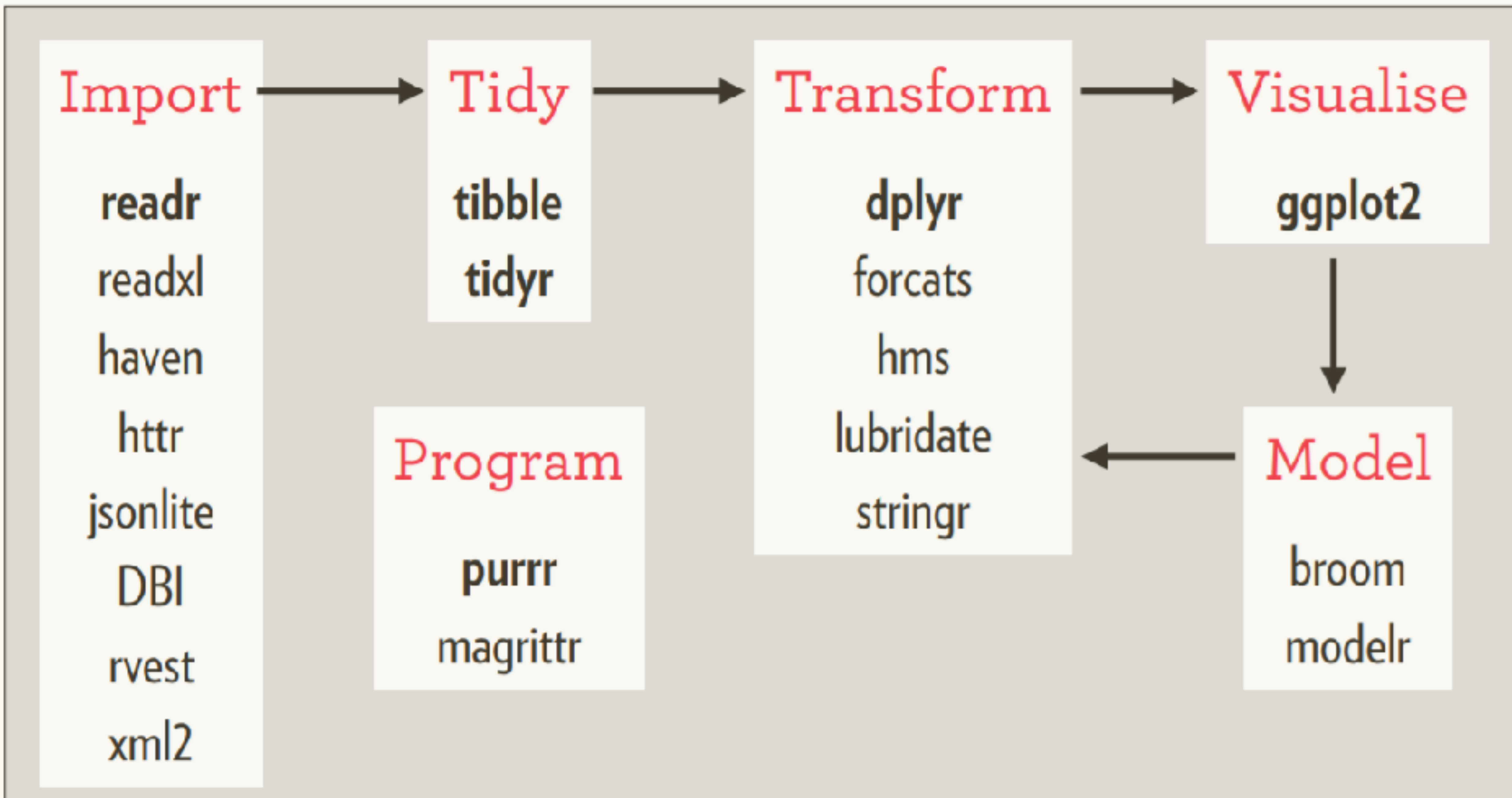
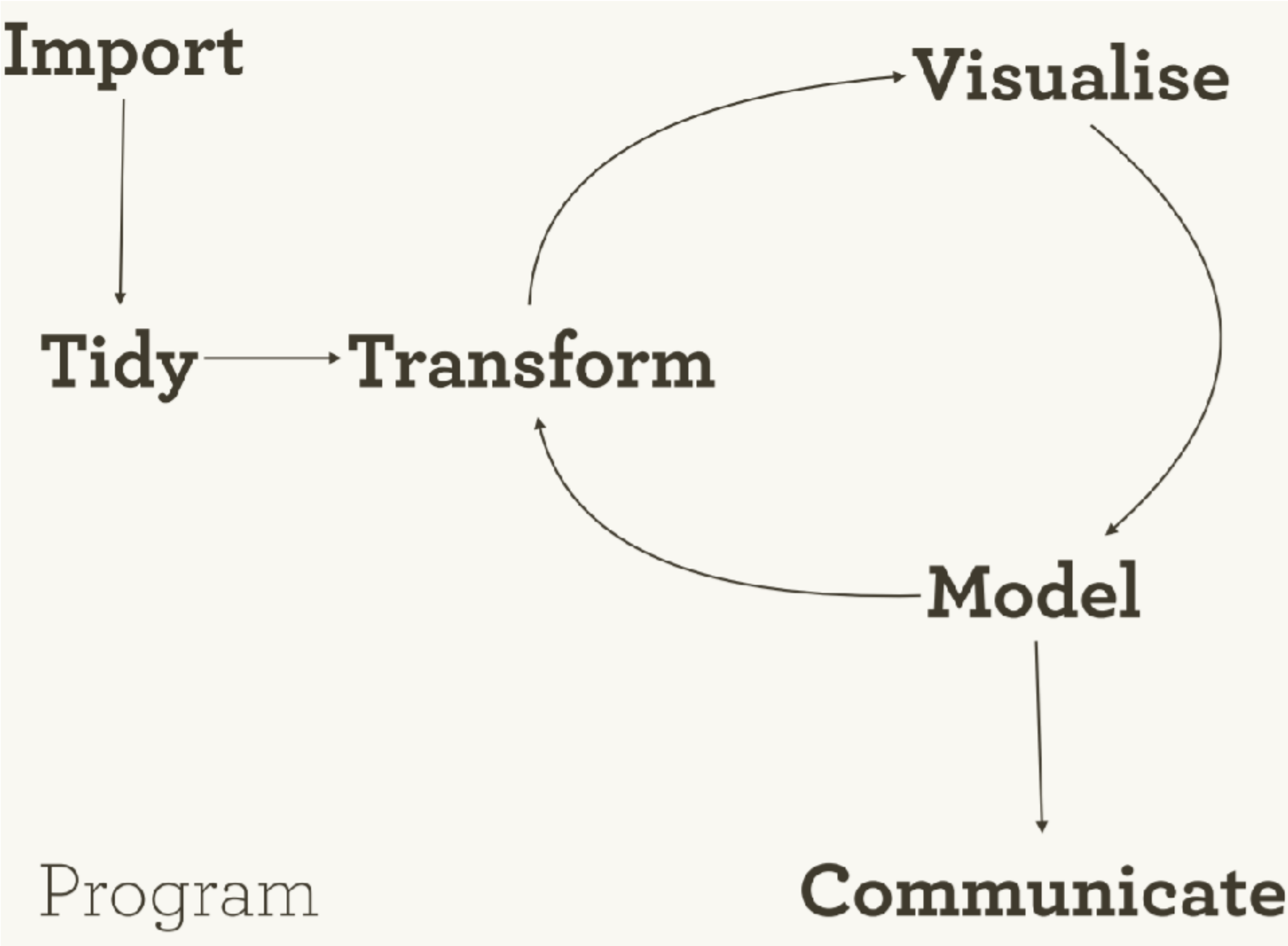
The tidyverse

ARGH meeting
Will Hall


Hadley Wickham
@hadleywickham

tidyverse


Import
readr
readxl
haven
httr
jsonlite
DBI
rvest
xml2
Tidy
tibble
tidyr
Transform
dplyr
forcats
hms
lubridate
stringr
Visualise
ggplot2
Model
broom
modelr
Program
purrr
magrittr


Import
Visualise
Tidy
Transform
Model
Program
Communicate

dplyr

Big Data Borat
@BigDataBorat

In Data Science, 80% of time spent prepare data, 20% of time spent complain about need for prepare data.

Π **Stat Fact**
@StatFact

Data cleaning code cannot be clean. It's a sort of sin eater.

Transform

dplyr
forcats
hms
lubridate
stringr

A language for data manipulation

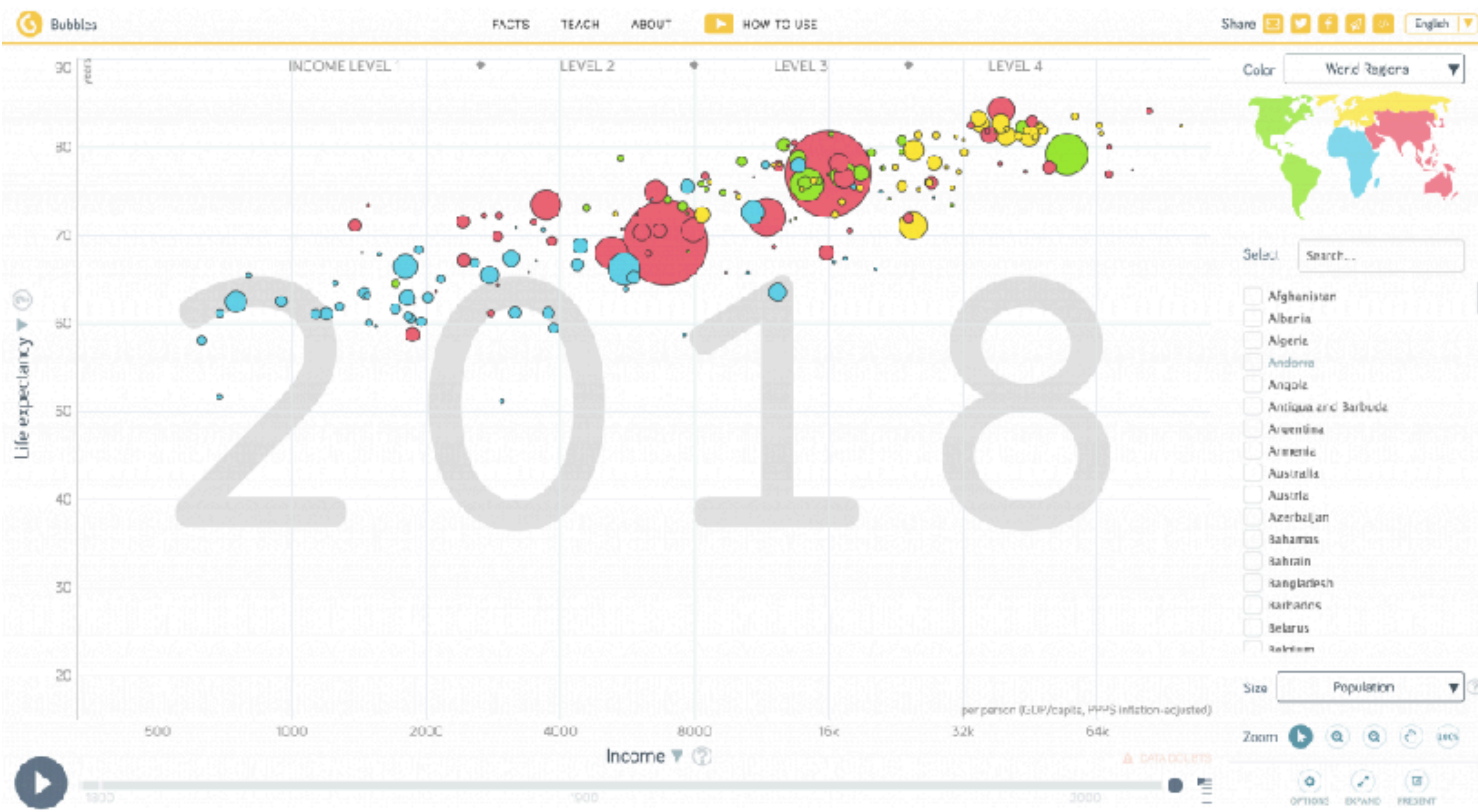
- Six functions for data manipulation:
  - select()
  - filter()
  - summarize()
  - group_by()
  - mutate()
  - arrange()
- These functions provide the verbs for a language of data manipulation.

Operating principles

- Each function uses consistent principles:
  - The first argument is a data frame.
  - The subsequent arguments describe what to do with the data frame.
  - Each function returns a data frame.

gapminder package

- Provides an excerpt from the Gapminder data.



```
library(gapminder)
```

> 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.
```

```
library(tidyverse)
```

select()

- Lets you extract a subset of **columns** from a data frame

data frame **column names**

- select(gapminder, year, country, gdpPercap) 😀
- gapminder[c('year', 'country', 'gdpPercap')] 😰
- Select all but one variable:
  - select(gapminder, -year)

filter()

- Lets you extract a subset of **rows** from a data frame.

- filter(gapminder, continent == “Europe")

- Filtering a data frame and then selecting specific variables:

  “then”

  - filter(gapminder, continent == "Europe") **%>%**
    select(year, country, gdpPercap)

```
# A tibble: 360 x 3
    year country gdpPercap
   <int> <fct>       <dbl>
 1  1952 Albania     1601.
 2  1957 Albania     1942.
 3  1962 Albania     2313.
 4  1967 Albania     2760.
 5  1972 Albania     3313.
 6  1977 Albania     3533.
 7  1982 Albania     3631.
 8  1987 Albania     3739.
 9  1992 Albania     2497.
10  1997 Albania     3193.
# … with 350 more rows
```

%>%

CTRL + SHIFT + M (or CMD + SHIFT + M for OSX)

```r
filter(gapminder, continent == "Europe") %>%
    select(year, country, gdpPercap)
```

```r
gapminder %>%
  filter(continent=="Europe") %>%
  select(year, country, gdpPercap)
```

Challenge 1

- Use filter and select to produce a data frame that has only the columns lifeExp, country and year for the countries of Africa. How many rows does your data frame have?

Challenge 1 solution

```
gapminder %>%
    filter(continent =="Africa") %>%
    select(year, country, lifeExp)
```

summarize()

- Collapse a data frame down to a summary statistic.

- summarize(gapminder,
    mean_gdpPercap = mean(gdpPercap))

New variable name **Summary operation**

- gapminder %>%
    summarize(mean_gdpPercap = mean(gdpPercap))

- What if we wanted the mean gdpPercap for each continent?

group_by()

- Adds an grouping structure to a data frame
- Subsequent functions operate on each group.
- group_by(gapminder, continent)

```
# A tibble: 1,704 x 6
# Groups:   continent [5]
  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.
```

group_by()

- gapminder %>%
    group_by(continent) %>%
    summarize(mean_gdpPercap = mean(gdpPercap))

```
# A tibble: 5 x 2
  continent mean_gdpPercap
  <fct>              <dbl>
1 Africa             2194.
2 Americas           7136.
3 Asia               7902.
4 Europe            14469.
5 Oceania           18622.
```

```
gapminder %>%
  group_by(continent, year) %>%
  summarize(mean_gdpPercap = mean(gdpPercap),
            sd_gdpPercap = sd(gdpPercap),
            mean_pop = mean(pop),
            sd_pop = sd(pop))
```

```
# A tibble: 60 x 6
# Groups:   continent [?]
  continent  year mean_gdpPercap sd_gdpPercap mean_pop    sd_pop
  <fct>     <int>          <dbl>        <dbl>    <dbl>     <dbl>
1 Africa     1952          1253.         983.  4570010.  6317450.
2 Africa     1957          1385.        1135.  5093033.  7076042.
3 Africa     1962          1598.        1462.  5702247.  7957545.
4 Africa     1967          2050.        2848.  6447875.  8985505.
5 Africa     1972          2340.        3287.  7305376. 10130833.
6 Africa     1977          2586.        4142.  8328097. 11585184.
7 Africa     1982          2482.        3243.  9602857. 13456243.
8 Africa     1987          2283.        2567. 11054502. 15277484.
9 Africa     1992          2282.        2644. 12674645. 17562719.
```

Challenge 2

- Calculate the average life expectancy for each country in the gapminder data frame.

- Bonus: Use the arrange() function to find out which country has the highest life expectancy?

Challenge 2 solution

```
gapminder %>%
  group_by(country) %>%
  summarize(mean_lifeExp = mean(lifeExp)) %>%
  arrange(desc(mean_lifeExp))
```

mutate()

- Create new variables.

- mutate(gapminder,
    gdp_billion = gdpPercap*pop/10^9)

- gapminder %>%
    mutate(gdp_billion = gdpPercap*pop/10^9)

```
A tibble: 1,704 x 7
  country     continent  year lifeExp      pop gdpPercap gdp_billion
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>       <dbl>
  Afghanistan Asia       1952    28.8  8425333      779.        6.57
  Afghanistan Asia       1957    30.3  9240934      821.        7.59
  Afghanistan Asia       1962    32.0 10267083      853.        8.76
  Afghanistan Asia       1967    34.0 11537966      836.        9.65
  Afghanistan Asia       1972    36.1 13079460      740.        9.68
  Afghanistan Asia       1977    38.4 14880372      786.       11.7
  Afghanistan Asia       1982    39.9 12881816      978.       12.6
  Afghanistan Asia       1987    40.8 13867957      852.       11.8
  Afghanistan Asia       1992    41.7 16317921      649.       10.6
  Afghanistan Asia       1997    41.8 22227415      635.       14.1
… with 1,694 more rows
```

```
gapminder %>%
   mutate(gdp_billion=gdpPercap*pop/10^9) %>%
   group_by(continent, year) %>%
   summarize(mean_gdp_billion = mean(gdp_billion),
                            sd_gdp_billion = sd(gdp_billion))
```

```
# A tibble: 60 x 4
# Groups:   continent [?]
   continent  year mean_gdp_billion sd_gdp_billion
   <fct>     <int>            <dbl>          <dbl>
 1 Africa     1952             5.99           11.4
 2 Africa     1957             7.36           14.5
 3 Africa     1962             8.78           17.2
 4 Africa     1967            11.4            23.2
 5 Africa     1972            15.1            30.4
 6 Africa     1977            18.7            38.1
 7 Africa     1982            22.0            46.6
 8 Africa     1987            24.1            51.4
 9 Africa     1992            26.3            55.1
10 Africa     1997            30.0            63.0
```

Advanced Challenge

- Calculate the average life expectancy in 2002 of 2 randomly selected countries from each continent. Then arrange this data so that it is ordered by mean lifeExp (highest to lowest).

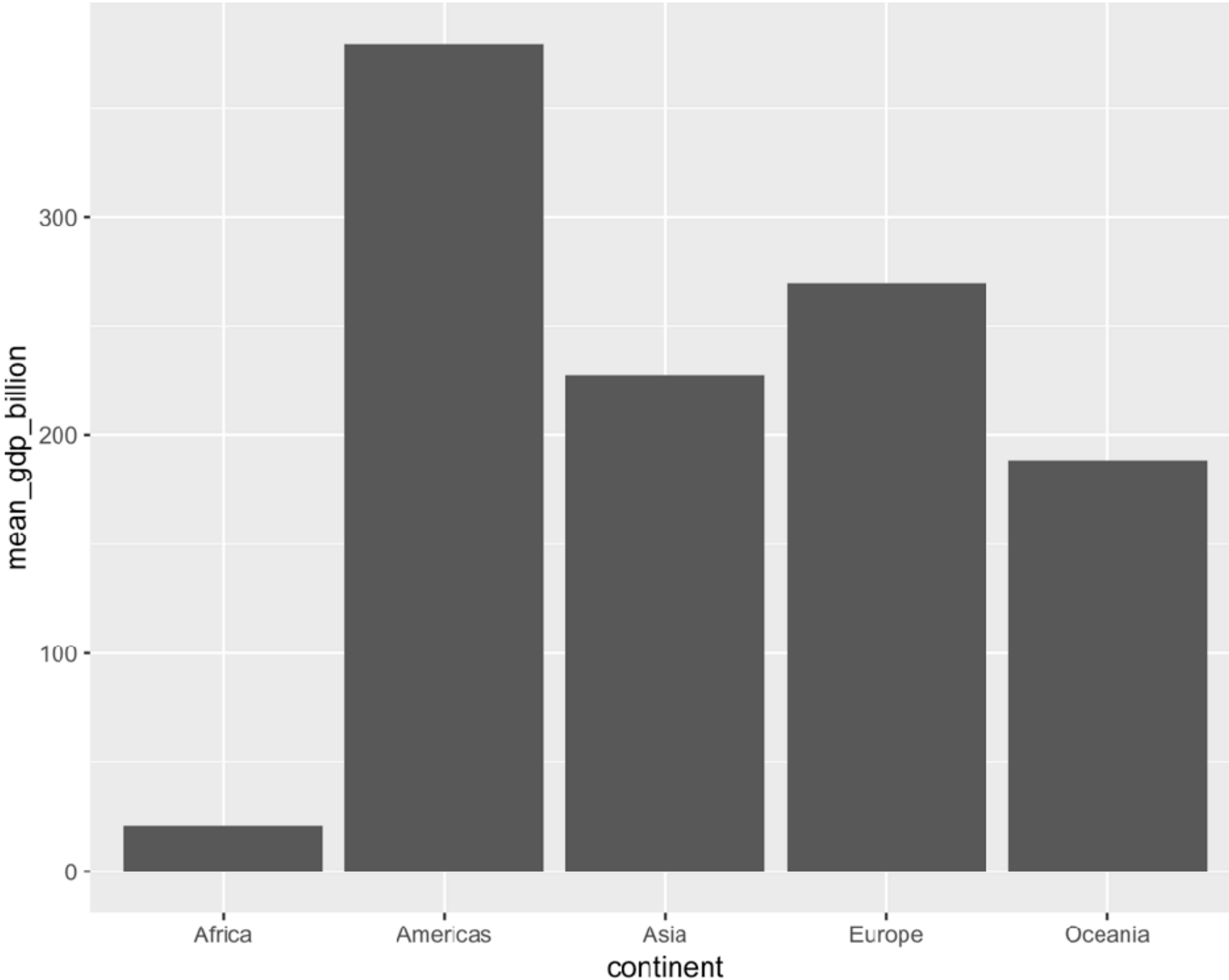
- Hint: Use the functions sample_n() and arrange(); they have similar syntax to other dplyr functions.

Advanced Challenge Solution

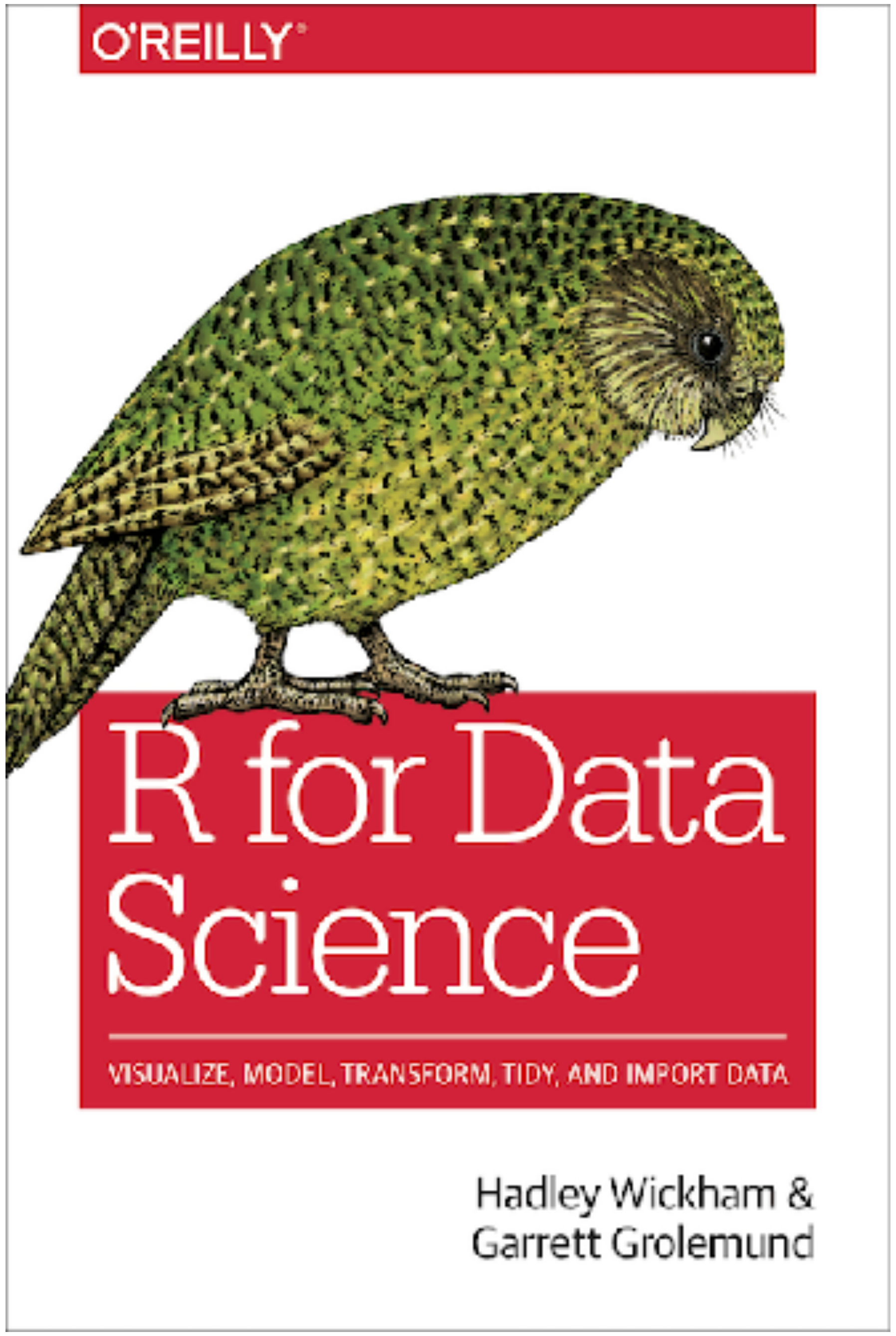
```
gapminder %>%
  filter(year==2002) %>%
  group_by(continent) %>%
  sample_n(2) %>%
  summarize(mean_lifeExp=mean(lifeExp)) %>%
  arrange(desc(mean_lifeExp))
```

Where you can go

```
gapminder %>%
  mutate(gdp_billion=gdpPercap*pop/10^9) %>%
  group_by(continent) %>%
  summarize(mean_gdp_billion=mean(gdp_billion)) %>%
  ggplot(aes(x=continent, y = mean_gdp_billion)) +
  geom_col()
```


mean_gdp_billion
300
200
100
0
Africa
Americas
Asia
Europe
Oceania
continent

Learning more



http://r4ds.had.co.nz

Journal of Statistical Software
MMMMMM YYYY, Volume VV, Issue II. http://www.jstatsoft.org/

Tidy Data

Hadley Wickham
RStudio

Abstract

A huge amount of effort is spent cleaning data to get it ready for analysis, but there has been little research on how to make data cleaning as easy and effective as possible. This paper tackles a small, but important, component of data cleaning: data tidying. Tidy datasets are easy to manipulate, model and visualise, and have a specific structure: each variable is a column, each observation is a row, and each type of observational unit is a table. This framework makes it easy to tidy messy datasets because only a small set of tools are needed to deal with a wide range of un-tidy datasets. This structure also makes it easier to develop tidy tools for data analysis, tools that both input and output tidy datasets. The advantages of a consistent data structure and matching tools are demonstrated with a case study free from mundane data manipulation chores.

http://rpubs.com/aelhabr/tidyverse-basics

People to follow on twitter

- Hadley Wickham: @hadleywickham
- Jenny Bryan: @JennyBryan
- David Robinson: @drob
- Mara Averick: @dataandme
- Julia Silge: @juliasilge

Thanks for listening!