2022.10.24 FYP 多因子策略开发 阶段小记

backtest()里会调用getOrders()循环整个测试集，即getOrders()只是一天的策略，backtest()循环n次来模拟策略进行n天的情况。

getOrders()有一个parameter可以得到循环内的天数（backtest()跑到第几次循环了）: store$iter

今天之前的想法：计算出一个固定的α值，再假定一个阈值T，看α和T之间的关系来决定某天交易的方向（做多或做空）。

至于阈值定位多少，通过循环计算每个阈值下运行改策略能得到的总收益，取总收益最高者来确定最优T值。

逻辑问题：假如股票数据共1000天，将策略运行1000天并循环得到最优T值，再取最优T值并运用到策略中，是“预测未来”的逻辑错误。

因为假如getOrders()循环到第10天，而判断第11天是买是卖的的阈值T缺失是通过总1000天的数据得出的，而实际上在第11天时根本无法知道未来(11到1000)天的股票数据，因此知道最优阈值T是个悖论。

10支股票数据是用来测试策略的，而不是根据1000天的数据求最优策略来预测这几支股票1000天后的走势。

可用解决方法：将股票数据分割in-sample为0到500天，out-sample分割为500-1000天，根据0-500天的数据求出最优阈值T，再将该T用于500-1000天的测试。

这样getOrders()是从第501天开始循环的，不会用到“未来”的数据，可以解决逻辑问题。

现核心问题：无法在getOrders()里计算总收益率(PD ratio)来得到最优阈值T。

由于getOrders()是每天的交易策略，要算出最佳阈值T，得跑出总收益需要用到backtest()。而backtest()又会调用getOrders()n次来计算第n天的收益。

于是会遇到getOrders()和backtest()互相调用的无限循环情况。

尝试了不调用backtest()将其拆开来自己写，也摆脱不了在getOrders()内调用自己getOrders()n次的无限递归情况。

getOrders()是一天的策略，在没写好一天策略的情况下又如何在getOrders()内调用自己n次模拟n天策略？

什么狗屁逻辑，头疼。凌晨1点多了。反正就是有很多bug和逻辑问题。

目前想的三个可能的解决方案：

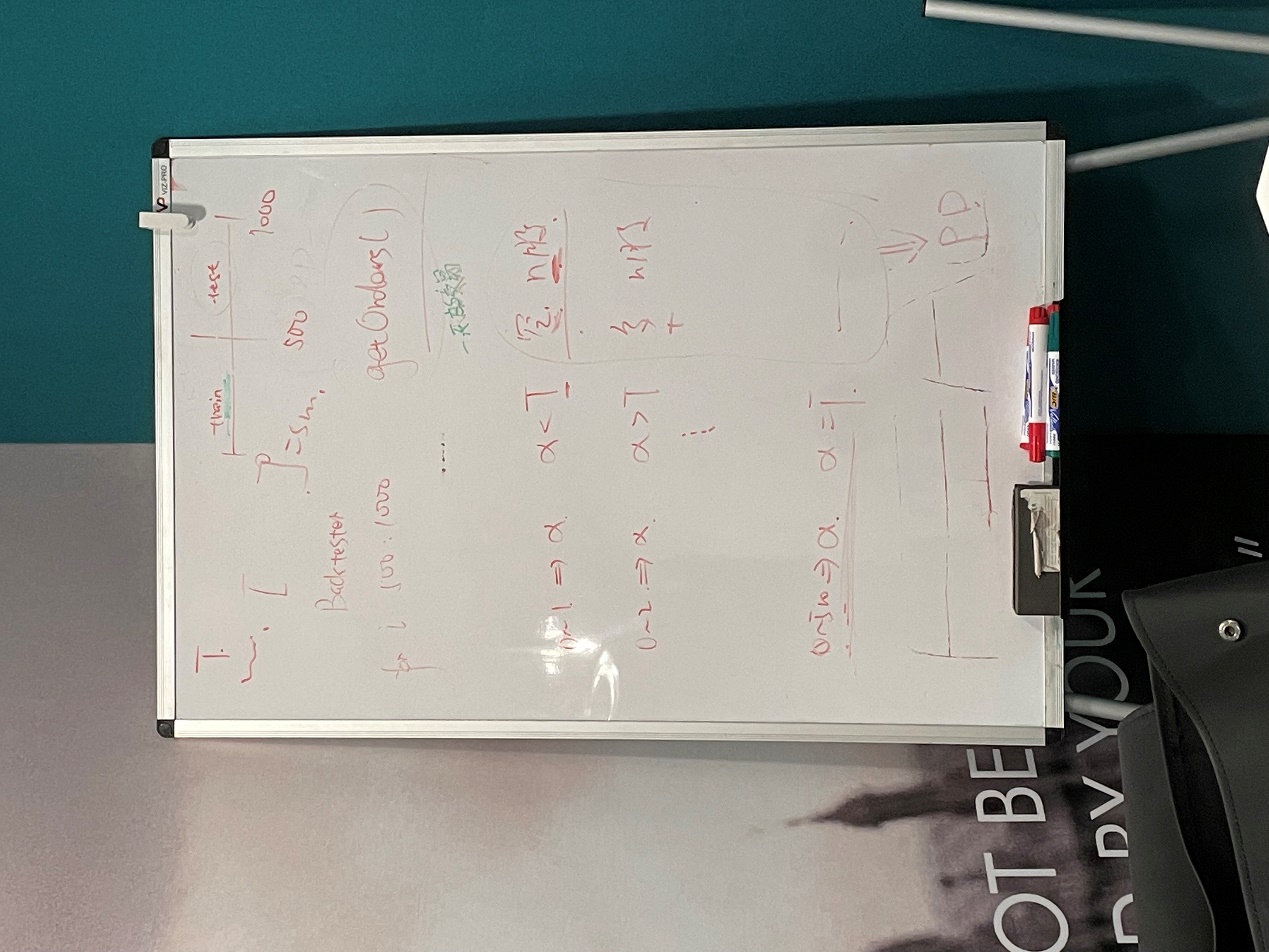
1.在getOrders()里增加一个parameter就叫T。然后在main里面跑getOrders的时候，使用循环将某范围的T当parameter输入进getOrders()。

在main里面save.csv将所有T值和其对应的PD-ratio存进一个csv里。

再另起一个R文件，得到新csv里最大PD-ratio的T值，再在新csv里重写getOrders()，并假如最佳T值。

2.在getOrders()里用公式只得到该天的收益，避免算总收益。backtest()循环得到每天的收益的累加，循环完训练集500次得到前500天的收益。

T是个getOrders()内的变量，backtest()循环后求二次函数。



当时的白板。我现在也看不懂了

3.摆烂，不求最佳阈值T了，这样就避免了在getOrders()里算总收益。T直接随便取个值，爱赚钱赚钱亏死也不关我事，反正这学期的report能讨论目前的缺陷。要优化等下学期再说。

附上目前出来的学术垃圾：

source('framework/data.R');

source('framework/backtester.R')

source('framework/processResults.R');

source('example\_strategies.R');

maxRows <- 3100 # used to initialize a matrix to store closing prices

getOrders <- function(store, newRowList, currentPos, info, params) {

#Initializing

allzero <- rep(0,length(newRowList)) # used for initializing vectors

if (is.null(store)) store <- initStore(newRowList,params$series)

store <- updateStore(store, newRowList, params$series)

marketOrders <- -currentPos; pos <- allzero

#Get data

dataList <- getData(directory="PART1")

#Initialize PD ratio and Threshold

BestThreshold <- 0

PD <- -Inf

Iteration <- 0

#Iterate through all possible alpha rate

for (a in -100:100){

#Iterate through the series in params$series

for (i in params$series){

#Get every stock's volume and closed price data

VOLUME = store$vol[,i]

CLOSE = store$cl[,i]

#Identify VWAP index

VWAP = sum(CLOSE\*VOLUME, 10)/sum(VOLUME, 10)

#Alpha006

alpha = -1\*cor(CLOSE, VOLUME, method="pearson")

#Change Position

if (alpha\*100 < a){

pos[params$series[i]] <- -1

}

else if (alpha\*100 > a ){

pos[params$series[i]] <- 1

}

else if (alpha\*100 == a){

pos[params$series[i]] <- 0

}

#Update market orders

marketOrders <- -currentPos + pos

}

#The return list of the strategy

store=store

marketOrders=marketOrders

limitOrders1=allzero

limitPrices1=allzero

limitOrders2=allzero

limitPrices2=allzero

#Store a temporary "getOrders" list

temOrders <- list(store, marketOrders, limitOrders1, limitPrices1, limitOrders2, limitPrices2)

#Calculate the result by calling backtester

sMult <- 0.20 # slippage multiplier

is\_valid\_example\_strategy <- function(strategy) {

strategy %in% example\_strategies

}

stopifnot(is\_valid\_example\_strategy(strategy))

load\_strategy(strategy) # function from example\_strategies.R

results <- backtest(dataList,temOrders,params,sMult) #Error-infinite recursive

#####################################(Calculate Result)########################################

numOfDays <- nrow(dataList[[1]])

numOfSeries <- length(dataList)

# initialise as 0-vector of length length(dataList)

newPosList <- vector(mode="numeric", length = length(dataList))

# Initialisation of getOrders with first row of data, via is.null(store)

store <- NULL #(Note: Variable "store" occur potentially bug!)

# pnlList will store trading results

# initialize lists of 0 rows; getRowList(dataList,1) used to get date for each via index()

# pnlList <- mapply(function(x, y) xts(x, index(y)),0, getRowList(dataList,1), SIMPLIFY = FALSE)

pnlList <- lapply(1:numOfSeries,function(x) matrix(0,nrow=numOfDays,ncol=1))

positionValuesList <- lapply(1:numOfSeries,function(x) matrix(0,nrow=numOfDays,ncol=1))

netWorthList <- rep(0, numOfDays)

# vector that stores a 1 for every day a position was taken in some

# series and a 0 otherwise

# initialized as all zero vecotr

posCounter <- 0

nonxtsDataList <- lapply(dataList, function(x) as.matrix(x))

balance <- 1000000

newNetWorth <- balance

netWorthList[[1]] <- balance

bankrupt <- FALSE # Are we bankrupt?

# MAIN LOOP

for (i in 2: (numOfDays-1)) { # cannot hold on day 1; day 1 data is given to strategy on day 2

oldPosList <- newPosList

info = list(balance=balance, netWorth=newNetWorth)

#Modified - Not using "getOrders" function which might occur infinite recursion

#Instead, using temporary list

x <- ?????

x <- getOrders(store, getRowList(dataList,i-1), oldPosList, info, params) #Error-infinite recursive

}

############################################################################################

pfolioPnL <- plotResults(dataList,results,plotType='ggplot2')

#Get the PD-ratio

NewPD <- pfolioPnL$fitAgg

#Compare Pd-ratio and Update the best threshold

if (NewPD>=PD){

PD <- NewPD

BestThreshold <- a/100

}

#Print Iteration

Iteration++

print("Iteration" +Iteration)

}

#Initializing Again

allzero <- rep(0,length(newRowList)) # used for initializing vectors

if (is.null(store)) store <- initStore(newRowList,params$series)

store <- updateStore(store, newRowList, params$series)

marketOrders <- -currentPos; pos <- allzero

#Apply the Best Threshold

for (i in params$series){

VOLUME = store$vol[,i]

CLOSE = store$cl[,i]

#Alpha006

alpha = -1\*cor(CLOSE, VOLUME, method="pearson")

#Change Position

if (alpha\*100 < BestThreshold){

pos[params$series[i]] <- -1

}

else if (alpha\*100 > BestThreshold){

pos[params$series[i]] <- 1

}

else if (alpha\*100 == BestThreshold){

pos[params$series[i]] <- 0

}

#Update market orders

marketOrders <- -currentPos + pos

}

return(list(store=store,marketOrders=marketOrders,

limitOrders1=allzero,limitPrices1=allzero,

limitOrders2=allzero,limitPrices2=allzero))

}

initClStore <- function(newRowList,series) {

clStore <- matrix(0,nrow=maxRows,ncol=length(series))

return(clStore)

}

updateClStore <- function(clStore, newRowList, series, iter) {

for (i in 1:length(series))

clStore[iter,i] <- as.numeric(newRowList[[series[i]]]$Close)

return(clStore)

}

initVolStore <- function(newRowList,series) {

volStore <- matrix(0,nrow=maxRows,ncol=length(series))

return(volStore)

}

updateVolStore <- function(volStore, newRowList, series, iter) {

for (i in 1:length(series))

volStore[iter,i] <- as.numeric(newRowList[[series[i]]]$Volume)

return(volStore)

}

initStore <- function(newRowList,series) {

return(list(iter=0,cl=initClStore(newRowList,series),vol=initVolStore(newRowList,series)))

}

updateStore <- function(store, newRowList, series) {

store$iter <- store$iter + 1

store$cl <- updateClStore(store$cl,newRowList,series,store$iter)

store$vol <- updateVolStore(store$vol,newRowList,series,store$iter)

return(store)

}

注意——α的表达都是错的。比如getOrders()循环到第10天，应该求前9天的α。α值每天都在变，不应该像此代码一样只算了1000天的总α。

注意——此代码包含了上述提及的无限递归问题，未解决，小心电脑炸。