クレジット:
UTokyo Online Educationデータマイニング入門 2018 森　純一郎

東京大学
THE UNIVERSITY OF TOKYO

CC BY NC ND

データマイニング入門

Pythonの基礎

pandasライブラリ

pandasライブラリにはデータ分析作業を支援するためのモジュールが含まれています。以下では、pandasライブラリのモジュールの基本的な使い方について説明します。

pandasライブラリを使用するには、まず `pandas` モジュールをインポートします。慣例として、同モジュールを `pd` と別名をつけてコードの中で使用します。

In [ ]:
```
import pandas as pd
```

シリーズとデータフレーム

pandasは、リストや辞書などのデータを**シリーズ**(**Series**)あるいは**データフレーム**(**DataFrame**)のオブジェクトとして保持します。シリーズは列、データフレームは複数の列で構成されます。シリーズやデータフレームの行は**インデックス** `index` で管理され、インデックスには0から始まる番号、または任意のラベルが付けられています。シリーズやデータフレームは、インデックスをキー、各行を値とした辞書とみなすことができます。

シリーズ（Series）の作成

シリーズのオブジェクトは、以下のように、リストや辞書から作成することができます。また、Numpyの配列からもシリーズやオブジェクトを作成することができます。

In [ ]:
```
# リストからシリーズの作成
s1 = pd.Series([0,1,2,3,4,5])
print(s1)

# 辞書からシリーズの作成
s2 = pd.Series({0:'boo',1:'foo',2:'woo'})
print(s2)
```

以下では、シリーズ（列）より一般的なデータフレームの操作と機能について説明していきますが、データフレームオブジェクトの多くの操作や機能はシリーズオブジェクトにも適用できます。

データフレーム（DataFrame）の作成

データフレームのオブジェクトは、以下のように、リストや辞書、NumPyの配列から作成することができます。行のラベルは、 `DataFrame` の `index` 引数で指定できますが、以下のデータフレーム作成の例、d2, では同インデックスを省略しているため、０から始まるインデックス番号がラベルとして行に自動的に付けられます。列のラベルは `columns` 引数で指定します。辞書からデータフレームを作成する際は、 `columns` 引数で列の順番を指定することになります。

In [ ]:
```
# 多次元リストからデータフレームの作成
d1 = pd.DataFrame([[0,1,2],[3,4,5],[6,7,8],[9,10,11]], index=[10,11,12
print(d1)

# 辞書からデータフレームの作成
d2 = pd.DataFrame({'Initial':['B','F','W'], 'Name':['boo', 'foo', 'woo
print(d2)
```

csvファイルからのデータフレームの作成

`pandas` の `read_csv()` 関数を用いて、以下のように**csvファイル**を読み込んで、データフレームのオブジェクトを作成することができます。 `read_csv()` 関数の `encoding` 引数にはファイルの文字コードを指定します。csvファイル"simple_score.csv"には、以下のようにuserのID(user)、国語の点数（kokugo）、社会の点数（shakai）、数学の点数（sugaku）、理科の点数（rika）のデータが1行ずつ含まれています。

```
## simple_score.csv
user, kokugo, shakai, sugaku, rika
1, 30, 43, 51, 63
2, 39, 21, 49, 56
...
```

`head()` 関数を使うとデータフレームの先頭の複数行を表示させることができます。引数には表示させたい行数を指定し、行数を指定しない場合は、5行分のデータが表示されます。

In [ ]:
```
# csvファイルの読み込み
df = pd.read_csv('simple_score.csv')

# 先頭3行のデータを表示
df.head(3)
```

データフレームオブジェクトの `index` 属性により、データフレームのインデックスの情報が確認できます。 `len()` 関数を用いると、データフレームの行数が取得できます。

In [ ]:
```
print(df.index) #インデックスの情報
len(df.index) #インデックスの長さ
```

データの参照

シリーズやデータフレームでは、行の位置（行は0から始まります）を **スライス** として指定することで任意の行を抽出することができます。

In [ ]:
```
# データフレームの先頭3行のデータ
df[:3]
```

練習

データフレーム `df` の終端2行のデータをスライスを使って抽出してください

In [ ]:
```
# データフレームの終端2行のデータ
df[-2:]
```

データフレームから任意の列を抽出するには、 `DataFrame.列名` のように、データフレームオブジェクトに'.'で列名をつなげることで、その列を指定してシリーズオブジェクトとして抽出することができます。なお、列名を文字列として、 `DataFrame['列名']` のように添字指定しても同様です。

In [ ]:
```
# データフレームの'sugaku'の列の先頭3行のデータ
df['sugaku'].head(3)
```

データフレームの添字として、列名のリストを指定すると複数の列をデータフレームオブジェクトとして抽出することができます。

In [ ]:
```
# データフレームの'user'と'sugaku'の列の先頭3行のデータ
df[['user','sugaku']].head(3)
```

練習

データフレーム `df` の'user'と'rika'の列の先頭3行のデータを抽出してください

In [ ]:
```

```

ilocとloc

データフレームオブジェクトの **iloc** 属性を用いると、行と列の位置を指定して任意の行と列を抽出することができます。

In [ ]:
```
# データフレームの2行目のデータ
df.iloc[1]
```

In [ ]:
```
# データフレームの2行,2列目のデータ
df.iloc[1, 1]
```

In [ ]:
```
# データフレームの1から2行目,1から2列目のデータ
df.iloc[0:2, 0:2]
```

データフレームオブジェクトの **loc** 属性を用いると、抽出したい行のインデックス・ラベルや列のラベルを指定して任意の行と列を抽出することができます。複数のラベルはリストで指定します。行のインデックスは各行に割当てられた番号で、 `iloc` で指定する行の位置とは必ずしも一致しないことに注意してください。

In [ ]:
```
# データフレームの行インデックス1のデータ
df.loc[1]
```

In [ ]:
```
# データフレームの行インデックス2と'sugakuと列のデータ
df.loc[2, 'sugaku']
```

In [ ]:
```
# データフレームの行インデックス0から1と'user'と'kokugo'の列のデータ
df.loc[0:1, ['user','sugaku']]
```

練習

データフレーム `df` の3から4行目,1列目と5列目のデータを `iloc` , `loc` それぞれを使って抽出してください

In [ ]:
```

```

データの条件取り出し

データフレームの列の指定と併せて条件を指定することで、条件にあった行からなるデータフレームを抽出することができます。条件式のブール演算では、 `and` 、 `or` 、 `not` の代わりに `&` 、 `|` 、 `~` を用います。

In [ ]:
```
# データフレームの'sugaku'列の値が50より大きく、'rika'列の値が50より大きいデータ
df[(df['sugaku'] > 50) & (df['rika'] > 50)]
```

練習

データフレーム `df` から `kokugo` と `shakai` がどちらとも50点以上でのデータを抽出してください

In [ ]:
```

```

列の追加と削除

データフレームに列を追加する場合は、以下のように、追加したい新たな列名を指定し、シリーズ、リストやNumPyの配列を代入すると新たな列を追加できます。

In [ ]:
```
# データフレームに'eigo'という列を追加
df['eigo']=[50,50,60,60,60]

# データフレームに'ongaku'という列を追加
df['ongaku']=pd.Series([40,50,60,70,80])
df
```

`del` ステートメントを用いると、以下のようにデータフレームから任意の列を削除できます。

In [ ]:
```
# データフレームから'ongaku'という列を削除
del df['ongaku']
df
```

`assign` `()` メソッドを用いると、追加したい列名とその値を指定することで、以下のように新たな列を追加したデータフレームを新たに作成することができます。この際、元のデータフレームは変更されないことに注意してください。

In [ ]:
```
# データフレームに'mycolumn'という列を追加し新しいデータフレームを作成
df1 = df.assign(ongaku=pd.Series([40,50,60,70,80]))
df1
```

drop () メソッドを用いると、削除したい列名を指定することで、以下のように任意の列を削除したデータフレームを新たに作成することができます。列を削除する場合は、 axis 引数に1を指定します。この際、元のデータフレームは変更されないことに注意してください。

In [ ]:
```
# データフレームから'mycolumn'という列を削除し、新しいデータフレームを作成
df2 = df1.drop('ongaku',axis=1)
df2
```

練習

データフレーム df2 に新たな列を追加して新しいデータフレームを作成してください

In [ ]:
```

```

行の追加と削除

pandas モジュールの **append** () 関数を用いると、元のデータフレームに新たな行を追加した新たなデータフレームを作成することができます。以下では、 df データフレームの最終行に新たな行を追加し、新たなデータフレームを作成しています。 ignore_index 引数をTrueにすると追加した行に新たなインデックス番号がつけられます。

In [ ]:
```
# 追加する行のデータフレーム
row = pd.DataFrame([[6,10,20,30,40,50]], columns=df.columns)

# データフレームに行を追加し新しいデータフレームを作成
df3 = df.append(row, ignore_index=True)
df3
```

drop () メソッドを用いると、行のインデックスまたはラベルを指定することで行を削除することもできます。この時に、 axis 引数は省略することができます。

In [ ]:
```
# データフレームから行インデックス5の行を削除し、新しいデータフレームを作成
df4 = df3.drop(5)
df4
```

練習

データフレーム df4 に新たな行を追加して新しいデータフレームを作成してください

In [ ]:
```

```

データの並び替え

データフレームオブジェクトの **`sort_index()`** メソッドで、データフレームのインデックスに基づくソートができます。また、 **`sort_values()`** メソッドで、任意の列の値によるソートができます。列は複数指定することもできます。いずれのメソッドでも、 `inplace` 引数により、ソートにより新しいデータフレームを作成する（False）か、元のデータフレームを更新する（True)を指定できます。デフォルトは `inplace` はFalseになっており、 `sort_index()` メソッドは新しいデータフレームを作成します。

In [ ]:
```
# データフレームの2つ列の値に基づいて昇順にソート
df5 = df.sort_values(['eigo', 'sugaku'])
df5
```

列の値で降順にソートする場合は、 `sort_values()` メソッドの `ascending` 引数をFalseにしてください。

In [ ]:
```
# データフレームの2つ列の値に基づいて降順にソート
df6 = df.sort_values(['eigo', 'sugaku'],ascending=False)
df6
```

練習

データフレーム `df6` を `eigo` と `rika` の列の値に基づいて降順に並び替えて新しいデータフレーム作成してください

In [ ]:
```

```

データの統計量

データフレームオブジェクトの **`describe()`** メソッドで、データフレームの各列の要約統計量を求めることができます。要約統計量には平均、標準偏差、最大値、最小値などが含まれます。その他の統計量を求める `pandas` モジュールのメソッドは以下を参照してください。

pandasでの統計量計算 (https://pandas.pydata.org/pandas-docs/stable/api.html#api-dataframe-stats)

In [ ]:
```
# データフレームの各数値列の要約統計量を表示
df.describe()
```

練習

データフレーム `df` の各列ごと値の和を求めてください

In [ ]:
```

```

データの連結

pandas モジュールの **concat** () 関数を用いると、データフレームを連結して新たなデータフレームを作成することができます。以下では、 df データフレームの先頭2行と最終2行を連結して、新しいデータフレームを作成しています。

In [ ]:
```
# データフレームの先頭2行と最終2行を連結
concat_row_df = pd.concat([df[:2],df[-2:]])
concat_row_df
```

concat() 関数の axis 引数に1を指定すると、以下のように、データフレームを列方向にすることができます。

In [ ]:
```
# データフレームの'user'列と'sugaku'列を連結
concat_column_df = pd.concat([df.loc[:, ['user']],df.loc[:, ['sugaku']
concat_column_df.head(10)
```

データの結合

pandas モジュールの **merge** () 関数を用いると、任意の列の値をキーとして異なるデータフレームを結合することができます。結合のキーとする列名は on 引数で指定します。以下では、まずデータフレーム user_df を作成し、2つのデータフレーム、 user_df と df 、に共通の'user'の列の値をキーに、2つのデータフレームを結合して新しいデータフレーム score_df を作成しています。

In [ ]:
```
user_df = pd.DataFrame({'user':[1,2,3,4,5], 'name':['Paul', 'John', 'R
                        'class':['A', 'A', 'A','B','B']}, \
                        columns=['user','name', 'class'])
user_df
```

In [ ]:
```
# dfとuser_dfを'user'をキーにして結合
score_df = pd.merge(df, name_df, on='user')
score_df
```

練習

以下のデータフレームとデータフレーム df を'user'列の値をキーにして結合して新たなデータフレームを作成してくだい

In [ ]:
```
campus_df = pd.DataFrame({'user':[1,2,3,4,5], 'campus':['Komaba', 'Koma
campus_df
```

データのグループ化

データフレームオブジェクトの **`groupby`** `()` メソッドを使うと、データフレームの任意の列の値に基づいて、同じ値を持つ行をグループにまとめることができます。列は複数指定することもできます。 `groupby()` メソッドを適用するとグループ化オブジェクト（DataFrameGroupBy)が作成されますが、データフレームと同様の操作を多く適用することができます。

In [ ]:
```
# データフレームの'class'の値で行をグループ化
score_df.groupby('class')
```

In [ ]:
```
# グループごとの先頭2行を表示
score_df.groupby('class').head(2)
```

In [ ]:
```
# グループごとの"sugaku"列,"rika"列の値の平均を表示
score_df.groupby('class')[["sugaku","rika"]].mean()
```

練習

データフレーム `score_df` で'class'ごとの'kokugo'と'shakai'の平均を求めてください

In [ ]:
```

```