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. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 571-580). ACM. (video)

In this paper 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).

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.

Easiest option: Implement the bar coloring as described above - a color scale with only 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 [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import matplotlib.transforms as transforms
from scipy import stats

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])
```

In [2]:

```
#calculate mean of each row/year
mean = df.mean(axis=1)
#calculate std of each row
std = df.std(axis = 1)
#number of rows/sample size
n= df.shape[1]
```

```
#confidence interval for each row/year: Standard Error * a constant derived from it being a 95% CI
yerr = std / np.sqrt(n) * stats.t.ppf(1-0.05/2, n - 1)
plt.figure()
#create value for vertical line you will draw
line=44000
```

In [11]:

```
#create dataframe from which you will draw color value for each year
colors=pd.DataFrame()
#put mean and confidence interval (distance from mean to end of CI) of each year in as new columns
colors['mean']=mean
colors['yerr']=.5*yerr
#put in color column and values based on whether line is within CI for each year
colors['color']=np.where(line>mean+yerr, 'blue', np.where(line<mean-yerr, 'red', 'white'))
#create bar plot: x values=index, y values=mean for each year, add a yerr (CI), edgecolor so white
bar shows up
plt.bar(df.index.values, mean, yerr = yerr, color=colors['color'], edgecolor=['black', 'black', 'bl
ack', 'black'])
#make the ticks the years
plt.xticks(df.index.values)
plt.xlabel('Year')
plt.ylabel('Estimated Y Value')
#for the legend, create colored patches showing what color of bars means
red_patch = mpatches.Patch(color='red', label='Above Conf. Int.')
white_patch=mpatches.Patch(color='white', label='Within Conf. Int.', ec='black')
blue_patch=mpatches.Patch(color='blue', label='Below Conf. Int.')
#add legend and place it off the chart
plt.legend(handles=[red_patch, white_patch, blue_patch], loc='upper center', bbox_to_anchor=(0.5, -
0.05), ncol=3, title='Estimated Y Value is:')
#add the horizontal line
plt.axhline(line)
#make the horizontal line red and add the value on the y axis
ax=plt.gca()
trans = transforms.blended_transform_factory(ax.get_yticklabels()[0].get_transform(), ax.transData
)
ax.text(0,line, "{:.0f}".format(line), color="red", transform=trans, ha="right", va="center")
#make sure everything fits
plt.tight_layout()
plt.savefig('Assignment 3')
plt.show()
```