analisis_contaminantes

October 7, 2022

1 Análisis contaminantes registrados

Nota: hacer el trabajo con el nuevo conjunto de datos llamado "CSV_Datos_Ok.csv"

[ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime

guardar_imagenes = True
```

[ ]:
```
datosContaminantes = pd.read_csv("CSV_Datos_OK.csv", parse_dates={"Fecha":
 ↪["Año", "Mes", "Día"]})
datosContaminantes = datosContaminantes.drop(["Minuto", "Segundo"], axis=1)
datosContaminantes
```

[ ]:
```
              Fecha  Hora  valor_contaminante  calidad_contaminante  calidad  \
0       2012-11-27    12             43.0000                     1        1
1       2012-11-27    13             22.0000                     1        1
2       2012-11-27    14             31.0000                     1        1
3       2012-11-27    15             29.0000                     1        1
4       2012-11-27    16             32.0000                     1        1
…              …   …                   …                     …       …
1185035 2022-07-31    19             10.6495                     1        1
1185036 2022-07-31    20              9.4167                     1        1
1185037 2022-07-31    21             15.2623                     1        1
1185038 2022-07-31    22             16.4119                     1        1
1185039 2022-07-31    23             16.3906                     1        1

         estacion contaminante
0              12         pm25
1              12         pm25
2              12         pm25
3              12         pm25
4              12         pm25
…           …          …
1185035        86         pm25
```

```
1185036         86          pm25
1185037         86          pm25
1185038         86          pm25
1185039         86          pm25

[1185040 rows x 7 columns]
```

[ ]:
```python
# Funcion para convertir valores str en float
def ToFloat(df):
    if isinstance(df, str):
        if df.count('.') == 2:
            return float( df.replace('.', '', 1) )
        else:
            return float(df)
    else:
        return float(df)

# Funcion para extraer los dias de la semana
def DayOfWeek(df):
    return df.strftime('%A')
```

[ ]:
```python
# Reemplazando valores en la columna a aplicar la funcion ToFloat
datosContaminantes["valor_contaminante"] =␣
 ↪datosContaminantes["valor_contaminante"].apply(ToFloat)

# insertando los dias
datosContaminantes.insert(1, "Dia_Semana", np.array(␣
 ↪datosContaminantes["Fecha"].apply(DayOfWeek) ) )

datosContaminantes
```

[ ]:
```
              Fecha Dia_Semana  Hora  valor_contaminante  calidad_contaminante  \
0        2012-11-27    Tuesday    12             43.0000                     1
1        2012-11-27    Tuesday    13             22.0000                     1
2        2012-11-27    Tuesday    14             31.0000                     1
3        2012-11-27    Tuesday    15             29.0000                     1
4        2012-11-27    Tuesday    16             32.0000                     1
…               …          …   …                   …                     …
1185035  2022-07-31     Sunday    19             10.6495                     1
1185036  2022-07-31     Sunday    20              9.4167                     1
1185037  2022-07-31     Sunday    21             15.2623                     1
1185038  2022-07-31     Sunday    22             16.4119                     1
1185039  2022-07-31     Sunday    23             16.3906                     1

          calidad  estacion contaminante
0               1        12         pm25
1               1        12         pm25
```

```
2              1        12          pm25
3              1        12          pm25
4              1        12          pm25
…            …        …           …
1185035        1        86          pm25
1185036        1        86          pm25
1185037        1        86          pm25
1185038        1        86          pm25
1185039        1        86          pm25

[1185040 rows x 8 columns]
```

Arreglamos los datos que presenten inconsistencias, ordenamos de menor a mayor por fechas y borramos columna de índice duplicada tras la última operación:

[ ]:
```
datosContaminantes =␣
 ↪datosContaminantes[(datosContaminantes["valor_contaminante"] >= 0.0) & \
                                                ␣
 ↪(datosContaminantes["valor_contaminante"] != 999.0) & \
                                                ␣
 ↪(datosContaminantes["calidad_contaminante"] != 151)]
datosContaminantes
```

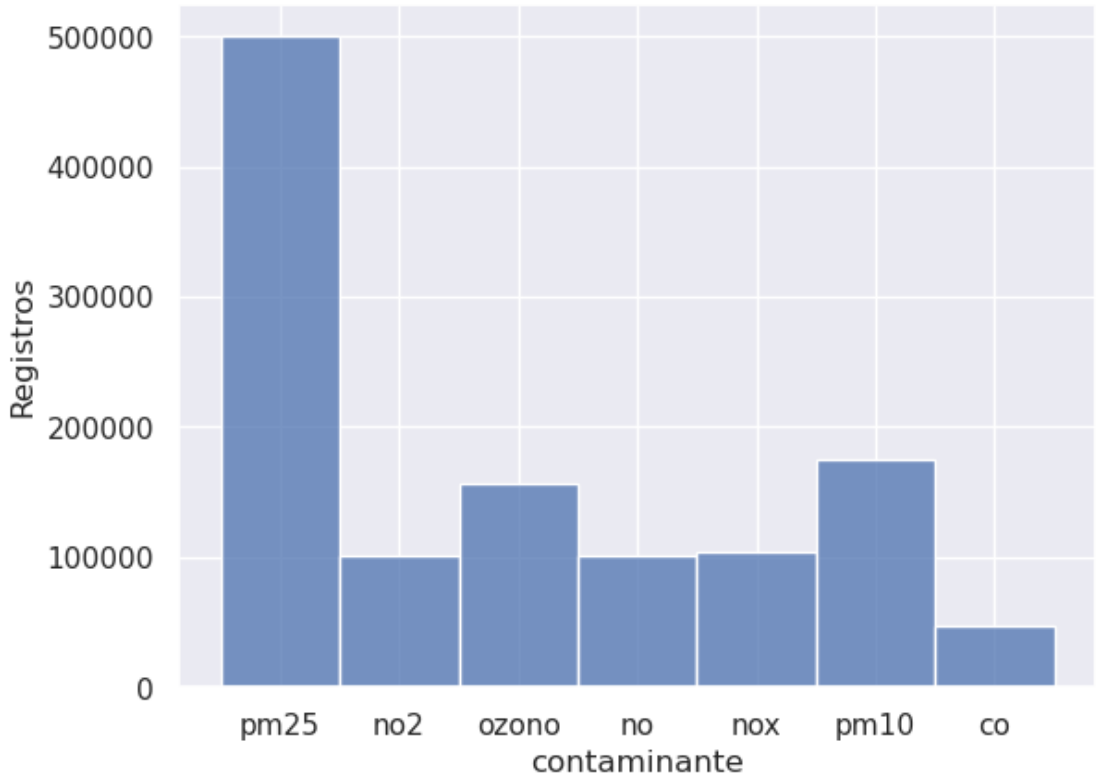
[ ]:
```
             Fecha Dia_Semana  Hora  valor_contaminante  calidad_contaminante  \
0       2012-11-27    Tuesday    12             43.0000                     1
1       2012-11-27    Tuesday    13             22.0000                     1
2       2012-11-27    Tuesday    14             31.0000                     1
3       2012-11-27    Tuesday    15             29.0000                     1
4       2012-11-27    Tuesday    16             32.0000                     1
…              …        …   …                  …                       …
1185035 2022-07-31     Sunday    19             10.6495                     1
1185036 2022-07-31     Sunday    20              9.4167                     1
1185037 2022-07-31     Sunday    21             15.2623                     1
1185038 2022-07-31     Sunday    22             16.4119                     1
1185039 2022-07-31     Sunday    23             16.3906                     1

         calidad  estacion contaminante
0              1        12          pm25
1              1        12          pm25
2              1        12          pm25
3              1        12          pm25
4              1        12          pm25
…            …        …           …
1185035        1        86          pm25
1185036        1        86          pm25
1185037        1        86          pm25
1185038        1        86          pm25
```

```
1185039        1        86         pm25

[1185040 rows x 8 columns]
```

[ ]:
```
datosContaminantes = datosContaminantes.sort_values(by=["Fecha", "Hora"]).
 ↪reset_index().drop("index", axis=1)
datosContaminantes
```

[ ]:
```
             Fecha Dia_Semana  Hora  valor_contaminante  calidad_contaminante  \
0       2012-09-18    Tuesday    12             16.0000                     1
1       2012-09-18    Tuesday    13             18.0000                     1
2       2012-09-18    Tuesday    14             18.0000                     1
3       2012-09-18    Tuesday    15             17.0000                     1
4       2012-09-18    Tuesday    16             17.0000                     1
…              …        …   …                  …                     …
1185035 2022-08-31  Wednesday    19            113.3006                     1
1185036 2022-08-31  Wednesday    20             46.5652                     1
1185037 2022-08-31  Wednesday    21             51.3882                     1
1185038 2022-08-31  Wednesday    22             88.6126                     1
1185039 2022-08-31  Wednesday    23             61.1303                     1

         calidad  estacion contaminante
0              1        25         pm25
1              1        25         pm25
2              1        25         pm25
3              1        25         pm25
4              1        25         pm25
…          …        …           …
1185035        1         6          nox
1185036        1         6          nox
1185037        1         6          nox
1185038        1         6          nox
1185039        1         6          nox

[1185040 rows x 8 columns]
```

[ ]:
```
sns.set_theme(style='dark')
sns.color_palette("dark:salmon_r", as_cmap=True)

sns.histplot(data=datosContaminantes, x="contaminante")

plt.ylabel('Registros')
sns.set(rc={'figure.figsize':(9, 8)})

#plt.savefig('registros_contaminantes.png', bbox_inches='tight', dpi=200)
plt.grid(); plt.show()
```



1.1 Muestra aleatoria de tamaño definido

Se ha elegido un tamaño de muestra representativo de $n = 1600$ datos

```
[ ]: muestraContaminantes = datosContaminantes.sample(n=1600)  # muestra aleatoria␣
      ↪de tamaño 1600
     muestraContaminantes
```

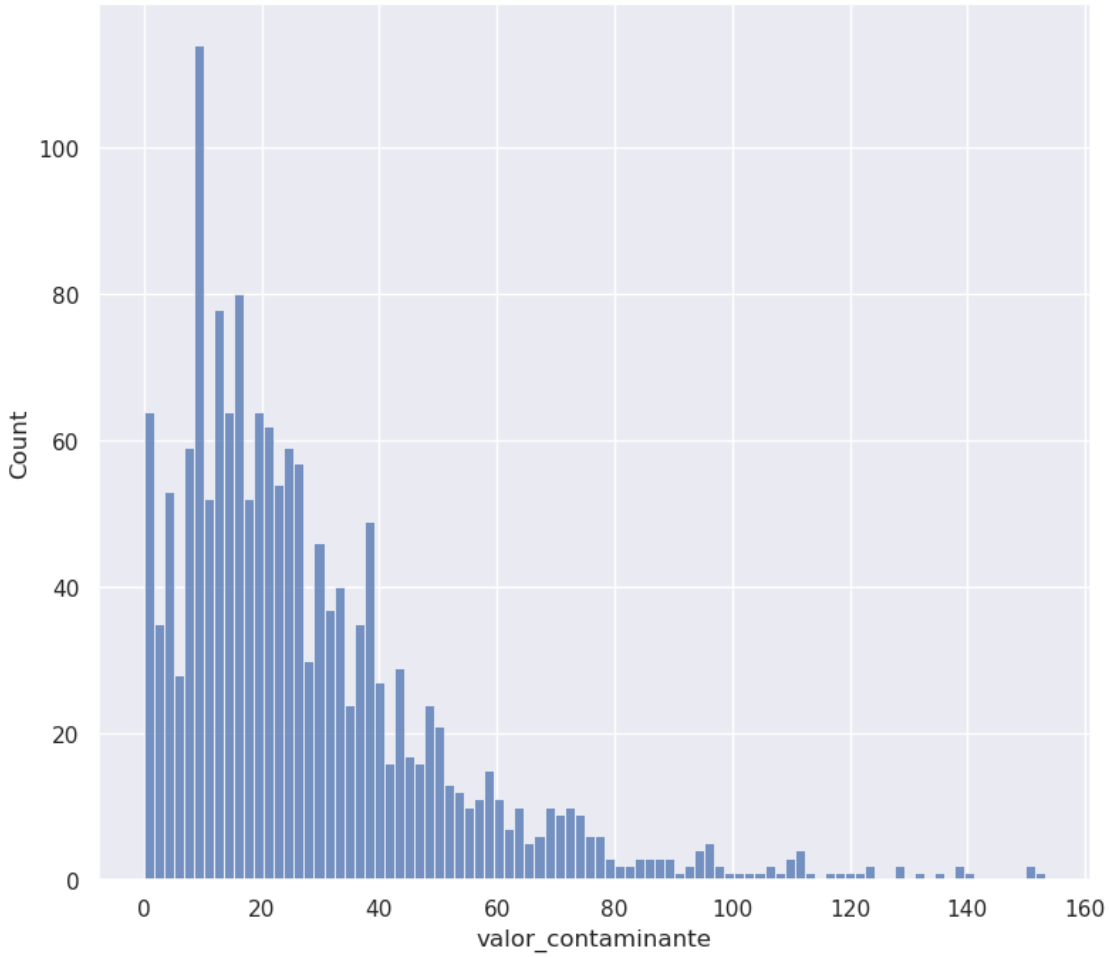
```
[ ]:              Fecha Dia_Semana  Hora  valor_contaminante  calidad_contaminante  \
     524604   2018-07-03    Tuesday     3              3.0000                     1
     1039272  2021-06-28     Monday    20             64.0828                     1
     1077890  2021-10-06  Wednesday    16             27.0000                     1
     400592   2017-11-11   Saturday     1             10.7161                     1
     347691   2017-07-16     Sunday     2              0.0111                     1
     …              …         …   …                  …                          …
     272554   2017-01-06     Friday     1             32.8443                     1
     341066   2017-06-29   Thursday     4              8.4608                     1
     390941   2017-10-23     Monday    22             68.2465                     1
     79539    2015-05-15     Friday     2             37.0000                     1
     576586   2018-10-11   Thursday     2             19.7762                     1

              calidad  estacion contaminante
```

```
524604          1         44          pm25
1039272         1          6           nox
1077890         1         12          pm25
400592          1         25            no
347691          1         41         ozono
…           …         …           …
272554          1         48          pm25
341066          1         25            no
390941          1          6          pm10
79539           1         48          pm25
576586          1          6           nox

[1600 rows x 8 columns]
```

[ ]:
```
# ordenar los valores de la muestra
muestraContaminantes = muestraContaminantes.sort_values(by=["Fecha", "Hora"]).
↪reset_index().drop("index", axis=1)
muestraContaminantes
```

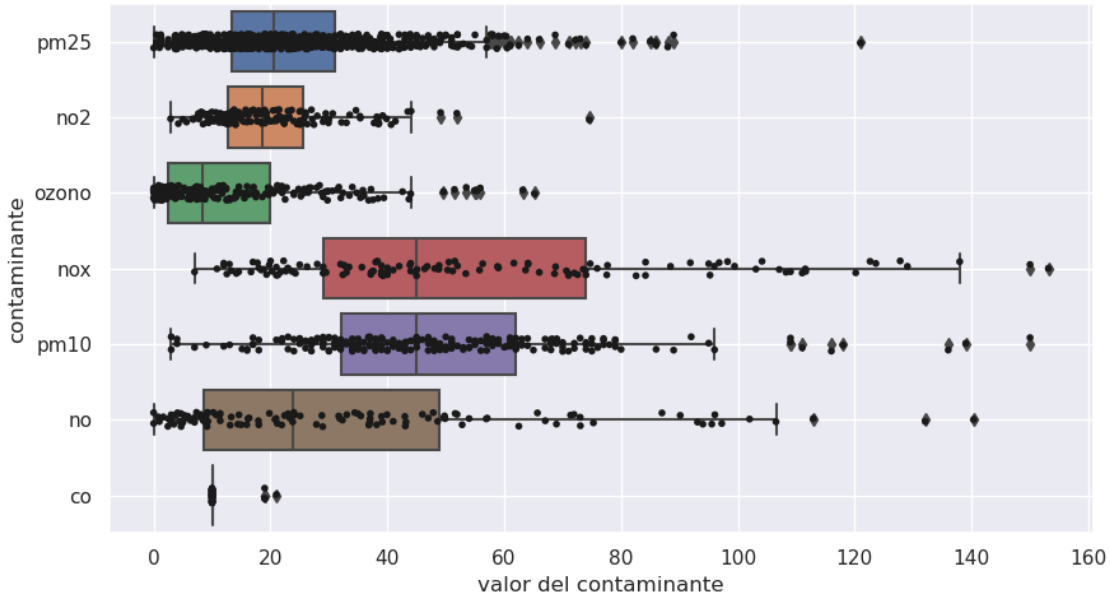
[ ]:
```
           Fecha Dia_Semana  Hora  valor_contaminante  calidad_contaminante  \
0     2012-09-19  Wednesday    12             18.0000                     1
1     2012-12-02     Sunday    14             19.0000                     1
2     2012-12-23     Sunday    20             35.0000                     1
3     2012-12-25    Tuesday     3             43.0000                     1
4     2013-01-03   Thursday     2             32.0000                     1
…          …         …   …                  …                     …
1595 2022-07-24     Sunday    23              4.8690                     1
1596 2022-07-26    Tuesday     7             10.5203                     1
1597 2022-07-29     Friday    17              3.6992                     1
1598 2022-07-30   Saturday    17              1.2315                     1
1599 2022-07-31     Sunday     5             10.6736                     1

      calidad  estacion contaminante
0           1        25         pm25
1           1        25         pm25
2           1        12         pm25
3           1        25         pm25
4           1        25         pm25
…       …        …           …
1595        1        85         pm25
1596        1        79         pm25
1597        1        80         pm25
1598        1        83         pm25
1599        1        80         pm25

[1600 rows x 8 columns]
```

[ ]:
```
sns.set_theme(style='dark')

sns.histplot(muestraContaminantes, x='valor_contaminante', bins=90)
sns.set(rc={'figure.figsize':(9, 5)}); plt.grid(); plt.show()
```



[ ]:
```
sns.set_theme(style='dark')

sns.boxplot(data=muestraContaminantes, x='valor_contaminante', y='contaminante')

sns.stripplot(data=muestraContaminantes, x='valor_contaminante',␣
 ↪y='contaminante',
              size=4, color='k', linewidth=0)

sns.set(rc={'figure.figsize':(9, 7)})
plt.xlabel("valor del contaminante"); plt.grid()
plt.tight_layout()
```

```
# para guardar el plot
if guardar_imagenes:
    plt.savefig('boxplots_contaminantes.png', bbox_inches='tight', dpi=200)

plt.show()
```

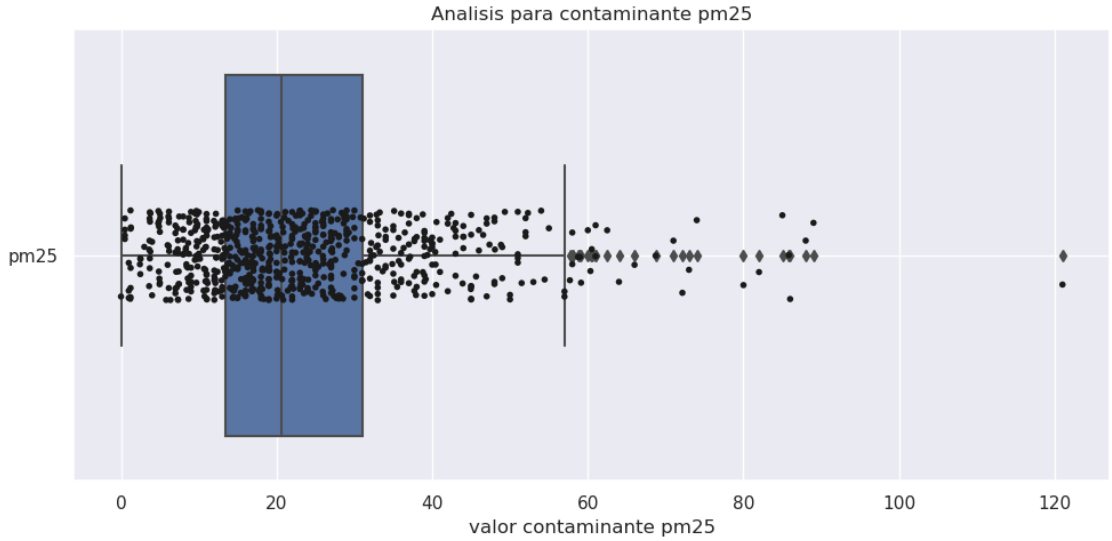


1.2 Contaminante pm2.5 para la muestra aleatoria

Veamos la distribución del contaminante pm2.5 para la muestra aleatoria extraída de tamaño $n = 1000$ representando los datos mediante un diagrama de cajas y bigotes

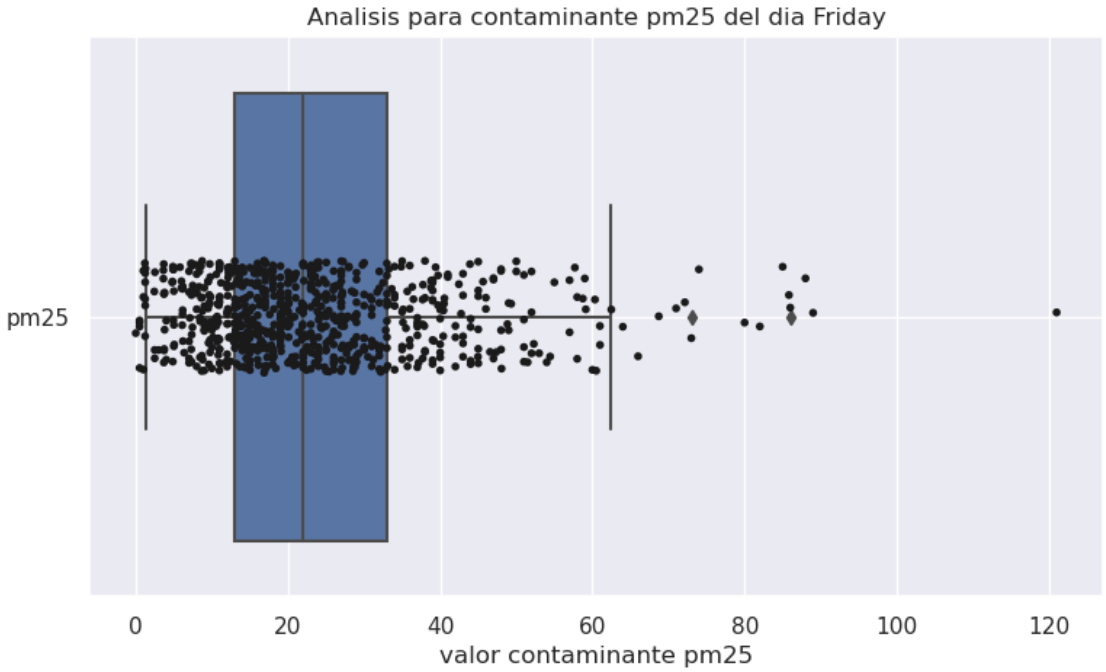
[ ]:
```
sns.set_theme(style='dark')

sns.boxplot(
    data = muestraContaminantes[ muestraContaminantes['contaminante'] == 'pm25'␣
 ↪],
    x = 'valor_contaminante',
    y = 'contaminante'
)

sns.stripplot(
    data = muestraContaminantes[ muestraContaminantes['contaminante'] == 'pm25'␣
 ↪],
    x='valor_contaminante',
    y='contaminante',
    size=4, color='k', linewidth=0
)
```

```
sns.set(rc={'figure.figsize':(9, 5)})

plt.title("Analisis para contaminante pm25")
plt.ylabel(''); plt.xlabel('valor contaminante pm25'); plt.grid()
plt.tight_layout()

# para guardar el plot
if guardar_imagenes:
    plt.savefig("boxplot_pm25.png", bbox_inches='tight', dpi=200)

plt.show()
```



[ ]:
```
dia = 'Friday'

sns.set_theme(style='dark')

sns.boxplot(data=muestraContaminantes[ (muestraContaminantes['contaminante'] ==␣
 ↪'pm25') & \
                                       (muestraContaminantes['Dia_Semana'] ==␣
 ↪dia) ],
            x='valor_contaminante', y='contaminante')

sns.stripplot(data=muestraContaminantes[ muestraContaminantes['contaminante']␣
 ↪== 'pm25' ],
              x='valor_contaminante',
              y='contaminante',
              size=4, color='k', linewidth=0)
```

```
sns.set(rc={'figure.figsize':(12, 6)})

plt.title(f"Analisis para contaminante pm25 del dia {dia}")
plt.ylabel(''); plt.xlabel('valor contaminante pm25'); plt.grid(); plt.show()
```

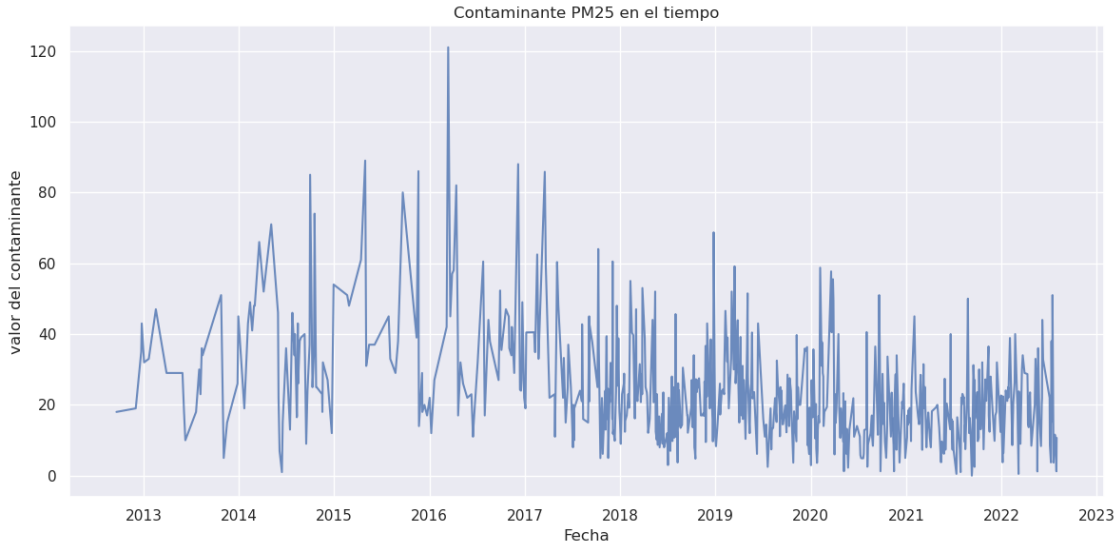


1.3 Comportamiento en el tiempo de los contaminantes

- **PM25**

[ ]:
```
sns.lineplot(
    data = muestraContaminantes[muestraContaminantes["contaminante"] == "pm25"],
    x = "Fecha",
    y = "valor_contaminante",
    ci = False,
    alpha = 0.8
)

sns.set(rc={'figure.figsize':(9, 5)})

plt.title("Contaminante PM25 en el tiempo"); plt.ylabel("valor del␣
 ↪contaminante")
plt.tight_layout()

# para guardar el plot
```

```
if guardar_imagenes:
    plt.savefig("muestraPM25_tiempo.png", bbox_inches='tight', dpi=200)

plt.show()
```



1.4 Tendencia

Para visualizar con mejor detalle lo que sucede en términos de la tendencia del contaminante, veamos con una linea de tendencia los datos en su forma de dispersión

[ ]:
```
def toTimestamp(df):
    return datetime.timestamp(df)
```

Para calcular la linea de tendencia, es necesario que los datos en el eje x sean de tipo `timestamp`, de esta forma los calculos serán permitidos y podremos determinar el orden de ajuste.

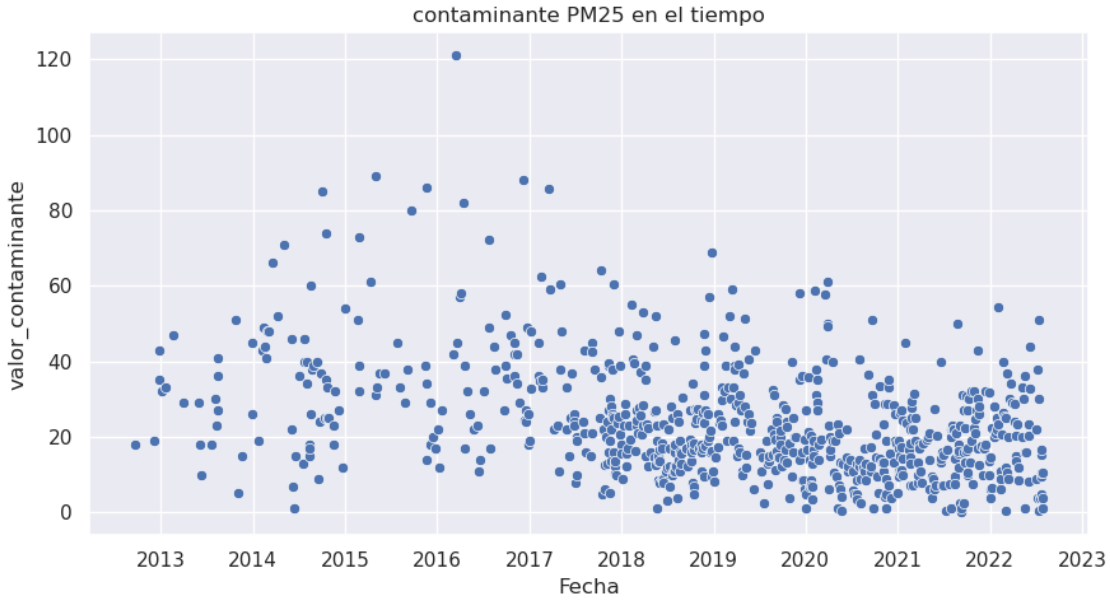
[ ]:
```
# para calcular la linea de tendencia se requieren datos tipo
# 'timestamp' en el eje x
muestraFechaTimestamp = muestraContaminantes.copy()
muestraFechaTimestamp["Fecha"] = muestraFechaTimestamp["Fecha"].
 ↪apply(toTimestamp)

muestraFechaTimestamp
```

[ ]:

	Fecha	Dia_Semana	Hora	valor_contaminante	calidad_contaminante	\
0	1.348031e+09	Wednesday	12	18.0000	1	
1	1.354424e+09	Sunday	14	19.0000	1	
2	1.356239e+09	Sunday	20	35.0000	1	
3	1.356412e+09	Tuesday	3	43.0000	1	

```
4     1.357189e+09   Thursday     2              32.0000                    1
…              …         …    …                 …                        …
1595  1.658639e+09     Sunday    23               4.8690                    1
1596  1.658812e+09    Tuesday     7              10.5203                    1
1597  1.659071e+09     Friday    17               3.6992                    1
1598  1.659157e+09   Saturday    17               1.2315                    1
1599  1.659244e+09     Sunday     5              10.6736                    1

      calidad  estacion contaminante
0           1        25         pm25
1           1        25         pm25
2           1        12         pm25
3           1        25         pm25
4           1        25         pm25
…       …        …          …
1595        1        85         pm25
1596        1        79         pm25
1597        1        80         pm25
1598        1        83         pm25
1599        1        80         pm25

[1600 rows x 8 columns]
```

[ ]:
```
sns.scatterplot(
    data = muestraContaminantes[muestraContaminantes["contaminante"] == "pm25"],
    x = "Fecha",
    y = "valor_contaminante"
)

sns.set(rc={'figure.figsize':(10, 5)})

plt.title("contaminante PM25 en el tiempo")
plt.tight_layout(); plt.show()
```



Veamos, para los distintos contaminantes, una linea de tendencia que se ajuste a los datos, para un orden y un contaminante determinado con la siguiente función que permite graficar estos datos:

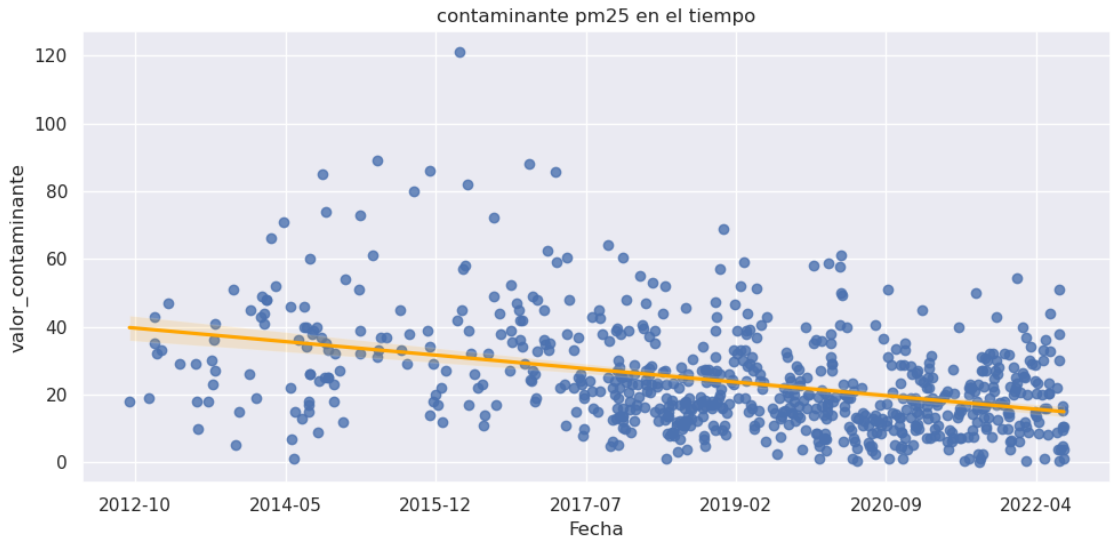
[ ]:
```
def scatterFit(contaminante, ordenAjuste=1, save=False):

    # dispersion y linea de tendencia para los datos con timestamp
    sns.regplot(
        data=muestraFechaTimestamp[muestraFechaTimestamp["contaminante"] ==␣
↪contaminante],
        x="Fecha",
        y="valor_contaminante",
        line_kws={"color": "orange"},
        order=ordenAjuste  # orden del ajuste
    )

    # configurar las fechas como datetime para el eje x
    ax = plt.gca()
    xticks = ax.get_xticks()
    xticks_dates = [datetime.fromtimestamp(x).strftime('%Y-%m') for x in xticks]
    ax.set_xticklabels(xticks_dates)

    sns.set(rc={'figure.figsize':(10, 5)})

    plt.title(f"contaminante {contaminante} en el tiempo")
    plt.tight_layout();

    if save == True:
```

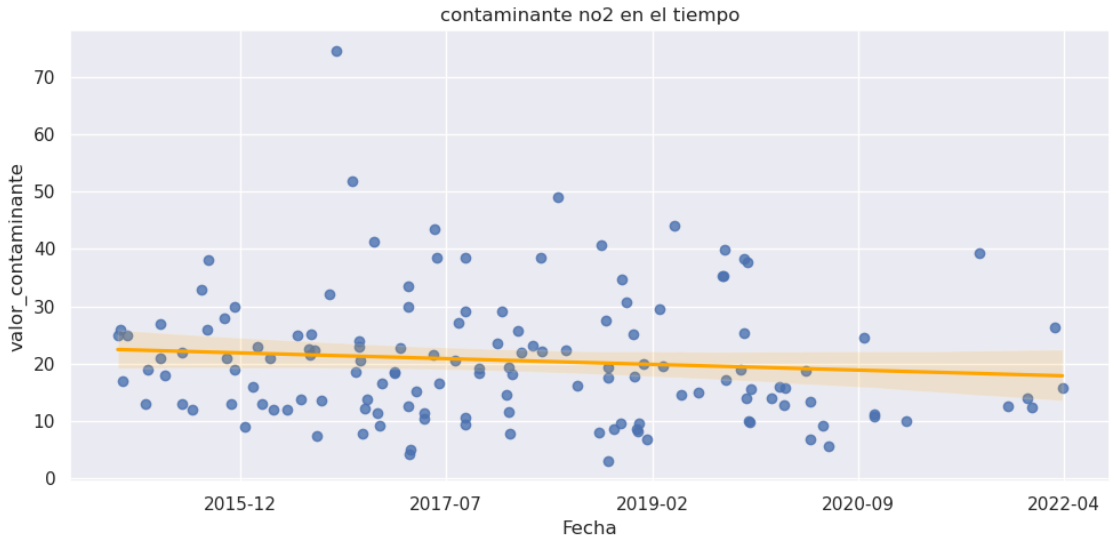
```
        plt.savefig(f'trendline{contaminante}.png', bbox_inches='tight',␣
↪dpi=200)

    plt.show()

    pass
```

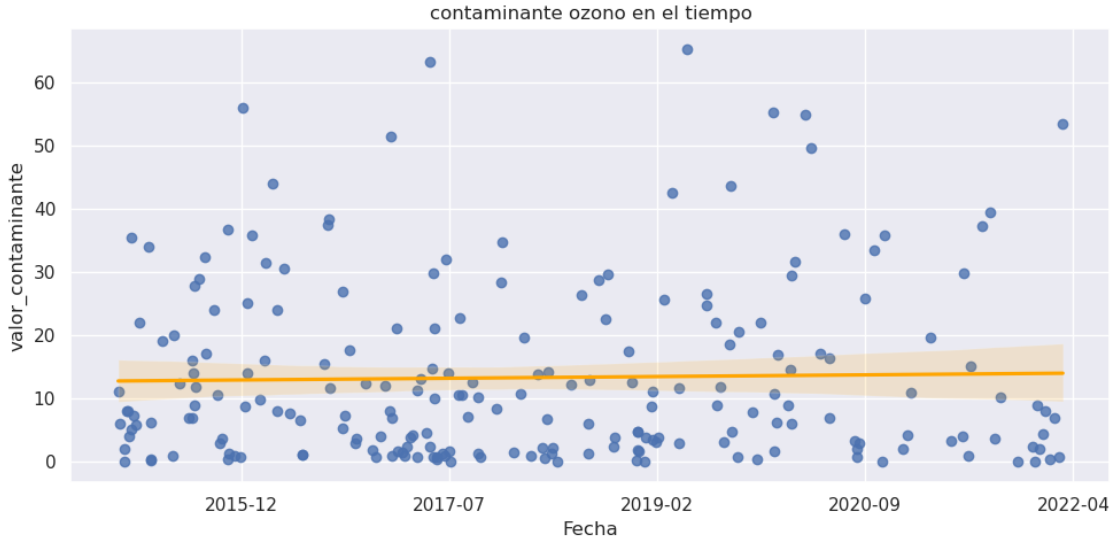
Veamos los ajustes para cada uno de los contaminantes:

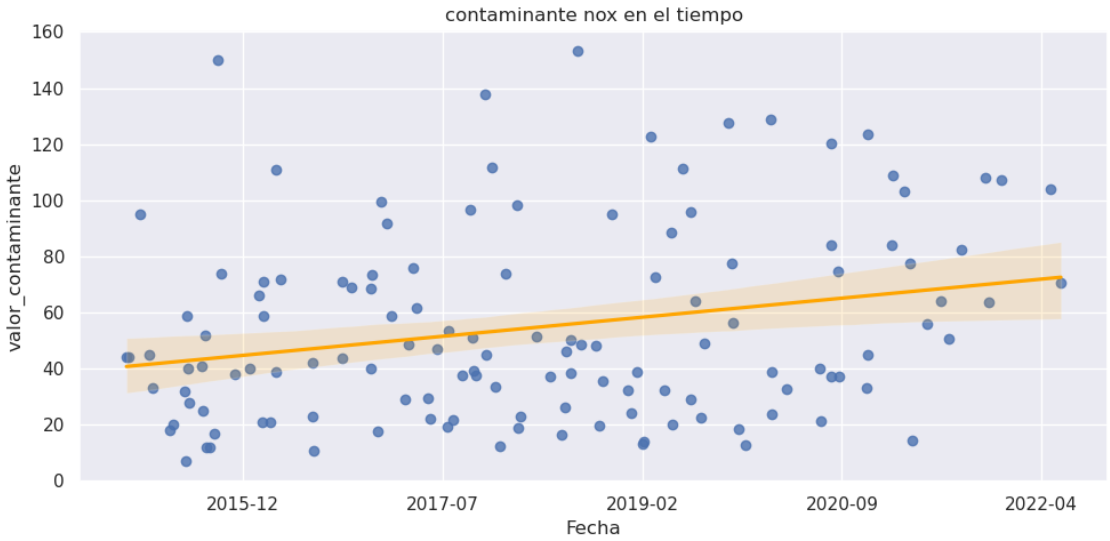
[ ]:
```
contaminantes = muestraContaminantes["contaminante"].unique()

for cont in contaminantes:
    scatterFit(cont, ordenAjuste=1, save=guardar_imagenes)
```

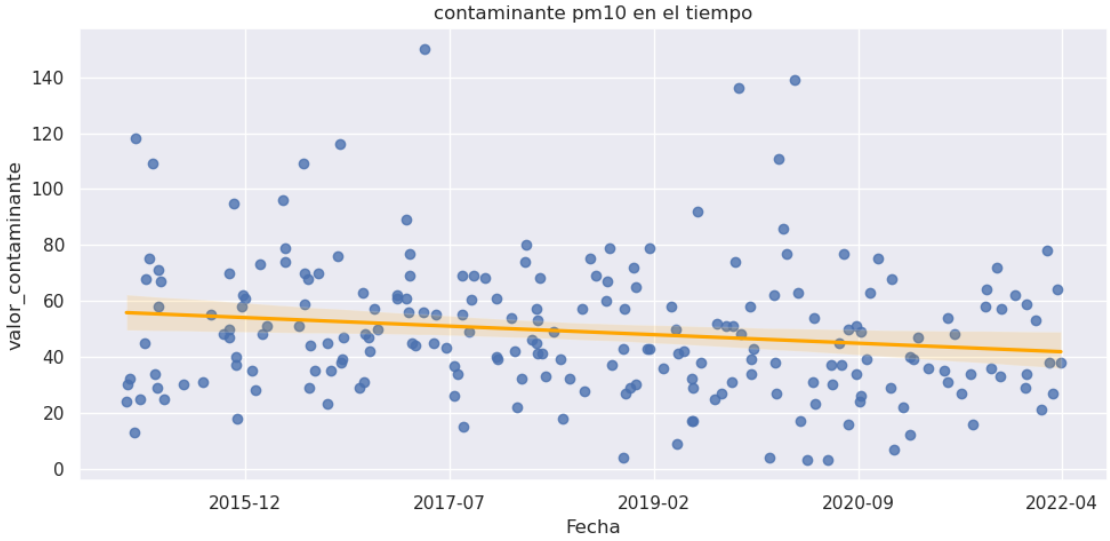
```
/tmp/ipykernel_365338/3476521949.py:16: UserWarning: FixedFormatter should only
be used together with FixedLocator
  ax.set_xticklabels(xticks_dates)
```

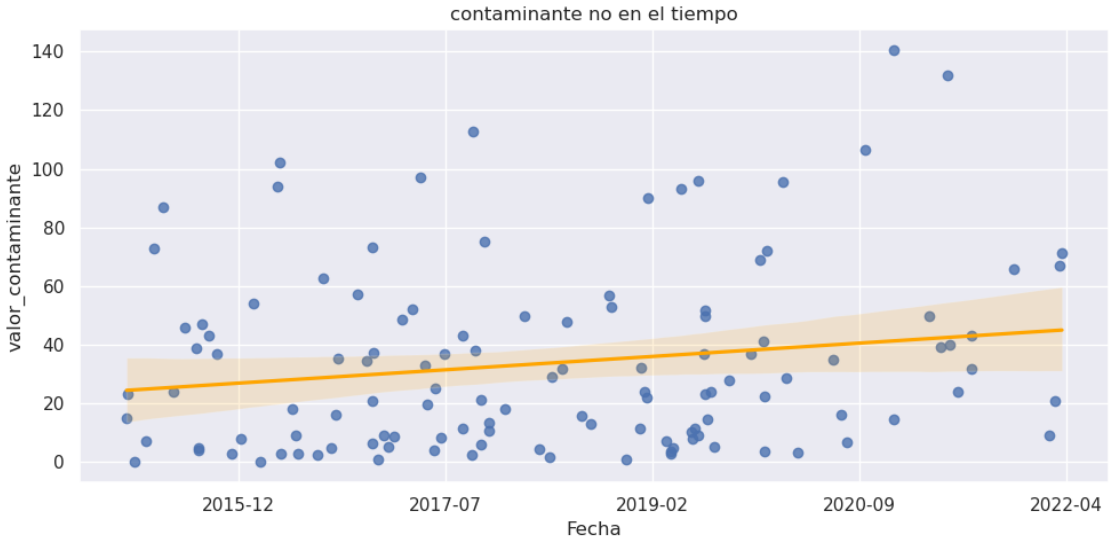


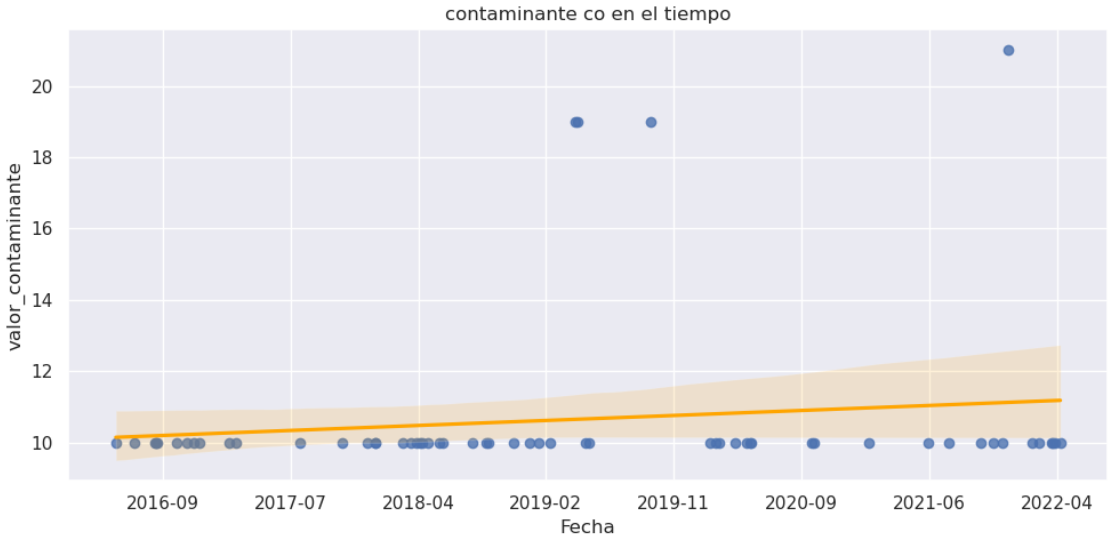

contaminante no2 en el tiempo
valor_contaminante
0
10
20
30
40
50
60
70
2015-12
2017-07
2019-02
2020-09
2022-04
Fecha


contaminante ozono en el tiempo
valor_contaminante
0
10
20
30
40
50
60
2015-12
2017-07
2019-02
2020-09
2022-04
Fecha


contaminante nox en el tiempo
valor_contaminante
Fecha
0
20
40
60
80
100
120
140
160
2015-12
2017-07
2019-02
2020-09
2022-04


contaminante pm10 en el tiempo
valor_contaminante
Fecha
0
20
40
60
80
100
120
140
2015-12
2017-07
2019-02
2020-09
2022-04





1.5 Comportamiento por semana

[ ]:
```
# obtener las semanas para cada año

def getWeek(df):
    return df.isocalendar()[1]

def getYear(df):
    return df.year
```

```
muestraContaminantes.insert( 1, "semana", muestraContaminantes["Fecha"].
 ↪apply(getWeek) )
muestraContaminantes.insert( 1, "año", muestraContaminantes["Fecha"].
 ↪apply(getYear) )
muestraContaminantes
```

[ ]:

	Fecha	año	semana	Dia_Semana	Hora	valor_contaminante	\
0	2012-09-19	2012	38	Wednesday	12	18.0000	
1	2012-12-02	2012	48	Sunday	14	19.0000	
2	2012-12-23	2012	51	Sunday	20	35.0000	
3	2012-12-25	2012	52	Tuesday	3	43.0000	
4	2013-01-03	2013	1	Thursday	2	32.0000	
…	…	…	…	…	…	…	
1595	2022-07-24	2022	29	Sunday	23	4.8690	
1596	2022-07-26	2022	30	Tuesday	7	10.5203	
1597	2022-07-29	2022	30	Friday	17	3.6992	
1598	2022-07-30	2022	30	Saturday	17	1.2315	
1599	2022-07-31	2022	30	Sunday	5	10.6736	

	calidad_contaminante	calidad	estacion	contaminante
0	1	1	25	pm25
1	1	1	25	pm25
2	1	1	12	pm25
3	1	1	25	pm25
4	1	1	25	pm25
…	…	…	…	…
1595	1	1	85	pm25
1596	1	1	79	pm25
1597	1	1	80	pm25
1598	1	1	83	pm25
1599	1	1	80	pm25

[1600 rows x 10 columns]

Unicamente para el PM25 tendremos:

[ ]:
```
muestraPM25 = muestraContaminantes[ muestraContaminantes["contaminante"] ==␣
 ↪"pm25" ]
muestraPM25
```

[ ]:

	Fecha	año	semana	Dia_Semana	Hora	valor_contaminante	\
0	2012-09-19	2012	38	Wednesday	12	18.0000	
1	2012-12-02	2012	48	Sunday	14	19.0000	
2	2012-12-23	2012	51	Sunday	20	35.0000	
3	2012-12-25	2012	52	Tuesday	3	43.0000	
4	2013-01-03	2013	1	Thursday	2	32.0000	
…	…	…	…	…	…	…	

```
1595 2022-07-24  2022      29     Sunday    23             4.8690
1596 2022-07-26  2022      30    Tuesday     7            10.5203
1597 2022-07-29  2022      30     Friday    17             3.6992
1598 2022-07-30  2022      30   Saturday    17             1.2315
1599 2022-07-31  2022      30     Sunday     5            10.6736

      calidad_contaminante  calidad  estacion contaminante
0                        1        1        25         pm25
1                        1        1        25         pm25
2                        1        1        12         pm25
3                        1        1        25         pm25
4                        1        1        25         pm25
…                      …      …        …           …
1595                     1        1        85         pm25
1596                     1        1        79         pm25
1597                     1        1        80         pm25
1598                     1        1        83         pm25
1599                     1        1        80         pm25

[724 rows x 10 columns]
```

Calculando los promedios por estación, para cada semana de cada año:

[ ]:
```
estaciones = muestraPM25["estacion"].unique()
contaminantes = muestraPM25["contaminante"].unique()
semanas = muestraPM25["semana"].unique()
years = muestraPM25["año"].unique()
```

[ ]:
```
promedio_semana = []; newYears = []; newSemanas = []; newEstaciones = []

for year in years:
    for semana in semanas:
        for estacion in estaciones:
            promedio_semana.append(
                np.mean(
                    muestraPM25[
                        (muestraPM25["año"] == year) & \
                        (muestraPM25["semana"] == semana) & \
                        (muestraPM25["estacion"] == estacion)
                    ]["valor_contaminante"]
                )
            )
        newEstaciones += list(estaciones)
        newSemanas += len(estaciones)*[semana]
        newYears += len(estaciones)*[year]
```

[ ]:
```
# funcion para concatenar dos listas, elemento a elemento
def addLists(firstList, secondList):
    finalList= []
    for i in range( len(firstList) ):
        finalList.append( str(firstList[i]) + '-' + str(secondList[i]) )
    return finalList
```

[ ]:
```
prom_muestra_pm25 = {
    "fecha":addLists(newYears, newSemanas),
    "año":newYears,
    "semana":newSemanas,
    "estacion":newEstaciones,
    "promedio_contaminante":promedio_semana
}

prom_muestra_pm25_df = pd.DataFrame(data=prom_muestra_pm25)

prom_muestra_pm25_df = prom_muestra_pm25_df.dropna()

prom_muestra_pm25_df
```

[ ]:

	fecha	año	semana	estacion	promedio_contaminante
0	2012-38	2012	38	25	18.0
10	2012-48	2012	48	25	19.0
21	2012-51	2012	51	12	35.0
30	2012-52	2012	52	25	43.0
560	2013-52	2013	52	25	26.0
…	…	…	…	…	…
5792	2022-11	2022	11	48	30.0
5821	2022-28	2022	28	12	30.0
5822	2022-28	2022	28	48	51.0
5823	2022-28	2022	28	44	38.0
5829	2022-28	2022	28	80	0.5

[642 rows x 5 columns]

Nota: queda pendiente ordenar por año y semana.

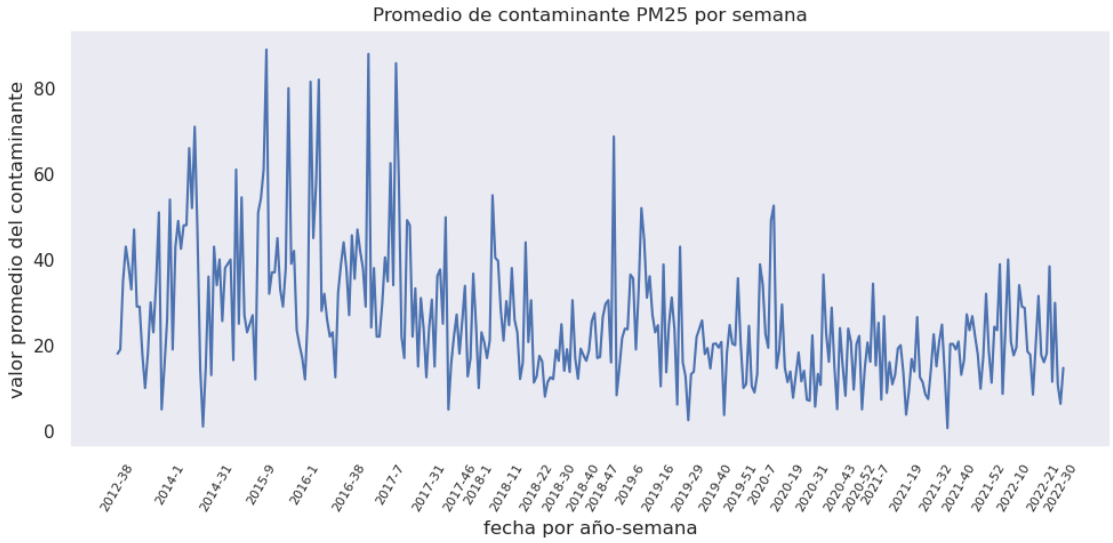
[ ]:
```
prom_muestra_pm25_df = prom_muestra_pm25_df.sort_values(by=["año", "semana"]).
  ↪reset_index().drop("index", axis=1)
prom_muestra_pm25_df
```

[ ]:

	fecha	año	semana	estacion	promedio_contaminante
0	2012-38	2012	38	25	18.0000
1	2012-48	2012	48	25	19.0000
2	2012-51	2012	51	12	35.0000
3	2012-52	2012	52	25	43.0000

```
4     2013-1  2013       1        25                32.0000
..       …    …     …       …                   …
637   2022-29  2022      29        83                15.3943
638   2022-30  2022      30        79                10.5203
639   2022-30  2022      30        83                 1.2315
640   2022-30  2022      30        80                 7.1864
641   2022-52  2022      52        84                14.6964

[642 rows x 5 columns]
```

[ ]:
```
sns.set_theme(style="dark")

sns.lineplot(
    data=prom_muestra_pm25_df,
    x="fecha",
    y="promedio_contaminante",
    ci = False
)

elements = [ np.array(prom_muestra_pm25_df["fecha"])[i] for i in range(0,␣
 ↪len(prom_muestra_pm25_df["fecha"]), 20) ]

plt.xticks(elements, fontsize=8, rotation=60)
plt.title("Promedio de contaminante PM25 por semana")
plt.xlabel("fecha por año-semana"); plt.ylabel("valor promedio del␣
 ↪contaminante")
plt.tight_layout()

# save figure
if guardar_imagenes:
    plt.savefig("muestra_pm25_semana.png", bbox_inches="tight", dpi=200)

plt.show()
```



Adición de la **media movil** para una cantidad n de intervalo:

[ ]:
```
valores = 5
media_movil = prom_muestra_pm25_df["promedio_contaminante"].rolling(valores).
 ↪mean()

sns.set_theme(style="dark")

sns.lineplot(
    data = prom_muestra_pm25_df,
    x = "fecha",
    y = "promedio_contaminante",
    ci = False,
    alpha = 0.3,
    label = "promedio PM2.5 por semana"
)

plt.plot(
    np.array(prom_muestra_pm25_df["fecha"]),
    media_movil,
    label = f"media móvil a {valores} valores",
)

elements = [ np.array(prom_muestra_pm25_df["fecha"])[i] for i in range(0,␣
 ↪len(prom_muestra_pm25_df["fecha"]), 20) ]

plt.xticks(elements, fontsize=8, rotation=60)
plt.title("Promedio de contaminante PM25 por semana")
```

```
plt.xlabel("fecha por año-semana"); plt.ylabel("valor promedio del␣
 ↪contaminante")

plt.legend(); plt.tight_layout();

# save figure
if guardar_imagenes:
    plt.savefig("movil_pm25_semana.png", bbox_inches="tight", dpi=200)

plt.show()
```