# Introduction to R

## Lecture 2: Introduction to `tidy` Environment

Swapnil Singh
Lietuvos Bankas | Course Link

# Lecture's Objectives

1. Quick overview of how to transform the data
    - operation on `rows`
    - operation on `columns`
    - operation $=$ verbs
2. Learn about `pipes`
    - Concatenation of verbs through pipes
3. How to work with `groups`

# Illustrative Data

- We are going to use the `gapminder` data

```
head(gapminder)
```

```
## # A tibble: 6 × 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.
```

```
colnames(gapminder)
```

```
## [1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

```
nrow(gapminder)
```

```
## [1] 1704
```

# Illustrative Data

- We are going to use the `gapminder` data

```
glimpse(gapminder)
```

```
## Rows: 1,704
## Columns: 6
## $ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", …
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, …
## $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, …
## $ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8…
## $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12…
## $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, …
```

- Notice the difference between `head` and `glimpse`

# Difference: *tibble* and *dataframe*

- `tibbles` are special types of data frame
- Main differences
  1. tibbles are designed for large datasets
  2. tibble prints are concise: only few rows and only those columns that fit the screen
- To view the full dataset:
  - `View(gapminder)`: interactive dataviewer
  - `print(gapminder, width = Inf)`: show all columns
- **Recommendation**
  - Use the combination of `glimpse()` and `View()` functions

# Data transformation using `dplyr`

- To transform data, you have to rely on `dplyr` package's functions
- Automatically loads with command `library(tidyverse)`
- Huge repository of functions

```
#number of functions in dplyr package
funcsDplyr ← ls('package:dplyr')  #note the package has to be loaded first
numFuncsDplyr  ← length(funcsDplyr)
numFuncsDplyr
```

```
## [1] 297
```

```
#first twenty functions
print(funcsDplyr[1:20])
```

```
##  [1] "%>%"           "across"        "add_count"     "add_count_"    "add_row"
##  [6] "add_rownames"  "add_tally"     "add_tally_"    "all_equal"     "all_of"
## [11] "all_vars"      "anti_join"     "any_of"        "any_vars"      "arrange"
## [16] "arrange_"      "arrange_all"   "arrange_at"    "arrange_if"    "as.tbl"
```

# Data transformation using `dplyr`

- Each function in `dplyr` perform a **single** task
- Data transformation requires multiple task
- Single task $\longrightarrow$ multiple task: use `pipes` i.e. `▷`
- An inscrutable example at this moment

```
newGapminder ← gapminder ▷

  # keep only Asian continent
  filter(continent == "Asia") ▷

  # life  expectancy > 70
  filter(lifeExp > 70)
```

```
glimpse(newGapminder)
```

```
## Rows: 93
## Columns: 6
## $ country   <fct> "Bahrain", "Bahrain", "Bahrain", "Bahrain", "Bahrain", "Chin…
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, …
## $ year      <int> 1987, 1992, 1997, 2002, 2007, 1997, 2002, 2007, 1972, 1977, …
## $ lifeExp   <dbl> 70.750, 72.601, 73.925, 74.795, 75.635, 70.426, 72.028, 72.9…
## $ pop       <int> 454612, 529491, 598561, 656397, 708573, 1230075000, 12804000…
## $ gdpPercap <dbl> 18524.024, 19035.579, 20292.017, 23403.559, 29796.048, 2289.…
```

# Organization of dplyr functions

- Functions can be broadly classified as
  - `row` functions
  - `column` functions
  - `group` functions
  - `tables` functions
- This course we mostly cover the first three
- Let's deal them one by one

# Row functions: *filter*

- `filter`
  - keep only rows satisfying certain criteria

```
#keep only those observation with gdp per capital greater thann 1000
newGapminder ←  gapminder ▷
  filter(lifeExp > 1000)
```

- Operations
  - > greater than
  - >= greater than or equal to
  - < less than
  - <= less than or equal to
  - == equal to
  - != not equal to
- Combination:
  - `&` and
  - `|` or

# Row functions: *filter*

```
gapminder ▷

  # focus only on Asian countries with life expectancy greater than 70
  filter(lifeExp > 70 & continent=='Asia') ▷

  #focus  only on countries with per capita GDP greater than 2000
  filter(gdpPercap > 2000)
```

```
## # A tibble: 91 × 6
##    country          continent  year lifeExp        pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>      <int>     <dbl>
##  1 Bahrain          Asia       1987    70.8     454612    18524.
##  2 Bahrain          Asia       1992    72.6     529491    19036.
##  3 Bahrain          Asia       1997    73.9     598561    20292.
##  4 Bahrain          Asia       2002    74.8     656397    23404.
##  5 Bahrain          Asia       2007    75.6     708573    29796.
##  6 China            Asia       1997    70.4 1230075000     2289.
##  7 China            Asia       2002    72.0 1280400000     3119.
##  8 China            Asia       2007    73.0 1318683096     4959.
##  9 Hong Kong, China Asia       1972    72       4115700     8316.
## 10 Hong Kong, China Asia       1977    73.6     4583700    11186.
## # ℹ 81 more rows
```

# Row functions: *filter*

- Use `%in%` when combining `|` and `==`

```
gapminder ▷
  filter(country == 'Afghanistan' | country == 'Mali' )
```

```
## # A tibble: 24 × 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 14 more rows
```

# Row functions: *filter*

```
gapminder ▷
  filter(country %in% c('Afghanistan', 'Mali') )
```

```
## # A tibble: 24 × 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 14 more rows
```

# Row functions: *filter*

- When filtering, `dplyr::filter` operates on `gapminder` and prints the final results
- `gapminder` data by itself is untouched
- You can store the result in the new `dataframe`

```
newGapminder <-  gapminder %>%

  filter(country %in% c('Afghanistan', 'Mali') )

newGapminder
```

```
## # A tibble: 24 × 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 14 more rows
```

# Row functions: *arrange*

- Change the ordering of rows based on the value of column(s)

```
gapminder ▷
  arrange(year, gdpPercap)
```

```
## # A tibble: 1,704 × 6
##    country           continent  year lifeExp       pop gdpPercap
##    <fct>             <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Lesotho           Africa     1952    42.1    748747      299.
##  2 Guinea-Bissau     Africa     1952    32.5    580653      300.
##  3 Eritrea           Africa     1952    35.9   1438760      329.
##  4 Myanmar           Asia       1952    36.3  20092996      331
##  5 Burundi           Africa     1952    39.0   2445618      339.
##  6 Ethiopia          Africa     1952    34.1  20860941      362.
##  7 Cambodia          Asia       1952    39.4   4693836      368.
##  8 Malawi            Africa     1952    36.3   2917802      369.
##  9 Equatorial Guinea Africa     1952    34.5    216964      376.
## 10 China             Asia       1952    44   556263527      400.
## # ℹ 1,694 more rows
```

# Row functions: *arrange*

- Use `desc` within `arrange` for sorting from largest to smallest
  - Caveat: can use only one argument

```
gapminder ▷
  arrange(desc( gdpPercap))
```

```
## # A tibble: 1,704 × 6
##    country   continent  year lifeExp     pop gdpPercap
##    <fct>     <fct>     <int>   <dbl>   <int>     <dbl>
##  1 Kuwait    Asia       1957    58.0  212846   113523.
##  2 Kuwait    Asia       1972    67.7  841934   109348.
##  3 Kuwait    Asia       1952    55.6  160000   108382.
##  4 Kuwait    Asia       1962    60.5  358266    95458.
##  5 Kuwait    Asia       1967    64.6  575003    80895.
##  6 Kuwait    Asia       1977    69.3 1140357    59265.
##  7 Norway    Europe     2007    80.2 4627926    49357.
##  8 Kuwait    Asia       2007    77.6 2505559    47307.
##  9 Singapore Asia       2007    80.0 4553009    47143.
## 10 Norway    Europe     2002    79.0 4535591    44684.
## # ℹ 1,694 more rows
```

# Row functions: *distinct*

- *distinct* searches and keeps all unique rows in a dataset

```
gapminder ▷
  distinct()
```

```
## # 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
```

# Row functions: *distinct*

- *distinct* on it own is not very useful
- When we want to keep *distinct* combination of some variables, it is very useful

```
gapminder ▷
  distinct(year) ▷
  head(n=4)
```

```
## # A tibble: 4 × 1
##    year
##   <int>
## 1  1952
## 2  1957
## 3  1962
## 4  1967
```

# Row functions: *distinct*

- Note that only `year` column is kept
- To rectify, use `.keep_all = TRUE`

```
gapminder ▷
  distinct(year, .keep_all = TRUE) ▷
  head(n=4)
```

```
## # A tibble: 4 × 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.
```

# Column functions: *mutate*

- *mutate* helps add new columns using existing columns

```
gapminder ▷
  mutate(logGdpPerCapita = log(gdpPercap))
```

```
## # A tibble: 1,704 × 7
##    country     continent  year lifeExp      pop gdpPercap logGdpPerCapita
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>            <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.             6.66
##  2 Afghanistan Asia       1957    30.3  9240934      821.             6.71
##  3 Afghanistan Asia       1962    32.0 10267083      853.             6.75
##  4 Afghanistan Asia       1967    34.0 11537966      836.             6.73
##  5 Afghanistan Asia       1972    36.1 13079460      740.             6.61
##  6 Afghanistan Asia       1977    38.4 14880372      786.             6.67
##  7 Afghanistan Asia       1982    39.9 12881816      978.             6.89
##  8 Afghanistan Asia       1987    40.8 13867957      852.             6.75
##  9 Afghanistan Asia       1992    41.7 16317921      649.             6.48
## 10 Afghanistan Asia       1997    41.8 22227415      635.             6.45
## # ℹ 1,694 more rows
```

# Column functions: *mutate*

- Imagine you have a dataset of `500` columns
- You do a `mutate` and then `View` it
- By default new column is added to the last
- Use `.before` to put the new column first

```
gapminder ▷
  mutate(logGdpPerCapita = log(gdpPercap), .before=1)
```

```
## # A tibble: 1,704 × 7
##    logGdpPerCapita country     continent  year lifeExp      pop gdpPercap
##              <dbl> <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1            6.66 Afghanistan Asia       1952    28.8  8425333      779.
##  2            6.71 Afghanistan Asia       1957    30.3  9240934      821.
##  3            6.75 Afghanistan Asia       1962    32.0 10267083      853.
##  4            6.73 Afghanistan Asia       1967    34.0 11537966      836.
##  5            6.61 Afghanistan Asia       1972    36.1 13079460      740.
##  6            6.67 Afghanistan Asia       1977    38.4 14880372      786.
##  7            6.89 Afghanistan Asia       1982    39.9 12881816      978.
##  8            6.75 Afghanistan Asia       1987    40.8 13867957      852.
##  9            6.48 Afghanistan Asia       1992    41.7 16317921      649.
## 10            6.45 Afghanistan Asia       1997    41.8 22227415      635.
## # ℹ 1,694 more rows
```

# Column functions: *mutate*

- Can use `.after` also

```
gapminder ▷
  mutate(logGdpPerCapita = log(gdpPercap), .after=year)
```

```
## # A tibble: 1,704 × 7
##    country     continent  year logGdpPerCapita lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>           <dbl>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952            6.66    28.8  8425333      779.
##  2 Afghanistan Asia       1957            6.71    30.3  9240934      821.
##  3 Afghanistan Asia       1962            6.75    32.0 10267083      853.
##  4 Afghanistan Asia       1967            6.73    34.0 11537966      836.
##  5 Afghanistan Asia       1972            6.61    36.1 13079460      740.
##  6 Afghanistan Asia       1977            6.67    38.4 14880372      786.
##  7 Afghanistan Asia       1982            6.89    39.9 12881816      978.
##  8 Afghanistan Asia       1987            6.75    40.8 13867957      852.
##  9 Afghanistan Asia       1992            6.48    41.7 16317921      649.
## 10 Afghanistan Asia       1997            6.45    41.8 22227415      635.
## # ℹ 1,694 more rows
```

# Column functions: *select*

- Select columns by name

```
gapminder ▷
  select(country,year, lifeExp)
```

```
## # A tibble: 1,704 × 3
##    country      year lifeExp
##    <fct>       <int>   <dbl>
##  1 Afghanistan  1952    28.8
##  2 Afghanistan  1957    30.3
##  3 Afghanistan  1962    32.0
##  4 Afghanistan  1967    34.0
##  5 Afghanistan  1972    36.1
##  6 Afghanistan  1977    38.4
##  7 Afghanistan  1982    39.9
##  8 Afghanistan  1987    40.8
##  9 Afghanistan  1992    41.7
## 10 Afghanistan  1997    41.8
## # ℹ 1,694 more rows
```

# Column functions: *select*

```
colnames(gapminder)
```

```
## [1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

```
# select columns between country and pop, inclusive
# don't recommend this though
gapminder ▷
  select(country:pop)
```

```
## # A tibble: 1,704 × 5
##    country     continent  year lifeExp      pop
##    <fct>       <fct>     <int>   <dbl>    <int>
##  1 Afghanistan Asia       1952    28.8  8425333
##  2 Afghanistan Asia       1957    30.3  9240934
##  3 Afghanistan Asia       1962    32.0 10267083
##  4 Afghanistan Asia       1967    34.0 11537966
##  5 Afghanistan Asia       1972    36.1 13079460
##  6 Afghanistan Asia       1977    38.4 14880372
##  7 Afghanistan Asia       1982    39.9 12881816
##  8 Afghanistan Asia       1987    40.8 13867957
##  9 Afghanistan Asia       1992    41.7 16317921
## 10 Afghanistan Asia       1997    41.8 22227415
## # ℹ 1,694 more rows
```

# Column functions: *select*

- Select based on column characteristics

```
# select only numeric columns
gapminder ▷
  select(where(is.numeric))
```

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

# Column functions: *select*

- select columns which starts with some patterns

```
gapminder ▷
  select(starts_with('co'))
```

```
## # A tibble: 1,704 × 2
##    country     continent
##    <fct>       <fct>
##  1 Afghanistan Asia
##  2 Afghanistan Asia
##  3 Afghanistan Asia
##  4 Afghanistan Asia
##  5 Afghanistan Asia
##  6 Afghanistan Asia
##  7 Afghanistan Asia
##  8 Afghanistan Asia
##  9 Afghanistan Asia
## 10 Afghanistan Asia
## # ℹ 1,694 more rows
```

# Column functions: *select*

- select columns which ends with some patterns

```
gapminder ▷
  select(ends_with('p'))
```

```
## # A tibble: 1,704 × 3
##    lifeExp      pop gdpPercap
##      <dbl>    <int>     <dbl>
##  1    28.8  8425333      779.
##  2    30.3  9240934      821.
##  3    32.0 10267083      853.
##  4    34.0 11537966      836.
##  5    36.1 13079460      740.
##  6    38.4 14880372      786.
##  7    39.9 12881816      978.
##  8    40.8 13867957      852.
##  9    41.7 16317921      649.
## 10    41.8 22227415      635.
## # ℹ 1,694 more rows
```

# Column functions: *select*

- select columns which contains some patterns

```
gapminder ▷
  select(contains('t'))
```

```
## # A tibble: 1,704 × 2
##    country     continent
##    <fct>       <fct>
##  1 Afghanistan Asia
##  2 Afghanistan Asia
##  3 Afghanistan Asia
##  4 Afghanistan Asia
##  5 Afghanistan Asia
##  6 Afghanistan Asia
##  7 Afghanistan Asia
##  8 Afghanistan Asia
##  9 Afghanistan Asia
## 10 Afghanistan Asia
## # ℹ 1,694 more rows
```

# Column functions: *rename*

```
gapminder ▷
  rename(life_expectancy = lifeExp)
```

```
## # A tibble: 1,704 × 6
##    country     continent  year life_expectancy      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
```

# Combination of *pipes*

```
gapminder ▷
  # keep only Asian continent
  filter(continent == "Asia") ▷
  #select some columns
  select(country, lifeExp, year) ▷
  #take the log of lifeExp
  mutate(logLifeExp = log(lifeExp), .after=lifeExp)
```

```
## # A tibble: 396 × 4
##    country     lifeExp logLifeExp  year
##    <fct>         <dbl>      <dbl> <int>
##  1 Afghanistan    28.8       3.36  1952
##  2 Afghanistan    30.3       3.41  1957
##  3 Afghanistan    32.0       3.47  1962
##  4 Afghanistan    34.0       3.53  1967
##  5 Afghanistan    36.1       3.59  1972
##  6 Afghanistan    38.4       3.65  1977
##  7 Afghanistan    39.9       3.69  1982
##  8 Afghanistan    40.8       3.71  1987
##  9 Afghanistan    41.7       3.73  1992
## 10 Afghanistan    41.8       3.73  1997
## # ℹ 386 more rows
```

# Working with *groups*

- We learned how to use functions for *rows* or *columns*
- What can we do, if we want to operate these functions within *groups*?
  - Which country has highest life expectancy in **Asia**?
  - Which country has lowest life expectancy in **Africa**?
- We use `group_by()` function

```
gapminder ▷
  group_by(continent)
```

```
## # A tibble: 1,704 × 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.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
```

# Working with *groups*

- Once you used `group_by()`, any row operations are applicable within each group

```
gapminder ▷
  group_by(continent) ▷
  summarize(avgLifeExp = mean(lifeExp, na.rm = TRUE))
```

```
## # A tibble: 5 × 2
##   continent avgLifeExp
##   <fct>          <dbl>
## 1 Africa          48.9
## 2 Americas        64.7
## 3 Asia            60.1
## 4 Europe          71.9
## 5 Oceania         74.3
```

# Working with *groups*

- More elaborate summary

```
gapminder ▷
  group_by(continent) ▷
  summarize(avgLifeExp = mean(lifeExp, na.rm = TRUE),
            medianLifeExp = median(lifeExp, na.rm=TRUE),
            numObs = n(),
            avgGdpPerCap = mean(gdpPercap, na.rm = TRUE))
```

```
## # A tibble: 5 × 5
##   continent avgLifeExp medianLifeExp numObs avgGdpPerCap
##   <fct>          <dbl>         <dbl>  <int>        <dbl>
## 1 Africa          48.9          47.8    624        2194.
## 2 Americas        64.7          67.0    300        7136.
## 3 Asia            60.1          61.8    396        7902.
## 4 Europe          71.9          72.2    360       14469.
## 5 Oceania         74.3          73.7     24       18622.
```

# Multiple *groups*

- Nothing stops you from using multiple groups

```
gapminder ▷
  group_by(continent, year)
```

```
## # A tibble: 1,704 × 6
## # Groups:   continent, year [60]
##    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
```

# Coding Exercise

# World Bank Fiscal Space Data

- Download World Bank's Fiscal Space data: link
- Unzip it and put it in folder where you can access it
- Download both `STATA` and `Excel` files
  - `Excel` file has glossary

```
fspace ← haven::read_dta(file = paste0('raw_data/', 'Fiscal-space-data.dta'))

glimpse(fspace)
```

```
## Rows: 6,666
## Columns: 31
## $ ccode     <chr> "USA", "USA", "USA", "USA", "USA", "USA", "USA", "USA", "USA…
## $ ifscode   <dbl> 111, 111, 111, 111, 111, 111, 111, 111, 111, 111, 111, 111, …
## $ country   <chr> "United States", "United States", "United States", "United S…
## $ group     <dbl+lbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
## $ region    <dbl+lbl> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,…
## $ inc       <dbl+lbl> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,…
## $ year      <dbl> 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, …
## $ ggdy      <dbl> 61.71774, 66.34741, 68.65210, 70.56248, 69.38563, 68.96503, …
## $ pby       <dbl> -0.4214212, -0.8911726, -1.8340608, -1.1126696, 0.1118852, 0…
## $ cby       <dbl> -4.4734335, -4.3848381, -5.5710871, -4.6584245, -3.5298614, …
## $ fby       <dbl> -4.1060773, -4.9237521, -5.9139839, -5.0755329, -3.6825231, …
## $ dfggd     <dbl> 318.1225, 341.9860, 353.8655, 363.7125, 357.6465, 355.4785, …
```

# World Bank Fiscal Space Data

- Count the number of countries in each year and store it in a `dataframe` called `fspaceNumCountries`

```
fspaceNumCountries ←  fspace ▷
  group_by(year) ▷
  summarize(numCountries  = n())

fspaceNumCountries
```

```
## # A tibble: 33 × 2
##     year numCountries
##    <dbl>        <int>
##  1  1990          202
##  2  1991          202
##  3  1992          202
##  4  1993          202
##  5  1994          202
##  6  1995          202
##  7  1996          202
##  8  1997          202
##  9  1998          202
## 10  1999          202
## # ℹ 23 more rows
```

# World Bank Fiscal Space Data

- Compute the mean and median of following variables each year
  - general government gross debt (`ggdy`)
  - general government debt in foreign currency (`fxsovsh`)
  - total external debt in stocks (`xtdebty`)

```
fspaceSummary ← fspace ▷
  group_by(year) ▷
  summarise(numCountries = n(),
            mean_ggdy = mean(ggdy, na.rm=TRUE),
            mean_fxsovsh = mean(fxsovsh, na.rm=TRUE),
            mean_xtdebty = mean(xtdebty, na.rm=TRUE),
            median_ggdy = median(ggdy, na.rm=TRUE),
            median_fxsovsh = median(fxsovsh, na.rm=TRUE),
            median_xtdebty = median(xtdebty, na.rm=TRUE))
```

# World Bank Fiscal Space Data

```
glimpse(fspaceSummary)
```

```
## Rows: 33
## Columns: 8
## $ year           <dbl> 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1…
## $ numCountries   <int> 202, 202, 202, 202, 202, 202, 202, 202, 202, 202, 202, …
## $ mean_ggdy      <dbl> 69.20511, 66.70285, 70.51369, 68.55063, 70.04780, 58.74…
## $ mean_fxsovsh   <dbl> 0.1345489, 0.1247254, 0.1083296, 0.1014340, 0.1026522, …
## $ mean_xtdebty   <dbl> 69.90365, 76.24725, 66.24232, 65.76774, 70.31518, 63.80…
## $ median_ggdy    <dbl> 63.0220, 62.2260, 60.5550, 60.6290, 60.4210, 55.1510, 5…
## $ median_fxsovsh <dbl> 0.1345489, 0.1247254, 0.1083296, 0.1014340, 0.1026522, …
## $ median_xtdebty <dbl> 46.19101, 49.57336, 46.39834, 48.51647, 50.44743, 47.92…
```

# World Bank Fiscal Space Data

- Construct a new dataset which contains following variables
  - `year`
  - `country`
  - `ggdy`
  - `fxsovsh`
  - `xtdebty`

```
fspaceNew ← fspace ▷
  select(year, country, ggdy, fxsovsh, xtdebty)
glimpse(fspaceNew)
```

```
## Rows: 6,666
## Columns: 5
## $ year    <dbl> 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 20…
## $ country <chr> "United States", "United States", "United States", "United Sta…
## $ ggdy    <dbl> 61.71774, 66.34741, 68.65210, 70.56248, 69.38563, 68.96503, 68…
## $ fxsovsh <dbl> 0.13454895, 0.12472536, 0.10832959, 0.10143403, 0.10265220, 0.…
## $ xtdebty <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 60.69567, …
```

# World Bank Fiscal Space Data

- Construct a new variable which is the logarithm of `ggdy`

```
fspaceNew ← fspaceNew ▷
  mutate(log_ggdy = log(ggdy))

glimpse(fspaceNew)
```

```
## Rows: 6,666
## Columns: 6
## $ year     <dbl> 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2…
## $ country  <chr> "United States", "United States", "United States", "United St…
## $ ggdy     <dbl> 61.71774, 66.34741, 68.65210, 70.56248, 69.38563, 68.96503, 6…
## $ fxsovsh  <dbl> 0.13454895, 0.12472536, 0.10832959, 0.10143403, 0.10265220, 0…
## $ xtdebty  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 60.69567,…
## $ log_ggdy <dbl> 4.122571, 4.194905, 4.229052, 4.256499, 4.239680, 4.233599, 4…
```

# World Bank Fiscal Space Data

- Keep only those observations where `log_ggdy > 5`

```
fspaceNew ← fspaceNew ▷
  filter(log_ggdy > 5)

glimpse(fspaceNew)
```

```
## Rows: 174
## Columns: 6
## $ year     <dbl> 2020, 2021, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2…
## $ country  <chr> "Italy", "Italy", "Japan", "Japan", "Japan", "Japan", "Japan"…
## $ ggdy     <dbl> 154.890, 149.809, 154.096, 160.021, 169.490, 174.594, 174.142…
## $ fxsovsh  <dbl> 0.1160312, 0.1195998, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
## $ xtdebty  <dbl> 149.98938, 131.50735, NA, 29.96744, 31.82130, 31.48246, 32.87…
## $ log_ggdy <dbl> 5.042715, 5.009361, 5.037576, 5.075305, 5.132794, 5.162463, 5…
```

# World Bank Fiscal Space Data

- Do all operations we discussed till now in one go

```
fspaceNew ← fspace ▷
  select(year, country, ggdy, fxsovsh, xtdebty) ▷
  mutate(log_ggdy = log(ggdy)) ▷
  filter(log_ggdy > 5)
glimpse(fspaceNew)
```

```
## Rows: 174
## Columns: 6
## $ year     <dbl> 2020, 2021, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2…
## $ country  <chr> "Italy", "Italy", "Japan", "Japan", "Japan", "Japan", "Japan"…
## $ ggdy     <dbl> 154.890, 149.809, 154.096, 160.021, 169.490, 174.594, 174.142…
## $ fxsovsh  <dbl> 0.1160312, 0.1195998, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
## $ xtdebty  <dbl> 149.98938, 131.50735, NA, 29.96744, 31.82130, 31.48246, 32.87…
## $ log_ggdy <dbl> 5.042715, 5.009361, 5.037576, 5.075305, 5.132794, 5.162463, 5…
```