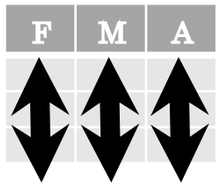
Data Wrangling with pandas Cheat Sheet

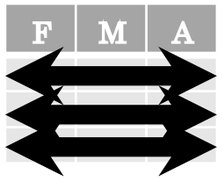
http://pandas.pydata.org

データの整然化 – データを「飼いならす」初歩

整然データでは:



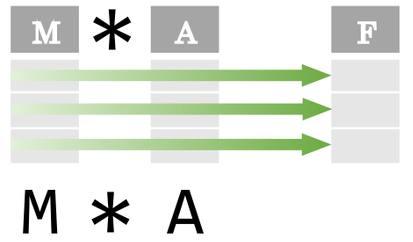
&



各**変数**はそれぞれの列で保持される

各**観測点**はそれぞれの行で保持される

整然データは pandas の **ベクトル操作**を補完する。pandas は変数に演算操作をしても自動的に観測点を保つ。直感的に機能するフォーマットはpandas を置いて他にない。



構文 – DataFrames 作成

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
         {"a" : [4 ,5, 6],
          "b" : [7, 8, 9],
          "c" : [10, 11, 12]},
        index = [1, 2, 3])
```
列ごとの値を指定する。

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```
行ごとの値を指定する。

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

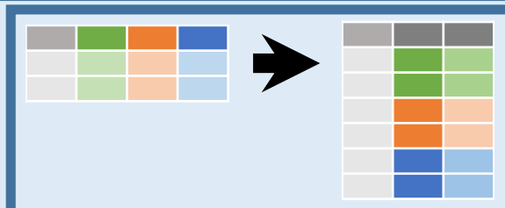
```
df = pd.DataFrame(
         {"a" : [4 ,5, 6],
          "b" : [7, 8, 9],
          "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
         [('d',1),('d',2),('e',2)],
           names=['n','v'])))
```
MultiIndex のある DataFrame を作成

メソッドチェイン

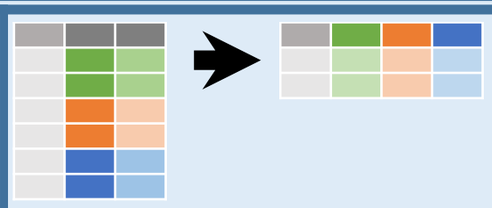
ほとんどの pandas メソッドは DataFrame を返す。返り値にさらにメソッドを適用できるようにするためだ。これによりコードの可読性が向上する。

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
    )
```

データの変形 – データセットのレイアウト変更



pd.melt(df)
各列を行へ集約 (ワイドからロングへ)。



df.pivot(columns='var', values='val')
行を列へと展開 (ロングからワイドへ)。

pd.concat([df1,df2])
複数 DataFrame の行を連結。

pd.concat([df1,df2], axis=1)
複数 DataFrame の列を連結。

df.sort_values('mpg')
列の値で行をソート (昇順)。

df.sort_values('mpg',ascending=False)
列の値で行をソート (降順)。

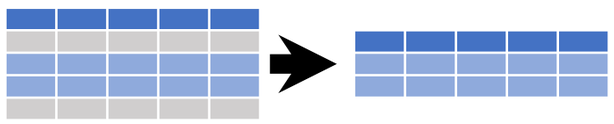
df.rename(columns = {'y':'year'})
DataFrame の列名を変更。

df.sort_index()
DataFrame のインデックスでソート。

df.reset_index()
現在の DataFrame インデックスを廃棄し行番号に変更。元のインデックスは列に移動。

df.drop(columns=['Length','Height'])
DataFrame から列を除外。

観測点 (行) の一部の取出し



df[df.Length > 7]
論理条件に合う行を取出す。

df.drop_duplicates()
重複する行を除外 (指定した列だけで判定)。

df.head(n)
冒頭の n 行を取出す。

df.tail(n)
末尾の n 行を取出す。

df.sample(frac=0.5)
行を指定割合だけ無作為抽出。

df.sample(n=10)
n 行だけ無作為抽出。

df.iloc[10:20]
指定位置で行を抽出。

df.nlargest(n, 'value')
上位から順に n 行を取出す。

df.nsmallest(n, 'value')
下位から順に n 行を取出す。

Python (と Pandas) の論理演算子

<	より小さい	!=	等しくない
>	より大きい	df.column.isin(*values*)	グループ関係
==	等しい	pd.isnull(*obj*)	NaN である
<=	以下	pd.notnull(*obj*)	NaN ではない
>=	以上	&,|,~,^,df.any(),df.all()	and, or, not, xor, any, all

変数 (列) の一部の取出し

df[['width','length','species']]
複数列を名前で指定して取出す。

df['width'] ***または*** **df.width**
1つの列を名前で指定して取出す。

df.filter(regex='*regex*')
正規表現 *regex* にマッチする名前の列を取出す。

regex (正規表現) の例

'¥.'	ピリオド '.' を含む文字列にマッチ
'Length$'	末尾に 'Length' のある文字列にマッチ
'^Sepal'	冒頭に 'Sepal' のある文字列にマッチ
'^x[1-5]$'	冒頭が 'x' で始まり末尾が 1,2,3,4,5 いずれかの文字列にマッチ
''^(?!Species$).*'	'Species' を含まない文字列にマッチ

df.loc[:,'x2':'x4']
x2 から x4 の間にある全ての列を取出す (両端含む)。

df.iloc[:,[1,2,5]]
1, 2, 5 番目の位置にある列を取出す。(最初の列は 0)。

df.loc[df['a'] > 10, ['a','c']]
論理条件に合う行、かつ指定した列を取出す。

データの要約

df['w'].value_counts()
変数の値の重複を除きカウント。
len(df)
DataFrame の行数。
df['w'].nunique()
列の値の重複を除きカウント。
df.describe()
各列 (または GroupBy) の基本的な要約統計量を表示。

pandas では様々な種類の pandas オブジェクト (DataFrame, columns, Series, GroupBy, Expanding, Rolling (下記参照)) を操作する多くの**要約関数**が提供され、各グループに対して1つの値を出力する。DataFrame に適用した場合、結果は1列に対し1要素の Series で返される。要約関数の例:

sum()
各オブジェクトの値の合計値。
count()
各オブジェクトの NA/null でない値のカウント。
median()
各オブジェクトの中央値。
quantile([0.25,0.75])
各オブジェクトの分位点。
apply(*function*)
各オブジェクトに関数を適用。
min()
各オブジェクトの最小値。
max()
各オブジェクトの最大値。
mean()
各オブジェクトの平均値。
var()
各オブジェクトの分散値。
std()
各オブジェクトの標準偏差。

データのグループ化

df.groupby(by="col")
“col” 列の値でグループ化した GroupBy オブジェクトを返す。

df.groupby(level="ind")
“ind” という名前のインデックスレベルの値でグループ化した GroupBy オブジェクトを返す。

上で紹介した要約関数は全て Groupby にも適用可能。GroupBy 専用の要約関数:

size()
各グループの大きさ(行数)。
agg(*function*)
関数でグループを要約。

以下の関数の例もグループに対して適用できる。この場合、各関数はグループ毎の成分に適用され、返り値のベクトルは元のデータフレームと同じ長さを持つ。

shift(1)
1 行先にずらした値のコピー。
rank(method='dense')
タイをギャップなしでランク付け。
rank(method='min')
タイを最下位でランク付け。
rank(pct=True)
ランクを[0, 1] の割合で出力。
rank(method='first')
タイを最上位でランク付け。
shift(-1)
1行ラグをとった値のコピー。
cumsum()
累積和。
cummax()
累積最大値。
cummin()
累積最小値。
cumprod()
累積積。

窓関数

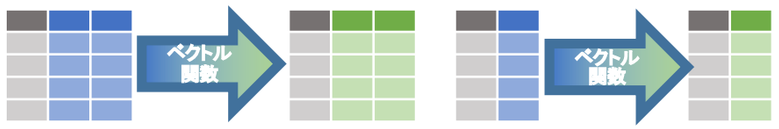
df.expanding()
要約関数を累積的に適用できるようにした Expand オブジェクトを返す。
df.rolling(n)
長さ n の窓に要約関数を適用できるようにした Rolling オブジェクトを返す。

欠損値の制御

df.dropna()
いずれかの列に NA/null のある行を除外 (ペアワイズ除去)。
df.fillna(value)
NA/null を value に置換。

新しい列の作成

df.assign(Area=lambda df: df.Length*df.Height)
1つ以上の新しい列を計算し追加。
df['Volume'] = df.Length*df.Height*df.Depth
列を1つ追加。
pd.qcut(df.col, n, labels=False)
列の値を n 個に区分けして並べる。

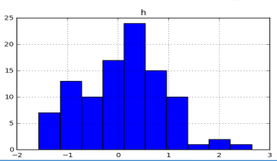


pandas ではデータフレームの全ての列または単一の列 (Series) に対して操作する多くの**ベクトル関数**が提供されている。これらの関数は各列に対して値のベクトルを返し、単一の Series に対しては1つの Series を出力する。ベクトル関数の例:

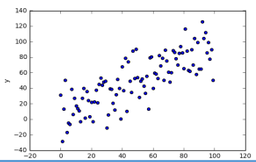
max(axis=1)
要素ごとの最大値。
clip(lower=-10,upper=10)
入力したしきい値でトリムする。
min(axis=1)
要素ごとの最小値。
abs()
絶対値。

プロット

df.plot.hist()
各列のヒストグラムを出力。



df.plot.scatter(x='w',y='h')
列のペアを散布図として出力。



データセットの結合

adf

x1	x2
A	1
B	2
C	3

bdf

x1	x3
A	T
B	F
D	T

標準的な結合

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

pd.merge(adf, bdf, how='left', on='x1')
bdf を adf のマッチする行へ結合 (左結合).

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

pd.merge(adf, bdf, how='right', on='x1')
adf を bdf のマッチする列へ結合 (右結合)。

x1	x2	x3
A	1	T
B	2	F

pd.merge(adf, bdf, how='inner', on='x1')
両方にある行のみ残して結合 (内部結合)。

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

pd.merge(adf, bdf, how='outer', on='x1')
全ての行を残して結合 (完全外部結合)。

フィルタリング結合

x1	x2
A	1
B	2

adf[adf.x1.isin(bdf.x1)]
adf の中で bdf にマッチする全ての行。

x1	x2
C	3

adf[~adf.x1.isin(bdf.x1)]
adf の中で bdf にマッチしない全ての行。

ydf

x1	x2
A	1
B	2
C	3

zdf

x1	x2
B	2
C	3
D	4

集合ライクな操作

x1	x2
B	2
C	3

pd.merge(ydf, zdf)
ydf と zdf の両方に現れる行 (積集合)。

x1	x2
A	1
B	2
C	3
D	4

pd.merge(ydf, zdf, how='outer')
ydf と zdf の片方もしくは両方に現れる行 (和集合)。

x1	x2
A	1

pd.merge(ydf, zdf, how='outer', indicator=True)
.query('_merge == "left_only"')
.drop(columns=['_merge'])
ydf にあるが zdf にはない行 (差集合)。