More Data Wrangling and Data Joins

STAT 220

Bastola
January 19 2022

Slicing and selecting data

The slice_ operators let you slice (subset) rows:

- `slice_head(n=5)` : view the first 5 rows
- `slice_tail(n=5)` : view the last 5 rows
- `slice_sample(n=5)` : view 5 random rows
- `slice_min(column, n=5)` : view the 5 smallest values of a column
- `slice_max(column, n=5)` : view the 5 largest values of a column

slice()

```
library(gapminder)
slice(gapminder, 1:5)
# A tibble: 5 × 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
4 Afghanistan Asia       1967    34.0 11537966      836.
5 Afghanistan Asia       1972    36.1 13079460      740.
```

slice()

```
slice(gapminder, -(1:3))
# A tibble: 1,701 × 6
   country     continent  year lifeExp      pop gdpPercap
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
 1 Afghanistan Asia       1967    34.0 11537966      836.
 2 Afghanistan Asia       1972    36.1 13079460      740.
 3 Afghanistan Asia       1977    38.4 14880372      786.
 4 Afghanistan Asia       1982    39.9 12881816      978.
 5 Afghanistan Asia       1987    40.8 13867957      852.
 6 Afghanistan Asia       1992    41.7 16317921      649.
 7 Afghanistan Asia       1997    41.8 22227415      635.
 8 Afghanistan Asia       2002    42.1 25268405      727.
 9 Afghanistan Asia       2007    43.8 31889923      975.
10 Albania     Europe     1952    55.2  1282697     1601.
# … with 1,691 more rows
```

slice_max()

```
gapminder %>%
  slice_max(gdpPercap, n=6)
# A tibble: 6 × 6
  country continent  year lifeExp     pop gdpPercap
  <fct>   <fct>     <int>   <dbl>   <int>     <dbl>
1 Kuwait  Asia       1957    58.0  212846   113523.
2 Kuwait  Asia       1972    67.7  841934   109348.
3 Kuwait  Asia       1952    55.6  160000   108382.
4 Kuwait  Asia       1962    60.5  358266    95458.
5 Kuwait  Asia       1967    64.6  575003    80895.
6 Kuwait  Asia       1977    69.3 1140357    59265.
```

summarize() vs. *mutate()*

summarize() : summarize collapses all variable values down to one number (by group)

```
gapminder %>%
  group_by(continent) %>%
  summarize(avg_life_expectancy = mean(lifeExp))
# A tibble: 5 × 2
  continent avg_life_expectancy
  <fct>                   <dbl>
1 Africa                   48.9
2 Americas                 64.7
3 Asia                     60.1
4 Europe                   71.9
5 Oceania                  74.3
```

summarize() vs. *mutate()*

mutate() : transforms all variable values but preserves the variable length (by group)

```
gapminder %>%
  group_by(continent) %>%
  mutate(meanPop = mean(pop)/1000000)
# A tibble: 1,704 × 7
# Groups:   continent [5]
   country     continent  year lifeExp      pop gdpPercap meanPop
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>   <dbl>
 1 Afghanistan Asia       1952    28.8  8425333      779.    77.0
 2 Afghanistan Asia       1957    30.3  9240934      821.    77.0
 3 Afghanistan Asia       1962    32.0 10267083      853.    77.0
 4 Afghanistan Asia       1967    34.0 11537966      836.    77.0
 5 Afghanistan Asia       1972    36.1 13079460      740.    77.0
 6 Afghanistan Asia       1977    38.4 14880372      786.    77.0
 7 Afghanistan Asia       1982    39.9 12881816      978.    77.0
 8 Afghanistan Asia       1987    40.8 13867957      852.    77.0
 9 Afghanistan Asia       1992    41.7 16317921      649.    77.0
```

group_by()

```
gapminder %>%
  group_by(continent, year) %>%
  summarise(avg_life_expectancy = mean(lifeExp)) %>%
  slice_max(avg_life_expectancy, n = 1)
# A tibble: 5 × 3
# Groups:   continent [5]
  continent  year avg_life_expectancy
  <fct>     <int>               <dbl>
1 Africa     2007                54.8
2 Americas   2007                73.6
3 Asia       2007                70.7
4 Europe     2007                77.6
5 Oceania    2007                80.7
```

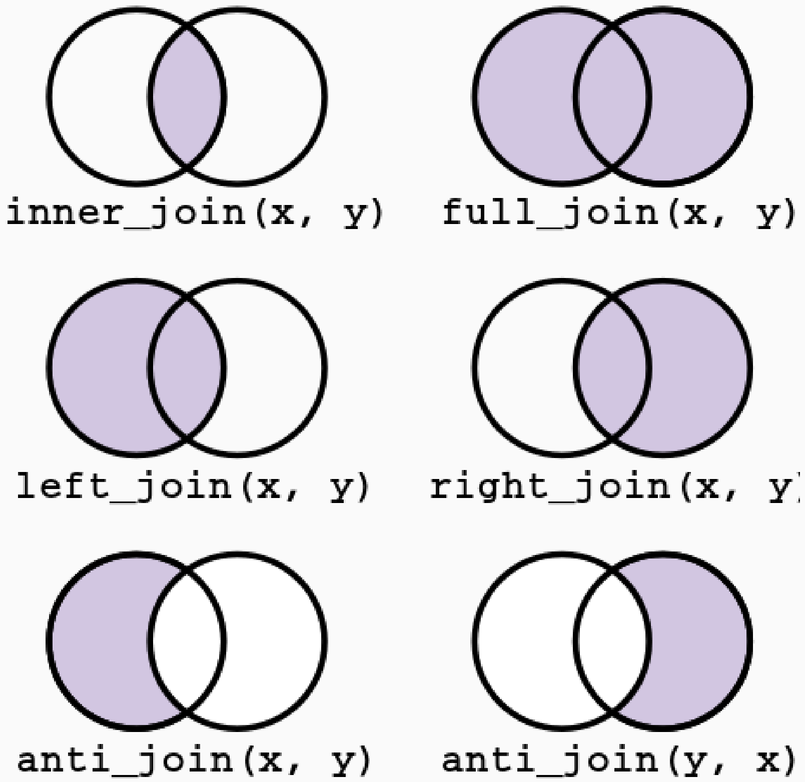
ungroup()

Any further mutations called on it would not use the grouping for aggregate statistics.

```
gapminder %>%
  group_by(continent, year) %>%
  summarise(avg_life_expectancy = mean(lifeExp)) %>%
  ungroup() %>%
  slice_max(avg_life_expectancy, n = 1)
# A tibble: 1 × 3
  continent  year avg_life_expectancy
  <fct>     <int>               <dbl>
1 Oceania    2007                80.7
```

Two-table verbs

- `inner_join()` - Merge two datasets. Exclude all unmatched rows.
- `full_join()` - Merge two datasets. Keep all observations.
- `left_join()` - Merge two datasets. Keep all observations from the origin table.
- `right_join()` - Merge two datasets. Keep all observations from the destination table.
- `anti_join()` - Drops all observations in origin that have a match in destination table.



Mutating Joins

- Mutating joins
    - `left_join()`
    - `right_join()`
    - `inner_join()`
    - `full_join()`
- Differ in their behaviour when a match is not found

Flights and airlines data

```
library(nycflights13)
flights2 <- flights %>%
  select(year:day, hour, origin, dest, tailnum, carrier)
```

```
head(flights2)
# A tibble: 6 × 8
   year month   day  hour origin dest  tailnum carrier
  <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>
1  2013     1     1     5 EWR    IAH   N14228  UA
2  2013     1     1     5 LGA    IAH   N24211  UA
3  2013     1     1     5 JFK    MIA   N619AA  AA
4  2013     1     1     5 JFK    BQN   N804JB  B6
5  2013     1     1     6 LGA    ATL   N668DN  DL
6  2013     1     1     5 EWR    ORD   N39463  UA
```

Airline information

```
head(airlines)
# A tibble: 6 × 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.
```

left_join()

```
flights2 %>%
  left_join(airlines)
# A tibble: 336,776 × 9
    year month   day  hour origin dest  tailnum carrier name
   <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>   <chr>
 1  2013     1     1     5 EWR    IAH   N14228  UA      United Air Lines Inc.
 2  2013     1     1     5 LGA    IAH   N24211  UA      United Air Lines Inc.
 3  2013     1     1     5 JFK    MIA   N619AA  AA      American Airlines Inc.
 4  2013     1     1     5 JFK    BQN   N804JB  B6      JetBlue Airways
 5  2013     1     1     6 LGA    ATL   N668DN  DL      Delta Air Lines Inc.
 6  2013     1     1     5 EWR    ORD   N39463  UA      United Air Lines Inc.
 7  2013     1     1     6 EWR    FLL   N516JB  B6      JetBlue Airways
 8  2013     1     1     6 LGA    IAD   N829AS  EV      ExpressJet Airlines Inc.
 9  2013     1     1     6 JFK    MCO   N593JB  B6      JetBlue Airways
10  2013     1     1     6 LGA    ORD   N3ALAA  AA      American Airlines Inc.
# … with 336,766 more rows
```

Keys: controlling how the tables are matched

```
flights2 %>% left_join(planes, by = "tailnum")
# A tibble: 336,776 × 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 Fixed wing mult…
 2   2013     1     1     5 LGA    IAH   N24211  UA        1998 Fixed wing mult…
 3   2013     1     1     5 JFK    MIA   N619AA  AA        1990 Fixed wing mult…
 4   2013     1     1     5 JFK    BQN   N804JB  B6        2012 Fixed wing mult…
 5   2013     1     1     6 LGA    ATL   N668DN  DL        1991 Fixed wing mult…
 6   2013     1     1     5 EWR    ORD   N39463  UA        2012 Fixed wing mult…
 7   2013     1     1     6 EWR    FLL   N516JB  B6        2000 Fixed wing mult…
 8   2013     1     1     6 LGA    IAD   N829AS  EV        1998 Fixed wing mult…
 9   2013     1     1     6 JFK    MCO   N593JB  B6        2004 Fixed wing mult…
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>
```

Matching keys

```
flights2 %>% left_join(airports, c("origin" = "faa"))
# A tibble: 336,776 × 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      Newar…  40.7 -74.2    18
 2  2013     1     1     5 LGA    IAH   N24211  UA      La Gu…  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 Gu…  40.8 -73.9    22
 6  2013     1     1     5 EWR    ORD   N39463  UA      Newar…  40.7 -74.2    18
 7  2013     1     1     6 EWR    FLL   N516JB  B6      Newar…  40.7 -74.2    18
 8  2013     1     1     6 LGA    IAD   N829AS  EV      La Gu…  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 Gu…  40.8 -73.9    22
# … with 336,766 more rows, and 3 more variables: tz <dbl>, dst <chr>,
#   tzone <chr>
```

inner_join()

```
df1 <- tibble(x = c(1, 2), y = 2:1)
df2 <- tibble(x = c(3, 1), a = 10, b = "a")
```

df1

x	y
1	2
2	1

df2

x	a	b
3	10	a
1	10	a

```
df1 %>% inner_join(df2)
```

x	y	a	b
1	2	10	a

left_join()

df1

x	y
1	2
2	1

df2

x	a	b
3	10	a
1	10	a

```
df1 %>% left_join(df2)
```

x	y	a	b
1	2	10	a
2	1	NA	NA

```
df2 %>% left_join(df1)
```

x	a	b	y
3	10	a	NA
1	10	a	2

right_join()

df1

x	y
1	2
2	1

df2

x	a	b
3	10	a
1	10	a

```
df1 %>% right_join(df2)
```

x	y	a	b
1	2	10	a
3	NA	10	a

```
df2 %>% right_join(df1)
```

x	a	b	y
1	10	a	2
2	NA	NA	1

Your Turn 1

- Please git clone the repository on joining data frames from the course GitHub organization.

- Use the provided `artists` and `bands` tibbles to perform `left_join()` ans `right_join()`.

    - Use `left_join()` to join artists to bands.
    - Use `right_join()` to join bands to artists.
    - Use `setequal()` to check that the datasets are the same.

04:00

full_join()

df1

x	y
1	2
2	1

df2

x	a	b
3	10	a
1	10	a

```
df1 %>% full_join(df2)
```

x	y	a	b
1	2	10	a
2	1	NA	NA
3	NA	10	a

Your Turn 2

Work with the tibbles: `albums`, `songs`, and `labels`.

- Use `inner_join()` to join `albums` to `songs`.
- Use `full_join()` to join `bands` to `artists`.
- Repeat the above using the pipe operator, `%>%`.
- Create one table that combines all information

05:00

Filtering joins

Filtering joins return a copy of the dataset that has been filtered, not augmented (as with mutating joins)

- `semi_join(x,y)` : keeps all observations in x that have a match in y.
- `anti_join(x,y)` : drops all observations in x that have a match in y.

most useful for diagnosing join mismatches

Another example

```
df1 <- tibble(x = c(1, 1, 3, 4), y = 1:4)
df2 <- tibble(x = c(1, 1, 2), z = c("a", "b", "a"))
```

df1

x	y
1	1
1	2
3	3
4	4

df2

x	z
1	a
1	b
2	a

semi_join()

df1

x	y
1	1
1	2
3	3
4	4

df2

x	z
1	a
1	b
2	a

```
df1 %>% semi_join(df2, by = "x")
```

x	y
1	1

```
df2 %>% semi_join(df1, by = "x")
```

x	z
1	a

anti_join()

df1

x	y
1	1
1	2
3	3
4	4

df2

x	z
1	a
1	b
2	a

```
df1 %>% anti_join(df2, by = "x")
```

x	y
3	3

```
df2 %>% anti_join(df1, by = "x")
```

x	z
2	a

Your Turn 3

Continue working with the previous tibble to practice `semi_join()` and `anti_join()`

- Collect `artists` that have `songs` provided.
- Collect the `albums` made by a `band` and count them.
- Return rows of `artists` that don't have bands info. Hint use `anti_join()`.
- Find the rows of `songs` that match a row in `labels` and find the number of rows.

05:00

Set Operations

These expect the x and y inputs to have the same variables, and treat the observations like sets:

- `intersect(x,y)`
  - will return only the rows that appear in both datasets
- `union(x,y)`
  - return every row that appears in one or more of the datasets
  - If a row appears multiple times union will only return it once
- `setdiff(x,y)`
  - will return the rows that appear in the first dataset but not the second

One more example

```
df1 <- tibble(x = 1:2, y = c(1L, 1L))
df2 <- tibble(x = 1:2, y = 1:2)
```

df1

x	y
1	1
2	1

df2

x	y
1	1
2	2

Set operations

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

x	y
1	1

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

x	y
1	1
2	1
2	2

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

x	y
2	1

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

x	y
2	2