Handling missing values in Python

September 25, 2019

0.1 Preparation

```
In [20]: import numpy as np
         import pandas as pd
         PREVIOUS_MAX_ROWS=pd.options.display.max_rows
         pd.options.display.max_rows=20
         np.random.seed(123)
         import matplotlib.pyplot as plt
         plt.rc('figure',figsize=(10,6))
         np.set_printoptions(precision=4, suppress=True)
```

0.2 Identify missing strings

```
In [21]: string_data=pd.Series(['annie','bob',np.nan,'donald'])
         string_data
         string_data.isnull()

Out[21]: 0    False
         1    False
         2     True
         3    False
         dtype: bool

In [22]: string_data[0]=None #change the first element to missing
         string_data.isnull()

Out[22]: 0     True
         1    False
         2     True
         3    False
         dtype: bool
```

0.3 Treating missing values

```
In [23]: from numpy import nan as NA
         data=pd.Series([1,NA,3,NA,5])
         data.dropna()  #drop missing observations
```

```
Out[23]: 0    1.0
         2    3.0
         4    5.0
         dtype: float64

In [24]: data[data.notnull()]  #show nonmissing observations

Out[24]: 0    1.0
         2    3.0
         4    5.0
         dtype: float64

In [25]: data=pd.DataFrame([ [1.,6.5,3.], [1.,NA,NA],
                             [NA,NA,NA], [NA,6.5,3.]])
         cleaned=data.dropna() #drops the entire row with missing observations
         data

Out[25]:      0    1    2
         0  1.0  6.5  3.0
         1  1.0  NaN  NaN
         2  NaN  NaN  NaN
         3  NaN  6.5  3.0

In [26]: cleaned

Out[26]:      0    1    2
         0  1.0  6.5  3.0

In [27]: data.dropna(how='all') #do not show rows where all elements are NA

Out[27]:      0    1    2
         0  1.0  6.5  3.0
         1  1.0  NaN  NaN
         3  NaN  6.5  3.0

In [30]: data[4]=NA #add a fourth column with all missing values
         data
         data.dropna(axis=1,how='all')

Out[30]:      0    1    2
         0  1.0  6.5  3.0
         1  1.0  NaN  NaN
         2  NaN  NaN  NaN
         3  NaN  6.5  3.0

In [47]: df=pd.DataFrame(np.random.randn(7,3)) # a 7x3 matrix with random values
         df.iloc[:4,1]=NA
         df.iloc[:2,2]=NA
         df
```

```
Out[47]:           0         1         2
         0 -0.314758       NaN       NaN
         1 -1.212523       NaN       NaN
         2  1.150206       NaN  0.181035
         3  1.177862       NaN  1.031114
         4 -1.084568 -1.363472  0.379401
         5 -0.379176  0.642055 -1.977888
         6  0.712265  2.598304 -0.024626

In [48]: df.dropna() #drops rows with any missing value

Out[48]:           0         1         2
         4 -1.084568 -1.363472  0.379401
         5 -0.379176  0.642055 -1.977888
         6  0.712265  2.598304 -0.024626

In [49]: df.dropna(thresh=2) #drops rows with missing values in two columns

Out[49]:           0         1         2
         2  1.150206       NaN  0.181035
         3  1.177862       NaN  1.031114
         4 -1.084568 -1.363472  0.379401
         5 -0.379176  0.642055 -1.977888
         6  0.712265  2.598304 -0.024626
```

0.4 Interpolating missing data

```
In [46]: df.fillna(0) #change missing to zero

Out[46]:           0         1         2
         0 -1.326265  0.000000  0.000000
         1  0.045490  0.000000  0.000000
         2  0.199524  0.000000 -0.831155
         3  1.162204  0.000000 -2.123100
         4  1.039727 -0.403366 -0.126030
         5 -0.837517 -1.605963  1.255237
         6 -0.688869  1.660952  0.807308

In [37]: df.fillna({1:0.5,2:0}) #change column 1's missing to 0.5, column 2's missing to zero

Out[37]:           0         1         2
         0 -1.085631  0.500000  0.000000
         1 -1.506295  0.500000  0.000000
         2 -2.426679  0.500000  1.265936
         3 -0.866740  0.500000 -0.094709
         4  1.491390 -0.638902 -0.443982
         5 -0.434351  2.205930  2.186786
         6  1.004054  0.386186  0.737369
```

```
In [50]: _=df.fillna(0,inplace=True) #sets equal to zero
         df
```

```
Out[50]:           0         1         2
         0 -0.314758  0.000000  0.000000
         1 -1.212523  0.000000  0.000000
         2  1.150206  0.000000  0.181035
         3  1.177862  0.000000  1.031114
         4 -1.084568 -1.363472  0.379401
         5 -0.379176  0.642055 -1.977888
         6  0.712265  2.598304 -0.024626
```

```
In [51]: df=pd.DataFrame(np.random.randn(6,3)) #another random matrix
         df.iloc[2:,1]=NA
         df.iloc[4:,2]=NA
         df
```

```
Out[51]:           0         1         2
         0  0.034142  0.179549 -1.861976
         1  0.426147 -1.605410 -0.427680
         2  1.242870       NaN  0.501249
         3  1.012739       NaN -1.370948
         4 -0.332475       NaN       NaN
         5 -0.275786       NaN       NaN
```

```
In [53]: df.fillna(method='ffill') # fill out the missing values using forward fill
```

```
Out[53]:           0         1         2
         0  0.034142  0.179549 -1.861976
         1  0.426147 -1.605410 -0.427680
         2  1.242870 -1.605410  0.501249
         3  1.012739 -1.605410 -1.370948
         4 -0.332475 -1.605410 -1.370948
         5 -0.275786 -1.605410 -1.370948
```

```
In [54]: df.fillna(method='ffill',limit=2) # forward fill up to two steps
```

```
Out[54]:           0         1         2
         0  0.034142  0.179549 -1.861976
         1  0.426147 -1.605410 -0.427680
         2  1.242870 -1.605410  0.501249
         3  1.012739 -1.605410 -1.370948
         4 -0.332475       NaN -1.370948
         5 -0.275786       NaN -1.370948
```

```
In [56]: data=pd.Series([1.,NA,3.5,NA,7]) # a vector with missing values
         data
```

```
Out[56]: 0    1.0
         1    NaN
```

```
2    3.5
3    NaN
4    7.0
dtype: float64
```

```
In [57]: data.fillna(data.mean()) # replace by mean
```

```
Out[57]: 0    1.000000
         1    3.833333
         2    3.500000
         3    3.833333
         4    7.000000
         dtype: float64
```