import pandas as pd

df：Pandas DataFrame对象

s: Pandas Series对象

只有numpy array才支持对其中的每个元素进行单一计算

取列是筛选标签，取行是筛选标签的值

**显示设置**

**pd.set\_option('display.max\_rows',None)** #Show all rows

**pd.set\_option('display.max\_columns',None)** #Show all cols

**数据导入**

**pd.read\_csv(filename,header=1) #**从CSV文件导入数据

**pd.read\_table(filename, header=1) #**从限定分隔符的文本文件导入数据

**pd.read\_excel(filename,sheet名,header=1,na\_values=[‘NA’]) #**从Excel文件导入数据

**数据导出**

**df.to\_csv(filename, index=False)** #导出数据到CSV文件

**df.to\_excel(filename,sheet\_name=表名,)** #导出数据到Excel文件

**df.to\_numpy()** #导出数据为numpy格式

**df.values** #转化为矩阵

**df.列.values.tolist()** #用list导出某一列值

**数据显示设置**

**pd.set\_option('display.max\_rows',显示的最大行数)**

**pd.set\_option('display.max\_columns',显示的最大列数)**

#最大行列数设置为None时，为全显示

## Pandas Series

类似R中添加了元素别名的数组

**s=pd.Series(data,index,dtype)**

#data,index均为矢量

#当data为字典时，可通过index指定s中的元素顺序

#如果data元素个数小于index，会循环填充

#index缺省时，默认为0,1,2,3,…

**Pandas Series的切片操作及元素索引操作类似R的矢量，只是第一个元素序列从0开始，而负的序列号是指从末尾排序**

**s.head(n)** #前n个元素

**s.tail(n)** #后n个元素

**s.index**  #获取名称

**元素去重：**

**s.unique()** #此功能相当于R中levels(as.factor())

**两个Series相运算，在运算中自动对齐不同索引的数据，如果索引不对应，则补NaN**

**s.isnull()**

**s.notnull()**

**s[s.isnull()] # 取所有空值**

**s[s.notnull()] # 取出不为空的数据**

**s=s.astype(float)：将Series中的数据类型更改为float类型**

**int 整型 float 浮点型 bool 布尔类型str字符串类型category 种类 datetime 时间类型**

**s.replace(1,'one')：用‘one’代替所有等于1的值**

**s.to\_frame() 转DataFrame**

**元素提取：Series具有loc和iloc方法**

**增加元素：s=s.append(a)**

**合并Series: s1=s1.append(s2)**

**计算两个序列间的相关系数：s.corr(s2)** #通过index进行元素配对，所以不同序列注意index

## Pandas DataFrame

**定义：df=pd.DataFrame(data,columns=[列索引],index=[行索引])**

**df=pd.DataFrame(data={列名:数据})**

**基本取数据：**

**df[列名] 或 df.列名** #取列(结果格式为series)

**df[[列名]]** #取列(结果格式为DataFrame)

**df[index]** #取行

**df[条件]** #条件取行或所有值，注意方括号中没有逗号

！诸如df.A==n这种写法不能应用于多条件判定，如df.A<n1 and df.A>n2

**df[df.A.isin(值集合)]** #isin()的条件取行

**df[df.A.str.contains(“正则表达式”)]** #字符串列包含特定字符的行筛选

**基于位置提取数据：**

**df.iloc[a]** #取第a行（行数从0开始）

**df.iloc[a:b]** #取第a到第b-1行

**df.iloc[a:b,c]** #取第a到第b-1行的第c列的值

**df.iloc[a:b,c:d]** #取第a到第b-1行的第c列到第d-1列的值

**df.iloc[[a,b]]** #取第a、b行

**df.iloc[[a,b],[c,d]]** #取第a、b行的第c、d列值

**df.iloc[布尔向量1,布尔向量]** #用布尔值向量来取值

**基于标签提取数据：**

**df.loc[行标签]** #提取对应行标签的行

**df.loc[:,列标签]** #提取对应列标签的列

**df.loc[行标签,列标签]** #提取对应行列标签的值

**基于条件提取数据：**

**df[(条件1)&(条件2)]** #与筛选

**df[(条件1)|(条件2)]** #或筛选

**！**条件必须用括号括起来且逻辑运算不能用and or not

**df.iloc[~条件]** #逻辑条件取反~

**数据查看：**

**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

**df.info()** #查看索引、数据类型和内存信息

**df.dtypes** #查看每一列数据格式

**df.describe()** #查看数值型列的汇总统计

**df.列.value\_counts()** #相当于R中table(列)

**df.列.describe()**  #相当于R中summary(列)

**df.列.quantile(小数list)** #查看某列的分位数

**df.列.round(decimals=小数位数)** #保留小数

**df.columns** #查看数据列名

**df.index** #查看行名

**df.列.unique()**  #查看唯一值

**df.nunique()** #查看各列的唯一值个数

**np.where(条件)**#获取df中符合条件的行的索引值，**结果是一个元组，需取[0]**

**数据操作**

**重定义列名df.columns = ['a','b','c']**

**df.rename(columns={旧列名:新列名},inplace=True)**

**重定义行名df.index=['a','b','c']**

**更新索引 df.reset\_index()** #常用于groupby后

**使用某列作为index df.set\_index('列名')**

**设置新index df.reindex(new\_index, fill\_value=None)**

#原df中没有对应index的行默认填充None，可用于处理index不连续的问题

#原df中不存在于new\_index的行会被舍弃

#例：处理日期不连续，先set\_index(日期列), 然后用pd.date\_range生成一个完整的日期Series并reindex到原数据中。

**字段重排序 df=df[[重排序后的字段顺序]]**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**df.isnull()**：检查DataFrame对象中的空值

**df.isnull().sum()**: 统计每列缺失值

**df.notnull()**：检查DataFrame对象中的非空值

**df.A.fillna(value,inplace=True)**：填充某一列空值

**df.interpolate(\*method='linear', \*limit\_direction='forward', \*axis=0)**: 对缺失值插值

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**df.dropna(how='any'/’all’,axis=0/1,inplace=True,subset=[列名])** #去除含缺失值的行或全是缺失值的行/列

**df.drop(index=行索引, inplace=True)** #删除某些行

**df.drop(columns=列名, inplace=True)** #删除某些列

**del df.A** #删除某一列

!drop函数会打乱index排序,故而在drop后应该随手reset\_index()

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**df.duplicated(列名或列名集,keep='first'/'last'/False)** #生成布尔向量，标记每一行是否重复。

**df[df.duplicated(列名或列名集,keep='first'/'last'/False)]** #查看重复的行

**data =** **data.drop\_duplicates() #**删除重复行，需要reset\_index()

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**采样 df.sample(n=样本数,frac=小数比例,replace=False,axis=0/1)**

**打乱 from sklearn.utils import shuffle**

**shuffle(df)**

**转置 df.T**

**排序 df.sort\_values(by=列名或列名集,ascending=F)**

**df.sort\_index(inplace=True)** #按索引值排序

**分类汇总 df.groupby(列名).函数() #**函数可以是count、sum、mean、describe等, 如果不加汇总函数，则会返回一个指针

**自定义分类汇总函数 df.groupby(列名).agg({’Name1’:func1, ’Name2’:func2,…})**

#func为内置函数传字符串如’sum’,’count’；为自定义函数时直接传函数名

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**转换为日期格式 pd.to\_datetime(data,format=’****%Y-%m-%d %H:%M:%S’)**

**仅保留date pd.to\_datetime(data,format=’%Y-%m-%d %H:%M:%S’).dt.date**

**timedelta取日 data.dt.days**

**timedelta取时间 data.dt.time**

#dt类似于调用字符串函数时的str, 但用于调用datetime方法

**生成日期Series pd.date\_range(start = '2016-01-04 00:00:00',end = '2016-01-04 23:59:59',\*freq='s'/'min'/…, \*periods=元素个数)**

**转换数据类型 data=data.astype(int/float/str)**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**字符串数据切片 data.str[a:b]**

**连续型数据分桶 pd.cut(data,bins=边界数组,labels=标签矢量)**

**均等分桶 pd.qcut(data,q=桶数,labels=标签矢量)**

**字符串列操作 df.列名.str.字符串操作命令**

**字符串操作命令strip()、split()等**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**数据行合并 newdata=pd.concat([data1,data2,…],keys=[label1,label2,…],axis=0) #**keys参数在合并表index加一列以标明各行数据的源表

**批量读取并合并数据**

**data=[pd.read\_csv(data\_part) for data\_part in data\_list ]**

**data=pd.concat(data,ignore\_index=True)**

**数据链接 pd.merge(data1,data2,how=”inner”/”outer”/”left”/”right”,on=键名,left\_on=左键名,right\_on=右键名,suffixes=[labels])**

**#只能两两merge**

#键名共有时用on参数，键名不同时用left\_on和right\_on,遇到有一列两个表都同名，但是值不同，合并的时候又都想保留下来，就可以用suffixes给每个表的重复列名增加后缀。

**追加行 df=df.append(pd.Series,** **ignore\_index=True)**

**批量追加行**

**df2.columns=df1.columns** #先统一列名

**newdata=pd.concat([df1,df2],** **ignore\_index=True)**

**追加列 df[新列名]=data**  #！Pandas不可以使用df.新列名来创建新列

**批量追加列 df.join(new\_df)** #! 要求df和new\_df的index相同

**利用数据链接打标签**

1. 取冗余行为data\_tmp ②drop\_duplicates ③为data\_tmp创建label列 ④用冗余行作为主键与原数据进行链接，label列会自动扩展到原数据的每一行

#此方法适用于大型数据，比for循环快很多

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**替换特定值 df.replace(old\_value,new\_value,replace=True)**

**仅对某一列替换 df.A.replace(old\_value,new\_value,replace=True)**

**批量映射 df.A.map({old\_value1:new\_value1,…, old\_valuen:new\_valuen},inplace=True)**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**数据逆重整（数据透视表）**

**pd.pivot\_table(df, index=保留列, columns=逆重整列, values=用以填充逆重整生成列的列, \*aggfunc=[”count”,”mean”,”sum”,…])**

**#**数据透视表实际上是多维度的分类显示或汇总

#使用reset\_index来把透视表化为平整格式

当Values包含多列时，aggfunc可通过字典方式来对不同列应用不同汇总函数

例：table=pd.pivot\_table(df,index=['对手','胜负'],columns=['主客场'],values=['得分','助攻','篮板'],aggfunc={'得分':np.mean, '助攻':[min, max, np.mean]}

**数据重整**

**pd.melt(df,id\_vars=保留列, \*value\_vars=被重整列, \*var\_name=重整生成的列名, \*value\_name='value')**

#id\_vars和value\_vars只用设一个，如果仅设置保留列，除保留列外，所有其他列都会被重整。反之亦然。

#数据重整用于将被重整列的列名，转换为重整生成列的水平，原被重整列下的值会被排在重整生成列对应的水平后。

**基于分类变量进行One-hot化**

**pd.get\_dummies(基于One-hot化的列)**

#要求基于One-hot化的列是分类变量

## 数据计算

**df.describe()**：查看数据值列的汇总统计

**df.mean()**：返回所有列的均值

**df.mean(1)**：返回所有行的均值

**df.corr()**：返回数字类型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

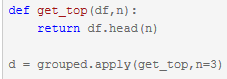
**df.diff(periods=n,axis=0/1)**: 对数据行或列进行n阶差分

**df.A.apply(function)**: 对某一列应用某一函数（可以是lambda函数），仅能作用于列

df.列.apply(匿名函数或自定义函数)

lambda x: 5 if… else 0

自定义函数可以在apply外定义好，然后在apply里调用



**df.rolling(window,** \***center=False,\*closed=控制左右开闭).somefunction()**：滑动窗口计算

**df.corr()**：协相关矩阵

## 字符处理

!每调用一次字符处理，就要申明一次str方法，如str.contains.str.isin()

**转大写**：s.str.upper()

**转小写**：s.str.lower()

**取长度**：s.str.len()

**替换**：s.str.replace(待替换字符,替换字符)

**分割**：s.str.split(分割符,expand=True,n=每个字符串的切分次数) #会将序列中每个字符串分割，并以列表形式返回

**索引分割结果**：new\_col=col\_splited.str.get(index)

**利用分割结果创建新表**：col\_splited.columns=列名list

**利用逆zip实现分割结果批量创建新列**：新列1,新列2,…=zip(\*data.col.str.split(分割符))

**拼接**：s.str.cat(sep=连接符,na.rep=缺失值替代符) #连接符默认为空；如果不设置缺失值替代符，缺失值会被忽略

**两个字符序列对应元素拼接**：s.str.cat(字符序列t,na.rep=缺失值替代符) #会将序列s和t中的字符串对应拼接并返回一个列表

**正则提取：**s.str.extract(正则表达式,expand=True/False) #!与re库不同，此处的正则需用括号指明捕获组；此方法当无匹配项时，返回Nan

**包含判定：**s.str.contains(string)

**isin判定：**s.str.isin(字符串数组)

样本打标签思路：先添加一列，然后根据筛选条件赋值

## 画图

**散点图**：df.plot(kind=’scatter’,x=X轴列名,y=Y轴列名,alpha=透明度,figsize=(图长,图宽))

**气泡图**：df.plot(kind=’scatter’,x=X轴列名,y=Y轴列名,s=气泡半径列,label=标签名,c=颜色列)

plt.legend()

**散点图矩阵：**pd.plotting.scatter\_matrix(data,figsize=(图长,图宽))