MARMORAT Guillaume 01/12/2020

GAL Yoann

IVP 62

# Lien pour le répertoire Github: [Guillaume-Marmorat/projet\_td\_eivp at master (github.com)](https://github.com/Guillaume-Marmorat/projet_td_eivp/tree/master)

# **Rapport projet de programmation informatique**

# **Sujet 2 : groupe D**

Lors de tout ce projet nous avons fonctionné de la même manière. Chaque semaine sur le créneau du mardi matin nous faisions un point sur l’avancé de notre projet. Cela nous permettait dans un premier temps de maîtriser l’avancée de nos tâches, mais aussi de se poser les questions lorsque nous bloquions.

Après une analyse du sujet commune, nous avons très vite décidé de se répartir le travail comme suit :

* Guillaume avait pour mission de s’occuper de l’importation du fichier CSV sous la forme d’une liste de listes, de créer le répertoire GitHub pour héberger notre travail et grâce à Git enregistrer notre travail sur ce répertoire. Par la suite, il a également été chargé de programmer l’affichage des courbes.
* Yoann devait quant à lui programmer en Python de telle sorte que notre script final puisse avoir les fonctionnalités attendues par le sujet.
* Nous avons travaillé ensemble pour implémenter l’algorithme permettant de mesurer les similarités des capteurs automatiquement, et de les montrer sur des courbes.

Concernant l’obtention de notre liste de listes, Guillaume a rencontré les problèmes suivants :

1. Il obtenait une succession de listes. Il a résolu ce problème à l’aide d’un programme simple, puis a stocké notre liste de liste dans une variable globale que nous avons choisi de nommer T (pour Tableau).
2. Après l’obtention de sa liste de liste, lorsqu’il relançait le programme cela lui retournait une liste vide. Après un rendez-vous avec un professeur nous avons trouvé la source du problème : une fois que le tableau CSV était lu en entier, il fallait bien penser à le relire une nouvelle fois lors de sa prochaine utilisation. Il ne se stocke nulle part.
3. Chaque sous listes étaient en fait des chaînes de caractères, pour obtenir la forme souhaitée il a tout d’abord séparé en chaînes de caractères plus petites comportant seulement une valeur. Enfin, avec une boucle « for » imbriquée, il a converti les petites chaînes de caractères en entier ou en flottant selon les valeurs. Tout cela a été insérer dans le programme de création du tableau afin de limiter le nombre de fonctions.
4. Pour tracer les courbes, Guillaume a codé un algorithme qui nous semble bon mais à chaque test nous obtenons un message d’erreur (ValueError: x and y must have same first dimension, but have shapes (1345,) and (1,)) et ce quelques soit le capteur.

Pour la partie de Yoann, plusieurs questions se sont posées :

1. Pour l’indice humidex, comment le renvoyer ? Après concertation, nous avons décidé de renvoyer une liste de listes où chaque sous-liste comporte deux éléments (l’indice du capteur et l’indice humidex correspondant). Cela nous a semblé être le choix le plus judicieux pour ensuite pouvoir appliquer à cette nouvelle liste de listes les algorithmes codés précédemment (moyenne, médiane, etc…).
2. Quelle méthode de tri employer ? En effet, pour afficher la médiane d’une liste de nombres, il faut que cette liste soit préalablement triée. Nous devions donc implémenter un algorithme de tri. Dans un premier temps nous avons pensé à faire un « tri fusion » car il possède une faible complexité temporelle. Cependant notre choix final s’est penché sur la méthode dîtes de « tri-rapide » car en se documentant, nous avons réalisé que le tri fusion était très mauvais lorsqu’il s’agissait d’une grande liste (car il y a une très mauvaise complexité au niveau de l’espace mémoire) or c’est précisément notre cas ici !
3. Pour coder la fonction « min » qui prend comme arguments un numéro de capteur ainsi qu’un paramètre afin de calculer la valeur minimale de ce paramètre pour un capteur précis, on s’est rendu compte que l’initialisation ne fonctionnait pas correctement (car la première valeur ne correspondait pas au bon capteur). Pour résoudre cela Yoann à implémenter une fonction « separationcapteur » qui renvoie une liste de listes qui comprend uniquement les valeurs correspondantes au capteur souhaité. Cela permet, en utilisant cette fonction dans le code de « min » de réaliser une initialisation correcte.
4. Pour la fonction « maxi » qui prend comme arguments un numéro de capteur et un paramètre, on a choisi de résoudre ce même problème d’initialisation en prenant 0 comme valeur initiale car toutes les valeurs du tableau sont positives ! Il pouvait se poser la question pour la température, mais à l’intérieur d’un bureau on peut conjecturer que la température dépasse tout le temps les 0°C… Grâce à cette astuce on évite l’appel de la fonction « separationcapteur » ce qui est intéressant du point de vue de la complexité temporelle.

Au fur et à mesure que les séances avancées, nous prenions de plus en plus conscience de l’importance capitale de la forme de nos algorithmes. En effet, nous nous sommes rendus compte que même nous, nous avions du mal à comprendre d’une semaine sur l’autre, à quoi correspondait les choix de nos notations… C’est pourquoi nous avons pris le temps de tout reprendre pour faciliter ensuite l’utilisation de nos algorithmes. Par exemple, nous nous sommes rendus compte que (machinalement), nous avions mis notre liste T en argument dans tous nos algorithmes ! Or il est laborieux de connaître la commande d’appel du tableau CSV et de la rentrer à chaque fois dans les arguments de nos fonctions… Nous avons donc choisi de ranger notre liste de listes dans une variable globale et de l’appeler sans qu’elle soit mise dans les arguments. Nous avons donc pris conscience que nous ne faisions pas les algorithmes pour nous, mais pour des personnes tierces qui veulent pouvoir s’en servir le plus simplement possible. Il en est de même pour l’affichage des courbes. Notre approche a évolué car il ne fallait pas simplement afficher la courbe, il fallait l’afficher de telle sorte qu’une personne qui ne connaisse pas l’implémentation de l’algorithme puisse comprendre les résultats exposés en lisant les informations sur cette courbe.

Pour la partie durant laquelle nous avons travaillé en commun (cela nous semblait indispensable notamment à cause du grand nombre de nuances que comportait le sujet), nous avions pris le parti de travailler avec une liste de listes, dans laquelle chaque sous-liste donnait dans l’ordre : le numéro de ligne (en partant de 0), l’identifiant du capteur, le niveau sonore en dBA, la température en °C, l’humidité relative en %, le niveau lumineux en lux, la quantité de CO2 en ppm et la date. Nous avons choisi ce format car nous avions tous les deux eu l’habitude de travailler avec de tels formats lors de notre apprentissage de Python.

Pour répondre à notre sujet (rappel du sujet : « Mesurer les similarités des capteurs pour chaque dimension, qu’en concluez-vous ? Proposez et implémentez un algorithme permettant de mesurer la similarité automatiquement et de la montrer sur les courbes. »), nous avons adopté le point de vue suivant : nous devons comparer les moyennes de chaque capteur pour un même paramètre. Plus la différence des moyennes sera faible, plus les capteurs seront similaires concernant le paramètre choisi. De ce fait, nous avons décidé de laisser le choix à l’exécutant de la fonction « similarites » de choisir l’intervalle pour lequel il considère que deux capteurs sont similaires. Graphiquement, nous avons pris la décision de représenter sur un même graphique les moyennes de tous les capteurs concernant le paramètre étudié. Pour renforcer la lisibilité de ce graphique, nous avons eu l’idée de tracer l’intervalle de confiance choisit sur ce graphique. Cela permet facilement de repérer quels capteurs peuvent-être assimilés comme similaire au capteur de référence (le « capteur de référence » est celui qui est pris comme premier argument lors de l’appel de la fonction) : si la moyenne du capteur numéro x se situe entre les deux courbes symbolisant l’intervalle de confiance, alors il est considéré comme similaire au capteur de référence.

Par ce projet, nous avons découvert les outils Git et GitHub. Ces derniers nous ont semblé utiles après une période d’acclimatation. En effet, pour les utiliser correctement il faut s’y habituer car toutes les fonctionnalités ne sont pas très intuitives. Nous avons trouvé que son utilisation pouvait être utile lors de travail de groupe nombreux mais pour un travail à deux l’utilité est réduite car nous pouvons échanger facilement nos fichiers sans passer par GitHub. De même, il nous semble que son utilisation en groupe peut être un peu limitée si tous les membres du groupe ne suivent pas au préalable une formation adéquate.

Pour conclure, ce projet nous a permis d’apprendre à manipuler les fichiers CSV avec Python, de consolider nos compétences en codage et nous a fait découvrir Git et Github. De plus nous avons appris à nous corriger en autonomie pour nos codes et à rechercher les informations utiles pour y parvenir.