Plot of pandas MultiIndex dataframe: Smart formatting of xlabels, ylabels, xticklabels and legend

I really like the pandas MultiIndex feature but find it hard to create nice looking plots with it.

All I want to do is to assign the xticklabels and legend entries to the right level values instead of showing the level names/tuples from the MultiIndex. Additionally, want to set the xlabels and ylabels manually.

Here's my first approach which already seems to work:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use("ggplot")

# dataframe with multi-index
arrays = [['foo', 'foo', 'bar', 'bar', 'bla', 'bla', 'asd', 'asd'],
          ['A', 'A', 'B', 'B', 'C', 'C', 'D', 'D'],
          [1, 2, 1, 2, 1, 2, 1, 2]]
tuples = list(zip(*arrays))
index = pd.MultiIndex.from_tuples(tuples, names=['first', 'second', 'third'])
df = pd.Series(np.random.randn(8), index=index)
df.sort_index(inplace=True)
print(df)

# subset to be plotted
idx = pd.IndexSlice
subset = df.loc[idx['foo', :, :], :]
subset = subset.unstack(level=2)
print(subset)

# plot
ax = subset.plot(kind='bar', colormap="Spectral", stacked=False, title="Title")
ax.set_xlabel("x label")
ax.set_ylabel("y label")
ax.set_xticks(range(0, len(subset.index.get_level_values(1)), 1), minor=False)
ax.set_xticklabels(
    [item for item in subset.index.get_level_values(1).tolist()],
    rotation=0, minor=False)
ax.legend(subset.columns.tolist(), loc='upper right')
```

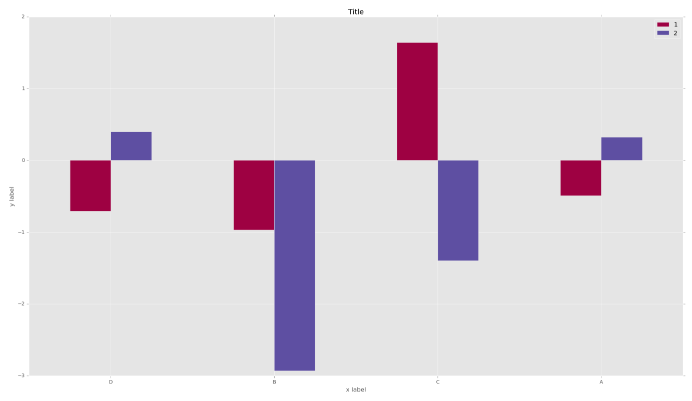
which returns for *df*:

```
first  second  third
asd    D       1        1.191413
               2        0.264764
bar    B       1       -0.611437
               2       -0.026589
bla    C       1        1.987642
               2        1.464153
foo    A       1        0.604582
               2       -0.268275
```

respectively for *subset*:

```
third                 1         2
first second
asd   D        1.191413  0.264764
bar   B       -0.611437 -0.026589
bla   C        1.987642  1.464153
foo   A        0.604582 -0.268275
```

and the following plot:



But this way (using the axes-object) seems quite heavy-handed to me. Is there any smart (generic) way to get the same result?

Thanks in advance!

python pandas

asked Jan 17 at 21:41

Cord Kaldemeyer
348 2 10