Discussion Forums

Week 3

SUBFORUMS

All

Assignment: Understanding Sampling from Distributions

Assignment: Building a Custom Visualization

← Week 3

RK

Setting color to bars using cmap

Rohit Kapoor Week 3 · a year ago · Edited

Hi Sophie,

I was able to get to a point where I have to decide the color the bars. I have mean and confidence interval.

Now I don't know how to calculate the value which should be passed to cmap (cmap = matplotlib.cm.get_cmap('Spectral')) function.

I used

if y<=low:

out = 1-(low-y)/(high-low)

else:

out = 1-(y-high)/(high-low)

But gives incorrect results.

I looked the wiki page of cmap but did not understand much. Any insight to this function will be of great help.

Thank you

0 Upvotes Reply Follow this discussion

HIGHLIGHTED POST

SG Sophie Greene Teaching Staff · a year ago

Hi,

here is the example from before modified to include the sign in the remapped/interpolated values

```
#nearest
nearest =100
y=43000
df_p = pd.DataFrame()
df_p['diff'] = nearest*((y-mean)//nearest)
#find the sign of each diff
df_p['sign'] = df_p['diff'].abs()/df_p['diff']
old_range = abs(df_p['diff']).min(),df_p['diff'].abs().max()
new_range =.5,1
#  multiply it by the remaped values
df_p['shade'] = df_p['sign']*np.interp(df_p['diff'].abs(),
                                       old_range,new_range)

df_p['select colour'] = df_p['shade'].apply(
    lambda x:'white' if x==0 else
    'use reds cmap to get color' if x>0 else
    'use blues cmap to get color')
df_p
```

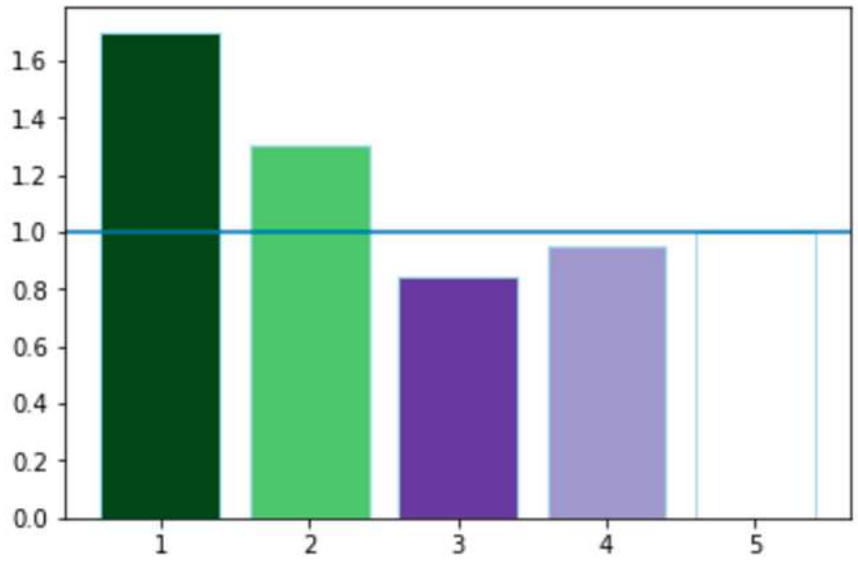
	diff	sign	shade	select colour
1992	9600.0	1.0	1.000000	use reds cmap to get color
1993	1100.0	1.0	0.500000	use reds cmap to get color
1994	3500.0	1.0	0.641176	use reds cmap to get color
1995	-4800.0	-1.0	-0.717647	use blues cmap to get color

In the select colour column , I used plain english to describe what need to be done in each case

the simplest way to go about this is to use the Reds and Blues cmaps and pass them the absolute value of x

here is an example of how to get a cmap and use the shade value to get the desired colour

```
values =np.array([1.7,1.3,.84,.95,1])
y=1
shade = [-.99,-.5,.8,.5,0]
from matplotlib import cm
import matplotlib.pyplot as plt
greens = cm.Greens
purples = cm.Purples
#use shades of green when diff is negative,
# shades of purple when diff is positive
colour=['white' if x==0 else greens(abs(x))
        if x<0 else purples(abs(x)) for x in shade]

plt.bar(range(1,len(values)+1),values,
        edgecolor='lightblue',
        color=colour)
plt.axhline(y=y)
plt.show()
```



HTH!

Sophie

5 Upvotes Jump to post

Earliest **Top** **Most Recent**

BT Baris Tasdelen · 4 months ago

I found a function called get_cmap. There are many color scales you can find at https://matplotlib.org/examples/color/colormaps_reference.html. If you scroll down, there is a color scale called Seismic, which is red - white - blue. The

function works by giving it a integer between 0-255; as in

get_cmap('seismic')(1) would be dark blue;

get_cmap('seismic')(124) would be whitish and

get_cmap('seismic')(255) would be dark red.

It simplified the whole coloring process to a single line. You would need to calculate the difference between the threshold and your column height, and convert it to a number between 0 and 255. It has to be an integer, if you give it a floating number, you get the darkest red.

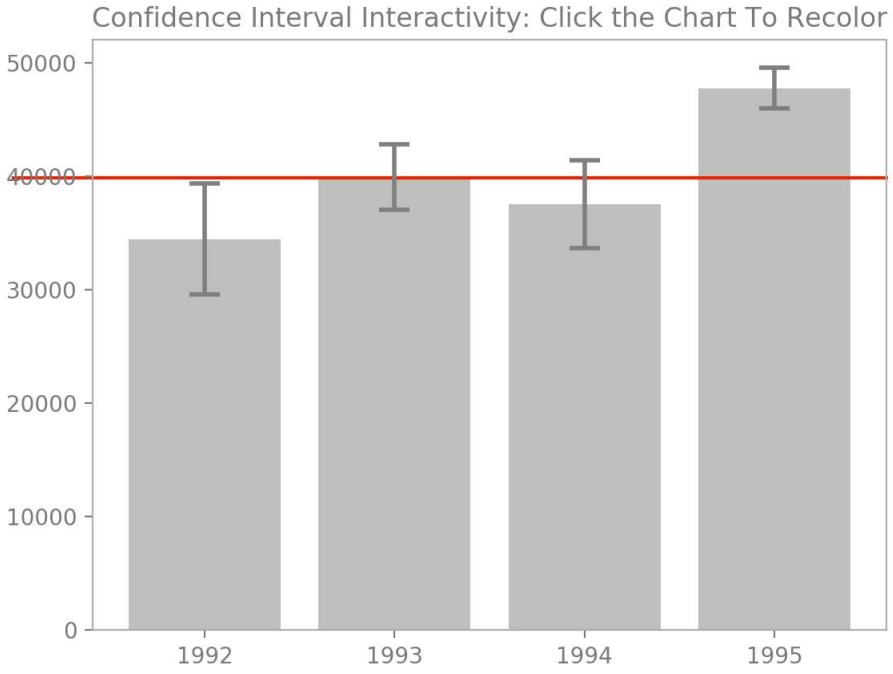
3 Upvotes Reply

David Williams · a year ago · Edited

Sophie, I am having trouble understanding how to use Normalize() and ScalarMappable() to get a shade and color for my bars. Here's my function so far, which simply uses the set_color() method. (Notice I am using references to self. since I have implemented this assignment in a class.)

```
1    def recolorBars(self, event):"""Handles all recoloring of the bars
         based on the confidence that the selectedy-value is within a
         given interval on the chart.This function is called on a button
         press event and receives that data as an argument.""" pass  #
         plt.gca().set_title('Event data {}'.format(event.ydata))  # get
         the yval y = event.ydata # colors = [ cm.ScalarMappable(norm=col
         .Normalize(vmin=i[0] , vmax=i[-1]), cmap=self.cmap) for i in
         self.c_i ]  # apply the colors in a list comprehension # [ rect
         .set_color(color) for rect, color in zip(self.rects, colors) ]
```

My chart looks good except that I am not yet grasping how recoloring works based on confidence intervals and hoped that you'd share a more visual description of how the confidence interval, yval, and mean values all become a part of the final color, as well as share a brief description of how Normalize creates this color mapping.



Thanks in advance.

0 Upvotes Hide 1 Reply

SG Sophie Greene Teaching Staff · a year ago · Edited

ScalarMappable() is used to create a customised colorbar

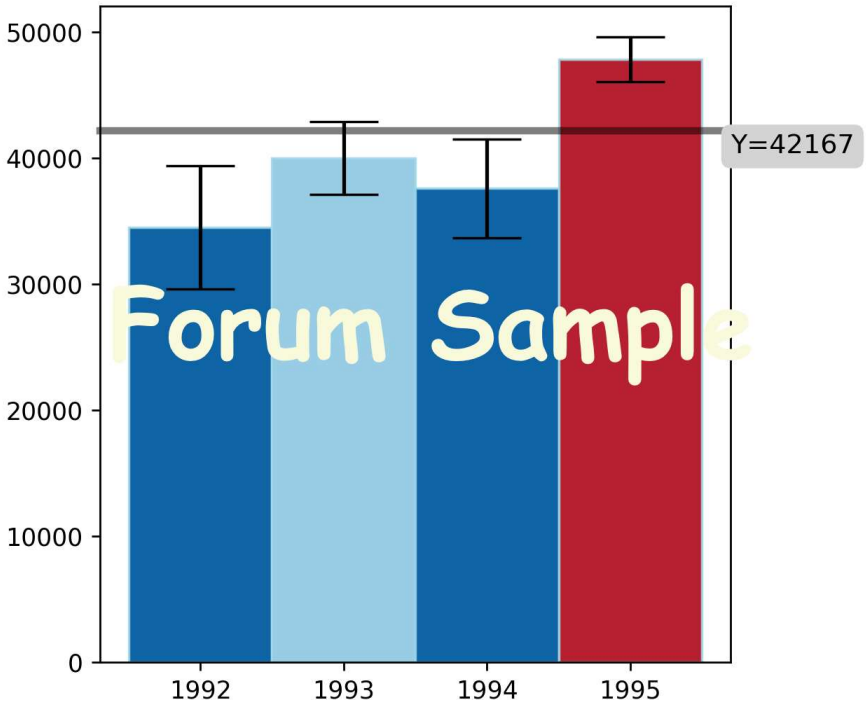
your aim here is to generate the colours of each bar in a bar chart based on certain conditions with respect to the mean/ confidence interval

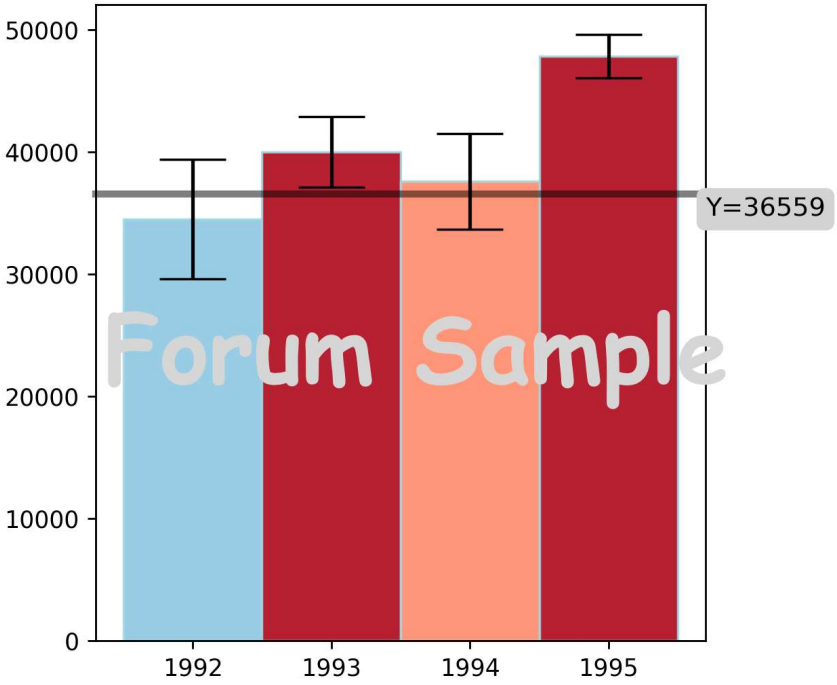
according to the paper,

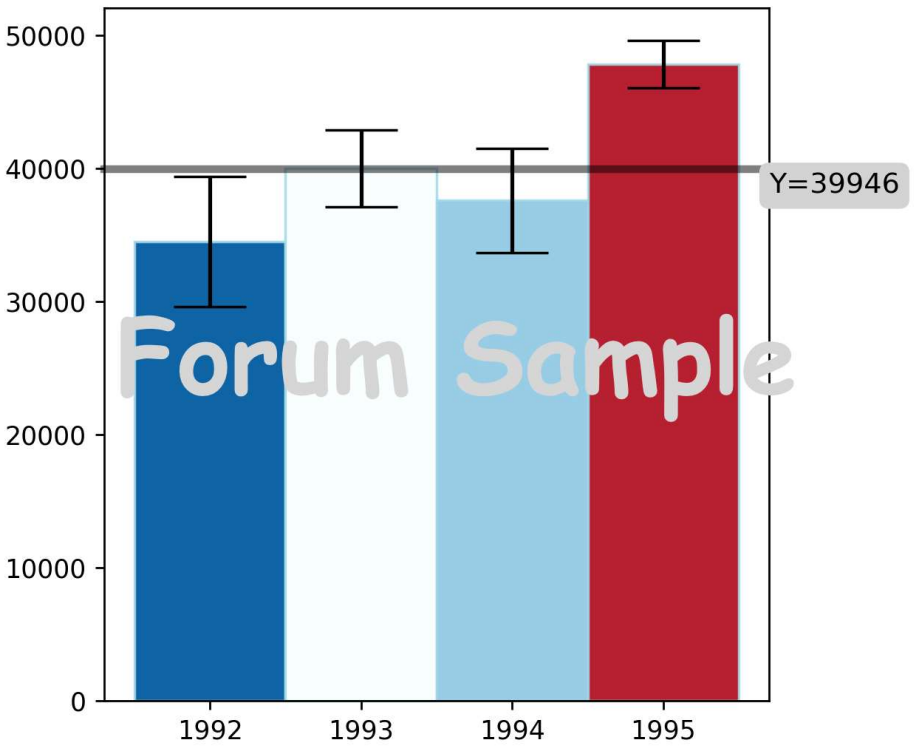
```
(ci['Ci_low'] > y)  #use dark red colors

(ci['Ci_low'] < y) &
(ci['Ci_high'] > y )&
(ci['mean'] > y)  #use light blue colors

(ci['Ci_low'] < y) &
(ci['Ci_high'] > y )&
(ci['mean'] < y) #use light red colors

(ci['Ci_high'] < y)  #use dark blue colors
(ci['mean'] == y) #use white color
```

I found that using the y-mean is simpler, the difference is interpolated to the interval [.4,1], using numpy.interp,

here are a couple of screenshots


50000
40000
30000
20000
10000
0
1992
1993
1994
1995
Y=42167
Forum Sample


50000
40000
30000
20000
10000
0
1992
1993
1994
1995
Y=36559
Forum Sample



1 Upvote

SS

Reply

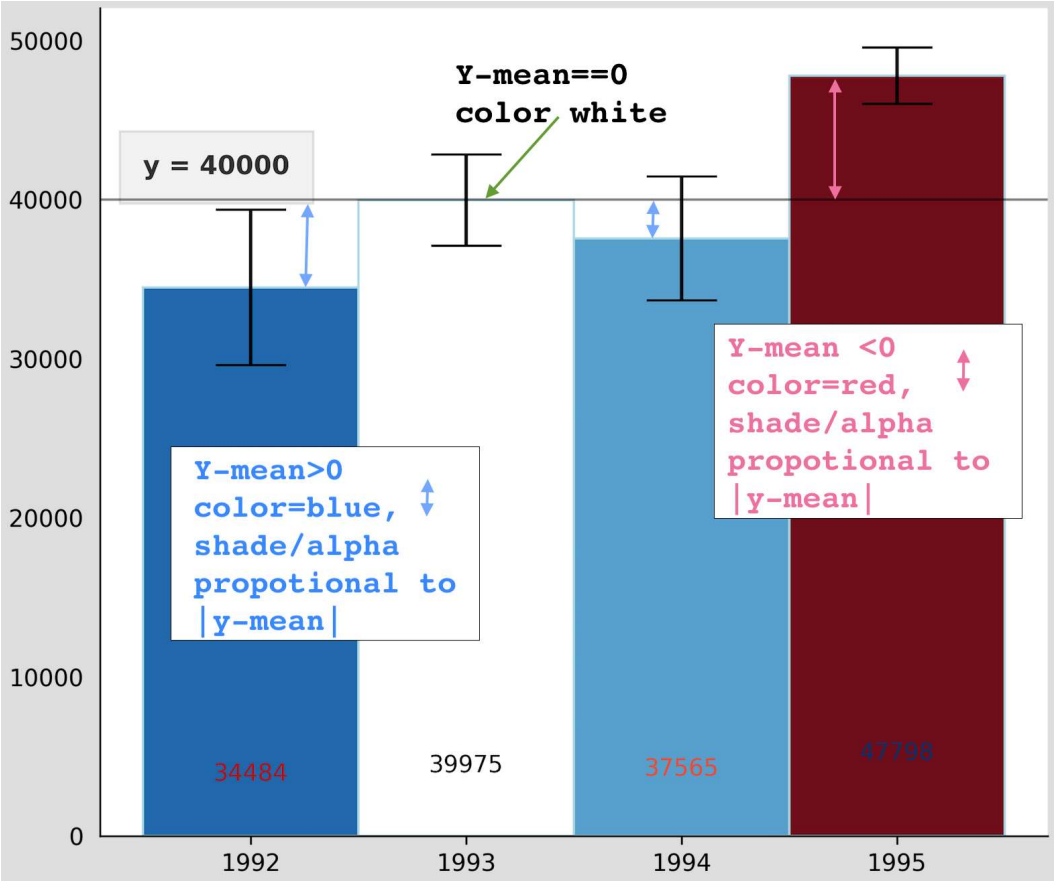
Reply

SG Sophie Greene Teaching Staff · a year ago · Edited

The mean and the dotted line position y, to determine the colour of the bar i.e

```
if y - (mean) < 0:
  color = reds
elif y - (mean) >0:
  colors = blues
else:
  colors = whites
```

and the absolute difference (|mean -y|)determines the colour shade or intensity.



I hope this answers your question

3 Upvotes Hide 23 Replies

See earlier replies

P Pini · a year ago · Edited

Hi all,

First of all, it turns out that I misinterpret the the task in the paper (overthinking it probably..), and the value should be compared to the sample mean +- Confidence Interval (1:34 video), and not to the generating distribution as I previously believed. Hopefully I didn't misled anyone.

Christopher,

I'm afraid that you are giving the wrong interpretation of Confidence Intervals (CI), so as most people (even professionals).

TM is not a random variable - it is an **unknown constant**.

After drawing the sample, and calculating the sample mean (Ms), the Ms is also a constant (and its CI) - not a random variable.

Hence, P(x<(TM-Ms)) != (0, 1), but rather x<(TM-Ms)={0, 1}.

In words, the probability that a constant is smaller than another constant is not a stochastic question - it doesn't exists, but rather yes-no question (& don't know).

CI are not describing the probability that the MS+- CI contains the Tm, but the level of confidence **you** have s.t. the sample you got is a true representative of the generating distribution.

Hypothetically, one can sample (S1) from an unknown normal distribution, and calculate its Ms1 and 95%CI1, then draw S2 and calculate Ms2 & 95%CI2 and so on, up to Sn.

What the 95%CI means is that for 95% of the samples, the TM will be contained in the corresponding CI, and that 5% of the samples are so far off, that they won't contain it.

But you can't draw infinite number of samples - only 1. and for that single sample **you** can be 95% sure that the the TM **is contained**.

The question presented to the subjects in the paper is not 'what is the probability that a value is bigger than Sm?' but rather 'how certain they are that the value is bigger than Sm?' (table 6 and the paragraph under it).

There might be a subtle difference between the two, but it is still important.

⇧ 0 Upvotes

CT Christopher Tabb · a year ago

Thank for your reply and time in posting it.

⇧ 0 Upvotes

KG Kanika Grover · a year ago

Hi Sophie,

In your posts above, you have written two separate ways of setting the colors:

FIRST

abs_v = abs(y-(m)))

if y - (m) < 0:

color = blues_cmap(remap(abs_v))

elif y - (m) >0:

colors = reds_cmap(remap(abs_v))

else:

color = "white"

SECOND

if y - (mean+conf) < 0:

color = reds

elif y - (mean+conf) >0:

colors = blues

else:

colors = whites

Among these approaches, I understand that the first one seems to be more appropriate. Correct me if I am wrong. Also, for the colors, which approach is correct red, blue, white or blue, red, white? because you have mentioned different colors.

Thank you!

0 Upvotes

SG Sophie Greene Teaching Staff · a year ago

The assignment is open to interpretation, so there is no correct or wrong approach, that said if you want to follow the exact approach the paper followed. you need to use

```
if y< m-ci :
    return darkreds
if y<m and y> m-ci :
    return lightreds
if y> m+ci :
    return darkblues
if y > m and y< m+ci:
    return lightblues
if y == m:
    return whites
```

0 Upvotes

KG Kanika Grover · a year ago

Great. Thanks!

0 Upvotes

AC Alexander Carratt · a year ago

Hi Sophie,

Is the remap function one that you have created on the fly, or does it come from a module? I am getting an error when trying to create the df_p['shade'] column. Could you please indicate the module?

Thanks

0 Upvotes

SG Sophie Greene Teaching Staff · a year ago · Edited

Hi Alexander,

At the time; I wrote my own; but I found out later about numpy.interp which can be applied directly to a pandas series

```
from numpy import interp
old_range = [df['diff'].min(),df['diff'].max()]
new_range =[.5,1]
interp(df['diff'],oldrange,new_range)
```

```
array([ 1.        ,  0.52573158,  1.        ])
```

HTH

Sophie

0 Upvotes

AC Alexander Carratt · a year ago · Edited

Sophie,

Thank you for your prompt response.

I have been trying to figure out the coloring part of the assignment. Up to this point, I have my bars and CIs plotted and I have caclulated my df_p['colorselect'] column. My code which produces an error is shown below:

```
plt.figure()

cmaps = [('Sequential', ['Greys', 'Purples', 'Blues',
    'Greens', 'Oranges', 'Reds', 'YlOrBr', 'YlOrRd', 'OrRd',
    'PuRd', 'RdPu', 'BuPu', 'GnBu', 'PuBu', 'YlGnBu',
    'PuBuGn', 'BuGn', 'YlGn'])]

_ = plt.bar(range(len(df.columns)), means , yerr=conf,
    capsize=13, color=plt.get_cmap(df_p.ix[:,'colorselect']
    ))

line = plt.axhline(y,  xmin=-.1, c='k', clip_on=False)
plt.xticks(range(len(df.columns)), ('1992','1993','1994'
    ,'1995'))
plt.tick_params(bottom='off', left='on')
```

The problem is whith my color=df_p.ix[: , 'colorselect'] argument. Are you able to provide any hints as to how to iterate over the bars and apply the coloring based on the df_p['colorselect'] column suggested previously?

Thanks a lot

0 Upvotes

SG Sophie Greene Teaching Staff · a year ago

Hi,

here is the example from before modified to include the sign in the remapped/interpolated values

```
#nearest
nearest =100
y=43000
df_p = pd.DataFrame()
df_p['diff'] = nearest*((y-mean)//nearest)
#find the sign of each diff
df_p['sign'] = df_p['diff'].abs()/df_p['diff']
old_range = abs(df_p['diff']).min(),df_p['diff'].abs().max()
new_range =.5,1
#  multiply it by the remaped values
df_p['shade'] = df_p['sign']*np.interp(df_p['diff'].abs(),
                                       old_range,new_range)

df_p['select colour'] = df_p['shade'].apply(
    lambda x:'white' if x==0 else
    'use reds cmap to get color' if x>0 else
    'use blues cmap to get color')
df_p
```

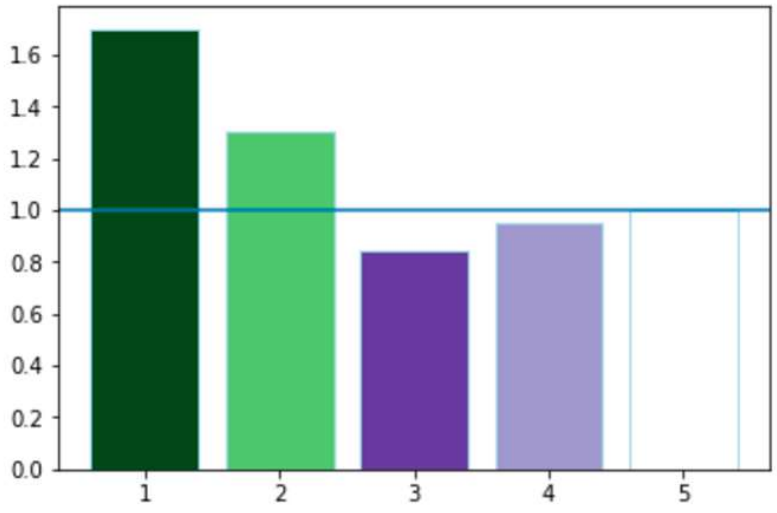
	diff	sign	shade	select colour
1992	9600.0	1.0	1.000000	use reds cmap to get color
1993	1100.0	1.0	0.500000	use reds cmap to get color
1994	3500.0	1.0	0.641176	use reds cmap to get color
1995	-4800.0	-1.0	-0.717647	use blues cmap to get color

In the select colour column , I used plain english to describe what need to be done in each case

the simplest way to go about this is to use the Reds and Blues cmaps and pass them the absolute value of x

here is an example of how to get a cmap and use the shade value to get the desired colour

```
values =np.array([1.7,1.3,.84,.95,1])
y=1
shade = [-.99,-.5,.8,.5,0]
from matplotlib import cm
import matplotlib.pyplot as plt
greens = cm.Greens
purples = cm.Purples
#use shades of green when diff is negative,
# shades of purple when diff is positive
colour=['white' if x==0 else greens(abs(x))
        if x<0 else purples(abs(x)) for x in shade]

plt.bar(range(1,len(values)+1),values,
        edgecolor='lightblue',
        color=colour)
plt.axhline(y=y)
plt.show()
```



HTH!

Sophie

5 Upvotes

AC Alexander Carratt · a year ago

Thanks Sophie. The above really helped. I was unclear, as I am sure many will be, on how to initialize the Reds and Blues cmaps.

0 Upvotes

CB Chris Brake · 8 months ago · Edited

Hi all,

I found that the simplest way to do the bar colouring was to use the scipy stats (as st) cumulative distribution function for a normal distribution: st.norm.cdf()

If you define sem (std. error of the mean) as:

sem = std. deviation of sample / sqrt(sample size)

and want to define colour for a partcular test value of y (y_test) then call:

```
import scipy.stats as st

prob = st.norm.cdf((y_test - mean)/(sem))
```

Which gives the cumulative probability that the actual mean for a population is below the defined 'y_test'. So it will be 0.5 if the test is exactly the same as the mean, near 1 if much higher, and near 0 if much lower.

Then use the matlpotlib cm module, and pick colour scale RdBu (which is white at the middle, and red and blue at each end), and call colour as:

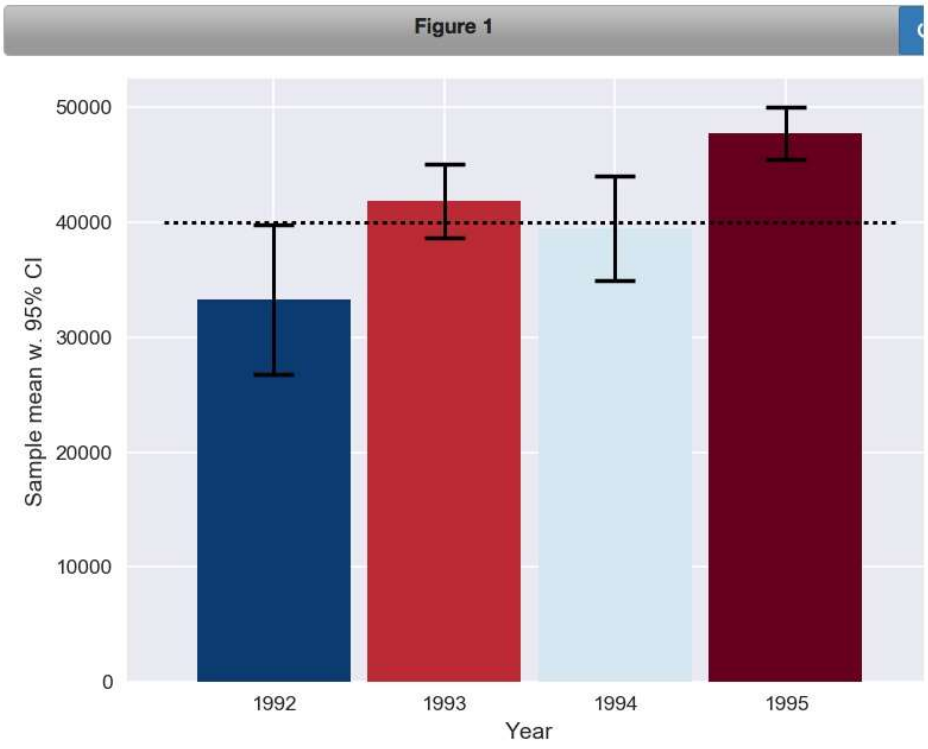
bar_colour = cm.RdBu(prob)

```
from matplotlib import cm
bar_colour = cm.RdBu(prob)
```

to colour bar. (As far as I can see, it is the RdBu colour spectrum that the authors used in their original paper).

Both the operations above can be performed on arrays, so you can create a 'colour array' for all bars (against a given y_test) at once in 2 lines of code.

Output appears as below:



Hope this helps.

Chris

⇧ 6 Upvotes

ShangPei · 6 months ago

Good solution!

But I think if we change one normal distribution into the standard normal distribution we should apply:

(y_test - mean)/std(samples)

so:

prob = st.norm.cdf((y_test - mean)/std(samples))

⇧ 0 Upvotes

CB Chris Brake · 6 months ago · Edited

You are quite right - and in fact there was an error in my original post (compared to the actual working code).

The first equation SHOULD be for the standard error of the mean (sem) of the sample.

sem = std. deviation of sample / sqrt(sample size)

I have amended my original post above.

(Note, I used sem (as opposed to std) as I assumed that the 'use case' for this was to take a given sample of data (with mean and std.) and compare it to historic data, and see the probability that it belonged to a particular year - in which case I *think* the sem is the appropriate metric).

But welcome further thoughts / feedback.

1 Upvote

ShangPei · 6 months ago

Yes, I agree with you!

0 Upvotes

Kevin Duffy · 5 months ago

Chris:

Thanks great solution

SS

0 Upvotes
Reply

Reply

1

SS

Reply

Reply