David Backus <david.backus@gmail.com>

not urgent

Spencer Lyon <spencerlyon2@gmail.com> Sat, Sep 13, 2014 at 4:20 PM
To: David Backus <dbackus@stern.nyu.edu>

Hey Dave,

That is actually one of two commands I would have suggested to you.

The other one was

```
df = df.reset_index("year")  # or df.reset_index(1)
```

This will simply pop off the year part of the index and make it a column of your DataFrame. I think I like your solution better because it doesn't alter the index of the data frame (after I did that with your df the index was just "France" repeated a bunch of times — not very helpful). One solution that mixes the two pieces of functionally would be `df['year'] = df.get_level_index("year")`. Then it says as part of the index, but you have it as a column so it is ready to plot.

As far as the difference between Index and Series goes, for most things they will function equivalently (meaning most methods you call on a Series or functions you would pass a Series to would work just as well with an Index). If you want a more detailed answer about Index vs. Series let me know.

Finally, there are many ways to do the plotting code from section "3" of your file. Here is an alternative:

```
"""
3. Plot expenditure shares, saving and investment
"""
# this creates new series -- ok or better way?
df["Consumption"] = df['c2']/df['y']
df["Investment"] = df['gcf']/df['y']
df["Government"] = df['g']/df['y']
df["Net Exports"] = nxy = df['nx']/df['y']
df["Saving"] = (df['y']-df['c2']-df['g'])/df['y']


# Now, define a function that I can reuse.
def wb_plot(df, country, cols, title_str=None, save_str=None, **kwargs):
    """
    Quick function to plot specific columns for a particular country
    in the DataFrame obtained from the world bank

    Any additional keyword arguments are passed directly to the
    DataFrame.plot method

    Parameters
    ----------
    df : DataFrame
        The DataFrame that contains the data

    country : string
        A string specifying the name of the country. This must also be
        the entry on the outer level of the hierarchical index

    cols : list
        A list of column names to be plotted.
```

```
        title_str : optional(default=None)
            An optional title to be added to the plot. Note that you can
            leave a placeholder for country and it will automatically
            be filled for you with the country argument. An example of how
            this can be done is the following::

                title_str = "{country}: Expenditure Shares of GDP"

        save_str : optional (default=None)
            Similar to title_str, but provides the name the figure should be
            saved under. If none is given, then the figure is not saved
            automatically for you.

        Returns
        -------
        ax : matplotlib.pyplot.Axes
            The axes containing the plot

        """
        # return flat (non hierarchical-indexed) data frame single country
        new_df = df.loc[country]
        ax = new_df[cols].plot()
        ax.axhline(y=0, color="k", linestyle="-", linewidth=1)
        ax.legend(loc="best")

        if title_str is not None:
            ax.set_title(title_str.format(**{"country": country}))

        if save_str is not None:
            ax.get_figure().savefig(save_str.format(**{"country": country}))

        return ax

    # Now generate the first figure
    ax1 = wb_plot(df,
                  country="France",
                  cols=['Net Exports', 'Investment', 'Consumption', 'Government'],
                  title_str="{country}:  Expenditure Shares of GDP",
                  save_str="shares_sav_%s.pdf" % country_list[0])

    ax2 = wb_plot(df,
                  country="France",
                  cols=['Net Exports', 'Investment', 'Saving'],
                  title_str="{country}:  Saving and Investment",
                  save_str="shares_exp_%s.pdf" % country_list[0])
```

The great thing about this function is that if you wanted to you could pass in more than one country up at the top (in `country_list`), do all the data processing on all of them at a time (section 2 and start of section 3 where I make new columns), then just plot them one at a time using this function.

It might be overkill, but I like writing functions to do things if I repeat multiple steps more than once. Also, I really like to use the `plot` method on DataFrames because it constructs things like axes labels, tick marks, and legend labels automatically for me.

Let me know what you think

// Spencer

On September 13, 2014 at 12:17:45 PM, David Backus (dbackus@stern.nyu.edu) wrote:

> I declined your offer to explain how to extract a component of the

index of a df and now I find myself trying to figure it out on my own. Serves me right!

My example uses World Bank data, which seems to set up a df with a hierarchical index of country and year. I'm doing one country at a time, but it still has that structure. I need the year so I can plot series over time. I ran across this command, which seems to work: year = df.index.get_level_values(1)

Does that sound right? It's a little strange because year is some kind of index file (pandas.core.index.Index) and the other series are (after I whip them into shape) standalone series (pandas.core.series.Series), but it seems to work when I plot them.

Here's the code. Not urgent at all, but comments welcome if you think there's a better way to do this.
https://www.dropbox.com/s/wl7xgrpx75a79h0/wb_expenditure_shares.py?dl=0

Cheers.