PythonとStapy5年のあゆみ

辻 真吾（@tsjshg）

みんなのPython勉強会＃57（3度目のオンライン開催）

5/14, 2020

https://startpython.connpass.com/event/163366/

お前、誰よ？

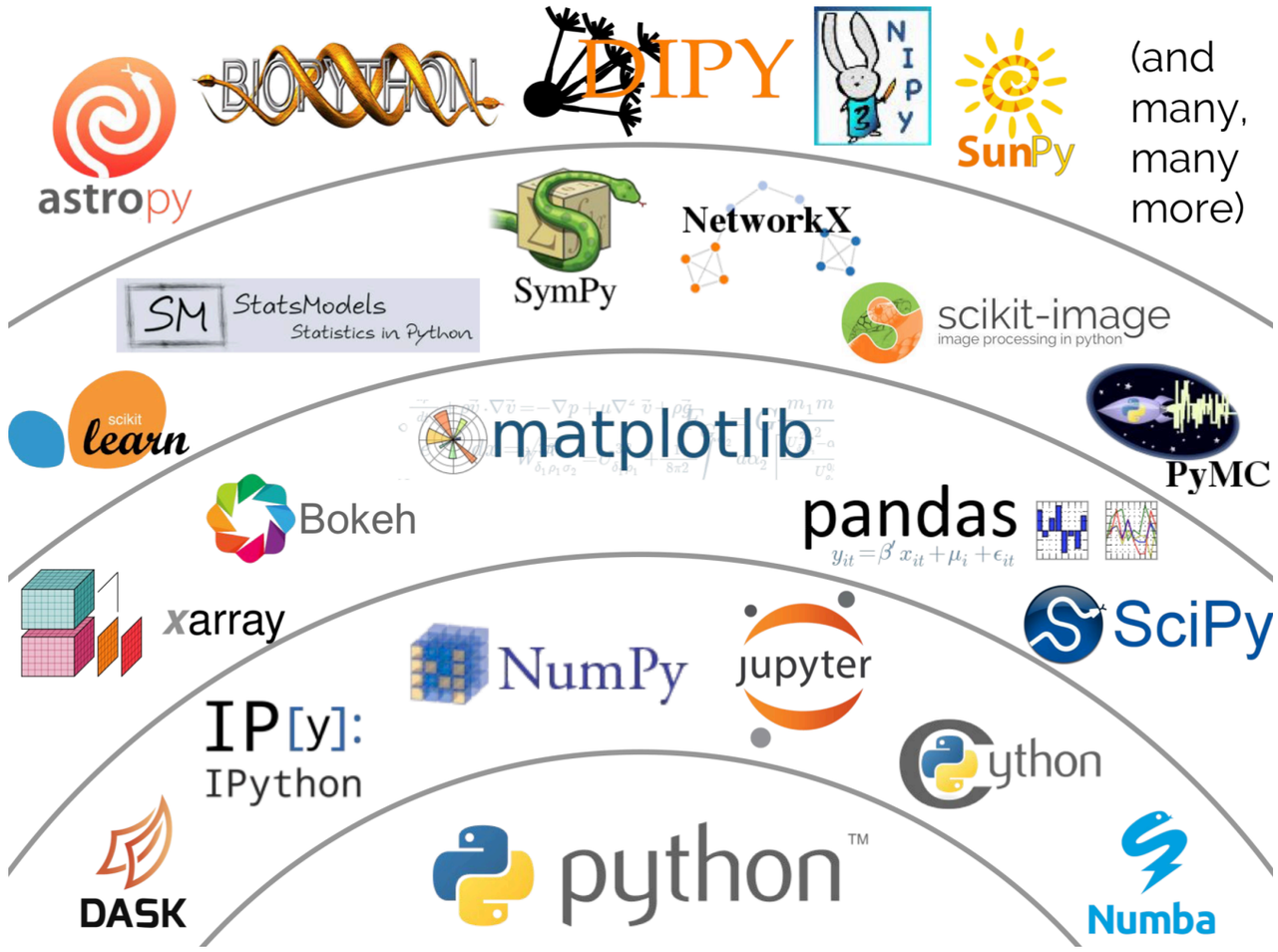
- 東京大学先端科学技術研究センターに勤務
  - 生命科学とコンピュータサイエンスの両方にそこそこ詳しいです
- 詳しくは、www.tsjshg.info をご覧ください

- 今日は、この5年のPython界隈の話題と、stapyを絡めた話をしようと思っていましたが・・・
- ただPythonデータサイエンスに関連するライブラリの話になってしまいました。ごめんなさい。

https://www.ianlewis.org/jp/pycon-mini-jp

Python
データサイエンス

- Pythonは汎用プログラミング言語
  - python.org
- データサイエンスを可能にするのはライブラリ群
  - pipで個別にインストール
- Anacondaディストリビューション
  - condaコマンドも便利

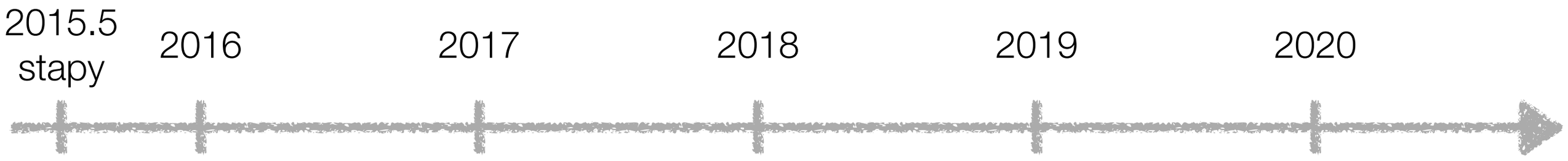


Jake vanderPlas, "The Unexpected Effectiveness of Python in Science", PyCon 2017

ANACONDA®

https://anaconda.org/

Python, NumPy, pandasバージョンの変遷（5年）



2010.7
Python 2.7

2020.1
End of Python 2

NumPy	2015.10 1.10.0	2016.3 1.11.0		2017.1 1.12.0	2017.6 1.13.0	2018.1 1.14.0	2018.7 1.15.0	2019.1 1.16.0	2019.7 1.17.0	2019.12 1.18.0
pandas	2015.10 0.17.0	2016.3 0.18.0	2016.10 0.19.0	2017.5 0.20.0	2017.10 0.21.0	2017.12 0.22.0	2018.5 0.23.0	2019.1 0.24.0	2019.7 0.25.0	2020.1 1.0.0

- Python本体の2から3への完全移行
- 外部ライブラリの活発な開発状況

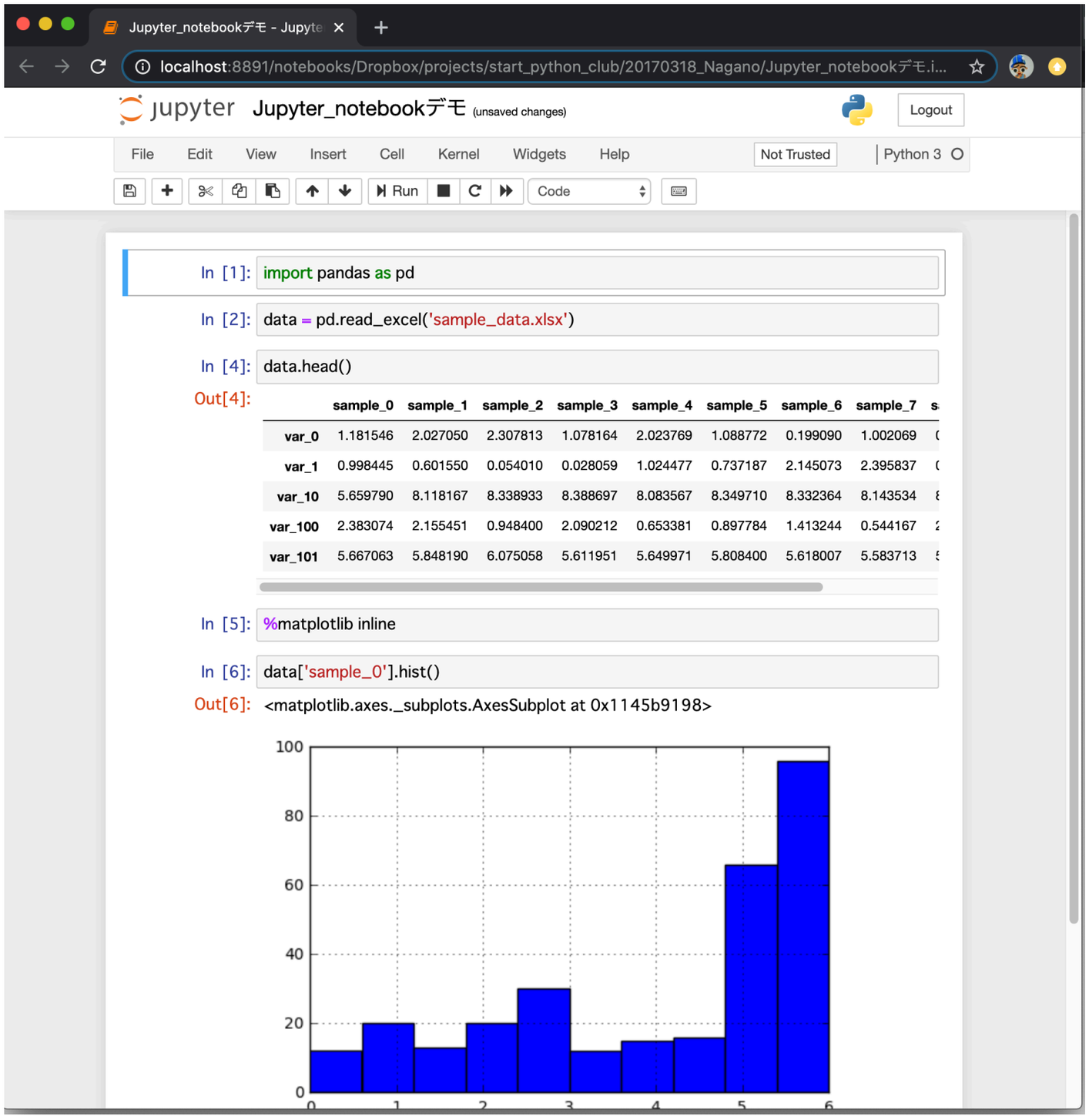
Pythonとライブラリの進化でみるこれまでとこれから

- Jupyter NotebookからJupyter Labへ
- NumPyを支えるテクノロジー
- pandasとZen of Python
- PyCaretの登場

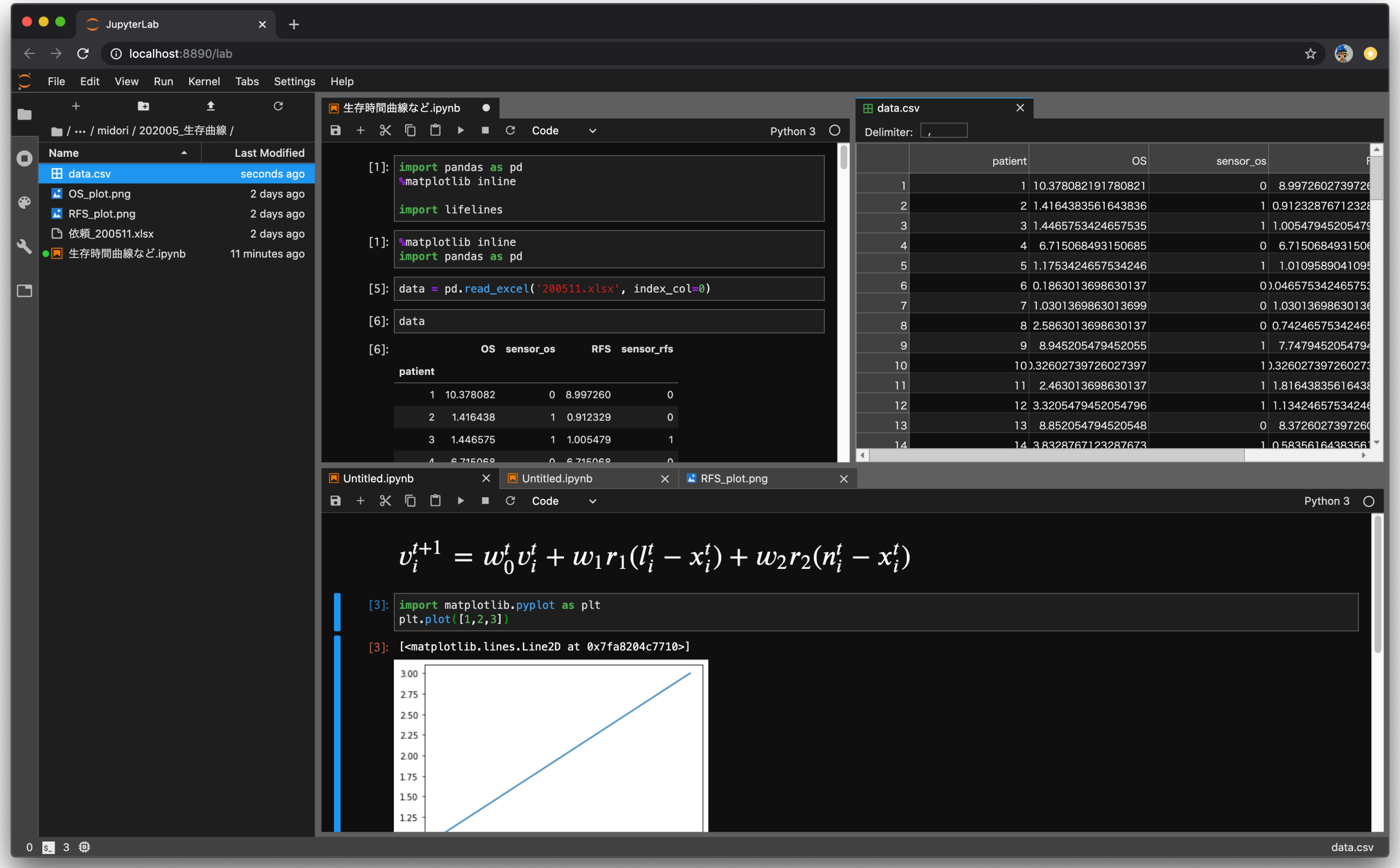
Jupyter NotebookからJupyter Labへ

Jupyter Notebook

- データサイエンスに限らず、Pythonコードの実行環境として人気
- Webブラウザ内でコードの記述、実行、結果の保存が可能
  - 拡張子.ipynb
- Markdownを使ったドキュメントの記述
- カーネルを追加することで、さまざまな言語を利用可能



Jupyter Notebookのサンプル画面



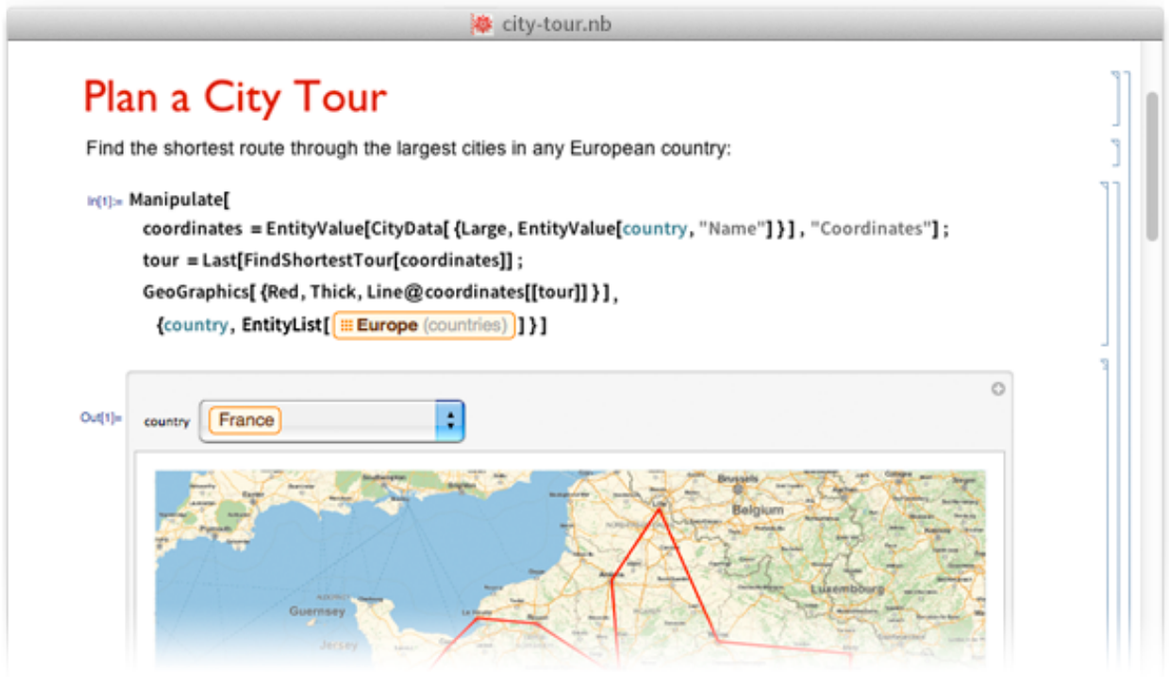
Jupyter Labのサンプル画面

ノートブックの分割表示、CSVや画像のプレビュー、セルのノートブック間でのコピーなどが可能

ノートブックの元祖

ノートブックドキュメント

Wolframノートブックは，テキスト，グラフィックス，インターフェース等とコードを組み合せることができ，デスクトップでもWebでも使えます：



PYTHONのプログラマー向けの注意事項

Wolframノートブックは25年以上に渡り常に開発され続けており，Wolfram言語のデスクトップ版とクラウド版のどちらにおいても完全に統合されています．Wolframノートブックはjupyterのようなノートブックライブラリのきっかけとなっています．

- おもに数式処理ソフトMathematicaで使われるWolfram言語はノートブックの先駆け
- このWolframエンジンは現在無償公開されていて、Jupyterに組み込むことも可能（らしい）

（拡大）

Wolframノートブックはjupyterのようなノートブックライブラリのきっかけとなっています。

http://www.wolfram.com/language/fast-introduction-for-programmers/ja/notebook-documents/

NumPyを支えるテクノロジー

NumPyとは？

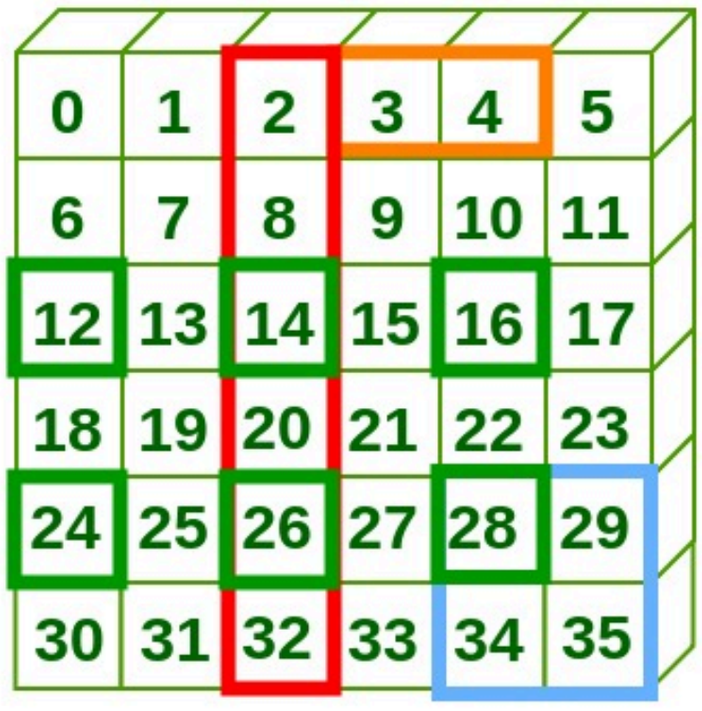
- Pythonで数値計算をするときの必須ライブラリ
- ベクトルや行列の計算が効率良くできるnumpy.ndarray型
  - Pythonのリストと違い、一度定義するとサイズを変更できず、格納できるデータ型も1種類
- 要素のアクセスや演算など便利な機能がたくさん

```
>>> a[0,3:5]
array( [3,4] )

>>> a[4:, 4:]
array( [ 28, 29],
       [ 34, 35] ] )

>>> a[ :, 2]
array( [2, 8, 14, 20, 26, 32] )

>>> a[2 : : 2, : : 2]
array( [ 12, 14, 16],
       [ 24, 26, 28] ] )
```



https://www.geeksforgeeks.org/numpy-indexing/

とにかく速い

この速度比較はほとんど意味ないですが・・・

In [1]:
```
import numpy as np
```

In [5]:
```
vec_a = np.random.randn(200_0000)
vec_b = np.random.randn(200_0000)
```

200万次元のベクトル2つ

In [6]:
```
%%timeit
vec_a @ vec_b
```

内積を計算

```
877 µs ± 12.8 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

In [ ]:
```
```

In [7]:
```
%%timeit
sum([a * b for a, b in zip(vec_a, vec_b)])
```

```
609 ms ± 12.1 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

CPUコアがいくつかあると並列化してくれる

```
In [2]: import numpy as np

In [3]: # 10000次元の説明変数を持つサンプルが4万個
        X = np.random.randn(4_0000, 10000)

In [*]: # Mは4万x4万の正方行列 i行j列はX[i]とX[j]の内積
        M = X @ X.T

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```

4万行×1万列の行列Xを用意

行列Mのi, j成分はX[i]とX[j]の内積になる

（単純には）16万回の内積計算

100%以上なので

並列化してくれている

Basic Linear Algebra Subprograms（BLAS）

- ベクトルや行列など線形代数の演算を実行するAPIのデファクトスタンダード（Reference BLAS）
  - 1970年代からの歴史がある*
- 最適化されたOpenBLASやIntelが無料で配布しているMath Kernel Library（MKL）などがある
- numpy.show_config()で確認できる
  - AnacondaディストリビューションはMKLを利用

* http://history.siam.org/pdfs2/Dongarra_%20returned_SIAM_copy.pdf

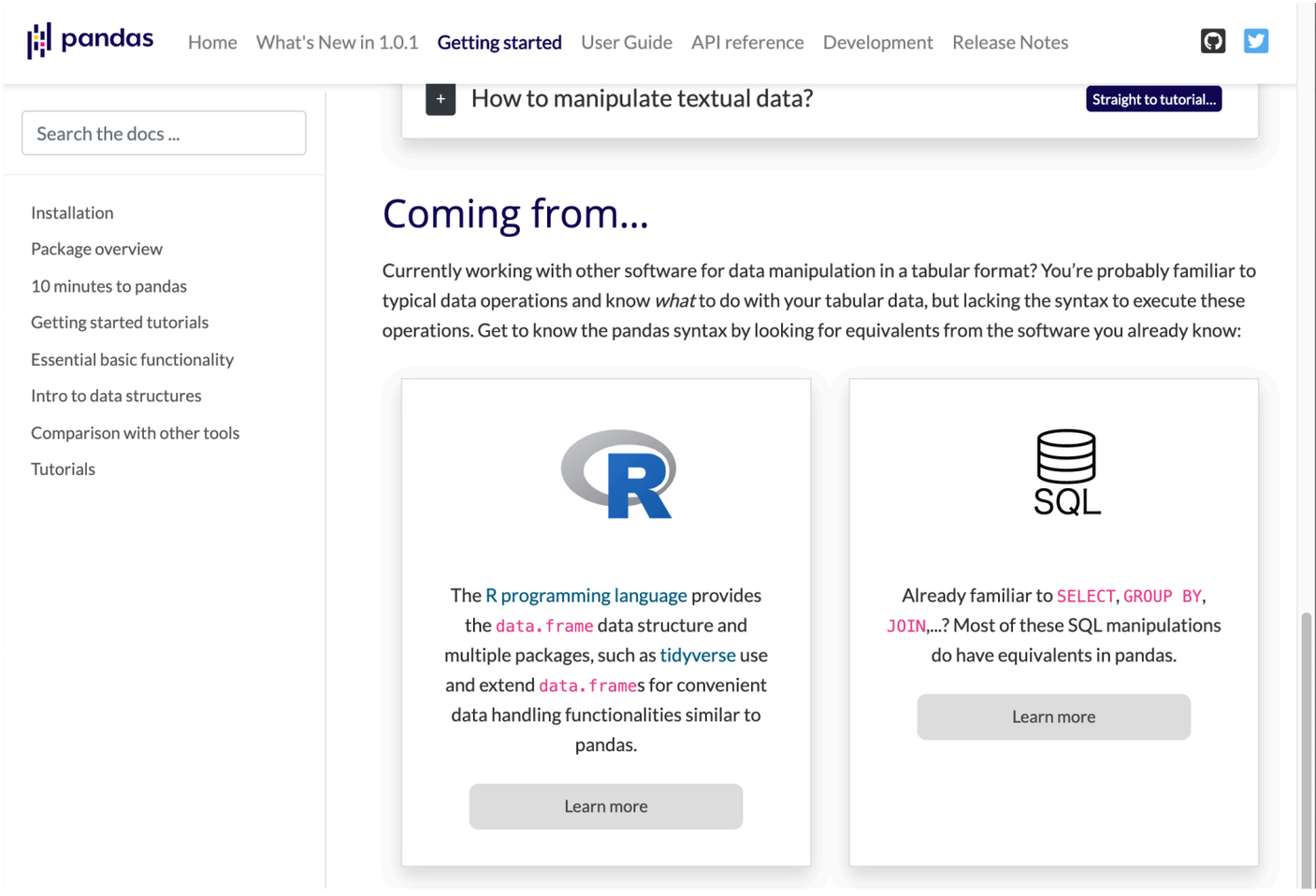
pandasとZen of Python

pandas

- 行と列に名前が付いた表形式の2次元データ（DataFrame型）と1次元のSeries型を提供する超高機能ライブラリ

- Pythonデータサイエンスに欠かせない存在

- 機能が多すぎるので、最低限これだけはという記事*もある



RのDataFrameやSQLのテーブルからの影響

* https://medium.com/dunder-data/minimally-sufficient-pandas-a8e67f2a2428

やり方がいろいろありすぎる

```
import pandas as pd

df = pd.DataFrame({'足': [2,4,6], 'genome(Gbp)':[3.2, 2.4, 0.5], 'size': [1.65, 1.7, 0.005]},
                  index=['ヒト', 'パンダ', 'アブラムシ'])
df
```

	足	genome(Gbp)	size
ヒト	2	3.2	1.650
パンダ	4	2.4	1.700
アブラムシ	6	0.5	0.005

```
df.足
```

```
ヒト      2
パンダ      4
アブラムシ    6
Name: 足, dtype: int64
```

```
df.size
```

```
9
```

列へのアクセスはこちらがおすすめ

```
df['size']
```

```
ヒト      1.650
パンダ      1.700
アブラムシ    0.005
Name: size, dtype: float64
```

Zen of Pythonに反する？

```
import this
```

The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

ixメソッドの廃止

- DataFrameの要素へのアクセス
  - 行や列の名前ならloc
  - 行や列の位置ならiloc
- これらを混在できるixメソッドが0.20まで使えた
- 賛否はあった*が廃止を決定
- “Explicit is better than implicit.”の哲学に沿った変更と言える（実際は別の理由かも）

```
df
```

	足	genome(Gbp)	size
ヒト	2	3.2	1.650
パンダ	4	2.4	1.700
アブラムシ	6	0.5	0.005

```
df.loc['パンダ', '足']
```
行・列名で指定

```
4
```

```
df.iloc[1, 0]
```
行・列の位置で指定

```
4
```

＊ https://github.com/pandas-dev/pandas/issues/14218

PyCaretの登場

caret 機械学習を便利にするRのライブラリ

- 機械学習モデルを作る時によくある作業を自動化
  - データの分割、クロスバリデーションやハイパーパラメータの調整など
- scikit-learnを使っても10行以上になるような作業を1〜数行で実行

The `caret` Package

Max Kuhn

2019-03-27

1 Introduction

The `caret` package (short for **C**lassification **A**nd **RE**gression **T**raining) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for:

- data splitting
- pre-processing
- feature selection
- model tuning using resampling
- variable importance estimation

as well as other functionality.

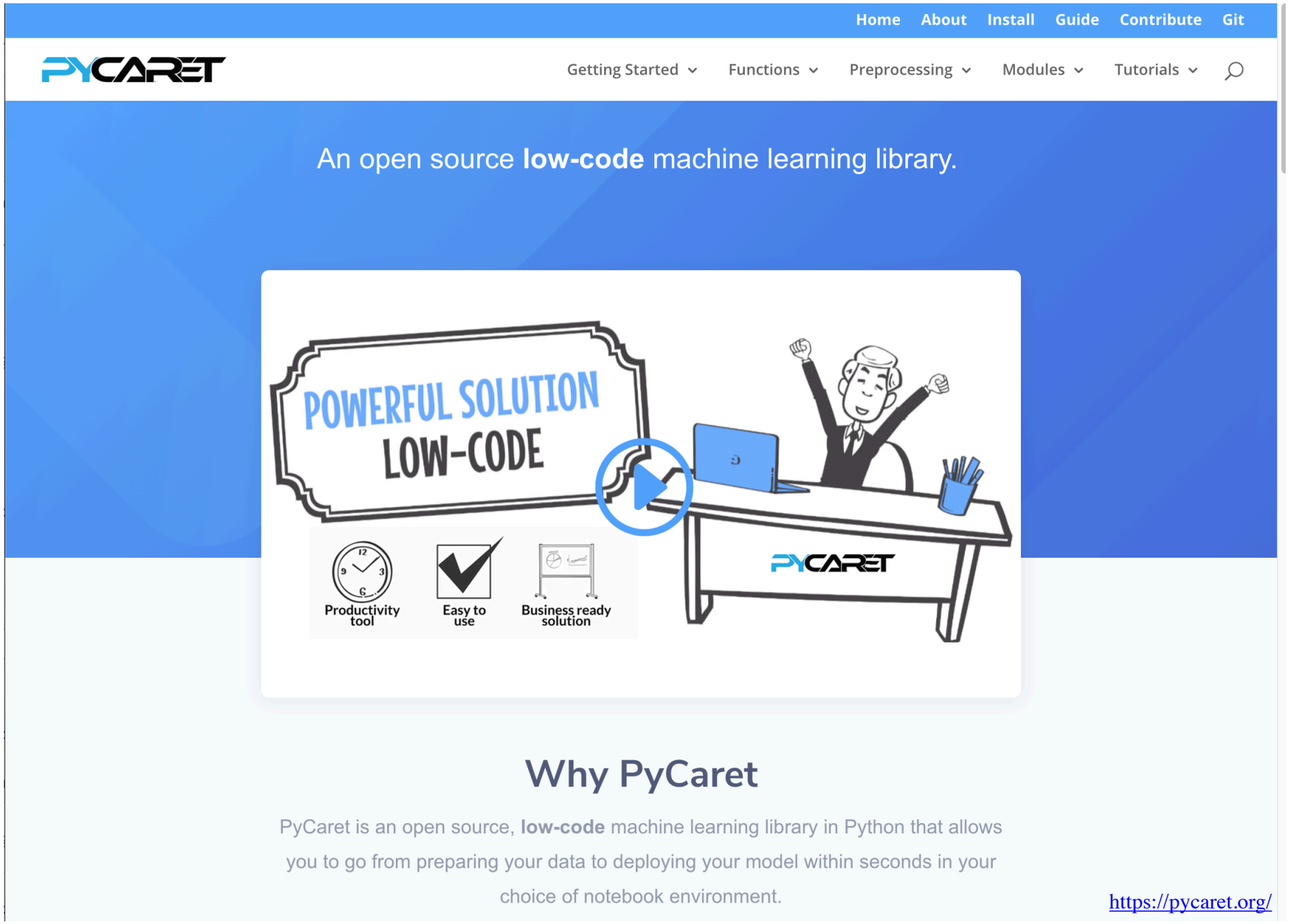
There are many different modeling functions in R. Some have different syntax for model training and/or prediction. The package started off as a way to provide a uniform interface the functions themselves, as well as a way to standardize common tasks (such parameter tuning and variable importance).

The current release version can be found on CRAN and the project is hosted on github.

Some resources:

https://topepo.github.io/caret/

でた！（2020年4月）



scikit-learnを含む既存ライブラリのラッパー

caretと名前は似ているが、作者Moez Ali氏の1人プロジェクト？

まとめ

- Pythonデータサイエンスは外部ライブラリとその進化に支えられている
  - たとえば、IPython→Jupyter→JupyterLab
- Pythonの哲学（Zen of Python）とライブラリの変化
- 少ないコードで大きな効果
  - NumpyをOpenBLASやMKLによる並列化
  - PyCaretに代表される機械学習自動化の流れ

当日時間切れで対応できなかった質問への回答

> **Anonymous**
> 7:54 PM
>
> 5
>
> pycaretに限らずpythonはscikit-learnなど機械学習用の便利なライブラリが多いです。正直良く中身がわかってなくても使えてしまいます。辻さんはどこまで外部ライブラリの中身・原理を理解しておくべきだと思いますか。

機械学習モデルでしたら、手法の概要くらいは理解するべきだと思っています。ロジスティクス回帰とランダムフォレストの違いがまったくわからないのに、ライブラリを使うのはすこし危険な気がします。一方で、実際の実装を細かく見るのは難しいので、そこまでは気にしなくてもよい気がします。自分で実装したり、だれかの実装を読むのはすごく勉強になるのでおすすめですが、時間がかかるので、連休中とか盆暮れ正月あたりの課題ですかね。

Anonymous

7:37 PM

3

stapyの5年間で数値計算ライブラリに起こった変化のうち、辻さんがイチオシのものはなんですか？

本編でもお話ししたIntel Math Kernel Library（MKL）がAnacondaにデフォルトで組み込まれたのが、（調べたら）Anaconda 2.5（2016年2月リリース）だったので、stapy5年の範囲に入っているのでこれを挙げたいと思います。MKLはすごい値段が高いライブラリだというイメージがあったので、このニュースを聞いた時は驚きました。マイクロソフトのVSCodeもそうですが、最近IT業界の老舗巨大企業がオープン化（MKLはオープンソースではないですが）へ舵を切っているように見えて、一般ユーザには嬉しい流れです。

Anonymous
7:55 PM

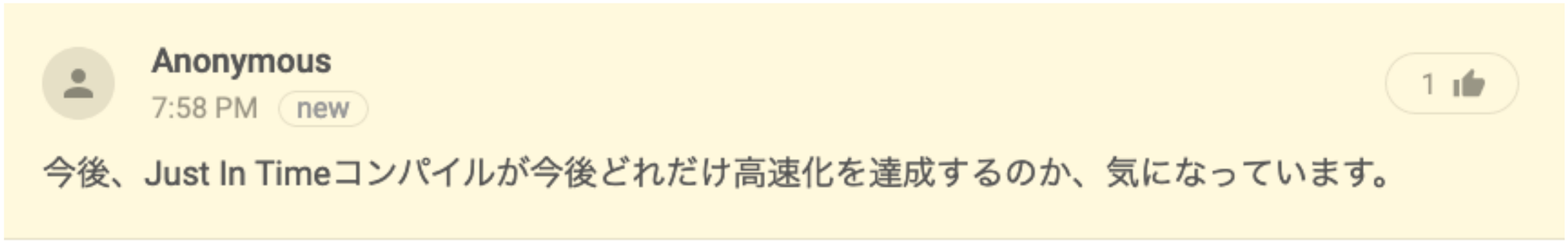
1

pandas だから

df.loc['パンダ', '足']

だったんですか・・？

はい。



PythonのJITと言えばNumbaが有名ですよね。昔はpyscoなんかもあって、手軽に速くなるので、すこし使っていた時期もありましたが、残念ながらpyscoは2012年にメンテされなくなってしまいまいた。PyPyはいまでも活発に開発されているようで、使ってみたい気もしますが、結局作るプロブラムが外部ライブラリに依存することが多く、それほど高速化の恩恵を受けられないこともあったりするので、難しいですよね。

http://psyco.sourceforge.net/

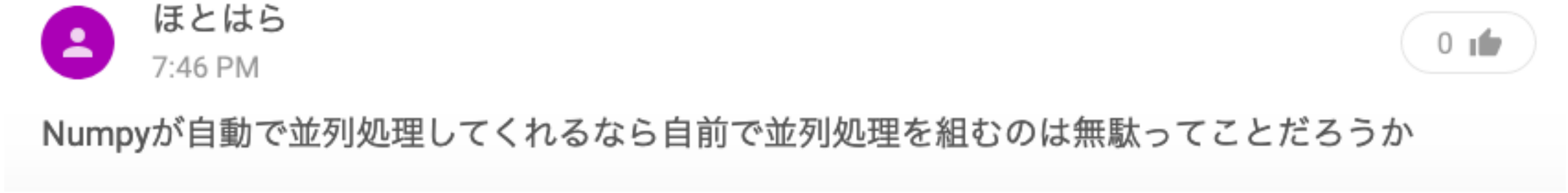
Anonymous
7:41 PM

0

np.random.randn(200_0000)　で200万個の乱数を計算するんですね？

大きな整数はアンダースコアで区切って表記できます。普通3桁ごとですが、200万だとこの方が分かり易いかなと思いまして。

また、np.random.randn(3, 4)とすると、3行4列の行列になって返ってくるので便利です。



これはそうではないと思います。ちょっとした計算をするときに並列化してくれるのは便利ですが、独立した手続きにできる処理を自分で並列化する方が全体として高速化できる可能性は十分あると思います。

ただ、こうしたチューンニングは作業としては一番最後にもってくるべきかと思います。まず動くものを作って、その後、自分が書いたコードの範囲で無駄なことをやっていないかチェックした後という感じです。（だいたい、このあたりで力尽きて、まあいっかとなりそうですが・・・）