

Data wrangling with **dplyr**

Spring 2023

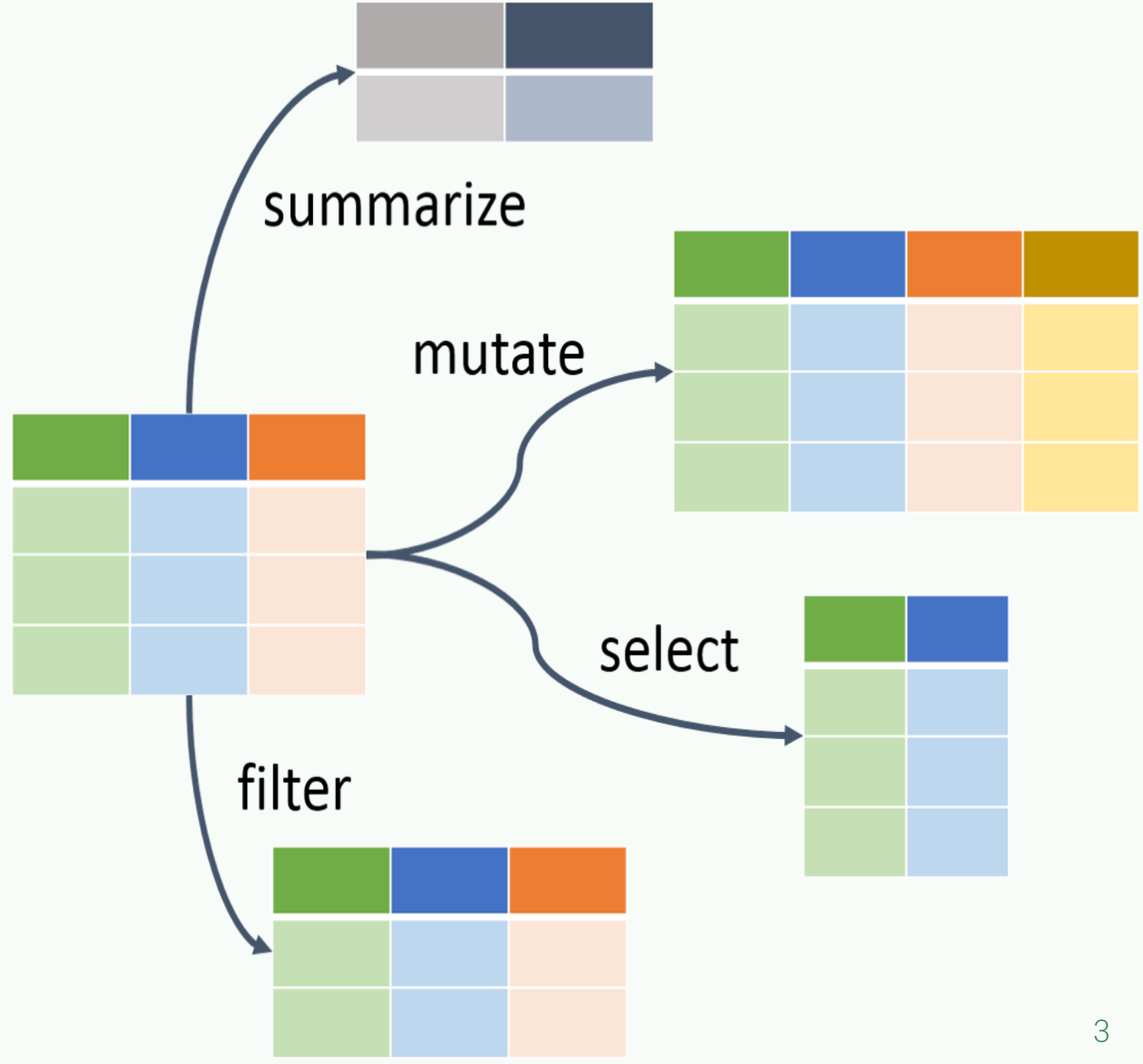
April 07 2023

Data Wrangling

the process of cleaning and unifying messy and complex data sets for easy access and analysis

- "data janitor work"
- importing data
- cleaning data
- changing shape of data

- fixing errors and poorly formatted data elements
- transforming columns and rows
- filtering, subsetting



The Five Verbs

Most of the operations on a data table can be achieved with

- ***select()*** : *extract a subset of columns*
- ***filter()*** : *extract a subset of rows*
- ***mutate()*** : *create new columns*
- ***arrange()*** : *order the rows from smallest to largest (or largest to smallest)*
- ***summarize()*** : *compute a table of summary statistics*

Some Operators

| Operator | Definition |
|----------|--------------------------|
| < | less than |
| <= | less than or equal to |
| > | greater than |
| >= | greater than or equal to |
| == | exactly equal to |
| != | not equal to |
| x & y | x AND y |
| x %in% y | test if x is in y |

Find a subset of the columns using *select()*:

- `select()`: take a subset of the columns (variables/features)

```
library(babynames)
babynames %>%
  select(year, name, n) %>%
  head()
# A tibble: 6 × 3
   year name      n
  <dbl> <chr>   <int>
1  1880 Mary    7065
2  1880 Anna    2604
3  1880 Emma    2003
4  1880 Elizabeth 1939
5  1880 Minnie   1746
6  1880 Margaret 1578
```

Using %>% (pipe operator)

- %>% passes result on left into first argument of function on right
- Chaining functions together lets you read Left-to-right, top-to-bottom

```
babynames %>%                                # dataframe first and then...  
  select(year, name, n) %>%  
  head()
```

Using %>% (pipe operator)

- %>% passes result on left into first argument of function on right
- Chaining functions together lets you read Left-to-right, top-to-bottom

```
babynames %>%                               # dataframe first and then...  
  select(year, name, n) %>%                 # select columns `year`, `name`, and `n` and then...  
  head()
```


Using %>% (pipe operator)

- %>% passes result on left into first argument of function on right
- Chaining functions together lets you read Left-to-right, top-to-bottom

```
babynames %>%           # dataframe first and then...  
  select(year, name, n) %>% # select columns year, name, and n  
  head()                # display header of the data frame
```

Using %>% (pipe operator)

- %>% passes result on left into first argument of function on right
- Chaining functions together lets you read Left-to-right, top-to-bottom

```
babynames %>% # dataframe first and then...  
  select(year, name, n) %>% # select columns year, name, and n  
  head() # display header of the data frame
```

A tibble: 6 × 3

| | year | name | n |
|---|-------|-----------|-------|
| | <dbl> | <chr> | <int> |
| 1 | 1880 | Mary | 7065 |
| 2 | 1880 | Anna | 2604 |
| 3 | 1880 | Emma | 2003 |
| 4 | 1880 | Elizabeth | 1939 |
| 5 | 1880 | Minnie | 1746 |
| 6 | 1880 | Margaret | 1578 |

select() helpers

: select range of columns

```
select(gapminder, income:population)
```

- select every column but

```
select(gapminder, -c(income, population))
```

***starts_with()* select columns
that start with...**

```
select(gapminder, starts_with("p"))
```

***ends_with()* select columns
that end with...**

```
select(gapminder, ends_with("y"))
```

***contains()* select columns
whose names contain...**

```
select(gapminder, contains("e"))
```

Find a subset of the rows using *filter()*:

- `filter()`: take a subset of the rows (observations)

```
babynames %>%  
  filter(name == "Bella") %>%  
  head()  
# A tibble: 6 × 5  
  year sex   name     n   prop  
  <dbl> <chr> <chr> <int> <dbl>  
1  1880 F    Bella    13 0.000133  
2  1881 F    Bella    24 0.000243  
3  1882 F    Bella    16 0.000138  
4  1883 F    Bella    17 0.000142  
5  1884 F    Bella    31 0.000225  
6  1885 F    Bella    25 0.000176
```

Use both *filter()* and *select()*

```
bella <- babynames %>%  
  filter(name == "Bella") %>%  
  select(year, name, sex, n)
```

```
head(bella, 10)  
# A tibble: 10 × 4  
   year name  sex    n  
  <dbl> <chr> <chr> <int>  
1  1880 Bella  F     13  
2  1881 Bella  F     24  
3  1882 Bella  F     16  
4  1883 Bella  F     17  
5  1884 Bella  F     31  
6  1885 Bella  F     25  
7  1886 Bella  F     22  
8  1887 Bella  F     26  
9  1888 Bella  F     31  
10 1889 Bella  F     37
```

```
michael_selected_years <- babynames %>%  
  filter(name == "Michael", sex == "M",  
         year %in% c(1990, 2000, 2010)) %>%  
  select(year, name, sex, n)
```

```
head(michael_selected_years, 10)  
# A tibble: 3 × 4  
   year name  sex    n  
  <dbl> <chr> <chr> <int>  
1  1990 Michael M    65282  
2  2000 Michael M    32035  
3  2010 Michael M    17342
```

arrange()

Order rows from smallest to largest

```
arrange(babynames, n)
# A tibble: 1,924,665 × 5
  year sex   name      n      prop
  <dbl> <chr> <chr>   <int>   <dbl>
1  1880 F    Adelle     5 0.0000512
2  1880 F    Adina     5 0.0000512
3  1880 F  Adrienne   5 0.0000512
4  1880 F  Albertine  5 0.0000512
5  1880 F     Alys     5 0.0000512
6  1880 F     Ana     5 0.0000512
7  1880 F  Araminta   5 0.0000512
8  1880 F   Arthur   5 0.0000512
9  1880 F   Birtha   5 0.0000512
10 1880 F    Bulah    5 0.0000512
# ... with 1,924,655 more rows
```

desc()

Changes ordering from largest to smallest.

```
arrange(babynames, desc(n))
```

```
# A tibble: 1,924,665 × 5
```

| | year | sex | name | n | prop |
|----|-------|-------|---------|-------|--------|
| | <dbl> | <chr> | <chr> | <int> | <dbl> |
| 1 | 1947 | F | Linda | 99686 | 0.0548 |
| 2 | 1948 | F | Linda | 96209 | 0.0552 |
| 3 | 1947 | M | James | 94756 | 0.0510 |
| 4 | 1957 | M | Michael | 92695 | 0.0424 |
| 5 | 1947 | M | Robert | 91642 | 0.0493 |
| 6 | 1949 | F | Linda | 91016 | 0.0518 |
| 7 | 1956 | M | Michael | 90620 | 0.0423 |
| 8 | 1958 | M | Michael | 90520 | 0.0420 |
| 9 | 1948 | M | James | 88588 | 0.0497 |
| 10 | 1954 | M | Michael | 88514 | 0.0428 |

```
# ... with 1,924,655 more rows
```

Most common names in 1990

```
babynames %>%
```

```
  filter(year == 1990) %>%
```

```
  arrange(desc(prop))
```

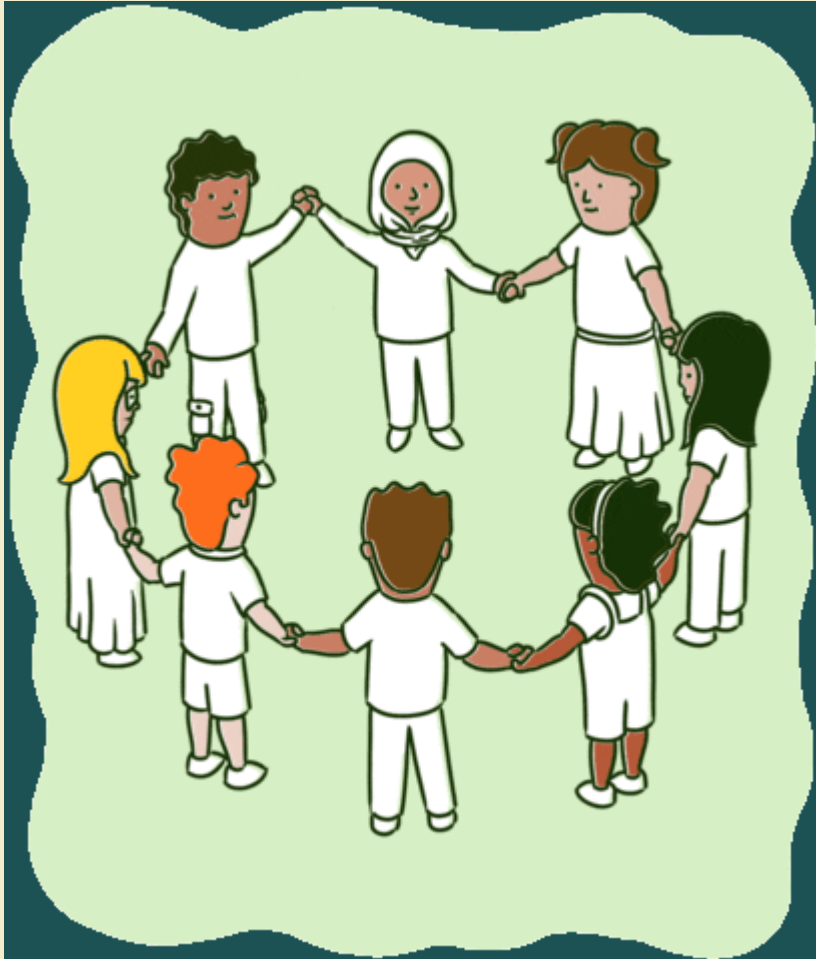
```
# A tibble: 24,719 × 5
```

| | year | sex | name | n | prop |
|----|-------|-------|-------------|-------|--------|
| | <dbl> | <chr> | <chr> | <int> | <dbl> |
| 1 | 1990 | M | Michael | 65282 | 0.0303 |
| 2 | 1990 | M | Christopher | 52332 | 0.0243 |
| 3 | 1990 | F | Jessica | 46475 | 0.0226 |
| 4 | 1990 | F | Ashley | 45558 | 0.0222 |
| 5 | 1990 | M | Matthew | 44800 | 0.0208 |
| 6 | 1990 | M | Joshua | 43216 | 0.0201 |
| 7 | 1990 | F | Brittany | 36538 | 0.0178 |
| 8 | 1990 | F | Amanda | 34408 | 0.0168 |
| 9 | 1990 | M | Daniel | 33815 | 0.0157 |
| 10 | 1990 | M | David | 33742 | 0.0157 |

```
# ... with 24,709 more rows
```


GROUP ACTIVITY 1

10:00



- *Let's go over to maize server/
local Rstudio and our class **moodle***
- *Get the class activity 6.Rmd file*
- *Work on problems 1-3*
- *Ask me questions*

summarize() or *summarise()*

If we want to compare summary statistics, we might use `summarize()`.

```
babynames %>%  
  filter(name == "Bella", sex == "F") %>%  
  summarise(total = sum(n), max = max(n), mean = mean(n))  
# A tibble: 1 × 3  
  total    max  mean  
  <int> <int> <dbl>  
1  57411  5121  416.
```

summarize()

```
babynames %>%  
  filter(name == "Bella", sex == "F") %>%  
  summarize(n = n())  
# A tibble: 1 × 1  
      n  
  <int>  
1  138
```

```
babynames %>%  
  summarize(nname = n_distinct(name))  
# A tibble: 1 × 1  
  nname  
  <int>  
1 97310
```

Using `group_by()`

```
babynames %>%
```

```
  group_by(year, sex)
```

```
# A tibble: 1,924,665 × 5
```

```
# Groups:   year, sex [276]
```

| | year | sex | name | n | prop |
|----|-------|-------|-----------|-------|--------|
| | <dbl> | <chr> | <chr> | <int> | <dbl> |
| 1 | 1880 | F | Mary | 7065 | 0.0724 |
| 2 | 1880 | F | Anna | 2604 | 0.0267 |
| 3 | 1880 | F | Emma | 2003 | 0.0205 |
| 4 | 1880 | F | Elizabeth | 1939 | 0.0199 |
| 5 | 1880 | F | Minnie | 1746 | 0.0179 |
| 6 | 1880 | F | Margaret | 1578 | 0.0162 |
| 7 | 1880 | F | Ida | 1472 | 0.0151 |
| 8 | 1880 | F | Alice | 1414 | 0.0145 |
| 9 | 1880 | F | Bertha | 1320 | 0.0135 |
| 10 | 1880 | F | Sarah | 1288 | 0.0132 |

```
# ... with 1,924,655 more rows
```

Using *group_by()* along with *summarize()*

```
babynames %>%  
  group_by(year) %>%  
  summarise(total = sum(n))  
# A tibble: 138 × 2  
   year  total  
  <dbl> <int>  
1  1880 201484  
2  1881 192696  
3  1882 221533  
4  1883 216946  
5  1884 243462  
6  1885 240854  
7  1886 255317  
8  1887 247394  
9  1888 299473  
10 1889 288946  
# ... with 128 more rows
```

mutate()

- `mutate()` lets us create new variables based on manipulations of the old variables

```
babynames <- babynames %>%  
  group_by(year) %>%
```

```
  mutate(percent = prop * 100)
```

```
head(babynames)
```

```
# A tibble: 6 × 6
```

```
# Groups:   year [1]
```

| | year | sex | name | n | prop | percent |
|---|-------|-------|-----------|-------|--------|---------|
| | <dbl> | <chr> | <chr> | <int> | <dbl> | <dbl> |
| 1 | 1880 | F | Mary | 7065 | 0.0724 | 7.24 |
| 2 | 1880 | F | Anna | 2604 | 0.0267 | 2.67 |
| 3 | 1880 | F | Emma | 2003 | 0.0205 | 2.05 |
| 4 | 1880 | F | Elizabeth | 1939 | 0.0199 | 1.99 |
| 5 | 1880 | F | Minnie | 1746 | 0.0179 | 1.79 |
| 6 | 1880 | F | Margaret | 1578 | 0.0162 | 1.62 |

top_n()

Most common name in each year

```
babynames %>%
```

```
  group_by(year) %>%
```

```
  top_n(1, prop)
```

```
# A tibble: 138 × 6
```

```
# Groups:   year [138]
```

| | year | sex | name | n | prop | percent |
|----|-------|-------|-------|-------|--------|---------|
| | <dbl> | <chr> | <chr> | <int> | <dbl> | <dbl> |
| 1 | 1880 | M | John | 9655 | 0.0815 | 8.15 |
| 2 | 1881 | M | John | 8769 | 0.0810 | 8.10 |
| 3 | 1882 | M | John | 9557 | 0.0783 | 7.83 |
| 4 | 1883 | M | John | 8894 | 0.0791 | 7.91 |
| 5 | 1884 | M | John | 9388 | 0.0765 | 7.65 |
| 6 | 1885 | M | John | 8756 | 0.0755 | 7.55 |
| 7 | 1886 | M | John | 9026 | 0.0758 | 7.58 |
| 8 | 1887 | M | John | 8110 | 0.0742 | 7.42 |
| 9 | 1888 | M | John | 9247 | 0.0712 | 7.12 |
| 10 | 1889 | M | John | 8548 | 0.0718 | 7.18 |

```
# ... with 128 more rows
```

min_rank() : A go to ranking function (ties share the lowest rank)

```
min_rank(c(50, 100, 1000))  
[1] 1 2 3
```

```
min_rank(desc(c(50, 100, 1000)))  
[1] 3 2 1
```


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_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  
10 Afghanistan Asia      1997    41.8 22227415    635.    77.0  
# ... with 1,694 more rows
```

group_by(var1, var2)

```
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
```

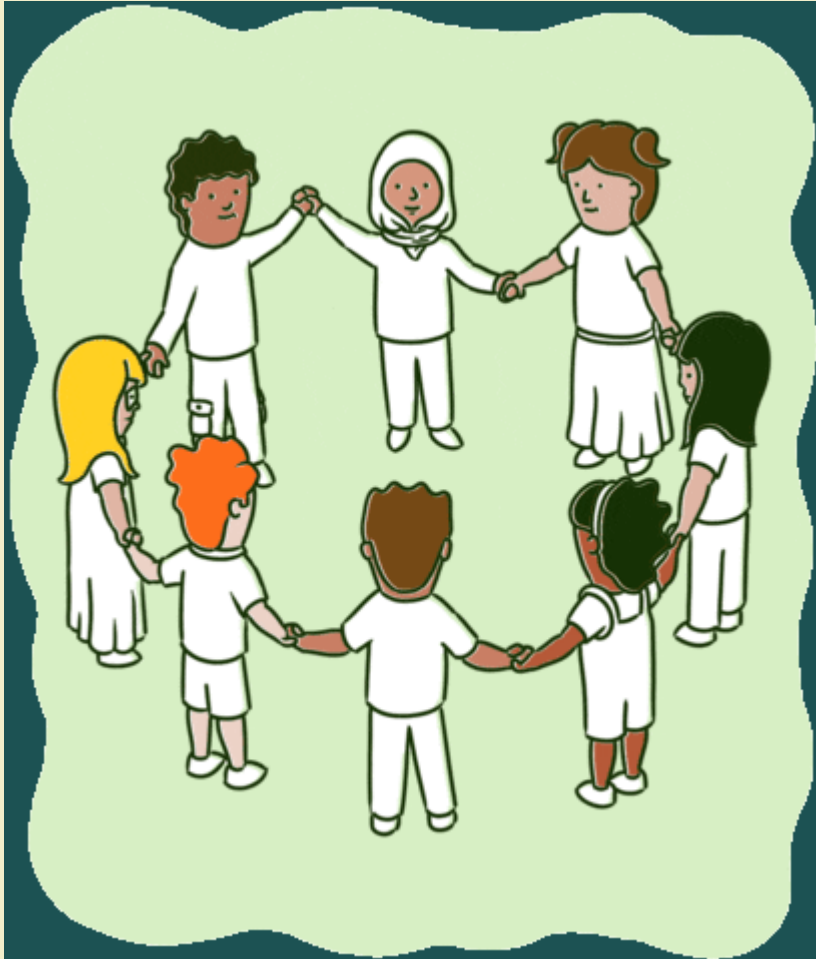
ungroup()

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
```

✎ GROUP ACTIVITY 2

10:00



- *Work on problems 4-6*
- *Ask me questions*