```
import numpy as np
import pandas as pd
```

```
iris = 'https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/639388c2cbc2120a14dcf466e85730eb8be498bb/iris.csv'
df_iris = pd.read_csv(iris, sep=',') #tsv
print(type(df_iris))
```

```
    <class 'pandas.core.frame.DataFrame'>
```

```
file_id='11J_owJhixVPaRfruDJ_Gj-pT5G2qZFMd'
link='http://drive.google.com/uc?export=download&id={FILE_ID}'       #当我从kaggle调用别的数据库的时候老是出问题，需要以cvs结尾?
csv_url=link.format(FILE_ID=file_id)
df_uk_rain = pd.read_csv(csv_url)
print(type(df_uk_rain))
```

```
    <class 'pandas.core.frame.DataFrame'>
```

```
df_uk_rain.columns
```

```
    Index(['Water Year', 'Rain (mm) Oct-Sep', 'Outflow (m3/s) Oct-Sep',
           'Rain (mm) Dec-Feb', 'Outflow (m3/s) Dec-Feb', 'Rain (mm) Jun-Aug',
           'Outflow (m3/s) Jun-Aug'],
          dtype='object')
```

```
df_uk_rain.columns = ["water_year","rain_octsep","outflow_octsep",
                      "rain_decfeb","outflow_decfeb","rain_junaug","outflow_junaug"]
```

```
df_uk_rain.columns
```

```
    Index(['water_year', 'rain_octsep', 'outflow_octsep', 'rain_decfeb',
           'outflow_decfeb', 'rain_junaug', 'outflow_junaug'],
          dtype='object')
```

```
df_uk_rain.head()
```

	Water Year	Rain (mm) Oct-Sep	Outflow (m3/s) Oct-Sep	Rain (mm) Dec-Feb	Outflow (m3/s) Dec-Feb	Rain (mm) Jun-Aug	Outflow (m3/s) Jun-Aug
0	1980	1182	5408	292	7248	174	2212
1	1981	1098	5112	257	7316	242	1936
2	1982	1156	5701	330	8567	124	1802
3	1983	993	4265	391	8905	141	1078
4	1984	1182	5364	217	5813	343	4313

```
s=pd.Series([1,3,5,6,8])
print(type(s))
s
```

```
    <class 'pandas.core.series.Series'>
    0    1
    1    3
    2    5
    3    6
    4    8
    dtype: int64
```

```
d=pd.DataFrame({"col1":[1,2,3,4,5,6],"col2":["1","2","3","4","5","6"],"col3":["1",2,3,4,5,None]})
print(d)
```

```
       col1 col2  col3
    0     1    1     1
    1     2    2     2
    2     3    3     3
    3     4    4     4
    4     5    5     5
    5     6    6  None
```

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   col1    6 non-null      int64
 1   col2    6 non-null      object
 2   col3    5 non-null      object
dtypes: int64(1), object(2)
memory usage: 272.0+ bytes
```

```
df_iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
df_iris.head(10)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

```
df_iris.tail(10)
```

	sepal_length	sepal_width	petal_length	petal_width	species
140	6.7	3.1	5.6	2.4	virginica
141	6.9	3.1	5.1	2.3	virginica
142	5.8	2.7	5.1	1.9	virginica
143	6.8	3.2	5.9	2.3	virginica
144	6.7	3.3	5.7	2.5	virginica
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

```
df_iris.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
df_iris.dtypes
```

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

```
print(df_iris.index)
```

```
RangeIndex(start=0, stop=150, step=1)
```

```
df_iris.columns
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

```
attributes =["sepal_length","sepal_width","petal_length","petal_width","class"]
df_iris.columns = attributes
df_iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
df1=df_iris["sepal_length"]
print(df1.head())
print(type(df1))
```

```
0    5.1
1    4.9
2    4.7
3    4.6
4    5.0
Name: sepal_length, dtype: float64
<class 'pandas.core.series.Series'>
```

```
df2=df_iris[["sepal_length"]]
print(df2.head())
print(type(df2))
```

```
   sepal_length
0           5.1
1           4.9
2           4.7
3           4.6
4           5.0
<class 'pandas.core.frame.DataFrame'>
```

```
df3=df_iris[["sepal_length","petal_length"]]
print(df3.head())
print(type(df3))
```

```
   sepal_length  petal_length
0           5.1           1.4
1           4.9           1.4
2           4.7           1.3
3           4.6           1.5
4           5.0           1.4
<class 'pandas.core.frame.DataFrame'>
```

```
df4=df_iris[1:3][["sepal_length","petal_length"]]。#切片是要放在前面的对吧?
print(df4)
print(type(df4))
```

```
   sepal_length  petal_length
1           4.9           1.4
2           4.7           1.3
<class 'pandas.core.frame.DataFrame'>
```

```
df_iris["sepal_length"] >6.0 #什么时候用下划线，什么时候用点?
```

```
0      False
1      False
2      False
3      False
4      False
       ...
145     True
146     True
147     True
148     True
149    False
Name: sepal_length, Length: 150, dtype: bool
```

```
df1=df_iris[(df_iris["sepal_length"] > 6.0)&(df_iris["petal_length"]<5.0)] #所有是true的值都打印出来?
print(df1)
print(type(df1))
```

```
     sepal_length  sepal_width  petal_length  petal_width       class
50            7.0          3.2           4.7          1.4  versicolor
51            6.4          3.2           4.5          1.5  versicolor
52            6.9          3.1           4.9          1.5  versicolor
54            6.5          2.8           4.6          1.5  versicolor
56            6.3          3.3           4.7          1.6  versicolor
58            6.6          2.9           4.6          1.3  versicolor
63            6.1          2.9           4.7          1.4  versicolor
65            6.7          3.1           4.4          1.4  versicolor
68            6.2          2.2           4.5          1.5  versicolor
71            6.1          2.8           4.0          1.3  versicolor
72            6.3          2.5           4.9          1.5  versicolor
73            6.1          2.8           4.7          1.2  versicolor
74            6.4          2.9           4.3          1.3  versicolor
75            6.6          3.0           4.4          1.4  versicolor
76            6.8          2.8           4.8          1.4  versicolor
86            6.7          3.1           4.7          1.5  versicolor
87            6.3          2.3           4.4          1.3  versicolor
91            6.1          3.0           4.6          1.4  versicolor
97            6.2          2.9           4.3          1.3  versicolor
123           6.3          2.7           4.9          1.8   virginica
126           6.2          2.8           4.8          1.8   virginica
127           6.1          3.0           4.9          1.8   virginica
<class 'pandas.core.frame.DataFrame'>
```

```
(df_iris["sepal_length"] > 6.0)&(df_iris["petal_length"]<5.0)
```

```
0      False
1      False
2      False
3      False
4      False
       ...
145    False
146    False
147    False
148    False
149    False
Length: 150, dtype: bool
```

```
df1=df1.reset_index(drop=False). #没有特别明白这句话的含义?
print(df1)
```

```
    level_0  index  sepal_length  sepal_width  petal_length  petal_width  \
0         0     50           7.0          3.2           4.7          1.4
1         1     51           6.4          3.2           4.5          1.5
2         2     52           6.9          3.1           4.9          1.5
3         3     54           6.5          2.8           4.6          1.5
4         4     56           6.3          3.3           4.7          1.6
5         5     58           6.6          2.9           4.6          1.3
6         6     63           6.1          2.9           4.7          1.4
7         7     65           6.7          3.1           4.4          1.4
8         8     68           6.2          2.2           4.5          1.5
9         9     71           6.1          2.8           4.0          1.3
10       10     72           6.3          2.5           4.9          1.5
11       11     73           6.1          2.8           4.7          1.2
12       12     74           6.4          2.9           4.3          1.3
13       13     75           6.6          3.0           4.4          1.4
14       14     76           6.8          2.8           4.8          1.4
15       15     86           6.7          3.1           4.7          1.5
16       16     87           6.3          2.3           4.4          1.3
17       17     91           6.1          3.0           4.6          1.4
18       18     97           6.2          2.9           4.3          1.3
19       19    123           6.3          2.7           4.9          1.8
20       20    126           6.2          2.8           4.8          1.8
21       21    127           6.1          3.0           4.9          1.8

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

```
df_right=pd.DataFrame({"year":np.arange(1980,1990),"rain_cn":np.arange(800,810)})
df_right2=pd.DataFrame({"year":[1991],"rain_cn":[800]})
df_right
```

	year	rain_cn
0	1980	800
1	1981	801
2	1982	802
3	1983	803
4	1984	804
5	1985	805
6	1986	806
7	1987	807
8	1988	808
9	1989	809

```
df_right2
```

	year	rain_cn
0	1991	800

```
df_right3=pd.concat([df_right,df_right2],ignore_index = True)
df_right3
```

	year	rain_cn
0	1980	800
1	1981	801
2	1982	802
3	1983	803
4	1984	804
5	1985	805
6	1986	806
7	1987	807
8	1988	808
9	1989	809
10	1991	800

```
df_right2.columns=["year","rain_cn1"]
df_right2
```

	year	rain_cn1
0	1991	800

```
df_right3=pd.concat([df_right,df_right2],ignore_index = True)
df_right3
```

	year	rain_cn	rain_cn1
0	1980	800.0	NaN
1	1981	801.0	NaN
2	1982	802.0	NaN
3	1983	803.0	NaN
4	1984	804.0	NaN
5	1985	805.0	NaN
6	1986	806.0	NaN
7	1987	807.0	NaN
8	1988	808.0	NaN
9	1989	809.0	NaN
10	1991	NaN	800.0

```
df_right3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   year      11 non-null     int64  
 1   rain_cn   10 non-null     float64
 2   rain_cn1  1 non-null      float64
dtypes: float64(2), int64(1)
memory usage: 392.0 bytes
```

```
df_uk_rain.head(10)
```

	Water Year	Rain (mm) Oct-Sep	Outflow (m3/s) Oct-Sep	Rain (mm) Dec-Feb	Outflow (m3/s) Dec-Feb	Rain (mm) Jun-Aug	Outflow (m3/s) Jun-Aug
0	1980	1182	5408	292	7248	174	2212
1	1981	1098	5112	257	7316	242	1936
2	1982	1156	5701	330	8567	124	1802
3	1983	993	4265	391	8905	141	1078
4	1984	1182	5364	217	5813	343	4313
5	1985	1027	4991	304	7951	229	2595
6	1986	1151	5196	295	7593	267	2826
7	1987	1210	5572	343	8456	294	3154
8	1988	976	4330	309	6465	200	1440
9	1989	1130	4973	470	10520	209	1740

```
df_right3.head(11)
```

	year	rain_cn	rain_cn1
0	1980	800.0	NaN
1	1981	801.0	NaN
2	1982	802.0	NaN
3	1983	803.0	NaN
4	1984	804.0	NaN
5	1985	805.0	NaN
6	1986	806.0	NaN
7	1987	807.0	NaN
8	1988	808.0	NaN
9	1989	809.0	NaN
10	1991	NaN	800.0

```
df_join=pd.merge(df_uk_rain,df_right3,left_on="water_year",right_on="year",how="inner")#第一个uk-rain没有1991，为什么最后一行能够出现
df_join
```

	water_year	rain_octsep	outflow_octsep	rain_decfeb	outflow_decfeb	rain_jun
0	1980	1182	5408	292	7248	
1	1981	1098	5112	257	7316	
2	1982	1156	5701	330	8567	
3	1983	993	4265	391	8905	
4	1984	1182	5364	217	5813	
5	1985	1027	4991	304	7951	
6	1986	1151	5196	295	7593	
7	1987	1210	5572	343	8456	
8	1988	976	4330	309	6465	
9	1989	1130	4973	470	10520	

```
df_uk_rain.groupby(df_uk_rain["water_year"]//10*10)[["rain_octsep","outflow_octsep","rain_decfeb","outflow_decfeb"]].max()
```

water_year	rain_octsep	outflow_octsep	rain_decfeb	outflow_decfeb
1980	1210	5701	470	10520
1990	1268	5824	484	11486
2000	1387	6391	437	10926
2010	1285	5500	350	9615

```
df_uk_rain["water_year"]//10*10
```

```
0     1980
1     1980
2     1980
3     1980
4     1980
5     1980
6     1980
7     1980
8     1980
9     1980
10    1990
11    1990
12    1990
13    1990
14    1990
15    1990
16    1990
17    1990
18    1990
19    1990
20    2000
21    2000
22    2000
23    2000
24    2000
25    2000
26    2000
27    2000
28    2000
29    2000
30    2010
31    2010
32    2010
Name: water_year, dtype: int64
```

```
##Data Preprocessing
#Duplication
#Outlier
#Missing Value
```

```
df_uk_rain[df_uk_rain["rain_octsep"].duplicated(keep=False)]#这就直接将重复的部分删掉了吗?
```

	water_year	rain_octsep	outflow_octsep	rain_decfeb	outflow_decfeb	rain_jun
0	1980	1182	5408	292	7248	
4	1984	1182	5364	217	5813	
6	1986	1151	5196	295	7593	
9	1989	1130	4973	470	10520	
11	1991	1151	4506	246	5493	

```
df_dup=df_uk_rain
df_dup_addon=df_dup.iloc[[0]]
df_dup=pd.concat([df_dup,df_dup_addon],ignore_index=True).    #ignore index?keep=false?一个意思?
df_dup.iloc[[0,-1]]                                             #倒着切? slicing
```

	water_year	rain_octsep	outflow_octsep	rain_decfeb	outflow_decfeb	rain_ju
0	1980	1182	5408	292	7248	

```
df_dup[df_dup.duplicated(keep=False)]  #上下两个式子是一样的功能？只是找出来了？然后有返回值？并没有去除掉？
```

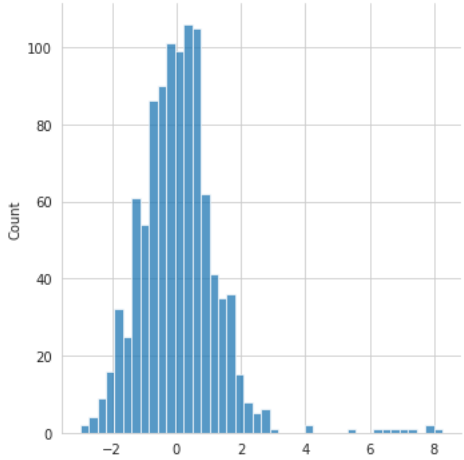
	water_year	rain_octsep	outflow_octsep	rain_decfeb	outflow_decfeb	rain_ju
0	1980	1182	5408	292	7248	

```
df_dedup=df_dup.drop_duplicates()
df_dedup.iloc[[0,-1]]
```

	water_year	rain_octsep	outflow_octsep	rain_decfeb	outflow_decfeb	rain_ju
0	1980	1182	5408	292	7248	

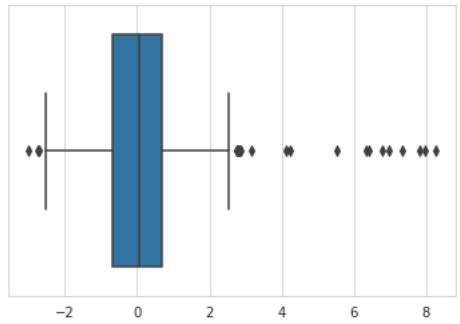
```
import seaborn as sns
dt_outlier=np.concatenate([np.random.randn(1000),np.random.normal(7,1,10)])#选了1000个数，用正太分布函数？7，1，10分别代表的是什么？
sns.set_style("whitegrid")                                  #做出来的图和课件上不太一样，横纵坐标不太一样
sns.displot(dt_outlier)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f573cceda90>
```




```
sns.boxplot(dt_outlier,orient="v" )#orient=v的意义是什么？没有的时候画出来的图是一样的？
```

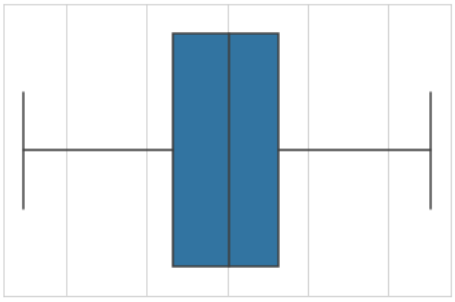
```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_core.py:1326: UserWarning: Verti
  warnings.warn(single_var_warning.format("Vertical", "x"))
<AxesSubplot:>
```



```
def iqr_outlier_rm(dt_input):
  lq,uq=np.percentile(dt_input,[25,75])
  lower_l=lq-1.5*(uq-lq)              #1.5这个数字是哪里来的? 是一个常量吗?
  upper_l=uq+1.5*(uq-lq)
  return dt_input[(dt_input >= lower_l)&(dt_input <= upper_l)]

dt_outlier_ws=iqr_outlier_rm(dt_outlier)
sns.boxplot(dt_outlier_ws,orient="v")
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_core.py:1326: UserWarning: Verti
  warnings.warn(single_var_warning.format("Vertical", "x"))
<AxesSubplot:>
```




```
raw_data = {'name': ['Jason', np.nan, 'Mike', 'Rayman', 'Alex', 'Meimei'],
       'age': [36, np.nan, 36, 18, 36, 16],
       'gender': ['m', np.nan, 'm', np.nan, 'f', 'f'],
       'preMLScore': [1, np.nan, np.nan, 2, 3, 90],
       'postMLScore': [65, np.nan, np.nan, 62, 70, 100]}

# create a dataframe by passing a dictionary
df = pd.DataFrame(raw_data, columns = ['name', 'age', 'gender', 'preMLScore', 'postMLScore'])
df
```

	name	age	gender	preMLScore	postMLScore
0	Jason	36.0	m	1.0	65.0
1	NaN	NaN	NaN	NaN	NaN
2	Mike	36.0	m	NaN	NaN
3	Rayman	18.0	NaN	2.0	62.0
4	Alex	36.0	f	3.0	70.0
5	Meimei	16.0	f	90.0	100.0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   name         5 non-null      object
 1   age          5 non-null      float64
 2   gender       4 non-null      object
 3   preMLScore   4 non-null      float64
 4   postMLScore  4 non-null      float64
dtypes: float64(3), object(2)
memory usage: 368.0+ bytes
```

```
df.isnull()
```

	name	age	gender	preMLScore	postMLScore
0	False	False	False	False	False
1	True	True	True	True	True
2	False	False	False	True	True
3	False	False	True	False	False
4	False	False	False	False	False
5	False	False	False	False	False

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

```
name           1
age            1
gender         2
preMLScore     2
postMLScore    2
dtype: int64
```

```
df.notnull().sum()
```

```
name           5
age            5
gender         4
preMLScore     4
postMLScore    4
dtype: int64
```

```
df.isnull().all()#这个为什么age这一列不是true
```

```
name           False
age            False
gender         False
preMLScore     False
postMLScore    False
dtype: bool
```

```
df.isnull().all(axis=1)
```

```
0    False
1     True
2    False
3    False
4    False
5    False
dtype: bool
```

```
df.isnull().any(axis=0)
```

```
name           True
age            True
gender         True
preMLScore     True
postMLScore    True
dtype: bool
```

```
df
```

	name	age	gender	preMLScore	postMLScore
0	Jason	36.0	m	1.0	65.0
1	NaN	NaN	NaN	NaN	NaN
2	Mike	36.0	m	NaN	NaN
3	Rayman	18.0	NaN	2.0	62.0
4	Alex	36.0	f	3.0	70.0
5	Meimei	16.0	f	90.0	100.0

```
df.dropna(axis=0,how="any")#为什么是how不是row?
```

	name	age	gender	preMLScore	postMLScore
0	Jason	36.0	m	1.0	65.0
4	Alex	36.0	f	3.0	70.0
5	Meimei	16.0	f	90.0	100.0

```
df=df.dropna(how="all",inplace=False)
df
```

	name	age	gender	preMLScore	postMLScore
0	Jason	36.0	m	1.0	65.0
2	Mike	36.0	m	NaN	NaN
3	Rayman	18.0	NaN	2.0	62.0
4	Alex	36.0	f	3.0	70.0
5	Meimei	16.0	f	90.0	100.0

```
df["playgames"]=np.nan
df
```

```
<ipython-input-109-26c78c39d6a9>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  df["playgames"]=np.nan
```

	name	age	gender	preMLScore	postMLScore	playgames
0	Jason	36.0	m	1.0	65.0	NaN
2	Mike	36.0	m	NaN	NaN	NaN
3	Rayman	18.0	NaN	2.0	62.0	NaN
4	Alex	36.0	f	3.0	70.0	NaN
5	Meimei	16.0	f	90.0	100.0	NaN

```
df.dropna(axis=1,how="any")
```

	name	age
0	Jason	36.0
2	Mike	36.0
3	Rayman	18.0
4	Alex	36.0
5	Meimei	16.0

```
df
```

	name	age	gender	preMLScore	postMLScore	playgames
0	Jason	36.0	m	1.0	65.0	NaN
2	Mike	36.0	m	NaN	NaN	NaN
3	Rayman	18.0	NaN	2.0	62.0	NaN
4	Alex	36.0	f	3.0	70.0	NaN
5	Meimei	16.0	f	90.0	100.0	NaN

```
df.dropna(thresh=5)                    #至少有5个有效值?
```

	name	age	gender	preMLScore	postMLScore	playgames
0	Jason	36.0	m	1.0	65.0	NaN

```
df.fillna(0)
```

	name	age	gender	preMLScore	postMLScore	playgames
0	Jason	36.0	m	1.0	65.0	0.0
2	Mike	36.0	m	0.0	0.0	0.0
3	Rayman	18.0	0	2.0	62.0	0.0
4	Alex	36.0	f	3.0	70.0	0.0
5	Meimei	16.0	f	90.0	100.0	0.0

```
df
```

	name	age	gender	preMLScore	postMLScore	playgames
0	Jason	36.0	m	1.0	65.0	NaN
2	Mike	36.0	m	NaN	NaN	NaN
3	Rayman	18.0	NaN	2.0	62.0	NaN
4	Alex	36.0	f	3.0	70.0	NaN
5	Meimei	16.0	f	90.0	100.0	NaN

```
df["gender"].fillna(method="bfill")
```

```
0    m
2    m
3    f
4    f
5    f
Name: gender, dtype: object
```