2. Data Management

Data Management
Fall 2020
Osaka U

Shuhei Kitamura

Outline

- Data structure
- Relational databases
- Changing the data structure
- Directory structure
- Naming tips

Data structure

Cross-sectional data at the county level in the U.S.

county	state	cnty_pop	state_pop	region
36037	NY	3817735	43320903	1
36038	NY	422999	43320903	1
36039	NY	324920	.	1
36040	.	143432	43320903	1
.	VA	.	7173000	3
37001	VA	3228290	7173000	4

Source: Code and Data for the Social Sciences

Can you spot what's wrong with the data?

Data structure (cont.)

county	state	cnty_pop	state_pop	region
36037	NY	3817735	43320903	1
36038	NY	422999	43320903	1
36039	NY	324920	.	1
36040	.	143432	43320903	1
.	VA	.	7173000	3
37001	VA	3228290	7173000	4

- County id missing for one observation
  - Do we really need that observation?
- State name and state_pop missing
  - Guessing from state population, state name for 36040 may be “NY”
  - Guessing from state name, state population for 36039 may be “43320903”
  - But can we trust state population and state name?
- There are two different regional numbers for VA

Data structure (cont.)

If data look like this, you cannot readily use them in your analysis

- Can you use the data in different projects or share them with others? (Can you recall all the problems that you have just found in future?)

→ You need to clean the data first and store cleaned ones

Relational databases

county	state	population
36037	NY	3817735
36038	NY	422999
36039	NY	324920
36040	NY	143432
37001	VA	3228290

state	population	region
NY	43320903	1
VA	7173000	3

Two separate data (tables): a county table and a state table

- Each table is self-documenting. For example, it is clear to anyone that population in the state table means state population

Each table has a *key*, or an *id*

- Any id should not be missing or duplicated

They are called a *relational database*

Relational databases (cont.)

county	state	population
36037	NY	3817735
36038	NY	422999
36039	NY	324920
36040	NY	143432
37001	VA	3228290

state	population	region
NY	43320903	1
VA	7173000	3

A table can contain a *foreign key (id)*

- It is used when you merge tables
- E.g. "state" in the county table

Data stored in this form are considered *normalized*

- By contrast, if the county table contains variables that are not properties of counties, the data are *not* normalized

We need data cleaning just for making normalized data

- Later, you will merge tables, make new variables (e.g. log population), etc. using normalized data

Relational databases (cont.)

county	state	population
36037	NY	3817735
36038	NY	422999
36039	NY	324920
36040	NY	143432
37001	VA	3228290

state	population	region
NY	43320903	1
VA	7173000	3

Keep the original information as much as possible

- Do not delete or add information beyond necessity

Do not make data just for a single project

- You may use them in another project and/or release them to a broad group of users with differing needs in future

Changing the data structure

There are several ways to change the structure of your data

Most common operations are:

- Append
- Merge
- Collapse (Aggregate)
- Reshape

Append

county	state	population
36037	NY	3817735
36038	NY	422999
36039	NY	324920
36040	NY	143432

+

county	state	population
37001	VA	3228290

You use append when you want to combine two datasets vertically

Some columns may not exist in one of the datasets

- For example, Pandas' `concat` function allows you to choose how you treat columns that do not exist in one of the datasets

Check if columns with the same name are the same type

- If not, the types of columns in the bottle table will be automatically converted to those of columns in the top table

Merge

county	state	population
36037	NY	3817735
36038	NY	422999
36039	NY	324920
36040	NY	143432
37001	VA	3228290

+

state	population	region
NY	43320903	1
VA	7173000	3

You use merge when you want to combine two datasets horizontally

At least one of the datasets must have a unique key

Check if two datasets contain different content with the same column name

- The worst case would be that one column overwrites the other one
- For example, Pandas' `merge` function keeps both columns

Check if two keys are the same type

- Typical mistake: One key is string, the other is integer

Collapse (Aggregate)

county	state	population
36037	NY	3817735
36038	NY	422999
36039	NY	324920
36040	NY	143432
37001	VA	3228290

→

state	population
NY	4709086
VA	3228290

You use collapse when you want to collapse a dataset (and compute the aggregated values)

Check if some columns include missing values

- For example, Pandas' `groupby` and `agg` methods ignore missing values

Reshape

county	state	year	population
36037	NY	2018	3817735
36037	NY	2019	3818735
36037	NY	2020	3820735
36038	NY	2018	422999
36038	NY	2019	423000
36038	NY	2020	424000

↕

couty	state	population2018	population2019	population2020
36037	NY	3817735	3818735	3820735
36038	NY	422999	423000	424000

You use reshape when you want to change the format of the dataset

- The format of the top dataset is called the *long* format, while that of the bottom dataset is called the *wide* format

The long format is used for e.g. panel data analysis

Directory structure

A directory $\approx$ a folder

How many directories and subdirectories do I need?

How should I structure them?

- Make a directory for each project/data?

How should I name them?

Directory structure (cont.)

Directories should have the following structure

- **(a)** A directory and subdirectories for cleaning data
- **(b)** A directory and subdirectories for analyzing data and writing papers & slides

Example:

(a) `clean_xxx`

- `code`: code files
- `input`: data to be cleaned
- `temp`: temp files
- `output`: cleaned data

(b) `project_yyy`

- `code`: code files
- `input`: combined data
- `temp`: temp files
- `output`: tables, figures, tex files
  - `tables`
  - `figures`

Workflow

General workflow is the following:

(1) In (a), put uncleaned data in `input`, clean them, and produce cleaned data in `output`

(2) In (b), import data from `output` in (a), produce combined data in `input`, analyze them, and produce tables and figures in `output`

- Make sure that you will not overwrite the data

(3) Also in (b), write papers and slides in `output`

If data are already clean, you can skip **(1)**

Why a separate directory for data cleaning?

Since you might use the same data in several projects and/or share them with others, it is useful to have a separate directory just for cleaning data

Several advantages:

- You do not need to import all irrelevant files in another project
- You can easily find relevant code just for cleaning and analyzing data
- You can easily share code and data

Naming tips

Directory names

- No single answer. Use the simplest and self-explanatory one
- Shorten words and use underscores (not hyphens!) if needed
- Using names like `clean_xxx` and `project_yyy` allows you to easily recognize the purposes of the directories
  - You may even consider making parent directories, `clean` and `project`, and put the subdirectories `xxx` and `yyy` in them, respectively

Subdirectory names

- Recommended: `code`, `input`, etc. as already mentioned above

Naming tips (cont.)

File, variable, function names

- No single answer. Use the simplest and self-explanatory one
- Shorten words and use underscores (not hyphens!) if needed
- E.g., `ln_pop`, `conv_int_str()`

You SHOULD NOT use

- An abstract name (e.g., `var1`, `conv_i_s`)
- A long name (e.g., `log_state_population_of_year_2000`)
- A name starting from a number (e.g., `1984_year`)
- A name with symbols (e.g., `rate_%`)

Summary

- Data structure
- Relational databases
- Changing the data structure
- Directory structure
- Naming tips