Data Wrangling with pandas Cheat Sheet

http://pandas.pydata.org

Syntax – Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
      index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

Specify values for each row.

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
          {"a" : [4 ,5, 6],
           "b" : [7, 8, 9],
           "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
          [('d',1),('d',2),('e',2)],
             names=['n','v'])))
```

Create DataFrame with a MultiIndex

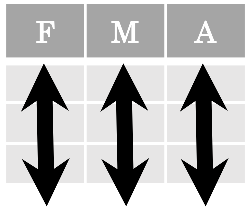
Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
     )
```

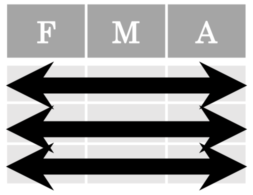
Tidy Data – A foundation for wrangling in pandas

In a tidy data set:



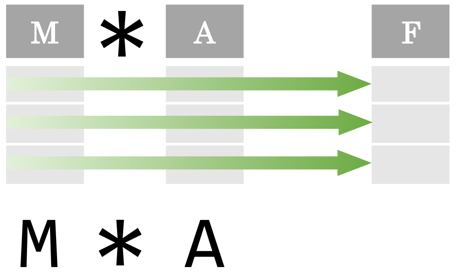
Each **variable** is saved in its own **column**

&

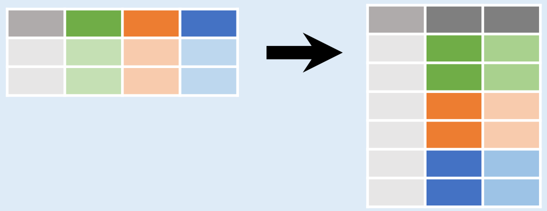


Each **observation** is saved in its own **row**

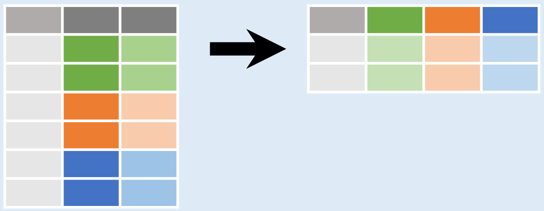
Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.



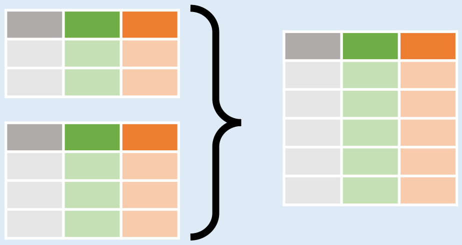
Reshaping Data – Change the layout of a data set



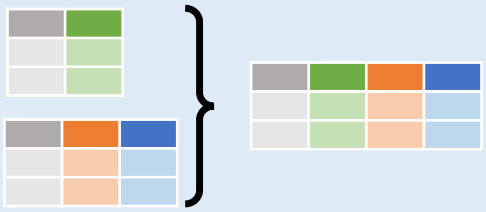
`pd.melt(df)`
Gather columns into rows.



`df.pivot(columns='var', values='val')`
Spread rows into columns.



`pd.concat([df1,df2])`
Append rows of DataFrames



`pd.concat([df1,df2], axis=1)`
Append columns of DataFrames

`df=df.sort_values('mpg')`
Order rows by values of a column (low to high).

`df=df.sort_values('mpg',ascending=False)`
Order rows by values of a column (high to low).

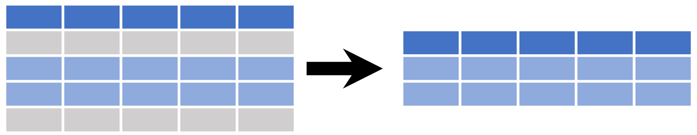
`df=df.rename(columns = {'y':'year'})`
Rename the columns of a DataFrame

`df=df.sort_index()`
Sort the index of a DataFrame

`df=df.reset_index()`
Reset index of DataFrame to row numbers, moving index to columns.

`df=df.drop(['Length','Height'], axis=1)`
Drop columns from DataFrame

Subset Observations (Rows)



`df[df.Length > 7]`
Extract rows that meet logical criteria.

`df.drop_duplicates()`
Remove duplicate rows (only considers columns).

`df.head(n)`
Select first n rows.

`df.tail(n)`
Select last n rows.

`df.sample(frac=0.5)`
Randomly select fraction of rows.

`df.sample(n=10)`
Randomly select n rows.

`df.iloc[10:20]`
Select rows by position.

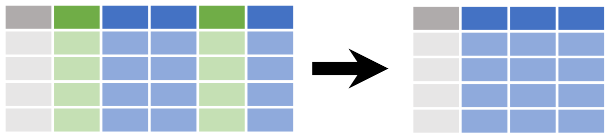
`df.nlargest(n, 'value')`
Select and order top n entries.

`df.nsmallest(n, 'value')`
Select and order bottom n entries.

Logic in Python (and pandas)

`<`	Less than	`!=`	Not equal to
`>`	Greater than	`df.column.isin(`*`values`*`)`	Group membership
`==`	Equals	`pd.isnull(`*`obj`*`)`	Is NaN
`<=`	Less than or equals	`pd.notnull(`*`obj`*`)`	Is not NaN
`>=`	Greater than or equals	`&,|,~,^,df.any(),df.all()`	Logical and, or, not, xor, any, all

Subset Variables (Columns)



`df[['width','length','species']]`
Select multiple columns with specific names.

`df['width']` *or* `df.width`
Select single column with specific name.

`df.filter(regex='`*`regex`*`')`
Select columns whose name matches regular expression *regex*.

regex (Regular Expressions) Examples

`'\.'`	Matches strings containing a period '.'
`'Length$'`	Matches strings ending with word 'Length'
`'^Sepal'`	Matches strings beginning with the word 'Sepal'
`'^x[1-5]$'`	Matches strings beginning with 'x' and ending with 1,2,3,4,5
`''^(?!Species$).*'`	Matches strings except the string 'Species'

`df.loc[:,'x2':'x4']`
Select all columns between x2 and x4 (inclusive).

`df.iloc[:,[1,2,5]]`
Select columns in positions 1, 2 and 5 (first column is 0).

`df.loc[df['a'] > 10, ['a','c']]`
Select rows meeting logical condition, and only the specific columns .

http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants

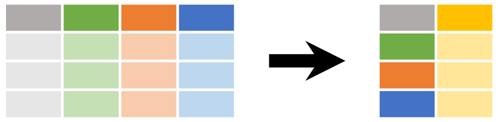
Summarize Data

`df['Length'].value_counts()`
Count number of rows with each unique value of variable

`len(df)`
# of rows in DataFrame.

`len(df['w'].unique())`
# of distinct values in a column.

`df.describe()`
Basic descriptive statistics for each column (or GroupBy)



pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

`sum()`
Sum values of each object.

`count()`
Count non-NA/null values of each object.

`median()`
Median value of each object.

`quantile([0.25,0.75])`
Quantiles of each object.

`apply(`*`function`*`)`
Apply function to each object.

`min()`
Minimum value in each object.

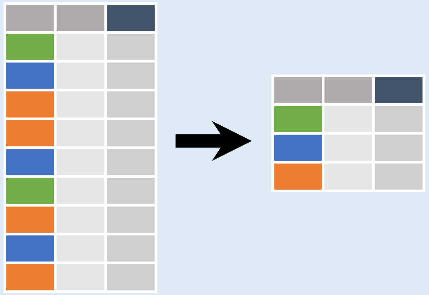
`max()`
Maximum value in each object.

`mean()`
Mean value of each object.

`var()`
Variance of each object.

`std()`
Standard deviation of each object.

Group Data



`df.groupby(by="col")`
Return a GroupBy object, grouped by values in column named "col".

`df.groupby(level="ind")`
Return a GroupBy object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group.
Additional GroupBy functions:

`size()`
Size of each group.

`agg(`*`function`*`)`
Aggregate group using function.

Windows

`df.expanding()`
Return an Expanding object allowing summary functions to be applied cumulatively.

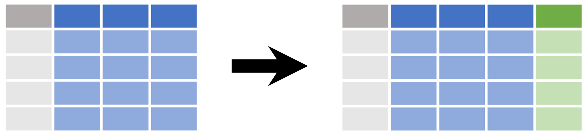
`df.rolling(n)`
Return a Rolling object allowing summary functions to be applied to windows of length n.

Handling Missing Data

`df=df.dropna()`
Drop rows with any column having NA/null data.

`df=df.fillna(value)`
Replace all NA/null data with value.

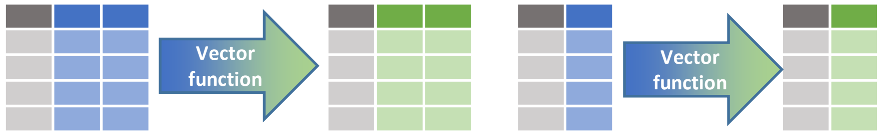
Make New Variables



`df=df.assign(Area=lambda df: df.Length*df.Height)`
Compute and append one or more new columns.

`df['Volume'] = df.Length*df.Height*df.Depth`
Add single column.

`pd.qcut(df.col, n, labels=False)`
Bin column into n buckets.



pandas provides a large set of **vector functions** that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

`max(axis=1)`
Element-wise max.

`clip(lower=-10,upper=10)`
Trim values at input thresholds

`min(axis=1)`
Element-wise min.

`abs()`
Absolute value.

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

`shift(1)`
Copy with values shifted by 1.

`rank(method='dense')`
Ranks with no gaps.

`rank(method='min')`
Ranks. Ties get min rank.

`rank(pct=True)`
Ranks rescaled to interval [0, 1].

`rank(method='first')`
Ranks. Ties go to first value.

`shift(-1)`
Copy with values lagged by 1.

`cumsum()`
Cumulative sum.

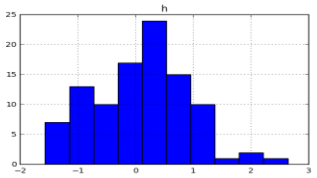
`cummax()`
Cumulative max.

`cummin()`
Cumulative min.

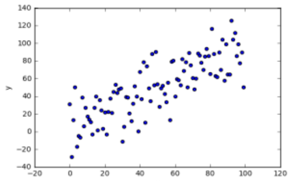
`cumprod()`
Cumulative product.

Plotting

`df.plot.hist()`
Histogram for each column



`df.plot.scatter(x='w',y='h')`
Scatter chart using pairs of points



Combine Data Sets

adf

x1	x2
A	1
B	2
C	3

+

bdf

x1	x3
A	T
B	F
D	T

=

Standard Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

```
pd.merge(adf, bdf,
         how='left', on='x1')
```
Join matching rows from bdf to adf.

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

```
pd.merge(adf, bdf,
         how='right', on='x1')
```
Join matching rows from adf to bdf.

x1	x2	x3
A	1	T
B	2	F

```
pd.merge(adf, bdf,
         how='inner', on='x1')
```
Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

```
pd.merge(adf, bdf,
         how='outer', on='x1')
```
Join data. Retain all values, all rows.

Filtering Joins

x1	x2
A	1
B	2

`adf[adf.x1.isin(bdf.x1)]`
All rows in adf that have a match in bdf.

x1	x2
C	3

`adf[~adf.x1.isin(bdf.x1)]`
All rows in adf that do not have a match in bdf.

ydf

x1	x2
A	1
B	2
C	3

+

zdf

x1	x2
B	2
C	3
D	4

=

Set-like Operations

x1	x2
B	2
C	3

`pd.merge(ydf, zdf)`
Rows that appear in both ydf and zdf (Intersection).

x1	x2
A	1
B	2
C	3
D	4

`pd.merge(ydf, zdf, how='outer')`
Rows that appear in either or both ydf and zdf (Union).

x1	x2
A	1

```
pd.merge(ydf, zdf, how='outer',
         indicator=True)
.query('_merge == "left_only"')
.drop(['_merge'],axis=1)
```
Rows that appear in ydf but not zdf (Setdiff).

http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants

利用pandas 整理数据 Cheat Sheet

http://pandas.pydata.org

wmsby翻译（wmsbywwzyx [at] gmail.com）

创建DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
          {"a" : [4 ,5, 6],
           "b" : [7, 8, 9],
           "c" : [10, 11, 12]},
        index = [1, 2, 3])
```

为每列指定值.

```
df = pd.DataFrame(
     [[4, 7, 10],
      [5, 8, 11],
      [6, 9, 12]],
     index=[1, 2, 3],
     columns=['a', 'b', 'c'])
```

为每行指定值.

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
          {"a" : [4 ,5, 6],
           "b" : [7, 8, 9],
           "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
          [('d',1),('d',2),('e',2)],
             names=['n','v'])))
```

创建有多重索引的DataFrame.

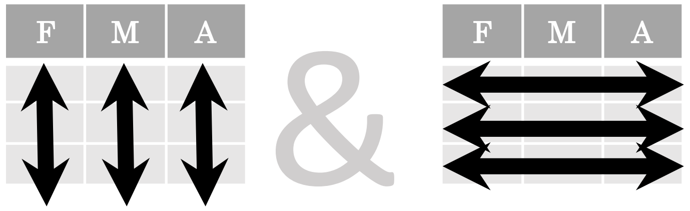
链式方法

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
     )
```

整洁的数据 – 利用pandas整理数据的基础

在整洁的数据中：

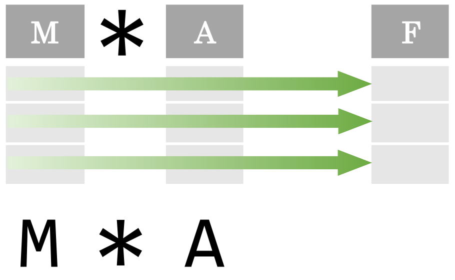


变量（variables）保存在各自的列（column）中

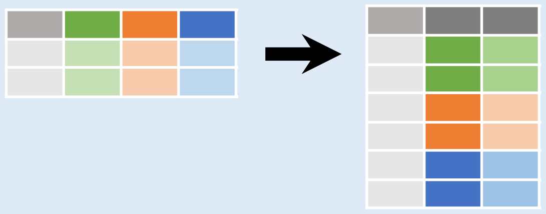
其观测值（observations）保存在各自对应的行(row)里

整洁的数据与pandas矢量化操作相辅相成。当你操作变量时，pandas将自动操作对应变量的每个观测值，非常直观。

译注：变量就是列的意思，观测值就是行的意思。

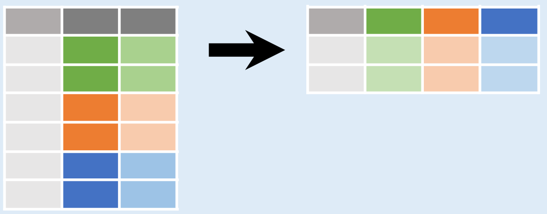


数据重塑Reshaping – 改变数据的形状



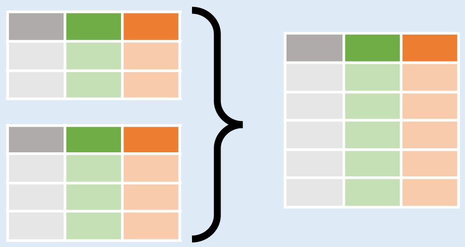
```
pd.melt(df)
```

列 “ 旋转 ” 为行（逆透视）



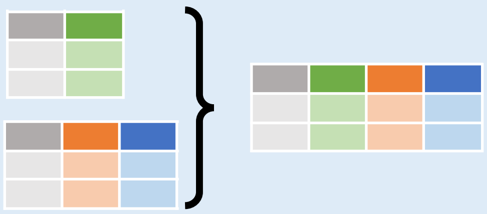
```
df.pivot(columns='var', values='val')
```

行 “ 旋转 ” 为列（透视）



```
pd.concat([df1,df2])
```

按行连接



```
pd.concat([df1,df2], axis=1)
```

按列连接

```
df=df.sort_values('mpg')
```

根据某列的值对行进行排序（升序）

```
df=df.sort_values('mpg',ascending=False)
```

根据某列的值对行进行排序（降序）

```
df=df.rename(columns = {'y':'year'})
```

更改DataFrame的列名

```
df=df.sort_index()
```

对DataFrame的索引进行排序

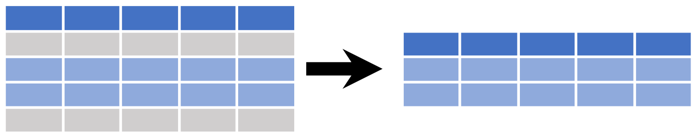
```
df=df.reset_index()
```

将DataFrame的索引移到列里，重置索引为行数

```
df=df.drop(['Length','Height'], axis=1)
```

删除DataFrame的列

选取观测值（选取行）



```
df[df.Length > 7]
```

提取符合标准的行

```
df.drop_duplicates()
```

删除重复行

```
df.head(n)
```

选取前n行

```
df.tail(n)
```

选取后n行

```
df.sample(frac=0.5)
```

随机选取部分(按比例)行

```
df.sample(n=10)
```

随机选取n行

```
df.iloc[10:20]
```

按位置选取行

```
df.nlargest(n, 'value')
```

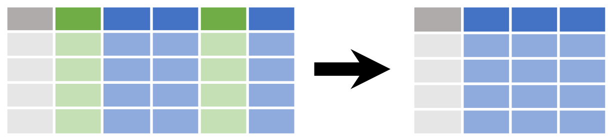
对指定列排序并选择数值最大的n行

```
df.nsmallest(n, 'value')
```

对指定列排序并选择数值最小的n行

Python(和pandas)的逻辑运算			
<	Less than	!=	Not equal to
>	Greater than	df.column.isin(*values*)	Group membership
==	Equals	pd.isnull(*obj*)	Is NaN
<=	Less than or equals	pd.notnull(*obj*)	Is not NaN
>=	Greater than or equals	&,|,~,^,df.any(),df.all()	Logical and, or, not, xor, any, all

选取变量（选取列）



```
df[['width','length','species']]
```

根据列名选取多列

`df['width']` *or* `df.width`

根据列名选取单列

```
df.filter(regex='regex')
```

选取列名匹配正则表达式的列

正则表达式举例	
'\.'	匹配包含'.'的字符串
'Length$'	匹配结尾是'Length'的字符串
'^Sepal'	匹配开头是'Sepal'的字符串
'^x[1-5]$'	匹配开头是'x'，结尾是1-5的字符串
''^(?!Species$).*'	匹配除'Species'以外的所有字符串

```
df.loc[:,'x2':'x4']
```

选取'x2'、'x4'之间的列（包括x2和x4列）

```
df.iloc[:,[1,2,5]]
```

按位置选取第1，2，5列 (第一列的列数为0).

```
df.loc[df['a'] > 10, ['a','c']]
```

选取行符合逻辑条件的指定列

http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants

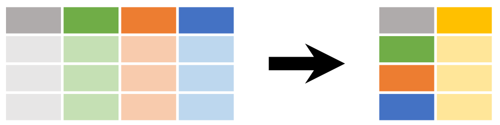
汇总数据

df['Length'].value_counts()
计算某列中各值出现的频率

len(df)
DataFrame 的行数

len(df['w'].unique())
计算某列中各值去重后的个数 .

df.describe()
DataFrame各列（或分组）的基本统计描述



pandas提供了许多*汇总*函数，用来处理不同的pandas数据（如DataFrame，Series，GroupBy，Expanding，Rolling等对象），产生各组的*单个*汇总值。当应用到DataFrame时，每一列分别返回一个汇总值，返回的结果是索引为列名的Series序列。部分常用汇总函数如下：

sum()
对各个对象的值进行求和

count()
对各个对象的非空值进行计数

median()
各个对象的值之中位数

quantile([0.25,0.75])
各个对象的值之分位数

apply(*function*)
对各个对象应用某个函数

min()
各个对象的值之最小值

max()
各个对象的值之最大值

mean()
各个对象的值之平均值

var()
各个对象的值之方差

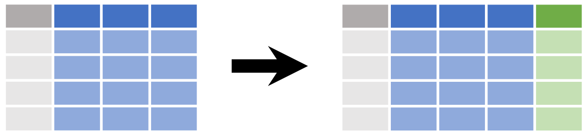
std()
各个对象的值之标准差

处理缺失数据

df=df.dropna()
删除含有空值的行

df=df.fillna(value)
替换所有空值为某个值

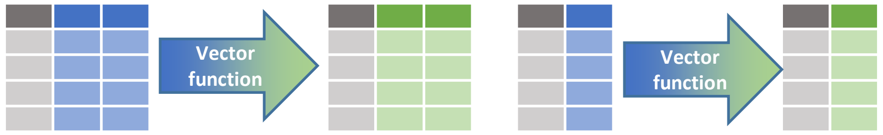
创建新的变量（创建新列）



df=df.assign(Area=lambda df: df.Length*df.Height)
计算和追加一列或多个新列

df['Volume'] = df.Length*df.Height*df.Depth
增加一列

pd.qcut(df.col, n, labels=False)
将列切成n块



pandas提供了许多*向量*函数，用来处理DataFrame的所有列或者选定的单列（即Series对象）。这些函数应用于各列的每个元素，产生长度相等的向量（区别于汇总函数只产生一个汇总值，即长度为1）。对于单列(即Series对象)，产生新的单列（即Series）

max(axis=1)
求各行的最大值（逐元素求最大值）

clip(lower=-10,upper=10)
根据输入阈值修剪数值

min(axis=1)
求各行的最小值（逐元素求最小值）

abs()
求绝对值

合并数据集

adf

x1	x2
A	1
B	2
C	3

✚

bdf

x1	x3
A	T
B	F
D	T

=

标准连接

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

```
pd.merge(adf, bdf,
         how='left', on='x1')
```
以x1列作为键值，将bdf对应的行合并到adf（左连接）

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

```
pd.merge(adf, bdf,
         how='right', on='x1')
```
以x1列作为键值，将abf对应的行合并到bdf（右连接）

x1	x2	x3
A	1	T
B	2	F

```
pd.merge(adf, bdf,
         how='inner', on='x1')
```
以x1列作为键值，合并adf和bdf，仅保留共有键值（内连接）

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

```
pd.merge(adf, bdf,
         how='outer', on='x1')
```
以x1列作为键值，合并adf和bdf，保留所有键值（外连接）

过滤连接

x1	x2
A	1
B	2

adf[adf.x1.isin(bdf.x1)]
选取adf的行，这些行的键值同时存在于bdf中。

x1	x2
C	3

adf[~adf.x1.isin(bdf.x1)]
选取adf的行，这些行的键值不存在于bdf中。

ydf

x1	x2
A	1
B	2
C	3

✚

zdf

x1	x2
B	2
C	3
D	4

=

Set-like Operations

x1	x2
B	2
C	3

pd.merge(ydf, zdf)
合并ydf和zdf的相同行。
(交集).

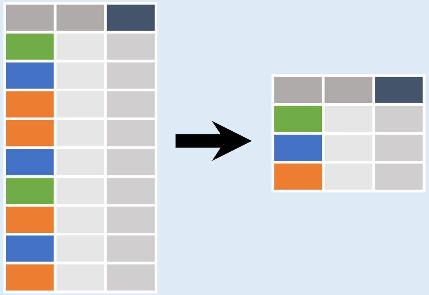
x1	x2
A	1
B	2
C	3
D	4

pd.merge(ydf, zdf, how='outer')
合并ydf和zdf的所有行。
(并集).

x1	x2
A	1

```
pd.merge(ydf, zdf, how='outer',
         indicator=True)
.query('_merge == "left_only"')
.drop(['_merge'],axis=1)
```
合并存在于ydf但不存在于zdf的行。(差集)

分组数据



df.groupby(by="col")
根据“ col ”列的值进行分组，返回GroupBy对象。

df.groupby(level="ind")
根据名为ind的索引级别的值进行分组，返回GroupBy对象。

所有汇总函数都可以应用于GroupBy对象，GroupBy函数还有：

size()
每组的大小

agg(*function*)
利用自定义函数聚合各个group

以下例子也可以应用到分组中。在这种情况下，函数应用在每一个group，返回的向量与原始DataFrame具有相同的长度。

shift(1)
向前偏移一个位置

rank(method='dense')
返回连续的排名值

rank(method='min')
使用分组的最小排名

rank(pct=True)
将排名值缩放到[0, 1]之间

rank(method='first')
按值在原始数据中的出现顺序分配排名

shift(-1)
向后偏移一个位置

cumsum()
累计求和

cummax()
累计最大值

cummin()
累计最小值

cumprod()
累计求积

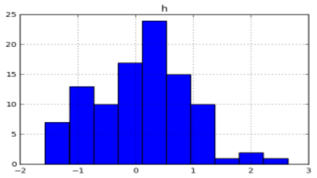
窗口

df.expanding()
返回扩展窗口对象（Expanding object），使得汇总函数渐增地应用到不断扩张的窗口中。

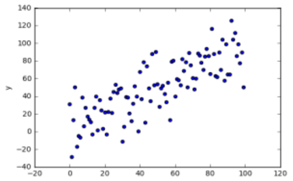
df.rolling(n)
返回移动窗口对象（Rolling object），使得汇总函数移动地应用到长度为n的窗口中。

作图

df.plot.hist()
产生每列的直方图



df.plot.scatter(x='w',y='h')
利用成对的数据点产生散点图



http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants