Data Handling: Import, Cleaning and Visualisation

Lecture 8:

Basic Data Analysis with R

Dr. Aurélien Sallin

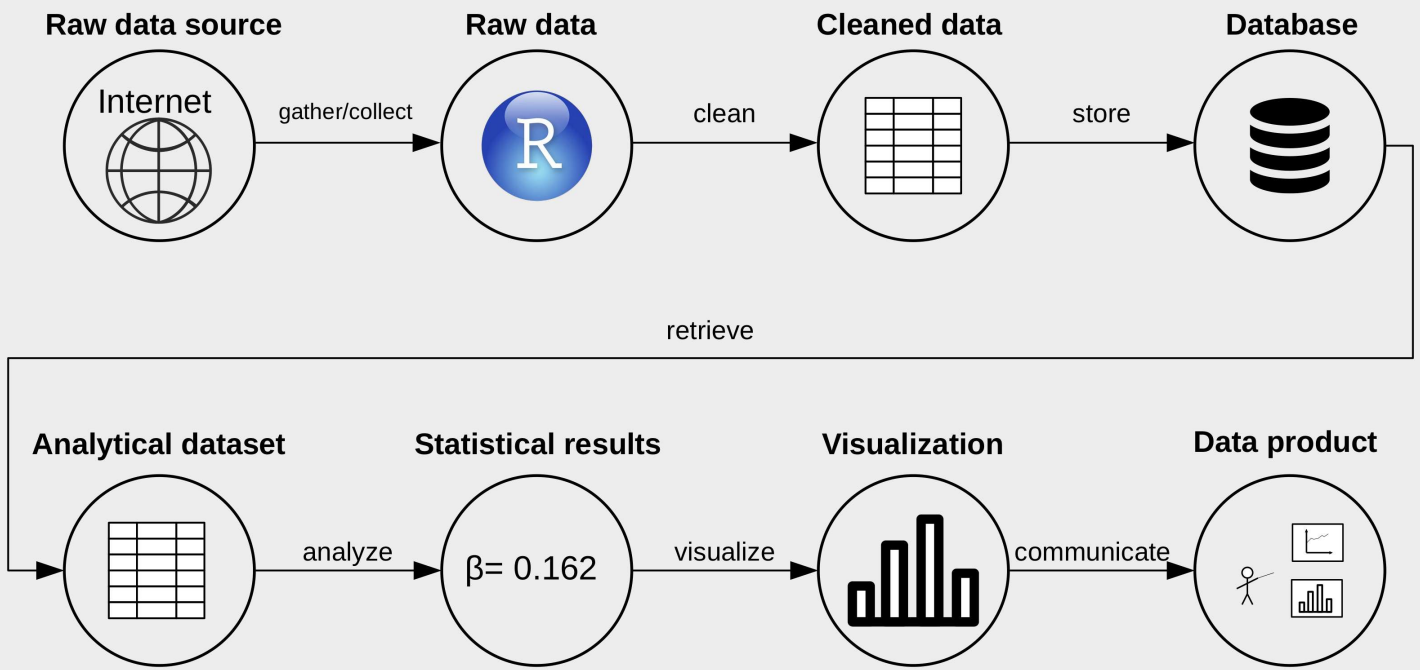
Updates

Reminder

- Last lecture: content is still open. Any wishes?
- Send questions for the last lecture to me per email: aurelien.sallin@unisg.ch

Recap: Data Preparation

Data (science) pipeline



Data preparation/data cleaning

- **Goal** of data preparation: Dataset is ready for analysis.
- **Key conditions**:
  1. Data values are consistent/clean within each variable.
  2. Variables are of proper data types.
  3. Dataset is in ‘tidy’ (in long format)!

Reshaping

"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

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

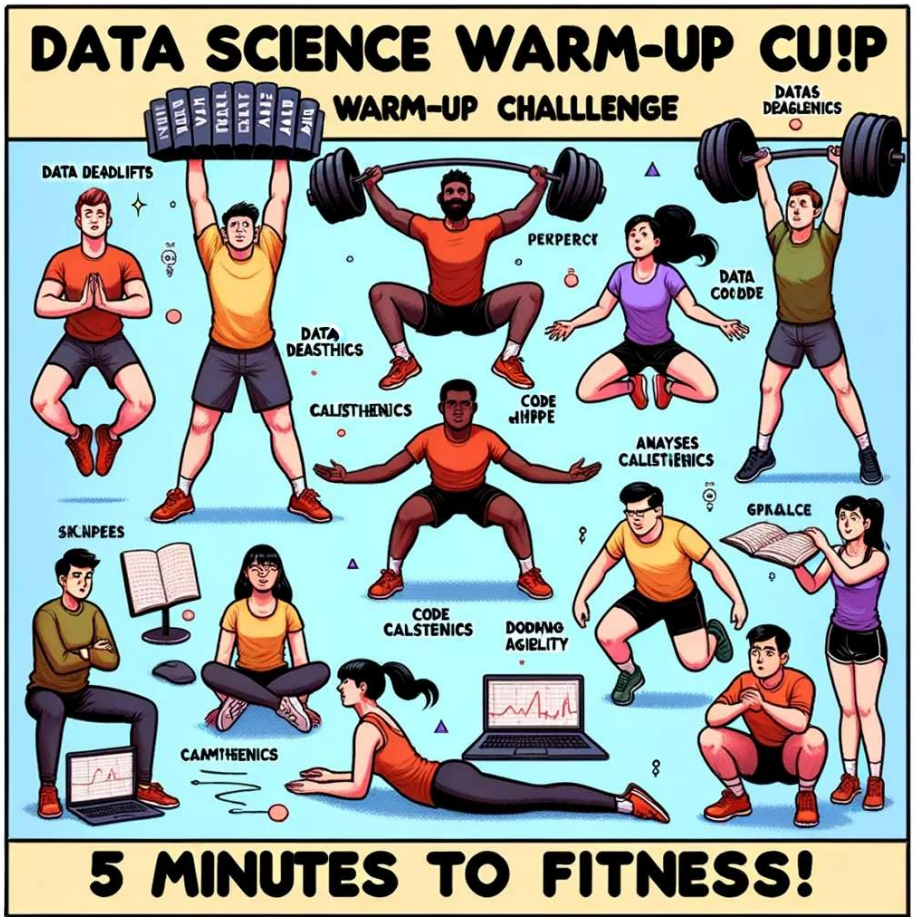
```
pivot_wider(names_from = "year",
            names_prefix = "yr",
            values_from = "metric")
```

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")
```

Long and wide data. Source: Hugo Tavares

Warm up


DATA SCIENCE WARM-UP CU!P
WARM-UP CHALLLENGE
DATA DEADLIFTS
DATAS DEAGLENICS
PERPERCT
DATA
DATA DEASTHICS
CALISTHENICS
CODE
ANAYSES
CALISTIENICS
SKNPEES
CODE
CALSTENICS
AGILITY
CAMITHENICS
5 MINUTES TO FITNESS!

Reshaping: multiple/one/none answers correct

Consider the following **R** code that creates the data frame schwiizerChuchi:

```
schwiizerChuchi <- data.frame(
  Region = c("Zurich", "Geneva", "Lucerne"),
  Fondue = c(8, 9, 7),
  Raclette = c(7, 8, 10),
  Rosti = c(9, 6, 8),
  Olma = c(10, 7, 8)
)
```

This dataset records the popularity ratings (on a scale of 1 to 10) of various Swiss dishes in different regions of Switzerland.

```
schwiizerChuchiLong <- pivot_longer(schwiizerChuchi,
                                    cols = c(Fondue, Raclette, Rosti, Olma),
                                    values_to = "Popularity",
                                    names_to = "Dish")
```

Which of the following statements is true?

- **`nrow(schwiizerChuchiLong) == 12`** returns **`TRUE`**
- **`dim(schwiizerChuchiLong)`** returns **`c(3, 12)`**
- **`dim(schwiizerChuchi)`** returns **`c(3, 12)`**

Tidy data: essay question

Why is this data frame not tidy, and what would you do to make it tidy? Write down your reasoning in numbered steps. You can write down some exact code, some higher-level code concepts, or in plain text.

```
temp_location_data <- data.frame(
  temperature_location = c("22C_London", "18C_Paris", "25C_Rome")
)
```

Tidy data: essay question

Why is this data frame not tidy, and what would you do to make it tidy? Write down your reasoning in numbered steps. You can write down some exact code, some higher-level code concepts, or in plain text.

```
grades_data <- data.frame(
  Student = c("Johannes", "Hannah", "Igor"),
  Econ = c(5, 5.25, 4),
  DataHandling = c(4, 4.5, 5),
  Management = c(5.5, 6, 6)
)
```

Data Analysis with R

Data Analysis with R

1. Stacking (recap last time)
2. Merging (joining) datasets
3. Data manipulation with `tidyverse()`
4. Aggregation of statistics

Stack/row-bind: the concept (recap)

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` (recap)

- 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
```

Stack/row-bind: code example

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

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

Stack/row-bind: code example

```
subset1; subset2; subset3
```

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

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

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

```
# Stack data frames and inspect results
combined_df_bind_rows <- bind_rows(subset1, subset2, subset3)
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
```

Stack/row-bind: code example

```
subset1; subset2; subset3
```

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

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

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

```
# Stack data frames and inspect results
combined_df_rbind     <- rbind(subset1, subset2, subset3)
```

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

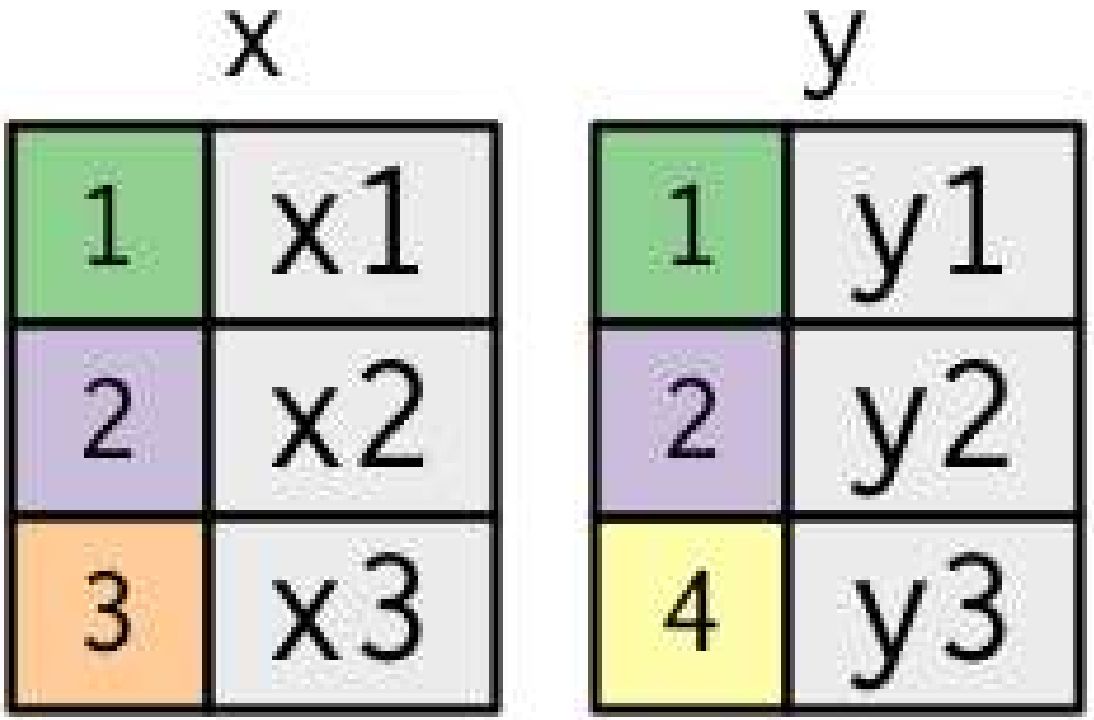
```
combined_df_rbind
```

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

Merging (Joining) datasets

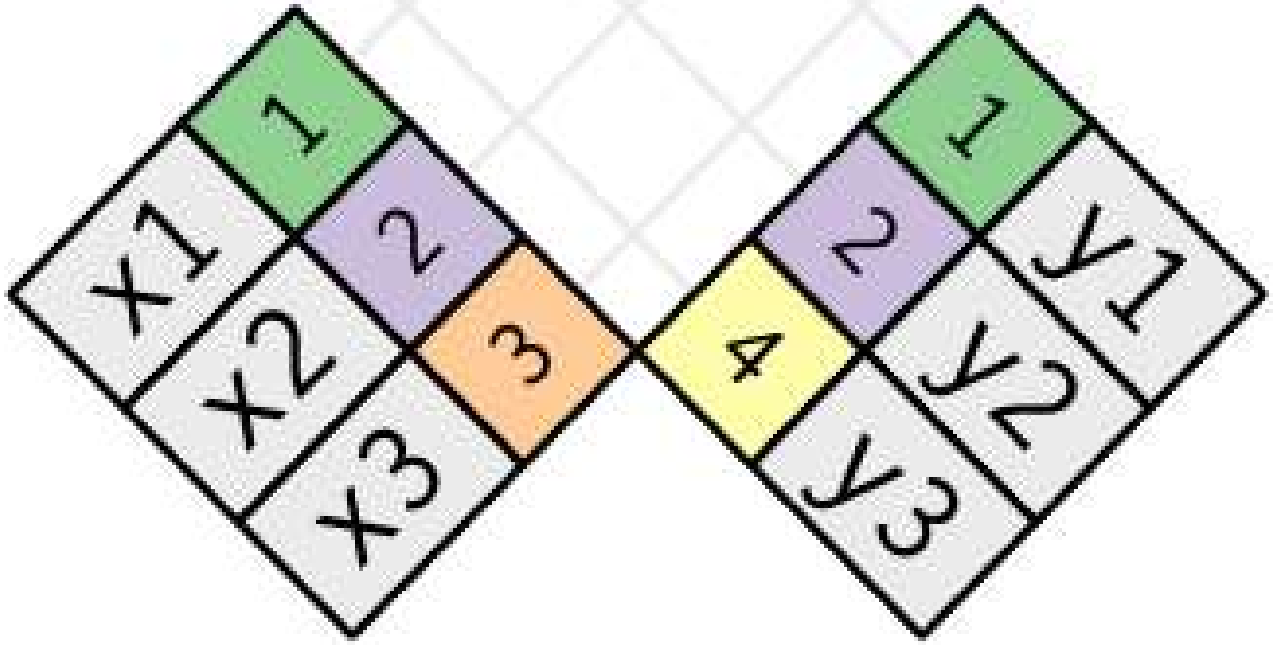
- Combine data of two datasets in one dataset.
- Needed: Unique identifiers for observations (‘keys’).

Merging (joining) datasets: concept



Join setup. Source: Wickham and Grolemund (2017), licensed under the Creative Commons Attribution-Share Alike 3.0 United States license.

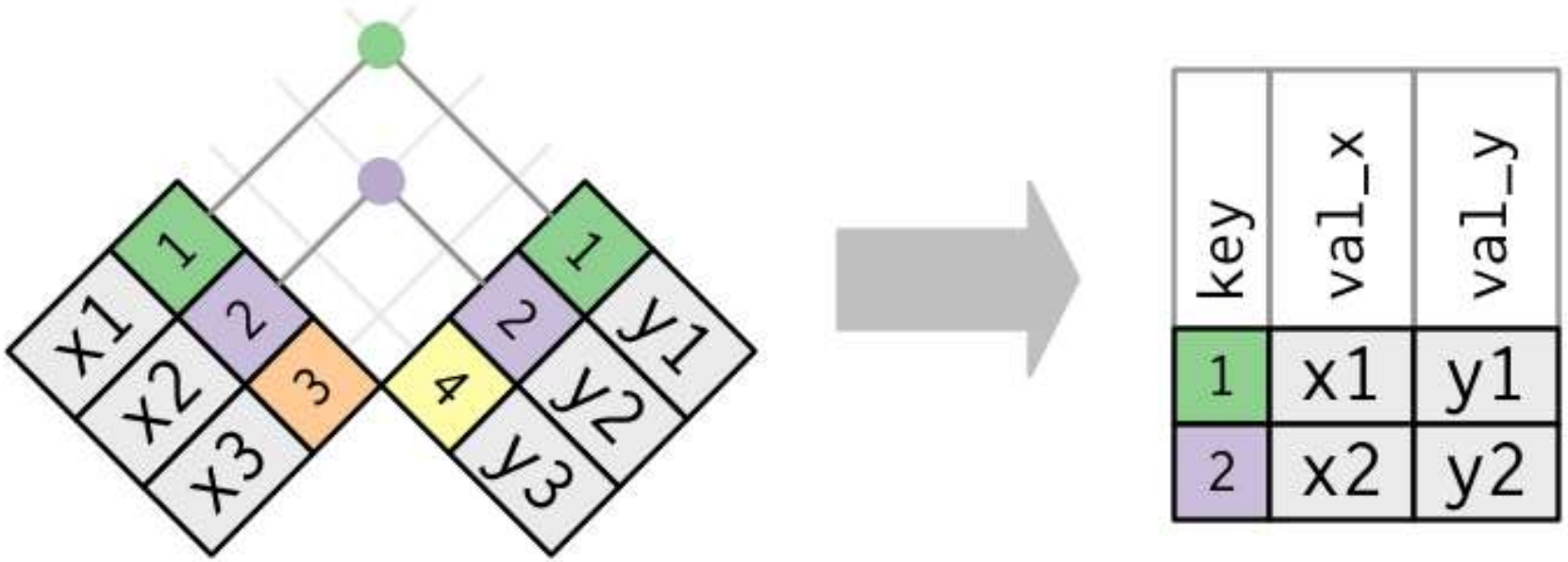
Merging (joining) datasets: concept



Join setup. Source: Wickham and Grolemund (2017), licensed under the Creative Commons Attribution-Share Alike 3.0 United States license.

Merging (joining) datasets: concept

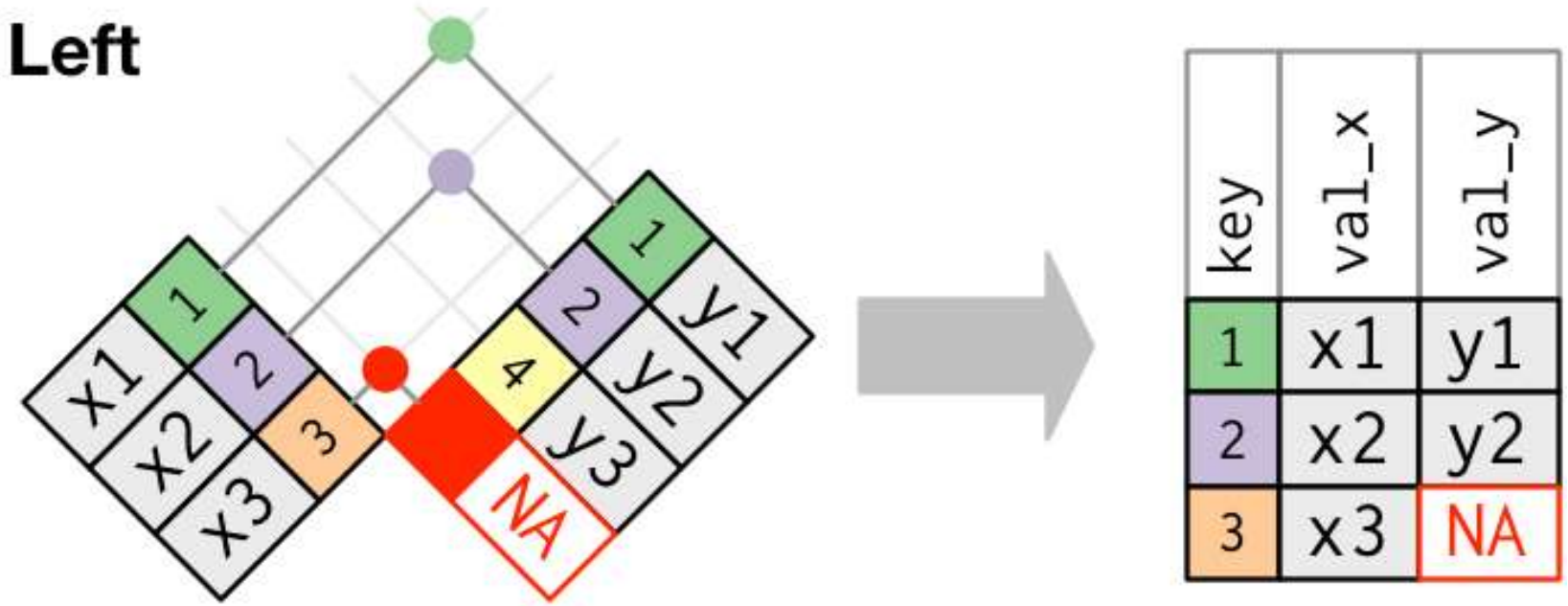
Merge: Inner join



Inner join. Source: Wickham and Grolemund (2017), licensed under the Creative Commons Attribution-Share Alike 3.0 United States license.

Merging (joining) datasets: concept

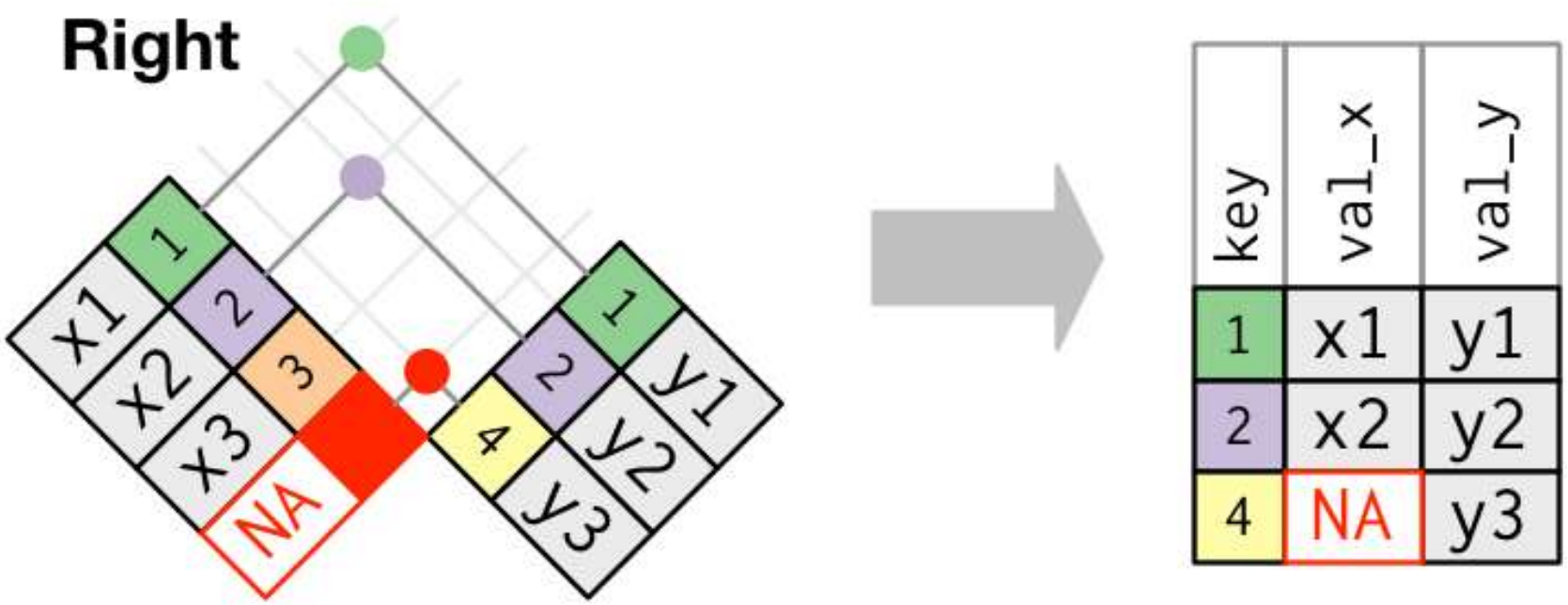
Merge all x: Left join



Outer join. Source: Wickham and Grolemund (2017), licensed under the Creative Commons Attribution-Share Alike 3.0 United States license.

Merging (joining) datasets: concept

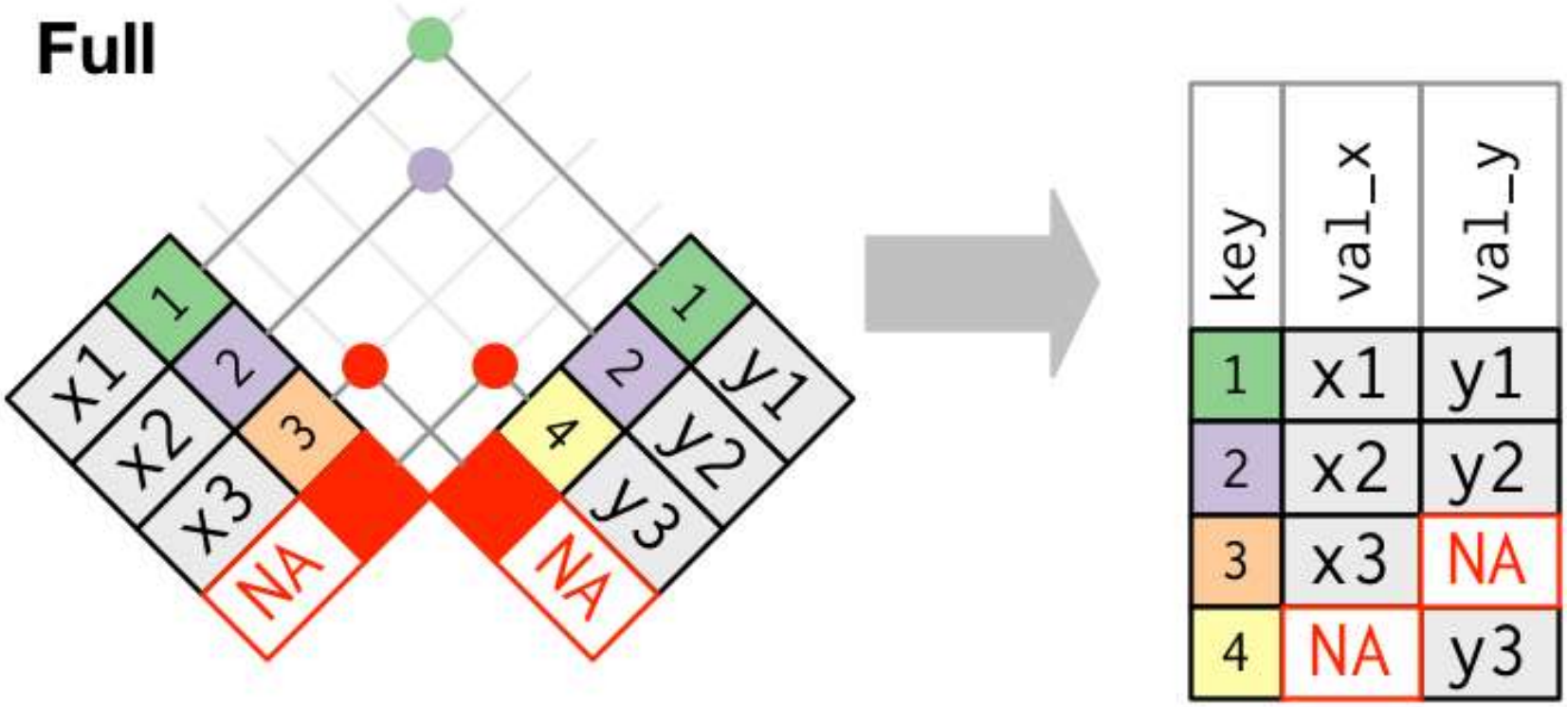
Merge all y: Right join



Outer join. Source: Wickham and Grolemund (2017), licensed under the Creative Commons Attribution-Share Alike 3.0 United States license.

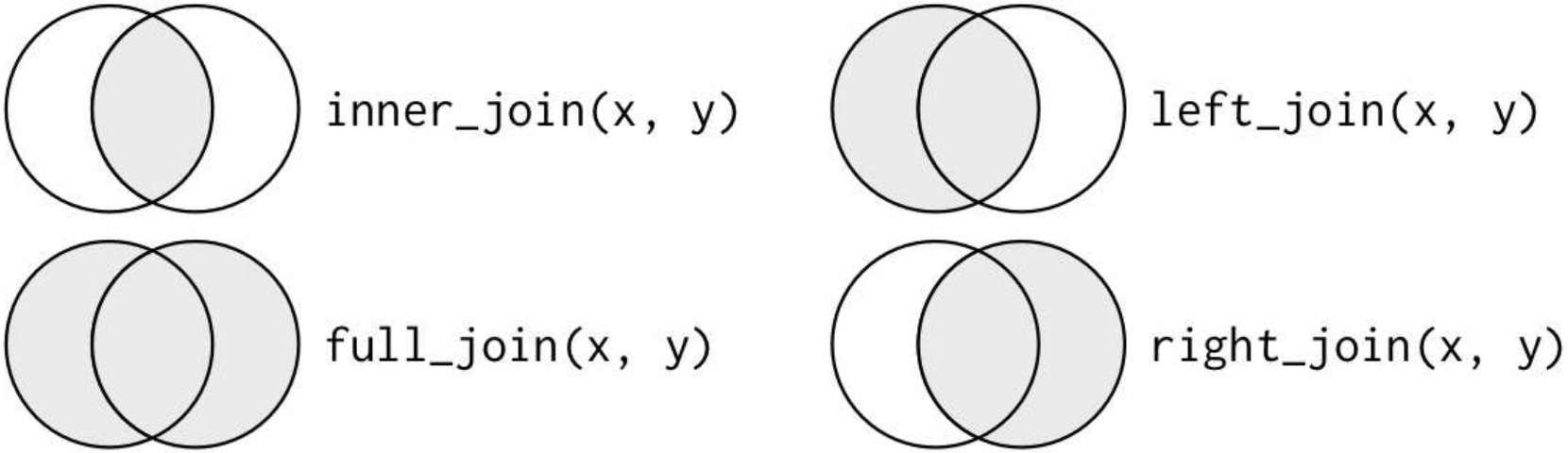
Merging (joining) datasets: concept

Merge all x and all y: Full join



Outer join. Source: Wickham and Grolemund (2017), licensed under the Creative Commons Attribution-Share Alike 3.0 United States license.

Merging (joining) datasets: concept



Join Venn Diagramm. Source: Wickham and Grolemund (2017), licensed under the Creative Commons Attribution-Share Alike 3.0 United States license.

Merging (joining) datasets: example

```r
# load packages
library(tidyverse)

# initiate data frame on persons personal spending
df_c <- data.frame(id = c(1:3,1:3),
                   money_spent= c(1000, 2000, 6000, 1500, 3000, 5500),
                   currency = c("CHF", "CHF", "USD", "EUR", "CHF", "USD"),
                   year=c(2017,2017,2017,2018,2018,2018))
df_c
```

```
##   id money_spent currency year
## 1  1        1000      CHF 2017
## 2  2        2000      CHF 2017
## 3  3        6000      USD 2017
## 4  1        1500      EUR 2018
## 5  2        3000      CHF 2018
## 6  3        5500      USD 2018
```

Merging (joining) datasets: example

```r
# initiate data frame on persons' characteristics
df_p <- data.frame(id = 1:4,
                   first_name = c("Anna", "Betty", "Claire", "Diane"),
                   profession = c("Economist", "Data Scientist",
                                  "Data Scientist", "Economist"))
df_p
```

```
##   id first_name     profession
## 1  1       Anna      Economist
## 2  2      Betty Data Scientist
## 3  3     Claire Data Scientist
## 4  4      Diane      Economist
```

Merging (joining) datasets: example

```
df_merged <- merge(df_p, df_c, by="id")
df_merged
```

```
##   id first_name     profession money_spent currency year
## 1  1       Anna      Economist        1000      CHF 2017
## 2  1       Anna      Economist        1500      EUR 2018
## 3  2      Betty Data Scientist        2000      CHF 2017
## 4  2      Betty Data Scientist        3000      CHF 2018
## 5  3     Claire Data Scientist        6000      USD 2017
## 6  3     Claire Data Scientist        5500      USD 2018
```

Move to Nuvolos

Merging (joining) datasets: R

Overview by Wickham and Grolemund (2017):

dplyr (tidyverse)	base::merge
`inner_join(x, y)`	`merge(x, y)`
`left_join(x, y)`	`merge(x, y, all.x = TRUE)`
`right_join(x, y)`	`merge(x, y, all.y = TRUE)`,
`full_join(x, y)`	`merge(x, y, all = TRUE)`

Selecting, Filtering, and Mutating

Data summaries

- First step of analysis.
- Get overview over dataset.
- Show key aspects of data.
    - Inform your own statistical analysis.
    - Inform audience (helps understand advanced analytics parts)

Data summaries: first steps

- Quick overview: `summary()`
- Cross-tabulation: `table()`

Data summaries and preparatory steps

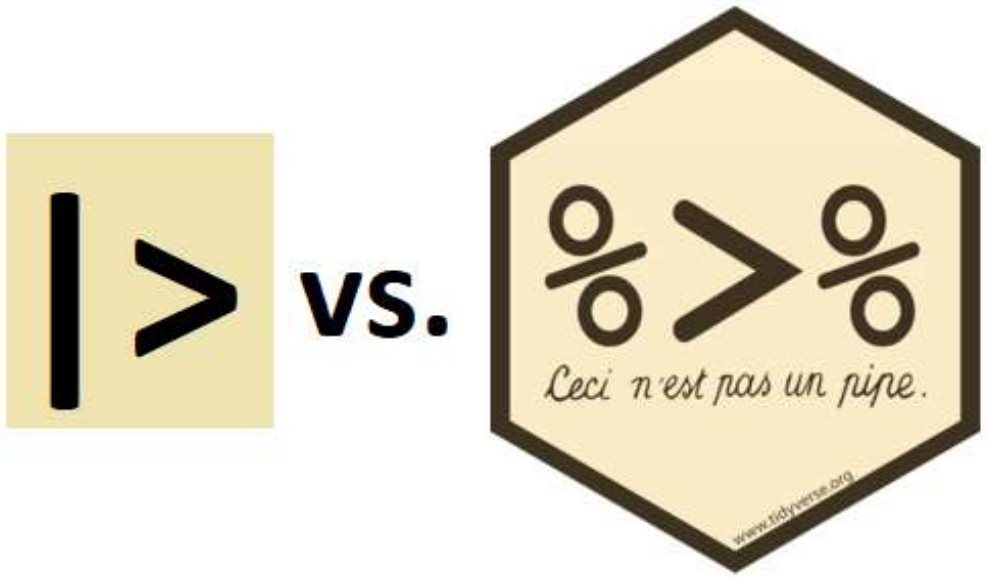
- **Arrange** the dataset by reordering the rows.
- **Select** the subset of variables you need (e.g., for comparisons).
- **Filter** the dataset by restricting your dataset to observations needed in **this** analysis.
- **Mutate** the dataset by adding the values you need for your analysis.

Select, filter, mutate in R (tidyverse)

- `arrange()`
- `select()`
- `filter()`
- `mutate()`

Prepare your data in a `pipe`line

- Using the **`piping %>%`** operator is to chain one function after another without the need to assign intermediate variables.
- The operator has been now replaced with `|>`.



Prepare your data in a pipeline

```r
# Traditional way
mydf <- data(swiss)
mydf <- arrange(mydf, -Catholic)
mydf <- filter(mydf, Education > 8 & Catholic > 90)
mydf <- mutate(mydf, Country = "Switzerland")
mydf <- select(mydf, Examination)

# The pipe way
mydf <- data(swiss)  |>
  arrange(-Catholic) |>
  filter(Education > 8 & Catholic > 90) |>
  mutate(Country = "Switzerland") |>
  select(Examination)
```

Data Summaries: Aggregate Statistics

Descriptive/aggregate statistics

- Overview of key characteristics of main variables used in analysis.
- Key characteristics:
  - Mean
  - Standard deviation
  - No. of observations
  - etc.

Aggregate statistics in R

1. Functions to compute statistics (e.g., `mean()`).
2. Functions to **apply** the statistics function to one or several columns in a tidy dataset.

- Including all values in a column.
- By group (observation categories, e.g. by location, year, etc.)

Aggregate statistics in R

- `summary()` in (`base`)
- `summarise()` (in `tidyverse`)
- `group_by()` (in `tidyverse`)
- `sapply()`, `apply()`, `lapply()`, etc. (in `base`)
- `skimr` package

Move to Nuvolos

Some practice

Summarizing categorical variables: challenge

Use what we just saw in the lecture to solve the following problem. You have the following dataset:

```
df_p <- data.frame(id = 1:5,
                   first_name = c("Anna", "John", "Claire", "Evan", "Brigitte"),
                   profession = c("Economist", "Data Scientist",
                                  "Data Scientist", "Economist", "Economist"),
                   salaryK = c(100, 120, 90, 110, 105),
                   experienceY = c(10, 10, 10, 10, 10))
```

1. Clean the data
2. Summarize the data.
3. Give summary statistics on the categorical variable “profession”. What can you show, and how can you code it?
4. You are interested in quantifying the gender pay gap. Prepare the data accordingly and give an estimate of the gender pay gap.

Q&A

References

Wickham, Hadley, and Garrett Grolemund. 2017. Sebastopol, CA: O'Reilly. http://r4ds.had.co.nz/.