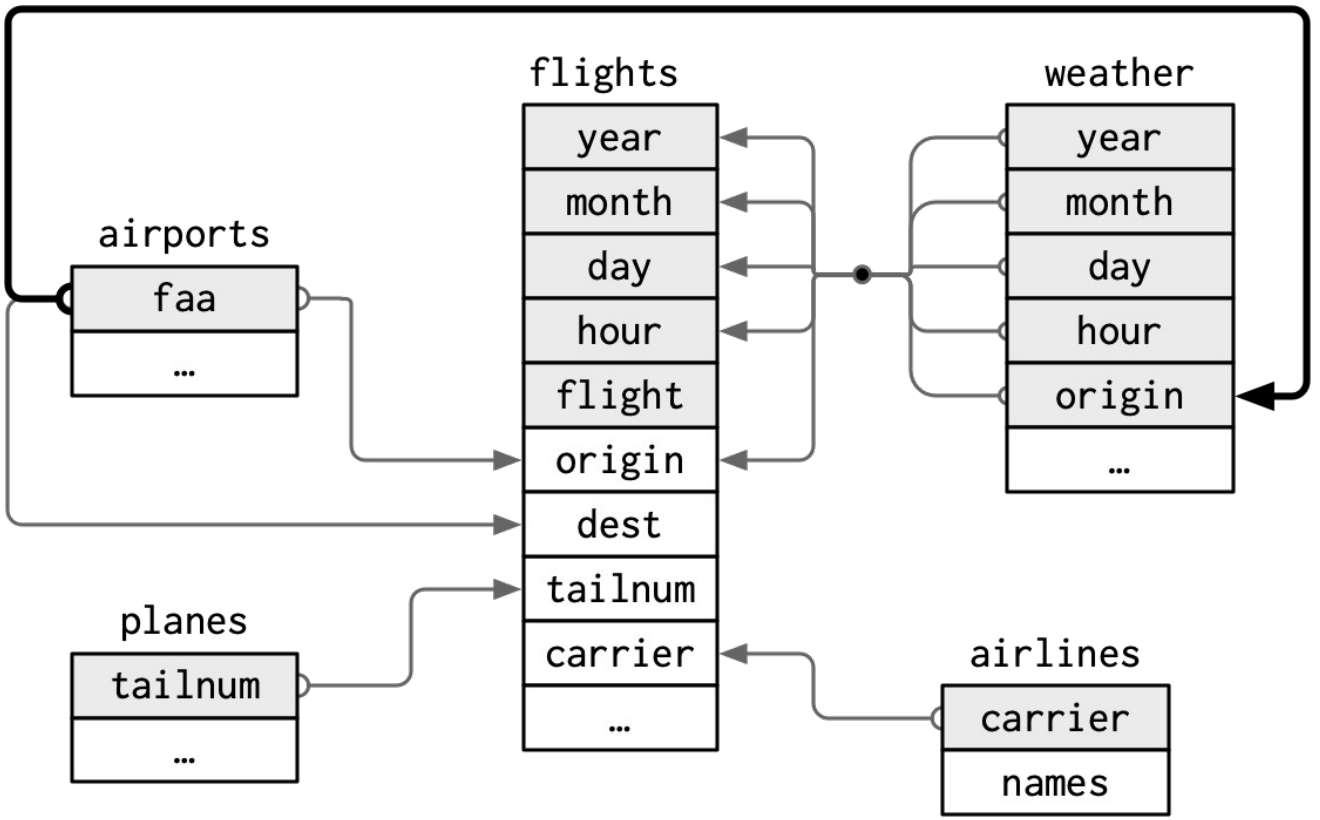
```
weather
#> # A tibble: 26,115 x 15
#>   origin  year month   day  hour  temp
#>   <chr>  <int> <int> <int> <int> <dbl>
#> 1 EWR     2013     1     1     1  39.0
#> 2 EWR     2013     1     1     2  39.0
#> 3 EWR     2013     1     1     3  39.0
#> 4 EWR     2013     1     1     4  39.9
#> 5 EWR     2013     1     1     5  39.0
#> 6 EWR     2013     1     1     6  37.9
#> # … with 26,109 more rows, and 4 more
#> #   visib <dbl>, time_hour <dttm>
```

nycfligts13



walk through; airports to weather?

nycfligts13



walk through; airports to weather?

Keys

- Primary: uniquely identifies obs in own table
- Foreign: uniquely identifies abs in other table

Keys

- Primary: uniquely identifies obs in own table
- Foreign: uniquely identifies abs in other table
- ```
  planes %>% count(tailnum) %>%
    filter(n > 1)
  ```
- ```
  flights %>% count(tailnum) %>%
    filter(n > 1)
  ```

Keys

- Primary: uniquely identifies obs in own table
- Foreign: uniquely identifies obs in other table
- Surrogate: added primary key, like row number

Keys

- Surrogate: added primary key, like row number
- ```
  flights %>%
    count(year, month, day, flight) %>%
    filter(n > 1)
  ```
- ```
  flights %>%
    count(year, month, day, tailnum) %>%
    filter(n > 1)
  ```

Add Surrogate Key

- Add `row_number()` as column `flight_id`
- What verb to add new column?

Add Surrogate Key

- Add `row_number()` as column `flight_id`
- ```
  flights %>%
    arrange(year, month, day,
  sched_dep_time, carrier, flight) %>%
    mutate(flight_id = row_number()) %>%
    glimpse()
  ```
  ```

Mutate Join

- Add airline carrier name to `flights`
- `flights2 ← flights %>% select(year:day, hour, origin, dest, tailnum, carrier)`
-

Mutate Join

- Add airline carrier name to `flights`
-
```
flights2 ← flights %>%
  select(year:day, hour, origin, dest,
tailnum, carrier)
```
-
```
flights2 %>%
  select(-origin, -dest) %>%
  left_join(airlines, by = "carrier")
```

`?left_join()`

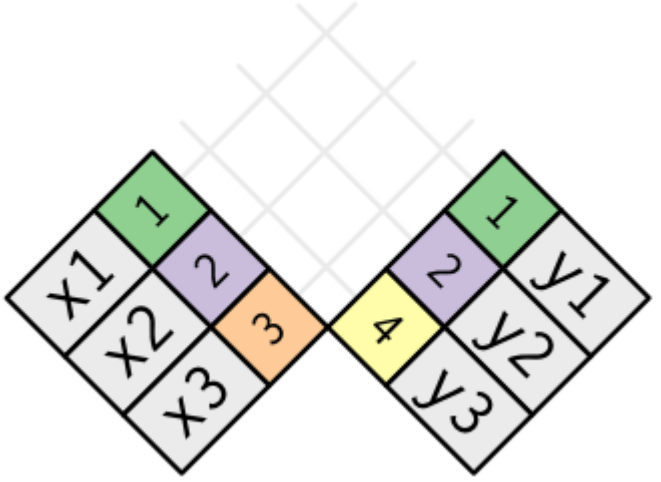
?mutate-joins

x

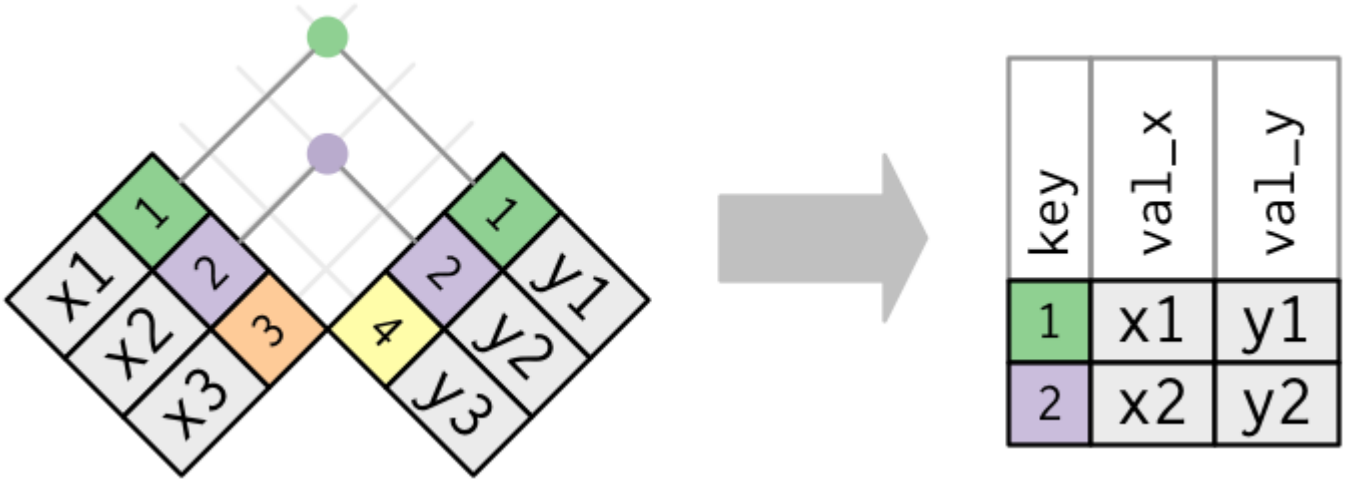
1	x1
2	x2
3	x3

y

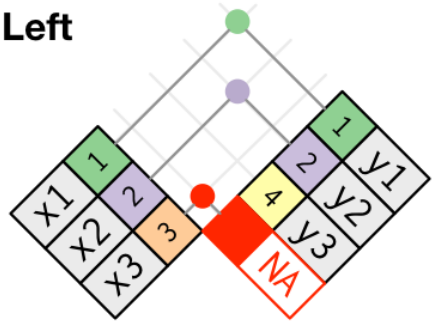
1	y1
2	y2
4	y3



?inner_join

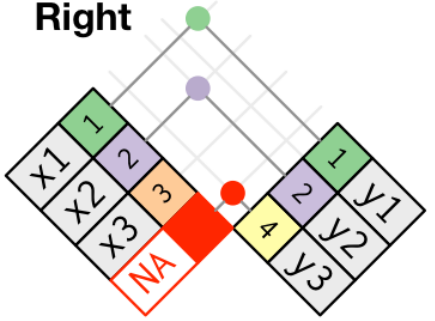


Left



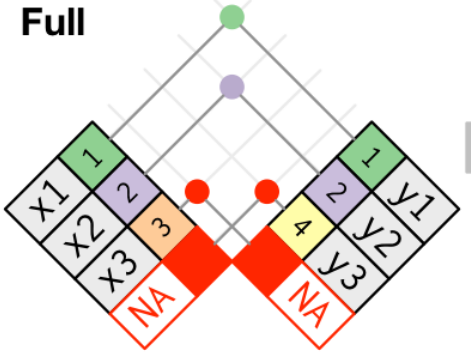
key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA

Right



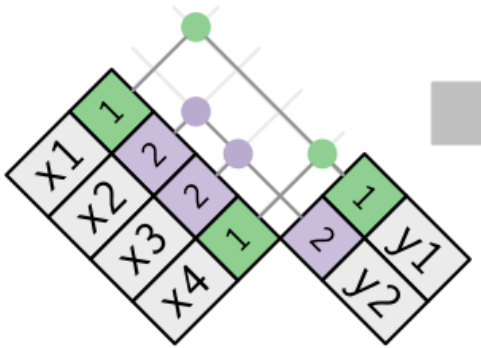
key	val_x	val_y
1	x1	y1
2	x2	y2
4	NA	y3

Full

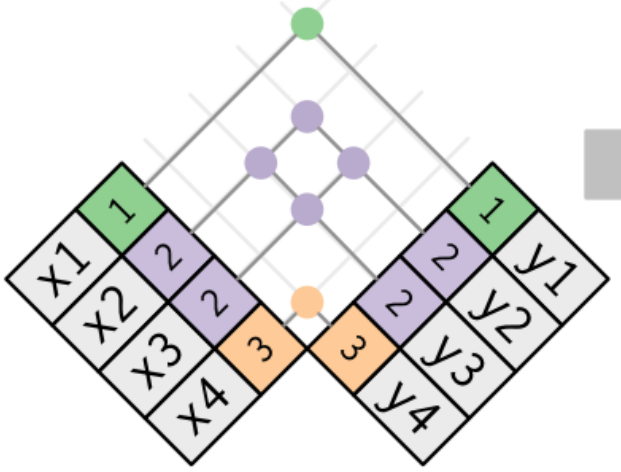


key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA
4	NA	y3

Duplicates



val_x	key	val_y
x1	1	y1
x2	2	y2
x3	2	y2
x4	1	y1



key	val_x	val_y
1	x1	y1
2	x2	y2
2	x2	y3
2	x3	y2
2	x3	y3
3	x4	y4

Might indicate an error/problem

Join by

- Compare
- `flights2 %>% left_join(weather)`
- `flights2 %>% left_join(planes, by = "tailnum")`

```
flights2 %>%
  left_join(weather)
#> Joining, by = c("year", "month", "day", "hour", "origin")
#> # A tibble: 336,776 x 18
#>    year month   day  hour origin dest  tailnum carrier  temp
#>   <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>   <dbl>
#> 1  2013     1     1     5 EWR    IAH   N14228  UA       39.0
#> 2  2013     1     1     5 LGA    IAH   N24211  UA       39.9
```

```
flights2 %>%
  left_join(planes, by = "tailnum")
#> # A tibble: 336,776 x 16
#>   year.x month   day  hour origin dest  tailnum carrier year.y
#>    <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>    <int>
#> 1   2013     1     1     5 EWR    IAH   N14228  UA        1999
#> 2   2013     1     1     5 LGA    IAH   N24211  UA        1998
```

SQL is the inspiration for dplyr's conventions, so the translation is straightforward:

dplyr	SQL
`inner_join(x, y, by = "z")`	`SELECT * FROM x INNER JOIN y USING (z)`
`left_join(x, y, by = "z")`	`SELECT * FROM x LEFT OUTER JOIN y USING (z)`
`right_join(x, y, by = "z")`	`SELECT * FROM x RIGHT OUTER JOIN y USING (z)`
`full_join(x, y, by = "z")`	`SELECT * FROM x FULL OUTER JOIN y USING (z)`

Note that "INNER" and "OUTER" are optional, and often omitted.

Filter Join

- Works on obs not on variables
- `semi_join()` return all rows from x with a match in y
- `anti_join()` return all rows from x without a match in y

Filter Join (vs Filter)

- Get top 10 destinations
- `top_dest ← flights %>% count(dest, sort = TRUE) %>% head(10)`
- `dest` columns is key

```
top_dest <- flights %>%
  count(dest, sort = TRUE) %>%
  head(10)
top_dest
#> # A tibble: 10 x 2
#>   dest      n
#>   <chr> <int>
#> 1 ORD   17283
#> 2 ATL   17215
#> 3 LAX   16174
#> 4 BOS   15508
#> 5 MCO   14082
#> 6 CLT   14064
#> # … with 4 more rows
```

Filter Join (vs Filter)

- Two ways:
- ```
  flights %>%
    filter(dest %in% top_dest$dest)
  ```
- ```
  flights %>%
    semi_join(top_dest)
  ```


stringr
www.rstudio.com


forcats
www.rstudio.com

Strings

- Covers basics
- regexps are powerful
- Tips, gotchas
- Assembling labels
- Things that update/results

Strings

- "This is a string" vs “This is a string”
- Generally use ""
- `string1 ← "This is a string"`
- `string2 ← 'If I want to include a "quote" inside a string, I use single quotes'`

```
> string1
[1] "This is a string"
> string2
[1] "If I want to include a \"quote\" inside
 a string, I use single quotes"
```

Special

- \ is special
- ‘escape’ some characters like \' and \"
- \n (newline) and \t (tab) useful

```
Single quotes need to be escaped by backslash in single-quoted
strings, and double quotes in double-quoted strings.

  ‘\n’          newline
  ‘\r’          carriage return
  ‘\t’          tab
  ‘\b’          backspace
  ‘\a’          alert (bell)
  ‘\f’          form feed
  ‘\v’          vertical tab
  ‘\\’          backslash ‘\’
  ‘\'’          ASCII apostrophe ‘'’
  ‘\"’          ASCII quotation mark ‘"’
  ‘\`’          ASCII grave accent (backtick) ‘`’
  ‘\nnn’        character with given octal code (1, 2 or 3 digits)
  ‘\xnn’        character with given hex code (1 or 2 hex digits)
  ‘\unnnn’      Unicode character with given code (1-4 hex digits)
  ‘\Unnnnnnnn’  Unicode character with given code (1-8 hex digits)
```

?Quotes

`Stringr` functions

- Most begin with `str_`
- (Many base R string functions are fragile)
- `str_c("x", "y", "z")`
  `c("x", "y", "z")`

`Stringr` functions

- `x ← c("Apple", "Banana", "Pear")`
- `str_sub(x, 1, 3)`
- `str_sub(x, -3, -1)`
- Similar to left() and right() in Excel?

Stringr functions

- `x ← c("Apple", "Banana", "Pear")`
- `str_to_lower(x)`
- `str_to_sentence(x)`
- `str_to_title(x)`
- `str_to_upper(x)`

Stringr locales

- `str_to_upper(c("i", "ı"))`
- `str_to_upper(c("i", "ı"), locale = "tr")`

```
# Turkish has two i's: with and without a dot, and it
# has a different rule for capitalising them:
str_to_upper(c("i", "ı"))
#> [1] "I" "I"
str_to_upper(c("i", "ı"), locale = "tr")
#> [1] "İ" "I"
```

Stringr locales

- `y ← c("apple", "eggplant", "banana")`
- `str_sort(y, locale = "en")`
- `str_sort(y, locale = "haw")`

```
str_sort(x, locale = "en")  # English
#> [1] "apple"    "banana"   "eggplant"

str_sort(x, locale = "haw") # Hawaiian
#> [1] "apple"    "eggplant" "banana"
```

Strings

- Frustrating but powerful
- Good strings make good factors
- Excellent resource: https://r4ds.had.co.nz/strings.html