Séminaire d'introduction à R

séance 4 - Modélisation

Joël Gombin

CURAPP - UPJV

24 février 2012

Plan de la séance

Plan de la séance

1. Introduction

2. La régression linéaire
   - Débuter avec la régression linéaire
   - Inclure des variables catégorielles
   - Inclure des interactions
   - Vérifier qu'un modèle est adéquat
   - Rendre compte des résultats d'un modèle

Plan de la séance

Plan

Introduction

Cette séance, comme la suivante, concerne les cas où l'on cherche à rendre compte des relations entre plusieurs variables. Contrairement à l'analyse géométrique des données, les méthodes évoquées ici sont asymétriques : il y a une variable dépendante et des variables indépendantes.
On parle alors de modélisation : on cherche à modéliser une variable en fonction d'autres. Cette modélisation a souvent une ambition causale et/ou prédictive. Elle prend la forme d'une relation numérique.

Plan

Débuter avec la régression linéaire

Le modèle linéaire simple est extrêmement facile à utiliser dans `R`. On utilise la fonction `lm()`, comme "linear model".

```
# charger mini_picardie
modele <- lm(AbsIns ~ proprietaire, data = mini_picardie)
summary(modele)


##
## Call:
## lm(formula = AbsIns ~ proprietaire, data = mini_picardie)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -16.58  -2.47   0.03   2.45  86.53
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  18.65769    0.69566   26.82  < 2e-16 ***
## proprietaire -0.06917    0.00879   -7.87  5.3e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.41 on 2288 degrees of freedom
## Multiple R-squared: 0.0264, Adjusted R-squared: 0.026
## F-statistic:   62 on 1 and 2288 DF,  p-value: 5.29e-15
##
```

Débuter avec la régression linéaire

(Les résultats du modèle peuvent aussi assez facilement se représenter comme un tableau - par exemple via la fonction `genere.tableau` du package `rgrs`)

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	18.6577	0.6957	26.82	0.0000
proprietaire	-0.0692	0.0088	-7.87	0.0000

On peut accéder aux différentes composantes du résultat :

```
coef(modele)
```

```
##  (Intercept) proprietaire
##     18.65769     -0.06917
```

```
summary(modele)$r.squared
```

```
## [1] 0.02638
```

```
summary(modele$fitted.values)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    11.7    12.7    13.1    13.2    13.6    17.2
```

Inclure des variables catégorielles

On inclut une variable catégorielle (de type `factor`) comme n'importe quelle autre variable :

```
modele2 <- lm(AbsIns ~ proprietaire + type_urbain,
    data = mini_picardie)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	17.7018	0.7101	24.93	0.0000
proprietaire	-0.0599	0.0090	-6.67	0.0000
type_urbainCommune monopolarisée	0.0655	0.2107	0.31	0.7559
type_urbainCommune multipolarisée	0.4810	0.2602	1.85	0.0646
type_urbainPole urbain	2.6095	0.4562	5.72	0.0000

Inclure des interactions

Parfois, on souhaite savoir comment des variables indépendantes interagissent entre elles. On utilise pour cela le signe : (ou le raccourci *, qui teste toutes les interactions possibles entre les variables).

```
modele3 <- lm(AbsIns ~ proprietaire * type_urbain,
    data = mini_picardie)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	17.9369	1.1526	15.56	0.0000
proprietaire	-0.0630	0.0148	-4.26	0.0000
type_urbainCommune monopolarisée	-2.5229	1.7559	-1.44	0.1509
type_urbainCommune multipolarisée	-0.9471	2.0178	-0.47	0.6388
type_urbainPole urbain	7.4167	2.0504	3.62	0.0003
proprietaire :type_urbainCommune monopolarisée	0.0321	0.0220	1.46	0.1442
proprietaire :type_urbainCommune multipolarisée	0.0183	0.0257	0.71	0.4761
proprietaire :type_urbainPole urbain	-0.0692	0.0278	-2.49	0.0128

Vérifier qu'un modèle est adéquat

La modélisation linéaire repose sur un certain nombre d'hypothèses sur les données (normalité, indépendance, linéarité, homoscédasticité). Pour vérifier que ces hypothèses sont vérifiées, on recourt à une inspection visuelle.
Sans rentrer ici dans le détail de ces inspections, on pourra recourir aux fonctions :

```
plot(modele)

library(car)
qqplot(modele)
crPlots(modele)
```

Rendre compte numériquement des résultats d'un modèle

Les coefficients d'un modèle n'ont pas en soi beaucoup d'intérêt (sauf pour les variables catégorielles). Il faut donc essayer essayer de les traduire en quantités intéressantes.
Par exemple, comment varie l'abstention lorsque on a peu de propriétaires ou beaucoup de propriétaires ?
Pour cela, on estime la valeur de l'abstention en fonction de la valeur que prennent les variables dépendantes (voir aussi le package `Zelig` de Gary King).

```
a <- quantile(mini_picardie$proprietaire, 0.2)
b <- quantile(mini_picardie$proprietaire, 0.8)
new.data <- data.frame(proprietaire = c(a, b))
predict(modele, new.data, se.fit = T)


## $fit
##   20%   80%
## 13.70 12.67
##
## $se.fit
##    20%    80%
## 0.1100 0.1164
##
## $df
## [1] 2288
##
## $residual.scale
## [1] 4.41
```

Les coefficients normalisés

Une manière de représenter les résultats d'un modèle consiste à estimer un modèle sur les variables dépendantes standardisées. Cela permet de comparer les coefficients les uns aux autres. On appelle cela des coefficients β.

```
modele4 <- lm(AbsIns ~ proprietaire + ouvriers,
    data = mini_picardie)
```

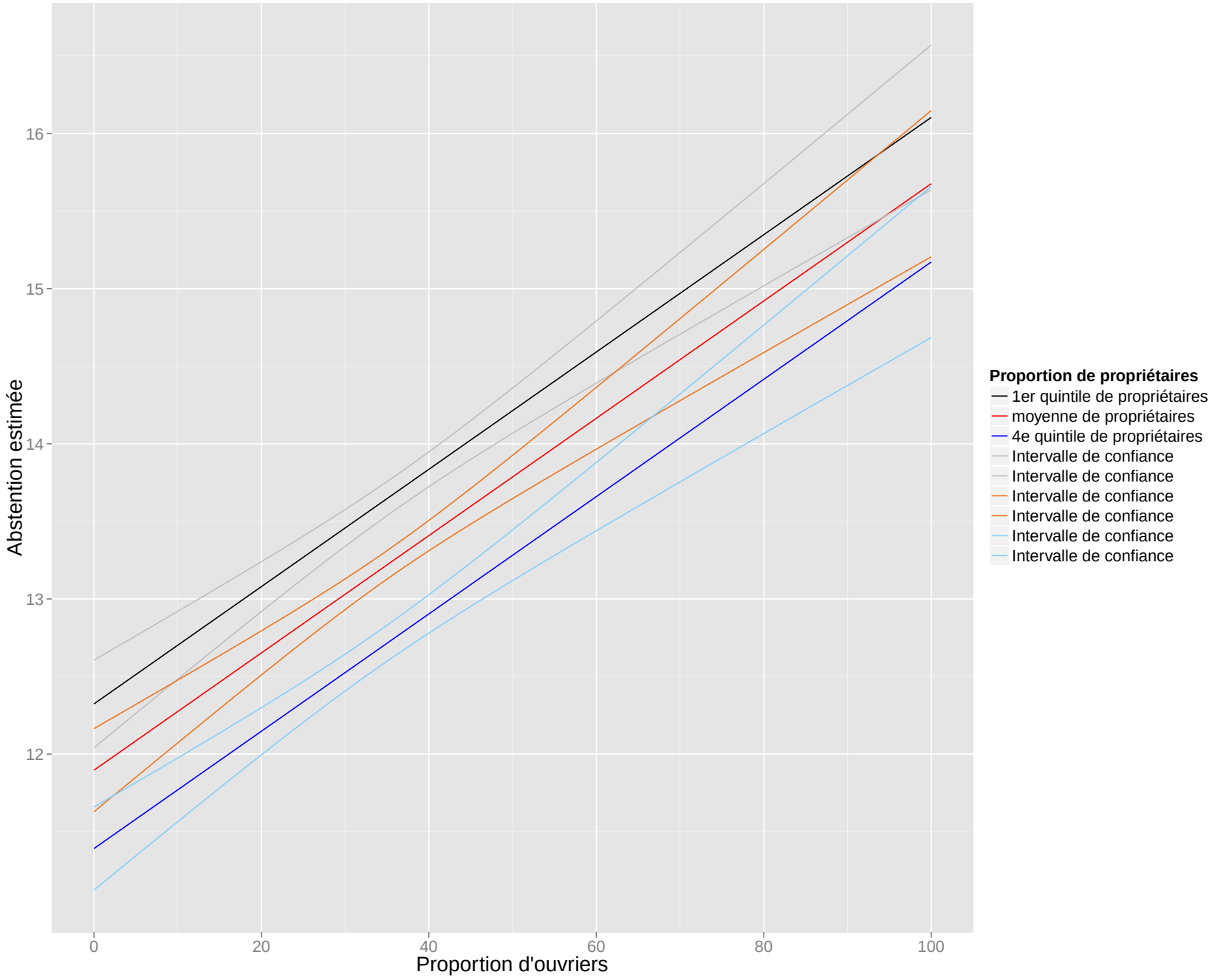
	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	16.7939	0.7761	21.64	0.0000
proprietaire	-0.0624	0.0088	-7.07	0.0000
ouvriers	0.0378	0.0071	5.29	0.0000

```
library(QuantPsyc)
lm.beta(modele4)


## proprietaire     ouvriers
##      -0.1465       0.1097
```

Rendre compte graphiquement des résultats d'un modèle

```
new.ouvriers <- seq(0, 100, 1)
new.proprietaire1 <- rep(quantile(mini_picardie$proprietaire,
    0.2), length(new.ouvriers))
new.proprietaire2 <- rep(mean(mini_picardie$proprietaire),
    length(new.ouvriers))
new.proprietaire3 <- rep(quantile(mini_picardie$proprietaire,
    0.8), length(new.ouvriers))
prev1 <- predict(modele4, newdata = data.frame(ouvriers = new.ouvriers,
    proprietaire = new.proprietaire1), se.fit = T)
prev2 <- predict(modele4, newdata = data.frame(ouvriers = new.ouvriers,
    proprietaire = new.proprietaire2), se.fit = T)
prev3 <- predict(modele4, newdata = data.frame(ouvriers = new.ouvriers,
    proprietaire = new.proprietaire3), se.fit = T)
data <- data.frame(new.ouvriers, new.proprietaire1,
    new.proprietaire2, new.proprietaire3, prev1 = prev1$fit,
    prev2 = prev2$fit, prev3 = prev3$fit, prev1a = prev1$fit -
        prev1$se.fit, prev1b = prev1$fit + prev1$se.fit,
    prev2a = prev2$fit - prev2$se.fit, prev2b = prev2$fit +
        prev2$se.fit, prev3a = prev3$fit - prev3$se.fit,
    prev3b = prev3$fit + prev3$se.fit)
library(ggplot2, scales)
data2 <- melt(data, id = c("new.ouvriers", "new.proprietaire1",
    "new.proprietaire2", "new.proprietaire3"))
graph <- ggplot(data2, aes(new.ouvriers, value,
    group = variable), ) + geom_line(aes(colour = variable)) +
    scale_colour_manual("Proportion de propriétaires", values = c(alpha(c("black",
        "red", "blue"), 1), c("grey", "grey", colors()[54],
        colors()[54], colors()[430], colors()[430])), labels = c("1er quintile de propriétaires",
        "moyenne de propriétaires", "4e quintile de propriétaires",
        rep("Intervalle de confiance ", 6))) + scale_x_continuous("Proportion d'ouvriers") +
    scale_y_continuous("Abstention estimée")
```


Abstention estimée
16
15
14
13
12
0
20
40
60
80
100
Proportion d'ouvriers
Proportion de propriétaires
1er quintile de propriétaires
moyenne de propriétaires
4e quintile de propriétaires
Intervalle de confiance
Intervalle de confiance
Intervalle de confiance
Intervalle de confiance
Intervalle de confiance
Intervalle de confiance

Plan

Le modèle linéaire généralisé

Certaines relations sont difficiles ou impossibles à modéliser directement par une relation linéaire. Par exemple, lorsque la variable dépendante est une variable dichotomique ou polytomique.
Dans ce cas, on recours au modèle linéaire généralisé, dont la régression logistique est un cas particulier.
La syntaxe est très proche, avec la fonction `glm`. Il faut juste préciser la manière dont on modélise la relation entre les variables.
On va prendre comme exemple des données de la troisième vague du Baromètre politique français (BPF) de 2007.

Spécifier le modèle

```
#charger le fichier BPFv3.Rdata
attributes(BPFv3)$variable.labels[c("Q3", "SEXE",
    "RS2B")]
```

```
##                          Q3                        SEXE
## "Intérêt pour la politique"                    "Sexe"
##                        RS2B
##         "Niveau de diplôme"
```

```
BPFv3$PolInt <- BPFv3$Q3
levels(BPFv3$PolInt) <- c("Oui", "Oui", "Non",
    "Non", "Non")
BPFv3$PolInt <- relevel(BPFv3$PolInt, ref = "Non")
logit1 <- glm(PolInt ~ SEXE + RS2B, data = BPFv3,
    family = binomial(link = "logit"))
```

Résultats

```
summary(logit1)
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.1642	0.0874	-1.88	0.0603
SEXEFemme	-0.7209	0.0584	-12.35	0.0000
RS2BCertificat d'Etudes Primaires	0.4590	0.1094	4.20	0.0000
RS2BAncien brevet, BEPC	0.6763	0.1207	5.60	0.0000
RS2BCertificat d'Aptitude Professionnelle (CAP)	0.1304	0.1072	1.22	0.2239
RS2BBrevet d'Enseignement Professionnel (BEP)	0.1440	0.1213	1.19	0.2352
RS2BBAC d'enseignement technique ou professionnel	0.4652	0.1322	3.52	0.0004
RS2BBAC d'enseignement général	0.8222	0.1298	6.33	0.0000
RS2BBAC+2 ou niveau BAC+2 ans (DUT, BTS, Instituteurs, DEUG...)	0.7823	0.1240	6.31	0.0000
RS2BDiplôme de l'enseignement supérieur (2ème ou 3ème cycles, gr	1.5756	0.1278	12.33	0.0000

```
predict(logit1, newdata = data.frame(SEXE = c("Homme",
    "Femme", "Homme", "Femme"), RS2B = c("Sans diplôme",
    "Diplôme de l'enseignement supérieur (2ème ou 3ème cycles, gr",
    "Diplôme de l'enseignement supérieur (2ème ou 3ème cycles, gr",
    "Sans diplôme")), type = "response")


##      1      2      3      4
## 0.4590 0.6661 0.8040 0.2921
```

Modèle linéaire généralisé

Pour le reste, explorer... On utilise globalement les mêmes outils qu'avec `lm`.
Beaucoup de ressources en ligne !