R4DS bookclub

Chapter 13: Relational Data

Ruth Alsancak

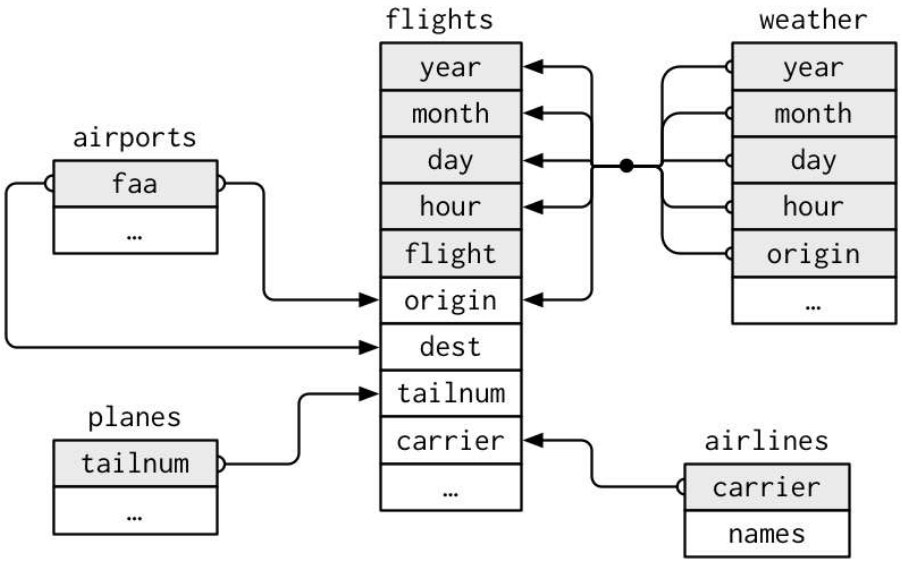
05/10/2020

1 / 14

Relational data

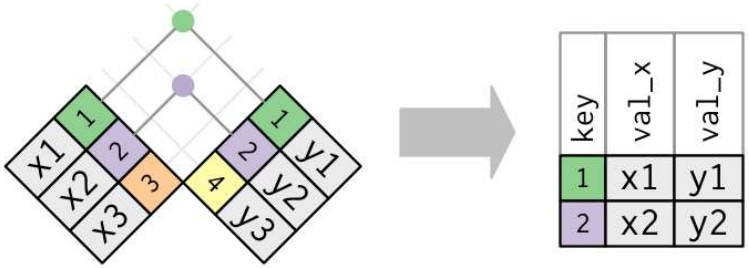
Collectively, multiple tables of data are called **relational data** because it is the relations, not just the individual datasets, that are important.

A variable in one table and its corresponding variable in another table form a **relation**.

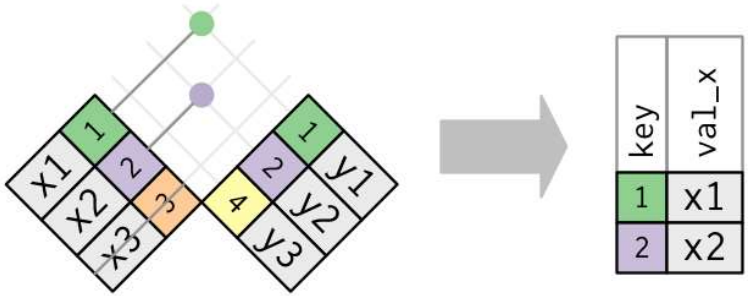


Verbs for working with pairs of tables

- Mutating joins



- Filtering joins



- Set operations

Working with keys

A **key** is a variable (or set of variables) that uniquely identifies an observation:

- A **primary key** uniquely identifies an observation in its own table
- A **foreign key** uniquely identifies an observation in another table
- A **surrogate key** can be added if a table doesn't have a primary key (use mutate() and row_number())

It's good practice to check your primary keys are unique:

```
planes %>%
  count(tailnum) %>%
  filter(n>1)
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: tailnum <chr>, n <int>
```

Duplicate keys in one table may be useful if adding in additional information, but duplicate keys in both tables usually means there's an error (neither of the keys uniquely identify an observation).

Defining the key columns to use for the join

Default, **by = NULL**, joins using all variables that appear in both tables (natural join):

```
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  dewp humid
##    <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>   <dbl> <dbl> <dbl>
##  1  2013     1     1     5 EWR    IAH   N14228  UA       39.0  28.0  64.4
##  2  2013     1     1     5 LGA    IAH   N24211  UA       39.9  25.0  54.8
##  3  2013     1     1     5 JFK    MIA   N619AA  AA       39.0  27.0  61.6
##  4  2013     1     1     5 JFK    BQN   N804JB  B6       39.0  27.0  61.6
##  5  2013     1     1     6 LGA    ATL   N668DN  DL       39.9  25.0  54.8
##  6  2013     1     1     5 EWR    ORD   N39463  UA       39.0  28.0  64.4
##  7  2013     1     1     6 EWR    FLL   N516JB  B6       37.9  28.0  67.2
##  8  2013     1     1     6 LGA    IAD   N829AS  EV       39.9  25.0  54.8
##  9  2013     1     1     6 JFK    MCO   N593JB  B6       37.9  27.0  64.3
## 10  2013     1     1     6 LGA    ORD   N3ALAA  AA       39.9  25.0  54.8
## # ... with 336,766 more rows, and 7 more variables: wind_dir <dbl>,
## #   wind_speed <dbl>, wind_gust <dbl>, precip <dbl>, pressure <dbl>,
## #   visib <dbl>, time_hour <dttm>
```

Defining the key columns to use for the join

by = "key" joins using a single variable that has the same name in both tables:

```
flights2 %>%
  select(-origin, -dest) %>%
  left_join(airlines, by = "carrier")
```

```
## # A tibble: 336,776 x 7
##     year month   day  hour tailnum carrier name
##    <int> <int> <int> <dbl> <chr>   <chr>   <chr>
##  1  2013     1     1     5 N14228  UA      United Air Lines Inc.
##  2  2013     1     1     5 N24211  UA      United Air Lines Inc.
##  3  2013     1     1     5 N619AA  AA      American Airlines Inc.
##  4  2013     1     1     5 N804JB  B6      JetBlue Airways
##  5  2013     1     1     6 N668DN  DL      Delta Air Lines Inc.
##  6  2013     1     1     5 N39463  UA      United Air Lines Inc.
##  7  2013     1     1     6 N516JB  B6      JetBlue Airways
##  8  2013     1     1     6 N829AS  EV      ExpressJet Airlines Inc.
##  9  2013     1     1     6 N593JB  B6      JetBlue Airways
## 10  2013     1     1     6 N3ALAA  AA      American Airlines Inc.
## # ... with 336,766 more rows
```

Defining the key columns to use for the join

by = "x" (character vector) is like a natural join, but only uses some of the common variables:

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

```
## # A tibble: 336,776 x 16
##    year.x month   day  hour origin dest  tailnum carrier year.y type
##     <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>    <int> <chr>
##  1   2013     1     1     5 EWR    IAH   N14228  UA        1999 Fixe~
##  2   2013     1     1     5 LGA    IAH   N24211  UA        1998 Fixe~
##  3   2013     1     1     5 JFK    MIA   N619AA  AA        1990 Fixe~
##  4   2013     1     1     5 JFK    BQN   N804JB  B6        2012 Fixe~
##  5   2013     1     1     6 LGA    ATL   N668DN  DL        1991 Fixe~
##  6   2013     1     1     5 EWR    ORD   N39463  UA        2012 Fixe~
##  7   2013     1     1     6 EWR    FLL   N516JB  B6        2000 Fixe~
##  8   2013     1     1     6 LGA    IAD   N829AS  EV        1998 Fixe~
##  9   2013     1     1     6 JFK    MCO   N593JB  B6        2004 Fixe~
## 10   2013     1     1     6 LGA    ORD   N3ALAA  AA          NA <NA>
## # ... with 336,766 more rows, and 6 more variables: manufacturer <chr>,
## #   model <chr>, engines <int>, seats <int>, speed <int>, engine <chr>
```

Defining the key columns to use for the join

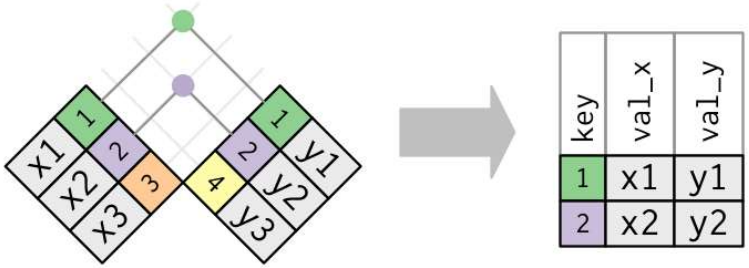
by = c("a" = "b") (named character vector) matches variable a in table x to column b in table y:

```
flights2 %>%
  left_join(airports, c("origin" = "faa"))
```

```
## # A tibble: 336,776 x 15
##     year month   day  hour origin dest  tailnum carrier name    lat   lon   alt
##    <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>   <chr> <dbl> <dbl> <dbl>
##  1  2013     1     1     5 EWR    IAH   N14228  UA      Newa~  40.7 -74.2    18
##  2  2013     1     1     5 LGA    IAH   N24211  UA      La G~  40.8 -73.9    22
##  3  2013     1     1     5 JFK    MIA   N619AA  AA      John~  40.6 -73.8    13
##  4  2013     1     1     5 JFK    BQN   N804JB  B6      John~  40.6 -73.8    13
##  5  2013     1     1     6 LGA    ATL   N668DN  DL      La G~  40.8 -73.9    22
##  6  2013     1     1     5 EWR    ORD   N39463  UA      Newa~  40.7 -74.2    18
##  7  2013     1     1     6 EWR    FLL   N516JB  B6      Newa~  40.7 -74.2    18
##  8  2013     1     1     6 LGA    IAD   N829AS  EV      La G~  40.8 -73.9    22
##  9  2013     1     1     6 JFK    MCO   N593JB  B6      John~  40.6 -73.8    13
## 10  2013     1     1     6 LGA    ORD   N3ALAA  AA      La G~  40.8 -73.9    22
## # ... with 336,766 more rows, and 3 more variables: tz <dbl>, dst <chr>,
## #   tzone <chr>
```

Mutating joins: inner join

Inner join: matches pairs of observations whenever the keys are equal:




```
x %>%
  inner_join(y, by = "key")
```

```
## # A tibble: 2 x 3
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
```

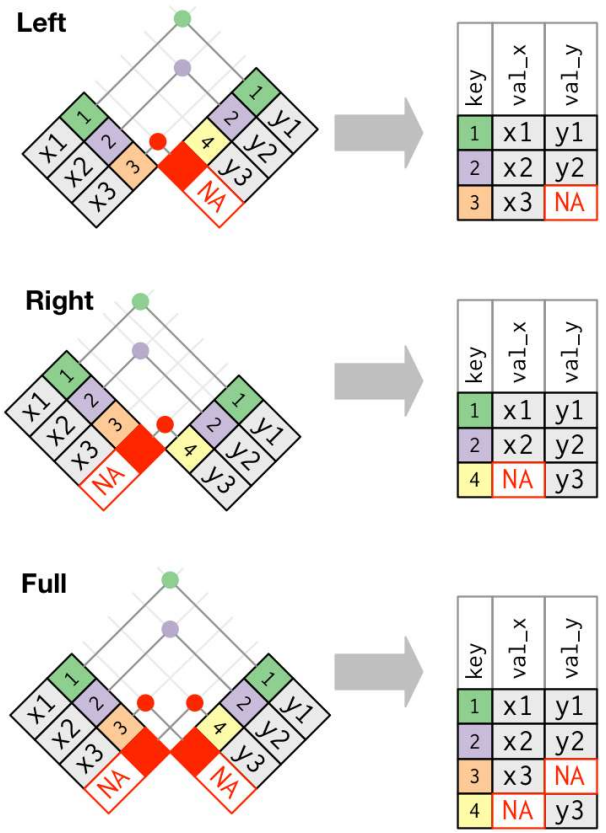
Mutating joins: outer join

Outer join: keeps observations that appear in at least one of the tables:

1. A **left join** keeps all observations in x
2. A **right join** keeps all observations in y
3. A **full join** keeps all observations in x and y

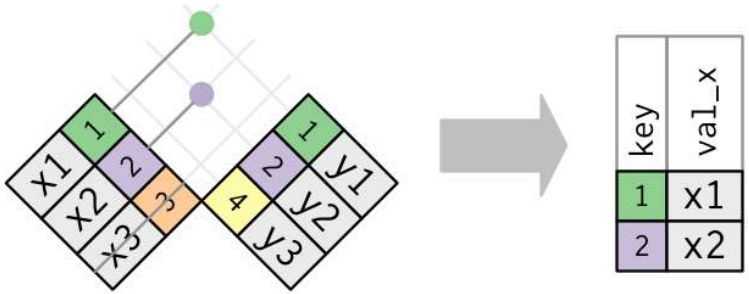
```
x %>%
  full_join(y, by ="key")
```

```
## # A tibble: 4 x 3
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     3 x3    <NA>
## 4     4 <NA>  y3
```



Filtering joins: semi-join

A **semi-join** keeps all observations in x that have a match in y:



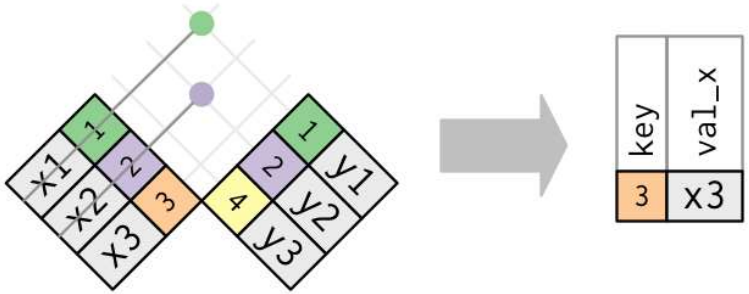

```
flights %>%
  semi_join(top_dest) %>%
  head(2)
```

```
## Joining, by = "dest"
```

```
## # A tibble: 2 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      542            540         2      923            850
## 2  2013     1     1      554            600        -6      812            837
## # ... with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Filtering joins: anti-join

An **anti-join** drops all observations in x that have a match in y:




```
flights %>%
  anti_join(planes, by = "tailnum") %>%
  count(tailnum, sort = TRUE) %>%
  head(5)
```

```
## # A tibble: 5 x 2
##   tailnum     n
##   <chr>   <int>
## 1 <NA>     2512
## 2 N725MQ    575
## 3 N722MQ    513
## 4 N723MQ    507
## 5 N713MQ    483
```

Set operations

Set operations expect x and y inputs to have same variables. They work with complete rows, treat the observations like sets, and compare the values of every variable.

df1

x	y
1	1
2	1

df2

x	y
1	1
1	2

intersect(x, y) returns only observations in both x and y:

```
intersect(df1, df2)
```

```
## # A tibble: 1 x 2
##       x     y
##   <dbl> <dbl>
## 1     1     1
```

Set operations

union(x, y) returns unique observations in x and y:

```
union(df1, df2)
```

```
## # A tibble: 3 x 2
##       x     y
##   <dbl> <dbl>
## 1     1     1
## 2     2     1
## 3     1     2
```

setdiff(x, y) returns observations in x, but not in y:

```
setdiff(df1, df2)
```

```
## # A tibble: 1 x 2
##       x     y
##   <dbl> <dbl>
## 1     2     1
```

```
setdiff(df2, df1)
```

```
## # A tibble: 1 x 2
##       x     y
##   <dbl> <dbl>
## 1     1     2
```