Introduction to Data Science

Metro Health Department

NASHVILLE SOFTWARE SCHOOL

Goals for today

- **Finish notebook from last week**
- **Practice notebook to review the topics covered so far.**
- **Learn ways to find help (module API, Stack Overflow, etc)**
- **Intro to markdown**
- **Learn some more *pandas*, including plotting with *matplotlib***

Resources for help when you get stuck

- Google
- Stack Overflow
- Doc Strings

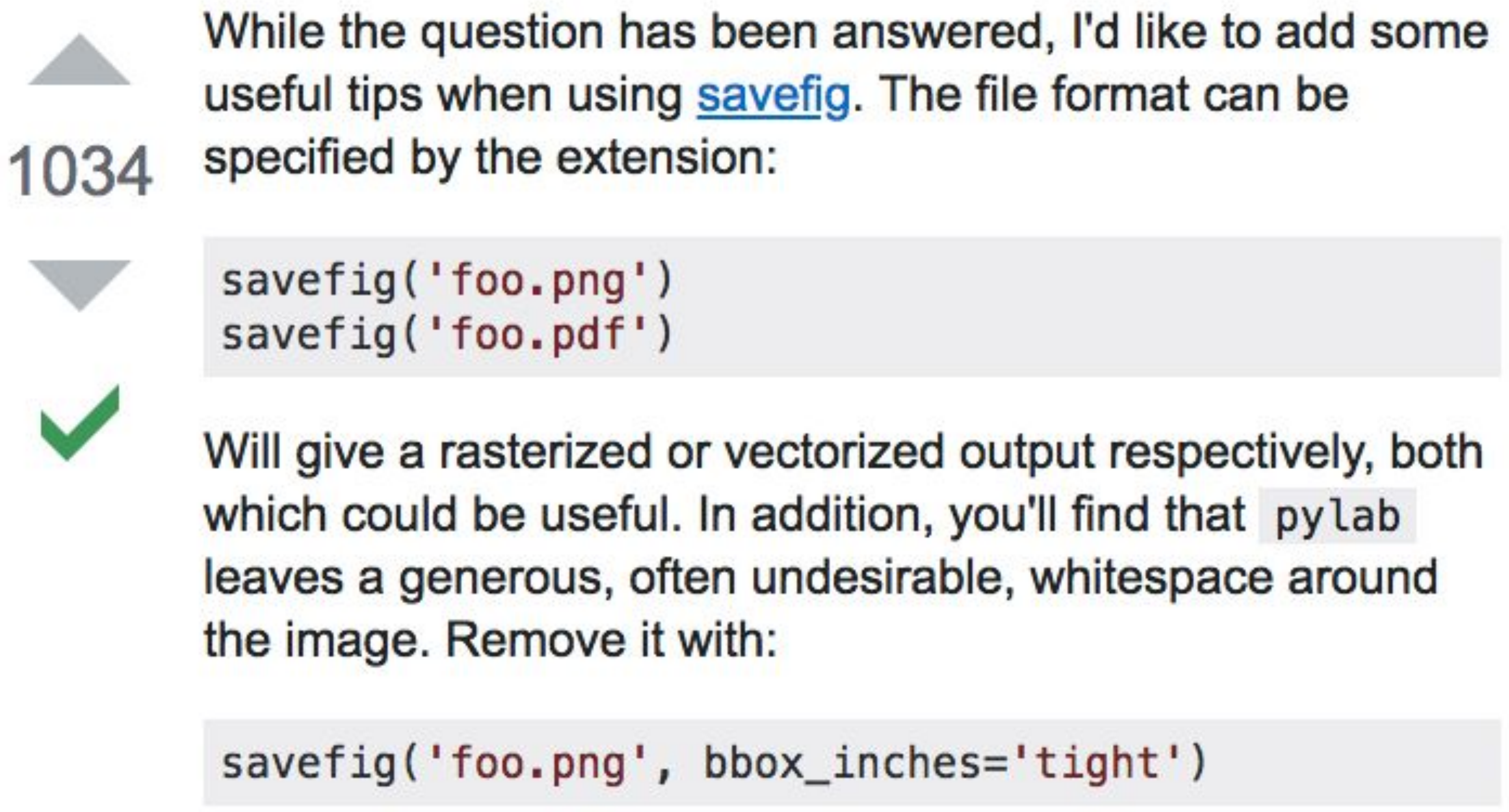
Google

- 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 are outdated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!

stackoverflow

- 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:

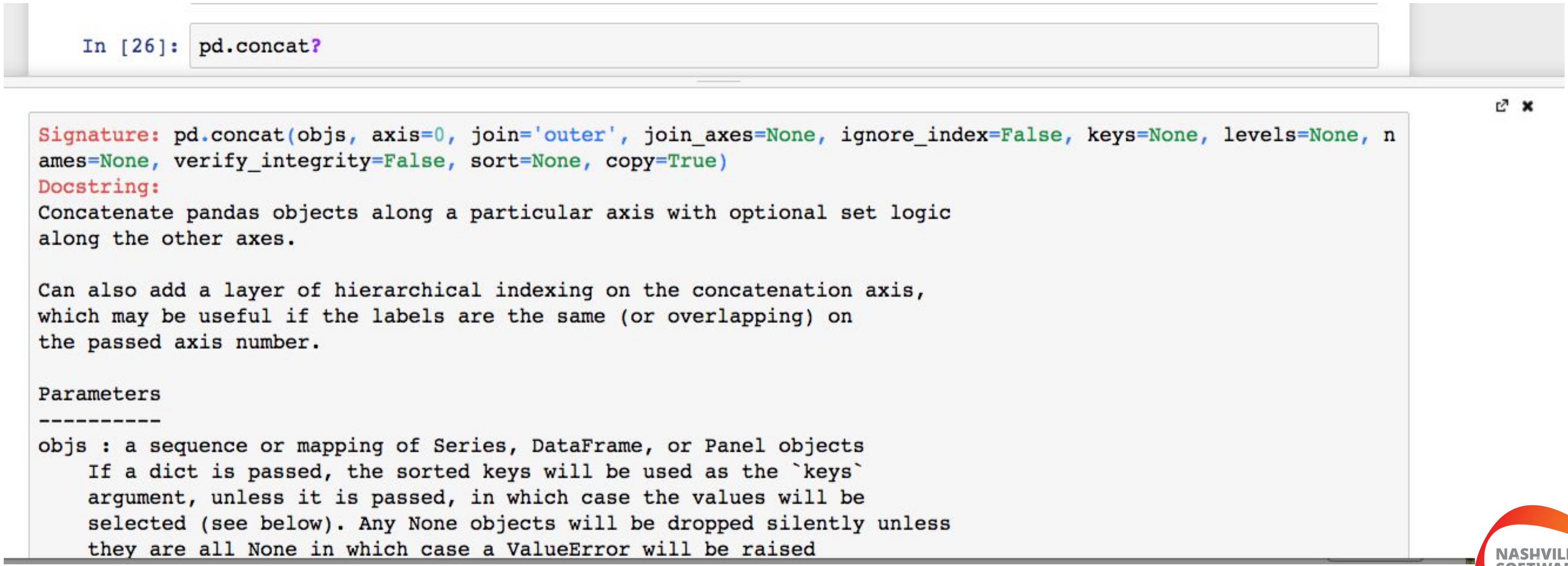
```
savefig('foo.png')
savefig('foo.pdf')
```

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 (do it four times to pin the doc string to the bottom of the page)
- ? + keyword in a Jupyter cell



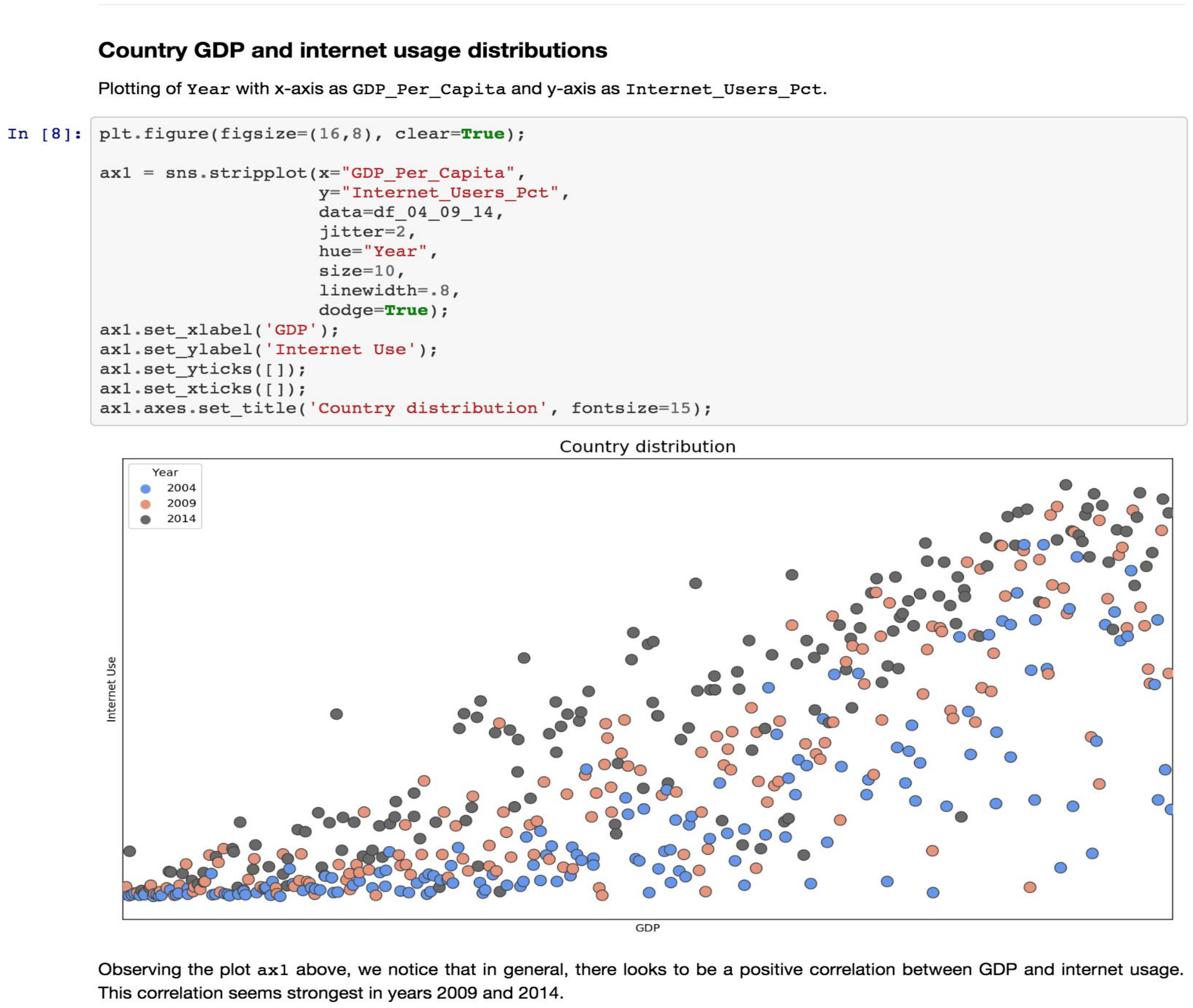
```
In [26]: pd.concat?
```

```
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

- **series.value_counts()** - gets counts for each unique value in a column (we saw this one last week)

- **df.groupby()** – group by a categorical variable to get summary statistics (count, sum, mean, etc.) by category

- **df.reset_index()** – resets the index to the default 0-based index; moves the current index to a column value unless **drop = True** is specified

- **Aggregate functions:**
  - **series.count(), series.nunique(), series.sum(), series.mean(), series.min(), series.max(), series.std()**

- **Functions to calculate values across rows:**
  - **series.shift(), series.diff(), series.cumsum()**

2_more_pandas_plus_plotting notebook

Questions?