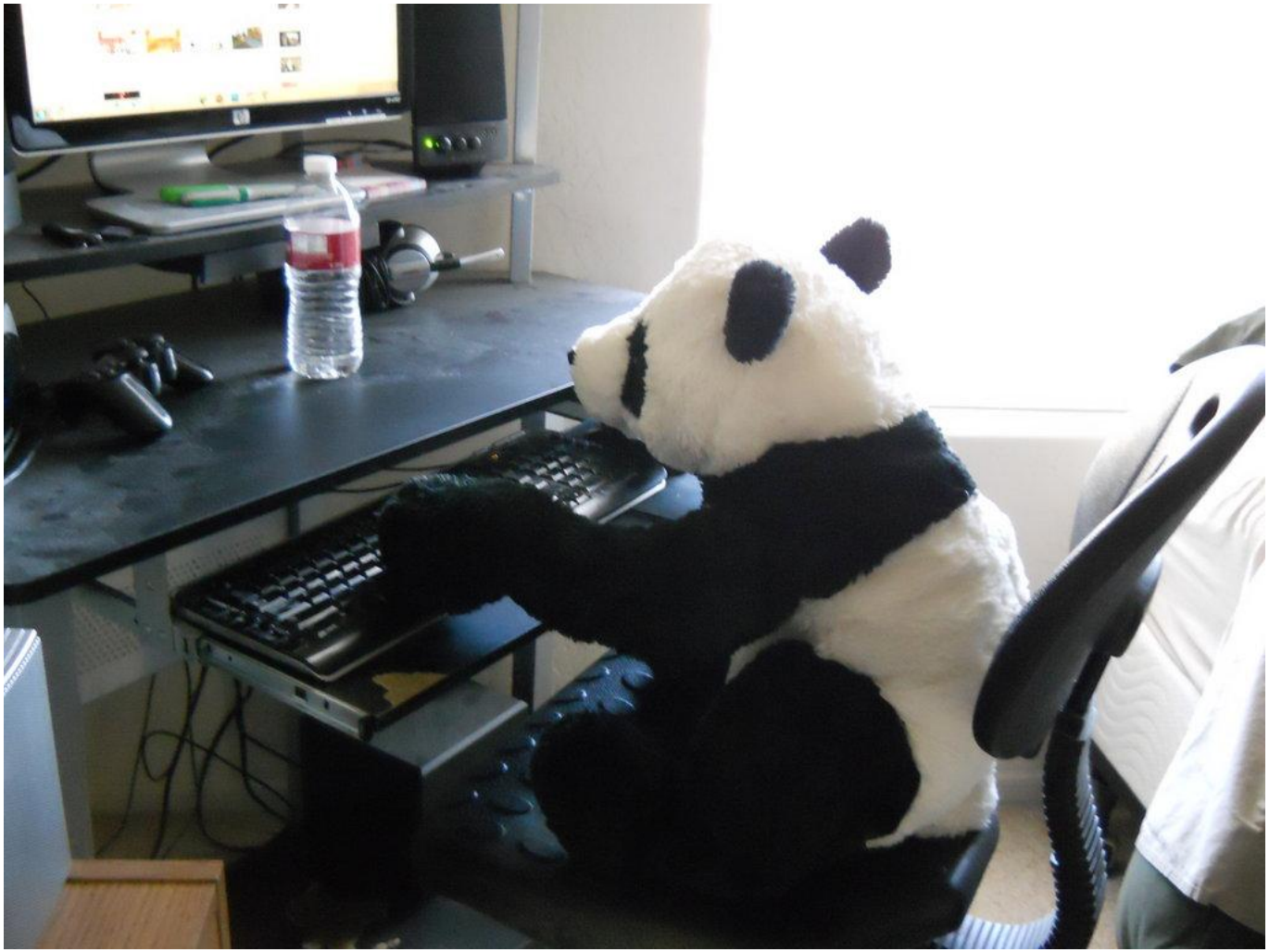
T7.GT03 : Préparation des données avec Pandas

Laurent Risser et Yves Auda

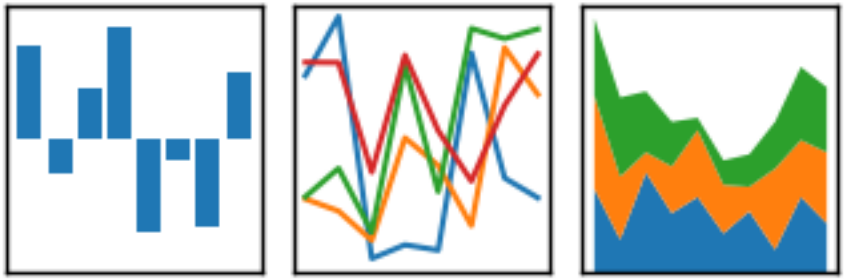
Institut de Mathématiques de Toulouse (IMT) et Géosciences Environnement Toulouse (GET)
lrisser@math.univ-toulouse.fr – yves.auda@get.omp.eu



www.quickmeme.com

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Pandas : Librairie Python pour extraire, préparer et éventuellement analyser, des données

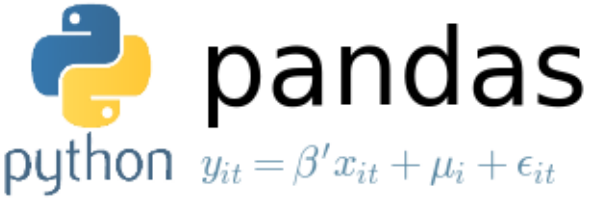
- Contient les classes Series et DataFrame (tables de données)
- Lecture des fichiers .csv, xls, hdf5, HTML, XML, JSON, MongoDB, SQL, …
- Selection/Suppression/Ajout de lignes et de colonnes, fusion de DataFrames
- Gestion de données manquantes et abérantes
- Génération de nombres aléatoires
- Tests statistiques élémentaires
- Fonctions graphiques
- Gestion de très grosses données (via HDF5)
- …

Pandas – Présentation générale – Petit exemple

```
import pandas as pd

data = {"state": ["Ohio", "Ohio", "Ohio","Nevada"],
        "year": [2000, 2001, 2002, 2001],
        "pop": [1.5, 1.7, 3.6, 2.4]}

frame = pd.DataFrame(data , columns=["year", "state", "pop"])


print frame
→       year    state   pop
→   0   2000    Ohio    1.5
→   1   2001    Ohio    1.7
→   2   2002    Ohio    3.6
→   3   2001    Nevada  2.4


frame2=pd.DataFrame(data,
                    columns=["year", "state", "pop", "debt"],
                    index=["one", "two", "three", "four"])


print frame2
→         year  state   pop  debt
→   one   2000  Ohio    1.5  NaN
→   two   2001  Ohio    1.7  NaN
→   three 2002  Ohio    3.6  NaN
→   four  2001  Nevada  2.4  NaN
...
```

```
...

frame["state"]
→   0   Ohio
→   1   Ohio
→   2   Ohio
→   3   Nevada
→   4   Nevada

frame2["debt"] = 16.5
frame2.set_value('four', 'debt', 10)

print frame2
→         year  state   pop  debt
→   one   2000  Ohio    1.5  16.5
→   two   2001  Ohio    1.7  16.5
→   three 2002  Ohio    3.6  16.5
→   four  2001  Nevada  2.4  10.0
```

→ Utilisation massive des dataframes comme en R !



Qu'est ce qu'un dataframe ?

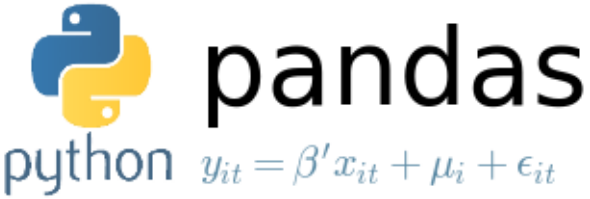
- Tableau de données
- Colonnes : observations / Lignes : variables observées
- Attribution de noms aux observations et variables (~ dictionnaire 2D)

Pourquoi des dataframes dans Python ?

- Permet la manipulation de données aussi efficacement qu'en R …
- ... tout en profitant des innombrables fonctionnalités des packages Python existants.

Périmètre de Pandas :

- Se restreint à la manipulation, lecture et prétraitement des données.
- Utilisation d'autres packages, comme Scikit-learn, pour l'analyse de données avancée.
- Pas accès aux nombreux algorithmes R issus de la communauté statistique.



Plan :

- Lecture/écriture de différents formats de données
- Manipulation de données
- Données manquantes
- Dates
- Exemple en analyse de données
- **Discussions**

Pandas – Lecture/écriture de différents formats de données

Lecture de fichier CSV :

```
[transformation des ‘,’ de chimie.csv en ‘.’]

import pandas
df1=pandas.read_csv("chimie_wd.csv" , sep=";" ,
                    index_col=0)
df2=pandas.read_csv("bacterie.csv",sep="," ,
                    index_col=0)
df1
→                T      EC   DO    pH    ORP   Tur
→    station
→    ST1         28.24  22   92.2  6.75  96.4  319
→    …
df2
→                burkholderia
→    station
→    ST1         1
→    ...
```

Autres formats (json, xls, hdf5) :

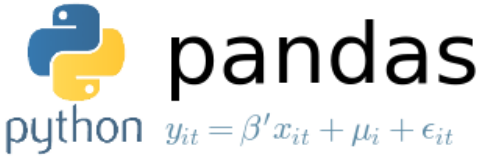
```
mydata=pandas.read_json(‘toto.json’)
mydata=pandas.read_excel(‘toto.xls’)
mydata=pandas.read_hdf(‘toto.hdf’)
```

Lecture de fichier CSV :

```
df1 <- read.csv2("chimie.csv",row.names=1)
df2 <- read.csv("bacterie.csv",row.names=1)

df1
→          T  EC   DO   pH   ORP   Tur
→    ST1  28.24  22 92.2 6.75  96.4 319.0
→    ST2  28.18  24 80.0 5.27 261.9 585.0
→    …

df2
→                burkholderia
→    ST1         1
→    …
```

Autres formats :

R bibliothèque foreign (dbf)
bibliothèque rgdal readGDAL
bibliothèque ncdf4 netcdf
bibliothèque r-cran-hdf5 hdf5



Concaténation dataframe pandas.concat

```
df12=pandas.concat([df1,df2],axis=1)
```

Statistique par groupes
- moyenne des 'T' pour chaque valeur de 'burkholderia'

```
g = df12.groupby(['burkholderia'])
g[['T']].mean()
```

- toutes les valeurs moyennes pour chaque valeur de 'burkholderia'

```
import numpy as np
g.aggregate(np.mean)
```

Fonction appliquée à une colonne, ligne

```
mT=df12.mean(0)

df1-mT     #NaN for burkhlderia
```

Concaténation dataframe cbind, rbind

```
df12 <- cbind(df1,df2)
```

Statistique par groupes
- moyenne des 'T' pour chaque valeur de 'burkholderia'

```
by(df12["T"], df12["burkholderia"], mean)
```

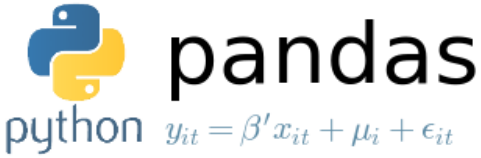
- toutes les valeurs moyennes pour chaque valeur de 'burkholderia'

```
aggregate(df12, list(df12["burkholderia"]), mean)
```

Fonction appliquée à une colonne, ligne

```
mT <- apply(df12,2,mean)

sweep(df1, 2, mT, FUN = "-")
scale(df1, scale=FALSE)
```

Pandas – Données manquantes



Lecture d'un fichier avec données manquantes

```
[transformation des ',' de chimieNA.csv en '.']

df1=pandas.read_csv("chimieNA_wd.csv" ,
                    sep=";" ,index_col=0,
                    na_values='missing')
```

Supprime les lignes correspondantes

```
df1.dropna(axis=0)
```

Estime les lignes correspondantes

```
df1.fillna(0)
df1.fillna(df1.mean())
```

Calcul avec données manquantes

```
df1.mean()
df1.mean(skipna=False)
```

Lecture d'un fichier avec données manquantes

```
df1 <-read.csv2("chimieNA.csv",
                row.names=1,
                na.strings='missing')
```

Supprime les lignes correspondantes

```
subset(df1, DO != "NA")
subset(df1, !is.na(DO))
na.omit(df1)
```

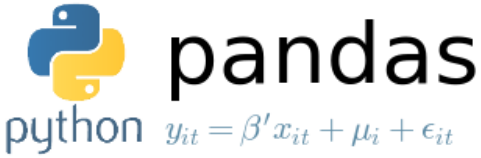
Estime les lignes correspondantes

```
df1 <- read.csv2("chimieNA.csv",row.names=1,
na.strings='missing')
df9 <- data.frame(t(apply(df1,2,mean,na.rm=TRUE)))
pos9 <- which(is.na(df1), arr.ind=TRUE)
df1[pos9] <- df9[pos9[,2]]
```

Calcul avec données manquantes

```
mean(df1[,"DO"], na.rm=TRUE)
apply(df1,2,mean,na.rm=TRUE)
```

Pandas – Dates



Différence entre deux dates

```
date1 =pandas.to_datetime('20090102',
                          format="%Y%m%d")
date2 = pandas.to_datetime('20090110',
                           format="%Y%m%d")

date2 -date1
→    Time difference of 8 days
```

Calcul du jour de l'année

```
timeDiff=date1-pandas.to_datetime('20090101',
                          format="%Y%m%d")
timeDiff.days+1
→    2
```

Différence entre deux dates

```
date1 <- as.Date("20090102", "%Y%m%d")
date2 <- as.Date("20090110", "%Y%m%d")

date2 -date1
→    Time difference of 8 days
```

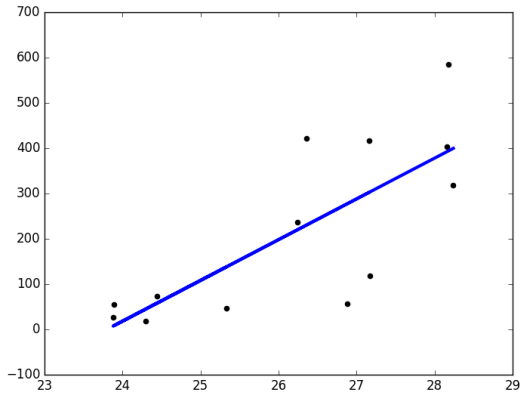
Calcul du jour de l'année

```
julian(date1, origin = as.Date("2009-01-01")) + 1
→    2
```

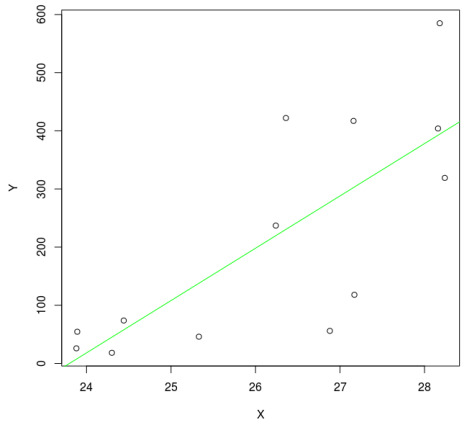
Pandas – exemple d'analyse de données – Régression linéaire

On sort du périmètre de Pandas → Scikit-learn

```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt

df1=pd.read_csv("chimie_wd.csv" , sep=";" ,
index_col=0)

X = df1['T'].values
Y = df1['Tur'].values
X=X.reshape(X.size,1)
Y=Y.reshape(Y.size,1)

regr=linear_model.LinearRegression().fit(X,Y)

plt.scatter(X, Y,  color='black')
plt.plot(X, regr.predict(X), color='blue', linewidth=3)
plt.show()
```




```
df1 <- read.csv2("chimie.csv",row.names=1)
df2 <- read.csv("bacterie.csv",row.names=1)

X <- df1[,"T"]
Y <- df1[,"Tur"]
reg1 <- lm(formula = Y ~ X)
summary(reg1)
plot(X,Y)
abline(reg1,col="green")
```



Pandas – exemple d'analyse de données – Classification de Bayes

On sort du périmètre de Pandas → Scikit-learn

```
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB

df1=pd.read_csv("chimie_wd.csv" , sep=";" , index_col=0)
df2=pd.read_csv("bacterie.csv",sep="," , index_col=0)
df12=pd.concat([df1,df2],axis=1)

data=df12.values[:,0:-1]
target=df12.values[:,-1].reshape(13,1)

gnb = GaussianNB()
classif = gnb.fit(data, target).predict(data)
```

```
allrows=np.arange(df12.shape[0])
np.random.shuffle(allrows)
rows_train=allrows[0:8]
rows_test=allrows[8:-1]

data_train=df12.values[rows_train,0:-1]
target_train=df12.values[rows_train,-1].reshape(8,1)

data_test=df12.values[rows_test,0:-1]
target_test=df12.values[rows_test,-1].reshape(4,1)

target_pred = gnb.fit(data_train,
                      target_train).predict(data_test)

100.*(target_pred==target_test.reshape(1,4)).sum()/4
```

```
library(naivebayes)


classifieur <- naive_bayes(as.factor(burkholderia) ~
        T + EC + DO + pH + ORP + Tur, df12)
classif <- predict(classifieur, df12)
```

```
row2 <- row.names(df12)
entrain <- sample(row2,8)
etest <- row2[!(row2 %in% entrain)]

classifieur <- naive_bayes(as.factor(burkholderia) ~ T + EC
+                          DO + pH + ORP + Tur,
df12[entrain,])
classif <- predict(classifieur, df12[etest,])

sum(df12[etest,"burkholderia"] == classif) / length(classif) *
100
```

Au final :

Librairie Python utilisant un format de dataframe très inspiré de celui de R

Pratique pour la manipulation de données sous Python
- → Accès aux librairies très variées de Python
- → Pas accès aux librairies de statistiques avancées de R
- → Très complémentaire de scikit-learn pour l'analyse de données

Aller plus loin avec Pandas:

http://pandas.pydata.org/pandas-docs/stable/

https://pandas.pydata.org/pandas-docs/stable/comparison_with_r.html

https://github.com/wikistat/Intro-Python/blob/master/Cal2-PythonPandas.ipynb