Correction TP1 : Initiation au logiciel "R"

1 Premiers pas avec "R"

Tapez les commandes suivantes dans la fenêtre "R".

```
a = 5
a
a + 3
b = 11
a + b
c = a + b
c
```

Vecteurs et Matrices

Tapez les commandes suivantes et observez le résultat.

```
a = c(1,4,2,5,2,7,10,23,1,6)
a
length(a)
b = c(1 :10)
b
c = a + b
a
b
c
d = 1/c
d
e = 2*a
couleur = c("noir","rouge","vert","bleu")
couleur
length(couleur)
couleur[1]
couleur[3]
```

Tapez les commandes suivantes et observez le résultat.

```
f = rep(0,times=10)    //génère un vecteur de longueur 10 et initialiser à 0
x = c(1,3,5)
```

```
g = rep(x,times=2)
g
h = rep(x,each=3)
h
```

Dans la fenêtre, tapez les commandes suivantes et observez le résultat.

mat = matrix(0,10,5)

mat

dim(mat)

mat[1,3]

mat[1,]

mat[,2]

mat[1,] = 11 //Affecte 11 à toutes les cases de la ligne 1

mat

val = c(-4 :5)

mat[,4] = val

mat

Tapez les commandes suivantes et observez le résultat :

```
x = c(-2,0,1,-1,3,-4,3,-5)
x[x >= 0]
x[x == 3]
```

Gestion des variables

Tapez dans la fenêtre les commandes suivantes et observez le résultat.

```
ls()
rm(a,b,c)
ls()
```

Définition de fonctions

Tapez dans la fenêtre les commandes suivantes.

```
f←function(x){return(x*x)}
f(2)
a = c(0 :10)
b = f(a)
b
```

Tapez les commandes et observez le résultat.

```
h←function(x)
{
y1 = x*x ;
y2 = x+1 ;
```

```
z = y1/y2 ;
return(z) ;
}
h(2)
a = c(0 :10)
b = h(a)
b
```

Les tests et les boucles

Fonction Comb(n,p)

```
Comb ← function(n,p)
{
 if(n<p)
  return(0)
 else
  return(factorial(n)/(factorial(p)*factorial(n-p)))
}
```

Pascal(n)

```
Pascal ← function(n)
{
  x = matrix(0,nrow=n+1,ncol=n+1) ;

  for(j in 0 :n)
  {
   for(i in j :n)
   {
    x[i+1,j+1] = Comb(i,j)
   }
  }

  return(x)
}
```

Exercice

Fonction Discretise(a,b,n)

```
Discretise ← function(a,b,n)
{
 x = rep(0,times=n+1) ;
```

```
 for(i in 0 :n)
 {
  x[i+1] = a + i*(b-a)/n ;
 }
 return(x)
}
```

2 Passons aux Probabilités

Les Histogrammes

Dans la fenêtre, tapez les commandes suivantes.

```
> u = runif(100)
> hist(u)
> u = rnorm(100)
> hist(u)
> u = rnorm(1000)
> hist(u)
> u = rnorm(10000)
> hist(u)
```

Générez une série de 1000 nombres aléatoires suivants les distributions ci-dessous et tracez l'histogramme correspondant (consultez l'aide en ligne pour les paramêtres des différentes fonctions) :

- poisson de paramêtre λ = 5 (fonction `rpois()`) :
  `> u = rpois(1000,lambda=5)`
- binomiale de paramêtre `n = 20` et `p = 0.1` (fonction `rbinom()`) :
  `> u = rbinom(1000,20,0.1)`
- Bernouilli de paramêtre `p = 0.1` (loi binomiale sur 1 tirage : `rbinom(n,1,p)`) :
  `> u = rbinom(1000,1,0.1)`
- exponentielle de paramêtre λ = 5 (fonction `rexp()`) :
  `> u = rexp(1000,5)`

Loi normale de moyenne $\mu = 100$ et d'écart-type $\sigma = 5$

```
> u = rnorm(1000,mean=100,sd=5)
> hist(u)
> u = rnorm(1000,mean=100,sd=7)
> hist(u)
> u = rnorm(1000,mean=100,sd=10)
> hist(u)

> u = runif(10000,min=1,max=100)
> hist(u)
> mean(u)
> [1] 50.69143
```

```
> var(u)
[1] 823.4137
> sd(u)
[1] 28.69519
```

Exemple :

```
> u = runif(1000,min=0,max=100)
> hist(u)
> hist(u,breaks=discretise(0,100,5))
```

La fonction `plot()`

Tapez les commandes suivantes et observez le résultat.

```
> f←function(x){x*x}
> x = c(-10 :10)
> y = f(x)
> plot(x,y)
> plot(x,y,type="l",col=2,pch="0")
> plot(x,y,type="l",col=2,pch="1")
> plot(x,y,type="l",col=2,pch="2")
> plot(x,y,type="l",col=5,pch="0")
> plot(x,y,type="l",col=6,pch="0")
```

3 Etude d'un générateur de nombres aléatoires

Q1.

```
> u = scan("rand.txt")
Read 20000 items
> mean(u)
[1] 0.4983168
> var(u)
[1] 0.0827913
> sd(u)
[1] 0.2877348
```

Ces valeurs sont très proches de la moyenne, de la variance et de l'écart-type d'une loi uniforme : $\frac{1}{2}$, $\frac{1}{12} = 0.08333333$ et $\sqrt{\frac{1}{12}} = 0.2886751$.

Q2.

```
> u1 = scan("rand30.txt")
Read 20000 items
> u2 = scan("rand30-2.txt")
Read 20000 items
```

Q3.

```
> hist(u1,breaks=discretise(0,30,20),right=FALSE)$counts
```

En observant l'histogramme, on voit que les effectifs dans des classes successives sont très différents. Cela peut s'expliquer par le fait que les valeurs du fichier sont entières alors que les intervalles sont de longueurs 1.5. Les différentes valeurs sont donc réparties de manière inégales dans les différentes classes. Pour mieux observer la répartition des nombres, on doit divisez l'intervalle $[0, 30]$ en des classes de longeur entière (par exemple de longueur 2).

Q4.

```
> mean(u1)
[1] 14.58165
> var(u1)
[1] 74.81147
> sd(u1)
[1] 8.649363
```

On constate qu'en générant un plus grand échantillon de nombre entre 0 et 30, les moyennes et l'écart-type se rapproche plus de la moyenne et l'écart-type d'une loi uniforme sur $[0, 30]$.

Q5. Test du χ^2

Pour le test du χ^2, on divise l'intervalle $[0, 30]$ en 15 classes. Il faut utiliser le paramêtre `rigth=FALSE` avec la fonction `hist()` pour que "R" répartisse les valeurs dans des intervalles de forme $[a, b[$ (c'est-à-dire $a \leq x < b$). Par défaut, le paramêtre `right` est à `TRUE`, ce qui correspond à des intervalles de la forme $]a, b]$.

```
> obs = hist(u1,breaks = discretise(0,30,15), rigth=FALSE)$counts
> theo = rep(2/30,times=15)
> chisq.test(obs,p=theo)

        Chi-squared test for given probabilities
data : obs
X-squared = 11.62, df = 14, p-value = 0.6368
```

La valeur `X-squared` correspond à la valeur du χ^2. Le degrés de liberté est donné par `df = 14` (le nombre de classes - 1). La valeur du χ^2 correspondant est donc 11.62. Pour un niveau de confiance $100(1-\alpha) = 99\%$, on doit comparer cette valeur au quantile d'ordre $1-\alpha$ d'une loi du χ^2 à 14 degrés de liberté. On peut obtenir cette valeur grâce à la fonction `qchisq()` :

```
> qchisq(0.99,df=14)
[1] 29.14124
```

La valeur du χ^2 étant $<$ à 29.14, on accepte l'hypothèse selon laquelle l'échantillon `u1` est distribué suivant une loi uniforme.

Q6.

```
> obs = hist(u2,breaks = discretise(0,30,15), rigth=FALSE)$counts
> theo = rep(2/30,times=15)
> chisq.test(obs,p=theo)

        Chi-squared test for given probabilities
```

```
data : obs
X-squared = 368.5, df = 14, p-value < 2.2e-16
```

La valeur du χ^2 correspondant est largement supérieure au quantile d'ordre 0.99 d'une loi du χ^2 à 14 degrés de liberté. On conclue donc que l'échantillon `u2` n'est pas distribué suivant une loi uniforme. On remarque d'ailleurs que la proportion de nombre dans la tranche $[28, 30]$ est supérieure à celle des autres tranches.

Q7. En conclusion, pour générer en C/C++ des nombres entier suivant une loi uniforme entre a et b, il vaut mieux utiliser la méthode `x = (int)(a + (b-a)*(1.0*rand()/RAND_MAX))`.

4 Théorème central limite

Ecrivez la fonction suivante

```
Genere = function(n,N)
{
x = matrix(rep(0,times=n*N),nrow=N,ncol=n,byrow=TRUE) ;
y = rep(0,times=n) ;

for(i in 1 :N)
x[i,] = rexp(n,rate=0.5) ;

for(i in 1 :n)
y[i] = sum(x[,i]) ;

return(y) ;
}

> x = genere(10000,1)
> hist(x)
> x = genere(10000,2)
> hist(x)
> mean(x)
[1] 3.973125
> 2*1/0.5
[1] 4
> x = genere(10000,5)
> hist(x)
> mean(x)
[1] 9.922085
> 5*1/0.5
[1] 10
> x = genere(10000,10)
> hist(x)
> mean(x)
[1] 19.98233
> 10*1/0.5
[1] 20
```

```
> x = genere(10000,20)
> hist(x)
> mean(x)
[1] 40.04896
> 20*1/0.5
[1] 40
> x = genere(10000,30)
> hist(x)
> mean(x)
[1] 60.0328
> 30*1/0.5
[1] 60
```