

Lecture 15: Data wrangling in Python

Data wrangling ideas so far

- choose certain rows and columns
- calculate summary statistics
- group rows together
- create new columns
- apply functions across columns
- reshape data by pivoting

These ideas are language-agnostic! The implementation is just a bit different

Titanic data

```
1 import pandas as pd
2 import numpy as np
3 titanic = pd.read_csv("https://raw.githubusercontent.com/pandas-dev/pandas/master/datasets/titanic.csv")
4
5 titanic
```

	PassengerId	Survived	Pclass	...	Fare	Cabin	Embarked
0	1	0	3	...	7.2500	NaN	S
1	2	1	1	...	71.2833	C85	C
2	3	1	3	...	7.9250	NaN	S
3	4	1	1	...	53.1000	C123	S
4	5	0	3	...	8.0500	NaN	S
..
886	887	0	2	...	13.0000	NaN	S
887	888	1	1	...	30.0000	B42	S
888	889	0	3	...	23.4500	NaN	S
889	890	1	1	...	30.0000	C148	C
890	891	0	3	...	7.7500	NaN	Q

[891 rows x 12 columns]

Basic information

```
1 titanic.shape
```

```
(891, 12)
```

```
1 titanic.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age',  
      'SibSp',  
      'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],  
      dtype='object')
```

Choosing a column

```
1 titanic['Pclass']
```

```
0      3
```

```
1      1
```

```
2      3
```

```
3      1
```

```
4      3
```

```
..
```

```
886    2
```

```
887    1
```

```
888    3
```

```
889    1
```

```
890    3
```

```
Name: Pclass, Length: 891, dtype: int64
```

Multiple columns

```
1 titanic[['Pclass', 'Survived']]
```

	Pclass	Survived
0	3	0
1	1	1
2	3	1
3	1	1
4	3	0
..
886	2	0
887	1	1
888	3	0
889	1	1
890	3	0

```
[891 rows x 2 columns]
```

Alternative way to choose columns

```
1 titanic.filter(['Pclass', 'Survived'])
```

	Pclass	Survived
0	3	0
1	1	1
2	3	1
3	1	1
4	3	0
..
886	2	0
887	1	1
888	3	0
889	1	1
890	3	0

```
[891 rows x 2 columns]
```

What would the equivalent R code be?

Alternative way to choose columns

```
1 titanic.filter(['Pclass', 'Survived'])
```

	Pclass	Survived
0	3	0
1	1	1
2	3	1
3	1	1
4	3	0
..
886	2	0
887	1	1
888	3	0
889	1	1
890	3	0

```
[891 rows x 2 columns]
```

What would the equivalent R code be?

```
1 titanic |>  
2   select(Pclass, Survived)
```


Choosing rows

Suppose we only want the rows for the first-class passengers:

```
1 titanic[titanic['Pclass'] == 1]
```

	PassengerId	Survived	Pclass	...	Fare		Cabin	Embarked
1	2	1	1	...	71.2833		C85	C
3	4	1	1	...	53.1000		C123	S
6	7	0	1	...	51.8625		E46	S
11	12	1	1	...	26.5500		C103	S
23	24	1	1	...	35.5000		A6	S
..
871	872	1	1	...	52.5542		D35	S
872	873	0	1	...	5.0000	B51 B53	B55	S
879	880	1	1	...	83.1583		C50	C
887	888	1	1	...	30.0000		B42	S
889	890	1	1	...	30.0000		C148	C

```
[216 rows x 12 columns]
```

Multiple conditions

We can also choose only the first class passengers who survived:

```
1 titanic[(titanic['Pclass'] == 1) & (titanic['Survived'] == 1)]
```

	PassengerId	Survived	Pclass	...	Fare	Cabin	Embarked
1	2	1	1	...	71.2833	C85	C
3	4	1	1	...	53.1000	C123	S
11	12	1	1	...	26.5500	C103	S
23	24	1	1	...	35.5000	A6	S
31	32	1	1	...	146.5208	B78	C
..
862	863	1	1	...	25.9292	D17	S
871	872	1	1	...	52.5542	D35	S
879	880	1	1	...	83.1583	C50	C
887	888	1	1	...	30.0000	B42	S
889	890	1	1	...	30.0000	C148	C

```
[136 rows x 12 columns]
```

Alternative syntax

```
1 titanic.query('Pclass == 1 & Survived == 1')
```

	PassengerId	Survived	Pclass	...	Fare	Cabin	Embarked
1	2	1	1	...	71.2833	C85	C
3	4	1	1	...	53.1000	C123	S
11	12	1	1	...	26.5500	C103	S
23	24	1	1	...	35.5000	A6	S
31	32	1	1	...	146.5208	B78	C
..
862	863	1	1	...	25.9292	D17	S
871	872	1	1	...	52.5542	D35	S
879	880	1	1	...	83.1583	C50	C
887	888	1	1	...	30.0000	B42	S
889	890	1	1	...	30.0000	C148	C

```
[136 rows x 12 columns]
```

What would the equivalent R code be?

Alternative syntax

```
1 titanic.query('Pclass == 1 & Survived == 1')
```

	PassengerId	Survived	Pclass	...	Fare	Cabin	Embarked
1	2	1	1	...	71.2833	C85	C
3	4	1	1	...	53.1000	C123	S
11	12	1	1	...	26.5500	C103	S
23	24	1	1	...	35.5000	A6	S
31	32	1	1	...	146.5208	B78	C
..
862	863	1	1	...	25.9292	D17	S
871	872	1	1	...	52.5542	D35	S
879	880	1	1	...	83.1583	C50	C
887	888	1	1	...	30.0000	B42	S
889	890	1	1	...	30.0000	C148	C

[136 rows x 12 columns]

What would the equivalent R code be?

```
1 titanic |>  
2   filter(Pclass == 1 & Survived == 1)
```

Calculating summary statistics

```
1 titanic.agg({'Survived': 'mean'})
```

```
Survived    0.383838  
dtype: float64
```

```
1 titanic.agg({'Survived': np.mean})
```

```
Survived    0.383838  
dtype: float64
```

Multiple summary statistics

```
1 titanic.agg({'Survived': ['mean', 'std']})
```

	Survived
mean	0.383838
std	0.486592

Summary statistics for multiple columns

```
1 titanic.agg({'Survived': ['mean', 'std'], 'Age': ['mean', 'std']})
```

	Survived	Age
mean	0.383838	29.699118
std	0.486592	14.526497

```
1 titanic[['Survived', 'Age']].agg(['mean', 'std'])
```

	Survived	Age
mean	0.383838	29.699118
std	0.486592	14.526497

Grouping and summarizing

```
1 titanic.groupby(by = ['Pclass', 'Sex']).agg({'Survived': 'mean'})
```

		Survived
Pclass	Sex	
1	female	0.968085
	male	0.368852
2	female	0.921053
	male	0.157407
3	female	0.500000
	male	0.135447

```
1 titanic.groupby(by = ['Pclass', 'Sex']).agg(  
2     survival_rate = ('Survived', 'mean'))
```

		survival_rate
Pclass	Sex	
1	female	0.968085
	male	0.368852
2	female	0.921053
	male	0.157407
3	female	0.500000
	male	0.135447

What would the equivalent R code be?

Grouping and summarizing

```
1 titanic.groupby(by = ['Pclass', 'Sex']).agg(  
2     survival_rate = ('Survived', 'mean'))
```

		survival_rate
Pclass	Sex	
1	female	0.968085
	male	0.368852
2	female	0.921053
	male	0.157407
3	female	0.500000
	male	0.135447

What would the equivalent R code be?

```
1 titanic |>  
2   group_by(Pclass, Sex) |>  
3   summarize(survival_rate = mean(Survived))
```

Class activity

https://sta279-f23.github.io/class_activities/ca_lecture_15.html

