Projet Traitement de Données
1A ENSAI 2022

Benjamin Girault

1 Avant Propos

Le *Projet Traitement de données* s'adresse aux étudiants en 1ère année à l'ENSAI. Chaque équipe projet sera constitué de 3 étudiants pour travailler sur ce projet. Ce projet permet avant tout d'approfondir et de mettre en pratique les connaissances acquises en informatique en général et plus particulièrement en Python. Vous devrez mettre en avant le côté *Programmation Orientée Objet* du langage Python. Votre code devra être bien structuré, et bien documenté.

Pour chaque trinôme, quatre séances encadrées sont consacrées à l'avancement et au suivi du projet. Attention, pour chacune de ces séances de 3h, un encadrant devra s'occuper de plusieurs trinômes en même temps. Une part du travail lors de ces séances se fera donc en autonomie. Des créneaux dans votre emploi du temps vous sont réservés pour vous permettre d'avancer en autonomie en dehors des séances encadrées.

La création d'un programme informatique est souvent structurée en 4 phases :

1. **Spécification des besoins :** analyser et reformuler le problème posé afin de rédiger le **cahier des charges**.
2. **Conception globale :** définir l'architecture globale du programme via une **analyse descendante** par exemple.
3. **Conception détaillée :** décrire la structure interne des composants utilisés dans la conception globale.
4. **Implémentation :** transcrire la conception détaillée dans un langage de programmation, **Python** ici.

En coordination avec l'encadrant, vous devrez organiser votre projet en sous-tâches réparties chronologiquement et/ou par sous-groupe. Chaque séance débute par un point d'avancement de chacune des sous-tâches et, si nécessaire, par une mise à jour des tâches. Lors de la première séance encadrée, vous validerez avec votre encadrant les spécifications de votre programme, sa conception et l'organisation prévisionnelle des tâches. Vous devez donc venir à cette séance avec un (pseudo-)cahier des charges illustrant le fruit de votre réflexion concernant ces sujets. Le cahier des charges résultant de cette première séance sera soumis sur Moodle au plus tard le 26 avril 2022 à 23h59. **Attention :** ce cahier des charges est important pour bien commencer la première séance de travail encadrée. Il n'est pas demandé un cahier des charges parfait, mais plutôt suffisamment de pistes de réflexion sur qui utilisera votre programme et comment (diagramme de cas d'utilisation), quelles sont les fonctionnalités de votre programme, ou comment une exécution se déroule. Un premier diagramme de classe sera attendu pour la 2e séance encadrée.

Vous trouverez sur Moodle les données que vous devez traiter. Le rapport doit être rédigé en LaTeX(un modèle a été déposé sur Moodle, vous pouvez l'utiliser librement). La date limite pour rendre le rapport est fixée au 22 mai 2022 à 23h59, et le 29 mai 2022 à 23h59 pour le code. La date limite pour le suivi personnel est le 24 mai 2022 à 23h59. La date limite pour le code sera communiquée ultérieurement.

Les soutenances sont programmées les [à définir] 2022. Une présentation avec démonstration est demandée. Vous aurez 30 minutes : 20 minutes de présentation et démonstration, 10 minutes de questions. Les attendus pour le cahier des charges, le rapport, le code et la soutenance sont détaillés dans le tableau 1.

Vous êtes libres d'organiser votre travail au sein de votre équipe. Cependant, voici quelques conseils :

Item	Attendu / Contenu
Cahier des charges ($\leqslant 5$ pages) **Pour la 1ère séance**	• Rédigez de façon claire et propre • Décrivez "la maquette" de votre application : Que doit-il se passer lors de l'exécution ? • Décrivez les fonctionnalités que vous coderez
Suivi personnel ($\leqslant 2$ pages) **Pour la 4ème séance**	• Rôle dans l'équipe • Tâches réalisées **Un par étudiant** • Tâches en cours de réalisation
Code	• Votre programme doit fonctionner • Vos codes doivent être cohérent avec ce que vous décrivez dans le rapport • La quantité et la qualité des fonctionnalités sont importantes pour la validation • Le code doit utiliser l'héritage dans les parties préparation et modélisation des données • Le code doit proposer des actions pour effectuer les taches de préparation et modélisation • Le code doit être découpé en modules (fichiers), et sous-paquets (dossiers) • Effectuez des tests unitaires sur (au moins) un des modules que vous choisissez • Choisissez un module que vous commenterez de façon détaillée pour générer une documentation automatique
Rapport ($\leqslant 25$ pages)	• Rédigez de façon claire et propre. Afin de rédiger un document LaTeX, vous pouvez utiliser le logiciel TeXstudio ou des outils en ligne tel que `overleaf.com` • Énoncez les fonctionnalité du cahier des charges qui ont été implémentées. Dans le cas contraire, expliquer pourquoi • Un chapitre sera dédié à la gestion des données (import, stockage, transformation, types supportés, export) • Présentez l'architecture de votre application et des classes. Il est indispensable d'utiliser des outils d'UML (diagramme de classes pour la deuxième séance). Pour cela, vous pouvez utiliserez des outils en ligne comme `https://www.draw.io/`, ou hors ligne comme `dia` • Les différents diagrammes ainsi que les images doivent être clairs et lisibles • L'architecture dans votre rapport doit être cohérente avec ce que vous codez • Annexe optionnelle : la documentation générée automatique à partir des commentaires dans le code (vous pourrez vous limiter à une partie intéressante de cette documentation)
Soutenance	• Une démonstration de votre application • Une explication de l'architecture de votre code. Vous n'avez que 20mn, donc vous pouvez vous focaliser sur un point précis • Les pistes d'amélioration (architecture, gestion de projet, ...)

TABLEAU 1 – Les Attendus

— Pour le développement collaboratif de code, rien de mieux qu'un système de gestion de version, qui permettra de garder un historique des changements permettant de revenir en arrière si besoin, et une trace de l'activité de chacun. Il vous est conseillé d'utiliser Git pour cela. Consultez la page Moodle pour les détails. Vous pourrez également utiliser Git pour versionner vos rapports (vous pourrez, par exemple, créer deux dossier `src/` et `doc/` à la racine de votre dépôt Git).

— Un espace de discussion est disponible sur Moodle. En particulier, consultez cet espace lorsque vous avez des questions, et formulez vos questions dans cet espace pour que les réponses soient partagées.

2 Données

Le thème général est la relation entre climat et énergie. Deux jeux de données vous sont proposés sur ce thème. Le premier concerne le climat et inclus des données météorologiques des principales stations en France. Le second concerne l'énergie avec la consommation d'électricité des régions de France.

2.1 Données Météorologiques

Source Ce jeu de données est disponible sur le site https://donneespubliques.meteofrance.fr/?fond=produit&id_produit=90&id_rubrique=32. Les données pour 2013 à mars 2022 sont disponible sur Moodle (onglet *Données* de la page du cours *Projet Traitement de Données*). Vous pouvez télécharger directement des mises à jour des données si vous le souhaitez.

Contenu Le site officiel propose un fichier CSV compressé (`.csv.gz`) par mois. Inspirez-vous du Listing 1 pour charger ce type de fichier en Python. Notez en particulier que les fichiers `.csv.gz` ne sont pas décompressés, et qu'ils sont décompressés à la volée par Python. Chaque ligne correspond à une station et un instant. L'échelle de temps est de 3h entre deux lignes pour une même station. Les variables (colonnes) sont documentés dans le PDF `parametres-inclus-dans-les-fichiers-de-donnees-synop_283.pdf`, disponible sur la page de Météo France, et sur Moodle.

```python
import gzip
import csv

# Dossier où se trouve le fichier  :
folder = ''
filename = 'synop.201301.csv.gz'
data = []
with gzip.open(folder + filename, mode='rt') as gzfile :
    synopreader = csv.reader(gzfile, delimiter=';')
    for row in synopreader :
        data.append(row)
```

Listing 1 – Code pour lire un fichier `.csv.gz` du sujet. Chaque ligne de la liste `data` contient une liste des entrées de la ligne correspondante du CSV.

Métadonnées La description des stations est incluse dans un fichier séparé `postesSynop.csv`. Cette description associe un identifiant de station à son nom et ses coordonnées géographiques. Il fait partie du jeu de données de Météo France. Vous trouverez sur Moodle deux version : une identique à Météo France, et une avec le nom de la région à laquelle appartient la station. Cette deuxième version pourra être générée par votre programme (voir le bonus en section 6).

2.2 Données de Consommation Électrique

Source Ce jeu de données est disponible sur https://opendata.reseaux-energies.fr/explore/dataset/consommation-quotidienne-brute-regionale/information/. La version `.json` a été chargée et compressée en `.gz` avant upload sur Moodle.

Contenu Ce fichier est au format JSON (*JavaScript Object Notation*) qui est essentiellement un dictionnaire encodé au format texte (vous pouvez utiliser le code listé dans Listing 2 pour lire un tel fichier en Python). Ici, le fichier json est un tableau de dictionnaires, chacun correspondant à une entrée du jeu de donnée. Chacun de ces dictionnaires à un certain nombre de champs documenté dans tableau 2a, dont `fields` qui contient les données à proprement parler, documentées dans tableau 2b. Écrit en Python, on aura `data[0]['fields']['region']` qui contient le nom de la région de la première entrée du jeu de données.

```python
import gzip
import json

# Dossier où se trouve le fichier  :
folder = ""
filename = "consommation-quotidienne-brute-regionale.json.gz"
with gzip.open(folder + filename, mode='rt') as gzfile :
    data = json.load(gzfile)
```

Listing 2 – Code pour lire le fichier JSON du sujet.

3 Activités sur les Données

Le but de ce projet est d'implémenter un ensemble d'opérations sur les données précédentes pour en faire l'analyse. Ces opérations devront pouvoir être combinées pour créer un *pipeline* de traitement de données. En d'autres termes, votre programme chargera un ou plusieurs fichiers, les transformera suivant une ou plusieurs opérations de transformation, et produira un nouvel ensemble de données plus facilement utilisable pour une étude statistique. Deux buts spécifiques devront être atteints en utilisant les possibilités offertes par la programmation orientée objet :

1. **Réutilisation du Code :**
   - Réutilisation du code entre les opérations implémentées
   - Combinaison de briques (classes / instances) simple de transformation
   - Conception d'un nouveau pipeline facilité par l'agencement intuitif de ces briques
2. **Extensibilité du Code :**

Nom	Description
`datasetid`	"consommation-quotidienne-brute-regionale"
`recordId`	Un identifiant de l'entrée
`fields`	Les données
`record_timestamp`	La date de saisie des données

(a) Entrée du fichier json.

Nom	Description
`date_heure`	Date - Heure
`date`	Date
`heure`	Heure
`code_insee_region`	Code INSEE région
`region`	Région
`consommation_brute_gaz_grtgaz`	Consommation brute gaz (MW PCS 0°C) - GRTgaz
`statut_grtgaz`	Statut - GRTgaz
`consommation_brute_gaz_terega`	Consommation brute gaz (MW PCS 0°C) - Teréga
`statut_terega`	Statut - Teréga
`consommation_brute_gaz_totale`	Consommation brute gaz totale (MW PCS 0°C)
`consommation_brute_electricite_rte`	Consommation brute électricité (MW) - RTE
`statut_rte`	Statut - RTE
`consommation_brute_totale`	Consommation brute totale (MW)

(b) Entrée de `fields`.

TABLEAU 2 – Variables des consommation régionale d'électricité.

— L'écriture de nouvelles opérations est facilitée

— L'utilisation de votre programme avec d'autres données est facilité par sa conception

Vous argumenterez pourquoi votre conception facilite la réutilisation avec d'autres données, mais vous n'en ferez pas la démonstration.

Le reste de cette section donne de plus amples détails, et indique un ensemble minimal d'opérations à implémenter. À noter que l'évaluation se focalisera sur la conception de votre programme et non ses fonctionnalités non requises. En d'autres termes, implémenter plus d'opérations n'aura pas d'influence directe sur l'évaluation, mais seulement indirecte, si ces implémentations donnent lieu à une meilleure conception.

3.1 Transformations

Ce type d'opération prend en entrée un jeu de données et produit un autre jeu de données. Par extension, on considérera que l'opération de chargement d'un jeu de données est une transformation (transformation de la chaîne de caractères contenant l'emplacement du fichier de données en un vrai jeu de données).

Vous implémenterez les transformations suivantes :

— **Chargement d'un jeu de données**

— **Fenêtrage :** Sélection des données sur une période temporelle

— **Sélection de variables :** Par exemple, on ne souhaite garder que les températures

— **Agrégation spatiale :** Passage d'une granularité régionale à nationale

— **Combinaison de jeux de données (jointure) :** Par exemple, construction du jeu de données du nombre de consommation hebdomadaire d'une région et de sa température moyenne

— **Moyenne glissante**

— **Centrage**

— **Normalisation (variable centrée réduite)**

3.2 Paquets Autorisés

Vous pourrez utiliser `numpy`, `matplotlib`, `scipy`, mais pas `pandas`. Pour les autres paquets ne faisant pas partie de Python, demandez confirmation avant de les utiliser.

4 Exemples de Questions

Pour fixer les idées, voici une courte liste de questions qu'on souhaiterait étudier avec ces données, et pour lesquels vous créerez des pipeline produisant un nouveau jeu de données plus adapté à de telles études :

— Y a-t-il une relation entre température et consommation électrique ?

— Le vent joue-t-il un rôle dans cette consommation ?

— Peut-on observer des tendances à long terme pour la consommation électrique régionale ou nationale ?

— Observe-t-on des disparités régionales pour la relation entre température et consommation ?

Vous pourrez ajouter d'autres questions à votre rapport, d'autant plus si elles mettent en valeur la conception de votre programme.

À noter que l'étude statistiques des données n'est pas demandée et ne sera pas évaluée, ne perdez donc pas de temps sur ce point. En particulier, nous n'attendons pas de réponses aux questions ci-dessus. Si vous voulez rapidement lancer l'exécution d'outils plus avancés, vous pourrez utiliser une bibliothèque tierce, telle que `scipy` – mais cette partie ne sera pas évaluée.

5 Affichage / Sortie

Votre programme devra permettre de sauver le résultat de son exécution. Il devra sauver dans un fichier `.csv` (ou `.csv.gz`) ce résultat. Vous veillerez à ce que l'implémentation future de formats de fichier additionnels soit aisée. À noter que le résultat de l'exécution de votre programme peut être considéré comme un jeu de données.

De plus, vous pourrez implémenter des moyens simple d'afficher visuellement (sur des graphiques) les information contenues dans le jeu de données générées. Vous pourrez par exemple utiliser le paquet `matplotlib` (`https://matplotlib.org/`) pour des séries temporelles, des nuages de points, ou autres graphiques classiques, ou le module d'affichage sur un fond de carte disponible sur Moodle. Celui-ci pourra être utilisé inchangé, ou le modifié, mais une figure plus raffinée n'aura pas d'influence directe sur l'évaluation (comme pour les opérations, si la conception de votre programme est positivement modifiée, alors le raffinement de l'affichage aura un impact indirect sur l'évaluation).

6 Bonus : Obtention de l'Association Région-Station Météo

Moodle contient un fichier `.csv` associant à chaque station sa région d'appartenance. Pour ce bonus, vous utiliser l'API Nominatim, et en particulier sa commande `Reverse` documentée ici : `https://nominatim.org/`

`release-docs/develop/api/Reverse/` pour obtenir des informations pour n'importe quelles coordonnées géographiques, dont la région administrative.

Attention : Nominatim requiert un usage raisonnable de ce service. Afin de pas se faire bloquer par ce service web, vous vous assurerez que vous n'effectuez pas plus de 1 requête toutes les 5s. Pour palier à cette limite, vous implémenterez un système de cache, avec les régions déjà obtenues sauvées dans un fichier `.csv`, de telle sorte que ces régions ne soient calculées qu'une seule fois.