SY09 Printemps 2020
TP 01 — Manipulation de données

1 Chargement d'un jeu de données

Les jeux de données sont communément stockés dans des fichiers textes au format dit « csv » (*comma separated value*). Il s'agit d'un format décrivant un tableau individus–variables : une ligne liste les caractéristiques d'un individu, séparées par une virgule ; et une colonne liste les valeurs d'une variable pour tous les individus. Dans certains fichiers, la première ligne est parfois une ligne d'en-tête (ou `header`) spécifiant le nom de chacun des prédicteurs. Parfois, la première colonne n'est pas un prédicteur mais un identifiant ou un nom d'individu qui n'est pas un prédicteur. Les fichiers « csv » ont plusieurs variantes, le séparateur (la virgule pour le fichier « csv ») peut changer. La plupart du temps, le séparateur est une virgule, une espace, un point virgule ou une tabulation.

Pour charger des données représentant un tableau individus–variables, on utilise la bibliothèque `pandas`. On la charge avec l'instruction suivante

```
In [1]: import pandas as pd
```

Pour charger un fichier `csv`, on utilise la fonction `pd.read_csv` en spécifiant le chemin du fichier `csv` à charger.

(1) Charger le fichier `data/sy02-p2016.csv` dans la variable `X`.

Pour contrôler le bon chargement des données, on peut vérifier le nombres de caractéristiques ainsi que le nombre d'individus avec l'attribut `shape`, le type des caractéristiques avec la méthode `info`.

(2) Vérifier qu'il y a 296 individus et 11 caractéristiques.

(3) En utilisant les options de chargement `sep`, `index_col` et `header`, charger les fichiers suivants :
- `data/sy02-p2016-2.csv`
- `data/sy02-p2016-3.csv`
- `data/sy02-p2016-4.csv`
- `data/sy02-p2016-5.csv`

Vérifier qu'ils contiennent les mêmes informations que le premier jeu de données.

2 Conversion de types

Lors du chargement d'un fichier texte, si le type de la colonne n'est pas spécifié avec l'argument `dtype`, `Pandas` essaie de deviner le type de chaque prédicteur. Les types les plus utilisés sont les suivants
- `np.float64` : Correspond à une variable quantitative continue
- `np.int64` : Correspond à une variable quantitative discrète (les entiers naturels)
- `bool` : Correspond à une variable binaire
- `object` : Lorsqu'aucune des classes ci-dessus ne convient, le type générique `object` est utilisé
- `category` : Correspond à une variable qualitative à plusieurs modalités. `Pandas` ne convertit jamais automatiquement vers ce type, il faut le faire *a posteriori*.

```
In  [2]: from io import StringIO

         pd.read_csv(StringIO("0\n1.4"), header=None).dtypes
Out [2]: 0    float64
         dtype: object
In  [3]: pd.read_csv(StringIO("0\n1"), header=None).dtypes
Out [3]: 0    int64
         dtype: object
In  [4]: pd.read_csv(StringIO("T\nF"), header=None).dtypes
Out [4]: 0    object
         dtype: object
In  [5]: pd.read_csv(StringIO("True\nFalse"), header=None).dtypes
Out [5]: 0    bool
         dtype: object
In  [6]: pd.read_csv(StringIO("Vrai\nFaux"), header=None).dtypes
Out [6]: 0    object
         dtype: object
```

Lorsque le type n'est pas correctement détecté, on peut le corriger manuellement en faisant appel à la méthode `astype(<type>)`.

Pour les variables catégorielles, le type n'est pas encore défini. Il faut donc d'abord le définir

```
ects_type = pd.CategoricalDtype(categories=["R", "G", "B"])
```

et l'utiliser ensuite avec `astype(<type>)`

```
X.col = X.col.astype(ects_type)
```

Si le type n'est pas réutilisé pour d'autres prédicteurs, on peut directement le créer en même temps que la colonne.

```
X.col = pd.Categorical(X.col, categories=["R", "G", "B"])
```

Si les modalités sont ordonnées, on peut le spécifier avec l'argument `ordered`.

(4) Corriger le type de chaque prédicteur présent dans le fichier `data/data.csv`.

3 Transformation

Même lorsque le jeu de données est nettoyé et qu'il ne présente plus d'erreurs manifestes, il est souvent nécessaire de transformer certains prédicteurs voire la structure elle-même du jeu de données.

Lorsque la donnée sous-jacente est de type chaine de caractères, `Pandas` fournit un nombre important de fonctions pour extraire l'information utile. On peut par exemple utiliser les *slices* :

```
In  [7]: X = pd.read_csv("data/sy02-p2016.csv")
         X.nom
```

```
Out [7]: 0        Etu1
         1        Etu2
         2        Etu3
         3        Etu4
         4        Etu5
                 ...
         291    Etu292
         292    Etu293
         293    Etu294
         294    Etu295
         295    Etu296
         Name: nom, Length: 296, dtype: object
In  [8]: X.nom.str[3:]
Out [8]: 0        1
         1        2
         2        3
         3        4
         4        5
                ...
         291    292
         292    293
         293    294
         294    295
         295    296
         Name: nom, Length: 296, dtype: object
```

Il faut utiliser la méthode `str` pour avoir accès à toute ces fonctions d'extraction. Pour lister toutes ces fonctions, on pourra exécuter l'instruction

```
dir(X.nom.str)
```

(5) Le prédicteur `Semestre` du jeu de données présent dans le fichier `data/effectifs.csv` contient des données de la forme `SemestreXXXXX`. En utilisant les *slices* extraire la donnée `XXXXX`.

La donnée est maintenant de la forme « `SDDDD` » avec `S` le semestre « A » ou « B » et `DDDD` l'année. Cependant, cette donnée n'est toujours pas exploitable.

(6) Créer deux autres colonnes contenant respectivement le semestre et l'année. On pourra utiliser la fonction `assign`.

Il est souvent souhaitable de factoriser plusieurs colonnes stockant des données ayant la même signification en deux colonnes seulement : une colonne stocke le nom de la colonne et l'autre la valeur correspondante. Un exemple classique est présent dans la table 1.

Pour réaliser cette opération avec `Pandas`, on utilise la fonction `melt`.

```
In  [9]: X1 = pd.DataFrame(
             dict(
                 Person=["Bob", "Alice", "Steve"],
                 Age=[32, 24, 64],
                 Weight=[128, 86, 95],
                 Height=[180, 175, 165],
             )
         )
         X1.melt(id_vars=["Person"])
```

Table 1 – Représentation « wide » et « long »

(a) Format « wide »

Person	Age	Weight	Height
Bob	32	128	180
Alice	24	86	175
Steve	64	95	165

(b) Format « long »

Person	Variable	Value
Bob	Age	32
Bob	Weight	128
Bob	Height	180
Alice	Age	24
Alice	Weight	86
Alice	Height	175
Steve	Age	64
Steve	Weight	95
Steve	Height	165

```
Out [9]:   Person variable  value
        0     Bob      Age     32
        1   Alice      Age     24
        2   Steve      Age     64
        3     Bob   Weight    128
        4   Alice   Weight     86
        5   Steve   Weight     95
        6     Bob   Height    180
        7   Alice   Height    175
        8   Steve   Height    165
```

On peut renommer les colonnes `variable` et `value` en utilisant les arguments `var_name` et `value_name`.

(7) Convertir le jeu de données précédent au format « long », enlever les effectifs inexistants et convertir en nombre entier.

(8) Convertir le jeu de données iris en format « long ». On pourra charger le jeu de donnée iris avec les instructions suivantes.

```
import seaborn as sns
iris = sns.load_dataset("iris")
```

(9) Scinder la colonne des longueurs/largeurs des sépales/pétales en deux colonnes.

4 Jeu de données babies

Le jeu de données contenu dans le fichier babies23.data est constitué de 1236 bébés décrits par 23 variables.

(10) Charger le jeu de données et sélectionner les colonnes `wt`, `gestation`, `parity`, `age`, `ht`, `wt.1`, `smoke`, `ed` que l'on renommera en `bwt`, `gestation`, `parity`, `age`, `height`, `weight`, `smoke`, `education`.

(11) Faites l'histogramme des durées de gestation en jours. Que remarquez-vous ?

D'une manière générale dans ce jeu de données, lorsque la valeur de certains prédicteurs est inconnue une valeur prédéfinie est utilisée :
- Pour la colonne `bwt`, on utilise 999
- Pour la colonne `gestation`, on utilise 999

- Pour la colonne `age`, on utilise 99
- Pour la colonne `height`, on utilise 99
- Pour la colonne `weight`, on utilise 999
- Pour la colonne `smoke`, on utilise 9
- Pour la colonne `education`, on utilise 9

(12) Remplacer toutes ces valeurs prédéfinies par `np.nan`.

(13) Pour la variable `smoke`, la documentation du jeu de données dit

```
smoke: does mother smoke?
0=never,
1=smokes now,
2=until current pregnancy,
3=once did, not now,
9=unknown
```

Recoder la variable `smoke` de manière à ce que la modalité « 1 » soit recodée en `Smoking` et les autres modalités en `NonSmoking`.