数据转换

主要内容	主要方法
删除重复值	duplicated , drop_duplicates
使用函数或映射进行数据转换	map
替代值	replace
重命名轴索引	index , rename
离散化和分箱	cut , qcut , value_counts
检测和过滤异常值	sign
置换和随机抽样	take , sample
计算指标/虚拟变量	get_dummies

- ## 删除重复值

```
#定义数据
df = pd.DataFrame({'k1':['one','two']*3 + ['two'],'k2':[1,1,2,3,3,4,4]})
df
Out[139]:
    k1  k2
0  one   1
1  two   1
2  one   2
3  two   3
4  one   3
5  two   4
6  two   4

#duplicated : 返回布尔值Series,反映每一行是否存在重复
#函数签名 : duplicated(subset=None, keep='first')
df.duplicated()
Out[140]:
0    False
1    False
2    False
3    False
4    False
5    False
6     True
dtype: bool
```

```
#drop_duplicates ：返回duplicated返回数组中False部分
#函数签名 ： drop_duplicates(subset=None, keep='first', inplace=False)
df.drop_duplicates()
Out[142]:
    k1  k2
0  one   1
1  two   1
2  one   2
3  two   3
4  one   3
5  two   4

#增加一列k3
df['k3'] = range(7)
df
Out[154]:
    k1  k2  k3
0  one   1   0
1  two   1   1
2  one   2   2
3  two   3   3
4  one   3   4
5  two   4   5
6  two   4   6
#duplicated和drop_duplicates都默认保留第一次观测到的值，可传入keep = 'last'控制保留最后一个
df.drop_duplicates(subset = ['k1','k2'])
Out[155]:
    k1  k2  k3
0  one   1   0
1  two   1   1
2  one   2   2
3  two   3   3
4  one   3   4
5  two   4   5
df.drop_duplicates(subset = ['k1','k2'],keep = 'last')
Out[156]:
    k1  k2  k3
0  one   1   0
1  two   1   1
2  one   2   2
3  two   3   3
4  one   3   4
6  two   4   6
```

- **使用函数或映射进行数据转换**

```
#定义数据
df = pd.DataFrame({'food':['bacon','pulled pork','bacon','Pastrami','corned
beef','Bacon','pastrami','honey ham','nova lox'],'ounces':[4,3,12,6,7.5,8,3,5,6]})
df
```

```
Out[167]:
          food  ounces
0        bacon     4.0
1  pulled pork     3.0
2        bacon    12.0
3     Pastrami     6.0
4  corned beef     7.5
5        Bacon     8.0
6     pastrami     3.0
7    honey ham     5.0
8     nova lox     6.0

#转换映射表
food_to_animal = {'bacon':'pig','pulled pork':'pig','pastrami':'cow',
                  'corned beef':'cow','honey ham':'pig','nova lox': 'salmon'}

#将df中food列按映射关系生成新列animal
df['animal'] = df['food'].str.lower().map(food_to_animal)
                          or
df['animal'] = df['food'].map(lambda x:food_to_animal[x.lower()])
df
Out[179]:
          food  ounces  animal
0        bacon     4.0     pig
1  pulled pork     3.0     pig
2        bacon    12.0     pig
3     Pastrami     6.0     cow
4  corned beef     7.5     cow
5        Bacon     8.0     pig
6     pastrami     3.0     cow
7    honey ham     5.0     pig
8     nova lox     6.0  salmon
```

- **替代值**

```
函数签名 : replace(['to_replace=None', 'value=None', 'inplace=False', 'limit=None',
                  'regex=False', "method='pad'"])
df
Out[187]:
0       1
1    -999
2       2
3       3
4    -999
5   -1000
6       4
dtype: int64

df.replace([-999,-1000],[0,np.nan])
Out[185]:
```

```
0    1.0
1    0.0
2    2.0
3    3.0
4    0.0
5    NaN
6    4.0
dtype: float64

df.replace({-999:np.nan,-1000:0})
Out[186]:
0    1.0
1    NaN
2    2.0
3    3.0
4    NaN
5    0.0
6    4.0
dtype: float64
```

- **重命名轴索引**

```
#定义数据
df = pd.DataFrame(np.arange(12).reshape((3,4)),index = ['a','b','c'],
                  columns = ['one','two','three','four'])
df
Out[189]:
   one  two  three  four
a    0    1      2     3
b    4    5      6     7
c    8    9     10    11

#将index设置为大写
df.index = df.index.map(lambda x:x.upper())
df
Out[192]:
   one  two  three  four
A    0    1      2     3
B    4    5      6     7
C    8    9     10    11

#将index设置小写，columns设置大写
df.rename(index = str.lower,columns = str.upper)
Out[194]:
   ONE  TWO  THREE  FOUR
a    0    1      2     3
b    4    5      6     7
c    8    9     10    11
```

- **离散化与分箱**

```
age = [20,22,25,27,21,23,37,31,61,45,41,32]
bins = [18,25,35,60,100]
#函数签名 : pd.cut(['x', 'bins', 'right=True', 'labels=None', 'retbins=False',
                'precision=3', 'include_lowest=False', "duplicates='raise'"])
#使用pd.cut方法将age分为(18,25],(26,35],(36,60],(61:]等若干组
cats = pd.cut(age,bins)
cats
Out[214]:
[(18, 25], (18, 25], (18, 25], (25, 35], (18, 25], ..., (25, 35], (60, 100], (35, 60], (35,
60], (25, 35]]
Length: 12
Categories (4, interval[int64]): [(18, 25] < (25, 35] < (35, 60] < (60, 100]]
#对应属性
cats.codes
Out[215]: array([0, 0, 0, 1, 0, 0, 2, 1, 3, 2, 2, 1], dtype=int8)

#对pd.cut的结果中的箱数量计数
pd.value_counts(cats)
Out[216]:
(18, 25]     5
(35, 60]     3
(25, 35]     3
(60, 100]    1
dtype: int64

#right = True --> ( ]   right = False --> [ )
 pd.cut(age,bins,right = False)
Out[217]:
[[18, 25), [18, 25), [25, 35), [25, 35), [18, 25), ..., [25, 35), [60, 100), [35, 60), [35,
60), [25, 35)]
Length: 12
Categories (4, interval[int64]): [[18, 25) < [25, 35) < [35, 60) < [60, 100)]

#传入labels自定义箱名
group_name = ['Youth','YouthAdult','MiddleAged','Senior']
pd.cut(age,bins,labels = group_name)
Out[219]:
[Youth, Youth, Youth, YouthAdult, Youth, ..., YouthAdult, Senior, MiddleAged, MiddleAged,
YouthAdult]
Length: 12
Categories (4, object): [Youth < YouthAdult < MiddleAged < Senior]

#传入整数个箱时会根据最大值与最小值计算等长的箱
pd.value_counts(pd.qcut(list(range(1,21)),4))
Out[235]:
(15.25, 20.0]    5
(10.5, 15.25]    5
(5.75, 10.5]     5
(0.999, 5.75]    5
dtype: int64
```

```
 #qcut 与 cut  :  qcut基于样本分位数进行分箱，取决与数据的分布，使用cut通常不会使每个箱具有相同数据量的数据点，但由于qcut使用样本的分位数，故使用qcut通常能获得等长的箱， percision控制精度
pd.value_counts(pd.cut(np.random.randn(1000),4,precision=2))
Out[236]:
(-1.43, 0.0068]    437
(0.0068, 1.45]     421
(-2.88, -1.43]      72
(1.45, 2.89]        70
dtype: int64
pd.value_counts(pd.qcut(np.random.randn(1000),4,precision=2))
Out[237]:
(0.68, 3.35]     250
(0.012, 0.68]    250
(-0.7, 0.012]    250
(-3.59, -0.7]    250
dtype: int64

#可以自定义分位数
pd.value_counts(pd.cut(np.random.randn(1000),[0,0.1,0.5,0.9,1.0]))
Out[238]:
(0.1, 0.5]    152
(0.5, 0.9]    136
(0.0, 0.1]     46
(0.9, 1.0]     18
dtype: int64
```

- **检测和过滤异常值**

```
data = pd.DataFrame(np.random.randn(1000,4))
data.describe()
Out[265]:
                0            1            2            3
count  1000.000000  1000.000000  1000.000000  1000.000000
mean      0.005654    -0.016033    -0.055155     0.047587
std       1.021494     1.010585     0.995419     1.011725
min      -3.137398    -3.543126    -3.500937    -2.891401
25%      -0.698787    -0.685780    -0.741967    -0.629200
50%      -0.024466     0.020146    -0.026774     0.010484
75%       0.714335     0.678014     0.611139     0.700051
max       3.130173     3.545143     3.185124     3.480585

#使用any方法选出值大于3或小于-3的行
data[(np.abs(data) > 3).any(1)]
Out[266]:
            0         1         2         3
19   0.577422  0.321914 -3.214733  0.934499
204  2.554789 -3.543126  0.955935  0.472027
388 -0.402665  3.545143  1.300517 -1.000192
513 -3.137398  0.075207  0.182248  1.008345
609 -1.350325 -3.199599 -1.052347  0.655328
637 -0.241344  1.152901  0.253493  3.480585
```

```
639 -0.087683  1.083851 -3.500937  0.218803
672  0.157239 -1.314230  3.185124  1.029440
737 -0.376555 -3.387871  0.555219  0.019190
831  3.130173  0.302588  0.430529 -0.524471
856  0.760830 -1.611911  3.102480 -0.548820
943 -0.300680  0.407731 -3.104880  0.845051

#限制-3到3之间的值
data[np.abs(data) > 3] = np.sign(data)*3
data.describe()
Out[268]:
                0            1            2            3
count  1000.000000  1000.000000  1000.000000  1000.000000
mean      0.005662    -0.015447    -0.054622     0.047106
std       1.020690     1.005213     0.991922     1.010205
min      -3.000000    -3.000000    -3.000000    -2.891401
25%      -0.698787    -0.685780    -0.741967    -0.629200
50%      -0.024466     0.020146    -0.026774     0.010484
75%       0.714335     0.678014     0.611139     0.700051
max       3.000000     3.000000     3.000000     3.000000

#np.sign():根据数据中的值的正负分别生成-1和1
np.sign(data).head()
Out[269]:
     0    1    2    3
0  1.0 -1.0 -1.0 -1.0
1 -1.0 -1.0  1.0 -1.0
2 -1.0 -1.0  1.0  1.0
3 -1.0  1.0 -1.0 -1.0
4 -1.0  1.0 -1.0  1.0
```

- **置换和随机抽样**

```
#使用np.random.permutation进行置换
df = pd.DataFrame(np.arange(20).reshape(5,4))
df
Out[271]:
    0   1   2   3
0   0   1   2   3
1   4   5   6   7
2   8   9  10  11
3  12  13  14  15
4  16  17  18  19
#定义随机轴序
sampler = np.random.permutation(5)
sampler
Out[273]: array([4, 2, 3, 1, 0])
#使用take方法或iloc索引置换
df.take(sampler)   OR    df.iloc[sampler]
Out[274]:
    0   1   2   3
```

```
4  16  17  18  19
2   8   9  10  11
3  12  13  14  15
1   4   5   6   7
0   0   1   2   3

#使用sample方法选出一个不含有替代值的随机子集
df.sample(n = 3)
Out[275]:
    0   1   2   3
2   8   9  10  11
1   4   5   6   7
3  12  13  14  15
#若要允许重复选择，则传入replace = True
df.sample(n = 3,replace = True)
Out[279]:
    0   1   2   3
0   0   1   2   3
3  12  13  14  15
3  12  13  14  15
```

- **计算指标/虚拟变量**

```
#定义数据
df = pd.DataFrame({'key':['b','b','a','c','a','b'],'data1':range(6)})
df
Out[287]:
  key  data1
0   b      0
1   b      1
2   a      2
3   c      3
4   a      4
5   b      5

#get_dummies方法生成值为0，1的矩阵
pd.get_dummies(df['key'])
Out[288]:
   a  b  c
0  0  1  0
1  0  1  0
2  1  0  0
3  0  0  1
4  1  0  0
5  0  1  0

#给矩阵列增加前缀
pd.get_dummies(df['key'],prefix = 'key')
Out[291]:
   key_a  key_b  key_c
0      0      1      0
```

```
1      0      1      0
2      1      0      0
3      0      0      1
4      1      0      0
5      0      1      0

#get_dummies 与 cut 相结合
np.random.seed(12345)
values = np.random.rand(10)
values
Out[294]:
array([0.92961609, 0.31637555, 0.18391881, 0.20456028, 0.56772503,
       0.5955447 , 0.96451452, 0.6531771 , 0.74890664, 0.65356987])
pd.get_dummies(pd.cut(values,[0,0.2,0.4,0.6,0.8,1]))
Out[295]:
   (0.0, 0.2]  (0.2, 0.4]  (0.4, 0.6]  (0.6, 0.8]  (0.8, 1.0]
0           0           0           0           0           1
1           0           1           0           0           0
2           1           0           0           0           0
3           0           1           0           0           0
4           0           0           1           0           0
5           0           0           1           0           0
6           0           0           0           0           1
7           0           0           0           1           0
8           0           0           0           1           0
9           0           0           0           1           0
```