

10. Combining information in Pandas

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Often information is coming from different sources and it is necessary to combine it into one object. We are going to see the different ways in which information contained within separate Dataframes can be combined in a meaningful way.

10.1 Concatenation

The simplest way we can combine two Dataframes is simply to "paste" them together:

```
In [2]: composers1 = pd.read_excel('Data/composers.xlsx', index_col='composer', sheet
_name='Sheet1')
composers1
```

Out[2]:

	birth	death	city
composer			
Mahler	1860	1911	Kaliste
Beethoven	1770	1827	Bonn
Puccini	1858	1924	Lucques
Shostakovich	1906	1975	Saint-Petersburg

```
In [3]: composers2 = pd.read_excel('Data/composers.xlsx', index_col='composer', sheet
_name='Sheet3')
composers2
```

Out[3]:

	birth	death	city
composer			
Verdi	1813	1901	Roncole
Dvorak	1841	1904	Nelahozeves
Schumann	1810	1856	Zwickau
Stravinsky	1882	1971	Oranienbaum
Mahler	1860	1911	Kaliste

To be concatenated, Dataframes need to be provided as a list:

```
In [4]: all_composers = pd.concat([composers1,composers2])
```

In [5]: `all_composers`

Out[5]:

	birth	death	city
composer			
Mahler	1860	1911	Kaliste
Beethoven	1770	1827	Bonn
Puccini	1858	1924	Lucques
Shostakovich	1906	1975	Saint-Petersburg
Verdi	1813	1901	Roncole
Dvorak	1841	1904	Nelahozeves
Schumann	1810	1856	Zwickau
Stravinsky	1882	1971	Oranienbaum
Mahler	1860	1911	Kaliste

One potential problem is that two tables contain duplicated information:

In [6]: `all_composers.loc['Mahler']`

Out[6]:

	birth	death	city
composer			
Mahler	1860	1911	Kaliste
Mahler	1860	1911	Kaliste

It is very easy to get rid of it using. `duplicated()` gives us a boolean series of duplications and we can just selected non-duplicated rows:

In [7]: `all_composers.duplicated()`

Out[7]:

```
composer
Mahler      False
Beethoven   False
Puccini      False
Shostakovich False
Verdi        False
Dvorak       False
Schumann     False
Stravinsky   False
Mahler       True
dtype: bool
```

```
In [8]: all_composers[~all_composers.duplicated()]
```

```
Out[8]:
```

	birth	death	city
composer			
Mahler	1860	1911	Kaliste
Beethoven	1770	1827	Bonn
Puccini	1858	1924	Lucques
Shostakovich	1906	1975	Saint-Petersburg
Verdi	1813	1901	Roncole
Dvorak	1841	1904	Nelahozeves
Schumann	1810	1856	Zwickau
Stravinsky	1882	1971	Oranienbaum

10.2 Joining two tables

An other classical case is that of two list with similar index but containing different information, e.g.

```
In [9]: composers1 = pd.read_excel('Data/composers.xlsx', index_col='composer', sheet_name='Sheet1')
composers1
```

```
Out[9]:
```

	birth	death	city
composer			
Mahler	1860	1911	Kaliste
Beethoven	1770	1827	Bonn
Puccini	1858	1924	Lucques
Shostakovich	1906	1975	Saint-Petersburg

```
In [10]: composers2 = pd.read_excel('Data/composers.xlsx', index_col='composer', sheet_name='Sheet4')
composers2
```

```
Out[10]:
```

	first name
composer	
Mahler	Gustav
Beethoven	Ludwig van
Puccini	Giacomo
Brahms	Johannes

If we use again simple concatenation, this doesn't help us much. We just end up with a large matrix with lots of NaN's:

```
In [11]: pd.concat([composers1, composers2])
```

```
Out[11]:
```

	birth	death	city	first name
composer				
Mahler	1860.0	1911.0	Kaliste	NaN
Beethoven	1770.0	1827.0	Bonn	NaN
Puccini	1858.0	1924.0	Lucques	NaN
Shostakovich	1906.0	1975.0	Saint-Petersburg	NaN
Mahler	NaN	NaN	NaN	Gustav
Beethoven	NaN	NaN	NaN	Ludwig van
Puccini	NaN	NaN	NaN	Giacomo
Brahms	NaN	NaN	NaN	Johannes

The better way of doing this is to **join** the tables. This is a classical database concept available in Pandas.

`join()` operates on two tables: the first one is the "left" table which uses `join()` as a method. The other table is the "right" one.

Let's try the default join settings:

```
In [12]: composers1
```

```
Out[12]:
```

	birth	death	city
composer			
Mahler	1860	1911	Kaliste
Beethoven	1770	1827	Bonn
Puccini	1858	1924	Lucques
Shostakovich	1906	1975	Saint-Petersburg

```
In [13]: composers2
```

```
Out[13]:
```

	first name
composer	
Mahler	Gustav
Beethoven	Ludwig van
Puccini	Giacomo
Brahms	Johannes

```
In [14]: composers1.join(composers2)
```

```
Out[14]:
```

	birth	death	city	first name
composer				
Mahler	1860	1911	Kaliste	Gustav
Beethoven	1770	1827	Bonn	Ludwig van
Puccini	1858	1924	Lucques	Giacomo
Shostakovich	1906	1975	Saint-Petersburg	NaN

We see that Pandas was smart enough to notice that the two tables had a index name and used it to combine the tables. We also see that one element from the second table (Brahms) is missing. The reason for this is the way indices not present in both tables are handled. There are four ways of doing this with two tables called here the "left" and "right" table.

10.2.1. Join left

Here "left" and "right" just represent two Dataframes that should be merged. They have a common index, but not necessarily the same items. For example here Shostakovich is missing in the second table, while Brahms is missing in the first one. When using the "right" join, we use the first Dataframe as basis and only use the indices that appear there.

```
In [15]: composers1.join(composers2, how = 'left')
```

```
Out[15]:
```

	birth	death	city	first name
composer				
Mahler	1860	1911	Kaliste	Gustav
Beethoven	1770	1827	Bonn	Ludwig van
Puccini	1858	1924	Lucques	Giacomo
Shostakovich	1906	1975	Saint-Petersburg	NaN

Hence Brahms is left out.

10.2.2. Join right

We can do the the opposite and use the indices of the second Dataframe as basis:

```
In [16]: composers1.join(composers2, how = 'right')
```

```
Out[16]:
```

	birth	death	city	first name
composer				
Mahler	1860.0	1911.0	Kaliste	Gustav
Beethoven	1770.0	1827.0	Bonn	Ludwig van
Puccini	1858.0	1924.0	Lucques	Giacomo
Brahms	NaN	NaN	NaN	Johannes

Here we have Brahms but not Shostakovich.

10.2.3. Inner, outer

Finally, we can just say that we want to recover either only the items that appear in both Dataframes (inner, like in a Venn diagram) or all the items (outer).

```
In [17]: composers1.join(composers2, how = 'inner')
```

```
Out[17]:
```

	birth	death	city	first name
composer				
Mahler	1860	1911	Kaliste	Gustav
Beethoven	1770	1827	Bonn	Ludwig van
Puccini	1858	1924	Lucques	Giacomo

```
In [18]: composers1.join(composers2, how = 'outer')
```

```
Out[18]:
```

	birth	death	city	first name
composer				
Beethoven	1770.0	1827.0	Bonn	Ludwig van
Brahms	NaN	NaN	NaN	Johannes
Mahler	1860.0	1911.0	Kaliste	Gustav
Puccini	1858.0	1924.0	Lucques	Giacomo
Shostakovich	1906.0	1975.0	Saint-Petersburg	NaN

10.3.4 Joining on columns : merge

Above we have used `join` to join based on indices. However sometimes tables don't have the same indices but similar contents that we want to merge. For example let's imagine we have the two Dataframes below:

```
In [19]: composers1 = pd.read_excel('Data/composers.xlsx', sheet_name='Sheet1')
composers2 = pd.read_excel('Data/composers.xlsx', sheet_name='Sheet6')
```

```
In [20]: composers1
```

```
Out[20]:
```

	composer	birth	death	city
0	Mahler	1860	1911	Kaliste
1	Beethoven	1770	1827	Bonn
2	Puccini	1858	1924	Lucques
3	Shostakovich	1906	1975	Saint-Petersburg

In [21]: composers2

Out[21]:

	last name	first name
0	Puccini	Giacomo
1	Beethoven	Ludwig van
2	Brahms	Johannes
3	Mahler	Gustav

The indices don't match and are not the composer name. In addition the columns containing the composer names have different labels. Here we can use `merge()` and specify which columns we want to use for merging, and what type of merging we need (inner, left etc.)

In [22]: `pd.merge(composers1, composers2, left_on='composer', right_on='last name')`

Out[22]:

	composer	birth	death	city	last name	first name
0	Mahler	1860	1911	Kaliste	Mahler	Gustav
1	Beethoven	1770	1827	Bonn	Beethoven	Ludwig van
2	Puccini	1858	1924	Lucques	Puccini	Giacomo

Again we can use another variety of join than the default inner:

In [23]: `pd.merge(composers1, composers2, left_on='composer', right_on='last name', how = 'outer')`

Out[23]:

	composer	birth	death	city	last name	first name
0	Mahler	1860.0	1911.0	Kaliste	Mahler	Gustav
1	Beethoven	1770.0	1827.0	Bonn	Beethoven	Ludwig van
2	Puccini	1858.0	1924.0	Lucques	Puccini	Giacomo
3	Shostakovich	1906.0	1975.0	Saint-Petersburg	NaN	NaN
4	NaN	NaN	NaN	NaN	Brahms	Johannes

In [24]: `pd.merge(composers1, composers2, left_on='composer', right_on='last name', how = 'right')`

Out[24]:

	composer	birth	death	city	last name	first name
0	Mahler	1860.0	1911.0	Kaliste	Mahler	Gustav
1	Beethoven	1770.0	1827.0	Bonn	Beethoven	Ludwig van
2	Puccini	1858.0	1924.0	Lucques	Puccini	Giacomo
3	NaN	NaN	NaN	NaN	Brahms	Johannes