Question 6

Tashen Naidoo

2023-11-27

```r
library(tidyverse);library(gt);library(tbl2xts);library(PerformanceAnalytics);library(lubridate);library
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0     v purrr   1.0.1
## v tibble  3.2.1     v dplyr   1.1.3
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.4     v forcats 0.5.2

## Warning: package 'tibble' was built under R version 4.2.3

## Warning: package 'dplyr' was built under R version 4.2.3

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Warning: package 'gt' was built under R version 4.2.3

## Loading required package: xts

## Warning: package 'xts' was built under R version 4.2.3

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.2.3

##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
##
## ######################### Warning from 'xts' package ##########################
## #                                                                             #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                           #
```

```
## #                                                                              #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop          #
## # dplyr from breaking base R's lag() function.                               #
## #                                                                              #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                              #
## ###############################################################################
##
## Attaching package: 'xts'
##
## The following objects are masked from 'package:dplyr':
##
##     first, last
##
##
## Attaching package: 'PerformanceAnalytics'
##
## The following object is masked from 'package:graphics':
##
##     legend
##
## Loading required package: timechange
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

## Warning: package 'RcppRoll' was built under R version 4.2.3

## Warning: package 'rugarch' was built under R version 4.2.3

## Loading required package: parallel
##
## Attaching package: 'rugarch'
##
## The following object is masked from 'package:purrr':
##
##     reduce
##
## The following object is masked from 'package:stats':
##
##     sigma

## Warning: package 'forecast' was built under R version 4.2.3

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

```
list.files('/code',full.names = T, recursive = T) %>%
    as.list() %>%
    walk(~source(.))
```

Question Six: Portfolio Construction

The data sets provided contain different asset classes.

```
MAA <- read_rds("data/MAA.rds")
msci <-read_rds("data/msci.rds") %>%
filter(Name %in% c("MSCI_ACWI", "MSCI_USA", "MSCI_RE", "MSCI_Jap"))
```

Let's add the Asset class names to each data set. The *MAA* data set refers to the Credit and Bond types, whilst the *msci* data set contains equity information. The *msci* data set has already been filtered, so a simple character column will be added. The text and image documens were used to provided more insight into what each of the asset Names are classified as.

```
#Identify what is important from the MAA data set, by looking at the text document and image provided.
Bonds <- c("Bbg_EuroBonds_UnhedgedEUR", "Bbg_GlBonds_HedgedUSD", "Bbg_USBonds_UnhedgedUSD")
Credit <- c("Bbg_EUCorpCred_Unhedged_USD","Bbg_GlCorpCred_Hedged_USD","Bbg_USCorpCred_Unhedged_USD")
Equity <- c("MSCI_ACWI", "MSCI_USA", "MSCI_RE", "MSCI_Jap")

MAA <- MAA %>%
    mutate(Class = ifelse(Name %in% Bonds, "Bonds",
                ifelse(Name %in% Credit, "Credit",
                ifelse( Name == c("Asia_dollar_Idx","Dollar_Idx"), "Currency", "Commodity"))))

#msci only contains equity data
msci <- msci %>%
    mutate(Class = "Equity")

#Combine Data sets
full_set <- full_join(MAA %>% select(date, Name, Price, Class), msci) %>%
    arrange(date)
```

```
## Joining with `by = join_by(date, Name, Price, Class)`
```

Now that we have the full data set, let's filter it by the constraints provided.

```
library(lubridate)
#Only want data from 2010 onwards and only select that is present for more than 3 years.

full_set <- full_set %>%
    filter(date > lubridate::ymd(20091231) ) %>%
    mutate(Year = format(date, "%Y"), Month = format(date,"%B"), Day = format(date, "%A")) %>%
    group_by(Name) %>%
    filter(n_distinct(Year) > 3) %>%
    ungroup()

#no rebalnce period month and day was provided, only that it needs to be done every quarter, so I will
```

```r
full_set_reb <- full_set %>%
    filter(Month %in% c("March","June", "September", "December")) %>%
    select(date, Year, Month, Day) %>%
    unique() %>%
    group_by(Month) %>%
    filter(Day == "Friday") %>%
    group_by(Year, Month) %>%
    slice(3) %>% #to get the third friay of the month
    ungroup() %>%
    arrange(date)

#Now let's filter this rebalance period out.
full_set_filtered <- full_set %>%
    filter(date %in% full_set_reb$date)
```

Now the constraints are implemented, let's look at the returns for each.

```r
Return <- full_set_filtered %>%
    group_by(Name,Class) %>%
    mutate(Returns = Price/lag(Price)-1) %>% #simple returns
    filter(date > first(date)) %>%
    select(-Year,-Month,-Day) %>%
    ungroup()
```

In order to optimise the portfolio, we need to assess the data and note if there are any missing values.

```r
library(tbl2xts)
Returnx <- Return %>% select(date, Name, Returns) %>%
    spread(Name, Returns)

colSums(is.na(Returnx))
```

```
##                        date              Asia_dollar_Idx
##                           0                            1
## Bbg_EUCorpCred_Unhedged_USD   Bbg_EuroBonds_UnhedgedEUR
##                           0                            0
##       Bbg_GlBonds_HedgedUSD   Bbg_GlCorpCred_Hedged_USD
##                           0                            0
##     Bbg_USBonds_UnhedgedUSD Bbg_USCorpCred_Unhedged_USD
##                           0                            0
##                  Commod_Idx                   Dollar_Idx
##                           0                            1
##                   MSCI_ACWI                     MSCI_Jap
##                           0                            0
##                     MSCI_RE                     MSCI_USA
##                           0                            0
```

We note that there is only two missing pieces of data, as not all the asset classes begin from the same date. Therefore, we will need to utilise the *filling_in_the_gaps* function to deal with this issue. We will use the *Drawn_Distribution_Own* method.

```r
source("C:/Users/tashe/Desktop/Financial Econometrics Exam/code/Missing_Data.R")
Returnx <- filling_in_the_gaps(Returnx, fill_amalgam = "Drawn_Distribution_Own")
#now convert to xts format
colSums(is.na(Returnx))
```

```
##                         date              Asia_dollar_Idx
##                            0                            0
## Bbg_EUCorpCred_Unhedged_USD   Bbg_EuroBonds_UnhedgedEUR
##                            0                            0
##       Bbg_GlBonds_HedgedUSD   Bbg_GlCorpCred_Hedged_USD
##                            0                            0
##     Bbg_USBonds_UnhedgedUSD Bbg_USCorpCred_Unhedged_USD
##                            0                            0
##                  Commod_Idx                  Dollar_Idx
##                            0                            0
##                   MSCI_ACWI                    MSCI_Jap
##                            0                            0
##                     MSCI_RE                    MSCI_USA
##                            0                            0
```

Now that the data set is square, we are able to get the variance, mean and add additional constraints for the portfolio. This needs to be done before we are able to identify the optimal weights vector.

```r
Returnx_dateless <- data.matrix(Returnx[,-1])


#variance (Sigma)
Sigma <- RiskPortfolios::covEstimation(Returnx_dateless)
Mean <- Returnx %>%
    summarise(across(.cols = -date, .fns = ~prod(1+.)^(1/n())-1 )) %>%
    purrr::as_vector()
```

```r
#Re-organise
Returnx_upper <- Returnx %>%
    mutate(across(.cols = -date, .fns = ~prod(1+.)^(1/n())-1 )) %>%
    gather(Ticker, Returns,-date) %>%
    mutate(Class = ifelse(Ticker %in% Bonds, "Bonds",
               ifelse(Ticker %in% Credit, "Credit",
                      ifelse( Ticker %in% Equity, "Equity",
               ifelse( Ticker == c("Asia_dollar_Idx","Dollar_Idx"), "Currency", "Commodity"))))) %>%
    mutate(UB = ifelse(Class == 'Equity',0.6,
                       ifelse(Class %in% c('Commodity','Currency'), 0.15, 0.125))) #I want the split be
```

Now we have the appropriate variance (Sigma) and returns (Mean). Furthermore, the upperbound of each asset class is provided. I now want to get the optimal weights allocation for the portfolio.

```r
#Let's first run the upper-bound limit on all asset classes
source("C:/Users/tashe/Desktop/Financial Econometrics Exam/code/Optimiser.R")

UB <- 0.4 # maximum exposure of each asset
LB <- 0.001
mu <- Mean
```

```r
#Now let's get the optimal weights vector for each asset class
Weight_rest <- optim_foo(Type = "mv", mu, Sigma, LB, UB, printmsg = T )

#now we need to reallocate the optimal weights according to the constraints provided
Weight_Rest <- Weight_rest %>%
    mutate(Weight_cap = ifelse(Tickers %in% Bonds, 0.125,
                               ifelse(Tickers %in% Credit, 0.125,
                                      ifelse(Tickers %in% Equity, 0.6, 0.075))))
```

The optimal portfolio weights are provided under the weight column. All the weights are under the restriction, but do not balance to 100. Therefore, I am not certain what occured.

```r
names(Weight_Rest) <- c("Tickers", "weight", "Result", "Weight_cap")
source("C:/Users/tashe/Desktop/Financial Econometrics Exam/code/Cap.R")

Proportional_Cap_Foo(Weight_Rest %>% filter(Weight_cap == 0.6), 0.6)
```

```
## # A tibble: 4 x 4
##   Tickers    weight Result        Weight_cap
##   <chr>       <dbl> <glue>             <dbl>
## 1 MSCI_ACWI 0.400   Converged: mv        0.6
## 2 MSCI_Jap  0.150   Converged: mv        0.6
## 3 MSCI_RE   0.00100 Converged: mv        0.6
## 4 MSCI_USA  0.400   Converged: mv        0.6
```

This is an attempt to see if the portfolio weights shift proportionally to other assets, based on the restrictions. However, the weights are exactly the same for the Equities.