😎 Estadística con Pandas

Para el ejemplo se utilizarán los datos que provee la AEMET (Agencia estatal de Meteorología de España) (https://datosclima.es/Aemet2013/DescargaDatos.html)

- Los datos contienen información diaria sobre temperatura (máxima, mínima y media), viento (racha y velocidad máxima de las medias) y precipitaciones (total y por tramo horario) para cada estación meteorológica de España. *Racha: aumento brusco del viento con respecto a su velocidad media tomada en un cierto intervalo de tiempo.*
- En algunos casos falta información y eso se tiene que tener en cuenta durante el proceso de limpieza de datos.
- Se realiza el análisis de datos del mes de Enero del año 2021
- Se utiliza la librería **glob** para lrealizar la carga de varios archivos de forma más sencilla.

In [1]:
```
import pandas as pd
import glob
import numpy as np
```

In [2]:
```
all_files = glob.glob("archs/Aemet2021-01/Aemet2021-*.xls")
# all_files
```

In [3]:
```
file_list = []
for f in all_files:
    data = pd.read_excel(f,skiprows=4)
    data['fuente'] = f # nueva columna conteniendo el nombre del archivo leido
    file_list.append(data)
df = pd.concat(file_list)
df.head()
```

Out[3]:

	Estación	Provincia	Temperatura máxima (ºC)	Temperatura mínima (ºC)	Temperatura media (ºC)	Racha (km/h)	Velocidad máxima (km/h)	Precipitación 00-24h (mm)	Precipitación 00-06h (mm)	Precipitación 06-12h (mm)	Precipitación 12-18h (mm)	Precipitación 18-24h (mm)	
0	Estaca de Bares	A Coruña	10.0 (08:20)	6.4 (12:40)	8.2	68 (12:40)	53 (02:00)	5.8	1.0	1.6	3.2	0.0	a
1	Ferrol	A Coruña	9.6 (12:30)	4.2 (09:50)	6.9	44 (14:00)	19 (17:00)	14.7	1.9	8.6	1.5	2.7	a
2	As Pontes	A Coruña	6.9 (13:40)	1.3 (08:40)	4.1	NaN	NaN	12.8	3.6	4.4	2.8	2.0	a
3	A Coruña	A Coruña	9.8 (12:20)	6.0 (08:00)	7.9	61 (14:20)	33 (14:20)	12.0	6.8	0.6	1.6	3.0	a
4	A Coruña Aeropuerto	A Coruña	9.1 (12:30)	3.3 (09:30)	6.2	54 (13:00)	30 (13:00)	14.9	5.8	4.6	2.3	2.2	a

In [4]:
```
df.shape
```

Out[4]: (24662, 13)

In [5]:
```
df.tail()
```

Out[5]:

	Estación	Provincia	Temperatura máxima (ºC)	Temperatura mínima (ºC)	Temperatura media (ºC)	Racha (km/h)	Velocidad máxima (km/h)	Precipitación 00-24h (mm)	Precipitación 00-06h (mm)	Precipitación 06-12h (mm)	Precipitación 12-18h (mm)	Precipitación 18-24h (mm)
790	San Juan de la Rambla	Santa Cruz de Tenerife	18.3 (23:59)	10.1 (00:00)	14.2	27 (04:00)	16 (23:59)	0.0	0.0	0.0	0.0	0.0
791	Cerler-Cogulla	Huesca	3.1 (15:50)	-6.2 (00:00)	-1.5	82 (09:30)	66 (09:30)	0.2	0.0	0.2	0.0	0.0
792	Panticosa-Petrosos	Huesca	4.9 (17:10)	-2.2 (00:00)	1.4	68 (22:50)	42 (22:50)	5.2	1.4	1.6	1.0	1.2
793	Valverde	Santa Cruz de Tenerife	14.7 (15:10)	12.0 (00:30)	13.3	36 (00:30)	26 (00:30)	0.2	0.0	0.2	0.0	0.0
794	Sóller, Puerto	Illes Balears	16.4 (15:20)	13.8 (05:30)	15.1	36 (08:10)	22 (00:40)	0.0	0.0	0.0	0.0	0.0

Limpieza de datos

Luego de obtener los datos, una parte importante es limpiar su contenido para poder trabajarlos. En principio se puede renombrar las columnas con nombres largos.

In [6]:
```
df=df.rename(columns={'Estación': 'estacion','Provincia':'provincia','Temperatura máxima (ºC)':'temp_max',
                      'Temperatura mínima (ºC)':'temp_min','Temperatura media (ºC)':'temp_med',
                      'Racha (km/h)':'viento_racha','Velocidad máxima (km/h)':'viento_vel_max',
                      'Precipitación 00-24h (mm)':'prec_dia','Precipitación 00-06h (mm)':'prec_0_6h',
                      'Precipitación 06-12h (mm)':'prec_6_12h','Precipitación 12-18h (mm)':'prec_12_18h',
                      'Precipitación 18-24h (mm)':'prec_18_24h','fuente':'fecha'})
df.head()
```

Out[6]:

	estacion	provincia	temp_max	temp_min	temp_med	viento_racha	viento_vel_max	prec_dia	prec_0_6h	prec_6_12h	prec_12_18h	prec_18_24h	
0	Estaca de Bares	A Coruña	10.0 (08:20)	6.4 (12:40)	8.2	68 (12:40)	53 (02:00)	5.8	1.0	1.6	3.2	0.0	archs/ 01\
1	Ferrol	A Coruña	9.6 (12:30)	4.2 (09:50)	6.9	44 (14:00)	19 (17:00)	14.7	1.9	8.6	1.5	2.7	archs/ 01\
2	As Pontes	A Coruña	6.9 (13:40)	1.3 (08:40)	4.1	NaN	NaN	12.8	3.6	4.4	2.8	2.0	archs/ 01\
3	A Coruña	A Coruña	9.8 (12:20)	6.0 (08:00)	7.9	61 (14:20)	33 (14:20)	12.0	6.8	0.6	1.6	3.0	archs/ 01\
4	A Coruña Aeropuerto	A Coruña	9.1 (12:30)	3.3 (09:30)	6.2	54 (13:00)	30 (13:00)	14.9	5.8	4.6	2.3	2.2	archs/ 01\

En algunas columnas, además del valor numérico también se tiene la hora entre paréntesis. No nos interesa ese dato ya que la información tiene que ser numérica para el análisis, en consecuencia se puede quitar:

In [7]:
```
df=df.replace(to_replace=r'.\(.+\)$', value='', regex=True)
df.head()
```

Out[7]:

	estacion	provincia	temp_max	temp_min	temp_med	viento_racha	viento_vel_max	prec_dia	prec_0_6h	prec_6_12h	prec_12_18h	prec_18_24h	
0	Estaca de Bares	A Coruña	10.0	6.4	8.2	68	53	5.8	1.0	1.6	3.2	0.0	archs/ 01\
1	Ferrol	A Coruña	9.6	4.2	6.9	44	19	14.7	1.9	8.6	1.5	2.7	archs/ 01\
2	As Pontes	A Coruña	6.9	1.3	4.1	NaN	NaN	12.8	3.6	4.4	2.8	2.0	archs/ 01\
3	A Coruña	A Coruña	9.8	6.0	7.9	61	33	12.0	6.8	0.6	1.6	3.0	archs/ 01\
4	A Coruña Aeropuerto	A Coruña	9.1	3.3	6.2	54	30	14.9	5.8	4.6	2.3	2.2	archs/ 01\

La nueva columna 'fecha' contiene información del nombre del archivo además de la fecha, procederemos a obtener sólo la fecha:

In [8]:
```
df.fecha=df.fecha.replace(regex={'archs/Aemet2021-01\\\\Aemet':'',r'\.+xls':''})
df.head()
```

Out[8]:

	estacion	provincia	temp_max	temp_min	temp_med	viento_racha	viento_vel_max	prec_dia	prec_0_6h	prec_6_12h	prec_12_18h	prec_18_24h	fecha
0	Estaca de Bares	A Coruña	10.0	6.4	8.2	68	53	5.8	1.0	1.6	3.2	0.0	2021-01-01
1	Ferrol	A Coruña	9.6	4.2	6.9	44	19	14.7	1.9	8.6	1.5	2.7	2021-01-01
2	As Pontes	A Coruña	6.9	1.3	4.1	NaN	NaN	12.8	3.6	4.4	2.8	2.0	2021-01-01
3	A Coruña	A Coruña	9.8	6.0	7.9	61	33	12.0	6.8	0.6	1.6	3.0	2021-01-01
4	A Coruña Aeropuerto	A Coruña	9.1	3.3	6.2	54	30	14.9	5.8	4.6	2.3	2.2	2021-01-01

Revisamos los tipos de datos:

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24662 entries, 0 to 794
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   estacion        24662 non-null  object 
 1   provincia       24662 non-null  object 
 2   temp_max        23636 non-null  object 
 3   temp_min        23636 non-null  object 
 4   temp_med        23636 non-null  float64
 5   viento_racha    19973 non-null  object 
 6   viento_vel_max  20154 non-null  object 
 7   prec_dia        23161 non-null  float64
 8   prec_0_6h       23440 non-null  float64
 9   prec_6_12h      23376 non-null  float64
 10  prec_12_18h     23395 non-null  float64
 11  prec_18_24h     23446 non-null  float64
 12  fecha           24662 non-null  object 
dtypes: float64(6), object(7)
memory usage: 2.6+ MB
```

Puede observarse que hay columnas con tipo "object" cuando debería ser un float (por ejemplo: temp_max). Convertiremos a los tipos adecuados cada columna:

In [10]:
```
df=df.astype({'estacion': 'string','provincia': 'string', 'temp_max':'float64', 'temp_min':'float64',
              'viento_racha':'float64', 'viento_vel_max':'float64','fecha': 'datetime64[ns]'})
df['fecha']=pd.to_datetime(df["fecha"].dt.strftime('%Y-%m-%d'))
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24662 entries, 0 to 794
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   estacion        24662 non-null  string        
 1   provincia       24662 non-null  string        
 2   temp_max        23636 non-null  float64       
 3   temp_min        23636 non-null  float64       
 4   temp_med        23636 non-null  float64       
 5   viento_racha    19973 non-null  float64       
 6   viento_vel_max  20154 non-null  float64       
 7   prec_dia        23161 non-null  float64       
 8   prec_0_6h       23440 non-null  float64       
 9   prec_6_12h      23376 non-null  float64       
 10  prec_12_18h     23395 non-null  float64       
 11  prec_18_24h     23446 non-null  float64       
 12  fecha           24662 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(10), string(2)
memory usage: 2.6 MB
```

Para este ejemplo eliminaremos las estaciones que tienen filas a las que les faltan datos aunque no se recomienda. Lo ideal es utilizar técnicas de rellenado (media, moda, propagate last valid observation forward, etc):

In [11]:
```
estaciones_con_nan=df[df.isnull().any(axis=1)]['estacion'].unique()
df.drop(df[df['estacion'].isin(estaciones_con_nan)].index, inplace=True)
df.isna().sum()
```

Out[11]:
```
estacion          0
provincia         0
temp_max          0
temp_min          0
temp_med          0
viento_racha      0
viento_vel_max    0
prec_dia          0
prec_0_6h         0
prec_6_12h        0
prec_12_18h       0
prec_18_24h       0
fecha             0
dtype: int64
```

Extracción de información

Con el método "describe"se obtiene información estadística. Incluso se puede aplicar a una porción de los datos:

In [12]:
```
df.iloc[:,np.r_[0:8,12:13]].describe()
```

Out[12]:

	temp_max	temp_min	temp_med	viento_racha	viento_vel_max	prec_dia
count	5394.000000	5394.000000	5394.000000	5394.000000	5394.000000	5394.000000
mean	11.497608	3.214127	7.357045	35.581943	20.772525	2.530460
std	6.295150	6.974918	6.312166	19.300200	11.898887	7.038373
min	-9.700000	-23.000000	-12.800000	0.000000	0.000000	0.000000
25%	6.900000	-2.100000	2.500000	21.000000	12.000000	0.000000
50%	11.800000	2.900000	7.300000	32.000000	18.000000	0.000000
75%	16.000000	8.600000	12.000000	48.000000	27.000000	1.200000
max	29.200000	20.200000	23.400000	179.000000	105.000000	116.800000

Emitiremos 5 filas al azar como muestreo de los datos. Es un ejemplo de selección de columnas por expresion regular (seleccionamos las que empiezan por el prefijo "temp_"):

In [13]:
```
df.filter(regex='^temp_*', axis=1).sample(5)
```

Out[13]:

	temp_max	temp_min	temp_med
99	5.0	1.3	3.1
421	22.7	17.5	20.1
96	12.0	0.3	6.2
407	17.6	8.3	13.0
38	16.2	2.9	9.6

Medias por provincia

Emitiremos la media de los valores de cada columna agrupándolos por provincia:

In [14]:
```
df.groupby('provincia').mean().head()
```

Out[14]:

provincia	temp_max	temp_min	temp_med	viento_racha	viento_vel_max	prec_dia	prec_0_6h	prec_6_12h	prec_12_18h	prec_18_24h
A Coruña	11.525000	4.818750	8.184375	31.500000	13.656250	6.925000	1.971875	1.825000	1.131250	1.996875
Alacant/Alicante	16.027174	5.877174	10.955435	35.402174	19.043478	2.325000	0.778261	0.536957	0.468478	0.541304
Araba/Álava	8.109677	1.403226	4.754839	34.064516	17.225806	2.096774	0.651613	0.658065	0.400000	0.387097
Asturias	10.824324	2.981081	6.908108	33.243243	19.189189	4.843243	1.389189	1.372973	1.459459	0.621622
Badajoz	11.984804	2.608824	7.301471	30.598039	19.308824	1.616667	0.303922	0.567647	0.409804	0.335294

Mayores temperaturas máximas por estación

Emitiremos los 10 valores más altos de temp_max que se han registrado en el mes, sólo con las columnas que nos interesan:

In [15]:
```
df.sort_values(by='temp_max',ascending=False)[['fecha','estacion','provincia','temp_max']].head()
```

Out[15]:

	fecha	estacion	provincia	temp_max
38	2021-01-28	Alicante/Alacant	Alacant/Alicante	29.2
419	2021-01-28	Antigua	Las Palmas	28.4
37	2021-01-28	Alicante-Elche Aeropuerto	Alacant/Alicante	28.3
694	2021-01-28	Estación de Tortosa	Tarragona	28.1
786	2021-01-29	Totana	Murcia	28.0

Menores temperaturas mínimas por estación

De forma similar obtenemos los 10 valores más bajos de temp_min:

In [16]:
```
df.sort_values(by='temp_min',ascending=True)[['fecha','estacion','provincia','temp_min']].head()
```

Out[16]:

	fecha	estacion	provincia	temp_min
702	2021-01-12	Santa Eulalia del Campo	Teruel	-23.0
700	2021-01-14	Santa Eulalia del Campo	Teruel	-19.5
702	2021-01-11	Santa Eulalia del Campo	Teruel	-18.8
770	2021-01-12	Daroca	Zaragoza	-18.5
702	2021-01-13	Santa Eulalia del Campo	Teruel	-17.8

Máximas diferencias entre temperatura max y min por provincia

Calculamos una nueva columna restando de la temperatura máxima la temperatura mínima:

In [17]:
```
df['diff_temp_min_max']=df['temp_max']-df['temp_min']
df.iloc[df.reset_index().groupby('provincia')['diff_temp_min_max'].idxmax()].sort_values(by='diff_temp_min_max',
        ascending=False)[['fecha','estacion','provincia','diff_temp_min_max','temp_min','temp_max']].head()
```

Out[17]:

	fecha	estacion	provincia	diff_temp_min_max	temp_min	temp_max
167	2021-01-18	Morón de Almazán	Soria	24.4	-14.1	10.3
459	2021-01-17	Astorga	León	24.1	-7.7	16.4
234	2021-01-17	Puebla de Don Rodrigo	Ciudad Real	23.7	-4.8	18.9
165	2021-01-17	Nuñomoral	Cáceres	23.3	-5.1	18.2
626	2021-01-19	Cuéllar	Segovia	22.8	-8.4	14.4

Mayores rachas de viento en un día por provincia

Agrupando por columna (provincia), obtenemos el máximo de los valores de otra columna (viento_racha) para cada grupo y luego ordenamos:

In [18]:
```
df.iloc[df.reset_index().groupby('provincia')['viento_racha'].idxmax()].sort_values(by='viento_racha',
                ascending=False)[['fecha','estacion','provincia','viento_racha']].head()
```

Out[18]:

	fecha	estacion	provincia	viento_racha
792	2021-01-22	Panticosa-Petrosos	Huesca	179.0
363	2021-01-22	Sierra de Alfabia, Bunyola	Illes Balears	144.0
61	2021-01-20	Cabrales	Asturias	136.0
224	2021-01-22	Morella	Castelló/Castellón	124.0
747	2021-01-22	Fuente el Sol	Valladolid	123.0

Mayores velocidades de viento medio en un día por provincia

Similarmente hacemos el cálculo de las mayores velocidades de viento medio en un día por provincia:

In [19]:
```
df.iloc[df.reset_index().groupby('provincia')['viento_vel_max'].idxmax()].sort_values(by='viento_vel_max',
                ascending=False)[['fecha','estacion','provincia','viento_vel_max']].head()
```

Out[19]:

	fecha	estacion	provincia	viento_vel_max
792	2021-01-22	Panticosa-Petrosos	Huesca	105.0
597	2021-01-22	Autilla del Pino	Palencia	80.0
771	2021-01-22	Zaragoza Aeropuerto	Zaragoza	72.0
747	2021-01-22	Fuente el Sol	Valladolid	71.0
363	2021-01-31	Sierra de Alfabia, Bunyola	Illes Balears	70.0

Máximas velocidades de viento por provincia de media

Calculamos una media por grupo:

In [20]:
```
df.groupby('provincia').mean()[['viento_vel_max']].sort_values(by='viento_vel_max',ascending=False).head()
```

Out[20]:

provincia	viento_vel_max
Castelló/Castellón	33.116667
Teruel	28.537634
Zaragoza	25.307087
Soria	24.590909
Valladolid	24.433333

Máximas diferencias de temperatura máxima entre días

Ejemplo de cómo podemos hacer una operación (restar) entre la fila actual y la fila anterior de forma sencilla: con un "shift" desplazamos una columna hacia abajo.

In [21]:
```
df['temp_max_shifted'] = df.groupby('estacion')['temp_max'].shift(1)
df['dif_prev_temp_max'] = df.temp_max - df.temp_max_shifted
df['dif_prev_temp_max']
```

Out[21]:
```
16     NaN
37     NaN
38     NaN
39     NaN
56     NaN
      ...
786    0.0
787    0.3
791   -0.2
792    0.6
793   -4.9
Name: dif_prev_temp_max, Length: 5394, dtype: float64
```

Días seguidos lloviendo por provincia y estación

El proceso de obtener número de días en los que pasa un suceso (en este caso llover)es un proceso algo más elaborado:

1. Creación de una nueva columna cuyo valor se obtiene a partir de otra columna aplicando una función lambda (cálculo de la columna "llueve").
2. Comparación de valores (operador «ne») entre columnas de una misma fila.
3. Desplazamiento (método "shift") de una columna para disponer de datos anteriores en la misma fila.
4. Generación de un número secuencial (identificador) de los elementos de cada grupo (método «cumsum»).
5. Cálculo del número de elementos de un grupo (método "cumcount").
6. Asignación de un valor a una columna para las filas que cumplan una condición (llueve==0).

In [22]:
```
df['llueve']=df.prec_dia.apply(lambda x: 1 if x>0 else 0)

df.sort_values(by=['estacion','fecha'],ascending=True,inplace=True,ignore_index=True)

df['start_of_streak'] = df.llueve.ne(df['llueve'].shift())

df['estacion_anterior']=df['estacion'].shift()
df.loc[(df['estacion'].ne(df['estacion_anterior'])), 'start_of_streak'] = True

df['streak_id'] = df['start_of_streak'].cumsum()
df['dias_seguidos_lloviendo'] = df.groupby('streak_id').cumcount() + 1
df.loc[(df['llueve']==0), 'dias_seguidos_lloviendo'] = 0

df.iloc[df.reset_index().groupby(['provincia','estacion'])['dias_seguidos_lloviendo'].idxmax()].sort_values(by='dias_seguidos_ll
    ascending=False)[['estacion','provincia','dias_seguidos_lloviendo']].head(10)
```

Out[22]:

	estacion	provincia	dias_seguidos_lloviendo
3819	Pontevedra	Pontevedra	13
5259	Villanueva de los Infantes	Ciudad Real	12
5041	Valverde del Fresno	Cáceres	12
1405	Cazalla de la Sierra	Sevilla	12
726	Arure	Santa Cruz de Tenerife	11
1893	Escorca, Lluc	Illes Balears	11
4069	San Andrés, Valverde	Santa Cruz de Tenerife	11
1594	Coruña del Conde	Burgos	10
3925	Puntagorda	Santa Cruz de Tenerife	10
2549	Irún	Gipuzkoa	10

Dónde llueve menos días seguidos por provincia y estación

Aprovechando la nueva columna "dias_seguidos_lloviendo" podemos sacar esta nueva métrica.

In [23]:
```
df.iloc[df.reset_index().groupby(['provincia','estacion'])['dias_seguidos_lloviendo'].idxmax()].sort_values(by='dias_seguidos_ll
                            ascending=False)[['estacion','provincia','dias_seguidos_lloviendo']].head()
```

Out[23]:

	estacion	provincia	dias_seguidos_lloviendo
3819	Pontevedra	Pontevedra	13
5259	Villanueva de los Infantes	Ciudad Real	12
5041	Valverde del Fresno	Cáceres	12
1405	Cazalla de la Sierra	Sevilla	12
726	Arure	Santa Cruz de Tenerife	11

Mostrar cuántas estaciones hay por provincia y el total general

Podemos tener copias "temporales" del dataframe para hacer operaciones sin destruir el original. Supongamos que queremos obtener el número de estaciones meteorológicas que hay por cada provincia.

Para ello nos quedamos con solo las columnas relevantes: provincia y estación. Luego eliminamos las filas duplicadas para tener los pares únicos de provincia-estación.

Ya nos queda simplemente agrupar por provincia y sacar el número de filas por cada grupo. Finalmente agregamos otra fila adicional al resultado con el total de estaciones:

In [24]:
```
df_estaciones=df[['provincia','estacion']].drop_duplicates().groupby('provincia').count()
df_estaciones.loc['Total de estaciones'] = df_estaciones['estacion'].sum()
df_estaciones
```

Out[24]:

provincia	estacion
A Coruña	3
Alacant/Alicante	5
Araba/Álava	2
Asturias	7
Badajoz	14
Barcelona	4
Bizkaia	2
Burgos	7
Cantabria	4
Castelló/Castellón	3
Ceuta	1
Ciudad Real	7
Cuenca	2
Cáceres	19
Cádiz	9
Córdoba	7
Gipuzkoa	3
Girona	7
Granada	3
Guadalajara	2
Huelva	3
Huesca	22
Illes Balears	11
Jaén	4
La Rioja	5
Las Palmas	26
León	7
Lleida	4
Lugo	2
Madrid	4
Melilla	1
Murcia	6
Málaga	9
Navarra	13
Palencia	7
Pontevedra	6
Salamanca	8
Santa Cruz de Tenerife	21
Segovia	5
Sevilla	9
Soria	6
Tarragona	2
Teruel	6
Toledo	1
Valladolid	3
València/Valencia	6
Zamora	8
Zaragoza	10
Ávila	4
Total de estaciones	330