



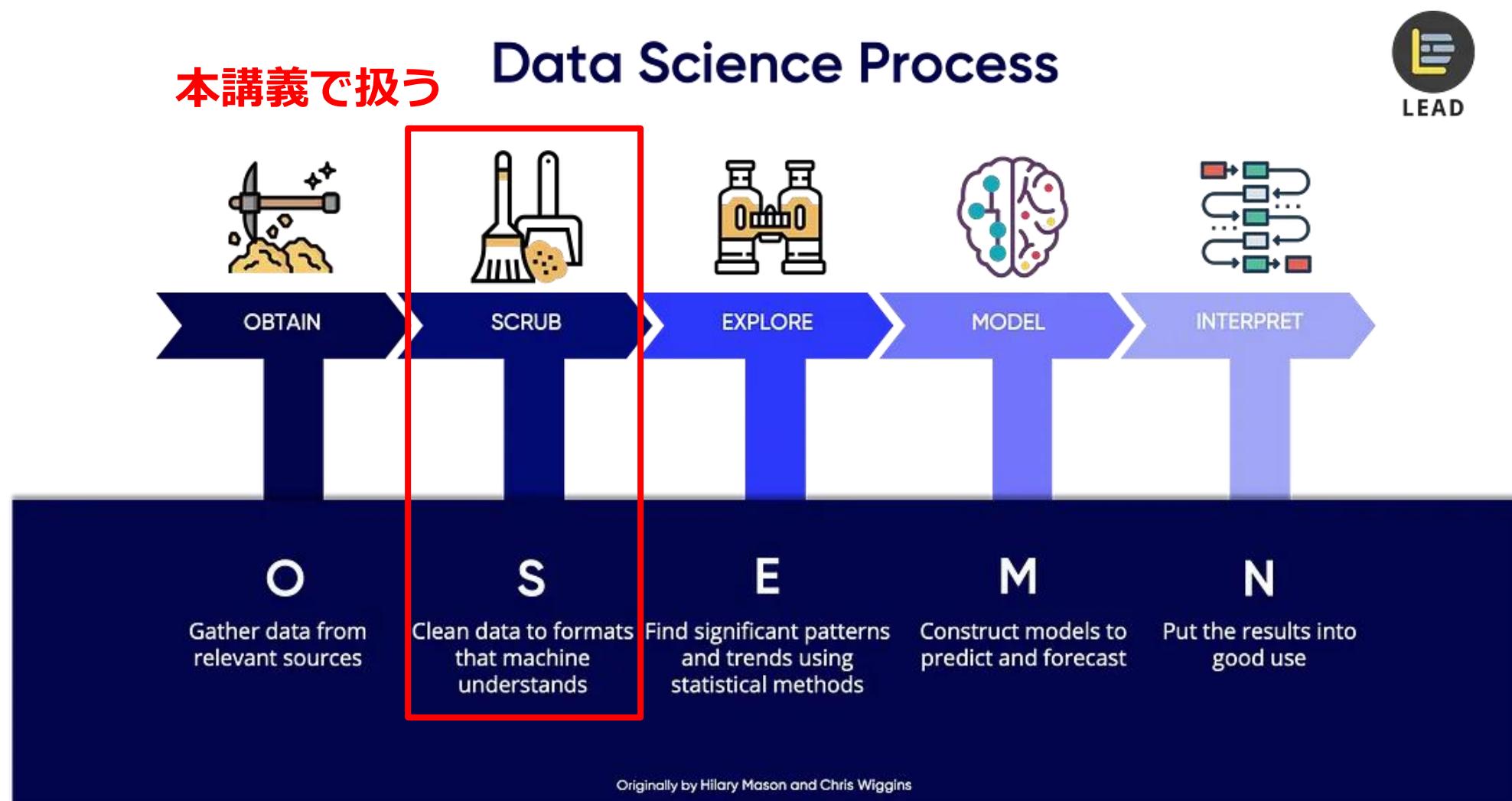
GCI week3 Pythonによるデータ加工処理の基礎 (Pandas)



松尾・岩澤研究室
MATSUO-IWASAWA LAB UTOKYO

講師：鯛 涼太

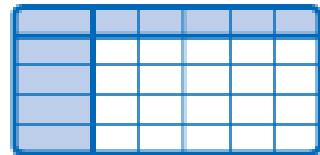
2024/05/14



出典：<https://towardsdatascience.com/5-steps-of-a-data-science-project-lifecycle-26c50372b492>

本講義で扱う

構造化データ



固定長ファイル

Excel / CSV

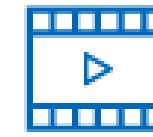
RDB内のデータ

二次元の表形式など

値が数値・記号で

1テーブルに整理されている

非構造化データ



テキスト

音声

画像

動画

センサーログ

半構造化データ

XML JSON html

表形式ではないが
データに規則性がある

データに規則性がなく
表形式に変換できない

出典 : https://solution.toppan.co.jp/digital/contents/cdp_contents07_0831.html



- データを数表として扱うことを可能にするライブラリ
- 独自のデータ構造により様々なデータの加工が可能になる
- 簡単に言えば、Excelを操作するようにデータを扱える

Pandasの強み

- 数値データや時系列データ、文字列を処理するメソッドが揃っている
- データ集約などの統計処理に強い
- csv、excel、jsonなど様々なファイル形式を読み書きすることが可能

Pandasの弱み

- 計算速度が遅い
- メモリ消費が激しい

Pandasのデータ構造



Pandasでよく使われるデータ構造は Series と DataFrame

Series (1次元)

ID

1行目 0 100

2行目 1 101

3行目 2 102

4行目 3 103

5行目 4 104

DataFrame (2次元)

1列目

ID

1行目 0 100

2行目 1 101

3行目 2 102

4行目 3 103

5行目 4 104

2列目

City

Tokyo

Osaka

Kyoto

Hokkaido

Tokyo

3列目

Birth_year

1990

1989

1992

1997

1982

4列目

Name

Hiroshi

Akiko

Yuki

Satoru

Steve

Pandasのデータ構造



DataFrameはNumpyの二次元配列に行ラベル・列ラベルがついたもの
行ラベルを **index** ・列ラベルを **columns** という

Series (1次元)

	ID
1行目	0 100
2行目	1 101
3行目	2 102
4行目	3 103
5行目	4 104

index values

DataFrame (2次元)

	1列目	2列目	3列目	4列目	columns
	ID	City	Birth_year	Name	
1行目	0 100	Tokyo	1990	Hiroshi	
2行目	1 101	Osaka	1989	Akiko	
3行目	2 102	Kyoto	1992	Yuki	
4行目	3 103	Hokkaido	1997	Satoru	
5行目	4 104	Tokyo	1982	Steve	

index values

- PandasではDataFrame形式を扱うメソッドが数多くあります.

本講義で扱う内容・キーワード

データの選択と代入 / データの抽出 / 値のソート / データの結合 / データの削除

データの集約とグループ演算 / 階層型インデックス / 欠損値の取り扱い

- 全てを覚える必要はありません.

- Pandasを使ってどのようなことができるのか,
大まかなイメージを理解することが本日の目標になります.

実際にコードを動かしてみましょう
教材のノートブックを開いてください

データの抽出



データを抽出する方法は、**何を取得するか・何を指定するか**によって使い分ける

ラベル指定

インデックス指定

Seriesや
DataFrame
を取得する

loc df.loc[0:3, ['ID','Birth_year']]

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

単独の要素
を取得する

at df.at[2,'Birth_year']

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

iloc df.iloc[0:4, [0,2]]

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

iat df.iat[2, 2]

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

データの結合 (merge)

M

内部結合では、両方のデータにキーが存在する場合に結合する

	df1			
	id	city	birth_year	name
0	100	Tokyo	1990	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	Hokkaido	1997	Satoru
4	104	Tokyo	1982	Steeve
5	106	Tokyo	1991	Mituru
6	108	Osaka	1988	Aoi
7	110	Kyoto	1990	Tarou
8	111	Hokkaido	1995	Suguru
9	113	Tokyo	1981	Mitsuo

	df2				
	id	math	english	sex	index_num
0	100	50	90	M	0
1	101	43	30	F	1
2	102	33	20	F	2
3	105	76	50	M	3
4	107	98	30	M	4

【キーの指定】

- on, left_on, right_on,
left_index, right_index引数で指定可能
- 指定なしの場合,
両方のデータに共通のカラムがキーとなる

id (キー) がdf1・df2ともに
存在する行のみ結合される

pd.merge(df1, df2)

	id	city	birth_year	name	math	english	sex	index_num
0	100	Tokyo	1990	Hiroshi	50	90	M	0
1	101	Osaka	1989	Akiko	43	30	F	1
2	102	Kyoto	1992	Yuki	33	20	F	2

データの結合 (merge)

M

左外部結合では、左のデータのキーをもとに結合する

df1				df2							
	id	city	birth_year	name		id	math	english	sex	index_num	
0	100	Tokyo	1990	Hiroshi		0	100	50	90	M	0
1	101	Osaka	1989	Akiko		1	101	43	30	F	1
2	102	Kyoto	1992	Yuki		2	102	33	20	F	2
3	103	Hokkaido	1997	Satoru		3	105	76	50	M	3
4	104	Tokyo	1982	Steeve		4	107	98	30	M	4
5	106	Tokyo	1991	Mituru							
6	108	Osaka	1988	Aoi							
7	110	Kyoto	1990	Tarou							
8	111	Hokkaido	1995	Suguru							
9	113	Tokyo	1981	Mitsuo							

pd.merge(df1, df2, how = 'left')

左のデータ (df1) の
情報量は失われない

	id	city	birth_year	name	math	english	sex	index_num
0	100	Tokyo	1990	Hiroshi	50.0	90.0	M	0.0
1	101	Osaka	1989	Akiko	43.0	30.0	F	1.0
2	102	Kyoto	1992	Yuki	33.0	20.0	F	2.0
3	103	Hokkaido	1997	Satoru	NaN	NaN	NaN	NaN
4	104	Tokyo	1982	Steeve	NaN	NaN	NaN	NaN
5	106	Tokyo	1991	Mituru	NaN	NaN	NaN	NaN
6	108	Osaka	1988	Aoi	NaN	NaN	NaN	NaN
7	110	Kyoto	1990	Tarou	NaN	NaN	NaN	NaN
8	111	Hokkaido	1995	Suguru	NaN	NaN	NaN	NaN
9	113	Tokyo	1981	Mitsuo	NaN	NaN	NaN	NaN

データの結合 (merge)

M

完全外部結合では、どちらかのデータにキーがあれば結合する

	df1			
	id	city	birth_year	name
0	100	Tokyo	1990	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	Hokkaido	1997	Satoru
4	104	Tokyo	1982	Steeve
5	106	Tokyo	1991	Mituru
6	108	Osaka	1988	Aoi
7	110	Kyoto	1990	Tarou
8	111	Hokkaido	1995	Suguru
9	113	Tokyo	1981	Mitsuo

	df2				
	id	math	english	sex	index_num
0	100	50	90	M	0
1	101	43	30	F	1
2	102	33	20	F	2
3	105	76	50	M	3
4	107	98	30	M	4

どちらのデータも
情報量は失われない

	id	city	birth_year	name	math	english	sex	index_num
0	100	Tokyo	1990.0	Hiroshi	50.0	90.0	M	0.0
1	101	Osaka	1989.0	Akiko	43.0	30.0	F	1.0
2	102	Kyoto	1992.0	Yuki	33.0	20.0	F	2.0
3	103	Hokkaido	1997.0	Satoru	NaN	NaN	NaN	NaN
4	104	Tokyo	1982.0	Steeve	NaN	NaN	NaN	NaN
5	106	Tokyo	1991.0	Mituru	NaN	NaN	NaN	NaN
6	108	Osaka	1988.0	Aoi	NaN	NaN	NaN	NaN
7	110	Kyoto	1990.0	Tarou	NaN	NaN	NaN	NaN
8	111	Hokkaido	1995.0	Suguru	NaN	NaN	NaN	NaN
9	113	Tokyo	1981.0	Mitsuo	NaN	NaN	NaN	NaN
10	105	NaN	NaN	NaN	76.0	50.0	M	3.0
11	107	NaN	NaN	NaN	98.0	30.0	M	4.0

pd.merge(df1, df2, how = 'outer')

データの結合 (concat)

縦結合では、キーを指定せずにデータを積み上げて結合する

df1				
	id	city	birth_year	name
0	100	Tokyo	1990	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	Hokkaido	1997	Satoru
4	104	Tokyo	1982	Steeve
5	106	Tokyo	1991	Mituru
6	108	Osaka	1988	Aoi
7	110	Kyoto	1990	Tarou
8	111	Hokkaido	1995	Suguru
9	113	Tokyo	1981	Mitsuo

df3				
	id	city	birth_year	name
0	117	Chiba	1990	Suguru
1	118	Kanagawa	1989	Kouichi
2	119	Tokyo	1992	Satochi
3	120	Fukuoka	1997	Yukie
4	125	Okinawa	1982	Akari

	id	city	birth_year	name
0	100	Tokyo	1990	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	Hokkaido	1997	Satoru
4	104	Tokyo	1982	Steeve
5	106	Tokyo	1991	Mituru
6	108	Osaka	1988	Aoi
7	110	Kyoto	1990	Tarou
8	111	Hokkaido	1995	Suguru
9	113	Tokyo	1981	Mitsuo
0	117	Chiba	1990	Suguru
1	118	Kanagawa	1989	Kouichi
2	119	Tokyo	1992	Satochi
3	120	Fukuoka	1997	Yukie
4	125	Okinawa	1982	Akari

共通カラムで
積み上げる

pd.concat([df1,df3])

データの結合 (concat)



縦結合でカラム名が異なる場合には、データが不足している部分が欠損値となる

df1				df2					結果								
	id	city	birth_year		id	math	english	sex	index_num	birth_year	city	english	id	index_num	math	name	sex
0	100	Tokyo	1990	Hiroshi	0	100	50	90	M	1990.0	Tokyo	NaN	100	NaN	NaN	Hiroshi	NaN
1	101	Osaka	1989	Akiko	1	101	43	30	F	1989.0	Osaka	NaN	101	NaN	NaN	Akiko	NaN
2	102	Kyoto	1992	Yuki	2	102	33	20	F	1992.0	Kyoto	NaN	102	NaN	NaN	Yuki	NaN
3	103	Hokkaido	1997	Satoru	3	105	76	50	M	1997.0	Hokkaido	NaN	103	NaN	NaN	Satoru	NaN
4	104	Tokyo	1982	Steeve	4	107	98	30	M	1982.0	Tokyo	NaN	104	NaN	NaN	Steeve	NaN
5	106	Tokyo	1991	Mituru						1991.0	Tokyo	NaN	106	NaN	NaN	Mituru	NaN
6	108	Osaka	1988	Aoi						1988.0	Osaka	NaN	108	NaN	NaN	Aoi	NaN
7	110	Kyoto	1990	Tarou						1990.0	Kyoto	NaN	110	NaN	NaN	Tarou	NaN
8	111	Hokkaido	1995	Suguru						1995.0	Hokkaido	NaN	111	NaN	NaN	Suguru	NaN
9	113	Tokyo	1981	Mitsuo						1981.0	Tokyo	NaN	113	NaN	NaN	Mitsuo	NaN

pd.concat([df1, df2], sort=True)

データの結合 (concat)

M

横結合では、カラムを無視してインデックスをもとに結合する

df1

	id	city	birth_year	name
0	100	Tokyo	1990	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	Hokkaido	1997	Satoru
4	104	Tokyo	1982	Steeve
5	106	Tokyo	1991	Mituru
6	108	Osaka	1988	Aoi
7	110	Kyoto	1990	Tarou
8	111	Hokkaido	1995	Suguru
9	113	Tokyo	1981	Mitsuo

df2

	id	math	english	sex	index_num
0	100	50	90	M	0
1	101	43	30	F	1
2	102	33	20	F	2
3	105	76	50	M	3
4	107	98	30	M	4

不足しているデータは
欠損値となる

pd.concat([df1, df2], axis=1)

	id	city	birth_year	name	id	math	english	sex	index_num
0	100	Tokyo	1990	Hiroshi	100	50.0	90.0	M	0.0
1	101	Osaka	1989	Akiko	101	43.0	30.0	F	1.0
2	102	Kyoto	1992	Yuki	102	33.0	20.0	F	2.0
3	103	Hokkaido	1997	Satoru	105	76.0	50.0	M	3.0
4	104	Tokyo	1982	Steeve	107	98.0	30.0	M	4.0
5	106	Tokyo	1991	Mituru	NaN	NaN	NaN	NaN	NaN
6	108	Osaka	1988	Aoi	NaN	NaN	NaN	NaN	NaN
7	110	Kyoto	1990	Tarou	NaN	NaN	NaN	NaN	NaN
8	111	Hokkaido	1995	Suguru	NaN	NaN	NaN	NaN	NaN
9	113	Tokyo	1981	Mitsuo	NaN	NaN	NaN	NaN	NaN

階層型インデックス

インデックスやカラムを階層化し、視認性と集計のしやすさを向上する

	店名	場所	商品名	色	売上
0	A店	大阪	商品1	赤	200.0
1	A店	大阪	商品1	青	0.0
2	A店	大阪	商品2	赤	100.0
3	A店	東京	商品1	赤	500.0
4	A店	東京	商品1	青	300.0
5	A店	東京	商品2	赤	400.0
6	B店	大阪	商品1	赤	800.0
7	B店	大阪	商品1	青	600.0
8	B店	大阪	商品2	赤	700.0



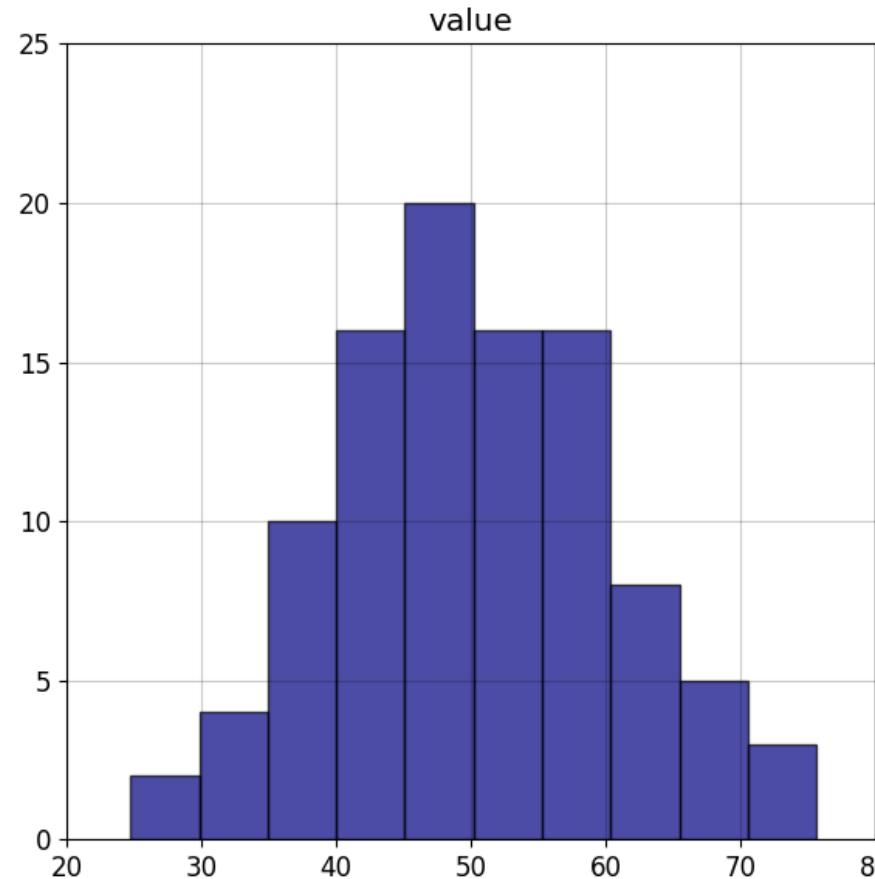
	商品名	商品1	商品2	商品1	level 0
	色	青	赤	赤	level 1 (level -1)
	店名	場所			
A店	大阪	0	100	200	
	東京	300	400	500	
B店	大阪	600	700	800	

level 0 level 1
(level -1)

- データ自体を変えずに、視認性が向上する
- フィルタリングや集計がしやすくなる

BIN分割を行うときは、データを等間隔または等個数で分割する

```
data = DataFrame({"value":np.random.normal(50,10,100)}) # 正規分布に従う乱数
```



cut() : データを等間隔で分割

```
pd.cut(data["value"], bins=5)
```

	(24.702, 34.927]	(34.927, 45.101]	(45.101, 55.275]	(55.275, 65.449]	(65.449, 75.624]
count	6	26	36	24	8

qcut() : データを等個数で分割

```
pd.qcut(data["value"], q=5)
```

	(24.752, 42.199]	(42.199, 46.881]	(46.881, 53.138]	(53.138, 58.002]	(58.002, 75.624]
count	20	20	20	20	20



松尾・岩澤研究室
MATSUO-IWASAWA LAB UTOKYO