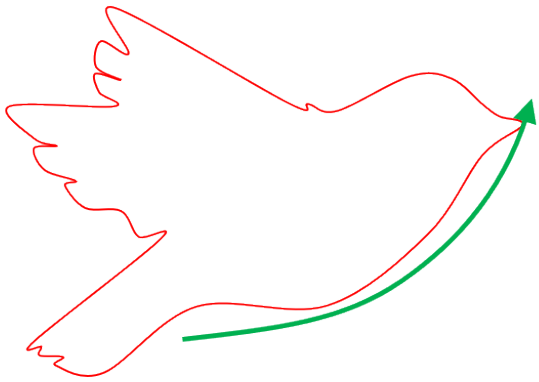
TORI 於永明彩虹強積金計劃的回測 – Backtesting TORI in SunLife Rainbow Scheme

jchan-gi (Jimmy Chan)

April 2nd, 2020



TORI

Trend- & Oversell-based Rebalance Instructions

TORI (Trend- & Oversell-based Rebalancing Instructions) in SunLife Rainbow MPF Scheme

永明彩虹強積金應用 TORI 進行資產調配

Civil Servant Management fee pricing

公務員收費

Earliest data available from 2001/09/17

數據由 2001/09/17 開始

Disclaimer

1. The author (jchan-gi) expressly stated that nothing my repositories and webpages constitutes any advices or recommendation on investing or allocating assets on any stock or financial products.

2. No content in my study have been prepared with respect to specific personal need and investment objects of individuals. All materials reveal to the public are applicable to the author (jchan-gi) only.

3. All results from this repository are simulated from historical data only. It may portray inaccurate information and hence the repository must not applied in production and must used as academic exchange only. The author is not responsible in any loss induced from using this repository.

4. All investors shall obtain kind advices from professional financial consultants before making any decisions. Professional financial consultants should recommend products or decisions that suit your needs and objectives.

5. The author is not licensed nor professional in the field hence this studies are not professional advice and may not be suitable for anyone except myself.

6. You shall not invest on any financial products using the information on this code repository, website or any information made public by jchan-gi alone. You agree you have fully understand and willing to undertake the risks of this script in any circumstances involves in using this script.

7. English version shall be used if any discrepencies exists between English and Chinese version.

免責聲明

1. 本文作者現此聲明，本人的程式及網頁並非對閣下提供或作出對任何金融產品的投資或資產調配的建議。

2. 本文內容沒有針對閣下或其他人的實際個人需要而編撰。所有公開的內容只適用於本文作者。

3. 本文內容根據歷史數據模擬回報率等一切資訊。該等資訊可能非常不準確，本文及程式庫任何內容不應於實際環境使用，及只供學術討論。如因使用本程式庫任何內容而招致損失，作者一概不會負責。

4. 所有投資者應在作出任何決定前，先向專業理財顧問查詢及要求提供意見。只有理財顧問的意見才能符合閣下的實際理財需要及風險胃納。

5. 本文作者非專業人士或持牌從業者，因此，本文內容並非專業意見。而且，本文內容極有可能並不合適於除作者以外的任何人。

6. 在作出任何投資決定前，你不應單靠此程序作出決定。當你作出任何與本程序算法有關的投資決定時，你已完全知悉並願意接受本程序或算法所帶來的風險。

7. 本聲明如有歧義，將以英文版本作準。

最後更新日期 Last update: 2020/04/02.

Changelog

更新記錄

2020-04-02 v2.1.3a
Release results accounted up to 2020/03/31.
公佈直至 2020/03/31 的結果。

2020-03-31 v2.1.2 Version synchronize before uploading 2020Q1 results.
將所有程式版本同步，以準備發佈 2020Q1 結果。

2020-03-27 v2.1.1a
1. Fix return matrix column misalignment in Tori @ SunLife.

2. Fix faulty code on inconsistent total asset price across time. Description: Subsetting would not work to produce consistent total asset value when the data is splited by train-test partition. This is due to the hedge flag detect the future signal if the subsetting is being applied to the weight but not the return. This is a characteristics of Return.portfolio but not a glitch of the algorithm.
1. 修正回報欄位錯置。
2. 修正於不同時間下出現不一致的總資產。

2020-03-25 v2.1.1
1. Adding Golden Butterfly Portfolio as a benchmark portfolio.
1. 增加 Golden Butterfly Portfolio 作為基準指標。

2020-03-24 v2.1
1. Further diversify the hedge mode with Sixty-Fifth Plus Fund.
2. Performance indicator calculation on first installment revised extensively, includes:
* Adding SPY (SPDR S&P 500 ETF), QQQ (Invesco QQQ Trust), and HSI (Hang Seng Index) as a benchmark.
* Captial Asset Pricing Model CAPM and Information Ration results are available. (The author disapprove the use of CAPM, the results of CAPM are for reference only)
* Charting with benchmark are accomplished through tidyquant and ggplot now.
1. 使用 65 歲後基金進一步分散避險狀態下的風險。
2. 計算第一期投資的回報已作出重大修改，包括：
加入 SPY,QQQ 及恒生指數作基準
以三項基準計算資本資產定價模型 CAPM 及資訊比率 Information Ratio 的比較。(作者不認同 CAPM 是有效的評估方法。數值僅供參考。)
圖表改以 tidyquant + ggplot 展示，並加入與基準的比較。

2020-03-18 v2
1. Separate the return used in calculating AKANE, TORI (log) and return (simple).
1. 修正計算回報時使用的方式

2020-01-17 v1.1
1. Minor enhancement on background information.
1. 更改背景資料

2020-01-13 v1
1. TORI is officially published and backtested with SunLife Rainbow MPF
1. 正式推出以永明彩虹強積金作回測的 TORI 介紹

WARNING: Please understand nature of TORI in this section before reading further:

This script relies on BIAS indicator and RSI indicator for investment rebalancing.
Please understand BIAS indicator and RSI indicator before reading further.
The script perform walk-forward validation only.
A model will be created for exactly next re-balance only based on most updated data for that moment.
There is no other validation or assessment on the data.
Please be reminded that backtesting and/or walk-forward validation is no way to be accurate in predicting future performance.
Using this script may exposed to various risk, including but not limited to overfitting, market, timing, etc.

警告：在應用 TORI 之前，請先於此節了解 TORI 的特性

TORI 根據乖離值及相對強弱指數作投資組合再配置。
閱讀下文前，請先了解乖離值及相對強弱指數。
此模式只使用了前移式回測 (walk-forward validation)。
每次資產調配都會基於擁有當時而言的最新資料。
然而，請切記過往表現不代表將來表現。
使用此代碼將會受到包括但不限於以下的風險所影響：過擬合風險、市場風險、時差風險等。

Background Introduction

Mandatory Provident Fund (MPF) is one of the legal pensions in Hongkong.
All Employers and some employees are required to pay at least 5% of monthly salary to MPF services provider.
MPF providers are required to provide approved funds for employees to invest, while the team could earn management fees.

However, the average annualized return of MPF is 3.1% only in from 2000-2016 (Mandatory Provident Fund Schemes Authority, 2016).
Most Hongkong employees feels they are forced to give their salary to MPF fund manager.
Indeed, the reasons of low return may due to inactive management by employees.
In this example, we will explore n-m MA BIAS to rise the annualized return in TORI.

背景簡介

在香港，強積性公積金 (強積金, Mandatory Provident Fund, MPF) 是其中一種法定退休金。
僱主及部份僱員須扣除 5% 薪金供強積金營運商。
強積金營運商將每月收取管理費，並提供認可基金供僱員作投資儲蓄。
但是，強積金的平均回報僅 3.1%，因而被批評回報有三份之一被基金經理蠶食。
誠言，每位僱員如以主動方式管理強積金，有助提升回報而減少收費的影響。
這文記錄了作者以時差平均線乖離率 (n-m MA BIAS) 提升強積金回報的探索 (即 TORI)。

TORI used NMMA BIAS & Relative Strength Index (RSI)

The author used n-m MA BIAS indicator (NMMA BIAS) to extract trend and RSI for overbuy/oversell status in TORI.
This script combine two indicators and consider recent stock returns for the three outcome:
1. Tactical Asset Allocation (TAA) using TORI 2. Strategic Asset Allocation (SAA) using TORI and designated funds 3. Hedging mode (cash mode)

TORI 應用了時差平均線乖離率及相對強

作者嘗試以時差平均線乖離率 (n-m MA BIAS, NMMA BIAS) 及相對強弱指數 (RSI) 分配投資物。
時差平均線乖離率的預測考慮了投資物的趨勢 (Trend)。　相對強弱指數則考慮投資物的超買超賣狀態。
本程序結合兩項指標後，再考慮考慮最近三至六個月回報後，程式將自動決定以 TORI 佈置戰術性資產部署 (tactical asset allocation, TAA) 、或是以 TORI 結果配合債券及現金的策略性資產部署 (Strategic asset allocation)、或進入避險模式。

When to manage my portfolio using TORI?

The author collect historical pricing in the last day of every month to calculate NMMA BIAS and RSI in TORI.
The same day would be the day of successful rebalance.
In this example, MPF would have T+2 time delay for obtaining historical price, hence it is expected to have timing risks in TORI in MPF.

何時使用 TORI 管理及調配投資組合?

作者以每月最後一日取得投資物的歷史價格計算每月回報、NMMA BIAS 及相對強弱指數。
然後於同日重新配置投資物。
在本示例中，強積金大部分均有 T+2 的時延，因此實際操作將有時延誤差。

TORI in action: Walk-forwark validation results

TORI 前移式回測結果

單一供款 One-off MPF Installment:

指標 Indicator	Tori @ SunLife Rainbow GS Pricing	SPY	HSI	Golden Butterfly
年率化回報 Annualized Return	10.70%	7.39%	4.84%	7.88%
累積回報 Cumulative Return	556.08%	/	/	/
年率化標準誤差 Annualized SD	9.00%	14.13%	19.98%	7.26%
夏普比率 Sharpe Ratio	0.8895	0.5228	0.2421	1.0856
索丁諾比率 Sortino Ratio	2.9337	/	/	/
資訊比率 Information Ratio	–>	0.2041	0.2785	0.2275
Annualised Jensen's Alpha	–>	0.1074	0.1079	0.1040
Beta	–>	0.0428	0.0460	-0.0925
關聯系數 Correlation Coeff.	–>	0.0673	0.1021	0.0477
預期損失 Expected Shortfall:	5.33% loss	/	/	/

月供投資 Monthly MPF installment (或有誤差 maybe slightly differs):
每月供款 Monthly installment amount: 2301
總供款 Total contribution: 513123
最新結餘 Latest asset value: 1337808
算術年均回報 Mean annual return: 8.39%
內部回報率 Internal Rate of Return (IRR): 9.03%
年率化標準誤差 Annualized Standard Deviation: 8.99%
夏普比率 Sharpe Ratio: 0.9337
索丁諾比率 Sortino Ratio: 2.6600 (MAR = 0%)

流程 Detailed Workflow

Package Preparation

1. Install necessary packages

```
r = getOption("repos")
r["CRAN"] = "https://cran.r-project.org/"
options(repos = r)


install.packages("zoo", type="binary")
install.packages("xts", type="binary")
install.packages("fBasics", type="binary")
install.packages("quantmod", dependencies = TRUE, type="binary")
install.packages("PerformanceAnalytics", dependencies = TRUE, type="binary")
install.packages("tidyverse", dependencies = TRUE, type="binary")
install.packages("tibbletime", dependencies = TRUE, type="binary")
install.packages("tidyquant", dependencies = TRUE, type="binary")
install.packages("reshape", type="binary")
```

```
library(knitr)
hook_output = knit_hooks$get('output')
knit_hooks$set(output = function(x, options) {
  # this hook is used only when the linewidth option is not NULL
  if (!is.null(n <- options$linewidth)) {
```

```
    x = knitr:::split_lines(x)
    # any lines wider than n should be wrapped
    if (any(nchar(x) > n)) x = strwrap(x, width = n)
    x = paste(x, collapse = '\n')
  }
  hook_output(x, options)
})
```

2. Now load necessary packages.

```
library("zoo")
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library("xts")

library("fBasics")
```

```
## Loading required package: timeDate
```

```
## Loading required package: timeSeries
```

```
##
## Attaching package: 'timeSeries'
```

```
## The following object is masked from 'package:zoo':
##
##     time<-
```

```
library("tidyverse")
```

```
## -- Attaching packages ---------------------------------------------------
------------------------------------ tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------
------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks timeSeries::filter(), stats::filter()
## x dplyr::first()  masks xts::first()
## x dplyr::lag()    masks timeSeries::lag(), stats::lag()
## x dplyr::last()   masks xts::last()
```

```
library("timetk")
```

```
## Loading required package: recipes
```

```
##
## Attaching package: 'recipes'
```

```
## The following object is masked from 'package:stringr':
##
```

```
##     fixed
## The following object is masked from 'package:stats':
##
##     step
```

```
library("tibbletime")
```

```
##
## Attaching package: 'tibbletime'
## The following object is masked from 'package:timeSeries':
##
##     filter
## The following object is masked from 'package:stats':
##
##     filter
```

```
library("reshape")
```

```
##
## Attaching package: 'reshape'
## The following object is masked from 'package:dplyr':
##
##     rename
## The following objects are masked from 'package:tidyr':
##
##     expand, smiths
```

```
library("tidyquant")
```

```
## Loading required package: lubridate
##
## Attaching package: 'lubridate'
## The following object is masked from 'package:reshape':
##
##     stamp
## The following object is masked from 'package:base':
##
##     date
## Loading required package: PerformanceAnalytics
##
## Attaching package: 'PerformanceAnalytics'
## The following objects are masked from 'package:timeDate':
##
##     kurtosis, skewness
## The following object is masked from 'package:graphics':
##
##     legend
## Loading required package: quantmod
## Loading required package: TTR
```

```
##
## Attaching package: 'TTR'

## The following object is masked from 'package:fBasics':
##
##     volatility

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Version 0.4-0 included new data defaults. See ?getSymbols.

## == Need to Learn tidyquant? ====================================================
## Business Science offers a 1-hour course - Learning Lab #9: Performance Analysis & Portfolio O
## </> Learn more at: https://university.business-science.io/p/learning-labs-
pro </>
```

```
library("quantmod")
library("PerformanceAnalytics")
```

Load Prices and Calculate Return

0. Parameters

```
# Determine the n-m MA BIAS percentage
nmBIAS_Percent <- 1

top <- 2
RSI_Overbuy <- 0.85
nm <- 1
MA_long <- 6
MA_Short <- 2
RSI_Period <- 18
Min_NMMA <- 0.00002
```

```
## [1] "LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MONETAR
```

1. Load the price into zoo format

```
MPF.SunLife <-
  as.xts(
    read.zoo(
      "F:/OneDrive/MPF/Sun Life Rainbow/SunLife.csv",
      format = "%d/%m/%Y",
      header = TRUE,
      read = read.csv,
      na.strings = "0"
    )
  )
daily <- index(MPF.SunLife)
```

```
## [1] "LC_COLLATE=Chinese (Traditional)_Hong Kong SAR.950;LC_CTYPE=Chinese (Traditional)_Hon
```

2. Calculate Simple Moving Average (SMA)

```
MPF.SunLife.SMA.hy <- na.fill(apply(MPF.SunLife, 2, function(x) SMA(x, n=21*MA_long)), 0)
MPF.SunLife.SMA.hy <- as.xts(MPF.SunLife.SMA.hy, order.by = daily)
```

```
MPF.SunLife.SMA.mt <- na.fill(apply(MPF.SunLife, 2, function(x) SMA(x, n=21*MA_Short)), 0)
MPF.SunLife.SMA.mt <- as.xts(MPF.SunLife.SMA.mt, order.by = daily)
```

3. Calculate Relative Strength Index (RSI)

```
MPF.SunLife.RSI <- na.fill(apply(MPF.SunLife, 2, function(x) RSI(x, n=21*RSI_Period)),0)
MPF.SunLife.RSI <- as.xts(MPF.SunLife.RSI, order.by = daily)
```

4. Calculate Returns

```
MPF.SunLife.AE <- monthlyReturn(as.xts(MPF.SunLife$AE), type="log")
MPF.SunLife.B <- monthlyReturn(as.xts(MPF.SunLife$B), type="log")
MPF.SunLife.CA <- monthlyReturn(as.xts(MPF.SunLife$CA), type="log")
MPF.SunLife.CE <- monthlyReturn(as.xts(MPF.SunLife$CE), type="log")
MPF.SunLife.HKT <- monthlyReturn(as.xts(MPF.SunLife$FTSE.HK), type="log")
MPF.SunLife.G <- monthlyReturn(as.xts(MPF.SunLife$G), type="log")
MPF.SunLife.GB <- monthlyReturn(as.xts(MPF.SunLife$GB), type="log")
MPF.SunLife.GE <- monthlyReturn(as.xts(MPF.SunLife$GE), type="log")
MPF.SunLife.HKB <- monthlyReturn(as.xts(MPF.SunLife$HKB), type="log")
MPF.SunLife.HKE <- monthlyReturn(as.xts(MPF.SunLife$HKE), type="log")
MPF.SunLife.MPFC <- monthlyReturn(as.xts(MPF.SunLife$MPFC), type="log")
MPF.SunLife.SFP <- monthlyReturn(as.xts(MPF.SunLife$SFP), type="log")

MPF.SunLife.returns <-
  merge(
    MPF.SunLife.AE,
    MPF.SunLife.B,
    MPF.SunLife.CA,
    MPF.SunLife.CE,
    MPF.SunLife.HKT,
    MPF.SunLife.G,
    MPF.SunLife.GB,
    MPF.SunLife.GE,
    MPF.SunLife.HKB,
    MPF.SunLife.HKE,
    MPF.SunLife.MPFC,
    MPF.SunLife.SFP
  )

MPF.SunLife.AE <- monthlyReturn(as.xts(MPF.SunLife$AE), type="arithmetic")
MPF.SunLife.B <- monthlyReturn(as.xts(MPF.SunLife$B), type="arithmetic")
MPF.SunLife.CA <- monthlyReturn(as.xts(MPF.SunLife$CA), type="arithmetic")
MPF.SunLife.CE <- monthlyReturn(as.xts(MPF.SunLife$CE), type="arithmetic")
MPF.SunLife.HKT <- monthlyReturn(as.xts(MPF.SunLife$FTSE.HK), type="arithmetic")
MPF.SunLife.G <- monthlyReturn(as.xts(MPF.SunLife$G), type="arithmetic")
MPF.SunLife.GB <- monthlyReturn(as.xts(MPF.SunLife$GB), type="arithmetic")
MPF.SunLife.GE <- monthlyReturn(as.xts(MPF.SunLife$GE), type="arithmetic")
MPF.SunLife.HKB <- monthlyReturn(as.xts(MPF.SunLife$HKB), type="arithmetic")
MPF.SunLife.HKE <- monthlyReturn(as.xts(MPF.SunLife$HKE), type="arithmetic")
MPF.SunLife.MPFC <- monthlyReturn(as.xts(MPF.SunLife$MPFC), type="arithmetic")
MPF.SunLife.SFP <- monthlyReturn(as.xts(MPF.SunLife$SFP), type="arithmetic")

MPF.SunLife.simple.returns <-
  merge(
    MPF.SunLife.AE,
```

```
    MPF.SunLife.B,
    MPF.SunLife.CA,
    MPF.SunLife.CE,
    MPF.SunLife.HKT,
    MPF.SunLife.G,
    MPF.SunLife.GB,
    MPF.SunLife.GE,
    MPF.SunLife.HKB,
    MPF.SunLife.HKE,
    MPF.SunLife.MPFC,
    MPF.SunLife.SFP
  )

MPF.SunLife.original.cost <-
  c(
    0.0201,
    0.0187,
    0.0085,
    0.0206,
    0.0102,
    0.0183,
    0.0202,
    0.0204,
    0.0180,
    0.0177,
    0.0096,
    0.0085
  ) / 12
MPF.SunLife.cs.cost <-
  c(
    0.0096,
    0.0089,
    0.0095,
    0.0096,
    0.0096,
    0.0089,
    0.0089,
    0.0096,
    0.0089,
    0.0089,
    0.0004,
    0.0095
  ) / 12

for (col in 1:length(MPF.SunLife.returns[1, ])) {
  MPF.SunLife.returns[, col] <-
    MPF.SunLife.returns[, col] - MPF.SunLife.cs.cost[col] + MPF.SunLife.original.cost[col]
  MPF.SunLife.simple.returns[, col] <-
    MPF.SunLife.simple.returns[, col] - MPF.SunLife.cs.cost[col] + MPF.SunLife.original.cos
}

monthly <- index(MPF.SunLife.returns)
colnames(MPF.SunLife.returns) <-
```

```
  c("AE",
    "B",
    "CA",
    "CE",
    "FTSE HK",
    "G",
    "GB",
    "GE",
    "HKB",
    "HKE",
    "MPFC",
    "SFP")
colnames(MPF.SunLife.simple.returns) <-
  c("AE",
    "B",
    "CA",
    "CE",
    "FTSE HK",
    "G",
    "GB",
    "GE",
    "HKB",
    "HKE",
    "MPFC",
    "SFP")


rm(
  MPF.SunLife.AE,
  MPF.SunLife.B,
  MPF.SunLife.CA,
  MPF.SunLife.CE,
  MPF.SunLife.HKT,
  MPF.SunLife.G,
  MPF.SunLife.GB,
  MPF.SunLife.GE,
  MPF.SunLife.HKB,
  MPF.SunLife.HKE,
  MPF.SunLife.MPFC,
  MPF.SunLife.SFP
)
```

Calculate average RSI of the month, and then adjustment factor

```
MPF.SunLife.RSI.month <-
  as.xts(do.call(rbind, lapply(split(as.xts(MPF.SunLife.RSI), "months"), function(x)
    colAvgs(x))), order.by = monthly)

MPF.SunLife.RSI.p <- MPF.SunLife.returns
MPF.SunLife.RSI.p[, ] <- 0

for (col in 1:length(MPF.SunLife.RSI.month[1, ])) {
  if (col != 11) {
```

```
    for (row in 1:length(MPF.SunLife.RSI.month[, col])) {
      percentile <- ecdf(as.numeric(MPF.SunLife.RSI.month[1:row, col]))
      if (percentile(MPF.SunLife.RSI.month[row, col]) >=
          (RSI_Overbuy - 1 /length(1:row) ^ (1 / 2))) {
        MPF.SunLife.RSI.p[row, col] <- 0.2
      } else {
        MPF.SunLife.RSI.p[row, col] <-
          1.2 - (percentile(MPF.SunLife.RSI.month[row, col]) ^ (3))
      }
    }
  } else {
    MPF.SunLife.RSI.p[, col] <- 1
  }
}

MPF.SunLife.RSI.sum <-
  as.xts(rowSums(MPF.SunLife.RSI.p), order.by = monthly)

for (row in 1:length(MPF.SunLife.RSI.p[, col])) {
  MPF.SunLife.RSI.p[row, ] <-
    apply(MPF.SunLife.RSI.p[row, ], 2, function(x)
      (x / MPF.SunLife.RSI.sum[row, 1]) ^ (1 / 4))
}
MPF.SunLife.RSI.sum <-
  as.xts(rowSums(MPF.SunLife.RSI.p), order.by = monthly)
```

Allocate MPF for n-m MA BIAS

```
rm(
  MPF.SunLife.BIAS.diff,
  MPF.SunLife.BIAS.diff.qu,
  MPF.SunLife.BIAS.month,
  MPF.SunLife.BIAS.p
)
MPF.SunLife.BIAS.diff <-
  (MPF.SunLife.SMA.mt - MPF.SunLife.SMA.hy) / MPF.SunLife.SMA.hy
MPF.SunLife.BIAS.diff.qu <-
  MPF.SunLife.BIAS.diff - stats::lag(MPF.SunLife.BIAS.diff, k = 21 * nm)
MPF.SunLife.BIAS.diff.qu <- na.fill(MPF.SunLife.BIAS.diff.qu, 0)


MPF.SunLife.BIAS.month <-
  do.call(rbind, lapply(split(
    as.xts(MPF.SunLife.BIAS.diff.qu), "months"
  ), function(x)
    colAvgs(x)))

MPF.portf.weight <- MPF.SunLife.returns
MPF.portf.weight[,] <- NA

MPF.SunLife.stock.return <-
  as.xts(rowSums(MPF.SunLife.BIAS.month), order.by = monthly)
```

```
MPF.SunLife.stock.return[] <- NA

MPF.portf.return <-
  as.xts(rowSums(MPF.SunLife.BIAS.month), order.by = monthly)
MPF.portf.return[] <- NA

up <- TRUE
hedge <- FALSE

round_percent <- function(x) {
  x <- x * 100
  result <- floor(x)    # Find integer bits
  remain <- x - result
  rsum <- sum(result)   # Find out how much we are missing
  i <- 1
  if (rsum < 100) {
    o <- order(remain, decreasing = TRUE)
    while (rsum < 100) {
      if (i > length(remain)) i <- 1
      idx <- o[i]
      if (result[idx] == 0) {
        i <- i+1
        next
      }
      result[idx] <- result[idx] + 1
      rsum <- sum(result)
      i <- i+1
    }
  }
  result <- result/100
  return(result)
}



MPF.SunLife.returns.mat <- as.matrix(MPF.SunLife.returns)
MPF.SunLife.stock.mean <- 0
for (row in 1:length(MPF.SunLife.BIAS.month[, 1])) {
  MPF.SunLife.BIAS.month[row, ] <-
    (MPF.SunLife.BIAS.month[row, ]) * MPF.SunLife.RSI.p[row, ]

  MPF.SunLife.stock.mean <- 0
  i <- 0
  for (col in 1:length(MPF.SunLife.BIAS.month[1, ])) {
    if (col != 2 &&
        col != 3 && col != 7 && col != 9 && col != 11 && col != 12) {
      if (!is.na(MPF.SunLife.returns.mat[row, col]) &&
          MPF.SunLife.returns.mat[row, col] != 0) {
        MPF.SunLife.stock.mean <-
          MPF.SunLife.stock.mean + MPF.SunLife.returns.mat[row, col]
        i <- i + 1
      }
      if (MPF.SunLife.BIAS.month[row, col] < 1e-6) {
```

```
      MPF.SunLife.BIAS.month[row, col] <- 0
    }
  } else {
    if (MPF.SunLife.BIAS.month[row, col] < 0) {
      MPF.SunLife.BIAS.month[row, col] <- 0
    }
  }

}

MPF.SunLife.stock.return[row] <- MPF.SunLife.stock.mean / i

if (is.nan(MPF.SunLife.BIAS.month[row, col])) {
  MPF.SunLife.BIAS.month[row, col] <- 0
}

# Retain two most increasing fund
last <- length(MPF.SunLife.BIAS.month[1, ]) - top

order <- order(MPF.SunLife.BIAS.month[row, ])[1:last]
for (col in order) {
  MPF.SunLife.BIAS.month[row, col] <- 0
}

MPF.SunLife.sum <- sum(MPF.SunLife.BIAS.month[row, ])
MPF.SunLife.BIAS.p <- MPF.SunLife.BIAS.month[row, ]
MPF.SunLife.BIAS.p[] <- 0


if (row > 8 && MPF.SunLife.stock.return[row] <
    quantile(na.omit(MPF.SunLife.stock.return), c(.35)) &&
    MPF.SunLife.stock.return[row - 3] <
    quantile(na.omit(MPF.SunLife.stock.return), c(.45))) {
  up <- FALSE
}

if (row > 8 && hedge &&
    MPF.SunLife.stock.return[row] >
    quantile(na.omit(MPF.SunLife.stock.return), c(.45))  &&
    MPF.SunLife.stock.return[row - 3] >
    quantile(na.omit(MPF.SunLife.stock.return), c(.35))) {
  hedge <- FALSE
  up <- TRUE
}

if (row > 8 &&  (MPF.SunLife.stock.return[row] < 0 &&
                 MPF.SunLife.stock.return[row - 1] >
                 quantile(na.omit(MPF.SunLife.stock.return), c(.85)))) {
  hedge <- TRUE
}

MPF.SunLife.sum <- sum(MPF.SunLife.BIAS.month[row,])
MPF.SunLife.BIAS.p[] <- 0
```

```r
if (row <= 12 ||
    MPF.SunLife.sum == MPF.SunLife.BIAS.month[row, 11] ||
    MPF.SunLife.sum < 1e-6 || hedge == TRUE) {
  if (row >= 189) {
    MPF.SunLife.BIAS.p[7] <- 0.3
    MPF.SunLife.BIAS.p[11] <- 0.3
    MPF.SunLife.BIAS.p[12] <- 0.4
  } else if (row >= 102) {
    MPF.SunLife.BIAS.p[7] <- 0.5
    MPF.SunLife.BIAS.p[11] <- 0.5
  } else {
    MPF.SunLife.BIAS.p[11] <- 1
  }
} else if (min(MPF.SunLife.stock.return[(row - 3):row]) < -0.08) {
  if (row >= 189) {
    MPF.SunLife.BIAS.p[] <-
      MPF.SunLife.BIAS.month[row,] / MPF.SunLife.sum / 10 * 7

    MPF.SunLife.BIAS.p[12] <- MPF.SunLife.BIAS.p[12] + 0.12
    MPF.SunLife.BIAS.p[11] <- MPF.SunLife.BIAS.p[11] + 0.08
    MPF.SunLife.BIAS.p[7] <- MPF.SunLife.BIAS.p[7] + 0.1
  } else if (row >= 102) {
    MPF.SunLife.BIAS.p[] <-
      MPF.SunLife.BIAS.month[row,] / MPF.SunLife.sum / 10 * 7
    MPF.SunLife.BIAS.p[11] <- MPF.SunLife.BIAS.p[11] + 0.12
    MPF.SunLife.BIAS.p[7] <- MPF.SunLife.BIAS.p[7] + 0.18
  } else {
    MPF.SunLife.BIAS.p[] <-
      MPF.SunLife.BIAS.month[row,] / MPF.SunLife.sum / 10 * 7
    MPF.SunLife.BIAS.p[11] <- MPF.SunLife.BIAS.p[11] + 0.3
  }
} else {
  MPF.SunLife.BIAS.p[] <-
    MPF.SunLife.BIAS.month[row,] / MPF.SunLife.sum
}


MPF.portf.weight[row, ] <-
  round_percent(MPF.SunLife.BIAS.p)

if (row == 1) {
  portf.rebal.fm <- na.fill(MPF.SunLife.simple.returns[row, ], 0) *
    MPF.portf.weight[row, ]
} else {
  portf.rebal.fm <-
    Return.portfolio(
      na.fill(MPF.SunLife.simple.returns[1:row, ], 0),
      weight = MPF.portf.weight,
      geometric = TRUE,
      rebalance_on = "months"
    )
}
```

```
  MPF.portf.return[row] <-
    tail(na.omit(portf.rebal.fm), 1)
  MPF.portf.drawdown <- Drawdowns(MPF.portf.return,
                                  geometric = TRUE)

  if (tail(na.omit(MPF.portf.drawdown), 1) < -0.065 &&
      up == FALSE) {
    hedge = TRUE
  }
}
```

Performance Analysis

```
portf.rebal.fm <- Return.portfolio(
  MPF.SunLife.simple.returns,
  weight = MPF.portf.weight,
  geometric = TRUE,
  rebalance_on = "months"
)

etfs <- c('SPY', 'QQQ', '^HSI')
golden.butterfly <- c('VTI', 'IWM', 'SHY', 'TLT', 'GLD')

weights <- c(1, 0, 0,
             0, 1, 0,
             0, 0, 1)

weights.table <-  tibble(etfs) %>%
  tq_repeat_df(n = 3) %>%
  bind_cols(tibble(weights)) %>%
  group_by(portfolio)

etf.dat <- c(etfs) %>%
  map_df(
    function(i)
      i %>%
      tq_get(
        "stock.prices",
        from = "2001-09-01" %>%
          as.Date,
        to = "2020-04-01" %>%
          as.Date
      )
  ) %>% group_by(symbol)  %>%
  tq_transmute(
    select     = adjusted,
    mutate_fun = periodReturn,
    period     = "monthly",
    col_rename = "Rb"
  )

GoldBut.dat <- c(golden.butterfly) %>%
```

```
  map_df(
    function(i)
      i %>%
      tq_get(
        "stock.prices",
        from = "2004-11-01" %>%
          as.Date,
        to = "2020-04-01" %>%
          as.Date
      )
  ) %>% group_by(symbol)  %>%
  tq_transmute(
    select     = adjusted,
    mutate_fun = periodReturn,
    period     = "monthly",
    col_rename = "Rb"
  )


etf.by.column <- spread(etf.dat, symbol, Rb)
SPY.etf <-
  na.omit(select(etf.by.column, 'date', 'SPY')) %>% mutate(symb = 'SPY')
QQQ.etf <-
  na.omit(select(etf.by.column, 'date', 'QQQ')) %>% mutate(symb = 'QQQ')
HSI.etf <-
  na.omit(select(etf.by.column, 'date', '^HSI')) %>% mutate(symb = 'HSI')

SPY.return.monthly <- SPY.etf %>%
  tq_portfolio(assets_col  = symb,
               returns_col = SPY,
               col_rename  = "Rb.SPY")
```

```
## No portfolio weights supplied. Implementing equal weighting.
```

```
QQQ.return.monthly <- QQQ.etf %>%
  tq_portfolio(assets_col  = symb,
               returns_col = QQQ,
               col_rename  = "Rb.QQQ")
```

```
## No portfolio weights supplied. Implementing equal weighting.
```

```
HSI.return.monthly <- HSI.etf %>%
  tq_portfolio(assets_col  = symb,
               returns_col = `^HSI`,
               col_rename  = "Rb.HSI")
```

```
## No portfolio weights supplied. Implementing equal weighting.
```

```
GoldBut.return.monthly <- GoldBut.dat %>%
  tq_portfolio(
    assets_col = symbol,
    returns_col = Rb,
    col_rename = "Rb.GoldBut",
    rebalance_on = "years"
  )
```

```
## No portfolio weights supplied. Implementing equal weighting.
portf.rebal.fm.dollar <- Return.portfolio(
  MPF.SunLife.simple.returns,
  weight = MPF.portf.weight,
  geometric = TRUE,
  rebalance_on = "months"
)

portf.rebal.fm.dollar.cpy <- Return.portfolio(
  MPF.SunLife.simple.returns,
  weight = MPF.portf.weight,
  geometric = TRUE,
  rebalance_on = "months",
  wealth.index = TRUE
)

portf.rebal.fm.dollar.gb <- Return.portfolio(
  tail(MPF.SunLife.simple.returns,-39),
  weight = tail(MPF.portf.weight,-39),
  geometric = TRUE,
  rebalance_on = "months"
)

portf.rebal.fm.dollar.gb.cpy <- Return.portfolio(
  tail(MPF.SunLife.simple.returns, -39),
  weight = tail(MPF.portf.weight, -39),
  geometric = TRUE,
  rebalance_on = "months",
  wealth.index = TRUE
)

xts2df <- function(x) {
  data.frame(date = index(x), coredata(x))
}

colnames(portf.rebal.fm.dollar) <- "Ra"


portf.rebal.fm.dollar.tb <-
  tbl_df(melt(xts2df(portf.rebal.fm.dollar), id = "date"))
portf.rebal.fm.dollar.tb <- portf.rebal.fm.dollar.tb[, -2]
colnames(portf.rebal.fm.dollar.tb) <- c("date", "Ra")
portf.rebal.fm.dollar.gb.tb <-
  tbl_df(melt(xts2df(portf.rebal.fm.dollar.gb), id = "date"))
portf.rebal.fm.dollar.gb.tb <- portf.rebal.fm.dollar.gb.tb[, -2]
colnames(portf.rebal.fm.dollar.gb.tb) <- c("date", "Ra")

portf.rebal.fm.dollar.tb[, 1] <- SPY.etf[, 1]
RaSPY.multiple.portfolio <- left_join(portf.rebal.fm.dollar.tb,
                                      SPY.return.monthly, by = "date")
portf.rebal.fm.dollar.tb[, 1] <- QQQ.etf[, 1]
RaQQQ.multiple.portfolio <- left_join(portf.rebal.fm.dollar.tb,
                                      QQQ.return.monthly, by = "date")
```

```
portf.rebal.fm.dollar.gb.tb[, 1] <- GoldBut.dat[, 2]
RaGoldBut.multiple.portfolio <-
  left_join(portf.rebal.fm.dollar.gb.tb,
            GoldBut.return.monthly, by = "date")
portf.rebal.fm.dollar.tb[, 1] <- HSI.etf[, 1]
RaHSI.multiple.portfolio <- left_join(portf.rebal.fm.dollar.tb,
                                      HSI.return.monthly, by = "date")


print("Annualised return of SPY:")
```

```
## [1] "Annualised return of SPY:"
```

```
TORI.SPY <- RaSPY.multiple.portfolio %>%
  tq_performance(Rb.SPY, NULL, performance_fun = table.AnnualizedReturns)
print(TORI.SPY)
```

```
## # A tibble: 1 x 3
##   AnnualizedReturn `AnnualizedSharpe(Rf=0%)` AnnualizedStdDev
##              <dbl>                     <dbl>            <dbl>
## 1           0.0739                     0.523            0.141
```

```
print("Annualised return of QQQ:")
```

```
## [1] "Annualised return of QQQ:"
```

```
TORI.QQQ <- RaQQQ.multiple.portfolio %>%
  tq_performance(Rb.QQQ, NULL, performance_fun = table.AnnualizedReturns)
print(TORI.QQQ)
```

```
## # A tibble: 1 x 3
##   AnnualizedReturn `AnnualizedSharpe(Rf=0%)` AnnualizedStdDev
##              <dbl>                     <dbl>            <dbl>
## 1            0.108                     0.542            0.199
```

```
print("Annualised return of Golden Butterfly:")
```

```
## [1] "Annualised return of Golden Butterfly:"
```

```
TORI.GoldBut <- RaGoldBut.multiple.portfolio %>%
  tq_performance(Rb.GoldBut, NULL, performance_fun = table.AnnualizedReturns)
print(TORI.GoldBut)
```

```
## # A tibble: 1 x 3
##   AnnualizedReturn `AnnualizedSharpe(Rf=0%)` AnnualizedStdDev
##              <dbl>                     <dbl>            <dbl>
## 1           0.0788                      1.09           0.0726
```

```
print("Annualised return of HSI:")
```

```
## [1] "Annualised return of HSI:"
```

```
TORI.HSI <- RaHSI.multiple.portfolio %>%
  tq_performance(Rb.HSI, NULL, performance_fun = table.AnnualizedReturns)
print(TORI.HSI)
```

```
## # A tibble: 1 x 3
##   AnnualizedReturn `AnnualizedSharpe(Rf=0%)` AnnualizedStdDev
##              <dbl>                     <dbl>            <dbl>
```

```
## 1           0.0484                      0.242            0.200
```

```
print("CAPM Information against SPY:")
```

```
## [1] "CAPM Information against SPY:"
```

```
TORI.SPY <- RaSPY.multiple.portfolio %>%
  tq_performance(Ra, Rb.SPY, performance_fun = table.CAPM)
print(TORI.SPY)
```

```
## # A tibble: 1 x 12
##   ActivePremium  Alpha AnnualizedAlpha   Beta `Beta-` `Beta+` Correlation
##           <dbl>  <dbl>           <dbl>  <dbl>   <dbl>   <dbl>       <dbl>
## 1        0.0331 0.0085           0.107 0.0428   0.128  0.0963      0.0673
## # ... with 5 more variables: `Correlationp-value` <dbl>,
## #   InformationRatio <dbl>, `R-squared` <dbl>, TrackingError <dbl>,
## #   TreynorRatio <dbl>
```

```
print("CAPM Information against QQQ:")
```

```
## [1] "CAPM Information against QQQ:"
```

```
TORI.QQQ <- RaQQQ.multiple.portfolio %>%
  tq_performance(Ra, Rb.QQQ, performance_fun = table.CAPM)
print(TORI.QQQ)
```

```
## # A tibble: 1 x 12
##   ActivePremium  Alpha AnnualizedAlpha   Beta `Beta-` `Beta+` Correlation
##           <dbl>  <dbl>           <dbl>  <dbl>   <dbl>   <dbl>       <dbl>
## 1     -0.000600 0.0086           0.108 0.0253  0.0681   0.087      0.0558
## # ... with 5 more variables: `Correlationp-value` <dbl>,
## #   InformationRatio <dbl>, `R-squared` <dbl>, TrackingError <dbl>,
## #   TreynorRatio <dbl>
```

```
print("CAPM Information against GoldBut:")
```

```
## [1] "CAPM Information against GoldBut:"
```

```
TORI.GoldBut <- RaGoldBut.multiple.portfolio %>%
  tq_performance(Ra, Rb.GoldBut, performance_fun = table.CAPM)
print(TORI.GoldBut)
```

```
## # A tibble: 1 x 12
##   ActivePremium  Alpha AnnualizedAlpha   Beta `Beta-` `Beta+` Correlation
##           <dbl>  <dbl>           <dbl>  <dbl>   <dbl>   <dbl>       <dbl>
## 1         0.026 0.0083           0.104 0.0604 -0.0925   0.134      0.0477
## # ... with 5 more variables: `Correlationp-value` <dbl>,
## #   InformationRatio <dbl>, `R-squared` <dbl>, TrackingError <dbl>,
## #   TreynorRatio <dbl>
```

```
print("CAPM Information against HSI:")
```

```
## [1] "CAPM Information against HSI:"
```

```
TORI.HSI <- RaHSI.multiple.portfolio %>%
  tq_performance(Ra, Rb.HSI, performance_fun = table.CAPM)
print(TORI.HSI)
```

```
## # A tibble: 1 x 12
```

```
##   ActivePremium  Alpha AnnualizedAlpha  Beta `Beta-` `Beta+` Correlation
##           <dbl>  <dbl>           <dbl> <dbl>   <dbl>   <dbl>       <dbl>
## 1        0.0587 0.0086           0.108 0.046  0.0079 -0.0009       0.102
## # ... with 5 more variables: `Correlationp-value` <dbl>,
## #   InformationRatio <dbl>, `R-squared` <dbl>, TrackingError <dbl>,
## #   TreynorRatio <dbl>
```

```
print("Information Ratio against SPY:")
```

```
## [1] "Information Ratio against SPY:"
```

```
TORI.SPY <- RaSPY.multiple.portfolio %>%
  tq_performance(Ra, Rb.SPY, performance_fun = table.InformationRatio)
print(TORI.SPY)
```

```
## # A tibble: 1 x 3
##   AnnualisedTrackingError InformationRatio TrackingError
##                     <dbl>            <dbl>         <dbl>
## 1                   0.162            0.204        0.0469
```

```
print("Information Ratio against QQQ:")
```

```
## [1] "Information Ratio against QQQ:"
```

```
TORI.QQQ <- RaQQQ.multiple.portfolio %>%
  tq_performance(Ra, Rb.QQQ, performance_fun = table.InformationRatio)
print(TORI.QQQ)
```

```
## # A tibble: 1 x 3
##   AnnualisedTrackingError InformationRatio TrackingError
##                     <dbl>            <dbl>         <dbl>
## 1                   0.213         -0.00290        0.0616
```

```
print("Information Ratio against GoldBut:")
```

```
## [1] "Information Ratio against GoldBut:"
```

```
TORI.GoldBut <- RaGoldBut.multiple.portfolio %>%
  tq_performance(Ra, Rb.GoldBut, performance_fun = table.InformationRatio)
print(TORI.GoldBut)
```

```
## # A tibble: 1 x 3
##   AnnualisedTrackingError InformationRatio TrackingError
##                     <dbl>            <dbl>         <dbl>
## 1                   0.114            0.228         0.033
```

```
print("Information Ratio against HSI:")
```

```
## [1] "Information Ratio against HSI:"
```

```
TORI.HSI <- RaHSI.multiple.portfolio %>%
  tq_performance(Ra, Rb.HSI, performance_fun = table.InformationRatio)
print(TORI.HSI)
```

```
## # A tibble: 1 x 3
##   AnnualisedTrackingError InformationRatio TrackingError
##                     <dbl>            <dbl>         <dbl>
## 1                   0.211            0.278        0.0608
```

```
SPY.dollar.monthly <- SPY.etf %>%
  tq_portfolio(
    assets_col  = symb,
    returns_col = SPY,
    col_rename  = "Rb.SPY",
    wealth.index = TRUE
  ) %>%
  mutate(Rb.SPY = Rb.SPY * 2301)
```

```
## No portfolio weights supplied. Implementing equal weighting.
```

```
QQQ.dollar.monthly <- QQQ.etf %>%
  tq_portfolio(
    assets_col  = symb,
    returns_col = QQQ,
    col_rename  = "Rb.QQQ",
    wealth.index = TRUE
  ) %>%
  mutate(Rb.QQQ = Rb.QQQ * 2301)
```

```
## No portfolio weights supplied. Implementing equal weighting.
```

```
GoldBut.dollar.monthly <- GoldBut.dat %>%
  tq_portfolio(
    assets_col = symbol,
    returns_col = Rb,
    col_rename = "Rb.GoldBut",
    rebalance_on = "years",
    wealth.index = TRUE
  ) %>%
  mutate(Rb.GoldBut = Rb.GoldBut * 2301)
```

```
## No portfolio weights supplied. Implementing equal weighting.
```

```
HSI.dollar.monthly <- HSI.etf %>%
  tq_portfolio(
    assets_col  = symb,
    returns_col = `^HSI`,
    col_rename  = "Rb.HSI",
    wealth.index = TRUE
  ) %>%
  mutate(Rb.HSI = Rb.HSI * 2301)
```

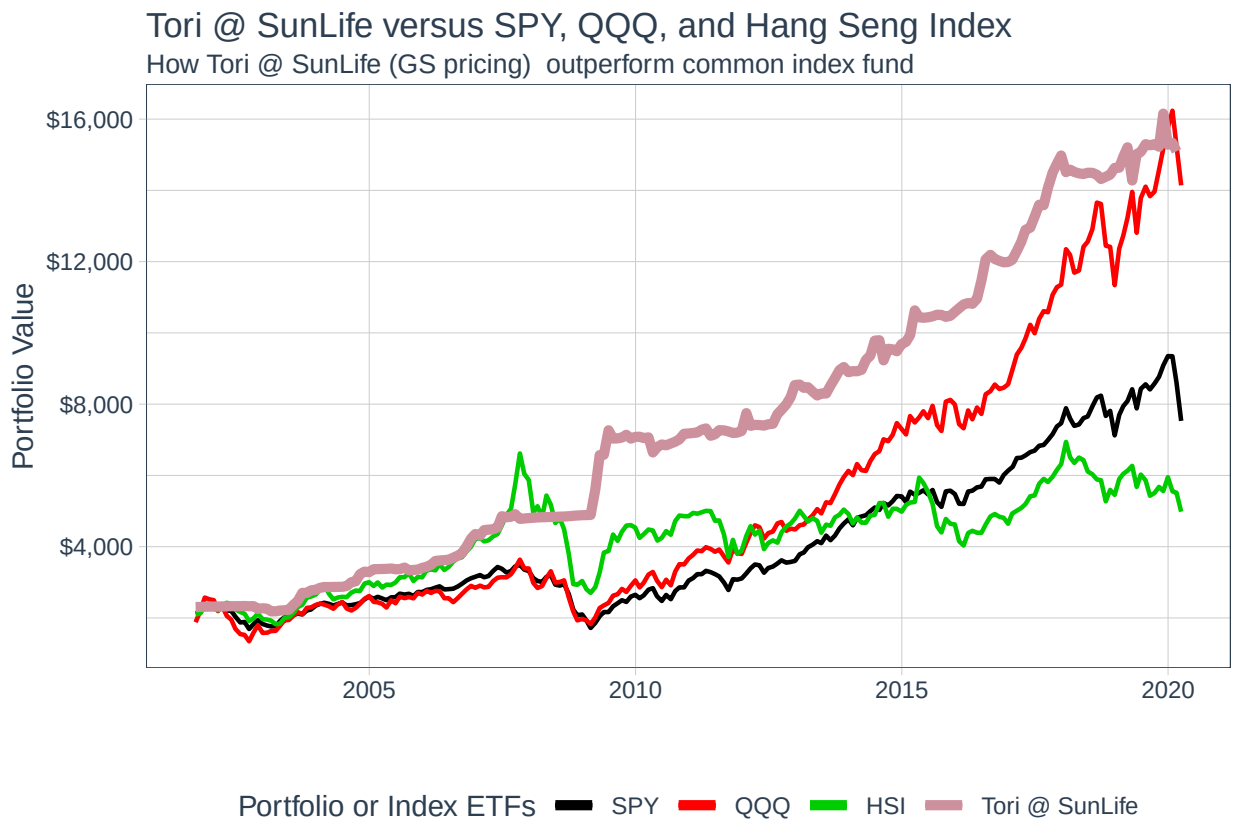
```
## No portfolio weights supplied. Implementing equal weighting.
```

```
portf.rebal.fm.dollar.tb.cpy <-
  tbl_df(melt(
    xts2df(portf.rebal.fm.dollar.cpy),
    id = "date",
    variable_name = "Ra"
  ))
portf.rebal.fm.dollar.tb.cpy <- portf.rebal.fm.dollar.tb.cpy[, -2]
colnames(portf.rebal.fm.dollar.tb.cpy) <- c("date", "Ra")
portf.rebal.fm.dollar.tb.cpy[, 1] <- HSI.etf[, 1]
portf.dollar.monthly <- portf.rebal.fm.dollar.tb.cpy %>%
  mutate(Ra = Ra * 2301)
```

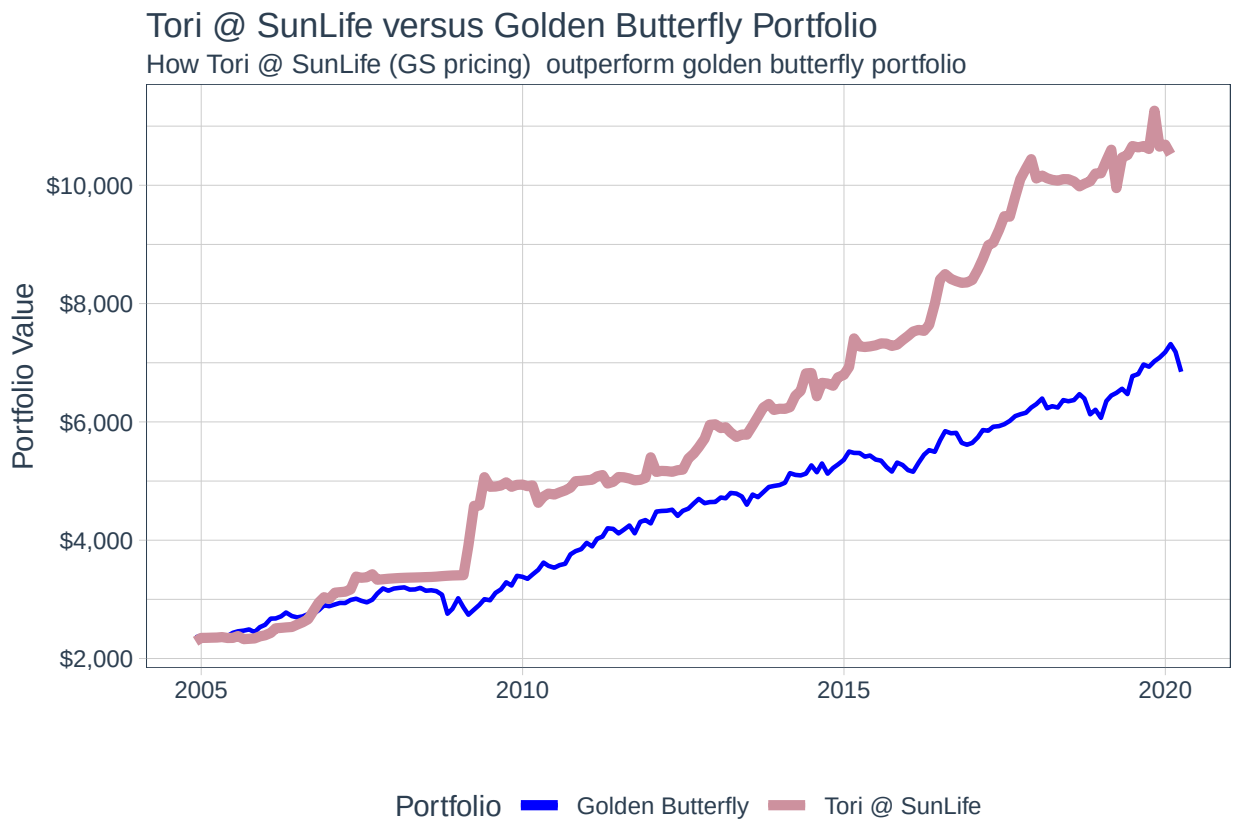
```r
portf.rebal.fm.dollar.gb.tb.cpy <-
  tbl_df(melt(
    xts2df(portf.rebal.fm.dollar.gb.cpy),
    id = "date",
    variable_name = "Ra"
  ))
portf.rebal.fm.dollar.gb.tb.cpy <-
  portf.rebal.fm.dollar.gb.tb.cpy[, -2]
colnames(portf.rebal.fm.dollar.gb.tb.cpy) <- c("date", "Ra")
portf.rebal.fm.dollar.gb.tb.cpy[, 1] <- GoldBut.dat[, 2]
portf.dollar.monthly.gb <- portf.rebal.fm.dollar.gb.tb.cpy %>%
  mutate(Ra = Ra * 2301)



ggplot() +
  scale_color_manual(
    labels = c("SPY", "QQQ", "HSI", "Tori @ SunLife"),
    values = c(1, 2, 3, "pink3")
  ) +
  geom_line(aes(x = date, y = Rb.SPY, color = as.factor(1)),
            SPY.dollar.monthly,
            size = 0.8) +
  geom_line(aes(x = date, y = Rb.QQQ, color = as.factor(2)),
            QQQ.dollar.monthly,
            size = 0.8) +
  geom_line(aes(x = date, y = Rb.HSI, color = as.factor(3)),
            HSI.dollar.monthly,
            size = 0.8) +
  geom_line(aes(x = date, y = Ra, color = "pink3"), portf.dollar.monthly, size =
              1.8) +
  labs(
    title = "Tori @ SunLife versus SPY, QQQ, and Hang Seng Index",
    subtitle = "How Tori @ SunLife (GS pricing)  outperform common index fund",
    x = "",
    y = "Portfolio Value",
    color = "Portfolio or Index ETFs"
  ) +
  theme_tq() +
  scale_y_continuous(labels = scales::dollar)
```




```
ggplot() +
  scale_color_manual(
    labels = c("Golden Butterfly", "Tori @ SunLife"),
    values = c(4, "pink3")
  ) +
  geom_line(aes(x = date, y = Rb.GoldBut, color = as.factor(3)),
            GoldBut.dollar.monthly,
            size = 0.8) +
  geom_line(aes(x = date, y = Ra, color = "pink3"),
            portf.dollar.monthly.gb,
            size = 1.8) +
  labs(
    title = "Tori @ SunLife versus Golden Butterfly Portfolio",
    subtitle = "How Tori @ SunLife (GS pricing)  outperform golden butterfly portfolio",
    x = "",
    y = "Portfolio Value",
    color = "Portfolio"
  ) +
  theme_tq() +
  scale_y_continuous(labels = scales::dollar)
```




```
cat("Cumulative return: ",
    Return.cumulative(portf.rebal.fm, geometric = TRUE),
    "\n")
```

```
## Cumulative return:  5.560826
```

```
portf.rebal.fm.sharpe <-
  Return.annualized(portf.rebal.fm, geometric = TRUE) /
  (StdDev.annualized(portf.rebal.fm))
rownames(portf.rebal.fm.sharpe) <- "Sharpe Ratio"
cat("Annualised return: ",
    Return.annualized(portf.rebal.fm, geometric = TRUE),
    "\n")
```

```
## Annualised return:  0.1070313
```

```
mean.annual.return <-
  mean(do.call(rbind, lapply(split(portf.rebal.fm, "years"), function(x)
    colMeans(x))) * 12)
cat("Mean annual return (for ref. only): ", mean.annual.return, "\n")
```

```
## Mean annual return (for ref. only):  0.08894564
```

```
portf.rebal.fm.sharpe.mean <-
  mean.annual.return / (StdDev.annualized(portf.rebal.fm))
rownames(portf.rebal.fm.sharpe.mean) <-
  "Sharpe Ratio (Mean annual return)"
```

```
SharpeRatio(portf.rebal.fm, annualize = TRUE, method = "modified")
```

```
##                                          portfolio.returns
## Annualized StdDev Sharpe (Rf=0%, p=95%):          1.189139
## Annualized VaR Sharpe (Rf=0%, p=95%):            10.678239
## Annualized ES Sharpe (Rf=0%, p=95%):              2.007761
```

```
cat("Annualised Standard Deviation: ",
    StdDev.annualized(portf.rebal.fm),
    "\n")
```

```
## Annualised Standard Deviation:  0.09000745
```

```
cat("Sortino Ratio: ", SortinoRatio(portf.rebal.fm) * sqrt(12), "\n")
```

```
## Sortino Ratio:  2.933701
```

```
cat(
  "Coditional Value of Risk (CVaR, Expected Shortfall ES",
  ES(portf.rebal.fm, method = "modified"),
  "\n"
)
```

```
## Coditional Value of Risk (CVaR, Expected Shortfall ES -0.05330881
```

```
tail(head(MPF.portf.weight,n=-1), n = 6)
```

```
##               AE B CA   CE FTSE HK G   GB GE HKB  HKE MPFC SFP
## 2019-09-30 0.00 0  0 0.00       0 0 0.94  0   0 0.00 0.06 0.0
## 2019-10-31 0.48 0  0 0.52       0 0 0.00  0   0 0.00 0.00 0.0
## 2019-11-29 0.51 0  0 0.49       0 0 0.00  0   0 0.00 0.00 0.0
## 2019-12-31 0.55 0  0 0.00       0 0 0.00  0   0 0.45 0.00 0.0
## 2020-01-31 0.00 0  0 0.00       0 0 0.30  0   0 0.00 0.30 0.4
## 2020-02-28 0.00 0  0 0.00       0 0 0.30  0   0 0.00 0.30 0.4
```

Monthly Installment

```
MPF.SunLife.units <- MPF.SunLife.returns
MPF.SunLife.units[,] <- 0

MPF.monthly.asset <- MPF.SunLife.returns
MPF.monthly.asset[,] <- 0

MPF.monthly.returns <-
  as.xts(rowSums(MPF.SunLife.returns), order.by = monthly)
MPF.monthly.returns[] <- 0

MPF.time <- 0:length(MPF.SunLife.returns[, 1]) / 12
MPF.pay <- -2301 + 0 * MPF.time


for (row in 1:length(MPF.SunLife.returns[, 1])) {
  #for (row in 1:13) {
  this.price <- as.matrix(MPF.SunLife[monthly[row]])
  MPF.SunLife.units[row,] <- this.price
```

```
  if (row == 1) {
    last.value <- 2301
    this.value <- last.value * this.price[11]
    MPF.monthly.returns[row] <-
      (sum(na.fill(this.value, 0)) - 2301) / 2301
    MPF.monthly.asset[row, ] <-
      na.fill(((this.value + 2301) / this.price * MPF.portf.weight[row,]), 0)
    last.price <- this.price
  } else {
    last.value <-
      as.numeric(sum(na.fill(last.price * MPF.monthly.asset[row - 1, ], 0)))
    this.value <-
      as.numeric(sum(na.fill(this.price * MPF.monthly.asset[row - 1, ], 0)))
    MPF.monthly.returns[row] <- (this.value -last.value) / last.value
    MPF.monthly.asset[row,] <-
      (this.value + 2301) / this.price * MPF.portf.weight[row, ]
    last.price <- this.price
  }
}


total.asset.value <- sum(MPF.monthly.asset[row,] * this.price)
total.contribution <- 2301 * length(MPF.SunLife.returns[, 1])

MPF.pay[row + 1] <- total.asset.value
IRR.f <- function(r)
  sum(MPF.pay * exp(-r * MPF.time))
IRR.root <- uniroot(IRR.f, c(0, 1))

cat("Total asset in the last valuation date: ", total.asset.value, "\n")
```

```
## Total asset in the last valuation date:  1337808
```

```
cat("Total contribution starting from 2001/09/17: ",
    total.contribution,
    "\n")
```

```
## Total contribution starting from 2001/09/17:  513123
```

```
mean.monthly.annual.return <-
  mean(do.call(rbind, lapply(split(MPF.monthly.returns, "years"), function(x)
    colMeans(x))) * 12)
cat("Mean Annual Return: ", mean.monthly.annual.return, "\n")
```

```
## Mean Annual Return:  0.08393725
```

```
cat("Internal Rate of Return (IRR): ", IRR.root$root, "\n")
```

```
## Internal Rate of Return (IRR):  0.09034327
```

```
stddev.monthly <- (StdDev(MPF.monthly.returns) * sqrt(12))
monthly.installment.sharpe.ratio <-
  mean.monthly.annual.return / stddev.monthly
rownames(monthly.installment.sharpe.ratio) <-
  "Sharpe Ratio (mean annual return)"
```

```r
StdDev.annualized(MPF.monthly.returns)
```

```
##                                       [,1]
## Annualized Standard Deviation 0.08989431
```

```r
monthly.installment.sharpe.ratio
```

```
##                                         [,1]
## Sharpe Ratio (mean annual return) 0.9337326
```

```r
SortinoRatio(MPF.monthly.returns) * sqrt(12)
```

```
##                                [,1]
## Sortino Ratio (MAR = 0%) 2.660037
```

```r
ES(MPF.monthly.returns, method = "historical")
```

```
##           [,1]
## ES -0.04125601
```