pandas

October 11, 2023

Author: lcdse7en
Email: 2353442022@qq.com

摘要

Pandas是Python的核心数据分析支持库，提供了快速、灵活、明确的数据结构，旨在简单、直观地处理关系型、标记型数据。Pandas的目标是成为 Python 数据分析实践与实战的必备高级工具，其长远目标是成为最强大、最灵活、可以支持任何语言的开源数据分析工具。经过多年不懈的努力，Pandas离这个目标已经越来越近了。

Pandas的主要数据结构是Series（一维数据）与DataFrame（二维数据），这两种数据结构足以处理金融、统计、社会科学、工程等领域里的大多数典型用例。对于R用户，DataFrame提供了比R语言data.frame更丰富的功能。Pandas基于NumPy开发，可以与其它第三方科学计算支持库完美集成。

Pandas 就像一把万能瑞士军刀，下面仅列出了它的部分优势 ：

- 处理浮点与非浮点数据里的缺失数据，表示为 NaN；
- 大小可变：插入或删除 DataFrame 等多维对象的列；
- 自动、显式数据对齐：显式地将对象与一组标签对齐，也可以忽略标签，在 Series、DataFrame 计算时自动与数据对齐；
- 强大、灵活的分组（group by）功能：拆分-应用-组合数据集，聚合、转换数据；
- 把 Python 和 NumPy 数据结构里不规则、不同索引的数据轻松地转换为 DataFrame 对象；
- 基于智能标签，对大型数据集进行切片、花式索引、子集分解等操作；
- 直观地合并（merge）、连接（join）数据集；
- 灵活地重塑（reshape）、透视（pivot）数据集；
- 轴支持结构化标签：一个刻度支持多个标签；
- 成熟的 IO 工具：读取文本文件（CSV 等支持分隔符的文件）、Excel 文件、数据库等来源的数据，利用超快的 HDF5 格式保存 / 加载数据；
- 时间序列：支持日期范围生成、频率转换、移动窗口统计、移动窗口线性回归、日期位移等时间序列功能。

https://github.com/lcdse7en/pandas

Basic Operations

Series.

Series Method and Properties

DataFrame Method	Issue
Series.head()	Series.head(int)
Series.tail()	Series.tail(int)
Series.unique()	Series.unique() -> 去重, 返回array
Series.isnull()	Series.isnull()
Series.notnull()	Series.notnull()
Series.dtype	Series.dtype -> 数据类型
Series.tolist()	Series.tolist()
Series.value_counts()	Series.value_counts()
Series.nlargest()	Series.nlargest(int)
Series.nsmallest()	Series.nsmallest(int)
Series.str()	Series.str.replace("old", "new")
	…
	…

Series.map(dic) - 映射

```
# map --> 映射
dic = {
    "py": "python",
    "js": "javascript",
}
df1 = df["lang"].map(dic)

# map --> 运算
def after_sal(s):
    return s - (s-3000)*0.5
df["after_sal"] = df["salary"].map(after_sal)
```

DataFrame Method	Example	Isuue
Series.map()	Series.map(dict)	映射
	Series.map(func)	运算

DataFrame.

DataFrame() - Structure DataFrame(构造 DataFrame)

- data: dict or Two-dimensional array
- columns: list of string ,设置列的显式索引
- index: list of string ,设置行的显式索引

```
import numpy as np
from pandas import DataFrame

df = DataFrame(data=np.random.randint(low=0, high=100, size(8,4)))
df1 = df[["column1 Name", "column2 Name"]] --> 取多列
df1 = df.iloc[[0, 3, 5]] --> 取多行
df1 = df[0:2] --> 取1-2行
df1 = df.iloc[:, 0:2] --> 取1-2列 iloc: [
```

DataFrame Method	Example	Isuue
df.replace()	df.replace(to_replace=2, value="Two")	全局替换
	df.replace(to_replace={4,2}, value="Two")	将第四列的2替换为Two

Pandas read_file and DataFrame to_file.

pd.read_csv()
- filepath_or_buffer: string
- sep: character, default “,”
- usecols: list of string, use name of columns
- encoding: string, “utf-8”, “GBK”
- index_col: int, sequence or boolean, optional. default None, use index_col=False
- header: int or None, default 0
- names: list of string, add custom columns name, use: header=None

df.to_excel()

```
df.to_excel(
    excel_writer = "test.xlsx",
    sheet_name = "test",
    index = False,
    freeze_panes = (1,1)
)
```

Get DataFrame the number of rows and columns.

```
# get rows
len(df)
df.shape[0]
# get columns
df.shape[1]
len(df.columns)
```

DataFrame insert column.

df.insert()
- loc: int
- column: string
- value: int, Series or array-like
- allow_duplicates: bool, default False

```
# method one
df.insert(
    loc=0,
    column="ID",
    value=range(1, len(df) + 1)
)
# method two
df["ID"] = range(1, len(df) + 1)
```

DataFrame sort.

df.sort_values()
- by: string or list of string
- ascending: boolean or list of boolean, False: descending, True: ascending
- inplace: boolean

Advanced Operations

Data Cleaning (数据清洗).

df.fillna() - (空值[NaN]处理: 填充空值)

- method: string, “ffill” (向前填充), “bfill” (向后填充)
- axis: int, 1(row), 0(column)
- value: int, string

```
import numpy as np
from pandas import DataFrame

df = DataFrame(data=np.random.randint(low=0, high=100, size(3,5)))
df.iloc[1,2] = None --> (np.nan) NaN

# 查看存在空值的行
print(df.loc[df.isnull().any(axis=1)])
# 查看不存在空值的行
print(df.loc[df.notnull().all(axis=1)])

# 空值(NaN)填充
df1 = df.fillna(method="ffill", axis=0)
# 检测是否填充完整
print(df1.isnull().any(axis=0)) --> boolean
```

df.drop() - (空值[NaN]处理: 删除空值行)

- labels: list of int64Index
- axis: 0(row), 1(column)

```
import numpy as np
from pandas import DataFrame

df = DataFrame(data=np.random.randint(low=0, high=100, size(3,5)))
df.iloc[1,2] = None --> (np.nan) NaN

# 查看存在空值的行
print(df.loc[df.isnull().any(axis=1)])
# 查看不存在空值的行
print(df.loc[df.notnull().all(axis=1)])

# 空值(NaN)删除
drop_index = df.loc[df.isnull().any(axis=1)].index
df1 = df.drop(labels=drop_index, axis=0, inplace=True) --> df.drop*: 0(row), 1(column)

# 检测是否填存在空值
print(df1.isnull().any(axis=0)) --> boolean
```

df.dropna() - (可以直接将存在缺失值[NaN]的行或者列删除)

- axis: 0(row), 1(column)

```
df1 = df.dropna(axis=0)
```

df.drop_duplicate() - Handle duplicate data (删除重复数据)

- keep: “first”, “last”, False

```
df1 = df.drop_duplicate(keep="first")
```

pd.concat() - Cascade (内-外级联).

pd.concat()

- objs:
- axis: 0(column), 1(row)
- join: inner, outer, default inner
- ignore_index: boolean, default False

```
df = pd.concat(objs=(df1, df2), axis=1)  --> col + col
# 不匹配级联
df = pd.concat(objs=(df1, df2), axis=0, join="outer") --> row + row --> + NaN
```

pd.merge() - Merge (合并).

pd.merge()
- left: DataFrame
- right: DataFrame
- on: string of column name, default None
- how: “inner”, “outer”, “left”, “right”, default “inner”
- left_on: string of column name
- right_on: string of column name

```
df = pd.merge(left=df1, right=df2, on="column Name", how="outer")
```

pd.apply().

Apply a function along an axis of the DataFrame.

Objects passed to the function are Series objects whose index is either the DataFrame’s index (axis=0) or the DataFrame’s columns (axis=1)

pd.apply()

```
def get_type(x):
    if x["bwendu"] > 33:
        return "高温"
    if x["ywendu"] < -10:
        return "低温"
    return "常温"

df.loc[:,"wendu_type"] = df.apply(get_type, axis=1)
```

Grouped aggregations (分组聚合).