[Python] Pandas 사용법 - 그룹화 및 그룹 함수 (groupby, qcut, cut, transfrom) — 나무늘보의 개발 블로그

노트북:	첫 번째 노트북
만든 날짜:	2020-11-08 오후 9:19
URL:	https://continuous-development.tistory.com/139?category=736681

Python

[Python] Pandas 사용법 - 그룹화 및 그룹 함수 (groupby, qcut, cut, transfrom)

2020. 10. 18. 18:51 수정 삭제 공개

그룹화 (groupby)

```
#사용법
value = dataframe.groupby(dataframe['컬럼'])
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

기본적인 함수들을 import 해주고

In [2]:
```
df = pd.DataFrame({ "학과" : ["컴퓨터","체육교육과","컴퓨터","체육교육과","컴퓨터"],
                    "학년" : [1, 2, 3, 2, 3],
                    "이름" : ["홍길동","김연아","최길동","아이유","신사임당"],
                    "학점" : [1.5, 4.4, 3.7, 4.5, 3.8]})
```

데이터 프레임을 생성해준다.

```
In [10]: dept = df.groupby(df['학과'])
```

그다음 grouping을 해보자. 여기서는 학과를 기준으로 groupby 해주었다. 지금은 묶인 상태이고 여기서 어떻게 사용하냐에 따라 값이 나온다.

```
In [12]: dept.size()
Out[12]: 학과
         체육교육과    2
         컴퓨터      3
         dtype: int64
```

묶어놨던 그룹의 size를 확인해 본다. 해당 값들이 몇 개가 있는지가 나온다.

```
In [11]: dept.mean()
```

Out[11]:

학과	학년	학점
체육교육과	2.000000	4.45
컴퓨터	2.333333	3.00

. mean()이라는 명령어를 통해 평균을 볼 수도 있다.

```
In [14]: dept.sum()
```

Out[14]:

학과	학년	학점
체육교육과	4	8.9
컴퓨터	7	9.0

. sum을 통해 합계를 볼 수도 있다.. 이런 다양한 집계 함수가 가능하다.(max, min, sum, mean 등등)

```
In [15]: df.groupby(['학과','학년']).mean()
```

Out[15]:

학과	학년	학점
체육교육과	2	4.45
컴퓨터	1	1.50
	3	3.75

멀티로 group을 묶을 수도 있다.

예제)

```
In [16]: import seaborn as sns
```

```
In [21]: iris = sns.load_dataset('iris')
         iris
```

Out[21]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

seaborn 을 통해 iris 데이터를 받아온다.

```
In [24]: iris.groupby(iris.species).sum()
```

Out[24]:

species	sepal_length	sepal_width	petal_length	petal_width
setosa	250.3	171.4	73.1	12.3
versicolor	296.8	138.5	213.0	66.3
virginica	329.4	148.7	277.6	101.3

```
In [25]: iris.groupby(iris.species).mean()
```

Out[25]:

species	sepal_length	sepal_width	petal_length	petal_width
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

```
In [26]: iris.groupby(iris.species).min()
```

Out[26]:

species	sepal_length	sepal_width	petal_length	petal_width
setosa	4.3	2.3	1.0	0.1
versicolor	4.9	2.0	3.0	1.0
virginica	4.9	2.2	4.5	1.4

```
In [27]: iris.groupby(iris.species).max()
```

Out[27]:

species	sepal_length	sepal_width	petal_length	petal_width
setosa	5.8	4.4	1.9	0.6
versicolor	7.0	3.4	5.1	1.8
virginica	7.9	3.8	6.9	2.5

이렇게 기본적으로 제공해주는 집계 함수 사용이 가능하다.

In [34]:
```
iris.groupby(iris.species).describe().T
```

Out[34]:

species		setosa	versicolor	virginica
sepal_length	count	50.000000	50.000000	50.000000
	mean	5.006000	5.936000	6.588000
	std	0.352490	0.516171	0.635880
	min	4.300000	4.900000	4.900000
	25%	4.800000	5.600000	6.225000
	50%	5.000000	5.900000	6.500000
	75%	5.200000	6.300000	6.900000
	max	5.800000	7.000000	7.900000
sepal_width	count	50.000000	50.000000	50.000000
	mean	3.428000	2.770000	2.974000
	std	0.379064	0.313798	0.322497
	min	2.300000	2.000000	2.200000
	25%	3.200000	2.525000	2.800000
	50%	3.400000	2.800000	3.000000
	75%	3.675000	3.000000	3.175000
	max	4.400000	3.400000	3.800000
petal_length	count	50.000000	50.000000	50.000000
	mean	1.462000	4.260000	5.552000
	std	0.173664	0.469911	0.551895
	min	1.000000	3.000000	4.500000
	25%	1.400000	4.000000	5.100000
	50%	1.500000	4.350000	5.550000
	75%	1.575000	4.600000	5.875000
	max	1.900000	5.100000	6.900000
petal_width	count	50.000000	50.000000	50.000000
	mean	0.246000	1.326000	2.026000
	std	0.105386	0.197753	0.274650

. **describe()는** 데이터의 개략적인 값을 볼 때 사용된다.

agg()

groupby 된 값에 사용자 정의 함수를 정의해서 사용할 수 있다.
모든 열에 여러 함수를 매핑 : group객체. agg([함수 1, 함수 2, 함수 3,...])
각 열마다 다른 함수를 매핑 : group객체. agg({'열 1': 함수 1, '열 2':함수 2, ...})

In [20]:
```
# 각 종별로 가장 큰 값과 가장 작은 값의 비율을 구한다면?
# agg() - 매개변수로 집계함수를 받는다.
def get_ratio(x):
    return x.max() / x.min()
```

```
In [32]: iris.groupby(iris.species).agg(get_ratio)
```

Out[32]:

species	sepal_length	sepal_width	petal_length	petal_width
setosa	1.348837	1.913043	1.900000	6.000000
versicolor	1.428571	1.700000	1.700000	1.800000
virginica	1.612245	1.727273	1.533333	1.785714

이런 식으로 내가 정의한 함수를 사용할 때는 agg를 사용한다. 열을 지정해주지 않는다면 모든 함수에 적용된다.

apply()

사용자 정의 함수를 적용할 수 있다. agg와의 차이는 agg는 여러 개의 집계 함수를 사용할 수 있는 반면 apply는 하나의 집계 함수만 사용 가능하다.

```
In [36]: # 붓꽃 종별로 가장 큰 꽃잎 길이가 큰 3개의 데이터를 뽑아내는 함수 정의
```

```
In [46]: iris.sort_values(by="petal_length", ascending=False).groupby(iris.species).head(3)
```

Out[46]:

	sepal_length	sepal_width	petal_length	petal_width	species
118	7.7	2.6	6.9	2.3	virginica
122	7.7	2.8	6.7	2.0	virginica
117	7.7	3.8	6.7	2.2	virginica
83	6.0	2.7	5.1	1.6	versicolor
77	6.7	3.0	5.0	1.7	versicolor
52	6.9	3.1	4.9	1.5	versicolor
24	4.8	3.4	1.9	0.2	setosa
44	5.1	3.8	1.9	0.4	setosa
5	5.4	3.9	1.7	0.4	setosa

해당 값에 sort_values를 통해 정렬을 하는데 기준을 petal_length로 잡았다.
그리고 ascending=Fase로 내림차순으로 만든다.
그 후 이것들을 groupby를 하는데 그 기준을 종에 따라 group을 묶는다.
그 뒤 head(3)으로 앞에서 3개의 값을 가져온다.

```
In [50]: def max3_petal_length_func(df):
             return df.sort_values(by="petal_length", ascending=False)[:3]
```

```
In [53]: iris.groupby(iris.species).apply(max3_petal_length_func)
```

Out[53]:

species		sepal_length	sepal_width	petal_length	petal_width	species
setosa	24	4.8	3.4	1.9	0.2	setosa
	44	5.1	3.8	1.9	0.4	setosa
	23	5.1	3.3	1.7	0.5	setosa
versicolor	83	6.0	2.7	5.1	1.6	versicolor
	77	6.7	3.0	5.0	1.7	versicolor
	72	6.3	2.5	4.9	1.5	versicolor
virginica	118	7.7	2.6	6.9	2.3	virginica
	117	7.7	3.8	6.7	2.2	virginica
	122	7.7	2.8	6.7	2.0	virginica

apply 함수를 통해 적용하는 방법이다.

```
In [74]: iris.groupby(iris.species).agg([np.mean,np.sum]).loc[:,['sepal_length','sepal_width']]
```

Out[74]:

	sepal_length		sepal_width	
	mean	sum	mean	sum
species				
setosa	5.006	250.3	3.428	171.4
versicolor	5.936	296.8	2.770	138.5
virginica	6.588	329.4	2.974	148.7

```
In [75]: iris.groupby(iris.species).apply([np.mean,np.sum]).loc[:,['sepal_length','sepal_width']]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-75-524f68dabd53> in <module>()
----> 1 iris.groupby(iris.species).apply([np.mean,np.sum]).loc[:,['sepal_length','sepal_width']]

~₩Anaconda3₩lib₩site-packages₩pandas₩core₩groupby₩groupby.py in apply(self, func, *args, **kwargs)
    666     def apply(self, func, *args, **kwargs):
    667
--> 668         func = self._is_builtin_func(func)
    669
    670         # this is needed so we don't try and wrap strings. If we could

~₩Anaconda3₩lib₩site-packages₩pandas₩core₩base.py in _is_builtin_func(self, arg)
    658         otherwise return the arg
    659         """
--> 660         return self._builtin_table.get(arg, arg)
    661
    662

TypeError: unhashable type: 'list'
```

cut() , quct()

- cut() : 동일 길이로 나누어서 범주를 만들어서 그룹에 대한 통계량
- quct() : 동일 개수로 나누어서 범주를 만들어서 그룹에 대한 통계량 그룹에 대한 통계량

In [59]:
```
def cat3_petal_length(s):
    return pd.qcut(s,3,labels=['소','중','대']).astype(str)
```

함수로 정의해보았다. s는 value고 이것을 3개의 labels(대, 중, 소) 구분 지어 나눈다. 그 후 astype으로 str으로 변환은 해주었다.

In [61]:
```
iris['category'] = iris.groupby(iris.species).petal_length.transform(cat3_petal_length)
iris.head()
```

Out[61]:

	sepal_length	sepal_width	petal_length	petal_width	species	category
0	5.1	3.5	1.4	0.2	setosa	소
1	4.9	3.0	1.4	0.2	setosa	소
2	4.7	3.2	1.3	0.2	setosa	소
3	4.6	3.1	1.5	0.2	setosa	중
4	5.0	3.6	1.4	0.2	setosa	소

그 값을 이렇게 받고 transform을 이용해 데이터 프레임 자체로 변경한 후 칼럼을 추가한다. 그리고 그 함수는 cat3_petal_length를 넣는다.

그러면 위와 같이 데이터 프레임 형태가 바뀌고 소중대 가 추가된 걸 볼 수 있다.

참조 : yganalyst.github.io/data_handling/Pd_13/#3-2-%EC%97% B0% EC%82% B0-%ED% 9B%84-%EA% B8% B0% EC% A1% B4-%EB% 8D% B0% EC% 9D% B4% ED%84% B0% ED%94%84% EB% A0%88% EC% 9E%84% EC% 9D%98-%ED%98%95% ED%83% 9C% EB% A1%9C--transfrom

'**Python**' 카테고리의 다른 글

[Python] 시각화 사용법 - matplotlib을 통한 line plot 그리기(lim,ticks 등등)

[Python] Pandas 사용법 - 피벗 테이블 생성(pivot,pivot_table)

[Python] Pandas 사용법 - 그룹화 및 그룹 함수 (groupby, qcut, cut, transfrom)

[Python] Pandas 사용법 - 두가지의 DataFrame 합치기 (merge, join)

[Python] Pandas 사용법 - 다양한 인덱스 함수(reset_index,set_index,sort_index)

[Python] Pandas 사용법 - 인덱싱 접근,데이터 조작, 인덱스조작(loc,iloc)

python apply python groupby python qctu 파이썬 groupby 파이썬 그룹함수

나무늘보스

혼자 끄적끄적하는 블로그 입니다.