Projet : Manipulation des données avec Pandas

In [ ]:

```
```

Pandas est une librairie Python spécialisée dans l'analyse des données qui a beaucoup de succès. Dans ce projet pratique, nous nous intéresserons surtout aux fonctionnalités de manipulations de données qu'elle propose. Un objet de type "DataFrame" (tableau de données), permet de réaliser de nombreuses opérations de pré-traitements, de filtrage, de nettoyage, de construction de nouvelles variables à partir des existentes, etc.,... préalables à la modélisation statistique.

La librairie est très largement documentée. Il faut prendre le temps de consulter le Help et de s'exercer simplement ! Deux liens du Help sont incontournables, celui relative aux DataFrame (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html#pandas.DataFrame (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html#pandas.DataFrame)), celui relative aux Series que nous travaillerons également dessus (vecteur de données : http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html#pandas.Series (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html#pandas.Series)).

Librairie Pandas - Options et version

Il faut charger la librairie et, éventuellement, la configurer selon vos attentes. Pensez à vérifier votre version, elle doit être (non seulement la même sur tous les postes mais aussi la plus récente).

In [46]:

```
# 1è étape : il faut charger Pandas
import pandas as pd

# Vérifiez sa version ?
pd.__version__

# Quelle est la dernière version et MAJ la votre en ligne de commande (au cas où Pa
#pip install --upgrade pandas

# Rappelez les commandes qui permettent d'installer, MAJ, utiliser 1version moins r
# R installer ===>  pip install python
# MAJ ===> pip install --upgrade mylib
# utiliser ===> import mylib
# désinstaller ===> pip uninstall mylib

pip install pandas==XXX

# Modifiez le nb de lignes à afficher dans les print à 10. L'idée est d'éviter que
# à de multiples affichages de longs tableaux !
# R :
import pandas as pd


df = pd.read_table('sante.txt')
print (df[0:10])
# ou print (df.head(10))
```

```
   age      sexe typedouleur sucre  tauxmax angine  depression coeur
col9  \
0   70  masculin           D     A      109    non          24   pre
sence
1   67   feminin           C     A      160    non          16    ab
sence
2   57  masculin           B     A      141    non           3   pre
sence
3   64  masculin           D     A      105    oui           2    ab
sence
4   74   feminin           B     A      121    oui           2    ab
sence
5   65  masculin           D     A      140    non           4    ab
sence
6   56  masculin           C     B      142    oui           6   pre
sence
7   59  masculin           D     A      142    oui          12   pre
sence
8   60  masculin           D     A      170    non          12   pre
sence
9   63   feminin           D     A      154    non          40   pre
sence

   col10  col11  col13
0    NaN    NaN    NaN
1    NaN    NaN    NaN
2    NaN    NaN    NaN
3    NaN    NaN    NaN
4    NaN    NaN    NaN
5    NaN    NaN    NaN
```

```
6    NaN    NaN    NaN
7    NaN    NaN    NaN
8    NaN    NaN    NaN
9    NaN    NaN    NaN
```

In [74]:

```
# Modifiez le nb de colonnes à afficher dans les print à 10. L'idée est d'éviter qu
# par un grand nombre de col !
# R :

matable = pd.read_table('sante.txt')
#matable.iloc[:, :10]
print(matable[matable.columns[0:10]])


# Malgré le fait que ns avons fixé 1petit nb de col à afficher (à 10), les résultat
# En d'autres termes, l'idée est d'améliorer l'affichage ds la console en évitant u
# Activez l'option nécessaire à ce besoin ?
# R :
matable = pd.read_table('sante.txt')
matable.iloc[:, :10]
```

```
     age      sexe typedouleur sucre  tauxmax angine  depression coeur
col9  \
0     70  masculin           D     A      109    non          24   pre
sence
1     67   feminin           C     A      160    non          16    ab
sence
2     57  masculin           B     A      141    non           3   pre
sence
3     64  masculin           D     A      105    oui           2    ab
sence
4     74   feminin           B     A      121    oui           2    ab
sence
..   ...       ...         ...   ...      ...    ...         ...
...
265   52  masculin           C     B      162    non           5    ab
sence
266   44  masculin           B     A      173    non           0    ab
sence
267   56   feminin           B     A      153    non          13    ab
sence
268   57  masculin           D     A      148    non           4    ab
sence
269   67  masculin           D     A      108    oui          15   pre
sence

     col10  col11
0      NaN    NaN
1      NaN    NaN
2      NaN    NaN
3      NaN    NaN
4      NaN    NaN
..     ...    ...
265    NaN    NaN
266    NaN    NaN
267    NaN    NaN
268    NaN    NaN
269    NaN    NaN

[270 rows x 10 columns]
```

Out[74]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur col9	col10	col11
0	70	masculin	D	A	109	non	24	presence	NaN	NaN
1	67	feminin	C	A	160	non	16	absence	NaN	NaN
2	57	masculin	B	A	141	non	3	presence	NaN	NaN
3	64	masculin	D	A	105	oui	2	absence	NaN	NaN
4	74	feminin	B	A	121	oui	2	absence	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
265	52	masculin	C	B	162	non	5	absence	NaN	NaN
266	44	masculin	B	A	173	non	0	absence	NaN	NaN
267	56	feminin	B	A	153	non	13	absence	NaN	NaN
268	57	masculin	D	A	148	non	4	absence	NaN	NaN
269	67	masculin	D	A	108	oui	15	presence	NaN	NaN

270 rows × 10 columns

In [42]:

```
pd.read_table('sante.txt')
```

Out[42]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur col9	col10	col11
0	70	masculin	D	A	109	non	24	presence	NaN	NaN
1	67	feminin	C	A	160	non	16	absence	NaN	NaN
2	57	masculin	B	A	141	non	3	presence	NaN	NaN
3	64	masculin	D	A	105	oui	2	absence	NaN	NaN
4	74	feminin	B	A	121	oui	2	absence	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
265	52	masculin	C	B	162	non	5	absence	NaN	NaN
266	44	masculin	B	A	173	non	0	absence	NaN	NaN
267	56	feminin	B	A	153	non	13	absence	NaN	NaN
268	57	masculin	D	A	148	non	4	absence	NaN	NaN
269	67	masculin	D	A	108	oui	15	presence	NaN	NaN

270 rows × 11 columns

```
Après avoir réussi ces dernieres manip, créez 1script (startup.py) avec ces diff
commandes et indiquez son path à comme PYTHON_STARTUP.
L'idée est qu'à chaque lancement de Python, vos différents imports et options
soient activées automatiquement !
```

Chargement, structure DataFrame et description des

données

1DataFrame correspond à 1tableau (lignes x cols) que je noterai à partir de maintenant "df".
Concernant notre fichier "sante.txt" :
La 1è ligne correspond aux noms des vars. A partir de la 2è ligne, ns disposons des valeurs pr chaque enregistrement (individu).
Le caractère tabulation "\t" fait office de séparateur de cols.
Ns allons vérifier tt ça ds 1 1er tps en lignes de commandes Linux et ds un 2sd temps sur Python.

```
En ligne de commande Linux :

1) Créez 1répertoire "data" (où vs stockerez d'1façon permanente ts les objets que ns travaillerons dessus).
2) Télécharez à partir du google-drive-datartisan le fichier "sante.txt" (secteur de la sante).
# R : mkdir data
cd data
wget url:google-drive-datartisan/sante.txt

Dans le terminal linux (et non ds 1éditeur de texte), déterminez :
3) la taille du fichier, le nombre de lignes et affichez les 5 1è (lignes).
wc -l sante.txt
head -6 sante.txt

# L'idée est d'avoir un aperçu du fichier ET surtout de déterminer :
# Le fait s'il tiens en mémoire (données massives ou non),

# Le type de séparateur (tabulation, virgule, pipe,...),

# La présence ou non des noms des vars.

# Le tout pour préparer la commande Python de création de df (à partir de ce fichier) !
# R :
```

In [ ]:

```
# Refaites ces diff manip strictement à l'aide de commandes Python : création de ré
# R : indice import os, open()
import os
import requests

if not os.path.exists('data'):
    os.makedirs('data')

req = requests.get("mon_url")

with open('sante.txt', 'w') as fp:
    fp.write(req.content)

# Revenez sur le script "startup.py" et y mettre les commandes qui vs permettent de
# Créer le rép "data" (s'il n'existe pas) ?
# Se positinner dessus automatiquement (à chaque démarrage).
# R :
```

In [103]:

```
# Chargez le fichier en tant qu'objet DataFrame (nom = df_sante)
# R :

df_sante = pd.read_table("sante.txt")
print(df_sante)

# Vérifiez que le type obtenu est bien un 'pandas.core.frame.DataFrame'
# R :
type(df_sante)
```

```
     age      sexe typedouleur sucre  tauxmax angine  depression coeur
col9  \
0     70  masculin           D     A      109    non          24   pre
sence
1     67   feminin           C     A      160    non          16    ab
sence
2     57  masculin           B     A      141    non           3   pre
sence
3     64  masculin           D     A      105    oui           2    ab
sence
4     74   feminin           B     A      121    oui           2    ab
sence
..   ...       ...         ...   ...      ...    ...         ...
...
265   52  masculin           C     B      162    non           5    ab
sence
266   44  masculin           B     A      173    non           0    ab
sence
267   56   feminin           B     A      153    non          13    ab
sence
268   57  masculin           D     A      148    non           4    ab
sence
269   67  masculin           D     A      108    oui          15   pre
sence

     col10  col11  col13
0      NaN    NaN    NaN
1      NaN    NaN    NaN
2      NaN    NaN    NaN
3      NaN    NaN    NaN
4      NaN    NaN    NaN
..     ...    ...    ...
265    NaN    NaN    NaN
266    NaN    NaN    NaN
267    NaN    NaN    NaN
268    NaN    NaN    NaN
269    NaN    NaN    NaN

[270 rows x 11 columns]
```

Out[103]:

pandas.core.frame.DataFrame

Le type DataFrame est bien reconnu. Voyons maintenant sa structure.

Faire un print(df) où vs vérifiez à l'oeil nu, dans la console, ligne par ligne la structure du df n'a pas de sens (contexte Big Data) !
Croyez moi, ça ne va pas vs faire avancer (inversement, au bon entendeur !).
Ns préferons alors déterminer la dimension du df, afficher un bout des 1ères et dernières lignes (et comparer avec les résultats déjà obtenu avec le fichier ".txt").

In [84]:

```
# Déterminez les dimensions : nb de lignes et nb de colonnes ?
# R :

print(len(df_sante))

print(len(df_sante.columns))

# OU

df_sante.shape

# Vérifier que ce sont les bonnes dimensions avec le fichier ".txt"
# Yes good

# Rq : la ligne d'en-tête n'est pas comptabilisée dans le nombre de lignes !
```

```
270
11
```

In [86]:

```
# Affichez les premières lignes du jeu de données ?
# R :
df_sante.head(10)
```

Out[86]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur col9	col10	col11	col
0	70	masculin	D	A	109	non	24	presence	NaN	NaN	Na
1	67	feminin	C	A	160	non	16	absence	NaN	NaN	Na
2	57	masculin	B	A	141	non	3	presence	NaN	NaN	Na
3	64	masculin	D	A	105	oui	2	absence	NaN	NaN	Na
4	74	feminin	B	A	121	oui	2	absence	NaN	NaN	Na
5	65	masculin	D	A	140	non	4	absence	NaN	NaN	Na
6	56	masculin	C	B	142	oui	6	presence	NaN	NaN	Na
7	59	masculin	D	A	142	oui	12	presence	NaN	NaN	Na
8	60	masculin	D	A	170	non	12	presence	NaN	NaN	Na
9	63	feminin	D	A	154	non	40	presence	NaN	NaN	Na

In [87]:

```
# Affichez les dernières lignes du jeu de données ?
# R :
df_sante.tail(10)
```

Out[87]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur col9	col10	col11
260	58	feminin	C	A	172	non	0	absence	NaN	NaN
261	60	masculin	D	A	132	oui	24	presence	NaN	NaN
262	58	masculin	B	A	160	non	18	presence	NaN	NaN
263	49	masculin	B	A	171	non	6	absence	NaN	NaN
264	48	masculin	B	A	168	non	10	presence	NaN	NaN
265	52	masculin	C	B	162	non	5	absence	NaN	NaN
266	44	masculin	B	A	173	non	0	absence	NaN	NaN
267	56	feminin	B	A	153	non	13	absence	NaN	NaN
268	57	masculin	D	A	148	non	4	absence	NaN	NaN
269	67	masculin	D	A	108	oui	15	presence	NaN	NaN

```
Nous passons beaucoup de notre temps à charger des df et à afficher le head
(tail) pour vérifier que tt est OK !
Il vaut mieux en faire de ça 1fonc 1fois pr tte que vs appelez au besoin ?
```

In [ ]:

```
# Ecrivez cette fonc où vs gérez les exceptions (contexte script en production) ?
# R :
```

Les colonnes ?

In [94]:

```
# Lister les colonnes (les noms des vars) ?
# R :
df_sante.columns
```

Out[94]:

```
Index(['age', 'sexe', 'typedouleur', 'sucre', 'tauxmax', 'angine',
       'depression', 'coeur col9', 'col10', 'col11', 'col13'],
      dtype='object')
```

In [99]:

```
# Supposons que ce dernier affichage ne vous conviens pas (à l'horizontal, surtout
# Mettez en place 1boucle "for" qui vous permez d'afficher le résultat (à la vertic
# R :
list(df_sante)


for i in range (0,len(df_sante.columns)-1):
    print(df.columns[i])
    i+=1
```

```
age
sexe
typedouleur
sucre
tauxmax
angine
depression
coeur col9
col10
col11
```

In [106]:

```
# Quelle est le type de chaque colonne : int64, object,... ?
# R :

df_sante.dtypes
```

Out[106]:

```
age              int64
sexe            object
typedouleur     object
sucre           object
tauxmax          int64
angine          object
depression       int64
coeur col9      object
col10          float64
col11          float64
col13          float64
dtype: object
```

```
Il y'a combien type ? Que veut dire le type "object" ?
il y a 3 type : int64, object, float64
le type object veut dire un type référence prédéfinit en amont lors de la
création de la classe

Question à revenir dessus plus tard (après avoir travaillé sur les Series),
combien de type (en ligne de commande) ?
# R :
```

In [107]:

```
# Au lieu de taper ces 2 dernières commandes,
# il existe une commande qui vous permet de retrouver ces 2 derniers résultats (à l
# Laquelle ?
# R :
df_sante.dtypes
```

Out[107]:

```
age              int64
sexe            object
typedouleur     object
sucre           object
tauxmax          int64
angine          object
depression       int64
coeur col9      object
col10          float64
col11          float64
col13          float64
dtype: object
```

Pandas a tendance d'abuser de la mémoire lors de chargement des df.
Aussi, il est vivement recommandé de bien affecter les bons types à chaque variable ne serait-ce que pour éviter de faire des opérations douteuses, par exemple arithmétiques sur des vars de type "object".
Ds notre cas, vérifiez que notre fichier ne comporte pas de nombres avec des chiffres après la virgule ?
Malgré cela, Pandas a stocké les cols concernées en int64 au lieu de int8 (gaspillage) !

In [123]:

```
# Par souci d'optimisation, mettez les cols qui sont en int64 en int8 ?
# R :

df_sante["age"].astype(int)

df_sante.dtypes
```

Out[123]:

```
age              int64
sexe            object
typedouleur     object
sucre           object
tauxmax          int64
angine          object
depression       int64
coeur col9      object
col10          float64
col11          float64
col13          float64
dtype: object
```

In [ ]:

```
Q à traiter plus tard : Comment faire la même manip (sans relire le fichier) ?
```

```
Supposons que ns avons à travailler sur un fichier avec des nombres avec des
chiffres après la virgule (ou point pr spécifier un décimal (US Vs. EU)
Trouvez l'option ds pd.read_table() qui permet de le spécifier ?
# R :
```

Description des données

In [124]:

```
# Trouvez une commande pr présenter les statistiques descriptives de tt le df (min,
# R :

df_sante.describe()
```

Out[124]:

	age	tauxmax	depression	col10	col11	col13
count	270.000000	270.000000	270.000000	0.0	0.0	0.0
mean	54.433333	149.677778	10.500000	NaN	NaN	NaN
std	9.109067	23.165717	11.452098	NaN	NaN	NaN
min	29.000000	71.000000	0.000000	NaN	NaN	NaN
25%	48.000000	133.000000	0.000000	NaN	NaN	NaN
50%	55.000000	153.500000	8.000000	NaN	NaN	NaN
75%	61.000000	166.000000	16.000000	NaN	NaN	NaN
max	77.000000	202.000000	62.000000	NaN	NaN	NaN

```
Explication :
Certains indicateurs statistiques ne sont valables que pour :
# les variables numériques (ex. moyenne, min, etc. pour age, tauxmax,...),
# et inversemment pour les non-numériques (ex. top, freq, etc. pour sexe,
typedouleur, ...),
# d'où les NaN dans certaines situations.
Est-ce que ce résultat vs convient ?!
# Pr plus de clarté, ns souhaitons afficher les stats pr les vars num dans un 1er
tps (1è commande ?)
# Et ds un 2sd tps les vars de type objects (2è commande ?)
# Comment faire ?
```

In [137]:

```
# Stats desc pr les vars num ?
# R :

df_sante2 = df_sante.loc[:, df.dtypes == int]

df_sante2.describe()
```

Out[137]:

	age	tauxmax	depression
count	270.000000	270.000000	270.000000
mean	54.433333	149.677778	10.500000
std	9.109067	23.165717	11.452098
min	29.000000	71.000000	0.000000
25%	48.000000	133.000000	0.000000
50%	55.000000	153.500000	8.000000
75%	61.000000	166.000000	16.000000
max	77.000000	202.000000	62.000000

In [142]:

```
# Stats desc pr les vars non-num ?
# R :


df_sante3 = df_sante.loc[:, df.dtypes != int]

df_sante3.describe()
```

Out[142]:

	col10	col11	col13
count	0.0	0.0	0.0
mean	NaN	NaN	NaN
std	NaN	NaN	NaN
min	NaN	NaN	NaN
25%	NaN	NaN	NaN
50%	NaN	NaN	NaN
75%	NaN	NaN	NaN
max	NaN	NaN	NaN

Manipulation des variables

Accès aux variables

Il est possible d'accéder explicitement aux variables. Dans un premier temps, nous utilisons directement les noms des champs (les noms des variables, en en-tête de colonne).

In [144]:

```
# Accès à une colonne de votre choix ?
# R :

df_sante["age"].head(10)
```

Out[144]:

```
0    70
1    67
2    57
3    64
4    74
5    65
6    56
7    59
8    60
9    63
Name: age, dtype: int64
```

In [148]:

```
# Autre manière d'accéder à une colonne avec le ".  ?
# R :

df_sante.age.head(10)
```

Out[148]:

```
0    70
1    67
2    57
3    64
4    74
5    65
6    56
7    59
8    60
9    63
Name: age, dtype: int64
```

In [172]:

```
# Accéder à un ensemble de colonnes ?
# R :

df_sante4 = df_sante[['age','sexe']]

df_sante4.head(10)
```

Out[172]:

	age	sexe
0	70	masculin
1	67	feminin
2	57	masculin
3	64	masculin
4	74	feminin
5	65	masculin
6	56	masculin
7	59	masculin
8	60	masculin
9	63	feminin

In [174]:

```
# Une colonne est un vecteur (Series en terminologie Pandas)
# Affichage des premières valeurs
# R :
df_sante.columns

df_sante.iloc[:10,:1]
```

Out[174]:

	age
0	70
1	67
2	57
3	64
4	74
5	65
6	56
7	59
8	60
9	63

In [180]:

```
# Affichage des dernières valeurs
# R :
df_sante.age.tail(10)
```

Out[180]:

```
260    58
261    60
262    58
263    49
264    48
265    52
266    44
267    56
268    57
269    67
Name: age, dtype: int64
```

In [181]:

```
# Statistique descriptive d'une col. Pour plus de détails, voir :
# http://pandas.pydata.org/pandas-docs/stable/basics.html#summarizing-data-describe
# R :
df_sante.age.describe()
```

Out[181]:

```
count    270.000000
mean      54.433333
std        9.109067
min       29.000000
25%       48.000000
50%       55.000000
75%       61.000000
max       77.000000
Name: age, dtype: float64
```

In [182]:

```
# Calculer explicitement la moyenne, par ex col "age"
# R :
df_sante.age.mean()
```

Out[182]:

```
54.43333333333333
```

In [189]:

```
# Comptage des valeurs, par ex col "typedouleur"
# R :

#all
df_sante.typedouleur.count()
#distinct
df_sante.typedouleur.nunique()
```

Out[189]:

```
0      D
1      C
2      B
3      D
4      B
      ..
265    C
266    B
267    B
268    D
269    D
Name: typedouleur, Length: 270, dtype: object
```

In [190]:

```
# Un type Series est un vecteur, il est possible d'utiliser des indices
# Première valeur ?
# R :
df_sante.age[0]
```

Out[190]:

```
70
```

In [195]:

```
# 3 premières valeurs ?
# R :
df_sante.age.iloc[:3]
```

Out[195]:

```
0    70
1    67
2    57
Name: age, dtype: int64
```

In [201]:

```
# Triez les valeurs d'une variable de manière croissante ?
# R :
sorted(df_sante.age)

sort_value(df_sante.age)
```

Out[201]:

```
[29,
 34,
 34,
 35,
 35,
 35,
 37,
 37,
 38,
 39,
 39,
 39,
 40,
 40,
 40,
 41,
 41,
 41,
```

In [205]:

```
df_sante.sort_values('age')
```

Out[205]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur col9	col10	col11
214	29	masculin	B	A	202	non	0	absence	NaN	NaN
174	34	masculin	A	A	174	non	0	absence	NaN	NaN
138	34	feminin	B	A	192	non	7	absence	NaN	NaN
224	35	feminin	D	A	182	non	14	absence	NaN	NaN
81	35	masculin	D	A	130	oui	16	presence	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
15	71	feminin	D	A	125	non	16	absence	NaN	NaN
255	71	feminin	B	A	162	non	4	absence	NaN	NaN
4	74	feminin	B	A	121	oui	2	absence	NaN	NaN
73	76	feminin	C	A	116	non	11	absence	NaN	NaN
199	77	masculin	D	A	162	oui	0	presence	NaN	NaN

270 rows × 11 columns

In [206]:

```
# le tri peut être généralisé aux DataFrame
# par exemple : triez le df selon l'âge puis afficher les 1è lignes
# R :
df_sante.sort_values('age').head(10)
```

Out[206]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur col9	col10	col11	c
214	29	masculin	B	A	202	non	0	absence	NaN	NaN	
174	34	masculin	A	A	174	non	0	absence	NaN	NaN	
138	34	feminin	B	A	192	non	7	absence	NaN	NaN	
224	35	feminin	D	A	182	non	14	absence	NaN	NaN	
81	35	masculin	D	A	130	oui	16	presence	NaN	NaN	
193	35	masculin	D	A	156	oui	0	presence	NaN	NaN	
32	37	feminin	C	A	170	non	0	absence	NaN	NaN	
209	37	masculin	C	A	187	non	35	absence	NaN	NaN	
160	38	masculin	A	A	182	oui	38	presence	NaN	NaN	
77	39	feminin	C	A	179	non	0	absence	NaN	NaN	

Accès indicé aux données d'un DataFrame

On peut accéder aux valeurs du DataFrame via des indices ou plages d'indice. La structure se comporte alors comme une matrice. La cellule en haut et à gauche est de coordonnées (0,0).

Il y a différentes manières de le faire, l'utilisation de .iloc[,] constitue une des solutions les plus simples. N'oublions pas que Shape permet d'obtenir les dimensions (lignes et colonnes) du DataFrame.

In [211]:

```
# Accèdez à la valeur située en (0,0) ?
# R :
df_sante.iloc[:1,:1]
```

Out[211]:

	age
0	70

In [215]:

```
# Accédez à la valeur située en dernière ligne, première colonne ?
# indice : utilisez l'indiçage négatif
# R :
df_sante.iloc[-1:-2:-1,:1]
```

Out[215]:

	age
269	67

In [221]:

```
# Accédez la valeur située en dernière ligne, première colonne ?
# indice : shape[0] renvoie le nombre de lignes (1è dimension) qu'il faut réduire de
# sinon on déborde
# R :

df_sante.iloc[df_sante.shape[0]:-2:-1,:1]
```

Out[221]:

	age
269	67

In [224]:

```
# Accédez aux 5 premières valeurs de toutes les colonnes ?
# lignes => 0:5 (0 à 5 [non inclus])
# colonnes = : (toutes les colonnes)

df_sante.iloc[:5,:]
# OU
df_sante.head(5)
```

Out[224]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur col9	col10	col11	col
0	70	masculin	D	A	109	non	24	presence	NaN	NaN	Na
1	67	feminin	C	A	160	non	16	absence	NaN	NaN	Na
2	57	masculin	B	A	141	non	3	presence	NaN	NaN	Na
3	64	masculin	D	A	105	oui	2	absence	NaN	NaN	Na
4	74	feminin	B	A	121	oui	2	absence	NaN	NaN	Na

In [234]:

```
# Accéder aux 5 dernières lignes du df ? avec l'indiçage négatif, on le peut facile

#df_sante.tail()

df_sante.iloc[-1:-6:-1,:]
```

Out[234]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur col9	col10	col11
269	67	masculin	D	A	108	oui	15	presence	NaN	NaN
268	57	masculin	D	A	148	non	4	absence	NaN	NaN
267	56	feminin	B	A	153	non	13	absence	NaN	NaN
266	44	masculin	B	A	173	non	0	absence	NaN	NaN
265	52	masculin	C	B	162	non	5	absence	NaN	NaN

In [235]:

```
# Accédez aux 5 premières lignes et deux premières colonnes ?
df_sante.iloc[:5,:2]
```

Out[235]:

	age	sexe
0	70	masculin
1	67	feminin
2	57	masculin
3	64	masculin
4	74	feminin

In [251]:

```
# Accédez aux 5 1è lignes et colonnes 0, 1 et 4 ?
# indice : on a une liste d'indices en colonne
df_sante.iloc[:5,[0,1,4]]
```

Out[251]:

	age	sexe	tauxmax
0	70	masculin	109
1	67	feminin	160
2	57	masculin	141
3	64	masculin	105
4	74	feminin	121

In [256]:

```
# faites la même chose autrement ?
# indice : remarquez le rôle de 2 dans 0:5:2
df_sante.iloc[:,[0,1,4]].head()
```

Out[256]:

	age	sexe	tauxmax
0	70	masculin	109
1	67	feminin	160
2	57	masculin	141
3	64	masculin	105
4	74	feminin	121

Restrictions avec les conditions - Les requêtes

Nous pouvons isoler les sous-ensembles d'observations répondant à des critères définis sur les champs. Nous utiliserons préférentiellement la méthode .loc[,] dans ce cadre.

In [270]:

```
# Listez les individus présentant une douleur de type A

df_sante.query('typedouleur == "A"')
```

Out[270]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur col9	col10	col11
13	61	masculin	A	A	145	non	26	presence	NaN	NaN
18	64	masculin	A	A	144	oui	18	absence	NaN	NaN
19	40	masculin	A	A	178	oui	14	absence	NaN	NaN
37	59	masculin	A	A	125	non	0	presence	NaN	NaN
63	60	feminin	A	A	171	non	9	absence	NaN	NaN
64	63	masculin	A	B	150	non	23	absence	NaN	NaN
85	42	masculin	A	A	178	non	8	absence	NaN	NaN
87	59	masculin	A	A	145	non	42	absence	NaN	NaN
118	66	feminin	A	A	114	non	26	absence	NaN	NaN
143	51	masculin	A	A	125	oui	14	absence	NaN	NaN
158	56	masculin	A	A	162	non	19	absence	NaN	NaN
160	38	masculin	A	A	182	oui	38	presence	NaN	NaN
169	65	masculin	A	B	174	non	14	presence	NaN	NaN
170	69	masculin	A	B	131	non	1	absence	NaN	NaN
174	34	masculin	A	A	174	non	0	absence	NaN	NaN
198	69	feminin	A	A	151	non	18	absence	NaN	NaN
205	52	masculin	A	B	178	non	12	absence	NaN	NaN
210	59	masculin	A	A	159	non	2	presence	NaN	NaN
228	58	feminin	A	B	162	non	10	absence	NaN	NaN
229	52	masculin	A	A	190	non	0	absence	NaN	NaN

In [265]:

```
df_sante
```

Out[265]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur col9	col10	col11
0	70	masculin	D	A	109	non	24	presence	NaN	NaN
1	67	feminin	C	A	160	non	16	absence	NaN	NaN
2	57	masculin	B	A	141	non	3	presence	NaN	NaN
3	64	masculin	D	A	105	oui	2	absence	NaN	NaN
4	74	feminin	B	A	121	oui	2	absence	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
265	52	masculin	C	B	162	non	5	absence	NaN	NaN
266	44	masculin	B	A	173	non	0	absence	NaN	NaN
267	56	feminin	B	A	153	non	13	absence	NaN	NaN
268	57	masculin	D	A	148	non	4	absence	NaN	NaN
269	67	masculin	D	A	108	oui	15	presence	NaN	NaN

270 rows × 11 columns

In [34]:

```
# Nous constatons que l'on indexe avec un vecteur de booléens si on va dans le déta
print(df['typedouleur'] == "A")
```

```
0      False
1      False
2      False
3      False
4      False
       ...
265    False
266    False
267    False
268    False
269    False
Name: typedouleur, dtype: bool
```

Seules les observations correspondant à True sont repris par .loc[,]. Nous pouvons les comptabiliser :

In [277]:

```
# Comptez le nombre d'individus qui présente une douleur de type "A" ?
df_sante.query('typedouleur == "A"').count()

# ou

df_sante.query('typedouleur == "A"').shape(0)
```

Out[277]:

```
age            20
sexe           20
typedouleur    20
sucre          20
tauxmax        20
angine         20
depression     20
coeur col9     20
col10           0
col11           0
col13           0
dtype: int64
```

In [36]:

```
# pour un ensemble de valeurs de la même variable,
# nous utilisons isin()
print(df.loc[df['typedouleur'].isin(['A','B']),:])
```

```
     age      sexe typedouleur sucre  tauxmax angine  depression     c
oeur
2     57  masculin           B     A      141    non           3  pres
ence
4     74   feminin           B     A      121    oui           2   abs
ence
13    61  masculin           A     A      145    non          26  pres
ence
18    64  masculin           A     A      144    oui          18   abs
ence
19    40  masculin           A     A      178    oui          14   abs
ence
..   ...       ...         ...   ...      ...    ...         ...
...
262   58  masculin           B     A      160    non          18  pres
ence
263   49  masculin           B     A      171    non           6   abs
ence
264   48  masculin           B     A      168    non          10  pres
ence
266   44  masculin           B     A      173    non           0   abs
ence
267   56   feminin           B     A      153    non          13   abs
ence

[62 rows x 8 columns]
```

Des opérateurs logiques permettent de combiner les conditions. Nous utilisons respectivement : & pour ET, | pour OU, et ~ pour la négation.

In [279]:

```
# Listez les individus présentant une douleur de type A et angine == oui ?

df_sante.query('typedouleur == "A" and angine == "oui" ')
```

Out[279]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur col9	col10	col11
18	64	masculin	A	A	144	oui	18	absence	NaN	NaN
19	40	masculin	A	A	178	oui	14	absence	NaN	NaN
143	51	masculin	A	A	125	oui	14	absence	NaN	NaN
160	38	masculin	A	A	182	oui	38	presence	NaN	NaN

In [300]:

```
df_sante = df_sante.rename(columns = {'coeur col9': 'coeur_col9'})
```

In [302]:

```
# Liste les personnes de moins de 45 ans, de sexe masculin, présentant une maladie

#df_sante
df_sante.query(' age < 45 and sexe == "masculin" and coeur_col9 == "presence" ')
```

Out[302]:

	age	sexe	typedouleur	sucre	tauxmax	angine	depression	coeur_col9	col10	col11
40	40	masculin	D	A	181	non	0	presence	NaN	NaN
47	44	masculin	D	A	177	non	0	presence	NaN	NaN
50	42	masculin	D	A	125	oui	18	presence	NaN	NaN
81	35	masculin	D	A	130	oui	16	presence	NaN	NaN
147	40	masculin	D	A	114	oui	20	presence	NaN	NaN
160	38	masculin	A	A	182	oui	38	presence	NaN	NaN
182	41	masculin	D	A	158	non	0	presence	NaN	NaN
193	35	masculin	D	A	156	oui	0	presence	NaN	NaN
231	39	masculin	D	A	140	non	12	presence	NaN	NaN
237	43	masculin	D	A	120	oui	25	presence	NaN	NaN
252	44	masculin	D	A	153	non	0	presence	NaN	NaN

In [305]:

```
# On peut n'afficher qu'une partie des colonnes
# On définit la projection dans une liste
colonnes = ['age','sexe','coeur_col9','tauxmax']
# que l'on utilise en paramètre dans .loc[]
# pour la même restruction que précédemment
print(df_sante.loc[(df_sante['age'] < 45) & (df_sante['sexe'] == "masculin") & (df_
```

```
     age      sexe coeur_col9  tauxmax
40    40  masculin   presence      181
47    44  masculin   presence      177
50    42  masculin   presence      125
81    35  masculin   presence      130
147   40  masculin   presence      114
160   38  masculin   presence      182
182   41  masculin   presence      158
193   35  masculin   presence      156
231   39  masculin   presence      140
237   43  masculin   presence      120
252   44  masculin   presence      153
```

Calculs récapitulatifs - Croisement des variables

A la manière des tableaux croisés dynamiques (TCD) d'Excel, nous pouvons procéder à des croisements et opérer des calculs récapitulatifs, qui vont du comptage simple aux calculs statistiques mettent en jeu d'autres variables.

In [38]:

```
# Fréquences selon sexe et coeur (tri croisé) ?
# voir : http://pandas.pydata.org/pandas-docs/stable/generated/pandas.crosstab.html
```

In [39]:

```
# Le même tri croisé mais avec un pourcentage en ligne ?
# indice : ns pouvons demander un post-traitement après la commande précedente !
```

In [40]:

```
# Calculez la moyenne d'âge selon le sexe et la maladie ?
# indice : ns utilisons la fonction mean() de la classe Series de la librairie Pand
```

In [43]:

```
# une autre manière de faire avec la commande pivot_table() pour exactement le même
print(df.pivot_table(index=['sexe'],columns=['coeur'],values=['age'],aggfunc=pandas
```

```
               age
coeur       absence presence
sexe
feminin   54.582090    59.35
masculin  51.192771    56.04
```

L'utilisation de groupby() permet d'accéder aux sous-DataFrame associés à chaque item de la variable de regroupement. Il est dès lors possible d'appliquer explicitement d'autres traitements sur ces sous-ensembles de données.

In [46]:

```
# Scission des données selon le sexe
g = df.groupby('sexe')

# Calculer la dimension du sous-DataFrame associé aux hommes
print(g.get_group('masculin').shape)
```

```
(183, 8)
```

In [41]:

```
# Calculez la moyenne de l'âge chez les hommes.

```

In [48]:

```
# On peut appliquer différentes fonctions
# agg() permet de revenir sur quelque chose qui ressemble au crosstab()
print(g[['age','depression']].agg([pandas.Series.mean, pandas.Series.std]))
```

```
               age           depression
              mean       std       mean        std
sexe
feminin   55.678161  9.626144   8.885057  11.332630
masculin  53.841530  8.818189  11.267760  11.459408
```

In [49]:

```
# Nous pouvons itérer sur les groupes
for groupe in g:
    # groupe est un tuple
    print(groupe[0]) #étiquette du groupe
    # accès à la variable 'age' du groupe concerné
    print(pandas.Series.mean(groupe[1]['age']))
```

```
feminin
55.67816091954023
masculin
53.84153005464481
```

Construction de variables calculées

Les calculs sont vectorisés pour les vecteurs de type Series de Pandas. Ce qui évite de passer par des boucles fastidieuses pour manipuler les valeurs des vecteurs.

In [42]:

```
# Création d'une variable tauxnet (qui n'a aucune signification médicale)
# Utilisation de la libraire numpy (log = logarithme népérien)

```

In [43]:

```
# Ns cherchons à la concaténer au DataFrame
```

La construction d'une variable ex-nihilo est également possible. Par ex., nous souhaitons créer une indicatrice pour la variable sexe, 1 pour masculin, 0 pour féminin.

In [52]:

```
# Création d'une Série de 0 de la même longueur
# que notre DataFrame(nombre de lignes)
# nous utilisons la méthode de numpy pour cela
code = pandas.Series(numpy.zeros(df.shape[0]))
print(code.shape)
```

```
(270,)
```

In [53]:

```
#les "sexe = masculin" sont codés 1
#de fait, "sexe = feminin" est codé zéro puisque le
#vecteur a préalablement été créé avec des valeurs 0
code[df['sexe']=='masculin'] = 1
print(code.value_counts())
```

```
1.0    183
0.0     87
dtype: int64
```

In [54]:

```
#une autre solution plus simple, mais il faut connaître eq()
codebis = df['sexe'].eq('masculin').astype('int')
print(codebis.value_counts())
```

```
1    183
0     87
Name: sexe, dtype: int64
```

Graphiques

Passer par matplotlib permet de réaliser des graphiques performants (http://matplotlib.org/ (http://matplotlib.org/)). Mais il faut connaître les procédures de la librairie, ce qui nécessite un apprentissage supplémentaire qui n'est pas toujours évident.

Heureusement, Pandas propose des commandes simples qui encapsulent l'appel à ces procédures et nous simplifie grandement la vie. Il faut importer matplotlib pour que l'ensemble fonctionne correctement.

In [313]:

```
#indiquer que l'on veut voir apparaître les graphiques dans le notebook
#/!\ très important, sinon on ne verrait rien
#!matplotlib inline
#!pip install matplotlib
#importation de la librairie
import matplotlib as plt
```

In [316]:

```
# Tracer 1histogramme de l'âge ?
fig, ax = plt.subplots()
ax.hist(df_sante['age'], range=(0,5))
plt.show()
```

```
---------------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
 last)
<ipython-input-316-37e803e68052> in <module>
      1 # Tracer 1histogramme de l'âge ?
----> 2 fig, ax = plt.subplots()
      3 ax.hist(df_sante['age'], range=(0,5))
      4 plt.show()

AttributeError: module 'matplotlib' has no attribute 'subplots'
```

In [45]:

```
# Tracer 1density plot ?
```

In [46]:

```
# Tracer l'histogramme de l'âge selon le sexe ?
```

In [47]:

```
# Comparaison des distributions avec un boxplot ?
```

In [48]:

```
# Tracer 1scatterplot : age vs. tauxmax ?
```

In [49]:

```
# Tracer 1scatterplot (age vs. tauxmax) en distinguant les points ?
# (niveau de gris) selon les valeurs de dépression
```

In [51]:

```
# Tracer 1 scatterplot (age vs. tauxmax) en distinguant les points selon les valeur
# indice : nécessite un recodage de coeur - ici en 0/1
# afficher le graphique en spécifiant la couleur (blue = 0, green = 1)
```

In [52]:

```
# Tracer 1diagramme à secteurs - comptage de sexe ?
```

In [53]:

```
# TRacez le scatterplot des variables pris deux à deux
# Cela n'a d'intérêt que pour les variables quantitatives bien évidemment
```