Projet Gestion des risques

Edi Verbovi, Mohamed Issam Fradi, Timothée Dangleterre

2024-04-28

Introduction

L'objectif de ce projet est de calculer une VaR de crédit à 99% pour un portefeuille d'obligations.

Hypothèses de calcul

Pour réaliser ce calcul de VaR, plusieurs hypothèses sont prises : - toutes les obligations ont la même maturité (5 ans) ; - toutes les obligations ne versent qu'un coupon par an ; - toutes les obligations sont exprimées dans la même devise ; - toutes les obligations ont le même rang de séniorité.

Présentation des fichiers de code

Plusieurs fichiers sont utilisés pour réaliser le calcul de VaR : - un fichier excel - un fichier R

Classeur excel - préparation des données

Le premier fichier à lancer est le classeur excel Data_var_Projet_GDR. Il contient deux macros sur la feuille "Main" : - macro1 - macro2

La première macro lance un code VBA afin de permettre à l'utilisateur de rentrer les caractéristiques de son portefeuille, à savoir : - Le nombre d'obligations noblig - Le rating / taux de coupon / taux de recovery pour chacune de ces obligations

Le fichier excel renseigne également les informations suivantes : - Les ratings possibles pour chaque obligation (feuille Rating) ; - La matrice de transition du modèle Credit Metrics (feuille proba_transition); - La matrice des taux forwards (feuille forward) - La matrice de corrélation (feuille Correlation)

Une fois que l'utilisateur a fini de renseigner les caractéristiques du portefeuille (rating, recovery, coupon sur la feuille Portfolio et correlations entre les titres sur la feuille Correlation), il peut lancer la seconde macro qui permet de lancer l'exécution du fichier R qui fait les différents calcul de CVaR demandés.

Fichiers R

Dans un premier temps, le fichier charge les différentes librairies utilisées pour réaliser les calculs : rstudioapi, readxl, mvtnorm, openxlsx, et définit le répertoire de travail de l'utilisateur.

```
{
  library(rstudioapi)
  library(readxl)
  library(mvtnorm)
  library(openxlsx)
}
```

```r
#Appel des librairies
library(rstudioapi)
library(readxl)
library(mvtnorm)
library(openxlsx)

#Définition du répertoire de travail
setwd(dir = dirname(rstudioapi::getSourceEditorContext()$path))

#Nettoyage de l'interface R
rm(list=ls())
graphics.off()
```

Présentation des fonctions utilisées

Fonction Combi La fonction combi renvoie un dataframe avec l'ensemble des scénarios possibles. Chaque ligne est associé à un scénario j et chaque colonne correspond au rating de l'obligation i dans le scénario j.

Les inputs de la fonction sont : - Le nombre d'obligations du portefeuille noblig - Les ratings possibles pour chaque obligation

La fonction boucle sur le nombre d'obligations pour remplir une liste avec, pour chaque itération, le vecteur de ratings possibles puis calcule tous les scénarios possibles avec la fonction expand.grid)

```r
combi <- function(nb_oblig){

  #Création d'une liste vierge pour récupérer les rating possibles pour chaque obligation
  liste_rating <- list()

  #Vecteur contenant les ratings possibles
  rating <- c("AAA","AA","A","BBB","BB","B","CCC","D")

  #Boucle pour récupérer les ratings possibles pour chaque obligation
  for(i in 1:nb_oblig){
    liste_rating[[length(liste_rating)+1]]<- rating
  }

  #Création d'un dataframe récupérant toutes les combinaisons que l'on récupère
  df_rating <- expand.grid(liste_rating,KEEP.OUT.ATTRS = FALSE ,stringsAsFactors = FALSE)
  return(df_rating)
}
```

Fonction proba La fonction proba calcule la probabilité d'occurence de chaque scénario dans le cas où les obligations sont indépendantes.

Les inputs sont : - n : nombre d'obligations du portefeuille - liste_ri : rating des n obligations aujourd'hui - liste_rf : rating des n obligations dans un an - df_transi : matrice de transition

La fonction boucle sur chaque obligation et récupère, selon le rating initial et le rating final, la probabilité de changement associée donnée par la matrice de transition.

```r
proba <- function(n, liste_ri, liste_rf, df_transition){

  #Création d'un vecteur pour récupérer les proba
  prob <- rep(0, n)
```

```
  #Boucle
  for(i in 1:n){
    prob[i]<-df_transition[liste_ri[[i]], liste_rf[[i]]]
  }

  #Récupération de la proba d'occurence du scénario
  proba_scen <- prod(prob)
  return(proba_scen)
}
```

Fonction forward La fonction forward permet de récupérer un vecteur contenant le prix de chaque obligation dans un an en utilisant les taux forwards associés au rating de l'obligation dans un an.

Les inputs sont : - n : nombre d'obligations - liste_rf : rating des n obligations dans un an - liste_coupon : coupon des n obligations - liste_reco : recovery des n obligations en cas de défaut - df_forward : dataframe contenant les taux forward

```
forward <- function(n, liste_rf, liste_coupon, liste_reco, df_forward){

  #Création d'un vecteur de prix
  prix <- rep(0, n)

  #Double boucle pour calculer le prix des n obligations
  #Première boucle : gestion du cas du défaut et détermination de la rémunération dans un an

  for(i in 1:n){

    #Condition pour vérifier le défaut
    if(liste_rf[[i]]=='D'){

      #Si défaut, prix = recovery * 100
      prix[i]<-liste_reco[[i]]*100

    }else{
      #Calcul de la rémunération dans un an : i * 100
      prix[i] <- liste_coupon[[i]]*100

      #Deuxième boucle sur le dataframe contenant les taux forward pour calculer le prix
      for(j in 1:length(df_forward)){

        #Condition pour vérifier si l'on est à maturité. Lorsque l'on est à maturité, CF = 1+i, sinon C
        if(j == length(df_forward)){
          CF <- 1+liste_coupon[[i]]
        }else{
          CF <- liste_coupon[[i]]
        }

        #Calcul du prix : somme des cash flows actualisés au taux forward
        prix[i]<-prix[i] + (CF*100)/((1+df_forward[liste_rf[[i]], j])^j)
      }


    }
  }
```

```
  #Récupération du vecteur de prix
  return(prix)
}
```

Fonction VaR La fonction VaR permet de trier le dataframe selon les pertes, de calculer la fonction de probabilité cumulée des pertes et de récupérer le quantile associé à 99%. Elle permet de récupérer un tableau trié en fonction des pertes qui contient la distribution des pertes et un tableau contenant spécifiquement la ligne associée à la VaR 99%.

Les inputs sont : - Le dataframe contenant le scénario / proba / valeur du portefeuille / perte associée

```
VaR <- function(df_loss){
  #Tri du dataframe par rapport à la perte
  df_loss <- df_loss[order(df_loss$`Pertes L p/r au scénario sans changement de rating`, decreasing = FA

  cdf <- rep(0, length(df_loss$`Probabilité (%)`))
  cdf[1]<-df_loss$`Probabilité (%)`[1]

  for(l in 2:length(df_loss$`Probabilité (%)`)){
    cdf[l] <- cdf[l-1] + df_loss$`Probabilité (%)`[l]
  }

  #Création de la colonne contenant la fonction de probabilité cumulée
  df_loss['cdf of loss'] <- cdf

  #Récupération des résultats à partir d'une proba de défaut cumulé supérieure à 99%
  df_var_99 <- subset(df_loss, df_loss$`cdf of loss`> 98.99)

  #Récupération de la VAR 99%
  VaR_99 <- df_var_99[1, ]
  return(list(df_loss = df_loss, VaR=VaR_99))
}
```

Fonction seuil_rating La fonction seuil_rating permet de récupérer une liste contenant les dataframes avec les seuils de changement de rating pour chacune des obligations du portefeuille. Ces seuils sont obtenus à partir de la distribution des probabilités de transition.

Les inputs sont : - n : nombre d'obligations - liste_ri : liste contenant le rating des n obligations dans un an - df_p : dataframe contenant les probas de transition

```
seuil_rating <- function(n, liste_ri, df_p){

  #Création d'une liste vierge pour stocker les dataframe des seuils
  liste_df_seuil <- list()

  #Initialisation des probabilités cumulées
  prob_cum <- 0

  #Boucle sur les n obligations
  for(j in 1:n){

    #Création du dataframe des seuils pour l'obligation j
    df_seuil <- as.data.frame(matrix(0, nrow=length(df_p),ncol=3,
                              dimnames=list(c("AAA","AA","A","BBB","BB","B","CCC","D")
                                            ,c("Proba (%)","¨Proba cumulées(%)", "Seuils quantil
```

```
    #Récupération des probabilités de transition selon le rating de l'obligation
    prob_transi <- df_p[liste_ri[[j]], ]

    #Boucle parcourant le rating de D à AAA (probabilité cumulée de D à AAA)
    for(i in length(prob_transi):1){

      #Récupération de la probabilité
      df_seuil[i, 1] <- prob_transi[i]

      #Incrémentation de la distribution cumulée de proba et récupération
      prob_cum <- prob_cum + df_seuil[i, 1]
      df_seuil[i, 2] <- prob_cum

      #Calcul du quantile correspondant et récupération
      df_seuil[i,3] <- qnorm(df_seuil[i,2])

      #Mise en forme des colonnes 1 et 2 (multiplication par 100 pour avoir les valeurs en proba)
      df_seuil[i, 1] <- df_seuil[i, 1] * 100
      df_seuil[i, 2] <- df_seuil[i, 2] * 100
    }

    #Récupération du dataframe contenant les seuils de changement de rating de l'obligation j
    liste_df_seuil[[length(liste_df_seuil)+1]] <- df_seuil

    #Reset de la proba cumulée à 0 pour les obligations suivantes
    prob_cum <- 0
  }

  #Récupération de la liste des dataframe de seuil pour chaque obligation
  return (liste_df_seuil)
}
```

Fonction bornes La fonction bornes permet de récupérer une liste contenant les vecteurs de bornes supérieures et inférieures pour chacune des obligations en fonction de leur rating final.

Les inputs sont : - n : nombre d'obligations en portefeuille ; - liste_rf : liste contenant les ratings des n obligations dans un an ; - liste_df_seuil : liste contenant les dataframe des seuils de changement de rating de chaque obligation.

```
bornes <- function(n, liste_rf, liste_df_seuil){

  #Création de vecteurs vierges pour récupérer les bornes inf et sup
  borne_sup <- rep(0, n)
  borne_inf <- rep(0, n)

  #Boucle sur les ratings
  for(i in 1:n){

    #Affectation du dataframe contenant les seuils de changement de rating pour l'obligation
    df_seuil <- liste_df_seuil[[i]]

    #Gestion du cas où une obligation fait défaut
    if(liste_rf[[i]] == "D"){
      borne_sup[i] <- df_seuil['D', 3]
```

```
      borne_inf[i] <- -Inf

    }else{
      #Parcours les seuils jusqu'à avoir trouvé celui correspondant au nouveau rating
      for(j in 1:nrow(df_seuil)){

        #Condition pour vérifier que le nouveau rating de l'obligation
        if(rownames(df_seuil)[j] == liste_rf[[i]]){

          #Récupération de la borne sup
          borne_sup[i] <- df_seuil[j, 3]

          #Récupération de la borne inf
          borne_inf[i] <- df_seuil[j+1, 3]
        }
      }
    }
  }

  #Récupération des bornes
  return(list(borne_sup = borne_sup, borne_inf = borne_inf))
}
```

Le code est organisé en 5 parties.

Première partie : Récupération des données

Cette partie permet de récupérer, depuis le fichier Excel préalablement rempli, les informations suivantes : - Le nombre d'obligations dans le portefeuille (noblig) - Les coupons pour chaque obligation en portefeuille (liste liste_coupon) - Les ratings initiaux pour chaque obligation en portefeuille (liste liste_ri) - Les taux de recovery pour chaque obligation en portefeuille (liste_reco) - Les ratings possibles pour chaque obligation en portefeuille (vecteur rating) - Les taux forward associés aux années suivantes jusqu'à l'année 5 (forward_rates) - La matrice de transition de Credit metrics (df_transition)

On récupère ensuite le nombre de scénario en fonction du nombre d'obligations dans le portefeuille avec la fonction combi :

```
  #Récupération des ratings possibles pour une obligation et transformation en vecteur
  rating <-(read_excel("Data_var_Projet_GDR.xls", range="Rating!A1:A8"))
  vect_rating <- rating$Rating

  #Récupération du nombre d'obligation en portefeuille
  noblig <-read_excel("Data_var_Projet_GDR.xls", range="Portfolio!D1:D1", col_names = FALSE)[[1]]
```

```
## New names:
## * `` -> `...1`
```

```
  #Récupération des coupon / rating / recovery des n obligations en portefeuille
  df_ptf <- as.data.frame(read_excel("Data_var_Projet_GDR.xls",range=paste("Portfolio!A1:C", as.charact
  liste_coupon <- as.list(df_ptf$Coupon)
  liste_ri <- as.list(df_ptf$Rating)
  liste_reco <- as.list(df_ptf$`Racovery Rate`)

  #Récupération des taux forward selon le rating
  forward_rates <- as.data.frame(read_excel("Data_var_Projet_GDR.xls", range="forward!B1:E8"))
  colnames(forward_rates)<-c(1,2,3,4)
```

```
  rownames(forward_rates)<-vect_rating
  View(forward_rates)

  #Récupération de la matrice de transition
  df_transition <- as.data.frame(read_excel("Data_var_Projet_GDR.xls", range="proba_transition!B1:I8"))
  colnames(df_transition)<-c(vect_rating, "D")
  rownames(df_transition)<-vect_rating
  View(df_transition)

  #Récupération des différents scénarios
  df_scenario <- combi(noblig)
```

Deuxième partie : Calcul de la CVaR avec indépendance

Cette partie se découpe en 4 étapes : - la création d'un dataframe vierge dans lequel on stockera les résultats attendus (scénarios, probabilité d'occurence, valeur du portefeuille, perte et distribution des pertes) - Le calcul de la valeur du portefeuille s'il n'y a aucun changement de rating à l'aide de la fonction forward (cas de base) - Le remplissage du tableau via une boucle : récupération du scénario, calcul de la probabilité via la fonction proba, calcul de la valeur du portefeuille à partir de la fonction forward et enfin le calcul de la perte associée au scénario. - Le tri en fonction des pertes, le calcul de la fonction de probabilité cumulée des pertes et la récupération de la CVaR 99% associée grâce à la fonction VaR.

```
############## Partie 1 : VaR de crédit pour des variables indépendantes
{
#Création du dataframe dans lequel on stockera les résultats
mat <- matrix(0, nrow=length(df_transition)^noblig, ncol=4)
df_loss <- as.data.frame(mat)
colnames(df_loss) <- c("Migrations de rating", "Probabilité (%)", "Prix forwards", "Pertes L p/r au scé

#Récupération des prix du scénario où il n'y a aucun changement de rating
prix_init <- forward(noblig, liste_ri, liste_coupon, liste_reco, forward_rates)

#Récupération de la valeur du portefeuille dans le cas où il n'y a pas de changement de rating
ptf_init <- sum(prix_init)

############### Boucle pour calculer les pertes dans les différents scénarios

#Boucle pour déterminer scénario / prix / proba / perte p/r au scénario initial :
for(i in 1:nrow(df_loss)){

  #Détermination des ratings final selon le scénario et récupération du scénario
  liste_rf <- as.list(df_scenario[i, ])
  scenario <- paste(liste_ri, liste_rf, sep="->",collapse = ", ")

  #Récupération de la probabilité
  prob <- proba(noblig, liste_ri, liste_rf, df_transition)

  #Récupération du prix de chaque obligation
  prix <- forward(noblig, liste_rf, liste_coupon, liste_reco, forward_rates)

  #Récupération de la valeur du portefeuille
  ptf <- sum(prix)

  #Récupération de la perte par rapport au scénario initial
```

```
  perte <- ptf_init - ptf

  ######################################################################
  ################## Récupération des données ##########################
  ######################################################################

  df_loss[i, 1] <- scenario
  df_loss[i, 2] <- prob * 100
  df_loss[i, 3] <- ptf
  df_loss[i, 4] <- perte
}

View(df_loss)

results <- VaR(df_loss)

#Récupération du dataframe contenant les scénarios retraités
df_loss <- results$df_loss
View(df_loss)

#Récupération de la VaR 99% et du scénario associé
VaR_99 <- results$VaR
print(VaR_99)
}
```

```
##       Migrations de rating Probabilité (%) Prix forwards
## 474 AA->AA, BBB->BBB, B->D        4.097706      240.2991
##     Pertes L p/r au scénario sans changement de rating cdf of loss
## 474                                          69.67619    99.24689
```

Troisième partie : Calcul de la CVaR avec corrélation et intégration déterministe

Cette partie se découpe en 4 étapes : - La récupération de la matrice de corrélation et des seuils pour chaque obligation. On crée ensuite un dataframe vierge, identique à celui utilisé dans la première partie, pour récupérer les résultats. - Comme dans la partie 1, une boucle permet de remplir le tableau. Le scénario est reporté dans le tableau puis récupération des bornes supérieures et inférieures en fonction du rating final des différentes obligations. La probabilité d'occurence est ensuite calculée ainsi que la valeur du portefeuille ce qui permet d'en déduire les pertes par rapport au scénario central. - Comme dans la partie 1, le tableau est trié selon les pertes pour calculer la fonction de probabilité cumulée des pertes et la CVaR 99% est récupérée, via la fonction VaR.

```
############# Partie 2 : Modèle Credit Metrics
{

  #Création de la matrice des corrélations
  #Création de la matrice des corrélations
  mat_corr <-(read_excel("Data_var_Projet_GDR.xls", sheet='Correlation'))
  rat_oblig <- colnames(mat_corr[1:noblig+1])
  mat_corr <- mat_corr[1:noblig,2:(noblig+1)]
  corr <- as.matrix(mat_corr)
  colnames(corr) <- rat_oblig
  rownames(corr)<-rat_oblig


  #corr <- matrix(data=c(1,0.3,0.1,0.3,1,0.2,0.1,0.2,1),nrow=3,ncol=3, dimnames=list(c("AA","BBB","B"),
```

```r
#Récupération des seuils pour chaque obligation
liste_seuil <- seuil_rating(noblig, liste_ri, df_transition)

#Création du dataframe qui contiendra les résultats (df_loss_2)
df_loss_2 <- as.data.frame(matrix(0, nrow=length(df_transition)^3, ncol=4))
colnames(df_loss_2) <- c("Migrations de rating", "Probabilité (%)", "Prix forwards", "Pertes L p/r au

#Boucle pour déterminer la valeur du portefeuille et les pertes dans les différents scénarios
for (i in 1:nrow(df_scenario)){

  #Récupération des ratings finaux et du scénario associé
  liste_rf <- as.list(df_scenario[i, ])
  scenario <- paste(liste_ri, liste_rf, sep="->",collapse = ", ")

  #Récupération des bornes
  liste_bornes <- bornes(noblig, liste_rf, liste_seuil)

  #Récupération de l'encadrement de seuils de rating pour chaque obligation
  borne_sup <- liste_bornes$borne_sup
  borne_inf <- liste_bornes$borne_inf

  #Calcul de la probabilité associée au scénario
  proba <- pmvnorm(lower=borne_inf, upper=borne_sup, mean=rep(0,noblig),corr=corr)

  #Calcul du prix associé au scénario
  #Récupération du prix de chaque obligation
  prix <- forward(noblig, liste_rf, liste_coupon, liste_reco, forward_rates)

  #Récupération de la valeur du portefeuille
  ptf <- sum(prix)

  #Récupération de la perte par rapport au scénario initial
  perte <- ptf_init - ptf

  ######################################################################
  ################### Récupération des données #########################
  ######################################################################

  df_loss_2[i, 1] <- scenario
  df_loss_2[i, 2] <- proba * 100
  df_loss_2[i, 3] <- ptf
  df_loss_2[i, 4] <- perte

}
View(df_loss_2)

results <- VaR(df_loss_2)

#Récupération du dataframe contenant les scénarios retraités
df_loss_2 <- results$df_loss
View(df_loss_2)

#Récupération de la VaR 99% et du scénario associé
```

```
  VaR_99_2 <- results$VaR
  print(VaR_99_2)
}
```

```
## New names:
## * `` -> `...1`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...11`
## * `` -> `...12`
## * `` -> `...13`
## * `` -> `...14`
## * `` -> `...15`
## * `` -> `...16`
## * `` -> `...17`
## * `` -> `...18`
## * `` -> `...19`
## * `` -> `...20`
## * `` -> `...21`
## * `` -> `...22`
## * `` -> `...23`
## * `` -> `...24`
## * `` -> `...25`
## * `` -> `...26`
## * `` -> `...27`
## * `` -> `...28`
## * `` -> `...29`
## * `` -> `...30`
```

```
##      Migrations de rating Probabilité (%) Prix forwards
## 475 AA->A, BBB->BBB, B->D       0.4199151      239.7912
##     Pertes L p/r au scénario sans changement de rating cdf of loss
## 475                                             70.184    99.23346
```

Quatrième partie : Calcul de la CVaR via des simulations de Monte-Carlo

Cette partie se découpe en 4 étapes :

- La génération de n vecteurs gaussiens contenant 10000 valeurs par la méthode de Monte-Carlo.
- On réalise ensuite une triple boucle. La première permet de boucler sur chacun des 10000 scénarios simulés. La seconde permet de récupérer les seuils associés à chacune des n obligations. La troisième compare les seuils obtenus par Monte-Carlo aux seuils de changement de rating afin de récupérer les bornes inférieures et supérieures associées.

Une fois ce travail effectué, le code reporte chaque scénario. La probabilité d'occurence est calculée de la même manière que dans la partie 2, la valeur du portefeuille associée est récupérée par la fonction forward ce qui permet d'obtenir la perte associée à chaque scénario.

- Le calcul de la CVaR ne se fait cette fois pas à partir de la distribution cumulée des pertes. À la place, on trie le dataframe en fonction des pertes et on récupère le quantile associée, soit la 99%*1000 = 990e ligne.

```
############## Partie 3 : Simulation de Monte-Carlo
{
  #Installation du package MASS
  #install.packages("MASS")
  library(MASS)

  #Génération de 10000 seuils par simulation
  MC_seuil <- mvrnorm(n=10000, rep(0,noblig), corr)

  #Création d'un vecteur pour récupérer les nouveaux ratings
  ratings <- rep(0,noblig)

  #Création d'un vecteur pour récupérer les bornes sup et inf
  borne_sup <- rep(0,noblig)
  borne_inf <- rep(0,noblig)

  #Création du dataframe qui contiendra les résultats (df_loss_2)
  df_loss_3 <- as.data.frame(matrix(0, nrow=length(MC_seuil[,1]), ncol=3))
  colnames(df_loss_3) <- c("Migrations de rating", "Prix forwards", "Pertes L p/r au scénario sans chan

  #Boucle pour récupérer le scénario et les bornes
  for(i in 1:length(MC_seuil[,1])){

    #Boucle pour récupérer le dataframe contenant les seuils pour chaque obligation
    for(j in 1:length(liste_seuil)){
      df_seuil<-liste_seuil[[j]]

      #Boucle sur les lignes du dataframe des seuils pour identifier le nouveau rating
      for(k in length(df_seuil[,1]):1){

        #Condition sur le défaut
        if(MC_seuil[i,j]<df_seuil['D',3]){

          ratings[j]<-'D'
          borne_sup[j]<-df_seuil['D',3]
          borne_inf[j]<- -Inf

          #Condition : dès que la valeur de MC est supérieure au seuil, on récupère le rating (part des
        } else if(MC_seuil[i, j]<df_seuil[k,3]){

          #Récupération du rating
          ratings[j] <- rownames(df_seuil[k,])

          #Récupération des bornes
          borne_sup[j] <- df_seuil[k,3]
          borne_inf[j] <- df_seuil[k+1,3]

          #Sortie de la boucle
          break
        }

      }
    }
```

```
    #Récupération du scénario
    liste_rf <- as.list(ratings)
    scenario <- paste(liste_ri, liste_rf, sep="->",collapse = ", ")

    #Récupération de la probabilité
    proba <- pmvnorm(lower=borne_inf, upper=borne_sup, mean=rep(0,noblig),corr=corr)

    #Récupération du prix
    prix <- forward(noblig, liste_rf, liste_coupon,liste_reco, forward_rates)

    #Récupération de la valeur du portefeuille
    ptf <- sum(prix)

    #Récupération de la perte par rapport au scénario initial
    perte <- ptf_init - ptf

    ########################################################################
    ################### Récupération des données ###########################
    ########################################################################

    df_loss_3[i, 1] <- scenario
    df_loss_3[i, 2] <- ptf
    df_loss_3[i, 3] <- perte
  }

  #Tri des données en fonction des pertes
  df_VaR <- df_loss_3[order(df_loss_3$`Pertes L p/r au scénario sans changement de rating`, decreasing =

  #Récupération de la VaR
  VaR_99_3 <- df_VaR[9900, ]
  print(VaR_99_3)
}
```

```
##       Migrations de rating Prix forwards
## 6243 AA->A, BBB->BBB, B->D      239.7912
##      Pertes L p/r au scénario sans changement de rating
## 6243                                            70.184
```

Cinquième partie :

Maintenant que tous les calculs ont été réalisés, on exporte les résultats sous excel.

```
############## Exportation des résultats
{

  df_VaR_1 <- as.data.frame(VaR_99)
  df_VaR_2 <- as.data.frame(VaR_99_2)
  df_VaR_3 <- as.data.frame(VaR_99_3)
  list_of_data <- list("CDF - partie 1"=df_loss,
                       "CDF - partie 2"=df_loss_2,
                       "CDF - partie 3" = df_VaR,
                       "VaR partie 1" = df_VaR_1,
                       "VaR partie 2"= df_VaR_2,
                       "VaR partie 3"=df_VaR_3)
```

```
  write.xlsx(list_of_data, file="Résultats_VAR.xlsx")
}
```