Data Handling: Import, Cleaning and Visualisation

Lecture 7:
Data Preparation

Dr. Aurélien Sallin

2024-11-14

Warm-up

JSON files: open-ended question

Be the JSON file

```
{
  "students": [
    {
      "id": 19091,
      "firstName": "Peter",
      "lastName": "Mueller",
      "grades": {
          "micro": 5,
          "macro": 4.5,
          "data handling": 5.5
          }
    },
    {
      "id": 19092,
      "firstName": "Anna",
      "lastName": "Schmid",
      "grades": {
          "micro": 5.25,
          "macro": 4,
          "data handling": 5.75
          }
    },
    {
```

XML

```xml
<students>
  <student>
    <id>19091</id>
    <firstName>Peter</firstName>
    <lastName>Mueller</lastName>
    <grades>
      <micro>5</micro>
      <macro>4.5</macro>
      <dataHandling>5.5</dataHandling>
    </grades>
  </student>
  <student>
    <id>19092</id>
    <firstName>Anna</firstName>
    <lastName>Schmid</lastName>
    <grades>
      <micro>5.25</micro>
      <macro>4</macro>
      <dataHandling>5.75</dataHandling>
    </grades>
  </student>
  <student>
    <id>19093</id>
```

- ‘students’ is the root-node, ‘grades’ are its children
- The siblings of Trevor Noah are Anna Schmid and Peter Mueller
- The code below would be an alternative, equivalent notation for the third student in the xml file above.

```
<student id="19093" firstName="Noah" lastName="Trevor">
      <grades micro="4" macro="4.5" dataHandling="5" />
</student>
```

Part II: Data preparation, analysis, and visualization

The dataset is imported, now what?

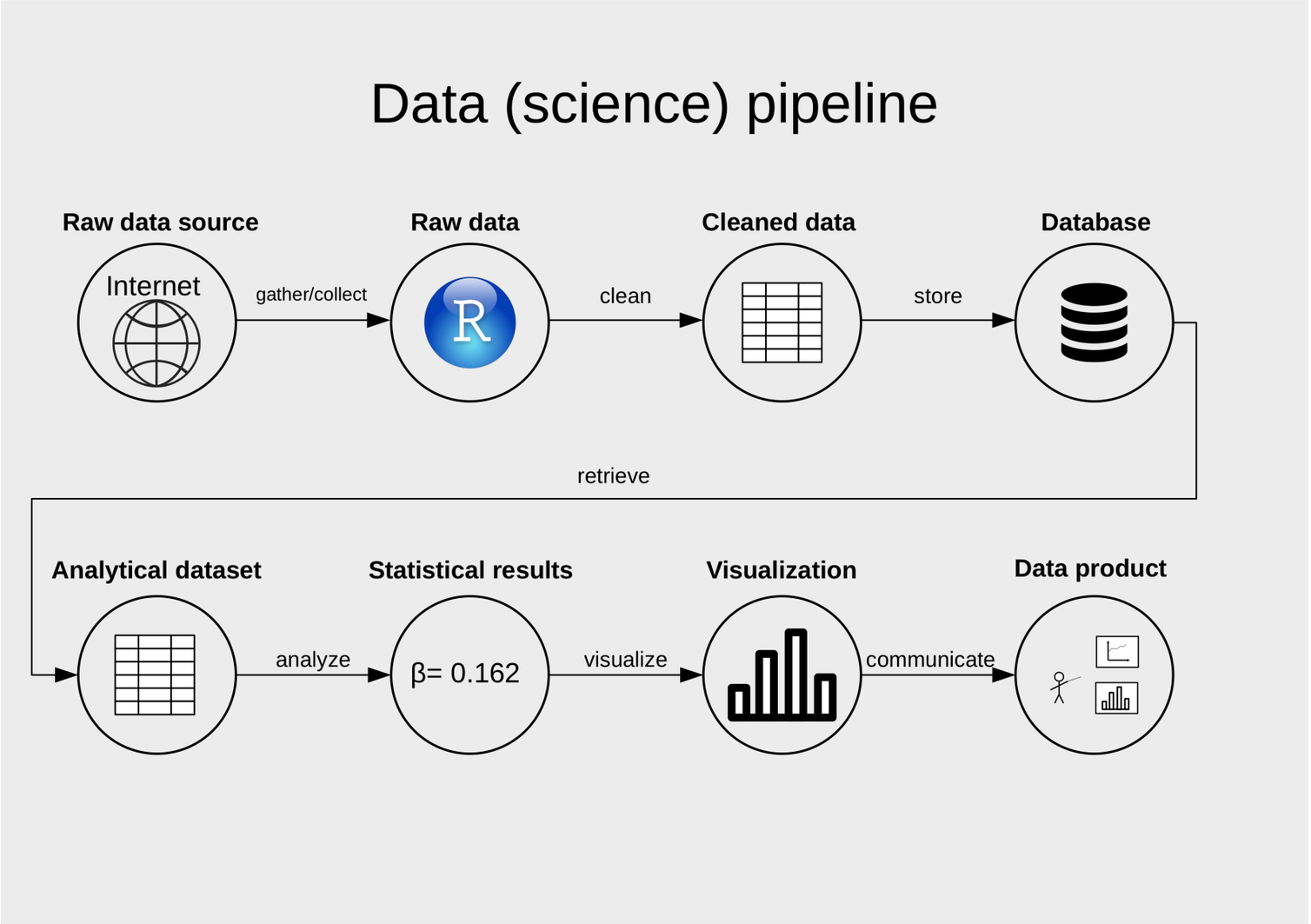
- In practice: still a long way to go.
- Parsable, but messy data: inconsistencies, data types, missing observations, wide format.

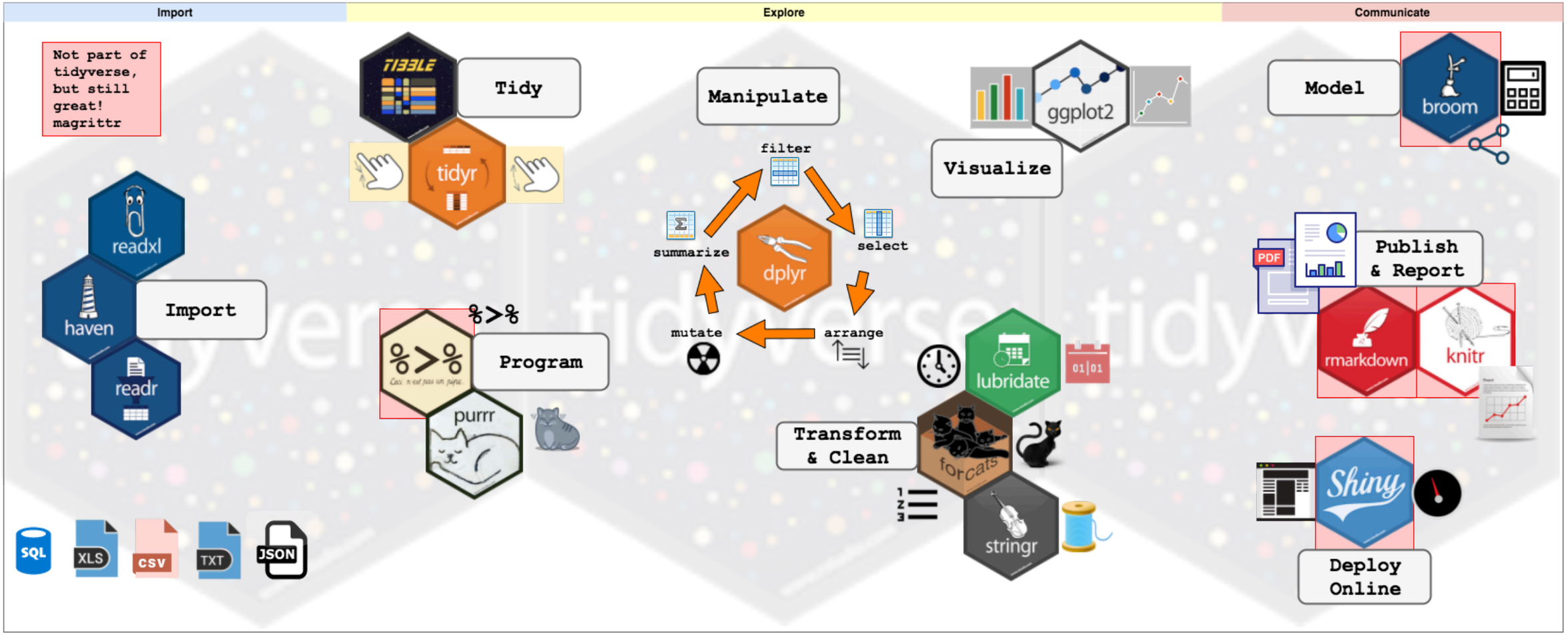
The dataset is imported, now what?

- In practice: still a long way to go.
- Parsable, but messy data: inconsistencies, data types, missing observations, wide format.

- **Goal** of data preparation: dataset is ready for analysis.

Part II: Data gathering and preparation





Source: https://www.storybench.org/wp-content/uploads/2017/05/tidyverse.png

Part II: Data preparation, analysis, and visualization

Date	Topic
14.11.2024	Data preparation and manipulation
21.11.2024	Basic statistics and data analysis with R
21.11.2024	Exercises/Workshop 4: Data gathering, data import
28.11.2024	Visualisation

Part II: Data preparation, analysis, and visualization

Date	Topic
05.12.2024	Guest Lecture: Data Handling @Deloitte (Rachel Lund, Senior Economist)
05.12.2024	Exercises/Workshop 5: Data preparation and applied data analysis with R
12.12.2024	Analytics, more visualisation, and data products
19.12.2024	Summary, Wrap-up, Final workshop
19.12.2024	Exercises/Workshop 6: Visualization, dynamic documents
19.12.2024	Exam for Exchange Students

Part II: Data gathering and preparation

Data preparation: Beware of the “Garbage in garbage out” problem

Key conditions:

1. Data values are consistent/clean within each variable.
2. Variables are of proper data types.
3. Dataset is in ‘tidy’ (long) format.

Goals for today and next time: master data preparation

Data preparation can be understood as consisting of five main actions:

- **Tidy** data.
- **Reshape** datasets from wide to long (and vice versa).
- **Bind** or stack rows in datasets.
- **Join** datasets (covered next time).
- **Clean** data.

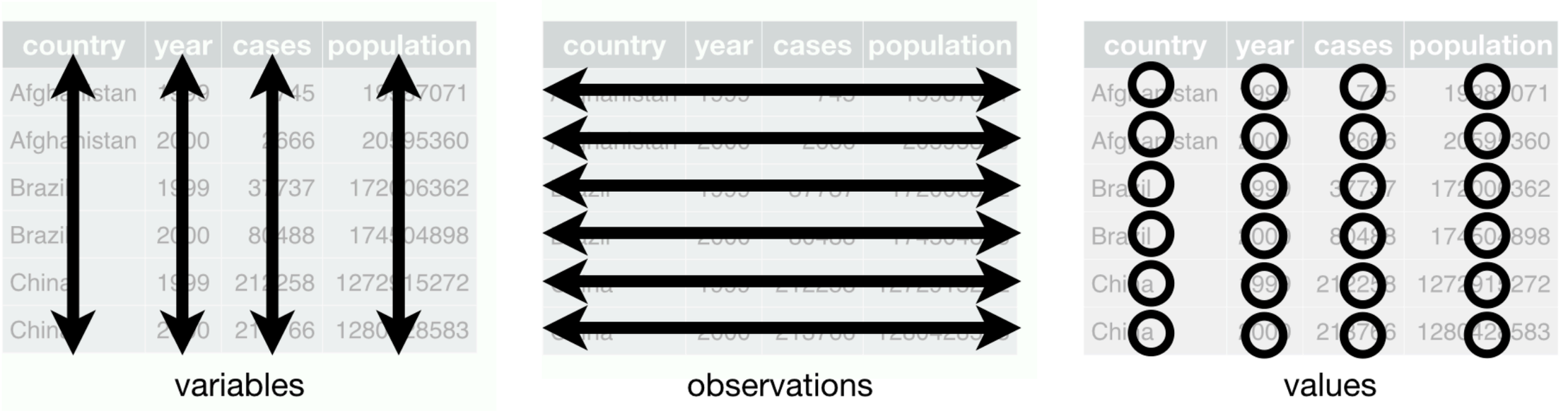
Tidy data

Tidy data: some vocabulary

Following R4DS, a tidy dataset is tidy when...

1. Each variable is a column; each column is a variable.
2. Each observation is a row; each row is an observation.
3. Each value is a cell; each cell is a single value.

Tidy data



Tidy data. Source R4DS.

Tidy data is not trivial

In Economics, the definition of an *observation* can vary:

- Panel data (longitudinal data)
- Cross-sectional data
- Time series

Tidy data is not trivial

Panel Data (Longitudinal Data):

Panel data tracks the **same units over time**: each unit has multiple observations across time periods.
Observation: a measurement for a specific unit at a particular point in time.

Cross-Sectional Data:

“Snapshot” of **different units at the same moment**.
Observation: single measurement for each unit at a single point in time.

Time Series Data:

Single unit tracked over time.
Observation: measurement of a single variable for a single unit (or aggregate) over multiple points

Three examples of non-tidy data (1)

Messy 💩

```
# A tibble: 2 × 4
  measure     `Jan 1` `Jan 2` `Jan 3`
  <chr>         <dbl>   <dbl>   <dbl>
1 Temperature      20      22      21
2 Humidity         80      78      82
```

Tidy 😎

...

Three examples of non-tidy data (1)

Messy 💩

```
# A tibble: 2 × 4
  measure     `Jan 1` `Jan 2` `Jan 3`
  <chr>         <dbl>   <dbl>   <dbl>
1 Temperature      20      22      21
2 Humidity         80      78      82
```

Tidy 😎

```
# A tibble: 3 × 3
  Date  Temperature Humidity
  <chr>       <dbl>    <dbl>
1 Jan 1          20       80
2 Jan 2          22       78
3 Jan 3          21       82
```

Three examples of non-tidy data (2)

Messy 💩

```
# A tibble: 3 × 2
   year temperature_location
  <dbl> <chr>
1  2019 22C_London
2  2019 18C_Paris
3  2019 25C_Rome
```

Tidy 😎

```
homework..
```

Three examples of non-tidy data (3)

Messy 💩

```
   Student Econ DataHandling Management
1 Johannes 5.00          4.0        5.5
2   Hannah 5.25          4.5        6.0
3     Igor 4.00          5.0        6.0
```

Tidy 😎

```
homework..
```

Reshaping

Reshaping: the concept

"Long" format

country	year	metric
x	1960	10
x	1970	13
x	2010	15
y	1960	20
y	1970	23
y	2010	25
z	1960	30
z	1970	33
z	2010	35

"Wide" format

country	yr1960	yr1970	yr2010
x	10	13	15
y	20	23	25
z	30	33	35

Long and wide data. Source: Hugo Tavares

Reshaping: implementation in R

- From wide to long: `melt()`, `gather()`,

👉 We'll use `tidyverse::pivot_longer()`.

- From long to wide: `cast()`, `spread()`,

👉 We'll use `tidyverse::pivot_wider()`.

Reshaping: implementation in R with `tidyverse()`

country	year	metric
x	1960	10
x	1970	13
x	2010	15
y	1960	20
y	1970	23
y	2010	25
z	1960	30
z	1970	33
z	2010	35

```
pivot_wider(names_from = “year”,
            names_prefix = “yr”,
            values_from = “metric”)
```

country	yr1960	yr1970	yr2010
x	10	13	15
y	20	23	25
z	30	33	35

```
pivot_longer(cols = yr1960:yr2010,
             names_to = “year”,
             names_prefix = “yr”
             values_to = “metric”)
```

Stack and row-bind

Stack/row-bind: the concept

ID	X	Y
1	a	50
2	b	10

ID	Z
3	M
4	O

ID	X	Z
5	c	P

ID	X	Y	Z
1	a	50	NA
2	b	10	NA
3	NA	NA	M
4	NA	NA	O
5	c	NA	P

Stack/row-bind: implementation in R

- Use `rbind()` in base `R`
  - Requires that the data frames have the same column names and same column classes.

- Use `bind_rows()` from `dplyr()`
  - More flexible
  - Binds data frames with different column names and classes
  - Automatically fills missing columns with `NA`

For these reasons (+ performance, handling or row names, and handling of factors), `dplyr::bind_rows()` *is preferred in most applications.*

Stack/row-bind: code example

```
# Create three dfs
subset1 <- data.frame(ID = c(1,2),
                      X = c("a", "b"),
                      Y = c(50,10))

subset2 <- data.frame(ID = c(3,4),
                      Z = c("M", "O"))

subset3 <- data.frame(ID = c(5),
                      X = c("c"),
                      Z = "P")

# Inspect
subset1
```

```
  ID X  Y
1  1 a 50
2  2 b 10
```

```
subset2
```

```
  ID Z
1  3 M
2  4 O
```

```
subset3
```

```
  ID X Z
1  5 c P
```

Stack/row-bind: code example

```
# Stack data frames
combined_df_bind_rows <- bind_rows(subset1, subset2, subset3)
combined_df_rbind     <- rbind(subset1, subset2, subset3)
```

```
Error in rbind(deparse.level, ...): numbers of columns of arguments do not match
```

```
# What are the following objects?
combined_df_bind_rows
```

```
  ID    X  Y    Z
1  1    a 50 <NA>
2  2    b 10 <NA>
3  3 <NA> NA    M
4  4 <NA> NA    O
5  5    c NA    P
```

```
combined_df_rbind
```

```
Error in eval(expr, envir, enclos): object 'combined_df_rbind' not found
```

The errors are due to the requirements of `rbind()`.