**关键词和导入**

在这个速查卡中，我们会用到一下缩写：

|  |  |
| --- | --- |
| df | 二维的表格型数据结构DataFrame |
| s | 一维数组Series |

您还需要执行以下导入才能开始：

* import pandas as pd
* import numpy as np

**导入数据**

|  |  |
| --- | --- |
| pd.read\_csv(filename) | 导入CSV文档 |
| pd.read\_table(filename) | 导入分隔的文本文件 (如TSV) |
| pd.read\_excel(filename) | 导入Excel文档 |
| pd.read\_sql(query, connection\_object) | 读取SQL 表/数据库 |
| pd.read\_json(json\_string) | 读取JSON格式的字符串, URL或文件. |
| pd.read\_html(url) | 解析html URL，字符串或文件，并将表提取到数据框列表 |
| pd.read\_clipboard() | 获取剪贴板的内容并将其传递给read\_table（） |
| pd.DataFrame(dict) | 从字典、列名称键、数据列表的值导入 |

**输出数据**

|  |  |
| --- | --- |
| df.to\_csv(filename) | 写入CSV文件 |
| df.to\_excel(filename) | 写入Excel文件 |
| df.to\_sql(table\_name, connection\_object) | 写入一个SQL表 |
| df.to\_json(filename) | 写入JSON格式的文件 |

**创建测试对象**

用于测试的代码

|  |  |
| --- | --- |
| pd.DataFrame(np.random.rand(20,5)) | 5列、20行的随机浮动 |
| pd.Series(my\_list) | 从可迭代的my\_list创建一维数组 |
| df.index=pd.date\_range('1900/1/30', periods=df.shape[0]) | 添加日期索引 |

**查看/检查数据**

|  |  |
| --- | --- |
| df.head(n) | 数据框的前n行 |
| df.tail(n) | 数据框的后n行 |
| df.shape() | 行数和列数 |
| df.info() | 索引，数据类型和内存信息 |
| df.describe() | 数值列的汇总统计信息 |
| s.value\_counts(dropna=False) | 查看唯一值和计数 |
| df.apply(pd.Series.value\_counts) | 所有列的唯一值和计数 |

**选择**

|  |  |
| --- | --- |
| df[col] | 返回一维数组col的列 |
| df[[col1, col2]] | 作为新的数据框返回列 |
| s.iloc[0] | 按位置选择 |
| s.loc['index\_one'] | 按索引选择 |
| df.iloc[0,:] | 第一行 |
| df.iloc[0,0] | 第一列的第一个元素 |

**数据清洗**

|  |  |
| --- | --- |
| df.columns = ['a','b','c'] | 重命名列 |
| pd.isnull() | 检查空值，返回逻辑数组 |
| pd.notnull() | 与pd.isnull()相反 |
| 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) | 删除所有小于n个非空值的行 |
| df.fillna(x) | 用x替换所有空值 |
| s.fillna(s.mean()) | 将所有空值替换为均值（均值可以用统计部分中的几乎任何函数替换） |
| s.astype(float) | 将数组的数据类型转换为float |
| s.replace(1,'one') | 将所有等于1的值替换为'one' |
| s.replace([1,3],['one','three']) | 将所有1替换为'one'，将3替换为'three' |
| df.rename(columns=lambda x: x + 1) | 批量重命名列 |
| df.rename(columns={'old\_name': 'new\_ name'}) | 选择重命名 |
| df.set\_index('column\_one') | 更改索引 |
| df.rename(index=lambda x: x + 1) | 批量重命名索引 |

**筛选，排序和分组**

|  |  |
| --- | --- |
| df[df[col] > 0.5] | col列大于0.5的行 |
| df[(df[col] > 0.5) & (1.7)] | 0.7> col> 0.5的行 |
| df.sort\_values(col1) | 将col1按升序对值排序 |
| df.sort\_values(col2,ascending=False) | 将col2按降序对值排序 |
| df.sort\_values([col1,ascending=[True,False]) | 将col1按升序排序，然后按降序排序col2 |
| df.groupby(col) | 从一列返回一组对象的值 |
| df.groupby([col1,col2]) | 从多列返回一组对象的值 |
| df.groupby(col1)[col2] | 返回col2中的值的平均值，按col1中的值分组（平均值可以用统计部分中的几乎任何函数替换） |
| df.pivot\_table(index=col1,values=[col2,col3],aggfunc=max) | 创建一个数据透视表，按col1分组并计算col2和col3的平均值 |
| df.groupby(col1).agg(np.mean) | 查找每个唯一col1组的所有列的平均值 |
| data.apply(np.mean) | 在每个列上应用函数 |
| data.apply(np.max,axis=1) | 在每行上应用一个函数 |

**加入/合并**

|  |  |
| --- | --- |
| df1.append(df2) | 将df1中的行添加到df2的末尾（列数应该相同） |
| df.concat([df1, df2],axis=1) | 将df1中的列添加到df2的末尾（行数应该相同） |
| df1.join(df2,on=col1,how='inner') | SQL类型的将df1中的列与df2上的列连接，其中col的行具有相同的值。 可以是“左”，“右”，“外”，“内”连接 |

**统计**

以下这些都可以应用于一个数组。

|  |  |
| --- | --- |
| df.describe() | 数值列的汇总统计信息 |
| df.mean() | 返回所有列的平均值 |
| df.corr() | 查找数据框中的列之间的相关性 |
| df.count() | 计算每个数据框的列中的非空值的数量 |
| df.max() | 查找每个列中的最大值 |
| df.min() | 查找每列中的最小值 |
| df.median() | 查找每列的中值 |
| df.std() | 查找每个列的标准差 |