IML#6 発表資料

Pandas超入門 -Pandasのデータ構造と演算-

2016/10/12
sunouchi

自己紹介

- 名前：sunouchi
- 仕事：デザイン、フロントエンド、データマイニングなど
- 近況：最近はデータマイニングを勉強中

目的

- Pandasの使い方を構造的に理解したい

話すこと

Pandasの...

- データ構造
- 演算処理

話さないこと

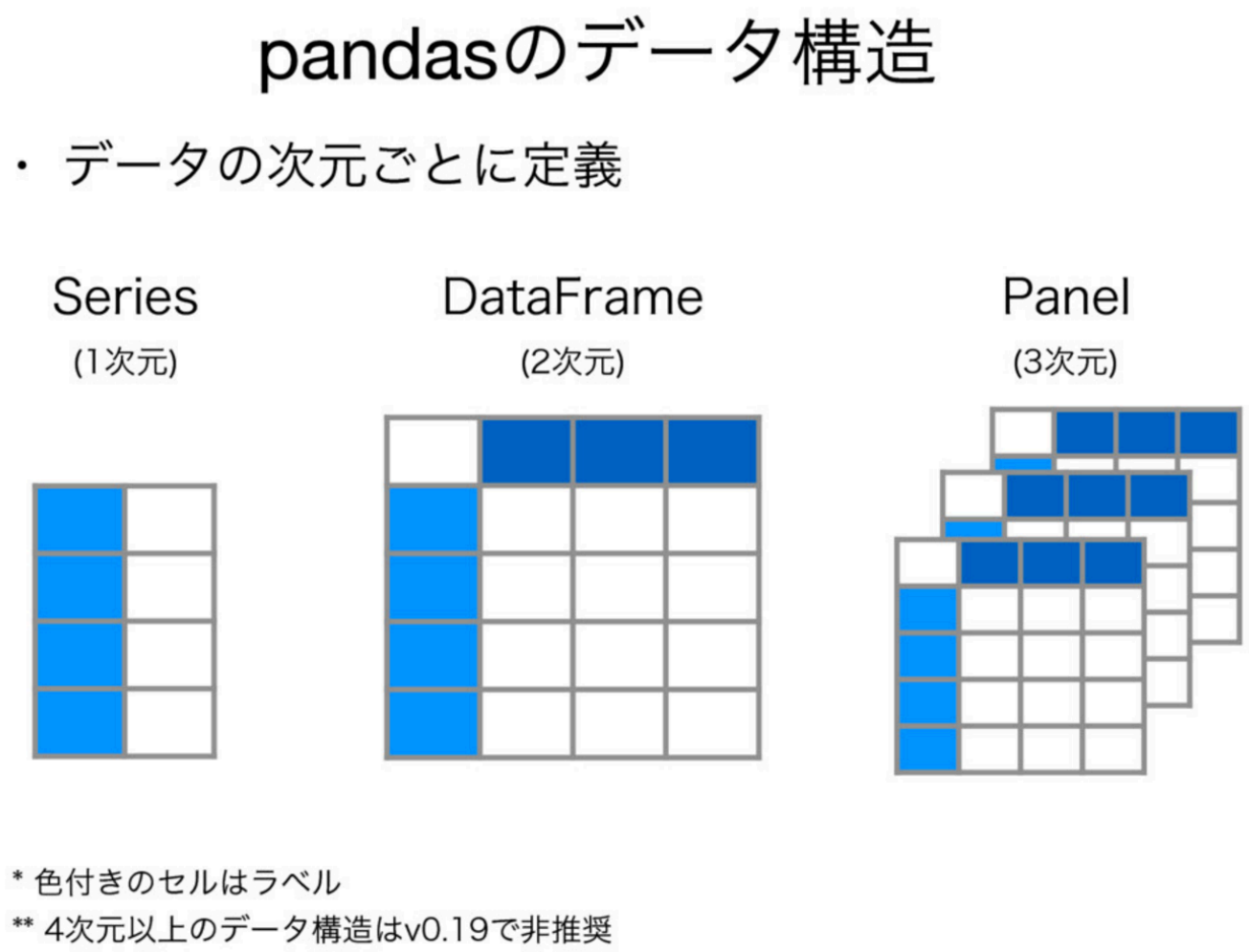
- インストール方法
- RやSQLなど他ツールとの比較
- データの視覚化 など

Pandasとは

- データ解析を支援する機能を提供するPythonのライブラリ
- スプレッドシートおよび時系列データを操作するためのデータ構造
- データの前処理や集計における便利な演算機能
- 作者: Wes McKinney
- 意味: PANel DAta System (=PANDAS)

データ構造

- Series: 1次元
- DataFrame: 2次元
- ~~Panel: 3次元（今回は対象外）~~



https://speakerdeck.com/sinhrks/pyconjp-2016-pandas-niyoru-shi-xi-lie-detachu-li

Series

- 1次元の配列のようなオブジェクト
- データ配列と、インデックスというデータのラベル配列から構成される

Seriesの作成

- データ配列のみ

```
In [4]: obj = pd.Series([4,7,-5,3])

In [5]: obj
Out[5]:
0    4
1    7
2   -5
```

```
8 3    3
9 dtype: int64
```

- データ配列とインデックス

```
1 In [6]: obj2 = pd.Series([4,7,-5,3], index=['aaa','bbb','ccc','ddd'])
2
3 In [7]: obj2
4 Out[7]:
5 aaa    4
6 bbb    7
7 ccc   -5
8 ddd    3
9 dtype: int64
```

DataFrame

- 2次元のスプレッドシート風のデータ構造を持ったオブジェクト
- 行と列の両方にインデックスを持っている

DataFrameの作成

- 同じ長さを持つリスト型の値を持ったディクショナリかNumPy配列

```
1 # 列はソートされた順番に配置される
2 In [9]: data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
3    ...:         'year' : [2000, 2001, 2002, 2001, 2002],
4    ...:         'pop':   [1.5, 1.7, 3.6, 2.4, 2.9]}
5    ...: frame = pd.DataFrame(data)
6    ...:
7
8 In [10]: frame
9 Out[10]:
10    pop   state  year
11 0  1.5    Ohio  2000
12 1  1.7    Ohio  2001
13 2  3.6    Ohio  2002
14 3  2.4  Nevada  2001
15 4  2.9  Nevada  2002
```

- DataFrameはSeriesの集まり

```
1 # ひとつの列がSeries型になっている
2 In [12]: type(frame['pop'])
3 Out[12]: pandas.core.series.Series
```

インデックスオブジェクト

- Series、DataFrameのインデックスの正体
- 軸のラベルやその他のメタデータ（軸のname属性やnames属性など）を保持する役割を持つ

- シリーズやデータフレームを初期化する際に指定されたラベルは、内部的にはインデックスオブジェクトに変換される

```
In [29]: obj = pd.Series(range(3), index=['aaa','bbb','ccc'])

In [30]: index = obj.index

In [31]: index
Out[31]: Index(['aaa', 'bbb', 'ccc'], dtype='object')
```

- インデックスオブジェクトはイミュータブルである（変更できない）
    - データ構造の中でインデックスオブジェクトを安全に共有するため

```
In [32]: index[1] = 'ddd'
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-32-7a0b071ec6b6> in <module>()
----> 1 index[1] = 'ddd'

/Users/sunouchi/.pyenv/versions/3.5.2/lib/python3.5/site-packages/pandas/indexes/base.py
in __setitem__(self, key, value)
   1243
   1244     def __setitem__(self, key, value):
-> 1245         raise TypeError("Index does not support mutable operations")
   1246
   1247     def __getitem__(self, key):

TypeError: Index does not support mutable operations
```

インデックスオブジェクトのメソッドと属性

メソッド	説明
append	追加のインデックスオブジェクトを連結し、新しくインデックスオブジェクトを生成する
diff	集合の差を計算して、その差をインデックスオブジェクトとして表現する
intersection	集合の論理積を計算する
union	和集合を計算する
isin	各値が集合に含まれているかどうかを示すブール型の配列を計算する
delete	指定したi番目の要素を削除した新しインデックスオブジェクトを作成する
drop	指定した値を削除した新しいインデックスオブジェクトを作成する
insert	指定したi番目に要素を挿入して新しいインデックスオブジェクトを作成する
is_monotonic	各要素が1つ前の要素と等しいか大きい場合にTrueが戻される
is_unique	インデックスオブジェクトが重複した値を持たない場合にTrueが戻される
unique	インデックスオブジェクトから重複のない値の配列を計算する

演算

演算あれこれ

- 四則演算: 基本的な演算
- 算術演算: 演算の挙動をもう少し制御
- 集約関数: 列グループに対して行う演算
- 統計関数: 統計で用いる指標など（中央値、分散、標準偏差など）

四則演算

- 基本的な演算

```
In [33]: df = pd.DataFrame({'C1': [1, 1, 1],
    ...:                    'C2': [1, 1, 1],
    ...:                    'C3': [1, 1, 1]})

In [34]: df
Out[34]:
   C1  C2  C3
0   1   1   1
1   1   1   1
2   1   1   1

In [35]: df + 1
Out[35]:
   C1  C2  C3
0   2   2   2
1   2   2   2
2   2   2   2

In [36]: df + np.array([1, 2, 3])
Out[36]:
   C1  C2  C3
0   2   3   4
1   2   3   4
2   2   3   4
```

算術演算

- 列に対するブロードキャスト (axis=0)
    - 列全体に対して演算する
    - axis=0 もしくは axis='index' で列に対する演算
    - axis=1 もしくは axis='column' で行に対する演算

```
# データを元に戻す
In [37]: df = pd.DataFrame({'C1': [1, 1, 1],
    ...:                    'C2': [1, 1, 1],
    ...:                    'C3': [1, 1, 1]})

```

```
In [38]: df
Out[38]:
   C1  C2  C3
0   1   1   1
1   1   1   1
2   1   1   1

# 列に対する演算
In [39]: df.add(np.array([1, 2, 3]), axis=0)
Out[39]:
   C1  C2  C3
0   2   2   2
1   3   3   3
2   4   4   4

# 行に対する演算
In [40]: df.add(np.array([1, 2, 3]), axis=1)
Out[40]:
   C1  C2  C3
0   2   3   4
1   2   3   4
2   2   3   4
```

- 欠測値 (NaN) 要素へのパディング (fill_value)
    - データに欠測知(NaN)が含まれる時、その要素への演算の結果もNaNになる
    - fill_valueでNaNを指定した値に変換し、演算されるようにする

```
In [41]: # NaN を含む DataFrame を定義する
    ...: df_nan = pd.DataFrame({'C1': [1, np.nan, np.nan],
    ...:                        'C2': [np.nan, 1, np.nan],
    ...:                        'C3': [np.nan, np.nan, 1]})

In [42]: df_nan
Out[42]:
    C1   C2   C3
0  1.0  NaN  NaN
1  NaN  1.0  NaN
2  NaN  NaN  1.0

In [43]: # NaN を含むセルの演算結果は NaN
    ...: df.add(df_nan)
Out[43]:
    C1   C2   C3
0  2.0  NaN  NaN
1  NaN  2.0  NaN
2  NaN  NaN  2.0

In [44]: # NaNをfill_valueで変換することで、演算可能となる
    ...: df.add(df_nan, fill_value=0)
Out[44]:
    C1   C2   C3
0  2.0  1.0  1.0
1  1.0  2.0  1.0
2  1.0  1.0  2.0

```

- 主な算術メソッド

メソッド	説明
add	加算を行うメソッド(+)
sub	減算を行うメソッド(-)
div	除算を行うメソッド(/)
mul	乗算を行うメソッド(*)

一覧はこちら http://pandas.pydata.org/pandas-docs/stable/api.html#id4

集約関数

- 列グループに対して行う演算
- 基本的には数値型のみが対象。だが、一部の関数では数値以外の型も対象となる

```
In [45]: df6 = pd.DataFrame({'C1': ['A', 'B', 'C'],
    ...:                     'C2': [1, 2, 3],
    ...:                     'C3': [4, 5, 6]})

In [46]: df6
Out[46]:
  C1  C2  C3
0  A   1   4
1  B   2   5
2  C   3   6

In [47]: # mean は数値型のみ集約
    ...: df6.mean()
Out[47]:
C2    2.0
C3    5.0
dtype: float64

In [48]: # sum は 文字列も集約
    ...: df6.sum()
Out[48]:
C1    ABC
C2      6
C3     15
dtype: object

In [49]: # 数値以外が不要な場合は numeric_only=True
    ...: df6.sum(numeric_only=True)
Out[49]:
C2     6
C3    15
dtype: int64

In [50]: # 行方向へ適用したい場合は axis = 1
    ...: df6.sum(axis=1)
Out[50]:
0    5
1    7
2    9
```

```
dtype: int64
```

統計関数

- 統計学での指標となる値を求める関数

```
In [51]: df7 = pd.DataFrame({'C1': [1, 2, 3, 4],
    ...:                     'C2': [4, 5, 6, 7],
    ...:                     'C3': [2, 3, 3, 2]})

In [52]: # 不偏分散
    ...: df7.var()
Out[52]:
C1    1.666667
C2    1.666667
C3    0.333333
dtype: float64

In [53]: # 標本分散
    ...: df7.var(ddof=False)
Out[53]:
C1    1.25
C2    1.25
C3    0.25
dtype: float64

In [54]: # 不偏標準偏差
    ...: df7.std()
Out[54]:
C1    1.290994
C2    1.290994
C3    0.577350
dtype: float64

In [55]: # 標本標準偏差
    ...: df7.std(ddof=False)
Out[55]:
C1    1.118034
C2    1.118034
C3    0.500000
dtype: float64

In [56]: # 要約統計量の表示
    ...: df7.describe()
Out[56]:
              C1        C2       C3
count  4.000000  4.000000  4.00000
mean   2.500000  5.500000  2.50000
std    1.290994  1.290994  0.57735
min    1.000000  4.000000  2.00000
25%    1.750000  4.750000  2.00000
50%    2.500000  5.500000  2.50000
75%    3.250000  6.250000  3.00000
max    4.000000  7.000000  3.00000
```

- 要約統計量の一覧

メソッド	説明
count	NA値ではない要素の数
describe	シリーズやデータフレームの列に対して、複数のようやく統計量を求める
min, max	最小値、最大値を求める
argmin, argmax	最小値、最大値が得られた場所のインデックス値（整数）を求める
idxmin, idxmax	最小値、最大値が得られた場所のインデックス値を求める
quantile	データのパーセント点を0から1の範囲で求める
sum	合計値
mean	平均値
median	中央値
mad	平均値からの平均絶対偏差
var	分散
std	標準偏差
skew	歪度（3次のモーメント）
kurt	尖度（4次のモーメント）
cumsum	累積合計値
cummin, cummax	それぞれ累積の最小値と最大値
cumprod	累積の積
diff	1次の階差を求める（時系列データで便利）
pct_change	パーセントの変化を求める

まとめ

データ構造

- Series: 1次元
- DataFrame: 2次元
- Panel: 3次元

データ構造はインデックスオブジェクトによって列・行がラベル付けされている（ラベル名で操作できるから便利）

演算

- 四則演算: 基本的な演算
- 算術演算: 演算の挙動をもう少し制御
- 集約関数: 列グループに対して行う演算

- 統計関数: 統計で用いる指標など（中央値、分散、標準偏差など）

参考資料

- 『Pythonによるデータ分析入門』Wes McKinney著
- pandas: powerful Python data analysis toolkit — pandas 0.19.0 documentation
- PyConJP 2016: pandas による 時系列データ処理 // Speaker Deck
- Python pandas の算術演算 / 集約関数 / 統計関数まとめ - StatsFragments
- Pandasを用いた基礎分析 - Platinum Data Blog by BrainPad
- カンファレンススケジュール | PyCon JP 2016 in TOKYO