TP 5 - Régression linéaire de Poisson - Compte rendu de groupe

Matthias MAZET, Garance MALNOË

2025-12-10

Contents


```
# Packages statistiques
library(glmnet)

# Packages de style
library(ggplot2)
library(ggpubr)
library(GGally)
library(tidyverse)
library(knitr)
library(kableExtra)
```

1 *Préparation des données.*

Nous récupérons le jeu de données *bike sharing* à partir du fichier CSV disponible sur le site UCI.

```
data <- read.csv("day.csv")
```

Nous supprimons la variable "dteday" qui indique le jour étudié qui se trouve au format "chr" et qui ne nous semble pas pertienent. Nous supprimons également les variables "casual" et "registered" qui correspondent au deux types de location possibles dont la somme est égale à la variable à prédire "cnt". Enfin, nous supprimons aussi la variable "instant" qui correspond simplement à l'index de chaque observation.

```
# Nettoyage
data <- data[, -c(1, 2, 14, 15)] # Suppression de variables
```

Nous vérifions ensuite les valeurs manquantes, la dimension des données et les valeurs aberrantes.

```
sum(is.na(data))
```

```
## [1] 0
```

```
dim(data)
```

```
## [1] 731  12
```

Le jeu de données ne compte aucune valeur valeur manquante (`sum(is.na(data))` $= 0$) et compte ainsi $n = 731$ observations de $p = 12$ variables (une variable à prédire et onze variables prédictives). Il est bon de noter que nous sommes dans un cas où $n > p$.

```
# Résumés statistiques des variables
kable(
  summary(data[, 1:6], digits = 2), caption = "Résumé statistique des 6 premières variables",
  format = "latex", booktabs = TRUE, escape = FALSE, align = "r"
) %>%
  kable_styling(
    latex_options = c("striped", "HOLD_position", "scale_down"), font_size = 9
  ) %>%
  row_spec(0, bold=TRUE)
```

Table 1: Résumé statistique des 6 premières variables

season	yr	mnth	holiday	weekday	workingday
Min. :1.0	Min. :0.0	Min. : 1.0	Min. :0.000	Min. :0	Min. :0.00
1st Qu.:2.0	1st Qu.:0.0	1st Qu.: 4.0	1st Qu.:0.000	1st Qu.:1	1st Qu.:0.00
Median :3.0	Median :1.0	Median : 7.0	Median :0.000	Median :3	Median :1.00
Mean :2.5	Mean :0.5	Mean : 6.5	Mean :0.029	Mean :3	Mean :0.68
3rd Qu.:3.0	3rd Qu.:1.0	3rd Qu.:10.0	3rd Qu.:0.000	3rd Qu.:5	3rd Qu.:1.00
Max. :4.0	Max. :1.0	Max. :12.0	Max. :1.000	Max. :6	Max. :1.00

```
kable(
  summary(data[, 7:12], digits = 2), caption = "Résumé statistique des 6 dernières variables",
  digit = 3, format = "latex", booktabs = TRUE, escape = FALSE, align = "r"
) %>%
  kable_styling(
    latex_options = c("striped", "HOLD_position", "scale_down"), font_size = 9
  ) %>%
  row_spec(0, bold=TRUE)
```

Table 2: Résumé statistique des 6 dernières variables

weathersit	temp	atemp	hum	windspeed	cnt
Min. :1.0	Min. :0.059	Min. :0.079	Min. :0.00	Min. :0.022	Min. : 22
1st Qu.:1.0	1st Qu.:0.337	1st Qu.:0.338	1st Qu.:0.52	1st Qu.:0.135	1st Qu.:3152
Median :1.0	Median :0.498	Median :0.487	Median :0.63	Median :0.181	Median :4548
Mean :1.4	Mean :0.495	Mean :0.474	Mean :0.63	Mean :0.190	Mean :4504
3rd Qu.:2.0	3rd Qu.:0.655	3rd Qu.:0.609	3rd Qu.:0.73	3rd Qu.:0.233	3rd Qu.:5956
Max. :3.0	Max. :0.862	Max. :0.841	Max. :0.97	Max. :0.507	Max. :8714

D'après les résumés statistiques, aucune variable ne semble avoir de valeur aberrante qui serait à exlcure.

Dans le cadre de notre étude, nous souhaitons prédire la variable "cnt", qui compte le nombre d'emprunts de vélos réalisés dans une journée et est donc une variables de comptage. Ainsi, nous pouvons donc bien appliquer une régression de Poisson afin de modéliser cette variable.

2 *Estimation et stabilité de l'estimateur.*

Nous souhaitons ajuster un modèle de régression de Poisson tout en applicant une régularisation LASSO afin d'effectuer une sélection des variables prédictive. Pour cela, nous allons utiliser le package `glmnet` en précisant l'argument `family = "poisson"` dans l'appel des fonctions. Nous conservons arbitrairement la valeur de `lambda.1se` obtenue par validation croisée commme meilleure valeur du paramètre de régularisation.

```
# Format matriciel pour glmnet
X <- as.matrix(data[colnames(data) != "cnt"])
y <- as.matrix(data$cnt)

# Seed pour la reproductibilité
set.seed(5)

# Fit du modèle
lambda_1se <- cv.glmnet(x = X, y = y, alpha = 1, family = "poisson")$lambda.1se
mod_lasso <- glmnet(x = X, y = y, alpha = 1, lambda = lambda_1se, family = "poisson")
```

Le paramètre de régularisation retenu est `lambda.1se` $= 99.094$, et les variables sélectionnées avec les données de base et cette valeur sont : season, yr, holiday, weekday, weathersit, temp, atemp, windspeed.

Nous souhaitons maintenant analyser la stabilité de la sélection des variables. Pour cela, nous effectuons un bootstrap sur les données avec 100 itérations. Pour chaque itération, nous regardons quelles variables ont été sélectionnées lorsque l'on choisit comme paramètre de régularisation le `lambda.1se` obtenu par validation croisée sur les nouvelles données. Nous obtenons finalement une matrice nous indiquant, pour chaque itération, quelles variables ont été sélectionnées et ainsi estimer la fréquence de sélection de chaque variable.

```
set.seed(2) # Reproductibilité

# Nombre d'itérations
n_it <- 100

# Matrices des variables sélectionnées à chaque itérations
mat_vars <- matrix(data = 0, ncol = ncol(X), nrow = n_it)
colnames(mat_vars) <- colnames(X)

# Itérations boostrap
for (i in 1:n_it) {
  # Individus bootstrap
  ind_al <- sample(1:nrow(data), size = nrow(data), replace = TRUE)
  X_new <- as.matrix(data[ind_al, colnames(data) != "cnt"])
  y_new <- data[ind_al, "cnt"]
  # Nouvelle CV et Fit du modèle
  lambda_1se_i <- cv.glmnet(x = X_new, y = y_new, alpha = 1, family = "poisson")$lambda.1se
  mod_lasso_i <- glmnet(x = X_new, y = y_new, alpha = 1, lambda = lambda_1se_i, family = "poisson")
  # Variables sélectionnées à lambda.1se
```

```
  mat_vars[i, which(coef(mod_lasso_i) != 0)[-1] - 1] <- 1
}
```

```
# Stabilité des variables
### Dataset de résultat au format long
df_lasso <- t(mat_vars) %>%
  as.data.frame() %>%
  mutate(variable = rownames(.)) %>%
  pivot_longer(cols = -variable, names_to = "iteration", values_to = "selected") %>%
  mutate(iteration = as.numeric(gsub("V", "", iteration)))
### Figure ggplot
ggplot(df_lasso, aes(x = iteration, y = variable, fill = selected)) +
  geom_tile() +
  scale_fill_gradientn(colours = c("white", "black"), limits = c(0, 1), name = "Sélection") +
  labs(x = "Itération", y = "Variables") +
  theme_light() +
  theme(
    axis.text = element_text(size = 8),
    axis.line = element_line(colour = "grey"),
    panel.border = element_blank(),
    panel.grid = element_blank(),
    legend.title = element_text(size = 10, face = "bold"),
    legend.position = "right"
  )
```

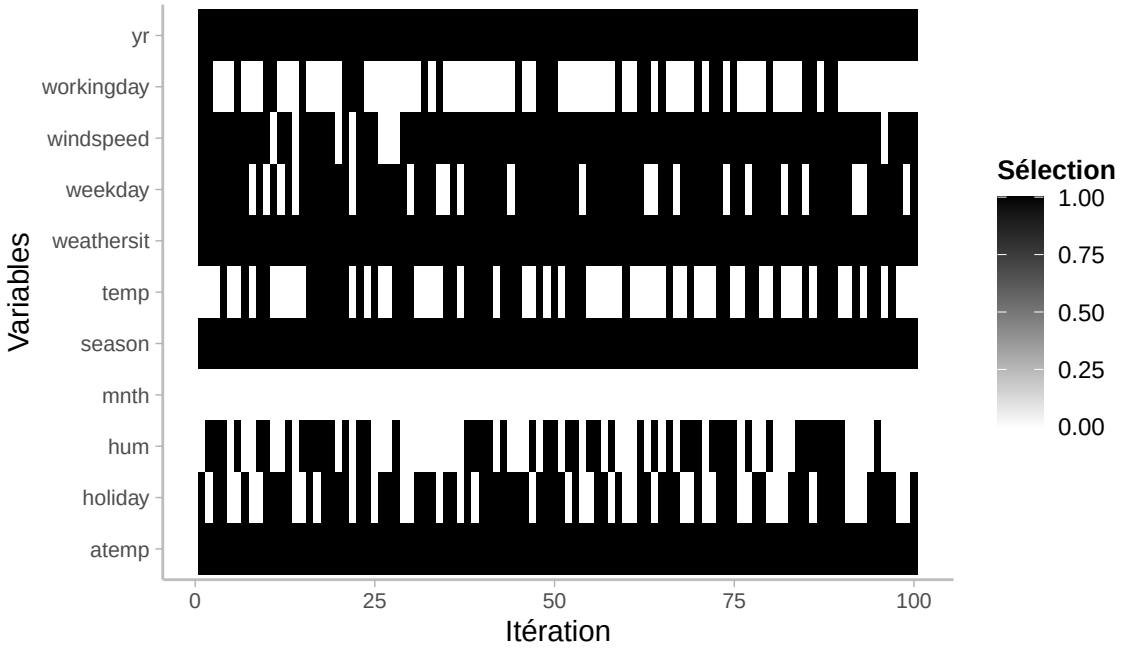

Figure 1: Stabilité à lambda.1se au fil des itérations

```
pourc_selection <- apply(mat_vars, FUN = mean, MARGIN = 2)

kable(
  t(pourc_selection), caption = "Fréquence de sélection de chaque variable",
  format = "latex", booktabs = TRUE, escape = FALSE, align = "r"
) %>%
  kable_styling(latex_options = c("striped", "HOLD_position", "scale_down"), font_size = 9) %>%
  row_spec(0, bold=TRUE)
```

Table 3: Fréquence de sélection de chaque variable

season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed
1	1	0	0.62	0.79	0.28	1	0.45	1	0.49	0.92

Nous obtenons finalement une matrice nous indiquant, pour chaque itération, quelles variables ont été sélectionnées et ainsi estimer la fréquence de sélection de chaque variable.

La figure précédente **(Fig. 1)** présente les variables qui ont été sélectionnées à chaque itération, avec 1 = la variable a été sélectionnée et 0 sinon (la légende de couleur devrait être discrète (0 et 1) mais ggplot ne l'accepte pas sur une heatmap). Nous constatons que les variables "seasons", "yr", "weathersit" et "atemp" sont systématiquement sélectionnées et qu'à l'inverse la variable "mnth" n'est jamais sélecionnée, toutes ces variables sont sélectionnées de manière stable. Au contraire, les variables "holiday", "weekday", "workingday","temp", "hum" et "windspeed" sont sélectionnées de manière instable (entre 5% et 95% du temps). La méthode LASSO en choisissant `lambda.1se` comme paramètre de régularisation présente donc une instabilité sur la sélection des variables, potentiellement du à la présence de multicolinéarité entre certaines variables.

3 *Sélection post-inférence.*

1. *Construction d'un intervalle de confiance.*

Nous souhaitons maintenant construire un intervalle de confiance pour $\hat{\beta}$ l'estimateur du maximum de vraisemblance restreint à $\hat{S}$, l'ensemble des variables sélectionnées par `glmnet` avec pour paramètre de régularisation `lambda.1se`. Pour cela, on utilise une approche naïve basée sur la normalité asymptotique de l'estimateur du maximum de vraisemblance pour obtenir un intervalle de confiance au niveau 1 - α pour chaque coefficient $\hat{\beta}_j$ de $\hat{\beta}$:

$$\hat{IC}_\alpha^j = [\hat{\beta}_j - \frac{q_{1-\alpha/2}\hat{\sigma}_j}{\sqrt{n}}; \hat{\beta}_j + \frac{q_{1-\alpha/2}\hat{\sigma}_j}{\sqrt{n}}]$$

Les coefficients sélectionnés par LASSO sur les données originales avec lambda = `lambda.1se` sont : season, yr, holiday, weekday, weathersit, temp, atemp, windspeed. Construisons le modèle de régression de Poisson avec ces variables uniquement.

```
# Données avec uniquement les variables sélectionnées
y <- data[, 12]
X_bis <- data[, -c(3,6,10,12)]

# Régression de poisson correspondante
model_poiss <- glm(y ~ . , data = X_bis, family = poisson)

# Esimateur de beta
kable(
  t(model_poiss$coefficients), caption = "Coefficient de l'estimateur du maximum
  de vraisemblance beta.hat",
  format = "latex", booktabs = TRUE, escape = FALSE, align = "r"
) %>%
  kable_styling(latex_options = c("striped", "HOLD_position", "scale_down"), font_size = 9) %>%
  row_spec(0, bold=TRUE)
```

Table 4: Coefficient de l'estimateur du maximum de vraisemblance beta.hat

(Intercept)	season	yr	holiday	weekday	weathersit	temp	atemp	windspeed
7.485732	0.1087586	0.4732415	-0.1677349	0.0147011	-0.1818121	0.1561047	1.178628	-0.4378111

Nous pouvons alors construire les intervalles de confiance pour tous les $\hat{\beta}_j$ au niveau $\alpha = 0.05$ ou 95% :

```
# Coefficients beta et mise en forme
beta_true <- numeric(ncol(X_bis) + 1)
names(beta_true) <- c("(Intercept)", colnames(X_bis))
beta_true[names(coef(model_poiss))] <- coef(model_poiss)
```

```r
# Ecarts-types
se <- summary(model_poiss)$coefficients[, 2]

# Calcul des bornes supérieures et inférieures
n <- dim(data)[1]
borne.inf <- beta_true - qnorm(0.975)*se/sqrt(n)
borne.sup <- beta_true + qnorm(0.975)*se/sqrt(n)

# Résultat
kable(
  t(data.frame(borne.inf=borne.inf,borne.sup=borne.sup)), caption = "Intervalles
  de confiance des variables sélectionnées",
  format = "latex", booktabs = TRUE, escape = FALSE, align = "r"
) %>%
  kable_styling(latex_options = c("striped", "HOLD_position", "scale_down"), font_size = 9) %>%
  row_spec(0, bold=TRUE)
```

Table 5: Intervalles de confiance des variables sélectionnées

	(Intercept)	season	yr	holiday	weekday	weathersit	temp	atemp	windspeed
borne.inf	7.485461	0.1087175	0.473159	-0.1679985	0.0146810	-0.1818922	0.1544097	1.176699	-0.4383770
borne.sup	7.486004	0.1087998	0.473324	-0.1674714	0.0147213	-0.1817319	0.1577997	1.180557	-0.4372453

Nous constatons dans la table 5 que les intervalles de confiances sont tous très ressérés autour de la valeur du coefficient $\hat{\beta}_j$.

2. *Evaluation du niveau empirique.*

Nous souhaitons maintenant évaluer le niveau empirique de ces intervalles de confiance, sachant que le niveau de confiance attendue serait de 95%. Pour cela, nous allons générer 100 fois de nouvelles données simulées à partir du modèle : pour chaque individu i nous simulons une valeur selon une loi de Poisson $\mathcal{P}(\lambda_i)$ avec $\lambda_i = \mathbb{E}[Y_i|X_i]$ l'intensité de de l'individu i. Pour chaque itération, nous ajustons un modèle régularisé en utilisant les données simulées pour obtenir de nouveaux intervalles de confiance pour les variables qui auront été sélectionnées puis nous vérifions si la véritable valeur de $\hat{\beta}_j$ (qui servi à la simulation) se trouve dans l'intervalle. Ce qui nous permettra après les 100 itérations d'obtenir une approximation du niveau empirique de cette approche.

```r
# Nombre de répétition
nrep <- 100

# Compteurs pour la couverture et le nombre d'intervales
compteur_couverture <- setNames(numeric(length(beta_true)), names(beta_true))
nbr_intervales      <- setNames(numeric(length(beta_true)), names(beta_true))

# On récupère les lambda_i pour les n individus
lambda_hat <- predict(model_poiss, type = "response") # pred pour les données originale taille n

for(i in 1:nrep){
  # On simule de nouvelles valeurs pour cnt suivant le modèle, avec lambda_i = l'intensité pour l'obs i =
  set.seed(100+i)
  y_sim <- rpois(n, lambda = lambda_hat)

  # Ajustement d'un modèle LASSO avec lambda = lambda.1se
  cv_i <- cv.glmnet(as.matrix(X_bis),y_sim,family="poisson")
  lambda_i <- cv_i$lambda.1se

  # Récupération du support correspondant
  coef_i <- coef(cv_i, s=lambda_i)
  index_col_select <- which(as.vector(coef_i[-1,1])!=0)

  # Calcul du modèle de poisson, récupération du beta.hat et des intervalles de confiance
  X_act_i <- X_bis[,index_col_select,drop=FALSE]
  model_poiss_i <- glm(y_sim ~ . , data = X_act_i, family = poisson)
```

```r
  # Récupération du beta_i et se_i pour les coefficients actifs
  beta_i <- numeric(ncol(X_act_i) + 1)
  names(beta_i) <- c("(Intercept)", colnames(X_act_i))
  beta_i[names(coef(model_poiss_i))] <- coef(model_poiss_i)
  se <- summary(model_poiss_i)$coefficients[, 2]
  borne.inf_i <- beta_i - qnorm(0.975)*se/sqrt(n)
  borne.sup_i <- beta_i + qnorm(0.975)*se/sqrt(n)

  # Évaluation si coefficient vrai est dans l'intervalle de confiance
  for(name in names(borne.inf_i)){
    beta_true_j <- beta_true[name]
    inf <- borne.inf_i[name]
    sup <- borne.sup_i[name]
    nbr_intervales[name] <- nbr_intervales[name] + 1 # On augmente le compteur d'intervalle de 1
    if (beta_true_j >= inf && beta_true_j <= sup) {
      # Si la vraie valeur de beta est dans l'intervalle de l'itération i
      # on augmente le compteur de 1.
      compteur_couverture[name] <- compteur_couverture[name] + 1
    }
  }
}

niveau_empirique <- compteur_couverture / nbr_intervales
```

```r
kable(
  t(compteur_couverture), caption = "Nombre de fois où le véritable coefficient
  beta.hat.j s'est trouvé dans les intervalles de confiance",
  format = "latex", booktabs = TRUE, escape = FALSE, align = "r"
) %>%
  kable_styling(latex_options = c("striped", "HOLD_position", "scale_down"), font_size = 9) %>%
  row_spec(0, bold=TRUE)
```

Table 6: Nombre de fois où le véritable coefficient beta.hat.j s'est trouvé dans les intervalles de confiance

(Intercept)	season	yr	holiday	weekday	weathersit	temp	atemp	windspeed
7	3	7	8	9	6	5	10	10

```r
kable(
  t(nbr_intervales), caption = "Nombre d'occurence de la sélection des variables",
  format = "latex", booktabs = TRUE, escape = FALSE, align = "r"
) %>%
  kable_styling(latex_options = c("striped", "HOLD_position", "scale_down"), font_size = 9) %>%
  row_spec(0, bold=TRUE)
```

Table 7: Nombre d'occurence de la sélection des variables

(Intercept)	season	yr	holiday	weekday	weathersit	temp	atemp	windspeed
100	100	100	100	100	100	100	100	100

```r
kable(
  t(nbr_intervales), caption = "Niveau empirique moyen pour chaque variable",
  format = "latex", booktabs = TRUE, escape = FALSE, align = "r"
) %>%
  kable_styling(latex_options = c("striped", "HOLD_position", "scale_down"), font_size = 9) %>%
  row_spec(0, bold=TRUE)
```

Table 8: Niveau empirique moyen pour chaque variable

(Intercept)	season	yr	holiday	weekday	weathersit	temp	atemp	windspeed
100	100	100	100	100	100	100	100	100

Nous obtenons un niveau de confiance moyen de 7.22% contre les 100(1-alpha) = 95% qui seraient attendus. Cependant, cette différence peut-être expliquée par le fait que l'hypothèse que le modèle a été choisi sans voir les données est violée dans ce contexte. En effet, nous effectuons une inférence naïve en utilisant des données qui ont été déjà été utilisées afin d'effectuer la sélection des variables avec le LASSO. Cela donne alors lieu à des intervalles de confiance trop optimistes, trop resserrés comme nous l'avions remarqué précédemment et donc qui ne ne contiennent par la véritable valeur du coefficient : l'erreur est bien élevée que ce qui est attendu.

Ce résultat est un bon exemple des défis spécifiques à l'inférence après la sélection de variables : puisque les données ont déjà été utilisées pour la sélection de variables, il n'est plus possible des les utiliser naïvement pour l'inférence sur les coefficients. Il est donc nécessaire d'opter pour une autre approche qui prend en compte cette problématique comme le data-splitting qui consiste à séparer le jeu de données en 2 parties (une pour la sélection des variables et une pour l'inférence) mais qui nécessite d'avoir suffisamment de données ou les méthodes du package selectiveInference que nous avons vu précédemment.