**TELECOM Paris** **MDI720**
**2021/2022** **Anne Sabourin**
**contributeurs: F. Portier, J. Salmon**

**TP : Introduction: Python, Numpy, Pandas, *et al.***

# Quelques remarques avant de démarrer

On va utiliser `Jupyter notebook` durant cette séance. Pour cela choisissez la version Anaconda sur les machines de l'école. Quelques points importants à retenir :

- Chargements divers -

```
import math                          # importe un package
import numpy as np                   # importe un package sous un nom particulier
from sklearn import linear_model     # importe tout un module
from os import mkdir                 # importe une fonction
```

- Utilisation de l'aide -

```
help(mkdir)                          # pour obtenir de l'aide sur mkdir
linear_model.LinearRegression?       # pour obtenir de l'aide sur LinearRegression
```

- Versions de package, localisation des fonctions -

```
print(np.__version__)                # obtenir la version d'un package
from inspect import getsourcelines   # obtenir le code source de fonctions
getsourcelines(linear_model.LinearRegression)
```

Rem: Pour tous les traitements numériques, on utilisera `numpy` (pour la gestion des matrices notamment) et `scipy`.

# 1 Introduction à Python, Numpy et Scipy

1) Écrire une fonction `nextpower` qui calcule la première puissance de 2 supérieure ou égale à un nombre $n$ (on veillera à ce que le type de sortie soit un `int`, tester cela avec `type` par exemple).
2) En partant du mot contenant toutes les lettres de l'alphabet, générer par une opération de *slicing* la chaîne de caractère `cfilorux` et, de deux façons différentes, la chaîne de caractère `vxz`.
3) Afficher le nombre $\pi$ avec 9 décimales après la virgule.
4) Compter le nombre d'occurrences de chaque caractère dans la chaîne de caractères `s="HelLo WorLd!!"`. On renverra un dictionnaire qui à chaque lettre associe son nombre d'occurrences.
5) Écrire une fonction de codage par inversion de lettres [1] : chaque lettre d'un mot est remplacée par une (et une seule) autre. On se servira de la fonction `shuffle` sur la chaîne de caractères contenant tout l'alphabet pour associer les lettres codées.
6) Calculer $2\prod_{k=1}^{\infty}\frac{4k^2}{4k^2-1}$ efficacement. On pourra utiliser `time` (ou `%timeit`) pour déterminer la rapidité de votre méthode. Proposer une version avec et une version sans boucle (utilisant `Numpy`).

1. aussi connu sous le nom de code de César.

7) Créer une fonction `quicksort` qui trie une liste, en remplissant les éléments manquants dans le code suivant. On testera que la fonction est correcte sur l'exemple `quicksort([-2, 3, 5, 1, 3])` :

```
def quicksort(ll):
    """ a sorting algorithm with a pivot value"""
    if len(ll) <= 1:
        return ll
    else:
        TODO  # pivot = last element of the list ll.
        less = []
        greater = []
        for x in ll:
            if x <= pivot:
                TODO # append 'x' to 'less'
            else:
                TODO # append 'x' to 'greater'
        return TODO # concatenate quicksort(less), pivot and quicksort(greater)
```

Indices : la longueur d'une liste est donnée par `len(l)` deux listes peuvent être concaténées avec `l1 + l2` et `l.pop()` retire le dernier élément d'une liste.

8) Sans utiliser de boucles `for` / `while` : créer une matrice $M \in \mathbb{R}^{5\times 6}$ aléatoire à coefficients uniformes dans $[-1,1]$, puis remplacer une colonne sur deux par sa valeur moins le double de la colonne suivante. Remplacer enfin les valeurs négatives par 0 en utilisant un masque binaire.

9) Créer une matrice $M \in \mathbb{R}^{5\times 20}$ aléatoire à coefficients uniformes dans $[-1,1]$. Tester que $G = M^\top M$ est symétrique et que ses valeurs propres sont positives ou nulles (on parle de alors de matrice positive). Attention à la tolérance machine (erreurs d'arrondi). Quel est le rang de $G$ ?
Aide : on utilisera par exemple `np.allclose, np.logical_not, np.all` pour les tests numériques.

## 2 Galton Watson database

Le terme *régression* a été introduit par Sir Francis Galton (cousin de C. Darwin) alors qu'il étudiait la taille des individus au sein d'une descendance. Il tentait de comprendre pourquoi les grands individus d'une population semblaient avoir des enfants d'une taille plus petite, plus proche de la taille moyenne de la population ; d'où l'introduction du terme "régression". Dans la suite on va s'intéresser aux données récoltées par Galton.

1) Récupérer les données du fichier https://perso.telecom-paristech.fr/sabourin/mdi720/GaltonsHeightData.csv (voir aussi leur description ici [2] :et charger les avec `Pandas`. On utilisera `read_csv` pour cela et on transformera les tailles en cm [3], en arrondissant sans chiffre après la virgule.
2) Combien de données manquantes y-t-il dans cette base de données ? Enlever si besoin les lignes ayant des données manquantes.
3) Afficher sur un même graphe un estimateur de la densité (on utilisera une méthode à noyaux avec un noyau gaussien) de la taille des pères en bleu, et de celles des mères en orange.
4) Afficher la taille du père en fonction de la taille de la mère pour les $n$ observations figurant dans les données. Ajouter la droite de prédiction obtenue par la méthode des moindres carrés (avec constante et sans centrage/normalisation).
5) Afficher un histogramme du nombre d'enfants par famille.
6) Créer une colonne supplémentaire appelée `'MidParents'` qui contient la taille du « parent moyen », et valant `('Father'+ 1.08 * 'Mother')/2`.

Pour la $i^{\text{e}}$ observation, on note $x_i$ la taille du parent moyen et $y_i$ la taille de l'enfant. On se base sur le modèle linéaire suivant : $y_i = \theta_0 + \theta_1 x_i + \varepsilon_i$ et on suppose que les variables $\varepsilon_i$ sont centrées, indépendantes et de même variance $\sigma^2$ inconnue.

2. https://www.randomservices.org/random/data/Galton.html
3. pour cela on pourra consulter la description des données proposées en http://www.math.uah.edu/stat/data/Galton.html

7) Estimer $\theta_0$, $\theta_1$, par $\hat{\theta}_0$, $\hat{\theta}_1$ en utilisant la fonction `LinearRegression` de `sklearn`, puis vérifier numériquement [4] les formules vues en cours pour le cas unidimensionnel

$$\hat{\theta}_0 = \overline{y}_n - \hat{\theta}_1 \overline{x}_n, \qquad \hat{\theta}_1 = \frac{\sum_{i=1}^n (x_i - \overline{x}_n)(y_i - \overline{y}_n)}{\sum_{i=1}^n (x_i - \overline{x}_n)^2}.$$

On fera attention aux normalisations utilisées pour la variance qui peuvent changer selon les packages.

8) Calculer et visualiser les valeurs prédites $\hat{y}_i = \hat{\theta}_0 + \hat{\theta}_1 x_i$ et les $y_i$ sur un même graphique. On affichera de couleurs différentes les données correspondant aux garçons et celles correspondant aux filles.

9) Visualiser un estimateur de la densité des résidus $r_i = y_i - \hat{y}_i$. L'hypothèse de normalité est-elle crédible selon vous ? Calculer ensuite $\alpha_g$ (resp. $\alpha_f$) les proportions de garçons (resp. de filles) dans la population. On ajoutera ensuite sur le graphique précédent, les fonctions $\alpha_g p_g$ et $\alpha_f p_f$, avec $p_g$ (resp. $p_f$) les densités des résidus pour les garçons (resp. pour les filles).

# 3 Introduction à Pandas, Matplotlib, etc.

On pourra commencer par consulter le tutoriel (en anglais) :
http://pandas.pydata.org/pandas-docs/stable/tutorials.html

- Charger des données -

On utilise la base de données **Individual household electric power consumption Data Set**, que l'on pourra télécharger depuis https://perso.telecom-paristech.fr/sabourin/mdi720/household_power_consumption.txt.

On s'intéresse aux grandeurs `Global_active_power` et `Sub_metering_1`.

10) Charger la base. Les valeurs manquantes sont : soit des blancs, soit des '?'. On utilisera donc l'option `na_values = ['?', '']` dans `pd.read_csv`.
Ensuite détecter et dénombrer le nombre de lignes ayant des valeurs manquantes.

11) Supprimer toutes les lignes avec des valeurs manquantes.

12) Modifier la variable `Sub_metering_1` en la multipliant par 0.06.

13) Utiliser `to_datetime` et `set_index` pour créer un DataFrame (on prendra garde au format des dates internationales qui diffère du format français).

14) Afficher le graphique des moyennes journalières entre le 1er janvier et le 30 avril 2007. Proposer une cause expliquant la consommation fin février et début avril. On pourra utiliser en plus de `matplotlib` le package `seaborn` pour améliorer le rendu visuel.

On ajoute des informations de température pour cette étude : les données utiles étant disponibles ici https://perso.telecom-paristech.fr/sabourin/mdi720/TG_STAID011249.txt [5]. Ici les températures relevées sont celles d'Orly (noter cependant qu'on ne connaît pas le lieux de relève de la précédente base de données).

15) Charger les données avec `pandas`, et ne garder que les colonnes `DATE` et `TG`. Diviser par 10 la colonne `TG` pour obtenir des températures en degrés Celsius. Traiter les éléments de température aberrants comme des NaN.

16) Créer un DataFrame `pandas` des températures journalières entre le 1er janvier et le 30 avril 2007. Afficher sur un même graphique ces températures et la séries `Global_active_power`.

## Pour aller plus loin

- http://blog.yhat.com/posts/aggregating-and-plotting-time-series-in-python.html
- http://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-tutor2-python-pandas.pdf

4. On pourra utiliser par exemple `np.isclose`
5. on peut aussi trouver d'autres informations sur le site http://eca.knmi.nl/dailydata/predefinedseries.php