Transform data with dplyr

Your turn #0: Load data

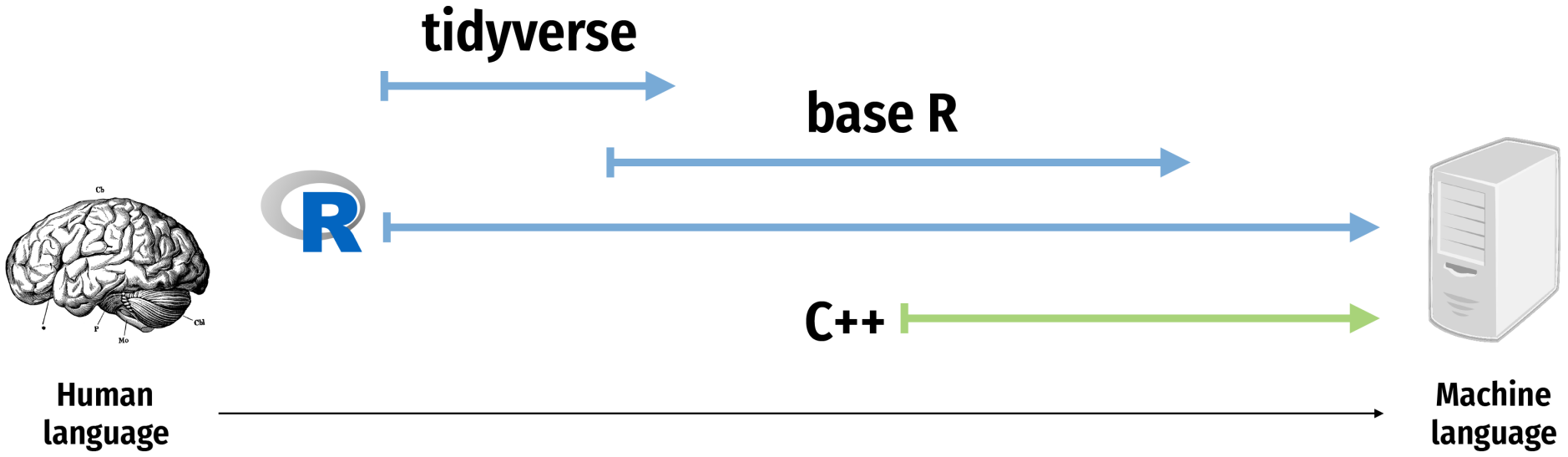
1. Run the setup chunk
2. Take a look at the `gapminder` data

02:00

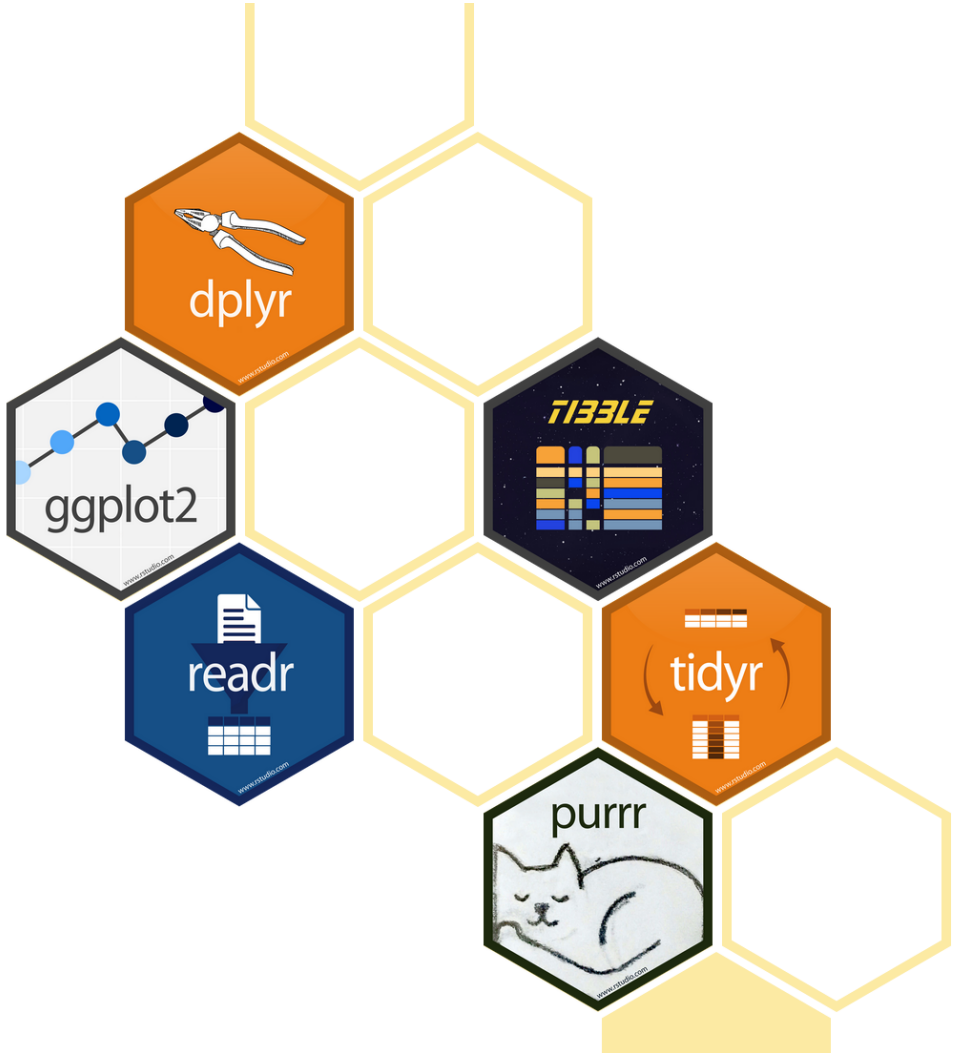
```
gapminder
```

```
## # A tibble: 1,704 × 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.
## # ℹ 1,694 more rows
```

The tidyverse

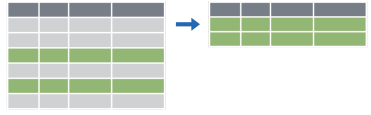


The tidyverse

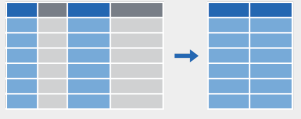


dplyr: verbs for manipulating data

Extract rows with `filter()`



Extract columns with `select()`

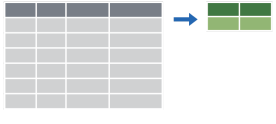


Arrange/sort rows with `arrange()`

Make new columns with `mutate()`

Make group summaries with

`group_by() |> summarize()`



filter()

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

```
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		

Your turn #1: Filtering

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

04:00

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

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

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

Common mistakes

Using = instead of ==

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

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

Quote use

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

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

`filter()` with multiple conditions

Extract rows that meet *every* test

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

```
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

Default is "and"

These do the same thing:

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

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

Your turn #2: Filtering

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

04:00

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

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

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

Common mistakes

Collapsing multiple tests into one

```
filter(gapminder, 1960 < year < 1980)
```

```
filter(gapminder,
       year > 1960, year < 1980)
```

Using multiple tests instead of `%in%`

```
filter(gapminder,
       country == "Mexico",
       country == "Canada",
       country == "United States")
```

```
filter(gapminder,
       country %in% c("Mexico", "Canada",
                      "United States"))
```

Common syntax

Every dplyr verb function follows the same pattern

First argument is a data frame; returns a data frame

```
VERB(DATA, ...)
```

- `VERB` = dplyr function/verb
- `DATA` = Data frame to transform
- `...` = Stuff the verb does

mutate()

Create new columns

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

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

```
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

```
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

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

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

Your turn #3: Mutating

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

05:00

```
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"))
```

What if you have multiple verbs?

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))
```

What if you have multiple verbs?

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

Solution 2: Nested functions

```
filter(mutate(gapminder_2002,
              log_gdpPercap = log(gdpPercap)),
       year == 2002)
```

What if you have multiple verbs?

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

Solution 3: Pipes!

The `|>` operator (pipe) takes an object on the left and passes it as the first argument of the function on the right

```
gapminder |> filter(_, country == "Canada")
```

What if you have multiple verbs?

These do the same thing!

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

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

What if you have multiple verbs?

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

Solution 3: Pipes!

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

|>

```
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)
```

|> VS %>%

There are actually multiple pipes!

%>% was invented first, but requires a package to use

|> is part of base R

They're interchangeable 99% of the time

(Just be consistent)

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

summarize()

```
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

Your turn #4: Summarizing

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)

04:00

```
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

Your turn #5: Summarizing

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

04:00

```
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

```
pollution |>
  summarize(mean = mean(amount), sum = sum(amount), n = n())
```

city	particle_size	amount
New York	Large	23
New York	Small	14
London	Large	22
London	Small	16
Beijing	Large	121
Beijing	Small	56

mean	sum	n
42	252	6

```
pollution |>
  group_by(city) |>
  summarize(mean = mean(amount), sum = sum(amount), n = n())
```

city	particle_size	amount
New York	Large	23
New York	Small	14
London	Large	22
London	Small	16
Beijing	Large	121
Beijing	Small	56

city	mean	sum	n
Beijing	88.5	177	2
London	19.0	38	2
New York	18.5	37	2

```
pollution |>
  group_by(particle_size) |>
  summarize(mean = mean(amount), sum = sum(amount), n = n())
```

city	particle_size	amount
New York	Large	23
New York	Small	14
London	Large	22
London	Small	16
Beijing	Large	121
Beijing	Small	56

particle_size	mean	sum	n
Large	55.33333	166	3
Small	28.66667	86	3

Your turn #6: Grouping and summarizing

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

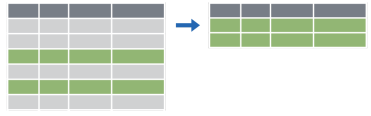
05:00

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

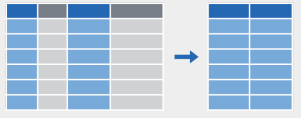
```
gapminder |>
  filter(year == 2007) |>
  group_by(continent) |>
  summarize(min_le = min(lifeExp),
            max_le = max(lifeExp),
            med_le = median(lifeExp))
```

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()`