

Rapport sur le travail d'anonymisation de données concernant des patients réels

VINHAS Jérémy

I) Structures utilisées

a) Le code C++

On retrouve la classe **DataAnonymizer** qui va permettre de modifier chacune de nos valeurs afin d'anonymiser les patients. La méthode d'anonymisation sera décrite plus tard dans le rapport. Il y a également la classe **DataProcessor**, qui va permettre dans un premier temps de lire les fichiers des patients, puis de modifier les valeurs et d'enregistrer ces valeurs modifiées dans un fichier csv. Les fichiers "events" et "series" sont fusionnés pour ainsi n'avoir qu'un fichier par patient en sortie. Ces fichiers sont ensuite stockés dans un répertoire préalablement créé.

b) Le code Python

La classe **DataProcessor** contient des méthodes permettant de lire les fichiers csv, mais aussi de calculer la dtw.

La classe **AnalysisGenerator** va générer tous les fichiers que l'on retrouve dans le dossier 'stats'. On retrouve au sein de ce dossier, des fichiers contenant les valeurs minimum, maximum, variance et autres paramètres statistiques, et ceux pour chaque paramètre physiologique. De plus, la classe contient une méthode permettant de générer un boxplot montrant la distribution entre les données de patients réels et des patients anonymes.

La classe **StatisticsGenerator** permet quant à elle de réaliser les tests statistiques entre les données brutes et les données anonymisées. Ici aussi, on va générer un fichier par paramètre statistique et par paramètre physiologique. Cette classe permet l'évaluation de notre méthode d'anonymisation.

Au sein de la fonction main, on crée des attributs pour chacune de ces classes. Puis certaines méthodes sont appelées qui vont permettre la génération des fichiers.

II) Méthode d'anonymisation

Notre programme comporte une seule méthode d'anonymisation. Un nombre va être généré aléatoirement entre deux bornes (qui seront choisis par l'utilisateur de lors de l'appel du programme). Puis, toutes les valeurs seront modifiées selon ce nombre qui sera converti en un pourcentage. Une fois les valeurs modifiées, elles seront enregistrées dans un nouveau fichier CSV, avec un fichier par patient. On modifie seulement les valeurs, donc l'ordonné, mais on ne modifie pas le temps à laquelle les événements se produisent. De plus, lors de l'exécution du programme, il est possible de ne sélectionner uniquement la génération des données anonymisées, sans réaliser les tests statistiques ensuite. Cependant, les bornes

ont un seuil minimum qui est des -4 / +4 afin d'éviter de réaliser des changements trop faibles et donc de ne pas anonymiser les valeurs. Cette borne a été choisie de façon arbitraire.

III) Bilan

Par rapport au cahier des charges, ce programme respecte de nombreux points : la méthode d'anonymisation est fonctionnelle, ainsi que l'analyse de celle-ci, avec toutes les sorties demandées. Les valeurs statistiques sont enregistrées dans un répertoire, tandis que les résultats des tests statistiques sont stockés dans un autre répertoire. Nous avons également la génération d'un boxplot, et d'un fichier CSV permettant d'évaluer la distribution des valeurs entre les valeurs originales et les valeurs anonymisées.

Concernant la structure de notre répertoire, les programmes sont séparés selon la consigne. Il y a la présence d'un readme dans le répertoire principal, ainsi qu'un makefile dans le répertoire contenant les programmes C++. On retrouve de plus un fichier install permettant d'installer les librairies nécessaires à l'utilisation du programme. Il y a un jeu de données d'exemple disponible afin d'essayer le programme sur un plus faible échantillon, avec un temps de calcul plus court donc. Au sein d'un fichier paramètre, le nom du répertoire final doit être renseigné, ainsi que le nombre d'échantillons sur lequel on veut effectuer les tests statistiques.

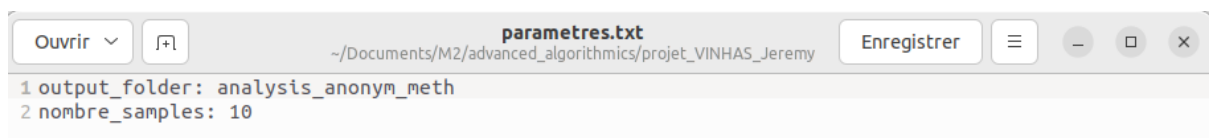
Concernant les points à améliorer, il serait préférable d'avoir au minimum une autre méthode d'anonymisation afin de pouvoir comparer les sorties entre les deux. De plus, cette méthode pourrait modifier le temps des événements, et ainsi influencer les valeurs selon l'abscisse.

IV) Annexes

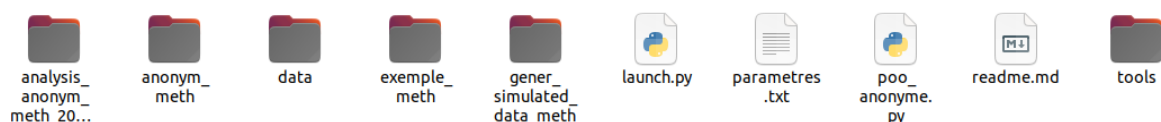
Appel de programme n°1 :

```
jeremy@carlos: ~/Documents/M2/advanced_algorithmics/projet_VINHAS_Jeremy$ python3 launch.py data 1000 -8 8 gener_simulated_data_meth parametres.txt o
```

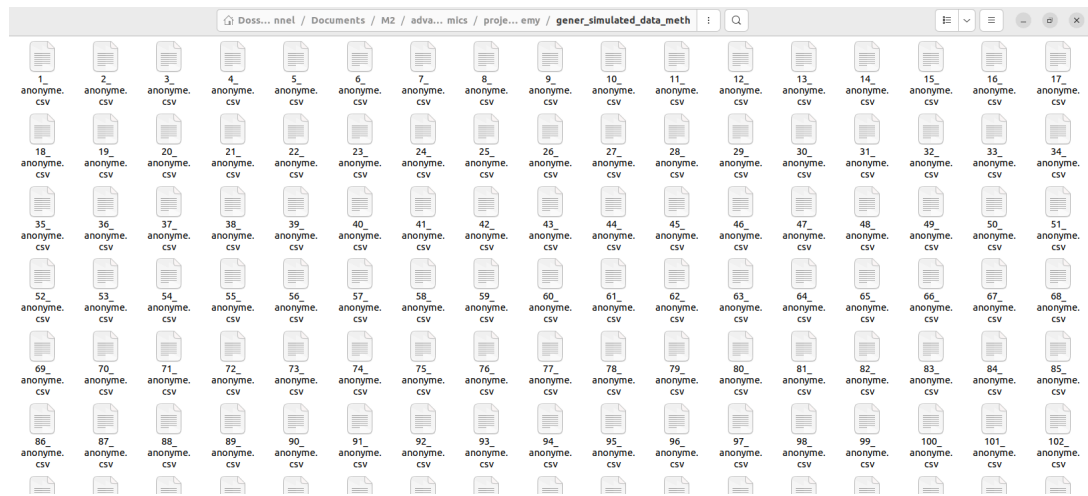
Fichier paramètres :



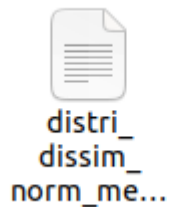
Résultat dans le répertoire courant :



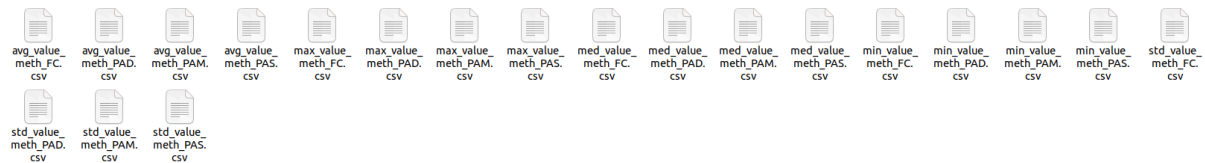
Résultat dans gener_simulated_data_meth :



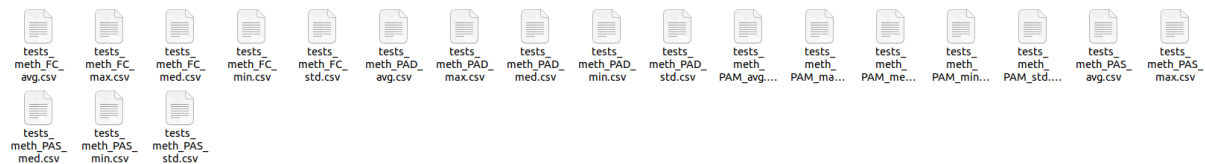
Résultat dans analysis_anonym_meth :



Résultat dans le fichier stats :





Résultat dans le fichier tests :



Appel du programme n°2 :

```
Le fichier parametres.txt a été trouvé
jeremy@carlos:~/Documents/M2/advanced_algorithmics/projet_VINHAS_Jeremy$ python3 launch.py data 30 -8 8 gener_simulated_data_meth parametres.txt o
Le fichier parametres.txt a été trouvé
```





Fichier paramètre :

Ouvrir 

parametres.txt

~/Documents/M2/advanced_algorithmics/projet_VINHAS_Jeremy

Enregistrer



```
1 output_folder: essai_2
2 nombre_samples: 10
```

Répertoire courant :