Data Visualization - Homework2

2018 年 4 月 18 日

Create by Linyang He(15307130240)

1 Find/design 5 sets of different data, and use 5 different types of plots to visualize the data using Python and matplotlib; please take a few sentences to describe the data information, background, and visualization effects for analysis. Submit your 5 data sets and code.

Import libs we need

```
In [225]: import numpy as np
          import pandas as pd
          from matplotlib import pyplot as plt
          import matplotlib as mpl
          from matplotlib import animation
          import os
          mpl.style.use('default')
```

1.1 Bar Chart

First, we will create some data. Let's take the males' height data as the example.

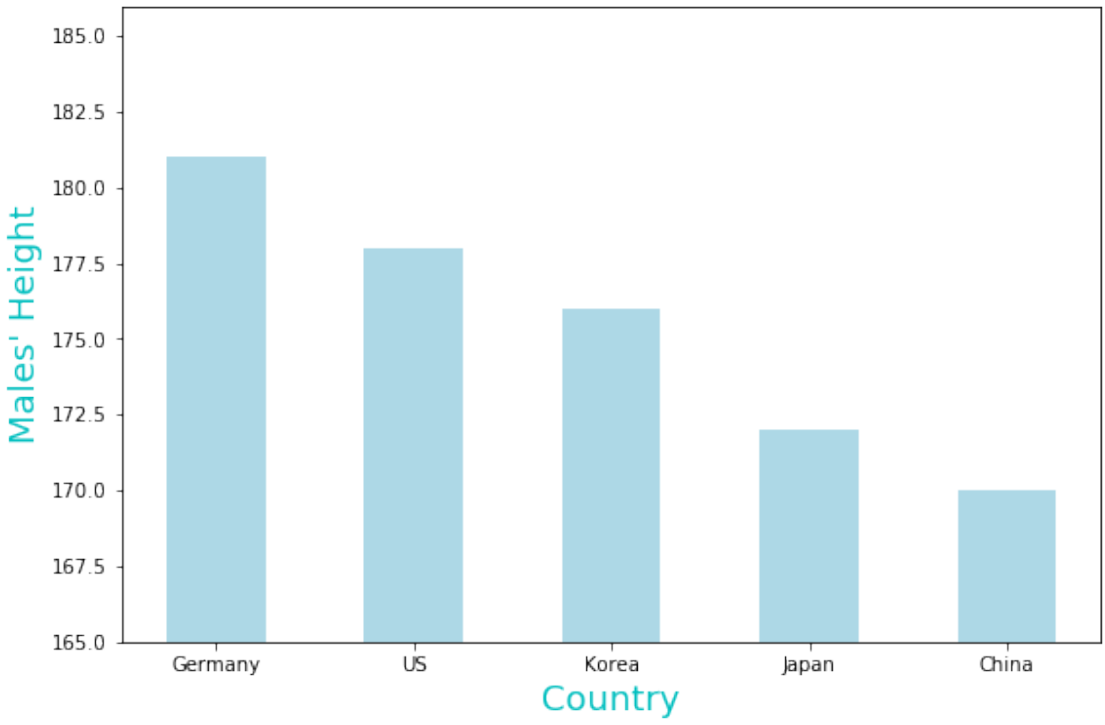
```
In [26]: Males_height = {"Germany":181, "US": 178, "Korea": 176,"Japan":172, "China": 170}
```

```
In [39]: data = list(Males_height.values())
         labels = list(Males_height.keys())
         print(data)
         print(labels)
```

```
[181, 178, 176, 172, 170]
['Germany', 'US', 'Korea', 'Japan', 'China']
```

Next, we will use the matplotlib bar plot to implement data visualization.

```
In [53]: width = .5
         index = [i for i in range(5)]
         fig, ax = plt.subplots(nrows = 1, ncols = 1, figsize=(9,6))
         p1 = ax.bar(left=index,height = data, width = width, color = 'lightblue')
         ax.set_ylim(min(data)-5, max(data)+5)
         ax.set_xticks(index)
         ax.set_xticklabels(labels)
         ax.set_xlabel('Country', color='c', fontsize=18)
         ax.set_ylabel("Males' Height", color='c', fontsize=18)
         plt.show()
```



1.2 Pie Chart

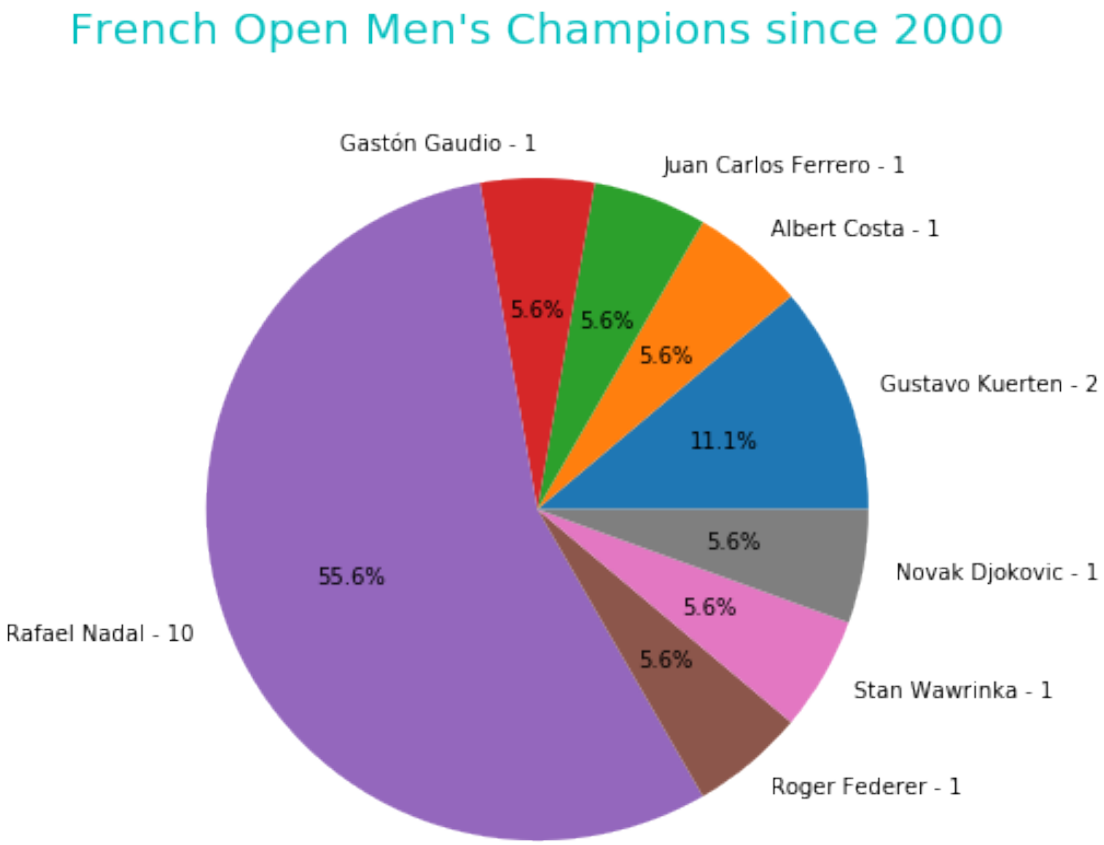
Pie chart is often used in expressing the part-whole information. We will give the examlpe of French Open Men's Champions' data since 21 century. First, we build the dataset.

```
In [59]: French_Open_champions = {"Gustavo Kuerten":2,"Albert Costa":1, "Juan Carlos Ferrero":1
                                  "Gastón Gaudio": 1, "Rafael Nadal":10, "Roger Federer":1,
                                  "Stan Wawrinka": 1, "Novak Djokovic": 1}
         player_names = list(French_Open_champions.keys())
         champion_times = list(French_Open_champions.values())
         print(player_names)
         print(champion_times)
```

```
['Gustavo Kuerten', 'Albert Costa', 'Juan Carlos Ferrero', 'Gastón Gaudio', 'Rafael Nadal', 'R
[2, 1, 1, 1, 10, 1, 1, 1]
```

We will use the some related functions in malplotlib to make a pie chart.

```
In [145]: fig2, pie_ax = plt.subplots(nrows = 1, ncols = 1, figsize=(6,6))
          explode = []
          for champion_time in champion_times:
              if champion_time >= 2:
                  explode.append(0.1)
              else:
                  explode.append(0)
          labels = []
          for i in range(len(champion_times)):
              labels.append(player_names[i] + ' - ' + str(champion_times[i]))
          p2 = pie_ax.pie(champion_times, explode = None, labels= labels, autopct='%1.1f%%')
          pie_ax.axis('equal')
          plt.title("French Open Men's Champions since 2000\n\n", fontsize=20,color = 'c')
          plt.show()
```



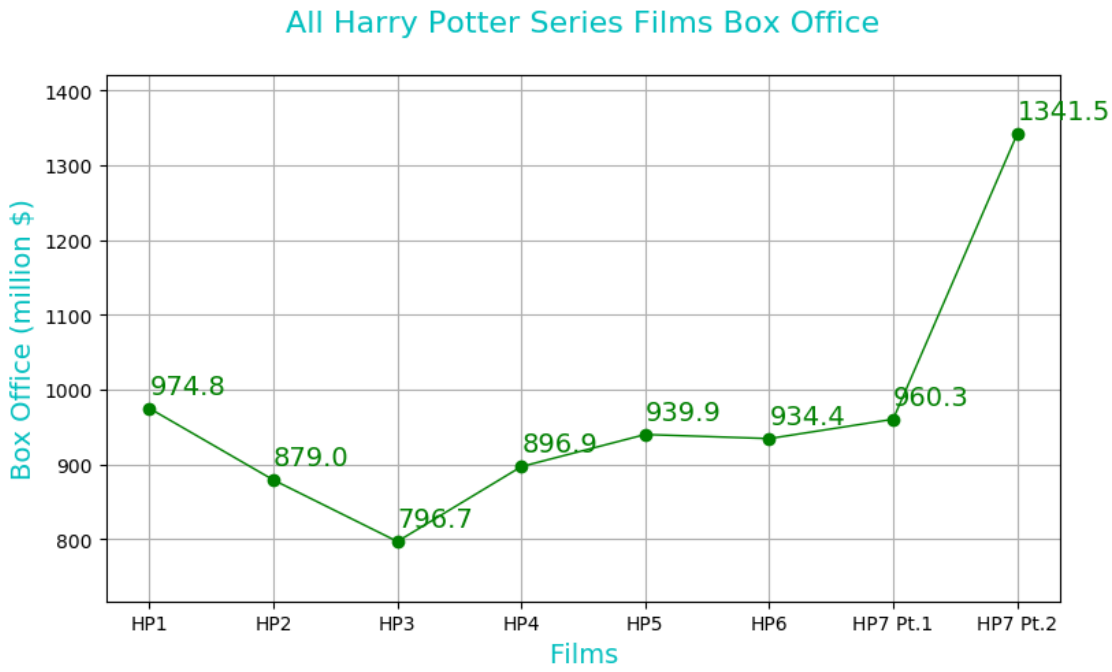
We can find that more half of all the champions were taken by Rafael Nadal. He is definitely the King of Clay!

1.3 Line Chart

Line chart is often used to focus on something's changing. We will take Harry Potter film series' world wide box office as the example. First, let's create the data.

```
In [38]: HP_box = {"Sorcerer's Stone":974.8, "Chamber of Secrets": 879.0,
                   "Prisoner of Azakaban":796.7,"Goblet of Fire":896.9,
                   "Order of the Phoenix":939.9,"Half-Blood Prince":934.4,
                   "Deathly Hallows Part 1": 960.3,"Deathly Hallows Part 2":1341.5}
         films = list(HP_box.keys())
         box_office = list(HP_box.values())

In [39]: films
```

```
Out[39]: ["Sorcerer's Stone",
          'Chamber of Secrets',
          'Prisoner of Azakaban',
          'Goblet of Fire',
          'Order of the Phoenix',
          'Half-Blood Prince',
          'Deathly Hallows Part 1',
          'Deathly Hallows Part 2']

In [5]: box_office

Out[5]: [974.8, 879.0, 796.7, 896.9, 939.9, 934.4, 960.3, 1341.5]

In [56]: fig3, line_ax = plt.subplots(1,1,figsize = (9,5))
         film_index = [i for i in range(8)]
         line_ax.plot(film_index, box_office, data = box_office,
                      label='Harry Potter Film',linewidth=1,color='green',marker='o')
         # line_ax.set_xlim(min(film_index)-1,max(film_index)+1)
         line_ax.set_ylim(min(box_office) - 80, max(box_office) + 80)
         line_ax.set_ylabel("Box Office (million $)", fontsize = 14, color = 'c')
         line_ax.set_xlabel("Films",fontsize = 14, color = 'c')
         line_ax.set_xticks(film_index)
         line_ax.set_xticklabels(['HP1','HP2','HP3','HP4','HP5','HP6','HP7 Pt.1','HP7 Pt.2'])
         # line_ax.set_xticklabels(films, fontsize = 10, color = 'r')
         for i,j in zip(film_index,box_office):
             line_ax.annotate(str(j),xy=(i,j+20),color = 'g',fontsize=14)
         plt.title("All Harry Potter Series Films Box Office\n", fontsize = 16, color = 'c')
         plt.grid()
         plt.show()
```



We can find that the first and the last movie of Harry Potter series have the highest box office, which is quite reasonable.

1.4 Scatter Plot

Next, we will build a scatter diagram using the dataset of men's heights and weights in North America.

First, we need to use the pandas lib to read a csv document.

```
In [3]: human_info = pd.read_csv('./data/NHIS_data_2007.csv')
        human_info.head()
```

Out[3]:

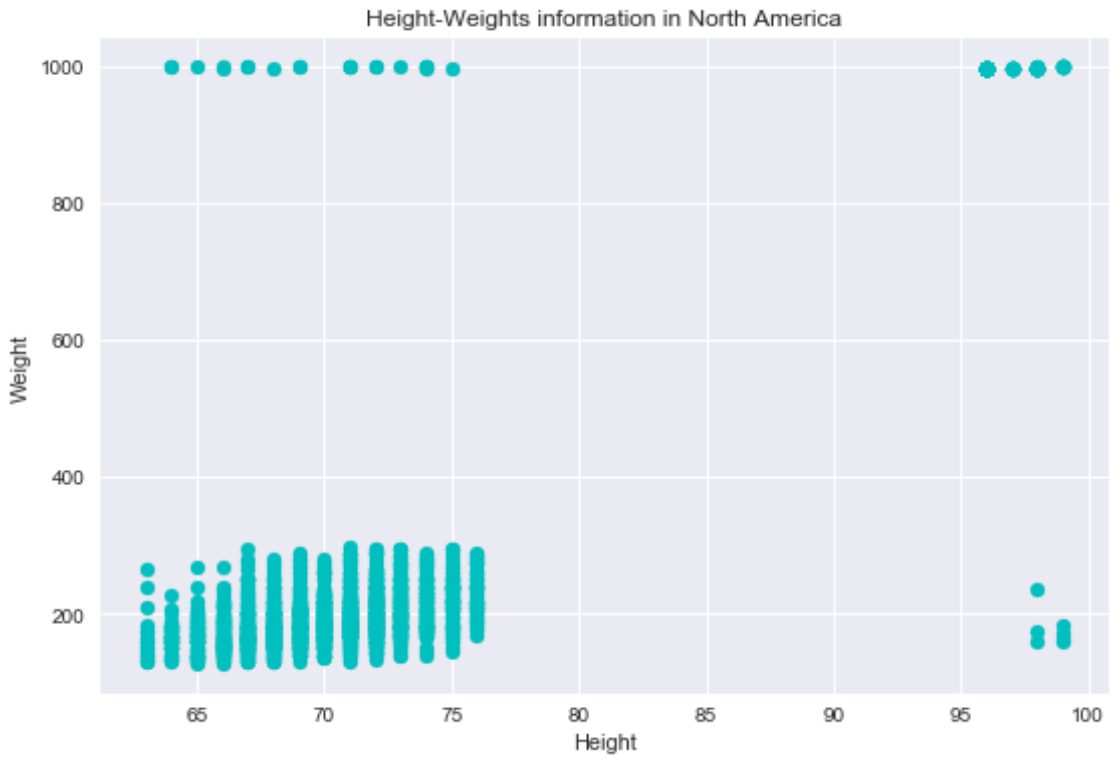
	HHX	FMX	FPX	SEX	BMI	SLEEP	educ	height	weight
0	16	1	2	1	33.36	8	16	74	260
1	20	1	1	1	26.54	7	14	70	185
2	69	1	2	2	32.13	7	9	61	170
3	87	1	1	1	26.62	8	14	68	175
4	88	1	1	2	27.13	8	13	66	168

```
In [4]: heights = human_info[human_info.SEX == 1].height
        heights = list(heights)
```

```
weights = human_info[human_info.SEX == 1].weight
weights = list(weights)
```

Then we will create a scatter plot using the matplotlib.

```
In [10]: fig4, scatter_ax = plt.subplots(1,1,figsize=(9,6))
         p4 = plt.scatter(heights, weights,color = 'c')
         scatter_ax.set_xlabel("Height")
         scatter_ax.set_ylabel("Weight")
         plt.title("Height-Weights information in North America")
         plt.show()
```



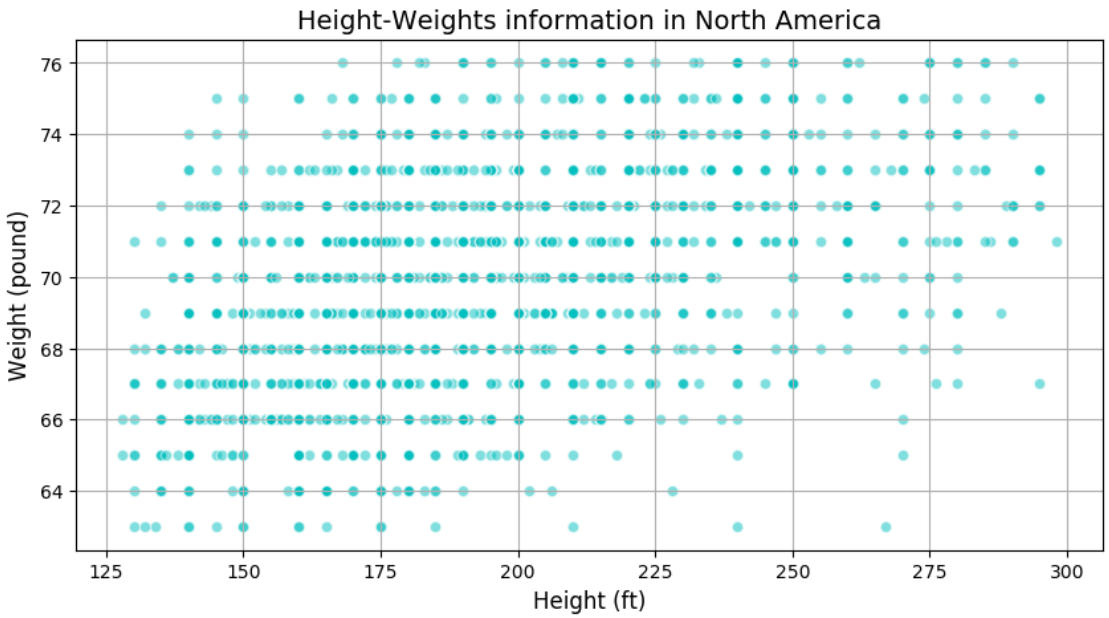
We can find that there are a lot of noisy nodes, so we need to strip them. After we look through the csv file, we find that the noisy weight information will be denoted as '996' and '96' for noisy height.

```
In [30]: heights = human_info[human_info.SEX == 1][human_info.height < 96][human_info.weight <
         heights = list(heights)
```

```
        weights = human_info[human_info.SEX == 1][human_info.height < 96][human_info.weight <
        weights = list(weights)
```

```
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Boolean Series key will be
  """Entry point for launching an IPython kernel.
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: UserWarning: Boolean Series key will be
  after removing the cwd from sys.path.
```

In [78]:
```
fig4, scatter_ax = plt.subplots(1,1,figsize=(10,5))
color = np.arctan2([i/80 for i in heights],[i/300 for i in weights])
p4 = plt.scatter(weights, heights,color = 'c', marker = 'o', alpha = 0.5, edgecolors=
scatter_ax.set_xlabel("Height (ft)", fontsize = 12)
scatter_ax.set_ylabel("Weight (pound)", fontsize = 12)
plt.title("Height-Weights information in North America", fontsize = 14)
plt.grid()
plt.show()
```



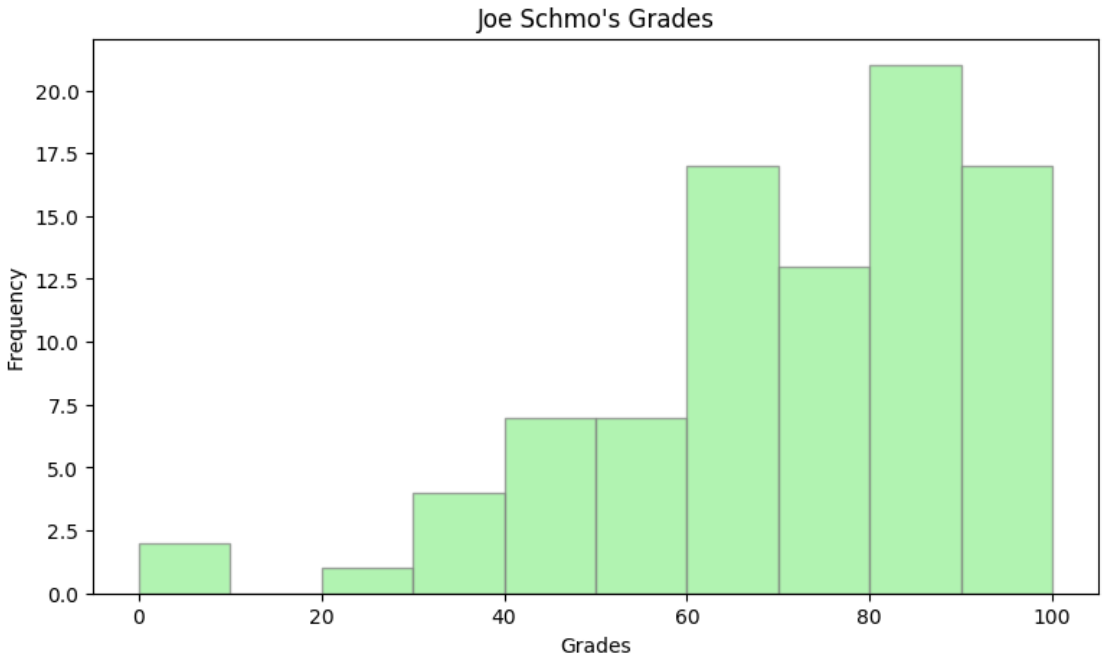
1.5 Histogram

People often uses hisogram to caculate the frequency. We will take the grades of Joe Schmo as the example.

```
In [81]: grades_info = pd.read_csv('./data/grades.csv')
         grades_info.head()
```

```
Out[81]:       Name\t  Grades
         0  Joe Schmo      56
         1  Joe Schmo      98
         2  Joe Schmo      87
         3  Joe Schmo      88
         4  Joe Schmo      58
```

```
In [98]: Grades = list(grades_info.Grades)
         fig5, hist_ax = plt.subplots(1,1,figsize=(9,5))
         p5 = plt.hist(Grades,facecolor="lightgreen", edgecolor="grey", alpha=0.7)
         hist_ax.set_ylabel("Frequency")
         hist_ax.set_xlabel("Grades")
         plt.title("Joe Schmo's Grades")
         plt.show()
```



2 Find/design a dataset and visualize the data using either the techniques of animation or metaphor, or both of them. Please take a few sentences to describe the data information, background, and visualization effects for analysis. Submit your data and code.

We will use the World Cup prediction dataset to create a animation. ## Build the dataset.

In [108]:
```
world_cup_predictions_path = './data/world-cup-predictions/'
path_dir = os.listdir(world_cup_predictions_path)
world_cup_predictions = []
for file_name in path_dir:
    info = pd.read_csv(world_cup_predictions_path + file_name)
    world_cup_predictions += [info]
# world_cup_predictions
countries = world_cup_predictions[0].country
countries.head()
```

Out[108]:
```
0                   Algeria
1                 Argentina
2                 Australia
3                   Belgium
4    Bosnia and Herzegovina
Name: country, dtype: object
```

In [109]:
```
world_cup_predictions[0].head()
```

Out[109]:

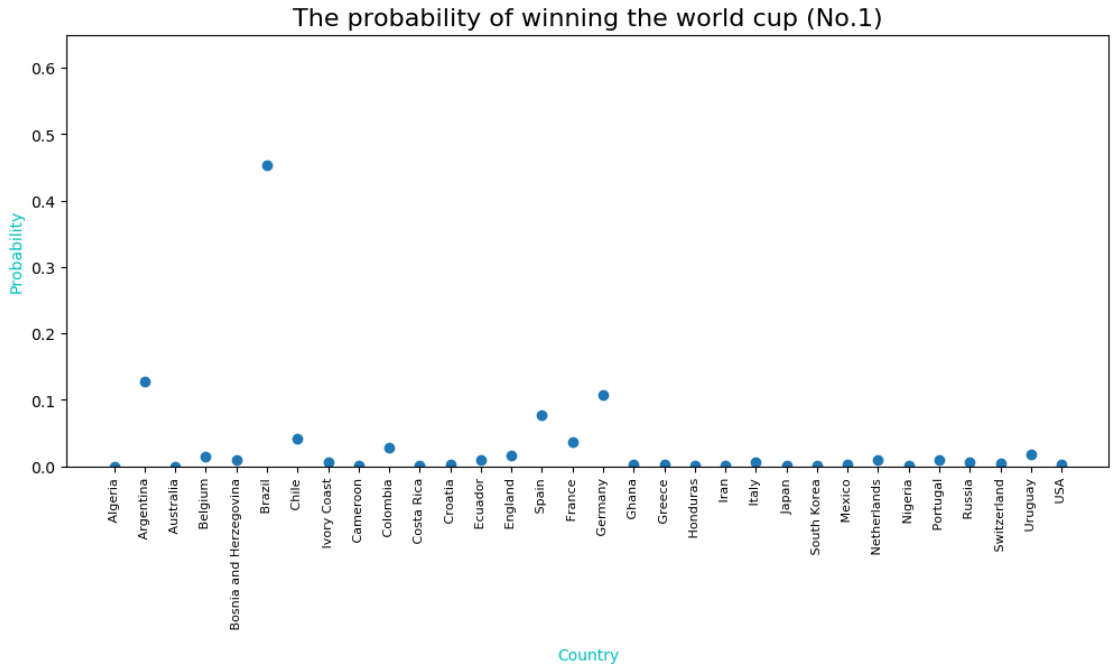
	country	country_id	group	spi	spi_offense	spi_defense	win_group	sixteen	quarter	semi	cup	win
0	Algeria	ALG	h	63.43	1.1208	1.1636	0.0631	0.2032	0.038517	0.007996	0.001021	0.000126
1	Argentina	ARG	f	90.00	2.8541	0.4494	0.7350	0.9279	0.669904	0.468159	0.281758	0.127799
2	Australia	AUS	b	69.45	1.6395	1.2349	0.0151	0.0762	0.009646	0.002943	0.000671	0.000093
3	Belgium	BEL	h	81.97	2.1373	0.7410	0.4781	0.7688	0.351536	0.148459	0.054136	0.014904
4	Bosnia and Herzegovina	BIH	f	80.31	2.3113	0.9861	0.1599	0.5589	0.261950	0.112098	0.031611	0.008964

```
In [128]: countries = list(countries)
          countries_predictions = {}
          for item in world_cup_predictions:
              for cty in countries:
                  try:
                      countries_predictions[cty].append(float(item[item.country==cty].win))
                  except KeyError:
                      countries_predictions[cty] = []
                      countries_predictions[cty].append(float(item[item.country==cty].win))

In [135]: countries_predictions['Algeria'][:5]

Out[135]: [0.000126121865295,
           0.000126121865295,
           0.00012498032398200001,
           0.000120317254567,
           0.000124982295711]

In [136]: len(countries)

Out[136]: 32

In [138]: len(countries_predictions['Algeria'])

Out[138]: 84

In [155]: predictions = np.zeros((32,84))
          index = 0
          for country in countries_predictions:
              predictions[index][:] = countries_predictions[country]
          #     predictions[index] = countries_predictions[country]
              index += 1
          predictions

Out[155]: array([[  1.26121865e-04,   1.26121865e-04,   1.24980324e-04, ...,
                    0.00000000e+00,   0.00000000e+00,   0.00000000e+00],
                 [  1.27798661e-01,   1.27798661e-01,   1.27124613e-01, ...,
                    3.66664253e-01,   3.66610090e-01,   3.78657660e-01],
                 [  9.31090514e-05,   9.31090514e-05,   1.07672710e-04, ...,
                    0.00000000e+00,   0.00000000e+00,   0.00000000e+00],
```

```
       ...,
       [  4.28777499e-03,   4.28777499e-03,   4.27569468e-03, ...,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00],
       [  1.78924584e-02,   1.78924584e-02,   1.79580797e-02, ...,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00],
       [  3.51704115e-03,   3.51704115e-03,   3.55514349e-03, ...,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00]])
```

2.1 Design an animation

```
In [181]: fig6, ani_ax = plt.subplots(1, 1, figsize=(12, 5))

          country_index = [i for i in range(32)]
          p6, = plt.plot(country_index, predictions[:, 0], marker='o', linestyle='None')


          def animate(i):
              p6.set_ydata(predictions[:, i])
              plt.title("The probability of winning the world cup (No." +
                        str(i + 1) + ")", fontsize=16)
              return p6,
          ani = animation.FuncAnimation(
              fig=fig6, func=animate, frames=84, interval=100, blit=False)
          ani_ax.set_xlabel("Country", color='c')
          ani_ax.set_ylabel("Probability", color='c')
          ani_ax.set_ylim(0, 0.65)
          ani_ax.set_xticks(country_index)
          ani_ax.set_xticklabels(["   " + i for i in countries], fontsize=8)
          plt.xticks(rotation=90)
          plt.show()
```



You need to run python script to see the animation. I have put a single python file in the same folder as well as the video I recorded in advance.

3 Visualize the GDP values and changes of GDP of the countries for the past 20 years. The data can be download from http://www.gapminder.org.

This one is just quite similar to the last question.

```
In [183]: gdp_file = pd.read_csv('./data/GDP.csv')
          gdp_file
```

Out[183]:

	Country Name	1960	1961	1962	1963	\
0	China	5.971647e+10	5.005687e+10	4.720936e+10	5.070680e+10	
1	Canada	4.109345e+10	4.076797e+10	4.197885e+10	4.465717e+10	
2	France	6.265147e+10	6.834674e+10	7.631378e+10	8.555111e+10	
3	United Kingdom	7.232805e+10	7.669436e+10	8.060194e+10	8.544377e+10	
4	Japan	4.430734e+10	5.350862e+10	6.072302e+10	6.949813e+10	
5	Korea, Rep.	3.957874e+09	2.417238e+09	2.813934e+09	3.988246e+09	
6	United States	5.433000e+11	5.633000e+11	6.051000e+11	6.386000e+11	

	1964	1965	1966	1967	1968	\
0	5.970834e+10	7.043627e+10	7.672029e+10	7.288163e+10	7.084654e+10	
1	4.888294e+10	5.390957e+10	6.035863e+10	6.476883e+10	7.075903e+10	
2	9.490659e+10	1.021610e+11	1.105970e+11	1.194660e+11	1.298470e+11	
3	9.338760e+10	1.005960e+11	1.070910e+11	1.111850e+11	1.047030e+11	
4	8.174901e+10	9.095028e+10	1.056280e+11	1.237820e+11	1.466010e+11	
5	3.458518e+09	3.120308e+09	3.928171e+09	4.854576e+09	6.117260e+09	
6	6.858000e+11	7.437000e+11	8.150000e+11	8.617000e+11	9.425000e+11	

	...	2007	2008	2009	2010	\
0	...	3.552180e+12	4.598210e+12	5.109950e+12	6.100620e+12	
1	...	1.464980e+12	1.549130e+12	1.371150e+12	1.613460e+12	
2	...	2.663110e+12	2.923470e+12	2.693830e+12	2.646840e+12	
3	...	3.074360e+12	2.890560e+12	2.382830e+12	2.441170e+12	
4	...	4.515260e+12	5.037910e+12	5.231380e+12	5.700100e+12	
5	...	1.122680e+12	1.002220e+12	9.019350e+11	1.094500e+12	
6	...	1.447760e+13	1.471860e+13	1.441870e+13	1.496440e+13	

	2011	2012	2013	2014	2015	\
0	7.572550e+12	8.560550e+12	9.607220e+12	1.048240e+13	1.106470e+13	
1	1.788650e+12	1.824290e+12	1.842630e+12	1.792880e+12	1.552810e+12	
2	2.862680e+12	2.681420e+12	2.808510e+12	2.849310e+12	2.433560e+12	
3	2.619700e+12	2.662090e+12	2.739820e+12	3.022830e+12	2.885570e+12	
4	6.157460e+12	6.203210e+12	5.155720e+12	4.848730e+12	4.383080e+12	
5	1.202460e+12	1.222810e+12	1.305600e+12	1.411330e+12	1.382760e+12	
6	1.551790e+13	1.615530e+13	1.669150e+13	1.739310e+13	1.812070e+13	

	2016
0	1.119910e+13
1	1.529760e+12
2	2.465450e+12
3	2.647900e+12
4	4.940160e+12
5	1.411250e+12
6	1.862450e+13

```
          [7 rows x 58 columns]
```

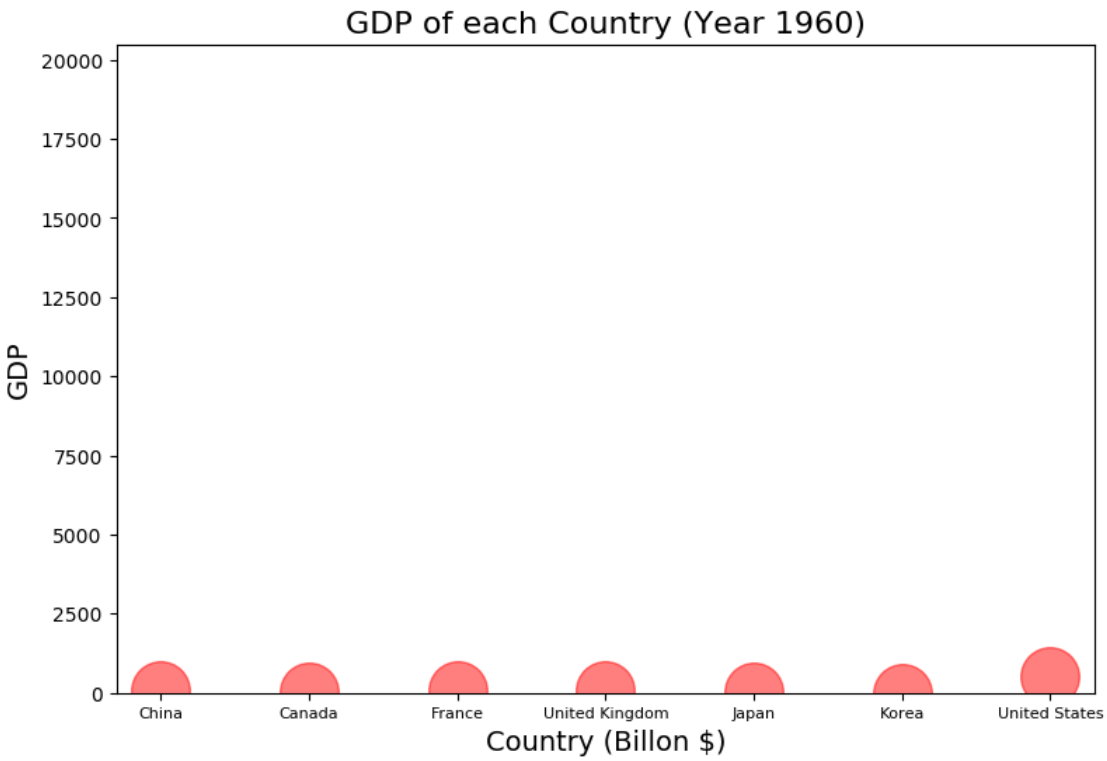
In [221]:
```
gdps = np.zeros((7, 2016-1960 + 1))
countries = []

with open('./data/GDP.csv') as gdp_file:
    gdp_file.readline()
    index = 0
    for line in gdp_file:
        line = line.strip().split(',')
        countries.append(line[0])
        gdps[index] = [float(i)/1e9 for i in line[1:]]
        index += 1
```

In [222]:
```
gdps[5]
```

Out[222]:
```
array([    3.95787393,     2.41723775,     2.8139339 ,     3.98824611,
           3.45851849,     3.12030781,     3.9281713 ,     4.85457637,
           6.11726008,     7.67580511,     8.9992272 ,     9.88996111,
          10.84222047,    13.84188592,    19.48203822,    21.70475207,
          29.77933884,    38.26508264,    51.70061983,    66.56797521,
          64.98082084,    72.42559065,    77.77343109,    87.02442797,
          96.59743418,   100.273     ,   115.537     ,   146.133     ,
         196.964     ,   243.526     ,   279.349     ,   325.734     ,
         350.051     ,   386.303     ,   455.603     ,   556.131     ,
         598.099     ,   557.503     ,   374.241     ,   485.248     ,
         561.633     ,   533.052     ,   609.02      ,   680.521     ,
         764.881     ,   898.137     ,  1011.8       ,  1122.68      ,
        1002.22      ,   901.935     ,  1094.5       ,  1202.46      ,
        1222.81      ,  1305.6       ,  1411.33      ,  1382.76      ,
        1411.25      ])
```

In [224]:
```
fig7, ani_ax_gdp = plt.subplots(1, 1, figsize=(9, 6))

country_index = [i for i in range(7)]
p7, = plt.plot(country_index, gdps[
               :, 0], marker='o', markersize=30, linestyle='None', color='red', alpha
```

```
def animate_gdp(i):
    p7.set_ydata(gdps[:, i])
    plt.title("GDP of each Country (Year " +
              str(i + 1960) + ")", fontsize=16)
    return p7,

ani_gdp = animation.FuncAnimation(fig=fig7, func=animate_gdp,
                                  frames=2016 - 1960 + 1, interval=200, blit=False)
ani_ax_gdp.set_xlabel("Country (Billon $)", fontsize=14)
ani_ax_gdp.set_ylabel("GDP", fontsize=14)
ani_ax_gdp.set_xticks(country_index)
ani_ax_gdp.set_xticklabels(["" + i for i in countries], fontsize=8)

lower = min([min(i) for i in gdps]) * 0.9
upper = max([max(i) for i in gdps]) * 1.1
ani_ax_gdp.set_ylim(lower, upper)
plt.show()
```



You can find an animation in the same diretory of this fie.

4 Program a function for computing the histogram of a data set of N dimension. The function should include the definition/setting of (1) number of bins, (2) width(s) of bin(s), (3) frequencies of each bin, and (4) the dimension of the input data. Test the function using a dataset and visualize the result using bar plot. Submit your Python code and test data.

In [ ]: