Proyecto de Emisiones Contaminantes en Madrid - Grupo 2

Integrantes

- Amada Vargas
- Felipe Bravo
- Sebastián Garrido
- Patricio Zuñiga
- Camilo Valenzuela

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime
%matplotlib notebook
```

```
diccionario_estacion = {'1': 'Pº. Recoletos','2': 'Glta. de Carlos V','35': 'Pza. del Carmen'
                        '39': 'Barrio del Pilar', '6': 'Pza. Dr. Marañón', '7': 'Pza. M. de S
                        '9': 'Pza. Luca de Tena', '38': 'Cuatro Caminos', '11': 'Av. Ramón y
                        '40': 'Vallecas', '14': 'Pza. Fdez. Ladreda', '15': 'Pza. Castilla',
                        '18': 'Calle Farolillo', '19': 'Huerta Castañeda', '36': 'Moratalaz',
                        '22': 'Po. Pontones', '23': 'Final C/ Alcalá', '24': 'Casa de Campo',
                        '26': 'Urb. Embajada (Barajas)', '27': 'Barajas', '47': 'Méndez Álvar
                        '50': 'Pza. Castilla', '54': 'Ensanche Vallecas', '55': 'Urb. Embajac
                        '57': 'Sanchinarro', '58': 'El Pardo','59': 'Parque Juan Carlos I', '
                        '10':'Partículas < 10 µm'}

diccionario_contaminante = {'1':'Dióxido de Azufre','6':'Monóxido de Carbono', '9':'Partícula
                            '12':'Óxidos de Nitrógeno', '14':'Ozono', '20':'Tolueno', '30':'E
                            '37':'Metaxileno', '38':'Paraxileno', '39':'Ortoxileno', '42':'Hi
                            '43':'Metano', '44':'Hidrocarburosno metánicos (hexano)', '7':"Mo
```

1. Generar un DataFrame con los datos de los cuatro ficheros

```
#Solución
emision_2016 = pd.read_csv('emisiones-2016.csv', sep=';')
emision_2017 = pd.read_csv('emisiones-2017.csv', sep=';')
emision_2018 = pd.read_csv('emisiones-2018.csv', sep=';')
emision_2019 = pd.read_csv('emisiones-2019.csv', sep=';')

concatenar_emisiones = [emision_2016, emision_2017, emision_2018, emision_2019]
emisiones_general = pd.concat(concatenar_emisiones)
emisiones_general
```

	PROVINCIA	MUNICIPIO	ESTACION	MAGNITUD	PUNTO_MUESTREO	ANO	MES	D01	V
0	28	79	4	1	28079004_1_38	2016	1	8.0	
1	28	79	4	1	28079004_1_38	2016	2	12.0	
2	28	79	4	1	28079004_1_38	2016	3	11.0	
3	28	79	4	1	28079004_1_38	2016	4	8.0	
4	28	79	4	1	28079004_1_38	2016	5	7.0	
...	...	...	...	...	...	...	...	...	
1831	28	79	60	14	28079060_14_6	2019	8	94.0	
1832	28	79	60	14	28079060_14_6	2019	9	88.0	
1833	28	79	60	14	28079060_14_6	2019	10	44.0	
1834	28	79	60	14	28079060_14_6	2019	11	41.0	
1835	28	79	60	14	28079060_14_6	2019	12	47.0	

7266 rows × 69 columns

2. Filtrar las columnas del DataFrame para quedarse con las columnas ESTACION, MAGNITUD, AÑO, MES y las correspondientes a los días D01, D02, etc.

```
#Solución
new = []
columnas = emisiones_general.columns.tolist()
for c in columnas:
  if c.startswith('P') or c.startswith('MUN') or c.startswith('V'):
    new.append(c)
limpio = emisiones_general.drop(new, axis=1)
limpio
```

	ESTACION	MAGNITUD	ANO	MES	D01	D02	D03	D04	D05	D06	...	D22
0	4	1	2016	1	8.0	7.0	6.0	6.0	7.0	6.0	...	10.0
1	4	1	2016	2	12.0	13.0	9.0	9.0	11.0	9.0	...	11.0
2	4	1	2016	3	11.0	10.0	9.0	9.0	7.0	8.0	...	8.0
3	4	1	2016	4	8.0	9.0	9.0	8.0	8.0	9.0	...	8.0

3. Reestructurar el DataFrame para que los valores de los contaminantes de las columnas de los días aparezcan en una única columna.

```
#Solución
new_3 = []
columnas = limpio.columns.tolist()
column = columnas[:4]
nuevo = pd.melt(limpio, id_vars=column, var_name='DIAS', value_name='VALOR')
nuevo
```

	ESTACION	MAGNITUD	ANO	MES	DIAS	VALOR
0	4	1	2016	1	D01	8.0
1	4	1	2016	2	D01	12.0
2	4	1	2016	3	D01	11.0
3	4	1	2016	4	D01	8.0
4	4	1	2016	5	D01	7.0
...	...	...	...	...	...	...
225241	60	14	2019	8	D31	98.0
225242	60	14	2019	9	D31	0.0
225243	60	14	2019	10	D31	47.0
225244	60	14	2019	11	D31	0.0
225245	60	14	2019	12	D31	4.0

225246 rows × 6 columns

4. Añadir una columna con la fecha a partir de la concatenación del año, el mes y el día (usar el módulo datetime).

Pandas cuenta con libreria de datetime, por lo que se guió en el siguiente enlace sobre la documentación de pandas:

https://pandas.pydata.org/docs/reference/api/pandas.to_datetime.html

```
#Solución
nuevo["DIAS"] = nuevo["DIAS"].str.strip('D')
nuevo["FECHA"] = nuevo["ANO"].astype(str) + "-" + nuevo["MES"].astype(str) + "-" + nuevo["DIA
nuevo['FECHA'] = pd.to_datetime(nuevo['FECHA'], format='%Y/%m/%d', errors='coerce')
nuevo
```

	ESTACION	MAGNITUD	ANO	MES	DIAS	VALOR	FECHA
0	4	1	2016	1	01	8.0	2016-01-01
1	4	1	2016	2	01	12.0	2016-02-01
2	4	1	2016	3	01	11.0	2016-03-01
3	4	1	2016	4	01	8.0	2016-04-01
4	4	1	2016	5	01	7.0	2016-05-01
...	...	...	...	...	...	...	...
225241	60	14	2019	8	31	98.0	2019-08-31
225242	60	14	2019	9	31	0.0	NaT
225243	60	14	2019	10	31	47.0	2019-10-31
225244	60	14	2019	11	31	0.0	NaT
225245	60	14	2019	12	31	4.0	2019-12-31

225246 rows × 7 columns

```
#Pruebas con librería Pandas con strip
lst = ["DAA","DAD","AAD", "ADA", "AAA", "DDA"]
df = pd.DataFrame(lst, columns = ['Ejemplo'])
df["Ejemplo"].str.strip('D')
```

```
0     AA
1      A
2     AA
3    ADA
4    AAA
5      A
Name: Ejemplo, dtype: object
```

5. Eliminar las filas con fechas no válidas (utilizar la función isnat del módulo numpy) y ordenar el Data Frame por estaciones, contaminantes y fecha.

```
#Prueba con libreria Numpy con isnat
np.isnat(nuevo["FECHA"])
```

```
0         False
1         False
2         False
3         False
4         False
          ...
225241    False
225242     True
225243    False
225244     True
225245    False
Name: FECHA, Length: 225246, dtype: bool
```

```
#Solución
#nuevo = nuevo.dropna(subset=["FECHA"]) # Forma 1 con dropnan

nuevo = nuevo.drop(nuevo[np.isnat(nuevo["FECHA"])].index) # Forma 2 con np.isnat, requerimier
nuevo = nuevo.sort_values(['ESTACION', 'MAGNITUD','FECHA']).reset_index(drop = True)
nuevo
```

	ESTACION	MAGNITUD	ANO	MES	DIAS	VALOR	FECHA
0	4	1	2016	1	01	8.0	2016-01-01
1	4	1	2016	1	02	7.0	2016-01-02
2	4	1	2016	1	03	6.0	2016-01-03
3	4	1	2016	1	04	6.0	2016-01-04
4	4	1	2016	1	05	7.0	2016-01-05
...	...	...	...	...	...	...	...
221153	60	14	2019	12	27	17.0	2019-12-27
221154	60	14	2019	12	28	13.0	2019-12-28
221155	60	14	2019	12	29	14.0	2019-12-29
221156	60	14	2019	12	30	5.0	2019-12-30
221157	60	14	2019	12	31	4.0	2019-12-31

221158 rows × 7 columns

6. Mostrar por pantalla las estaciones y los contaminantes disponibles en el DataFrame.

```
#Solución
contaminantes_disp = nuevo["MAGNITUD"].unique() #Se obtiene los codigos que contiene
estacion_disp  = nuevo["ESTACION"].unique()
```

```
print("Los contaminantes disponibles en los archivos son: ")
for contaminante in contaminantes_disp:
  print(str(contaminante) + " - " + str(diccionario_contaminante[str(contaminante)]))


print("\nLas estaciones disponibles en los archivos son: ")
for estacion in estacion_disp :
  print(str(estacion) + " - " + str(diccionario_estacion[str(estacion)]))
```

```
Los contaminantes disponibles en los archivos son:
1 - Dióxido de Azufre
6 - Monóxido de Carbono
7 - Monóxido de Nitrógeno
8 - Dióxido de Nitrógeno
12 - Óxidos de Nitrógeno
9 - Partículas < 2.5 µm
10 - Partículas < 10 µm
14 - Ozono
20 - Tolueno
30 - Benceno
35 - Etilbenceno
42 - Hidrocarburos totales(hexano)
43 - Metano
44 - Hidrocarburosno metánicos (hexano)

Las estaciones disponibles en los archivos son:
4 - Pza. de España
8 - Escuelas Aguirre
11 - Av. Ramón y Cajal
16 - Arturo Soria
17 - Villaverde Alto
18 - Calle Farolillo
24 - Casa de Campo
27 - Barajas
35 - Pza. del Carmen
36 - Moratalaz
38 - Cuatro Caminos
39 - Barrio del Pilar
40 - Vallecas
47 - Méndez Álvaro
48 - Po. Castellana
49 - Retiro
50 - Pza. Castilla
54 - Ensanche Vallecas
55 - Urb. Embajada (Barajas)
56 - Plaza Elíptica
57 - Sanchinarro
58 - El Pardo
59 - Parque Juan Carlos I
60 - Tres Olivos
```

7. Mostrar un resumen descriptivo (mímino, máximo, media, etc) para cada contaminante.

Pandas cuenta con libreria de groupby, por lo que se guió en el siguiente enlace sobre la documentación de pandas:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html

```
#Solución
nuevo.groupby(['MAGNITUD']).VALOR.describe() #Dado que las otras columnas son solo codigos y
```

MAGNITUD	count	mean	std	min	25%	50%	75%	max
1	14610.0	7.428953	7.012504	0.00	4.00	7.00	10.00	610.00
6	14610.0	0.350233	0.215935	0.00	0.20	0.30	0.40	14.90
7	35064.0	20.446412	135.123509	0.00	4.00	9.00	23.00	24742.00
8	35064.0	37.677618	20.118050	0.00	22.00	35.00	50.00	148.00
9	8948.0	10.087729	10.643591	0.00	6.00	9.00	13.00	850.00
10	17897.0	18.772923	35.723619	0.00	10.00	16.00	24.00	4481.00
12	35064.0	67.959417	61.443940	0.00	29.00	48.00	84.00	1005.00
14	20454.0	49.941772	24.753120	0.00	31.00	52.00	69.00	336.00
20	8766.0	2.364944	4.236706	0.00	0.80	1.60	2.80	195.00
30	8766.0	0.531371	0.538180	0.00	0.20	0.40	0.70	15.10
35	8766.0	0.479751	1.183618	0.00	0.10	0.20	0.50	35.70
42	4383.0	1.400897	0.251836	-0.01	1.25	1.38	1.54	3.09
43	4383.0	1.292923	0.230898	-0.14	1.17	1.28	1.43	2.77
44	4383.0	0.108941	0.068776	0.00	0.06	0.10	0.14	1.31

8. Mostrar un resumen descriptivo para cada contaminente por distritos.

```
#Solución
nuevo.groupby(['ESTACION','MAGNITUD']).VALOR.describe() #Dado que las otras columnas son solc
```

		count	mean	std	min	25%	50%	75%	max
ESTACION	MAGNITUD								
4	1	1461.0	7.329911	16.379050	1.0	4.0	7.0	9.0	610.0
	6	1461.0	0.411499	0.172902	0.1	0.3	0.4	0.5	1.3
	7	1461.0	31.939767	37.667968	0.0	8.0	16.0	42.0	239.0
	8	1461.0	44.398357	17.766063	0.0	31.0	43.0	55.0	105.0
	12	1461.0	93.341547	72.436531	0.0	44.0	69.0	119.0	467.0
...	...	...	...	...	...	...	...	...	...
60	7	1461.0	12.326489	19.593109	1.0	2.0	4.0	12.0	151.0
	8	1461.0	31.125941	18.101896	3.0	18.0	27.0	41.0	101.0

9. Crear una función que reciba una estación y un contaminante y devuelva un resumen descriptivo de las emisiones del contaminante indicado en la estadión indicada.

```
#Solución
'''
Se crea una función con lambda, donde recibe los siguientes parámetros, según el orden:
1) estacion
2) contaminante
3) df (dataframe)
Retorna un resumen descriptivo de las emisiones del contaminante
'''
resumen_estacion_contaminante = lambda estacion, contaminante, df: df[(df["ESTACION"] == esta
try:
  es = abs(int(input("Ingrese el código de estación, vease a punto 6: ")))
  con = abs(int(input("Ingrese el código de contaminación, vease a punto 6: ")))
  print("\nResumen descriptivo\n", resumen_estacion_contaminante(es, con, nuevo))
except ValueError:
  print("Error en el ingreso de las notas, debe ser entero, vuelve a intentar.")
```

```
Ingrese el código de estación, vease a punto 6: 5
Ingrese el código de contaminación, vease a punto 6: 5

Resumen descriptivo
 count    0.0
mean     NaN
std      NaN
min      NaN
25%      NaN
50%      NaN
75%      NaN
max      NaN
Name: VALOR, dtype: float64
```

10. Crear una función que reciba una estación de medición y una magnitud y devuelva una lista con todas las mediciones de la magnitud en la estación.

```
#Solución
'''
Se crea una función con lambda, donde recibe los siguientes parámetros, según el orden:
1) e (estacion)
2) c (contaminante)
3) df (dataframe)
Retorna una lista de la columna "VALOR", mediciones de la magnitud de
contaminación, según corresponda la estación y contaminante
'''

lista_mediciones = lambda e, c, df: list(df[(df['ESTACION'] == e) & (df['MAGNITUD'] == c)].VA

#Dado que los códigos
try:
  es = abs(int(input("Ingrese el código de estación, vease a punto 6: ")))
  con = abs(int(input("Ingrese el código de contaminación, vease a punto 6: ")))
  print(lista_mediciones(es, con, nuevo))
except ValueError:
  print("Error en el ingreso de las notas, debe ser entero, vuelve a intentar.")
```

```
Ingrese el código de estación, vease a punto 6: 14
Ingrese el código de contaminación, vease a punto 6: 4
[]
```

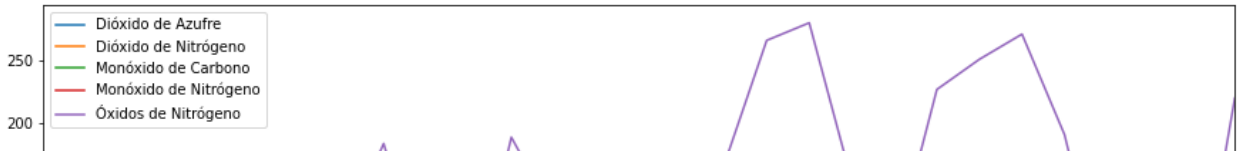
```
#Verificación del ejercicio
nuevo[(nuevo['ESTACION'] == 50) & (nuevo['MAGNITUD'] == 12)]
```

	ESTACION	MAGNITUD	ANO	MES	DIAS	VALOR	FECHA
163449	50	12	2016	1	01	80.0	2016-01-01

11. Crear una función que reciba un rango de fechas y una estación de medición y genere un gráfico con la evolución diaria de las magnitudes de esa estación en las fechas indicadas.

```
#Solución
'''
Se crea una función "rango_f_magnitud" donde recibe los siguientes parámetros, según el order
1) data (dataframe). IMPORTANTE, la columna "FECHA" debe ser de tipo datetime
2) fecha_inicio (Fecha de comienzo del rango)
3) fecha_termino (Fecha de termino del rango)
4) estacion
Esta función y genere un gráfico con la evolución diaria de las magnitudes de esa estación er
'''

def rango_f_magnitud(data, fecha_inicio, fecha_termino, estacion):
  data = data[(data['ESTACION'] == estacion) & (data['FECHA'] >= fecha_inicio) & (data['FECHA
  data['NOMBRE CONTAMINACION'] = data['MAGNITUD'].apply(lambda x: diccionario_contaminante[st
  data.set_index('FECHA', inplace = True) #Se reemplaza el index por la fecha
  data.groupby('NOMBRE CONTAMINACION').VALOR.plot(figsize=(15, 5) , legend = True)
  plt.savefig("Grafico_evolucion_diaria_estacion_"+ str(estacion) +".jpg") #Permite guardar e
  plt.show()

try:
  inicio = input("Ingrese la fecha de inicio, en formato 'Año-Mes-Dia' (Ej: 2018-01-01): ")
  final = input("Ingrese la fecha de inicio, en formato Año-Mes-Dia (Ej: 2018-03-31): ")
  est = abs(int(input("Ingrese el código de estación, vease a punto 6: "))) #Codigo estacion
  rango_f_magnitud(nuevo, inicio, final, est)

except: #Dado que puede desplegar distintos tipos de errores, no se ingresa tipo de error
  print("Error con el ingreso de datos, vuelve a intentarlo.")
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:13: SettingWithCopyWa
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
  del sys.path[0]
```



12. Crear una función que reciba un mes y una magnitud y devuelva un diccionario con las medias de la magnitud dentro de Madrid Central y fuera de ella.

```
#Solución
'''
Se crea una función "dict_medida", donde recibe los siguientes parámetros, según el orden:
1) df (dataframe).
2) magnitud (contaminaciójn)
3) mes
Esta función retorna un diccionario con las medias en las mediciones de contaminación,
según la magnitud y el mes correspondiente.
'''

def dict_medidas(df, magnitud, mes):
  dicc = df[(df.MAGNITUD == magnitud) & (df.MES == mes)].groupby(['ESTACION', 'MES']).mean()
  dicc2 = dicc.drop('ANO', axis=1)

  for key, value in dicc2.items():
    dicc2[key] = round(value, 2)


  dicc3 = dicc2.to_dict('dict')
  return dicc3

try:
  con= abs(int(input("Ingrese el código de estación, vease a punto 6: "))) #Codigo estacion e
  month = abs(int(input("Ingrese el mes donde deseas extraer medidas de la magnitud: ")))
  x = dict_medidas(nuevo, con, month)
  print("\nFormato del diccionario\n (ESTACION, MES): MAGNITUD/VALOR\n")
  for i in x.keys():
    print(i," : ",x[i])

except ValueError: #Dado que puede desplegar distintos tipos de errores, no se ingresa tipo c
  print("Error con el ingreso de datos, vuelve a intentarlo.")
```

```
Ingrese el código de estación, vease a punto 6: 6
Ingrese el mes donde deseas extraer medidas de la magnitud: 2

Formato del diccionario
```

```
 (ESTACION, MES): MAGNITUD/VALOR

MAGNITUD  :  {(4, 2): 6.0, (8, 2): 6.0, (16, 2): 6.0, (18, 2): 6.0, (24, 2): 6.0, (35,
VALOR  :  {(4, 2): 0.47, (8, 2): 0.4, (16, 2): 0.39, (18, 2): 0.48, (24, 2): 0.26, (35,
```

✓ 1 s completado a las 18:55