Master de Bioinformatique
de Nantes Université
2023– 2024

Advanced algorithmics and programming for biologists /
Models, methods and algorithms for bioinformatics

Encadrement C. Sinoquet

Mise en garde : Aucun fichier à générer pour le projet ne doit être terminé par une ligne vide. En d'autres termes, il n'y a pas de retour à la ligne sans texte avant.

Calendrier :
- lancement du projet :
  jeudi 28 septembre :
- points de suivi intermédiaires :
  vendredi 20 Octobre
  vendredi 10 novembre (contrôle intermédiaire)
  mardi 28 novembre (à confirmer)
- date limite de remise :
  dimanche 10 décembre
- évaluation :
  mardi 12 décembre
  (démonstration en salle sur matériel sur lequel votre TP a été réalisé (portable admis))

Pénalité de 1 point par jour de retard, ouvré ou non.
Aucun devoir rendu en janvier 2024 ne sera évalué (notation 0).

Sujet à traiter en langage C++ (et R ou Python pour les scripts, et **seulement pour les scripts**[1]), sous Linux.
Sauvegardes nécessaires en cours de travail, sur divers supports.
Sauvegardes nécessaires de versions partielles en cours de travail, sur divers supports.
(Aucun délai ne sera accordé si le projet a été perdu, faute de sauvegarde).

Anonymisation de séries temporelles multivariées, associées à des traces d'événements

Chaque binôme d'étudiants devra implémenter
- une méthode d'anonymisation des données décrites ci-après
- un protocole d'évaluation de la qualité de l'anonymisation.

[1] Dans le contexte de ce projet, un script est un code destiné à lier des unités de code avancé entre elles, ou à mettre en œuvre des conversions de fichiers d'un format dans un autre.

Les données sont constituées de séries temporelles multivariées (MTS[2]) et de traces d'événements (ET[3]). A chaque MTS est associée une ET. Une MTS est constituée de plusieurs séries temporelles univariées (UTS). Chaque UTS décrit l'évolution temporelle d'une variable donnée (par exemple le paramètre physiologique d'un patient monitoré au bloc opératoire), sous l'influence d'une ou de plusieurs actions annotant l'ET. Dans le contexte du bloc opératoire, une action est par exemple l'administration d'un analgésique.

Il existe donc potentiellement des dépendances entre chaque UTS et l'ET associée à la MTS à laquelle elle appartient. Il existe peut-être également des causes extérieures provoquant des dépendances entre les variables décrites par ces données.

On considère que n variables sont décrites par chacune des MTS, et que p observations sont disponibles (e.g., p patients).

- Pour chaque MTS, l'horodatage est réalisé en temps discret : les mesures sont séparées par un pas de temps constant (time_step), pour chaque UTS considérée et les UTS d'une même MTS sont synchrones.

- Les longueurs des UTS de la même MTS sont nécessairement égales. On parle alors de longueur de la MTS.

- D'une MTS à une autre, les longueurs ne sont pas nécessairement égales.

- L'horodatage de chaque ET est réalisé en temps continu.

Les données à anonymiser peuvent présenter l'un des deux formats suivants :

- format 1 :

  time_series_var1_1.txt event_trace_1.txt
  time_series_var2_1.txt
  …
  time_series_varn_1.txt
  time_series_var1_2.txt event_trace_2.txt
  time_series_var2_2.txt
  …
  time_series_varn_2.txt
  ...
  time_series_var1_p.txt event_trace_p.txt
  time_series_var2_p.txt
  …
  time_series_varn_p.txt

- format 2 :

  time_series_var1.txt event_traces.txt
  time_series_var2.txt
  ...
  time_series_varn.txt

[2]multivariate time series
[3]event trace

Chacun des fichiers du format 2 comporte p lignes (qui n'ont pas nécessairement la même longueur).

Il n'existe pas de méthode standard permettant de répondre au problème d'anonymisation de ce type de données caractérisées par des dépendances multiples. Toute proposition de méthode d'anonymisation devra préserver les dépendances ET-MTS. Il ne doit pas y avoir de perte d'information. L'anonymisation d'une (MTS, ET) génèrera une MTS* anonymisée et une ET* modifiée pour préserver la dépendance entre MTS* et ET*.

- exemple :
  - rajouter un pourcentage de q % points à une UTS (avec lissage) et gérer l'ET
  - rajouter un pourcentage de q % segments de longueur distribuée aléatoirement dans [lb, ub] à une UTS (avec lissage)
  - mixer les deux approches précédentes
  - ...

Input :

Jeu de données mis à disposition sur https://github.com/alpharty/BDLBS_dataset

Attention : dans multivariate/<number>_series.txt, ne pas tenir compte de la ligne Time = -1. Même remarque pour les binômes qui utilisent univariate/<parameter>.txt

Output :

Pour toute méthode, ou pour chaque instance d'une méthode lorsque celle-ci est paramétrée (par exemple par un pourcentage plus ou moins élevé de pas de temps rajoutés lors de l'anonymisation etc), générer :

- dans le répertoire projet_codes_<nom1>_<nom2>_2021_12_<jour-num>_<jour-lettres>/**gener_simulated_data_meth**

ou dans le répertoire projet_codes_<nom1>_<nom2>_2021_12_<jour-num>_<jour-lettres>/**gener_simulated_data_meth_<par>_<valeur>**

exemple :
projet_codes_<leroy>_<leprince>_2021_12_16_thur/gener_simulated_data_meth_noise-percent_5

* les fichiers **<number>_series.csv** des 1000 séries multivariées anonymisées au moyen d'une méthode créée par vos soins, suivant exactement la structure des fichiers sous https://github.com/alpharty/BDLBS_dataset/multivariate, avec une en-tête, et l'horodatage 0, 30, 60 etc (ne pas faire commencer l'horodatage à -1 comme c'est le cas sous le dépôt git) , séparateur : virgule

* les fichiers **<number>_events.csv** des 1000 traces d'événements « anonymisées », suivant exactement la structure des fichiers sous https://github.com/alpharty/BDLBS_dataset/ multivariate, avec une en-tête, et l'horodatage 0, 30, 60 etc, séparateur : virgule

Analyse de l'anonymisation :

Pour votre méthode, ou pour chaque instance d'une méthode lorsque celle-ci est paramétrée (par exemple par un pourcentage plus ou moins élevé de pas de temps rajoutés lors de l'anonymisation etc), générer :

- dans le répertoire projet_codes_<nom1>_<nom2>_2021_12_<jour-num>_<jour-lettres>/**analysis_anonym_meth**

ou dans le répertoire projet_codes_<nom1>_<nom2>_2021_12_<jour-num>_<jour-lettres>/**analysis_anonym_meth_<par>_<valeur>**

* le fichier **distri_dissim_norm_meth[_<par>_<valeur>].csv** des 1000 dissimilarités normalisées obtenues comme suit :

- pour chacun des 1000 patients anonymes pa :

-- tirer au hasard 10 patients réels pr_i,

-- calculer la DTW[4] univariée entre pa et pr_i pour FC, PAS, PAM et PAD,

-- calculer la DTW multivariée (DTWm) entre pa et pr_i comme la moyenne des 4 DTW univariées précédentes

-- mémoriser (temporairement) la DTWm minimale sur les 10 patients réels

- sur la distribution des 1000 DTWm_min minimales ainsi obtenue, calculer la moyenne E et l'écart-type S, normaliser chacune des 1000 dissimilarités : DTWm_min normalisée = (DTWm_min – E)/S ,

- en-tête du fichier : dissim_norm, et séparateur : virgule

- produire un boxplot ou un violinplot **violinplot_meth[_<par>_<valeur>].png** (regrouper tous les violinplots de toutes vos instances méthodes (le cas échéant) sur le même graphique)

* les fichiers
avg_values_meth_<param_physio>[_<par>_<valeur>].csv (en-tête : avg_anonym, avg_real, séparateur : virgule)
std_values_meth_<param_physio>[_<par>_<valeur>].csv (en-tête : std_anonym, std_real, séparateur : virgule)
med_values_meth_<param_physio>[_<par>_<valeur>].csv (en-tête : med_anonym, med_real, séparateur : virgule)
min_values_meth_<param_physio>[_<par>_<valeur>].csv (en-tête : min_anonym, min_real, séparateur : virgule)
max_values_meth_<param_physio>[_<par>_<valeur>].csv (en-tête : max_anonym, max_real, séparateur : virgule
avec param_physio appartenant à {FC, PAS, PAM, PAD}
obtenus comme suit :

- pour chacun des 4 paramètres param_physio FC, PAS, PAM et PAD,

- - pour chacun des 1000 patients anonymes pa_i, calculer la moyenne param_avg, l'écart-type param_std, la médiane param_med, le minimum param_min, le maximum param_max sur les valeurs de la série univariée correspondant au paramètre param et au patient pa_i

- - procéder de même pour chacun des 1000 patients réels pr_i

- - mémoriser les résultats dans les fichiers ci-dessus

* le fichier **tests_meth_<param_physio>_avg_[_<par>_<valeur>].csv**
avec param_physio appartenant à {FC, PAS, PAM, PAD}

- en-tête : statKS, pvalKS, statWMW_p, pvalWMW_p,statWMW_up, pvalWMW_up, diste_avg, diste_std, diste_med, diste_min, diste_max et séparateur : virgule

Les deux distributions de 1000 valeurs chacune servant à faire successivement le test KS, le test WMM_p (si possible, i.e., si un appariement ligne à ligne a du sens) ou le test

[4] utiliser la librairie https://dtaidistance.readthedocs.io/en/latest/

WMW_up sont prises dans le fichier
avg_values_meth_<param_physio>[_<par>_<valeur>].csv adéquat.

- - **statKS, pvalKS** : statistique de test et p-valeur du test de Kolmogorov-Smirnov
- - **statWMW_p, pvalWMW_p** : statistique de test et p-valeur du test apparié (paired) de Wilcoxon / Mann-Whiney (quand il existe une correspondance entre un patient anonyme et un patient réel qui a servi de fil conducteur dans la génération du patient anonyme)
- - **statWMW_up, pvalWMW_up** : statistique de test et p-valeur du test non apparié (unpaired) de Wilcoxon / Mann-Whiney (à faire à titre de comparaison si vous avez pu faire le test WMW apparié)
- - (si possible, i.e., si un appariemment ligne à ligne a du sens) **diste_avg**: distance enclidienne entre les deux vecteurs colonnes du fichier **avg_values_meth_<param_physio>[_<par>_<valeur>].csv** adéquat.
  rappel calcul distance euclidienne entre vecteur x et y de longueur n chacun :
  diste ← racine carrée (somme_i=1 à i=n [(xi - yi)^2])
- - idem pour les 4 autres fichiers (std, med, min, max).

Note : On doit pouvoir installer votre code en étant guidé de bout en bout. On ne doit pas « deviner comment ça s'installe ou comment ça compile». Le meilleur test est de faire installer votre code par un autre binôme qui doit pouvoir aller jusqu'à la compilation (avec un Makefile (OBLIGATOIRE)) et la vérification de la bonne installation au moyen d'un des scripts du type example_meth/launch_<method>_toy_example.py (OBLIGATOIRE).

Points techniques

format obligatoire pour tout fichier de paramètres fournis à vos logiciels :

<description of first parameter>
<possibly spread on several lines>
<identifier of first parameter> <value of first parameter>

<description of second parameter>
<possibly spread on several lines>
<identifier of second parameter> <value of second parameter>

…

<description of last parameter>
<possibly spread on several lines>
<identifier of last parameter> <value of last parameter>

N.B. 1 : Des lignes « vides » séparent les descriptions de paramètres.

N.B. 2 : Il n'y a pas de « dernière ligne vide » (i.e., correspondant à un symbole de fin de ligne supplémentaire).
N.B. 3 : Un identifiant de paramètre est du type (exemple) :
somme_mismatches
N.B. 4 : Un identifiant de paramètre ne comporte pas de majuscules.
N.B. 5 : Un identifiant de paramètre comporte uniquement des « _ » pour séparer des sous-mots.
N.B. 6 : Les identifiants de paramètres sont communs à tous les binômes du M2 Bioinformatique / parcours Bioinformatique pour les Biologistes.
N.B. 7 : Un paramètre réel se note par exemple 3.14116 et non pas 3,14116.
N.B. 8 : **Les chemins de noms de fichiers entrée et sortie ne sont pas indiqués dans le fichier des paramètres, mais sont indiqués dans le script de lancement de la méthode.**

En plus du fichier readme.md, vous fournirez un script Python launch_<method>_toy_example.py dédié à l'exécution de <method> sur l'exemple jouet (script à inclure dans le sous-répertoire toy_example). **OBLIGATOIRE**

Script de lancement de méthode **(pénalité si absence)**
Le script de lancement de toute méthode, intitulé launch_<method>.py prendra en paramètres :
- les chemins du répertoire contenant les fichiers input,
- le chemin du fichier des paramètres propres à la méthode
- le chemin du répertoire dans lequel seront sauvegardés les fichiers output.

Livrables finaux

Logiciel
Chaque binôme doit déposer via un freeware de transfert de fichier (comme We Transfer, https://wetransfer.com/), une archive dénommée projet_codes_<nom1>_<nom2>_2023_12_<jour-num>_<jour-lettres>.zip (par exemple projet_codes_<nom1>_<nom2>_2023_12_12_tues.zip [5)] comportant les codes implémentés.

L'archive project_codes_<nom1>_<nom2>_2023_12_<jour-num>_<jour-lettres>.zip contiendra le répertoire project_codes_<nom1>_<nom2>_2024_12_<jour-num>_<jour-lettres>.
Ce répertoire contiendra
- un sous-répertoire data
- un sous-répertoire anonym_meth (codes sources de la méthode, à structurer en src, obj, exe, avec un Makefile)
- un sous-répertoire analyse_anonym_meth
- un sous-répertoire tools convenablement structuré le cas échéant
- un fichier readme.md
- un fichier install (si nécessaire, avec les commandes en dur pour rapatrier les librairies nécessaires à votre projet)

[5] avec nom1 et nom2 les noms des étudiants du binôme concerné

- un fichier project_.pdf correspondant au mémoire du projet (4 pages maximum)
- un sous-répertoires example_meth structuré en :
    input
    output
    launch_<method>_toy_example.py
    parameters.txt
    readme.md (sauf si redondance avec le readme du répertoire père)

N.B. : Dans le script launch_<method>_toy_example.py, par exemple, au lieu de copier la ligne de commande
mon_executable_pour_methode <chemin1/>patients_reels.txt <chemin2/>par.txt <chemin3>patients_simules 25,
soyez plus informatifs, et écrivez plutôt:
real_profiles = <chemin1/>patients_reels.txt
parameter_file = <chemin2/>par.txt
anonym_profiles = <chemin3/>patients_simules
noise_percent = 25 // percentage of time-stamps added during the anonymization
mon_executable_pour_methode real_profiles parameter_file anonym_profiles 25 noise_percent

Rapport

- rapport, d'au plus 4 pages, présentant
    - les structures de données utilisées,
    - une présentation simple de votre méthode d'anonymisation mais néanmoins « self-contained » (faire tester la compréhension par un autre binôme et indiquer le nom du binôme qui a testé l'intelligibilité de votre présentation ; règles : chaque binôme teste la méthode de deux autres binômes)
    - un bilan concis récapitulant la réalisation effective par rapport au cahier des charges du projet, et récapitulant ce qui reste à faire (par rapport au cahier des charges du projet,), ou au regard des améliorations que vous apporteriez si vous aviez plus de temps,
    - éventuellement, un bilan concis des erreurs que vous n'avez pas réussi à corriger (il vaut mieux le placer ici, plutôt que nous le découvrions nous-mêmes…)

Le rapport devra comporter en annexe les résultats d'exécution de votre algorithme sur plusieurs jeux d'essai significatifs.

• Un rapport redécrivant les fonctionnalités des diverses fonctions est totalement inutile: tout doit être lisible dans le code source.

Evaluation

La notation prendra en compte les points suivants :

• correction des algorithmes,

• lisibilité du code produit,

• choix soigneux et judicieux des identifiants de variables,

• modularité du code et des tests,

• intelligibilité de la présentation de vos méthodes,

• votre contribution pertinente dans le test d'intelligibilité de la présentation de la méthode de deux autres binômes (fournissez une trace (1/2 page maximum) de vos demandes éventuelles de modification et fournissez votre appréciation finale : validé d'emblée, validé après demande de modifications réalisées, non validé mais prise en compte d'un certain nombre de demandes de modification, non validé car aucune demande de modification prise en compte ou prise en compte de façon pertinente)

• preuve de réalisation de tests, production des jeux d'essais et résultats (copies d'écran, fichiers texte – correctement présentés),

• facilité de maintenance du code produit,

• séparation claire des définitions de fonctions par des lignes du type ********** (OBLIGATOIRE),

• présence de commentaires en anglais correct dans les programmes et/ou choix d'identifiants « parlants »,

• *si nécessaire*, mention des préconditions et postconditions, en commentaire, immédiatement après le prototype de la fonction définie.

• Le nombre de trinômes est limité à 1.

• Une importante modulation selon la part réelle effective apportée au travail en binôme sera éventuellement appliquée.

• Les étudiants seront évalués selon les critères détaillés ci-dessus, ainsi que sur la vérification effective, le mardi 12 décembre 2023, que le logiciel élaboré fonctionne totalement ou partiellement.