2023-05-15_ChartTypes_001_Bar

May 15, 2023

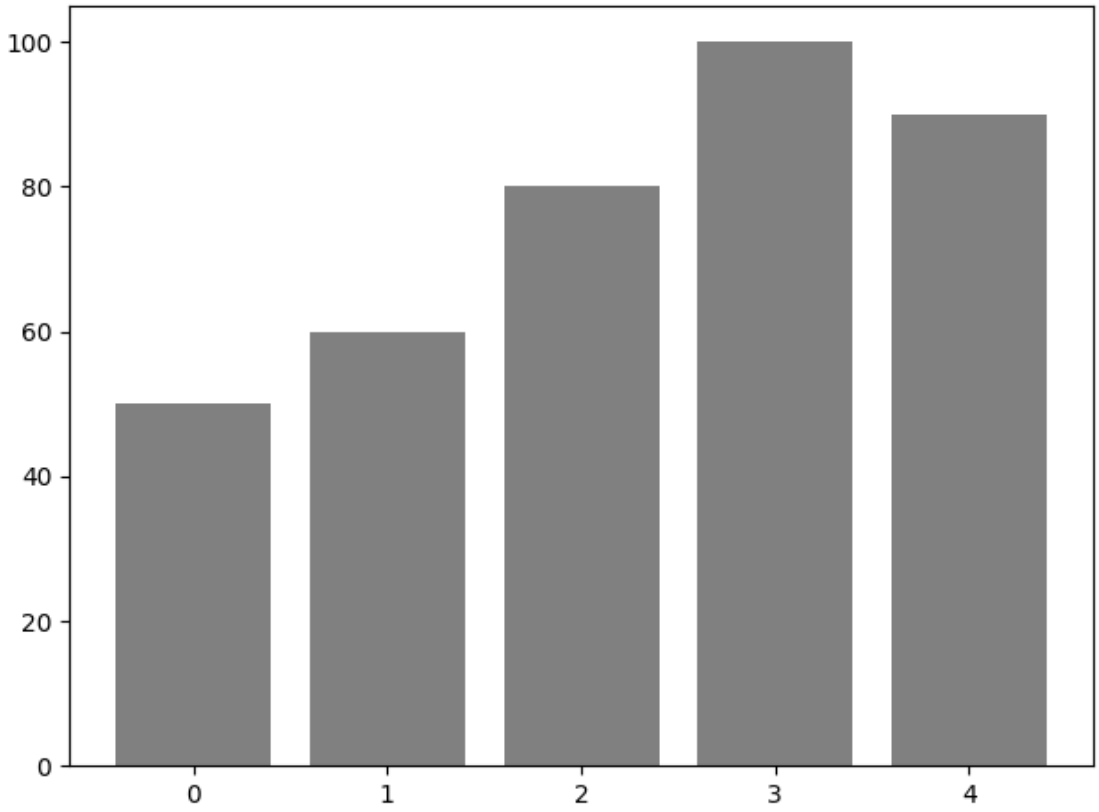
[1]:
```
import numpy as np
import scipy
import imageio

import matplotlib
import matplotlib.pyplot as plt
import matplotlib.cm as cm

matplotlib.rc('image', interpolation='nearest')
matplotlib.rc('figure',facecolor='white')
matplotlib.rc('image',cmap='viridis')
colors=plt.rcParams['axes.prop_cycle'].by_key()['color']
%matplotlib inline
```

1 Bar charts

1.1 Basic version

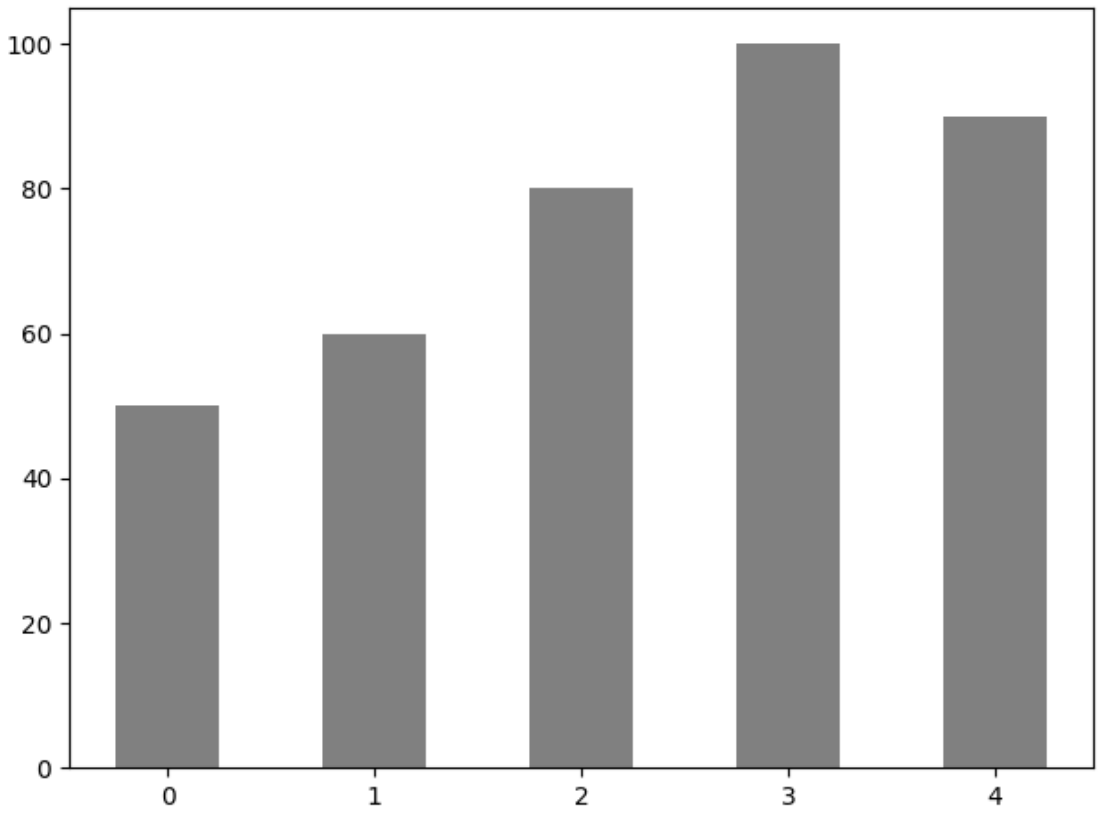
[2]:
```
# raw data: number per category
data=np.array([50,60,80,100,90])
data2=np.array([55,55,70,107,96])
n=len(data)
# horizontal position of each bar
x=np.arange(n)
```

[3]:
```
# basic bar chart
plt.bar(x,data,color="#808080")
plt.tight_layout()
plt.show()
```

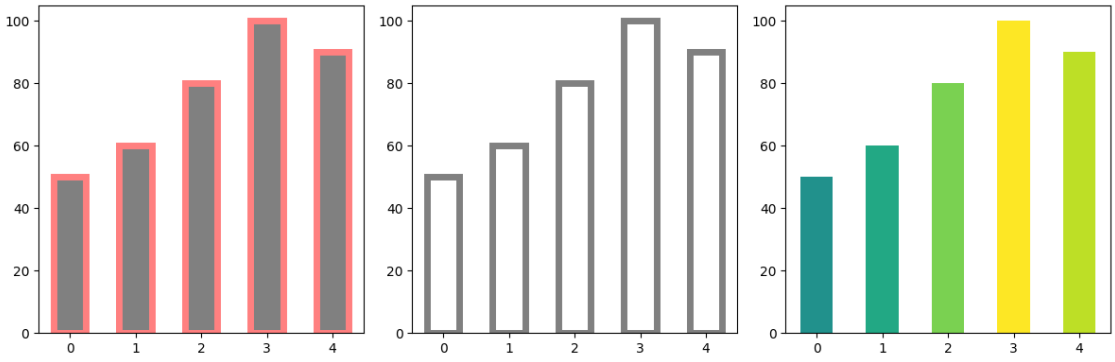


1.2 Adjusting appearance / additional degrees of freedom

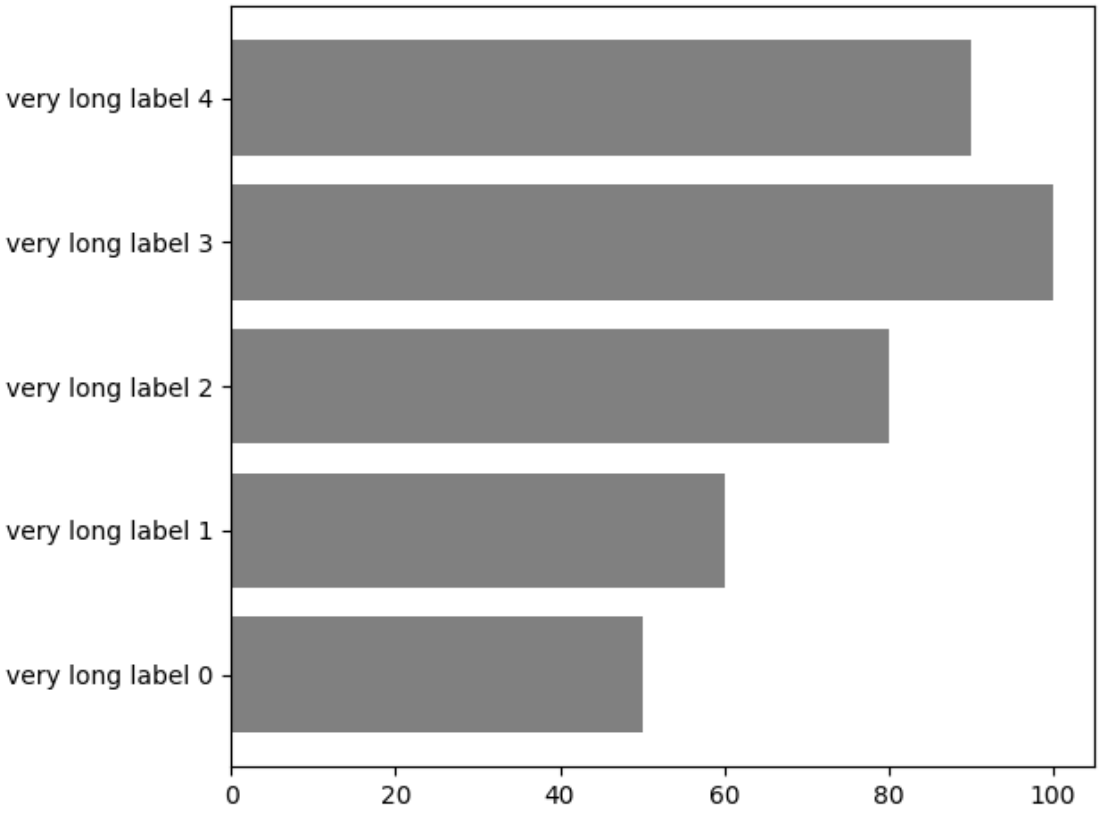
[4]:
```
# adjusting width of bars
# be careful with visual vibrations due to oscillating contrast beween bars!
width=.5
plt.bar(x,data,width=width,color="#808080")
plt.tight_layout()
plt.show()
```



[5]:
```
# draw fancy bars
# careful: this will almost always be visual ducks (in the sense of Tufte)
# use with caution and restraint

fig=plt.figure(figsize=(12,4))
# adding lines
fig.add_subplot(1,3,1)
plt.bar(x,data,width=width,color="#808080",lw=5,edgecolor="#ff8080")
# only lines
fig.add_subplot(1,3,2)
plt.bar(x,data,width=width,fill=False,lw=5,edgecolor="#808080")
# more colors
fig.add_subplot(1,3,3)
plt.bar(x,data,width=width,color=[cm.viridis(i/100) for i in data])

plt.tight_layout()
plt.show()
```

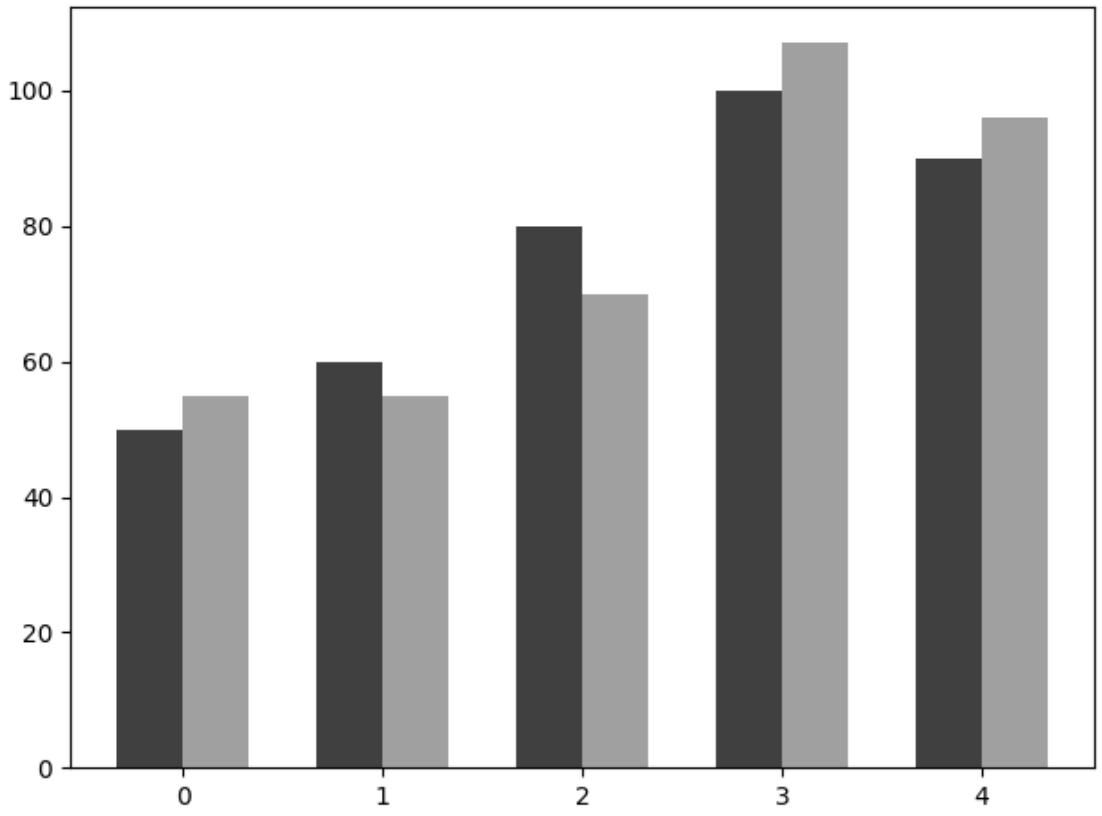


[6]:
```
# horizontal
# often useful for better labelling, may fit better on page
plt.barh(x,data,color="#808080")
labelList=["very long label "+str(i) for i in x]
plt.yticks(ticks=x,labels=labelList)
plt.tight_layout()
plt.show()
```



1.3 Grouped bar chart

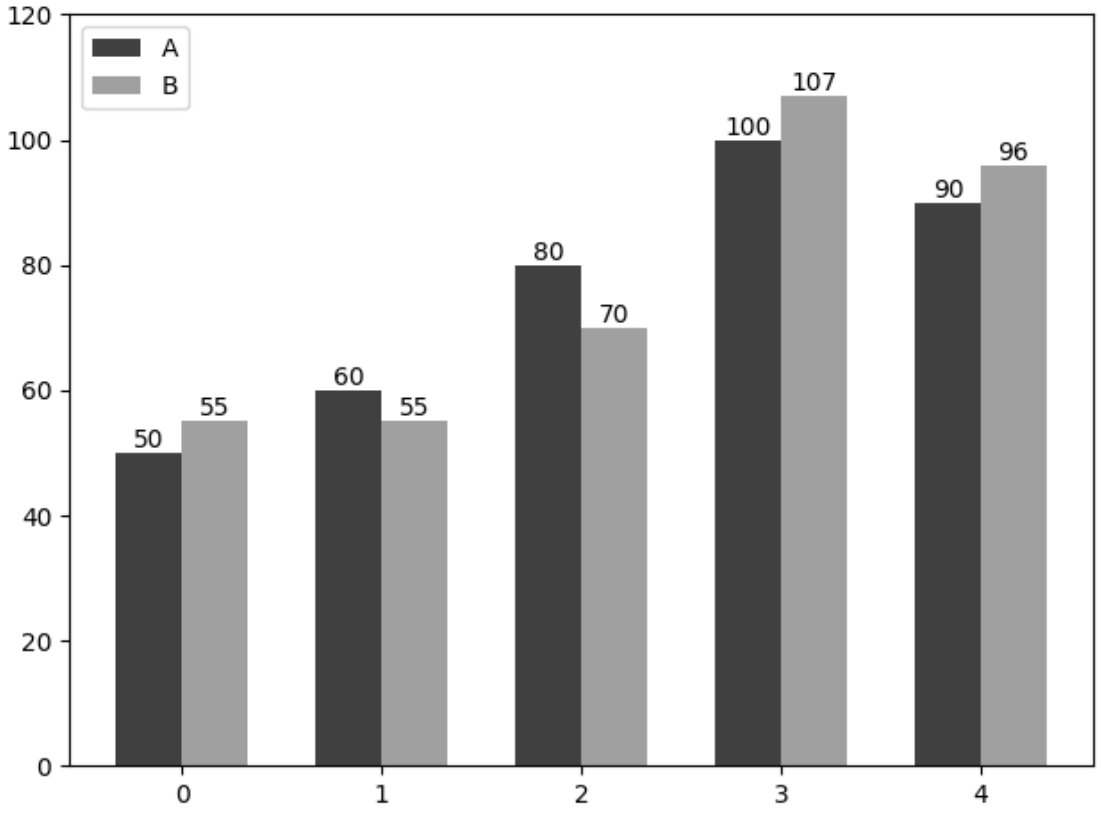
[7]:
```
# can easily be implemented with matplotlib tools
width=0.33
plt.bar(x-width/2,data,width=width,color="#404040")
plt.bar(x+width/2,data2,width=width,color="#A0A0A0")
plt.tight_layout()
plt.show()
```



1.4 Legends and annotations

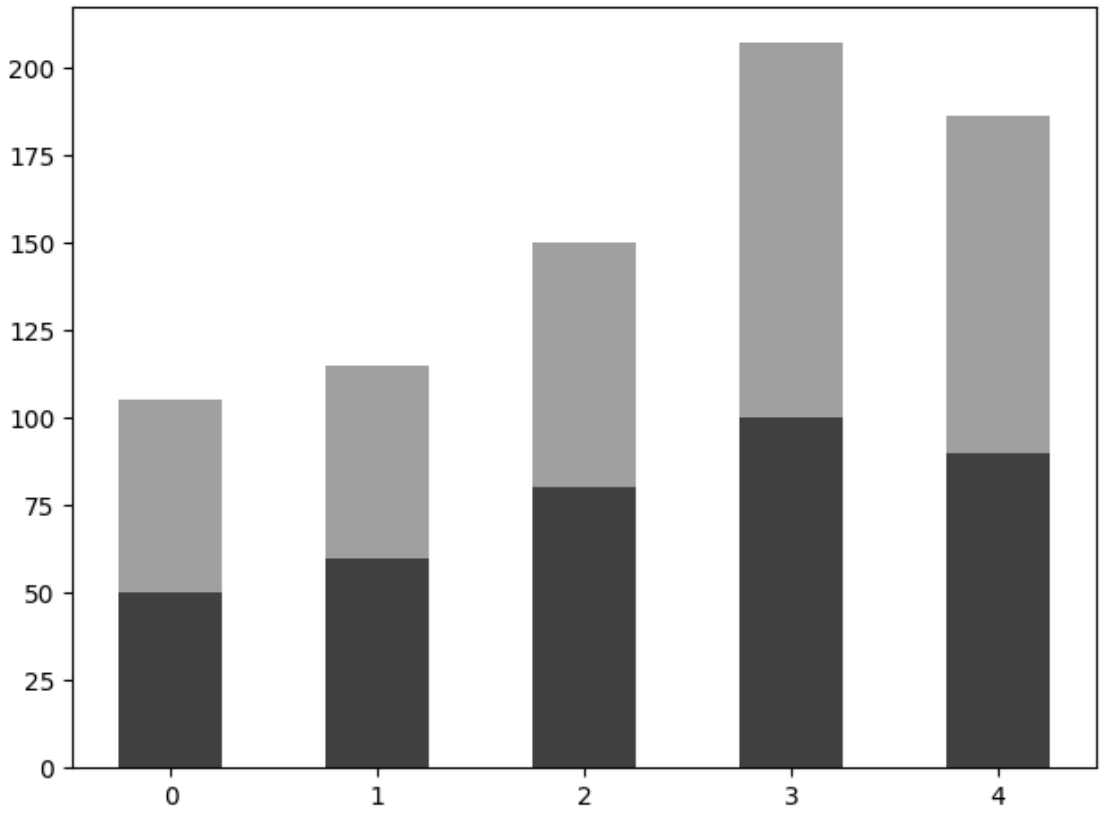
[8]:
```
# adding annotations
width=0.33
b1=plt.bar(x-width/2,data,width=width,label="A",color="#404040")
b2=plt.bar(x+width/2,data2,width=width,label="B",color="#A0A0A0")

# add data annotation (relatively new function)
plt.bar_label(b1)
plt.bar_label(b2)
```

```
# manual implementation
#ax=plt.gca()
#for a,b in zip(x,data):
#    ax.text(a-width/2,b+2,b,va="bottom",ha="center")
#for a,b in zip(x,data2):
#    ax.text(a+width/2,b+2,b,va="bottom",ha="center")


plt.ylim([0,120])

# adding a legend (note the "label" argument above)
plt.legend(loc=2)
plt.tight_layout()
plt.show()
```

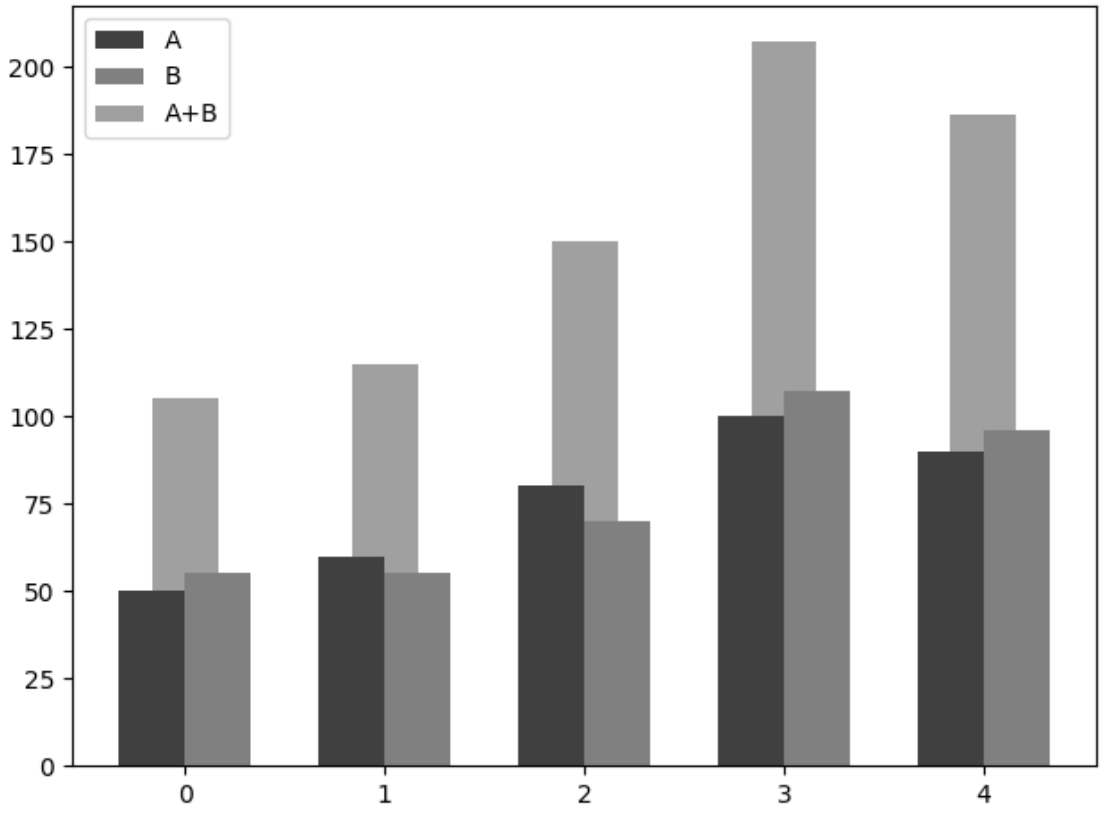


1.5 Stacked

[9]:
```
width=0.5
plt.bar(x,data,width=width,color="#404040")
plt.bar(x,data2,width=width,bottom=data,color="#A0A0A0")
plt.tight_layout()
plt.show()
```

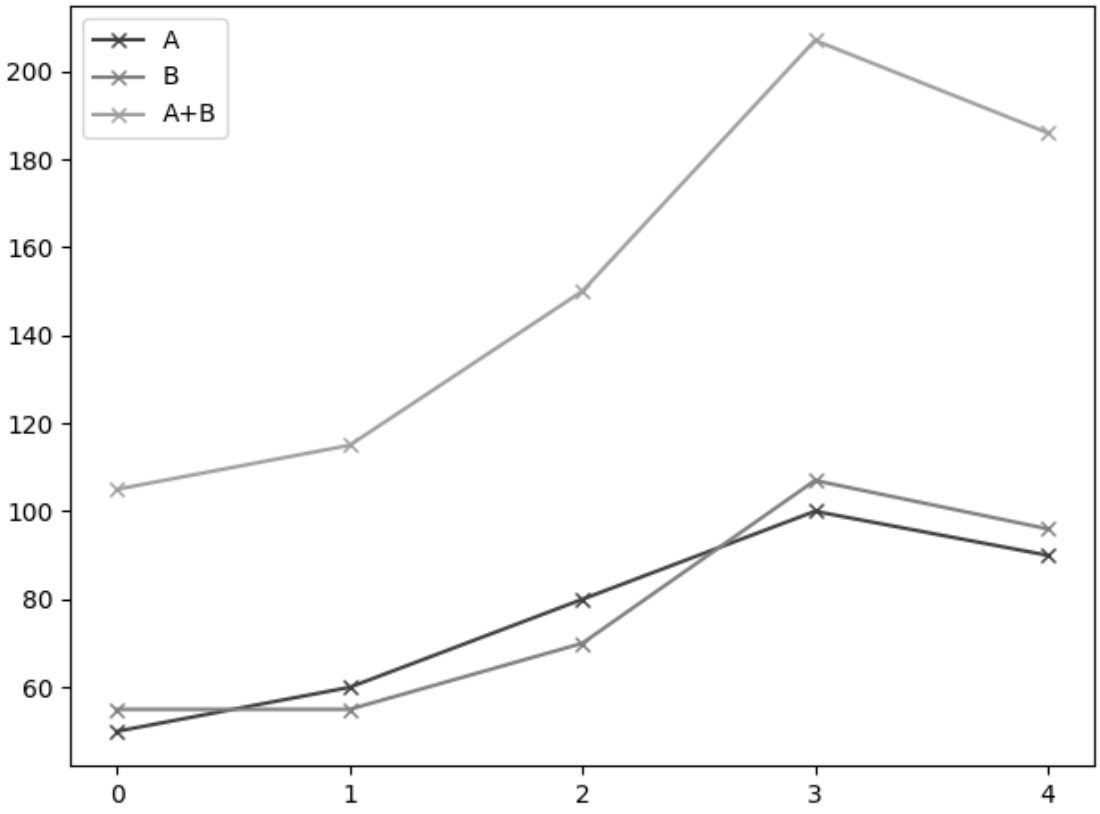


[10]:
```
# an alternative
width=0.33
plt.bar(x-width/2,data,width=width,color="#404040",label="A")
plt.bar(x+width/2,data2,width=width,color="#808080",label="B")
plt.bar(x,data+data2,width=width,color="#A0A0A0",label="A+B",zorder=-1)
plt.legend(loc=2)
plt.tight_layout()
plt.show()
```



1.6 Maybe a scatter /line plot after all?

[11]:
```
# if the x-axis is ordinal/interval data, a line graph often works better
# add lines to visualize trends, visually link points in each data series
n=len(data)
x=np.arange(n)
width=0.5
plt.plot(x,data,marker="x",color="#404040",label="A")
plt.plot(x,data2,marker="x",color="#808080",label="B")
plt.plot(x,data+data2,marker="x",color="#A0A0A0",label="A+B")
plt.xticks(x)
plt.legend(loc=2)
plt.tight_layout()
plt.show()
```


A
B
A+B
200
180
160
140
120
100
80
60
0
1
2
3
4

[ ]: