- AI 使用说明
 - 通义千问-Qwen3
 - 对话 1 (工具使用询问)
 - 📌 一、基本语法
 - 📌 二、准备数据(示例)
 - 📌 三、常用操作示例
 - 1. 按单列分组并聚合
 - 2. 按多列分组
 - 3. 多种聚合函数
 - 4. 对多列应用不同聚合函数
 - 5. 使用自定义函数
 - 6. 分组后筛选 (filter)
 - 7. 分组后转换(transform)
 - 8. 查看每组内容
 - 9. 获取某组数据
 - 📌 四、常用聚合函数
 - ★ 五、注意事项
 - ✓ 总结
 - 采纳情况
 - 对话 2 (工具使用询问)
 - 🗸 基本语法
 - 🖈 示例: 计算两列的和、条件判断、自定义逻辑
 - 1. 准备数据
 - 2. 方法一: 使用 lambda 函数 (简单操作)
 - 3. 方法二: 使用自定义函数(推荐, 更清晰)
 - 4. 方法三: 带条件判断(如: 分类)
 - 5. 方法四: 返回多个值(创建多列)
 - 替误用法(避免)
 - **V** 高级技巧: 使用 numpy.where 替代 apply (性能更好)
 - ✓ 总结
 - 采纳情况
 - 对话 3 (遇到难以突破的困难, 试图借助 AI 帮忙解答)
 - 一、数据集中在竖直或水平线的含义
 - 二、如何检验相关性?
 - 1. 计算相关系数
 - 2. 可视化辅助分析
 - 三、如何检验显著性?

- 1. 显著性检验(p值)
- 2. 注意事项
- 四、实际案例分析
 - 案例 1: 数据集中在竖直线上
 - 案例 2: 数据集中在水平线上
- 五、进一步分析建议
 - 1. 检查变量是否固定
 - 2. 探索非线性关系
 - 3. 分组分析
- 六、总结
 - 采纳情况
- 豆包
 - 对话 4 (代码报错修复)
 - 解决方法
 - 原因说明
 - 采纳情况
 - 对话 5 (辅助引用修改格式)
 - 说明:
 - 采纳情况

AI 使用说明

通义千问-Qwen3

对话1(工具使用询问)

Q: pandas gropby 如何使用

A: pandas 中的 groupby() 是一个**极其强大且常用**的功能,用于对数据进行"分组-应用-合并"(Split-Apply-Combine)操作。它类似于 SQL 中的 GROUP BY,可以按一个或多个列的值将数据分组,然后对每个组应用聚合、转换或过滤操作。



```
df.groupby('列名')
df.groupby(['列1', '列2'])
```

返回一个 DataFrameGroupBy 对象,需要再调用聚合函数(如.sum(),.mean()等) 才能得到结果。



★ 二、准备数据(示例)

```
import pandas as pd
data = {
   '班级': ['A', 'A', 'B', 'B', 'A', 'B'],
   '姓名':['张三','李四','王五','赵六','钱七','孙八'],
   '性别':['男', '女', '男', '女', '男', '男'],
   '数学': [85, 92, 78, 88, 90, 82],
   '英语': [76, 85, 80, 90, 88, 75]
}
df = pd.DataFrame(data)
print(df)
```

输出:

```
班级 姓名 性别 数学 英语
0 A 张三 男 85 76
1 A 李四 女 92 85
2 B 王五 男 78 80
3 B 赵六 女 88 90
4 A 钱七 男 90 88
5 B 孙八 男 82 75
```



★ 三、常用操作示例

1. 按单列分组并聚合

```
# 按"班级"分组,计算数学平均分
df.groupby('班级')['数学'].mean()
```

输出:

```
班级
A 89.0
B 82.7
Name: 数学, dtype: float64
```

2. 按多列分组

```
# 按"班级"和"性别"分组 , 计算数学平均分 df.groupby(['班级', '性别'])['数学'].mean()
```

输出:

```
班级 性别
A 男 87.5
女 92.0
B 男 80.0
女 88.0
Name: 数学, dtype: float64
```

3. 多种聚合函数

```
# 使用 .agg() 进行多种聚合
df.groupby('班级')['数学'].agg(['mean', 'sum', 'count', 'std'])
```

输出:

```
mean sum count std
班级
A 89.0 267 3 3.605551
B 82.7 248 3 5.128226
```

4. 对多列应用不同聚合函数

输出:

```
数学 英语
mean sum min max
班级
A 89.0 267 76 88
B 82.7 248 75 90
```

5. 使用自定义函数

```
# 自定义函数: 极差

def range_func(x):
    return x.max() - x.min()

df.groupby('班级')['数学'].agg(range_func)
```

输出:

```
班级
A 7
B 10
Name: 数学,dtype: int64
```

或使用 lambda:

```
df.groupby('班级')['数学'].agg(lambda x: x.max() - x.min())
```

6. 分组后筛选(filter)

保留数学平均分 > 85 的班级:

```
df.groupby('班级').<mark>filter</mark>(lambda x: x['数学'].mean() > 85)
```

输出:

```
班级 姓名 性别 数学 英语
0 A 张三 男 85 76
1 A 李四 女 92 85
4 A 钱七 男 90 88
```

7. 分组后转换(transform)

对每组数据进行标准化(减去组均值):

```
df['数学_组内偏差'] = df.groupby('班级')['数学'].transform(lambda x: x - x.mean())
print(df)
```

输出:

```
班级 姓名 性别 数学 英语 数学_组内偏差
0 A 张三 男 85 76 -4.0
1 A 李四 女 92 85 3.0
2 B 王五 男 78 80 -4.7
3 B 赵六 女 88 90 5.3
4 A 钱七 男 90 88 1.0
5 B 孙八 男 82 75 -0.7
```

8. 查看每组内容

```
for name, group in df.groupby('班级'):
    print(f"班级: {name}")
    print(group)
    print("-" * 20)
```

9. 获取某组数据

```
group_A = df.groupby('班级').get_group('A')
print(group_A)
```



★ 四、常用聚合函数

函数	说明
.mean()	平均值
.sum()	求和
.count()	计数 (非空)
.size()	计数 (含空值)
.std()	标准差
.var()	方差
.min()/.max()	最小/最大值
.first()/.last()	第一个/最后一个值
.median()	中位数
.agg()	自定义聚合



★ 五、注意事项

- 1. groupby 后返回的是一个 GroupBy 对象,必须调用聚合函数才会计算。
- 2. count() 不统计 NaN, size() 统计所有行(包括 NaN)。
- 3. 多列分组时传入列表 ['col1', 'col2']。
- 4. 可以链式操作: df.groupby(...).agg(...).reset_index() 转回普通 DataFrame.



总结

groupby 是 pandas 数据分析的核心工具,典型流程:

df.groupby('分组列')[目标列].agg(聚合函数)

适用于:

- 分组统计(均值、总和等)
- 数据清洗与转换
- 特征工程
- 生成透视表前的准备

掌握 groupby 是成为 Pandas 高手的关键一步!

采纳情况

通过本次对话,我们学会了如何使用 pandas 的 groupby 方法,这次对话加快了我们的代码验证过程,在代码中予以采纳。

对话 2 (工具使用询问)

Q:pandas 如何对两列使用 apply 函数

A:在 pandas 中,如果你想对 两列(或多列)同时应用一个函数,可以使用 .apply() 方法,并配合 axis=1 参数,表示按 行 进行操作。



基本语法

df[['列1', '列2']].apply(你的函数, axis=1)

- axis=1: 表示按 行 遍历(即每一行的两个值传入函数)
- axis=0: 默认按列遍历(不适用于此场景)

★ 示例: 计算两列的和、条件判断、自定义逻辑

1. 准备数据

```
import pandas as pd

df = pd.DataFrame({
    'A': [1, 2, 3, 4],
    'B': [5, 6, 7, 8],
    'C': ['x', 'y', 'z', 'w']
})

print(df)
```

输出:

```
A B C
0 1 5 x
1 2 6 y
2 3 7 z
3 4 8 w
```

2. 方法一: 使用 lambda 函数 (简单操作)

```
# 两列相加
df['A+B'] = df[['A', 'B']].apply(lambda x: x['A'] + x['B'], axis=1)

# 或者使用 x[0] + x[1](按位置)
df['A+B_v2'] = df[['A', 'B']].apply(lambda x: x[0] + x[1], axis=1)

print(df)
```

3. 方法二: 使用自定义函数(推荐,更清晰)

```
def my_func(row):
   a = row['A']
   b = row['B']
    return a * b + 10 # 自定义逻辑: A*B + 10
df['A*B+10'] = df.apply(my_func, axis=1)
```

4. 方法三: 带条件判断(如:分类)

```
def classify(row):
    if row['A'] > 2 and row['B'] < 8:
       return '条件满足'
   else:
       return '不满足'
df['分类'] = df.apply(classify, axis=1)
```

5. 方法四:返回多个值(创建多列)

```
def compute_stats(row):
   total = row['A'] + row['B']
   diff = row['A'] - row['B']
   return pd.Series([total, diff]) #返回两个值
df[['总和', '差值']] = df.apply(compute_stats, axis=1)
```

最终结果:

```
A B C A+B A+B_v2 A*B+10 分类 总和 差值
0 1 5 x 6
            6
                15 不满足 6 -4
                22 不满足 8 -4
1 2 6 y 8
            8
2 3 7 z 10
                31 条件满足 10 -4
            10
3 4 8 w 12
            12
                 42 不满足 12 -4
```



错误用法(避免)

X 错误:axis=0 是按列操作,不能同时传两列 df[['A','B']].apply(my_func, axis=0) # 会分别对A列和B列调用函数

☑ 高级技巧: 使用 numpy.where 替代 apply(性能更好)

如果只是简单条件判断,推荐用 np.where:

```
import numpy as np
df['标签'] = np.where((df['A'] > 2) & (df['B'] < 8), '满足', '不满足')
```

比 apply 更快,适合大数据集。



✓ 总结

场景	推荐方法	
简单计算(如加减)	df['A'] + df['B'] (向量化)	
简单条件	np.where()	
复杂逻辑、多列交互	df.apply(func, axis=1)	
返回多列	return pd.Series([值1, 值2])	

☑ 核心要点:

- 使用 axis=1 才能按行处理两列数据
- 函数参数是每一行的 Series 对象
- 适用于复杂逻辑,但大数据集注意性能

掌握这个技巧, 你就能灵活处理多列之间的交互逻辑了!

采纳情况

通过本次对话,我们学会了使用 pandas 的 apply 操作方法,在代码中很有帮助,予以采纳。

对话 3 (遇到难以突破的困难, 试图借助 AI 帮忙解答)

Q: 通过观察图表发现数据比较集中但集中在某一竖直或水平线附近如何检验其相 关性或是否存在相关性?如果可以检验如何检验其显著性?

A: 当数据集中在某一 **竖直** 或 **水平线** 附近时,通常意味着两个变量之间 **没有线性相关性**,但可能存在 **非线性关系** 或 **单变量主导** 的模式。以下是检验相关性及显著性的方法和步骤:

一、数据集中在竖直或水平线的含义

- 1. 竖直线(X固定,Y变化):
 - 。 X 值几乎不变, Y 值变化。
 - 。 例如: 固定温度下,不同时间的测量值。
 - 。 **皮尔逊相关系数(Pearson)** 会接近 **0**(无线性关系),但可能存在 **单变量 变化**。
- 2. 水平线(Y 固定, X 变化):
 - 。Y值几乎不变,X值变化。
 - 。 例如: 固定剂量下,不同个体的反应值。
 - 。 皮尔逊相关系数 同样接近 0,但可能存在 单变量主导 的模式。

二、如何检验相关性?

- 1. 计算相关系数
 - 皮尔逊相关系数(Pearson):
 - 。 适用于 线性关系。
 - 。若数据集中在竖直或水平线,皮尔逊相关系数 接近 0。

。 示例代码:

```
from scipy.stats import pearsonr
r, p = pearsonr(x, y)
print(f"Pearson r = {r:.3f}, p = {p:.3f}")
```

- 斯皮尔曼等级相关系数(Spearman):
 - 。 基于变量的 排名,适用于 单调关系。
 - 。 若数据集中在竖直/水平线, 斯皮尔曼相关系数也可能接近 0。
 - 。 示例代码:

```
from scipy.stats import spearmanr
r, p = spearmanr(x, y)
print(f"Spearman r = {r:.3f}, p = {p:.3f}")
```

- 肯德尔等级相关系数(Kendall):
 - 。 适用于 小样本 或 有序数据。
 - 。 示例代码:

```
from scipy.stats import kendalltau
r, p = kendalltau(x, y)
print(f"Kendall τ = {r:.3f}, p = {p:.3f}")
```

2. 可视化辅助分析

- 散点图: 直接观察数据分布。
- 核密度估计图 (KDE): 检查变量的边缘分布。
- 分组箱线图: 若 X 或 Y 固定,观察另一变量的分布差异。

三、如何检验显著性?

- 1. 显著性检验(p值)
 - 零假设: 两变量 无相关性(相关系数为 0)。
 - **备择假设**: 两变量 **存在相关性** (相关系数 ≠ 0)。

• 判断标准:

- 。 p 值 < 0.05: 拒绝零假设,认为相关性显著。
- 。 p 值 ≥ 0.05: 无法拒绝零假设,认为相关性不显著。

2. 注意事项

- 当变量完全固定(如 X 全为常数):
 - 。 无法计算相关系数 (分母为 0)。
 - 。 需检查数据是否存在 单变量主导 或 数据缺失。
- 当数据集中在竖直/水平线:
 - 。即使 p 值显著,相关系数也可能接近 0 (如噪声干扰)。
 - 。 需结合 业务逻辑 解释结果。

四、实际案例分析

案例 1:数据集中在竖直线上

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import pearsonr, spearmanr

# 生成数据:X固定为 5, Y服从正态分布
x = np.repeat(5, 100)
y = np.random.normal(0, 1, 100)

# 计算相关系数
print("Pearson:", pearsonr(x, y)) # r = 0, p = 1
print("Spearman:", spearmanr(x, y)) # r = 0, p = 1
# 可视化
plt.scatter(x, y, alpha=0.6)
plt.title("Data集中在竖直线上")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

结果解释:

- 皮尔逊和斯皮尔曼相关系数均接近 0, p 值接近 1, 说明无显著相关性。
- 数据集中在竖直线上, X 完全固定, 无法通过 X 预测 Y。

案例 2: 数据集中在水平线上

```
# 生成数据:Y固定为 10,X服从正态分布
x = np.random.normal(0, 1, 100)
y = np.repeat(10, 100)

# 计算相关系数
print("Pearson:", pearsonr(x, y)) # r ≈ 0, p ≈ 1
print("Spearman:", spearmanr(x, y)) # r ≈ 0, p ≈ 1

# 可视化
plt.scatter(x, y, alpha=0.6)
plt.title("Data集中在水平线上")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

结果解释:

- 与案例 1 类似,相关系数和 p 值均表明无显著相关性。
- 数据集中在水平线上, Y 完全固定, 无法通过 Y 预测 X。

五、进一步分析建议

1. 检查变量是否固定

- 若变量完全固定(如 X 全为常数),需检查:
 - 。 数据是否缺失或异常。
 - 。 是否应使用单变量分析(如描述性统计)。

2. 探索非线性关系

- 若数据呈现 非线性模式 (如抛物线、指数曲线), 可尝试:
 - 。多项式回归。
 - 。 **互信息(Mutual Information)**: 衡量非线性依赖。
 - 。 距离相关系数 (Distance Correlation): 适用于任意关系。

3. 分组分析

- 若数据集中在某条线上,可能是分组变量导致的:
 - 。例如:不同类别(A/B/C)的数据分别集中在不同竖直线上。

六、总结

情况	方法	结果解释
数据集中在竖直/水平线	皮尔逊/斯皮尔曼相关系数	相关系数接近 0, p 值不显著
变量完全固定	检查数据质量	单变量主导,无法分析相关性
非线性关系	互信息/距离相关系数	检测非线性依赖
分组变量	分组分析	不同类别间可能存在关系

通过结合统计检验、可视化和业务逻辑,可以更全面地判断数据中的相关性是否存在及 其显著性。

采纳情况

我们不太有判断相关性系数大小的经验,在本题遇到的情况让我们不得要领,问过 AI 解答了我们的困惑,让我们不必深究于此。

豆包

对话 4 (代码报错修复)

Q:

tmp/ipykernel_340123/3817895775.py:30: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df[col].fillna(df[col].mode()[0], inplace=True) 这是什么情况 A: 这个警告是由于 pandas 中使用链式 chained assignment(链式赋值)和 inplace=True 导致的潜在问题。具体来说,你的代码 df[col].fillna(df[col].mode()[0], inplace=True) 存在以下问题:

- 1. **链式赋值风险**: df[col] 会返回 DataFrame 的一个子集(可能是副本),然后你在这个子集上使用 inplace=True 尝试修改,这在 pandas 中是不确定的行为——有时候可能修改原始数据,有时候可能只修改了副本(导致修改无效)。
- 2. **pandas 版本兼容问题**: 从警告信息可知,这种用法在 pandas 3.0 中将彻底失效,因为中间对象(df[col])会被明确视为副本,**inplace=True** 无法作用于原始 DataFrame。

解决方法

按照警告提示,修改为直接对原始 DataFrame 操作,避免链式赋值:

```
# 替换原来的写法
# df[col].fillna(df[col].mode()[0], inplace=True)

# 新写法:直接对 df 操作,指定列名
df.fillna({col: df[col].mode()[0]}, inplace=True)

# 或者不使用 inplace(更推荐的方式,避免副作用)
df[col] = df[col].fillna(df[col].mode()[0])
```

原因说明

- df.fillna({col: 值}, inplace=True): 直接在原始 DataFrame 上操作,明确 指定要填充的列和值,避免了链式赋值的不确定性。
- 不使用 inplace=True 而是直接赋值(df[col] = ...): 这是 pandas 推荐的方式,更清晰且不易出错,因为 inplace=True 可能会导致一些隐蔽的副作用。

修改后,警告会消失,且代码在 pandas 3.0 中也能正常工作。

采纳情况

由于我们对 pandasAPI 的使用还不太熟练,在使用过程中出现了意料之外的警告,在问过 AI 后得知解决方法并在代码中采纳。

对话 5 (辅助引用修改格式)

Q:

```
[1]蒋丽雅,卢劭侃,杜佳恩,等.无创产前检测技术的发展与应用[J].临床医学研究与实践,2025,10(23):191-194.DOI:10.19347/j.cnki.2096-1413.202523047.
请将这篇引用转换为bibtex格式
```

A:

```
@article{蒋丽雅2025无创产前检测技术的发展与应用,
   author = {蒋丽雅 and 卢劭侃 and 杜佳恩 and 等},
   title = {无创产前检测技术的发展与应用},
   journal = {临床医学研究与实践},
   year = {2025},
   volume = {10},
   number = {23},
   pages = {191-194},
   doi = {10.19347/j.cnki.2096-1413.202523047}
}
```

说明:

- 1. 条目类型采用 @article (期刊论文),符合该文献的发表载体;
- 2. author 字段中,中文作者间用 and 分隔,"等"直接保留(符合中文文献 BibTeX 引用习惯);
- 3. 期刊名 journal 填写原文的中文刊名"临床医学研究与实践";
- 4. 卷(volume)、期(number)、页码(pages)、DOI 均严格对应原文引用信息,确保引用准确性。

采纳情况

由于我们使用的文章撰写排版工具是 latex, 所以需要将论文写成 bibTeX 格式, 而有些平台引用导出没有此项, 故而借助 AI 快速准确完成文献的引用。