ORDENAR I AGRUPAR DADES NO DISPONIBLES

En aquest article, començarem descobrint una nova funció que ens permetrà generar una sèrie de dades agrupades entorn dels valors d'una columna. El primer pas, com és habitual, és carregar la llibreria i crear un nou objecte amb la base de dades que hem fet servir en l'article anterior:

```
import pandas as pd

data_frame = pd.read_csv('insurance.csv')
```

Anem a veure com es distribueixen els valors de *region:*

```
data_frame.groupby('region').size()
```

```
region
northeast    324
northwest    325
southeast    364
southwest    325
dtype: int64
```

Aquí veiem quantes observacions tenen cada una de les diferents regions de la taula mare.

Si volem veure quin és el valor màxim de *bmi* per a cada regió, fem:

```
data_frame.groupby('region')['bmi'].max()
```

```
region
northeast    48.07
northwest    42.94
southeast    53.13
southwest    47.60
Name: bmi, dtype: float64
```

I, si volem saber la mitjana de *bmi* per a cada regió, teclegem:

```
data_frame.groupby('region')['bmi'].mean()
```

```
region
northeast    29.173503
northwest    29.199785
southeast    33.355989
southwest    30.596615
Name: bmi, dtype: float64
```

Aquí, veiem que la mitjana de valors de *bmi* més elevada la trobem a la regió *southeast* i la més baixa a la *northeast*.

Tenim l'opció de generar la llista d'aquestes mitjanes de manera ordenada. Ho podem fer amb la funció *sorted().* El codi seria aquest:

```
sorted(data_frame.groupby('region')['bmi'].mean())
```

La llista que obtindríem seria:

```
[29.17350308641976, 29.199784615384626, 30.59661538461538,33.35598901098903]
```

Ara, farem una agrupació més complexa. Mostrarem les mitjanes dels preus de les pòlisses agrupades, en un primer nivell, per la regió i, en un segon nivell, per la variable *smoker*. Aquest és el codi:

```
data_frame.groupby(['region', 'smoker'])['charges'].mean()
```

```
region     smoker
northeast  no          9165.531672
           yes        29673.536473
northwest  no          8556.463715
           yes        30192.003182
southeast  no          8032.216309
           yes        34844.996824
southwest  no          8019.284513
           yes        32269.063494
Name: charges, dtype: float64
```

Veiem clarament que, independentment de la regió, la mitjana de preus sempre és més alta, si la persona és fumadora, que quan no ho és.

Si ara mantenim l'agrupació de la regió, però canviem la variable smoker per sex en el segon nivell de l'agrupació, necessitem només fer aquest petit canvi:

```
data_frame.groupby(['region', 'sex'])['charges'].mean()
```

La taula que obtenim és:

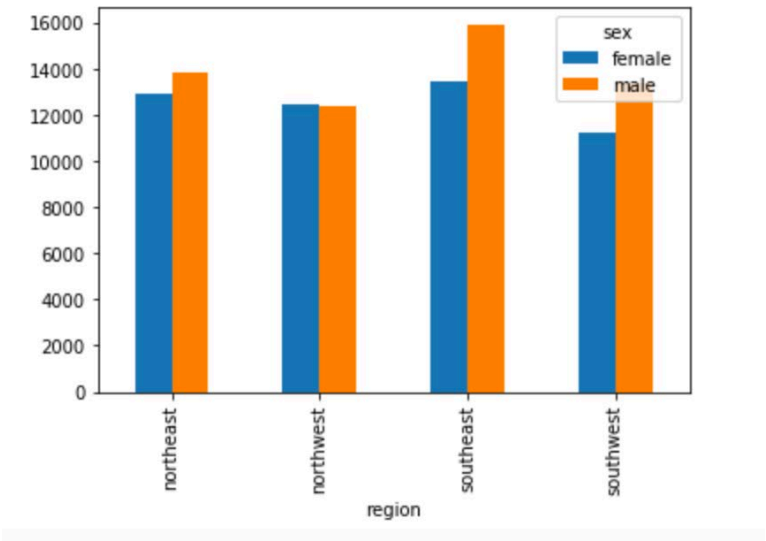
```
region     sex
northeast  female    12953.203151
           male      13854.005374
northwest  female    12479.870397
           male      12354.119575
southeast  female    13499.669243
           male      15879.617173
southwest  female    11274.411264
           male      13412.883576
Name: charges, dtype: float64
```

Veiem que, en totes les regions, excepte a *southwest*, la mitjana de preus és més alta per als homes que per a les dones.

Si assignem aquesta agrupació a una nova variable i apliquem la funció *plot.bar()*, obtindrem una gràfica de barres coherent amb l'agrupació:

group = data_frame.groupby(['region', 'sex'])['charges'].mean()

group.unstack(fill_value=0).plot.bar()



El gràfic mostra, d'una manera molt més visual, el que la taula de dalt ens deia.

Si, donada la base de dades *data_frame*, volem mostrar-la de manera ordenada segons els valors d'una variable determinada, només ens cal fer servir la funció *sort_values().* Ho farem així:

data_frame.sort_values(['age','bmi'])

Si volem mostrar només els primers 12 registres d'aquest ordre, haurem d'aplicar la funció *head()* i especificar el nombre de registres que volem mostrar:

data_frame.sort_values(['age','bmi']).head(12)

La taula que obtenim, doncs, serà aquesta:

	age	sex	bmi	children	smoker	region	charges
172	18	male	15.960	0	no	northeast	1694.79640
250	18	male	17.290	2	yes	northeast	12829.45510
359	18	female	20.790	0	no	southeast	1607.51010
1212	18	male	21.470	0	no	northeast	1702.45530
1033	18	male	21.565	0	yes	northeast	13747.87235
1282	18	female	21.660	0	yes	northeast	14283.45940
1080	18	male	21.780	2	no	southeast	11884.04858
295	18	male	22.990	0	no	northeast	1704.56810
1041	18	male	23.085	0	no	northeast	1704.70015
940	18	male	23.210	0	no	southeast	1121.87390
1023	18	male	23.320	1	no	southeast	1711.02680
121	18	male	23.750	0	no	northeast	1705.62450

El primer criteri per ordenar les files correspon al valor de l'edat (*age*) i el segon correspon a *bmi*. El codi, primer, ordena per edats i, en cas que hi hagi files amb la mateixa edat, el codi aplica un segon criteri, el *bmi*. Per defecte, l'ordre s'aplica sempre de manera ascendent, és a dir, de valor més petit a més gran. Si això ho volem invertir, haurem d'afegir un altre paràmetre d'aquesta manera:

data_frame.sort_values(['age','bmi'], ascending=[False, False]).head(12)

Ara, la taula que obtenim és la següent:

	age	sex	bmi	children	smoker	region	charges
534	64	male	40.480	0	no	southeast	13831.11520
768	64	female	39.700	0	no	southwest	14319.03100
199	64	female	39.330	0	no	northeast	14901.51670
418	64	male	39.160	1	no	southeast	14418.28040
603	64	female	39.050	3	no	southeast	16085.12750
635	64	male	38.190	0	no	northeast	14410.93210
752	64	male	37.905	0	no	northwest	14210.53595
1241	64	male	36.960	2	yes	southeast	49577.66240
801	64	female	35.970	0	no	southeast	14313.84630
335	64	male	34.500	0	no	southwest	13822.80300
420	64	male	33.880	0	yes	southeast	46889.26120
328	64	female	33.800	1	yes	southwest	47928.03000

El paràmetre *ascendint*, per defecte, assoleix el valor *True* i és per això que les taules sempre mostren un ordre ascendent. Si especifiquem que l'*ascending* ha de tenir un valor *False*, obtindrem una taula amb ordre descendent, com aquesta última que hem obtingut.

Fins ara, hem suposat que totes les cel·les d'una taula tenien valors. La realitat, en canvi, sempre és diferent i aquesta ens presentarà, sovint, taules que tenen cel·les sense valors. Aquests valors es diuen *Nas*.

Una de les funcions més utilitzades del paquet pandes, per veure si tenim *Nas* o no, és la funció *isna()*. Aquesta funció torna, com a resultat, False, quan el valor existeix i torna True, quan el valor és *Na*. Per exemple, si volem saber si entre els registres 230 i 238 de la columna *smoker* hi ha *Nas* o no, necessitem escriure aquest codi:

```
data_frame['smoker'][230:239].isna()
```

La llista de valors que obtenim és:

```
230    False
231    False
232    False
233    False
234    False
235    False
236    False
237    False
238    False
Name: smoker, dtype: bool
```

Observant els resultats, podem concloure que, en aquest interval de valors, no tenim cap *Na*, ja que tots els valors que torna la funció *isna()* són *False*.

Anem, ara, a copiar la base de dades *data_frame* en una nova variable:

```
data_frame2 = data_frame[:]
```

Important: data_frame2 i data_frame no són el mateix objecte. No hem apuntat realment data_frame a la nova variable, ja que n'hem fet servir una còpia amb l'operador [:].

```
data_frame2['smoker'][230] = None
```

```
data_frame2['smoker'][230:239].isna()
```

La taula que obtenim ara sí que té un *Na*, segons indica la funció *isna()*:

```
230     True
231    False
232    False
233    False
234    False
235    False
236    False
237    False
238    False
Name: smoker, dtype: bool
```

La funció *notna()* funciona justament al contrari. Allà on hi ha un *Na*, torna un *False* i, on no n'hi ha, torna un *True*. Ho veiem?

data_frame2['smoker'][230:239].notna()

```
230    False
231     True
232     True
233     True
234     True
235     True
236     True
237     True
238     True
Name: smoker, dtype: bool
```

Seguim!

Descobreix tot el que Barcelona Activa pot fer per a tu

Acompanyament durant tot el procés de recerca de feina

barcelonactiva.cat/treball

Suport per posar en marxa la teva idea de negoci

barcelonactiva.cat/emprenedoria

Serveis a les empreses i iniciatives socioempresarials

barcelonactiva.cat/empreses

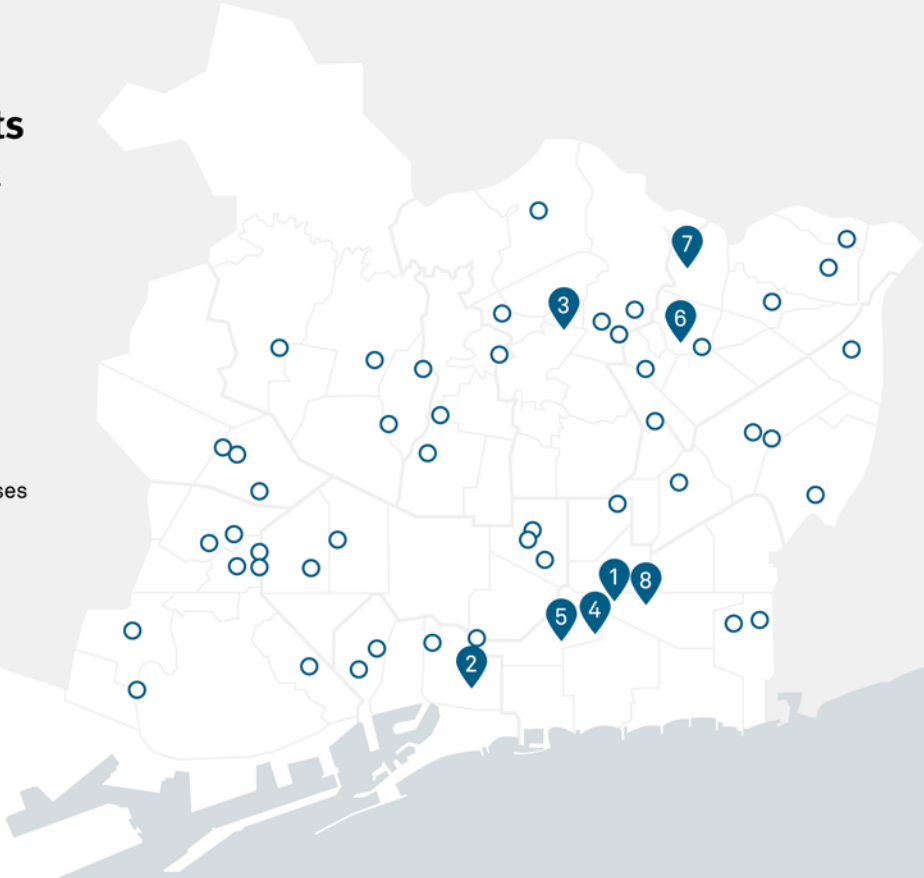
Formació tecnològica i gratuïta per a la ciutadania

barcelonactiva.cat/cibernarium

Xarxa d'equipaments de Barcelona Activa

1. Seu Central Barcelona Activa
   Porta 22
   Centre per a la Iniciativa Emprenedora Glòries
   Incubadora Glòries
2. Convent de Sant Agustí
3. Ca n'Andalet
4. Oficina d'Atenció a les Empreses
   Cibernàrium
   Incubadora MediaTIC
5. Incubadora Almogàvers
6. Parc Tecnològic
7. Nou Barris Activa
8. innoBA

○ Punts d'atenció a la ciutat



Barcelona Activa

Ajuntament de Barcelona

Darrera actualització 2019

Cofinançat per:

UNIÓ EUROPEA
Fons Europeu de Desenvolupament Regional

Segueix-nos a les xarxes socials:

barcelonactiva.cat/cibernarium
barcelonactiva
barcelonactiva
company/barcelona-activa