

# AUTOMATIZACIÓN DE ANÁLISIS Y VISUALIZACIÓN DE DATOS CLIMÁTICOS

Jordan Aguilar

Valeria Franco

Karla Villamar

# INTRODUCCIÓN

En la actualidad, el cambio climático está teniendo un importante impacto negativo en la salud física y mental de las personas (Frontiers, 2023).Este estudio se centra en evaluar las condiciones climáticas de 10 ciudades de Estados Unidos, automatizando la carga, limpieza, análisis y visualización de datos que incluye 100 000 registros climáticos.



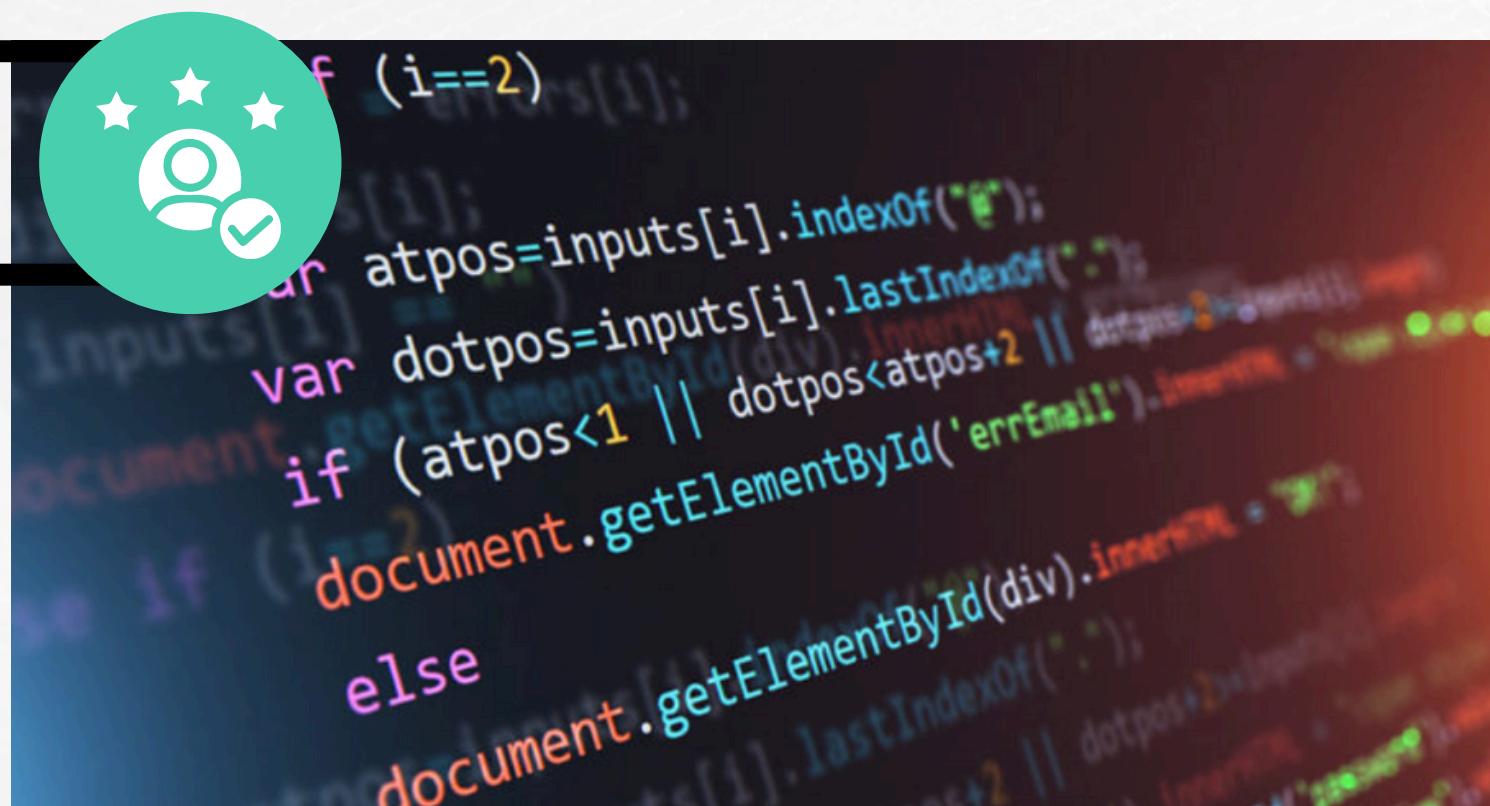


## OBJETIVO GENERAL

Evaluar y caracterizar las condiciones climáticas de varias ciudades de Estados Unidos automatizando el análisis de los datos climáticos.

## OBJETIVO ESPECIFICOS

- Recopilar y procesar los datos climáticos
- Analizar las tendencias de temperatura, humedad, precipitaciones y velocidad del viento a lo largo del tiempo en diferentes ciudades
- Investigar las condiciones meteorológicas extremas.



# CARGA Y PREPARACIÓN DE DATOS:

Se utilizaron diferentes funciones de Pandas para la limpieza básica de datos y se crearon funciones para facilitar la reutilización del código.

## Descripción del conjunto de datos

```
: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

### 1. Carga y Preparación de Datos

#### 1.1 Cargar el conjunto de datos utilizando pandas.

```
: data=pd.read_csv("Weather_Data-1.csv")
: data
```

	Location	Date_Time	Temperature_C	Humidity_pct	Precipitation_mm	Wind_Speed_kmh
0	San Diego	1/14/2024 21:12	10.683001	41.195754	4.020119	8.233540
1	San Diego	5/17/2024 15:22	8.734140	58.319107	9.111623	27.715161
2	San Diego	5/11/2024 9:30	11.632436	38.820175	4.607511	28.732951
3	Philadelphia	2/26/2024 17:32	-8.628976	54.074474	3.183720	26.367303
4	San Antonio	4/29/2024 13:23	39.808213	72.899908	9.598282	29.898622
...	...	...	...	...	...	...
999995	Dallas	1/1/2024 20:29	23.416877	37.705024	3.819833	16.538119
999996	San Antonio	1/20/2024 15:59	6.759080	40.731036	8.182785	29.005558
999997	New York	4/14/2024 8:30	15.664465	62.201884	3.987558	0.403909
999998	Chicago	5/12/2024 20:10	18.999994	63.703245	4.294325	6.326036
999999	New York	4/16/2024 16:11	10.725351	43.804584	1.883292	15.363828

1000000 rows × 6 columns

# LIMPIEZA DE DATOS

```
data.info()
```

```
#Transformar Date_Time a formato de fecha y hora estándar  
data['Date_Time'] = pd.to_datetime(data['Date_Time'])
```

```
#Eliminar duplicados  
x= data.drop_duplicates(subset=['Location', 'Date_Time'], keep='first', inplace=False, ignore_index=False)
```

```
# Crear función para eliminar duplicados  
def supr_duplicados(DataFrame, Columnas):  
    no_duplicados = DataFrame.drop_duplicates(subset=Columnas, ignore_index = True)  
    return no_duplicados  
  
datosfinales = supr_duplicados(data, ['Location', 'Date_Time'])  
datosfinales
```

```
#Eliminar duplicados  
x= data.drop_duplicates(subset=['Location', 'Date_Time'], keep='first', inplace=False, ignore_index=False)
```

1.3 Implementar funciones para cada paso de la limpieza de datos para facilitar la reutilización del código.

```
# Crear función para eliminar duplicados  
def supr_duplicados(DataFrame, Columnas):  
    no_duplicados = DataFrame.drop_duplicates(subset=Columnas, ignore_index = True)  
    return no_duplicados  
  
datosfinales = supr_duplicados(data, ['Location', 'Date_Time'])  
datosfinales
```

# RESUMEN ESTADÍSTICO DE LOS DATOS

```
# Estadística descriptiva de la variable temperatura por locación
PromTemp = datosfinales.groupby('Location')[['Temperature_C']].describe()
PromTemp
```

Location	count	mean	std	min	Temperature_C			
					25%	50%	75%	max
<b>Chicago</b>	78734.0	15.045266	14.440476	-9.999959	2.574750	15.052587	27.546153	39.998561
<b>Dallas</b>	78700.0	14.973336	14.481602	-9.999588	2.383706	15.013138	27.516724	39.998804
<b>Houston</b>	78787.0	14.893002	14.440294	-9.999874	2.358749	14.893656	27.330327	39.998913
<b>Los Angeles</b>	78405.0	15.121807	14.486618	-9.999913	2.548060	15.109511	27.670073	39.999592
<b>New York</b>	78514.0	14.997349	14.415080	-9.999870	2.531978	14.973082	27.492992	39.999801
<b>Philadelphia</b>	78678.0	14.985475	14.406098	-9.999282	2.514576	15.001733	27.445691	39.999642
<b>Phoenix</b>	78806.0	12.758808	14.765683	-19.969311	0.239635	12.702985	25.269871	39.998889
<b>San Antonio</b>	78706.0	15.037004	14.438606	-9.999964	2.516044	15.135638	27.495075	39.998314
<b>San Diego</b>	78528.0	14.917072	14.417957	-9.999986	2.428470	14.802373	27.393347	39.999692
<b>San Jose</b>	78611.0	14.948212	14.393331	-9.999966	2.480728	14.997264	27.346540	39.997461

# Automatización del Análisis Exploratorio de Datos (EDA)

```
#Función para separar meses y años
def month_year(Dt,columna):
    Dt[columna] = pd.to_datetime(Dt[columna])
    Dt['Month'] = Dt[columna].dt.month
    Dt['Year'] = Dt[columna].dt.year
    return Dt
```

```
def convertir_meses(data, nombre_columna):
    mapeo_meses = {
        1: 'Enero',
        2: 'Febrero',
        3: 'Marzo',
        4: 'Abril',
        5: 'Mayo',
        6: 'Junio',
        7: 'Julio',
        8: 'Agosto',
        9: 'Septiembre',
        10: 'Octubre',
        11: 'Noviembre',
        12: 'Diciembre'}
    data[nombre_columna] = data[nombre_columna].apply(lambda x: mapeo_meses[x] if x in mapeo_meses else x)
    return data
```

# Análisis específicos al conjunto de datos

```
z=df3.groupby(["Location","Month"])[["Temperature_C","Humidity_pct","Precipitation_mm","Wind_Speed_kmh"]].mean()
```

#Función para determinar máximo valor promedio de cada variable para cada ciudad y el mes.

```
def valorpromediomayor (variable,data,columna):  
    maximo=data[columna].idxmax()  
    max_1=data[columna].max()  
    print(f"La {variable} promedio mas alta es de {data[columna].max():.2f} °C y se registra en el mes de {maximo[1]} en la ubicación de {maximo[0]}")
```

```
variable="temperatura"  
columna="Temperature_C"  
valorpromediomayor(variable,z,columna)
```

La temperatura promedio mas alta es de 15.39 °C y se registra en el mes de Mayo en la ubicación de Los Angeles.

# Ciudad con velocidades de viento mas altas

```
zmax=df3.groupby(["Location"])[["Temperature_C","Humidity_pct","Precipitation_mm","Wind_Speed_kmh"]].max()
```

```
def valormax (variable,data,columna):  
    maximo=data[columna].idxmax()  
    max_1=data[columna].max()  
    print(f"La ciudad con {variable} mas alta es {maximo} , cuyo valor es de {max_1:.5f} km/h.")
```

```
variable="la velocidad de viento"  
columna="Wind_Speed_kmh"  
valormax(variable,zmax,columna)
```

La ciudad con la velocidad de viento mas alta es Philadelphia , cuyo valor es de 29.99997 km/h.

# Manipulación de Datos con Pandas

```
# Función para contar el número de registros
def num_registro(df):
    return df.groupby('Location').size().reset_index(name='Cantidad Registros')

result = num_registro(dff)
result
```

	Location	Cantidad Registros
0	Chicago	78734
1	Dallas	78700
2	Houston	78787
3	Los Angeles	78405
4	New York	78514
5	Philadelphia	78678
6	Phoenix	78806
7	San Antonio	78706
8	San Diego	78528
9	San Jose	78611

```
#Cálculo de sensación térmica
dff['Wind chill'] = 13.12 + 0.6215 * dff['Temperature_C'] - 11.37 * (dff['Wind_Speed_kmh'] ** 0.16) + 0.3965 * (dff['Temperature_C']) * (dff['Wind_Speed_kmh'] ** 0.16)
dff
```

```
# Cálculo de punto de rocío
dff['dew_point'] = (dff['Temperature_C'] - ((100 - dff['Humidity_pct']) / 5)).round().astype(int)
dff
```

# Condiciones Climáticas con Baja Probabilidad de Impactos Adversos

	25 °C	37 °C
Humedad	30%	60%
Precipitación	5 mm	10 mm

```
c1 = df3.loc[(dff.Temperature_C>=25) & (df3.Temperature_C<=37) & (df3.Humidity_pct>=30) & (df3.Humidity_pct<=60)& (df3.Precipitation_mm>=5) & (df3.Precipitation_mm<=10)
c1.info()
print()
print(c1["Location"].unique())
print()
print(c1["Month"].unique())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 46751 entries, 5 to 786449
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Month       46751 non-null   object 
 1   Location    46751 non-null   object 
dtypes: object(2)
memory usage: 1.1+ MB

['San Diego' 'New York' 'Philadelphia' 'San Antonio' 'San Jose' 'Houston'
 'Chicago' 'Los Angeles' 'Phoenix' 'Dallas']
['Enero' 'Marzo' 'Mayo' 'Febrero' 'Abril']
```

**Condiciones climáticas severas sin protección adecuada ponen al ser humano en peligro de muerte por frío extremo.**

Temperatura	< -10 °C
Humedad	> 20%
Velocidad del Viento	> 10 km/h

```
c3 = df3.loc[(dff.Temperature_C<=-10) & (df3.Humidity_pct>=20) & (df3.Wind_Speed_kmh>=10), ["Month","Location","Date_Time","Temperature_C","Humidity_pct","Wind_Speed_kmh"]]
c3.info()
print()
print(c3["Location"].unique())
print()
print(c3["Month"].unique())

<class 'pandas.core.frame.DataFrame'>
Index: 2298 entries, 493 to 785975
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Month       2298 non-null   object 
 1   Location    2298 non-null   object 
 2   Date_Time   2298 non-null   datetime64[ns]
 3   Temperature_C 2298 non-null   float64 
 4   Humidity_pct 2298 non-null   float64 
 5   Wind_Speed_kmh 2298 non-null   float64 
dtypes: datetime64[ns](1), float64(3), object(2)
memory usage: 125.7+ KB

['Phoenix']
['Enero' 'Febrero']
```

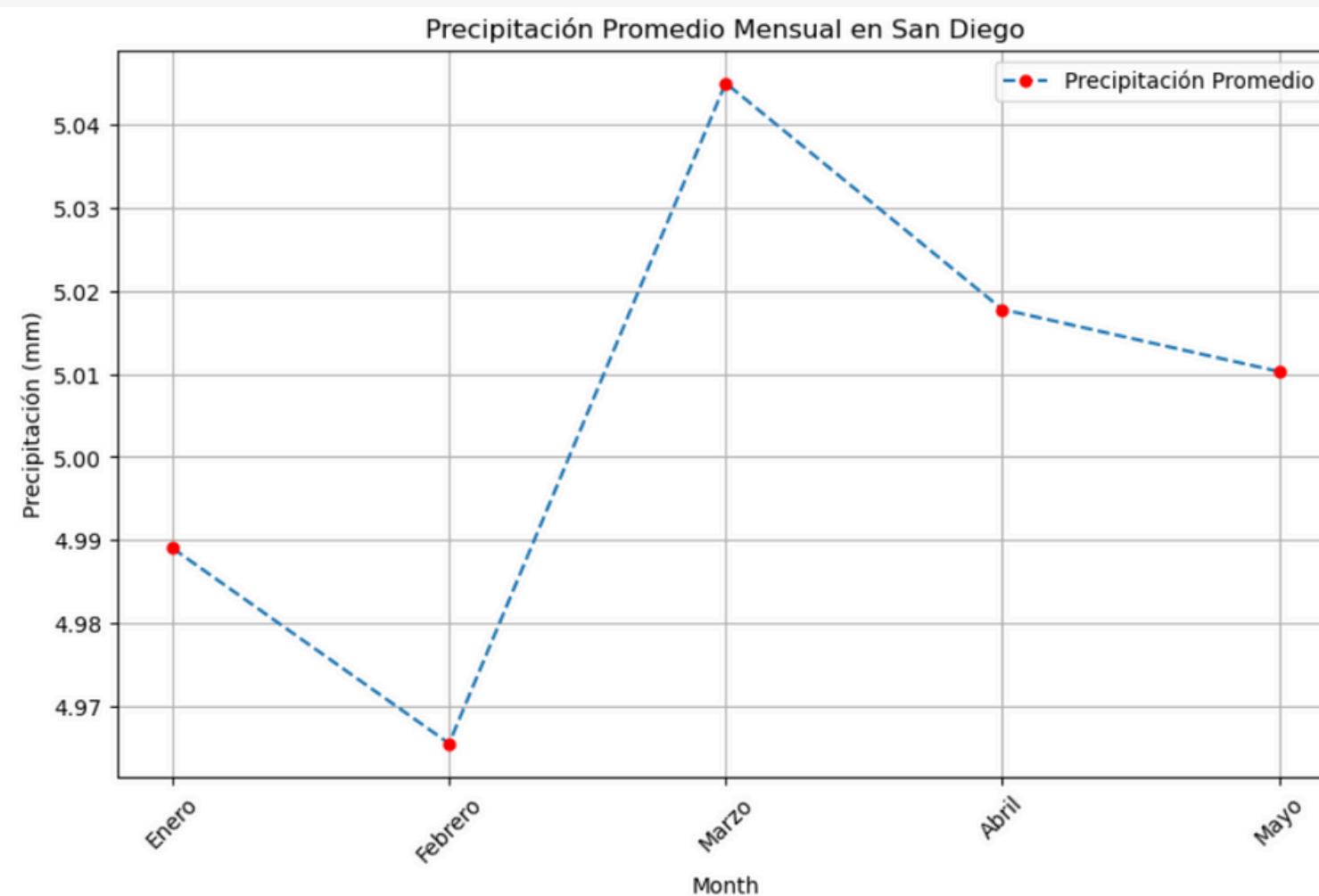
# Visualización de Datos Automatizada

## Gráficos de línea

```
#Función para definir la temperatura promedio por mes según la ciudad
def plot_Tmonthmean(df, city):
    df_city = df[df['Location'] == city]
    mesnum = {"Enero": 1, "Febrero": 2, "Marzo": 3, "Abril": 4, "Mayo": 5}
    df_city['Mes_num'] = df_city['Month'].map(mesnum)
    mean_temp = df_city.groupby('Mes_num')['Precipitation_mm'].mean().reset_index()

#Crear el gráfico de líneas
plt.figure(figsize=(10, 6))
plt.plot(mean_temp['Mes_num'], mean_temp['Precipitation_mm'], marker='o', ms=5, mfc="red", markeredgecolor="red", linestyle='--', label='Precipitación Promedio')
plt.xlabel('Month')
plt.ylabel('Precipitación (mm)')
plt.title(f'Precipitación Promedio Mensual en {city}')
plt.xticks(ticks=list(mesnum.values()), labels=list(mesnum.keys()), rotation=45)
plt.grid(True)
plt.legend()
plt.show()

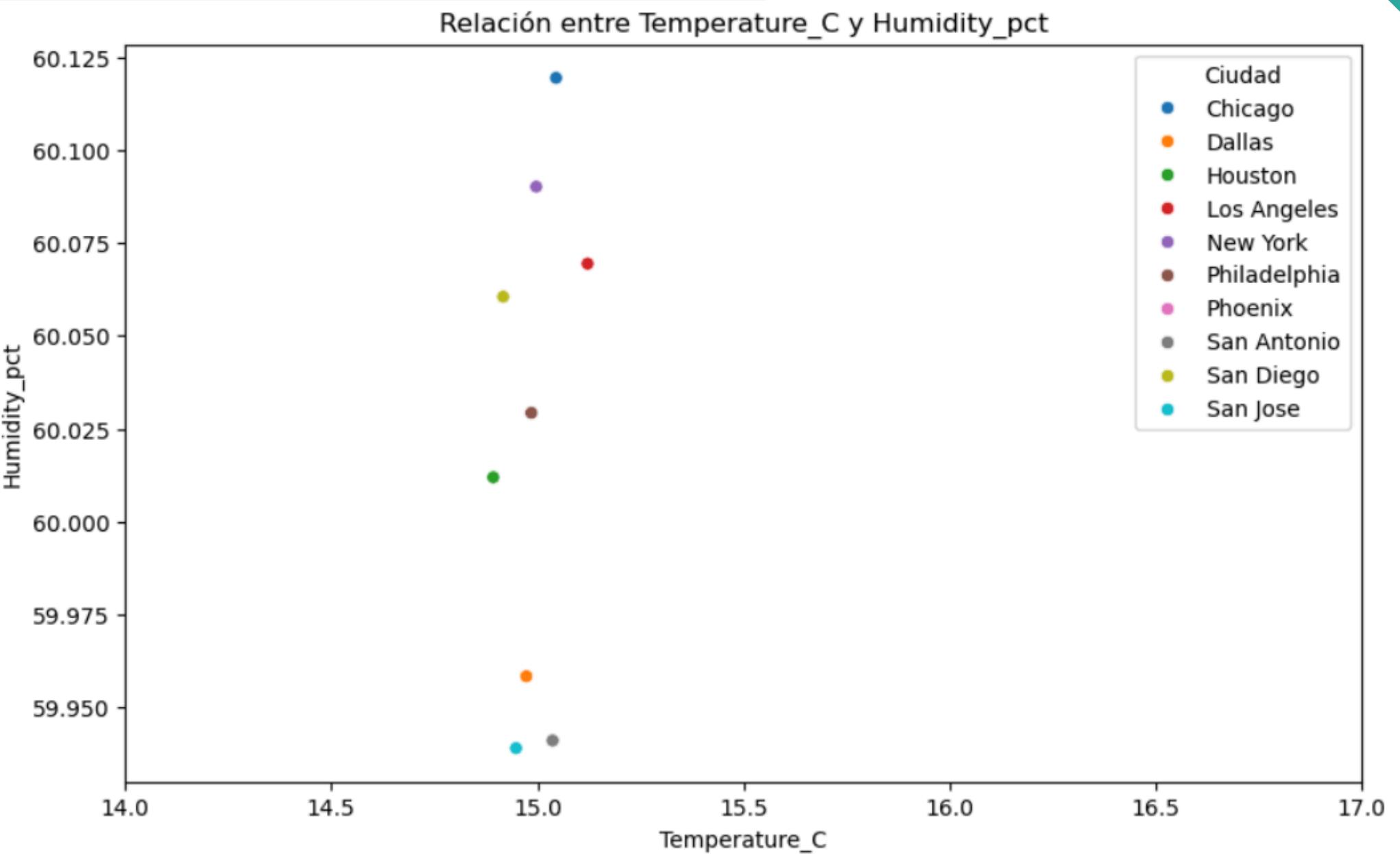
plot_Tmonthmean(dff, 'San Diego')
```



## Diagrama de dispersión

```
def scatter_city(df, v1,v2):
    plt.figure(figsize=(10, 6))
    mn = df.groupby('Location')[[v1, v2]].mean().reset_index()
    sns.scatterplot(x= mn[v1], y= mn[v2], hue=mn['Location'], data=mn)
    plt.xlabel(v1)
    plt.ylabel(v2)
    plt.xlim(14, 17)
    plt.title(f'Relación entre {v1} y {v2}')
    plt.legend(title='Ciudad')
    plt.show()

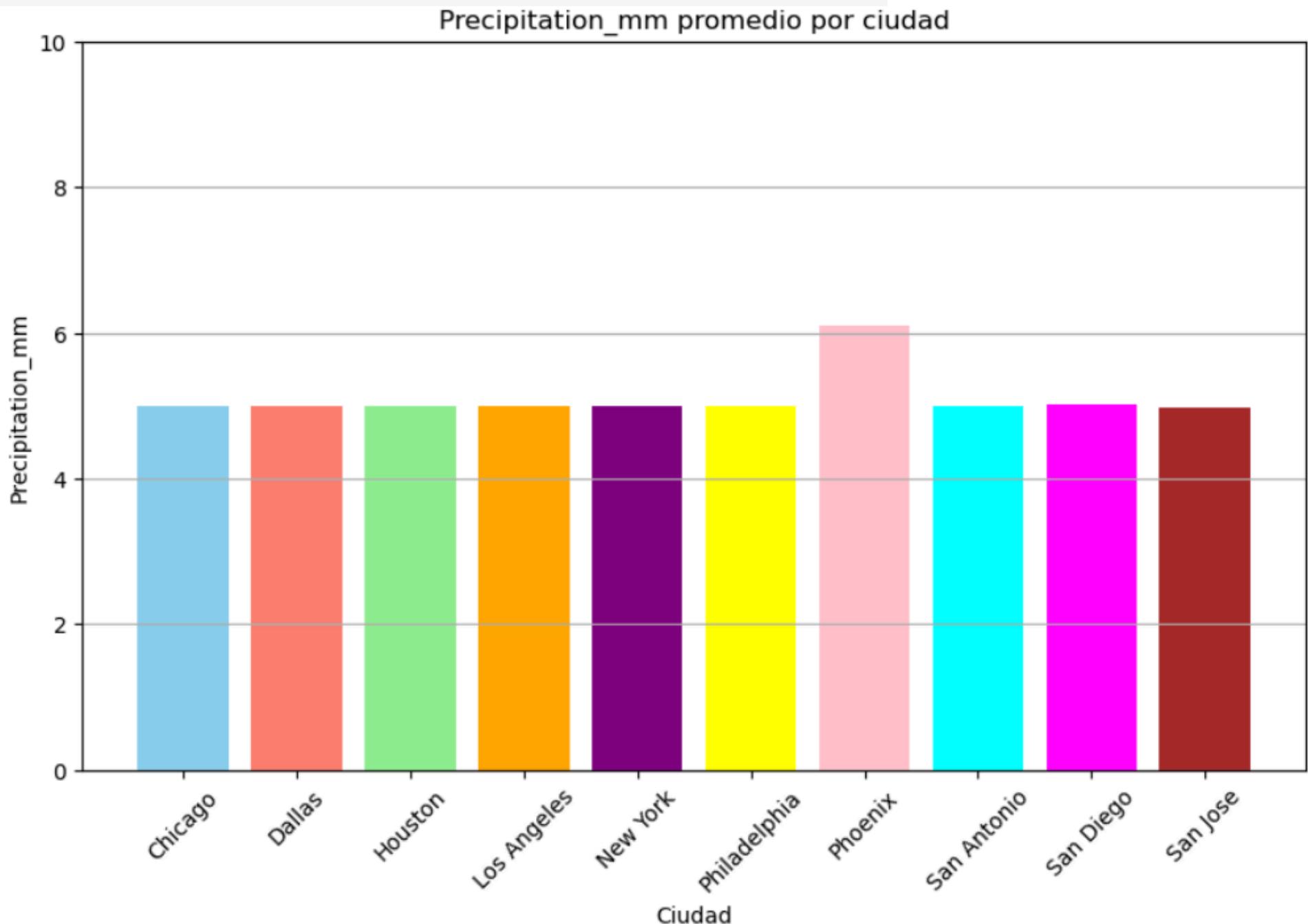
scatter_city(dff, 'Temperature_C', 'Humidity_pct')
```



## Gráfico de barras

```
#Funcion para definir la variable promedio por ciudad
def bar_city(df, variabl):
    plt.figure(figsize=(10, 6))
    mean_prec = df.groupby('Location')[variabl].mean().reset_index()
    colors = ['skyblue', 'salmon', 'lightgreen', 'orange', 'purple', 'yellow', 'pink', 'cyan', 'magenta', 'brown']
    plt.bar(mean_prec['Location'], mean_prec[variabl], color=colors)
    plt.xlabel('Ciudad')
    plt.ylabel(variabl)
    plt.ylim(0, 10)
    plt.title(f'{variabl} promedio por ciudad')
    plt.xticks(rotation=45)
    plt.grid(axis='y')
    plt.show()

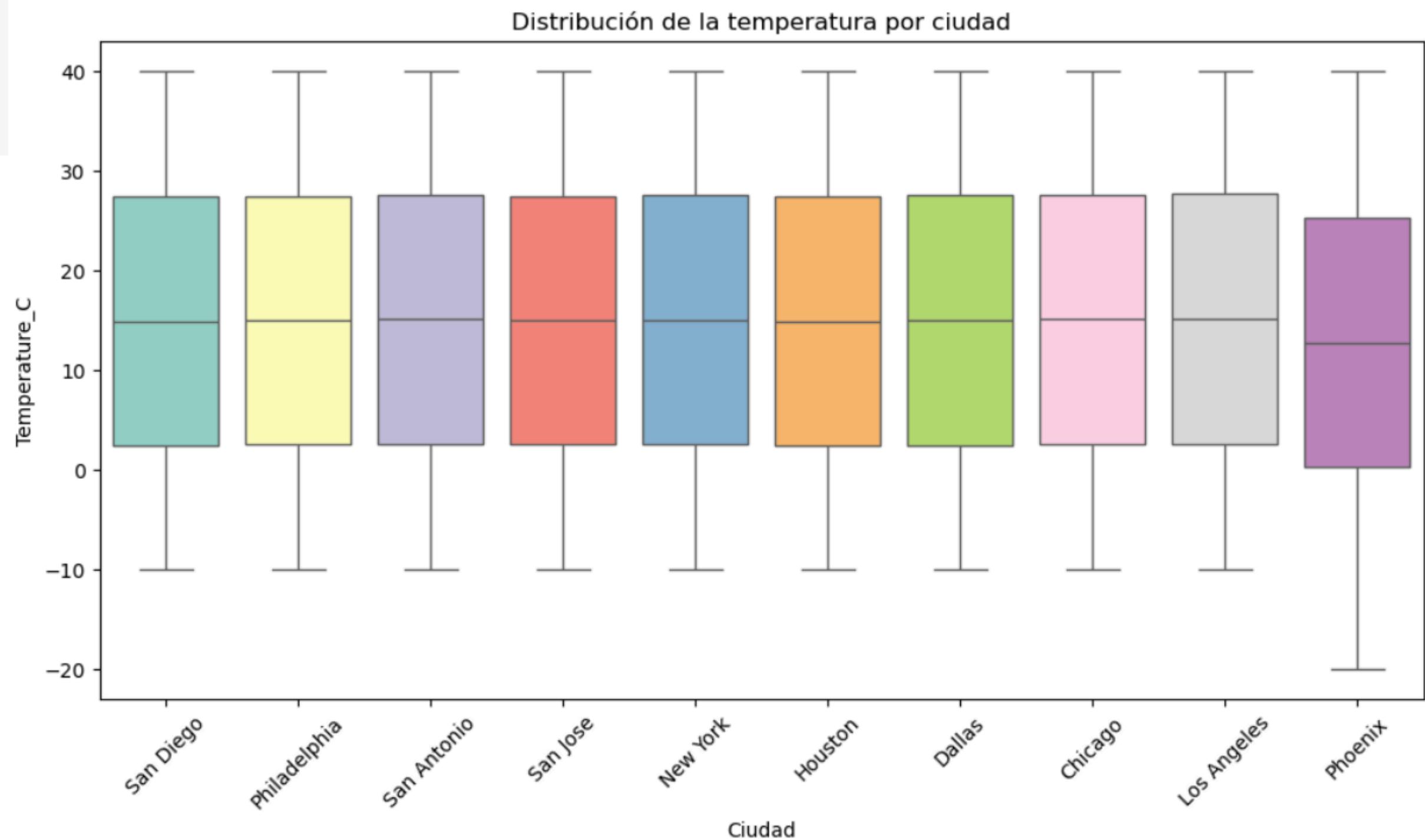
bar_city(dff, 'Precipitation_mm')
```



## Gráfico de cajas

```
def grafbox(das,variabl):
    plt.figure(figsize=(12, 6))
    sns.boxplot(x=das["Location"], y=das[variabl], saturation=0.9,data=das, palette="Set3")
    plt.xlabel('Ciudad')
    plt.ylabel(variabl)
    plt.title('Distribución de la temperatura por ciudad')
    plt.xticks(rotation=45)
    plt.show()

grafbox(dff, 'Temperature_C')
```

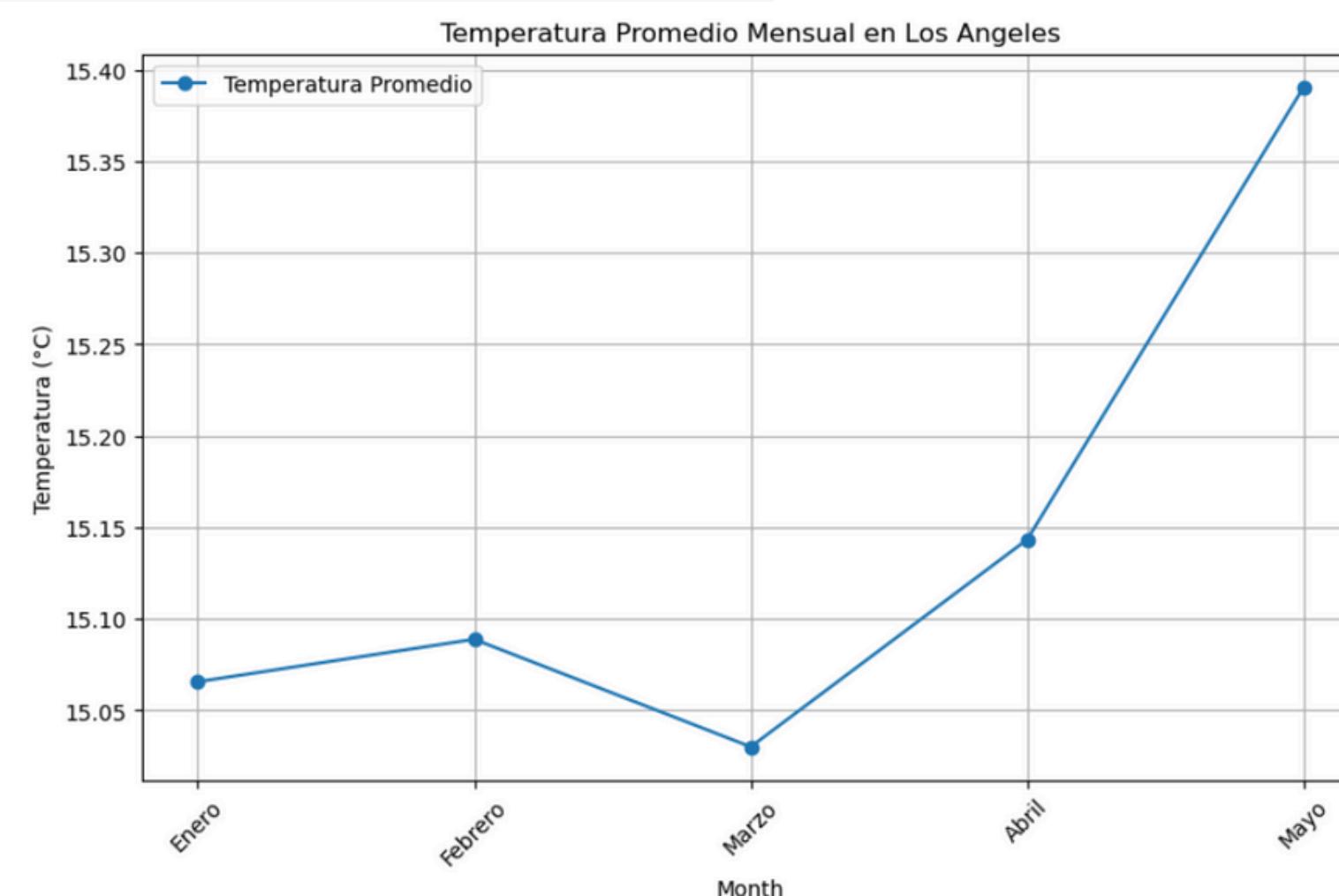


## Función para generar y guardar gráficos

```
# Función para definir la temperatura máxima por mes según la ciudad
def plot_precmeanmensualP(df, city):
    df_city = df[df['Location'] == city]
    mesnum = {"Enero": 1, "Febrero": 2, "Marzo": 3, "Abril": 4, "Mayo": 5}
    df_city['Mes_num'] = df_city['Month'].map(mesnum)
    mean_temp = df_city.groupby('Mes_num')['Temperature_C'].mean().reset_index()

    plt.figure(figsize=(10, 6))
    plt.plot(mean_temp['Mes_num'], mean_temp['Temperature_C'], marker='o', linestyle='-', label='Temperatura Promedio')
    plt.xlabel('Month')
    plt.ylabel('Temperatura (°C)')
    plt.title(f'Temperatura Promedio Mensual en {city}')
    plt.xticks(ticks=list(mesnum.values()), labels=list(mesnum.keys()), rotation=45)
    plt.grid(True)
    plt.legend()
    plt.savefig(f'{city}_Temperatura_Promedio_Mensual.png')
    plt.show()
```

```
cities = dff["Location"].unique()
for city in cities:
    plot_precmeanmensualP(dff, city)
    plt.close()
```

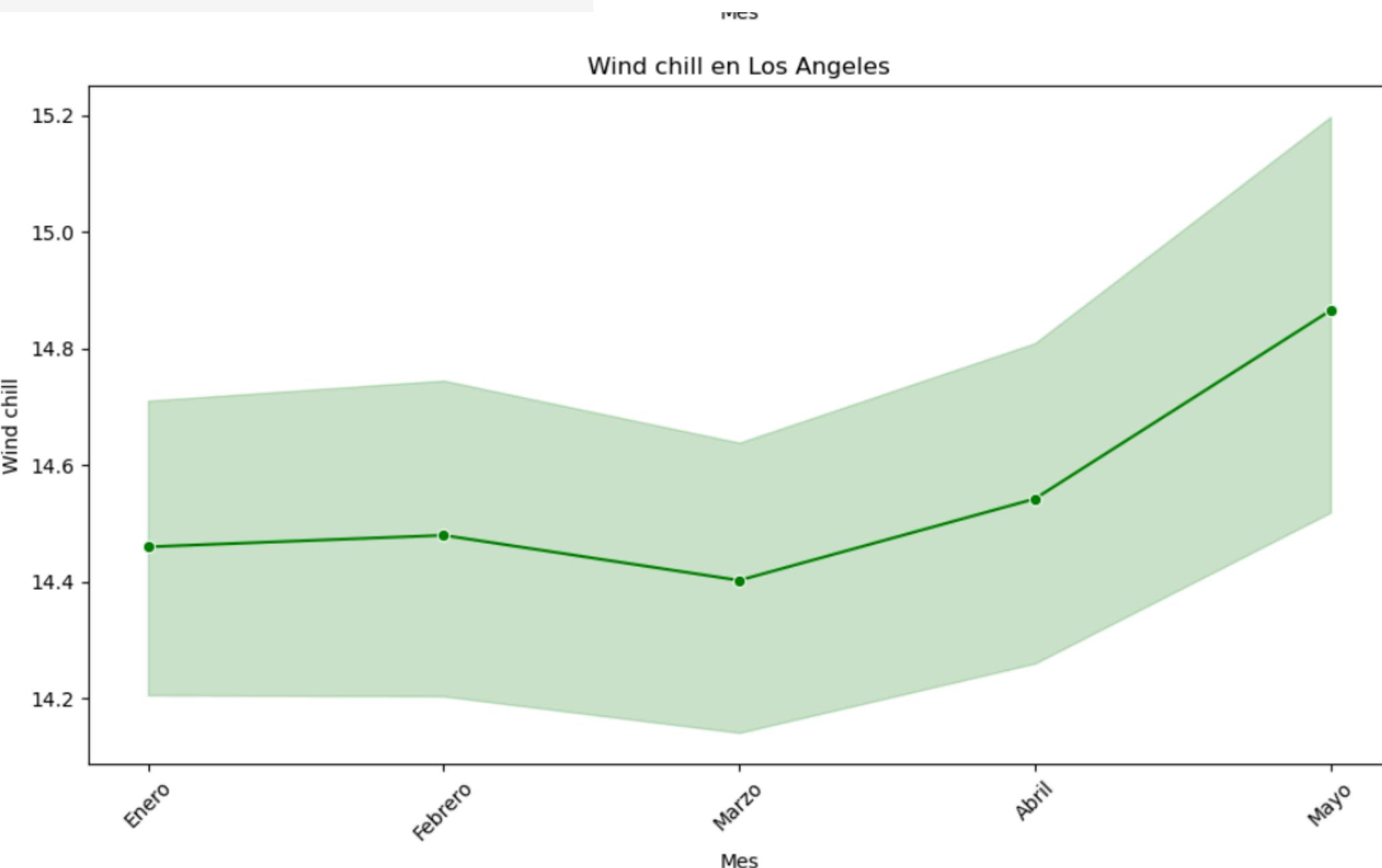


```

def variable_city(dat,variable):
    mean_var = dat.groupby('Location')[variable].mean().reset_index()
    month_order = ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo']
    dat['Month'] = pd.Categorical(dat['Month'], categories=month_order, ordered=True)
    cities= mean_var['Location'].unique()
    for city in cities:
        city_data = dat[dat['Location'] == city]
        # Crear un gráfico para la temperatura
        plt.figure(figsize=(10, 6))
        sns.lineplot(x='Month', y=variable, data=city_data, marker='o', color='green')
        plt.title(f'{variable} en {city}')
        plt.xlabel('Mes')
        plt.ylabel(variable)
        plt.xticks(rotation=45)
        plt.tight_layout()
        # plt.ylim(14.0, 16.0)
        plt.savefig(f'{variable}_{city}.png')
        plt.show()
        plt.close()

```

```
variable_city(dff, 'Wind chill')
```



# CONCLUSIONES



1

Se confirmó la utilidad de la automatización en el análisis de datos climáticos, permitiendo una evaluación más rápida y precisa

2

Las técnicas automatizadas pueden ser implementadas efectivamente para el análisis climático

3

Se recomienda siempre comprobar la cantidad de registros por cada variable a analizar

4

Se puede comprobar variables que están relacionadas gracias a los gráficos relacionados como la temperatura y la sensación térmica

5

Se recomienda la realización de estudios a largo plazo que utilicen la automatización para monitorear cambios en los patrones climáticos.