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).

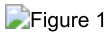


Figure 1 from (Ferreira et al, 2014).

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.

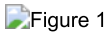


Figure 2c from (Ferreira et al. 2014). Note that the colorbar legend at the bottom as well as the arrows are not required in the assignment descriptions below.

Easiest option: Implement the bar coloring as described above - a color scale with 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.

Import Libraries

In [1]:
```
'''
@author:  Steven Ponce
Date:     07 May 2021
'''
# Importing important libraries
import sys
from sys import exit

import pandas as pd
import numpy as np

%matplotlib notebook
import matplotlib.pyplot as plt
import matplotlib
import matplotlib as mpl

from matplotlib.colors import Normalize
from matplotlib.cm import get_cmap
import seaborn as sns

# Hide warnings
import warnings
warnings.filterwarnings('ignore')

print('You\'re running python %s' % sys.version.split(' ')[0])
print('You\'re running matplotlib: {}'.format(matplotlib.__version__))

# The following lines adjust the granularity of reporting
pd.options.display.max_rows = 10
pd.options.display.float_format = '{:.1f}'.format
```

```
You're running python 3.6.2
You're running matplotlib: 2.0.0
```

Loading the data

In [2]:
```
# Use the following data for this assignment:

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])
```

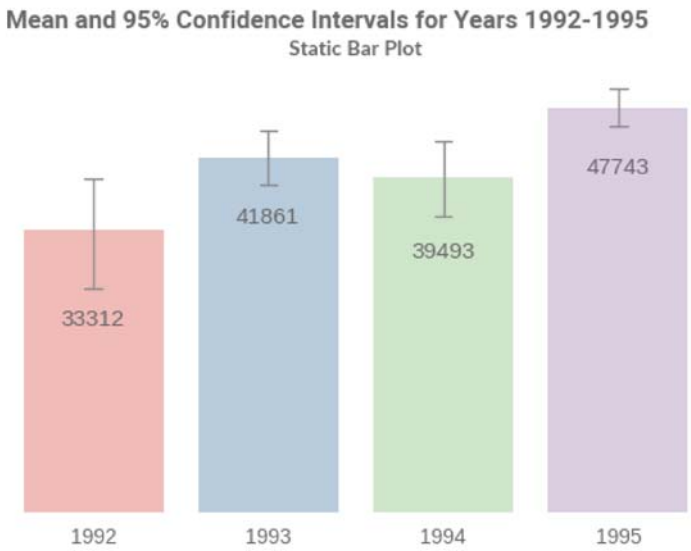
In [3]:
```
# lets transpose the df
data = df.T
data.tail()
```

Out[3]:

	1992	1993	1994	1995
3645	24185.0	-37333.5	-71861.8	39937.2
3646	-56826.7	103019.8	26375.1	139472.1
3647	-67319.8	179746.1	-29328.1	59386.2
3648	113377.3	13455.5	65858.8	73362.2
3649	-4494.9	34442.9	-91542.0	28705.1

Static Bar Plot

In [4]:
```
'''
@author:  Steven Ponce
Date:     07 May 2021
'''
sns.set(style='white')
fig, ax = plt.subplots(figsize=(6,4), dpi=100)

# bring the title down
plt.subplots_adjust(left=None, bottom=None, right=None, top=.85, wspace=None, hspace=None)

# format barplot
sns.barplot(data=data,ci=95, orient='v', errwidth=1, errcolor='0.55', palette='Pastel1', capsize=0.1, rasterized=True)

# remove x-axis
plt.yticks([], []);

# manually label each bar
plt.text(x=0, y=22000, s="33312", color='dimgray', fontsize=11, horizontalalignment='center')
plt.text(x=1, y=34000, s="41861", color='dimgray', fontsize=11, horizontalalignment='center')
plt.text(x=2, y=30000, s="39493", color='dimgray', fontsize=11, horizontalalignment='center')
plt.text(x=3, y=40000, s="47743", color='dimgray', fontsize=11, horizontalalignment='center')

plt.show();
```



In [5]:
```
'''
@author:  Steven Ponce
Date:     07 May 2021
'''

def format_axes(ax):

    '''
    Formatting figure background color and tick size and color
    '''

    ax.set_facecolor('white')
    ax.tick_params(labelsize=10, length=0, labelcolor='dimgray' )
    ax.grid(False)
    ax.set_axisbelow(True)
    [spine.set_visible(False) for spine in ax.spines.values()]

format_axes(ax)
```

In [6]:
```
'''
@author:  Steven Ponce
Date:     07 May 2021
'''

def title_labels(ax):

    '''
    Formatting font type, size, and style
    Add title and x,y labels (if necessary)
    '''

    font_title = {'family': 'roboto',
            'color':  'dimgray',
            'weight': 'bold',
            'size': 12,
    }

    font_mini_title = {'family': 'lato',
            'color':  'dimgray',
            'weight': 'bold',
            'size': 10,
    }
    font_axes = {'family': 'Lato',
            'color':  'dimgray',
            'weight': 'normal',
            'size': 10,
    }

    ax.set_title('Mean and 95% Confidence Intervals for Years 1992-1995\n', loc='left', fontdict=font_title)
    ax.set_title('Static Bar Plot', loc='center', fontdict=font_mini_title)
    #ax.set_xlabel('x-label here', fontdict=font_axes)
    #ax.set_ylabel('y-label here', fontdict=font_axes)

title_labels(ax)
```

In [ ]:
```
```

Animated Bar Plot

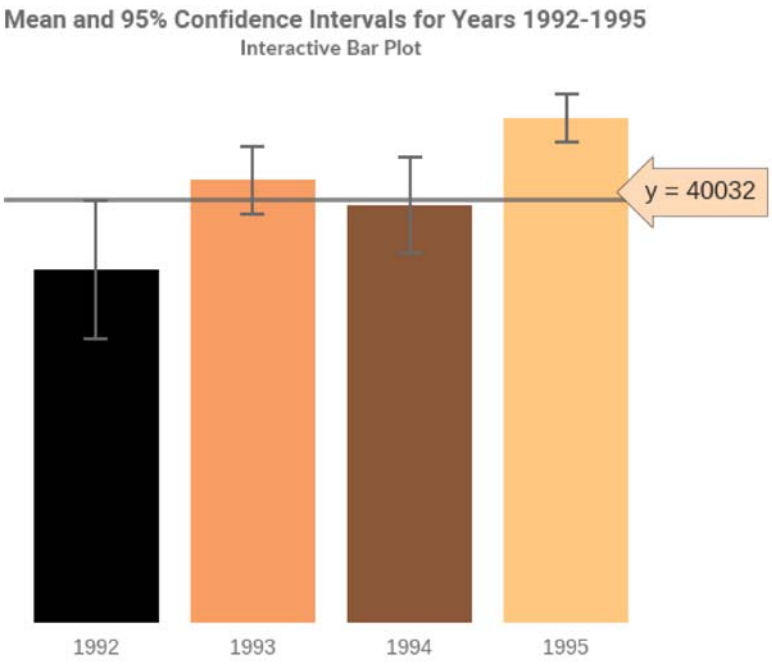
In [7]:
```
# Lets define some variable we're going to use in the box
mean = np.mean(data)

# standard deviation
sd = np.std(data)

# standard error
SE = sd / np.sqrt(df.shape[1])

# CI95
CI = SE * 1.96

# x values
years = data.columns
```

In [8]:
```
# User input for horizontal line value

'''
@author:  Steven Ponce
Date:     07 May 2021
'''

while True:

    try:
        hline_value = input('Enter a value for the horizontal line or type done: ')
        hline_value = int(hline_value)
        break

    except:
        if hline_value.lower().startswith('d'):
            print('goodbye')
            exit()
```

```
Enter a value for the horizontal line or type done: 40032
```

In [9]:

```
'''
@author:  Steven Ponce
Date:     07 May 2021
'''

"""
Confidence Intervals - SPH - Boston University
Link:https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_confidence_intervals/bs704_confidence_intervals_print.html

For the standard normal distribution,  P(-1.96 < Z < 1.96) = 0.95,
i.e., there is a 95% probability that a standard normal variable, Z, will fall between -1.96 and 1.96
"""
norm = Normalize(vmin=-1.96, vmax=1.96)

# bring the title down
fig, ax = plt.subplots(figsize=(6,5), dpi=100)
plt.subplots_adjust(left=None, bottom=None, right=0.85, top=.85, wspace=None, hspace=None)

# remove x-axis
plt.yticks([], []);

# colors
cmap = get_cmap('copper')
colors = pd.DataFrame([])
colors['Int'] = norm((mean-hline_value)/SE)
colors['hue'] = [cmap(x) for x in colors['Int']]

# format barplot
barplot = plt.bar(df.index, mean, color=colors['hue'], rasterized=True)


# format y error bars
plt.errorbar(years, mean, yerr=CI, fmt='.', capsize=6, color="dimgray", lw=1.5, mew=1.5, ms=0)

# Line based on hline_value input
hline = plt.axhline(y=hline_value, color='dimgray', linewidth=2, linestyle='-', alpha = 0.8)

# format of the annotation text
loc = 1995.5
boxtext = plt.text(loc, hline_value, 'y = %d' %hline_value, bbox=dict(boxstyle='larrow', pad=0.5, fc='peachpuff'))

# x-axis labels
plt.xticks(years);

# mouse click event that will show y-axis value
def onclick(event):

    for i in range(4):

        shade = cmap(norm((mean.values[i]-event.ydata)/SE.values[i]))

        barplot[i].set_color(shade)

    hline.set_ydata(event.ydata)

    boxtext.set_text('y = %d' %event.ydata);

    boxtext.set_position((loc, event.ydata));

cid = fig.canvas.mpl_connect('button_press_event', onclick)
```



In [10]:
```
'''
@author:  Steven Ponce
Date:     07 May 2021
'''

def title_labels_2(ax):

    '''
    Formatting font type, size, and style
    Add title and x,y labels (if necessary)
    '''

    font_title = {'family': 'roboto',
            'color':  'dimgray',
            'weight': 'bold',
            'size': 12,
    }

    font_mini_title = {'family': 'lato',
            'color':  'dimgray',
            'weight': 'bold',
            'size': 10,
    }
    font_axes = {'family': 'Lato',
            'color':  'dimgray',
            'weight': 'normal',
            'size': 10,
    }

    ax.set_title('Mean and 95% Confidence Intervals for Years 1992-1995\n', loc='left', fontdict=font_title)
    ax.set_title('Interactive Bar Plot', loc='center', fontdict=font_mini_title)
    #ax.set_xlabel('x-label here', fontdict=font_axes)
    #ax.set_ylabel('y-label here', fontdict=font_axes)

title_labels_2(ax)
```

In [11]:
```
'''
@author:  Steven Ponce
Date:     07 May 2021
'''

def format_axes_2(ax):

    '''
    Formatting figure background color and tick size and color
    '''

    ax.set_facecolor('white')
    ax.tick_params(labelsize=10, length=0, labelcolor='dimgray' )
    ax.grid(False)
    ax.set_axisbelow(True)
    [spine.set_visible(False) for spine in ax.spines.values()]

format_axes_2(ax)
```

In [ ]: