BaoStock 前复权日 K 线数据计算简介

在策略研发与回测过程中，复权数据（尤其是向前复权数据）是不可或缺的基础。虽然 BaoStock 直接提供了向前复权数据，但这并不意味着每次股票发生除权除息时都需要重新下载全量复权数据。

实际上，BaoStock 提供了每日新增的日 K 线数据以及独立的复权因子数据。只需保存这些每日增量数据，即可随时动态计算出任一时点的复权价格。BaoStock 采用的是基于“涨跌幅”的复权算法，其核心在于复权因子的运用（具体原理可参考 媒体文件:BaoStock 复权因子简介.pdf。）。

接下来，将以向前复权为例，详细展示利用复权因子进行计算的具体方法。

```python
import baostock as bs
import pandas as pd

# 初始化 baostock
lg = bs.login()
print('login respond error_code:' + lg.error_code)
print('login respond  error_msg:' + lg.error_msg)

# 定义股票代码和时间范围
stock_code = "sh.600000"  # 浦发银行
start_date = "2024-01-01"
end_date = "2026-02-27"

# 1. 获取非复权日 K 线数据
print("\n1. 获取非复权日 K 线数据...")
kline_data = bs.query_history_k_data_plus(
    stock_code,
    "date,open,high,low,close,volume",
    start_date=start_date,
    end_date=end_date,
    frequency="d",
    adjustflag="3"  # 3 表示未复权
)

# 将数据转换为 DataFrame
kline_df = kline_data.get_data()
print(f"获取到 {len(kline_df)} 条非复权日 K 线数据")
print(kline_df.head())

# 转换数据类型
kline_df["open"] = kline_df["open"].astype(float)
```

```
kline_df["high"] = kline_df["high"].astype(float)
kline_df["low"] = kline_df["low"].astype(float)
kline_df["close"] = kline_df["close"].astype(float)
kline_df["volume"] = kline_df["volume"].astype(float)

# 2. 获取复权因子数据
print("\n2. 获取复权因子数据...")
# 注意：为了获取完整的复权因子历史，需要扩大时间范围
factor_start = "2015-01-01"  # 扩大时间范围以获取历史复权因子
factor_end = end_date

adjust_factor_data = bs.query_adjust_factor(
    stock_code,
    start_date=factor_start,
    end_date=factor_end
)

# 将数据转换为 DataFrame
adjust_factor_df = adjust_factor_data.get_data()
print(f"获取到 {len(adjust_factor_df)} 条复权因子数据")
if not adjust_factor_df.empty:
    print(adjust_factor_df[['dividOperateDate',
'foreAdjustFactor']].head())

    # 转换数据类型
    adjust_factor_df['dividOperateDate'] =
pd.to_datetime(adjust_factor_df['dividOperateDate'])
    adjust_factor_df['foreAdjustFactor'] =
adjust_factor_df['foreAdjustFactor'].astype(float)

    # 按日期排序
    adjust_factor_df =
adjust_factor_df.sort_values('dividOperateDate')
else:
    print("警告：未获取到复权因子数据")

# 3. 计算复权日 K 线数据（正确的计算方法）
print("\n3. 计算复权日 K 线数据...")

if not adjust_factor_df.empty:
    # 创建复权因子查找函数
    def get_factor_for_date(trade_date):
        """
        查找小于等于交易日期的最接近的复权因子
```

```
        """
        # 查找所有小于等于交易日的复权因子
        mask = adjust_factor_df['dividOperateDate'] <= trade_date
        if mask.any():
            # 取最后一个（最接近的）
            return adjust_factor_df.loc[mask,
'foreAdjustFactor'].iloc[-1]
        else:
            # 如果没有找到，返回 1（表示不复权）
            return 1.0


    # 将 kline 的日期转换为 datetime
    kline_df['date'] = pd.to_datetime(kline_df['date'])

    # 为每个交易日查找对应的复权因子
    print("正在为每个交易日匹配复权因子...")
    kline_df['adj_factor'] =
kline_df['date'].apply(get_factor_for_date)

    # 显示复权因子匹配情况
    print("\n 复权因子匹配示例：")
    print(kline_df[['date', 'close', 'adj_factor']].head(10))

    # 计算复权数据
    kline_df["adj_open"] = kline_df["open"] * kline_df["adj_factor"]
    kline_df["adj_high"] = kline_df["high"] * kline_df["adj_factor"]
    kline_df["adj_low"] = kline_df["low"] * kline_df["adj_factor"]
    kline_df["adj_close"] = kline_df["close"] * kline_df["adj_factor"]

    print("\n 计算完成，复权后的数据：")
    print(kline_df[['date', 'open', 'adj_open', 'close', 'adj_close',
'adj_factor']].head())
else:
    print("未获取到复权因子数据，无法计算复权数据")

# 4. 从 baostock 获取向前复权数据并比较
print("\n4. 从 baostock 获取向前复权数据并比较验证...")
forward_adj_data = bs.query_history_k_data_plus(
    stock_code,
    "date,open,high,low,close,volume",
    start_date=start_date,
    end_date=end_date,
    frequency="d",
```

```
    adjustflag="2"  # 1 表示向前复权
)

# 将数据转换为 DataFrame
forward_adj_df = forward_adj_data.get_data()
print(f"获取到 {len(forward_adj_df)} 条向前复权日 K 线数据")
print(forward_adj_df.head())

# 转换数据类型
forward_adj_df['date'] = pd.to_datetime(forward_adj_df['date'])
forward_adj_df["open"] = forward_adj_df["open"].astype(float)
forward_adj_df["high"] = forward_adj_df["high"].astype(float)
forward_adj_df["low"] = forward_adj_df["low"].astype(float)
forward_adj_df["close"] = forward_adj_df["close"].astype(float)

# 保存数据（可选）
kline_df.to_csv("sh600000_calculated.csv", index=False)
forward_adj_df.to_csv("sh600000_baostock.csv", index=False)
print("\n 数据已保存到 CSV 文件")

# 退出 baostock
bs.logout()
```