Exploring a data frame

Table of contents

1 readr, dplyr, and ggplot2

```
library(dplyr)
```

```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(readr)
library(ggplot2)
```

These are the three libraries you need most when you explore a tabular dataset.

2 Read the Gapminder labor cost data set

```
myfilepath <-
 ↪  "datasets_ATRIUM/gapminder_hourly_labour_cost_constant_2017_usd--by--geo--time.csv"
laborcost_df <- read_csv(file = myfilepath,
                         show_col_types = TRUE)
```

```
Rows: 548 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): geo
dbl (2): time, hourly_labour_cost_constant_2017_usd

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

It is the file demonstrated in the previous session. We saved it into the folder `datasets_ATRIUM`. You can also get it at https://raw.githubusercontent.com/open-numbers/ddf--gapminder--systema_globalis/refs/heads/master/countries-etc-datapoints/ddf--datapoints--hourly_labour_cost_constant_2017_usd--by--geo--time.csv.

Watch the message `read_csv` gives you about the file. You can suppress it by overriding the default to `show_col_types = FALSE`.

3 `dplyr::glimpse`

- peek at the dataset (tilted 90°)

```
glimpse(laborcost_df)
```

```
Rows: 548
Columns: 3
$ geo                                   <chr> "arg", "arg", "arm", "arm", "arm"~
$ time                                  <dbl> 2011, 2012, 2011, 2012, 2013, 201~
$ hourly_labour_cost_constant_2017_usd <dbl> 0.92, 1.04, 4.23, 4.59, 6.12, 6.0~
```

Gives you the number of rows and columns, the column names with their data class, and it also shows as many elements (values) in each column as to fit your screen.

4 summary

```
summary(laborcost_df)
```

```
     geo                  time      hourly_labour_cost_constant_2017_usd
 Length:548         Min.   :1994   Min.   : 0.000                      
 Class :character   1st Qu.:2005   1st Qu.: 9.867                      
 Mode  :character   Median :2011   Median :18.320                      
                    Mean   :2010   Mean   :19.686                      
                    3rd Qu.:2017   3rd Qu.:26.915                      
                    Max.   :2020   Max.   :48.720                      
```

Gives you the "five-number summary" of each numeric column (it's often called this way, although the numbers are obviously six…). With categorical columns, it depends, whether the column is a character vector or a factor.

5 summary with categorical columns as factors

```
      geo            time      hourly_labour_cost_constant_2017_usd
 cze    : 22   Min.   :1994   Min.   : 0.000                      
 svn    : 22   1st Qu.:2005   1st Qu.: 9.867                      
 cyp    : 21   Median :2011   Median :18.320                      
 deu    : 21   Mean   :2010   Mean   :19.686                      
 pol    : 21   3rd Qu.:2017   3rd Qu.:26.915                      
 svk    : 21   Max.   :2020   Max.   :48.720                      
 (Other):420                                                       
```

If you have a data frame with categorical variables converted to factors, the summary will show you a glimpse of their **levels** (unique values) and their frequencies, as well as tell you how many levels there are.

So far, do not worry about factors. The dplyr as well as the ggplot2 libraries do this factor conversion on the fly whenever they need it.

6 Rename a column with base R

hourly_labour_cost_constant_2017_usd too long, shorten to labor_cost.

```
colnames(laborcost_df)[colnames(laborcost_df) ==
                          "hourly_labour_cost_constant_2017_usd"] <-
 ↪  "labor_cost"
```

```
colnames(laborcost_df)
```

```
[1] "geo"        "time"       "labor_cost"
```

7 Rename a column with dplyr

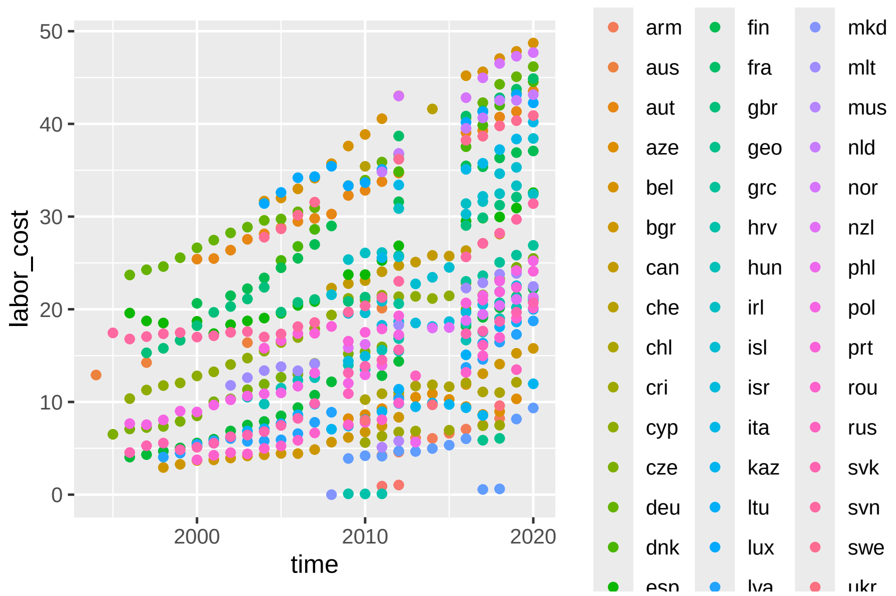
```
laborcost_df <- rename(.data = laborcost_df,
                         labor_cost =
 ↪  hourly_labour_cost_constant_2017_usd
                         )
colnames(laborcost_df)
```

```
[1] "geo"        "time"       "labor_cost"
```

You already know you could have named all columns your way when reading in the file. Here are two ways to rename a column: one base-R-like and the other one provided by dplyr.

8 Plot the data set

```
ggplot(data = laborcost_df,
       mapping = aes(x = time,
                     y = labor_cost,
                     color = geo)) + geom_point()
```



labor_cost
0
10
20
30
40
50
2000
2010
2020
time
arm
aus
aut
aze
bel
bgr
can
che
chl
cri
cyp
cze
deu
dnk
esp
fin
fra
gbr
geo
grc
hrv
hun
irl
isl
isr
ita
kaz
ltu
lux
lva
mkd
mlt
mus
nld
nor
nzl
phl
pol
prt
rou
rus
svk
svn
swe
ukr