Data transformation using .groupby().transform

WRITING EFFICIENT CODE WITH PANDAS

Leonidas Souliotis
PhD Candidate

datacamp

The restaurant dataset

```
| total_bill  | tip   | sex      | smoker   | day   | time      |
|-------------|-------|----------|----------|-------|-----------|
| 16.99       | 1.01  | Female   | No       | Sun   | "Dinner"  |
| 10.34       | 1.66  | Male     | No       | Sun   | "Dinner"  |
```

```python
restaurant_grouped = restaurant.groupby('smoker')
```

```python
print(restaurant_grouped.count())
```

```
|                 | total_bill | tip   | sex    | day    | time   |
|-----------------|------------|-------|--------|--------|--------|
| smoker          |            |       |        |        |        |
| No              | 151        | 151   | 151    | 151    | 151    |
| Yes             | 93         | 93    | 93     | 93     | 93     |
```

Data transformation

```python
zscore = lambda x: (x - x.mean() ) / x.std()
```

```python
restaurant_grouped = restaurant.groupby('time')
restaurant_transformed = restaurant_grouped.transform(zscore)
```

```python
restaurant_transformed.head()
```

```
   total_bill        tip       size
0   -0.416446  -1.457045  -0.692873
1   -1.143855  -1.004475   0.405737
2    0.023282   0.276645   0.405737
3    0.315339   0.144355  -0.692873
4    0.616880   0.353234   1.504347
```

Comparison with native methods

```python
restaurant.groupby('sex').transform(zscore)


mean_female = restaurant.groupby('sex').mean()['total_bill']['Female']
mean_male = restaurant.groupby('sex').mean()['total_bill']['Male']
std_female = restaurant.groupby('sex').std()['total_bill']['Female']
std_male = restaurant.groupby('sex').std()['total_bill']['Male']


for i in range(len(restaurant)):
    if restaurant.iloc[i][2] == 'Female':
        restaurant.iloc[i][0] = (restaurant.iloc[i][0] - mean_female)/std_female
    else:
        restaurant.iloc[i][0] = (restaurant.iloc[i][0] - mean_male)/std_male
```

```
Time using .groupby(): 0.016291141 seconds
Time using native Python: 3.937326908 seconds


Difference in time: 24,068.5145%
```

Let's practice!

WRITING EFFICIENT CODE WITH PANDAS

Missing value imputation using transform()

WRITING EFFICIENT CODE WITH PANDAS

Leonidas Souliotis
PhD Candidate

Counting missing values

```python
prior_counts = restaurant.groupby('time')
['total_bill'].count()
```

```python
missing_counts = restaurant_nan.groupby('time')
['total_bill'].count()
print(prior_counts - missing_counts)
```

```
time
Dinner     32
Lunch      13
Name: total_bill, dtype: int64
```

Missing value imputation

```python
missing_trans = lambda x: x.fillna(x.mean())
```

```python
restaurant_nan_grouped = restaurant_nan.groupby('time')['total_bill']
restaurant_nan_grouped.transform(missing_trans)
```

```
Time using .transform(): 0.00368881225586 sec
```

```
0    20.676573
1    10.340000
2    21.010000
3    23.680000
4    24.590000
5    25.290000
6    20.676573
Name: total_bill, dtype: float64
```

Comparison with native methods

```python
start_time = time.time()
mean_din = restaurant_nan.loc[restaurant_nan.time ==
'Dinner']['total_bill'].mean()
mean_lun = restaurant_nan.loc[restaurant_nan.time ==
'Lunch']['total_bill'].mean()

for row in range(len(restaurant_nan)):
    if restaurant_nan.iloc[row]['time'] == 'Dinner':
        restaurant_nan.loc[row, 'total_time'] = mean_din
    else:
        restaurant_nan.loc[row, 'total_time'] = mean_lun
print("Results from the above operation calculated in %s seconds" % (time.time() - start_time))
```

Time using native Python: 0.172566890717 sec

Difference in time: 4,578.115%

Let's do it!

WRITING EFFICIENT CODE WITH PANDAS

Data filtration using the filter() function

WRITING EFFICIENT CODE WITH PANDAS

Leonidas Souliotis
PhD Candidate

datacamp

Purpose of filter()

Limit results based on an aggregate feature

- Number of missing values

- Mean of a specific feature

- Number of occurrences of the group

Filter using groupby().filter()

```
restaurant_grouped = restaurant.groupby('day')
filter_trans = lambda x : x['total_bill'].mean() > 20
restaurant_filtered = restaurant_grouped.filter(filter_trans)
```

```
Time using .filter() 0.00414085388184 sec
```

```
print(restaurant_filtered['tip'].mean())
```

```
3.11527607362
```

```
print(restaurant['tip'].mean())
```

```
2.9982786885245902
```

Comparison with native methods

```python
t=[restaurant.loc[df['day'] == i]['tip'] for i in restaurant['day'].unique()
    if restaurant.loc[df['day'] == i]['total_bill'].mean()>20]
restaurant_filtered = t[0]
for j in t[1:]:
    restaurant_filtered=restaurant_filtered.append(j,ignore_index=True)
```

```
Time using native Python: 0.00663900375366 sec
```

```
print(restaurant_filtered.mean())
```
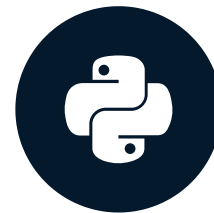
```
3.11527607362
```

```
Difference in time: 60.329341317157024%
```

Let's do it!

WRITING EFFICIENT CODE WITH PANDAS

Congratulations!

WRITING EFFICIENT CODE WITH PANDAS

Leonidas Souliotis
PhD Candidate

datacamp

What you have learned

- Why and how to time operations

- Select targeted rows and columns efficiently

- Select random rows and columns efficiently

- Replace values of a DataFrame efficiently using replace()
  - Replace multiple values using lists

  - Replace multiple values using dictionaries

What you have learned

- Iterate on a DataFrame using the .iterrows() function

- Iterate on a DataFrame using the .apply() function

- Iterate on a DataFrame using pandas optimization

- Iterate on a DataFrame using numpy optimization

- Comparison of the groupby() function compared to native python code
  - When transforming the data group-wise
  - When imputing missing values group-wise
  - When filtering groups with specific characteristics

Congratulations!

WRITING EFFICIENT CODE WITH PANDAS