

Basic Function

<code>df = pd.read_csv("filename")</code>	读取csv文件
<code>df.info()</code>	返回每一列的非空行数、数据类型
<code>df.dtypes</code>	返回每个字段的类型
<code>df.describe()</code>	返回每一列的数量、均值、方差、最值、四分位值
<code>df.columns</code>	返回所有列名
<code>df.head(5)</code>	返回前5行的全部信息
<code>df.columns = [each.strip(), for each in df.columns]</code>	去除列名前后的空格，并更新列名

Basic Calculation on One Column

<code>df.loc[:, 'columnName'].std()</code>	返回某一列的标准差。 <code>var()</code> 表示方差
<code>df.loc[:, 'columnName'].mean()</code>	返回某一列的均值
<code>df.ID.unique()</code>	返回ID列的所有不重复值
<code>df.ID.nunique()</code>	不重复的ID值数量
<code>df.loc[:, 'columnName'].max()</code>	某一列的最大值
<code>df['columnName'].mode()</code>	某一列的众数
<code>df.ColumnName.value_counts()</code>	分组统计某一列不同值的数量，并从高到低排序
<code>df.ColumnName.value_counts()[:1]</code>	取数量最大的一行
<code>df.groupby('col1').col2.sum()</code>	根据col1的值分组，统计col2的和
<code>df.groupby("col1").col2.mean().sort_values()</code>	分组后取均值，并由低到高排序
<code>df.groupby("month").emp_combined_income.mean().sort_values(ascending=False)</code>	分组后取均值，由高到低排序
<code>df.groupby("col1").col2.mean().sort_values()[:1]</code>	排序后取第一个值（最小值）

Create new column/ Extract column

<code>df['NewColumn'] = df.ColumnName.apply(lambda x: "No" if x != 0 else "Yes")</code>	将某一列的值逻辑判断的结果存放在新列
<code>df['NewColumn'] = pd.cut(df.BMI, bins = [a,b,c,d,e], labels=["label1", "label2", "label3", "4"])</code>	根据某列值的返回添加标签（a至b属于label1）
<code>df_a = df[df.ColumnName3.isna() == False].loc[:, ["Column1", "Column2"]]</code>	依据条件提取某两列组成新的dataframe
<code>df = df.drop('ColumnName', axis = 1)</code>	删除某一列

Time-related manipulation

<code>df = pd.read_csv("filename", parse_dates=['Date'])</code>	读取文件的同时将Date列转化成指定日期格式
<code>df = pd.read_csv("employment_rate.csv", parse_dates=[["year", "month", "day"]])</code>	把分别存放年、月、日的三列合并为同一列
<code>df['year'] = df.Date.dt.year</code>	提取Date列的年份存放在新列year
<code>df['month'] = df.Date.dt.month</code>	提取Date列的月份存放在新列month
<code>df['day'] = df.Date.dt.day</code>	提取Date列的日期存放在新列day
<code>df['day_name'] = df.Date.dt.day_name()</code>	根据Date计算星期信息存放在新列day_name
<code>dfa['Date'] = [datetime.strptime(x, "%b-%y") for x in df.Period]</code>	把形式为Jan-2020的日期列转化成标准日期的格式

Correlation Analysis

返回所有列和目标列的相关系数，并按从低到高排序（所有列必须都是数值型，反映两个dataframe之间的相关关系）	<code>df.corrwith(df.targetColumn).sort_values()</code>
例子：指定列与目标列的相关系数，并排序	<code>selected_col = ["a", "b", "c"]</code>



By **Molly_6075**
cheatography.com/molly-6075/

Not published yet.
 Last updated 16th October, 2023.
 Page 1 of 2.

Sponsored by **CrosswordCheats.com**
 Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

Correlation Analysis (cont)

```
df[selected_col].corrwith(df.targetC-  
olumn).sort_values()
```

一个dataframe中，所有列之
间的两两相关关系

```
df[selected_col].corr(method="spea-  
rman")
```

```
df.loc[:, ["a", "b", "c"]].corr()
```

Plot

```
df.groupby("column1").column2.me-  
an().plot(kind = 'bar')
```

柱状图

```
df.column1.plot(kind = 'hist')
```

频数分布直方图

```
df.plot(kind="line", y="column1", x="c-  
olumn2")
```

普通折线图

```
df.groupby("column3").colu-  
mn1.mean().plot()
```

分组统计后的折线图

```
df.groupby("column4")[["column1", "c-  
olumn2", "column3"]].mean().plot()
```

多个列用相同标准分组，并
将折线展示在同一个图中

Complex Plot

```
from pandas.plotting  
import lag_plot
```

绘制时间序列数据的滞后图

```
lag_plot(df, lag = 2)
```

默认的滞后期=1,如果数据存在自相关性，
将沿着某条对角线分布

```
from pandas.plotting import*  
autocorrelation_plot(df)
```

观察时间序列数据中的自相关性



By **Molly_6075**
cheatography.com/molly-6075/

Not published yet.
Last updated 16th October, 2023.
Page 2 of 2.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>