Bibliographie I

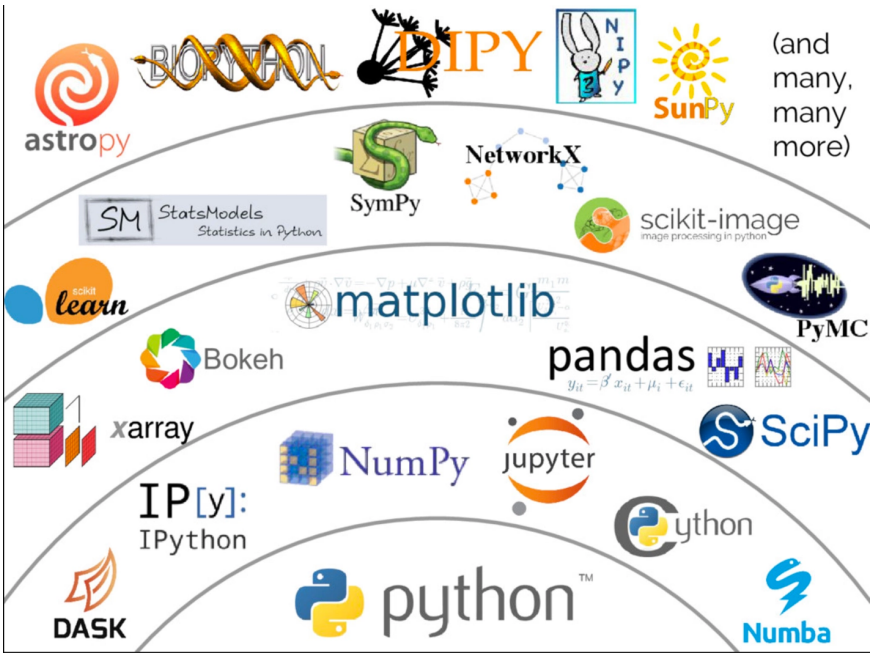
- GIL
  - http://www.dabeaz.com/python/UnderstandingGIL.pdf
  - https://en.wikipedia.org/wiki/Preemption_(computing)
  - https://lwn.net/Articles/612483/

Python scientifique

Écosystème

- écosystème très riche
- utilisé aussi bien par les scientifiques que les entreprises
- origine : développeurs et scientifiques
- sponsorisé par de grandes entreprises (google, facebook, Enthought, Anaconda Inc)
- tout ce que vous cherchez existe probablement déjà

Écosystème



Source : https://www.datacamp.com/community/blog/python-scientific-computing-case

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Performances
Dask
Bibliographie

Écosystème

Calculs :

- numpy [54]
- scipy [55]
- pandas [56]
- ...

Plotting :

- matplotlib [57]
- seaborn [58]
- bokeh [59]
- ...

54.http://www.numpy.org/
55.https://www.scipy.org/
56.https://pandas.pydata.org/
57.https://matplotlib.org/
58.https://seaborn.pydata.org/
59.https://bokeh.pydata.org/en/latest/

Python scientifique

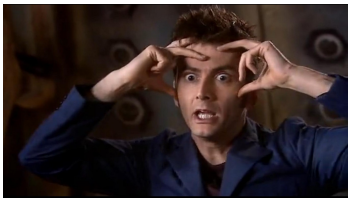
```
import numpy as np

xs = np.arange(-2*np.pi, 2*np.pi, 100)
ys = np.sin(xs) - 3*xs + 2

############################

A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
print(A.dot(B))
```

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Performances
Dask
Bibliographie

IPython – Jupyter

- `shell` interactif avec auto-complétion
- fonctions *magique* (mesure du temps, infos shell...)
- `notebook` (et maintenant `lab`) → programmation littérale, IDE en ligne
- utilisation d'autres "noyaux" (R, Julia, C, Haskell...)
- calcul parallèle
- présentation des résultats
- ...



Présentation

Paquet fondateur de la `stack` scientifique Python.

- propose un type de tableaux multidimensionnels
- met les performances au premier plan
- manipulation vectorielles / matricielles faciles
- outils pour manipuler ces tableaux (algèbre linéaire, traitement du signal, ...)

Utilise des bibliothèques `Fortran` ou `C/C++` → bonnes performances

Tableaux numpy

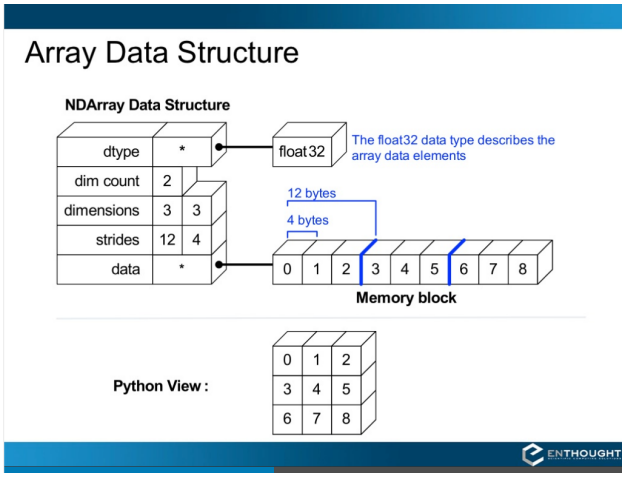
- tableaux multidimensionnels (`NDarray`)
- facilement indexable
- vectorisation du code (`UFunc` et `broadcasting`)

Même manipulation que `Matlab`

```
import numpy as np

xs = np.arange(-2*np.pi, 2*np.pi, 100)
ys = np.sin(xs) - 3*xs + 2

#########################

A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
print(A.dot(B))
```


Structure

- un seul type de données par tableau (ou objet python)
- métadonnées + données linéaires
- orientation (C ou Fortran) → important pour les performances



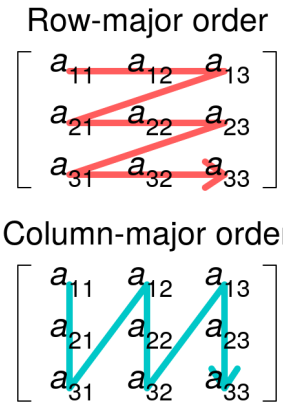
`https://www.slideshare.net/enthought/numpy-talk-at-siam`

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Présentation
Tableaux
Structure des tableaux
Création des tableaux
Manipulations usuelles
Broadcasting
Fonctions universelles
Scipy
Pandas
Sympy
Analyse de réseaux

Structure

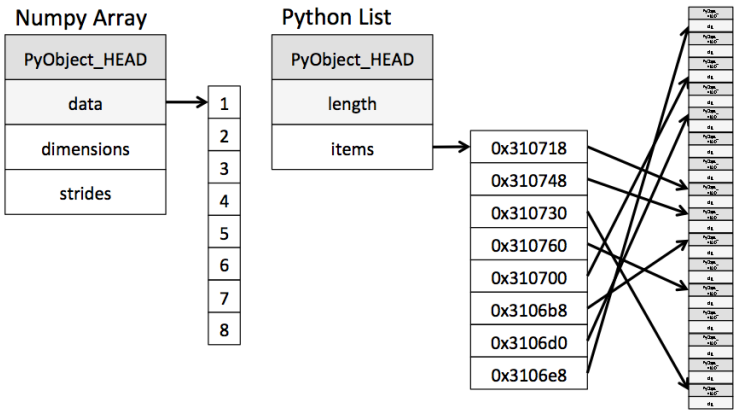
- un seul type de données par tableau (ou objet python)
- métadonnées + données linéaires
- orientation (C ou Fortran) → important pour les performances



By Cmglee - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=65107030

Structure

- un seul type de données par tableau (ou objet python)
- métadonnées + données linéaires
- orientation (C ou Fortran) → important pour les performances



https://jakevdp.github.io/PythonDataScienceHandbook/02.01-understanding-data-types.html

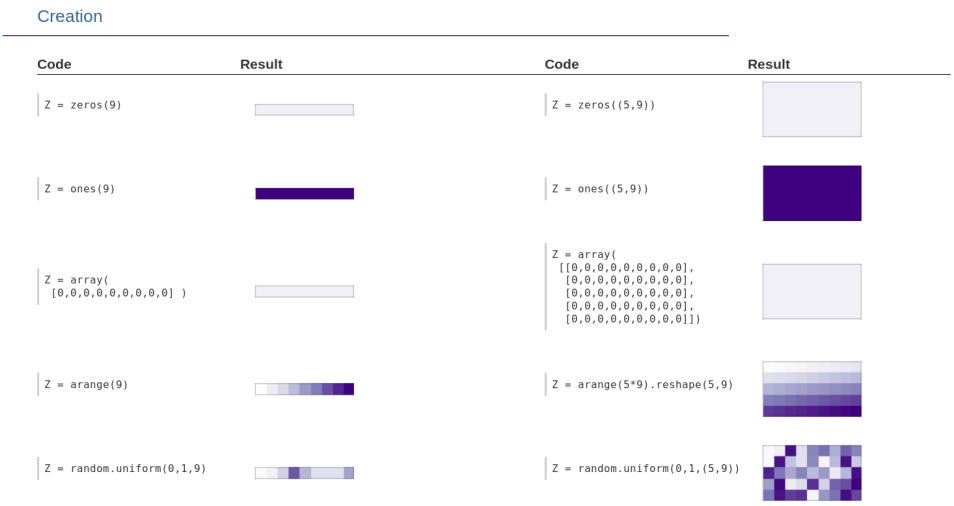
Matthieu Falce
Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Présentation
Tableaux
Structure des tableaux
Création des tableaux
Manipulations usuelles
Broadcasting
Fonctions universelles
Scipy
Pandas
Sympy
Analyse de réseaux

Création des tableaux

```python
import numpy as np

# à partir d'une liste / liste de liste...
xs = np.array([i for i in range(10)])
A = np.array([[1, 2], [3, 4]])

# équivalent de range
rs = np.arange(10, 50, 2)

# valeurs régulièrement espacées
es = np.linspace(-3.65, np.pi, 100)

# forcer les types
ts = np.linspace(-3.65, np.pi, 100, dtype=np.int)
ts2 = np.linspace(-3.65, np.pi, 100, dtype=np.complex) + 2j

# tous les utilitaires classiques
ones = np.ones((3, 1))
diag = np.eye(3)
full = np.full((3, 2), np.pi)
```


Création des tableaux

Matthieu Falce

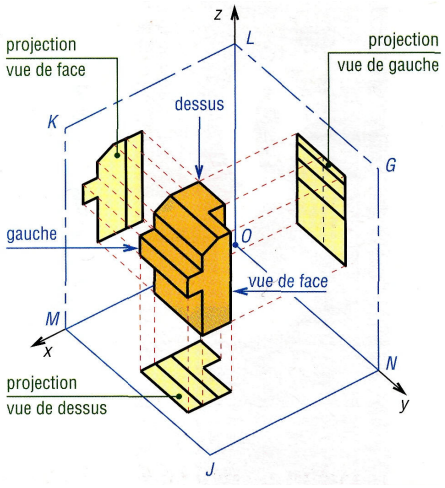


http://www.labri.fr/perso/nrougier/teaching/numpy/numpy.html

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Présentation
Tableaux
Structure des tableaux
Création des tableaux
Manipulations usuelles
Broadcasting
Fonctions universelles
Scipy
Pandas
Sympy
Analyse de réseaux

Changements de forme / dimensions

Certaines opérations vont s'opérer sur certaines dimensions.
On peut les visualiser comme étant des projections.



http://www.zpag.net/Tecnologies_Indistrielles/projections_orthogonales_normali.htm

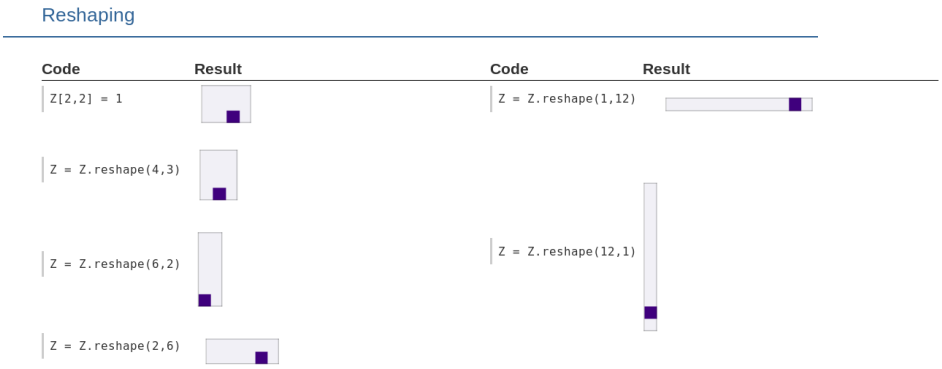
Changements de forme / dimensions

```python
import numpy as np

# changement du nombre de dimensions
A = np.arange(1, 3 * 4 + 1).reshape((3, 4))
print(A)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]

# on peut effectuer des actions selon
# certaines dimensions
B = np.arange(1, (3 * 4 * 5) + 1).reshape((3, 4, 5))
B.mean(axis=0)
B.mean(axis=1)
B.mean(axis=2)

B.sum(axis=1)
```


Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Présentation
Tableaux
Structure des tableaux
Création des tableaux
Manipulations usuelles
Broadcasting
Fonctions universelles
Scipy
Pandas
Sympy
Analyse de réseaux

Changements de forme / dimensions



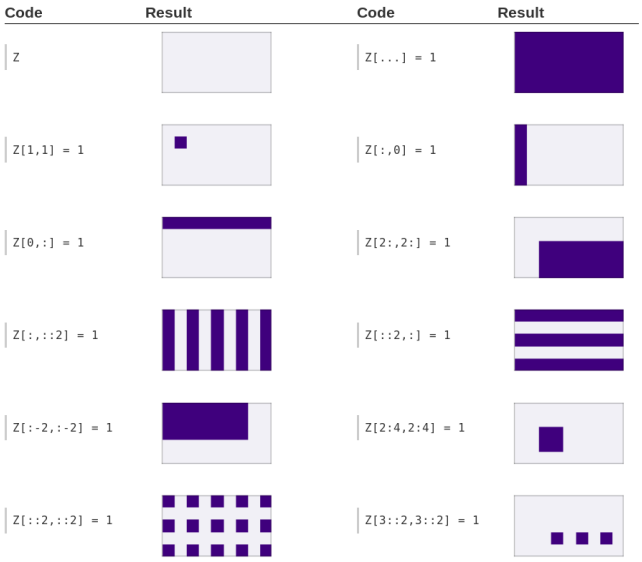
http://www.labri.fr/perso/nrougier/teaching/numpy/numpy.html

Indexing

```python
import numpy as np

# découpage
D = np.arange((100 * 5)).reshape((100, 5))
split = 30
test, train = D[:split, :], D[split:, :]


# indéxing booléen
# on met toutes les valeurs paires à 0
A = np.arange(1, 3 * 4 + 1).reshape((3, 4))
pairs = (A % 2 == 0)
pairs = pairs.astype(bool)
A[pairs] = 0

# slicing et réassignation partielle
B = np.arange(100, dtype=np.uint8).reshape((10, 10))
B[::2, ::2] = B[1::2, 1::2] / 2
```

Indexing

Slicing

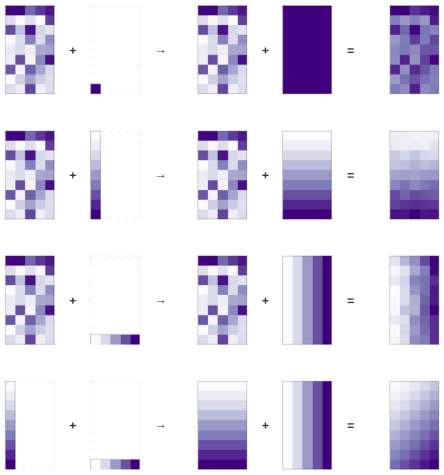


http://www.labri.fr/perso/nrougier/teaching/numpy/numpy.html

Broadcasting

Le broadcasting permet de manipuler entre eux des tableaux de tailles différentes.

Broadcasting



http://www.labri.fr/perso/nrougier/teaching/numpy/numpy.html

Broadcasting

Le `broadcasting` permet de manipuler entre eux des tableaux de tailles différentes.

Ce n'est pas magique

```
In [10]: A = np.arange(9).reshape((3, 3))
In [11]: B = np.arange(4)
In [12]: A + B

---------------------------------------------------

ValueError        Traceback (most recent call last)
<ipython-input-12-151064de832d> in <module>()
----> 1 A + B
ValueError: operands could not be broadcast
together with shapes (3,3) (4,)
```


Fonctions universelles

Fonctions universelles s'appliquent sur les éléments d'un tableau de façon vectorisée (sans boucles apparentes).

Exemple : `y = np.sin(x)`

On peut créer ses propres fonctions vectorisées avec `np.frompyfunc` ou `np.vectorize`.

Ce n'est pas pour ça qu'elles seront plus rapides.

Présentation

Méthodes additionnelles à `Numpy` pour le calcul scientifique.

- fonctions spéciales
- intégration
- optimisation
- équations différentielles
- traitement du signal
- algèbre linéaire
- ...

Si des routines sont partagées avec `Numpy` → plus complètes dans `Scipy` [60]

`Scipy` utilise LAPACK, `Numpy` pas forcément [61]

60.https://docs.scipy.org/doc/numpy/reference/routines.dual.html

61.https://www.scipy.org/scipylib/faq.html#what-is-the-difference-between-numpy-and-scipy

Scikits

SciPy Toolkits → extensions pour `Scipy` indépendantes

Liste non exhaustive [62]

- `scikit-learn`
- `scikit-image`
- `scikit-bio`
- ...

62.liste complète : https://scikits.appspot.com/scikits

FFT [63]

```python
import numpy as np
import matplotlib.pyplot as plt

t = np.arange(256)
sig = np.sin(t)

# sp est un tableau de complexes
sp1 = np.fft.fft(sig)
freq1 = np.fft.fftfreq(t.shape[-1])
plt.plot(freq1, sp1.real, freq1, sp1.imag)
plt.show()

# comparaisons fréquences
sig2 = np.sin(2 * t)
sp2 = np.fft.fft(sig2)
freq2 = np.fft.fftfreq(t.shape[-1])

plt.plot(freq2, sp2.real, label="F2")
plt.plot(freq1, sp1.real, label="F1")
plt.legend()
plt.show()
```

63.https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fft.html#numpy.fft.fft

Optimisation [64]

```python
from scipy.optimize import minimize

minimize(lambda x: x**2, x0=1000)

#      fun: 5.713415792109052e-17
# hess_inv: array([[0.50000012]])
#      jac: array([-2.16266884e-10])
#  message: 'Optimization terminated successfully.'
#     nfev: 24
#      nit: 3
#     njev: 8
#   status: 0
#  success: True
#        x: array([-7.55871404e-09])
```

64.https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html

Optimisation [64]

```python
from scipy.optimize import minimize

minimize(lambda x: x[0]**2 + x[1] ** 2, x0=(1000, 33))

 #      fun: 1.3011523813214424e-13
 # hess_inv: array([[0.54198343, 0.00254437],
 #        [0.00254437, 0.5001542 ]])
 #      jac: array([7.35719908e-07, 4.45876263e-08])
 #  message: 'Optimization terminated successfully.'
 #     nfev: 40
 #      nit: 5
 #     njev: 10
 #   status: 0
 #  success: True
 #        x: array([3.60409374e-07, 1.48432326e-08])
```

On peut mettre des contraintes sur les paramètres et changer de méthode d'optimisation aussi.

64.https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html

Interpolation [65]

- en 2D
- splines et courbes paramétrées
- ...

```python
import numpy as np
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt

x = np.linspace(0, 10, num=11, endpoint=True)
y = np.cos(-x ** 2 / 9.0)
f = interp1d(x, y)
f2 = interp1d(x, y, kind='cubic')

xnew = np.linspace(0, 10, num=41, endpoint=True)
plt.plot(x, y, 'o', xnew, f(xnew), '-', xnew, f2(xnew), '--')
plt.legend(['data', 'linear', 'cubic'], loc='best')
plt.show()
```

65.https://docs.scipy.org/doc/scipy/reference/tutorial/interpolate.html

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Présentation
Scikits
Fonctionnalités
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Performances
Dask

Intégration – équations différentielles [66] [67]

- en 1D
- en 2D
- ... jusqu'à N dimensions
- paramétrages complexes
- ODE non `stiff` et `stiff`, pas adaptatif...

66. https://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html
67. https://docs.scipy.org/doc/scipy/reference/integrate.html

Intégration – équations différentielles [66] [67]

```python
from scipy.integrate import quad

def fonction_a_integrer(x, a, b):
    return a * x ** 3 + b

a = 1
b = 0
surface = quad(
        fonction_a_integrer,
        -10,
        10,
        args=(a, b)
)
print(surface)
```

66. https://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html
67. https://docs.scipy.org/doc/scipy/reference/integrate.html

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Présentation
Scikits
Fonctionnalités
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Performances
Dask

Intégration – équations différentielles [66] [67]

```python
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

def carre_der(t, x):
    return 2*x

ts = np.arange(10)
ys = odeint(carre_der, 2, ts)
plt.plot(ts, ys)
plt.show()
```

66. https://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html
67. https://docs.scipy.org/doc/scipy/reference/integrate.html

Traitement signal [69]

"Tout" pour le traitement du signal :[68]

- convolutions
- conceptions / analyses de filtres (FIR, FII) / analyse réponse
- analyses de spectres
- détections de pics

68. https://docs.scipy.org/doc/scipy/reference/signal.html#module-scipy.signal
69. https://docs.scipy.org/doc/scipy/reference/tutorial/signal.html

Algebre linéaire [70]

Fonctions potentiellement légèrement différentes de celles de `Numpy`
Compilées avec BLAS et LAPACK

70.https://docs.scipy.org/doc/scipy/reference/tutorial/linalg.html

Matrices creuses [71] [72]

Numpy supporte les matrices creuses de différents types :

- csc_matrix: Compressed Sparse Column format
- csr_matrix: Compressed Sparse Row format
- bsr_matrix: Block Sparse Row format
- lil_matrix: List of Lists format
- dok_matrix: Dictionary of Keys format
- coo_matrix: COOrdinate format (aka IJV, triplet format)
- dia_matrix: DIAgonal format

Utiliser les méthodes de `scipy.sparse.linalg` pour faire des opérations et garder des matrices creuses. Selon le type de matrices utilisées ; différentes possibilités (perte du slicing...)

71.https://docs.scipy.org/doc/scipy-0.14.0/reference/sparse.html
72.https://docs.scipy.org/doc/scipy/reference/tutorial/arpack.html

Lois statistiques [73] [74]

- distributions discrètes et continues
- analyses de données statistiques (kurtosis, zscores...)
- tests statistiques
- ...

73. https://docs.scipy.org/doc/scipy/reference/tutorial/linalg.html
74. https://docs.scipy.org/doc/scipy/reference/stats.html#module-scipy.stats

Manipulation d'images

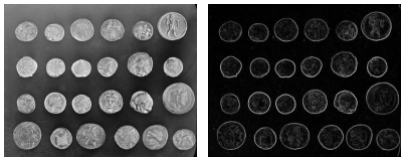
Les images (rasterisées ou pixelisées) sont des tableaux *numpy* classiques.
Voici les principales bibliothèques pour en faire :

- *Pillow* [75] : plutôt utilisée pour les manipulations classiques (redimensionnement, rotation, changement de format, ...)
- *numpy* / *scipy* classique [76] : permet de manipuler les tableaux de données de pixels directement
- *scikit image* [77] [78] : Propose de nombreux algorithmes pour faire du traitement d'images
- *openCV* [79] : le port de la Bibliothèque d'analyse d'image openCV. Permet de travailler en temps réel (à partir d'une caméra par exemple)

75. https://pillow.readthedocs.io/en/stable/
76. Plus d'infos ici : http://scipy-lectures.org/advanced/image_processing/
77. https://scikit-image.org/
78. https://scipy-lectures.org/packages/scikit-image/index.html
79. https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

Manipulation d'images

```python
from skimage import data, io, filters

image = data.coins()
edges = filters.sobel(image)
io.imshow(edges)
io.show()
```



Manipulation d'images avec scikit image.
Source : https://scikit-image.org/

Manipulation d'images

```python
import cv2 as cv

# Read image from your local file system
original_image = cv.imread("path/to/your-image.jpg")

# Convert color image to grayscale for Viola-Jones
grayscale_image = cv.cvtColor(original_image, cv.COLOR_BGR2GRAY)

# Load the classifier and create a cascade object for face detection
face_cascade = cv.CascadeClassifier("path/to/haarcascade_frontalface_alt.xml")

detected_faces = face_cascade.detectMultiScale(grayscale_image)

for (column, row, width, height) in detected_faces:
    cv.rectangle(
        original_image, (column, row), (column + width, row + height), (0, 255, 0), 2
    )

cv.imshow("Image", original_image)
cv.waitKey(0)
cv.destroyAllWindows()
```

Détection de visages avec openCV.
Source : https://realpython.com/traditional-face-detection-python/

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Présentation
Scikits
Fonctionnalités
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Performances
Dask

Présentation

Bibliothèque essentielle à l'analyse de données.

Données structurées (lignes / colonnes à la SQL) et séries temporelles.

Dataframes et séries

- `series` : suite de données à 1 dimension (comme un tableau `numpy` avec d'autres fonctionnalités)
- `dataframes` : regroupement de plusieurs séries de même taille (comme un tableur)

Manipulations de dataframes

Création de dataframes et de séries

```python
import numpy as np
import pandas as pd

# créer une série
s = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])


# créer un dataframe
d = {
    "one": pd.Series([1.0, 2.0, 3.0], index=["a", "b", "c"]),
    "two": pd.Series([1.0, 2.0, 3.0, 4.0], index=["a", "b", "c", "d"]),
}
df = pd.DataFrame(d)
df["three"] = s

print("description de df :")
df.describe()
```


Manipulations de dataframes

Indexation et accès aux données

```python
# https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html

import numpy as np
import pandas as pd

# créer une série
s = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])
df = pd.DataFrame({"one": s, "two": s[1:], "three": s[2:]})

# accès aux colonnes
one, two = df["one"], df.two
df[["one", "two"]]

# accès aux lignes
df[:1]  # slicing
a, bcd, adb = df.loc["a"], df.loc["b":], df.loc[["a", "b", "d"]]
df.iloc[2:]
type(df[1:2])  # pandas.core.frame.DataFrame
type(df.iloc[1])  # pandas.core.series.Series


# accès aux éléments
df.loc["a", "one"]  # index ligne, colonne

# échantillonage
seed = 1
df.sample(n=3, replace=True, random_state=seed)
```

Manipulations de dataframes

Aparté sur les performances d'indexation

```
## vitesse
# # bad practice
# In [89]: %timeit df["one"]["a"]
# 8.9 µs ± 107 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

# # high level loc
# In [90]: %timeit df.loc["a", "one"]
# 6.6 µs ± 230 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

# # low level at
# In [91]: %timeit df.at["a", "one"]
# 3.88 µs ± 33.3 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```


Manipulations de dataframes

Lecture de CSV et nettoyage de données

```
import pandas as pd
import matplotlib.pyplot as plt


url = ("http://facweb.cs.depaul.edu/mobasher/classes"
       "/csc478/Data/titanic-trimmed.csv")
titanic = pd.read_csv(url)
titanic.head(10)

age_mean = titanic.age.mean()
titanic.age.fillna(age_mean, axis=0, inplace=True)
titanic.dropna(axis=0, inplace=True)
titanic.age.describe()

titanic.age.plot.kde()
plt.show()
```


Matthieu Falce

Vue d'ensemble

Langage Python

Programmation Orientée objet (POO)

Bonnes pratiques

Bibliothèque standard

Création de modules

Programmation concurrente

Python scientifique

Écosystème scientifique

Jupyter

Numpy

Scipy

Pandas

Présentation

Dataframes

Manipulations de dataframes

Sympy

Analyse de réseaux

Machine learning / statistiques

Graphiques

Performances

Manipulations de dataframes

Exemples de graphiques possibles

```
import numpy as np
import pandas as pd
from pandas.plotting import lag_plot
from pandas.plotting import autocorrelation_plot
from matplotlib import pyplot as plt

xs = np.linspace(-1, 1, 100)
ys = np.cos(xs)
ys2 = np.random.random(100)

df = pd.DataFrame({"cos": ys, "rand": ys2})

lag_plot(df.cos)
plt.show()
lag_plot(df.rand)
plt.show()

autocorrelation_plot(df.rand)
plt.show()
autocorrelation_plot(df.cos)
plt.show()

ax = df.cos.plot()
fig = ax.get_figure()
fig.savefig("cos.png")
```


Manipulations de dataframes

Opération sur des groupes de données

```
import pandas as pd
import numpy as np

df = pd.DataFrame(
    [
        ("bird", "Falconiformes", 389.0),
        ("bird", "Psittaciformes", 24.0),
        ("mammal", "Carnivora", 80.2),
        ("mammal", "Primates", np.nan),
        ("mammal", "Carnivora", 58),
    ],
    index=["falcon", "parrot", "lion", "monkey", "leopard"],
    columns=("class", "order", "max_speed"),
)

grouped1 = df.groupby("class")
grouped2 = df.groupby("order", axis="columns")
grouped3 = df.groupby(["class", "order"])

for n, g in grouped3:
    print(n)
    print(g)
    print(type(g))
    print("")
```

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Présentation
Dataframes
Manipulations de dataframes
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Performances

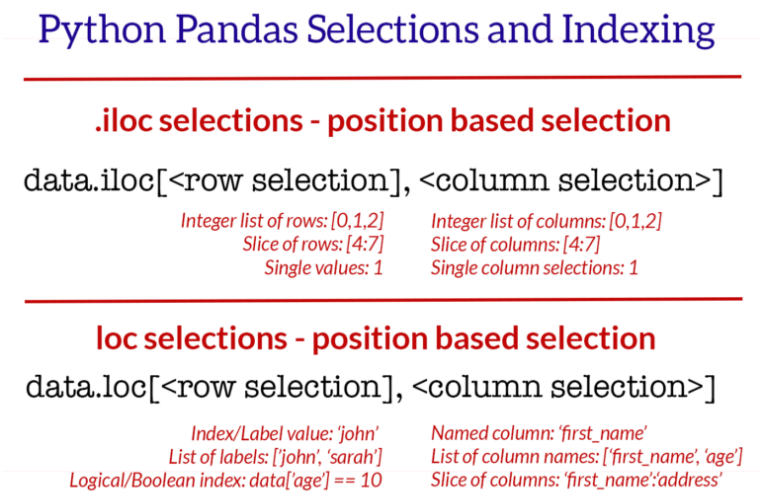
Manipulations de dataframes

Manipulation de séries temporelles

```python
import pandas as pd
import numpy as np

# time index
dti = pd.date_range("2018-01-13", periods=3, freq="H")
dti = dti.tz_localize("UTC")
dti.tz_convert("US/Pacific")

## offsets
# https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#timeseries-offset-aliases
start, end = "2019-01-12", "2019-12-25"
pd.date_range(start, end, freq="BM")

# conversion
## https://docs.python.org/3/library/datetime.html#strftime-and-strptime-behavior
pd.to_datetime("12-11-2010 00:00", format="%d-%m-%Y %H:%M")

# resampling
idx = pd.date_range("2018-01-01", periods=48, freq="H")
ts = pd.Series(range(len(idx)), index=idx)
ts.resample("2H").mean()

s = pd.Series(range(len(idx)), index=idx)
for i in s.resample("6H"):
    print(i)
```


Indexation (représentation graphique)

Plus d'informations sur les différents modes d'accès ici : https://stackoverflow.com/questions/28757389/pandas-loc-vs-iloc-vs-at-vs-iat



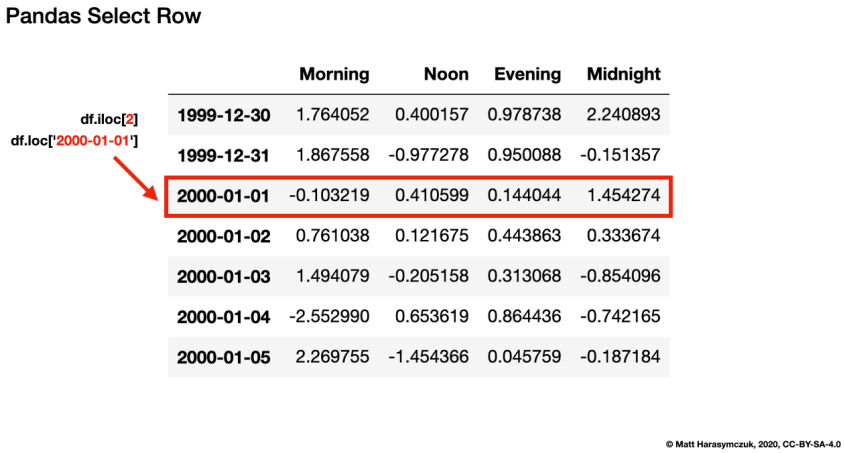
Source : https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Présentation
Dataframes
Manipulations de dataframes
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Performances

Indexation (représentation graphique)

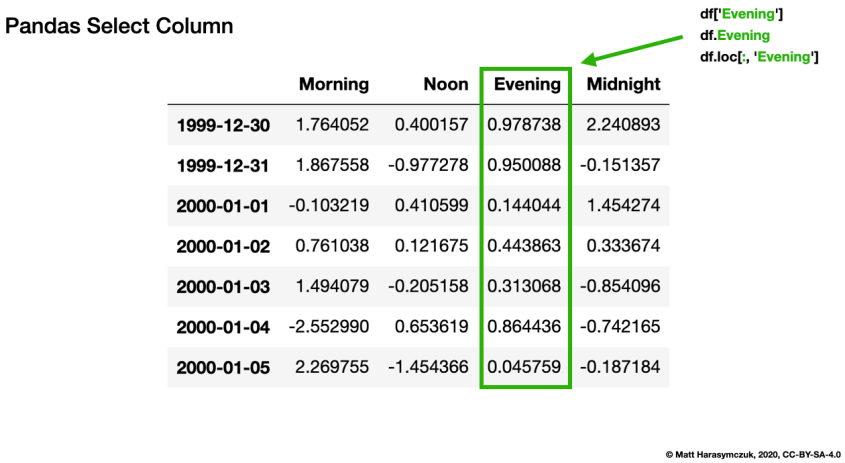
Plus d'informations sur les différents modes d'accès ici : https://stackoverflow.com/questions/28757389/pandas-loc-vs-iloc-vs-at-vs-iat



Source : https://python.astrotech.io/pandas/dataframe/loc.html

Indexation (représentation graphique)

Plus d'informations sur les différents modes d'accès ici : https://stackoverflow.com/questions/28757389/pandas-loc-vs-iloc-vs-at-vs-iat



Source : https://python.astrotech.io/pandas/dataframe/loc.html

Présentation

Sympy[75] est une bibliothèque permettant le calcul symbolique :

- simplification d'expression
- analyse (dérivation, intégration)
- affichage des sorties en LaTeX

75.http://www.sympy.org

Calcul symbolique

```python
from sympy import *

# permet l'affichage des formules
init_session()

# on déclare des variables
x, a, b = symbols("x a b")

# on défini une intégrale
a = Integral(cos(x) * exp(x), x)
print(a)
latex(a)  # on affiche son code latex

# on va simplifier des expressions
simplify(sin(x) ** 2 + cos(x) ** 2)
simplify(x ** a * x ** b)
```

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Performances
Dask
Bibliographie

Présentation

Il existe plusieurs bibliothèques pour manipuler des graphes (réseaux) en python :

- `igraph` [76]
- `networkx` [77]
- `graph-tool` [78]

La plus connue est networkX mais peut être lente (codée en pur python), les autres sont potentiellement plus rapide.

76.http://igraph.org
77.http://networkx.github.io
78.http://graph-tool.skewed.de

Manipulation de graphes

```python
import networkx as nx

# on crée un graphe en 2 parties
G = nx.Graph()
G.add_edges_from([(1, 2), (1, 3)])
G.add_node(4)

# on calcule des propriétés du graphe
nx.connected_components(G)
list(nx.connected_components(G))
sorted(d for n, d in G.degree)
nx.clustering(G)
```

Etat des lieux

Il existe plusieurs bibliothèques dédiées apprentissage :

- modèles statistiques (régressions, tests statistiques, méthodes sur séries temporelles) : `statsmodels` [79]
- algorithmes de *machine learning* : `scikit-learn` [80]
- *curve feating* : `prophet`[81] (analyse de séries temporelles)
- *deep learning* : `tensorflow`[82], `keras`[83]

79.https://www.statsmodels.org/stable/index.html
80.https://scikit-learn.org/stable/
81.https://facebook.github.io/prophet/
82.https://www.tensorflow.org/
83.https://keras.io/

Graphiques

1. amélioration et diversification des outils
2. réalisation de graphiques de qualité
3. écosystème riche (seuls `R` et `Javascript` ont de meilleures bibliothèques à mon avis)

Matplotlib

Bibliothèques de `plot` la plus célèbre

API inspirée de `Matlab`

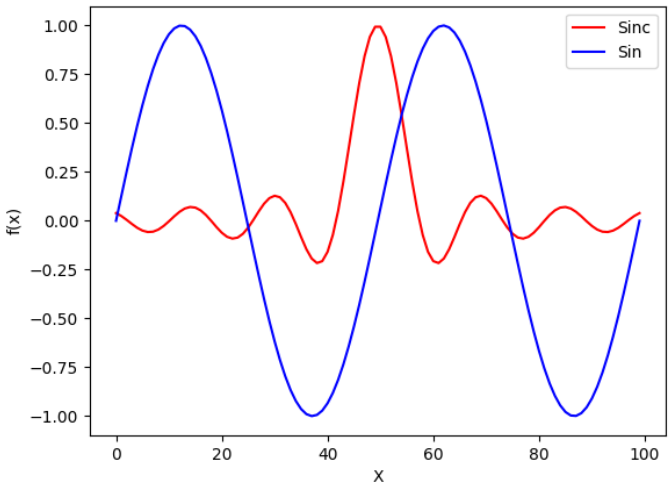
```
from matplotlib import pyplot as plt
import numpy as np

xs = np.linspace(-2*np.pi, 2*np.pi, 100)
ys1 = np.sinc(xs)
ys2 = np.sin(xs)

plt.plot(ys1, c="r", label="Sinc")
plt.plot(ys2, c="b", label="Sin")
plt.xlabel("X")
plt.ylabel("f(x)")
plt.legend()
plt.show()
```

`plt` agit comme une machine à état

Matplotlib

Bibliothèques de `plot` la plus célèbre

API inspirée de `Matlab`
Le style aussi



Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Matplotlib
Personnalisation
Seaborn
Concurrents 2D
Graphiques 3D

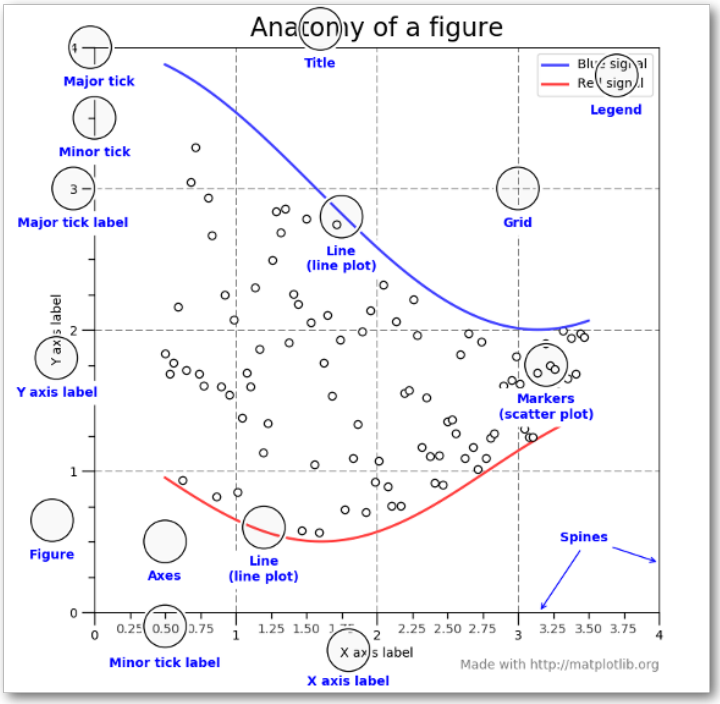
Matplotlib – types de plots

`Matplotlib` est une bibliothèque graphique → peut tout faire.

Si l'on s'en donne la peine

- lineplot
- scatterplot
- cartographie
- hexbin
- nuages de mots
- 3D
- animations
- ...

Anatomie d'une figure



`https://matplotlib.org/faq/usage_faq.html`

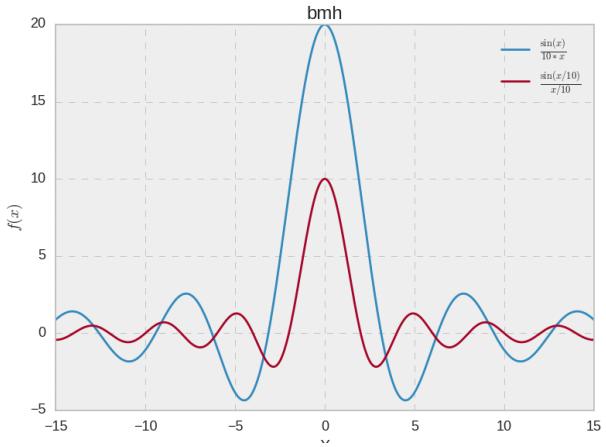
Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Matplotlib
Personnalisation
Seaborn
Concurrents 2D
Graphiques 3D

Personnalisation

Vous pouvez surcharger la configuration de `Matplolib` :

- passer des paramètres à la fonction de dessin
- changer de style
- modifier le `pyplot.rcParams`

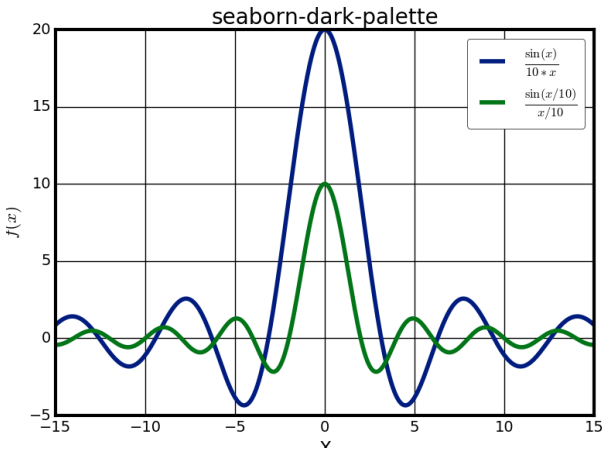


Thème : bmh

Personnalisation

Vous pouvez surcharger la configuration de `Matplolib` :

- passer des paramètres à la fonction de dessin
- changer de style
- modifier le `pyplot.rcParams`



Thème : seaborn-dark-palette

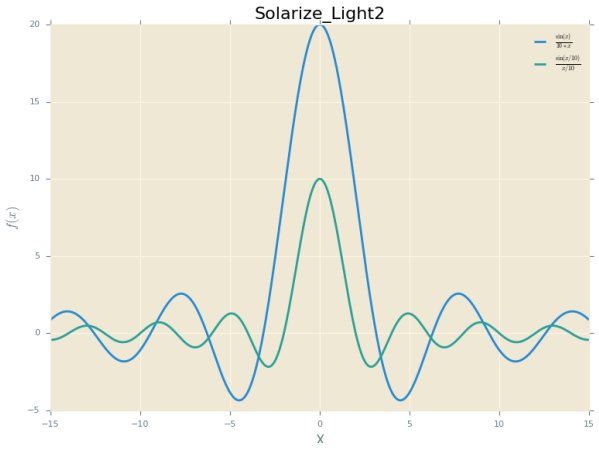
Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Matplotlib
Personnalisation
Seaborn
Concurrents 2D
Graphiques 3D

Personnalisation

Vous pouvez surcharger la configuration de `Matplolib` :

- passer des paramètres à la fonction de dessin
- changer de style
- modifier le `pyplot.rcParams`



Thème : Solarize_Light2

Personnalisation

Vous pouvez surcharger la configuration de `Matplolib` :

- passer des paramètres à la fonction de dessin
- changer de style
- modifier le `pyplot.rcParams`

```python
from matplotlib import pyplot as plt
import numpy as np

plt.rcParams['font.family'] = 'serif'
plt.rcParams['font.serif'] = 'Ubuntu'
plt.rcParams['ytick.labelsize'] = 8
plt.rcParams['legend.fontsize'] = 10
plt.rcParams['figure.titlesize'] = 20
plt.rcParams['lines.linewidth'] = 10

xs = np.linspace(-15, 15, 1000)
ys1 = 20 * np.sin(xs) / xs
ys2 = 10 * np.sinc(xs / 2)

plt.plot(xs, ys1, label=r"$\frac{\sin(x)}{10 * x}$")
plt.plot(xs, ys2, label=r"$\frac{\sin(x/10)}{x/10}$")

plt.title("Courbes")
plt.legend()
plt.xlabel("X")
plt.ylabel("$f(x)$")
plt.title("Surcharge rcParams")
plt.savefig("styles/rcParams.png")
plt.show()
```

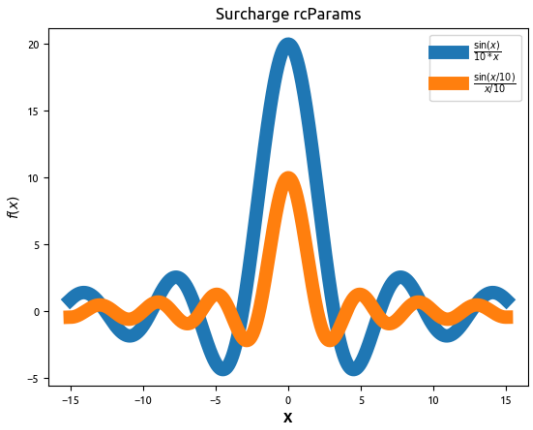
Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Matplotlib
Personnalisation
Seaborn
Concurrents 2D
Graphiques 3D

Personnalisation

Vous pouvez surcharger la configuration de `Matplolib` :

- passer des paramètres à la fonction de dessin
- changer de style
- modifier le `pyplot.rcParams`



Surcharge de rcParams

Seaborn

Seaborn se base sur `matplotlib`.

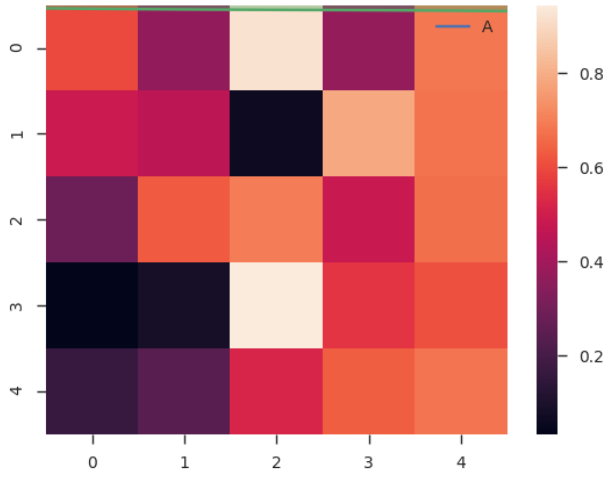
Il rajoute des styles et des fonctionnalités (surtout utile en statistiques).

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

R = np.random.random((5, 5))
sns.heatmap(R)
plt.savefig("sns_heatmap.png")


plt.clf()
A = np.random.normal(10, 1, 100)
B = np.random.normal(6, 5, 100)
sns.boxplot(x=["A", "B"], y=[A, B])
plt.savefig("sns_boxplot.png")


plt.clf()
sns.kdeplot(A, shade=True, label="A")
sns.distplot(B, label="B")
plt.legend()
plt.savefig("sns_distplot.png")
```

Seaborn

Seaborn se base sur `matplotlib`.

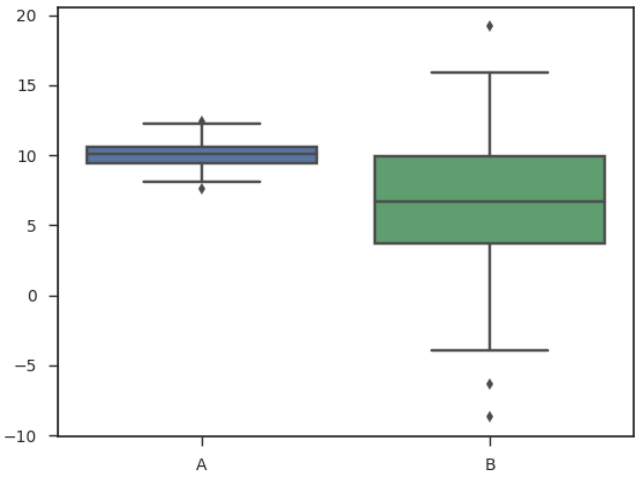
Il rajoute des styles et des fonctionnalités (surtout utile en statistiques).



Seaborn

Seaborn se base sur `matplotlib`.

Il rajoute des styles et des fonctionnalités (surtout utile en statistiques).



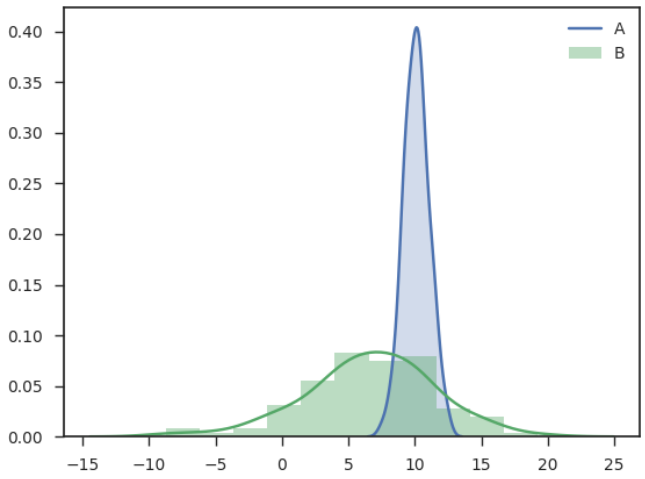
Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Matplotlib
Personnalisation
Seaborn
Concurrents 2D
Graphiques 3D

Seaborn

Seaborn se base sur `matplotlib`.

Il rajoute des styles et des fonctionnalités (surtout utile en statistiques).



Concurrents 2D

Matplotlib n'est pas la seule bibliothèque de graphiques pour Python :

- (seaborn)
- plotly (figures web interactives)
- mpld3 (transforme une figure mpl en Javascript)
- bokeh (figures web interactive)
- plotly (figures web interactive)
- ggplot (port de la bibliothèque `ggplot2` de R)

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Matplotlib
Personnalisation
Seaborn
Concurrents 2D
Graphiques 3D

Concurrent : *plotly*

- interactive
- web charts

```
import plotly.express as px

xs = [i for i in range(100)]
fig = px.scatter(x=xs, y=[i ** 2 for i in xs])
fig.show()
```


Concurrent : *plotly*

```
import plotly.graph_objects as go

fig = go.Figure(
    data=go.Scatter(
        x=[1, 2, 3, 4],
        y=[10, 11, 12, 13],
        mode="markers",
        marker=dict(
            size=[40, 60, 80, 100],
            color=[0, 1, 2, 3]),
    )
)

fig.show()
```

Concurrent : *plotly*

```python
import matplotlib.pyplot as plt
import plotly
from plotly.tools import mpl_to_plotly

fig, ax = plt.subplots()
ax.plot([1, 2, 3], [1, 4, 9], "o")

plotly_fig = mpl_to_plotly(fig)
```


Dashboards

On peut utiliser `Dash` [84]

- utilise plotly
- basé sur le framework web `flask`
- permet de créer des dashboards web interactifs sans faire de HTML / JS
- bindings en R et Python
- exemples : `https://dash-gallery.plotly.host/Portal/`

84. `https://plot.ly/dash/`

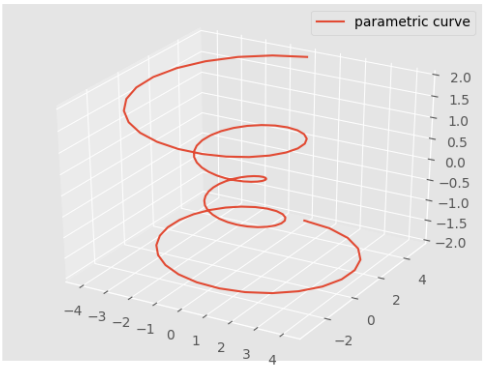
Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Matplotlib
Personnalisation
Seaborn
Concurrents 2D
Graphiques 3D

Graphiques 3D

```python
# source
# https://matplotlib.org/examples/mplot3d/lines3d_demo.html

import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt

mpl.rcParams['legend.fontsize'] = 10

fig = plt.figure()
ax = fig.gca(projection='3d')
theta = np.linspace(-4 * np.pi, 4 * np.pi, 100)
z = np.linspace(-2, 2, 100)
r = z**2 + 1
x = r * np.sin(theta)
y = r * np.cos(theta)
ax.plot(x, y, z, label='parametric curve')
ax.legend()

plt.savefig("test_3d.png")
plt.show()
```


Graphiques 3D



Résultat graphique 3D

Ce n'est pas de la "vraie" 3D... (pas de notion de volumes)

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Matplotlib
Personnalisation
Seaborn
Concurrents 2D
Graphiques 3D

Graphiques 3D

Pour faire de la vraie 3d :

- mayavi
- https://lorensen.github.io/VTKExamples/site/Python/
- (ParaView)
- moteurs de jeu 3D

Avant-propos

N'optimisez que ce qui est nécessaire

- faîtes des tests de performances ("profiling")
- n'optimisez que ce qui est nécessaire
- ne commencez que quand tout fonctionne et est testé
- évitez les copies et les mauvaises structures mémoires
- utilisez de bons algorithmes
- préférez les méthodes de `Scipy` souvent plus rapide que celles de `Numpy`
- zen of `Numpy`
- ...

Numexpr [85]

Les calculs Numpy se font en générant des tableaux intermédiaires. Numexpr permet de les supprimer en effectuant les calculs directement

```python
import numpy as np
import numexpr as ne

a = np.arange(1e6)
b = np.arange(1e6)

c = ne.evaluate("a + 1")
#  %timeit c = ne.evaluate("a + 1")
#  866 µs ± 74.6 µs per loop
#  (mean ± std. dev. of 7 runs, 1000 loops each)

#  %timeit c = a + 1
#  845 µs ± 37.2 µs per loop
#  (mean ± std. dev. of 7 runs, 1000 loops each)


d = ne.evaluate("sin(a) + arcsinh(a/b)")
#  %timeit np.sin(a) + np.arcsinh(a/b)
#  The slowest run took 6.65 times longer than the fastest.
#  This could mean that an intermediate result is being cached.
#  154 ms ± 139 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

#  %timeit  ne.evaluate("sin(a) + arcsinh(a/b)")
#  66.2 ms ± 2.11 ms per loop
#  (mean ± std. dev. of 7 runs, 10 loops each)
```

85. https://github.com/pydata/numexpr

Dask [86]

Framework de parallélisme.
Pour les *medium data* (ne tient plus en RAM mais sur un SSD)

S'intègre avec (en réutilisant les mêmes API) :

- numpy
- pandas
- scikit learn

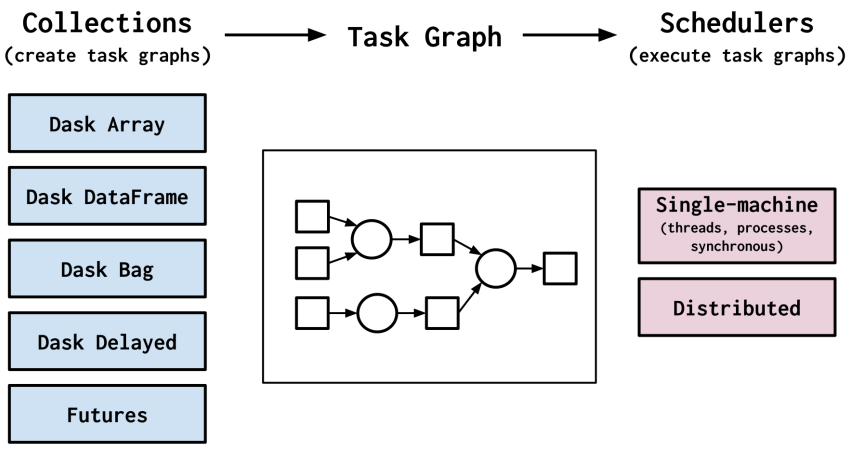
2 concepts :

- `scheduler` : exécute des graphes de calculs (comme make, luigi, celery...)
- `big data collections` : partitionnement des données ne tenant pas en RAM

86. https://dask.org/

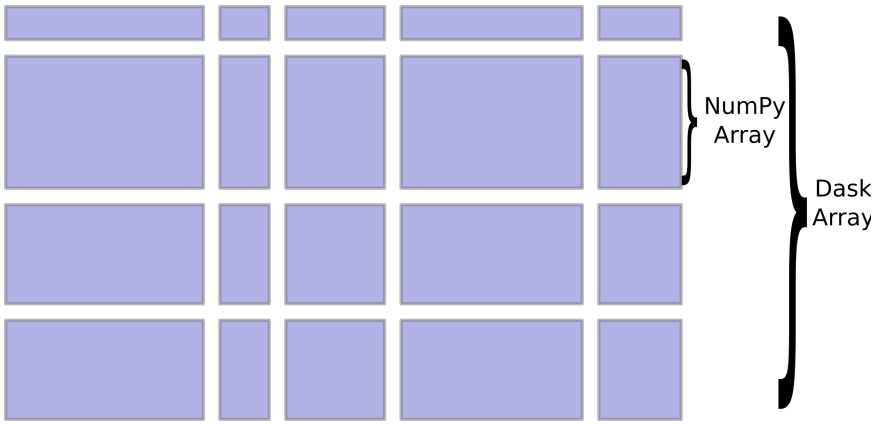
Dask

Dask : vue d'ensemble
Source: https://docs.dask.org/en/latest/



Dask

Dask : structure d'un *dask array*
Source: https://docs.dask.org/en/latest/array.html



Dask

```python
from dask.distributed import Client, progress

client = Client(
    n_workers=2,
    threads_per_worker=2,
    memory_limit="1GB"
)  # workers configuration
# we can change for process workers
# to deal with GIL perf issues

# go to : http://127.0.0.1:8787
```


Dask

```python
# source : https://examples.dask.org/dataframe.html
import dask
import dask.dataframe as dd


# lazy operation, they are only performed when we need the result
df = dask.datasets.timeseries()
df.head()

df2 = df[df.y > 0]
df3 = df2.groupby("name").x.std()

computed_df = df3.compute()
type(computed_df)

df[["x", "y"]].resample("24h").mean().compute().plot()
df[["x", "y"]].rolling(window="24h").mean().head()

# display the call graph
df[["x", "y"]].resample("24h").mean().visualize()

# store in RAM for faster computation
df = df.persist()
```

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Performances
Dask
Autres
Bibliographie

Autres techniques

D'autres techniques, plus ou moins matures permettent d'améliorer les temps de calculs également :

- Numba
- Pythran
- (Theano)
- distribution python par Intel

Bibliographie I

- graphiques
  - http://www.labri.fr/perso/nrougier/teaching/matplotlib/matplotlib.html#id8
  - https://python-graph-gallery.com/matplotlib/
  - https://matplotlib.org/gallery.html
  - http://pbpython.com/effective-matplotlib.html
  - https://matplotlib.org/faq/usage_faq.html
  - http://futurile.net/2016/02/27/matplotlib-beautiful-plots-with-style/#id16
- numpy
  - http://www.scipy-lectures.org/numpy/numpy.html
  - http://www.scipy-lectures.org/advanced/advanced_numpy/#block-of-memory
  - https://docs.scipy.org/doc/numpy/reference/internals.html

Bibliographie II

- https://jakevdp.github.io/PythonDataScienceHandbook/02.01-understanding-data-types.html
- http://www.labri.fr/perso/nrougier/teaching/numpy/numpy.html
- http://www.labri.fr/perso/nrougier/teaching/numpy.100/index.html

- scipy
  - https://scipy-cookbook.readthedocs.io/
  - https://docs.scipy.org/doc/scipy/reference/tutorial/index.html
  - https://docs.scipy.org/doc/scipy/reference/index.html
  - https://makina-corpus.com/blog/metier/2017/presentation-de-lecosysteme-python-scientifique

Bibliographie III

- pandas
  - la feuille de triche pandas officielle : https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf
  - inline vs copy operations : https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#indexing-view-versus-copy
  - les explications sur les différentes méthodes d'indexation : https://stackoverflow.com/questions/28757389/pandas-loc-vs-iloc-vs-at-vs-iat
  - https://pandas.pydata.org/pandas-docs/stable/dsintro.html
  - https://pandas.pydata.org/pandas-docs/stable/visualization.html
  - http://falce.net/presentation/python_pandas_monaco_parking
  - https://pandas.pydata.org/pandas-docs/stable/user_guide/cookbook.html#cookbook-resample

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Performances
Dask
Bibliographie

Bibliographie IV

- dask
  - tutoriel sur comment charger de grandes quantités de données : https://blog.dask.org/2019/06/20/load-image-data
  - notebooks d'exemples / tutos en lignes : https://hub-binder.mybinder.ovh/user/dask-dask-examples-irbwzcml/lab
  - cas d'usages réels : https://stories.dask.org/en/latest/
  - spark vs dask vs base de données : https://docs.dask.org/en/latest/spark.html
  - mise en place + vidéo : https://docs.dask.org/en/latest/setup.html
- manipulation d'images
  - documentation d'openCV : https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

Bibliographie V

- les exemples sur scipy-lectures : https://scipy-lectures.org/packages/scikit-image/index.html et http://scipy-lectures.org/advanced/image_processing/

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Création de modules
Programmation concurrente
Python scientifique
Écosystème scientifique
Jupyter
Numpy
Scipy
Pandas
Sympy
Analyse de réseaux
Machine learning / statistiques
Graphiques
Performances
Dask
Bibliographie