Librerías en Python para ciencia de datos.

Ejercicio U10\_E01

MasterD

# Manejo básico de datos con Pandas

En este ejercicio vamos a comprobar qué tal te desenvuelves realizando algunas operaciones básicas de manejo de datos con la librería Pandas. Seguiremos los mismos pasos que ya tuviste que hacer con R en la unidad 2. De paso, te servirá para contrastar las diferencias que hay entre un lenguaje y otro.

Vamos a empezar leyendo datos de uno de los ficheros incluidos en el directorio de ejercicios de esta unidad. Antes que nada, conviene que cambiemos el directorio de trabajo de nuestra sesión en Python. Lo haremos con la función os.chdir().

Ajusta la ruta del directorio a la que estés utilizando en tu ordenador:

**import** **os  
import** **numpy** **as** **np  
import** **pandas** **as** **pd**  
os.chdir("RUTA\_DIRECTORIO\_CUSO/10PythonLibreriasCienciaDatos/Ejercicios")

Ahora ya puedes leer el fichero consumo\_horario\_cliente.csv Recuerda que Pandas ya incluye funciones para leer distintos tipos de fichero, y presta atención a los caracteres delimitadores y a los encabezados.

consumo = pd.read\_csv("./consumo\_horario\_cliente.csv", sep = ";", header=0)

Comprobemos que los datos se han cargado bien

consumo.head()

|  | **datetime** | **consumo\_kwh** | **precio\_kwh** |
| --- | --- | --- | --- |
| **0** | 2016-09-01 00:00:00 | 0.188 | 0.097510 |
| **1** | 2016-09-01 01:00:00 | 0.124 | 0.097756 |
| **2** | 2016-09-01 02:00:00 | 0.111 | 0.097713 |
| **3** | 2016-09-01 03:00:00 | 0.129 | 0.099275 |
| **4** | 2016-09-01 04:00:00 | 0.108 | 0.100793 |

La columna datetime ahora mismo es de tipo texto. Pero podemos decirle a Pandas que la convierta a un tipo de fecha y hora más acorde, que justamente se llama tipo datetime. Lo podemos hacer así:

consumo['datetime'] = pd.to\_datetime(consumo['datetime'])

La ventaja de usar el tipo datetime es que nos proporciona muchas funciones para extraer campos de la fecha y hora a través de la propiedad dt. Veamos unos ejemplos:

*# Tomamos unas pocas filas para el ejemplo*

consumo\_h5 = consumo.head()

*# Extraer la fecha*

consumo\_h5['datetime'].dt.date

0 2016-09-01

1 2016-09-01

2 2016-09-01

3 2016-09-01

4 2016-09-01

Name: datetime, dtype: object

*# Extraer la hora*

consumo\_h5['datetime'].dt.hour

0 0

1 1

2 2

3 3

4 4

Name: datetime, dtype: int64

De forma análoga, hay atributos para extraer el año (year), mes (month), día del mes (day) o el día de la semana (weekday\_name)

Otro método importante es assign(), que nos permite añadir nuevas columnas a un DataFrame a partir de columnas existentes:

consumo = consumo.assign(

fecha = consumo['datetime'].dt.date

)

consumo.head()

|  | **datetime** | **consumo\_kwh** | **precio\_kwh** | **fecha** |
| --- | --- | --- | --- | --- |
| **0** | 2016-09-01 00:00:00 | 0.188 | 0.097510 | 2016-09-01 |
| **1** | 2016-09-01 01:00:00 | 0.124 | 0.097756 | 2016-09-01 |
| **2** | 2016-09-01 02:00:00 | 0.111 | 0.097713 | 2016-09-01 |
| **3** | 2016-09-01 03:00:00 | 0.129 | 0.099275 | 2016-09-01 |
| **4** | 2016-09-01 04:00:00 | 0.108 | 0.100793 | 2016-09-01 |

Fijándote en los ejemplos anteriores, ahora te vamos a pedir que escribas un código en el que asignes a nuestro DataFrame de consumos unas columnas con los siguientes valores:

* coste\_eur : con el coste en euros de la energía consumida, multiplicando el consumo por el precio
* fecha : extrayendo la fecha de la columna datetime
* hora : extrayendo la hora de la columna datetime
* dia\_semana : extrayendo el día de la semana de la columna datetime

consumo = consumo.assign(

coste\_eur = consumo['consumo\_kwh'] \* consumo['precio\_kwh'],

fecha = consumo['datetime'].dt.date,

hora = consumo['datetime'].dt.hour,

mes = consumo['datetime'].dt.month,

dia\_semana = consumo['datetime'].dt.weekday\_name

)

Al final deberías obtener algo así:

consumo.head()

|  | datetime | consumo\_kwh | precio\_kwh | coste\_eur | dia\_semana | fecha | hora | mes |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2016-09-01 00:00:00 | 0.188 | 0.097510 | 0.018332 | Thursday | 2016-09-01 | 0 | 9 |
| 1 | 2016-09-01 01:00:00 | 0.124 | 0.097756 | 0.012122 | Thursday | 2016-09-01 | 1 | 9 |
| 2 | 2016-09-01 02:00:00 | 0.111 | 0.097713 | 0.010846 | Thursday | 2016-09-01 | 2 | 9 |
| 3 | 2016-09-01 03:00:00 | 0.129 | 0.099275 | 0.012806 | Thursday | 2016-09-01 | 3 | 9 |
| 4 | 2016-09-01 04:00:00 | 0.108 | 0.100793 | 0.010886 | Thursday | 2016-09-01 | 4 | 9 |

¿Cuál es el consumo máximo que se ha registrado en los datos que tenemos?

max(consumo['consumo\_kwh'])

consumo['consumo\_kwh'].max()

3.608

Extrae los datos del día y hora en el que se ha registrado el mayor consumo.

consumo.loc[consumo['consumo\_kwh'] == max(consumo['consumo\_kwh'])]

*# Alternativamente:*

consumo.iloc[consumo['consumo\_kwh'].idxmax()]

|  | datetime | consumo\_kwh | precio\_kwh | coste\_eur | dia\_semana | fecha | hora | mes |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 86 | 2016-09-04 14:00:00 | 3.608 | 0.104123 | 0.375675 | 6 | 2016-09-04 | 14 | 9 |

Filtra las horas con consumo inferior a 0.02 kwh

consumo.loc[consumo.consumo\_kwh < 0.02]

|  | datetime | consumo\_kwh | precio\_kwh | coste\_eur | dia\_semana | fecha | hora | mes |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 589 | 2016-12-27 13:00:00 | 0.010 | 0.131662 | 0.001317 | Tuesday | 2016-12-27 | 13 | 12 |
| 590 | 2016-12-27 14:00:00 | 0.011 | 0.130844 | 0.001439 | Tuesday | 2016-12-27 | 14 | 12 |
| 591 | 2016-12-27 15:00:00 | 0.010 | 0.130225 | 0.001302 | Tuesday | 2016-12-27 | 15 | 12 |
| 592 | 2016-12-27 16:00:00 | 0.010 | 0.129591 | 0.001296 | Tuesday | 2016-12-27 | 16 | 12 |
| 593 | 2016-12-27 17:00:00 | 0.011 | 0.135941 | 0.001495 | Tuesday | 2016-12-27 | 17 | 12 |

¿Puedes calcular el consumo promedio de los martes?

consumo['consumo\_kwh'].loc[consumo['dia\_semana'] == 'Tuesday'].mean()

0.2827034313725488

¿Y el precio promedio del kwh?

consumo['precio\_kwh'].loc[consumo['dia\_semana'] == 'Tuesday'].mean()

0.12017461443627442

¿Serías capaz de calcular el consumo y precio promedio para los martes y los viernes usando una sola expresión? Pista: puedes utilizar el método isin() de una serie o columna para filtrar los días de la semana que caigan en martes o viernes, y utilizar groupby() para agregar después y calcular las medias.

consumo.loc[consumo['dia\_semana'].isin(['Tuesday','Friday'])].groupby('dia\_semana')['consumo\_kwh','precio\_kwh'].mean()

|  | consumo\_kwh | precio\_kwh |
| --- | --- | --- |
| dia\_semana |  |  |
| Friday | 0.285625 | 0.120325 |
| Tuesday | 0.282703 | 0.120175 |

Calcula ahora los consumos y precios medios por mes.

consumo.groupby('mes')['consumo\_kwh','precio\_kwh']

|  | consumo\_kwh | precio\_kwh |
| --- | --- | --- |
| mes |  |  |
| 9 | 252.628 | 75.915174 |
| 10 | 218.408 | 86.918709 |
| 11 | 219.052 | 86.885759 |
| 12 | 246.768 | 95.380217 |

Ahora vamos a cargar los datos de precio de la electricidad en el mercado diario mayorista (distinto del precio al que se cobra la electricidad al consumidor).

precio\_md = pd.read\_csv("./precio\_md.csv", sep = ";", header=0)  
precio\_md.head()

|  | datetime | preciomd\_eurMw |
| --- | --- | --- |
| 0 | 2016-09-01 00:00:00 | 37.76 |
| 1 | 2016-09-01 01:00:00 | 36.00 |
| 2 | 2016-09-01 02:00:00 | 35.26 |
| 3 | 2016-09-01 03:00:00 | 35.26 |
| 4 | 2016-09-01 04:00:00 | 35.26 |

Transforma la columna datetime del nuevo DataFrame para que sea del tipo fecha y hora.

precio\_md['datetime'] = pd.to\_datetime(precio\_md['datetime'])

Ahora cruza los datos de consumo con los datos de precio del mercado diario que acabamos de cargar.

consumo\_pmd = pd.merge(consumo, precio\_md, on='datetime')  
consumo\_pmd.head()

|  | datetime | consumo\_kwh | precio\_kwh | coste\_eur | dia\_semana | fecha | hora | mes | preciomd\_eurMw |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2016-09-01 00:00:00 | 0.188 | 0.097510 | 0.018332 | Thursday | 2016-09-01 | 0 | 9 | 37.76 |
| 1 | 2016-09-01 01:00:00 | 0.124 | 0.097756 | 0.012122 | Thursday | 2016-09-01 | 1 | 9 | 36.00 |
| 2 | 2016-09-01 02:00:00 | 0.111 | 0.097713 | 0.010846 | Thursday | 2016-09-01 | 2 | 9 | 35.26 |
| 3 | 2016-09-01 03:00:00 | 0.129 | 0.099275 | 0.012806 | Thursday | 2016-09-01 | 3 | 9 | 35.26 |
| 4 | 2016-09-01 04:00:00 | 0.108 | 0.100793 | 0.010886 | Thursday | 2016-09-01 | 4 | 9 | 35.26 |

Calcula una nueva columna 'dif\_precio' con la diferencia entre el precio pagado por el consumidor cada hora y el precio de la electricidad en el mercado diario. Ojo, ten en cuenta que el precio del consumidor está en €/kWh y el precio del mercado está en €/MWh.

consumo\_pmd = consumo\_pmd.assign(dif\_precio = consumo\_pmd['precio\_kwh'] - (consumo\_pmd['preciomd\_eurMw']/1000))

consumo\_pmd.head()

|  | **datetime** | **consumo\_kwh** | **precio\_kwh** | **coste\_eur** | **dia\_semana** | **fecha** | **hora** | **mes** | **preciomd\_eurMw** | **dif\_precio** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2016-09-01 00:00:00 | 0.188 | 0.097510 | 0.018332 | Thursday | 2016-09-01 | 0 | 9 | 37.76 | 0.059750 |
| **1** | 2016-09-01 01:00:00 | 0.124 | 0.097756 | 0.012122 | Thursday | 2016-09-01 | 1 | 9 | 36.00 | 0.061756 |
| **2** | 2016-09-01 02:00:00 | 0.111 | 0.097713 | 0.010846 | Thursday | 2016-09-01 | 2 | 9 | 35.26 | 0.062453 |
| **3** | 2016-09-01 03:00:00 | 0.129 | 0.099275 | 0.012806 | Thursday | 2016-09-01 | 3 | 9 | 35.26 | 0.064015 |
| **4** | 2016-09-01 04:00:00 | 0.108 | 0.100793 | 0.010886 | Thursday | 2016-09-01 | 4 | 9 | 35.26 | 0.065533 |

Para terminar, calcula la diferencia de precios promedio para cada mes.

consumo\_pmd.groupby('mes')['dif\_precio'].mean()

mes

9 0.061850

10 0.063992

11 0.064543

12 0.067714

Name: dif\_precio, dtype: float64