Rio 2016 Olympics Data Analysis

April 10, 2018

```
In [86]: #import statements for visualization and data analysis
         import pandas as pd
         import numpy as np
         from pandas import DataFrame as df
         import matplotlib.pyplot as plt
         from datetime import datetime
         from datetime import date
         import seaborn as sns
         import math
         import random
         from math import sqrt
         from scipy import stats
         from sklearn.svm import SVC
         from sklearn.model_selection import GridSearchCV
         import warnings
         warnings.filterwarnings("ignore")
         from mpl_toolkits.mplot3d import Axes3D
```

0.1 Initialize Data

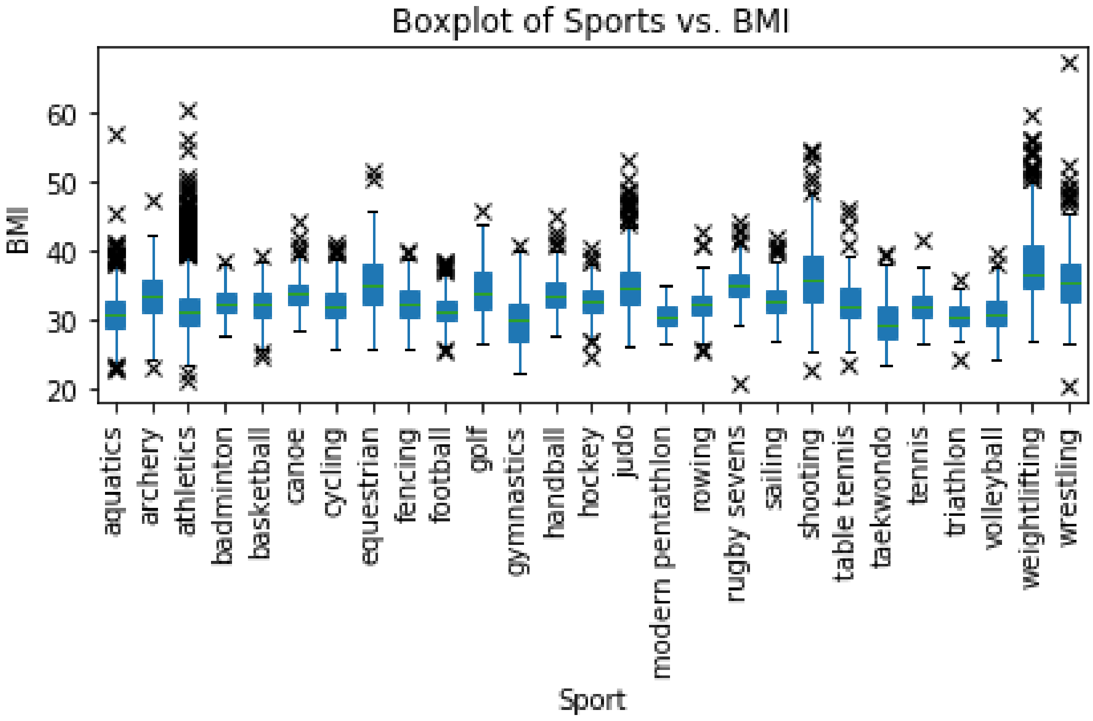
Import 'athletes.csv' and store the data in a DataFrame

```
In [87]: ath_data = pd.read_csv('athletes.csv', index_col=0)
         dobs = ath_data['dob'].values
         #calculates the person's age
         ages = []
         for i in range(dobs.size):
             dob = datetime.strptime((dobs[i]), '%m/%d/%Y')
             age = int((date.today()-dob.date()).days/365)
             ages.append(age)
         #appends the list of age to the existing DataFrame
         ath_data['age']=ages
         #appends the total amount of medals earned from an athlete to the DataFrame
         col_list=['gold', 'silver', 'bronze']
         ath_data['totals'] = ath_data[col_list].sum(axis=1)

         col_list = ['height', 'weight', 'age', 'sex']
         physical=ath_data[col_list]
         physical = physical[physical.height.notnull()]
         physical = physical[physical.weight.notnull()]
         physical['BMI'] = (physical[col_list].sum(axis=1))/(physical['height'])**2

         col_list = ['height', 'weight']
         physical2 = physical
         physical2['sport'] = ath_data['sport']
```

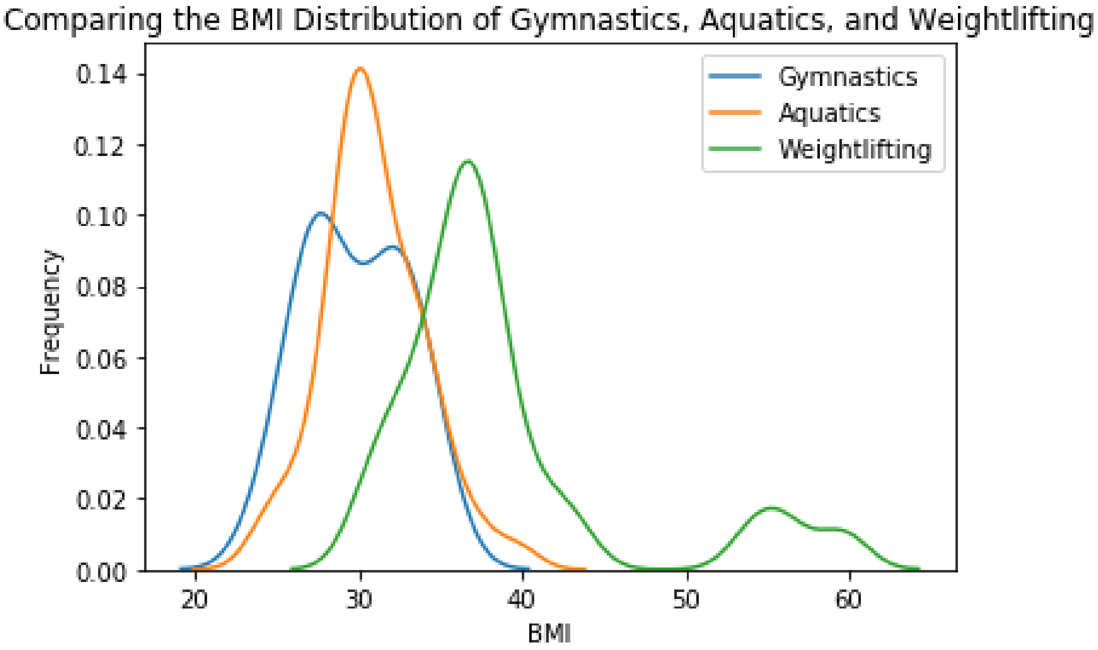
1 Investigation 1: Comparing Mean BMI Across Gymnastics, Aquatics, and Weightlifting

Is the mean height/weight distribution (i.e BMI) equal across aquatics, gymnastics, and weightlifting? If not, which sports have different means?

```
In [41]: #plots boxplot of the BMI distribution by sports
         plot = physical.boxplot(column = 'BMI', by='sport',  patch_artist=True,
             flierprops=dict(marker='x', color='cyan'))
         plot.grid(False)
         plot.set_xlabel('Sport')
         plot.set_ylabel('BMI')
         plt.title("Boxplot of Sports vs. BMI")
         plt.suptitle("")
         plt.xticks(rotation=90)
         plt.tight_layout()
         plt.show()
```



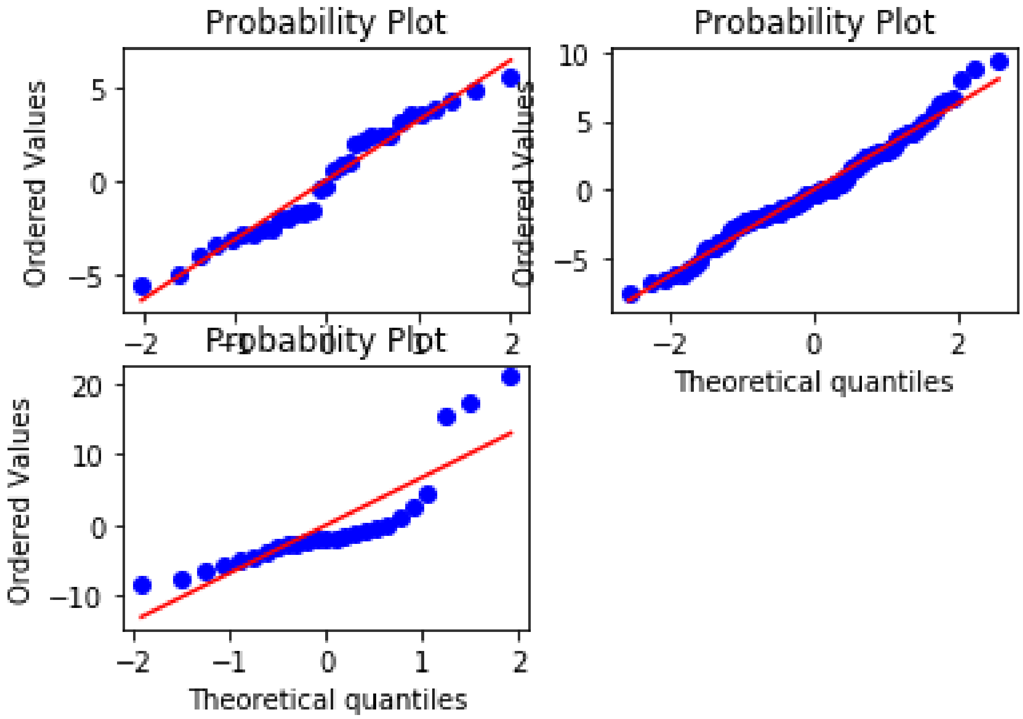
```
In [83]: #pulls respective Series of aq/gy/wl athletes of their BMIs
         gy = physical[physical['sport']=='gymnastics']['BMI']
         aq = physical[physical['sport']=='aquatics']['BMI']
         wl = physical[physical['sport']=='weightlifting']['BMI']
         sampleintgy = [random.randint(0, gy.shape[0]-1) for i in range(int(.1*gy.shape[0]))]
         sampleintaq = [random.randint(0, aq.shape[0]-1) for i in range(int(.1*aq.shape[0]))]
         sampleintwl = [random.randint(0, wl.shape[0]-1) for i in range(int(.1*wl.shape[0]))]
         gy = gy.iloc[sampleintgy]
         aq = aq.iloc[sampleintaq]
         wl = wl.iloc[sampleintwl]
         #density plot of BMI distribution
         sns.distplot(gy, hist=False, label='Gymnastics')
         sns.distplot(aq, hist=False, label='Aquatics')
         sns.distplot(wl, hist=False, label='Weightlifting')
         plt.xlabel('BMI')
         plt.ylabel('Frequency')
         plt.title("Comparing the BMI Distribution of Gymnastics, Aquatics, and Weightlifting"
         plt.show()
```



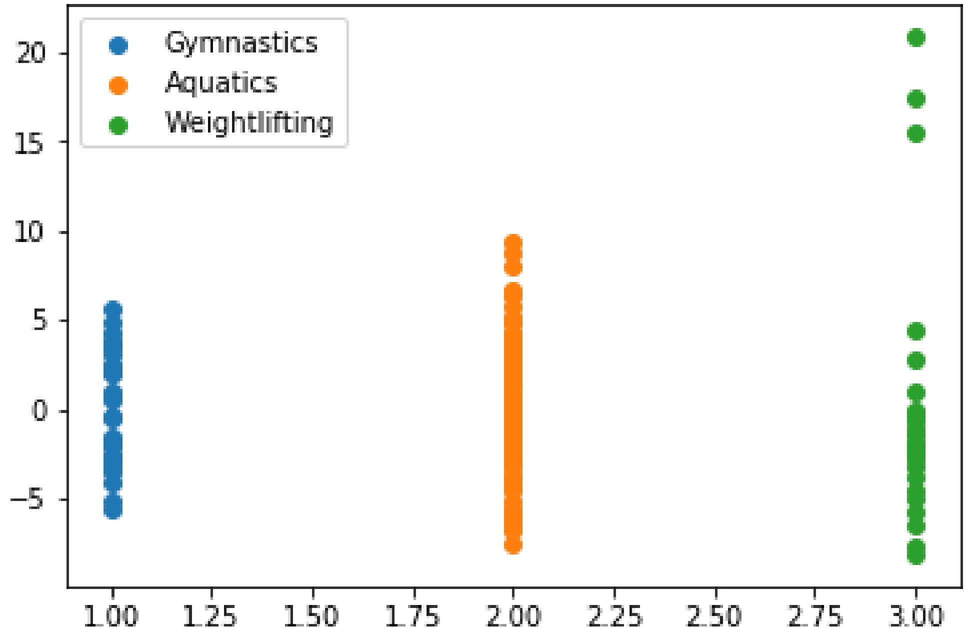
```
In [84]: print("n1 =", len(sampleintgy))
         print("n2 =", len(sampleintaq))
         print("n3 =", len(sampleintwl))
         n1 = len(sampleintgy)
         n2 = len(sampleintaq)
         n3 = len(sampleintwl)

         gy_sum = sum(gy)
         gy_sample_sq = gy**2
         gy_sum_sq = sum(gy_sample_sq)
         gy_avg = gy.mean()

         aq_sum = sum(aq)
         aq_sample_sq = aq**2
         aq_sum_sq = sum(aq_sample_sq)
         aq_avg = aq.mean()

         wl_sum = sum(wl)
         wl_sample_sq = wl**2
         wl_sum_sq = sum(wl_sample_sq)
         wl_avg = wl.mean()

         total_sum = gy_sum+aq_sum+wl_sum
```

```
n1 = 31
n2 = 139
n3 = 25
```

```
In [85]: gy_residuals = gy-gy_avg
         aq_residuals = aq-aq_avg
         wl_residuals = wl-wl_avg
         ax1 = plt.subplot(221)
         stats.probplot(gy_residuals, plot=plt)
         ax2 = plt.subplot(222)
         stats.probplot(aq_residuals, plot=plt)
         ax3 = plt.subplot(223)
         stats.probplot(wl_residuals, plot=plt)
         plt.show()
```



```
In [86]: x = [1 for i in range(gy_residuals.shape[0])]
         gym = plt.scatter(x, gy_residuals.values, label='Gymnastics')
         x = [2 for i in range(aq_residuals.shape[0])]
         aqu = plt.scatter(x, aq_residuals.values, label='Aquatics')
         x = [3 for i in range(wl_residuals.shape[0])]
         wei = plt.scatter(x, wl_residuals.values, label='Weightlifting')
         plt.legend(handles=[gym, aqu, wei])
         plt.show()
```



```
In [87]: ss_total = (gy_sum_sq+aq_sum_sq+wl_sum_sq) - ((total_sum)**2)/(n1+n2+n3)
         print("SS_total =", ss_total)
         ss_treat = ((gy_sum**2)/n1+(aq_sum**2)/n2+(wl_sum**2)/n3) - ((total_sum)**2)/(n1+n2+n
         print("SS_treatments =", ss_treat)
         ss_error = ss_total-ss_treat
         print("SS_error =", ss_error)

         ms_treat = ss_treat/2
         print("MS_treatments =", ms_treat)
         ms_error = ss_error/((n1+n2+n3)-2)
         print("MS_error =", ms_error)
         F0 = ms_treat/ms_error
         print("Test statistic =", F0)
```

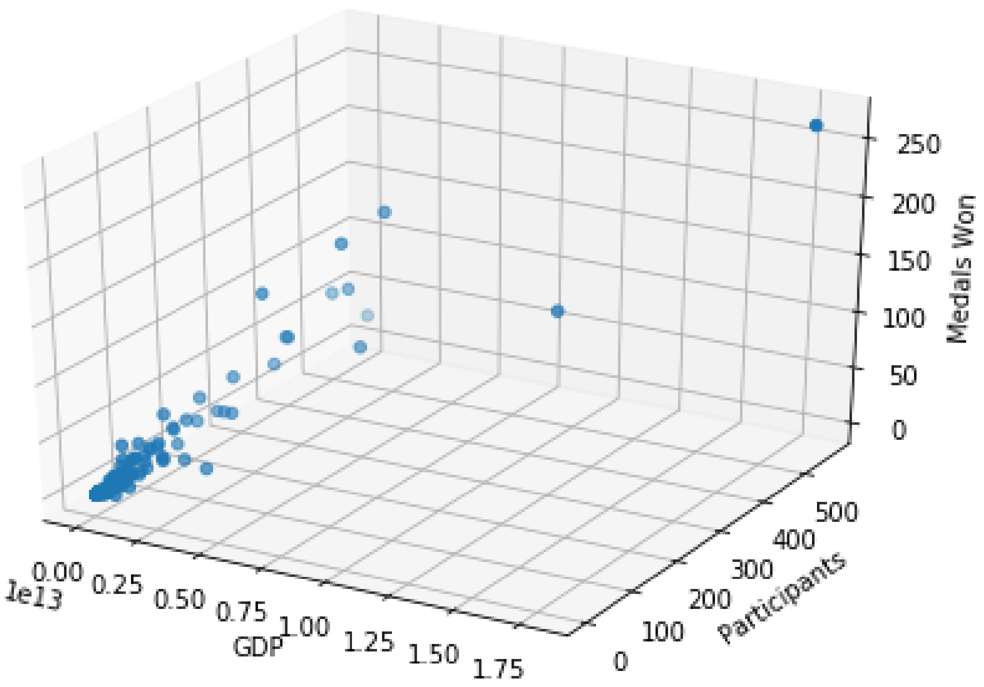
```
SS_total = 4374.6962813
SS_treatments = 1431.95234487
SS_error = 2942.74393643
MS_treatments = 715.976172435
MS_error = 15.247377909
Test statistic = 46.957331071
```

```
In [88]: print("Gymnastics Average BMI:", gy_avg)
         print("Aquatics Average BMI:", aq_avg)
         print("Weightlifting Average BMI:", wl_avg)
         LSD_gyaq = 1.972*sqrt(ms_error*(1/n1+1/n2))
         gyaq_diff = abs(gy_avg-aq_avg)
         print("Mean difference of average BMI between Gymnastics and Aquatics is", gyaq_diff)
         print("LSD of Gymnastics and Aquatics is", LSD_gyaq)

         LSD_gywl = 1.972*sqrt(ms_error*(1/n1+1/n3))
         gywl_diff = abs(gy_avg-wl_avg)
         print("Mean difference of average BMI between Gymnastics and Weightlifting is", gywl_
         print("LSD of Gymnastics and Weightlifting is", LSD_gywl)


         LSD_aqwl = 1.972*sqrt(ms_error*(1/n2+1/n3))
         aqwl_diff = abs(aq_avg-wl_avg)
         print("Mean difference of average BMI between Aquatics and Weightlifting is", aqwl_di
         print("LSD of Aquatics and Weightlifting is", LSD_aqwl)
```

```
Gymnastics Average BMI: 29.7958607882
Aquatics Average BMI: 30.9436728076
Weightlifting Average BMI: 38.7449305116
Mean difference of average BMI between Gymnastics and Aquatics is 1.14781201936
LSD of Gymnastics and Aquatics is 1.5294690971104137
Mean difference of average BMI between Gymnastics and Weightlifting is 8.9490697234
LSD of Gymnastics and Weightlifting is 2.069891852634907
Mean difference of average BMI between Aquatics and Weightlifting is 7.80125770404
LSD of Aquatics and Weightlifting is 1.6728192304487266
```

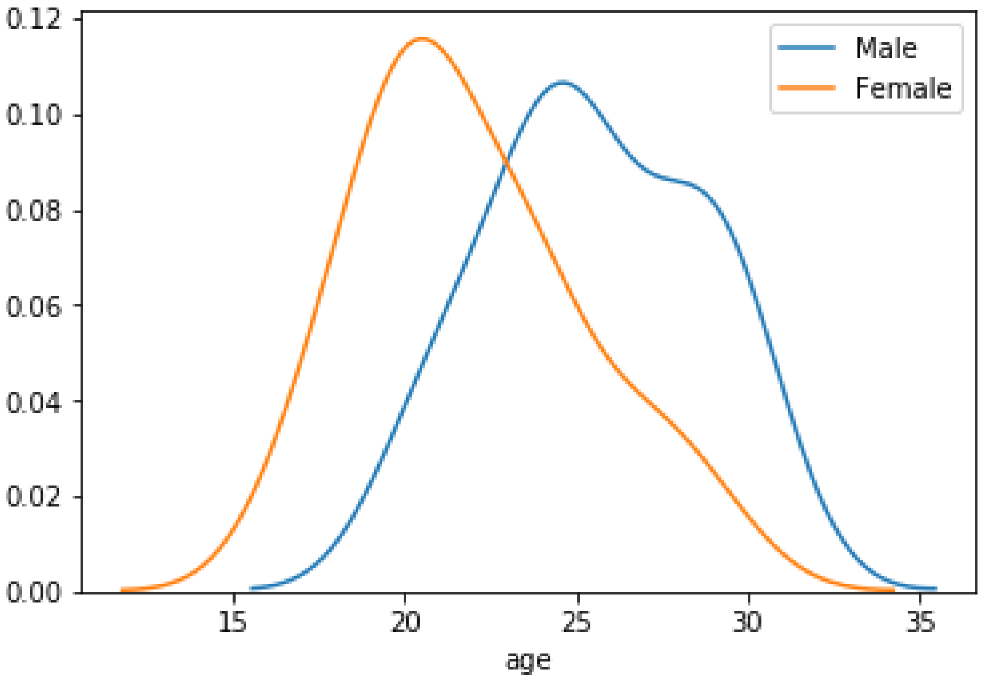
1.1 INVESTIGATION 2: Determining Relationship Between GDP and Medals Won

```
In [85]: ath_data_copy = ath_data
         country_data = pd.read_csv('countries.csv', index_col=0)
         country_data['gdp']=country_data['gdp_per_capita']*country_data['population']
         country_data = country_data.drop('gdp_per_capita', 1)
         country_data = country_data.rename(index = str, columns={"code":"nationality"})
         ath_data_copy = ath_data_copy.merge(country_data, on='nationality', how='left')
         ath_data_copy = ath_data_copy.sort_values(by='gdp')
         ath_data_copy = ath_data_copy[ath_data_copy.gdp.notnull()]
         gdp_data = ath_data_copy.groupby(by="gdp")['totals'].agg(['count', 'sum'])

         fig = plt.figure()
         ax = Axes3D(fig)
         ax.scatter(xs=gdp_data.index, ys=gdp_data['count'].values, zs=gdp_data['sum'].values,
         ax.set_xlabel('GDP')
         ax.set_ylabel('Participants')
         ax.set_zlabel('Medals Won')
         plt.show()
```



2 INVESTIGATION 3: Comparing Mean Age of Male/Female Medalists in Gymnastics

```
In [64]: gymnastics = ath_data[ath_data['sport']=='gymnastics']
         gymnastics_m = gymnastics[gymnastics['sex']=='male']
         gymnastics_f = gymnastics[gymnastics['sex']=='female']
         sampleintgy_m = [random.randint(0, gymnastics_m.shape[0]-1) for i in
             range(int(.5*gymnastics_m.shape[0]))]
         sampleintgy_f = [random.randint(0, gymnastics_f.shape[0]-1) for i in
             range(int(.5*gymnastics_f.shape[0]))]
         gymnastics_m = gymnastics_m.iloc[sampleintgy_m]
         gymnastics_f = gymnastics_f.iloc[sampleintgy_f]
         gymnastics_winners_m = gymnastics_m[gymnastics_m['totals']>0]['age']
         gymnastics_winners_f = gymnastics_f[gymnastics_f['totals']>0]['age']
         sns.distplot(gymnastics_winners_m, hist=False, label='Male')
         sns.distplot(gymnastics_winners_f, hist=False, label='Female')
         plt.show()
```

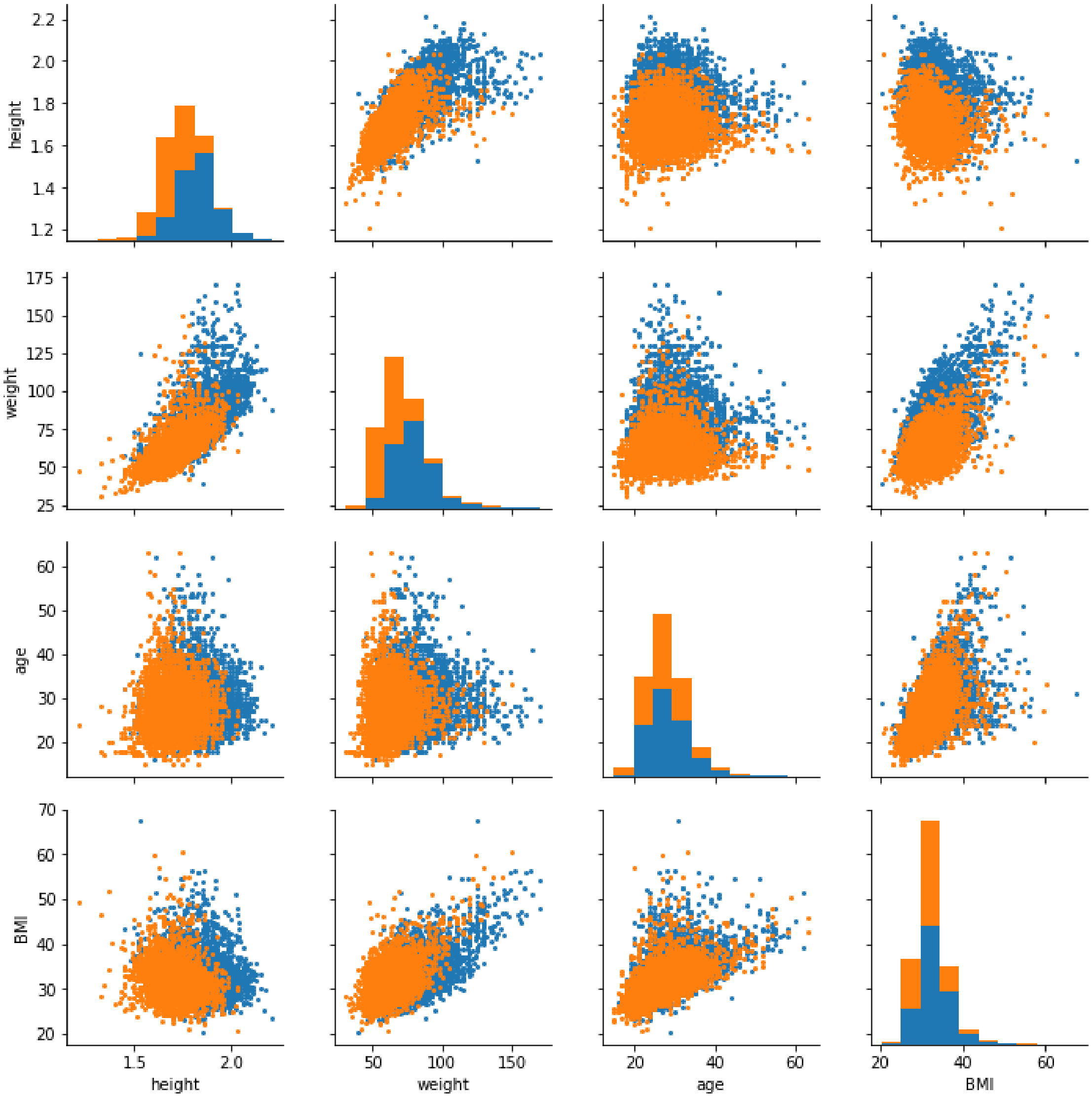


```
In [65]: gymnastics = ath_data[ath_data['sport']=='gymnastics']
         gymnastics_m = gymnastics[gymnastics['sex']=='male']
         gymnastics_f = gymnastics[gymnastics['sex']=='female']
         sampleintgy_m = [random.randint(0, gymnastics_m.shape[0]-1) for i in
             range(int(.5*gymnastics_m.shape[0]))]
         sampleintgy_f = [random.randint(0, gymnastics_f.shape[0]-1) for i in
             range(int(.5*gymnastics_f.shape[0]))]
         gymnastics_m = gymnastics_m.iloc[sampleintgy_m]
         gymnastics_f = gymnastics_f.iloc[sampleintgy_f]
         gymnastics_winners_m = gymnastics_m[gymnastics_m['totals']>0]['age']
         gymnastics_winners_f = gymnastics_f[gymnastics_f['totals']>0]['age']
         print("Average age of male Gymnastics medalists:", gymnastics_winners_m.mean())
         print("Average age of female Gymnastics medalists:", gymnastics_winners_f.mean())
         print("Standard deviation of ages of male Gymnastics medalists:", gymnastics_winners_
         print("Standard deviation of ages of female Gymnastics medalists:", gymnastics_winner
         print("n_male =", gymnastics_winners_m.shape[0])
         print("n_female =", gymnastics_winners_f.shape[0])
```

```
Average age of male Gymnastics medalists: 26.75
Average age of female Gymnastics medalists: 21.68
Standard deviation of ages of male Gymnastics medalists: 3.17280107581
Standard deviation of ages of female Gymnastics medalists: 3.49666507785
n_male = 16
n_female = 25
```

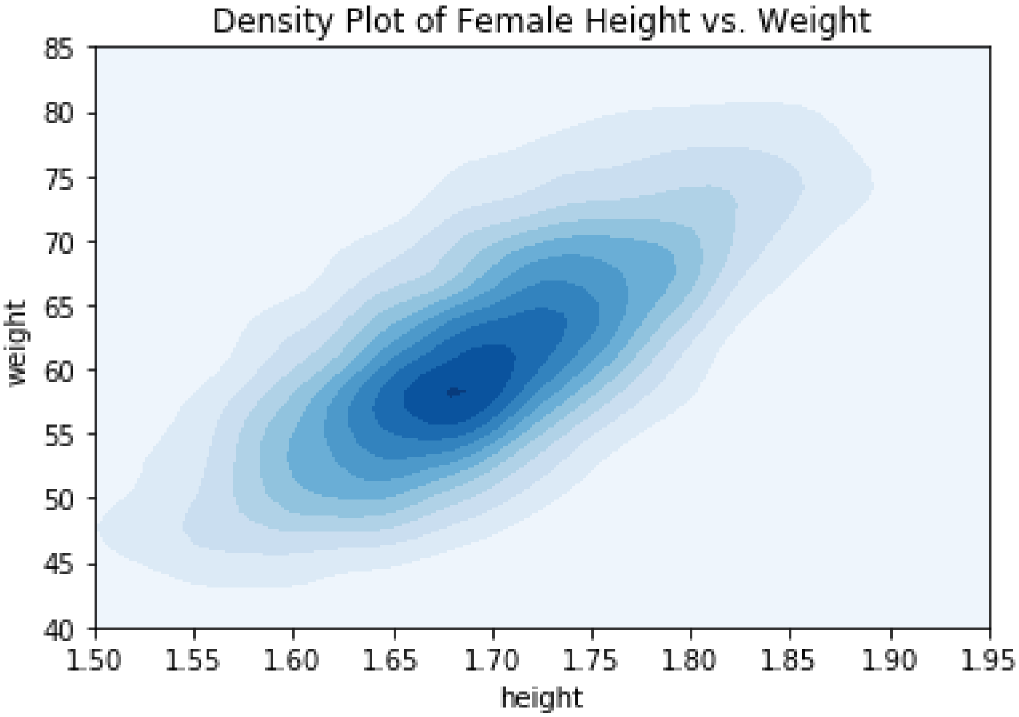
3 INVESTIGATION 4: Analyzing Heights/Weights Between Male/Female Athletes

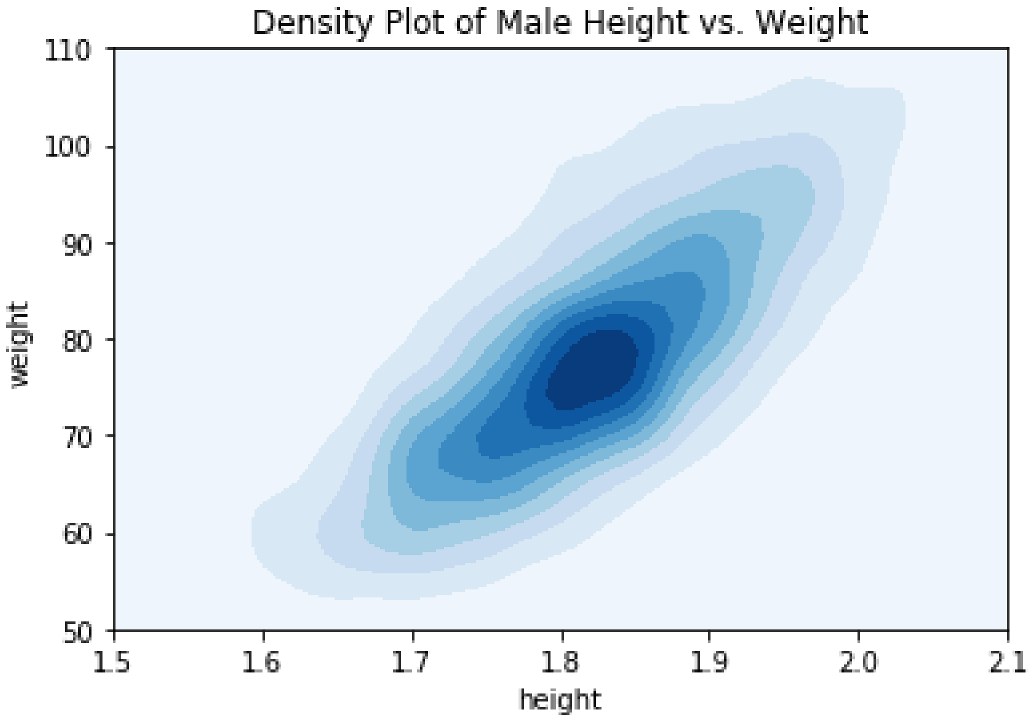
```
In [88]: #pulls the height/weight/age/sex of athletes; removes athletes with NaN
         #values in any height/weight columns
         #plots a scatterplot matrix to observe relationship between features
         plot = sns.PairGrid(physical, hue='sex')
         plot = plot.map_diag(plt.hist)
         plot = plot.map_offdiag(plt.scatter, s = 4)
         #sns.pairplot(physical)
         plt.show()
```



```
In [89]: #plots density plot of female/male height vs. weight
```

```
physical_f=physical[physical['sex']=='female']
physical_m=physical[physical['sex']=='male']
sns.kdeplot(physical_f.height, physical_f.weight, cmap="Blues", shade=True)
plt.xlim(1.5, 1.95)
plt.ylim(40, 85)
plt.title('Density Plot of Female Height vs. Weight')
plt.show()
sns.kdeplot(physical_m.height, physical_m.weight, cmap="Blues", shade=True)
plt.xlim(1.5, 2.1)
plt.title('Density Plot of Male Height vs. Weight')
plt.ylim(50,110)
plt.show()
```





```
In [138]: male = physical[physical['sex']=='male']
          female = physical[physical['sex']=='female']
          sampleint_m = [random.randint(0, male.shape[0]-1) for i in
              range(int(.5*male.shape[0]))]
          sampleint_f = [random.randint(0, female.shape[0]-1) for i in
              range(int(.5*female.shape[0]))]
          male = male.iloc[sampleint_m]
          female = female.iloc[sampleint_f]
          print("num_males = ", len(sampleint_m))
          print("num_females = ", len(sampleint_f))
          print("Male average height is ", male['height'].mean())
          print("Female average height is ", female['height'].mean())
          print("Male height standard deviation is ", male['height'].std())
          print("Female height standard deviation is ", female['height'].std())
```

```
num_males =  2931
num_females =  2498
Male average height is  1.82363357216
Female average height is  1.69742994396
Male height standard deviation is  0.102126874276
Female height standard deviation is  0.0879301891154
```

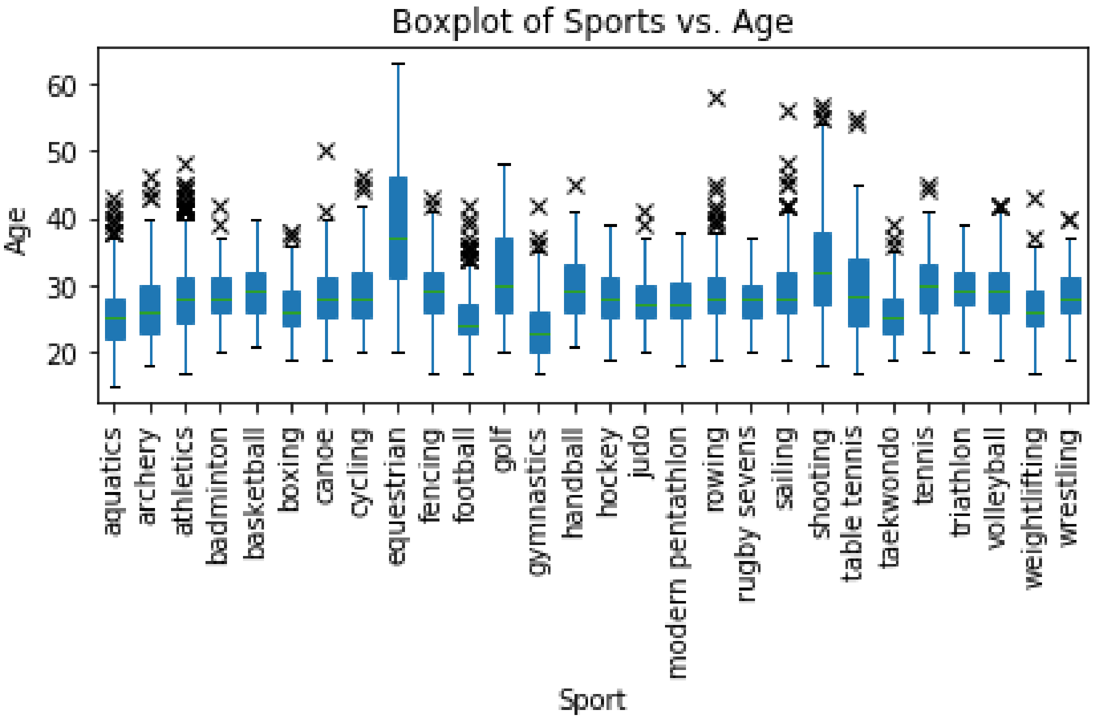
```
In [139]: print("Male average weight is ", male['weight'].mean())
          print("Female average weight is ", female['weight'].mean())
```

```
        print("Male weight standard deviation is ", male['weight'].std())
        print("Female weight standard deviation is ", female['weight'].std())
```

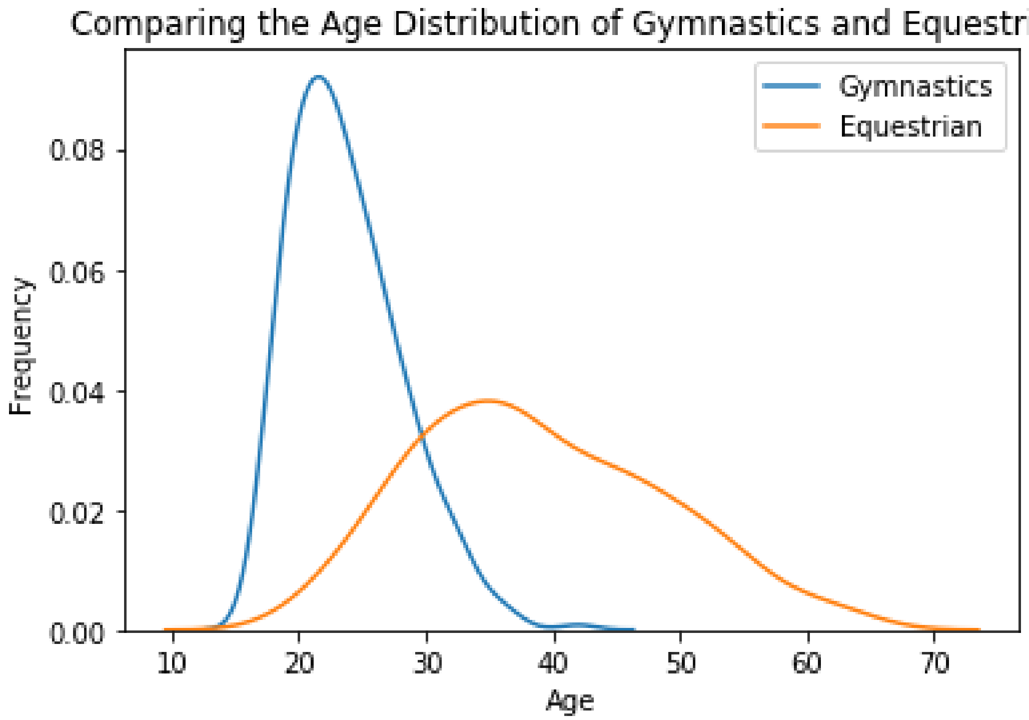
```
Male average weight is  79.9099283521
Female average weight is  62.5940752602
Male weight standard deviation is  15.646524449
Female weight standard deviation is  11.5126715776
```

4 INVESTIGATION 5: Comparing Variability of Ages of Equestrians and Gymnasts

```
In [52]: plot = ath_data.boxplot(column='age', by='sport', patch_artist=True,
             flierprops=dict(marker='x', color='cyan'))
         plot.grid(False)
         plot.set_xlabel('Sport')
         plot.set_ylabel('Age')
         plt.title("Boxplot of Sports vs. Age")

         plt.suptitle("")
         plt.xticks(rotation=90)
         plt.tight_layout()
         plt.show()
```



```
In [53]: #pulls respective Series of the equestrian/gymnastics athletes of their ages
         gymnastics_age = gymnastics['age']
         equestrian = ath_data[ath_data['sport']=='equestrian']
         equestrian_age = equestrian['age']
         #density plot of the age distribution
         sns.distplot(gymnastics_age, hist=False, label='Gymnastics')
         sns.distplot(equestrian_age, hist=False, label='Equestrian')
         plt.title("Comparing the Age Distribution of Gymnastics and Equestrian")
         plt.xlabel('Age')
         plt.ylabel('Frequency')
         plt.show()
```



```
In [54]: gymnastics_age = gymnastics['age']
         equestrian = ath_data[ath_data['sport']=='equestrian']
         equestrian_age = equestrian['age']
         sampleint_gy = [random.randint(0, gymnastics_age.shape[0]-1) for i in
             range(int(.5*gymnastics_age.shape[0]))]
         sampleint_eq = [random.randint(0, equestrian_age.shape[0]-1) for i in
             range(int(.5*equestrian_age.shape[0]))]
         gymnastics_age = gymnastics_age.iloc[sampleint_gy]
         equestrian_age = equestrian_age.iloc[sampleint_eq]

         print("n_gymnasts =", gymnastics_age.shape[0])
         print("n_equestrians =", equestrian_age.shape[0])
```

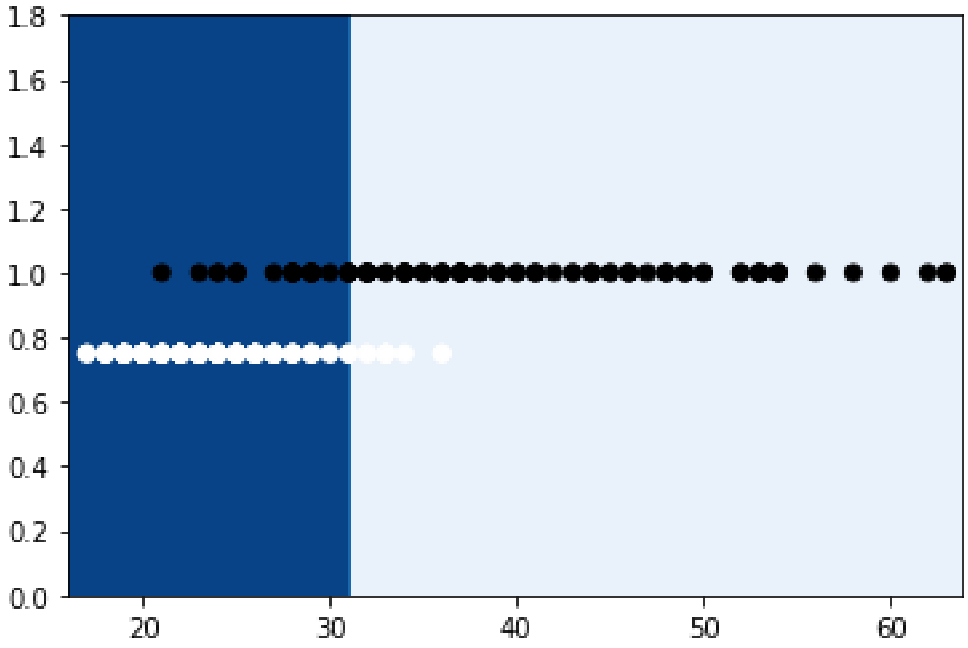
```
        print("Mean of gymnast ages is", gymnastics_age.mean())
        print("Mean of equestrian ages is", equestrian_age.mean())
        print("Standard deviation of gymnast ages is", gymnastics_age.std())
        print("Standard deviation of equestrian ages is", equestrian_age.std())
```

```
n_gymnasts = 162
n_equestrians = 111
Mean of gymnast ages is 23.5432098765
Mean of equestrian ages is 39.4234234234
Standard deviation of gymnast ages is 4.24424896301
Standard deviation of equestrian ages is 9.57415987252
```

```
In [63]: data = gymnastics.append(equestrian)
         sample_gym = [random.randint(0, gymnastics.shape[0]) for i in
             range(int(.8*gymnastics.shape[0]))]
         sample_test_gym = []
         for i in list(range(gymnastics.shape[0])):
             if i not in sample_gym:
                 sample_test_gym.append(i)
         train = data.iloc[sample_gym]
         ones = [1 for i in range(train.shape[0])]
         train['label'] = ones
         test = data.iloc[sample_test_gym]
         n1 = len(sample_test_gym)
         sample_eq = [gymnastics.shape[0]-1+random.randint(0, equestrian.shape[0]) for i in
             range(int(.8*equestrian.shape[0]))]
         sample_test_eq = []
         list_range = list(range(gymnastics.shape[0]-1, gymnastics.shape[0]-1+equestrian.shape
         for i in list_range:
             if i not in sample_eq:
                 sample_test_eq.append(i)
         eq_train = data.iloc[sample_eq]
         zeros = [0 for i in range(eq_train.shape[0])]
         eq_train['label']=zeros
         train = train.append(eq_train)
         eq_test = data.iloc[sample_test_eq]
         test = test.append(eq_test)

         param_grid = {'C': [1, 5, 10, 50], 'kernel': ['linear']},
         grid = GridSearchCV(SVC(), param_grid, cv=5, scoring='accuracy')
         grid.fit(train['age'].values.reshape(-1,1), train['label'])
         print("Best values of the parameters for the SVM are", grid.best_params_, "with an ac
         predict = grid.predict(test['age'].values.reshape(-1,1))

         h = 0.2
         x_min, x_max = min(test['age']) - 1, max(test['age']) + 1
         y_min, y_max = 0, 2
```

```
xx, yy = np.meshgrid(
    np.arange(x_min, x_max, h),
    np.arange(y_min, y_max, h))

Z = grid.predict(np.c_[xx.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap="Blues")
ones = [1 for i in range(test.shape[0])]
color= ['black' if l == 'equestrian' else 'white' for l in test['sport']]
vec = []
for i in list(range(test.shape[0])):
    if test['sport'].iloc[i]=='gymnastics':
        vec.append(0.75)
    else:
        vec.append(1)
plt.scatter(test['age'].values, vec, c=color, cmap=plt.cm.Paired)
plt.figure
plt.show()
```

```
Best values of the parameters for the SVM are {'C': 10, 'kernel': 'linear'} with an accuracy o
```



5 INVESTIGATION 6: Comparing Proportion of Male and Female Medalists

```
In [56]: males = ath_data[ath_data['sex']=='male']
         females = ath_data[ath_data['sex']=='female']
         sampleint_m = [random.randint(0, males.shape[0]-1) for i in
             range(int(.5*males.shape[0]))]
         sampleint_f = [random.randint(0, females.shape[0]-1) for i in
             range(int(.5*females.shape[0]))]
         males = males.iloc[sampleint_m]
         females = females.iloc[sampleint_f]
         male_winners = males[males['totals']>0]
         female_winners = females[females['totals']>0]
         print("n_males =", males.shape[0])
         print("n_females =", females.shape[0])
         print("Proportion of male medalists:", male_winners.shape[0]/males.shape[0])
         print("Proportion of female medalists:", female_winners.shape[0]/females.shape[0])
```

```
n_males = 3166
n_females = 2602
Proportion of male medalists: 0.15476942514213518
Proportion of female medalists: 0.16986933128362797
```