Robust Asset Allocation, Replication

matReturns.com

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

########### Replication of Robust Asset Allocation by Wes Gray ##########################################
###################### By: matReturns.com
###  Replication of the Robust Asset Allocation strategy from Alpha Architect
##  The robust_asset_allocation function takes daily price data for seven ETFs representing different a
##  Converts daily prices to monthly prices and calculates monthly returns.
##  Creates the RAA Balanced portfolio based on monthly return data, with annual rebalancing.
##  Implements Rule 1 (TMOM): Checks if the cumulative returns of RAA - Bonds are greater than 0, and i
##  Calculates the realized returns based on Rule 1.
##  Implements Rule 2 (SMA10): Checks if the cumulative returns of RAA balanced are greater than the Si
##  Calculates the realized returns based on Rule 2.
##  Combines the two strategy returns with 50% allocation each to obtain the RAA Balanced Returns.

robust_asset_allocation <- function(startDate = '2000-01-01',
                                    endDate = '2023-12-31',
                                    retLookBack = 10,
                                    smaLookBack = 10) {
  # 7major Global Asset Classes represented by ETF tickers
  tickers <- c("QMOM", # Alpha Architect U.S. Quantitative Momentum ETF
               "QVAL", # Alpha Architect U.S. Quantitative Value ETF
               "IMOM", # Alpha Architect International Quantitative Momentum ETF
               "IVAL", # Alpha Architect International Quantitative Value ETF
               "VNQ", # Vanguard Real Estate Index Fund ETF
               "DBC", # Invesco DB Commodity Index Tracking Fund
               "IEF" # iShares 7-10 Year Treasury Bond ETF
  )

  # Get daily Adjusted Prices
  prices <- list()
  for(i in 1:length(tickers)) {

    ticker <- Ad(get(getSymbols(tickers[i],
                                from = startDate,
                                to = endDate)))
    colnames(ticker) <- tickers[i]
    prices[[i]] <- ticker

  }

  # Bind price columns, make monthly price and calc monthly rets
  prices <- na.omit(do.call(cbind, prices))
```

```r
# Convert daily prices to monthly prices and calculate monthly returns
monthly.prices <- to.period(prices, period = "months", OHLC = FALSE)
monthly.rets <- na.omit(Return.calculate(monthly.prices))

# Create the framework portfolio (RAA) based on monthly return data, rebalance yearly
RAAbalancedMONTHLY <- Return.portfolio(
  monthly.rets,
  weights = c(10/100, 10/100, 10/100, 10/100, 2/10, 2/10, 1/5),
  rebalance_on = 'months'
)
colnames(RAAbalancedMONTHLY) <- "RAA_Bal_RetsM"

# Rule 1: TMOM, RAA - Bond > 0 = 50% long of portfolio
stratSig <- list()
for (i in 1:(nrow(monthly.rets) - (retLookBack - 1))) {
  # Calculate 10 months of returns
  tenmonthsStrat <- RAAbalancedMONTHLY[c(i:(i + (retLookBack - 1))), ]
  tenmonthsIEF <- monthly.rets$IEF[c(i:(i + (retLookBack - 1))), ]

  # Compute cumulative returns
  stratCum <- Return.cumulative(tenmonthsStrat)
  iefCum <- Return.cumulative(tenmonthsIEF)

  # Excess returns = RAA cumulative returns - IEF cumulative returns
  # If excess returns > 0, go long with 50%
  xRetsSig <- ((stratCum - iefCum) > 0)
  stratSig[[i]] <- xRetsSig
}

# Combine signal(matrix) list into one column
stratSig <- do.call(rbind, stratSig)

# Create a data frame with the signal column and RAA balanced returns (make equal length)
DFxsRets <- cbind(stratSig, RAAbalancedMONTHLY[retLookBack:length(RAAbalancedMONTHLY), ])
colnames(DFxsRets) <- c("stratSig", "RAAbalMrets")

# Multiply the signal row by the next RAA returns row to obtain realized return
RAA_B_xsRets <- list()
for (i in 1:(nrow(DFxsRets) - 1)) {
  returns <- as.numeric(DFxsRets$stratSig[i]) * as.numeric(DFxsRets$RAAbalMrets[i + 1])
  RAA_B_xsRets[[i]] <- returns
}

# Bind rows into a single column of realized returns, len=60, class = list
RAA_B_xsRets <- do.call(rbind, RAA_B_xsRets)

# Convert realized returns into an XTS object
firstpart <- cbind(DFxsRets[2:length(stratSig)], RAA_B_xsRets)

# Rule 2: SMA10 on RAA balanced, if ret > SMA = 50% of Portfolio
cumRetsRAA <- cumsum(RAAbalancedMONTHLY)
SMAcumRetsRAA <- SMA(cumRetsRAA, smaLookBack)
```

```r
  # Create a data frame with RAA returns, cumulative returns, and SMA
  RAA_DF_sma <- na.omit(cbind(RAAbalancedMONTHLY, cumRetsRAA, SMAcumRetsRAA))
  colnames(RAA_DF_sma) <- c("RetsRAA", "CumRetsRAA", "SMA")

  # Signal for SMA strategy: cumulative returns > SMA
  smaSig <- list()
  for (i in 1:nrow(RAA_DF_sma)) {
    momDiff <- (as.numeric(RAA_DF_sma$CumRetsRAA[i]) - as.numeric(RAA_DF_sma$SMA[i])) > 0
    smaSig[[i]] <- momDiff
  }

  # Bind rows
  smaSig <- do.call(rbind, smaSig)

  # Create a data frame with the signal and RAA returns
  RAA_DF_sma <- cbind(RAA_DF_sma, smaSig)

  # Multiply the signal row by the next RAA returns row to obtain realized return
  RAA_B_smaRets <- list()
  for (i in 1:(nrow(RAA_DF_sma) - 1)) {
    returns <- as.numeric(RAA_DF_sma$smaSig[i]) * as.numeric(RAA_DF_sma$RetsRAA[i + 1])
    RAA_B_smaRets[[i]] <- returns
  }

  # Bind rows into a single column of realized returns, len=60, class = list
  RAA_B_smaRets <- do.call(rbind, RAA_B_smaRets)

  # Convert SMA returns into an XTS object
  secondpart <- cbind(RAA_DF_sma[2:length(smaSig)], RAA_B_smaRets)

  # Combine two strategy returns, 50% to each allocation, sum up parts
  TMOMrets <- firstpart$RAA_B_xsRets * 0.5
  SMArets <- secondpart$RAA_B_smaRets * 0.5
  RAA_B_rets <- TMOMrets + SMArets
  colnames(RAA_B_rets) <- "RAA Balanced Returns"

  return(RAA_B_rets)
}

startDate = '2000-01-01'
endDate = '2023-12-31'
retLookBack = 10
smaLookBack = 10

raaReturns <- robust_asset_allocation(startDate, endDate, retLookBack, smaLookBack)
```

Thanks for reading, visit matReturns.com for more.

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.