Adaptive Asset Allocation, Replication

matReturns.com

```r
require(quantmod)
require(PerformanceAnalytics)
require(tseries)

#################################################################################################
########### Replication of Adaptive Asset Allocation AAA by Adam Butler ReSolve Asset Management

### By: Matthew Kiegerl
#
# In this illustration, I will attempt to replicate the
# Adaptive Asset Allocation strategy from ReSolve Asset Management
#
# Step 1: Select from 5 Globally Diversified Asset Classes
# from the list of 10 based on 6 month momentum
#
# Step 2: weight portfolio mean-var optimization every month
# using 6m correlations and 1 month volatilities
#

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

# The Adaptive Asset Allocation algorithm
adaptive_asset_allocation <- function(startDate = '2000-01-01',
                                      endDate = '2023-12-31',
                                      retLookBack = 6,
                                      topAssets = 5) {

  # List of major Global Asset Classes represented by ETF tickers
  tickers <- c("SPY", "VGK", "EWJ", "EEM", "VNQ", "RWX", "IEF", "TLT", "DBC", "GLD")

  # Store Log returns data calculated from adjusted prices to a list
  returns <- list()

  for(i in 1:length(tickers)) {
    rets <- Return.calculate(Ad(get(getSymbols(tickers[i],
                                               from = startDate,
                                               to = endDate,
                                               method = "log"))))
    colnames(rets) <- tickers[i]
    returns[[i]] <- rets
  }

  # Combine rets
  returns <- na.omit(do.call(cbind, returns))
```

```r
# Get monthly endpoints, apply crash prevention
endMonth <- endpoints(returns)
endMonth[endMonth < 1] <- 1
endMonth[endMonth > nrow(returns)] <- nrow(returns)
endMonth <- unique(endMonth)
emDiff <- diff(endMonth)

if (last(emDiff) == 1) { # if the last period only has one observation, remove it
  endMonth <- endMonth[-length(endMonth)]
}

# Initialize vector holding zeroes for assets placement
emptyVector <- data.frame(t(rep(0, length(tickers))))
colnames(emptyVector) <- tickers[1:length(tickers)]

# Create List to store weights
allWeights <- list()

# Start loop for retLookBack filter
for (i in 1:(length(endMonth) - retLookBack)) {

  # Subset for retLookBack of returns - store subsetted returns data
  retSubset <- returns[c((endMonth[i] + 1):endMonth[(i + retLookBack)]), ]
  emSub <- endMonth[i:(i + retLookBack)]
  retsRetLookBack <- returns[(emSub[1] + 1):emSub[(retLookBack + 1)], ]

  # Compute retLookBack cumulative returns as momentum factor
  momentums <- Return.cumulative(retsRetLookBack)

  # Find top assets based on the given topAssets parameter
  highRankAssets <- rank(momentums) >= (length(tickers) - topAssets + 1)
  indx <- which(highRankAssets)

  # Compute correlation and volatility for the selected assets
  correlations <- cor(retsRetLookBack[, indx])
  volatilities <- StdDev(retsRetLookBack[, indx])
  covariances <- t(volatilities) %*% volatilities * correlations

  # Compute standard min vol optimization
  minVolRets <- t(matrix(rep(1, sum(highRankAssets))))
  minVolWts <- portfolio.optim(x = minVolRets, covmat = covariances)$pw
  names(minVolWts) <- colnames(covariances)
  investedAssets <- emptyVector
  investedAssets[, highRankAssets] <- minVolWts

  # Append to list of monthly allocations
  weights <- xts(investedAssets, order.by = last(index(retSubset)))
  allWeights[[i]] <- weights
}

# Combine all weights and returns together
allWeights <- do.call(rbind, allWeights)
investedRets <- returns[, 1:length(tickers)]
```

```
  # Compute AAA portfolio returns
  AAA <- Return.portfolio(R = investedRets, weights = allWeights)
  colnames(AAA) <- "AAA Strategy"

  Benchmark <- Return.portfolio(investedRets, weights = rep((1 / ncol(investedRets)), ncol(investedRets
  colnames(Benchmark) <- "Benchmark-EW"

  AAAdf <- na.omit(cbind(AAA, Benchmark))
  return(AAAdf)
}

# Set parameters
startDate <- '2000-01-01'
endDate <- '2023-12-31'
retLookBack <- 6
topAssets <- 5

# Calculate Adaptive Asset Allocation
AAAdf <- adaptive_asset_allocation(startDate, endDate, retLookBack, topAssets)
aaaReturns <- AAAdf$AAA.Strategy
```

Thanks so much! Check out matReturns.com for more data driven ideas.

Creative Commons License

Note: The code and data provided in this analysis are for illustrative purposes only and do not constitute financial advice. Investors should conduct thorough due diligence before making any investment decisions.