Automatic Rebalancing

Nikoloz Jaghiashvili

11/1/2020

The following project aims to decrease human workload and the cost of month-end fixed income portfolio rebalancing. During the end of each month, the benchmark for the fixed income portfolio changes, which in turn changes the net exposures of the portfolio. Hence, the need to rebalance the portfolio and return it to the original net exposures. For the code to independently carry out the task, the rebalancing process needs to be divided into several parts.

First, the key rate durations of a desirable trade need to be identified, which equals the difference between exposures of the projected and current benchmarks. Second, it is essential to identify the pool of potential securities, to construct the rebalancing trade. Finally, after selecting the most attractive set of securities, the code adjusts the size of each position to achieve desirable net exposure. The code also minimizes the residual cash after the rebalancing, as well as the net size of the trade.

At all times, there are around 300 US treasuries available to purchase, each with different maturity, coupon, and duration. Hence, a bond price is not a useful indicator of its attractiveness. To differentiate between treasuries the code calibrates the Nelson Siegel Svensson model (1994) and fits the treasury yield curve. Each treasury is then re-priced using the new yield curve. The lower the actual price compared to the fitted prices of the treasury, the higher the probability it will be purchased, and the lower the probability it will be sold for the rebalancing. The logic works in reverse for treasuries that are more expensive than their fitted price.

Figure 1 shows the key rate dollar duration of the portfolio and the original benchmark. The difference between the two or the net exposure of the portfolio is shown in the second axes. Figure 2 shows the key rate dollar duration of the portfolio and the new benchmark. It can be seen that substituting the benchmark changes the net exposure of the portfolio. Figure 3 shows the difference between the two net exposures, which is the characteristic of the trade the code will try to construct.

After selecting the securities and the characteristics of target trade, Nelder–Mead minimization algorithm minimizes the root squared error of weighted key rate duration and residual cash errors, as well as the size of the trade.

The user specifies the desired number of trades before the optimization. The Table shows the resulting five trade rebalancing. This example presents the most straightforward use of the program, which can also be used to turn investment ideas into trades. While the program can measure other instruments, as presented in Figure 4, it is not able to trade them. More work is required to allow the program to trade other instruments.

	Security	ISIN		Sign	position	Bid	Ask
1	T 2 1/4 02/15/21	9128283X	Govt	Sell	$47,962,096	101.0924	101.1002
2	T 8 1/8 05/15/21	912810EJ	Govt	Buy	$6,744,513	107.9747	108.0020
3	T 8 11/15/21	912810EL	Govt	Sell	$4,794,737	102.8122	102.8434
4	T 2 07/31/22	912828XQ	Govt	Buy	$10,499,135	103.7140	103.7296
5	T 1 3/8 09/30/23	912828T2	Govt	Buy	$38,749,852	103.5543	103.5778

Figure 1. Current Exposures (Dollar Duration)

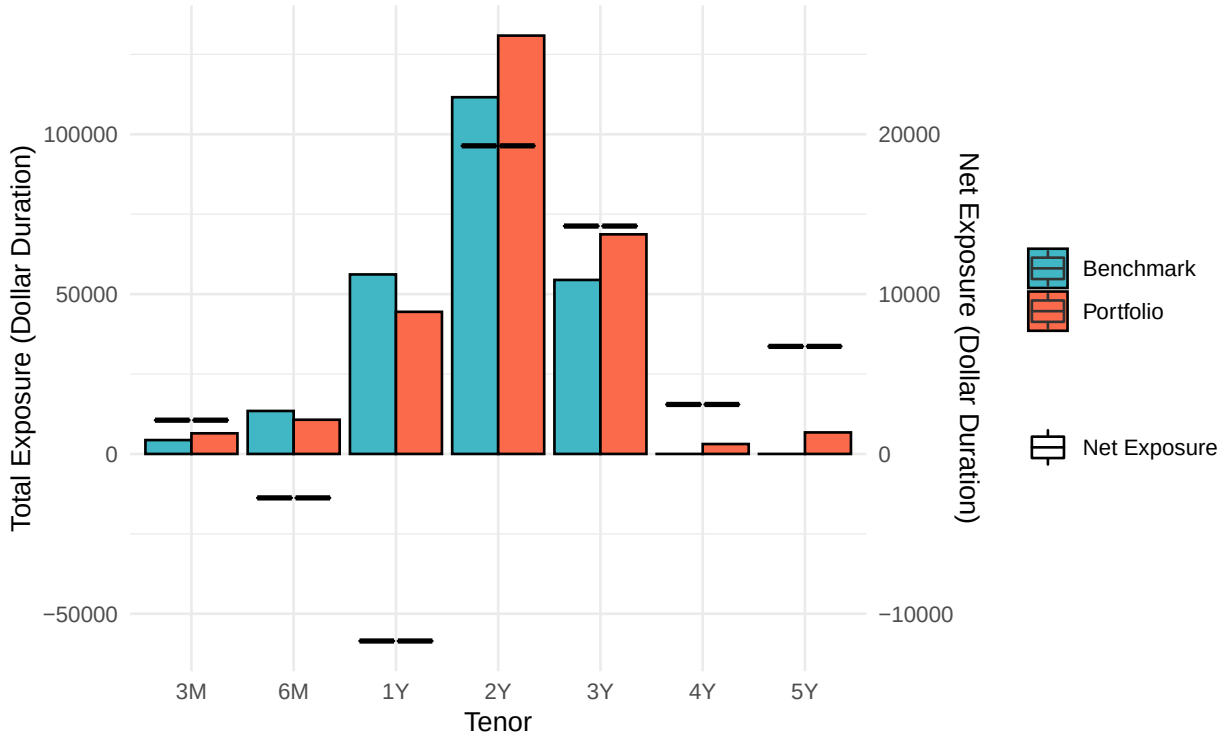


Figure 2. Projected Exposures (Dollar Duration)

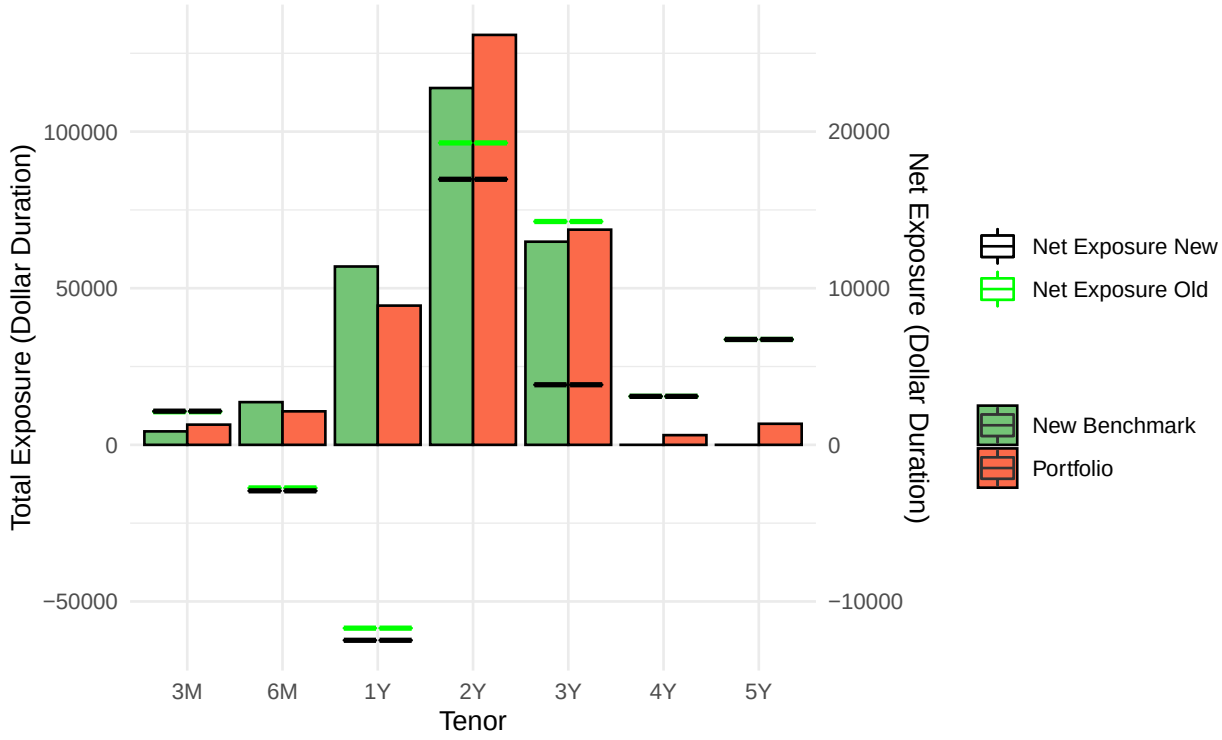


Figure 3. Trade Exposures (Dollar Duration)

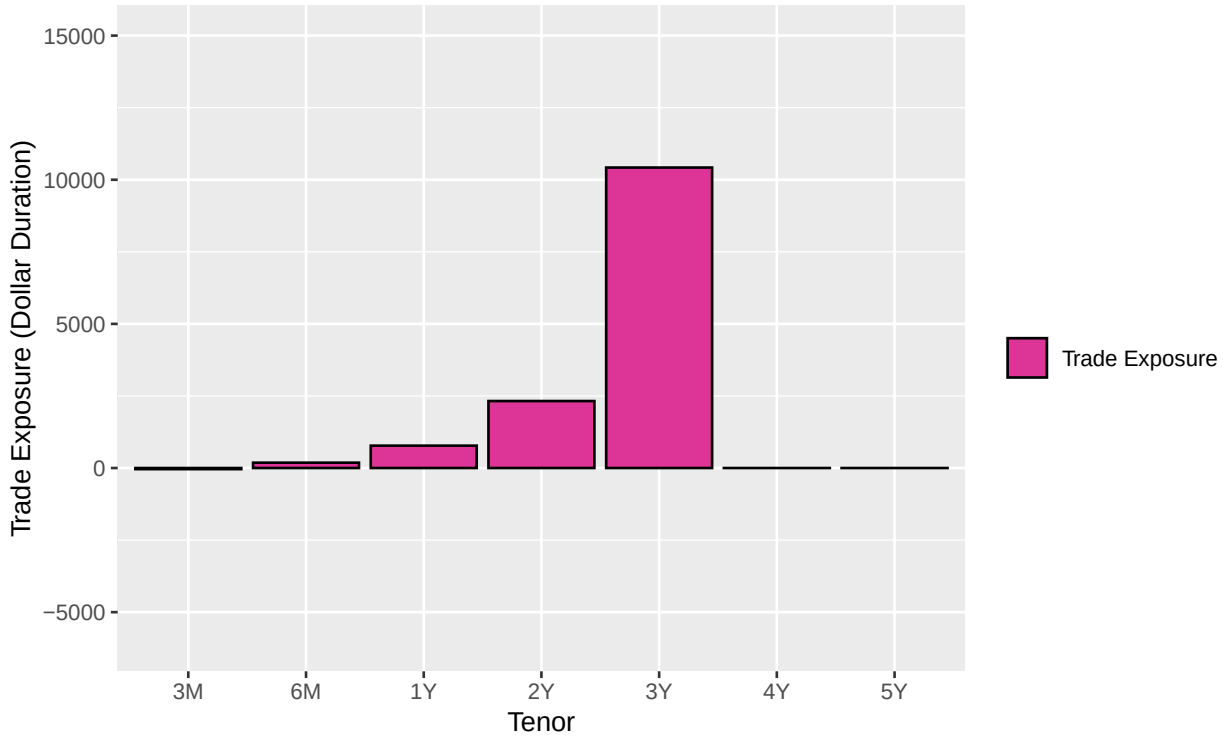
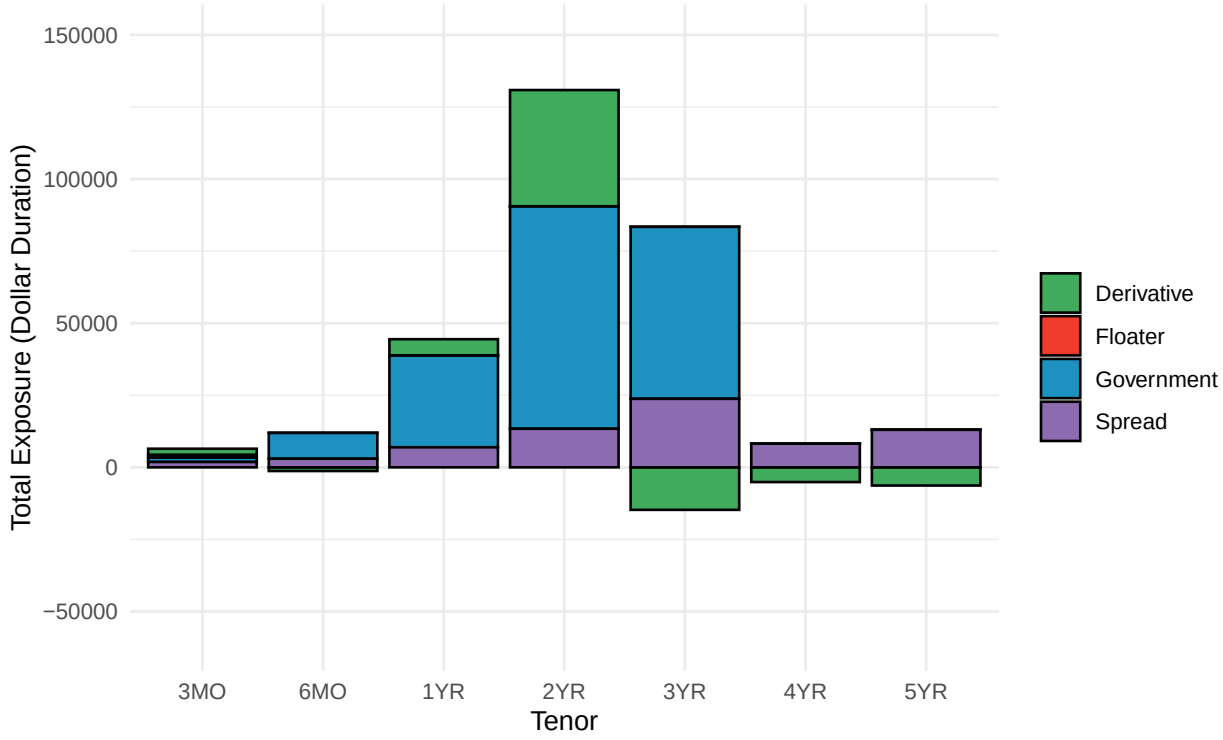


Figure 4. Instrument Exposures (Dollar Duration)



Source Code

```
library("Rblpapi")
con <-blpConnect()
library(jrvFinance)
library(neldermead)
library(optimization)
library(lme4)
library(data.table)
library(dplyr)
library(formattable)
library(tidyr)
library(RColorBrewer)
library(openxlsx)
library(timeDate)
library(NMOF)
library(arrangements)
library(lpSolve)
library(readxl)
library(shiny)
library(stringr)
library(ggplot2)
library(gridExtra)
library(grid)
library(priceR)
options(scipen=999)


Data_gen = function(type){

  set_param = function(){

    # set_param() Function sets start and end date for the PCA dataset,
    # timestamp for portfolio call, portfolio IDs, and tenors

    starting <<- "2019-07-09"
    ending <<- Sys.Date()-1
    port_time <<- format(as.POSIXct(ending,format='%Y-%m-%d'),format='%Y%m%d')
    portfolio <<- "U7024829-2 Client"
    benchmark <<- "U7024829-7 Client"
    projected_benchmark <<- "U20477317-98 Client"
    tenor <<- c("3MO","6MO","1YR","2YR","3YR","4YR","5YR")
    tenor_der <<- c("3M","6M","1Y","2Y","3Y","4Y","5Y")
  }



  dt_prep = function(type){

    # dt_prep() function takes portfolio type as an input and calls portfolio data using
    # Bloomberg API the output includes a data frame with individual securities as rows
    # and Security ISIN, position size type of instrument, and type of coupon as rows
```

```
  if (type == "Portfolio"){
    overrides = c("REFERENCE_DATE" = port_time)
    data =  as.data.frame(getPortfolio(portfolio,
                                       "Portfolio_Data", overrides = overrides))
    Port = data$Security
    Pos = data$Position
  }

  if (type == "Benchmark"){
    overrides = c("REFERENCE_DATE" = port_time)
    data =  as.data.frame(getPortfolio(benchmark,
                                       "Portfolio_Data", overrides = overrides))
    Port = data$Security
    Pos = data$Position
  }

  if (type == "Projected Benchmark"){
    overrides = c("REFERENCE_DATE" = port_time)
    data =  as.data.frame(getPortfolio(projected_benchmark,
                                       "Portfolio_Data", overrides = overrides))
    Port = data$Security
    Pos = data$Position
  }

  Stype = bdp(Port,"SECURITY_TYP")$SECURITY_TYP
  cpn = bdp(Port,"CPN_TYP")$CPN_TYP

  dt = cbind.data.frame(Port,Pos, Stype, cpn)
  dt = dt[order(dt[,3]),]
  rownames(dt) <- NULL
  colnames(dt) <- c("Sec", "Pos", "type","cpn")
  return(dt)
}



Splitter = function(type){

  # Splitter() function splits portfolio dataset into five categories, including:
  # derivatives, floating rate notes, government products, spread products, and cash.
  # The function takes type of portfolio as an input and assigns each of the five
  # categories to the 15 data frames (each category X each type of portfolio).

  w = dt_prep(type)
  assign(paste0(str_sub(type,1,4)),w,envir=globalenv())
  assign(paste0(str_sub(type,1,4),"_der"),
         subset(w,w[,3]=="Financial commodity future."),envir=globalenv())
  assign(paste0(str_sub(type,1,4),"_frn"),
         subset(w,w[,4]=="FLOATING"),envir=globalenv())
  assign(paste0(str_sub(type,1,4),"_gov"),
         subset(w,w[,4]!="FLOATING" & w[,3]!="Financial commodity future." &
                  w[,3]!="SPOT"& w[,3]!="Currency OTC option"& w[,3]=="US GOVERNMENT"),
         envir=globalenv())
```

```
  assign(paste0(str_sub(type,1,4),"_spr"),
         subset(w,w[,4]!="FLOATING" & w[,3]!="Financial commodity future." &
                  w[,3]!="SPOT"& w[,3]!="Currency OTC option"& w[,3]!="US GOVERNMENT"),
         envir=globalenv())
  assign(paste0(str_sub(type,1,4),"_spo"),
         subset(w, w[,3]=="SPOT"|w[,3]=="Currency OTC option"),
         envir=globalenv())
}



KR_der = function(type){

  # KR_der() takes portfolio type as an input. The function finds derivative data
  # frame of the respective portfolio type (from 15 data frames mentioned above).
  # The function then computes and assigns key rate duration matrix as a global variable.
  # The function returns NULL if source data frame is empty or has a missing value.
  # The function also assigns a data frame with market value of each position to the
  # global environment.

  data = get(paste0(str_sub(type,1,4),"_der"))
  if (all(is.na(data)) == FALSE & nrow(data)>0){

    KRM_der = c()
    pos_der = as.numeric(data[,2])
    val_der = bdp(data[,1],"FUT_CONT_SIZE")[,1]
    mvl_der = as.data.frame(pos_der * val_der)
    colnames(mvl_der) = "Market Value"

    for (x in tenor_der) {
      fld = paste("GRID_PT_DELTA_",x,sep = "")
      k = bdp(data[,1], fld)
      KRM_der = cbind(KRM_der, k[,1])
    }
    KRM_der = as.data.frame(KRM_der)
    colnames(KRM_der) = tenor
  } else {
    KRM_der = NULL
    mvl_der= NULL
  }
  assign(paste0("KRM_",str_sub(type,1,4),"_der"),KRM_der,envir=globalenv())
  assign(paste0("Mvl_",str_sub(type,1,4),"_der"),mvl_der,envir=globalenv())
}



KR_gov = function(type){

  # KR_gov() takes portfolio type as an input. The function finds government data
  # frame of the respective portfolio type (from 15 data frames mentioned above).
  # The function then computes and assigns key rate duration matrix as a global variable.
  # The function returns NULL if source data frame is empty or has a missing value.
  # The function also assigns a data frame with market value of each position to the
```

```
  # global environment.

  data = get(paste0(str_sub(type,1,4),"_gov"))
  if (all(is.na(data)) == FALSE & nrow(data)>0){
    KRM_gov = c()
    pos_gov = as.numeric(data[,2])
    val_gov = bdp(data[,1],"PX_MID")[,1]
    mvl_gov = as.data.frame(pos_gov * val_gov*10)
    colnames(mvl_gov) = "Market Value"

    for (x in tenor) {
      fld = paste("KEY_RATE_DUR_",x,sep = "")
      k = bdp(data[,1], fld)
      KRM_gov = cbind(KRM_gov, k[,1])
    }
    KRM_gov = as.data.frame(KRM_gov)
    colnames(KRM_gov) = tenor
  } else {
    KRM_gov = NULL
    mvl_gov = NULL
  }
  assign(paste0("KRM_",str_sub(type,1,4),"_gov"),KRM_gov,envir=globalenv())
  assign(paste0("Mvl_",str_sub(type,1,4),"_gov"),mvl_gov,envir=globalenv())
}



KR_spr = function(type){

  # KR_spr() takes portfolio type as an input. The function finds spread product
  # data frame of the respective portfolio type (from 15 data frames mentioned above).
  # The function then computes and assigns key rate duration matrix as a global variable.
  # The function returns NULL if source data frame is empty or has a missing value.
  # The function also assigns a data frame with market value of each position to the
  #global environment.

  data = get(paste0(str_sub(type,1,4),"_spr"))
  if (all(is.na(data)) == FALSE & nrow(data)>0){
    KRM_spr = c()
    pos_spr = as.numeric(data[,2])
    val_spr = bdp(data[,1],"PX_MID")[,1]
    mvl_spr = as.data.frame(pos_spr * val_spr*10)
    colnames(mvl_spr) = "Market Value"

    for (x in tenor) {
      fld = paste("KEY_RATE_DUR_",x,sep = "")
      k = bdp(data[,1], fld)
      KRM_spr = cbind(KRM_spr, k[,1])
    }
    KRM_spr = as.data.frame(KRM_spr)
    colnames(KRM_spr) = tenor
  } else {
    KRM_spr = NULL
```

```
    mvl_spr = NULL
  }
  assign(paste0("KRM_",str_sub(type,1,4),"_spr"),KRM_spr,envir=globalenv())
  assign(paste0("Mvl_",str_sub(type,1,4),"_spr"),mvl_spr,envir=globalenv())
}



KR_frn = function(type){

  # KR_frn() takes portfolio type as an input. The function finds floating rate
  # note data frame of the respective portfolio type (from 15 data frames mentioned above).
  # The function then computes and assigns key rate duration matrix as a global variable.
  # The function returns NULL if source data frame is empty or has a missing value.
  # The function also assigns a data frame with market value of each position to the
  # global environment.

  data = get(paste0(str_sub(type,1,4),"_frn"))
  if (all(is.na(data)) == FALSE & nrow(data)>0){
    KRM_frn = c()
    pos_frn = as.numeric(data[,2])
    val_frn = bdp(data[,1],"PX_MID")[,1]
    mvl_frn = as.data.frame(pos_frn*val_frn*10)
    colnames(mvl_frn) = "Market Value"

    KRM_frn = cbind(KRM_frn,bdp(data[,1], "DUR_ADJ_OAS_BID")[,1])
    KRM_frn = cbind(KRM_frn, matrix(0,nrow(KRM_frn),length(tenor)-1))
    KRM_frn = as.data.frame(KRM_frn)
    colnames(KRM_frn) = tenor
  }else {
    KRM_frn = NULL
    mvl_frn = NULL
  }
  assign(paste0("KRM_",str_sub(type,1,4),"_frn"),KRM_frn,envir=globalenv())
  assign(paste0("Mvl_",str_sub(type,1,4),"_frn"),mvl_frn,envir=globalenv())
}



KR_spo = function(type){

  # KR_spo() takes portfolio type as an input. The function finds cash data
  # frame of the respective portfolio type (from 15 data frames mentioned above).
  # The function then computes and assigns key rate duration matrix as a global variable.
  # The function returns NULL if source data frame is empty or has a missing value.
  # The function also assigns a data frame with market value of each position to the
  # global environment.

  data = get(paste0(str_sub(type,1,4),"_spo"))
  if (all(is.na(data)) == FALSE & nrow(data)>0){
    pos_spo = as.numeric(data[,2])
    val_spo = bdp(data[,1],"PX_MID")[,1]
    val_spo[is.na(val_spo)] <- 0
```

```
    val_spo[grepl("JPY Curncy",data$Sec)] = 1/ val_spo[grepl("JPY Curncy",data$Sec)]
    mvl_spo = as.data.frame(val_spo*pos_spo*1000)
    colnames(mvl_spo) = "Market Value"

    KRM_spo = as.data.frame(matrix(0,nrow(data),length(tenor)))
    colnames(KRM_spo) = tenor
  } else {
    KRM_spo = NULL
    mvl_spo = NULL
  }
  assign(paste0("KRM_",str_sub(type,1,4),"_spo"),KRM_spo,envir=globalenv())
  assign(paste0("Mvl_",str_sub(type,1,4),"_spo"),mvl_spo,envir=globalenv())
}

Merger = function(type){

  # Merger() merges different categories together. The function takes
  # type as an input and assigns 3 merged datasets to the global environment.
  # Datasets include basic portfolio data, key rate duration matrix,
  # and position sizes.

  data = rbind.data.frame(get(paste0(str_sub(type,1,4),"_der")),
                          get(paste0(str_sub(type,1,4),"_gov")),
                          get(paste0(str_sub(type,1,4),"_spr")),
                          get(paste0(str_sub(type,1,4),"_frn")),
                          get(paste0(str_sub(type,1,4),"_spo")))

  KRM = rbind.data.frame(get(paste0("KRM_",str_sub(type,1,4),"_der")),
                         get(paste0("KRM_",str_sub(type,1,4),"_gov")),
                         get(paste0("KRM_",str_sub(type,1,4),"_spr")),
                         get(paste0("KRM_",str_sub(type,1,4),"_frn")),
                         get(paste0("KRM_",str_sub(type,1,4),"_spo")))

  Mvl = rbind.data.frame(get(paste0("Mvl_",str_sub(type,1,4),"_der")),
                         get(paste0("Mvl_",str_sub(type,1,4),"_gov")),
                         get(paste0("Mvl_",str_sub(type,1,4),"_spr")),
                         get(paste0("Mvl_",str_sub(type,1,4),"_frn")),
                         get(paste0("Mvl_",str_sub(type,1,4),"_spo")))
  Ddm = KRM*Mvl$`Market Value`/10000
  Dd = colSums(Ddm)
  assign(paste0("Dta_",str_sub(type,1,4)),data,envir=globalenv())
  assign(paste0("KRM_",str_sub(type,1,4)),KRM,envir=globalenv())
  assign(paste0("Mvl_",str_sub(type,1,4)),Mvl,envir=globalenv())
  assign(paste0("Ddm_",str_sub(type,1,4)),Ddm,envir=globalenv())
  assign(paste0("Dd_",str_sub(type,1,4)),Dd,envir=globalenv())

  PAD <- bdp(data$Sec,"PX_DIRTY_ASK")
  PBD <- bdp(data$Sec,"PX_DIRTY_BID")
  PMD <- bdp(data$Sec,"PX_DIRTY_MID")
  PAC <- bdp(data$Sec,"PX_ASK")
  PBC <- bdp(data$Sec,"PX_BID")
  PMC <- bdp(data$Sec,"PX_MID")
  MT  <- bdp(data$Sec,"DAYS_TO_MTY")
```

```
    DUR <- bdp(data$Sec,"DUR_MID")



    dataset = cbind.data.frame(data$Sec,MT,DUR,data$Pos,Mvl$`Market Value`,KRM,
                               PAD$PX_DIRTY_ASK,PBD$PX_DIRTY_BID,PMD$PX_DIRTY_MID,
                               PAC$PX_ASK,PBC$PX_BID,PMC$PX_MID)
    colnames(dataset) = c("N","MT","dur","Pos","MV","3M","6M","1Y","2Y","3Y","4Y",
                          "5Y","ASK_DRT","BID_DRT","MID_DRT","ASK_CLN","BID_CLN",
                          "MID_CLN")
    assign(paste0(str_sub(type,1,4),"_dataset"),dataset,envir=globalenv())

  }



  # Below are function calls in order. First, parameters are being set.
  # Second, Splitter() functions create 3 highest level data frames,
  # which contain high level data on each of the 3 portfolios.
  # Third, splitter() functions split each of the 3 high level data frames
  # into 5 subcategories based on instrument type. Fourth, key rate matrices
  # are generated for each of the subcategory.

  set_param()
  Splitter(type)
  KR_der(type)
  KR_gov(type)
  KR_spr(type)
  KR_frn(type)
  KR_spo(type)
  Merger(type)

}


Data_gen("Portfolio")
Data_gen("Benchmark")
Data_gen("Projected Benchmark")


Portdata = Port_dataset
Pbenchdata = Proj_dataset
benchdata = Benc_dataset

Portdt = Portdata
Portdata = Portdata[!is.na(Portdata["MT"]),]



cash = as.numeric(as.character(Portdata["USD Curncy","MV"]))
bp = colSums(Pbenchdata[,6:12]*as.numeric(as.character(Pbenchdata[,5]))/10000, na.rm = TRUE)
b = colSums(benchdata[,6:12]*as.numeric(as.character(benchdata[,5]))/10000, na.rm = TRUE)
p = colSums(Portdt[,6:12]*as.numeric(as.character(Portdt[,5]))/10000, na.rm = TRUE)
```

```
target = bp
exposure = bp-b
MV = sum(benchdata$MV)
dtarget = 7
cash = as.numeric(as.character(Portdata["USD Curncy","MV"]))


CR = function(y){
  dataS <- bsrch("FI:UST")
  dataS <- dataS$id
  x = as.numeric(as.Date(y))

  adjmat <- bdp(as.character(dataS),"DAYS_TO_MTY_TDY")/365
  adjiss <- (x-as.numeric(bdp(as.character(dataS),"ISSUE_DT")$ISSUE_DT))/365
  start= 0.5
  end=10
  old = 30
  new = 0

  adjm <- matrix(0,ncol=1, nrow = nrow(adjmat))
  adjm[adjmat>start & adjmat<end]=1

  adji <- matrix(0,ncol=1, nrow = length(adjiss))
  adji[adjiss>new & adjiss<old]=1

  dateCF <- matrix(0, ncol= 60, nrow = length(dataS))
  CF <- matrix(0, ncol= 60, nrow = length(dataS))

  for (i in 1:length(dataS)){
    print(i)
    data <- bds(as.character(dataS[i]),"DES_CASH_FLOW")
    date <- data$`Payment Date`
    dateCF[i,1:length(date)] <- date

    coupon <- data$`Coupon Amount`
    face <- data$`Principal Amount`
    pay <- coupon +face
    CF[i,1:length(pay)] <- pay
  }


  dataS <- dataS[adjm==1 & adji==1]
  CF <- CF[adjm==1 & adji==1,]
  dateCF <- dateCF[adjm==1 & adji==1,]




  CF = CF/10000

  b0 = 0.1
  b1 = 0.1
```

```
b2 = 0.1
l = 1
dt=0.5


dateTCF = dateCF
for (i in 1:length(dataS)){
  dateTCF[i,]=(dateCF[i,]-x)/365
}
dateTCF[dateTCF<0] = 0


NS <- function (b0,b1,b2,l,dt){
  r <- b0+b1*(1-exp(-dt/l))/(dt/l)+b1*((1-exp(-dt/l))/(dt/l)-exp(-dt/l))
  r
}
PQD <- bdp(as.character(dataS),"PX_DIRTY_MID")
MY <- bdp(as.character(dataS),"YLD_YTM_MID")
MY <- MY$YLD_YTM_MID/100

#####
setd <- bdp(as.character(dataS),"SETTLE_DT")
setd <- setd$SETTLE_DT
matd <- bdp(as.character(dataS),"MATURITY")
matd <- matd$MATURITY
cpnd <- bdp(as.character(dataS),"CPN")
cpnd <- cpnd$CPN/100
freqd <- bdp(as.character(dataS),"CPN_FREQ")
freqd <- freqd$CPN_FREQ

#####

#####
cpn = cpnd * 100
pay = cpn/freqd
nextcpn = bdp(as.character(dataS),"NXT_CPN_DT")
nextcpn =as.numeric(nextcpn$NXT_CPN_DT)
prevcpn = bdp(as.character(dataS),"PREV_CPN_DT")
prevcpn =as.numeric(prevcpn$PREV_CPN_DT)
acc = pay*(x-prevcpn)/(nextcpn-prevcpn)

#####
mat <- bdp(as.character(dataS),"DAYS_TO_MTY_TDY")/365
mat <- mat$DAYS_TO_MTY_TDY
dur <- bdp(as.character(dataS),"DUR_MID")
dur = dur$DUR_MID
wd <- (1/dur)/(1/sum(1/dur))
yld <- bdp(as.character(dataS),"YLD_YTM_MID")/100
yld <- yld$YLD_YTM_MID
######


m <- function(c){
```

```
  discr <- c[1]+c[2]*(1-exp(-dateTCF/c[4]))/(dateTCF/c[4])+c[3]*
    ((1-exp(-dateTCF/c[4]))/(dateTCF/c[4])-exp(-dateTCF/c[4]))+
    c[5]*((1-exp(-dateTCF/c[6]))/(dateTCF/c[6])-exp(-dateTCF/c[6]))
  discf <- (1+discr/freqd)^(-freqd*dateTCF)
  discf[is.nan(discr)] = NaN
  CFNSD <- CF*discf
  CFNSD[is.nan(CFNSD)] = 0
  PNSD <- rowSums(CFNSD)
  PNSC <- PNSD - acc
  NSY <- bond.yields(setd, matd,cpnd,freqd, PNSC,comp.freq = freqd)

  dataP <- cbind(MY,NSY)
  SE <- ((dataP[,1]-dataP[,2])^2)*wd
  RMSE <- sum(SE)/nrow(dataP)
  print(1)
  RMSE
}
solc <-Nelder_Mead(m, c(0.1,0.1,-0.1,4,0.1,6))
c = solc$par




NSS <- function(x){
  c[1]+c[2]*(1-exp(-x/c[4]))/(x/c[4])+c[3]*
    ((1-exp(-x/c[4]))/(x/c[4])-exp(-x/c[4]))+
    c[5]*((1-exp(-x/c[6]))/(x/c[6])-exp(-x/c[6]))
}




kNSS <- function(x,y){
  (((1+NSS(x+y))^(x+y))/((1+NSS(y))^y))^(1/(x))-1
}




discrNSS <- c[1]+c[2]*(1-exp(-dateTCF/c[4]))/(dateTCF/c[4])+c[3]*
  ((1-exp(-dateTCF/c[4]))/(dateTCF/c[4])-exp(-dateTCF/c[4]))+
  c[5]*((1-exp(-dateTCF/c[6]))/(dateTCF/c[6])-exp(-dateTCF/c[6]))
discfNSS <- (1+discrNSS/freqd)^(-freqd*dateTCF)

CFNSD <- CF*discfNSS
CFNSD[is.nan(CFNSD)] = 0
PNSD <- rowSums(CFNSD)
PNSC <- PNSD - acc
NSY <- bond.yields(setd, matd,cpnd,freqd, PNSC,comp.freq = freqd)
dataP <- cbind(MY,NSY)
CR <- (dataP[,1]-dataP[,2])
RV <- cbind(mat, CR)
```

```
  #####

  name = bdp(as.character(dataS),"SECURITY_NAME")
  name = name$SECURITY_NAME
  CUSIP = bdp(as.character(dataS),"ID_CUSIP")
  CUSIP = CUSIP$ID_CUSIP

  num = cbind(name, CUSIP,
              format(round(CR*10000, 2), nsmall = 2)
  )
  colnames(num) = c("Security","ISIN","Z-Score")
  num = as.data.frame(num)
  num
}
```

```
dataPZ = CR(Sys.Date())
dataZ = dataPZ[,2:3]
dataZ[,1] = paste0(substr(dataZ[,1],1,8),"     Govt")
colnames(dataZ) = c("N","Z")

Portdt$N = as.character(Portdt$N)
datap = inner_join(Portdt,dataZ, by = "N")


Pbenchdata$N = as.character(Pbenchdata$N)
datapb = inner_join(Pbenchdata,dataZ, by = "N")


benchdata$N = as.character(benchdata$N)
datab = inner_join(benchdata,dataZ, by = "N")


bkt1 = c(0,0.5)
bkt2 = c(0.5,0.9)
bkt3 = c(0.9,1.4)
bkt4 = c(1.5,2.5)
bkt5 = c(2.5,3.6)



sec1B = Pbenchdata[Pbenchdata["dur"]>bkt1[1] & Pbenchdata["dur"]<bkt1[2],]
sec1B = sec1B[!is.na(sec1B[,1]),]
sec1B = sec1B[grepl(min(as.numeric(as.character(sec1B$MID_CLN))),
                    as.numeric(as.character(sec1B$MID_CLN))),"N"]
```

```
PA1B = Pbenchdata[grep(sec1B,Pbenchdata[["N"]]),"ASK_DRT"]
PB1B = Pbenchdata[grep(sec1B,Pbenchdata[["N"]]),"BID_DRT"]
M1B = Pbenchdata[grep(sec1B,Pbenchdata[["N"]]),c(5)]
K1B = Pbenchdata[grep(sec1B,Pbenchdata[["N"]]),c(6:10)]

sec1S = Portdata[Portdata["dur"]>bkt1[1] & Portdata["dur"]<bkt1[2],]
sec1S = sec1S[!is.na(sec1S[,1]),]
sec1S = sec1S[grepl(max(as.numeric(as.character(sec1S$MID_CLN))),
                    as.numeric(as.character(sec1S$MID_CLN))),"N"]
PA1S = Portdata[grep(sec1S,Portdata[["N"]]),"ASK_DRT"]
PB1S = Portdata[grep(sec1S,Portdata[["N"]]),"BID_DRT"]
M1S = Portdata[grep(sec1S,Portdata[["N"]]),c(5)]
K1S = Portdata[grep(sec1S,Portdata[["N"]]),c(6:10)]

sec2B = datapb[datapb["dur"]>bkt2[1] & datapb["dur"]<bkt2[2],]
sec2B = sec2B[grepl(max(as.numeric(as.character(sec2B$Z))),
                    as.numeric(as.character(sec2B$Z))),"N"]
PA2B = datapb[grep(sec2B,datapb[["N"]]),"ASK_DRT"]
PB2B = datapb[grep(sec2B,datapb[["N"]]),"BID_DRT"]
M2B = datapb[grep(sec2B,datapb[["N"]]),c(5)]
K2B = datapb[grep(sec2B,datapb[["N"]]),c(6:10)]

sec2S = datap[datap["dur"]>bkt2[1] & datap["dur"]<bkt2[2],]
sec2S = sec2S[grepl(min(as.numeric(as.character(sec2S$Z))),
                    as.numeric(as.character(sec2S$Z))),"N"]
PA2S = datap[grep(sec2S,datap[["N"]]),"ASK_DRT"]
PB2S = datap[grep(sec2S,datap[["N"]]),"BID_DRT"]
M2S = datap[grep(sec2S,datap[["N"]]),c(5)]
K2S = datap[grep(sec2S,datap[["N"]]),c(6:10)]

sec3B = datapb[datapb["dur"]>bkt3[1] & datapb["dur"]<bkt3[2],]
sec3B = sec3B[grepl(max(as.numeric(as.character(sec3B$Z))),
                    as.numeric(as.character(sec3B$Z))),"N"]
PA3B = datapb[grep(sec3B,datapb[["N"]]),"ASK_DRT"]
PB3B = datapb[grep(sec3B,datapb[["N"]]),"BID_DRT"]
M3B = datapb[grep(sec3B,datapb[["N"]]),c(5)]
K3B = datapb[grep(sec3B,datapb[["N"]]),c(6:10)]

sec3S = datap[datap["dur"]>bkt3[1] & datap["dur"]<bkt3[2],]
sec3S = sec3S[grepl(min(as.numeric(as.character(sec3S$Z))),
                    as.numeric(as.character(sec3S$Z))),"N"]
PA3S = datap[grep(sec3S,datap[["N"]]),"ASK_DRT"]
PB3S = datap[grep(sec3S,datap[["N"]]),"BID_DRT"]
M3S = datap[grep(sec3S,datap[["N"]]),c(5)]
K3S = datap[grep(sec3S,datap[["N"]]),c(6:10)]


sec4B = datapb[datapb["dur"]>bkt4[1] & datapb["dur"]<bkt4[2],]
sec4B = sec4B[grepl(max(as.numeric(as.character(sec4B$Z))),
                    as.numeric(as.character(sec4B$Z))),"N"]
PA4B = datapb[grep(sec4B,datapb[["N"]]),"ASK_DRT"]
PB4B = datapb[grep(sec4B,datapb[["N"]]),"BID_DRT"]
M4B = datapb[grep(sec4B,datapb[["N"]]),c(5)]
```

```
K4B = datapb[grep(sec4B,datapb[["N"]]),c(6:10)]

sec4S = datap[datap["dur"]>bkt4[1] & datap["dur"]<bkt4[2],]
sec4S = sec4S[grepl(min(as.numeric(as.character(sec4S$Z))),
                    as.numeric(as.character(sec4S$Z))),"N"]
PA4S = datap[grep(sec4S,datap[["N"]]),"ASK_DRT"]
PB4S = datap[grep(sec4S,datap[["N"]]),"BID_DRT"]
M4S = datap[grep(sec4S,datap[["N"]]),c(5)]
K4S = datap[grep(sec4S,datap[["N"]]),c(6:10)]

sec5B = datapb[datapb["dur"]>bkt5[1] & datapb["dur"]<bkt5[2],]
sec5B = sec5B[grepl(max(as.numeric(as.character(sec5B$Z)))*100,
                    as.numeric(as.character(sec5B$Z))*100),"N"]
sec5B = sec5B[1]
PA5B = datapb[grep(sec5B,datapb[["N"]]),"ASK_DRT"]
PB5B = datapb[grep(sec5B,datapb[["N"]]),"BID_DRT"]
M5B = datapb[grep(sec5B,datapb[["N"]]),c(5)]
K5B = datapb[grep(sec5B,datapb[["N"]]),c(6:10)]

sec5S = datap[datap["dur"]>bkt5[1] & datap["dur"]<bkt5[2],]
sec5S = sec5S[grepl(min(as.numeric(as.character(sec5S$Z))),
                    as.numeric(as.character(sec5S$Z))),"N"]
PA5S = datap[grep(sec5S,datap[["N"]]),"ASK_DRT"]
PB5S = datap[grep(sec5S,datap[["N"]]),"BID_DRT"]
M5S = datap[grep(sec5S,datap[["N"]]),c(5)]
K5S = datap[grep(sec5S,datap[["N"]]),c(6:10)]




optimize <- function(w){


  cost1 <<- w[1]*if(w[1]>0){PA1B}else{PB1S}
  cost2 <<- w[2]*if(w[2]>0){PA2B}else{PB2S}
  cost3 <<- w[3]*if(w[3]>0){PA3B}else{PB3S}
  cost4 <<- w[4]*if(w[4]>0){PA4B}else{PB4S}
  cost5 <<- w[5]*if(w[5]>0){PA5B}else{PB5S}


  MKT1 <<- w[1]*if(w[1]>0){PB1B}else{PB1S}
  MKT2 <<- w[2]*if(w[2]>0){PB2B}else{PB2S}
  MKT3 <<- w[3]*if(w[3]>0){PB3B}else{PB3S}
  MKT4 <<- w[4]*if(w[4]>0){PB4B}else{PB4S}
  MKT5 <<- w[5]*if(w[5]>0){PB5B}else{PB5S}



  KRT1 <<- MKT1*if(w[1]>0){K1B/10000}else{K1S/10000}
  KRT2 <<- MKT2*if(w[2]>0){K2B/10000}else{K2S/10000}
  KRT3 <<- MKT3*if(w[3]>0){K3B/10000}else{K3S/10000}
  KRT4 <<- MKT4*if(w[4]>0){K4B/10000}else{K4S/10000}
```

```
  KRT5 <<- MKT5*if(w[5]>0){K5B/10000}else{K5S/10000}


  err = -exposure[1:5]+KRT1+KRT2+KRT3+KRT4+KRT5
  dur <<- (KRT1+KRT2+KRT3+KRT4+KRT4)*10000*365/MV
  err1 = err[1]
  err2 = err[2]
  err3 = err[3]
  err4 = err[4]
  err5 = err[5]
  err6 = (cash -cost1-cost2-cost3-cost4-cost5)/5000
  err7 = dtarget - dur
  trade = abs(MKT1)+abs(MKT2)+abs(MKT3)+abs(MKT4)+abs(MKT5)
  positions = c(w[1],w[2],w[3],w[4],w[5])

  RMSE <- ((err1^2+err2^2+err3^2+err4^2+err5^2+err6^2)/6)^(0.5)

  error = cbind.data.frame(err1,err2,err3,err4,err5,err6)
  print(positions)
  print(trade)
  print(RMSE[[1]])
  print(error)
  RMSE[[1]]

}
optim = Nelder_Mead(optimize, c(1,1,1,1,1))


ex_port = as.data.frame(p)
colnames(ex_port) = "exposure"
ex_port = data.frame(tenor = rownames(ex_port),
                     exposure = ex_port$exposure,type = "Portfolio")
ex_bench = as.data.frame(b)
colnames(ex_bench) = "exposure"
ex_bench = data.frame(tenor = rownames(ex_bench),
                      exposure = ex_bench$exposure,type = "Benchmark")
ex_current = rbind(ex_bench,ex_port)
ex_current$tenor <- factor(ex_current$tenor,
                           levels = unique(ex_current$tenor))
ex_net = data.frame(net = c(ex_current[ex_current$type=="Portfolio",2]-
                              ex_current[ex_current$type=="Benchmark",2],
                            ex_current[ex_current$type=="Portfolio",2]-
                              ex_current[ex_current$type=="Benchmark",2]))
ex_current = cbind.data.frame(ex_current,ex_net)

fig1 = ggplot(data=ex_current, aes(x=tenor,fill = type)) +
  geom_bar(aes(y=exposure),stat="identity", position=position_dodge(),
           color = "black")+
  theme_minimal()+
  geom_boxplot(aes(y = net*5,colour = "Net Exposure"))+
  scale_colour_manual(name = "",values = c("black","green"))+
  scale_fill_manual(name = "",values = c("#41b6c4","#fb6a4a"))+
  labs(y = "Total Exposure (Dollar Duration)",
```

```
      x = "Tenor",title="Figure 1. Current Exposures (Dollar Duration)")+
  ylim(-60000,150000)+
  scale_y_continuous(sec.axis =
                       sec_axis(~./5,
                                name = "Net Exposure (Dollar Duration)"))




ex_port = as.data.frame(p)
colnames(ex_port) = "exposure"
ex_port = data.frame(tenor = rownames(ex_port),
                     exposure = ex_port$exposure,type = "Portfolio")
ex_proj = as.data.frame(bp)
colnames(ex_proj) = "exposure"
ex_proj = data.frame(tenor = rownames(ex_proj),
                     exposure = ex_proj$exposure,type = "New Benchmark")
ex_new = rbind(ex_proj,ex_port)
ex_new$tenor <- factor(ex_new$tenor,levels = unique(ex_new$tenor))
ex_net_new = data.frame(net_new = c(ex_new[ex_new$type=="Portfolio",2]-
                                      ex_new[ex_new$type=="New Benchmark",2],
                                    ex_new[ex_new$type=="Portfolio",2]-
                                      ex_new[ex_new$type=="New Benchmark",2]))
ex_new = cbind.data.frame(ex_new,ex_net_new,ex_net)

fig2 = ggplot(data=ex_new, aes(x=tenor,fill = type)) +
  geom_bar(aes(y=exposure),stat="identity", position=position_dodge(),
           color = "black")+
  theme_minimal()+
  geom_boxplot(aes(y = net*5,colour = "Net Exposure Old"))+
  geom_boxplot(aes(y = net_new*5,colour = "Net Exposure New"))+
  scale_colour_manual(name = "",values = c("black","green"))+
  scale_fill_manual(name = "",values = c("#74c476","#fb6a4a"))+
  labs(y = "Total Exposure (Dollar Duration)",
       x = "Tenor",title="Figure 2. Projected Exposures (Dollar Duration)")+
  ylim(-60000,150000)+
  scale_y_continuous(sec.axis =
                       sec_axis(~./5,
                                name = "Net Exposure (Dollar Duration)"))


grid.arrange(fig1, fig2, nrow = 2)

ex_trade = as.data.frame(bp - b)
colnames(ex_trade) = "exposure"
ex_trade = data.frame(tenor = rownames(ex_trade),
                      exposure = ex_trade$exposure)
ex_trade$tenor <- factor(ex_trade$tenor,levels = ex_trade$tenor)
fig3 = ggplot(data=ex_trade, aes(x=tenor)) +
  geom_bar(aes(y=exposure,fill = "Trade Exposure"),
           stat="identity",  color = "black")+
  ylim(c(-60000,150000)/10)+
  labs(y = "Trade Exposure (Dollar Duration)",
```

```
       x = "Tenor",title="Figure 3. Trade Exposures (Dollar Duration)")+
  scale_fill_manual(name = "",values = c("#dd3497"))


p_der = colSums(KRM_Port_der*Mvl_Port_der$`Market Value`)/10000
p_der = as.data.frame(p_der)
colnames(p_der) = "exposure"
p_der = data.frame(tenor = rownames(p_der),
                   exposure = p_der$exposure, type = "Derivative")

p_gov = colSums(KRM_Port_gov*Mvl_Port_gov$`Market Value`)/10000
p_gov = as.data.frame(p_gov)
colnames(p_gov) = "exposure"
p_gov = data.frame(tenor = rownames(p_gov),
                   exposure = p_gov$exposure, type = "Government")

p_spr = colSums(KRM_Port_spr*Mvl_Port_spr$`Market Value`)/10000
p_spr = as.data.frame(p_spr)
colnames(p_spr) = "exposure"
p_spr = data.frame(tenor = rownames(p_spr),
                   exposure = p_spr$exposure, type = "Spread")

p_frn = colSums(KRM_Port_frn*Mvl_Port_frn$`Market Value`)/10000
p_frn = as.data.frame(p_frn)
colnames(p_frn) = "exposure"
p_frn = data.frame(tenor = rownames(p_frn),
                   exposure = p_frn$exposure, type = "Floater")

ex_port_sub = rbind(p_der,p_gov,p_spr,p_frn)
ex_port_sub$tenor <- factor(ex_port_sub$tenor
                            ,levels = unique(ex_port_sub$tenor))

fig4 = ggplot(data=ex_port_sub, aes(x=tenor,
                                    y=exposure,fill = type)) +
  geom_bar(stat="identity", color = "black")+
  theme_minimal()+
  ylim(-60000,150000)+
  scale_fill_manual(name = "",
                    values = c("#41ab5d","#ef3b2c","#1d91c0","#8c6bb1"))+
  labs(y = "Total Exposure (Dollar Duration)",
       x = "Tenor",title="Figure 4. Instrument Exposures (Dollar Duration)")

grid.arrange(fig3, fig4, nrow = 2)

buysell = c()
for (i in optim$par){
  if (i>0){buysell = c(buysell,"Buy")
  }else{buysell = c(buysell,"Sell")}
}
ISIN = c()
for (i in 1:length(optim$par)){
  if (optim$par[i]>=0){ISIN = c(ISIN,get(paste0("sec",i,"B")))
  }else {ISIN = c(ISIN,get(paste0("sec",i,"S")))}
```

```
}
pask = c()
pbid = c()
for (i in 1:length(optim$par)){
  if (optim$par[i]>=0){
    pask = c(pask,get(paste0("PA",i,"B")))
    pbid = c(pbid,get(paste0("PB",i,"B")))
  }else{
    pask = c(pask,get(paste0("PA",i,"S")))
    pbid = c(pbid,get(paste0("PB",i,"S")))
  }
}
name = bdp(c(sec1S,sec2B,sec3B,sec4B,sec5B),"SECURITY_NAME")$SECURITY_NAME
trade = data.frame(Security= name,
                   ISIN = c(sec1S,sec2B,sec3B,sec4B,sec5B),
                   Sign = buysell,
                   position = format_dollars(abs(optim$par*100)),
                   Bid = pbid,
                   Ask = pask)
grid.table(trade)



ex_port = as.data.frame(p)
colnames(ex_port) = "exposure"
ex_port = data.frame(tenor = rownames(ex_port),
                     exposure = ex_port$exposure)
ex_port$tenor <- factor(ex_port$tenor,levels = ex_port$tenor)
ggplot(data=ex_port, aes(x=tenor, y=exposure)) +
  geom_bar(stat="identity", fill = "#74c476", color = "black")+
  theme_minimal()



ex_bench = as.data.frame(b)
colnames(ex_bench) = "exposure"
ex_bench = data.frame(tenor = rownames(ex_bench),
                      exposure = ex_bench$exposure)
ex_bench$tenor <- factor(ex_bench$tenor,
                         levels = ex_bench$tenor)
ggplot(data=ex_bench, aes(x=tenor, y=exposure)) +
  geom_bar(stat="identity", fill = "#74c476", color = "black")+
  theme_minimal()
```