

pandasとMecabを使って 遊ぼう

お品書き

1. pandasの基本
2. pandas演習（ネガポジ解析）

pandasとは

- pandasは、プログラミング言語Pythonにおいて、データ解析を支援する機能を提供するライブラリである。特に、数表および時系列データを操作するためのデータ構造と演算を提供する。PandasはBSDライセンスのもとで提供されている。
※wikipediaより

pandasのデータ型

1. 一次元(配列) : **Series**
2. 二次元(表) : **DataFrame** ← pandasといえばこれ！
3. 三次元 : Panel

index と column

	colmun1	column2	...
index1			
index2			
...			

- 両方OK：
 - 数値
 - 文字列

series

```
series = pd.Series(["a", "b", "c", 1, 2, 3, np.nan])
series2 = pd.Series({"June": 37, "July": 40, "August": 40})

print(series[0])
# numpyと同じくスライス記法が使える (start : end : step)
print(series[1:2:2])
# インデックスが文字列でもOK
print(series2["June"])
```

dataframe

```
df = pd.DataFrame([[ "A", 1], [ "B", 2], [ "C", 3], [ "D", 4], [ "E", 5]])  
df2 = pd.DataFrame({ "row1": [1, 2, 3], "row2": [11, 12, 13]})
```

- df
引数に行のリストを取る（行ベクトル）
- df2
引数にcolumn名とその列の値を取る（列ベクトル）

データの中身を確認

- [df.head\(\)](#): デフォルトで最初の5行を返す．引数に表示する行数を設定できる．
- [df.tail\(\)](#): デフォルトで最後の5行を返す．引数に表示する行数を設定できる．
- [df.shape](#): データフレームオブジェクトの行数と列数を確認する
- [df.describe\(\)](#): データの大まかな特徴量を調べる
- [df.corr\(\)](#): データの相関係数を算出する

データの選択

1. 列の名前で選択

```
noun_df["word"]  
noun_df[0:3:1] # スライス表記
```

- 列の値ごとの合計を出してくれて便利

```
noun_df["emotion"].value_counts()
```

参考：[Indexing and selecting data](#)

行のindexで選択

- tmp_dfの形

tmp_df	A	B
one	a1	b1
two	a2	b2
three	a3	b3

- スライス表記でないと取得できない → iloc, locを使おう

```
tmp_df["one":"one"]
```

高度な値の選択

SeriesかDataFrameを返す

- [loc](#) ... 条件に合うデータを取得する。結構柔軟。
- [iloc](#) ... インデックスでデータを取得する。

データをただ一つだけ取り出す

- [at](#) ... index名, column名で指定
- [iat](#) ... データのインデックス番号で指定

loc

- スライス指定

```
tmp_df.loc["one":"three"]
```

- 条件指定

```
tmp_df.loc[tmp_df["A"] == "a3"]  
tmp_df.loc[(tmp_df["A"] == "a3") | (tmp_df["B"] == 1)]
```

iloc

- インデックスの番号で指定

```
tmp_df.iloc[1,1]
```

at

- index名，column名で指定

```
tmp_df.at["three", "A"]
```

- locはdf/seriesを返すので，atで指定もできる

```
tmp_df.loc["six"].at["B"]
```

iat

- 0行目の1列目

```
tmp_df.iat[0,1]
```

view と copy

- view

元のオブジェクトと同じメモリを参照する

`df["hoge"]` とかはこっち

- copy

元のオブジェクトと違うメモリを参照する

`df.loc` とかはこっち

もっと詳しく：[View and Copy of pandas.DataFrame\(zenn\)](#).

要素を置き換える

```
tmp_df["A"] = tmp_df["A"].str.replace("a", "z")
```

- 右辺はオブジェクトのコピーを返す。オブジェクト自身は書き換えない。

(ちなみにこのstrは `pandas.core.strings.accessor.StringMethods` であり、str型ではない)

列を結合する

- [pd.concat](#): dataframe, seriesを結合する。縦方向に結合するか、横方向に結合するかは `axis=0/axis=1` で指定する。
- [pd.append](#): v1.40で廃止されることになった。concatを使おう。

tmp_dfとtmp_dfを結合する

```
# ignore_index=Trueとすると元のインデックスを無視して新しく数字を振り直す.  
df = pd.concat([tmp_df, tmp_df], ignore_index=True)
```

演習

追記 (あれば)