ALM

La gestion du risque de liquidité

Approche ALM avec "rstudio"

Encadré par Mme: AKDIM Khadija

Realisé par:EL OUAHMANI Karima

EL AMIRI Abdelaziz

26.10.2021

Contents

1 Préparation de la base de donnée:

Soit "Balance sheet" la source de données qui répertorie les postes du bilan.

id	account	account_name	volume	ir_binding	reprice_freq	spread	issue	maturity	repayment	payment_freq	yieldcurve
1	cb_1	Cash and balances with central ban	930	FIX		5	09/30/2014	10/01/2014	BULLET	1	EUR01
2	mmp_1	Money market placements	1404	FIX		7	08/30/2014	11/30/2014	BULLET	1	EUR01
3	mmp_1	Money market placements	996	FIX		10	06/15/2014	12/15/2014	BULLET	1	EUR01
4	cl_1	Corporate loans	515	LIBOR	3	301	05/15/2014	04/15/2016	LINEAR	3	EUR01
5	cl_1	Corporate loans	655	LIBOR	6	414	04/15/2014	04/15/2016	LINEAR	6	EUR01
6	cl_1	Corporate loans	560	LIBOR	3	345	03/15/2014	02/15/2018	LINEAR	3	EUR01
7	cl_1	Corporate loans	260	LIBOR	3	328	02/15/2014	04/15/2015	LINEAR	3	EUR01
8	cl_1	Corporate loans	194	LIBOR	3	378	01/15/2014	02/15/2018	LINEAR	3	EUR01
9	cl_1	Corporate loans	267	LIBOR	6	428	12/15/2013	01/15/2018	LINEAR	6	EUR01
10	cl_1	Corporate loans	463	LIBOR	3	332	11/15/2013	01/15/2016	LINEAR	3	EUR01
11	cl_1	Corporate loans	332	LIBOR	6	350	10/15/2013	02/15/2016	LINEAR	6	EUR01
12	cl_1	Corporate loans	518	LIBOR	3	424	09/15/2013	10/15/2014	LINEAR	3	EUR01
13	cl_1	Corporate loans	188	LIBOR	3	363	08/15/2013	06/15/2016	LINEAR	3	EUR01
14	cl_1	Corporate loans	149	LIBOR	6	414	07/15/2013	06/15/2018	LINEAR	6	EUR01
15	cl_1	Corporate loans	560	LIBOR	3	314	06/15/2013	11/15/2015	LINEAR	3	EUR01

Figure 1.1: L'actif d'un bilan bancaire simplifié.

118	mmt_1	Unsecured money market funding	-421	LIBOR	3	22	10/31/2014	07/31/2015	BULLET	1	EUR01
119	mmt_1	Unsecured money market funding	-271	LIBOR	6	29	09/30/2014	05/31/2015	BULLET	1	EUR01
120	mmt_1	Unsecured money market funding	-509	LIBOR	9	34	09/15/2014	06/15/2015	BULLET	1	EUR01
121	rep_1	Repurchase agreements	-500	FIX		12	09/15/2014	10/15/2014	BULLET	1	EUR01
122	is_1	Own issues	-651	LIBOR	3	105	03/15/2014	03/15/2019	BULLET	1	EUR01
123	is_1	Own issues	-1023	LIBOR	3	120	09/15/2013	09/15/2018	BULLET	1	EUR01
124	is_1	Own issues	-1093	LIBOR	3	156	03/15/2013	03/15/2018	BULLET	1	EUR01
125	is_1	Own issues	-577	LIBOR	3	143	09/15/2012	09/15/2017	BULLET	1	EUR01
126	is_1	Own issues	-1822	LIBOR	3	110	03/15/2012	03/15/2017	BULLET	1	EUR01
127	is_1	Own issues	-40	LIBOR	3	132	09/15/2011	09/15/2016	BULLET	1	EUR01
128	is_1	Own issues	-482	LIBOR	3	122	03/15/2011	03/15/2016	BULLET	1	EUR01
129	is_1	Own issues	-283	LIBOR	3	124	09/15/2010	09/15/2015	BULLET	1	EUR01
130	is_1	Own issues	-2379	LIBOR	3	130	03/15/2010	03/15/2015	BULLET	1	EUR01
131	cor_sd_1	Corporate sight deposit	-13000	FIX		10	09/30/2014	09/30/2019	LINEAR	1	EUR01
132	ret_sd_1	Retail sight deposit	-22000	FIX		15	09/30/2014	09/30/2024	LINEAR	1	EUR01
133	cor_td_1	Corporate term deposit	-1087	FIX		28	08/30/2014	01/30/2015	BULLET	1	EUR01
134	cor_td_1	Corporate term deposit	-2393	FIX		28	07/30/2014	12/30/2014	BULLET	1	EUR01
135	cor_td_1	Corporate term deposit	-805	FIX		33	06/30/2014	01/30/2015	BULLET	1	EUR01
136	cor_td_1	Corporate term deposit	-437	FIX		30	05/30/2014	11/30/2014	BULLET	1	EUR01
137	cor_td_1	Corporate term deposit	-2278	FIX		40	04/30/2014	02/28/2015	BULLET	1	EUR01

Figure 1.2: Passif d'un bilan bancaire simplifié.

L'ensemble de données du bilan contient:

1. L'identifiant du produit
2. L'identifiant du compte (ou son abréviation)
3. Le nom du compte.

Les types de comptes qui sont liés aux produits typiques des banques commerciales ou aux postes du bilan sont :

- Available for sale portfolio
- Cash and balances with central bank
- Corporate loans
- Corporate sight deposit
- Corporate term deposit
- Money market placements
- Other non-interest bearing assets
- Other non-interest bearing liabilities
- Own issues
- Repurchase agreements
- Retail overdrafts 2
- Retail residential mortgage
- Retail sight deposit
- Retail term deposit
- Unsecured money market funding

L'ensemble de données du bialn contient également:

1. Le volume notionnel en EUR,
2. Le type de fixation des intérêts (FIX ou LIBOR),
3. La fréquence de refixation des prix du compte en nombre de mois (si la fixation des intérêts est LIBOR)
4. La composante spread du taux d'intérêt en points de base.
5. La date d'émission (il s'agit du premier jour de refixation des prix),
6. La date d'échéance
7. Le type de structure de remboursement du principal (in fine, linéaire ou annuité)
8. La fréquence de remboursement en nombre de mois.
9. L'identifiant de la courbe des taux d'intérêt(yieldcurve), que nous utilisons pour le calcul des futurs paiements variables

Nous avons des données historiques et nous nous situons comme date de départ de notre étude "09/30/2014".

Soit maintenant "Market data" la base de données qui comporte l'ensemble de données du marché et qui contient les taux d'intérêt réels.

type	date	rate	comment
EUR01	09/01/2014	0.3	1M
EUR01	12/01/2014	0.336255845	3M
EUR01	03/01/2015	-2.353646265	6M
EUR01	09/01/2015	-5.69187635	1Y
EUR01	09/01/2016	-5.654177435	2Y
EUR01	09/01/2017	1.015957599	3Y
EUR01	09/01/2018	12.10951189	4Y
EUR01	09/01/2019	25.95057231	5Y
EUR01	09/01/2020	41.28355798	6Y
EUR01	09/01/2021	57.18267728	7Y
EUR01	09/01/2022	72.97941716	8Y
EUR01	09/01/2023	88.20463868	9Y
EUR01	09/01/2024	102.5424837	10Y
EUR01	09/01/2025	115.7938149	11Y
EUR01	09/01/2026	127.847336	12Y
EUR01	09/01/2027	138.656884	13Y
EUR01	09/01/2028	148.2236714	14Y
EUR01	09/01/2029	156.5824857	15Y
EUR01	09/01/2030	163.7910444	16Y
EUR01	09/01/2031	169.9218561	17Y
EUR01	09/01/2032	175.0560664	18Y
EUR01	09/01/2033	179.2788658	19Y
EUR01	09/01/2034	182.6761244	20Y
EUR01	09/01/2035	185.3319805	21Y

Figure 1.3: Market interest database .

La colonne type, indique le type de courbe de rendement, La colonne date indique l'échéance du taux actuel, et rate indique la valeur du taux en points de base. Comme vous pouvez le voir, la courbe de rendement est très inhabituelle en ce moment car il y a des points de courbe de rendement négatifs pour certaines échéances.

2 Génération des cash-flow:

Nous avons programmée une fonction sous R pour génerer les cash-flow de la banque en fonction des types de paiement (Paiement in fine "Bullet", Paiement Linéaire "A amortissement constant", Paiement avec annuité "A annuité constante")

```
cf<-function(rate=0, maturity=1, volume=1, type)
{
  if (type=="BULLET")
  {
    interest          <- rep_len(rate,maturity)*volume
    capital           <- rep_len(0, maturity)
    capital[maturity] <- volume
    remaining         <- c(volume, volume - capital)
    cashflow          <- capital + interest
  }
  if (type=="LINEAR")
  {
    capital           <- rep_len(1/maturity,maturity)*volume
    remaining         <- c(volume,volume-cumsum(capital))
    interest          <- head(remaining,maturity)*rate
    cashflow          <- capital + interest
  }
  if (type=="ANNUITY")
  {
    interest          <- rep_len(0, maturity)
    capital           <- rep_len(0, maturity)
    cashflow          <- rep_len(0, maturity)
    remaining         <- volume
    if (length(rate==1))  {rate <- rep_len(rate, maturity)}
    for (i in 1:maturity)
    {
      cashflow[i]     <- (rate[i] / (1 - (1+rate[i])^(-(maturity-i+1) )))*remaining[i]
      if (rate[i]==0){cashflow[i]<-1/(maturity-i+1)*remaining[i]}
      interest[i]     <- remaining[i]*rate[i]
      capital[i]      <- cashflow[i] -  interest[i]
      remaining[i+1]  <- remaining[i] - capital[i]
    }
  }
  cf <- structure(list(cashflow=cashflow, interest=interest,
                       capital=capital, remaining=remaining[2:(maturity+1)]))
  return(cf)
}
```

Figure 2.1: la fonction génératrice des Cash-Flow.

Nous avons programmé par la suite une fonction de la courbe de rendement "get.yieldcurve.spot". Elle fournit une courbe de rendement au comptant ajustée à une certaine séquence de dates.

```
get.yieldcurve.spot<- function(market, dates, type="EUR01", now=Sys.Date(), showplot=FALSE)
{
  market.maturity  = as.vector( (market$date[market$type==type]-now)/365.25 )
  market.yields    = xts( t(market$rate[market$type==type]), now )
  maturity         = as.vector( (as.Date(dates)-now)/365.25 )
  coeffs           = Svensson(market.yields, market.maturity)
  yieldcurve.spot  = data.frame(date=as.Date(dates),
                                rate=as.numeric(t(Srates(coeffs, maturity, "Spot" ))) )

  if (is.na(yieldcurve.spot$rate[1])==TRUE) {yieldcurve.spot$rate[1]=yieldcurve.spot$rate[2]}

  if (showplot==TRUE) {
    plot(market.maturity, market.yields,
         xlab=c("Pillars in years"), ylab="rate", type="p", col=3)
    title(main="Fitted yield curve (Svensson method)",
          sub =paste("Actual date: ",as.character(as.Date(NOW))),
          cex=0.8)
    lines(maturity, yieldcurve.spot$rate,col=2, type="l")
    legend("topleft",
           legend=c("observed yield curve","fitted yield curve"),
           col=c(3,2),lty=c(0,1), pch=c("o",""), bty = "n")
    grid()           }

  return(yieldcurve.spot)
}
```

Figure 2.2: The spot yield curve function.

La courbe des taux spot va nous permettre de dériver la courbe des rendements forward "get.yieldcurve.forward"

```
get.yieldcurve.forward<- function(market, dates, type="EUR01", now=Sys.Date())
{
  yieldcurve.spot    = get.yieldcurve.spot(market, dates, type=type, now=now)

  maturity           = as.vector((as.Date(dates)-now)/365.25)
  yieldcurve.forward = data.frame(date=dates,
                                  rate=  (
                                    (1+yieldcurve.spot$rate/10000)^maturity /
                                      c(1,((1+yieldcurve.spot$rate/10000)^maturity)[1:length(maturity)-1])
                                    -1)*10000  )

  return(yieldcurve.forward)
}
```

Figure 2.3: The forward yield curve function.

La courbe des taux forward a été crée dans le but de générer les cash-flows des produits variables.Et ceci à l'aide de la fonction "get.floating"

```
get.floating <- function(market, payment.dates, reprice.dates,
                         type="EURO1", now=Sys.Date(), showplot=FALSE)
{

  yieldcurve.forward.original = get.yieldcurve.forward(market, reprice.dates,
                                                       type="EURO1", now=Sys.Date())
  yieldcurve.forward = merge(payment.dates,
                             yieldcurve.forward.original, by=1, all.x=TRUE, all.y=TRUE)

  yieldcurve.spot = get.yieldcurve.spot(market, reprice.dates,
                                        type="EURO1", now=Sys.Date())
  yieldcurve.forward$rate[1]=yieldcurve.spot$rate[1]
  for (i in 1:NROW(yieldcurve.forward))
  { if (is.na(yieldcurve.forward$rate[i])==TRUE)
  { yieldcurve.forward$rate[i]<-yieldcurve.forward$rate[i-1] }
  }

  floating = merge(payment.dates,
                             yieldcurve.forward, by=1, all.x=TRUE)

  if (showplot==TRUE) {
    plot(yieldcurve.forward.original, type="b", col="green",
         xlab="Maturity", ylab="rate")
    lines(floating, type="s", col="red")
    title(main="Forward curve and floating rate forecast",
          sub =paste("Actual date: ",as.character(as.Date(NOW))),
          cex=0.8)
    legend("topleft",
           legend=c("forward yield curve","forecasted floating rate"),
           col=c(3,2),lty=c(1,1), pch=c("o",""), bty = "n")
    grid()             }

  return(floating)
}
```

Figure 2.4: The floating cash-flows function.

3 Les cash-flows du bilan

Maintenant que toutes les fonctions génératrices des cash-flows ont été programées, il est le temps de les appliquer sur notre bilan bancaire à travers la fonction cf.table.

```
cf.table<-function(portfolio, market, now=Sys.Date(), id)
{
  payment.periods = sort(seq(from = as.Date(portfolio$maturity[id],    format = "%m/%d/%Y"), to    = now+1,
                             by   = "-1 month") )
  payment.freq  = ifelse(is.na(portfolio$payment_freq[id]),1,portfolio$payment_freq[id])
  payment.dates = payment.periods[seq(from=1,
                                      to  =length(payment.periods),
                                      by  =payment.freq)]
  if (portfolio$ir_binding[id]=="FIX") {
    r = portfolio$spread[id]
  }  else  {
    reprice.periods  =  seq(from = as.Date(portfolio$issue[id],    format = "%m/%d/%Y"),
                            to   = as.Date(portfolio$maturity[id],    format = "%m/%d/%Y"),
                            by   = "1 month")
    reprice.freq = ifelse(is.na(portfolio$reprice_freq[id]),1,portfolio$reprice_freq[id])
    reprice.dates = reprice.periods[seq(from=1, to  =length(reprice.periods), by  =reprice.freq)]
    reprice.dates =reprice.dates[reprice.dates>=now]
    if (length(reprice.dates)>0) {
      floating = get.floating(market, payment.dates, reprice.dates)
      r        = portfolio$spread[id] + floating$rate
    } else{ r = portfolio$spread[id] }
  }
  cft = cf(rate     = r/10000*(payment.freq/12),maturity = length(payment.dates), volume   = portfolio$volume[id],
           type     = as.character(portfolio$repayment[id]))
  cf.table <- data.frame(
    id      =portfolio$id[id],
    account =portfolio$account[id],
    date    =payment.dates,
    cf      =cft$cashflow,
    interest=cft$interest,
    capital =cft$capital,
    remaining=cft$remaining)
  return(cf.table)
}
```

Figure 3.1: Balance sheet's CF generating function.

La fonction cf.table fournit un cash-flow de chaque compte figurant sur le bilan .

id	account	date	cf	interest	capital	remaining
31	ro_1	2014-10-30	466.666667	50.00000000	416.666667	4.583333e+03
31	ro_1	2014-11-30	462.500000	45.83333333	416.666667	4.166667e+03
31	ro_1	2014-12-30	458.333333	41.66666667	416.666667	3.750000e+03
31	ro_1	2015-01-30	454.166667	37.50000000	416.666667	3.333333e+03
31	ro_1	2015-03-02	450.000000	33.33333333	416.666667	2.916667e+03
31	ro_1	2015-03-30	445.833333	29.16666667	416.666667	2.500000e+03
31	ro_1	2015-04-30	441.666667	25.00000000	416.666667	2.083333e+03
31	ro_1	2015-05-30	437.500000	20.83333333	416.666667	1.666667e+03
31	ro_1	2015-06-30	433.333333	16.66666667	416.666667	1.250000e+03
31	ro_1	2015-07-30	429.166667	12.50000000	416.666667	8.333333e+02
31	ro_1	2015-08-30	425.000000	8.33333333	416.666667	4.166667e+02
31	ro_1	2015-09-30	420.833333	4.16666667	416.666667	0.000000e+00
32	rm_1	2014-10-30	6.028201	2.62573027	3.402471	6.425975e+02
32	rm_1	2014-11-30	6.028201	2.61190059	3.416301	6.391812e+02
32	rm_1	2014-12-30	6.028201	2.59801470	3.430187	6.357510e+02
32	rm_1	2015-01-30	6.028201	2.58407237	3.444129	6.323069e+02
32	rm_1	2015-03-02	6.106467	2.70106054	3.405406	6.289015e+02
32	rm_1	2015-03-30	6.106467	2.68651348	3.419953	6.254816e+02
32	rm_1	2015-04-30	6.106467	2.67190427	3.434563	6.220470e+02

Figure 3.2: Tableau des cash-flow des comptes du bilan.

4 Mesure du risque de liquidité

Parmi les méthodes utilisées pour mesurer le risque de liquidité: La mesure de volume (Impasse prévisionnelle) Rappellons qu'une impasse constitue la différence entre les emplois et les ressources des postes du bilan pour un ensemble de processus d'opérations, à une date ultérieure prédéfinie.

$$Total = GAP = \sum_{i=1}^{n} (Actif(t_i) - Passif(t_i))$$

Le tableau ci-dessous montre l'écoulement des cash-flows dans le temps. Le total désigne le gap de liquidité :

	1M	2-3M	3-6M	6-12M	1-2Y	2-5Y	5-10Y	>10Y
afs_1	2.4827667	3068.5059333	14939.4216667	0.0000	0.0000	0.000	0.0000	0.0000
cb_1	930.0387500	0.0000000	0.0000000	0.0000	0.0000	0.000	0.0000	0.0000
cl_1	3111.1312549	0.0000000	650.2485770	2224.7136	2831.7267	1651.805	0.0000	0.0000
cor_sd_1	-217.7500000	-217.7319444	-653.0875000	-1305.6875	-2609.4250	-7812.675	-216.6847	0.0000
cor_td_1	-1.9019583	-439.6612917	-6566.0270500	0.0000	0.0000	0.000	0.0000	0.0000
is_1	-8.6947450	-18.0201888	-2407.5107815	-323.5708	-589.6882	-5245.904	0.0000	0.0000
mmp_1	0.1649000	2400.2479000	0.0000000	0.0000	0.0000	0.000	0.0000	0.0000
mmt_1	-0.1216195	-0.5372018	-0.9365594	-1202.5828	0.0000	0.000	0.0000	0.0000
oth_a_1	0.0000000	0.0000000	0.0000000	0.0000	0.0000	0.000	0.0000	2300.0000
oth_l_1	0.0000000	0.0000000	0.0000000	0.0000	0.0000	0.000	0.0000	-3930.0000
rep_1	-500.0500000	0.0000000	0.0000000	0.0000	0.0000	0.000	0.0000	0.0000
ret_sd_1	-186.0833333	-186.0604167	-558.0437500	-1115.4688	-2228.4625	-6665.588	-10859.9333	-366.7354
ret_td_1	-4038.9578500	-5.3393500	-5358.1323000	-3382.9076	0.0000	0.000	0.0000	0.0000
rm_1	414.4079365	808.8792518	1248.1807305	2112.9722	4993.7710	14540.834	15965.8503	3523.8387
ro_1	466.6666667	462.5000000	1362.5000000	2612.5000	420.8333	0.000	0.0000	0.0000
total	-28.6672315	5872.7826917	2656.6130333	-380.0316	2818.7553	-3531.528	4889.2323	1527.1032

Figure 4.1: Liquidity table

```r
lq.table<-function(cashflow.table, now=Sys.Date())
{
  periods.days  <- c(0,30,90,180,360,720,1800,3600,7200)
  periods.names <- c("1M","2-3M","3-6M","6-12M","1-2Y","2-5Y","5-10Y",">10Y")

  N           <- nlevels(as.factor(cashflow.table$account))
  table       <- cast(cashflow.table, date ~ account,  value="cf", fun=sum)
  table       <- table[table$date>=now,]
  table$total <- as.vector(t(rowSums(table[,2:(2+N-1)], na.rm=TRUE)))
  table$days  <- table$date-now
  table$period  <- cut(as.numeric(table$days),
                       breaks=as.numeric(periods.days))

  lq.table<-aggregate(table[,2:(2+N)],
                      list(time=table$period), sum, na.rm=TRUE)
  rownames(lq.table)<-periods.names[1:length(rownames(lq.table))]
  lq.table          <- t(subset(lq.table, select = -c(time) ))

  lqt<-data.frame(id=row.names(lq.table),
                  M1=lq.table[,2],
                  M2=lq.table[,3],
                  M3=lq.table[,4],
                  M6=lq.table[,5],
                  M7=lq.table[,6],
                  M8=lq.table[,7],
                  M9=lq.table[,8])
  return(lq.table)
}
```

Figure 4.2: Liquidity table code in r

```r
lq      <- lq.table(cashflow.table, now=NOW)
View(lq)
#

#plot
plot.new()
par.backup<-par()
par(oma = c(1, 1, 1, 6), new=TRUE)
lq.bar<-barplot(lq[1:(NROW(lq)-1),], col=heat.colors(NROW(lq)-1),
                xlab="Maturity", cex.names=0.8,
                ylab="EUR", cex.axis=0.8,
                args.legend = list(x = "right"))
title(main="Liquidity gap table",
      sub=paste("Actual date: ",as.character(as.Date(NOW))), cex=0.8 )
lines(x=lq.bar,y=lq[NROW(lq),],lwd=4, col="red", lty=5, type="b", pch=0 )
lines(x=lq.bar,y=cumsum(lq[NROW(lq),]),lwd=4, col="black", type="b" )

par(fig = c(0, 1, 0, 1), oma = c(0, 0, 0, 0),mar = c(0, 0, 0, 0),  new = TRUE)
plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")
legend("right", legend = row.names(lq[1:(NROW(lq)-1),]),
       fill = heat.colors(NROW(lq)-1), bty = "n", cex=1)
par(par.backup)
```

Figure 4.3: Liquidity gap table code in r

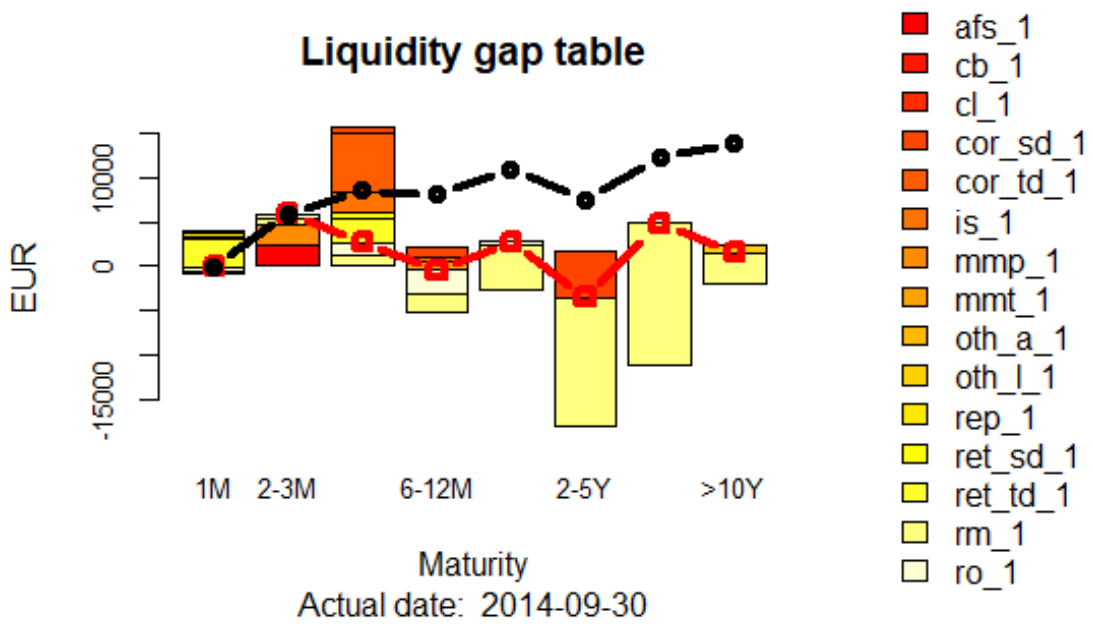

Figure 4.4: GAP de liquidité dans chaque tranche de temps.

La ligne pointillée avec des carrés représente la position de liquidité nette (besoin financier), tandis que la ligne noire continue montre GAP de liquidité cumulé.