Pandas的应用

1、Pandas入门

2、Pandas索引

3、Pandas数据清洗之空数据

4、Pandas多层索引

5、Pandas多层索引计算

6、Pandas数据集成concat

7、Pandas数据集成merge

8、Pandas分组聚合操作

9、Pandas数据集成实战

10、美国大选项目

pandas是python环境下最有名的数据统计包，而DataFrame翻译为数据框，是一种数据组织方式，这么说你可能无法从感性上认识它，举个例子，你大概用过Excel，而它也是一种数据组织和呈现的方式，简单说就是表格，而在在pandas中用DataFrame组织数据，如果你不print DataFrame，你看不到这些数据。

pandas和numpy的区别：

> 1.numpy是数值计算的扩展包，panadas是做数据处理。
> 2.NumPy简介：N维数组容器NumPy系统是Python的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比Python自身的嵌套列表（nested list structure)结构要高效的多（该结构也可以用来表示矩阵（matrix））。据说NumPy将Python相当于变成一种免费的更强大的MatLab系统。
> Pandas简介：表格容器 pandas 是基于NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。Pandas 纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具。pandas提供了大量快速便捷地处理数据的函数和方法。使Python成为强大而高效的数据分析环境的重要因素之一。

一、生成数据表

1、首先导入pandas库，一般都会用到numpy库，所以我们先导入备用：

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

2、导入CSV或者xlsx文件：

```
df = pd.DataFrame(pd.read_csv('name.csv',header=1))
df = pd.DataFrame(pd.read_excel('name.xlsx'))
```

3、用pandas创建数据表：

```
df = pd.DataFrame({"id":[1001,1002,1003,1004,1005,1006],
 "date":pd.date_range('20130102', periods=6),
  "city":['Beijing ', 'SH', ' guangzhou ', 'Shenzhen', 'shanghai', 'BEIJING '],
 "age":[23,44,54,32,34,32],
 "category":['100-A','100-B','110-A','110-C','210-A','130-F'],
  "price":[1200,np.nan,2133,5433,np.nan,4432]},
  columns =['id','date','city','category','age','price'])
```

二、数据表信息查看

1、维度查看：

df.shape

2、数据表基本信息（维度、列名称、数据格式、所占空间等）：

df.info()

3、每一列数据的格式：

df.dtypes

4、某一列格式：

df['B'].dtype

5、空值：

df.isnull()

6、查看某一列空值：

df.isnull()

7、查看某一列的唯一值：

df['B'].unique()

8、查看数据表的值：

df.values

9、查看列名称：

df.columns

10、查看前10行数据、后10行数据：

df.head() #默认前10行数据
df.tail() #默认后10 行数据

三、数据表清洗

1、用数字0填充空值：

df.fillna(value=0)

2、使用列prince的均值对NA进行填充：

df['prince'].fillna(df['prince'].mean())

3、清楚city字段的字符空格：

df['city']=df['city'].map(str.strip)

4、大小写转换：

df['city']=df['city'].str.lower()

5、更改数据格式：

df['price'].astype('int')

6、更改列名称：

df.rename(columns={'category': 'category-size'})

7、删除后出现的重复值：

df['city'].drop_duplicates()

8、删除先出现的重复值：

df['city'].drop_duplicates(keep='last')

9、数据替换：

df['city'].replace('sh', 'shanghai')

四、数据预处理

```
df1=pd.DataFrame({"id":[1001,1002,1003,1004,1005,1006,1007,1008],
"gender":['male','female','male','female','male','female','male','female'],
"pay":['Y','N','Y','Y','N','Y','N','Y',],
"m-point":[10,12,20,40,40,40,30,20]})
```

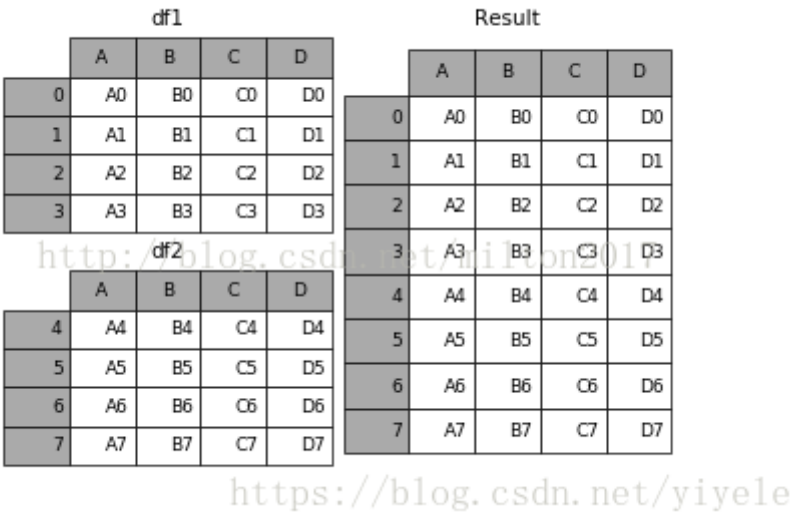
1、数据表合并

1.1 merge

```
df_inner=pd.merge(df,df1,how='inner')  # 匹配合并，交集
df_left=pd.merge(df,df1,how='left')        #
df_right=pd.merge(df,df1,how='right')
df_outer=pd.merge(df,df1,how='outer')  #并集
```

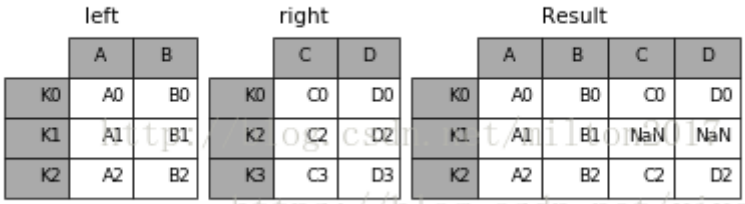
1.2 append

```
result = df1.append(df2)
```



1.3 join

```
result = left.join(right, on='key')
```



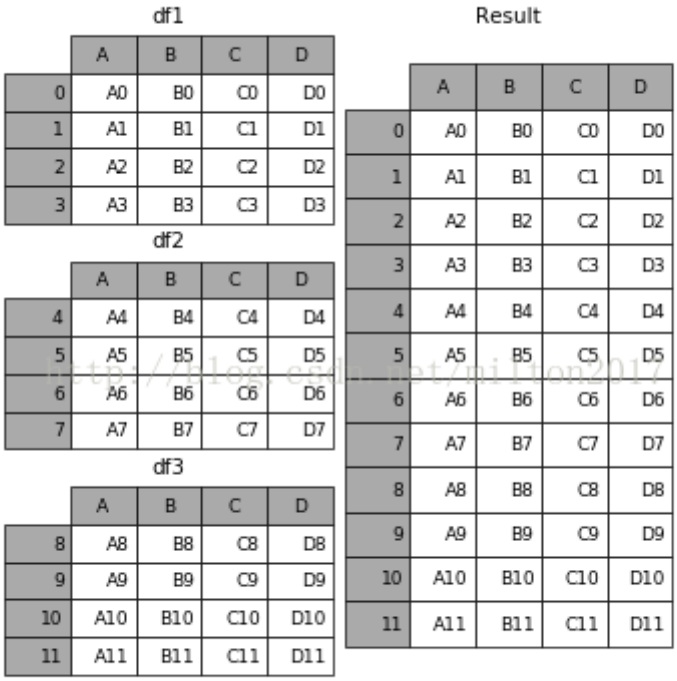
1.4 concat

```
pd.concat(objs, axis=0, join='outer', join_axes=None, ignore_index=False,
          keys=None, levels=None, names=None, verify_integrity=False,
          copy=True)
```

objs： 一个序列或系列、 综合或面板对象的映射。如果字典中传递，将作为键参数，使用排序的键，除非它传递，在这种情况下的值将会选择 （见下文）。任何没有任何反对将默默地被丢弃，除非他们都没有在这种情况下将引发 ValueError。
axis: {0，1，...}，默认值为 0。要连接沿轴。
join: {‘内部’、 ‘外’}，默认 ‘外’。如何处理其他 axis(es) 上的索引。联盟内、 外的交叉口。
ignore_index： 布尔值、 默认 False。如果为 True，则不要串联轴上使用的索引值。由此产生的轴将

标记 0，…，n-1。这是有用的如果你串联串联轴没有有意义的索引信息的对象。请注意在联接中仍然受到尊重的其他轴上的索引值。
join_axes： 索引对象的列表。具体的指标，用于其他 n-1 轴而不是执行内部/外部设置逻辑。
keys： 序列，默认为无。构建分层索引使用通过的键作为最外面的级别。如果多个级别获得通过，应包含元组。
levels： 列表的序列，默认为无。具体水平 （唯一值） 用于构建多重。否则，他们将推断钥匙。
names： 列表中，默认为无。由此产生的分层索引中的级的名称。
verify_integrity： 布尔值、 默认 False。检查是否新的串联的轴包含重复项。这可以是相对于实际数据串联非常昂贵。
副本： 布尔值、 默认 True。如果为 False，请不要，不必要地复制数据。

例子：1.frames = [df1, df2, df3]
2.result = pd.concat(frames)



2、设置索引列

df_inner.set_index('id')

3、按照特定列的值排序：

df_inner.sort_values(by=['age'])

4、按照索引列排序：

df_inner.sort_index()

5、如果prince列的值>3000，group列显示high，否则显示low：

df_inner['group'] = np.where(df_inner['price'] > 3000,'high','low')

6、对复合多个条件的数据进行分组标记

df_inner.loc[(df_inner['city'] == 'beijing') & (df_inner['price'] >= 4000), 'sign']=1

7、对category字段的值依次进行分列，并创建数据表，索引值为df_inner的索引列，列名称为category和size

pd.DataFrame((x.split('-') for x in df_inner['category']),index=df_inner.index,columns=['category','size']))

8、将完成分裂后的数据表和原df_inner数据表进行匹配

df_inner=pd.merge(df_inner,split,right_index=True, left_index=True)

五、数据提取

主要用到的三个函数：loc,iloc和ix，loc函数按标签值进行提取，iloc按位置进行提取，ix可以同时按标签和位置进行提取。

1、按索引提取单行的数值

df_inner.loc[3]

2、按索引提取区域行数值

df_inner.iloc[0:5]

3、重设索引

df_inner.reset_index()

4、设置日期为索引

df_inner=df_inner.set_index('date')

5、提取4日之前的所有数据

df_inner[:'2013-01-04']

6、使用iloc按位置区域提取数据

df_inner.iloc[:3,:2] #冒号前后的数字不再是索引的标签名称，而是数据所在的位置，从0开始，前三行，前两列。

7、适应iloc按位置单独提起数据

df_inner.iloc[[0,2,5],[4,5]] #提取第0、2、5行，4、5列

8、使用ix按索引标签和位置混合提取数据

df_inner.ix[:'2013-01-03',:4] #2013-01-03号之前，前四列数据

9、判断city列的值是否为北京

df_inner['city'].isin(['beijing'])

10、判断city列里是否包含beijing和shanghai，然后将符合条件的数据提取出来

df_inner.loc[df_inner['city'].isin(['beijing','shanghai'])]

11、提取前三个字符，并生成数据表

pd.DataFrame(category.str[:3])

六、数据筛选

使用与、或、非三个条件配合大于、小于、等于对数据进行筛选，并进行计数和求和。

1、使用"与"进行筛选

df_inner.loc[(df_inner['age'] > 25) & (df_inner['city'] == 'beijing'), ['id','city','age','category','gender']]

2、使用"或"进行筛选

df_inner.loc[(df_inner['age'] > 25) | (df_inner['city'] == 'beijing'),
['id','city','age','category','gender']].sort(['age'])

3、使用"非"条件进行筛选

df_inner.loc[(df_inner['city'] != 'beijing'), ['id','city','age','category','gender']].sort(['id'])

4、对筛选后的数据按city列进行计数

df_inner.loc[(df_inner['city'] != 'beijing'), ['id','city','age','category','gender']].sort(['id']).city.count()

5、使用query函数进行筛选

df_inner.query('city == ["beijing", "shanghai"]')

6、对筛选后的结果按prince进行求和

df_inner.query('city == ["beijing", "shanghai"]').price.sum()

七、数据汇总

主要函数是groupby和pivote_table

1、对所有的列进行计数汇总

df_inner.groupby('city').count()

2、按城市对id字段进行计数

df_inner.groupby('city')['id'].count()

3、对两个字段进行汇总计数

df_inner.groupby(['city','size'])['id'].count()

4、对city字段进行汇总，并分别计算prince的合计和均值

df_inner.groupby('city')['price'].agg([len,np.sum, np.mean])

八、数据统计

数据采样，计算标准差，协方差和相关系数

1、简单的数据采样

df_inner.sample(n=3)

2、手动设置采样权重

weights = [0, 0, 0, 0, 0.5, 0.5]
df_inner.sample(n=2, weights=weights)

3、采样后不放回

df_inner.sample(n=6, replace=False)

4、采样后放回

df_inner.sample(n=6, replace=True)

5、 数据表描述性统计

df_inner.describe().round(2).T #round函数设置显示小数位，T表示转置

6、计算列的标准差

df_inner['price'].std()

7、计算两个字段间的协方差

df_inner['price'].cov(df_inner['m-point'])

8、数据表中所有字段间的协方差

df_inner.cov()

9、两个字段的相关性分析

df_inner['price'].corr(df_inner['m-point']) #相关系数在-1到1之间，接近1为正相关，接近-1为负相关，0为不相关

10、数据表的相关性分析

df_inner.corr()

九、数据输出

分析后的数据可以输出为xlsx格式和csv格式

1、写入Excel

df_inner.to_excel('excel_to_python.xlsx', sheet_name='bluewhale_cc')

2、写入到CSV

df_inner.to_csv('excel_to_python.csv')