実データの分析、意味の抽出、外れ値の判断

https://www.kkaneko.jp/ai/ae/index.html

金子邦彦

アウトライン

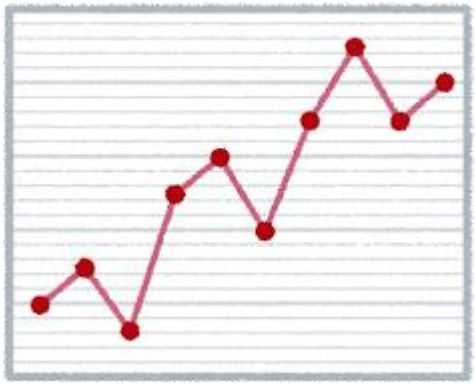
番号	項目
	復習
11-1	Python まとめ
11-2	Python のデータフレーム
11-3	ヒストグラム
11-4	クラスタリング
11-5	外れ値
11-6	演習

各自，資料を読み返したり，課題に取り組んだりも行う

データサイエンス

- **データの正しい取り扱いと活用**
- 統計，数学を基礎とする

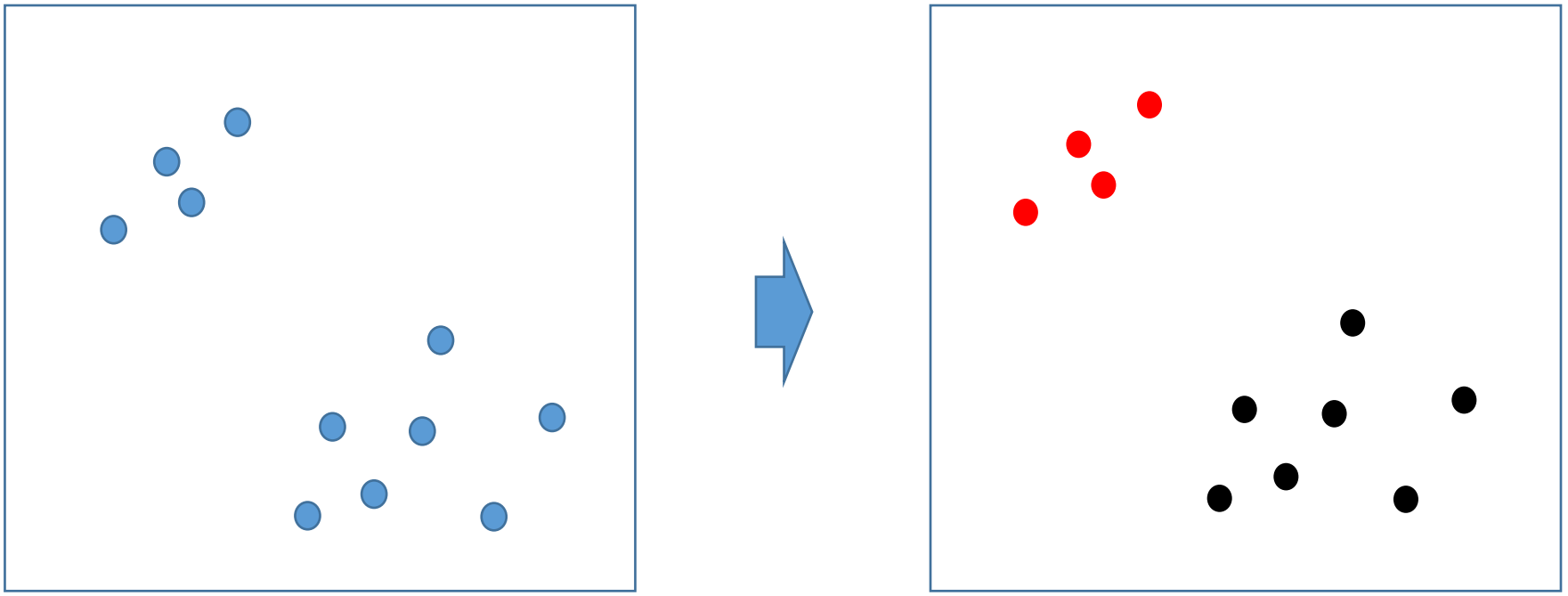
データサイエンスの用途はさまざま
　クラスタリング
　予測
　など



クラスタリング

訓練データ

近くにあるデータを
１つにまとめる

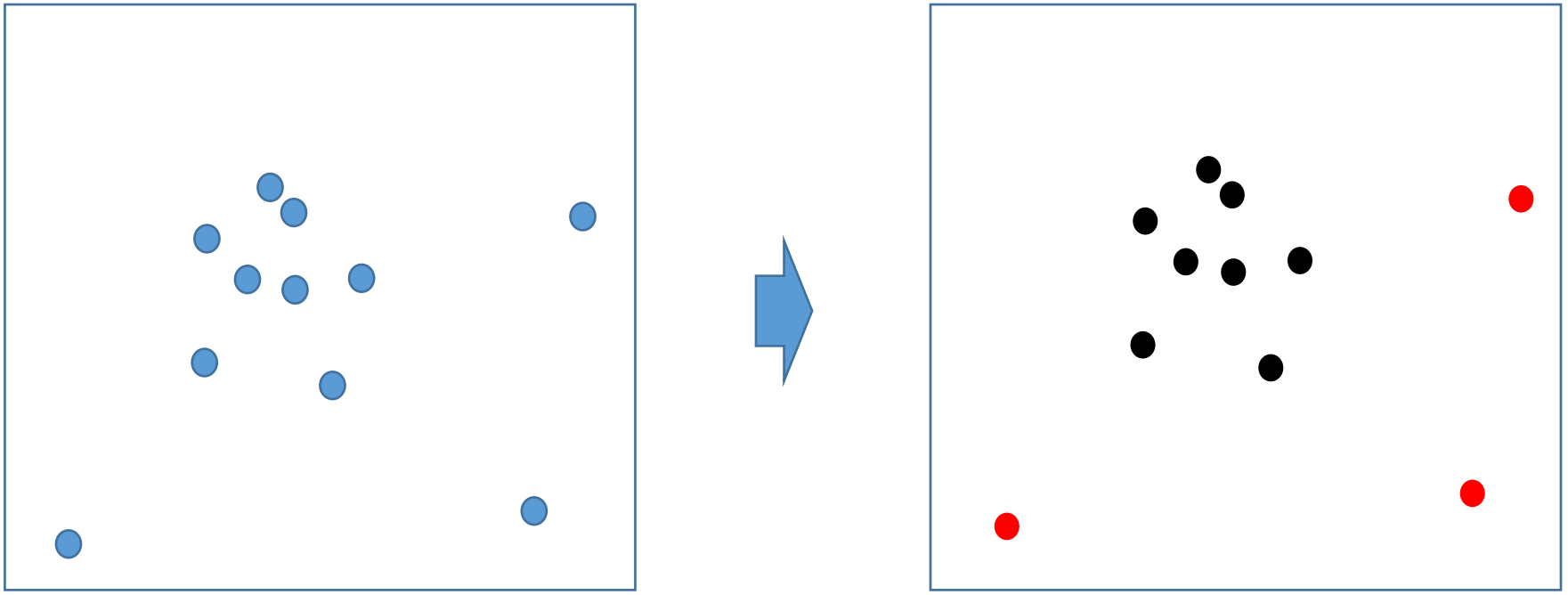


クラスタリング：データの密集を見ることによる分析

分類

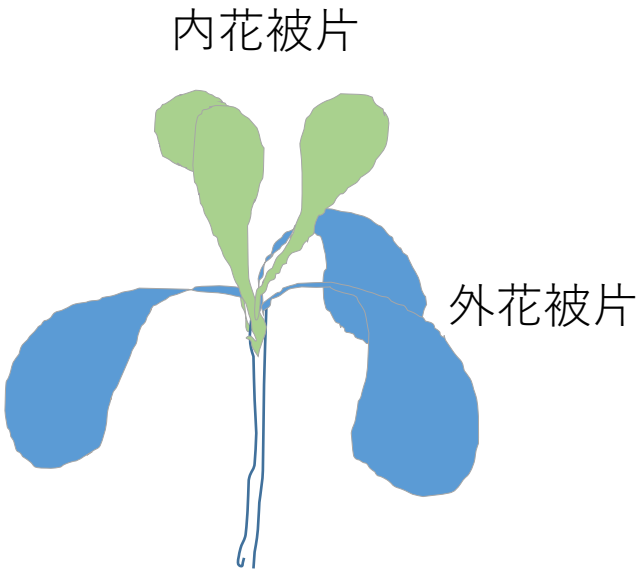
訓練データ

集まっているものを正常と考え，それ以外は異常



分類（異常と正常の分類など）

アヤメ属 (Iris)



- 多年草
- 世界に 150種. 日本に 9種.
- 花被片は 6個
- **外花被片**（がいかひへん） Sepal
  3個（大型で下に垂れる）
- **内花被片**（ないかひへん） Petal
  3個（直立する）

Iris データセット

Iris データセット（データ数は 50 × 3）
のうち、先頭 10 行

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa

外花被片(**Sepal**)の長さと幅　**内花被片**(**Petal**)の長さと幅　**種類**

特徴量　**ラベル**

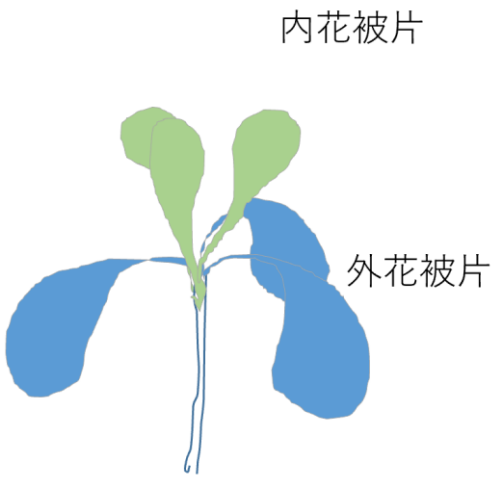
- **3種**のアヤメの**外花被辺**、**内花被片**を計測
- 種類も記録
  - **setosa**
  - **versicolor**
  - **virginica**
- データ数は **50** × **3**

作成者：Ronald Fisher

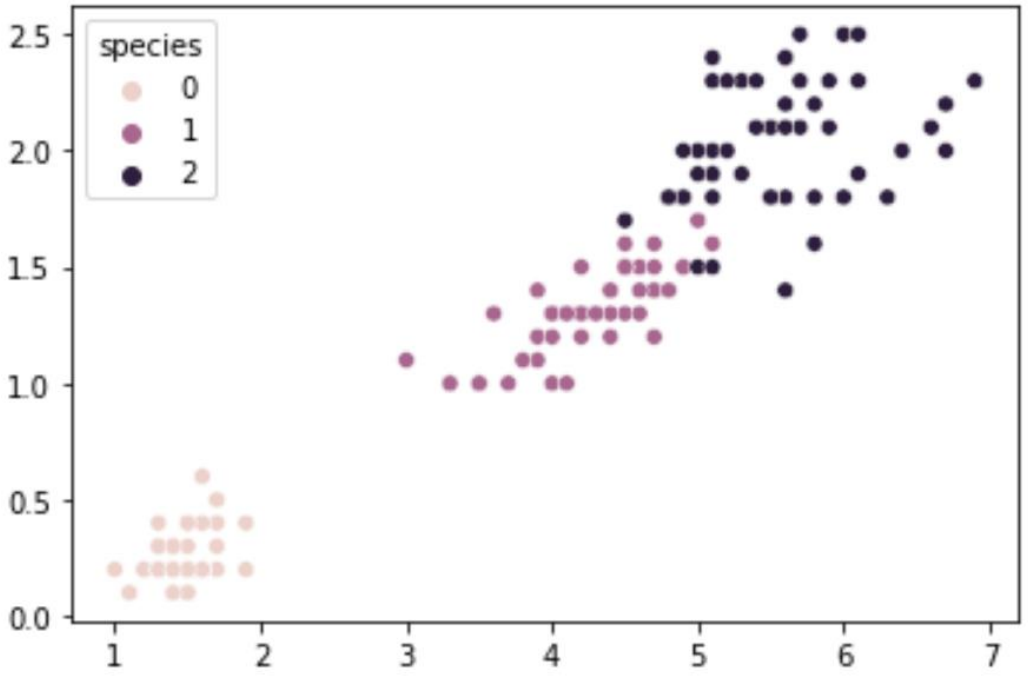
作成年：1936

Iris データセットの散布図

アヤメ属 (Iris)



縦軸：内花被片の**幅**



横軸：内花被片の**長さ**

次の**3種類**の分類済みのデータ

0: **setosa**
1: **versicolor**
2: **virginica**

Iris データセットと配列（アレイ）

Iris データセット

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
```

特徴量（数値）
サイズ：**150 × 4**

```
[0
0
0
0
0
0
0
0
0
0
```

ラベル（数値）
サイズ：**150**

setosa → 0
versicolor → 1
virginia → 2

ラベルの数値化

Iris データセットは Python でも利用可能

ソースコード

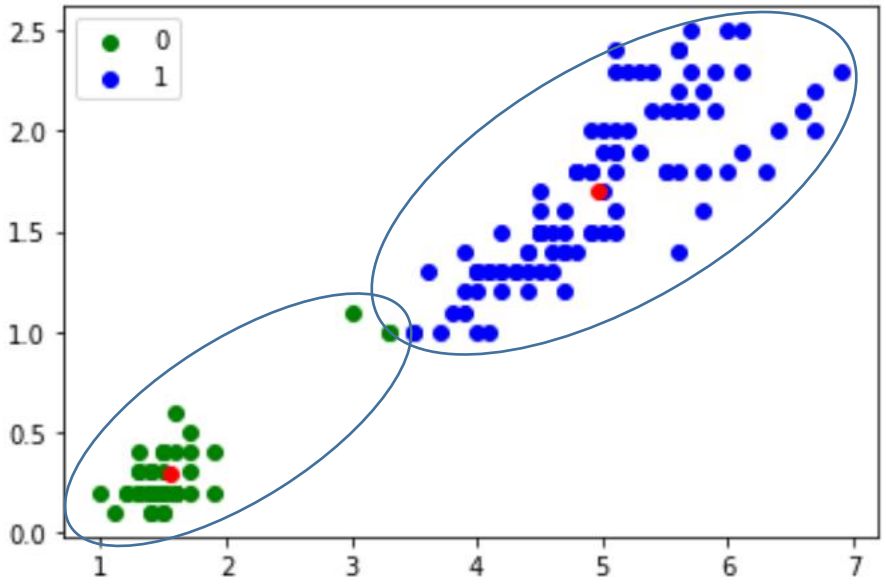
```
import matplotlib.pyplot as plt
import seaborn

iris = seaborn.load_dataset('iris')
print(iris)
```

```
import matplotlib.pyplot as plt
import seaborn

iris = seaborn.load_dataset('iris')
print(iris)
```

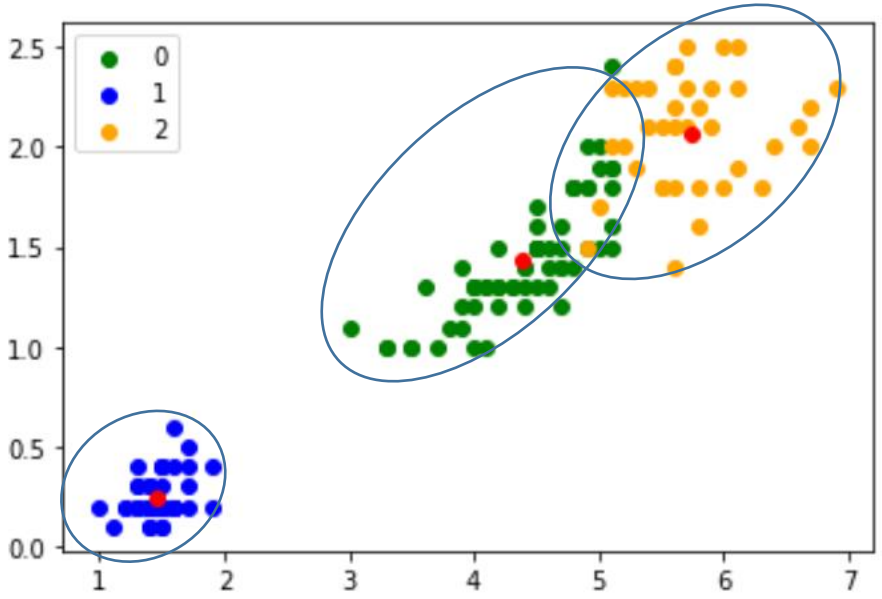
```
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

Iris データセットのクラスタリング

内花被片，外花被片の長さと幅で，クラスタリングを行う．



クラスタ数 = 2 に設定



クラスタ数 = 3 に設定

11-1. Python まとめ

Python

- プログラミング言語
- 「**入門者に学習しやすい**」とされる
- **多数の拡張機能**（外部プログラムのインポートによる）

```
x = 100
if (x > 20):
    print("big")
else:
    print("small")
s = 0
for i in [1, 2, 3, 4, 5]:
    s = s + i
print(s)
```

ソースコード

```
kaneko@www:/tmp$ python foo.py
big
15
```

実行結果

オブジェクト，変数，メソッド，代入，変数

- **オブジェクト**：コンピュータでの**操作や処理の対象となるもの**のこと
- **変数**：**名前の付いたオブジェクト**には，**変数**，**関数**などがある（「変数」は，数学の変数とは違う意味）

- **メソッド: オブジェクト**に属する操作や処理．**メソッド**呼び出しでは，**引数**を指定することがある．**引数**（ひきすう）は，**メソッド**に渡す値のこと

```
Hero.attack("fence", 36, 26)
```

- **代入**：「=」を使用．オブジェクトの**値が変化**する

```
b = a + 100
```

オブジェクトとメソッド

hero.moveDown()　Python プログラム

hero　　**オブジェクト**
moveDown()　**メソッド**
間を「.」で**区切って**いる

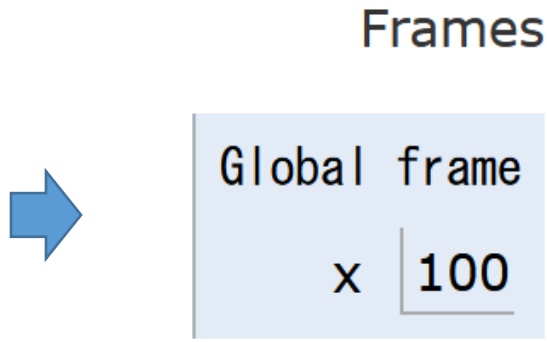
- **メソッド: オブジェクト**に属する操作や処理.
- **メソッド**呼び出しでは，**引数**を指定することがある．**引数**（ひきすう）は，**メソッド**に渡す値のこと

```
Hero.attack("fence", 36, 26)
```

代入

- **代入**：プログラムで，「**x = 100**」のように書くと，**x の値が 100 に変化**する

```
x = 100
```



プログラム　　**実行結果**

メソッドアクセス，代入

Python プログラムの例

```
x = 100
a = x + 200
enermy1 = hero.findNearestEnemy()
hero.attack(enemy1)
```

- **代入**：**オブジェクト名** ＋ 「**=**」
  ＋ 式または値またはメソッド呼び出し
- **メソッドアクセス**：**オブジェクト名** ＋ 「**.**」
  ＋ **メソッド名** ＋「**()**」 （引数を付けることも）

Python プログラムでは，その他にも，属性アクセス，関数呼び出し，制御，「*」，「+」などの演算子，コマンド，定義など

11-2. Python のデータフレーム

Python のデータフレーム

表形式のデータ

	x	y
0	1	4
1	1	2
2	1	5
3	2	4
4	3	5
5	3	3

データ本体

Python のデータフレームを組み立てるプログラム

	x	y
0	1	4
1	1	2
2	1	5
3	2	4
4	3	5
5	3	3

データフレーム

```
import pandas as pd

df = pd.DataFrame([[1, 4], [1, 2], [1, 5], [2, 4], [3, 5], [3, 3]], columns=['x', 'y'])
print(df)
```

```
   x  y
0  1  4
1  1  2
2  1  5
3  2  4
4  3  5
5  3  3
```

データフレームの作成と確認表示

データフレームの組み立て，結果は **df** に代入

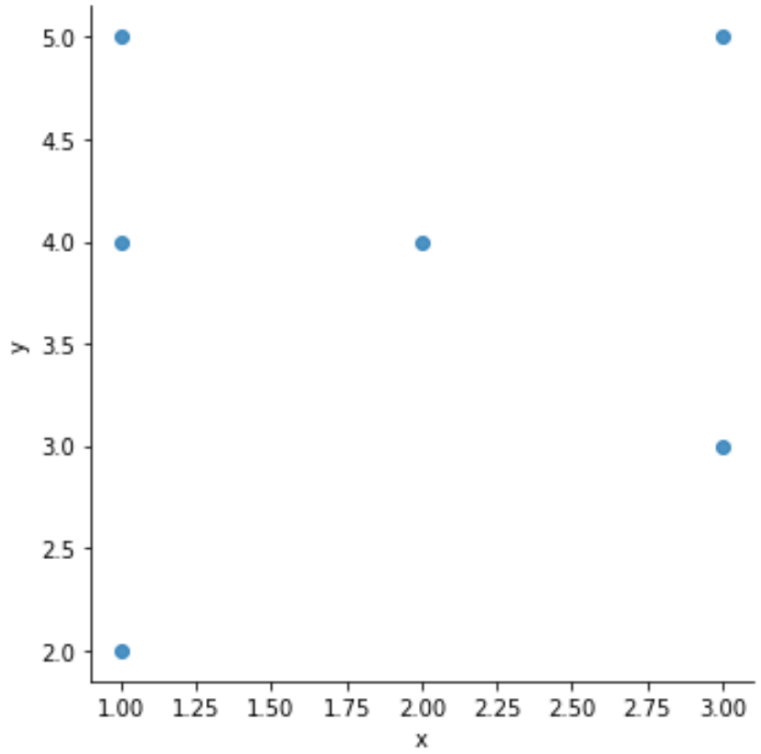
```
df = pd.DataFrame([[1, 4], [1, 2], [1, 5], [2, 4], [3, 5], [3, 3]], columns=['x', 'y'])
```

データフレームから散布図を作成するプログラム

「data=**df**」では，散布図を作成したいデータフレームのオブジェクト名が df であることを指定

```
import matplotlib.pyplot as plt
import seaborn

seaborn.lmplot(x='x', y='y', data=df, fit_reg=False)
```

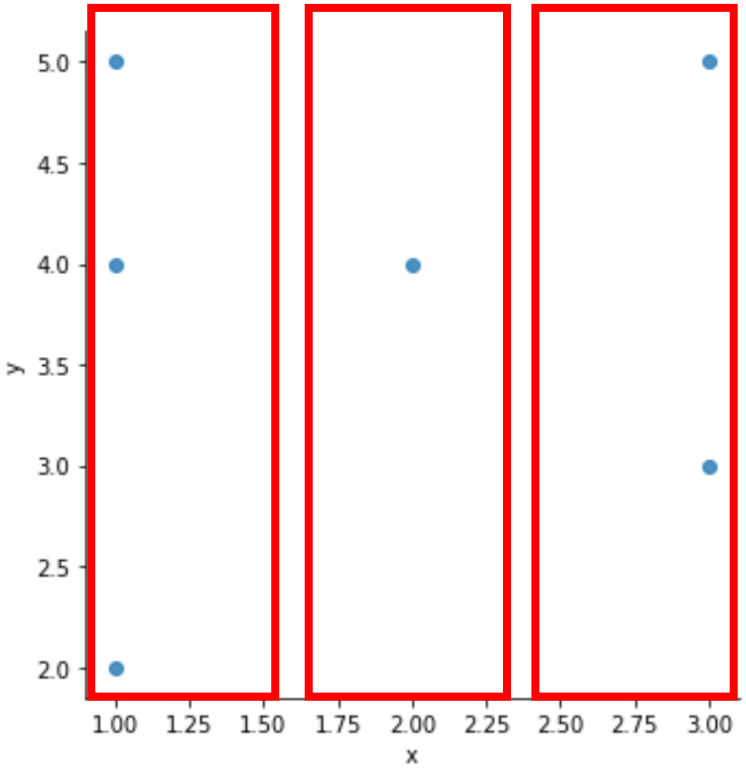
```
<seaborn.axisgrid.FacetGrid at 0x7f8299ae7be0>
```



11-3. ヒストグラム

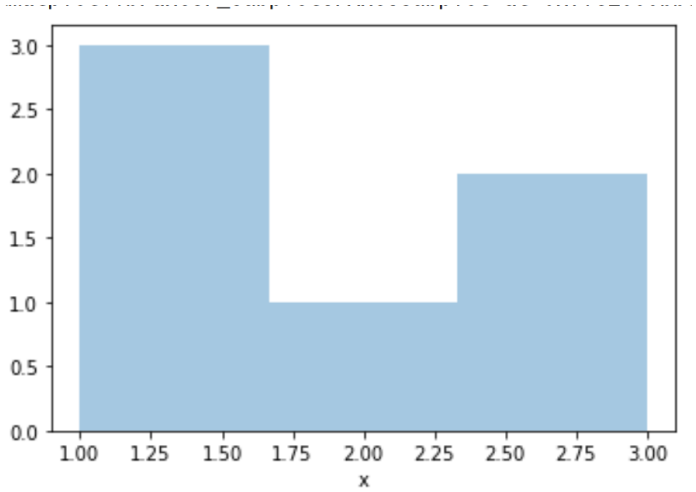
ヒストグラム①

- **ヒストグラム**は，データ全体を同じ大きさで分割し，数え上げを行う．データ全体の分布をみる



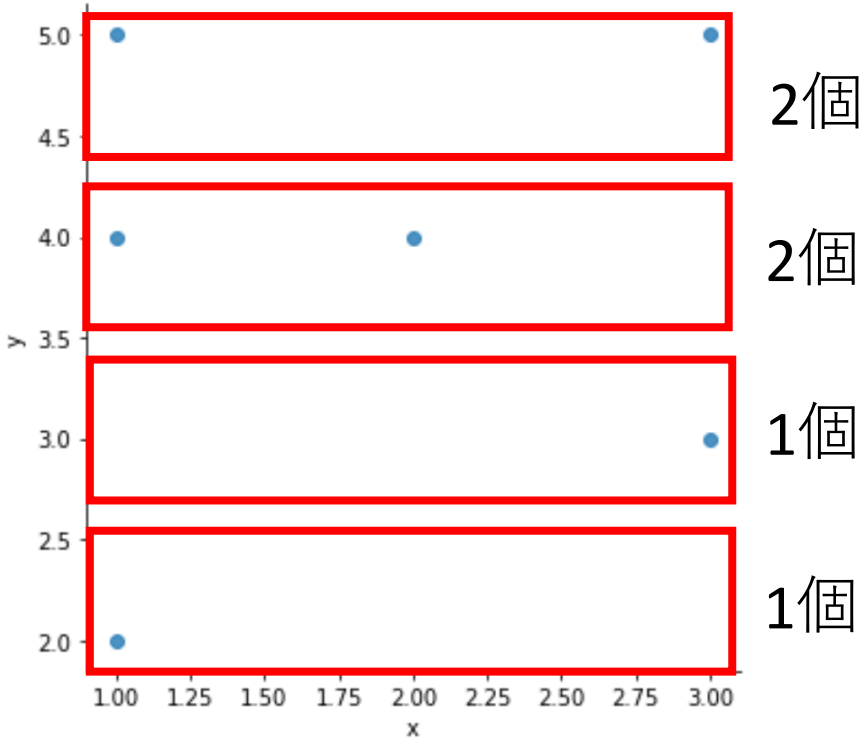
3個　　1個　　2個

元データ

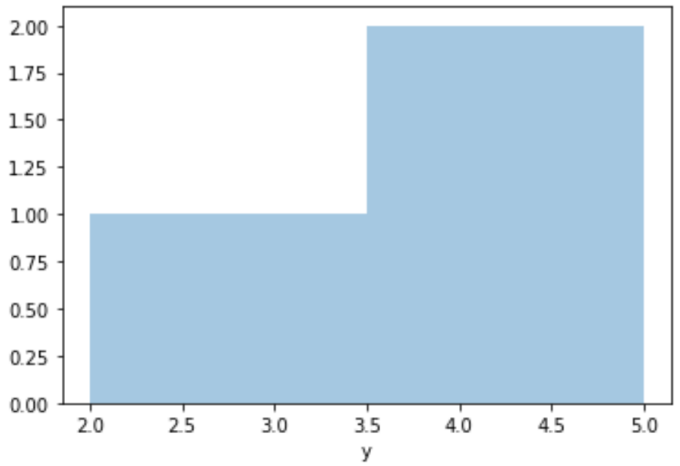


ヒストグラム
（帯の数は３，
基準は **x**）

ヒストグラム②

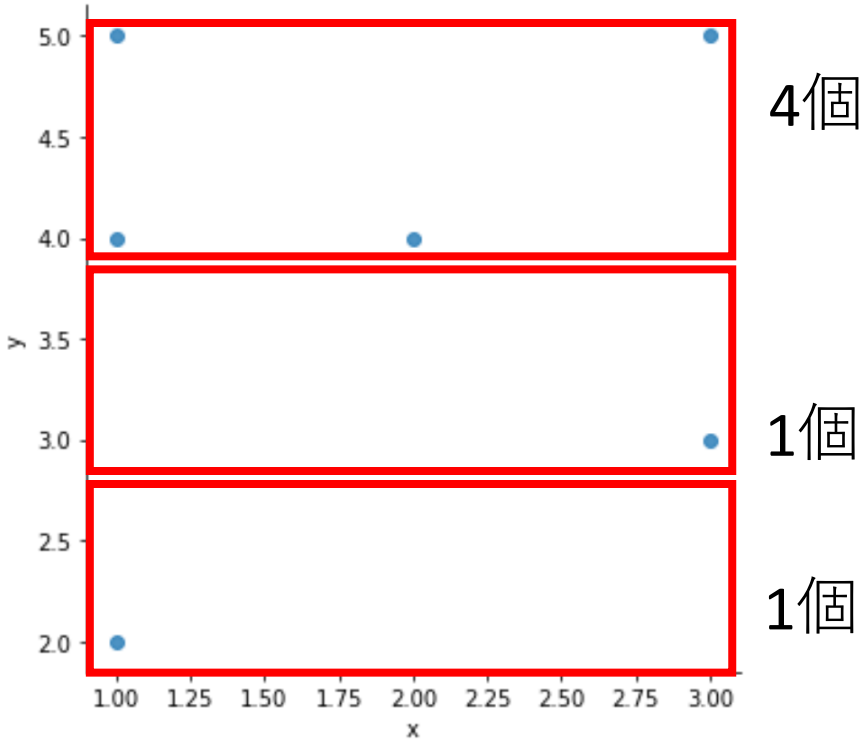


元データ



ヒストグラム
（帯の数は**4**,
基準は **y**）

ヒストグラム②

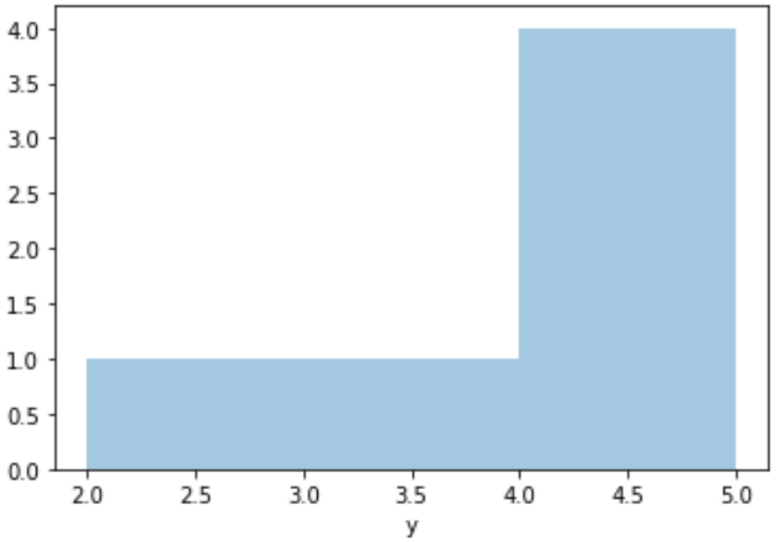


4個

1個

1個

元データ



ヒストグラム
（帯の数は**4**,
基準は **y**）

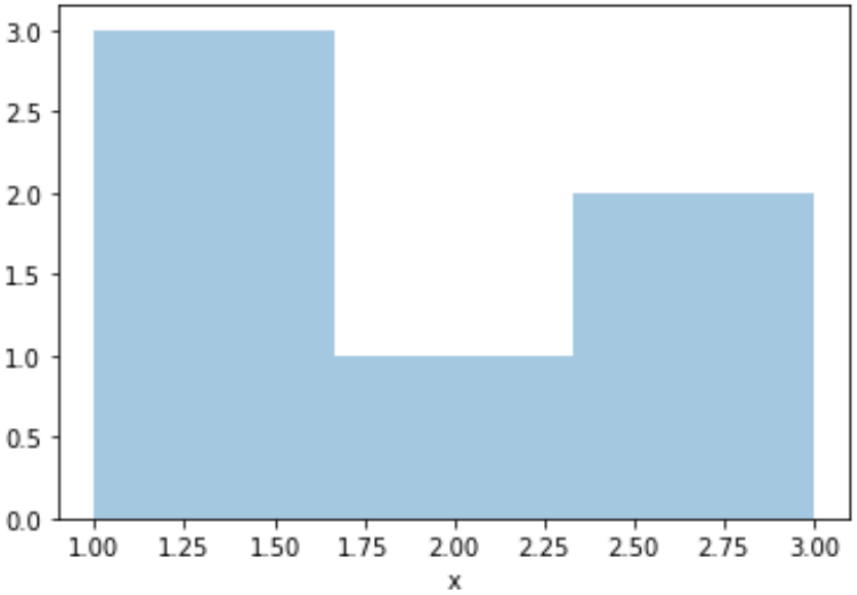
ヒストグラム①を得る Python プログラム

- 帯の数: **3**
- 基準: **x**

x 3

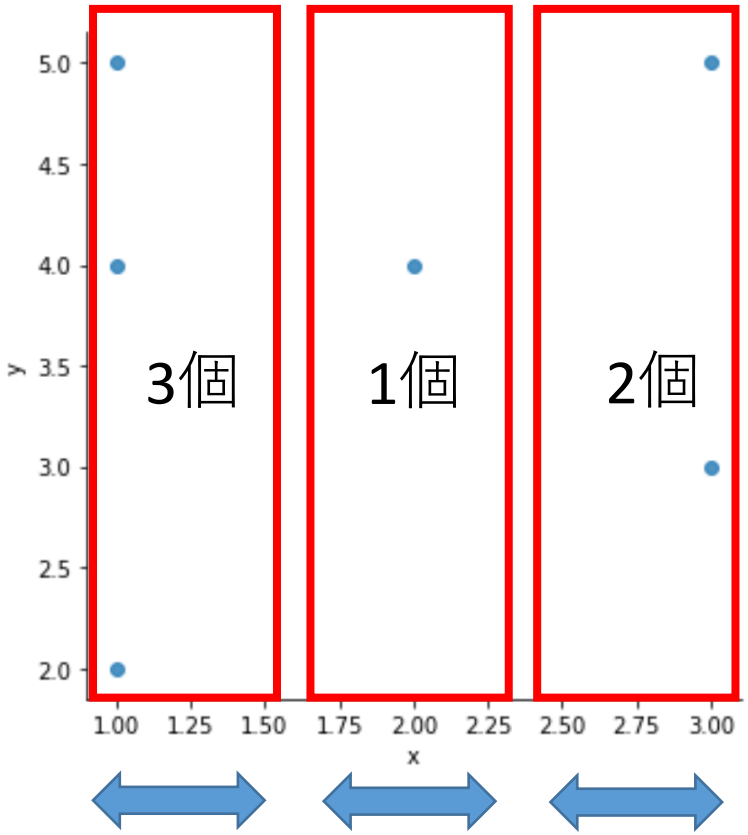
```
import matplotlib.pyplot as plt
import seaborn

seaborn.distplot(df['x'], bins=3, kde=False)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributic
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f82999bb310>
```

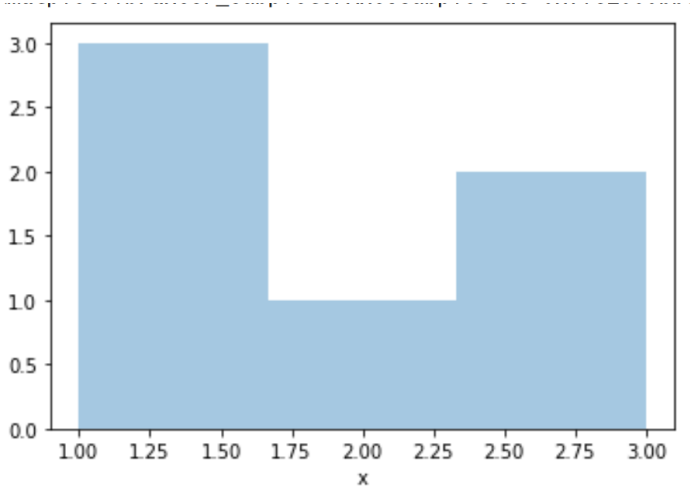


まとめ

- **ヒストグラム**は，データ全体を同じ大きさで分割し，数え上げを行う．データ全体の分布をみる

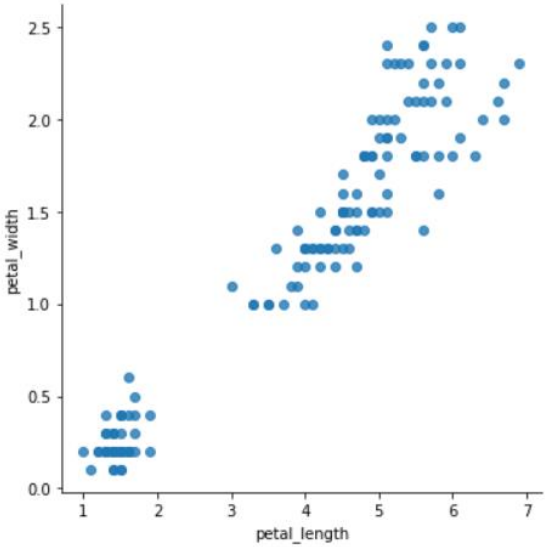


同じ大きさで分割し、数え上げ

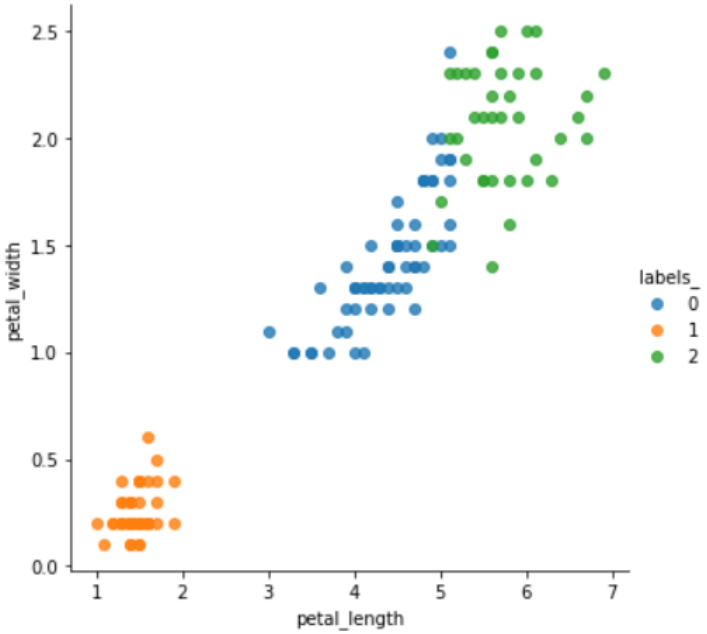


11-4. クラスタリング

クラスタリング



クラスタリング



- **クラスタリング**は，**近いデータをまとめて**，１グループ（＝クラスタという）にする

クラスタリングの例①

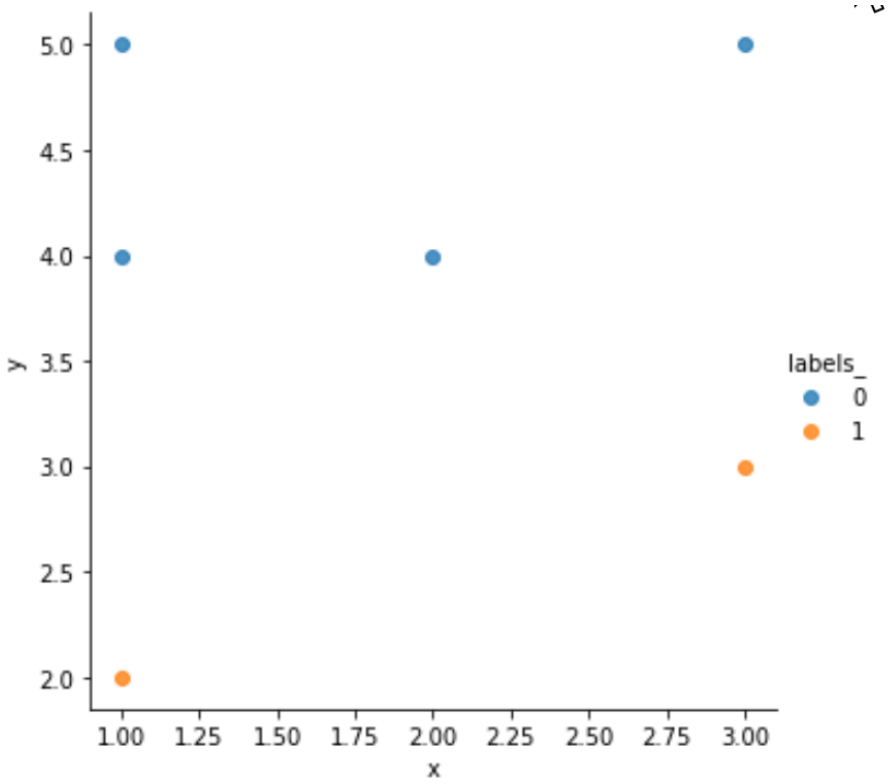
クラスタ数を **2** に設定

	x	y
0	1	4
1	1	2
2	1	5
3	2	4
4	3	5
5	3	3

元データ

1
0
1
1
1
0

クラスタリング結果は番号



クラスタリング結果を色付きの散布図でプロット

クラスタリングの例②

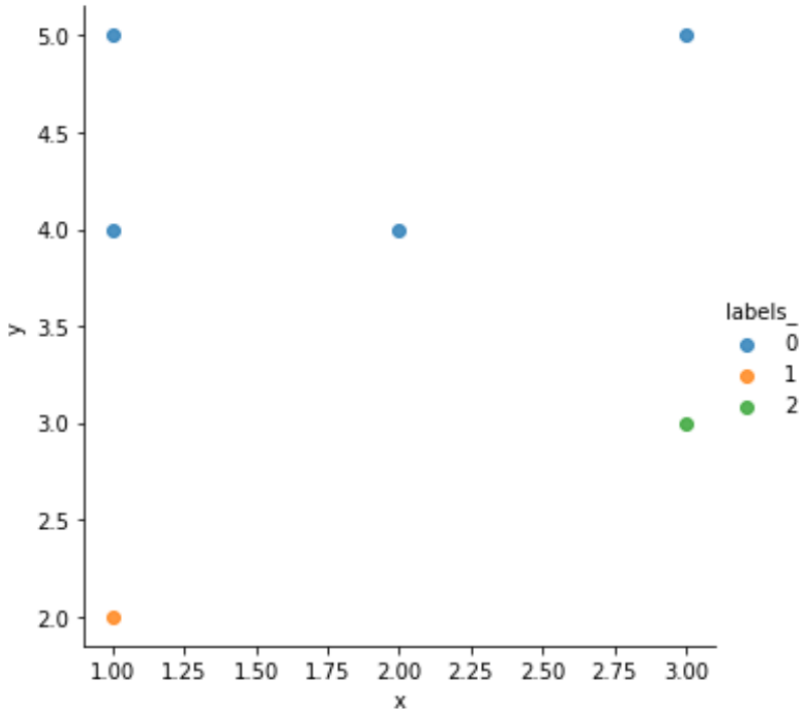
クラスタ数を **3** に設定

	x	y
0	1	4
1	1	2
2	1	5
3	2	4
4	3	5
5	3	3

元データ

0
1
0
0
0
2

クラスタリング結果は番号



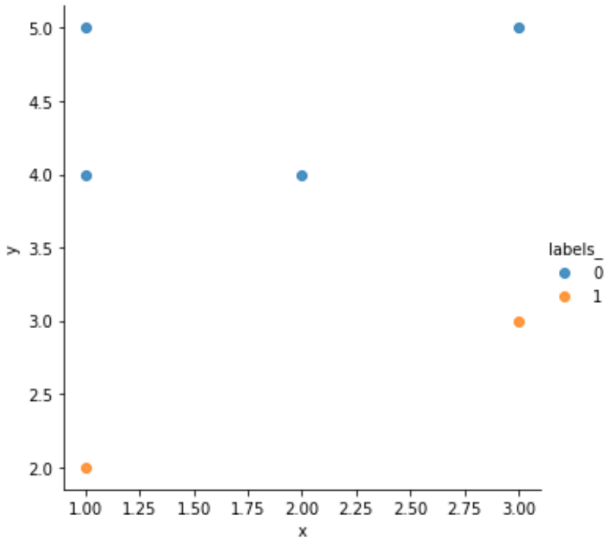
クラスタリング結果を色付きの散布図でプロット

クラスタリング①を行う Python プログラム

- クラスタ数: **2**

2

```
import matplotlib.pyplot as plt
import seaborn
import sklearn
import sklearn.cluster

estimator = sklearn.cluster.KMeans(n_clusters = 2)
a = df
estimator.fit(a)
a['labels_']=estimator.labels_

print(a.labels_)
seaborn.lmplot(x='x', y='y', data=a, hue='labels_', fit_reg=False)
```



クラスタリングでは，乱数が使用されるため，
実行のたびに違った値になる可能性がある

まとめ

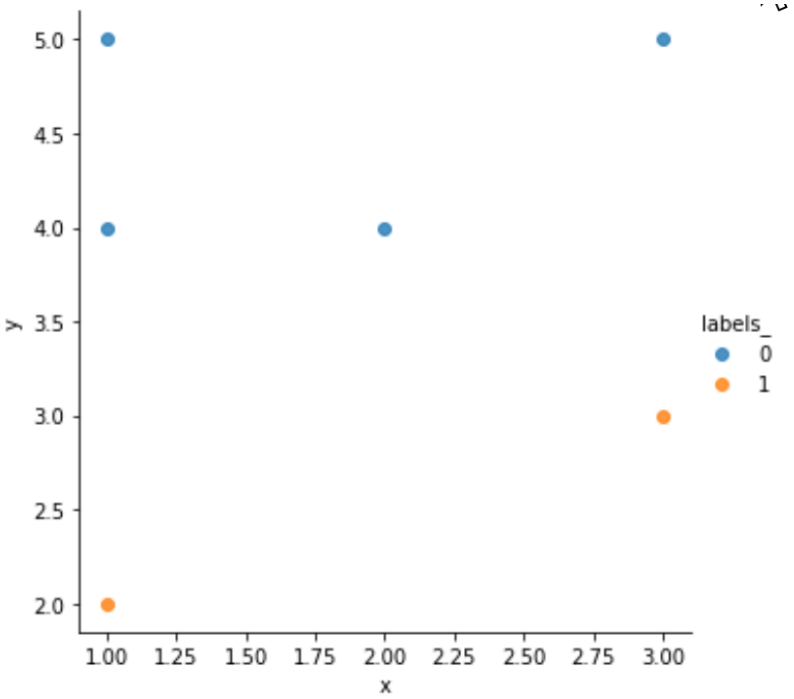
クラスタリングは，**近いデータをまとめて**，１グループにする．クラスタ数を設定可能

	x	y
0	1	4
1	1	2
2	1	5
3	2	4
4	3	5
5	3	3

元データ

1
0
1
1
1
0

クラスタリング結果は番号



クラスタリング結果を色付きの散布図でプロット

11-5. 外れ値

平均

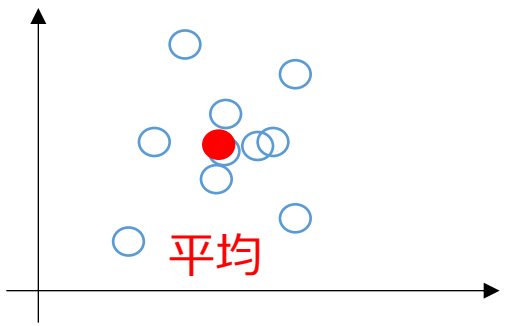
- **平均**の基本，**合計**して，**データの個数で割る**

  10, 40, 30, 40 の**平均**: **120** ÷ **4** で **30**

- **複数の値の組**の**平均**を考えることもある

  (10, 5), (40, 10), (30, 5), (40, 20) の平均:

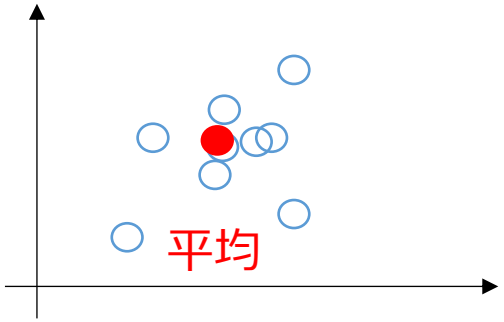
  合計は 120 と 40．4で割って (30, 10)

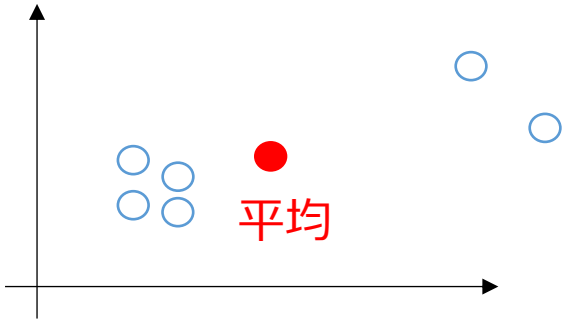


平均は，**データ集合**の**代表**とみることができる場合がある

計測に**誤差**があるとき，
複数の計測を繰り返し，**平均**をとることで，**誤差を軽減**できることも

平均を使うときの注意点





このような平均に，
意味があるでしょうか？

データの分布によっては，**平均**では**役に立たない**こともある．
（平均は万能ではない）

クラスタリングでの各クラスタの中心

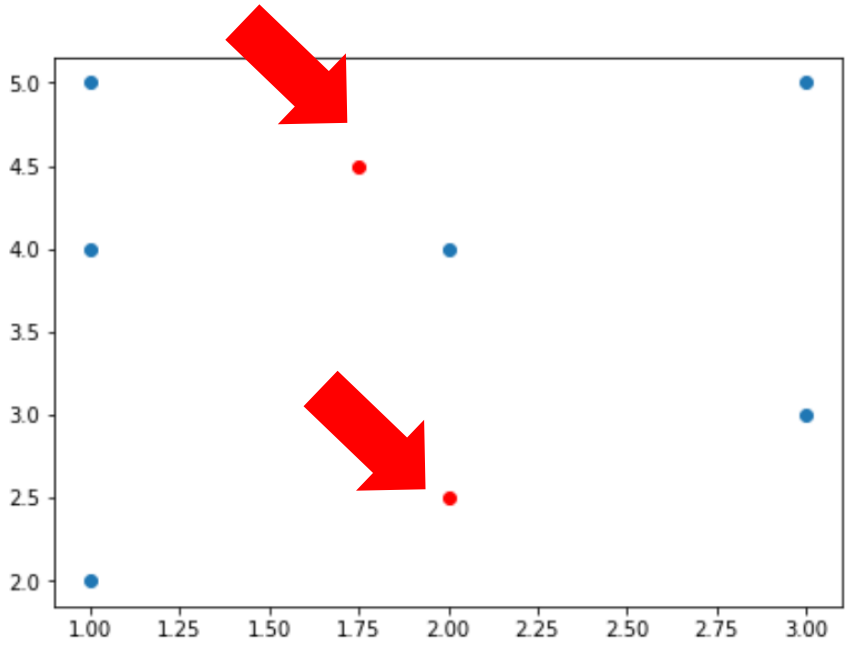
- クラスタ数: **2**

	x	y
0	1	4
1	1	2
2	1	5
3	2	4
4	3	5
5	3	3

元データ

1
0
1
1
1
0

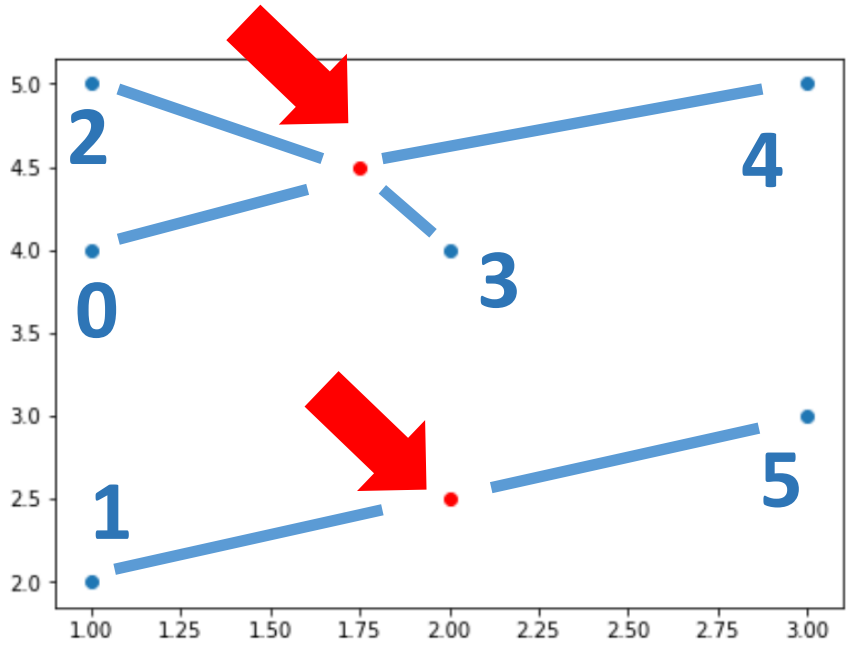
クラスタリング結果は番号



２つのクラスタの中心

各クラスタの中心からの距離

- 元データが 6個のとき，**距離**は **6個算出**される



２つのクラスタの中心からの距離

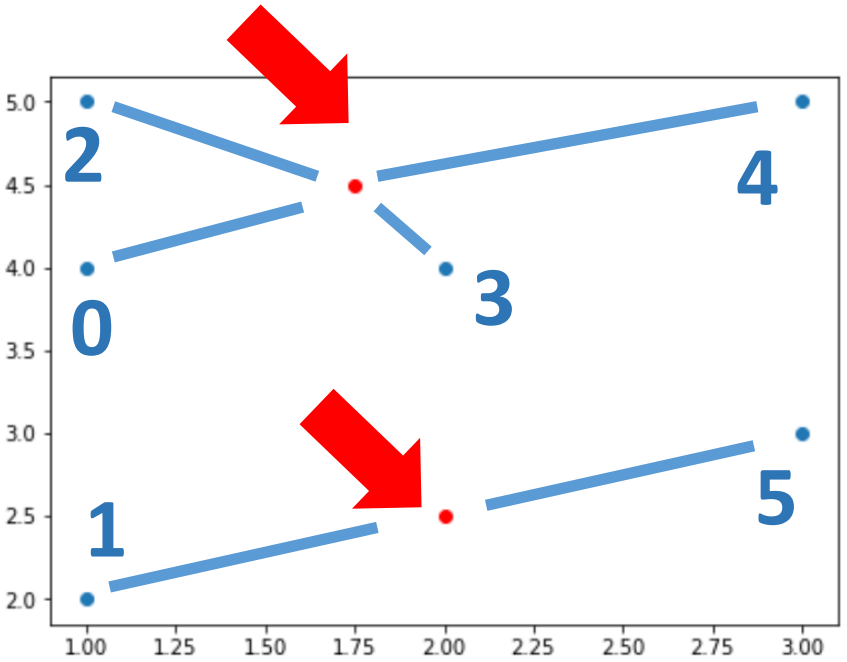
	x	y	labels_	cx	cy	distance
0	1	4	0	1.75	4.5	0.901388
1	1	2	1	2.00	2.5	1.118034
2	1	5	0	1.75	4.5	0.901388
3	2	4	0	1.75	4.5	0.559017
4	3	5	0	1.75	4.5	1.346291
5	3	3	1	2.00	2.5	1.118034

算出された距離

- **距離が極端に大きいものは，外れ値と判断できる**

まとめ

クラスタの中心からの遠近が，外れ値であるかの判断に使える場合がある



２つのクラスタの中心からの距離

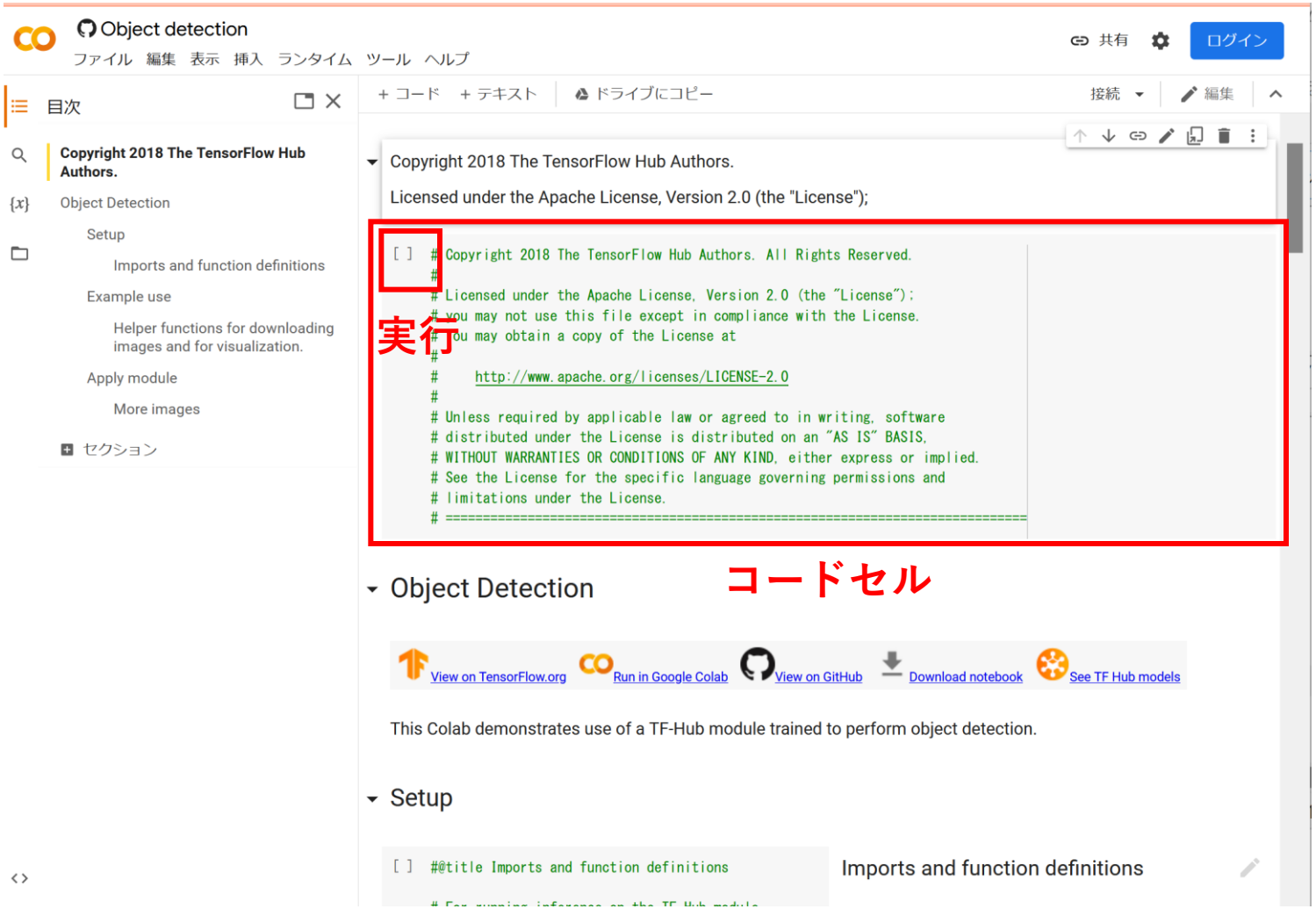
11-6. 演習

今回の演習の特徴

- Python で**自動化**する
- Excel よりも、**きれいなグラフ**をめざす
- Excel では難しいこと（クラスタリング，外れ値の判断）も行う
- **Python の簡単なプログラム**で行う

  書き換えて違う結果を得る，考察することにもチャレンジ

Google Colaboratory の使い方概要 ①



Google Colaboratory ノートブック
コードセルの再実行や変更には,
Google アカウントでのログインが必要

Google Colaboratory の使い方概要 ②

実行

```
[ ] files = ['a.png', 'b.png','c.png', '126.png', '127.png']
```

6．顔検出

顔検出は，**写真やビデオの中の顔を検出すること**．顔とそれ以外のオブジェクトを区別することも行う．顔検出の結果は，バウンディングボックスで得られるのが普通である．

次のプログラムは，Dlib を用いて，画像からの顔検出を行う。

- 「**dets = cnn_face_detector(img, 6)**」・・・**顔検出**の実行
- 「**cv2.rectangle(disp, (d.rect.left(), d.rect.top()), (d.rect.right(), d.rect.bottom()), (255, 0, 0), 1)**」・・・**顔検出の結果**を**四角形で表示**

結果は，赤い四角で表示される．１，３，４，５番目の画像（a.png, c.png, 126.png, 127.png）からは，顔が検出される．２番目の画像（b.png, 手で顔を覆い隠したもの）からは顔が検出されない．**少し隠れていたり，顔が傾いていても顔検出ができる**が，**大きく隠れていると顔検出できない**．

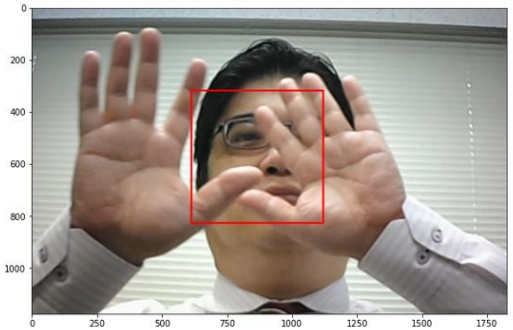
実行結果が長いので，スクロールして**全体を確認**すること．

謝辞: ここの Python プログラムは，Dlib に付属の cnn_face_detector.py を書き換えて使用している

実行

```
import sys
import dlib
import os
import urllib.request
import cv2
import matplotlib.pyplot as plt

cnn_face_detector = dlib.cnn_face_detection_model_v1('mmod_human_face_detector.dat')

for f in files:
    print("*** file: {} ***".format(f))
    img = dlib.load_rgb_image(f)
    dets = cnn_face_detector(img, 1)
    print("Number of faces detected: {}".format(len(dets)))
    for i, d in enumerate(dets):
        print("Detection {}: Left: {} Top: {} Right: {} Bottom: {} Confidence: {}".format(
            i, d.rect.left(), d.rect.top(), d.rect.right(), d.rect.bottom(), d.confidence))
    disp = img.copy()
    for i, d in enumerate(dets):
        cv2.rectangle(disp, (d.rect.left(), d.rect.top()), (d.rect.right(), d.rect.bottom()), (255, 0, 0), 8)
    plt.figure(figsize=(10,10))
    plt.imshow(disp)
    plt.show()
```

```
*** file: a.png ***
Number of faces detected: 1
Detection 0: Left: 614 Top: 319 Right: 1121 Bottom: 827 Confidence: 0.20801100134849548
```




```
*** file: b.png ***
Number of faces detected: 0
```

一番上のコードセルから順々に実行

コードセル

テキストセル

コードセル

- **WEBブラウザでアクセス**
- **コードセル**は **Python プログラム**．
  各自の Google アカウントでログインすれば，変更，再実行可能

Google アカウント

Google アカウントの取得が必要

- 次のページを使用

https://accounts.google.com/SignUp

- 次の情報を登録する

氏名

自分が希望するメールアドレス

＜ユーザー名＞ @gmail.com

パスワード

生年月日，性別

演習

1. **使用するページ:**

https://colab.research.google.com/drive/1eGdELBNegyEoF43ueYqwa6WDk13_7yuY?usp=sharing

2. **必要な事前知識**

Python の基本，散布図，クラスタリング，クラスタリングの中心からの距離

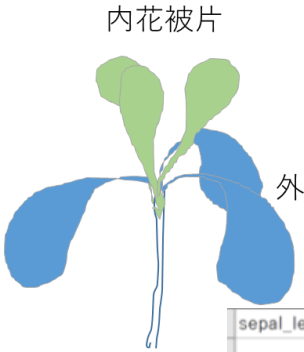
3. **各自で行うこと**

①各自で説明、ソースコード、実行結果を確認する.

②セレッソでの「各自の演習の指示」

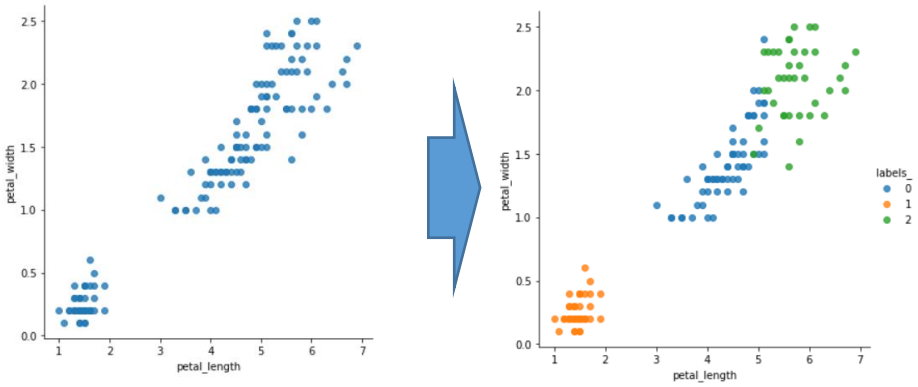
③セレッソの「課題」

全体まとめ

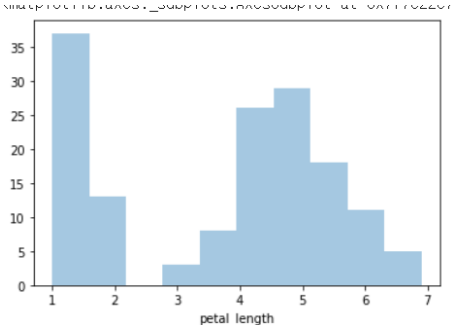


sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa

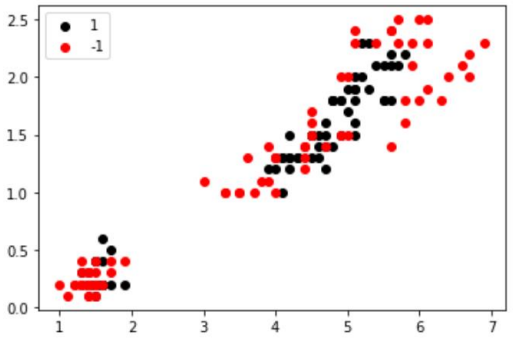
Iris データセット



クラスタリング



ヒストグラム



クラスタリングののち，外れ値を除去