Algorithmie et programmation

Première année (groupe D)

RAPPORT DU PROJET PROGRAMMATION

Etudiantes : Leeloo ROUCH

Ambre BOUSSUGE-AUBERT

Enseignants : Nassim LAGA

Pierre-Aymeric MASSE

Sommaire

I. Usage de Github

Nous nous sommes tout d'abord créé un compte sur Github.

Nous avons ensuite téléchargé chacune Git et Pyzo, si ce n'était pas déjà fait.

1: Logos Pyzo et Github

Pour pouvoir utiliser la fenêtre Git Bash pour mettre nos avancements sur Github, nous avons commencé par ouvrir Git Bash et rentrer la commande git init pour créer un repository dans notre ordinateur.

```
bejad@Ambrouille MINGW64 ~
$ cd eivppython/

bejad@Ambrouille MINGW64 ~/eivppython
$ git init
Initialized empty Git repository in C:/Users/bejad/eivppython/.git/

bejad@Ambrouille MINGW64 ~/eivppython (master)
$ git status
On branch master

No commits yet
```

2: Initialisation du projet

Ci-dessus on voit que nos programmes seront dans le fichiers nommé eivppython. Depuis, les premières manipulations les noms de fichier ont évolué mais les captures des étapes n'ont pas été reprises.

A chaque fois que nous terminions de travailler sur le code, nous ouvrons la fenêtre Git Bash et à l'aide des commandes git commit -> git push, nous réuploadions notre programme.

Ou sinon nous chargions directement sur Github nous programmes modifiés, soit un créant des nouvelles versions de programmes à chaque fois, pour être surs d'avoir toujours une ancienne version fonctionnelle quelque part, soit en écrasant le précédent si nous étions sûres de la nouvelle version.

Ainsi nous avons réussi à toujours travailler sur le dernier programme sans perdre d'information en route.

II. Prise en main du sujet

Nos variables sont appelées de la façon suivante : noise, temp, humidity, lum et CO2. Elles sont rentrées comme des chaines de caractères dans toutes nos fonctions.

Pour les fonctionnalités demandées dans le projet :

1. Afficher des courbes montrant l'évolution d'une variable en fonction du temps.

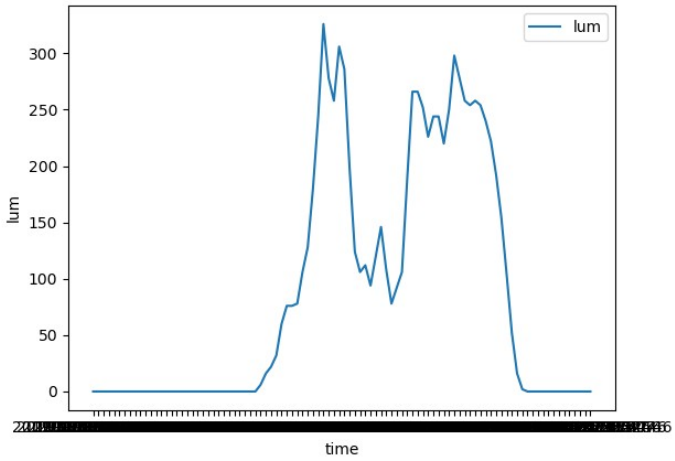
La fonction dans le projet s'appelle par tab_donnees et génère une liste des données pour le capteur voulu et la variable voulue, sur l'intervalle de temps voulu. En outre, la valeur du 'end_date' correspond à la date de fin, exclue. Sur l'exemple ci-dessous, les valeurs considérées sont celles allant jusqu'au 15/08/2019 à 23h59min59sec.

Cette fonction nous est utile pour toutes les autres fonctionnalités demandées dans le programme.

Ainsi pour afficher une courbe montrant l'évolution d'une variable dans le temps nous avons fait une autre fonction nommée courbe.

Pour la commande suivante : `>>> courbe(1, 'lum', '2019-08-15', '2019-08-16')`

On obtient la courbe suivante, ici pour une journée :



3: Exemple de courbe de variable en fonction du temps

2. Afficher les valeurs statistiques sur la courbe

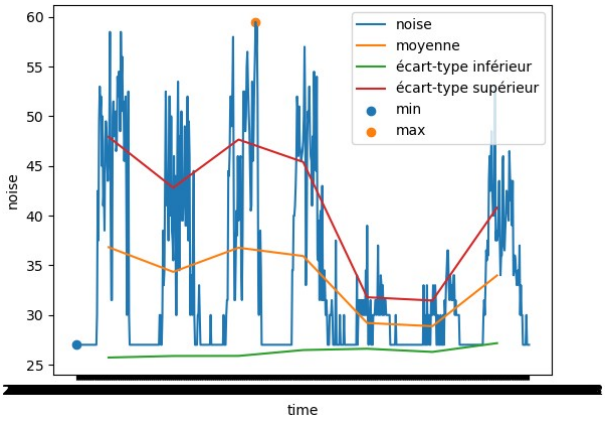
Nos moyennes, min, max, écarts-type sont calculées sans utiliser de fonctions python.

Nous avons calculé la moyenne de la variable étudiée sur chaque jour de l'intervalle de mesure, ce sont également ces moyennes qui nous permettent de calculer les différents écarts-types. Nous avons ainsi tracé la ligne brisée passant par les points de moyenne de chaque jour de mesure en prenant pour abscisse l'heure médiane de la journée. Les écarts-types inférieur et supérieur viennent encadrer le tracé de la moyenne, pour représenter l'intervalle dans lequel fluctuent les valeurs.

En rentrant la commande suivante :

```
>>> courbe_stat(1, 'noise', '2019-08-15', '2019-08-22')
```

On obtient le graphe :



4: Exemple de courbes avec les statistiques affichées

3. Calculer l'indice "humidex" :

Pour le calcul de l'indice "humidex", nous avons utilisé la formule suivante, à l'aide des données de la température de l'air et de l'humidité relative :

$$\text{Humidex} = \text{Air temperature} + \frac{5}{9} \times (6.112 * 10^{7.5 \times \frac{T}{237.7+T}} \times \frac{H}{100} - 10)$$

5: Formule indice humidex

Pour utiliser la fonction dans la fenêtre de commande il faut l'appeler de la manière suivante, avec le capteur et les dates :

```
>>> humidex(1,'2019-08-11','2019-08-22')
```

Pour simplifier la lecture des indices humidex, nous voulons afficher un humidex par jour demandé. Pour ce faire, nous faisons la moyenne de tous les humidex calculés en une journée puis l'affichons.

```
>>> humidex(1,'2019-08-11','2019-08-22')
le 2019-08-11, le coef humidex vaut : 29.26
le 2019-08-12, le coef humidex vaut : 26.89
le 2019-08-13, le coef humidex vaut : 26.38
le 2019-08-14, le coef humidex vaut : 28.46
le 2019-08-15, le coef humidex vaut : 31.78
le 2019-08-16, le coef humidex vaut : 30.59
le 2019-08-17, le coef humidex vaut : 29.56
le 2019-08-18, le coef humidex vaut : 29.84
le 2019-08-19, le coef humidex vaut : 30.77
le 2019-08-20, le coef humidex vaut : 29.11
le 2019-08-21, le coef humidex vaut : 28.93
```

6: Affiche des indices humidex

4. Calculer l'indice de corrélation entre un couple de variables

L'indice de corrélation r se calcule comme suit :

$$Cov(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(X_i - \overline{X}) \cdot (Y_i - \overline{Y})$$

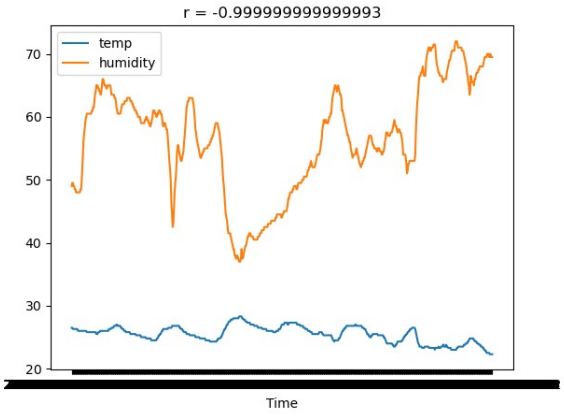
7: Formule utilisé pour la covriance

$$r(X,Y) = \frac{Cov(X,Y)}{\sigma_X \cdot \sigma_Y}$$

8: Formule utilisé pour l'indice de corrélation

On se sert de cette fonction de la manière suivante :

```
>>> coef_correlation(1,'temp','humidity','2019-08-15', '2019-08-22')
```



9: Exemple de courbes avec l'affiche de r, l'indice de corrélation

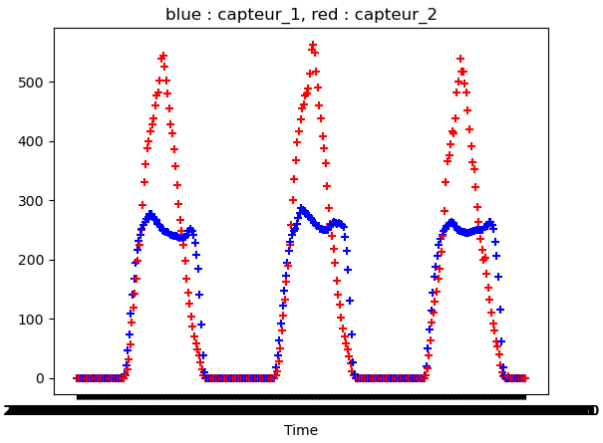
Nous sommes dans le groupe D. Nos compléments de consignes étaient :

5. Mesurer similarités des capteurs pour chaque dimension

Nous avons donc fusionné les deux listes de temps et les deux listes de valeurs, puis trié la liste de temps obtenue, par ordre chronologique, ce qui a aussi trié la liste des valeurs obtenues. Ensuite, nous avons tracé chaque point séparément, d'une couleur bleue s'il correspondait au capteur_1, rouge s'il correspondait au capteur_2, pour artificiellement superposer les 2 relevés de valeurs et ainsi observer le comportement similaire ou non des données.

Nous avons choisi cette méthode suite à un échec explicité dans la partie III.

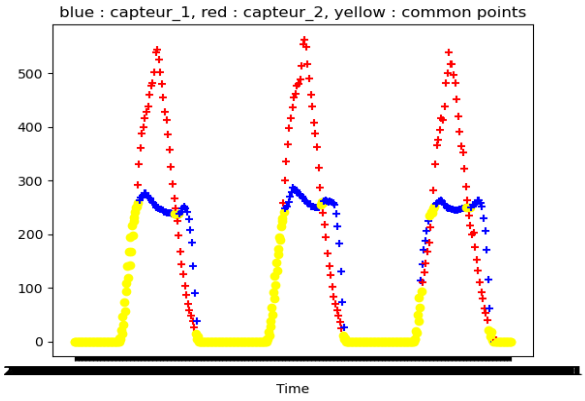
```
>>> superposition_courbes(1, 4, 'lum', '2019-08-12', '2019-08-15')
```



10: Exemple de superposition de deux relevés de mesure

Concernant la mise en évidence des similitudes, nous avons tracé d'une certaine couleur les points des deux courbes qui, pour des dates proches, évoluaient dans le même intervalle de valeurs.

```
>>> similarites(1, 4, 'lum', '2019-08-12', '2019-08-15')
```

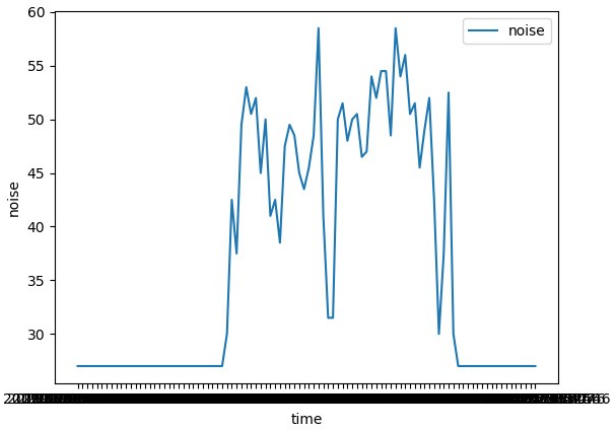


11: Exemple de représentation des similitudes

6. Bonus

Nous avons trouvé les périodes horaires d'occupations des bureaux par déduction.

Nous avons remarqué que les bureaux étaient occupés environ entre 8h - 12h30 et 13h30 - 19h en fonction du bruit capté les journées.



12: Exemple avec le bruit au cours d'une journée

III. Essais, échecs

Nous avons rencontré des difficultés pour :

- Le calcul d'indice de corrélation : le programme nous retourne des indices de corrélation toujours égaux à 1 ou -1 (si l'on ne tient pas compte des décimaux dus aux approximations de python lors des différentes étapes de calcul) malgré la bonne formule employée et les étapes de calcul qui nous paraissent correctes à l'intérieur du code.

- Pour la mesure des similarités, le but était de superposer les courbes de deux capteurs pour une même variable et un même intervalle de mesure. SI les courbes étaient suffisamment rapprochées (par exemple pour un écart inférieur à 10% des valeurs de la courbe 2 par rapport à celles de la courbe 1), la similarité entre les 2 courbes aurait été mise en évidence par une seule courbe moyennant des deux. Là où l'écart dépassait les 10%, les parties de courbes seraient restées disjointes.

  Cette solution n'a pas abouti car nous n'avons pas réussi à fusionner les deux listes de temps des deux capteurs tout en traçant séparément la courbe associée à chaque capteur. En outre, si nous ne fusionnions pas les deux listes de temps, les deux courbes s'affichaient l'une après l'autre sans se superposer.

IV. Le travail en binôme

Nous avons effectué un travail régulier en séances d'appel sur Messenger.

Nous avons su nous compléter et exploiter les points forts de chacune. Le partage des fichiers de travail sur Github s'est fait de manière très fluide et a permis un travail collaboratif efficace.

Table des légendes