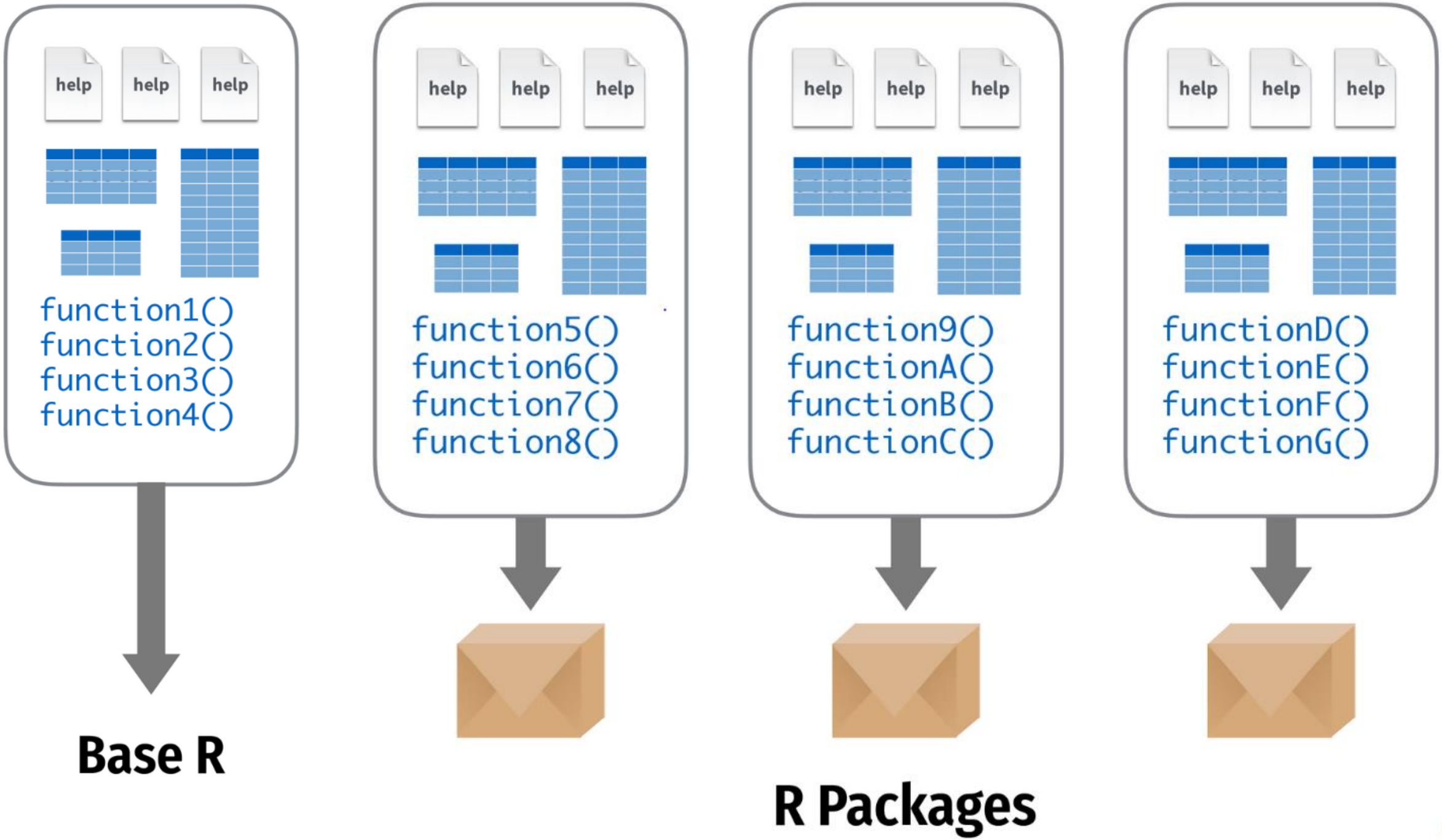
The Tidyverse

Plan for this session

- Packages and data
- Transform data with dplyr


help
help
help
function1()
function2()
function3()
function4()
Base R
help
help
help
function5()
function6()
function7()
function8()
help
help
help
function9()
functionA()
functionB()
functionC()
R Packages
help
help
help
functionD()
functionE()
functionF()
functionG()

Using packages

```
install.packages("name")
```

Downloads files to your computer

Do this once per computer

```
library("name")
```

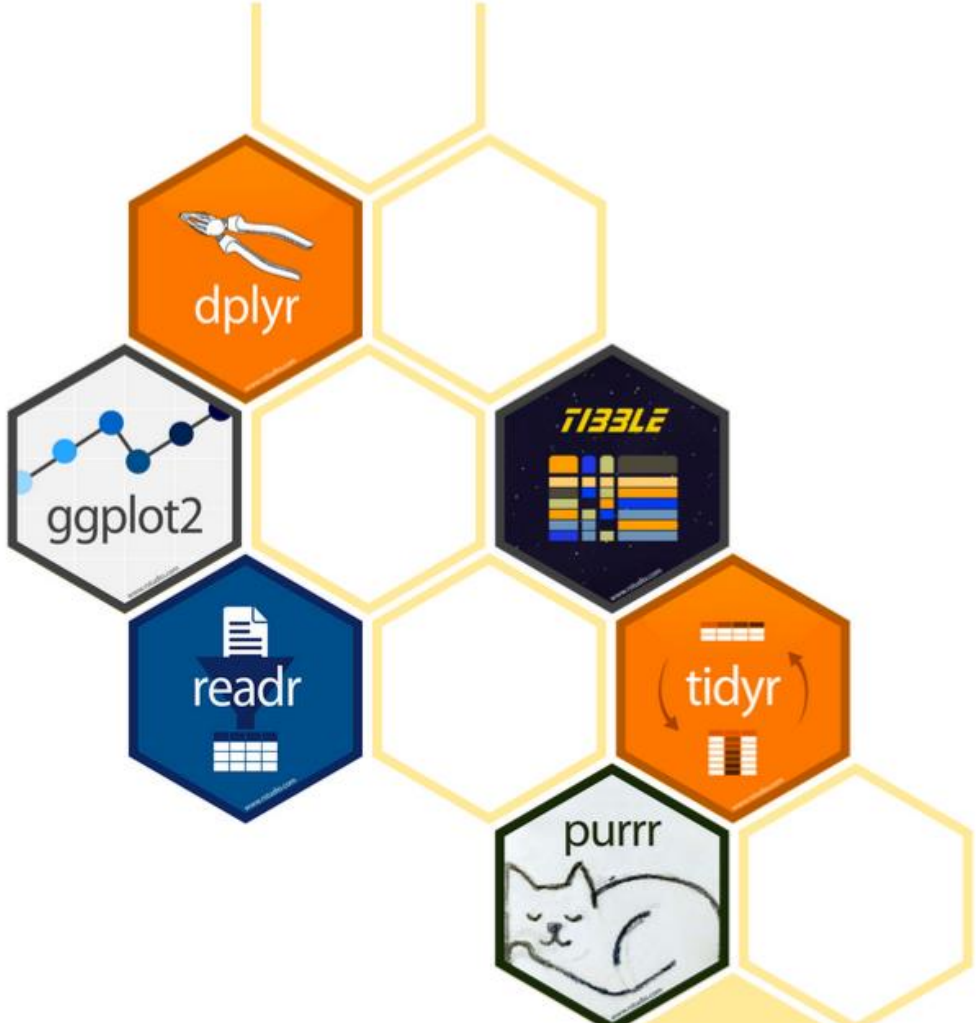
Loads the package

Do this once per R session

The tidyverse

"The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures."

... the tidyverse makes data science faster, easier and more fun...



Loading the tidyverse package(s)

```
library(tidyverse)
```

The tidyverse package is a shortcut for installing and loading all the key tidyverse packages

```r
install.packages("tidyverse")
```

```r
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("readr")
install.packages("purrr")
install.packages("tibble")
install.packages("stringr")
install.packages("forcats")
install.packages("lubridate")
install.packages("hms")
install.packages("DBI")
install.packages("haven")
install.packages("httr")
install.packages("jsonlite")
install.packages("readxl")
install.packages("rvest")
install.packages("xml2")
install.packages("modelr")
install.packages("broom")
```

Data frames and tibbles

- Data frames are the most common kind of data objects; used for rectangular data (like spreadsheets)
- Data frames: R's native data object
- Tibbles (tbl): a fancier enhanced kind of data frame
- (You really won't notice a difference in this class)

Vectors

Vectors are a list of values of the same time (all text, or all numbers, etc.)

Make them with `c()`:

```
c(1, 4, 2, 5, 7)
```

You'll usually want to assign them to something:

```
neat_numbers <- c(1, 4, 2, 5, 7)
```

Data types

Integer	Whole numbers	`c(1, 2, 3, 4)`
Double	Numbers	`c(1, 2.4, 3.14, 4)`
Character	Text	`c("1", "blue", "fun", "monster")`
Logical	True or false	`c(TRUE, FALSE, TRUE, FALSE)`
Factor	Category	`c("Strongly disagree", "Agree", "Neutral")`

Packages for importing data

readr	Work with plain text data	`my_data <- read_csv("file.csv")`
readxl	Work with Excel files	`my_data <- read_excel("file.xlsx")`
haven	Work with Stata, SPSS, and SAS data	`my_data <- read_stata("file.dta")`

Transform data with dplyr

Gapminder

- Excerpt of the Gapminder data on life expectancy, GDP per capita, and population by country.
- The main data frame gapminder has 1704 rows and 6 variables
  - Country -factor with 142 levels
  - Continent - factor with 5 levels
  - Year - ranges from 1952 to 2007 in increments of 5 years
  - lifeExp - life expectancy at birth, in years
  - Pop - population
  - gdpPercap - GDP per capita (US$, inflation-adjusted)

```r
gapminder
```

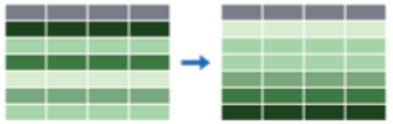
```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # … with 1,694 more rows
```

dplyr: verbs for manipulating data

Extract rows with `filter()`

Extract columns with `select()`

Arrange/sort rows with `arrange()`



Make new columns with `mutate()`

Make group summaries with `group_by() %>% summarize()`

filter()

Extract rows that meet some sort of test

```
filter(.data = DATA, ...)
```

- **DATA** = Data frame to transform
- **...** = One or more tests

`filter()` returns each row for which the test is TRUE

```r
filter(.data = gapminder, country == "Denmark")
```

country	continent	year
Afghanistan	Asia	1952
Afghanistan	Asia	1957
Afghanistan	Asia	1962
Afghanistan	Asia	1967
Afghanistan	Asia	1972
...	...	...

country	continent	year
Denmark	Europe	1952
Denmark	Europe	1957
Denmark	Europe	1962
Denmark	Europe	1967
Denmark	Europe	1972
Denmark	Europe	1977

filter()

```
filter(.data = gapminder,
       country ==
"Denmark")
```

One = sets an argument

Two == tests if equal

returns TRUE or FALSE)

Logical tests

Test	Meaning	Test	Meaning
`x < y`	Less than	`x %in% y`	In (group membership)
`x > y`	Greater than	`is.na(x)`	Is missing
`==`	Equal to	`!is.na(x)`	Is not missing
`x <= y`	Less than or equal to		
`x >= y`	Greater than or equal to		
`x != y`	Not equal to		

Use `filter()` and logical tests to show...

1. The data for Canada
2. All data for countries in Oceania
3. Rows where the life expectancy is greater than 82

Your turn!!

Solution

```r
filter(gapminder, country == "Canada")
```

```r
filter(gapminder, continent == "Oceania")
```

```r
filter(gapminder, lifeExp > 82)
```

filter() with multiple conditions

Extract rows that meet *every* test

```
filter(gapminder, country == "Denmark", year > 2000)
```

```r
filter(gapminder, country == "Denmark", year > 2000)
```

country	continent	year
Afghanistan	Asia	1952
Afghanistan	Asia	1957
Afghanistan	Asia	1962
Afghanistan	Asia	1967
Afghanistan	Asia	1972
...	...	...

country	continent	year
Denmark	Europe	2002
Denmark	Europe	2007

Boolean operators

Operator	Meaning
a & b	and
a | b	or
!a	not

These do the same thing:

```
filter(gapminder, country == "Denmark", year > 2000)
```

```
filter(gapminder, country == "Denmark" & year > 2000)
```

Use `filter()` and Boolean logical tests to show...

1. Canada before 1970
2. Countries where life expectancy in 2007 is below 50
3. Countries where life expectancy in 2007 is below 50 and are not in Africa

Your turn #2

Solution

```
filter(gapminder, country == "Canada", year < 1970)
```

```
filter(gapminder, year == 2007, lifeExp < 50)
```

```
filter(gapminder, year == 2007, lifeExp < 50,
       continent != "Africa")
```

mutate()

Create new columns

```
mutate(.data, ...)
```

- **`DATA`** = Data frame to transform
- **`...`** = Columns to make

```r
mutate(gapminder, gdp = gdpPercap * pop)
```

country	year	gdpPercap	pop
Afghanistan	1952	779.4453145	8425333
Afghanistan	1957	820.8530296	9240934
Afghanistan	1962	853.10071	10267083
Afghanistan	1967	836.1971382	11537966
Afghanistan	1972	739.9811058	13079460
...	...	...	...

country	year	...	gdp
Afghanistan	1952	...	6567086330
Afghanistan	1957	...	7585448670
Afghanistan	1962	...	8758855797
Afghanistan	1967	...	9648014150
Afghanistan	1972	...	9678553274
Afghanistan	1977	...	11697659231

```r
mutate(gapminder, gdp = gdpPercap * pop,
                  pop_mil = round(pop / 1000000))
```

country	year	gdpPercap	pop
Afghanistan	1952	779.4453145	8425333
Afghanistan	1957	820.8530296	9240934
Afghanistan	1962	853.10071	10267083
Afghanistan	1967	836.1971382	11537966
Afghanistan	1972	739.9811058	13079460
...	...	...	...

country	year	...	gdp	pop_mil
Afghanistan	1952	...	6567086330	8
Afghanistan	1957	...	7585448670	9
Afghanistan	1962	...	8758855797	10
Afghanistan	1967	...	9648014150	12
Afghanistan	1972	...	9678553274	13
Afghanistan	1977	...	11697659231	15

ifelse()

Do conditional tests within `mutate()`

```
ifelse(TEST,
       VALUE_IF_TRUE,
       VALUE_IF_FALSE)
```

- **`TEST`** = A logical test
- **`VALUE_IF_TRUE`** = What happens if test is true
- **`VALUE_IF_FALSE`** = What happens if test is false

```r
mutate(gapminder,
       after_1960 = ifelse(year > 1960, TRUE, FALSE))
```

```r
mutate(gapminder,
       after_1960 = ifelse(year > 1960,
                           "After 1960",
                           "Before 1960"))
```

Use `mutate()` to...

1. Add an `africa` column that is TRUE if the country is on the African continent
2. Add a column for logged GDP per capita (hint: use `log()`)
3. Add an `africa_asia` column that says “Africa or Asia” if the country is in Africa or Asia, and “Not Africa or Asia” if it’s not

Your turn

Solution

```
mutate(gapminder, africa = ifelse(continent == "Africa",
                                  TRUE, FALSE))
```

```
mutate(gapminder, log_gdpPercap = log(gdpPercap))
```

```
mutate(gapminder,
       africa_asia =
         ifelse(continent %in% c("Africa", "Asia"),
                "Africa or Asia",
                "Not Africa or Asia"))
```

Make a dataset for just 2002 *and* calculate logged GDP per capita

Make a dataset for just 2002 *and* calculate logged GDP per capita

Solution 1: Intermediate variables

```
gapminder_2002 <- filter(gapminder, year == 2002)

gapminder_2002_log <- mutate(gapminder_2002,
                             log_gdpPercap = log(gdpPercap))
```

Solution 2

Make a dataset for just 2002 *and* calculate logged GDP per capita

```
gapminder %>%
  filter(year == 2002) %>%
  mutate(log_gdpPercap = log(gdpPercap))
```

Pipe use

These do the same thing!

```
filter(gapminder, country == "Canada")
```

```
gapminder %>% filter(country == "Canada")
```

Pipe operator %>%

```
leave_house(get_dressed(get_out_of_bed(wake_up(me, time =
"8:00"), side = "correct"), pants = TRUE, shirt = TRUE), car
= TRUE, bike = FALSE)
```

```
me %>%
  wake_up(time = "8:00") %>%
  get_out_of_bed(side = "correct") %>%
  get_dressed(pants = TRUE, shirt = TRUE) %>%
  leave_house(car = TRUE, bike = FALSE)
```

summarize()

Compute a table of summaries

```
gapminder %>% summarize(mean_life = mean(lifeExp))
```

country	continent	year	lifeExp
Afghanistan	Asia	1952	28.801
Afghanistan	Asia	1957	30.332
Afghanistan	Asia	1962	31.997
Afghanistan	Asia	1967	34.02
...	...	...	...

mean_life
59.47444

```r
gapminder %>% summarize(mean_life = mean(lifeExp),
                        min_life = min(lifeExp))
```

country	continent	year	lifeExp
Afghanistan	Asia	1952	28.801
Afghanistan	Asia	1957	30.332
Afghanistan	Asia	1962	31.997
Afghanistan	Asia	1967	34.02
Afghanistan	Asia	1972	36.088
...	...	...	...

mean_life	min_life
59.47444	23.599

Use `summarize()` to calculate...

1. The first (minimum) year in the dataset
2. The last (maximum) year in the dataset
3. The number of rows in the dataset (use the cheatsheet)
4. The number of distinct countries in the dataset (use the cheatsheet)

Your turn

```r
gapminder %>%
  summarize(first = min(year),
            last = max(year),
            num_rows = n(),
            num_unique = n_distinct(country))
```

first	last	num_rows	num_unique
1952	2007	1704	142

Use `filter()` and `summarize()` to calculate (1) the number of unique countries and (2) the median life expectancy on the African continent in 2007

Your turn

Solution

```
gapminder %>%
  filter(continent == "Africa", year == 2007) %>%
  summarise(n_countries = n_distinct(country),
            med_le = median(lifeExp))
```

n_countries	med_le
52	52.9265

group_by()

Put rows into groups based on values in a column

```
gapminder %>% group_by(continent)
```

Nothing happens by itself!

Powerful when combined with `summarize()`

```
gapminder %>%
  group_by(continent) %>%
  summarize(n_countries = n_distinct(country))
```

continent	n_countries
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

Find the minimum, maximum, and median life expectancy for each continent

Find the minimum, maximum, and median life expectancy for each continent in 2007 only

Your turn

```r
gapminder %>%
  group_by(continent) %>%
  summarize(min_le = min(lifeExp),
            max_le = max(lifeExp),
            med_le = median(lifeExp))
```

```r
gapminder %>%
  filter(year == 2007) %>%
  group_by(continent) %>%
  summarize(min_le = min(lifeExp),
            max_le = max(lifeExp),
            med_le = median(lifeExp))
```

??