Introduction to Data Science

Data Science Essentials

NASHVILLE SOFTWARE SCHOOL

Goals for today

- Review last session coding tasks
- Learn about some common data wrangling approaches
- Learn ways to find help (module API, Stack Overflow, etc)
- Intro to markdown

Review: Week 1 Coding Tasks

week1_review notebook

Resources for help when you get stuck

- Google
- Stack Overflow
- Doc Strings

Google

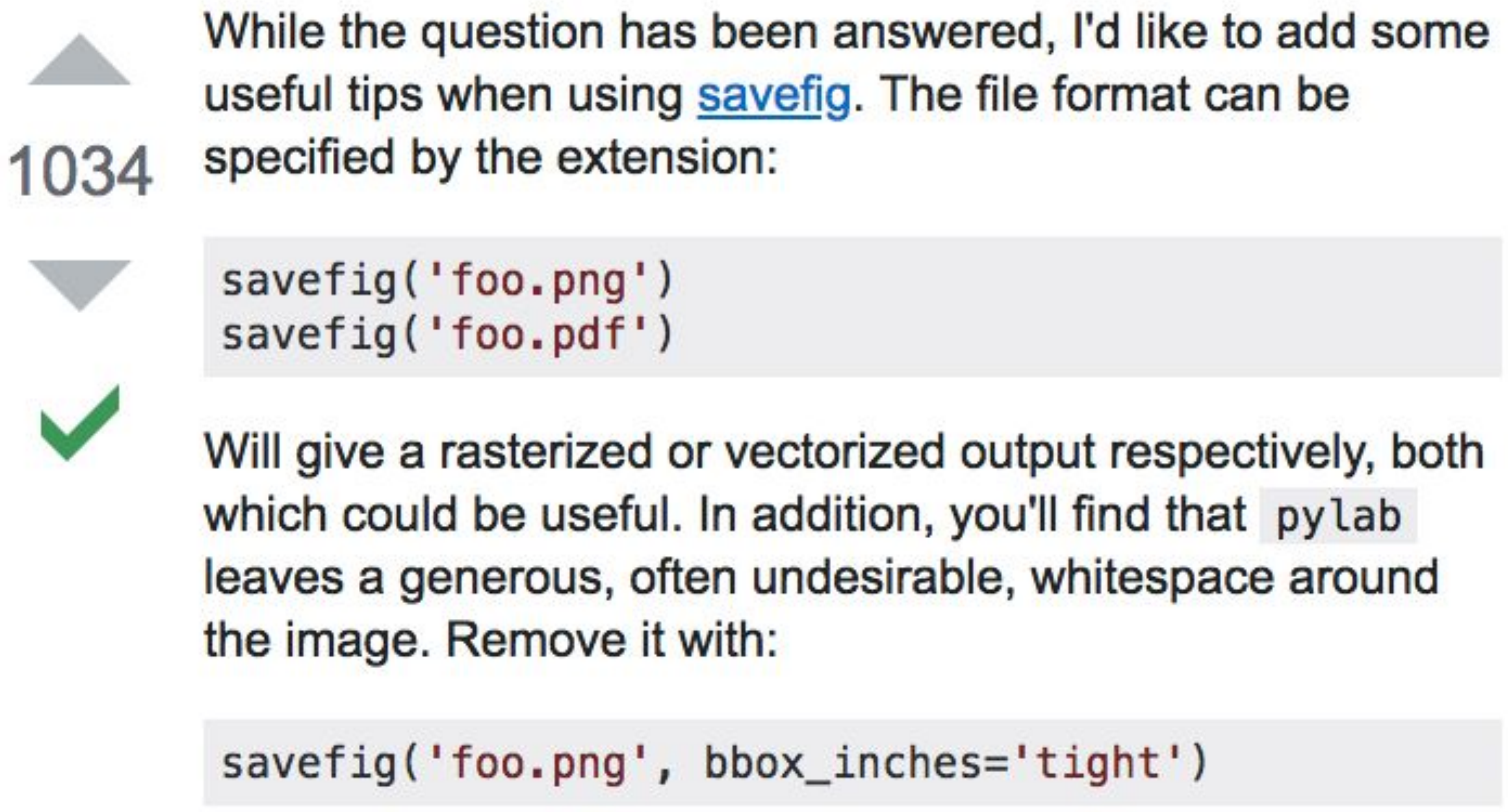
- Be as specific as you can: search for python + package + what you are trying to do.
- Copy the error from Jupyter and paste it right in the search box
- Pay attention to the dates of results - sometimes blog posts, etc. are outdated
- If you're not sure what text to use try asking your question exactly like you would ask another person!

NASHVILLE SOFTWARE SCHOOL

stack overflow

- Many times your google search will lead you here
- The question is at the top. Remember this is someone’s question and not the answer! Skim the question to ascertain that the issue is similar to yours.
- Scroll through the answers looking for:
  - A green check – this means the original poster accepted this as the best solution.
  - The largest number – this means the most people agreed this is the best solution. Sometimes the largest number is next to the question. This just means a lot of people had the same question!

NASHVILLE SOFTWARE SCHOOL



While the question has been answered, I'd like to add some useful tips when using savefig. The file format can be specified by the extension:

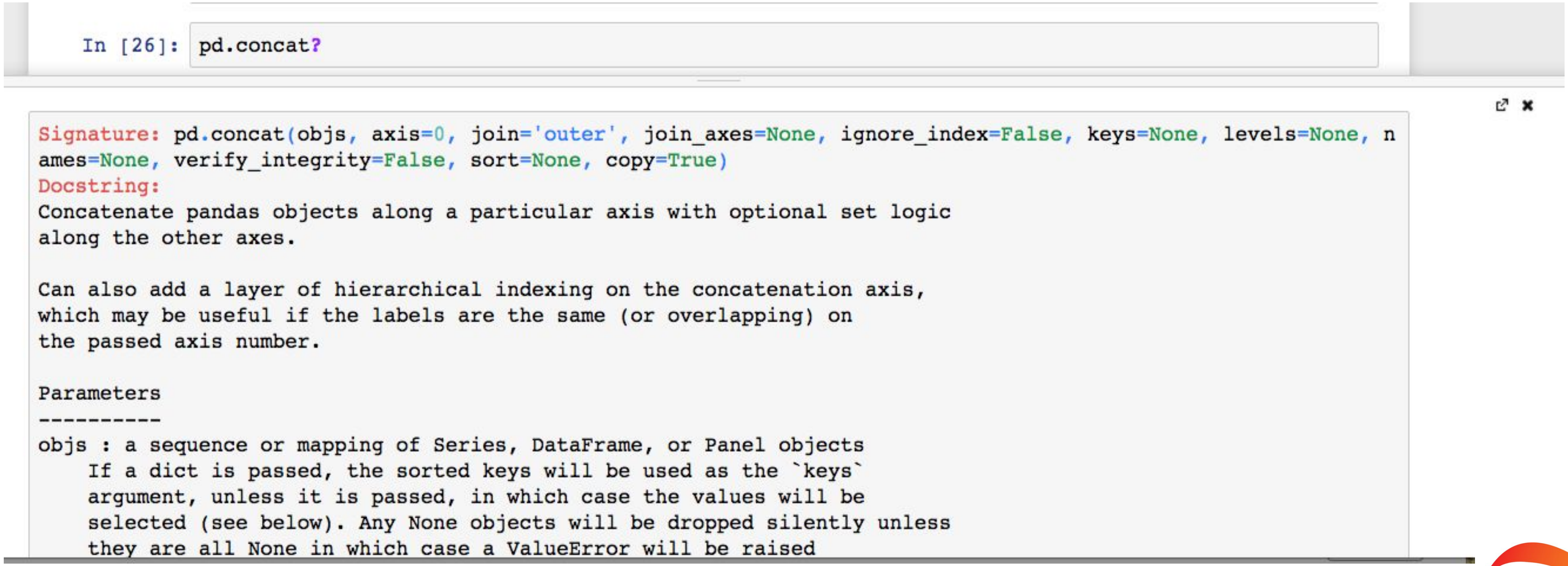
```
savefig('foo.png')
savefig('foo.pdf')
```

Will give a rasterized or vectorized output respectively, both which could be useful. In addition, you'll find that `pylab` leaves a generous, often undesirable, whitespace around the image. Remove it with:

```
savefig('foo.png', bbox_inches='tight')
```

Help within Jupyter

- shift + tab after keyword in a Jupyter cell
- ? + keyword in a Jupyter cell



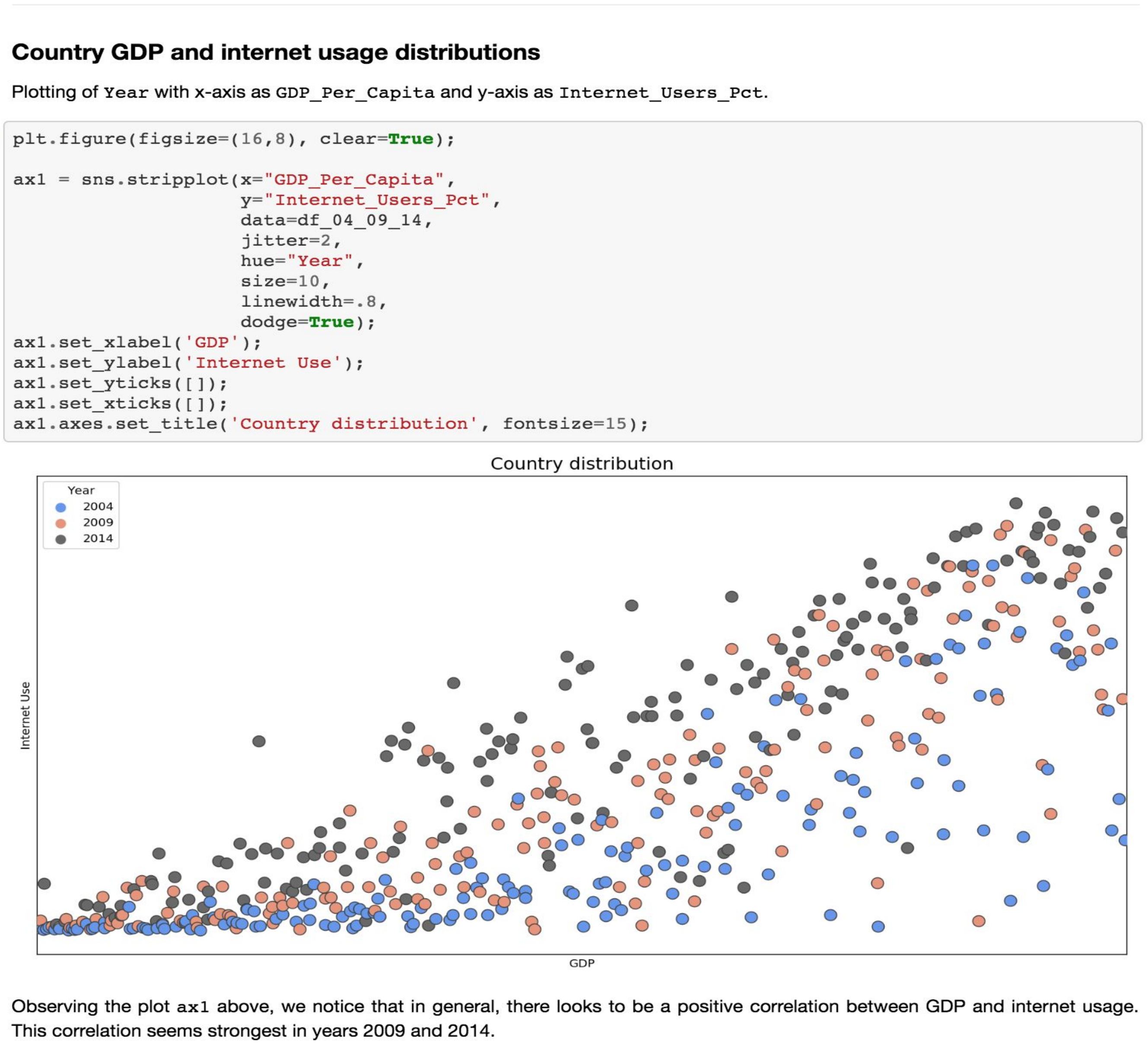
```
In [26]: pd.concat?
```

```
Signature: pd.concat(objs, axis=0, join='outer', join_axes=None, ignore_index=False, keys=None, levels=None, n
ames=None, verify_integrity=False, sort=None, copy=True)
Docstring:
Concatenate pandas objects along a particular axis with optional set logic
along the other axes.

Can also add a layer of hierarchical indexing on the concatenation axis,
which may be useful if the labels are the same (or overlapping) on
the passed axis number.

Parameters
----------
objs : a sequence or mapping of Series, DataFrame, or Panel objects
    If a dict is passed, the sorted keys will be used as the `keys`
    argument, unless it is passed, in which case the values will be
    selected (see below). Any None objects will be dropped silently unless
    they are all None in which case a ValueError will be raised
```

Markdown cells are a useful way to annotate your work:

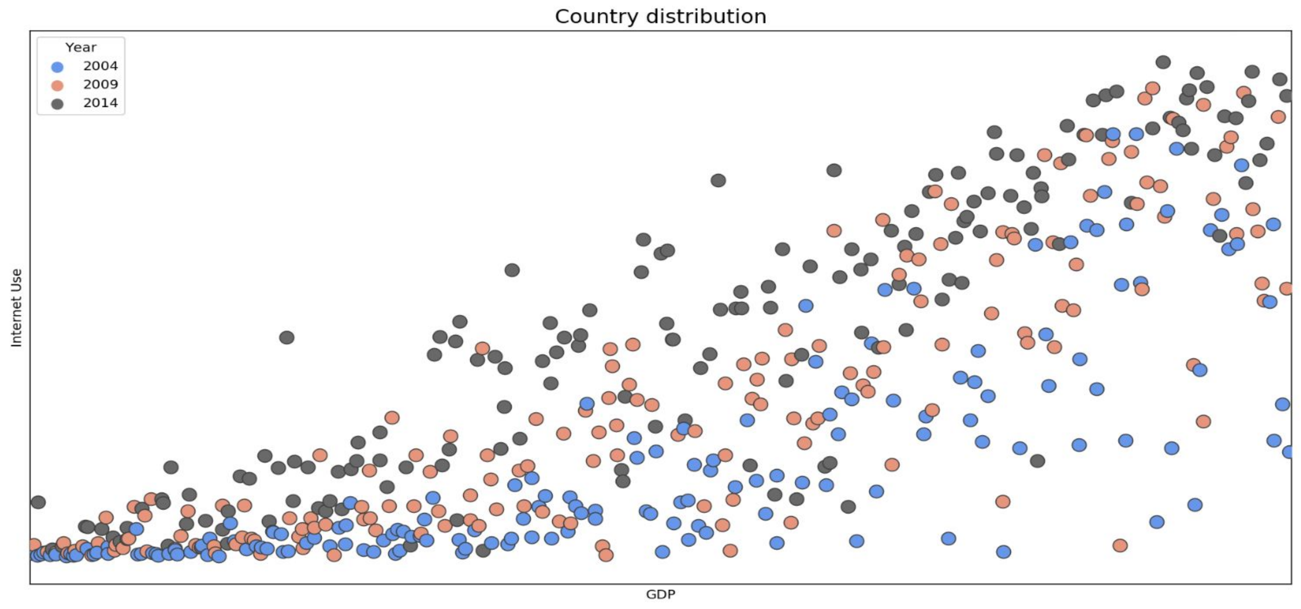


Country GDP and internet usage distributions

Plotting of `Year` with x-axis as `GDP_Per_Capita` and y-axis as `Internet_Users_Pct`.

```
In [8]: plt.figure(figsize=(16,8), clear=True);

        ax1 = sns.stripplot(x="GDP_Per_Capita",
                            y="Internet_Users_Pct",
                            data=df_04_09_14,
                            jitter=2,
                            hue="Year",
                            size=10,
                            linewidth=.8,
                            dodge=True);
        ax1.set_xlabel('GDP');
        ax1.set_ylabel('Internet Use');
        ax1.set_yticks([]);
        ax1.set_xticks([]);
        ax1.axes.set_title('Country distribution', fontsize=15);
```



Observing the plot `ax1` above, we notice that in general, there looks to be a positive correlation between GDP and internet usage. This correlation seems strongest in years 2009 and 2014.

- Comment on choices made
- Comment on trends observed
- Note anomalies/surprises

https://www.markdownguide.org/cheat-sheet/

pandas – https://pandas.pydata.org/pandas-docs/stable/api.html

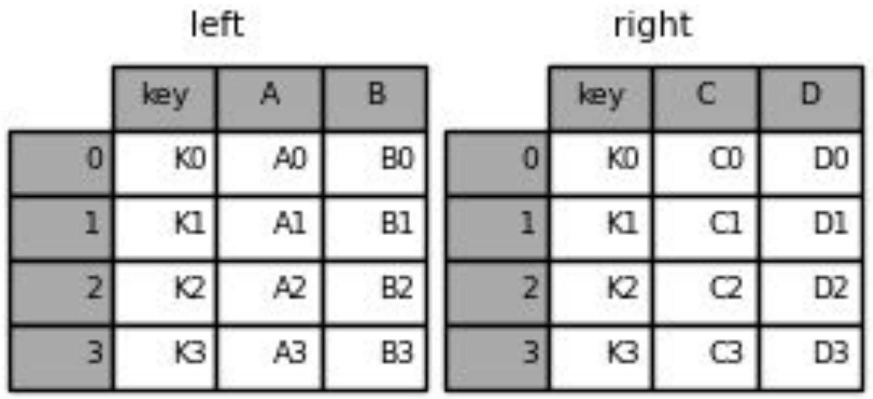
- **String Methods:**
  - **.str.lower()** - convert a column to lowercase
  - **.str.upper()** - convert a column to uppercase
  - **.str.split()** - divide a string column into a list by specifying a delimiter character
  - **.str.replace()** - replace each instance of a string with a different string

- **df.describe() and series.describe()** – returns statistical info (count, mean, sd, quartiles)

- **pd.merge()** - Combines two DataFrames by joining along one or more columns

Merging two DataFrames:

pd.merge(<df1>, <df2>, on = <col or list of cols to join on>, how = <join_type>*)*



left

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

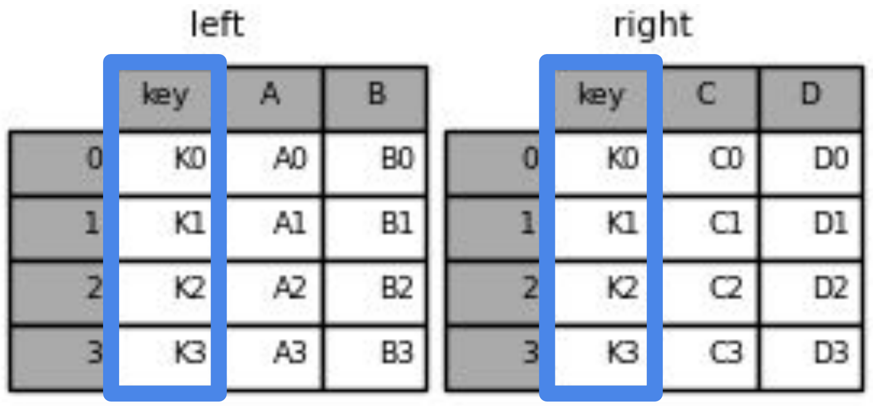
right

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

- Need one or more “key” columns to join on
- Pastes matching rows together along the key column(s)

Merging two DataFrames:

pd.merge(<df1>, <df2>, on = <col or list of cols to join on>, how = <join_type>*)*



left

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

right

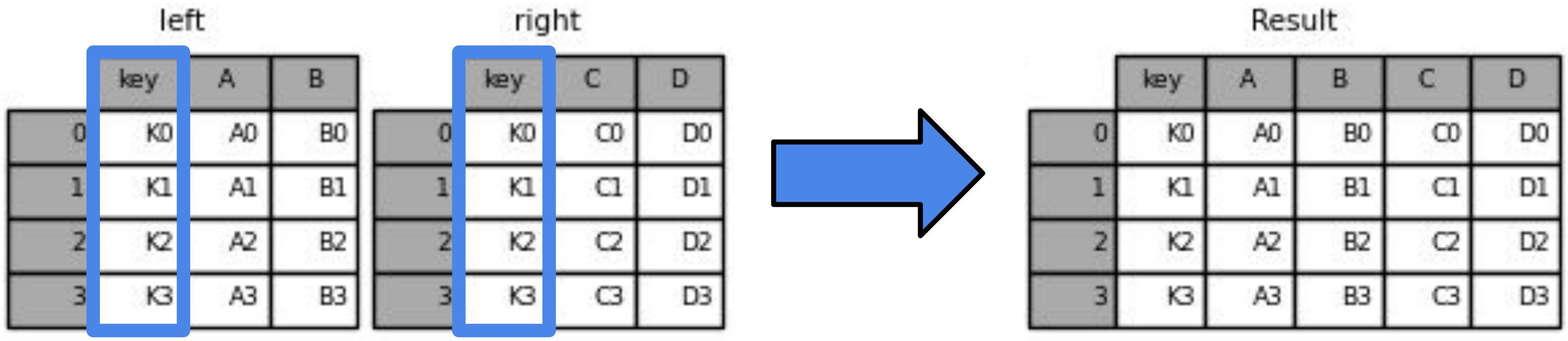
	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

- Need one or more “key” columns to join on
- Pastes matching rows together along the key column(s)

NASHVILLE SOFTWARE SCHOOL

Merging two DataFrames:

pd.merge(<df1>, <df2>, on = <col or list of cols to join on>, how = <join_type>)



left

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

right

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

Result

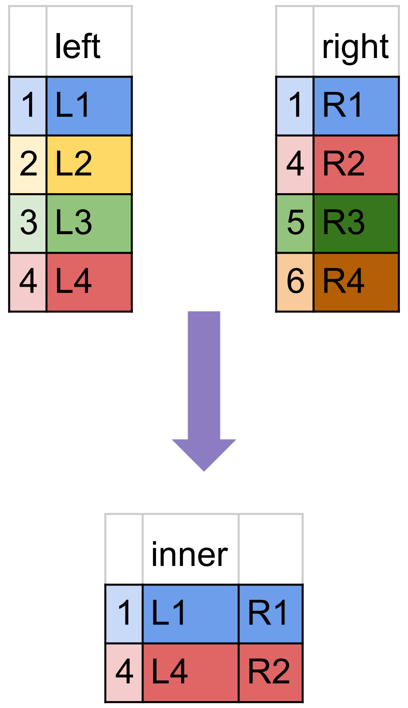
	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

- **Need one or more “key” columns to join on**
- **Pastes matching rows together along the key column(s)**

INNER JOIN

An **INNER JOIN** keeps **only the rows that have matching values in both tables**.

This is the default type of join when using *pd.merge()*.



	left
1	L1
2	L2
3	L3
4	L4

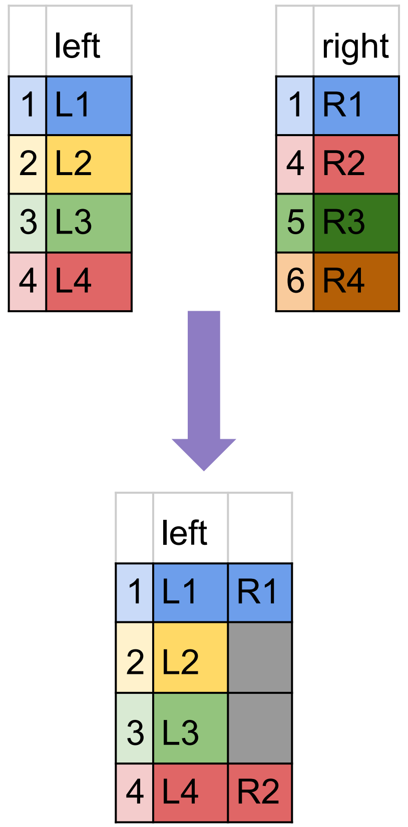
	right
1	R1
4	R2
5	R3
6	R4

	inner	
1	L1	R1
4	L4	R2

LEFT JOIN and RIGHT JOIN

A **LEFT JOIN** keeps all rows from the left table and all matching rows from the right table. A **RIGHT JOIN** works similarly, except all rows from the right table are kept.

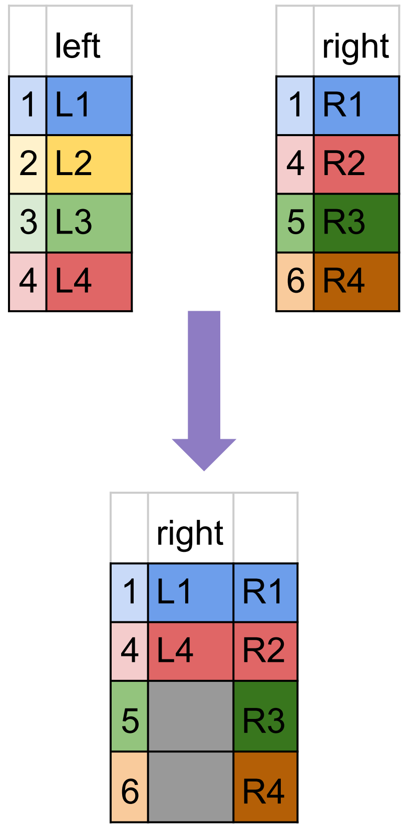
how = “left” or *how = “right”*



	left
1	L1
2	L2
3	L3
4	L4

	right
1	R1
4	R2
5	R3
6	R4

	left	
1	L1	R1
2	L2	
3	L3	
4	L4	R2



	left
1	L1
2	L2
3	L3
4	L4

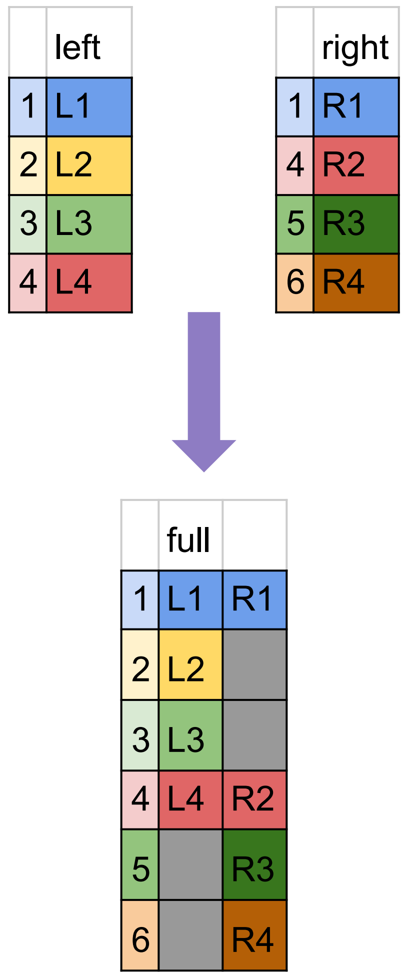
	right
1	R1
4	R2
5	R3
6	R4

	right	
1	L1	R1
4	L4	R2
5		R3
6		R4

OUTER/FULL JOIN

AN **OUTER JOIN** keeps all rows from both tables.

how = “outer”



	left
1	L1
2	L2
3	L3
4	L4

	right
1	R1
4	R2
5	R3
6	R4

	full	
1	L1	R1
2	L2	
3	L3	
4	L4	R2
5		R3
6		R4

Questions?

NASHVILLE SOFTWARE SCHOOL