8 数据清洗

数据准备（第一章练习的解答）

In [ ]:

```
def m_readdata(filename, startline = 2):
    return pd.read_csv(filename, header = startline,
                       usecols = [0,2,3,4,5,6,7,9,10])

bj08 = m_readdata("pm25/Beijing_2008_HourlyPM2.5_created20140325.csv")
bj09 = m_readdata("pm25/Beijing_2009_HourlyPM25_created20140709.csv")
bj10 = m_readdata("pm25/Beijing_2010_HourlyPM25_created20140709.csv")
bj11 = m_readdata("pm25/Beijing_2011_HourlyPM25_created20140709.csv")
bj12 = m_readdata("pm25/Beijing_2012_HourlyPM2.5_created20140325.csv")
bj13 = m_readdata("pm25/Beijing_2013_HourlyPM2.5_created20140325.csv")
bj14 = m_readdata("pm25/Beijing_2014_HourlyPM25_created20150203.csv")
bj15 = m_readdata("pm25/Beijing_2015_HourlyPM25_created20160201.csv",3)
bj16 = m_readdata("pm25/Beijing_2016_HourlyPM25_created20170201.csv",3)
bj17 = m_readdata("pm25/Beijing_2017_HourlyPM25_created20170803.csv",3)

bj = bj08.append([bj09,bj10,bj11,bj12,bj13,bj14,bj15,bj16,bj17])
bj
```

8.1 处理缺失值

8.1.1 系统默认的缺失值设定

系统默认的缺失值

None和np.nan

确定相应数值是否为缺失值

df.isna() # 别名为isnull，反函数为notna

In [ ]:

```
df2.名次.iloc[:3] = None
```

In [ ]:

```
df2.名次.isna()
```

In [ ]:

```
import numpy as np

df2.名次.iloc[:5] = np.nan
df2.名次.isna()
```

None和np.nan的核心区别：能否进行比较

In [ ]:

```
None == None
```

In [ ]:

```
np.nan == np.nan
```

设定inf和-inf是否被认定为缺失值

```
pandas.options.mode.use_inf_as_na
```

In [ ]:

```
pd.options.mode.use_inf_as_na
```

8.1.2 处理自定义缺失值

目前Pandas不支持设定自定义缺失值，因此只能考虑将其替换为系统缺失值

df.replace('自定义缺失值', np.nan)

In [ ]:

```
df2.所在省份.replace("北京", np.nan)
```

In [ ]:

```
# 设定为None后的效果完全不同
df2.所在省份.replace("北京", None)
```

In [ ]:

```
df2na = df2.replace(["北京", 100], [np.nan, np.nan])# 后面的中括号可以简写
df2na
```

8.1.3 标识缺失值案例

标识缺失值

df.isna()

检查相应的数据是否为缺失值

```
df.isnull()
```

In [ ]:

```
df2na.replace(["北京", 100],[np.nan, np.nan]).isna()
```

检查多个单元格的取值是否为指定数值

df.any(

```
    axis : {index (0), columns (1)}
    skipna = True : 检查时是否忽略缺失值
    level = None : 多重索引时指定具体的级别
```

)

df.all(

```
    axis : {index (0), columns (1)}
    skipna = True : 检查时是否忽略缺失值
    level = None : 多重索引时指定具体的级别
```

)

In [ ]:

```
df2na.isna().any(1)
```

In [ ]:

```
df2na[df2na.isna().any(1)]
```

8.1.4 填充缺失值

df.fillna(

```
    value : 用于填充缺失值的数值
        也可以提供dict/Series/DataFrame以进一步指明哪些索引/列会被替换
        不能使用list
    method = None : 有索引时具体的填充方法，向前填充，向后填充等
        limit = None : 指定了method后设定具体的最大填充步长，>此步长不能填充
    axis : {0 or 'index', 1 or 'columns'}
    inplace = False
```

)

在构建新索引的同时完成缺失值的填充任务

```
    df.reindex(labels = None, fill_value = np.NaN)
```

In [ ]:

```
df2.replace(["北京", 100],[np.nan, np.nan]).fillna('未知')
```

In [ ]:

```
df2.replace(["北京", 100, 1, 2, 3], np.nan).fillna(df2.mean())
```

In [ ]:

```
df2.mean()
```

8.1.5 删除缺失值

df.dropna(

```
    axis = 0 : {0 or 'index', 1 or 'columns'}
    how = any : {'any', 'all'}
        any : 任何一个为NA就删除
        all : 所有的都是NA才删除
    thresh = None : 删除的数量阈值，int
    subset : 希望在处理中包括的行/列子集
    inplace = False :
```

)

In [ ]:

```
df = pd.DataFrame([[np.nan, 2, np.nan, 0], [3, 4, np.nan, 1],
        [np.nan, np.nan, np.nan, 5]],
        columns=list('ABCD'))
df
```

In [ ]:

```
df.dropna(axis=1, how='all')
```

8.2 数据查重

8.2.1 标识出重复的行

标识出重复行的意义在于进一步检查重复原因，以便将可能的错误数据加以修改

```
    duplicated
```

In [ ]:

```
df2['dup'] = df2.duplicated(['类型', '所在省份'])
df2
```

利用索引进行重复行标识

df.index.duplicated()

In [ ]:

```
df2.set_index(['类型', '所在省份']).index.duplicated()
```

8.2.2 直接删除重复的行

```
    drop_duplicates : 如果drop_duplicates(subset=""), 则按照指定的行进行去重
    keep='first' / 'last' / False (是否直接删除有重复的所有记录)
```

In [ ]:

```
df2.drop_duplicates(['类型', '所在省份'])
```

In [ ]:

```
df2.drop_duplicates(['类型', '所在省份'], keep = False)
```

利用查重标识结果直接删除

df[~df.duplicated()]

In [ ]:

```
df2[~df2.duplicated(['类型', '所在省份'])]
```

8.3 实战：进一步整理PM2.5数据

要求:

PM2.5数据中数值-999表示缺失，请将这些数据替换为np.nan
基于上述处理结果，删除缺失值记录
在数据中查找到PM2.5数值完全相同的记录
在数据中查找到同一年中PM2.5数值完全相同的记录

9 处理日期时间变量

9.1 建立Timestamp类和Period类

Pandas中和时间有关的类

Class名称	功能/用途	创建方法
Timestamp	表示具体的日期时间(Timestamp)	to_datetime, Timestamp
DatetimeIndex	日期时间(Timestamp)的索引	to_datetime, date_range, bdate_range, DatetimeIndex
Period	表示具体的日期时间范围(Peroid)	Period
PeriodIndex	Period的索引	period_range, PeriodIndex

9.1.1 Timestamp对象

In [ ]:

```
from datetime import datetime # 从datetime包中引入datetime
datetime(2012, 5, 1)
```

In [ ]:

```
pd.Timestamp(datetime(2012, 5, 1))
```