

DM

CASTANO Nicolas et LAPLANTE Juliette

19 novembre 2018

Question 1

- Si m est la stratégie qui consiste à jouer systématiquement la machine **A** alors $G_m(T)$ vaut $\sum_{t=1}^T X_{m(t),t} = \sum_{t=1}^T X_{A,t} = \sum_{t=1}^T B(pA) = Bi(T, pA)$ avec $m(t) = A \forall t \in [1, T]$
Ainsi, $E[G_m(T)] = T \times pA$.
- Si m est la stratégie qui consiste à jouer systématiquement la machine **B** alors, de façon analogue à **A**, $E[G_m(T)] = T \times pB$ avec $m(t) = B \forall t \in [1, T]$
- Si m est la stratégie qui consiste à jouer la machine **A** avec une probabilité de 1/2 et la machine **B** avec une probabilité de 1/2 alors on jouera en moyenne $E[B(T, \frac{1}{2})] = \frac{T}{2}$ fois sur la machine A et $1 - \frac{T}{2}$ fois sur la machine B.

$$\text{On a donc : } G_m(T) = \sum_{t=1}^T X_{m(t),t} = \sum_{t=1}^{\frac{T}{2}} X_{A,t} + \sum_{t=\frac{T}{2}+1}^T X_{B,t} = Bi(\frac{T}{2}, pA) + Bi(\frac{T}{2}, pB)$$

On a alors

$$E[G_m(T)] = E[Bi(\frac{T}{2}, pA) + Bi(\frac{T}{2}, pB)] = E[Bi(\frac{T}{2}, pA)] + E[Bi(\frac{T}{2}, pB)] = \frac{T \times pA}{2} + \frac{T \times pB}{2}$$

On s'intéresse au regret de la stratégie m car le regret est une manière de quantifier l'écart entre la probabilité théorique de gagner et la valeur réelle obtenue lors d'une expérience.

Ici, le gain vaut soit 0, soit 1 euro, la probabilité de gagner une épreuve correspondra donc au gain moyen. Si notre stratégie nous fait en moyenne moins gagner que la probabilité théorique pA , alors on adopte une mauvaise stratégie (si notre nombre d'échantillon est grand).

Une "bonne stratégie" fait tendre le regret vers 0. Soit pA (resp. pB) la probabilité de gagner en jouant sur la machine **A** (resp. **B**), notre regret va être compris entre 0 et $pA - (T \times pB / T) = pA - pB = 1/10$.

```
library(ggplot2)
```

```
pA = 0.65
```

```
pB = 0.55
```

```
Tmax = 1000
```

```
epsilon=0.05
```

Question 2

Intuition : Cette méthode semble "logique". On étudie quel machine semble la plus rentable ($\epsilon \times T$ premiers essais) et selon le résultat on joue les parties suivantes ($T - (\epsilon \times T)$ derniers essais) sur la même machine (celle qui a donné les meilleurs résultats pendant la phase d'essai).

Il faudra donc faire attention à avoir une phase de tests avec assez d'échantillons pour ne pas se tromper de machine. L'efficacité de cette stratégie augmente avec l'écart entre les probabilités de victoire des machines **A** et **B**. En effet, si les probabilités de victoire de **A** et **B** sont proches, la différence entre les gains totaux des jeux sur chaque machine sera faible. L'impact de la mauvaise sélection de la machine à jouer dans ces cas est donc faible.

Programme :

```

# Quelques stratégies présentées dans la dm
#On joue sur la machine A
onlyA = function(Tmax,pA,pB){
  vals=sample(c(TRUE,FALSE),Tmax,replace = TRUE, prob = c(pA,1-pA))
}

#On joue sur la machine B
onlyB = function(Tmax,pA,pB){
  vals=sample(c(TRUE,FALSE),Tmax,replace = TRUE, prob = c(pB,1-pB))
}

#On joue la machine A avec une probabilité de 1/2 et la machine B avec une probabilité 1/2
half = function(Tmax,pA,pB){
  #Si le boolean vaut vrai alors on joue sur la machine A sinon sur la machine B
  tabChoixMachine=sample(c(TRUE,FALSE),Tmax, replace = TRUE, prob=c(0.5,0.5))
  #calcul du nombre de partie sur la machine A
  nbEpreuveSurMachineA=sum(tabChoixMachine)
  #calcul du nombre de partie sur la machine B
  nbEpreuveSurMachineB=Tmax-nbEpreuveSurMachineA

  vals=rep(FALSE,Tmax)
  vals[tabChoixMachine] = runif(nbEpreuveSurMachineA) <= pA
  vals[!tabChoixMachine] = runif(nbEpreuveSurMachineB) <= pB
  vals
}

#Strategie de la question 2
epsilonT = function(Tmax,pA,pB,epsilon){
  nbTest=round(epsilon*Tmax/2) #Correspond au nombre de test que l'on va effectuer
  # Si le nombre de test n'est pas un multiple de 2 on onleve 1.
  if(nbTest %% 2 != 0){
    nbTest = nbTest-1
  }
  t=rep(c(TRUE,FALSE),nbTest/2) # Creation d'un vecteur de taille nbTest alternant True et False
  vals=rep(FALSE,nbTest)

  vals[t]=runif(nbTest/2) <= pA
  vals[!t]=runif(nbTest/2) <= pB

  nbRestant = Tmax-nbTest

  if(mean(vals[t]) >= mean(vals[!t])){
    # on choisit la machine A
    suite = sample(c(TRUE,FALSE),nbRestant,replace=TRUE,prob=c(pA,1-pA))
  } else{
    #On choisit la machine B
    suite = sample(c(TRUE,FALSE),nbRestant,replace=TRUE,prob=c(pB,1-pB))
  }
  valsFinal=c(vals,suite)
}

```

```

  valsFinal
}

```

```

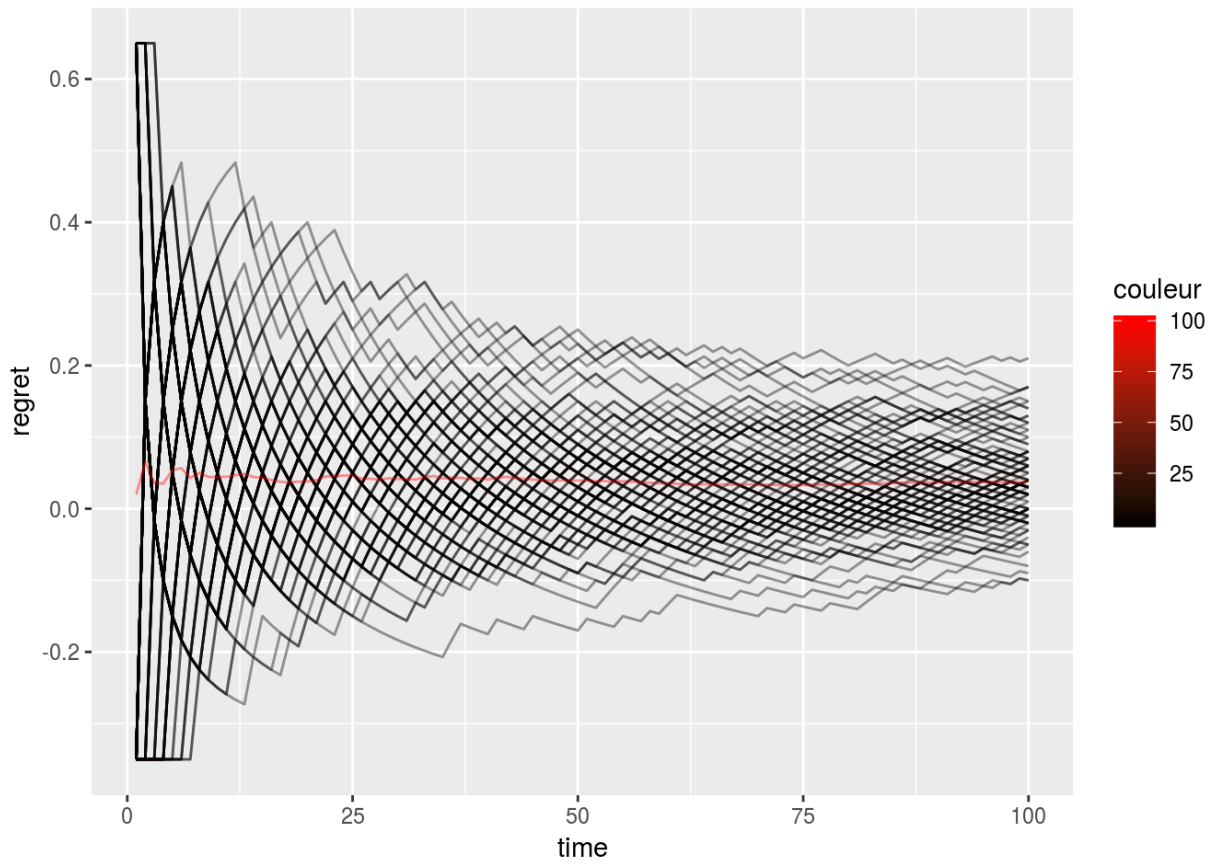
#Ici des fonctions qui vont servir a plot nos simultions
regret = function(vals,p){
  i=1
  tableauRegret = rep(0,length(vals))
  while(i<= length(vals)){
    tableauRegret[i]=p-(sum(vals[1:i])/i)
    i=i+1
  }
  tableauRegret
}
#Simule un fois Tmax experience
simulation = function(Tmax,pA,pB,strategie,epsilon=0.1){
  vals = switch(strategie,
    "epsilonT" = epsilonT(Tmax,pA,pB,epsilon),
    "onlyA" = onlyA(Tmax,pA,pB),
    "onlyB" = onlyB(Tmax,pA,pB),
    "half" = half(Tmax,pA,pB),
    "unMoinsEpsilon" = unMoinsEpsilon(Tmax,pA,pB,epsilon),
    "beta" = beta(Tmax,pA,pB)
  )
  vals
}
#Simule N fois Tmax experience et affiche le resultat avec ggplot
simulationNfois= function(N=100,Tmax,pA,pB,strategie,epsilon=0.05){
  i=0
  df = data.frame()
  regret_mean = rep(0,Tmax)
  while(i<N){
    vals=simulation(Tmax,pA,pB,strategie,epsilon)
    tableauRegret = regret(vals,pA)
    regret_mean= regret_mean + tableauRegret
    df=rbind(df,data.frame(id=i,time = seq(1,Tmax), regret = tableauRegret,couleur=1))
    i = i+1
  }
  regret_mean = regret_mean/N
  df=rbind(df,data.frame(id=N,time = seq(1,Tmax),regret= regret_mean,couleur=100)
)
  df
}
affichage=function(dataframe){
  lp <- ggplot(dataframe, aes(x = time, y =regret,group = id,color = couleur)) +
  geom_line(alpha = 0.4)
  lp + scale_color_gradient2(low="black",mid="black",high="red")
}

```

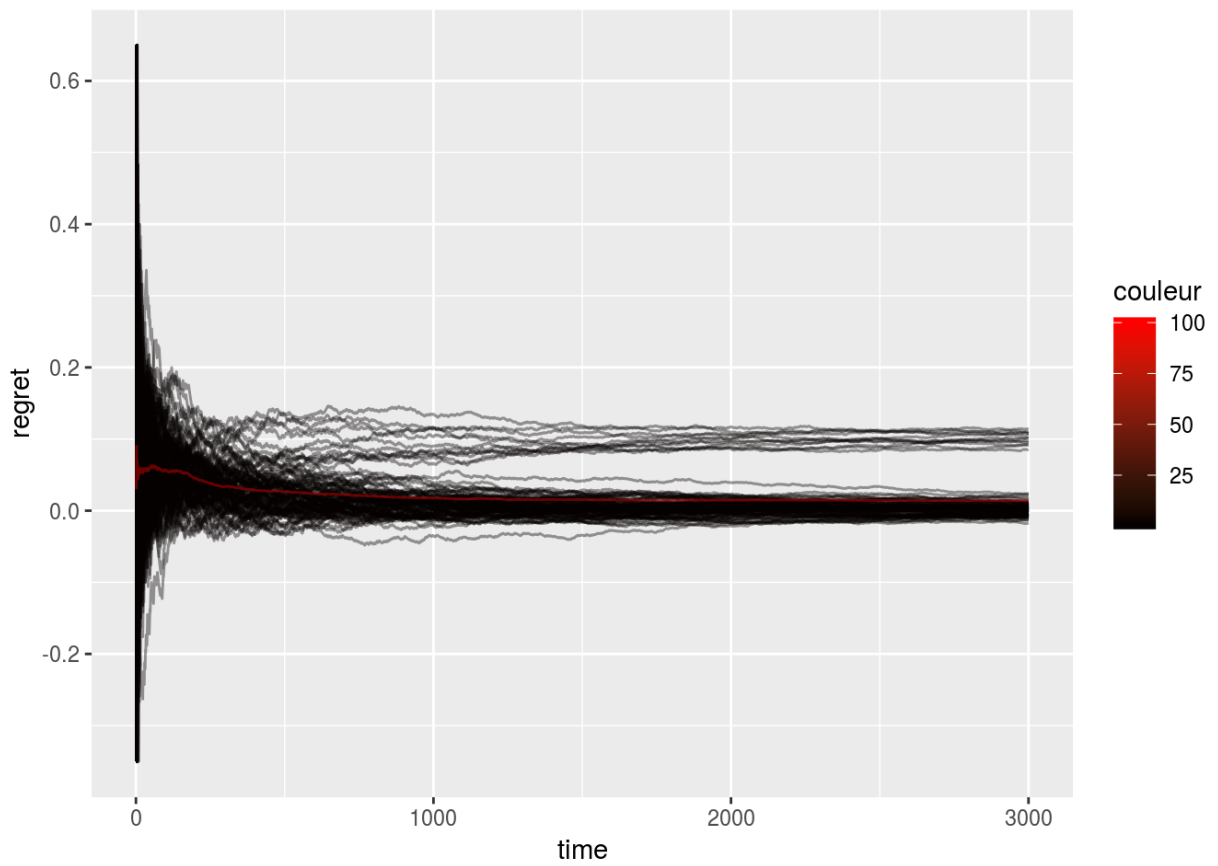
```

#q2
affichage(simulationNfois(N=100,Tmax=100,pA,pB,"epsilonT",0.2))

```



```
affichage(simulationNfois(N=100,Tmax=3000,pA,pB,"epsilonT",0.1))
```



```
#affichage(simulationNfois(N=100,Tmax=300,pA,pB,"onlyA"))  
#affichage(simulationNfois(N=100,Tmax=300,pA,pB,"onlyB"))  
#affichage(simulationNfois(N=100,Tmax=300,pA,pB,"half"))
```

La stratégie epsilon sur de grands échantillons est vraiment efficace puisque la phase de test est assez grande pour être toujours sûr de choisir la bonne machine. Ainsi avec epsilon à 0.05 on a $95 + 2.5 = 97.5\%$ des parties qui se jouent sur la machine **A**.

Donc le regret tend vers 0 et on maximise nos gains. Cependant sur de grands échantillons, 5% de parties tests est parfois trop élevé. On pourrait, en choisissant plus astucieusement la valeur de epsilon, augmenter le nombre d'épreuves faites sur la machine **A**.

Les rares cas où la phase de test semble pointer la mauvaise machine, sera le scénario le plus catastrophique possible. Il n'y aura pas de rattrapage possible et on tendra vers le pire scénario possible. Avec cette stratégie sur de grands échantillons, l'écart à la moyenne est faible. Car une majorité des scénarios tendent vers un regret nul. Mais on a une petite part qui tendra vers le regret maximal possible. C'est qu'il est double.

Par contre si l'on travaille avec de plus petits échantillons et que par hasard on choisit la machine **B** car elle semblait plus favorable, les résultats sont catastrophiques et tendent vers le minimum des gains moyens. (Soit 0.55 euros par coup joué). Pour cela pour assurer le coup il vaut mieux augmenter epsilon qu'à jouer un plus grand nombre de coups sur la machine la moins rentable. Dans ce genre de cas on a un graph très dispersé. L'écart à la moyenne est maximal et on est sûr de rien.

En conclusion cette stratégie est extrêmement efficace pour de grands échantillons et donnera des résultats en général extrêmement bons. Pour de petits échantillons cette stratégie est assez inefficace et l'écart à la moyenne du regret est maximal.

Question 3 :

Intuition : On a une stratégie qui va pouvoir continuellement apprendre. Réduisant ainsi drastiquement le nombre de scénarios où la machine B est favorisée. Cela coûtera cependant sûrement un plus grand nombre d'expériences sur la machine B pour les grands échantillons. Le résultat risque d'être assez chaotique sur les petits échantillons. Là où les premiers résultats risquent d'avoir un fort impact.

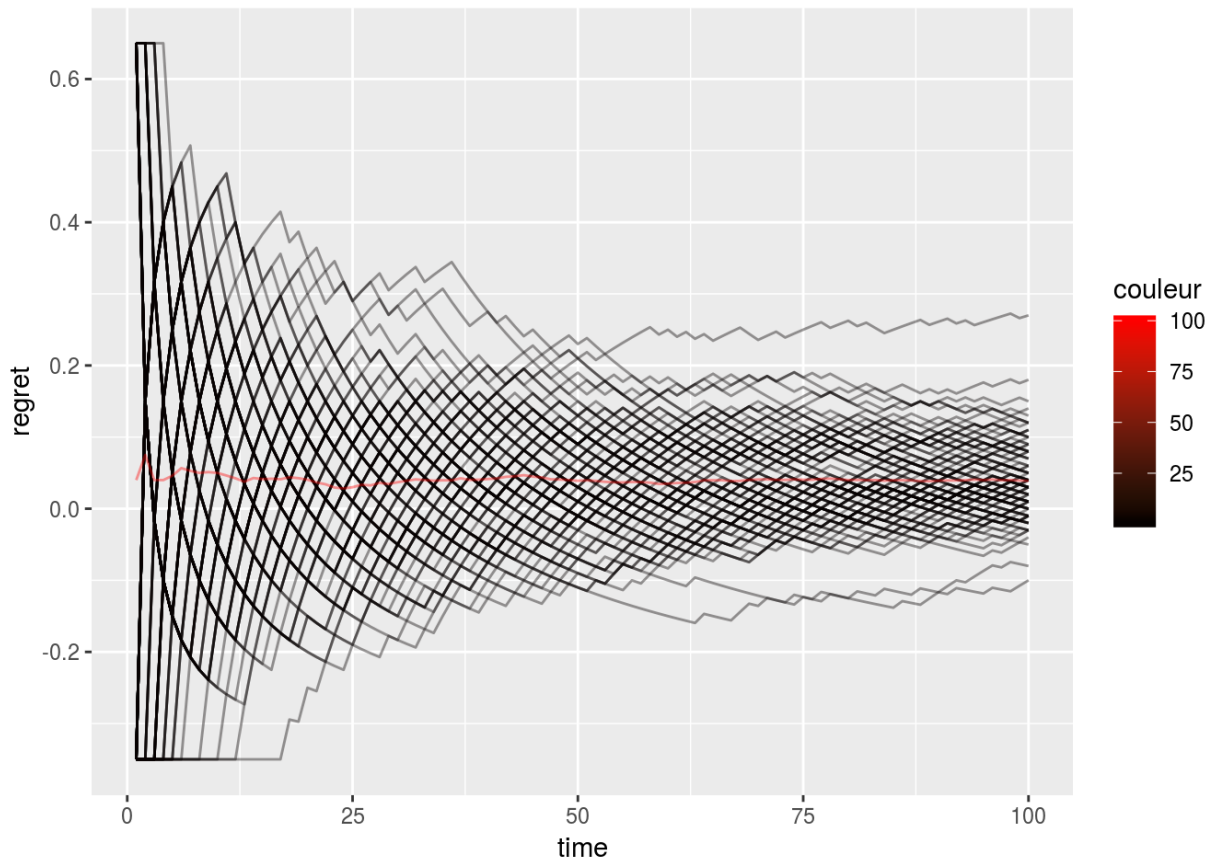
Programme :

```
#Stratégie question 3
unMoinsEpsilon = function(Tmax,pA,pB,epsilon){
  vals = rep(FALSE,Tmax)
  i=1
  nb_succes_A=0
  nb_essais_A=0
  nb_succes_B=0
  nb_essais_B=0
  p=0
  while(i<=Tmax){
    if( nb_succes_A/(nb_essais_A+1) == nb_succes_B/(nb_essais_B+1)){
      if(runif(1)<0.5){
        p=pA
        nb_essais_A = nb_essais_A +1
      }else{
        p=pB
        nb_essais_B = nb_essais_B +1
      }
    } else {
      choix_machine=runif(1)<(1-epsilon)
      if(nb_succes_A/(nb_essais_A+1) > nb_succes_B/(nb_essais_B+1)
        && choix_machine
        || nb_succes_A/(nb_essais_A+1) < nb_succes_B/(nb_essais_B+1)
        && !choix_machine ){
        p=pA
        nb_essais_A = nb_essais_A +1
      }else{
        p=pB
        nb_essais_B = nb_essais_B +1
      }
    }
  }

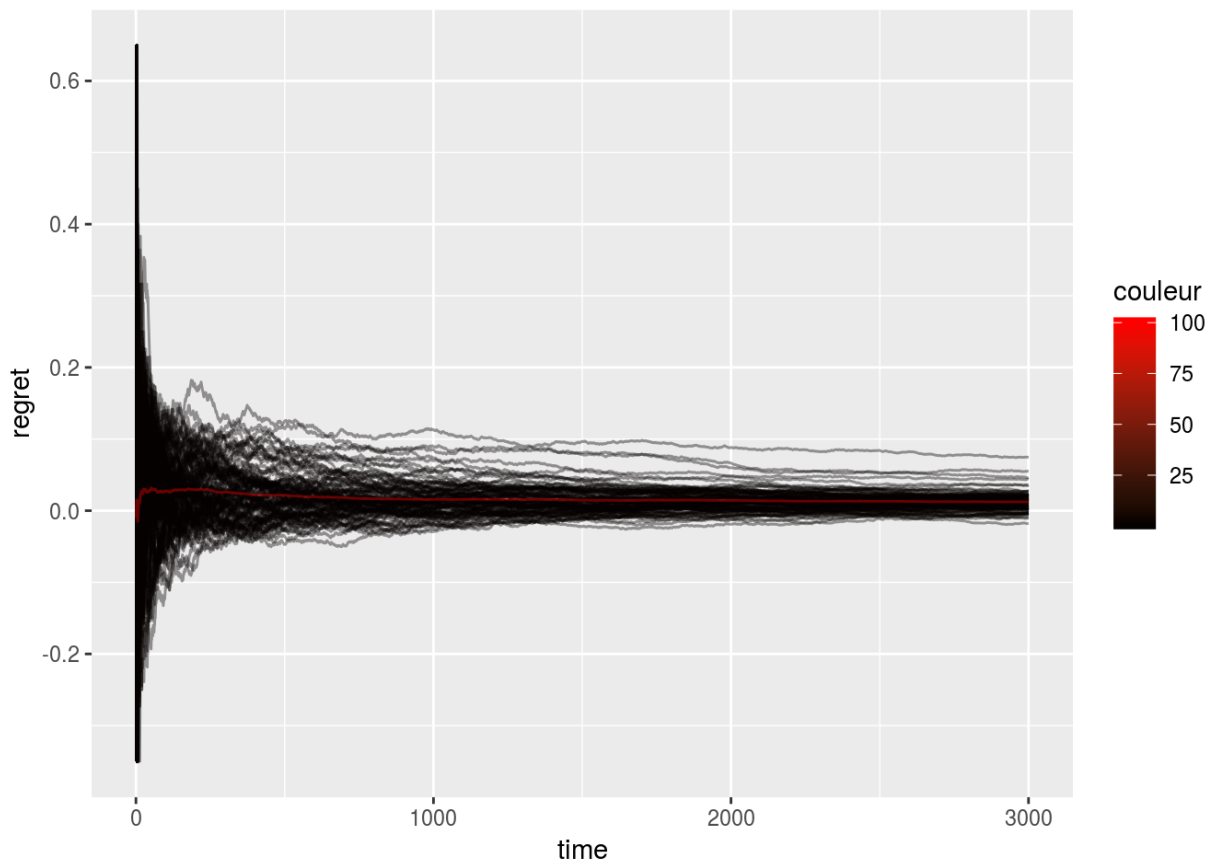
  vals[i] = runif(1)<p
  if(vals[i]){
    if(p == pA){
      nb_succes_A=nb_succes_A+1
    }else {
      nb_succes_B=nb_succes_B+1
    }
  }
  i=i+1
}

vals
}

#q3
affichage(simulationNfois(N=100,Tmax=100,pA,pB,"unMoinsEpsilon",0.1))
```



```
affichage(simulationNfois(N=100,Tmax=3000,pA,pB,"unMoinsEpsilon",0.1))
```



Cette stratégie sur de grands échantillons va tendre vers la stratégie précédente de paramètre 2ϵ . Donc sur de grands échantillons on va passer le double de temps sur la "mauvaise machine" car on continuera dans $\epsilon\%$ des cas à tester sur la machine la moins probable. Cela reste néanmoins assez négligeable étant donné que l'on passe beaucoup de temps sur la bonne machine. La stratégie est quand même efficace et les résultats sont bons et plutôt très proche de la stratégie précédente. Bien que légèrement moins bien, quand tout se passe bien. On a aussi une quasi certitude que l'on finira par tendre vers un regret très faible, mais non nul.

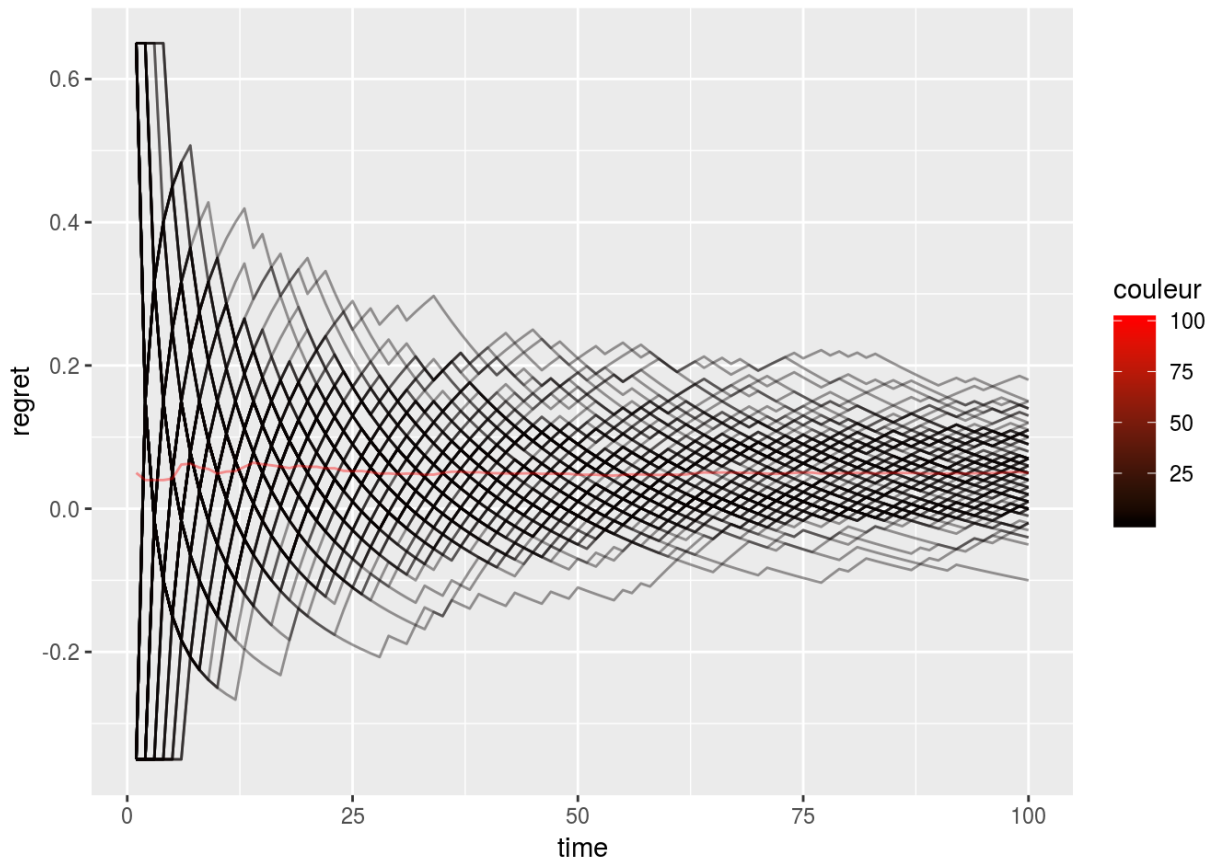
Là où cette stratégie est aussi intéressante c'est sur les échantillons plus réduits. Étant donné que l'on n'arrête jamais "d'apprendre", même si par malchance la mauvaise machine semble être désignée, on va continuer de douter de cette machine, et éventuellement, à un moment pouvoir rectifier le tir. Si le début de l'expérience pointe la machine B comme étant la plus rentable par "malchance", on ne finira pas systématiquement comme avec la stratégie précédente à jouer tout nos coups sur la machine B. On passe plus de temps à apprendre en continu, mais ça permet d'avoir une approche critique et de remise en question. Les résultats sur de petits échantillons ne sont quand même pas extrêmement probants, mais ils sont en moyenne mieux avec cette stratégie que la précédente. À l'allure des courbes, on voit que même si la moyenne du regret est légèrement plus haute il y a une plus grosse majorité de scénarios qui finissent avec un regret proche de 0 et beaucoup moins de scénarios qui dévient totalement vers le regret maximum. Là où avec la stratégie précédente, il arrivait que des expériences, par malchance sur les premiers tests, finissent pas jouer uniquement sur la mauvaise machine.

Question 4

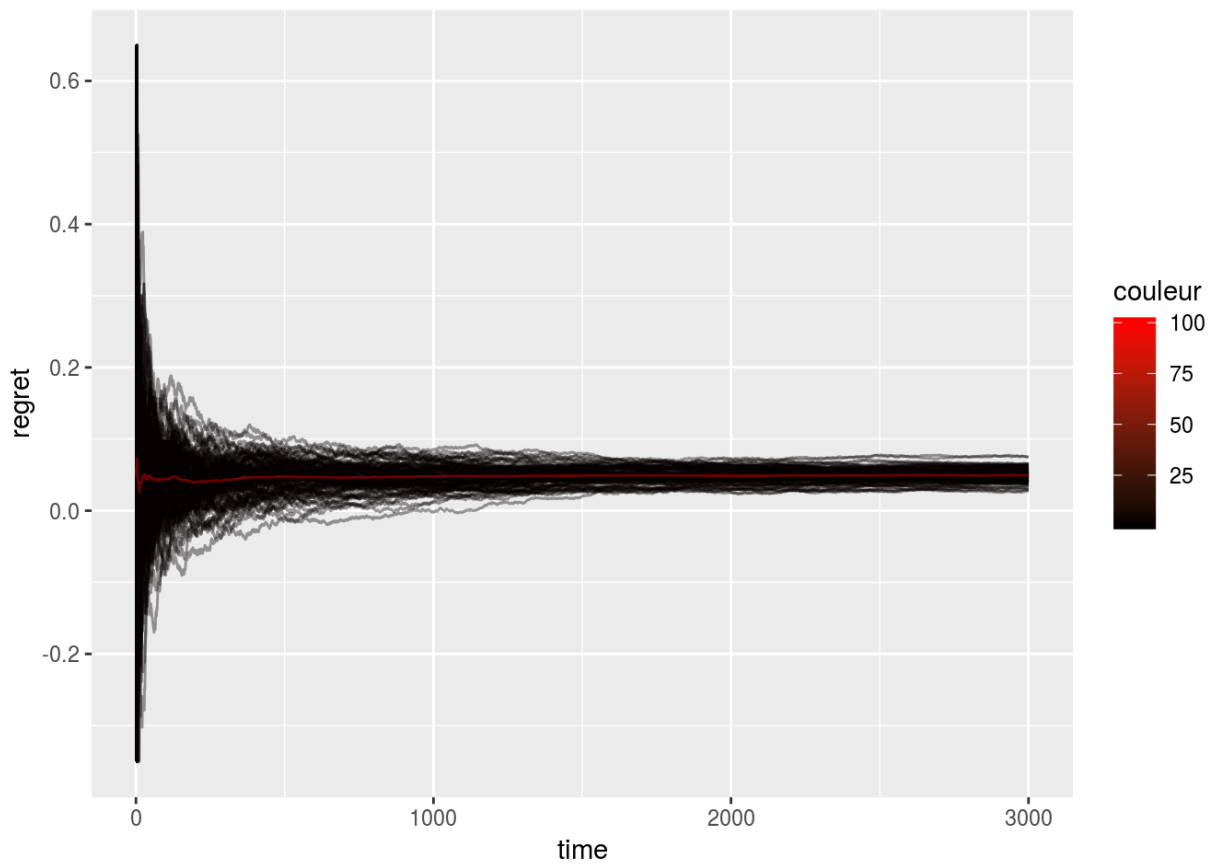
Intuition : Stratégie assez floue au premier abord. J'imagine que l'on va prendre la bonne machine quasiment systématiquement à partir d'un moment. Cette stratégie serait un mixe de la question 2 et 3. On continue toujours d'apprendre, mais quand on est suffisamment convaincu, on va arrêter de remettre en cause la bonne machine. Cela permet un apprentissage au fur et à mesure comme avec la question 3, mais qui saura arrêter, en général, de "gaspiller" des coups sur la mauvaise machine à partir d'un moment.

```
# Strategie question 4
beta = function(Tmax,pA,pB){
  n1_A=0
  n2_A=0
  n1_B=0
  n2_B=0
  i=1
  vals=rep(FALSE,Tmax)
  #peA la probabilité estimée de A
  #peB la probabilité estimée de B
  while(i<=Tmax){
    peA=rbeta(1,n1_A+1,n2_A+1)
    peB=rbeta(1,n1_B+1,n2_B+1)
    if(peA > peB){
      b = runif(1)<pA
      vals[i] = b
      n1_A = n1_A + b
      n2_A = n2_A + b
    }else{
      b = runif(1)<pB
      vals[i] = b
      n1_B = n1_B + b
      n2_B = n2_B + b
    }
    i=i+1
  }
  vals
}

#q4
affichage(simulationNfois(N=100,Tmax=100,pA,pB,strategie ="beta"))
```



```
affichage(simulationNfois(N=100,Tmax=3000,pA,pB,strategie ="beta"))
```



Cette stratégie est plus dure à appréhender conceptuellement. Mais au vu des résultats on comprend plusieurs choses.

En moyenne cette méthode est celle qui génère le plus de regret. Mais même si en moyenne on a des résultats moins bons qu'avec les autres stratégies, celle-ci a le mérite d'être plutôt très fiable. L'écart à la moyenne est très faible. Et même sur de petits échantillons, cet écart est meilleur que dans toutes les stratégies précédentes.

C'est la stratégie de la sécurité, on n'aura en moyenne pas les meilleurs gains, mais on réduit significativement nos chances d'avoir un regret assez fort.

En bilan on a une stratégie de quitte ou double, avec de très fortes chances de réussite avec la première stratégie si les échantillons sont grands.

Pour un peu plus de sécurité avec un grand nombre de d'échantillon la deuxième stratégie est aussi excellente. Cependant quand on en vient à de petits échantillons ou que l'on veut juste jouer au maximum la sécurité, la troisième stratégie est toute adaptée.