The Engineering World #DataScience 1 & 2

May 31, 2018

AKKAL BAHADUR BIST
DATA SCIENTIST AT
KATHMANDU INSTITUTE OF APPLIED SCIENCES (KIAS)
Center for Conservation Biology (CCB)

1 FILTERING AND SELECTING DATA WITH PANDAS

```
In [32]: import numpy as np
         import pandas as pd
         from pandas import Series, DataFrame
```

1.0.1 Selecting and retriving data

```
In [33]: series_obj = Series(np.arange(8), index = ['row 1', 'row 2', 'row 3', 'row 4', 'row 5',
```

```
In [34]: series_obj
```

```
Out[34]: row 1    0
         row 2    1
         row 3    2
         row 4    3
         row 5    4
         row 6    5
         row 7    6
         row 8    7
         dtype: int64
```

```
In [35]: series_obj['row 7']
```

```
Out[35]: 6
```

```
In [36]: series_obj[[0,7]]
```

```
Out[36]: row 1    0
         row 8    7
         dtype: int64
```

```
In [37]: np.random.seed(25)
         DF_obj = DataFrame(np.random.rand(64) .reshape(8,8), index = ['row 1', 'row 2', 'row 3'
In [38]: DF_obj
```

```
Out[38]:        column 1  column 2  column 3  column 4  column 5  column 6  column 7  \
         row 1  0.870124  0.582277  0.278839  0.185911  0.411100  0.117376  0.684969
         row 2  0.556229  0.367080  0.402366  0.113041  0.447031  0.585445  0.161985
         row 3  0.326051  0.699186  0.366395  0.836375  0.481343  0.516502  0.383048
         row 4  0.514244  0.559053  0.034450  0.719930  0.421004  0.436935  0.281701
         row 5  0.669612  0.456069  0.289804  0.525819  0.559242  0.745284  0.828346
         row 6  0.077140  0.644862  0.309258  0.524254  0.958092  0.883201  0.295432
         row 7  0.088702  0.641717  0.132421  0.766486  0.076742  0.331044  0.679852
         row 8  0.655146  0.602120  0.719055  0.415219  0.396542  0.825139  0.712552

                column 8
         row 1  0.437611
         row 2  0.520719
         row 3  0.997541
         row 4  0.900274
         row 5  0.823694
         row 6  0.512376
         row 7  0.509213
         row 8  0.097937
```

```
In [39]: DF_obj.loc[['row 2', 'row 2'], ['column 5', 'column 2']]
```

```
Out[39]:        column 5  column 2
         row 2  0.447031   0.36708
         row 2  0.447031   0.36708
```

1.0.2 Data Slicing

```
In [40]: series_obj['row 3':'row 7']
```

```
Out[40]: row 3    2
         row 4    3
         row 5    4
         row 6    5
         row 7    6
         dtype: int64
```

1.0.3 Comparing with Scalars

```
In [41]: DF_obj < .2
```

```
Out[41]:        column 1  column 2  column 3  column 4  column 5  column 6  column 7  \
         row 1     False     False     False      True     False      True     False
         row 2     False     False     False      True     False     False      True
         row 3     False     False     False     False     False     False     False
         row 4     False     False      True     False     False     False     False
         row 5     False     False     False     False     False     False     False
row 6      True     False     False     False     False     False     False
row 7      True     False      True     False      True     False     False
row 8     False     False     False     False     False     False     False

       column 8
row 1     False
row 2     False
row 3     False
row 4     False
row 5     False
row 6     False
row 7     False
row 8      True
```

1.0.4 FIltering with scalars

```
In [42]: series_obj[series_obj > 6]

Out[42]: row 8    7
         dtype: int64
```

1.0.5 Setting values with scalars

```
In [43]: series_obj ['row 1', 'row 5', 'row 7', 'row 8'] = 8

In [44]: series_obj

Out[44]: row 1    8
         row 2    1
         row 3    2
         row 4    3
         row 5    8
         row 6    5
         row 7    8
         row 8    8
         dtype: int64

In [45]:  DF_obj ['row 1', 'row 5', 'row 8'] = 8

In [46]: DF_obj

Out[46]:        column 1  column 2  column 3  column 4  column 5  column 6  column 7  \
         row 1  0.870124  0.582277  0.278839  0.185911  0.411100  0.117376  0.684969
         row 2  0.556229  0.367080  0.402366  0.113041  0.447031  0.585445  0.161985
         row 3  0.326051  0.699186  0.366395  0.836375  0.481343  0.516502  0.383048
         row 4  0.514244  0.559053  0.034450  0.719930  0.421004  0.436935  0.281701
         row 5  0.669612  0.456069  0.289804  0.525819  0.559242  0.745284  0.828346
         row 6  0.077140  0.644862  0.309258  0.524254  0.958092  0.883201  0.295432
         row 7  0.088702  0.641717  0.132421  0.766486  0.076742  0.331044  0.679852
row 8  0.655146  0.602120  0.719055  0.415219  0.396542  0.825139  0.712552

       column 8  (row 1, row 5, row 8)
row 1  0.437611                      8
row 2  0.520719                      8
row 3  0.997541                      8
row 4  0.900274                      8
row 5  0.823694                      8
row 6  0.512376                      8
row 7  0.509213                      8
row 8  0.097937                      8
```

2 TREATING MISSING VALUES

```
In [47]: missing =np.NaN
         series_obj = Series(['row 1', 'row 2', missing, 'row 4', 'row 5', missing, 'row 6'])
         series_obj
```

```
Out[47]: 0    row 1
         1    row 2
         2      NaN
         3    row 4
         4    row 5
         5      NaN
         6    row 6
         dtype: object
```

```
In [48]: series_obj
```

```
Out[48]: 0    row 1
         1    row 2
         2      NaN
         3    row 4
         4    row 5
         5      NaN
         6    row 6
         dtype: object
```

```
In [49]: series_obj.isnull()
```

```
Out[49]: 0    False
         1    False
         2     True
         3    False
         4    False
         5     True
         6    False
         dtype: bool
```

2.0.1 Filling on the missing values

```
In [50]: np.random.seed(25)
         DF_obj = DataFrame(np.random.randn(36) .reshape(6, 6))
         DF_obj
```

```
Out[50]:          0         1         2         3         4         5
         0  0.228273  1.026890 -0.839585 -0.591182 -0.956888 -0.222326
         1 -0.619915  1.837905 -2.053231  0.868583 -0.920734 -0.232312
         2  2.152957 -1.334661  0.076380 -1.246089  1.202272 -1.049942
         3  1.056610 -0.419678  2.294842 -2.594487  2.822756  0.680889
         4 -1.577693 -1.976254  0.533340 -0.290870 -0.513520  1.982626
         5  0.226001 -1.839905  1.607671  0.388292  0.399732  0.405477
```

```
In [51]: DF_obj.loc[3:5, 0] = missing
         DF_obj.loc[1:4, 5] = missing
```

```
In [52]: DF_obj
```

```
Out[52]:          0         1         2         3         4         5
         0  0.228273  1.026890 -0.839585 -0.591182 -0.956888 -0.222326
         1 -0.619915  1.837905 -2.053231  0.868583 -0.920734       NaN
         2  2.152957 -1.334661  0.076380 -1.246089  1.202272       NaN
         3       NaN -0.419678  2.294842 -2.594487  2.822756       NaN
         4       NaN -1.976254  0.533340 -0.290870 -0.513520       NaN
         5       NaN -1.839905  1.607671  0.388292  0.399732  0.405477
```

```
In [53]: filled_DF = DF_obj.fillna(0)
```

```
In [54]: filled_DF
```

```
Out[54]:          0         1         2         3         4         5
         0  0.228273  1.026890 -0.839585 -0.591182 -0.956888 -0.222326
         1 -0.619915  1.837905 -2.053231  0.868583 -0.920734  0.000000
         2  2.152957 -1.334661  0.076380 -1.246089  1.202272  0.000000
         3  0.000000 -0.419678  2.294842 -2.594487  2.822756  0.000000
         4  0.000000 -1.976254  0.533340 -0.290870 -0.513520  0.000000
         5  0.000000 -1.839905  1.607671  0.388292  0.399732  0.405477
```

```
In [55]: filled_DF = DF_obj.fillna({0:0.1, 5:1.25})
         filled_DF
```

```
Out[55]:          0         1         2         3         4         5
         0  0.228273  1.026890 -0.839585 -0.591182 -0.956888 -0.222326
         1 -0.619915  1.837905 -2.053231  0.868583 -0.920734  1.250000
         2  2.152957 -1.334661  0.076380 -1.246089  1.202272  1.250000
         3  0.100000 -0.419678  2.294842 -2.594487  2.822756  1.250000
         4  0.100000 -1.976254  0.533340 -0.290870 -0.513520  1.250000
         5  0.100000 -1.839905  1.607671  0.388292  0.399732  0.405477
```

```
In [56]: filled_DF = DF_obj.fillna(method = 'ffill')
```

```
In [57]: filled_DF
```

```
Out[57]:           0         1         2         3         4         5
         0  0.228273  1.026890 -0.839585 -0.591182 -0.956888 -0.222326
         1 -0.619915  1.837905 -2.053231  0.868583 -0.920734 -0.222326
         2  2.152957 -1.334661  0.076380 -1.246089  1.202272 -0.222326
         3  2.152957 -0.419678  2.294842 -2.594487  2.822756 -0.222326
         4  2.152957 -1.976254  0.533340 -0.290870 -0.513520 -0.222326
         5  2.152957 -1.839905  1.607671  0.388292  0.399732  0.405477
```

2.0.2 Counting missing values

```
In [58]: np.random.seed(25)
         DF_obj = DataFrame(np.random.randn(36) .reshape(6, 6))
         DF_obj.loc[3:5, 0] = missing
         DF_obj.loc[1:4, 5] = missing
         DF_obj
```

```
Out[58]:           0         1         2         3         4         5
         0  0.228273  1.026890 -0.839585 -0.591182 -0.956888 -0.222326
         1 -0.619915  1.837905 -2.053231  0.868583 -0.920734       NaN
         2  2.152957 -1.334661  0.076380 -1.246089  1.202272       NaN
         3       NaN -0.419678  2.294842 -2.594487  2.822756       NaN
         4       NaN -1.976254  0.533340 -0.290870 -0.513520       NaN
         5       NaN -1.839905  1.607671  0.388292  0.399732  0.405477
```

```
In [59]: DF_obj.isnull().sum()
```

```
Out[59]: 0    3
         1    0
         2    0
         3    0
         4    0
         5    4
         dtype: int64
```

2.0.3 Filtering out missing values

```
In [60]: DF_no_NaN = DF_obj.dropna(axis = 1)
```

```
In [61]: DF_no_NaN
```

```
Out[61]:           1         2         3         4
         0  1.026890 -0.839585 -0.591182 -0.956888
         1  1.837905 -2.053231  0.868583 -0.920734
         2 -1.334661  0.076380 -1.246089  1.202272
         3 -0.419678  2.294842 -2.594487  2.822756
         4 -1.976254  0.533340 -0.290870 -0.513520
         5 -1.839905  1.607671  0.388292  0.399732
```

```
In [62]: DF_obj.dropna (how = 'all')
```

```
Out[62]:           0         1         2         3         4         5
         0  0.228273  1.026890 -0.839585 -0.591182 -0.956888 -0.222326
         1 -0.619915  1.837905 -2.053231  0.868583 -0.920734       NaN
         2  2.152957 -1.334661  0.076380 -1.246089  1.202272       NaN
         3       NaN -0.419678  2.294842 -2.594487  2.822756       NaN
         4       NaN -1.976254  0.533340 -0.290870 -0.513520       NaN
         5       NaN -1.839905  1.607671  0.388292  0.399732  0.405477
```