



MANIPULATING DATAFRAMES WITH PANDAS

Categoricals and groupby



Sales data

```
In [1]: sales = pd.DataFrame(  
    ...: {  
    ...:     'weekday': ['Sun', 'Sun', 'Mon', 'Mon'],  
    ...:     'city': ['Austin', 'Dallas', 'Austin', 'Dallas'],  
    ...:     'bread': [139, 237, 326, 456],  
    ...:     'butter': [20, 45, 70, 98]  
    ...: }  
    ...: )
```

```
In [2]: sales
```

```
Out[2]:
```

	bread	butter	city	weekday
0	139	20	Austin	Sun
1	237	45	Dallas	Sun
2	326	70	Austin	Mon
3	456	98	Dallas	Mon



Boolean filter and count

```
In [3]: sales.loc[sales['weekday'] == 'Sun'].count()
Out[3]:
bread      2
butter     2
city       2
weekday    2
dtype: int64
```



Groupby and count

```
In [4]: sales.groupby('weekday').count()
```

```
Out[4]:
```

	bread	butter	city
weekday			
Mon	2	2	2
Sun	2	2	2



Split-apply-combine

- `sales.groupby('weekday').count()`
 - split by 'weekday'
 - apply `count()` function on each group
 - combine counts per group



Aggregation/Reduction

- Some reducing functions
 - `mean()`
 - `std()`
 - `sum()`
 - `first()`, `last()`
 - `min()`, `max()`



Groupby and sum

```
In [5]: sales.groupby('weekday')['bread'].sum()  
Out[5]:  
weekday  
Mon      782  
Sun      376  
Name: bread, dtype: int64
```



Groupby and sum: multiple columns

```
In [6]: sales.groupby('weekday')[['bread', 'butter']].sum()  
Out[6]:
```

	bread	butter
weekday		
Mon	782	168
Sun	376	65



Groupby and mean: multi-level index

```
In [7]: sales.groupby(['city', 'weekday']).mean()  
Out[7]:
```

		bread	butter
city	weekday		
Austin	Mon	326	70
	Sun	139	20
Dallas	Mon	456	98
	Sun	237	45



Customers

```
In [8]: customers = pd.Series(['Dave', 'Alice', 'Bob', 'Alice'])
```

```
In [9]: customers
```

```
Out[9]:
```

```
0      Dave
```

```
1     Alice
```

```
2       Bob
```

```
3     Alice
```

```
dtype: object
```



Groupby and sum: by series

```
In [10]: sales.groupby(customers)['bread'].sum()
```

```
Out[10]:
```

```
Alice      693
```

```
Bob        326
```

```
Dave       139
```

```
Name: bread, dtype: int64
```



Categorical data

```
In [11]: sales['weekday'].unique()
```

```
Out[11]: array(['Sun', 'Mon'], dtype=object)
```

```
In [12]: sales['weekday'] = sales['weekday'].astype('category')
```

```
In [13]: sales['weekday']
```

```
Out[13]:
```

```
0      Sun
```

```
1      Sun
```

```
2      Mon
```

```
3      Mon
```

```
Name: weekday, dtype: category
```

```
Categories (2, object): [Mon, Sun]
```



Categorical data

- Advantages
 - Uses less memory
 - Speeds up operations like `groupby()`



MANIPULATING DATAFRAMES WITH PANDAS

Let's practice!



MANIPULATING DATAFRAMES WITH PANDAS

Groupby and aggregation



Sales data

```
In [1]: sales = pd.DataFrame(  
    ...: {  
    ...:     'weekday': ['Sun', 'Sun', 'Mon', 'Mon'],  
    ...:     'city': ['Austin', 'Dallas', 'Austin', 'Dallas'],  
    ...:     'bread': [139, 237, 326, 456],  
    ...:     'butter': [20, 45, 70, 98]  
    ...: }  
    ...: )
```

```
In [2]: sales
```

```
Out[2]:
```

	bread	butter	city	weekday
0	139	20	Austin	Sun
1	237	45	Dallas	Sun
2	326	70	Austin	Mon
3	456	98	Dallas	Mon



Review: groupby

```
In [3]: sales.groupby('city')[['bread', 'butter']].max()
```

```
Out[3]:
```

	bread	butter
city		
Austin	326	70
Dallas	456	98



Multiple aggregations

```
In [4]: sales.groupby('city')[['bread', 'butter']].agg(['max', 'sum'])
```

```
Out[4]:
```

	bread		butter	
	max	sum	max	sum
city				
Austin	326	465	70	90
Dallas	456	693	98	143

Aggregation functions

- string names
 - 'sum'
 - 'mean'
 - 'count'



Custom aggregation

```
In [5]: def data_range(series):  
        ....:     return series.max() - series.min()
```



Custom aggregation

```
In [6]: sales.groupby('weekday')[['bread', 'butter']].agg(data_range)
```

```
Out[6]:
```

	bread	butter
weekday		
Mon	130	28
Sun	98	25



Custom aggregation: dictionaries

```
In [7]: sales.groupby(customers)[['bread', 'butter']]  
....:      .agg({'bread': 'sum', 'butter': data_range})
```

```
Out[7]:
```

	butter	bread
Alice	53	693
Bob	0	326
Dave	0	139



MANIPULATING DATAFRAMES WITH PANDAS

Let's practice!



MANIPULATING DATAFRAMES WITH PANDAS

Groupby and transformation



The z-score

```
In [1]: def zscore(series):  
...:     return (series - series.mean()) / series.std()
```



The automobile dataset

```
In [2]: auto = pd.read_csv('auto-mpg.csv')
```

```
In [3]: auto.head()
```

```
Out[3]:
```

	mpg	cyl	displ	hp	weight	accel	yr	origin	name
0	18.0	8	307.0	130	3504	12.0	70	US	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	US	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	US	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	US	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	US	ford torino



MPG z-score

```
In [4]: zscore(auto['mpg']).head()
```

```
Out[4]:
```

	mpg
0	-0.697747
1	-1.082115
2	-0.697747
3	-0.953992
4	-0.825870



MPG z-score by year

```
In [5]: auto.groupby('yr')['mpg'].transform(zscore).head()
```

```
Out[5]:
```

	mpg
0	0.058125
1	-0.503753
2	0.058125
3	-0.316460
4	-0.129168



Apply transformation and aggregation

```
In [6]: def zscore_with_year_and_name(group):  
...:     df = pd.DataFrame(  
...:         {'mpg': zscore(group['mpg']),  
...:          'year': group['yr'],  
...:          'name': group['name']})  
...:     return df
```



Apply transformation and aggregation

```
In [7]: auto.groupby('yr').apply(zscore_with_year_and_name).head()  
Out[7]:
```

	mpg		name	year
0	0.058125	chevrolet	chevelle malibu	70
1	-0.503753		buick skylark 320	70
2	0.058125		plymouth satellite	70
3	-0.316460		amc rebel sst	70
4	-0.129168		ford torino	70



Apply transformation and aggregation

```
In [8]: def zscore_with_year_and_name(group):  
...:     df = pd.DataFrame(  
...:         {'mpg': zscore(group['mpg']),  
...:          'year': group['yr'],  
...:          'name': group['name']})  
...:     return df
```

```
In [9]: auto.groupby('yr').apply(zscore_with_year_and_name).head()
```

```
Out[9]:
```

	mpg		name	year
0	0.058125	chevrolet	chevelle malibu	70
1	-0.503753		buick skylark 320	70
2	0.058125		plymouth satellite	70
3	-0.316460		amc rebel sst	70
4	-0.129168		ford torino	70



MANIPULATING DATAFRAMES WITH PANDAS

Let's practice!



MANIPULATING DATAFRAMES WITH PANDAS

Groupby and filtering



The automobile dataset

```
In [1]: auto = pd.read_csv('auto-mpg.csv')
```

```
In [2]: auto.head()
```

```
Out[2]:
```

	mpg	cyl	displ	hp	weight	accel	yr	origin	name
0	18.0	8	307.0	130	3504	12.0	70	US	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	US	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	US	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	US	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	US	ford torino



Mean MPG by year

```
In [3]: auto.groupby('yr')['mpg'].mean()
```

```
Out[3]:
```

```
yr
```

```
70    17.689655
```

```
71    21.111111
```

```
72    18.714286
```

```
73    17.100000
```

```
74    22.769231
```

```
75    20.266667
```

```
76    21.573529
```

```
77    23.375000
```

```
78    24.061111
```

```
79    25.093103
```

```
80    33.803704
```

```
81    30.185714
```

```
82    32.000000
```

```
Name: mpg, dtype: float64
```



groupby object

```
In [4]: splitting = auto.groupby('yr')
```

```
In [4]: type(splitting)
```

```
Out[4]: pandas.core.groupby.DataFrameGroupBy
```

```
In [5]: type(splitting.groups)
```

```
Out[5]: dict
```

```
In [6]: print(splitting.groups.keys())
```

```
Out[6]: dict_keys([70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82])
```



groupby object: iteration

```
In [7]: for group_name, group in splitting:
....:     avg = group['mpg'].mean()
....:     print(group_name, avg)
```

```
Out[7]:
```

```
70 17.6896551724
71 21.1111111111
72 18.7142857143
73 17.1
74 22.7692307692
75 20.2666666667
76 21.5735294118
77 23.375
78 24.0611111111
79 25.0931034483
80 33.8037037037
81 30.1857142857
82 32.0
```



groupby object: iteration and filtering

```
In [8]: for group_name, group in splitting:
...:     avg = group.loc[group['name'].str.contains('chevrolet'), 'mpg'].mean()
...:     print(group_name, avg)
```

Out[8]:

```
70 15.6666666667
71 20.25
72 15.3333333333
73 14.8333333333
74 18.6666666667
75 17.6666666667
76 23.25
77 20.25
78 23.2333333333
79 21.6666666667
80 30.05
81 23.5
82 29.0
```



groupby object: comprehension

```
In [9]: chevy_means = {year:group.loc[group['name'].str.contains('chevrolet'),'mpg'].mean()  
....:                      for year,group in splitting}
```

```
In [10]: pd.Series(chevy_means)
```

```
Out[10]:
```

```
70    15.666667  
71    20.250000  
72    15.333333  
73    14.833333  
74    18.666667  
75    17.666667  
76    23.250000  
77    20.250000  
78    23.233333  
79    21.666667  
80    30.050000  
81    23.500000  
82    29.000000
```

```
dtype: float64
```



Boolean groupby

```
In [11]: chevy = auto['name'].str.contains('chevrolet')
```

```
In [12]: auto.groupby(['yr', chevy])['mpg'].mean()
```

```
Out[12]:
```

yr	name	
70	False	17.923077
	True	15.666667
71	False	21.260870
	True	20.250000
72	False	19.120000
	True	15.333333
73	False	17.500000
	True	14.833333
74	False	23.304348
	True	18.666667
75	False	20.555556
	True	17.666667
76	False	21.350000
	True	23.250000



MANIPULATING DATAFRAMES WITH PANDAS

Let's practice!