In [1]:
```
import numpy as np
import pandas as pd
pd.__version__
```

Out[1]:
```
'1.2.4'
```

一、缺失值的处理

In [2]:
```
df = pd.DataFrame({'a':[1,np.nan,21],'c':[11,14,16]},index=['a','c','b'])
df
```

Out[2]:

	a	c
a	1.0	11
c	NaN	14
b	21.0	16

（1）.首先把不符合要求的值变成缺失值NaN（np.nan）

这一步一般是在读取csv文件时，通过设置na_values参数来使某些不合理的值变成NaN，参考前面的read_csv()

（2）.将所有包含缺失值的行或者列，直接删除，最省事的处理方式

DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)

参数：

- axis：指定沿着哪个轴进行过滤。如果为0/'index'，则沿着0轴；如果为1/'columns'，则沿着1轴。你也可以同时提供两个轴（以列表或者元组的形式）
- how：指定过滤方式。如果为'any'，则如果该label对应的数据中只要有任何NaN，则抛弃该label；如果为'all'，则如果该label对应的数据中必须全部为NaN才抛弃该label。
- thresh：一个整数，要求该label必须有thresh个非NaN才保留下来。它比how的优先级较高。
- subset：一个label的array-like。比如axis=0，则subset为轴 1 上的标签，它指定你考虑哪些列的子集上的NaN
- inplace：一个布尔值。如果为True，则原地修改。否则返回一个新创建的DataFrame

对于Series，其签名为： Series.dropna(axis=0, inplace=False, **kwargs)

In [3]:
```
df.dropna(axis=0)    ## how默认为any，只要这一行有NaN就删除这行
```

Out[3]:

	a	c
a	1.0	11
b	21.0	16

In [4]:
```
df.dropna(axis=1)
```

Out[4]:

	c
a	11
c	14
b	16

（3）.用指定值或者插值方法来填充缺失数据

DataFrame/Series.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None, **kwargs)

参数：

- value：一个标量、字典、Series或者DataFrame。注意：value与method只能指定其中之一，不能同时提供。
  - 如果为标量，则它指定了填充NaN的数据。
  - 如果为Series/dict，则它指定了填充每个index的数据
  - 如果为DataFrame，则它指定了填充每个DataFrame单元的数据
- method：指定填充方式。可以为None，也可以为：
  - 'backfill'/'bfill'：使用下一个可用的有效值来填充（后向填充backward）
  - 'ffill'/'pad'：使用前一个可用的有效值来填充（前向填充forward）
- axis：指定沿着哪个轴进行填充。如果为0/'index'，则沿着0轴；如果为1/'columns'，则沿着1轴
- inplace：一个布尔值。如果为True，则原地修改。否则返回一个新创建的DataFrame
- limit：一个整数。如果method提供了，则当有连续的N个NaN时，只有其中的limit个NaN会被填充（注意：对于前向填充和后向填充，剩余的空缺的位置不同）

- downcast：一个字典，用于类型转换。字典形式为： {label->dtype}，dtype可以为字符串，也可以为np.float64等。

In [5]:
```
df1 = pd.DataFrame({'a':[1,np.nan,21,11],'c':[12,11,np.nan,16]},index=['a','c','b','d'])
df1
```

Out[5]:

	a	c
a	1.0	12.0
c	NaN	11.0
b	21.0	NaN
d	11.0	16.0

In [6]:
```
df1.fillna(value=10)    ##直接将NaN换成数值10
```

Out[6]:

	a	c
a	1.0	12.0
c	10.0	11.0
b	21.0	10.0
d	11.0	16.0

In [7]:
```
df1.fillna({'a':15,'c':10})  ##标签a列用15填充，标签c列用10填充
```

Out[7]:

	a	c
a	1.0	12.0
c	15.0	11.0
b	21.0	10.0
d	11.0	16.0

In [8]:
```
df1.fillna(method='bfill')  ##使用这列的下一个值填充
```

Out[8]:

	a	c
a	1.0	12.0
c	21.0	11.0
b	21.0	16.0
d	11.0	16.0

In [9]:
```
df1.fillna(method='ffill',axis=1) ##使用这行中前一个元素填充，如果没有前一个元素，不变
```

Out[9]:

	a	c
a	1.0	12.0
c	NaN	11.0
b	21.0	21.0
d	11.0	16.0

二、重复值处理

DataFrame.duplicated(*args, **kwargs)
返回一个布尔Series，默认指示哪些行是重复的（重复行标记为True）

参数：

- keep：一个字符串或者False，指示如何标记
  - 'first'：对于重复数据，第一次出现时标记为False，后面出现时标记为True
  - 'last'：对于重复数据，最后一次出现时标记为False，前面出现时标记为True
  - False：对于重复数据，所有出现的地方都标记为True

In [10]:
```
df2 = pd.DataFrame({'k1':['one'] * 3 + ['two'] * 4,'k2':[1, 1, 2, 3, 3, 4, 4]})
df2
```

Out[10]:

	k1	k2
0	one	1
1	one	1
2	one	2
3	two	3
4	two	3
5	two	4
6	two	4

In [11]:
```
s1=df2.duplicated()                 ##重复的数据，默认标记后者为True
s2=df2.duplicated(keep='last')      ##重复的数据，标记前者为True
s1,s2
```

Out[11]:
```
(0    False
 1     True
 2    False
 3    False
 4     True
 5    False
 6     True
 dtype: bool,
 0     True
 1    False
 2    False
 3     True
 4    False
 5     True
 6    False
 dtype: bool)
```

In [12]:
```
df2[-s1]     ##注意，s为True的表示这行跟其他行是重复的，如果要剔除这行，用-s把True和False反转
```

Out[12]:

	k1	k2
0	one	1
2	one	2
3	two	3
5	two	4

In [13]:
```
df2[-s2]
```

Out[13]:

	k1	k2
1	one	1
2	one	2
4	two	3
6	two	4