



米国データサイエンティストがやさしく教えるデータサイエンスのためのPython講座

变更履歴

2022/11/1 初版

本講座で学べること

- ✓ Dockerを使った環境構築
- ✓ Pythonの基本
- ✓ データサイエンス関連のライブラリ(NumPy, Pandas, Matplotlib, Seaborn, OpenCV, etc...)
- ✓ コーディングの考え方

注意

- ✓ データサイエンス(統計学や機械学習)を学ぶ講座ではありません。
- ✓ scikit-learnやtensorflowなどは扱いません。
- ✓ 本講座はMac推奨ですが、Docker環境を作れればOSは問いません

レベル

プログラミングレベル

- Python経験者
- Python未経験者/プログラミング経験者
- プログラミング初心者
- △ プログラミング未経験者

データサイエンスレベル

- 全員

ブログ講座(無料)：https://datawokagaku.com/python_for_ds_summary/

本講座の特徴

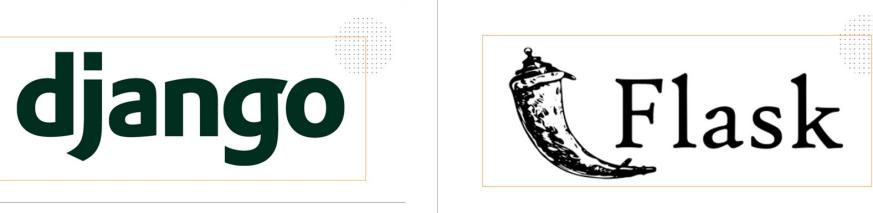
- ✓教科書的な内容ではなく、実践的な内容(現場のプロ目線)
- ✓実際にどこでどう使うのかを教える
- ✓考え方を教える
- ✓学習した内容を何度も使って覚える

心得

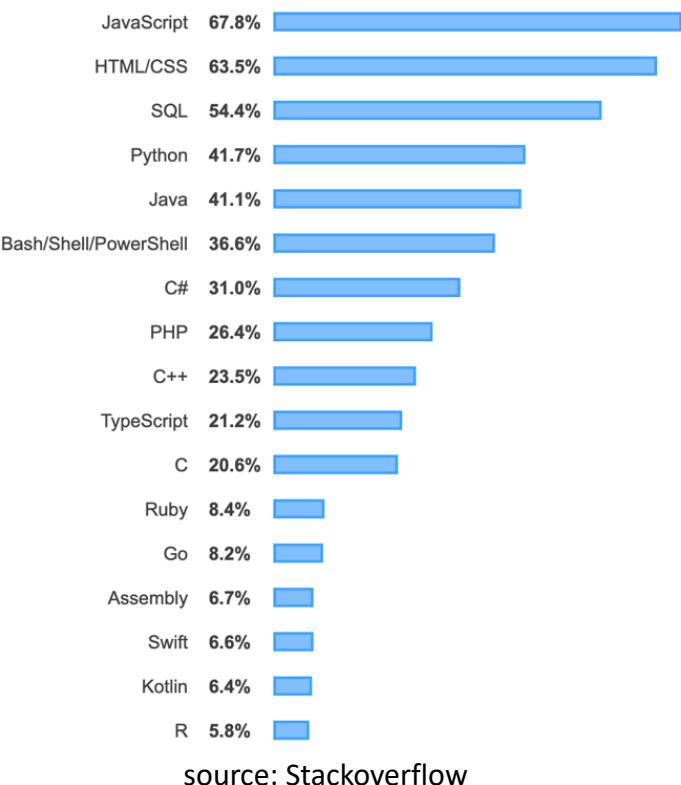
- ✓プログラミングは、短期間で習得できるスキルではない
- ✓一度に全てを理解しようとしない

(Rではなく)なぜPython?

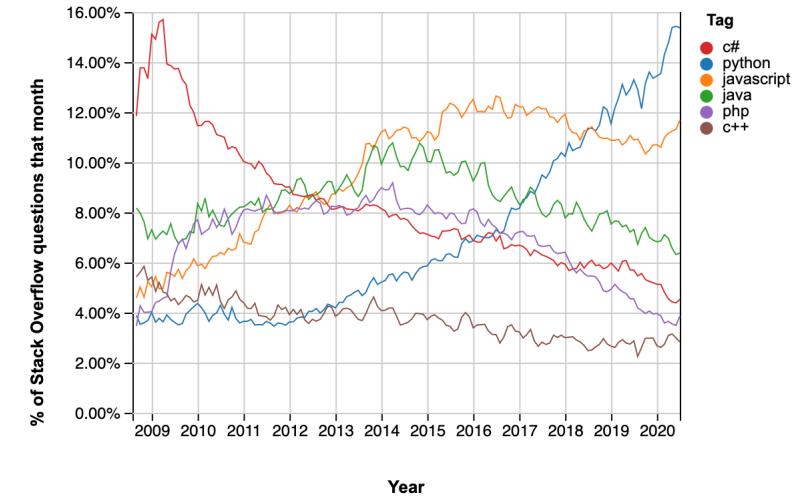
1. 高い汎用性



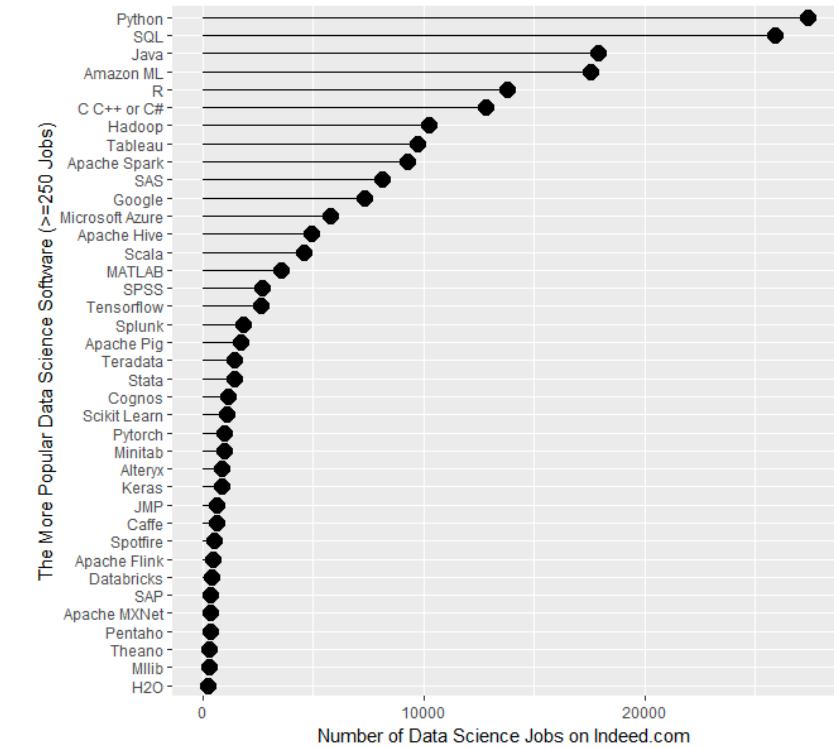
2. 他の言語との類似性



3. 求人が多い



4. ユーザが多い



5. 処理が速い

source: Stackoverflow

PEP8 <https://www.python.org/dev/peps/pep-0008/>

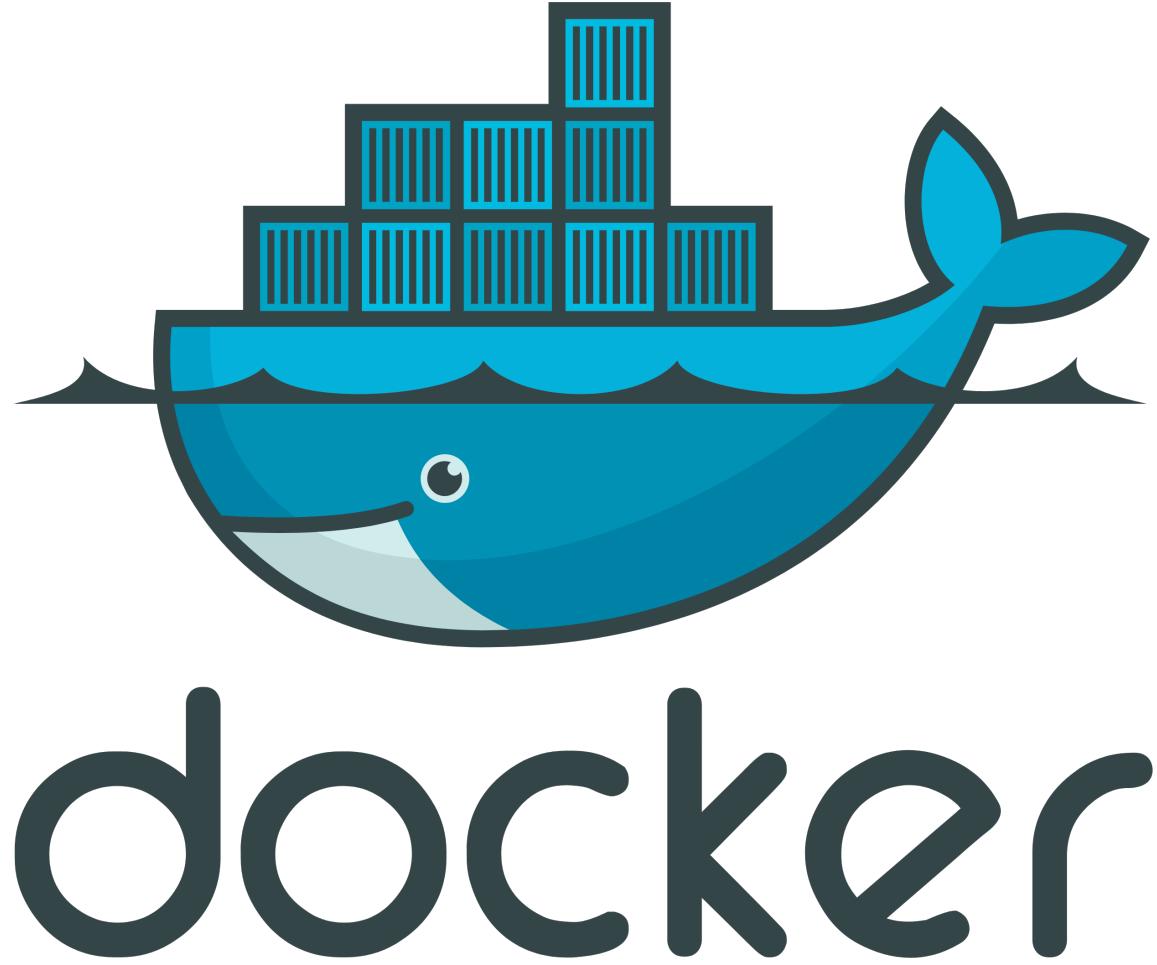
本講座はなるべく PEP8(Style Guide for Python Code)に準拠

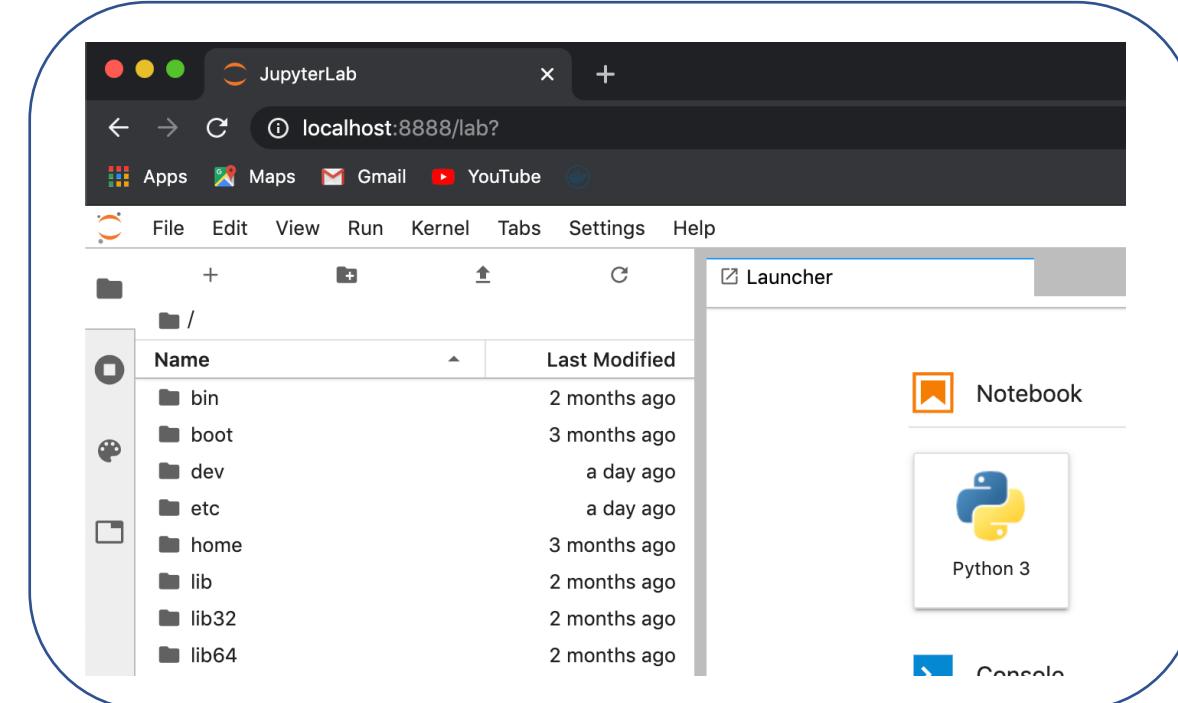
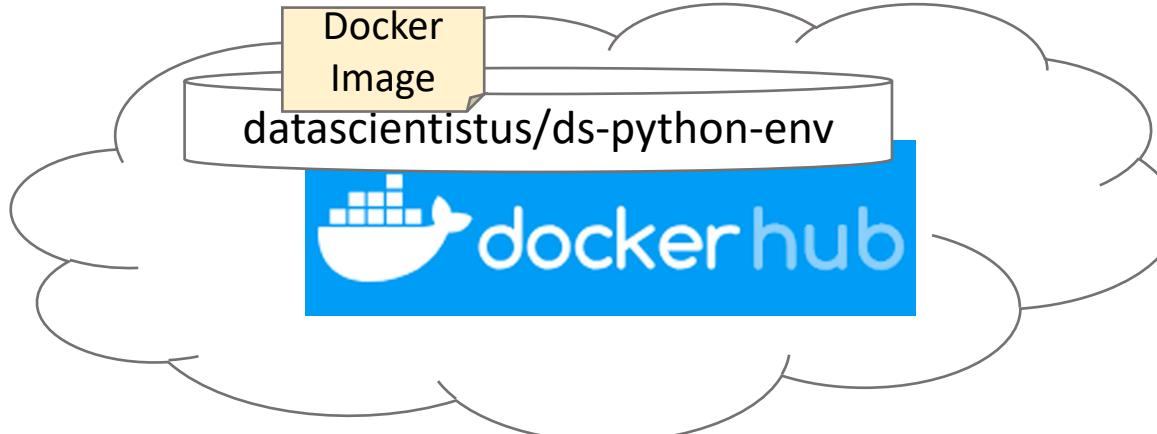
```
# Correct:  
dct['key'] = lst[index]
```

```
# Wrong:  
dct ['key'] = lst [index]
```

```
# Correct:  
spam(1)
```

```
# Wrong:  
spam (1)
```



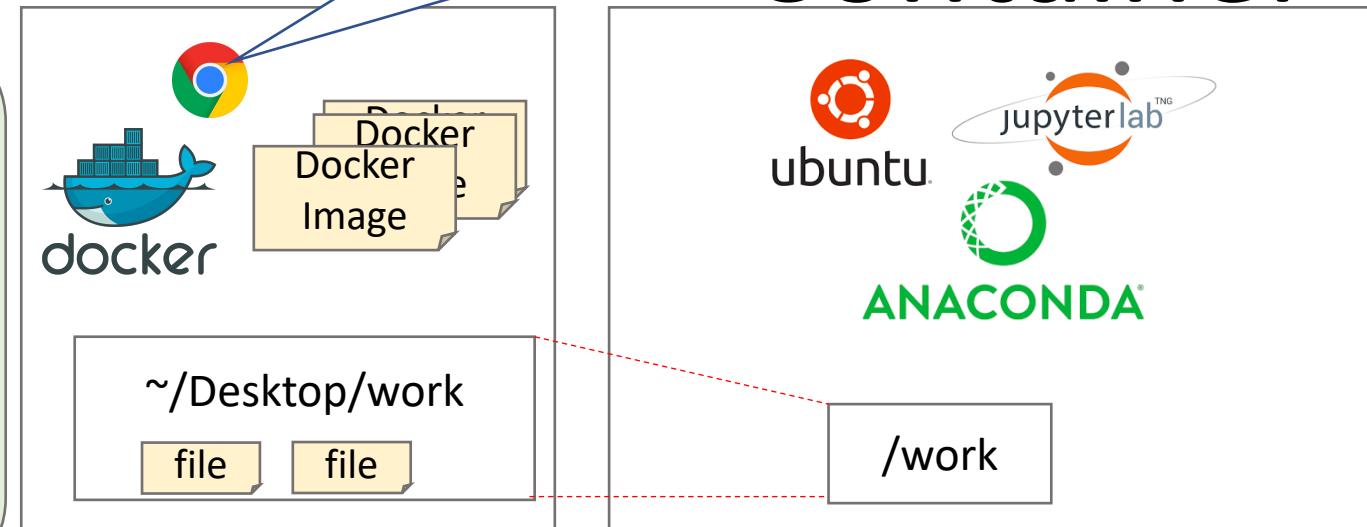


Terminalで実行

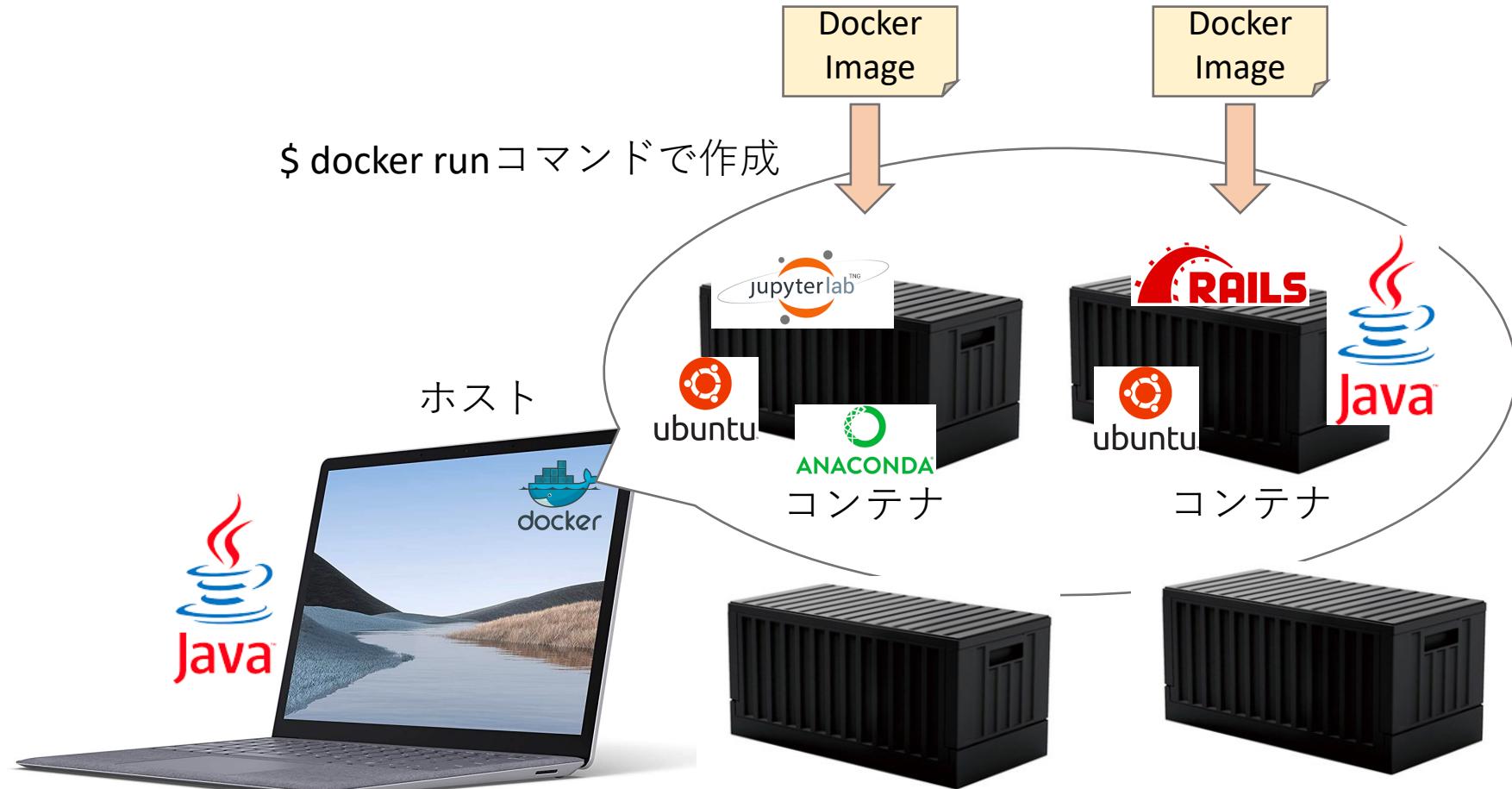
```
$ docker login
$ docker pull datascientistus/ds-python-env
$ mkdir ~/Desktop/work
$ docker run -v ~/Desktop/work:/work -p
8888:8888 --name my-env datascientistus/ds-
python-env
```

ブラウザでlocalhost:8888にアクセス

Host Container



Dockerの基本操作



Dockerの基本操作

コンテナー一覧表示

```
$ docker ps -a
```

Docker image一覧表示

```
$ docker images
```

Docker imageからコンテナを起動

```
$ docker run <image> (オプション付き : $ docker run -v ~/Desktop/work:/work -p 8888:8888 --name my-env <image>)
```

コンテナを止める

```
$ docker stop <container>
```

コンテナを削除

```
$ docker rm <container>
```

コンテナをrestart

```
$ docker restart <container>
```

さらにDockerを勉強した人は・・・

Docker超入門講座(テキスト)：

<https://datawokagaku.com/whatisdocker/>

ゼロから教えるDocker講座(動画)：

https://datawokagaku.com/docker_lecture/

[‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’, ...,]

(0, 0)	(0, 1)	(0, 2)	(0, 3)
(1, 0)	(1, 1)	(1, 2)	(1, 3)
(2, 0)	(2, 1)	(2, 2)	(2, 3)

Mutable vs Immutable

変更**可能**なオブジェクト

list

dict

set

変更**不可**なオブジェクト

integer

float

bool

string

tuple

ちょっとした計算をする時にLambda関数を使う

Date	Type	City	Revenue	Sales Tax Rate	Tax
10/1	credit	Tokyo	1000	8%	80
10/2	credit	Tokyo	3000	10%	300
10/3	cash	Osaka	2000	8%	160

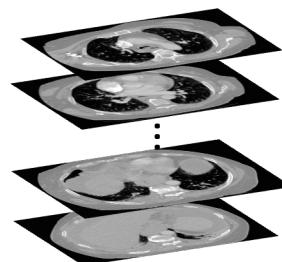
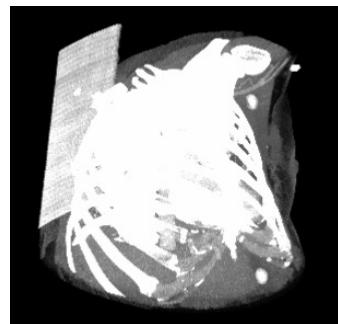
表計算についてはまた後ほど. . .

演習：関数

シナリオ(例)：ある患者のCT画像が順番に保存されている。

ファイル名にはスライスIDしか書いていない。

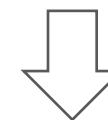
ファイル名にスライス番号(index)と拡張子(.png)をつけたリストを作りたい。



スライスID
スライスID
スライスID



スライスID_0.png
スライスID_1.png
スライスID_2.png

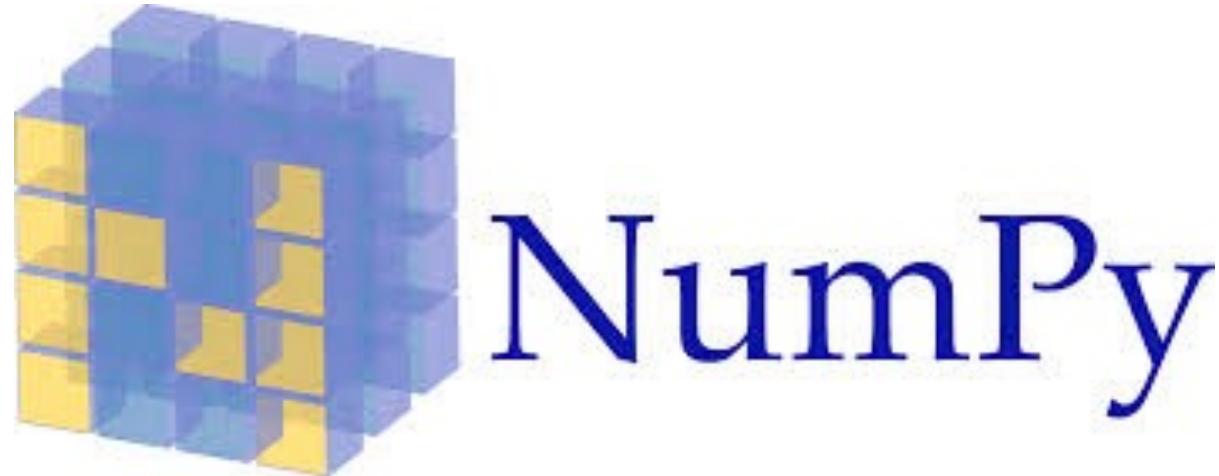


もう少し汎用化

課題：文字列のリストを引数にして、各要素に'_{<index>番号}.png'をつけたリストを返す関数を作る

['filename1', 'filename2', 'filename2']

['filename1_0.png', 'filename2_1.png', 'filename3_2.png']



- ✓ Numerical Pythonの略
- ✓ 科学や数学のための数値計算ライブラリ
- ✓ 行列計算が得意

ベクトル: 大きさだけではなく向きを持つ(速度や力など)

$$v = (3, 2)$$



行列: ベクトルを複数行(もしくは列)にまとめたもの

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Broadcasting

0	0	0	0
10	10	10	10
20	20	20	20
30	30	30	30

+

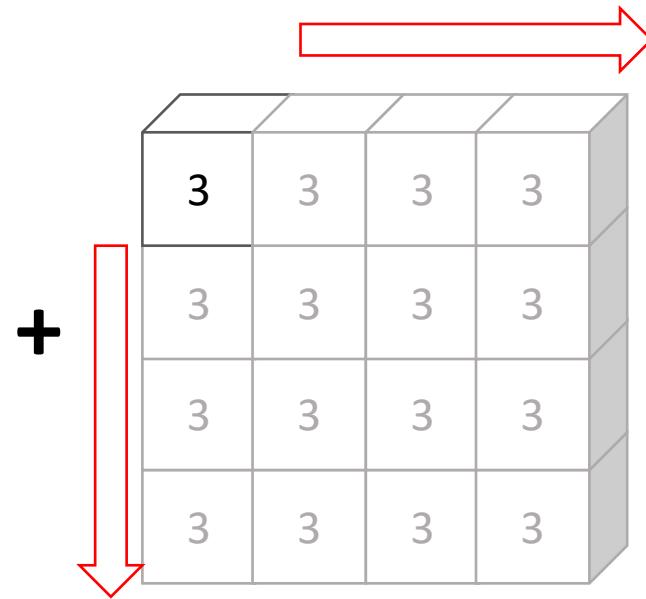
0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3



=

0	1	2	3
10	11	12	13
20	21	22	23
30	31	32	33

0	0	0	0
10	10	10	10
20	20	20	20
30	30	30	30



3	3	3	3
13	13	13	13
23	23	23	23
33	33	33	33

0	0	0	0
---	---	---	---

1次元 (rank: 1)

2次元 (rank: 2)

例：白黒画像

0	0	0	0
10	10	10	10
20	20	20	20
30	30	30	30

(高さ,横)

3次元 (rank: 3)

例：カラー画像

0	0	0	0
10	10	10	10
20	20	20	20
30	30	30	30

(高さ,横,奥行き)

4次元 (rank: 4)

例：カラー画像が複数枚ある

0	0	0	0
10	10	10	10
20	20	20	20
30	30	30	30

0	0	0	0
10	10	10	10
20	20	20	20
30	30	30	30

0	0	0	0
10	10	10	10
20	20	20	20
30	30	30	30

(個数, 高さ, 横, 奥行き)

要素が全て0の ndarray

`np.zeros()`

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

要素が全て1の ndarray

`np.ones()`

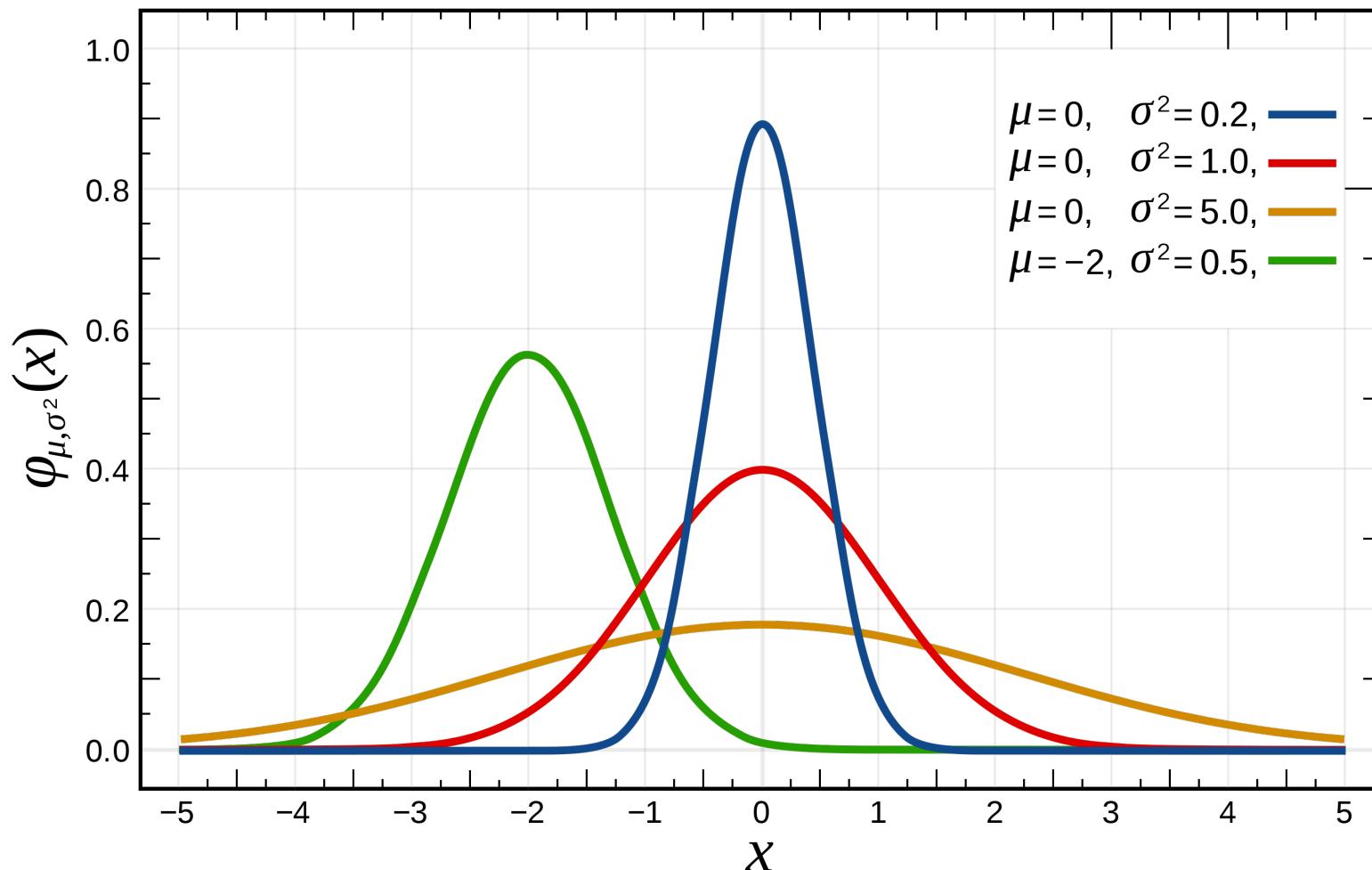
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

対角要素が全て1の ndarray

`np.eye()`

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

正規分布



μ : 平均

σ^2 : 分散

(データのばらつき)

平均が0, 分散が1のとき標準正規分布

出典: wikipedia

中央値：平均値より時間がかかる。外れ値に強い

5, 16, 22, 1, 530, 22, 10, 2, 21



並び替え

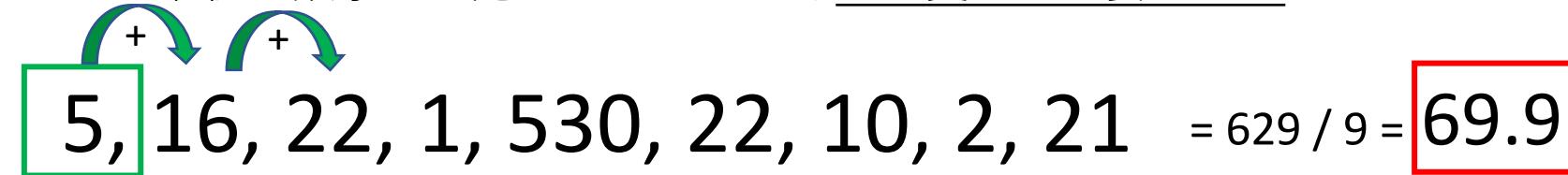
外れ値

1, 2, 5, 10, 16, 21, 22, 22, 530

中央の値

平均値

最初の数字から足していくばOK→並び変える必要がない


$$5, 16, 22, 1, 530, 22, 10, 2, 21 = 629 / 9 = \boxed{69.9}$$

外れ値の影響を受ける

標準偏差：平均との差を2乗した合計を、データ数で割った正の平方根

5, 16, 22, 1, 530, 22, 10, 2, 21

-69.9 -69.9 ...

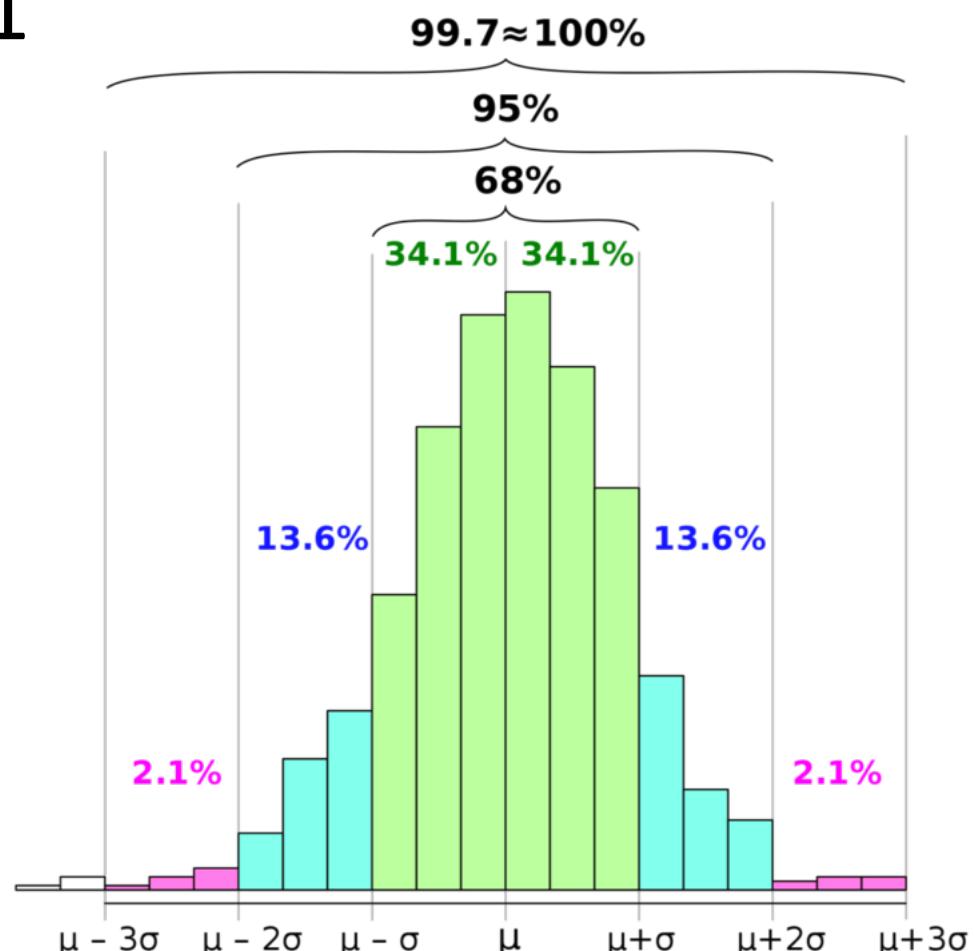
平均との差：-64.9, -53, 9....

2乗： $4210.6, 2904\dots = \text{合計 } 238734 / \text{個数 } 9 = 162.8$

68–95–99.7 rule

正規分布では

平均から ± 1 標準偏差には約68%のデータが,
平均から ± 2 標準偏差には約95%のデータが,
平均から ± 3 標準偏差には約99%のデータが含まれるという経験則

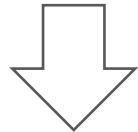


Log (Logarithm)

$$2^? = 8$$

$$\log_2 8 = \log_2 2^3 = 3\log_2 2 = 3$$

$$y = a^x$$

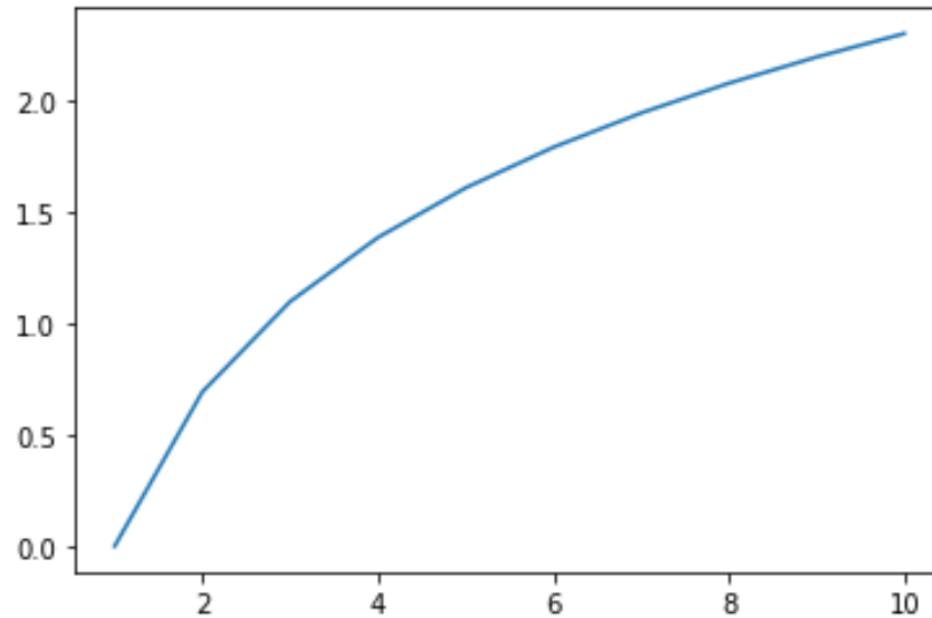


微分

$$y' = a^x \log_e a$$

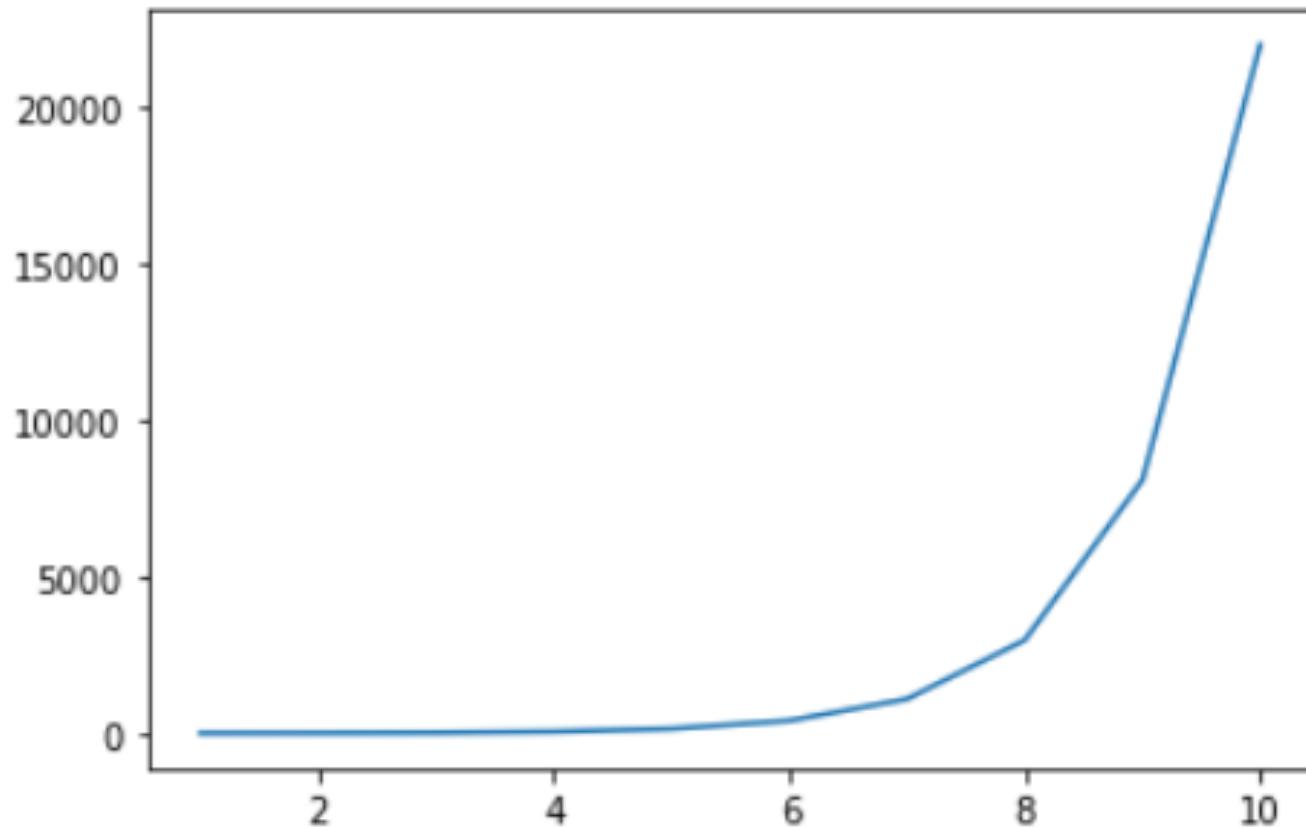
(e : ネイピア数)

$$y = \log_e x$$



Exp (exponential)

$$y = e^x$$



`np.concatenate()`

0	2	4	1	3	5
6	8	10	7	9	11
12	14	16	13	15	17

`np.stack()`

0	2	4
6	8	10
12	14	16

Pandas



- ✓ Pythonでデータサイエンスをするのに欠かせないライブラリ
 - ✓ パンダスもしくはパンダズと読む
 - ✓ 表形式のデータ処理が得意
 - ✓ DataframeとSeriesというデータ形式を使う

The diagram illustrates the structure of a Pandas DataFrame. At the top, the word "Header" is centered above a red arrow pointing downwards. To the left of the first column, the word "Column" is written in red, with a red box highlighting the "PassengerId" header cell. A red arrow points from the "Column" label to the "PassengerId" cell. The DataFrame itself consists of six rows of data. Row 0 contains the columns PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, and Embarked. Row 1 is highlighted with a red box and has an additional column Cumings, Mrs. John Bradley (Florence Briggs Th... before the Name column. Row 4 also has an additional column Allen, Mr. William Henry before the Name column. The data for each row includes values for each of the 12 columns.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	Allen, Mr. William Henry	male	35.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Pandas



- ✓ Pythonでデータサイエンスをするのに欠かせないライブラリ
- ✓ パンダスもしくはパンダズと読む
- ✓ 表形式のデータ処理が得意
- ✓ DataframeとSeriesというデータ形式を使う

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Nan	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0 26.0	1 0	0	PC 17599 STON/O2. 3101282	71.2833 7.9250	C85 NaN	C S
2	3	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
3	4	0	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Nan	S

DataFrame

表データ

Name	Sex	Age
John	male	22
Emily	female	31
Ben	male	42
George	male	23

表データ

Series
Series
Series
Series

Name	Sex	Age
John	male	22
Emily	female	31
Ben	male	42
George	male	23

表データ

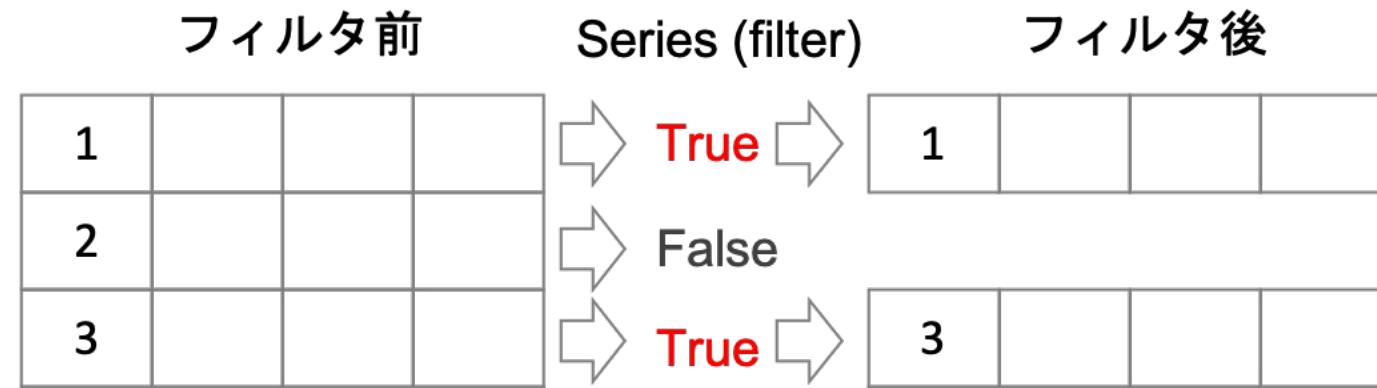
Index	Index Name	Index Sex	Index Age
0	0 John	0 male	0 22
1	1 Emily	1 female	1 31
2	2 Ben	2 male	2 42
3	3 George	3 male	3 23

Diagram illustrating the structure of the DataFrame:

- The DataFrame is composed of three Series: "Name", "Sex", and "Age".
- Each Series is represented by a red box around its corresponding column.
- Red arrows point from the labels "Series" below each column to the respective red boxes.

今回使用するデータセット

<https://www.kaggle.com/ibtesama/getting-started-with-a-movie-recommendation-system>



language	revenue
en	1000
en	2000
ja	0
ja	1000
ja	5000
cn	2000
cn	5000
cn	2000

groupby



各グループの平均値など

language	revenue
en	1500
ja	2000
cn	3000

languageカラムでグループ化されている

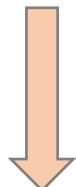
左の表

右の表

df1.merge(df2)

左の表に右の表を結合するイメージ

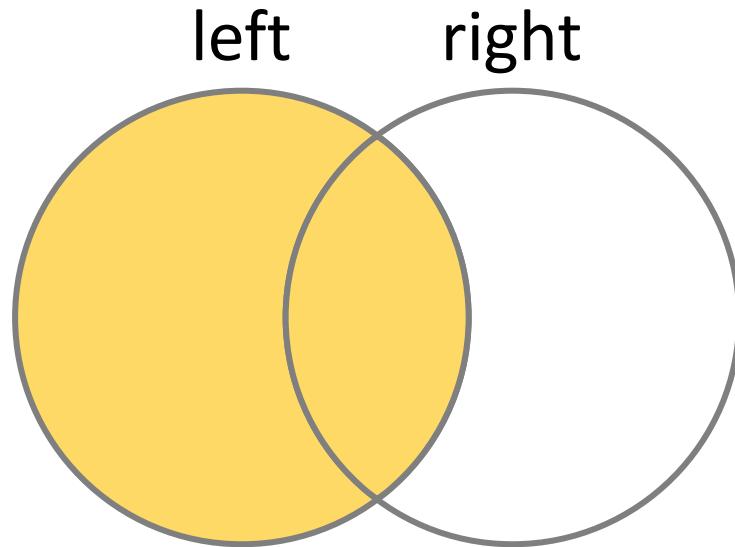
Left				Right			
Key	A	B		Key	C	D	
0	k0	a0	b0	0	c0	d0	
1	k1	a1	b1	1	c1	d1	
2	k2	a2	b2	2	c2	d2	



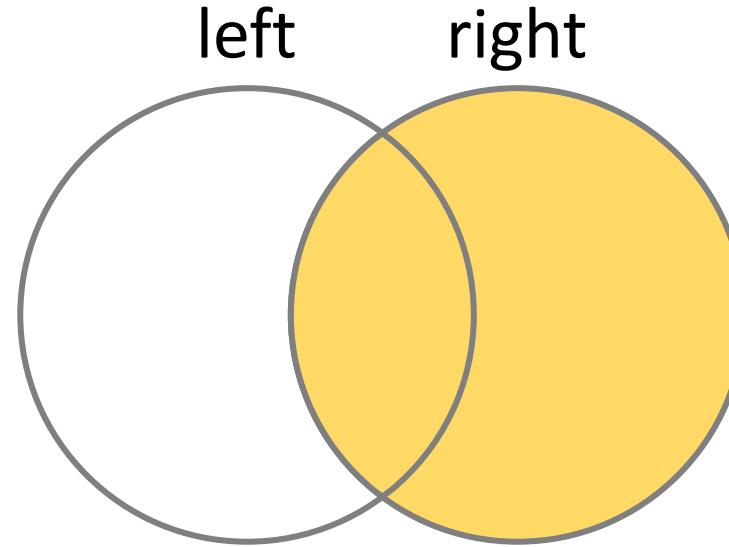
Merged

Key	A	B	C	D	
0	k0	a0	b0	c0	d0
1	k1	a1	b1	c1	d1
2	k2	a2	b2	c2	d2

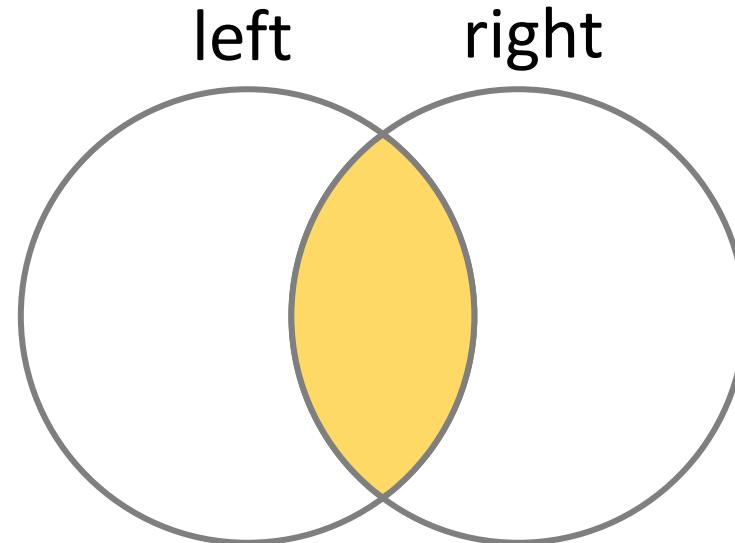
how='left'



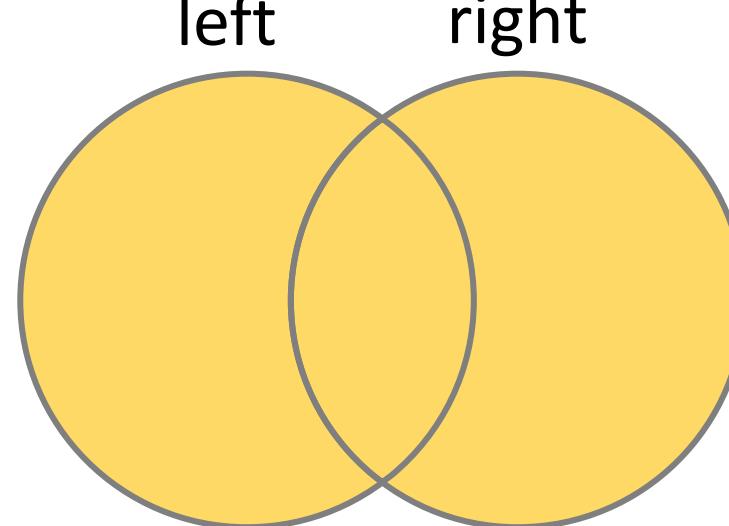
how='right'



how='inner'



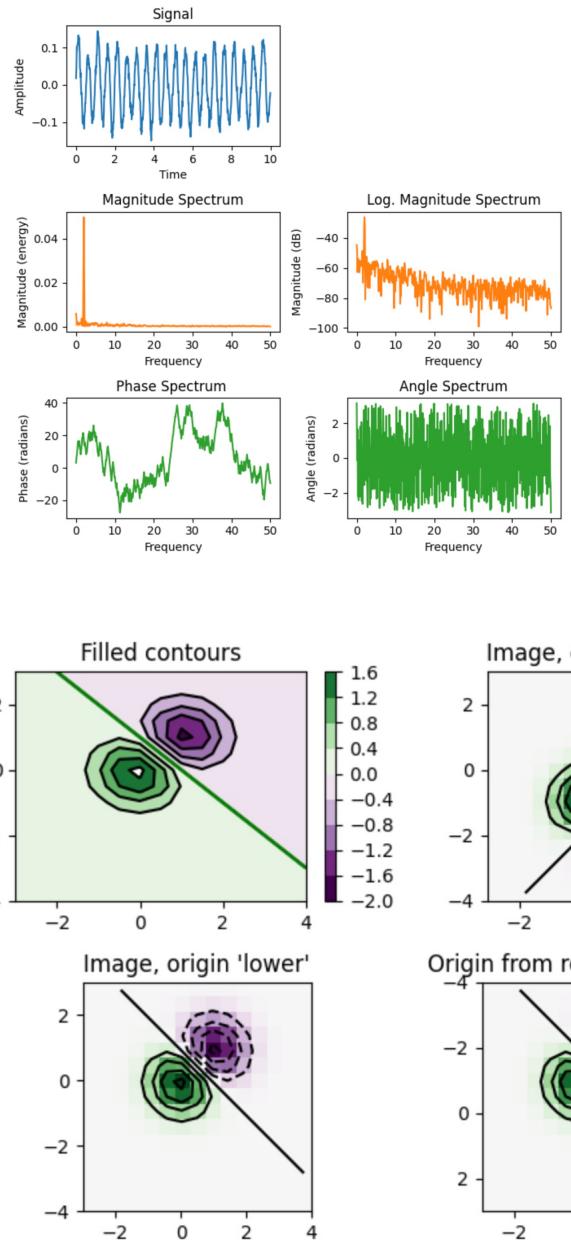
how='outer'



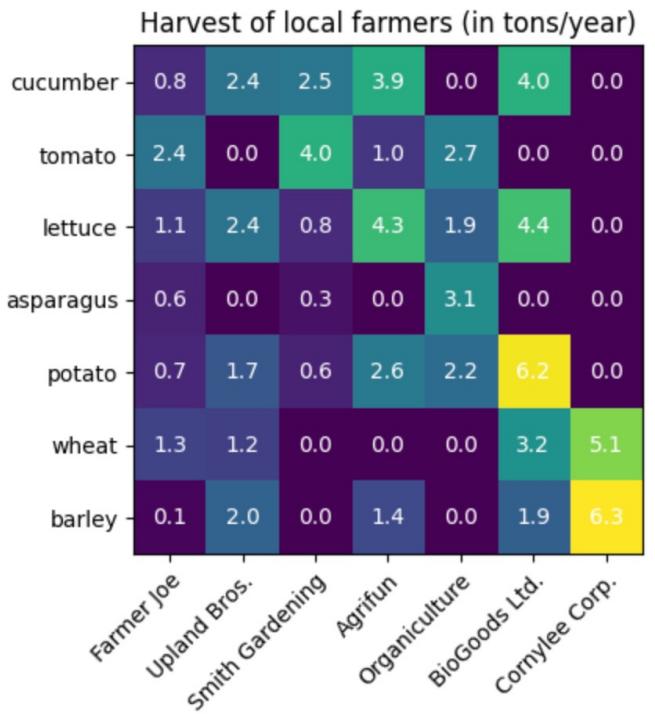
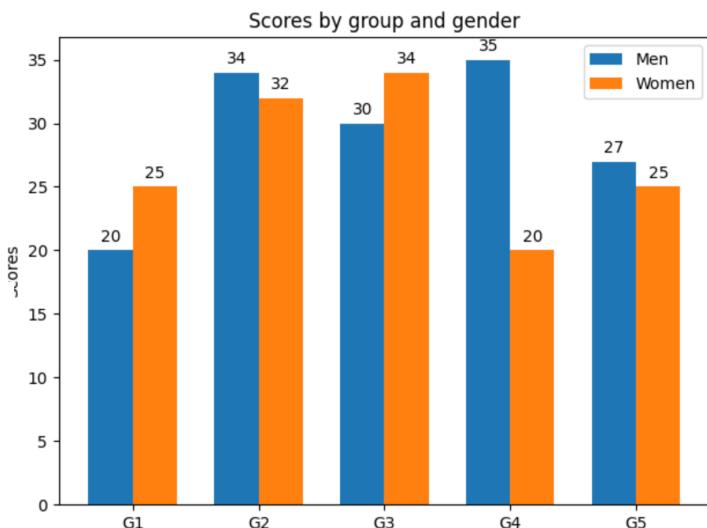
ちょっとした計算をする時にLambda関数を使う

Date	Type	City	Revenue	Sales Tax Rate	Tax
10/1	credit	Tokyo	1000	8%	80
10/2	credit	Tokyo	3000	10%	300
10/3	cash	Osaka	2000	8%	160

表計算についてはまた後ほど. . .



matplotlib

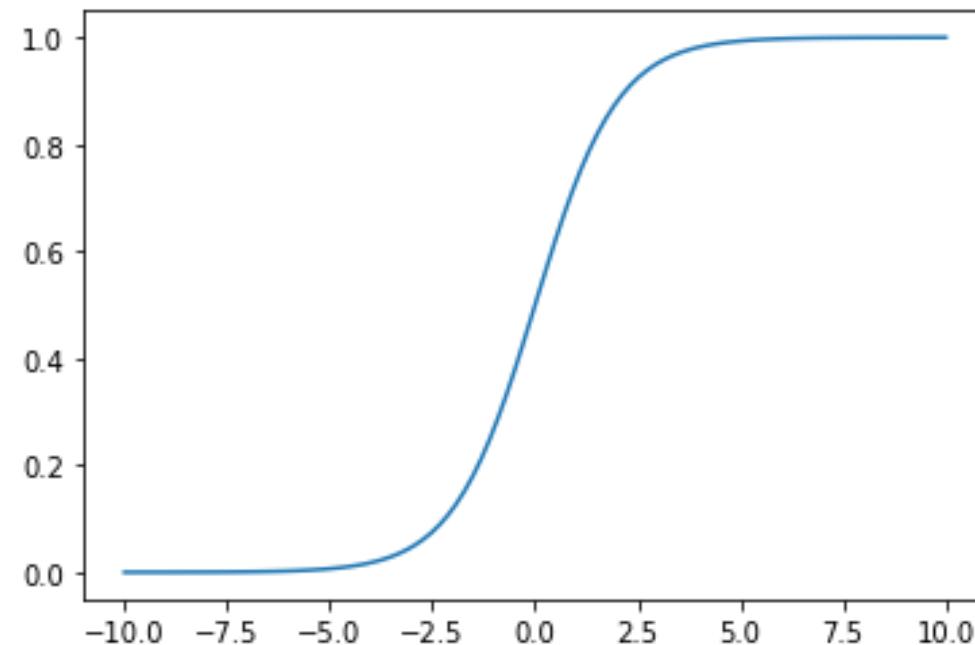


Examples

<https://matplotlib.org/gallery/index.html>

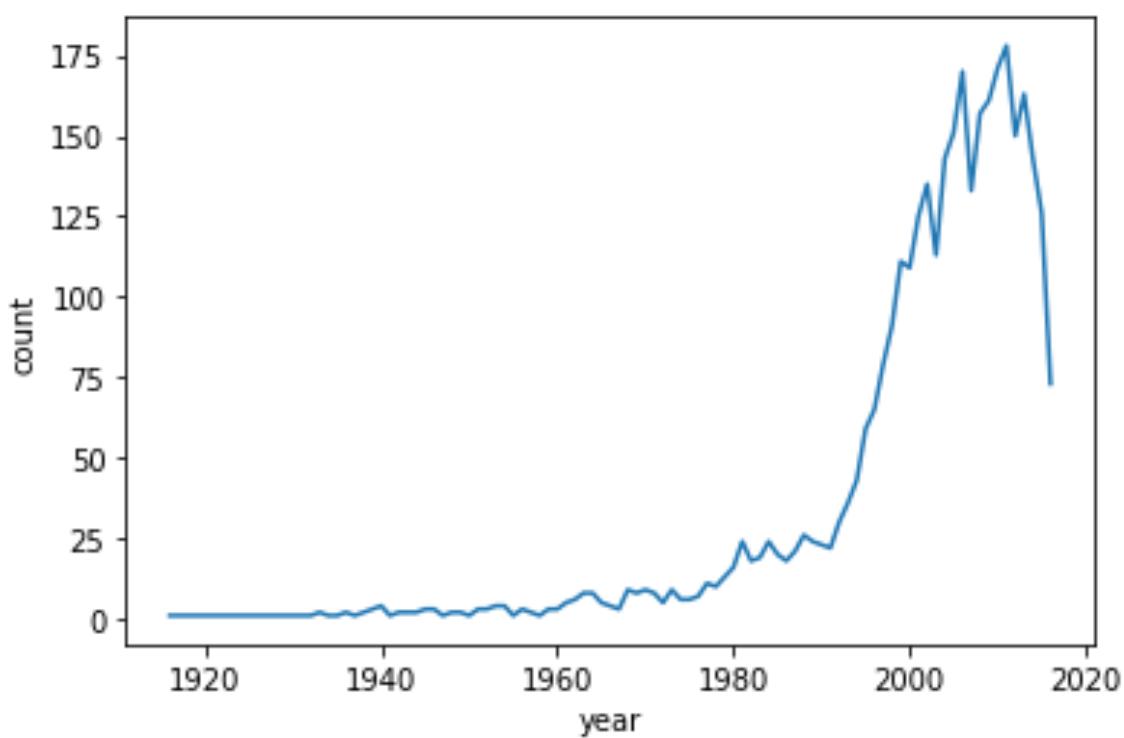
シグモイド関数を描画してみよう！

$$S(x) = \frac{1}{1 + e^{-x}}$$

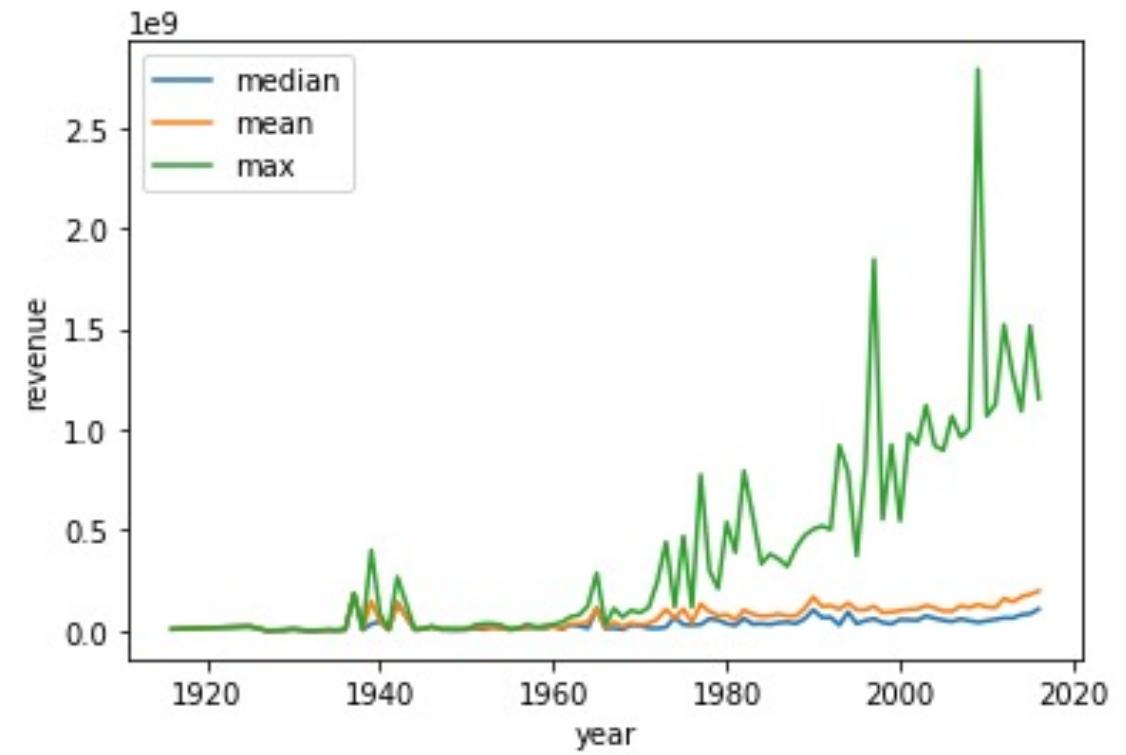


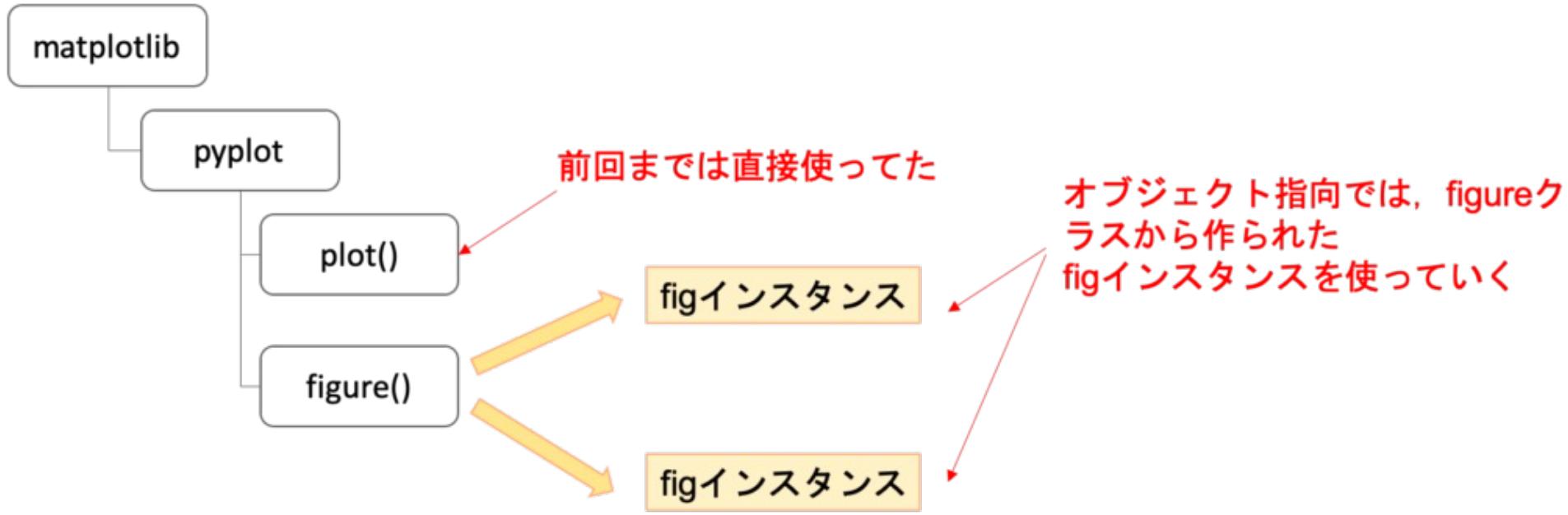
Plotしてみよう！

映画数の年推移

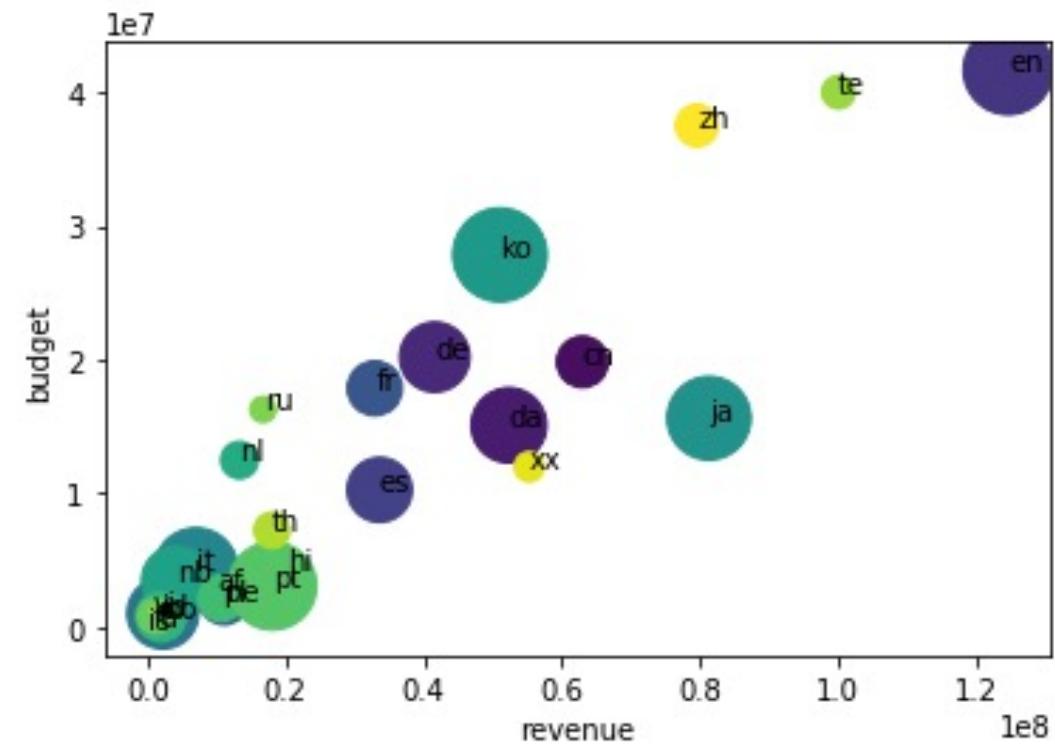
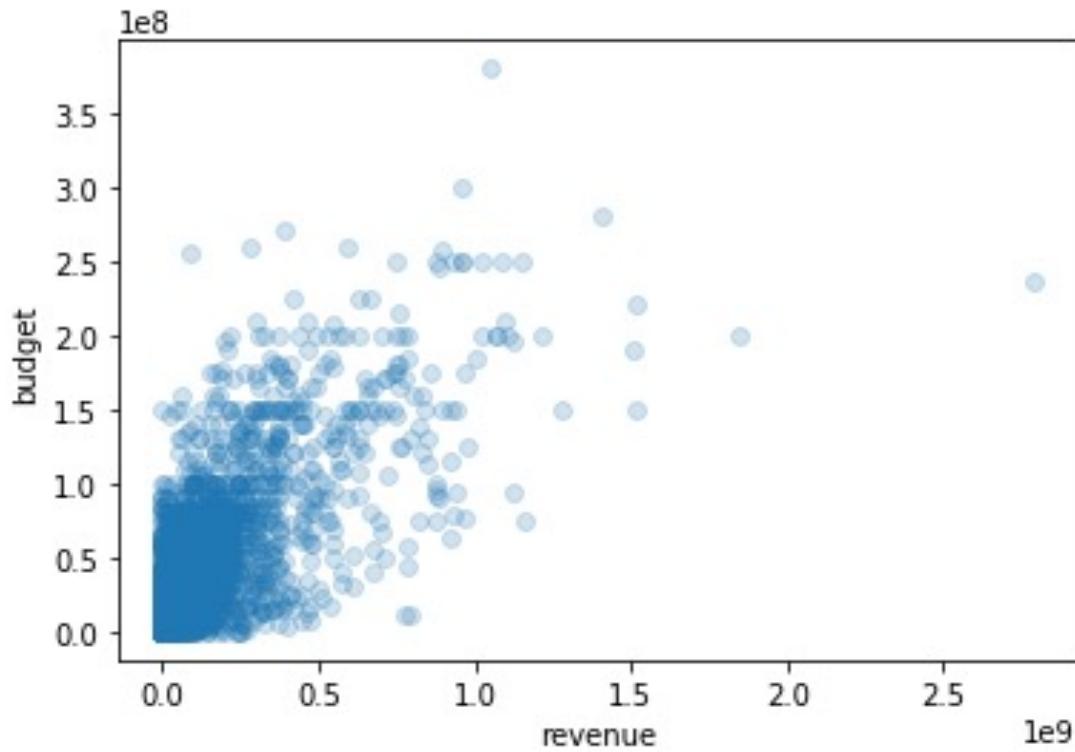


revenueの年推移(中央値, 平均, 最大値)

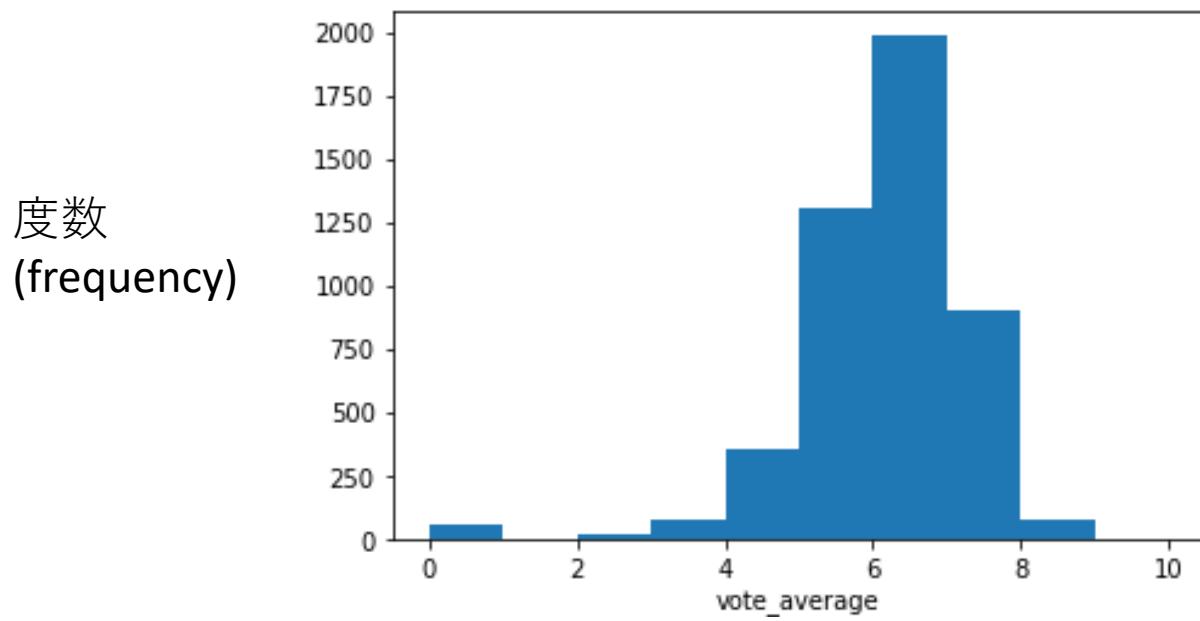




散布図: plt.scatter()



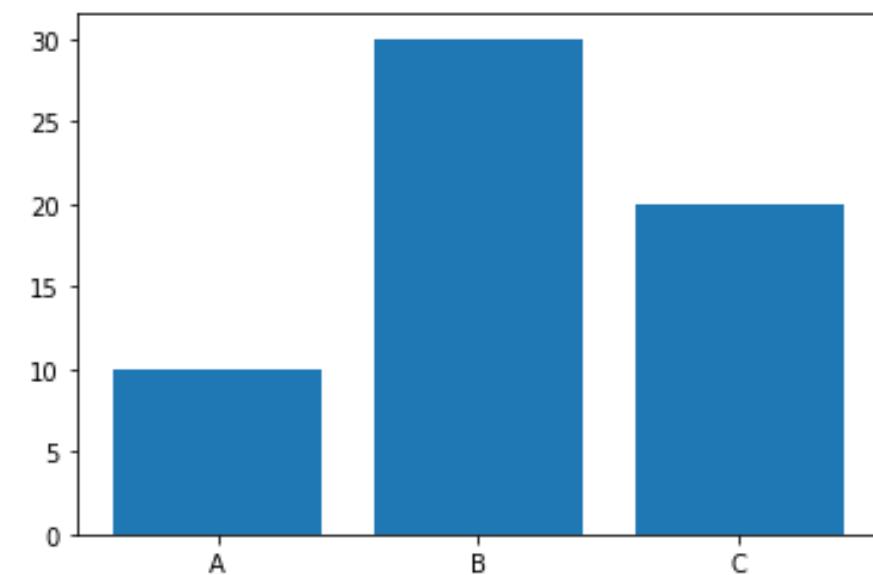
ヒストグラム



階級(bin)

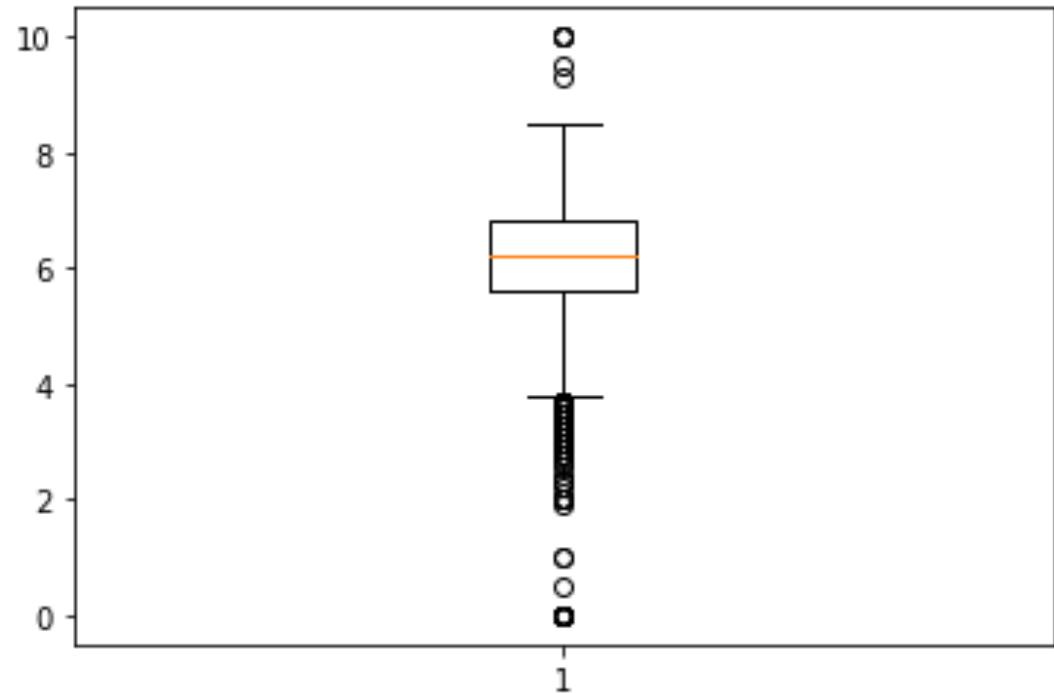
連続値に対して使う

棒グラフ



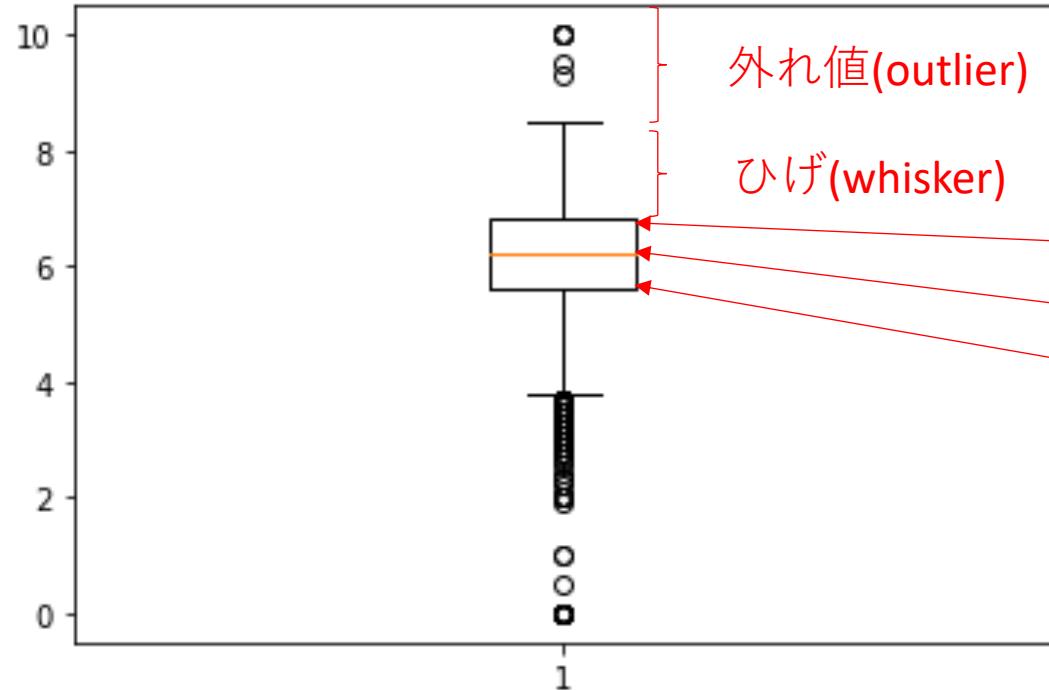
カテゴリカルな値に対して使う

箱ひげ図 (box plot)



vote averageのbox plot

箱ひげ図 (box plot)



IQR = 第3四分位点(Q3) – 第1四分位点(Q1)

第3四分位点(upper quartile)

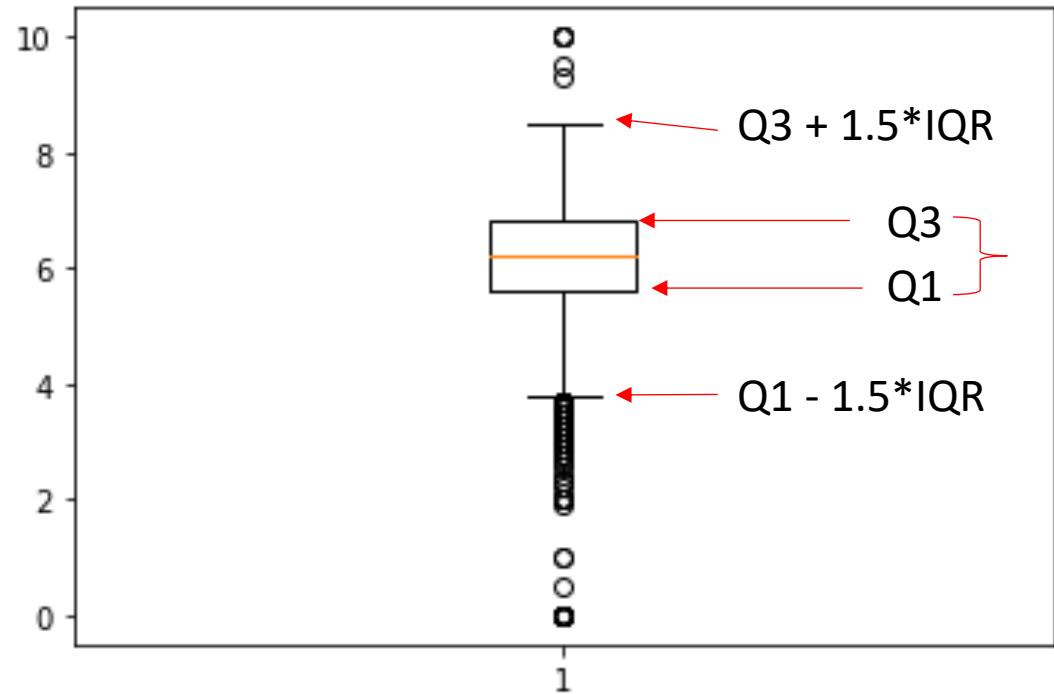
中央値(median)

第1四分位点(lower quartile)

$Q1 - 1.5 * IQR$ と $Q3 + 1.5 * IQR$

vote average の box plot

箱ひげ図 (box plot)

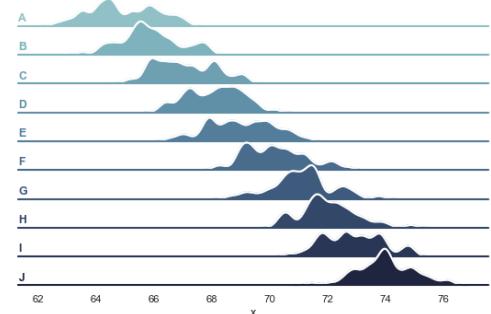
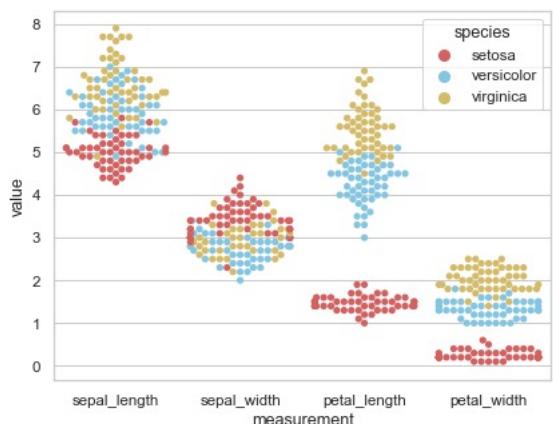
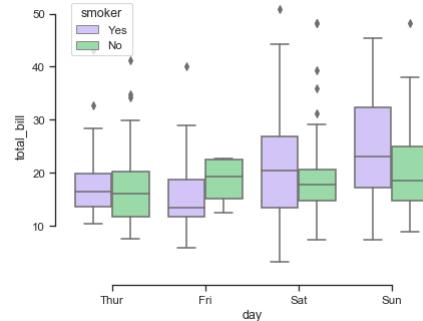
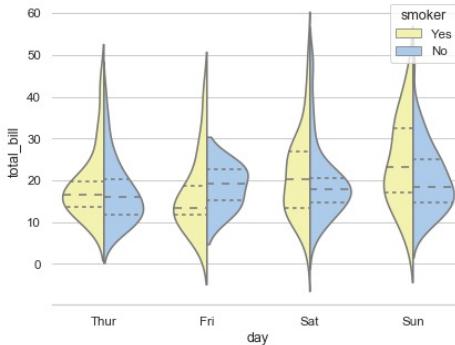


IQR (Interquartile range) = $Q3 - Q1$

vote average の box plot

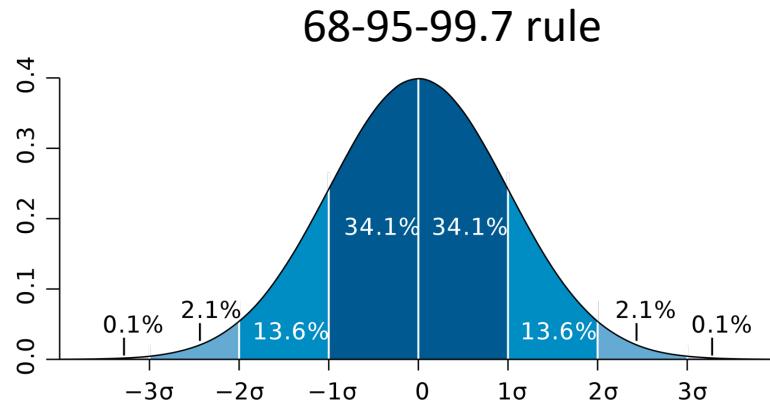
Seaborn

- matplotlibのラッパー ライブライアリ
- より洗練されたデザイン
- 簡単に使える



確率密度関数 (probability density function: PDF)

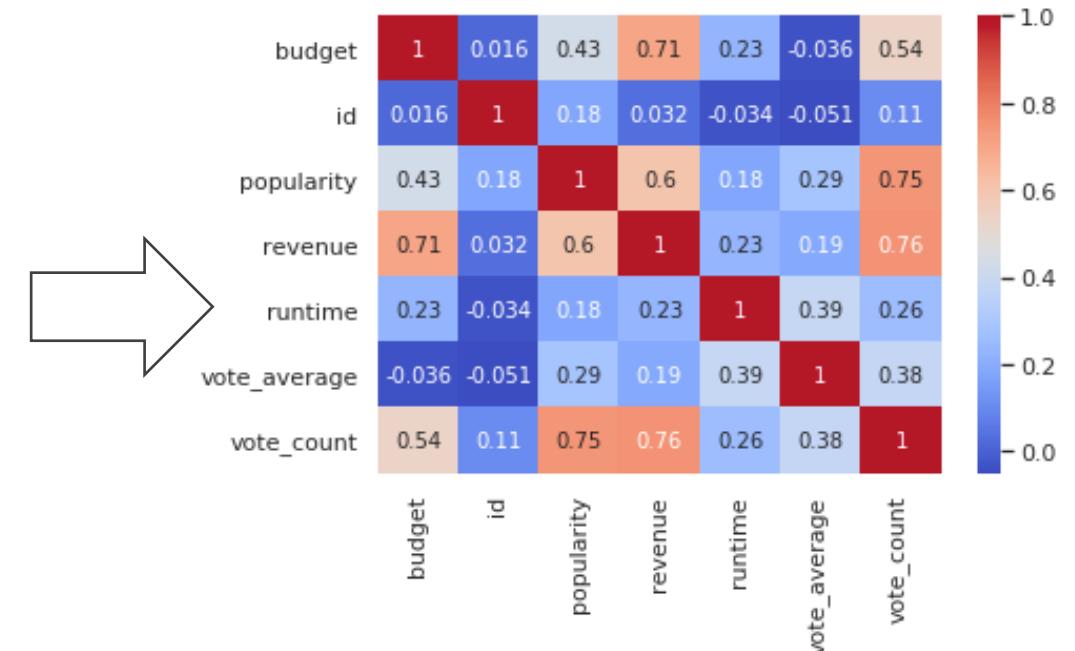
確率密度関数がわかると、確率変数がある値の範囲を取る確率が求められる！



カーネル密度推定 (kernel density estimation: KDE) は確率密度関数を推定する 1 つの方法。

Heatmap

	budget	id	popularity	revenue	runtime	vote_average	vote_count
budget	1.000000	0.015819	0.431744	0.705306	0.229712	-0.035757	0.539997
id	0.015819	1.000000	0.180645	0.031826	-0.034371	-0.050815	0.109066
popularity	0.431744	0.180645	1.000000	0.602122	0.182388	0.288189	0.749005
revenue	0.705306	0.031826	0.602122	1.000000	0.233236	0.188014	0.756143
runtime	0.229712	-0.034371	0.182388	0.233236	1.000000	0.386199	0.258101
vote_average	-0.035757	-0.050815	0.288189	0.188014	0.386199	1.000000	0.380825
vote_count	0.539997	0.109066	0.749005	0.756143	0.258101	0.380825	1.000000



- ✓ 表の各値の大きさに応じて色付けしたもの
- ✓ 一目で値の大小がわかる
- ✓ データの傾向を見るのにも使える。

データサイエンスでは特に相関表や混同行列によく使う

-相関表：相関係数を表にしたもの

-混同行列：分類器(AIモデル)がうまく分類できたもの・できなかったものを表にしたもの

	Class A (predicted)	Class B (predicted)
Class A (actual)	120	2
Class B (actual)	4	132

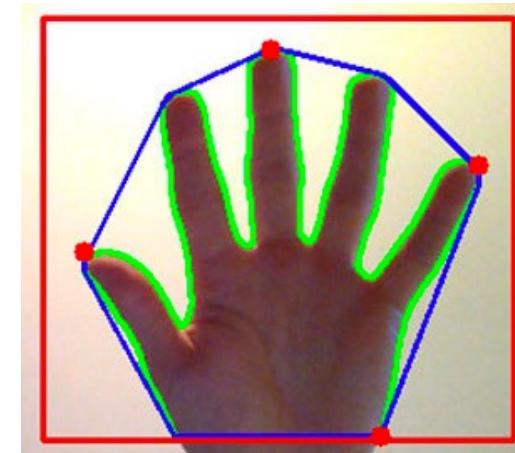
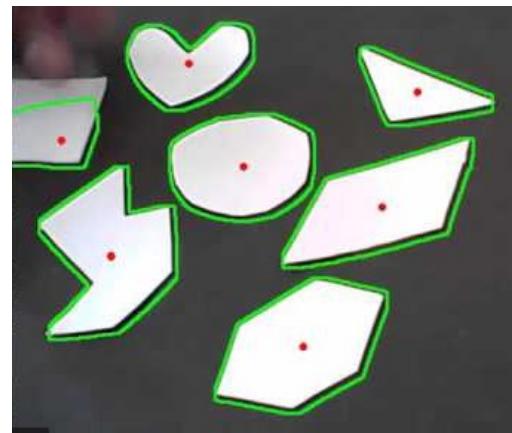
※混同行列の例



(CV: コンピュータ・ビジョン)

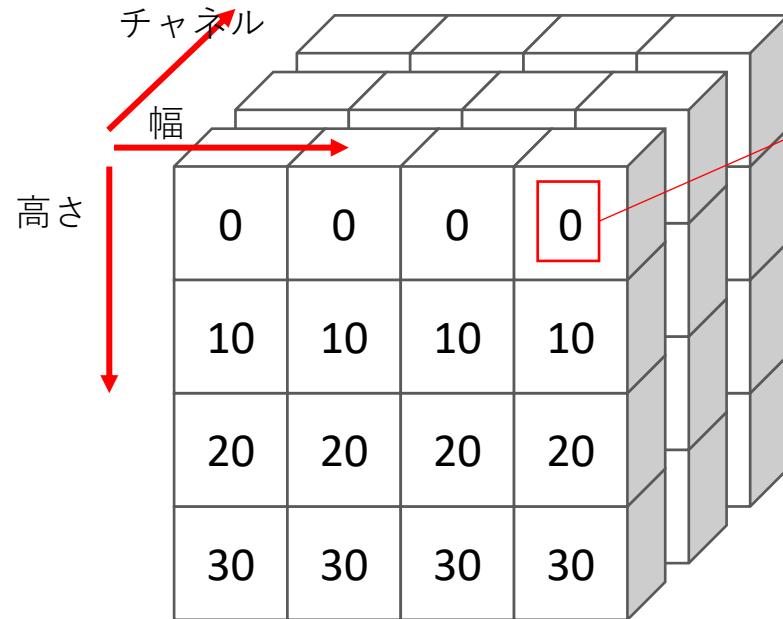
Pythonで画像を扱うためのライブラリ

OpenCVで様々な画像処理ができる。例) 輪郭取得, 中心点取得



※他にもPython Imaging Library (PIL)というライブラリがあるが、OpenCVの方がNumPyとの相性がいいのでお勧め

カラー画像のデータ構造 (3rankのNumPy Array)



shape = (高さ, 横幅, チャネル数)

輝度値 : 0 ~ 255 (unsigned の 8bit の int)

3channelあるので(高さ, 横, 3)のndarrayになる

画像の高さx画像の横幅の行列

```
[[[127, 137, 225],  
 [127, 137, 224],  
 [119, 134, 227],  
 ...,  
 [128, 141, 227],  
 [124, 150, 232],  
 [104, 120, 213]],
```

```
[[127, 137, 225],  
 [127, 136, 224],  
 [119, 134, 227],  
 ...,  
 [130, 144, 230],  
 [126, 155, 238],  
 [105, 124, 219]],
```

一つ一つの値がピクセルの値(輝度値)を表している



RGBを足し合わせてカラーになる



Binarization (2値化)

グレースケール



2値画像 (Binary Image)



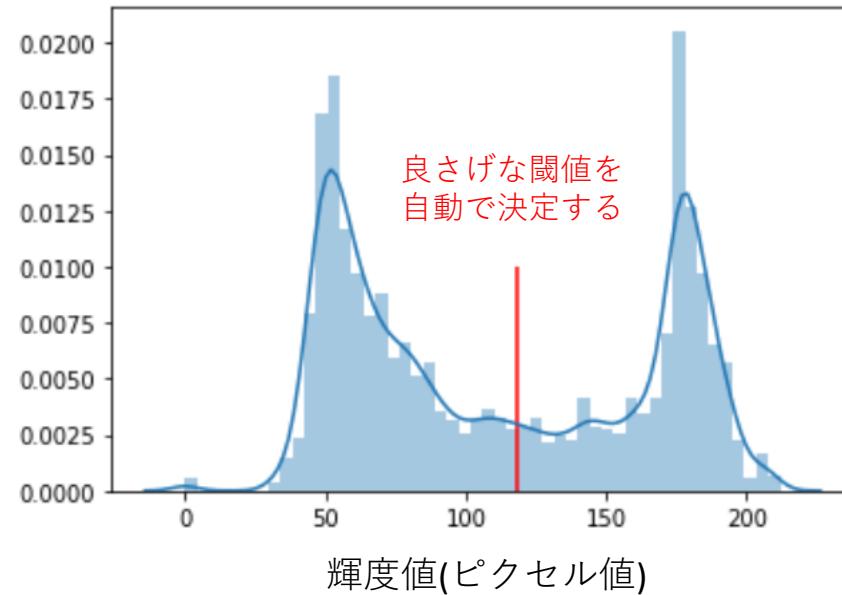
1. 単純に閾値を指定して2値化する
2. 大津の 2 値化 (Otsu's binarization)
3. Adaptive Thresholding

大津の 2 値化 (Otsu's binarization)

画像

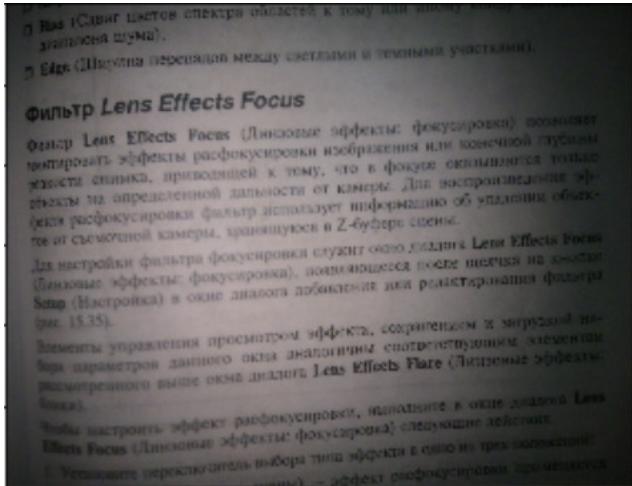


輝度値のヒストグラム

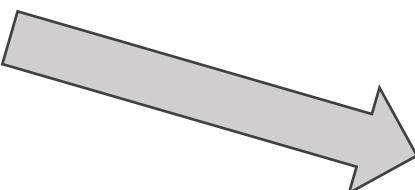
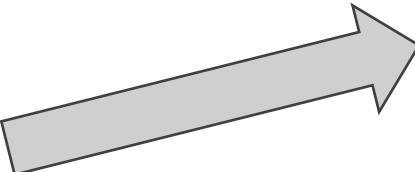


- ✓ 閾値を自動で決定してくれる
- ✓ 双峰性(bimodal)ヒストグラムに使える
- ✓ 線形判別分析法 (LDA: Linear Discriminant Analysis)を画像に適用したアルゴリズム

Adaptive Thresholding

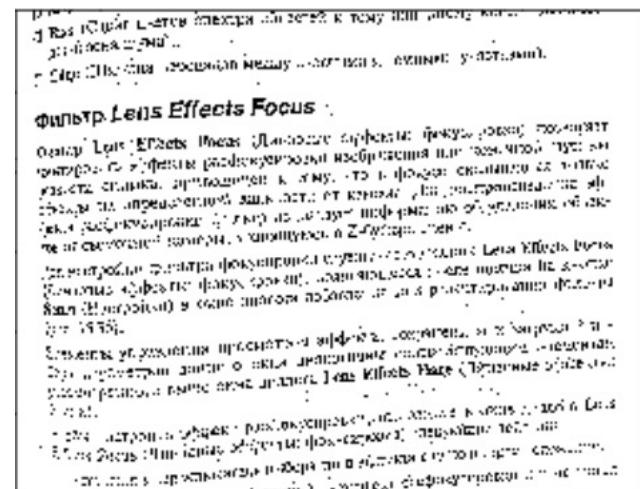
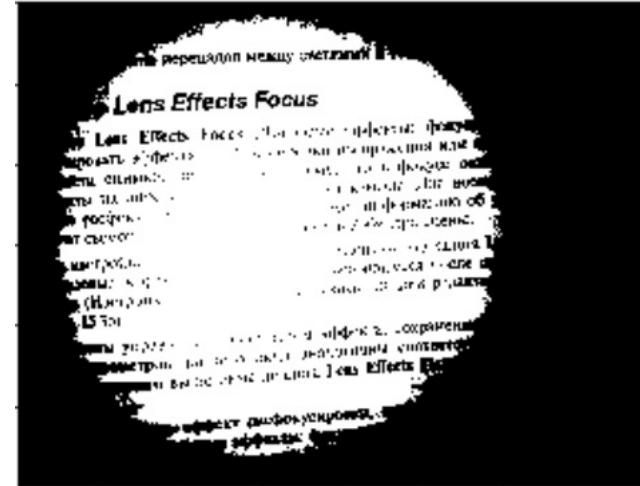


一律に閾値を指定
(global thresholding)

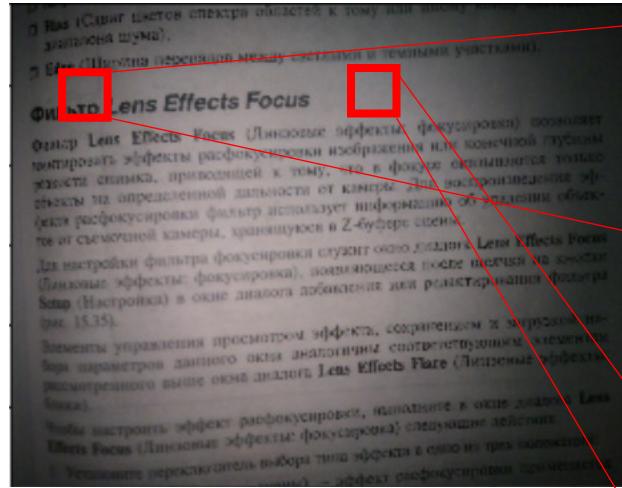


Adaptive Thresholding

- ✓ 光の条件が画像内で異なる場合によく使われる
- ✓ OCR (文字認識アルゴリズム)と組み合わせることも



Adaptive Thresholdingのアルゴリズム



この小さいwindowの中での平均輝度値を閾値にして2値化
→これを画像全体に繰り返す



平均を閾値にしてしまうと文字がない箇所では
うまくいかないので、実際には平均値から定数を引いた値を使う

glob

今回使うデータセット(COVID-19のCTデータ)
<http://medicalsegmentation.com/covid19/>

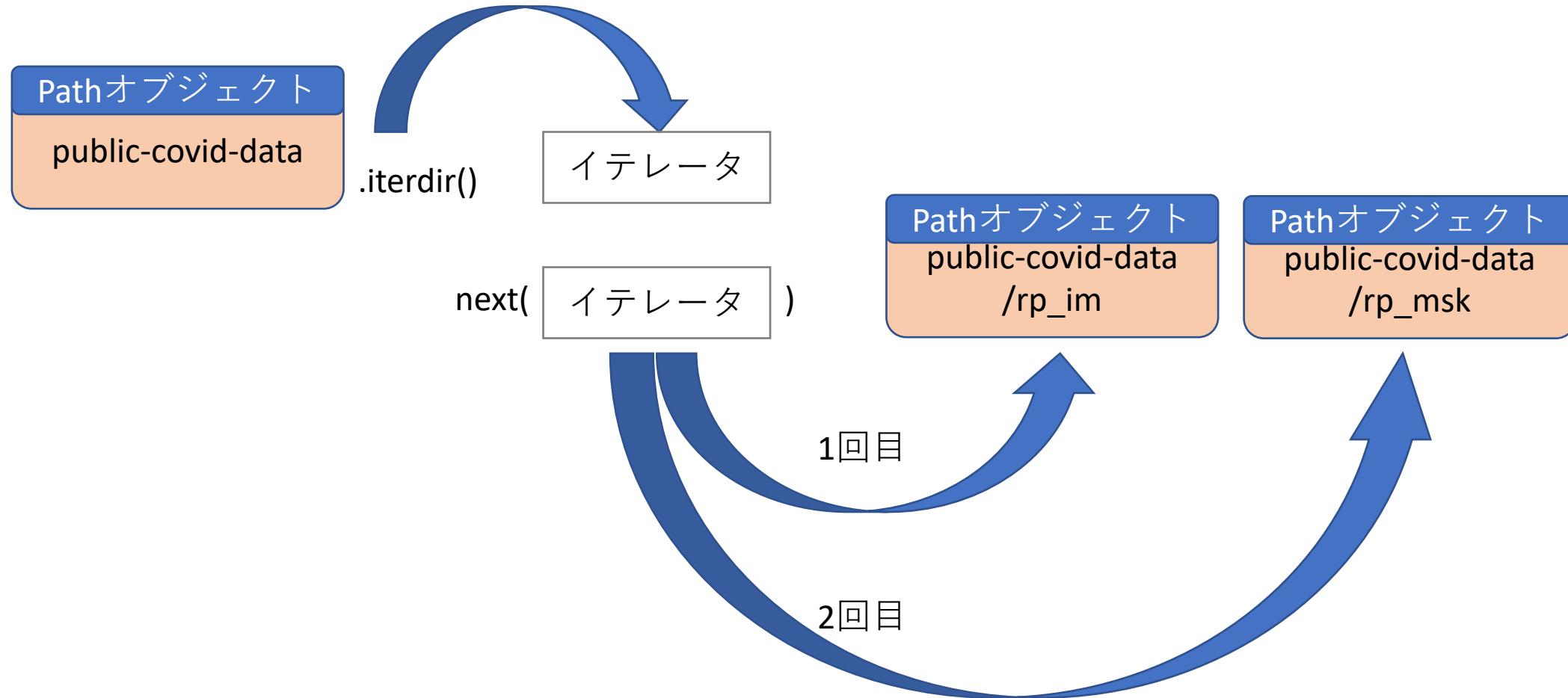
ワイルドカードを使ってファイル(パス)名のリストを取得する

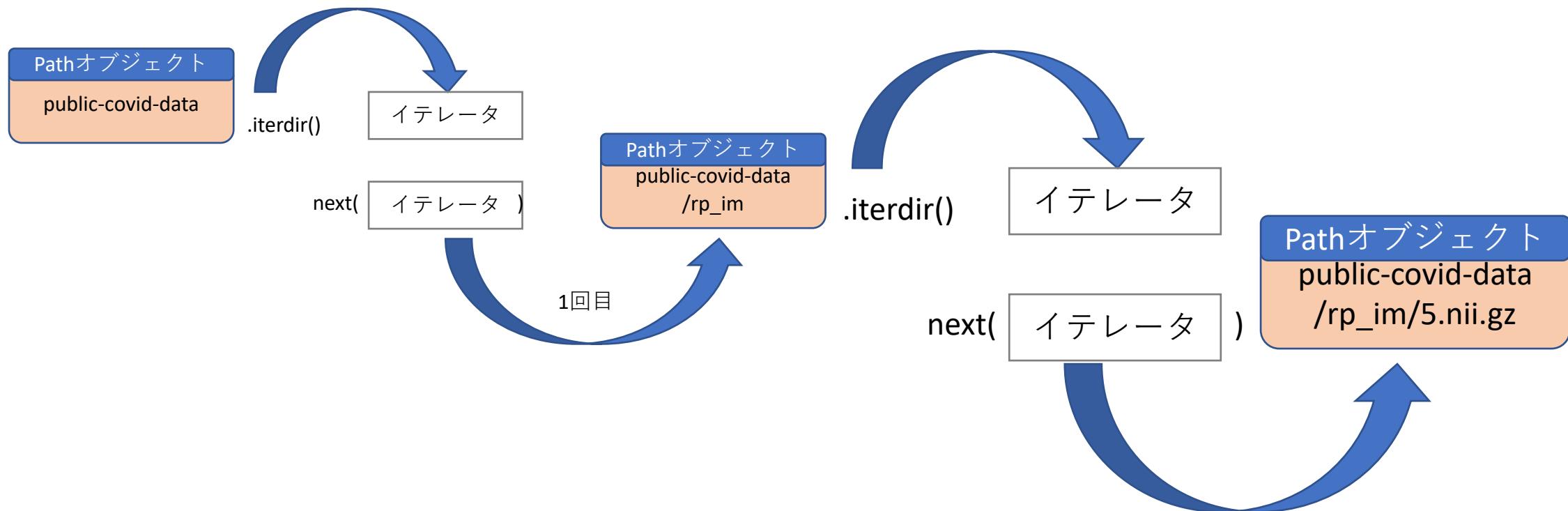
よく使うワイルドカードは「*」

ワイルドカード	解説	使用例		
		グロブ表現	マッチする文字列の例	マッチしない文字列の例
*	0文字以上の任意の文字列にマッチ	営業*	営業、 営業実績	営、 2019年度営業、 2019年度営業計画
		営業	営業、 営業実績、 2019年度営業、 2019年度営業計画	営
?	任意の1文字にマッチ	営業実績?	営業実績1、 営業実績B	営業実績、 営業実績12
[c ₁ c ₂ …c _N]	括弧内で列挙されたどれか1文字にマッチ	営業実績[123]	営業実績1、 営業実績2、 営業実績3	営業実績、 営業実績4、 営業実績12、
[c _{min} -c _{max}]	括弧内で指定された範囲内の1文字にマッチ	営業実績[0-9]	営業実績1、 営業実績3、 営業実績4、 営業実績5、 営業実績9	営業実績、 営業実績10、 営業実績B、 営業実績2019

(出典: wikipedia)

pathlib





NIfTI (Neuroimaging Informatics Technology Initiative)

- ✓ 主に脳のMRI画像のデータ形式に使われている
- ✓ nibabelというライブラリを使う(SimpleITKというツールでも可能)

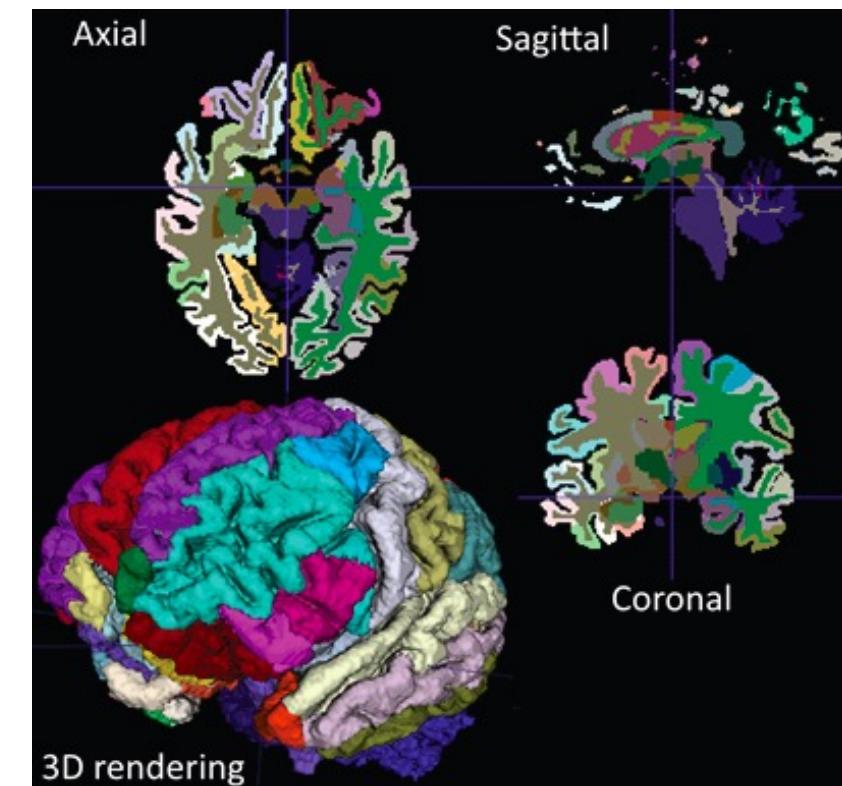
CT/MRIデータを3Dで見るならOsirix



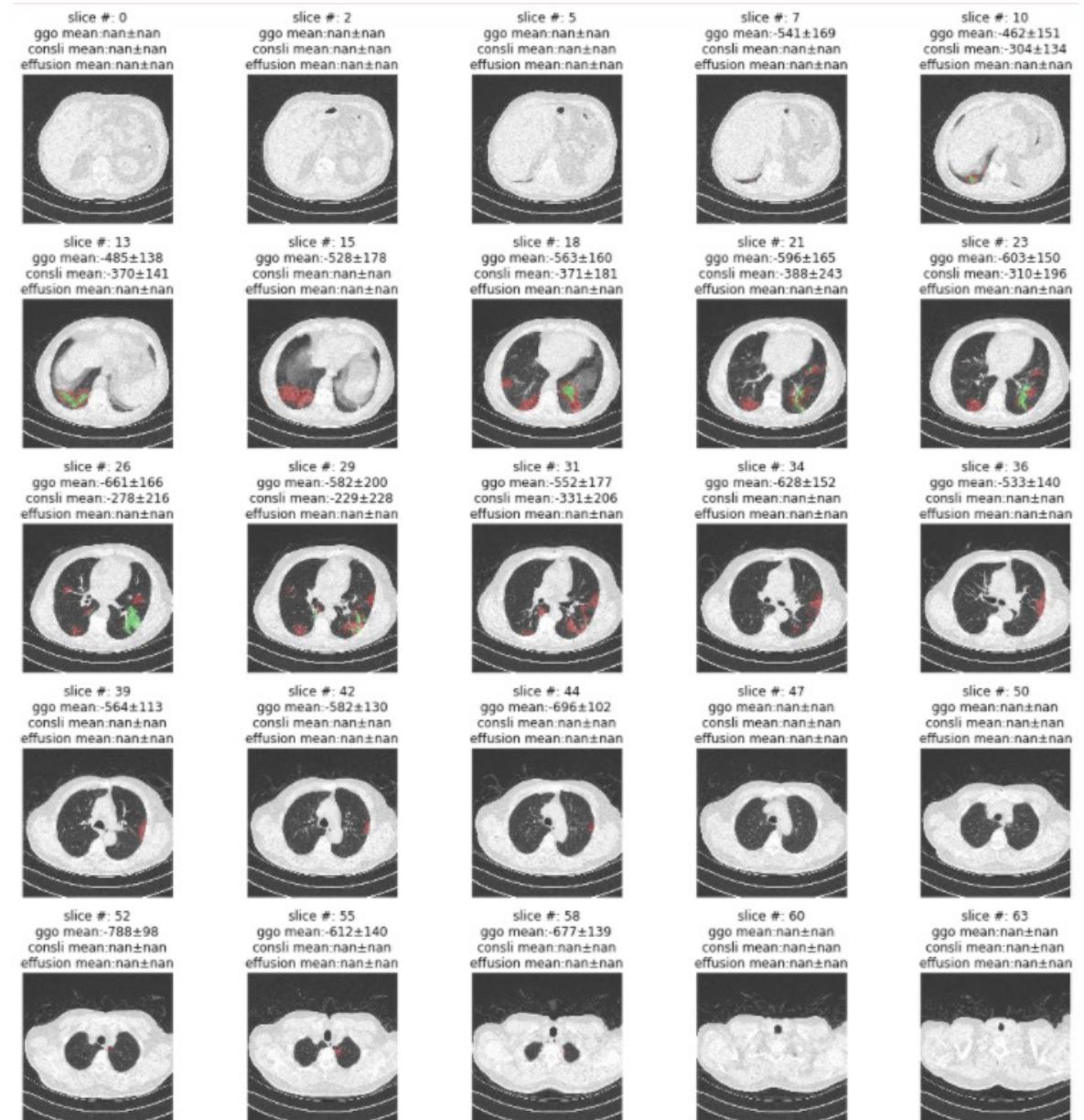
やitk-SNAP

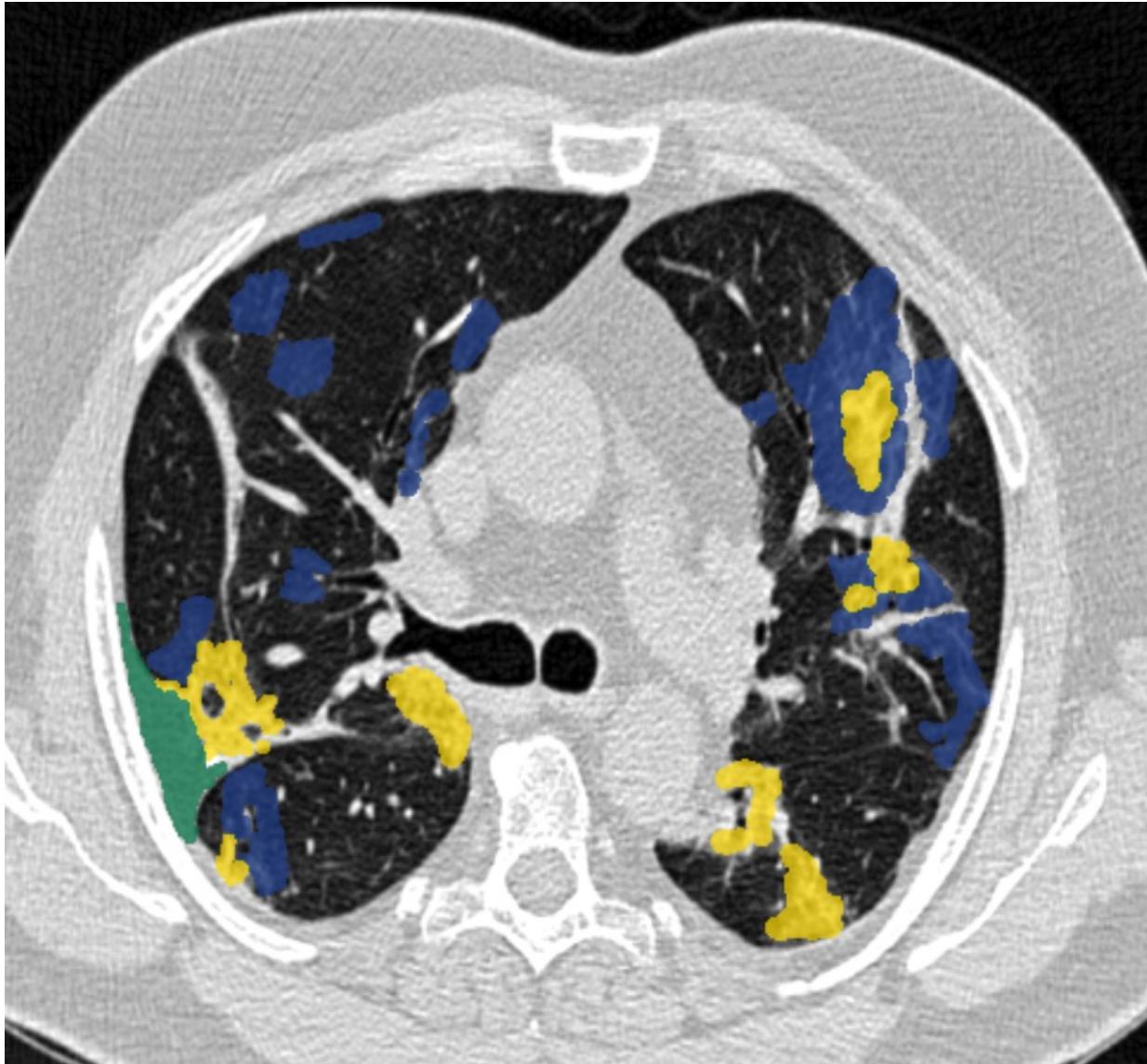


などのツールを使う



各スライスのアノテーションを一覧で表示する





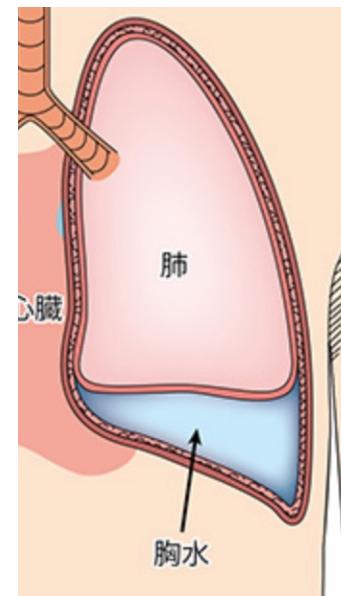
アノテーションマスク

青: *ground-glass* (mask value =1),

黄: *consolidation* (=2)

緑: *pleural effusion* (=3).

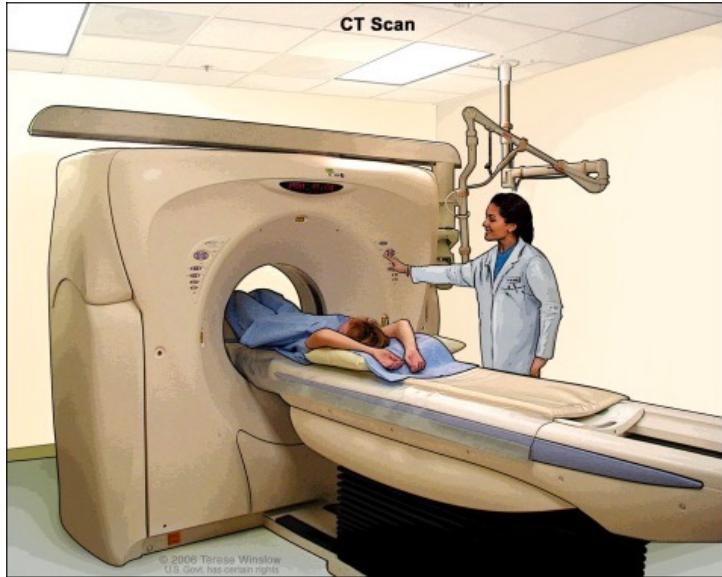
pleural effusion(胸水)



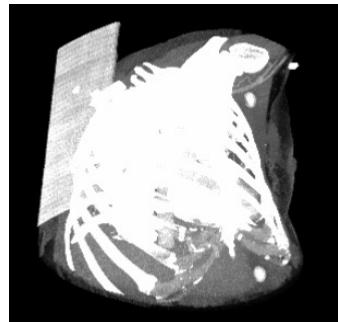
出典: 日本呼吸器学会

https://www.jrs.or.jp/modules/citizen/index.php?content_id=68

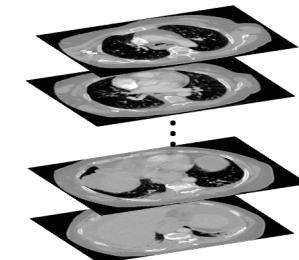
CT : Computerized Tomography



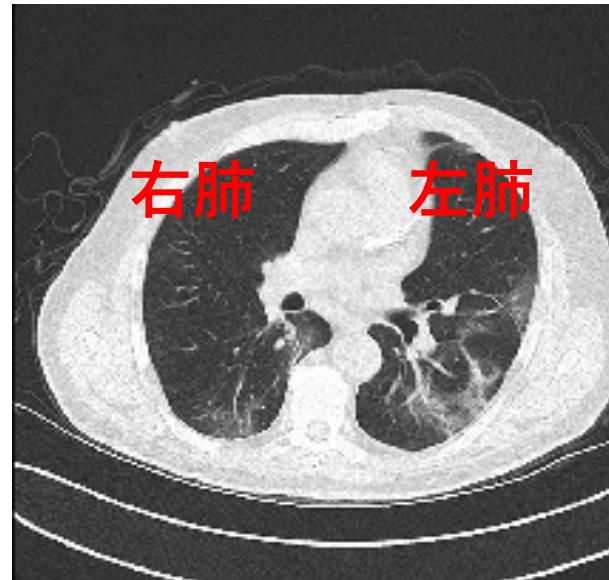
- ✓ X線を使って体の内部を画像化
- ✓ 通常,複数の**Axial**画像で保存
- ✓ 3次元のデータ
- ✓ 画像左側=体の右側
- ✓ ピクセルの値(CT値)はHounsfield Unit(HU)を使う
- ✓ HUは水が0, 空気が-1000



CTは3次元データ



Axial面の画像で保存



Relation Between Chest CT Findings and Clinical Conditions of Coronavirus Disease (COVID-19) Pneumonia: A Multicenter Study [Wei Zhao et.al]

of patients:

101 COVID-19 confirmed patients

Characteristic or Finding	All Patients (n=101)	Nonemergency Group (n=87)	Emergency Group (n=14)	p
Imaging finding				
Ground-glass opacities	87 (86.1)	73 (83.9)	14 (100)	0.106
Consolidation	44 (43.6)	36 (41.4)	8 (57.1)	0.270
Mixed ground-glass opacities and consolidation	65 (64.4)	56 (64.4)	9 (64.3)	0.995
Centrilobular nodules	23 (22.8)	20 (23.0)	3 (21.4)	0.897
Architectural distortion	22 (21.8)	16 (18.4)	6 (42.8)	0.040
Bronchial wall thickening	29 (28.7)	22 (25.3)	6 (42.8)	0.173
Reticulation	49 (48.5)	39 (44.8)	9 (64.3)	0.176
Subpleural bands	28 (27.7)	23 (26.4)	4 (28.6)	0.867
Traction bronchiectasis	53 (52.5)	41 (47.1)	12 (85.7)	0.007
Intrathoracic lymph node enlargement	1 (1.0)	0 (0)	1 (7.1)	0.012
Vascular enlargement	72 (71.3)	59 (67.8)	13 (92.9)	0.109
Pleural effusions	14 (13.9)	9 (10.3)	5 (35.7)	0.011
Craniocaudal distribution				0.258
Upper lung predominant	6 (5.9)	6 (6.9)	0 (0)	
Lower lung predominant	55 (54.5)	48 (55.2)	7 (50)	
No craniocaudal distribution	32 (31.7)	25 (28.7)	7 (50)	
Transverse distribution				0.493
Central	1 (1.0)	1 (1.1)	0 (0)	
Peripheral	88 (87.1)	74 (85.1)	14 (100)	
No transverse distribution	4 (4.0)	4 (4.6)	0 (0)	
Lung region distribution				0.172
Unilateral	10 (9.9)	10 (11.5)	0 (0)	
Bilateral	83 (82.2)	69 (79.3)	14 (100)	
Scattered distribution				0.001
Focal	6 (5.9)	6 (6.9)	0 (0)	
Multifocal	55 (54.5)	52 (59.8)	3 (21.4)	
Diffuse	32 (31.7)	21 (24.1)	11 (78.6)	
Extent of lesion	6.39 (0–20) ^b	5.34 (3.84) ^a	12.86 (4.59) ^a	0.000
No. without CT findings	8 (7.9)	8 (9.2)	0 (0)	NA

Note—Except where otherwise indicated, data are number with percentage in parentheses. NA = not applicable.

^aMean (SD).

^bMean (range).

マスクデータ

2	2	3	3
1	1	2	3
1	2	3	0
0	0	0	0

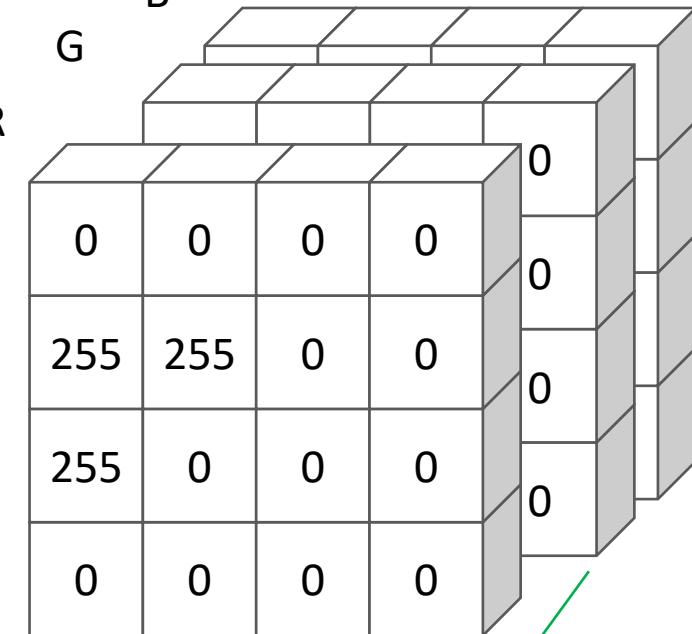
0: background [0, 0, 0]

1: Red [255, 0, 0]

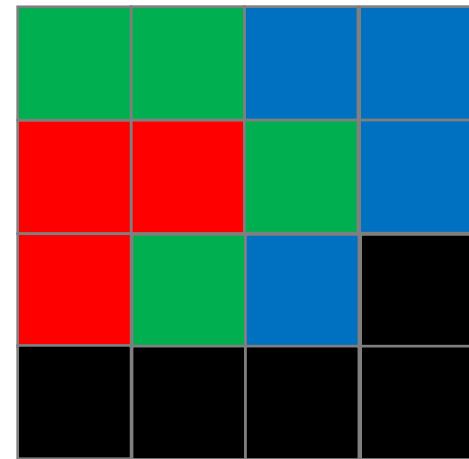
2: Green [0, 255, 0]

3: Blue [0, 0, 255]

ラベル別に色付けする(RGB) B



マスクデータ(RGB)



`plt.imshow()`

R

G

B

0	0	0	0
255	255	0	0
255	0	0	0
0	0	0	0

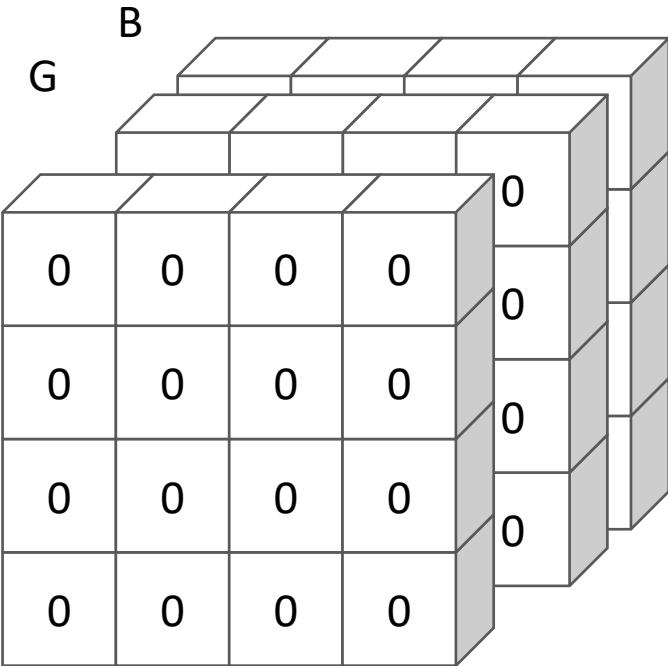
255	255	0	0
0	0	255	0
0	255	0	0
0	0	0	0

0	0	255	255
0	0	0	255
0	0	255	0
0	0	0	0

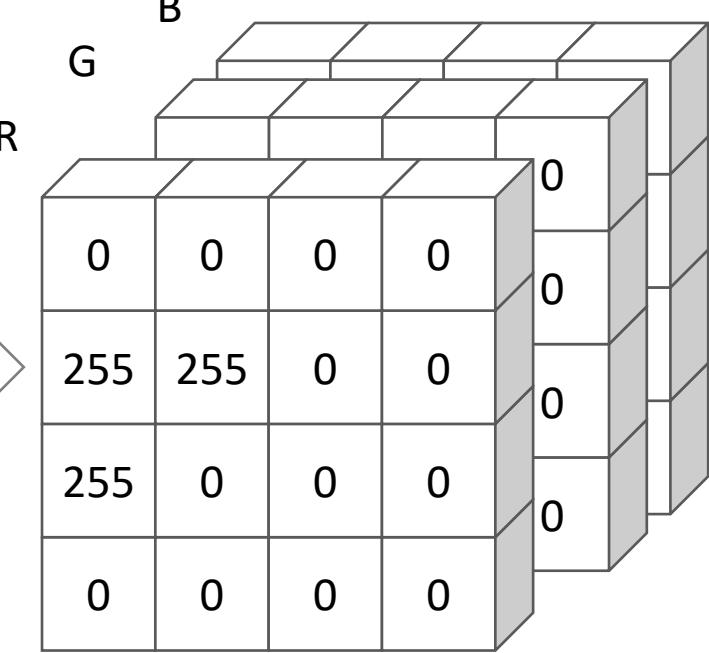
マスクデータ

2	2	3	3
1	1	2	3
1	2	3	0
0	0	0	0

全て0の箱を作成

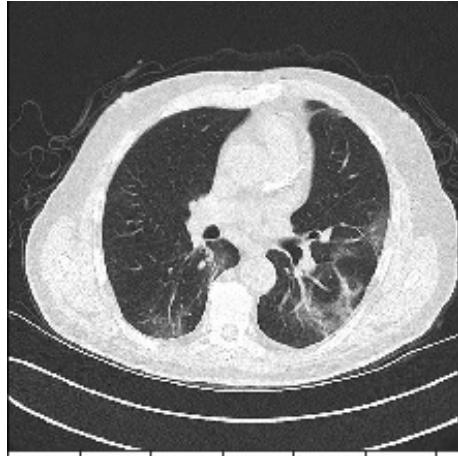


ラベル別に値を入れる(RGB)

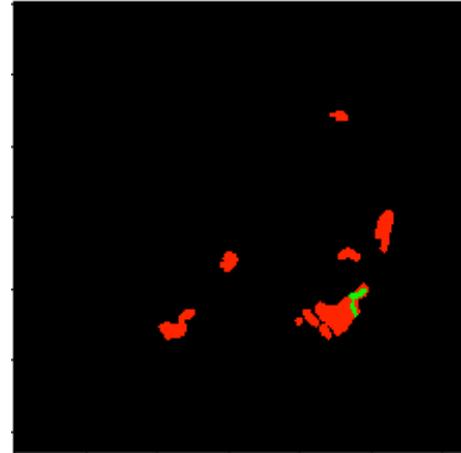


演習セクション

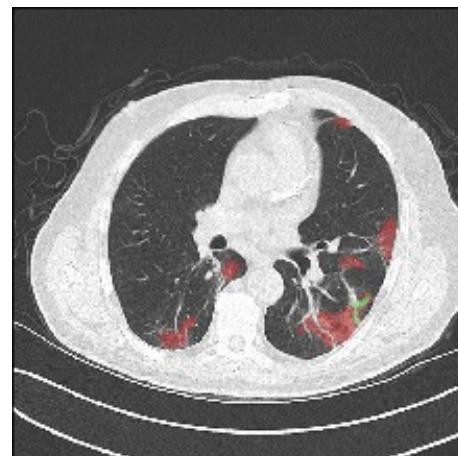
今回使うデータセット(COVID-19のCTデータ)
<http://medicalsegmentation.com/covid19/>



元のCT画像



アノテーションマスク



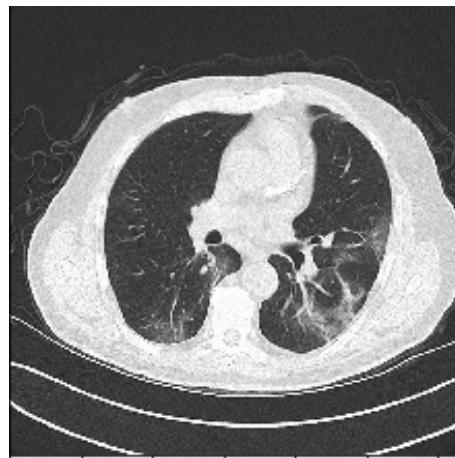
Overlay

前回のセクションで作成

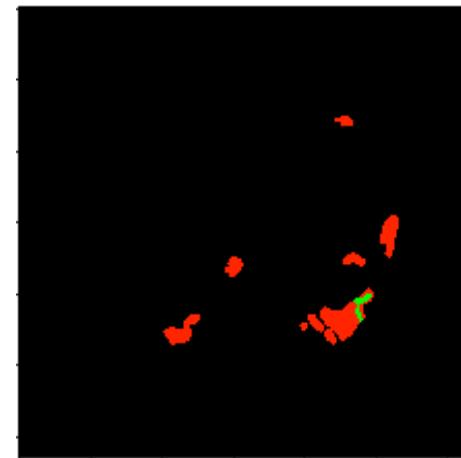
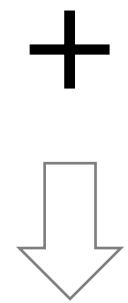
CTデータ

133	231	-123
-213	143	432
11	232	400
-1000	-900	-400

0~255ではない！！



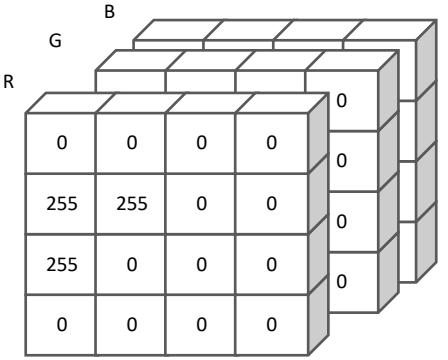
元のCT画像



アノテーションマスク



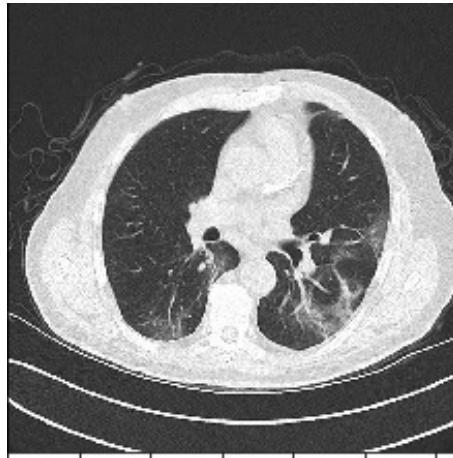
Overlay



前回のレクチャーで作成

(RGB全て同じ値)

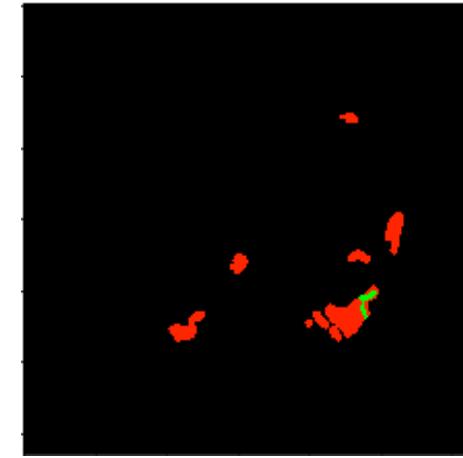
B				
G				
R				
34	34	223	213	3
43	53	53	143	0
43	121	22	40	0
20	42	167	200	0



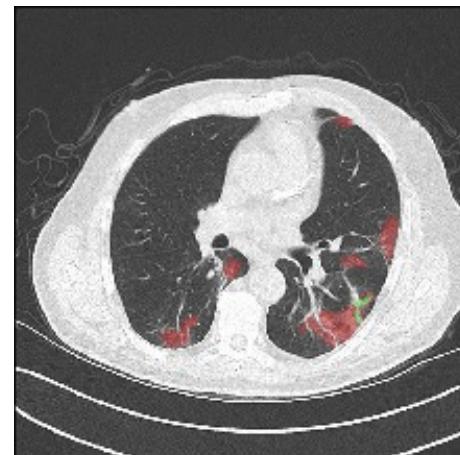
元のCT画像

前々回のレクチャーで作成

B				
G				
R				
0	0	0	0	0
255	255	0	0	0
255	0	0	0	0
0	0	0	0	0

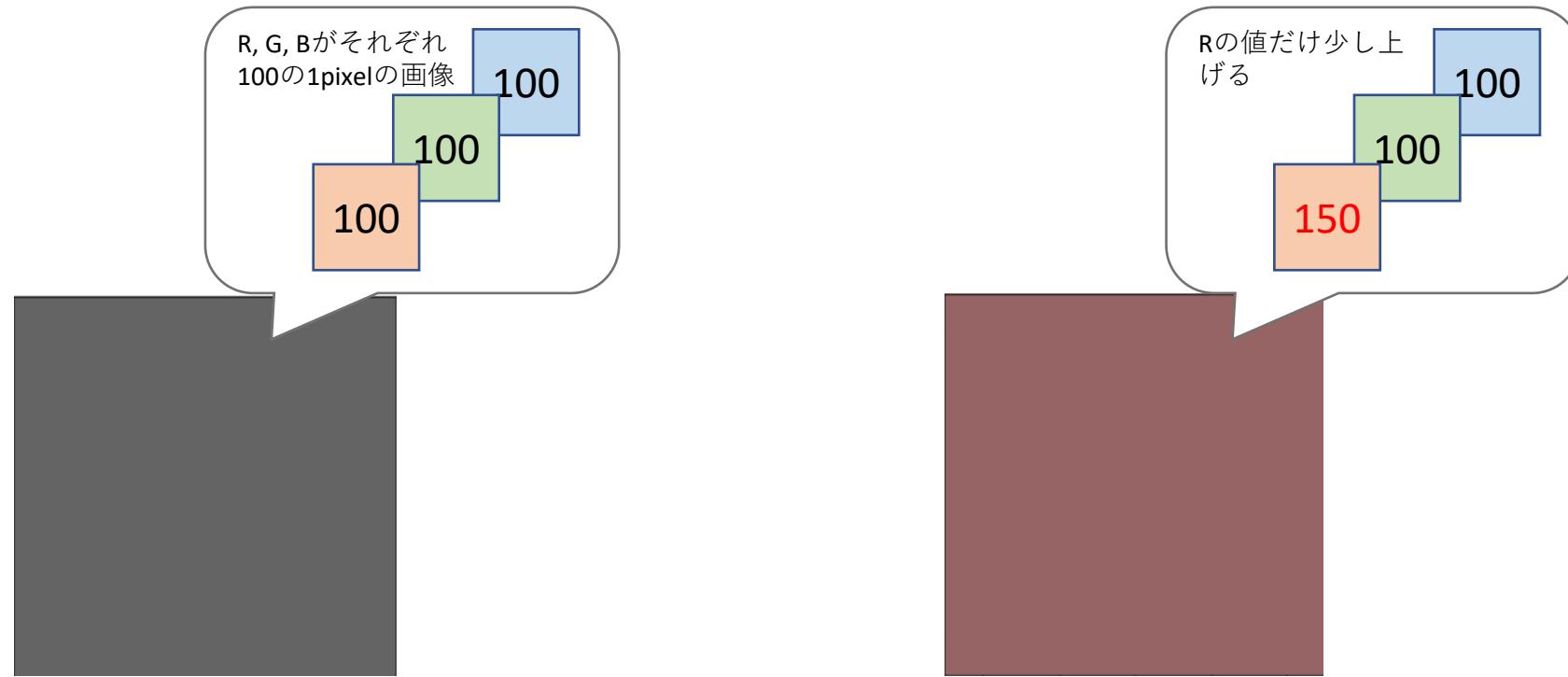


アノテーションマスク



Overlay

Overlay



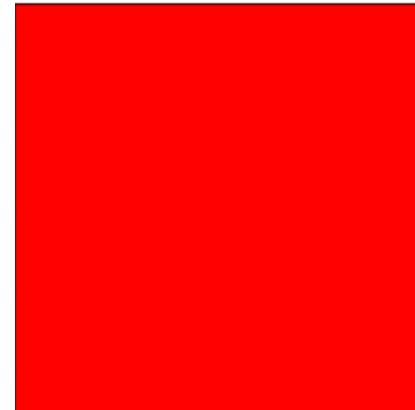
元のpixelの値 + 少しの値で色がつく

Overlay

元のCT画像のpixel

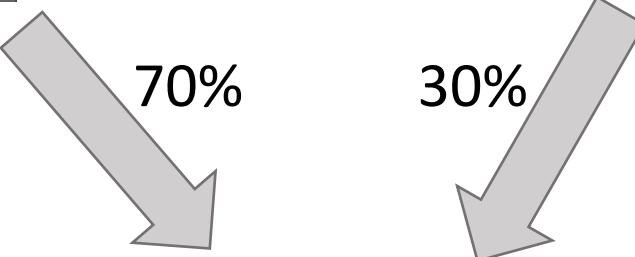


アノテーションマスクのpixel

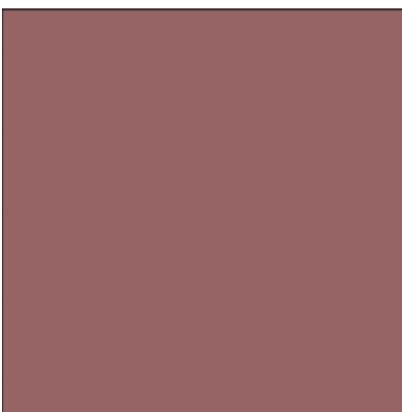
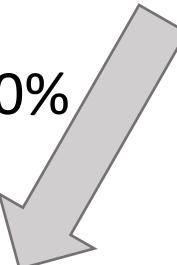


+

70%



30%



例) $100 * 0.7 + 255 * 0.3 = 146.5$

R

0	0	0	0
255	255	0	0
255	0	0	0
0	0	0	0

G

255	255	0	0
0	0	255	0
0	255	0	0
0	0	0	0

B

0	0	255	255
0	0	0	255
0	0	255	0
0	0	0	0

`np.where(a > 0, True, False)`

R

F	F	F	F
T	T	F	F
T	F	F	F
F	F	F	F

G

T	T	F	F
F	F	T	F
F	T	F	F
F	F	F	F

B

F	F	T	T
F	F	F	T
F	F	T	F
F	F	F	F

R

F	F	F	F
T	T	F	F
T	F	F	F
F	F	F	F

G

T	T	F	F
F	F	T	F
F	T	F	F
F	F	F	F

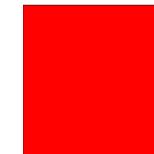
B

F	F	T	T
F	F	F	T
F	F	T	F
F	F	F	F

100x0.7



255x0.3



146.5



Rチャネル

+

=

Gチャネル

100



100



GとBチャネルはx0.7されない

Bチャネル

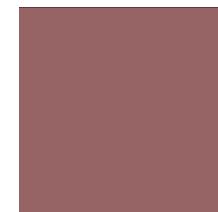
100



100

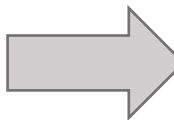


その分赤色感が減る



元のマスクデータ

2	2	3	3
1	1	2	3
1	2	3	0
0	0	0	0



R

T	T	T	T
T	T	T	T
T	T	T	F
F	F	F	F

G

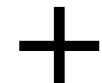
T	T	T	T
T	T	T	T
T	T	T	F
F	F	F	F

B

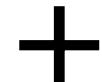
T	T	T	T
T	T	T	T
T	T	T	F
F	F	F	F

Rチャネル

$$100 \times 0.7 + 255 \times 0.3 = 146.5$$

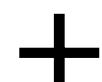


$$0 \times 0.3 + 70 = 70$$



Gチャネル

$$100 \times 0.7 + 0 \times 0.3 = 70$$

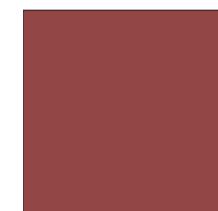


Bチャネル

$$100 \times 0.7 + 0 \times 0.3 = 70$$

GとBチャネルも同様にx0.7する

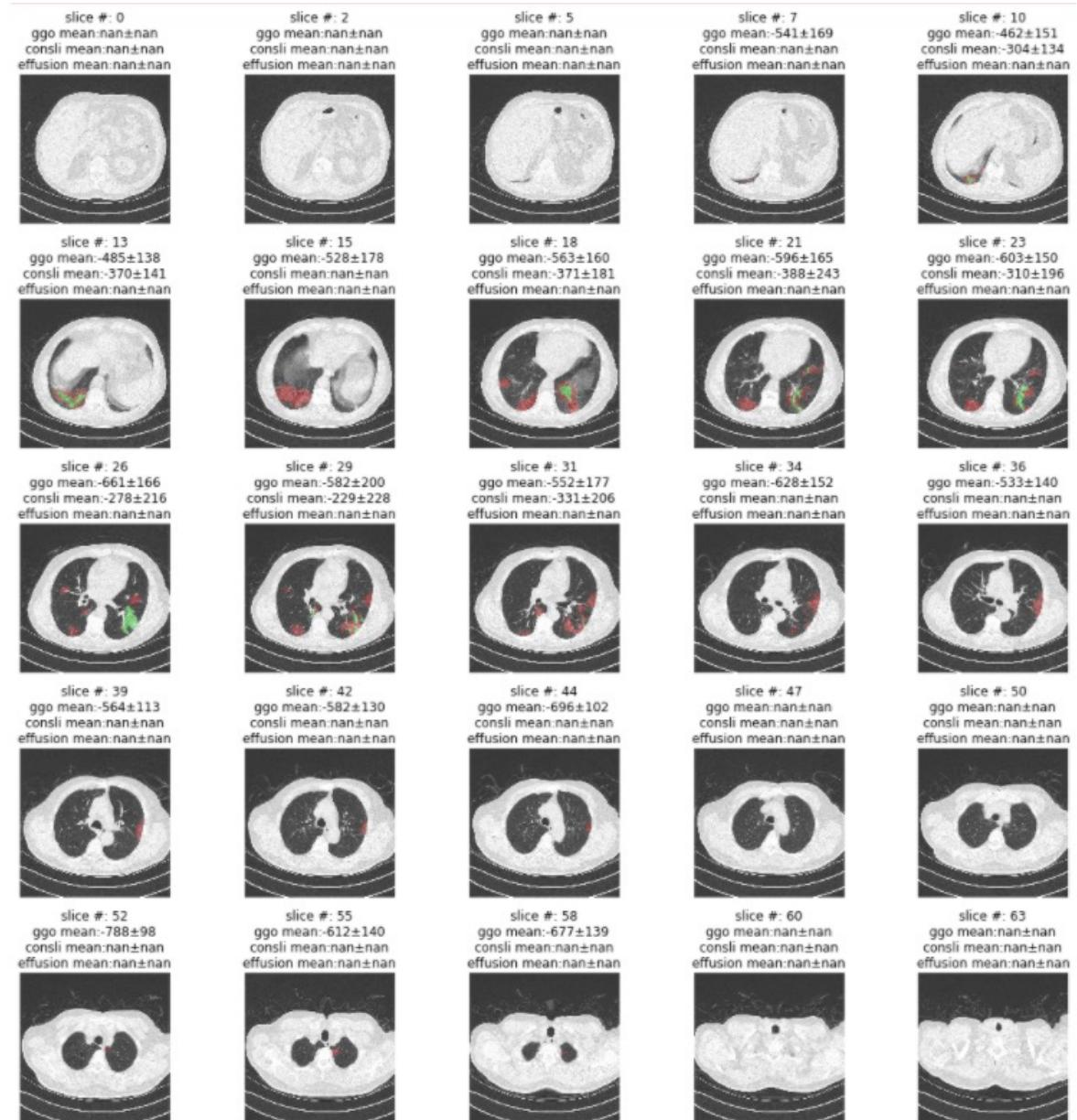
赤色感が増す

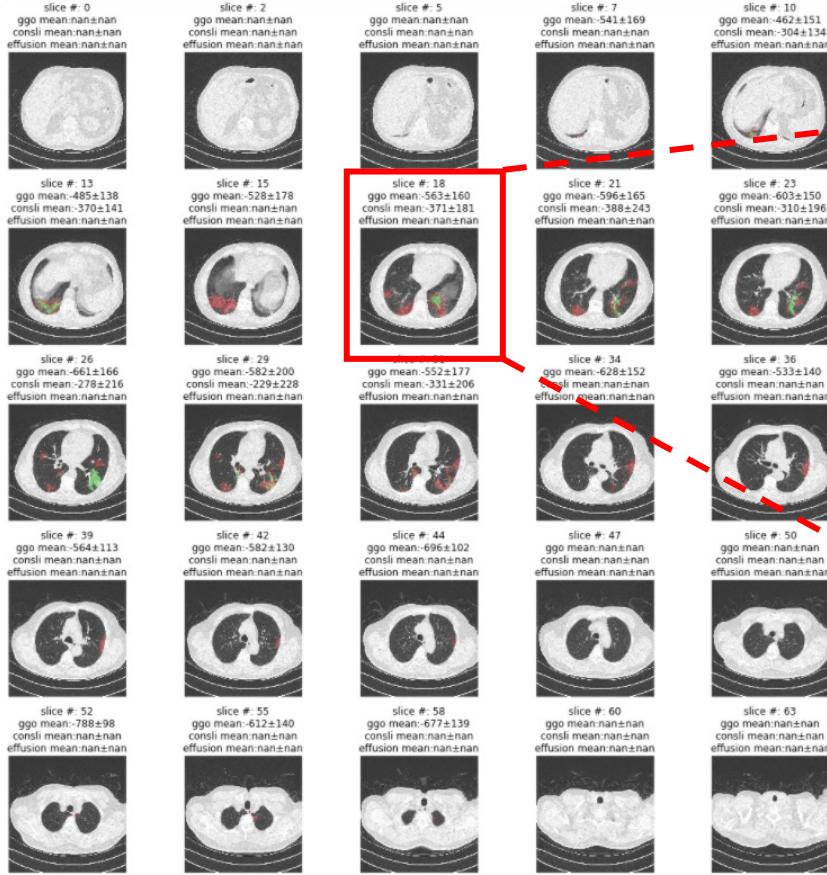


Docker + AWSを使いたい人はこちらの講座を参考にしてください。

https://datawokagaku.com/docker_lecture/

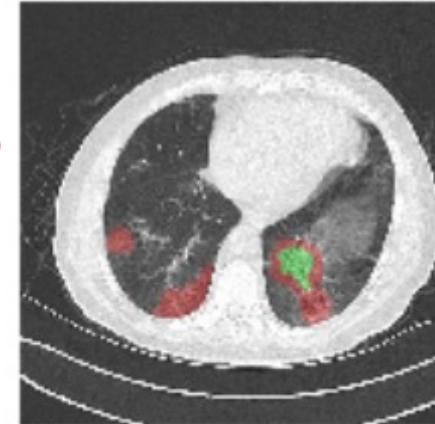
各スライスのアノテーションを一覧で表示する





- ①スライス番号(index)を表示
- ②各ラベルのHUの統計量を表示
- ③軸を削除

① slice #: 18
 ② ggo mean:-563±160
 consli mean:-371±181
 effusion mean:nan±nan



マスクデータ

2	2	3	3
1	1	2	3
1	2	3	0
0	0	0	0

作成

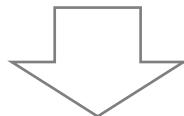
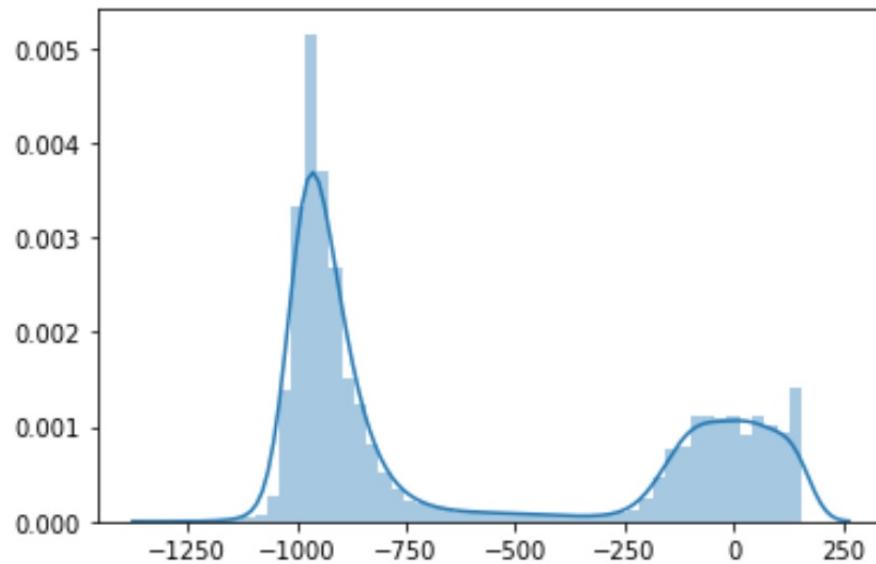
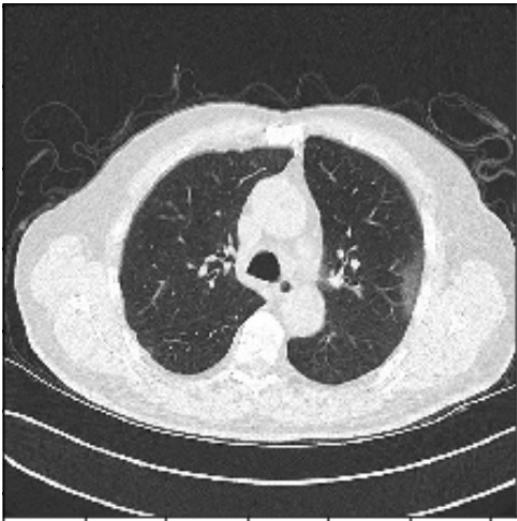
フィルタ

F	F	F	F
T	T	F	F
T	F	F	F
F	F	F	F

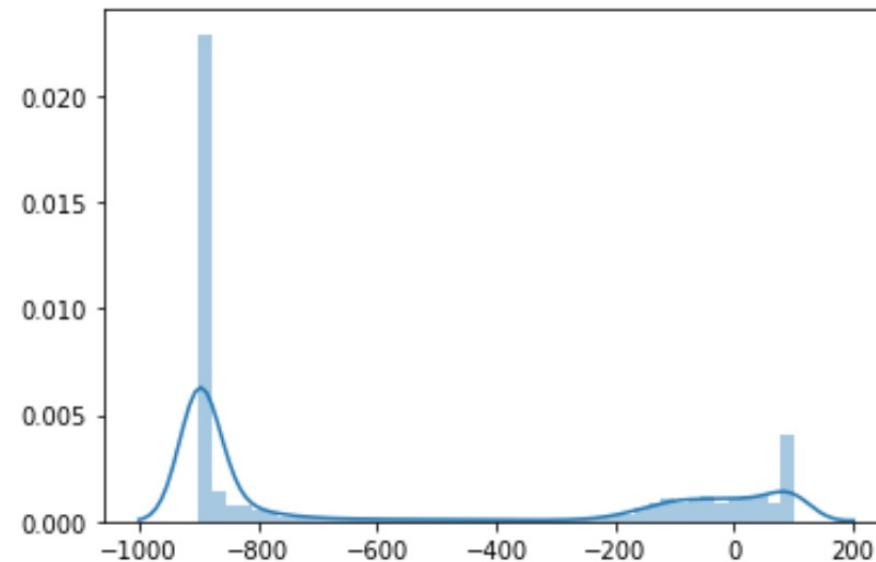
適用

CTデータ

-432	133	231	-123
132	-213	143	432
234	11	232	400
-200	-1000	-900	-400



HUを -900 ~ 100でclip



例) -1300 ~ 300 の値をもつCTデータを、 0 ~ 255に変換する

$$x - \text{最小値} / (\text{最大値} - \text{最小値})$$

① -1300 ~ 300 \rightarrow 0 ~ 1

② 0 ~ 1 $\xrightarrow{\times 255}$ 0 ~ 255