Effective Programming Practices for Economists

Data management with pandas

Merging datasets

Janoś Gabler and Hans-Martin von Gaudecker

Motivation

- Often when you download data, it comes in several files
- While you might not like it, this is often because the data providers respected the normal forms!
- Or it comes from very different sources
- In this screencast we show you how to merge or concatenate DataFrames

Concatenating DataFrames vertically

```
>>> top
```

		continent	life_exp
country	year		
Cuba	2002	Americas	77.16
	2007	Americas	78.27

```
>>> bottom
```

		continent	life_exp
country	year		
Spain	2002	Europe	79.78
	2007	Europe	80.94

```
>>> pd.concat([top, bottom])
```

		continent	life_exp
country	year		
Cuba	2002	Americas	77.16
	2007	Americas	78.27
Spain	2002	Europe	79.78
	2007	Europe	80.94

- concat stacks DataFrames on top of each other
- aligned by columns
- index needs to be compatible
- list can have more than two elements

Concatenating DataFrames horizontally

```
>>> left
```

		continent	life_exp
country	**year**		
Cuba	**2002**	Americas	77.16
	2007	Americas	78.27
Spain	**2002**	Europe	79.78
	2007	Europe	80.94

```
>>> right
```

		gdp_per_cap	pop
country	**year**		
Cuba	**2002**	6340.65	11226999
	2007	8948.10	11416987
Spain	**2002**	24835.47	40152517
	2007	28821.06	40448191

```
>>> pd.concat([left, right], axis="columns")
```

		continent	life_exp	gdp_per_cap	pop
country	**year**				
Cuba	**2002**	Americas	77.16	6340.65	11226999
	2007	Americas	78.27	8948.10	11416987
Spain	**2002**	Europe	79.78	24835.47	40152517
	2007	Europe	80.94	28821.06	40448191

- with `` `axis="columns"` ``, DataFrames are stacked horizontally
- Used to be `` `axis=1` ``

Careful with non-meaningful indices

```
>>> left
```

	country	continent	year	life_exp
0	Cuba	Americas	2002	77.16
1	Cuba	Americas	2007	78.27
2	Spain	Europe	2002	79.78

```
>>> right
```

	country	year	gdp_per_cap	pop
0	Cuba	2007	8948.10	11416987
1	Spain	2002	24835.47	40152517
2	Spain	2007	28821.06	40448191

```
>>> pd.concat([left, right], axis="columns")
```

	country	continent	year	life_exp	country	year	gdp_per_cap	pop
0	Cuba	Americas	2002	77.16	Cuba	2007	8948.10	11416987
1	Cuba	Americas	2007	78.27	Spain	2002	24835.47	40152517
2	Spain	Europe	2002	79.78	Spain	2007	28821.06	40448191

1:1 merges

```
>>> left
```

	country	continent	year	life_exp
0	Cuba	Americas	2002	77.16
1	Cuba	Americas	2007	78.27
2	Spain	Europe	2002	79.78

```
>>> right
```

	country	year	gdp_per_cap	pop
0	Cuba	2007	8948.10	11416987
1	Spain	2002	24835.47	40152517
2	Spain	2007	28821.06	40448191

```
>>> pd.merge(left, right, on=["country", "year"])
```

	country	continent	year	life_exp	gdp_per_cap	pop
0	Cuba	Americas	2007	78.27	8948.10	11416987
1	Spain	Europe	2002	79.78	24835.47	40152517

- merge does not align on index by default
- can change using arguments `` `left_index=True` `` and `` `right_index=True` ``
- can also use `` `merge` `` method on DataFrame (becomes "left" frame)
- by default, it does an inner join

```
>>> pd.merge(left, right, on=["country", "year"], how="inner")
```

	country	continent	year	life_exp	gdp_per_cap	pop
0	Cuba	Americas	2007	78.27	8948.10	11416987
1	Spain	Europe	2002	79.78	24835.47	40152517

```
>>> pd.merge(left, right, on=["country", "year"], how="left")
```

	country	continent	year	life_exp	gdp_per_cap	pop
0	Cuba	Americas	2002	77.16	nan	nan
1	Cuba	Americas	2007	78.27	8948.10	11416987.00
2	Spain	Europe	2002	79.78	24835.47	40152517.00

```
>>> pd.merge(left, right, on=["country", "year"], how="outer")
```

	country	continent	year	life_exp	gdp_per_cap	pop
0	Cuba	Americas	2002	77.16	nan	nan
1	Cuba	Americas	2007	78.27	8948.10	11416987.00
2	Spain	Europe	2002	79.78	24835.47	40152517.00
3	Spain	nan	2007	nan	28821.06	40448191.00

- The `` `how` `` argument determines which rows are kept
- The default `` `"inner"` `` is not always a good choice

m:1 merges

```
>>> left
```

	country	year	life_exp
0	Cuba	2002	77.16
1	Cuba	2007	78.27
2	Spain	2002	79.78
3	Spain	2007	80.94

```
>>> right
```

	country	capital
0	Cuba	Havana
1	Spain	Madrid

```
>>> pd.merge(left, right, on="country")
```

	country	year	life_exp	capital
0	Cuba	2002	77.16	Havana
1	Cuba	2007	78.27	Havana
2	Spain	2002	79.78	Madrid
3	Spain	2007	80.94	Madrid

- The type of merge is determined by the data, not by calling a different function
- m:1 means that many entries in `` `left` `` are matched to one entry in `` `right` ``

Other merges

- There are also "1:m" and "m:m" merges
- Check the pandas tutorial for details

Concat vs. merge

- Use `` `concat` `` if it is safe to do
  - Index / columns are compatible
  - Only 1:1 merging
- Use `` `merge` ``
  - if you do anything outside of 1:1 merging
  - if you need more control

Check your data before and after

- Many people are afraid of merging
- This is because merges often go wrong
- Reason: badly prepared data
  - Want to do a 1:1 merge but merge key contains duplicates
  - Merge keys are not properly cleaned
  - ...
- Check your data before merging to avoid problems
- Check that you get the expected number of observations after merging