Getting and cleaning data

ADS2 week 7

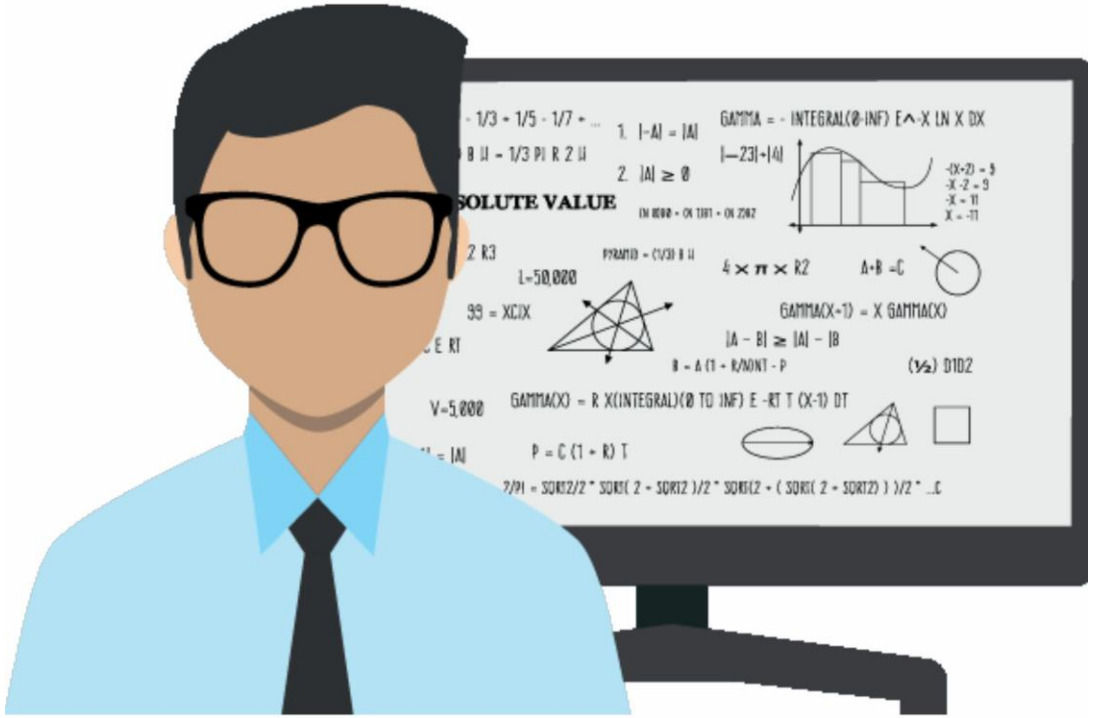
Dmytro Shytikov (adapted from Xianghua Li’s slides)

2023-10-30

dmytroshytikov@intl.zju.edu.cn

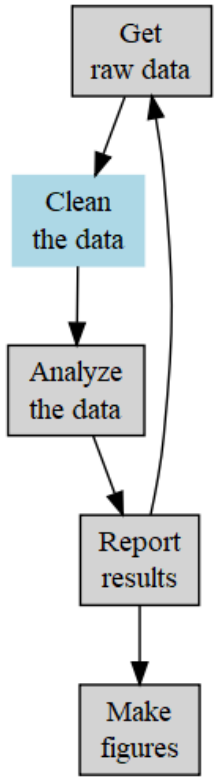
Why do we care about cleaning data?

What you imagine a data scientists do:



What data scientists really do:

Data analysis workflow



Before you analyze your data, answer these questions:

- Is your data ready to be analyzed?
- Is your data ready to be modelled?
- Is your data ready to be plotted?

What does it mean – raw data?

Raw data are “raw” if:

- No software processed it;
- No values are modified;
- No data are removed;
- No summary available.

Examples:

- RNA- and RNA-sequencing data;
- Data from flow cytometry;
- Data from image analysis;
- HTML data after data mining;
- Data from observations;
- etc.

Why do we need to clean raw data?

I want to make use of my flow cytometry data. But there are a few issues:

1. The data are not summarized;
2. From the current output, you get no valuable information;
3. You need to process it to get anything valuable.

	FSC.H	SSC.H	FITC.H	APC.H	Other data
1	370669	40987	309	102564	...
2	4007789	289260	3751	29742	...
3	1827661	613884	5290	3685276	...
4	7669807	727412	9280	52017	...
5	1228843	229826	3162	705259	...
6	1315640	289438	6748	1313375	...
7	1799229	99692	1558	2105	...
8	12735706	1892834	17371	339346	...
9	4009833	400802	2410	17196	...
10	...	...	...	...	...

Can these data be analyzed?

```
head(weather, 9) %>% select(2:14)
```

	year	month	measure	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	Other data
1	2014	12	Max.TemperatureF	64	42	51	43	42	45	38	29	49	48	...
2	2014	12	Mean.TemperatureF	52	38	44	37	34	42	30	24	39	43	...
3	2014	12	Min.TemperatureF	39	33	37	30	26	38	21	18	29	38	...
4	2014	12	Max.Dew.PointF	46	40	49	24	37	45	36	28	49	45	...
5	2014	12	MeanDew.PointF	40	27	42	21	25	40	20	16	41	39	...
6	2014	12	Min.DewpointF	26	17	24	13	12	36	-3	3	28	37	...
7	2014	12	Max.Humidity	74	92	100	69	85	100	92	92	100	100	...
8	2014	12	Mean.Humidity	63	72	79	54	66	93	61	70	93	95	...
9	2014	12	Min.Humidity	52	51	57	39	47	85	29	47	86	89	...

“Dirty” data is a problem

The data may not be analyzed easily due to:

- Special characters where not needed;
- Use of the wrong data structures;
- Duplicated rows;
- Misspelling;
- White spaces;
- Missing data;
- Zeroes instead of `NULL` or `NA` values;
- Other inaccuracies;
- Poor structure.

```
weather %>% head(9) %>% select(2:8)
```

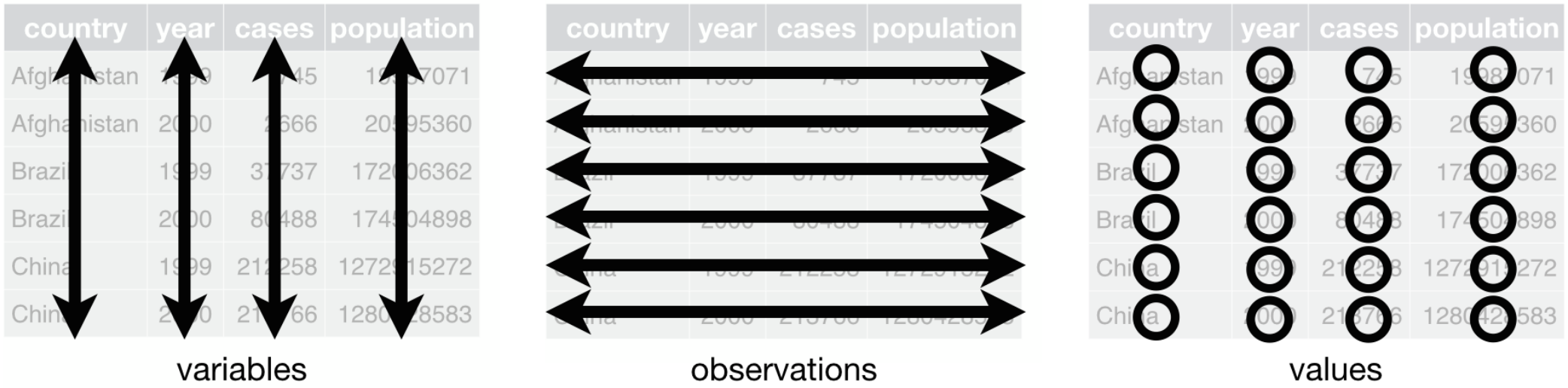
	year	month	measure	X1	X2	X3	X4
1	2014	12	Max.TemperatureF	64	42	51	43
2	2014	12	Mean.TemperatureF	52	38	44	37
3	2014	12	Min.TemperatureF	39	33	37	30
4	2014	12	Max.Dew.PointF	46	40	49	24
5	2014	12	MeanDew.PointF	40	27	42	21
6	2014	12	Min.DewpointF	26	17	24	13
7	2014	12	Max.Humidity	74	92	100	69
8	2014	12	Mean.Humidity	63	72	79	54
9	2014	12	Min.Humidity	52	51	57	39

Learning objectives

1. Describe features of **tidy** data and explain advantages of tidy datasets
2. Explain the process of cleaning data
3. Describe ways of handling different data types and data structures in R
4. Introduce the key data cleaning tools: `tidyverse` package and its elements

TIDY DATA, ITS FEATURES AND ADVANTAGES

Tidy data: main features



Tidy data: advantages

```
head(weather, 9) %>% select(2:15)
```

	year	month	measure	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	Other data
1	2014	12	Max.TemperatureF	64	42	51	43	42	45	38	29	49	48	39	...
2	2014	12	Mean.TemperatureF	52	38	44	37	34	42	30	24	39	43	36	...
3	2014	12	Min.TemperatureF	39	33	37	30	26	38	21	18	29	38	32	...
4	2014	12	Max.Dew.PointF	46	40	49	24	37	45	36	28	49	45	37	...
5	2014	12	MeanDew.PointF	40	27	42	21	25	40	20	16	41	39	31	...
6	2014	12	Min.DewpointF	26	17	24	13	12	36	-3	3	28	37	27	...
7	2014	12	Max.Humidity	74	92	100	69	85	100	92	92	100	100	92	...
8	2014	12	Mean.Humidity	63	72	79	54	66	93	61	70	93	95	87	...
9	2014	12	Min.Humidity	52	51	57	39	47	85	29	47	86	89	82	...

Wide data format

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

table1

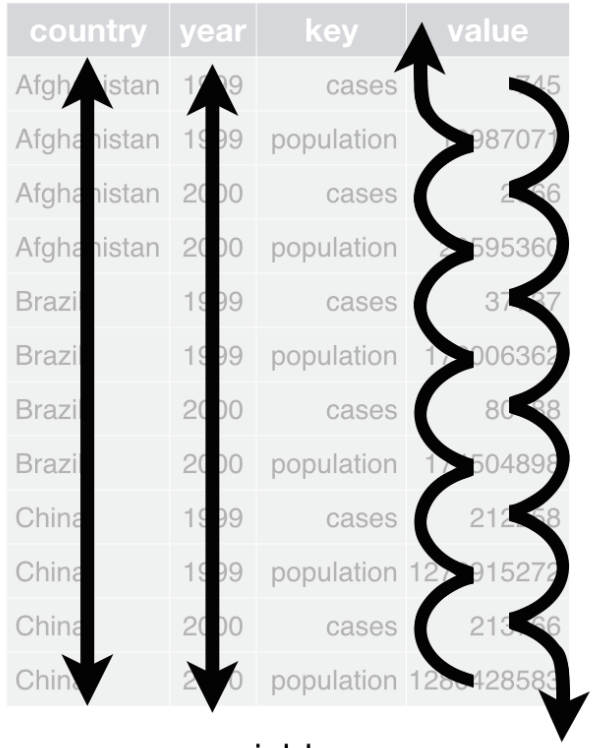
variables

observations

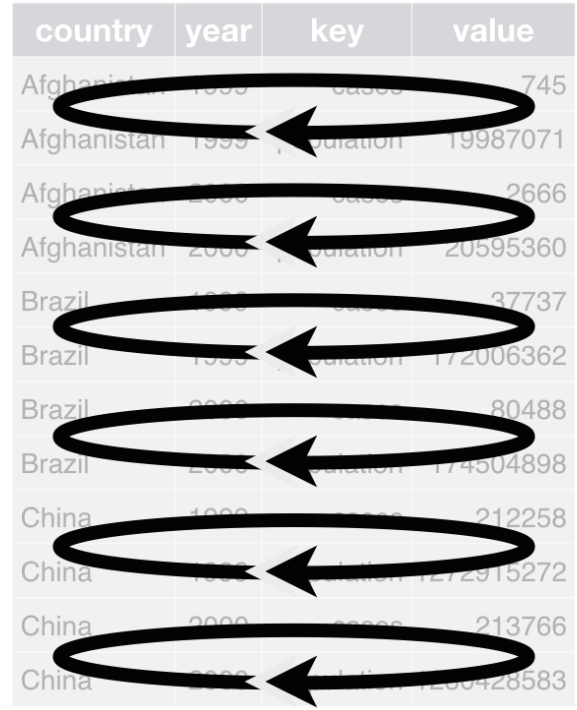
Long data format

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2



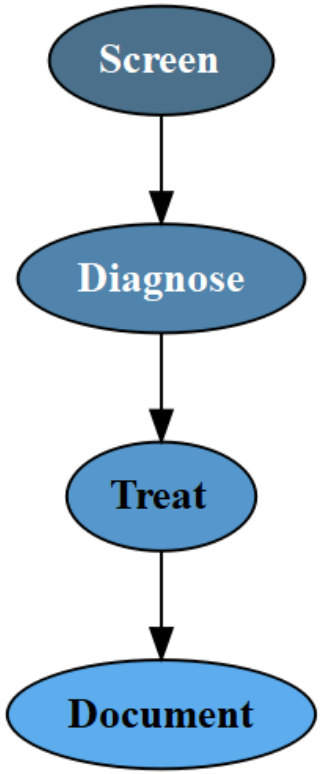
variables



observations

DATA CLEANING

The key steps of data cleaning



The key steps of the data cleaning:

- **Screen**: Check the data set systematically
- **Diagnose**: Find out the nature of the problem.
- **Treat**: Delete, edit, or leave the data as it is.
- **Document**: Comment on each step to make sure you will not forget the reason of the action and the original state.

HANDLING DATA IN R

Data types and structures in R

The main classes of data:

- Character
- Numeric
- Integer
- Logical
- Complex

Data structures:

- Vectors
- Factors
- Lists
- Matrices and arrays
- Data frames
- …

Check properties of the object

There is a set of commands that allow you to check the properties of the studied object

```
class(), mode(), typeof()
is.<DATA_CLASS/TYPE>(), as.<DATA_CLASS/TYPE>()

str(), summary()

attributes(), names()
dimnames(), colnames(), rownames()
dim(), length()

...
```

Subsetting and rearranging data in R

Subset or rearrange 1D data structures

```
vec.num <- c(1, 2.8, 3, 4.4)
vec.num[2]
[1] 2.8
vec.num[c(1, 4)]
[1] 1.0 4.4
vec.num_2 <- vec.num[c(2, 4, 3, 1)]

someList <- list(first_vec = vec.num,
second_vec = vec.num_2)

someList[[2]]
[1] 2.8 4.4 3.0 1.0
```

Subset or rearrange 2D data structures

```
matrix.num <- matrix(data = c(vec.num,
sqrt(vec.num_2)), nrow = 2, byrow = T)

matrix.num
         [,1]     [,2]     [,3] [,4]
[1,] 1.00000 2.800000 3.000000  4.4
[2,] 1.67332 2.097618 1.732051  1.0

dim(matrix.num)
[1] 2 4

matrix.num[1, c(4, 1)]
[1] 4.4 1.0
```

Working with data frames in R

```
head(diamonds, 10) %>% as.data.frame()
```

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48
7	0.24	Very Good	I	VVS1	62.3	57	336	3.95	3.98	2.47
8	0.26	Very Good	H	SI1	61.9	55	337	4.07	4.11	2.53
9	0.22	Fair	E	VS2	65.1	61	337	3.87	3.78	2.49
10	0.23	Very Good	H	VS1	59.4	61	338	4.00	4.05	2.39

Working with data frames in R

```
head(diamonds[diamonds$cut == "Ideal", ], 3)
```

```
   carat   cut color clarity depth table price    x    y    z
1   0.23 Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
12  0.23 Ideal     J     VS1  62.8    56   340 3.93 3.90 2.46
14  0.31 Ideal     J     SI2  62.2    54   344 4.35 4.37 2.71
```

```
diamonds[diamonds$cut == "Ideal" & diamonds$clarity == "VS1", ][1:3,]
```

```
   carat   cut color clarity depth table price    x    y    z
12  0.23 Ideal     J     VS1  62.8    56   340 3.93 3.90 2.46
52  0.23 Ideal     G     VS1  61.9    54   404 3.93 3.95 2.44
61  0.35 Ideal     I     VS1  60.9    57   552 4.54 4.59 2.78
```

CLEANING DATA IN R: `tidyverse` and related packages

`tidyverse` and friends

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✓ dplyr     1.1.2     ✓ readr     2.1.4
✓ forcats   1.0.0     ✓ stringr   1.5.0
✓ ggplot2   3.4.3     ✓ tibble    3.2.1
✓ lubridate 1.9.2     ✓ tidyr     1.3.0
✓ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

Cleaning data in R: `dplyr`

Subsetting tables in R

```
tibble(), as_tibble()

select(dataframe, variables)
relocate(dataframe, variables)

rename(dataframe, variables)

filter(dataframe, variable == value)

slice(dataframe, rows)
```

Modifying tables in R

```
mutate(dataframe, new_variable =
your_code)
transmute()

arrange(dataframe, variable_to_arrange)
group_by(dataframe, grouping_variable)

summarise(dataframe, summary_stat =
your_code)

Pipe operator (%>%)
```

Cleaning data in R: chaining operations with the %>% operator

ADS2, week 1, R refresher

1. Create a vector consisting of 5 numbers: 5, 6, 4, 5, 10.

2. Add "1" to all elements of the vector.

3. Extract the first element of the vector and add 3 to it.

4. Add the resultant number as the sixth element of the vector.

5. Take the square root from the resultant vector and assign it as a new vector.

6. Convert it to a matrix with 3 columns

Regular way of solving the task

```
vec <- c(5, 6, 4, 5, 10)
vec <- vec + 1
vec[6] <- vec[1] + 3


vec
```

```
[1]  6  7  5  6 11  9
```

```
vec_new <- sqrt(vec)
vec_new <- matrix(data = vec_new, ncol
= 3, byrow = T)


vec_new
```

```
         [,1]     [,2]     [,3]
[1,] 2.44949 2.645751 2.236068
[2,] 2.44949 3.316625 3.000000
```

Cleaning data in R: chaining operations with the %>% operator

```
head(case1_col_desc, 4)
```

```
     Var_name Difference pooledSD Description Measure_unit p_value
1 splenocytes       5.63     5.05 splenocytes  10^6^ cells   0.117
2  CD8T_cells       0.59     0.97 CD8 T-cells            %   0.381
3  CD4T_cells       0.10     3.06 CD4 T-cells            %   0.958
4     B_cells       0.78     1.51     B-cells            %   0.440
```

```
dim(case1_col_desc)
```

```
[1] 16  6
```

Cleaning data in R: chaining operations with the %>% operator

```
case1_differ <- case1_col_desc %>%
  filter(p_value < 0.05) %>%
  mutate(Cohen_d = (Difference/pooledSD) %>%
           round(2)) %>%
  select(4,2,7) %>%
  arrange(desc(Cohen_d)) %>%
  rename(`Absolute difference, %` = Difference,
         `Variable` = Description)

head(case1_differ, 3)
```

	Variable	Absolute difference, %	Cohen_d
1	memory phenotype CD8 T-cells	10.21	9.20
2	IgM^+^ B-cells	6.16	3.08
3	follicular CD4 T-cells	1.58	3.04

Reshaping data in R: `tidyr`

```
tuberculosisAlgria <- who2

head(tuberculosisAlgria, 9) %>%
select(1:4)
```

```
  country year sp_m_014 sp_m_1524 Others
1 Algeria 2000       59        927    ...
2 Algeria 2001       41       1345    ...
3 Algeria 2002       39       1364    ...
4 Algeria 2003       40       1316    ...
5 Algeria 2004       63       1326    ...
6 Algeria 2005       53       1309    ...
7 Algeria 2006       41       1173    ...
8 Algeria 2007       95       1388    ...
9 Algeria 2008       99       1505    ...
```

```
gather(data = tuberculosisAlgria, key =
"Group", value = "Cases",
3:ncol(tuberculosisAlgria)) %>% head(9)
```

```
  country year    Group Cases
1 Algeria 2000 sp_m_014    59
2 Algeria 2001 sp_m_014    41
3 Algeria 2002 sp_m_014    39
4 Algeria 2003 sp_m_014    40
5 Algeria 2004 sp_m_014    63
6 Algeria 2005 sp_m_014    53
7 Algeria 2006 sp_m_014    41
8 Algeria 2007 sp_m_014    95
9 Algeria 2008 sp_m_014    99
```

Reshaping data in R: `tidyr`

Separate one column into several

```
separate(data = tub_AlgeriaGather, col = "Group", into = c("Form", "Gender", "Age"), sep = "_") %>% head(9)
```

```
  country year Form Gender Age Cases
1 Algeria 2000   sp      m 014    59
2 Algeria 2001   sp      m 014    41
3 Algeria 2002   sp      m 014    39
4 Algeria 2003   sp      m 014    40
5 Algeria 2004   sp      m 014    63
6 Algeria 2005   sp      m 014    53
7 Algeria 2006   sp      m 014    41
8 Algeria 2007   sp      m 014    95
9 Algeria 2008   sp      m 014    99
```

Spread the dataset

```
tub_AlgeriaSample <- tub_AlgeriaGather %>%

  separate(col = "Group", into = c("Form", "Gender", "Age"), sep = "_") %>%

  filter(Form == "sp", Age == "2534") %>%

  select(1,2,4,6)


spread(tub_AlgeriaSample, key = "year", value = "Cases") %>% select(1:5)
```

```
  country Gender 2000 2001 2002 Other data
1 Algeria      f 1293  782  730          ...
2 Algeria      m 1516 1614 1580          ...
```

Conclusions

By now, you should

1. Know the criteria of the **tidy** data.
2. Know how to clean data.
3. Be aware of different ways how to handle data in R.
4. Got introduced to the `tidyverse` package.

Further reading

- `library(swirl)  # An R package for self-learning Exploratory data analysis course`

- Wickham H. Tidy data. J Stat Softw [Internet]. 2014;59(10). Available from: http://dx.doi.org/10.18637/jss.v059.i10

- Wickham H. Ggplot2: Elegant graphics for data analysis. 2nd ed. Cham, Switzerland: Springer International Publishing; 2016.

- `library(data.table)  # A useful R package for data cleaning`

THANK YOU FOR ATTENTION