Sequential Invest

Claudiu & Oz

Mon Aug 29 17:45:51 2016

#main - calls all other sequential invest modules  
  
#packages  
library(xts)

set.seed(1234) #for Monkey strategy reproducibility  
if (basename(getwd())=="Code") setwd(normalizePath("..")) #for knitr workaround  
#knitr::opts\_knit$set(root.dir="..")  
  
#source all modules and execute before trading  
#builds StockPrices dataframe  
source("Code/StockDataBuilder.R",echo=TRUE,max.deparse.length = 10000)

##   
## > SDB <- function(Regular = 0) {  
## + Market <- read.csv("DataRaw/MarketPortfolio.csv", header = T,   
## + as.is = T, skip = 3)  
## + Marketreturn <- Market[, 4]/Market[, 2]  
## + Marketreturn <- c((Marketreturn - 1) \* 100)  
## + COlNames <- c("Date", "PNominal", "Pclose", "return", "Open",   
## + "Basis")  
## + CompanyNames <- c("Teva", "Africa", "poalim", "Bezek", "cil",   
## + "Delek", "discountInvesment", "leumi", "migdal", "partner",   
## + "Mizrahi")  
## + indecies <- c(5, 3, 6, 4)  
## + Stop <- c(1:length(CompanyNames))  
## + for (i in Stop) {  
## + FileName <- paste("DataRaw/", CompanyNames[i], ".csv",   
## + sep = "")  
## + ReadindCSV <- read.csv(FileName, header = T, skip = 3,   
## + as.is = T, quote = "")[, 1:6]  
## + COlumnNames <- paste("S", i, COlNames, sep = "")  
## + colnames(ReadindCSV) <- COlumnNames  
## + if (i == 1) {  
## + DataRaw <- ReadindCSV[, c(1, indecies)]  
## + }  
## + else {  
## + DataRaw <- cbind.data.frame(DataRaw, ReadindCSV[,   
## + indecies])  
## + }  
## + if (Regular == 1) {  
## + IndicesO <- grep("Open", names(ReadindCSV))  
## + IndicesC <- grep("close", names(ReadindCSV))  
## + O2CR <- (ReadindCSV[, IndicesC]/ReadindCSV[, IndicesO] -   
## + 1) \* 100  
## + if (i == 1) {  
## + O2C <- cbind.data.frame(O2CR)  
## + }  
## + if (i != 1) {  
## + O2C <- cbind.data.frame(O2C, O2CR)  
## + }  
## + }  
## + }  
## + if (Regular == 1) {  
## + colnames(O2C) <- paste("O2Creturn", 1:i, sep = "")  
## + }  
## + bond <- rep(0, dim(DataRaw)[1])  
## + DataRaw <- cbind.data.frame(DataRaw, cashreturn = bond, Marketreturn)  
## + DataRaw$S1Date <- strptime(DataRaw$S1Date, "%d/%m/%Y")  
## + StockPrices <- xts(DataRaw[, -1], DataRaw$S1Date)  
## + if (Regular == 1) {  
## + DataRaw <- cbind.data.frame(S1Date = DataRaw$S1Date,   
## + O2C, cashreturn = bond, Marketreturn)  
## + StockPrices <- xts(DataRaw[, -1], DataRaw$S1Date)  
## + }  
## + Date <- as.Date(DataRaw$S1Date)  
## + rm(DataRaw, ReadindCSV, COlNames, FileName, CompanyNames,   
## + Stop, i, indecies, COlumnNames)  
## + save(Date, file = "DataWork/Date.Rdata")  
## + save(StockPrices, file = "DataWork/StockPrices.Rdata")  
## + StockPrices <- StockPrices  
## + }

#splits StockPrices to K-period days and returns KVec index vector  
source("Code/K\_Histogram.R",echo=TRUE,max.deparse.length = 10000)

##   
## > K\_Histogram <- function(K = 20, DDate = 20160101,   
## + Holiday = 0) {  
## + load("DataWork/StockPrices.Rdata")  
## + load("DataWork/Date.Rdata")  
## + StockData <- StockPrices  
## + Timeperiod <- paste("20060101/", DDate, sep = "")  
## + StockData <- StockData[Timeperiod, ]  
## + Date <- sort(Date, decreasing = F)  
## + Date <- Date[1:dim(StockData)[1]]  
## + LenDate <- 1:(length(Date) - K)  
## + if (Holiday == 0) {  
## + Kvec <- 0  
## + for (i in LenDate) {  
## + DateFlage <- Date[i:(i + K - 1)]  
## + Flag <- sum(DateFlage[1:(K - 1)] + 3 >= DateFlage[2:K],   
## + na.rm = T)  
## + if (Flag == K - 1) {  
## + Kvec <- c(Kvec, i)  
## + }  
## + }  
## + Kvec <- Kvec[-1]  
## + }  
## + if (Holiday == 1) {  
## + Kvec <- LenDate  
## + }  
## + return(Kvec)  
## + }

#classifer of trading days groups  
source("Code/Classification.R",echo=TRUE,max.deparse.length = 10000)

##   
## > Classification <- function(KVec, K = 2, L = 8, KKR = "Monkey") {  
## + set.seed(123)  
## + CLASS <- c("K-means", "Kernel", "Randomforest", "Monkey")  
## + load("DataWork/StockPrices.Rdata")  
## + Columns2Save <- grep("return", colnames(StockPrices))  
## + StockPrices <- StockPrices[, Columns2Save]  
## + IndexOut <- 1:length(KVec)  
## + listOfDataFrames <- list()  
## + for (i in IndexOut) {  
## + DataSet1 <- c(t(StockPrices[KVec[i]:(KVec[i] + K - 1),   
## + ]))  
## + listOfDataFrames[[i]] <- data.frame(DataSet1)  
## + }  
## + DataSet <- do.call("cbind", listOfDataFrames)  
## + Flag <- which(CLASS == KKR)  
## + if (Flag == 1) {  
## + Class <- kmeans(t(DataSet[, 1:i]), L, nstart = L)  
## + Classifier <- cbind.data.frame(KVec, Class = Class$cluster)  
## + ClassCenters <- Class$centers  
## + rownames(ClassCenters) <- paste("center", 1:L, sep = "")  
## + save(ClassCenters, file = "DataWork/ClassCenters.Rdata")  
## + }  
## + if (Flag == 4) {  
## + Num\_Of\_Windowdim <- length(KVec)  
## + Class <- round(runif(Num\_Of\_Windowdim, 1, L))  
## + Classifier <- cbind.data.frame(KVec, Class)  
## + }  
## + save(Classifier, file = "DataWork/Classifier.Rdata")  
## + return(Classifier)  
## + }

#SDB StockDataBuilder function, imports data  
#default basis to close returns, call with (Regular=1) for open 2 close  
SDB()  
  
################# Parameters Monkey #####################  
k <- 10 #how many days to use for k parameter  
l <- 10 #number of different classes used in classification  
KKR <- "Monkey" #clustering method, Monkey - random, K-Means - K-means  
  
################# Train and Backtest ####################  
#matchfunc function load  
source("Code/match.r",echo=TRUE,max.deparse.length = 10000)

##   
## > library(xts)  
##   
## > load("DataWork/StockPrices.Rdata")  
##   
## > retcolnames <- grep("return", colnames(StockPrices))  
##   
## > matchfunc <- function(DDate, k, l, Classifier) {  
## + DDate <- index(first(StockPrices[paste0(DDate, "/")], "1 day"))  
## + DDateindx <- which(index(StockPrices) == DDate)  
## + kRetMat <- StockPrices[(DDateindx - k):(DDateindx - 1), retcolnames]  
## + MinClass <- -1  
## + LowestDist <- -1  
## + for (i in 1:l) {  
## + ClassSegments <- Classifier[Classifier$Class == i, 1]  
## + TotalDist <- -1  
## + for (j in ClassSegments) {  
## + RetMat <- StockPrices[j:(j + k - 1), retcolnames]  
## + DistEuclid <- sum((coredata(RetMat) - coredata(kRetMat))^2)  
## + if (TotalDist < 0) {  
## + TotalDist <- DistEuclid  
## + }  
## + else {  
## + TotalDist <- TotalDist + DistEuclid  
## + }  
## + }  
## + if ((LowestDist == -1) && (TotalDist > 0)) {  
## + LowestDist <- TotalDist  
## + MinClass <- i  
## + }  
## + else {  
## + if ((TotalDist < LowestDist) && (TotalDist > 0)) {  
## + LowestDist <- TotalDist  
## + MinClass <- i  
## + }  
## + }  
## + }  
## + if (MinClass < 0)   
## + stop("Error matchfunc: no available class in data")  
## + return(MinClass)  
## + }

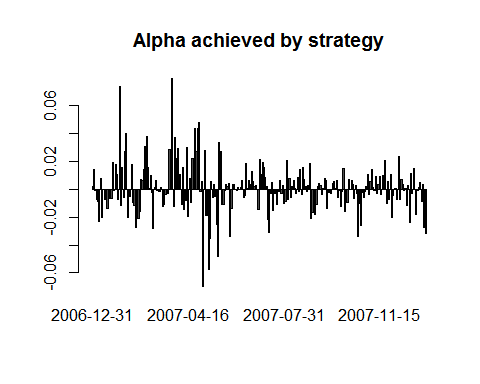
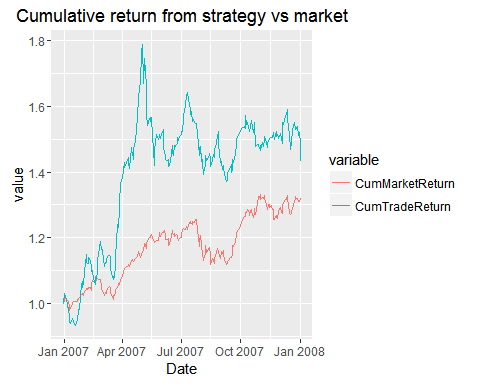
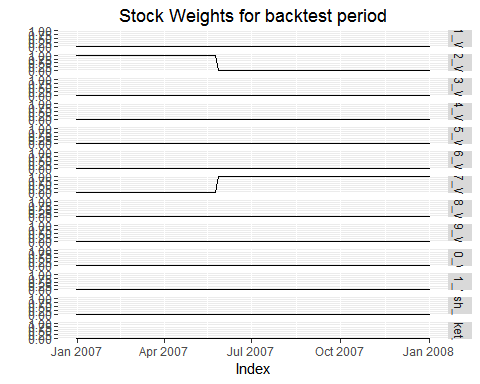
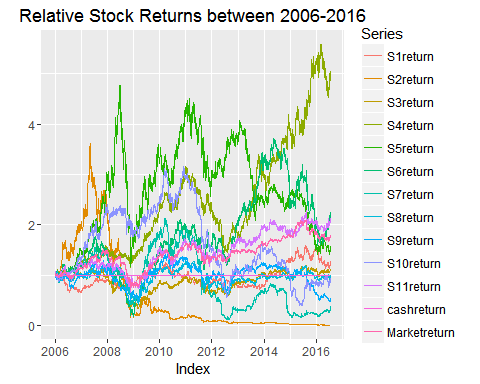
#beststrat function load, uses matchfunc  
source("Code/strategy.r",echo=TRUE,max.deparse.length = 10000)

##   
## > library(xts)  
##   
## > library(Rsolnp)  
##   
## > load("DataWork/StockPrices.Rdata")  
##   
## > fopt <- function(x, r) {  
## + -sum(log(x %\*% t(r)))  
## + }  
##   
## > eqfun <- function(x, r) {  
## + sum(x)  
## + }  
##   
## > beststrat <- function(DDate, k, l, window\_th = 60) {  
## + DDate <- as.Date(DDate)  
## + DDate <- index(first(StockPrices[paste0(DDate, "/")], "1 day"))  
## + DDateindx <- which(index(StockPrices) == DDate)  
## + options(warn = -1)  
## + try(load("DataWork/Classifier.Rdata"), silent = T)  
## + options(warn = 0)  
## + if (!exists("Classifier")) {  
## + Kvec <- K\_Histogram(K = k, DDate = DDate)  
## + Classifier <- Classification(KVec = Kvec, K = k, L = l,   
## + KKR = KKR)  
## + }  
## + if (last(Classifier$KVec) < (DDateindx - window\_th)) {  
## + Kvec <- K\_Histogram(K = k, DDate = DDate)  
## + Classifier <- Classification(KVec = Kvec, K = k, L = l,   
## + KKR = KKR)  
## + }  
## + BestClass <- matchfunc(DDate, k, l, Classifier)  
## + ClassSegments <- Classifier[Classifier$Class == BestClass,   
## + 1]  
## + retcolnames <- grep("return", colnames(StockPrices))  
## + DDateReturns <- StockPrices[ClassSegments + k, retcolnames]  
## + DDateReturns <- DDateReturns/100 + 1  
## + nstocks <- length(retcolnames)  
## + s <- solnp(rep(1/nstocks, nstocks), fopt, eqfun = eqfun,   
## + eqB = 1, LB = rep(0, nstocks), r = DDateReturns, control = list(trace = 0))  
## + xvec <- s$pars  
## + return(xvec)  
## + }

#backtest function load, calls beststrat  
source("Code/trading.r",echo=TRUE,max.deparse.length = 10000)

##   
## > library(xts)  
##   
## > library(ggplot2)  
##   
## > library(reshape2)  
##   
## > load("DataWork/StockPrices.Rdata")  
##   
## > default1ystart <- index(last(first(StockPrices, "1 year"),   
## + "1 day"))  
##   
## > backtest <- function(StartDate = default1ystart, DDate,   
## + k, l) {  
## + retcolnames <- grep("return", colnames(StockPrices))  
## + print(autoplot(cumprod(StockPrices[, retcolnames]/100 + 1),   
## + facets = NULL, main = "Relative Stock Returns between 2006-2016"))  
## + period <- paste0(StartDate, "/", DDate)  
## + test\_prices <- StockPrices[period]  
## + xvec <- t(apply(as.matrix(as.Date(index(test\_prices))), 1,   
## + beststrat, k, l))  
## + colnames(xvec) <- colnames(StockPrices)[retcolnames]  
## + colnames(xvec) <- gsub("return", "\_Wt", colnames(xvec))  
## + BacktestAllocation <- xts(xvec, index(test\_prices))  
## + print(autoplot(BacktestAllocation, main = "Stock Weights for backtest period"))  
## + TradeReturn <- rowSums(xvec \* (test\_prices[, retcolnames]/100 +   
## + 1))  
## + CumTradeReturn <- cumprod(TradeReturn)  
## + SDTradeReturn <- sd(TradeReturn) \* 16  
## + yearsperiod <- as.numeric(index(last(test\_prices)) - index(first(test\_prices)))/360  
## + SharpeTradeReturn <- (last(CumTradeReturn)^(1/yearsperiod) -   
## + 1)/SDTradeReturn  
## + MktReturn <- test\_prices$Marketreturn/100 + 1  
## + CumMktReturn <- cumprod(MktReturn)  
## + plotdata <- data.frame(CumMktReturn, CumTradeReturn, Date = index(test\_prices))  
## + colnames(plotdata) <- c("CumMarketReturn", "CumTradeReturn",   
## + "Date")  
## + plotdata <- melt(plotdata, id = "Date")  
## + g1 <- ggplot(plotdata, aes(x = Date, y = value, colour = variable)) +   
## + geom\_line() + ggtitle("Cumulative return from strategy vs market")  
## + print(g1)  
## + bardat <- xts(TradeReturn - MktReturn, index(test\_prices))  
## + colnames(bardat) <- "Alpha"  
## + barplot(bardat, main = "Alpha achieved by strategy")  
## + print("Comparison of positive/negative returns vs market")  
## + print(table(Market = sign(MktReturn - 1), Strategy = sign(TradeReturn -   
## + 1)))  
## + print(sprintf("Total Profit percentage %.1f%% with volatility of %.1f and Sharpe %.1f",   
## + 100 \* last(CumTradeReturn) - 100, 100 \* SDTradeReturn,   
## + SharpeTradeReturn))  
## + save(BacktestAllocation, file = "DataWork/BacktestAllocation.Rdata")  
## + save(TradeReturn, file = "DataWork/TradeReturn.Rdata")  
## + }

#backtest starting from 1 year after initial date in import up to DDate  
startDate <- index(last(first(StockPrices,"1 year"),"1 day")) #1 year from start of data  
DDate <- as.Date("2008/1/2") #date for prediction, last date for backtest  
backtest(startDate,DDate,k,l) #for period backtesting and graphs/statistics



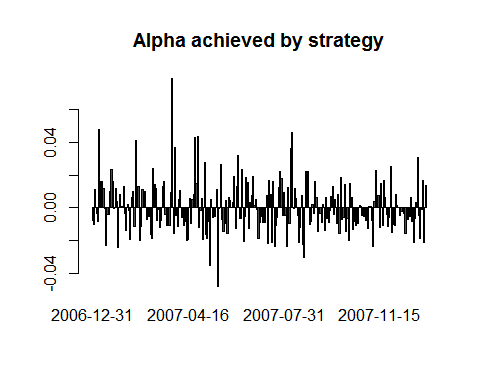
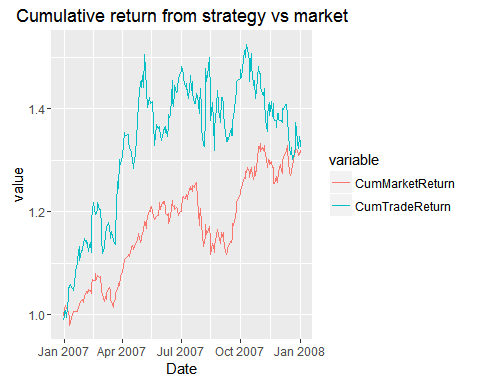
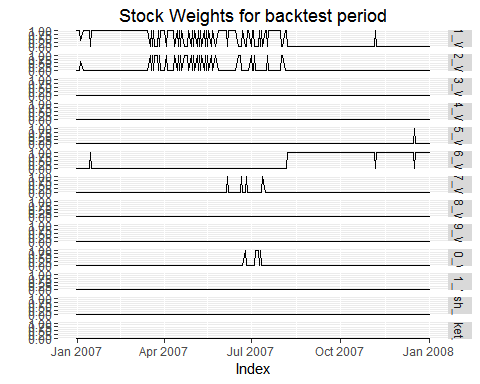
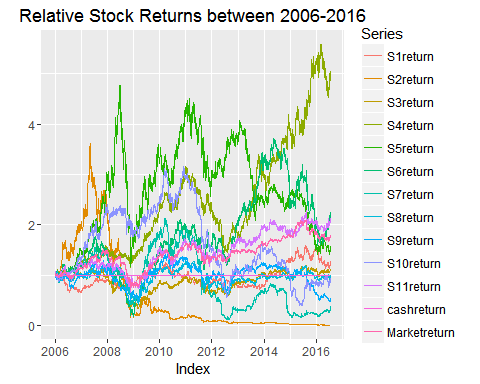
## [1] "Comparison of positive/negative returns vs market"  
## Strategy  
## Market -1 1  
## -1 81 25  
## 1 32 110  
## [1] "Total Profit percentage 43.3% with volatility of 32.7 and Sharpe 1.3"

file.remove("DataWork/Classifier.Rdata")

## [1] TRUE

################# Parameters K-Means ####################  
KKR <- "K-means" #clustering method, Monkey - random, K-Means - K-means  
backtest(startDate,DDate,k,l) #for period backtesting and graphs/statistics

## Warning: did not converge in 10 iterations



## [1] "Comparison of positive/negative returns vs market"  
## Strategy  
## Market -1 1  
## -1 75 31  
## 1 44 98  
## [1] "Total Profit percentage 33.9% with volatility of 29.9 and Sharpe 1.1"

file.remove("DataWork/Classifier.Rdata")

## [1] TRUE

################# Predict next day ######################  
DDate <- as.Date("2009/1/2") #date for prediction  
xvec <- beststrat(DDate,k,l) #generate single prediction for new day  
file.remove("DataWork/Classifier.Rdata")

## [1] TRUE