Análisis y Manipulación de Datos 3

Data BootCamp
Cámara Comercio Bilbao

Germán Alonso Lascurain
germanalonso@opendeusto.es

Índice de contenidos

1. Carga de DataFrames

Carga de DataFrames - TSV

Invocación:

```
import pandas as pd
df = pd.read_table(filepath_or_buffer)
```

Parámetros de interés:

- sep: carácter separador de campos
- header: registros a utilizar como nombre de columnas
- names: nombres de columas
- usecols: columnas a utilizar
- dtype: tipos dedatos
- true_values/false_values: valores verdadero/falso
- na_values: valores nulos
- parse_dates: transformar columnas a fecha
- compression: compresión del fichero
- decimal: separador decimal
- encoding: codificación del fichero
- error_bad_lines: lanzar excepción en errores

Carga de DataFrames - CSV

Invocación:

```
import pandas as pd
df = pd.read_csv(filepath_or_buffer)
```

Parámetros de interés:

- sep: carácter separador de campos
- header: registros a utilizar como nombre de columnas
- names: nombres de columas
- usecols: columnas a utilizar
- dtype: tipos dedatos
- true_values/false_values: valores verdadero/falso
- na_values: valores nulos
- parse_dates: transformar columnas a fecha
- compression: compresión del fichero
- decimal: separador decimal
- encoding: codificación del fichero
- error_bad_lines: lanzar excepción en errores

Carga de DataFrames - Excel

Invocación:

```
import pandas as pd
df = pd.read_excel(filepath_or_buffer)
```

Parámetros de interés:

- sheet_name: hoja del excel a parsear
- header: registros a utilizar como nombre de columnas
- names: nombres de columas
- usecols: columnas a utilizar
- dtype: tipos dedatos
- true_values/false_values: valores verdadero/falso
- na_values: valores nulos
- parse_dates: transformar columnas a fecha
- error_bad_lines: lanzar excepción en errores

Carga de DataFrames - Excel (Alternativa)

Invocación:

```
import pandas as pd
excel_file = pd.ExcelFile(filepath_or_buffer)
df = excel_file.parse()
```

Parámetros de interés:

- sheet_name: hoja del excel a parsear
- header: registros a utilizar como nombre de columnas
- names: nombres de columas
- usecols: columnas a utilizar
- dtype: tipos dedatos
- true_values/false_values: valores verdadero/falso
- na_values: valores nulos
- parse_dates: transformar columnas a fecha
- error_bad_lines: lanzar excepción en errores

Carga de DataFrames - JSON

Invocación:

```
import pandas as pd
df = pd.read_json(filepath_or_buffer)
```

Parámetros de interés:

- orient: MUY IMPORTANTE! Orientación de los registros
- dtype: tipos de datos
- convert_dates: transformar columnas a fecha
- encoding: codificación del fichero
- compression: compresión del fichero
- error_bad_lines: lanzar excepción en errores

Carga de DataFrames - SQL (tabla)

Invocación:

```
import pandas as pd
df = pd.read_sql_table(table, conexion)
```

Parámetros de interés:

- conexion: MUY IMPORTANTE! Conector SQLAlchemy
- parse_dates: transformar columnas a fecha
- columns: columnas a elegir de la tabla

Carga de DataFrames - SQL (query)

Invocación:

```
import pandas as pd
df = pd.read_sql_query(sql, conexion)
```

Parámetros de interés:

- sql: query a enviar contra la BBDD
- conexion: MUY IMPORTANTE! Conector SQLAlchemy
- parse_dates: transformar columnas a fecha

2. Resumen de DataFrames

Resumen de DataFrames

Ver principio:

```
import pandas as pd
df = pd.read_csv(...)
df.head()
```

Ver final:

```
df.tail()
```

Ver resumen:

```
df.info()
df.describe()
```

Resumen de DataFrames

Consultas:

```
import pandas as pd
df = pd.read_csv(...)
df.query('...consulta...')
```

Por tipo de dato:

```
df.select_dtypes([...])
```

Ordenar datos:

```
columnas = [...]
orden = [True, False, ...]
df.sort_values(columnas, ascending=orden)
```

3. Modificación de DataFrames

Modificación de DataFrames

Modificar nombre columnas:

```
import pandas as pd
df = pd.read_csv(...)
columnas = [...]
df.columns = columnas
```

Otra opción:

```
columnas = {'antigua': 'nueva', ...}
df = df.rename(columns=columns)
```

Modificación de DataFrames

Modificar tipo columnas:

```
import pandas as pd
df = pd.read_csv(...)
df[‘columna’] = df[‘columna’].astype(‘tipo’)
```

Otra opción:

```
dtype = {‘columna’: ‘tipo’, ...}
df = df.astype(dtype=dtype)
```

Modificación de DataFrames

Modificar contenido columnas:

```
import pandas as pd
df = pd.read_csv(...)
df[‘columna’] = ‘...MODIFICACIONES A NIVEL DE SERIES...’
```

Otra opción:

```
df = df.apply(‘...funcion...', axis=’columns’)
df[‘columna’] = df[‘columna’].apply(‘...funcion...')
df[‘columna’] = df[‘columna’].map(diccionario)
```

4. Agrupamientos de DataFrames

Agrupamientos de DataFrames

Agrupar columnas:

```
import pandas as pd
df = pd.read_csv(...)
columnas_a_agrupar = [...]
df_agrupado = df.groupby(columnas_a_agrupar)
```

Operaciones:

```
df_agrupado.sum()
df_agrupado.agg(['sum', 'min'])
df_agrupado.agg({'columna1': 'sum', 'columna2': 'min'})
```

Hay que tener en cuenta que el resultado es un multiindex

Eliminar duplicados

Eliminar duplicados:

```
import pandas as pd
df = pd.read_csv(...)
columnas_a_filtrar = [...]
df.drop_duplicates(columnas_a_filtrar, keep='first')
```

Con el argumento keep controlamos que registro de los duplicados nos quedarnos (first, last, False)

Germán Alonso Lascurain
germanalonso@opendeusto.es