***实训6：简单时间序列分析***

· ***目标***：掌握时间序列处理和可视化。

· ***任务***：读取时间序列CSV，计算移动平均，检测异常点，绘制折线图。

· ***详细要求***：

o 读取CSV（time_series.csv），字段：日期（YYYY-MM-DD）、值（浮点数）。

o 使用Pandas解析日期，计算7天移动平均。

o 检测异常点：值超出2倍标准差的记录。

o 使用Matplotlib绘制折线图（原始数据、移动平均），标记异常点（红色散点）。

o 保存分析结果到analyzed_series.csv，包含移动平均和异常标记。

o 验证日期格式和数据完整性，记录错误日志。

o 添加配置选项（如自定义移动平均窗口大小）。

· ***技能***：Pandas时间序列、Matplotlib、异常检测、日志记录。

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import logging
import argparse
import sys
import os

# 修复中文字体显示问题
if sys.platform == 'win32':
    plt.rcParams['font.sans-serif'] = ['SimHei']  # Windows中文支持
elif sys.platform == 'darwin':
    plt.rcParams['font.sans-serif'] = ['Arial Unicode MS']  # macOS中文支持
else:
    plt.rcParams['font.sans-serif'] = ['WenQuanYi Zen Hei']  # Linux中文支持

plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题


def main():
    # 配置参数解析器
    parser = argparse.ArgumentParser(description='时间序列分析工具')
    parser.add_argument('--window', type=int, default=7, help='移动平均窗口大小（默认：7）')
    parser.add_argument('--std_multiplier', type=float, default=2.0, help='异常检测标准差倍数（默认：2.0）')
    parser.add_argument('--no-display', action='store_true', help='不显示图表（仅保存文件）')
    args = parser.parse_args()

    # 配置日志记录
    logging.basicConfig(
        filename='analysis_errors.log',
```

```
        level=logging.ERROR,
        format='%(asctime)s - %(levelname)s - %(message)s'
    )

    try:
        # 检查文件是否存在
        if not os.path.exists('time_series.csv'):
            raise FileNotFoundError("未找到time_series.csv文件")

        # 读取CSV文件（使用英文列名）
        df = pd.read_csv('time_series.csv', parse_dates=['date'])

        # 验证数据完整性
        if df.empty:
            raise ValueError("CSV文件为空")
        if 'date' not in df.columns or 'value' not in df.columns:
            raise ValueError("CSV缺少必要列名：'date'或'value'")

        # 按日期排序并重置索引
        df = df.sort_values('date').reset_index(drop=True)

        # 检查日期格式一致性
        if not pd.api.types.is_datetime64_any_dtype(df['date']):
            raise TypeError("日期列格式错误")

        # 检查并处理缺失值
        missing_mask = df['value'].isnull()
        if missing_mask.any():
            missing_dates = df.loc[missing_mask, 'date'].dt.strftime('%Y-
%m-%d').tolist()
            error_msg = f"发现缺失值，日期：{missing_dates}，已删除这些记录"
            logging.error(error_msg)
            print(f"警告: {error_msg}")
            df = df.dropna(subset=['value'])

        # 计算移动平均
        ma_col = f'{args.window}天移动平均'
        df[ma_col] = df['value'].rolling(window=args.window,
min_periods=1).mean()

        # 检测异常点（基于移动平均和标准差）
        # 计算移动标准差
        rolling_std = df['value'].rolling(window=args.window,
min_periods=1).std()
        # 计算上下边界
        upper_bound = df[ma_col] + (args.std_multiplier * rolling_std)
        lower_bound = df[ma_col] - (args.std_multiplier * rolling_std)
        # 标记异常点
        df['is_anomaly'] = (df['value'] > upper_bound) | (df['value'] <
lower_bound)

        # 保存分析结果
        df.to_csv('analyzed_series.csv', index=False, date_format='%Y-%m-
%d')
        print("分析结果已保存到 analyzed_series.csv")

        # 可视化
        plt.figure(figsize=(14, 8))
```

```
85
86          # 绘制原始数据
87          plt.plot(df['date'], df['value'], label='原始数据', alpha=0.7,
    marker='o', markersize=4)
88
89          # 绘制移动平均
90          plt.plot(df['date'], df[ma_col], label=ma_col, color='orange',
    linewidth=2.5)
91
92          # 绘制上下边界
93          plt.fill_between(df['date'], upper_bound, lower_bound,
    color='gray', alpha=0.2,
94                           label=f'{args.std_multiplier}倍标准差范围')
95
96          # 标记异常点
97          anomalies = df[df['is_anomaly']]
98          if not anomalies.empty:
99              plt.scatter(anomalies['date'], anomalies['value'],
100                         color='red', s=100, zorder=5,
101                         edgecolors='black', label='异常点')
102             # 添加异常点标注
103             for _, row in anomalies.iterrows():
104                 plt.annotate(f"{row['value']:.1f}",
105                              (row['date'], row['value']),
106                              xytext=(0, 15),
107                              textcoords='offset points',
108                              ha='center', fontsize=9,
109                              arrowprops=dict(arrowstyle="->", color='red',
    alpha=0.7))
110
111         plt.title(f'时间序列分析 (移动平均窗口={args.window}天)', fontsize=14)
112         plt.xlabel('日期', fontsize=12)
113         plt.ylabel('数值', fontsize=12)
114         plt.legend(loc='best')
115         plt.grid(True, linestyle='--', alpha=0.5)
116
117         # 设置日期格式
118         plt.gcf().autofmt_xdate()
119         plt.tight_layout()
120
121         # 保存图表
122         plt.savefig('time_series_analysis.png', dpi=300)
123         print("可视化图表已保存到 time_series_analysis.png")
124
125         # 可选显示图表
126         if not args.no_display:
127             try:
128                 plt.show()
129             except Exception as e:
130                 print(f"显示图表时出错: {str(e)}")
131                 print("图表已保存为文件，请直接查看 time_series_analysis.png")
132
133     except FileNotFoundError as fnf:
134         error_msg = f"文件错误: {str(fnf)}"
135         logging.exception(error_msg)
136         print(error_msg)
137     except pd.errors.EmptyDataError:
138         error_msg = "CSV文件为空或格式不正确"
```

```
        logging.exception(error_msg)
        print(error_msg)
    except Exception as e:
        logging.exception("分析过程中发生致命错误")
        print(f"处理失败: {str(e)} (详细日志见analysis_errors.log)")


if __name__ == "__main__":
    main()
```