CS 306 : Data Analysis and Visualization
Lab 1: Temperature Data Analysis and Visualization

Name: Pratvi Shah
Student ID: 201801407

Code is copy pasted at the end of this pdf but for evaluation purposes colab file would be better as when the given code is transferred to other IDE the formatting might prevent the correct execution of the same. Also, to load the data, .csv file shall be present in the same folder as the code file.
Colab link:
https://colab.research.google.com/drive/1MAaR0Bl1xF8XM23gOghqkIqHD4I8-QeM?usp=sharing

All the questions are answered with respect to STATION GHCND:ASN00023887.

Q1) **After that get the results of several statistical measures (of Tmax and Tmin separately) like Maximum value, Minimum value , Mode value, Median and Average Temperature values using existing functions in excel.**

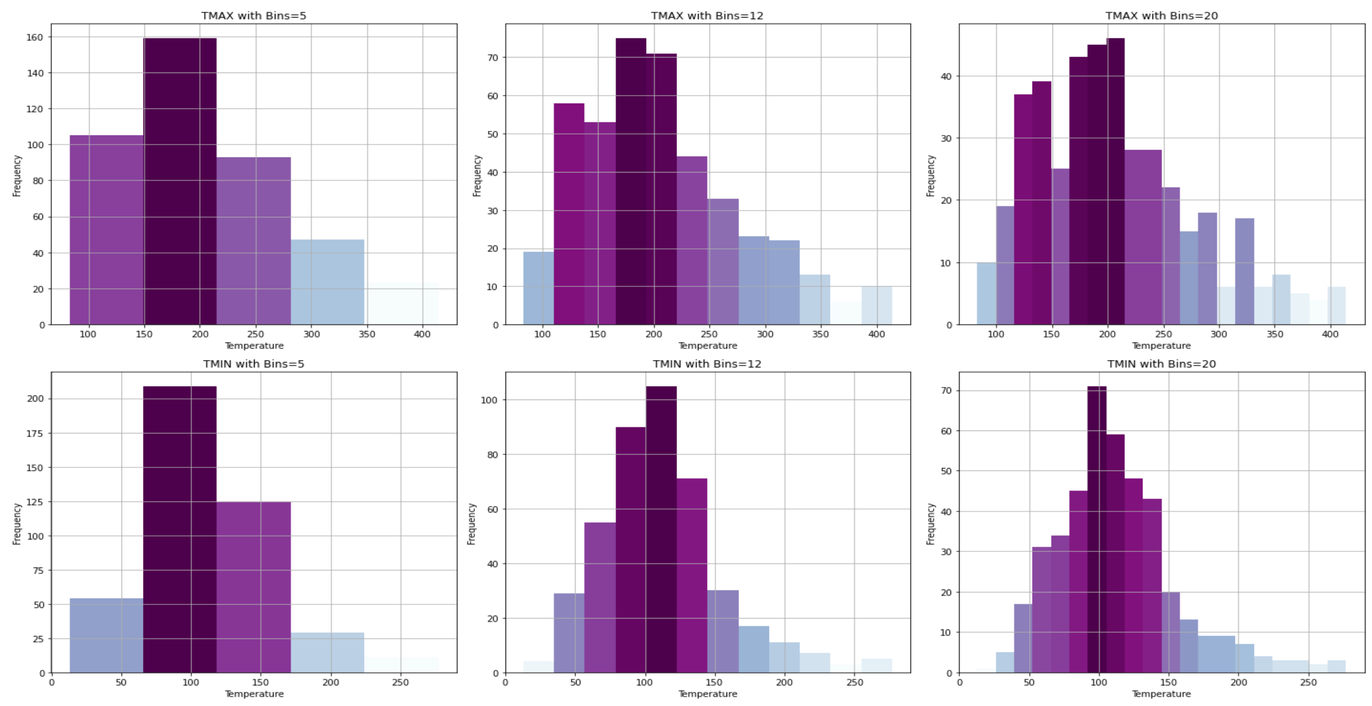
Ans) The values obtained are:
TMAX mean: 183.62(changed to 207.46 after removal of unacceptable data)
TMAX mode: 202
TMAX median: 197.0
TMAX min: -9999.0 (changed to 83.0 after changing negative values to mean)
TMAX max: 414.0

TMIN mean: 112.61
TMIN mode: 106 , 117
TMIN median: 107.0
TMIN min: 13.0
TMIN max: 277.0

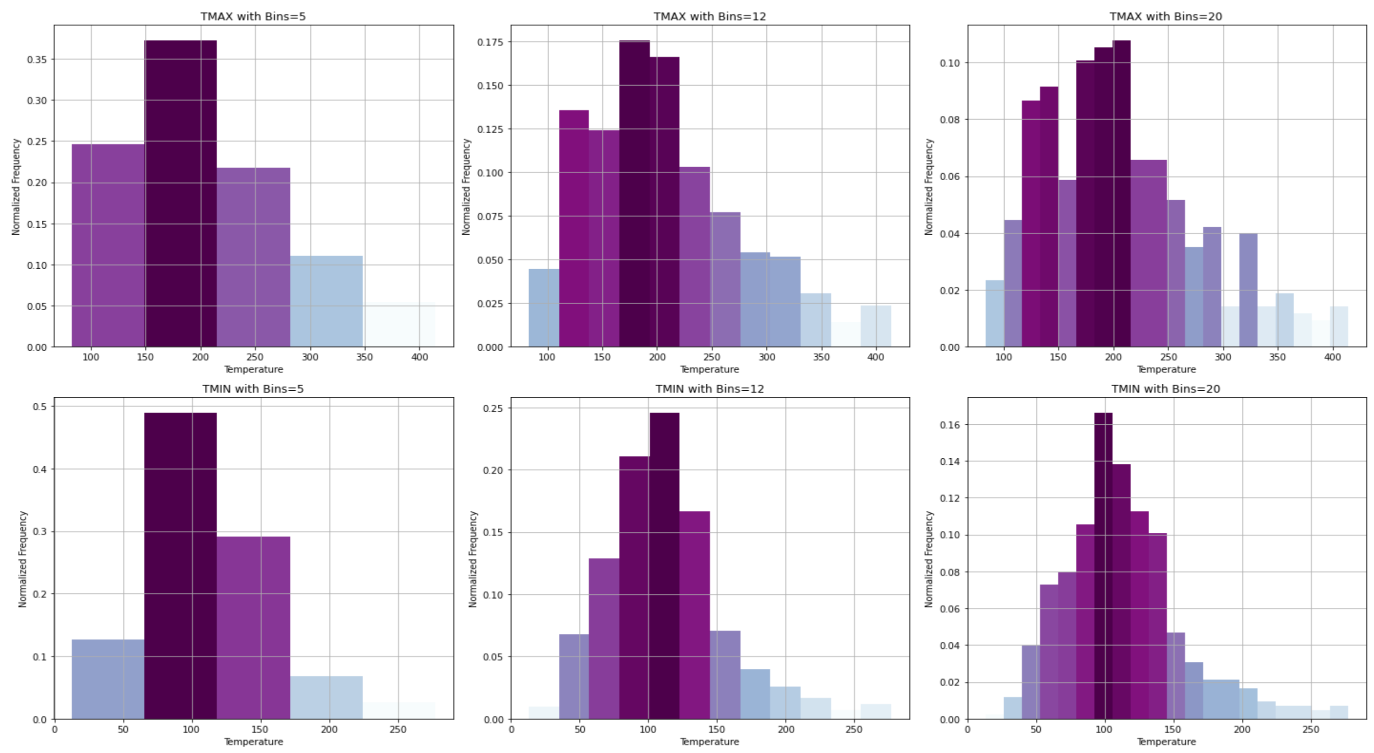
Q2) **Plot histogram for at least TWO different bin size. And also comment on both visualization.**

Ans) The values of frequency and the corresponding probability is presented in tabular form in the colab file.The following plots show the normalized frequency= frequency/total count of frequency. The summation of the heights of the bars of histogram (which represent the normalized frequency) will come out to be 1. The plots have Temperature on x-axis and Normalized frequency on y-axis. There are three plots for three different numbers of bins(5-12-20). First row shows the plots of TMAX and the second row has plots for TMIN.

These are the plots before normalizing the frequency:



Plots after normalizing the frequency:



OBSERVATIONS:

- The data of TMAX for the station chosen had some outliers (temperature<0 as the unit for temperature is assumed to be Kelvin) which had to be handled for better understanding/correct results of the dataset. Here, those values were replaced by mean of TMAX. As, there was only one such value so replacing it by any other measure of central tendency would not have been problematic and would have given us similar plots.
- We can definitely notice that when the bins are increased from 5 to 12 then to 20 the plots obtained are a better representation of the data given to us as it clearly brings out the nitty-gritties of the data.
- Increasing the number of bins gave a clear distinction between two peaks present in the TMAX data: one in range 130-150K and other around 200-220K. Whereas, the TMIN plot showed a single peak around 100K.

CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sbrn
from matplotlib import colors
import statistics
from matplotlib.ticker import PercentFormatter
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
#READING THE FILE
data=pd.read_csv('Temperature_2020.csv')
print('Imported data: \n',data.head())
print()
print()
print('Unique Stations : ',data['STATION'].unique())
print()
print()
#**CHOOSING STATION GHCND:ASN00023887**
df=data.loc[data['STATION'] == 'GHCND:ASN00023887']
print('Data of Station GHCND:ASN00023887:\n ',df.describe())
print()
print()
print('TMAX mean: ',df['TMAX'].mean())
print('TMIN mean: ',df['TMIN'].mean())
print('TMAX mode: ',statistics.mode(df['TMAX']))
print('TMIN mode: ',df['TMIN'].mode())
print('TMAX median: ',df['TMAX'].median())
print('TMIN median: ',df['TMIN'].median())
print()
print()
#COUNTING THE NEGATIVE VALUES
count_unacceptable_value=0;
for i in df['TMAX']:
  if(i<0):
    count_unacceptable_value+=1;
print('Count of negative numbers in TMAX:
',count_unacceptable_value)
df.dropna()
mx_mean=df['TMAX'].mean()
mn_mean=df['TMIN'].mean()
pd.set_option('mode.chained_assignment',None)
#REPLACING THE OUTLIER AND THE UNACCEPTED VALUES(LIKE NEGATIVE
FOR TEMPERATURE IN KELVIN) WITH THE MEAN OBTAINED
df.loc[df['TMAX']<0,'TMAX']=df['TMAX'].mean();
df.loc[df['TMIN']<0,'TMIN']=df['TMIN'].mean();
print('\n\nData after replacing negative values with mean:\n
',df.describe());
print()
print()
#**In the following cell the normalized frequency represents
the value of the ratio of frequency of that bin with respect to
total number of observations(=427). For the graphs in the
following cell the summation of all the heights of histogram
comes out to be 1.**

#FOR TMAX
tmax=df['TMAX']
# The leftmost and rightmost bin edges
first_edge, last_edge = tmax.min(), tmax.max()

weight=[1 ,len(tmax)];
ylabel=['Normalized Frequency' , 'Frequency']

for w in weight:
  n_equal_bin = 5  # Number of bins
  freqMX , binsMX= np.histogram(tmax,bins=n_equal_bin) #getting
the frequencies of various bins
  norm_freqMX=pd.Series(freqMX/freqMX.sum())#normalizing the
frequencies
  freqMX=pd.Series(freqMX)
  binsMX=pd.Series(binsMX)
  fig = plt.figure()
  fig, axs = plt.subplots(2, 3,figsize =(20, 12),tight_layout =
True)
  N, bins, patches=axs[0][0].hist(tmax,
bins=binsMX,weights=np.zeros_like(tmax) + 1. / w)
  bars = ((N**(1 / 5)) / N.max())
  bar = colors.Normalize(bars.min(), bars.max())
  for i, thisbar in zip(bars, patches):
      division = plt.cm.BuPu(bar(i))
      thisbar.set_facecolor(division)
  axs[0][0].set_xlabel('Temperature')
  axs[0][0].set_ylabel(ylabel[int(1./w)]);
  axs[0][0].set_title('TMAX with Bins=5')
  axs[0][0].grid()



  n_equal_bin = 12  # Number of bins
  freqMX2 , binsMX2= np.histogram(tmax,bins=n_equal_bin)
#getting the frequencies of various bins
  norm_freqMX2=pd.Series(freqMX2/freqMX2.sum())#normalizing the
frequencies
  freqMX2=pd.Series(freqMX2)
  binsMX2=pd.Series(binsMX2)
  N, bins, patches=axs[0][1].hist(tmax,
bins=binsMX2,weights=np.zeros_like(tmax) + 1. / w)
  bars = ((N**(1 / 5)) / N.max())
  bar = colors.Normalize(bars.min(), bars.max())
  for i, thisbar in zip(bars, patches):
      division = plt.cm.BuPu(bar(i))
      thisbar.set_facecolor(division)
  axs[0][1].set_xlabel('Temperature')
  axs[0][1].set_ylabel(ylabel[int(1./w)]);
  axs[0][1].set_title('TMAX with Bins=12')
  axs[0][1].grid()



  n_equal_bin = 20  # Number of bins
  freqMX3 , binsMX3= np.histogram(tmax,bins=n_equal_bin)
#getting the frequencies of various bins
  norm_freqMX3=pd.Series(freqMX3/freqMX3.sum())#normalizing the
frequencies
  freqMX3=pd.Series(freqMX3)
  binsMX3=pd.Series(binsMX3)
  N, bins, patches=axs[0][2].hist(tmax,
bins=binsMX3,weights=np.zeros_like(tmax) + 1. / w)
  bars = ((N**(1 / 5)) / N.max())
  bar = colors.Normalize(bars.min(), bars.max())
  for i, thisbar in zip(bars, patches):
      division = plt.cm.BuPu(bar(i))
      thisbar.set_facecolor(division)
  axs[0][2].set_xlabel('Temperature')
  axs[0][2].set_ylabel(ylabel[int(1./w)]);
  axs[0][2].set_title('TMAX with Bins=20')
  axs[0][2].grid()


  #FOR TMIN
  tmin=df['TMIN']
  # The leftmost and rightmost bin edges
  first_edge, last_edge = tmin.min(), tmin.max()
  n_equal_bin = 5  # Number of bins
  freqMI , binsMI= np.histogram(tmin,bins=n_equal_bin) #getting
the frequencies of various bins
  norm_freqMI=pd.Series(freqMI/freqMI.sum())#normalizing the
frequencies
  freqMI=pd.Series(freqMI)
  binsMI=pd.Series(binsMI)
  N, bins, patches=axs[1][0].hist(tmin,
bins=binsMI,weights=np.zeros_like(tmax) + 1. / w)
  bars = ((N**(1 / 5)) / N.max())
  bar = colors.Normalize(bars.min(), bars.max())
  for i, thisbar in zip(bars, patches):
      division = plt.cm.BuPu(bar(i))
      thisbar.set_facecolor(division)
  axs[1][0].set_xlabel('Temperature')
  axs[1][0].set_ylabel(ylabel[int(1./w)]);
  axs[1][0].set_title('TMIN with Bins=5')
  axs[1][0].grid()

  n_equal_bin = 12  # Number of bins
  freqMI2 , binsMI2= np.histogram(tmin,bins=n_equal_bin)
#getting the frequencies of various bins
  norm_freqMI2=pd.Series(freqMI2/freqMI2.sum())#normalizing the
frequencies
  freqMI2=pd.Series(freqMI2)
  binsMI2=pd.Series(binsMI2)
  N, bins, patches=axs[1][1].hist(tmin,
bins=binsMI2,weights=np.zeros_like(tmax) + 1. / w)
```

```
  bars = ((N**(1 / 5)) / N.max())
  bar = colors.Normalize(bars.min(), bars.max())
  for i, thisbar in zip(bars, patches):
      division = plt.cm.BuPu(bar(i))
      thisbar.set_facecolor(division)
  axs[1][1].set_xlabel('Temperature')
  axs[1][1].set_ylabel(ylabel[int(1./w)]);
  axs[1][1].set_title('TMIN with Bins=12')
  axs[1][1].grid()


  n_equal_bin = 20  # Number of bins
  freqMI3 , binsMI3= np.histogram(tmin,bins=n_equal_bin)
#getting the frequencies of various bins
  norm_freqMI3=pd.Series(freqMI3/freqMI3.sum())#normalizing the
frequencies
  freqMI3=pd.Series(freqMI3)
  binsMI3=pd.Series(binsMI3)
  N, bins, patches=axs[1][2].hist(tmin,
bins=binsMI3,weights=np.zeros_like(tmax) + 1. / w)
  bars = ((N**(1 / 5)) / N.max())
  bar = colors.Normalize(bars.min(), bars.max())
  for i, thisbar in zip(bars, patches):
      division = plt.cm.BuPu(bar(i))
      thisbar.set_facecolor(division)
  axs[1][2].set_xlabel('Temperature')
  axs[1][2].set_ylabel(ylabel[int(1./w)]);
  axs[1][2].set_title('TMIN with Bins=20')
  axs[1][2].grid()
```

#PRINTING VALUES OF BIN RANGE , FREQUENCY AND THE CORRESPONDING NORMALIZED FREQUENCY

```
print("\n\n-------------------------------BINS=
5--------------------------------")
frame={'BinsTMAX ':binsMX,'FreqTMAX':freqMX,'NormFreqTMAX':
norm_freqMX ,'BinsTMIN
':binsMI,'FreqTMIN':freqMI,'NormFreqTMIN': norm_freqMI }
norm_data=pd.DataFrame(frame)
print(norm_data)
print("\n\n-------------------------------BINS=
12--------------------------------")
frame={'BinsTMAX
':binsMX2,'FreqTMAX':freqMX2,'NormFrequencyTMAX': norm_freqMX2
,'BinsTMIN ':binsMI2,'FreqTMIN':freqMI2,'NormFreqTMIN':
norm_freqMI2 }
norm_data2=pd.DataFrame(frame)
print(norm_data2)
print("\n\n--------------------------------BINS=
20---------------------------------")
frame={'BinsTMAX
':binsMX3,'FreqTMAX':freqMX3,'NormFrequencyTMAX': norm_freqMX3
,'BinsTMIN ':binsMI3,'FreqTMIN':freqMI3,'NormFreqTMIN':
norm_freqMI3 }
norm_data2=pd.DataFrame(frame)
print(norm_data2)
```