1 TD 7 : Calcul matriciel, pandas, numpy

(correction page ??)

Abordé lors de cette séance	
programmation	calcul matriciel, pandas
algorithme	manipulation d'une série temporelle

Durant cette séance, on s'intéressera à la station vélib proche de la porte de Vanves : on dispose pour cette station du nombre de vélos disponibles, du nombre de places vides pour les deux mois d'été 2013 à raison d'une donnée toutes les cinq minutes. Les données sont accessibles via un fichier texte. Vous pouvez les télécharger en exécutant les lignes suivantes :

```
import pyensae
file = pyensae.download_data("velib_vanves.zip", website = "xd")
print (file)   # affiche ['velib_vanves.txt']
```

Le fichier récupéré a l'aspect suivant :

```
address        available_bike_stands        available_bikes        banking        bike_stands        bonus        contract_name
112 RUE VERCINGETORIX - 75014 PARIS        65        2        0        67        0        Paris        15/07/2013 15:00
112 RUE VERCINGETORIX - 75014 PARIS        65        2        0        67        0        Paris        15/07/2013 15:05
112 RUE VERCINGETORIX - 75014 PARIS        66        1        0        67        0        Paris        15/07/2013 15:10
112 RUE VERCINGETORIX - 75014 PARIS        66        1        0        67        0        Paris        15/07/2013 15:15
```

Le module `pyensae` est accessible à l'adresse suivante : http://www.xavierdupre.fr/site2013/index_code.html [1].

Les périodes plates (figure 1) de quelques jours sont dûes à un bug lors de la récupération des données. Il y en eut d'autres de quelques dizaines de minutes correspondant à un redémarrage de la machine collectant des données. Les parties suivantes s'appuient sur le module `pandas` [2].

Première demi-heure : Récupérer les données .

Les données sont accessibles via un fichier texte `velib_vanves_2013_07_08.txt`. Il faut pouvoir les manipuler depuis un programme *Python* ce qu'on va faire à l'aide du module `pandas`. Voici quelques exemples :

– Lire une table depuis un fichier texte [3] :

```
import pandas
df = pandas.read_table(file, header = False, sep = "\t", decimal = ",")
```

Il faudra considérer le paramètre `parse_dates` pour convertir les dates dans un format numérique.

1. Le module est également référencé sur pipy : https://pypi.python.org/pypi/pyensae/. La documentation est disponible ici : http://www.xavierdupre.fr/app/pyensae/helpsphinx/index.html.
2. disponible ici : http://pandas.pydata.org/, voir également http://www.xavierdupre.fr/blog/2013-09-14_nojs.html.
3. `read_table` : http://pandas.pydata.org/pandas-docs/dev/generated/pandas.io.parsers.read_table.html

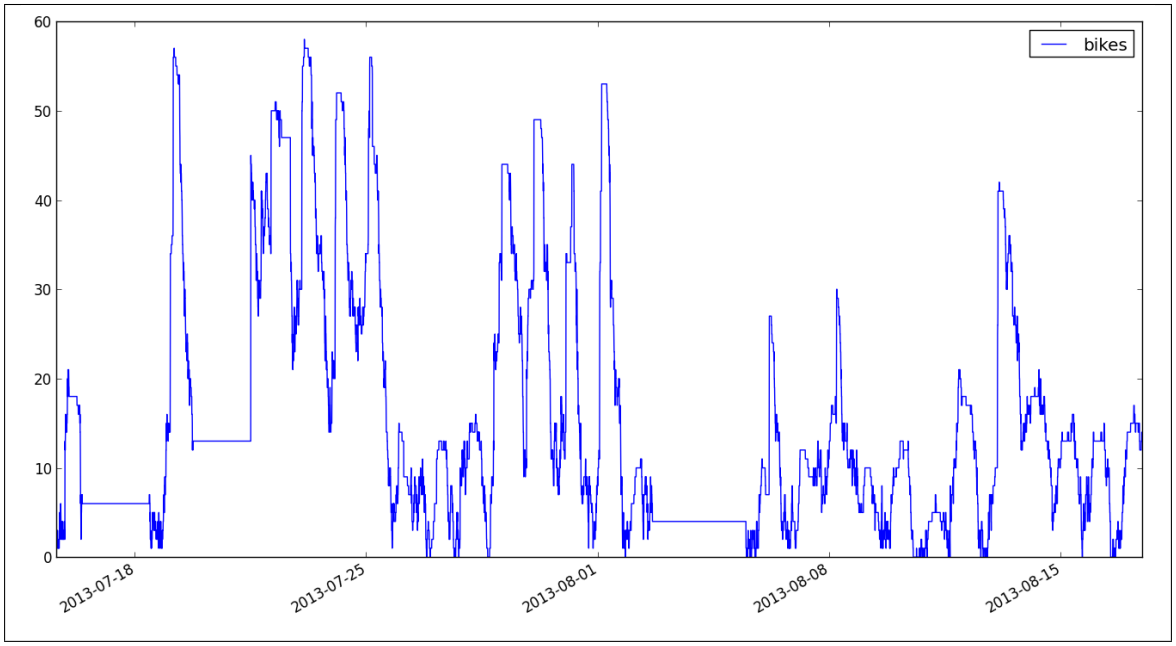

FIGURE 1 : *Nombre de vélos disponibles à la station Vélib, porte de Vanves à Paris*

– Accéder ou modifier des données de la table [4] :

```
une_colonne = df.ix[:,"available_bikes"])
```

– Obtenir le minimum ou maximum :

```
mx_col = df["available_bikes"].max()      # maximum d'une colonne
mx_col = df.ix[:,"available_bikes"].max() # même maximum, autre écriture
mx = df.max()                             # maximum sur toutes les colonnes
```

– Obtenir un résumé des variables (min,max, ...) :

```
df.describe()
```

– Obtenir le nombre de lignes d'une table :

```
len(df)
```

– Construire un sous-ensemble de lignes vérifiant une condition :

```
zero =  df [ df["available_bikes"]==0 ]
```

– Ajouter deux colonnes :

```
add = df["available_bikes"] + df["available_bike_stands"]
```

Exercice :

A partir de là, on cherche à extraire les informations suivantes :
– la première et la dernière dates,
– le nombre maximum de vélos et de places vides,

4. `ix` : http://pandas.pydata.org/pandas-docs/dev/indexing.html#advanced-indexing-with-ix

– la proportion du temps durant lequel la station n'a pas de vélos disponibles,
– la proportion du temps durant lequel la station n'a pas de places disponibles,
– le nombre d'emplacements de vélos à cette station.
Chaque information peut être obtenue à l'aide d'une ou deux lignes écrites en *Python*.

Seconde demi-heure : Visualisation des données

On veut maintenant dessiner la série temporelle. On utilise le code suivant :

```
df = df.set_index("last_update")  # on identifie chaque ligne avec une date
import matplotlib.pyplot as plt   # module matplotblig
ts = df.ix[:,"available_bikes"]   # on extrait la série à dessiner
ts.plot()                         # on dessine
plt.show()                        # on affiche le dessin
```

Deux questions à propos de ce graphe :

1. Que pensez-vous de certaines valeurs aberrantes ?
2. Vos observations remettent-elles en question vos résultats obtenus précédemment ?

On souhaite visualiser les données de façon légèrement différente :

– Regarder la page : http://matplotlib.org/gallery.html.
– Choisir le graphe qui vous convient [5].
– Utiliser l'exemple pour représenter les données.

Troisième demi-heure : Visualisation des semaines superposées

La courbe paraît périodique. Pour s'en convaincre, on veut tracer le même graphe que précédemment en superposant les semaines. La figure 2 illustre ce qu'on veut faire. Tout d'abord on va ne garder que les informations qui nous intéressent :

```
df2 = df.ix[:, ["last_update","available_bikes"]]
```

Il faut pouvoir apparier les lignes correspondant à des dates différentes, par exemple :
– `2013-07-15 15:10:00` (lundi)
– `2013-14-15 15:10:00` (lundi)
– `2013-21-15 15:10:00` (lundi)
– ...

Exercice

Dans un premier temps, il faut ajouter une colonne à la matrice `df2` qui contient le jour de la semaine et l'heure : `2013-07-15 15:10:00` devient `lundi15 : 10 : 00` [6]. Quelques instructions qui pourraient vous inspirer :

```
import datetime
d = datetime.datetime.now()
```

5. par exemple http://matplotlib.org/examples/api/date_index_formatter.html
6. Le jour de la semaine peut-être une chaîne de caractère ou un nombre.

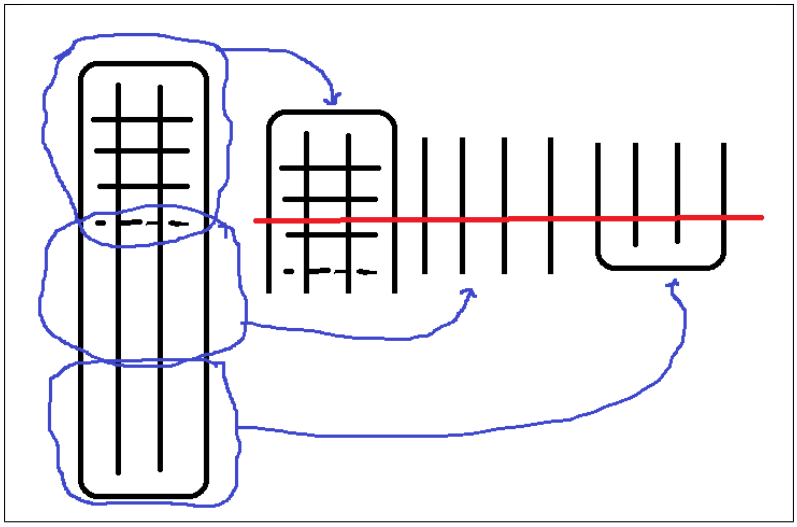

Figure 2 : *Manipulation des données : la ligne rouge désigne les données correspondant à la même heure et au même jour de la semaine pour plusieurs semaines.*

```
print (d.weekday())
print (d.strftime("%H:%M"))
print (d.isocalendar())
```

Pour ajouter une colonne dans un objet `DataFrame`, vous pouvez utiliser un moteur de recherche avec une requête du type `pyhon pandas add a column` [7] Le mot-clé `lambda` est souvent utilisé pour condenser l'écriture de fonctions [8].

Exercice

Une fois la troisième colonne ajoutée, il faut effectuer la transformation suggérée par la figure 2 et dessiner la courbe. On pourra utiliser la fonction `Dataframe.pivot` [9].

Quatrième demi-heure : Données manquantes et lissage

Deux exercices au choix.

Exercice 1

Les graphiques montrent des courbes très saccadées. On souhaite les lisser : chaque point de la courbe non lissée correspond à une moyenne horaire de la courbe saccadée.

```
mat = numpy.matrix ( [ 0.0, 0.1, 0.3, 0.4 ] ).transpose()  # crée une matrice 4x1
print (mat [ 1:-1,:] )    # enlève la première et la dernière ligne
mat = mat + mat * 2.4     # fait des additions avec cette matrice
```

7. Vous trouverez peut-être cette page : http://stackoverflow.com/questions/12555323/adding-new-column-to-existing-dataframe-in-python-pandas.
8. voir http://www.siteduzero.com/informatique/tutoriels/apprenez-a-programmer-en-python/les-fonctions-lambda.
9. voir http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html?highlight=pivot#pandas.DataFrame.pivot

Exercice 2

On souhaite proposer une façon de remplacer les données manquantes. On désigne $X(h,s)$ le nombre de vélos disponibles pour l'heure h et la semaine s. On note $X(.,s)$ la moyenne pour l'heure h sur plusieurs semaines et $X(h,.)$ la moyenne hebdomadaire. On remplace une valeur manquante par $X(h,s) = \sqrt{X(.,s)X(h,.)}$. Corriger les données pour une plage de données manquantes puis tracer le résultat.

Tout d'abord il faut remplacer les mauvaises valeurs par la valeur `numpy.NaN`, on propose les dernières valeurs de la semaine 31 (après `'4-16:00'`). Pour vérifier cela, on utilise Excel pour observer les données :

```
piv.to_excel("pivot.xlsx")  # cela nécessite le module openpyxl
# si celui n'est pas installé, il faut enregistré le fichier au format texte
piv.to_csv("pivot.txt", sep="\t")
```

On remplace les valeurs de la plage détectée par `NaN` :

```
piv.ix['4-16:00':,31] = numpy.NaN
# vous pouvez vérifier que sous Excel (en utilisant les lignes précédentes),
# les cellules sont vides.
```

Ensuite, ces bouts de code pourrait vous aider :

```
# on crée une matrice de type numpy.array
mx = numpy.array ( [ [ 10.0, 1.0], [2.0, 3.] ] )

# on calcule la moyenne de la première colonne
print ( numpy.average( mx[:,0] ))

# on remplace la première valeur par NaN
mx[0,0] = numpy.NaN

# on calcule la moyenne de la première colonne sans les valeurs NaN
print ( numpy.average( mx[~numpy.isnan(mx[:,0]),0]) )

# on utilise la fonction isnan car numpy.NaN != numpy.NaN est vrai
print ( numpy.NaN != numpy.NaN )  # aussi surprenant que cela soit

# on parcourt toutes les dates de la matrice obtenue après pivot :
for i,k in enumerate(piv.index) :
    print (i,k)
```

Ensuite c'est à vous de jouer.

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

La complétion des valeurs manquantes vous paraît-elle judicieuse ? Comment l'améliorer ?

La société qui gère les vélib déplace les vélos des stations qui ont en trop vers les stations qui n'en ont pas assez. Dans quelle catégorie classez-vous cette station ? Pensez-vous pouvoir construire une fonction qui classe les stations vélib dans l'une de ces deux catégories ? Vous pourrez vous inspirez de ce blog : http://www.xavierdupre.fr/blog/2013-09-26_nojs.html.