

# **Data 301**

## **Conditioning and Slicing, Split-Applv-Combine, Reshaping Data**

Dennis Sun

April 7, 2016

1 Data Conditioning

2 Data Slicing

3 Split-Apply-Combine

4 Reshaping Data

# Data Conditioning

Let's continue looking at the Harris Bank data set.

```
import pandas as pd
data = pd.read_csv("/data/harris.csv")
```

Bsal	Sal77	Sex	Senior	Age	Educ	Exper
5220	8340	Female	70	468	12	127
5040	12420	Male	96	329	15	14
6300	12060	Male	82	357	15	72
6000	15120	Male	67	315	15	35.5
⋮						⋮

Last class, we asked questions like “What was the average beginning salary for male employees?”

```
data.ix[data['Sex'] == 'Male', 'Bsal'].mean()
```

In other words, we **conditioned** on `Sex = 'Male'`. The term *conditioning* comes from conditional probability.

# Data Conditioning

Conditioning is easy. We simply restrict our analysis to a subset of rows that pass some filter.

## In-Class Exercise

*Last class, we said that the fact that females earned a lower salary than males is not itself conclusive evidence of discrimination because females and males may differ in their education, experience, etc.*

*Let's compare the average beginning salary of men and women with **at least** 15 years of education.*

1 Data Conditioning

2 Data Slicing

3 Split-Apply-Combine

4 Reshaping Data

# Data Slicing

To **slice** by a variable simply means to repeat an analysis, conditioning on all possible values of that variable.

So when we computed confidence intervals for the average beginning salary for men and women, we were *slicing* our analysis by the variable **Sex**.

To slice by a variable in Pandas, you use the `.groupby()` method.

**Try it:** `data.groupby("Sex").mean()`

How would you extract the average beginning salary?

- `data.groupby("Sex")["Bsal"].mean()`
- `data.groupby("Sex").mean()["Bsal"]`

## Slice by Multiple Variables

You can also slice by multiple variables. This will repeat the analysis for all combinations of values for those variables.

**Try it:** `data_summary = data.groupby(["Sex", "Educ"]).mean()`

This is an example of a data frame with a **hierarchical index**. (Notice that the row indexes are a combination of sex and education.) Can you figure out how to access a row that is hierarchically indexed?

Use tuples! For example, `data_summary.ix[("Female", 12), :]`

## Custom Aggregation Functions

Pandas DataFrames have a few nice built-in aggregation methods. We've already seen `.mean()`, but there are also:

```
.sum() .max() .std() .count() .median()
```

...the list goes on. Use tab completion to get a complete list.

What if we wanted to use a custom aggregation?

Pandas provides `.agg()`, which allows you to specify an aggregation function that operates on a Series.

For example, if we wanted a function that calculates the range on each slice, we might start by defining

```
def range(x):  
    return max(x) - min(x)
```

and then pass this function into `.agg()`:

```
data.groupby(["Sex", "Educ"]).agg(range)
```

You can also pass in a list of functions you want to calculate:

```
data.groupby(["Sex", "Educ"]).agg([np.mean, np.std, range])
```



# Custom Aggregation Functions

## In-Class Exercise

*Write a function that take a Series as input and computes the lower bound of a 95% confidence interval for the mean. Do the same for the upper bound. Then, pass them into `.agg()` to get confidence intervals for the average beginning salary, sliced by **Sex** and **Educ**.*

# Method Chaining

In the exercise just now, you probably wrote code like:

```
data.groupby(["Sex", "Educ"]).agg([lower, upper])["Bsal"]
```

Ugh! This makes for terrible reading. The selection of **Bsal** just seems incongruous after you read `.groupby()` and `.agg()`.

Wouldn't it be nicer if we could chain the methods, one after the other?

```
data.groupby(["Sex", "Educ"]).agg([lower, upper]).get("Bsal")
```

Much nicer! This is possible only because Pandas methods always return a DataFrame or Series, so you can apply one method immediately to the output of another method.

1 Data Conditioning

2 Data Slicing

3 Split-Apply-Combine

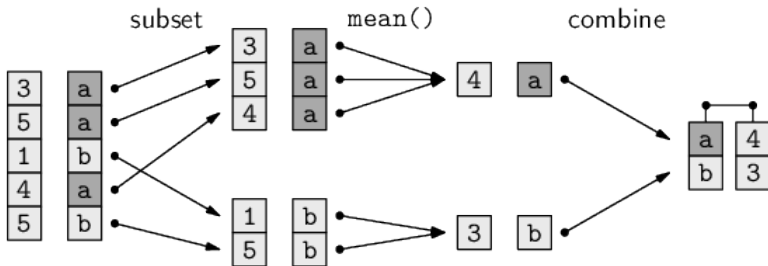
4 Reshaping Data

# Split-Apply-Combine

`.groupby()` + `.agg()` is an example of a strategy we'll see over and over called **split-apply-combine**.

- **Split** the data set into smaller data sets. (We first sliced our data by some variables.)
- **Apply** an analysis to each of the smaller data sets. (We calculated confidence intervals for each slice.)
- **Combine** the outputs together. (We put the confidence intervals into one data frame.)

# Split-Apply-Combine



1 Data Conditioning

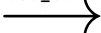
2 Data Slicing

3 Split-Apply-Combine

4 Reshaping Data

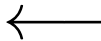
# Reshaping Data

`data_summary = data.groupby(["Sex", "Educ"]).mean().reset_index()`  
gives us something like

`.pivot_table()`  


	Sex	Educ	Bsal	Sal77	Senior	Age	Exper
0	Female	8	4974.55	9163.64	80.64	564.09	121.16
1	Female	10	4020.00	9840.00	92.00	528.00	44.00
2	Female	12	5119.71	9673.71	80.80	471.91	88.90
3	Female	15	5470.00	10355.00	83.08	518.50	105.29
4	Female	16	4950.00	9630.00	85.50	539.50	168.50
5	Male	8	6000.00	8940.00	78.00	659.00	320.00
6	Male	12	5652.86	11130.00	85.57	435.79	110.96
7	Male	15	6092.00	12044.00	82.60	402.47	88.27
8	Male	16	7050.00	14190.00	81.00	385.50	50.00

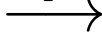
	Bsal		Sal77		Senior	
Sex	Female	Male	Female	Male	Female	Male
Educ						
8	4974.55	6000.00	9163.64	8940.0	80.64	78
10	4020.00	NaN	9840.00	NaN	92.00	Na
12	5119.71	5652.86	9673.71	11130.0	80.80	85
15	5470.00	6092.00	10355.00	12044.0	83.08	82
16	4950.00	7050.00	9630.00	14190.0	85.50	81

  
`pd.melt`

For ease of comparison, it would be nice to have the female and male entries side by side.

# Examples

`.pivot_table()`



	Sex	Educ	Bsal	Sal77	Senior	Age	Exper
0	Female	8	4974.55	9163.64	80.64	564.09	121.16
1	Female	10	4020.00	9840.00	92.00	528.00	44.00
2	Female	12	5119.71	9673.71	80.80	471.91	88.90
3	Female	15	5470.00	10355.00	83.08	518.50	105.29
4	Female	16	4950.00	9630.00	85.50	539.50	168.50
5	Male	8	6000.00	8940.00	78.00	659.00	320.00
6	Male	12	5652.86	11130.00	85.57	435.79	110.96
7	Male	15	6092.00	12044.00	82.60	402.47	88.27
8	Male	16	7050.00	14190.00	81.00	385.50	50.00

	Bsal		Sal77		Senior	
Sex	Female	Male	Female	Male	Female	Male
Educ						
8	4974.55	6000.00	9163.64	8940.0	80.64	78.00
10	4020.00	NaN	9840.00	NaN	92.00	NaN
12	5119.71	5652.86	9673.71	11130.0	80.80	85.57
15	5470.00	6092.00	10355.00	12044.0	83.08	82.60
16	4950.00	7050.00	9630.00	14190.0	85.50	81.00



`pd.melt`

```
data_pivoted = data_summary.pivot_table(index="Educ", columns="Sex")
```