PYTHONによるデータサイエンス

Pythonの入門と橋渡し

用いるリンク先

- https://www-bcf.usc.edu/~gareth/ISL/. (http://bit.ly/ISLRusc)
- https://www.r-bloggers.com/in-depth-introduction-to-machine-learning-in-15-hours-of-expert-videos/ (http://bit.ly/ISLRrbloggers)
- https://github.com/JWarmenhoven/ISLR-python (http://bit.ly/ISLRpython)

今回は

- Introduction to Statistical Learning with Applications in Rは良い本です
- 最近、Rによる統計的学習入門と題した訳書が朝倉書店から出版されました
- この本はすべてがRのコードで用意されていますが、好事家が大半をPythonで実行できるようにコードを用意してくれています
  （JWarmenhoven,jcrouser,mscaudill,sujitpal）
  - 前回は、柔軟な学習機械ほど、複雑な問題ほど、大きなデータを必要とすること、当てはまりの良さよりは汎化能力が大事なこと、教師あり学習と教師なし学習について述べました。今回はClassificationとResampling Methodsを見ていこうと思います。

今回は

- Pythonのおさらいを少ししてから、またPythonのコードを眺めていこうと思います

PYTHONの復習

- Pythonはプログラミング言語と処理系の双方の名称
- Pythonのプログラムコードを実行するにはvi(m)、emacs、atomなど、自分の好みのテキストエディタで次の一行を入力し、hello.pyというテキストファイルに保存する

```
print “Hello, World!”
```

- そしてpythonというコマンドに、そのファイルを引数で与えてあげる

```
> python hello.py
```

(“>”はコマンドプロンプト)

- 拡張子.pyを持つファイルとpythonという名前のアプリケーションを連携させる機能をもつOSなら、ダブルクリックするだけでいい
- ループや関数の中身など、階層構造を字下げ（タブ）で表す

PYTHONの使い方

- 準備
  - Anaconda、本家のPythonインストーラ、ActiveStateのPythonインストーラ、Linuxディストリビューションのパッケージ、HomeBrew、MacPortsなどから、どれか一つをインストールします
  - pip、pyenv、virtualenvが便利と聞いた人は、それらの指示に従ってください
- 実行
  - pythonを実行して、コードをコピペする（最初だけ）
  - プログラムコードを記したソースファイルを作成し、pythonの実行時に読み込ませる（本番はこれが便利）
  - Jupyter[Hub, Notebook]を通してipythonを使う（学習時はこれが便利）

PYTHONはデータ型が少ない

- Pythonは圧倒的に標準のデータ型も少ない
- 論理型はbool
- 数値型はint(整数), float(浮動小数点), complex(複素数)の3種類
- 文字列型はstr (文字型はない)
- 配列はlist(リスト、[1,2])、tuple((1,2))、range、set(集合)、dict(辞書)
- 他にbitearray、bitsなどもある
- たとえば行列が標準ではないなど、ミニマルな設計方針のよう

PYTHONは予約語も少ない

- Pythonは圧倒的に予約語が少ないです

Pythonの予約語一覧

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

- これらは自分の変数や関数に使えません
- 言語自身について覚えることが少ないので、使用目的に早めに専念できます

何かにつけてIMPORT

- Python本体は極めてミニマル
- モジュールは拡張子.pyを持つファイルで、Python言語で記述されたオブジェクトやメソッドの定義が入っています
- モジュールは、importすると利用可能になります

```
import numpy as np;
```

- これでNumPyのモジュールnumpyを、npという名前でimportします
- パッケージは、__init__.pyというファイルを持つ複数のモジュールを収めたフォルダです

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

- これで、Matplotlibというパッケージの中のPyPlotというモジュールをimportします

FROMも使う

- Matplotlibというパッケージの中のPyPlotというモジュールをimportするには、次の２つのやり方があります

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

```
from matplotlib import pyplot as plt;
```

- どちらでもいいです

IMPORTの仕方

- モジュールのimportの仕方は少なくとも次の４種類がある

```
import numpy;
numpy.array([1,2,3,4]);
```

```
from numpy import *;
array([1,2,3,4]);
```

```
import numpy as np;
np.array([1,2,3,4]);
```

```
from numpy import array;
array([1,2,3,4]);
```

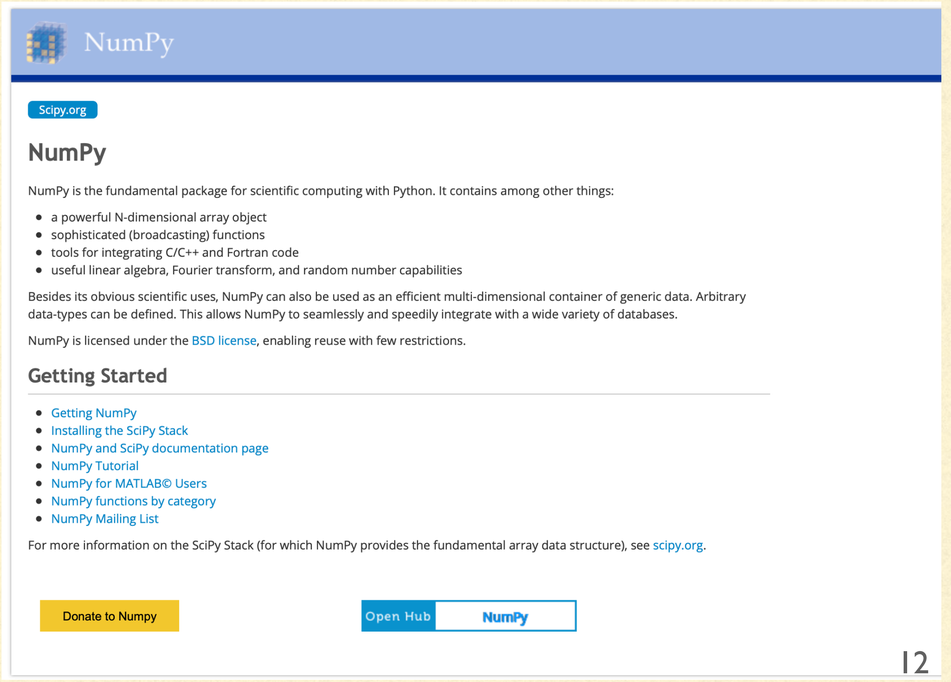
- どれを用いても良い
- importした関数や変数などの名前が「衝突」することがある
- 衝突を回避するには、importした関数や変数にモジュール名かモジュール名のイニシャルを付けて使う方法が望ましい
- 小規模のコードであれば、気にしなくていい

NUMPY

- NumPyは数値計算のためのモジュール群
- 多次元配列ndarrayとその操作、また配列同士や配列を用いる数学関数を提供する
- NumPyが提供する配列arrayが重要

```
import numpy as np;
A = np.array([[1,2,3],[4,5,6]]);
B = np.array([[1,2],[3,4],[5,6]]);
print A;
print A.dot(B);
```

- 行列matrixもあり、行列の和・差・積も可能
- 円周率pi、Napier数e、Eulerのγeuler_gammaもここで定義される

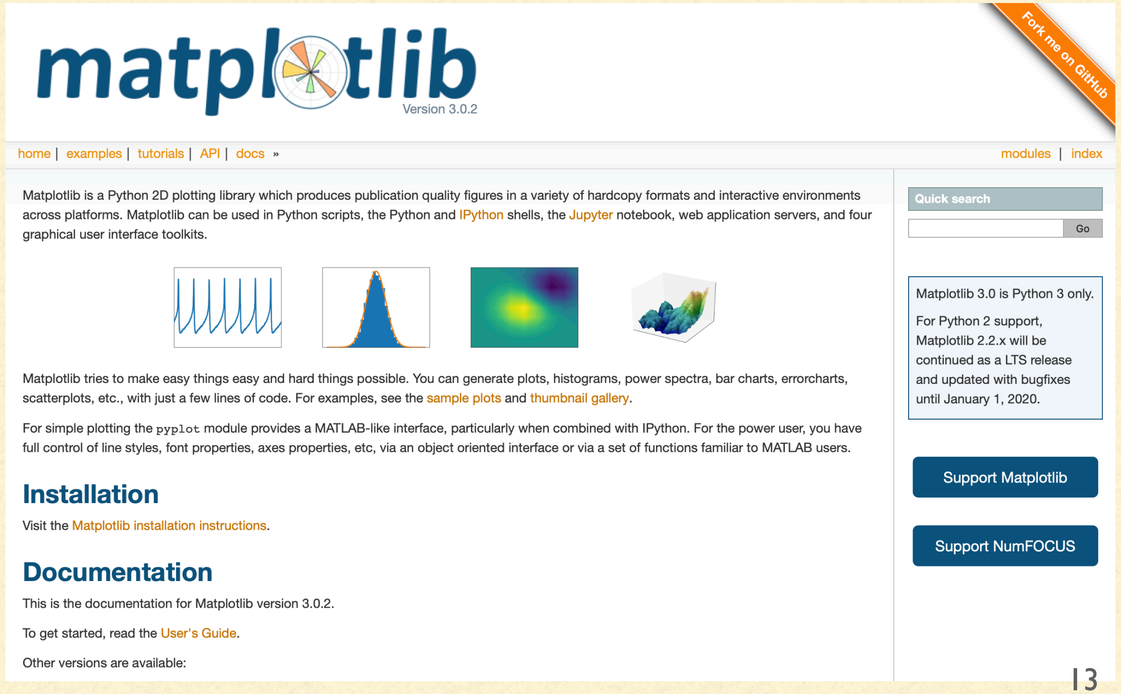


MATPLOTLIB

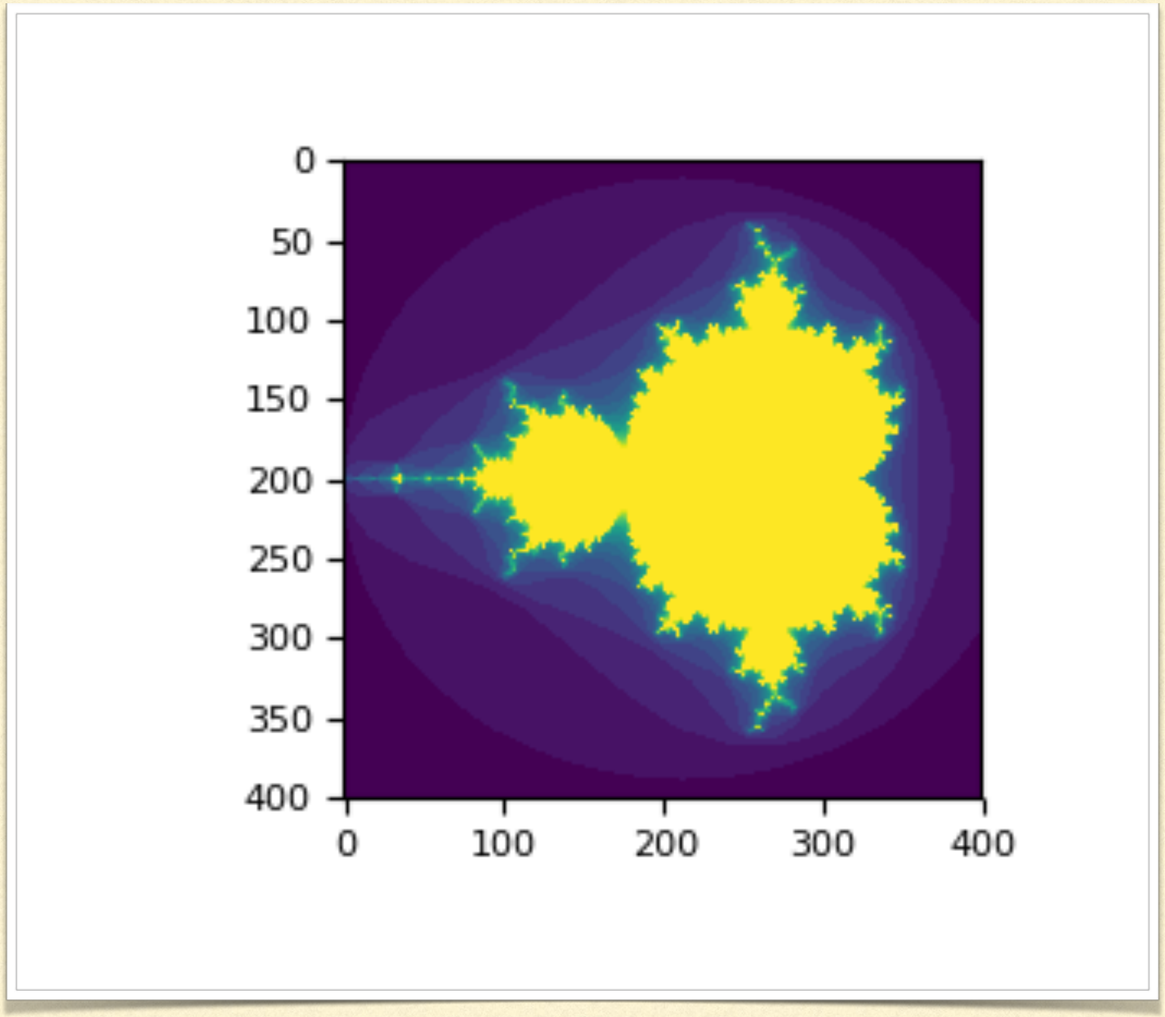
- Matplotlibはデータの可視化モジュール
- 多くの２次元グラフを提供する（ギャラリーを各自で見ていただくと良い）

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

```
from matplotlib import pyplot as plt;
```



NUMPYとMATPLOTLIBの例



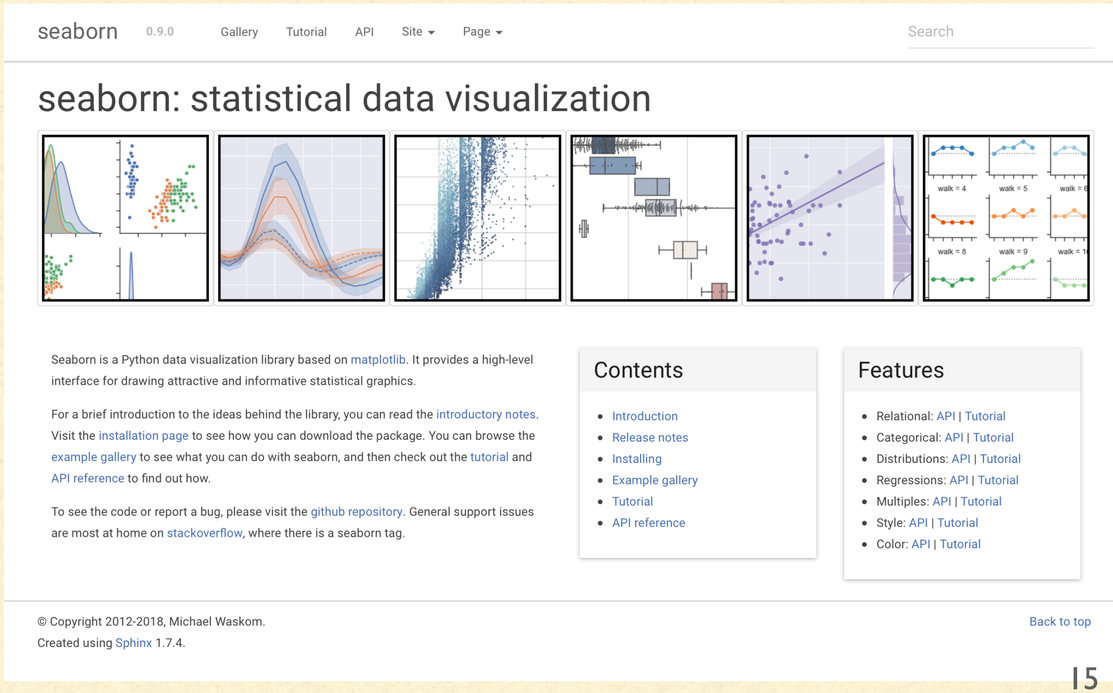

```
import numpy as np;
import matplotlib.pyplot as plt;
def mandelbrot( h,w, maxit=20 ):
    """Returns an image of the Mandelbrot fractal of size (h,w)."""
    y,x = np.ogrid[ -1.4:1.4:h*1j, -2:0.8:w*1j ]
    c = x+y*1j
    z = c
    divtime = maxit + np.zeros(z.shape, dtype=int)

    for i in range(maxit):
        z = z**2 + c
        diverge = z*np.conj(z) > 2**2            # who is diverging
        div_now = diverge & (divtime==maxit)  # who is diverging now
        divtime[div_now] = i                  # note when
        z[diverge] = 2                        # avoid diverging too much

    return divtime
plt.imshow(mandelbrot(400,400));
plt.show();
```

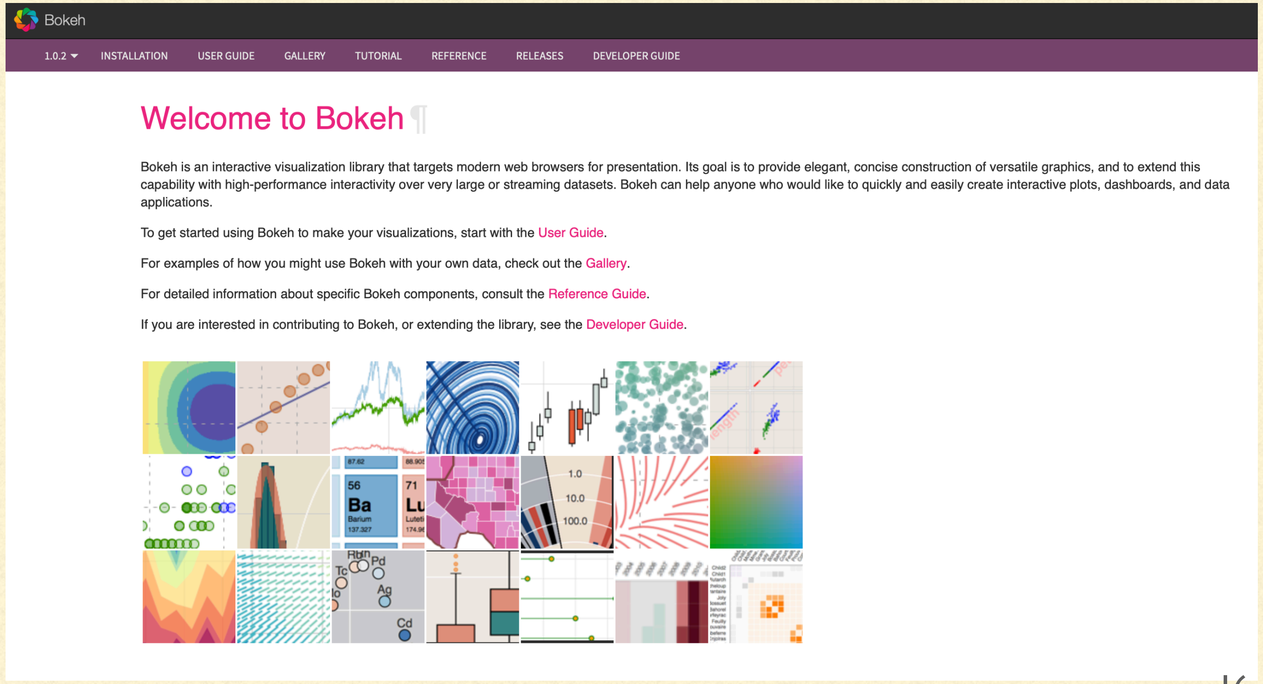
SEABORN

- Seabornも統計的なデータの可視化モジュールで、Matplotlibをベースに開発されています
- これもギャラリーを見ていただくと良いです



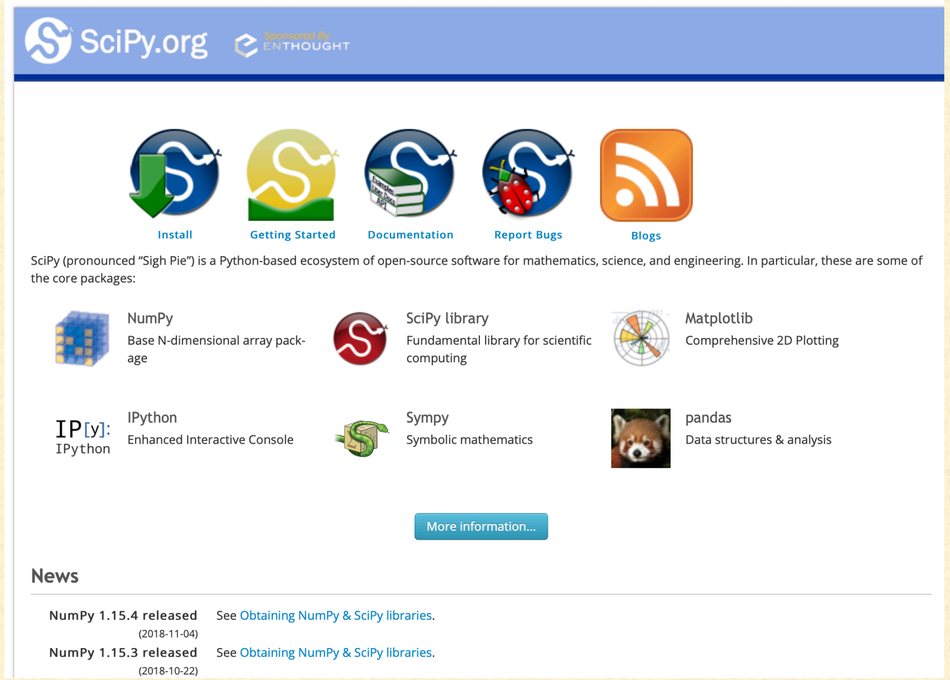
BOKEH

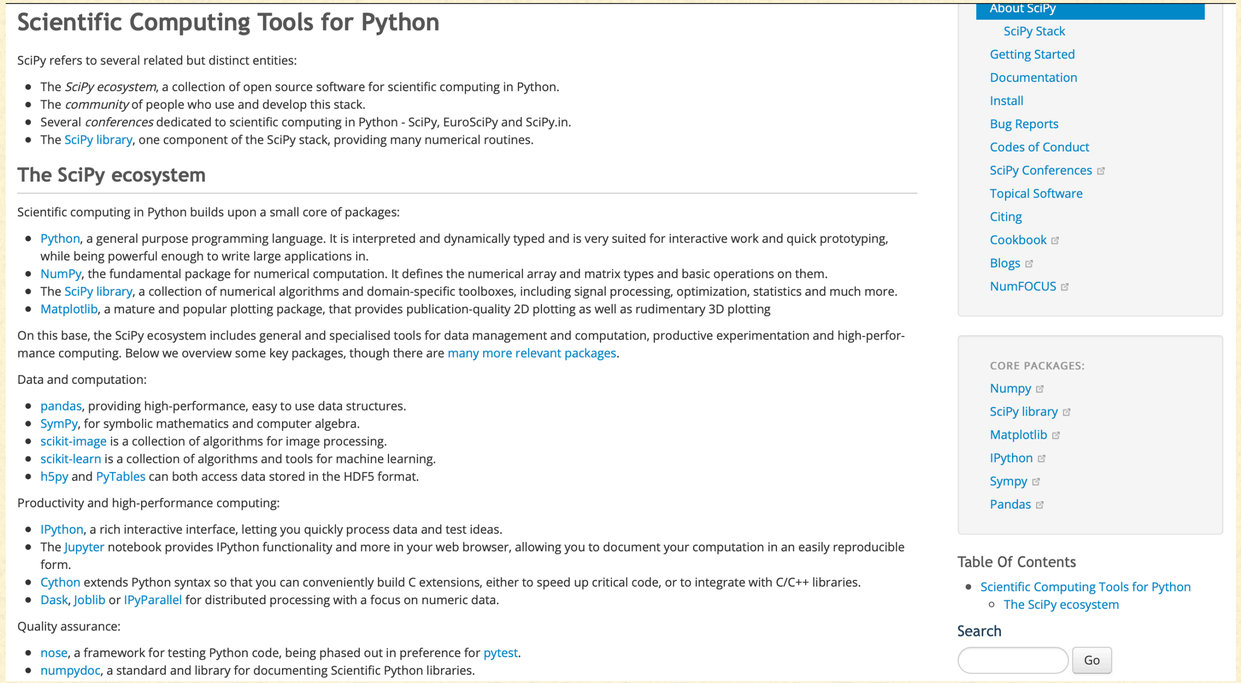
- Bokehは、ブラウザを用いたインタラクティブな可視化を実現するモジュールです
- Bokehサーバを経由します
- こちらのギャラリーは素晴らしいの一言です



SCIPY

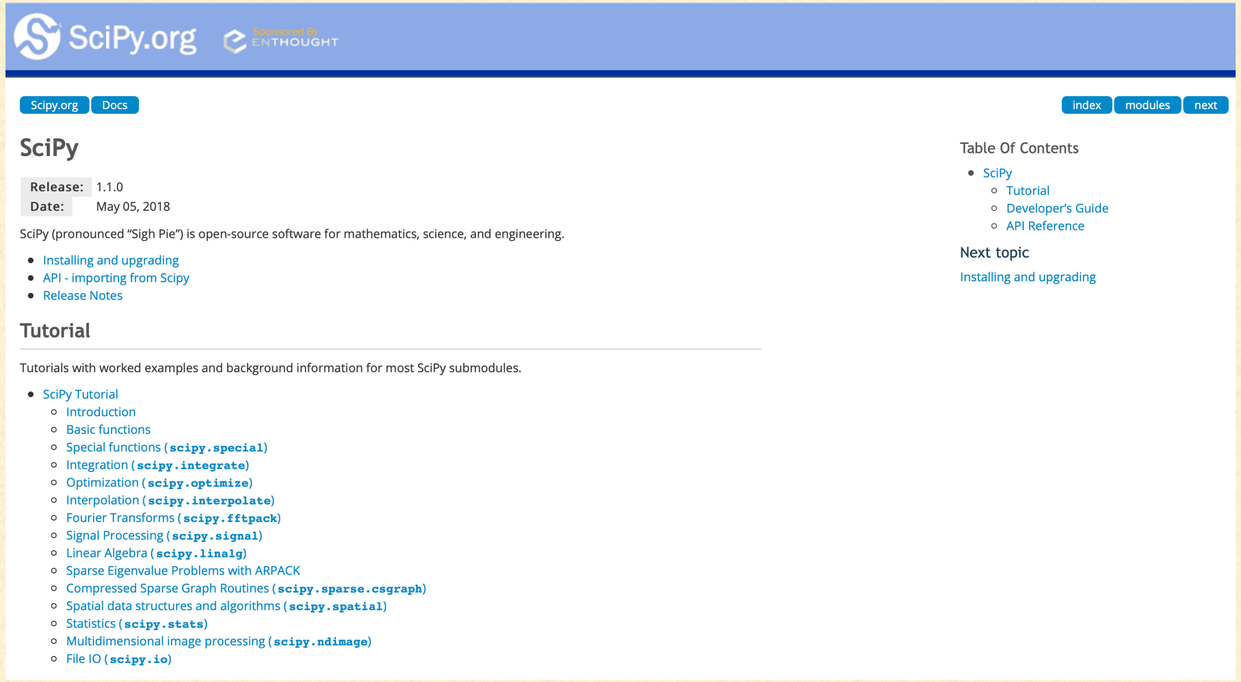
- SciPyはPythonによる数学、科学、工学のためのオープンソースソフトウェアのエコシステム（要するに科学技術計算のためのパッケージ群）でNumPyも含まれる
- Python, NumPy, SciPyライブラリ, Matplotlibの上に構築されている



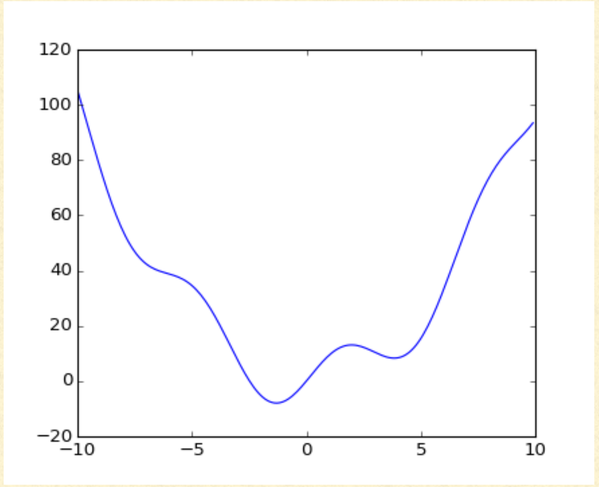


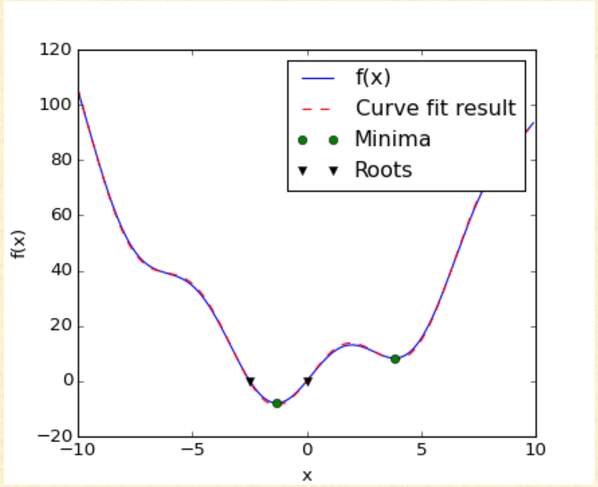
SCIPY

- ライブラリもSciPy （SciPy libraryと呼び分けている）
- クラスタリングcluster、定数constants、離散フーリエ変換fftpack、
  積分と常微分方程式integrate、補間interpolate、入出力io、線形代数linalg、
  その他misc、多次元画像処理ndimage、
  直交距離回帰odr、最適化optimize、
  信号処理signal、疎行列sparse、
  疎行列の固有値問題、
  疎グラフsparse.csgraph、
  計算幾何spatial、特殊関数special、
  確率・統計stats、
  C言語とC++言語のインタフェースweave



SCIPYの例






```
import numpy as np
from scipy import optimize
def f(x):
    return x**2 + 10*np.sin(x)
x = np.arange(-10, 10, 0.1)
plt.plot(x, f(x))
plt.show()
optimize.fmin_bfgs(f, 0)
```

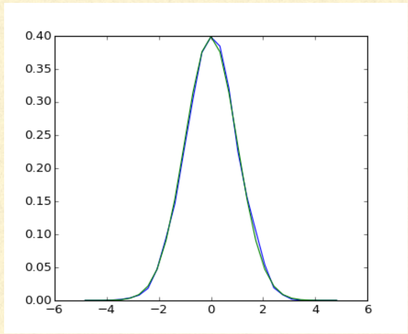
```
Optimization terminated successfully.
         Current function value: -7.945823
         Iterations: 5
         Function evaluations: 24
         Gradient evaluations: 8
array([-1.30644003])
```

```
optimize.fmin_bfgs(f, 3, disp=0)
array([ 3.83746663])
optimize.basinhopping(f, 0)
```

```
                    nfev: 1725
 minimization_failures: 0
                     fun: -7.9458233756152845
                       x: array([-1.30644001])
                 message: ['requested number of basinhopping iterations completed successfully']
                    njev: 575
                     nit: 100
```

```
xmin_local = optimize.fminbound(f, 0, 10)
xmin_local
```

```
3.8374671...
```




```
a = np.random.normal(size=1000)
bins = np.arange(-4, 5)
bins
```

```
array([-4, -3, -2, -1,  0,  1,  2,  3,  4])
```

```
histogram = np.histogram(a, bins=bins, normed=True)[0]
bins = 0.5*(bins[1:] + bins[:-1])
bins
```

```
array([-3.5, -2.5, -1.5, -0.5,  0.5,  1.5,  2.5,  3.5])
```

```
from scipy import stats
b = stats.norm.pdf(bins)  # norm is a distribution

plt.plot(bins, histogram)
```

```
[<matplotlib.lines.Line2D object at ...>]
```

```
plt.plot(bins, b)
```

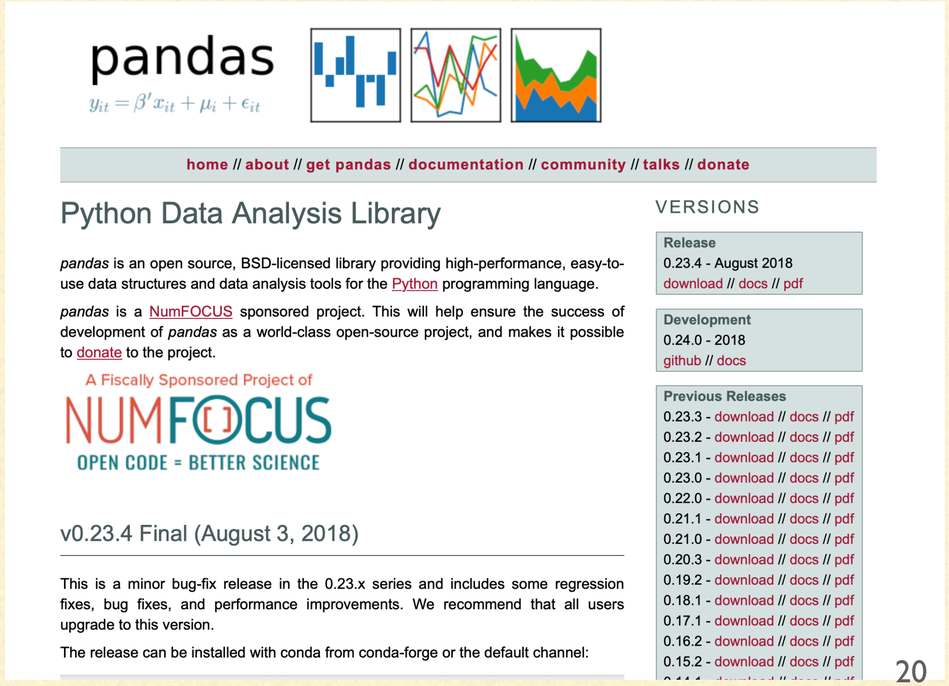
```
[<matplotlib.lines.Line2D object at ...>]
```

PANDAS

- pandasはNumPyをベースに開発されているモジュールです
- Rと似ているデータフレームDataFrameというオブジェクトと、時系列解析のためのシリーズSeriesというオブジェクトを提供します
- Excel形式やCSV形式の入出力が可能です

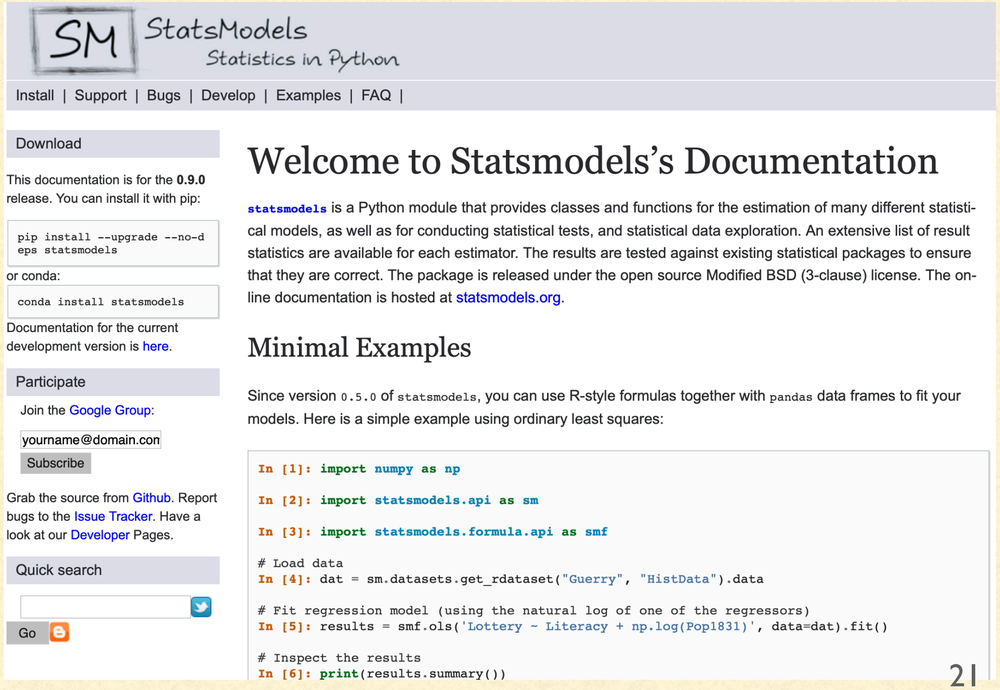
```
import pandas as pd;
```

- pandasはpdという名前でimportされる例を多く見かけます



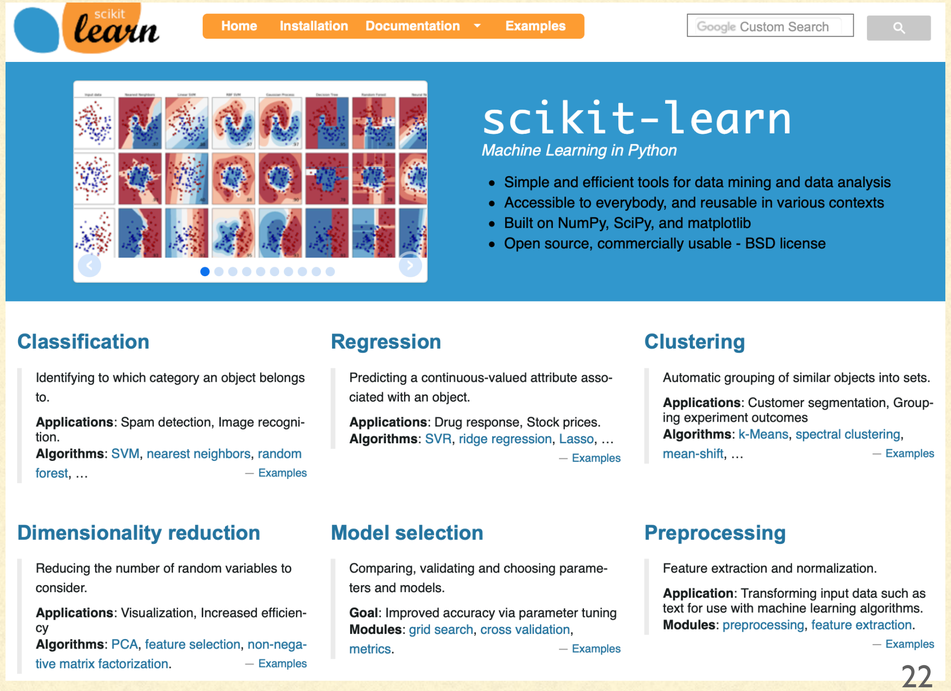
STATSMODELS

- statsmodelsは統計モデルを多く含むモジュールです
- 開発グループのウェブサイトは、トップページに使い方の例題を載せています



SCIKIT-LEARN

- scikit-learnは、教師あり学習と教師なし学習のための多くの機械学習の手法を含む、巨大なモジュール群です
  - 分類問題 Classification
  - 回帰問題 Regression
  - クラスタリング Clustering
  - 次元圧縮 Dimension reduction
  - モデル選択 Model selection
  - 前処理 Preprocessing
- これもチュートリアルが充実しています



細かいこと

- 個々のモジュールやメソッドの仕様や使い方については、それぞれのドキュメントを自ら参照することをお勧めします
- どんな言語でもそうですが、同じ名称の関数が、いつのまにか仕様変更となっていることがあるためです