Módulo de adaptación

Master en Business Intelligence y Big Data

PROFESOR/A
Antonio Sarasa Cabezuelo

Escuela de organización industrial

www.eoi.es

Pandas

Pandas

- Pandas es una librería construida sobre Numpy que ofrece estructuras de datos de alto nivel que facilitan el análisis de datos desde Python.
- Se van a estudiar dos estructuras de datos:
  - Series.
  - DataFrame.
- Antes de trabajar con estas estructuras se importan las librerías necesarias:

```
In [1]: import numpy as np

In [2]: import pandas as pd

In [3]: from pandas import *
```

Pandas

- Series:
  - Una serie es un objeto como un array que esta formado por un array de datos y un array de etiquetas denominado índice.
  - El array de datos puede ser de 3 tipos:
    - ndarray
      - Si el array de datos en un ndarray, entonces el índice debe ser de la misma longitud que el array de datos.

```
In [5]: s = Series([1,2,3,4,5], index=['a', 'b', 'c', 'd', 'e'])

In [6]: s

Out[6]: a    1
        b    2
        c    3
        d    4
        e    5
        dtype: int64

In [7]: s.index

Out[7]: Index([u'a', u'b', u'c', u'd', u'e'], dtype='object')
```

Pandas

- Si ningún índice es pasado, entonces se crea uno formado por valores que van desde [0,…,len(data)-1]

```
In [8]: Series([1,2,3,4,5])
Out[8]: 0    1
        1    2
        2    3
        3    4
        4    5
        dtype: int64
```

- Diccionario
  - Si se pasa un índice entonces los valores del diccionario son asociados a los valores del índice.

```
In [10]: d = {'a' : 0., 'b' : 1., 'c' : 2.}

In [11]: Series(d, index=['b', 'c', 'd', 'a'])

Out[11]: b    1
         c    2
         d  NaN
         a    0
         dtype: float64
```

Pandas

- Si no se proporciona un índice, entonces se construye el índice a partir de las claves ordenadas del diccionario.

```
In [12]: d = {'a' : 0., 'b' : 1., 'c' : 2.}

In [13]: Series(d)

Out[13]: a    0
         b    1
         c    2
         dtype: float64
```

- Valor escalar
  - En este caso se debe proporcionar un índice, y el valor será repetido tantas veces como la longitude del índice.

```
In [14]: Series(5., index=['a', 'b', 'c', 'd', 'e'])

Out[14]: a    5
         b    5
         c    5
         d    5
         e    5
         dtype: float64
```

Pandas

- Las series actúan de forma similar a ndarray, siendo un argumento válido de funciones de Numpy.

```
In [15]: s = Series([1,2,3,4,5], index=['a', 'b', 'c', 'd', 'e'])

In [16]: s[0]
Out[16]: 1

In [17]: s[:3]
Out[17]: a    1
         b    2
         c    3
         dtype: int64

In [18]: s[s > s.median()]
Out[18]: d    4
         e    5
         dtype: int64

In [19]: s[[4, 3, 1]]
Out[19]: e    5
         d    4
         b    2
         dtype: int64

In [20]: np.exp(s)
Out[20]: a      2.718282
         b      7.389056
         c     20.085537
         d     54.598150
         e    148.413159
         dtype: float64
```

Pandas

- Las series actúan como un diccionario de tamaño fijo en el que se pueden gestionar los valores a través de los índices.

```
In [21]: s = Series([1,2,3,4,5], index=['a', 'b', 'c', 'd', 'e'])

In [22]: s['a']

Out[22]: 1

In [23]: s['e'] = 12.

In [24]: s

Out[24]: a     1
         b     2
         c     3
         d     4
         e    12
         dtype: int64

In [25]: 'e' in s

Out[25]: True

In [26]: 'f' in s

Out[26]: False
```

Pandas

- Sobre las series se pueden realizar operaciones vectoriales.

```
In [27]: s = Series([1,2,3,4,5], index=['a', 'b', 'c', 'd', 'e'])

In [28]: s + s

Out[28]: a     2
         b     4
         c     6
         d     8
         e    10
         dtype: int64

In [29]: s * 2

Out[29]: a     2
         b     4
         c     6
         d     8
         e    10
         dtype: int64
```

Pandas

- La principal diferencia entre series y ndarray es que las operaciones entre series alinean automáticamente los datos basados en las etiquetas, de forma que pueden realizarse cálculos sin tener en cuenta si las series sobre las que se operan tienen las mismas etiquetas.

```
In [30]: s = Series([1,2,3,4,5], index=['a', 'b', 'c', 'd', 'e'])

In [31]: s[1:] + s[:-1]

Out[31]: a   NaN
         b     4
         c     6
         d     8
         e   NaN
         dtype: float64
```

Pandas

- Observar que el resultado de una operación entre series no alineadas será la unión de los índices. Si una etiqueta no se encuentra en una de las series, entonces se le asocia el valor nulo.
- Los datos y el índice de una serie tienen un atributo name que puede asociarle un nombre.

```
In [32]: s = Series([1,2,3,4,5], index=['a', 'b', 'c', 'd', 'e'])

In [33]: s.name="datos"

In [34]: s.index.name="indice"

In [35]: s

Out[35]: indice
         a          1
         b          2
         c          3
         d          4
         e          5
         Name: datos, dtype: int64
```

Pandas

- Dataframe
  - Es una estructura que contiene una colección ordenada de columnas cada una de las cuales puede tener valores de diferentes tipos.
  - Un dataframe está formado por datos ,y opcionalmente un índice(etiquetas de las filas) y un conjunto de columnas(etiquetas de las columnas). En caso de no existir un índice, entonces se genera a partir de los datos.

Pandas

- Los datos de un dataframe pueden proceder de:
  - Diccionario de ndarrays. Los arrays deben ser de la misma longitud. En caso de existir un índice, éste debe ser de la misma longitud que los arrays, y en caso de no existir entonces se genera como índice la secuencia de números 0…longitud(array)-1

```
In [1]: d = {'one' : [1., 2., 3., 4.], 'two' : [4., 3., 2., 1.]}
```

```
In [6]: DataFrame(d)
```

Out[6]:

	one	two
0	1	4
1	2	3
2	3	2
3	4	1

```
In [7]: DataFrame(d, index=['a', 'b', 'c', 'd'])
```

Out[7]:

	one	two
a	1	4
b	2	3
c	3	2
d	4	1

Pandas

– Lista de diccionarios.

```
In [8]: data2 = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]
```

```
In [9]: DataFrame(data2)
```

Out[9]:

	a	b	c
0	1	2	NaN
1	5	10	20

```
In [10]: DataFrame(data2, index=['first', 'second'])
```

Out[10]:

	a	b	c
first	1	2	NaN
second	5	10	20

```
In [11]: DataFrame(data2, columns=['a', 'b'])
```

Out[11]:

	a	b
0	1	2
1	5	10

Pandas

– Diccionario de tuplas.

```
In [12]: DataFrame({('a', 'b'): {('A', 'B'): 1, ('A', 'C'): 2},
                    ('a', 'a'): {('A', 'C'): 3, ('A', 'B'): 4},
                    ('a', 'c'): {('A', 'B'): 5, ('A', 'C'): 6},
                    ('b', 'a'): {('A', 'C'): 7, ('A', 'B'): 8},
                    ('b', 'b'): {('A', 'D'): 9, ('A', 'B'): 10}})
```

Out[12]:

		a			b	
		a	b	c	a	b
A	B	4	1	5	8	10
	C	3	2	6	7	NaN
	D	NaN	NaN	NaN	NaN	9

Pandas

– Diccionario de series. El índice que resulta es la union de los índices de las series. En caso de existir diccionarios anidados,entonces primero se convierten en diccionarios. Además si no se pasan columnas, se toman como tales la lista ordenada de las claves de los diccionarios.

```
In [13]: d = {'one' : Series([1., 2., 3.], index=['a', 'b', 'c']), 'two' : Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}

In [14]: df = DataFrame(d)

In [15]: df
```

Out[15]:

	one	two
a	1	1
b	2	2
c	3	3
d	NaN	4

Pandas

```
In [16]: DataFrame(d, index=['d', 'b', 'a'])
```

Out[16]:

	one	two
d	NaN	4
b	2	2
a	1	1

```
In [17]: DataFrame(d, index=['d', 'b', 'a'], columns=['two', 'three'])
```

Out[17]:

	two	three
d	4	NaN
b	2	NaN
a	1	NaN

Pandas

– Las etiquetas de las columnas y de las filas pueden ser accedidas a través de los atributos indice y columnas. Observar que cuando un conjunto particular de columnas se pasa como argumento con el diccionario, entonces las columans sobreescriben a las claves del diccionario.

```
In [18]: df.index
Out[18]: Index([u'a', u'b', u'c', u'd'], dtype='object')

In [19]: df.columns
Out[19]: Index([u'one', u'two'], dtype='object')
```

Pandas

– Array estructurado.

```
In [20]: data = np.zeros((2,),dtype=[('A', 'i4'),('B', 'f4'),('C', 'a10')])
```

```
In [21]: data[:] = [(1,2.,'Hello'),(2,3.,"World")]
```

```
In [22]: DataFrame(data)
```

Out[22]:

	A	B	C
0	1	2	Hello
1	2	3	World

```
In [23]: DataFrame(data, index=['first', 'second'])
```

Out[23]:

	A	B	C
first	1	2	Hello
second	2	3	World

```
In [24]: DataFrame(data, columns=['C', 'A', 'B'])
```

Out[24]:

	C	A	B
0	Hello	1	2
1	World	2	3

Pandas

- Existen otras formas alternativas de construir un dataframe:
  - DataFrame.from_dict
  - DataFrame.from_records
  - DataFrame.from_items

- A continuación se van a presentar algunas acciones de manipulación con dataframes.

Pandas

- Manipulación de un dataframe
  - Un dataframe es como un diccionario de series indexado, por lo que se pueden usar las mismas operaciones usadas con los diccionarios.

```
In [25]: df['one']
Out[25]: a      1
         b      2
         c      3
         d    NaN
         Name: one, dtype: float64

In [26]: df['three'] = df['one'] * df['two']

In [27]: df['flag'] = df['one'] > 2

In [28]: df
```

Out[28]:

	one	two	three	flag
a	1	1	1	False
b	2	2	4	False
c	3	3	9	True
d	NaN	4	NaN	False

Pandas

- Se pueden borrar columnas

```
In [29]: del df['two']

In [30]: three = df.pop('three')

In [31]: df
```

Out[31]:

	one	flag
a	1	False
b	2	False
c	3	True
d	NaN	False

Pandas

- Cuando se inserta un valor escalar, entonces se propaga a toda la columna.

```
In [32]: df['foo'] = 'bar'

In [33]: df
```

Out[33]:

	one	flag	foo
a	1	False	bar
b	2	False	bar
c	3	True	bar
d	NaN	False	bar

Pandas

- Cuando se inserta una serie que no tiene el mismo índice, entonces se crea el índice para el dataframe.

```
In [34]: df['one_trunc'] = df['one'][:2]

In [35]: df
```

Out[35]:

	one	flag	foo	one_trunc
a	1	False	bar	1
b	2	False	bar	2
c	3	True	bar	NaN
d	NaN	False	bar	NaN

Pandas

- Las columnas por defecto se insertan al final, sin embargo se puede elegir el lugar de inserción usando la función insert

```
In [36]: df.insert(1, 'bar', df['one'])

In [37]: df
```

Out[37]:

	one	bar	flag	foo	one_trunc
a	1	1	False	bar	1
b	2	2	False	bar	2
c	3	3	True	bar	NaN
d	NaN	NaN	False	bar	NaN

Pandas

- Indexación/Selección
  - Selección por columna

```
In [46]: df['one']

Out[46]: a     1
         b     2
         c     3
         d   NaN
         Name: one, dtype: float64
```

  - Selección de fila por etiqueta

```
In [47]: df.loc['b']

Out[47]: one              2
         bar              2
         flag         False
         foo            bar
         one_trunc        2
         Name: b, dtype: object
```

Pandas

- Selección de fila por entero.

```
In [48]: df.iloc[2]

Out[48]: one              3
         bar              3
         flag          True
         foo            bar
         one_trunc      NaN
         Name: c, dtype: object
```

- Selección de rangos

```
In [49]: df[5:10]
```

Out[49]:

	one	bar	flag	foo	one_trunc

Pandas

- Alineamiento y aritmética
  - Cuando se alinea un dataframe se realiza sobre las columnas y el índice, de manera que se hace la unión de las etiquetas de las columnas y de las filas.

```
In [56]: randn = np.random.randn

In [57]: df = DataFrame(randn(10, 4), columns=['A', 'B', 'C', 'D'])

In [58]: df2 = DataFrame(randn(7, 3), columns=['A', 'B', 'C'])

In [59]: df + df2
```

Out[59]:

	A	B	C	D
0	0.546287	0.865511	1.682961	NaN
1	2.519263	-1.110058	-0.726808	NaN
2	-0.204730	-1.706083	-2.449516	NaN
3	0.616714	0.839159	-1.405825	NaN
4	1.602854	0.049653	0.920070	NaN
5	-0.019357	1.454747	0.999837	NaN
6	3.069917	2.602170	-1.685954	NaN
7	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN

Pandas

- Cuando se operan con dataframe y series, se alinea el índice de las series sobre las columnas del dataframe.

```
In [60]: df - df.iloc[0]
```

Out[60]:

	A	B	C	D
0	0.000000	0.000000	0.000000	0.000000
1	2.086730	-0.100902	-1.628989	1.292327
2	-0.362595	-0.248868	-2.055229	-0.085145
3	0.767276	-0.736967	-1.445053	-0.322223
4	1.119372	-0.656277	0.893438	0.282466
5	-1.115363	2.131909	-0.275902	1.816295
6	2.331868	2.070876	-2.003618	0.600362
7	1.211905	0.146791	-0.887329	1.097930
8	1.208581	1.006734	-0.479561	1.360532
9	0.311178	-0.203858	-0.138574	0.433787

Pandas

- Se pueden hacer operaciones con escalares.

```
In [61]: df * 5 + 2
```

Out[61]:

	A	B	C	D
0	2.066368	2.239286	5.606465	-3.119945
1	12.500019	1.734776	-2.538482	3.341692
2	0.253394	0.994946	-4.669679	-3.545671
3	5.902746	-1.445551	-1.618799	-4.731059
4	7.663228	-1.042099	10.073654	-1.707616
5	-3.510446	12.898833	4.226953	5.961529
6	13.725707	12.593665	-4.411623	-0.118136
7	8.125892	2.973241	1.169821	2.369707
8	8.109272	7.272955	3.208658	3.682718
9	3.622255	1.219993	4.913596	-0.951010

Pandas

- Se pueden hacer operaciones lógicas.

```
In [66]: df1 = DataFrame({'a' : [1, 0, 1], 'b' : [0, 1, 1] }, dtype=bool)

In [67]: df2 = DataFrame({'a' : [0, 1, 1], 'b' : [1, 1, 0] }, dtype=bool)

In [68]: df1 & df2
```

Out[68]:

	a	b
0	False	False
1	False	True
2	True	False

Pandas

- Transposición.
  - Para transponer se utilizar el atributo T.

```
In [38]: df[:5].T
```

Out[38]:

	a	b	c	d
one	1	2	3	NaN
bar	1	2	3	NaN
flag	False	False	True	False
foo	bar	bar	bar	bar
one_trunc	1	2	NaN	NaN

Pandas

- Visualización
  - Hay varias formas de visualizar la información de un dataframe:
    - Todos los datos a partir del nombre del dataframe.

```
In [40]: df
```

Out[40]:

	one	bar	flag	foo	one_trunc
a	1	1	False	bar	1
b	2	2	False	bar	2
c	3	3	True	bar	NaN
d	NaN	NaN	False	bar	NaN

Pandas

- Un resumen de la información contenido, usando el método info()

```
In [41]: df.info()
<class 'pandas.core.frame.DataFrame'>
Index: 4 entries, a to d
Data columns (total 5 columns):
one          3 non-null float64
bar          3 non-null float64
flag         4 non-null bool
foo          4 non-null object
one_trunc    2 non-null float64
dtypes: bool(1), float64(3), object(1)
```

Pandas

- Como una cadena usando el método to_string()

```
In [43]: print(df.to_string())
```

	one	bar	flag	foo	one_trunc
a	1	1	False	bar	1
b	2	2	False	bar	2
c	3	3	True	bar	NaN
d	NaN	NaN	False	bar	NaN

- Se puede configurar el número de columnas con display.width()

Pandas

- Interoperabilidad con Numpy
  - Siempre que los datos incluidos en un dataframe entonces pueden ser usados con funciones de Numpy.

```
In [69]: np.exp(df)
```

Out[69]:

	A	B	C	D
0	1.013362	1.049021	2.057091	0.359159
1	8.166201	0.948337	0.403453	1.307790
2	0.705167	0.817904	0.263438	0.329844
3	2.182670	0.502023	0.484926	0.260224
4	3.103857	0.544210	5.026534	0.476388
5	0.332176	8.844242	1.561100	2.208483
6	10.434748	8.320589	0.277392	0.654668
7	3.404774	1.214883	0.847016	1.076744
8	3.393475	2.870800	1.273452	1.400100
9	1.383271	0.855558	1.790902	0.554215

Pandas

- Con el método dot() se pueden realizar multiplicaciones.

```
In [70]: df.T.dot(df)
```

Out[70]:

	A	B	C	D
A	16.237908	2.775302	-3.383723	-2.509654
B	2.775302	11.305355	-1.747232	2.829424
C	-3.383723	-1.747232	8.523037	0.895589
D	-2.509654	2.829424	0.895589	5.987156

Pandas

- Por ultimo indicar que las columnas de un dataframe pueden ser accedidas como si fueran atributos al estilo de Python.

```
In [73]: df.A

Out[73]: 0    0.013274
         1    2.100004
         2   -0.349321
         3    0.780549
         4    1.132646
         5   -1.102089
         6    2.345141
         7    1.225178
         8    1.221854
         9    0.324451
         Name: A, dtype: float64
```

Pandas

- Para profundizar:

http://pandas.pydata.org/pandas-docs/stable/index.html