data wrangling with dplyr

Credit: JustTooLazy. Licensed under CC BY 2.0.

Andrew Jones | Strategy Unit

Wrangling

Reshaping or transforming data into a format which is easier to work with

(...for later visualisation, computing of statistics, or modelling.)

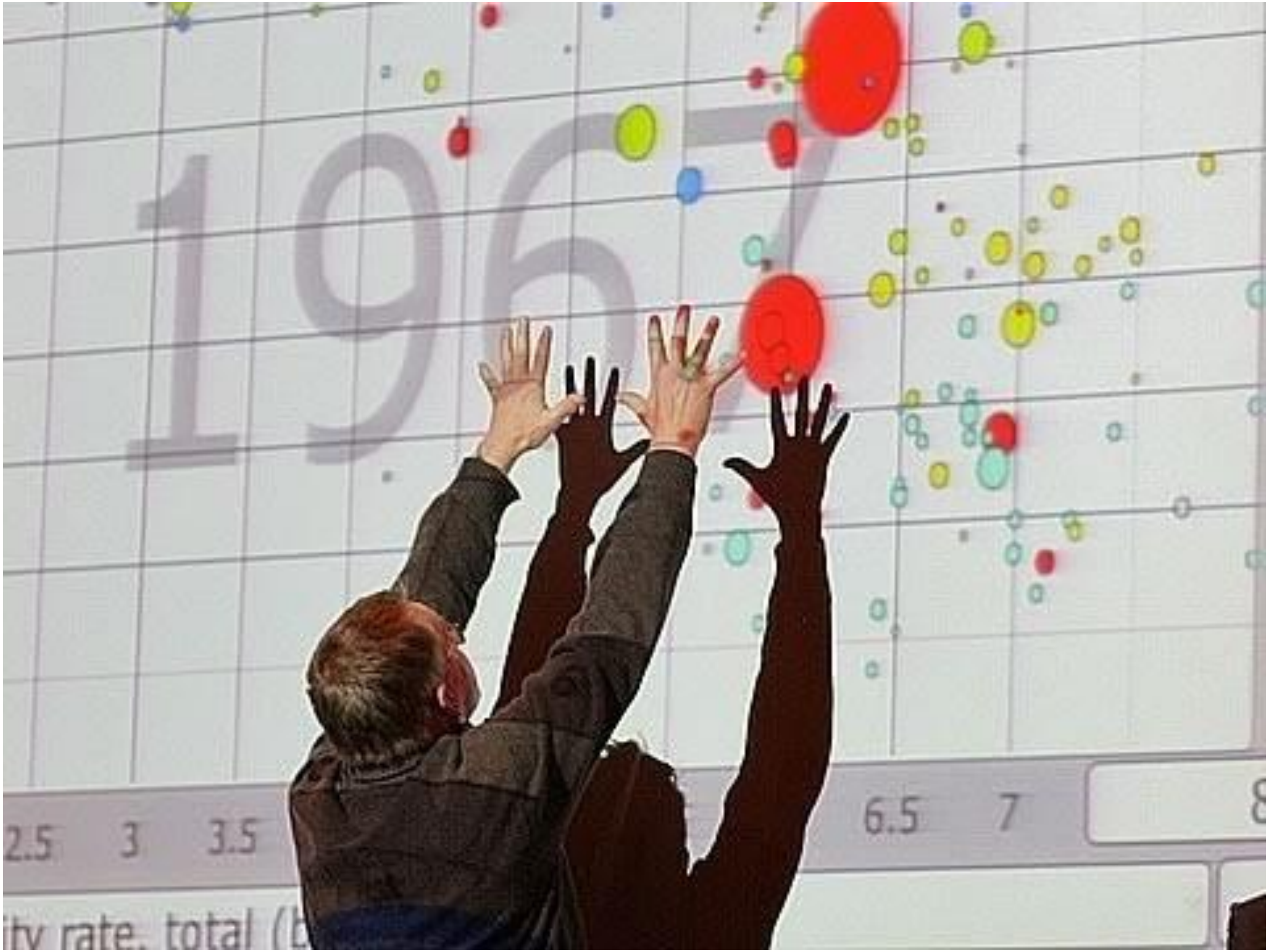
The dplyr package

dplyr is a language for data manipulation

Most wrangling puzzles can be solved with knowledge of just 5 dplyr verbs (5 functions).

These verbs will be the subject of this session.

Gapminder



Gapminder

Data from gapminder.org

```
install.packages(“gapminder”)
library(gapminder)
```

Q. How many variables here?

Meaningful names?

What type?

dplyr

5 verbs

arrange
filter
mutate
summarise
group_by

will help us gain a deeper understanding of our data sets.

An aside:

Very soon we will want to use a series of these dplyr commands…

Series of commands = Recipe

Imagine a recipe for mashed potato:

Start with a...

potato then
peel then
slice into medium sized pieces then
boil for 25 minutes

An aside:

Imagine a recipe for mashed potato:

Start with an object

`potato` then
  `peel()` then
  slice into medium sized pieces then
  boil for 25 minutes then
  mash

An aside:

Imagine a recipe for mashed potato:

```
potato then
   peel()  then
   slice(size = “medium”) then
   boil for 25 minutes then
   mash
```

An aside:

Imagine a recipe for mashed potato:

```
potato then
  peel()  then
  slice(size = “medium”) then
  boil(t = 25) then
  mash
```

An aside:

Imagine a recipe for mashed potato:

```
potato %>%
  peel() %>%
  slice(size = “medium”) %>%
  boil(t = 25)
```

An aside:

Imagine a recipe for mashed potato:

Input object

```
potato %>%
  peel() %>%
  slice(size = “medium”) %>%
  boil(t = 25)
```

Output = hot chopped potato

An aside:

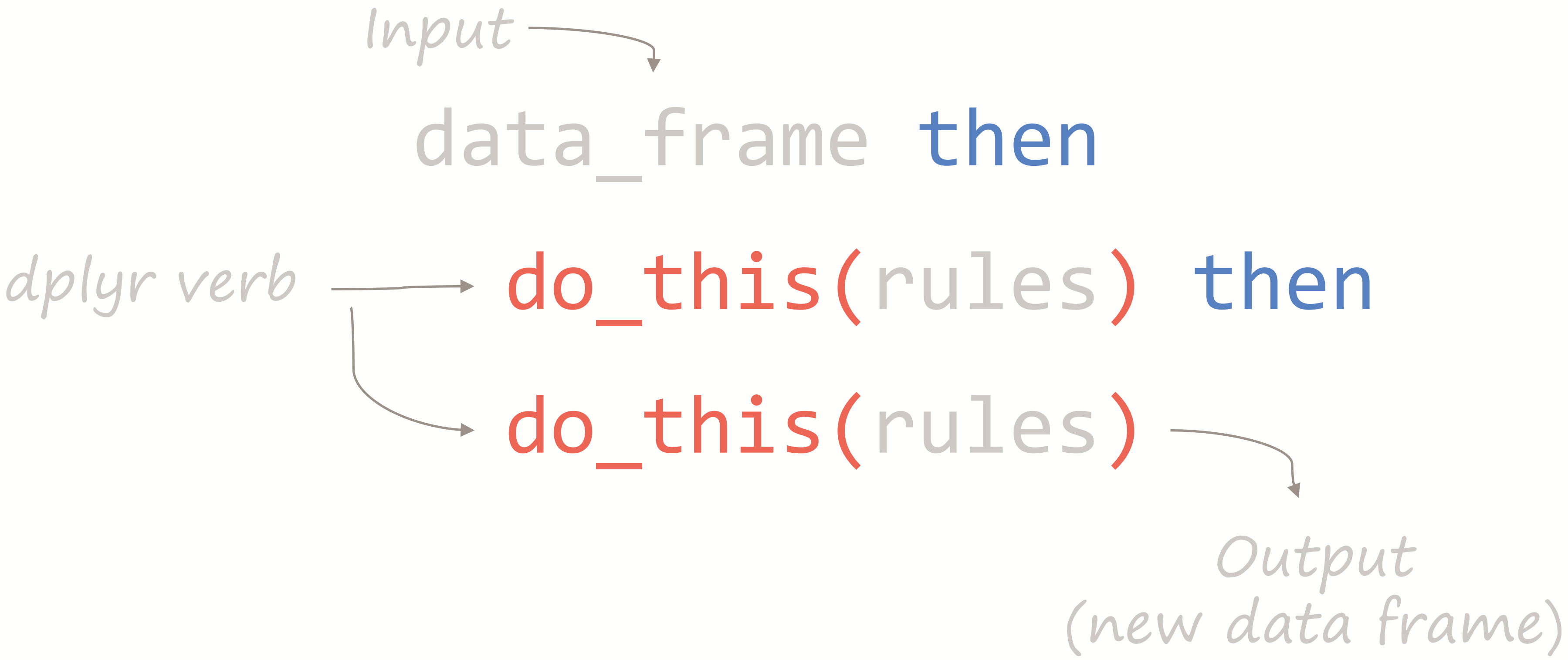
Imagine a recipe for mashed potato:

```
potato %>%
  peel() %>%
  slice(size = “medium”) %>%
  boil(t = 25)
```

What is the output after this step?

Each step builds on the previous one

Tidyverse syntax



Input

```
data_frame then
  do_this(rules) then
  do_this(rules)
```

dplyr verb

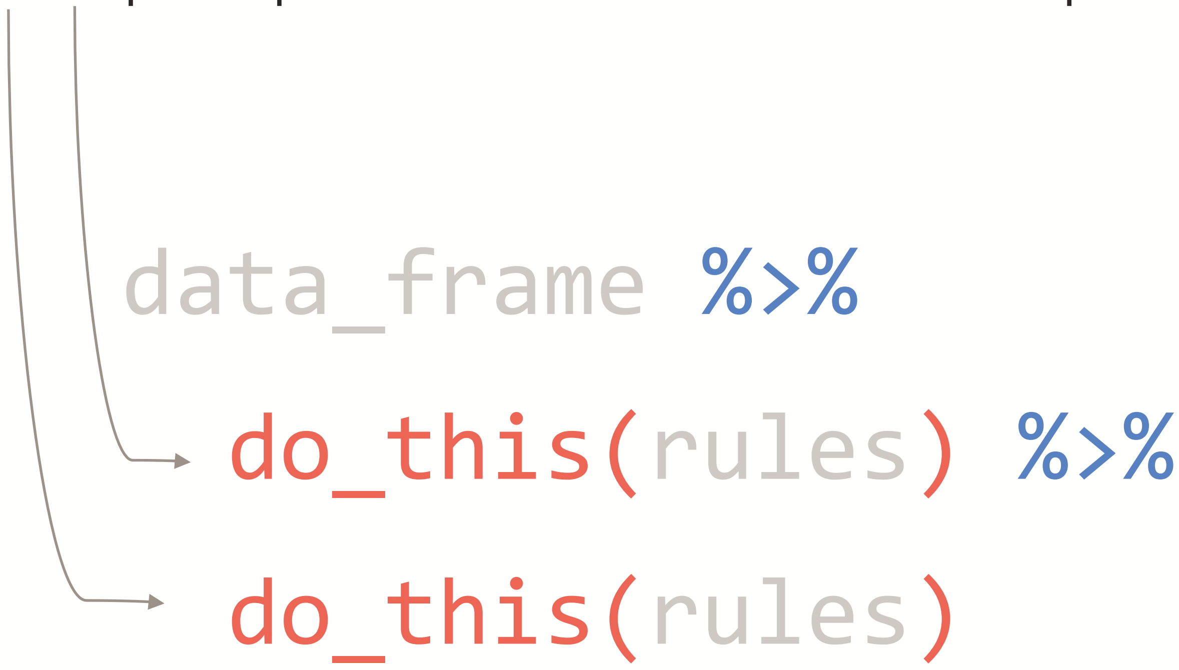
Output
(new data frame)

Tidyverse syntax

```
data_frame %>%
  do_this(rules) %>%
  do_this(rules)
```

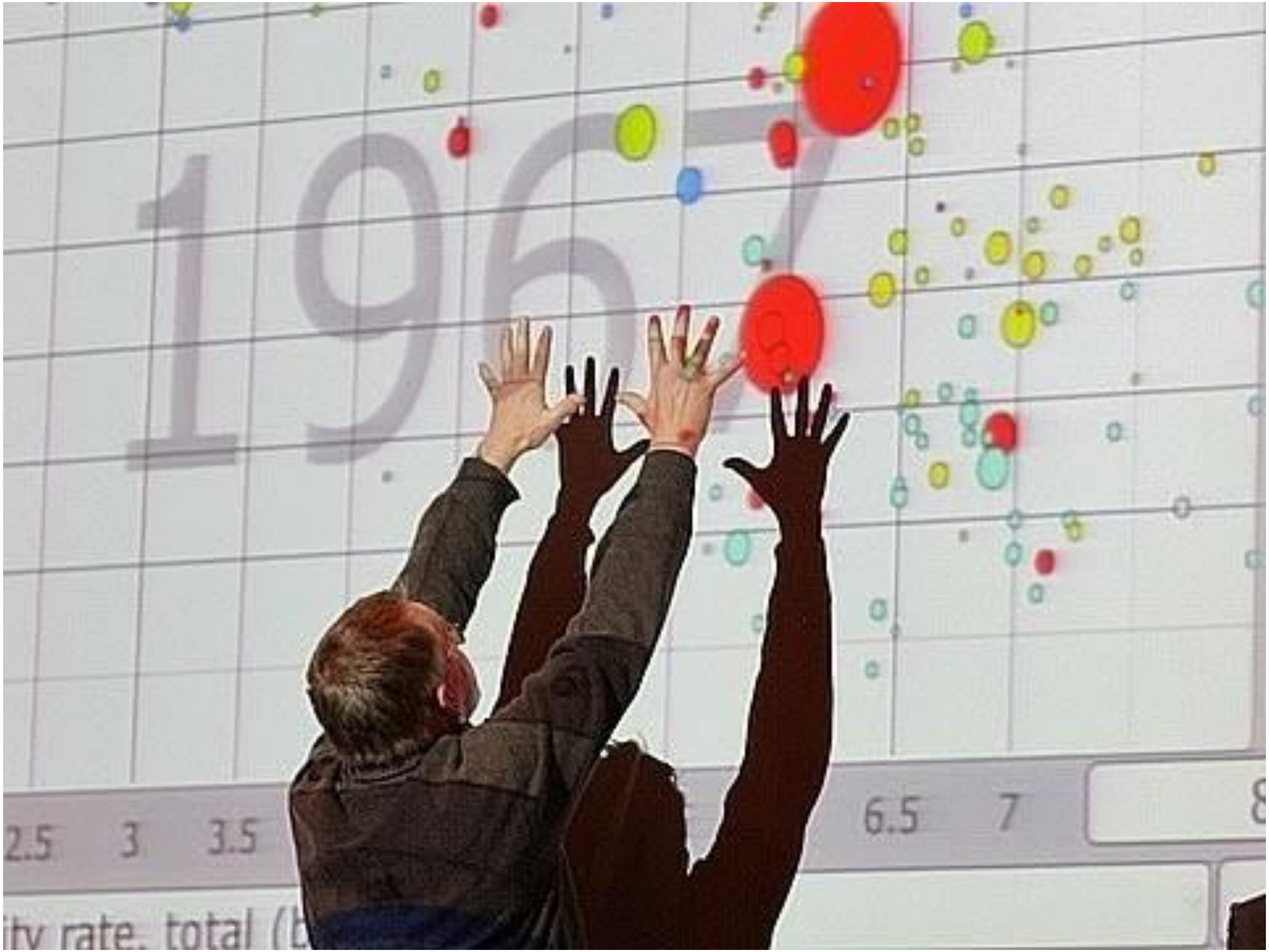
The tidyverse

Combine simple pieces to solve complex puzzles



```
data_frame %>%
  do_this(rules) %>%
  do_this(rules)
```

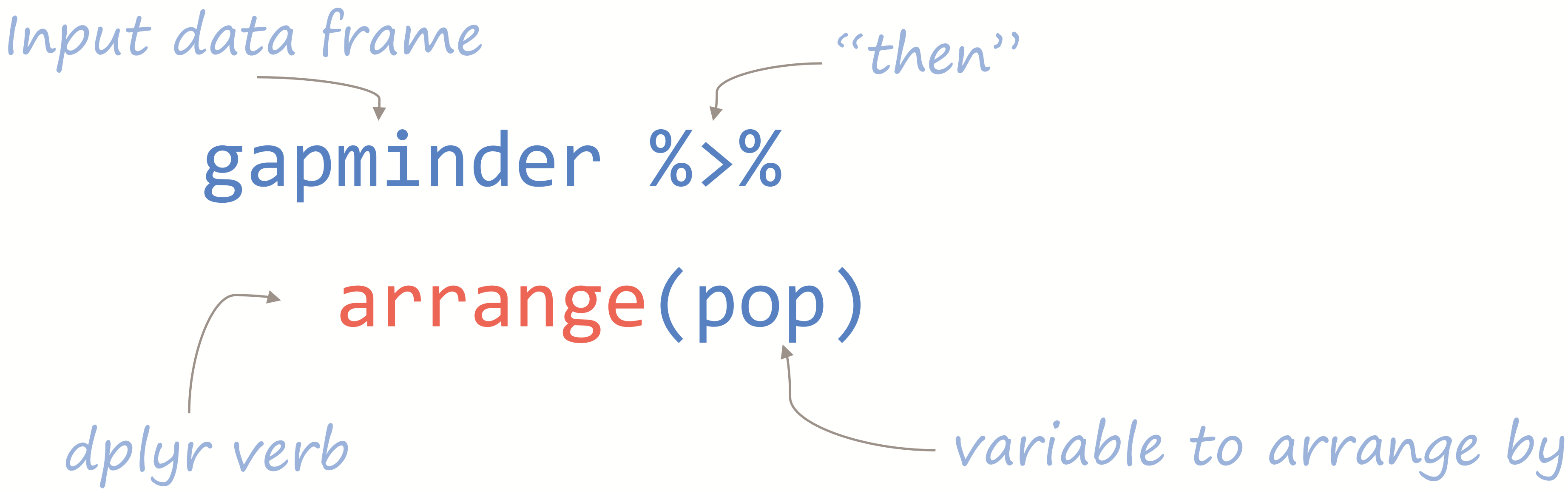
Gapminder



Q1. Which country in this Gapminder excerpt has the lowest population?

1. arrange

Reorder rows based on selected variable



Input data frame

"then"

```
gapminder %>%
    arrange(pop)
```

dplyr verb

variable to arrange by

1. arrange

Reorder rows based on selected variable

Input data frame

"then" = Ctrl + Shift + m

```
gapminder %>%
  arrange(pop)
```

dplyr verb

variable to arrange by

1. arrange

If we wanted descending order:

```
gapminder %>%
  arrange(desc(pop))
```

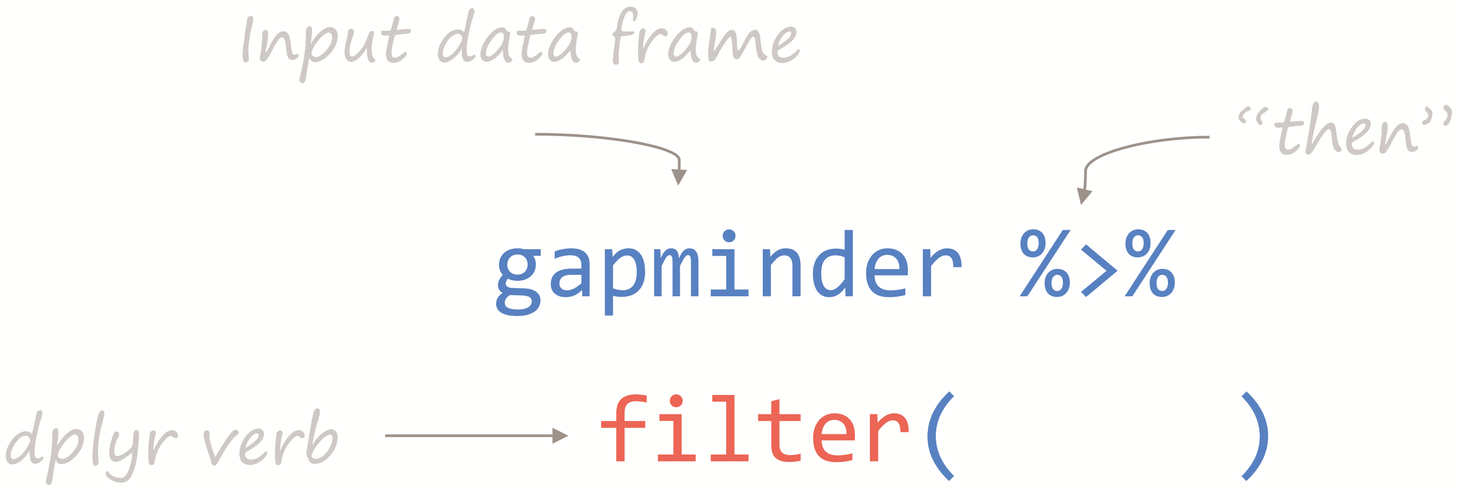
for text and numeric variables

Q2. Which 5 countries have the highest human populations? (in 2007)

```
gapminder %>%
  arrange(desc(pop))
```

2. filter

pick observations by their value



Input data frame

“then”

```
gapminder %>%
  filter(      )
```

dplyr verb

2. filter

pick observations by their value

```
gapminder %>%
  filter(year == 2007)
```

2. filter

pick observations by their value

```
gapminder %>%
  filter(year == 2007)
```

The expression inside brackets should return TRUE or FALSE

We are testing equality so ==

we are choosing rows where this expression is TRUE

2. filter

"then"
strings multiple
verbs
together

```
gapminder %>%
  arrange(desc(pop)) %>%
  filter(year == 2007)
```

2. filter

Use quotes if referring to text (character) strings

```
gapminder %>%
    filter(continent == “Africa”) %>%
    arrange(desc(pop))
```

‘single’ or “double” as you wish

Break

Picture: White, K. (2011). 101 Things to Learn in Art School. MIT Press.

Q3. Which 5 countries* have the lowest GDP? (2007)

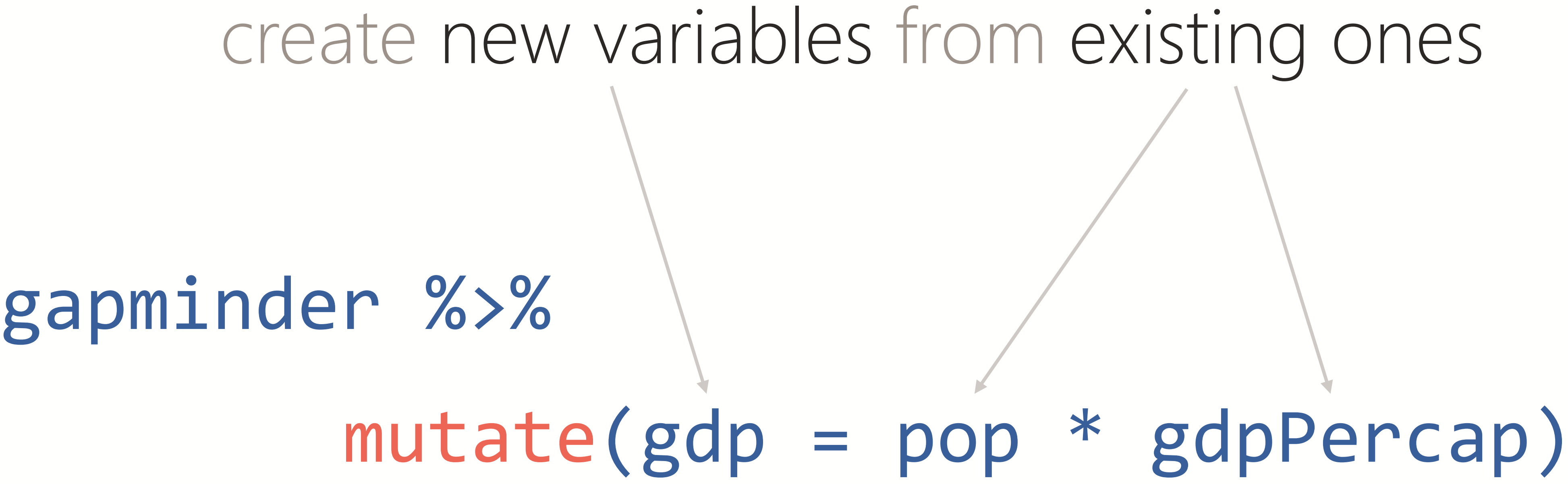
*Not all countries represented in data

Q3. Which 5 countries have the lowest GDP?

```
gapminder %>%
  filter(year == 2007) %>%
  arrange(gdpPercap)
```

This is per capita GDP
(but we can get what we need from existing variables)

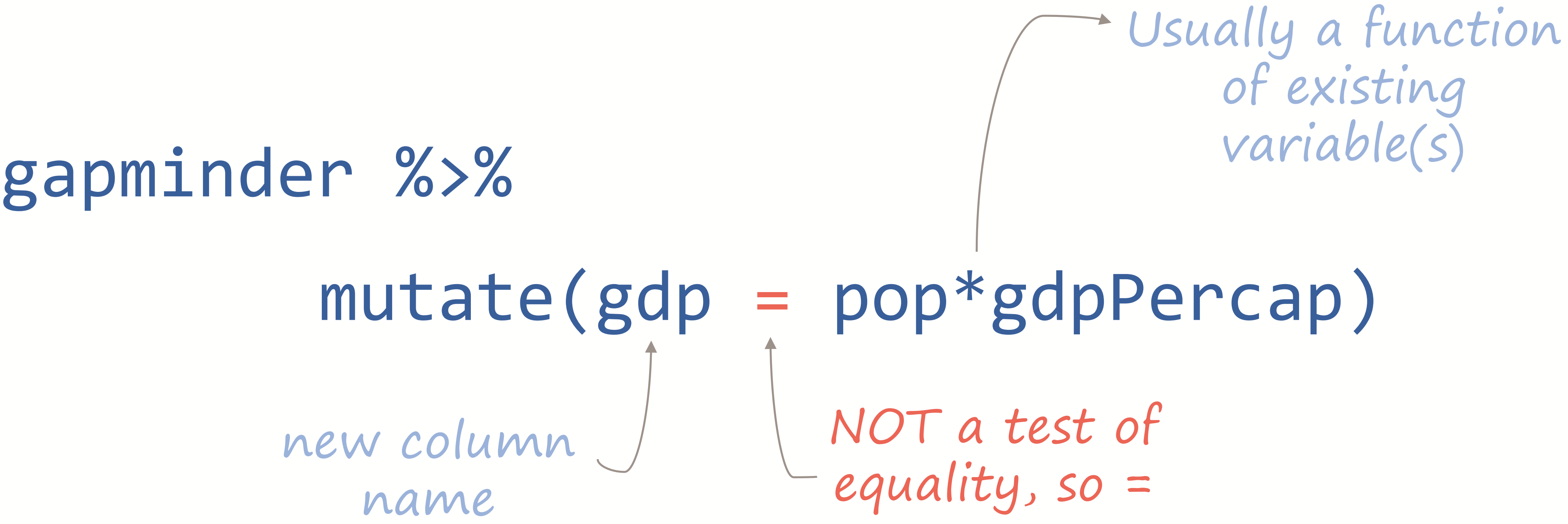
3. mutate



create new variables from existing ones

```
gapminder %>%
        mutate(gdp = pop * gdpPercap)
```

3. mutate



```
gapminder %>%
        mutate(gdp = pop*gdpPercap)
```

Usually a function of existing variable(s)

new column name

NOT a test of equality, so =

3. mutate

```
gapminder %>%
           mutate(gdp = pop*gdpPercap) %>%
            filter(year == 2007) %>%
            arrange(gdp)
```

We can refer to variables we've just created in the same pipe chain

Q4. Which country has the highest population for each year of data?

5. summarise

collapse many values into a single summary value

```
gapminder %>%
   summarise(pop_high = max(pop))
```

Similar to mutate:

new column name

(summary) function

4. group_by

For each group…

… summarise (collapse into a single summary value)

```
gapminder %>%
  group_by(year) %>%
   summarise(pop_high = max(pop))
```

4. group_by

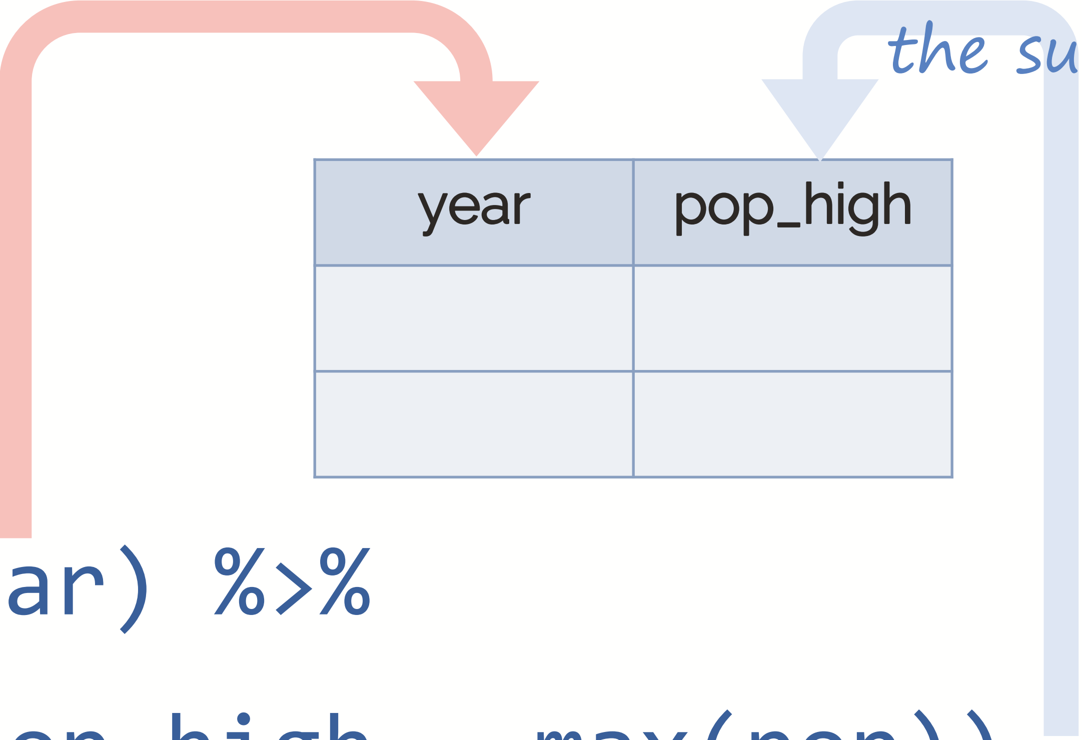
Useful if we desire breakdowns by variable(s)

```
gapminder %>%
  group_by(year) %>%
   summarise(pop_high = max(pop))
```

4.group_by *and* 5.summarise

A column is created for each grouping variable

A column for the summary



year	pop_high

```
gapminder %>%
    group_by(year) %>%
    summarise(pop_high = max(pop))
```

4.group_by *and* 5.summarise

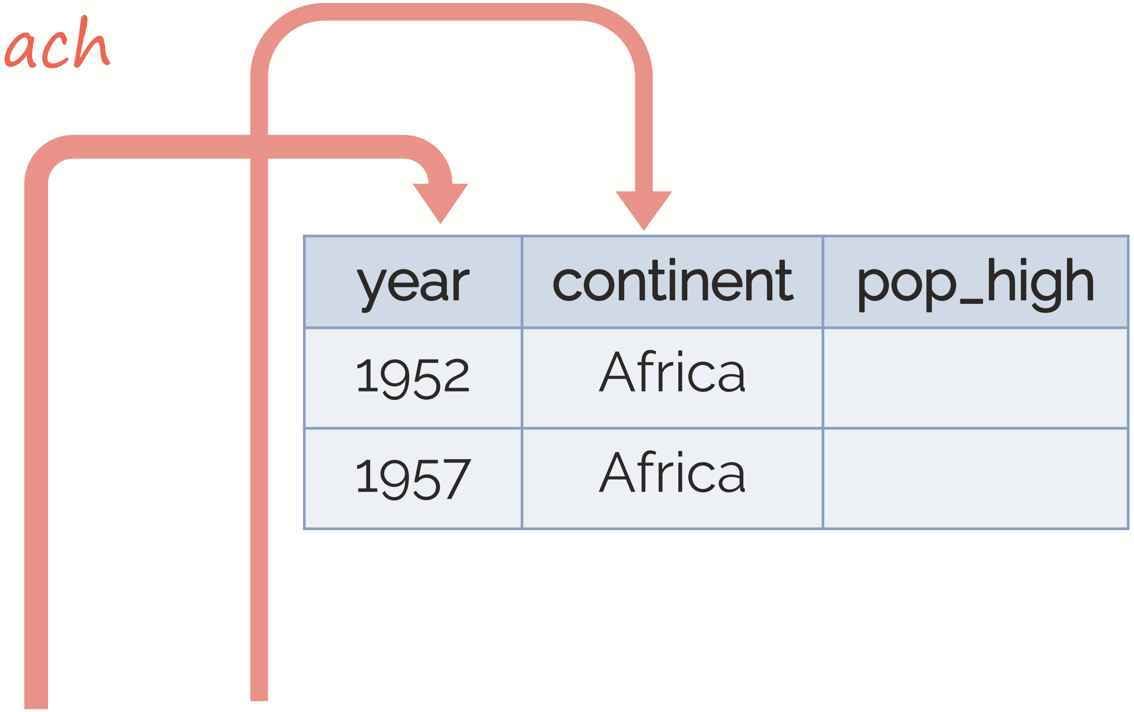
A row for each year group

year	pop_high
1952	
1957	

```
gapminder %>%
    group_by(year) %>%
    summarise(pop_high = max(pop))
```

4.group_by *and* 5.summarise

A column is created for each grouping variable



year	continent	pop_high
1952	Africa	
1957	Africa	

```
gapminder %>%
    group_by(year, continent) %>%
    summarise(mean_life = mean(lifeExp))
```

4.group_by *and* 5.summarise

A row for each unique combo of the grouping variables

year	continent	pop_high
1952	Africa	
1957	Africa	

```
gapminder %>%
    group_by(year, continent) %>%
    summarise(mean_life = max(pop))
```

Q5. How has mean life expectancy in Africa changed (1952–2007)?

A summary value...

Q5. How has mean life expectancy in Africa changed (1952–2007)?

A summary value...

Q5. How has mean life expectancy in Africa changed (1952–2007)?

for each year...

A summary value...

Q5. How has mean life expectancy in Africa changed (1952–2007)?

but pick only the African continent

for each year...

Over to you:

A summary value...

Q5. How has mean life expectancy in Africa changed (1952–2007)?

but pick only the African continent

for each year...

Extension:

Q. How many countries from each continent?

Hint:
filter for one year then use:

```
summarise(your_col_name = n())
```

This is a common pattern – it will count the number of rows in each group

Q5.~solution

```
gapminder %>%
  filter(continent == "Africa") %>%
  group_by(year) %>%
  summarise(mean_life = mean(lifeExp))
```

Extension~solution

```
gapminder %>%
  filter(year == 2007) %>%
  group_by(continent) %>%
  summarise(n = n())
```

I'll just call this column "n"

6. select

select a subset of variables from existing data set

```
gapminder %>%
        select(var1, var2)
```

6. select

select a subset of variables from existing data set

```
gapminder %>%
          select(-var2)
```

To remove a column

6. select

select a subset of variables from existing data set

```
gapminder %>%
        select(1:5)
```

You can also refer to columns by number. Here 1:5 saves having to type: 1,2,3,4,5

6. select

select a subset of variables from existing data set

```
gapminder %>%
        select(var6, everything())
```

If you want this column at the start of your data frame

This work is licensed as

End