Introduction à R

Exercices, Simulation

Pascal Bessonneau

05/2016

La loi normale

La fonction *rnorm* permet de créer un vecteur contenant des nombres tirés de la loi normale :

```
> a <- rnorm(10000)
> plot(a)
```

La loi normale

La fonction *rnorm* permet de créer un vecteur contenant des nombres tirés de la loi normale :

```
> hist(a)
```

La loi normale

La fonction *rnorm* permet de créer un vecteur contenant des nombres tirés de la loi normale :

```
> a <- rnorm(10000,mean=2,sd=4)
```

La loi normale

La fonction *rnorm* permet de créer un vecteur contenant des nombres tirés de la loi normale :

```
> hist(a)
```

Les dés

On peut aisement sous R simuler un tirage d'une pièce :

```
> a <- rbinom(10000,size = 1, prob = 1 / 2 )
> mean(a)
> sd(a)
> table(a)
```

La convergence vers la loi normale

On peut simuler k tirages de n pièces :

```
> k <- 10
> n <- 10
> m <- c()
> for (ii in 1:k) {
+   m <- c( m, mean( rbinom(n, size = 1, prob = 1/2 ) ) )
+ }
> m
```

La convergence vers la loi normale

```
> hist(m)
```

La convergence vers la loi normale

```
> k <- 10000
> n <- 10
> m <- c()
> for (ii in 1:k) {
+   m <- c( m, mean(rbinom(10,size = 1, prob = 1 / 2 )) )
+ }
```

La convergence vers la loi normale

```
> hist(m)
```

La convergence vers la loi normale

On dit en statistiques qu'il y a convergence vers la loi normale. Quand on tire une pièce, la valeur obtenu suit une loi binomiale. C'est lié à la méthode de tirage et au *design* de l'expérience. Cette moyenne varie car on tire un échantillon des valeurs possibles. En conséquence la moyenne obtenue va varier pour chaque échantillon. Et la valeur de la moyenne va varier en suivant une loi normale.

Simuler une matrice de k x k nombres aléatoires

la loi peut être la loi uniforme, la loi normale, . . .
Le but, obtenir une matrice de k x k avec k^2 nombres aléatoires.
Etonnez moi !
Un conseil commencez avec k égal 10 ou 50. Puis avec k égal à 1000.

Simuler une matrice de k x k nombres aléatoires

Pour créer une matrice la commande est :

```
> matrix(valeurs,nrow=k,ncol=k)
```

Proposition 1

```
> matrice <- matrix(NA,nrow=k,ncol=k)
> for (ii in 1:k) {
+   for (jj in 1:k) {
+     matrice[ii,jj] <- rnorm(1)
+   }
+ }
```

Proposition 2

```
> matrice <- matrix(NA,nrow=k,ncol=k)
> for (ii in 1:k) {
+   matrice[ii,] <- rnorm(k)
+ }
```

Proposition 3

```
> matrice <- matrix(1,nrow=k,ncol=k)
> matrice <- apply(matrice,1:2,rnorm)
```

Proposition 4

```
> matrice <- rnorm(k^2)
> dim(matrice) <- c(k,k)
>
> # je sais, c'est pas esthétique.
```

Mouvements brownien

Random walking : un homme avance de pas en pas à chaque fois d'une valeur tiré dans une loi normale dans la direction x et idem pour la direction y. Réaliser les simulations pour n points

Mouvements brownien

```
> n <- 10000
> position.x <- c(0)
> position.y <- c(0)
> for (ii in 1:n) {
+   position.x <- c(position.x,position.x[ii-1]+rnorm(1))
+   position.y <- c(position.y,position.y[ii-1]+rnorm(1))
+ }
> plot(0,0,type="n",xlim=range(position.x),ylim=range(position.y))
> lines(position.x,position.y)
```

Mouvements brownien

```
> n <- 10000
> position.x <- cumsum(c( 0, rnorm(n)))
> position.y <- cumsum(c( 0, rnorm(n)))
> plot(0,0,type="n",xlim=range(position.x),ylim=range(position.y))
> lines(position.x,position.y)
```