T1_Indices_de_Contaminacion

May 20, 2023

1 Tarea 1 - Índices de Contaminación

1.0.1 Héctor Hibran Tapia Fernández - A01661114

Hecho en Google Colab:

[ ]:
```
!pip install numpy
!pip install pandas
!pip install unidecode
!pip install matplotlib
```

[74]:
```
import unidecode
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Vamos a iniciar el análisis con una base de datos con menos variables.

http://www.aire.cdmx.gob.mx/default.php?opc=%27aKBhnmI=%27&opcion=aw==Links

[ ]:
```
# Espacio para que cargar las 4 bases de datos.
# indice_2019, indice_2020, indice_2021, indice_2022.

# Las fechas generalmente se leen como strings en lugar de como objetos de␣
 ↪fecha y hora, con parse_dates se leen como fecha y hora.

filename_2019 = '/content/indice_2019.csv'
data_2019 = pd.read_csv(filename_2019, encoding = 'latin-1', parse_dates =␣
 ↪['Fecha'])

filename_2020 = '/content/indice_2020.csv'
data_2020 = pd.read_csv(filename_2020, encoding = 'latin-1', parse_dates =␣
 ↪['Fecha'])

filename_2021 = '/content/indice_2021.csv'
```

```
data_2021 = pd.read_csv(filename_2021, encoding = 'latin-1', parse_dates =␣
 ↪['Fecha'])

filename_2022 = '/content/indice_2022.csv'
data_2022 = pd.read_csv(filename_2022, encoding = 'latin-1', parse_dates =␣
 ↪['Fecha'])
```

Preparamos la base de datos para que pueda ser analizada.

[76]:
```
combined_data = pd.concat([data_2019, data_2020, data_2021, data_2022],␣
 ↪ignore_index=True) # Juntamos las cuatro bases de datos, en un solo␣
 ↪dataframe.
combined_data = combined_data.sort_values(by = 'Fecha') # Ordena las fechas en␣
 ↪orden cronológico.
combined_data.columns = [unidecode.unidecode(col) for col in combined_data.
 ↪columns] # Normaliza los nombres de las columnas eliminando los acentos.

combined_data.head()
```

[76]:

	Fecha	Hora	Noroeste ozono	Noroeste dioxido de azufre	\
0	2019-01-01	1	NaN	5.0	
22	2019-01-01	23	22.0	7.0	
21	2019-01-01	22	22.0	5.0	
20	2019-01-01	21	31.0	4.0	
19	2019-01-01	20	26.0	4.0	

	Noroeste dioxido de nitrogeno	Noroeste monoxido de carbono	\
0	NaN	10.0	
22	15.0	3.0	
21	16.0	3.0	
20	23.0	3.0	
19	26.0	3.0	

	Noroeste PM10	Noroeste PM25	Noreste ozono	Noreste dioxido de azufre	\
0	64.0	74.0	NaN	12	
22	114.0	126.0	9.0	4	
21	114.0	126.0	19.0	4	
20	115.0	127.0	15.0	4	
19	116.0	127.0	19.0	4	

	…	Suroeste dioxido de nitrogeno	Suroeste monoxido de carbono	\
0	…	NaN	9.0	
22	…	20.0	5.0	
21	…	22.0	5.0	
20	…	19.0	5.0	
19	…	25.0	5.0	

```
    Suroeste PM10  Suroeste PM25  Sureste ozono  Sureste dioxido de azufre  \
0            38.0           59.0            NaN                        5.0
22          100.0          107.0           13.0                        2.0
21          101.0          107.0           15.0                        2.0
20          101.0          108.0           20.0                        2.0
19          101.0          109.0           25.0                        2.0

    Sureste dioxido de nitrogeno  Sureste monoxido de carbono  Sureste PM10  \
0                            NaN                          9.0          52.0
22                          20.0                          4.0         114.0
21                          22.0                          3.0         116.0
20                          18.0                          3.0         117.0
19                          12.0                          3.0         117.0

    Sureste PM25
0           65.0
22         141.0
21         141.0
20         141.0
19         141.0

[5 rows x 32 columns]
```

[77]:
```
print(combined_data.shape)
```

```
(35064, 32)
```

###1. ¿Cuales son las variables de esta base de datos?

[78]:
```
# Fecha y Hora.
# Noroeste Ozono,        Noroeste Dióxido de Azufre,        Noroeste Dióxido de␣
 ↪Nitrógeno,        Noroeste Monóxido de Carbono,        Noroeste PM10,␣
 ↪Noroeste PM25.
# Noreste Ozono,        Noreste Dióxido de Azufre,        Noreste Dióxido de␣
 ↪Nitrógeno,        Noreste Monóxido de Carbono,        Noreste PM10, Noreste␣
 ↪PM25.
# Centro Ozono,        Centro Dióxido de Azufre,        Centro Dióxido de␣
 ↪Nitrógeno,        Centro Monóxido de Carbono,        Centro PM10, Centro␣
 ↪PM25.
# Suroeste Ozono,        Suroeste Dióxido de Azufre,        Suroeste Dióxido de␣
 ↪Nitrógeno,        Suroeste Monóxido de Carbono,        Suroeste PM10,␣
 ↪Suroeste PM25.
# Sureste Ozono,        Sureste Dióxido de Azufre,        Sureste Dióxido de␣
 ↪Nitrógeno,        Sureste Monóxido de Carbono,        Sureste PM10, Sureste␣
 ↪PM25.
```

###2. Identifica cuales son las dimensiones de variabilidad en esta base de datos.

[79]:
```
# Tiempo, espacio y contaminantes.
```

###3. Selecciona 4 años consecutivos y descarga el archivo csv o xml. Extrae los datos con Python (ya sea numpy o pandas). Identifica los valores faltantes y cámbialos por NaN (por el momento).

[80]:
```
combined_data_filled = combined_data.fillna("NaN")
combined_data_filled.head()
```

[80]:

	Fecha	Hora	Noroeste ozono	Noroeste dioxido de azufre	\
0	2019-01-01	1	NaN	5.0	
22	2019-01-01	23	22.0	7.0	
21	2019-01-01	22	22.0	5.0	
20	2019-01-01	21	31.0	4.0	
19	2019-01-01	20	26.0	4.0	

	Noroeste dioxido de nitrogeno	Noroeste monoxido de carbono	Noroeste PM10	\
0	NaN	10.0	64.0	
22	15.0	3.0	114.0	
21	16.0	3.0	114.0	
20	23.0	3.0	115.0	
19	26.0	3.0	116.0	

	Noroeste PM25	Noreste ozono	Noreste dioxido de azufre	…	\
0	74.0	NaN	12	…	
22	126.0	9.0	4	…	
21	126.0	19.0	4	…	
20	127.0	15.0	4	…	
19	127.0	19.0	4	…	

	Suroeste dioxido de nitrogeno	Suroeste monoxido de carbono	Suroeste PM10	\
0	NaN	9.0	38.0	
22	20.0	5.0	100.0	
21	22.0	5.0	101.0	
20	19.0	5.0	101.0	
19	25.0	5.0	101.0	

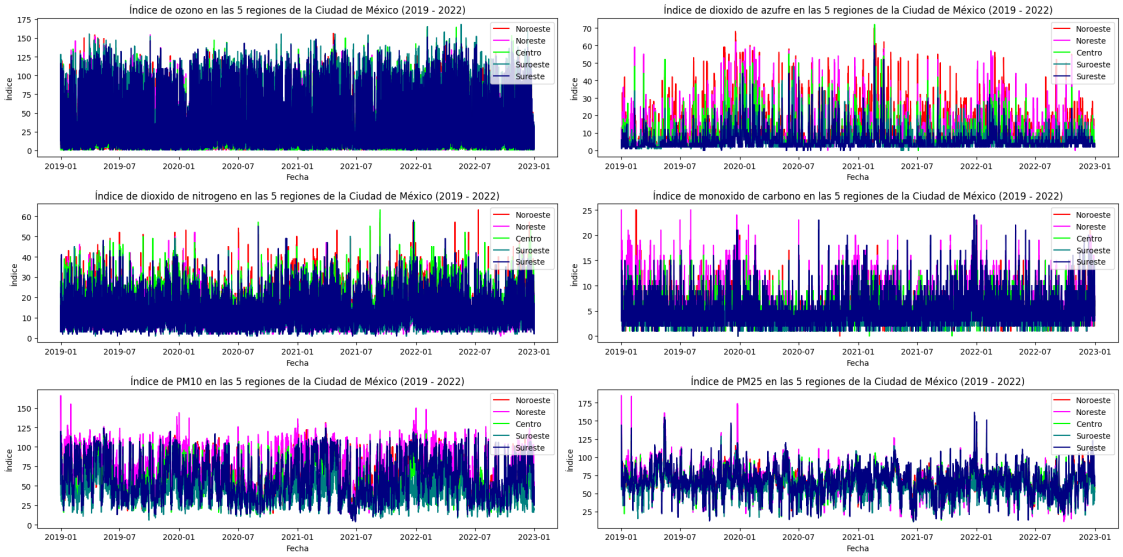
	Suroeste PM25	Sureste ozono	Sureste dioxido de azufre	\
0	59.0	NaN	5.0	
22	107.0	13.0	2.0	
21	107.0	15.0	2.0	
20	108.0	20.0	2.0	
19	109.0	25.0	2.0	

	Sureste dioxido de nitrogeno	Sureste monoxido de carbono	Sureste PM10	\
0	NaN	9.0	52.0	
22	20.0	4.0	114.0	
21	22.0	3.0	116.0	

```
20                      18.0                      3.0        117.0
19                      12.0                      3.0        117.0

   Sureste PM25
0          65.0
22        141.0
21        141.0
20        141.0
19        141.0

[5 rows x 32 columns]
```

###4. Elabora una gráfica de línea con la evolución temporal de la variable. (Original)

[81]:
```
regions = ['Noroeste', 'Noreste', 'Centro', 'Suroeste', 'Sureste']
gases = ['ozono', 'dioxido de azufre', 'dioxido de nitrogeno', 'monoxido de␣
 ↪carbono', 'PM10', 'PM25']
colors = ['red', 'magenta', 'lime', 'teal', 'navy']

fig, axs = plt.subplots(3, 2, figsize=(20,10))  # 3 filas, 2 columnas.

# Loop de cada gas.
for gas_index, gas in enumerate(gases):
    row = gas_index // 2
    col = gas_index % 2

    # Loop de cada región.
    for i, region in enumerate(regions):
        # Crea un nombre de columna combinando la región y el gas.
        column_name = f'{region} {gas}'

        # Plot de los datos para dicha región y gas, con etiqueta y color.
        axs[row, col].plot(combined_data['Fecha'], combined_data[column_name],␣
 ↪color = colors[i], label = region)

    axs[row, col].set_title(f'Índice de {gas} en las 5 regiones de la Ciudad de␣
 ↪México (2019 - 2022)')
    axs[row, col].set_xlabel('Fecha')
    axs[row, col].set_ylabel('Índice')
    axs[row, col].legend(loc='upper right')

plt.tight_layout()
plt.show()
```



###5. Versión suavizada (de los datos interpolados) con media móvil.

[82]:
```
# Interpola los datos de cada columna
for col in combined_data.columns:
    if col != 'Fecha':  # Todas las columnas menos la de Fecha
        combined_data[col] = combined_data[col].interpolate()
```

[83]:
```
# Tamaño de ventana para el promedio móvil (ancho del kernel).
window_size = 5000 #Es el que más me conviene.
```

1.0.2 Kernel Rectangular

[84]:
```
kernel = np.ones(window_size) / window_size # Promedio Simple.
```

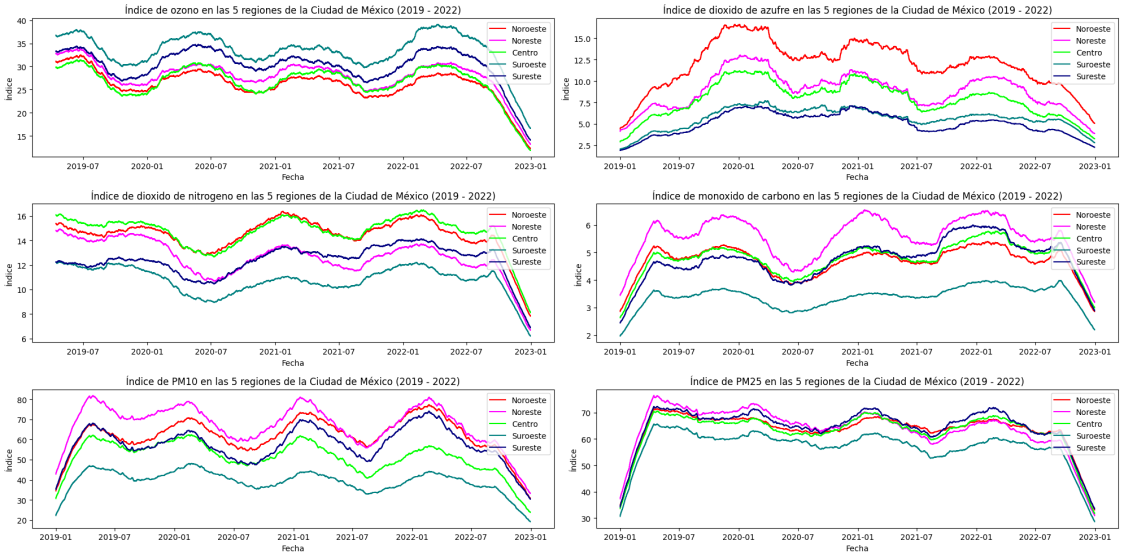
Esto crea una matriz de longitud **5000** con cada entrada siendo **1/5000**. Esto implica que cada punto dentro del tamaño de la ventana contribuye igualmente al resultado, por lo que es un promedio simple.

Básicamente deslizamos esta ventana a través de nuestros datos y en cada punto, calculamos el promedio de los valores en la ventana. Esto tiene el efecto de suavizar las fluctuaciones y revelar las tendencias subyacentes.

Este kernel no siempre es la mejor opción para las operaciones de convolución. Por ejemplo, si queremos que los puntos de datos más recientes tengan más influencia en el promedio, entonces debemos usar otro kernel, como el triangular o gaussiano, que asigna más peso a los puntos cerca del centro del ventana.

[85]:
```
# Se plotea con el mismo código de arriba, solo añade dos lineas de código.

fig, axs = plt.subplots(3, 2, figsize=(20,10))  # 3 filas, 2 columnas.
```

```
# Loop de cada gas.
for gas_index, gas in enumerate(gases):
    row = gas_index // 2
    col = gas_index % 2

    # Loop de cada región.
    for i, region in enumerate(regions):
        # Crea un nombre de columna combinando la región y el gas.
        column_name = f'{region} {gas}'

        # Uso la data interpolada.
        data = combined_data[column_name].interpolate()

        # Aplico el convolve para smoothear el data.
        smoothed_data = np.convolve(data, kernel, 'same')

        # Plot de los datos para dicha región y gas, con etiqueta y color.
        axs[row, col].plot(combined_data['Fecha'], smoothed_data, color =␣
 ↪colors[i], label = region)

    axs[row, col].set_title(f'Índice de {gas} en las 5 regiones de la Ciudad de␣
 ↪México (2019 - 2022)')
    axs[row, col].set_xlabel('Fecha')
    axs[row, col].set_ylabel('Índice')
    axs[row, col].legend(loc='upper right')

plt.tight_layout()
plt.show()
```



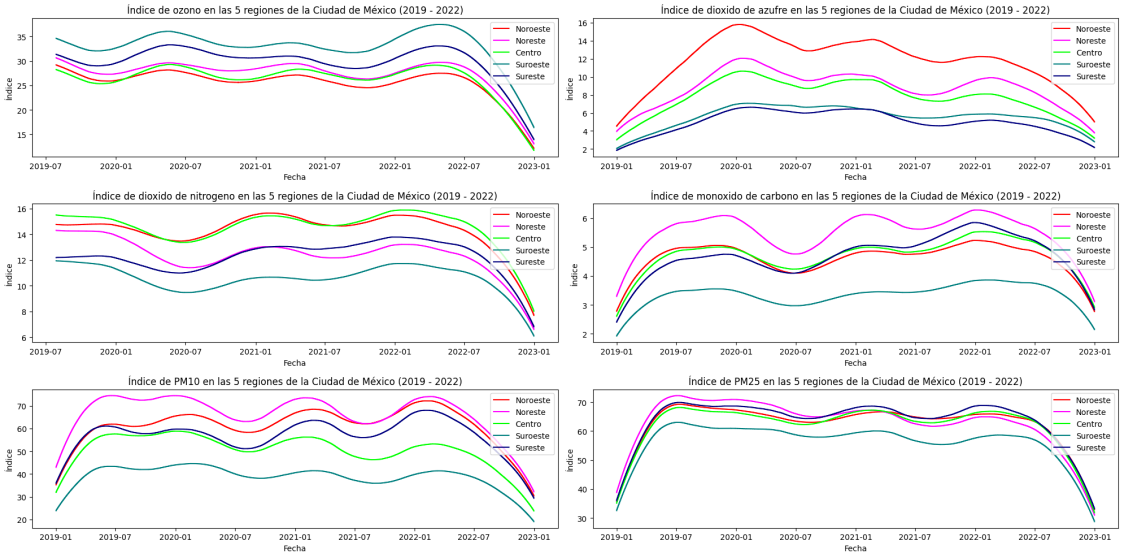
1.0.3 Kernel Triangular

[86]:
```
kernel = np.convolve(np.ones(window_size), np.ones(window_size), 'full') # Crea␣
 ↪dos matrices y las convoluciona.
kernel = kernel / kernel.sum()  # Normaliza y se asegura que la suma de los␣
 ↪pesos sea 1.
```

Crea dos matrices de unos, los convoluciona para formar una forma triangular y luego normaliza el resultado para que la suma de los valores del núcleo sea 1.

Asigna pesos de manera linealmente decreciente desde el centro de la ventana. El punto central obtiene el peso más alto y los pesos disminuyen linealmente a medida que nos movemos hacia los bordes. Básicamente, esto calcula un promedio móvil ponderado sobre el tamaño de la ventana.

[87]:
```
# Se plotea con el mismo código de arriba, solo añade dos lineas de código.

fig, axs = plt.subplots(3, 2, figsize=(20,10))  # 3 filas, 2 columnas.

# Loop de cada gas.
for gas_index, gas in enumerate(gases):
    row = gas_index // 2
    col = gas_index % 2

    # Loop de cada región.
    for i, region in enumerate(regions):
        # Crea un nombre de columna combinando la región y el gas.
        column_name = f'{region} {gas}'

        # Uso la data interpolada.
        data = combined_data[column_name].interpolate()

        # Aplico el convolve para smoothear el data.
        smoothed_data = np.convolve(data, kernel, 'same')

        # Plot de los datos para dicha región y gas, con etiqueta y color.
        axs[row, col].plot(combined_data['Fecha'], smoothed_data, color =␣
 ↪colors[i], label = region)

    axs[row, col].set_title(f'Índice de {gas} en las 5 regiones de la Ciudad de␣
 ↪México (2019 - 2022)')
    axs[row, col].set_xlabel('Fecha')
    axs[row, col].set_ylabel('Índice')
    axs[row, col].legend(loc='upper right')

plt.tight_layout()
plt.show()
```



###Kernel Gaussiano

Este kernel otorga el mayor peso a los datos en el centro de la ventana y los pesos disminuyen simétricamente a medida que nos alejamos del centro siguiendo una distribución gaussiana.

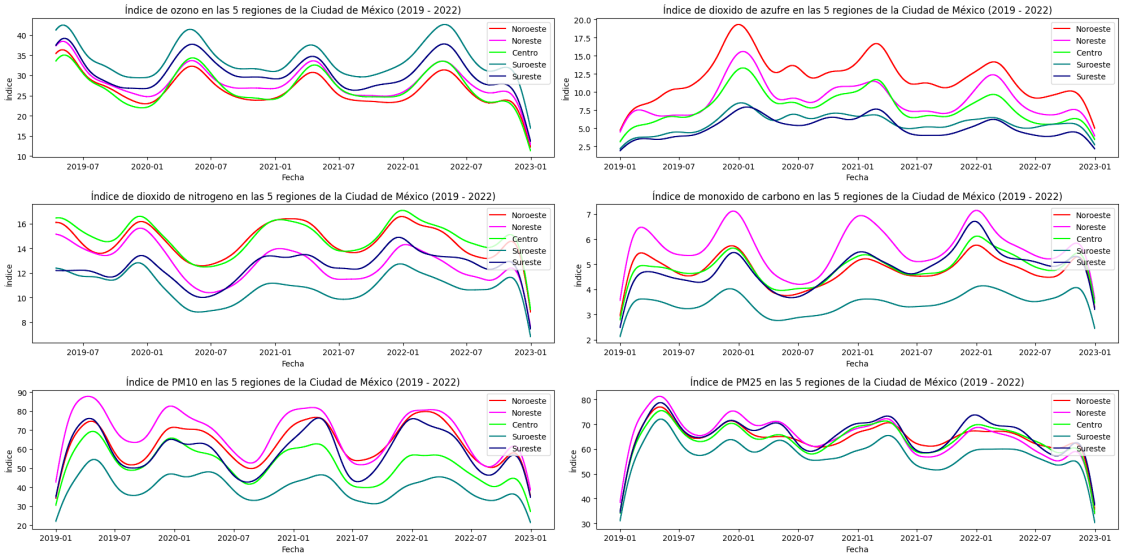
La principal ventaja del kernel gaussiano sobre un el rectangular o el triangular es que tiene una disminución más suave de los pesos. Los bordes son menos abruptos, lo que puede ser una propiedad deseable en muchas aplicaciones, como procesamiento de imágenes, aprendizaje automático y otras tareas de suavizado de datos.

[88]:
```
def gaussian_kernel(size, sigma=1): # Size = tamaño del kernel, Sigma =␣
↪Desviación Estándar de la Distribución, usualmente es 1 si no lo dan.
    size = int(size) // 2           # Divide el tamaño del kernel entre 2, ya␣
↪que el KG es simétrico.
    x = np.linspace(-size, size, 2*size+1) # Crea una matriz de numeros␣
↪espaciados uniformemente, incluyendo al cero. El total es 2*size**2.
    g = np.exp(-(x**2) / (2*sigma**2))  # Aplica la función gaussiana.
    return g / g.sum()    # Se normaliza al igual que el kernel triangular, debe␣
↪dar igual a 1.

kernel = gaussian_kernel(window_size, sigma=window_size/6) # Llama la función y␣
↪se lo aplica al tamaño ya definido, la sigma es 5000/6.
```

La función gaussiana es e^(-x² / 2 ²). Esto da una curva en forma de campana donde los valores son más altos en x=0 y disminuyen a medida que x se aleja de cero.

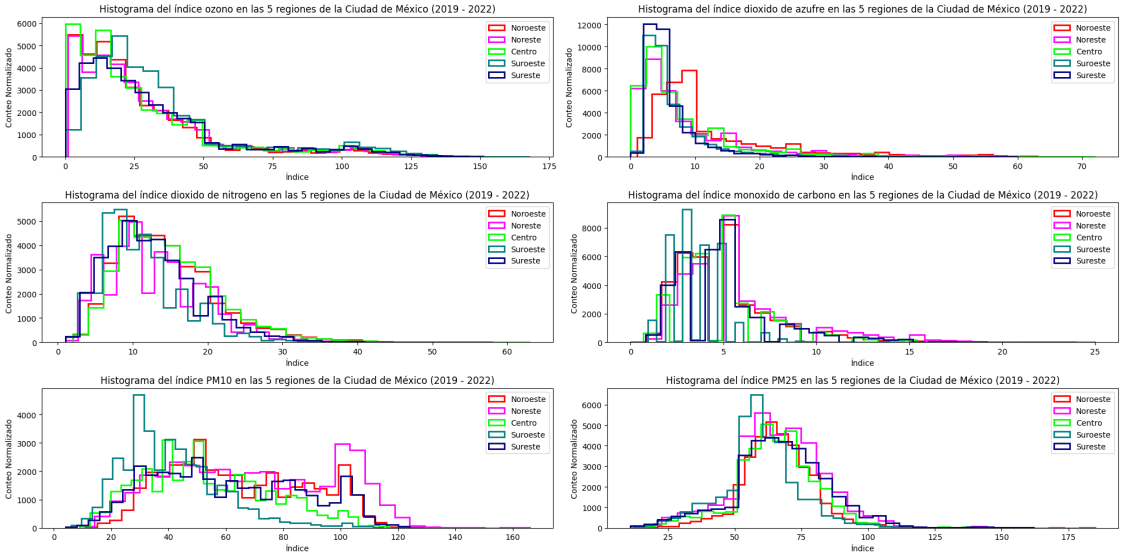
[89]:
```
# Se plotea con el mismo código de arriba, solo añade dos lineas de código.

fig, axs = plt.subplots(3, 2, figsize=(20,10))  # 3 filas, 2 columnas.
```

```
# Loop de cada gas.
for gas_index, gas in enumerate(gases):
    row = gas_index // 2
    col = gas_index % 2

    # Loop de cada región.
    for i, region in enumerate(regions):
        # Crea un nombre de columna combinando la región y el gas.
        column_name = f'{region} {gas}'

        # Uso la data interpolada.
        data = combined_data[column_name].interpolate()

        # Aplico el convolve para smoothear el data.
        smoothed_data = np.convolve(data, kernel, 'same')

        # Plot de los datos para dicha región y gas, con etiqueta y color.
        axs[row, col].plot(combined_data['Fecha'], smoothed_data, color =␣
 ↪colors[i], label = region)

    axs[row, col].set_title(f'Índice de {gas} en las 5 regiones de la Ciudad de␣
 ↪México (2019 - 2022)')
    axs[row, col].set_xlabel('Fecha')
    axs[row, col].set_ylabel('Índice')
    axs[row, col].legend(loc='upper right')

plt.tight_layout()
plt.show()
```



6. Cerremos los ojos a la evolución temporal. Despliega los histogramas de cada uno de los índices. (Para todos los contaminantes y lugares.)

[90]:
```
regions = ['Noroeste', 'Noreste', 'Centro', 'Suroeste', 'Sureste']
gases = ['ozono', 'dioxido de azufre', 'dioxido de nitrogeno', 'monoxido de␣
 ↪carbono', 'PM10', 'PM25']
colors = ['red', 'magenta', 'lime', 'teal', 'navy']

fig, axs = plt.subplots(3, 2, figsize=(20,10))  # 3 filas, 2 columnas.

# Modifica los ejes a una matriz 1D para iterar.
axs = axs.ravel()

for i, gas in enumerate(gases):
    # Loop de cada región.
    for region, color in zip(regions, colors):
        # Crea un nombre de columna combinando la región y el gas.
        column_name = f'{region} {gas}'

        # Plotea el histograma.
        axs[i].hist(combined_data[column_name].dropna(), bins=30,␣
 ↪histtype='step', label=region, color=color, alpha=1, linewidth=2)  #␣
 ↪dropna() excluye valores faltantes.

    axs[i].set_title(f'Histograma del índice {gas} en las 5 regiones de la␣
 ↪Ciudad de México (2019 - 2022)')
    axs[i].set_xlabel('Índice')
    axs[i].set_ylabel('Conteo Normalizado')
    axs[i].legend(loc='upper right')

plt.tight_layout()
plt.show()
```


Histograma del índice ozono en las 5 regiones de la Ciudad de México (2019 - 2022)
Histograma del índice dioxido de azufre en las 5 regiones de la Ciudad de México (2019 - 2022)
Histograma del índice dioxido de nitrogeno en las 5 regiones de la Ciudad de México (2019 - 2022)
Histograma del índice monoxido de carbono en las 5 regiones de la Ciudad de México (2019 - 2022)
Histograma del índice PM10 en las 5 regiones de la Ciudad de México (2019 - 2022)
Histograma del índice PM25 en las 5 regiones de la Ciudad de México (2019 - 2022)
Conteo Normalizado
Indice
Noroeste
Noreste
Centro
Suroeste
Sureste