SY09 : Rapport TP0

Introduction à R

CATHELAIN Valentin MARECHAL Anaig

29 mars 2016

Résumé

R est un logiciel de traitement statistique des données. C'est également un langage de programmation interprété dérivé de S. L'objectif de ce premier TP est de prendre le logiciel en main et d'en apprendre quelques fonctions principales.

Chapitre 1

Prise en main de R

1.1 ByRow

Nous travaillons ici avec la matrice A, créée grâce à la commande

```
matrix(1:9, nrow=3, byrow=T)
```

Pour tester l'effet de l'argument **byrow** , nous allons successivement créer la matrice avec *byrow=T* et *byrow=F*.

Table 1.1 – byrow= TRUE

1	2	3
4	5	6
7	8	9

Table 1.2 – byrow= FALSE

1	4	7
2	5	8
3	6	9

On observe que quand l'argument R **byrow** est activé, les données informées dans la liste sont remplies ligne à ligne. Dans le cas contraire, elles sont remplies colonne par colonne. L'argument permet donc de préciser la manière dont on veut remplir la matrice.
Si on enlève l'argument de l'instruction, la matrice est remplie colonne par colonne. On en déduit que par défaut, la valeur de **byrow** est false.

1.2 Les opérations * et %*%

Toujours en travaillant sur notre matrice A, nous allons comparer les deux instructions

```
A\%*\%A
```

et

```
A*A
```

Table 1.3 – A%*%A

30	36	42
66	81	96
102	126	150

Table 1.4 – A*A

1	4	9
16	25	36
49	64	81

On en déduit que l'opérateur * multiplie les composants un à un tandis que l'opérateur %*% permet de faire des produits matriciels.

1.3 Programmation

Voici la fonction *podtrans* qui permet de calculer le produit d'un matrice X avec sa transposée :

```
podtrans <- function(X) {
t(X)%*%X;
}
```

Voilà ensuite comment exécuter cette fonction sur les matrices A et B :

```
A <- matrix(1:6, nrow=3, byrow=T)
X <- podtrans(A)
X

B <- matrix(c(5,3,7,4,6,3,1,6,3,2,8,5), nrow=4, byrow=T)
Y <- podtrans(B)
Y
```

1.4 Indicateurs numériques, graphiques de base

On tape successivement les fonctions :

```
apply(A,1,max)
apply(A,2,max)
```

On observe que dans le premier cas, la fonction **apply** renvoie le maximum de chaque ligne, et dans le second cas la fonction renvoie le maximum de chaque colonne. On en déduit que apply prend en argument une matrice, 1 ou 2 pour appliquer en ligne ou en colonne puis la fonction à appliquer (max dans notre exemple).
Les fonctions **var** et **sd** permettent respectivement de calculer la variance et l'écart-type d'un vecteur ou d'une matrice.

1.5 Centrage en colonne

Voici une fonction possible pour centrer une matrice en colonne :

```
centre <- function(X){
  rows <- nrow(X); #nombre de lignes de la matrice
  means <- apply(X,2,mean); #moyenne de chaque colonnes
  matrixmeans <- matrix(1,rows,1)\%*\%mean;
#matrice de même dimension que X, contenant les moyennes de chaque colonne sur chaque ligne
        X-matrixmeans #soustraction
}
```

La matrice de covariance empirique est donnée par la formule : $S_n^2 = \frac{1}{n}\sum_{j=1}^{n}(X_j - \overline{X}_n)(X_j - \overline{X}_n)^T$.
Pour la calculer à partir des fonctions centre et podtrans, nous avons procédé ainsi :

```
  covariance <- function(X){
  rows <- nrow(X);
  centre <- centre(X);
  (podtrans(centre))*(rows-1)^-1;
 }
```

Ainsi, la fonction covariance appliquée sur une matrice B nous renvoie bien le même résultat que la fonction **cov()** de R.

```
> B <- matrix(c(5,3,7,4,6,3,1,6,3,2,8,5, 1, 4, 5, 3), nrow=4, byrow=T)
> B
     [,1] [,2] [,3] [,4]
[1,]    5    3    7    4
[2,]    6    3    1    6
[3,]    3    2    8    5
[4,]    1    4    5    3
> covariance(B)
           [,1]       [,2]      [,3]       [,4]
[1,]  4.9166667 -0.6666667 -2.916667  2.1666667
[2,] -0.6666667  0.6666667 -1.000000 -0.6666667
[3,] -2.9166667 -1.0000000  9.583333 -1.8333333
[4,]  2.1666667 -0.6666667 -1.833333  1.6666667
> cov(B)
           [,1]       [,2]      [,3]       [,4]
[1,]  4.9166667 -0.6666667 -2.916667  2.1666667
[2,] -0.6666667  0.6666667 -1.000000 -0.6666667
[3,] -2.9166667 -1.0000000  9.583333 -1.8333333
[4,]  2.1666667 -0.6666667 -1.833333  1.6666667
```

1.6 Fonction hist.factor

La fonction hist.factor permet d'afficher un histogramme montrant dans chaque "bin" les effectifs selon les modalités de la variable qualitative.

```
hist.factor<-function(x, y){
    inter <- seq(min(x), max(x), by=(max(x) - min(x))/10)
    h <- hist(plot=F, x[y == levels(y)[1]], breaks=inter) \$count
    for (i in 2: nlevels(y))
      h <- rbind(h, hist(plot=F,x[y==levels(y)[i]], breaks = inter) $count)
    barplot(h, main='Iris', space=0, legend = levels(y), col=c('green', 'pink', 'blue')
}
hist.factor(iris$Sepal.Length, iris$Species)
```

Voici le résultat obtenu :

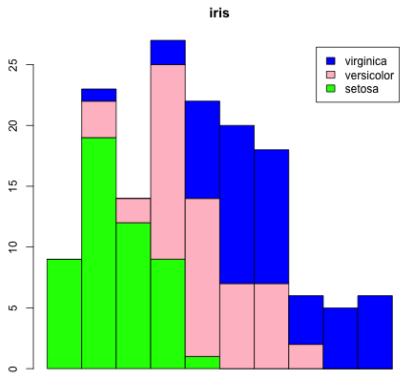

FIGURE 1.1 – Résultat de la fonction hist.factor.