PSY 503: Foundations of Statistical Methods in Psychological Science

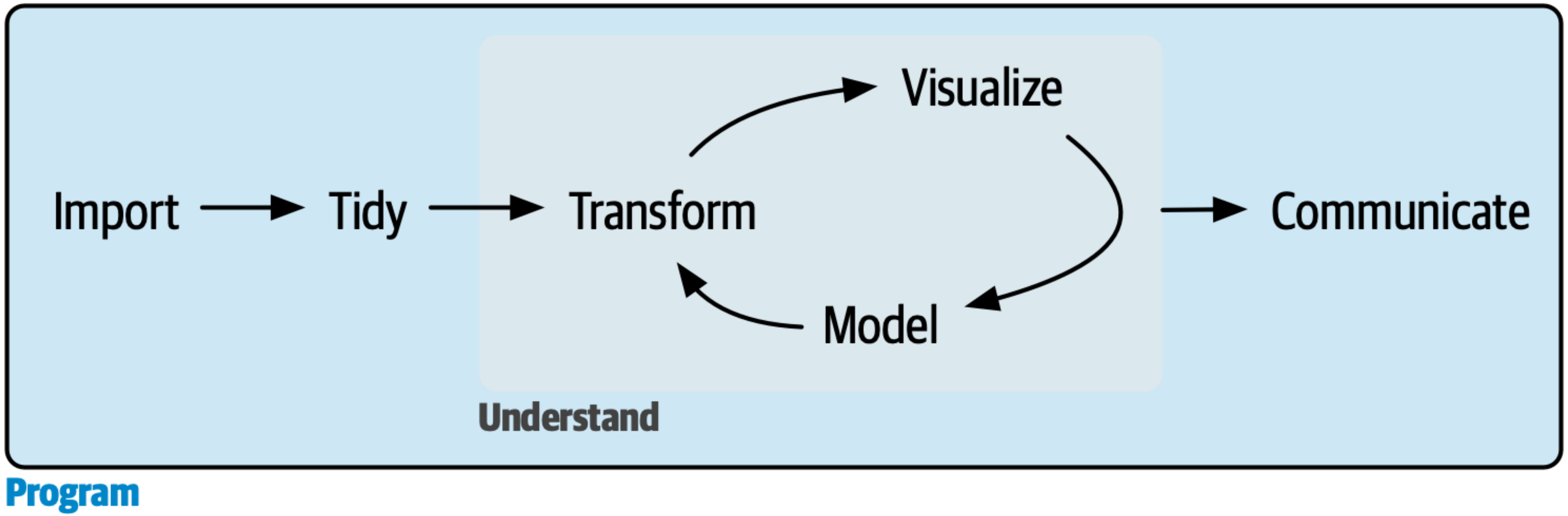
Lab 03

Data Wrangling

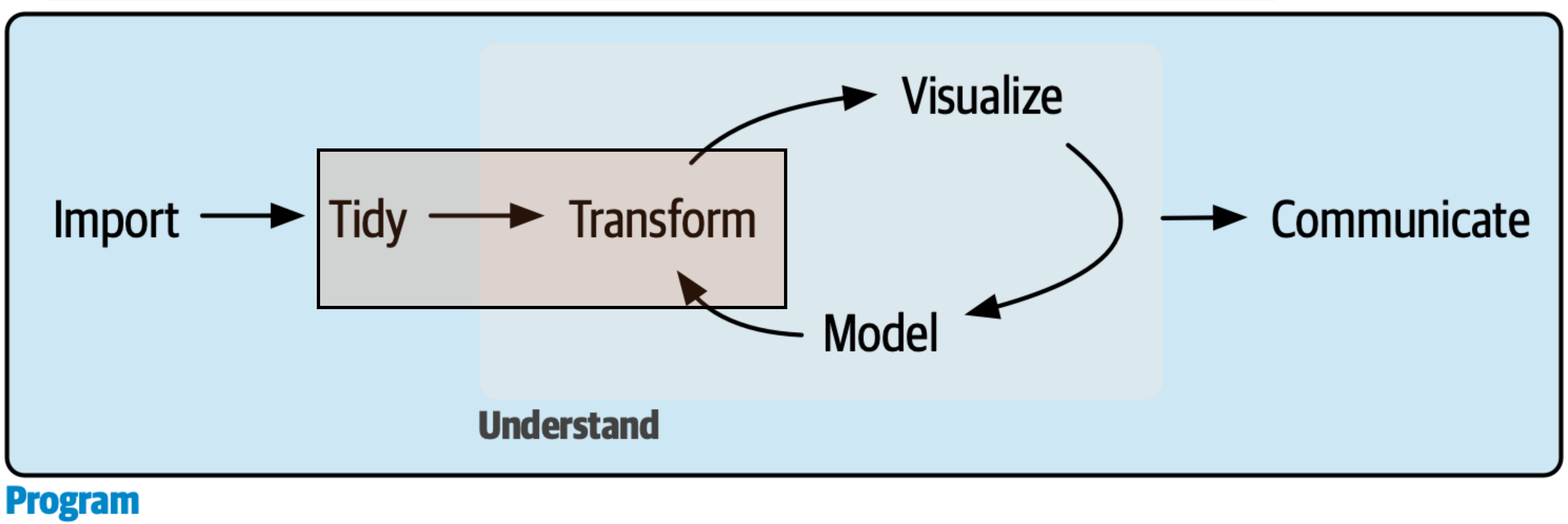
Suyog Chandramouli

Zoom & 311 PSH (Princeton University)

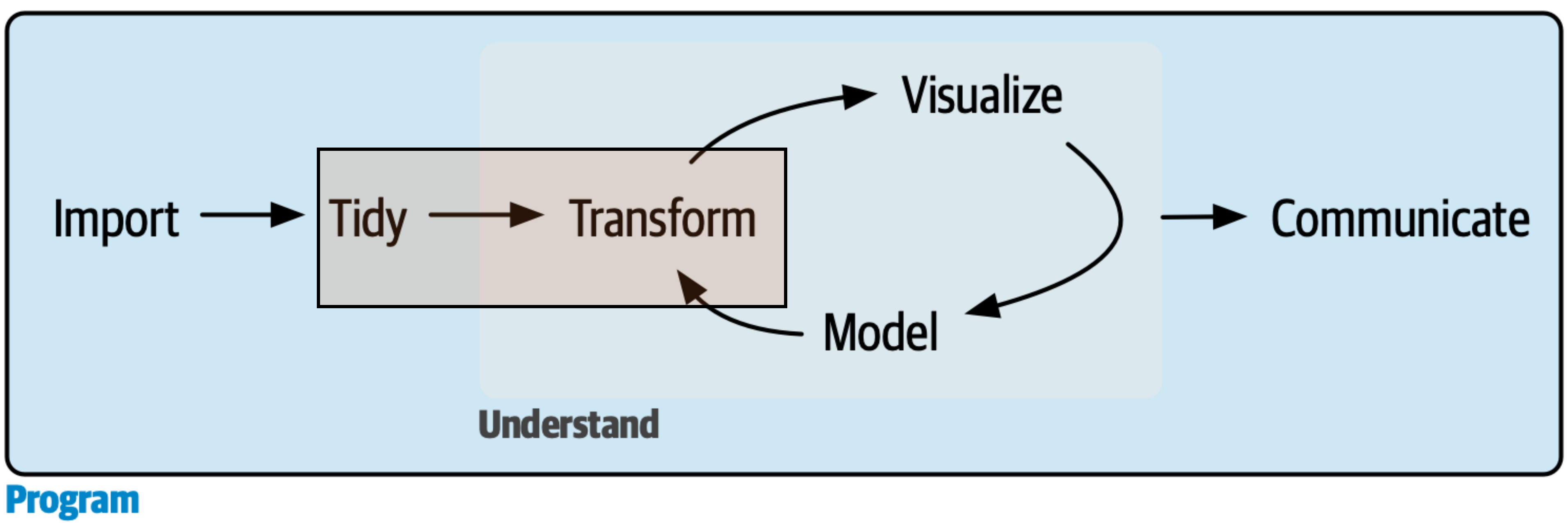
24th September, 2025



R for Data Science (2e)

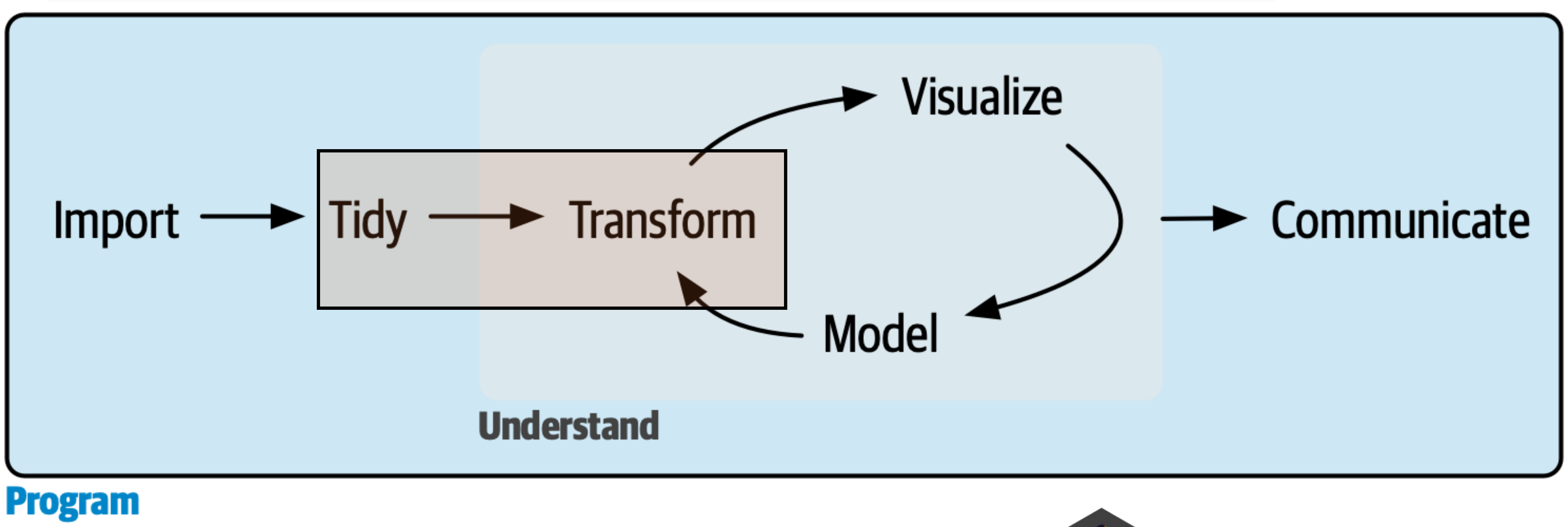


R for Data Science (2e)



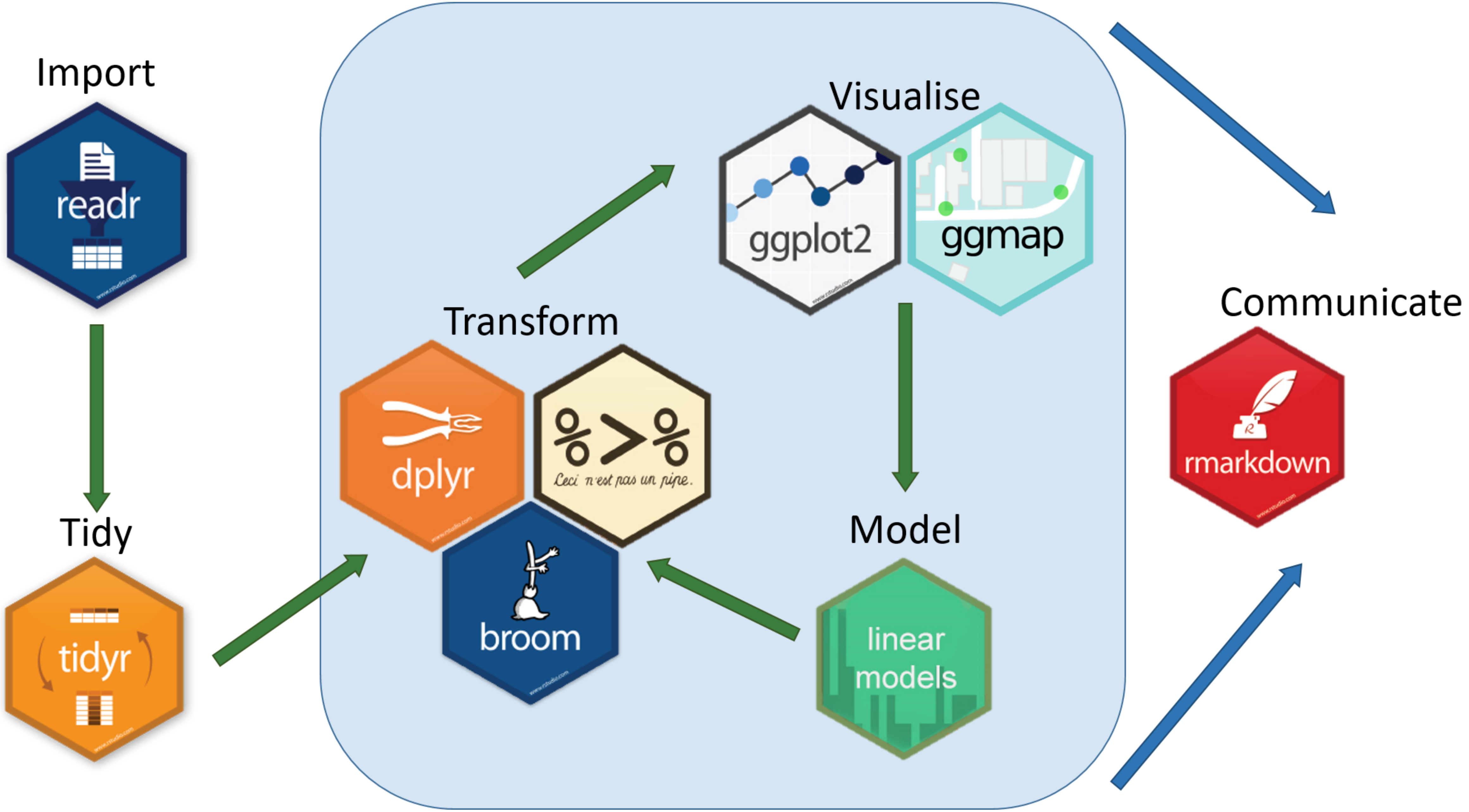
Data wrangling is the process of cleaning, structuring, and enriching raw data.

R for Data Science (2e)



Data wrangling is the process of cleaning, structuring, and enriching raw data.

R for Data Science (2e)



Import

Tidy

Transform

Visualise

Model

Communicate

Cheatsheets are your friends

- https://rstudio.github.io/cheatsheets/html/data-transformation.html

- https://rstudio.github.io/cheatsheets/html/tidyr.html

- https://rstudio.github.io/cheatsheets/html/data-visualization.html

Outline

- Gapminder dataset
- Data
    - Dataframe Structure
    - Examining data
    - Working with factors
- Pipes %>%

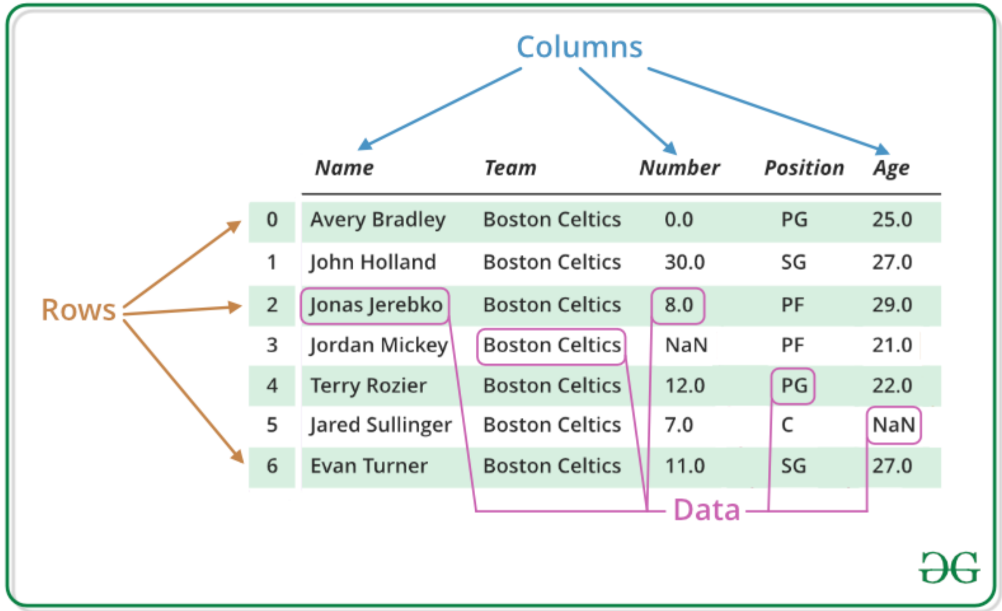
- Tidyverse verbs (operations on Data)
    - filter
    - arrange
    - select
    - mutate
    - bind
    - summarize
    - pivot
    - join

Dataframes

- Structure for handling “rectangular”/ tabular / spreadsheet-like data
  - Holds similar data together in a column
  - Data types can change across columns (unlike with matrices)
  - Great as a standard

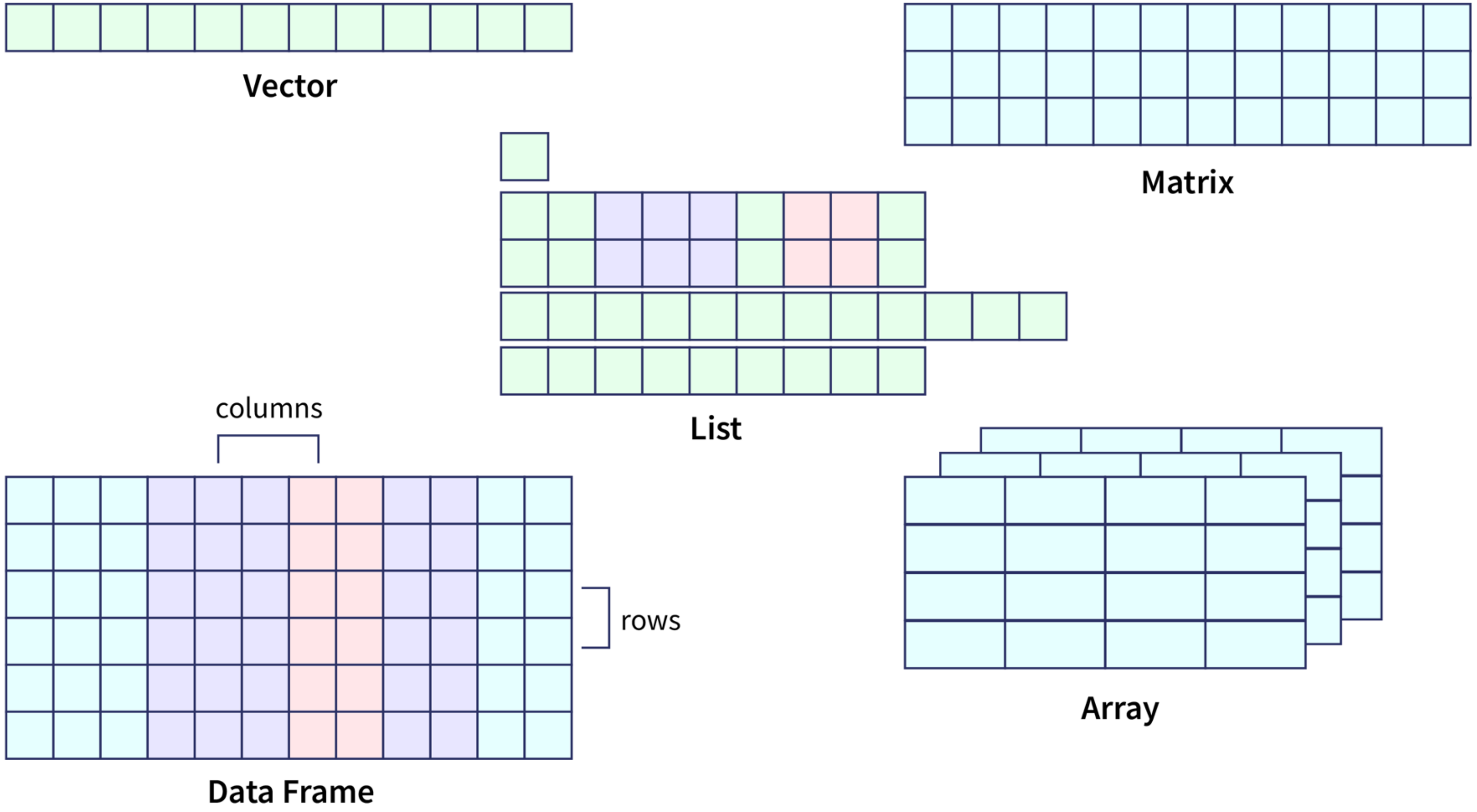
- Works great with R functions for analysis & visualization
  - Works well with R’s vectorized nature
  - A whole universe of tools for working with

[inspiration for Pandas, in Python]



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

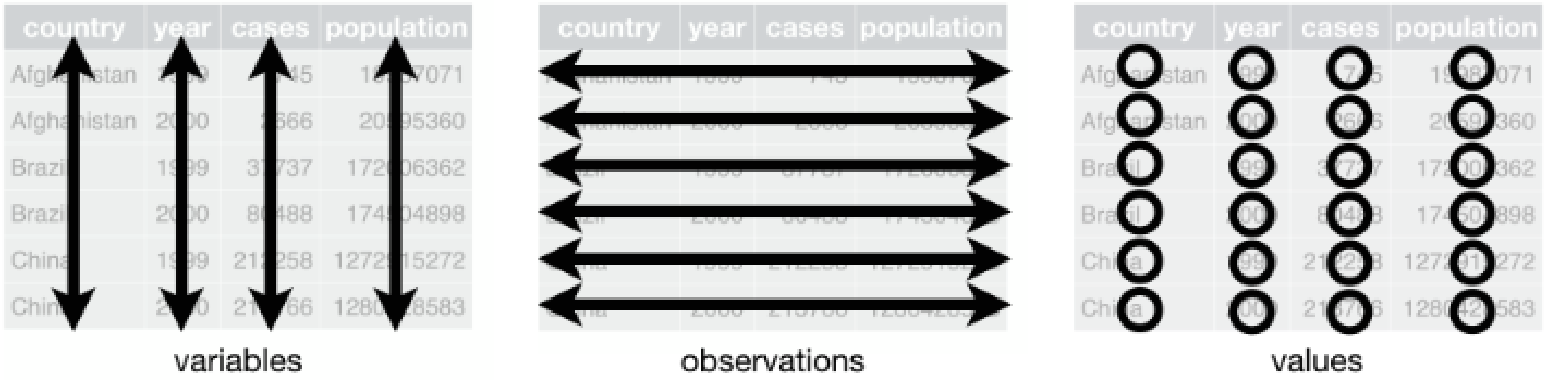
Data Structures

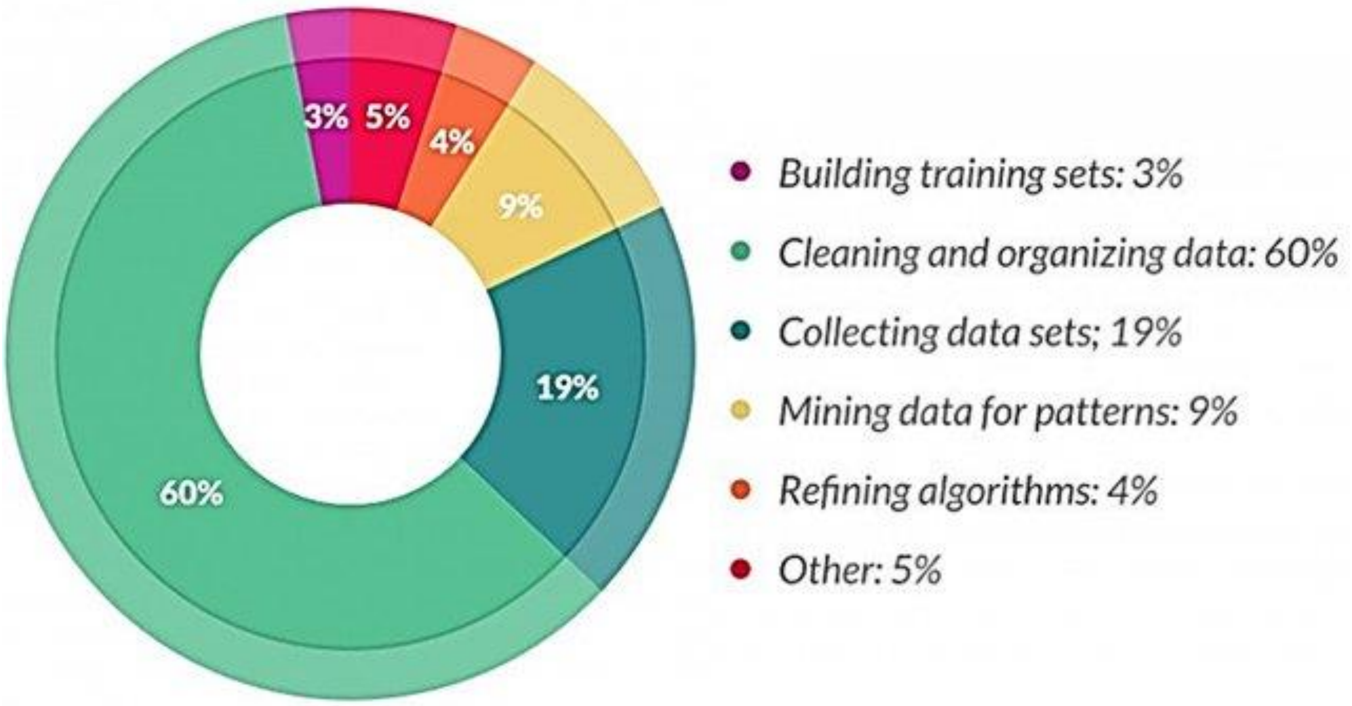


Lists are pervasive, btw.

- JSON, XML, web APIs
- String processing (e.g. when strings are split)
- A lot of base R

“Tidy datasets are all alike, but every messy dataset is messy in its own way.”
— Hadley Wickham





What data scientists spend the most time doing.

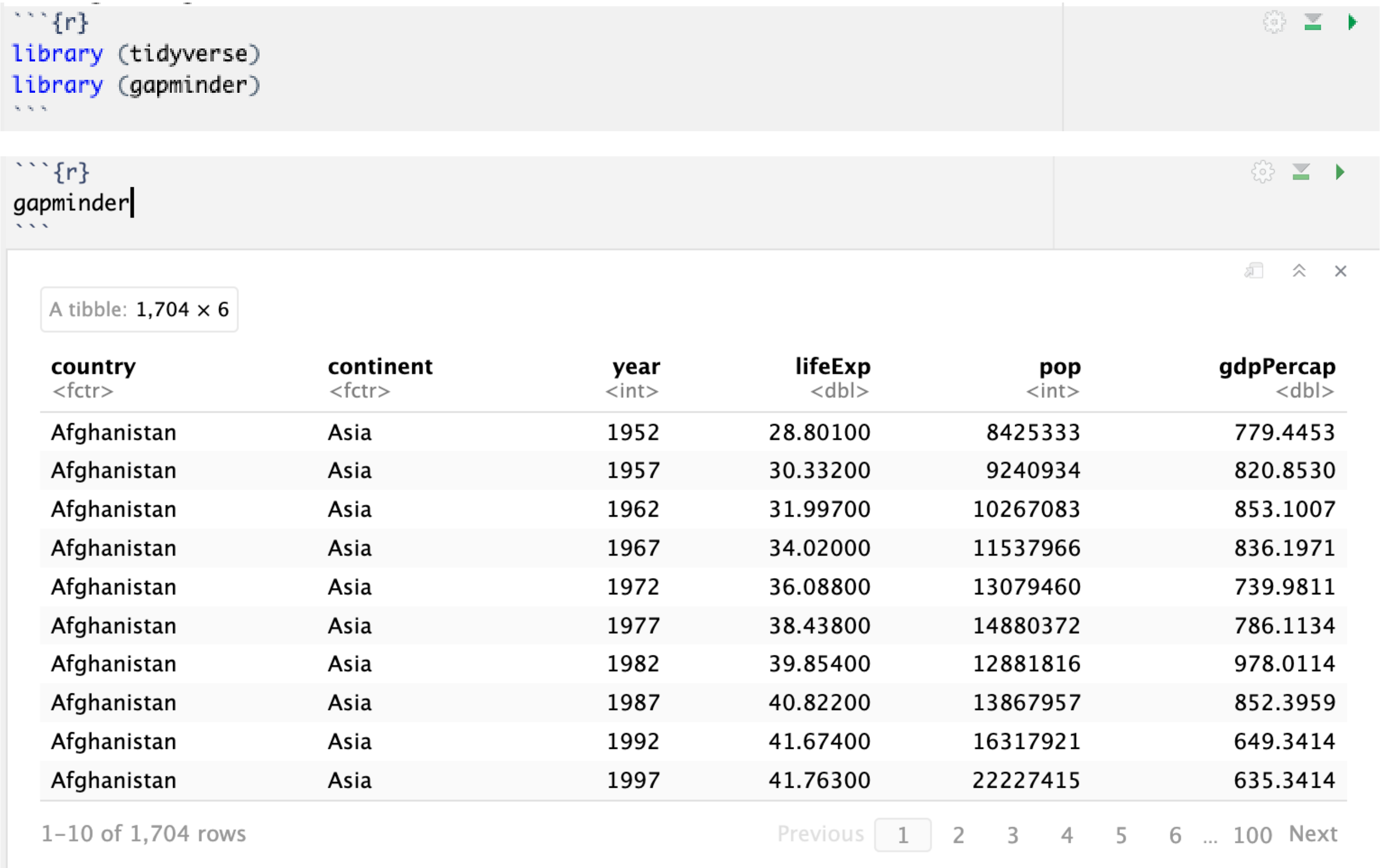
Gapminder data

- From the gapminder project
    - “Gapminder identifies systematic misconceptions about important global trends and proportions and uses reliable data to develop easy to understand teaching materials to rid people of their misconceptions.”

- Available in the gapminder package

→ Demo

Gapminder



```{r}
library (tidyverse)
library (gapminder)
```

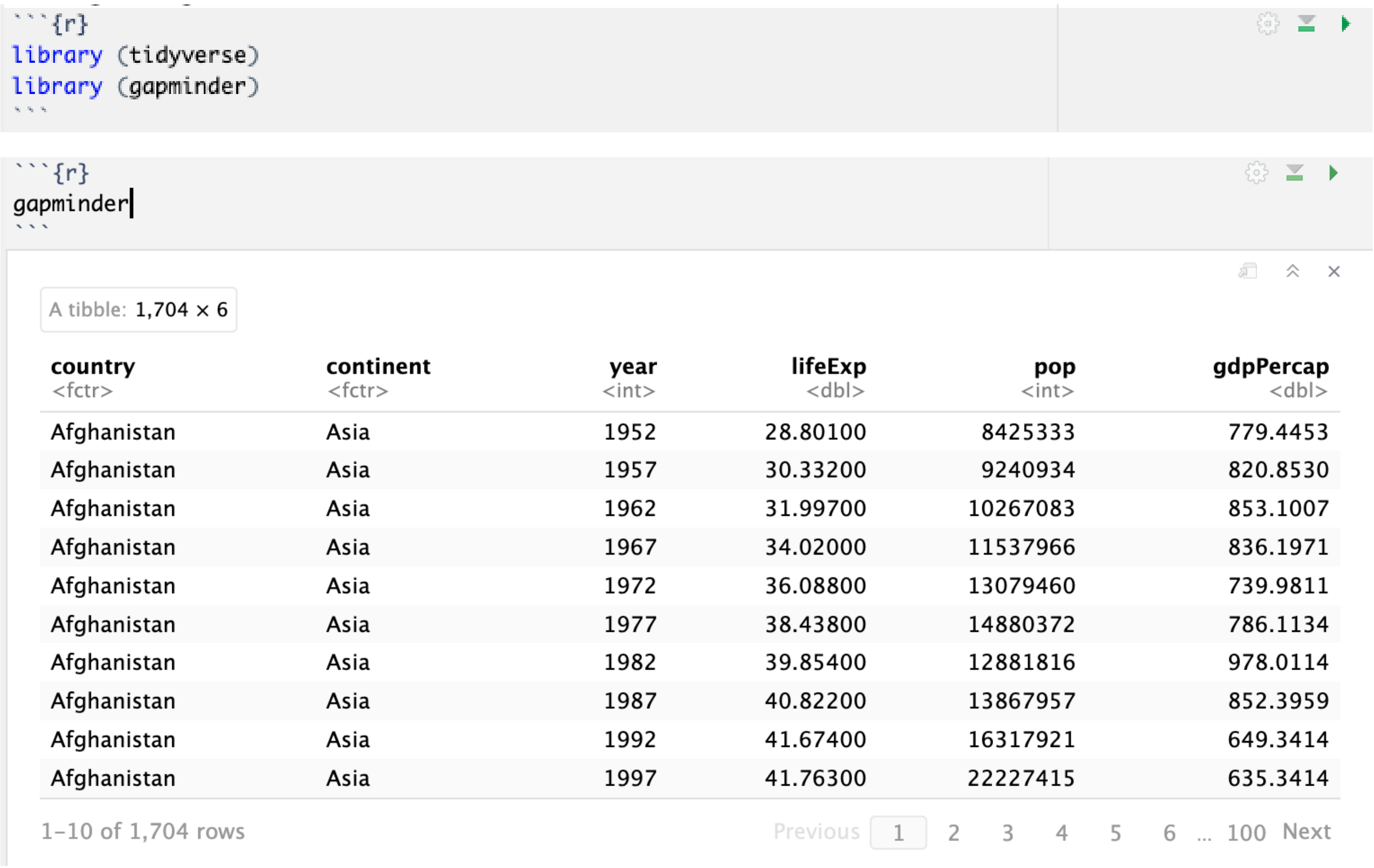
```{r}
gapminder
```

A tibble: 1,704 × 6

country	continent	year	lifeExp	pop	gdpPercap
<fctr>	<fctr>	<int>	<dbl>	<int>	<dbl>
Afghanistan	Asia	1952	28.80100	8425333	779.4453
Afghanistan	Asia	1957	30.33200	9240934	820.8530
Afghanistan	Asia	1962	31.99700	10267083	853.1007
Afghanistan	Asia	1967	34.02000	11537966	836.1971
Afghanistan	Asia	1972	36.08800	13079460	739.9811
Afghanistan	Asia	1977	38.43800	14880372	786.1134
Afghanistan	Asia	1982	39.85400	12881816	978.0114
Afghanistan	Asia	1987	40.82200	13867957	852.3959
Afghanistan	Asia	1992	41.67400	16317921	649.3414
Afghanistan	Asia	1997	41.76300	22227415	635.3414

1–10 of 1,704 rows

Gapminder



```{r}
library (tidyverse)
library (gapminder)
```

```{r}
gapminder
```

A tibble: 1,704 × 6

country <fctr>	continent <fctr>	year <int>	lifeExp <dbl>	pop <int>	gdpPercap <dbl>
Afghanistan	Asia	1952	28.80100	8425333	779.4453
Afghanistan	Asia	1957	30.33200	9240934	820.8530
Afghanistan	Asia	1962	31.99700	10267083	853.1007
Afghanistan	Asia	1967	34.02000	11537966	836.1971
Afghanistan	Asia	1972	36.08800	13079460	739.9811
Afghanistan	Asia	1977	38.43800	14880372	786.1134
Afghanistan	Asia	1982	39.85400	12881816	978.0114
Afghanistan	Asia	1987	40.82200	13867957	852.3959
Afghanistan	Asia	1992	41.67400	16317921	649.3414
Afghanistan	Asia	1997	41.76300	22227415	635.3414

1–10 of 1,704 rows

Tibble vs dataframe

- Tidyverse’s user-friendly implementation of dataframes
- Essentially the same
- Some differences
  - Printing
    - is more aesthetically pleasing; shows only first few rows and columns
  - Subsetting
    - a tibble is always returned
  - ..

Nouns, and verbs

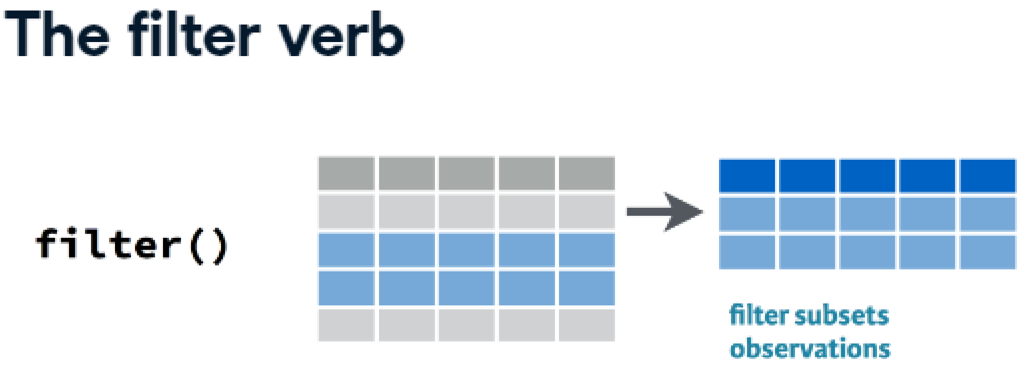
- If you consider a line of code to be a sentence..
    - Data (tibble, dataframes) are the “nouns”
        - often the first argument in function calls.
    - dplyr functions are the “verbs” that act on data
        - rename ()
        - filter ()
        - select ()
        - arrange ()
        - mutate ()
        - summarise ()
        - group_by ()
        - ….

2. Filtering

- To keep rows that satisfy a condition
- Condition operators:
  - ==, !=, <,>, >=, <=, %in%
- Usecase:
  - Inspect subsets of data (based on a condition)
  - Use when you care about a portion of the dataset

2. Filtering

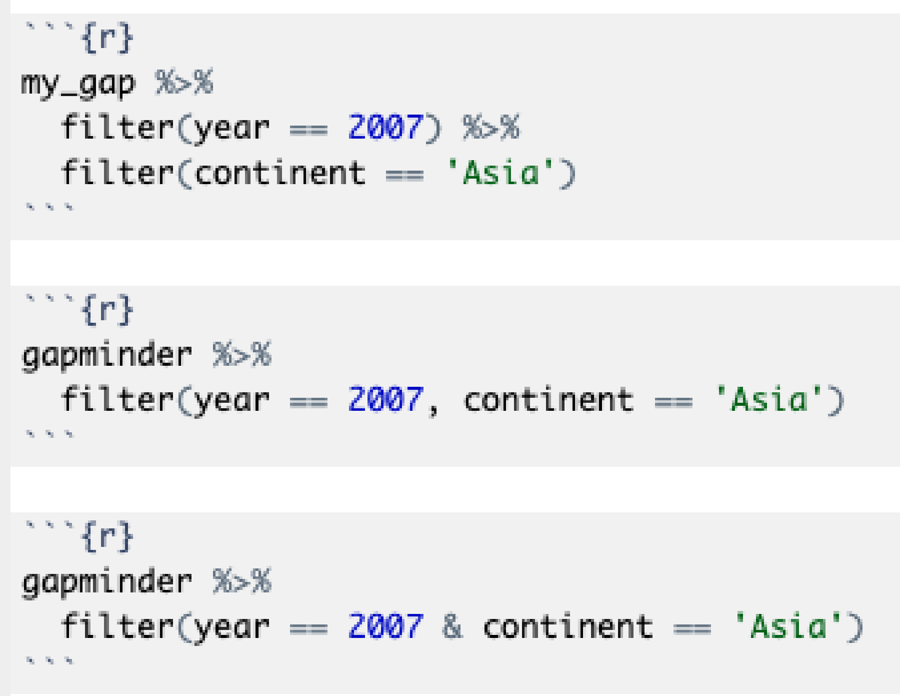
- To keep rows that satisfy a condition
- Condition operators:
  - ==, !=, <,>, >=, <=, %in%
  - !, &, |
- You can use functions of variables
  - max(), min(), etc.
- Usecase:
  - Inspect subsets of data (based on a condition)
  - Use when you care about a portion of the dataset



→ Demo

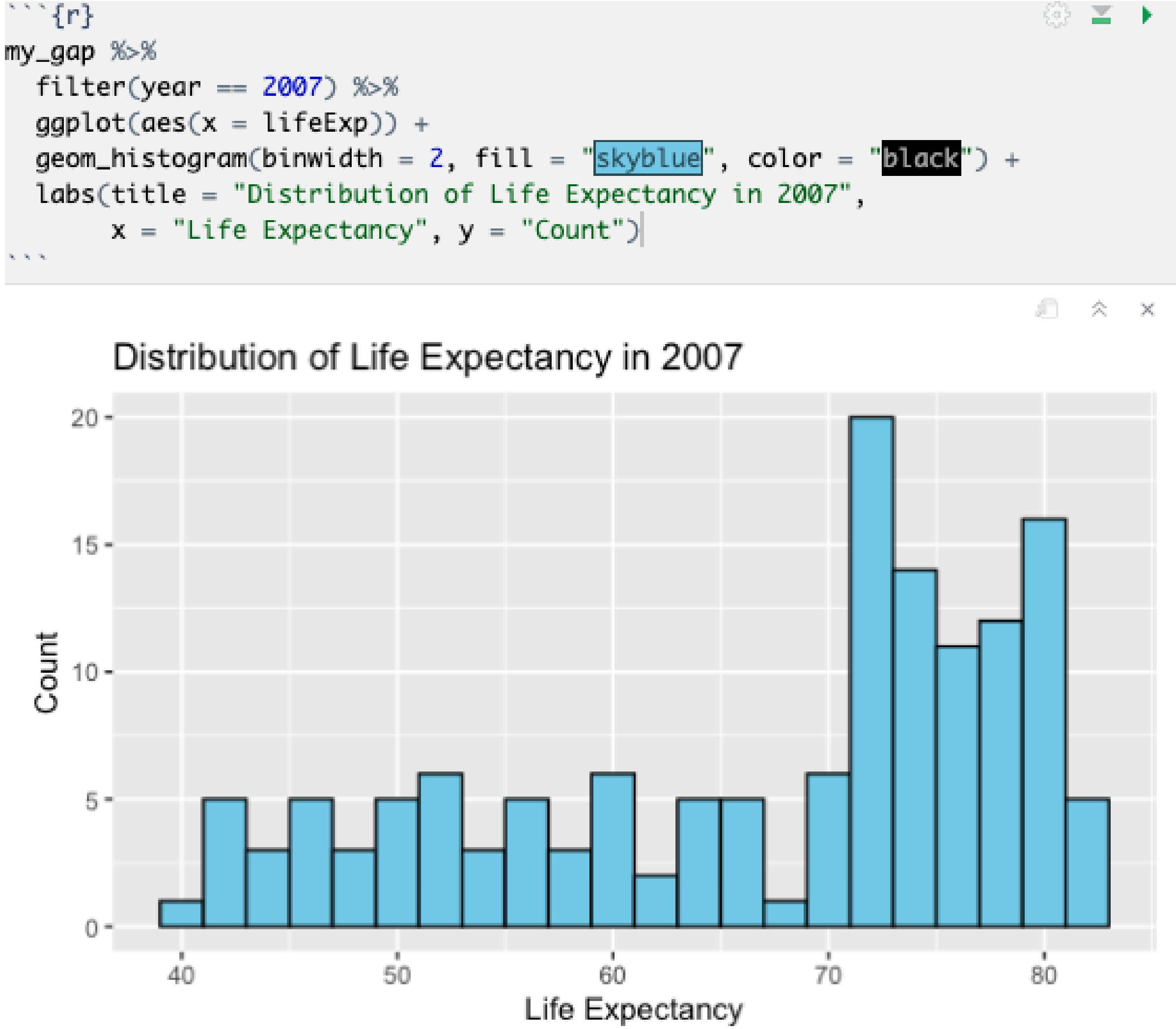
Double filter

- These are all equivalent

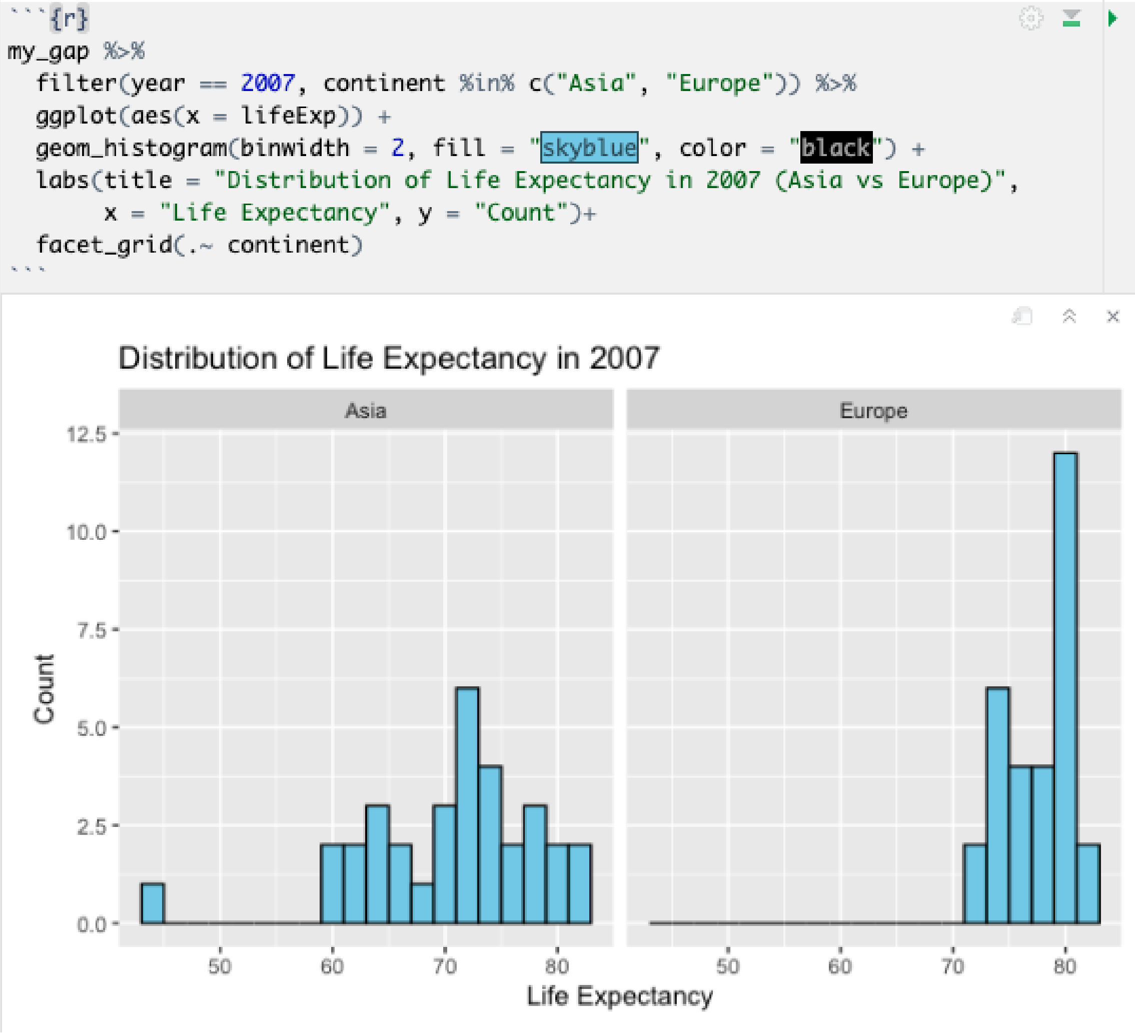


~~~
```{r}
my_gap %>%
  filter(year == 2007) %>%
  filter(continent == 'Asia')
```

```{r}
gapminder %>%
  filter(year == 2007, continent == 'Asia')
```

```{r}
gapminder %>%
  filter(year == 2007 & continent == 'Asia')
```
~~~

But it’s hard to examine filtered tibbles and understand data…



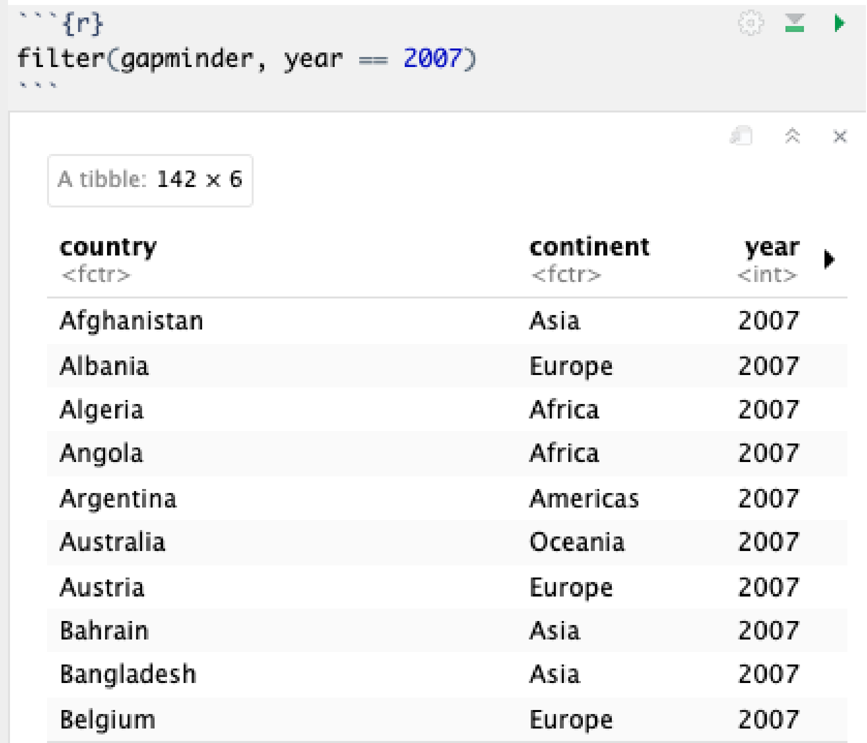
~~~
```{r}
my_gap %>%
  filter(year == 2007) %>%
  ggplot(aes(x = lifeExp)) +
  geom_histogram(binwidth = 2, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Life Expectancy in 2007",
       x = "Life Expectancy", y = "Count")
```
~~~



~~~
```{r}
my_gap %>%
  filter(year == 2007, continent %in% c("Asia", "Europe")) %>%
  ggplot(aes(x = lifeExp)) +
  geom_histogram(binwidth = 2, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Life Expectancy in 2007 (Asia vs Europe)",
       x = "Life Expectancy", y = "Count")+
  facet_grid(.~ continent)
```
~~~

Pipes %>% (for a pipeline)

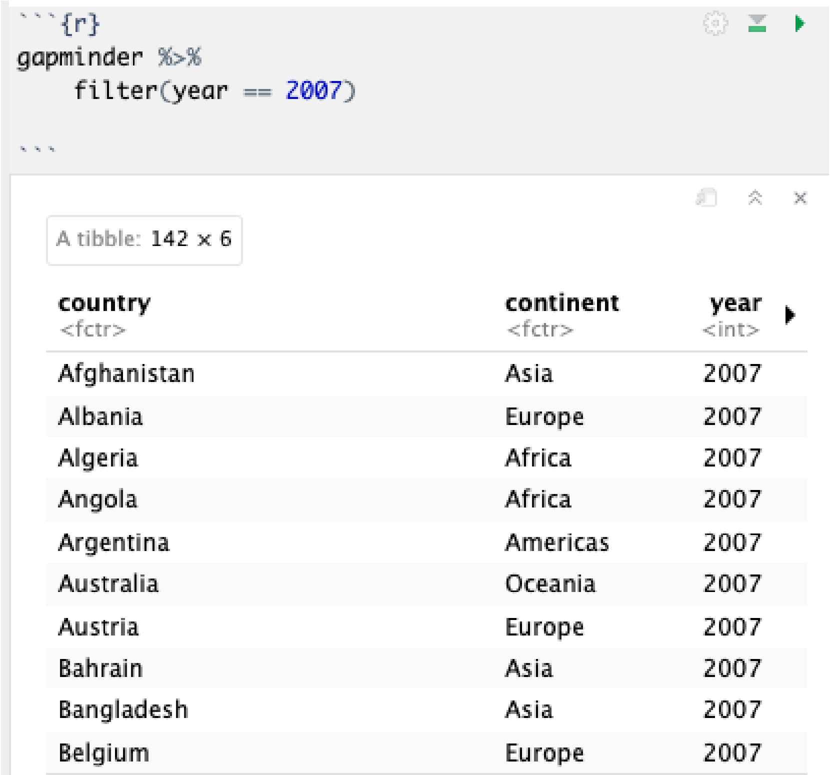
- Basic idea:
  - Use the value on the left-hand side as the first argument to the function on the right-hand side.



~~~
```{r}
filter(gapminder, year == 2007)
```
~~~

A tibble: 142 x 6

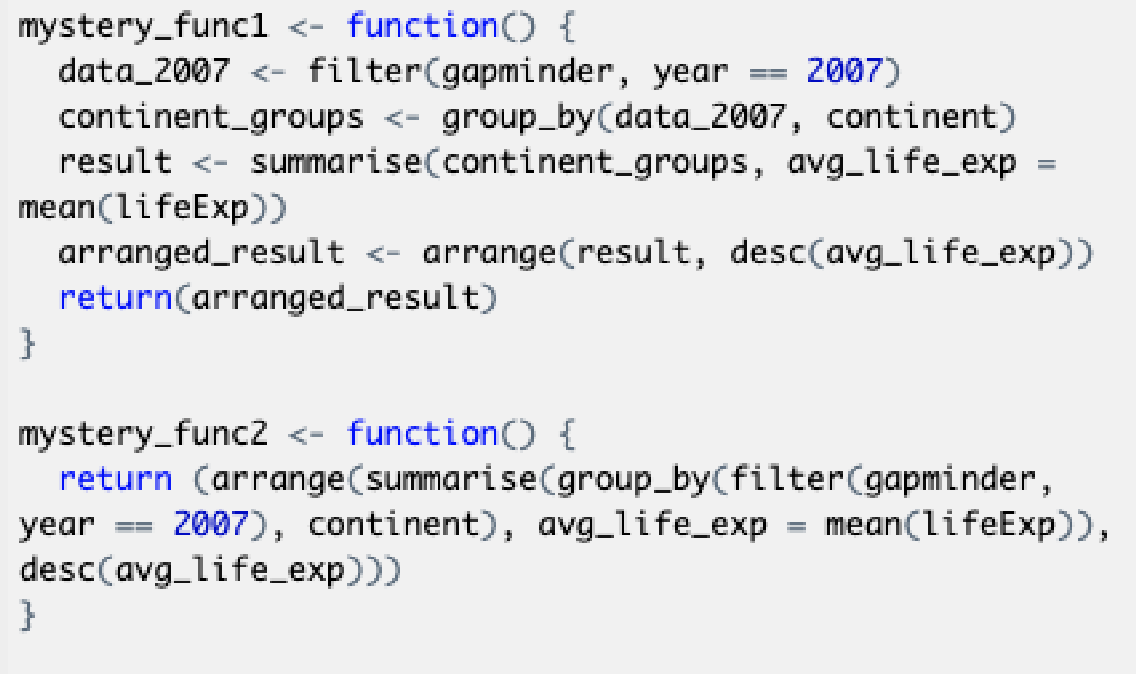
country <fctr>	continent <fctr>	year <int>
Afghanistan	Asia	2007
Albania	Europe	2007
Algeria	Africa	2007
Angola	Africa	2007
Argentina	Americas	2007
Australia	Oceania	2007
Austria	Europe	2007
Bahrain	Asia	2007
Bangladesh	Asia	2007
Belgium	Europe	2007



~~~
```{r}
gapminder %>%
    filter(year == 2007)

```
~~~

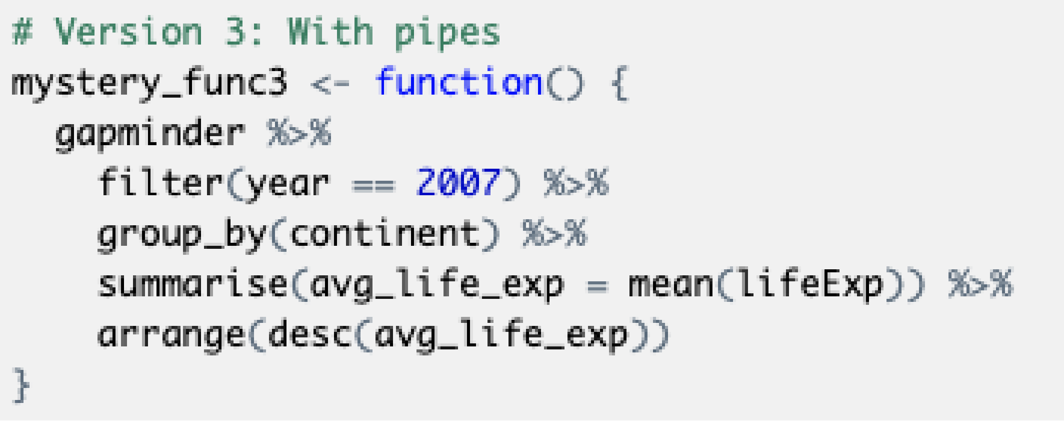
A tibble: 142 x 6

country <fctr>	continent <fctr>	year <int>
Afghanistan	Asia	2007
Albania	Europe	2007
Algeria	Africa	2007
Angola	Africa	2007
Argentina	Americas	2007
Australia	Oceania	2007
Austria	Europe	2007
Bahrain	Asia	2007
Bangladesh	Asia	2007
Belgium	Europe	2007

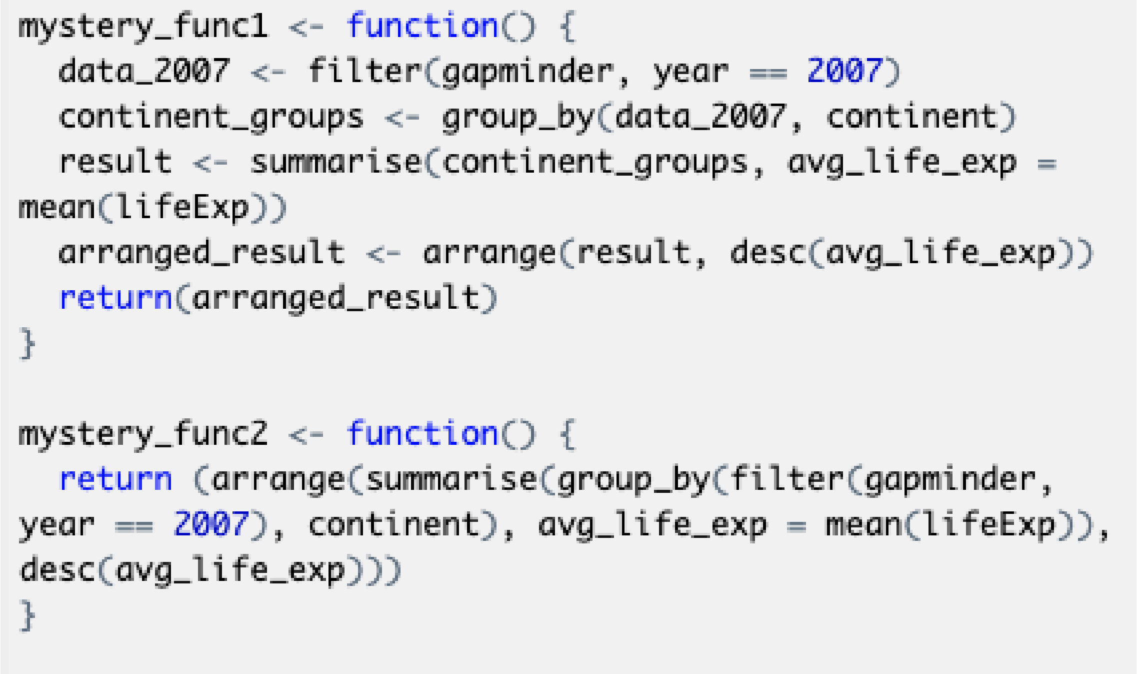


```
mystery_func1 <- function() {
  data_2007 <- filter(gapminder, year == 2007)
  continent_groups <- group_by(data_2007, continent)
  result <- summarise(continent_groups, avg_life_exp =
mean(lifeExp))
  arranged_result <- arrange(result, desc(avg_life_exp))
  return(arranged_result)
}

mystery_func2 <- function() {
  return (arrange(summarise(group_by(filter(gapminder,
year == 2007), continent), avg_life_exp = mean(lifeExp)),
desc(avg_life_exp)))
}
```

vs

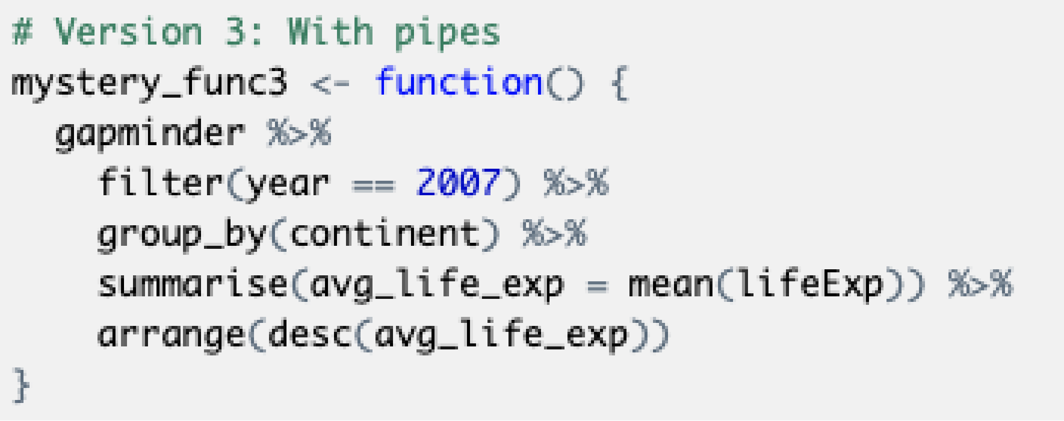


```
# Version 3: With pipes
mystery_func3 <- function() {
  gapminder %>%
    filter(year == 2007) %>%
    group_by(continent) %>%
    summarise(avg_life_exp = mean(lifeExp)) %>%
    arrange(desc(avg_life_exp))
}
```



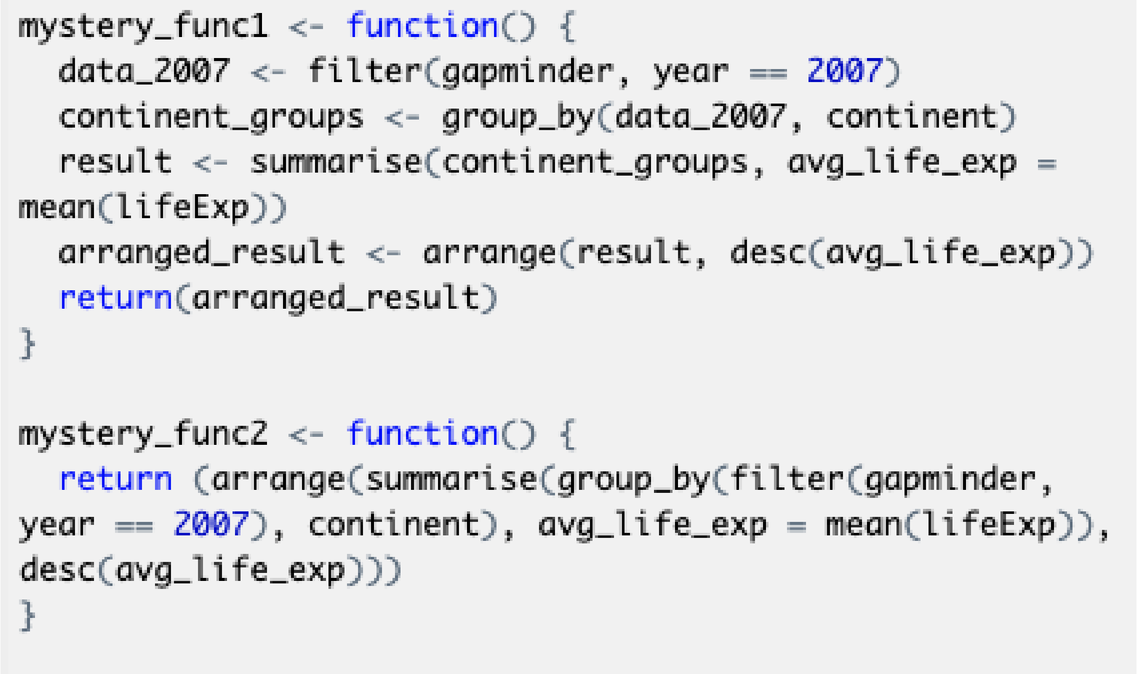
```r
mystery_func1 <- function() {
  data_2007 <- filter(gapminder, year == 2007)
  continent_groups <- group_by(data_2007, continent)
  result <- summarise(continent_groups, avg_life_exp =
mean(lifeExp))
  arranged_result <- arrange(result, desc(avg_life_exp))
  return(arranged_result)
}

mystery_func2 <- function() {
  return (arrange(summarise(group_by(filter(gapminder,
year == 2007), continent), avg_life_exp = mean(lifeExp)),
desc(avg_life_exp)))
}
```

vs



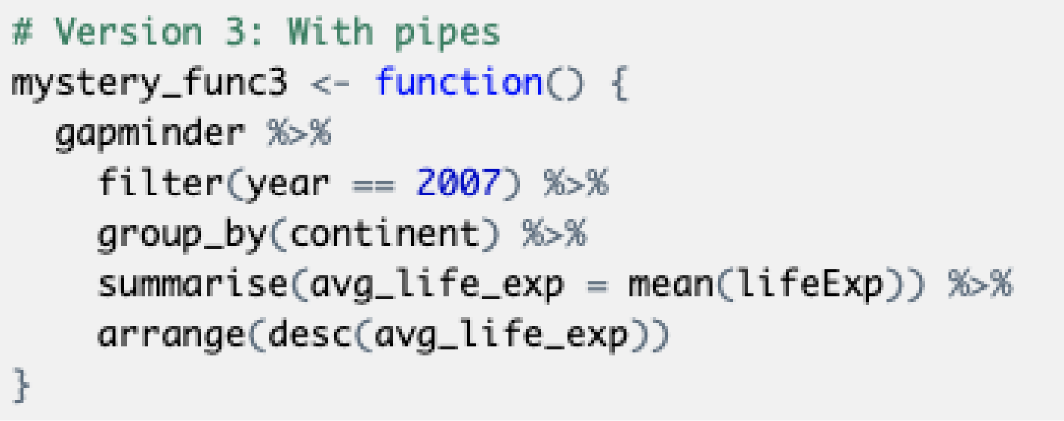
```r
# Version 3: With pipes
mystery_func3 <- function() {
  gapminder %>%
    filter(year == 2007) %>%
    group_by(continent) %>%
    summarise(avg_life_exp = mean(lifeExp)) %>%
    arrange(desc(avg_life_exp))
}
```

- No need of intermediate values
- No need to have nesting dolls of function calls
- Easy for interactive data analysis
- Easily create a pipeline of verbs



```r
mystery_func1 <- function() {
  data_2007 <- filter(gapminder, year == 2007)
  continent_groups <- group_by(data_2007, continent)
  result <- summarise(continent_groups, avg_life_exp =
mean(lifeExp))
  arranged_result <- arrange(result, desc(avg_life_exp))
  return(arranged_result)
}

mystery_func2 <- function() {
  return (arrange(summarise(group_by(filter(gapminder,
year == 2007), continent), avg_life_exp = mean(lifeExp)),
desc(avg_life_exp)))
}
```

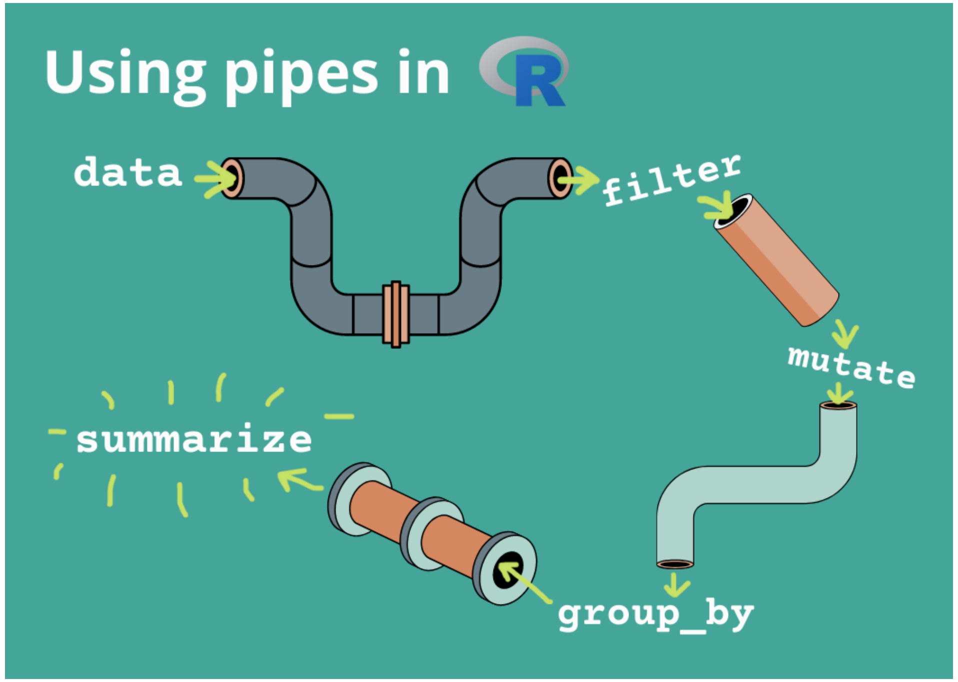
vs



```r
# Version 3: With pipes
mystery_func3 <- function() {
  gapminder %>%
    filter(year == 2007) %>%
    group_by(continent) %>%
    summarise(avg_life_exp = mean(lifeExp)) %>%
    arrange(desc(avg_life_exp))
}
```

- No need of intermediate values
- No need to have nesting dolls of function calls
- Easy for interactive data analysis
- Easily create a pipeline of verbs

https://github.com/tidyverse/magrittr



vs

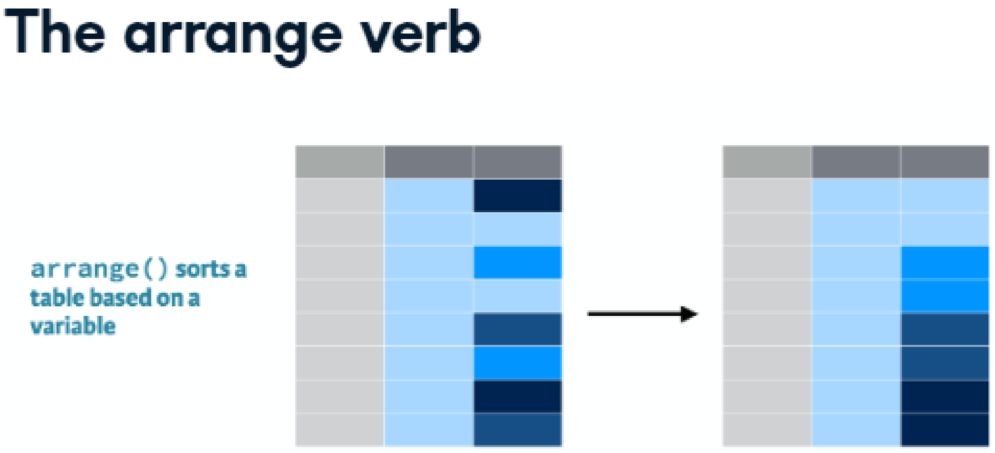
FYI, `|>` are new pipes in Base-R

- Simpler, faster
- I recommend using %>%

3. Arranging with arrange()

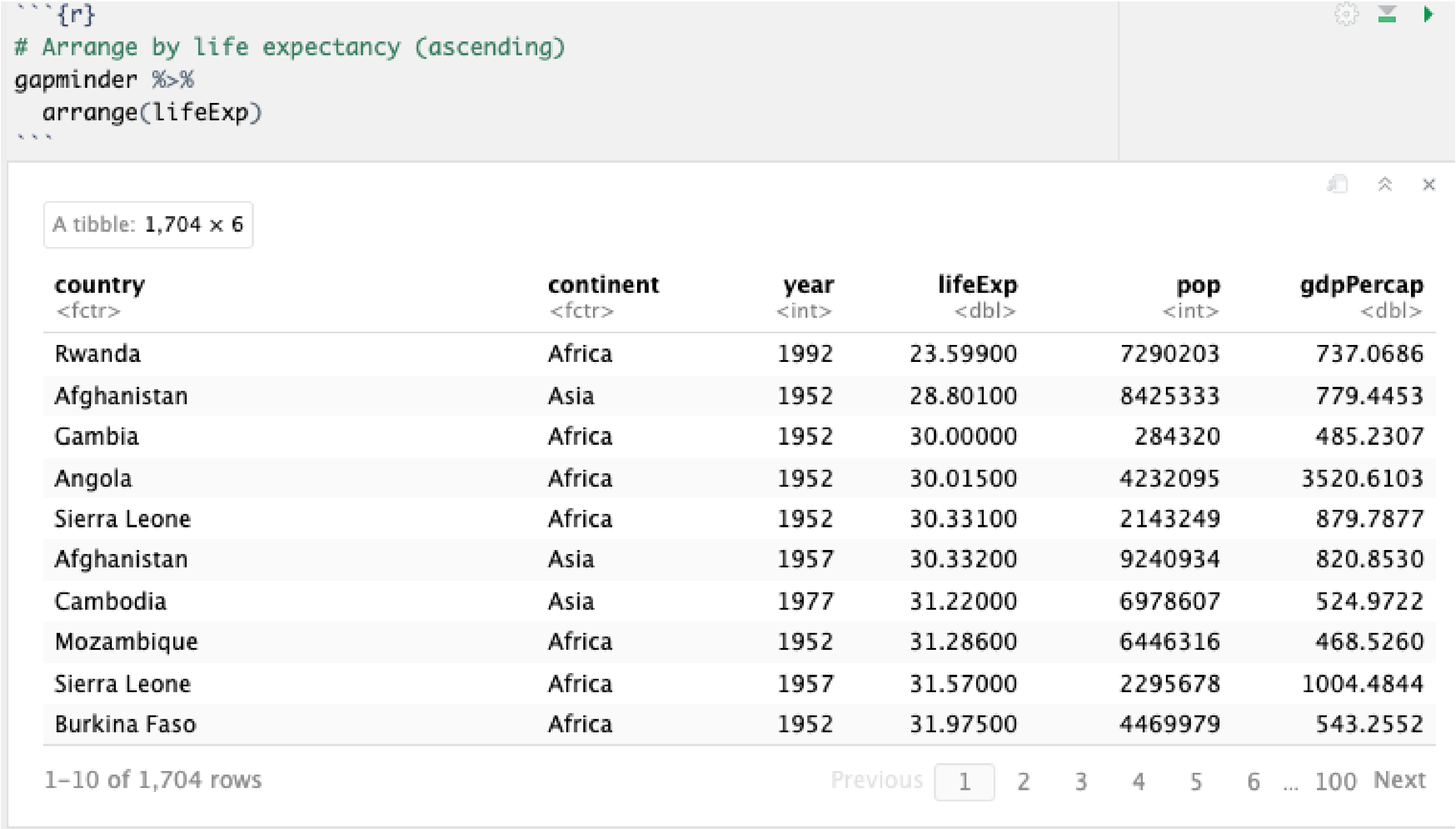
- Used to sort the rows of a data frame
  - Default is ascending order
  - Use desc() for descending order

- Usecase:
  - when you want to see your data in a specific order,
    - perhaps to identify the highest or lowest values quickly, or
    - to prepare your data for presentation in a table or graph.



The arrange verb

`arrange()` sorts a table based on a variable

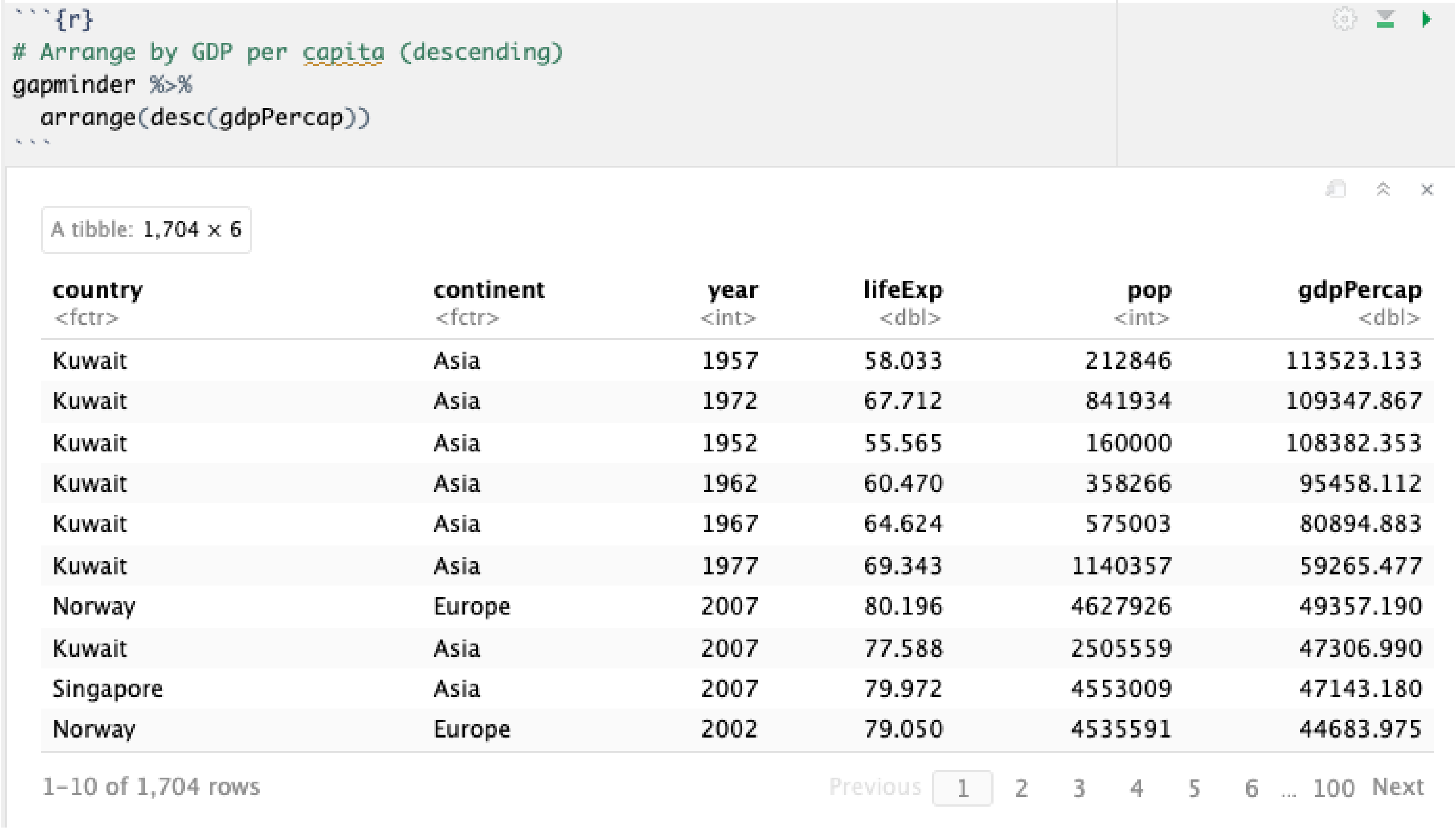


~~~
```{r}
# Arrange by life expectancy (ascending)
gapminder %>%
  arrange(lifeExp)
```
~~~

A tibble: 1,704 × 6

country	continent	year	lifeExp	pop	gdpPercap
<fctr>	<fctr>	<int>	<dbl>	<int>	<dbl>
Rwanda	Africa	1992	23.59900	7290203	737.0686
Afghanistan	Asia	1952	28.80100	8425333	779.4453
Gambia	Africa	1952	30.00000	284320	485.2307
Angola	Africa	1952	30.01500	4232095	3520.6103
Sierra Leone	Africa	1952	30.33100	2143249	879.7877
Afghanistan	Asia	1957	30.33200	9240934	820.8530
Cambodia	Asia	1977	31.22000	6978607	524.9722
Mozambique	Africa	1952	31.28600	6446316	468.5260
Sierra Leone	Africa	1957	31.57000	2295678	1004.4844
Burkina Faso	Africa	1952	31.97500	4469979	543.2552

1-10 of 1,704 rows

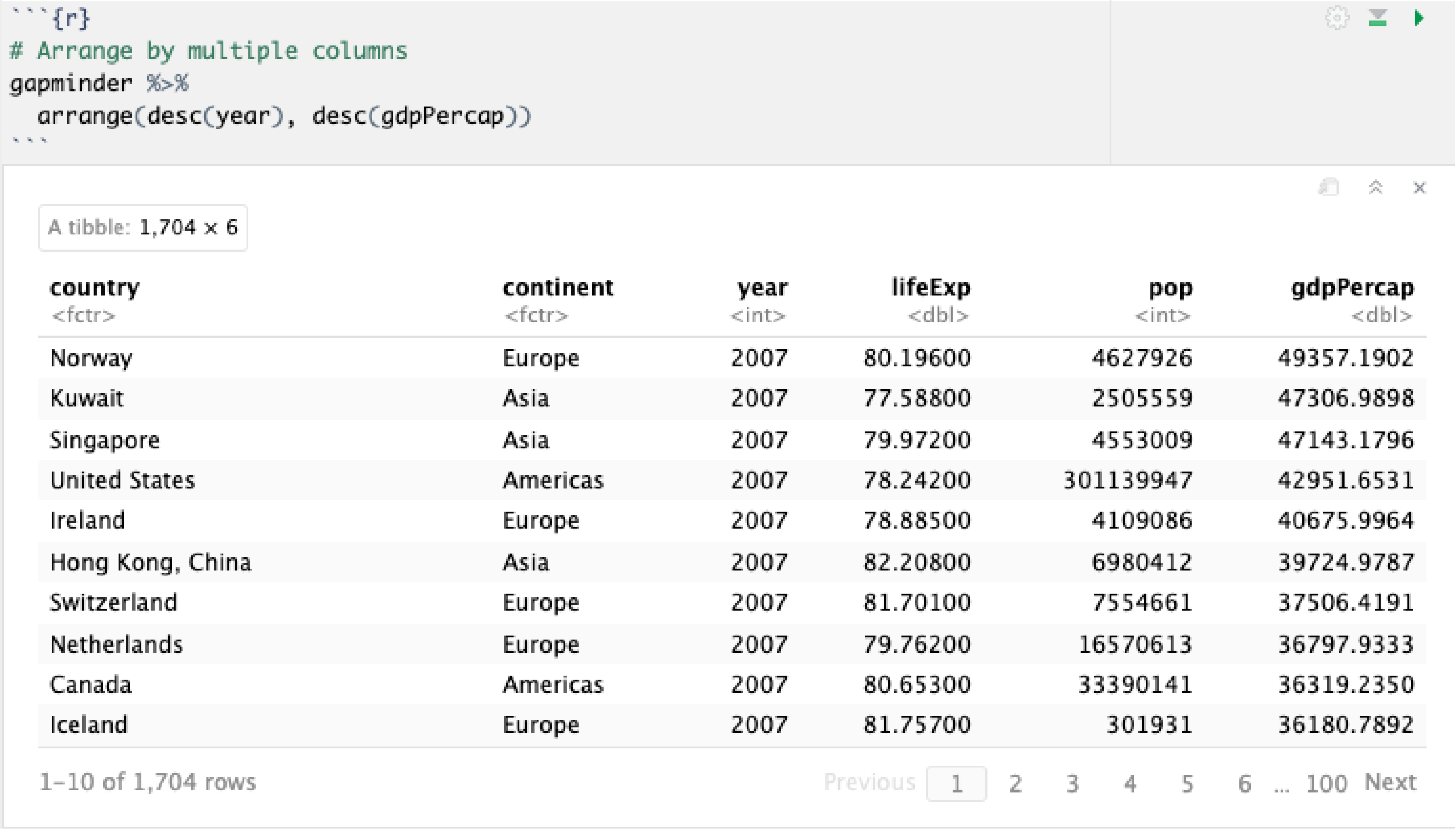


~~~
```{r}
# Arrange by GDP per capita (descending)
gapminder %>%
  arrange(desc(gdpPercap))
```
~~~

A tibble: 1,704 × 6

country <fctr>	continent <fctr>	year <int>	lifeExp <dbl>	pop <int>	gdpPercap <dbl>
Kuwait	Asia	1957	58.033	212846	113523.133
Kuwait	Asia	1972	67.712	841934	109347.867
Kuwait	Asia	1952	55.565	160000	108382.353
Kuwait	Asia	1962	60.470	358266	95458.112
Kuwait	Asia	1967	64.624	575003	80894.883
Kuwait	Asia	1977	69.343	1140357	59265.477
Norway	Europe	2007	80.196	4627926	49357.190
Kuwait	Asia	2007	77.588	2505559	47306.990
Singapore	Asia	2007	79.972	4553009	47143.180
Norway	Europe	2002	79.050	4535591	44683.975

1-10 of 1,704 rows



~~~
```{r}
# Arrange by multiple columns
gapminder %>%
  arrange(desc(year), desc(gdpPercap))
```
~~~

A tibble: 1,704 × 6

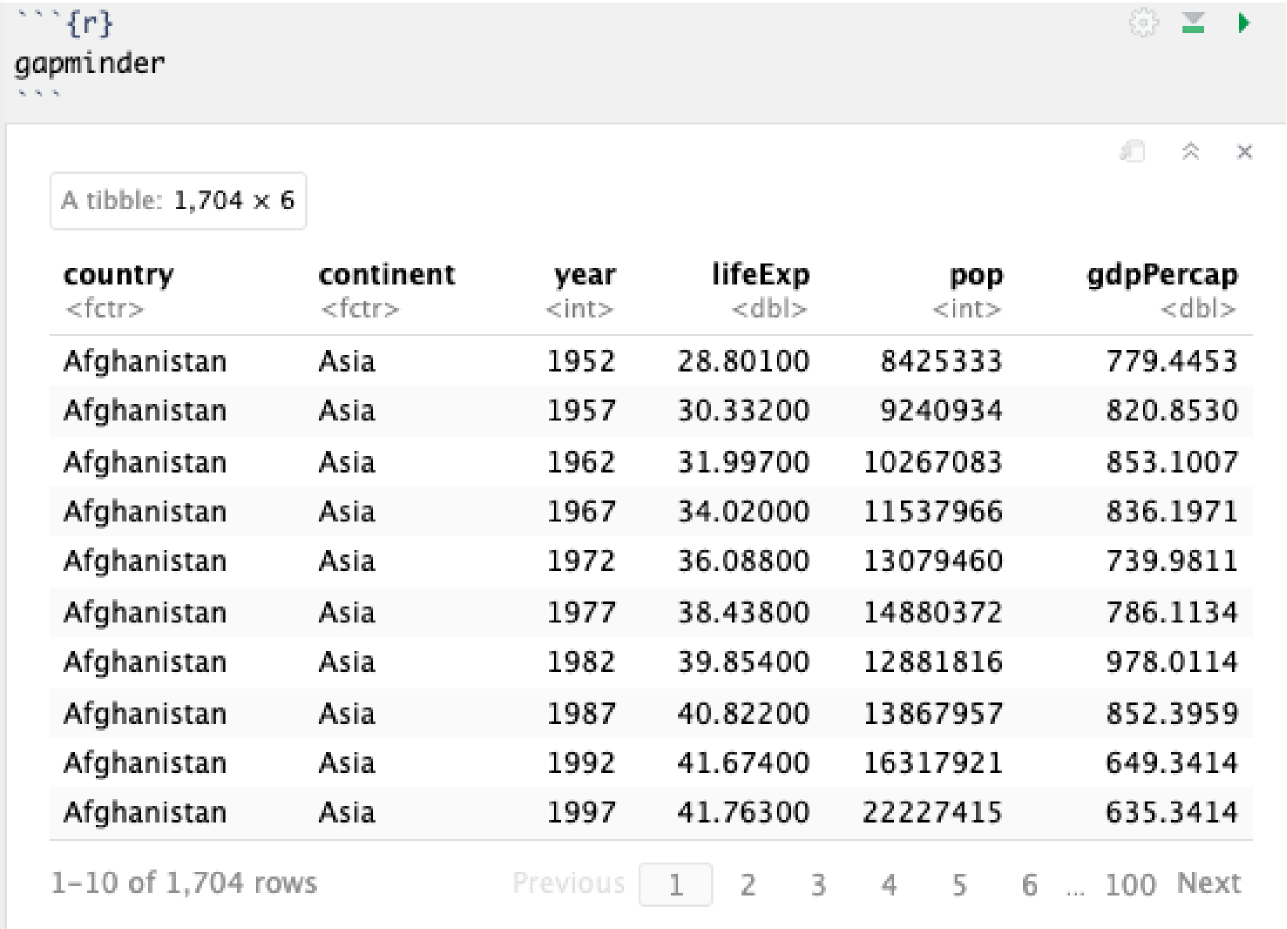
country <fctr>	continent <fctr>	year <int>	lifeExp <dbl>	pop <int>	gdpPercap <dbl>
Norway	Europe	2007	80.19600	4627926	49357.1902
Kuwait	Asia	2007	77.58800	2505559	47306.9898
Singapore	Asia	2007	79.97200	4553009	47143.1796
United States	Americas	2007	78.24200	301139947	42951.6531
Ireland	Europe	2007	78.88500	4109086	40675.9964
Hong Kong, China	Asia	2007	82.20800	6980412	39724.9787
Switzerland	Europe	2007	81.70100	7554661	37506.4191
Netherlands	Europe	2007	79.76200	16570613	36797.9333
Canada	Americas	2007	80.65300	33390141	36319.2350
Iceland	Europe	2007	81.75700	301931	36180.7892

1-10 of 1,704 rows

4. Variable Selection

- To choose or rename ***columns***
- Using the verb select()
    - But there are many helpers:
        - starts_with(), ends_with(), etc.

- Usecase
    - Focus only on relevant variables
    - Summarizing only across relevant groups of variables

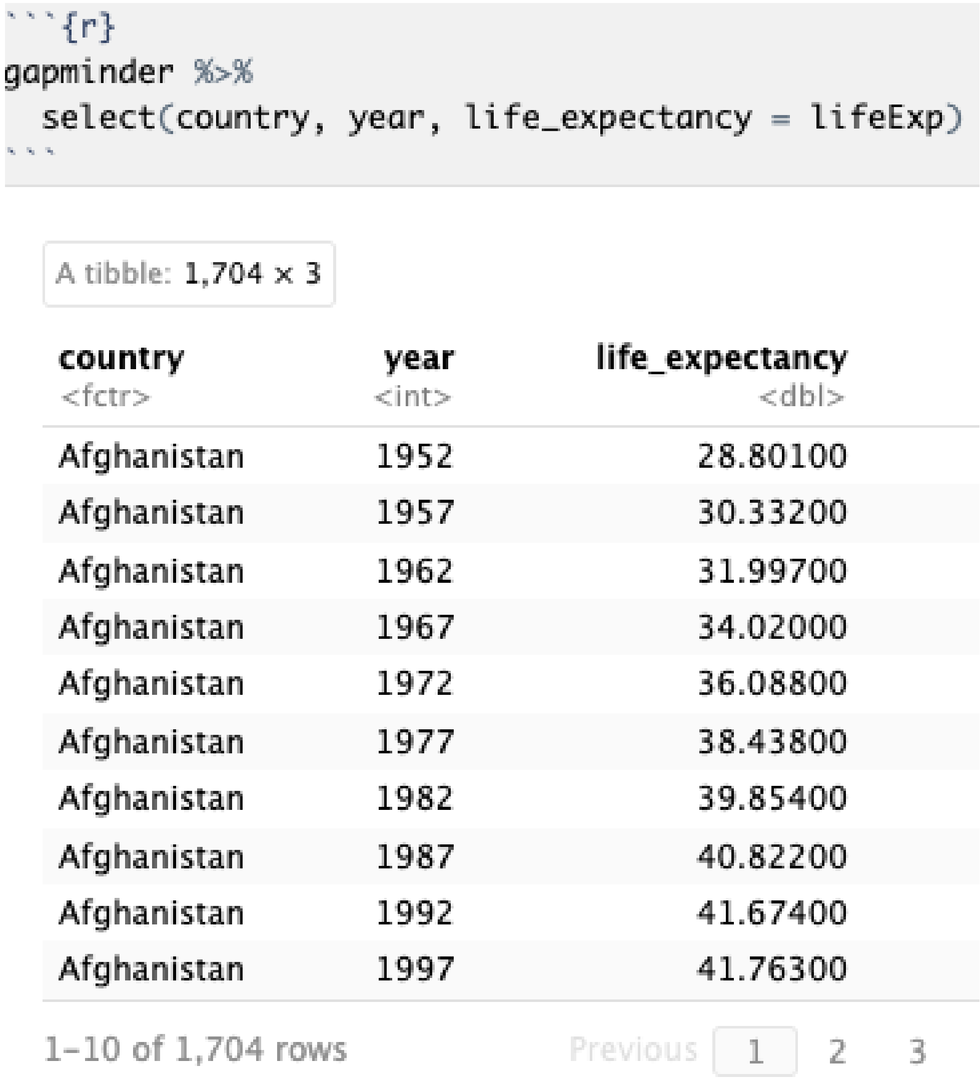


```{r}
gapminder
```

A tibble: 1,704 × 6

country <fctr>	continent <fctr>	year <int>	lifeExp <dbl>	pop <int>	gdpPercap <dbl>
Afghanistan	Asia	1952	28.80100	8425333	779.4453
Afghanistan	Asia	1957	30.33200	9240934	820.8530
Afghanistan	Asia	1962	31.99700	10267083	853.1007
Afghanistan	Asia	1967	34.02000	11537966	836.1971
Afghanistan	Asia	1972	36.08800	13079460	739.9811
Afghanistan	Asia	1977	38.43800	14880372	786.1134
Afghanistan	Asia	1982	39.85400	12881816	978.0114
Afghanistan	Asia	1987	40.82200	13867957	852.3959
Afghanistan	Asia	1992	41.67400	16317921	649.3414
Afghanistan	Asia	1997	41.76300	22227415	635.3414

1–10 of 1,704 rows



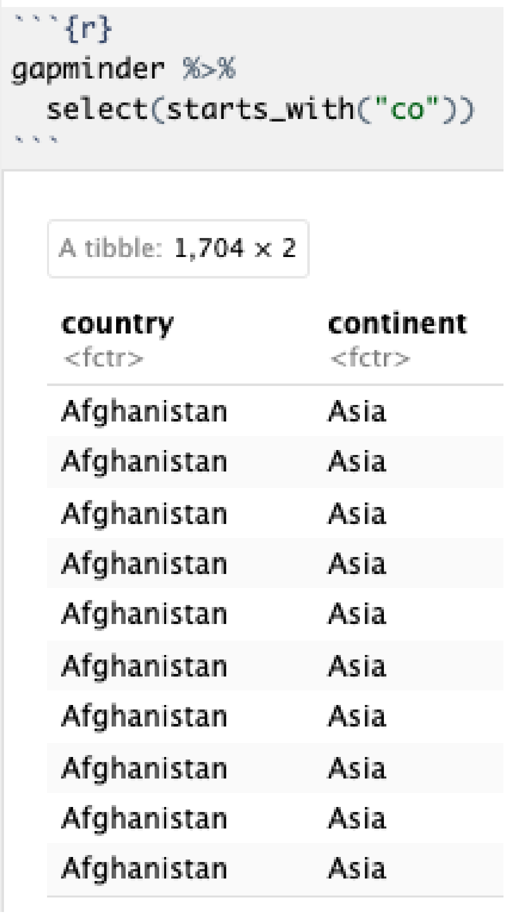
```{r}
gapminder %>%
  select(country, year, life_expectancy = lifeExp)
```

A tibble: 1,704 × 3

country <fctr>	year <int>	life_expectancy <dbl>
Afghanistan	1952	28.80100
Afghanistan	1957	30.33200
Afghanistan	1962	31.99700
Afghanistan	1967	34.02000
Afghanistan	1972	36.08800
Afghanistan	1977	38.43800
Afghanistan	1982	39.85400
Afghanistan	1987	40.82200
Afghanistan	1992	41.67400
Afghanistan	1997	41.76300

1–10 of 1,704 rows

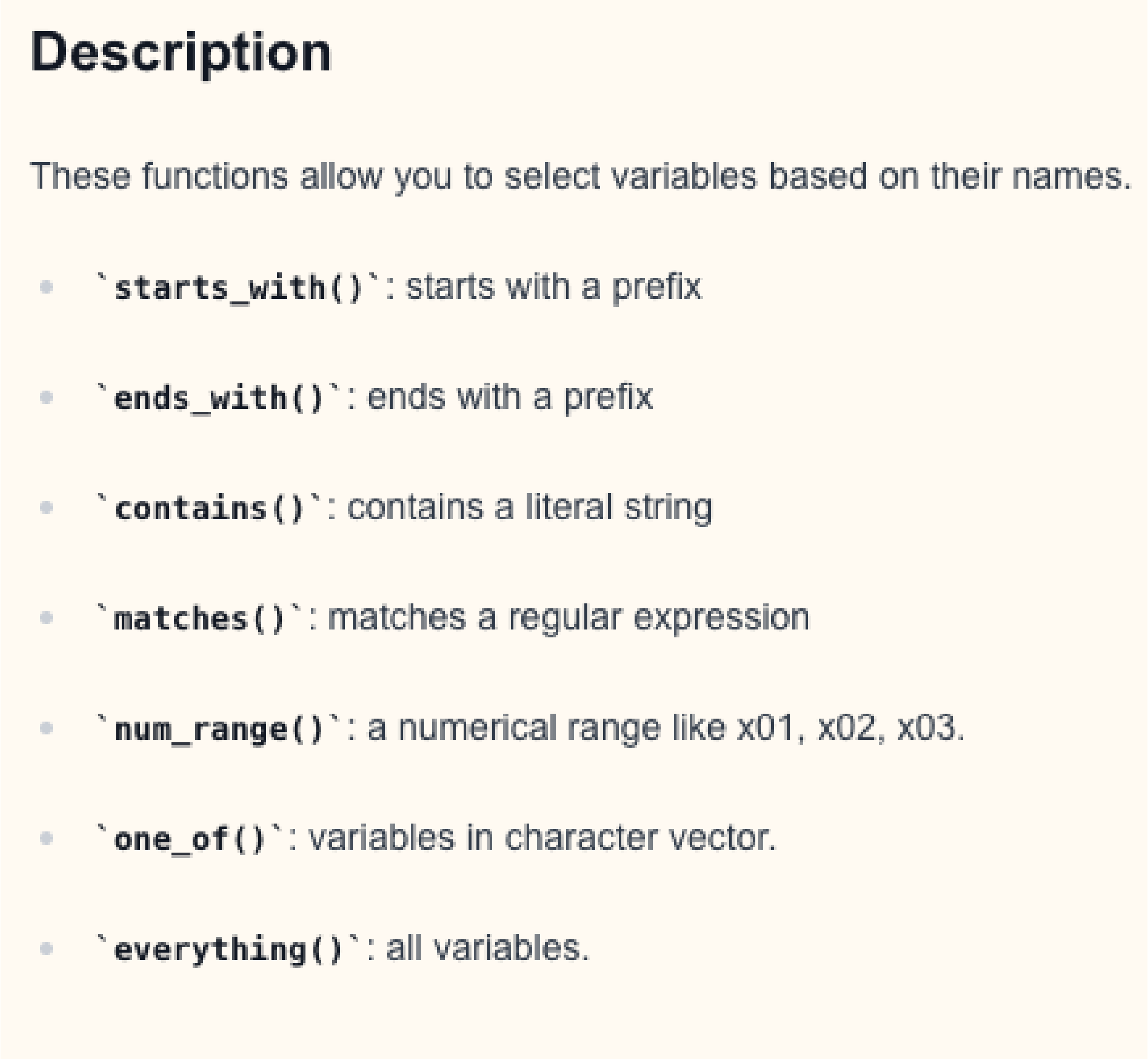
Select has many helpers



~~~
```{r}
gapminder %>%
  select(starts_with("co"))
```
~~~

A tibble: 1,704 x 2

country <fctr>	continent <fctr>
Afghanistan	Asia
Afghanistan	Asia
Afghanistan	Asia
Afghanistan	Asia
Afghanistan	Asia
Afghanistan	Asia
Afghanistan	Asia
Afghanistan	Asia
Afghanistan	Asia
Afghanistan	Asia

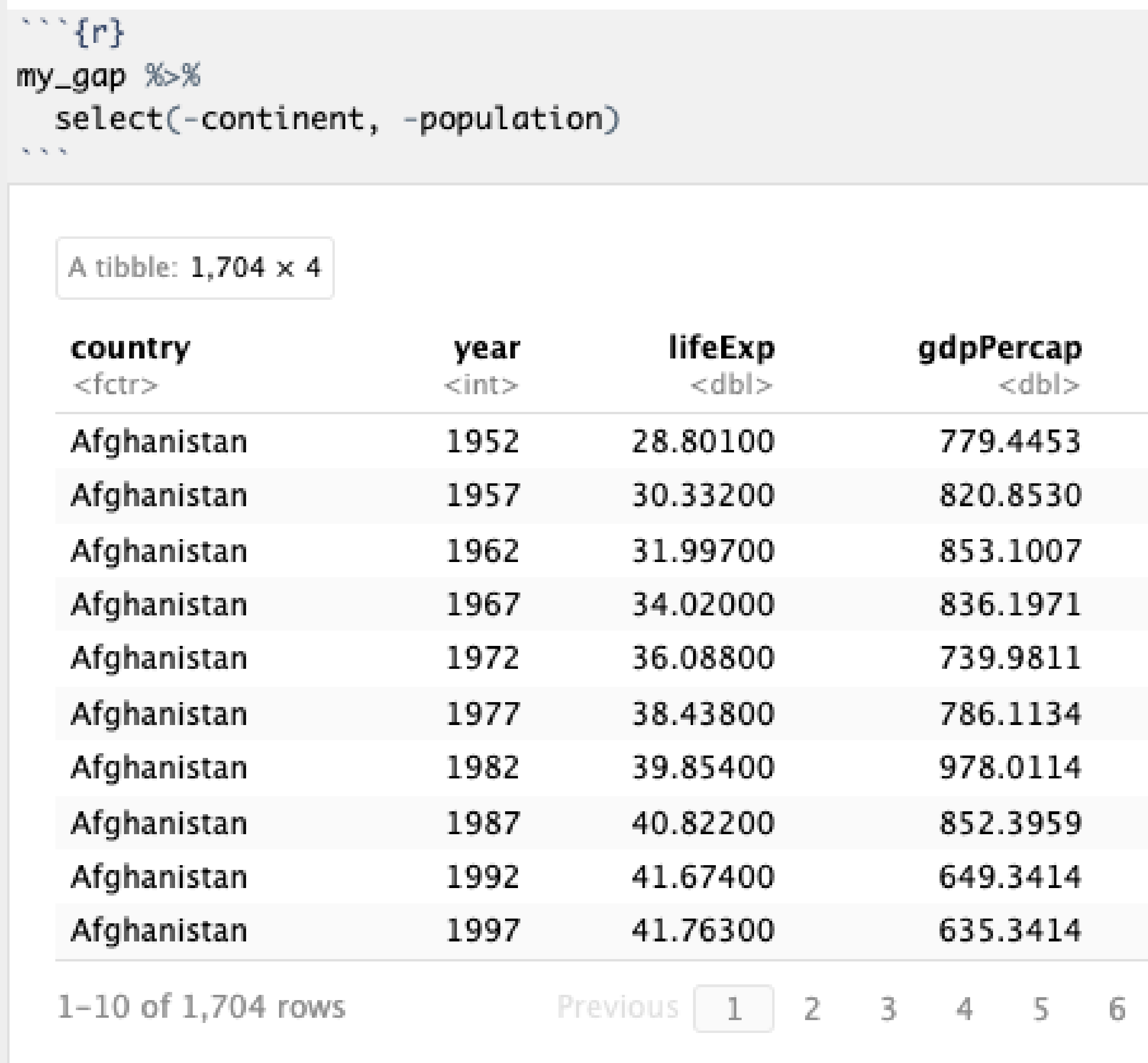


Description

These functions allow you to select variables based on their names.

- `starts_with()`: starts with a prefix
- `ends_with()`: ends with a prefix
- `contains()`: contains a literal string
- `matches()`: matches a regular expression
- `num_range()`: a numerical range like x01, x02, x03.
- `one_of()`: variables in character vector.
- `everything()`: all variables.

Negative select



~~~
```{r}
my_gap %>%
  select(-continent, -population)
```
~~~

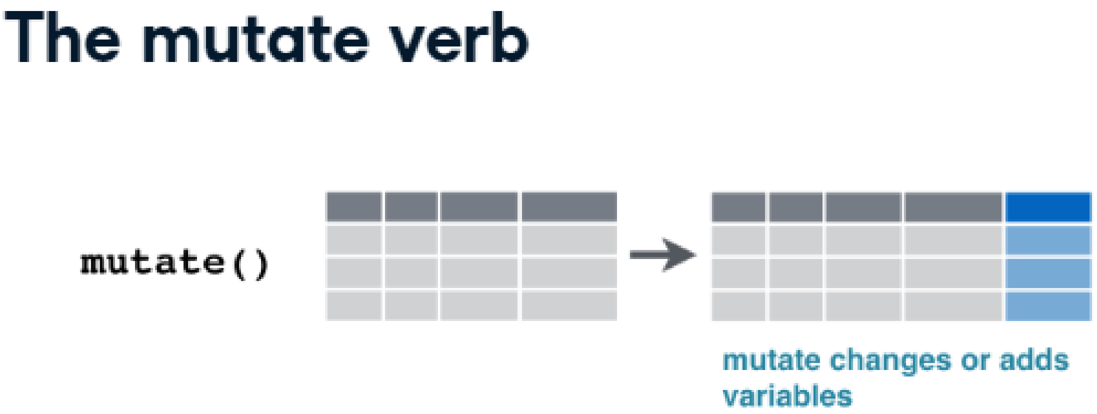
A tibble: 1,704 x 4

country	year	lifeExp	gdpPercap
<fctr>	<int>	<dbl>	<dbl>
Afghanistan	1952	28.80100	779.4453
Afghanistan	1957	30.33200	820.8530
Afghanistan	1962	31.99700	853.1007
Afghanistan	1967	34.02000	836.1971
Afghanistan	1972	36.08800	739.9811
Afghanistan	1977	38.43800	786.1134
Afghanistan	1982	39.85400	978.0114
Afghanistan	1987	40.82200	852.3959
Afghanistan	1992	41.67400	649.3414
Afghanistan	1997	41.76300	635.3414

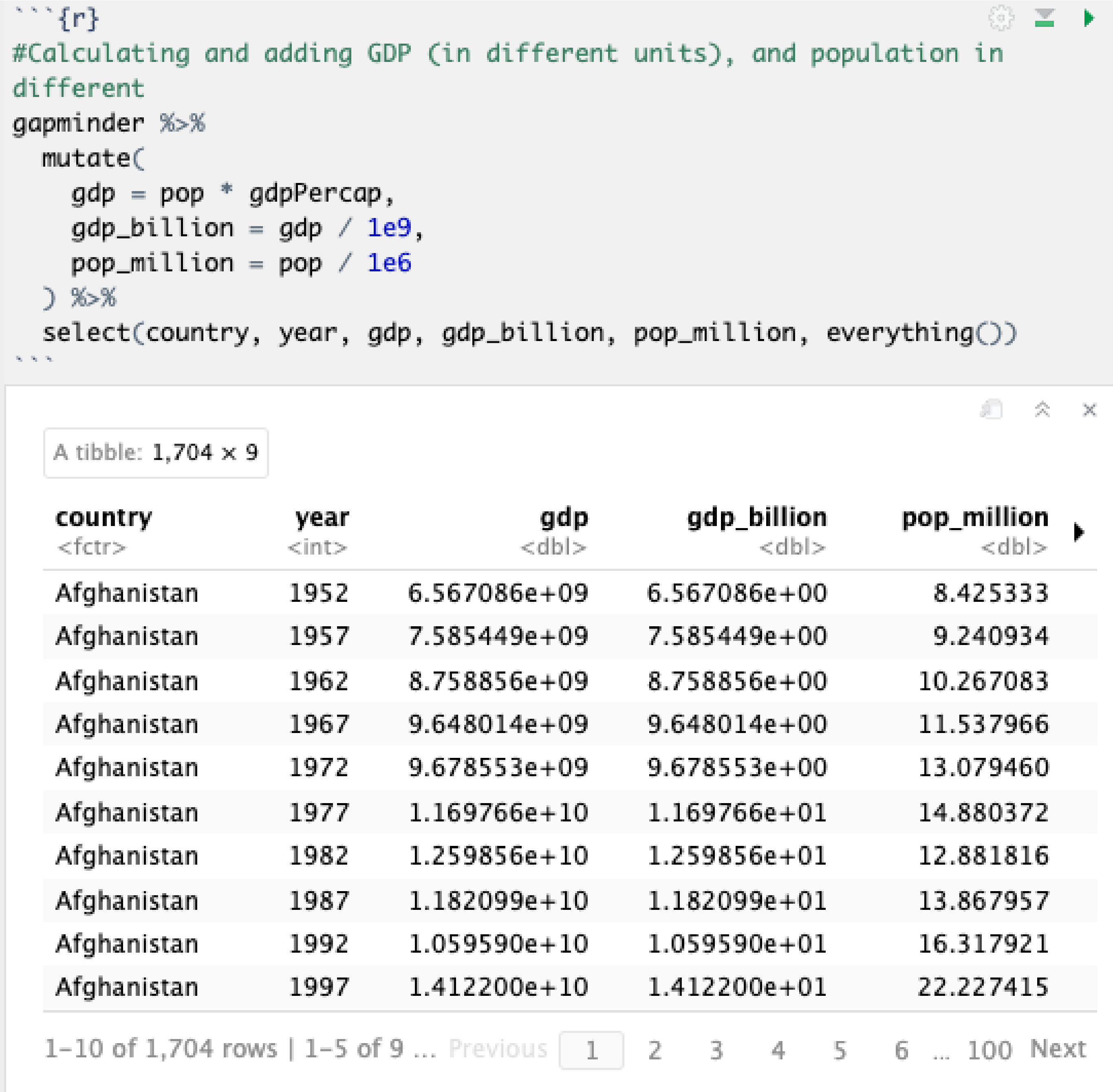
1–10 of 1,704 rows

5. Mutating with mutate()

- Useful for
    - Computing / deriving new variables and measures from existing data
    - Instead of computing some commonly used measure each time, you can just add it to an expanded dataframe



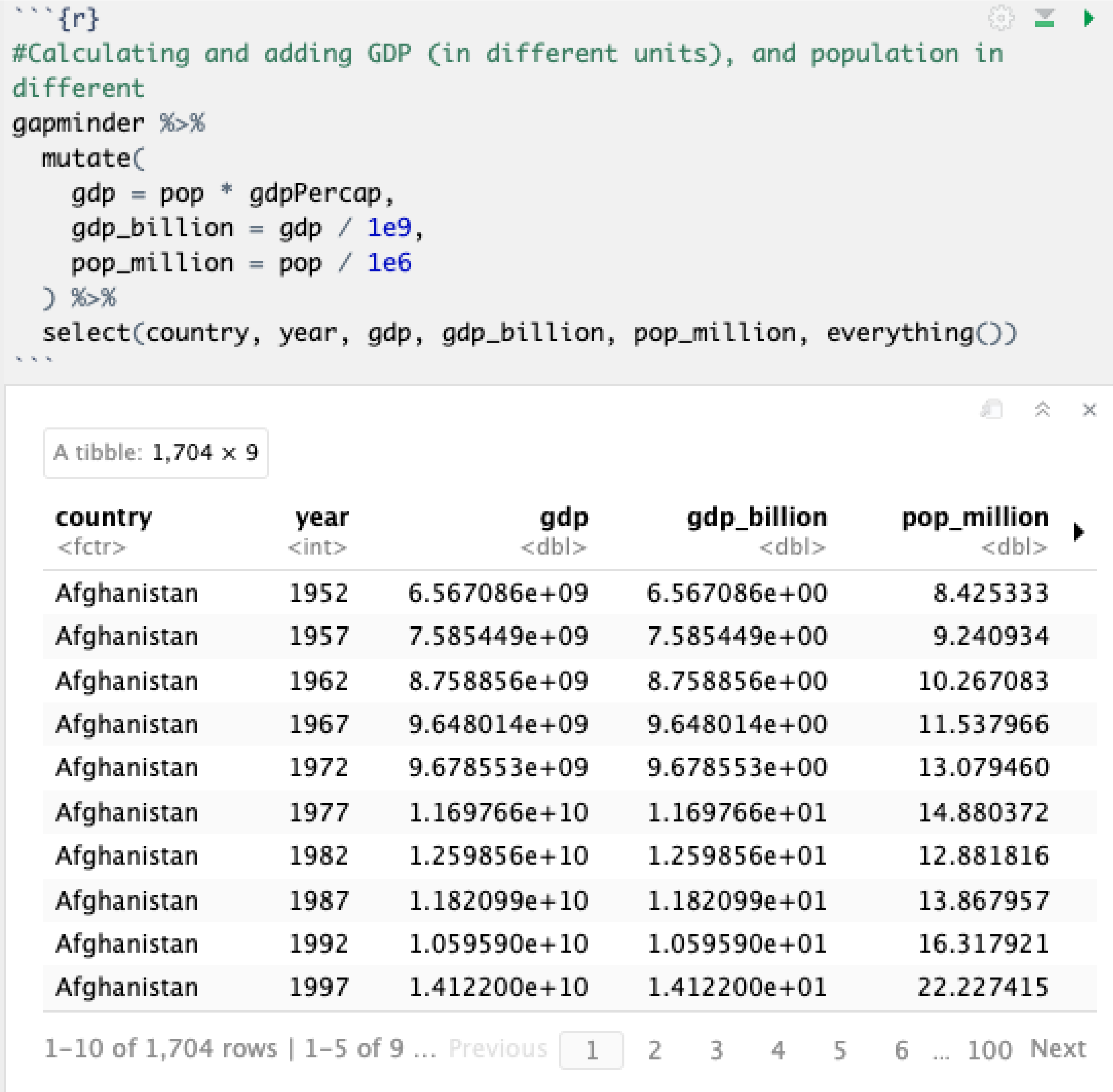
The mutate verb



~~~
```{r}
#Calculating and adding GDP (in different units), and population in different
gapminder %>%
  mutate(
    gdp = pop * gdpPercap,
    gdp_billion = gdp / 1e9,
    pop_million = pop / 1e6
  ) %>%
  select(country, year, gdp, gdp_billion, pop_million, everything())
```
~~~

```
A tibble: 1,704 × 9

country      year          gdp   gdp_billion  pop_million
<fctr>      <int>        <dbl>         <dbl>        <dbl>
Afghanistan  1952 6.567086e+09  6.567086e+00     8.425333
Afghanistan  1957 7.585449e+09  7.585449e+00     9.240934
Afghanistan  1962 8.758856e+09  8.758856e+00    10.267083
Afghanistan  1967 9.648014e+09  9.648014e+00    11.537966
Afghanistan  1972 9.678553e+09  9.678553e+00    13.079460
Afghanistan  1977 1.169766e+10  1.169766e+01    14.880372
Afghanistan  1982 1.259856e+10  1.259856e+01    12.881816
Afghanistan  1987 1.182099e+10  1.182099e+01    13.867957
Afghanistan  1992 1.059590e+10  1.059590e+01    16.317921
Afghanistan  1997 1.412200e+10  1.412200e+01    22.227415

1-10 of 1,704 rows | 1-5 of 9 ...  Previous  1  2  3  4  5  6 ... 100  Next
```

Everything() refers to all columns not otherwise specified in select()



```{r}
#Calculating and adding GDP (in different units), and population in different
gapminder %>%
  mutate(
    gdp = pop * gdpPercap,
    gdp_billion = gdp / 1e9,
    pop_million = pop / 1e6
  ) %>%
  select(country, year, gdp, gdp_billion, pop_million, everything())
```

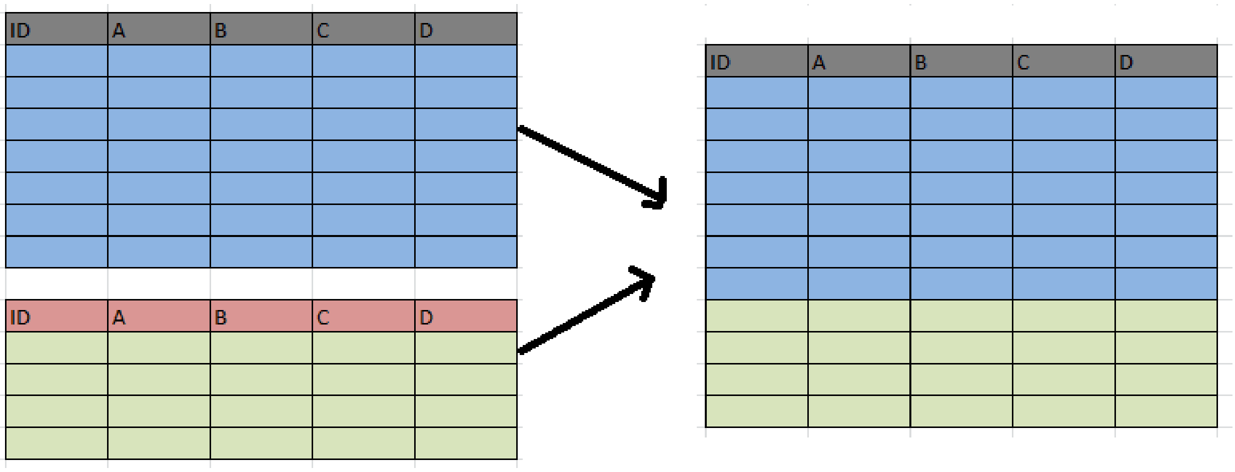
A tibble: 1,704 × 9

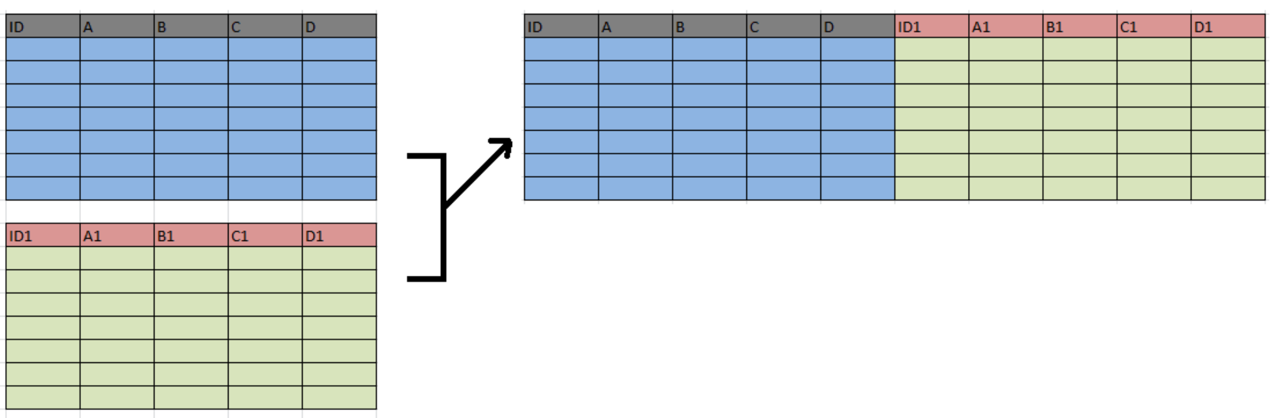
country <fctr>	year <int>	gdp <dbl>	gdp_billion <dbl>	pop_million <dbl>
Afghanistan	1952	6.567086e+09	6.567086e+00	8.425333
Afghanistan	1957	7.585449e+09	7.585449e+00	9.240934
Afghanistan	1962	8.758856e+09	8.758856e+00	10.267083
Afghanistan	1967	9.648014e+09	9.648014e+00	11.537966
Afghanistan	1972	9.678553e+09	9.678553e+00	13.079460
Afghanistan	1977	1.169766e+10	1.169766e+01	14.880372
Afghanistan	1982	1.259856e+10	1.259856e+01	12.881816
Afghanistan	1987	1.182099e+10	1.182099e+01	13.867957
Afghanistan	1992	1.059590e+10	1.059590e+01	16.317921
Afghanistan	1997	1.412200e+10	1.412200e+01	22.227415

1–10 of 1,704 rows | 1–5 of 9 ... Previous 1 2 3 4 5 6 ... 100 Next

6. Binding

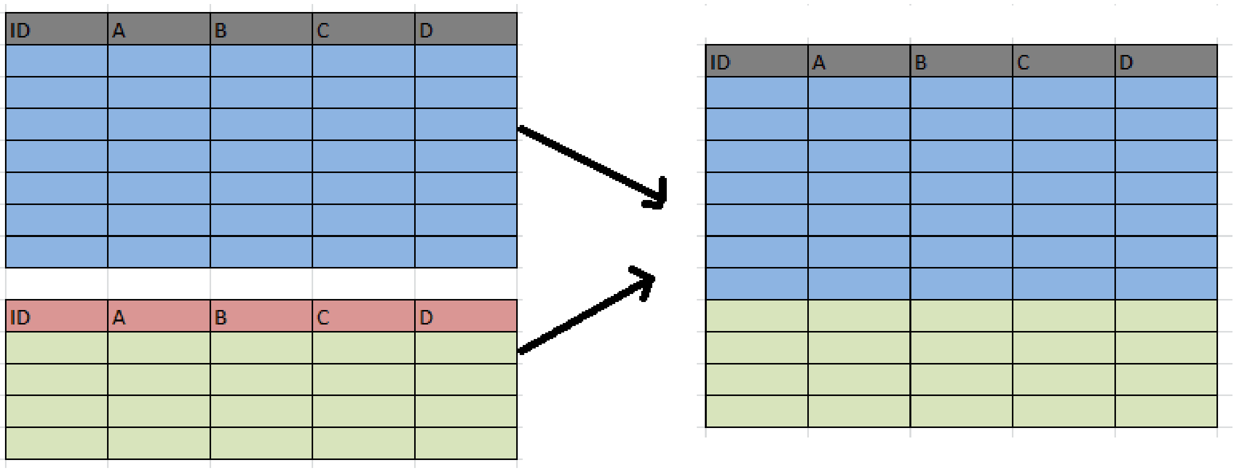
- Lets you combine datasets
- bind_rows()
  - Stack datasets on top of each other
- bind_columns()
  - Place them side to side



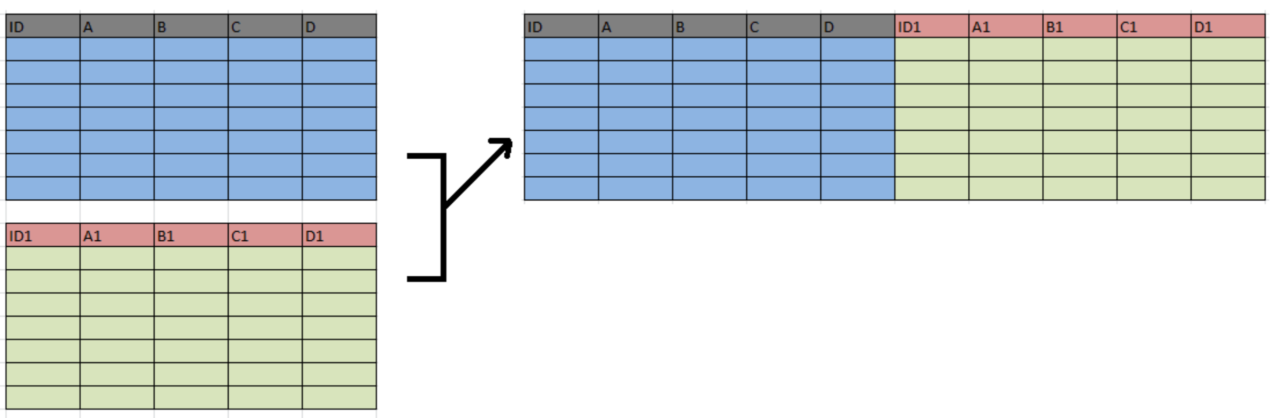


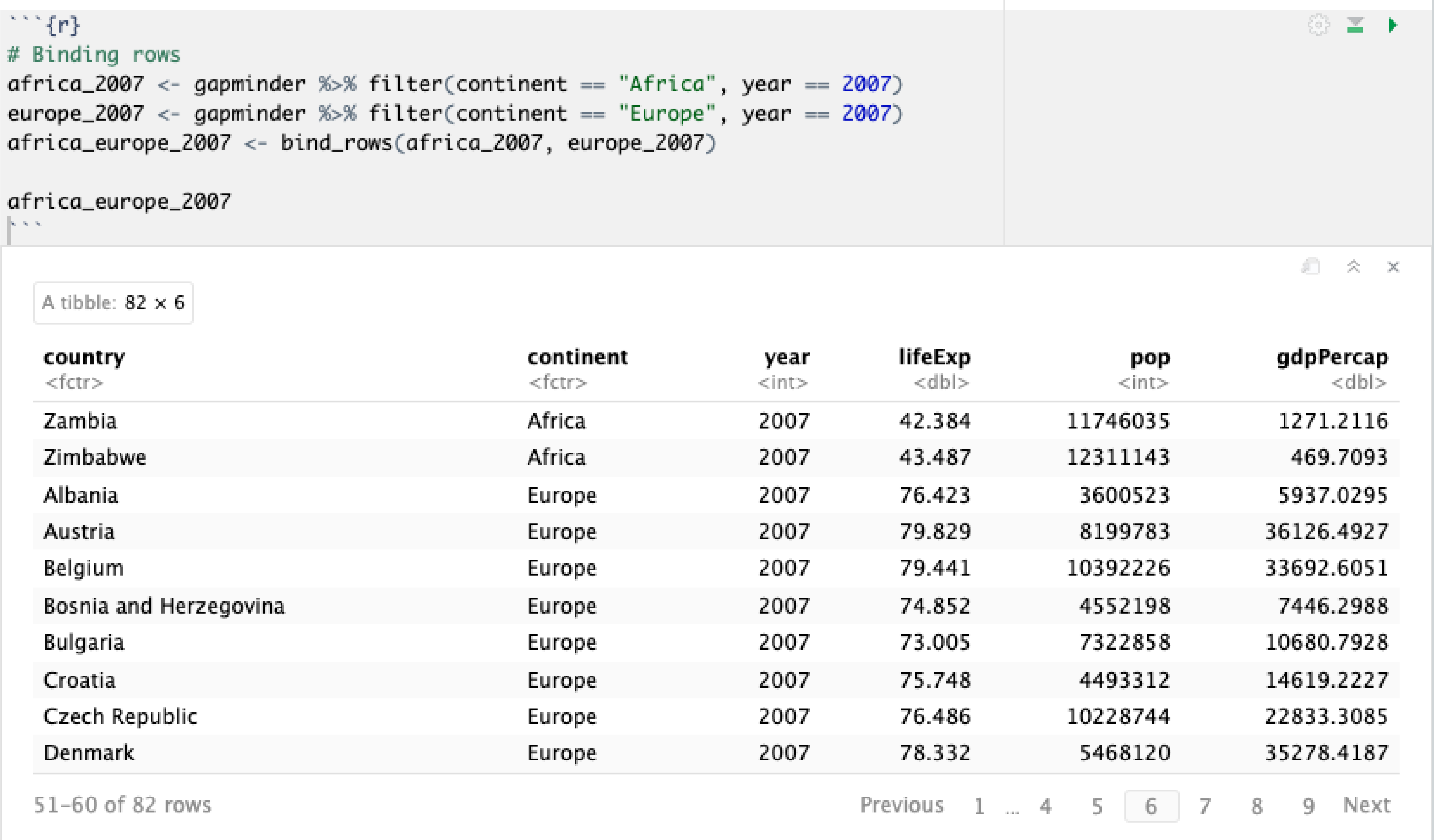
6. Binding – take care

- bind_rows()
  - Data being combined should have the same structure



- bind_columns()
  - Data being combined should have same number of rows



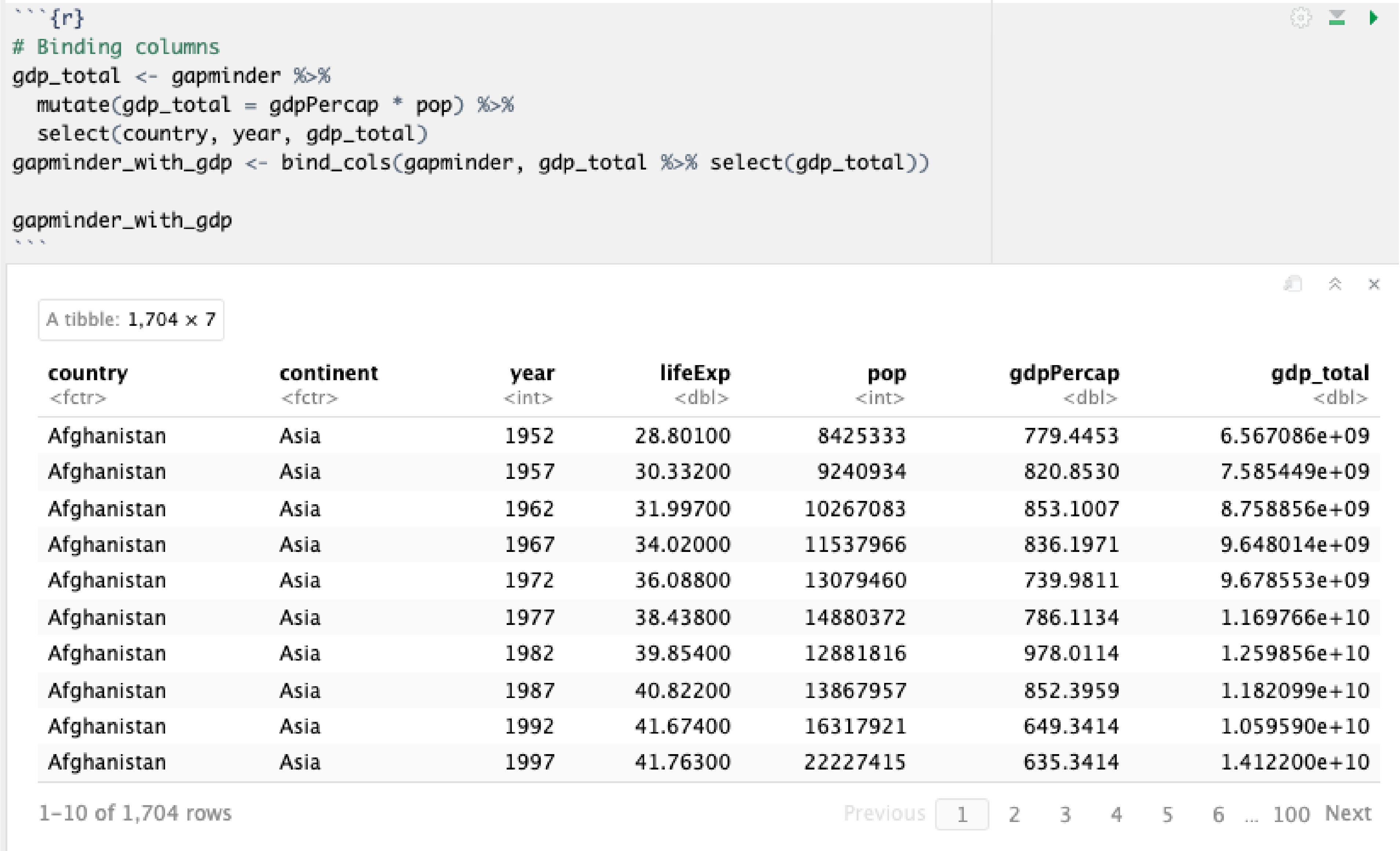


~~~
```{r}
# Binding rows
africa_2007 <- gapminder %>% filter(continent == "Africa", year == 2007)
europe_2007 <- gapminder %>% filter(continent == "Europe", year == 2007)
africa_europe_2007 <- bind_rows(africa_2007, europe_2007)

africa_europe_2007
```
~~~

A tibble: 82 × 6

country <fctr>	continent <fctr>	year <int>	lifeExp <dbl>	pop <int>	gdpPercap <dbl>
Zambia	Africa	2007	42.384	11746035	1271.2116
Zimbabwe	Africa	2007	43.487	12311143	469.7093
Albania	Europe	2007	76.423	3600523	5937.0295
Austria	Europe	2007	79.829	8199783	36126.4927
Belgium	Europe	2007	79.441	10392226	33692.6051
Bosnia and Herzegovina	Europe	2007	74.852	4552198	7446.2988
Bulgaria	Europe	2007	73.005	7322858	10680.7928
Croatia	Europe	2007	75.748	4493312	14619.2227
Czech Republic	Europe	2007	76.486	10228744	22833.3085
Denmark	Europe	2007	78.332	5468120	35278.4187

51-60 of 82 rows



~~~
```{r}
# Binding columns
gdp_total <- gapminder %>%
  mutate(gdp_total = gdpPercap * pop) %>%
  select(country, year, gdp_total)
gapminder_with_gdp <- bind_cols(gapminder, gdp_total %>% select(gdp_total))

gapminder_with_gdp
```
~~~

A tibble: 1,704 x 7

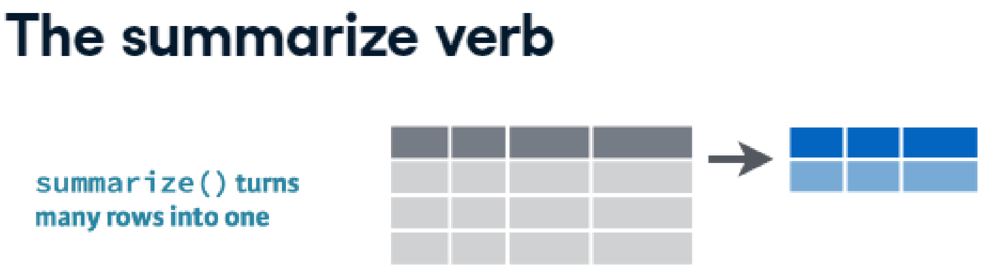
country <fctr>	continent <fctr>	year <int>	lifeExp <dbl>	pop <int>	gdpPercap <dbl>	gdp_total <dbl>
Afghanistan	Asia	1952	28.80100	8425333	779.4453	6.567086e+09
Afghanistan	Asia	1957	30.33200	9240934	820.8530	7.585449e+09
Afghanistan	Asia	1962	31.99700	10267083	853.1007	8.758856e+09
Afghanistan	Asia	1967	34.02000	11537966	836.1971	9.648014e+09
Afghanistan	Asia	1972	36.08800	13079460	739.9811	9.678553e+09
Afghanistan	Asia	1977	38.43800	14880372	786.1134	1.169766e+10
Afghanistan	Asia	1982	39.85400	12881816	978.0114	1.259856e+10
Afghanistan	Asia	1987	40.82200	13867957	852.3959	1.182099e+10
Afghanistan	Asia	1992	41.67400	16317921	649.3414	1.059590e+10
Afghanistan	Asia	1997	41.76300	22227415	635.3414	1.412200e+10

1-10 of 1,704 rows

7. Summarizng

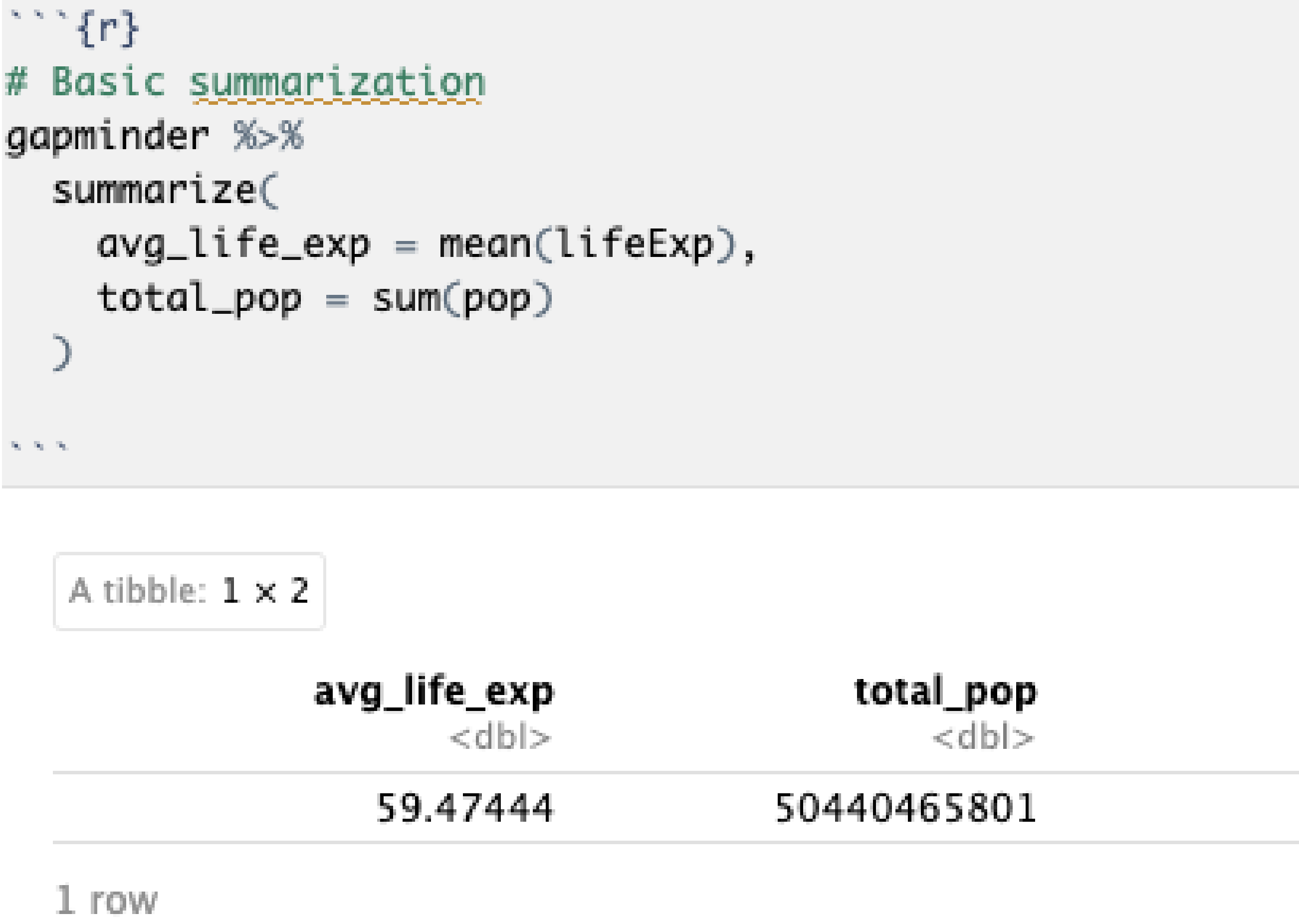
- Collapse groups to single row
  - With some function over the group
    - E.g. mean

- Usecases
  - Computing summary statistics
  - Often used with group_by()



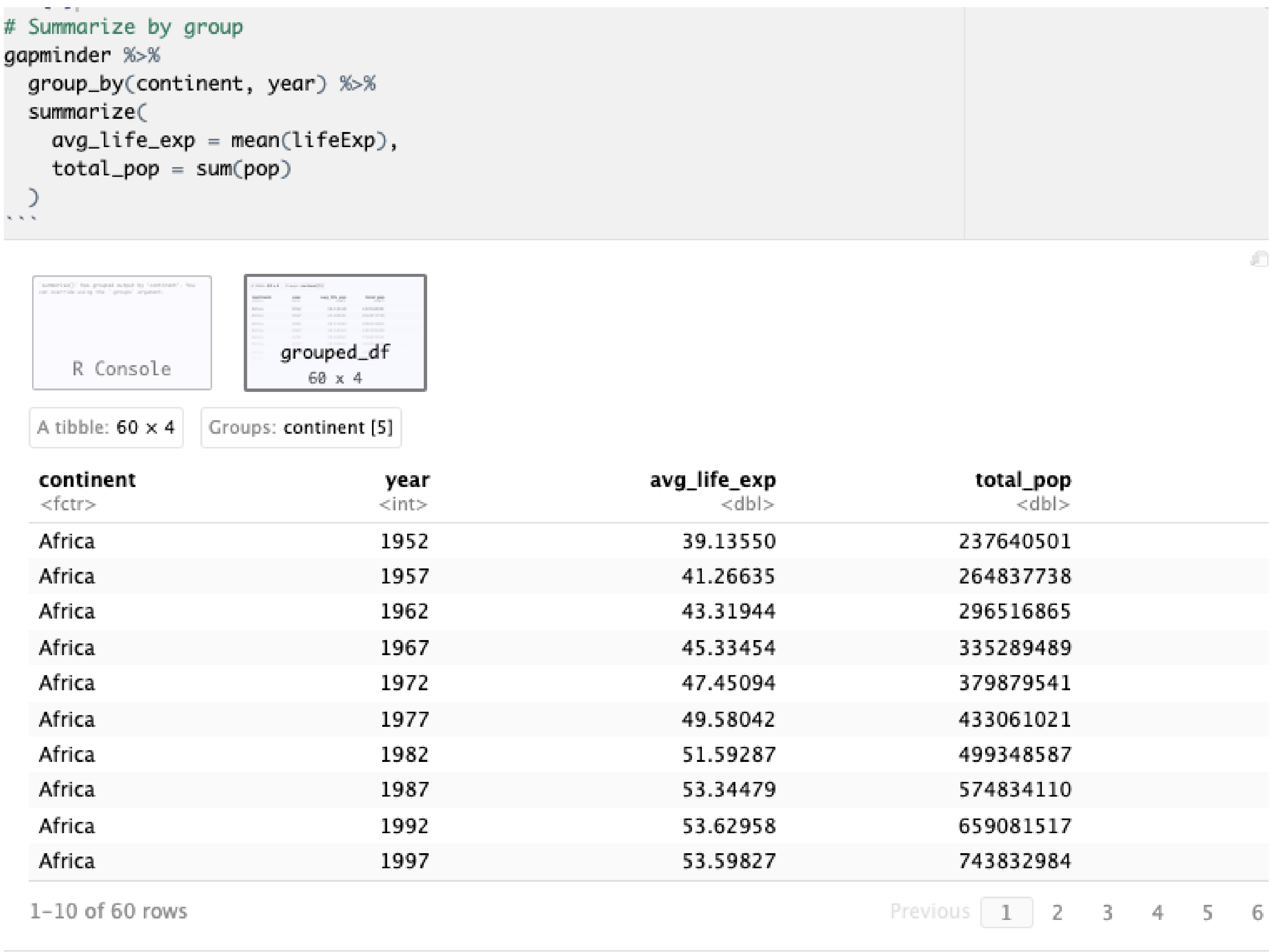
The summarize verb

summarize() turns many rows into one



~~~
```{r}
# Basic summarization
gapminder %>%
  summarize(
    avg_life_exp = mean(lifeExp),
    total_pop = sum(pop)
  )

```
~~~

```
A tibble: 1 × 2

         avg_life_exp       total_pop
                <dbl>           <dbl>
             59.47444     50440465801

1 row
```



~~~r
# Summarize by group
gapminder %>%
  group_by(continent, year) %>%
  summarize(
    avg_life_exp = mean(lifeExp),
    total_pop = sum(pop)
  )
```
~~~

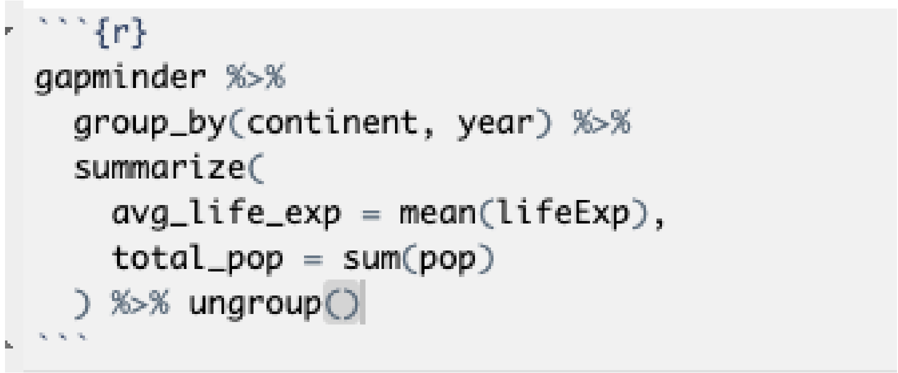
A tibble: 60 × 4

Groups: continent [5]

continent <fctr>	year <int>	avg_life_exp <dbl>	total_pop <dbl>
Africa	1952	39.13550	237640501
Africa	1957	41.26635	264837738
Africa	1962	43.31944	296516865
Africa	1967	45.33454	335289489
Africa	1972	47.45094	379879541
Africa	1977	49.58042	433061021
Africa	1982	51.59287	499348587
Africa	1987	53.34479	574834110
Africa	1992	53.62958	659081517
Africa	1997	53.59827	743832984

1–10 of 60 rows

Ungroup in case you don’t want it to influence subsequent plotting, grouping, etc.

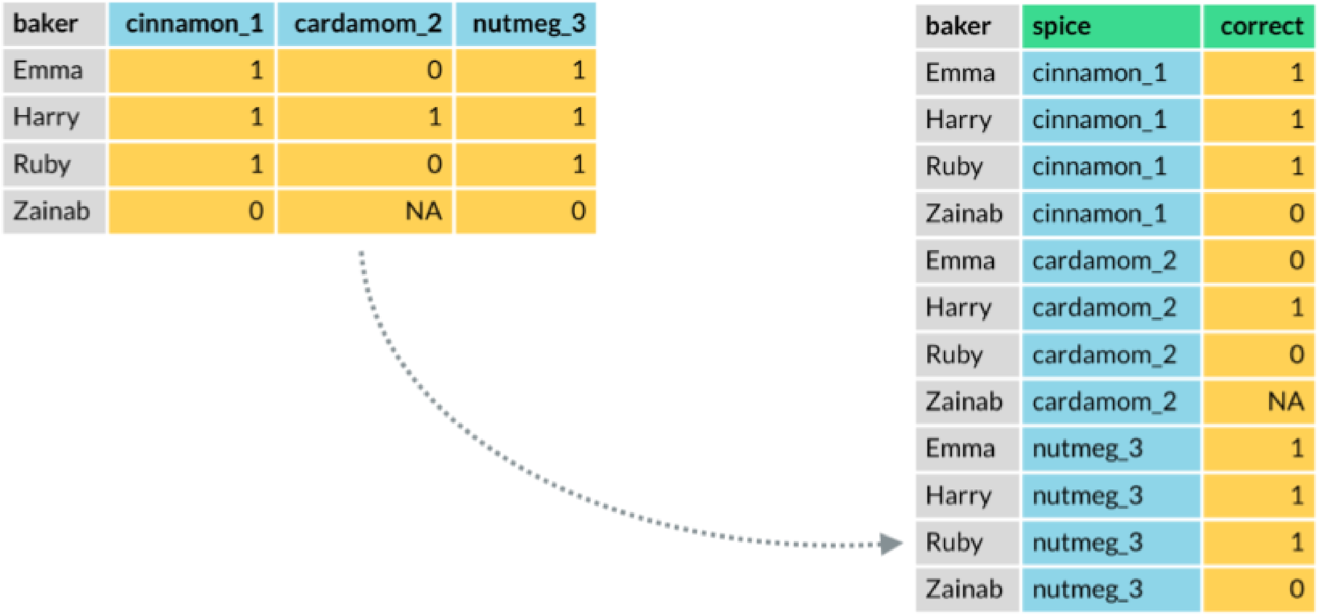


~~~
```{r}
gapminder %>%
  group_by(continent, year) %>%
  summarize(
    avg_life_exp = mean(lifeExp),
    total_pop = sum(pop)
  ) %>% ungroup()
```
~~~

8. Pivot

Two common "tidy" data formats

- wide
- long



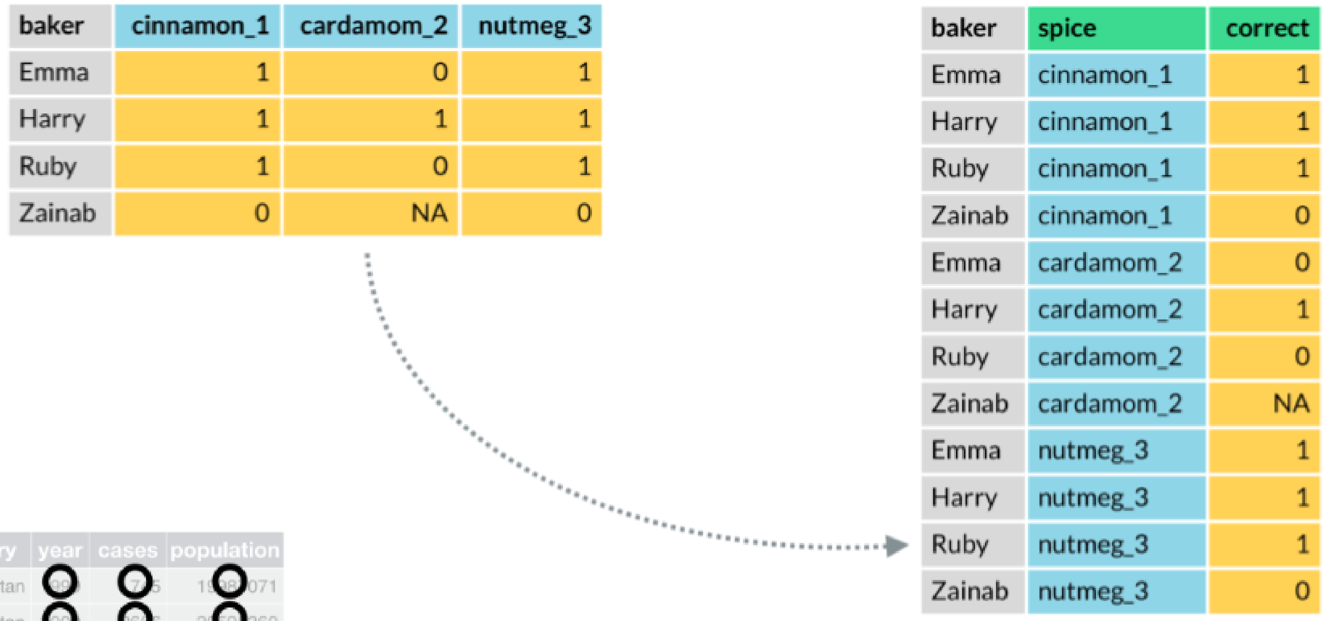
baker	cinnamon_1	cardamom_2	nutmeg_3
Emma	1	0	1
Harry	1	1	1
Ruby	1	0	1
Zainab	0	NA	0

baker	spice	correct
Emma	cinnamon_1	1
Harry	cinnamon_1	1
Ruby	cinnamon_1	1
Zainab	cinnamon_1	0
Emma	cardamom_2	0
Harry	cardamom_2	1
Ruby	cardamom_2	0
Zainab	cardamom_2	NA
Emma	nutmeg_3	1
Harry	nutmeg_3	1
Ruby	nutmeg_3	1
Zainab	nutmeg_3	0

8. Pivot to convert between long and wide

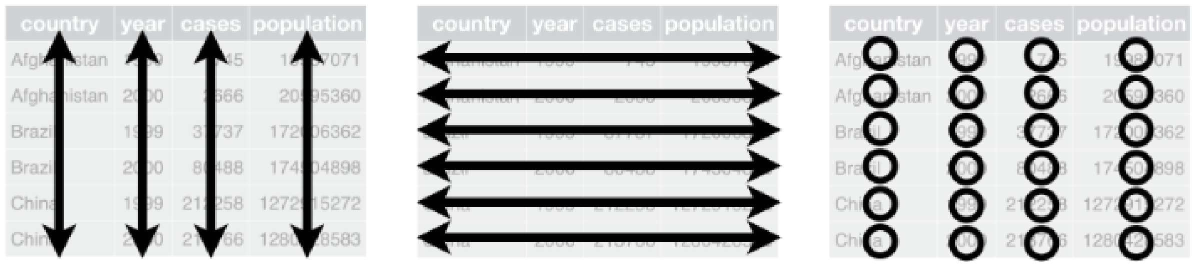
Two common “tidy” data formats

- wide
- long



baker	cinnamon_1	cardamom_2	nutmeg_3
Emma	1	0	1
Harry	1	1	1
Ruby	1	0	1
Zainab	0	NA	0

baker	spice	correct
Emma	cinnamon_1	1
Harry	cinnamon_1	1
Ruby	cinnamon_1	1
Zainab	cinnamon_1	0
Emma	cardamom_2	0
Harry	cardamom_2	1
Ruby	cardamom_2	0
Zainab	cardamom_2	NA
Emma	nutmeg_3	1
Harry	nutmeg_3	1
Ruby	nutmeg_3	1
Zainab	nutmeg_3	0



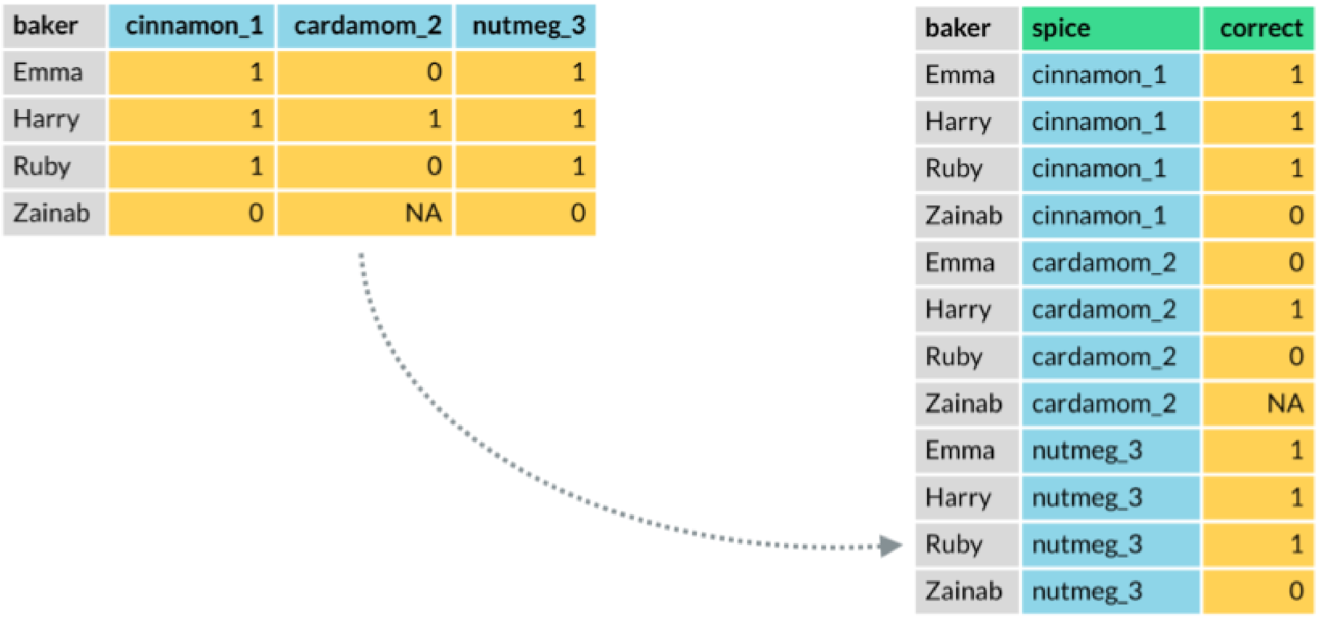
8. Pivot to convert between long and wide

Two common "tidy" data formats

- wide (spread-sheet like)
- long (tidyverse paradigm)

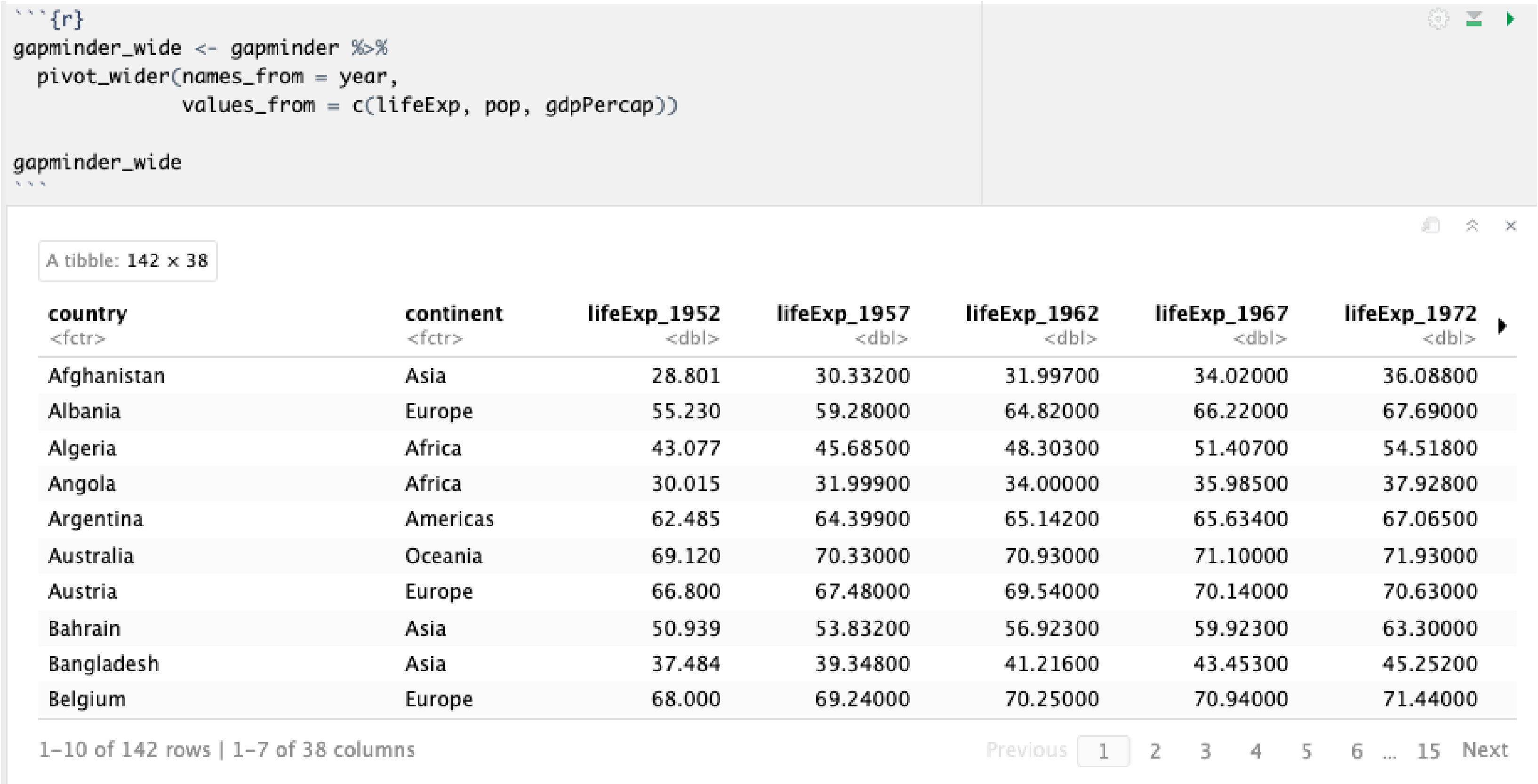
Questions to ask:

- What constitutes an observation in your analysis context?



baker	cinnamon_1	cardamom_2	nutmeg_3
Emma	1	0	1
Harry	1	1	1
Ruby	1	0	1
Zainab	0	NA	0

baker	spice	correct
Emma	cinnamon_1	1
Harry	cinnamon_1	1
Ruby	cinnamon_1	1
Zainab	cinnamon_1	0
Emma	cardamom_2	0
Harry	cardamom_2	1
Ruby	cardamom_2	0
Zainab	cardamom_2	NA
Emma	nutmeg_3	1
Harry	nutmeg_3	1
Ruby	nutmeg_3	1
Zainab	nutmeg_3	0

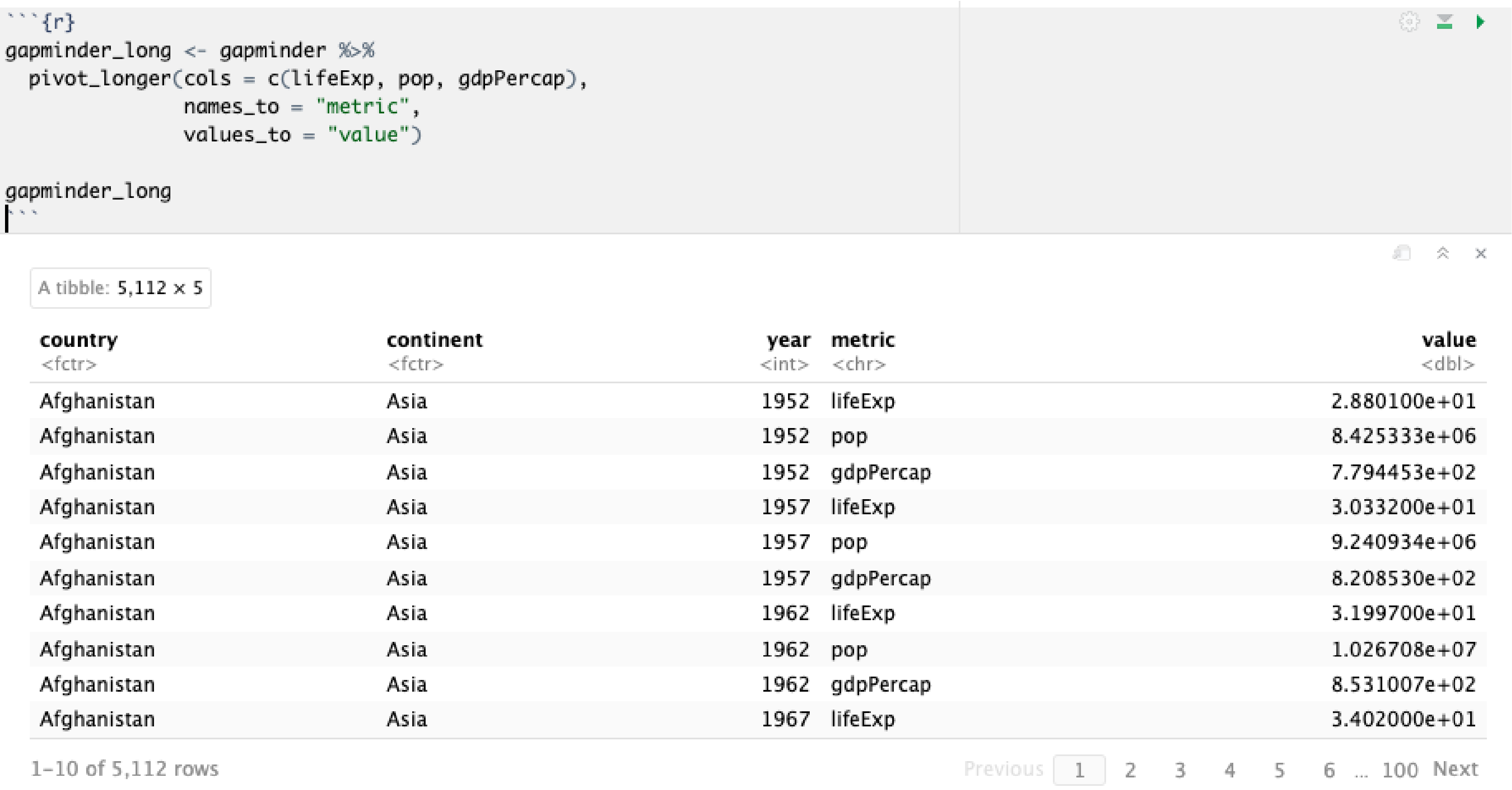


```{r}
gapminder_wide <- gapminder %>%
  pivot_wider(names_from = year,
              values_from = c(lifeExp, pop, gdpPercap))

gapminder_wide
```

A tibble: 142 × 38

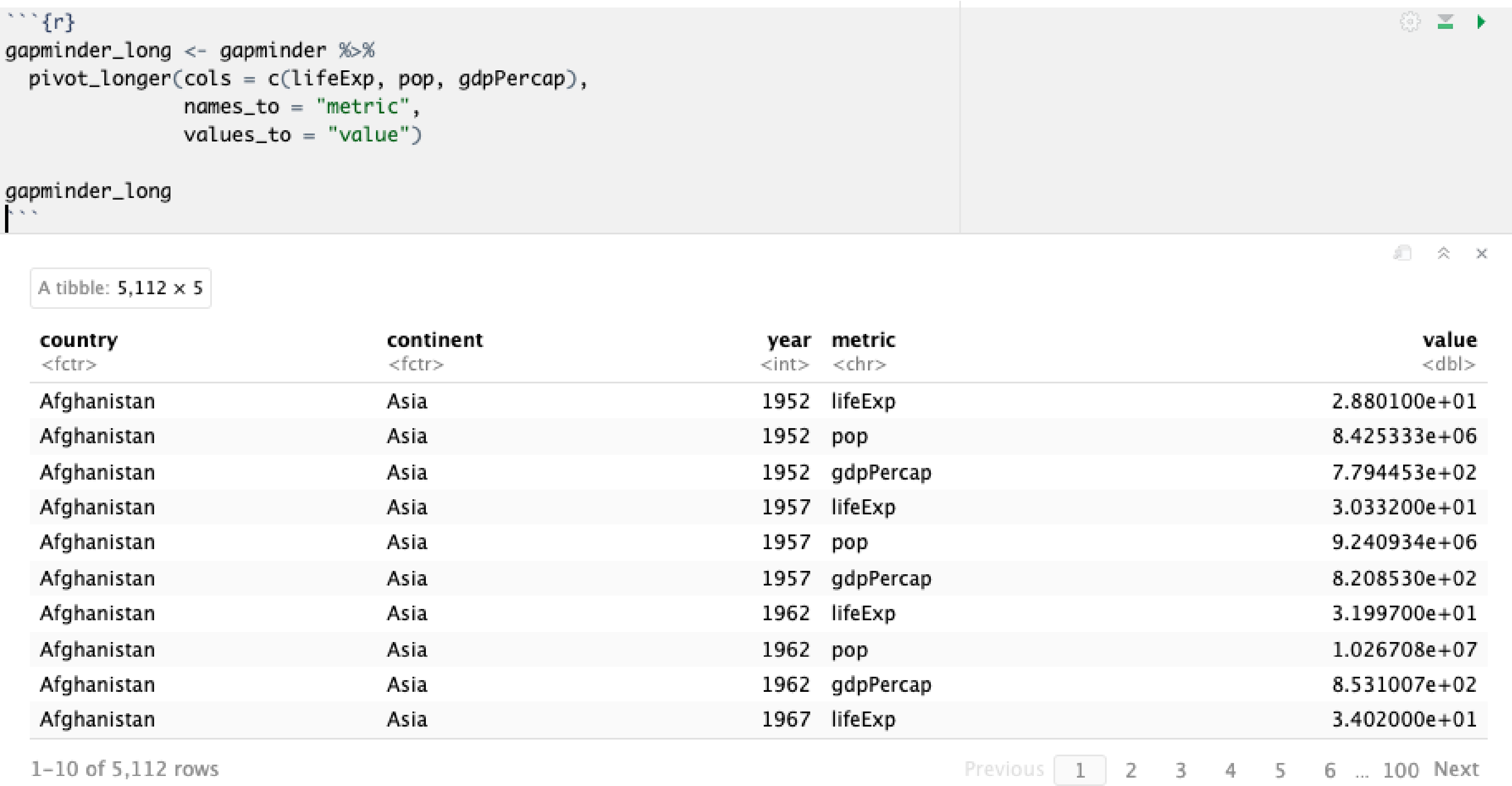
country <fctr>	continent <fctr>	lifeExp_1952 <dbl>	lifeExp_1957 <dbl>	lifeExp_1962 <dbl>	lifeExp_1967 <dbl>	lifeExp_1972 <dbl>
Afghanistan	Asia	28.801	30.33200	31.99700	34.02000	36.08800
Albania	Europe	55.230	59.28000	64.82000	66.22000	67.69000
Algeria	Africa	43.077	45.68500	48.30300	51.40700	54.51800
Angola	Africa	30.015	31.99900	34.00000	35.98500	37.92800
Argentina	Americas	62.485	64.39900	65.14200	65.63400	67.06500
Australia	Oceania	69.120	70.33000	70.93000	71.10000	71.93000
Austria	Europe	66.800	67.48000	69.54000	70.14000	70.63000
Bahrain	Asia	50.939	53.83200	56.92300	59.92300	63.30000
Bangladesh	Asia	37.484	39.34800	41.21600	43.45300	45.25200
Belgium	Europe	68.000	69.24000	70.25000	70.94000	71.44000

1–10 of 142 rows | 1–7 of 38 columns



~~~r
```{r}
gapminder_long <- gapminder %>%
  pivot_longer(cols = c(lifeExp, pop, gdpPercap),
               names_to = "metric",
               values_to = "value")

gapminder_long
~~~
```

A tibble: 5,112 × 5

| country<br>&lt;fctr&gt; | continent<br>&lt;fctr&gt; | year<br>&lt;int&gt; | metric<br>&lt;chr&gt; | value<br>&lt;dbl&gt; |
|---|---|---:|---|---:|
| Afghanistan | Asia | 1952 | lifeExp | 2.880100e+01 |
| Afghanistan | Asia | 1952 | pop | 8.425333e+06 |
| Afghanistan | Asia | 1952 | gdpPercap | 7.794453e+02 |
| Afghanistan | Asia | 1957 | lifeExp | 3.033200e+01 |
| Afghanistan | Asia | 1957 | pop | 9.240934e+06 |
| Afghanistan | Asia | 1957 | gdpPercap | 8.208530e+02 |
| Afghanistan | Asia | 1962 | lifeExp | 3.199700e+01 |
| Afghanistan | Asia | 1962 | pop | 1.026708e+07 |
| Afghanistan | Asia | 1962 | gdpPercap | 8.531007e+02 |
| Afghanistan | Asia | 1967 | lifeExp | 3.402000e+01 |

1–10 of 5,112 rows
```

Common Usecase:
before faceting for ggplot2



~~~
```{r}
gapminder_long <- gapminder %>%
  pivot_longer(cols = c(lifeExp, pop, gdpPercap),
               names_to = "metric",
               values_to = "value")

gapminder_long
```
~~~

A tibble: 5,112 × 5

country <fctr>	continent <fctr>	year <int>	metric <chr>	value <dbl>
Afghanistan	Asia	1952	lifeExp	2.880100e+01
Afghanistan	Asia	1952	pop	8.425333e+06
Afghanistan	Asia	1952	gdpPercap	7.794453e+02
Afghanistan	Asia	1957	lifeExp	3.033200e+01
Afghanistan	Asia	1957	pop	9.240934e+06
Afghanistan	Asia	1957	gdpPercap	8.208530e+02
Afghanistan	Asia	1962	lifeExp	3.199700e+01
Afghanistan	Asia	1962	pop	1.026708e+07
Afghanistan	Asia	1962	gdpPercap	8.531007e+02
Afghanistan	Asia	1967	lifeExp	3.402000e+01

1–10 of 5,112 rows