Bootstrap

Jawad Boulahfa

06/05/2020

Deuxième partie: bootstrap

```
rm(list = ls())
```

Installation du package nnls2d

```
#devtools::install_github("Jawad-Boulahfa/nnls2d")
```

Chargement des packages

```
# Pour pcls2 et les fonctions de simulation
library(nnls2d, quietly = TRUE)

# Pour le calcul parallèle
library(foreach, quietly = TRUE)
library(iterators, quietly = TRUE)
library(parallel, quietly = TRUE)
library(doParallel, quietly = TRUE)

# Pour ggplot2 et la manipulation des dataframes
library(tidyverse, quietly = TRUE)

# Pour tracer plusieurs graphiques en même temps
library(gridExtra, quietly = TRUE)

# Pour construire des heatmap
library(reshape2, quietly = TRUE)

# Pour construire des heatmap
library(hrbrthemes)
library(plotly)
library(webshot)
```

Initialisation

```
n <- 1000 # Nombre de lignes
sigma <- 1 # Ecart-type du bruit gaussien
beta <- c(0, 1) # Choix du beta

prop <- 0.7 # Proportion de données prises pour rééchantillonner
```

```
replace <- TRUE # On autorise les répétitions dans le rééchantillonnage

B <- 200 # Nombre de fois où on répète le réechantillonnage (pour le bootstrap)

alpha <- 0.05 # Pour les intervalles de confiance à 95%

nb_classes <- 50 # Nombre de classes pour les histogrammes

# Nombre de fois où on refait un bootstrap
# pour construire un nouvel intervalle de confiance
# Autrement dit, on construira 100 IC ici (via 100 bootstrap)
nb_repetitions <- 100
```

Premier essai

On calcule $\hat{\beta}_{nnls}$ et $\hat{\beta}_{lm}$ pour chaque jeu de données rééchantillonné (*nombre de rééchantillonnages* = 200, $n = 1000$, $\sigma = 1, \beta = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$).

```
essai <- nnls2d::beta_list(n = n, sigma = sigma,
                           beta = beta, prop = prop,
                           replace = replace, B = B)
```

On affiche un aperçu des résultats obtenus.

```
print(essai, max = 10)
```

```
## $beta_nnls
##   beta_nnls_1 beta_nnls_2
## 1  0.21741038   0.7405441
## 2  0.07637371   0.8706269
## 3  0.15187386   0.8540611
## 4  0.18999654   0.9091505
## 5  0.15201434   0.7952496
##  [ reached 'max' / getOption("max.print") -- omitted 195 rows ]
##
## $beta_lm
##    beta_lm_1 beta_lm_2
## 1 0.21741038 0.7405441
## 2 0.07637371 0.8706269
## 3 0.15187386 0.8540611
## 4 0.18999654 0.9091505
## 5 0.15201434 0.7952496
##  [ reached 'max' / getOption("max.print") -- omitted 195 rows ]
##
## $comparison_1
##       beta_1 model
## 1 0.21741038  nnls
## 2 0.07637371  nnls
## 3 0.15187386  nnls
## 4 0.18999654  nnls
## 5 0.15201434  nnls
##  [ reached 'max' / getOption("max.print") -- omitted 395 rows ]
##
## $comparison_2
```

```
##      beta_2 model
## 1 0.7405441  nnls
## 2 0.8706269  nnls
## 3 0.8540611  nnls
## 4 0.9091505  nnls
## 5 0.7952496  nnls
##  [ reached 'max' / getOption("max.print") -- omitted 395 rows ]
##
## $biais_df
##              biais
## nnls_1  0.10481914
## lm_1    0.09527102
## nnls_2 -0.07614797
## lm_2   -0.06894340
##
## $var_df
##           variance
## nnls_1 0.007283589
## lm_1   0.010342283
## nnls_2 0.007832891
## lm_2   0.009638693
##
## $MSE_df
##               MSE
## nnls_1 0.01827064
## lm_1   0.01941885
## nnls_2 0.01363141
## lm_2   0.01439189
##
## $IC_df
##           IC_nnls_1    IC_lm_1 IC_nnls_2   IC_lm_2
## Borne inf 0.0000000 -0.1063044 0.7356334 0.7356334
## Borne sup 0.2903657  0.2903657 1.0798324 1.1228030
```

La variable essai contient:

- un dataframe contenant les valeurs de $\hat{\beta}_{nnls_1}$ et $\hat{\beta}_{nnls_2}$
- un dataframe contenant $\hat{\beta}_{lm_1}$ et $\hat{\beta}_{lm_2}$
- un dataframe contenant les valeurs de $\hat{\beta}_{nnls_1}$ et $\hat{\beta}_{lm_1}$ regroupées sur une colonne (et qu'on peut distinguer à l'aide de la colonne "model") un dataframe contenant les valeurs de $\hat{\beta}_{nnls_2}$ et $\hat{\beta}_{lm_2}$ regroupées sur une colonne (et qu'on peut distinguer à l'aide de la colonne "model")
- un dataframe contenant le biais de chaque composante des deux estimateurs.
- un dataframe contenant la variance de chaque composante des deux estimateurs.
- un dataframe contenant l'erreur quadratique moyenne de chaque composante des deux estimateurs
- un dataframe contenant les intervalles de confiance au niveau de confiance 0.95% de chaque composante des deux estimateurs.

Histogrammes et indicateurs

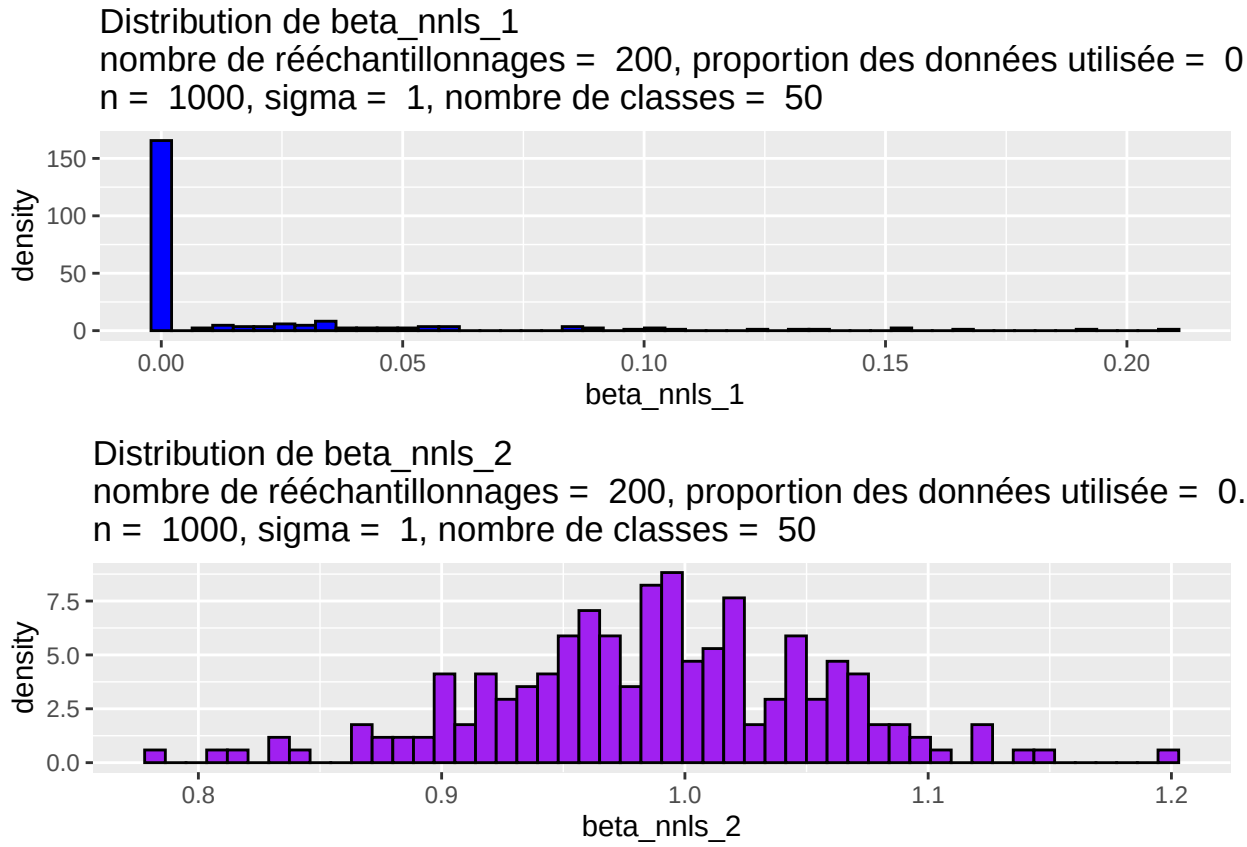
```
distribution <- nnls2d::distribution_beta(
  n = n, sigma = sigma, beta = beta,
```

```
  prop = prop, replace = replace, B = B)
```

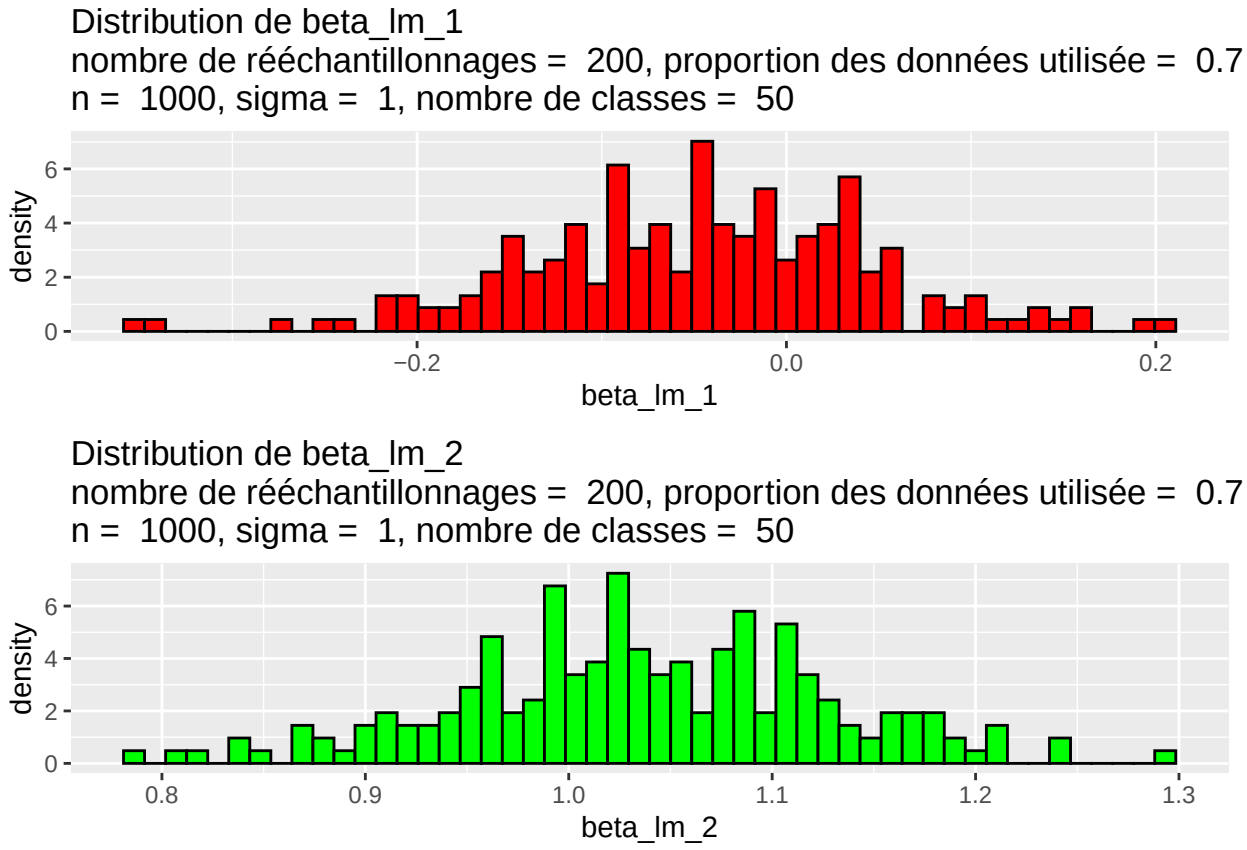
Affichage des 4 histogrammes séparément

On affiche la distribution de chacune des composantes de $\hat{\beta}_{nnls}$ et $\hat{\beta}_{lm}$ ($nombre\ de\ rééchantillonnages = 200,\ n = 1000,\ \sigma = 1$).

```
grid.arrange(distribution$beta_nnls_1_hist,
             distribution$beta_nnls_2_hist,
             nrow = 2, ncol = 1) # OK
```

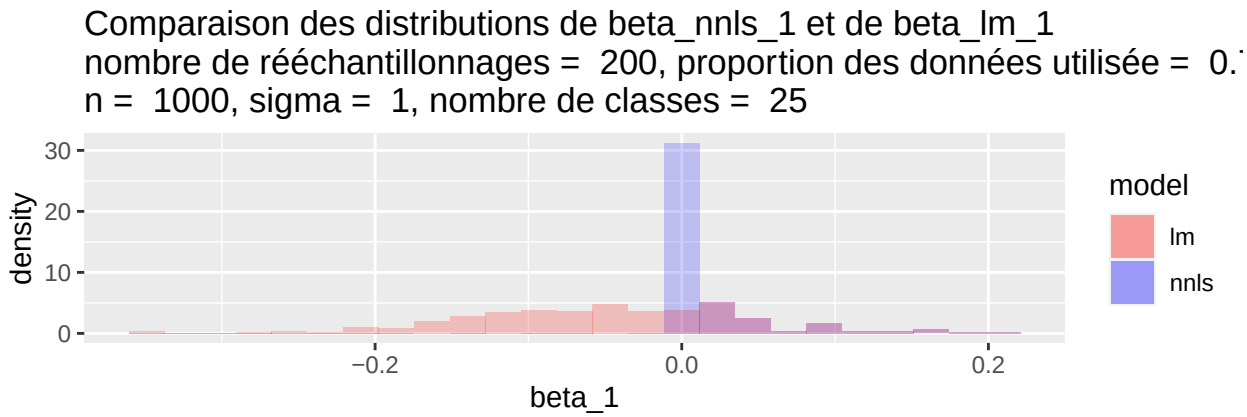


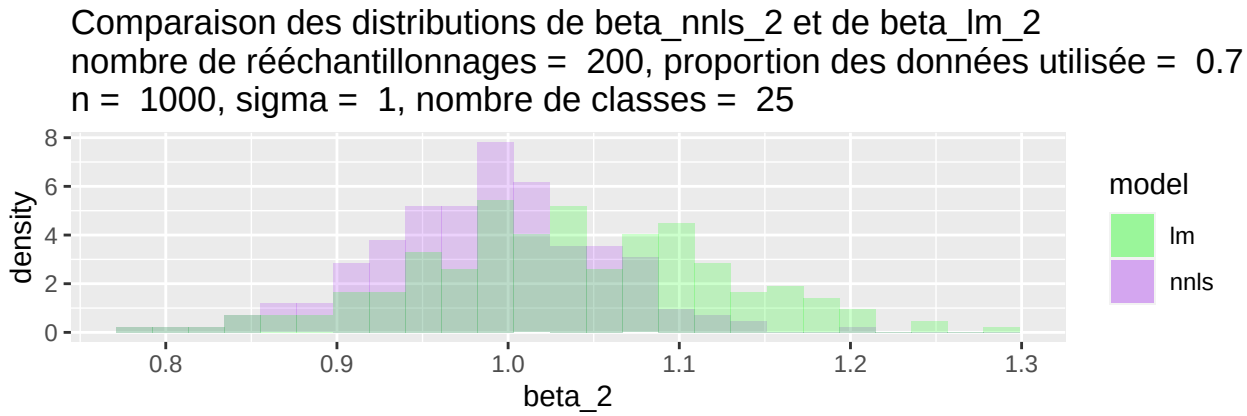

```
grid.arrange(distribution$beta_lm_1_hist,
             distribution$beta_lm_2_hist,
             nrow = 2, ncol = 1) # OK
```



Comparaisons entre les modèles nnls et lm

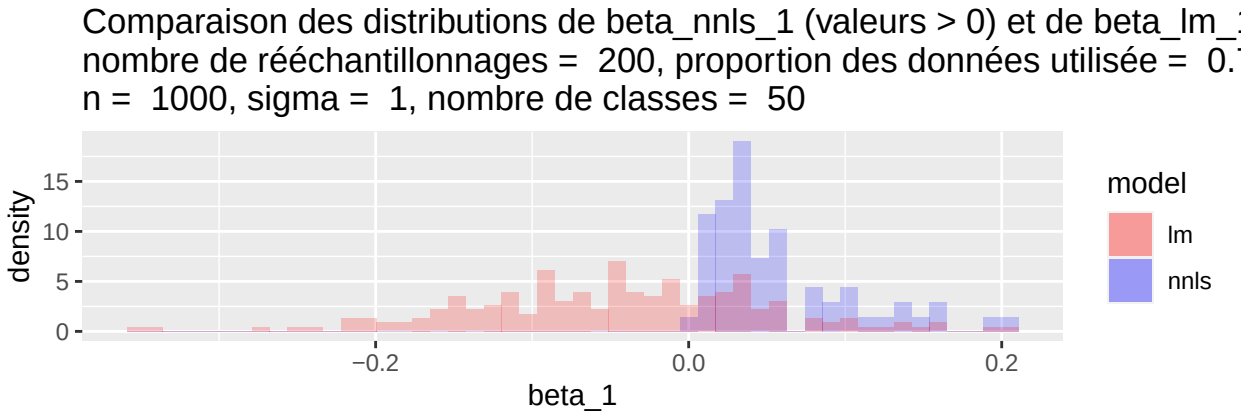
```
grid.arrange(distribution$comparison_1_hist,
             distribution$comparison_2_hist,
             nrow = 2, ncol = 1)
```

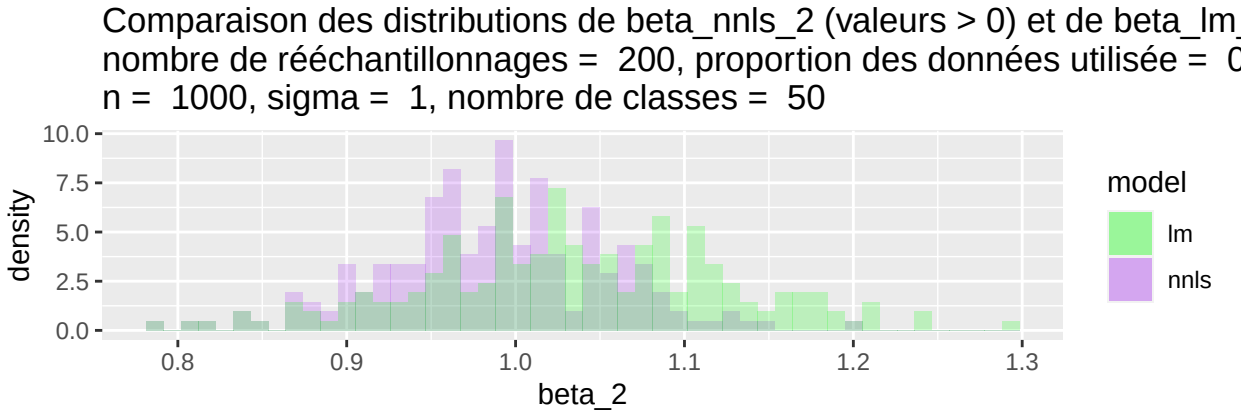




Comparaisons sans le pic en 0 pour nnls

```
grid.arrange(distribution$comparison_1_hist_without_0,
             distribution$comparison_2_hist_without_0,
             nrow = 2, ncol = 1)
```





Biais, variance, erreur quadratique moyenne

On affiche le biais, la variance, et l'erreur quadratique moyenne de chaque composante de chacun des estimateurs.

```
# Biais
print(distribution$biais_df)
```

```
##              biais
## nnls_1  0.01734581
## lm_1   -0.04737290
## nnls_2 -0.01254416
## lm_2    0.03556944
```

```
# Variance
print(distribution$var_df)
```

```
##           variance
## nnls_1 0.001399324
## lm_1   0.008736334
## nnls_2 0.004455683
## lm_2   0.008127677
```

```
# MSE
print(distribution$MSE_df)
```

```
##                MSE
## nnls_1 0.001700202
```

```
## lm_1   0.010980526
## nnls_2 0.004613039
## lm_2   0.009392862
```

On peut rassembler tous ces résultats dans un seul dataframe pour plus de lisibilité.

```
biais_var_MSE_df <- cbind(distribution$biais_df,
                          distribution$var_df,
                          distribution$MSE_df)
```

```
print(biais_var_MSE_df)
```

```
##              biais    variance         MSE
## nnls_1  0.01734581 0.001399324 0.001700202
## lm_1   -0.04737290 0.008736334 0.010980526
## nnls_2 -0.01254416 0.004455683 0.004613039
## lm_2    0.03556944 0.008127677 0.009392862
```

Intervalles de confiance au niveau de confiance 0.95%

Intervalles de confiance pour $\beta = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

```
print(distribution$IC_df)
```

```
##           IC_nnls_1    IC_lm_1 IC_nnls_2   IC_lm_2
## Borne inf 0.0000000 -0.2216390 0.8431888 0.8431888
## Borne sup 0.1346782  0.1346782 1.1224185 1.2063043
```

Rapport d'amplitude des intervalles de confiance $\beta = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

```
coeff_1 <- diff(distribution$IC_df$IC_nnls_1)/
  diff(distribution$IC_df$IC_lm_1)
coeff_2 <- diff(distribution$IC_df$IC_nnls_2)/
  diff(distribution$IC_df$IC_lm_2)
```

```
print(coeff_1)
```

```
## [1] 0.3779728
```

```
print(coeff_2)
```

```
## [1] 0.7689831
```

L'amplitude de IC_nnls_1 vaut 0.3779728 fois celle de IC_lm_1. L'amplitude de IC_nnls_2 vaut 0.7689831 fois celle de IC_lm_2.

Ainsi, les intervalles de confiance obtenus pour chaque composante de $\hat{\beta}_{nnls}$ ont une plus petite amplitude que ceux obtenus pour chaque composante de $\hat{\beta}_{lm}$.

Intervalles de confiance pour $\beta \in \left\{ \begin{pmatrix} 0 \\ k \end{pmatrix} \mid k \in \{0, 1k' \mid 0 \leq k' \leq 10\} \right\}$

On construit une liste de beta.

```
nb_beta <- 11
end <- 1
```

```r
liste_beta_1 <- rep(0, nb_beta)

liste_beta_2 <- seq(from = 0, to = end, by = end/(nb_beta-1))

list_of_beta <- vector("list", length = nb_beta)

for(i in 1:nb_beta)
{
  list_of_beta[[i]] <- c(liste_beta_1[[i]], liste_beta_2[[i]])
}
```

On affiche la liste.

```r
print(list_of_beta)
```

```
## [[1]]
## [1] 0 0
##
## [[2]]
## [1] 0.0 0.1
##
## [[3]]
## [1] 0.0 0.2
##
## [[4]]
## [1] 0.0 0.3
##
## [[5]]
## [1] 0.0 0.4
##
## [[6]]
## [1] 0.0 0.5
##
## [[7]]
## [1] 0.0 0.6
##
## [[8]]
## [1] 0.0 0.7
##
## [[9]]
## [1] 0.0 0.8
##
## [[10]]
## [1] 0.0 0.9
##
## [[11]]
## [1] 0 1
```

On calcule les intervalles de confiance à 0.95%.

```r
liste_IC <- nnls2d::IC_beta(list_of_beta = list_of_beta,
                    n = n, sigma = sigma,
                    prop = prop,
                    replace = replace, B = B) # OK
```

On affiche les intervalles de confiance obtenus.

```r
print(liste_IC)
```

```
## $`beta = (0, 0)`
##           IC_nnls_1    IC_lm_1 IC_nnls_2    IC_lm_2
## Borne inf 0.0000000 -0.1579504 0.0000000 -0.1955646
## Borne sup 0.1151055  0.1791632 0.1057535  0.1644202
##
## $`beta = (0, 0.1)`
##           IC_nnls_1    IC_lm_1  IC_nnls_2   IC_lm_2
## Borne inf 0.0000000 -0.2044479 0.01248661 0.0125410
## Borne sup 0.1949526  0.2047898 0.33623250 0.4027064
##
## $`beta = (0, 0.2)`
##            IC_nnls_1    IC_lm_1 IC_nnls_2     IC_lm_2
## Borne inf 0.00000000 -0.2798026 0.0000000 -0.09061909
## Borne sup 0.09323954  0.1301525 0.1917376  0.31415268
##
## $`beta = (0, 0.3)`
##           IC_nnls_1    IC_lm_1  IC_nnls_2    IC_lm_2
## Borne inf 0.0000000 -0.1399837 0.06763307 0.06763307
## Borne sup 0.2856756  0.2856756 0.39067720 0.43734670
##
## $`beta = (0, 0.4)`
##           IC_nnls_1      IC_lm_1 IC_nnls_2   IC_lm_2
## Borne inf 0.0000000 -0.008535479 0.1423452 0.1423452
## Borne sup 0.3454104  0.345410368 0.4771059 0.4771059
##
## $`beta = (0, 0.5)`
##           IC_nnls_1    IC_lm_1 IC_nnls_2   IC_lm_2
## Borne inf 0.0000000 -0.2304124 0.3954165 0.3956249
## Borne sup 0.2156927  0.2156927 0.6772569 0.7827578
##
## $`beta = (0, 0.6)`
##            IC_nnls_1    IC_lm_1 IC_nnls_2   IC_lm_2
## Borne inf 0.03214508 0.03214508 0.3285991 0.3285991
## Borne sup 0.42520320 0.42520320 0.6874317 0.6874317
##
## $`beta = (0, 0.7)`
##           IC_nnls_1     IC_lm_1 IC_nnls_2   IC_lm_2
## Borne inf 0.0000000 -0.08016169 0.3689108 0.3689108
## Borne sup 0.2477277  0.24772768 0.7412880 0.7694915
##
## $`beta = (0, 0.8)`
##           IC_nnls_1    IC_lm_1 IC_nnls_2   IC_lm_2
## Borne inf 0.0000000 -0.1475325 0.5847603 0.5847603
## Borne sup 0.2361045  0.2361045 0.9541764 1.0299797
##
## $`beta = (0, 0.9)`
##           IC_nnls_1    IC_lm_1 IC_nnls_2   IC_lm_2
## Borne inf 0.0000000 -0.2355302 0.8083052 0.8083052
## Borne sup 0.1919281  0.1919281 1.1156351 1.2240690
##
## $`beta = (0, 1)`
##           IC_nnls_1    IC_lm_1 IC_nnls_2   IC_lm_2
```

```
## Borne inf 0.0000000 -0.2531538 0.8092002 0.8202203
## Borne sup 0.1360439  0.1360439 1.0949030 1.2247598
```

Tests

Tests pour $\beta = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

On effectue 100 bootstrap et on construit un intervalle de confiance à chaque bootstrap. Ensuite, nous effectuons deux tests. On teste $\beta_1 = 0$ contre $\beta_1 \neq 0$ pour les deux modèles en utilisant les intervalles de confiance construits pour $\hat{\beta}_{nnls_1}$ et $\hat{\beta}_{lm_1}$. On teste $\beta_2 = 0$ contre $\beta_2 \neq 0$ pour les deux modèles en utilisant les intervalles de confiance construits pour $\hat{\beta}_{nnls_2}$ et $\hat{\beta}_{lm_2}$.

```
results_test_beta <- nnls2d::test_beta(
  nb_repetitions = nb_repetitions,
  n = n, sigma = sigma, beta = beta,
  prop = prop, replace = replace,
  B = B, alpha = alpha,
  nb_classes = nb_classes)
```

On affiche les résultats obtenus.

```
print(results_test_beta)
```

```
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1              0.02                 1            0.03               1
```

Les deux modèles détectent ici aussi bien la non-nullité de β_2. Néanmoins, le modèle nnls est meilleur pour détecter la nullité de β_1. On va maintenant effectuer davantage de tests en changeant les valeurs de β et commenter les résultats obtenus.

Tests pour $\beta \in \left\{ \begin{pmatrix} 0 \\ k \end{pmatrix} \mid k \in \{0, 1k' \mid 0 \leq k' \leq 10\} \right\}$

On effectue les deux tests évoqués ci-dessus, de la même manière que précédemment, mais pour chaque valeur de β dans la liste donnée en paramètre.

```
results_test_list_of_beta <- nnls2d::test_beta_grid(
  beta_grid = list_of_beta,
  nb_repetitions = nb_repetitions,
  n = n, sigma = sigma, prop = prop,
  replace = replace, B = B,
  alpha = alpha, nb_classes = nb_classes)
```

On affiche les résultats des tests.

```
print(results_test_list_of_beta)
```

```
## $`beta = (0, 0)`
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1                 0                 0            0.03            0.01
##
## $`beta = (0, 0.1)`
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1              0.02               0.1            0.03            0.16
##
## $`beta = (0, 0.2)`
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1                 0              0.55               0            0.57
```

```
##
## $`beta = (0, 0.3)`
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1                 0              0.91            0.01            0.91
##
## $`beta = (0, 0.4)`
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1              0.02              0.99            0.03            0.99
##
## $`beta = (0, 0.5)`
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1              0.01                 1            0.04               1
##
## $`beta = (0, 0.6)`
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1                 0                 1            0.01               1
##
## $`beta = (0, 0.7)`
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1              0.01                 1            0.05               1
##
## $`beta = (0, 0.8)`
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1                 0                 1            0.01               1
##
## $`beta = (0, 0.9)`
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1                 0                 1            0.01               1
##
## $`beta = (0, 1)`
##   freq_rejet_nnls_1 freq_rejet_nnls_2 freq_rejet_lm_1 freq_rejet_lm_2
## 1              0.01                 1               0               1
```

On remarque que la fréquence de rejet obtenue avec le modèle nnls est toujours inférieure ou égale à celle obtenue avec le modèle lm.

C'est une bonne chose lorsque $\beta = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, car cela montre que le modèle nnls détecte mieux la nullité des deux coefficients.

Sur les autres valeurs de β, comme on a gardé $\beta_1 = 0$, cela montre que le modèle nnls détecte également mieux la nullité d'un seul coefficient.

Néanmoins, cela signifie aussi que le modèle nnls détecte moins bien la non-nullité d'un coefficient.

En effet, ici, on souhaiterait que, lorsque $\beta_2 \neq 0$, la fréquence de rejet obtenue avec le modèle nnls soit proche voire égale à 1, et supérieure ou égale à celle obtenue avec le modèle lm.

Si la fréquence de rejet obtenue avec le modèle nnls augmente bien jusqu'à finalement atteindre 1, elle augmente moins vite que celle obtenue avec le modèle lm.

Sauvegarde des résultats

```r
save.image(file = "bootstrap_results.RData")
```

Chargement des résultats

```
rm(list = ls())
```

```
load(file = "bootstrap_results.RData")
```

```
rm(list = ls())
```