Bank of Benchmarks

matReturns.com

```r
# Define a function to create a 60/40 benchmark portfolio
sixtyforty_portfolio <- function(startDate = "2018-01-01",
                                 endDate = "2023-12-31") {

  # Load required packages
  require(quantmod)
  require(PerformanceAnalytics)
  # List of symbols
  symbols <- c("SPY", # SPDR S&P 500 ETF Trust
               "TLT") # iShares 20+ Year Treasury Bond ETF
  # Create list called "rets" to store returns
  rets <- list()
  # Fetch and calculate returns for each symbol
  for(i in 1:length(symbols)) {
    returns <- Return.calculate(Ad(get(getSymbols(symbols[i],
                                                  from = startDate,
                                                  to = endDate))),
                                method = "log")
    colnames(returns) <- symbols[i]
    rets[[i]] <- returns
  }
  # Combine returns and remove missing values
  rets <- na.omit(do.call(cbind, rets))
  rets <- round(rets, 5)
  # Define weight variables for the portfolio
  w <- c(0.6, 0.4) # 60% SPY, 40% TLT
  sixtyforty <- cbind(rets$SPY, rets$TLT) # Combine SPY and TLT returns
  # Calculate portfolio returns
  # Rebalanced portfolio on years
  port.rebal.yearly <- Return.portfolio(sixtyforty,
                                        weights = w,
                                        rebalance_on = "years")
  # Rename the column for clarity
  colnames(port.rebal.yearly) <- "60/40"
  # Return the portfolio returns
  return(port.rebal.yearly)
}
# Define the start and end dates
startDate <- "2018-01-01"
endDate <- "2023-12-31"
# Call the function to get portfolio returns
sixtyFortyReturns <- sixtyforty_portfolio(startDate, endDate)
```

```r
# Define a function to create an All Weather Portfolio
allweather_portfolio <- function(startDate = "2018-01-01",
                                 endDate = "2023-12-31") {

  # Load required packages
  require(quantmod)
  require(PerformanceAnalytics)

  # List of symbols
  symbols <- c("VTI", # Vanguard Total Stock Market Index Fund
               "TLT", # iShares 20+ Year Treasury Bond ETF
               "IEI", # iShares 3-7 Year Treasury Bond ETF
               "GLD", # SPDR Gold Shares
               "GSG") # iShares S&P GSCI Commodity-Indexed Trust (GSG)

  # Create list called "rets" to store returns
  rets <- list()

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

  # Combine returns and remove missing values
  rets <- na.omit(do.call(cbind, rets))
  rets <- round(rets, 5)

  # Create weight variable for the portfolio
  w <- c(0.3, 0.4, 0.15, 0.075, 0.075) # 30%, 40%, 15%, 7.5%, 7.5%

  # Calculate portfolio returns
  # Rebalanced portfolio on years
  port.rebal.years <- Return.portfolio(rets, weights = w, rebalance_on = "years")

  # Rename the column for clarity
  colnames(port.rebal.years) <- "All Weather"

  # Return the portfolio returns
  return(port.rebal.years)
}

# Define the start and end dates
startDate <- "2018-01-01"
endDate <- "2023-12-31"

# Call the function to get portfolio returns
allWeatherReturns <- allweather_portfolio(startDate, endDate)
```

```r
# Define a function to create a Golden Butterfly Portfolio
golden_butterfly_portfolio <- function(startDate = "2018-01-01",
                                       endDate = "2023-12-31") {

  # Load required packages
  require(quantmod)
  require(PerformanceAnalytics)

  # List of symbols
  symbols <- c("VTI", # Vanguard Total Stock Market Index Fund
               "IWN", # iShares Russell 2000 Value ETF
               "TLT", # iShares 20+ Year Treasury Bond ETF
               "SHY", # iShares 1-3 Year Treasury Bond ETF
               "GLD") # SPDR Gold Shares

  # Create list called "rets" to store returns
  rets <- list()

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

  # Combine returns and remove missing values
  rets <- na.omit(do.call(cbind, rets))
  rets <- round(rets, 5)

  # Calculate portfolio returns
  # Rebalanced portfolio on years
  port.rebal.years <- Return.portfolio(rets,
                                       weights = c(0.2, 0.2, 0.2, 0.2, 0.2),
                                       rebalance_on = "years")

  # Rename the column for clarity
  colnames(port.rebal.years) <- "Golden Butterfly"

  # Return the portfolio returns
  return(port.rebal.years)
}

# Define the start and end dates
startDate <- "2018-01-01"
endDate <- "2023-12-31"

# Call the function to get portfolio returns
goldenButterflyReturns <- golden_butterfly_portfolio(startDate, endDate)
```

```r
# Define a function to create a Permanent Portfolio
permanent_portfolio <- function(startDate = "2018-01-01",
                                endDate = "2023-12-31") {

  # Load required packages
  require(quantmod)
  require(PerformanceAnalytics)

  # List of symbols
  symbols <- c("SPY", # SPDR S&P 500 ETF Trust
               "TLT", # iShares 20+ Year Treasury Bond ETF
               "SHY", # iShares 1-3 Year Treasury Bond ETF
               "GLD") # SPDR Gold Shares

  # Create list called "rets" to store returns
  rets <- list()

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

  # Combine returns and remove missing values
  rets <- na.omit(do.call(cbind, rets))
  rets <- round(rets, 5)

  # Calculate portfolio returns
  # Rebalanced portfolio on years
  port.rebal.years <- Return.portfolio(rets,
                                       weights = c(0.25, 0.25, 0.25, 0.25),
                                       rebalance_on = "years")

  # Rename the column for clarity
  colnames(port.rebal.years) <- "Permanent"

  # Return the portfolio returns
  return(port.rebal.years)
}

# Define the start and end dates
startDate <- "2018-01-01"
endDate <- "2023-12-31"

# Call the function to get portfolio returns
permanentPortfolioReturns <- permanent_portfolio(startDate, endDate)
```

```r
# Define a function to create an IVY Portfolio
ivy_portfolio <- function(startDate = "2018-01-01",
                          endDate = "2023-12-31") {
  # Load required packages
  require(quantmod)
  require(PerformanceAnalytics)

  # List of symbols
  symbols <- c("VTI", # Vanguard Total Stock Market Index Fund
               "VEU", # Vanguard FTSE All-World ex-US Index Fund
               "VNQ", # Vanguard Real Estate Index Fund
               "BND", # Vanguard Total Bond Market Index Fund
               "GSG") # iShares S&P GSCI Commodity-Indexed Trust (GSG)


  # Create list called "rets" to store returns
  rets <- list()

  # Fetch and calculate returns for each symbol
  for(i in 1:length(symbols)) {
    returns <- Return.calculate(Ad(get(getSymbols(symbols[i],
                                                  from = startDate,
                                                  to = endDate))))
    colnames(returns) <- symbols[i]
    rets[[i]] <- returns
  }

  # Combine returns and remove missing values
  rets <- na.omit(do.call(cbind, rets))
  rets <- round(rets, 5)

  # Calculate portfolio returns
  # Rebalanced portfolio on years
  port.rebal.years <- Return.portfolio(rets,
                                       weights = c(0.20, 0.20, 0.20, 0.20, 0.20),
                                       rebalance_on = "years")

  # Rename the column for clarity
  colnames(port.rebal.years) <- "Ivy"

  # Return the portfolio returns
  return(port.rebal.years)
}

# Define the start and end dates
startDate <- "2018-01-01"
endDate <- "2023-12-31"

# Call the function to get portfolio returns
ivyPortfolioReturns <- ivy_portfolio(startDate, endDate)
```

```r
# Define a function to create the Max Sharpe Portfolio
max_sharpe_portfolio <- function(startDate = "2017-01-01",
                                 endDate = "2023-12-31") {
  # Load required packages
  require(quantmod)
  require(PerformanceAnalytics)
  require(tseries)

  # Define the symbols
  symbols <- c("SPY", # SPDR S&P 500 ETF Trust
               "TLT")  # iShares 20+ Year Treasury Bond ETF

  # Get data for the specified symbols
  rets <- list()
  for(i in 1:length(symbols)) {
    returns <- Return.calculate(Ad(get(getSymbols(symbols[i],
                                                  from = startDate,
                                                  to = endDate))),
                                method = "log")
    colnames(returns) <- symbols[i]
    rets[[i]] <- returns
  }
  rets <- na.omit(do.call(cbind, rets))
  rets <- round(rets, 5)

  # Initialize an empty vector for assets
  emptyVec <- data.frame(t(rep(0, length(symbols))))
  colnames(emptyVec) <- symbols[1:(length(symbols))]

  allWts <- list()
  for(i in 1:(nrow(rets) - 20)) {
    # Get the subset of returns for the last 20 days
    retSubset <- rets[c((i + 1):(i + 20)), ]
    moms <- Return.cumulative(retSubset)

    # Find qualifying assets
    highRankAssets <- rank(moms) >= (length(symbols))
    posReturnAssets <- moms
    selectedAssets <- highRankAssets & posReturnAssets

    # Perform mean-variance/quadratic optimization
    investedAssets <- emptyVec
    if (sum(selectedAssets) == 0) {
      investedAssets <- emptyVec
    } else if (sum(selectedAssets) == 1) {
      investedAssets <- emptyVec + selectedAssets
    } else {
      idx <- which(selectedAssets)
      cors <- cor(retSubset[, idx])
      vols <- StdDev(retSubset[, idx])
      covs <- t(vols) %*% vols * cors

      # Perform min vol optimization
```

```r
      minVolRets <- t(matrix(rep(1, sum(selectedAssets))))
      minVolWt <- portfolio.optim(x = minVolRets, covmat = covs)$pw
      names(minVolWt) <- colnames(covs)
      investedAssets <- emptyVec
      investedAssets[, selectedAssets] <- minVolWt
    }

    # Crash protection
    investedAssets <- investedAssets

    # Append to the list of daily allocations
    wts <- xts(investedAssets, order.by = last(index(retSubset)))
    allWts[[i]] <- wts
  }

  # Combine all weights and compute cash allocation
  allWts <- do.call(rbind, allWts)

  # Add cash returns to the universe of investments
  investedRets <- rets[, 1:(length(symbols))]
  investedRets$CASH <- 0

  # Compute portfolio returns
  out <- Return.portfolio(R = investedRets, weights = allWts)
  colnames(out) <- "Max Sharpe"
  return(out)
}

# Define start and end dates
startDate = "2017-11-30"
endDate = "2023-12-31"

# Calculate and plot Max Sharpe Portfolio returns
msReturns <- max_sharpe_portfolio(startDate, endDate)
```

```r
  # Define a function to create the Equal Risk Portfolio
  equal_risk_portfolio <- function(startDate = "2018-10-01",
                                   endDate = "2023-12-31") {
  require(quantmod)
  require(PerformanceAnalytics)
  require(RiskPortfolios)

  symbols <- c("SPY", # SPDR S&P 500 ETF Trust
               "TLT") # iShares 20+ Year Treasury Bond ETF
  # Get data
  rets <- list()
  for(i in 1:length(symbols)) {
    returns <- Return.calculate(Ad(get(getSymbols(symbols[i],
                                                  from = startDate,
                                                  to = endDate))),
                                method = "log")
    colnames(returns) <- symbols[i]
    rets[[i]] <- returns
  }
  rets <- na.omit(do.call(cbind, rets))
  rets <- round(rets, 5)

  allWts <- list()
  for(i in 1:(nrow(rets) - 60)) {
    # Subset of returns for the last 60 days
    retSubset <- rets[i:(i + 59), ]
    cors <- cor(retSubset)
    vols <- StdDev(retSubset)
    covs <- t(vols) %*% vols * cors

    # Compute equal risk contribution (ERC) portfolio
    rpp <- optimalPortfolio(Sigma = covs, control = list(type = 'erc', constraint = 'lo'))
    allWts[[i]] <- rpp
  }
  # Combine all weights and compute cash allocation
  allWts <- do.call(rbind, allWts)
  erpDF <- cbind(rets[(60 + 1):nrow(rets), ], allWts)
  colnames(erpDF) <- c("spyRets", "tltRets", "spyW", "tltW")
  erpDF$spyW <- lag(erpDF$spyW, n = 1)
  erpDF$tltW <- lag(erpDF$tltW, n = 1)
  erpDF <- na.omit(erpDF)

  # Calculate ERC portfolio returns
  erpReturns <- erpDF$spyRets * erpDF$spyW + erpDF$tltRets * erpDF$tltW
  colnames(erpReturns) <- "Equal Risk"
  return(erpReturns)
}
  startDate = "2017-10-04"
  endDate = "2023-12-31"
  erpReturns <- equal_risk_portfolio(startDate, endDate)
```

```r
# Define a function to create the Max Diversification Portfolio
max_diversification_portfolio <- function(startDate = "2018-10-01",
                                          endDate = "2023-12-31") {
  require(quantmod)
  require(PerformanceAnalytics)
  require(RiskPortfolios)

  symbols <- c("SPY", # SPDR S&P 500 ETF Trust
               "TLT") # iShares 20+ Year Treasury Bond ETF
  # Get data
  rets <- list()
  for(i in 1:length(symbols)) {
    returns <- Return.calculate(Ad(get(getSymbols(symbols[i],
                                                  from = startDate,
                                                  to = endDate))),
                                method = "log")
    colnames(returns) <- symbols[i]
    rets[[i]] <- returns
  }
  rets <- na.omit(do.call(cbind, rets))
  rets <- round(rets, 5)

  allWts <- list()
  for(i in 1:(nrow(rets) - 60)) {
    # Subset of returns for the last 60 days
    retSubset <- rets[i:(i + 59), ]
    cors <- cor(retSubset)
    vols <- StdDev(retSubset)
    covs <- t(vols) %*% vols * cors
    # Maximum diversification portfolio with the long-only constraint
    mdWts <- optimalPortfolio(Sigma = covs, control = list(type = 'maxdiv', constraint = 'lo'))
    allWts[[i]] <- mdWts
  }
  # Combine all weights and compute cash allocation
  allWts <- do.call(rbind, allWts)
  mdDF <- cbind(rets[(60 + 1):nrow(rets), ], allWts)
  colnames(mdDF) <- c("spyRets", "tltRets", "spyW", "tltW")
  mdDF$spyW <- lag(mdDF$spyW, n = 1)
  mdDF$tltW <- lag(mdDF$tltW, n = 1)
  mdDF <- na.omit(mdDF)
  # Calculate Max Diversification portfolio returns
  mdReturns <- mdDF$spyRets * mdDF$spyW + mdDF$tltRets * mdDF$tltW
  colnames(mdReturns) <- "Max Diversification"
  return(mdReturns)
}
startDate = "2017-10-04"
endDate = "2023-12-31"
mdReturns <- max_diversification_portfolio(startDate, endDate)
```

```r
# Define a function to calculate Risk Premium Value Weighted Portfolio returns
risk_premium_value_weighted <- function(startDate = "2017-12-29",
                                        endDate = "2023-12-31") {
  require(quantmod)
  require(PerformanceAnalytics)

  symbols <- c("TLT", # iShares 20+ Year Treasury Bond ETF
               "LQD", # iShares iBoxx $ Investment Grade Corporate Bond ETF
               "SPY", # SPDR S&P 500 ETF Trust
               "SHY") # iShares 1-3 Year Treasury Bond ETF

  # Get data
  rets <- list()
  for (i in 1:length(symbols)) {
    returns <- Return.calculate(Ad(get(getSymbols(symbols[i], from = startDate, to = endDate))), method
    colnames(returns) <- symbols[i]
    rets[[i]] <- returns
  }
  rets <- na.omit(do.call(cbind, rets))
  rets <- round(rets, 5)

  # Calculate yield differences
  tlt_yield_diff <- rets$TLT - rets$SHY
  lqd_yield_diff <- rets$LQD - rets$TLT
  spy_yield_diff <- rets$SPY - rets$TLT

  tmpTLT <- list()
  tmpLQD <- list()
  tmpSPY <- list()
  # calc the normalized risk premium
  for (i in 1:nrow(tlt_yield_diff)) {
    checkTLT <- Return.cumulative(tlt_yield_diff[1:i, ]) > na.omit(SMA(tlt_yield_diff, n = i))[1, ]
    tmpTLT[[i]] <- checkTLT
    checkLQD <- Return.cumulative(lqd_yield_diff[1:i, ]) > na.omit(SMA(lqd_yield_diff, n = i))[1, ]
    tmpLQD[[i]] <- checkLQD
    checkSPY <- Return.cumulative(spy_yield_diff[1:i, ]) > na.omit(SMA(spy_yield_diff, n = i))[1, ]
    tmpSPY[[i]] <- checkSPY
  }
  tmpTLT <- do.call(rbind, tmpTLT)
  tmpLQD <- do.call(rbind, tmpLQD)
  tmpSPY <- do.call(rbind, tmpSPY)

  tltDF <- cbind(rets$TLT, tmpTLT)
  colnames(tltDF) <- c('TLT RP', "TLT sig")
  tltDF$`TLT sig` <- lag(tltDF$`TLT sig`, 1)
  tltDF <- na.omit(tltDF)
  tltRPreturns <- tltDF$`TLT RP` * tltDF$`TLT sig`

  lqdDF <- cbind(rets$LQD, tmpLQD)
  colnames(lqdDF) <- c('LQD RP', "LQD sig")
  lqdDF$`LQD sig` <- lag(lqdDF$`LQD sig`, 1)
  lqdDF <- na.omit(lqdDF)
  lqdRPreturns <- lqdDF$`LQD RP` * lqdDF$`LQD sig`
```

```r
  spyDF <- cbind(rets$SPY, tmpSPY)
  colnames(spyDF) <- c('SPY RP', "SPY sig")
  spyDF$`SPY sig` <- lag(spyDF$`SPY sig`, 1)
  spyDF <- na.omit(spyDF)
  spyRPreturns <- spyDF$`SPY RP` * spyDF$`SPY sig`
  rpReturnsDF <- cbind(tltRPreturns, lqdRPreturns, spyRPreturns)

  rpvReturns <- list()
  for (i in 1:(nrow(tmpTLT) - 1)) {
    idx <- which(rpReturnsDF[i, ] != 0)
    return_val <- sum(rpReturnsDF[i, ]) / length(idx)
    rpvReturns[[i]] <- return_val
  }
  rpvReturns <- do.call(rbind, rpvReturns)
  colnames(rpvReturns) <- "RPV Returns"
  rpReturnsDF <- cbind(rpReturnsDF, rpvReturns)

  return(rpReturnsDF$RPV.Returns)
}

# Define start and end dates
startDate <- "2017-12-29"
endDate <- "2023-12-31"

# Calculate Risk Premium Value Weighted Portfolio returns
rpvwReturns <- risk_premium_value_weighted(startDate, endDate)
```

```r
# Define a function to calculate Risk Premium Value Best Value Portfolio returns
risk_premium_value_bestValue <- function(startDate = "2017-12-29",
                                         endDate = "2023-12-31") {
  require(quantmod)
  require(PerformanceAnalytics)

  symbols <- c("TLT", # iShares 20+ Year Treasury Bond ETF
               "LQD", # iShares iBoxx $ Investment Grade Corporate Bond ETF
               "SPY", # SPDR S&P 500 ETF Trust
               "SHY") # iShares 1-3 Year Treasury Bond ETF

  # Get data
  rets <- list()
  for (i in 1:length(symbols)) {
    returns <- Return.calculate(Ad(get(getSymbols(symbols[i],
                                                  from = startDate,
                                                  to = endDate))),
                                method = "log")
    colnames(returns) <- symbols[i]
    rets[[i]] <- returns
  }
  rets <- na.omit(do.call(cbind, rets))
  rets <- round(rets, 5)

  # Calculate yield differences
  tlt_yield_diff <- rets$TLT - rets$SHY
  lqd_yield_diff <- rets$LQD - rets$TLT
  spy_yield_diff <- rets$SPY - rets$TLT

  tmpTLT <- list()
  tmpLQD <- list()
  tmpSPY <- list()
  for (i in 1:nrow(tlt_yield_diff)) {
    # Calculate cumulative yield differences
    checkTLT <- Return.cumulative(tlt_yield_diff[1:i, ]) -
      na.omit(SMA(tlt_yield_diff, n = i))[1, ]
    tmpTLT[[i]] <- checkTLT

    checkLQD <- Return.cumulative(lqd_yield_diff[1:i, ]) -
      na.omit(SMA(lqd_yield_diff, n = i))[1, ]
    tmpLQD[[i]] <- checkLQD

    checkSPY <- Return.cumulative(spy_yield_diff[1:i, ]) -
      na.omit(SMA(spy_yield_diff, n = i))[1, ]
    tmpSPY[[i]] <- checkSPY
  }
  tmpTLT <- do.call(rbind, tmpTLT)
  tmpLQD <- do.call(rbind, tmpLQD)
  tmpSPY <- do.call(rbind, tmpSPY)
  bestValueDF <- cbind(tmpTLT, tmpLQD, tmpSPY)

  maxRP <- list()
  for (i in 1:nrow(tmpTLT)) {
```

```r
    # Identify index of maximum value for each row
    max <- which.max(bestValueDF[i, ])
    maxRP[[i]] <- max
  }
  maxRP <- do.call(rbind, maxRP)
  colnames(maxRP) <- "max idx"
  bvDFsig <- cbind(bestValueDF, maxRP)

  sig <- list()
  for (i in 1:nrow(tmpTLT)) {
    # Create a binary signal based on max index values
    tmp <- as.numeric(bvDFsig[i, bvDFsig$max.idx[i, ]] > 0)
    sig[[i]] <- tmp
  }
  sig <- do.call(rbind, sig)
  bvDFsig <- cbind(bvDFsig, sig)
  idx <- bvDFsig$max.idx * bvDFsig$sig
  colnames(idx) <- "Index"
  bvDFrets <- cbind(rets, idx)
  bvDFrets$Index <- lag(bvDFrets$Index, n = 1)
  bvDFrets <- na.omit(bvDFrets)

  stratRets <- list()
  for (i in 1:(nrow(tmpTLT) - 1)) {
    if (bvDFrets$Index[i, ] == 0) {
      stratRets[[i]] <- as.numeric(0)
    } else {
      stratRets[[i]] <- as.numeric(bvDFrets[i, bvDFrets$Index[i, ]])
    }
  }
  rpvbReturns <- do.call(rbind, stratRets)
  colnames(rpvbReturns) <- "rpvbReturns"
  rpReturnsDF <- cbind(bvDFrets, rpvbReturns)

  return(rpReturnsDF$rpvbReturns)
}

# Define start and end dates
startDate <- "2017-12-29"
endDate <- "2023-12-31"

# Calculate Risk Premium Value Best Value Portfolio returns
rpbvReturns <- risk_premium_value_bestValue(startDate, endDate)
```

Thanks for reading. Check out matReturns.com for more quality content!!

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.