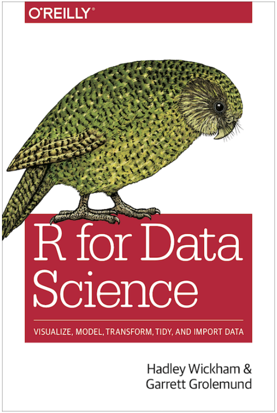
Socio-Informatics 348

Data Transformation
Joins

Dr Lisa Martin

Department of Information Science
Stellenbosch University

Today's Reading



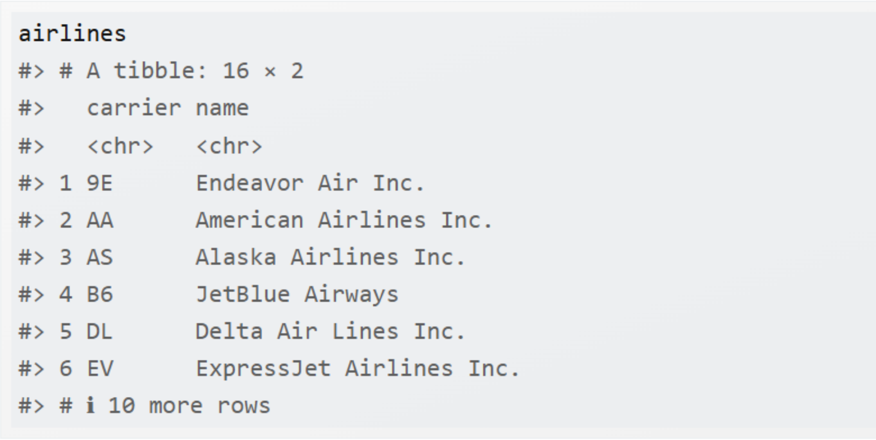
R for Data Science, Chapter 19

Introduction

- Analyses usually involve multiple data frames.
- Two main types of joins:
  - Mutating joins – add new variables to one data frame from matching observations in another.
  - Filtering joins – filter observations from one data frame based on whether or not they match an observation in another.

Keys: Primary & Foreign

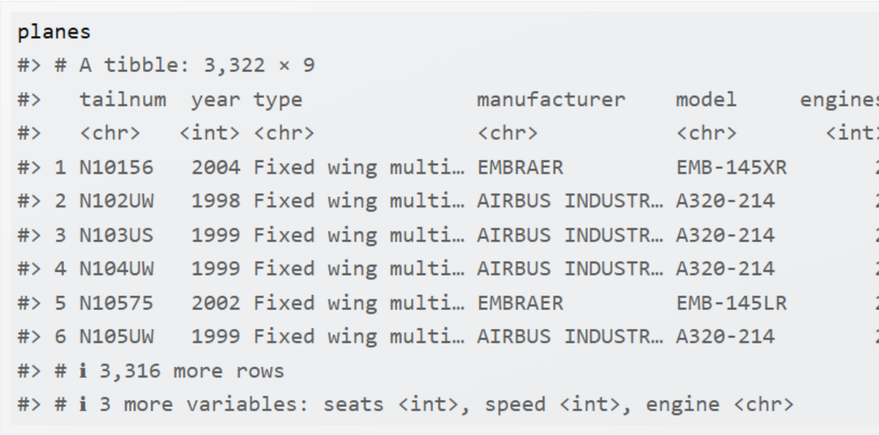
- Primary key: uniquely identifies observations in a table.
- Foreign key: matches a primary key in another table.



```
airlines
#> # A tibble: 16 × 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.
#> # ℹ 10 more rows
```

Keys: Primary & Foreign

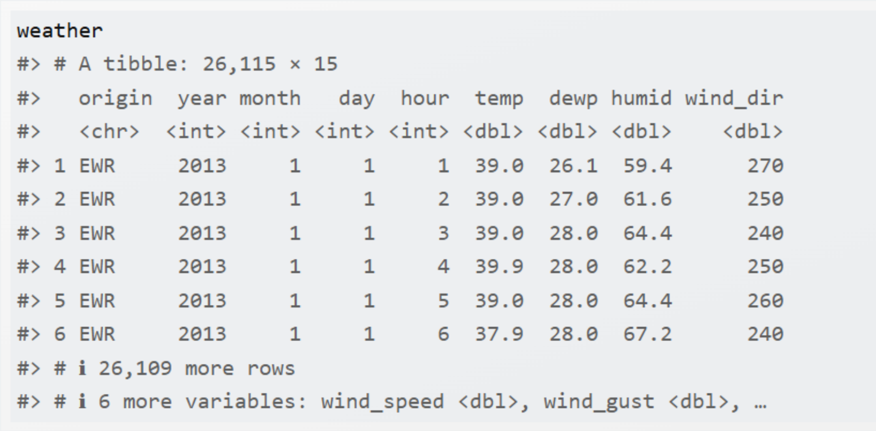
- Primary key: uniquely identifies observations in a table.
- Foreign key: matches a primary key in another table.



```
planes
#> # A tibble: 3,322 × 9
#>   tailnum  year type                    manufacturer     model     engines
#>   <chr>   <int> <chr>                   <chr>            <chr>       <int>
#> 1 N10156   2004 Fixed wing multi… EMBRAER          EMB-145XR       2
#> 2 N102UW   1998 Fixed wing multi… AIRBUS INDUSTR… A320-214        2
#> 3 N103US   1999 Fixed wing multi… AIRBUS INDUSTR… A320-214        2
#> 4 N104UW   1999 Fixed wing multi… AIRBUS INDUSTR… A320-214        2
#> 5 N10575   2002 Fixed wing multi… EMBRAER          EMB-145LR       2
#> 6 N105UW   1999 Fixed wing multi… AIRBUS INDUSTR… A320-214        2
#> # ℹ 3,316 more rows
#> # ℹ 3 more variables: seats <int>, speed <int>, engine <chr>
```

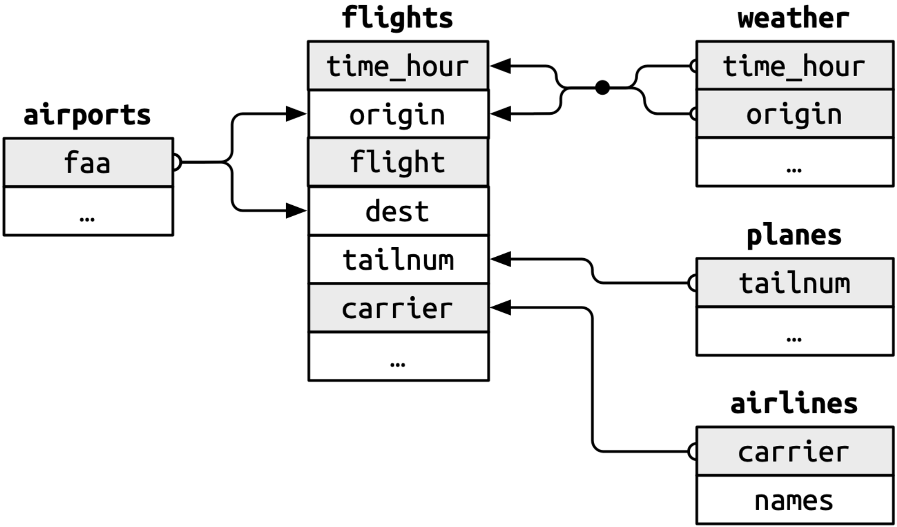
Keys: Primary & Foreign

- Composite keys: multiple variables together uniquely identify observations.



```
weather
#> # A tibble: 26,115 × 15
#>   origin  year month   day  hour  temp  dewp humid wind_dir
#>   <chr>  <int> <int> <int> <int> <dbl> <dbl> <dbl>    <dbl>
#> 1 EWR     2013     1     1     1  39.0  26.1  59.4      270
#> 2 EWR     2013     1     1     2  39.0  27.0  61.6      250
#> 3 EWR     2013     1     1     3  39.0  28.0  64.4      240
#> 4 EWR     2013     1     1     4  39.9  28.0  62.2      250
#> 5 EWR     2013     1     1     5  39.0  28.0  64.4      260
#> 6 EWR     2013     1     1     6  37.9  28.0  67.2      240
#> # ℹ 26,109 more rows
#> # ℹ 6 more variables: wind_speed <dbl>, wind_gust <dbl>, …
```

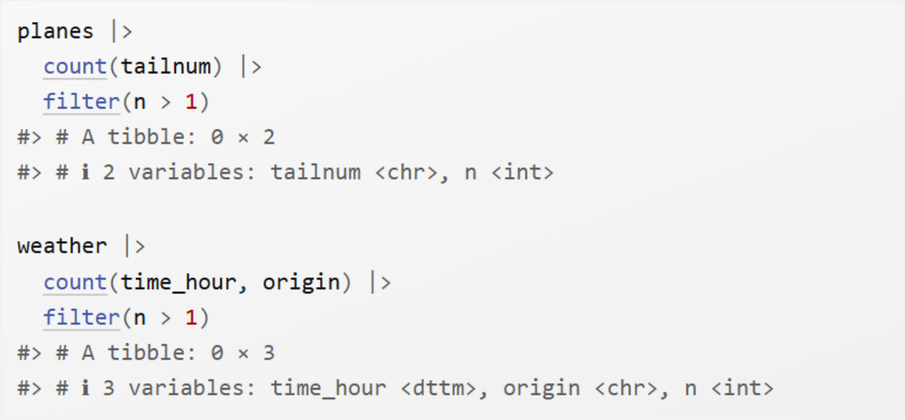
Keys: Primary & Foreign



Checking Keys

- Verify uniqueness of primary keys.
- Check for missing values – NAs can't identify observations.
- Use `count()` and `filter(n > 1)`.
- Surrogate keys: useful when no obvious key exists.

Checking Keys



```
planes |>
  count(tailnum) |>
  filter(n > 1)
#> # A tibble: 0 × 2
#> # ℹ 2 variables: tailnum <chr>, n <int>

weather |>
  count(time_hour, origin) |>
  filter(n > 1)
#> # A tibble: 0 × 3
#> # ℹ 3 variables: time_hour <dttm>, origin <chr>, n <int>
```

Surrogate Keys

- Useful when no obvious key exists.

Types of Joins

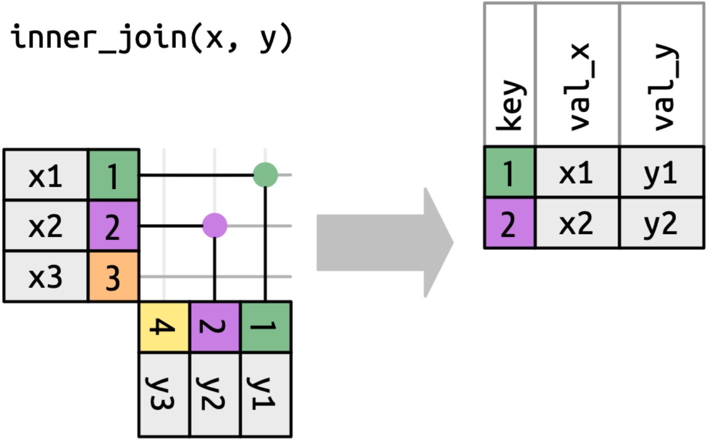
dplyr provides 6 join functions:

- left_join()
- inner_join()
- right_join()
- full_join()
- semi_join()
- anti_join()

Mutating Joins

- Add variables from one table to another.
- Four types: inner, left, right, full.
- `left_join()` as the most common example.
- Unmatched rows → `NA`.

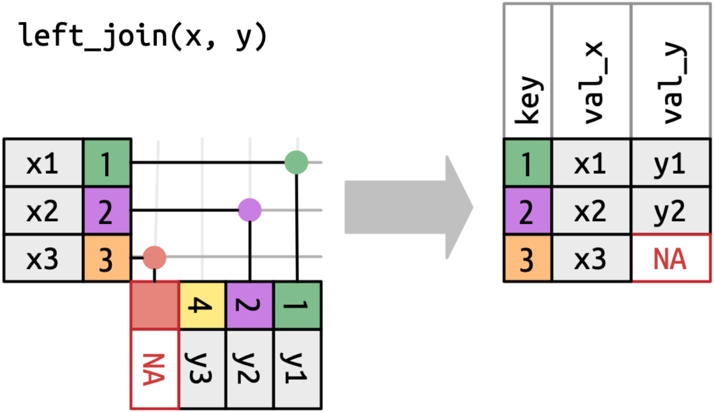
Mutating Joins



R for Data Science, Chapter 19, Figure 19.4

Mutating Joins

```
left_join(x, y)
```



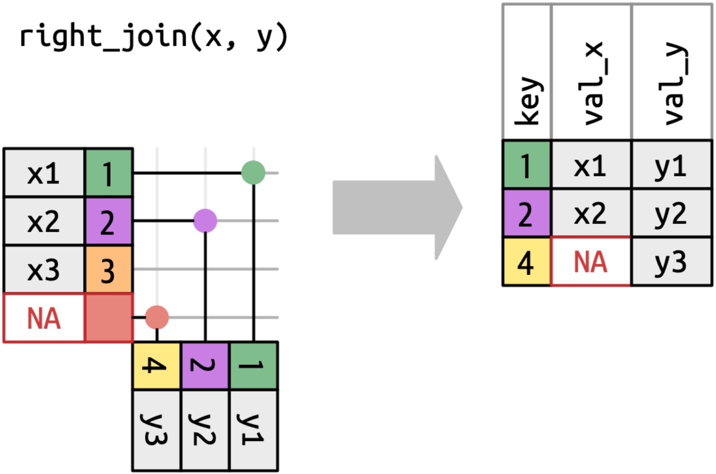
x1	1
x2	2
x3	3

	4	2	1
NA	y3	y2	y1

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

R for Data Science, Chapter 19, Figure 19.5

Mutating Joins

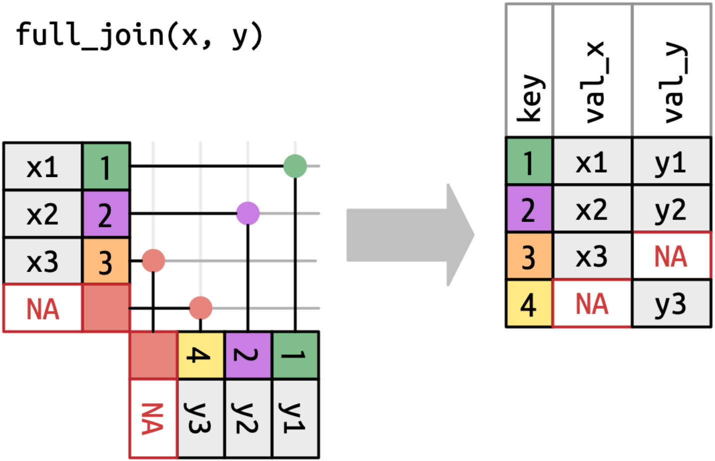


```
right_join(x, y)
```

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

R for Data Science, Chapter 19, Figure 19.6

Mutating Joins



```
full_join(x, y)
```

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

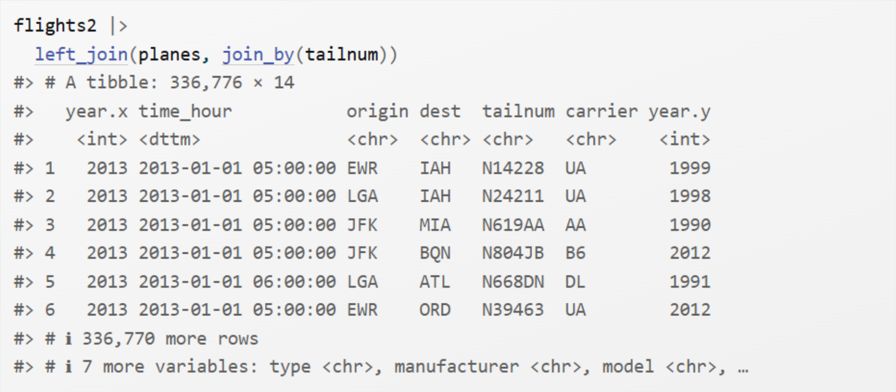
R for Data Science, Chapter 19, Figure 19.7

Specifying Join Keys

- Default: use variables with same name (so called natural join).
- Potential Problem: same name, different meaning.

Specifying Join Keys

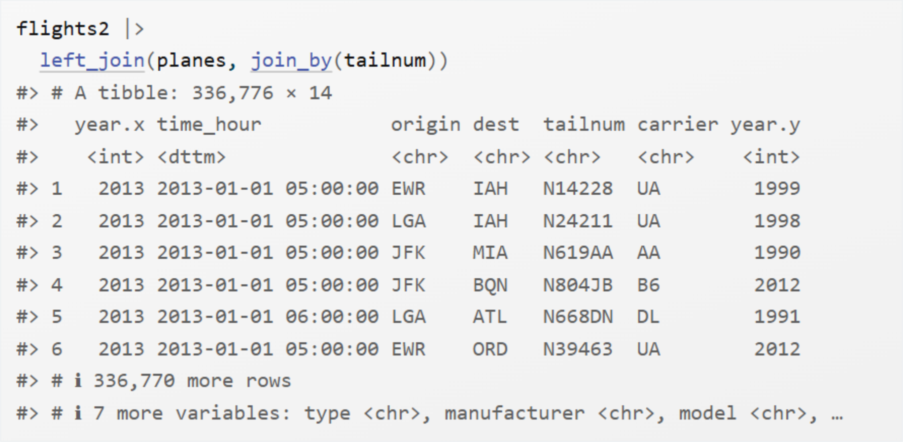
- Potential Problem: same name, different meaning.



```
flights2 |>
  left_join(planes, join_by(tailnum))
#> # A tibble: 336,776 × 14
#>   year.x time_hour           origin dest  tailnum carrier year.y
#>    <int> <dttm>              <chr>  <chr> <chr>   <chr>    <int>
#> 1   2013 2013-01-01 05:00:00 EWR    IAH   N14228  UA        1999
#> 2   2013 2013-01-01 05:00:00 LGA    IAH   N24211  UA        1998
#> 3   2013 2013-01-01 05:00:00 JFK    MIA   N619AA  AA        1990
#> 4   2013 2013-01-01 05:00:00 JFK    BQN   N804JB  B6        2012
#> 5   2013 2013-01-01 06:00:00 LGA    ATL   N668DN  DL        1991
#> 6   2013 2013-01-01 05:00:00 EWR    ORD   N39463  UA        2012
#> # ℹ 336,770 more rows
#> # ℹ 7 more variables: type <chr>, manufacturer <chr>, model <chr>, …
```

Specifying Join Keys

- Solution: `join_by()` to specify matches.
- `join_by(tailnum)` is the same as `join_by(tailnum == tailnum)`.
- Multiple matching variables: `join_by(a, b, c == d)`.
- Control suffixes for overlapping names.



```
flights2 |>
  left_join(planes, join_by(tailnum))
#> # A tibble: 336,776 × 14
#>   year.x time_hour           origin dest  tailnum carrier year.y
#>    <int> <dttm>              <chr>  <chr> <chr>   <chr>    <int>
#> 1   2013 2013-01-01 05:00:00 EWR    IAH   N14228  UA        1999
#> 2   2013 2013-01-01 05:00:00 LGA    IAH   N24211  UA        1998
#> 3   2013 2013-01-01 05:00:00 JFK    MIA   N619AA  AA        1990
#> 4   2013 2013-01-01 05:00:00 JFK    BQN   N804JB  B6        2012
#> 5   2013 2013-01-01 06:00:00 LGA    ATL   N668DN  DL        1991
#> 6   2013 2013-01-01 05:00:00 EWR    ORD   N39463  UA        2012
#> # ℹ 336,770 more rows
#> # ℹ 7 more variables: type <chr>, manufacturer <chr>, model <chr>, …
```

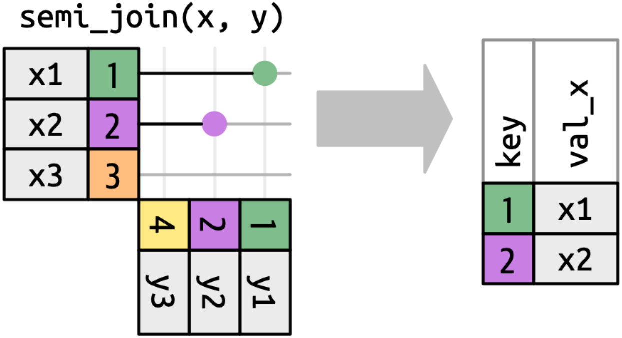
Multiple Matching keys?

- If row in `x` matches more than 1 row in `x` – duplicated once for each match.
- What if multiple rows in `x` match multiple rows in `y`?
- `Warning:  Detected an unexpected many-to-many relationship between ‘x’ and ‘y’.`
- If this is expected, use `relationship = "many-to-many"` to silence the warning.

Filtering Joins

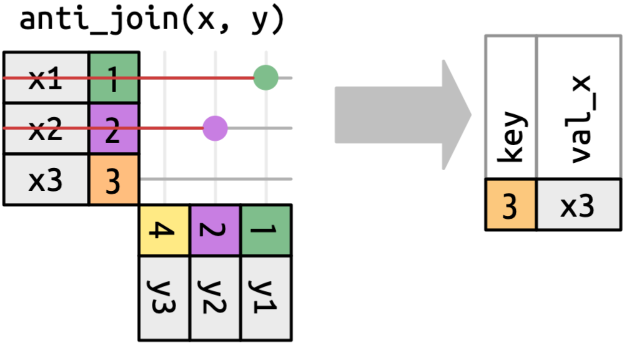
- Keep or drop rows depending on matches.
- `semi_join()`: keep rows in `x` if match exists in `y`.
- `anti_join()`: keep rows in `x` if no match exists in `y`.

Filtering Joins



R for Data Science, Chapter 19, Figure 19.10

Filtering Joins

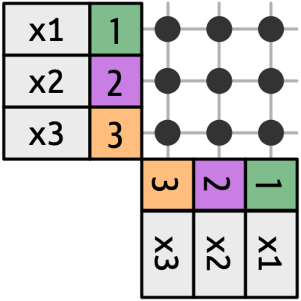


R for Data Science, Chapter 19, Figure 19.11

Non-equi Joins

- Previous joins have been equi-joins (exact matches).
- Sometimes you want to match on inequalities.
- Types:
  - Cross joins (Cartesian product)
  - Inequality joins
  - Rolling joins (closest match)
  - Overlap joins (range matching)

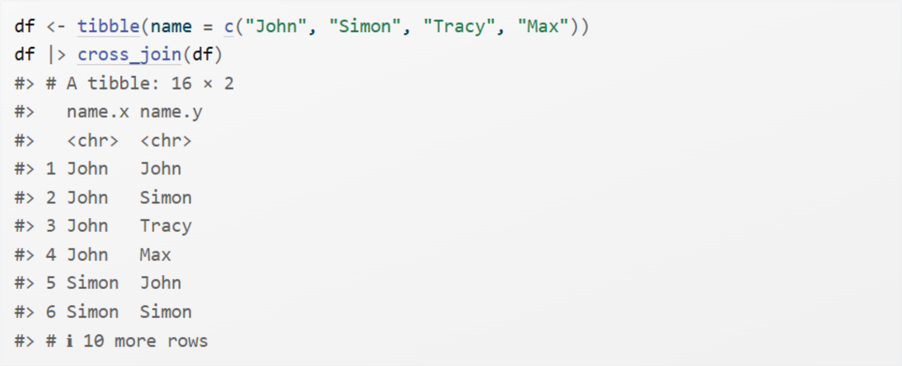
Cross Joins



R for Data Science, Chapter 19, Figure 19.14

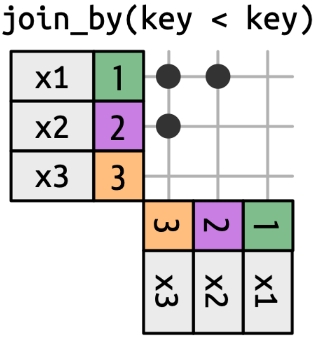
Cross Joins

Note function: `cross_join()`



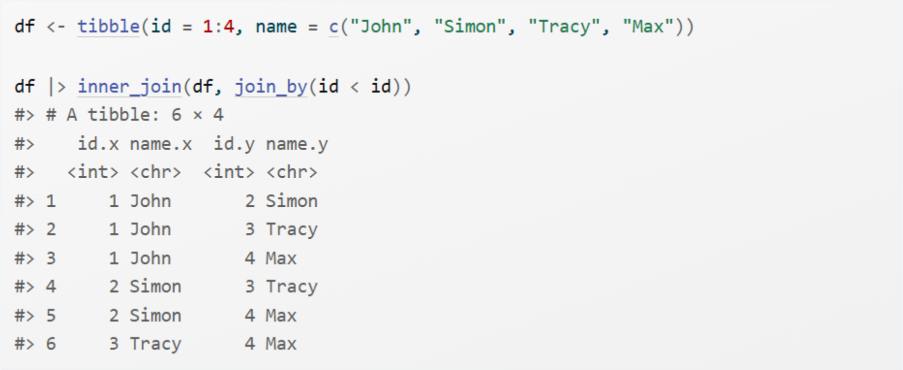
```
df <- tibble(name = c("John", "Simon", "Tracy", "Max"))
df |> cross_join(df)
#> # A tibble: 16 × 2
#>   name.x name.y
#>   <chr>  <chr>
#> 1 John   John
#> 2 John   Simon
#> 3 John   Tracy
#> 4 John   Max
#> 5 Simon  John
#> 6 Simon  Simon
#> # ℹ 10 more rows
```

Inequality Joins



R for Data Science, Chapter 19, Figure 19.15

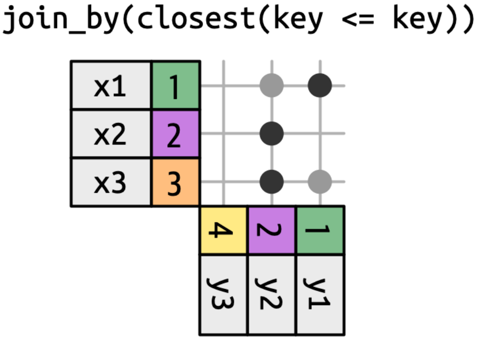
Inequality Joins



```
df <- tibble(id = 1:4, name = c("John", "Simon", "Tracy", "Max"))

df |> inner_join(df, join_by(id < id))
#> # A tibble: 6 × 4
#>    id.x name.x  id.y name.y
#>   <int> <chr>  <int> <chr>
#> 1     1 John       2 Simon
#> 2     1 John       3 Tracy
#> 3     1 John       4 Max
#> 4     2 Simon      3 Tracy
#> 5     2 Simon      4 Max
#> 6     3 Tracy      4 Max
```

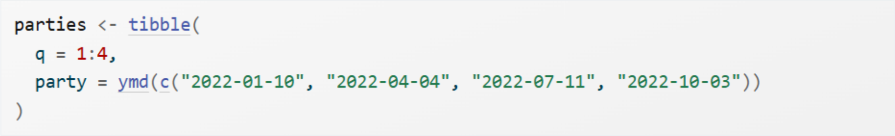
Rolling Joins



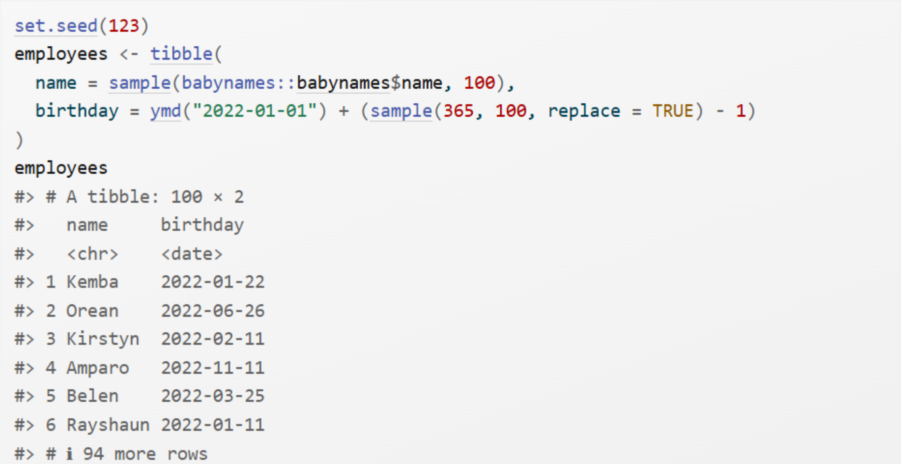
```
join_by(closest(key <= key))
```

R for Data Science, Chapter 19, Figure 19.16

Rolling Joins

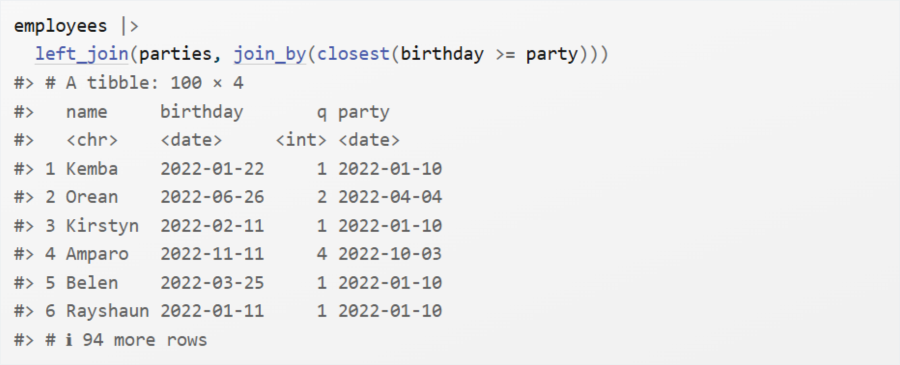


```
parties <- tibble(
  q = 1:4,
  party = ymd(c("2022-01-10", "2022-04-04", "2022-07-11", "2022-10-03"))
)
```



```
set.seed(123)
employees <- tibble(
  name = sample(babynames::babynames$name, 100),
  birthday = ymd("2022-01-01") + (sample(365, 100, replace = TRUE) - 1)
)
employees
#> # A tibble: 100 × 2
#>   name     birthday
#>   <chr>    <date>
#> 1 Kemba    2022-01-22
#> 2 Orean    2022-06-26
#> 3 Kirstyn  2022-02-11
#> 4 Amparo   2022-11-11
#> 5 Belen    2022-03-25
#> 6 Rayshaun 2022-01-11
#> # ℹ 94 more rows
```

Rolling Joins

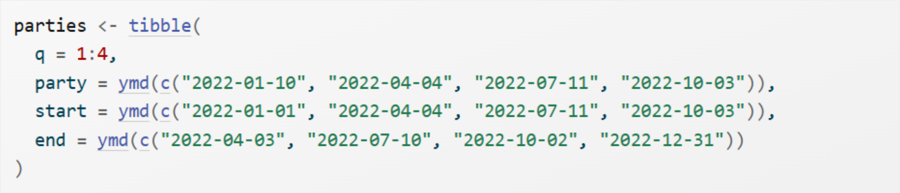


```
employees |>
  left_join(parties, join_by(closest(birthday >= party)))
#> # A tibble: 100 × 4
#>   name     birthday       q party
#>   <chr>    <date>     <int> <date>
#> 1 Kemba    2022-01-22     1 2022-01-10
#> 2 Orean    2022-06-26     2 2022-04-04
#> 3 Kirstyn  2022-02-11     1 2022-01-10
#> 4 Amparo   2022-11-11     4 2022-10-03
#> 5 Belen    2022-03-25     1 2022-01-10
#> 6 Rayshaun 2022-01-11     1 2022-01-10
#> # ℹ 94 more rows
```

- Note: Birthdays before 2022-01-10 don't get matched ($>=$).

Overlap Joins

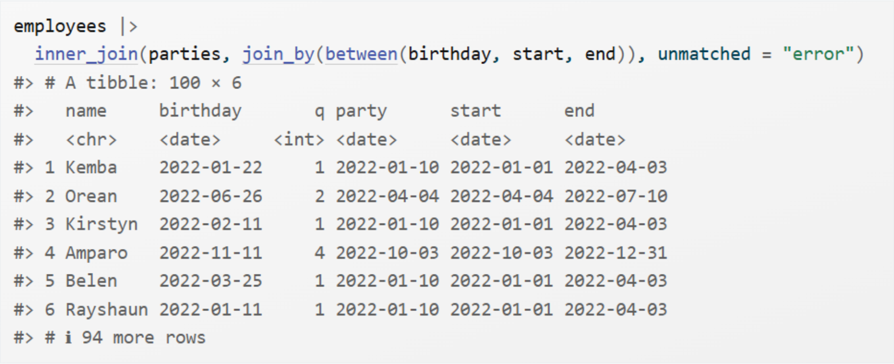
- `between()`, `within()`, `overlaps()`
- Solve problem of unmatched birthdays by using ranges.



```
parties <- tibble(
  q = 1:4,
  party = ymd(c("2022-01-10", "2022-04-04", "2022-07-11", "2022-10-03")),
  start = ymd(c("2022-01-01", "2022-04-04", "2022-07-11", "2022-10-03")),
  end = ymd(c("2022-04-03", "2022-07-10", "2022-10-02", "2022-12-31"))
)
```

Overlap Joins

- `unmatched = "error"`
- `between(x, y_lower, y_upper)`
- `x` is from employees, `y_lower` and `y_upper` are from parties.



```
employees |>
  inner_join(parties, join_by(between(birthday, start, end)), unmatched = "error")
#> # A tibble: 100 × 6
#>   name     birthday       q party      start      end       
#>   <chr>    <date>     <int> <date>     <date>     <date>    
#> 1 Kemba    2022-01-22     1 2022-01-10 2022-01-01 2022-04-03
#> 2 Orean    2022-06-26     2 2022-04-04 2022-04-04 2022-07-10
#> 3 Kirstyn  2022-02-11     1 2022-01-10 2022-01-01 2022-04-03
#> 4 Amparo   2022-11-11     4 2022-10-03 2022-10-03 2022-12-31
#> 5 Belen    2022-03-25     1 2022-01-10 2022-01-01 2022-04-03
#> 6 Rayshaun 2022-01-11     1 2022-01-10 2022-01-01 2022-04-03
#> # ℹ 94 more rows
```