Pràctica 4: els paquets `numpy` i `matplotlib`

8 de desembre de 2020

1 Els paquets `numpy` i `matplotlib`

1.1 El paquet numpy

El paquet `numpy` és un paquet fonamental per fer càlculs científics amb **Python**. Proporciona matrius multidimensionals (`array`) com una generalització de les llistes i funcions per a poder fer operacions amb elles, incloent operacions matemàtiques, lògiques, manipulacions, ordenacions, operacions d'àlgebra lineal i estadística.

1.1.1 Matrius

Comencem definint matrius, que s'han de definir amb la funció `array` del paquet `numpy`, amb la mateixa sintaxis de les llistes. Per comprovar que es tracta d'on objecte diferent d'una simple llista, li podem demanar les propietats.

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

[2]:
```
a=np.array([1,2,3])
```

[3]:
```
print(a.shape)
```

```
(3,)
```

[4]:
```
print(type(a))
```

```
<class 'numpy.ndarray'>
```

[5]:
```
b=np.array([[1,2,3],[4,5,6]])
```

[6]:
```
print(b)
```

```
[[1 2 3]
 [4 5 6]]
```

[7]:
```
print(b.shape)
```

```
(2, 3)
```

[8]:
```
print(b[0,0],b[1,1])
```

```
1 5
```

Podem definir matrius de la mida que necessitem amb valors fixats. Per això podem utilitzar la funció `zeros`, `ones` o `full`, que tenen una sintaxis molt semblant:

[9]:
```
a=np.zeros((2,3))
print(a)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

[10]:
```
b=np.ones((3,2))
print(b)
```

```
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

[11]:
```
c=np.full((3,3),np.pi)
print(c)
```

```
[[3.14159265 3.14159265 3.14159265]
 [3.14159265 3.14159265 3.14159265]
 [3.14159265 3.14159265 3.14159265]]
```

[12]:
```
d=np.pi*np.ones((3,3))
print(d)
```

```
[[3.14159265 3.14159265 3.14159265]
 [3.14159265 3.14159265 3.14159265]
 [3.14159265 3.14159265 3.14159265]]
```

[13]:
```
print(d-c)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

Igual que a les llistes, a les matrius el nom indica les posicions de memòria, pel que si assignem a un altre nom una matriu o submatriu ja definida, els valors d'una i de l'altra ocupen les mateixes posicions de memòria:

[14]:
```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(a)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

[15]:
```
b=a[:2,1:3]
print(b)
```

```
[[2 3]
 [5 6]]
```

[16]:
```
b[1,1]=99
```

[17]:
```
print(a)
```

```
[[ 1  2  3]
 [ 4  5 99]
 [ 7  8  9]]
```

En el cas particular de les matrius 1×1, no és el mateix considerar-les com a escalars, que com a matrius:

[18]:
```
c=a[0,0]
print(c)
```

```
1
```

[19]:
```
c=23
print(a)
```

```
[[ 1  2  3]
 [ 4  5 99]
 [ 7  8  9]]
```

[20]:
```
d=a[0:1,0:1]
print(d)
```

```
[[1]]
```

[21]:
```
d[0,0]=23
print(a)
```

```
[[23  2  3]
 [ 4  5 99]
 [ 7  8  9]]
```

Si enlloc d'identificar amb una altra variable una matriu existent volem treballar amb una còpia de la primera sense que es modifiqui l'original, podem utilitzar la funció `copy`.
Aprofitem aquest exemple per a veure com es treu tota una columna, fent referència a `::` als possibles índexos de les files (totes les files).

[22]:
```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
```

[23]:
```
b=np.copy(a[::,1:3])
```

[24]:
```
print("a=",a)
print("b=",b)
```

```
a= [[1 2 3]
 [4 5 6]
 [7 8 9]]
b= [[2 3]
 [5 6]
 [8 9]]
```

[25]:
```
b[1,1]=99
```

[26]:
```
print("a=",a)
print("b=",b)
```

```
a= [[1 2 3]
 [4 5 6]
 [7 8 9]]
b= [[ 2  3]
 [ 5 99]
 [ 8  9]]
```

Podem utilitzar sintaxi de les llistes a les definicions de les matrius:

[27]:
```
a=np.array([[x**2+y for x in range(5)] for y in range(4)])
```

[28]:
```
print(a)
```

```
[[ 0  1  4  9 16]
 [ 1  2  5 10 17]
 [ 2  3  6 11 18]
 [ 3  4  7 12 19]]
```

Exercici Donada una llista de números `llista` definim la matriu de Van-der-Monde com la matriu quadrada amb tantes files i columnes com elements té la llista i on, a la fila `i`(començant per la fila `i=0`) hi ha a cada posició el valor que hi ha a `llista` a la mateixa posició, però elevada a `i`.
Per exemple `VdM([1,2,3])` hauria de ser la matriu $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{pmatrix}$.
Definiu una funció que es digui `VdM` que, a partir d'una llista donada, torni la matriu de Van-der-Monde corresponent.
Comproveu el funcionament executant `VdM([1,2,3,4,5])`.

Podem utilitzar matrius amb nombres enters per accedir a posicions concretes d'altres matrius (o sigui, la primera matriu serveix per a decidir l'índex de la segona i agafar-ne els valors):

[29]:
```
a=np.array(list(range(1,10)))
a.shape=(3,3)
print(a)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

[30]:
```
print(np.array([a[0,0],a[1,1],a[0,2]]))
```

```
[1 5 3]
```

[31]:
```
print(a[[0,1,0],[0,1,2]])
```

```
[1 5 3]
```

[32]:
```
i=np.array([0,1,0])
j=np.array([0,1,2])
print(a[i,j])
```

```
[1 5 3]
```

El paquet `numpy`té el seu propi `range` per definir matrius:

[33]:
```
a=np.arange(1,20,2)
print(a)
```

```
[ 1  3  5  7  9 11 13 15 17 19]
```

També es poden demanar les posicions dels nombres d'una matriu que compleixen certa condició, obtenint una matriu amb valors booleans (`True` o `False`). Aquesta matriu es pot utilitzar per extreure els valors corresponents a `True`.

[34]:
```
print(a>4)
```

```
[False False  True  True  True  True  True  True  True  True]
```

[35]:
```
print(a[a>4])
```

```
[ 5  7  9 11 13 15 17 19]
```

Sobre la sintaxis per a accedir als coeficients d'una matriu es pot veure com `inici:final:pas` i si es deixa buit s'entèn que `inici=0`, `final=últim` i `pas=-1`.

[36]:
```
a=np.arange(1,21)
a.shape=(4,5)
print(a)
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]]
```

[37]:
```
print(a[::-1,::])
```

```
[[16 17 18 19 20]
 [11 12 13 14 15]
 [ 6  7  8  9 10]
 [ 1  2  3  4  5]]
```

A continuació llistem tot de funcions que es poden aplicar a les matrius que permeten extreure'n informació:

[38]:
```
np.amin(a) # retorna el nombre més petit
```

[38]:
```
1
```

[39]:
```
np.amin(a,0) # retorna un `array` amb el mínim de cada columna
```

[39]:
```
array([1, 2, 3, 4, 5])
```

[40]:
```
np.amin(a,1) # reotrna un `array`amb el mínim de cada fila
```

[40]:
```
array([ 1,  6, 11, 16])
```

[41]:
```
np.amax(a) # retorna el màxima de tota la matriu
```

[41]:
```
20
```

[42]:
```
np.amax(a,0) # retorna un `array` amb el màxim de cada columna
```

[42]:
```
array([16, 17, 18, 19, 20])
```

[43]:
```
np.amax(a,1) # retorna un `array` amb el màxim de cada fila
```

[43]:
```
array([ 5, 10, 15, 20])
```

Exercici La funció `ptp` té la mateixa sintaxis que `amin` o `amax` i el que retorna és la diferència entre el valor màxim i el mínim (el que podríem anomenar *recorregut*).
Comproveu-ne el funcionament amb la matriu `a`.
Definiu una funció nova anomenada `recorregut` (utilitzant `amini amax`) que faci el mateix que `ptp`.

Acabem aquest apartat dedicat a les matrius veient algunes funcions d'estadística descriptiva:

[44]:
```
a=np.random.normal(0,1,10000) # Generem 10000 nombres aleatoris
# d'una normal amb mitjana 0 i desviació típica 1
```

[45]:
```
a.shape=(100,100) # Ho convertim a 100 mostres amb 100 valors
```

[46]:
```
print(np.percentile(a,50))
```

```
-0.005054341379625719
```

[47]:
```
print(np.percentile(a,5))
```

```
-1.64486862574702
```

[48]:
```
print(np.percentile(a,50,axis=1)) # Percentil 50 de cada fila
```

```
[ 0.13430614 -0.08809469 -0.26189557  0.096536   -0.02152839 -0.08772514
 -0.05975246 -0.05686017  0.00702203  0.00447882 -0.01206839 -0.01412457
 -0.10013364 -0.21199902  0.04692229 -0.04026659  0.09035181  0.13654066
 -0.04667052 -0.22369638  0.01486242  0.0163459   0.06421475 -0.08425408
  0.07354118  0.03487513  0.08475696  0.16673465  0.01780605 -0.01553925
 -0.05382863  0.05420638 -0.02427463  0.05031312  0.27587797 -0.1677327
 -0.00505485 -0.23402658  0.17763964 -0.00317304  0.00732304 -0.1414667
 -0.21979551 -0.06295166  0.1959206   0.11622817 -0.13424581 -0.26609823
  0.06567321 -0.00658917  0.17696384 -0.05459092 -0.00576853 -0.08119564
 -0.10769587 -0.08324314 -0.01469532  0.03153944  0.27259538 -0.06007089
 -0.01800183 -0.3238337  -0.08048431  0.03113841  0.07714582 -0.16870498
  0.09880637 -0.10188146  0.09018169  0.14713457 -0.04338948  0.14186182
  0.09395672 -0.1035583   0.07179954  0.21640282 -0.00393068  0.04494614
  0.16715669 -0.05964776 -0.05491205 -0.14103191  0.13320687  0.13846309
 -0.1739314  -0.16741447 -0.32528972 -0.03336742 -0.04436832 -0.00347908
 -0.21871271  0.18188701 -0.02918743  0.52891011  0.0603868   0.01591226
  0.10325308  0.05759015  0.12852359 -0.07782711]
```

[49]:
```
print(np.mean(a)) # La mitjana
```

```
-0.005444152643115992
```

[50]:
```
print(np.mean(a, axis=0)) # La mitjana de cada columna
```

```
[-0.15522525 -0.17890911  0.0171091  -0.07555172 -0.18831605 -0.06416646
 -0.08708167  0.0787124   0.03086632 -0.07369119  0.02411815  0.05548587
 -0.0016122  -0.01087157  0.03894504  0.09839658  0.03804911 -0.13061186
 -0.07779573 -0.06288938 -0.0133512   0.03644658  0.16141909 -0.02319143
  0.06461944  0.02544173 -0.0662912   0.03627372  0.12019362  0.07390809
  0.07399393  0.01441274  0.2358607  -0.01074487 -0.03693542 -0.02173263
  0.07148767 -0.18277399  0.10072364 -0.03255462  0.05696017  0.20615072
 -0.03733371 -0.02815253 -0.11001265  0.09168294  0.02368538 -0.03924675
  0.10962186 -0.0338098  -0.00190641  0.0515878   0.10356273 -0.04816802
  0.09011097 -0.07176007  0.15521708 -0.08504865 -0.00451971  0.04263668
  0.04874715  0.11377232 -0.09785247 -0.24637441  0.10270966 -0.0617074
 -0.0931345  -0.09708969 -0.03331991 -0.01619904 -0.01971239 -0.11626809
 -0.14654441  0.04121371 -0.25396062 -0.06205465 -0.05230175 -0.02202663
  0.13650358 -0.14669164  0.05871505  0.068562    0.02587224  0.15280697
 -0.11931208 -0.01947568 -0.21396601  0.03467072  0.00260841 -0.10123858
  0.08823031 -0.01846775 -0.08050321  0.02804344 -0.04001226  0.07294468
  0.08600759 -0.01994878  0.07551507  0.02339977]
```

[51]:
```
print(np.std(a))
```

```
0.9972235915604041
```

[52]:
```
print(np.var(a))
```

```
0.9944548915646315
```

[53]:
```
print(np.var(a)-np.std(a)**2)
```

```
-1.1102230246251565e-16
```

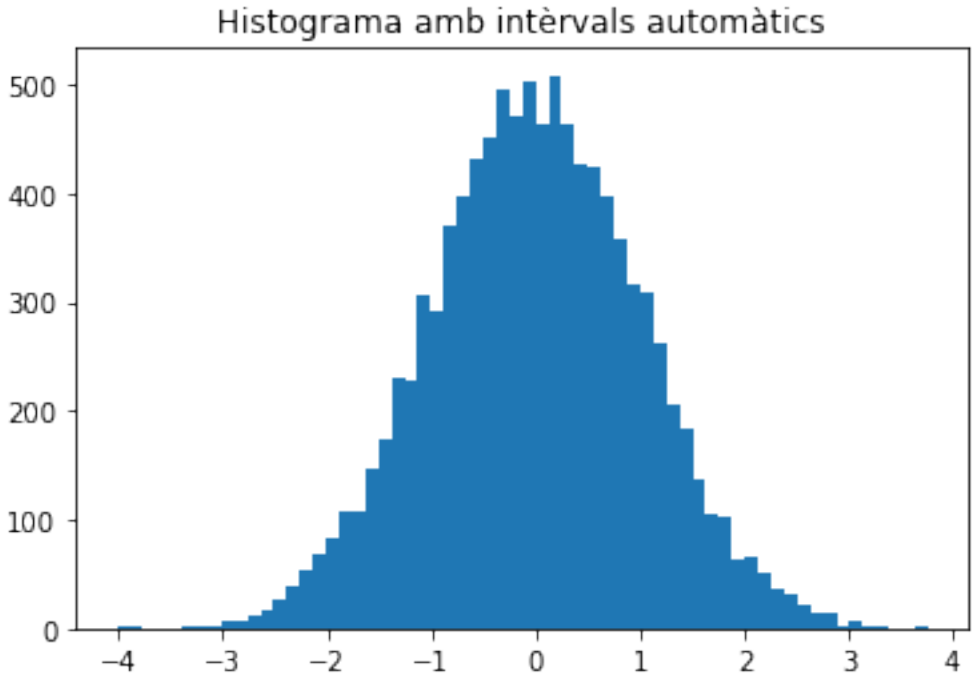
1.2 Gràfics amb `matplotlib.pyplot`.

La biblioteca `matplotlib.pyplot` permet obtenir gràfics de les dades o funcions:

[54]:
```
import numpy as np
import matplotlib.pyplot as plt
```

En general, es va construint el gràfic pas a pas: es crea, s'afegeixen títols, eixos, ... i finalment es mostra amb la instrucció `show` o bé es guarda amb la instrucció `savefig`.

[55]:
```
a=np.random.normal(0,1,10000)
plt.hist(a,bins='auto')
plt.title('Histograma amb intèrvals automàtics')
plt.show()
```



1.2.1 Exemple

Volem representar gràficament les dades d'estudiants a universitats públiques catalanes al curs 2018/19 (font: Idescat) corresponents a la taula següent:

Universitat	Dones	Homes
UB	27523	16919
UAB	18529	12397
UPC	5891	16506
UPF	8900	6901
UdG	7682	5775
UdL	5077	3678
URV	7022	4923

Comencem definint llistes i matrius:

[56]:
```
universitats=['UB','UAB','UPC','UPF','UdG','Udl','URV']
```

[57]:
```
estudiants=np.
↪array([[27523,16919],[18529,12397],[5891,16506],[8900,6901],[7682,5775],␣
↪[5077,3678],[7022,4923]])
```

[58]:
```
estudiants.transpose()
```

[58]:
```
array([[27523, 18529,  5891,  8900,  7682,  5077,  7022],
       [16919, 12397, 16506,  6901,  5775,  3678,  4923]])
```

Calculem la suma de dones i homes a una tercera columna:

[59]:
```
estudiants=np.c_[estudiants,estudiants[:,0]+estudiants[:,1]]
```

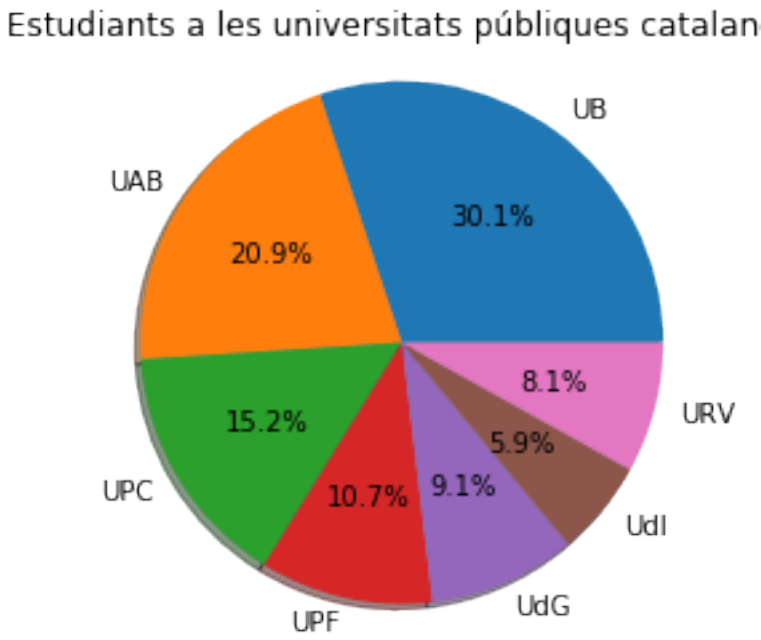
[60]:
```
print(estudiants)
```

```
[[27523 16919 44442]
 [18529 12397 30926]
 [ 5891 16506 22397]
 [ 8900  6901 15801]
 [ 7682  5775 13457]
 [ 5077  3678  8755]
 [ 7022  4923 11945]]
```

[61]:
```
estudiants[:,2]
```

[61]:
```
array([44442, 30926, 22397, 15801, 13457,  8755, 11945])
```

[62]:
```
plt.pie(estudiants[:,2],labels=universitats,autopct='%1.1f%%',shadow=True)
plt.title("Estudiants a les universitats públiques catalanes")
plt.axis('equal')
plt.show()
```



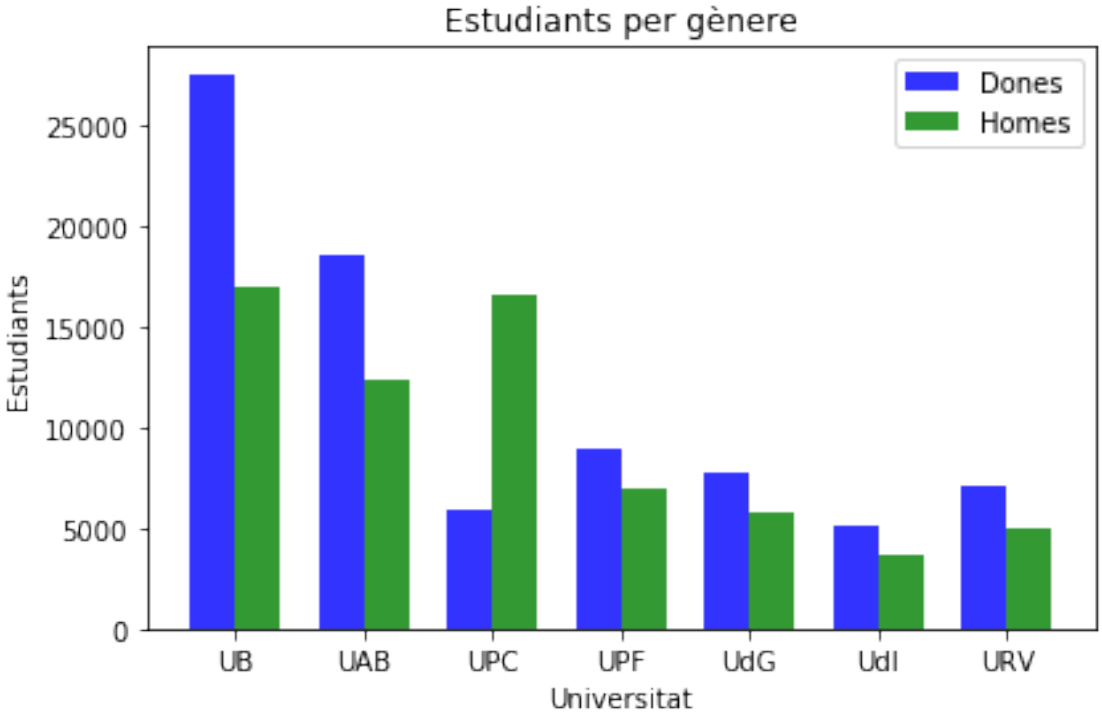
[63]:
```
fig, ax = plt.subplots()
index = np.arange(len(estudiants[:,0]))
amplada = 0.35
opacitat = 0.8

dones = plt.bar(index, estudiants[:,0], amplada,
                alpha=opacitat,
                color='b',
                label='Dones')

homes = plt.bar(index+amplada, estudiants[:,1], amplada,
                alpha=opacitat,
                color='g',
                label='Homes')

plt.xlabel('Universitat')
plt.ylabel('Estudiants')
plt.title('Estudiants per gènere')
plt.xticks(index + amplada/2, universitats)
plt.legend()
```

```
plt.tight_layout()
plt.show()
```



1.2.2 Exercici:

Feu la gràfica anterior, però amb les columnes apilades enlloc d'una al costat de l'altra.

1.2.3 Exercici:

Feu una gràfica de les mateixes dades, però amb dues columnes: una per cada gènere i de colors diferents segons correspongui a cada universitat.