Dealing with missing data

A. Identifying missing values in tabular data

In [1]:

```
import pandas as pd
from io import StringIO
import sys

csv_data = \
'''A,B,C,D
1.0,2.0,3.0,4.0
5.0,6.0,,8.0
10.0,11.0,12.0,'''

# If you are using Python 2.7, you need
# to convert the string to unicode:

if (sys.version_info < (3, 0)):
    csv_data = unicode(csv_data)
```

Step 1: Read the csv file as a pandas dataframe

In [2]:

```
df = pd.read_csv(StringIO(csv_data))
df
```

Out[2]:

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

Step 2: Check the number of missing values for the columns

In [3]:

```
df.isnull()
```

Out[3]:

	A	B	C	D
0	False	False	False	False
1	False	False	True	False
2	False	False	False	True

In [2]:

```
df.isnull().sum()
```

Out[2]:

```
A    0
B    0
C    1
D    1
dtype: int64
```

Step 3: access the underlying NumPy array via the `values` attribute

In [4]:

```
df.values
```

Out[4]:

```
array([[ 1.,  2.,  3.,  4.],
       [ 5.,  6., nan,  8.],
       [10., 11., 12., nan]])
```

Step 4: Remove rows from df that contain missing values

In [9]:

```
# remove rows that contain missing values

df.dropna(axis=0)
```

Out[9]:

	A	B	C	D
0	1.0	2.0	3.0	4.0

Step 5: Remove columns from df that contain missing values

In [10]:

```
df.dropna(axis=1)
```

Out[10]:

	A	B
0	1.0	2.0
1	5.0	6.0
2	10.0	11.0

Step 6: Only drop rows where all columns are NaN

In [7]:

```
df.dropna(how='all')
```

Out[7]:

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

Step 7: Drop rows that have less than 3 real values

In [8]:

```
df.dropna(thresh=4)
```

Out[8]:

	A	B	C	D
0	1.0	2.0	3.0	4.0

Step 8: Only drop rows where NaN appear in specific columns (here: 'C')

In [11]:

```
df.dropna(subset=['C'])
```

Out[11]:

	A	B	C	D
0	1.0	2.0	3.0	4.0
2	10.0	11.0	12.0	NaN

B. Imputing missing values

In [12]:

```
# again: our original array
df.values
```

Out[12]:

```
array([[ 1.,  2.,  3.,  4.],
       [ 5.,  6., nan,  8.],
       [10., 11., 12., nan]])
```

Step 1: impute missing values via the column mean

```
from sklearn.impute import SimpleImputer

import numpy as np
```

In [13]:

```
from sklearn.impute import SimpleImputer
import numpy as np

imr = SimpleImputer(missing_values=np.nan, strategy='mean')

    #imr = SimpleImputer(missing_values=np.nan, add_indicator=True,
     #                      strategy='constant', fill_value=0)
imr = imr.fit(df.values)
imputed_data = imr.transform(df.values)
imputed_data
```

Out[13]:

```
array([[ 1. ,  2. ,  3. ,  4. ],
       [ 5. ,  6. ,  7.5,  8. ],
       [10. , 11. , 12. ,  6. ]])
```