**ACCESO LOCAL: Ficheros Excel**

¿Qué empresa no trabaja con Excel? **Nos vamos a encontrar los formatos de datos de Excel en cualquier sitio**. Las extensiones de archivo más habituales son .xslx y .xls. Por suerte, **pandas tiene una función para leer los formatos de archivo de Excel y un método para escribirlos**.

El problema que presenta este tipo de lectura de datos es que **no es un formato tan cerrado como el CSV**. En el CSV tenemos una estructura compacta, con todos los datos separados por comas y con una línea de cabecera en la primera fila. El Excel permite tener datos en un formato mucho más flexible, con tablas en cualquier sitio de las hojas, información en varias hojas y demás.

Teniendo esto en cuenta, y sabiendo bien el formato del Excel en cuestión, podremos leerlo sin problemas con pandas, debido a la cantidad de argumentos que tiene la función read\_excel que nos van a permitir adaptarnos hasta cierto punto a esa flexibilidad. [En la documentación tienes todo el detalle](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_excel.html).

**Pandas y Excel: Lectura**

El método para leer archivos excel tiene una lógica simiar al read\_csv pero se llama read\_excel. Leamos nuestro archivo de resultados de futbol de 2019, pero esta vez en formato excel:

[1]:

**import** pandas **as** pd

df**=**pd.read\_excel("./data/df\_liga\_2019.xlsx")

df

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

592 rows × 14 columns

Al igual que con read\_csv, vamos a ver algunos de los argumentos más interesantes del método y que nos ayudarán cuando la hoja no esté tan ordenadita como la que acabamos de leer:

**sheet\_name**

Dado que un "libro" excel (yo siempre lo he llamado hoja, pero puede llevar a confusión) puede tener varias hojas, este argumento permite indicar las hojas queremos leer usando su nombre o posición en el "libro". Si no se pone nada lee la primera según esté ordenado el fichero excel.

[3]:

df**=**pd.read\_excel("./data/df\_liga\_2019.xlsx", sheet\_name**=**"futbol\_2")

df

Captura de pantalla de computadora

Descripción generada automáticamente

[3]:

[4]:

df**=**pd.read\_excel("./data/df\_liga\_2019.xlsx",sheet\_name**=**0)

df

Imagen que contiene Calendario

Descripción generada automáticamente

[4]:

592 rows × 14 columns

Puedes indicarle una lista con las posiciones o los nombres de varias hojas, e incluso None si quieres cargar todas. En estos casos la función devuelve un diccionario. Prueba con alguna de tus hojas excel en local o con la que usamos aquí

**index\_col**

Tiene el mismo uso que en read\_csv, sirve para indicar la columna que funciona de índice (si lees varias hojas a la vez puedes pasarle una lista con los índices para cada una pero eso no lo veremos en esta sesión)

[5]:

df**=**pd.read\_excel("./data/df\_liga\_2019.xlsx", index\_col**=**"id\_partido")

df

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

[5]:

592 rows × 13 columns

**usecols y skiprows**

Si los datos no empiezan en la columna A y en la fila 1, leeremos mal estos si hacemos uso de la función tal y como hasta ahora:

[7]:

df**=**pd.read\_excel("./data/df\_liga\_2019.xlsx",sheet\_name**=**"futbol\_3")

df

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

[7]:

Si haces unos cálculos, verás que nuestros datos realmente van de la columna "C" a la "N" y, si te fijas además empiezan en la cuarta fila y que las filas 8 y 14 (ojo que en este caso empezamos en 1 a contar) no son buenas. La función nos permite tener en cuenta esto con los parámetros usecols con el que le diremos las columnas que queremos usar y skiprows que nos permite decir que líneas no incluir.

[ ]:

df**=**pd.read\_excel("./data/df\_liga\_2019.xlsx", sheet\_name**=**"futbol\_3", usecols**=**"C:N",skiprows**=**[0,1,2,

**Pandas y excel: Escritura**

Al igual que con el CSV, tenemos el método to\_excel(), para escribir el DataFame en un archivo Excel.

**Recuerda poner la extensión del Excel (.xlsx) en el nombre del archivo**. Tienes [el enlace a la documentación para ver más detalle](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_excel.html).

Aquí no usaremos el parámetro sep, pero sí el index si queremos que incluya o no el índice, con las mismas consideraciones que para read\_csv (index está a True por defecto, si el DataFrame no tiene un índice explícito se creará una columna en la excel llamada Unname:0 con los valores del índice implícito)

[ ]:

df.to\_excel("./data/df\_liga\_2019\_

Si además quieres saber más sobre como manejar hojas excel desde Python, te recomiendo que visites [la documentacion de openpyxl](https://openpyxl.readthedocs.io/en/stable/)