Project Report

Marilyn Walsh

# GitHub URL

[mpwcodes/UCDPA\_marilynwalsh (github.com)](https://github.com/mpwcodes/UCDPA_marilynwalsh)

# Abstract

This project analyses a dataset consisting of monthly and yearly average temperature values of a number of cities around the world. It includes two methods for importing data for use, processes for cleaning the dataset, indexing and grouping data, and unique functions and conditional loops to run reusable code. Multiple DataFrames are created by extracting different sets of data from the main dataset, and various charts are generated to compare the values of the smaller datasets.

# Introduction

I chose this use case as I was interested in looking at the temperature values of different areas and comparing the different temperature values for different countries and regions.

# Dataset

API dataset (FX\_INTRADAY dataset):

[API Documentation | Alpha Vantage](https://www.alphavantage.co/documentation/#fx-intraday)

This API dataset is data for the “intraday time series…of the FX currency pair specified” (Alpha Vantage, 2022)

CSV dataset:

[Average Temperature of Cities... | Kaggle](https://www.kaggle.com/datasets/swapnilbhange/average-temperature-of-cities)

MY CSV dataset is “Average Temperature of Cities.csv”, sourced from Kaggle.com. The information on the page describes the data as “a dataset of cities by average temperature (monthly and yearly)” (Bhange, 2021). I chose this dataset as I found it easy to read and work with and was interested in comparing the temperatures from a few different countries.

# Implementation Process

I started by importing the necessary libraries required to perform my analysis: pandas, numpy, and matplotlib.

*import pandas as pd*

*import numpy as np*

*import matplotlib.pyplot as plt*

Next, I imported the requests module, and imported and printed the FX\_INTRADAY dataset from the Alpha Vantage site, using my API key.

*import requests*

*url = 'https://www.alphavantage.co/query?function=FX\_INTRADAY&from\_symbol=EUR&to\_symbol=USD&interval=5min&apikey=QBZFK4EGJ4PF631R'*

*datarequest = requests.get(url)*

*APIdata = datarequest.json()*

*print(APIdata)*

I then imported the “Average Temperature of Cities.csv” dataset from Kaggle and assigned the data to the temps\_data DataFrame. I also printed out the number of rows and columns in the DataFrame, using the shape function.

*temps\_data = pd.read\_csv("Average Temperature of Cities.csv")*

*temps\_data*

*print(temps\_data.shape)*

Next, I began to perform some cleaning actions on the data. I ran isnull().sum() functions on the temps\_data DataFrame to check for any null values in the data, and assigned the result to missing\_values\_count variable. The result returned that there were no null values.

*missing\_values\_count = temps\_data.isnull().sum()*

*print(missing\_values\_count[0:16])*

I ran the dropna() function on temps\_data, to drop rows where there were any null values, and assigned the result to variable temps\_data\_dropped. From printing the number of rows and columns of both temps\_data and temps\_data\_dropped, I could see that the values were the same for both, so there were no records dropped, which confirmed the 0 result from the isnull()check above.

*temps\_data\_dropped = temps\_data.dropna()*

*print(temps\_data.shape, temps\_data\_dropped.shape)*

Since having a temperature with a value of 0 is a valid temperature value, I did not want to change any data that had any 0 values, so I did not run the fillna() function to fill any 0 values.

The temps\_data DataFrame contains information on temperatures for countries across the globe, but I wanted to put the focus on analysing the data for European countries only, so I extracted the data from temps\_data, where the “Continent” column had a value of “Europe”, and assigned this data to a new DataFrame called temps\_data\_europe, and printed the data in this new DataFrame.

*temps\_data\_europe = temps\_data[(temps\_data['Continent'] == 'Europe')]*

*temps\_data\_europe*

I then copied the data for the “City” and “Year” columns into a new DataFrame, europe\_city\_year.

*europe\_city\_year = temps\_data\_europe[['City','Year']].copy()*

*print(europe\_city\_year)*

Next, I set the index for temps\_data\_europe to be the “City” columns, as the values in this column were all unique for the European countries. I also dropped the “Year” column from this DataFrame, as I did not need it for now.

*temps\_data\_europe = temps\_data\_europe.set\_index('City')*

*temps\_data\_europe = temps\_data\_europe.drop('Year', axis=1)*

*temps\_data\_europe*

The values for the temperatures in the dataset had two sets of temperature values – Celsius values outside parentheses, and Fahrenheit values inside parentheses. I wanted to use the Celsius values only, and every row had both values for the 12 month columns. So, I wrote a function called remove\_and\_convert, passing in the DataFrame name and the column name, that would remove any data contained in parentheses. This function also converts the temperature values in the DataFrame from string to float, to make some later analysis easier.

*def remove\_and\_convert(df\_name,col\_name):*

*df\_name[col\_name] = df\_name[col\_name].str.replace(r"\(.\*\)","", regex=True)*

*df\_name[col\_name] = df\_name[col\_name].astype(float)*

To help with running the above function to remove and convert the temp data, I created a list of the column names from temps\_data\_europe, and assigned the values to month\_list. I didn’t need the column names “Country” and “Continent” in this list, so I removed them, leaving the column names corresponding to the months.

*month\_list = list(temps\_data\_europe.columns.values)*

*month\_list*

*month\_list.remove('Country')*

*month\_list.remove('Continent')*

*month\_list*

To iterate through temps\_data\_europe to remove the Fahrenheit data and convert the Celsius data to float, I used a for loop, that iterated through each value in month\_list, to perform the remove\_and\_convert function’s processes on each column that had a corresponding value in month\_list. I passed temps\_data\_europe and month\_name into the function to run the function’s processes on the data in the DataFrame.

*for month\_name in month\_list :*

*remove\_and\_convert(temps\_data\_europe,month\_name)*

Checking the data on temps\_data\_europe, and printing the datatypes of the columns shows that the function processes ran successfully on the data (the values in parentheses are no longer in the data, and the datatypes of the month columns is now set to float).

*temps\_data\_europe*

*print(temps\_data\_europe.dtypes)*

The values saved for the Yearly temperatures in europe\_city\_year also need to have the Fahrenheit values removed and the datatype converted to float, so the remove\_and\_convert function was run again, passing in Europe\_city\_year as df\_name, and “Year” as col\_name.

*remove\_and\_convert(europe\_city\_year,'Year')*

*print(europe\_city\_year.dtypes)*

*europe\_city\_year*

The dataset lists the average temperatures for cities, but I wanted to calculate the average temperature per country in Europe. To do this, I used the groupby() and mean() functions to group the data by the “Country” column, and calculate the average value per month by this group.

This data was saved in DataFrame countries\_grouped

*countries\_grouped = temps\_data\_europe.groupby(['Country']).mean()*

*countries\_grouped*

I then calculated the yearly average for each country by running the mean() function using axis=1 to specify selecting the values across the columns horizontally. These averages were assigned to count\_av, and these were appended to the countries\_grouped DataFrame as an extra column.

*count\_av = countries\_grouped.mean(axis=1)*

*countries\_grouped['Yearly Average'] = count\_av*

*print(countries\_grouped)*

I wanted to then focus on the temperature values for Ireland, so I extracted the values for the index 18, which was the index for Ireland in countries\_grouped, using the to\_numpy() function to put them into an array called ireland\_temps.

*ireland\_temps =countries\_grouped.iloc[18].to\_numpy()*

*print(ireland\_temps)*

I also created an array in a similar fashion to above, so hold the UK temperatures averages, so I can compare these to the Ireland averages. Passing in an index of 44 to pick up the UK values.

*uk\_temps =countries\_grouped.iloc[44].to\_numpy()*

*print(uk\_temps)*

To prepare for plotting some charts, I appended the value of “Year” to the month\_list list, so it would have the same amount of values as the ireland\_temps and uk\_temps arrays.

*month\_list.append('Year')*

*month\_list*

I then set to show the average temperatures for Ireland on a bar chart. I set month\_list to x, and ireland\_temps to y, and plotted them on a bar chart. The chart was titled “Average temperatures for Ireland”, the x axis label was set to “Month”, and the y axis label was set to “Temperature”.

*x=month\_list*

*y=ireland\_temps*

*plt.xlabel('Month')*

*plt.ylabel('Temperature')*

*plt.title('Average temperatures for Ireland')*

*plt.bar(x,y)*

*plt.show()*

I followed this chart with a chart containing multiple plots: one for Ireland temperatures, and one for UK temperatures, to compare both. My x values remained as month\_list, and y1 was set to ireland\_temps, and y2 was set to uk\_temps. I plotted these on a line chart. The chart was titled “Average temperatures for Ireland vs Average temperatures for UK”, and the x and y axes labels were the same as chart 1.

*x=month\_list*

*y1=ireland\_temps*

*y2=uk\_temps*

*plt.xlabel('Month')*

*plt.ylabel('Temperature')*

*plt.title('Average temperatures for Ireland vs Average temperatures for UK')*

*plt.plot(x,y1)*

*plt.plot(x,y2)*

*plt.show()*

I wanted to plot a chart showing how Ireland’s average temperatures compared to the average temperatures for the whole of Europe, so I created a pandas Series, eur\_av, that would contain the Europe averages. These were calculated by running the mean() function on countries\_grouped on axis = 0, selecting the values across the rows.

*eur\_av = countries\_grouped.mean(axis=0)*

*print(type(eur\_av))*

*print(eur\_av)*

I then plotted a chart to compare Ireland’s and UK’s temperatures to the European averages. This chart has three plots: one for the average temperatures form Ireland, one for the average temperatures from the UK, and one for the average temperatures from Europe. My x values remained as month\_list, and y1 was set to ireland\_temps, y2 was set to uk\_temps, and y3 was set to the new eur\_av Series. I plotted these on a line chart. The chart was titled “Average temperatures for Ireland and UK vs Average temperatures for Europe”, and the x and y axes labels were the same as charts 1 and 2.

*x=month\_list*

*y1=ireland\_temps*

*y2=uk\_temps*

*y3=eur\_av.values*

*plt.xlabel('Month')*

*plt.ylabel('Temperature')*

*plt.title('Average temperatures for Ireland and UK vs Average temperatures for Europe')*

*plt.plot(x,y1)*

*plt.plot(x,y2)*

*plt.plot(x,y3)*

*plt.show()*

I wanted to bring the Yearly average temps back onto the rest of the data for Europe. To do this, I used the merge() function on temps\_data\_europe to join this DataFrame with the europe\_city\_year, joining on the “City” column, as both DataFrames have that column in common.

*europe\_city\_merge = temps\_data\_europe.merge(europe\_city\_year, on='City')*

*print (europe\_city\_merge)*

# Results

Chart 1: Ireland average temperatures for each month of the year, and yearly average

Chart, bar chart

Description automatically generated

Chart 2: Comparing the Ireland average temperatures to the UK average temperatures for each month of the year, and yearly average (Ireland temperatures in blue, UK temperatures in orange)

Chart, line chart

Description automatically generated

Chart 3: Comparing the Ireland and UK average temperatures to Europe average temperatures for each month of the year, and yearly average (Ireland temperatures in blue, UK temperatures in orange, Europe temperatures in green)

Chart, line chart

Description automatically generated

# Insights

1. Chart 2 indicates that the average yearly temperature for Ireland and the UK are more or less the same.
2. Chart 2 also shows that, during the winter months, Ireland’s average temperatures are above the UK’s average temperatures, and during the summer months, they are below the UK’s average values
3. From looking at Chart 3, it is interesting that Ireland’s temperatures during the Winter months are above the European average, but Ireland’s temperatures during the summer months are below the European average.
4. Chart 3 also indicates that there is an approximate 3 degrees difference in Europe’s and Ireland’s average coldest temperatures, and an approximate 4 degree difference in the warmest temperatures
5. In chart 3, you can see that, considering the visible differences in the monthly temperatures, the values for the average yearly temperatures are much closer, with just a 1.2 approximate difference (Europe’s value being the highest).

I think the regression methods of machine learning could be used in predicting changes in local and regional temperatures in future years.

# References

Bhange, S. (2021), Average Temperature of Cities.csv Available at:

[Average Temperature of Cities... | Kaggle](https://www.kaggle.com/datasets/swapnilbhange/average-temperature-of-cities)

Alpha Vantage (2022) FX\_INTRADAY. Available at

[API Documentation | Alpha Vantage](https://www.alphavantage.co/documentation/#fx-intraday)