République Tunisienne

Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

Université de Tunis El Manar

Faculté des Sciences de Tunis

Rapport du projet académique

Réalisé par

Azouni Louai , Fathallah Rayen , Ayadi Aziz , Jouini Wajih

Régression : Linéaire , Logistique, de Poisson

Encadrant Académique : **Salah Khardani** Professeur

2023-2024

Table des matières

Chapitre 1

Regression Lineaire Simple

Plan

1.1 Description des données

Chaque ligne dans le jeu de données représente une observation individuelle, avec une paire de valeurs (salaire, années d'expérience). Ces données sont collectées dans le but d'explorer la relation entre le salaire et l'expérience professionnelle, et éventuellement de construire un modèle de régression linéaire simple pour prédire les salaires en fonction des années d'expérience.

```
# Charger les donn es
data <- read.csv("Salary_Data.csv", header=TRUE)
# Extraire les variables
sal <- data$Salary # Y
ann_exp <- data$YearsExperience # X1
n <- length(sal)   # Nombre d'observations
head(data)
```

	YearsExperience	Salary
1	1.1	39343
2	1.3	46205
3	1.5	37731
4	2.0	43525
5	2.2	39891
6	2.9	56642

Il existe une seul variable prédictive : **YearsExperience** .

YearsExperience : la variable prédictive numérique et représente le nombre des années d'experience pour chaque individu.

Salary : la variable à prédire numérique représente le salaire d'un individu.

```
summary(data)
```

summary : pour obtenir des descriptions de base pour l'ensemble des données.

```
> summary(data)
 YearsExperience      Salary
 Min.   : 1.100   Min.   : 37731
 1st Qu.: 3.200   1st Qu.: 56721
 Median : 4.700   Median : 65237
 Mean   : 5.313   Mean   : 76003
 3rd Qu.: 7.700   3rd Qu.:100545
 Max.   :10.500   Max.   :122391
```

1.2 Approche Mathematique

L'approche mathématique de la régression linéaire simple implique de trouver la meilleure droite de régression qui minimise la somme des carrés des erreurs entre les valeurs observées (le salaire dans ce cas) et les valeurs prédites par le modèle à traver des relations mathématiques.

1.2.1 Les formules utilisés

$$Y = \beta_0 + \beta_1 \cdot X + \varepsilon$$

$$\beta_1 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

$$\beta_0 = \bar{Y} - \beta_1 \cdot \bar{X}$$

$$Y = \beta_0 + \beta_1 \cdot X + \varepsilon$$

$$\beta_1 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

$$\beta_0 = \bar{Y} - \beta_1 \cdot \bar{X}$$

1.2.2 Etape1 : Calcule des sommes

$$\text{sum\_x} = \sum_{i=1}^{n} \text{ann\_exp}_i$$

$$\text{sum\_y} = \sum_{i=1}^{n} \text{sal}_i$$

$$\text{sum\_x\_squared} = \sum_{i=1}^{n} (\text{ann\_exp}_i)^2$$

$$\text{sum\_xy} = \sum_{i=1}^{n} \text{ann\_exp}_i \cdot \text{sal}_i$$

```r
# R gression lin aire simple - Approche classique avec des formules math mati
# Calculer les sommes n cessaires
sum_x <- sum(ann_exp)
sum_y <- sum(sal)
sum_x_squared <- sum(ann_exp^2)
sum_xy <- sum(ann_exp * sal)
```

1.2.3 Etape2 : Calcule des coefficient

$$\text{slope} = \frac{n \cdot \text{sum\_xy} - \text{sum\_x} \cdot \text{sum\_y}}{n \cdot \text{sum\_x\_squared} - (\text{sum\_x})^2}$$

$$\text{intercept} = \frac{\text{sum\_y} - \text{slope} \cdot \text{sum\_x}}{n}$$

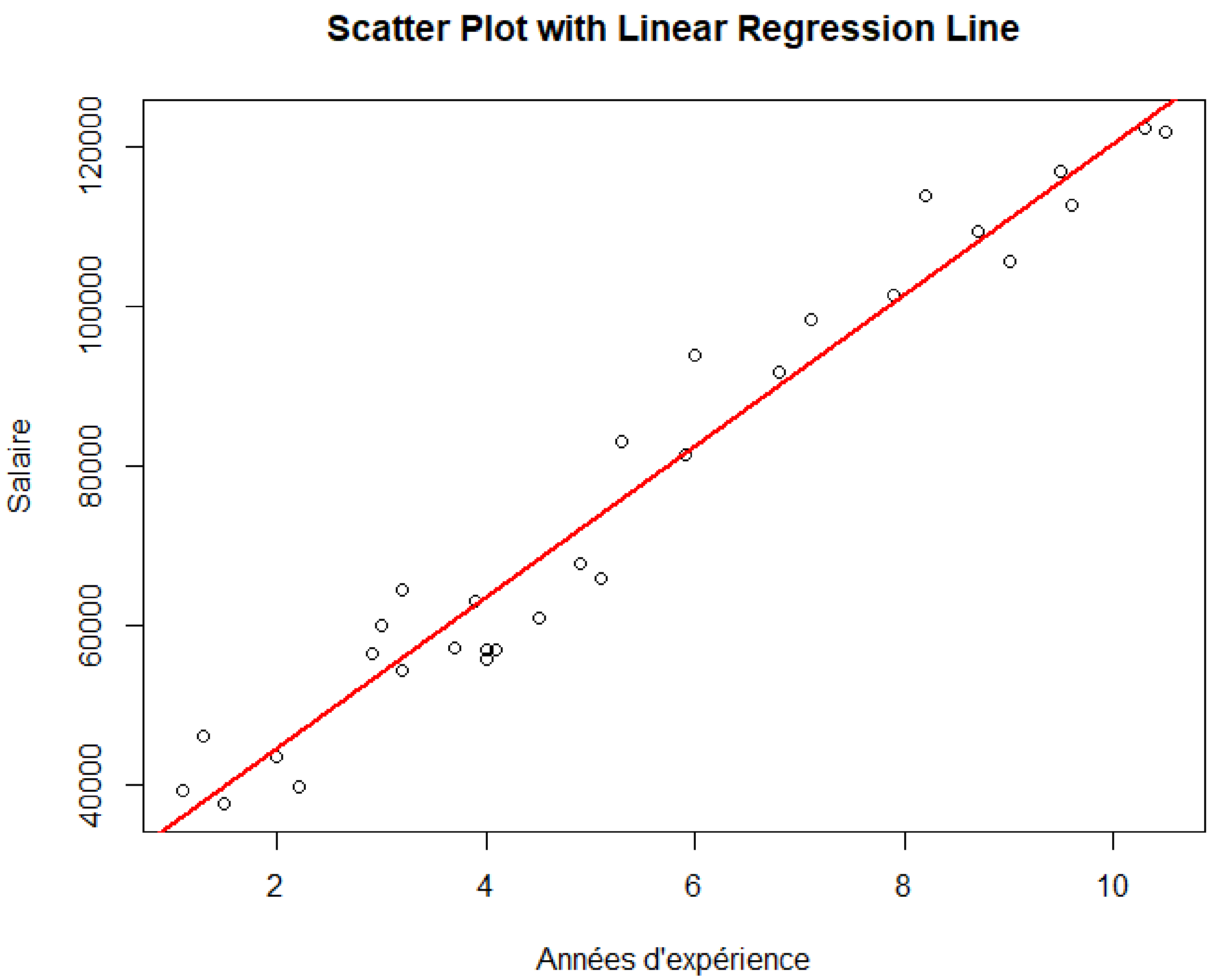
$$sal = \text{intercept} + \text{slope} \cdot ann_exp$$

```r
# Calculer les coefficients
slope <- (n * sum_xy - sum_x * sum_y) / (n * sum_x_squared - sum_x^2)
intercept <- (sum_y - slope * sum_x) / n

# Afficher l' quation
cat("Approche␣classique:\n")
cat("Y␣=", round(intercept, 2), "+", round(slope, 2), "*␣X\n\n")
```

```
Y = 25792.2 + 9449.96 * X
```

1.2.4 Etape3 : Visualisation des résultats

```
plot(ann_exp, sal, main="Scatter␣Plot␣with␣Linear␣Regression␣Line",
    xlab="Ann es␣d'exp rience", ylab="Salaire")
abline(intercept, slope, col="red", lwd=2)
```



1.3 Approche par modèle , La fonction lm()

1.3.1 Etape1 : Utilisation de lm()

La fonction lm() en R est utilisée pour ajuster des modèles linéaires. Elle fait partie du système statistique de R et est largement utilisée pour effectuer des analyses de régression.

```
# R gression lin aire simple - Utilisation de la fonction lm
lm_model <- lm(sal ~ ann_exp, data=data)
```

1.3.2 Etape2 : Utilisation de summary

La fonction summary() appliquée à un modèle linéaire (lm) en R est utilisée pour obtenir un résumé statistique détaillé du modèle linéaire ajusté à l'aide de la fonction lm().

```
# Afficher le r sum  de lm
summary(lm_model)
```

```
Call:
lm(formula = sal ~ ann_exp, data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-7958.0 -4088.5  -459.9  3372.6 11448.0 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  25792.2     2273.1   11.35 5.51e-12 ***
ann_exp       9450.0      378.8   24.95  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5788 on 28 degrees of freedom
Multiple R-squared:  0.957,	Adjusted R-squared:  0.9554 
F-statistic: 622.5 on 1 and 28 DF,  p-value: < 2.2e-16
```

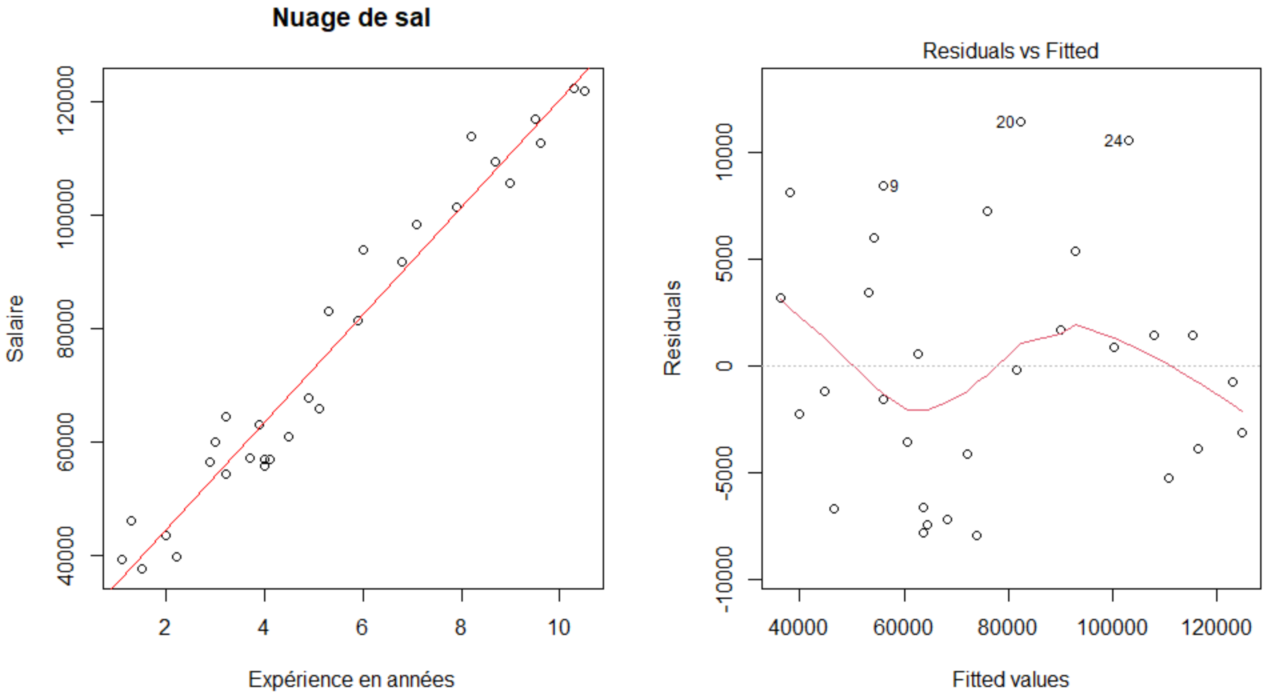
1.3.3 Etape3 : Visualisation des résultats

Dans cette étape, nous visualisons les résultats de la régression linéaire en utilisant deux graphiques. Le premier graphique est un nuage de points qui illustre la relation entre les années d'expérience (sur l'axe des x) et les salaires (sur l'axe des y). Une ligne de régression est ajoutée pour représenter la tendance générale des données.

Le deuxième graphique est un graphique des résidus par rapport aux valeurs ajustées. Les résidus représentent les erreurs de prédiction du modèle et leur distribution est examinée pour évaluer la qualité de l'ajustement du modèle linéaire. Ces visualisations offrent une compréhension graphique de la relation entre les variables et aident à identifier d'éventuelles anomalies ou violations des hypothèses du modèle.

```
# Tracer les donn es et la ligne de r gression
```

```
par(mfrow=c(1, 2)) # Cr er une disposition 1x2 pour les graphiques

# Nuage de sal
plot(ann_exp, sal, main="Nuage␣de␣sal", xlab="Exp rience␣en␣ann es", ylab="Sal

# Ajouter la ligne de r gression
abline(coef=c(intercept, slope), col="red")

# Graphique des r sidus par rapport aux valeurs ajust es
plot(lm_model, which=1)
```



1.3.4 Etape4 : Test d'hypothèse

```
cat("\nTest␣d'hypoth se:\n")
t_test <- summary(lm_model)$coefficients["ann_exp", "t␣value"]
p_value <- 2 * pt(-abs(t_test), df=n-2) # Test bilat ral
```

Interprétation du Test d'Hypothèse :

Hypothèse Nulle (H_0) : L'hypothèse nulle postule qu'il n'y a pas de relation significative entre les années d'expérience (`ann_exp`) et les salaires (`sal`), c'est-à-dire que la pente de la ligne

```
> cat("valeur t :", r
valeur t : 24.95
> cat("valeur p :", f
valeur p : 1.14e-20
```

de régression est égale à zéro.

Hypothèse Alternative (H_1) : L'hypothèse alternative suggère qu'il existe une relation significative entre les années d'expérience et les salaires, indiquant que la pente de la ligne de régression est différente de zéro.

Statistique de Test : La statistique de test (t) est calculée comme 24.95.

Valeur p : La valeur p est extrêmement petite (1.1410^{-20}), bien en dessous d'un seuil de significativité communément choisi (comme 0.05).

Conclusion :

La valeur p très faible suggère que nous avons des preuves statistiques significatives pour rejeter l'hypothèse nulle. En d'autres termes, il y a une relation significative entre les années d'expérience et les salaires. La pente de la ligne de régression est statistiquement différente de zéro, indiquant que l'expérience a un impact significatif sur les salaires dans notre modèle.

En résumé, le test d'hypothèse confirme que l'inclusion de la variable d'expérience dans le modèle de régression est justifiée, car elle a une influence significative sur les salaires.

1.4 Conclusion Génerale

En conclusion, cette étude a exploré la relation entre les années d'expérience et les salaires en utilisant deux approches complémentaires : une approche mathématique basée sur les formules de la régression linéaire simple, et une approche pratique implémentée avec le langage de programmation R. L'approche mathématique a permis de formaliser le modèle et de calculer les coefficients clés, tandis que l'approche avec R a offert une mise en œuvre pratique, incluant la visualisation des données et la réalisation d'un test d'hypothèse pour évaluer la signification de la relation.

Les résultats obtenus des deux approches convergent vers une conclusion unanime : il existe une relation significative entre les années d'expérience et les salaires. Le test d'hypothèse a renforcé cette conclusion en fournissant des preuves statistiques solides. La visualisation des données, notamment le nuage de points avec la ligne de régression et le graphique des résidus, a enrichi la compréhension de la dynamique sous-jacente.

Regression Lineaire Multiple

Plan

2.1 Description des données

Les données proviennent du fichier `basketball.csv` et comprennent plusieurs variables qui caractérisent les performances des joueurs de basketball

```
# Lire les donn es.
data <- read.csv("basketball.csv", header=TRUE)
points = data$X5 # Y : points
height = data$X1 # X1: height
weight = data$X2 # X2: weight
field = data$X3  # X3: field
free = data$X4   # X4: free
n = length(points)   # 54 joueurs
# Statistiques descriptives du jeu de donn es.
head(data)
```

	X1	X2	X3	X4	X5
1	6.8	225	0.442	0.672	9.2
2	6.3	180	0.435	0.797	11.7
3	6.4	190	0.456	0.761	15.8
4	6.2	180	0.416	0.651	8.6
5	6.9	205	0.449	0.900	23.2
6	6.4	225	0.431	0.780	27.4

Points : Les données relatives aux points marqués dans les matchs peuvent être représentées par la variable `points`, extraite du fichier `basketball.csv`.

Height : La variable `height` correspond à la hauteur des joueurs, extraite de la colonne `X1` dans le fichier de données.

Weight : La variable `weight` représente le poids des joueurs, extraite de la colonne `X2` dans le fichier de données.

Field : La variable `field` est associée à la spécialité ou au domaine de jeu des joueurs, extraite de la colonne `X3` dans le fichier de données.

Free : Enfin, la variable `free` représente une caractéristique liée à la liberté des joueurs, extraite de la colonne `X4` dans le fichier de données.

```
summary(data)
```

```
       X1              X2             X3               X4               X5       
 Min.   :5.700   Min.   :105.0   Min.   :0.2910   Min.   :0.2440   Min.   : 2.80  
 1st Qu.:6.225   1st Qu.:185.0   1st Qu.:0.4153   1st Qu.:0.7130   1st Qu.: 8.15  
 Median :6.650   Median :212.5   Median :0.4435   Median :0.7535   Median :10.75  
 Mean   :6.587   Mean   :209.9   Mean   :0.4491   Mean   :0.7419   Mean   :11.79  
 3rd Qu.:6.900   3rd Qu.:235.0   3rd Qu.:0.4835   3rd Qu.:0.7953   3rd Qu.:13.60  
 Max.   :7.600   Max.   :263.0   Max.   :0.5990   Max.   :0.9000   Max.   :27.40  
```

summary : pour obtenir des descriptions de base pour l'ensemble des données.

2.2 Approche Mathematique

L'approche mathématique de la régression linéaire simple implique de trouver la meilleure droite de régression qui minimise la somme des carrés des erreurs entre les valeurs observées (le salaire dans ce cas) et les valeurs prédites par le modèle à traver des relations mathématiques.

2.2.1 Les formules utilisés

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{Y}$$

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$$

$$\hat{\boldsymbol{\varepsilon}} = \mathbf{Y} - \hat{\mathbf{Y}}$$

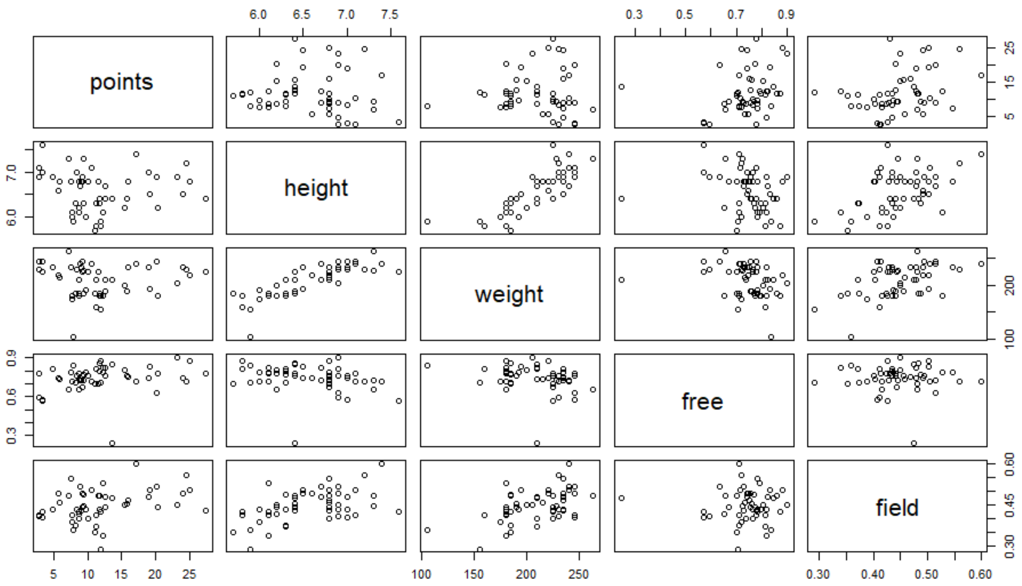
2.2.2 Etape1 : Matrice de diagramme de dispersion

L'instruction R fournie crée une matrice de diagrammes de dispersion entre la variable de réponse "points" et chacun des prédicteurs "height", "weight", "free", et "field". Chaque cellule de la matrice présente un nuage de points qui illustre graphiquement la relation entre la variable de réponse et un prédicteur spécifique.

La fonction pairs() est utilisée pour générer cette matrice de diagrammes de dispersion.

Les arguments passés à la fonction indiquent les variables à inclure dans la matrice. En l'occurrence, points height + weight + free + field signifie que la variable de réponse est "points", et les prédicteurs sont "height", "weight", "free", et "field".

```
# Matrice de diagramme de dispersion de la variable de r ponse et de chaque pr
pairs(points ~ height + weight + free + field, cex.labels = 2)
```



2.2.3 Etape2 : Calcule des coefficient

$$\hat{\beta} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$$

```
x = cbind(1, field, free)
xtx = t(x) %*% x

# ( X X )   1
xtxi = solve(xtx)

# ( X X )( X X )   1 = In
xtx %*% xtxi
# proche de la matrice identit


# Calculer beta chapeau
```

```
#    = ( X X )   1  X Y ,.
beta.hat = xtxi %*% t(x) %*% points
beta.hat


# Afficher l' quation
cat("Approche␣classique:\n")
cat("Y␣=", round(intercept, 2), "+", round(slope, 2), "*␣X\n\n")
```

```
> # (X'X)-1
> xtxi = solve(xtx)
> # (X'X)(X'X)-1 = In
> xtx %*% xtxi
                                   field free
      1.000000e+00  1.199041e-14    0
field 7.105427e-15  1.000000e+00    0
free  7.105427e-15 -7.327472e-15    1
> # proche de la matrice identité
> # 1.000000e+00  1.199041e-14    0
> # 7.105427e-15  1.000000e+00    0
> # 7.105427e-15 -7.327472e-15    1
>
> # Calculer beta chapeau
> #β = (X'X)-1 X'Y,.
> beta.hat = xtxi %*% t(x) %*% points
> beta.hat
           [,1]
      -15.27738
field  35.82503
free   14.79905
> # intercept -15.27738
> # field      35.82503
> # free       14.79905
```

2.2.4 Etape3 : Calcul des résidus et de sigma

$$\text{hat.matrix} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$$

$$\text{residual} = (\mathbf{I}_n - \text{hat.matrix})\mathbf{Y}$$

$$SS_{\text{res}} = \sum(\text{residual} \cdot \text{residual})$$

$$\sigma^2_{\text{hat}} = \frac{SS_{\text{res}}}{(n-p)}$$

```
#res = Y-Y.hat = Y - X ( X X )   1  X Y  =(In - X ( X X )   1  X  ) Y
hat.matrix = x %*% xtxi %*% t(x)
residual = (diag(n) - hat.matrix) %*% points
# somme des carr s des r sidus
SS.res = sum(residual * residual)
SS.res # 1516.422
# carr  moyen des r sidus  2.hat  =    .hat ( X X )  1 = (|  |  /(n    p)
sigma.squared.hat = SS.res/(n - 2)
sigma.squared.hat # 29.16196
```

```
> #res = Y-Y.hat = Y - X (X'X)-1 X'Y =(In - X (X'X)-1 X') Y
> hat.matrix = x %*% xtxi %*% t(x)
> residual = (diag(n) - hat.matrix) %*% points
> # somme des carrés des résidus
> SS.res = sum(residual * residual)
> SS.res # 1516.422
[1] 1516.422
> # carré moyen des résidus σ2.hatβ = σ².hat (X'X)-1 = (|ε|²/(n - p)) (X'X)-1
> sigma.squared.hat = SS.res/(n - 2)
> sigma.squared.hat # 29.16196
[1] 29.16196
>
```

2.2.5 Etape4 : Test d'hypothèse

```
# Test d'hypoth se
tvaluebeta1hat <- beta.hat[2] / sqrt(sigma.squared.hat * xtxi[2, 2])
```

```
tvaluebeta1hat
pvalue1 = 2 * (1 - pt(abs(tvaluebeta1hat), df = n - 2))
pvalue1
tvaluebeta2hat <- beta.hat[3] / sqrt(sigma.squared.hat * xtxi[3, 3])
tvaluebeta2hat
pvalue2 = 2 * (1 - pt(abs(tvaluebeta2hat), df = n - 2))
pvalue2
```

```
> # Test d'hypothèse
> tvaluebeta1hat <- beta.hat[2] / sqrt(sigma.squared.hat * xtxi[2, 2])
> tvaluebeta1hat
[1] 2.730768
> pvalue1 = 2 * (1 - pt(abs(tvaluebeta1hat), df = n - 2))
> pvalue1
[1] 0.008606811
>
> tvaluebeta2hat <- beta.hat[3] / sqrt(sigma.squared.hat * xtxi[3, 3])
> tvaluebeta2hat
[1] 1.997663
> pvalue2 = 2 * (1 - pt(abs(tvaluebeta2hat), df = n - 2))
> pvalue2
[1] 0.0509969
```

Interprétation du Test d'Hypothèse :

Pour $\hat{\beta}_1$, la t-value est d'environ 2.73 et la p-value est d'environ 0.0086. Étant donné que la p-value est inférieure à un seuil de 0.05, on rejette l'hypothèse nulle et conclut que $\hat{\beta}_1$ est significativement différent de zéro.

Pour $\hat{\beta}_2$, la t-value est d'environ 1.9977 et la p-value est d'environ 0.051. Bien que la p-value soit légèrement supérieure à 0.05, cela suggère une tendance à ne pas rejeter l'hypothèse nulle, bien que cela dépende du seuil de significativité choisi.

2.3 Approche par modèle , La fonction lm()

2.3.1 Etape1 : Utilisation de lm()

La fonction lm() en R est utilisée pour ajuster des modèles linéaires. Elle fait partie du système statistique de R et est largement utilisée pour effectuer des analyses de régression.

```
modele <- lm(points ~ height + weight + free + field, data = data)
```

2.3.2 Etape2 : Utilisation de summary

La fonction summary() appliquée à un modèle linéaire (lm) en R est utilisée pour obtenir un résumé statistique détaillé du modèle linéaire ajusté à l'aide de la fonction lm().

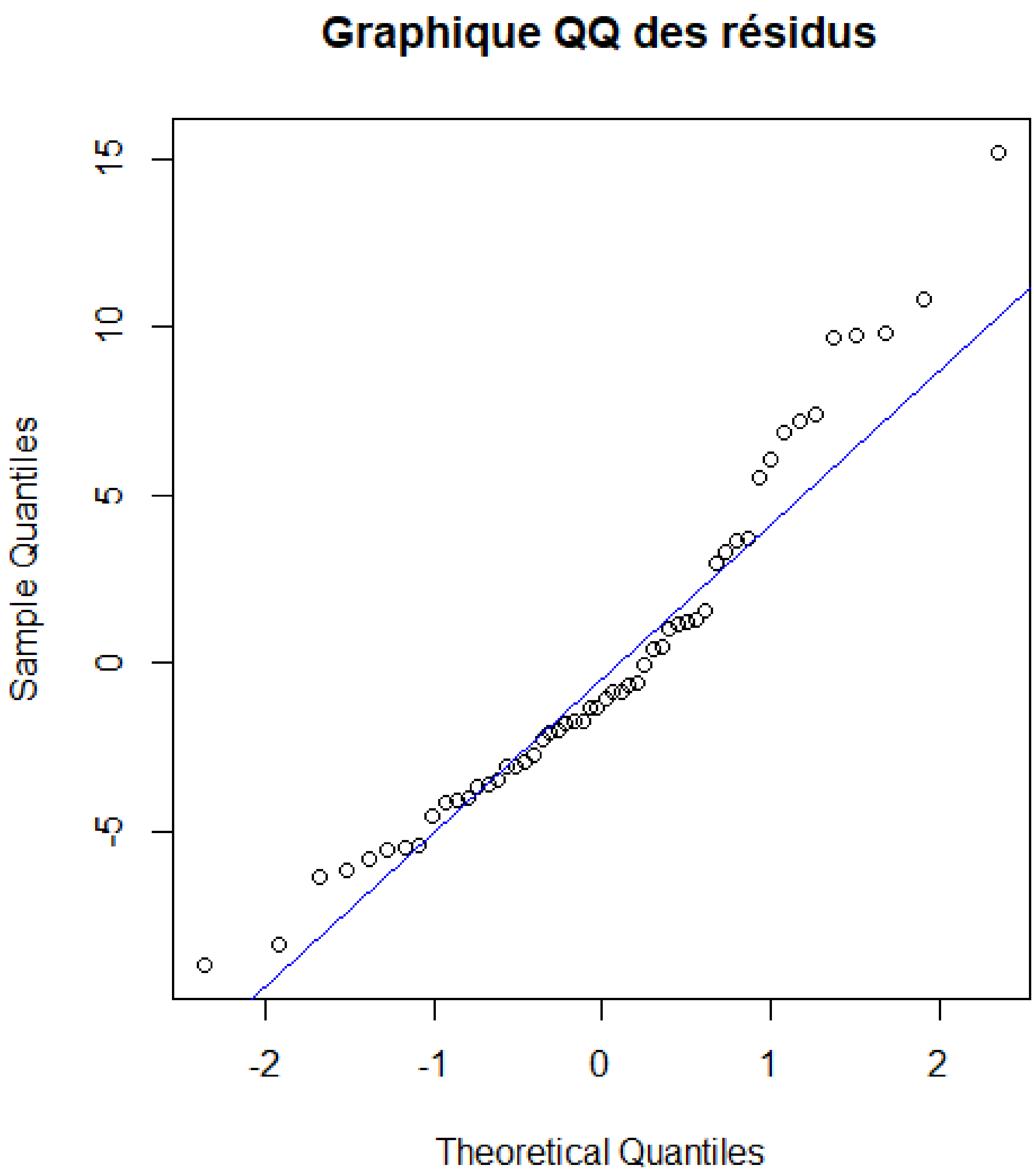
```
# R sum  du mod le
summary(modele)
```

```
Call:
lm(formula = points ~ height + weight + free + field, data = data)

Residuals:
   Min     1Q Median     3Q    Max 
-8.966 -3.545 -1.187  2.613 15.211 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)   
(Intercept)  4.148707  14.855006   0.279  0.78121   
height      -3.690499   2.970780  -1.242  0.22005   
weight       0.009458   0.046297   0.204  0.83897   
free        11.371019   7.868536   1.445  0.15479   
field       47.940199  15.709131   3.052  0.00367 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.411 on 49 degrees of freedom
Multiple R-squared:  0.2223,	Adjusted R-squared:  0.1588 
F-statistic: 3.501 on 4 and 49 DF,  p-value: 0.01364
```

2.3.3 Etape3 : Visualisation des résultats

Le code R fourni génère un graphique quantile-quantile (QQ plot) pour évaluer la normalité des résidus d'un modèle statistique. La fonction qqnorm() est utilisée pour créer le graphique QQ, affichant les quantiles théoriques sur l'axe x et les quantiles observés des résidus sur l'axe y. La ligne diagonale tracée avec qqline() représente la droite des quantiles attendus pour une distribution normale.

```
# Trac  d'un graphique QQ pour v rifier la normalit
qqnorm(resid(modele), main = "Graphique␣QQ␣des␣r sidus")
qqline(resid(modele), col = "blue")
```



2.3.4 Etape4 : Test d'hypothèse

```
# Test des hypoth ses
# Vous pouvez extraire les valeurs t et les p-valeurs directement   partir de l

# Acc s aux estimations des coefficients
beta.hat <- coef(modele)

# Acc s aux  carts -types
```

```
se <- summary(modele)$coefficients[, "Std.␣Error"]

# Calcul des valeurs t
t_values <- beta.hat / se

# Calcul des p-valeurs   deux queues
p_values <- 2 * (1 - pt(abs(t_values), df = n - length(beta.hat)))

# Affichage des valeurs t et des p-valeurs
cbind(t_values, p_values)
```

```
> # Test des hypothèses
> # Vous pouvez extraire les valeurs t et les p-valeurs directement à partir de la sortie summary(modele).
>
> # Accès aux estimations des coefficients
> beta.hat <- coef(modele)
>
> # Accès aux écarts-types
> se <- summary(modele)$coefficients[, "Std. Error"]
>
> # Calcul des valeurs t
> t_values <- beta.hat / se
>
> # Calcul des p-valeurs à deux queues
> p_values <- 2 * (1 - pt(abs(t_values), df = n - length(beta.hat)))
>
> # Affichage des valeurs t et des p-valeurs
> cbind(t_values, p_values)
              t_values    p_values
(Intercept)  0.2792801 0.781205356
height      -1.2422661 0.220051123
weight       0.2042986 0.838966361
free         1.4451251 0.154788006
field        3.0517411 0.003668459
```

Interprétation du Test d'Hypothèse :

Les résultats du test des hypothèses indiquent les valeurs t et les p-valeurs associées pour chaque coefficient du modèle de régression. Voici une interprétation rapide :

— Pour l'intercept (constante), la valeur t est d'environ 0.28 avec une p-value de 0.78. Étant donné que la p-value est bien supérieure à un seuil communément choisi de 0.05, on ne rejette pas l'hypothèse nulle, suggérant que l'intercept n'est pas significativement différent de zéro.

— Pour la variable "height", la valeur t est d'environ -1.24 avec une p-value de 0.22. La p-value étant supérieure à 0.05, on ne rejette pas l'hypothèse nulle, indiquant que la pente associée à "height" n'est pas significativement différente de zéro.

— Pour la variable "weight", la valeur t est d'environ 0.20 avec une p-value de 0.84. La p-value étant bien au-dessus de 0.05, on ne rejette pas l'hypothèse nulle, suggérant que la pente associée à "weight" n'est pas significativement différente de zéro.

— Pour la variable "free", la valeur t est d'environ 1.45 avec une p-value de 0.15. Bien que la p-value ne soit pas inférieure à 0.05, elle est relativement proche, suggérant une tendance à rejeter l'hypothèse nulle.

— Pour la variable "field", la valeur t est d'environ 3.05 avec une p-value de 0.0037. La p-value étant inférieure à 0.05, on rejette l'hypothèse nulle, indiquant que la pente associée à "field" est significativement différente de zéro.

2.4 Conclusion Génerale

En résumé, l'analyse de régression linéaire multiple a exploré les relations complexes entre une variable de réponse et plusieurs prédicteurs. Les tests d'hypothèses ont permis d'évaluer la significativité des coefficients associés à chaque prédicteur, offrant des insights sur leur contribution à la prédiction du phénomène étudié. Le graphique QQ des résidus a fourni une indication de la normalité des erreurs du modèle.

Chapitre 3

Regression Logistique

Plan

3.1 Description des données

Un chercheur s'intéresse à la manière dont les variables, telles que le GRE ("Graduate Record Exam scores" : scores aux examens d'études supérieures), la moyenne cumulative GPA ("grade point average" :moyenne pondérée cumulative) et le prestige de l'établissement de premier cycle, affectent l'admission aux études supérieures. La variable de réponse, admettre/ne pas admettre, est une variable binaire.

```
library(aod)
library(ggplot2)
mydata <- read.csv("C:/Users/ASUS/Desktop/binary.csv")
head(mydata)
```

```
> head(mydata)
  admit gre  gpa rank
1     0 380 3.61    3
2     1 660 3.67    3
3     1 800 4.00    1
4     1 640 3.19    4
5     0 520 2.93    4
6     1 760 3.00    2
```

admit : une variable de réponse binaire (résultat, dépendante).

Il existe trois variables prédictives : **gre**, **gpa** et **rank**.

Nous traiterons les variables **gre** et **gpa** comme continues.

La variable **rank** prend les valeurs de 1 à 4.

Les établissements de **rank** 1 ont le prestige le plus élevé, tandis que ceux de **rank** 4 ont le plus bas.

summary : pour obtenir des descriptions de base pour l'ensemble des données.

```
summary(mydata)
```

```
> summary(mydata)
     admit             gre             gpa         rank
 Min.   :0.0000   Min.   :220.0   Min.   :2.260   1: 61
 1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   2:151
 Median :0.0000   Median :580.0   Median :3.395   3:121
 Mean   :0.3175   Mean   :587.7   Mean   :3.390   4: 67
 3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670
 Max.   :1.0000   Max.   :800.0   Max.   :4.000
>
```

sapply : pour appliquer la fonction **sd** à chaque variable de l'ensemble de données.

```
sapply(mydata, sd)
```

```
> sapply(mydata, sd)
      admit         gre         gpa        rank 
  0.4660867 115.5165364   0.3805668   0.9444602 
```

```
xtabs(~admit + rank, data = mydata)
```

```
> xtabs(~admit + rank, data = mydata)
     rank
admit  1  2  3  4
    0 28 97 93 55
    1 33 54 28 12
```

3.2 Utilisation du modèle logit

3.2.1 Estimation des paramètres du modèle

Premièrement, nous convertissons rank en facteur pour indiquer que rank doit être traité comme une variable catégorielle.

Le code ci-dessous estime un modèle de régression logistique à l'aide de la fonction **glm**.

```
mydata$rank <- factor(mydata$rank)
mylogit <- glm(admit ~gre + gpa + rank, data = mydata, family = "binomial")
summary(mylogit)
```

```
Call:
glm(formula = admit ~ gre + gpa + rank, family = "binomial",
    data = mydata)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.989979   1.139951  -3.500 0.000465 ***
gre          0.002264   0.001094   2.070 0.038465 *
gpa          0.804038   0.331819   2.423 0.015388 *
rank2       -0.675443   0.316490  -2.134 0.032829 *
rank3       -1.340204   0.345306  -3.881 0.000104 ***
rank4       -1.551464   0.417832  -3.713 0.000205 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 458.52  on 394  degrees of freedom
AIC: 470.52

Number of Fisher Scoring iterations: 4
```

On utilise la fonction confint pour obtenir des intervalles de confiance pour les estimations de coefficients.

```
confint(mylogit)
```

```
> confint(mylogit)
Waiting for profiling to be done...
                    2.5 %       97.5 %
(Intercept) -6.2716202334 -1.792547080
gre          0.0001375921  0.004435874
gpa          0.1602959439  1.464142727
rank2       -1.3008888002 -0.056745722
rank3       -2.0276713127 -0.670372346
rank4       -2.4000265384 -0.753542605
```

3.2.2 Test de Wald

wald.test : c'est une fonction de la bibliothèque aod pour tester un effet global de rank.

Sigma : fournit la matrice de covariance de variance des termes d'erreur.

Terms : indique à R quels termes du modèle doivent être testés, dans ce cas, les termes 4, 5 et 6 sont les trois termes pour les niveaux de rank.

```
wald.test(b = coef(mylogit), Sigma = vcov(mylogit), Terms = 4 :6)
```

```
> wald.test(b = coef(mylogit), Sigma = vcov(mylogit), Terms = 4 :6)
Wald test:
----------

Chi-squared test:
X2 = 20.9, df = 3, P(> X2) = 0.00011
```

La statistique de test du chi-deux de 20,9, avec 3 degrés de liberté (df), est associée à une p-value =0,00011 indiquant que l'effet global du rank est statistiquement significatif.

```
# Tester la difference entre le coeff de rank=2 et le coeff de rank=3 :
l <- cbind(0, 0, 0, 1, -1, 0)
wald.test(b = coef(mylogit), Sigma = vcov(mylogit), L = l)
```

```
> wald.test(b = coef(mylogit), Sigma = vcov(mylogit), L = l)
Wald test:
----------

Chi-squared test:
X2 = 5.5, df = 1, P(> X2) = 0.019
```

La statistique de test du chi-deuxde 5,5 avec 1 degré de liberté est associée à p-value=0,019, indiquant que la différence entre le coefficient de rank=2 et le coefficient de rank=3 est statistiquement significatif.

3.2.3 Odds-ratios (OR)

```
# odds-ratios seulement :
exp(coef(mylogit))
```

```
> exp(coef(mylogit))
(Intercept)         gre         gpa       rank2       rank3       rank4 
  0.0185001   1.0022670   2.2345448   0.5089310   0.2617923   0.2119375 
```

```
# odds ratios et 95% IC :
exp(cbind(OR = coef(mylogit), confint(mylogit)))
```

```
> exp(cbind(OR = coef(mylogit), confint(mylogit)))
Waiting for profiling to be done...
                   OR       2.5 %    97.5 %
(Intercept) 0.0185001 0.001889165 0.1665354
gre         1.0022670 1.000137602 1.0044457
gpa         2.2345448 1.173858216 4.3238349
rank2       0.5089310 0.272289674 0.9448343
rank3       0.2617923 0.131641717 0.5115181
rank4       0.2119375 0.090715546 0.4706961
> 
```

Maintenant, nous pouvons dire que pour une augmentation d'une unité de gpa, les chances d'être admis à l'université (par rapport à ne pas être admis) augmentent d'un facteur de 2,23.

3.2.4 Probabilités prédites

Nous commencerons par calculer la probabilité d'admission prédite à chaque valeur de rank, en tenant gre et gpa à leur moyenne. Nous créons et visualisons d'abord le bloc de données.

```
newdata1 <- with(mydata, data.frame(gre = mean(gre), gpa = mean(gpa),
        rank = factor(1 :4)))
newdata1
```

```
> newdata1
    gre    gpa rank
1 587.7 3.3899    1
2 587.7 3.3899    2
3 587.7 3.3899    3
4 587.7 3.3899    4
>
```

newdata1$rankP : indique à R que nous voulons créer une nouvelle variable dans l'ensemble de données (data frame) newdata1 appelée rankP. Le reste de la commande indique à R que les valeurs de rankP doivent être des prédictions faites à l'aide de la fonction predict() :

```
newdata1$rankP <- predict(mylogit, newdata = newdata1, type = "response")
newdata1
```

```
> newdata1
    gre    gpa rank     rankP
1 587.7 3.3899    1 0.5166016
2 587.7 3.3899    2 0.3522846
3 587.7 3.3899    3 0.2186120
4 587.7 3.3899    4 0.1846684
```

Dans le résultat ci-dessus, nous voyons que la probabilité prédite d'être accepté dans un programme d'études supérieures est de 0,52 pour les étudiants des établissements de premier cycle les plus prestigieux (rank=1), et 0,18 pour les étudiants des établissements les moins bien classes (rank=4).

Nous pouvons faire quelque chose de très similaire pour créer une table de probabilités prédites faisant varier la valeur de gre et de rank.

Nous allons les tracer, nous allons donc créer 100 valeurs de gre entre 200 et 800, à chaque valeur de rank (c'est-à-dire 1, 2, 3 et 4).

```
newdata2 <- with(mydata, data.frame(gre = rep(seq(from = 200, to = 800,
length.out = 100), 4), gpa = mean(gpa), rank = factor(rep(1 :4, each = 100))))
```

Le code pour générer les probabilités prédites (la première ligne ci-dessous) est le même que précédemment, sauf que nous allons également demander des erreurs standard afin de pouvoir tracer un intervalle de confiance. Nous obtenons les estimations sur l'échelle des liens et transformons à la fois les valeurs prédites et les limites de confiance en probabilités :

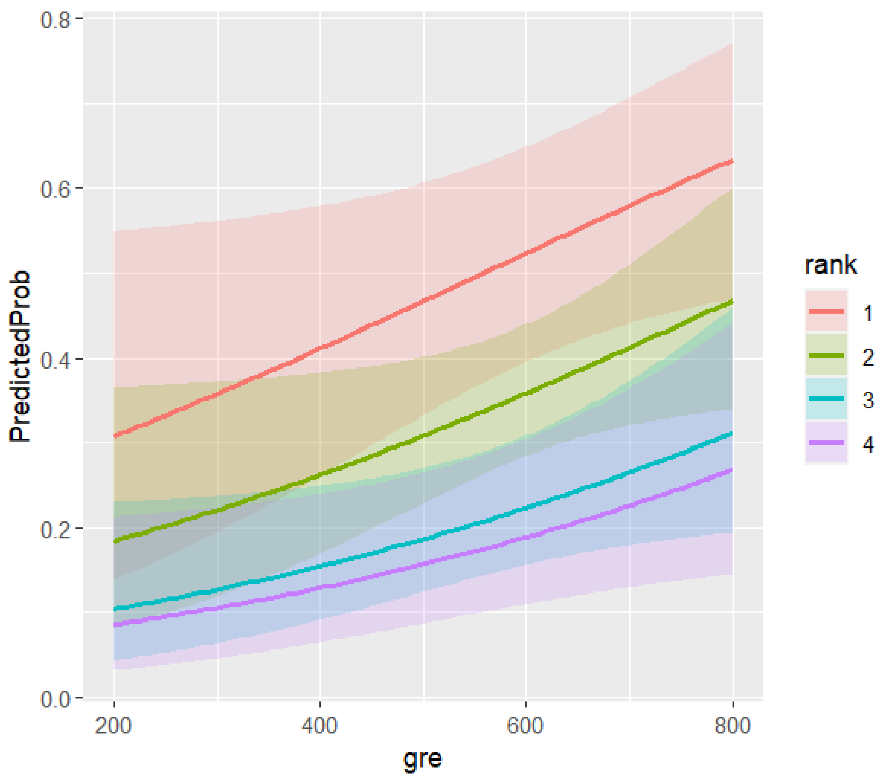
```
newdata3 <- cbind(newdata2, predict(mylogit, newdata = newdata2,
```

```
type = "link",se= TRUE))
newdata3 <- within(newdata3, { PredictedProb <- plogis(fit)
        LL <- plogis(fit - (1.96 * se.fit))
        UL <- plogis(fit + (1.96 * se.fit))})
head(newdata3)
```

```
> head(newdata3)
       gre    gpa rank        fit    se.fit residual.scale        UL        LL PredictedProb
1 200.0000 3.3899    1 -0.8114870 0.5147714              1 0.5492064 0.1393812     0.3075737
2 206.0606 3.3899    1 -0.7977632 0.5090986              1 0.5498513 0.1423880     0.3105042
3 212.1212 3.3899    1 -0.7840394 0.5034491              1 0.5505074 0.1454429     0.3134499
4 218.1818 3.3899    1 -0.7703156 0.4978239              1 0.5511750 0.1485460     0.3164108
5 224.2424 3.3899    1 -0.7565919 0.4922237              1 0.5518545 0.1516973     0.3193867
6 230.3030 3.3899    1 -0.7428681 0.4866494              1 0.5525464 0.1548966     0.3223773
>
```

Il peut également être utile d'utiliser des graphiques de probabilités prédites pour comprendre et/ou présenter le modèle. Ci-dessous, nous faisons un graphique avec les probabilités prédites et des intervalles de confiance à 95% .

```
ggplot(newdata3, aes(x = gre, y = PredictedProb)) + geom_ribbon(aes(ymin = LL,
              ymax = UL, fill = rank), alpha = 0.2)
              + geom_line(aes(colour = rank), size = 1)
```



3.2.5 La statistique de test de la différence de déviance pour les deux modèles

Trouver la différence de déviance pour les deux modèles (c'est-à-dire la statistique de test) :

```
with(mylogit, null.deviance - deviance)
-> [1] 41.45903
```

Les degrés de liberté pour la différence entre les deux modèles sont égaux au nombre de variables prédictives dans le mode et peuvent être obtenus en utilisant :

```
with(mylogit, df.null - df.residual)
-> [1] 5
```

Enfin, p-value peut être obtenue en utilisant :

```
with(mylogit, pchisq(null.deviance - deviance, df.null - df.residual,
lower.tail = FALSE))
-> [1] 7.578194e-08
```

Le chi-deux de 41,46 avec 5 degrés de liberté (dl) et une p-value associée inférieure à 0,001 nous indique que notre modèle dans son ensemble s'adapte nettement mieux qu'un modèle vide. C'est ce qu'on appelle parfois un test de rapport de vraisemblance (le résidu de déviance est de -2*log de vraisemblance).
Pour voir le log de vraisemblance du modèle :

```
logLik(mylogit)
-> 'log␣Lik.' -229.2587 (df=6)
```

Chapitre 4

Regression de Poisson

Plan

4.1 Description des données

Le nombre de bourses obtenues par les élèves d'une école secondaire : Les prédicteurs du nombre de bourses obtenues suivant le type de programme dans lequel l'étudiant était inscrit (par exemple, professionnel, général ou académique) et le résultat de son examen final en mathématiques.

```
library(sandwich)
library(msm)
library(ggplot2)
p <- read.csv("poisson_sim.csv")
head(p)
```

	id	num_awards	prog	math
	<dbl>	<dbl>	<dbl>	<dbl>
1	45	0	3	41
2	108	0	1	41
3	15	0	3	44
4	67	0	3	42
5	153	0	3	40
6	51	0	1	42

num awards : la variable de réponse et indique le nombre de boursses obtenues par les élèves d'un lycée au cours d'une année.

Il existe deux variables prédictives : prog et math.

math : la variable prédictive continue et représente les notes des élèves à leur examen final de mathématiques.

prog : la variable prédictive catégorique avec trois niveaux indiquant le type de programme dans lequel les étudiants étaient inscrits. Il est codé comme 1 = "General", 2 = "Academic" et 3 = "Vocational".

summary : pour obtenir des descriptions de base pour l'ensemble des données.

```
p <- within(p, {
  prog <- factor(prog, levels=1:3, labels=c("General", "Academic",
                                            "Vocational"))
  id <- factor(id)
})
summary(p)
```

```
       id        num_awards           prog          math      
 1      :  1   Min.   :0.00   General   : 45   Min.   :33.00  
 2      :  1   1st Qu.:0.00   Academic  :105   1st Qu.:45.00  
 3      :  1   Median :0.00   Vocational: 50   Median :52.00  
 4      :  1   Mean   :0.63                    Mean   :52.65  
 5      :  1   3rd Qu.:1.00                    3rd Qu.:59.00  
 6      :  1   Max.   :6.00                    Max.   :75.00  
 (Other):194                                                  
```

Num awards : une variable qui represente le nombre des recompenses aquieuis durant une année scolaire. Le min est 0, le max est 6.

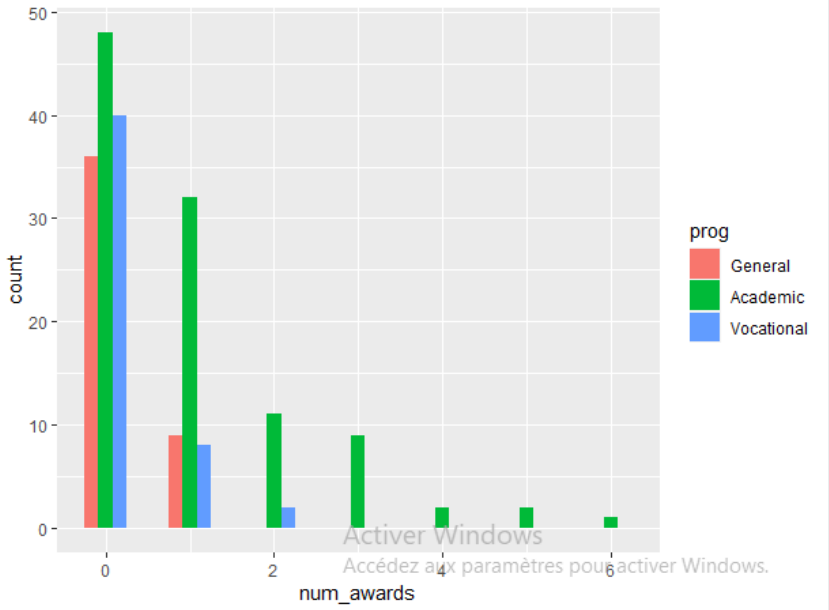
Prog : Le programme d'étude : general, academic et vocational.

Math : le score final dans la matiere mathematique : Min 33, max 75 (/100).

La moyenne et la mediane du score finale sont egaux, ce qui peut être interpreté par le faite de ne pas avoir trop de valeurs aberrantes.

ggplot : Histograme qui montre la distribution des données

```
ggplot(p, aes(num_awards, fill = prog)) +
geom_histogram(binwidth=.5, position="dodge")
```



Le histogramme resultant du bloc au dessus, montre les chiffres moyens des récompenses par type de programme et semble suggérer que le type de programme est un bon candidat pour prédire le nombre de récompenses, notre variable de résultat, car la valeur moyenne du résultat semble varier selon le programme.

4.2 Modélisation

```
m1 <- glm(num_awards ~ prog + math, family="poisson", data=p)
summary(m1)
```

```
Call:
glm(formula = num_awards ~ prog + math, family = "poisson", data = p)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2043  -0.8436  -0.5106   0.2558   2.6796

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -5.24712    0.65845  -7.969 1.60e-15 ***
progAcademic    1.08386    0.35825   3.025  0.00248 **
progVocational  0.36981    0.44107   0.838  0.40179
math            0.07015    0.01060   6.619 3.63e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 287.67  on 199  degrees of freedom
Residual deviance: 189.45  on 196  degrees of freedom
AIC: 373.5

Number of Fisher Scoring iterations: 6
```

```
cov.m1 <- vcovHC(m1, type="HC0")
std.err <- sqrt(diag(cov.m1))
r.est <- cbind(Estimate= coef(m1), "Robust␣SE" = std.err,
               "Pr(>|z|)" = 2 * pnorm(abs(coef(m1)/std.err), lower.tail=FALSE),
               LL = coef(m1) - 1.96 * std.err,
               UL = coef(m1) + 1.96 * std.err)

r.est
```

	Estimate	Robust SE	Pr(>|z|)	LL	UL
(Intercept)	-5.2471244	0.64599839	4.566630e-16	-6.51328124	-3.98096756
progAcademic	1.0838591	0.32104816	7.354745e-04	0.45460476	1.71311353
progVocational	0.3698092	0.40041731	3.557157e-01	-0.41500870	1.15462716
math	0.0701524	0.01043516	1.783975e-11	0.04969947	0.09060532

Le coefficient du variable math est de 0,07. Cela signifie que le log count pour une augmentation d'une unité en mathématiques est de 0,07.

La variable indicatrice progAcademic compare entre prog = "Academic" et prog = "General", le log count attendu pour prog = "Academic" augmente d'environ 1,1.

La variable indicatrice progVocational représente la différence attendue dans le log count (environ 0,37) entre prog = "Vocational" et le groupe de référence (prog = "General").

4.3 Test de siginificativité

```
with(m1, cbind(res.deviance = deviance, df = df.residual,
               p = pchisq(deviance, df.residual, lower.tail=FALSE)))
```

	res.deviance	df	p
[1,]	189.4496	196	0.6182274

On a utilisé chi squared test pour determiné la siginificativité globale du modèle, si p-value $<$ 0.05, alors le modele n'est pas significative et n'explique pas les données (donc l'hypothese de linearité est detruite). LE test chi-deux se base sur la comparaison de la déviance entre le modèle ideale et le note modèle.

P $>$ .05 donc notre modèle explique bien les données et globalelement siginificatif.

4.4 Test du modèle

```
(s1 <- data.frame(math = mean(p$math),
                  prog = factor(1:3, levels = 1:3, labels = levels(p$prog))))
predict(m1, s1, type="response", se.fit=TRUE)
```

```
> (s1 <- data.frame(math = mean(p$math),
+                   prog = factor(1:3, levels = 1:3, labels = levels(p$prog))))
    math       prog
1 52.645    General
2 52.645   Academic
3 52.645 Vocational
> predict(m1, s1, type="response", se.fit=TRUE)
$fit
        1         2         3
0.2114109 0.6249446 0.3060086

$se.fit
         1          2          3
0.07050108 0.08628117 0.08833706

$residual.scale
[1] 1
```

On remarque que le nombre prédit des recompenses est 0.2 pour le programme 1 (general), 0.62 pour le programme Academique, et 0.3 pour le programme vocational.

```
p$phat <- predict(m1, type="response")
p <- p[with(p, order(prog, math)), ]
ggplot(p, aes(x = math, y = phat, colour = prog)) +
  geom_point(aes(y = num_awards), alpha=.5, position=position_jitter(h=.2)) +
  geom_line(size = 1) +
  labs(x = "Math␣Score", y = "Expected␣number␣of␣awards")
```

On calcule et enregistre les valeurs prédits du variable num awards comme une nouvelle variable de notre dataset.

On enregistre les données de notre dataset triées par programme et aprés par math.

On montre les résultats par un graphe qui illustre les résultats.

Remarque : Le graphe resultant du bloc ci-dessus confirme les resultats precedentes. La graphe confirme que le modèle favorise les programme academique dans l'attribution des récompenses, essentiellement si la personne ait un bon score en math.