Projet R 1

Table des matières

Dans ce projet on étudie les propriétés statistiques de quatre jeux de société par simulations.

0. Instructions

- Vous rendrez un rapport écrit sous format `.Rmd` **et** `.pdf` (ou `.html`).
- Vous enverrez vos fichiers `NOM1_NOM2.Rmd` et `NOM1_NOM2.pdf` dans un email avec le sujet **Projet Programmation R 2017** à mon adresse vittorio.perduca@parisdescartes.fr
- Vous écrirez les noms des auteurs dans le format **NOM1 Prénom1, NOM2 Prénom2.**
- Vous écrirez vos programmes `R` complets dans des blocs de code, de façon à ce que le lecteur puisse les faire tourner facilement, et vos commentaires et explications dans des paragraphes de texte.
- La notation prendra en compte le soin et la clarté des réponses.

1. Exercice d'échauffement

1.1 Prise en main

On commence par un jeu très simple, avec une seule urne contenant cinq boules et une pièce de monnaie. Chaque tour du jeu consiste en deux opérations :

- avant de tirer la pièce, on fait un choix x entre pile ou face
- on tire la pièce : si on obtient x on enlève une boule de l'urne.

On répète ces deux opérations jusqu'il y n'ait plus de boules dans l'urne.

On se pose la question d'estimer le nombre moyen de tours dans une partie : pour cela on simulera un très grand nombre de parties pour ensuite calculer la moyenne de leur durées.

Chaque tour de la partie peut être représenté par un vecteur `tour` avec quatre composantes : le nombre de boules dans l'urne, le choix, le résultat du lancement de la pièce, et le numéro du tour. Une partie sera donc représentée par le data frame `partie` dont les lignes sont les tours.

On peut simuler une partie avec la fonction suivante :

```
une_partie1=function(){
  #initialisation
  tour=c(urne=5,choix=NA,piece=NA,ntour=0)
  partie=rbind(tour)

  #tours
  while(tour[1]!=0){
    tour[4]=tour[4]+1

    choix=1 #dans chaque tour on choisit pile
    tour[2]=choix

    piece=sample(1:2,size=1) #codes: 1=pile, 2=face
    tour[3]=piece

    if(choix==piece) tour[1]=tour[1]-1

    partie=rbind(partie,tour)
  }
  return(partie)
}
```

Notez que dans cette fonction, à chaque tour on choisit pile, mais rien ne nous aurez empêché d'opter pour une autre stratégie.

```
une_partie1()
```

```
##      urne choix piece ntour
## tour    5    NA    NA     0
## tour    4     1     1     1
## tour    3     1     1     2
## tour    3     1     2     3
## tour    3     1     2     4
## tour    3     1     2     5
## tour    2     1     1     6
## tour    1     1     1     7
## tour    1     1     2     8
## tour    0     1     1     9
```

On simule $n = 100$ parties, on récupère la durée de chaque partie simulée l_i et on estime la durée moyenne par

$$\hat{l} = \frac{\sum_{i=1}^{n} l_i}{n}$$

Cette procédure d'estimation est dite estimation par **méthode de Monte-Carlo.**

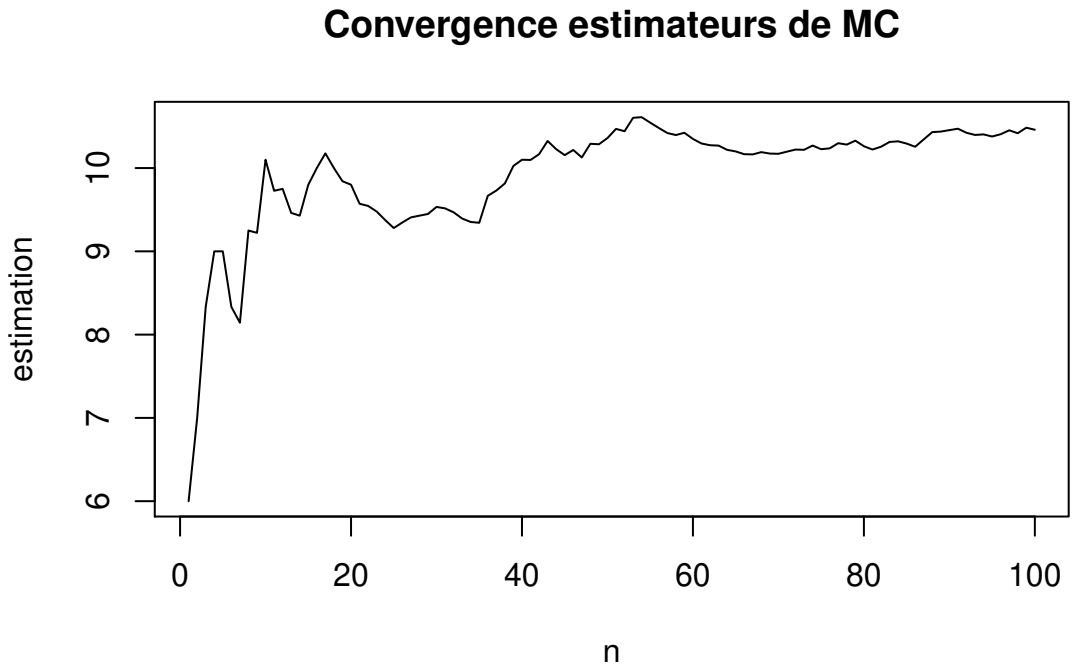
```
n=100
res=replicate(n,une_partie1())
duree=sapply(res,FUN=function(partie) nrow(partie)-1)
mean(duree)
```

```
## [1] 10.46
```

On peut monitorer la converge de l'estimation pour savoir si le n choisi est suffisant :

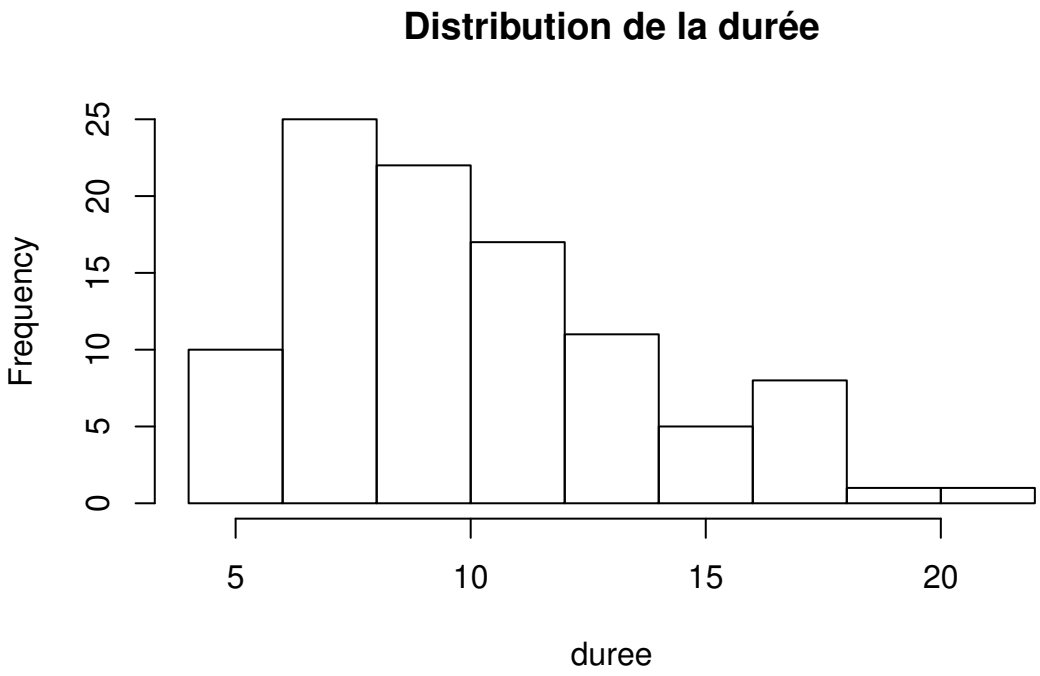
```
plot(cumsum(duree)/(1:n),type='l',
     main='Convergence estimateurs de MC',
```

```
     xlab='n',
     ylab='estimation')
```



On peut aussi regarder la distribution empirique de la durée :

```
hist(duree, main='Distribution de la durée')
```



1.2 Un jeu un peu plus compliqué

On considère maintenant un jeu avec deux urnes appelées respectivement “pile” et “face”, chacune avec cinq boules. Chaque tour consiste en deux opérations :

— avant de tirer la pièce, on fait un choix x entre pile ou face (ce choix est à discrétion du joueur).
— on tire la pièce : **si on obtient x on enlève une boule de l'urne x ; si non n'obtient pas x on ne fait rien.**

La partie termine quand il n'y a plus de boules dans les deux urnes. On remarque que dans ce jeu, comme dans le précédent, le choix de l'urne dans la première étape de chaque tour ne va pas affecter la durée de la partie.

Questions :

1. Ecrire une fonction `une_partie2()` qui simule une partie. Attention : les urnes ne peuvent pas contenir un nombre négatif de boules, donc on ne choisira pas l'urne x si x n'a plus de boules !
2. Estimer par la méthode de Monte-Carlo la durée moyenne d'une partie et son écart type. Choisir un nombre n de simulations suffisamment grand.

1.3 Un jeu qui demande un peu de strategie ?

On complique un peu le jeu précédent de la façon suivante :

— avant de tirer la pièce, on fait toujours un choix x entre pile ou face.
— on tire la pièce : **si on obtient x on enlève une boule de l'urne x ; si on n'obtient pas x on remet dans l'urne x les boules qu'on lui avait précédemment enlevé de façon à qu'elle ait à nouveu toutes ses cinq boules.**

La partie termine quand il n'y a plus de boules dans les deux urnes.

4. Ecrire une fonction `une_partie3()` qui simule une partie en implémentant la stratégie consistant à choisir l'urne avec le nombre maximal de boules.
5. Estimer par la méthode de Monte-Carlo la durée moyenne d'une partie et son écart type.
6. Comparer le résultat obtenu avec l'estimation qu'on obtient quand la stratégie consiste à choisir l'urne avec le nombre minimal de boules. Que faudrait-il faire d'un point de vue statistique pour pouvoir conclure sur la stratégie qui minimise la durée moyenne des parties ?

2. Le jeu Chut, Coco !

FIGURE 1 – Chut, Coco ! est un jeu pour enfants à partir de 2 ans.

On considère maintenant le jeu Chut, Coco ! de Haba.

7. Lire attentivement les instructions qu'on peut trouver dans le fichier annexe

Par rapport aux jeux de la section précédente, dans ce jeu on peut perdre une partie. Cela arrive si on retourne la dernière carte de jour avant que tous les petits animaux ne soient endormis.

8. D'un point de vue intuitif, quelle stratégie adopteriez-vous pour minimiser la durée d'une partie ?
9. Ecrire une fonction `partie4()` qui simule une partie en implémentant la stratégie ci-dessus. De chaque tour on retiendra le nombre de chaque petit animal, le nombre de cartes de jour, le coin choisi pour mettre le coq et le résultat du dé.
10. Estimer par la méthode de Monte-Carlo, la probabilité de gagner une partie, la durée moyenne des parties et son écart type.