df.describe(include=['O'])

- 输入include=['O']，计算离散型变量的统计特征.
- 可以看出非空数量count，唯一值种类unique，出现最多的类型top和出现次数freq
- df遍历和替换

```
# 把文件中的 ? 替换成 'nan'
# 把文件中 str 类型的值全部转换成小写
temp = 'aa22'
feature_list = train_df.columns.tolist()
for index in train_df.index:
    for feature in feature_list:
        if type(train_df.loc[index, feature]) == type(temp):
            train_df.loc[index, feature] =
train_df.loc[index, feature].lower()
        if train_df.loc[index, feature] == '?':
            # print(train_df.loc[index, feature])
            train_df.loc[index, feature] = np.nan
```

指定列名读入df

```
names=['timestamp','cylinder number','customer','job
number','grain screened','ink color','proof on ctd ink',
      'blade mfg','cylinder division','paper type','ink
type','direct steam','solvent type','type on cylinder',
      'press type','press','unit number','cylinder size','paper
mill location','plating tank','proof cut',
      'viscosity','caliper','ink
temperature','humifity','roughness','blade pressure','varnish
pct','press speed',
      'ink pct','solvent pct','ESA Voltage','ESA
Amperage','wax','hardener','roller durometer','current density',
      'anode space ratio','chrome content','band type']

train_df = pd.read_csv('bands_init.data',names=names)
```

按列数循环df

```
# 从20列到39列 将列属性转换成float类型
i=20
while(i<39):
    count = 0
    for row in train_df[feature_list[i]]:
        try:
            float(row)
        except ValueError:
            train_df.loc[count,feature_list[i]] = np.nan
            pass
        count=count+1

 train_df[feature_list[i]]=train_df[feature_list[i]].astype(np.float64)
    # 输出转换过的列名
    print(feature_list[i])
    i=i+1
```

df替换列值

```
num_encode = {
    'band type': {'noband':0, "band":1},
}
train_df.replace(num_encode,inplace=True)
```

导入数据

pd.read_csv(filename)：从CSV文件导入数据
pd.read_table(filename)：从限定分隔符的文本文件导入数据
pd.read_excel(filename)：从Excel文件导入数据
pd.read_sql(query, connection_object)：从SQL表/库导入数据
pd.read_json(json_string)：从JSON格式的字符串导入数据
pd.read_html(url)：解析URL、字符串或者HTML文件，抽取其中的tables表格
pd.read_clipboard()：从你的粘贴板获取内容，并传给read_table()
pd.DataFrame(dict)：从字典对象导入数据，Key是列名，Value是数据

导出数据

df.to_csv(filename)：导出数据到CSV文件
df.to_excel(filename)：导出数据到Excel文件
df.to_sql(table_name, connection_object)：导出数据到SQL表
df.to_json(filename)：以Json格式导出数据到文本文件

创建测试对象

pd.DataFrame(np.random.rand(20,5))：创建20行5列的随机数组成的DataFrame对象
pd.Series(my_list)：从可迭代对象my_list创建一个Series对象
df.index = pd.date_range('1900/1/30', periods=df.shape[0])：增加一个日期索引

查看、检查数据

df.head(n)：查看DataFrame对象的前n行
df.tail(n)：查看DataFrame对象的最后n行
df.shape()：查看行数和列数 # Windows加括号报错
df.info()：查看索引、数据类型和内存信息
df.columns 查看列
df.index 查看索引
df.describe()查看数值型列的汇总统计会对数字进行统计显示总数最大最小差值
s.value_counts(dropna=False)：查看Series对象的唯一值和计数
df.apply(pd.Series.value_counts)：查看DataFrame对象中每一列的唯一值和计数

数据选取

df[col]：根据列名，并以Series的形式返回列
df[[col1, col2]]：以DataFrame形式返回多列
s.iloc[0]：按位置选取数据 支持索引、切片
s.loc['index_one']：按索引选取数据 没看懂这是什么鬼
df.iloc[0,:]：返回第一行 冒号表示从头到尾，可以指定切片长度
df.iloc[0,0]：返回第一列的第一个元素
df.iloc[:,0]：返回第一列数据

数据清洗

df.columns = ['a','b','c']：重命名列名
pd.isnull().any()：检查DataFrame对象中的空值，并返回一个Boolean数组
pd.notnull().any()：检查DataFrame对象中的非空值，并返回一个Boolean数组
pd[pd.notnull() == True] 过滤所有的空值
pd[pd.列名.notnull() == True] 过滤本列中是空值得数据
df.dropna()：删除所有包含空值的行
df.dropna(axis=1)：删除所有包含空值的列
df.dropna(axis=1,thresh=n)：删除所有小于n个非空值的行
df.fillna(x)：用x替换DataFrame对象中所有的空值
s.astype(float)：将Series中的数据类型更改为float类型
s.replace(1,'one')：用‘one’代替所有等于1的值 测试中将浮点数替换int 整列变成int类型
s.replace([1,3],['one','three'])：用'one'代替1，用'three'代替3

```
df.rename(columns=lambda x: x + 1)：批量更改列名
df.rename(columns={'old_name': 'new_ name'})：选择性更改列名
df.set_index('column_one')：更改索引列
df.rename(index=lambda x: x + 1)：批量重命名索引
```

数据处理

- Filter、Sort和GroupBy

```
df[df[col] > 0.5]：选择col列的值大于0.5的行
df.sort_values(col1)：按照列col1排序数据，默认升序排列
df.sort_values(col2, ascending=False)：按照列col1降序排列数据
df.sort_values([col1,col2], ascending=[True,False])：先按列col1升序排列，后按col2降序排列数据
df.groupby(col)：返回一个按列col进行分组的Groupby对象 .  真的返回个队形地址
df.groupby([col1,col2])：返回一个按多列进行分组的Groupby对象 .
df.groupby(col1)[col2]：返回按列col1进行分组后，列col2的均值 .   还是返回地址
df.pivot_table(index=col1, values=[col2,col3], aggfunc=max)：创建一个按列col1进行分组，并计算col2和col3的最大值的数据透视表
customer_data.pivot_table(index='refer', values='age', aggfunc=[max, min]) .  显示每个渠道的最大最小值
df.groupby(col1).agg(np.mean)：返回按列col1分组的所有列的均值
经常用于按渠道显示每个渠道的平均值，每个渠道的年龄平均值（最大最小不行整条数据）

data.apply(np.mean)：对DataFrame中的每一列应用函数np.mean
data.apply(np.max,axis=1)：对DataFrame中的每一行应用函数np.max
```

- 添加新列
- 根据当前处理结果将结果添加到新的列宗/增加一列
  - frame['test'] = frame.apply(lamubda x: function(x.city, x.year), axis = 1)
  - function是编写的函数

数据合并

```
df1.append(df2)：将df2中的行添加到df1的尾部
df.concat([df1, df2],axis=1)：将df2中的列添加到df1的尾部
df1.join(df2,on=col1,how='inner')：对df1的列和df2的列执行SQL形式的join
```

数据统计

```
df.describe()：查看数据值列的汇总统计
df.mean()：返回所有列的均值
df.corr()：返回列与列之间的相关系数
df.count()：返回每一列中的非空值的个数
df.max()：返回每一列的最大值
df.min()：返回每一列的最小值
df.median()：返回每一列的中位数
df.std()：返回每一列的标准差    离散度
数值越大表示数据越散
```

数据排序

```
df.sort_values(column,ascending=False) #df按照column降序排列

df.sort_values(column,ascending=False).head(1) #取df最大的一行

df.groupby('column1').sum().sort_values('column2',ascending=False).
head(1) #df按c1分组、求和，然后按照c2降序排序，取最大一行
```

merge

- 默认内连接
- left_on、right_on指定用于连接的列
- left_index=True right=true 用航索引来连接
- suffixes:用于指定附加到左右连个DataFrame对象的重叠列名上的字符串

join

- 按索引合并两个没有重复列的DataFrame

- 默认左连接
- 可同时连接两个以上的DataFrame :df.join([df1,df2])

concat

- 默认纵向连接，axis=1会横向连接series 变成DataFrame
- 默认连接方式是outer

1、两个DataFrame纵向合并：pd.concat()

```
cars =pd. concat([cars1,cars2])



cars.reset_index()



#reset_index可以还原索引，从新变为默认的整型索引



#DataFrame.reset_index(level=None, drop=False, inplace=False,
col_level=0, col_fill=”)



#level控制了具体要还原的那个等级的索引



#drop为False则索引列会被还原为普通列，否则会丢失
```

2、两个dataFrame横向合并：axis=1

```
all_data_col = pd.concat([data1,data2],axis=1)
```

3、按照相同的列左右连接：默认内连接

```
pd.merge(all_data,data3,on='subject_id') #subject_id 为两个df的连接列
```

4、按照相同的列左右连接：外连接的方式

```
pd.merge(data1,data2,on='subject_id',how='outer')
```

日期数据处理

pandas有着强大的日期数据处理功能，本期我们来了解下pandas处理日期数据的一些基本功能，主要包括以下三个方面：

- 按日期筛选数据
- 按日期显示数据
- 按日期统计数据

1.读取并整理数据

- 首先引入pandas库

```
import pandas as pd
```

- 从csv文件中读取数据

```
df = pd.read_csv('date.csv', header=None)
print(df.head(2))
             0  1
0  2013-10-24  3
1  2013-10-25  4
```

- 整理数据

```
df.columns = ['date','number']
df['date'] = pd.to_datetime(df['date']) #将数据类型转换为日期类型
df = df.set_index('date') # 将date设置为index
```

- df的行数一共是425行。

查看DataFrame的数据类型

```
print(type(df))
print(df.index)
print(type(df.index))
<class 'pandas.core.frame.DataFrame'>
```

```
DatetimeIndex(['2013-10-24', '2013-10-25', '2013-10-29', '2013-10-
30',
               '2013-11-04', '2013-11-06', '2013-11-08', '2013-11-
12',
               '2013-11-14', '2013-11-25',
               ...
               '2017-01-03', '2017-01-07', '2017-01-14', '2017-01-
17',
               '2017-01-23', '2017-01-25', '2017-01-26', '2017-02-
07',
               '2017-02-14', '2017-02-22'],
              dtype='datetime64[ns]', name='date', length=425,
freq=None)
<class 'pandas.tseries.index.DatetimeIndex'>
```

2. 按日期筛选数据

获取年度数据

```
df['2013']  #获取2013年数据
df['2016':'2017']  #获取2016至2017年数据
```

获取某月数据

```
df['2013-11']  #获取2013年11月数据
```

dataframe结构的数据用区间的方式获取某天的数据

```
df['2013-11-06':'2013-11-06']
```

3. 按日期显示数据

3.1 to_period()方法

- 请注意df.index的数据类型是DatetimeIndex；
- df_peirod的数据类型是PeriodIndex

按月显示

```
df.to_period('M') #按月显示，但不统计
```

按季度显示

```
df.to_period('Q') #按季度显示，但不统计
```

按年度显示

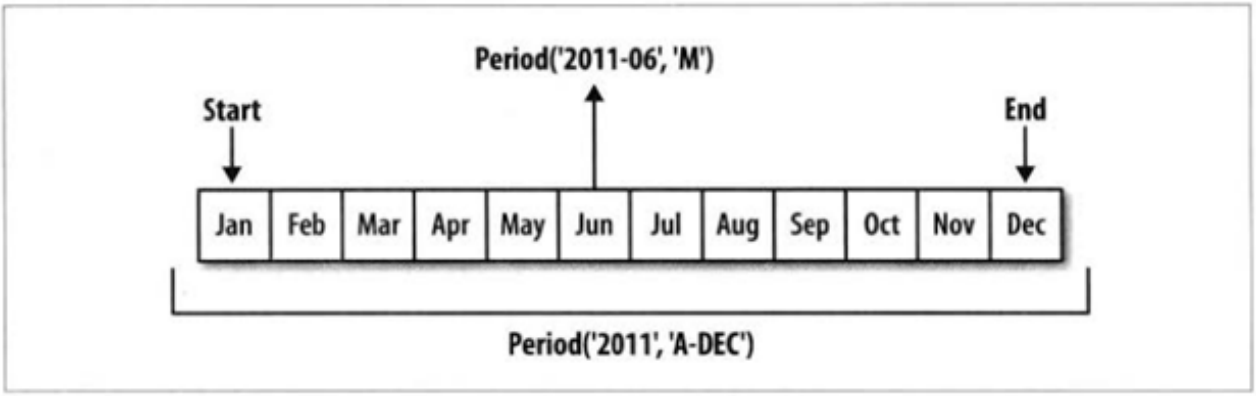
```
df.to_period('A') #按年度显示，但不统计
```

3.2 asfreq()方法

按年度频率显示

```
df.index.asfreq('A') #'A'默认是'A-DEC',其他如'A-JAN'
PeriodIndex(['2013', '2013', '2013', '2013', '2013', '2013',
'2013', '2013',
             '2013', '2013',
             ...
             '2017', '2017', '2017', '2017', '2017', '2017',
'2017', '2017',
             '2017', '2017'],
            dtype='period[A-DEC]', name='date', length=425,
freq='A-DEC')
df_period.index.asfreq('A-JAN') # 'A'默认是'A-DEC',其他如'A-JAN'
PeriodIndex(['2014', '2014', '2014', '2014', '2014', '2014',
'2014', '2014',
             '2014', '2014',
             ...
             '2017', '2017', '2017', '2017', '2017', '2017',
'2017', '2018',
             '2018', '2018'],
            dtype='period[A-JAN]', name='date', length=425,
freq='A-JAN')
```

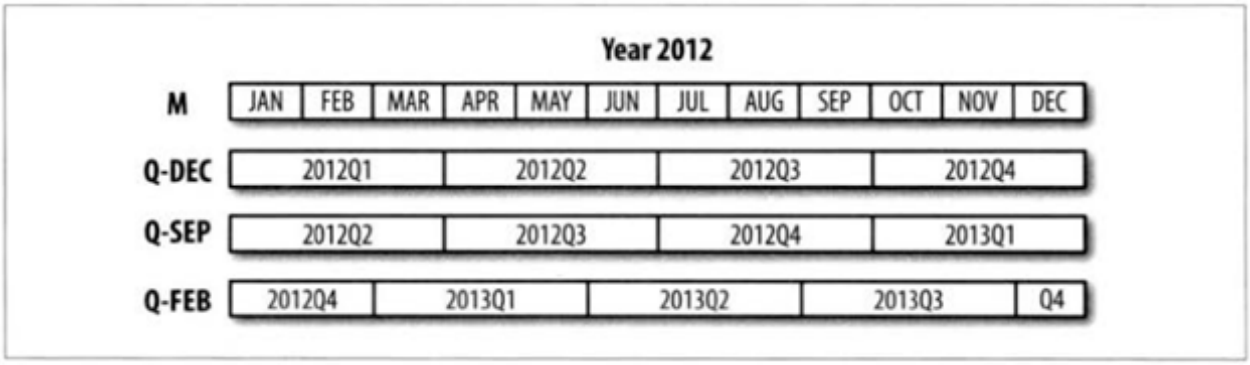
- 按年度频率在不同情形下的显示，可参考下图所示：



按季度频率显示

```
df_period.index.asfreq('Q') # 'Q'默认是'Q-DEC',其他如“Q-SEP”, “Q-FEB”
```

- 按季度频率在不同情形下的显示，可参考下图所示：



按工作日显示

- method 1

```
df_period.index.asfreq('B', how='start') # 按工作日期显示
PeriodIndex(['2013-10-01', '2013-10-01', '2013-10-01', '2013-10-
01',
             '2013-11-01', '2013-11-01', '2013-11-01', '2013-11-
01',
             '2013-11-01', '2013-11-01',
             ...
             '2017-01-02', '2017-01-02', '2017-01-02', '2017-01-
02',
             '2017-01-02', '2017-01-02', '2017-01-02', '2017-02-
01',
             '2017-02-01', '2017-02-01'],
            dtype='period[B]', name='date', length=425, freq='B')
```

- method 2

```
df_period.index.asfreq('B', how='end') # 按工作日期显示
PeriodIndex(['2013-10-31', '2013-10-31', '2013-10-31', '2013-10-
31',
             '2013-11-29', '2013-11-29', '2013-11-29', '2013-11-
29',
             '2013-11-29', '2013-11-29',
             ...
             '2017-01-31', '2017-01-31', '2017-01-31', '2017-01-
31',
             '2017-01-31', '2017-01-31', '2017-01-31', '2017-02-
28',
             '2017-02-28', '2017-02-28'],
            dtype='period[B]', name='date', length=425, freq='B')
```

4. 按日期统计数据

4.1按日期统计数据

按周统计数据

```
print(df.resample('w').sum().head())
# “w”, week
            number
date
2013-10-27     7.0
2013-11-03     3.0
2013-11-10     5.0
2013-11-17     7.0
2013-11-24     NaN
```

按月统计数据

```
print(df.resample('M').sum().head())
#MS是每个月第一天，M是每个月最后一天
```

按季度统计数据

```
print(df.resample('Q').sum().head())
#QS是每个季度第一天为开始日期，“Q是每个季度最后一天”
            number
date
2013-12-31      51
2014-03-31      73
2014-06-30      96
2014-09-30     136
2014-12-31     148
```

按年统计数据

```
print(df.resample('AS').sum())
#AS是每年第一天为开始日期，“A”是每年最后一天
            number
date
2013-01-01      51
2014-01-01     453
2015-01-01     743
2016-01-01    1552
2017-01-01      92
```

关于日期的类型，按参考下图所示来选择合适的分期频率：

Alias	Description
B	business day frequency
C	custom business day frequency (experimental)
D	calendar day frequency
W	weekly frequency
M	month end frequency
SM	semi-month end frequency (15th and end of month)
BM	business month end frequency
CBM	custom business month end frequency
MS	month start frequency
SMS	semi-month start frequency (1st and 15th)
BMS	business month start frequency
CBMS	custom business month start frequency
Q	quarter end frequency
BQ	business quarter endfrequency
QS	quarter start frequency
BQS	business quarter start frequency
A	year end frequency
BA	business year end frequency
AS	year start frequency
BAS	business year start frequency
BH	business hour frequency
H	hourly frequency
T, min	minutely frequency
S	secondly frequency
L, ms	milliseconds
U, us	microseconds
N	nanoseconds

4.2 按日期统计后，按年或季度或月份显示

按年统计并显示

```
print(df.resample('AS').sum().to_period('A'))
# 按年统计并显示
      number
date
2013      51
2014     453
2015     743
2016    1552
2017      92
```

按季度统计并显示

```
print(df.resample('Q').sum().to_period('Q').head())
# 按季度统计并显示
        number
date
2013Q4      51
2014Q1      73
2014Q2      96
2014Q3     136
2014Q4     148
```

按月度统计并显示

```
print(df.resample('M').sum().to_period('M').head())
# 按月度统计并显示
         number
date
2013-10      10
2013-11      14
2013-12      27
2014-01      16
2014-02       4
```

更多参考

https://www.cnblogs.com/valorchang/p/11690888.html

https://blog.csdn.net/zhoukaixuan_zkx/article/details/96151591

pandas学习：https://github.com/SeafyLiang/learn-pandas