Notes pratiques
pour les analyses linéaires sous R

Partie II : Régressions

Version 1.1 (novembre 2014)

Florent Aubry, Inserm/UPS U825, Toulouse

Ce document a comme objectif de donner des informations pratiques pour réaliser des analyses statistiques sous R. Ce n'est donc ni un cours de statistiques ni un document de référence et d'aide sur les fonctions citées dans ce document. Les explications 'statistiques', quand il y en a, ne sont qu'illustratives. Elles ne peuvent être considérées comme des développements formels mais seulement comme des indications intuitives pour permettre de mieux comprendre le propos.

En ce qui concerne l'utilisation des fonctions et la signification de leurs arguments, se reporter à leurs pages d'aide en ligne (help("nom de la fonction") ou ?nomFonction), aux vignettes associées aux packages et autres documents accessibles en ligne.

Pour l'utilisation générale de R, voir mon document 'Programmer sous R'.

Pour des détails sur les modèles linéaires sous R, voir notamment le livre de J. Faraway "Practical Regression and Anova using R" (http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf) dont ce document peut être considéré comme un complément en ce qui concerne notamment la programmation des comparaisons planifiées et des tests *post-hoc*.

De nombreuses analyses peuvent être réalisées à partir de l'interface développée par John Fox, R commander. Bien que je ne l'ai pas utilisée pour ce document, je la recommande en pratique et de ne programmer à la console que si l'analyse qu'on désire réaliser n'est pas disponible à partir de cette interface.

C'est à l'utilisateur de faire la correspondance entre ce document et les possibilités offertes par cette interface.

Cette seconde partie traite des problèmes spécifiques de l'analyse de régression. Elle suppose que le lecteur ait pris connaissance de la première partie traitant des généralités.

N. B. :

1) Comme tout document de synthèse, ce document peut contenir des erreurs ou des imprécisions. Si vous en relevez, faites les moi connaître, ainsi que tout commentaire, propositions d'exemples, retours d'expérience, ... Mon adresse électronique est la suivante

'\u0066\u006C\u006F\u0072\u0065\u006E\u0074\u002E\u0061\u0075\u0062\u0072\u0079\u0040\u0069\u006E\u0073\u0065\u0072\u006D\u002E\u0066\u0072'

Pour l'obtenir en clair, il suffit d'imprimer cette chaîne de caractères.

2) Ce document présente les fonctions les plus couramment utilisées, c'est-à-dire répondant à une majorité de problèmes. Malgré tout, ces fonctions ne sont pas universelles et doivent utilisées avec discernement. L'objectif de ce document est donc de donner les bases permettant d'utiliser ces fonctions sans trop d'erreurs.

Table des matières

V - La régression

Ce chapitre ne traite que de points spécifiques de l'analyse de régression. Pour les généralités, de reporter aux premiers chapitres[1][2].

Les types de régression

Linéaire

La régression linéaire estime les coefficients de la relation affine entre une variable numérique dépendante et une variable numérique indépendante. Sa formule est donc

Y ~ X

Bien qu'on puisse utiliser aov pour cette opération, il est préférable d'utiliser lm. Pour les modèles à effets mixtes, on pourra choisir entre lme (nlme) et lmer (lme4).

Multiple

La régression multiple estime les coefficients de la relation linéaire entre une variable numérique dépendante et des variables numériques indépendantes :

Y ~ X.1 + X.2 + ...

Factorielle

La régression factorielle estime les coefficients de la relation multilinéaire entre une variable numérique dépendante et des variables numériques indépendantes, c'est-à-dire qu'elle estime les coefficients des interactions entre plusieurs variables indépendantes. Exemple :

Y ~ X.1 * X.2

Polynomiale

La régression polynomiale estime les coefficients de la relation entre la variable numérique dépendante et un polynôme de degré donné de la variable numérique indépendante. Comme les symbole * ou ^ a un sens particulier dans les formules (par exemple, les identités des formules : X.1 * X.1 ::= X.1 ; X.1 * X.2 équivalent à X.1 + X.2 + X.1:X.2 ::= (X.1 + X.2)^2), il est nécessaire d'utiliser la fonction I() pour interdire l'interprétation des symboles comme des symboles d'opération sur les formules. Exemple, estimation d'un polynôme de degré 2 :

Y ~ I(X) + I(X^2)

[1] Rappelons qu'on peut remplace n'importe quel régresseur par une fonction de ce même régresseur ou une fonction de plusieurs régresseurs. Cependant, hormis les cas où cette fonction est une fonction spline, un polynôme ou des fonctions simples, il est souvent préférable d'utiliser des méthodes non linéaires.
[2] Les principales fonctions utilisables sont répertoriées dans le premier chapitre.

À noter :

i) le I() dans I(X) est inutile mais peut-être utilisé pour homogénéiser la présentation de la formule.

ii) on peut estimer des fonctions polynômes des puissances de combinaisons linéaires de régresseurs.

iii) on peut étendre la régression polynomiale à la régression multiple ou factorielle.

Logistique

La régression logistique estime les coefficients de la relation entre une variable binaire (régression logistique) ou nominale ou ordinale multinomiale (régression logistique multinomiale) dépendante et des variables numériques indépendantes en termes de facteurs de risque car un des niveaux est considéré comme celui des succès et les autres, comme différents types d'échecs.

Cette régression peut être présenté différemment, par exemple en termes de matrice avec une colonne pour le nombre de succès et les autres colonnes pour le nombre de cas relevant des 'échecs'.

Note sur la régression sur des splines

Il est parfois possible, et même préférable, de remplacer la régression polynomiale par une régression sur un ajustement par splines. Cette solution se justifie surtout quand l'analyse a un objectif prédictif plus qu'elle ne se justifie dans le cas inférentiel pur, c'est-à-dire de l'analyse des liens entre caractéristiques à l'intérieur d'une population[3] car l'interprétation des résultats de l'analyse est plus délicate. R offre plusieurs possibilités qui n'aboutissent pas tous aux mêmes résultats.

La première solution est d'utiliser la fonction bs (du package splines) dans la formule. Cette solution utilise les B-splines qui sont explicitement codées dans l'appel à la fonction d'estimation, par exemple lm.

Une autre solution qui permet de tenir compte d'incertitudes sur la valeur des régresseurs est d'utiliser des splines pénalisantes ou lissantes. Hastie et Tibshirani ont développé un modèle linéaire spécial qui permet ce type d'analyse, le modèle additif généralisé[4] (*generalized additive model*.) Il en existe plusieurs implantations dans R, dont une dans le package VGAM (fonction vgam) et une autre dans le package gam (fonction gam). Les auteurs de ces packages soulignent que l'utilisation simultanée de ces deux packages peut engendrer des problèmes.

[3] Les splines sont des fonctions d'interpolation optimales dans certains espaces fonctionnels (dits espaces de Sobolev) dans lesquels les fonctions et leurs dérivées jusqu'à un certain ordre sont continues et de carré sommable. L'utilisation d'une méthode des moindres carrés ordinaires ou pondérés ne supposent que des fonctions de carré sommable.

[4] Hastie, T, Tibsharini, R, 1986, Generalized additive models, *Statistical Science*, 1(3), pp297-318.

Exemple :

J'utilise les données women du package datasets.

```
lm.p5 <- lm( weight ~ height + I( height^2) + I( height^3) + I( height^4) + I( height^5),
                        women)
lm.bs5 <- lm( weight ~ bs( height, df=5), women)
lm.vgl5 <- vglm( weight ~ bs( height, df=5), women, family=gaussianff)
lm.vga5 <- vgam( weight ~ s( height, df=5), women, family=gaussianff)
plot( predict( lm.p5) - women$weight ~ women$height, col=2, type="b", ylim=c( -1, 1))
points( predict( lm.bs5) - women$weight ~ women$height, col=3, type="b")
points( predict( lm.vgl5) - women$weight ~ women$height, col=4, type="b")
points( predict( lm.vga5) - women$weight ~ women$height, col=5, type="b")
```

Note : Je ne développerai pas plus cette approche donc les personnes intéressées par cette approche doivent consulter la littérature théorique et pratique (pages d'aide dans R) sur ce sujet.

Comparaisons planifiées

Une analyse de régression conduira à la détermination de plusieurs paramètres :

- d'une constante, l'ordonnée à l'origine,

- de paramètres multiplicatifs associés selon les cas, aux régresseurs seuls ou à leurs produits (régression linéaire, multiple ou factorielle) ou aux régresseurs et leurs puissances (régression polynomiale).

La significativité de ces paramètres est donnée par la fonction summary.

R donne des outils qui permettent de spécifier des tests *a priori* de ces paramètres, tests connus sous le nom de comparaisons planifiées. Ces comparaisons à une valeur donnée passent par l'utilisation de la fonction offset dans la formule (ou de l'argument offset de la procédure)[5], qui permet d'ajouter un terme à un prédicteur linéaire en forçant son **coefficient à 1** au lieu de l'estimer.

Je donne ci-dessous les principes de ces tests mais je ne passe pas en revue tous les cas possibles, au lecteur d'adapter les exemples à son problème. Sauf mention spéciale, j'utiliserai la fonction lm mais la procédure décrite vaudra pour toutes les procédures.

Test de l'ordonnée à l'origine

Le test de l'ordonnée à l'origine à la valeur zéro est donnée par l'analyse. Il faut cependant garder à l'esprit qu'une ordonnée à l'origine non significativement nulle n'a pas la même signification en termes de modèle des données que de forcer cette ordonnée à zéro dans la formule. En effet, dans le second cas, on fait l'hypothèse que l'on **sait** que cette ordonnée est nulle et qu'il y a donc une relation de proportionnalité tandis que dans le premier cas on

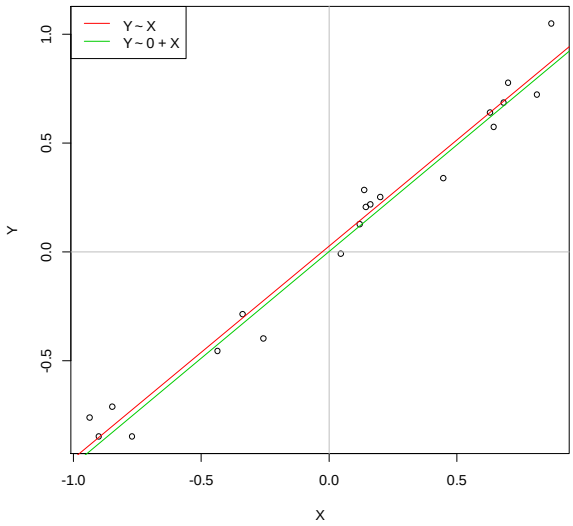
[5] Je rappelle ici que la fonction offset de la formule fait partie du modèle et est donc utilisé par la prédiction, ce qui n'est pas le cas de l'argument offset de la procédure.

constate a posteriori que cette ordonnée est sans doute nulle aux erreurs de type II près, c'est-à-dire quand on déclare cette ordonnée nulle alors qu'elle ne l'est pas. Cette hypothèse doit alors être intégrée dans le test de puissance *a posteriori* de l'analyse.

De ce fait,

```
lm.1 <- lm( Y ~ X, data=donnees)
lm.0 <- lm( Y ~ 0 + X, donnees)
```

ne donneront pas les mêmes estimations pour la pente de X même si Y est proportionnel à X à l'erreur près (*cf.* exemple à droite).



Pour tester l'ordonnée à l'origine à une valeur a_0 donnée, on utilisera de préférence l'argument offset de la procédure :

```
lm( Y ~ X + offset( rep( a0, nrow( donnees)), data=donnees)
```

On peut alors conclure que l'ordonnée à l'origine vaut a_0 si la fonction summary donne une valeur non significative pour l'*intercept*. ***A contrario*, forcer l'ordonnée à l'origine à zéro et obtenir un résultat significatif ne garantit pas que cette origine est statistiquement significativement différente de zéro.** Je laisse au lecteur le soin de vérifier cette assertion.

Test de la pente d'un régresseur

Pour tester si la pente d'un régresseur vaut une valeur donnée b_0, on retire aux données dépendantes la droite correspondante, ce qui donne la formulation suivante :

```
lm.o <- lm( Y ~ X + offset( b0 * X), data=donnees)
```

Si l'analyse donne une pente non significativement différente de zéro pour X, alors on peut conclure que la pente vaut b_0. En effet, cette formulation revient à tester le résidu Y - b_0 X par rapport à X. Si la pente à l'origine vaut b_0, cette différence doit être statistiquement non significativement différente de l'ordonnée à l'origine.

Ce résultat devrait être complété par une analyse des résidus pour s'assurer qu'il n'existe pas une relation entre les résidus et les variables[6], non révélée par l'analyse à cause de la plage de variation des variables.

[6] *Cf. infra*, analyse de la forme des résidus.

Exemple :

```
X <- runif( 20, min=-1, max=1)
W <- X + 0.1 * (X - 1)^2
```

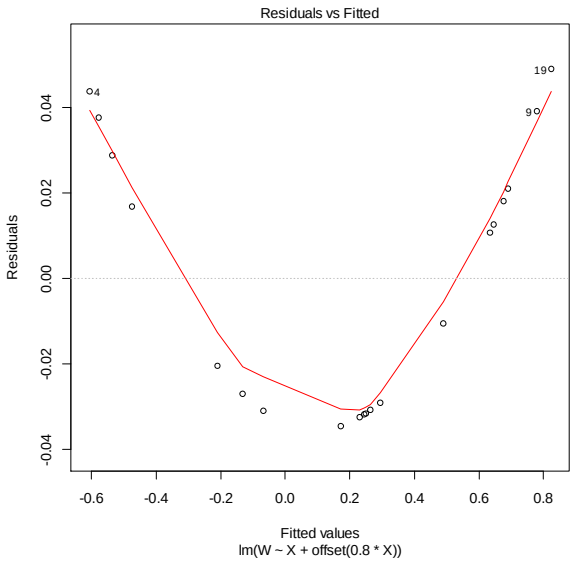
Une première analyse permet d'estimer le coefficient linéaire de lm(W ~ X) à 0.8 Refaisons alors l'analyse en testant le coefficient de X à cette valeur :

```
res.w <- lm( W ~ X + offset( 0.8 * X))
```

Comme prévu, on obtient bien un coefficient associé à X non significativement différent de zéro (p = 0.433). Cependant, le tracé ci-contre

```
plot( r.w, which=1)
```

conduit à soupçonner une relation polynômiale non détectable par l'analyse à cause de la plage des valeurs étudiées.



Une autre manière de faire est de comparer deux modèles :

```
anova( lm( Y ~ X, data=donnees), lm( Y ~ offset( b0 * X), data=donnees))
```

Si le différence entre les deux modèles est non significative, alors la pente vaut b_0. On peut vérifier que *p-value* de la différence entre les deux modèles est la même que celle de la pente de X dans le modèle lm.o.

Cependant, cette approche doit elle aussi être complétée par l'analyse des résidus.

Exercice :

Comment tester que la relation entre X et Y obéit à l'équation affine :

$$Y = a0 + b0 * X ?$$

Régression multiple

Pour comparer l'égalité des pentes entre deux régresseurs, on part de la contrainte suivante qui est vraie si les pentes sont égales : $a_{x.1}$ x1 + $a_{x.2}$ x2 = a (x1 + x2) avec a = $a_{x.1}$ = $a_{x.2}$.

Avec R cela peut se traduire par le code suivant, donné en exemple avec la fonction lm :

```
lm.2 <- lm( Y ~ I( X1) + I( X2), data=donnees)
lm.1 <- lm( Y ~ I( X1 + X2), data=donnees)
            anova( lm.1, lm.2)
```

Si la différence entre les deux modèles n'est pas significative, alors on peut conclure que les pentes sont identiques.

On peut aussi tester que la pente $a_{x.2}$ de X2 vaut k fois (k négatif ou positif) celle de X1, $a_{x.1}$. La contrainte à tester est alors : $a_{x.1}$ x1 + $a_{x.2}$ x2 = a (x1 + k * x2). On posera alors :

```
lm.1 <- lm( Y ~ I( X1) + I( X2), data=donnees)
lm.2 <- lm( Y ~ I( X1 + k * X2), data=donnees)
```

Si la différence entre les deux modèles n'est pas significative, alors on peut conclure que la pente de X2 vaut k fois celle de X1.

Cette procédure peut être étendue à plus de deux régresseurs.

Notes :

1) Il ne faut pas confondre ce test avec celui découlant de la formule :

```
Y ~ offset( X1 + k * X2)
```

qui vérifie que le résidu de Y - X1 + k * X2 est en moyenne non significativement différent de zéro, indépendamment du fait que ce soit une constante ou qu'il soit corrélé avec les variables.

2) Pour vérifier que le modèle est bien Y = a + X1 + k * X2, on le testera à partir de la formule :

```
Y ~ X1 + X2 + offset( X1 + k * X2)
```

Régression factorielle

La principale question est de savoir si la régression factorielle se justifie, c'est-à-dire si le coefficient de l'interaction X1:X2 est nul. On peut alors utiliser la même stratégie que pour la nullité d'une pente dans le cadre de la régression.

Régression polynomiale

On peut étendre les procédures présentées précédemment pour la régression multiple aux tests d'ajustements polynomiaux pour tester l'égalité des deux polynômes. Dans ce cas, on utilisera une procédure pas à pas en explorant chaque degré, en commençant par le plus élevé.

Par exemple, supposons qu'on veut tester si les coefficients des ajustements paraboliques des deux variables sont égaux, la séquence sera la suivante :

```
    lm.comp <- lm( Y ~ I( X.1) + I( X.2) + I( X.1^2) + I( X.2^2), data)
    lm.d2 <- lm( Y ~ I( X.1) + I( X.2) + I( X.1^2 + X.2^2), data)
    anova( lm.comp, lm.d2)
    if( anova( lm.comp, lm.d2)[2,"Pr(>F)"] < niveauTest) {     # difference significative
            stop( "le coefficient de degre 2 differe pour les deux variables")
    } else {
            lm.d1 <- lm( Y ~ I( X.1 + X.2) + I( X.1^2 + X.2^2), data)
            anova( lm.comp, lm.d1)
}
```

Régression logistique

Les tests précédents peuvent aussi être utilisés dans le cadre de la régression logistique.

Modèles mixtes

Dans les modèles mixtes, on peut vouloir tester des hypothèses sur le modèle fixe. Ces tests suivront la même logique que les tests précédents, à la différence qu'il faudra spécifier que la méthode d'ajustement est les moindres carrés (ML) et non les moindres carrés restreints (REML).

Les chapitres précédents ont donné des indications quant aux tests concernant les effets aléatoires, indications que je ne répète pas ici.

Résultats de l'analyse

La fonction summary donne

- pour lm, les valeurs des coefficients estimés et leur significativité en termes de t de Student ; Elle donne aussi le R^2 et le R^2 *ajusté* ;

- pour glm, l'AIC et non le R^2 ; celui-ci peut être obtenu par la fonction AIC ;

- pour vglm[7], la déviance et la log-vraisemblance.

Pour les modèles mixtes, elle donne en supplément des informations sur les effets aléatoires (variance, corrélations entre effets…)

Considérations pour améliorer l'interprétation des résultats

Corrélation des régresseurs

Les résultats d'une analyse sont d'autant plus fiable que les prédicteurs sont peu corrélés entre eux. La robustesse des coefficients se mesure la facteur d'inflation de la variance (ou *VIF*, fonction vif) dont les valeurs sont, rappelons le, la diagonale de l'inverse de la matrice de corrélation des estimateurs des paramètres. Il convient donc de minimiser cette corrélation. Dans le cadre de la régression, les paramètres estimés sont d'une part l'ordonnée à l'origine et, d'autre part, les coefficient de régression linéaire et, pour les régressions factorielles et polynômiales, les coefficients des interactions[8]. La solution simple pour arriver à cet objectif est de centrer les régresseurs sur leur valeur moyenne. On remplacera alors chaque régresseur (ici, X) par sa valeur centrée par scale(X, scale=FALSE).

Ce centrage est inutile pour les régressions simples et multiples.

[7] Il n'existe actuellement pas d'implantation de la fonction anova pour vglm.

[8] Dans une régression polynômiale, le coefficient d'ordre deux peut être considéré comme celui de l'interaction du régresseur avec lui-même.

Exemples :

- régression factorielle

```
X1 <- runif( 20, min=0, max=1)
X2 <- rnorm( 20, mean=2)
T <- X1 + X2 + X1 * X2 + rnorm( 20, sd=0.1)
vif( lm( T ~ X1 * X2))
```

X1	X2	X1:X2
5.010992	7.197542	10.588235

```
vif( lm( T ~ scale( X1, scale=FALSE) * scale( X2, scale=FALSE)))
```

scale(X1, scale=FALSE)	scale(X2, scale=FALSE)	interaction
1.003997	1.008478	1.005002

- régression polynômiale

```
QR <- X2 + 0.7 X2^2 + rnorm( 20, sd=0.3)
vif( lm( QR ~ X2 + I( X2^2)))
```

X2	I(X2^2)
9.920602	9.920602

```
vif( lm( QR ~ scale( X2, scale=FALSE) + I( scale( X2, scale=FALSE)^2)))
```

scale(X2, scale = FALSE)	I(scale(X2, scale = FALSE)^2)
1.043272	1.043272

Contrairement à ce qu'on pourrait croire, ce centrage facilite l'interprétation des résultats. En effet, si on suppose l'échantillon étudié comme représentatif de la population, alors aucun individu de la population (aux limites près de l'échantillonnage) ne peut avoir de valeur nulle pour le prédicteur. On raisonne alors à partir du barycentre, c'est-à-dire la tendance moyenne, de l'échantillon. Par exemple, dans la cadre d'une étude concernant des sujets atteints de la maladie d'Alzheimer, tous les sujets inclus doivent être âgés entre 60 et 90 ans. On voit bien qu'il n'est pas logique de calculer une ordonnée à l'âge '0'. Ce centrage, comme nous le verrons ultérieurement, est encore plus pertinent dans tous les cas pour les analyses de type Ancova.

Une des critique qui peut être faire au centrage des prédicteurs est que la moyenne dépend de l'échantillon concerné et est relativement sensible aux valeurs aberrantes et aux *outliers*. De ce fait, la détection de ces types de valeurs est nécessaire pour s'assurer de la pertinence de l'analyse ainsi centrée.

Comparaison des régresseurs

Les coefficients de régression dépendent des échelles de mesure. Par exemple, si on raisonne en centimètre, le coefficient sera cent fois supérieur à celui obtenu en raisonnant à partir des mètres, avec pourtant le même sens. Il est donc nécessaire de trouver une unité de comparaison pour comparer l'influence des régresseurs dans l'ajustement. Un certain nombre d'auteurs proposent alors de calculer les coefficients à partir des variables standardisées[9] (variable dépendante et régresseurs), c'est-à-dire de les centrer et de les diviser par leur écart type. Dans la littérature, ces coefficients sont généralement appelés les *betas*.

[9] encore appelées, variables centrées réduites.

Les coefficients des régresseurs standardisés sont facilement interprétable (variation de la variable dépendante en fonction d'une unité de dispersion du régresseur), même si les termes d'interaction de type polynômiale sont un peu plus délicats à interpréter.

Cette standardisation est facilement réalisable par la fonction scale.

A contrario, d'autres auteurs font valoir que cette standardisation dépend de l'échantillon et que de ce fait, la division par l'écart type peut menacer la robustesse de l'analyse. Ils proposent donc de comparer l'influence des différents régresseurs en utilisant le valeur de t de l'estimation. Celle-ci valant le rapport entre la valeur estimée et l'erreur type de l'estimateur, elle est donc moins sensible à l'échantillon, à condition que l'ajustement est de qualité (pas de points influents…) et elle est indépendante des décalage d'origine et des mises à l'échelle :

	Estimate	Std. Error	t value	Pr(>|t|)	
summary(lm(S ~ X2 + X1))					
(Intercept)	0.03278	0.07111	0.46	0.651	
X2	0.99184	0.01983	50.02	< 2e-16	***
X1	0.98983	0.09219	10.74	5.4e-09	***
summary(lm(S ~ scale(X2) + scale(X1)))					
(Intercept)	2.59879	0.02262	114.91	<2e-16	***
scale(X2)	1.16288	0.02325	50.02	<2e-16	***
scale(X1)	0.24961	0.02325	10.74	5.4e-09	***
summary(lm(scale(S) ~ scale(X2) + scale(X1)))					
(Intercept)	-9.173e-17	1.920e-02	0.00	1	
scale(X2)	0.9870	1.973e-02	50.02	< 2e-16	***
scale(X1)	0.2119	1.973e-02	10.74	5.4e-09	***

Tests post-hoc

Nous avons vu dans la paragraphe précédent comment effectuer des comparaisons planifiées, c'est-à-dire tester des hypothèses *a priori* sur le modèle. Les résultats de l'analyse peut conduire à tester *a posteriori* certaines relations.

Ces tests peuvent être conduit de deux manières différentes, soit par l'utilisation de la fonction linearHypothesis (package car) soit par celle de glht du package multcomp, le package lsmeans n'étant que peu d'intérêt dans le cas des régressions.

linearHypothesis

Cette fonction permet une spécification souple des tests à effectuer. Par exemple, on veut tester que l'ordonnée à l'origine vaut la valeur a. On écrira :

```
linearHypothesis( lm.resultat, "(Intercept) = a")
```

puisque le nom du coefficient correspondant à l'ordonnée à l'origine est "(Intercept)" (*cf.* la fonction coef).

Si par exemple, le modèle était :

```
Y ~ X1 + X2
```

et qu'on veut tester la pente de X1 à 2 et celle de X2 à 1, on écrira :

```
linearHypothesis( lm.resultat, "X1 = 2, X2 = 1")
```

Dans le cas de plusieurs comparaisons, on pourra :

1) vouloir corriger du fait que les variances des estimateurs ne sont pas identiques en passant à la fonction un estimateur de la matrice de variance/covariance ; il y a deux méthodes pour cela

i) passer la fonction d'estimation par l'argument vcov., c'est-à-dire poser vcov.=vcov,

ii) utiliser l'argument white.adjust, en lui donnant la valeur TRUE (identique à la solution précédente), soit une autre valeur, *cf.* la documentation de la procédure.

2) utiliser un test de type test de Wald à la place du test du F, ce qui est préférable pour les modèles linéaires généralisés, notamment les modèles logit ; on donnera alors comme valeur à l'argument test, la chaîne de caractères "Chisq".

La fonction linearHypothesis permet de tester n'importe quel paramètres ou ensemble de paramètres pourvu que leurs noms soient correctement codés (par exemple, pour le coefficients du carré du régresseur dans une régression polynomiale, son nom est "I(X^2)". On peut aussi utiliser la fonction matchCoefs pour récupérer tous les coefficients correspondant à un motif donné.

La spécification des tests peut aussi être faite sous la forme d'une matrice de contrastes des coefficients. Il faudra alors donner le vecteur des valeurs à tester (si elles sont différentes de zéro) par l'argument rhs.

La matrice de contrastes est simple à construire, ses colonnes correspondent aux coefficients estimées et ses lignes, aux combinaisons linéaires à tester. Il y a dons autant de lignes que de tests. Reprenons l'exemple ci-dessus en supposant que :

```
lm.resultat <- lm( Y ~ X1 * X2, data=donnees).
```

Quatre coefficients sont estimés, respectivement (Intercept), X1, X2 et X1:X2. La matrice de contraste a donc 4 colonnes et vaut :

```
hypothesis.matrix=matrix( c( 0, 1, 0, 0, 0, 0, 1, 0), ncol=4, byrow=TRUE)
```

tandis qu'on posera rhs=c(2, 1).

Si on avait voulu tester que le pente relative à X est deux fois celle de X2, on aurait posé :

```
hypothesis.matrix=matrix( c( 0, 2, -1, 0), ncol=4, byrow=TRUE), rhs=0
```

ce qui pourrait aussi s'écrire :

```
hypothesis.matrix="2 * X1 - X2 = 0"
```

Note : linearHypothesis compare un modèle restreint défini par les hypothèses testées au modèle courant. Cette procédure est donc proche des procédures de tests décrites dans la paragraphe précédent.

Dans le paragraphe précédent, j'ai présenté une méthode permettant de tester la valeur des coefficients de régression par rapport à certaines hypothèses sur ces valeurs en utilisant la construction offset à l'intérieur des formules.

L'exemple ci-dessous permet de comparer les résultats donnés par cette approche et celle utilisant linearHypothesis :

```
# Generation aleatoire des variables
nb.var <- 6
nb.sujets <- 50

sd.bruit <- 0.1
data.gen.colnames <- paste( "innov", 1:nb.var, sep=".")
data.gen <- matrix( rnorm( nb.var * nb.sujets), ncol=nb.var, nrow=nb.sujets,
                                                dimnames=list( NULL, data.gen.colnames))
bruit <- rnorm( nb.sujets, sd=sd.bruit)

# Generation des combinaisons lineaire de variables pour créer des regresseurs correles
var.names <- paste( "X", 1:nb.var, sep=".")
c.scale <- 5                    # plus c.scale est grand, plus les regresseurs seront correles
x.coefs <- matrix( rcauchy( nb.var * nb.var, scale=c.scale), ncol=nb.var,
                                                dimnames=list( data.gen.colnames, var.names))
diag( x.coefs) <- 1
x <- data.gen %*% x.coefs

#       Calcul de la variable dependante
lin.y.min <- 0.1
lin.y.max <- 2
y.coefs <- round( runif( nb.var, min=lin.y.min, max=lin.y.max), 3)
y <- x %*% y.coefs + bruit
d.f <- data.frame( Y=y, x)

#       Test modele sans contrainte
lm.y <- lm( Y ~ ., d.f)

#       Modele avec contraintes sur les valeurs souhaites des pentes
f.h <- formula( paste( "Y ~", paste( "X", 1:6, sep=".", collapse=" + "), " +
                        offset( ", paste( paste( y.coefs, "* X."), 1:6, sep="", collapse=" + "), ")",
                        sep=""))
lm.c <- lm( f.h, d.f)

# Comparaisons des resultats
#       en testant l'egalite des parametres estimes aux parametres theoriques
l.h <- paste( paste( "X", 1:6, sep="."), y.coefs, sep=" = ")
print( linearHypothesis( lm.y, l.h))
print( summary( lm.c))
```

Les résultats entre linearHypothesis et summary sont différents puisque linearHypothesis teste la conjonction des hypothèses et si le test est rejeté, cela signifie qu'au moins une des hypothèses est rejetée. La solution utilisant la construction offset dans la formule teste

chacune des hypothèses séparément comme le fait lm(Y ~ ., d.f). Cela permet alors de trouver les régresseurs qui ne répondent pas aux hypothèses.

Notons que :

for(h in l.h) print(linearHypothesis(lm.y, h))

donnera les mêmes résultats que ceux obtenus par summary.

multcomp

La procédure glht se programme à l'identique de la procédure linearHypothesis. L'argument décrivant les tests, linfct, aura comme valeur :

- soit par un vecteur de chaîne de caractères à l'exception de la chaîne "(Intercept)" qui doit 'quotée' dans la chaîne de caractères, c'est-à-dire entouré de backquote (ou apostrophes à l'envers) :

`(Intercept)`

- soit une matrice construite à l'identique de celle de linearHypothesis ; à ce moment, il faudra aussi donner une valeur à l'argument rhs.

Le résultat des test est donné par la fonction summary appliquée au résultat de glht.

Exemples

```
library( multcomp)
set.seed( 1)
x.1 <- runif( 30, min=1, max=5)
x.2 <- x.1 * sample( c( -1, 1), size=30, replace=TRUE)
# x.1 et x.2 ne sont pas correles
d.f <- data.frame( Y=1 + x.1 + x.2 + rnorm( 30, sd=0.1), X.1=x.1, X.2=x.2)
d.f$Y.1 <- d.f$Y - x.2

lm.fact <- lm( Y ~ X.1 + X.2, d.f)
lm.i <- lm( Y ~ I( X.1 + X.2), d.f)

# Statistique sur la difference des pentes
glht.fact <- glht( lm.fact, linfct=matrix( c( 0, 1, -1), ncol=3, nrow=1))
summary( glht.fact)
# Difference des pentes par comparaison de modèles
anova( lm.fact, lm.i)

lm.1 <- lm( Y.1 ~ X.1, d.f)
lm.off <- lm( Y.1 ~ offset( X.1), d.f)
lm.diff <- lm( Y.1 ~ offset( X.1) + X.1, d.f)

glht.1 <- glht( lm.1, linfct=matrix( c( 0, 1), ncol=2, nrow=1), rhs=1)
summary( glht.1)
anova( lm.1, lm.off)
summary( lm.diff)
```

On constate que les différentes approches donnent le même résultat quant à la significativité des différences entre pentes (exemple 1) ou de l'égalité de la pente à 1 (exemple 2).

lsmeans

La procédure centrale est lstrends. Cependant, dans le cas des régressions, ce package présente moins d'intérêt, ce qui ne sera pas le cas pour les (M)Ancova.

Intervalles de confiance

Les intervalles de confiance des coefficients seront obtenus par la fonction confint. Appliquée au résultat de l'analyse directement, elle donne les intervalles de confiance des coefficients individuellement, appliquée au résultat de glht, elle donne les intervalles corrigés des comparaisons multiples.

cld n'est pas utilisable car la notion de groupes homogènes n'a aucun sens puisque cette notion est conditionnelle à un critère exprimé par un facteur.

Résidus des modèles linéaires généralisés

Le modèle linéaire simple (fonction lm) est insuffisant dans certains cas, par exemple à cause de la distribution des données (modèle ricien…) ou de leur nature (comptage, étiquettes, facteur ordonné…) qui nécessite d'utiliser le modèle linéaire généralisé (glm ou vglm). Ce dernier ne considère plus que les données sont normales mais que leur distribution appartient au modèle exponentiel dont le modèle normal est un cas particulier. Les procédures demandent alors qu'on caractérise cette distribution grâce à un paramètre supplémentaire family. Les paramètres estimés ne sont plus les relations linéaires / affines qui relient les prédicteurs aux variables dépendantes, mais les paramètres qui lient les prédicteurs à la fonction associée à la famille à laquelle appartient les données.

Par exemple, si la variable dépendante un une variable binaire, on peut utiliser la régression logistique et la procédure estimera les paramètres β du logit c'est-à-dire de la transformations suivante :

$$\log\left(\frac{\Pr(Y=1)}{\Pr(Y=0)}\right)=\log\left(\frac{\Pr(Y=1)}{1-\Pr(Y=1)}\right)=\beta_0+\sum_i \beta_i X_i \ .$$

Dans ces conditions, les résidus 'classiques'[10] qui mesurent la différence entre la valeur estimée et la valeur réelle n'ont pas beaucoup de sens. C'est pourquoi, il existe d'autres types de résidus (argument type de la fonction residuals appliquée au résultat de ces analyses).

Il existe alors quatre types principaux de résidus :

- "working" : ce sont les résidus associés à la transformation, c'est-à-dire ceux associés à la relation linéaire estimée (membre residuals du résultat) ;

[10] Ces résidus sont ceux donnés par le membre appelé residuals de l'objet résultat de l'analyse.

- "response" : ce sont les résidus associés à la réponse analysée, par exemple dans l'exemple ci-dessus ce seront les résidus associés à la probabilité de chacun des niveaux ;

- "deviance" : ce sont les résidus signés associés à la valeur de la déviance (*cf. infra* pour la définition) pour le sujet donné, c'est-à-dire à la différence entre l'estimation courante et celle du modèle complet ;

- "pearson" : ce sont les résidus "working" normalisés par la déviation standard de l'estimation de la moyenne car dans les modèles linéaires généraux, cette déviation standard peut être fonction de la moyenne, sauf si les données sont gaussiennes.

Diagnostics des erreurs de modélisation[11]

Nous avons vu qu'il existait des méthodes basée sur une approche pas à pas (ascendante, descendante ou bidirectionnelle, *cf.* step et stepAIC) permettant de choisir le 'meilleur' modèle parmi une classe de modèles. Cependant ces méthodes ne garantissent pas que le modèle est bon, c'est-à-dire qu'il existe réellement une relation linéaire ou polynomiale entre un régresseur et la variable dépendante. Ce paragraphe traite de ce problème.

Analyse de la forme des résidus

La forme de la relation entre les résidus et les régresseurs permettent :

1) d'estimer l'homsocédasticité des résidus ;

2) le biais de l'estimation, c'est-à-dire l'erreur due au modèle, par exemple l'erreur due à une régression linéaire sur l'analyse d'une relation quadratique.

Cette analyse des résidus, essentiellement graphique, est l'une des premières choses à faire afin d'évaluer la qualité de l'analyse et avant d'exploiter les résultats.

Les résidus de l'ajustement sont donnés par la fonction residuals. Il existe aussi deux autres types de résidus obtenus par les fonctions suivantes.

- rstandard résidus corrigés du levier (*leverage*), c'est-à-dire du coefficient biaisant l'estimation de l'erreur (diagonale de la matrice chapeau [*hat matrix*], *cf.* hatvalues), et normalisés, c'est-à-dire ramenés à la variance unité. C'est donc l'erreur estimée sur la mesure contrairement à residuals qui est celle de l'ajustement.

- rstudent estimation de l'erreur normalisée quand on enlève ce point de l'ajustement. De manière informelle, le résultat peut être considéré comme l'erreur de prédiction pour ce point au sens de rstandard.

Dans tous les tests, on préférera des résidus normalisés (c'est-à-dire divisé par leur écart type) plus faciles à interpréter.

[11] Bien que présentée dans ce chapitre sur la régression, l'analyse des résidus est valable pour toutes les analyses linéaires, sauf mention explicite.

L'un des premiers tests a effectuer est de tester la normalité des résidus (notamment rstandard) par le test de Shapiro et Wilks (shapiro.test). Visuellement, on peut utiliser la fonction qqplot.

Résidus vs. variable dépendante

On trace la valeur du résidu en fonction de la celle de la variable dépendante :

```
fit.y <- fitted( lm.resultat)
o.y <- order( fit.y)
res <- residuals( lm.resultat) / sd( residuals( lm.resultat))
plot( res[o.y] ~ fit.y[o.y]
abline( h=0, lty=2)
```

Théoriquement, les résidus doivent être également répartis de part et d'autre de l'ordonnée nulle et de manière aléatoire en fonction de l'abscisse. De plus, ils doivent être concentrés autour de cette valeur nulle. Un point qui s'en écarte trop indique que l'ajustement en cette valeur n'est pas bon. Il convient alors de vérifier le poids de ce point dans l'ajustement par les indices présentés dans les chapitre précédents (*hat-value*, distance de Cook, …)

Si ce type de motif peut indiquer un problème éventuel, d'autres motifs sont beaucoup plus explicite :

- la distribution autour de l'abscisse nulle est asymétrique, ce qui fait suspecter que le modèle ne suit pas une loi normale ;

- les valeurs positives et négatives ne sont pas distribuées aléatoirement mais par plages en fonction de la valeur de la variable dépendante, ce qui indique que la relation est non linéaire ;

- on peut voir des blocs à l'intérieur des quels le motif de distribution est le même mais de motifs différents entre blocs, ce qui indique que la population n'est pas homogène et qu'il est nécessaire d'introduire de nouvelles variables indépendantes sous forme de facteur ; dans ce cas, l'analyse deviendra une Ancova ;

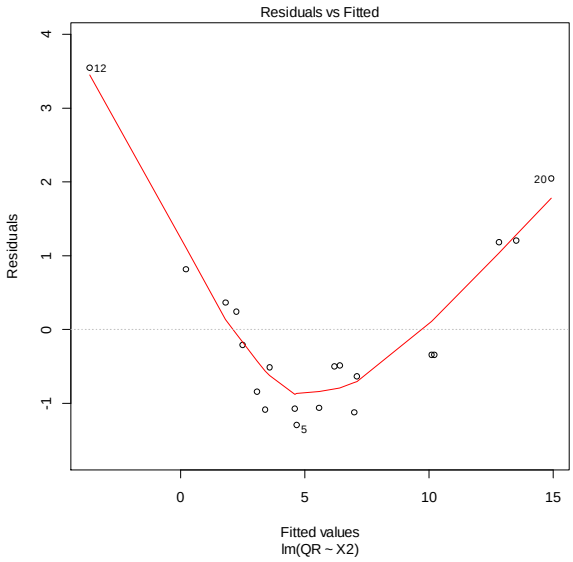
- la distribution des résidus varie en bande (ou non) non plus autour de l'abscisse nulle mais autour d'une forme régulière (droite, …), ce qui indique un biais d'estimation.

Exemple de détection d'une régression polynômiale

Reprenons la variable QR utilisé ci-dessus (*cf.* § considération pour améliorer l'interprétation des résultats). On aura :

```
summary( lm( QR ~X2))
```

qui donnera comme valeur au coefficient linéaire de X2, 3.9824 (t_{18}=16.06, p < 0.001) avec un R^2 *ajusté* valant 0.9311.



L'ajustement linéaire semble donc parfait. Pourtant, le tracé des résidus ci-contre

```
plot( lm( QR ~X2), which=1)
```

montre un motif typique d'un ajustement linéaire à la place d'un ajustement par un polynôme de degré 2.

L'estimation de l'ajustement polynômial donne alors les résultats suivant :

```
summary( lm( QR ~X2 + I(X2^2)))
```

```
Coefficients:
             Estimate   Std. Error  t value   Pr(>|t|)
(Intercept)  0.12223    0.20759     0.589     0.563746
X2           0.93676    0.19798     4.732     0.000193   ***
I(X2^2)      0.69886    0.04308     16.223    8.87e-12   ***

Residual standard error: 0.3212 on 17 degrees of freedom
Multiple R-squared:  0.996,    Adjusted R-squared:  0.9956
F-statistic:  2139 on 2 and 17 DF,  p-value: < 2.2e-16
```

Et la comparaison de modèles

```
anova( lm( QR ~X2),  lm( QR ~X2 + I(X2^2)))
```

donne

```
Analysis of Variance Table
Model 1: QR ~ X2
Model 2: QR ~ X2 + I(X2^2)
  Res.Df  RSS      Df  Sum of Sq  F       Pr(>F)
1 18      28.9132
2 17      1.7543   1   27.159     263.19  8.871e-12  ***
```

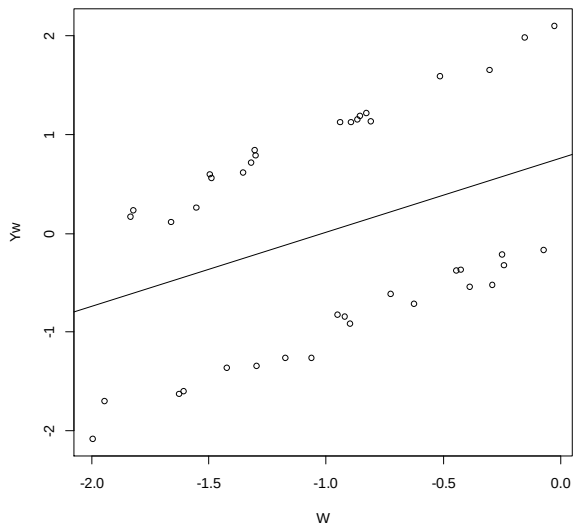
L'ajustement polynômial est donc préférable à l'ajustement linéaire.

Exemple de détection de plusieurs blocs :

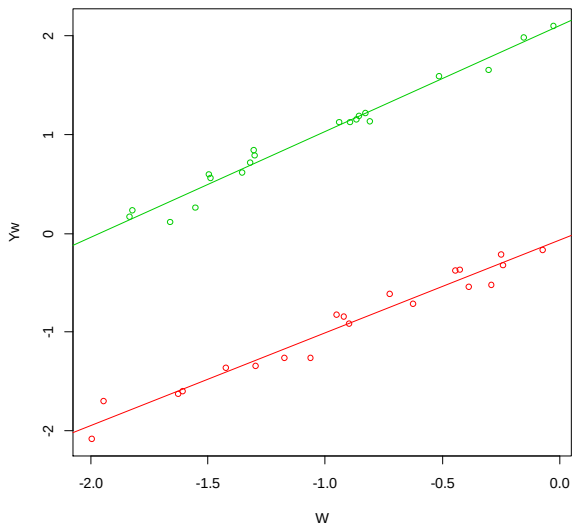
Soit les données générées par le script suivant :

```
W <- -runif( 40, min=0, max=2)
Yw <- W + rep( c( 0, 2), times=20) + rnorm( 40, sd=0.1)
```

```
GR <- rep( c( "G1", "G2"), times=20)
lm.0 <- lm( Yw ~ W)
lm.g <- lm( Yw ~ W * GR)
```

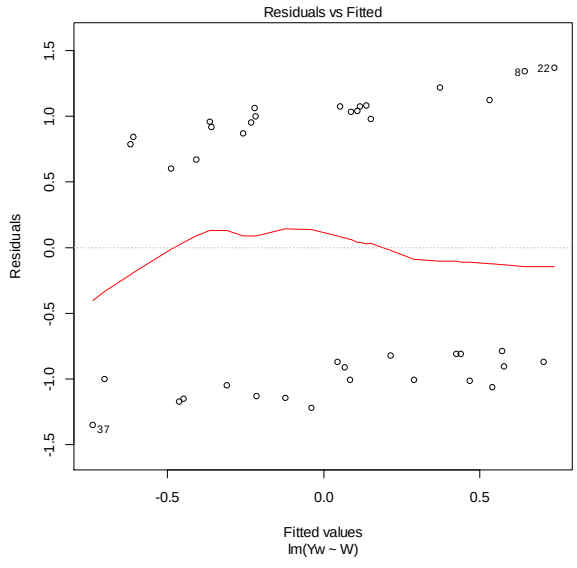



```
plot( Yw ~ W)
abline( lm.0)
```




```
plot( Yw[GR == "G1"] ~ W[GR == "G1"], col=2,
        xlim=range( W), ylim=range( Yw),
        xlab="W", ylab="Yw")
points( Yw[GR == "G2"] ~ W[GR == "G2"], col=3)
abline( lm( Yw[GR == "G1"] ~ W[GR == "G1"]),    col=2)
abline( lm( Yw[GR == "G2"] ~ W[GR == "G2"]),    col=3)
```

La figure de gauche ci-dessus semble indiquer qu'une régression est possible (même si dans le cas de l'exemple, la différence entre les deux groupes est très accentuée) alors que la figure de droite montre qu'il existe bien deux groupes avec des caractéristiques différentes ce que permet de suspecter le graphique suivant.



Le tracé des résidus en fonction de l'ajustement (figure ci-contre) :

```
plot( lm.0, which=1)
```

montre un motif incluant deux groupes de résidus caractéristique de cette situation.

Résidus vs. variable indépendante

Dans ces graphiques, on remplace la variable dépendante par chacune des variables indépendantes. En théorie, on doit obtenir un motif indiquant que les résidus doivent être également répartis de part et d'autre de l'ordonnée nulle et concentrés autour de cette valeur, de manière aléatoire en fonction de l'abscisse.

Si ce n'est pas le cas, le motif permet d'avoir une idée sur le problème :

- des résidus également répartis dans une bande autour d'une forme polynomiale fait suspecter la nécessité d'une régression polynomiale pour ce régresseur, le degré du polynôme étant indiqué par la forme, par exemple un U ou un U inversé indique sans doute la nécessité d'un ajustement quadratique ;

- la distributions des résidus autour de la forme (incluant la droite d'abscisse nulle) allant en s'élargissant ou s'amincissant en fonction de la valeur du régresseur indique que la variance est fonction de la variable et donc la nécessité d'une transformation de la variable dépendante pour stabiliser cette variance.

Résidus dans les modèles mixtes[12,13]

Contrairement aux modèles à effets fixes qui ne considère qu'une source de variabilité, l'erreur sur les mesures (ou variabilité inter-sujets), les modèles à effets fixes en incluent deux, la variabilité inter-sujets et la variabilité intra-sujet. De ce fait, il existe trois types de résidus générés par l'ajustement :

1) les résidus marginaux qui mesurent la différence entre la mesure et son estimation en termes d'effets fixes (Y - E[Y]) ; donc ils incluent l'effet aléatoire du sujet et la variabilité inter-sujets ;

2) les résidus conditionnels à la connaissance du sujet, c'est-à-dire l'erreur pure d'estimation de l'effet fixe (Y - E[Y | sujet]) ou, en d'autres termes, la variabilité inter-sujets ;

3) la variabilité intra-sujet qui prédit l'effet aléatoire (E[Y | sujet] - E[Y]) nommé en termes statistiques le meilleur prédicteur linéaire non biaisé (*Best Linear Unbiased Predictor* ou BLUP).

L'analyse des résidus étant une approche empirique, il ne semble pas utile de la poursuivre pour les niveaux inférieurs dans les modèles hiérarchiques. Par contre, dans les modèles comportant plusieurs effets aléatoires non corrélés, il faut analyser les résidus pour chacun des effets. Il y aura donc un ensemble de résidus marginaux (et de BLUP) par effet aléatoire.

Les deux packages proposent une fonction plot qui permet de tracer par défaut les résidus normalisés en fonction des valeurs ajustées (fitted). Voir les documentations pour les options.

---

[12] Nobre, JS et Singer, JM, (2007), Residual analysis for linear mixed models, Biometrical Journal, 49(6), pp863-875.
Singer, JM, Nobre, JS, Rocha, FMM, (2013), Diagnostic and treatment for linear mixed models, 59th ISI World Statistics Congress

[13] On trouvera à l'adresse http://www.ime.usp.br/~jmsinger/lmmdiagnostics.zip les codes R de deux fonctions écrites par Singer et col. réalisant les tracés mentionnés dans ce paragraphe pour les procédures lme et lmer. La fonction cookDiag écrite par ces mêmes auteurs permet de tester les résultats du modèle linéaire généralisé (adresse : http://www.ime.usp.br/~jmsinger/GLMMdiagnostics.zip)

L'appel à la fonction residuals diffère entre les packages nlme et lme4 puisque nlme n'autorise que les modèles hiérarchiques.

Dans nlme, l'argument level permet de définir les différents résidus (conditionnels pour level=0, puis marginaux pour chaque niveau de la hiérarchie). Par exemple, pour

lme.res <- lme(modele.fixe, random=~ 1 | F1/F2)

le niveau 0 est le résidu conditionnel, le niveau 1, le résidu marginal de F1 et le niveau 2, celui de F2 %in% F1 (ou F1:F2).

Dans lme4, residuals ne donne que les résidus conditionnels. Si on désire récupérer les résidus marginaux, il faut passer par la prédiction des valeurs en fonction de l'effet aléatoire étudié (fonction predict). Cette fonction predict possède un argument re.form qui définit l'effet aléatoire étudié. Ses valeurs possibles sont :

re.form=NULL : inclut tous les effets aléatoires

re.form=NA ou ~ 0 : effet fixe seul

re.form=~ 1 | F : prédiction marginale pour le facteur aléatoire F, c'est-à-dire, effet fixe + effet aléatoire de F ; s'il y a plusieurs effets aléatoires, on peut vouloir inclure leurs effets, la formule sera alors re.form=~ (form.alea1) + (form.alea2) + …

On retire alors aux valeurs mesurées donnees$Y les prédictions correspondants aux effets à tester. Par exemple :

res.cdn <- donnees$Y - predict(lme.res, re.form=NA)

correspond aux résidus conditionnels tandis que le résidus marginal au facteur aléatoire F se calculera par :

res.margF <- donnees$Y - predict(lme.res, re.form=~ 1 | F)

La variabilité intra-sujet pour ce facteur est alors la différence entre ces deux valeurs :

eplup <- res.margF - res.cdn

En appelant 'unité de regroupement', les niveaux du facteur aléatoire F, on peut alors créer différents tracés permettant des diagnostics visuels de certains problèmes (tableau adapté de Singer, JM, Nobre, JS et Rocha, FMM) :

- linéarité des effets fixes ; 2 types de tracés interprétables comme les tracés similaires pour les effets fixes,

  i) résidus marginaux vs. valeurs ajustées ;

  ii) résidus marginaux vs. variables indépendantes ;

- présence d'*outliers* ;

  i) résidus marginaux vs. unité de regroupement ;

  ii) résidus conditionnels vs. unité de regroupement ;

  iii) BLUP vs. numéro de l'unité de regroupement définissant l'effet aléatoire ;

- homoscédasticité de l'erreur conditionnelle ; résidus conditionnels vs. unité de regroupement ;

- normalité de l'erreur conditionnelle ; qqplot des résidus conditionnels ;

- normalité de l'effet aléatoire ; qqplot du BLUP.

Autres informations sur les effets aléatoires

Dans les deux packages nlme et lme4, la fonction ranef permet de récupérer la description des effets aléatoires alors que leurs caractéristiques en termes de covariance peut être obtenues par la fonction VarCorr.

- pour nlme, la fonction renvoie des chaines de caractères, une par effet, la dernière étant celle des résidus ; pour les manipuler dans un script, il faut donc les transcoder sous forme de valeurs numériques par la fonction as.numeric ; s'il y a plusieurs effets aléatoires, sa structure est plus complexe et les informations doivent être récupérées au cas par cas ;

- pour lme4, la fonction revoie un objet assez complexe qu'on peut transformer en data.frame par la fonction as.data.frame, l'ordre des lignes est le même que l'ordre des valeurs dans nlme ; ce data.frame comporte une colonne de nom "vcov" contenant les valeurs des covariance.

La fonction fixef qui estime les paramètres des effets fixes.

Ces deux packages proposent aussi la fonction plot qui permet de différents tracé. De plus, nlme, possède aussi une fonction qqnorm. Voir les pages d'aide de ces fonctions pour plus de détails.

Transformations de la variable

Quand l'analyse des résidus suggère la nécessité d'une transformation des données (notamment dans le cas de l'hétéroscédasticité des résidus), les tracés des résidus en fonction des régresseurs donnent des indications quant à la transformation à choisir. Ce paragraphe donne donc quelques règles simples pour choisir la relation de transformation des données quand les résidus font apparaître des structures (*cf.* paragraphe précédent).

Il est à noter qu'il y a un compromis à faire entre la stabilisation de la variance et le biais possible dans l'estimation induit par la transformation.

Transformation logarithmique

Si l'écart type est proportionnel à la moyenne ou si les effets sont multiplicatifs, on peut utiliser une transformation logarithmique. Elle est même recommandée dans le cas d'effets multiplicatifs à moins qu'on utilise des lois log-normales.

On peut détecter la proportionnalité de d'écart type à la moyenne soit en repérant une structure de type parabolique mais couché sur l'axe des x ($y = \sqrt{x}$) sur le graphe des résidus studentisés[14] en fonction des valeurs ajustées par le modèle et classées par ordre croissant, ou une droite sur un graphe des valeurs absolus des résidus.

[14] On utilise dans cette analyse les résidus studentisés et non les résidus centrés réduits pour éviter les biais que pourraient induire les corrélations entre la valeur (estimée ou expérimentale) en un point et le biais en ce point

On aura alors :

$$\text{var}(\log(x)) \approx \frac{1}{x^2}\text{var}(x)$$

Si le jeu de données comporte des petites valeurs (inférieures à 1), il est souvent préférable d'utiliser la transformation $\log(1+x)$ à la place de $\log(x)$.

Transformation par la racine carrée

Si la variance est proportionnelle à la moyenne, on obtient une droite de pente positive sur le graphe des résidus studentisés en fonction des valeurs ajustées par le modèle et classées par ordre croissant. On peut alors utiliser une transformation des données par la racine carrée. Si les valeurs sont petites (inférieurs à 10), il est préférable d'utiliser la transformation $\sqrt{y+0,5}$ plutôt que $\sqrt{y}$. On a alors :

$$\text{var}(\sqrt{x}) \approx \frac{1}{x}\text{var}(x)$$

Notons que si les variables sont des variables poissonniennes, on montre que la variable transformée $\sqrt{y+0,5}$ suit pratiquement une loi normale et dont la variance est indépendante de la moyenne contrairement à la loi de Poisson.

Transformation arc sinus

Si les données représentent des proportions, les valeurs sont comprises entre 0 et 1 et toutes les conditions d'utilisation des tests statistiques sur ces données brutes ne sont pas obligatoirement remplies. C'est aussi le cas s'il s'agit de pourcentage. Dans ce cas, on les ramène à des proportions.

Pour analyser ces données on peut les transformer en utilisant la fonction arc sinus, à condition que la relation entre variance et la moyenne soit quadratique et de la forme $s^2 \propto \mu(1-\mu)$.

Il existe plusieurs versions de cette transformation :

- $y_t = \arcsin\left(\sqrt{p}\right)$ d'où $\text{var}(y) \approx \frac{1}{p\sqrt{1-p}}\text{var}(p)$;
- $y_t = \arcsin\left(\sqrt{2p}\right)$;
- $y_t = \arcsin\left(\sqrt{(y+3/8/(N+3/4)}\right)$, si y et N sont petits.

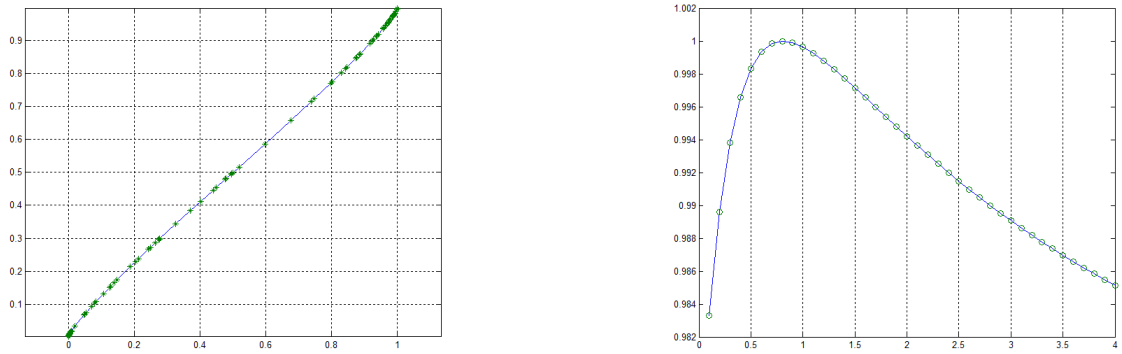
puisque c'est justement cela qu'on essaie de mettre à jour. De ce fait, l'intégrer dans l'analyse risque de la faire disparaître, notamment quand il s'agit d'un point influent.

Pour les deux premières formules, on calcule p comme suit :

- si $y = 0$, $p = 1/(4N)$, avec N la taille de la population de référence,
- si $1 \leq y < N$, $p = y / N$
- si $y = N$, $p = (N - 1/4) / N \; 1 - 0,01/4N$.

Il faut cependant remarquer que l'interprétation des résultats n'est pas simple et donc qu'il faut utiliser avec circonspection cette transformation. Elle n'est vraiment utile que quand l'approximation gaussienne n'est pas valable (existence de proportions faibles, etc.)

Autres transformations



À gauche la droite de Henry pour une variable provenant d'une distribution gaussienne à la puissance 1,25. À droite, le 'Box-Cox Normality Plot' ; le maximum de corrélation est environ 0,8, c'est-à-dire la transformation inverse de la transformation initiale.

La littérature propose d'autres transformations des données à partir de l'étude directe de la répartition des valeurs observées. Parmi les plus utilisées, on peut citer :

- la transformation de Box-Cox s'applique à la variable dépendante :

$$y_t = B(y, \lambda) = \begin{cases} \dfrac{y^{\lambda} - 1}{\lambda} & \lambda \neq 0 \\ \log(y) & \lambda = 0 \end{cases}$$; pour estimer le coefficient λ, on calcule le coefficient de corrélation entre les quantiles théoriques d'une distribution normale et les quantiles empiriques de la variable transformée (comme si on voulait tracer la droite de Henry) pour différentes valeurs de λ et on trace le graphe avec λ en abscisse et le coefficient de corrélation en ordonnée (*Box-Cox Normality Plot*) ; la valeur maximale (ou minimale si la corrélation est négative) de cette courbe donne le coefficient à appliquer (*cf.* figures ci-dessus) ;

- la transformation de Tukey s'applique à la variable dépendante :

$$y_t = \begin{cases} y^{\lambda} & \lambda > 0 \quad \text{réduction d'asymétrie négative} \\ \log(y) & \lambda = 0 \\ -y^{-\lambda} & \lambda < 0 \quad \text{réduction d'asymétrie positive} \end{cases} ;$$

le signe négatif pour $\lambda < 0$ restitue l'ordre des données.

Pour réaliser ces transformation, on peut utiliser les procédures powerTransform, bcPower, yjPower et basicPower du package car.

Défaut d'ajustement

Le défaut d'ajustement permet de tester si l'ajustement obtenu par la régression suit correctement les données. Pour cela, on vérifie que les valeurs estimées par régression sont proches des valeurs moyennes dans chaque classe obtenue en divisant le régresseur en N intervalles.

Le principe est de diviser la plage de variation du régresseur en N classes de même largeur. Ce nombre de classes doit être au minimum de 2 (tendance linéaire) mais ne doit pas être trop important. En effet, il faut que le nombre de points dans chacune des classes soit suffisant pour que la moyenne des valeurs de la variable dépendante ait en sens dans la classe et qu'on puisse assimiler le centre de la classe à la moyenne des valeurs des régresseurs. Par exemple, il faudra que

N < nclass.Sturges(donnees$X)[15]

La procédure de test peut ainsi ressembler à :

```
lm.r <- lm( Y ~X, data=donnees)
# Constructions des intervalles
x.range <- range( donnees$X)
```

Il faut prendre en compte les extrémités. cut crée des intervalles semi-ouverts à gauche ou à droite. Mon choix est de créer des intervalles semi-ouverts à droite. Le dernier intervalle doit donc contenir le point le plus à droite.

```
x.breaks <- seq( x.range[1],
                 ifelse( x.range[2] < 0, 0.98 * x.range[2], 1.05 * x.range[2]),
                 length.out=n.inter + 1)
x.factor <- cut( donnees$X, breaks=x.breaks, labels=paste( "X", 1:n.inter, sep="."),
                 include.lowest=TRUE, ordered_result=TRUE)
lm.f <- lm( Y ~ x.factor, data=donnees)
anova( lm.f, lm.r)
```

Si la différence entre les deux modèles n'est pas significative, alors le modèle de régression linéaire lm.r est correct.

[15] Calcul du nombre de classes pour un histogramme par la formule de Sturges. Il existe d'autres formules, *cf.* la page d'aide de la fonction.

Notes : i) le nombre de classes N doit être supérieur d'au moins une unité au degré du polynôme d'ajustement qu'on désire tester.

ii) en ordonnant le facteur et en s'assurant que le contraste utilisé pour estimer lm.f est le contraste polynomial (contr.poly), ce qui est généralement défini par défaut (*cf.* getOption("constrasts"), on réalise alors une analyse de tendance et summary(lm.f) donnera une idée du degré du meilleur polynôme d'ajustement. Cependant, ne pas oublier que le R^2 *ajusté* donne une indication sur la qualité de l'ajustement. Par exemple, si on obtient une tendance cubique avec un R^2 ajusté de l'ordre de 0,5 ou inférieur, on peut s'interroger sur la validité d'un ajustement par un polynôme de degré 3.

iii) si les intervalles sont de largeurs différentes, l'analyse de tendance avec les contrastes par défaut de contr.poly n'a pas de sens. Il faut calculer des contrastes polynomiaux spécifiques tenant compte du pas d'échantillonnage (*cf.* chapitres précédents).

Analyse des performances

Nous avons vu dans les chapitres précédents un certain nombre de concepts et de fonction permettant de diagnostiquer certains problèmes comme l'existence de points influents. Ces fonctions sont bien sûr applicables dans le cas de la régression.

Pour les résultats des fonctions standard lm et glm, elles sont regroupées sous le vocable de influence.measures dans la package de base stats. Pour les modèles à effets mixtes, le package influence.ME propose des fonctions similaires.

Diagnostic des corrélations[16]

Ce paragraphe présente quelques piste pour tester les corrélations entre variables (dépendantes et indépendantes) ou entre résidus en fonction des valeurs des variables des régresseur. À noter que les approches utilisant les résidus partiels et la régression partielle ne fonctionnent que s'il n'y a pas d'interactions entre les régresseurs.

Résidus partiels

Ce tracé des résidus partiels donnent une idée sur la relation entre l'une des variables indépendantes et la variable dépendante, corrélativement à l'existence des autres variables indépendantes dans le modèle :

```
# lm.modele est le resultat de l'analyse lineaire
# "v.i" est le nom du régresseur analyse
part.res["v.i"] <- residuals( lm.modele) + coef( "v.i") * data$v.i
plot( part.res["v.i"] ~ data$v.i)
```

Il convient cependant de se méfier de ce tracé qui a ses limites, notamment si la variable d'intérêt est fortement corrélée avec une des autres variables indépendantes. Dans ce cas, la variance des résidus partiels indiquée par le tracé est très inférieure à la variance réelle.

[16] Parmi les références intéressantes, une en français : Ricco Rakotomalala, Pratique de la régression multiple, http://eric.univ-lyon2.fr/~ricco/cours/cours/La_regression_dans_la_pratique.pdf

À un facteur de centrage près, cette opération est effectuée par la fonction crPlots du package car.

Régression partielle

Une autre méthode souvent utilisée est celle de la régression partielle. Cette technique évalue l'effet d'une variable additionnelle (la variable d'intérêt) à la régression effectuée sur les autres variables.

Cette opération s'effectue en trois étapes :

1) calcul des résidus sur le modèle composé de toutes les variables sauf la variable d'intérêt ;

2) calcul des résidus en régressant la variable d'intérêt sur toutes les autres variables ;

3) tracé des résidus du 1) en fonction de ceux du 2).

Ce tracé a de nombreuses propriétés dont :

1) l'ajustement linéaire par les moindres carrés est une droite passant par l'origine et de pente le coefficient de la variable dans le modèle de l'étape 1) ci-dessus ;

2) les résidus de cet ajustement sont identique à ceux du modèle de l'étape 1) ci-dessus ;

3) il est aisé de détecter l'influence de chacune des données sur l'estimation du coefficient de régression de la variable considérée ;

4) il est aisé de détecter plusieurs types de problèmes comme la non linéarité, l'hétéroscédasticité…

5) la corrélation entre les deux types de résidus correspondant à la corrélation partielle entre les deux variables dans le modèle incluant toutes les variables.

Cette opération est effectuée par la fonction avPlots du package car.

La régression partielle sert donc plus à identifier des points influents, qu'ils soient des points 'leviers' ou non tandis que les résidus partiels permettent mieux de se faire une idée de la relation entre une variable indépendante et la variable dépendante, moyennant l'existence des autres variables. Son intérêt est plus limité pour estimer la relation entre les deux variables puisque les abscisses ne sont pas la variable elle-même.

Régresseurs corrélés

Les régresseurs doivent être non corrélés entre eux sinon l'analyse n'est pas robuste : on peut dans ce cas avoir un modèle qui n'est pas significatif alors que certains paramètres le sont.

Pour tester l'existence possible de corrélations entre régresseurs, on utilise le facteur d'inflation de la variance (*VIF* pour *Variance Inflation Factor*) donné par la fonction vif du package car.

Le problème qui se pose quand on détecte des colinéarités entre régresseurs est de trouver quels régresseurs sont à conserver et lesquels peuvent être rejetés.

La technique pour détecter quels régresseurs sont colinéaires peut être fondée sur une approche appelée la régression croisée basée sur la régression de chacun des régresseurs sur les autres régresseurs, approche déjà mentionnée lors de la présentation des régressions partielles.

Partons de la matrice de corrélation entre régresseurs C_R et calculons son inverse :

$$C_R^{-1} = \begin{pmatrix} v_{11} & \cdots & v_{1k} & \cdots & v_{1K} \\ \vdots & \ddots & \vdots & & \vdots \\ v_{k1} & & v_{kk} & & v_{kK} \\ \vdots & & \vdots & \ddots & \vdots \\ v_{K1} & \cdots & v_{Kk} & \cdots & v_{KK} \end{pmatrix}.$$

La diagonale de son inverse donne les facteurs d'inflation de la variance :

$$VIF = (v_{11}, \cdots; v_{kk}, \cdots; v_{KK}).$$

On peut aussi montrer qu'on obtient à partir de cette inverse le coefficient de détermination des régressions partielles des régresseurs sur les autres régresseurs (ou coefficient R^2) :

$$R_k^2 = 1 - \frac{1}{v_{kk}}.$$

Sous l'hypothèse nulle que tous les coefficients sont nuls, donc que le régresseurs *k* n'est pas corrélés aux autres, et que les hypothèses de validité du modèle linéaire sont respectées, alors :

$F_k = \dfrac{R_k^2 / (K-1)}{(1-R_k^2) / (N-K)}$ suit une loi de Fisher à (*K-1, N-K*) degrés de liberté avec *N* le nombre de sujets.

On peut aussi utiliser les corrélations partielles puisque celles-ci mesurent la relation entre deux variables quand on l'a corrigée de l'influence de toutes les autres variables (influence directe) tandis que la corrélation standard de Pearson mesure aussi bien l'influence directe que les influences au travers des autres variables. Une forte diminution de la corrélation partielle par rapport à la corrélation de Pearson montre alors le caractère induit de la corrélation au travers d'autres variables et on ne conservera comme candidats de variables potentiellement corrélées que les variables de forte corrélation partielle.

Parmi les autres méthodes pour détecter les colinéarités, on peut aussi citer la méthode BKW[17] ou décomposition de la variance (*variance decomposition proportion*) basé sur l'analyse de la *model matrix*.

Quelque soit la méthode employée, celle-ci ne donne que des indications sur les corrélations entre régresseurs.

Résidus corrélés

Les techniques dont il a été précédemment question permettent de détecter des corrélations entre régresseurs. Il est aussi possible que l'erreur de mesure soit corrélée entre les valeurs du régresseur. Par exemple, supposons un processus accumulatif tel que la mesure m_n faite au temps t_n soit :

$$m_n = a\, m_{n-1} + b + \varepsilon_n$$

où a est un coefficient non nul,

b une certaine quantité correspondant à l'accumulation

et ε_n l'erreur de mesure au temps t_n.

Il est dans ce cas évident que les erreurs sont corrélées puisque on aura :

$$erreur(t_n) = \varepsilon_n + a\, \varepsilon_{n-1}.$$

Il existe un certain nombre de technique pour estimer la corrélation entre les résidus. Le plus connu est les test de Durbin-Watson qui est implanté sous R (durbinWatsonTest), voir la liste des tests dans les chapitres précédents. Son argument max.lag permet de tester les corrélations d'ordre 1 (lag=1), 2 (lag=2)…

Quand les résidus sont corrélés et que cela n'est pas pris en compte, les résultats sont biaisés. Une solution est d'utiliser une technique de transformation permettant de décorréler les erreurs (connu sous le nom de "blanchiment", *whitening* en anglais), mais cela relève alors de l'analyse des séries temporelles, hors du propos de ce document.

Une autre solution est d'utiliser une procédure permettant de spécifier la structure de la matrice de corrélation comme le fait la procédure gls du package nlme. Cette procédure possède un argument correlation qui permet de définir la forme de la corrélation. Par exemple, s'il existe une corrélation du premier ordre entre deux valeurs contigües du régresseur, on peut songer à utiliser comme descripteur de la corrélation, corAR1.

[17] Belsley, D. A., 1991, A guide to using collinearity diagnotics, Computer Science in Econimics and Management, 4, pp33-50

Régression logistique[18]

La régression logistique permet d'analyser la relation entre est une technique très couramment utilisée pour décrire la relation existant entre une variable à expliquer de type qualitatif et une ou plusieurs variables explicatives, quantitatives ou qualitatives[19].

Quand la variable à expliquer ne présente que deux modalités, on utilise la régression logistique binomiale. Si elle présente plus de deux modalités, le type de régression dépendra du fait que le facteur est ordonné ou non. Dans le premier cas, on utilise ce que la littérature nomme la régression multinomiale ou polychotomique ordinale ; dans le second, on utilise la régression multinomiale ou polychotomique nominale.

Notes :

1) Le package Zelig propose une interface pour ce type d'analyse qui permet d'avoir accès facilement à l'analyse des performances par courbes ROC (*cf. infra*) ;

2) Je laisse le soin au lecteur intéressé d'étendre mon propos aux modèles à effets mixtes.

Rappel sur les fonctions de lien

Ce type d'analyse conduit à estimer la probabilité d'occurrence d'un des événements référencés par les niveaux de la variable dépendante en fonction des variables explicatives et d'indiquer l'influence de ces dernières. Ces modèles sont des modèles linéaires généralisés. Leurs spécification en R passe par la spécification de la famille du modèle (argument family) qui sera alors ou binomiale (facteur à deux niveaux) ou multinomiale, et de la fonction de lien qui spécifie le type de relation entre les variables indépendantes et les probabilités d'occurrence des événements (argument link de la famille).

Je rappelle ci-dessous les principales fonctions de lien prédéfinies dans R (*cf.* la documentation de chacune des procédures d'analyse pour le codage exact) et leur signification :

- logit : c'est la fonction de lien la plus standard qui suppose que le comptage du nombre d'évènements suit une loi binomiale ou multionomiale ; elle est basée sur l'estimation du rapport de succès (*odd-ratio*) ou probabilité d'occurrence divisée par la probabilité de non occurrence ; le logarithme de ce rapport est antisymétrique par rapport à zéro (*i.e.*, l.rap(-x) = -l.rap(x)) ;

- probit : cette fonction de lien est basée sur la distribution normale et ses résultats sont très proches de ceux du logit ; le choix entre probit et logit est une question de préférences personnelles cependant la littérature suggère l'utilisation du logit quand il s'agit d'un véritable comptage et celle du probit quand la variable dépendante correspond à la discrétisation d'une variable sous-jacente de distribution normale ; il faut cependant souligner que les coefficients estimés ne sont pas directement interprétables contrairement au logit et que pour ce faire, il faut les transformer en *z-value* (ou utiliser directement des variables centrées réduites) ;

[18] Par extension, je mets sous ce vocable toute analyse dont la variable dépendante est un facteur ordonné ou non.

[19] Les développements ci-dessous, bien que dans le chapitre sur la régression c'est-à-dire supposant des prédicteurs quantitatifs, sont valable pour ces derniers aussi bien que pour des facteurs. Je ne reparlai donc pas de l'analyse logistique dans les autres chapitres.

- cauchy inverse : rarement utilisée sauf s'il y a de nombreuses valeurs extrêmes

- *complentary log log* : cette fonction de lien n'est pas symétrique; elle est utilisée pour des événements se produisant fréquemment ou rarement ou quand la réponse représente des regroupements de durée ou des temps de survie (probabilité de survie au-delà d'une certaine catégorie) ; elle conduit au modèle des risques proportionnels (*proportional hazard models*).

Régression binomiale

La fonction à utiliser est glm avec, au minimum, les arguments suivants :

formula : le modèle

data : les données

family=binomial : indiquant qu'il s'agit d'une analyse logistique ; binomial a un argument link qui indique quelle fonction de lien à appliquer ; cet argument vaut "logit" par défaut mais peut prendre d'autres valeurs dans l'ensemble suivant :

```
c( "logit", "probit", "cauchit", "log", "cloglog")
```

La formule peut prendre trois formes :

1) la variable dépendante est un facteur F à deux nivaux : F ~ … , le premier niveau du facteur est le niveau d'intérêt (ou le 'succès') ;

2) une variable binaire de valeurs 0 ou 1 (FALSE ou TRUE), avec 0 le niveau d'intérêt (ou le succès) ;

3) la variable dépendante est une matrice à deux colonnes mat[20] ; on aura alors :

```
# Chaque ligne est le nombre de succès et d'échecs pour le profil de valeurs des prédicteurs
        colnames( mat) <- c( "nb.succes", "nb.echecs")
        glm.data <- glm( cbind( nb.succes, nb.echecs) ~ …
```

ou

```
# Chaque ligne est le nombre de succès et le nombre de sujets pour le profil :
        colnames( mat) <- c( "nb.succes", "nb.sujets")
        glm.data <- glm( cbind( nb.succes, nb. Sujets - nb.succes) ~ …
```

Selon la valeur de l'argument type, la fonction predict permet de récupérer la valeur du lien ("link"), la probabilité du niveau d'intérêt ("reponse") ou les valeurs des coefficients ("terms"), soit pour les données analysées (pas d'argument newdata), soit pour de nouvelles données.

Le lien est négatif quand la probabilité est inférieure à 0,5, positive dans le cas contraire[21].

[20] Cette matrice représente donc les performances associées au profil composé par l'ensemble des valeurs des prédicteurs.

[21] On pourra faire un parallèle avec l'analyse linéaire discriminante (fonction lda du package MASS) pour constater que les deux approches sont assez semblables.

Pour aider à l'interprétation des résultats, reprenons l'exemple donné sur la page d'aide de predict.glm et tiré du livre de Venables and Ripley, en supprimant l'interaction :

```
ldose <- rep( 0:5, 2)
numdead <- c( 1, 4, 9, 13, 18, 20, 0, 2, 6, 10, 12, 16)
sex <- factor( rep(c( "M", "F"), c( 6, 6)))
SF <- cbind( numdead, numalive=20 - numdead)
budworm.lg <- glm( SF ~ sex + ldose, family=binomial,
                   contrasts=list( sex=contr.Treatment))
```

```
sex
[1] M M M M M M F F F F F F
Levels: F M
```

```
summary( budworm.lg)
```

```
Coefficients:
             Estimate  Std.-Error  z-value  Pr(>|z|)
(Intercept)  -3.4732   0.4685      -7.413   1.23e-13  ***
sex[T.2]      1.1007   0.3558       3.093   0.00198   **
ldose         1.0642   0.1311       8.119   4.70e-16  ***
```

En calculant le nombre la mortalité en fonction du sexe :

```
aggregate( SF ~ sex, FUN=sum)
```

ou de la dose

```
aggregate( SF ~ ldose, FUN=sum)
```

on constate que la mortalité est plus élevée chez les hommes que chez les femmes, c'est donc un facteur de risque positif comme l'indique le paramètres sex[T.2][22] ainsi que la dose (paramètres ldose).

[22] Voir dans le second chapitre, l'interprétation du contraste 'treatment'.

```
data.frame( SF, SEX=as.factor( sex), ldose,
                RESP=predict( budworm.lg, type="r"),
                LINK=predict( budworm.lg))
```

	numdead	numalive	SEX	ldose	RESP	LINK
1	1	19	M	0	0.08530076	-2.372412
2	4	16	M	1	0.21278854	-1.308198
3	9	11	M	2	0.43930479	-0.243984
4	13	7	M	3	0.69428515	0.820230
5	18	2	M	4	0.86812073	1.884444
6	20	0	M	5	0.95020002	2.948658
7	0	20	F	0	0.03008577	-3.473155
8	2	18	F	1	0.08249341	-2.408941
9	6	14	F	2	0.20673372	-1.344727
10	10	10	F	3	0.43032791	-0.280513
11	12	8	F	4	0.68647712	0.783701
12	16	4	F	5	0.86388206	1.847915

On peut étudier l'ajustement ou comparer deux ajustements en utilisant la fonction anova en donnant "Chisq" comme valeur à l'argument test. Cette comparaison peut aussi être faite en appelant directement la fonction pchisq :

```
pchisq( deviance( glm.1) - deviance( glm.2), df.residual( glm.1) - df.residual( glm.2))
```

On peut aussi tester l'effet d'un facteur ou certaines hypothèses de nullité de combinaisons de paramètres estimés en utilisant un test de Wald par la fonction wald.test du package aod.

Ce test possède deux arguments obligatoires :

Sigma : une matrice de variance/covariance,

b : un vecteur de coefficients de matrice de variance/covariance Sigma.

Dans notre cas, on codera :

```
wald.test( b=coef( glm.1), Sigma=vcov( glm.1)
```

Elle doit avoir un des deux arguments suivants :

Terms : vecteur d'index des coefficients devant être testés conjointement,

L : matrice des combinaisons linéaires de b à tester, ce qui signifie qu'on teste alors L %*% b.

Enfin, elle peut posséder un quatrième argument H0 qui contient les valeurs de la partie droite du test si on ne teste pas à zéro, c'est-à-dire que le test réel est L %*% b = H0 ou b[Terms] = 0.

Exemple :

Soit une analyse à un seul facteur F à trois niveaux et Classe le facteur de classement.

```
glm.1 <- glm( Classe ~ F, data=d.f, family=binomial, contrasts=list( F="contr.sdif"))
```

On teste que la différence entre le 1 et le second vaut 2 et entre le second et le troisième -2

```
wald.test( b=coef( glm.1), Sigma=vcov( glm.1), Terms=c( 2, 3), H0=c( 2, -2))
```

On teste que les coefficients dans chaque niveaux sont égaux à la grande moyenne

```
wald.test( b=coef( glm.1), Sigma=vcov( glm.1), L=contr.sdif( 3))
```

coef permet d'obtenir les coefficients et les *odd ratio* (rapport de gain) associés à ces coefficients sont obtenus par :

```
exp( coef( glm.1))
```

confint permet d'obtenir l'intervalle de confiance des coefficients estimés. Si on veut à la fois les *odd ratio* et leurs intervalles de confiance, on utilisera l'écriture suivante :

```
exp( cbind( OR=coef( glm.1), confint( glm.1)))
```

vglm permet d'analyser les données binaires soit :

- en utilisant l'approche valable pour les données multinomiales non ordonnées (*cf.* paragraphe suivant) ;

- en donnant comme famille de données, binomial : family=binomialff. Notons que dans ce cas, les valeurs doivent être 0 ou 1.

Si y est un vecteur de 0 et de 1, les quatre écritures suivantes donnent les mêmes résultats :

```
glm( y ~ …, family=binomial, …)
vglm( y ~ …, family=binomialff, …)
vglm( 1 - y ~ …, family=multinomial, …)
vglm( cbind( y, 1 - y) ~ …, family=multinomial, …)
```

Régression logistique binomiale et courbes ROC

Le logit binomial est équivalent à l'analyse discriminante linéaire. On peut étendre cette analogie à toutes les analyses binomiales. On peut alors conduire une analyse des performances en termes de sensibilité/spécificité (de l'ajustement ou de la prédiction par des méthodes de validation croisée ou *cross-validation*) en construisant le tableau de contingence entre données réelles et données prédites :

- en termes de comptage : table(donnees$Y, predict(lg.result, type="response"…
- en termes de proportions : prop.table(table(…

Cette approche est facilement extensible au cas de données multinomiales.

On peut aussi aller plus loin et vouloir analyser plus finement les résultats ou comparer deux modèles graphiquement[23]. On peut alors construire des courbes ROC. R propose de nombreux packages pour construire et analyser les courbes ROC dont les deux principaux sont pROC et ROCR.

Par exemple, avec pROC, on peut construire la courbe ROC de la manière suivante :

```
df.roc <- data.frame( Y=donnees$Y, L=predict( lg.result, type="link"…
lg.roc <- roc( Y ~ L, df.roc)
```

Régression logistique multinomiale non ordonnée

La variable dépendante est un facteur non ordonnée à plusieurs niveaux. Un des niveaux sert de référence.

Pour ce type d'analyse, on peut employer l'une des deux procédures suivantes :

- vglm du package VGAM, avec family=multionmial ;
- multinom du package nnet.

Les deux procédures acceptent comme membre de gauche soit un facteur à N niveaux ($N \geq 2$[24]), soit une matrice à N colonnes contenant le nombre de cas décomptés pour ce niveaux et pour un profil donné des variables indépendantes. Cependant, le niveau qui sert de référence diffère selon la procédure. Pour multinom, il s'agit du premier niveau du facteur ou de la première colonne de la matrice. Par défaut, vglm prend le dernier niveau du facteur ou la dernière colonne de la matrice.

Ainsi, en reprenant l'exemple du modèle logit multinomial de la page d'aide de vglm,

```
vglm( cbind( normal, mild, severe) ~ let, family=multinomial, data=pneumo)
```

et

```
multinom( cbind( severe, mild, normal) ~ let, pneumo)
```

donneront les mêmes coefficients.

vglm permet de modifier le facteur de référence en donnant la position du niveau de référence, ou le nom du niveau dans le cas où la variable dépendante est un facteur, comme valeur de l'argument refLevel de multinomial. Par exemple, si la première colonne est le niveau de référence comme pour multinom :

```
vglm( cbind( normal, mild, severe) ~ let, family=multinomial( refLevel=1), data=pneumo)
```

Note : summary ne donne pas de *p-value*. Cela est vrai aussi pour polr (*cf. infra*). On peut approximativement estimer cette valeur en testant la *z-value* (pour vglm), la *t-value* (pour polr) en la considérant comme une variable gaussienne.

```
pnorm( abs( va), lower.tail=FALSE) * 2
```

[23] La fonction anova(model1, model2, test="Chisq") n'est applicable que si les deux modèles utilisent la même fonction de lien et sont emboités.
[24] Ces deux procédures peuvent donc gérer le cas binomial.

Asymptotiquement, quand le nombre de degrés de liberté devient important, cela est vrai mais le résultat est d'autant plus biaisé (avec notamment inflation d'erreurs de type I) que le nombre de degré de liberté est faible :

Comme indiqué dans le paragraphe précédent, on peut aussi utiliser la fonction wald.test en donnant l'argument Terms ou la matrice L correspondante.

Regression logistique multinomiale ordonnée

Quand la réponse, ou variable dépendante, est un facteur ordonné, on peut vouloir analyser ses données de différentes manières. La première consiste à déterminer le rôle de chacune des variables indépendantes et, éventuellement, de leurs interactions, dans la définition de chacun des niveaux du facteur de classement. Dans ce cas, la stratégie d'analyse ne diffère pas de la celle pour la régression logistique multinomiale non ordonnée puisqu'on n'utilise pas le fait que le facteur est ordonné.

Il existe 3 classes principales de modèles pour l'analyse des facteurs ordonnés qui peuvent se décliner en différents sous-modèles selon les contraintes et la fonction de lien utilisés :

- le modèle des probabilités cumulatives (*cumulative categories model*) basé sur l'estimation de $P(Y \le i)$;

- le modèle des catégories adjacentes (*adjacent categories model*) basé sur l'estimation du rapport des probabilités entre deux catégories adjacentes $P(Y = i+1) \big/ P(Y = i)$;

- le modèle "d'analyse de séquences" comme par exemple les conditions de progression d'une pathologie, c'est-à-dire sur la probabilité d'arrêter la progression au niveau *i*, sachant que celui-ci est déjà atteint $P(Y = i | Y \ge i)$ (*stopping ratio model*) ou de continuer de progresser au-delà de ce niveau $P(Y > i | Y \ge i)$ (*continuation ratio model*).

Le choix du modèle dépendra de l'objectif recherché. Bien que relevant plus d'un cours de statistiques que de ce document, je donne ci-dessous les contextes d'utilisation des ces modèles et l'interprétation des coefficients avant de décrire comment les utiliser dans R.

La procédure polr du package MASS gère un sous type du modèle des probabilités cumulatives, celui des risques proportionnels (*proportional odds ratio*) tandis que vglm (package VGAM) propose les 3 classes. Les codages des appels et les tests annexes sont présentés ci-dessous.

J'ai signalé que dans vglm le modèle des données à analyser était décrit par l'argument family. Sa valeur est le nom d'une fonction qui possède des valeurs d'arguments par défaut, qui peuvent être modifiées par l'utilisateur. Ces fonctions possèdent leurs arguments propres mais un certain nombre d'arguments sont communs à toutes ces fonctions, dont les principaux sont :

link : nom de la fonction de lien,

parallel : valeur logique permettant de préciser si les paramètres des régresseurs sont identiques (TRUE) ou non (FALSE) dans toutes les classes ; si le parallélisme est vrai,

cela réduit le nombre de paramètres à estimer, mais cette contrainte doit être vérifiée avant toute utilisation,

reverse : modifie l'ordre de parcours des niveaux de la variable dépendante.

Modèle des probabilités cumulatives

Ce modèle est à utiliser de préférence quand l'objectif est d'étudier la valeur des différents prédicteurs en fonction du niveau *i* atteint, c'est-à-dire quand l'objectif est d'étudier la possibilité d'atteindre un niveau donné *i* en fonction de la valeur des prédicteurs.

L'analyse donne une ordonnée à l'origine α_i pour chacun des niveaux *i* du facteur qui est liée à la probabilité de tomber dans la catégorie *i* ou dans une catégorie inférieure quand les régresseurs sont nuls ou indépendamment des risques associés aux niveaux des prédicteurs de type facteur.

vglm	family=cumulative	$P(Y \leq i)$
	family=cumulative(reverse=true)	$P(Y \geq i)$

Par défaut, la fonction de lien est le logit, mais on peut aussi utiliser le probit, le cloglog ou cauchit.

proportional odds ratio

La spécification :

family=cumulative(reverse=TRUE, link="logit", parallel=TRUE) peut être codée plus efficacement par :

family=propodds

Ce modèle est connu sous le nom de celui des probabilités proportionnelles (*proportional odds ratio*). C'est aussi le modèle traité par la procédure polr (package MASS).

L'analyse donne alors une ordonnée à l'origine α_i pour chacun des niveaux *i* du facteur tandis que les paramètres associés aux prédicteurs sont les mêmes pour toutes les catégories. Cela signifie par exemple que s'il n'y a qu'un prédicteur, toutes les droites sont parallèles.

L'ordonnée à l'origine est la le logarithme du risque de tomber dans la catégorie *i* ou dans une catégorie inférieure quand les régresseurs sont nuls ou indépendamment des risques associés aux niveaux des prédicteurs de type facteur.

L'avantage de ce modèle par rapport au modèle général est qu'il est plus robuste puisqu'il y a moins de paramètres à estimer. Cependant, il est moins général. Il faut donc tester sa pertinence. Ce test est simple. Posons fit.gen le résultat du modèle général par vglm et fit.prop celui du modèle proportionnel, alors la significativité de la différence entre les deux ajustements est mesurée par[25] :

[25] La déviance est égale à moins deux fois le log vraisemblance du modèle testé par rapport au modèle complet. On test donc ici le logarithme rapport de vraisemblance entre les deux modèles que l'on sait avoir comme

```
pchisq( deviance( fit.prop) - deviance( fit.gen), df=df.residual( fit.prop) - df.residual( fit.gen),
        lower.tail=FALSE)
```

On obtient un résultat équivalent en utilisant la fonction lrtest.

proportional hazards model

Si la fonction de lien est cloglog, le modèle est aussi connus comme étant celui des risques proportionnels (*proportional hazards model*).

Modèle des catégories adjacentes

Ce modèle, implanté dans vglm, teste le rapport des probabilités entre deux catégories adjacentes :

	family
$P(Y=i+1)/P(Y=i)$	acat
$P(Y=i)/P(Y=i+1)$	acat(reverse=TRUE)

Ce modèle est donc surtout utile quand il s'agit de rechercher les variables explicatives qui permettent de prédire au mieux le fait d'appartenir à un niveau donné plutôt qu'au niveau suivant ou précédent.

On voit que le modèle multinomial est assez proche de ce modèle puisque dans le modèle multinomial on prend comme référence non pas la catégorie adjacente (inférieure ou supérieure) mais toujours la même catégorie.

Modèle "d'analyse de séquences"

J'ai regroupé deux modèles sous ce vocable :

1) le modèle "d'arrêt" (*stopping ratio model*) qui teste la probabilité d'interruption de la progression à un niveau *i* donné, sachant celui-ci déjà atteint[26] : $P(Y=i|Y\geq i)$ (familiy=sratio dans vglm) ;

2) le modèle de "prolongation" (*continuation ratio model*) qui teste la probabilité que cette progression continue au-delà du niveau déjà atteint : $P(Y>i|Y\geq i)$ (familiy=cratio dans vglm) ;

car il est facile de démontrer que $P(Y=i|Y\geq i)=1-P(Y>i|Y\geq i)$, ce qui signifie qu'en termes de logit, l'un et l'autre ont la même valeur absolue, mais des signes différents.

Bien entendu, on peut aussi raisonner selon un parcours descendant (reverse=TRUE).

---

distribution asymptotique un khi-deux dont le nombre de degrés de liberté est la différence des nombres de degrés de libertés de deux modèles.

[26] En utilisant comme fonction de lien, la fonction cloglog, on obtient un modèle des risques proportionnels proche de l'analyse de survie (*cf.* analyse de survie.)