

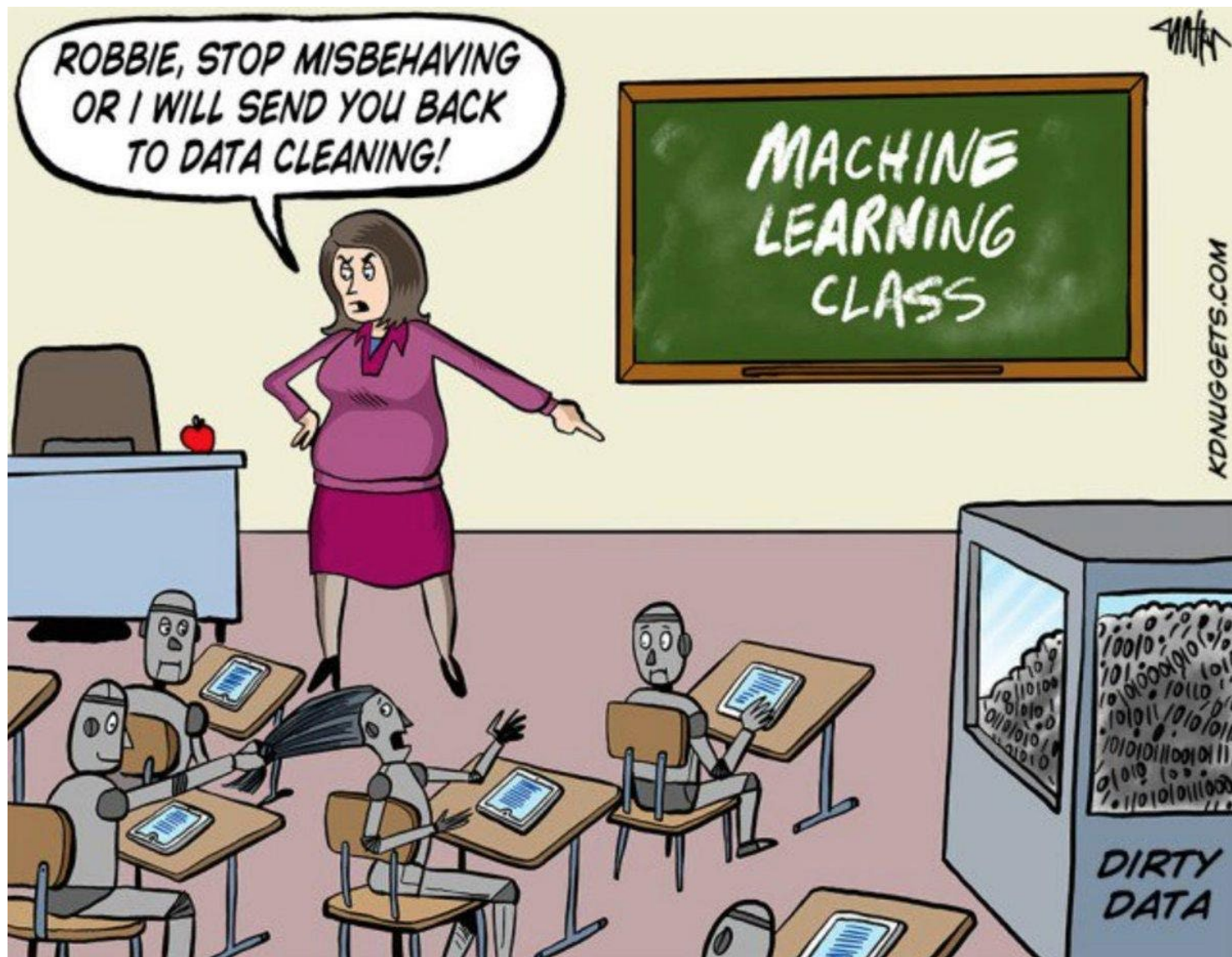
Data Science Techniques and Real-World Applications

Data Wrangling (Munging)

Zahra Sharafi

<https://sites.google.com/view/zahrasharafi/home>

Email: z.sharafi@fs.de



Agenda

1. Data Wrangling (ch2. Data Analytics for Business, Economics, and Policy)
 - understand types of variables and observations;
 - organize data in a tidy way;
 - merging tidy tables
 - clean the data:
 - identify and address problems with observations and variables;
 - create a reproducible workflow to clean and organize data;
 - document data cleaning and understand such documentation.
2. Pandas



Types of Variables

- **Quantitative** variables:
 - The data born as numbers (*integers, floats,...*)
 - Two subtypes:
 - **Flow variables**: measured in a time-frame. E.g., sales in a month
 - **Stock variables**: measured at a point in time. E.g., Closing price of Apple Inc.
- **Qualitative** variables:
 - Could be numbers in discrete values
 - **Categorical** or Factors
 - In the form of labels(text → *strings*), or self-defined numbers
 - E.g., Headquarter: (0) US , (1) EU, (2) RoW
 - Special case: **Binary variables** (*Boolean*, dummy, indicator variables) with 0/1 values

Types of Observations

Observation type	Observations	ID variable	Example
Cross-sectional (xsec)	x_i : different cross-sectional units observed at same time	Identifies each cross-sectional unit	People, companies, countries observed in the same year/month
Time series (tseries)	x_t : same cross-sectional unit observed at different time periods	Identifies each time period	A single person/ company/country observed at different times
Cross-section time series (xt, longitudinal, or panel)	x_{it} multiple cross-sectional units observed across multiple time periods	One ID identifies cross-sectional units; one ID identifies time periods	Retailer stores observed weekly, countries observed yearly

Tidy Data

A tidy dataset, organized as one or many data tables in a way that:

1. In a data table, each observation forms a row.
2. In a data table, each variable forms a column.
3. Each kind of observation forms a data table.
4. In the relational datasets: A clear link exists (and is documented) between the tables

Example:

A dataset on customer purchases of various products:

- a data table on customers (their age, income, and so on) > *unique customer ID*
- a data table on products (type, brand, price, quality indicators) > *unique product ID*
- a data table on purchase events (which customer purchased which product > *unique purchase ID that maps customer and product IDs*

Work file

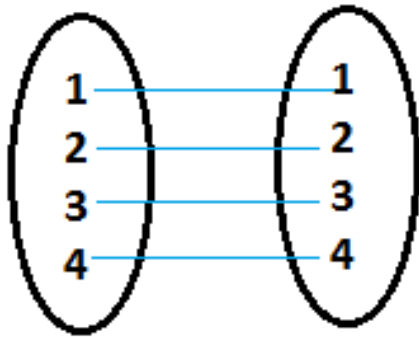
You have a data analysis task to perform:

1. Make sure your dataset is tidy
2. Create a work file:
 - combine the variables you need from the various data tables to form a single data table
3. Do the analysis (later in the course!)

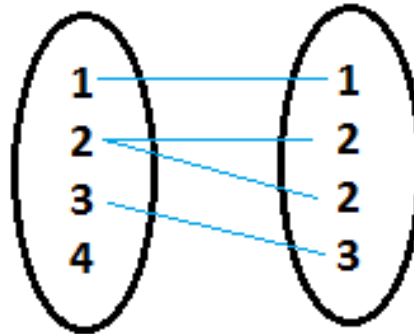
The process of pulling different variables from different data tables for well-identified entities to create a new data table is called **linking, joining, merging, or matching** data tables.

Merging

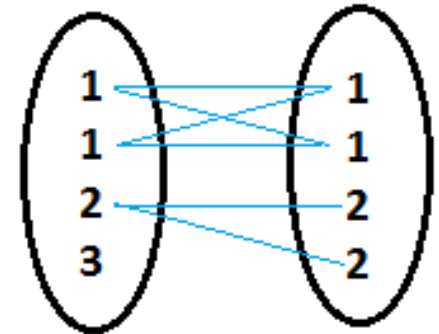
- Merging types:



One-to-One



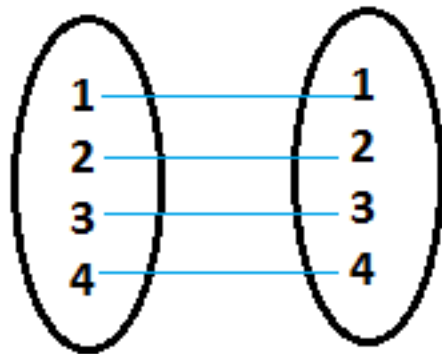
One-to-Many



Many-to-Many

Merging

- Merging types:



One-to-One

One-to-One Merging

geography.dta

country	Land Area (sq km)
ARG	2,736,690
FRA	640,053
GER	349,223
ITA	294,020
USA	9,161,923

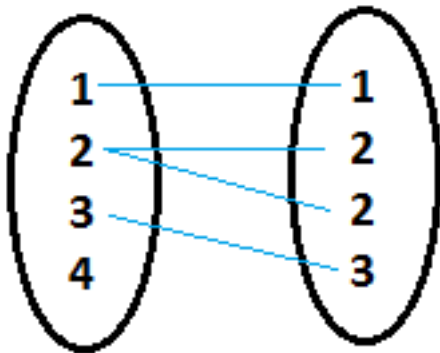
economy.dta

country	GDP per Capita
ARG	12,468
FRA	27,913
GER	28,889
ITA	28,172
USA	39,498

country	Land Area (sq km)	GDP per Capita
ARG	2,736,690	12,468
FRA	640,053	27,913
GER	349,223	28,889
ITA	294,020	28,172
USA	9,161,923	39,498

Merging

- Merging types:



One-to-Many

One-to-Many Merging

individuals.dta

indiv_ID Years of Schooling fam_ID

1	12	1
2	12	1
3	8	1
4	9	1
5	18	2
6	22	2
7	3	2

households.dta

fam_ID Food Expenses

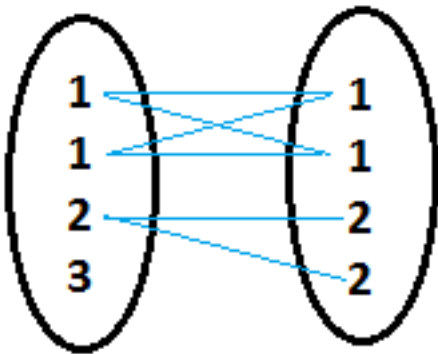
1	8,543
2	7,872

indiv_ID Years of Schooling fam_ID Food Expenses

1	12	1	8,543
2	12	1	8,543
3	8	1	8,543
4	9	1	8,543
5	18	2	7,872
6	22	2	7,872
7	3	2	7,872

Merging

- Merging types:



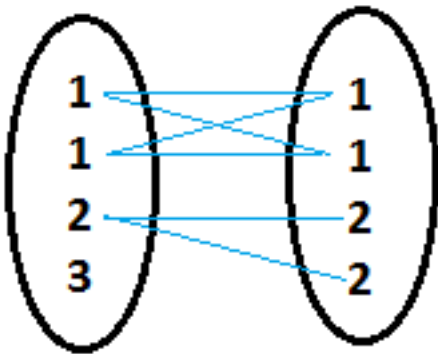
Many-to-Many



- Be careful, m:m merging is rarely needed and results in an untidy data!
 - Try to formulate your questions in a way to be able to work in a series of m:1 matchings.
 - In case of tables with m & n observations, be sure that your machine is capable of having $m \times n$ obs table in memory.
- Always check your post merging data (in pandas: “Validate”)

Merging

- Merging types:



Many-to-Many

	student ID	course	Grade
0	1	math	B
1	1	biology	B
2	2	math	A
3	2	biology	F

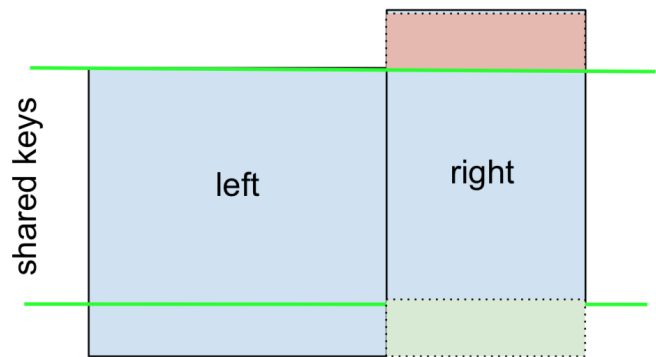
	Grade	Points
0	A	10
1	B	9
2	B	8
3	C	7
4	C	6
5	D	5
6	D	4
7	E	3
8	F	2
9	F	1
10	F	0



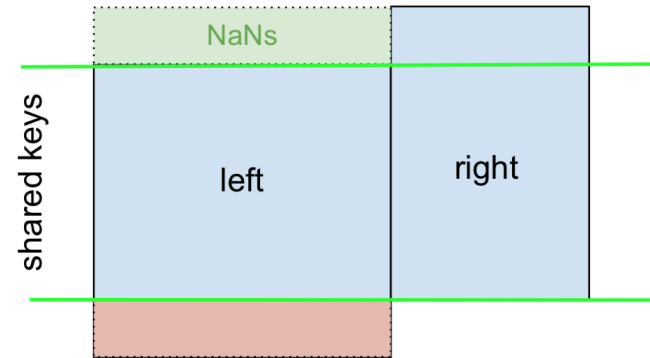
	student ID	course	Grade	Points
0	1	math	B	9
1	1	math	B	8
2	1	biology	B	9
3	1	biology	B	8
4	2	math	A	10
5	2	biology	F	2
6	2	biology	F	1
7	2	biology	F	0

Merging

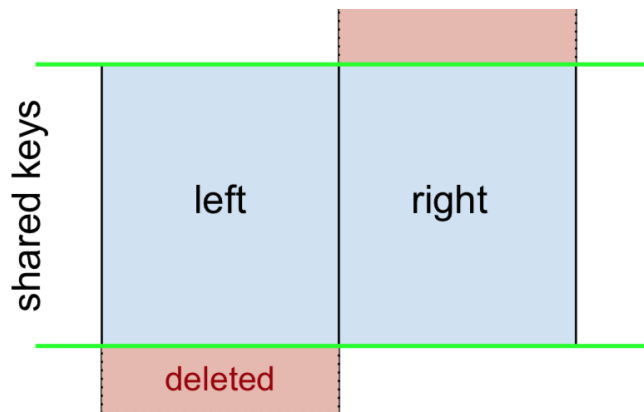
When merging two data tables, be that 1:1, 1:m, m:1, or m:m matching, data analysts may be able to link all observations or only some of them.



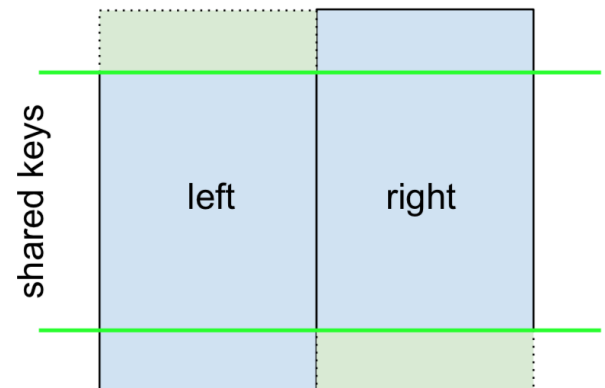
how='left'



how='right'



how='inner'



how='outer'

Merging

Example of one-to-many, left merge:

	id	name
0	1	Tom
1	2	Jenny
2	3	James
3	4	Dan

	id	product	quantity	date
0	2	A	31	2019-02-24
1	2	B	21	2019-02-25
2	4	A	20	2019-02-26
3	4	C	40	2019-02-27

df_customer

df_order_2

```
merge(df_customer, df_order_2, on='id', how='left')
```

	id	name	product	quantity	date
0	1	Tom	NaN	NaN	NaT
1	2	Jenny	A	31.0	2019-02-24
2	2	Jenny	B	21.0	2019-02-25
3	3	James	NaN	NaN	NaT
4	4	Dan	A	20.0	2019-02-26
5	4	Dan	C	40.0	2019-02-27

Entity Resolution

Think deeply about the observation level of your data:

- If you have **a cross-sectional** dataset, e.g., on countries, each row should represent one country.
- If you have **panel data**, e.g., annual revenue of several companies, your observation unit is firm-year.

1. Check for duplicates at the observation level

- Duplicates could occur during the data collection or data processing
- Drop duplicates
 - Be careful, sometimes duplicates contain additional information:

Firm	year	...	State
Apple Inc.	2020	...	N/A
Apple Inc.	2020	...	CA

2. Ambiguous identifications:

- Avoid using names (*strings*) as identifiers

Team ID	Unified name	Original name
19	Man City	Manchester City
19	Man City	Man City
19	Man City	Man. City
19	Man City	Manchester City F.C.
20	Man United	Manchester United
20	Man United	Manchester United F.C.
20	Man United	Manchester United Football Club
20	Man United	Man United

Missing Values


Why are missing values problematic?

1. Difficult to identify:
 - If they are replaced with zeros in quantitative variables...
 - In a column of *string* variables, the value is like “N/A...”
 - The problem: your software (e.g., Pandas, R, SQL) won’t automatically recognize those as *missing*. It will treat them as *real values*.

→ *That’s why in data cleaning, we try to explicitly convert all “fake missing indicators” (0, "N/A", "?", "None") into proper NaN values. Then they are easier to detect and handle.*
2. Reduction in the number of observations.
3. **Sample selection bias**
 - If the missing values are not happening randomly...
 - Check the severity of the issue with “**benchmarking**”
 - Use the distribution of other variables and compare them for the missing and non-missing observations
 - How was the data born? Is it a survey?...

Missing Values

Handling missing values:

0. Make sure that there is no alternative dataset
 - Check for non-ideal alternatives
1. Ignore them (?)
 - Check for **sample selection bias**
 - Missing values are not many
 - Or, only a subsample of data has most of the missing values > drop that subsample in a **robustness check** (example: missing values for smaller companies)
2.  Imputation
 - E.g., replace missing values with the mean, or higher degree polyfits
 - This doesn't add information!

Cleaning Data

Review Box 2.4 Data wrangling: common steps

1. Write code – it can be repeated and improved later.
2. Understand the types of observations and what actual entities make an observation.
3. Understand the types of variables and select the ones you'll need.
4. Store data in tidy data tables.
5. Resolve entities: find and deal with duplicates, ambiguous entities, non-entity rows.
6. Get each variable in an appropriate format; give variable and value labels when necessary.
7. Make sure values are in meaningful ranges; correct non-admissible values or set them as missing.
8. Identify missing values and store them in an appropriate format; make edits if needed.
9. Make a habit of looking into the actual data tables to spot issues you didn't think of.
10. Have a description of variables.
11. Document every step of data cleaning.

README

- No one can read your mind >> create a README
- A .txt or .md file located in the root folder
- Contents:
 - Title for the dataset
 - Contact data
 - Data origin
 - Folder structures
 - For each filename, a short description of what data it contains
 - Licences or restrictions placed on the data
 - Description of methods used for data processing (describe how the data were generated from the raw or collected data)
 - Variable list, including full names and definitions (spell out abbreviated words) of column headings for tabular data
 - Definitions for codes or symbols used to record missing data
- Use templates: <https://data.research.cornell.edu/content/readme>

```
```bash
|- README.md
|- FolderExample
| |- fileA.xls
| |- fileB.csv
|- Code1.py
...
```
```

Now to the code...

- P3&4 Data Wrangling.ipynb

And

- Homeworks 1, 2, and 3.
- Submission delivery: Wednesday before the next session.