04数据处理与统计方法

一、Pandas数据统计函数

0.读取数据

```
df = pd.read_csv('res/beijing_tianqi_2018.csv')
# 替换掉温度的后缀℃
df.loc[:, "bWendu"] = df["bWendu"].str.replace("℃", "").astype('int32')
df.loc[:, "yWendu"] = df["yWendu"].str.replace("℃", "").astype('int32')
```

1.汇总类统计

```
# 一下子提取所有数字列统计结果
df.describe()
## 查看单个Series的数据
df["bWendu"].mean()
# 最高温
df["bWendu"].max()
# 最低温
df["bWendu"].min()
```

2.去重与按值计数

```
# 1.唯一性去重
# 一般不用于数值列，而是枚举、分类列
df["fengxiang"].unique()
df["tianqi"].unique()
df["fengli"].unique()
# 2.按值计数
df["fengxiang"].value_counts()
df["tianqi"].value_counts()
df["fengli"].value_counts()

# 3.相关系数和协方差
'''
用途：
两只股票，是不是同涨同跌？程度多大？正相关还是负相关？
产品销量的波动，跟哪些因素正相关、负相关，程度有多大？

来自知乎，对于两个变量X、Y：
```

```
协方差：衡量同向反向程度，如果协方差为正，说明X，Y同向变化，协方差越大说明同向程度越高；如果协方差为负，说明X，Y反向运动，协方差越小说明反向程度越高。
相关系数：衡量相似度程度，当他们的相关系数为1时，说明两个变量变化时的正向相似度最大，当相关系数为－1时，说明两个变量变化的反向相似度最大
'''
df.cov() # 协方差矩阵
df.corr() # 相关系数矩阵
df["aqi"].corr(df["bWendu"]) #空气质量和最高温度的相关系数
```

二、字符串处理

`Pandas`的字符串处理：

1. 使用方法：先获取`Series`的`str`属性，然后在属性上调用函数；
2. 只能在字符串列上使用，不能数字列上使用；
3. `Dataframe`上没有`str`属性和处理方法
4. `Series.str`并不是`Python`原生字符串，而是自己的一套方法，不过大部分和原生`str`很相似；

Series.str字符串方法列表参考文档:
https://pandas.pydata.org/pandas-docs/stable/reference/series.html#string-handling

1.获取`Series`的`str`属性，使用各种字符串处理函数

```
df["bWendu"].str
# 字符串替换函数
df["bWendu"].str.replace("℃", "")
# 判断是不是数字
df["bWendu"].str.isnumeric()
df["aqi"].str.len() #报错
```

3.使用`str`方法得到的`bool`值可以做条件查询

```
condition = df["ymd"].str.startswith("2018-03")
df[condition].head()
```

4.需要多次`str`处理的链式操作

怎样提取201803这样的数字月份？
1、先将日期2018-03-31替换成20180331的形式
2、提取月份字符串201803

```
df["ymd"].str.replace("-", "")
# 每次调用函数，都返回一个新Series
df["ymd"].str.replace("-", "").slice(0, 6) #报错
df["ymd"].str.replace("-", "").str.slice(0, 6)
# slice就是切片语法，可以直接用
df["ymd"].str.replace("-", "").str[0:6]
```

三、数据排序

`Series`的排序：
`Series.sort_values(ascending=True, inplace=False)`
参数说明：

- `ascending`：默认为`True`升序排序，为`False`降序排序
- `inplace`：是否修改原始`Series`

`DataFrame`的排序：
`DataFrame.sort_values(by, ascending=True, inplace=False)`
参数说明：

- `by`：字符串或者`List<字符串>`，单列排序或者多列排序
- `ascending`：`bool`或者`List`，升序还是降序，如果是`list`对应`by`的多列
- `inplace`：是否修改原始`DataFrame`

1.Series的排序

```
df["aqi"].sort_values()
df["aqi"].sort_values(ascending=False)
```

2.DataFrame的排序

2.1 单列排序

```
df.sort_values(by="aqi")
df.sort_values(by="aqi", ascending=False)
```

2.2 多列排序

```
# 按空气质量等级、最高温度排序，默认升序
df.sort_values(by=["aqiLevel", "bWendu"])
# 两个字段都是降序
df.sort_values(by=["aqiLevel", "bWendu"], ascending=False)
# 分别指定升序和降序
df.sort_values(by=["aqiLevel", "bWendu"], ascending=[True,
False])
```

四、新增数据列

在进行数据分析时，经常需要按照一定条件创建新的数据列，然后进行进一步分析。

1. 直接赋值
2. `df.apply( )`方法
3. `df.assign( )`方法

4. 按条件选择分组分别赋值

1.直接赋值

```
# 注意，df["bWendu"]其实是一个Series，后面的减法返回的是Series
df.loc[:, "wencha"] = df["bWendu"] - df["yWendu"]
```

2.apply方法

添加一列温度类型：

1. 如果最高温度大于33度就是高温
2. 低于-10度是低温
3. 否则是常温

```
def get_wendu_type(x):
    if x["bWendu"] > 33:
        return '高温'
    if x["yWendu"] < -10:
        return '低温'
    return '常温'

# 注意需要设置axis==1，这是series的index是columns
df.loc[:, "wendu_type"] = df.apply(get_wendu_type, axis=1)
# 查看温度类型的计数
df["wendu_type"].value_counts()
```

3.assign方法

将温度从摄氏度变成华氏度

```
# 可以同时添加多个新的列
# 摄氏度转华氏度
df.assign(
    yWendu_huashi = lambda x : x["yWendu"] * 9 / 5 + 32,
    bWendu_huashi = lambda x : x["bWendu"] * 9 / 5 + 32
)
```

4. 按条件选择分组分别赋值

按条件先选择数据，然后对这部分数据赋值新列
实例：高低温差大于10度，则认为温差大

```python
# 先创建空列（这是第一种创建新列的方法）
df['wencha_type'] = ''

df.loc[df["bWendu"]-df["yWendu"]>10, "wencha_type"] = "温差大"

df.loc[df["bWendu"]-df["yWendu"]<=10, "wencha_type"] = "温差正常"

df["wencha_type"].value_counts()
```

五、缺失值处理

`Pandas`使用这些函数处理缺失值：

- `isnull`和`notnull`：检测是否是空值，可用于`df`和`series`
- `dropna`：丢弃、删除缺失值
- `axis`：删除行还是列，`{0 or 'index', 1 or 'columns'}`
    - `how`：如果等于any则任何值为空都删除，如果等于all则所有值都为空才删除
    - `inplace`：如果为`True`则修改当前`df`，否则返回新的`df`
- `fillna`：填充空值
- `value`：用于填充的值，可以是单个值，或者字典（key是列名，value是值）
- `method`：等于`ffill`使用前一个不为空的值填充`forword fill`；等于`bfill`使用后一个不为空的值填充`backword fill`
- `axis`：按行还是列填充，`{0 or 'index', 1 or 'columns'}`
- `inplace`：如果为`True`则修改当前`df`，否则返回新的`df`

```python
# 1.读取数据
df = pd.read_excel('res/student_excel.xlsx',skiprows=2)
# 2.检测空值
df.isnull()
df['分数'].isnull()
df['分数'].notnull()
df.loc[df['分数'].notnull()] # 筛选没有空分数的所有行
# 3.删除全是空值的列
df.dropna(axis=1,how='all',inplace=True)
# 4.删除全是空值的行
df.dropna(axis=0,how='all',inplace=True)
# 6.将分数列为空的填充为0分
df.fillna({'分数':0},inplace=True)
# 等同于
df.loc[:,'分数'] = df['分数'].fillna(0)
# 7.填充姓名的缺失值
# 使用前面的有效值填充，用ffill：forward fill
df.loc[:, '姓名'] = df['姓名'].fillna(method="ffill")
# 8.将清洗好的数据保存
df.to_excel('res/student_excel_clean.xlsx',index=False)
```