3-8 クロス集計

groupby メソッドで集約する

groupby() メソッドを使用して DataFrame を集約します。指定された列で集約されたオブジェクト DataFrameGroupBy が返されます。**リスト 3.8.1** では type 列で集約されたオブジェクトを作成しています（結果は**リスト 3.8.2**）。

● リスト 3.8.1 : DataFrame の集約

```
import os
import pandas as pd
import numpy as np

base_url = 'https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/'
anime_master_csv = os.path.join(base_url, 'anime_master.csv')
df = pd.read_csv(anime_master_csv)

grouped = df.groupby('type')
type(grouped)
```

● リスト 3.8.2 : DataFrame の集約（結果）

```
pandas.core.groupby.DataFrameGroupBy
```

前節（3-7 : 統計量を求める）同様の数学的、統計的なメソッドが使用できます（**リスト 3.8.3** と**表 3.29**、**リスト 3.8.4** と**表 3.30**）。詳細は pandas documentation API Reference[注3.17] を参照してください。

● リスト 3.8.3 : 集約されたデータの平均値

```
grouped.mean().round(1)
```

● 表 3.29 : 集約されたデータの平均値（結果）

type	anime_id	episodes	rating	members
Movie	14322.5	1.1	6.3	10654.0
Music	22495.1	1.1	5.6	1273.0
ONA	22738.0	6.8	5.6	4401.8
OVA	12207.7	2.5	6.5	6849.5
Special	16802.3	2.5	6.5	7424.6
TV	10929.6	37.5	6.9	41832.3

注 3.17　GroupBy URL http://pandas.pydata.org/pandas-docs/stable/api.html#groupby

● リスト 3.8.4 : 集約されたデータの基本統計量

```
grouped.describe().round(1).head(16)
```

● 表 3.30 : 集約されたデータの基本統計量（結果）

type		anime_id	episodes	members	rating
Movie	count	2220.0	2220.0	2220.0	2220.0
	mean	14322.5	1.1	10654.0	6.3
	std	10925.7	2.2	31603.6	1.2
	min	5.0	1.0	13.0	2.5
	25%	4396.8	1.0	119.0	5.4
	50%	10677.5	1.0	489.5	6.5
	75%	24071.5	1.0	4239.0	7.3
	max	34201.0	100.0	466254.0	10.0
Music	count	485.0	485.0	485.0	485.0
	mean	22495.1	1.1	1273.0	5.6
	std	10175.0	1.3	4489.0	1.0
	min	731.0	1.0	24.0	3.3
	25%	12101.0	1.0	97.0	5.0
	50%	24903.0	1.0	226.0	5.6
	75%	31925.0	1.0	797.0	6.2
	max	34412.0	24.0	71136.0	8.4

NOTE

pandas のバージョンによっては describe() メソッドの表示形式が異なることがあります。

複数の要素で集約する場合、groupby() メソッドの引数にリストを渡します。**リスト 3.8.5** では type 列と episodes 列に基づいてデータを集約し、平均を求めています（結果は**表 3.31**）。

● リスト 3.8.5 : 集約された複数列データの平均値

```
df.groupby(['type', 'episodes']).mean().round(1).head(20)
```

● 表 3.31 : 集約された複数列データの平均値（結果）

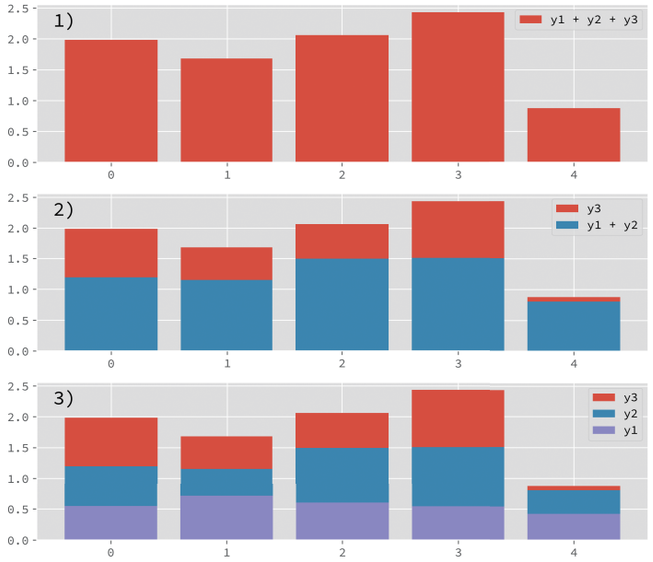
type	episodes	anime_id	rating	members
Movie	1	14320.0	6.3	10588.6
	2	13802.0	6.9	6638.9
	3	11339.3	6.7	53598.1
	4	15723.5	7.3	3566.5
	5	12558.3	6.1	3641.0
	6	8433.5	6.0	178.5
	7	13602.5	6.9	11989.5
	9	8928.0	6.2	267.0
	10	31020.0	6.9	57.0

積み上げ棒グラフを作成する

積み上げ棒グラフを描画する場合も、複数グループの棒グラフと同様、描画の際に工夫が必要です。**図 4.5.10** はy1、y2、y3という3つの値を積み上げる場合の描画順を説明する図です。実際の描画手順は次のようになります。

① y1とy2とy3の和を描画する
② ①にy2とy3の和を重ねて描画する
③ ②にy1を重ねて描画する

つまり、**リスト 4.5.7**、**図 4.5.6** と同様、同じX値を与えて描画すると後から書いた棒に上書きされてしまうため、手動で値を積算し、積算量の多いほうから順番に描画するという作業をします。

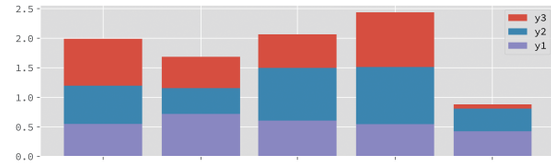


● 図 4.5.10 : 積み上げ棒グラフの描画順

図 4.5.10 − 3) のグラフを描画するためのコードは**リスト 4.5.12** で、出力結果は**図 4.5.11** になります。

● リスト 4.5.12 : 積み上げ棒グラフの描画

```
# データセットの作成
x = np.arange(5)
np.random.seed(0)
y = np.random.rand(15).reshape((3, 5))

y1, y2, y3 = y

y1b = np.array(y1)
y2b = y1b + np.array(y2)
y3b = y2b + np.array(y3)

# 積み上げ棒グラフの描画
fig = plt.figure(figsize=(10, 3))
ax = fig.add_subplot(111)

ax.bar(x, y3b, label='y3')
ax.bar(x, y2b, label='y2')
ax.bar(x, y1b, label='y1')

ax.legend()

plt.show()
```



● 図 4.5.11 : 積み上げ棒グラフ

NOTE

配列の加算を行う際にnumpy.ndarrayを用いているのは、データの要素どうしを加算するためです。Pythonのリストどうしの加算とは挙動が異なるので注意してください。

bottomオプションで積み上げ設定をする

積み上げ棒グラフを描画する際のオプションとしてbottomオプションがあります。下段にくるリスト-ライク・オブジェクトを引数bottomに指定することにより、積み上げ表示が行われます。

リスト 4.5.13 の出力結果は**図 4.5.11** と同じになります。しかし、**リスト 4.5.13** を見てわかるとおり、3グループ以上を積み上げる場合は、**リスト 4.5.12** と同様の処理をする必要があります。2つのグループの積み上げまではbottomオプションは有効ですが、それ以上を積み上げる場合は**リスト 4.5.12** の方法が有効です。

● リスト 4.5.13 : bottomオプションを用いて積み上げ棒グラフを描画

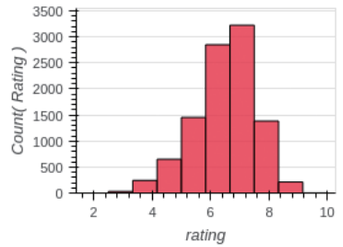
```
figure = plt.figure(figsize=(10, 3))
ax = figure.add_subplot(111)

ax.bar(x, y3, bottom=y2b, label='y3')
```

ビン数を変更する場合

ビン数を変更する場合は、キーワード引数binsに自然数を渡します（**リスト6.7.3**と**図6.7.3**）。

● リスト6.7.3：ビン数を設定したヒストグラム

```
p = Histogram(df, values='rating', plot_width=300, plot_height=200, bins=10)
show(p)
```



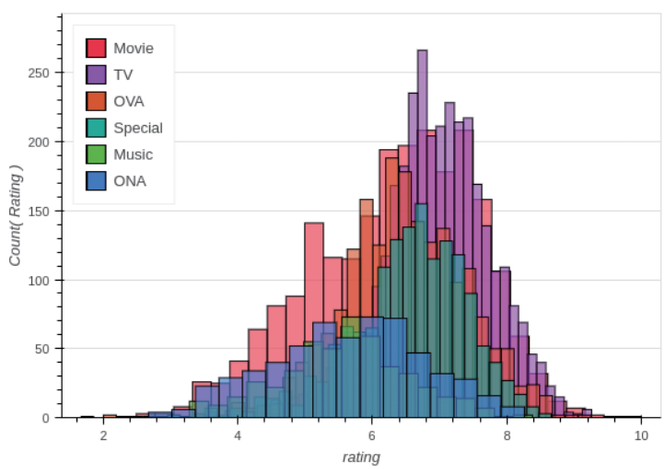
● 図6.7.3：ビン数を設定したヒストグラム（出力結果）

色分けする

データを色分けし、複数のヒストグラムをひとつの図に重ねて描画できます。キーワード引数colorにグループ化したいデータのラベルを渡します（**リスト6.7.4**と**図6.7.4**）。

● リスト6.7.4：色分けしたヒストグラム

```
p = Histogram(df, values='rating', color='type',
              plot_width=600, plot_height=400)
show(p)
```



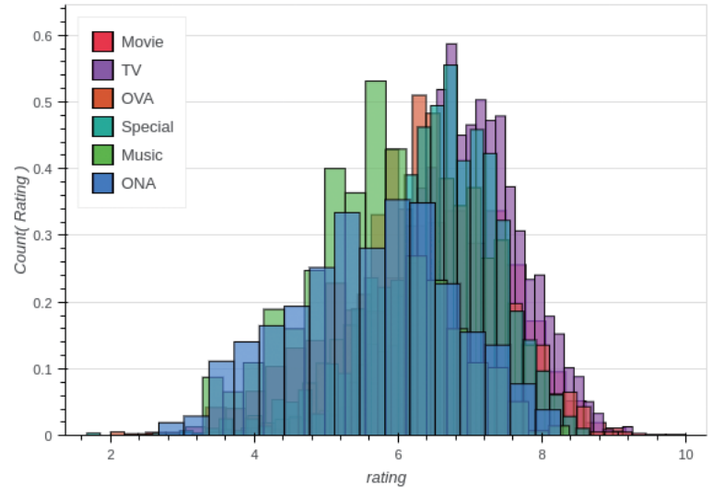
● 図6.7.4：色分けしたヒストグラム（出力結果）

相対度数のヒストグラムを作成する

サンプル数にばらつきがある場合は相対度数のヒストグラムが有効です。キーワード引数densityにTrueを設定することで、相対度数のヒストグラムになります（**リスト6.7.5**と**図6.7.5**）。

● リスト6.7.5：相対度数のヒストグラム

```
p = Histogram(
    df, values='rating', color='type',
    density=True,
    plot_width=600, plot_height=400)
show(p)
```



● 図6.7.5：相対度数のヒストグラム（出力結果）

中レベルのインタフェース

vbar()メソッドを使用し、棒グラフを並べることでヒストグラムを表現します。中レベルのインタフェースには集約機能がないため、numpy.histogramを使用してヒストグラムのデータを算出します。**リスト6.7.6**ではrating列のデータをヒストグラムにし、正規分布に近似した曲線を重ねて描画しています（出力結果は**図6.7.6**）。

● リスト6.7.6：近似曲線を加えたヒストグラム

```
import numpy as np
from bokeh.plotting import figure

bins = 100  # binの数
```

↗