HW1_Lastname_Firstname

January 7, 2022

1 Homework 01: Data Cleaning and Exploratory Data Analysis

Name: Cal Brynestad

This assignment is due on Canvas by **6:00PM on Friday September 10**. Your solutions to theoretical questions should be done in Markdown directly below the associated question. Your solutions to computational questions should include any specified Python code and results as well as written commentary on your conclusions. Remember that you are encouraged to discuss the problems with your classmates, but **you must write all code and solutions on your own**.

NOTES:

- Any relevant data sets should be available in the Homework 01 assignment write-up on Canvas. To make life easier on the grader if they need to run your code, do not change the relative path names here. Instead, move the files around on your computer.
- If you're not familiar with typesetting math directly into Markdown then by all means, do your work on paper first and then typeset it later. Remember that there is a reference guide linked on Canvas on writing math in Markdown. **All** of your written commentary, justifications and mathematical work should be in Markdown.
- Because you can technically evaluate notebook cells is a non-linear order, it's a good idea to do Kernel → Restart & Run All as a check before submitting your solutions. That way if we need to run your code you will know that it will work as expected.
- It is **bad form** to make your reader interpret numerical output from your code. If a question asks you to compute some value from the data you should show your code output **AND** write a summary of the results in Markdown directly below your code.
- This probably goes without saying, but... For any question that asks you to calculate something, you **must show all work and justify your answers to receive credit**. Sparse or nonexistent work will receive sparse or nonexistent credit.

---

[1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
```

1.0.1 Problem 1 (10 Points)

Part A (5 of 10 points): There is a new co-ed engineering dorm on campus. It is called "Casa Matt & Trey", or CMT dorm. CMT dorm has 4 floors. Each floor has 24 students; 12 men and 12 women per floor. Furthermore the first floor is reserved for 18-year olds, the second floor for 19-year olds, the third floor for 20-year olds, and the 4th floor for 21-year olds. All 96 students are listed on a dorm roster.

The administration is interested in the student opinion of the new dorm. A survey is sent to 2 randomly chosen 18-year old women, 2 randomly chosen 19-year old women, 2 randomly chosen 20-year women and 2 randomly chosen 21-year old women. Then the same process is done for the various ages of men in the dorm all of whose names were listed on the roster.

Identify the following:

- the population
- the sample frame
- the sample
- the type of sample
- the variable of interest

Solution to Part A in this cell.

- The population is all students living in the CMT dorm.
- The sample frame is the 96 students on the dorm roster.
- The sample is 2 randomly chosen 18-year old women, 2 randomly chosen 19-year old women, 2 randomly chosen 20-year women and 2 randomly chosen 21-year old women, as well as 2 randomly chosen 18-year old men, 2 randomly chosen 19-year old men, 2 randomly chosen 20-year men and 2 randomly chosen 21-year old men.
- The type of sample is a stratified sample.
- The variable of interest is the student opinion of the new dorm.

Part B (5 of 10 points): After analyzing the results the administration decides it is wiser to perform a census sample of CMT dorm to determine the opinion of the new facility.

Answer the following:

- What is the population
- What is the sample frame
- What is the sample
- How many people were surveyed this time around
- If everyone honestly answered the survey, then is this data biased towards a particular age group or gender?

Solution to Part B in this cell. - The population is all students living in the CMT dorm. - The sample frame is the 96 students on the dorm roster. - The sample is everyone living in the CMT dorm, or the population. - 96 people were surveyed this time around. - No, there is an equivalent amount

of men and women in the dorm and an equivalent amount of 18, 19, 20 and 21 year-olds in the dorm. So, the data is not biased towards a particular age group or gender.

1.1 Problem 2 (30 points)

Let $x_1, x_2, \ldots, x_n$ be n observations of a variable of interest. Recall that the sample mean $\bar{x}$ and sample variance s^2 are given by

$$\bar{x} = \frac{1}{n}\sum_{k=1}^{n} x_k \quad \text{and} \quad s^2 = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \bar{x})^2$$

Recall that the standard deviation is typically found by taking the square root of the variance.

Notice that the above computation of the variance requires two passes over the data: one to compute the mean, and a second to subtract the mean from each observation and compute the sum of squares. Additionally, this manner of computing sample variance can lead to numerical problems and inefficiencies. For example, if the x's are "large" and the differences between them "small", computing the sample variance using the formula above requires computing a small number as the difference of two small numbers. Not good! Check out this optional wikipedia article about Loss of Significance to learn more.

It is often useful to be able to compute the variance and/or standard deviation in a single pass, inspecting each value x_k only once; for example, when the data are being collected without enough storage to keep all the values, or when costs of memory access dominate those of computation. In this problem (part B), we will compute the variance as the data arrives one at a time; we will thus not need to save the data for a second pass.

Part A (5 points): It is common to manipulate expressions algebraically to rewrite them in a more convenient way. We will practice this idea on the following identity. Please note that either side of the identity does not solve the numerical issues noted above. This is simply a mathematical exercise to see how to manipulate an expression and change its form.

Prove the following identity by algbraically manipulating one side to derive the other side.

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} = \frac{1}{n(n-1)}\left(n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right)$$

Solution to Part A in this cell.

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} = \frac{1}{n-1}\cdot\sum_{i=1}^{n}(x_i^2 - 2x_i\bar{x} + \bar{x}^2) = \frac{1}{n-1}(\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n}\bar{x}^2 - 2\bar{x}\sum_{i=1}^{n} x_i) = \frac{1}{n-1}(\sum_{i=1}^{n} x_i^2 + n\bar{x}^2 - 2n\bar{x}^2) =$$

Part B (10 points): Use the following algorithm to complete the function `running_variance` below that returns a running estimate of the variance. This function takes a list of integers as input and returns the variance of the list once we have iterated through the entire data set.

This algorithm provides a recursive way to compute the standard deviation and the mean as we make one pass through the data set.

Consider an arbitrary data set of n elements: $[x_1, x_2 ... x_i, ... x_{n-1}, x_n]$

We will let $\bar{x}_k$ represent the mean of the first k elements of this dataset. We will let s_k represent the standard deviation of the first k elements of this dataset.

- Initialize $\bar{x}_1 = x_1$ and $s_1 = 0$.
- Use the recursive formulas below to update the estimate of the mean and standard deviation as we iterate though our data set.

$$\bar{x}_k = \bar{x}_{k-1} + \frac{x_k - \bar{x}_{k-1}}{k}$$
$$s_k = s_{k-1} + (x_k - \bar{x}_{k-1}) \cdot (x_k - \bar{x}_k)$$

- For $2 \leq k \leq n$, $s^2 = \frac{s_k}{k-1}$

Note: Don't forget as you are coding that python is 0-indexed.

Solution to Part B below in code cell.

[2]:
```
def running_variance(x):
#type your code in here
    mean = x[0]
    std = 0
    k = 2

    for data in x[1:]:
        tempMean = mean
        mean = tempMean + ((data - tempMean) / k)

        std = std + (data - tempMean) * (data - mean)

        k+=1

    variance = std / (k-2)

    return variance

running_variance([1,2,3,4,5])
```

[2]: 2.5

Part C (3 points) Read in the 'Boxing.csv' datafile. Use the `running_variance` function from Part B to compute the variance of the 'Latino' column.

The 'Boxing.csv' dataset was obtained from a Harvard Dataverse site. You can read the website's description of the data set below: "All 704 male boxing champions among 8 weight divisions (Heavyweight, Light Heavyweight, Middleweight, Welterweight, Lightweight, Featherweight, Bantamweight, and Flyweight) from 1924-2011 were categorized according to his predominant or identified race; American Indian, Asian, Black, Latino, or White."

Solution to Part C in code cell below.

[3]:
```
#Type Code Here
file_path = 'Boxing.csv'

df = pd.read_csv(file_path)

latinoList = df['Latino'].to_list()
running_variance(latinoList)
```

[3]:
```
2.2272727272727275
```

Part D (8 points): Use the more "traditional" formula to compute variance. You may not use np.mean, np.std, or np.var here (or any other built in functions for these quantities.) You must complete the function `traditional_variance` below to compute variance.

Use the formula below.

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

Solution to Part D in code cell below.

[4]:
```
def traditional_variance(x):
#Put Code Here
    sum=0
    for data in x:
        sum+=data

    mean=sum/len(x)

    diffSq=0
    for data in x:
        diffSq+=(data-mean)**2

    variance_t=diffSq/(len(x)-1)

    return variance_t

traditional_variance([1,2,3,4,5])
```

[4]:
```
2.5
```

Part E (3 points) Use the `traditional_variance` function from Part D to again compute the variance of the 'Latino' column from the 'Boxing.csv' dataset.

Solution to Part E in code cell below.

[5]:
```
#Put Code Here
traditional_variance(latinoList)
```

[5]: 2.2272727272727275

Part F (1 point): Run the following cells to analyze each of your functions from **Part B** and **Part D**. You can read about `time` here. You can read about `tracemalloc` and monitoring memory usage in Python here.

Observe the results.

[6]:
```
# When this cell is executed, we time the running_variance function

from time import time
import numpy as np

xx = np.arange(1000000)

# timing for the running_variance function
tbeg = time()
running_variance(xx)
tend = time()

print('took {} seconds to calculate a running estimate of variance'.
 ↪format(tend-tbeg))
```

```
took 5.283790111541748 seconds to calculate a running estimate of variance
```

[7]:
```
# When this cell is executed, we time the traditional_variance function

tbeg = time()
traditional_variance(xx)
tend = time()

print('took {} seconds to calculate a direct computation of variance'.
 ↪format(tend-tbeg))
```

```
took 2.0435400009155273 seconds to calculate a direct computation of variance
```

[8]:
```
# When this cell is executed, we investigate memory usage of the␣
 ↪running_variance function.

import tracemalloc

tracemalloc.start()
running_variance(xx)
current, peak = tracemalloc.get_traced_memory()
print(f"Current memory usage is {current / 10**6}MB; Peak was {peak / 10**6}MB")
tracemalloc.stop()
```

```
Current memory usage is 0.000781MB; Peak was 0.011211MB
```

[9]:
```
# When this cell is executed, we investigate memory usage of the␣
 ↪traditional_variance function.

tracemalloc.start()
traditional_variance(xx)
current, peak = tracemalloc.get_traced_memory()
print(f"Current memory usage is {current / 10**6}MB; Peak was {peak / 10**6}MB")
tracemalloc.stop()
```

```
Current memory usage is 0.001582MB; Peak was 0.012012MB
```

Solution to Part F in this cell. What did you observe in the results? The running_variance function took longer to complete than the traditional_variance function did. The traditional_variance function is about 3 seconds faster. However, the current memory usage of running_variance is about 2 times smaller than the current memory usage of the traditional_variance function.

1.1.1 Problem 3 (30 points)

In this bit of data analysis we are going to look at the world of Boxing. Boxing is a sport that holds no bias towards competitors whether they are rich or poor, blue collar or white collar, big or small, professor or student. In amateur boxing every boxer is matched by gender, age, weight, and ability; everyone is equal in the ring.

We will now analyze some professional boxing data. This dataset contains the race of every one of eight weight-category champions in mens professional boxing between 1924 and 2011.

Each value in the dataset represents the number of champions that were a member of the particular race listed at the top of each respective column. Your job is to figure out what story the data are telling.

As usual make sure you have imported the proper libraries.

Part A (2 points): Read in the "Boxing.csv" datafile as a pandas dataframe. Take a look at the dataframe. You should see the number of male boxers from 8 weight categories and their race starting in 1924.

First solution to Part A in the code cell below.

[10]:
```
#Put Code Here
df.head()
#Boxing.csv was read in earlier
```

[10]:

	Year	Black	Latino	Asian	White	American Indian
0	1924	0	0	1	7	0
1	1925	1	0	0	7	0
2	1926	1	0	0	7	0
3	1927	0	0	0	8	0
4	1928	0	0	0	8	0

Now take a look at the more modern data (use the dot function .tail()). These should be the years ending in 2011.

Second solution to Part A in the code cell below.

[11]:
```
#Put Code Here
df.tail()
```

[11]:

	Year	Black	Latino	Asian	White	American Indian
83	2007	2	1	3	2	0
84	2008	0	2	3	3	0
85	2009	1	1	4	2	0
86	2010	1	4	2	1	0
87	2011	1	3	3	1	0

Part B (5 points): Suppose you think you see a different distribution of races in 1924 than you do in 2011. To help verify these thoughts, lets visualize the races of the eight various weight class champions as the years progress.

We begin exploring the data by creating smaller data sets. Specifically, create new dataframes in 11-year increments beginning in 1924. So for example, the first smaller data set will be the dataframe containing data from 1924 up to and including 1934. The second such data set will contain data from 1935 up to an including 1945, etc. Once you have finished generating these new smaller data sets, print "dfBoxing68_78" to the screen so that you (and we the graders) can verify your solution.

Note: Please follow the given naming convention for each new dataframe that you create:

- dfBoxing24_34
- dfBoxing35_45
- dfBoxing46_56
- dfBoxing57_67
- dfBoxing68_78
- dfBoxing79_89
- dfBoxing90_00
- dfBoxing01_11

Solution to Part B in the code cell below.

[12]:
```
#Put Code Here
dfBoxing24_34 = df[(df['Year'] >= 1924) & (df['Year'] <= 1934)]
dfBoxing35_45 = df[(df['Year'] >= 1935) & (df['Year'] <= 1945)]
dfBoxing46_56 = df[(df['Year'] >= 1946) & (df['Year'] <= 1956)]
dfBoxing57_67 = df[(df['Year'] >= 1957) & (df['Year'] <= 1967)]
dfBoxing68_78 = df[(df['Year'] >= 1968) & (df['Year'] <= 1978)]
dfBoxing79_89 = df[(df['Year'] >= 1979) & (df['Year'] <= 1989)]
dfBoxing90_00 = df[(df['Year'] >= 1990) & (df['Year'] <= 2000)]
dfBoxing01_11 = df[(df['Year'] >= 2001) & (df['Year'] <= 2011)]
print(dfBoxing68_78)
```

```
    Year  Black  Latino  Asian  White  American Indian
```

44	1968	4	2	1	1	0
45	1969	2	4	0	2	0
46	1970	2	2	2	2	0
47	1971	2	3	2	1	0
48	1972	2	5	1	0	0
49	1973	2	5	0	1	0
50	1974	2	5	1	0	0
51	1975	1	6	0	1	0
52	1976	1	7	0	0	0
53	1977	1	6	0	0	1
54	1978	1	5	0	1	1

Part C (8 points): Now lets create side-by-side blox plots for every one of these dataframes for each race.

In other words, each race will have 8 side-by-side box plots (one for each 11-year chunk of time) in an attempt to see the distribution of champions for that race as the years progress.

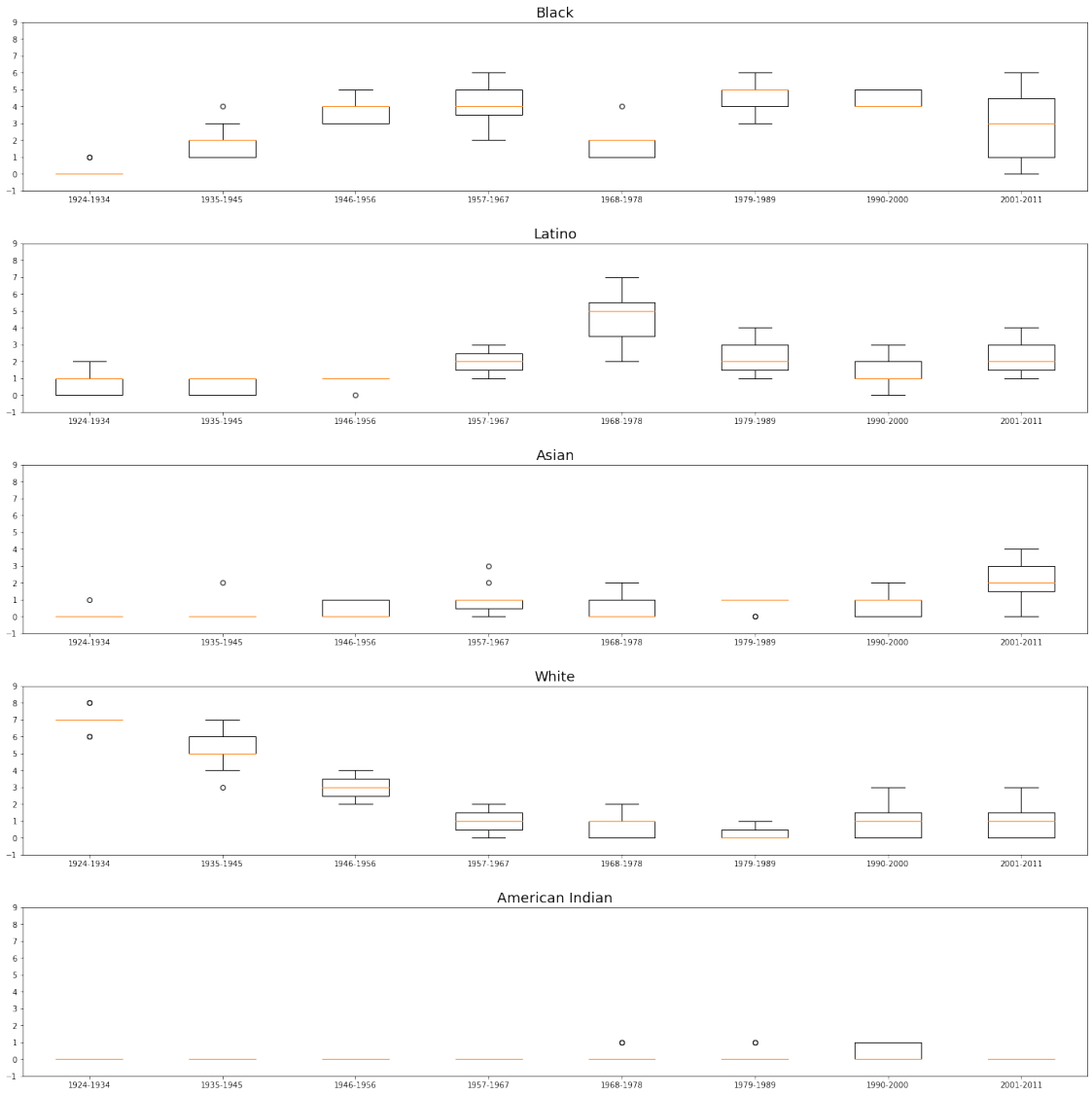
import matplotlib.pyplot in order to start creating some visualizations.

Plotting requirements: - Each group of boxplots must have a title that indicates which race's data is being displayed. - Customize the x-axis tick labels to show each time period. (i.e. 1924-1934, 1935-1945, etc) - Standardize the y axis limits to go from -1 to 9 for each figure. - Self-check: You should have a total of 5 figures, each with 8 boxplots.

Solution to Part C in the code cell below.

[13]:
```
#Put Code Here
%matplotlib inline
import matplotlib.pyplot as plt

fig, ax = plt.subplots(5, figsize=(20,20))

blackList = [dfBoxing24_34['Black'], dfBoxing35_45['Black'],␣
 ↪dfBoxing46_56['Black'], dfBoxing57_67['Black'], dfBoxing68_78['Black'],␣
 ↪dfBoxing79_89['Black'], dfBoxing90_00['Black'], dfBoxing01_11['Black']];
latinoList = [dfBoxing24_34['Latino'], dfBoxing35_45['Latino'],␣
 ↪dfBoxing46_56['Latino'], dfBoxing57_67['Latino'], dfBoxing68_78['Latino'],␣
 ↪dfBoxing79_89['Latino'], dfBoxing90_00['Latino'], dfBoxing01_11['Latino']];
asianList = [dfBoxing24_34['Asian'], dfBoxing35_45['Asian'],␣
 ↪dfBoxing46_56['Asian'], dfBoxing57_67['Asian'], dfBoxing68_78['Asian'],␣
 ↪dfBoxing79_89['Asian'], dfBoxing90_00['Asian'], dfBoxing01_11['Asian']];
whiteList = [dfBoxing24_34['White'], dfBoxing35_45['White'],␣
 ↪dfBoxing46_56['White'], dfBoxing57_67['White'], dfBoxing68_78['White'],␣
 ↪dfBoxing79_89['White'], dfBoxing90_00['White'], dfBoxing01_11['White']];
americanIndianList = [dfBoxing24_34['American Indian'], dfBoxing35_45['American␣
 ↪Indian'], dfBoxing46_56['American Indian'], dfBoxing57_67['American Indian'],␣
 ↪dfBoxing68_78['American Indian'], dfBoxing79_89['American Indian'],␣
 ↪dfBoxing90_00['American Indian'], dfBoxing01_11['American Indian']];
```

```
ax[0].boxplot(blackList);
ax[1].boxplot(latinoList);
ax[2].boxplot(asianList);
ax[3].boxplot(whiteList);
ax[4].boxplot(americanIndianList);

labels =␣
 ↪['1924-1934','1935-1945','1946-1956','1957-1967','1968-1978','1979-1989','1990-2000','2001-20
 ↪
ax[0].set_xticklabels(labels);
ax[1].set_xticklabels(labels);
ax[2].set_xticklabels(labels);
ax[3].set_xticklabels(labels);
ax[4].set_xticklabels(labels);

labelsy = [-1,0,1,2,3,4,5,6,7,8,9];
ax[0].set_yticks(labelsy);
ax[1].set_yticks(labelsy);
ax[2].set_yticks(labelsy);
ax[3].set_yticks(labelsy);
ax[4].set_yticks(labelsy);

ax[0].set_title('Black', fontsize=18);
ax[1].set_title('Latino', fontsize=18);
ax[2].set_title('Asian', fontsize=18);
ax[3].set_title('White', fontsize=18);
ax[4].set_title('American Indian', fontsize=18);

fig.tight_layout(pad=3.0)
```



Part D (3 points): Now we can interpret the graphics. Look at Q_2 in the progressing box-and-whisker plots for each race. Some rise, some fall, some are flat, and some oscillate.

In the context of this data, what does a fall in Q_2 indicate? In the context of this data, what does a flat Q_2 indicate? How would you categorize the 11-year span between 1968 and 1978 for Latino boxers?

During which 11-year span do the BNW plots show the largest IQR for Black boxers? In the context of this data, what does a large IQR indicate?

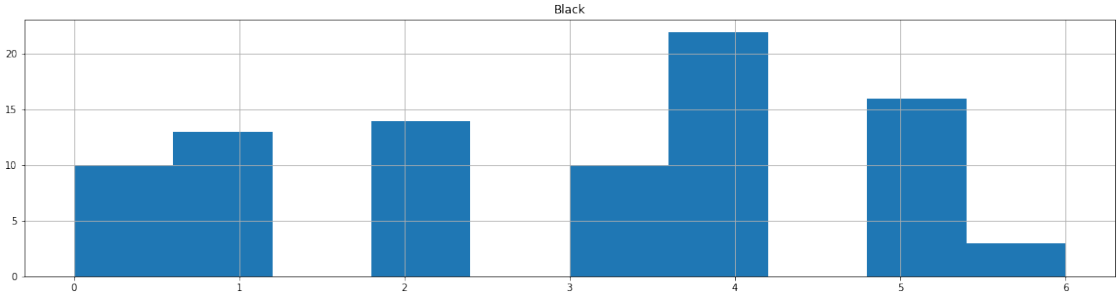
Solution to Part D in this cell. Q_2 is the central location of the data, so a fall in Q_2 most likely indicates a lower total number of champs for that time period. However, depending on how the data is skewed, this is not always true. A flat Q_2 indicates a fairly similar distribution of that data for each time period, this can be seen in the American Indian data and the Asian data. The data for the 11-year span between 1968 and 1978 for Latino boxers has a wide and fairly even distribution,

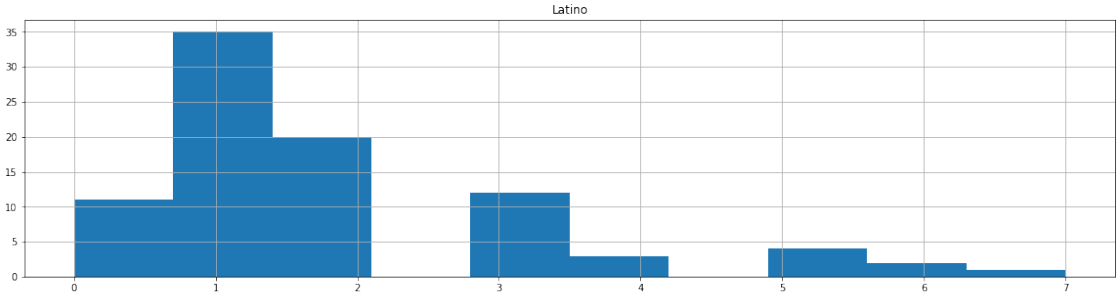
meaning some years the Latino boxers did very well and some years they didn't do as well, and most years fell some where in the middle. However, given the lower outlier is 2 championships, the years they didn't do as well weren't bad at all. 2001-2011. A large IQR indicates that the middle half of the data has a lot of variability. In the context of this data, that indicates a good but inconsistent 11-year span for championships. Given the data only spans 0-8, a large IQR means outliers aren't far away from Q_1 and Q_3, with Q_2 somewhere in the middle. So half of the the years were exceptional but the other half likely weren't nearly as good.

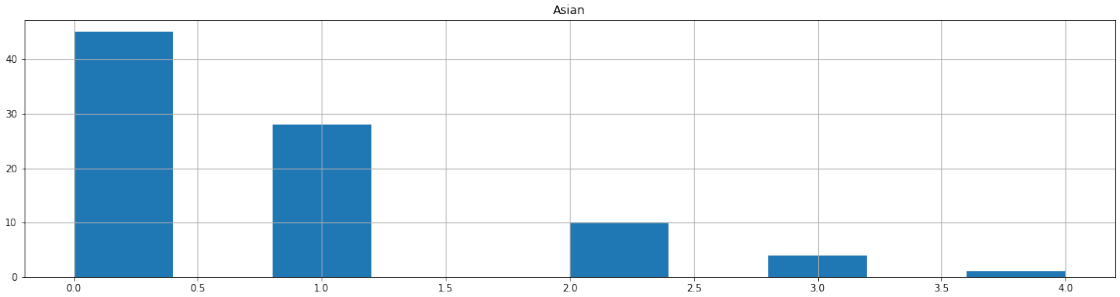
Part E (10 points): For each year between 1924 and 2011, how many times did each race have 8 champions, or 7 champions, or 6 champions,..., all the way down to how many times did each race have 0 champions? Create histograms to visualize an answer to this question. Note, you should have 5 seperate histograms.

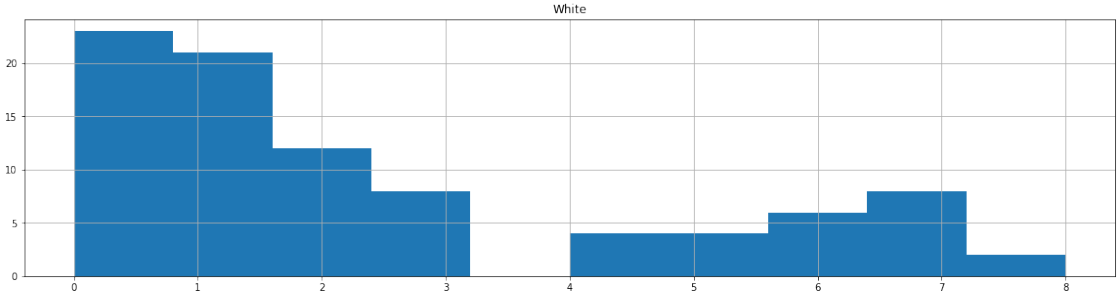
Solution to Part E in the cell below.

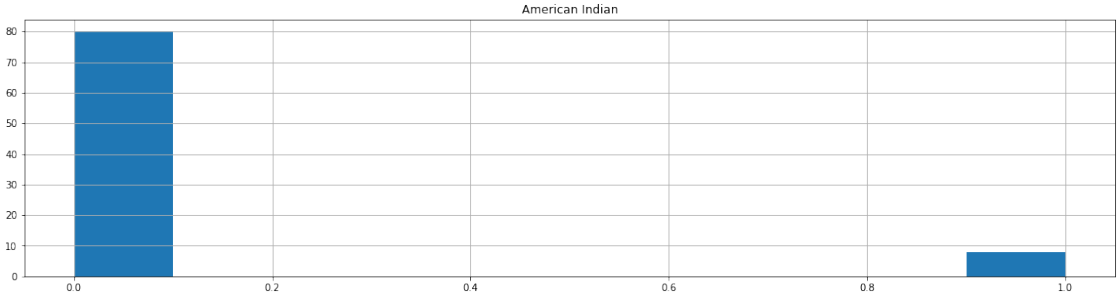
[14]:
```
# Put Code Here
#whiteList = dfBoxing24_34['White'].to_list()

fig, ax = plt.subplots(5, figsize=(20,30))

df.hist(column="Black", ax=ax[0]);
df.hist(column="Latino", ax=ax[1]);
df.hist(column="Asian", ax=ax[2]);
df.hist(column="White", ax=ax[3]);
df.hist(column="American Indian", ax=ax[4]);

#labels = [0,1,2,3,4,5,6,7,8];
#ax[0].set_xticks(labels);
#ax[1].set_xticks(labels);
#ax[2].set_xticks(labels);
#ax[3].set_xticks(labels);
#ax[4].set_xticks(labels);
```


Black
0
5
10
15
20
0
1
2
3
4
5
6


Latino
0
5
10
15
20
25
30
35
0
1
2
3
4
5
6
7


Asian
0
10
20
30
40
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


White
0
5
10
15
20
0
1
2
3
4
5
6
7
8


American Indian
0
10
20
30
40
50
60
70
80
0.0
0.2
0.4
0.6
0.8
1.0

Part F (2 points): Now we want to interpret the graphics. All of the histograms, except for one, are essentially right skewed, and the remaining histogram is nearly uniform.

Relative to this data, what does a right-skewed histogram indicate?

Relative to this data, what does a uniform histogram indicate?

Solution to Part F in this cell. A right-skewed histogram indicates most races have a larger number of years, or data points, where the total number of championships was a small number. For example, looking at the histogram for Latinos, the bulk of the data is around 0-2 championships for a year. The median is likely located in the set of data showing 0-2 championships. The data representing years where latinos won 3 or more championships is less and less as number of championships increases. A uniform histogram, as seen in the histogram for African Americans, indicates an even distribution of the number of championships won for the years between 1924-2011. In other words, African Americans won each number of championships close to the same number of times, at least comparatively to the other races in this case.

1.1.2 Problem 4 (16 points)

Consider the following 3 data sets:

`A=[8, 6, 7, 5, 3, 0, 9, 8, 6, 7, 5, 3, 0, 9]`

`B=[3, 14, 15, 9, 26, 5, 35, 8, 9, 7, 9]`

`C` is the random data set generated by using `np.random.randint(0,1000, size=200)`

For each data set, perform the following computations:

Part A (4 points): Compute and print the mean and standard deviation of the data set. Use built-in NumPy functions for calculation.

Solution to Part A in the code cell below.

[15]:
```
#Put Code Here
A=[8, 6, 7, 5, 3, 0, 9, 8, 6, 7, 5, 3, 0, 9]
meanA=np.mean(A)
stdA=np.std(A)
print("Mean of data set A is: " + str(meanA))
print("Standard Deviation of data set A is: " + str(stdA))

B=[3, 14, 15, 9, 26, 5, 35, 8, 9, 7, 9]
meanB=np.mean(B)
stdB=np.std(B)
print("Mean of data set B is: " + str(meanB))
print("Standard Deviation of data set B is: " + str(stdB))

C=np.random.randint(0,1000, size=200)
meanC=np.mean(C)
```

```
stdC=np.std(C)
print("Mean of data set C is: " + str(meanC))
print("Standard Deviation of data set C is: " + str(stdC))
```

```
Mean of data set A is: 5.428571428571429
Standard Deviation of data set A is: 2.8713930346059686
Mean of data set B is: 12.727272727272727
Standard Deviation of data set B is: 9.195758993746757
Mean of data set C is: 491.555
Standard Deviation of data set C is: 303.4689720136146
```

Part B (4 points): Compute and print the mean and standard deviation of the new data set formed by subtracting the original mean from each observation. Use built-in NumPy functions.

Solution to Part B in the code cell below.

[16]:
```
#Put Code Here
a=[]
for data in A:
    newData=data-meanA
    a.append(newData)

meanAa=np.mean(a)
stdAa=np.std(a)

print("Mean of the new data set A is: " + str(meanAa))
print("Standard Deviation of the new data set A is: " + str(stdAa))


b=[]
for data in B:
    newData=data-meanB
    b.append(newData)

meanBb=np.mean(b)
stdBb=np.std(b)

print("Mean of the new data set B is: " + str(meanBb))
print("Standard Deviation of the new data set B is: " + str(stdBb))


c=[]
for data in C:
    newData=data-meanC
    c.append(newData)

meanCc=np.mean(c)
stdCc=np.std(c)
```

```
print("Mean of the new data set C is: " + str(meanCc))
print("Standard Deviation of the new data set C is: " + str(stdCc))
```

```
Mean of the new data set A is: -2.5376526277146434e-16
Standard Deviation of the new data set A is: 2.8713930346059686
Mean of the new data set B is: 6.459479416000911e-16
Standard Deviation of the new data set B is: 9.195758993746757
Mean of the new data set C is: -1.8189894035458565e-14
Standard Deviation of the new data set C is: 303.4689720136146
```

Part C (4 points): Compute and print the mean and standard deviation of the new data set formed by subtracting the original mean from each observation and then dividing by the original standard deviation. Use built-in NumPy functions.

Solution to Part C in the code cell below.

[17]:
```
#Put Code Here
dataA=[]
for data in A:
    newData=(data-meanA)/stdA
    dataA.append(newData)

meanAaa=np.mean(dataA)
stdAaa=np.std(dataA)

print("Mean of the new data set A is: " + str(meanAaa))
print("Standard Deviation of the new data set A is: " + str(stdAaa))


dataB=[]
for data in B:
    newData=(data-meanB)/stdB
    dataB.append(newData)

meanBbb=np.mean(dataB)
stdBbb=np.std(dataB)

print("Mean of the new data set B is: " + str(meanBbb))
print("Standard Deviation of the new data set B is: " + str(stdBbb))


dataC=[]
for data in C:
    newData=(data-meanC)/stdC
    dataC.append(newData)

meanCcc=np.mean(dataC)
```

```
stdCcc=np.std(dataC)

print("Mean of the new data set C is: " + str(meanCcc))
print("Standard Deviation of the new data set C is: " + str(stdCcc))
```

```
Mean of the new data set A is: -6.344131569286608e-17
Standard Deviation of the new data set A is: 1.0
Mean of the new data set B is: 7.065055611250996e-17
Standard Deviation of the new data set B is: 1.0
Mean of the new data set C is: -1.7763568394002505e-17
Standard Deviation of the new data set C is: 1.0
```

Part D (4 points): What do you notice about the means? What do you notice about the standard deviations?

Solution to Part D in this cell. The means are all equal to 0. The standard deviations are all equal to 1.

1.1.3 Problem 5 (14 points)

Football teams are being matched for a competition. In an attempt to be efficient it was decided to match the teams based on a single descriptor of their weight using a measure of central tendancy.

The first team, team Red, sent in their weights: `R=[58, 63, 55, 58, 60, 62, 73, 51, 62, 72, 68]`

Team Blue-and-Yellow sent in their weights: `BY=[50, 165, 62, 151, 52, 160, 51, 163, 52, 159, 57]`

Team Shaq sent in their weights: Team Shaq: `Q = [55, 62, 62 ,65, 54, 64, 59, 51, 62, 51, 427]`

The teams that sent in their weights were not very accurate though. For instance, the coach for team shaq accidentilly listed his own weight with the players weights and Team Blue-Yellow thought the parents would play with the children, so they listed their own weights along with their childrens weights.

Part A1 (4 points) Teams must be no more than 11 pounds different in average weight in order to compete against each other. Compute and print the mean weight and the median weight of each team. You may use built-in mean and median functions to perform this computation.

Solution to Part A1 in the code cell below.

[18]:
```
#Put Code Here
R=[58, 63, 55, 58, 60, 62, 73, 51, 62, 72, 68]

BY=[50, 165, 62, 151, 52, 160, 51, 163, 52, 159, 57]

Q = [55, 62, 62 ,65, 54, 64, 59, 51, 62, 51, 427]
```

```
meanR=np.mean(R)
medianR=np.median(R)
print("Mean weight of team R is: " + str(meanR))
print("Median weight of team R is: " + str(medianR))

meanBY=np.mean(BY)
medianBY=np.median(BY)
print("Mean weight of team BY is: " + str(meanBY))
print("Median weight of team BY is: " + str(medianBY))

meanQ=np.mean(Q)
medianQ=np.median(Q)
print("Mean weight of team Q is: " + str(meanQ))
print("Median weight of team Q is: " + str(medianQ))
```

```
Mean weight of team R is: 62.0
Median weight of team R is: 62.0
Mean weight of team BY is: 102.0
Median weight of team BY is: 62.0
Mean weight of team Q is: 92.0
Median weight of team Q is: 62.0
```

Part A2 (2 points) According to the means, which teams can play each other? According to the medians, which teams can play each other?

Solution to Part A2 in this cell. According to the means, teams BY and R can play each other, as the difference of the teams two means is 102-92=10<11. Assuming the same principle can be applied to the medians, all of the teams can play against one another as the median weight of each team is 62.

Part B (2 points) List a pro and a con for computing the mean and a pro and a con for computing the median.

Solution to Part B in this cell. A pro for computing the mean is that the mean is the average, so if teams have a similar average weight then the total weights for each team will be about equal. Meaning even if one team has some bigger players, all the other players will be smaller than the other teams average weight of player, which should result in a fairly even matchup. A con for computing the mean is when this same concept is taken to the extreme. For example, team Q has only one HUGE player in Shaq and all the other players are smaller. And team BY is evenly distribtuted between small and medium players. So the mean of the two teams is within 11 pounds, but this matchup makes no sense due to the right skew of the mean weight of team Q. A pro for computing the median is that if teams have medians within 11 pounds of each other, then we know that 50% of the other players weigh less than the median weight and 50% of the players weigh more than the median, for each team, which should result in a good matchup for teams with normally distributed weights. A con for computing the median is when we have the teams in this problem. The median is really only one data point and tells us nothing about the other data points. For example, knowing the median weight of team Q doesn't alert us to the presence of Shaq in the team. When teams have much different weight distributions, they can still have the same median, so in this situation using the median to match up teams is a bad idea.

PART C (3 points) In a boxing competition, the team weights are also sent in for matching purposes. Teams are matched if their average weights are close together.

Two teams (CBC and DPD) each sent 5 boxers weights. Look closely at the individual weights.

Find the mean, median, and variance for each team. Again, use the built-in NumPy functions.

Team CBC weights = `[50, 54, 155, 200, 243]`

Team DPD wieghts = `[147, 150, 155, 156, 167]`

Solution to Part C in the code cell below.

[19]:
```
#Put Code Here
CBC=[50, 54, 155, 200, 243]
DPD=[147, 150, 155, 156, 167]

meanCBC=np.mean(CBC)
medianCBC=np.median(CBC)
varianceCBC=np.var(CBC, ddof=1)
print("The mean weight of team CBC is: " + str(meanCBC))
print("The median weight of team CBC is: " + str(medianCBC))
print("The variance of team CBC is: " + str(varianceCBC))
meanDPD=np.mean(DPD)
medianDPD=np.median(DPD)
varianceDPD=np.var(DPD, ddof=1)
print("The mean weight of team DPD is: " + str(meanDPD))
print("The median weight of team DPD is: " + str(medianDPD))
print("The variance of team DPD is: " + str(varianceDPD))
```

```
The mean weight of team CBC is: 140.4
The median weight of team CBC is: 155.0
The variance of team CBC is: 7482.3
The mean weight of team DPD is: 155.0
The median weight of team DPD is: 155.0
The variance of team DPD is: 58.5
```

Part D (3 points): What does the mean indicate about matching the teams?

What does the median indicate about matching the teams?

What does the variance say about matching the teams?

Solution to Part D in this cell. The means indicate that the average weight of team DPD is about 15 pounds greater than the average weight of team CBC. Using the metric that a weight class is 15 pounds, the means indicate that for each best matched fight between team CBC and team DPD, the DPD fighter would be about 1 weight class above the team CBC fighter The medians indicate that their would be atleasst one good fight between the teams between two 155 lb fighters, one from each team. And that both teams have 2 fighters who weigh above 155 lbs and 2 fighters who weigh below 155 lbs. If the two teams weights were similarly distributed this would indicate a good matchup but we know that this is not the case. The variance tells us that these two teams would not make a good matchup. Variance is a measure of how far a set of data is spread out from

the average value. Given team CBC has a variance of 7482.3 and team DPD has a variance of 58.5, we know that each fighter from each team will not be able to be matched up according to weight.

[ ]: