**# --Temprature-Rainfall 1901-2016--**

**# 19F-0113 (CS), 19F-0171 (CS), 19F-0254 (CS), 19F-0931 (SE)**

**# Talha Ahmad, M. Talha Shehroze, Muhammad Farhan, Daniyal Ahmed**

* **frontend.py**

from tkinter import \*

from backend import \*

from PIL import ImageTk,Image

import tkinter.font as font

root = Tk()

root.title("Pakistan's Rain-Temp Data")

myFont = font.Font(family='Montserrat', size=10)

def tempBarButtonClick():

avgTMBarG()

def rainBarButtonClick():

avgRMBarG()

def multiBarButtonClick():

tempRainMultiBarChart()

def scatterTempButtonClick():

scatterTempGraphFunc()

def scatterTempDecadeButtonClick():

scatterTempGraphFuncDecade()

def scatterRainButtonClick():

scatterRainGraphFunc()

def scatterRainDecadeButtonClick():

scatterRainGraphFuncDecade()

def scatterMultiButtonClick():

scatterMultiGraphFunc()

def scatterMultiDecadeButtonClick():

scatterMultiGraphFuncDecade()

def boxPlotButtonClick():

boxPlotTempRain()

def boxPlotDecadeButtonClick():

boxPlotTempRainDecade()

def LRButtonClick():

linearReg()

def LRButton2Click():

linearReg2()

def textbuttonClick():

def openFile():

tf = open("dataset/statstext.txt", 'r')

data = tf.read()

txtarea.insert(END, data)

tf.close()

ws = Tk()

ws.title("Statistics")

ws.geometry("600x450")

ws['bg']='#fb0'

txtarea = Text(ws, width=70, height=22)

txtarea.pack(pady=20)

Button(ws,text="Calculate & Show Stats",command=openFile).pack(side=RIGHT, expand=True, fill=X, padx=20)

ws.mainloop()

def OnHoverScatter(event):

scatterButton.config(bg='black', fg='white')

def OnLeaveScatter(event):

scatterButton.config(bg='green', fg='black')

def OnHoverTempScatter(event):

scatterTempButton.config(bg='black', fg='white')

def OnLeaveTempScatter(event):

scatterTempButton.config(bg='green', fg='black')

def OnHoverRainScatter(event):

scatterRainButton.config(bg='black', fg='white')

def OnLeaveRainScatter(event):

scatterRainButton.config(bg='green', fg='black')

def OnHoverScatterDecade(event):

scatterButtonDecade.config(bg='black', fg='white')

def OnLeaveScatterDecade(event):

scatterButtonDecade.config(bg='green', fg='black')

def OnHoverTempScatterDecade(event):

scatterTempButtonDecade.config(bg='black', fg='white')

def OnLeaveTempScatterDecade(event):

scatterTempButtonDecade.config(bg='green', fg='black')

def OnHoverRainScatterDecade(event):

scatterRainButtonDecade.config(bg='black', fg='white')

def OnLeaveRainScatterDecade(event):

scatterRainButtonDecade.config(bg='green', fg='black')

def OnHoverBarTemp(event):

rainBarButton.config(bg='black', fg='white')

def OnLeaveBarTemp(event):

rainBarButton.config(bg='green', fg='black')

def OnHoverBarRain(event):

tempBarButton.config(bg='black', fg='white')

def OnLeaveBarRain(event):

tempBarButton.config(bg='green', fg='black')

def OnHoverMultiBar(event):

multiBarButton.config(bg='black', fg='white')

def OnLeaveMultiBar(event):

multiBarButton.config(bg='green', fg='black')

def OnHoverBoxPlot(event):

boxPlotButton.config(bg='black', fg='white')

def OnLeaveBoxPlot(event):

boxPlotButton.config(bg='green', fg='black')

def OnLeaveBoxPlotDecade(event):

boxPlotDecadeButton.config(bg='green', fg='black')

def OnHoverBoxPlotDecade(event):

boxPlotDecadeButton.config(bg='black', fg='white')

def OnHoverText(event):

TextButton.config(bg='black', fg='white')

def OnLeaveText(event):

TextButton.config(bg='green', fg='black')

def OnHoverLR(event):

LRButton.config(bg='black', fg='white')

def OnLeaveLR(event):

LRButton.config(bg='green', fg='black')

def OnHoverLR2(event):

LRButton2.config(bg='black', fg='white')

def OnLeaveLR2(event):

LRButton2.config(bg='green', fg='black')

backgroundImage = Image.open("images/Background.jpg")

resizedBackground = backgroundImage.resize((1920,1080),Image.ANTIALIAS)

newBackground = ImageTk.PhotoImage(resizedBackground)

backgroundLabel = Label( root, image =newBackground)

backgroundLabel.place(x = 0, y = 0)

multiBarButton = Button(root,text="Multi Bar Graph",width=42,height=2,command=multiBarButtonClick, bg='green', relief='groove')

multiBarButton.bind('<Enter>', OnHoverMultiBar)

multiBarButton.bind('<Leave>', OnLeaveMultiBar)

multiBarButton['font'] = myFont

multiBarButton.place(x=620, y= 260)

tempBarButton = Button(root,text="Temp Bar Graph",width=42,height=2,command=tempBarButtonClick, bg='green', relief='groove')

tempBarButton.bind('<Enter>', OnHoverBarRain)

tempBarButton.bind('<Leave>', OnLeaveBarRain)

tempBarButton['font'] = myFont

tempBarButton.place(x=420, y= 320)

rainBarButton = Button(root,text="Rain Bar Graph",width=42,height=2,command=rainBarButtonClick, bg='green', relief='groove')

rainBarButton.bind('<Leave>', OnLeaveBarTemp)

rainBarButton.bind('<Enter>', OnHoverBarTemp)

rainBarButton['font'] = myFont

rainBarButton.place(x=820, y= 320)

scatterTempButton = Button(root,text="Temp Line Plot by Month",width=42,height=2,command=scatterTempButtonClick, bg='green', relief='groove')

scatterTempButton.bind('<Enter>', OnHoverTempScatter)

scatterTempButton.bind('<Leave>', OnLeaveTempScatter)

scatterTempButton['font'] = myFont

scatterTempButton.place(x=420, y= 380)

scatterRainButton = Button(root,text="Rain Line Plot by Month",width=42,height=2,command=scatterRainButtonClick, bg='green', relief='groove')

scatterRainButton.bind('<Enter>', OnHoverRainScatter)

scatterRainButton.bind('<Leave>', OnLeaveRainScatter)

scatterRainButton['font'] = myFont

scatterRainButton.place(x=420, y= 440)

scatterButton = Button(root,text="Scatter Plot by Month",width=42,height=2,command=scatterMultiButtonClick, bg='green', relief='groove')

scatterButton.bind('<Enter>', OnHoverScatter)

scatterButton.bind('<Leave>', OnLeaveScatter)

scatterButton['font'] = myFont

scatterButton.place(x=420, y= 500)

scatterTempButtonDecade = Button(root,text="Temp Line Plot by Decade",width=42,height=2,command=scatterTempDecadeButtonClick, bg='green', relief='groove')

scatterTempButtonDecade.bind('<Enter>', OnHoverTempScatterDecade)

scatterTempButtonDecade.bind('<Leave>', OnLeaveTempScatterDecade)

scatterTempButtonDecade['font'] = myFont

scatterTempButtonDecade.place(x=820, y= 380)

scatterRainButtonDecade = Button(root,text="Rain Line Plot by Decade",width=42,height=2,command=scatterRainDecadeButtonClick, bg='green', relief='groove')

scatterRainButtonDecade.bind('<Enter>', OnHoverRainScatterDecade)

scatterRainButtonDecade.bind('<Leave>', OnLeaveRainScatterDecade)

scatterRainButtonDecade['font'] = myFont

scatterRainButtonDecade.place(x=820, y= 440)

scatterButtonDecade = Button(root,text="Scatter Plot by Decade",width=42,height=2,command=scatterMultiDecadeButtonClick, bg='green', relief='groove')

scatterButtonDecade.bind('<Enter>', OnHoverScatterDecade)

scatterButtonDecade.bind('<Leave>', OnLeaveScatterDecade)

scatterButtonDecade['font'] = myFont

scatterButtonDecade.place(x=820, y= 500)

boxPlotButton = Button(root,text="Box Plot by Month",width=42,height=2,command=boxPlotButtonClick, bg='green', relief='groove')

boxPlotButton.bind('<Enter>', OnHoverBoxPlot)

boxPlotButton.bind('<Leave>', OnLeaveBoxPlot)

boxPlotButton['font'] = myFont

boxPlotButton.place(x=420, y= 560)

boxPlotDecadeButton = Button(root,text="Box Plot by Decade",width=42,height=2,command=boxPlotDecadeButtonClick, bg='green', relief='groove')

boxPlotDecadeButton.bind('<Enter>', OnHoverBoxPlotDecade)

boxPlotDecadeButton.bind('<Leave>', OnLeaveBoxPlotDecade)

boxPlotDecadeButton['font'] = myFont

boxPlotDecadeButton.place(x=820, y= 560)

LRButton = Button(root,text="Linear Regression",width=42,height=2,command=LRButtonClick, bg='green', relief='groove')

LRButton.bind('<Enter>', OnHoverLR)

LRButton.bind('<Leave>', OnLeaveLR)

LRButton['font'] = myFont

LRButton.place(x=420, y= 620)

LRButton = Button(root,text="Linear Regression T o R",width=42,height=2,command=LRButtonClick, bg='green', relief='groove')

LRButton.bind('<Enter>', OnHoverLR)

LRButton.bind('<Leave>', OnLeaveLR)

LRButton['font'] = myFont

LRButton.place(x=420, y= 620)

LRButton2 = Button(root,text="Linear Regression R o T",width=42,height=2,command=LRButton2Click, bg='green', relief='groove')

LRButton2.bind('<Enter>', OnHoverLR2)

LRButton2.bind('<Leave>', OnLeaveLR2)

LRButton2['font'] = myFont

LRButton2.place(x=820, y= 620)

TextButton = Button(root,text="Quantitative Stats",width=42,height=2,command=textbuttonClick, bg='green', relief='groove')

TextButton.bind('<Enter>', OnHoverText)

TextButton.bind('<Leave>', OnLeaveText)

TextButton['font'] = myFont

TextButton.place(x=620, y= 680)

root.state('zoomed')

root.mainloop()

* **backend.py**

import csv

import math

from os import sep

import statistics

from scipy import stats

import numpy as np

import pandas as pd

from matplotlib import colors

from collections import Counter

import matplotlib.pyplot as plt

from matplotlib.ticker import PercentFormatter

# --Temprature-Rainfall 1901-2016--

# 19F-0113 (CS), 19F-0171 (CS), 19F-0254 (CS), 19F-0931 (SE)

# Talha Ahmad, M. Talha Shehroze, Muhammad Farhan, Daniyal Ahmed

# Reading Data from CSV.

dataset1 = pd.read\_csv(r'dataset\19012016.csv')

# Taking Mean of Monthly Temprature and Rainfall from 1901-2016

def meanTempRainFunc():

tempMean=dataset1['Temperature (Celsius)'].mean().round(3)

strTempMean=str(tempMean)

temp="->Average Monthly Temprature (Celsius) from 1901-2016 was: "

file.write(temp)

file.write(strTempMean)

file.write(" °C\n")

rainMean=dataset1['Rainfall (MM)'].mean().round(3)

strrainMean=str(rainMean)

temp0="->Average Monthly Rainfall (MM) from 1901-2016 was: "

file.write(temp0)

file.write(strrainMean)

file.write(" mm\n\n")

# Taking Median of Monthly Temprature and Rainfall from 1901-2016

def medianTempRainFunc():

tempMedian=dataset1['Temperature (Celsius)'].median()

strTempMedian=str(tempMedian)

temp="->Median Monthly Temprature (Celsius) from 1901-2016 was: "

file.write(temp)

file.write(strTempMedian)

file.write(" °C\n")

rainMedian=dataset1['Rainfall (MM)'].median()

strRainMedian=str(rainMedian)

temp0="->Median Monthly Rainfall (MM) from 1901-2016 was: "

file.write(temp0)

file.write(strRainMedian)

file.write(" mm\n\n")

# Descriptive Statistics of Dataset of Temprature and Rainfall from 1901-2016

def dataDescript():

describeData1=dataset1[['Temperature (Celsius)']].describe().round()

print(describeData1)

describeData2=dataset1[['Rainfall (MM)']].describe().round()

print(describeData2)

# Taking Minimum and Maximum of Monthly Temprature from 1901-2016

tempMin=dataset1['Temperature (Celsius)'].min()

tempMax=dataset1['Temperature (Celsius)'].max()

# Taking Minimum and Maximum Value of Monthly Rainfall from 1901-2016

rainMin=dataset1['Rainfall (MM)'].min()

rainMax=dataset1['Rainfall (MM)'].max()

# Taking Minimum and Maximum Row of Monthly Temprature from 1901-2016

def minMaxTempratureRow():

tempMinRow = dataset1.loc[dataset1['Temperature (Celsius)'] == tempMin]

temp="->Minimum Monthly Temprature (Celsius) from 1901-2016 was in:"

file.write(temp+'\n')

temp1=tempMinRow.to\_string(index=False)

file.write(temp1+'\n\n')

tempMaxRow = dataset1.loc[dataset1['Temperature (Celsius)'] == tempMax]

temp0="->Maximum Monthly Temprature (Celsius) from 1901-2016 was in:"

file.write(temp0+'\n')

temp2=tempMaxRow.to\_string(index=False)

file.write(temp2+'\n\n')

# Taking Minimum and Maximum Row of Monthly Rainfall from 1901-2016

def minMaxRainfallRow():

rainMinRow = dataset1.loc[dataset1['Rainfall (MM)'] == rainMin]

temp="->Minimum Monthly Rainfall (MM) from 1901-2016 was in:"

file.write(temp+'\n')

temp1=rainMinRow.to\_string(index=False)

file.write(temp1+'\n\n')

rainMaxRow = dataset1.loc[dataset1['Rainfall (MM)'] == rainMax]

temp0 ="->Maximum Monthly Rainfall (MM) from 1901-2016 was in:"

file.write(temp0+'\n')

temp2=rainMaxRow.to\_string(index=False)

file.write(temp2+'\n\n')

# Reading Data from CSV by Month.

janDataset = dataset1.loc[dataset1['Month'] == 'January']

febDataset = dataset1.loc[dataset1['Month'] == 'February']

marDataset = dataset1.loc[dataset1['Month'] == 'March']

aprDataset = dataset1.loc[dataset1['Month'] == 'April']

mayDataset = dataset1.loc[dataset1['Month'] == 'May']

junDataset = dataset1.loc[dataset1['Month'] == 'June']

julDataset = dataset1.loc[dataset1['Month'] == 'July']

augDataset = dataset1.loc[dataset1['Month'] == 'August']

sepDataset = dataset1.loc[dataset1['Month'] == 'September']

octDataset = dataset1.loc[dataset1['Month'] == 'October']

novDataset = dataset1.loc[dataset1['Month'] == 'November']

decDataset = dataset1.loc[dataset1['Month'] == 'December']

# Calculating Mean Temprature and Rain by Month.

janTempMean=janDataset['Temperature (Celsius)'].mean(); janRainMean=janDataset['Rainfall (MM)'].mean()

febTempMean=febDataset['Temperature (Celsius)'].mean(); febRainMean=febDataset['Rainfall (MM)'].mean()

marTempMean=marDataset['Temperature (Celsius)'].mean(); marRainMean=marDataset['Rainfall (MM)'].mean()

aprTempMean=aprDataset['Temperature (Celsius)'].mean(); aprRainMean=aprDataset['Rainfall (MM)'].mean()

mayTempMean=mayDataset['Temperature (Celsius)'].mean(); mayRainMean=mayDataset['Rainfall (MM)'].mean()

junTempMean=junDataset['Temperature (Celsius)'].mean(); junRainMean=junDataset['Rainfall (MM)'].mean()

julTempMean=julDataset['Temperature (Celsius)'].mean(); julRainMean=julDataset['Rainfall (MM)'].mean()

augTempMean=augDataset['Temperature (Celsius)'].mean(); augRainMean=augDataset['Rainfall (MM)'].mean()

sepTempMean=sepDataset['Temperature (Celsius)'].mean(); sepRainMean=sepDataset['Rainfall (MM)'].mean()

octTempMean=octDataset['Temperature (Celsius)'].mean(); octRainMean=octDataset['Rainfall (MM)'].mean()

novTempMean=novDataset['Temperature (Celsius)'].mean(); novRainMean=novDataset['Rainfall (MM)'].mean()

decTempMean=decDataset['Temperature (Celsius)'].mean(); decRainMean=decDataset['Rainfall (MM)'].mean()

# Reading Data from CSV by Decades.

dataset1903 = dataset1.loc[dataset1['Year'] == 1903]

dataset1913 = dataset1.loc[dataset1['Year'] == 1913]

dataset1923 = dataset1.loc[dataset1['Year'] == 1923]

dataset1933 = dataset1.loc[dataset1['Year'] == 1933]

dataset1943 = dataset1.loc[dataset1['Year'] == 1943]

dataset1953 = dataset1.loc[dataset1['Year'] == 1953]

dataset1963 = dataset1.loc[dataset1['Year'] == 1963]

dataset1973 = dataset1.loc[dataset1['Year'] == 1973]

dataset1983 = dataset1.loc[dataset1['Year'] == 1983]

dataset1993 = dataset1.loc[dataset1['Year'] == 1993]

dataset2003 = dataset1.loc[dataset1['Year'] == 2003]

dataset2013 = dataset1.loc[dataset1['Year'] == 2013]

# Calculating Mean Temprature and Rain by Decades.

TempMean1903=dataset1903['Temperature (Celsius)'].mean(); RainMean1903=dataset1903['Rainfall (MM)'].mean()

TempMean1913=dataset1913['Temperature (Celsius)'].mean(); RainMean1913=dataset1913['Rainfall (MM)'].mean()

TempMean1923=dataset1923['Temperature (Celsius)'].mean(); RainMean1923=dataset1923['Rainfall (MM)'].mean()

TempMean1933=dataset1933['Temperature (Celsius)'].mean(); RainMean1933=dataset1933['Rainfall (MM)'].mean()

TempMean1943=dataset1943['Temperature (Celsius)'].mean(); RainMean1943=dataset1943['Rainfall (MM)'].mean()

TempMean1953=dataset1953['Temperature (Celsius)'].mean(); RainMean1953=dataset1953['Rainfall (MM)'].mean()

TempMean1963=dataset1963['Temperature (Celsius)'].mean(); RainMean1963=dataset1963['Rainfall (MM)'].mean()

TempMean1973=dataset1973['Temperature (Celsius)'].mean(); RainMean1973=dataset1973['Rainfall (MM)'].mean()

TempMean1983=dataset1983['Temperature (Celsius)'].mean(); RainMean1983=dataset1983['Rainfall (MM)'].mean()

TempMean1993=dataset1993['Temperature (Celsius)'].mean(); RainMean1993=dataset1993['Rainfall (MM)'].mean()

TempMean2003=dataset2003['Temperature (Celsius)'].mean(); RainMean2003=dataset2003['Rainfall (MM)'].mean()

TempMean2013=dataset2013['Temperature (Celsius)'].mean(); RainMean2013=dataset2013['Rainfall (MM)'].mean()

months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']

monthsN = [1,2,3,4,5,6,7,8,9,10,11,12]

avgTemp = [janTempMean,febTempMean,marTempMean,aprTempMean,mayTempMean,junTempMean,julTempMean,augTempMean,sepTempMean,octTempMean,novTempMean,decTempMean]

avgRain = [janRainMean,febRainMean,marRainMean,aprRainMean,mayRainMean,junRainMean,julRainMean,augRainMean,sepRainMean,octRainMean,novRainMean,decRainMean]

yearsN = [1903,1913,1923,1933,1943,1953,1963,1973,1983,1993,2003,2013]

avgTempDecade = [TempMean1903,TempMean1913,TempMean1923,TempMean1933,TempMean1943,TempMean1953,TempMean1963,TempMean1973,TempMean1983,TempMean1993,TempMean2003,TempMean2013]

avgRainDecade = [RainMean1903,RainMean1913,RainMean1923,RainMean1933,RainMean1943,RainMean1953,RainMean1963,RainMean1973,RainMean1983,RainMean1993,RainMean2003,RainMean2013]

avgTRdata = [avgTemp,avgRain]

avgTRdataDecade = [avgTempDecade,avgRainDecade]

months2 = [ 'November','December', 'January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October']

avgRain2 = [novRainMean,decRainMean,janRainMean,febRainMean,marRainMean,aprRainMean,mayRainMean,junRainMean,julRainMean,augRainMean,sepRainMean,octRainMean]

# Bar Graph of Pakistan's Average Temprature/Month from 1901-2016

def avgTMBarG():

fig = plt.figure()

ax = fig.add\_axes([0.1,0.1,0.8,0.8])

ax.bar(months,avgTemp,color='red')

plt.xlabel("x - Month")

plt.ylabel("y - Temprature (°C)")

plt.title("Pakistan's Average Temprature/Month from 1901-2016")

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

# Bar Graph of Pakistan's Average Rainfall/Month from 1901-2016

def avgRMBarG():

fig1 = plt.figure()

ax = fig1.add\_axes([0.1,0.1,0.8,0.8])

ax.bar(months2,avgRain2,color='green')

plt.xlabel("x - Month")

plt.ylabel("y - Rainfall (mm)")

plt.title("Pakistan's Average Rainfall/Month from 1901-2016")

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

def tempRainMultiBarChart():

fig2 = plt.figure()

ax = fig2.add\_axes([0.1,0.1,0.8,0.8])

X\_axis = np.arange(len(months))

plt.bar(X\_axis - 0.2, avgTemp, 0.4, label = 'Temp (°C)', color='red')

plt.bar(X\_axis + 0.2, avgRain, 0.4, label = 'Rain (mm)', color='green')

plt.xticks(X\_axis, months)

plt.xlabel("Months")

plt.ylabel("Temp (°C), Rain (mm)")

plt.title("Pakistan's Temprature/Month in each Month")

plt.legend()

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

# Scatter Plot of Pakistan's Average Temprature/Month from 1901-2016 (Showing Relation)

def scatterTempGraphFunc():

fig3 = plt.figure()

a1 = fig3.add\_axes([0.1,0.1,0.8,0.8])

a1.plot(monthsN, avgTemp, 'r-')

a1.set\_xlabel('Months')

a1.set\_ylabel('Temp (°C)')

#fig3.legend(labels = ('Temp (°C)'),loc='upper center')

plt.title("Pakistan's Temprature in each Month")

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

def scatterTempGraphFuncDecade():

fig10 = plt.figure()

a1 = fig10.add\_axes([0.1,0.1,0.8,0.8])

a1.plot(yearsN, avgTempDecade, 'r-')

a1.set\_xlabel('Years')

a1.set\_ylabel('Temp (°C)')

#fig3.legend(labels = ('Temp (°C)'),loc='upper center')

plt.title("Pakistan's Temprature in each Decade from 1900-2010")

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

# Scatter Plot of Pakistan's Average Temprature/Month from 1901-2016 (Showing Relation)

def scatterRainGraphFunc():

fig5 = plt.figure()

a1 = fig5.add\_axes([0.1,0.1,0.8,0.8])

a1.plot(monthsN, avgRain, 'g-')

a1.set\_xlabel('Months')

a1.set\_ylabel('Rain (mm)')

#fig5.legend(labels = ('Rain (mm)'),loc='upper center')

plt.title("Pakistan's Rainfall in each Month")

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

def scatterRainGraphFuncDecade():

fig15 = plt.figure()

a1 = fig15.add\_axes([0.1,0.1,0.8,0.8])

a1.plot(yearsN, avgRainDecade, 'g-')

a1.set\_xlabel('Years')

a1.set\_ylabel('Rain (mm)')

#fig5.legend(labels = ('Rain (mm)'),loc='upper center')

plt.title("Pakistan's Rainfall in each Decade from 1900-2010")

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

# Scatter Plot of Pakistan's Average Rainfall-Temprature/Month from 1901-2016 (Showing Relation)

def scatterMultiGraphFunc():

fig6 = plt.figure()

a1 = fig6.add\_axes([0.1,0.1,0.8,0.8])

a1.plot(monthsN, avgTemp, 'ro-')

a1.set\_xlabel('Months')

a1.set\_ylabel('Temp (°C)')

a2 = a1.twinx()

a2.plot(monthsN, avgRain, 'go-')

a2.set\_ylabel('Rain (mm)')

fig6.legend(labels = ('Temp (°C)','Rain (mm)'),loc='upper center')

plt.title("Relationship b/w Pakistan's Rainfall/Temprature in each Month")

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

# Scatter Plot of Pakistan's Average Rainfall-Temprature/Month from 1901-2016 (Showing Relation)

def scatterMultiGraphFuncDecade():

fig16 = plt.figure()

a1 = fig16.add\_axes([0.1,0.1,0.8,0.8])

a1.plot(yearsN, avgTempDecade, 'ro-')

a1.set\_xlabel('Years')

a1.set\_ylabel('Temp (°C)')

a2 = a1.twinx()

a2.plot(yearsN, avgRainDecade, 'go-')

a2.set\_ylabel('Rain (mm)')

fig16.legend(labels = ('Temp (°C)','Rain (mm)'),loc='upper center')

plt.title("Relationship b/w Pakistan's Rainfall/Temprature in each Decade from 1900-2010")

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

def boxPlotTempRain():

fig4 = plt.figure()

ax = fig4.add\_axes([0.1,0.1,0.8,0.8])

ax.set\_xticklabels(['Temprature', 'Rain']) # warning for fixed labels.

bp = ax.boxplot(avgTRdata)

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

def boxPlotTempRainDecade():

fig14 = plt.figure()

ax = fig14.add\_axes([0.1,0.1,0.8,0.8])

ax.set\_xticklabels(['Temprature', 'Rain']) # warning for fixed labels.

bp = ax.boxplot(avgTRdataDecade)

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

def linearReg():

slope, intercept, r, p, std\_err = stats.linregress(avgTempDecade, avgRainDecade)

def myfunc(avgTempDecade):

return slope \* avgTempDecade + intercept

mymodel = list(map(myfunc, avgTempDecade))

fig24 = plt.figure()

ax = fig24.add\_axes([0.1,0.1,0.8,0.8])

ax.set\_xlabel('Temp (°C)')

ax.set\_ylabel('Rain (mm)')

plt.scatter(avgTempDecade, avgRainDecade)

plt.plot(avgTempDecade, mymodel)

plt.title("Regression Model b/w Pakistan's Temprature on Rainfall by each Month")

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

def linearReg2():

slope, intercept, r, p, std\_err = stats.linregress(avgRainDecade, avgTempDecade)

def myfunc(avgRainDecade):

return slope \* avgRainDecade + intercept

mymodel = list(map(myfunc, avgRainDecade))

fig34 = plt.figure()

ax = fig34.add\_axes([0.1,0.1,0.8,0.8])

ax.set\_xlabel('Rain (mm)')

ax.set\_ylabel('Temp (°C)')

plt.scatter(avgRainDecade, avgTempDecade)

plt.plot(avgRainDecade, mymodel)

plt.title("Regression Model b/w Pakistan's Rainfall on Temprature by each Month")

wm = plt.get\_current\_fig\_manager()

wm.window.state('zoomed')

plt.show()

##### FunctionCalls()

file = open("dataset/statstext.txt","w")

meanTempRainFunc()

medianTempRainFunc()

# dataDescript()

minMaxTempratureRow()

minMaxRainfallRow()

file.close()

# avgTMBarG()

# avgTMBarGDecade()

# scatterTempGraphFuncDecade()

# scatterRainGraphFuncDecade()

# scatterMultiGraphFuncDecade()

# avgRMBarG()

# tempRainMultiBarChart()

# boxPlotTempRain()

# scatterTempGraphFunc()

# scatterRainGraphFunc()

# scatterMultiGraphFunc()

# boxPlotTempRain()

# boxPlotTempRainDecade()

# linearReg()

# linearReg2()

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_E\_\_\_\_N\_\_\_\_D\_\_\_\_\_\_\_\_O\_\_F\_\_\_\_\_\_\_\_P\_\_\_R\_\_\_O\_\_\_G\_\_\_R\_\_\_A\_\_\_M\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Graphical user interface

Description automatically generated

Chart, bar chart

Description automatically generated

Chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Chart

Description automatically generated with medium confidence

Chart, line chart, scatter chart

Description automatically generated

Chart, line chart

Description automatically generated

Graphical user interface, text, email

Description automatically generated

Thank you.