Week 2.b: Data structuring

Overview remainder today

- The pandas way
- Some new data types
- Advanced concepts
  - Split-apply-combine
  - Combining datasets
- Exercises: apply concepts and more details

Course survey

https://andreasabn.typeform.com/to/NGxcXV
(https://andreasabn.typeform.com/to/NGxcXV)

Motivation

Why learn data structuring?

- Necessary step before modelling
- Data driven thesis in economics - 80-90 pct. data structuring
- Big data prevalence

What is data structuring?

- Moving from raw data to analysis ready
- Social Fabric example
  - 200 million GPS samples
  - 40 million Bluetooth meetings
  - 100k calls / text messages

What is data structuring? (2)

Examples

- Converting data types
- Filtering data
- Aggregating over periods, users etc

The pandas module

Pandas

Why use Pandas?

1. easy to learn
    - built with Python's simplicity
    - great documentation
2. provides powerful tools for smart data structuring
3. speed - leverages numpy's fast computatition
4. development - breathtaking speed of new tools coming
5. use modern data science approach
    - makes packages in R’s tidyverse easy to undestand
    - can use R's packages directly

Pandas (2)

What is pandas? What does it add to numpy and basic Python?

- pandas is a framework where rows and columns are labelled. It provides som useful features for structuring data.
- Think of pandas is a mix of Python lists and dictionaries, providing access via positions and labels.
- pandas have some unique capabilities for data structuring.

Pandas (3)

What is a pandas ***Series***?

- A vector/one dimensional array.
- Each row has a label.

What is a pandas ***DataFrame***?

- A matrix/ two dimensional array.
- Each row has a label, each column has name.

Pandas (4)

Why are these data structures smart? Lots of useful tools:

- Manipulate columns numerically (all normal python works: `+`, `-`, `*`, `/`, `**`) or as strings
- Select subsets of data
- Sort data, change labels (indices or columns)
- Much more

Pandas (5)

How are we going to learn to use this

- your preparation?
- lecture > advanced concepts
- practice > exercises, external resources, summer course

Data types

Recap

What are some of the fundamental data types?

-
-

Data types

Basic data types

- String
- Numeric (float, integer)

Pandas extends this with (see: exercises)

- Categorical
- Temporal

What is not here?

Data types (2): missing data

Suppose we have the following 3 x 3 matrix with some missing elements:

.. *1 3*

2 1 ..

.. *6* ..

Data types (3): missing data in pandas

In numpy and pandas missing data is denoted as "Not a Number", i.e. NaN.

It is available as `np.nan`.

In [ ]:
```
import numpy as np
import pandas as pd

data_input = [[np.nan, 1],
              [2, 1]]

example_df = pd.DataFrame(data = data_input,
                          columns = ['A', 'B'])

print(example_df)
```

Data types (4): advice on missing data

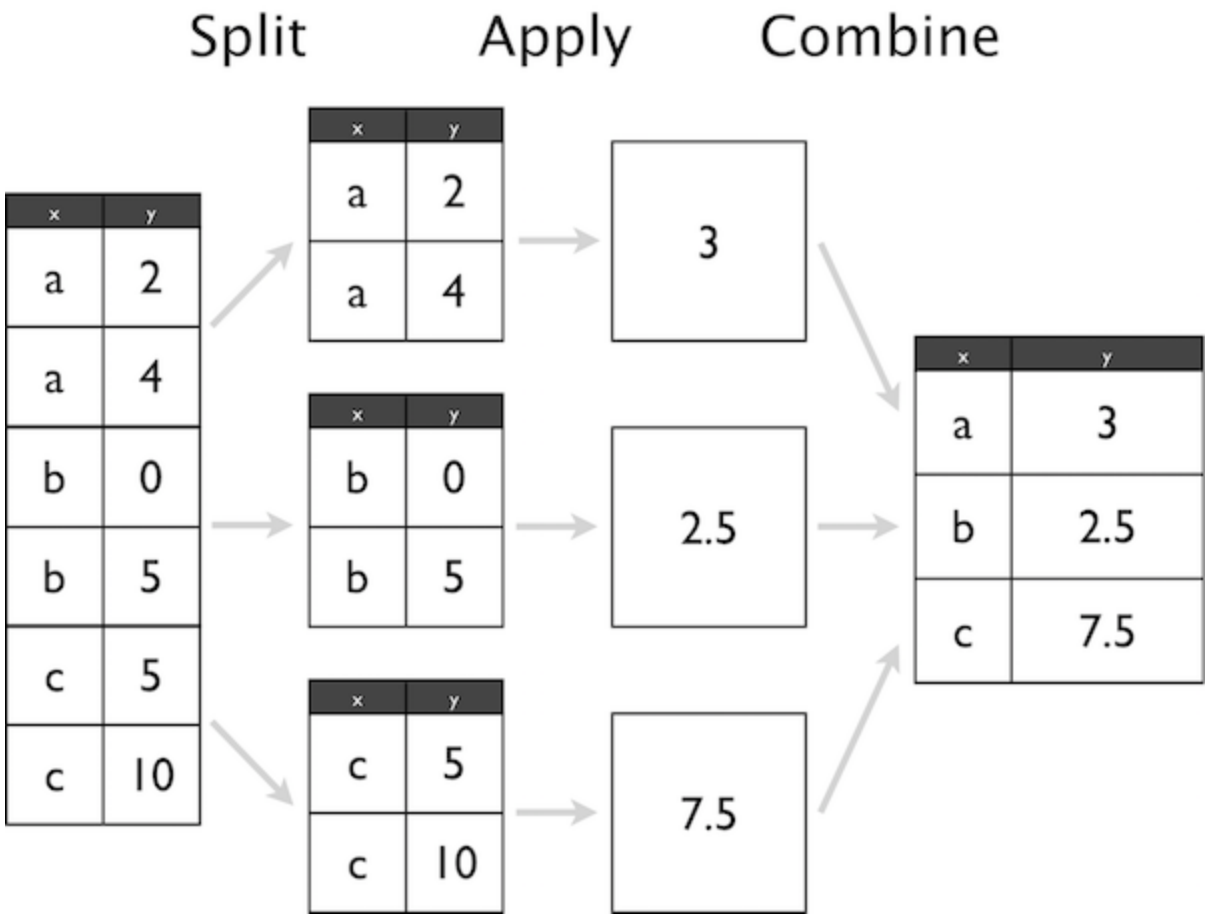
- be ***always alert*** of missing data!
- be explicit (drop, ignore, replace)
- careful of expression such as X>=1
  - False if missing or less than 1
- in computation, often dropped:
  - mean, median
  - models drop missing

In [ ]:
```
example_df.mean()#(skipna=True)
```

Split-apply-combine

The groupby method

Example: grouping by ***x*** and calculating mean of ***y***



The groupby method (2)

Why useful:

- split data based on one or variables, e.g. individual, day, country etc.
- calculate relevant statistics: sums, quantiles, custom functions

The groupby method (3)

Suppose you have a dataset with credit card spending. The data has the following columns: card_id, person_id, timestamp, amount.

When is groupby useful?

- calculating total number of spending per individual?
- compute the total number of spent money?
- find the day with least number of spending?

The groupby method (4)

How do we apply the groupby method?

In [ ]:
```
import pandas as pd

data_input = [[1, 1], [2, 2], [2, 3]]
col_names = ['x', 'y']

df = pd.DataFrame(data=data_input, columns=col_names)
split_vars = ['x']
compute_vars = ['y']

print(df)
#print(df.groupby(split_vars)[compute_vars].mean())
```

Combining datasets

Merging, joining and stacking data

It is possible to join different datasets together into one.

This often necessary to perform analyses.

Core methods

- vertical and horizontally stack datasets (see exercises)
- merging on shared columns and indices

Merging DataFrames

We can merge DataFrames which share common identifiers, row by row. Example:

In [ ]:
```
df1 = pd.DataFrame([[1, 2], [4, 5]], columns=['A', 'B'])
df2 = pd.DataFrame([[2, 3], [6, 7]], columns=["B", 'C'])
```

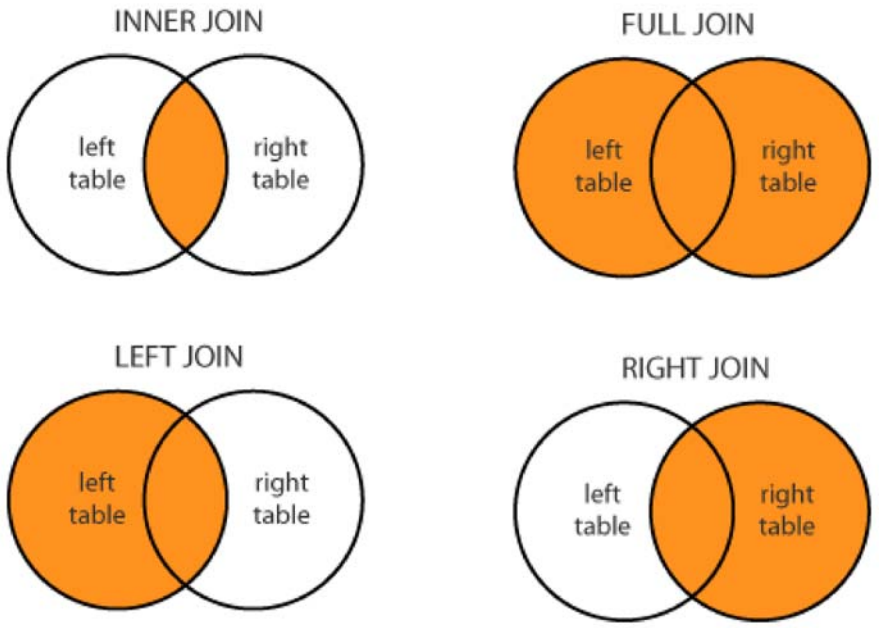
In [ ]:
```
print(df1)
print()
print(df2)
```

In [ ]:
```
print(pd.merge(df1, df2, on=['B'], how='inner'))
```

Merging DataFrames (continued)

Merging can be either of four types.

- inner merge: observations exist in both dataframes [default]
- left (right) merge: observations exist in left (right) dataframe
- outer merge: observations exist either in left or in right dataframe



Round up

- we have learned briefly about dataframes and some core capabilities
- data science is very applied you need to get fingers dirty:
    - exercises, building data pipelines, reading