
TP N° 1 : Prise en main de jupyter notebook

Objectifs du TP : savoir lancer `jupyter notebook`, lancer / sauvegarder un notebook, utiliser les premières commandes `Python` standard, importer une librairie.

Quelques conseils de base : Pour chaque fiche de TP, nous utiliserons un script `jupyter notebook`. Il est conseillé de créer un répertoire HLMA408, puis un sous-répertoire pour chaque TP (*e.g.*, TP1, TP2, TP3). Dans un répertoire TP1, vous stockerez donc :

- le sujet de TP au format PDF,
- les fichiers de données (qui y seront téléchargés, si possible automatiquement),
- le fichier `TP1-squelette.ipynb` qui contient les commandes (pratiquement) pré-remplies correspondant au TP, et que vous pouvez télécharger sur le site du cours.

Consignes :

Pour ce travail vous devez déposer un unique fichier sur le moodle du cours HLMA408. Vous rendrez un unique fichier d'extension `ext=ipynb`. Attention, vous devrez veiller à ce que le format soit correct : si le fichier ne peut pas être ouvert par le correcteur avec `jupyter-notebook`, la note sera de zéro.

Pour faciliter la correction, le nom du fichier devra respecter le format suivant :

`nom_fichier = tp_note1_hlma408_gr_nbgr_prenom_nom.ipynb`


le tout en minuscule et sans accent ni espace. Vous remplirez votre nom, prénom, le numéro de groupe qui vous concerne (remplacer `nbgr` par C, D ou E) de manière adéquate¹. Un point de malus sera appliqué pour les fichiers dont le format est erroné.

Vous devez charger votre fichier sur Moodle, avant le mardi 25/02/2020, 23h55. La note totale est sur 20 points, répartis comme suit :

- qualité des réponses aux questions : 14 pts,
- qualité de rédaction et d'orthographe : 1 pt,
- qualité des graphiques (légendes, couleurs) : 2 pts
- qualité d'écriture du code (noms de variable clairs, commentaires, code synthétique, etc.) : 1 pt
- Rendu reproductible et absence de bug : le code doit s'exécuter sur la machine du correcteur sans manipulation de sa part (par exemple le correcteur n'est pas supposé aller chercher les fichiers sur internet, les enregistrer, etc.). On veillera donc à ce que le chargement des bases de données soit automatisé : 2 pts

Les personnes qui n'auront pas soumis leur devoir sur Moodle avant la limite obtiendront zéro.

1 Utiliser jupyter notebook

Selon l'environnement que vous utilisez, veuillez vous reporter à l'aide correspondant à votre système d'exploitation ( : *operating system*, *OS*). Pour ceux qui utilisent les machines des salles TP, les indications concernant le chargement de `jupyter notebook` sont détaillées dans le polycopié du cours, Section 2.3.1 disponible ici :

<http://josephsalmon.eu/enseignement/Montpellier/HLMA310/IntroPython.pdf>

- 1) Télécharger le fichier `TP1-squelette.ipynb` dans votre espace dans votre répertoire HLMA408/TP1²
- 2) Sur les machines de la faculté des sciences vous pouvez **lancer jupyter notebook** de la manière suivante : cliquer sur l'icône **Anaconda3**, puis dans le terminal taper la commande `jupyter-notebook`.

1. Exemple : Joseph Salmon est dans le groupe C, son TP s'appelle `tp_note1_hlma408_gr_C_joseph_salmon.ipynb`
2. n'utiliser **JAMAIS** d'espace dans le nom d'un fichier ou d'un dossier !

- 3) Dans votre explorateur internet apparaît un environnement **jupyter notebook** qui vous permet de naviguer et de vous déplacer là où vous avez sauvegardé votre fichier. Cliquez alors sur le nom du fichier **TP1-squelette.ipynb** pour l'ouvrir.
- 4) Vérifier quelle version de Python vous avez (sur les machines de l'université cela devrait être 3.7.3), en exécutant :

```
from platform import python_version
print(python_version())
```

- 5) **Ajoutez une nouvelle cellule** tout en haut du **jupyter notebook** : soit en cliquant sur “+” soit en appuyant sur les touches “Esc” puis “ctrl+b” sur le clavier (sous Linux).
- 6) **Ajouter un titre de niveau 1** dans une cellule sous format markdown (en utilisant l'onglet “Cell” ou “Cellule” puis “Cell Type” ou “Type de Cellule” et en choisissant “Markdown”) ³. **(0.25 pt)**
Ce titre sera formé de la manière suivante :

```
# TP1 - HLMA408: Prénom Nom
```

où vous remplacerez “Nom” et “Prénom” par votre prénom et votre nom de famille.

Pour obtenir quelques rudiments de markdown, vous pouvez consulter le lien :

<http://www.boiteaoutils.info/2013/02/ecrire-tout-simplement-introduction/>

- 7) **Sauvegarder le notebook** sous le nom **tp_note1_hlma408_gr_nbgr_prenom_nom.ipynb**, où de nouveau vous remplacerez **prenom_nom** par votre nom de famille et votre prénom (sans majuscule). Vous pouvez soit utiliser le menu “File/save as”(ou “Fichier/enregistre sous”), ou bien cliquez sur le haut du fichier à côté de l'icône jupyter et remplacer le mot (par défaut pour un nouveau notebook il est écrit “Untitled”, mais ici son nom était “TP1-Squelette”) et cliquez sur l'icône de sauvegarde.
- 8) Tester l'**auto-complétion** de la manière suivante : exécuter la cellule de code que vous avez créée précédemment, puis dans une nouvelle cellule de type code taper **pyth** et appuyer sur la touche “tab” du clavier (celle à gauche de la lettre “a” sur un clavier azerty). Vous devriez alors avoir accès à un menu déroulant et pouvoir retrouver la fonction **python_version** sans avoir à taper le nom entier.

On veillera dans la suite à créer une cellule markdown pour chaque question, en utilisant un titre markdown de niveau 2, obtenu en tapant la commande suivante (par exemple pour la Question 1)


```
## Question 1:
```

On veillera selon que l'on demande du code ou des explications, à utiliser une cellule de code ou une cellule markdown pour la réponse à chaque question.

- 9) Comparer la différence d'affichage quand on utilise un #, ##, ### dans une cellule markdown.

2 Télécharger une base de données

2.1 La fonction read_csv

- 10) Pour utiliser la table de données ( : *data frame*) nommée **babies23.data** utilisée dans le cours, téléchargez le fichier dans le même répertoire où vous avez enregistré votre notebook. Vous pouvez le faire comme dans le cours (en ligne de commande) ou bien à la main en utilisant l'url du cours.
- 11) Vérifiez avec la fonction **ls** ⁴ que le fichier téléchargé se trouve bien dans le répertoire local dans lequel vous travaillez. Ceci est obtenu en exécutant la commande suivante :

```
ls
```

3. On peut faire cette opération en utilisant le menu déroulant, tout en haut, à côté des icônes

4. (**ls** est l'abréviation de “listing”)

- 12) Afin d'importer ces données dans votre notebook, on a besoin d'utiliser des fonctions additionnelles qui se trouvent dans la librairie : **pandas**. La première ligne de code permet de télécharger la librairie **pandas** sous l'acronyme **pd**. La deuxième ligne utilise la fonction **read_csv** de la librairie **pandas**, pour importer les données dans un dataframe nommé **df_babies**.

```
import pandas as pd
df_babies = pd.read_csv("babies23.data", skiprows=38, sep='\s+')
```

- 13) Pour lire l'aide de la fonction **read_csv** et comprendre les arguments possibles de la fonction, tapez :

```
pd.read_csv?
```

- 14) Utiliser la commande “Restart & run all” (🇫🇷 : *redémarrer & lancer tout*) de l'onglet “Kernel” (🇫🇷 : *noyau*). Quelle est la différence avec “Restart & clear output” (🇫🇷 : *redémarrer & effacer la sortie*) ?

- 15) Ouvrir le fichier **babies23.data** ⁵ (par exemple avec **gedit** ou un autre éditeur de texte) et parcourir le début du fichier pour comprendre brièvement ce que contient cette base de données. On pourra aussi consulter le fichier **babies.readme.txt** ⁶ pour plus de détails.

Rem. : pour plus d'indications sur la base de données voir les notebooks des premiers cours.

- 16) Afficher les 7 premières lignes du tableau de données babies avec la fonction **head**. **(0.25 pt)**
17) Afficher les 9 premières lignes du tableau de données babies. **(0.25 pt)**

2.2 Accéder aux lignes et colonnes de la table de données

Une table de données est un objet de type **dataframe** en **pandas**. Les différentes variables sont rangées dans des colonnes alors que les lignes correspondent aux différentes observations. Les lignes et les colonnes sont numérotées à partir de 0 (la convention de **Python** diffère de celle de **R** : on commence à partir de 1 en **R**).

Les colonnes ont des noms que l'on obtient ici avec la commande **df_babies.columns**. Il est donc important de nommer les objets (ici les colonnes) de manière pertinente pour les retrouver aisément. On peut extraire une sous-table de données ou bien un vecteur colonne depuis une table de données. Pour cela on utilise la commande **NomDeLaTable['NomDeLaColonne']**.

- 18) Dans le fichier texte, il y avait deux colonnes (c'est-à-dire deux variables) qui s'appelaient **wt**. Comment **pandas** a-t-il résolu le conflit ? **(0.25 pt)**
19) Afficher la colonne '**wt**' de la base. **(0.25 pt)**

Pour extraire les observations (ici en ligne dans le dataframe, suivant une convention classique en statistique) on peut utiliser la commande **iloc**. Ainsi la commande **df_babies.iloc[0]** extrait la première ligne du dataframe.

- 20) Observez ce que font les commandes : **df_babies.iloc[0]**, **df_babies.iloc[-1]**, **df_babies.iloc[:8]**, **df_babies.iloc[-5:]** et en déduire comment extraire les lignes des numéros 100 à 110 (c'est-à-dire les individus d' id 1552 à 1688 inclus). **(0.5 pt)**

3 Mettre en forme les colonnes du jeu de données

Les commandes de **Python** et de **pandas** s'adaptent aux types d'objets sur lesquels on les applique. Il est donc fondamental de mettre correctement en forme la table de données (🇬🇧 : *data frame*). Pour cela, il faut particulièrement :

- re-coder les valeurs manquantes avec le code spécial **nan** (acronyme de *not a number*)
- potentiellement changer les unités pour qu'elles soient dans le système international (mettre dans le système métrique international les tailles, les masses, etc.)



5. <http://josephsalmon.eu/enseignement/datasets/babies23.data>

6. <http://josephsalmon.eu/enseignement/datasets/babies.readme.txt>

La fonction `replace` est particulièrement efficace pour remplacer certaines valeurs par une autre. Ainsi la commande suivante charge la librairie `numpy` et remplace les 99 par des `nan` :

```
import numpy as np # import de numpy
df_babies['ht'].replace(99, np.nan, inplace=True) # gère les données manquantes
```


21) Exécuter la commande précédente.

En résumé on vient de remplacer, dans la colonne `ht` (représentant la taille ( : *height*) en pouces ⁷ ( : *inches*)) le nombre 99 par des `nan`. ⁸

22) Décrire ce que fait la suite de deux blocs suivants et donner l'intérêt que l'introduction de `is_preprocessing_done` peut avoir? (0.5 pt)

```
is_preprocessing_done = False
```

```
if is_preprocessing_done is False:
    df_babies['ht'] = df_babies['ht'] * 2.54
    is_preprocessing_done = True
```

23) Adapter la question précédente pour obtenir les masses et tailles des mères et des pères en cm et en kg (tout en traitant bien sûr les données manquantes). On admettra que les masses des parents sont données en livres ( : *pounds*). (1 pt)

24) Lancez la commande suivante qui permet de supprimer toutes les données manquantes (ce qu'on fera pour l'instant).

```
df_babies.dropna(inplace=True)
```

25) Créer un nouveau dataframe, appelé `df_babies_tailles_masses`, contenant uniquement les colonnes du data frame `df_babies` qui correspondent aux masses et tailles des pères et des mères. On pourra utiliser l'extraction d'une liste de colonnes : par exemple `df_babies[['ht', 'dht', 'sex']]` extrait la base de données composée de trois colonnes 'ht', 'dht' et 'sex'. (0.5pt).

4 Statistiques descriptives

On présente ci-dessous quelques fonctions pour résumer des données.

Commandes pour obtenir des résultats numériques

Commande	Description succincte
<code>df_babies.describe()</code>	Calcule des statistiques résumées de <code>df_babies</code>
<code>df_babies['ht'].mean()</code>	Calcule la moyenne de <code>df_babies['ht']</code>
<code>df_babies['ht'].std()</code>	Calcule l'écart-type de <code>df_babies['ht']</code>
<code>df_babies['ht'].median()</code>	Calcule la médiane de <code>df_babies['ht']</code>
<code>df_babies['ht'].quantile(0.99)</code>	Calcule les quantiles de <code>df_babies['ht']</code>

26) Testez ces fonctions sur la taille des pères et la taille des mères.

27) Pour ces deux variables, donner la valeur des quantiles suivants : 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99. (0.25pt)

7. 1 inch = 2.54 cm

8. 99 code pour les tailles manquantes : personne ne mesure $99 \times 2.54 = 251.46$ cm dans l'étude. Plus de détails sont disponibles sur les valeurs manquantes de chaque colonne dans l'entête du fichier `babies23.data`.

5 Affichage graphique

On utilisera surtout les librairies `matplotlib` et `seaborn` (une extension graphiquement plus élaborée que `matplotlib`).

28) Lancer la commande :

```
import matplotlib.pyplot as plt
import seaborn as sns
```

On résume ci-dessous un certain nombre de commandes utiles pour la visualisation.

Commandes pour obtenir des graphiques

Commande	Description succincte
<code>plt.figure(figsize=(5,5))</code>	créer une figure d'une taille 5×5
<code>plt.hist(df_babies['ht'], bins=18)</code>	créer un histogramme de la variable <code>ht</code>
<code>ax = sns.kdeplot(df_babies['ht'])</code>	trace un estimateur à noyaux de la densité
<code>plt.scatter(df_babies['ht'], df_babies['dht'])</code>	affiche un nuage de points

On pourra regarder l'aide et les exemples du cours pour améliorer :

- les légendes
- les titres
- le nom des axes
- les couleurs
- les styles de marqueurs
- etc.

- 29) Afficher un histogramme de la taille des mères avec 12 boîtes (🇬🇧 : *bins*). (1pt)
- 30) Afficher un estimateur de la taille des mères avec un estimateur à noyaux de la densité (🇬🇧 : *kernel density estimator, KDE*). (1pt)
- 31) Afficher côte à côte⁹ deux boîtes à moustache de la taille des pères et des mères. (1pt)
- 32) Afficher côte à côte deux violons de la taille des pères et des mères. (1pt)
- 33) Afficher un nuage de points avec pour abscisse (axe des x) la taille des pères et pour ordonné (axe des y) la taille des mères. En particulier on proposera un moyen de distinguer visuellement les couples (x, y) qui apparaissent souvent. (2pt)
- 34) Afficher un widget qui sur deux graphiques alignés l'un au-dessus de l'autre (avec `subplot` donc) affiche la masse des pères et des mères, et dont le curseur contrôle le nombre de boîtes (🇬🇧 : *bins*). (2pts)
- 35) Afficher un widget qui sur deux graphiques alignés l'un au-dessus de l'autre (avec `subplot` donc) affiche la taille des pères et des mères, et dont le curseur contrôle la taille de la fenêtre de lissage (🇬🇧 : *bandwidth*). (2pts)

6 Pour aller plus loin

Tutos/Vidéos de Jake Vanderplas : [Reproducible Data Analysis in Jupyter](https://reproducible-data-analysis-in-jupyter/)¹⁰

9. pour cela on pourra considérer la commande suivantes : `fig, axs = plt.subplots(nrow, ncol, figsize=(8, 5))`

10. <http://jakevdp.github.io/blog/2017/03/03/reproducible-data-analysis-in-jupyter/>