Introducción a Pandas

Inteligencia de Negocio

Grado en Informática

Índice

Índice

Pandas


```
import numpy as np
import pandas as pd
```

Pandas ofrece

- Leer y procesar datos en ficheros Excel y csv.
- Trabajar con *DataFrame* como R.
- Trabajar con gran cantidad de datos a la vez.
- Visualizar fácilmente.

Tipos de datos de Pandas

Tipos de Datos

Series Lista de datos

```
data = pd.Series([1, 2, 3, 4, 5, 6, 7])
data.tail()
```

```
2    3
3    4
4    5
5    6
6    7
dtype: int64
```

Tipos de datos de Pandas

Tipos de Datos

DataFrame Conjunto de datos (filas) con distinto valor de dimensión (columnas).

```
articulos = ['Pantalones', 'Zapatos',
'Camisas']
precios = [15,  30, 10]
combinados = list(zip(articulos, precios))
# Option 1
data = pd.DataFrame(data=combinados,
                    columns=['Name', 'Price'])
# Option 2
data = pd.DataFrame({'Name': articulos,
                     'Price': precios})
data
```

```
          Name  Price
0  Pantalones     15
1     Zapatos     30
2     Camisas     10
```

Índice

Procesando datos (operaciones básicas)

Añadir nueva columna

```
data['Cost'] = np.array([10, 40, 5])
data
```

```
          Name   Price   Cost
0   Pantalones      15     10
1      Zapatos      30     40
2      Camisas      10      5
```

Crear nueva columna de anteriores

```
data['Profit']  = data['Price']-data['Cost']
data
```

```
          Name   Price   Cost   Profit
0   Pantalones      15     10        5
1      Zapatos      30     40      -10
2      Camisas      10      5        5
```

Procesando datos (operaciones básicas)

Filtrar datos

```
data[data['Profit'] > 0]
```

```
          Name  Price  Cost  Profit
0  Pantalones     15    10       5
2     Camisas     10     5       5
```

Eliminar filas

```
good_data = data[data['Profit'] > 0]
```

Modificando valores

Acceso por columnas

```
data['Cost']

0     10
1     40
2      5
Name: Cost, dtype: int64

data[['Cost','Profit']]

   Cost  Profit
0    10       5
1    40     -10
2     5       5
```

Acceso por filas

Acceso por filas

```
data.loc[1]
```

```
Name       Zapatos
Price           30
Cost            40
Profit         -10
Name: 1, dtype: object
```

```
data.loc[0, 'Price']
```

```
15
```

```
data.loc[[0, 2], ['Name','Price']]
```

```
          Name  Price
0  Pantalones     15
2     Camisas     10
```

Modificando valores

Eliminar valores imposible

```
bad_arts = data['Cost'] > data['Price']
data.loc[bad_arts,'Cost'] = data.loc[bad_arts,'Price']
data.loc[bad_arts,'Profit'] = 0
```

Columnas

Renombrar columnas

```
data.columns
```

```
Index(['Name', 'Price', 'Cost', 'Profit'], dtype='object')
```

Renombrar columnas

```
data.columns = ['Type', 'Price', 'Cost', 'Profit']
data
```

```
          Type  Price  Cost  Profit
0  Pantalones     15    10       5
1      Zapatos     30    30       0
2      Camisas     10     5       5
```

Índices

Índices

```
data.index

RangeIndex(start=0, stop=3, step=1)
```

Cambiar índices

```
data2 = data.set_index('Type')
data2

             Price  Cost  Profit
Type
Pantalones      15    10       5
Zapatos         30    30       0
Camisas         10     5       5
```

Índices

Acceso por índices (loc)

```
data2.loc[['Camisas','Zapatos']]
```

Type	Price	Cost	Profit
Camisas	10	5	5
Zapatos	30	30	0

Acceso por posición (iloc)

```
data2.iloc[[1,2]]
```

Type	Price	Cost	Profit
Zapatos	30	30	0
Camisas	10	5	5

Operaciones globales

Operaciones globales

```
data2.sum()
```

```
Price      55
Cost       45
Profit     10
dtype: int64
```

Parcialmente global

```
data2[data2['Profit'] > 0].mean()
```

```
Price      12.5
Cost        7.5
Profit      5.0
dtype: float64
```

Índice

Agrupando valores

Añadimos categoria

```
data['Category'] = ['Ropa', 'Calzado', 'Ropa']
data
```

```
          Name  Price  Cost  Profit Category
0  Pantalones     15    10       5     Ropa
1     Zapatos     30    30       0  Calzado
2     Camisas     10     5       5     Ropa
```

Agrupando

```
data.groupby('Category').mean()
```

```
          Price  Cost  Profit
Category
Calzado    30.0  30.0     0.0
Ropa       12.5   7.5     5.0
```

Otras operaciones

Aplicar una operación a una columna

```
def capitalizer(str):
    return str.upper()

data['Category'] = data['Category'].apply(capitalizer)
data
```

```
          Name  Price  Cost  Profit Category
0  Pantalones     15    10       5     ROPA
1     Zapatos     30    30       0  CALZADO
2     Camisas     10     5       5     ROPA
```

Otras operaciones

Añadir filas

```
news = {'Name': ['Corbatas'], 'Price': [10],
                    'Category': ['ROPA']]}
df = pd.DataFrame(news)
data = pd.concat([data, df], ignore_index=True)
data
```

	Category	Cost	Name	Price	Profit
0	ROPA	10	Pantalones	15	5
1	CALZADO	30	Zapatos	30	0
2	ROPA	5	Camisas	10	5
3	NaN	3	Corbatas	10	7

Otras operaciones

Eliminar tuplas con NA en algún campo

```
data.dropna()
```

	Category	Cost	Name	Price	Profit
0	ROPA	10	Pantalones	15	5
1	CALZADO	30	Zapatos	30	0
2	ROPA	5	Camisas	10	5

Tabla pivotadas

Hay tablas en las que cada fila es una información.

Tabla original

```
values = np.random.rand(30)
cat = np.random.choice(list('ABCD'), 30)
field = np.random.choice(['one', 'two'], 30)
df = pd.DataFrame({'field': field, 'type':  cat, 'value':
values})
df.head(9)
```

```
  field type      value
0   two    C   0.275455
1   two    D   0.779945
2   two    D   0.056872
3   one    D   0.298926
4   two    A   0.001402
5   two    D   0.762490
6   one    A   0.997514
7   two    C   0.544569
8   one    B   0.639121
```

Tablas pivotadas

La función *pivot_table* permite extraer los valores.

```
pivot = pd.pivot_table(df, index=['field'],
                       values=['value'],
                       columns=['type'],
                       aggfunc=np.sum)
pivot.head()
```

	value			
type	A	B	C	D
field				
one	1.498814	1.093041	2.185960	2.330501
two	0.727464	0.872249	1.783891	2.493141

¿Preguntas?