La Statistique avec Python

Salim Lardjane
Université de Bretagne Sud

Packages Python pour la Statistique

- Le langage Python de base ne contient que l'essentiel nécessaire à la programmation.

- Il est étendu grâce à un ensemble de packages, notamment :

  - ipython (travail interactif)

  - numpy (vecteurs et tableaux)

  - scipy (algorithmes scientifiques de base, y compris statistiques)

  - matplotlib (graphiques et visualisation)

Packages Python pour la Statistique

- pandas (permet de manipuler des Data Frames - Tableaux de Données)
- patsy (formules statistiques)
- statsmodels (modélisation statistique avancée)
- seaborn (visualisation de données statistiques)
- scikit.learn (machine learning)

Packages Python pour la Statistique

- xlrd (lire et écrire des fichiers Microsoft Excel)
- PyMC (Statistique bayésienne, y compris MCMC)
- scikits.bootstrap (algorithmes de bootstrap)
- lifelines (analyse de survie)
- rpy2 (permet d'utiliser des fonctions R en Python)

PyPI

- PyPI : The Python Package Index

- https://pypi.python.org/pypi

- C’est un dépôt de l’ensemble des packages Python disponibles. Il y en a plus de 100 000.

- Pour installer des packages sous Anaconda, on peut utiliser les programmes conda et pip.

IPython/Jupyter

- Un aspect important de l'analyse statistique est la visualisation interactive des données.

- On peut lancer une session interactive à partir de la ligne de commande dans un terminal : la commande jupyter qtconsole permet de lancer Python dans une fenêtre

- La première commande à saisir est %pylab inline

- Celle-ci permet de charger numpy et matplotlib et d'afficher les graphiques en ligne

IPython/Jupyter

- Numpy permet de manipuler des vecteurs et matrices. Matplotlib permet d'obtenir des graphiques.

- Scipy fournit divers algorithmes. Pour l'analyse statistique, scipy.stats contient la plupart des algorithmes usuels.

- Pandas permet de manipuler des Data Frames (Tableaux de Données).

- Seaborn étend les fonctionnalités graphiques de matplotlib.

- Statsmodels contient divers modules de modélisation statistiques.

- Seaborn et statsmodels utilisent tous deux les Data Frames panda.

IPython/Jupyter

- Pour saisir plusieurs commandes à la suite dans la console IPython, on peut utiliser Ctrl-Enter

Notebook et rpy2

- A partir de 2013 environ, le Notebook IPython est devenu un moyen très populaire de partager des recherches et des résultats dans la communauté Python.

- En 2015, le développement de l'interface a débouché sur un nouveau projet : Jupyter.

- Le notebook Jupyter peut être utilisé avec Python, mais également avec R, Julia et 40 autres langages de programmation.

Notebook et rpy2

- Le notebook est une interface basée sur un navigateur, ce qui le rend particulièrement utile pour la communication et la documentation.

- Il permet de combiner des sorties structurées, des équations LaTeX, des images, des graphes et des vidéos avec les résultats des commandes Python.

Notebook et rpy2

- Bien que Python gagne du terrain, le monde de la statistique avancée est encore largement dominé par R.

- Alors que Python est un langage généraliste, R est optimisé pour la manipulation interactive de données statistiques.

- Afin de combiner les deux, le package rpy2 permet de transférer des données de Python à R, d'exécuter des commandes en R et de transférer les résultats sous Python.

- Dans le Notebook Jupyter, avec rpy2, on peut même utiliser les graphiques R.

De IPython aux Programmes

- IPython est très utile pour travailler de façon interactive en ligne de commande.

- Afin de récupérer les commandes saisies, on peut utiliser les commandes %hist et %history.

- On peut ensuite recopier la liste de commandes sous Spyder, PyCharm, Wing ou tout autre IDE utilisé.

Fonctions, Modules et Packages

- Une fonction est définie par le mot-clef def et peut être définie partout dans un programme Python. Elle renvoie l'objet spécifié par l'instruction return, typiquement localisée en fin de fonction.

- Un module est un fichier ayant l'extension .py. Un module peut contenir des définitions de variables et de fonctions et des instructions Python valides.

- Un package est un répertoire contenant plusieurs modules Python et doit contenir un fichier nommé __init__.py. Par exemple, numpy est un package.

Astuces Python

- Pour toute fonction, rédiger une ligne de documentation sous la définition de la fonction.

- Importer les packages sous leurs noms usuels :
  - import numpy as np
  - import matplotlib.pyplot as plt
  - import scipy as sp
  - import pandas as pd
  - import seaborn as sns

Astuces Python

- Pour obtenir le répertoire courant, faire os.path.abspath(os.curdir).
- Pour changer le répertoire courant, faire os.chdir(…).
- Connaître les listes, tuples et dictionnaires; connaître les tableaux numpy et les tableaux de données pandas.
- Utiliser le plus possible des fonctions et comprendre la construction if __name__=‘__main__’
- Si vos fonctions personnelles sont dans le répertoire midi, vous pouvez ajouter ce répertoire à votre PYTHONPATH avec la commande :
- import sys
- sys.path.append(‘mydir’)
- Pour utiliser des caractères non-ASCII comme les caractère français ou allemands, rajouter en première ou deuxième ligne # -*- coding: utf-8 -*-

Pandas

- Pandas est un packages Python très utilisé dû à Wes McKinney.

- Il fournit des structures de données adaptées à l'analyse statistique et des fonctions facilitant l'accès aux données, l'organisation des données et la manipulation des données.

- Il est usuel d'importer pandas en faisant : import pandas as pd.

Manipulation de données

- Pandas met à disposition de l'utilisateur la structure Tableau de Données (Data Frame)

- Il s'agit d'une structure bi-dimensionnelle dont les colonnes peuvent être de types différents

- Les tableaux de données sont les objets pandas les plus utilisés

- Exemple :

Manipulation de données

```
import numpy as np

import pandas as pd

t = np.arange(0,10,0.1)

x = np.sin(t)

y = np.cos(t)

df = pd.DataFrame({‘Time’:t, ‘x’:x, ‘y’:y})
```

Manipulation de données

- En pandas, les lignes sont identifiées par leur indice et les colonnes par leur nom.

- Exemple :

df.Time

df[‘Time’]

Manipulation de données

- Pour extraire deux colonnes en même temps, on utilise une liste.

- Exemple :

```
data = df[[‘Time’, ‘y’]]
```

Manipulation de données

- Pour afficher les premières ou les dernières lignes d'un tableau de données, on utilise :

```
data.head( )

data.tail( )
```

- Pour extraire les lignes de 5 à 10, faire :

```
data[4:10]
```

Manipulation de données

- Pour accéder simultanément à des lignes et colonnes, on utilise par exemple :

```
df[[‘Time’, ‘y’]][4:10]
```

- On peut également avoir recours à la notation standard lignes/colonnes en utilisant la méthode iloc :

```
df.iloc[4:10,[0,2]]
```

Manipulation de données

- Pour accéder directement aux données et non pas au Data Frame, on utilise :

```
data.values
```

qui renvoie un tableau numpy.

Différences pandas/numpy

- numpy : traite d'abord les lignes. Par exemple data[0] est la première ligne d'un tableau (array)

- pandas : commence par les colonnes. Par exemple df['values'][0] est le premier élément de la colonne 'values' du Data Frame df.

- Si les lignes d'un Data Frame ont des labels, on peut les extraire à l'aide de la méthode loc. Par exemple, pour le label 'rowlabel', on ferait df.loc['rowlabel']. Si on veut référer à une ligne par son numéro, on utilise la méthode iloc, par exemple df.iloc[15]. La méthode iloc peut également être utilisée simultanément pour les lignes et les colonnes, par exemple : df.iloc[2:4,3].

- Le slicing de lignes fonctionne, par exemple df[0:5] pour les cinq premières lignes. Pour extraire une seule ligne, par exemple la ligne 5, faire df[5:6].

Regroupement de valeurs

- pandas fournit diverses fonctions pour traiter les données manquantes, qui sont souvent codées par nan (Not-A-Number).

- Il permet également d'utiliser les Data Frames pour regrouper des objets et faire une analyse statistique sur chaque groupe.

- Exemple : données simulées sur le nombre d'heures devant la télé pour 'h'ommes et 'f'emmes :

Regroupement de valeurs

```
import pandas as pd

import matplotlib.pyplot as plt

data = pd.DataFrame({

‘Gender’ : [‘f’, ‘f’, ‘m’, ‘f’, ‘m’,

                ‘m’, ‘f’, ‘m’, ‘f’, ‘m’, ‘m’],

‘TV’ : [3.4, 3.5, 2.6, 4.7, 4.1, 4.1,

        5.1, 3.9, 3.7, 2.1, 4.3]

})
```

Regroupement de valeurs

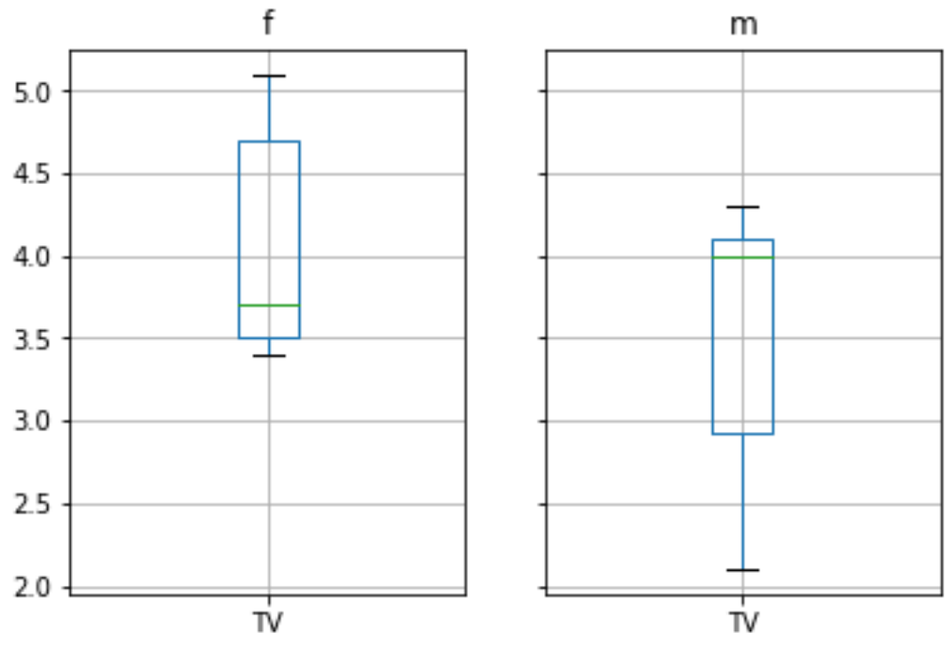
```
# Regroupement des données

grouped = data.groupby('Gender')

# Statistiques de base

print(grouped.describe( ))

# Représentations graphiques

grouped.boxplot( )

plt.show( )
```

Regroupement de valeurs

- Sortie :

TV

Gender	count	mean	std	min	25%	50%	75%	max
f	5.0	4.080000	0.769415	3.4	3.500	3.7	4.7	5.1
m	6.0	3.516667	0.926103	2.1	2.925	4.0	4.1	4.3



Pandas

- Pour les analyses statistiques, *pandas* devient réellement puissant lorsqu'il est combiné avec *statsmodels*.

Statsmodels

- Statsmodels est un package Python développé par une équipe (www.statsmodels.org)

- Au cours des cinq dernières années, statsmodels a enrichi considérablement les possibilités de Python en termes de statistique

- Il fournit des classes et des fonctions pour l'estimation de divers modèles statistiques, pour les tests statistiques et pour l'exploration des données.

Statsmodels

- Statsmodels autorise la formulation des modèles avec la notation de Wilkinson et Rogers (1973), également utilisée en S et en R.

- L’exemple suivant porte sur une régression linéaire :

Statsmodels

```
import numpy as np

import pandas as pd

import statsmodels.formula.api as sm

x = np.arange(100)

y = 0.5*x-20+np.random.randn(len(x))

df = pd.DataFrame({‘x’:x, ‘y’:y})

model = sm.ols(‘y~x’, data=df).fit( )

print(model.summary( ))
```

Statsmodels

- Sortie :

```
OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.994
Model:                            OLS   Adj. R-squared:                  0.994
Method:                 Least Squares   F-statistic:                 1.709e+04
Date:                Wed, 22 Nov 2017   Prob (F-statistic):          8.93e-112
Time:                        21:03:59   Log-Likelihood:                -150.76
No. Observations:                 100   AIC:                             305.5
Df Residuals:                      98   BIC:                             310.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
```

Statsmodels

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -19.9664      0.219    -91.121      0.000     -20.401     -19.532
x              0.4999      0.004    130.728      0.000       0.492       0.507
==============================================================================
Omnibus:                        0.194   Durbin-Watson:                   1.989
Prob(Omnibus):                  0.908   Jarque-Bera (JB):                0.377
Skew:                           0.021   Prob(JB):                        0.828
Kurtosis:                       2.702   Cond. No.                         114.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Statsmodels

- Les modèles disponibles incluent :
- La régression linéaire
- Les modèles linéaires généralisés
- L’ANOVA
- Les séries temporelles
- Les modèles de survie et de durée
- Les modèles non paramétriques
- etc.

Seaborn

- Seaborn est un package Python de visualisation basé sur matplotlib.

- L’objectif de seaborn est de fournir une interface de haut niveau et concise pour l’obtention de graphiques statistiques informatifs et attractifs.

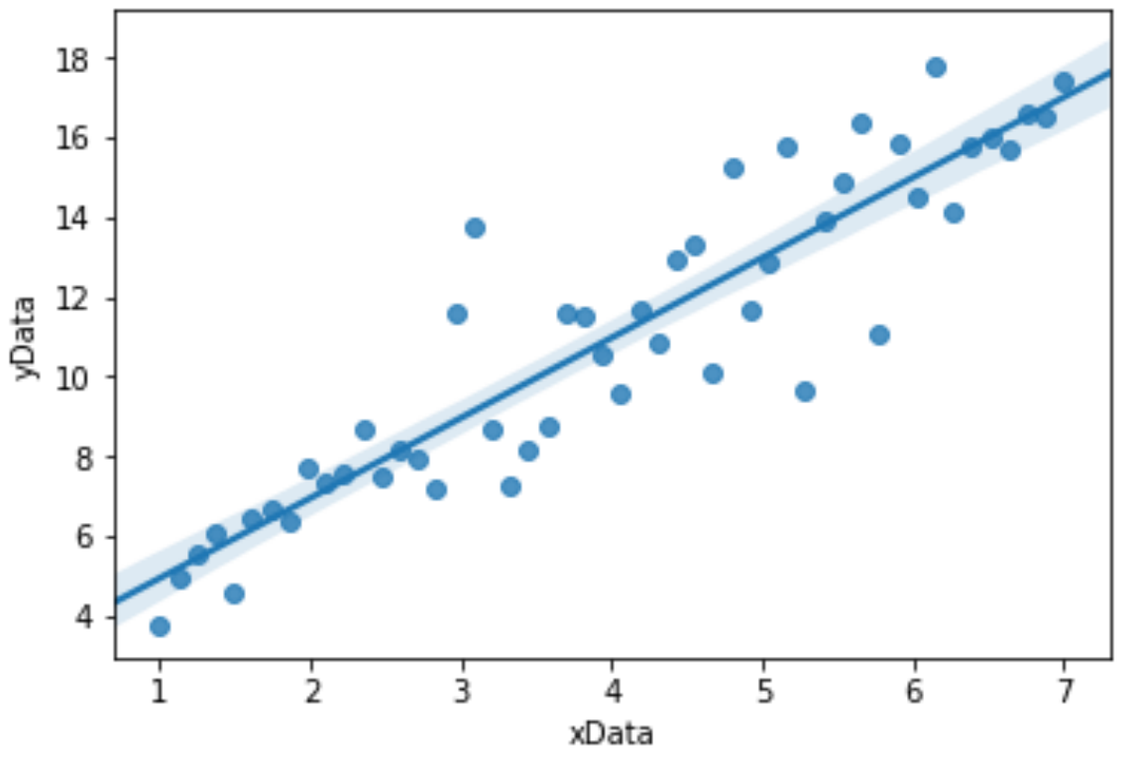
- L’exemple suivant porte sur la représentation d’une droite de régression :

Seaborn

```
import numpy as np

import matplotlib.pyplot as plu

import pandas as pd

import seaborn as sns

x = np.linspace(1,7,50)

y = 3+2*x+1.5*np.random.randn(len(x))

df = pd.DataFrame({‘xData’:x, ‘yData’:y})

sns.regplot(‘xData’, ‘yData’, data = df)

plt.show( )
```

Seaborn

- Sortie :



Exercices

- Ecrire un programme qui demande à l'utilisateur d'entrer une à une les coordonnées de deux points dans le plan puis affiche la distance euclidienne entre ces deux points.

- Ecrire un programme qui demande à l'utilisateur d'entrer son nom puis son prénom et affiche les initiales.

- Ecrire un programme qui demande à l'utilisateur d'entrer une chaîne de caractères puis l'affiche encadrée par deux étoiles.

- Ecrire un programme qui demande à l'utilisateur d'entrer un mot puis affiche un message disant si ce mot a plus de huit lettres ou pas et donne la longueur du mot.

Exercices

- Ecrire un programme qui demande à l'utilisateur d'entrer deux nombres et affiche un message précisant la différence entre le plus grand et le plus petit.

- Ecrire un programme qui résout l'équation du second degré a*x**2+b*x+c=0. Le programme demande les valeurs de a, b, c à l'utilisateur et affiche le nombre de solutions et leurs valeurs éventuelles.

- Ecrire un programme qui demande à l'utilisateur d'entrer trois nombres entiers naturels x, y, z, puis affiche la plus petite valeur.

Exercices

- Ecrire un programme qui demande à l'utilisateur d'entrer un entier strictement positif k et qui affiche les k premiers carrés non nuls.

- Ecrire un programme qui demande à l'utilisateur d'entrer une chaîne de caractères et qui affiche une nouvelle chaîne de caractères composée uniquement des voyelles de la chaîne initiale.

- Ecrire un programme qui demande à l'utilisateur d'entrer une chaîne de caractères et qui affiche les voyelles de la chaîne avec leur position dans la chaîne.

Exercices

- Un jeu consiste à lancer une pièce de monnaie; tant que Pile apparaît, on relance la pièce; si c'est Face qui apparaît le jeu se termine. Les instructions :

```
from random import randint

a = randint(0,1)
```

permettent d'affecter à la variable a la valeur 0 ou 1 de façon équiprobable et de simuler ainsi le jeu de Pile ou Face.

Ecrire un programme qui compte le nombre de fois où Pile est apparu au cours d'un jeu.

Compléter le programme afin de simuler cent fois ce jeu et d'afficher en sortie le nombre maximal de fois où Pile est apparu au cours d'un jeu.

Exercices

- Ecrire une fonction qui prend en argument une chaîne de caractères et affiche la valeur True si cette chaîne contient le caractère « e » et la valeur False sinon. La fonction sera ensuite testée sur une chaîne entrée au clavier par l'utilisateur.

- Ecrire une fonction qui prend en argument une chaîne de caractères et qui détermine puis renvoie le nombre d'occurrences du caractère « e » dans cette chaîne. La fonction sera ensuite testée sur une chaîne entrée au clavier par l'utilisateur.

Exercices

- Ecrire une fonction qui prend en argument une chaîne de caractères et renvoie une nouvelle chaîne de caractères en insérant un tiret ‘-‘ entre chaque caractère de la chaîne initiale.

- Ecrire une fonction qui prend en argument une chaîne de caractères et renvoie la chaîne écrite à l’envers

Exercices

- Ecrire une fonction qui calcule la médiane d'une liste de nombres. L'appliquer à une liste de 10 nombres aléatoires.

- En important la classe Counter du package collections, écrire une fonction qui détermine le mode d'une liste de nombres. L'appliquer à une liste de 10 nombres aléatoires à valeurs 0 et 1. On pourra utiliser la méthode most.common( ) de la classe Counter.

Exercices

- Ecrire une fonction qui détermine la distribution de fréquences d'une liste de nombres. On pourra utiliser la classe Counter du package collections.

- Ecrire une fonction qui détermine la variance et l'écart-type d'une liste de nombres.

Importation de Données

- On va voir à présent comment lire des données sous Python.

- Lire les données, s’assurer qu’elles ont été bien lues et qu’elles sont au format correct est en général la partie la plus longue du travail du statisticien.

Données Texte

- Il est recommandé d'utiliser la Jupyter Qtconsole pour lire et inspecter les données.

- Une fois que le code d'importation est mis au point, on peut le récupérer via l'historique et le coller dans un IDE tel que Spyder.

- La commande numpy : np.loadtxt permet de lire des données texte de format simple, on préfère en général utiliser panda qui propose des outils d'importation plus puissants.

Données Texte

- La démarche typique d'importation se compose des étapes suivantes :

1. Se déplacer dans le répertoire où les données sont présentes
2. Lister le contenu du répertoire
3. Sélectionner un fichier et le lire dans un tableau de données
4. Vérifier que les données ont été lues complètement et avec le bon format

Données Texte

- Sous Python, les commandes correspondantes sont par exemple :

```
import pandas as pd

cd ‘C:\Data\Storage’

pwd

ls

inFile = ‘data.txt’

df = pd.read_csv(inFile)

df.head( )

df.tail( )
```

Données Texte

- Lors de l'importation, on ajuste les options de pd.read_csv de façon à lire toutes les données correctement.

- On doit vérifier en particulier que le nombre de colonnes obtenu est correct.

- Il peut arriver que toutes les données soient lues en une seule colonne.

Fichiers texte simples

- Supposons qu'on ait un fichier data.txt de contenu :

1, 1.3, 0.6

2, 2.1, 0.7

3, 4.8, 0.8

4, 3.3, 0.9

- On peut le lire et l'afficher avec les commandes :

```
data = np.loadtxt('data.txt', delimiter = ',')

data
```

Fichiers texte simples

- Le résultat est un tableau numpy

- Une alternative est d'utiliser pandas :

```
df = pd.read_csv('data.txt', header = None)

df
```

- Le résultat est un tableau de données pandas

- La fonction pd.read_csv a l'avantage d'identifier en général correctement le type des variables (colonnes)

Fichiers texte complexes

- Les avantages de pandas deviennent évidents quand on a des fichiers texte plus compliqués.
- Considérons par exemple le fichier data2.txt de contenu :

ID, Weight, Value

1, 1.3, 0.6

2, 2.1, 0.7

3, 4.8, 0.8

4, 3.3, 0.9

Those are dummy values.

June, 2015

Fichiers texte complexes

- Une des options de pd.read_csv permet d'éviter de lire les dernières lignes : il s'agit de skipfooter.
- On importe donc les données avec :

```
df2 = pd.read_csv('data.txt', skipfooter = 3, delimiter = '[ ,]*')
```

- La dernière option delimiter = '[,]*' est une *expression régulière* spécifiant qu'un ou plusieurs espaces ou virgules peuvent séparer les données.

Fichiers Texte complexes

- Lorsque le fichier de données contient les noms des variables en première ligne, on peut accéder immédiatement à celles-ci par leur nom; par exemple :

df2 (affiche le tableau de données)

df2.Value (affiche la colonne Value)

Expressions régulières

- La manipulation de données textuelles requiert souvent d'avoir recours à des *expressions régulières*.

- Les *expressions régulières* fournissent des outils très puissants pour la manipulation ou la localisation de chaînes de caractères.

- Il existe de nombreux ouvrages sur ce sujet et divers sites web lui sont consacrés, par exemple : www.debuggex.com/cheatsheets/regex/python et www.regular-expressions.info.

Expressions régulières

- Exemples avec pandas :

- Lecture de données séparées par une combinaison de virgules, points-virgules ou espaces :

```
df = pd.read_csv(inFile, sep='[ ;,]+')
```

- Les crochets indiquent une combinaison de …, le + signifie *un ou plus*.

Expressions régulières

- Extraction de colonnes ayant des noms particuliers d'un tableau de données pandas. Par exemple, celles dont le nom commence par Vel :

```
data = np.round(np.random.randn(100,7),2)

df = pd.DataFrame(data, columns = ['Time', 'PosX', 'PosY', 'PosZ',
'VelX', 'VelY', 'VelZ'])

df.head( )

vel = df.filter(regex = 'Vel*')

vel.head( )
```

Lecture de données Excel

- Il y a deux façons de lire un fichier *Microsoft Excel* sous pandas : la fonction read_excel et la classe ExcelFile.

- read_excel permet de lire un fichier avec des arguments spécifiques au fichier (par exemple, un même format de données pour les différentes feuilles)

- ExcelFile permet de lire un fichier avec des arguments spécifiques aux feuilles (des formats de données différents pour les différentes feuilles)

- On choisit l'approche la plus rapide et la plus simple en termes de code.

Lecture de données Excel

- Exemples :

```
xls = pd.ExcelFile('path_to_the_file.xls')

data = xls.parse('Sheet1', index_col = None, na_values =
['NA'])

data = pd.read_excel('path_to_the_file.xls', 'Sheet1',
index_col = None, na_values = ['NA'])
```

- Si cela ne fonctionne pas, on peut avoir recours au package Python xlrd.

Autres formats

- La lecture de données Matlab se fait via scipy, avec la commande scipy.io.loadmat.

- On peut importer des données du presse-papier directement avec pd.read_clipboard( )

- pandas supporte également les bases de données SQL et d'autres formats. Afin d'en avoir la liste, faire

pd.read_ + TAB

Exercices

- A partir de l'observation x**(2*k) = (x**k)**2 et x**(2*k+1) =x*(x**k)**2, écrire une fonction *récursive* prenant en paramètre un nombre x quelconque et un entier naturel n et renvoyant la valeur x**n.

- Nous allons utiliser le module turtle qui permet de dessiner simplement à l'aide d'une « tortue » qui se déplace; écrire les instructions suivantes dans un fichier et les tester :

Exercices

```
from turtle import *


forward(120)
left(120)
color('red')
forward(80)


reset()
```

Exercices

```
up()

goto(-200,50)

down()

speed("fast")

circle(40)

speed("slowest")

circle(80,270)

speed("normal")

circle(50)


reset()
```

Exercices

```
width(2)

hideturtle()

color('green')

begin_fill()

for a in range(4):

    forward(50)

    left(90)

end_fill()



reset()
```

Exercices

```
color('purple')

a = 0

while (a < 12):
    a+=1
    forward(150)
    left(150)


reset()
```

Exercices

```
up()

goto(100,-80)

down()

color('blue')

write('bonjour')
```

Exercices

- Les principales fonctions sont les suivantes :
- reset( ) : efface l'écran est revient à la situation initiale
- goto(x,y) : amener le crayon à un point précis
- forward(longueur) et backward(longueur) pour avancer ou reculer d'une longueur en pixels
- left(angle) et right(angle) pour tourner à gauche ou a droite d'un angle en degrés

Exercices

- circle(rayon,angle) : tracer un arc de cercle
- up() et down() : lever ou poser le crayon
- color(‘couleur’) : choisir la couleur
- width(épaisseur) : choisir l’épaisseur du trait
- write(‘texte’) : écrire du texte
- begin_fill() et end_fill() : remplir une forme avec la couleur en cours
- speed(‘vitesse’) : choisir la vitesse
- hideturtle() : cacher la tortue

Exercices

- Tester ces fonctions en essayant de réaliser divers dessins afin de bien maîtriser leur utilisation.

- Ecrire à l'aide du module turtle un programme *récursif* qui trace la courbe de von Koch à différents niveaux (www.mathcurve.com/fractals/koch/koch.shtml)

- On obtient un flocon de von Koch en accolant trois courbes de von Koch aux trois sommets d'un triangle équilatéral. Ecrire à l'aide du module turtle un programme *récursif* qui trace le flocon de von Koch.

Exercices

- Reprendre l'exercice précédent en testant divers angles autres que 60. Pour cela, ajouter un troisième paramètre représentant l'angle aux fonctions traçant la courbe et le flocon. Attention, il faut modifier le code de la fonction traçant la courbe de façon à ce que la longueur de la figure reste la même à chaque niveau.

Graphiques statistiques

- Le langage Python de base ne fournit pas d'outils pour les représentations graphiques. Cette fonctionnalité est apportée par des packages additionnels.

- Le plus utilisé est matplotlib.

- matplotlib est conçu pour émuler Matlab. Avec matplotlib, l'utilisateur peut générer des graphiques dans le style Matlab ou dans le style Python traditionnel.

- matplotlib contient différents modules avec diverses caractéristiques.

Graphiques statistiques

- matplotlib.pyplot : c'est le module le plus utilisé pour générer des graphiques.
- Il est par convention importé par la commande :

```
import matplotlib.pyplot as plt
```

- matplotlib.mlab : contient diverses fonctions usuelles sous Matlab, telles que find, griddata, etc.

Graphiques statistiques

- matplotlib peut fournit ses sorties sous différents formats, qu'on appelle des backends :

- Dans un notebook Jupyter ou dans la console Jupyter QtConsole, la commande %matplotlib inline dirige la sortie vers la fenêtre courante.

- %pylab inline permet de combiner cela avec l'importation de pylab.

- %matplotlib qt4 (ou %matplotlib tk, selon l'installation de Python) dirige la sortie vers une fenêtre graphique séparée, ce qui permet de zoomer sur le graphique et de sélectionner des points à l'aide de la commande plt.ginput.

- plt.savefig permet de diriger la sortie vers des fichiers, par exemple aux formats PDF, JPG ou PNG.

Graphiques statistiques

- pylab est un module permettant d'importer matplotlib.pyplot et numpy dans un même espace de noms.

- Bien que de nombreux exemples sur le Net utilisent pylab, il n'est pas recommandé de l'utiliser hors de IPython.

Graphiques statistiques

- Les graphiques Python peuvent être générés dans le style Matlab ou dans un style orienté objet, plus « pythonesque ».

- Ces deux styles sont aussi valides l'un que l'autre et ont tous deux des avantages et des inconvénients.

- La seule précaution à prendre est d'éviter de mélanger les deux styles dans un même code.

Graphiques statistiques

- Exemple (dans le style pyplot / Matlab) :

```
import matplotlib.pyplot as plt

import numpy as np

x = np.arange(0, 10, 0.2)

y = np.sin(x)

plt.plot(x,y)

plt.show()
```

Graphiques statistiques

- La création de la figure et des axes est faite automatiquement par pyplot.
- Dans un style orienté objet, plus « pythonesque » :

```
fig = plt.figure() # génère le graphique

ax = fig.add_subplot(111) # ajoute des axes à la figure

ax.plot(x,y) # ajoute un graphe aux axes
```

Graphiques statistiques

- Pour l'analyse de données interactive, il est pratique de charger les commandes numpy et matplotlib.pyplot dans l'espace de travail. C'est fait à l'aide de pylab et conduit à un code de type Matlab :

```
from pylab import *

x = arange(0,10,0.2)

y = sin(x)

plot(x,y)

show()
```

Graphiques statistiques

- Le style orienté objet est de plus en plus utile au fur et à mesure que les graphiques deviennent plus compliqués.

- Il permet de mieux comprendre l'origine des objets graphiques et ce qui se passe.

- Par exemple, le code suivant produit une figure avec deux graphiques l'un au-dessus l'autre et identifie clairement quel graphique est associé à quels axes.

Graphiques statistiques

```
import matplotlib.pyplot as plt

import numpy as np

x = np.arange(0,10,0.2)

y = np.sin(x)

z = np.cos(x)

fig, axs = plt.subplots(nrows=2,ncols=1)

axs[0].plot(x,y)

axs[0].set_ylabel(‘sinus’)

axs[1].plot(x,z)

axs[1].set_ylabel(‘cosinus’)

plt.show()
```

Graphiques statistiques

- La façon la plus simple de trouver et d'implémenter l'un des divers types de graphiques pouvant être réalisés sous matplotlib est de consulter la galerie : www.matplotlib.org/gallery.html et de récupérer le code correspondant au graphique souhaité.

Graphiques statistiques

- Le package seaborn complémente matplotlib et offre une interface concise pour l'obtention de graphiques informatifs et attractifs.

- De même, pandas offre divers outils pour visualiser les Data Frames.

- D'autres packages graphiques intéressants sont :

Graphiques statistiques

- plot.ly : c'est un package disponible sous Python, Matlab et R et permet d'obtenir des graphiques de qualité (www.plot.ly).

- bokeh : c'est une librairie Python de visualisation interactive qui cible les navigateurs web. Elle permet de développer rapidement des graphiques interactifs, des tableaux de bord et des applications basées sur les données (http://bokeh.pydata.org/)

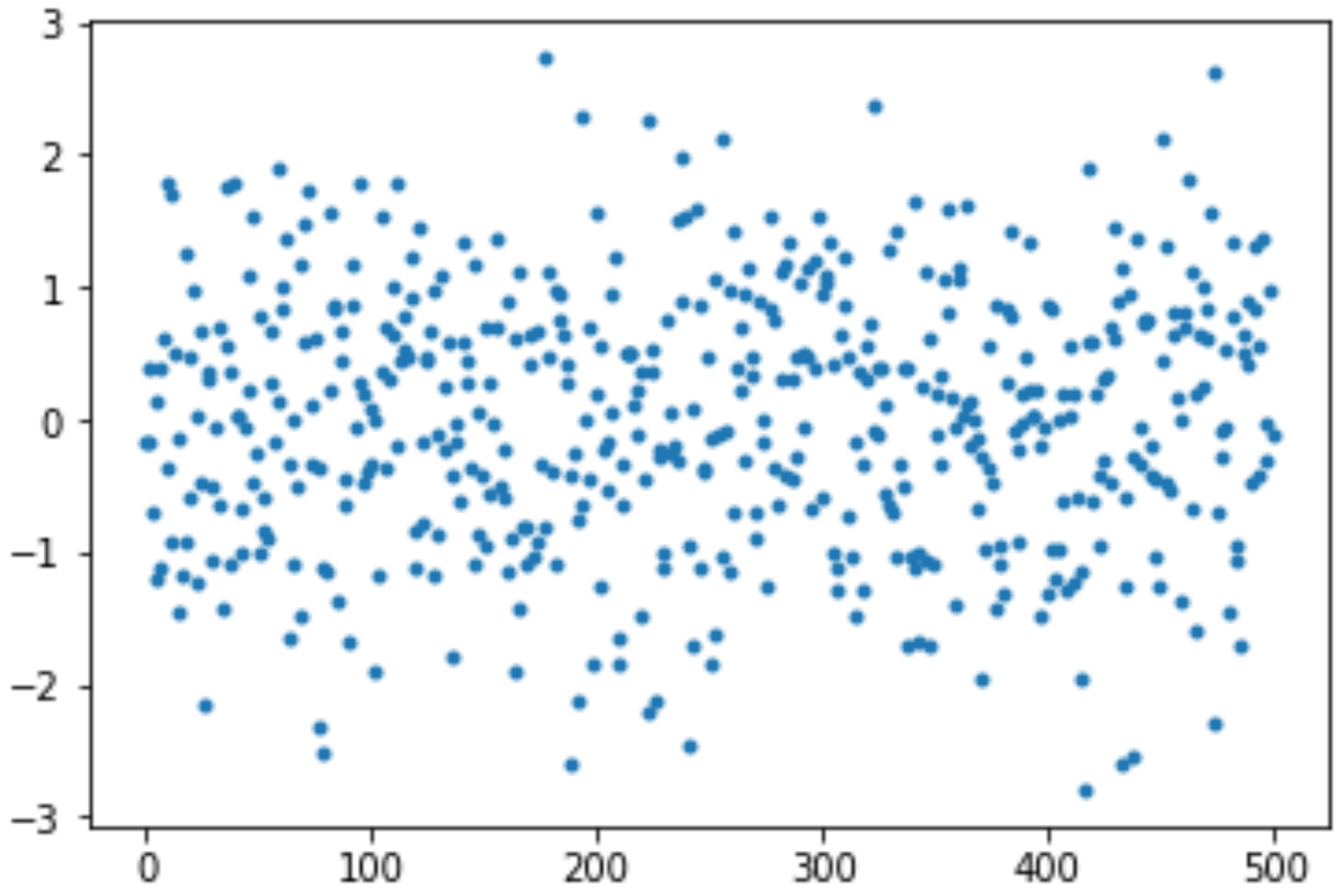
- ggplot : émule le package R ggplot, très apprécié des utilisateurs.

Graphiques statistiques

- Exemple (stripplot) :

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import scipy.stats as stats

import seaborn as sns

x = np.random.randn(500)

plt.plot(x,’.’)

plt.show()
```

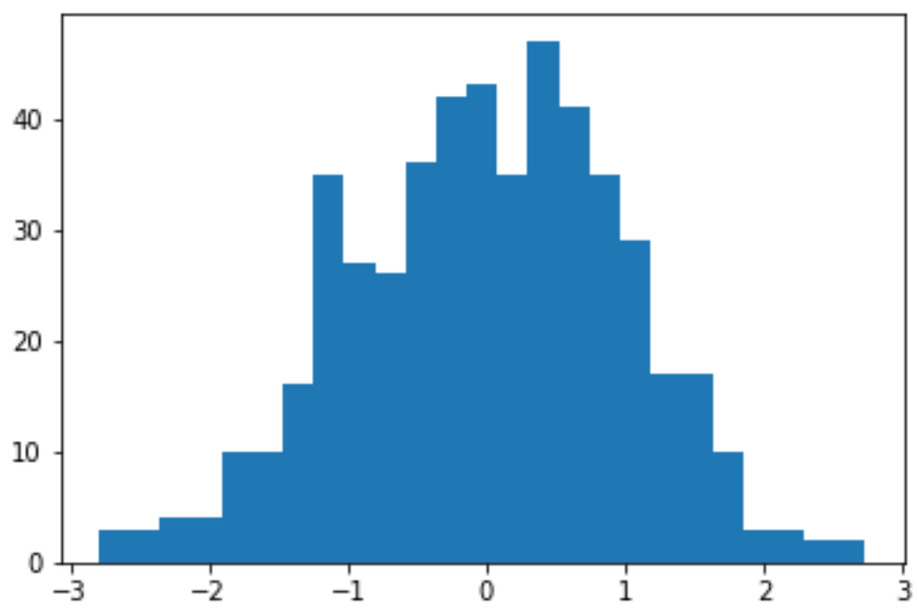
Graphiques statistiques



Graphiques statistiques

- Exemple (histogramme) :

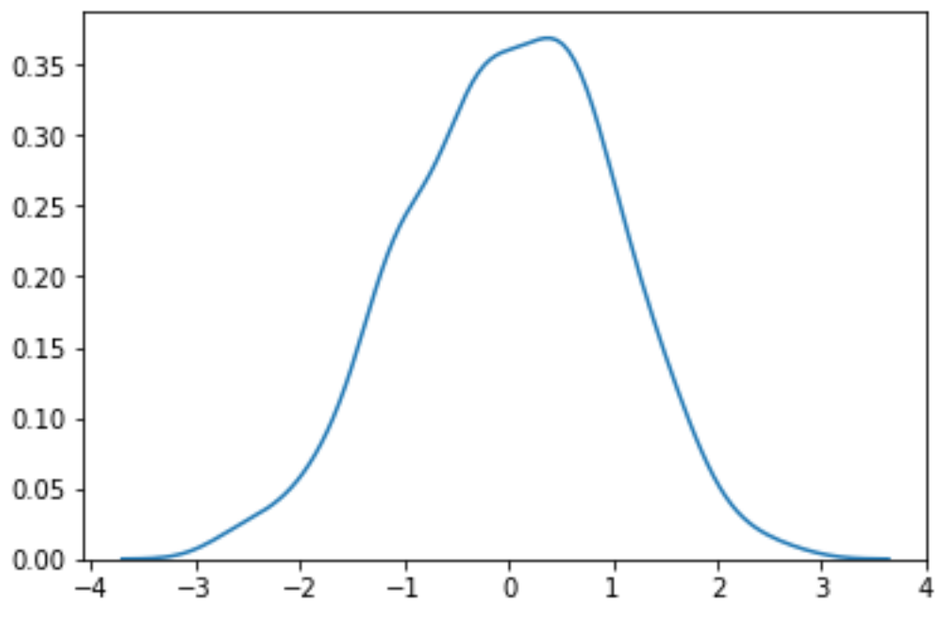
plt.hist(x,bins=25)



Graphiques statistiques

- Exemple (estimation de densité par noyaux) :

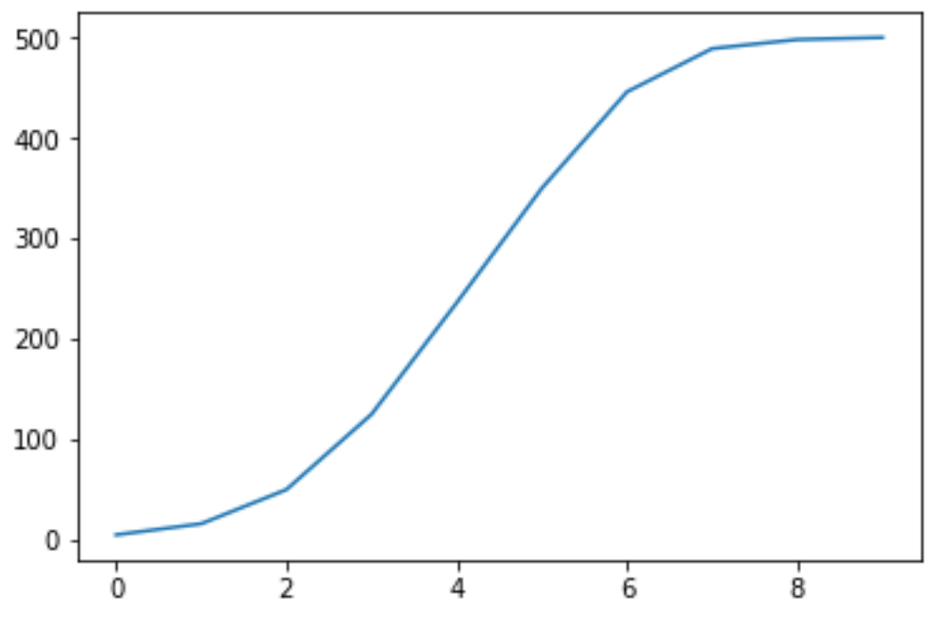
sns.kdeplot(x)



Graphiques statistiques

- Exemple (fréquences cumulées) :

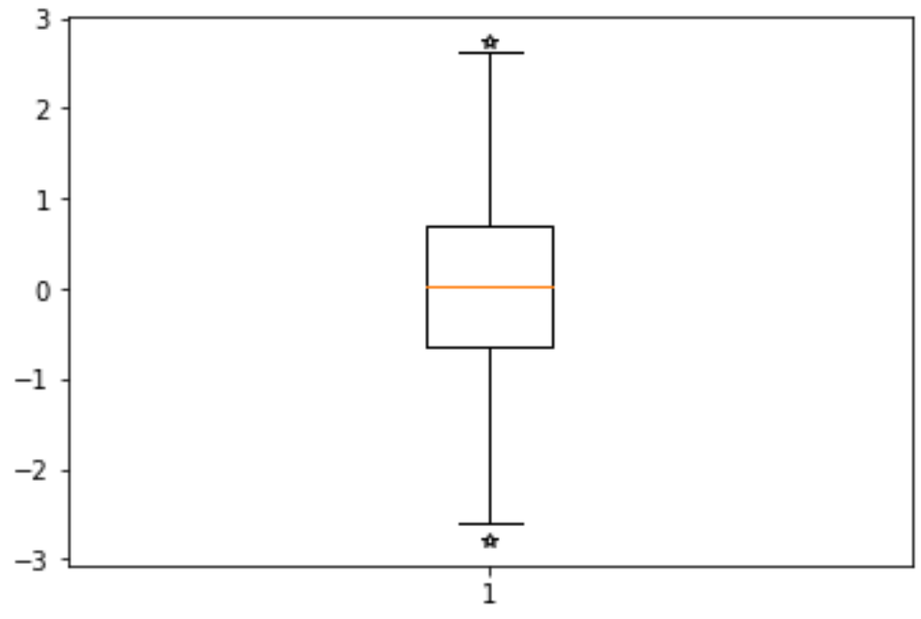
plt.plot(stats.cumfreq(x)[0])



Graphiques statistiques

- Exemple (boxplot) :

plt.boxplot(x,sym=‘*’)

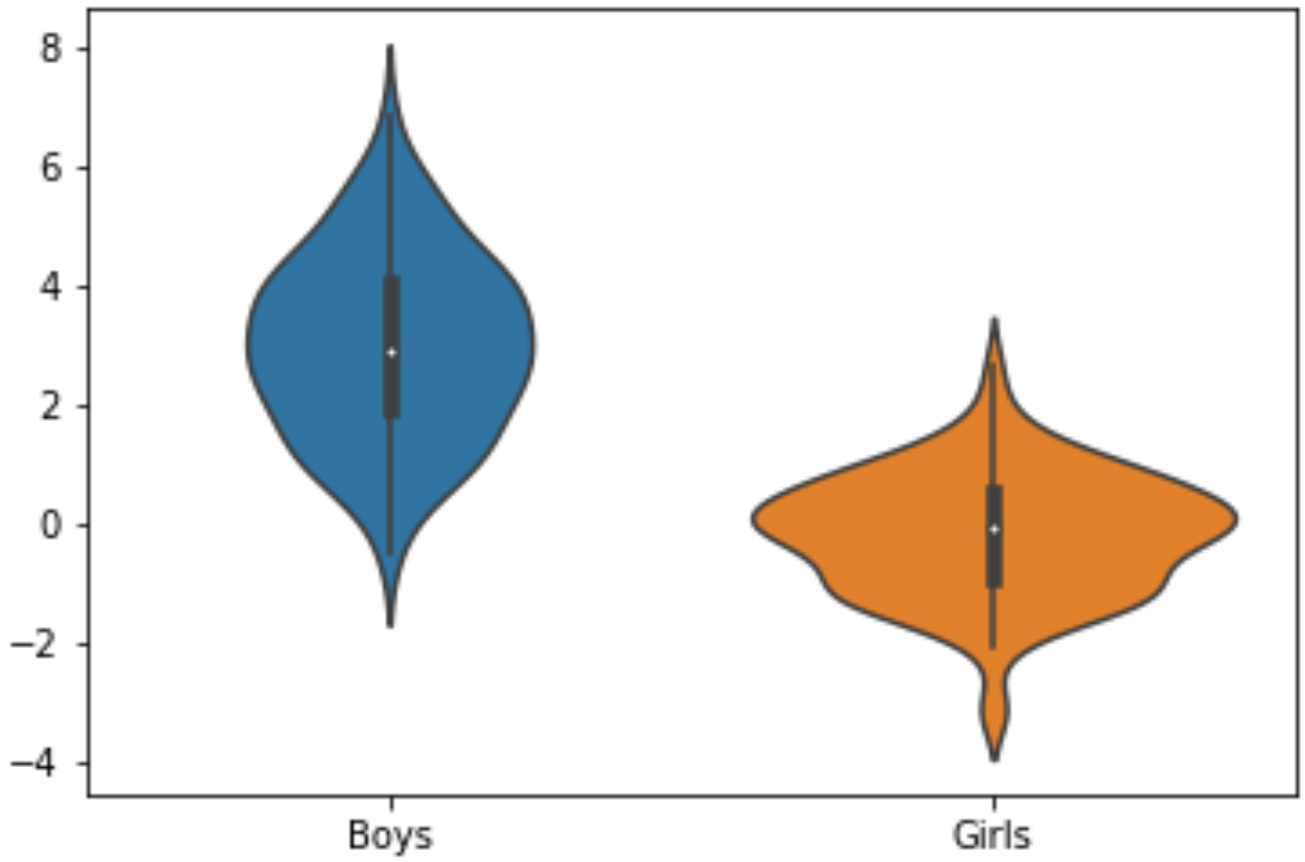


Graphiques statistiques

- Exemple (diagramme en violon) :

```
nd = stats.norm

data = nd.rvs(size=(100))

nd2 = stats.norm(loc=3,scale=1.5)

data2 = nd2.rvs(size=(100))

df = pd.DataFrame({‘Girls’:data,’Boys’:data2})

sns.violinplot(data=df)
```

Graphiques statistiques

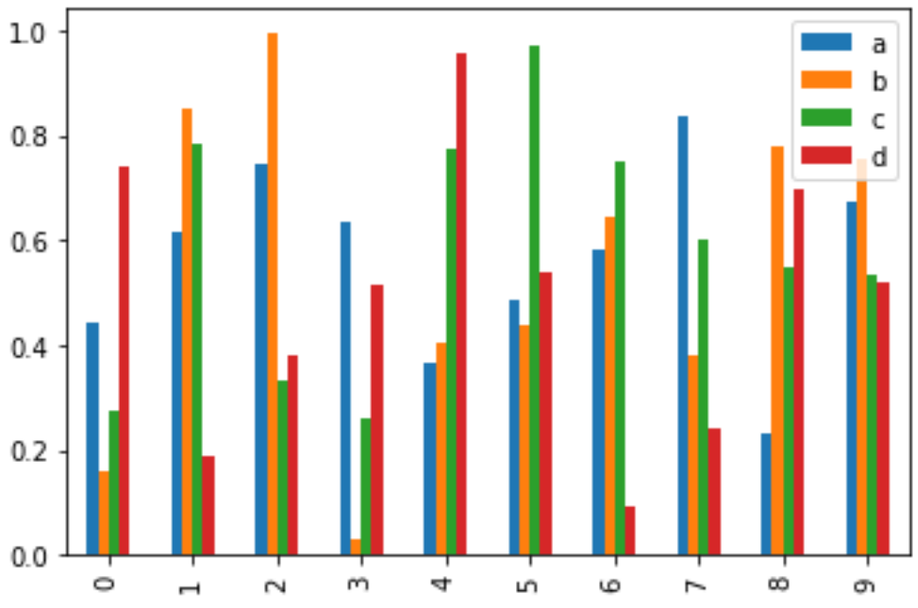


Graphiques statistiques

- Exemple (diagrammes en barres empilés) :

df = pd.DataFrame(np.random.rand(10,4),columns = [‘a’,’b’,’c’,’d’])

df.plot(kind=‘bar’,grid=False)

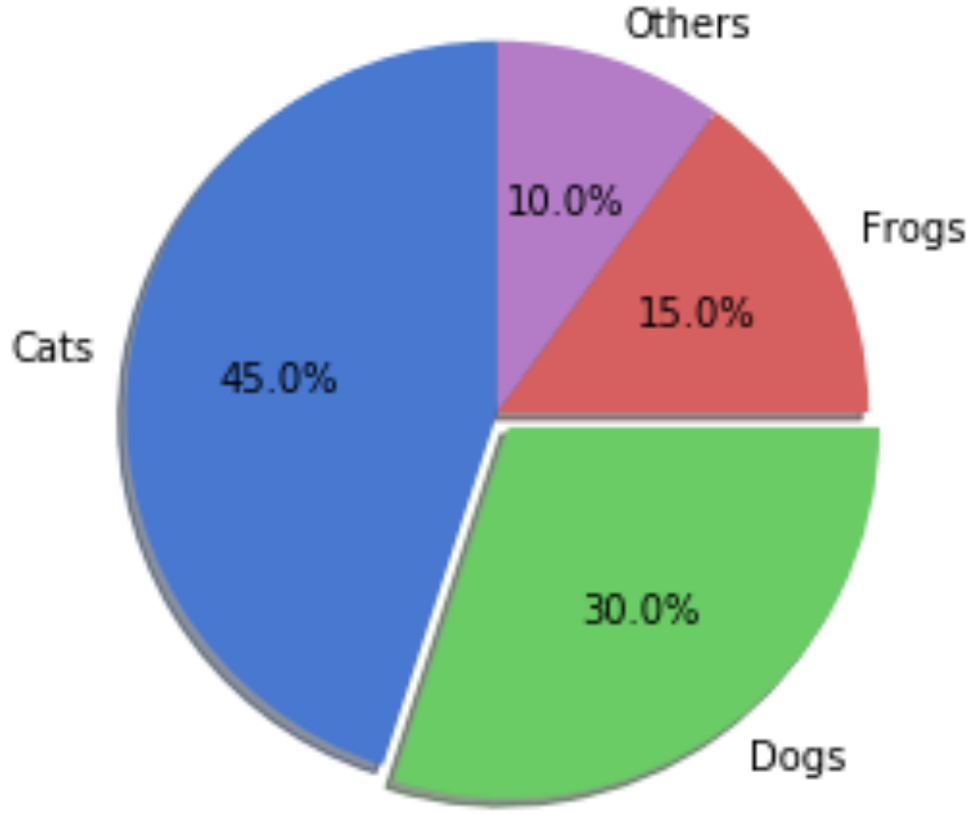


Graphiques statistiques

- Exemple (diagramme en secteurs) :

```
import seaborn as sns

import matplotlib.pyplot as plt

txtLabels = ‘Cats’, ‘Dogs’, ‘Frogs’, ‘Others’

fractions = [45,30,15,10]

offsets=(0,0.05,0,0)

plt.pie(fractions, explode=offsets,labels=txtLabels,autopct=‘%1.1f%%’,
shadow=True,startangle=90,colors=sns.color_palette(‘muted’))

plt.axis(‘equal’)
```

Graphiques statistiques

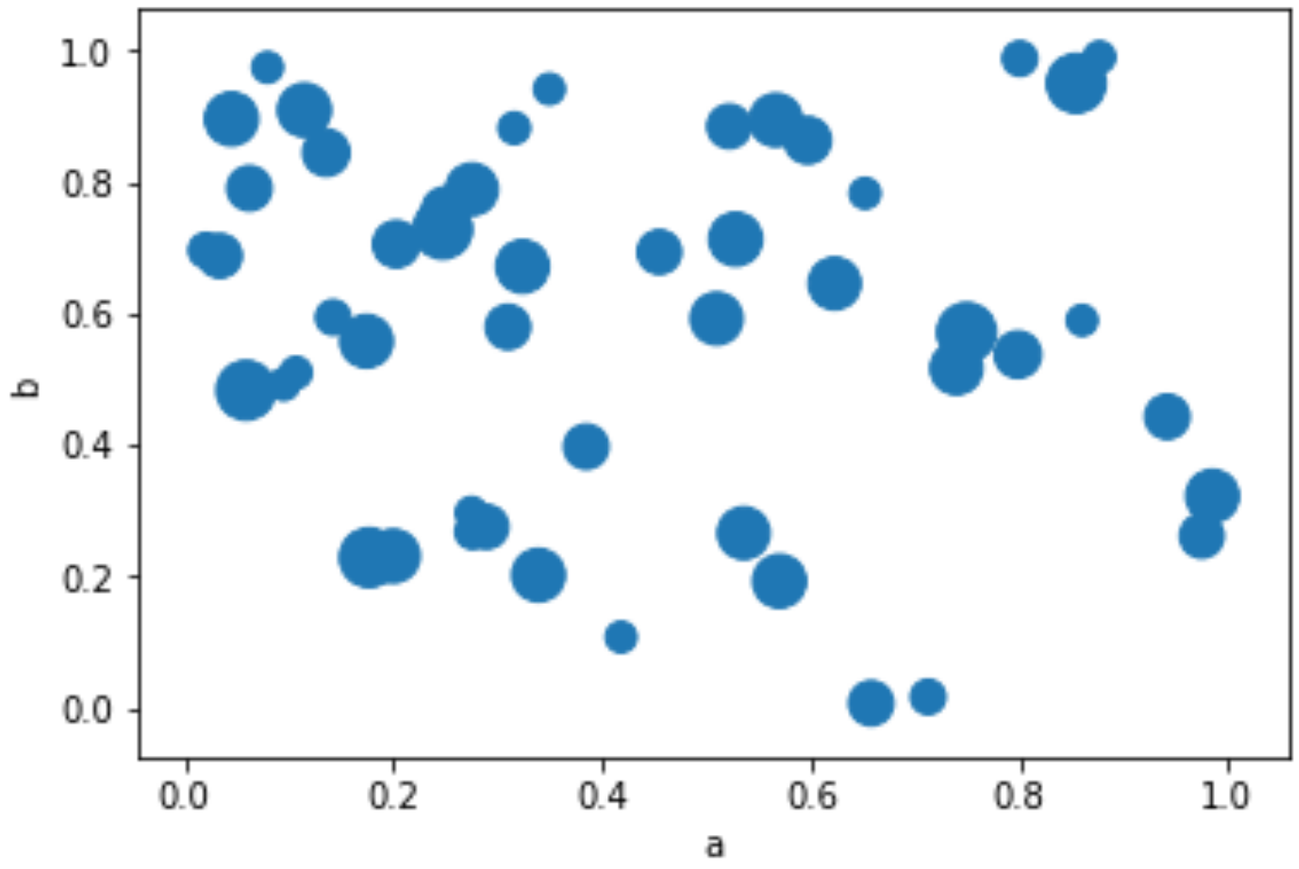


Graphiques statistiques

- Exemple (nuage de points) :

```
df2 =
pd.DataFrame(np.random.rand(50,4),columns=[‘a’,’b’,’c’,’d’])

df2.plot(kind=‘scatter’, x=‘a’, y=‘b’, s=df[‘c’]*300)
```

Graphiques statistiques



Graphiques statistiques

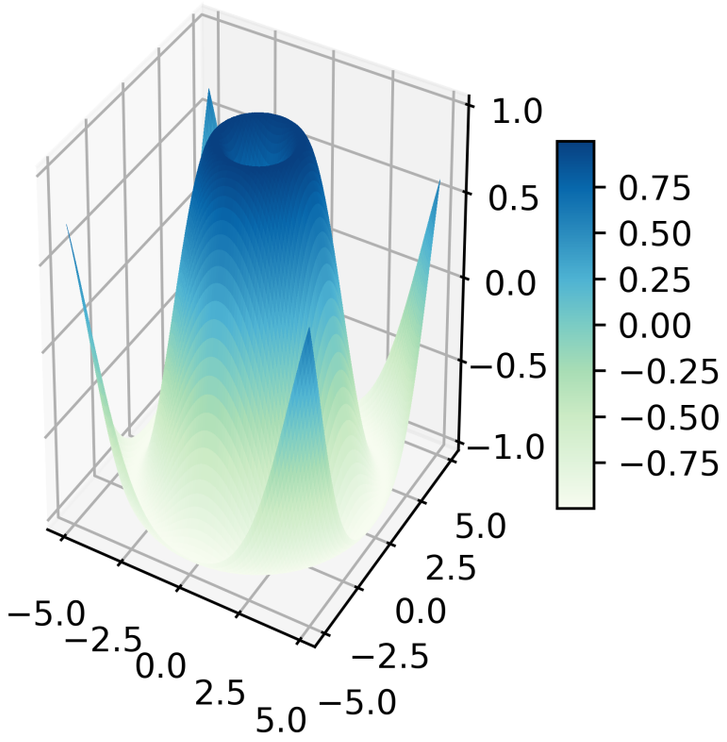
- Exemple (graphiques 3D) :

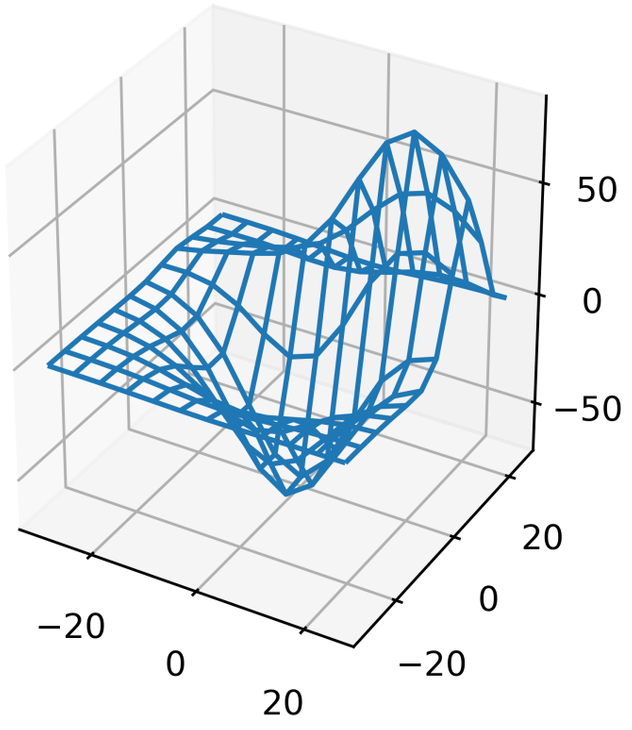
```
import numpy as np

from matplotlib import cm

from mpl_toolkit.mplot3d.axes3d import get_test_data

fig = plt.figure(figsize=plt.figaspect(0.5))

ax = fig.add_subplot(1,2,1, projection='3d')

X = np.arange(-5,5,0.1)

Y = np.arange(-5,5,0.1)

X, Y = np.meshgrid(X,Y)

R = np.sqrt(X**2+Y**2)

Z = np.sin(R)
```

Graphiques statistiques

```
surf = ax.plot_surface(X,Y,Z,rstride=1,cstride=1,cmap=cm.GnBu,  linewidth=0,antialiased=False)

ax.set_zlim3d(-1.01,1.01)

fig.colorbar(surf, shrink=0.5, aspect = 10)

ax = fig.add_subplot(1,2,2,projection='3d')

X,Y,Z = get_test_data(0.5)

ax.plot_wireframe(X,Y,Z,rstride=1,cstride=1)

outfile = '/Users/Salim/Desktop/3dGraph.png'

plt.savefig(outfile, dpi=600)

print('Image sauvegardée sous {0}'.format(outfile))

plt.show()
```

Graphiques statistiques





Exercices

- Sauvegarder les données iris de R dans un fichier csv.

Les lire sous Python dans un Data Frame pandas.

Effectuer diverses représentations graphiques des données iris (nuages de points, boxplots, histogrammes, etc).

- Lire les données de votre projet dans un Data Frame pandas.

Obtenir sous Python différentes représentations graphiques de vos données que vous avez déjà obtenues sous R et SAS.

Distributions et tests d'hypothèses

- Lorsqu'on dispose d'un échantillon d'une distribution, on peut caractériser celle-ci à l'aide de différentes mesures de position centrale et de dispersion.

- Sous Python, la moyenne d'un tableau peut-être obtenue à l'aide de la fonction np.mean.

- Si le tableau contient des valeurs manquantes, on utilise la fonction np.nanmean.

Distributions et tests d'hypothèses

- Exemple :

```
import numpy as np

x = np.arange(10)

np.mean(x)

xWithNan = np.hstack((x,np.nan))

np.mean(xWithNan)

np.nanmean(xWithNan)
```

Distributions et tests d'hypothèses

- Sous Python, la médiane est obtenue à l'aide de la fonction np.median.

- S'il y a des valeurs manquantes, on utilise np.nanmedian.

- La façon la plus simple de déterminer le mode (valeur la plus fréquente) d'un échantillon est d'utiliser la fonction mode de scipy.stats, qui fournit la valeur et l'effectif du mode.

Distributions et tests d'hypothèses

- Exemple :

```
from scipy import stats

data = [1, 3, 4, 4, 7]

stats.mode(data)
```

Distributions et tests d'hypothèses

- Dans certains cas, on utilise la moyenne géométrique comme caractéristique de tendance centrale d'une distribution.

- Une fonction lui est dédiée sous scipy.stats. Il s'agit de gmean.

- Exemple :

```
x = np.arange(1,101)

stats.gmean(x)
```

Distributions et tests d'hypothèses

- L'amplitude d'un échantillon, différence entre la valeur maximale et minimale, est obtenue à l'aide de la fonction np.ptp.

- La variance d'un échantillon est obtenue à l'aide de la fonction np.var.

- L'écart-type à l'aide de la fonction np.std.

- Si on souhaite avoir n (taille de l'éhantillon) au dénominateur, on doit spécifier ddof=0.

- Si on souhaite avoir n-1, on spécifie ddof=1.

- Par défaut, on a ddof=0, sauf sous pandas ou ddof=1 par défaut.

Distributions et tests d'hypothèses

- Sous Python, la façon la plus élégante de travailler avec des distributions (densité, fonction de répartition, fonction de survie, fonction quantile, générateur aléatoire) est de commencer par la définir (exemple : nd = stats.norm( )).

- En jargon Python, on dit qu'on a une « distribution gelée » (frozen distribution).

- Dans un deuxième temps, on choisit la fonction associée à cette distribution qu'on souhaite utiliser (exemple : y = nd.cdf(x), pour la fonction de répartition).

Distributions et tests d'hypothèses

- Exemple :

```
import numpy as np

from scipy import stats

myDF = stats.norm(5,3)

x = np.linspace(-5,15,101)

y = myDF.cdf(x)
```

Distributions et tests d'hypothèses

- CDF : fonction de répartition
- PDF : fonction de densité
- SF : fonction de survie (SF = 1-CDF)
- PPF : fonction quantile (Percentile Point function)
- ISF : inverse de la fonction de survie
- PMF : fonction de masse dans le cas discret
- RVS : échantillon aléatoire

Distributions et tests d’hypothèses

- La distribution discrète la plus simple est la distribution de Bernoulli.
- Exemple :

```
from scipy import stats

p = 0.5

bernoulliDist = stats.bernoulli( p ) # Distribution gelée

p_pile = bernoulliDist.pmf(0)

p_face = bernoulliDist.pmf(1)

essais = bernoulliDist.rvs(10)
```

Distributions et tests d'hypothèses

- Distribution binomiale :

```
from scipy import stats

import numpy as np

(p, num) = (0.5, 4) # paramètres p et n de la loi binomiale

binomDist = stats.binom(num,p)

binomDist.pmf(np.arange(5)) # fonction de masse
```

Distributions et tests d’hypothèses

- Distribution normale :

```
import numpy as np

from scipy import stats

mu = -2

sigma = 0.7

myDistribution = stats.norm(mu,sigma)

significanceLevel = 0.05

myDistribution.ppf([significanceLevel/2,1-significanceLevel/2])
```

Distributions et tests d'hypothèses

- Exercice : Le poids moyen d'un nouveau-né en bonne santé en France est de 3.5 kg avec un écart-type de 0.76 kg. Le poids d'un bébé donné est de 2.6 kg. Rejette-t-on l'hypothèse qu'il est en bonne santé au niveau de significativité de 5% ? Répondre à la question en utilisant Python, en considérant que la distribution des poids est gaussienne.

Distributions et tests d’hypothèses

- Pour répondre à la question, on procède ainsi :

- Identifier la distribution qui caractérise les nouveaux-nés en bonne santé : mu = 3.5, sigma = 0.76

- Calculer la FR à la valeur d’intérêt : FR(2.6) = 0.118. Ainsi, la probabilité qu’un bébé en bonne santé soit au moins 0.9 kg plus léger que le bébé moyen est de 11.8%.

- Comme la distribution est gaussienne, la probabilité qu’un bébé en bonne santé soit au moins 0.9 kg plus lourd que le bébé moyen est également de 11.8%.

- Interpréter le résultat : Si le bébé est en bonne santé, la probabilité que son poids s’écarte d’au moins 0.9 kg de la moyenne est 2*11.8% = 23.6 %, ce qui n’est pas significatif au seuil de 5%. On en rejette donc pas l’hypothèse que le bébé est en bonne santé.

Distributions et tests d'hypothèses

- Avec Python :

```
from scipy import stats

nd = stats.norm(3.5, 0.76)

nd.cdf(2.6)
```

Exercice (Monty Hall)

- Le jeu suivant (Monty Hall) oppose un présentateur à un candidat. Le joueur est placé devant trois portes fermées. Derrière l'une d'elles se trouve une voiture et derrière chacun des deux autres se trouve une chèvre. Il doit tout d'abord désigner une porte. Puis le présentateur ouvre une porte qui n'est ni celle choisie par le candidat, ni celle cachant a voiture (le présentateur sait quelle est la bonne porte dès le début). Le candidat a alors le droit ou bien d'ouvrir la porte qu'il a choisie initialement, ou bien d'ouvrir la troisième porte.

- Que doit-il faire ?

- Quelles sont les chances de gagner la voiture en agissant au mieux ?

Exercice (Monty Hall)

- Simuler le jeu du Monty Hall, identifier la meilleure stratégie et calculer la probabilité de gagner la voiture avec cette stratégie.

- Essayer de retrouver le résultat avec le calcul des probabilités.

Distribution de Student

- Si *m* est la moyenne arithmétique d'un échantillon i.i.d. de taille *n* associé à une variable aléatoire gaussienne X d'espérance *u*, et *s* son écart-type, la quantité t = (m - u)/(s/sqrt(n)) peut-être utilisée pour le calcul d'un intervalle de confiance pour l'espérance.

- Plus précisément, IC = [m - (s/sqrt(n))*t(df,alpha) ; m + (s/sqrt(n))*t(df,alpha)] où t(df,alpha) est le quantile d'ordre 1-alpha/2 de la loi de Student à df = *n*-1 degrés de liberté.

- Sous Python, ce quantile peut être obtenu à l'aide de la fonction quantile (PPF) ou la fonction de survie inverse (ISF).

Distribution de Student

- Exemple :

```
import numpy as np

n = 20

df = n-1

alpha = 0.05

stats.t(df).isf(alpha/2)
```

- On peut également obtenir l’intervalle de confiance de niveau 1-alpha en une fois :

```
alpha = 0.95

df = len(data)-1

ci = stats.t.interval(alpha,df, loc = np.mean(data), scale = stats.sem(data))
```

Distribution du Chi-2

- Exemple : Un producteur de pilules doit produire des pilules avec un écart-type spécifié de sigma = 0.05 sur le poids.

- Dans un lot de pilules, on prélève un échantillon de taille *n* = 13. On obtient les poids 3.04, 2.94, 3.01, 3.00, 2.94, 2.91, 3.02, 3.04, 3.09, 2.95, 2.99, 3.10, 3.02 g.

- Question : *L'écart-type de la production est-il plus élevé que la valeur spécifiée ? On supposera que le poids est gaussien.*

Distribution de Student

- La loi du Chi-2 décrivant la distribution de la somme des carrés de v.a. gaulliennes standard indépendantes, on doit normaliser les données avant de calculer la p-value.

- Celle-ci vaut : p-value = SF = 1-CDF(Somme((x-*m*)/sigma)**2) = 0.1929 où *m* désigne la moyenne arithmétique de l'échantillon et où sigma = 0.05. SF et CDF désignent respectivement la fonction de survie et la fonction de répartition d'une loi du Chi-2 à n-1 degrés de liberté.

Distribution du Chi-2

- Interprétation : Si le lot est issu d'une distribution gaussienne d'écart-type sigma=0.05, la probabilité d'obtenir une valeur de Chi-2 au moins égale à la valeur observée est d'environ 19%; la valeur observée n'est donc pas atypique. En d'autres termes, le lot correspond aux spécifications.

- Le nombre de degrés de liberté est *n*-1 car on a du estimer l'espérance par la moyenne arithmétique.

Distribution du Chi-2

- Avec Python :

```
import numpy as np

from scipy import stats

data = np.r_[3.04, 2.94, 3.01, 3.00, 2.94, 2.91, 3.02, 3.04, 3.09, 2.95, 2.99,
3.10, 3.02]

sigma = 0.05

chi2Dist = stats.chi2(len(data)-1)

statistic = sum(((data-np.mean(data))/sigma)**2)

chi2Dist.sf(statistic)
```

Distribution de Fisher

- Exemple : On souhaite comparer la précision de deux méthodes de mesure du mouvement des yeux. La précision est déterminée par la variance des mesures.

- En regardant 20° à droite, on obtient les mesures suivantes :

- Méthode 1 : [20.7, 20.3, 20.3, 20.3, 20.7, 19.9, 19.9, 19.9, 20.3, 20.3, 19.7, 20.3]

- Méthode 2 : [19.7, 19.4, 20.1, 18.6, 18.8, 20.2, 18.7, 19.0]

- Résoudre le problème à l'aide de Python.

Distribution de Fisher

- On souhaite tester l'égalité des variances pour deux échantillons issues de lois gaussiennes. On utilise donc la statistique de Fisher F (rapport des variances empiriques).

- On trouve que F = 0.244 pour n-1, m-1 degrés de libertés où n et m désignent la taille des échantillons.

- Le code Python ci-dessous permet de calculer la p-value; elle vaut 0.019, ce qui conduit à rejeter l'hypothèse d'égale précision des deux méthodes.

Distribution de Fisher

```
import numpy as np

from scipy import stats

method1 = np.array[20.7, 20.3, 20.3, 20.3, 20.7, 19.9, 19.9, 19.9, 20.3, 20.3, 19.7, 20.3]

method2 = np.array[19.7, 19.4, 20.1, 18.6, 18.8, 20.2, 18.7, 19.0]

val = np.var(method1, ddof=1)/np.var(method2, ddof=1)

fd = stats.f(len(method1)-1,len(method2)-1)

p_oneTail = fd.cdf(fval) # -> 0.019

if (p_oneTail < 0.025) or (p_oneTail > 0.975) :

   print(‘’Différence significative”)

else print(‘’Pas de difference significative”)
```

Exercices

- Créer un tableau numpy contenant les données 1, 2, 3, … , 10. Calculer la moyenne et l'écart-type (échantillonal) correspondant.

- Générer et représenter graphiquement la densité de probabilité (PDF) d'un distribution normale d'espérance 5 et d'écart-type 3.

  Générer 1000 pseudo-réalisations de cette distribution.

  Calculer une estimation de l'écart-type de la moyenne pour les données obtenues.

  Représenter un histogramme des données.

  Déterminer un intervalle de fluctuation contenant approximativement 95% des données.

Exercices

- Un docteur vous dit qu'il peut utiliser des implants de la hanche dans une opération chirurgicale même s'ils sont 1mm plus petits ou plus grands que la spécification. Un responsable financier vous dit qu'il est possible de jeter un implant sur 1000 tout en conservant une activité profitable. Quel est l'écart-type requis sur la production d'implants de la hanche pour vérifier simultanément ces deux exigences ?

Exercices

- Les poids d'un échantillon de vos collègues sont de 52, 70, 65, 85, 62, 83, 59 kg. Calculer la moyenne correspondante et un intervalle de confiance à 95% pour la moyenne sur la population, en supposant que cette dernière est gaussienne.

- Créer trois échantillons associés à la loi normale standard de taille 1000 chacun. Les élever au carré puis les sommer (de façon à avoir 1000 valeurs) et créer l'histogramme correspondant avec 100 classes. Comparer à la densité du Chi-2 à 3 degrés de liberté.

Exercices

- On étudie deux pommiers. Trois pommes du premier pèsent 110, 121 et 143 g et quatre pommes du deuxième pèsent 88, 93, 105 et 124 g. Les variances des productions des deux arbres sont-elles significativement différentes ?

- Dans un pays donné, il y a eu 62 accidents fatals de la route l'année dernière. On a donc en moyenne 62/(365/7) = 1.19 accident fatal par semaine. Quelle est la probabilité qu'une semaine donnée il y ait 0, 2, 5 accidents ? On utilisera la loi de Poisson.

Modélisation Statistique

- Un certain nombre de packages permettent d'étendre les fonctionnalités de Python en termes d'analyse statistique des données et de modélisation.
- On verra dans la suite des applications des packages suivants :
- statsmodels
- scikit-learn
- seaborn

Modélisation Statistique

- On peut programmer directement la régression linéaire simple ; c’est fait dans le programme regression-simple.py, disponible sur le forum.

- Ouvrir le programme sous Spyder, le faire tourner et essayer de comprendre son fonctionnement.

Modélisation Statistique

- Afin de voir comment ajuster différents modèles à un jeu de données, prenons l'exemple simple d'une régression quadratique, faiblement bruitée.

- Commençons avec les algorithmes disponibles sous numpy et ajustons un modèle linéaire, un modèle quadratique et un modèle cubique.

- Le code est le suivant :

Modélisation Statistique

```
import numpy as np

import matplotlib.pyplot as plt

x = np.arange(100)

y = 150 + 3*x + 0.03*x**2 + 5*np.random.randn((len(x)))

M1 = np.vstack ((np.ones_like(x),x)).T

M2 = np.vstack((np.ones_like(x),x,x**2)).T

M3 = np.vstack((np.ones_like(x), x, x**2, x**3)).T

p1 = np.linalg.lstsq(M1,y)

p2 = np.linalg.lstsq(M2,y)

p3 = np.linalg.lstsq(M3,y)
```

Modélisation Statistique

```
np.set_printoptions(precision=3)

print(‘The coefficients from the linear fit: {0}’.format(p1[0]))

print(‘The coefficients from the quadratic fit:
{0}’.format(p2[0]))

print(‘The coefficients from the cubic fit: {0}’.format(p3[0]))
```

Modélisation Statistique

- Si on veut déterminer l e « meilleur » ajustement, on peut avoir recours aux outils fournis par statsmodels pour ajuster le modèle. En effet, on obtient dans ce cas, non seulement les estimations des paramètres, mais bien d’autres informations sur le modèle.

```
import statsmodels.api as sm

Res1 = sm.OLS(y, M1).fit()

Res2 = sm.OLS(y, M2).fit()

Res3 = sm.OLS(y, M3).fit()

print(Res1.summary2())

print(Res2.summary2())

print(Res3.summary2())

print('La valeur AIC est {0:4.1f} pour la régression linéaire, {1:4.1f} pour la régression quadratique, {2:4.1f} pour la régression cubique'.format(Res1.aic,Res2.aic,Res3.aic))
```

Modélisation Statistique

- On peut également effectuer la régression, à l'aide de formules sous pandas, comme sous R, ce qui évite de construire explicitement les matrices de plan.

```
import pandas as pd

import statsmodels.formula.api as smf

df = pd.DataFrame({'x':x, 'y':y})

Res1F = smf.ols('y~x', df).fit()

Res2F = smf.ols('y~x+I(x**2)', df).fit()

Res3F = smf.ols('y~x+I(x**2)+I(x**3)',df).fit()

Res2F.params

Res2F.bse

Res2F.conf_int()
```

Exercice

- Lire les données de températures moyennes annuelles de la station météorologique autrichienne de Sonnblick à partir du fichier AvgTemp.xls, disponible sur le forum.

- Calculer le coefficient de corrélation de Pearson et de Spearman, puis le tau de Kendall, entre la variable de température et l'année.

- Calculer l'accroissement annuel de la température, en supposant que la dépendance de la température par rapport à l'année est linéaire. Cet accroissement est-il significatif ?

- Vérifier que le modèle de régression linéaire ajusté est satisfaisant en testant la normalité des résidus (par exemple, à l'aide du test de Kolmogorov-Smirnov).

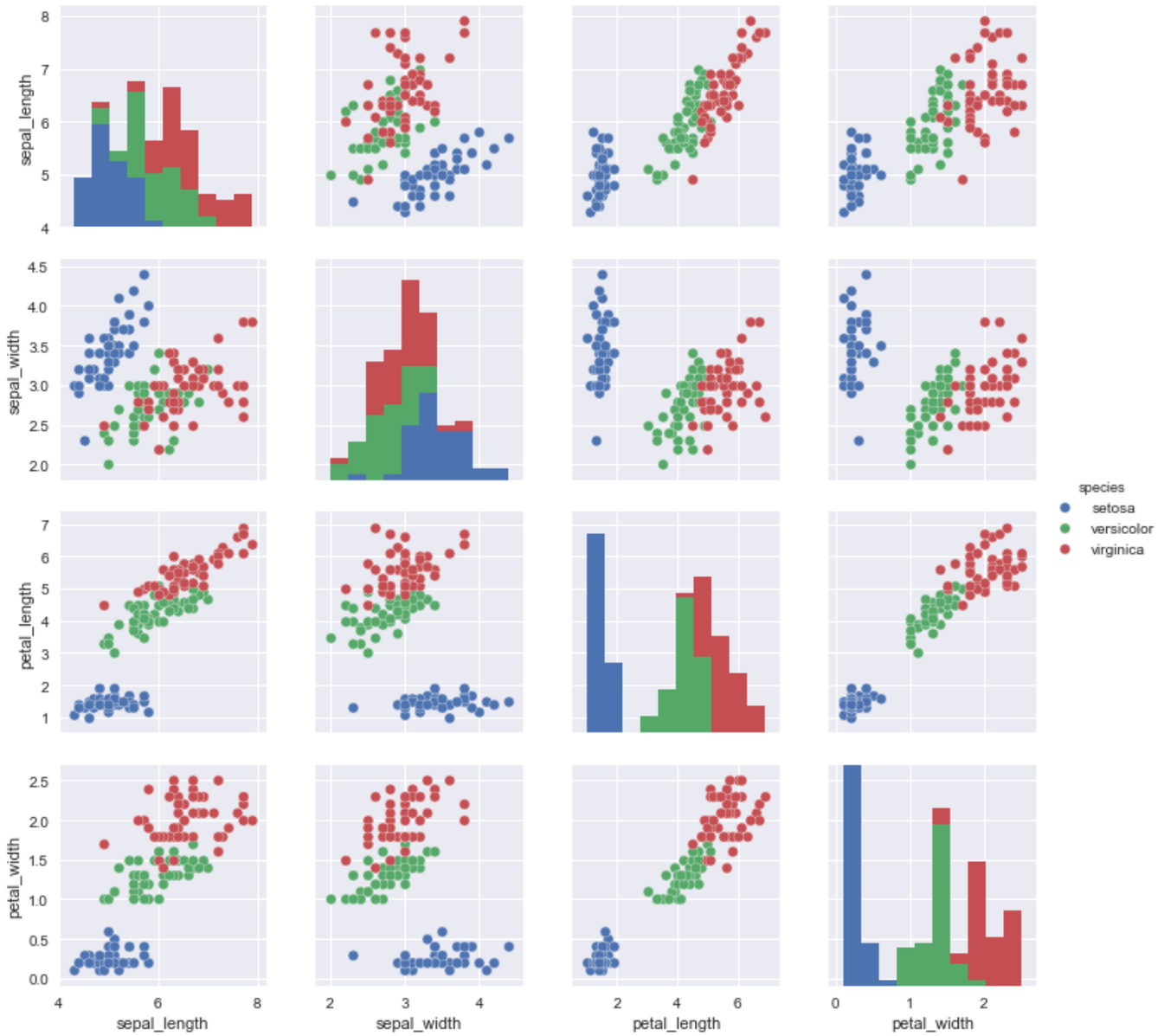
Matrice de nuages de points

- Si on souhaite étudier les corrélations de plusieurs variables, on peut utiliser une matrice de nuages de points.

- Exemple :

```
import seaborn as sns

sns.set()

df = sns.load_dataset(‘’iris”)

sns.pairplot(df, hue= ‘’species” , size = 2.5)
```

Matrice de nuages de points



Matrice des corrélations

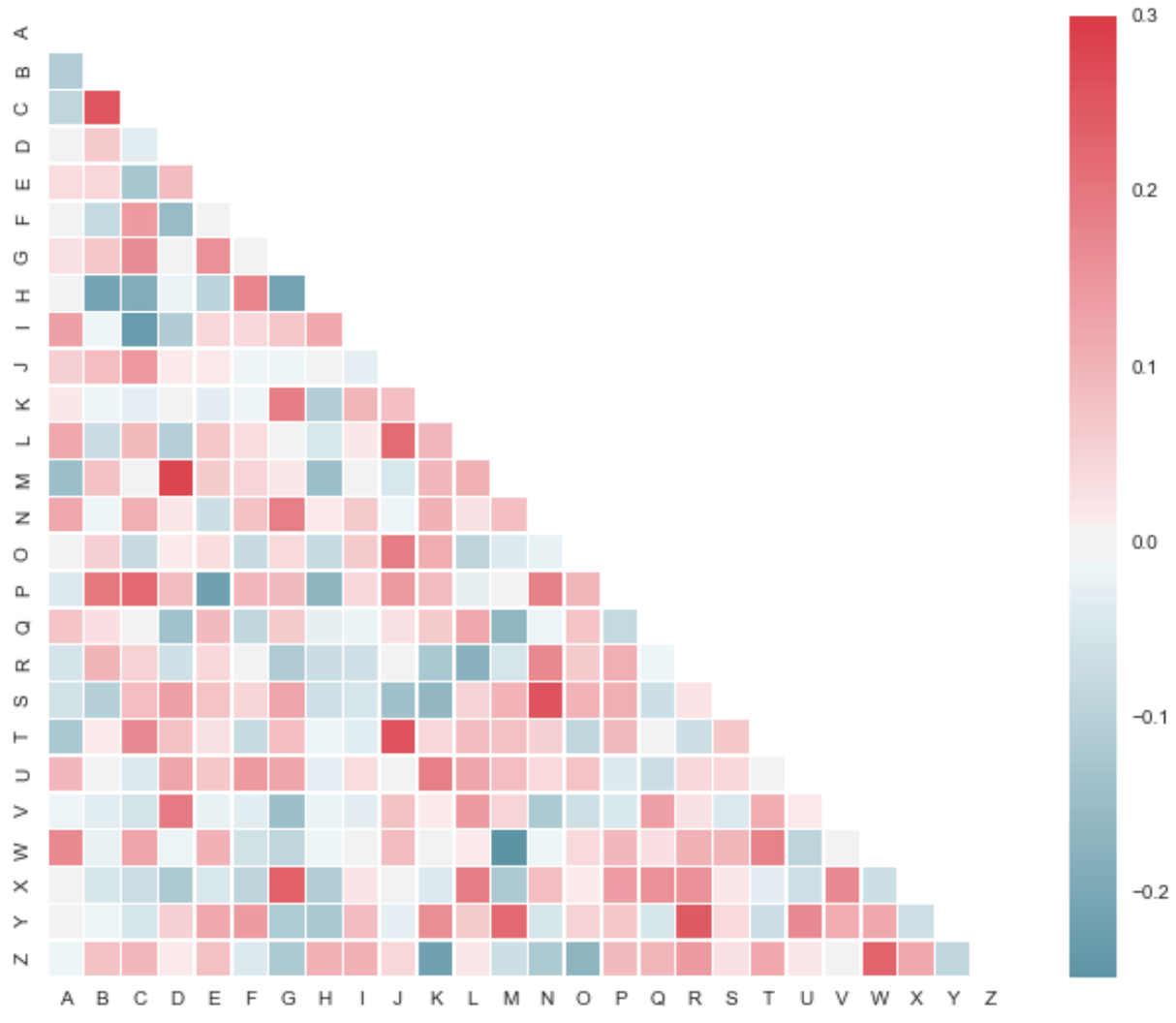
- Seaborn permet de visualiser de façon élégante une matrice de corrélations.
- Exemple : on simule 100 observations de chacune parmi 30 variables aléatoires distribuées selon une loi normale standard.

```
from string import ascii_letters

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

sns.set(style="white")

rs = np.random.RandomState(33) # Germe du générateur aléatoire

d = pd.DataFrame(data = rs.normal(size = (100,26)), columns = list(ascii_letters[26:]))
```

Matrice des corrélations

```
cor = d.corr()

mask = np.zeros_like(cor, dtype=np.bool)

mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize=(11, 9))

cmap = sns.diverging_palette(220, 10, as_cmap=True)

sns_plot = sns.heatmap(cor, mask=mask, cmap=cmap, vmax=.3, center=0,

            square=True, linewidths=.5, cbar_kws={"shrink": .5})

fig = sns_plot.get_figure()

fig.savefig("/Users/Salim/Desktop/corrmat.png")
```

Matrice des corrélations



Régression Logistique

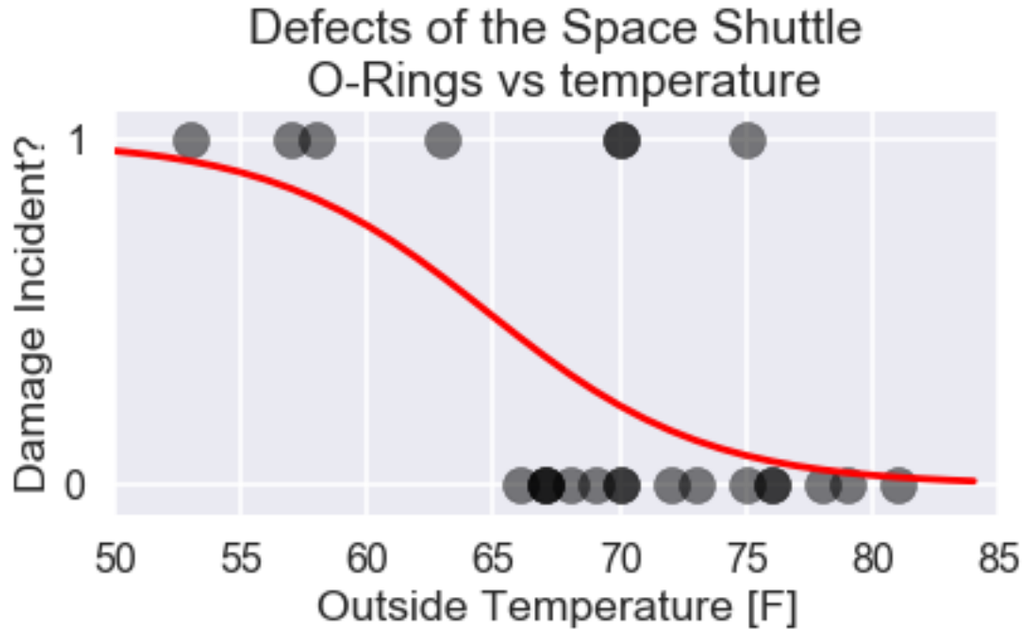
- On va prendre comme exemple l'évaluation de la probabilité d'une défaillance de l'O-ring (une pièce critique) en fonction de la température pour les navettes spatiales américaines.
- Le 28 janvier 1986, le 25ème vol de la navette spatiale américaine s'est terminé par un désastre lorsque l'un des moteurs de la navette a explosé, tuant les 7 membres d'équipage. La commission d'enquête a conclu à une défaillance de l'O-ring, cette défaillance étant due à une mauvaise conception de celui-ci. En effet, l'O-ring s'est avéré sensible à divers facteurs extérieurs, et parmi ceux-ci, la température.

Régression Logistique

- Sur les 24 vols précédents, des données étaient disponibles sur les défaillance de l'O-ring pour 23 vols (le 24ème avait été perdu en mer).
- La veille de l'accident, ces données avaient été discutées mais en se limitant aux 7 vols où les défaillances avaient eu des conséquences importantes, ne révélant pas de tendance particulière. La prise en compte de l'ensemble des données révèle toutefois, comme on va le voir, une tendance sur les défaillances, plus probables pour les températures basses.

Régression Logistique

- Récupérer le programme logistic.py sur le forum, le faire tourner et essayer de comprendre son fonctionnement. La dépendance mise en évidence est illustrée par le graphique suivant :



ACP

- On va illustrer l'ACP sous Python sur des données de cancer du sein.

- A cet effet, récupérer le programme acp.py sur le forum, le faire tourner et essayer de comprendre son fonctionnement.

Exercice

- Récupérer les données Iris sous Python, en faire une ACP sur deux composantes et représenter graphiquement le nuage de points en projection.

Bibliographie

- *An Introduction to Statistics with Python*, T. Halwanter, Springer 2016

- *Introduction to Machine Learning with Python,* A. C. Müller & S. Guido, O'Reilly 2017

- *Programmation en Python pour les Mathématiques*, 2ème ed., A. Casamayou-Boucau, P. Chauvin & G. Connan, Dunod 2016

- *Informatique avec Python*, S. Bays, Ellipses 2017

- *Introduction à l'Informatique*, S. Bays, Ellipses 2017