Manipulating a single data frame with `dplyr`

cinkova@ufal.mff.cuni.cz

`dplyr`: Operations on data frames

- on rows
- on columns

> **Tip**
>
> **https://dplyr.tidyverse.org/reference/index.html**
>
> download and print the cheatsheet!
>
> **https://github.com/rstudio/cheatsheets/blob/main/data-transformation.pdf**

Subset rows

- `filter()`
- `slice()`
- `distinct()`
- `arrange()`
- `group_by()`

Subset columns

- `select()`` (+ `rename()`, `relocate()` & Co.)
  `starts_with()`, `ends_with()`, `where()` … : “helper” functions selecting columns according to names; they only work inside the big functions
- `n()` other “helper” functions computing
- `pull` extracts one column as **vector**
- `glimpse()` displays all columns with a few values and a summary

Compute values new columns/modify original columns

- `mutate()`
- `add_count()`

Create an aggregated table from the original

- `summarize()`

`magrittr` pipe in `dplyr`: `%>%`

René Magritte 1898-1967: Belgian surrealist fascinated by semiotics

`magrittr`: R library providing workflow pipes named to his tribute.

`select` columns

https://dplyr.tidyverse.org/reference/select.html

This is `gapminder`.

```
1 gapminder %>% slice_head(n = 2)
```

```
# A tibble: 2 × 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.
```

`select` positively

```
gapminder %>% slice_head(n = 2) %>%
  select(c(1,4:6))
```

```
# A tibble: 2 × 4
  country     lifeExp     pop gdpPercap
  <fct>         <dbl>   <int>     <dbl>
1 Afghanistan    28.8 8425333      779.
2 Afghanistan    30.3 9240934      821.
```

```
# same as
gapminder %>% slice_head(n = 2) %>%
  select(c(country, lifeExp, pop, gdpPercap))
```

```
# A tibble: 2 × 4
  country     lifeExp     pop gdpPercap
  <fct>         <dbl>   <int>     <dbl>
1 Afghanistan    28.8 8425333      779.
2 Afghanistan    30.3 9240934      821.
```

`select` negatively

```
gapminder %>% slice_head(n = 2) %>%
  select(!c(2:3))
```

```
# A tibble: 2 × 4
  country     lifeExp     pop gdpPercap
  <fct>         <dbl>   <int>     <dbl>
1 Afghanistan    28.8 8425333      779.
2 Afghanistan    30.3 9240934      821.
```

```
# same as
gapminder %>% slice_head(n = 2) %>%
  select(!c(continent, year))
```

```
# A tibble: 2 × 4
  country     lifeExp     pop gdpPercap
  <fct>         <dbl>   <int>     <dbl>
1 Afghanistan    28.8 8425333      779.
2 Afghanistan    30.3 9240934      821.
```

```
# earlier you used minus instead of exclamation mark
```

Helper functions in columns

They work only **inside** the big functions (`select, mutate, summarize`...)

```
gapminder %>% slice_head(n = 2) %>%
  select(starts_with("co"))
```

```
# A tibble: 2 × 2
  country     continent
  <fct>       <fct>
1 Afghanistan Asia
2 Afghanistan Asia
```

```
# and similarly
gapminder %>% slice_head(n = 2) %>%
  select(ends_with("p"))
```

```
# A tibble: 2 × 3
  lifeExp     pop gdpPercap
    <dbl>   <int>     <dbl>
1    28.8 8425333      779.
2    30.3 9240934      821.
```

Helper functions in columns

on column names, like the previous one

```
# this one accepts regular expressions (like "wildcards")
gapminder %>% slice_head(n = 2) %>%
  select(matches("ou?n"))
```

```
# A tibble: 2 × 2
  country     continent
  <fct>       <fct>
1 Afghanistan Asia
2 Afghanistan Asia
```

Helper functions in columns

- on column values - typically you check vector class
- ***Formula notation:*** *~ somefunction(.x, …)*

```
gapminder %>% slice_head(n = 2) %>%
  select(where(is.numeric))
```

```
# A tibble: 2 × 4
   year lifeExp     pop gdpPercap
  <int>   <dbl>   <int>     <dbl>
1  1952    28.8 8425333      779.
2  1957    30.3 9240934      821.
```

```
# short for this formula notation
gapminder %>% slice_head(n = 2) %>%
  select(where(~ is.numeric(.x)))
```

```
# A tibble: 2 × 4
   year lifeExp     pop gdpPercap
  <int>   <dbl>   <int>     <dbl>
1  1952    28.8 8425333      779.
2  1957    30.3 9240934      821.
```

rather just for fun: feed it almost any function that works for all columns (no error)

```
gapminder %>% slice_head(n = 2) %>%
  select(where(~ is.numeric(.x))) %>%
  select(where(~ max(.x) > 100))
```

```
# A tibble: 2 × 3
```

```
   year     pop gdpPercap
  <int>   <int>     <dbl>
1  1952 8425333      779.
2  1957 9240934      821.
```

Helper functions in columns

select the last column of so many that you don't want to count

```
gapminder %>% slice_head(n = 2) %>%
  select(ncol(.))
```

```
# A tibble: 2 × 1
  gdpPercap
      <dbl>
1      779.
2      821.
```

```
# and here comes a convenient helper:
gapminder %>% slice_head(n = 2) %>%
  select(last_col())
```

```
# A tibble: 2 × 1
  gdpPercap
      <dbl>
1      779.
2      821.
```

Rename columns with `select`

- `select` keeps just the explicitly mentioned columns.
- Use the helper `everything()`

```
gapminder %>% slice_head(n = 2) %>%
  select(STATE = country, POPULATION = pop )
```

```
# A tibble: 2 × 2
  STATE       POPULATION
  <fct>            <int>
1 Afghanistan    8425333
2 Afghanistan    9240934
```

```
gapminder %>% slice_head(n = 2) %>%
  select(STATE = country, everything() )
```

```
# A tibble: 2 × 6
  STATE       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.
```

`rename` keeps all columns

```
gapminder %>% slice_head(n = 2) %>%
  rename(STATE = country, POPULATION = pop )
```

```
# A tibble: 2 × 6
  STATE       continent  year lifeExp POPULATION gdpPercap
  <fct>       <fct>     <int>   <dbl>      <int>     <dbl>
1 Afghanistan Asia       1952    28.8    8425333      779.
2 Afghanistan Asia       1957    30.3    9240934      821.
```

`mutate` adds a new column

```
gapminder %>% slice_head(n = 2) %>%
  mutate(new_column = NA)
```

```
# A tibble: 2 × 7
  country     continent  year lifeExp     pop gdpPercap new_column
  <fct>       <fct>     <int>   <dbl>   <int>     <dbl> <lgl>
1 Afghanistan Asia       1952    28.8 8425333      779. NA
2 Afghanistan Asia       1957    30.3 9240934      821. NA
```

```
gapminder %>% slice_head(n = 2) %>%
  mutate(GDP = pop * gdpPercap)
```

```
# A tibble: 2 × 7
  country     continent  year lifeExp     pop gdpPercap         GDP
  <fct>       <fct>     <int>   <dbl>   <int>     <dbl>       <dbl>
1 Afghanistan Asia       1952    28.8 8425333      779. 6567086330.
2 Afghanistan Asia       1957    30.3 9240934      821. 7585448670.
```

grouping with `mutate`

Depends on function: does it consider the whole column or row after row?

When you need row by row: `rowwise() %>% mutate(...`

`mutate` values of an existing column

use helper function `across()`
mind the formula notation inside

```
gapminder %>% slice_head(n = 2) %>%
  mutate(across(.cols = c(country, continent), ~ toupper(.x)))
```

```
# A tibble: 2 × 6
  country     continent  year lifeExp     pop gdpPercap
  <chr>       <chr>     <int>   <dbl>   <int>     <dbl>
1 AFGHANISTAN ASIA       1952    28.8 8425333      779.
2 AFGHANISTAN ASIA       1957    30.3 9240934      821.
```

```
# what is in .cols becomes .x in the formula
# cannot leave .x out, unlike in the pipe
```

```
# but whenever .x is the ONLY function argument you mention,
# possible shortcut is using just function name,
# without brackets/quotes/~
gapminder %>% slice_head(n = 2) %>%
  mutate(across(.cols = c(country, continent), toupper))
```

```
# A tibble: 2 × 6
  country     continent  year lifeExp     pop gdpPercap
  <chr>       <chr>     <int>   <dbl>   <int>     <dbl>
1 AFGHANISTAN ASIA       1952    28.8 8425333      779.
2 AFGHANISTAN ASIA       1957    30.3 9240934      821.
```

Helper functions for `mutate` (small selection!)

- add row numbers in a new column - good when you need unique IDs!

```
gapminder %>% slice_head(n = 2) %>%
  mutate(ID = row_number()) %>%
  select(ID, everything())
```

```
# A tibble: 2 × 7
     ID country     continent  year lifeExp     pop gdpPercap
  <int> <fct>       <fct>     <int>   <dbl>   <int>     <dbl>
1     1 Afghanistan Asia       1952    28.8 8425333      779.
2     2 Afghanistan Asia       1957    30.3 9240934      821.
```

Helper functions for `mutate`: `if_else`

```
set.seed(12)
gapminder %>% slice_sample(n = 5) %>% select(1:2) %>%
  mutate(just_Asia = if_else(continent == "Asia",
                             true = country, false = NA))
```

```
# A tibble: 5 × 3
  country          continent just_Asia
  <fct>            <fct>     <fct>
1 Ecuador          Americas  <NA>
2 Congo, Rep.      Africa    <NA>
3 Congo, Dem. Rep. Africa    <NA>
4 Reunion          Africa    <NA>
5 Kuwait           Asia      Kuwait
```

Helper functions for `mutate` : `if_else`

combined with `across`

```
set.seed(122)
gapminder %>% slice_sample(n = 5) %>% select(1:2) %>%
  mutate(across(continent,
                ~ if_else(.x == "Africa", true = country, false = NA)))
```

```
# A tibble: 5 × 2
  country       continent
  <fct>         <fct>
1 Angola        Angola
2 Burkina Faso  Burkina Faso
3 Guinea-Bissau Guinea-Bissau
4 Netherlands   <NA>
5 Angola        Angola
```

Helper functions for `mutate`: `case_when`

```
set.seed(8214444)
diamonds %>% slice_sample(n = 5) %>%
  select(c(cut, color, clarity, price)) %>%
  mutate(good_bargain =
           case_when(cut == "Ideal" &
                       price < mean(price) ~ "Slickdeal!",
                     color == "I" ~ "Maybe",
                     clarity == "VVS2" |
                       price < mean(price) ~ "Uhmm...",
                     .default = "No way!"
                     ))
```

```
# A tibble: 5 × 5
  cut   color clarity price good_bargain
  <ord> <ord> <ord>   <int> <chr>
1 Ideal H     VS2      1402 Slickdeal!
2 Ideal G     VS1     18178 No way!
3 Fair  H     SI1      3972 Uhmm...
4 Ideal F     SI1     12415 No way!
5 Fair  I     VVS2     3288 Maybe
```

Helper functions for mutate: rolling computations

- `lag` and `lead` give you the n-th value before or after

```
gapminder %>% filter(country == "Kuwait") %>%
  select(c(country,year, gdpPercap)) %>%
  mutate(year_bef = lag(gdpPercap, n = 1)) %>%
  filter(!is.na(year_bef)) # NA in 1st row
```

```
# A tibble: 11 × 4
   country  year gdpPercap year_bef
   <fct>   <int>     <dbl>    <dbl>
 1 Kuwait   1957   113523.  108382.
 2 Kuwait   1962    95458.  113523.
 3 Kuwait   1967    80895.   95458.
 4 Kuwait   1972   109348.   80895.
 5 Kuwait   1977    59265.  109348.
 6 Kuwait   1982    31354.   59265.
 7 Kuwait   1987    28118.   31354.
 8 Kuwait   1992    34933.   28118.
 9 Kuwait   1997    40301.   34933.
10 Kuwait   2002    35110.   40301.
11 Kuwait   2007    47307.   35110.
```

Helper functions for mutate: rolling computations

```
gapminder %>% filter(country == "Kuwait") %>%
  select(c(country,year, gdpPercap)) %>%
  mutate(year_bef = gdpPercap - lag(gdpPercap, n = 1)) %>%
  filter(!is.na(year_bef)) # NA in 1st row
```

```
# A tibble: 11 × 4
   country  year gdpPercap year_bef
   <fct>   <int>     <dbl>    <dbl>
 1 Kuwait   1957   113523.    5141.
 2 Kuwait   1962    95458.  -18065.
 3 Kuwait   1967    80895.  -14563.
 4 Kuwait   1972   109348.   28453.
 5 Kuwait   1977    59265.  -50082.
 6 Kuwait   1982    31354.  -27911.
 7 Kuwait   1987    28118.   -3236.
 8 Kuwait   1992    34933.    6814.
 9 Kuwait   1997    40301.    5368.
10 Kuwait   2002    35110.   -5191.
11 Kuwait   2007    47307.   12197.
```

Error ×