Programmation et approche contemporaine du calcul scientifique (IAF6002)

Jean-Charles Grégoire, Jonathan Perrault

Objectif du cours

- Se familiariser avec :
  - Le langage python, de bonnes pratiques de programmation, des applications spécialisées pour leur domaine de recherche, des environnements de calcul scientifique de masse, etc.
- Et mettre en pratique ces apprentissages.
- À l'issue du cours, les étudiant.e.s auront acquis des bases solides pour traiter avec efficacité leurs données de recherche.

Ce que ce cours est

- Une initiation à des principes d'acquisition et l'analyse de données, à l'outillage, aux principes de travail sur les données pour la recherche
- Un projet de mise en œuvre de ces principes
  - Projet en lien avec les objectifs de recherche de chaque participant
- Un environnement d'échange et de discussion autour de chaque projet.
- Un parcours individualisé.

Ce que ce cours n’est pas

- Un cours sur l’IA, les statistiques, sur python (voire R), sur les bibliothèques X ou Y
  - On n’en finirait pas
- Un cours traditionnel (« magistral »)
  - Les étudiants vont faire le gros du travail
  - Les projets feront l’objet d’échanges collectifs
  - Les bonnes pratiques seront partagées

Nos défis

- Hétérogénéité
  - Du groupe-classe
  - Des projets/sujets/disciplines
  - Des connaissances
  - Du degré d'avancement des travaux
- Pas de temps pour une « mise à niveau »
- Peu de temps pour accomplir un (mini) projet
- Aller droit au but

Rencontre du 12 mars

- Présentation des enseignants/coordonnateurs
- Présentation des participants
  - Objectifs personnels
- Identification des livrables
- Échéancier
- Revue des objectifs du cours

Informatique

Outillage informatique

- Langage de programmation
  - Fédérateur
  - Ou intégré (script)
- Bibliothèques
  - Ou progiciel ‘tout en un’
- Visualisation
- Stockage
  - Source, Résultats, Code
- Documentation

Python

- Recommandé mais non obligatoire
- Facilite l'accès à une grande variété de bibliothèques spécialisées sur de nombreuses thématiques
- Disponible sur toutes les plateformes
  - Windows, Linux, MacOS
- Utilisable de différentes manières
  - Ligne de commande
  - Environnement intégré (p.ex. Anaconda + Jupyter Notebook)
  - Hybride (PyCharm)

Les environnements

- VSCode
- Colab (Google)
- Jupyter
- PyCharm

Bibliothèques Python

- Visualization
  - matplotlib, seaborn, mpl_toolkits
- Traitement de données
  - pandas, numpy
- IA
  - sklearn, tensorflow
- Statistiques
  - statsmodel, scipy
- Biostats
- …

Bibliothèques Python - Que noter?

- Incontournables
  - C’est la bonne manière de travailler: éviter de réinventer la roue
- Éprouvées
  - Le grand nombre d’usagers et de développeurs est un gage de qualité et d’efficacité
- Largement utilisées
  - La probabilité de problèmes résiduels est faible
- De nombreuses « recettes » disponibles en ligne
  - Mais pas toujours sur les versions que vous utilisez
  - Parfois mal comprises

Bibliothèques Python - Que noter?

- Complétudes: tentative d'être complètes par rapport à un sujet donné
- Souplesse: de nombreux paramètres pour traiter une diversité dans les problèmes traités
- Structure hiérarchique (packages, subpackages)
- Espace de nom isolé pour chaque bibliothèque
- Inclut objets, classes, fonctions, méthodes, valeurs, …

Bibliothèques Python - Que noter?

- Redondantes
  - Elles offrent des (trop?) alternatives dans la manière de faire des choses
- Complexes à maîtriser

Exemple: pandas

- pandas.errors: Custom exception and warnings classes that are raised by pandas.
- pandas.plotting: Plotting public API.
- pandas.testing: Functions that are useful for writing tests involving pandas objects.
- pandas.api.extensions: Functions and classes for extending pandas objects.
- pandas.api.indexers: Functions and classes for rolling window indexers.
- pandas.api.interchange: DataFrame interchange protocol.
- pandas.api.types: Datatype classes and functions.
- pandas.api.typing: Classes that may be necessary for type-hinting. T

Example API:

- pandas.read_excel(io, sheet_name=0, *, header=0, names=None, index_col=None, usecols=None, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skiprows=None, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, parse_dates=False, date_parser=<no_default>, date_format=None, thousands=None, decimal='.', comment=None, skipfooter=0, storage_options=None, dtype_backend=<no_default>, engine_kwargs=None)

Bibliothèques non-standard

- Ou, « emprunter du code » (d'un autre travail de doctorat)
  - Ne pas réinventer la roue, mais …
- On trouve sur le Web du code créé par d'autres personnes pour des travaux de recherche similaires
- Mais quelle est la « qualité » de ce code, et son « utilité pratique »?
  - Est-il compatible avec votre environnement?
  - Est-il documenté?
  - Pouvez-vous le comprendre?
  - Pouvez-vous le modifier?

En pratique

Comment procéder?

- Choisir un environnement de travail propre à sa plateforme de travail (ordinateur/OS/…).
- Respecter le code de conduite de cette plateforme
  - Ligne de commande ou environnement intégré
- Prendre le temps de se familiariser avec l'environnement: *crawl/walk/run; don't crun*.

Règles de base

- Prenez le temps de comprendre les effets du code que vous créez/empruntez
- Validez votre code, sur des petits exemples
- Documentez (tout)
- Suivez les « meilleures pratiques »
- Constituez un « cahier de recettes »
- Partagez les recettes entre vous
- Stabilisez votre environnement de travail
- Faites vos sauvegardes
- Utilisez la « sagesse de l'Internet », avec modération, et un sain scepticisme

Les « recettes »

- Des solutions DOCUMENTÉES à des problèmes ponctuels (utilisation de bibliothèques)
- Des exemples de code (p.ex. I/O)
- Vos « bonnes pratiques »
- Vos expériences d'utilisation

Programmer, pourquoi?

- De nombreux progiciels (commerciaux) intègrent déjà de nombreuses fonctions statistiques ou autres
- Mais ils n'offrent pas de souplesse pour intégrer de nouvelles techniques
- Et évoluent lentement
- De plus, ils peuvent contraindre un fonctionnement en silo:
  - Difficulté d'exporter des données / résultats pour autres traitement.

Programmer, mais ...

- Votre objectif n'est pas de devenir des codeurs professionnels
  - Ce qui implique, entre autres, de gérer la taille et la complexité du code
- Évolution:
  - Code linéaire
  - Fonctions/procédures
  - Classes/objets
  - Modules
  - Bibliothèques
  - ...
- Systématiser, abstraire, rassembler

Programmer, mais ...

- On commence simple,
  - en mode apprentissage et exploration.
- On fait une relecture périodique du code.
- On révise le code pour le restructurer périodiquement.
- On documente. (dans le code)
- On documente. (hors du code)

Documenter

- L’utilisation
  - Variables, procédure
- L’objectif
- Les techniques
  - Algorithmes, recettes …
- Les choix/alternatives
- Les révisions/changements

Les « bonnes pratiques »

- On adopte des pratiques stylistiques normalisées pour faciliter la lecture (PEP-8)
  - Indentation et espaces
  - Nom des procédures, constantes, variables, etc. (majuscule, minuscule, _)
- Commentaires
  - En ligne, en prélude, ou les « docstring » (triple quotations, plusieurs lignes)
- Suivi (logs)
- Environnements virtuels (python)
- …

Les erreurs

- La survenue d'erreurs est tout à fait naturelle et attendue lors de l'apprentissage
- La résolution de ces erreurs est vraiment l'occasion de comprendre comment fonctionne le langage et de devenir autonome dans sa pratique de celui-ci. Pensez à
  - Bien lire les logs, i.e. les sorties renvoyées par Python en cas d'erreur.
  - Chercher sur internet (de préférence en Anglais et sur Google). Par exemple, donner le nom de l'erreur et une partie informative du message d'erreur renvoyé par Python permet généralement de bien orienter les résultats vers ce que l'on cherche.
  - Cherchez de préférence dans des forums respectés, comme StackOverflow, où vous pouvez également poser vos questions.
  - Détaillez bien le problème rencontré de sorte à ce que les utilisateurs du forum puissent le reproduire et trouver une solution.
- Lisez les documentations officielles (de Python et des différents packages), généralement exhaustives, même si leur lecture peut être aride.
- Elles permettent notamment de bien comprendre la manière d'utiliser les différents objets. Par exemple pour les fonctions : ce qu'elles attendent en entrée, les paramètres et leur type, ce qu'elles renvoient en sortie, etc.

Recherche ouverte …

- La recherche scientifique implique la possibilité de vérifier, et de pouvoir reproduire des résultats.
- Donc une clarté sur
  - Méthode exploitée
  - Moyens utilisés
- Ceci s'applique tant à la production de données/échantillons qu'à l'analyse de propriétés
- Pensez à diffuser le code
  - D'où l'importance accrue de ce qui a été dit précédemment.

Notre contrat

- Vous vous engagez à choisir
  - Un environnement de travail
  - Une discipline de travail
  - Des pratiques de codage et de documentation
- Et à vous y conformer pour le reste du projet
- Ainsi qu'à partager votre code
- Il y aura évaluation par les pairs

À méditer

- On ne maîtrise que ce qu'on met en pratique
- Compromis à trouver entre maîtrise et efficacité
  - Vous n'allez pas devenir des informaticiens
- Travaillez dans une perspective minimaliste
- Mais en gardant néanmoins les bonnes pratiques en tête
- Votre défi est de réaliser vos objectifs de recherche, pas de maîtriser l'art de Python.

Les échanges d’information

Échanges?

- Les outils informatiques ont typiquement une manière de stocker l'information dans des fichiers qui leur est propre:
  - Xls(x)
- Des formats plus génériques, mais plus pauvres, existent également
  - CSV
- Des langages de programmation peuvent offrir leurs propres solutions
  - Pickling (pour Python)
- Des formats peuvent être adoptés de manière plus générale (standards de fait)
  - JSON
- Les outils « libres » (open) offrent souvent des passerelles de et vers différents formats.
  - read_excel()

De plus

- Format « lisible » ou codé
- Format comprimé
- Format chiffré
- Format autodescriptif

Échange, ou sauvegarde?

- S’agit-il d’exporter de l’information vers un autre outil?
- S’agit-il de sauvegarder des données pour un traitement subséquent?
- S’agit-il d’exporter une partie de l’information pour l’inclure dans un document?
- S’agit-il de partager des données avec d’autres groupes de recherche?
- D’un travail collaboratif entre plusieurs personnes?

Différentes approches sont possibles

- Selon les besoins
  - Sauvegarder d'abord, comprimer ensuite
  - Sauvegarder d'abord, chiffrer ensuite
- Mais attention de s'entendre sur les algorithmes/normes à utiliser.
- Penser aussi aux bases de données
  - Individuelle ou partagée

Et enfin … les SAUVEGARDES

- Comment ne pas faire mention des sauvegardes?
  - Du code
  - Des données
  - Des documents
- Quitte à conserver des révisions
  - Différentes approches sont possibles
    - Archives chronologiques
    - Outils de gestion des révisions
    - Numérotation/datation
    - …

En marge de cette question

- Sur combien de machines différentes travaillez-vous?
- Ces machines ont-elles un rôle dédié, ou partagé?
- Comment intégrez-vous l'information de ces différentes machines?
  - Mécanisme de partage
  - Garantie d'intégrité
- Plusieurs approches sont possibles
  - Travail à distance sur la même machine
  - Utilisation de ressources externes comme Calcul Québec
  - …

Calcul Québec

- Accès gratuit ou payant à des larges ressources de calcul
  - Dont processeurs spécialisés pour le DL.
- À utiliser pour de gros problèmes
- Votre superviseur doit avoir un accès et faire une requête pour vous donner un compte
- Environnement très « linux »
- Également des formations

Formations CQ (exemples)

- Estimer le stockage, la mémoire et le temps pour ses tâches de calcul
- Comprendre quand choisir un processeur CPU ou GPU pour ses tâches de calcul
- Apprendre l'analyse et la visualisation de données avec Python (sur 2 jours)
- Explorer et nettoyer ses données avec OpenRefine

Questions résiduelles

- Travaillez-vous seul ou en équipe?
  - Intra-lab ou inter-lab?
  - Intra-institution ou inter-institution?
- Quelqu'un dépend-il de votre travail?
- Vous avez des contraintes complémentaires sur les mécanismes et pratiques de partage, synchronisation des activités et mises à jour, etc.
- Ceci se planifie.