td1a_cenonce_session_12_plot

June 19, 2019

1 1A.data - Visualisation des données

Les tableaux et les graphes sont deux outils incontournables des statisticiens. Petite revue des graphes.

[1]:
```
%matplotlib inline
```

Cette instruction fait apparaître les graphes dans le notebook. Si ce n'est pas le cas, il faut la réexécuer. Les deux lignes suivantes permettent de vérifier où matplotlib a prévu d'afficher ses résultats. Pour un notebook, cela doit être 'nbAgg' ou 'module://ipykernel.pylab.backend_inline'.

[2]:
```
import matplotlib
matplotlib.get_backend()
```

[2]:
```
'module://ipykernel.pylab.backend_inline'
```

[3]:
```
import matplotlib.pyplot as plt
import matplotlib
matplotlib.get_backend()
```

[3]:
```
'module://ipykernel.pylab.backend_inline'
```

[4]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[4]:
```
<IPython.core.display.HTML object>
```

1.1 Matplotlib, pandas

1.1.1 Récupération des données

On récupère les données disponibles sur le site de l'INSEE : Naissance, décès, mariages 2012. Il s'agit de récupérer la liste des mariages de l'année 2012. On souhaite représenter le graphe du nombre de mariages en fonction de l'écart entre les mariés.

[5]:
```
from urllib.error import URLError
import pyensae
from pyensae.datasource import dBase2df, DownloadDataException
files = ["etatcivil2012_nais2012_dbase.zip",
         "etatcivil2012_dec2012_dbase.zip",
         "etatcivil2012_mar2012_dbase.zip" ]

try:
    pyensae.download_data(files[-1],
                          website='http://telechargement.insee.fr/fichiersdetail/
 ↪etatcivil2012/dbase/')
except (DownloadDataException, URLError, TimeoutError):
    # backup plan
```

```
    pyensae.download_data(files[-1], website="xd")

df = dBase2df("mar2012.dbf")
df.shape, df.columns
```

[5]:
```
((246123, 16),
 Index(['ANAISH', 'DEPNAISH', 'INDNATH', 'ETAMATH', 'ANAISF', 'DEPNAISF',
        'INDNATF', 'ETAMATF', 'AMAR', 'MMAR', 'JSEMAINE', 'DEPMAR', 'DEPDOM',
        'TUDOM', 'TUCOM', 'NBENFCOM'],
       dtype='object'))
```

[6]:
```
df.head()
```

[6]:

	ANAISH	DEPNAISH	INDNATH	ETAMATH	ANAISF	DEPNAISF	INDNATF	ETAMATF	AMAR	MMAR
0	1982	75	1	1	1984	99	2	1	2012	01
1	1956	69	2	4	1969	99	2	4	2012	01
2	1982	99	2	1	1992	99	1	1	2012	01
3	1985	99	2	1	1987	84	1	1	2012	01
4	1968	99	2	1	1963	99	2	1	2012	01

	JSEMAINE	DEPMAR	DEPDOM	TUDOM	TUCOM	NBENFCOM
0	1	29	99	9		N
1	3	75	99	9		N
2	5	34	99	9		N
3	4	13	99	9		N
4	6	26	99	9		N

On récupère de la même manière la signification des variables :

[7]:
```
from pyensae.datasource import dBase2df
vardf = dBase2df("varlist_mariages.dbf")
vardf.shape, vardf.columns
```

[7]:
```
((16, 4), Index(['VARIABLE', 'LIBELLE', 'TYPE', 'LONGUEUR'], dtype='object'))
```

[8]:
```
vardf
```

[8]:

	VARIABLE	LIBELLE	TYPE	LONGUEUR
0	AMAR	Année du mariage	CHAR	4
1	ANAISF	Année de naissance de l'épouse	CHAR	
2	ANAISH	Année de naissance de l'époux	CHAR	
3	DEPDOMã	Département de domicile après le mariage	CHAR	
4	DEPMAR	Département de mariage	CHAR	
5	DEPNAISF	Département de naissance de l'épouse	CHAR	
6	DEPNAISH	Département de naissance de l'époux	CHAR	
7	ETAMATF	État matrimonial antérieur de l'épouse	CHAR	
8	ETAMATH	État matrimonial antérieur de l'époux	CHAR	
9	INDNATF	Indicateur de nationalité de l'épouse	CHAR	
10	INDNATH	Indicateur de nationalité de l'époux	CHAR	
11	JSEMAINE	Jour du mariage dans la semaine	CHAR	
12	MMAR	Mois du mariage	CHAR	
13	NBENFCOM	Enfants en commun avant le mariage	CHAR	
14	TUCOM	Tranche de commune du lieu de domicile des époux	CHAR	
15	TUDOM	Tranche d'unité urbaine du lieu de domicile de...	CHAR	

```
1           4
2           4
3           3
4           3
5           3
6           3
7           1
8           1
9           1
10          1
11          1
12          2
13          1
14          1
15          1
```

1.1.2 Exercice 1 : écart entre les mariés

1. En ajoutant une colonne et en utilisant l'opération group by, on veut obtenir la distribution du nombre de mariages en fonction de l'écart entre les mariés. Au besoin, on changera le type d'une colone ou deux.
2. On veut tracer un nuage de points avec en abscisse l'âge du mari, en ordonnée, l'âge de la femme. Il faudra peut-être jeter un coup d'oeil sur la documentation de la méthode plot.

[9]:
```
# df["colonne"] = df.apply (lambda r:  int(r["colonne"]), axis=1)  # pour changer de type
# df["difference"] = ...
```

1.1.3 Exercice 2 : graphe de la distribution avec pandas

Le module pandas propose un panel de graphiques standard faciles à obtenir. On souhaite représenter la distribution sous forme d'histogramme. A vous de choisir le meilleure graphique depuis la page Visualization.

[10]:
```
# df.plot(...)
```

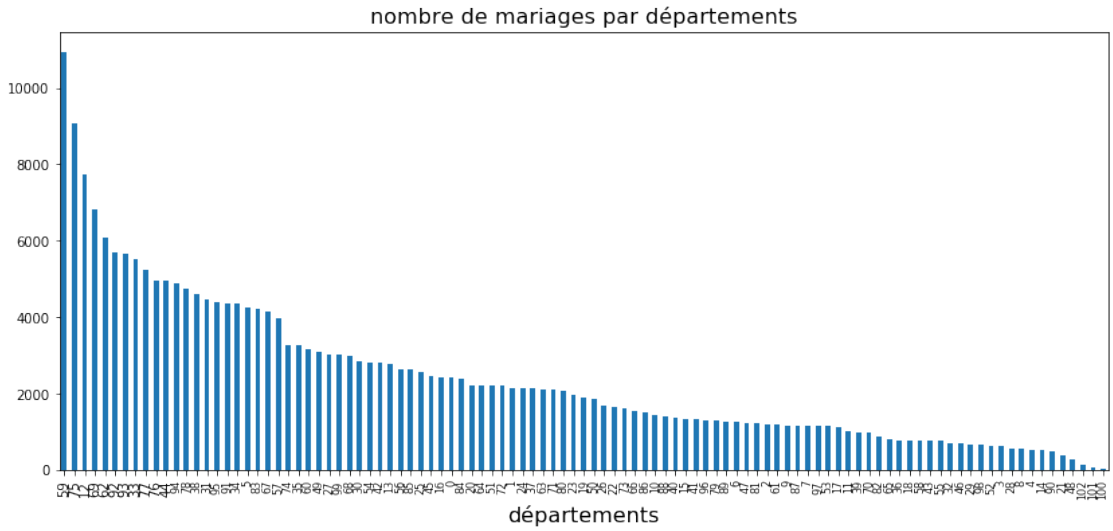
1.1.4 matplotlib

matplotlib est le module qu'utilise pandas. Ainsi, la méthode plot retourne un objet de type Axes qu'on peut modifier par la suite via les méthodes suivantes. On peut ajouter un titre avec set_title ou ajouter une grille avec grid. On peut également superposer deux courbes sur le même graphique, ou changer de taille de caractères. Le code suivant trace le nombre de mariages par département.

[11]:
```
df["nb"] = 1
dep = df[["DEPMAR","nb"]].groupby("DEPMAR", as_index=False).sum().sort_values("nb",ascending=False)
ax = dep.plot(kind = "bar", figsize=(14,6))
ax.set_xlabel("départements", fontsize=16)
ax.set_title("nombre de mariages par départements", fontsize=16)
ax.legend().set_visible(False)  # on supprime la légende

# on change la taille de police de certains labels
for i,tick in enumerate(ax.xaxis.get_major_ticks()):
    if i > 10 :
```

```
        tick.label.set_fontsize(8)
```



Quand on ne sait pas, le plus simple est d'utiliser un moteur de recherche avec un requête du type : `matplotlib + requête`. Pour créer un graphique, le plus courant est de choisir le graphique le plus ressemblant d'une gallerie de graphes puis de l'adapter à vos données.

1.1.5 Exercice 3 : distribution des mariages par jour

On veut obtenir un graphe qui contient l'histogramme de la distribution du nombre de mariages par jour de la semaine et d'ajouter une seconde courbe correspond avec un second axe à la répartition cumulée.

[12]:
```
```

1.2 Réseaux, graphes

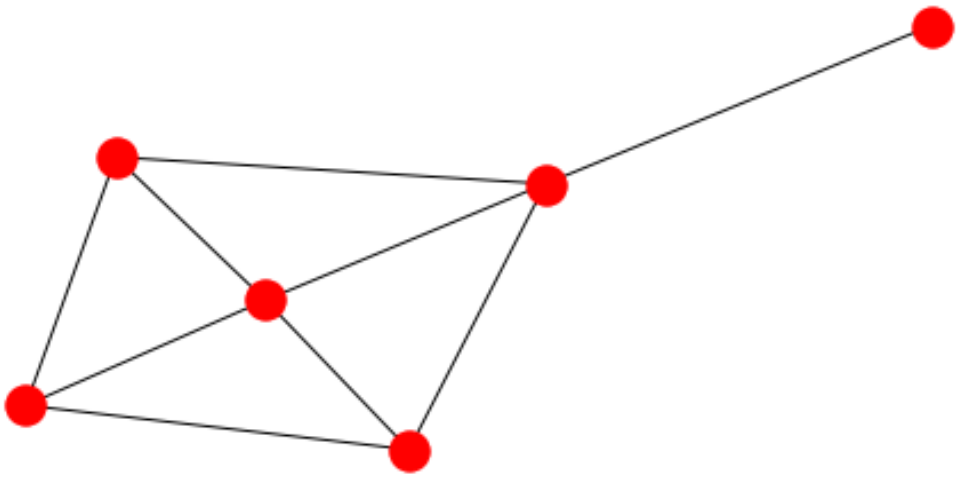
1.2.1 networkx

Le module networkx permet de représenter un réseau ou un graphe de petite taille (< 500 noeuds). Un graphe est défini par un ensemble de noeuds (ou *vertex* en anglais) reliés par des arcs (ou *edge* en anglais). La gallerie vous donnera une idée de ce que le module est capable de faire.

[13]:
```
import random
import networkx as nx
G=nx.Graph()
for i in range(15) :
    G.add_edge ( random.randint(0,5),  random.randint(0,5) )

import matplotlib.pyplot as plt
f, ax = plt.subplots(figsize=(8,4))
nx.draw(G, ax = ax)
```

```
c:\python370_x64\lib\site-packages\networkx\drawing\nx_pylab.py:611:
MatplotlibDeprecationWarning: isinstance(..., numbers.Number)
  if cb.is_numlike(alpha):
```



1.2.2 Graphviz

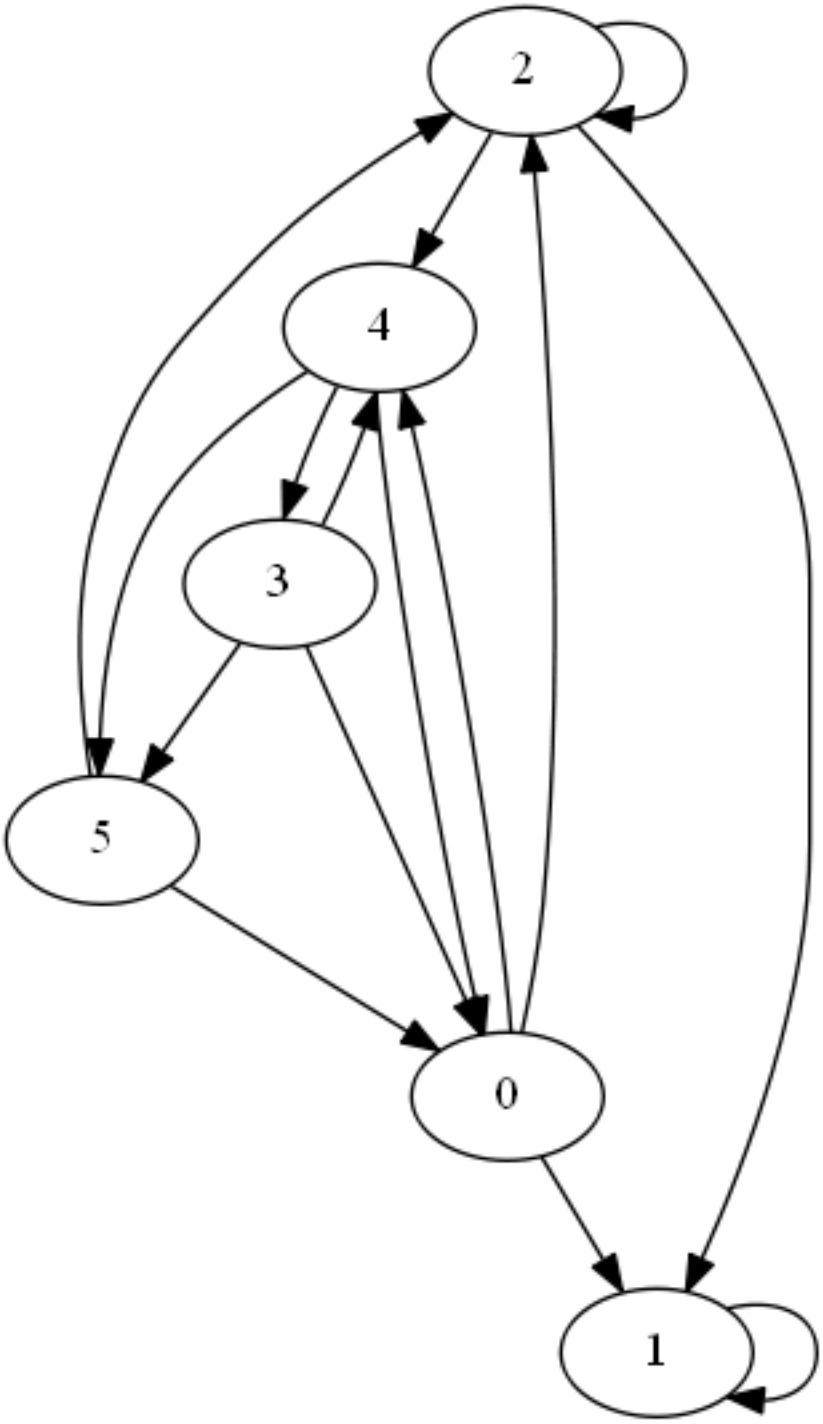
Graphviz est un outil développé depuis plusieurs années déjà qui permet de réprésenter des graphes plus conséquents (> 500 noeuds). Il propose un choix plus riche de graphes : gallerie. Il est utilisable via le module graphviz. Son installation requiert l'installation de l'outil Graphviz qui n'est pas inclus. La différence entre les deux modules tient dans l'algorithme utilisé pour assigner des coordonnées à chaque noeud du graphe de façon à ce que ses arcs se croisent le moins possibles. Au delà d'une certaine taille, le dessin de graphe n'est plus lisible et nécessite quelques tatônnements. Cela peut passer par une clusterisation du graphe (voir la méthode Louvain) de façon à colorer certains noeuds proches voire à les regrouper.

[14]:
```
import random, os
from graphviz import Digraph
from IPython.display import Image
from pyquickhelper.helpgen import find_graphviz_dot
bin = os.path.dirname(find_graphviz_dot())
if bin not in os.environ["PATH"]:
    os.environ["PATH"] = os.environ["PATH"] + ";" + bin

dot = Digraph(comment='random graph', format="png")
for i in range(15) :
    dot.edge ( str(random.randint(0,5)),  str(random.randint(0,5)) )

img = dot.render('t_random_graph.gv')
Image(img)
```

[14]:



1.2.3 Exercice 4 : dessin d'un graphe avec networkx

On construit un graphe aléatoire, ses 20 arcs sont obtenus en tirant 20 fois deux nombres entiers entre 1 et 10. Chaque arc doit avoir une épaisseur aléatoire. On regardera les fonctions spring_layout, draw_networkx_nodes, draw_networkx_edges. La gallerie peut aider aussi.

[15]:

[16]: