Travaux Dirigés ISV51 - Programmation et Développement - Correction

Julien Chiquet

20 et 21 novembre 2015

Objectifs de la séance

- manipulation des structures de contrôles
- premières fonctions
- premiers programmes `R`
- premères simulations et parallèlisation

Problème: population de bactéries

On souhaite modéliser la croissance d'une population bactérienne mise en culture dans une boîte de Petri. À cet effet, on distingue deux types de bactérie:

1. des bactéries jeunes et *immatures*, notées a, qui ne se divisent pas ;
2. des bactéries *matures*, notées b, susceptibles de se diviser par mitose.

On suppose que la reproduction a lieu à intervalles de temps discrets ; les bactéries b se divisent d'un instant à l'autre en une bactérie a et une bactérie b ; enfin, toute bactérie a devient mature d'un pas de temps à l'autre.

Première partie: définition du modèle et de la fonction de génération

1. On note $n_a(t)$ et $n_b(t)$ le nombre de bactéries de chaque type à l'instant t. Écrire le système de deux équations décrivant l'évolution de $n_a(t+1)$ et $n_b(t+1)$ en fonction de $n_a(t)$ et $n_b(t)$.
2. Écrire une fonction `PopBacteries(n0,T)` qui renvoie trois vecteurs de taille $T+1$ contenant l'évolution des deux catégories de bactérie de l'instant initial au temps T ainsi que l'évolution de la population totale. Le paramètre n_0 est le nombre $n_a(0)$, et l'on suppose que $n_b(0) = 0$.
3. Pour $T = 20$ et $n_0 = 1$, générer la population bactérienne correspondante et calculer le taux d'accroissement de la population totale. Représenter graphiquement ces résultats.
4. On souhaite maintenant introduire de l'aléa dans la dynamique bactérienne. À cet effet, on suppose qu'une bactérie de type b a une probabilité p d'accomplir une mitose en $a + b$. Modifier la fonction `PopBacteries(n0,T,p)` en ajoutant le paramètre p.

Commençons par la fonction générant la population (y compris avec le facteur aléatoire)

```
PopBacteries <- function(n0,T,p,last.only=FALSE) {

  Na <- integer(T+1)
  Nb <- integer(T+1)
  Na[1] <- n0

  for (t in 1:T) {
    Na[t+1] <- sum(runif(Nb[t]) <=p)
    Nb[t+1] <- Na[t] + Nb[t]
  }
  if (last.only) {
    return(Na[T+1]+Nb[T+1])
  } else {
    return(data.frame(Na=Na,Nb=Nb,N=Na+Nb))
  }
}
```

5. Étudier l'évolution de la population et son taux de croissance totale pour diverses valeurs de p.

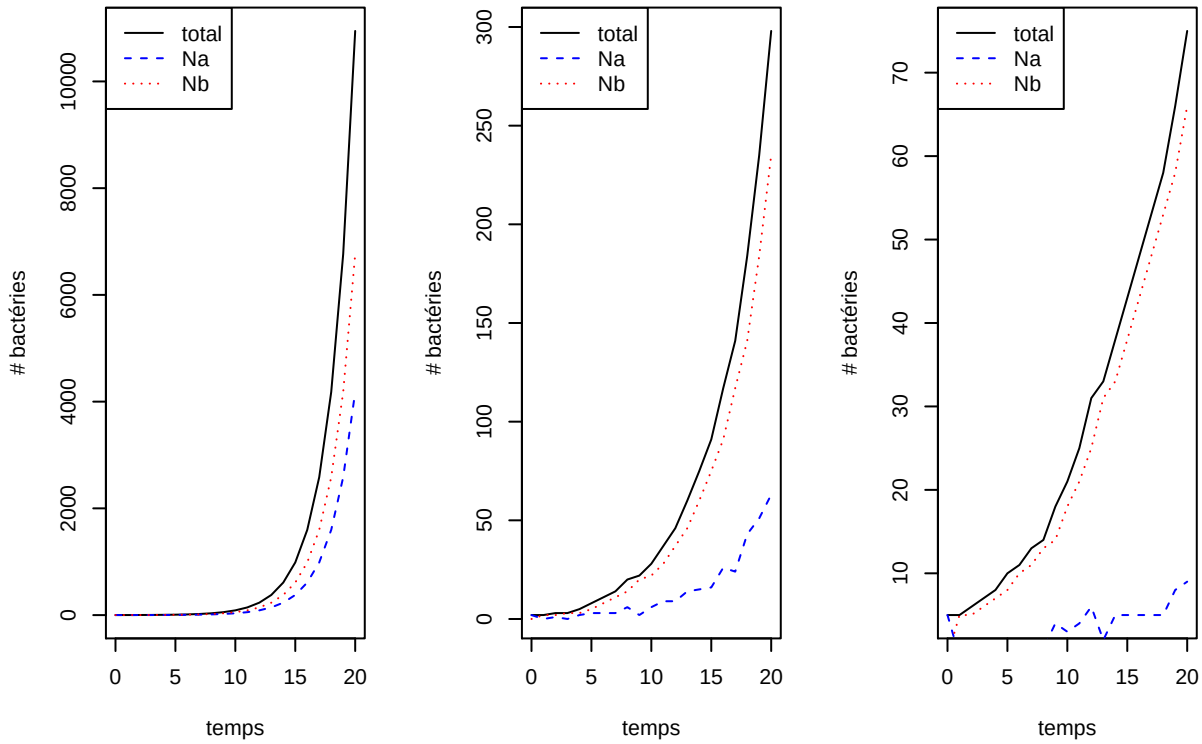
Voyons quelques exemples

```
T <- 20
Pop1 <- PopBacteries(1,T,1)
Pop2 <- PopBacteries(2,T,0.35)
Pop3 <- PopBacteries(5,T,0.15)

par(mfrow=c(1,3))
plot(0:T,Pop1$N, type="l", col="black",
     xlab="temps", ylab="# bactéries")
lines(0:T,Pop1$Na, col="blue", lty=2)
lines(0:T,Pop1$Nb, col="red", lty=3)
legend("topleft",c("total","Na","Nb"),
       col=c("black","blue","red"), lty=c(1,2,3))

plot(0:T,Pop2$N, type="l", col="black",
     xlab="temps", ylab="# bactéries")
lines(0:T,Pop2$Na, col="blue", lty=2)
lines(0:T,Pop2$Nb, col="red", lty=3)
legend("topleft",c("total","Na","Nb"),
       col=c("black","blue","red"), lty=c(1,2,3))

plot(0:T,Pop3$N, type="l", col="black",
     xlab="temps", ylab="# bactéries")
lines(0:T,Pop3$Na, col="blue", lty=2)
lines(0:T,Pop3$Nb, col="red", lty=3)
legend("topleft",c("total","Na","Nb"),
       col=c("black","blue","red"), lty=c(1,2,3))
```



Par taux d'accroissement d'une population N_t, on entend N_{t+1}/N_t. soit, en moyenne,

```
mean(Pop1$N[-1]/Pop1$N[-(T+1)])
```

```
## [1] 1.602111
```

```
mean(Pop2$N[-1]/Pop2$N[-(T+1)])
```

```
## [1] 1.294456
```

```
mean(Pop3$N[-1]/Pop3$N[-(T+1)])
```

```
## [1] 1.146891
```

S'il est strictement plus grand que 1, la population augmente; elle diminue s'il est strictement plus petit; enfin, elle est stable lorsque le taux d'accroissement vaut extactement 1.

Deuxième partie : expériences numériques

1. Rajouter une option `last.only` à la fonction `PopBacteries` permettant de retouner uniquement le nombre totale de bactéries en présence après un temps `T`.
2. On propose d'étudier la distribution empirique du nombre de bactérie après un temps $T = 10$ en fonction de diverses valeurs de p, et en partant d'une seule bactérie. Pour 50 valeurs de p échelonnées entre 0 et 1, simuler 100 populations de bactéries. Représenter la distribution d'intérêt sous forme de boxplot en fonction des valeurs de p. Vous utiliserez à cet effet la fonction `replicate`.

```r
T <- 10
n0 <- 1
nsim <- 100
p.seq <- seq(0, 1, len=100)

## la solution avec replicate
system.time(res <- sapply(p.seq, function(p) {
  return(replicate(nsim, PopBacteries(n0, T, p , last.only = TRUE)))
}))
```

```
##    user  system elapsed
##   0.521   0.009   0.530
```

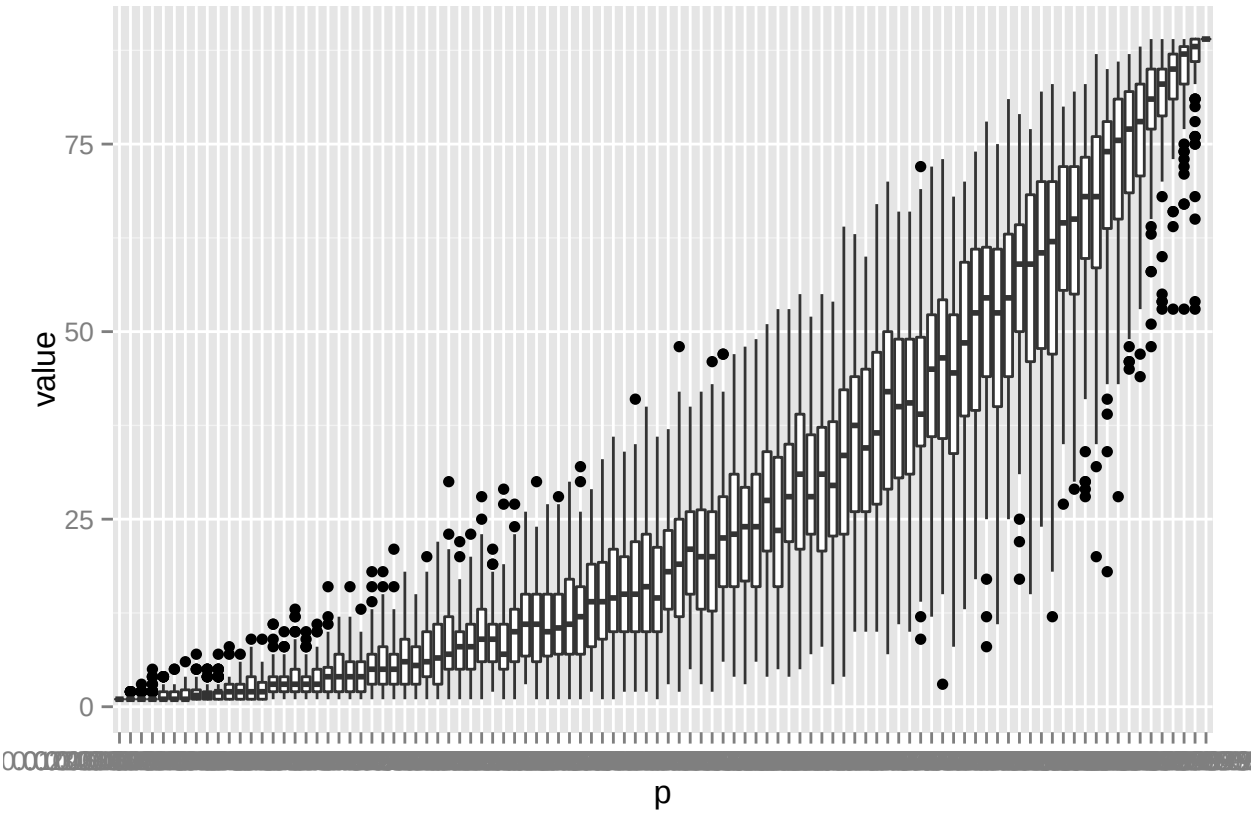
```r
res.replicate <- data.frame(value=c(res), p =factor(rep(p.seq, each=nsim)))

## la solution avec mclapply
library(parallel)
system.time(res <- mclapply(p.seq, function(p) {
  return(replicate(nsim, PopBacteries(n0, T, p , last.only = TRUE)))
}, mc.cores = 4))
```

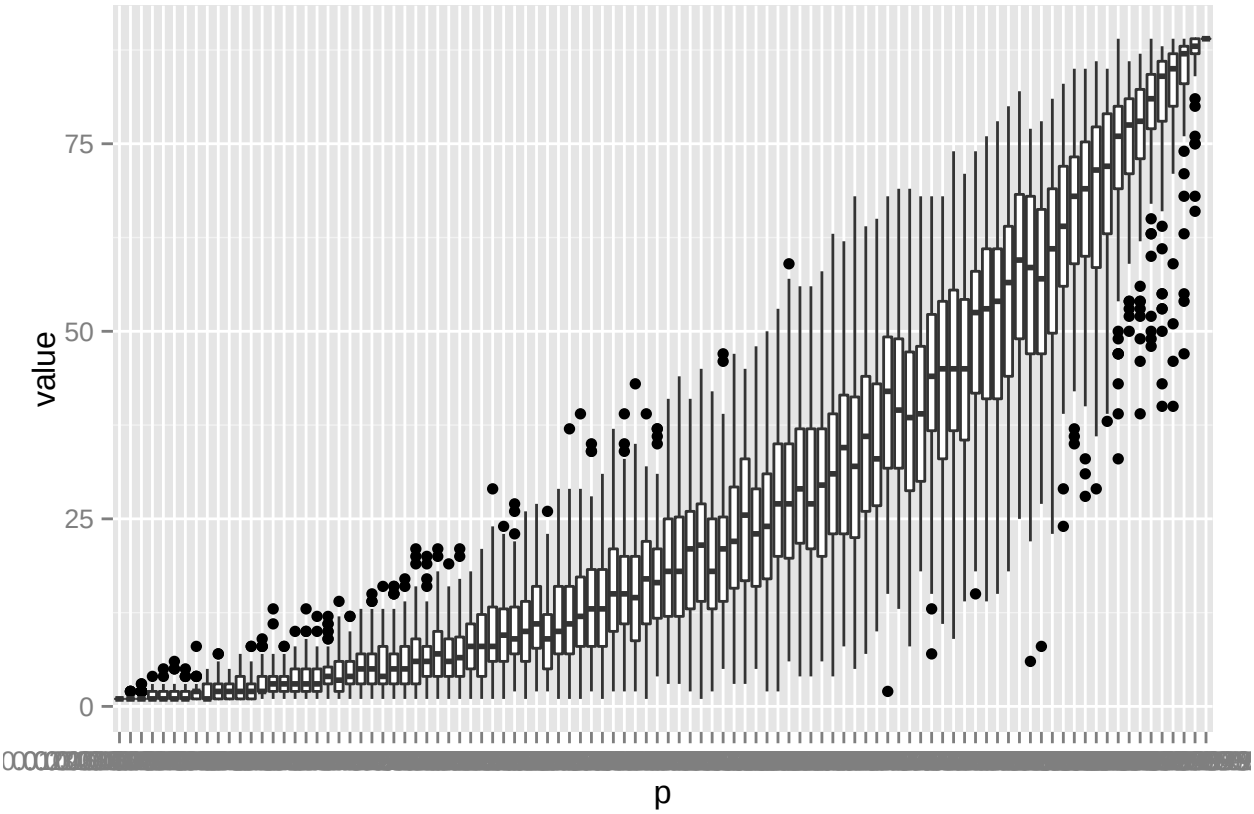
```
##    user  system elapsed
##   0.763   0.043   0.276
```

```r
res.mclapply <- data.frame(value=unlist(res), p =factor(rep(p.seq, each=nsim)))

library(ggplot2)
ggplot(res.replicate, aes(x=p, y=value)) + geom_boxplot()
```




```
ggplot(res.mclapply, aes(x=p, y=value)) + geom_boxplot()
```



Troisième partie : estimation de temps d'arrêt On souhaite maintenant déterminer combien de temps il est nécessaie à une certaine population pour atteindre une certaine taille.

1. Écrire une fonction `PopBacteries2(n0,nmax,p)` qui renvoie le temps nécessaire pour que la population atteigne la taille nmax.

```
PopBacteries2 <- function(n0,nmax,p) {

  t <- 0
  Na <- n0
  Nb <- 0

  while(Na+Nb< nmax)  {
    t <- t + 1
    Na.plus <- sum(runif(Nb) <=p)
    Nb.plus <- Na + Nb

    Na <- Na.plus; Nb <- Nb.plus
  }

  return(t)
}
```

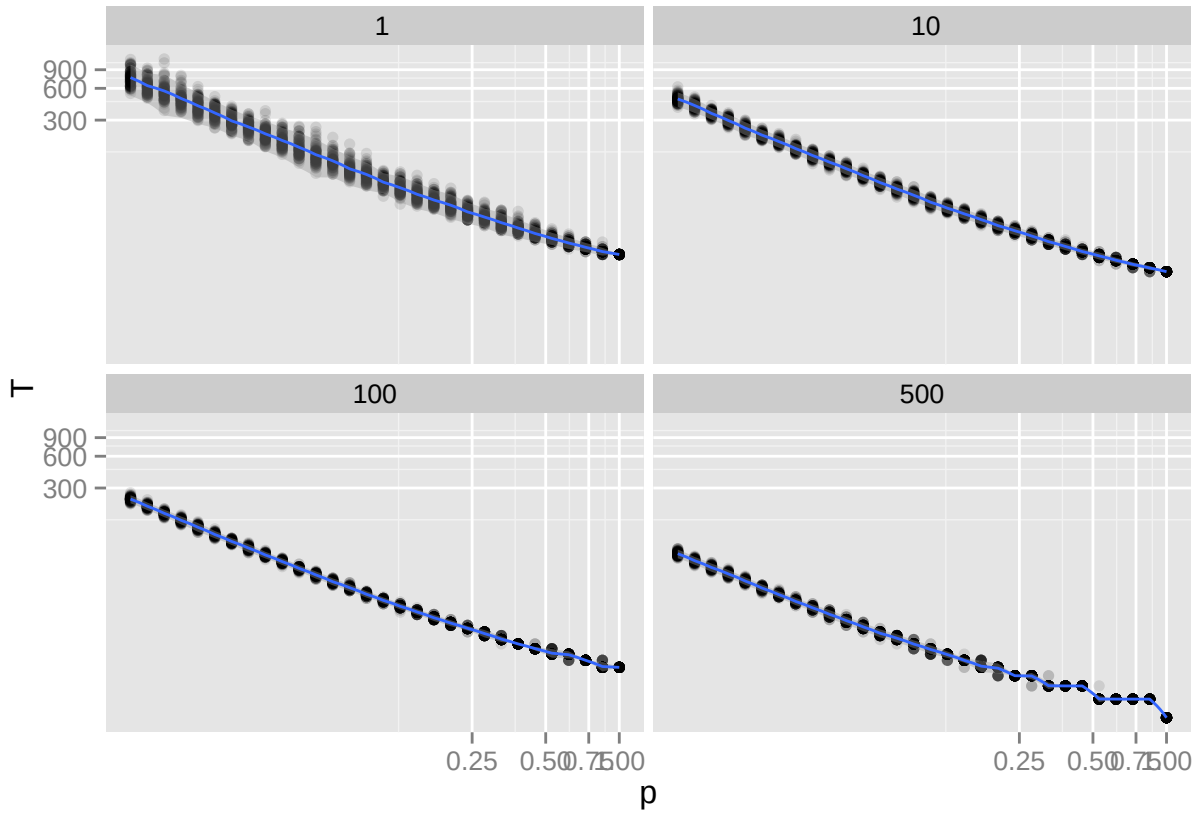
2. Faire des simulations permettant d'évaluer le temps moyens pour atteindre une population de bactérie de taille 1000 en partant de $n_0 = \{1, 2, 5, 10\}$ et pour des valeurs de p échelonnées entre 0.01 et 1.

```
# paramètre des simulations
nmax <- 1000
n0.seq <- c(1,10,100,500)
p.seq <- 10^seq(-2,0,len=30)
nsim <- 100

res <- mclapply(n0.seq, function(n0) {
  res <- sapply(p.seq, function(p) {
    return(replicate(nsim, PopBacteries2(n0, nmax, p)))
  })
  return(data.frame(T=c(res), p=rep(p.seq, each=nsim)))
}, mc.cores = 4)

res <- do.call(rbind, res)
res$n0 <- rep(n0.seq, each=length(p.seq)*nsim)

ggplot(res, aes(x=p, y=T)) + geom_point(alpha=.1) +  stat_summary(fun.data = "mean_sdl", geom = "smooth
```


1
10
100
500
900
600
300
T
0.25
0.50
0.75
1.00
p