TP 5 - Compte rendu de groupe

Matthias MAZET, Garance MALNOË

2025-12-09

Contents


```
# Packages statistiques
library(glmnet)

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

0.1 *Préparation des données.*

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

Pour simplifier les méthodes, nous supprimons la variable “dteday” qui est au format “chr”. Nous supprimons aussi “casual” et “registered” qui, en les sommant, donne la valeur de la variable à prédire “cnt”. Enfin, nous supprimons la variable “instant” qui correspond simplement à l’index de chaque observation. Nous vérifions ensuite les valeurs manquantes, la dimension des données et les valeurs aberrantes.

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

# Résumés statistiques des variables
kable(
  summary(data[, 1:6], digits = 2), caption = "Résumé statistiques 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 statistiques 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 statistiques 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 statistiques 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

Il n’y a aucune valeurs manquantes (`sum(is.na(data))` $= 0$), aucune variables ne semblent avoir de valeurs aberrantes (d’après les résumés statistiques) et nous sommes dans un cas où $(n = 731) > (p = 12)$, avec n le nombre d’observations et p le nombre de variables. Aussi, la variable à prédire, “cnt”, compte le nombre d’emprunts de vélos réalisés dans une journée. Nous pouvons donc bien appliquer une régression de Poisson pour modéliser une variable de comptage. Dans ce but, nous séparons les données en X les variables explicatives, et y la variable à prédire.

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

0.2 *Estimation et stabilité de l'estimateur.*

Nous ajustons un modèle de Poisson via `glmnet`. Pour cela, nous précisons `family = "poisson"` dans les fonctions de `glmnet`. Nous conservons arbitrairement la valeur de `lambda.1se` obtenue par validation croisée.

```
# 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")
```

Nous obtenons une valeur de lambda.1se = 99.094, et les variabbles sélectionnées avec les données de base et cette valeur sont : season, yr, holiday, weekday, weathersit, temp, atemp, windspeed.

Regardons maintenant la stabilité de sélection de variables sur 100 itérations. Pour chaque itération, nous regardons quelles variables sont sélectionnées au `lambda.1se` exhibé précédemment et nous stockons l'information dans une matrice.

```
set.seed(5)

# 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 <- as.matrix(data[ind_al, colnames(data) != "cnt"])
  y <- data[ind_al, "cnt"]
  # Fit du modèle
  mod_lasso_i <- glmnet(x = X, y = y, alpha = 1, lambda = lambda_1se, 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 = "Indices des 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"
  )
```

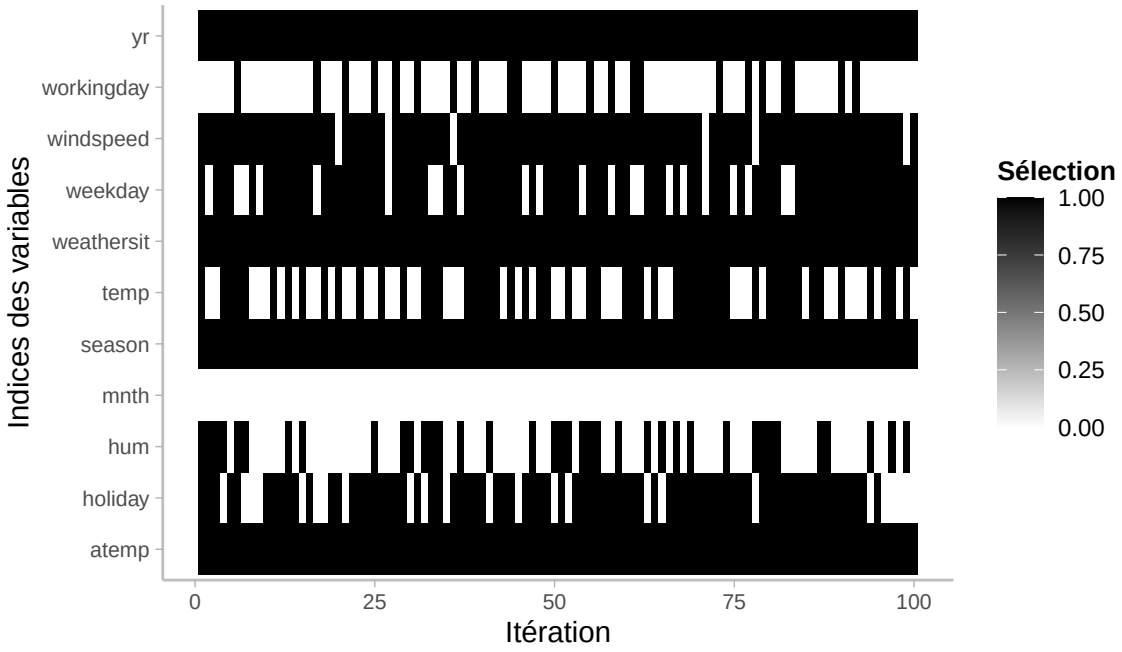

Figure 1: Stabilité à lambda.1se au fil des itérations

```
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	0.76	0.78	0.22	1	0.53	1	0.38	0.94

La figure précédente **(Fig. 1)** présente les variables sélectionnées à chaque itération, avec 1 = variable sélectionnée et 0 sinon (la légende de couleur devrait être discrète mais ggplot ne l'accepte pas sur une heatmap). Nous pouvons constater que certaines variables ("yr" ou "season" par exemple) sont toujours sélectionnées, tandis que d'autres le sont aléatoirement suivant l'itération ("hum" ou "temp" par exemple). La variable "mnth" n'est à l'inverse jamais sélectionnée. La méthode présente donc une légère instabilité sur la sélection de variables, ce qui peut témoigner de multicolinéarité entre certaines variables (notamment celles sélectionnées quelquefois mais pas toutes).

Pour renforcer l'analyse, nous pouvons regarder la stabilité de sélection de variables le long du chemin de régularisation.

```
set.seed(5)

# Noms des variables explicatives
vars <- colnames(data)[colnames(data) != "cnt"]

# Grille de lambda commune à toutes les répétitions.
lambda_grid <- seq(1, 300, by = 0.25)

# Paramètres fixes
n_it <- 100
p <- length(vars)
L <- length(lambda_grid)
n <- nrow(data)
```

```r
# Fonction pour récupérer les variables sélectionnées
get_support <- function(model) {
  as.matrix(coef(model))[-1, , drop = FALSE] != 0  # On enlève l'intercept
}

# Initialisation des vecteurs et matrices de résultats
path_lasso <- array(0, dim = c(p, L, n_it), dimnames = list(vars, lambda_grid, NULL))

for (r in 1:n_it) {
  # Ré-échantillonage
  ind_al <- sample(1:n, size = n, replace = TRUE)
  X <- as.matrix(data[ind_al, colnames(data) != "cnt"])
  y <- data[ind_al, "cnt"]
  # Lasso
  ### fit pour chaque lambda
  lasso_fit <- glmnet(x = X, y = y, alpha = 1, family = "poisson", lambda = lambda_grid)
  ### variables sélectionnées pour chaque fit
  path_lasso[, , r] <- get_support(lasso_fit)
}

# On compte pour chaque variable pour chaque le nombre moyen de fois où elle est incluse
stab_lasso <- apply(path_lasso, c(1,2), mean)
```

```r
# Transformations pour les visualisation
df_lasso <- as.data.frame(stab_lasso) %>%
  mutate(variable = rownames(.)) %>%
  pivot_longer(-variable, names_to = "lambda", values_to = "freq") %>%
  mutate(lambda = as.numeric(lambda))

# Visualisaiton avec ggplot
ggplot(df_lasso, aes(x = lambda, y = variable, fill = freq)) +
  geom_tile() +
  geom_vline(xintercept = lambda_1se, color="#b31e1e", linewidth = .9) +
  scale_fill_gradientn(
    colours = c("white", "black"), limits = c(0, 1), name = "Fréq. de sélection"
  ) +
  labs(x = "lambda", y = "Variable") +
  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"
  )
```

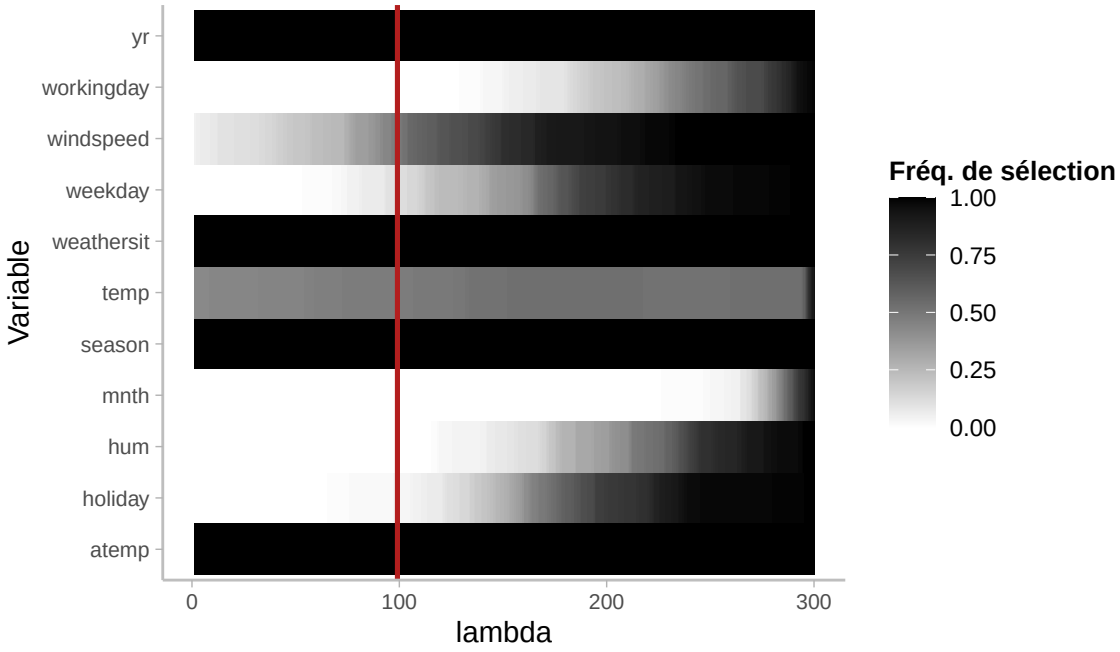

Figure 2: Stabilité sur le chemin de régularisation

Cette figure **(Fig. 2)** nous permet de constater que la valeur de `lambda.1se` exhibée avec les données initiales se généralise assez bien sur des données bootstrap. En effet, cette valeur (ligne rouge) se trouve dans une zone assez stable sur le chemin de régularisation : seules "temp", windspeed", "weekday" et "holiday" sont en grises, et donc sélectionnées aléatoirement suivant l'itération.

0.3 *Sélection post-inférence.*

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

```
coefs.1se <- get_support(mod_lasso)
coefs.1se # A mettre au propre avec kabble
```

```
##                s0
## season       TRUE
## yr           TRUE
## mnth        FALSE
## holiday      TRUE
## weekday      TRUE
## workingday FALSE
## weathersit   TRUE
## temp         TRUE
## atemp        TRUE
## hum         FALSE
## windspeed    TRUE
```

Les coefficients sélectionnés par LASSO sur les données originales avec lambda = lambda.1se sont : season, yr, holiday, weekday, weathsit, temp, atemp et windspeed.

Construisons le modèle de régression de poisson avec ces variables uniquement :

```
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
model_poiss$coefficients
```

```
## (Intercept)      season          yr     holiday     weekday  weathersit
##  7.48573239  0.10875864  0.47324148 -0.16773493  0.01470114 -0.18181207
##        temp       atemp   windspeed
##  0.15610472  1.17862813 -0.43781113
```

Mette commentaire sur les coefficients du beta, mise en forme dynamique.

On peut alors construire des intervalles de confiance pour tous les beta :

```r
# 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)

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

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

# Résultat
data.frame(borne.inf=borne.inf,borne.sup=borne.sup) # a mettre en kabble
```

```
##                borne.inf  borne.sup
## (Intercept)  7.48546080  7.4860040
## season       0.10871751  0.1087998
## yr           0.47315895  0.4733240
## holiday     -0.16799849 -0.1674714
## weekday      0.01468098  0.0147213
## weathersit  -0.18189225 -0.1817319
## temp         0.15440972  0.1577997
## atemp        1.17669898  1.1805573
## windspeed   -0.43837699 -0.4372453
```

2. *Evaluation du niveau empirique.* On a vu en cours que : à développer propre avec le cours.

$$\lambda_i = \mathbb{E}[Y_i|X_i] = exp(x_i^T \hat{\beta})$$

```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
compteur_couverture
```

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

```r
nbr_intervales
```

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

```r
niveau_empirique
```

```
## (Intercept)      season          yr     holiday     weekday  weathersit
##        0.07        0.03        0.07        0.08        0.09        0.06
##        temp       atemp   windspeed
##        0.05        0.10        0.10
```

```r
mean(niveau_empirique)
```

```
## [1] 0.07222222
```

Nous obtenons un niveau de confiance moyen de 7% contre les 100(1-alpha) = 95% qui seraient attendus. Cela est du au fait que nous effectuions une inférence naïve après avoir déjà fait une sélection : nous avons sélectionné les variables actives avec LASSO (qui dépend des données) puis nous faisons une seconde inférence naïve classique qui suppose que le modèle est fixé avant de regarder les données (or on les a déjà vues avec LASSO pour choisir le modèle). Cela donnent lieu à des intervalles de confiance très très resserrés, trop optimiste dans lequel la véritable valeur ne se trouve pas. Le modèle étant choisi via les données, les intervalles de confiance classiques sont fortement resserrés.

Il est donc nécessaire d'opter pour une approche afin d'obtenir des intervalles de confiance : data-splitting (problème c'est qu'on perd une partie des données) ou selectiveInference que nous avons vu en cours.

-> détailler ce qu'est le data-splitting, voir cours. 50% pour sélectionner les variables via LASSO. 50% pour faire l'inférence des intervalles de confiance. -> dire qu'on a déjà testé en TP et qu'on a montré que cela amenait au niveau de confiance attendu.

Le défi c'est d'avoir suffisament de données pour pouvoir faire le data-splitting