**导入数据**

pd.read\_csv(filename) # 导入CSV（以‘，’为分隔符）中的数据

pd.read\_table(filename) # 导入CSV（以‘/t’为分隔符）中的数据

pd.read\_excel(filename) # 导入EXCEL 中的数据

pd.read\_sql(query, connection\_object) # 导入数据库中的数据

（import pandas as pd

from sqlalchemy import create\_engine

# 初始化数据库连接，使用pymysql模块

# MySQL的用户：root, 密码:147369, 端口：3306,数据库：mydb

engine = create\_engine('mysql+pymysql://root:147369@localhost:3306/mydb')

# 查询语句，选出employee表中的所有数据

sql = '''

select \* from employee;

'''

# read\_sql\_query的两个参数: sql语句， 数据库连接

df = pd.read\_sql\_query(sql, engine)

pd.read\_json(json\_string) # 导入json 中的数据

pd.read\_html(url) # 导入html 中的数据

pd.read\_clipboard() # 取剪贴板的内容,并将其传递给 read\_table()

pd.DataFrame(dict) # 从字典导入,键为列,值为行

## 探索数据

df.index # 索引描述

df.columns # DataFrom中的列

df.shape # 在 DataFrame 中打印行数和列数

df.head(n) # 打印 DataFrame 的前 n 行

df.tail(n) # 打印 DataFrame 的最后 n 行

df.info() # 索引、数据类型和内存信息

df.describe() # 数字列的汇总统计信息

s.value\_counts() # //统计重复出现的数据的个数。（参数normalize=True显示占比 sort=false不按计数值排序）

df.apply(pd.Series.value\_counts) # 所有列的唯一值和计数

df.mean() # 返回所有列的平均值

df.corr() # 返回 DataFrame 中列之间的相关性

df.count() # 返回每个 DataFrame 列中的非空值数

df.max() # 返回每列中的最高值

df.min() # 返回每列中的最低值

df.median() # 返回每列的中位数

df.std() # 返回每列的标准偏差

s1.corr(s2)#求取是S1列与S2列的相关系数

df.idxmax() # 最低值的索引

df.idxmin() # 最高值的索引

dataframe.duplicated(subset=) # 查看整行是否重复, subset可指定

pd.set\_option('display.float\_format', lambda x: '%.3f' % x)# 这个设置的是保留小数点后3位。针对df.describe()输出科学计数法修改

df[pd.isna(df['col\_name'])] #查看指定列含有空值的行

df.isna().any() #查看所有含有空值的行，返回bool类型

df/s.isin()是否包含某些值

df[df.isnull().values==True] #查看所有含有空值的行

s.isin([‘value1’,’value2’]) #检测某列是否含有某值（可用于df）

## 选择

df[col] # 将col 列作为 Series返回

df[[col1, col2]] # 将列返回为新的 DataFrame

s.iloc[0] # 切片索引；按行切片(选择第一个元素)（具体范围[0:2]）

s.loc[0] # 普通索引；按索引（列）切片(选择元素作为索引 0)(具体名[‘colum/index\_name’])

df.ix[]#交叉索引，可使用切片+普通索引；参数都为先行后列

df.iloc[0, :] # 第一行

df.iloc[0, 0] # 第一列的第一个元素

df.iat[0, 0] # 选择单元格，只能使用整数索引

df.at[row\_label, col\_label] # 选择单元格，只能使用标签索引

## 数据清理

df.columns = ['a', 'b', 'c'] # 重命名列

pd.isnull() # 检查空值,返回布尔数组

pd.notnull() # pd.isnull() 的对立面

df.dropna() # 删除包含空值的所有行

df.dropna(axis=1) # 删除包含空值的所有列

df.dropna(axis=1, thresh=n) #保留至少有n个非NaN数据的列

具有subset= 参数可以设置哪一列的值重复就删除

df.fillna(x) # 将所有空值替换为

x

s.fillna(s.mean()) # 将所有空值替换为均值

s.astype(float) # 将序列的数据类型转换为浮点

s.replace(1, 'one') # Replaces all values equal to 1 with 'one'

s.replace([1, 3], ['one', 'three']) # Replace all 1 with 'one' and 3 with 'three'

df.rename(columns=lambda x: x + 1) # Mass renaming of columns

df.rename(columns={'old\_name': 'new\_name'}) # Selective renaming

df.set\_index('column\_one',inplace=true) # 更改索引

df.stack()#重构索引；表型结构转换成树形

df.reset\_index()#还原索引

df.index = pd.DatetimeIndex(df[‘date’])#转化为时间索引

df['date'] = pd.to\_datetime(df['date'])#转化为时间类型

df.rename(index=lambda x: x + 1) # Mass renaming of index

df.drop(labels) # Drop specified labels from rows or columns

dataframe.drop\_duplicats() #用于丢弃重复项（subset=‘col’基于某列去重）subset : column label or sequence of labels, optional

用来指定特定的列，默认所有列

keep : {‘first’, ‘last’, False}, default ‘first’

删除重复项并保留第一次出现的项

inplace : boolean, default False

是直接在原来数据上修改还是保留一个副本

del df['A'] #删除列

df.drop(df[np.isnan(df['col\_name'])].index, inplace=True)#删除指定列含有空值的行

df.replace('',np.nan)

df.replace(r'\s+', np.nan, regex=True)# regex=True启用正则；两个连用将‘’和‘ ’转换为Nan值

## 过滤，排序和分组

df[df[col] > 0.5] # Rows where the col column is greater than 0.5

df[(df[col] > 0.5) & (df[col] < 0.7)] # Rows where 0.5 < col < 0.7# | 为‘或’

df[df[‘成交时间’] == datetime（2018,8,8）]#当date为列时按时间筛选

s.rank(ascending=false,method=min)#列排序，参数ascending=false从大到小排；参数method=min（遇重复值处理参数）计最小的名次，有重复值、first第一个重复为n第二个为n+1；average重复值名次计平均如：1.5、2.5

df.sort\_values(col2, ascending=False) # 按降序按 col2 对值进行排序（参数na\_position = first将nan显示在最前面）（by=[‘col1’,’col2’],ascengding=[True,False]多列排序）

df.s = s.unique()#获取唯一值

df.groupby(col) # Returns a groupby object for values from one column

df.groupby([col1,col2]) # Returns a groupby object values from multiple columns

df.groupby(col1)[col2].mean() # Returns the mean of the values in col2, grouped by the values in col1 (mean can be replaced with almost any function from the statistics section)

df.groupby(‘col\_name’).aggregate([‘count’，‘sum’])#对每列进行count和sum；{‘col\_name’：‘count’，“col2\_name”:‘sum’}对指定列进行指定聚合函数

df.pivot\_table(index=col1, values= col2,col3], aggfunc=mean) # Creates a pivot table that groups by col1 and calculates the mean of col2 and col3 #透视表

#逆透视

df.groupby(col1).agg(np.mean) # Finds the average across all columns for every unique column 1 group

df.apply(np.mean) # Applies a function across each column

df.apply(np.max, axis=1) # Applies a function across each row

pd.cut(df[‘col’],bins=[0,3,6,10]) #数据规整（数据分组）参数bins=指定区间

pd.qcut(df[‘col’], ) #数据规整（数据分组）第二个参数指定切分成多少份。

From pandas.tseries.offsets import Dat,Hour,Minute

Dare + Day(1)#时间偏移

## 导出数据

df.to\_csv(filename) # Writes to a CSV file

df.to\_excel(filename) # Writes to an Excel file

df.to\_sql(table\_name, connectiion\_object) # Writes to a SQL table

df.to\_json(filename) # Writes to a file in JSON format

df.to\_html(filename) # Saves as an HTML table

df.to\_clipboard() # Writes to the clipboard

## 连接和合并

pd.concat(objs, axis=0, join='outer', join\_axes=None, ignore\_index=False,keys=None, levels=None, names=None, verify\_integrity=False,copy=True)#

1、aixs=0 行拼接，效果与obj1.append(obj2)是相同的

2、ignore\_index=False 是否忽略索引而重建

3、join “inner”：列的交集 “outer”：列的并集

4、append函数

pd.append(other, ignore\_index=False, verify\_integrity=False, sort=None)

a、other其它dataframe

b、忽略索引

c、类似于pd.concat(axis=0)

pd.merge(left,right,how='inner',on=None,left\_on=None,right\_on=None,left\_index=False,right\_index=False,sort=True,suffixes=('\_x', '\_y'), copy=True, indicator=False,validate=None)

1、left：DataFrame对象

2、right：DataFrame对象

3、how：连接的方式inner、left、right、outer，默认为inner

4、on：用于连接的列名称，必须存在于左右两个DataFrame中，如果没有指定且其他参数也没有指定，则以两个DataFrame列名交集作为连接键

1. left\_on，right\_on：指定左右两侧连接的列名

join(self,other,on=None,how='left',lsuffix='',rsuffix='',sort=False)#与merge功能类似，区别在于用于索引上的合并

pd.merge(left, right, left\_index=True, right\_index=True, how='outer') #merge索引并集

## Pandas绘图

pd.plot()#

s.plot()#

stacked=True#参数设置是否堆叠

参数kind = ‘’指定图表类型

‘line’ : 折线图 (默认)

‘bar’ : 柱状图

‘barh’ : 水平柱状图

‘hist’ : 直方图

‘box’ : 箱体图

‘kde’ : Kernel Density Estimation plot

‘density’ : same as ‘kde’

‘area’ : 面积图

‘pie’ : 饼图

| 参数 | 说明 |
| --- | --- |
| label | 用于图例的标签 |
| ax | 要在其上进行绘制的matplotlib subplot对象.如果没有设置，则使用当前matplotlib subplot |
| style | 将要传给matplotlib的风格字符串(如'ko--') |
| alpha | 图表的填充不透明度(0到1之间) |
| kind | 可以是'line', 'bar', 'barh', 'kde |
| logy | 在y轴上使用对数标尺 |
| use index | 将对象的索引用作刻度标签 |
| rot | 旋转刻度标签(0到360) |
| xticks | 用作x轴刻度的值 |
| yticks | 用作y轴刻度的值 |
| xlim | x轴的界限(例如[0,101) |
| ylim | y轴的界限 |
| grid | 显示轴网格线(默认打开) |

C:\Users\WD53259\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B322E8EF.tmp

Series.plot方法的参数如下：

| 参数 | 说明 |
| --- | --- |
| label | 用于图例的标签 |
| ax | 要在其上进行绘制的matplotlib subplot对象.如果没有设置，则使用当前matplotlib subplot |
| style | 将要传给matplotlib的风格字符串(如'ko--') |
| alpha | 图表的填充不透明度(0到1之间) |
| kind | 可以是'line', 'bar', 'barh', 'kde |
| logy | 在y轴上使用对数标尺 |
| use index | 将对象的索引用作刻度标签 |
| rot | 旋转刻度标签(0到360) |
| xticks | 用作x轴刻度的值 |
| yticks | 用作y轴刻度的值 |
| xlim | x轴的界限(例如[0,101) |
| ylim | y轴的界限 |
| grid | 显示轴网格线(默认打开) |

专用于DataFrame.plot方法的参数如下：

| 参数 | 说明 |
| --- | --- |
| subplots | 将各个DataFrame列绘制到单独的subplot中 |
| sharex | 如果subplots=True，则共用同一个x轴，包括刻度和界限 |
| sharey | 如果subplots=True，则共用同一个Y轴 |
| figsize | 表示图像大小的元组 |
| title | 表示图像标题的字符串 |
| legend | 添加一个subplot图例(默认为True) |
| sort\_columns | 以字母表顺序绘制各列，默认使用当前列顺序 |

## 创建测试对象

pd.DataFrame(np.random.rand(20,5)) # 5 columns and 20 rows of random floats

pd.Series(my\_list) # Create a series from an iterable my\_list

df.index = pd.data\_range('1900/1/30', periods=df.shape[0]) # Add a date index



