TP 11 (Contrôle continu n°2) : Simulation, estimation, model fitting, intervalles de confiance

Remarques générales : il y a d'assez bonnes choses globalement dans vos travaux, mais pour un trop grand nombre d'entre vous, je pense que les consignes ne sont pas assez bien lues et prises en compte. Certains se pressent pour écrire du code ou des mots qui n'ont parfois aucun lien avec ce qui est demandé. Ce que j'attendais de vous, ce n'était pas de "ressortir votre cours par coeur", c'était de réflechir par vous même et de dire les choses avec VOS mots. Durant les TPs, j'ai eu beaucoup trop peu de questions de la part de ceux qui avaient bien besoin d'éclaircissements (le devoirs le prouvent). A vrai dire, quasiment pas de questions. C'est dommage...

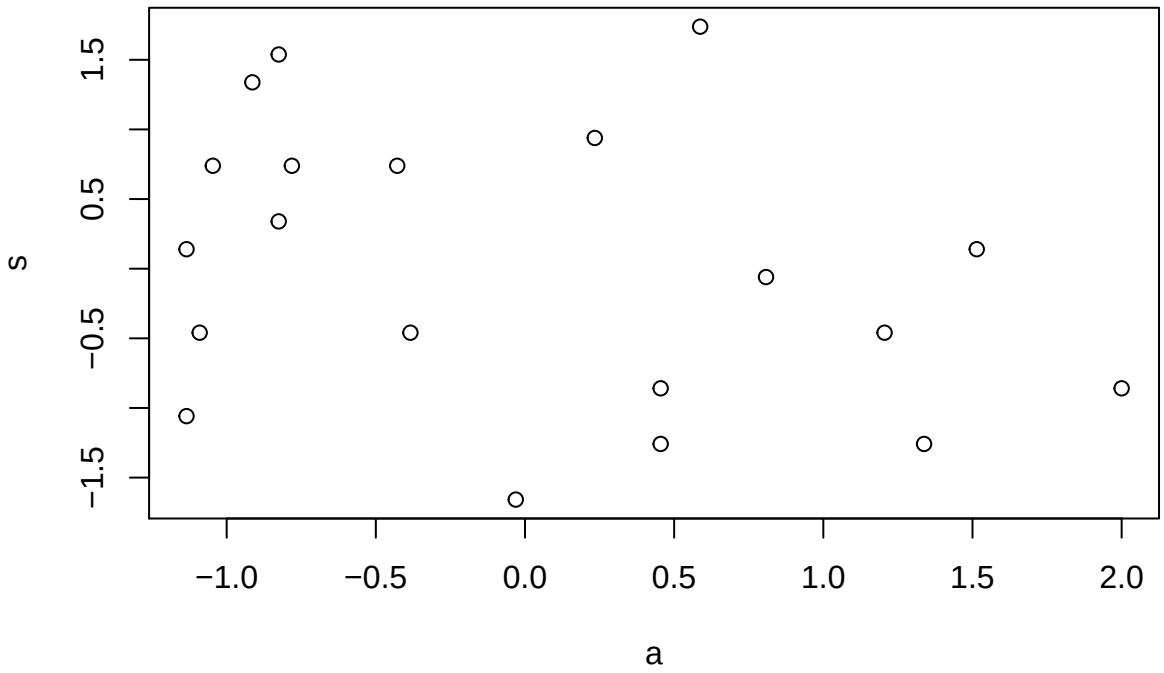
Exercice 1: régression logistique, détection de profils et prédiction (8 points)

1. (1 point) Une interprétation pertinente des paramètres de ce modèle était la suivante : les paramètres b_1 et b_2 quantifient l'influence du facteur âge (pour b_1) et exposition au soleil (pour b_2) sur l'apparition de la maladie X : un b_2 élevé dans le positif signifierait par exemple que l'exposition au soleil influe très positivement en faveur de la maladie. A contrario, si b_1 est très négatif par exemple, on peut conclure que l'âge fait diminuer la probabilité d'apparition de la maladie. Un coefficient b_1 proche de 0 signifierait que le facteur âge n'a pas d'impact sur l'apparition maladie.

Pour le facteur b_0, il y a plusieurs interprétations possibles : la première serait de dire que b_0 prend en compte "tous les autres facteurs d'apparition de la maladie", qu'on ne prend pas en compte dans notre étude. La deuxième est de dire que b_0 sert à renormaliser la probabilité d'apparition : en effet si $a = s = 0$, il n'y a pas a priori de raison pour que la probabilité d'apparition soit de $1/2$. D'où un paramètre b_0.

```
data = read.csv('data.csv')
a = (data$age-mean(data$age))/sd(data$age)
s = (data$exposition-mean(data$exposition))/sd(data$exposition)
Y = data$malade

plot(a,s)
```

2. (2 points)




```
cor(a,s)
```

```
## [1] -0.3254505
```

Nous voyons sur le graphique que s ne semble pas suivre de tendance très claire en fonction de a. On peut supposer que la covariance est faible. Le calcul nous le confirme, avec tout de même une corrélation légèrement négative : cela est assez logique, puisque les personnes âgées ont tendance à moins s'exposer au soleil.

Remarque : il est étonnant de voir qu'un certain nombre d'entre vous ne savent pas renormaliser un vecteur correctement. Attention à ce point essentiel.

3. (1 point) Avec une boucle for (mais c'est plus long à tourner) :

```
L = function(b0,b1,b2,Y,a,s){
  logL = 0
  n = length(Y)
  for (i in 1:n){
    logL = logL + Y[i]*(b0+b1*a[i]+b2*s[i]) - log(1+exp(b0+b1*a[i]+b2*s[i]))
  }
  exp(logL)
}
# L(0,0,0,Y,a,s) ## (pour un test)
```

Ou la façon la plus optimale (vectorielle) :

```
L = function(b0,b1,b2,Y,a,s){
  s = sum(Y*(b0+b1*a+b2*s) - log(1+exp(b0+b1*a+b2*s)))
  exp(s)
}
# L(0,0,0,Y,a,s) ## (pour un test)
```

Remarque : là encore cette question n'est pas bien traitée par un certain nombre d'entre vous, entraînant des propagation d'erreurs. Certains ne sont pas au clair avec les boucles for ! Il fallait bien tester votre fonction...

```
k = 120
b0s = seq(-2,2,length.out=k)
b1s = seq(-2,2,length.out=k)
b2s = seq(-2,2,length.out=k)

MV=0
for (i in 1:k){
  b0 = b0s[i]
  for (j in 1:k){
    b1 = b1s[j]
    for (k in 1:k){
      b2 = b2s[k]
      l = L(b0,b1,b2,Y,a,s)
      if (l>MV){ ## si la vraisemblance est plus grande que le max provisoire
        res = c(b0,b1,b2) ## on met à jour res
        MV = l ## on met à jour le max MV
      }
    }
  }
}
MV ## attention on revoie à la fois le max
```

4. (2 points)

```
## [1] 0.0003991071
```

```
res ## et l'argmax
```

```
## [1] -0.6218487  1.9327731 -0.4873950
```

Remarque : ce code n'était pas difficile à adapter des TPs précédents. Attention, si vous aviez défini L de façon non vectorielel (i.e. avec une boucle for), le code prenait beaucoup plus de temps à tourner... D'où l'intérêt de la programmation vectorielle !

5. (1 point) Le paramètre dominant est ici b_1 : c'est donc l'âge qui influe le plus (et positivement) sur l'apparition de la maladie X.

```
a0 = 0.
s0 = 0.6
b0=res[1]
b1 = res[2]
b2 = res[3]
proba = exp(b0+a0*b1+s0*b2)/(1+exp(b0+a0*b1+s0*b2))
proba
```

6. (1 point)

```
## [1] 0.2861236
```

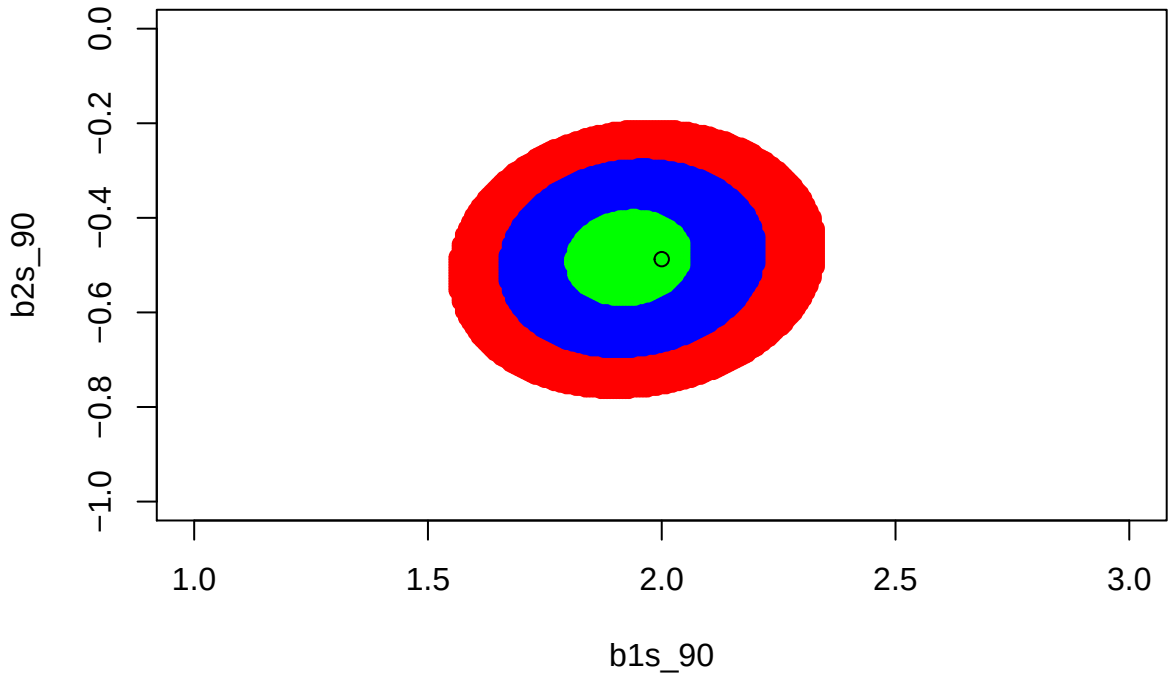
Dans ce cas la probabilité de développer la maladie X est d'environ 29.

Remarque : attention, vous étiez un certain nombre à réutiliser ici la vraisemblance !! Cela montre que vous n'avez pas vraiment compris le modèle. La vraisemblance n'est pas utilisable pour autre chose que pour de l'inférence ! Maintenant que les paramètres du modèle sont estimés, il faut réinjecter tout ça dans le modèle qu'on étudie, qui donne la probabilité que $Y = 1$ en fonction de a et s.

Partie bonus (jusqu'à + 2,5 points)

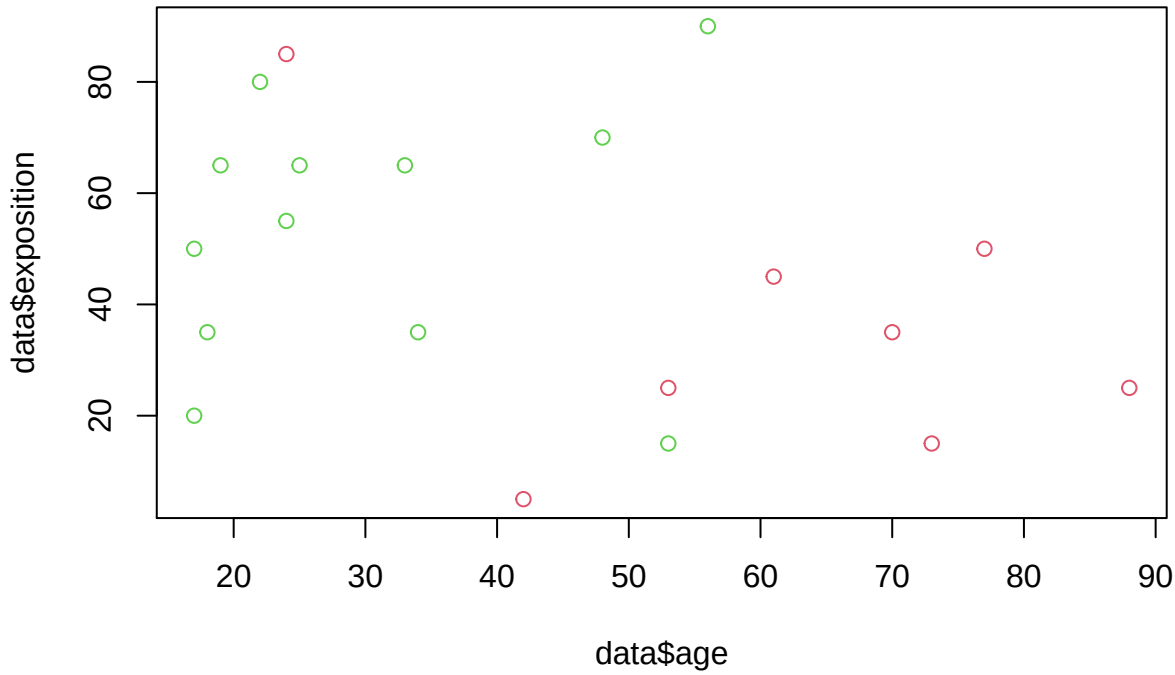
```
MV = 0.0003991071 ## on reporte la précédente valeur du maximum de vraisemblance
b0 = -0.62
n_sample = 300
b1s = seq(1,3,length.out=n_sample)
b2s = seq(-1,0,length.out=n_sample)

b1s_90 = c()
b2s_90 = c()
b1s_95 = c()
b2s_95 = c()
b1s_99 = c()
b2s_99 = c()
for (i in 1:n_sample){
  b1 = b1s[i]
  for (j in 1:n_sample){
    b2 = b2s[j]
    if (L(b0,b1,b2,Y,a,s)>0.9*MV){
        b1s_90 = c(b1s_90,b1) ## vecteur des abscisses des points dans la zone des 90%
        b2s_90 = c(b2s_90,b2) ## ordonnées de ces mêmes points
    }
    if (L(b0,b1,b2,Y,a,s)>0.95*MV){
        b1s_95 = c(b1s_95,b1) ## zone des 95%
        b2s_95 = c(b2s_95,b2)
    }
    if (L(b0,b1,b2,Y,a,s)>0.99*MV){
        b1s_99 = c(b1s_99,b1) ## zone des 99%
        b2s_99 = c(b2s_99,b2)
    }
  }
}

plot(b1s_90,b2s_90,ylim=c(-1,0),xlim=c(1,3),col='red') ## on dessine la zone des 90
points(b1s_95,b2s_95,col='blue') ## on superpose la zone des 95
points(b1s_99,b2s_99,col='green') ## on superpose la zone des 99
points(res[2:3],col='black') ## on superpose le point correspondant au MV
```

7. (2 points)




```
plot(data$age, data$exposition, col=3-data$malade) ## on peut paramétrer la
```

8. (0.5 point)




```
## couleur avec un nombre >=1, différent selon si l'individu est malade ou sain
```

On retrouvait bien le résultat établi plus haut : l'âge est le facteur déterminant dans l'apparition de la maladie X.

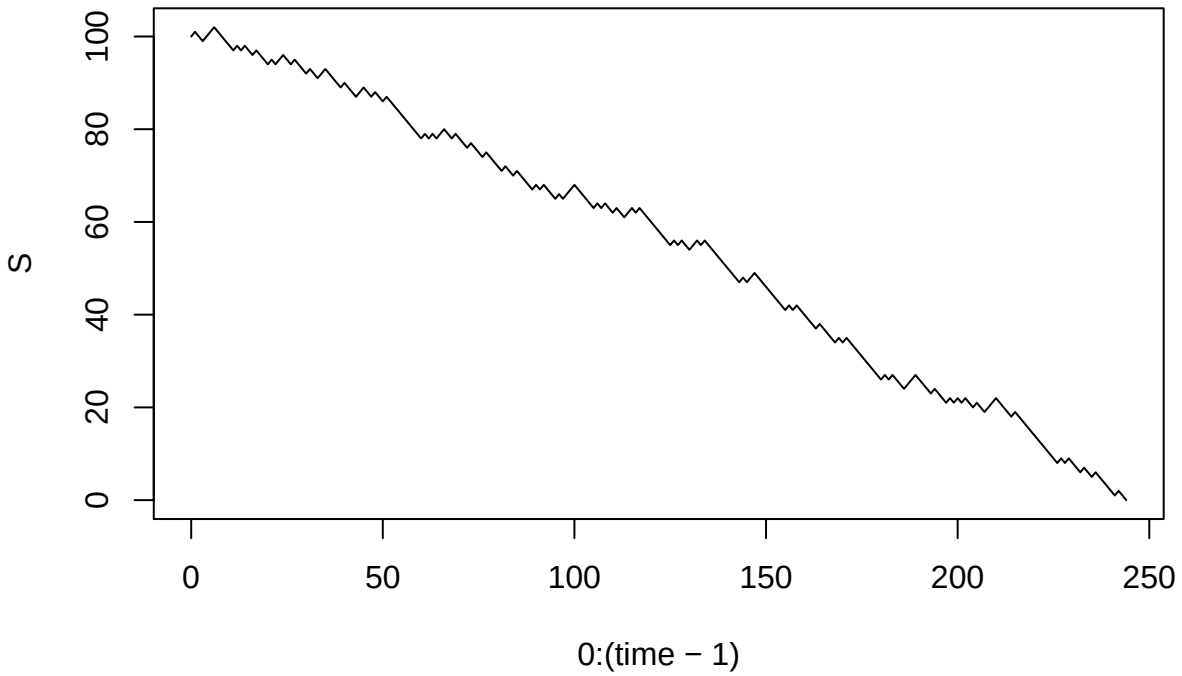
Exercice 2: la ruine du joueur (7,5 points)

1. (1,5 points) T correspond au premier temps où le joueur atteint 0 (et donc est ruiné), ou $a+b$ (et donc a gonflé son portefeuille de b euros). b peut par exemple être appelé "objectif de gain" du joueur. Au temps T, S_T ne peut valoir que 0 ou $a+b$.

Remarque : il est surprenant de vois que ces interprétations sont très floues chez certains... Il ne s'agissait pas de me ressortir votre cours sur les martingales ou autre... Certains me parlent de temps d'arrêt (ça c'est OK tant que c'est contextualisé), d'autres me parlent même de filtration... ça n'a rien à voir ici ! Il s'agissait de réfléchir et d'écrire en conséquence avec vos propres mots.

```
p = 0.3
a = 100
b = 25

x = a ## valeur initiale du portefeuille
S = c(x) ## valeur initiale du vecteur trajectoire
while (x>0 & x<(a+b)){ ## on teste si la trajectoire a atteint 0 ou a+b
  test = runif(1) ## on simule une uniforme dans [0,1]
  if (test<p){
    x = x+1 ## on ajoute un euro
  }
  else{
    x = x-1 ## on enlève un euro
  }
  S = c(S,x) ## on remplit la trajectoire (par la droite, en mettant à jour le dernier point).
}

time=length(S)
plot(0:(time-1),S,type="l")
```

2. (2 points)

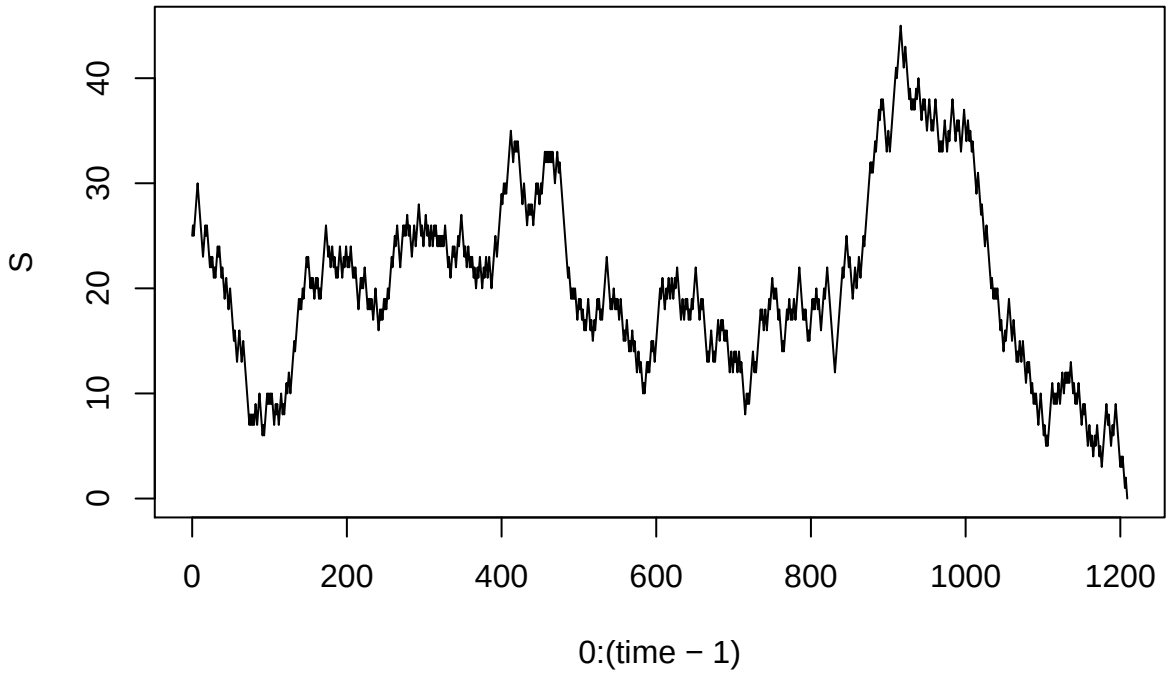


Avec ces paramètres, au vu des trajectoires, le joueur finit (quasiment) toujours ruiné !

Remarque : cette question n'est pas très bien traitée globalement. Certains ne renvoient pas même pas de vecteur, juste un nombre, certains ne renvoient rien. Beaucoup de confusions dans les boucles while. Certains mettent une condition avec un "ou" logique, ce qui fait tourner le code indéfiniment. En testant, je pense qu'on pouvait se rentre compte qu'il y avait un souci !

```
p = 0.5
a = 25
b = 25

x = a # valeur initiale du portefeuille
S = c(x)
while (x>0 & x<(a+b)){
  test = runif(1)
  if (test<p){
    x = x+1
  }
  else{
    x = x-1
  }
  S = c(S,x)
}

time=length(S)
plot(0:(time-1),S,type="l")
```

3. (1,5 points)



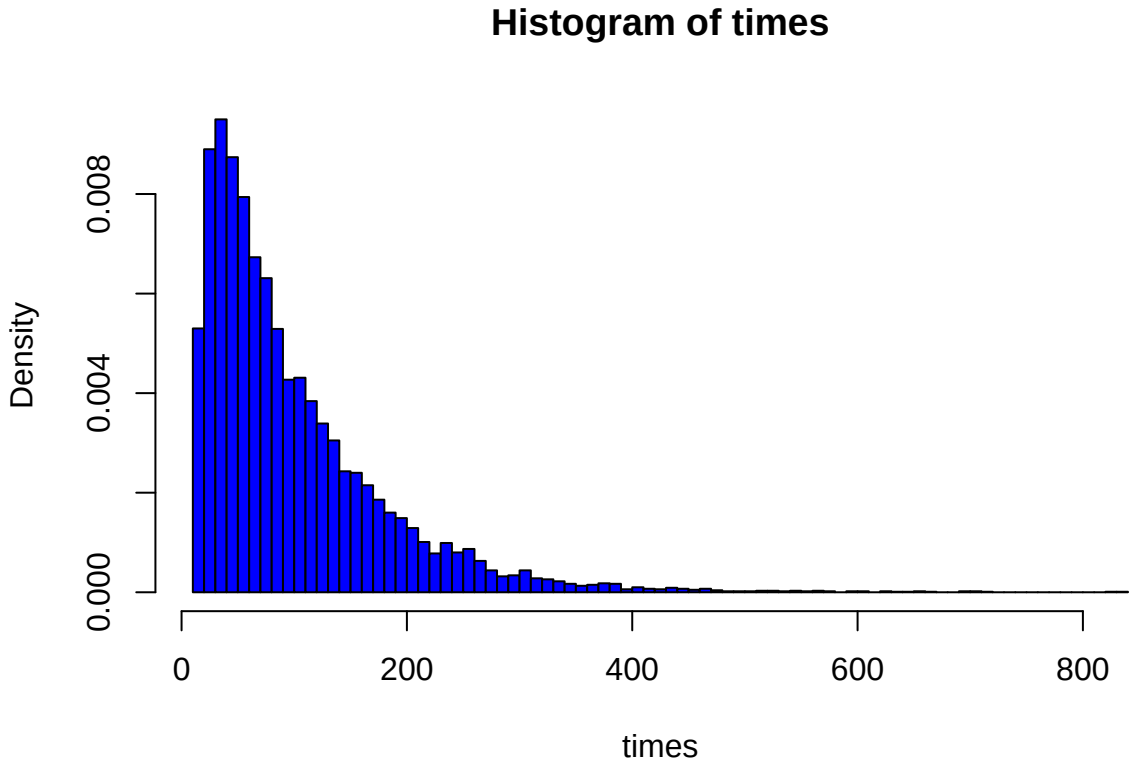
Dans ce cas, outre le fait que le jeu dure plus longtemps en moyenne, on observe que le joueur peut sortir du jeu en ayant atteint son objectif (doubler son portefeuille initial de 25 ici), ou en étant ruiné. Ces deux éventualités semblent arriver une fois sur deux. Cela est logique : comme $p = 0.5$ on peut utiliser un argument de symétrie : échanger les victoires et les défaites ne change pas la loi de la trajectoire, mais permute les issues du jeu (car $a = b$) ! Il est alors clair que sans plus de calcul, $\mathbb{P}(S_T = 0) = \mathbb{P}(S_T = a + b) = 1/2$ dans ce

cas.

```
time = function(p,a,b){
  t = 0 # on initialise le temps
  x = a # on initialise le portefeuille
  while (x>0 & x<(a+b)){
  t = t+1 # on rejoue une partie
  test = runif(1)
  if (test<p){
    x = x+1
  }
  else{
    x = x-1
  }
  }
  t
}
```

```
N = 10000
p = 0.5
a = 10
b = 10
times = replicate(N,0)
for (i in 1:N){
  times[i] = time(p,a,b)
}
hist(times,freq=FALSE,breaks = 100,col='blue')
```

4. (3 points)




```
mean(times)
```

```
## [1] 100.2246
```

T semble avec une queue de distribution exponentielle. Il semble "presque" suivre une loi exponentielle. A noter que la moyenne est autour de $100 = 10 \times 10 = a \times b$. On peut expliquer que le temps mis par une marche symétrique pour parcourir une distance d est d'ordre d^2.

Remarque : certains ne savent pas tracer un histogramme !! Attention aussi au calibrage de N et de breaks, on l'avait fait beaucoup ensemble...

Exercice 3: do it yourself Nous avons des paramètres de loi de Bernouilli à estimer. Le théorème central limite nous dit que si $X_1, \dots, X_n$ est un echantillon de variables de Bernoulli de paramètre p, alors

$$\sqrt{n}\frac{\bar{X}_n - p}{\sqrt{p(1-p)}} \overset{(d)}{\to} \mathcal{N}(0,1),$$

puis par le Lemme se Slutsky, comme $\sqrt{\bar{X}_n(1-\bar{X}_n)} \to \sqrt{p(1-p)}$ presque sûrement, on a

$$\sqrt{n}\frac{\bar{X}_n - p}{\sqrt{\bar{X}_n(1-\bar{X}_n)}} \overset{(d)}{\to} \mathcal{N}(0,1).$$

Un intervalle de confiance asymptotique de niveau $1-\alpha$ était donc donné par

$$\left[\bar{X}_n - q_{1-\alpha/2}\frac{\sqrt{\bar{X}_n(1-\bar{X}_n)}}{\sqrt{n}}, \bar{X}_n + q_{1-\alpha/2}\frac{\sqrt{\bar{X}_n(1-\bar{X}_n)}}{\sqrt{n}}\right],$$

où le quantile apparaissant est un quantile d'ordre $1-\alpha/2$ de la loi normale centrée réduite.

```r
alpha =0.0001

n0 = 1074309
S0 = 66671
n1 = 1041279
S1 = 56227

f0 = S0/n0
IC0 = c(f0 - qnorm(1-alpha/2)*sqrt(f0*(1-f0))/sqrt(n0),
        f0 + qnorm(1-alpha/2)*sqrt(f0*(1-f0))/sqrt(n0))

f1 = S1/n1
IC1 = c(f1 - qnorm(1-alpha/2)*sqrt(f1*(1-f1))/sqrt(n1),
        f1 + qnorm(1-alpha/2)*sqrt(f1+(1-f1))/sqrt(n1))

IC0
```

```
## [1] 0.06115381 0.06296504
```

```r
IC1
```

```
## [1] 0.05313629 0.05781071
```

Les intervalles sont disjoints, même avec un α très petit (d'ordre 0.0001). On en conclut que l'augmentation du taux d'incidence est (très) significatif.

Remarque : certains ont aussi pensé à Bienaymé-Tchebychev, ce qui marchait bien pour établir un autre intervalle de confiance.

Notation de la clarté du code (1 point) *J'estime qu'il est important de savoir prendre en compte la clarté du code dans un rapport de TP: quand on programme, on est pas le seul à se relire. D'autres personnes (vos collaborateurs) sont souvent amenés à regarder/modifier votre travail ! La clarté du code était donc évaluée de la façon suivante: -0.5 pt si votre code ne tournait pas (renvoyait un erreur), ou si on ne comprenait rien à son organisation, ou encore si vous affichiez des listes interminables, et -1 pt pour les boucles qui ne terminaient pas, les erreurs faisant bugger fortement le kernel, ou pour la présence de plusieurs erreurs d'origine différentes gênant l'éxécution.*