Assignment 3 - Building a Custom Visualization

In this assignment you must choose **one** of the options presented below and submit a visual as well as your source code for peer grading. The details of how you solve the assignment are up to you, although your assignment must use matplotlib so that your peers can evaluate your work. The options differ in challenge level, but there are no grades associated with the challenge level you chose. However, your peers will be asked to ensure you at least met a minimum quality for a given technique in order to pass. Implement the technique fully (or exceed it!) and you should be able to earn full grades for the assignment.

Ferreira, N., Fisher, D., & Konig, A. C. (2014, April). Sample-oriented task-driven visualizations: allowing users to make better, more confident decisions. (https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Ferreira_Fisher_Sample_Oriented_Tasks.pdf)   In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 571-580). ACM. (video (https://www.youtube.com/watch?v=BI7GAs-va-Q))

In this paper (https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Ferreira_Fisher_Sample_Oriented_Tasks.pdf) the authors describe the challenges users face when trying to make judgements about probabilistic data generated through samples. As an example, they look at a bar chart of four years of data (replicated below in Figure 1). Each year has a y-axis value, which is derived from a sample of a larger dataset. For instance, the first value might be the number votes in a given district or riding for 1992, with the average being around 33,000. On top of this is plotted the 95% confidence interval for the mean (see the boxplot lectures for more information, and the yerr parameter of barcharts).

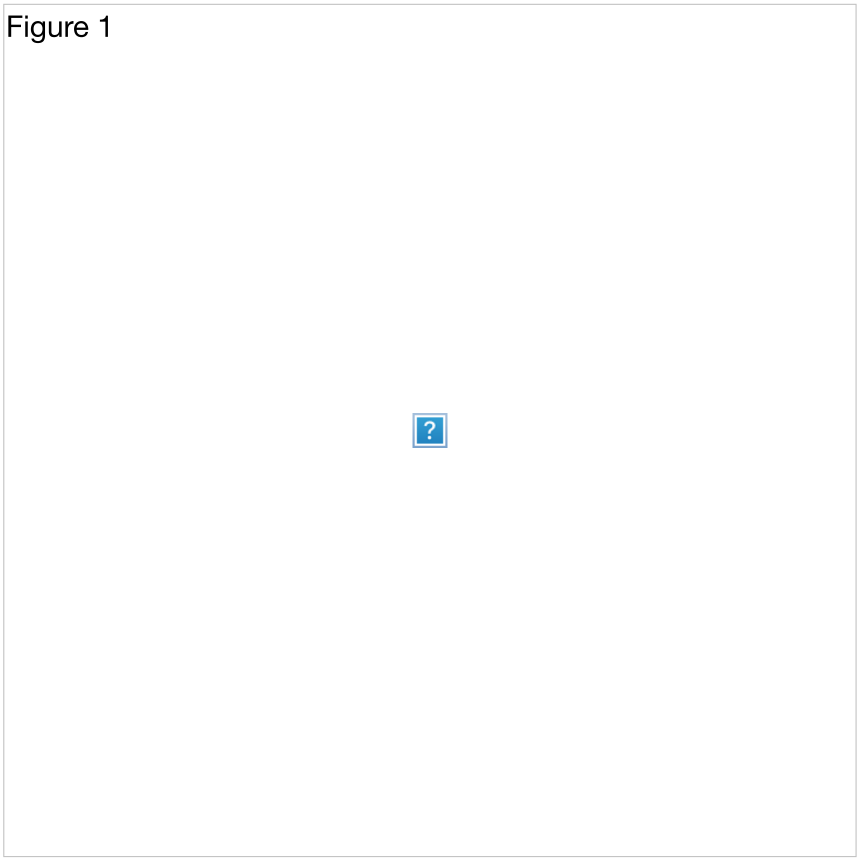


Figure 1 from (Ferreira et al, 2014).

A challenge that users face is that, for a given y-axis value (e.g. 42,000), it is difficult to know which x-axis values are most likely to be representative, because the confidence levels overlap and their distributions are different (the lengths of the confidence interval bars are unequal). One of the solutions the authors propose for this problem (Figure 2c) is to allow users to indicate the y-axis value of interest (e.g. 42,000) and then draw a horizontal line and color bars based on this value. So bars might be colored red if they are definitely above this value (given the confidence interval), blue if they are definitely below this value, or white if they contain this value.

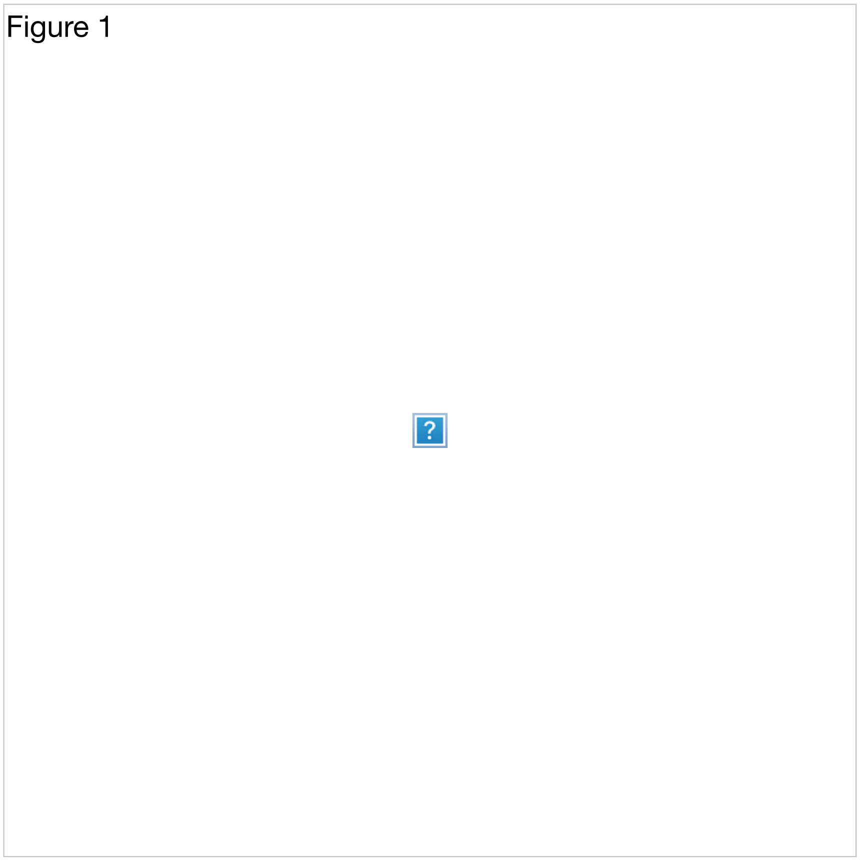


Figure 2c from (Ferreira et al. 2014). Note that the colorbar legend at the bottom as well as the arrows are not required in the assignment descriptions below.

Easiest option: Implement the bar coloring as described above - a color scale with at least three colors, (e.g. blue, white, and red). Assume the user provides the y axis value of interest as a parameter or variable.

Harder option: Implement the bar coloring as described in the paper, where the color of the bar is actually based on the amount of data covered (e.g. a gradient ranging from dark blue for the distribution being certainly below this y-axis, to white if the value is certainly contained, to dark red if the value is certainly not contained as the distribution is above the axis).

Even Harder option: Add interactivity to the above, which allows the user to click on the y axis to set the value of interest. The bar colors should change with respect to what value the user has selected.

Hardest option: Allow the user to interactively set a range of y values they are interested in, and recolor based on this (e.g. a y-axis band, see the paper for more details).

Note: The data given for this assignment is not the same as the data used in the article and as a result the visualizations may look a little different.

In [40]:

```
# Use the following data for this assignment:

import pandas as pd
import numpy as np

np.random.seed(12345)

df = pd.DataFrame([np.random.normal(32000,200000,3650),
                   np.random.normal(43000,100000,3650),
                   np.random.normal(43500,140000,3650),
                   np.random.normal(48000,70000,3650)],
                  index=[1992,1993,1994,1995])
df
```

Out[40]:

	0	1	2	3	4
1992	-8941.531897	127788.667612	-71887.743011	-79146.060869	425156.114501
1993	-51896.094813	198350.518755	-123518.252821	-129916.759685	216119.147314
1994	152336.932066	192947.128056	389950.263156	-93006.152024	100818.575896
1995	-69708.439062	-13289.977022	-30178.390991	55052.181256	152883.621657

4 rows × 3650 columns

In [41]:

```
#Your Code Here
df = df.transpose()
df.describe()
```

Out[41]:

	1992	1993	1994	1995
count	3650.000000	3650.000000	3650.000000	3650.000000
mean	33312.107476	41861.859541	39493.304941	47743.550969
std	200630.901553	98398.356203	140369.925240	69781.185469
min	-717071.175466	-321586.023683	-450827.613097	-189865.963265
25%	-102740.398364	-26628.302213	-57436.397393	1774.555612
50%	29674.931050	43001.976658	41396.781369	49404.322978
75%	167441.838695	108296.577923	137261.713785	94164.333867
max	817505.608159	395586.505068	490091.665037	320826.888044

In [42]:

```
import math
mean = list(df.mean())
std = list(df.std())
ye1 = []
for i in range (4) : ye1.append(1.96*(std[i]/math.sqrt(len(df))))
ye1
```

Out[42]:

```
[6508.897969970325, 3192.2543136890313, 4553.902287088243, 226
3.8517443103765]
```

In [43]:

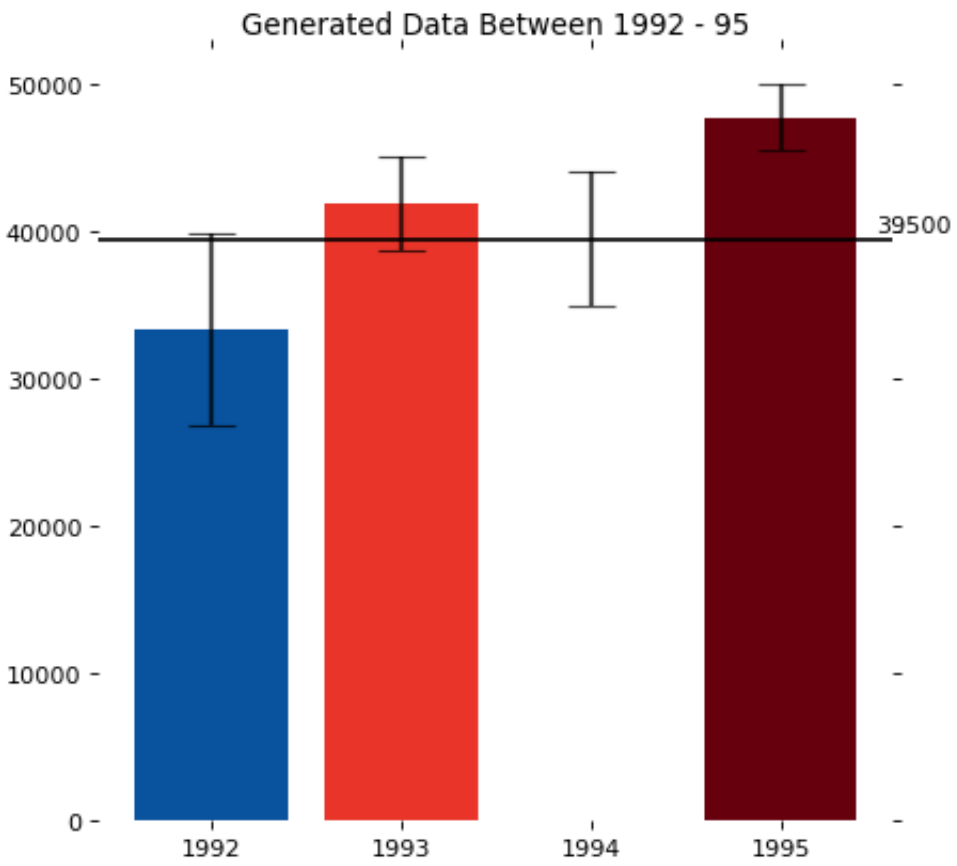
```
nearest = 100
Y = 39500
df_p = pd.DataFrame()
df_p['diff'] = nearest*((Y - df.mean())//nearest)
df_p['sign'] = df_p['diff'].abs()/df_p['diff']
old_range = abs(df_p['diff']).min(), df_p['diff'].abs().max()
new_range = .5,1
df_p['shade'] = df_p['sign']*np.interp(df_p['diff'].abs(), old_range, new_
range)
```

In [44]:

```
shade = list(df_p['shade'])
from matplotlib import cm
blues = cm.Blues
reds = cm.Reds
# using shades blues when diff is pos
# using Reds when when diff is neg
color = ['White' if x == 0 else reds(abs(x))
if x<0 else blues(abs(x)) for x in shade]
```

In [45]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(num=None, figsize=(6, 6), dpi=80, facecolor='w', edgecolor='k')
plt.bar(range(len(df.columns)), height = df.values.mean(axis = 0),
        yerr=ye1, error_kw={'capsize': 10, 'elinewidth': 2, 'alpha':0.7},
color = color)
plt.axhline(y=Y, color = 'black', label = 'Y')
plt.text(3.5, 40000, "39500")
plt.xticks(range(len(df.columns)), df.columns)
plt.title('Generated Data Between 1992 - 95')
# remove all the ticks (both axes), and tick labels on the Y axis
plt.tick_params(top='off', bottom='off',  right='off', labelbottom='on')
# remove the frame of the chart
for spine in plt.gca().spines.values(): spine.set_visible(False)
plt.show()
```


Generated Data Between 1992 - 95
50000
40000
30000
20000
10000
0
39500
1992
1993
1994
1995

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```