Introduction to Pandas

- · Basic Data Types
- Indexing and Selection
- Filtering

Import both pandas and numpy libraries.

1. Basic Data Types

Both NumPy and Pandas provide useful data types to work with data.

Numpy ndarray

The np.random.rand(n) function generates n number of random values uniformly distributed over [0,1).

- The np.random.seed(0) fix the seed value to 0 so that the randomly generated values are reproducible.
- Generating of random numbers are much more efficient using Numpy.

Exercise

Generate 3 random numbers using Numpy.

Pandas Series

Create a Panda Series using Numpy ndarray.

A Pandas Series can behave like a list too!

Question:

Since both Numpy ndarray and Pandas Series behave like a list. Why do we need to use Panda Series?

• Pandas Series contains not only data but also an index, aka a label, for each record.

Set Index Names

We can change the index value.

```
In [5]: N s.index = ['a', 'b', 'c', 'd', 'e']

Out[5]: a 0.548814
b 0.715189
c 0.602763
d 0.544883
e 0.423655
dtype: float64
```

Index can be set at creation.

```
In [6]: ► 1 s = pd.Series(r, index = ['a', 'b', 'c', 'd', 'e'])
```

Now we can access data using numeric index or text index.

Set Column Name

Besides index value, we can also assign a name to the column.

Pandas DataFrame

Pandas Series can only contains 1 column of data. Pandas DataFrame allows multiple columns, which is like combining of multiple Pandas Series objects which have same index value.

Create a ndarray of 3 rows and 2 columns with random number.

Create a Pandas DataFrame from above ndarray.

Out[10]:

	0	1	2
0	0.645894	0.437587	0.891773
1	0.963663	0.383442	0.791725
2	0.528895	0.568045	0.925597
3	0.071036	0.087129	0.020218
4	0.832620	0.778157	0.870012

Set Index Labels

Update index labels of the dataframe.

Set Column Names

Update columns names of the dataframe.

0.832620 0.778157 0.870012

Rename Index and/or Name

0.832620

0.778157 0.870012

You may use dataframe.rename() function to rename an index or a column.

- By default, rename() function returns a new DataFrame, i.e. it doesn't affect original DataFrame.
- To modify the original DataFrame directly, add parameter inplace=True.

2. Indexing and Selection

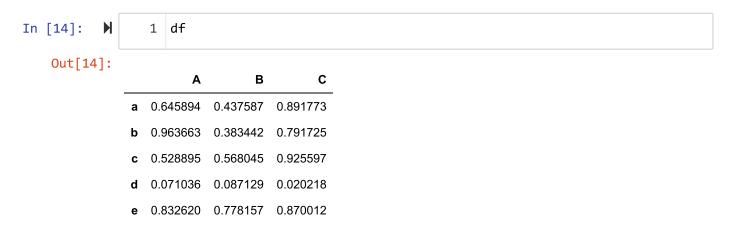
Select Rows by Position

Pandas provides iloc[R,C] selects rows by positions.

The iloc accepts a list row positions, and optionally a list column positions.

df.iloc[row positions, column positions]

Select cell at 1st and 2nd row, including all columns.



Select first 2 rows and first 2 columns.

0.963663 0.383442 0.791725

Select Rows by Label

Pandas provides loc[] function to select rows by labels.

The loc accepts a list row indexers which specifies row indexes, and a list column_indexes which specifies column names.

```
df.loc[row_indexers, column_indexers]
```

Get rows with label b and c.

Select a subset of the dataframe to include row b and c, column B and C.

If row indexer and column indexer are a single value instead of a list, the result is a cell value instead of a DataFrame.

Select Columns

We can now select columns by respective column names.

```
In [21]: M v 1 # df.loc[row_indexes_list, col_indexes_list]
2 df.loc[:, ['A', 'B']]

Out[21]:

A B

a 0.645894 0.437587

b 0.963663 0.383442

c 0.528895 0.568045

d 0.071036 0.087129

e 0.832620 0.778157
```

Here is a shortcut to select multiple columns using [].

Out[22]: pandas.core.frame.DataFrame

Each column is in fact a Pandas Series.

3. Filtering

Max and Min Value

The max() and min() functions return max and min values of each column.

```
In [24]:
          M
                1 print(df)
                2 # df.max()
                  df.min()
                      Α
                                В
                                          C
               0.645894
                         0.437587
                                   0.891773
               0.963663
                         0.383442
                                   0.791725
               0.528895
                         0.568045 0.925597
             d
               0.071036 0.087129 0.020218
               0.832620 0.778157 0.870012
   Out[24]: A
                 0.071036
                 0.087129
             C
                 0.020218
             dtype: float64
```

The idxmax() and idxmin() functions returns the row label whose row value is max or min value.

• To get the row poistion instead of row label, use argmax() and agrmin() instead.

Exercise:

Find the row whose B column is the minimum value of the column.

Filtering

Rows in dataframe can be filtered by list of boolean values.

To check if rows in dataframe fulfills certain condiction, we can use comparison expression with the dataframe.

For example, which rows in column A value are less than 0.5?

We can continue to use above value to filter dataframe.

```
In [30]:  

# df[m]
2 df[ df['A'] < 0.5 ]

Out[30]:

A B C

d 0.071036 0.087129 0.020218
```

Exercise:

Show the rows whose column B value is greater than 0.7?

Join Multiple Conditions

Multiple Conditions can be joined together using & (AND) and | (OR) operators.

Exercise:

Show the rows whose both column A and column B values are greater than 0.5?