1 TD 9 : Graphiques en javascript, d3.js

(correction page ??)

Abordé lors de cette séance	
programmation	d3.js, javascript
algorithme	graphique, animation

Les parties A,B,D sont indépendantes. Les parties B et C sont liées.

Première demi-heure : Représenter des données géographiques avec `matplotlib` et OpenStreetMap[1] et javascript[2]

On utilise toujours les données velib. On cherche à représenter graphiquement ces données.

```
from pyensae import download_data
import pandas

download_data("td9_data.zip", website = 'xd')
file1 = "td9_full.txt"
tbl = pandas.read_csv (file1, sep = "\t")
```

Les données contiennent deux fichiers `td9_full.txt` et `td9_sub.txt`. Ils contiennent tous deux les vélos et places disponibles toutes les cinq minutes pour toutes les stations velib pour la journée de $2013-09-10$. Le fichier `td9_sub.txt` ne contient que quelques stations et permet de tester son programme sur un petit jeu. On crée un graphe XY (*scatter plot*) avec la longitude et la latitude des stations (voir figure 1).

```
import pylab
from pandas.tools.plotting import scatter_plot

gr = tbl.groupby(['lat','lng'], as_index = False).agg(lambda x: len(x))
scatter_plot(gr, "lat", "lng" )
pylab.show()
```

Le résultat est satisfaisant excepté qu'il est difficile saisir où se trouve trouve exactement chaque station par rapport aux rues de Paris. Introduire une carte n'est pas toujours chose facile en *Python*. On va plutôt choisir le *javascript* et OpenStreetMap [3]. Tout d'abord, on va copier coller l'exemple suivant dans un fichier HTML (`td9.html` par exemple) puis l'ouvrir avec un navigateur. Une carte devrait apparaître.

```
<html><body>
  <div id="mapdiv"></div>
  <script src="http://www.openlayers.org/api/OpenLayers.js"></script>
  <script>
    map = new OpenLayers.Map("mapdiv");
```

1. http://www.openstreetmap.org/#map=19/48.82435/2.30318
2. http://fr.wikipedia.org/wiki/JavaScript
3. http://www.openstreetmap.org/

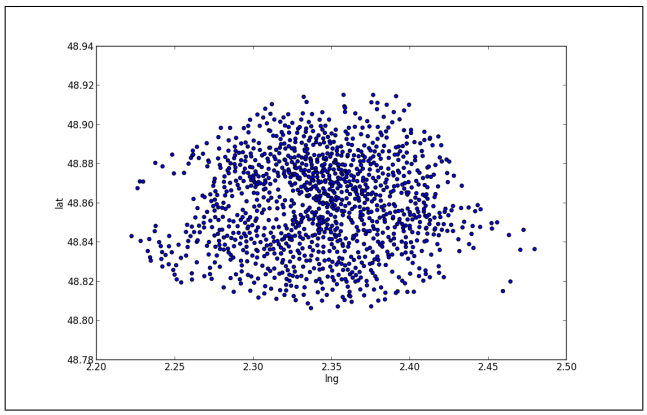

Figure 1 : *Positions des stations velib sur Paris.*

```
    map.addLayer(new OpenLayers.Layer.OSM());
    var proj =  new OpenLayers.Projection("EPSG:4326");

    var zoom=16;

    var markers = new OpenLayers.Layer.Markers( "Markers" );
    map.addLayer(markers);

    var lonLat = new OpenLayers.LonLat( -0.1279688 ,51.5077286 ).transform(proj, map.getProjectionObject() );
    markers.addMarker(new OpenLayers.Marker(lonLat));

    var lonLat2 = new OpenLayers.LonLat( -0.1279688 ,51.5067286 ).transform(proj, map.getProjectionObject() );
    markers.addMarker(new OpenLayers.Marker(lonLat2));

    map.setCenter (lonLat, zoom);
  </script>
</body></html>
```

Exercice 1

L'exemple HTML utilise du javascript. Sa syntaxe est différente mais la logique est semblable. Pour ajouter une position sur la carte, il suffit d'ajouter deux lignes. Et pour ajouter autant de positions que de stations velib, il suffit de se servir de Python et du sketch de programme qui suit (et vous devriez obtenir quelque chose de semblable à la figure 2) [4].

```
# ... ajouter des choses ici
lines = [ ]
for i,row in enumerate(gr.values) :
    y = lat = row[1]
    x = lng = row[0]
    # utiliser i,x,y pour construire la ligne de javascript correspondant à la station i
    lines.append(line)

text = "\n".join( lines )
html = html.replace("__VELIB__", text)
with open("velib.html", "w") as f : f.write(html)
```

Remarque 0.1 : confidentialité

A aucun moment, les données d'OpenStreetMap n'ont été téléchargées. Le script utilisé fait éga-

4. voir aussi http://www.xavierdupre.fr/blog/2013-09-26_nojs.html

lement référence à un code javascript [5] dont ne sait pas tout ce qu'il fait. Cela veut dire que vos données sont potentiellement envoyées à un site extérieur qui peut ou non décider de s'y intéresser. Il évident aussi que cet exemple ne fonctionnera pas sans connexion internet.

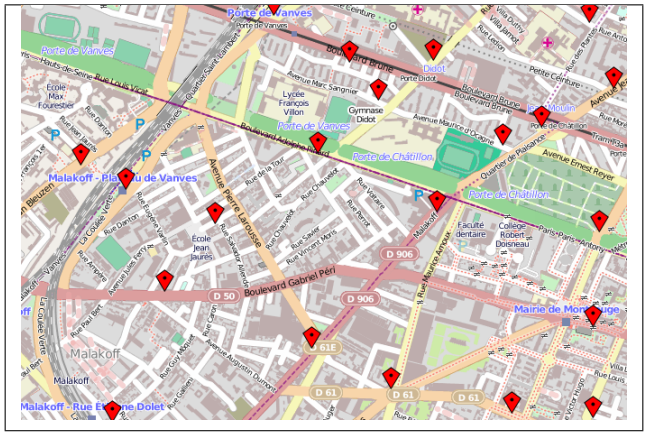

Figure 2 : *Positions des stations velib sur Paris en utilisant OpenStreetMap.*

Astuce

Pour lancer directement le navigateur depuis Python, vous pouvez utiliser le code suivant :

```
import webbrowser
webbrowser.open(<url>)
```

Seconde demi-heure : d3.js[6]

d3.js est l'outil à utiliser pour réaliser un graphique mobile ou qu'on souhaite intégrer à un site internet. Pour s'en servir, le plus simple est de regarder les galleries d'exemples [7] [8] [9], de choisir celui qu'on veut, de copier coller le code puis de remplacer les données de l'exemple par les siennes. En pratique, c'est vrai pour les graphiques les plus simples. Pour les plus complexes, cela va plus vite si on connaît le javascript.

Pour ce TD, on va dessiner un graphe : http://bl.ocks.org/mbostock/2706022. On souhaite représenter les connexions entre les différents personnages de la série *L-World* [10]. Pour que cela marche, il faut transformer des données d'un format à un autre.

```
agatha,frank
agatha,toni
alice,andrew
alice,april
alice,bette
...
```

5. http://www.openlayers.org/api/OpenLayers.js
6. http://d3js.org/
7. https://github.com/mbostock/d3/wiki/Gallery
8. https://github.com/mbostock/d3/wiki/Gallery
9. http://techslides.com/over-1000-d3-js-examples-and-demos/
10. http://en.wikipedia.org/wiki/The_L_Word

Autre format (le champ type n'eest pas important) :

```
var links = [
  {source: "Microsoft", target: "Amazon", type: "licensing"},
  {source: "Microsoft", target: "HTC", type: "licensing"},
  {source: "Samsung", target: "Apple", type: "suit"},
        // ...
        ] ;
```

Ce qu'on peut faire avec le programme suivant :

```
with open("td9_graph_lworld.txt", "r") as f :
    lines = f.readlines()

links =[]
for line in lines :
    spl = line.strip("\n\r ").split(",")
    if len(spl) == 2 :
        l = { "source":spl[0],"target":spl[1],"type":"-"}
        links.append(l)
        l = { "source":spl[1],"target":spl[0],"type":"-"}
        links.append(l)

with open("td9_graph_lworld.js", "w") as f :
    f.write ("var links = [\n")
    for l in links :
        f.write("{")
        f.write( ",".join ( [ "{0}:'{1}'".format (k,v) for k,v in l.items() ] ) )
        f.write("},\n")
    f.write("\n];\n")
```

Exercice 2 :

Il s'agit d'assembler les différents éléments pour construire le graphe. Les données ainsi que la page HTML peuvent être récupérées avec le programme suivant :

```
import pyensae
pyensae.download_data("td9_graph_lworld.zip", website = 'xd')
```

Troisième demi-heure : Un graphique avec des événements

Le code ayant servi d'exemple [11] contient deux fonctions introduisant des animations qui doivent se produire si le curseur de la souris passe au-dessus de la zone d'un nœud. Il faudra les ajouter à votre page HTML.

Exercice 3 :

Ensuite, à partir de la page Internet suivante http://stackoverflow.com/questions/14929212/changing-text-attribute-based-upon-event-handler-in-d3-js, il faudra changer le texte du nœud. Une astuce :

11. http://bl.ocks.org/mbostock/2706022

```
// bien placé, le code suivant modifie le texte affiché et sa taille
d3.select(this)
        .select("text")
                        .style("font-size", "10px")
                        .text(function(d,i){return d.name + "**" ;});
```

Quatrième demi-heure : Zoomer .

Pour cela, il vous faudra lire le blog : http://www.xavierdupre.fr/blog/2013-09-29_nojs.html puis essayer de répliquer ce qui y est décrit.

http ://mbostock.github.io/d3/tutorial/circle.html

Pour aller plus loin ou pour ceux qui ont fini plus tôt .

Si vous avez le courage, vous pouvez aussi regarder des zoom du style loupe : http://bost.ocks.org/mike/fisheye/.

Remarques .