SIMULACION HISTORICA

EQUIPO ALEJANDRO,DANIEL,ANTONIO,SOFIA,LUIS

21/4/2022

"TAREA SIMULACION HISTORICA" 22/03/2022

Abstract

SE AJUSTAN BASES DE DATOS DE DIVERSAS CURVAS DE RENDIMIENTO, SE AJUSTAN DATOS DE YAHOO FINANCE Y SE ALINEAN PARA GENERAR UNA SIMULACIÓN HISTORICA DE ESCENARIOS UTILIZANDO YA SEA SIMULACIÓN HISTORICA CON ALISADO O SIN ALISADO DE LOS INSTRUMENTOS QUE SE DETALLAN EN LA TAREA 1

Palabras clave: ejercicios, tarea.

Contents

CARGA DE BASES DE DATOS

```r
#require(quantmod)
#install.packages("quantmod")
library(quantmod)
```

```
## Loading required package: xts

## Warning: package 'xts' was built under R version 3.6.3

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.6.3

##
## Attaching package: 'zoo'

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

## Loading required package: TTR

## Warning: package 'TTR' was built under R version 3.6.3

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

```r
#require(data.table)
#install.packages("data.table")
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.6.3

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:xts':
##
##     first, last
```

```r
#require("PerformanceAnalytics")
#install.packages("PerformanceAnalytics")
library("PerformanceAnalytics")
```

```
## Warning: package 'PerformanceAnalytics' was built under R version 3.6.3

##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

```r
#install.packages("Deriv")
library(Deriv)
```

```
## Warning: package 'Deriv' was built under R version 3.6.3
```

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:xts':
##
##     first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
con = gzcon(url('https://github.com/systematicinvestor/SIT/raw/master/sit.gz', 'rb'))
source(con)
close(con)
```

```
talamb=function(nodos,curva,plazos) #función de interpolación de tasas por el método alamabrada
{
  n=max(ncol(plazos),1)
  m=max(ncol(nodos),1)
  TC=matrix(0,1,n)
  TL=matrix(0,1,n)
  TF=matrix(0,1,n)
  for (j in 1:n)
  {
    i=1
    repeat
    {
      if(nodos[i]<= plazos[j] && plazos[j] <=nodos[i+1])
      {
        TC[j]=curva[i]
        TL[j]=curva[i+1]
        TF[j]=((((1+TL[j]*nodos[i+1]/360)/(1+TC[j]*nodos[i]/360))^((plazos[j]-nodos[i])/(nodos[i+1]-nod
        break
      }
      else if (plazos[j]<nodos[1])
      {
        TC[j]=curva[1]
        TL[j]=curva[1]
        TF[j]=curva[1]
        break
      }
      else if (plazos[j]>nodos[m])
      {
        TC[j]=curva[m]
        TL[j]=curva[m]
        TF[j]=curva[m]
        break
      }
      else
      {i=i+1}
    }
  }
  as.matrix(t(as.numeric(rbind(TF))))
}


#funciones necesarias
diagv=function(x)          #función para diagonalizar un vector
{
  n01=nrow(as.matrix(x))
  m01=ncol(as.matrix(x))
```

```r
  dimmax=max(n01,m01)
  res=matrix(0,dimmax,dimmax)
  for (i in 1:dimmax)
  {
    res[i,i]=x[i]
  }
  res
}

#función de cuantil más cercano
equantile <- function(v,p=.5,ns=nrow(as.matrix(v)))
{
  if ( !is.numeric(p) || any( p<0 | p>1) )
    stop("Percentil tiene que ser 0<=p<=1")
  ranking <- order(v)
  vw=matrix(0,ns,1)
  vw[1:ns]=seq(1/ns,ns)
  sumw <- cumsum(vw[ranking])
  plist <- sumw / sumw[ length(sumw) ]
  v [ ranking [ which.max( plist >= p ) ] ]
}


wquantile <- function(v,w=rep(1,length(v)),p=.5)
{
  if ( !is.numeric(w) || length(v) != length(w) )
    stop("Los valores y los pesos tienen que tener misma longitud")
  if ( !is.numeric(p) || any( p<0 | p>1) )
    stop("Percentil tiene que ser 0<=p<=1")
  if ( min(w) < 0 ) stop("Los pesos tiene que ser mayores que 0")
  ranking <- order(v)
  sumw <- cumsum(w[ranking])
  plist <- sumw / sumw[ length(sumw) ]
  v [ ranking [ which.max( plist >= p ) ] ]
}


#CVaR con alisado
wcvar <- function(v,w=rep(1,length(v)),p=.5)
{
  if ( !is.numeric(w) || length(v) != length(w) )
    stop("Los valores y los pesos tienen que tener misma longitud")
  if ( !is.numeric(p) || any( p<0 | p>1) )
    stop("Percentil tiene que ser 0<=p<=1")
  if ( min(w) < 0 ) stop("Los pesos tiene que ser mayores que 0")
  ranking <- order(v)
  sumw <- cumsum(w[ranking])
  plist <- sumw / sumw[ length(sumw) ]
  loss= v [ ranking [ which( plist < p ) ] ]
  esc=w [ ranking [ which( plist < p ) ] ]
  sum(loss*esc)/(sum(esc))
}
```

```r
fval=as.Date("20220331",format="%Y%m%d") #Fecha de valoración
itpl=0 #poner 0 si se quiere interpolación lineal o 1 si se quiere tasa alambrada
alpha=0.98 #Nivel de confianza para obtener estimaciones de riesgo
#setwd(direc)

#ACCIONES Y DIVISAS
#Cargar los símbolos de yahoo finance para EQ
Symbols<-c("AMXL.MX","GCARSOA1.MX", "WMT.MX")#tienen que ir en orden alfabético
pos_eq=c(-5000,1000,1200) #monto inicial invertido en acciones
#Cargar los símbolos de yahoo finance para FX
SymbolsFX<-c("EURUSD=X", "GBPUSD=X","USDMXN=X") #tienen que ir en orden alfabético
pos_fx=c(700,-600, 1500) #monto inicial invertido en divisas
nh=3660 #días de historia

#CETES
base="RiesgosFinancieros/2020-2/Insumos/tasa_guber.txt"

#BONDES D
btasadescst="RiesgosFinancieros/2022-2/Insumos/tasa_guber_st.txt"
btasafondeo="RiesgosFinancieros/2022-2/Insumos/tfondeo.txt"
plazos_bdm=cbind( 3600,707) #Vencimiento del bono
plazocupon_bdm=cbind(28,28) #plazos_bdm fijos de cada cupón
contratos_bdm=cbind(1,1) #posición invertida
nominal_bdm=1000

#FORWARDS TDC
bext="RiesgosFinancieros/2020-2/Insumos/tasa_libor.txt"
bdom="RiesgosFinancieros/2020-2/Insumos/tasa_fwd.txt"
SymbolsFX_ftdc<-c("USDMXN=X","GBPUSD=X") #tienen que ir en orden alfabético
plazos_fwd=cbind(5)
contratos_fwd=cbind(100)
kst_fwd=cbind(20.83)
nominal_fwd=1
yext=1 #si se carga información de yahoo en la fecha definida por fval o SymbolsFX, en caso contrario s
trlib=1 #1 si la curva libor viene a 182 0 si no.

#FORWARDS DE IPC
#Descontamos con gubernamental
base="RiesgosFinancieros/2020-2/Insumos/tasa_guber.txt"
SymbolsEQ_find<-c("^MXX", "GCARSOA1.MX") #tienen que ir en orden alfabético
plazos_fwd_ind=cbind(53)
contratos_fwd_ind=cbind(50)
kst_fwd_ind=cbind(49525)
nominal_fwd_ind=1


#SWAP
btasadesc_sw="RiesgosFinancieros/2020-2/Insumos/tasa_TIIE_SW_OP.txt"
btasacupvar_sw="RiesgosFinancieros/2020-2/Insumos/tasa_DIRS_SW_OP.txt"
tasafija_sw=cbind(0.066,.59) #se establece la tasa fija a pagar para cada swap
plazos_sw=cbind(588,270) #se establece el número de días que vivirá el swap
plazocupon_sw=cbind(28,28) #se establece el número de días que se pagará cada cupón
contratos_sw=cbind(1,1) #se establece el número de contratos_sw de cada swap
```

```r
nominal_sw=cbind(16000000,12000000) #se establece el nominal_sw de cada swap
por_sw=cbind(0,1) #se establece 0 si se paga tasa fija y 1 si se paga tasa variable

#OPCIONES
btasadesc_oir="RiesgosFinancieros/2020-2/Insumos/tasa_TIIE_SW_OP.txt"
btasaspot_oir="RiesgosFinancieros/2020-2/Insumos/tasa_DIRS_SW_OP.txt"
bvolspot_oir="RiesgosFinancieros/2020-2/Insumos/tvoltiie_opc.txt"
plazos_oir=cbind( 1700, 700) #T-t
pr_oir=28 #plazo de referencia
dct_oir=360 #d_base
cp_oir=cbind(1,0) #si es call (cap) o put (floor)
K_oir=cbind( 0.058, 0.06)
contratos_oir=cbind(1000, 500)
nominal_oir=1
cs_oir=1 #1 si es continua la tasa 0 si es simple
```

Carga de datos

```r
library("quantmod")
#CARGA DE DATOS DE ACCIONES
pos=cbind(t(pos_fx),t(pos_eq))
start_date=Sys.Date()-nh #fecha inicial
#Creación del objeto para guardar los datos
dataEnv<-new.env()
dataEnvFX<-new.env()
#obtener los datos
?dataEnv
```

```
## No documentation for 'dataEnv' in specified packages and libraries:
## you could try '??dataEnv'
```

```r
getSymbols.yahoo(Symbols,env=dataEnv,from=start_date)
```

```
## [1] "AMXL.MX"     "GCARSOA1.MX" "WMT.MX"
```

```r
getSymbols.yahoo(SymbolsFX,env=dataEnvFX,from=start_date)
```

```
## Warning: EURUSD=X contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

## Warning: GBPUSD=X contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

## Warning: USDMXN=X contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "EURUSD=X" "GBPUSD=X" "USDMXN=X"
```

```
#muestra
#tail(dataEnvFX$`GBPUSD=X`)
#limpiarlos, alinearnos y quedarnos con el precio de cierre
bt.prep(dataEnv,align='remove.na',fill.gaps = T)
bt.prep(dataEnvFX,align='remove.na',fill.gaps=T)
#muestra de datos
#head(dataEnv$prices)
#head(dataEnvFX$prices)
#Nos quedamos con los precio

stock_prices = dataEnv$prices
#tail(stock_prices[,])
stock_pricesFX=dataEnvFX$prices
#tail(stock_pricesFX)
#cambiar todo a pesos mexicanos
head(stock_prices[,1,with = F])
```

```
##                     AMXL.MX
## 2012-04-22 19:00:00   16.47
## 2012-04-23 19:00:00   16.65
## 2012-04-24 19:00:00   16.57
## 2012-04-25 19:00:00   16.61
## 2012-04-26 19:00:00   16.98
## 2012-04-29 19:00:00   17.39
```

```
stock_pricesFX=cbind(stock_pricesFX[,1,with=F]*stock_pricesFX[,3,with=F],stock_pricesFX[,2,with=F]*stoc
#tail(stock_pricesFX)
stock_prices_EQFX=merge(stock_pricesFX,stock_prices,join = "inner")
#stock_prices_EQFX
#tail(stock_prices_EQFX)
#Preciso actuales
#x0=as.data.table(as.matrix(stock_prices_EQFX[nrow(stock_prices_EQFX),])) #valores actuales
x0=stock_prices_EQFX[nrow(stock_prices_EQFX),]
#x0

aux2=data.table(Date=as.Date(index(stock_prices_EQFX)),coredata(stock_prices_EQFX))
```

A continuación leemos los datos de CETES

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.6.3
```

```
library(data.table)
library(dplyr)
setwd("C:/Users/Gis/Documents/SIMULACION HISTORICA")
data = read_table2("tasa_guber.txt")
```

```
##
```

```
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```r
#LEER DATOS DE CETES #CARGA DE DATOS DE BONO CUPÓN CERO
#data<-read.table(base)
n<-nrow(data)
m_gov=ncol(data)
head(data)
```

```
## # A tibble: 6 x 21
##       DATE    `1`    `7`   `30`   `90`  `180`  `270`  `360`  `720` `1080` `1440`
##      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 20200306 0.0791 0.0781 0.0771 0.0790 0.0799 0.0804 0.0806 0.0811 0.0848 0.0880
## 2 20200305 0.0772 0.0781 0.0771 0.0788 0.0801 0.0804 0.0808 0.0809 0.0844 0.0889
## 3 20200304 0.0772 0.0781 0.0774 0.0790 0.0795 0.0802 0.0808 0.0809 0.0850 0.0900
## 4 20200303 0.0781 0.0781 0.0774 0.0790 0.0795 0.0802 0.0808 0.0802 0.0836 0.0874
## 5 20200302 0.0772 0.0781 0.0774 0.0795 0.0804 0.0815 0.0817 0.0814 0.0857 0.0903
## 6 20200228 0.0772 0.0780 0.0774 0.0797 0.0806 0.0818 0.0821 0.0818 0.0861 0.0901
## # ... with 10 more variables: 1800 <dbl>, 2160 <dbl>, 2520 <dbl>, 2880 <dbl>,
## #   3240 <dbl>, 3600 <dbl>, 5400 <dbl>, 7200 <dbl>, 9000 <dbl>, 10800 <dbl>
```

Necesitamos arreglar las fechas para que tengan el formato adecuado, lo cual hacemos tomando la clase Date de R.

```r
fecha = as.Date(as.character(data$DATE[1:n]),format="%Y%m%d")
x_orig_gov=as.data.table(mutate(data[1:n,1:m_gov],Date = fecha))
x_orig_gov=x_orig_gov%>%select(-DATE)
nodos_gov=data.frame(t(as.double(colnames(data)[2:m_gov])))
?data.frame
```

```
## starting httpd help server ... done
```

```r
head(x_orig_gov$DATE)
```

```
## NULL
```

```r
head(x_orig_gov)
```

```
##             1          7         30         90        180        270        360
## 1: 0.07906000 0.07808887 0.07709264 0.07897000 0.07988264 0.08042400 0.08063634
## 2: 0.07722139 0.07808887 0.07709264 0.07878888 0.08006502 0.08042400 0.08082002
## 3: 0.07722139 0.07808887 0.07744709 0.07897000 0.07951788 0.08023954 0.08082002
## 4: 0.07813334 0.07808887 0.07744709 0.07897000 0.07951788 0.08023954 0.08082002
## 5: 0.07722139 0.07808887 0.07744709 0.07951337 0.08042978 0.08150517 0.08174244
## 6: 0.07722139 0.07802004 0.07744709 0.07969450 0.08061216 0.08183675 0.08211584
##           720       1080       1440       1800       2160       2520       2880
## 1: 0.08111017 0.08475054 0.08799784 0.09131343 0.09413113 0.09718213 0.10142256
## 2: 0.08090854 0.08435811 0.08886566 0.09131224 0.09288080 0.09604609 0.10095870
```

```
## 3: 0.08090464 0.08498099 0.08997304 0.09229434 0.09333689 0.09589873 0.10165724
## 4: 0.08018166 0.08355430 0.08739567 0.08964867 0.09152134 0.09369363 0.09934898
## 5: 0.08137725 0.08570076 0.09028780 0.09301676 0.09466370 0.09680468 0.10243388
## 6: 0.08180786 0.08606051 0.09014564 0.09367154 0.09486520 0.09823601 0.10349057
##         3240      3600      5400      7200      9000     10800       Date
## 1: 0.1058224 0.1104565 0.1388027 0.1775079 0.2287571 0.2973447 2020-03-06
## 2: 0.1062402 0.1118325 0.1376561 0.1785716 0.2216318 0.2991903 2020-03-05
## 3: 0.1077483 0.1128102 0.1366114 0.1748460 0.2139762 0.2833846 2020-03-04
## 4: 0.1052630 0.1092318 0.1331730 0.1663083 0.2106910 0.2676971 2020-03-03
## 5: 0.1083077 0.1124022 0.1366660 0.1691053 0.2052162 0.2740801 2020-03-02
## 6: 0.1086765 0.1131309 0.1367676 0.1717620 0.2132537 0.2598649 2020-02-28
```

A continuación hacemos lo mismo con Bondes:

```r
##CARGA DE DATOS DE BONDE D
#carga de datos
#carga de tasas de descuento
library(readr)
data1 <- read_table2("tasa_guber.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```r
n<-nrow(data1)
m_bd=ncol(data1)
##X_orig_bd=as.data.table(mutate(data1[2:n,1:m_tybm],Date=as.Date(V1,format="%Y%m%d")))
#X_orig_bd%>%select(-V1)
```

```r
fecha = as.Date(as.character(data1$DATE[1:n]),format="%Y%m%d")
?as.Date
#x_orig_gov=data.frame(data[2:n,1:m_gov])
X1_orig=as.data.table(mutate(data1[1:n,1:m_gov],Date=fecha))
X1_orig=X1_orig%>%select(-DATE)
#nodos=data.frame(data1[1,2:m_bd])
n=n-1
head(X1_orig)
```

```
##             1          7         30         90        180        270        360
## 1: 0.07845614 0.07749243 0.07650381 0.07836683 0.07927250 0.07980972 0.08002044
## 2: 0.07663158 0.07749243 0.07650381 0.07818709 0.07945348 0.07980972 0.08020272
## 3: 0.07663158 0.07749243 0.07685555 0.07836683 0.07891052 0.07962667 0.08020272
## 4: 0.07753656 0.07749243 0.07685555 0.07836683 0.07891052 0.07962667 0.08020272
## 5: 0.07663158 0.07749243 0.07685555 0.07890605 0.07981546 0.08088263 0.08111809
## 6: 0.07663158 0.07742413 0.07685555 0.07908579 0.07999645 0.08121168 0.08148864
##           720       1080       1440       1800       2160       2520       2880
## 1: 0.08049065 0.08410321 0.08732571 0.09061598 0.09341216 0.09643985 0.10064789
## 2: 0.08029056 0.08371378 0.08818690 0.09061479 0.09217138 0.09531249 0.10018758
## 3: 0.08028669 0.08433190 0.08928583 0.09158939 0.09262398 0.09516626 0.10088078
```

```
## 4: 0.07956923 0.08291612 0.08672814 0.08896394 0.09082230 0.09297800 0.09859015
## 5: 0.08075569 0.08504618 0.08959818 0.09230629 0.09394066 0.09606529 0.10165149
## 6: 0.08118301 0.08540318 0.08945711 0.09295608 0.09414062 0.09748568 0.10270011
##         3240      3600      5400      7200      9000     10800       Date
## 1: 0.1050141 0.1096128 0.1377425 0.1761521 0.2270098 0.2950736 2022-03-31
## 2: 0.1054287 0.1109783 0.1366046 0.1772077 0.2199390 0.2969051 2022-03-30
## 3: 0.1069253 0.1119486 0.1355680 0.1735105 0.2123418 0.2812201 2022-03-29
## 4: 0.1044590 0.1083975 0.1321558 0.1650380 0.2090817 0.2656525 2022-03-28
## 5: 0.1074805 0.1115437 0.1356221 0.1678137 0.2036488 0.2719867 2022-03-25
## 6: 0.1078464 0.1122669 0.1357229 0.1704501 0.2116249 0.2578800 2022-03-24
```

```r
data3 <- read_table2("tasa_guber_st.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double()
## )
## i Use ‘spec()‘ for the full column specifications.
```

```r
n3<-nrow(data3)
m3_bd=ncol(data3)
fecha = as.Date(as.character(data3$DATE[1:n3]),format="%Y%m%d")
X3_orig_bd=as.data.table(mutate(data3[1:n3,1:m3_bd],Date=fecha))
X3_orig_bd=X3_orig_bd%>%select(-DATE)
nodos3_bd=data.frame(t(as.double(colnames(data3)[2:m3_bd])))
n3=n3-1
head(X3_orig_bd)
```

```
##             1          7         30         90        180        270        360
## 1: 0.02391828 0.02392822 0.02396637 0.02456272 0.02526796 0.02602048 0.02669864
## 2: 0.02336204 0.02392822 0.02396637 0.02450638 0.02532565 0.02602048 0.02675945
## 3: 0.02336204 0.02392822 0.02407656 0.02456272 0.02515258 0.02596080 0.02675945
## 4: 0.02363793 0.02392822 0.02407656 0.02456272 0.02515258 0.02596080 0.02675945
## 5: 0.02336204 0.02392822 0.02407656 0.02473172 0.02544103 0.02637028 0.02706486
## 6: 0.02336204 0.02390713 0.02407656 0.02478806 0.02549872 0.02647756 0.02718850
##           720       1080       1440       1800       2160       2520       2880
## 1: 0.02914780 0.03112396 0.03264148 0.03325475 0.03410190 0.03499666 0.03581823
## 2: 0.02907534 0.03097984 0.03296339 0.03325432 0.03364892 0.03458755 0.03565441
## 3: 0.02907393 0.03120859 0.03337416 0.03361198 0.03381416 0.03453449 0.03590110
## 4: 0.02881413 0.03068465 0.03241812 0.03264848 0.03315642 0.03374040 0.03508592
## 5: 0.02924377 0.03147292 0.03349091 0.03387508 0.03429484 0.03486073 0.03617538
## 6: 0.02939852 0.03160503 0.03343818 0.03411354 0.03436784 0.03537617 0.03654856
##          3240       3600       5400       7200       9000      10800       Date
## 1: 0.03665899 0.03752893 0.04185193 0.04549149 0.04972293 0.05445969 2022-03-31
## 2: 0.03680371 0.03799642 0.04150620 0.04576408 0.04817418 0.05479771 2022-03-30
## 3: 0.03732615 0.03832862 0.04119123 0.04480928 0.04651014 0.05190285 2022-03-29
## 4: 0.03646520 0.03711282 0.04015446 0.04262126 0.04579606 0.04902964 2022-03-28
## 5: 0.03751995 0.03818999 0.04120767 0.04333807 0.04460607 0.05019870 2022-03-25
## 6: 0.03764769 0.03843760 0.04123830 0.04401894 0.04635311 0.04759513 2022-03-24
```

```r
print(nodos3_bd)
```

```
##   X1 X2 X3 X4  X5  X6  X7  X8   X9  X10  X11  X12  X13  X14  X15  X16  X17  X18
## 1  1  7 30 90 180 270 360 720 1080 1440 1800 2160 2520 2880 3240 3600 5400 7200
##    X19   X20
## 1 9000 10800
```

```r
data2 <- read_table2("tfondeo.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   fecha = col_double(),
##   tfondeo = col_double()
## )
```

```r
n2<-nrow(data2)
X2_orig_bd=data.frame(data2[1:n2,1:2])
fecha1 = as.Date(as.character(data2$fecha[1:n2]),format="%Y%m%d")
X2_orig_bd=mutate(X2_orig_bd, fecha=fecha1, tfondeo=as.numeric(as.character(X2_orig_bd$tfondeo)),fecha
tfh=seq(min(X2_orig_bd$fecha),  max(X2_orig_bd$fecha), "days") #sucesión de dias para tasa fondeo
tfhd=data.frame(ID=1:length(tfh),fecha=tfh)
head(X2_orig_bd)
```

```
##        fecha tfondeo
## 1 1998-11-03   30.75
## 2 1998-11-04   29.20
## 3 1998-11-05   29.80
## 4 1998-11-06   31.30
## 5 1998-11-09   32.90
## 6 1998-11-10   34.54
```

```r
tail(tfhd)
```

```
##        ID      fecha
## 8545 8545 2022-03-26
## 8546 8546 2022-03-27
## 8547 8547 2022-03-28
## 8548 8548 2022-03-29
## 8549 8549 2022-03-30
## 8550 8550 2022-03-31
```

Como son tres bases de datos diferentes:

```r
#TASA DE DESCUENTO GUBER
head(X1_orig)
```

```
##             1          7         30         90        180        270        360
## 1: 0.07845614 0.07749243 0.07650381 0.07836683 0.07927250 0.07980972 0.08002044
## 2: 0.07663158 0.07749243 0.07650381 0.07818709 0.07945348 0.07980972 0.08020272
```

```
## 3: 0.07663158 0.07749243 0.07685555 0.07836683 0.07891052 0.07962667 0.08020272
## 4: 0.07753656 0.07749243 0.07685555 0.07836683 0.07891052 0.07962667 0.08020272
## 5: 0.07663158 0.07749243 0.07685555 0.07890605 0.07981546 0.08088263 0.08111809
## 6: 0.07663158 0.07742413 0.07685555 0.07908579 0.07999645 0.08121168 0.08148864
##           720       1080       1440       1800       2160       2520       2880
## 1: 0.08049065 0.08410321 0.08732571 0.09061598 0.09341216 0.09643985 0.10064789
## 2: 0.08029056 0.08371378 0.08818690 0.09061479 0.09217138 0.09531249 0.10018758
## 3: 0.08028669 0.08433190 0.08928583 0.09158939 0.09262398 0.09516626 0.10088078
## 4: 0.07956923 0.08291612 0.08672814 0.08896394 0.09082230 0.09297800 0.09859015
## 5: 0.08075569 0.08504618 0.08959818 0.09230629 0.09394066 0.09606529 0.10165149
## 6: 0.08118301 0.08540318 0.08945711 0.09295608 0.09414062 0.09748568 0.10270011
##         3240      3600      5400      7200      9000     10800       Date
## 1: 0.1050141 0.1096128 0.1377425 0.1761521 0.2270098 0.2950736 2022-03-31
## 2: 0.1054287 0.1109783 0.1366046 0.1772077 0.2199390 0.2969051 2022-03-30
## 3: 0.1069253 0.1119486 0.1355680 0.1735105 0.2123418 0.2812201 2022-03-29
## 4: 0.1044590 0.1083975 0.1321558 0.1650380 0.2090817 0.2656525 2022-03-28
## 5: 0.1074805 0.1115437 0.1356221 0.1678137 0.2036488 0.2719867 2022-03-25
## 6: 0.1078464 0.1122669 0.1357229 0.1704501 0.2116249 0.2578800 2022-03-24
```

```r
#SOBRE TASA GUBER
head(X3_orig_bd)
```

```
##             1          7         30         90        180        270        360
## 1: 0.02391828 0.02392822 0.02396637 0.02456272 0.02526796 0.02602048 0.02669864
## 2: 0.02336204 0.02392822 0.02396637 0.02450638 0.02532565 0.02602048 0.02675945
## 3: 0.02336204 0.02392822 0.02407656 0.02456272 0.02515258 0.02596080 0.02675945
## 4: 0.02363793 0.02392822 0.02407656 0.02456272 0.02515258 0.02596080 0.02675945
## 5: 0.02336204 0.02392822 0.02407656 0.02473172 0.02544103 0.02637028 0.02706486
## 6: 0.02336204 0.02390713 0.02407656 0.02478806 0.02549872 0.02647756 0.02718850
##           720       1080       1440       1800       2160       2520       2880
## 1: 0.02914780 0.03112396 0.03264148 0.03325475 0.03410190 0.03499666 0.03581823
## 2: 0.02907534 0.03097984 0.03296339 0.03325432 0.03364892 0.03458755 0.03565441
## 3: 0.02907393 0.03120859 0.03337416 0.03361198 0.03381416 0.03453449 0.03590110
## 4: 0.02881413 0.03068465 0.03241812 0.03264848 0.03315642 0.03374040 0.03508592
## 5: 0.02924377 0.03147292 0.03349091 0.03387508 0.03429484 0.03486073 0.03617538
## 6: 0.02939852 0.03160503 0.03343818 0.03411354 0.03436784 0.03537617 0.03654856
##          3240       3600       5400       7200       9000      10800       Date
## 1: 0.03665899 0.03752893 0.04185193 0.04549149 0.04972293 0.05445969 2022-03-31
## 2: 0.03680371 0.03799642 0.04150620 0.04576408 0.04817418 0.05479771 2022-03-30
## 3: 0.03732615 0.03832862 0.04119123 0.04480928 0.04651014 0.05190285 2022-03-29
## 4: 0.03646520 0.03711282 0.04015446 0.04262126 0.04579606 0.04902964 2022-03-28
## 5: 0.03751995 0.03818999 0.04120767 0.04333807 0.04460607 0.05019870 2022-03-25
## 6: 0.03764769 0.03843760 0.04123830 0.04401894 0.04635311 0.04759513 2022-03-24
```

```r
#TASA DE FONDEO
head(X2_orig_bd)
```

```
##        fecha tfondeo
## 1 1998-11-03   30.75
## 2 1998-11-04   29.20
## 3 1998-11-05   29.80
## 4 1998-11-06   31.30
## 5 1998-11-09   32.90
## 6 1998-11-10   34.54
```

NECESITAMOS LAS FECHAS COMUNES, POR LO QUE PROCEDEREMOS A INTERSECTAR LAS FECHAS DE ESTAS TRES TASAS:

```
#Cruzar la sucesión de todos los días versus el de tasa de fondeo
tfhd=setDT(tfhd)[, Date := tfh][order(-Date)]
X2_orig_bd=setDT(X2_orig_bd)[, Date := fecha][order(-Date)]
# rolling join unión por rolling, rellena las fechas que faltaban con el último valor conocido "roll=In
X2_orig_bd=X2_orig_bd[tfhd, on = .(Date), roll = Inf]
head(tfhd)
```

```
##      ID      fecha       Date
## 1: 8550 2022-03-31 2022-03-31
## 2: 8549 2022-03-30 2022-03-30
## 3: 8548 2022-03-29 2022-03-29
## 4: 8547 2022-03-28 2022-03-28
## 5: 8546 2022-03-27 2022-03-27
## 6: 8545 2022-03-26 2022-03-26
```

```
head(X2_orig_bd)
```

```
##         fecha tfondeo       Date   ID    i.fecha
## 1: 2022-03-31    6.52 2022-03-31 8550 2022-03-31
## 2: 2022-03-30    6.48 2022-03-30 8549 2022-03-30
## 3: 2022-03-29    6.46 2022-03-29 8548 2022-03-29
## 4: 2022-03-28    6.45 2022-03-28 8547 2022-03-28
## 5: 2022-03-25    6.47 2022-03-27 8546 2022-03-27
## 6: 2022-03-25    6.47 2022-03-26 8545 2022-03-26
```

Ahora alineamos con tasa de fondeo:

```
#buscar fecha de valuación en tfondeo
tf_act=X2_orig_bd[fecha==fval,]$tfondeo/100
tf_int=X2_orig_bd[fecha<=fval & fecha>=(fval-plazocupon_bdm[1])]$tfondeo/100

X1_orig=setDT(X1_orig)[, Date:= Date][order(-Date)] #Para alinear con valor presente y tasa de fondeo.
head(X1_orig)
```

```
##             1          7         30         90        180        270        360
## 1: 0.07845614 0.07749243 0.07650381 0.07836683 0.07927250 0.07980972 0.08002044
## 2: 0.07663158 0.07749243 0.07650381 0.07818709 0.07945348 0.07980972 0.08020272
## 3: 0.07663158 0.07749243 0.07685555 0.07836683 0.07891052 0.07962667 0.08020272
## 4: 0.07753656 0.07749243 0.07685555 0.07836683 0.07891052 0.07962667 0.08020272
## 5: 0.07663158 0.07749243 0.07685555 0.07890605 0.07981546 0.08088263 0.08111809
## 6: 0.07663158 0.07742413 0.07685555 0.07908579 0.07999645 0.08121168 0.08148864
##           720       1080       1440       1800       2160       2520       2880
## 1: 0.08049065 0.08410321 0.08732571 0.09061598 0.09341216 0.09643985 0.10064789
## 2: 0.08029056 0.08371378 0.08818690 0.09061479 0.09217138 0.09531249 0.10018758
## 3: 0.08028669 0.08433190 0.08928583 0.09158939 0.09262398 0.09516626 0.10088078
## 4: 0.07956923 0.08291612 0.08672814 0.08896394 0.09082230 0.09297800 0.09859015
## 5: 0.08075569 0.08504618 0.08959818 0.09230629 0.09394066 0.09606529 0.10165149
## 6: 0.08118301 0.08540318 0.08945711 0.09295608 0.09414062 0.09748568 0.10270011
##          3240       3600       5400       7200       9000      10800        Date
```

```
## 1: 0.1050141 0.1096128 0.1377425 0.1761521 0.2270098 0.2950736 2022-03-31
## 2: 0.1054287 0.1109783 0.1366046 0.1772077 0.2199390 0.2969051 2022-03-30
## 3: 0.1069253 0.1119486 0.1355680 0.1735105 0.2123418 0.2812201 2022-03-29
## 4: 0.1044590 0.1083975 0.1321558 0.1650380 0.2090817 0.2656525 2022-03-28
## 5: 0.1074805 0.1115437 0.1356221 0.1678137 0.2036488 0.2719867 2022-03-25
## 6: 0.1078464 0.1122669 0.1357229 0.1704501 0.2116249 0.2578800 2022-03-24
```

Cargaremos la información del Swap:

```
#CARGA DE DATOS PARA SWAP
data1 <- read_table2("tasa_TIIE_SW_OP.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   DATE = col_double(),
##   `1` = col_double(),
##   `7` = col_double(),
##   `30` = col_double(),
##   `90` = col_double(),
##   `180` = col_double(),
##   `270` = col_double(),
##   `360` = col_double(),
##   `720` = col_double(),
##   `1080` = col_double(),
##   `1440` = col_double(),
##   `1800` = col_double(),
##   `2160` = col_double(),
##   `2520` = col_double(),
##   `2880` = col_double(),
##   `3240` = col_double(),
##   `3600` = col_double()
## )
```

```
  n1<-nrow(data1)
  m1_orig_sw=ncol(data1)
  fecha = as.Date(as.character(data1$DATE[1:n1]),format="%Y%m%d")
  X1_orig_sw=data.table(mutate(data1[1:n1,1:m1_orig_sw],Date=fecha))
  X1_orig_sw=X1_orig_sw%>%select(-DATE)
   nodos1_sw=data.frame(t(as.double(colnames(data1)[2:m1_orig_sw])))
  head(X1_orig_sw)
```

```
##          1        7       30       90      180      270      360      720
## 1: 6.7325 6.741561 6.776711 6.876720 7.018236 7.101177 7.167819 7.333879
## 2: 6.7350 6.761978 6.807476 6.909721 7.058050 7.134662 7.201788 7.352636
## 3: 6.7300 6.749777 6.773426 6.882523 7.023032 7.099393 7.166193 7.316428
## 4: 6.7304 6.760944 6.817428 6.916747 7.052286 7.141966 7.202548 7.361393
## 5: 6.7370 6.761432 6.793198 6.891018 7.033097 7.115017 7.175475 7.325595
## 6: 6.7345 6.752606 6.805899 6.903903 7.046247 7.128320 7.188891 7.339292
##        1080     1440     1800     2160     2520     2880     3240     3600
## 1: 7.467442 7.719981 8.036193 8.430535 8.856091 9.317193 9.812606 10.34490
## 2: 7.486552 7.713437 8.000478 8.381864 8.794147 9.250461 9.741415 10.26873
## 3: 7.432990 7.675320 7.961553 8.341011 8.751209 9.201298 9.685005 10.20427
```

```
## 4: 7.470267 7.731231 8.010775 8.407156 8.837373 9.285331 9.764269 10.27793
## 5: 7.442012 7.710773 8.008414 8.400404 8.824418 9.263361 9.732015 10.23414
## 6: 7.455926 7.725140 8.032622 8.441382 8.884173 9.339574 9.826239 10.34854
##          Date
## 1: 2022-03-31
## 2: 2022-03-30
## 3: 2022-03-29
## 4: 2022-03-28
## 5: 2022-03-25
## 6: 2022-03-24
```

```
data2 <- read_table2("tasa_DIRS_SW_OP.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   DATE = col_double(),
##   `1` = col_double(),
##   `7` = col_double(),
##   `30` = col_double(),
##   `90` = col_double(),
##   `180` = col_double(),
##   `270` = col_double(),
##   `360` = col_double(),
##   `720` = col_double(),
##   `1080` = col_double(),
##   `1440` = col_double(),
##   `1800` = col_double(),
##   `2160` = col_double(),
##   `2520` = col_double(),
##   `2880` = col_double(),
##   `3240` = col_double(),
##   `3600` = col_double()
## )
```

```
n2<-nrow(data2)
m2_orig_sw=ncol(data2)
fecha1 = as.Date(as.character(data2$DATE[1:n1]),format="%Y%m%d")
X2_orig_sw=data.table(mutate(data2[1:n1,1:m1_orig_sw],Date=fecha1))
X2_orig_sw=X2_orig_sw%>%select(-DATE)
nodos2_sw=data.frame(t(as.double(colnames(data2)[2:m2_orig_sw])))
head(X2_orig_sw)
```

```
##           1        7       30       90      180      270      360      720
## 1: 6.605831 6.689449 6.775184 6.828315 6.896106 6.905513 6.898793 6.784002
## 2: 6.651778 6.735977 6.822308 6.875809 6.944071 6.946175 6.931947 6.795927
## 3: 6.633530 6.717499 6.803593 6.856946 6.925022 6.934019 6.926814 6.763071
## 4: 6.615352 6.699091 6.784949 6.838156 6.906045 6.944522 6.967215 6.804032
## 5: 6.597443 6.680955 6.766581 6.819644 6.887349 6.932852 6.962704 6.771289
## 6: 6.609779 6.693447 6.779232 6.832395 6.900226 6.945815 6.975722 6.719772
##        1080     1440     1800     2160     2520     2880     3240     3600
## 1: 6.652051 6.607669 6.597928 6.620858 6.645707 6.672339 6.698625 6.724239
## 2: 6.673120 6.592126 6.586188 6.598651 6.624055 6.659911 6.704141 6.737388
```

```
## 3: 6.617081 6.532031 6.530935 6.530724 6.537380 6.558274 6.599171 6.638533
## 4: 6.620208 6.589337 6.557648 6.519281 6.537343 6.567671 6.601735 6.628008
## 5: 6.563234 6.560197 6.559198 6.554876 6.551990 6.562624 6.591168 6.607032
## 6: 6.562808 6.508976 6.523621 6.481613 6.457319 6.465143 6.513621 6.532930
##          Date
## 1: 2022-03-31
## 2: 2022-03-30
## 3: 2022-03-29
## 4: 2022-03-28
## 5: 2022-03-25
## 6: 2022-03-24
```

Ahora vamos con las opciones(oir en abreviatura que significa "option of rate interest"):

```
  #CARGA DE DATOS PARA OPCIONES DE TASA DE INTERÉS
  #carga de datos
  #carga de rho
library(readr)
data1 <- read_table2("tasa_TIIE_SW_OP.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   DATE = col_double(),
##   `1` = col_double(),
##   `7` = col_double(),
##   `30` = col_double(),
##   `90` = col_double(),
##   `180` = col_double(),
##   `270` = col_double(),
##   `360` = col_double(),
##   `720` = col_double(),
##   `1080` = col_double(),
##   `1440` = col_double(),
##   `1800` = col_double(),
##   `2160` = col_double(),
##   `2520` = col_double(),
##   `2880` = col_double(),
##   `3240` = col_double(),
##   `3600` = col_double()
## )
```

```
  head(data1)
```

```
## # A tibble: 6 x 17
##       DATE   `1`   `7`  `30`  `90` `180` `270` `360` `720` `1080` `1440` `1800`
##      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1 20220331  6.73  6.74  6.78  6.88  7.02  7.10  7.17  7.33   7.47   7.72   8.04
## 2 20220330  6.74  6.76  6.81  6.91  7.06  7.13  7.20  7.35   7.49   7.71   8.00
## 3 20220329  6.73  6.75  6.77  6.88  7.02  7.10  7.17  7.32   7.43   7.68   7.96
## 4 20220328  6.73  6.76  6.82  6.92  7.05  7.14  7.20  7.36   7.47   7.73   8.01
## 5 20220325  6.74  6.76  6.79  6.89  7.03  7.12  7.18  7.33   7.44   7.71   8.01
## 6 20220324  6.73  6.75  6.81  6.90  7.05  7.13  7.19  7.34   7.46   7.73   8.03
```

```
## # ... with 5 more variables: 2160 <dbl>, 2520 <dbl>, 2880 <dbl>, 3240 <dbl>,
## #   3600 <dbl>
```

```r
n<-nrow(data1)
m1_orig_oir=ncol(data1)
 fecha1 = as.Date(as.character(data1$DATE[1:n]),format="%Y%m%d")
x1_orig_oir=data.table(mutate(data1[1:n,1:m1_orig_oir],Date=fecha1))
x1_orig_oir=x1_orig_oir%>%select(-DATE)
nodos1_oir=data.frame(t(as.double(colnames(data1)[2:m1_orig_oir])))
head(x1_orig_oir)
```

```
##          1        7       30       90      180      270      360      720
## 1: 6.7325 6.741561 6.776711 6.876720 7.018236 7.101177 7.167819 7.333879
## 2: 6.7350 6.761978 6.807476 6.909721 7.058050 7.134662 7.201788 7.352636
## 3: 6.7300 6.749777 6.773426 6.882523 7.023032 7.099393 7.166193 7.316428
## 4: 6.7304 6.760944 6.817428 6.916747 7.052286 7.141966 7.202548 7.361393
## 5: 6.7370 6.761432 6.793198 6.891018 7.033097 7.115017 7.175475 7.325595
## 6: 6.7345 6.752606 6.805899 6.903903 7.046247 7.128320 7.188891 7.339292
##        1080     1440     1800     2160     2520     2880     3240     3600
## 1: 7.467442 7.719981 8.036193 8.430535 8.856091 9.317193 9.812606 10.34490
## 2: 7.486552 7.713437 8.000478 8.381864 8.794147 9.250461 9.741415 10.26873
## 3: 7.432990 7.675320 7.961553 8.341011 8.751209 9.201298 9.685005 10.20427
## 4: 7.470267 7.731231 8.010775 8.407156 8.837373 9.285331 9.764269 10.27793
## 5: 7.442012 7.710773 8.008414 8.400404 8.824418 9.263361 9.732015 10.23414
## 6: 7.455926 7.725140 8.032622 8.441382 8.884173 9.339574 9.826239 10.34854
##          Date
## 1: 2022-03-31
## 2: 2022-03-30
## 3: 2022-03-29
## 4: 2022-03-28
## 5: 2022-03-25
## 6: 2022-03-24
```

La opción derivado de que depende de un activo subyacente y estas son opciones de tasa de interes, recae en el subyacente que es valor de la tasa spot, el cual importamos a continuación.

```r
#data1[1:3,]
 #carga de tasas spot
library(readr)
data2 <- read_table2("tasa_DIRS_SW_OP.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   DATE = col_double(),
##   `1` = col_double(),
##   `7` = col_double(),
##   `30` = col_double(),
##   `90` = col_double(),
##   `180` = col_double(),
##   `270` = col_double(),
##   `360` = col_double(),
```

```
##   `720` = col_double(),
##   `1080` = col_double(),
##   `1440` = col_double(),
##   `1800` = col_double(),
##   `2160` = col_double(),
##   `2520` = col_double(),
##   `2880` = col_double(),
##   `3240` = col_double(),
##   `3600` = col_double()
## )
```

```
head(data2)
```

```
## # A tibble: 6 x 17
##       DATE   `1`   `7`  `30`  `90` `180` `270` `360` `720` `1080` `1440` `1800`
##      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1 20220331  6.61  6.69  6.78  6.83  6.90  6.91  6.90  6.78   6.65   6.61   6.60
## 2 20220330  6.65  6.74  6.82  6.88  6.94  6.95  6.93  6.80   6.67   6.59   6.59
## 3 20220329  6.63  6.72  6.80  6.86  6.93  6.93  6.93  6.76   6.62   6.53   6.53
## 4 20220328  6.62  6.70  6.78  6.84  6.91  6.94  6.97  6.80   6.62   6.59   6.56
## 5 20220325  6.60  6.68  6.77  6.82  6.89  6.93  6.96  6.77   6.56   6.56   6.56
## 6 20220324  6.61  6.69  6.78  6.83  6.90  6.95  6.98  6.72   6.56   6.51   6.52
## # ... with 5 more variables: 2160 <dbl>, 2520 <dbl>, 2880 <dbl>, 3240 <dbl>,
## #   3600 <dbl>
```

```
n<-nrow(data2)
m2_orig_oir=ncol(data2)
fecha1 = as.Date(as.character(data2$DATE[1:n]),format="%Y%m%d")
x2_orig_oir=data.table(mutate(data2[1:n,1:m2_orig_oir],Date=fecha1))
x2_orig_oir=x2_orig_oir%>%select(-DATE)
nodos2_oir=data.frame(t(as.double(colnames(data2)[2:m2_orig_oir])))
head(x2_orig_oir)
```

```
##           1        7       30       90      180      270      360      720
## 1: 6.605831 6.689449 6.775184 6.828315 6.896106 6.905513 6.898793 6.784002
## 2: 6.651778 6.735977 6.822308 6.875809 6.944071 6.946175 6.931947 6.795927
## 3: 6.633530 6.717499 6.803593 6.856946 6.925022 6.934019 6.926814 6.763071
## 4: 6.615352 6.699091 6.784949 6.838156 6.906045 6.944522 6.967215 6.804032
## 5: 6.597443 6.680955 6.766581 6.819644 6.887349 6.932852 6.962704 6.771289
## 6: 6.609779 6.693447 6.779232 6.832395 6.900226 6.945815 6.975722 6.719772
##        1080     1440     1800     2160     2520     2880     3240     3600
## 1: 6.652051 6.607669 6.597928 6.620858 6.645707 6.672339 6.698625 6.724239
## 2: 6.673120 6.592126 6.586188 6.598651 6.624055 6.659911 6.704141 6.737388
## 3: 6.617081 6.532031 6.530935 6.530724 6.537380 6.558274 6.599171 6.638533
## 4: 6.620208 6.589337 6.557648 6.519281 6.537343 6.567671 6.601735 6.628008
## 5: 6.563234 6.560197 6.559198 6.554876 6.551990 6.562624 6.591168 6.607032
## 6: 6.562808 6.508976 6.523621 6.481613 6.457319 6.465143 6.513621 6.532930
##          Date
## 1: 2022-03-31
## 2: 2022-03-30
## 3: 2022-03-29
## 4: 2022-03-28
## 5: 2022-03-25
## 6: 2022-03-24
```

A continuación necesitamos la desviación estandar del movimiento browniano geometrico, el cual se traduce en una superficie de volatilidades:

```
  #carga de volatilidades de spot
library(readr)
data3 <- read_table2("tvoltiie_opc.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   DATE = col_double(),
##   `28` = col_double(),
##   `91` = col_double(),
##   `182` = col_double(),
##   `364` = col_double(),
##   `728` = col_double(),
##   `1092` = col_double(),
##   `1820` = col_double(),
##   `3640` = col_double()
## )
```

```
head(data3)
```

```
## # A tibble: 6 x 9
##       DATE  `28`  `91` `182` `364` `728` `1092` `1820` `3640`
##      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1 20220331 0.467 0.333 0.274 0.206 0.251  0.278  0.280  0.272
## 2 20220330 0.491 0.350 0.289 0.218 0.263  0.279  0.297  0.322
## 3 20220329 0.491 0.350 0.289 0.218 0.263  0.279  0.297  0.322
## 4 20220328 0.490 0.350 0.289 0.218 0.263  0.279  0.297  0.323
## 5 20220325 0.509 0.362 0.302 0.233 0.287  0.290  0.297  0.332
## 6 20220324 0.508 0.362 0.301 0.233 0.287  0.290  0.297  0.333
```

```
  n<-nrow(data3)
  m3_orig_oir=ncol(data3)
  fecha1 = as.Date(as.character(data3$DATE[1:n]),format="%Y%m%d")
  x3_orig_oir=data.table(mutate(data3[1:n,1:m3_orig_oir],Date=fecha1))
  x3_orig_oir=x3_orig_oir%>%select(-DATE)
  nodos3_oir=data.frame(t(as.double(colnames(data3)[2:m3_orig_oir])))
  head(x3_orig_oir)
```

```
##           28        91       182       364       728      1092      1820
## 1: 0.4665709 0.3328868 0.2742244 0.2059151 0.2510428 0.2783824 0.2798213
## 2: 0.4905821 0.3499227 0.2891028 0.2183196 0.2634969 0.2793499 0.2967162
## 3: 0.4905484 0.3498969 0.2890889 0.2183196 0.2634969 0.2793499 0.2967162
## 4: 0.4904293 0.3498066 0.2890363 0.2183196 0.2634969 0.2793499 0.2967162
## 5: 0.5088287 0.3620960 0.3017326 0.2328081 0.2869167 0.2902659 0.2967162
## 6: 0.5081509 0.3615701 0.3014171 0.2328081 0.2872888 0.2902659 0.2967162
##         3640       Date
## 1: 0.2724996 2022-03-31
## 2: 0.3224502 2022-03-30
## 3: 0.3224442 2022-03-29
```

```
## 4: 0.3225564 2022-03-28
## 5: 0.3322934 2022-03-25
## 6: 0.3330347 2022-03-24
```

Pasamos a los futuros, primero ajustaremos la información de las tasas libor y forward filtradas por fecha de forma decreciente tal cual lo hemos hecho arriba.

```r
#CARGA DE DATOS DE  FORWARDS DE TDC
#datas
#data<-read.table("tasa_tiie.txt")
########minimos para parametrizar
library(readr)
data1 <- read_table2("tasa_libor.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   DATE = col_double(),
##   `1` = col_double(),
##   `7` = col_double(),
##   `30` = col_double(),
##   `90` = col_double(),
##   `180` = col_double(),
##   `270` = col_double(),
##   `360` = col_double(),
##   `720` = col_double(),
##   `1080` = col_double(),
##   `1440` = col_double(),
##   `1800` = col_double(),
##   `2160` = col_double(),
##   `2520` = col_double(),
##   `2880` = col_double(),
##   `3240` = col_double(),
##   `3600` = col_double()
## )
```

```r
data2 <- read_table2("tasa_fwd.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   DATE = col_double(),
##   `1` = col_double(),
##   `7` = col_double(),
##   `30` = col_double(),
##   `90` = col_double(),
##   `180` = col_double(),
##   `270` = col_double(),
##   `360` = col_double(),
##   `720` = col_double(),
##   `1080` = col_double(),
##   `1440` = col_double(),
##   `1800` = col_double(),
```

```
##   `2160` = col_double(),
##   `2520` = col_double(),
##   `2880` = col_double(),
##   `3240` = col_double(),
##   `3600` = col_double()
## )
```

```
n1=nrow(data1)
n2=nrow(data2)
m1_ftdc=ncol(data1)
m2_ftdc=ncol(data2)
n=min(n1,n2)-1
###NODOS###
nodos1_ftdc=data.frame(t(as.double(colnames(data1[2:m1_ftdc]))))
nodos2_ftdc=data.frame(t(as.double(colnames(data2[2:m2_ftdc]))))
```

```
####MATRICES DEL MISMO TAMAÑO MENOS DOLAR
fecha1 = as.Date(as.character(data1$DATE[1:n]),format="%Y%m%d")
x1_ftdc=as.data.table(mutate(data1[1:n,1:m1_ftdc],Date=fecha1))
x1_ftdc=x1_ftdc%>%select(-DATE)
fecha1 = as.Date(as.character(data2$DATE[1:n]),format="%Y%m%d")
x2_ftdc=as.data.table(mutate(data2[1:n,1:m2_ftdc],Date=fecha1))
x2_ftdc=x2_ftdc%>%select(-DATE)
head(x1_ftdc)
```

```
##           1        7       30       90      180      270      360      720
## 1: 1.920598 1.963944 2.074285 2.307229 2.365464 2.462065 2.545069 2.782139
## 2: 1.912261 1.960613 2.073887 2.307381 2.367677 2.462023 2.545086 2.776117
## 3: 1.920327 1.966791 2.073757 2.309709 2.367626 2.462014 2.545079 2.776160
## 4: 1.908752 1.958053 2.074524 2.307065 2.363016 2.461702 2.542420 2.776219
## 5: 1.915810 1.963510 2.072762 2.304722 2.362984 2.459071 2.539739 2.770218
## 6: 1.911525 1.957288 2.072762 2.304722 2.362984 2.459071 2.539739 2.770218
##        1080     1440     1800     2160     2520     2880     3240     3600
## 1: 2.904359 2.986545 3.014051 3.059628 3.112997 3.170556 3.232499 3.297026
## 2: 2.898076 2.969958 2.986522 3.027635 3.076663 3.133020 3.193931 3.257336
## 3: 2.891626 2.969952 2.986746 3.027836 3.076840 3.131840 3.191200 3.252958
## 4: 2.888438 2.973377 2.986919 3.033271 3.088221 3.141205 3.197734 3.256495
## 5: 2.885324 2.973560 2.994144 3.039062 3.092065 3.142280 3.195823 3.251421
## 6: 2.885324 2.973540 2.997591 3.048187 3.107193 3.162220 3.220742 3.281633
##          Date
## 1: 2022-03-31
## 2: 2022-03-30
## 3: 2022-03-29
## 4: 2022-03-28
## 5: 2022-03-25
## 6: 2022-03-24
```

```
head(x2_ftdc)
```

```
##           1        7       30       90      180      270      360      720
## 1: 6.537747 6.537747 6.555484 6.746279 6.889754 6.976922 7.084811 7.220727
## 2: 6.540174 6.557546 6.585244 6.778653 6.928838 7.009822 7.118386 7.239195
```

```
## 3: 6.535319 6.545715 6.552305 6.751972 6.894462 6.975170 7.083204 7.203546
## 4: 6.535707 6.556544 6.594871 6.785546 6.923180 7.016998 7.119138 7.247817
## 5: 6.542117 6.557017 6.571432 6.760305 6.904342 6.990520 7.092378 7.212571
## 6: 6.539689 6.548458 6.583719 6.772945 6.917251 7.003590 7.105639 7.226056
##        1080     1440     1800     2160     2520     2880     3240     3600
## 1: 7.269200 7.440239 7.708522 8.044548 8.407624 8.802796 9.240754 9.727323
## 2: 7.287802 7.433931 7.674264 7.998106 8.348817 8.739748 9.173712 9.655704
## 3: 7.235662 7.397195 7.636925 7.959122 8.308053 8.693299 9.120589 9.595092
## 4: 7.271949 7.451080 7.684141 8.022239 8.389854 8.772693 9.195234 9.664353
## 5: 7.244444 7.431364 7.681875 8.015797 8.377555 8.751936 9.164860 9.623172
## 6: 7.257989 7.445210 7.705097 8.054899 8.434285 8.823941 9.253592 9.730748
##          Date
## 1: 2022-03-31
## 2: 2022-03-30
## 3: 2022-03-29
## 4: 2022-03-28
## 5: 2022-03-25
## 6: 2022-03-24
```

Tenemos que distinguir dos casos y para eso esta la variable yext, la cual nos indica si procedemos a bajar la información de una base de datos o de yahoo finance. No modificaremos el codigo si yext = 0 pues se trabaja con yahoo finance, pero de ser este el caso, deben introducirse al codigo, la vbase de datos correspondiente y cambiar “V1” de la forma en que se hizo arriba.

```
###Para Dolar

if (yext==1)
{
#Cargar los símbolos de yahoo finance para FX
  start_date=fval-3660 #fecha inicial



#Creación del objeto para guardar los datos
  dataEnvFX_ftdc<-new.env()

#obtener los datos
  getSymbols.yahoo(SymbolsFX_ftdc,env=dataEnvFX_ftdc,from=start_date, to=(fval))
#limpiarlos, alinearnos y quedarnos con el precio de cierre
  bt.prep(dataEnvFX_ftdc,align='remove.na',fill.gaps=T)

#muestra de datos
  head(dataEnvFX_ftdc$prices[,2])

#Nos quedamos con los precios
  X3_ftdc=data.table(Date=as.Date(index(dataEnvFX_ftdc$prices[,2])),coredata(dataEnvFX_ftdc$prices[,2])
} else
{
data3<-read.table(btsp)
print(head(data3))
n3<-nrow(data3)
m3<-ncol(data3)
X3=data.table(as.matrix(as.double(as.matrix(data3[2:(n+1),m3]))))
X3_find=as.data.table(mutate(data3[2:(n+1),1:m3],Date=as.Date(V1,format="%Y%m%d")))
```

```
}
```

```
## Warning: USDMXN=X contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

## Warning: GBPUSD=X contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
# CARGA DE DATOS DE  FORWARD DE IPC
library(readr)
data <- read_table2("tasa_guber.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```
n<-nrow(data)
m_gov=ncol(data)
fecha1 = as.Date(as.character(data$DATE[1:n]),format="%Y%m%d")
#x_orig_gov=data.frame(data[2:n,1:m_gov])
x_orig_gov=as.data.table(mutate(data[1:n,1:m_gov],Date=fecha1))
x_orig_gov=x_orig_gov%>%select(-DATE)
nodos_gov=data.frame(t(as.double(colnames(data)[2:m_gov])))
  #Cargar los símbolos de yahoo finance para EQ
  start_date=fval-nh #fecha inicial

  #Creación del objeto para guardar los datos
  dataEnvEQ<-new.env()

  #obtener los datos
  getSymbols.yahoo(SymbolsEQ_find,env=dataEnvEQ,from=start_date, to=(fval))
```

```
## Warning: ^MXX contains missing values. Some functions will not work if objects
## contain missing values in the middle of the series. Consider using na.omit(),
## na.approx(), na.fill(), etc to remove or replace them.

## [1] "^MXX"        "GCARSOA1.MX"
```

```
  #limpiarlos, alinearnos y quedarnos con el precio de cierre
  bt.prep(dataEnvEQ,align='remove.na',fill.gaps=T)

  #muestra de datos
#  head(dataEnvEQ$prices)

  #Nos quedamos con los precios
  X3_find=data.table(Date=as.Date(index(dataEnvEQ$prices[,2])),coredata(dataEnvEQ$prices[,2]))
```

INTEGRACIÓN DE INSUMOS

Primero a las acciones y divisas les añadimos sus factores de riesgo en un data table (lin_gub) y despues les añadimos en otro data table(lin_gub_bmybdst) los factores del bonde.

```
#INTERSECCIÓN DE FECHAS DE TODOS LOS INSUMOS

head(x_orig_gov)
```

```
##             1          7         30         90        180        270        360
## 1: 0.07845614 0.07749243 0.07650381 0.07836683 0.07927250 0.07980972 0.08002044
## 2: 0.07663158 0.07749243 0.07650381 0.07818709 0.07945348 0.07980972 0.08020272
## 3: 0.07663158 0.07749243 0.07685555 0.07836683 0.07891052 0.07962667 0.08020272
## 4: 0.07753656 0.07749243 0.07685555 0.07836683 0.07891052 0.07962667 0.08020272
## 5: 0.07663158 0.07749243 0.07685555 0.07890605 0.07981546 0.08088263 0.08111809
## 6: 0.07663158 0.07742413 0.07685555 0.07908579 0.07999645 0.08121168 0.08148864
##           720       1080       1440       1800       2160       2520       2880
## 1: 0.08049065 0.08410321 0.08732571 0.09061598 0.09341216 0.09643985 0.10064789
## 2: 0.08029056 0.08371378 0.08818690 0.09061479 0.09217138 0.09531249 0.10018758
## 3: 0.08028669 0.08433190 0.08928583 0.09158939 0.09262398 0.09516626 0.10088078
## 4: 0.07956923 0.08291612 0.08672814 0.08896394 0.09082230 0.09297800 0.09859015
## 5: 0.08075569 0.08504618 0.08959818 0.09230629 0.09394066 0.09606529 0.10165149
## 6: 0.08118301 0.08540318 0.08945711 0.09295608 0.09414062 0.09748568 0.10270011
##         3240      3600      5400      7200      9000     10800       Date
## 1: 0.1050141 0.1096128 0.1377425 0.1761521 0.2270098 0.2950736 2022-03-31
## 2: 0.1054287 0.1109783 0.1366046 0.1772077 0.2199390 0.2969051 2022-03-30
## 3: 0.1069253 0.1119486 0.1355680 0.1735105 0.2123418 0.2812201 2022-03-29
## 4: 0.1044590 0.1083975 0.1321558 0.1650380 0.2090817 0.2656525 2022-03-28
## 5: 0.1074805 0.1115437 0.1356221 0.1678137 0.2036488 0.2719867 2022-03-25
## 6: 0.1078464 0.1122669 0.1357229 0.1704501 0.2116249 0.2578800 2022-03-24
```

```
fecha1 = as.Date(aux2[x_orig_gov,on=.(Date),nomatch=0]$Date)
lin_gub=data.table(Date=fecha1) #Fechas acciones, equity y guber
nrow(lin_gub)
```

```
## [1] 255
```

```
head(lin_gub)
```

```
##          Date
## 1: 2022-03-31
## 2: 2022-03-30
## 3: 2022-03-29
## 4: 2022-03-28
## 5: 2022-03-25
## 6: 2022-03-24
```

```
fecha1 = as.Date(lin_gub[X3_orig_bd,on=.(Date),nomatch=0]$Date)
lin_gub_bmybdst=data.table(Date=fecha1) #Fechas acciones, equity, guber y st (bonde)
nrow(lin_gub_bmybdst)
```

```
## [1] 255
```

```r
head(lin_gub)
```

```
##          Date
## 1: 2022-03-31
## 2: 2022-03-30
## 3: 2022-03-29
## 4: 2022-03-28
## 5: 2022-03-25
## 6: 2022-03-24
```

```r
#lin_gub_bmybdst_flib=data.table(Date=as.Date(lin_gub_bmybdst[x1_ftdc,on=.(Date),nomatch=0]$Date)) #Fec
```

Continuamos añadiendo fechas comunes a la base anterior los futuros

```r
lin_gub_bmybdst=data.table(Date=as.Date(X3_ftdc[x_orig_gov,on=.(Date),nomatch=0]$Date)) #Fechas accione
print(nrow(lin_gub_bmybdst))
```

```
## [1] 255
```

```r
head(lin_gub_bmybdst)
```

```
##          Date
## 1: 2022-03-31
## 2: 2022-03-30
## 3: 2022-03-29
## 4: 2022-03-28
## 5: 2022-03-25
## 6: 2022-03-24
```

Hacemos lo propio con libor:

```r
lin_gub_bmybdst_flib=data.table(Date=as.Date(lin_gub_bmybdst[x1_ftdc,on=.(Date),nomatch=0]$Date)) #Fech
print(nrow(lin_gub_bmybdst_flib))
```

```
## [1] 254
```

```r
head(lin_gub_bmybdst_flib)
```

```
##          Date
## 1: 2022-03-31
## 2: 2022-03-30
## 3: 2022-03-29
## 4: 2022-03-28
## 5: 2022-03-25
## 6: 2022-03-24
```

Hacemos lo mismo con los factores restantes

```r
lin_gub_bmybdst_flibfwd=data.table(Date=as.Date(lin_gub_bmybdst_flib[x2_ftdc,on=.(Date),nomatch=0]$Date
print(nrow(lin_gub_bmybdst_flibfwd))
```

```
## [1] 254
```

```r
lin_gub_bmybdst_flibfwdspind=data.table(Date=as.Date(lin_gub_bmybdst_flibfwd[X3_find,on=.(Date),nomatch
print(nrow(lin_gub_bmybdst_flibfwdspind))
```

```
## [1] 253
```

```r
lin_gub_bmybdst_flibfwdspind_swcup=data.table(Date=as.Date(lin_gub_bmybdst_flibfwdspind[X1_orig_sw,on=.
print(nrow(lin_gub_bmybdst_flibfwdspind_swcup))
```

```
## [1] 253
```

```r
lin_gub_bmybdst_flibfwdspind_swcupvp=data.table(Date=as.Date(lin_gub_bmybdst_flibfwdspind_swcup[X2_orig
print(nrow(lin_gub_bmybdst_flibfwdspind_swcupvp))
```

```
## [1] 253
```

```r
lin_gub_bmybdst_flibfwdspind_swcupvp_oirs=data.table(Date=as.Date(lin_gub_bmybdst_flibfwdspind_swcupvp[
print(nrow(lin_gub_bmybdst_flibfwdspind_swcupvp_oirs))
```

```
## [1] 253
```

```r
lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvp=data.table(Date=as.Date(lin_gub_bmybdst_flibfwdspind_swcupv
print(nrow(lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvp))
```

```
## [1] 253
```

```r
lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol=data.table(Date=as.Date(lin_gub_bmybdst_flibfwdspind_swc
print(nrow(lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol))
```

```
## [1] 253
```

```r
print(lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol)
```

```
##            Date
##   1: 2022-03-30
##   2: 2022-03-29
##   3: 2022-03-28
##   4: 2022-03-25
##   5: 2022-03-24
##  ---
## 249: 2021-04-09
## 250: 2021-04-08
## 251: 2021-04-07
## 252: 2021-04-06
## 253: 2021-04-05
```

```
print(unique(lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol))
```

```
##           Date
##   1: 2022-03-30
##   2: 2022-03-29
##   3: 2022-03-28
##   4: 2022-03-25
##   5: 2022-03-24
##  ---
## 249: 2021-04-09
## 250: 2021-04-08
## 251: 2021-04-07
## 252: 2021-04-06
## 253: 2021-04-05
```

```
lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol=unique(lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol)
print(lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol)
```

```
##           Date
##   1: 2022-03-30
##   2: 2022-03-29
##   3: 2022-03-28
##   4: 2022-03-25
##   5: 2022-03-24
##  ---
## 249: 2021-04-09
## 250: 2021-04-08
## 251: 2021-04-07
## 252: 2021-04-06
## 253: 2021-04-05
```

Alineamiento de los insumos

Ahora alineamos con base en las fechas anteriormente calculadas en el orden correspondiente cada uno de los insumos descargados antes.

```
n=nrow(lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol) #Historia de todos

#historia de acciones y divisas
stock_prices_EQFX=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[aux2,on=.(Date),nomatch=0][order(-Date
stock_prices_EQFX=stock_prices_EQFX%>%select(-Date)

#historia de curva gubernamental
x_orig_gov=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[x_orig_gov,on=.(Date),nomatch=0][order(-Date)
x_orig_gov=x_orig_gov%>%select(-Date)




#Historia de curvas de bonde
#CONSIDERAR LA CURVA GUBERNAMENTAL X1_ORIG_GOV
X1_orig=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[X1_orig,on=.(Date),nomatch=0][order(-Date)]
```

```
X1_orig=X1_orig%>%select(-Date)
# X2_orig_bd=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[X2_orig_bd,on=.(Date),nomatch=0][order(-Dat
# X2_orig_bd=X2_orig_bd%>%select(-Date)
X3_orig_bd=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[X3_orig_bd,on=.(Date),nomatch=0][order(-Date)]
X3_orig_bd=X3_orig_bd%>%select(-Date)


#historia de curvas de forward tdc
x1_ftdc=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[x1_ftdc,on=.(Date),nomatch=0][order(-Date)]
x1_ftdc=x1_ftdc%>%select(-Date)/100
x2_ftdc=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[x2_ftdc,on=.(Date),nomatch=0][order(-Date)]
x2_ftdc=x2_ftdc%>%select(-Date)/100
X3_ftdc=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[X3_ftdc,on=.(Date),nomatch=0][order(-Date)]
X3_ftdc=X3_ftdc%>%select(-Date)

#historia de curvas de forward ind
#CONSIDERAR LA CURVA GUBERNAMENTAL X1_ORIG_GOV
X3_find=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[X3_find,on=.(Date),nomatch=0][order(-Date)]
X3_find=X3_find%>%select(-Date)


#historia de swaps
X1_orig_sw=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[X1_orig_sw,on=.(Date),nomatch=0][order(-Date)]
X1_orig_sw=X1_orig_sw%>%select(-Date)/100
X2_orig_sw=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[X2_orig_sw,on=.(Date),nomatch=0][order(-Date)]
X2_orig_sw=X2_orig_sw%>%select(-Date)/100

#historia de opciones
x1_orig_oir=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[x1_orig_oir,on=.(Date),nomatch=0][order(-Date
x1_orig_oir=x1_orig_oir%>%select(-Date)/100
x2_orig_oir=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[x2_orig_oir,on=.(Date),nomatch=0][order(-Date
x2_orig_oir=x2_orig_oir%>%select(-Date)/100
x3_orig_oir=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[x3_orig_oir,on=.(Date),nomatch=0][order(-Date
x3_orig_oir=x3_orig_oir%>%select(-Date)
```

SIMULACIÓN HISTÓRICA

Para todos los instrumentos hay que definir y calcular los siguientes elementos:

1.Historico de factores de riesgo

2.Vector de precios actual

3.Valoración al día actual

Acciones y Divisas

```
#Divisas y Acciones CÁLCULO

x0_acc_div=stock_prices_EQFX[1,]
DeltaX_acc_div=as.matrix(log(as.matrix(stock_prices_EQFX[1:(n-1)])/as.matrix(stock_prices_EQFX[2:(n)])))
V0_acc_div=cbind(t(pos_fx),t(pos_eq))*x0_acc_div
```

```r
m_fx=length(pos_fx)
m_acc=length(pos_eq)
x0_acc_div
```

```
##    EURUSD.X GBPUSD.X USDMXN.X AMXL.MX GCARSOA1.MX WMT.MX
## 1: 22.14488 26.14256  19.9656   20.75       73.53   2971
```

```r
print(V0_acc_div)
```

```
##    EURUSD.X  GBPUSD.X USDMXN.X AMXL.MX GCARSOA1.MX  WMT.MX
## 1: 15501.42 -15685.53  29948.4 -103750       73530 3565200
```

Bondes D

```r
#BONDE D CÁLCULO

X2_pr=lin_gub_bmybdst_flibfwdspind_swcupvp_oirsvpvol[X2_orig_bd, on = .(Date),nomatch=0][order(-Date)]
m=ncol(plazos_bdm)


N_bd=as.integer(plazos_bdm/plazocupon_bdm)+1 #número de cupones a pagar
VTplazos_bdm=matrix(0,1,sum(N_bd)) #vector de todos los plazos_bdm de todos los contratos_bdm
contratos_bdmT=matrix(0,1,sum(N_bd)) #vector de todos los contratos_bdm de todos los flujos de todos lo
nominal_bdmT=matrix(0,1,sum(N_bd)) #vector de todos los nominal_bdmes de todos los flujos de todos los
plazocupon_bdmT=matrix(0,1,sum(N_bd)) #vector de todos los plazos_bdmcupon de todos los flujos de todos
tasafijaT_bd=matrix(0,1,sum(N_bd)) #vector de tasas fijas de todos los flujos de todos los contratos_bd
ulNomT_bd=matrix(0,1,sum(N_bd)) #vector de contratos_bdm a final de flujo

plazini_bd=plazos_bdm-plazocupon_bdm*(N_bd-1) #vector de plazos_bdm iniciales
ddv=plazocupon_bdm-plazini_bd #dias trasncurridos del cupón vigente
tfcupon=matrix(0,1,m)  #El primero cupón de cada bono
tfcupondev=matrix(0,1,m) #cupón de los días devengados
tfcupgen=((1+tf_act/360)^(plazocupon_bdm[1])-1)*360/plazocupon_bdm[1] #el segundo al último cupón de to
#calcula cupones de bonos
for (j in (1:m))
{
  tfcupondev[j]=(prod(1+tf_int[(1:ddv[j])]/360)-1)*360/ddv[j]
  tfcupon[j]=((1+tfcupondev[j]*ddv[j]/360)*(1+tf_act/360)^(plazocupon_bdm[1]-ddv[j])-1)*360/plazocupon_
}


for (j in (1:m))
{
  if (j==1)
  {
    VTplazos_bdm[,1:sum(N_bd[1:j])]=seq(plazini_bd[j],plazos_bdm[j], by=plazocupon_bdm[j])
    contratos_bdmT[,1:sum(N_bd[1:j])]=seq(contratos_bdm[j],contratos_bdm[j])
    plazocupon_bdmT[,1:sum(N_bd[1:j])]=seq(plazocupon_bdm[j],plazocupon_bdm[j])
    ulNomT_bd[,sum(N_bd[1:j])]=contratos_bdm[j]
    tasafijaT_bd[,1]=tfcupon[j]
    tasafijaT_bd[,2:sum(N_bd[1:j])]=seq(tfcupgen,tfcupgen)
```

```
  }
  else
  {
    VTplazos_bdm[,(sum(N_bd[1:j-1])+1):sum(N_bd[1:j])]=seq(plazini_bd[j],plazos_bdm[j], by=plazocupon_bd
    contratos_bdmT[,(sum(N_bd[1:j-1])+1):sum(N_bd[1:j])]=seq(contratos_bdm[j],contratos_bdm[j])
    plazocupon_bdmT[,(sum(N_bd[1:j-1])+1):sum(N_bd[1:j])]=seq(plazocupon_bdm[j],plazocupon_bdm[j])
    tasafijaT_bd[,(sum(N_bd[1:j-1])+1)]=tfcupon[j]
    tasafijaT_bd[,(sum(N_bd[1:j-1])+2):sum(N_bd[1:j])]=seq(tfcupgen,tfcupgen)
    ulNomT_bd[,sum(N_bd[1:j])]=contratos_bdm[j]
  }
}

Xvp_bd=matrix(0,(n),ncol(VTplazos_bdm))
Xst_bd=matrix(0,(n),ncol(VTplazos_bdm))



for (i in (1:(n)))
{
  Xvp_bd[i,]=if(itpl==0){approx(nodos_gov,x_orig_gov[i,],VTplazos_bdm,rule=2)$y}else{talamb(nodos_gov,x_
  Xst_bd[i,]=if(itpl==0){approx(nodos3_bd,X3_orig_bd[i,],VTplazos_bdm,rule=2)$y}else{talamb(nodos3_bd,X3

}

X_bd_tc=matrix(1,n,ncol(contratos_bdmT))*X2_pr$tfondeo/100
X_bd_ext=cbind(X_bd_tc,as.matrix(Xvp_bd),as.matrix(Xst_bd))

bondeD=function(contratos_bdmT,  nominal_bdm, tf_act, plazocupon_bdmT, VTplazos_bdm, Xvp, Xst, N,ddv){
  tfcupon=matrix(0,1,m)  #El primero cupón de cada bono
  tfcupondev=matrix(0,1,m) #cupón de los días devengados
  tfcupgen=((1+tf_act/360)^(plazocupon_bdm[1])-1)*360/plazocupon_bdm[1] #el segundo al último cupón de
  tasafijaT=matrix(0,1,sum(N))
  #calcula cupones de bonos
  for (j in (1:m))
  {
    tfcupondev[j]=(prod(1+tf_int[(1:ddv[j])]/360)-1)*360/ddv[j]
    tfcupon[j]=((1+tfcupondev[j]*ddv[j]/360)*(1+tf_act/360)^(plazocupon_bdm[1]-ddv[j])-1)*360/plazocupon
  }


  for (j in (1:m))
  {
    if (j==1)
    {
      tasafijaT[,1]=tfcupon[j]
      tasafijaT[,2:sum(N[1:j])]=seq(tfcupgen,tfcupgen)
    }
    else
    {
      tasafijaT[,(sum(N[1:j-1])+1)]=tfcupon[j]
      tasafijaT[,(sum(N[1:j-1])+2):sum(N[1:j])]=seq(tfcupgen,tfcupgen)
    }
  }
```

```
  VO=matrix(0,1,count(N))
  VOf=((((contratos_bdmT*(tasafijaT)*(plazocupon_bdmT/360))+ulNomT_bd)/(1+(Xvp+Xst)*VTplazos_bdm/360)))
  for (j in (1:count(N)))
  {
    if(j==1)
    {
      VO[j]=sum(VOf[j:N[j]])
    }
    else
    {
      VO[j]=sum(VOf[(sum(N[1:j-1])+1):(sum(N[1:j]))])
    }
  }
  VO
}



VO_bd=bondeD(contratos_bdmT, nominal_bdm, tf_act, plazocupon_bdmT, VTplazos_bdm, Xvp_bd[1,], Xst_bd[1,]

VO_bd
```

```
##        [,1]     [,2]
## [1,] 824.78 942.9217
```

Forwards

Construiremos la función de valoración de forwards:

```
#FORWARDS Y/O FUTUROS DE TIPO DE CAMBIO CÁLCULO

################MATRICES DE INTERPOLACION LINEAL ###################

m=ncol(plazos_fwd)
X1_fwtdc=matrix(0,n,m)
X2_fwtdc=matrix(0,n,m)


for (j in 1:n)
{
  X1_fwtdc[j,]=if(itpl==0){approx(nodos1_ftdc,x1_ftdc[j,],plazos_fwd,rule=2)$y}else{talamb(nodos1_ftdc,
  X2_fwtdc[j,]=if(itpl==0){approx(nodos2_ftdc,x2_ftdc[j,],plazos_fwd,rule=2)$y}else{talamb(nodos2_ftdc,
  if(trlib==1){X1_fwtdc[j,]=((1+X1_fwtdc[j,])^(plazos_fwd/180)-1)*360/plazos_fwd} #transformación de ac
}


futuroTC = function(t,tl,tn,s,k) #t=dias por vencer, tn=tasa nacional para tipo de cambio forward, tl=
{
  f=s*((1+tn*t/360)/(1+tl*t/360)) #Se obtiene el tipo de cambio forward
  t(as.numeric((f-k)/(1+t*tn/360))) #Se obtiene el valor del payoff a valor presente con el valor z que
}
```

```
X3_ftdc=as.matrix(X3_ftdc)
X_futtdc=cbind(X1_fwtdc,X2_fwtdc,X3_ftdc)

V0_fwtdc=futuroTC(plazos_fwd,X1_fwtdc[1,],X2_fwtdc[1,],X3_ftdc[1,],kst_fwd)*contratos_fwd*nominal_fwd
# V0_fwtdc=futuroTC(plazos_fwd,X1_fwtdc[1,],X2_fwtdc[1,],X3_ftdc[1,],kst_fwd)*contratos_fwd*nominal_fwd
V0_fwtdc
```

```
##           [,1]
## [1,] -85.61414
```

```
#FORWARDS Y/O FUTUROS DE ÍNDICES CÁLCULO

###############MATRICES DE INTERPOLACION LINEAL ####################

m_ind=ncol(plazos_fwd_ind)
X1_fwind=matrix(0,n,m_ind) #DIVIDENDOS
X2_fwind=matrix(0,n,m_ind)

for (j in 1:n)
{
  #X1_fwind[j,]=if(itpl==0){approx(nodos1_,x1_ftdc[j,],plazos_fwd)$y}else{talamb(nodos1_ftdc,x1_ftdc[j,
  X2_fwind[j,]=if(itpl==0){approx(nodos_gov,x_orig_gov[j,],plazos_fwd_ind,rule=2)$y}else{talamb(nodos_go
  #if(trlib==1){X1_fwtdc[j,]=((1+X1_fwtdc[j,])^(plazos_fwd/180)-1)*360/plazos_fwd} #transformación de a
}

X3_find=as.matrix(X3_find)
X_futind=cbind(X1_fwind,X2_fwind,matrix(X3_find,n,ncol(X1_fwind)))

V0_fwind=futuroTC(plazos_fwd_ind,X1_fwind[1,],X2_fwind[1,],X3_find[1,],kst_fwd_ind)*contratos_fwd_ind*no
V0_fwind
```

```
##          [,1]
## [1,] 342308.9
```

SWAPS

```
##SWAP TASA FIJA VS TASA VARIABLE CÁLCULO

##Interpolamos

nodosvp=nodos1_sw
nodostc=nodos2_sw
curvavp=as.matrix(X1_orig_sw)
curvatc=X2_orig_sw
n1=nrow(curvavp)
n2=nrow(curvatc)


m=max(ncol(plazos_sw),1) #número de contratos_sw swap a valorar
N=matrix(0,1,m) #es un vector de m valores donde se cargarán los m número de cupones a pagar para cada
```

```
for (j in (1:m))
{
  N[j]=as.integer(plazos_sw[j]/plazocupon_sw[j])+1 #número de cupones a pagar
}
VTplazos_sw=matrix(0,1,sum(N)) #vector de todos los plazos_sw de todos los contratos_sw
contratos_swT=matrix(0,1,sum(N)) #vector de todos los contratos_sw de todos los flujos de todos los con
nominal_swT=matrix(0,1,sum(N)) #vector de todos los nominal_swes de todos los flujos de todos los contr
por_swT=matrix(0,1,sum(N)) #vector de todos los dummy si paga o recibe de todos los flujos de todos los
plazocupon_swT=matrix(0,1,sum(N)) #vector de todos los plazos_swcupon de todos los flujos de todos los
tasafija_swT=matrix(0,1,sum(N)) #vector de tasas fijas de todos los flujos de todos los contratos_sw
VTplazos_swc=matrix(0,1,sum(N)) #vector de todos los plazos_sw cortos de todos los contratos_sw


plazini=plazos_sw-plazocupon_sw*(N-1) #vector de plazos_sw iniciales

for (j in (1:m))
{
  if (j==1)
  {
    VTplazos_sw[,1:sum(N[1:j])]=seq(plazini[j],plazos_sw[j], by=plazocupon_sw[j])
    VTplazos_swc[,1:sum(N[1:j])]=c(0,VTplazos_sw[,1:(sum(N[1:j])-1)])
    contratos_swT[,1:sum(N[1:j])]=seq(contratos_sw[j],contratos_sw[j])
    nominal_swT[,1:sum(N[1:j])]=seq(nominal_sw[j],nominal_sw[j])
    por_swT[,1:sum(N[1:j])]=seq(por_sw[j],por_sw[j])
    plazocupon_swT[,1:sum(N[1:j])]=seq(plazocupon_sw[j],plazocupon_sw[j])
    tasafija_swT[,1:sum(N[1:j])]=seq(tasafija_sw[j],tasafija_sw[j])
  }
  else
  {
    VTplazos_sw[,(sum(N[1:j-1])+1):sum(N[1:j])]=seq(plazini[j],plazos_sw[j], by=plazocupon_sw[j])
    VTplazos_swc[,(sum(N[1:j-1])+1):sum(N[1:j])]=c(0,VTplazos_sw[,(sum(N[1:j-1])+1):(sum(N[1:j])-1)])
    contratos_swT[,(sum(N[1:j-1])+1):sum(N[1:j])]=seq(contratos_sw[j],contratos_sw[j])
    nominal_swT[,(sum(N[1:j-1])+1):sum(N[1:j])]=seq(nominal_sw[j],nominal_sw[j])
    por_swT[,(sum(N[1:j-1])+1):sum(N[1:j])]=seq(por_sw[j],por_sw[j])
    plazocupon_swT[,(sum(N[1:j-1])+1):sum(N[1:j])]=seq(plazocupon_sw[j],plazocupon_sw[j])
    tasafija_swT[,(sum(N[1:j-1])+1):sum(N[1:j])]=seq(tasafija_sw[j],tasafija_sw[j])
  }
}


Xvp=matrix(0,n,ncol(VTplazos_sw))
Xtc=matrix(0,n,ncol(VTplazos_sw))
Xtcc=matrix(0,n,ncol(VTplazos_sw))
XtfwdT=matrix(0,n,ncol(VTplazos_sw))

for (i in (1:n))
{
  Xvp[i,]=if(itpl==0){approx(nodosvp,curvavp[i,],VTplazos_sw,rule=2)$y}else{talamb(nodosvp,curvavp[i,],V
  Xtc[i,]=if(itpl==0){approx(nodostc,curvatc[i,],VTplazos_sw,rule=2)$y}else{talamb(nodostc,curvatc[i,],V
  Xtcc[i,]=if(itpl==0){approx(nodostc,curvatc[i,],VTplazos_swc, rule=2)$y}else{talamb(nodostc,curvatc[i

  XtfwdT[i,]=((1+Xtc[i,]*VTplazos_sw/360)/(1+Xtcc[i,]*VTplazos_swc/360)-1)*360/plazocupon_swT
  for (j in (1:ncol(VTplazos_sw)))
```

```
  {
    if (VTplazos_sw[j]<= plazocupon_swT[j])
    {
      XtfwdT[i,j]=Xtc[i,j]
    }
    else
    {
      j=sum(N[1:j])
    }
  }
}

X_sw=cbind(XtfwdT,Xvp)

swap=function(por_swT, contratos_swT, nominal_swT, XtfwdT, tasafija_swT, plazocupon_swT, VTplazos_sw, X
{
  V0=matrix(0,1,ncol(N))
  V0f=(((contratos_swT*(XtfwdT-tasafija_swT)*(plazocupon_swT/360)))/(1+Xvp*VTplazos_sw/360))*nominal_sw
  for (j in (1:ncol(N)))
  {
    if(j==1)
    {
      V0[j]=sum(V0f[j:N[j]])
    }
    else
    {
      V0[j]=sum(V0f[(sum(N[1:j-1])+1):(sum(N[1:j]))])
    }
  }
  V0
}

V0_sw=swap(por_swT, contratos_swT, nominal_swT, XtfwdT[1,], tasafija_swT, plazocupon_swT, VTplazos_sw, 
V0_sw
```

```
##          [,1]    [,2]
## [1,] -20224.6 4740583
```

Opciones

```
##opciones de tasa de interés, con inicio el día de la valuación CÁLCULO
#Posición inicial

#interpolación de tasas y volatilidades
m=ncol(plazos_oir)
x1=matrix(0,n,m)
x2tc=matrix(0,n,m)
x2tl=matrix(0,n,m)
x2=matrix(0,n,m)
x3=matrix(0,n,m)
for (i in 1:(n))
{
```

```r
  x1[i,]=if(itpl==0){approx(nodos1_oir,x1_orig_oir[i,],plazos_oir,rule=2)$y}else{talamb(nodos1_oir,x1_oi
  x2tc[i,]=if(itpl==0){approx(nodos2_oir,x2_orig_oir[i,],plazos_oir,rule=2)$y}else{talamb(nodos2_oir,x2
  x2tl[i,]=if(itpl==0){approx(nodos2_oir,x2_orig_oir[i,],(plazos_oir+pr_oir),rule=2)$y}else{talamb(nodos
  x3[i,]=if(itpl==0){approx(nodos3_oir,x3_orig_oir[i,],plazos_oir,rule=2)$y}else{talamb(nodos3_oir,x3_oi
  x2[i,]=((1+x2tl[i,]*(plazos_oir+pr_oir)/360)/(1+x2tc[i,]*(plazos_oir)/360)-1)*360/pr_oir
}


x01=x1[1,] #tasas de descuento
x02=x2[1,] #tasas spot
x03=x3[1,] #volatilidades

X_oir=cbind(x1,x2,x3)

opctint = function(d,S,K_oir,vol,t,cp_oir,cs_oir,pr_oir,dct_oir)    #función de una opción europea
{
  d1=if(cs_oir==1){(log(S/K_oir)+vol^2*t/(365*2))*(1/(vol*sqrt(t/365)))}else{(log(S/K_oir)+vol^2*t/(360
  d2=if(cs_oir==1){(log(S/K_oir)-vol^2*t/(365*2))*(1/(vol*sqrt(t/365)))}else{(log(S/K_oir)-vol^2*t/(360
  vp=if(cs_oir==1){log(1+d*t/360)*365/t}else{d}
  (if(cs_oir==1){(S*pnorm(d1*(-1)^cp_oir)-K_oir*pnorm(d2*(-1)^cp_oir))*(exp(-vp*t/365))*(-1)^cp_oir}else
}


V0_oir=opctint(x01,x02,K_oir,x03,plazos_oir,cp_oir,cs_oir,pr_oir,dct_oir)*contratos_oir*nominal_oir #Va
V0_oir
```

```
##          [,1]      [,2]
## [1,] 1.002318 0.2479957
```

Integración de factores y cálculo de riesgo en conjunto, y aplicación de simulación

Un enfoque más claro es suponer que tenemos:

1. Una matriz $X_{(n+1)\times m}$ de m factores de riesgo y $n+1$ observaciones.
2. Denotemos el vector de precios actual como $X_{00} := (x_{0,1}, x_{0,2}, \ldots, x_{0,m})$.
3. Sea r el número de instrumentos de un portafolio, entonces cada instrumento tiene una función de valuación $f_i \colon A_i \to R$ para todo $x \in X$, $i = 1, \ldots, r$, donde $A_i \subset X_i$ con $\#(A_i) \leq \#(X_i)$.
4. Sea r el número de instrumentos de un portafolio, entonces cada instrumento tiene una función de valuación $f_i \colon A_i \to R$ para todo $x \in X$, $i = 1, \ldots, r$, donde $A_i \subset X_i$ con $\#(A_i) \leq \#(X_i)$.
5. Sea $M_{1\times r} = (m_1, \ldots, m_r)$ el vector de posiciones nominales de cada instrumento, es decir, el número de contratos que se tienen por instrumento $m_i \in R$ $(i = 1, \ldots, r)$.

```r
#DIMENSION DE TODOS LOS INSTRUMENTOS
#Son 8 instrumentos financieros (9 si separamos acciones y divisas)
n_if=matrix(0,6,1)
n_if[1]=ncol(stock_prices_EQFX) #acciones y divisas
n_if[2]=ncol(X_bd_ext) #bonde
n_if[3]=ncol(X_sw) #swaps
n_if[4]=ncol(X_oir) #opciones tasa de interés
```

```r
n_if[5]=ncol(X_futtdc) #Forwards de tipo de cambio
n_if[6]=ncol(X_futind) #Forwards de índices
```

```r
#valor del portafolios

V0_port=cbind(V0_acc_div, V0_bd, V0_sw, V0_oir,V0_fwtdc,V0_fwind)
V0T_port=sum(V0_port)



#INTEGRACIÓN DE TODOS LOS FACTORES DE RIESGO EN UNA MATRIZ
X_port=cbind(stock_prices_EQFX,X_bd_ext,X_sw,X_oir,X_futtdc,X_futind) #Factores de riesgo del portafoli
```

```r
#Cálculo de variaciones Delta_X DEL PORTAFOLIOS
DeltaX_port=as.matrix(log(X_port[1:(n-1)]/X_port[2:(n)]))
DeltaX_port[is.nan(DeltaX_port)] <- 0 #quitamos NaN
DeltaX_port[is.na(DeltaX_port)] <- 0 #quitamos Na
DeltaX_port[is.infinite(DeltaX_port)] <- 0 #quitamos Na
```

```r
Ns=nrow(DeltaX_port) #Definimos número de escenarios históricos
#alpha=0.98 #Nivel de Confianza para las medidas de riesgo

DeltaX_s=DeltaX_port
#print(head(DeltaX_s))
print(ncol(DeltaX_s))
```

```
## [1] 547
```

```r
print(nrow(DeltaX_s))
```

```
## [1] 252
```

```r
print(n_if)
```

```
##      [,1]
## [1,]    6
## [2,]  465
## [3,]   64
## [4,]    6
## [5,]    3
## [6,]    3
```

MEDIDAS DE RIESGO CON ALISADO

A continuacion se definen las medidas de riesgo con alisado:

```r
n=n-1
```

```r
#Medidas de riesgo CON alisado

#Se necesita definir
#1) El valor del peso inicial del primer escenario "w0"
#2) La función de cuantil con vector de probabilidades no iguales
#3) La función de CVaR con probabilidades no iguales

#w0=0.05

#Creación de dos funciones que sirven para este fin
# Percentil con pesos de probabilidades
#
# v  un vector de observaciones
# w Un vector numérico de valores positivos, en general es la distrubición.
# p  el valor de la probabilidad entre 0 y 1.
#
# Esta función no interpola

wquantile <- function(v,w=rep(1,length(v)),p=.5)
{
  if ( !is.numeric(w) || length(v) != length(w) )
    stop("Los valores y los pesos tienen que tener misma longitud")
  if ( !is.numeric(p) || any( p<0 | p>1) )
    stop("Percentil tiene que ser 0<=p<=1")
  if ( min(w) < 0 ) stop("Los pesos tiene que ser mayores que 0")
  ranking <- order(v)
  sumw <- cumsum(w[ranking])
  plist <- sumw / sumw[ length(sumw) ]
  v [ ranking [ which.max( plist >= p ) ] ]
}

#CVaR con alisado
wcvar <- function(v,w=rep(1,length(v)),p=.5)
{
  if ( !is.numeric(w) || length(v) != length(w) )
    stop("Los valores y los pesos tienen que tener misma longitud")
  if ( !is.numeric(p) || any( p<0 | p>1) )
    stop("Percentil tiene que ser 0<=p<=1")
  if ( min(w) < 0 ) stop("Los pesos tiene que ser mayores que 0")
  ranking <- order(v)
  sumw <- cumsum(w[ranking])
  plist <- sumw / sumw[ length(sumw) ]
  loss= v [ ranking [ which( plist < p ) ] ]
  esc=w [ ranking [ which( plist < p ) ] ]
  sum(loss*esc)/(sum(esc))
}


#esc_cvar=which(cumsum(p_esc[order(PLT[,1])])<pdca)


#p_esc[esc_cvar]
```

```r
#tshs=cbind(PLT,p_esc)

w0=0.05
lambda =uniroot(function(x) w0*(1-x^(n))/(1-x)-1, c(0,0.99), tol = 1e-28)$root
lambda
```

```
## [1] 0.9500001
```

```r
#generamos la función que genera "n" escenarios con base en w0 y lambda
genera_esc=function(lamda,w0,n)
{
  p_esc=matrix(0,n,1)
  for (i in (1:n))
  {
    p_esc[i]=w0*lambda^(i-1)
  }
  p_esc
}


p_esc=genera_esc(lambda,w0,n)
print(t(p_esc[1:6]))
```

```
##      [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## [1,] 0.05 0.04750001 0.04512501 0.04286877 0.04072533 0.03868907
```

```r
sum(p_esc) #validamos que sume 1
```

```
## [1] 1
```

Medición de Riesgo

Acciones y divisas

Paso 1. (Generación de rendimientos) Construir $\Delta X_{n\times m}$ que es la matriz de diferencias basados en el operador T_j, es decir

```
$${\Delta X}_t=\Big[ T_j(\frac{x_{t,1}}{x_{t+1,1}}), T_j(\frac{x_{t,2}}{x_{t+1,2}}),\ldots, T_j(\fra
```

```r
#Medición de riesgo por instrumento, instrumento-factor de riesgo, instrumento - total


#Cálculo de matriz de pérdidas y ganancias Acciones y Divisas

#riesgo del acciones y divisas
m=m_fx+m_acc    #PASO CLAVE
X_s_acc_div=matrix(0,Ns,n_if[1]) #Factores de riesgo simulados con base en DeltaX_s x0*(1+Delta_Xs)   #
V_acc_div=matrix(0,Ns,m)
Vfr1_acc_div=matrix(0,Ns,m_fx)
```

```r
Vfr2_acc_div=matrix(0,Ns,m_acc)
PG_acc_div=matrix(0,Ns,m) #Pèrdidas y ganancias
PGfr1_acc_div=matrix(0,Ns,m_fx)
PGfr2_acc_div=matrix(0,Ns,m_acc)
PGT_acc_div=matrix(0,Ns,1)
PGfr1T_acc_div=matrix(0,Ns,1)
PGfr2T_acc_div=matrix(0,Ns,1)

DeltaX_s_acc_div=DeltaX_s[,(1:n_if[1])]    #PASO CLAVE
x0_acc_div=stock_prices_EQFX[1,]   #PASO CLAVE

for (i in 1:Ns)
{
  X_s_acc_div[i,]=as.matrix(x0_acc_div*exp(DeltaX_s_acc_div[i,]))
  #PASO CLAVE
  V_acc_div[i,]=cbind(t(pos_fx),t(pos_eq))*X_s_acc_div[i,]
  #PASO CLAVE
  Vfr1_acc_div[i,]=t(pos_fx)*X_s_acc_div[i,1:m_fx]
  #PASO CLAVE
  Vfr2_acc_div[i,]=t(pos_eq)*X_s_acc_div[i,(m_fx+1):(m_fx+m_acc)]
  #PASO CLAVE
  PG_acc_div[i,]=as.matrix(V_acc_div[i,]-V0_acc_div)
  PGfr1_acc_div[i,]=as.matrix(Vfr1_acc_div[i,]-V0_acc_div[,1:m_fx])
  PGfr2_acc_div[i,]=as.matrix(Vfr2_acc_div[i,]-V0_acc_div[,(m_fx+1):(m_fx+m_acc)])
  PGT_acc_div[i,]=sum(PG_acc_div[i,])
  PGfr1T_acc_div[i,]=sum(PGfr1_acc_div[i,])
  PGfr2T_acc_div[i,]=sum(PGfr2_acc_div[i,])
}



PG_acc_div[1:5,]
```

```
##            [,1]      [,2]       [,3]      [,4]        [,5]      [,6]
## [1,]   39.92302 109.73066 -187.29628 -250.5988  -752.22702  43717.57
## [2,]   64.20496  38.94973   91.35464  100.1348 -1474.28282  12739.07
## [3,]  -66.84335  51.37171  -53.95288 1383.3383    38.81845  88783.36
## [4,] -118.31490 136.45676 -237.40898 -446.5567   155.59457 -29121.38
## [5,]  -67.69231 106.55138  -67.30493 -900.8683  1895.36219   8695.61
```

```r
PGfr1_acc_div[1:5,]
```

```
##            [,1]      [,2]       [,3]
## [1,]   39.92302 109.73066 -187.29628
## [2,]   64.20496  38.94973   91.35464
## [3,]  -66.84335  51.37171  -53.95288
## [4,] -118.31490 136.45676 -237.40898
## [5,]  -67.69231 106.55138  -67.30493
```

```r
PGfr2_acc_div[1:5,]
```

```
##           [,1]        [,2]      [,3]
```

```
## [1,] -250.5988  -752.22702  43717.57
## [2,]  100.1348 -1474.28282  12739.07
## [3,] 1383.3383    38.81845  88783.36
## [4,] -446.5567   155.59457 -29121.38
## [5,] -900.8683  1895.36219   8695.61
```

```
PGT_acc_div[1:5,]
```

```
## [1]  42677.104  11559.428  90136.094 -29631.606   9661.658
```

Necesitamos la materia prima con la cual trabajaremos, "fr" indica el numero de factor de riesgo, se hace para acciones y divisas "acc_div" con o sin alisado "CA" y VaR o CVaR respectivamente.

```
#VaR por posición
VaRCont_acc_div=matrix(0,1,m)
VaRfr1_acc_div=matrix(0,1,m_fx)
VaRfr2_acc_div=matrix(0,1,m_acc)
CVaRCont_acc_div=matrix(0,1,m)
CVaRfr1_acc_div=matrix(0,1,m_fx)
CVaRfr2_acc_div=matrix(0,1,m_acc)

VaRCont_CA_acc_div=matrix(0,1,m)
VaRfr1_CA_acc_div=matrix(0,1,m_fx)
VaRfr2_CA_acc_div=matrix(0,1,m_acc)
CVaRCont_CA_acc_div=matrix(0,1,m)
CVaRfr1_CA_acc_div=matrix(0,1,m_fx)
CVaRfr2_CA_acc_div=matrix(0,1,m_acc)
```

Hacemos el llenado de las matrices anteriores por pedazos, primero calculamos sobre cada columna que representa un instrumento, el VaR y el CVar en primer lugar tomando el cuantil sobre todos los insrtumentos en el portafolio general (PG_acc_div) y despues tomandolo para las divisas sobre el portafolio de divisas (PGfr1_acc_div), y despues sobre las acciones sobre el portafolio de acciones (PGfr2_acc_div).(Aqui es medida por instrumento).

```
for (i in (1:m))
{
  VaRCont_acc_div[i]=equantile(PG_acc_div[,i],1-alpha,Ns)
  CVaRCont_acc_div[i]= mean(merge(which(PG_acc_div[,i]<VaRCont_acc_div[i]),cbind(seq(1,Ns),PG_acc_div[,i
  VaRCont_CA_acc_div[i]=wquantile(PG_acc_div[,i],p_esc, 1-alpha)
  CVaRCont_CA_acc_div[i]= wcvar(PG_acc_div[,i],p_esc, 1-alpha)

  if (i<=m_fx)
  {
  VaRfr1_acc_div[i]=equantile(PGfr1_acc_div[,i],1-alpha,Ns)
  CVaRfr1_acc_div[i]= mean(merge(which(PGfr1_acc_div[,i]<VaRfr1_acc_div[i]),cbind(seq(1,Ns),PGfr1_acc_di
  }
  if (i<=m_acc)
  {
  VaRfr2_acc_div[i]=equantile(PGfr2_acc_div[,i],1-alpha,Ns)
  CVaRfr2_acc_div[i]= mean(merge(which(PGfr2_acc_div[,i]<VaRfr2_acc_div[i]),cbind(seq(1,Ns),PGfr2_acc_di
  }
}
#IMPRIMIMOS LOS RESULTADOS ARRIBA BUSCADOS
VaRCont_acc_div
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] -149.0083 -160.1994 -317.2778 -2940.283 -2955.189 -73364.64
```

```r
VaRfr1_acc_div
```

```
##            [,1]      [,2]      [,3]
## [1,] -149.0083 -160.1994 -317.2778
```

```r
VaRfr2_acc_div
```

```
##            [,1]      [,2]      [,3]
## [1,] -2940.283 -2955.189 -73364.64
```

```r
CVaRCont_acc_div
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] -189.9811 -179.9433 -442.8901 -4142.288 -3435.131 -95595.48
```

```r
CVaRfr1_acc_div
```

```
##            [,1]      [,2]      [,3]
## [1,] -189.9811 -179.9433 -442.8901
```

```r
CVaRfr2_acc_div
```

```
##            [,1]      [,2]      [,3]
## [1,] -4142.288 -3435.131 -95595.48
```

Ya que tenemos los Vares y CVares por instrumento, los calculamos en total.

```r
#VaR Total

VaRTotal_acc_div=equantile(PGT_acc_div,1-alpha,Ns)
CVaRTotal_acc_div= mean(merge(which(PGT_acc_div<VaRTotal_acc_div),cbind(seq(1,Ns),PGT_acc_div), by.x=1,b
#VaR y CVar total para divisas
VaRTotalfr1_acc_div=equantile(PGfr1T_acc_div,1-alpha,Ns)
CVaRTotalfr1_acc_div= mean(PGfr1T_acc_div[which(PGfr1T_acc_div<VaRTotalfr1_acc_div),])
#VaR y CVar total para acciones
VaRTotalfr2_acc_div=equantile(PGfr2T_acc_div,1-alpha,Ns)
CVaRTotalfr2_acc_div= mean(PGfr2T_acc_div[which(PGfr2T_acc_div<VaRTotalfr2_acc_div),])

#IMPPRIMIMOS LOS VALORES ARRIBA CALCULADOS
print(cbind(VaRTotal_acc_div,sum(V0_acc_div), VaRCont_acc_div, V0_acc_div))
```

```
##    VaRTotal_acc_div      V2        V1        V2        V3        V4        V5
## 1:        -73799.24 3564744 -149.0083 -160.1994 -317.2778 -2940.283 -2955.189
##           V6 EURUSD.X  GBPUSD.X USDMXN.X AMXL.MX GCARSOA1.MX  WMT.MX
## 1: -73364.64 15501.42 -15685.53  29948.4 -103750       73530 3565200
```

```r
cbind(CVaRTotal_acc_div,sum(V0_acc_div), CVaRCont_acc_div, V0_acc_div)
```

```
##    CVaRTotal_acc_div      V2        V1        V2        V3        V4        V5
## 1:         -98138.85 3564744 -189.9811 -179.9433 -442.8901 -4142.288 -3435.131
##           V6 EURUSD.X  GBPUSD.X USDMXN.X AMXL.MX GCARSOA1.MX  WMT.MX
## 1: -95595.48 15501.42 -15685.53  29948.4 -103750       73530 3565200
```

```r
cbind(VaRTotal_acc_div,VaRTotalfr1_acc_div,VaRTotalfr2_acc_div)
```

```
##      VaRTotal_acc_div VaRTotalfr1_acc_div VaRTotalfr2_acc_div
## [1,]        -73799.24           -357.8263           -73942.81
```

```r
cbind(CVaRTotal_acc_div,CVaRTotalfr1_acc_div,CVaRTotalfr2_acc_div)
```

```
##      CVaRTotal_acc_div CVaRTotalfr1_acc_div CVaRTotalfr2_acc_div
## [1,]         -98138.85             -423.176            -98098.46
```

```r
length(PG_acc_div[,1])
```

```
## [1] 252
```

medidas con alisado

```r
#VaR Total
#SINTAXIS WQUANTILE: (v,w=rep(1,length(v)),p=.5)
VaRTotal_CA_acc_div=wquantile(v = PGT_acc_div, w=rep(1,length(PGT_acc_div)),1-alpha)
CVaRTotal_CA_acc_div= mean(merge(which(PGT_acc_div<VaRTotal_CA_acc_div),cbind(seq(1,Ns),PGT_acc_div), by
#VaR y CVar total para divisas
VaRTotalfr1_CA_acc_div=wquantile(PGfr1T_acc_div,w=rep(1,length(PGfr1T_acc_div)),1-alpha)
CVaRTotalfr1_CA_acc_div= mean(PGfr1T_acc_div[which(PGfr1T_acc_div<VaRTotalfr1_CA_acc_div),])
#VaR y CVar total para acciones
VaRTotalfr2_CA_acc_div=wquantile(PGfr2T_acc_div,w=rep(1,length(PGfr2T_acc_div)),1-alpha)
CVaRTotalfr2_CA_acc_div= mean(PGfr2T_acc_div[which(PGfr2T_acc_div<VaRTotalfr2_CA_acc_div),])

#IMPPRIMIMOS LOS VALORES ARRIBA CALCULADOS
print(cbind(VaRTotal_CA_acc_div,sum(V0_acc_div), VaRCont_CA_acc_div, V0_acc_div))
```

```
##    VaRTotal_CA_acc_div      V2        V1        V2      V3        V4        V5
## 1:            -85620.4 3564744 -214.0865 -130.5663 -640.96 -3426.218 -2592.736
##           V6 EURUSD.X  GBPUSD.X USDMXN.X AMXL.MX GCARSOA1.MX  WMT.MX
## 1: -124035.1 15501.42 -15685.53  29948.4 -103750       73530 3565200
```

```r
cbind(CVaRTotal_CA_acc_div,sum(V0_acc_div), CVaRCont_CA_acc_div, V0_acc_div)
```

```
##    CVaRTotal_CA_acc_div      V2        V1        V2  V3        V4        V5  V6
## 1:            -108414.3 3564744 -271.4772 -165.6511 NaN -3686.195 -3621.061 NaN
##    EURUSD.X  GBPUSD.X USDMXN.X AMXL.MX GCARSOA1.MX  WMT.MX
## 1: 15501.42 -15685.53  29948.4 -103750       73530 3565200
```

```r
cbind(VaRTotal_CA_acc_div,VaRTotalfr1_CA_acc_div,VaRTotalfr2_CA_acc_div)
```

```
##      VaRTotal_CA_acc_div VaRTotalfr1_CA_acc_div VaRTotalfr2_CA_acc_div
## [1,]            -85620.4              -357.8263              -85632.63
```

```r
cbind(CVaRTotal_CA_acc_div,CVaRTotalfr1_CA_acc_div,CVaRTotalfr2_CA_acc_div)
```

```
##      CVaRTotal_CA_acc_div CVaRTotalfr1_CA_acc_div CVaRTotalfr2_CA_acc_div
## [1,]            -108414.3                -423.176               -108389.6
```

Riesgo de Bondes D

PARA COMENZAR NECESITAOS NUESTRA MATERIA PRIMA, son tres factores de riesgo la sintaxis es la misma que antes "bd" expresa bondes.

En la siguiente ecuación se expresa de manera general un bono con dos factores de riesgo, aunque en este caso incluye el efecto de una sobretasa or lo que son 3 factores de riesgo.

$$V = \sum_{i=1}^{n} \frac{N \cdot C \cdot t_{c_{p_i}} \cdot p_c/360}{(1 + t_{vp_{p_i}} \cdot p_i/360)} + \frac{N \cdot C}{(1 + t_{vp_{p_n}} \cdot p_n/360)}$$

Donde: \ N: Valor Nominal del bono
C: Número de contratos
p_c: Plazo fijo para cada pago de intereses del cupón.
p_i: Plazo acumulado (en días) al cupón i.
$t_{c_{p_i}}$: Tasa cupón variable, se obtiene de la curva subyacente que le corresponda, casi siempre con la tasa forward entre p_{i-1} y p_i.
$t_{vp_{p_i}}$: Tasa valor presente que depende de la curva de bonos según el plazo acumulado al pago del cupón i.

En general tenemos dos factores de riesgo subyacente (la curva de valor presente y la curva de cupones) pero como cada cupón tiene "n" flujos entonces tiene n factores de riesgo para los valores presentes y n factores de riesgo para los cupones, por lo que tienen $2n$ factores de riesgo específicos que provienen de dos factores de riesgo subyacentes. Para el caso de K bonos cupón variable el número de factores de riesgo sería $2\sum_{i=1}^{K} n_i$, donde n_i es el número de cupones a pagar del bono i.

```r
options(warn = -1)

#Cálculo de matriz de pérdidas y ganancias BONDES
#dimensión
m=count(N_bd)    #PASO CLAVE
X_s_bd=matrix(0,Ns,n_if[2]) #Factores de riesgo simulados con base en DeltaX_s x0*(1+Delta_Xs)   #PASO
V_bd=matrix(0,Ns,m)
Vfr1_bd=matrix(0,Ns,m)
Vfr2_bd=matrix(0,Ns,m)
Vfr3_bd=matrix(0,Ns,m)
PG_bd=matrix(0,Ns,m) #Pèrdidas y ganancias
PGfr1_bd=matrix(0,Ns,m)
PGfr2_bd=matrix(0,Ns,m)
PGfr3_bd=matrix(0,Ns,m)
PGT_bd=matrix(0,Ns,1)
PGfr1T_bd=matrix(0,Ns,1)
PGfr2T_bd=matrix(0,Ns,1)
```

```r
PGfr3T_bd=matrix(0,Ns,1)

DeltaX_s_bd=DeltaX_s[,sum(n_if[1:1],1):sum(n_if[1:2])]    #PASO CLAVE
x0_bd=X_bd_ext[1,]    #PASO CLAVE
```

```r
options(warn = -1)


for (i in 1:Ns)
{
  X_s_bd[i,]=x0_bd*exp(DeltaX_s_bd[i,])
  #PASO CLAVE
  V_bd[i,]=bondeD(contratos_bdmT, nominal_bdm, X_s_bd[i,1], plazocupon_bdmT, VTplazos_bdm, X_s_bd[i,(n_
  #PASO CLAVE
  Vfr1_bd[i,]=bondeD(contratos_bdmT, nominal_bdm, X_s_bd[i,1], plazocupon_bdmT, VTplazos_bdm, x0_bd[(n_
  #PASO CLAVE
  Vfr2_bd[i,]=bondeD(contratos_bdmT, nominal_bdm, x0_bd[1], plazocupon_bdmT, VTplazos_bdm, X_s_bd[i,(n_
  #PASO CLAVE
  Vfr3_bd[i,]=bondeD(contratos_bdmT, nominal_bdm, x0_bd[1], plazocupon_bdmT, VTplazos_bdm, x0_bd[(n_if[
  #PASO CLAVE
  #Calculo de las diferencias de precios en valuación
  PG_bd[i,]=V_bd[i,]-V0_bd
  PGfr1_bd[i,]=Vfr1_bd[i,]-V0_bd
  PGfr2_bd[i,]=Vfr2_bd[i,]-V0_bd
  PGfr3_bd[i,]=Vfr3_bd[i,]-V0_bd
  PGT_bd[i,]=sum(PG_bd[i,])
  PGfr1T_bd[i,]=sum(PGfr1_bd[i,])
  PGfr2T_bd[i,]=sum(PGfr2_bd[i,])
  PGfr3T_bd[i,]=sum(PGfr3_bd[i,])
}

#Imprimimos los encabezados de las perdidas y ganancias


PG_bd[1:5,]
```

```
##          [,1]      [,2]
## [1,] 28.12959 -29.26294
## [2,] 28.07345 -29.16933
## [3,] 27.90659 -28.88754
## [4,] 31.59011 -34.21014
## [5,] 28.32711 -29.58790
```

```r
PGfr1_bd[1:5,]
```

```
##          [,1]      [,2]
## [1,] 29.30595 -28.72416
## [2,] 28.66269 -28.89946
## [3,] 26.73092 -29.42592
## [4,] 64.39793 -19.16079
## [5,] 31.54453 -28.11410
```

```r
PGfr2_bd[1:5,]
```

```
##          [,1]      [,2]
## [1,] 27.42943 -29.34473
## [2,] 27.72286 -29.21025
## [3,] 28.60545 -28.80615
## [4,] 11.76610 -36.61747
## [5,] 26.41010 -29.81239
```

```r
PGT_bd[1:5,]
```

```
## [1] -1.1333580 -1.0958728 -0.9809548 -2.6200244 -1.2607863
```

Calculamos las medidas de riesgo usanndo cuantiles:

```r
#VaR por posición
VaRCont_bd=matrix(0,1,m)
VaRfr1_bd=matrix(0,1,m)
VaRfr2_bd=matrix(0,1,m)
VaRfr3_bd=matrix(0,1,m)
CVaRCont_bd=matrix(0,1,m)
CVaRfr1_bd=matrix(0,1,m)
CVaRfr2_bd=matrix(0,1,m)
CVaRfr3_bd=matrix(0,1,m)
for (i in (1:m))
{
  VaRCont_bd[i]=equantile(PG_bd[,i],1-alpha,Ns)
  VaRfr1_bd[i]=equantile(PGfr1_bd[,i],1-alpha,Ns)
  VaRfr2_bd[i]=equantile(PGfr2_bd[,i],1-alpha,Ns)
  VaRfr3_bd[i]=equantile(PGfr3_bd[,i],1-alpha,Ns)
  CVaRfr1_bd[i]= mean(merge(which(PGfr1_bd[,i]<VaRfr1_bd[i]),cbind(seq(1,Ns),PGfr1_bd[,i]), by.x=1,by.y
  CVaRfr2_bd[i]= mean(merge(which(PGfr2_bd[,i]<VaRfr2_bd[i]),cbind(seq(1,Ns),PGfr2_bd[,i]), by.x=1,by.y
  CVaRfr3_bd[i]= mean(merge(which(PGfr3_bd[,i]<VaRfr3_bd[i]),cbind(seq(1,Ns),PGfr3_bd[,i]), by.x=1,by.y
  CVaRCont_bd[i]= mean(merge(which(PG_bd[,i]<VaRCont_bd[i]),cbind(seq(1,Ns),PG_bd[,i]), by.x=1,by.y=1)[
}

VaRCont_bd
```

```
##          [,1]      [,2]
## [1,] 27.27791 -32.12856
```

```r
VaRfr1_bd
```

```
##          [,1]     [,2]
## [1,] 19.16281 -31.4884
```

```r
VaRfr2_bd
```

```
##          [,1]      [,2]
## [1,] 18.39525 -33.51671
```

```
CVaRCont_bd
```

```
##           [,1]    [,2]
## [1,] 27.14754 -33.656
```

```
CVaRfr1_bd
```

```
##           [,1]      [,2]
## [1,] 17.52567 -31.93456
```

```
CVaRfr2_bd
```

```
##           [,1]      [,2]
## [1,] 13.53213 -35.79067
```

```
#VaR Total

VaRTotal_bd=equantile(PGT_bd,1-alpha,Ns)
CVaRTotal_bd= mean(merge(which(PGT_bd<VaRTotal_bd),cbind(seq(1,Ns),PGT_bd), by.x=1,by.y=1)[,2])
VaRTotalfr1_bd=equantile(PGfr1T_bd,1-alpha,Ns)
CVaRTotalfr1_bd= mean(PGfr1T_bd[which(PGfr1T_bd<VaRTotalfr1_bd),])
VaRTotalfr2_bd=equantile(PGfr2T_bd,1-alpha,Ns)
CVaRTotalfr2_bd= mean(PGfr2T_bd[which(PGfr2T_bd<VaRTotalfr2_bd),])
VaRTotalfr3_bd=equantile(PGfr3T_bd,1-alpha,Ns)
CVaRTotalfr3_bd= mean(PGfr3T_bd[which(PGfr2T_bd<VaRTotalfr3_bd),])


print(cbind(VaRTotal_bd,sum(V0_bd), VaRCont_bd, V0_bd))
```

```
##      VaRTotal_bd
## [1,]   -2.113036 1767.702 27.27791 -32.12856 824.78 942.9217
```

```
print(cbind(CVaRTotal_bd,sum(V0_bd), CVaRCont_bd, V0_bd))
```

```
##      CVaRTotal_bd
## [1,]    -2.488888 1767.702 27.14754 -33.656 824.78 942.9217
```

```
print(cbind(VaRTotal_bd,VaRTotalfr1_bd,VaRTotalfr2_bd,VaRTotalfr3_bd))
```

```
##      VaRTotal_bd VaRTotalfr1_bd VaRTotalfr2_bd VaRTotalfr3_bd
## [1,]   -2.113036       -12.3256      -15.12146      -15.12146
```

```
print(cbind(CVaRTotal_bd,CVaRTotalfr1_bd,CVaRTotalfr2_bd,CVaRTotalfr3_bd))
```

```
##      CVaRTotal_bd CVaRTotalfr1_bd CVaRTotalfr2_bd CVaRTotalfr3_bd
## [1,]    -2.488888       -14.40889       -22.25855       -22.25855
```

con alisado

```r
#VaR por posición
VaRCont_CA_bd=matrix(0,1,m)
VaRfr1_CA_bd=matrix(0,1,m)
VaRfr2_CA_bd=matrix(0,1,m)
VaRfr3_CA_bd=matrix(0,1,m)
CVaRCont_CA_bd=matrix(0,1,m)
CVaRfr1_CA_bd=matrix(0,1,m)
CVaRfr2_CA_bd=matrix(0,1,m)
CVaRfr3_CA_bd=matrix(0,1,m)
for (i in (1:m))
{
  VaRCont_CA_bd[i]=wquantile(PG_bd[,i],w=rep(1,length(PG_bd[,i])),1-alpha)
  VaRfr1_CA_bd[i]=wquantile(PGfr1_bd[,i],w=rep(1,length(PGfr1_bd[,i])),1-alpha)
  VaRfr2_CA_bd[i]=wquantile(PGfr2_bd[,i],w=rep(1,length(PGfr2_bd[,i])),1-alpha)
  VaRfr3_CA_bd[i]=wquantile(PGfr3_bd[,i],w=rep(1,length(PGfr3_bd[,i])),1-alpha)
  CVaRfr1_CA_bd[i]= mean(merge(which(PGfr1_bd[,i]<VaRfr1_CA_bd[i]),cbind(seq(1,Ns),PGfr1_bd[,i]), by.x=
  CVaRfr2_CA_bd[i]= mean(merge(which(PGfr2_bd[,i]<VaRfr2_CA_bd[i]),cbind(seq(1,Ns),PGfr2_bd[,i]), by.x=
  CVaRfr3_CA_bd[i]= mean(merge(which(PGfr3_bd[,i]<VaRfr3_CA_bd[i]),cbind(seq(1,Ns),PGfr3_bd[,i]), by.x=
  CVaRCont_CA_bd[i]= mean(merge(which(PG_bd[,i]<VaRCont_CA_bd[i]),cbind(seq(1,Ns),PG_bd[,i]), by.x=1,by
}

VaRCont_CA_bd
```

```
##          [,1]      [,2]
## [1,] 27.32239 -32.19448
```

```r
VaRfr1_CA_bd
```

```
##          [,1]      [,2]
## [1,] 19.71482 -31.33797
```

```r
VaRfr2_CA_bd
```

```
##          [,1]      [,2]
## [1,] 18.18622 -33.61397
```

```r
CVaRCont_CA_bd
```

```
##          [,1]     [,2]
## [1,] 27.17362 -33.9483
```

```r
CVaRfr1_CA_bd
```

```
##         [,1]      [,2]
## [1,] 17.8531 -31.84533
```

```r
CVaRfr2_CA_bd
```

```
##          [,1]      [,2]
## [1,] 12.60131 -36.22601
```

```r
#VaR Total

VaRTotal_CA_bd=wquantile(PGT_bd,w=rep(1,length(PGT_bd)),1-alpha)
CVaRTotal_CA_bd= mean(merge(which(PGT_bd<VaRTotal_CA_bd),cbind(seq(1,Ns),PGT_bd), by.x=1,by.y=1)[,2])
VaRTotalfr1_CA_bd=wquantile(PGfr1T_bd,w=rep(1,length(PGfr1T_bd)),1-alpha)
CVaRTotalfr1_CA_bd= mean(PGfr1T_bd[which(PGfr1T_bd<VaRTotalfr1_CA_bd),])
VaRTotalfr2_CA_bd=wquantile(PGfr2T_bd,w=rep(1,length(PGfr2T_bd)),1-alpha)
CVaRTotalfr2_CA_bd= mean(PGfr2T_bd[which(PGfr2T_bd<VaRTotalfr2_CA_bd),])
VaRTotalfr3_CA_bd=wquantile(PGfr3T_bd,w=rep(1,length(PGfr3T_bd)),1-alpha)
CVaRTotalfr3_CA_bd= mean(PGfr3T_bd[which(PGfr2T_bd<VaRTotalfr3_CA_bd),])


print(cbind(VaRTotal_CA_bd,sum(V0_bd), VaRCont_CA_bd, V0_bd))
```

```
##      VaRTotal_CA_bd
## [1,]      -2.131739 1767.702 27.32239 -32.19448 824.78 942.9217
```

```r
print(cbind(CVaRTotal_CA_bd,sum(V0_bd), CVaRCont_CA_bd, V0_bd))
```

```
##      CVaRTotal_CA_bd
## [1,]       -2.560318 1767.702 27.17362 -33.9483 824.78 942.9217
```

```r
print(cbind(VaRTotal_CA_bd,VaRTotalfr1_bd,VaRTotalfr2_CA_bd,VaRTotalfr3_CA_bd))
```

```
##      VaRTotal_CA_bd VaRTotalfr1_bd VaRTotalfr2_CA_bd VaRTotalfr3_CA_bd
## [1,]      -2.131739       -12.3256         -15.42775         -15.42775
```

```r
print(cbind(CVaRTotal_CA_bd,CVaRTotalfr1_CA_bd,CVaRTotalfr2_CA_bd,CVaRTotalfr3_CA_bd))
```

```
##      CVaRTotal_CA_bd CVaRTotalfr1_CA_bd CVaRTotalfr2_CA_bd CVaRTotalfr3_CA_bd
## [1,]       -2.560318          -13.99223           -23.6247           -23.6247
```

SWAPS

```r
#Cálculo de matriz de pérdidas y ganancias SWAP

#riesgo del swap
m=ncol(N)    #PASO CLAVE
X_s_sw=matrix(0,Ns,n_if[3]) #Factores de riesgo simulados con base en DeltaX_s x0*(1+Delta_Xs)   #PASO 
V_sw=matrix(0,Ns,m)
Vfr1_sw=matrix(0,Ns,m)
Vfr2_sw=matrix(0,Ns,m)
PG_sw=matrix(0,Ns,m) #Pèrdidas y ganancias
PGfr1_sw=matrix(0,Ns,m)
PGfr2_sw=matrix(0,Ns,m)
PGT_sw=matrix(0,Ns,1)
PGfr1T_sw=matrix(0,Ns,1)
PGfr2T_sw=matrix(0,Ns,1)
```

```r
DeltaX_s_sw=DeltaX_s[,sum(n_if[1:2],1):sum(n_if[1:3])]    #PASO CLAVE
x0_sw=as.numeric(c(XtfwdT[1,],Xvp[1,]))   #PASO CLAVE

for (i in 1:Ns)
{
  X_s_sw[i,]=x0_sw*exp(DeltaX_s_sw[i,])
  #PASO CLAVE
  V_sw[i,]=swap(por_swT, contratos_swT, nominal_swT, X_s_sw[i,1:(n_if[3]/2)], tasafija_swT, plazocupon_s
  #PASO CLAVE
  Vfr1_sw[i,]=swap(por_swT, contratos_swT, nominal_swT,X_s_sw[i,1:(n_if[3]/2)], tasafija_swT, plazocupon
  #PASO CLAVE
  Vfr2_sw[i,]=swap(por_swT, contratos_swT, nominal_swT,  XtfwdT[1,], tasafija_swT, plazocupon_swT, VTpla
  #PASO CLAVE
  PG_sw[i,]=V_sw[i,]-V0_sw
  PGfr1_sw[i,]=Vfr1_sw[i,]-V0_sw
  PGfr2_sw[i,]=Vfr2_sw[i,]-V0_sw
  PGT_sw[i,]=sum(PG_sw[i,])
  PGfr1T_sw[i,]=sum(PGfr1_sw[i,])
  PGfr2T_sw[i,]=sum(PGfr2_sw[i,])
}



PG_sw[1:5,]
```

```
##           [,1]       [,2]
## [1,]  5234.720 -1700.8612
## [2,] -8587.917  1389.7878
## [3,]  5188.272 -1481.3663
## [4,]  5903.510  1361.7587
## [5,] -4114.681  -583.7506
```

```r
PGfr1_sw[1:5,]
```

```
##           [,1]       [,2]
## [1,]  5219.744 -1091.8939
## [2,] -8565.571   744.9176
## [3,]  5175.147 -1057.2400
## [4,]  5908.546  1125.9226
## [5,] -4117.687  -482.8704
```

```r
PGfr2_sw[1:5,]
```

```
##            [,1]      [,2]
## [1,]  16.806096 -609.0483
## [2,] -18.791323  644.4915
## [3,]  14.686497 -424.1779
## [4,]  -6.275911  235.7808
## [5,]   2.664725 -100.8904
```

```r
PGT_sw[1:5,]
```

```
## [1]  3533.859 -7198.129  3706.905  7265.269 -4698.432
```

```r
#VaR por posición
VaRCont_sw=matrix(0,1,m)
VaRfr1_sw=matrix(0,1,m)
VaRfr2_sw=matrix(0,1,m)
CVaRCont_sw=matrix(0,1,m)
CVaRfr1_sw=matrix(0,1,m)
CVaRfr2_sw=matrix(0,1,m)
for (i in (1:m))
{
  VaRCont_sw[i]=equantile(PG_sw[,i],1-alpha,Ns)
  VaRfr1_sw[i]=equantile(PGfr1_sw[,i],1-alpha,Ns)
  VaRfr2_sw[i]=equantile(PGfr2_sw[,i],1-alpha,Ns)
  CVaRfr1_sw[i]= mean(merge(which(PGfr1_sw[,i]<VaRfr1_sw[i]),cbind(seq(1,Ns),PGfr1_sw[,i]), by.x=1,by.y
  CVaRfr2_sw[i]= mean(merge(which(PGfr2_sw[,i]<VaRfr2_sw[i]),cbind(seq(1,Ns),PGfr2_sw[,i]), by.x=1,by.y
  CVaRCont_sw[i]= mean(merge(which(PG_sw[,i]<VaRCont_sw[i]),cbind(seq(1,Ns),PG_sw[,i]), by.x=1,by.y=1)[
}
VaRCont_sw
```

```
##          [,1]      [,2]
## [1,] -33460.6 -30888.66
```

```r
VaRfr1_sw
```

```
##           [,1]      [,2]
## [1,] -33363.85 -25297.07
```

```r
VaRfr2_sw
```

```
##           [,1]      [,2]
## [1,] -63.35926 -5620.679
```

```r
CVaRCont_sw
```

```
##           [,1]      [,2]
## [1,] -46984.21 -49499.26
```

```r
CVaRfr1_sw
```

```
##           [,1]      [,2]
## [1,] -46922.08 -41114.72
```

```r
CVaRfr2_sw
```

```
##           [,1]      [,2]
## [1,] -112.1322 -8463.489
```

```
#VaR Total

VaRTotal_sw=equantile(PGT_sw,1-alpha,Ns)
CVaRTotal_sw= mean(merge(which(PGT_sw<VaRTotal_sw),cbind(seq(1,Ns),PGT_sw), by.x=1,by.y=1)[,2])
VaRTotalfr1_sw=equantile(PGfr1T_sw,1-alpha,Ns)
CVaRTotalfr1_sw= mean(PGfr1T_sw[which(PGfr1T_sw<VaRTotalfr1_sw),])
VaRTotalfr2_sw=equantile(PGfr2T_sw,1-alpha,Ns)
CVaRTotalfr2_sw= mean(PGfr2T_sw[which(PGfr2T_sw<VaRTotalfr2_sw),])


print(cbind(VaRTotal_sw,sum(V0_sw), VaRCont_sw, V0_sw))
```

```
##      VaRTotal_sw
## [1,]   -20442.74 4720358 -33460.6 -30888.66 -20224.6 4740583
```

```
print(cbind(CVaRTotal_sw,sum(V0_sw), CVaRCont_sw, V0_sw))
```

```
##      CVaRTotal_sw
## [1,]    -33935.66 4720358 -46984.21 -49499.26 -20224.6 4740583
```

```
print(cbind(VaRTotal_sw,VaRTotalfr1_sw,VaRTotalfr2_sw))
```

```
##      VaRTotal_sw VaRTotalfr1_sw VaRTotalfr2_sw
## [1,]   -20442.74      -21199.09      -5471.368
```

```
print(cbind(CVaRTotal_sw,CVaRTotalfr1_sw,CVaRTotalfr2_sw))
```

```
##      CVaRTotal_sw CVaRTotalfr1_sw CVaRTotalfr2_sw
## [1,]    -33935.66       -34962.25       -8244.764
```

con alisado

```
#VaR por posición
VaRCont_CA_sw=matrix(0,1,m)
VaRfr1_CA_sw=matrix(0,1,m)
VaRfr2_CA_sw=matrix(0,1,m)
CVaRCont_CA_sw=matrix(0,1,m)
CVaRfr1_CA_sw=matrix(0,1,m)
CVaRfr2_CA_sw=matrix(0,1,m)
for (i in (1:m))
{
  VaRCont_CA_sw[i]=wquantile(PG_sw[,i],w=rep(1,length(PG_sw[,i])),1-alpha)
  VaRfr1_CA_sw[i]=wquantile(PGfr1_sw[,i],w=rep(1,length(PGfr1_sw[,i])),1-alpha)
  VaRfr2_CA_sw[i]=wquantile(PGfr2_sw[,i],w=rep(1,length(PGfr2_sw[,i])),1-alpha)
  CVaRfr1_CA_sw[i]= mean(merge(which(PGfr1_sw[,i]<VaRfr1_CA_sw[i]),cbind(seq(1,Ns),PGfr1_sw[,i]), by.x=
  CVaRfr2_CA_sw[i]= mean(merge(which(PGfr2_sw[,i]<VaRfr2_CA_sw[i]),cbind(seq(1,Ns),PGfr2_sw[,i]), by.x=
  CVaRCont_CA_sw[i]= mean(merge(which(PG_sw[,i]<VaRCont_CA_sw[i]),cbind(seq(1,Ns),PG_sw[,i]), by.x=1,by
}
VaRCont_CA_sw
```

```
##           [,1]      [,2]
## [1,] -25983.79 -30888.66
```

```r
VaRfr1_CA_sw
```

```
##           [,1]      [,2]
## [1,] -25937.82 -25297.07
```

```r
VaRfr2_CA_sw
```

```
##           [,1]      [,2]
## [1,] -56.31597 -5620.679
```

```r
CVaRCont_CA_sw
```

```
##           [,1]      [,2]
## [1,] -40679.87 -49499.26
```

```r
CVaRfr1_CA_sw
```

```
##           [,1]      [,2]
## [1,] -40606.75 -41114.72
```

```r
CVaRfr2_CA_sw
```

```
##           [,1]      [,2]
## [1,] -91.52492 -8463.489
```

```r
#VaR Total

VaRTotal_CA_sw=wquantile(PGT_sw,w=rep(1,length(PGT_sw)),1-alpha)
CVaRTotal_CA_sw= mean(merge(which(PGT_sw<VaRTotal_CA_sw),cbind(seq(1,Ns),PGT_sw), by.x=1,by.y=1)[,2])
VaRTotalfr1_CA_sw=wquantile(PGfr1T_sw,w=rep(1,length(PGfr1T_sw)),1-alpha)
CVaRTotalfr1_CA_sw= mean(PGfr1T_sw[which(PGfr1T_sw<VaRTotalfr1_CA_sw),])
VaRTotalfr2_CA_sw=wquantile(PGfr2T_sw,w=rep(1,length(PGfr2T_sw)),1-alpha)
CVaRTotalfr2_CA_sw= mean(PGfr2T_sw[which(PGfr2T_sw<VaRTotalfr2_CA_sw),])


print(cbind(VaRTotal_CA_sw,sum(V0_sw), VaRCont_CA_sw, V0_sw))
```

```
##      VaRTotal_CA_sw
## [1,]      -18261.76 4720358 -25983.79 -30888.66 -20224.6 4740583
```

```r
print(cbind(CVaRTotal_CA_sw,sum(V0_sw), CVaRCont_CA_sw, V0_sw))
```

```
##      CVaRTotal_CA_sw
## [1,]       -28319.83 4720358 -40679.87 -49499.26 -20224.6 4740583
```

```r
print(cbind(VaRTotal_CA_sw,VaRTotalfr1_CA_sw,VaRTotalfr2_CA_sw))
```

```
##      VaRTotal_CA_sw VaRTotalfr1_CA_sw VaRTotalfr2_CA_sw
## [1,]      -18261.76         -19486.16         -5471.368
```

```r
print(cbind(CVaRTotal_CA_sw,CVaRTotalfr1_CA_sw,CVaRTotalfr2_CA_sw))
```

```
##      CVaRTotal_CA_sw CVaRTotalfr1_CA_sw CVaRTotalfr2_CA_sw
## [1,]       -28319.83          -29145.36          -8244.764
```

OPCIONES TASA DE INTERES

La formula en la que se basa la valuación es el modelo para opciones europeas de Black and Scholes que asume que las trayectorias de variación del activo subyacente son normales.

El valor de la opción que depende del subyacente adquiere la forma de un movimiento Browniano Geometrico:

$$S(t) = x_0 exp[(\mu - \frac{\sigma^2}{2})t + \sigma B_t]$$

donde $B(t)$ es un movimiento Browniano estandar, es decir, un proceso con incrementos independientes,trayectorias continuas e incrementos normales con media 0 y estacionarios con parametro de volatilidad $\sigma^2 = 1$.

Este proceso es solución de una ecuacion diferencial estocastica que expresa que la variación entre el valor final del activo y el inicial es igual al promedio del activo subyacente, ponderado de manera uniforme por una fuerza de interes mas el promedio del activo subyecente ponderado por una trayectoria Browniana multiplicado por una constante σ lo cual constituye un ruido aleatorio.

Resolviendo esta ecuación obtenemos $S(t)$ y calculando la esperanza de la parte positiva del valor del activo menos el precio de ejercicio o Strike K de la opción a plazo de ejercicio t, llegamos a la famosa formula de Black and Scholes:

$$\mathrm{C(S,t)} = \Phi(\mathrm{d}_1)\mathrm{S} - \Phi(\mathrm{d}_2)\mathrm{K}\mathrm{e}^{-rt}$$

$$\mathrm{d}_1 = \frac{1}{\sigma\sqrt{\mathrm{t}}}\left[\ln\left(\frac{S}{K}\right) + t\left(r + \frac{\sigma^2}{2}\right)\right]$$

$$\mathrm{d}_2 = \frac{1}{\sigma\sqrt{\mathrm{t}}}\left[\ln\left(\frac{S}{K}\right) + t\left(r - \frac{\sigma^2}{2}\right)\right]$$

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} \mathrm{e}^{-\frac{1}{2}z^2} dz$$

```r
#Cálculo de matriz de pérdidas y ganancias Opciones Tasa de interés
#dimensión
m=ncol(plazos_oir)  #PASO CLAVE
X_s_oir=matrix(0,Ns,n_if[4]) #Factores de riesgo simulados con base en DeltaX_s x0*(1+Delta_Xs) #PASO C
V_oir=matrix(0,Ns,m)
Vfr1_oir=matrix(0,Ns,m)
Vfr2_oir=matrix(0,Ns,m)
```

```r
Vfr3_oir=matrix(0,Ns,m)
PG_oir=matrix(0,Ns,m) #Pèrdidas y ganancias
PGfr1_oir=matrix(0,Ns,m)
PGfr2_oir=matrix(0,Ns,m)
PGfr3_oir=matrix(0,Ns,m)
PGT_oir=matrix(0,Ns,1)
PGfr1T_oir=matrix(0,Ns,1)
PGfr2T_oir=matrix(0,Ns,1)
PGfr3T_oir=matrix(0,Ns,1)

DeltaX_s_oir=DeltaX_s[,sum(n_if[1:3],1):sum(n_if[1:4])]  #PASO CLAVE
x0_oir=X_oir[1,] #PASO CLAVE

for (i in 1:Ns)
{
  X_s_oir[i,]=x0_oir*exp(DeltaX_s_oir[i,])
  #PASO CLAVE
  V_oir[i,]=  opctint(X_s_oir[i,(1:(n_if[4]/3))],X_s_oir[i,((n_if[4]/3+1):(n_if[4]/3*2))],K_oir,X_s_oir
  #PASO CLAVE
  Vfr1_oir[i,]=opctint(X_s_oir[i,(1:(n_if[4]/3))],x0_oir[((n_if[4]/3+1):(n_if[4]/3*2))],K_oir,x0_oir[((
  #PASO CLAVE
  Vfr2_oir[i,]=opctint(x0_oir[(1:(n_if[4]/3))],X_s_oir[i,((n_if[4]/3+1):(n_if[4]/3*2))],K_oir,x0_oir[((
  #PASO CLAVE
  Vfr3_oir[i,]=opctint(x0_oir[(1:(n_if[4]/3))],x0_oir[((n_if[4]/3+1):(n_if[4]/3*2))],K_oir,X_s_oir[i,((
  PG_oir[i,]=V_oir[i,]-V0_oir
  PGfr1_oir[i,]=Vfr1_oir[i,]-V0_oir
  PGfr2_oir[i,]=Vfr2_oir[i,]-V0_oir
  PGfr3_oir[i,]=Vfr3_oir[i,]-V0_oir
  PGT_oir[i,]=sum(PG_oir[i,])
  PGfr1T_oir[i,]=sum(PGfr1_oir[i,])
  PGfr2T_oir[i,]=sum(PGfr2_oir[i,])
  PGfr3T_oir[i,]=sum(PGfr3_oir[i,])
}



PG_oir[1:5,]
```

```
##              [,1]         [,2]
## [1,] -0.005367665  0.013020428
## [2,] -0.022653013 -0.002258036
## [3,]  0.024501047 -0.009382220
## [4,]  0.009719081  0.017544928
## [5,]  0.035551320 -0.009620467
```

```r
PGfr1_oir[1:5,]
```

```
##               [,1]          [,2]
## [1,] -0.0013378995 -1.532972e-04
## [2,]  0.0017599222  1.876434e-04
## [3,] -0.0002542780 -1.494674e-04
## [4,]  0.0007378206  5.784742e-05
## [5,] -0.0007412581 -2.430511e-05
```

```r
PGfr2_oir[1:5,]
```

```
##              [,1]         [,2]
## [1,] -0.004035152  0.013181874
## [2,] -0.024370145 -0.002443830
## [3,]  0.029006642  0.013527624
## [4,]  0.008974654  0.017827174
## [5,]  0.036319438 -0.008595355
```

```r
PGT_oir[1:5,]
```

```
## [1]  0.007652763 -0.024911049  0.015118827  0.027264009  0.025930853
```

```r
#VaR por posición
VaRCont_oir=matrix(0,1,m)
VaRfr1_oir=matrix(0,1,m)
VaRfr2_oir=matrix(0,1,m)
VaRfr3_oir=matrix(0,1,m)
CVaRCont_oir=matrix(0,1,m)
CVaRfr1_oir=matrix(0,1,m)
CVaRfr2_oir=matrix(0,1,m)
CVaRfr3_oir=matrix(0,1,m)
for (i in (1:m))
{
  VaRCont_oir[i]=equantile(PG_oir[,i],1-alpha,Ns)
  VaRfr1_oir[i]=equantile(PGfr1_oir[,i],1-alpha,Ns)
  VaRfr2_oir[i]=equantile(PGfr2_oir[,i],1-alpha,Ns)
  VaRfr3_oir[i]=equantile(PGfr3_oir[,i],1-alpha,Ns)
  CVaRfr1_oir[i]= mean(merge(which(PGfr1_oir[,i]<VaRfr1_oir[i]),cbind(seq(1,Ns),PGfr1_oir[,i]), by.x=1,
  CVaRfr2_oir[i]= mean(merge(which(PGfr2_oir[,i]<VaRfr2_oir[i]),cbind(seq(1,Ns),PGfr2_oir[,i]), by.x=1,
  CVaRfr3_oir[i]= mean(merge(which(PGfr3_oir[,i]<VaRfr3_oir[i]),cbind(seq(1,Ns),PGfr3_oir[,i]), by.x=1,
  CVaRCont_oir[i]= mean(merge(which(PG_oir[,i]<VaRCont_oir[i]),cbind(seq(1,Ns),PG_oir[,i]), by.x=1,by.y
}

VaRCont_oir
```

```
##             [,1]        [,2]
## [1,] -0.08360898 -0.04843656
```

```r
VaRfr1_oir
```

```
##              [,1]         [,2]
## [1,] -0.009428953 -0.001369044
```

```r
VaRfr2_oir
```

```
##             [,1]        [,2]
## [1,] -0.06126853 -0.03250889
```

```r
CVaRCont_oir
```

```
##            [,1]        [,2]
## [1,] -0.1228437 -0.06612845
```

```r
CVaRfr1_oir
```

```
##             [,1]         [,2]
## [1,] -0.01721673 -0.001992194
```

```r
CVaRfr2_oir
```

```
##             [,1]        [,2]
## [1,] -0.08075415 -0.04639936
```

```r
#VaR Total
VaRTotal_oir=equantile(PGT_oir,1-alpha,Ns)
CVaRTotal_oir= mean(merge(which(PGT_oir<VaRTotal_oir),cbind(seq(1,Ns),PGT_oir), by.x=1,by.y=1)[,2])
VaRTotalfr1_oir=equantile(PGfr1T_oir,1-alpha,Ns)
CVaRTotalfr1_oir= mean(PGfr1T_oir[which(PGfr1T_oir<VaRTotalfr1_oir),])
VaRTotalfr2_oir=equantile(PGfr2T_oir,1-alpha,Ns)
CVaRTotalfr2_oir= mean(PGfr2T_oir[which(PGfr2T_oir<VaRTotalfr2_oir),])
VaRTotalfr3_oir=equantile(PGfr3T_oir,1-alpha,Ns)
CVaRTotalfr3_oir= mean(PGfr3T_oir[which(PGfr2T_oir<VaRTotalfr2_oir),])


print(cbind(VaRTotal_oir,sum(V0_oir), VaRCont_oir, V0_oir))
```

```
##      VaRTotal_oir
## [1,]  -0.07299452 1.250313 -0.08360898 -0.04843656 1.002318 0.2479957
```

```r
print(cbind(CVaRTotal_oir,sum(V0_oir), CVaRCont_oir, V0_oir))
```

```
##      CVaRTotal_oir
## [1,]    -0.1375313 1.250313 -0.1228437 -0.06612845 1.002318 0.2479957
```

```r
print(cbind(VaRTotal_oir,VaRTotalfr1_oir,VaRTotalfr2_oir,VaRTotalfr3_oir))
```

```
##      VaRTotal_oir VaRTotalfr1_oir VaRTotalfr2_oir VaRTotalfr3_oir
## [1,]  -0.07299452     -0.01086969     -0.05581625     -0.05784975
```

```r
print(cbind(CVaRTotal_oir,CVaRTotalfr1_oir,CVaRTotalfr2_oir,CVaRTotalfr3_oir))
```

```
##      CVaRTotal_oir CVaRTotalfr1_oir CVaRTotalfr2_oir CVaRTotalfr3_oir
## [1,]    -0.1375313      -0.01909013      -0.07452019    -0.0007865348
```

con alisado

```r
#VaR por posición
VaRCont_CA_oir=matrix(0,1,m)
VaRfr1_CA_oir=matrix(0,1,m)
VaRfr2_CA_oir=matrix(0,1,m)
VaRfr3_CA_oir=matrix(0,1,m)
CVaRCont_CA_oir=matrix(0,1,m)
CVaRfr1_CA_oir=matrix(0,1,m)
CVaRfr2_CA_oir=matrix(0,1,m)
CVaRfr3_CA_oir=matrix(0,1,m)
for (i in (1:m))
{
  VaRCont_CA_oir[i]=wquantile(PG_oir[,i],w=rep(1,length(PGfr1T_sw)),1-alpha)
  VaRfr1_CA_oir[i]=wquantile(PGfr1_oir[,i],w=rep(1,length(PGfr1T_sw)),1-alpha)
  VaRfr2_CA_oir[i]=wquantile(PGfr2_oir[,i],w=rep(1,length(PGfr1T_sw)),1-alpha)
  VaRfr3_CA_oir[i]=wquantile(PGfr3_oir[,i],w=rep(1,length(PGfr1T_sw)),1-alpha)
  CVaRfr1_CA_oir[i]= mean(merge(which(PGfr1_oir[,i]<VaRfr1_CA_oir[i]),cbind(seq(1,Ns),PGfr1_oir[,i]), b
  CVaRfr2_oir[i]= mean(merge(which(PGfr2_oir[,i]<VaRfr2_CA_oir[i]),cbind(seq(1,Ns),PGfr2_oir[,i]), by.x
  CVaRfr3_oir[i]= mean(merge(which(PGfr3_oir[,i]<VaRfr3_CA_oir[i]),cbind(seq(1,Ns),PGfr3_oir[,i]), by.x
  CVaRCont_oir[i]= mean(merge(which(PG_oir[,i]<VaRCont_CA_oir[i]),cbind(seq(1,Ns),PG_oir[,i]), by.x=1,b
}

VaRCont_CA_oir
```

```
##             [,1]      [,2]
## [1,] -0.08360898 -0.047443
```

```r
VaRfr1_CA_oir
```

```
##             [,1]         [,2]
## [1,] -0.01341102 -0.001369044
```

```r
VaRfr2_CA_oir
```

```
##             [,1]       [,2]
## [1,] -0.06280167 -0.0323975
```

```r
CVaRCont_CA_oir
```

```
##      [,1] [,2]
## [1,]    0    0
```

```r
CVaRfr1_CA_oir
```

```
##             [,1]         [,2]
## [1,] -0.01797787 -0.001992194
```

```r
CVaRfr2_CA_oir
```

```
##      [,1] [,2]
## [1,]    0    0
```

```r
#VaR Total
VaRTotal_CA_oir=wquantile(PGT_oir,w=rep(1,length(PGT_oir)),1-alpha)
CVaRTotal_CA_oir= mean(merge(which(PGT_oir<VaRTotal_CA_oir),cbind(seq(1,Ns),PGT_oir), by.x=1,by.y=1)[,2
VaRTotalfr1_CA_oir=wquantile(PGfr1T_oir,w=rep(1,length(PGfr1T_oir)),1-alpha)
CVaRTotalfr1_CA_oir= mean(PGfr1T_oir[which(PGfr1T_oir<VaRTotalfr1_CA_oir),])
VaRTotalfr2_CA_oir=wquantile(PGfr2T_oir,w=rep(1,length(PGfr2T_oir)),1-alpha)
CVaRTotalfr2_CA_oir= mean(PGfr2T_oir[which(PGfr2T_oir<VaRTotalfr2_CA_oir),])
VaRTotalfr3_CA_oir=wquantile(PGfr3T_oir,w=rep(1,length(PGfr3T_oir)),1-alpha)
CVaRTotalfr3_CA_oir= mean(PGfr3T_oir[which(PGfr2T_oir<VaRTotalfr2_CA_oir),])


print(cbind(VaRTotal_CA_oir,sum(V0_oir), VaRCont_CA_oir, V0_oir))
```

```
##      VaRTotal_CA_oir
## [1,]     -0.07301903 1.250313 -0.08360898 -0.047443 1.002318 0.2479957
```

```r
print(cbind(CVaRTotal_CA_oir,sum(V0_oir), CVaRCont_CA_oir, V0_oir))
```

```
##      CVaRTotal_CA_oir
## [1,]       -0.1504338 1.250313 0 0 1.002318 0.2479957
```

```r
print(cbind(VaRTotal_CA_oir,VaRTotalfr1_CA_oir,VaRTotalfr2_CA_oir,VaRTotalfr3_CA_oir))
```

```
##      VaRTotal_CA_oir VaRTotalfr1_CA_oir VaRTotalfr2_CA_oir VaRTotalfr3_CA_oir
## [1,]     -0.07301903        -0.01478006        -0.05581625        -0.05784975
```

```r
print(cbind(CVaRTotal_CA_oir,CVaRTotalfr1_CA_oir,CVaRTotalfr2_CA_oir,CVaRTotalfr3_CA_oir))
```

```
##      CVaRTotal_CA_oir CVaRTotalfr1_CA_oir CVaRTotalfr2_CA_oir
## [1,]       -0.1504338         -0.01995214         -0.07452019
##      CVaRTotalfr3_CA_oir
## [1,]       -0.0007865348
```

Riesgo de Forwards TdC

```r
#Cálculo de matriz de pérdidas y ganancias FUTUROS TDC
#dimensión
m=ncol(plazos_fwd)  #PASO CLAVE
X_s_fwtdc=matrix(0,Ns,n_if[5]) #Factores de riesgo simulados con base en DeltaX_s x0*(1+Delta_Xs) #PASO
V_fwtdc=matrix(0,Ns,m)
Vfr1_fwtdc=matrix(0,Ns,m)
Vfr2_fwtdc=matrix(0,Ns,m)
Vfr3_fwtdc=matrix(0,Ns,m)
PG_fwtdc=matrix(0,Ns,m) #Pèrdidas y ganancias
PGfr1_fwtdc=matrix(0,Ns,m)
PGfr2_fwtdc=matrix(0,Ns,m)
PGfr3_fwtdc=matrix(0,Ns,m)
PGT_fwtdc=matrix(0,Ns,1)
PGfr1T_fwtdc=matrix(0,Ns,1)
```

```
PGfr2T_fwtdc=matrix(0,Ns,1)
PGfr3T_fwtdc=matrix(0,Ns,1)

DeltaX_s_fwtdc=DeltaX_s[,sum(n_if[1:4],1):sum(n_if[1:5])]  #PASO CLAVE
x0_fwtdc=X_futtdc[1,] #PASO CLAVE

for (i in 1:Ns)
{
  X_s_fwtdc[i,]=x0_fwtdc*exp(DeltaX_s_fwtdc[i,])
  #PASO CLAVE
  V_fwtdc[i,]=futuroTC(plazos_fwd,X_s_fwtdc[i,1:((n_if[5]-1)/2)],X_s_fwtdc[i,((n_if[5]-1)/2+1):(n_if[5]
  #PASO CLAVE
  Vfr1_fwtdc[i,]=futuroTC(plazos_fwd,X_s_fwtdc[i,1:((n_if[5]-1)/2)],x0_fwtdc[((n_if[5]-1)/2+1):(n_if[5]
  #PASO CLAVE
  Vfr2_fwtdc[i,]=futuroTC(plazos_fwd,x0_fwtdc[1:((n_if[5]-1)/2)],X_s_fwtdc[i,((n_if[5]-1)/2+1):(n_if[5]
  PG_fwtdc[i,]=V_fwtdc[i,]-V0_fwtdc
  PGfr1_fwtdc[i,]=Vfr1_fwtdc[i,]-V0_fwtdc
  PGfr2_fwtdc[i,]=Vfr2_fwtdc[i,]-V0_fwtdc
  PGfr3_fwtdc[i,]=Vfr3_fwtdc[i,]-V0_fwtdc
  PGT_fwtdc[i,]=sum(PG_fwtdc[i,])
  PGfr1T_fwtdc[i,]=sum(PGfr1_fwtdc[i,])
  PGfr2T_fwtdc[i,]=sum(PGfr2_fwtdc[i,])
  PGfr3T_fwtdc[i,]=sum(PGfr3_fwtdc[i,])
}



PG_fwtdc[1:5,]
```

```
## [1] -12.473326   6.079641  -3.592397 -15.819929  -4.484017
```

```
PGfr1_fwtdc[1:5,]
```

```
## [1]  0.003688629 -0.005272958  0.003252243 -0.003036069  0.001196320
```

```
PGfr2_fwtdc[1:5,]
```

```
## [1]  0.0027492092 -0.0021228650 -0.0007081384  0.0018833231 -0.0006142446
```

```
PGT_fwtdc[1:5,]
```

```
## [1] -12.473326   6.079641  -3.592397 -15.819929  -4.484017
```

```
#VaR por posición
VaRCont_fwtdc=matrix(0,1,m)
VaRfr1_fwtdc=matrix(0,1,m)
VaRfr2_fwtdc=matrix(0,1,m)
VaRfr3_fwtdc=matrix(0,1,m)
CVaRCont_fwtdc=matrix(0,1,m)
CVaRfr1_fwtdc=matrix(0,1,m)
```

```
CVaRfr2_fwtdc=matrix(0,1,m)
CVaRfr3_fwtdc=matrix(0,1,m)
for (i in (1:m))
{
  VaRCont_fwtdc[i]=quantile(PG_fwtdc[,i],1-alpha,Ns)
  VaRfr1_fwtdc[i]=quantile(PGfr1_fwtdc[,i],1-alpha,Ns)
  VaRfr2_fwtdc[i]=quantile(PGfr2_fwtdc[,i],1-alpha,Ns)
  VaRfr3_fwtdc[i]=quantile(PGfr3_fwtdc[,i],1-alpha,Ns)
  CVaRfr1_fwtdc[i]= mean(merge(which(PGfr1_fwtdc[,i]<VaRfr1_fwtdc[i]),cbind(seq(1,Ns),PGfr1_fwtdc[,i]),
  CVaRfr2_fwtdc[i]= mean(merge(which(PGfr2_fwtdc[,i]<VaRfr2_fwtdc[i]),cbind(seq(1,Ns),PGfr2_fwtdc[,i]),
  CVaRfr3_fwtdc[i]= mean(merge(which(PGfr3_fwtdc[,i]<VaRfr3_fwtdc[i]),cbind(seq(1,Ns),PGfr3_fwtdc[,i]),
  CVaRCont_fwtdc[i]= mean(merge(which(PG_fwtdc[,i]<VaRCont_fwtdc[i]),cbind(seq(1,Ns),PG_fwtdc[,i]), by.
}

VaRCont_fwtdc
```

```
##           [,1]
## [1,] -21.11645
```

```
VaRfr1_fwtdc
```

```
##              [,1]
## [1,] -0.009740411
```

```
VaRfr2_fwtdc
```

```
##              [,1]
## [1,] -0.004602614
```

```
CVaRCont_fwtdc
```

```
##          [,1]
## [1,] -28.1163
```

```
CVaRfr1_fwtdc
```

```
##             [,1]
## [1,] -0.01409534
```

```
CVaRfr2_fwtdc
```

```
##              [,1]
## [1,] -0.007956645
```

```
#VaR Total
VaRTotal_fwtdc=quantile(PGT_fwtdc,1-alpha,Ns)
CVaRTotal_fwtdc= mean(merge(which(PGT_fwtdc<VaRTotal_fwtdc),cbind(seq(1,Ns),PGT_fwtdc), by.x=1,by.y=1)[
VaRTotalfr1_fwtdc=quantile(PGfr1T_fwtdc,1-alpha,Ns)
CVaRTotalfr1_fwtdc= mean(PGfr1T_fwtdc[which(PGfr1T_fwtdc<VaRTotalfr1_fwtdc),])
```

```
VaRTotalfr2_fwtdc=quantile(PGfr2T_fwtdc,1-alpha,Ns)
CVaRTotalfr2_fwtdc= mean(PGfr2T_fwtdc[which(PGfr2T_fwtdc<VaRTotalfr2_fwtdc),])
VaRTotalfr3_fwtdc=quantile(PGfr3T_fwtdc,1-alpha,Ns)
CVaRTotalfr3_fwtdc= mean(PGfr3T_fwtdc[which(PGfr2T_fwtdc<VaRTotalfr2_fwtdc),])


cbind(VaRTotal_fwtdc,sum(V0_fwtdc), VaRCont_fwtdc, V0_fwtdc)
```

```
##    VaRTotal_fwtdc
## 2%      -21.11645 -85.61414 -21.11645 -85.61414
```

```
print(V0_fwtdc)
```

```
##           [,1]
## [1,] -85.61414
```

```
print(VaRCont_fwtdc)
```

```
##           [,1]
## [1,] -21.11645
```

```
cbind(CVaRTotal_fwtdc,sum(V0_fwtdc), CVaRCont_fwtdc, V0_fwtdc)
```

```
##      CVaRTotal_fwtdc
## [1,]        -28.1163 -85.61414 -28.1163 -85.61414
```

```
print(CVaRCont_fwtdc)
```

```
##          [,1]
## [1,] -28.1163
```

```
print(CVaRfr1_fwtdc)
```

```
##             [,1]
## [1,] -0.01409534
```

```
print(CVaRfr2_fwtdc)
```

```
##              [,1]
## [1,] -0.007956645
```

```
print(CVaRfr3_fwtdc)
```

```
##      [,1]
## [1,]  NaN
```

```r
print(cbind(VaRTotal_fwtdc,VaRTotalfr1_fwtdc,VaRTotalfr2_fwtdc,VaRTotalfr3_fwtdc))
```

```
##    VaRTotal_fwtdc VaRTotalfr1_fwtdc VaRTotalfr2_fwtdc VaRTotalfr3_fwtdc
## 2%      -21.11645      -0.009740411      -0.004602614          85.61414
```

```r
print(cbind(CVaRTotal_fwtdc,CVaRTotalfr1_fwtdc,CVaRTotalfr2_fwtdc,CVaRTotalfr3_fwtdc))
```

```
##      CVaRTotal_fwtdc CVaRTotalfr1_fwtdc CVaRTotalfr2_fwtdc CVaRTotalfr3_fwtdc
## [1,]        -28.1163        -0.01409534       -0.007956645           85.61414
```

con alisado

```r
#VaR por posición
VaRCont_CA_fwtdc=matrix(0,1,m)
VaRfr1_CA_fwtdc=matrix(0,1,m)
VaRfr2_CA_fwtdc=matrix(0,1,m)
VaRfr3_CA_fwtdc=matrix(0,1,m)
CVaRCont_CA_fwtdc=matrix(0,1,m)
CVaRfr1_CA_fwtdc=matrix(0,1,m)
CVaRfr2_CA_fwtdc=matrix(0,1,m)
CVaRfr3_CA_fwtdc=matrix(0,1,m)
for (i in (1:m))
{
  VaRCont_CA_fwtdc[i]=wquantile(PG_fwtdc[,i],w=rep(1,length(PG_fwtdc[,i])),1-alpha)
  VaRfr1_CA_fwtdc[i]=wquantile(PGfr1_fwtdc[,i],w=rep(1,length(PGfr1_fwtdc[,i])),1-alpha)
  VaRfr2_CA_fwtdc[i]=wquantile(PGfr2_fwtdc[,i],w=rep(1,length(PGfr2_fwtdc[,i])),1-alpha)
  VaRfr3_CA_fwtdc[i]=wquantile(PGfr3_fwtdc[,i],w=rep(1,length(PGfr3_fwtdc[,i])),1-alpha)
  CVaRfr1_CA_fwtdc[i]= mean(merge(which(PGfr1_fwtdc[,i]<VaRfr1_CA_fwtdc[i]),cbind(seq(1,Ns),PGfr1_fwtdc
  CVaRfr2_CA_fwtdc[i]= mean(merge(which(PGfr2_fwtdc[,i]<VaRfr2_CA_fwtdc[i]),cbind(seq(1,Ns),PGfr2_fwtdc
  CVaRfr3_CA_fwtdc[i]= mean(merge(which(PGfr3_fwtdc[,i]<VaRfr3_CA_fwtdc[i]),cbind(seq(1,Ns),PGfr3_fwtdc
  CVaRCont_CA_fwtdc[i]= mean(merge(which(PG_fwtdc[,i]<VaRCont_CA_fwtdc[i]),cbind(seq(1,Ns),PG_fwtdc[,i]
}

VaRCont_CA_fwtdc
```

```
##           [,1]
## [1,] -21.13341
```

```r
VaRfr1_CA_fwtdc
```

```
##              [,1]
## [1,] -0.009744203
```

```r
VaRfr2_CA_fwtdc
```

```
##              [,1]
## [1,] -0.004606392
```

```r
CVaRCont_CA_fwtdc
```

```
##            [,1]
## [1,] -29.51288
```

```r
CVaRfr1_CA_fwtdc
```

```
##              [,1]
## [1,] -0.01496557
```

```r
CVaRfr2_CA_fwtdc
```

```
##               [,1]
## [1,] -0.008626696
```

```r
#VaR Total
VaRTotal_CA_fwtdc=quantile(PGT_fwtdc,w=rep(1,length(PGT_fwtdc)),1-alpha)
CVaRTotal_CA_fwtdc= mean(merge(which(PGT_fwtdc<VaRTotal_CA_fwtdc),cbind(seq(1,Ns),PGT_fwtdc), by.x=1,by
VaRTotalfr1_CA_fwtdc=quantile(PGfr1T_fwtdc,w=rep(1,length(PGfr1T_fwtdc)),1-alpha)
CVaRTotalfr1_CA_fwtdc= mean(PGfr1T_fwtdc[which(PGfr1T_fwtdc<VaRTotalfr1_CA_fwtdc),])
VaRTotalfr2_CA_fwtdc=quantile(PGfr2T_fwtdc,w=rep(1,length(PGfr2T_fwtdc)),1-alpha)
CVaRTotalfr2_CA_fwtdc= mean(PGfr2T_fwtdc[which(PGfr2T_fwtdc<VaRTotalfr2_CA_fwtdc),])
VaRTotalfr3_CA_fwtdc=quantile(PGfr3T_fwtdc,w=rep(1,length(PGfr3T_fwtdc)),1-alpha)
CVaRTotalfr3_CA_fwtdc= mean(PGfr3T_fwtdc[which(PGfr2T_fwtdc<VaRTotalfr2_CA_fwtdc),])


cbind(VaRTotal_CA_fwtdc,sum(V0_fwtdc), VaRCont_CA_fwtdc, V0_fwtdc)
```

```
##    VaRTotal_CA_fwtdc
## 2%         -21.11645 -85.61414 -21.13341 -85.61414
```

```r
print(V0_fwtdc)
```

```
##            [,1]
## [1,] -85.61414
```

```r
print(VaRCont_CA_fwtdc)
```

```
##            [,1]
## [1,] -21.13341
```

```r
cbind(CVaRTotal_CA_fwtdc,sum(V0_fwtdc), CVaRCont_CA_fwtdc, V0_fwtdc)
```

```
##      CVaRTotal_CA_fwtdc
## [1,]           -28.1163 -85.61414 -29.51288 -85.61414
```

```r
print(CVaRCont_CA_fwtdc)
```

```
##           [,1]
## [1,] -29.51288
```

```r
print(CVaRfr1_CA_fwtdc)
```

```
##             [,1]
## [1,] -0.01496557
```

```r
print(CVaRfr2_CA_fwtdc)
```

```
##              [,1]
## [1,] -0.008626696
```

```r
print(CVaRfr3_CA_fwtdc)
```

```
##      [,1]
## [1,]  NaN
```

```r
print(cbind(VaRTotal_CA_fwtdc,VaRTotalfr1_CA_fwtdc,VaRTotalfr2_CA_fwtdc,VaRTotalfr3_CA_fwtdc))
```

```
##    VaRTotal_CA_fwtdc VaRTotalfr1_CA_fwtdc VaRTotalfr2_CA_fwtdc
## 2%         -21.11645         -0.009740411         -0.004602614
##    VaRTotalfr3_CA_fwtdc
## 2%             85.61414
```

```r
print(cbind(CVaRTotal_CA_fwtdc,CVaRTotalfr1_CA_fwtdc,CVaRTotalfr2_CA_fwtdc,CVaRTotalfr3_CA_fwtdc))
```

```
##      CVaRTotal_CA_fwtdc CVaRTotalfr1_CA_fwtdc CVaRTotalfr2_CA_fwtdc
## [1,]           -28.1163           -0.01409534          -0.007956645
##      CVaRTotalfr3_CA_fwtdc
## [1,]              85.61414
```

Riesgo Forward Índice

```r
#Cálculo de matriz de pérdidas y ganancias FUTUROS IPC
m=ncol(plazos_fwd_ind)  #PASO CLAVE
X_s_fwind=matrix(0,Ns,n_if[6]) #Factores de riesgo simulados con base en DeltaX_s x0*(1+Delta_Xs) #PASO
V_fwind=matrix(0,Ns,m)
Vfr1_fwind=matrix(0,Ns,m)
Vfr2_fwind=matrix(0,Ns,m)
Vfr3_fwind=matrix(0,Ns,m)
PG_fwind=matrix(0,Ns,m) #Pèrdidas y ganancias
PGfr1_fwind=matrix(0,Ns,m)
PGfr2_fwind=matrix(0,Ns,m)
PGfr3_fwind=matrix(0,Ns,m)
```

```
PGT_fwind=matrix(0,Ns,1)
PGfr1T_fwind=matrix(0,Ns,1)
PGfr2T_fwind=matrix(0,Ns,1)
PGfr3T_fwind=matrix(0,Ns,1)

DeltaX_s_fwind=DeltaX_s[,sum(n_if[1:5],1):sum(n_if[1:6])]  #PASO CLAVE
x0_fwind=X_futind[1,] #PASO CLAVE


for (i in 1:Ns)
{
  X_s_fwind[i,]=x0_fwind* exp(DeltaX_s_fwind[i,])
  #PASO CLAVE
  V_fwind[i,]=futuroTC(plazos_fwd_ind,X_s_fwind[i,1:(n_if[6]/3)],X_s_fwind[i,(n_if[6]/3+1):(n_if[6]*2/3)
  Vfr1_fwind[i,]=futuroTC(plazos_fwd_ind,X_s_fwind[i,1:(n_if[6]/3)],X_s_fwind[i,(n_if[6]/3+1):(n_if[6]*
  #PASO CLAVE
  Vfr2_fwind[i,]=futuroTC(plazos_fwd_ind,X_s_fwind[i,1:(n_if[6]/3)],X_s_fwind[i,(n_if[6]/3+1):(n_if[6]*
  #PASO CLAVE
  Vfr3_fwind[i,]=futuroTC(plazos_fwd_ind,X_s_fwind[i,1:(n_if[6]/3)],X_s_fwind[i,(n_if[6]/3+1):(n_if[6]*
  PG_fwind[i,]=V_fwind[i,]-V0_fwind
  PGfr1_fwind[i,]=Vfr1_fwind[i,]-V0_fwind
  PGfr2_fwind[i,]=Vfr2_fwind[i,]-V0_fwind
  PGfr3_fwind[i,]=Vfr3_fwind[i,]-V0_fwind
  PGT_fwind[i,]=sum(PG_fwind[i,])
  PGfr1T_fwind[i,]=sum(PGfr1_fwind[i,])
  PGfr2T_fwind[i,]=sum(PGfr2_fwind[i,])
  PGfr3T_fwind[i,]=sum(PGfr3_fwind[i,])
}


PG_fwind[1:5,]
```

```
## [1] -14834.44  21353.19  12466.42 -19709.56  34163.14
```

```
#PGfr1_fwind[1:5,]
#PGfr2_fwind[1:5,]
#PGT_fwind[1:5,]




#VaR por posición
VaRCont_fwind=matrix(0,1,m)
VaRfr1_fwind=matrix(0,1,m)
VaRfr2_fwind=matrix(0,1,m)
VaRfr3_fwind=matrix(0,1,m)
CVaRCont_fwind=matrix(0,1,m)
CVaRfr1_fwind=matrix(0,1,m)
CVaRfr2_fwind=matrix(0,1,m)
CVaRfr3_fwind=matrix(0,1,m)
for (i in (1:m))
{
  VaRCont_fwind[i]=quantile(PG_fwind[,i],1-alpha,Ns)
  VaRfr1_fwind[i]=quantile(PGfr1_fwind[,i],1-alpha,Ns)
```

```
  VaRfr2_fwind[i]=quantile(PGfr2_fwind[,i],1-alpha,Ns)
  VaRfr3_fwind[i]=quantile(PGfr3_fwind[,i],1-alpha,Ns)
  CVaRfr1_fwind[i]= mean(merge(which(PGfr1_fwind[,i]<VaRfr1_fwind[i]),cbind(seq(1,Ns),PGfr1_fwind[,i]),
  CVaRfr2_fwind[i]= mean(merge(which(PGfr2_fwind[,i]<VaRfr2_fwind[i]),cbind(seq(1,Ns),PGfr2_fwind[,i]),
  CVaRfr3_fwind[i]= mean(merge(which(PGfr3_fwind[,i]<VaRfr3_fwind[i]),cbind(seq(1,Ns),PGfr3_fwind[,i]),
  CVaRCont_fwind[i]= mean(merge(which(PG_fwind[,i]<VaRCont_fwind[i]),cbind(seq(1,Ns),PG_fwind[,i]), by.
}
VaRCont_fwind
```

```
##            [,1]
## [1,] -51221.9
```

```
VaRfr1_fwind
```

```
##            [,1]
## [1,] -51221.9
```

```
VaRfr2_fwind
```

```
##            [,1]
## [1,] -51221.9
```

```
CVaRCont_fwind
```

```
##             [,1]
## [1,] -57884.57
```

```
CVaRfr1_fwind
```

```
##             [,1]
## [1,] -57884.57
```

```
CVaRfr2_fwind
```

```
##             [,1]
## [1,] -57884.57
```

```
#VaR Total

VaRTotal_fwind=quantile(PGT_fwind,1-alpha,Ns)
CVaRTotal_fwind= mean(merge(which(PGT_fwind<VaRTotal_fwind),cbind(seq(1,Ns),PGT_fwind), by.x=1,by.y=1)[
VaRTotalfr1_fwind=quantile(PGfr1T_fwind,1-alpha,Ns)
CVaRTotalfr1_fwind= mean(PGfr1T_fwind[which(PGfr1T_fwind<VaRTotalfr1_fwind),])
VaRTotalfr2_fwind=quantile(PGfr2T_fwind,1-alpha,Ns)
CVaRTotalfr2_fwind= mean(PGfr2T_fwind[which(PGfr2T_fwind<VaRTotalfr2_fwind),])
VaRTotalfr3_fwind=quantile(PGfr3T_fwind,1-alpha,Ns)
CVaRTotalfr3_fwind= mean(PGfr3T_fwind[which(PGfr3T_fwind<VaRTotalfr3_fwind),])


print(cbind(VaRTotal_fwind,sum(V0_fwind), VaRCont_fwind, V0_fwind))
```

```
##    VaRTotal_fwind
## 2%       -51221.9 342308.9 -51221.9 342308.9
```

```r
print(cbind(CVaRTotal_fwind,sum(V0_fwind), CVaRCont_fwind, V0_fwind))
```

```
##      CVaRTotal_fwind
## [1,]       -57884.57 342308.9 -57884.57 342308.9
```

```r
print(cbind(VaRTotal_fwind,VaRTotalfr1_fwind,VaRTotalfr2_fwind,VaRTotalfr3_fwind))
```

```
##    VaRTotal_fwind VaRTotalfr1_fwind VaRTotalfr2_fwind VaRTotalfr3_fwind
## 2%       -51221.9          -51221.9          -51221.9          -51221.9
```

```r
cbind(CVaRTotal_fwind,CVaRTotalfr1_fwind,CVaRTotalfr2_fwind,CVaRTotalfr3_fwind)
```

```
##      CVaRTotal_fwind CVaRTotalfr1_fwind CVaRTotalfr2_fwind CVaRTotalfr3_fwind
## [1,]       -57884.57          -57884.57          -57884.57          -57884.57
```

con alisado

```r
#VaR por posición
VaRCont_CA_fwind=matrix(0,1,m)
VaRfr1_CA_fwind=matrix(0,1,m)
VaRfr2_CA_fwind=matrix(0,1,m)
VaRfr3_CA_fwind=matrix(0,1,m)
CVaRCont_CA_fwind=matrix(0,1,m)
CVaRfr1_CA_fwind=matrix(0,1,m)
CVaRfr2_CA_fwind=matrix(0,1,m)
CVaRfr3_CA_fwind=matrix(0,1,m)
for (i in (1:m))
{
  VaRCont_CA_fwind[i]=wquantile(PG_fwind[,i],w=rep(1,length(PG_fwind[,i])),1-alpha)
  VaRfr1_CA_fwind[i]=wquantile(PGfr1_fwind[,i],w=rep(1,length(PGfr1_fwind[,i])),1-alpha)
  VaRfr2_CA_fwind[i]=wquantile(PGfr2_fwind[,i],w=rep(1,length(PGfr2_fwind[,i])),1-alpha)
  VaRfr3_CA_fwind[i]=wquantile(PGfr3_fwind[,i],w=rep(1,length(PGfr3_fwind[,i])),1-alpha)
  CVaRfr1_CA_fwind[i]= mean(merge(which(PGfr1_fwind[,i]<VaRfr1_CA_fwind[i]),cbind(seq(1,Ns),PGfr1_fwind
  CVaRfr2_CA_fwind[i]= mean(merge(which(PGfr2_fwind[,i]<VaRfr2_CA_fwind[i]),cbind(seq(1,Ns),PGfr2_fwind
  CVaRfr3_CA_fwind[i]= mean(merge(which(PGfr3_fwind[,i]<VaRfr3_CA_fwind[i]),cbind(seq(1,Ns),PGfr3_fwind
  CVaRCont_CA_fwind[i]= mean(merge(which(PG_fwind[,i]<VaRCont_CA_fwind[i]),cbind(seq(1,Ns),PG_fwind[,i]
}
VaRCont_CA_fwind
```

```
##           [,1]
## [1,] -51237.82
```

```r
VaRfr1_CA_fwind
```

```
##           [,1]
## [1,] -51237.82
```

```r
VaRfr2_CA_fwind
```

```
##           [,1]
## [1,] -51237.82
```

```r
CVaRCont_CA_fwind
```

```
##           [,1]
## [1,] -59213.92
```

```r
CVaRfr1_CA_fwind
```

```
##           [,1]
## [1,] -59213.92
```

```r
CVaRfr2_CA_fwind
```

```
##           [,1]
## [1,] -59213.92
```

```r
#VaR Total

VaRTotal_CA_fwind=wquantile(PGT_fwind,w=rep(1,length(PGT_fwind)),1-alpha)
CVaRTotal_CA_fwind= mean(merge(which(PGT_fwind<VaRTotal_CA_fwind),cbind(seq(1,Ns),PGT_fwind), by.x=1,by
VaRTotalfr1_CA_fwind=wquantile(PGfr1T_fwind,w=rep(1,length(PGfr1T_fwind)),1-alpha)
CVaRTotalfr1_CA_fwind= mean(PGfr1T_fwind[which(PGfr1T_fwind<VaRTotalfr1_CA_fwind),])
VaRTotalfr2_CA_fwind=wquantile(PGfr2T_fwind,w=rep(1,length(PGfr2T_fwind)),1-alpha)
CVaRTotalfr2_CA_fwind= mean(PGfr2T_fwind[which(PGfr2T_fwind<VaRTotalfr2_CA_fwind),])
VaRTotalfr3_CA_fwind=wquantile(PGfr3T_fwind,w=rep(1,length(PGfr3T_fwind)),1-alpha)
CVaRTotalfr3_CA_fwind= mean(PGfr3T_fwind[which(PGfr3T_fwind<VaRTotalfr3_CA_fwind),])


print(cbind(VaRTotal_CA_fwind,sum(V0_fwind), VaRCont_CA_fwind, V0_fwind))
```

```
##      VaRTotal_CA_fwind
## [1,]         -51237.82 342308.9 -51237.82 342308.9
```

```r
print(cbind(CVaRTotal_CA_fwind,sum(V0_fwind), CVaRCont_CA_fwind, V0_fwind))
```

```
##      CVaRTotal_CA_fwind
## [1,]          -59213.92 342308.9 -59213.92 342308.9
```

```r
print(cbind(VaRTotal_CA_fwind,VaRTotalfr1_CA_fwind,VaRTotalfr2_CA_fwind,VaRTotalfr3_CA_fwind))
```

```
##      VaRTotal_CA_fwind VaRTotalfr1_CA_fwind VaRTotalfr2_CA_fwind
## [1,]         -51237.82            -51237.82            -51237.82
##      VaRTotalfr3_CA_fwind
## [1,]            -51237.82
```

```r
cbind(CVaRTotal_CA_fwind,CVaRTotalfr1_CA_fwind,CVaRTotalfr2_CA_fwind,CVaRTotalfr3_CA_fwind)
```

```
##      CVaRTotal_CA_fwind CVaRTotalfr1_CA_fwind CVaRTotalfr2_CA_fwind
## [1,]          -59213.92             -59213.92             -59213.92
##      CVaRTotalfr3_CA_fwind
## [1,]             -59213.92
```

RIESGO INTEGRAL

En esta sección, obetenemos las medidas de riesgo de toda la cartera:

Acciones

SUMAMOS PERDIDAS Y GANACIAS DE LAS ACCIONES Y DIVISAS Y SE ENCUENTRA EL CUALTIL DE LA PRECISION DESEADA $1 - \alpha$; se encuentra la esperanza condicional del portafolio de acciones y divisas dado que este no exceda al var.

```r
#Medición de riesgo por factor de riesgo de todo el portafolios
#Acciones
#1. Acciones
PGPort_ACC=PGfr2T_acc_div + PGfr3T_fwind #Pérdidas y ganancias
VaRPort_ACC=equantile(PGPort_ACC,1-alpha,Ns) #VaR
CVaRPort_ACC= mean(PGPort_ACC[which(PGPort_ACC<VaRPort_ACC)]) #CVaR
```

SUMAMOS LAS PERDIDAS Y GANANCIAS

```r
#Tasa de Interés
#1. Dado que swaps y bondes son de tasa de interés usaremos PGT_bd y PGT_sw
#2. Para futuros usaremos PGfr1T_fwtdc y PGfr2T_fwtdc
PGPort_TI=PGT_bd+PGT_sw+PGfr1T_oir +PGfr2T_oir + PGfr1T_fwind +PGfr2T_fwind + PGfr1T_fwtdc + PGfr2T_fwt
VaRPort_TI=equantile(PGPort_TI,1-alpha,Ns) #VaR
CVaRPort_TI= mean(PGPort_TI[which(PGPort_TI<VaRPort_TI)]) #CVaR

#Tipo de cambio
#1. Dado que swaps y bondes son de tasa de interés no usamos nada
#2. Para futuros usamos sólo PGfr3T_fwtdc
PGPort_TDC=PGfr1T_acc_div + PGfr3T_fwtdc  #Pérdidas y ganancias
VaRPort_TDC=equantile(PGPort_TDC,1-alpha,Ns) #VaR
CVaRPort_TDC= mean(PGPort_TDC[which(PGPort_TDC<VaRPort_TDC)]) #CVaR

#Volatilidad
#1. Sólo aplica la volatilidad de Opciones de tasa de interés

PGPort_VOL=PGfr3T_oir  #Pérdidas y ganancias
VaRPort_VOL=equantile(PGPort_VOL,1-alpha,Ns) #VaR
CVaRPort_VOL= mean(PGPort_VOL[which(PGPort_VOL<VaRPort_VOL)]) #CVaR


#Medición de riesgo de todo el portafolios
#Sumar todos los PGT de todos los instrumentos

PGT_Port=PGPort_ACC+PGPort_TI+PGPort_TDC+PGPort_VOL
```

```
VaRTotal_Port=equantile(PGT_Port,1-alpha,Ns) #VaR
CVaRTotal_Port= mean(PGT_Port[which(PGT_Port<VaRTotal_Port)]) #CVaR
print(VaRTotal_Port)
```

```
## [1] -178347.9
```

```
print(CVaRTotal_Port)
```

```
## [1] -206693.7
```

```
print(VOT_port)
```

```
## [1] 8629095
```

Con alisado

Repetimos el procedimiento anterior pero ahora con alisado.

```
#Medición de riesgo por factor de riesgo de todo el portafolios
#Acciones
#1. Acciones
PGPort_ACC=PGfr2T_acc_div + PGfr3T_fwind #Pérdidas y ganancias
VaRPort_CA_ACC=wquantile(PGPort_ACC,w=rep(1,length(PGPort_ACC)),1-alpha) #VaR
CVaRPort_CA_ACC= mean(PGPort_ACC[which(PGPort_ACC<VaRPort_CA_ACC)]) #CVaR
```

SUMAMOS LAS PERDIDAS Y GANANCIAS

```
#Tasa de Interés
#1. Dado que swaps y bondes son de tasa de interés usaremos PGT_bd y PGT_sw
#2. Para futuros usaremos PGfr1T_fwtdc y PGfr2T_fwtdc
PGPort_TI=PGT_bd+PGT_sw+PGfr1T_oir +PGfr2T_oir + PGfr1T_fwind +PGfr2T_fwind + PGfr1T_fwtdc + PGfr2T_fwt
VaRPort_CA_TI=wquantile(PGPort_TI,w=rep(1,length(PGPort_TI)),1-alpha) #VaR
CVaRPort_CA_TI= mean(PGPort_TI[which(PGPort_TI<VaRPort_CA_TI)]) #CVaR

#Tipo de cambio
#1. Dado que swaps y bondes son de tasa de interés no usamos nada
#2. Para futuros usamos sólo PGfr3T_fwtdc
PGPort_TDC=PGfr1T_acc_div + PGfr3T_fwtdc  #Pérdidas y ganancias
VaRPort_CA_TDC=wquantile(PGPort_TDC,w=rep(1,length(PGPort_TDC)),1-alpha) #VaR
CVaRPort_CA_TDC= mean(PGPort_TDC[which(PGPort_TDC<VaRPort_CA_TDC)]) #CVaR

#Volatilidad
#1. Sólo aplica la volatilidad de Opciones de tasa de interés

PGPort_VOL=PGfr3T_oir  #Pérdidas y ganancias
VaRPort_CA_VOL=wquantile(PGPort_VOL,w=rep(1,length(PGPort_VOL)),1-alpha) #VaR
CVaRPort_CA_VOL= mean(PGPort_VOL[which(PGPort_VOL<VaRPort_CA_VOL)]) #CVaR


#Medición de riesgo de todo el portafolios
```

```r
#Sumar todos los PGT de todos los instrumentos

PGT_Port=PGPort_ACC+PGPort_TI+PGPort_TDC+PGPort_VOL
VaRTotal_CA_Port=wquantile(PGT_Port,w=rep(1,length(PGT_Port)),1-alpha) #VaR
CVaRTotal_CA_Port= mean(PGT_Port[which(PGT_Port<VaRTotal_CA_Port)]) #CVaR
print(VaRTotal_CA_Port)
```

```
## [1] -192932.6
```

```r
print(CVaRTotal_CA_Port)
```

```
## [1] -209445.9
```

```r
print(VOT_port)
```

```
## [1] 8629095
```