

データの集計

データの集計

- データフレームがカテゴリー変数の列を含む場合、その列の値によってデータを分類し、集計することができます。
 - Pandas には、カテゴリー変数の1列の値によってデータをまとめ、統計量を計算するための `groupby`、カテゴリー変数の2列の値によってデータを分類し、それぞれのデータ数を求めるための `crosstab` (クロス集計)、データ数だけでなくさらに統計量を求めるための `pivot_table` などの集計用関数が用意されています。
- ※ CS2 のデータベースパートでも、SQL の集計用の `GROUP BY` や `count(*)` が出てきましたね。

1列の値による分類と集計(groupby)

- groupbyを用いると、カテゴリデータである1列の各値によってデータを分類し、残りの各列のデータの最大, 最小, 平均などの統計量を求めることができます。

変数 = df.groupby('列名').統計関数()

統計関数: 最大 max(), 最小 min(), 平均 mean(), 数 size() など

※ データフレームが統計関数が適用できない非数値列を含む場合は、あらかじめ削除しておくか、統計関数に numeric_only=True オプションをつける。

1列の値による分類と集計(groupby)

例 (Markdown) `#### Aggregation using groupby`

```
df_no_channel = df.drop(columns='Channel')
df_region_max = df_no_channel.groupby('Region').max()
display(df_region_max)
```

あらかじめ、カテゴリー変数の列である Channel を削除しておく (df_no_channelに代入)。Region列の値ごとに、残りの列の最大値が求まる。

Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
Lisbon	56083	28326	39694	18711	19410	6854
Oporto	32717	25071	67298	60869	38102	5609
Other	112151	73498	92780	36534	40827	47943

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	2	Other	12669	9656	7561	214	2674	1338
1	2	Other	7057	9810	9568	1762	3293	1776
2	2	Other	6353	8808	7684	2405	3516	7844
3	1	Other	13265	1196	4221	6404	507	1788
4	2	Other	22615	5410	7198	3915	1777	5185

カテゴリー変数の1列、たとえば Region列に着目して、

Region列がOtherの顧客

というように、その値をもとにデータを分類してまとめることができるのが `groupby()`。さらにそれに対して `mean()` などの統計関数を適用することで、

Region列がOtherの顧客の各列の平均

というように、各分類グループにおける統計量を一気に求めることができる。

(発展) groupby結果の各値の取得

得られた結果 (今の場合 df_region_max) もデータフレームであるため、それぞれの列や値を、これまでと同様に取り出すことができます。

※ データフレームのindexが Region列の値 (Lisbon, Oporto, Other) になっていることに注意

例 Region列がLisbonの顧客の Milk 列の最大値を取得

```
max_lisbon_milk = df_region_max.at['Lisbon', 'Milk']  
print(max_lisbon_milk)
```

28326

	Channel	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
Region							
Lisbon	2	56083	28326	39694	18711	19410	6854
Oporto	2	32717	25071	67298	60869	38102	5609
Other	2	112151	73498	92780	36534	40827	47943

2列の値によって分類し数をカウント(crosstab)

- カテゴリ変数である2列で全データを分類し、データ数を集計。

変数 = `pd.crosstab(列1のSeries, 列2のSeries)`

DataFrameのメソッドである `groupby` と違って、`crosstab` は Pandas の関数なので注意すること。

(Markdown) `#### Cross tabulation using crosstab`
例

```
df_ct = pd.crosstab(df['Region'], df['Channel'])  
display(df_ct)
```

※ `groupby()` と同様に、結果の `df_ct` はDataFrame。

Channel	1	2
Region		
Lisbon	59	18
Oporto	28	19
Other	211	105

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	2	Other	12669	9656	7561	214	2674	1338
1	2	Other	7057	9810	9568	1762	3293	1776
2	2	Other	6353	8808	7684	2405	3516	7844
3	1	Other	13265	1196	4221	6404	507	1788
4	2	Other	22615	5410	7198	3915	1777	5185

カテゴリー変数の2列、たとえば Region列とChannel列に着目して、

Region列の値が Lisbon でかつ Channel列の値が 2 の顧客の数は？

というように、2列の値の組み合わせでデータをグループ分けして、それぞれのグループのデータ数をカウントすることができるのが `crosstab()`。

2列の値によって分類し数をカウント(crosstab)

- `margins=True` オプションを指定すると、行・列ごとの合計の列 (All) を追加することができます。

```
df_ct2 = pd.crosstab(df['Region'], df['Channel'], margins=True)  
display(df_ct2)
```

Channel	1	2	All
Region			
Lisbon	59	18	77
Oporto	28	19	47
Other	211	105	316
All	298	142	440

2列の値によって分類し数をカウント(crosstab)

- (発展) 和が1になるように正規化することもできます。
 - 列の和が1: `normalize='columns'`, 行の和が1: `normalize='index'`
 - 全セルの和が1: `normalize='all'`

```
df_ct3 = pd.crosstab(df['Region'], df['Channel'], normalize='columns')  
display(df_ct3)
```

列の和が1
になるように
正規化されている

Channel	1	2
Region		
Lisbon	0.197987	0.126761
Oporto	0.093960	0.133803
Other	0.708054	0.739437

(発展) ピボットテーブル

- クロス集計と同様に、カテゴリ変数の2列でデータを分類。集計の際に、単に数を数えるだけでなく、平均、最大、最小などの統計量を求めることができます。

```
変数 = df.pivot_table(index='列1', columns='列2',  
                      values=['列3', ...], aggfunc='統計関数名')
```

- index, columns は複数列名をリストで与えることもできる(とくに、出力データフレームに残しておきたい列はindexで指定しておくとうよい)
- values=をつけないと残りの全列が対象となる
- aggfunc=をつけないと 'mean'。このほか 'max' などの統計量を求める関数を指定可能。なお、関数名のみを文字列で指定し、()をつけないことに注意。

(発展) ピボットテーブル

例 (Markdown) `#### Aggregation using pivot_table`

```
df_pt = df.pivot_table(index='Region', columns='Channel',  
                        values=['Fresh', 'Frozen'],  
                        aggfunc='mean')  
  
display(df_pt)
```

※ groupby() と同様に、結果の df_pt はDataFrame。

	Fresh		Frozen	
Channel	1	2	1	2
Region				
Lisbon	12902.254237	5200.000000	3127.322034	2584.111111
Oporto	11650.535714	7289.789474	5745.035714	1540.578947
Other	13878.052133	9831.504762	3656.900474	1513.200000

データを Region列の値 (Lisbon, Oporto, Other) x Channel列の値 (1, 2) で分類。
 $3 \times 2 = 6$ グループに分類されるので、それぞれのグループでの Fresh列および Frozen列の平均値 (np.mean) を表示している。

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	2	Other	12669	9656	7561	214	2674	1338
1	2	Other	7057	9810	9568	1762	3293	1776
2	2	Other	6353	8808	7684	2405	3516	7844
3	1	Other	13265	1196	4221	6404	507	1788
4	2	Other	22615	5410	7198	3915	1777	5185

たとえば、Region列の値が Lisbon で、かつ Channel列の値が 1 の顧客の Fresh列とFrozen列それぞれの平均値は？

というように、カテゴリー変数の2列で顧客をグループに分類し、それぞれのグループにおける平均や総和、標準偏差などを一気に求められるのが `pivot_table()`