Chapitre 4

Régression linéaire

Petit mot sur l'importance de la régression (historique ?)...

4.1 Modèle de régression linéaire simple

4.1.1 Introduction

Rappels sur le modèle linéaire : Y=aX+b+"bruit", retour sur les différentes quantités impliquées, etc.+ prédiction -> cf manuscrit

4.1.2 Fonctionnalités de la commande lm

L'étude d'un modèle de régression peut-être réalisé à l'aide de la commande lm. Cette dernière propose de nombreuses fonctionnalités que nous allons étudier à l'aide d'un exemple simple. Considérons les vecteurs x et y définis par :

```
> x <- c(1,4,6,8,2,4,8,9)
> y <- 2*x + rnorm(8,0,1)
```

Pour obtenir la régression de y en fonction de x, il suffit de taper :

```
> Reg <- lm(y~x)
> Reg

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   -0.02772      2.00395
```

Utilisée telle quelle, la commande lm renvoie simplement les deux coefficients de régression estimés. A noter : la commande lm admet un certain nombre d'options que nous n'utiliserons pas dans ce cours (voir donc l'aide en ligne pour plus de détails).

Comme pour les commandes chisq.test ou var.test, lm produit plus d'informations que ce qui est affiché. En fait, cette commande est tellement importante qu'une classe particulière a été crée. Pour plus de détail, il est possible d'utiliser :

```
> attributes(Reg)
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

```
$class
[1] "lm"
```

Pour accéder à chacune des informations contenues dans la rubrique names, on utiliser Reg$nom, où nom correspond à l'information désirée. Par exemple, pour obtenir les résidus, on tapera :

```
> Reg$residuals
         1          2          3          4          5          6          7
-0.3334642 -0.3041610  0.2804465  1.3167916 -0.5129902  1.1945033 -0.3511980
         8
-1.2899281
```

En ce qui nous concerne, les rubriques les plus utiles seront : coefficients, residuals, fitted.values et model (pour récupérer les données sous la forme d'un data.frame).

Avec toutes ces informations, il est maintenant possible de tracer la droite de régression. On utilise pour cela la commande abline permettant de tracer des droites affines :

```
> plot(x,y)
> abline(Reg$coef)
```

Remarque : La commande plot(Reg) permet d'afficher quatre graphiques, utiles pour une étude approfondie du modèle de régression. Nous n'aborderons pas l'utilisation de cette commande dans ce module.

4.1.3 Test du coefficient de corrélation linéaire

Ce test est équivalent à celui du caractère significatif de la régression (i.e. nullité de la pente de régression). La fonction utilisée pour réaliser cette opération est cor.test. Cette dernière prend en argument les deux vecteurs x et y. Il est également nécessaire de préciser le type de test utilisé par l'intermédiaire de l'option method. Reprenons l'exemple traité précédemment. Dans la mesure où nous nous sommes intéressés au test de Pearson, on utilise :

```
> cor.test(x,y,method="pearson")

        Pearson's product-moment correlation

data:  x and y
t = 16.0598, df = 6, p-value = 3.704e-06
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9357623 0.9980100
sample estimates:
      cor
0.9885675
```

Au vu de la p-valeur, nous sommes amenés à conclure à la non nullité du coefficient de corrélation linéaire.

Rappel : La nullité du coefficient de corrélation linéaire n'implique pas l'indépendance de X et de Y. On peut seulement en conclure qu'il n'y a pas de relation linéaire entre X et Y, d'autre liens sont cependant envisageables.

4.1.4 Prévision et fonctionnalités de la commande predict

Nous allons maintenant nous intéresser à la prévision de nouvelles valeurs. Cette opération peut être réalisée à l'aide de la commande predict. Cette dernière prend comme argument principale un objet dont la classe est issue de la fonction lm. L'instruction :

```
> predict(Reg)
       1        2        3        4        5        6        7        8
2.391090 8.486159 12.549537 16.612916 4.422780 8.486159 16.612916 18.644606
```

renvoie simplement la valeur des estimateurs $\hat{Y}_i$. Pour obtenir un intervalle de confiance de ces derniers, il suffit d'utiliser :

```
> predict(Reg,interval="confidence")
       fit        lwr       upr
1  2.391090  0.8202835  3.961897
2  8.486159  7.5446918  9.427626
3 12.549537 11.6604158 13.438659
4 16.612916 15.4040754 17.821757
5  4.422780  3.1003714  5.745188
6  8.486159  7.5446918  9.427626
7 16.612916 15.4040754 17.821757
8 18.644606 17.2009493 20.088262
```

On obtient un data.frame précisant les intervalles de confiance pour les termes $\mathbb{E}[Y/X = x]$.

Si on souhaite maintenant prédire de nouvelles valeurs, il suffit de créer un data.frame que l'on rajoutera en argument de la fonction predict. Reprenons l'exemple introduit précédemment. Pour prédire les valeurs du modèle aux points 10 et 11, il suffit de taper les instructions suivantes :

```
> nouveau <- data.frame(x=c(10,11))
> predict(Reg,newdata=nouveau)
       1        2
20.67630 22.70798
```

Il est également possible d'obtenir un intervalle de confiance pour ces prédictions en rajoutant simplement l'option interval="confidence".

Important : Le vecteur contenu dans le data.frame doit avoir le même nom que le vecteur entré en argument dans la commande lm.

Enfin, il est possible d'obtenir des intervalles de confiance non plus pour la valeur moyenne $\mathbb{E}[Y/X = x]$ mais pour la valeur de Y au point $X = x$ c'est-à-dire un intervalle de confiance tenant compte des erreurs éventuelles d'échantillonnage. Il suffit pour cela d'utiliser plutôt l'option interval="prediction".

4.2 Régression linéaire multiple

Dans le cadre de la régression linéaire multiple, les commandes sous **R** restent essentiellement les mêmes que dans le cas de la régression linéaire simple. Introduisons par exemple :

```
> x1 <- rnorm(10,0,1)
> x2 <- rnorm(10,0,1)
> y <- 2*x1-x2+rnorm(10,0,0.5)
```

Pour effectuer la régression de y en fonction des deux variables explicatives $x1$ et $x2$ par la méthode des moindres carrés, c'est encore la commande lm qui intervient :

```
> lm(y~x1+x2)

Call:
lm(formula = y ~ x1 + x2)

Coefficients:
(Intercept)           x1           x2
     0.0140       1.9858      -0.9552
```

L'ensemble des fonctionnalités de la commande lm sont conservées. Il sera en particulier intéressant de vérifier que les graphes des résidus en fonction des variables explicatives ne laissent apparaître aucune tendance.