基于 R 和 Wind 实现小市值翻转策略

MatrixSpk

目录

基于 Wind 数据库实现小市值轮动策略的 R 代码实现框架如下，综合参考了行业标准做法及文献中的优化方法：

1 数据准备（需连接 Wind 接口）

```
# 加载必要包
library(Rcpp)
#library(WindR)
library(PerformanceAnalytics)
library(riskParityPortfolio)
library(tidyr)
library(tidyquant)
library(lubridate)
library(dplyr)
library(data.table)
```

```
# connect wind
w.start()

# get data
stock_data <- w.wsd(
  codes = "a001010100000000",
  fields = "close,volume,mkt_cap_ard",
  beginTime = "2022-01-01",
  endTime = "2025-04-14",
  options = "Fill=Previous"
  )$Data %>%
  as.data.frame() %>%
  rename(symbol=CODE, date = DATETIME, close = CLOSE, volume = VOLUME, mkt_cap = MKT_CA
```

生成 Wind 模拟数据

```
# 没有数据源的，可以模拟数据生成数据
# 模拟数据的函数
simulate_wind_data <- function(
    codes = "a001010100000000",
```

```
    fields = c("close", "volume", "mkt_cap_ard"),
    beginTime = "2022-01-01",
    endTime = "2025-04-14",
    options = "Fill=Previous",
    n_stocks = 300,                # 新增: 股票数量参数
    annual_return = 0.15,          # 新增: 年化收益率（默认 10%）
    annual_volatility = 0.1       # 新增: 年化波动率（默认 30%）
) {

    # 生成交易日序列
    generate_trading_days <- function(beginTime, endTime) {
        all_dates <- seq.Date(as.Date(beginTime), as.Date(endTime), by = "day")
        all_dates[lubridate::wday(all_dates, week_start = 1) %in% 1:5]  # 保留工作日
    }

    # 核心优化: 使用几何布朗运动模型生成价格序列
    dates <- generate_trading_days(beginTime, endTime)
    n_dates <- length(dates)
    symbols <- sprintf("Stock%04d", 1:n_stocks)  # 生成固定数量股票

    # 参数计算
    daily_mu <- annual_return / 250             # 日收益率 = 年化收益/交易日数
    daily_sigma <- annual_volatility / sqrt(250) # 日波动率 = 年化波动率/sqrt(交易日数)
    initial_price <- 10                         # 初始价格统一设为 10

    # 生成价格矩阵（带趋势和波动）
    price_matrix <- matrix(initial_price, nrow = n_stocks, ncol = n_dates)
    for (i in 1:n_stocks) {

        stock_mu <- rnorm(1, mean=daily_mu, sd=0.001)
        stock_sigma <- abs(rnorm(1, mean=daily_sigma, sd=0.005))
        # 生成对数收益率序列
        log_returns <- rnorm(n_dates-1, mean = stock_mu, sd = stock_sigma)
```

```
        # 计算累积价格序列
        price_matrix[i, 2:n_dates] <- initial_price * exp(cumsum(log_returns))
    }
    close_prices <- round(as.vector(t(price_matrix)), 2)  # 二维转一维并保留两位小数

    # 生成其他字段
    volumes <- abs(round(rnorm(n_stocks*n_dates, mean = 1e6, sd = 5e5)))
    mkt_caps <- pmax(round(rnorm(n_stocks, mean = 1e9, sd = 5e8)), 1e8)
    mkt_caps_rep <- rep(mkt_caps, each = n_dates)

    # 构建数据表
    dt <- data.table(
        CODE = rep(symbols, each = n_dates),
        DATETIME = rep(dates, times = n_stocks),
        CLOSE = close_prices,
        VOLUME = volumes,
        MKT_CAP_ARD = mkt_caps_rep
    )

    selected_fields <- toupper(fields)
    dt[, .SD, .SDcols = c("CODE", "DATETIME", selected_fields)]
}
```

2 回测引擎的核心组件

```
# ------------------------------------------------------------
# 回测引擎核心组件
# ------------------------------------------------------------

# 生成调仓日期，这里调仓周期设为 200 天一次
generate_rebalance_dates <- function(stock_dt,hist.window=200,...) {
```

```
    beginTime <- first(stock_dt$DATETIME)+hist.window
    endTime <- last(stock_dt$DATETIME)
    all_dates <- seq.Date(as.Date(beginTime), as.Date(endTime), by = "year")
    rebalance_days <- all_dates[lubridate::wday(all_dates, week_start = 1) %in% 1:5]
    return(rebalance_days)
}
```

```
## 选股过滤条件市值最小的 30 支股票，滤掉流动性最差的 20%



select_stocks <- function(data, current_date, n = 30, liq_q = 0.2){
# 转换列名并设置键
  dt <- as.data.table(data)[
    , .(symbol = CODE, date = DATETIME, close = CLOSE,
        volume = VOLUME, mkt_cap = MKT_CAP_ARD)
  ]
  setkey(dt, date)

  dt[
    date == current_date &
    volume > quantile(volume, probs = liq_q, na.rm = TRUE),
    .(symbol = symbol, mkt_cap = mkt_cap)
  ][
    order(mkt_cap)
  ][1:n]$symbol
}
```

3 优化投资组合各成分权重

```
## 权重优化模块
```

```
optimize_weights <- function(selected_stocks, hist_data){
    returns <- hist_data[
      CODE %in% selected_stocks,
      .(symbol = CODE, date = DATETIME, close = CLOSE)
    ][
      order(symbol, date),
      ][
        , return := (close - shift(close, 1)) / shift(close, 1), by = symbol
        ][
          !is.na(return)
          ]

  ret_matrix <- dcast(returns, date ~ symbol, value.var = "return")[,-1]
  cov_matrix <- cov(ret_matrix, use = "complete.obs")

  diag(cov_matrix) <- diag(cov_matrix) + 1e-4
  rpp <- riskParityPortfolio(cov_matrix)

  setNames(rpp$w, selected_stocks)
}
```

4 组建为高性能回测引擎

```
# 高性能回测引擎

backtest <- function(stock_dt,
                     rebalance_dates,
                     n_stocks = 30,
                     cost = 0.001,...){

  returns <- numeric(length(rebalance_dates)-1)
```

```
prev_weights <- NULL


for(i in seq_along(rebalance_dates[-1])){
  current_date <- as.Date(rebalance_dates[i])
  next_date <- as.Date(rebalance_dates[i+1])

  selected_stocks<- select_stocks(stock_dt, current_date, n_stocks)

  hist_data <- stock_dt[
    DATETIME >= current_date - 120 &
    DATETIME <= current_date &
      CODE %in% selected_stocks
  ]

  weights <- optimize_weights(selected_stocks, hist_data)

       # 换手成本计算（处理新增/移除持仓）
    cost_penalty <- if (!is.null(prev_weights)) {
      # 名称对齐：确保 prev_weights 与当前 weights 顺序一致
      aligned_prev <- prev_weights[match(names(weights), names(prev_weights))]
      aligned_prev[is.na(aligned_prev)] <- 0  # 新增持仓视为零权重调入

      # 换手率计算（卖出旧权重 + 买入新权重）
      turnover <- sum(abs(weights - aligned_prev))
      turnover * cost  # 假设 cost 是单边交易费率
    } else {
      0
    }

    # 期间收益率计算
    period_ret <- hist_data[
      # 筛选时间窗口
```

```
        DATETIME >= as.Date(current_date) & DATETIME <= as.Date(next_date),
        # 按股票分组计算区间收益率
        .(ret = last(CLOSE, na.rm = TRUE)/first(CLOSE, na.rm = TRUE) - 1),
        by = .(CODE)
      ][
        # 筛选有效持仓股票
        CODE %in% names(weights),
        # 权重匹配（通过名称索引避免顺序错误）
        .(weighted_ret = sum(ret * weights[as.character(CODE)]))
      ]$weighted_ret - cost_penalty

    returns[i] <- period_ret
    prev_weights <- weights
  }

  xts::xts(returns, order.by = rebalance_dates[-1])
}
```

5 执行回测并计算策略关键指标

```
# 执行回测
stock_dt <- simulate_wind_data(beginTime = "2000-01-01",endTime = "2025-04-14",n_stocks
rebalance_dates <- generate_rebalance_dates(stock_dt)
returns <- backtest(stock_dt, rebalance_dates)

## 输出关键指标
library(PerformanceAnalytics)
metrics <- table.AnnualizedReturns(returns)
drawdown <- table.Drawdowns(returns)
```

```
## Warning in table.Drawdowns(returns): Only 1 available in the data.
```

```
cat(" 年化收益率:", round(metrics*100, 2), "%\n",
    " 最大回撤:", round(min(drawdown$Depth)*100, 2), "%\n",
    " 夏普比率:", round(SharpeRatio.annualized(returns), 2))
```

```
## 年化收益率:  %
##  最大回撤: -0.65 %
##  夏普比率: -3.14
```

6 策略优化建议

6.1 权重优化改进

可替换为最小方差组合：使用 portfolio.optim 函数计算有效前沿

6.2 加入交易量约束

限制单票权重不超过 15%（修改优化器约束条件）

6.3 流动性筛选增强

增加换手率指标筛选（需 Wind 换手率数据）

```
filter(turnover_ratio > quantile(turnover_ratio, 0.3))
```

6.4 风险控制模块

增加波动率过滤

```
filter(roll_sd(close, 20) < quantile(roll_sd(close, 20), 0.8))
```