

Introduction to Data Science

Data Wrangling Continued - Multiple Data Frames

Joanna Bieri DATA101

Join or Merge

ASK

You have two data frames about the same people.

What do they need to have in common before you can put them together?

Join or Merge

ANSWER

- A shared column, the **key**. Here it is `name`.
- The values have to match exactly: `Ana` and `ana` are different people to pandas.
- `on=` tells pandas which column to match, `how=` tells it which rows to keep.

Left Join

PREDICT

students

	name	year
0	Ana	1
1	Ben	2
2	Cal	4

majors

	name	major
0	Ana	Math
1	Ben	Econ
2	Dee	Bio

```
pd.merge(students, majors, on='name', how='left')
```

How many rows come out? What goes in the empty spots?

Left Join

ANSWER

```
pd.merge(students, majors, on='name', how='left')
```

	name	year	major
0	Ana	1	Math
1	Ben	2	Econ
2	Cal	4	NaN

Every name from the left data frame (students) stays. Cal has no major, so pandas fills in NaN. Dee is dropped.

Right Join

PREDICT

students

	name	year
0	Ana	1
1	Ben	2
2	Cal	4

majors

	name	major
0	Ana	Math
1	Ben	Econ
2	Dee	Bio

```
pd.merge(students, majors, on='name', how='right')
```

Which name disappears this time?

Right Join

ANSWER

```
pd.merge(students, majors, on='name', how='right')
```

	name	year	major
0	Ana	1.0	Math
1	Ben	2.0	Econ
2	Dee	NaN	Bio

Now the right data frame (majors) sets the names, so Cal is gone.
Notice year turned into decimals: a column with a NaN in it becomes a float.

Outer Join

PREDICT

students

	name	year
0	Ana	1
1	Ben	2
2	Cal	4

majors

	name	major
0	Ana	Math
1	Ben	Econ
2	Dee	Bio

```
pd.merge(students, majors, on='name', how='outer')
```

How many rows? How many NaN?

Outer Join

ANSWER

```
pd.merge(students, majors, on='name', how='outer')
```

	name	year	major
0	Ana	1.0	Math
1	Ben	2.0	Econ
2	Cal	4.0	NaN
3	Dee	NaN	Bio

Four rows, two NaN. Outer (also called a full join) keeps every name from both sides.

Inner Join

REVERSE

students

	name	year
0	Ana	1
1	Ben	2
2	Cal	4

majors

	name	major
0	Ana	Math
1	Ben	Econ
2	Dee	Bio

want

	name	year	major
0	Ana	1	Math
1	Ben	2	Econ

What is the command?

Inner Join

ANSWER

```
pd.merge(students, majors, on='name', how='inner')
```

	name	year	major
0	Ana	1	Math
1	Ben	2	Econ

Inner keeps only the names that are in both. No NaN, so year stays whole numbers.

Join with an Indicator

PREDICT

students

	name	year
0	Ana	1
1	Ben	2
2	Cal	4

majors

	name	major
0	Ana	Math
1	Ben	Econ
2	Dee	Bio

```
pd.merge(students, majors, on='name',  
         how='outer', indicator=True)
```

What new column shows up, and what is in it?

Join with an Indicator

ANSWER

```
pd.merge(students, majors, on='name',  
         how='outer', indicator=True)
```

	name	year	major	_merge
0	Ana	1.0	Math	both
1	Ben	2.0	Econ	both
2	Cal	4.0	NaN	left_only
3	Dee	NaN	Bio	right_only

`_merge` tells you where each row came from.

Anti Join

REVERSE

students

	name	year
0	Ana	1
1	Ben	2
2	Cal	4

majors

	name	major
0	Ana	Math
1	Ben	Econ
2	Dee	Bio

want

	name	year	major	_merge
2	Cal	4.0	NaN	left_only
3	Dee	NaN	Bio	right_only

What is the command? (It takes two steps.)

Anti Join

ANSWER

```
DF = pd.merge(students, majors, on='name',  
              how='outer', indicator=True)  
mask = DF['_merge'] != 'both'  
DF[mask]
```

	name	year	major	_merge
2	Cal	4.0	NaN	left_only
3	Dee	NaN	Bio	right_only

Merge with the indicator, then mask. This is an **anti join**: the names that did NOT match. A great way to find problems in your data.

Grocery Sales

PREDICT

prices

	item	price
0	apple	0.50
1	bread	1.00
2	milk	0.80

purchases

	customer	item
0	1	milk
1	1	apple
2	2	milk

```
pd.merge(prices, purchases, on='item', how='left')
```

How many rows come out? Which row has a NaN?

Grocery Sales

ANSWER

```
pd.merge(prices, purchases, on='item', how='left')
```

	item	price	customer
0	apple	0.5	1.0
1	bread	1.0	NaN
2	milk	0.8	1.0
3	milk	0.8	2.0

4 rows. Milk shows up twice because it was bought twice. Bread was never bought, but it is still here, so if you add up the price column to get revenue the bread gets counted. Using `how='right'` keeps only what was bought.

Questions?

Questions?