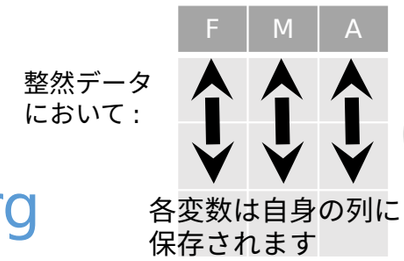
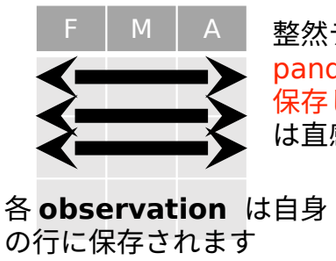
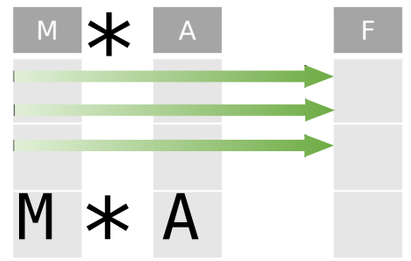
Data Wrangling
with pandas Cheat Sheet
http://pandas.pydata.org

整然データ（Tidy Data） – pandas における議論の基盤





整然データは**ベクトル操作**を補完する。
pandas は、あなたが変数を扱うがままに観測を保存します。他のどのフォーマットも pandas では直感的に動きません。



文法 – DataFrame の作成

	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])
```
各 column(列) の値をセットする
```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```
各 row(行) の値をセットする

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

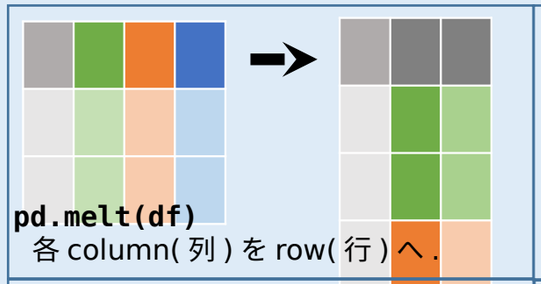
```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),
('e',2)],
              names=['n','v']))
```
MultiIndex で Dataframe を作成する

メソッドチェーン

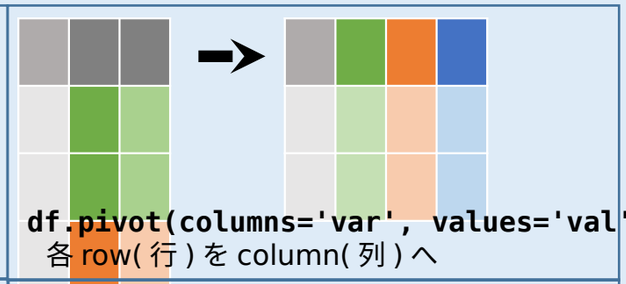
pandas における多くのメソッドは Dataframe を返します。そのため、メソッドの返り値にそのまま別のメソッドを適用するメソッドチェーンが便利に使えます。この手法はコードの可動性を大いにあげます。

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

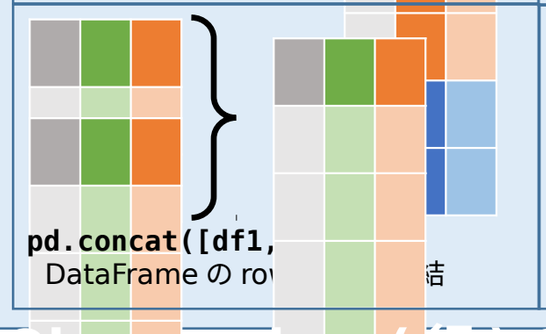
データの整形 (Reshaping Data) – データセットのレイアウト変更



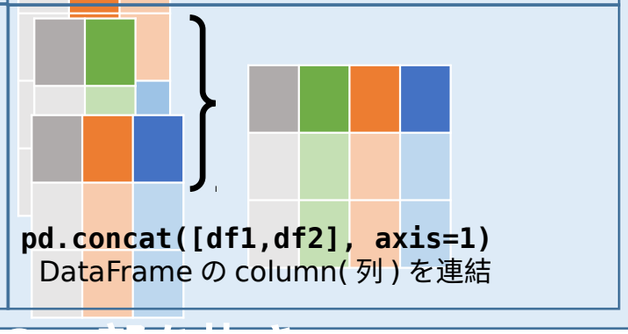
pd.melt(df)
各 column(列) を row(行) へ .



df.pivot(columns='var', values='val')
各 row(行) を column(列) へ



pd.concat([df1,
DataFrame の ro 結



pd.concat([df1,df2], axis=1)
DataFrame の column(列) を連結

df.sort_values('mpg')
column(列) の値を使って row(行) をソート (昇順)

df.sort_values('mpg',ascending=False)
column(列) の値を使って row(行) をソート (降順)

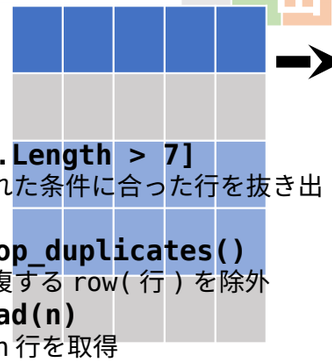
df.rename(columns = {'y':'year'})
DataFrame の column(列) 名を変更

df.sort_index()
DataFrame の index を使ってソート

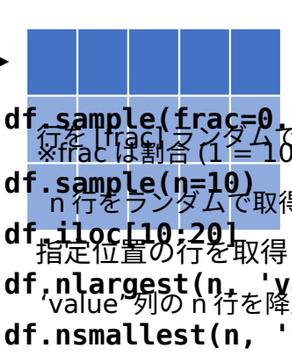
df.reset_index()
DataFrame の index をリセット

df.drop(columns=['Length','Height'])
指定した長さの column(列) を削除

Observations(行) の一部を抜き出し



df[df.Length > 7]
与えられた条件に合った行を抜き出す
df.drop_duplicates()
値の重複する row(行) を除外
df.head(n)
最初の n 行を取得
df.tail(n)
最初の n 行を取得

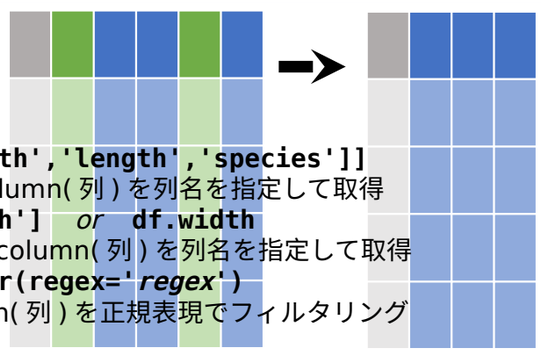


df.sample(frac=0.5)
行を frac ランダムで行を取得
※frac は割合 (1 = 100%)
df.sample(n=10)
n 行をランダムで取得
df.iloc[10:20]
指定位置の行を取得
df.nlargest(n, 'value')
'value' 列の n 行を降順で取得
df.nsmallest(n, 'value')
'value' 列の n 行を昇順で取得

Logic in Python (and pandas)

<	より小さい	!=	等しくない
>	より大きい	**df.column.isin(*values*)**	values が含まれている column に場合 true を返す
==	と等しい	**pd.isnull(*obj*)**	null である
<=	以下	**pd.notnull(*obj*)**	null でない
>=	以上	**&,|,~,^,df.any(),df.all()**	Logical and, or, not, xor, any, all

変数 (列) からの一部取得



df[['width','length','species']]
複数 column(列) を列名を指定して取得
df['width'] *or* **df.width**
1 つの column(列) を列名を指定して取得
df.filter(regex='*regex*')
column(列) を正規表現でフィルタリング

regex (正規表現) の例

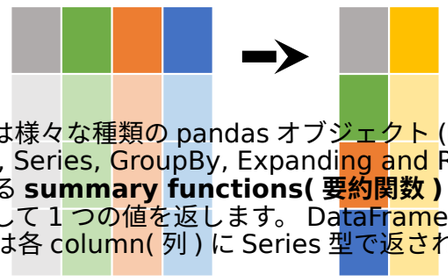
'\.'	ピリオド' .' を含む文字列にマッチ
'Length$'	末尾に' Length' のある文字列にマッチ
'^Sepal'	冒頭に' Sepal' のある文字列にマッチ
'^x[1-5]$'	'x' で始まり且つ末尾が 1~5 のいずれかである文字列にマッチ
''^(?!Species$).*'	Species' 以外の文字列とマッチ

df.loc[:,'x2':'x4']
x2 から x4 までの全ての column(列) を取得
df.iloc[:,[1,2,5]]
1,2,5 番目 (index が 5 番目) の列を取得 (index は 0 から数える)
df.loc[df['a'] > 10, ['a','c']]
与えられた条件に合った row(行) で且つ指定された column(列) を取得

http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants

データの要約

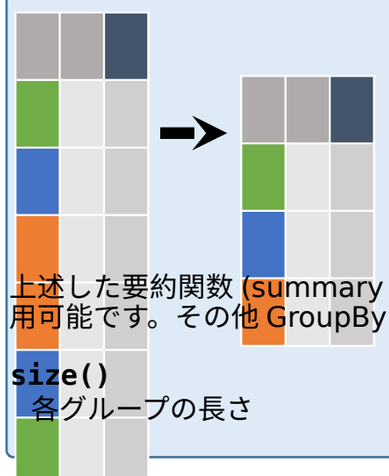
df['w'].value_counts()
　変数の出現回数をカウント
len(df)
　# DataFrame の行数を出力
df['w'].nunique()
　ユニークな値をカウントして出力
df.describe()
　Basic descriptive statistics for each column (or GroupBy)



pandas は様々な種類の pandas オブジェクト (DataFrame columns, Series, GroupBy, Expanding and Rolling(下記参照)) を操作する **summary functions(要約関数)** を提供し、各グループに対して 1 つの値を返します。DataFrame に適用された場合、結果は各 column(列) に Series 型で返されます。例 :

sum()
　各オブジェクトの値を合計
count()
　各オブジェクトの NA/null 以外の値をカウント
median()
　各オブジェクトの中央値を取得
quantile([0.25,0.75])
　各オブジェクトの分位値を取得
apply(*function*)
　各オブジェクトにを適用
min()
　各オブジェクトの最小値を取得
max()
　各オブジェクトの最大値を取得
mean()
　各オブジェクトの平均を取得
var()
　各オブジェクトの分散値を取得
std()
　各オブジェクトの標準偏差を取得

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")
　インデックスレベル “ind” でグループ化した GroupBy オブジェクトを返す

上述した要約関数 (summary function) は全て group にも適用可能です。その他 GroupBy の関数 :

size()
　各グループの長さ
agg(*function*)
　関数を使ってグループを集計

下記関数も group に対して適用できます。この場合、関数はグループ毎に適用され、返されるベクトルの長さは元の DataFrame と同じになります。

shift(1)
　1 行ずつ後ろにずらした値をコピー
rank(method='dense')
　ランク付け (同数はギャップなしで計算)
rank(method='min')
ランク付け (同数は小さい値にする)
rank(pct=True)
[0~1] の値でランク付け
rank(method='first')
ランク付け。同数の場合 index が小さい方が上位
shift(-1)
1 行ずつ前にずらした値をコピー
cumsum()
累積和
cummax()
累積最大値
cummin()
累積最小値
cumprod()
累積積

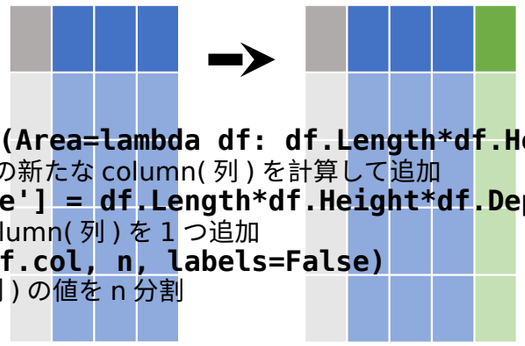
window 関数

df.expanding()
　要約関数を累積的に適用可能にした Expanding オブジェクトを返す
df.rolling(n)
　長さ n の window に要約関数を適用可能にした Rolling オブジェクトを返す

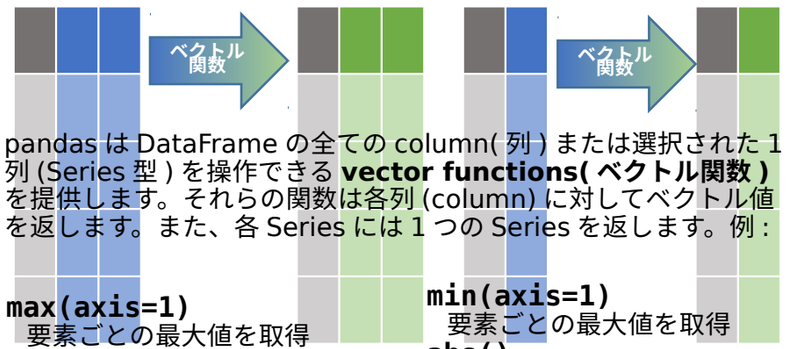
欠損データを扱う

df.dropna()
　NA/null を含む row(行) を除外する
df.fillna(value)
　NA/null を value に置換

新しい Column(列) の作成



df.assign(Area=lambda df: df.Length*df.Height)
　1 つ以上の新たな column(列) を計算して追加
df['Volume'] = df.Length*df.Height*df.Depth
　新たな column(列) を 1 つ追加
pd.qcut(df.col, n, labels=False)
　column(列) の値を n 分割

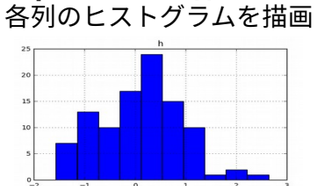


pandas は DataFrame の全ての column(列) または選択された 1 列 (Series 型) を操作できる **vector functions(ベクトル関数)** を提供します。それらの関数は各列 (column) に対してベクトル値を返します。また、各 Series には 1 つの Series を返します。例 :

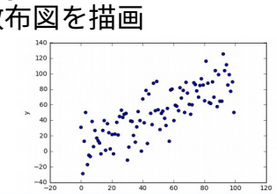
max(axis=1)
　要素ごとの最大値を取得
clip(lower=-10,upper=10)
　下限を − 10, 上限を 10 に設定してトリミング
min(axis=1)
　要素ごとの最大値を取得
abs()
　絶対値を取得

プロット (描画)

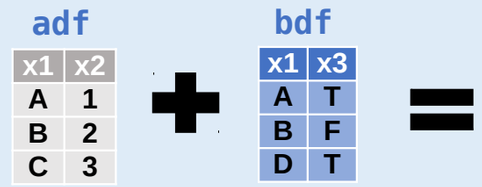
df.plot.hist()
　各列のヒストグラムを描画



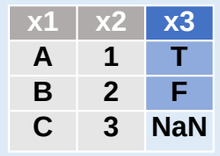
df.plot.scatter(x='w',y='h')
　散布図を描画



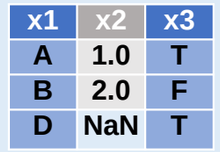
データの結合



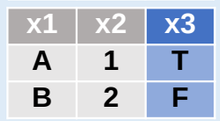
標準的な結合



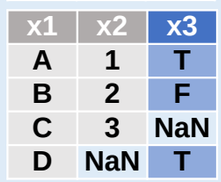
pd.merge(adf, bdf, how='left', on='x1')
　bdf を adf のマッチする行へ結合



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

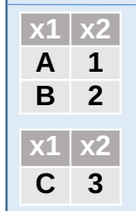


pd.merge(adf, bdf, how='inner', on='x1')
　adf と bdf を双方にある行のみ残して結合



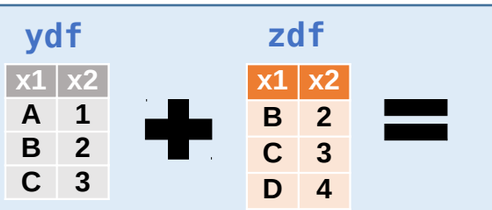
pd.merge(adf, bdf, how='outer', on='x1')
　全ての値と行を残して結合

フィルタリング結合

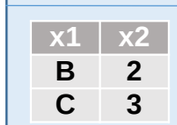


adf[adf.x1.isin(bdf.x1)]
　adf の中で bdf にマッチする行

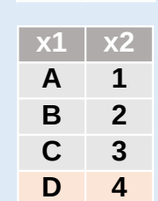
adf[~adf.x1.isin(bdf.x1)]
　adf の中で bdf にマッチしない行



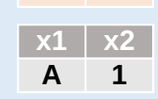
集合ライクな結合



pd.merge(ydf, zdf)
　ydf と zdf 両方にある行



pd.merge(ydf, zdf, how='outer')
　ydf と zdf の両方もしくは片方にある行



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