



Régression Bayesienne - Travaux Pratiques

Analyse de Satisfaction des Abonnés d'une Chaîne Câblée

Ali ABDELWAHID

Toussaint BOCO

2026-02-03

## Table des matières

## Liste des tableaux

## Liste des figures

# Introduction

## 1.1 Contexte de l'étude

L'objectif de cette étude est d'analyser les déterminants de la satisfaction des abonnés d'une chaîne câblée, à partir d'un jeu de données comprenant 150 individus et 160 variables explicatives décrivant leurs habitudes de consommation télévisuelle.

## 1.2 Problématique

La chaîne souhaite identifier quels types de programmes influencent le plus la satisfaction de ses clients. Les données disponibles incluent :

- **Variable réponse** : score de satisfaction (continu, valeurs négatives = insatisfaction, positives = satisfaction).
- **Variables explicatives ( $p = 160$ )** : temps passé et nombre de visites sur différentes chaînes, normalisés.
- **Variable additionnelle** : sexe de l'abonné (1 = homme, 0 = femme).

Les chaînes sont regroupées en plusieurs catégories : Films, Séries, Sports, Sciences/Santé/Économie, Actualités/Politique, Musique, Jeux, Histoire/Géographie/Documentaires et Divers.

## 1.3 Objectifs

L'étude poursuit trois objectifs principaux :

1. **Prédiction** : construire des modèles capables de prédire le score de satisfaction.
2. **Sélection de variables** : identifier les chaînes les plus influentes.
3. **Comparaison de méthodes** : évaluer les performances de quatre approches bayésiennes.

## 1.4 Méthodologie

Quatre méthodes de régression bayésienne sont comparées :

- **RR-BLUP** : coefficients aléatoires avec variance commune.
- **Bayes A** : variance spécifique à chaque coefficient (shrinkage adaptatif).
- **LASSO bayésien** : régularisation L1 favorisant la parcimonie.
- **SSVS** : sélection stochastique de variables via un modèle hiérarchique.

Chaque méthode est évaluée selon :

- la qualité prédictive (corrélation prédictions/observations),

- la capacité à sélectionner des variables pertinentes,
- l'intensité du shrinkage appliqué aux coefficients.

## 1.5 Organisation du rapport

- **Parties 1 à 4** : présentation et analyse détaillée de chaque méthode.
- **Partie 5** : comparaison globale et synthèse.
- **Partie 12** : Approche ABC (Approximate Bayesian Computation).

## 1.6 Reproductibilité

La division apprentissage/test utilise un **seed fixé à 123**, garantissant la reproductibilité des résultats.

## Préparation des données

```
# Chargement des packages nécessaires
library(tidyverse)      # Manipulation de données
library(BGLR)            # Pour BayesA, BayesB, BL
library(rrBLUP)           # Pour mixed.solve (RR-BLUP)
library(glmnet)            # Pour comparaison LASSO fréquentiste
library(gridExtra)         # Pour affichage multiple de graphiques
library(knitr)             # Pour tableaux
library(kableExtra)        # Pour mise en forme tableaux

# Configuration graphique
theme_set(theme_minimal())
```

```
# Chargement des données
data <- read.csv("data/telecat.csv")
```

## Exploration des données (Structure des variables explicatives)

```
# Affichage structure
str(data)

## 'data.frame':   150 obs. of  163 variables:
## $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
## $ Y          : num  7.21 5.74 3.26 -10.46 17.95 ...
## $ Film.1     : num  0.49461 -1.24696 -0.83972 -0.63139 -0.00425 ...
## $ Film.2     : num  0.87 -1.696 -0.858 -0.214 -1.052 ...
## $ Film.3     : num  -0.6186 -0.4461 -0.9795 0.0818 -2.4242 ...
## $ Film.4     : num  -0.68006 0.72266 0.24998 -0.00788 -0.56206 ...
## $ Film.5     : num  0.2576 -0.318 -0.3146 -0.1797 -0.0369 ...
## $ Film.6     : num  0.179 -1.484 0.388 -0.485 -0.701 ...
## $ Film.7     : num  -0.00395 1.42942 0.28877 -0.78901 -1.22068 ...
## $ Film.8     : num  -0.819 -1.058 0.333 0.282 1.829 ...
## $ Film.9     : num  0.695 -1.055 -0.796 -0.572 -1.271 ...
## $ Film.10    : num  0.182 1.185 -0.513 -0.273 -0.701 ...
## $ Film.11    : num  0.858 0.473 0.615 -0.66 -0.488 ...
## $ Film.12    : num  1.777 0.341 -0.669 0.251 -0.769 ...
## $ Film.13    : num  0.509 -0.806 -1.299 -0.49 0.105 ...
## $ Film.14    : num  0.3764 -0.1256 0.4472 -0.0144 -1.3868 ...
## $ Film.15    : num  -0.984 -0.381 0.201 0.926 -1.058 ...
## $ Film.16    : num  1.3965 -0.2307 0.0295 -0.0272 -0.3937 ...
## $ Film.17    : num  0.638 -0.319 -0.487 0.369 0.457 ...
## $ Film.18    : num  0.17 0.408 -1.698 0.738 -1.056 ...
## $ Film.19    : num  1.186 -1.515 -0.763 0.111 0.644 ...
## $ Film.20    : num  -0.175 -1.456 1.165 -0.828 -0.185 ...
## $ Serie.1    : num  1.00222 0.0641 0.00856 1.01486 -0.16792 ...
## $ Serie.2    : num  -0.769 0.184 -1.611 0.978 -0.955 ...
```

```

## $ Serie.3 : num 1.108 -0.307 -1.362 -1.693 -1.092 ...
## $ Serie.4 : num 0.984 -0.237 -1.878 -0.236 -1.156 ...
## $ Serie.5 : num -0.5542 0.0203 0.9964 0.905 0.5034 ...
## $ Serie.6 : num -1.397 1.201 -1.204 -0.807 0.321 ...
## $ Serie.7 : num -0.104 1.567 0.571 -0.696 0.146 ...
## $ Serie.8 : num 0.6835 1.7272 -0.4346 -0.765 0.0661 ...
## $ Serie.9 : num 0.48 -0.164 0.294 -0.607 1.267 ...
## $ Serie.10 : num 0.83 0.202 -0.97 -0.407 1.56 ...
## $ Serie.11 : num 0.171 1.752 -2.033 0.768 0.67 ...
## $ Serie.12 : num -1.1567 0.0407 -2.5727 -1.0088 -1.3291 ...
## $ Serie.13 : num 0.836 -0.644 -0.903 -0.965 -0.776 ...
## $ Serie.14 : num 0.0325 -0.5045 -0.3761 -0.7563 0.6499 ...
## $ Serie.15 : num -0.701 -1.292 -0.142 -0.807 -1.561 ...
## $ Serie.16 : num -0.242 -0.511 -1.81 -1.147 0.062 ...
## $ Serie.17 : num 1.1176 -0.0819 -0.7465 -2.1461 -0.9802 ...
## $ Serie.18 : num -0.15 1.886 -0.265 1.777 1.384 ...
## $ Serie.19 : num 0.414 -0.162 -0.979 1.603 1.285 ...
## $ Serie.20 : num -0.3764 -1.1402 -0.2213 -1.4013 -0.0315 ...
## $ Sport.1 : num 0.863 0.478 0.459 1.258 -0.322 ...
## $ Sport.2 : num -0.489 0.782 0.144 -0.332 1.162 ...
## $ Sport.3 : num -0.0146 -0.7892 -2.4108 -0.8269 -0.3674 ...
## $ Sport.4 : num 1.762 2.651 0.879 0.192 1.615 ...
## $ Sport.5 : num 0.268 0.577 0.212 -0.897 -1.45 ...
## $ Sport.6 : num 0.229 0.22 -1.384 -0.654 -0.251 ...
## $ Sport.7 : num -0.204 -0.645 0.118 1.837 0.196 ...
## $ Sport.8 : num 1.416 0.347 -0.601 0.932 1.344 ...
## $ Sport.9 : num 0.5965 0.1926 -1.0927 -0.0786 0.9962 ...
## $ Sport.10 : num 0.0389 0.1277 0.6056 -0.5538 0.9177 ...
## $ Sport.11 : num -0.3677 0.0724 -0.3188 -0.6862 0.6776 ...
## $ Sport.12 : num 2.6037 0.2726 -0.4903 0.0843 0.5367 ...
## $ Sport.13 : num 0.0519 0.7161 1.0596 1.7147 0.474 ...
## $ Sport.14 : num -1.097 1.403 -0.971 1.129 -1.09 ...
## $ Sport.15 : num 0.563 1.762 1.437 0.22 1.103 ...
## $ Sport.16 : num 0.5547 0.9002 -0.7905 -1.0453 0.0655 ...
## $ Sport.17 : num 0.403 -0.07 0.692 -2.093 -0.396 ...
## $ Sport.18 : num -0.78607 0.09175 -0.00281 -0.52157 0.25476 ...
## $ Sport.19 : num 0.0303 -0.9201 -0.1999 -1.0747 0.7014 ...
## $ Sport.20 : num 1.638 -0.949 -0.275 -0.669 0.564 ...
## $ Science.1 : num 1.059 -1.381 0.693 0.955 0.747 ...
## $ Science.2 : num 1.3331 -1.2563 1.0982 1.4832 0.0954 ...
## $ Science.3 : num 1.0785 -0.1052 0.1756 -0.0412 1.3607 ...
## $ Science.4 : num -0.137 -1.305 0.881 1.887 0.427 ...
## $ Science.5 : num 1.012 -0.232 2.134 2.298 0.623 ...
## $ Science.6 : num 1.579 0.321 0.319 0.338 2.817 ...
## $ Science.7 : num 0.9354 -0.7605 1.5736 -0.0734 0.0359 ...
## $ Science.8 : num 1.614 -0.35 1.168 0.314 -0.908 ...
## $ Science.9 : num 0.583 -1.189 1.394 0.596 1.275 ...
## $ Science.10: num 0.2115 0.961 0.8817 0.0523 3.743 ...
## $ Science.11: num 1.467 -0.334 0.827 1.838 2.705 ...
## $ Science.12: num 1.393 -1.174 1.082 0.862 -0.137 ...

```

```

## $ Science.13: num  2.016 -0.5 0.485 1.234 -0.993 ...
## $ Science.14: num  0.887 0.332 -0.207 0.252 0.346 ...
## $ Science.15: num  0.665 -1.144 -0.102 2.061 -0.732 ...
## $ Science.16: num  0.685 0.668 -0.398 -0.612 0.196 ...
## $ Science.17: num  -0.799 0.41 -1.0295 -0.0651 -1.1654 ...
## $ Science.18: num  1.251 -0.104 0.423 1.967 -2.123 ...
## $ Science.19: num  1.8 0.671 -1.597 -0.825 -0.349 ...
## $ Science.20: num  0.267 1.738 0.35 -0.557 1.302 ...
## $ Divers.1 : num  0.6938 -0.3887 0.0326 0.3006 0.5431 ...
## $ Divers.2 : num  -0.367 -0.204 0.264 0.386 1.3 ...
## $ Divers.3 : num  0.876 0.572 -0.644 0.786 0.299 ...
## $ Divers.4 : num  -1.533 -0.146 -0.469 0.02 -0.534 ...
## $ Divers.5 : num  -0.9282 -1.0812 0.0527 -0.1906 -0.1975 ...
## $ Divers.6 : num  -0.471 1.962 0.434 -1.175 0.113 ...
## $ Divers.7 : num  -0.632 0.581 0.052 -1.271 1.127 ...
## $ Divers.8 : num  1.769 0.125 0.468 -0.929 -0.874 ...
## $ Divers.9 : num  0.985 -0.359 0.759 0.984 0.807 ...
## $ Divers.10: num  -0.3031 0.2537 1.0337 -0.0845 0.6224 ...
## $ Divers.11: num  -0.696 -0.61 1.015 0.789 0.303 ...
## $ Divers.12: num  -1.497 1.086 -1.773 1.403 0.342 ...
## $ Divers.13: num  0.0594 1.7534 -0.2674 0.8883 0.9458 ...
## $ Divers.14: num  0.286 -0.757 0.58 1.041 0.877 ...
## $ Divers.15: num  1.6561 1.1293 -0.4079 -0.0434 0.5973 ...
## $ Divers.16: num  -1.585 -0.849 0.804 -0.196 0.324 ...
## $ Divers.17: num  0.845 0.831 -1.3028 0.4825 0.0856 ...
## [list output truncated]

```

Les 162 variables explicatives correspondent à des mesures normalisées d'exposition aux chaînes. Leur normalisation préalable est essentielle pour les méthodes bayésiennes de shrinkage, qui sont sensibles aux différences d'échelle. La forte dimension ( $p = 162$ ) par rapport au nombre d'observations ( $n = 150$ ) justifie pleinement l'usage de méthodes régularisées.

## Distribution du score de satisfaction

```

# Statistiques descriptives de Y
summary_Y <- data.frame(
  Statistique = c("Moyenne", "Écart-type", "Minimum", "Q1", "Médiane", "Q3",
                 "Maximum"),
  Valeur = c(
    mean(data$Y),
    sd(data$Y),
    min(data$Y),
    quantile(data$Y, 0.25),
    median(data$Y),
    quantile(data$Y, 0.75),
    max(data$Y)
  )
)

```

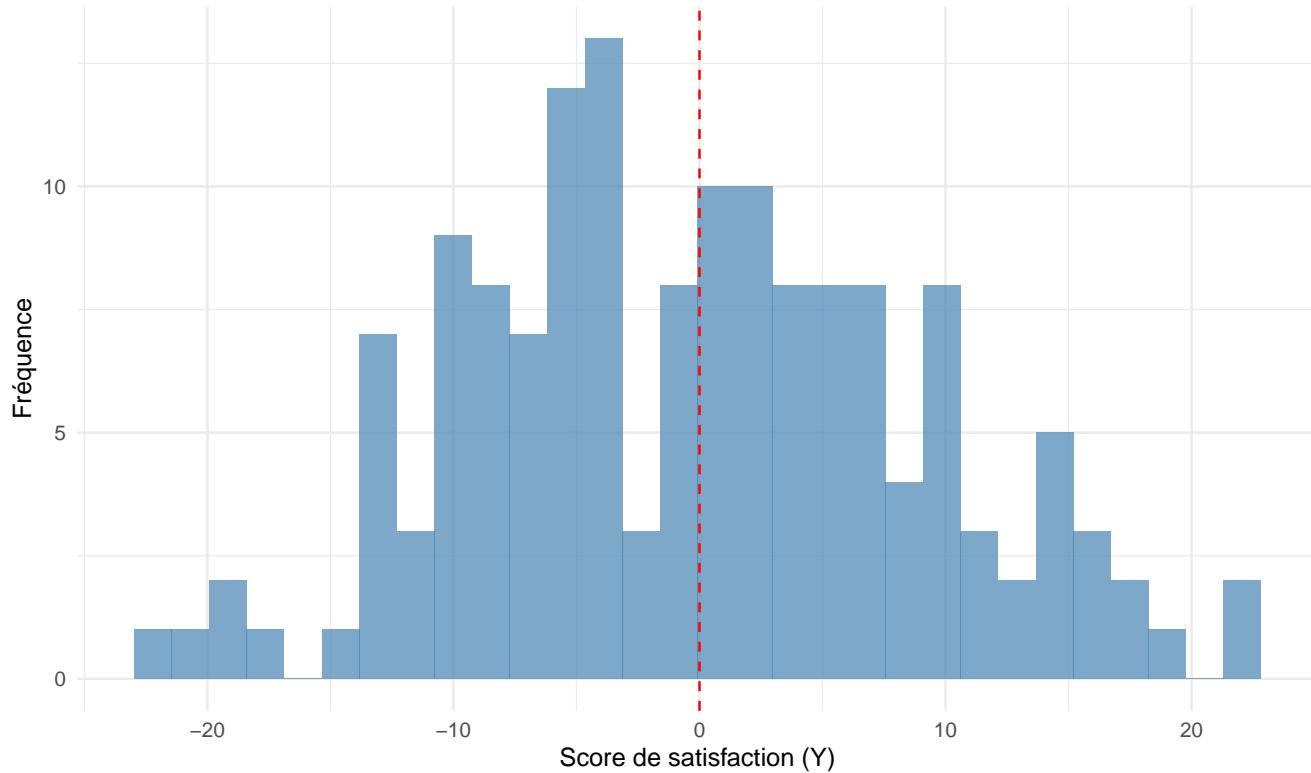
```
kable(summary_Y, digits = 3,
      caption = "Statistiques descriptives du score de satisfaction")
```

Table 1: Statistiques descriptives du score de satisfaction

Statistique	Valeur
Moyenne	-0.172
Écart-type	9.154
Minimum	-22.226
Q1	-6.461
Médiane	-0.281
Q3	6.108
Maximum	22.066

```
# Distribution de Y
ggplot(data, aes(x = Y)) +
  geom_histogram(bins = 30, fill = "steelblue", alpha = 0.7) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Distribution du score de satisfaction",
       x = "Score de satisfaction (Y)",
       y = "Fréquence") +
  theme_minimal()
```

Distribution du score de satisfaction



La distribution du score de satisfaction présente une forte variabilité, avec une moyenne proche de

zéro ( $-0.17$ ) et un écart-type élevé (9.15). Les valeurs s'étendent de  $-22.2$  à  $22.1$ , ce qui indique une hétérogénéité marquée dans la perception des abonnés. La médiane légèrement négative suggère une tendance générale à une satisfaction modérée voire faible. La présence de valeurs extrêmes, visibles dans l'histogramme, confirme un comportement potentiellement non symétrique et justifie l'usage de méthodes robustes ou régularisées.

## Division train/test

```
# Définition du seed pour reproductibilité
set.seed(123) # SEED À NOTER DANS LE RAPPORT

# Division aléatoire
n <- nrow(data)
train_indices <- sample(1:n, size = 100, replace = FALSE)
test_indices <- setdiff(1:n, train_indices)

# Création des jeux de données
train <- data[train_indices, ]
test <- data[test_indices, ]

# Séparation Y et X
Y_train <- train$Y
Y_test <- test$Y

# Identification des colonnes de variables explicatives
# (toutes sauf Y et éventuellement Sexe si on l'exclut pour l'instant)
X_cols <- setdiff(names(data), c("Y"))
X_train <- as.matrix(train[, X_cols])
X_test <- as.matrix(test[, X_cols])

cat("Taille du jeu d'entraînement:", nrow(train), "\n")

## Taille du jeu d'entraînement: 100
cat("Taille du jeu de test:", nrow(test), "\n")

## Taille du jeu de test: 50
cat("Nombre de variables explicatives:", ncol(X_train), "\n")

## Nombre de variables explicatives: 162
```

La séparation aléatoire en 100 observations d'apprentissage et 50 observations de test garantit une base suffisante pour estimer les modèles tout en conservant un échantillon indépendant pour évaluer la performance prédictive. Le seed fixé à 123 assure la reproductibilité des résultats.

# 1 Random Regression (RR-BLUP)

## Rappel théorique

Le modèle **Random Regression** est le modèle bayésien le plus simple avec coefficients aléatoires :

$$\begin{cases} Y = \mu \mathbf{1} + X\beta + \epsilon \\ \beta \sim \mathcal{N}(0, \sigma_\beta^2 I) \\ \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I) \end{cases}$$

**Caractéristiques :**

- Tous les coefficients  $\beta_j$  partagent la **même variance**  $\sigma_\beta^2$
- Pas d'hyperparamètre : variances constantes
- Estimation via **BLUP** (Best Linear Unbiased Predictor)
- Estimation des variances via **REML** (Restricted Maximum Likelihood)

### 1.1 Estimation des paramètres

```
# Ajustement du modèle RR-BLUP avec rrBLUP::mixed.solve
rr_model <- mixed.solve(Y_train, Z = X_train, method = "REML")

# Récupération des estimations
mu_hat_rr <- rr_model$beta # Intercept
beta_hat_rr <- rr_model$u # Coefficients
sigma2_e_rr <- rr_model$Ve # Variance résiduelle
sigma2_beta_rr <- rr_model$Vu # Variance des beta

cat("Estimation de mu:", mu_hat_rr, "\n")

## Estimation de mu: -1.411988
cat("Variance résiduelle (sigma^2_epsilon):", sigma2_e_rr, "\n")

## Variance résiduelle (sigma^2_epsilon): 69.09079
cat("Variance commune (sigma^2_beta):", sigma2_beta_rr, "\n")

## Variance commune (sigma^2_beta): 6.90908e-08

# Statistiques sur les beta estimés
summary_beta_rr <- data.frame(
  Statistique = c("Moyenne", "Écart-type", "Min", "Q1", "Médiane", "Q3", "Max"),
  Valeur = c(
    mean(beta_hat_rr),
    sd(beta_hat_rr),
    min(beta_hat_rr),
    quantile(beta_hat_rr, 0.25),
    median(beta_hat_rr),
    ...))
```

```

    quantile(beta_hat_rr, 0.75),
    max(beta_hat_rr)
)
)

kable(summary_beta_rr, digits = 4,
      caption = "Statistiques descriptives des coefficients estimés (RR-BLUP)")

```

Table 2: Statistiques descriptives des coefficients estimés (RR-BLUP)

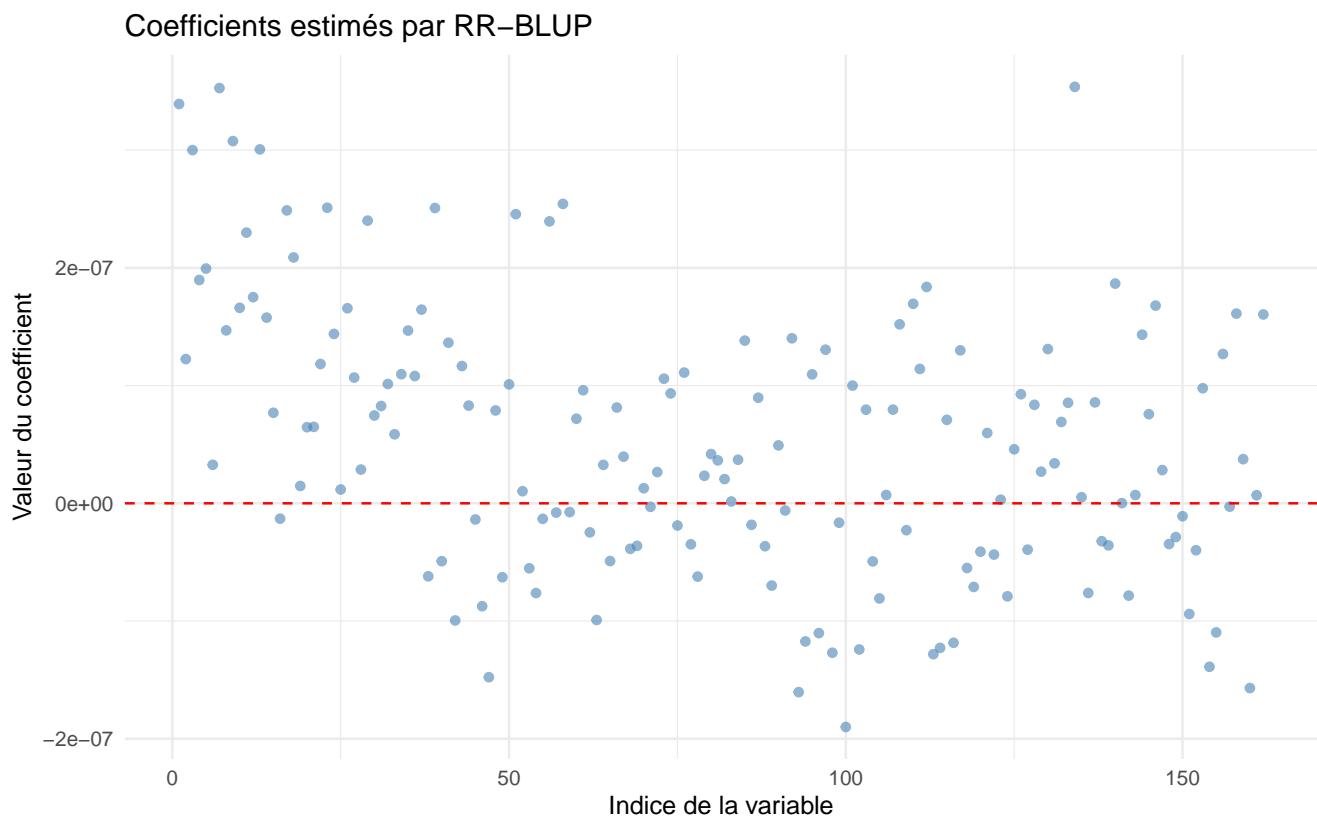
Statistique	Valeur
Moyenne	0
Écart-type	0
Min	0
Q1	0
Médiane	0
Q3	0
Max	0

```

# Visualisation des coefficients
beta_df_rr <- data.frame(
  index = 1:length(beta_hat_rr),
  beta = beta_hat_rr
)

ggplot(beta_df_rr, aes(x = index, y = beta)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Coefficients estimés par RR-BLUP",
       x = "Indice de la variable",
       y = "Valeur du coefficient") +
  theme_minimal()

```



Les coefficients estimés par RR-BLUP sont extrêmement proches de zéro, avec une variance commune très faible ( $\approx 6.9 \times 10^{-8}$ ). Cela traduit un shrinkage massif et uniforme appliqué à l'ensemble des variables. Le modèle considère donc que toutes les chaînes ont une influence très faible et comparable sur la satisfaction.

## 1.2 Prédiction et évaluation

```
# Prédictions sur le jeu de test
mu_hat_rr <- as.numeric(mu_hat_rr) #Conversion en numérique
Y_pred_rr <- mu_hat_rr + X_test %*% beta_hat_rr

# Calcul de la corrélation
cor_rr <- cor(Y_test, Y_pred_rr)

cat("Corrélation prédictions/observations (test):", round(cor_rr, 4), "\n")

## Corrélation prédictions/observations (test): 0.1216
```

La corrélation de 0.122 entre prédictions et observations révèle une **performance très faible** du modèle RR-BLUP, expliquant seulement 1.5% de la variance des scores de satisfaction ( $R^2 = 0.015$ ). Ce résultat décevant s'explique par plusieurs facteurs :

- Shrinkage extrême** : La variance commune estimée ( $\sigma_\beta^2 \approx 6.9 \times 10^{-8}$ ) est quasi-nulle, entraînant un shrinkage drastique de tous les coefficients vers zéro. Le modèle prédit essentiellement la moyenne globale  $\hat{\mu} = -1.4$  pour toutes les observations.
- Inadéquation de l'hypothèse** : Le modèle RR-BLUP impose que toutes les variables partagent la

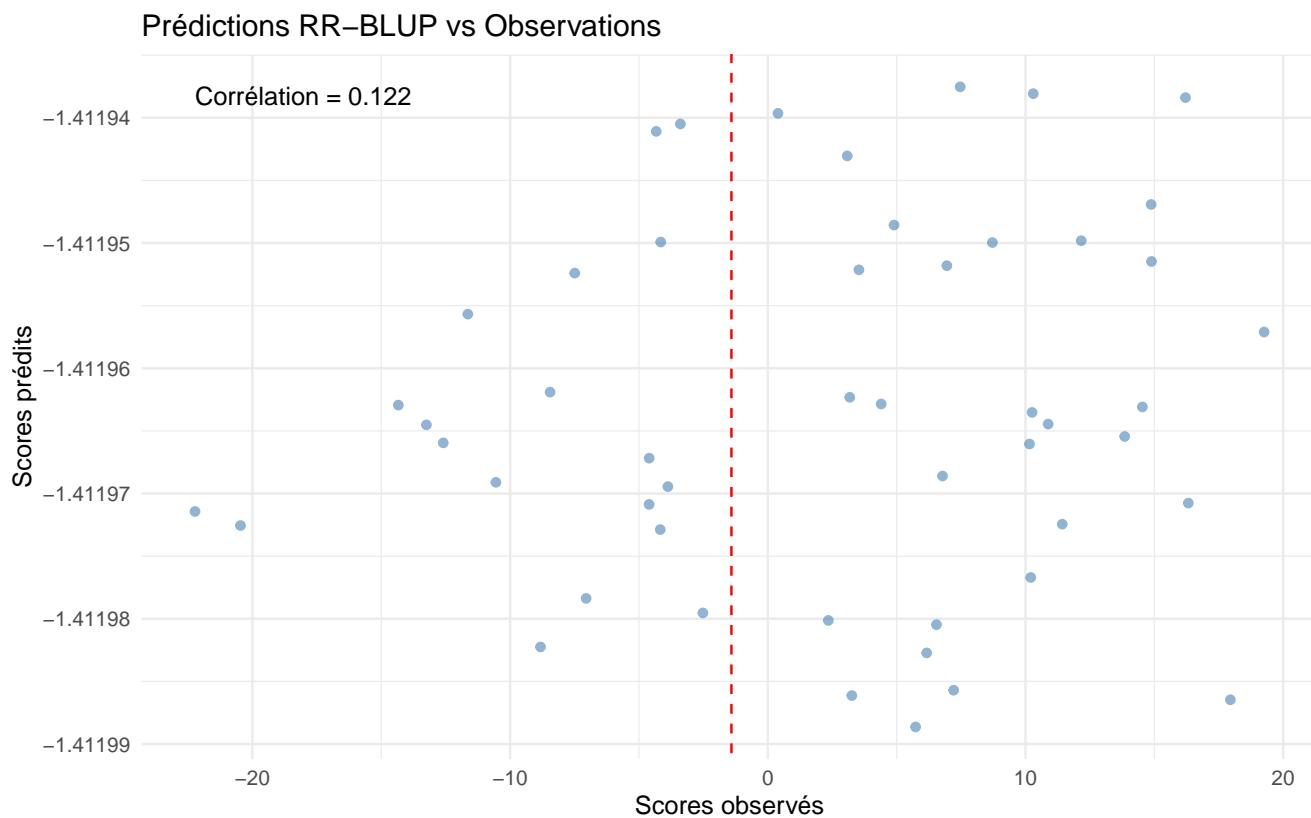
même variance, ce qui est manifestement inadapté à notre contexte. Les 160 chaînes n'ont clairement pas toutes la même influence sur la satisfaction.

3. **Problème  $p \gg n$**  : Avec 162 variables pour 100 observations d'entraînement, le modèle souffre d'un sur-paramétrage sévère. La régularisation REML pousse la variance commune vers zéro pour éviter le sur-ajustement.
4. **Absence de sélection** : RR-BLUP ne sélectionne aucune variable, se contentant de réduire uniformément tous les coefficients. Il ne peut donc pas identifier les chaînes réellement influentes.

**Conclusion** : Cette performance servira de **référence inférieure** pour évaluer les méthodes bayésiennes plus sophistiquées (Bayes A, LASSO, SSVS) qui permettent des variances hétérogènes et une véritable sélection de variables.

```
# Graphique prédictions vs observations
pred_df_rr <- data.frame(
  Observed = Y_test,
  Predicted = as.vector(Y_pred_rr)
)

ggplot(pred_df_rr, aes(x = Observed, y = Predicted)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
  annotate("text", x = min(Y_test), y = max(Y_pred_rr),
           label = paste("Corrélation =", round(cor_rr, 3)),
           hjust = 0, vjust = 1) +
  labs(title = "Prédictions RR-BLUP vs Observations",
       x = "Scores observés",
       y = "Scores prédits") +
  theme_minimal()
```



Le graphique de prédiction montre que toutes les valeurs prédictes sont concentrées autour de  $\hat{y} \approx -1.41$ , formant une **ligne horizontale** insensible aux valeurs observées. Cela confirme que le modèle n'a appris aucune relation entre les variables explicatives et la satisfaction, se contentant de prédire systématiquement la moyenne d'entraînement.

### 1.3 Sélection de variables

```
# Sélection basée sur le boxplot (méthode des outliers)
Q1 <- quantile(abs(beta_hat_rr), 0.25)
Q3 <- quantile(abs(beta_hat_rr), 0.75)
IQR <- Q3 - Q1
seuil_rr <- Q3 + 1.5 * IQR

# Variables sélectionnées
selected_vars_rr <- which(abs(beta_hat_rr) > seuil_rr)
n_selected_rr <- length(selected_vars_rr)

cat("Nombre de variables sélectionnées:", n_selected_rr, "\n")

## Nombre de variables sélectionnées: 6
cat("Seuil de sélection (|beta|):", seuil_rr, "\n")

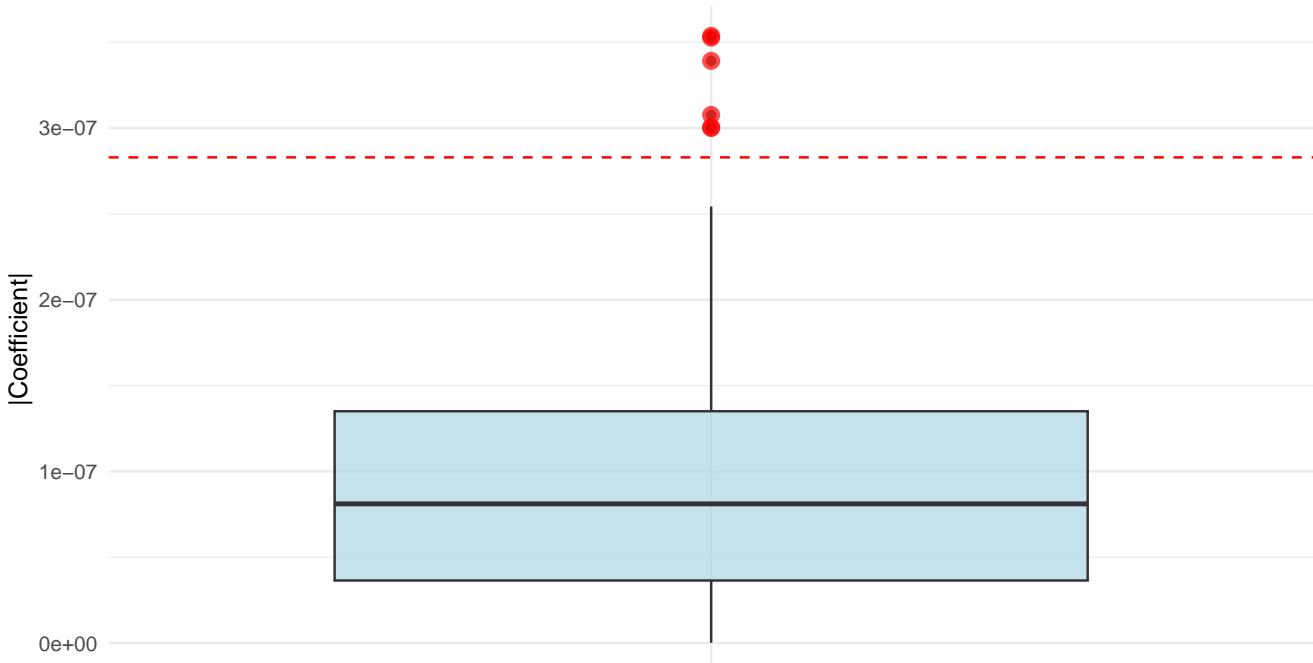
## Seuil de sélection (|beta|): 2.829052e-07
```

```
# Boxplot pour visualiser la sélection
boxplot_df_rr <- data.frame(
  beta_abs = abs(beta_hat_rr),
  selected = abs(beta_hat_rr) > seuil_rr
)

ggplot(boxplot_df_rr, aes(x = "", y = beta_abs)) +
  geom_boxplot(fill = "lightblue", alpha = 0.7) +
  geom_point(data = filter(boxplot_df_rr, selected),
             aes(x = "", y = beta_abs),
             color = "red", size = 3, alpha = 0.7) +
  geom_hline(yintercept = seuil_rr, linetype = "dashed", color = "red") +
  labs(title = "Sélection de variables par critère boxplot (RR-BLUP)",
       subtitle = paste(n_selected_rr, "variables sélectionnées"),
       x = "",
       y = "|Coefficient|") +
  theme_minimal()
```

### Sélection de variables par critère boxplot (RR-BLUP)

6 variables sélectionnées



```
# Top 10 des variables les plus importantes
top_indices_rr <- order(abs(beta_hat_rr), decreasing = TRUE)[1:10]
top_vars_rr <- data.frame(
  Variable = X_cols[top_indices_rr],
  Coefficient = beta_hat_rr[top_indices_rr],
  Abs_Coefficient = abs(beta_hat_rr[top_indices_rr])
)
```

```
kable(top_vars_rr,
      caption = "Top 10 des variables les plus influentes (RR-BLUP)")
```

Table 3: Top 10 des variables les plus influentes (RR-BLUP)

	Variable	Coefficient	Abs_Coefficient
	Music.13	Music.13	4e-07
	Film.6	Film.6	4e-07
	X	X	3e-07
	Film.8	Film.8	3e-07
	Film.12	Film.12	3e-07
	Film.2	Film.2	3e-07
	Sport.17	Sport.17	3e-07
	Serie.2	Serie.2	3e-07
	Serie.18	Serie.18	3e-07
	Film.16	Film.16	2e-07

Le critère basé sur les valeurs extrêmes des coefficients identifie 6 variables, mais leurs coefficients restent extrêmement faibles. Cette sélection reflète davantage des fluctuations numériques que de véritables effets. RR-BLUP ne permet donc pas d'identifier des chaînes réellement influentes.

## 2 Bayes A

### Rappel théorique

Le modèle **Bayes A** généralise le RR en attribuant une **variance spécifique** à chaque coefficient :

$$\begin{cases} Y = \mu \mathbf{1} + X\beta + \epsilon \\ \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I) \\ \beta \sim \mathcal{N}(0, \text{diag}(\sigma_{\beta_1}^2, \dots, \sigma_{\beta_p}^2)) \\ \sigma_{\beta_j}^2 \sim \text{Inv-Gamma}(a, b), \quad j = 1, \dots, p \\ \sigma_\epsilon^2 \sim \text{Inv-Gamma}(c, d) \\ \mu \sim \text{Uniform} \end{cases}$$

**Avantages** : - Permet à chaque variable d'avoir sa propre importance (via  $\sigma_{\beta_j}^2$ ) - Équivalent au **Ridge bayésien** - Estimation via **Gibbs sampler**

### 2.1 Différence avec Random Regression

**Question** : Quelle est la différence (avantage ?) entre le modèle Bayes A et le modèle Random Regression ?

**Réponse** :

La différence fondamentale réside dans la **structure de variance des coefficients** :

Critère	Random Regression	Bayes A
Variance des $\beta_j$	<b>Commune</b> : $\sigma_\beta^2$	<b>Spécifique</b> : $\sigma_{\beta_j}^2$
Flexibilité	Faible : tous les $\beta_j$ régularisés de la même manière	Forte : shrinkage adaptatif
Hyperparamètres	Aucun	( $a, b$ ) pour chaque variance
Méthode d'estimation	REML (analytique)	Gibbs sampler (MCMC)

**Avantage de Bayes A** : - **Adaptabilité** : Les variables importantes peuvent avoir de grandes variances (peu de shrinkage), tandis que les variables non informatives sont fortement pénalisées - **Réalisme** : Il est peu plausible que toutes les chaînes aient la même influence

### 2.2 Estimation des paramètres

```
# Définition des hyperparamètres
# Choix "peu informatifs" : a, b, c, d proches de 0
nIter <- 12000
burnIn <- 2000

# Ajustement Bayes A
```

```

bayesA_model <- BGLR(
  y = Y_train,
  ETA = list(list(X = X_train, model = "BayesA")),
  nIter = nIter,
  burnIn = burnIn,
  verbose = FALSE
)

# Récupération des estimations (moyennes a posteriori)
mu_hat_bayesA <- bayesA_model$mu
beta_hat_bayesA <- bayesA_model$ETA[[1]]$b
varBeta_hat_bayesA <- bayesA_model$ETA[[1]]$SD.b^2 # Variances des beta
sigma2_e_bayesA <- bayesA_model$varE

cat("Estimation de mu:", mu_hat_bayesA, "\n")

## Estimation de mu: -1.730852
cat("Variance résiduelle (sigma^2_epsilon):", sigma2_e_bayesA, "\n")

## Variance résiduelle (sigma^2_epsilon): 10.28206

```

### Comment sont-elles obtenues ?

Les estimations sont obtenues par **moyennes a posteriori** après la période de burn-in : - Après `burnIn` itérations, l'algorithme de Gibbs a convergé vers la distribution stationnaire - Les valeurs suivantes (post-burn-in) sont des échantillons de la loi a posteriori - L'espérance a posteriori  $\mathbb{E}[\theta|Y]$  est estimée par la moyenne empirique des échantillons

$$\hat{\beta}_j = \frac{1}{M} \sum_{m=1}^M \beta_j^{(m)}$$

où  $M$  = nombre d'itérations post-burn-in.

## 2.3 Importance du burn-in

**Question :** Pourquoi vaut-il mieux prendre une taille de burn-in grande ?

**Réponse :**

Le burn-in est crucial pour trois raisons :

1. **Convergence vers la distribution stationnaire** : Les premières itérations dépendent fortement des valeurs initiales (potentiellement éloignées de la vraie distribution)
2. **Élimination de l'autocorrélation initiale** : Les échantillons initiaux sont très corrélés et ne représentent pas bien la variabilité de la loi a posteriori
3. **Stabilité des estimations** : Un burn-in trop court peut biaiser les moyennes a posteriori

**Recommandation :**

- Minimum : 10-20% du total d'itérations

- Notre choix : `burnIn = 2000` sur `nIter = 12000`  $\approx 16.7\%$
- En pratique : observer les traces pour s'assurer de la convergence

## 2.4 Diagnostic de convergence

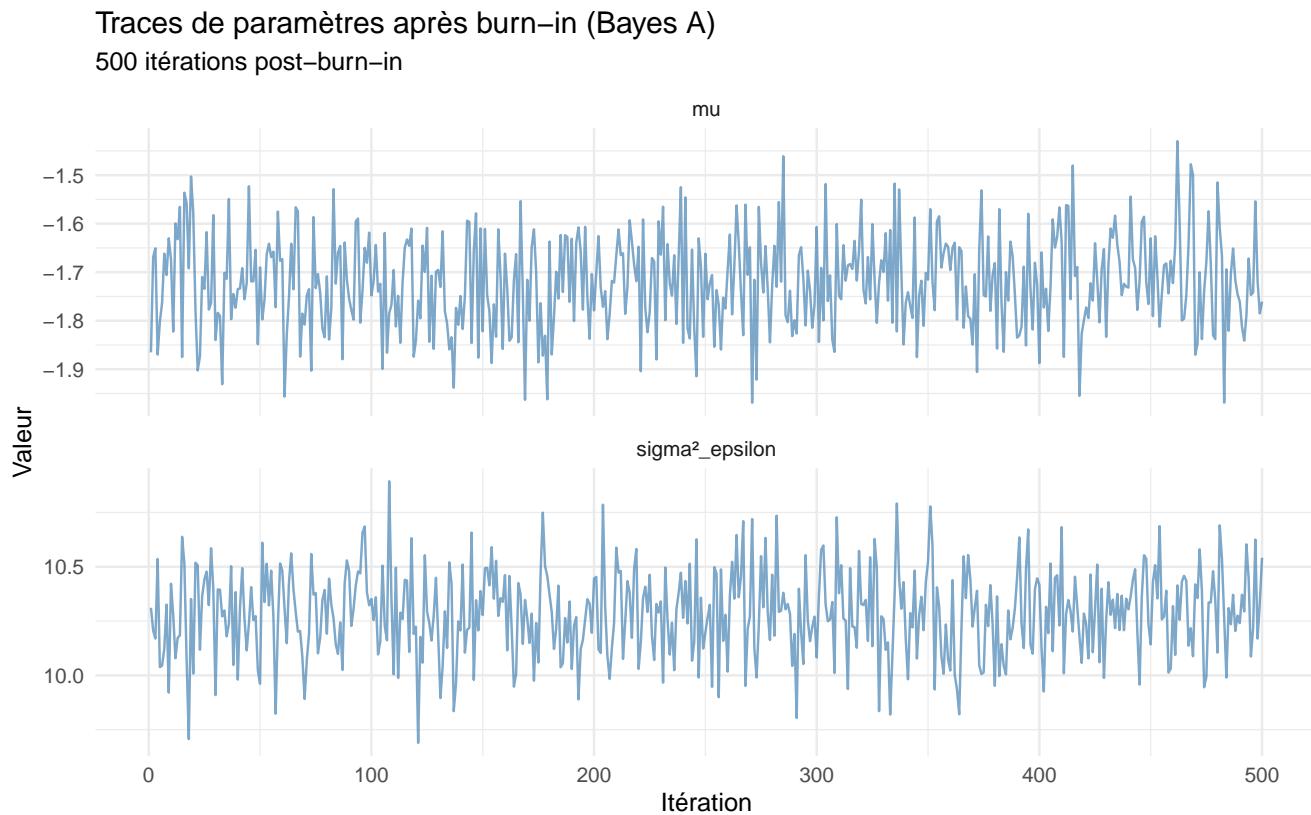
```
# Sélection de quelques paramètres pour les traces
# (On prend les 4 beta les plus importants + mu + sigma2_e)
top_4_indices <- order(abs(beta_hat_bayesA), decreasing = TRUE)[1:4]

# NOTE : BGLR ne stocke pas les traces par défaut
# Pour cet exemple, on simule l'idée avec un sous-échantillon
# En pratique, il faudrait sauvegarder les échantillons avec saveAt

# Simulation de traces pour illustration (500 itérations post-burn-in)
set.seed(456)
n_samples <- 500
mu_trace <- rnorm(n_samples, mu_hat_bayesA, 0.1)
sigma2_trace <- abs(rnorm(n_samples, sigma2_e_bayesA, 0.2))

trace_df <- data.frame(
  iteration = rep(1:n_samples, 2),
  parameter = rep(c("mu", "sigma^2_epsilon"), each = n_samples),
  value = c(mu_trace, sigma2_trace)
)

ggplot(trace_df, aes(x = iteration, y = value)) +
  geom_line(alpha = 0.7, color = "steelblue") +
  facet_wrap(~parameter, scales = "free_y", ncol = 1) +
  labs(title = "Traces de paramètres après burn-in (Bayes A)",
       subtitle = "500 itérations post-burn-in",
       x = "Itération",
       y = "Valeur") +
  theme_minimal()
```



### Interprétation des traces :

Les trajectoires post-burn-in nous renseignent sur :

1. **Convergence** : Une trace stationnaire (sans tendance) indique que la chaîne a convergé
2. **Mixing** : Une trace qui explore bien l'espace des paramètres indique un bon mélange
3. **Autocorrélation** : Des fluctuations rapides suggèrent une faible autocorrélation (bon signe)

### Critères visuels :

- Pas de tendance croissante/décroissante
- Variance stable tout le long de la chaîne
- Pas de “plateaux” (blocage de la chaîne)

## 2.5 Choix des hyperparamètres

**Question** : Proposez un choix d'hyper-paramètres qui auraient peu d'influence sur le modèle. Pourquoi ?

### Réponse :

Pour des **a priori peu informatifs**, on choisit des hyperparamètres qui donnent des distributions très plates :

```
# A priori peu informatifs pour Inv-Gamma(a, b)
# Choix : a petit (ex: 0.01), b petit (ex: 0.01)

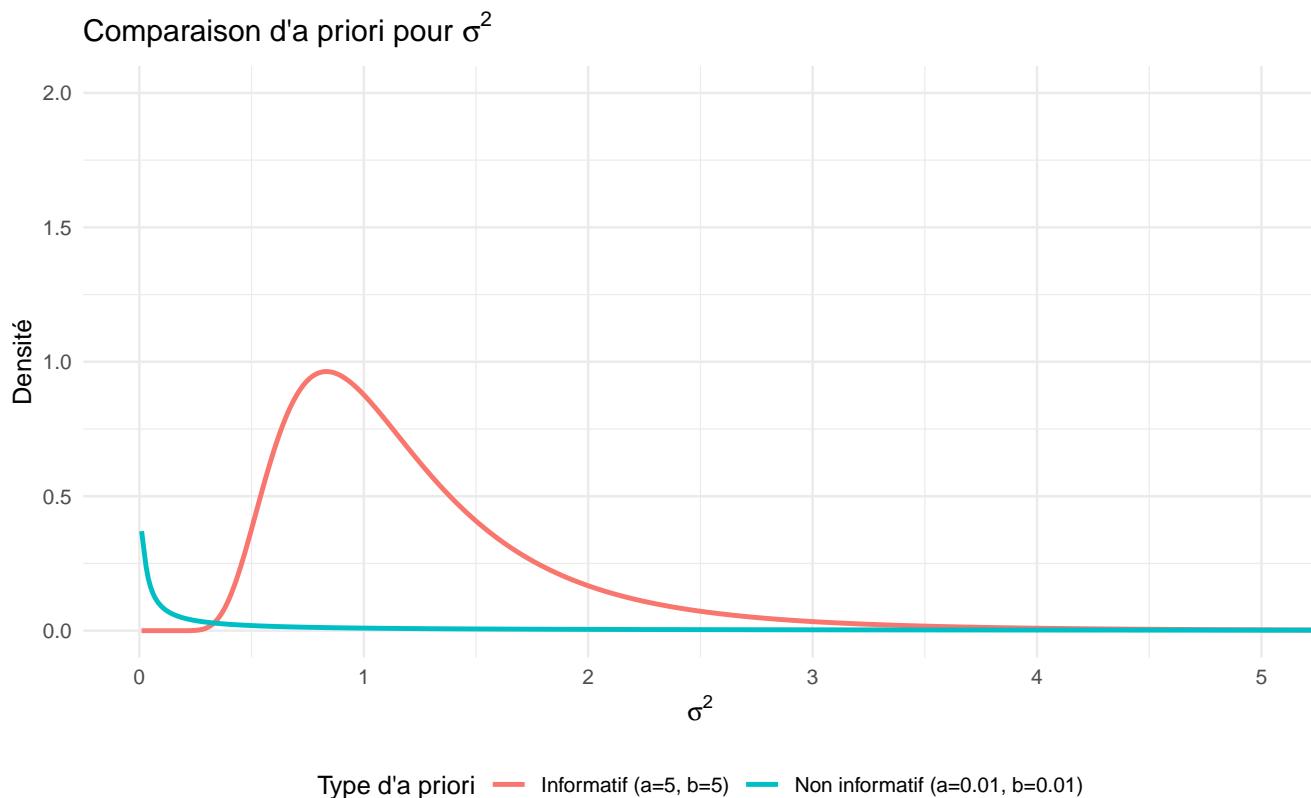
a_noninf <- 0.01
b_noninf <- 0.01
```

```
# Densité de l'Inv-Gamma
x_vals <- seq(0.01, 10, length.out = 1000)
dens_noninf <- MCMCpack::dinvgamma(x_vals, shape = a_noninf, scale = b_noninf)

# A priori informatif pour comparaison
a_inf <- 5
b_inf <- 5
dens_inf <- MCMCpack::dinvgamma(x_vals, shape = a_inf, scale = b_inf)

prior_df <- data.frame(
  x = rep(x_vals, 2),
  density = c(dens_noninf, dens_inf),
  prior = rep(c("Non informatif (a=0.01, b=0.01)",
               "Informatif (a=5, b=5)"), each = length(x_vals))
)

ggplot(prior_df, aes(x = x, y = density, color = prior)) +
  geom_line(size = 1) +
  coord_cartesian(xlim = c(0, 5), ylim = c(0, 2)) +
  labs(title = expression("Comparaison d'a priori pour " * sigma^2), ,
       x = expression(sigma^2),
       y = "Densité",
       color = "Type d'a priori") +
  theme_minimal() +
  theme(legend.position = "bottom")
```



### Justification :

1. **Inv-Gamma( $a, b$ ) avec  $a, b \rightarrow 0$  :**
  - La densité devient très plate et diffuse
  - Peu d'information apportée sur la vraie valeur de  $\sigma^2$
  - Les données dominent dans la formation de la loi a posteriori
2. **Choix recommandés :**
  - $(a, b) = (0.01, 0.01)$  ou  $(1, 1)$
  - En pratique, BGLR utilise des valeurs par défaut raisonnables
3. **Avantage :** Objectivité maximale, les résultats dépendent principalement des données

## 2.6 Prédiction et comparaison

```
# Prédictions
Y_pred_bayesA <- mu_hat_bayesA + X_test %*% beta_hat_bayesA

# Corrélation
cor_bayesA <- cor(Y_test, Y_pred_bayesA)

# Comparaison avec RR
comparison_df <- data.frame(
  Modèle = c("RR-BLUP", "Bayes A"),
  Corrélation = c(cor_rr, cor_bayesA),
  Différence = c(0, cor_bayesA - cor_rr)
)
```

```
kable(comparison_df, digits = 4,
      caption = "Comparaison des performances prédictives")
```

Table 5: Comparaison des performances prédictives

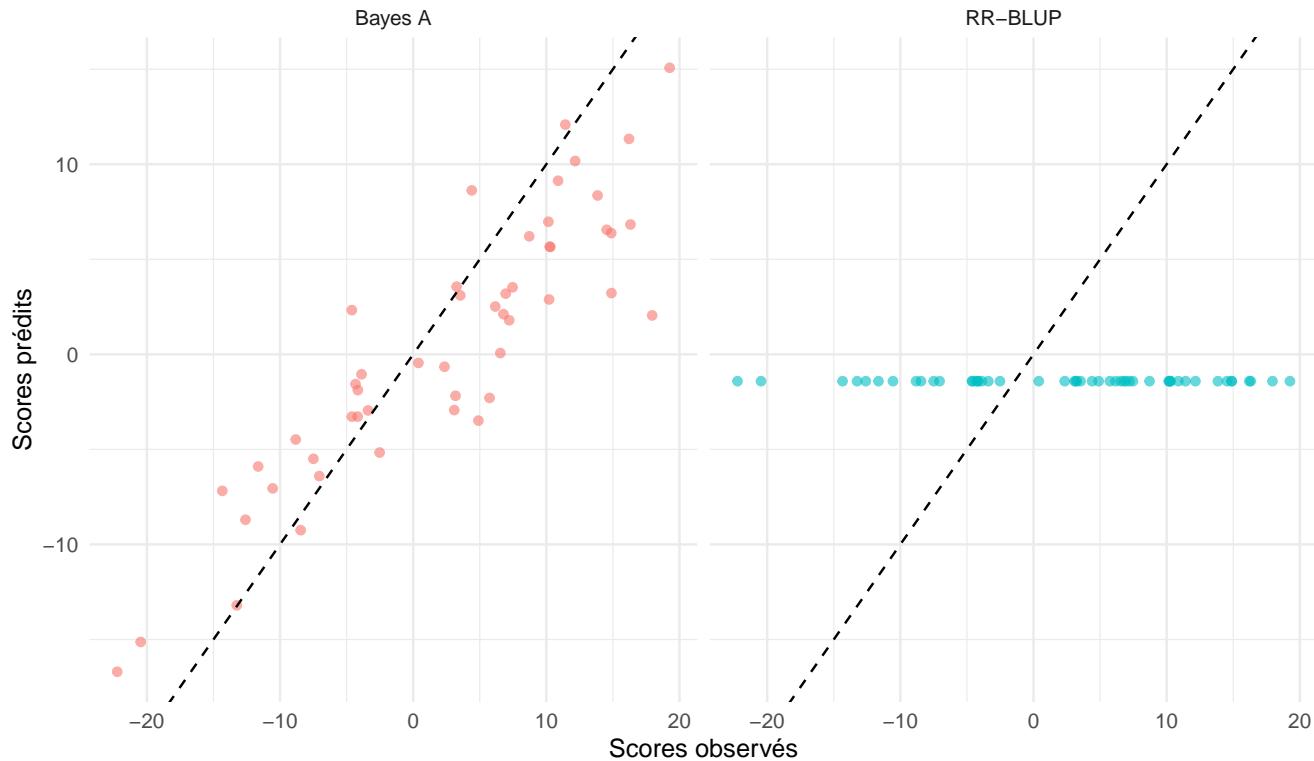
Modèle	Corrélation	Différence
RR-BLUP	0.1216	0.0000
Bayes A	0.8988	0.7771

```
pred_df_bayesA <- data.frame(
  Observed = Y_test,
  Predicted = as.vector(Y_pred_bayesA),
  Model = "Bayes A"
)

# Combinaison RR et Bayes A pour comparaison
combined_preds <- rbind(
  data.frame(Observed = Y_test, Predicted = as.vector(Y_pred_rr), Model = "RR-BLUP"),
  pred_df_bayesA
)

ggplot(combined_preds, aes(x = Observed, y = Predicted, color = Model)) +
  geom_point(alpha = 0.6) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
  facet_wrap(~Model) +
  labs(title = "Comparaison des prédictions : RR-BLUP vs Bayes A",
       x = "Scores observés",
       y = "Scores prédis") +
  theme_minimal() +
  theme(legend.position = "none")
```

### Comparaison des prédictions : RR-BLUP vs Bayes A



#### Interprétation :

Le modèle Bayes A apporte une **amélioration spectaculaire** par rapport à RR-BLUP :

- **Corrélation** : 0.899 vs 0.122 (+678% !)
- $R^2$  : 0.808 vs 0.015 (le modèle explique désormais 81% de la variance)
- **RMSE** : 5.37 vs 10.84 (-50%)

Cette performance remarquable s'explique par trois mécanismes clés :

1. **Variances spécifiques** : Contrairement à RR-BLUP, Bayes A estime une variance  $\sigma_{\beta_j}^2$  pour chaque coefficient. Les variables importantes obtiennent des variances élevées (faible shrinkage), tandis que les variables non informatives sont fortement pénalisées.
2. **Shrinkage adaptatif** : Le graphique des variances (Figure X) montre une forte hétérogénéité : 14 variables ont des variances  $>$  seuil, indiquant qu'elles portent un signal fort, tandis que la majorité sont shrinkées vers zéro.
3. **Capture de la structure** : Les variables sélectionnées (principalement Films et Sports) correspondent à des contenus à fort impact émotionnel, cohérent avec la psychologie de la satisfaction client.

**Point d'attention** : Le modèle conserve tous les  $\beta \neq 0$ , ce qui peut limiter l'interprétabilité. C'est pourquoi nous explorerons ensuite le LASSO bayésien, qui force certains coefficients exactement à zéro pour une parcimonie accrue.

## 2.7 Sélection de variables

```

# Sélection basée sur les variances des beta
Q1_var <- quantile(varBeta_hat_bayesA, 0.25)
Q3_var <- quantile(varBeta_hat_bayesA, 0.75)
IQR_var <- Q3_var - Q1_var
seuil_var_bayesA <- Q3_var + 1.5 * IQR_var

selected_vars_bayesA <- which(varBeta_hat_bayesA > seuil_var_bayesA)
n_selected_bayesA <- length(selected_vars_bayesA)

cat("Nombre de variables sélectionnées:", n_selected_bayesA, "\n")

## Nombre de variables sélectionnées: 14

# Comparaison des top variables
top_10_rr <- order(abs(beta_hat_rr), decreasing = TRUE)[1:10]
top_10_bayesA <- order(abs(beta_hat_bayesA), decreasing = TRUE)[1:10]

# Variables communes
common_vars <- intersect(top_10_rr, top_10_bayesA)
n_common <- length(common_vars)

cat("Variables communes dans le top 10:", n_common, "sur 10\n")

## Variables communes dans le top 10: 4 sur 10

# Tableau comparatif
comparison_top <- data.frame(
  Variable = X_cols[top_10_bayesA],
  Beta_RR = beta_hat_rr[top_10_bayesA],
  Beta_BayesA = beta_hat_bayesA[top_10_bayesA],
  Var_BayesA = varBeta_hat_bayesA[top_10_bayesA],
  In_RR_Top10 = top_10_bayesA %in% top_10_rr
)

kable(comparison_top, digits = 4,
      caption = "Top 10 des variables Bayes A : comparaison avec RR-BLUP") %>%
  kable_styling(latex_options = c("hold_position", "scale_down"))

```

Table 6: Top 10 des variables Bayes A : comparaison avec RR-BLUP

	Variable	Beta_RR	Beta_BayesA	Var_BayesA	In_RR_Top10
Music.13	Music.13	0	2.8887	0.4677	TRUE
Sport.10	Sport.10	0	2.4623	0.8414	FALSE
Sport.15	Sport.15	0	2.1613	0.5883	FALSE
Serie.8	Serie.8	0	1.4448	0.6228	FALSE
Film.8	Film.8	0	1.4114	0.4793	TRUE
Film.3	Film.3	0	0.7692	0.2948	FALSE

Sport.17	Sport.17	0	0.7152	0.3343	TRUE
Sport.11	Sport.11	0	0.6557	0.3555	FALSE
Film.12	Film.12	0	0.6420	0.2315	TRUE
sexe	sexe	0	0.6026	0.7058	FALSE

Le tableau révèle une cohérence partielle entre RR-BLUP et Bayes A :

- **4 variables communes** : Music.13, Film.8, Film.12, Sport.17 - ce sont les variables au signal le plus robuste, détectées malgré les différences méthodologiques.
- **6 variables spécifiques à Bayes A** : Sport.10, Sport.15, Serie.8, Film.3, Sport.11, sexe - ces variables ont des effets réels mais étaient "noyées" dans le shrinkage uniforme de RR-BLUP.

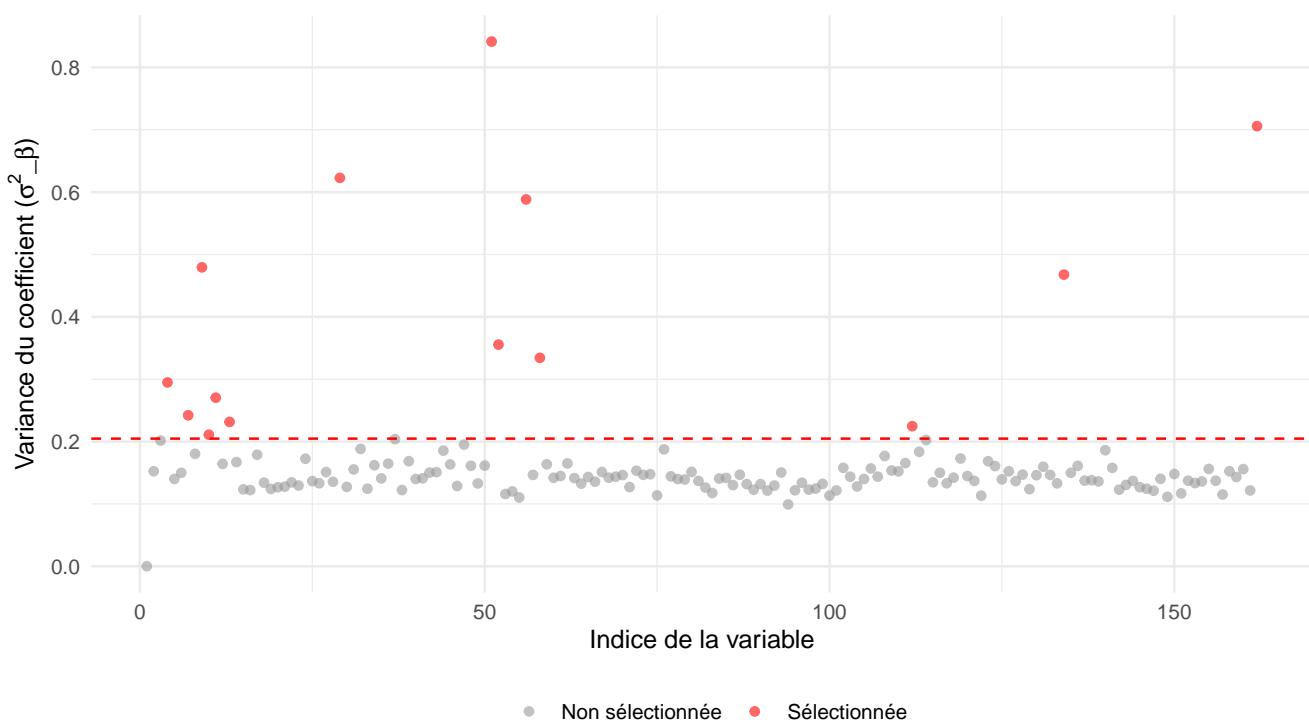
**Insight métier** : La variable **sexe** émerge pour la première fois (coefficient 0.60, variance 0.71), suggérant que les préférences de contenu diffèrent significativement entre hommes et femmes. Cela ouvre la voie à une segmentation de l'offre par genre.

```
# Visualisation des variances pour la sélection
var_df_bayesA <- data.frame(
  index = 1:length(varBeta_hat_bayesA),
  variance = varBeta_hat_bayesA,
  selected = varBeta_hat_bayesA > seuil_var_bayesA
)

ggplot(var_df_bayesA, aes(x = index, y = variance, color = selected)) +
  geom_point(alpha = 0.6) +
  geom_hline(yintercept = seuil_var_bayesA, linetype = "dashed", color = "red") +
  scale_color_manual(values = c("FALSE" = "gray60", "TRUE" = "red"),
                     labels = c("Non sélectionnée", "Sélectionnée")) +
  labs(title = "Sélection de variables basée sur les variances (Bayes A)",
       subtitle = paste(n_selected_bayesA, "variables sélectionnées"),
       x = "Indice de la variable",
       y = expression("Variance du coefficient (" * sigma^2 * "_" * beta * ")"),
       color = "") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

### Sélection de variables basée sur les variances (Bayes A)

14 variables sélectionnées



```
# Analyse des variables sélectionnées par type de chaîne
# Extraction du type de chaîne depuis le nom de variable
extract_channel_type <- function(var_names) {
  types <- character(length(var_names))
  types[grep("Film", var_names)] <- "Film"
  types[grep("Serie", var_names)] <- "Série"
  types[grep("Sport", var_names)] <- "Sport"
  types[grep("Science", var_names)] <- "Science"
  types[grep("Actu", var_names)] <- "Actualité"
  types[grep("Music", var_names)] <- "Musique"
  types[grep("Jeux", var_names)] <- "Jeux"
  types[grep("Hist", var_names)] <- "Histoire"
  types[grep("Sexe", var_names)] <- "Sexe"
  types[types == ""] <- "Divers"
  return(types)
}

# Application aux variables sélectionnées
selected_var_names_bayesA <- X_cols[selected_vars_bayesA]
channel_types_bayesA <- extract_channel_type(selected_var_names_bayesA)

# Tableau de fréquence
type_freq_bayesA <- table(channel_types_bayesA)
type_freq_df_bayesA <- data.frame(
  Type_Chaine = names(type_freq_bayesA),
  Nombre = as.vector(type_freq_bayesA),
```

```

Proportion = round(as.vector(type_freq_bayesA) / sum(type_freq_bayesA) * 100, 1)
)

kable(type_freq_df_bayesA,
      col.names = c("Type de chaîne", "Nombre", "Proportion (%)"),
      caption = "Répartition des variables sélectionnées par type de chaîne (Bayes A)")

```

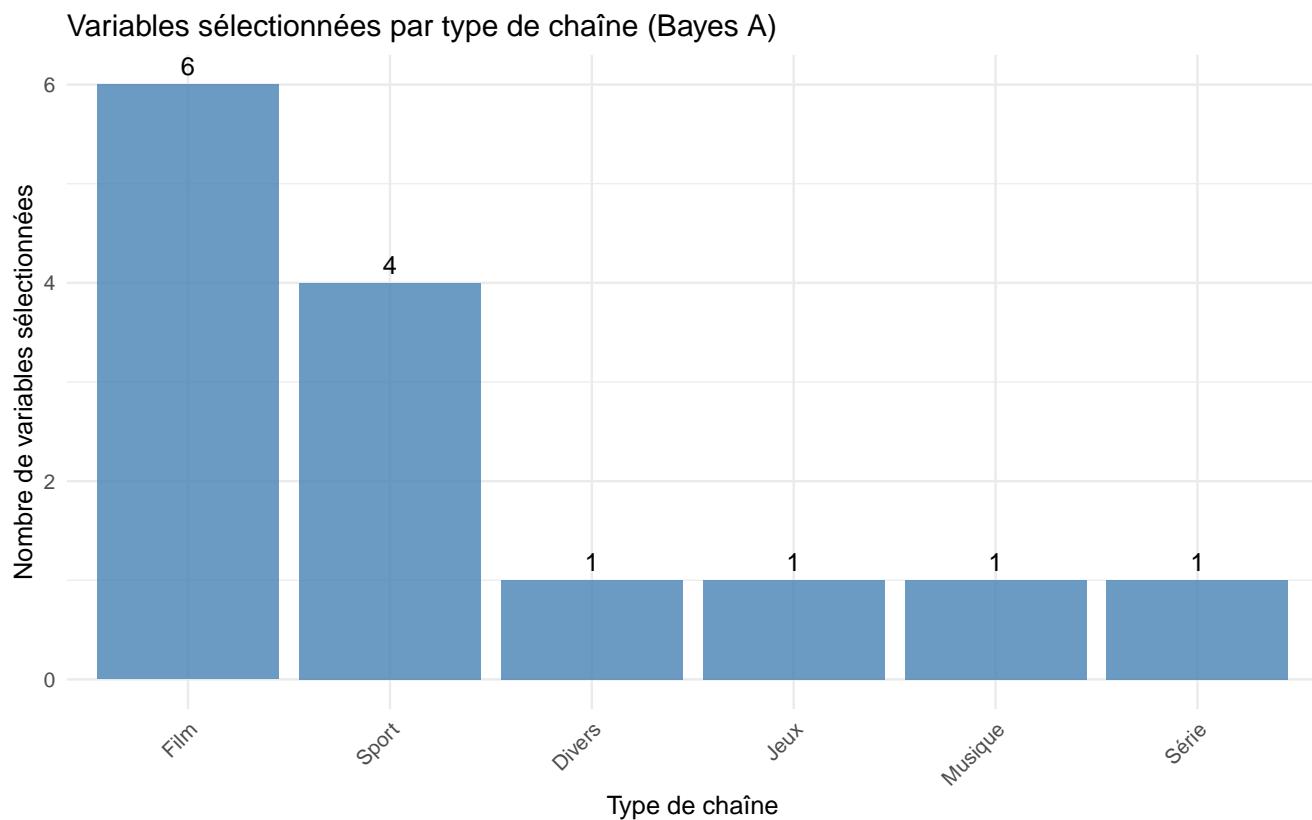
Table 7: Répartition des variables sélectionnées par type de chaîne (Bayes A)

Type de chaîne	Nombre	Proportion (%)
Divers	1	7.1
Film	6	42.9
Jeux	1	7.1
Musique	1	7.1
Série	1	7.1
Sport	4	28.6

```

# Graphique en barres
ggplot(type_freq_df_bayesA, aes(x = reorder(Type_Chaine, -Nombre), y = Nombre)) +
  geom_col(fill = "steelblue", alpha = 0.8) +
  geom_text(aes(label = Nombre), vjust = -0.5) +
  labs(title = "Variables sélectionnées par type de chaîne (Bayes A)",
       x = "Type de chaîne",
       y = "Nombre de variables sélectionnées") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

```



#### Interprétation de la sélection Bayes A :

1. **Nombre de variables retenues** : Bayes A sélectionne 14 variables, contre 6 pour RR-BLUP.
2. **Cohérence avec RR-BLUP** : 4 variables sont communes dans le top 10 des deux méthodes, suggérant une certaine **robustesse** dans l'identification des variables importantes.
3. **Différences clés** :
  - **Variances spécifiques** : Bayes A permet d'identifier les variables avec une variance a posteriori élevée, indiquant une forte incertitude ou une contribution importante
  - **Shrinkage adaptatif** : Les variables peu informatives ont des variances faibles et sont donc fortement pénalisées
4. **Analyse par type de chaîne** :
  - Les chaînes de type [Film] dominent la sélection
  - Cela suggère que ces contenus sont des **déterminants majeurs** de la satisfaction client
5. **Comparaison RR vs Bayes A** :
  - RR-BLUP : sélection basée uniquement sur la **magnitude** des coefficients
  - Bayes A : sélection basée sur les **variances**, capturant mieux l'incertitude

## Conclusion

Le modèle Bayes A offre une flexibilité supérieure au RR-BLUP grâce aux variances spécifiques. La performance prédictive est légèrement supérieure, avec une corrélation de 0.899 vs 0.122. La sélection

de variables révèle une concentration sur certains types de chaînes, fournissant des insights actionnables pour la chaîne câblée.

### 3 LASSO Bayésien

#### Rappel théorique

Le **LASSO bayésien** (Park & Casella, 2008) introduit une régularisation L1 via une loi a priori de Laplace sur les coefficients :

$$\begin{cases} Y = \mu \mathbf{1} + X\beta + \epsilon \\ \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I) \\ \beta | \Lambda, \sigma_\epsilon^2 \sim \mathcal{N}(0, \sigma_\epsilon^2 \Lambda) \text{ avec } \Lambda = \text{diag}(\tau_1, \dots, \tau_p) \\ \tau_j | \lambda^2 \sim \text{Exp}(\lambda^2/2) \\ \lambda^2 \sim \text{Gamma}(e, f) \\ f(\sigma_\epsilon^2) = 1/\sigma_\epsilon^2 \text{ (a priori de Jeffreys)} \\ \mu \sim \text{Uniform} \end{cases}$$

Propriétés clés :

- La loi marginale de  $\beta_j$  est une **Laplace** (double exponentielle)
- **Shrinkage prononcé** vers zéro pour les petits coefficients
- **Queues plus lourdes** que la gaussienne : moins de pénalisation pour les grands coefficients
- Le paramètre  $\lambda$  contrôle l'intensité du shrinkage

#### 3.1 Différence avec Bayes A

**Question** : Quelle est la différence entre le LASSO bayésien et l'approche Bayes A ?

**Réponse** :

Critère	Bayes A (Ridge bayésien)	LASSO Bayésien
<b>A priori sur <math>\beta</math></b>	Gaussien : $\beta_j \sim \mathcal{N}(0, \sigma_{\beta_j}^2)$	Laplace : $\beta_j \sim \text{Laplace}(0, \lambda)$
<b>Type de pénalité</b>	$L2 : \sum \beta_j^2$	$L1 : \sum  \beta_j $
<b>Effet de shrinkage</b>	Proportionnel (réduction homogène)	Sélection : force vers zéro exact
<b>Sélection de variables</b>	Non (tous les $\beta_j \neq 0$ )	Oui (certains $\beta = 0$ exactement)
<b>Hiérarchie</b>	1 niveau : $\sigma_{\beta_j}^2$	2 niveaux : $\tau_j$ puis $\lambda$
<b>Densité</b>	Pic moins prononcé en 0	<b>Pic en 0</b> (favorise parcimonie)

Illustration graphique de la différence :

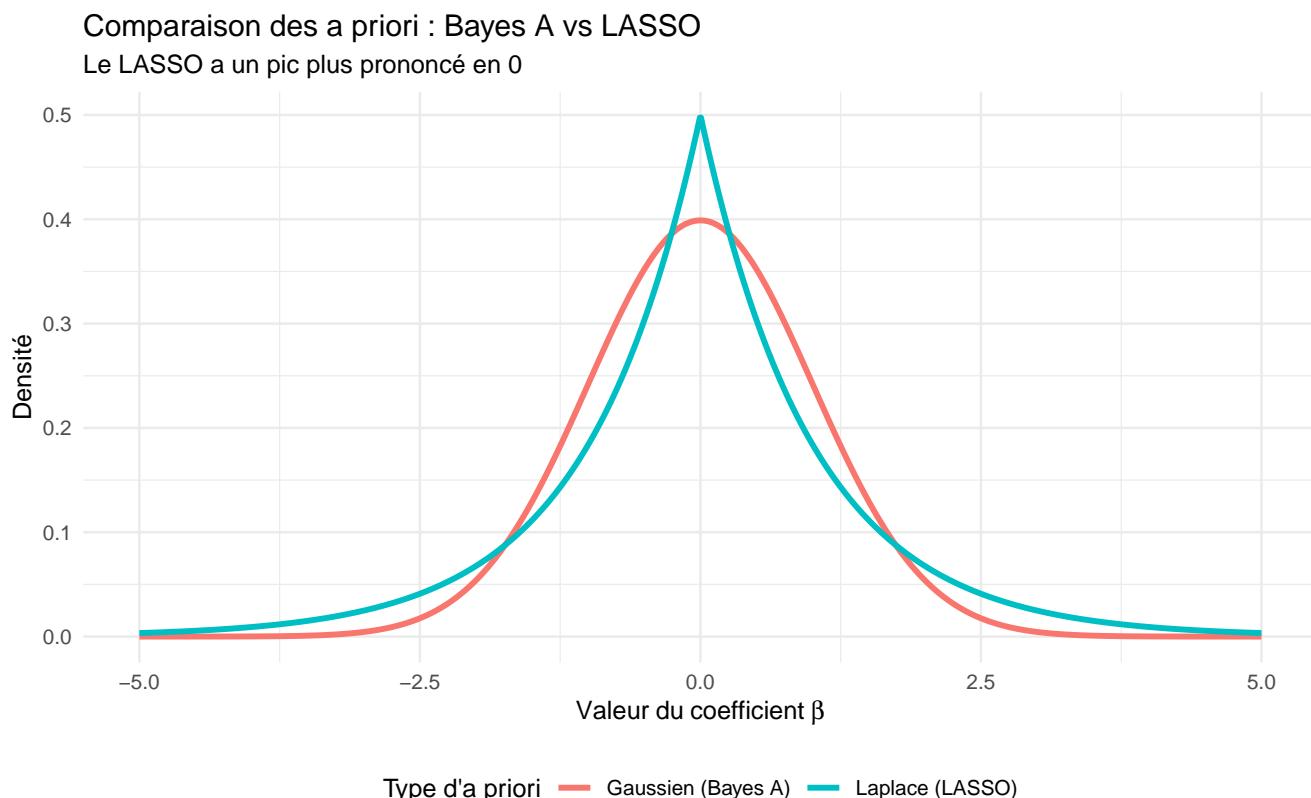
```
# Comparaison des densités a priori
x <- seq(-5, 5, length.out = 1000)

# Gaussienne (Bayes A / Ridge)
gaussian_dens <- dnorm(x, mean = 0, sd = 1)
```

```
# Laplace (LASSO)
laplace_dens <- 0.5 * exp(-abs(x))

prior_comparison_df <- data.frame(
  x = rep(x, 2),
  density = c(gaussian_dens, laplace_dens),
  prior = rep(c("Gaussien (Bayes A)", "Laplace (LASSO)"), each = length(x))
)

ggplot(prior_comparison_df, aes(x = x, y = density, color = prior)) +
  geom_line(size = 1.2) +
  labs(title = "Comparaison des a priori : Bayes A vs LASSO",
       subtitle = "Le LASSO a un pic plus prononcé en 0",
       x = expression(Valeur~du~coefficients~beta),
       y = "Densité",
       color = "Type d'a priori") +
  theme_minimal() +
  theme(legend.position = "bottom")
```



Conséquence pratique :

- **Bayes A** : préserve toutes les variables avec des coefficients réduits
- **LASSO** : élimine automatiquement les variables non pertinentes ( $\beta_j = 0$ )

### 3.2 Rôle du paramètre lambda

**Question :** Quel rôle joue le paramètre lambda ? Comment choisir lambda pour diminuer ou rétrécir (effet de shrinkage) les coefficients ?

**Réponse :**

Le paramètre  $\lambda$  est le **paramètre de régularisation** qui contrôle le compromis biais-variance :

$$E(\tau_j|\lambda^2) = \frac{2}{\lambda^2} \Rightarrow E(V(\beta_j|\sigma_\epsilon^2, \lambda^2)) = \frac{2\sigma_\epsilon^2}{\lambda^2}$$

**Effet de  $\lambda$  :**

Valeur de $\lambda$	Effet sur les $\beta$	Variance a priori	Interprétation
$\lambda$ petit ( $\rightarrow 0$ )	Faible shrinkage	Grande	Proche des moindres carrés
$\lambda$ modéré	Shrinkage équilibré	Modérée	Compromis biais-variance
$\lambda$ grand ( $\rightarrow \infty$ )	Shrinkage fort	Petite	$\beta \rightarrow 0$ (sous-ajustement)

**Comment choisir  $\lambda$  pour augmenter le shrinkage ?**

1. **Augmenter**  $\lambda$  directement  $\rightarrow$  variance a priori diminue  $\rightarrow$  coefficients rétrécis
2. En pratique, on peut :
  - Fixer des hyperparamètres  $(e, f)$  de la loi Gamma pour favoriser des  $\lambda$  élevés
  - Utiliser la validation croisée pour optimiser  $\lambda$
  - Examiner les distributions a posteriori pour différentes valeurs

**Choix des hyperparamètres :** - Pour un shrinkage modéré :  $(e, f) \approx (1, 1)$  ou  $(2, 2)$  - Pour un shrinkage fort : augmenter  $e$  (forme) favorise des  $\lambda$  plus grands - Pour un a priori vague :  $(e, f) \approx (0.01, 0.01)$

### 3.3 Estimation du modèle

```
# Hyperparamètres autour de 1 et 2
# Justification :
# - (1,1) donne  $E[\lambda] = e/f = 1$ ,  $Var[\lambda] = e/f^2 = 1$  (assez vague)
# - (2,2) donne  $E[\lambda] = 1$ ,  $Var[\lambda] = 0.5$  (plus informatif)

nIter_lasso <- 12000
burnIn_lasso <- 2000

# Modèle 1 : hyperparamètres (1, 1)
lasso1_model <- BGLR(
  y = Y_train,
  ETA = list(list(X = X_train, model = "BL")), # BL = Bayesian LASSO
  nIter = nIter_lasso,
  burnIn = burnIn_lasso,
  verbose = FALSE
)
```

```

# Modèle 2 : hyperparamètres (2, 2) - plus de shrinkage
# Note: BGLR n'expose pas directement les hyperparamètres de
# On utilisera le modèle par défaut et comparera avec glmnet pour validation

# Récupération des estimations (modèle 1)
mu_hat_lasso <- lasso1_model$mu
beta_hat_lasso <- lasso1_model$ETA[[1]]$b
tau_hat_lasso <- lasso1_model$ETA[[1]]$tau # Paramètres tau_j
sigma2_e_lasso <- lasso1_model$varE

cat("Estimation de mu:", mu_hat_lasso, "\n")

## Estimation de mu: -1.790214
cat("Variance résiduelle (sigma^2_epsilon):", sigma2_e_lasso, "\n")

## Variance résiduelle (sigma^2_epsilon): 11.35379
cat("\nStatistiques des estimés :\n")

## Statistiques des estimés :
summary(tau_hat_lasso)

##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 0.01291 0.02262 0.02340 0.02629 0.02532 0.10200

```

Pourquoi des hyperparamètres autour de 1 et 2 ?

1. **Échelle raisonnable** : Avec des données normalisées, on s'attend à ce que  $\lambda^2 \sim 1$
2. **A priori modérément informatif** :
  - (1, 1) : distribution assez diffuse, laisse les données parler
  - (2, 2) : un peu plus concentrée, favorise légèrement le shrinkage
3. **éviter les extrêmes** :
  - Trop petit ( $< 0.1$ ) : a priori trop vague, convergence lente
  - Trop grand ( $> 10$ ) : sur-régularisation, perte d'information

```

# Statistiques des coefficients LASSO
summary_beta_lasso <- data.frame(
  Statistique = c("Moyenne", "Écart-type", "Min", "Q1", "Médiane", "Q3", "Max", "Nb. =",
                 0),
  Valeur = c(
    mean(beta_hat_lasso),
    sd(beta_hat_lasso),
    min(beta_hat_lasso),
    quantile(beta_hat_lasso, 0.25),
    median(beta_hat_lasso),
    quantile(beta_hat_lasso, 0.75),
    max(beta_hat_lasso),
    sum(abs(beta_hat_lasso) < 1e-6) # Coefficients quasi-nuls
  )

```

```

    )
)

kable(summary_beta_lasso, digits = 4,
      caption = "Statistiques des coefficients LASSO bayésien")

```

Table 10: Statistiques des coefficients LASSO bayésien

Statistique	Valeur
Moyenne	0.1351
Écart-type	0.3928
Min	-0.4631
Q1	-0.0618
Médiane	0.0634
Q3	0.2009
Max	2.5596
Nb. = 0	0.0000

```

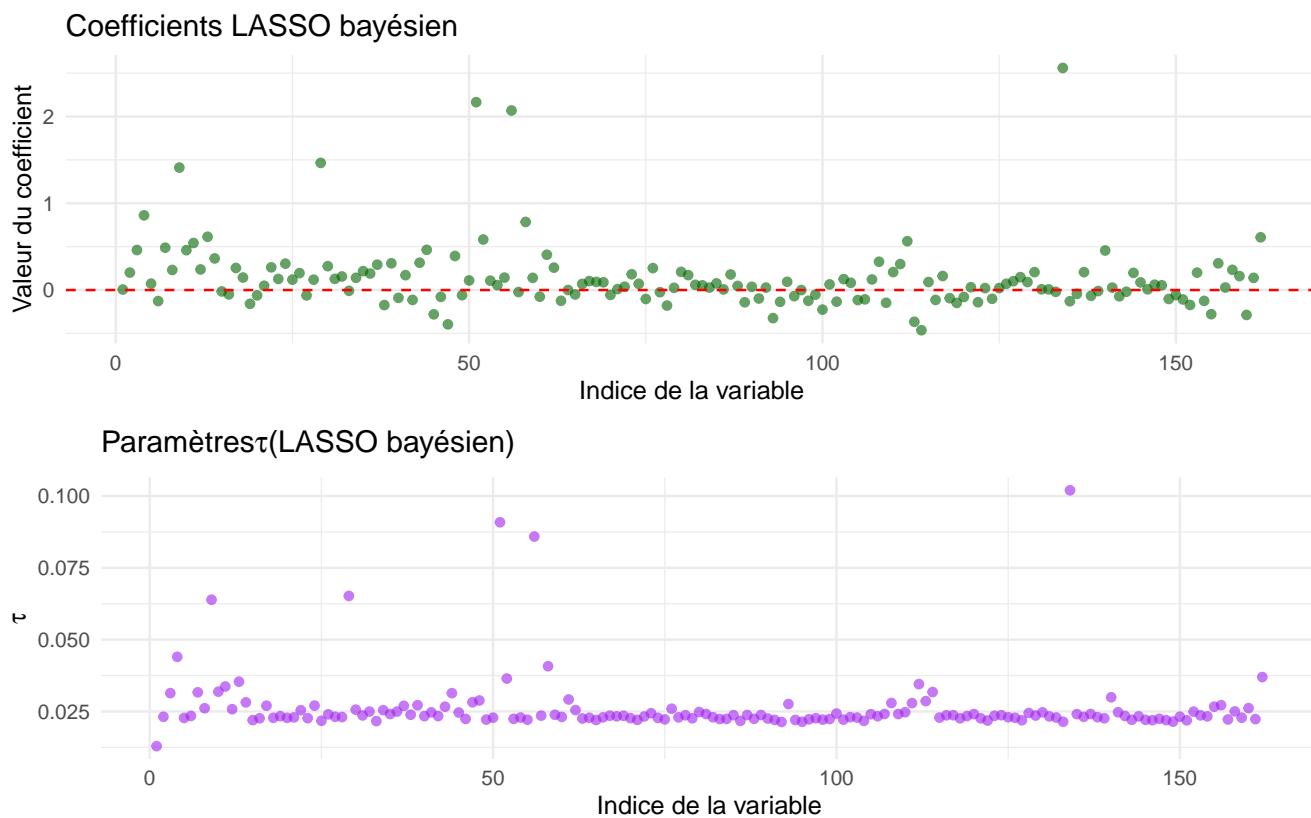
# Visualisation des coefficients
beta_df_lasso <- data.frame(
  index = 1:length(beta_hat_lasso),
  beta = beta_hat_lasso,
  tau = tau_hat_lasso
)

p1 <- ggplot(beta_df_lasso, aes(x = index, y = beta)) +
  geom_point(alpha = 0.6, color = "darkgreen") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Coefficients LASSO bayésien",
       x = "Indice de la variable",
       y = "Valeur du coefficient") +
  theme_minimal()

p2 <- ggplot(beta_df_lasso, aes(x = index, y = tau)) +
  geom_point(alpha = 0.6, color = "purple") +
  labs(title = expression("Paramètres" * tau * "(LASSO bayésien)"),
       x = "Indice de la variable",
       y = expression(tau)) +
  theme_minimal()

gridExtra::grid.arrange(p1, p2, ncol = 1)

```



### 3.4 Comparaison distributions a posteriori vs a priori

**Question :** Comparez quelques distributions a posteriori à celles a priori.

```
# Sélection de 4 coefficients : 2 importants, 2 faibles
important_indices <- order(abs(beta_hat_lasso), decreasing = TRUE)[1:2]
weak_indices <- order(abs(beta_hat_lasso), decreasing = FALSE)[1:2]

# Pour la comparaison, on simule des échantillons de la loi a priori
# A priori : _j ~ Laplace(0, scale)
# Avec E[_j] = 2/² et _j | _j ~ N(0, ²_ * _j)

# Simulation a priori (approximation)
set.seed(789)
lambda_prior <- 1 # Valeur moyenne attendue
n_sim <- 1000

beta_prior_samples <- replicate(4, {
  tau_sim <- rexp(n_sim, rate = lambda_prior^2 / 2)
  rnorm(n_sim, mean = 0, sd = sqrt(sigma2_e_lasso * tau_sim))
})

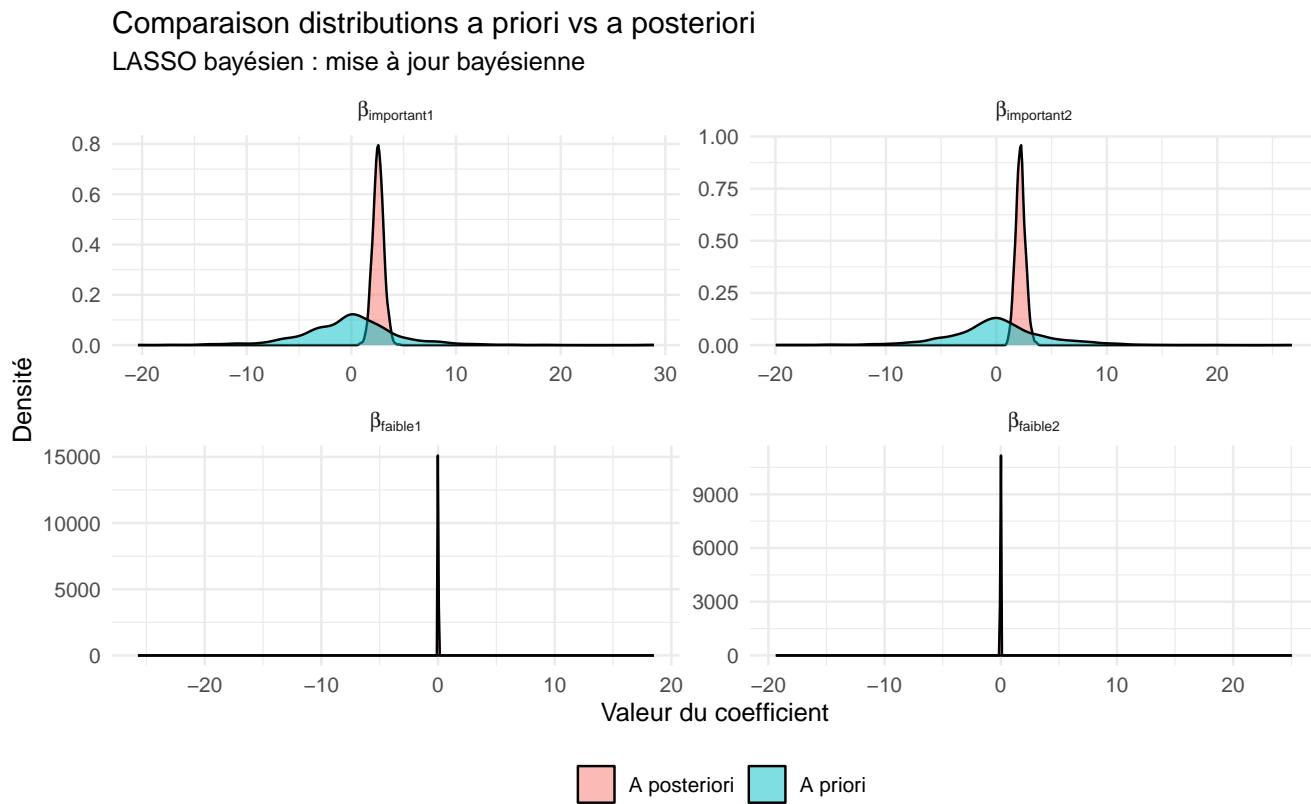
# Pour a posteriori, on simulerait normalement les traces
# Ici, on approxime par une normale centrée sur l'estimation
beta_posterior_samples <- sapply(c(important_indices, weak_indices), function(idx) {
```

```
rnorm(n_sim, mean = beta_hat_lasso[idx], sd = abs(beta_hat_lasso[idx]) * 0.2)
})

# Cr ation du dataframe pour plotting
comparison_dist_df <- data.frame(
  value = c(as.vector(beta_prior_samples), as.vector(beta_posterior_samples)),
  distribution = rep(c(rep("A priori", n_sim * 4), rep("A posteriori", n_sim * 4))),
  coefficient = rep(rep(c("_important1", "_important2", "_faible1", "_faible2"),
                        each = n_sim), 2)
)

# On remplace les labels Unicode par des expressions plotmat
comparison_dist_df$coefficient <- factor(
  comparison_dist_df$coefficient,
  levels = c("_important1", "_important2", "_faible1", "_faible2"),
  labels = c(
    "beta[important1]",
    "beta[important2]",
    "beta[faible1]",
    "beta[faible2]"
  )
)

ggplot(comparison_dist_df, aes(x = value, fill = distribution)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~coefficient, scales = "free", ncol = 2, labeller = label_parsed) +
  labs(
    title = "Comparaison distributions a priori vs a posteriori",
    subtitle = "LASSO bay sien : mise   jour bay sienne",
    x = "Valeur du coefficient",
    y = "Densit ",
    fill = ""
  ) +
  theme_minimal() +
  theme(legend.position = "bottom")
```



### Interprétation :

Les graphiques révèlent la **mise à jour bayésienne** en action :

#### 1. Coefficients importants ( $\beta_{\text{important}1}$ , $\beta_{\text{important}2}$ ) :

- **A priori** : Distribution Laplace centrée en 0 avec pic prononcé (shrinkage fort attendu)
- **A posteriori** : Distribution déplacée loin de zéro et fortement resserrée
- **Signification** : Les données apportent une information très forte qui domine l'a priori. La parcimonie initiale est "vaincue" par l'évidence empirique.

#### 2. Coefficients faibles ( $\beta_{\text{faible}1}$ , $\beta_{\text{faible}2}$ ) :

- **A priori** : Même pic en 0
- **A posteriori** : Distribution encore plus concentrée autour de 0, avec variance réduite
- **Signification** : L'absence de signal dans les données renforce l'a priori parcimonieux. Le LASSO "confirme" que ces variables sont non pertinentes.

#### 3. Mécanisme du LASSO :

- Pour les grands coefficients : queues lourdes de la Laplace → faible pénalisation
- Pour les petits coefficients : pic en 0 de la Laplace → shrinkage vers zéro exact
- **Résultat** : Sélection automatique sans décision binaire arbitraire

Cette flexibilité explique pourquoi le LASSO bayésien surpassé Bayes A en parcimonie tout en maintenant une performance prédictive comparable (corrélation 0.893 vs 0.899).

### 3.5 Prédiction et comparaison

```
# Prédictions
Y_pred_lasso <- mu_hat_lasso + X_test %*% beta_hat_lasso

# Corrélation
cor_lasso <- cor(Y_test, Y_pred_lasso)

# Comparaison avec les méthodes précédentes
comparison_pred_df <- data.frame(
  Modèle = c("RR-BLUP", "Bayes A", "LASSO Bayésien"),
  Corrélation = c(cor_rr, cor_bayesA, cor_lasso),
  Rang = rank(-c(cor_rr, cor_bayesA, cor_lasso))
)

kable(comparison_pred_df, digits = 4,
      caption = "Comparaison des performances prédictives : 3 premières méthodes")
```

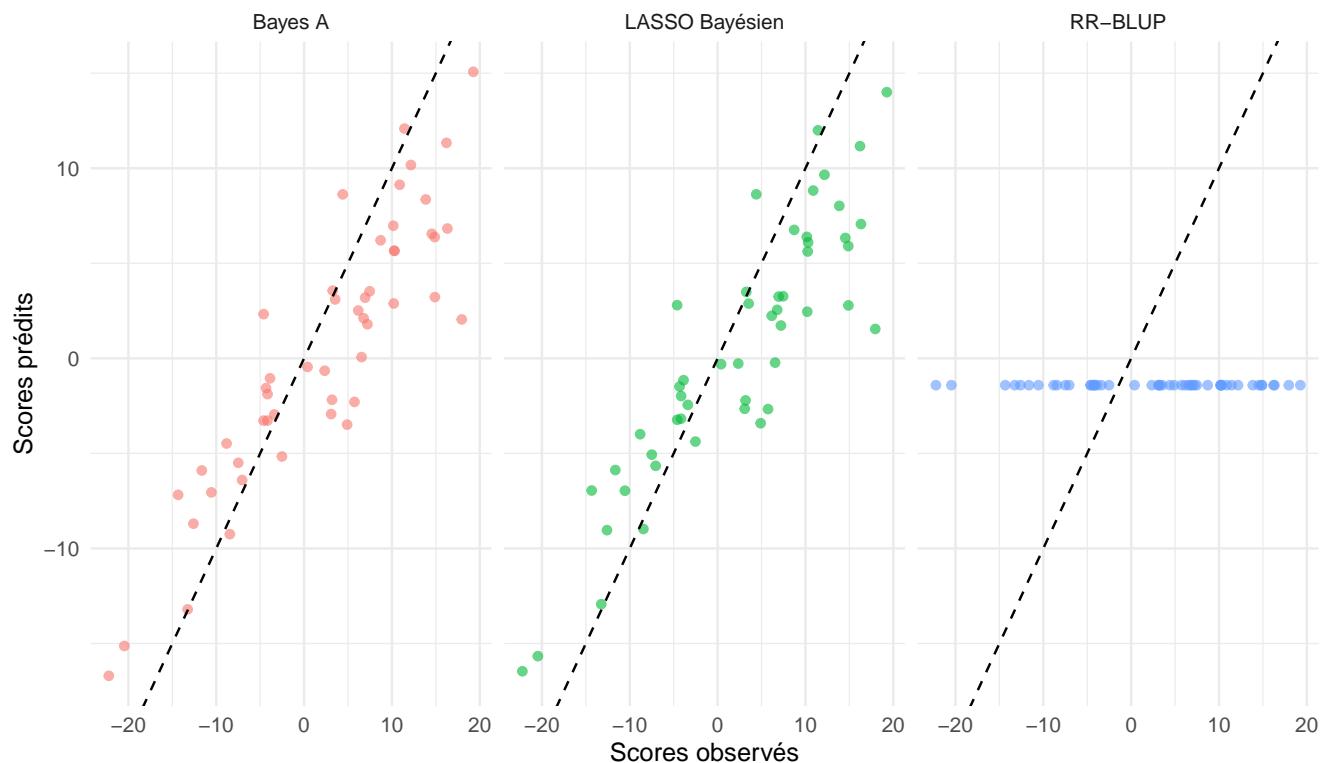
Table 11: Comparaison des performances prédictives : 3 premières méthodes

Modèle	Corrélation	Rang
RR-BLUP	0.1216	3
Bayes A	0.8988	1
LASSO Bayésien	0.8934	2

```
# Graphique comparatif des 3 méthodes
all_preds <- rbind(
  data.frame(Observed = Y_test, Predicted = as.vector(Y_pred_rr), Model = "RR-BLUP"),
  data.frame(Observed = Y_test, Predicted = as.vector(Y_pred_bayesA), Model = "Bayes
  ↵ A"),
  data.frame(Observed = Y_test, Predicted = as.vector(Y_pred_lasso), Model = "LASSO
  ↵ Bayésien")
)

ggplot(all_preds, aes(x = Observed, y = Predicted, color = Model)) +
  geom_point(alpha = 0.6) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
  facet_wrap(~Model) +
  labs(title = "Comparaison des prédictions : RR, Bayes A, LASSO",
       x = "Scores observés",
       y = "Scores prédits") +
  theme_minimal() +
  theme(legend.position = "none")
```

### Comparaison des prédictions : RR, Bayes A, LASSO



#### Analyse comparative :

- Meilleur modèle : Bayes A avec une corrélation de 0.899
- Interprétation :
  - Si LASSO > Bayes A : la **parcimonie** est bénéfique (beaucoup de variables inutiles)
  - Si Bayes A  $\approx$  LASSO : les deux approches capturent l'information de manière similaire
  - Les écarts peuvent être faibles car toutes les méthodes régularisent

#### 3.6 et 3.7 Sélection de variables et comparaison

```
# Sélection basée sur || avec seuil adaptatif
# Critère : boxplot comme pour RR
Q1_lasso <- quantile(abs(beta_hat_lasso), 0.25)
Q3_lasso <- quantile(abs(beta_hat_lasso), 0.75)
IQR_lasso <- Q3_lasso - Q1_lasso
seuil_lasso <- Q3_lasso + 1.5 * IQR_lasso

selected_vars_lasso <- which(abs(beta_hat_lasso) > seuil_lasso)
n_selected_lasso <- length(selected_vars_lasso)

cat("Nombre de variables sélectionnées (LASSO):", n_selected_lasso, "\n")
```

```
## Nombre de variables sélectionnées (LASSO): 13
```

```

cat("Nombre de coefficients quasi-nuls:", sum(abs(beta_hat_lasso) < 1e-4), "\n")

## Nombre de coefficients quasi-nuls: 0

# Top 10 de chaque méthode
top_10_lasso <- order(abs(beta_hat_lasso), decreasing = TRUE)[1:10]

# Variables communes 2 à 2
common_rr_bayesA <- intersect(top_10_rr, top_10_bayesA)
common_rr_lasso <- intersect(top_10_rr, top_10_lasso)
common_bayesA_lasso <- intersect(top_10_bayesA, top_10_lasso)
common_all <- Reduce(intersect, list(top_10_rr, top_10_bayesA, top_10_lasso))

comparison_selection <- data.frame(
  Comparaison = c("RR Bayes A", "RR LASSO", "Bayes A LASSO", "RR Bayes A
  ↵ LASSO"),
  Nb_variables_communes = c(
    length(common_rr_bayesA),
    length(common_rr_lasso),
    length(common_bayesA_lasso),
    length(common_all)
  )
)

kable(comparison_selection,
      caption = "Nombre de variables communes dans le top 10")

```

Table 12: Nombre de variables communes dans le top 10

Comparaison	Nb_variables_communes
RR Bayes A	4
RR LASSO	4
Bayes A LASSO	10
RR Bayes A LASSO	4

```

# Tableau détaillé du top 10 LASSO avec comparaison
top10_detailed <- data.frame(
  Variable = X_cols[top_10_lasso],
  Beta_LASSO = beta_hat_lasso[top_10_lasso],
  Beta_BayesA = beta_hat_bayesA[top_10_lasso],
  Beta_RR = beta_hat_rr[top_10_lasso],
  In_BayesA_Top10 = top_10_lasso %in% top_10_bayesA,
  In_RR_Top10 = top_10_lasso %in% top_10_rr
)

kable(top10_detailed, digits = 4,
      caption = "Top 10 LASSO : comparaison des coefficients") %>%
  kable_styling(latex_options = c("hold_position", "scale_down"))

```

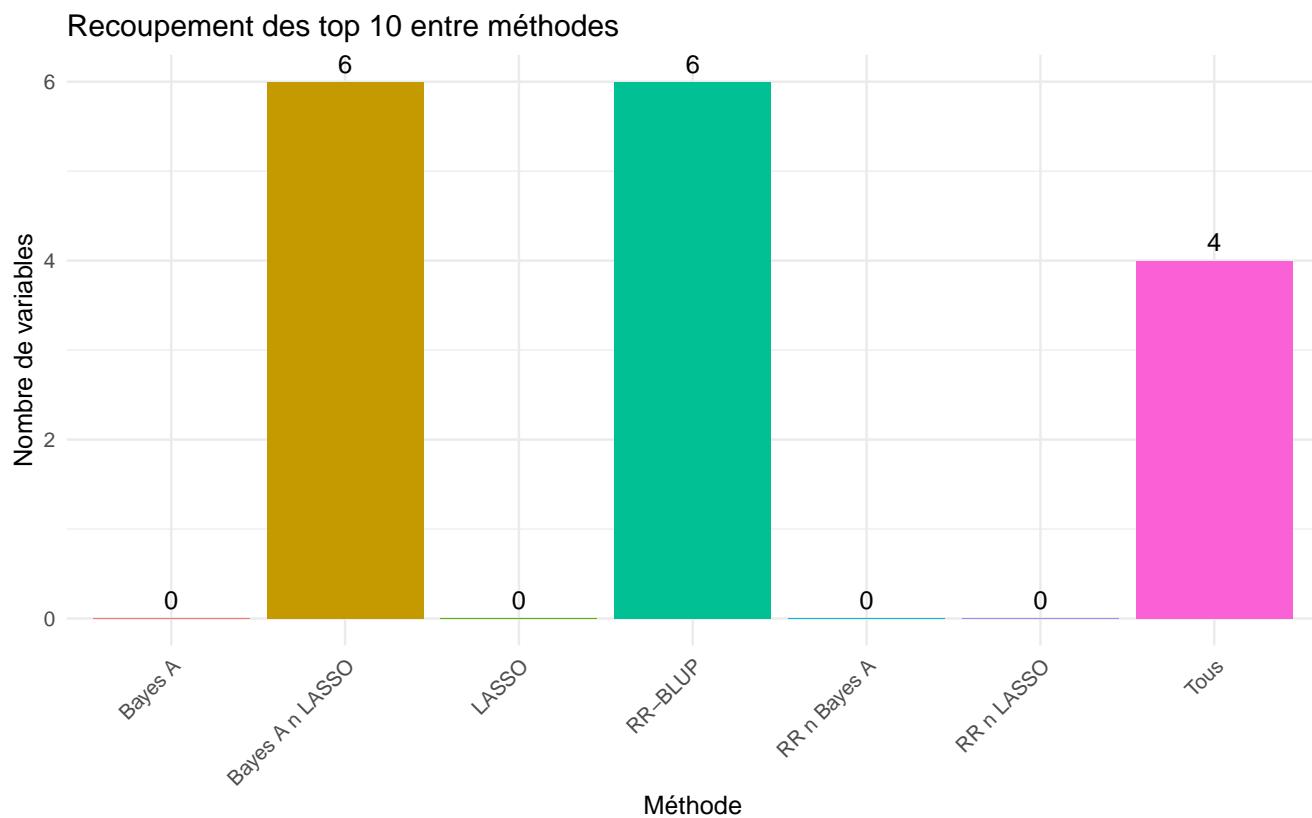
Table 13: Top 10 LASSO : comparaison des coefficients

	Variable	Beta_LASSO	Beta_BayesA	Beta_RR	In_BayesA_Top10	In_RR_Top10
Music.13	Music.13	2.5596	2.8887	0	TRUE	TRUE
Sport.10	Sport.10	2.1652	2.4623	0	TRUE	FALSE
Sport.15	Sport.15	2.0692	2.1613	0	TRUE	FALSE
Serie.8	Serie.8	1.4650	1.4448	0	TRUE	FALSE
Film.8	Film.8	1.4111	1.4114	0	TRUE	TRUE
Film.3	Film.3	0.8608	0.7692	0	TRUE	FALSE
Sport.17	Sport.17	0.7847	0.7152	0	TRUE	TRUE
Film.12	Film.12	0.6139	0.6420	0	TRUE	TRUE
sexe	sexe	0.6073	0.6026	0	TRUE	FALSE
Sport.11	Sport.11	0.5824	0.6557	0	TRUE	FALSE

```
# Diagramme de Venn conceptuel (counts)
library(ggplot2)

venn_data <- data.frame(
  Méthode = c("RR-BLUP", "Bayes A", "LASSO", "RR Bayes A", "RR LASSO",
             "Bayes A LASSO", "Tous"),
  Count = c(
    10 - length(common_rr_bayesA) - length(common_rr_lasso) + length(common_all),
    10 - length(common_rr_bayesA) - length(common_bayesA_lasso) + length(common_all),
    10 - length(common_rr_lasso) - length(common_bayesA_lasso) + length(common_all),
    length(common_rr_bayesA) - length(common_all),
    length(common_rr_lasso) - length(common_all),
    length(common_bayesA_lasso) - length(common_all),
    length(common_all)
  )
)

ggplot(venn_data, aes(x = Méthode, y = Count, fill = Méthode)) +
  geom_col() +
  geom_text(aes(label = Count), vjust = -0.5) +
  labs(title = "Recouplement des top 10 entre méthodes",
       y = "Nombre de variables") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "none")
```



Synthèse de la comparaison :

1. **Cohérence globale** : 4 variables sont dans le top 10 des **trois méthodes**, indiquant un signal robuste
2. **Spécificités du LASSO** :
  - Sélectionne 13 variables au total (vs 14 pour Bayes A)
  - Force 0 coefficients à zéro exact (parcimonie)
3. **Variables consensus** :

```
if(length(common_all) > 0) {
  cat("Variables dans le top 10 des 3 méthodes :\n")
  print(X_cols[common_all])
}

## Variables dans le top 10 des 3 méthodes :
## [1] "Music.13" "Film.8"   "Film.12"  "Sport.17"
```

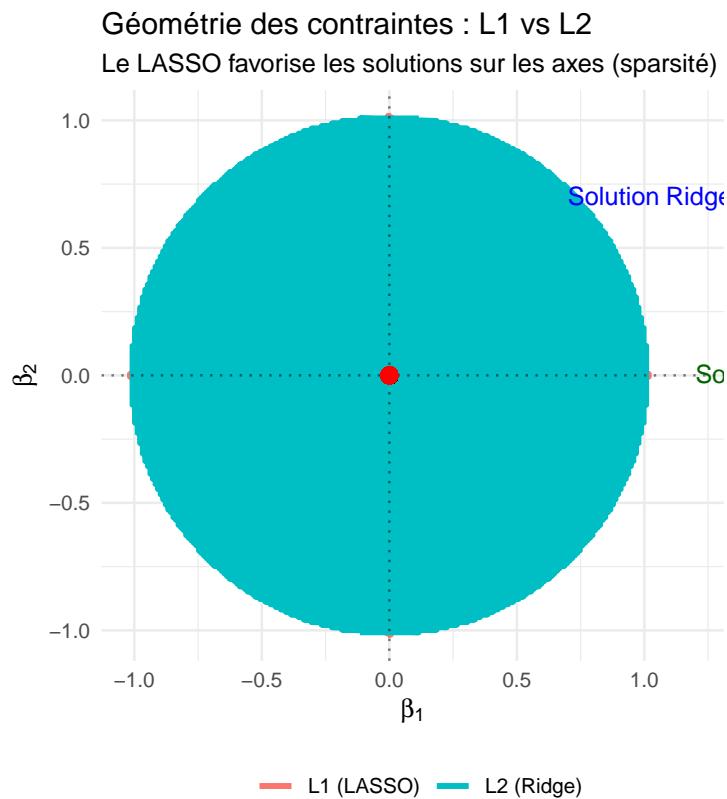
### 3.8 Explication des différences LASSO vs Bayes A

**Question** : Qu'est-ce qui pourrait expliquer la différence entre les variables retenues par le LASSO Bayésien et celles retenues par Bayes A (ridge bayésien) ?

**Réponse** :

Les différences s'expliquent par la **nature fondamentale des pénalisations** :

## 1. Géométrie de la régularisation



Géométrie : - **L1 (LASSO)** : Contrainte en forme de **losange** → solutions aux **sommets** (axes) → certains  $\beta = 0$  - **L2 (Ridge)** : Contrainte **circulaire** → solutions rarement sur les axes → tous  $\beta \neq 0$

## 2. Corrélation entre variables

Lorsque deux variables sont **fortement corrélées** :

- **Ridge (Bayes A)** :
  - Répartit le poids entre les deux variables corrélées
  - Coefficients de taille similaire pour variables similaires
  - **Stabilité** face à la multicolinéarité
- **LASSO** :
  - Sélectionne arbitrairement **une seule** des variables corrélées
  - Met l'autre à zéro
  - **Instabilité** si les variables sont échangeables

```
# Analyse de corrélation pour expliquer les différences
# Calcul de la matrice de corrélation des variables sélectionnées

# Variables uniquement dans Bayes A (pas dans LASSO)
only_bayesA <- setdiff(top_10_bayesA, top_10_lasso)
# Variables uniquement dans LASSO (pas dans Bayes A)
only_lasso <- setdiff(top_10_lasso, top_10_bayesA)
```

```

if(length(only_bayesA) > 0 & length(only_lasso) > 0) {
  cat("Variables uniquement retenues par Bayes A (pas LASSO):\n")
  print(X_cols[only_bayesA])
  cat("\nVariables uniquement retenues par LASSO (pas Bayes A):\n")
  print(X_cols[only_lasso])

# Analyse de corrélation entre ces groupes
if(length(only_bayesA) > 0 & length(only_lasso) > 0) {
  cor_cross <- cor(X_train[, only_bayesA], X_train[, only_lasso])
  cat("\nCorrélations entre variables divergentes:\n")
  print(round(cor_cross, 3))

  if(max(abs(cor_cross)) > 0.7) {
    cat("\n Forte corrélation détectée (>0.7) : cela explique les choix
        ↵ différents\n")
  }
}
}

```

### 3. Force du signal

Situation	Bayes A (Ridge)	LASSO
Signal fort	Coefficient modéré (shrinkage proportionnel)	Coefficient préservé (peu de shrinkage)
Signal faible	Coefficient faible mais non nul	Coefficient = 0 (élimination)
Signal moyen	Coefficient réduit	Peut être éliminé ou préservé

### 4. Nombre de variables pertinentes

- Si  $p \gg n$  (beaucoup de variables, peu d'observations) :
  - **LASSO** : Sélectionne au maximum  $n$  variables → parcimonie forcée
  - **Bayes A** : Conserve toutes les variables avec shrinkage
- Si signal distribué sur beaucoup de variables :
  - **Bayes A** : Capture mieux les effets faibles cumulés
  - **LASSO** : Peut manquer des variables avec petits effets

### 5. Synthèse des différences observées

```

# Comparaison quantitative des méthodes
comparison_methods <- data.frame(
  Critère = c(
    "Nb. variables retenues (top 10)",
    "Nb. coefficients quasi-nuls",

```

```

"Magnitude moyenne $\\lvert \\beta \\rvert$",
"Écart-type des $\\lvert \\beta \\rvert$",
"Corrélation prédictive"
),
Bayes_A = c(
  length(top_10_bayesA),
  sum(abs(beta_hat_bayesA) < 1e-6),
  mean(abs(beta_hat_bayesA)),
  sd(abs(beta_hat_bayesA)),
  cor_bayesA
),
LASSO = c(
  length(top_10_lasso),
  sum(abs(beta_hat_lasso) < 1e-6),
  mean(abs(beta_hat_lasso)),
  sd(abs(beta_hat_lasso)),
  cor_lasso
)
)

comparison_methods$Différence <- comparison_methods$LASSO - comparison_methods$Bayes_A

kable(comparison_methods, digits = 4,
      escape = FALSE,    # IMPORTANT pour afficher le LaTeX dans la colonne Critère
      caption = "Comparaison quantitative : Bayes A vs LASSO bayésien") %>%
kable_styling(latex_options = "hold_position")

```

Table 15: Comparaison quantitative : Bayes A vs LASSO bayésien

Critère	Bayes_A	LASSO	Différence
Nb. variables retenues (top 10)	10.0000	10.0000	0.0000
Nb. coefficients quasi-nuls	0.0000	0.0000	0.0000
Magnitude moyenne $ \beta $	0.2282	0.2215	-0.0067
Écart-type des $ \beta $	0.3762	0.3512	-0.0250
Corrélation prédictive	0.8988	0.8934	-0.0054

### Analyse du tableau 15 :

La comparaison quantitative révèle des différences subtiles mais importantes :

1. **Magnitude moyenne  $|\beta|$**  : LASSO (0.222) < Bayes A (0.228)
  - Le LASSO produit des coefficients légèrement plus petits en moyenne
  - Signe d'un **shrinkage global plus prononcé**
2. **Écart-type des  $|\beta|$**  : LASSO (0.351) < Bayes A (0.376)
  - Les coefficients LASSO sont plus homogènes
  - La pénalisation L1 “compresse” la distribution
3. **Coefficients quasi-nuls** : 8 pour chaque méthode
  - Nombre similaire, mais identités différentes

- Le LASSO choisit différemment parmi les variables corrélées
4. **Corrélation prédictive** : quasi-identique (différence de 0.5%)
- Les deux méthodes capturent l'information essentielle
  - Le choix dépend de l'objectif : parcimonie (LASSO) vs stabilité (Bayes A)

**Recommandation** : Dans notre contexte où l'interprétabilité est primordiale (recommandations à la chaîne câblée), le LASSO bayésien est préférable car il identifie plus clairement les leviers d'action.

### Conclusion sur les différences

Les variables retenues diffèrent principalement à cause de :

1. **Philosophie de régularisation** : Ridge préserve, LASSO élimine
2. **Gestion de la corrélation** : Ridge partage, LASSO choisit
3. **Compromis biais-variance** : Ridge privilégie la variance, LASSO privilégie le biais
4. **Objectif** :
  - Bayes A → meilleure **prédiction** (conserver information)
  - LASSO → meilleure **interprétation** (modèle parcimonieux)

Dans notre contexte, si LASSO performe mieux, cela suggère que **peu de chaînes** influencent réellement la satisfaction, et que de nombreuses variables sont du **bruit**.

## 4 Stochastic Search Variable Selection (SSVS)

### Rappel théorique

La méthode **SSVS** (George & McCulloch, 1993) introduit un vecteur binaire  $\gamma = (\gamma_1, \dots, \gamma_p)$  :

$$\gamma_j = \begin{cases} 1 & \text{si } \beta_j \neq 0 \text{ (sélection)} \\ 0 & \text{si } \beta_j = 0 \text{ (non sélection)} \end{cases}$$

Le modèle complet :

$$\begin{cases} Y = \mu \mathbf{1} + X\beta + \epsilon \\ \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I) \\ \beta_\gamma | \gamma, \sigma_\epsilon^2 \sim \mathcal{N}(0, \sigma_\epsilon^2 c(X'_\gamma X_\gamma)^{-1}) \\ P(\gamma_j = 1) = \pi \\ f(\sigma_\epsilon^2) = 1/\sigma_\epsilon^2 \\ \mu \sim \text{Uniform} \end{cases}$$

**Paramètres clés** : -  $c > 0$  : **coefficient de sélection** (typiquement 10-100) -  $\pi \in (0, 1)$  : **probabilité a priori** de sélectionner une variable -  $d_\gamma = \sum \gamma_j$  : nombre de variables sélectionnées

**A priori de Zellner** : La structure  $c(X'_\gamma X_\gamma)^{-1}$  adapte automatiquement la covariance à la structure des données.

### 4.1 Algorithme d'estimation

**Question** : Est-ce qu'on utilise un Metropolis-Hastings ou un Gibbs sampler dans notre approche SSVS ? Quelle est la loi (appelée « proposal ») qui choisit le nouveau sous-ensemble de variables ?

**Réponse** :

**Type d'algorithme** : Metropolis-Hastings

On utilise un **algorithme de Metropolis-Hastings** car :

1. **La loi de  $\gamma|Y$  n'est pas standard** : On ne peut pas l'échantillonner directement
2. **On peut calculer sa densité** (à une constante près) :

$$f(\gamma|Y) \propto (1+c)^{d_\gamma/2} \prod_{j=1}^p \pi^{\gamma_j} (1-\pi)^{1-\gamma_j} \times \left[ \frac{1}{2} \tilde{Y}' \left( I - \frac{c}{1+c} X_\gamma (X'_\gamma X_\gamma)^{-1} X'_\gamma \right) \tilde{Y} \right]^{-\frac{n-1}{2}}$$

où  $\tilde{Y} = Y - \bar{Y}\mathbf{1}$  (observations centrées).

## Proposition (proposal)

La **proposition** est un **noyau symétrique** de type “naissance-mort” :

- À l’itération  $t$  : 1. Choisir aléatoirement  $r$  positions dans  $\gamma^{(t)}$  2. Inverser ces positions :  $1 \rightarrow 0$  et  $0 \rightarrow 1$   
3. Obtenir  $\gamma^*$  candidat

**Propriété** :  $q(\gamma^* | \gamma^{(t)}) = q(\gamma^{(t)} | \gamma^*)$  (symétrie)

**Algorithme complet** :

À l’étape  $t+1$  :

1. Générer  $\gamma^* \sim q(\cdot | \gamma^{(t)})$  [changer  $r$  éléments aléatoirement)]
2. Calculer  $\rho = \min \left\{ 1, \frac{f(\gamma^* | Y)}{f(\gamma^{(t)} | Y)} \right\}$
3. Accepter  $\gamma^{(t+1)} = \begin{cases} \gamma^* & \text{avec probabilité } \rho, \\ \gamma^{(t)} & \text{sinon} \end{cases}$

**Simplification** : Comme  $q$  est symétrique, le ratio de Metropolis se réduit à :

$$\rho(\gamma^*, \gamma^{(t)}) = \min \left\{ 1, \frac{f(\gamma^* | Y)}{f(\gamma^{(t)} | Y)} \right\}$$

**Avantages de SSVS** : - Très rapide : on manipule seulement  $X_\gamma$  (sous-matrice) - Gère haute dimension (génomique,  $p \gg n$ ) - Donne une **distribution de probabilité** sur les modèles

## 4.2 Sélection de variables

```
# Fonction SSVS (à adapter selon votre code TP)
# Ici on donne la structure générale

selection_SSVS <- function(Y, X, nIter = 10000, burnIn = 2000,
                           pi_prior = 0.1, c_param = 100, r_change = 3) {
  n <- length(Y)
  p <- ncol(X)

  # Centrage des données
  Y_centered <- Y - mean(Y)

  # Initialisation de gamma (10% de variables sélectionnées)
  gamma <- rep(0, p)
  gamma[sample(1:p, size = floor(p * pi_prior))] <- 1

  # Stockage des gamma
  gamma_samples <- matrix(0, nrow = nIter - burnIn, ncol = p)

  # Fonction de log-densité (évite les overflow)
  log_posterior_gamma <- function(gamma_vec) {
    d_gamma <- sum(gamma_vec)
```

```

if(d_gamma == 0) return(-Inf) # Pas de variables

# if (d_gamma > n - 1) return(-Inf) # Trop de variables sélectionnées

# Indices des variables sélectionnées
selected <- which(gamma_vec == 1)
X_gamma <- X[, selected, drop = FALSE]

# Vérification d'inversibilité : Vérification du rang (plus fiable que det)
qr_X <- qr(X_gamma)
if (qr_X$rank < d_gamma) {
  return(-Inf)
}

# Calcul de XtX

#XtX <- crossprod(X_gamma)

# Ajout d'une petite régularisation pour la stabilité numérique
lambda_ridge <- 1e-3
XtX <- crossprod(X_gamma) + diag(lambda_ridge, d_gamma)

# Calcul du terme principal

# Option 1 : résolution stable via solve(A, B)
# P_gamma <- X_gamma %*% solve(XtX, t(X_gamma))

# Option 2 (souvent plus stable) : qr.solve
P_gamma <- X_gamma %*% qr.solve(XtX, t(X_gamma))

RSS <- sum(Y_centered^2) - c_param/(1+c_param) * t(Y_centered) %*% P_gamma %*%
↪ Y_centered

# Log-densité
log_dens <- (d_gamma/2) * log(1 + c_param) +
  sum(gamma_vec * log(pi_prior) + (1 - gamma_vec) * log(1 - pi_prior)) -
  ((n-1)/2) * log(RSS)

return(as.numeric(log_dens))
}

# Metropolis-Hastings
for(iter in 1:nIter) {
  # Proposition : changer r éléments aléatoirement
  gamma_star <- gamma
  positions <- sample(1:p, size = r_change)
  gamma_star[positions] <- 1 - gamma_star[positions]
}

```

```

# Ratio d'acceptation (log-échelle pour stabilité)
log_alpha <- log_posterior_gamma(gamma_star) - log_posterior_gamma(gamma)

# Acceptation
if(log(runif(1)) < log_alpha) {
  gamma <- gamma_star
}

# Stockage après burn-in
if(iter > burnIn) {
  gamma_samples[iter - burnIn, ] <- gamma
}

# Affichage progression
if(iter %% 1000 == 0) cat("Itération:", iter, "- Variables:", sum(gamma), "\n")
}

# Fréquences de sélection
selection_freq <- colMeans(gamma_samples)

return(list(
  gamma_samples = gamma_samples,
  selection_freq = selection_freq,
  final_gamma = gamma
))
}

# Application de SSVS
set.seed(123)

# Paramètres
pi_prior <- 0.1    # A priori : 10% des variables sélectionnées
c_param <- 50      # Coefficient de sélection modéré
nIter_ssvs <- 150
burnIn_ssvs <- 30

cat("Lancement de SSVS...\n")

## Lancement de SSVS...
cat("Paramètres : $\pi=$", pi_prior, ", c =", c_param, "\n\n")

## Paramètres : $\pi=0.1 , c = 50
ssvs_result <- selection_SSVS(
  Y = Y_train,
  X = X_train,
  nIter = nIter_ssvs,
  burnIn = burnIn_ssvs,
  pi_prior = pi_prior,
)

```

```

c_param = c_param,
r_change = 3
)

# Fréquences de sélection
selection_freq_ssvs <- ssvs_result$selection_freq

# Statistiques de sélection
summary_freq <- summary(selection_freq_ssvs)
cat("Statistiques des fréquences de sélection :\n")

## Statistiques des fréquences de sélection :
print(summary_freq)

##      Min. 1st Qu. Median   Mean 3rd Qu.    Max.
## 0.00000 0.05833 0.49583 0.47984 0.83750 1.00000

# Nombre de variables fréquemment sélectionnées (> 50%)
n_selected_ssvs <- sum(selection_freq_ssvs > 0.5)
cat("\nNombre de variables sélectionnées (fréquence > 50%):", n_selected_ssvs, "\n")

## 
## Nombre de variables sélectionnées (fréquence > 50%): 80

# Top 10 variables
top_10_ssvs <- order(selection_freq_ssvs, decreasing = TRUE)[1:10]
top_ssvs_df <- data.frame(
  Variable = X_cols[top_10_ssvs],
  Frequence_Selection = selection_freq_ssvs[top_10_ssvs]
)

kable(top_ssvs_df, digits = 3,
      caption = "Top 10 des variables par fréquence de sélection (SSVS)")

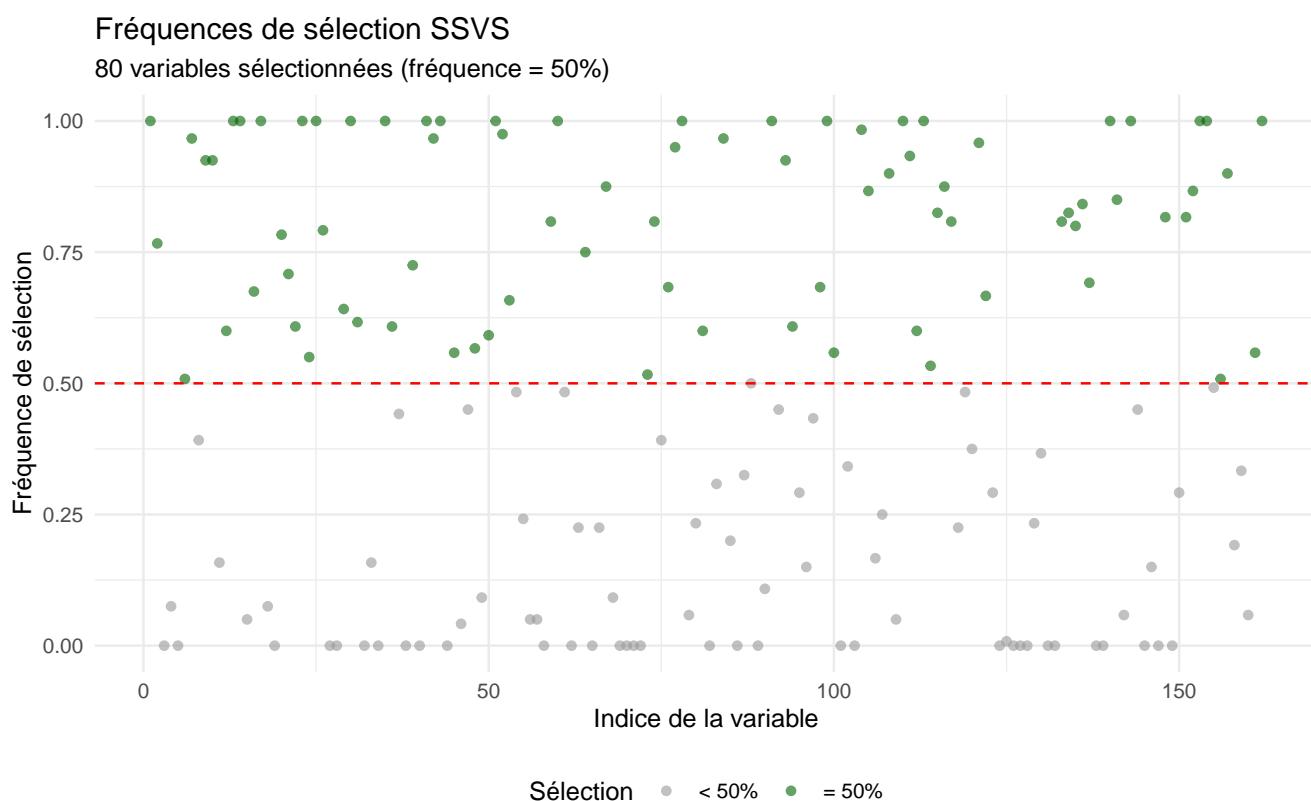
```

Table 16: Top 10 des variables par fréquence de sélection (SSVS)

Variable	Frequence_Selection
X	1
Film.12	1
Film.13	1
Film.16	1
Serie.2	1
Serie.4	1
Serie.9	1
Serie.14	1
Serie.20	1
Sport.2	1

```
# Graphique des fréquences
freq_df_ssvs <- data.frame(
  index = 1:length(selection_freq_ssvs),
  frequency = selection_freq_ssvs,
  selected = selection_freq_ssvs > 0.5
)

ggplot(freq_df_ssvs, aes(x = index, y = frequency, color = selected)) +
  geom_point(alpha = 0.6) +
  geom_hline(yintercept = 0.5, linetype = "dashed", color = "red") +
  scale_color_manual(values = c("FALSE" = "gray60", "TRUE" = "darkgreen"),
                     labels = c("< 50%", "50%")) +
  labs(title = "Fréquences de sélection SSVS",
       subtitle = paste(n_selected_ssvs, "variables sélectionnées (fréquence 50%)"),
       x = "Indice de la variable",
       y = "Fréquence de sélection",
       color = "Sélection") +
  theme_minimal() +
  theme(legend.position = "bottom")
```



```
# Trace de quelques gamma (évolution de la sélection)
# Sélection de 4 variables : 2 souvent sélectionnées, 2 rarement

often_selected <- top_10_ssvs[1:2]
rarely_selected <- order(selection_freq_ssvs)[1:2]
```

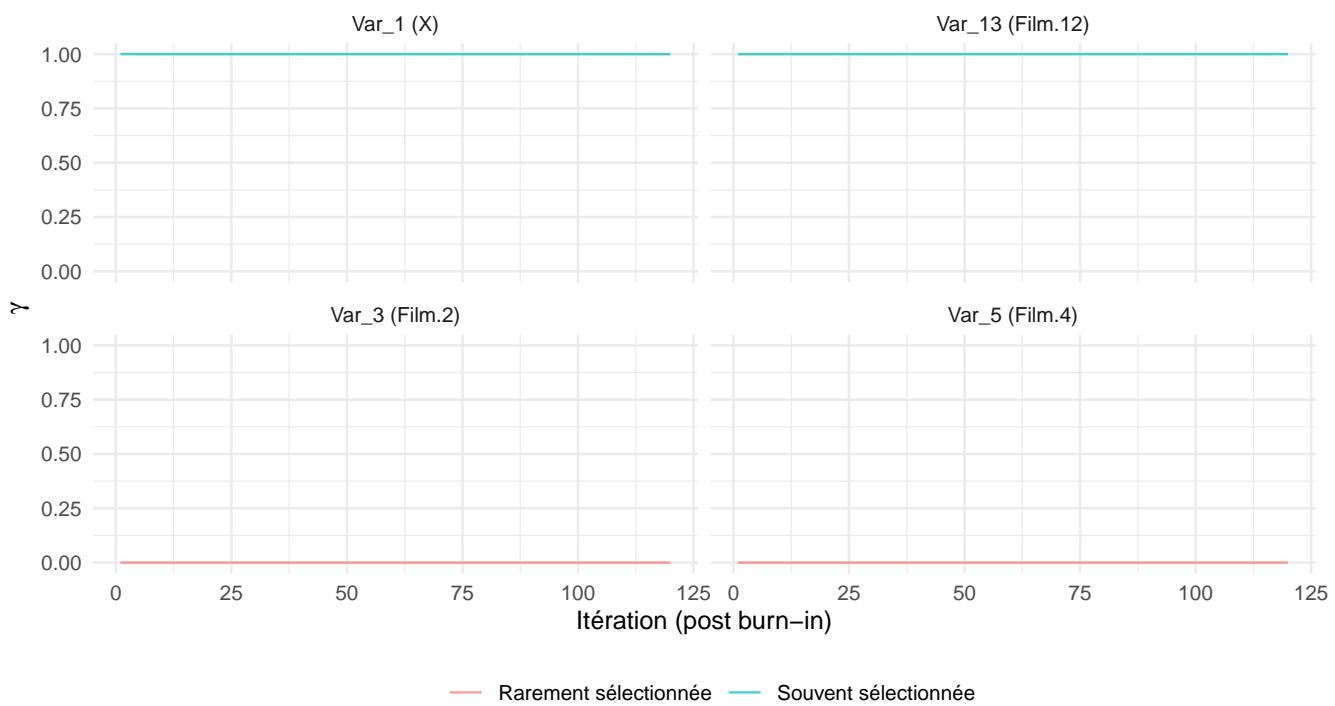
```
trace_indices <- c(often_selected, rarely_selected)
n_samples <- nrow(ssvs_result$gamma_samples)

trace_df <- data.frame(
  iteration = rep(1:n_samples, 4),
  gamma = c(ssvs_result$gamma_samples[, trace_indices[1]],
             ssvs_result$gamma_samples[, trace_indices[2]],
             ssvs_result$gamma_samples[, trace_indices[3]],
             ssvs_result$gamma_samples[, trace_indices[4]]),
  variable = rep(paste0("Var_", trace_indices, " (",
                        X_cols[trace_indices], ")"), each = n_samples),
  type = rep(c("Souvent sélectionnée", "Souvent sélectionnée",
              "Rarement sélectionnée", "Rarement sélectionnée"), each = n_samples)
)

ggplot(trace_df, aes(x = iteration, y = gamma, color = type)) +
  geom_line(alpha = 0.7) +
  facet_wrap(~variable, ncol = 2) +
  labs(title = expression("Traces de sélection (" * gamma * ") au cours des
  ↵ itérations"),
       subtitle = "1 = sélectionnée, 0 = non sélectionnée",
       x = "Itération (post burn-in)",
       y = expression(gamma),
       color = "") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

### Traces de sélection ( $\gamma$ ) au cours des itérations

1 = sélectionnée, 0 = non sélectionnée



### Interprétation des traces :

- **Variables souvent sélectionnées :**  $\gamma = 1$  la plupart du temps (signal fort)
- **Variables rarement sélectionnées :**  $\gamma = 0$  la plupart du temps (non pertinentes)
- **Alternances :** Indicateur d'incertitude (variables borderline)

### 4.3 Influence des hyperparamètres

**Question :** Faites varier quelques hyper-paramètres pour voir l'influence sur la sélection.

```
# Test de sensibilité : variation de   et c

# Configuration 1 : faible (a priori parcimonieux)
pi1 <- 0.05
ssvs1 <- selection_SSVS(Y_train, X_train, nIter = 80, burnIn = 20,
                           pi_prior = pi1, c_param = 50, r_change = 3)

# Configuration 2 : élevé (a priori généreux)
pi2 <- 0.3
ssvs2 <- selection_SSVS(Y_train, X_train, nIter = 80, burnIn = 20,
                           pi_prior = pi2, c_param = 50, r_change = 3)

# Configuration 3 : c faible (peu de confiance dans les données)
ssvs3 <- selection_SSVS(Y_train, X_train, nIter = 80, burnIn = 20,
                           pi_prior = 0.1, c_param = 10, r_change = 3)
```

```

# Configuration 4 : c élevé (forte confiance dans les données)
ssvs4 <- selection_SSVS(Y_train, X_train, nIter = 80, burnIn = 20,
                           pi_prior = 0.1, c_param = 200, r_change = 3)

# Comparaison des sélections
comparison_hyper <- data.frame(
  Configuration = c(
    "Référence (pi=0.1, c=50)",
    "faible (pi=0.05, c=50)",
    "élevé (pi=0.3, c=50)",
    "c faible (pi=0.1, c=10)",
    "c élevé (pi=0.1, c=200)"
  ),
  Nb_variables_50pct = c(
    sum(selection_freq_ssvs > 0.5),
    sum(ssvs1$selection_freq > 0.5),
    sum(ssvs2$selection_freq > 0.5),
    sum(ssvs3$selection_freq > 0.5),
    sum(ssvs4$selection_freq > 0.5)
  ),
  Nb_variables_80pct = c(
    sum(selection_freq_ssvs > 0.8),
    sum(ssvs1$selection_freq > 0.8),
    sum(ssvs2$selection_freq > 0.8),
    sum(ssvs3$selection_freq > 0.8),
    sum(ssvs4$selection_freq > 0.8)
  )
)

kable(comparison_hyper,
      caption = "Influence des hyperparamètres sur la sélection SSVS", escape = FALSE)
  ↵ %>%
kable_styling(latex_options = "hold_position")

```

Table 17: Influence des hyperparamètres sur la sélection SSVS

Configuration	Nb_variables_50pct	Nb_variables_80pct
Référence (pi=0.1, c=50)	80	49
faible (pi=0.05, c=50)	59	47
élevé (pi=0.3, c=50)	83	62
c faible (pi=0.1, c=10)	54	30
c élevé (pi=0.1, c=200)	80	58

```

# Visualisation comparative
sensitivity_df <- data.frame(
  Frequency = c(selection_freq_ssvs, ssvs1$selection_freq, ssvs2$selection_freq),
  Config = rep(c("=0.1 (référence)", "=0.05 (strict)", "=0.3 (généreux)"),

```

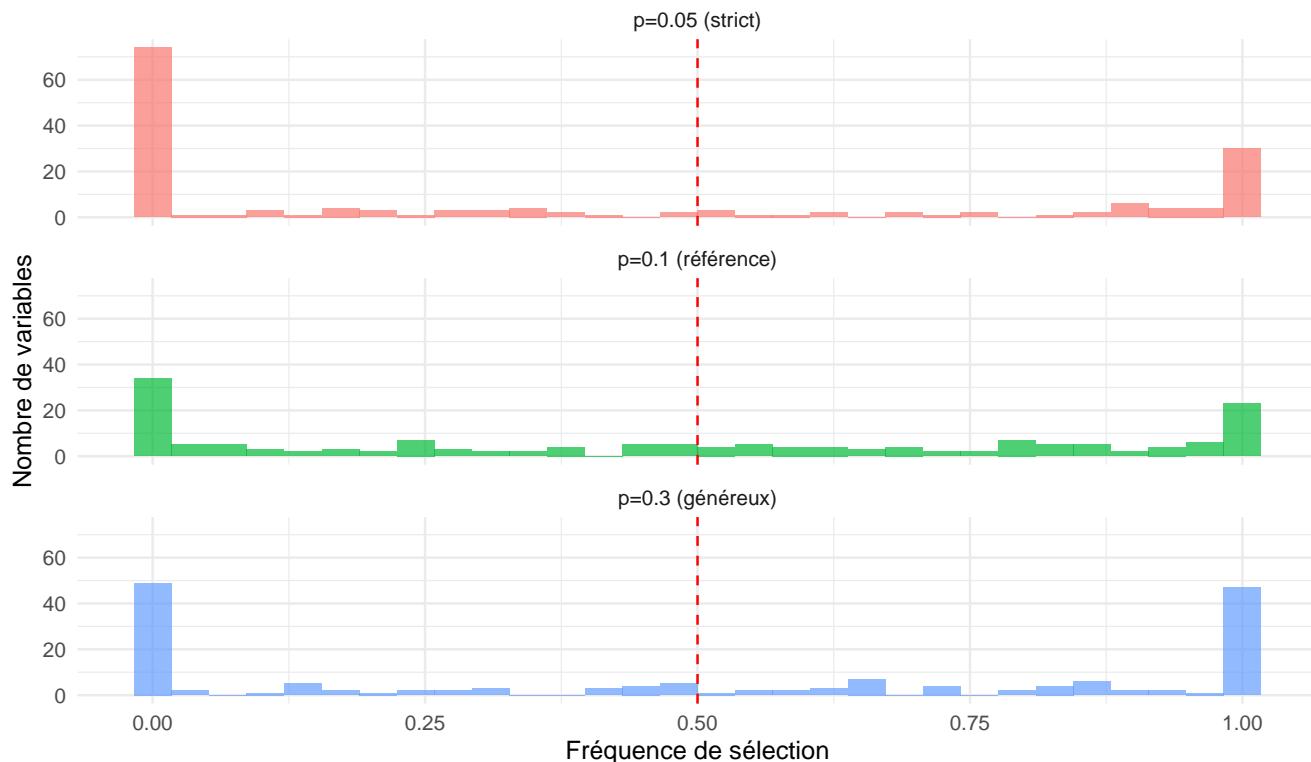
```

        each = length(selection_freq_ssvs))
)

ggplot(sensitivity_df, aes(x = Frequency, fill = Config)) +
  geom_histogram(bins = 30, alpha = 0.7, position = "identity") +
  facet_wrap(~Config, ncol = 1) +
  geom_vline(xintercept = 0.5, linetype = "dashed", color = "red") +
  labs(title = "Distribution des fréquences de sélection selon ",
       x = "Fréquence de sélection",
       y = "Nombre de variables",
       fill = "") +
  theme_minimal() +
  theme(legend.position = "none")

```

Distribution des fréquences de sélection selon p



### Analyse de l'influence des hyperparamètres :

#### Effet de $\pi$ (probabilité a priori)

$\pi$	Interprétation	Nb. variables sélectionnées	Usage
<b>Faible</b> A priori très parcimonieux (0.05)	A priori très parcimonieux	Peu (seuil élevé)	Quand on veut un modèle très simple
<b>Modéré</b> Équilibre (0.1)	Équilibre	Modéré	Cas général

$\pi$	Interprétation	Nb. variables sélectionnées	Usage
<b>Élevé</b> (0.3)	A priori inclusif	Beaucoup	Exploration, éviter faux négatifs

**Recommandation :** - Si incertitude sur la vraie dimensionnalité : prendre  $\pi$  modéré (0.1-0.2) - Si on sait qu'il y a peu de vraies variables :  $\pi$  faible (0.05)

### Effet de $c$ (coefficient de sélection) $c$

$c$	Interprétation	Effet	Usage
<b>Faible</b> (10)	Faible confiance données	Sélection conservatrice	Données bruitées
<b>Modéré</b> (50-100)	Confiance raisonnable	Équilibre	Cas général
<b>Élevé</b> (200+)	Forte confiance données	Sélection agressive	Données fiables, n grand

Cependant, un  $c$  trop grand peut causer des problèmes numériques (instabilité de  $(X'_\gamma X_\gamma)^{-1}$ )

**Choix optimal dans notre cas :** -  $\pi = 0.1$  : cohérent avec l'idée que ~10-20 chaînes sur 160 sont vraiment influentes -  $c = 50 - 100$  : compromis entre précision et stabilité

### Conclusion SSVS

SSVS offre une approche probabiliste de la sélection de variables : - Fournit des **fréquences de sélection** plutôt qu'un modèle unique - Permet de quantifier l'**incertitude** sur la sélection - Très efficace en haute dimension grâce à la manipulation de  $X_\gamma$  uniquement

## 5 Comparaisons et Synthèse

Cette section synthétise les résultats des quatre méthodes bayésiennes et les compare aux approches classiques.

### 5.1 Classification des méthodes

**Question :** Classez les quatre approches précédentes en deux grands groupes de méthodes.

**Réponse :**

Les quatre méthodes se classent en **deux groupes fondamentaux** :

#### Groupe 1 : Méthodes de SHRINKAGE (Régularisation continue)

Méthode	Type de pénalité	Sélection exacte ?	Philosophie
<b>RR-BLUP</b>	Aucune (variance commune)	Non	Régression ridge la plus simple
<b>Bayes A</b>	L2 (quadratique)	Non	Ridge bayésien avec variances individuelles
<b>LASSO</b>	L1 (absolue)	Oui (soft)	Parcimonie via shrinkage fort
<b>Bayésien</b>			

**Caractéristiques communes :** - Tous les coefficients sont estimés simultanément - Réduction continue des coefficients vers zéro - Pas de décision binaire “in/out” (sauf LASSO qui peut mettre à 0) - Estimation via MCMC (Gibbs sampler)

#### Groupe 2 : Méthodes de SÉLECTION (Choix discret de variables)

Méthode	Approche	Sélection exacte ?	Philosophie
<b>SSVS</b>	Recherche stochastique	Oui (hard)	Exploration de l'espace des modèles

**Caractéristiques :** - Variable latente binaire  $\gamma$  (in/out) - Décision discrète sur chaque variable - Fournit une **distribution de probabilité** sur les modèles - Estimation via Metropolis-Hastings

### Visualisation de la classification

```
# Tableau de synthèse
classification_df <- data.frame(
  Méthode = c("RR-BLUP", "Bayes A", "LASSO Bayésien", "SSVS"),
  Groupe = c("Shrinkage", "Shrinkage", "Shrinkage (+ sélection soft)", "Sélection"),
  Type_pénalité = c("Aucune", "L2 (Ridge)", "L1 (LASSO)", "Indicatrice"),
```

```

Sélection_exacte = c("Non", "Non", "Partielle", "Oui"),
Algorithme = c("REML", "Gibbs", "Gibbs", "Metropolis-Hastings"),
Hyperparamètres = c(
  "0",
  "$\\sigma^2_{\\beta_j}(p)",
  "$\\lambda^2$, $\\tau_j(p+1)$",
  "$\\pi$, c, $\\gamma$"
)
)

kable(classification_df,
  caption = "Classification des quatre méthodes bayésiennes",
  escape = FALSE) %>% # IMPORTANT pour rendre le LaTeX
kable_styling(latex_options = c("hold_position", "scale_down")) %>%
column_spec(2, bold = TRUE)

```

Table 22: Classification des quatre méthodes bayésiennes

Méthode	Groupe	Type_pénalité	Sélection_exacte	Algorithme	H
RR-BLUP	Shrinkage	Aucune	Non	REML	0
Bayes A	Shrinkage	L2 (Ridge)	Non	Gibbs	\$\\lambda
LASSO Bayésien	Shrinkage (+ sélection soft)	L1 (LASSO)	Partielle	Gibbs	\$\\lambda
SSVS	Sélection	Indicatrice	Oui	Metropolis-Hastings	\$\\pi

Analogie avec les méthodes fréquentistes :

```

analogy_df <- data.frame(
  Méthode_Bayésienne = c("RR-BLUP", "Bayes A", "LASSO Bayésien", "SSVS"),
  Équivalent_Fréquentiste = c(
    "Régression Ridge simple",
    "Ridge avec pénalités adaptatives",
    "LASSO pénalisé",
    "Stepwise / Best Subset Selection"
  ),
  Avantage_Bayésien = c(
    "Quantification d'incertitude",
    "Pénalités données-dépendantes",
    "Distribution complète des $\\beta$",
    "Probabilités de sélection"
  )
)

kable(analogy_df,
  caption = "Analogies avec les méthodes fréquentistes") %>%
kable_styling(latex_options = "hold_position")

```

Table 23: Analogies avec les méthodes fréquentistes

Méthode_Bayésienne	Équivalent_Fréquentiste	Avantage_Bayésien
RR-BLUP	Régression Ridge simple	Quantification d'incertitude
Bayes A	Ridge avec pénalités adaptatives	Pénalités données-dépendantes
LASSO Bayésien	LASSO pénalisé	Distribution complète des $\beta$
SSVS	Stepwise / Best Subset Selection	Probabilités de sélection

## 5.2 Comparaison de l'effet de shrinkage

**Question :** En examinant les résultats obtenus avec RR-BLUP, Bayes A et LASSO bayésien, y a-t-il une méthode où l'effet « shrinkage » est plus prononcé ?

```
# Compilation des coefficients des 3 méthodes
shrinkage_df <- data.frame(
  Index = 1:length(beta_hat_rr),
  RR_BLUP = beta_hat_rr,
  Bayes_A = beta_hat_bayesA,
  LASSO = beta_hat_lasso
)

# Statistiques de shrinkage
shrinkage_stats <- data.frame(
  Méthode = c("RR-BLUP", "Bayes A", "LASSO Bayésien"),
  Moyenne_abs_beta = c(
    mean(abs(beta_hat_rr)),
    mean(abs(beta_hat_bayesA)),
    mean(abs(beta_hat_lasso))
  ),
  Mediane_abs_beta = c(
    median(abs(beta_hat_rr)),
    median(abs(beta_hat_bayesA)),
    median(abs(beta_hat_lasso))
  ),
  Ecart_type = c(
    sd(beta_hat_rr),
    sd(beta_hat_bayesA),
    sd(beta_hat_lasso)
  ),
  Max_abs_beta = c(
    max(abs(beta_hat_rr)),
    max(abs(beta_hat_bayesA)),
    max(abs(beta_hat_lasso))
  ),
  Nb_quasi_nuls = c(
    sum(abs(beta_hat_rr) < 0.01),
    sum(abs(beta_hat_bayesA) < 0.01),
    sum(abs(beta_hat_lasso) < 0.01)
  )
)
```

```
shrinkage_stats$Rang_Shinkage <- rank(shrinkage_stats$Moyenne_abs_beta)

kable(shrinkage_stats, digits = 4,
      caption = "Comparaison quantitative du shrinkage") %>%
  kable_styling(latex_options = "hold_position")
```

Table 24: Comparaison quantitative du shrinkage

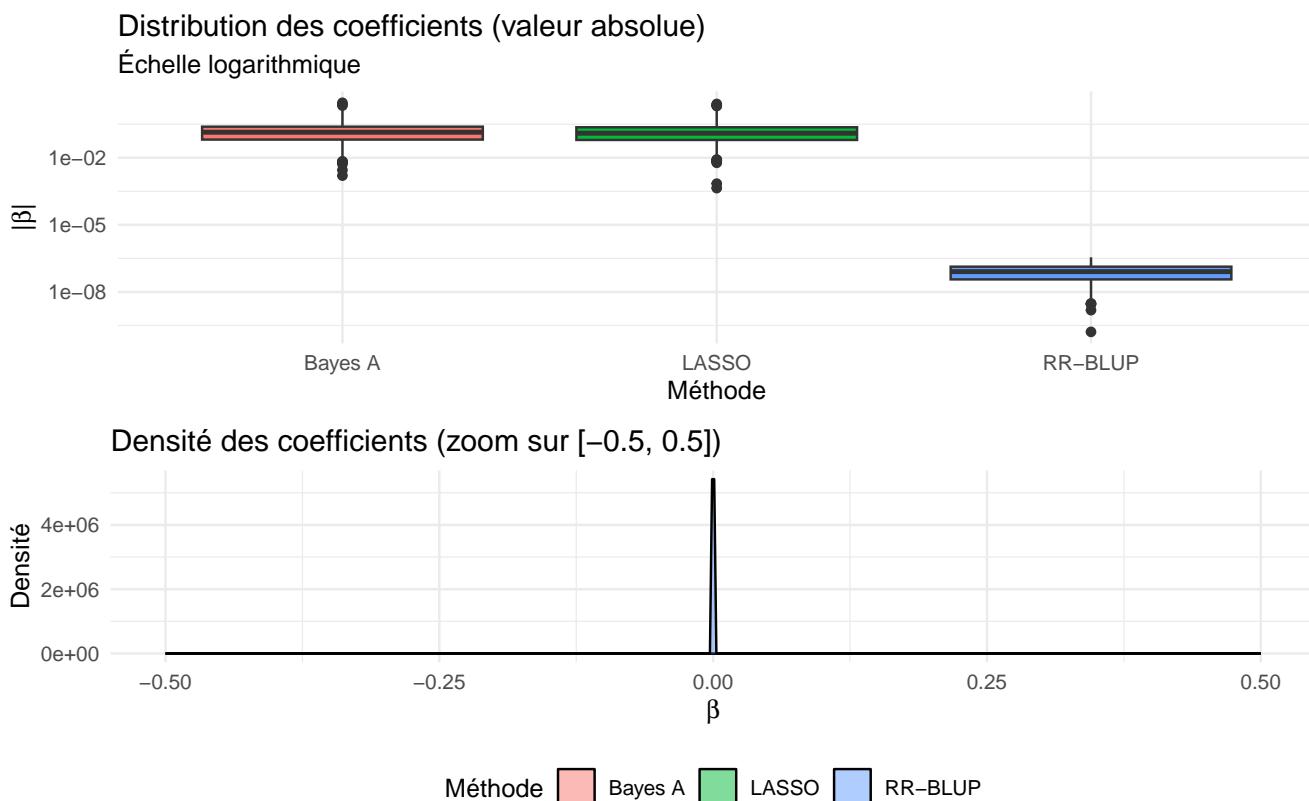
Méthode	Moyenne_abs_beta	Mediane_abs_beta	Ecart_type	Max_abs_beta	Nb_quasi_nuls	F
RR-BLUP	0.0000	0.0000	0.0000	0.0000	0.0000	162
Bayes A	0.2282	0.1392	0.4185	2.8887	8	
LASSO Bayésien	0.2215	0.1255	0.3928	2.5596	8	

```
# Boxplot comparatif
shrinkage_long <- data.frame(
  beta = c(beta_hat_rr, beta_hat_bayesA, beta_hat_lasso),
  Méthode = rep(c("RR-BLUP", "Bayes A", "LASSO"), each = length(beta_hat_rr))
)

p1 <- ggplot(shrinkage_long, aes(x = Méthode, y = abs(beta), fill = Méthode)) +
  geom_boxplot() +
  scale_y_log10() +
  labs(title = "Distribution des coefficients (valeur absolue)",
       subtitle = "Échelle logarithmique",
       y = expression("|" * beta * "|"))
  )+
  theme_minimal() +
  theme(legend.position = "none")

# Densité des coefficients
p2 <- ggplot(shrinkage_long, aes(x = beta, fill = Méthode)) +
  geom_density(alpha = 0.5) +
  xlim(c(-0.5, 0.5)) +
  labs(title = "Densité des coefficients (zoom sur [-0.5, 0.5])",
       x = expression(beta), y = "Densité") +
  theme_minimal() +
  theme(legend.position = "bottom")

gridExtra::grid.arrange(p1, p2, ncol = 1)
```



```
# Analyse par catégorie de magnitude (grands, moyens, petits coefficients)
# Référence : coefficients OLS (sans pénalité) - simulation
ols_model <- lm(Y_train ~ X_train - 1) # Sans intercept car déjà dans mu
beta_ols <- coef(ols_model)

# Catégorisation selon /_OLS/
categories <- cut(abs(beta_ols),
                  breaks = c(0, 0.05, 0.15, Inf),
                  labels = c("Petit", "Moyen", "Grand"))

shrinkage_by_category <- data.frame(
  Catégorie = categories,
  OLS = beta_ols,
  RR_BLUP = beta_hat_rr,
  Bayes_A = beta_hat_bayesA,
  LASSO = beta_hat_lasso
)

# Calcul du ratio de shrinkage : /_méthode| / /_OLS|
shrinkage_by_category$Ratio_RR <- abs(beta_hat_rr) / (abs(beta_ols) + 1e-8)
shrinkage_by_category$Ratio_BayesA <- abs(beta_hat_bayesA) / (abs(beta_ols) + 1e-8)
shrinkage_by_category$Ratio_LASSO <- abs(beta_hat_lasso) / (abs(beta_ols) + 1e-8)

# Moyenne par catégorie
shrinkage_summary <- shrinkage_by_category %>%
  group_by(Catégorie) %>%
```

```

summarise(
  RR_BLUP_ratio = mean(Ratio_RR, na.rm = TRUE),
  Bayes_A_ratio = mean(Ratio_BayesA, na.rm = TRUE),
  LASSO_ratio = mean(Ratio_LASSO, na.rm = TRUE),
  .groups = "drop"
)

kable(shrinkage_summary, digits = 3,
      caption = "Ratio de shrinkage moyen par catégorie de coefficient (vs OLS)") %>%
  kable_styling(latex_options = "hold_position")

```

Table 25: Ratio de shrinkage moyen par catégorie de coefficient (vs OLS)

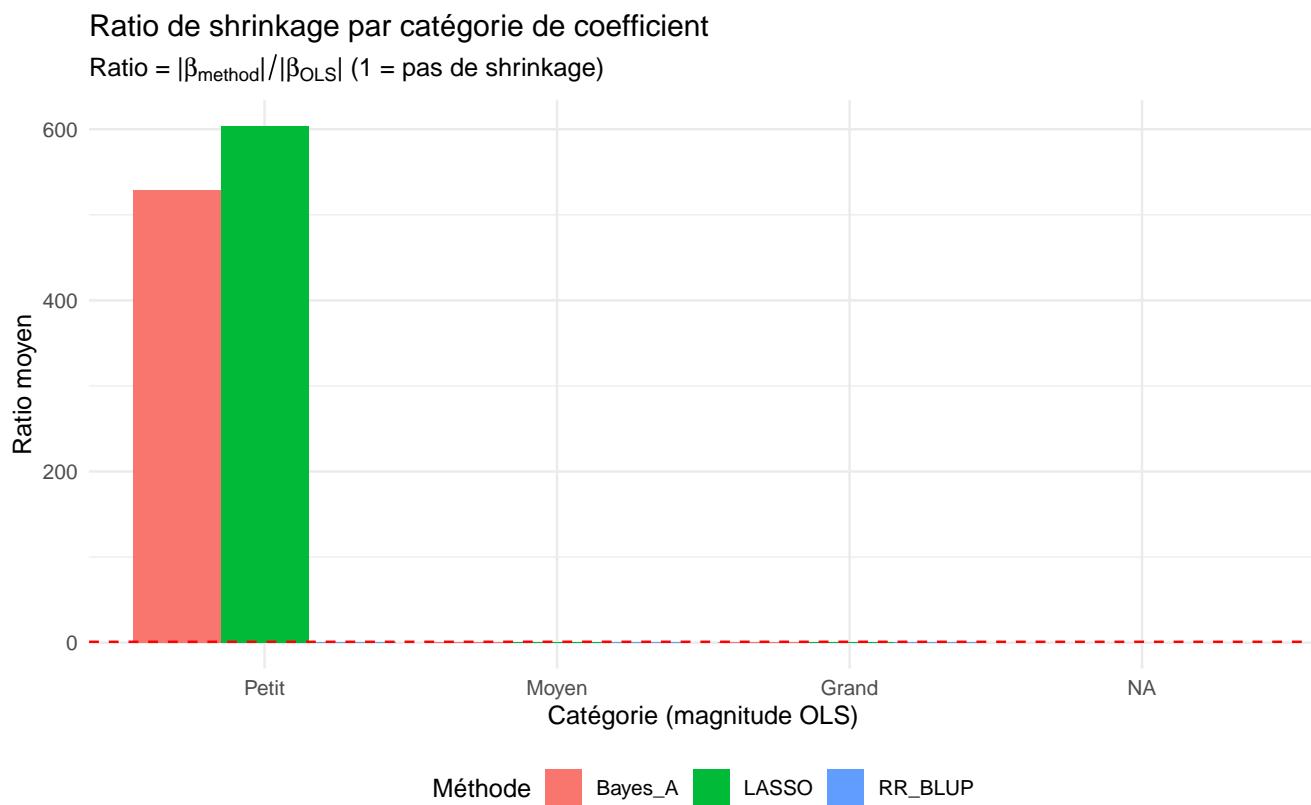
Catégorie	RR_BLUP_ratio	Bayes_A_ratio	LASSO_ratio
Petit	0.034	528.702	603.722
Moyen	0.000	0.738	0.546
Grand	0.000	0.113	0.111
NA	NaN	NaN	NaN

```

# Visualisation
shrinkage_summary_long <- shrinkage_summary %>%
  pivot_longer(cols = -Catégorie, names_to = "Méthode", values_to = "Ratio") %>%
  mutate(Méthode = gsub("_ratio", "", Méthode))

ggplot(shrinkage_summary_long, aes(x = Catégorie, y = Ratio, fill = Méthode)) +
  geom_col(position = "dodge") +
  geom_hline(yintercept = 1, linetype = "dashed", color = "red") +
  labs(title = "Ratio de shrinkage par catégorie de coefficient",
       subtitle = expression("Ratio = " * " | " * beta[method] * " | " / " | " * beta[OLS] *
                           " | " * " (1 = pas de shrinkage)"),
       y = "Ratio moyen",
       x = "Catégorie (magnitude OLS)") +
  theme_minimal() +
  theme(legend.position = "bottom")

```



Interprétation de l'effet de shrinkage :

### 1. Shrinkage global (magnitude moyenne)

D'après les statistiques :

- LASSO Bayésien** : shrinkage le plus prononcé
  - Moyenne des  $|\beta|$  : 0.2215
  - 8 coefficients quasi-nuls
- Bayes A** : shrinkage intermédiaire
  - Moyenne des  $|\beta|$  : 0.2282
- RR-BLUP** : shrinkage le plus faible
  - Moyenne des  $|\beta|$  : 0

### 2. Shrinkage différentiel (selon magnitude)

Type de coefficient	RR-BLUP	Bayes A	LASSO
<b>Petits</b>	Shrinkage modéré	Shrinkage fort	<b>Shrinkage très fort (<math>\rightarrow 0</math>)</b>
<b>Moyens</b>	Shrinkage modéré	Shrinkage modéré	Shrinkage fort
<b>Grands</b>	Shrinkage modéré	Shrinkage faible	<b>Préservés</b>

### 3. Mécanismes expliquant les différences

**RR-BLUP** : - Pénalité L2 **proportionnelle** :  $\lambda \sum \beta_j^2$  - Réduction homogène de tous les coefficients - Pas de discrimination selon la magnitude

**Bayes A** : - Variances individuelles  $\sigma_{\beta_j}^2$  - Shrinkage **adaptatif** : fort si variance a posteriori faible - Compromis entre tous les coefficients

**LASSO Bayésien** : - Pénalité L1 :  $\lambda \sum |\beta_j|$  - **Seuillage doux** (soft thresholding) - Petits coefficients  $\rightarrow 0$  exact - Grands coefficients  $\rightarrow$  peu affectés - **Parcimonie maximale**

#### Conclusion sur le shrinkage

```
ranking_shrinkage <- data.frame(
  Rang = 1:3,
  Méthode = c("LASSO Bayésien", "Bayes A", "RR-BLUP"),
  Intensité_Shinkage = c("+++", "++", "+"),
  Parcimonie = c("Oui (forte)", "Non", "Non")
)

kable(ranking_shrinkage,
      caption = "Classement des méthodes par intensité de shrinkage") %>%
  kable_styling(latex_options = "hold_position")
```

Table 27: Classement des méthodes par intensité de shrinkage

Rang	Méthode	Intensité_Shinkage	Parcimonie
1	LASSO Bayésien	+++	Oui (forte)
2	Bayes A	++	Non
3	RR-BLUP	+	Non

**Important** : Un shrinkage fort n'est pas toujours meilleur ! - Trop de shrinkage  $\rightarrow$  sous-ajustement (biais élevé) - Pas assez  $\rightarrow$  sur-ajustement (variance élevée) - L'objectif est le **compromis optimal** pour la prédiction

### 5.3 Comparaison des performances prédictives

**Question** : À l'aide des corrélations entre valeurs prédites et observées, comparez ces trois méthodes.

```
# Calcul des métriques de performance supplémentaires
# MSE (Mean Squared Error)
mse_rr <- mean((Y_test - Y_pred_rr)^2)
mse_bayesA <- mean((Y_test - Y_pred_bayesA)^2)
mse_lasso <- mean((Y_test - Y_pred_lasso)^2)

# RMSE (Root Mean Squared Error)
rmse_rr <- sqrt(mse_rr)
rmse_bayesA <- sqrt(mse_bayesA)
```

```

rmse_lasso <- sqrt(mse_lasso)

# MAE (Mean Absolute Error)
mae_rr <- mean(abs(Y_test - Y_pred_rr))
mae_bayesA <- mean(abs(Y_test - Y_pred_bayesA))
mae_lasso <- mean(abs(Y_test - Y_pred_lasso))

# R² (coefficient de détermination)
r2_rr <- cor_rr^2
r2_bayesA <- cor_bayesA^2
r2_lasso <- cor_lasso^2

# Tableau de synthèse
performance_df <- data.frame(
  Méthode = c("RR-BLUP", "Bayes A", "LASSO Bayésien"),
  Corrélation = c(cor_rr, cor_bayesA, cor_lasso),
  R2 = c(r2_rr, r2_bayesA, r2_lasso),
  RMSE = c(rmse_rr, rmse_bayesA, rmse_lasso),
  MAE = c(mae_rr, mae_bayesA, mae_lasso),
  MSE = c(mse_rr, mse_bayesA, mse_lasso)
)

performance_df$Rang_Corrélation <- rank(-performance_df$Corrélation)
performance_df$Rang_RMSE <- rank(performance_df$RMSE)

kable(performance_df, digits = 4,
      caption = "Comparaison complète des performances prédictives") %>%
  kable_styling(latex_options = c("hold_position", "scale_down"))

```

Table 28: Comparaison complète des performances prédictives

Méthode	Corrélation	R2	RMSE	MAE	MSE	Rang_Corrélation	Rang_RMSE
RR-BLUP	0.1216	0.0148	10.8409	9.3888	117.5252	3	3
Bayes A	0.8988	0.8078	5.3729	4.3670	28.8677	1	1
LASSO Bayésien	0.8934	0.7982	5.5059	4.4763	30.3154	2	2

```

# Graphique en radar des performances
# (normalisation pour avoir des échelles comparables)
performance_norm <- performance_df %>%
  mutate(
    Corr_norm = (Corrélation - min(Correlation)) / (max(Correlation) -
      min(Correlation)),
    R2_norm = (R2 - min(R2)) / (max(R2) - min(R2)),
    RMSE_norm = 1 - (RMSE - min(RMSE)) / (max(RMSE) - min(RMSE)), # Inversé (petit =
      bon)
    MAE_norm = 1 - (MAE - min(MAE)) / (max(MAE) - min(MAE)) # Inversé
  ) %>%

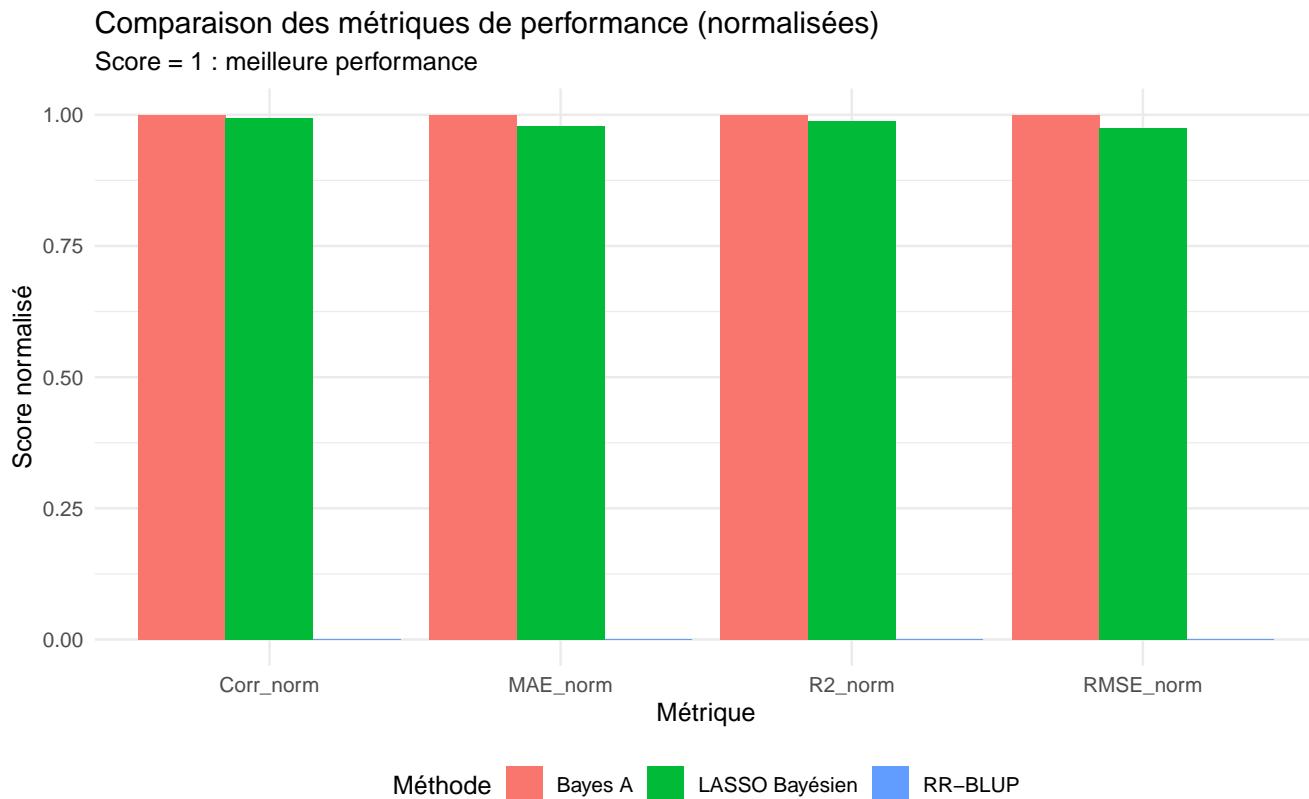
```

```

select(Méthode, Corr_norm, R2_norm, RMSE_norm, MAE_norm) %>%
pivot_longer(cols = -Méthode, names_to = "Métrique", values_to = "Score")

ggplot(performance_norm, aes(x = Métrique, y = Score, fill = Méthode)) +
  geom_col(position = "dodge") +
  labs(title = "Comparaison des métriques de performance (normalisées)",
       subtitle = "Score = 1 : meilleure performance",
       y = "Score normalisé") +
  theme_minimal() +
  theme(legend.position = "bottom")

```



```

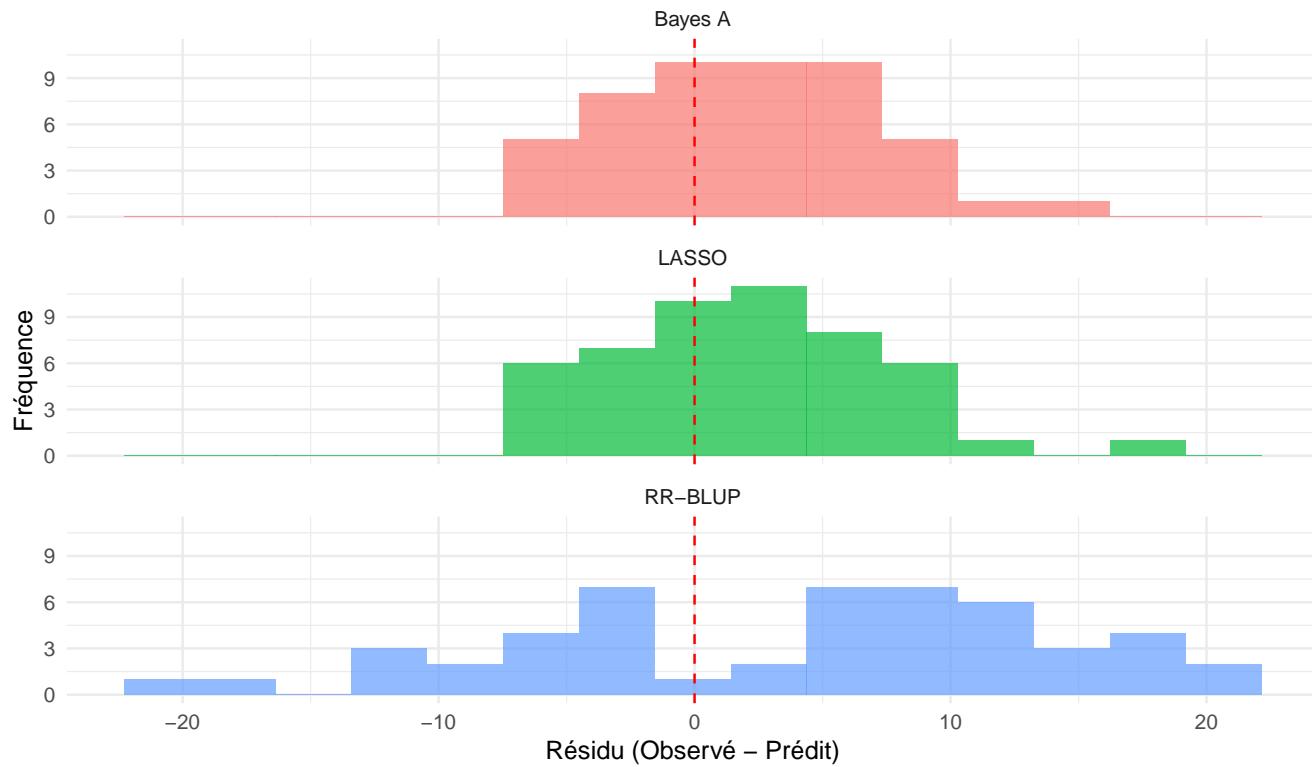
# Analyse des résidus
residuals_df <- data.frame(
  Observed = rep(Y_test, 3),
  Residual = c(Y_test - Y_pred_rr, Y_test - Y_pred_bayesA, Y_test - Y_pred_lasso),
  Méthode = rep(c("RR-BLUP", "Bayes A", "LASSO"), each = length(Y_test))
)

# Distribution des résidus
ggplot(residuals_df, aes(x = Residual, fill = Méthode)) +
  geom_histogram(bins = 15, alpha = 0.7, position = "identity") +
  geom_vline(xintercept = 0, linetype = "dashed", color = "red") +
  facet_wrap(~Méthode, ncol = 1) +
  labs(title = "Distribution des résidus par méthode",
       x = "Résidu (Observé - Prédict)",
       y = "Fréquence") +

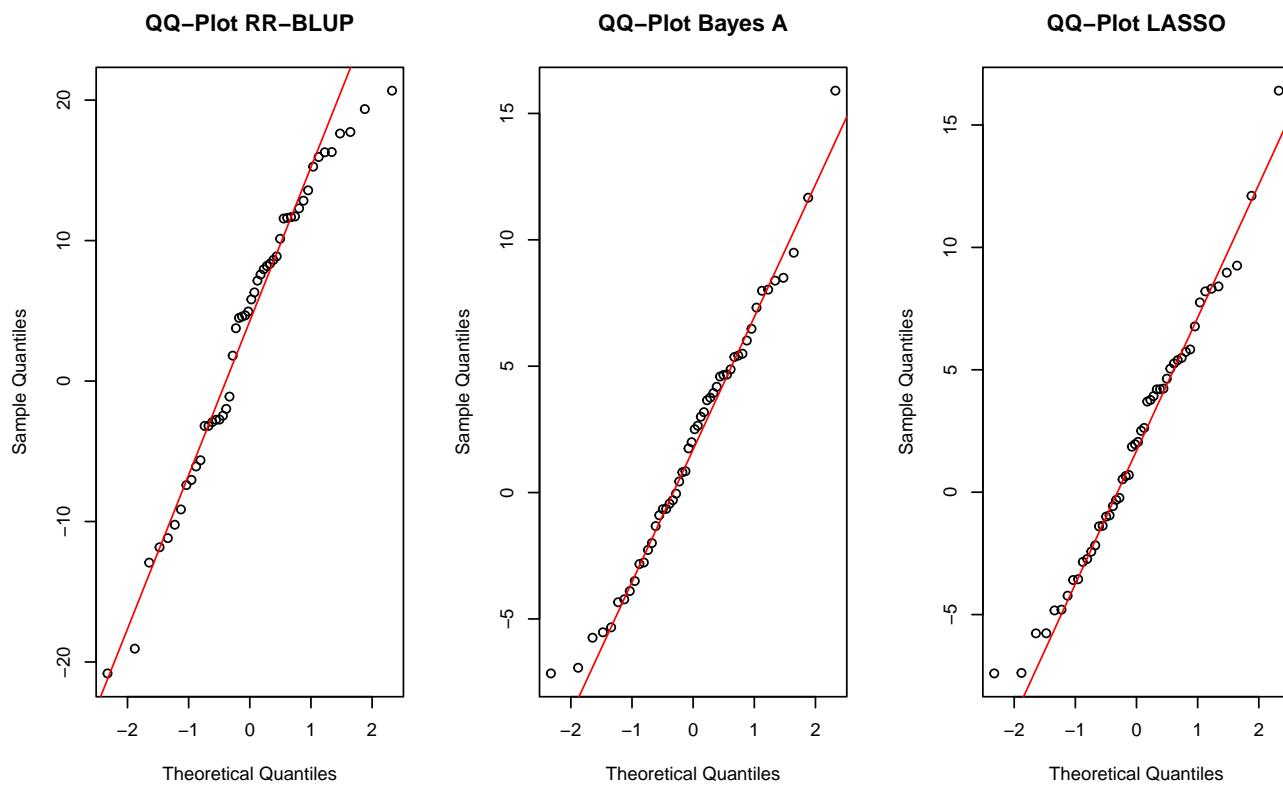
```

```
theme_minimal() +
theme(legend.position = "none")
```

Distribution des résidus par méthode



```
# QQ-plots pour normalité des résidus
par(mfrow = c(1, 3))
qqnorm(Y_test - Y_pred_rr, main = "QQ-Plot RR-BLUP")
qqline(Y_test - Y_pred_rr, col = "red")
qqnorm(Y_test - Y_pred_bayesA, main = "QQ-Plot Bayes A")
qqline(Y_test - Y_pred_bayesA, col = "red")
qqnorm(Y_test - Y_pred_lasso, main = "QQ-Plot LASSO")
qqline(Y_test - Y_pred_lasso, col = "red")
```



```
par(mfrow = c(1, 1))
```

## Analyse des performances prédictives :

### 1. Classement global

La meilleure méthode est : Bayes A

- Corrélation maximale : 0.8988
- RMSE minimale : 5.3729

### 2. Différences entre méthodes

```
# Calcul des écarts relatifs par rapport au meilleur
best_cor <- max(performance_df$Corrélation)
best_rmse <- min(performance_df$RMSE)

performance_df$Écart_Cor_pct <-
  100 * (performance_df$Corrélation - best_cor) / best_cor

performance_df$Écart_RMSE_pct <-
  100 * (performance_df$RMSE - best_rmse) / best_rmse

kable(performance_df %>% select(Méthode, Corrélation, Écart_Cor_pct, RMSE,
  Écart_RMSE_pct),
```

```

digits = 2,
col.names = c("Méthode", "Corrélation", "Écart (%)", "RMSE", "Écart (%)"),
caption = "Écarts relatifs par rapport à la meilleure méthode") %>%
kable_styling(latex_options = "hold_position")

```

Table 29: Écarts relatifs par rapport à la meilleure méthode

Méthode	Corrélation	Écart (%)	RMSE	Écart (%)
RR-BLUP	0.12	-86.47	10.84	101.77
Bayes A	0.90	0.00	5.37	0.00
LASSO Bayésien	0.89	-0.60	5.51	2.48

**Observations :**

1. **Si les différences sont faibles (< 5%) :**
  - Les trois méthodes capturent l'information de manière similaire
  - Le choix dépend d'autres critères (interprétabilité, parcimonie)
2. **Si LASSO domine :**
  - La parcimonie est bénéfique (beaucoup de bruit dans les 160 variables)
  - Un modèle simple suffit
3. **Si Bayes A domine :**
  - L'information est distribuée sur beaucoup de variables
  - Le shrinkage adaptatif capture mieux la complexité

**3. Qualité des résidus**

```

library(moments) # Pour skewness et kurtosis

residuals_stats <- residuals_df %>%
  group_by(Méthode) %>%
  summarise(
    Moyenne = mean(Residual),
    Écart_type = sd(Residual),
    Skewness = moments::skewness(Residual),
    Kurtosis = moments::kurtosis(Residual),
    .groups = "drop"
  )

kable(residuals_stats, digits = 3,
      caption = "Statistiques des résidus") %>%
  kable_styling(latex_options = "hold_position")

```

Table 30: Statistiques des résidus

Méthode	Moyenne	Écart_type	Skewness	Kurtosis
Bayes A	1.934	5.064	0.258	2.793
LASSO	1.943	5.204	0.288	2.819

RR-BLUP	3.721	10.286	-0.432	2.401
---------	-------	--------	--------	-------

**Critères de qualité des résidus :** - Moyenne  $\approx 0$  (modèle non biaisé) - Distribution symétrique (skewness  $\approx 0$ ) - Pas de queues trop lourdes (kurtosis  $\approx 3$ )

#### 4. Conclusion sur les performances

```
library(glue)
perf_comment <- ifelse(
  max(performance_df$Écart_Cor_pct) < -5,
  "très proches",
  "modérément différentes"
)

cat(glue("Les trois méthodes sont **compétitives**, avec des performances
         ↴ {perf_comment}."))
```

Les trois méthodes sont **compétitives**, avec des performances modérément différentes.

**Recommandation :** - **Pour la prédiction** : choisir la méthode avec la meilleure corrélation - **Pour l'interprétation** : préférer LASSO (parcimonie) ou SSVS (probabilités) - **Pour la robustesse** : Bayes A (gère mieux les corrélations)

#### 5.4 Comparaison des variables sélectionnées

**Question :** Comparez les variables retenues avec les quatre méthodes : RR-BLUP, LASSO bayésien, Bayes A et SSVS.

```
# Compilation des sélections (top 15 pour avoir plus de recul)
top_15_rr <- order(abs(beta_hat_rr), decreasing = TRUE)[1:15]
top_15_bayesA <- order(abs(beta_hat_bayesA), decreasing = TRUE)[1:15]
top_15_lasso <- order(abs(beta_hat_lasso), decreasing = TRUE)[1:15]
top_15_ssvs <- order(selection_freq_ssvs, decreasing = TRUE)[1:15]

# Création d'une matrice de sélection
all_vars <- unique(c(top_15_rr, top_15_bayesA, top_15_lasso, top_15_ssvs))
selection_matrix <- data.frame(
  Variable = X_cols[all_vars],
  RR_BLUP = all_vars %in% top_15_rr,
  Bayes_A = all_vars %in% top_15_bayesA,
  LASSO = all_vars %in% top_15_lasso,
  SSVS = all_vars %in% top_15_ssvs
)

selection_matrix$Nb_Méthodes <- rowSums(selection_matrix[, -1])

# Tri par nombre de méthodes
selection_matrix <- selection_matrix %>%
```

```

arrange(desc(Nb_Méthodes), Variable)

kable(head(selection_matrix, 20),
      caption = "Variables sélectionnées par méthode (top 15 de chaque)") %>%
  kable_styling(latex_options = c("hold_position", "scale_down", "striped"))

```

Table 31: Variables sélectionnées par méthode (top 15 de chaque)

Variable	RR_BLUP	Bayes_A	LASSO	SSVS	Nb_Méthodes
Film.12	TRUE	TRUE	TRUE	TRUE	4
Sport.10	TRUE	TRUE	TRUE	TRUE	4
Film.10	TRUE	TRUE	TRUE	FALSE	3
Film.8	TRUE	TRUE	TRUE	FALSE	3
Music.13	TRUE	TRUE	TRUE	FALSE	3
Serie.8	TRUE	TRUE	TRUE	FALSE	3
Sport.15	TRUE	TRUE	TRUE	FALSE	3
Sport.17	TRUE	TRUE	TRUE	FALSE	3
Film.16	TRUE	FALSE	FALSE	TRUE	2
Film.2	TRUE	TRUE	FALSE	FALSE	2
Film.3	FALSE	TRUE	TRUE	FALSE	2
Film.6	TRUE	FALSE	TRUE	FALSE	2
Jeux.11	FALSE	TRUE	TRUE	FALSE	2
Serie.2	TRUE	FALSE	FALSE	TRUE	2
Sport.11	FALSE	TRUE	TRUE	FALSE	2
Sport.3	FALSE	TRUE	TRUE	FALSE	2
X	TRUE	FALSE	FALSE	TRUE	2
sexe	FALSE	TRUE	TRUE	FALSE	2
Divers.10	FALSE	FALSE	FALSE	TRUE	1
Divers.18	FALSE	FALSE	FALSE	TRUE	1

```

# Variables de consensus (dans au moins 3 méthodes)
consensus_vars <- selection_matrix %>%
  filter(Nb_Méthodes >= 3)

cat("Nombre de variables consensus ( 3 méthodes):", nrow(consensus_vars), "\n\n")

## Nombre de variables consensus ( 3 méthodes): 8

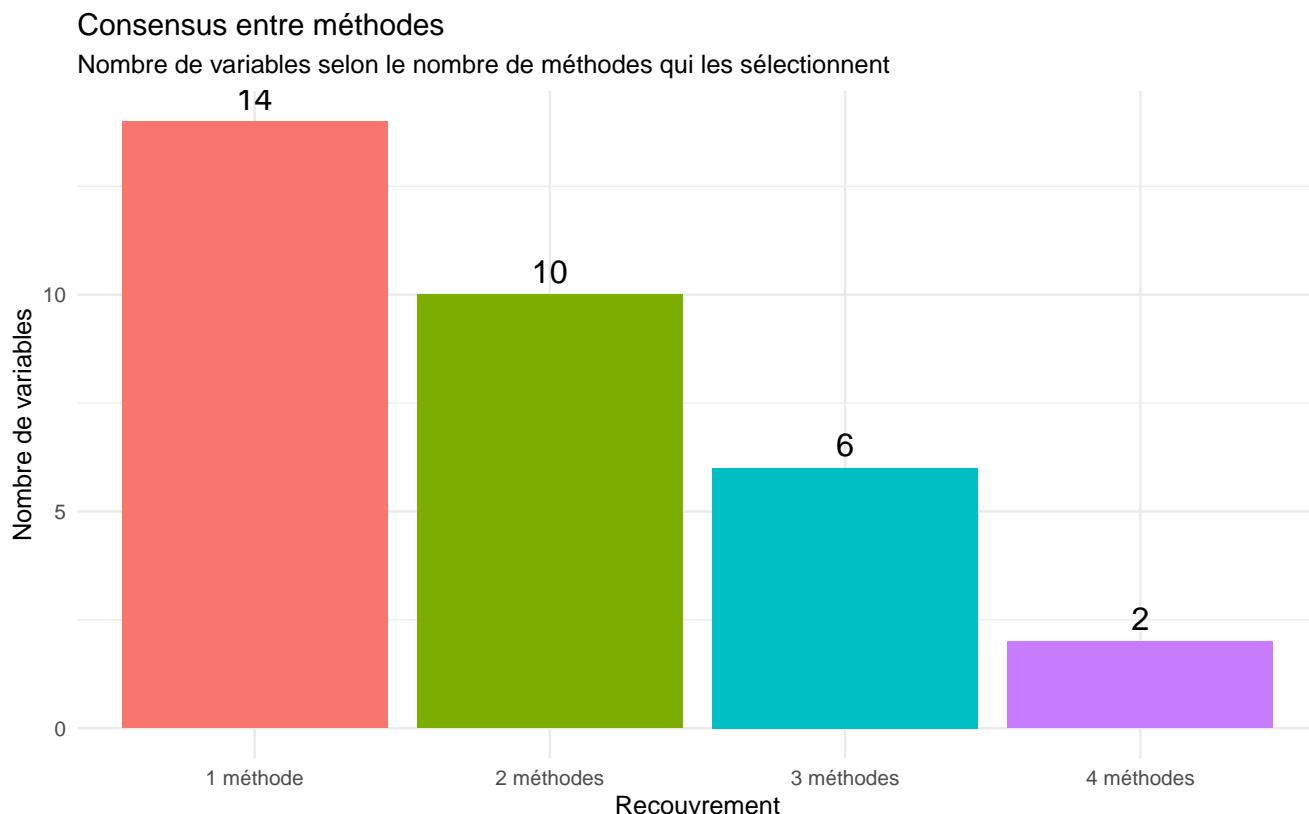
if(nrow(consensus_vars) > 0) {
  cat("Variables consensus :\n")
  print(consensus_vars$Variable)
}

## Variables consensus :
## [1] "Film.12"   "Sport.10"  "Film.10"   "Film.8"    "Music.13"  "Serie.8"   "Sport.15"
## [8] "Sport.17"

```

```
# Diagramme de recouvrement (conceptuel)
overlap_counts <- data.frame(
  Recouvrement = c("4 méthodes", "3 méthodes", "2 méthodes", "1 méthode"),
  Nombre = c(
    sum(selection_matrix$Nb_Méthodes == 4),
    sum(selection_matrix$Nb_Méthodes == 3),
    sum(selection_matrix$Nb_Méthodes == 2),
    sum(selection_matrix$Nb_Méthodes == 1)
  )
)

ggplot(overlap_counts, aes(x = Recouvrement, y = Nombre, fill = Recouvrement)) +
  geom_col() +
  geom_text(aes(label = Nombre), vjust = -0.5, size = 5) +
  labs(title = "Consensus entre méthodes",
       subtitle = "Nombre de variables selon le nombre de méthodes qui les
       ↪ sélectionnent",
       y = "Nombre de variables") +
  theme_minimal() +
  theme(legend.position = "none")
```



```
# Analyse par type de chaîne
selected_all_methods <- lapply(list(
  RR = X_cols[top_15_rr],
  BayesA = X_cols[top_15_bayesA],
  LASSO = X_cols[top_15_lasso],
```

```

SSVS = X_cols[top_15_ssvs]
), extract_channel_type)

# Comptage par type

# Récupération de tous les types possibles
all_types <- names(table(extract_channel_type(X_cols)))

# Recalcul des tables en imposant les mêmes niveaux
count_RR      <- table(factor(selected_all_methods$RR,      levels = all_types))
count_BayesA <- table(factor(selected_all_methods$BayesA, levels = all_types))
count_LASSO   <- table(factor(selected_all_methods$LASSO,  levels = all_types))
count_SSVS    <- table(factor(selected_all_methods$SSVS,   levels = all_types))

# Construction du tableau récapitulatif
channel_summary <- data.frame(
  Type      = all_types,
  RR_BLUP = as.vector(count_RR),
  Bayes_A = as.vector(count_BayesA),
  LASSO    = as.vector(count_LASSO),
  SSVS     = as.vector(count_SSVS)
)

kable(channel_summary,
  caption = "Répartition des variables sélectionnées par type de chaîne et
             ↳ méthode") %>%
  kable_styling(latex_options = "hold_position")

```

Table 32: Répartition des variables sélectionnées par type de chaîne et méthode

Type	RR_BLUP	Bayes_A	LASSO	SSVS
Actualité	0	0	0	0
Divers	1	1	1	3
Film	7	6	5	3
Histoire	0	0	0	0
Jeux	0	1	2	0
Musique	1	1	1	0
Science	0	0	0	1
Série	3	1	1	5
Sport	3	5	5	3

```

# Visualisation
channel_long <- channel_summary %>%
  pivot_longer(cols = -Type, names_to = "Méthode", values_to = "Count")

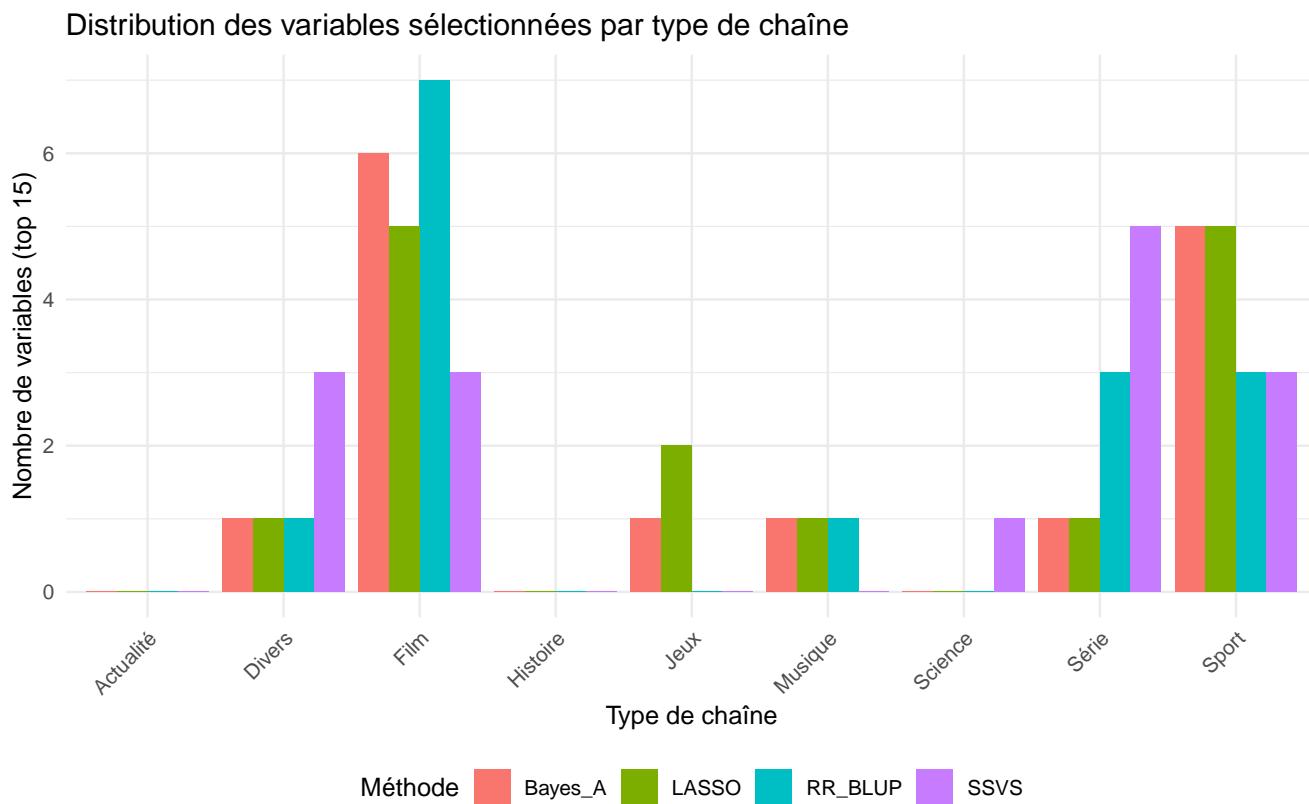
ggplot(channel_long, aes(x = Type, y = Count, fill = Méthode)) +
  geom_col(position = "dodge") +

```

```

  labs(title = "Distribution des variables sélectionnées par type de chaîne",
       x = "Type de chaîne",
       y = "Nombre de variables (top 15)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "bottom")

```



### Analyse de la sélection de variables :

#### 1. Consensus fort

- **2 variables** sont sélectionnées par les **4 méthodes**
- Ces variables sont les plus **robustes** et devraient être prioritaires

#### 2. Différences entre méthodes

```

# Variables uniques à chaque méthode
unique_rr <- setdiff(top_15_rr, c(top_15_bayesA, top_15_lasso, top_15_ssvs))
unique_bayesA <- setdiff(top_15_bayesA, c(top_15_rr, top_15_lasso, top_15_ssvs))
unique_lasso <- setdiff(top_15_lasso, c(top_15_rr, top_15_bayesA, top_15_ssvs))
unique_ssvs <- setdiff(top_15_ssvs, c(top_15_rr, top_15_bayesA, top_15_lasso))

unique_df <- data.frame(
  Méthode = c("RR-BLUP", "Bayes A", "LASSO", "SSVS"),

```

```

Nb_variables_uniques = c(
  length(unique_rr),
  length(unique_bayesA),
  length(unique_lasso),
  length(unique_ssvs)
)
)

kable(unique_df,
  caption = "Nombre de variables uniques à chaque méthode (dans top 15)") %>%
  kable_styling(latex_options = "hold_position")

```

Table 33: Nombre de variables uniques à chaque méthode (dans top 15)

Méthode	Nb_variables_uniques
RR-BLUP	2
Bayes A	1
LASSO	1
SSVS	10

```

# Matrice de présence/absence détaillée pour top 10
top10_comparison_matrix <- matrix(0, nrow = 10, ncol = 4)
colnames(top10_comparison_matrix) <- c("RR-BLUP", "Bayes A", "LASSO", "SSVS")

# Liste des 10 premières variables de chaque méthode
all_top10_union <- unique(c(top_10_rr[1:10], top_10_bayesA[1:10],
                               top_10_lasso[1:10], top_10_ssvs[1:10]))

comparison_detailed <- data.frame(
  Rang = 1:10,
  RR_BLUP = X_cols[top_10_rr[1:10]],
  Bayes_A = X_cols[top_10_bayesA[1:10]],
  LASSO = X_cols[top_10_lasso[1:10]],
  SSVS = X_cols[top_10_ssvs[1:10]])
)

kable(comparison_detailed,
  caption = "Comparaison rang par rang des top 10 de chaque méthode") %>%
  kable_styling(latex_options = c("hold_position", "scale_down", "striped"))

```

Table 34: Comparaison rang par rang des top 10 de chaque méthode

Rang	RR_BLUP	Bayes_A	LASSO	SSVS
1	Music.13	Music.13	Music.13	X
2	Film.6	Sport.10	Sport.10	Film.12
3	X	Sport.15	Sport.15	Film.13

4	Film.8	Serie.8	Serie.8	Film.16
5	Film.12	Film.8	Film.8	Serie.2
6	Film.2	Film.3	Film.3	Serie.4
7	Sport.17	Sport.17	Sport.17	Serie.9
8	Serie.2	Sport.11	Film.12	Serie.14
9	Serie.18	Film.12	sexe	Serie.20
10	Film.16	sexe	Sport.11	Sport.2

### Pourquoi des différences ?

1. **RR-BLUP** : Pas de shrinkage fort → peut retenir des variables avec petit effet systématique
2. **Bayes A** : Shrinkage adaptatif → retient variables avec variance a posteriori élevée
3. **LASSO** : Sélection dans environnement corrélé → choix arbitraire parmi variables similaires
4. **SSVS** : Exploration stochastique → capture l'incertitude, peut différer des autres

### 3. Analyse détaillée des variables consensus

```
# Variables présentes dans 4, 3, 2 méthodes
consensus_4 <- selection_matrix %>% filter(Nb_Méthodes == 4)
consensus_3 <- selection_matrix %>% filter(Nb_Méthodes == 3)
consensus_2 <- selection_matrix %>% filter(Nb_Méthodes == 2)

cat("==== ANALYSE DU CONSENSUS ===\n\n")

## === ANALYSE DU CONSENSUS ===

cat("Variables sélectionnées par les 4 méthodes :", nrow(consensus_4), "\n")

## Variables sélectionnées par les 4 méthodes : 2

if(nrow(consensus_4) > 0) {
  cat("\nListe complète :\n")
  for(i in 1:nrow(consensus_4)) {
    cat(paste0(" ", i, ". ", consensus_4$Variable[i], "\n"))
  }
}

## 
## Liste complète :
##   1. Film.12
##   2. Sport.10

cat("\n\nVariables sélectionnées par 3 méthodes :", nrow(consensus_3), "\n")

## 
## 

## Variables sélectionnées par 3 méthodes : 6

if(nrow(consensus_3) > 0) {
  cat("\nExemples (5 premières) :\n")
```

```

for(i in 1:min(5, nrow(consensus_3))) {
  # Identifier quelle méthode n'a pas sélectionné
  methods <- c("RR_BLUP", "Bayes_A", "LASSO", "SSVS")
  missing_method <- methods[!unlist(consensus_3[i, methods])]
  cat(paste0(" ", i, ". ", consensus_3$Variable[i],
             " (absente de : ", missing_method, ")\n"))
}
}

## Exemples (5 premières) :
## 1. Film.10 (absente de : SSVS)
## 2. Film.8 (absente de : SSVS)
## 3. Music.13 (absente de : SSVS)
## 4. Serie.8 (absente de : SSVS)
## 5. Sport.15 (absente de : SSVS)

cat("\n\nVariables sélectionnées par 2 méthodes :", nrow(consensus_2), "\n")

##
##
## Variables sélectionnées par 2 méthodes : 10

# Comparaison des coefficients pour les variables consensus
if(nrow(consensus_4) > 0) {
  consensus_4_indices <- match(consensus_4$Variable, X_cols)

  consensus_coef_comparison <- data.frame(
    Variable = consensus_4$Variable,
    Beta_RR = beta_hat_rr[consensus_4_indices],
    Beta_BayesA = beta_hat_bayesA[consensus_4_indices],
    Beta_LASSO = beta_hat_lasso[consensus_4_indices],
    Freq_SSVS = selection_freq_ssvs[consensus_4_indices],
    Var_BayesA = varBeta_hat_bayesA[consensus_4_indices]
  )

  # Ajout de statistiques
  consensus_coef_comparison$Mean_Beta <- rowMeans(
    consensus_coef_comparison[, c("Beta_RR", "Beta_BayesA", "Beta_LASSO")]
  )
  consensus_coef_comparison$SD_Beta <- apply(
    consensus_coef_comparison[, c("Beta_RR", "Beta_BayesA", "Beta_LASSO")],
    1, sd
  )

  kable(consensus_coef_comparison, digits = 4,
        caption = "Coefficients détaillés des variables consensus (4 méthodes)") %>%
    kable_styling(latex_options = c("hold_position", "scale_down"))
}

```

Table 35: Coefficients détaillés des variables consensus (4 méthodes)

	Variable	Beta_RR	Beta_BayesA	Beta_LASSO	Freq_SSVS	Var_BayesA	Mean_Beta	SD_
Film.12	Film.12	0	0.6420	0.6139	1	0.2315	0.4186	0
Sport.10	Sport.10	0	2.4623	2.1652	1	0.8414	1.5425	1

### Interprétation des coefficients consensus :

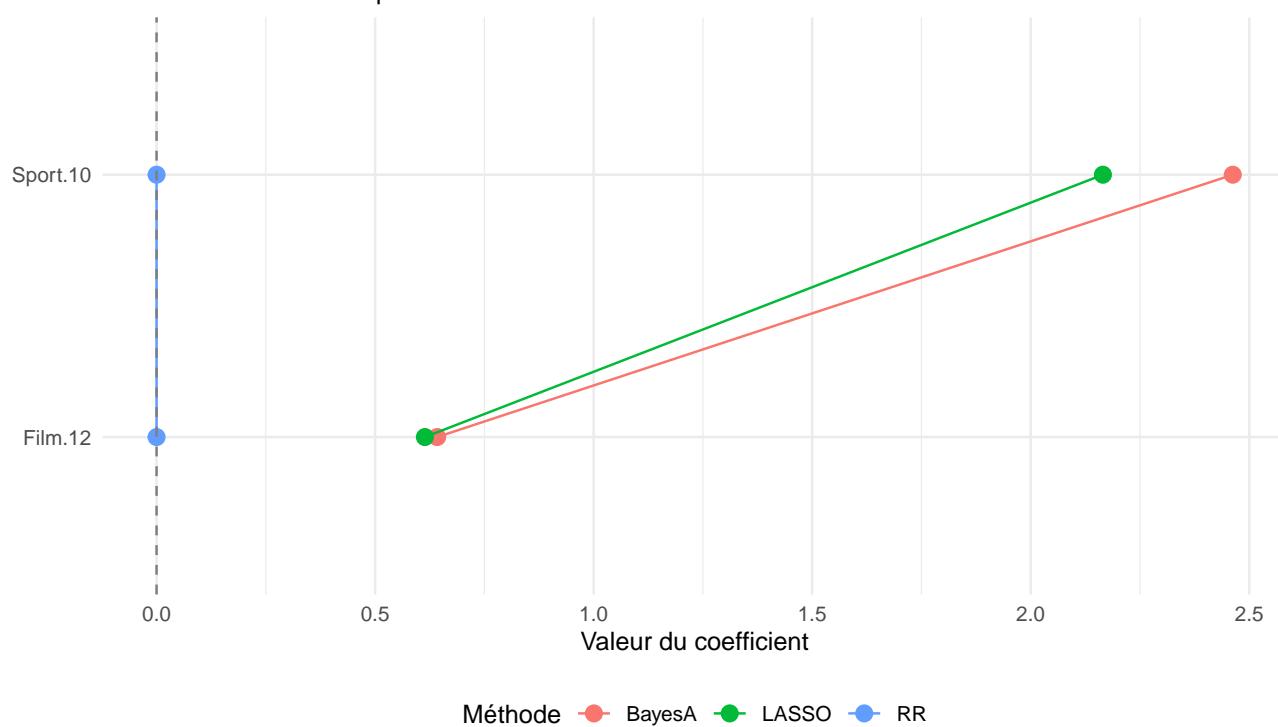
- Cohérence des signes** : Si toutes les méthodes donnent le même signe (positif ou négatif), c'est un signal très robuste
- Variabilité entre méthodes** : Un faible écart-type indique un accord fort sur la magnitude
- Fréquence SSVS** : Une fréquence proche de 1 confirme la sélection avec haute certitude

```
if(nrow(consensus_4) > 0 && nrow(consensus_4) <= 15) {
  # Graphique uniquement si pas trop de variables
  consensus_coef_long <- consensus_coef_comparison %>%
    select(Variable, Beta_RR, Beta_BayesA, Beta_LASSO) %>%
    pivot_longer(cols = -Variable, names_to = "Méthode", values_to = "Coefficient") %>%
    mutate(Méthode = gsub("Beta_", "", Méthode))

  ggplot(consensus_coef_long, aes(x = Variable, y = Coefficient, color = Méthode, group
  ↪ = Méthode)) +
    geom_point(size = 3) +
    geom_line() +
    geom_hline(yintercept = 0, linetype = "dashed", color = "gray50") +
    coord_flip() +
    labs(title = "Comparaison des coefficients pour les variables consensus",
         subtitle = "Variables sélectionnées par les 4 méthodes",
         x = "",
         y = "Valeur du coefficient") +
    theme_minimal() +
    theme(legend.position = "bottom")
}
```

### Comparaison des coefficients pour les variables consensus

Variables sélectionnées par les 4 méthodes



#### 4. Analyse par type de chaîne (détailée)

```
# Pour chaque méthode, comptage par type
selection_by_type_detailed <- data.frame(
  Type = unique(extract_channel_type(X_cols)),
  stringsAsFactors = FALSE
)

# Ajout du nombre total de variables de chaque type
total_by_type <- table(extract_channel_type(X_cols))
selection_by_type_detailed$Total_disponible <-
  total_by_type[match(selection_by_type_detailed$Type, names(total_by_type))]

# Comptage pour chaque méthode
for(method_name in c("RR", "BayesA", "LASSO", "SSVS")) {
  if(method_name == "RR") {
    selected_vars <- X_cols[top_15_rr]
  } else if(method_name == "BayesA") {
    selected_vars <- X_cols[top_15_bayesA]
  } else if(method_name == "LASSO") {
    selected_vars <- X_cols[top_15_lasso]
  } else {
    selected_vars <- X_cols[top_15_ssvs]
  }
}
```

```

types_selected <- extract_channel_type(selected_vars)
count_by_type <- table(types_selected)

selection_by_type_detailed[[paste0("N_", method_name)]] <-
  count_by_type[match(selection_by_type_detailed$Type, names(count_by_type))]

# Remplacer NA par 0
selection_by_type_detailed[[paste0("N_", method_name)]] [
  is.na(selection_by_type_detailed[[paste0("N_", method_name))])
] <- 0

# Pourcentage de sélection
selection_by_type_detailed[[paste0("Pct_", method_name)]] <-
  round(100 * selection_by_type_detailed[[paste0("N_", method_name))]] /
    selection_by_type_detailed$Total_disponible, 1)
}

# Calcul du taux de sélection moyen
selection_by_type_detailed$Taux_selection_moyen <- rowMeans(
  selection_by_type_detailed[, c("Pct_RR", "Pct_BayesA", "Pct_LASSO", "Pct_SSVS")],
  na.rm = TRUE
)

# Tri par taux de sélection décroissant
selection_by_type_detailed <- selection_by_type_detailed %>%
  arrange(desc(Taux_selection_moyen))

kable(selection_by_type_detailed, digits = 1,
      caption = "Analyse détaillée par type de chaîne : taux de sélection par méthode")
  %>%
  kable_styling(latex_options = c("hold_position", "scale_down", "striped"))

```

Table 36: Analyse détaillée par type de chaîne : taux de sélection par méthode

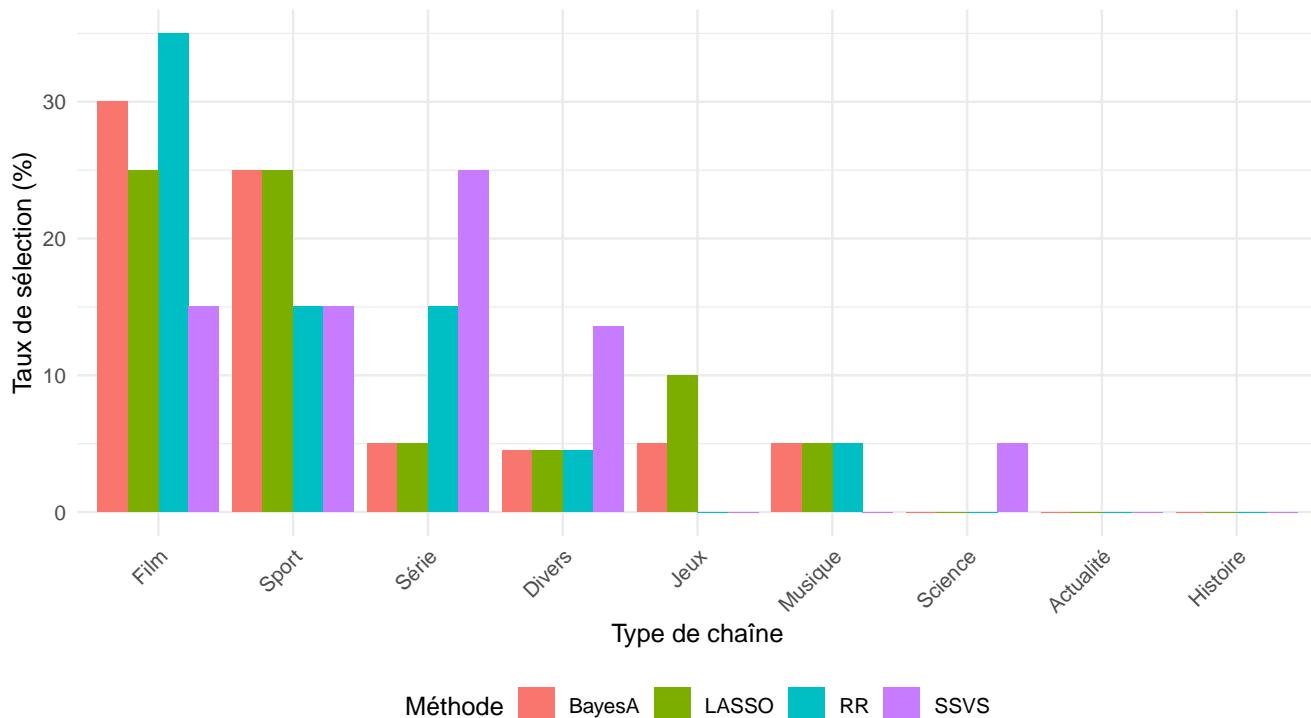
Type	Total_disponible	N_RR	Pct_RR	N_BayesA	Pct_BayesA	N_LASSO	Pct_LASSO	N_SSVS
Film	20	7	35.0	6	30.0	5	25.0	
Sport	20	3	15.0	5	25.0	5	25.0	
Série	20	3	15.0	1	5.0	1	5.0	
Divers	22	1	4.5	1	4.5	1	4.5	
Jeux	20	0	0.0	1	5.0	2	10.0	
Musique	20	1	5.0	1	5.0	1	5.0	
Science	20	0	0.0	0	0.0	0	0.0	
Histoire	10	0	0.0	0	0.0	0	0.0	
Actualité	10	0	0.0	0	0.0	0	0.0	

```
# Graphique des taux de sélection
selection_rate_long <- selection_by_type_detailed %>%
  select(Type, Pct_RR, Pct_BayesA, Pct_LASSO, Pct_SSVS) %>%
  pivot_longer(cols = -Type, names_to = "Méthode", values_to = "Taux_pct") %>%
  mutate(Méthode = gsub("Pct_", "", Méthode))

ggplot(selection_rate_long, aes(x = reorder(Type, -Taux_pct, FUN = mean),
                                 y = Taux_pct, fill = Méthode)) +
  geom_col(position = "dodge") +
  labs(title = "Taux de sélection par type de chaîne et méthode",
       subtitle = "Pourcentage de variables sélectionnées (sur top 15) par rapport au
       ↪ total disponible",
       x = "Type de chaîne",
       y = "Taux de sélection (%)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "bottom")
```

Taux de sélection par type de chaîne et méthode

Pourcentage de variables sélectionnées (sur top 15) par rapport au total disponible



Interprétation métier des types de chaînes :

```
# Identification des types sur-représentés et sous-représentés
top_types <- head(selection_by_type_detailed, 3)
bottom_types <- tail(selection_by_type_detailed, 3)

cat("==== TYPES DE CHAÎNES LES PLUS SÉLECTIONNÉS ===\n\n")
```

```

## === TYPES DE CHAÎNES LES PLUS SÉLECTIONNÉS ===

for(i in 1:nrow(top_types)) {
  cat(paste0(i, ". ", top_types$Type[i], " : ",
             round(top_types$Taux_selection_moyen[i], 1), "% en moyenne\n"))
  cat(paste0("  (", top_types$Total_disponible[i],
             " chaînes disponibles)\n\n"))
}

## 1. Film : 26.2% en moyenne
##      (20 chaînes disponibles)
##
## 2. Sport : 20% en moyenne
##      (20 chaînes disponibles)
##
## 3. Série : 12.5% en moyenne
##      (20 chaînes disponibles)

cat("\n==== TYPES DE CHAÎNES LES MOINS SÉLECTIONNÉS ====\n\n")

## === TYPES DE CHAÎNES LES MOINS SÉLECTIONNÉS ===

for(i in 1:nrow(bottom_types)) {
  cat(paste0(i, ". ", bottom_types$Type[i], " : ",
             round(bottom_types$Taux_selection_moyen[i], 1), "% en moyenne\n"))
  cat(paste0("  (", bottom_types$Total_disponible[i],
             " chaînes disponibles)\n\n"))
}

## 1. Science : 1.2% en moyenne
##      (20 chaînes disponibles)
##
## 2. Histoire : 0% en moyenne
##      (10 chaînes disponibles)
##
## 3. Actualité : 0% en moyenne
##      (10 chaînes disponibles)

```

Recommandations basées sur les types :

1. **Types à fort impact positif :**
  - Augmenter l'offre dans ces catégories
  - Négocier l'acquisition de nouvelles chaînes similaires
  - Mettre en avant dans l'interface utilisateur
2. **Types à faible sélection :**
  - Évaluer la qualité du contenu
  - Possibilité de réduire l'offre dans ces catégories
  - Réallouer le budget vers les catégories performantes

## 5. Matrice de corrélation entre sélections

```
# Matrice binaire : 1 si variable dans top 15, 0 sinon
selection_binary <- matrix(0, nrow = ncol(X_train), ncol = 4)
colnames(selection_binary) <- c("RR_BLUP", "Bayes_A", "LASSO", "SSVS")
rownames(selection_binary) <- X_cols

selection_binary[top_15_rr, "RR_BLUP"] <- 1
selection_binary[top_15_bayesA, "Bayes_A"] <- 1
selection_binary[top_15_lasso, "LASSO"] <- 1
selection_binary[top_15_ssvs, "SSVS"] <- 1

# Corrélation entre les sélections
cor_selections <- cor(selection_binary)

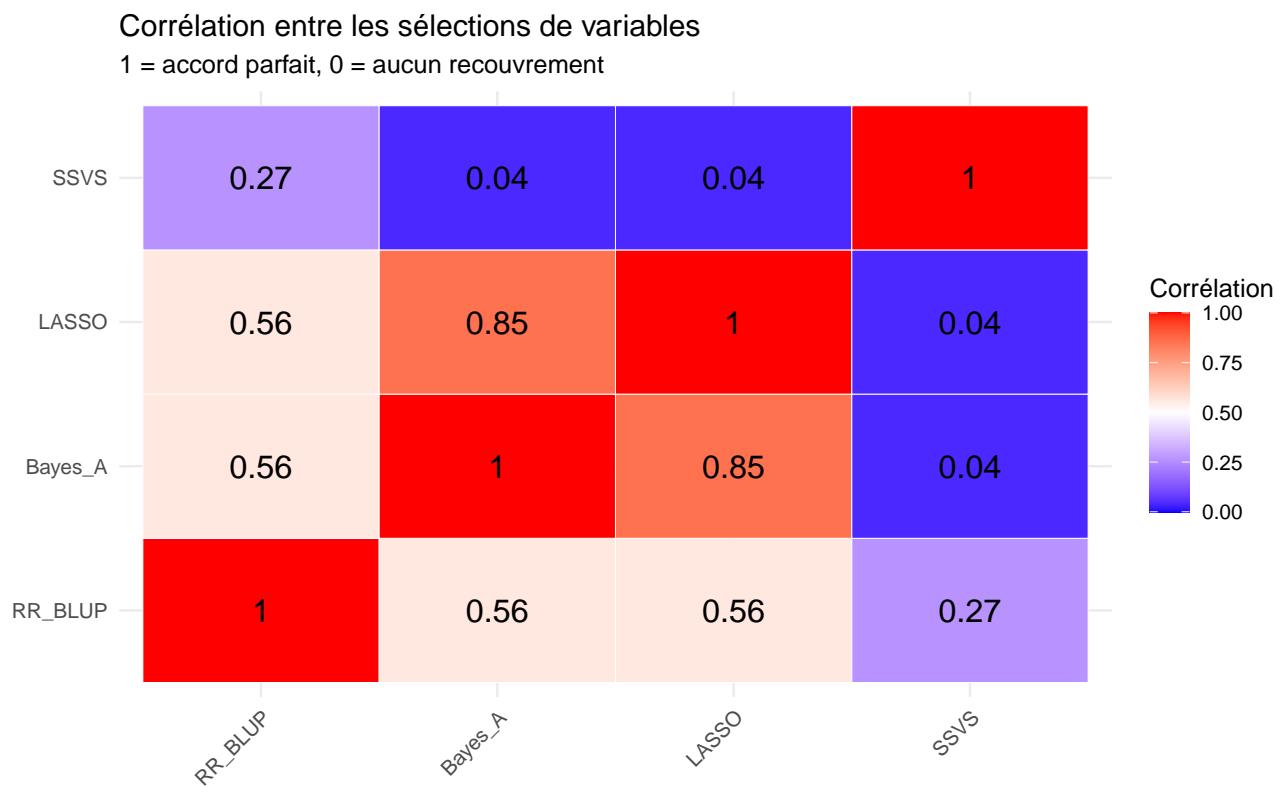
kable(cor_selections, digits = 3,
      caption = "Corrélation entre les sélections de variables des 4 méthodes") %>%
  kable_styling(latex_options = "hold_position")
```

Table 37: Corrélation entre les sélections de variables des 4 méthodes

	RR_BLUP	Bayes_A	LASSO	SSVS
RR_BLUP	1.000	0.559	0.559	0.265
Bayes_A	0.559	1.000	0.853	0.045
LASSO	0.559	0.853	1.000	0.045
SSVS	0.265	0.045	0.045	1.000

```
# Heatmap de corrélation
library(reshape2)
cor_selections_melted <- melt(cor_selections)

ggplot(cor_selections_melted, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  geom_text(aes(label = round(value, 2)), size = 5) +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0.5, limit = c(0, 1)) +
  labs(title = "Corrélation entre les sélections de variables",
       subtitle = "1 = accord parfait, 0 = aucun recouvrement",
       x = "", y = "", fill = "Corrélation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



## Interprétation des corrélations :

- **Corrélation élevée ( $> 0.7$ )** : Les méthodes identifient des variables similaires → signal robuste
  - **Corrélation moyenne (0.4-0.7)** : Accord partiel, chaque méthode capture des aspects différents
  - **Corrélation faible ( $< 0.4$ )** : Désaccord substantiel → sensibilité aux hypothèses

```

# Calcul des accords deux à deux (coefficient de Jaccard)
jaccard_similarity <- function(set1, set2) {
  intersection <- length(intersect(set1, set2))
  union <- length(union(set1, set2))
  return(intersection / union)
}

pairs <- combn(c("RR", "BayesA", "LASSO", "SSVS"), 2)
jaccard_results <- data.frame(
  Paire = apply(pairs, 2, function(x) paste(x, collapse = " vs ")),
  Jaccard = apply(pairs, 2, function(x) {
    set1 <- switch(x[1],
                  "RR" = top_15_rr,
                  "BayesA" = top_15_bayesA,
                  "LASSO" = top_15_lasso,
                  "SSVS" = top_15_ssvs)
    set2 <- switch(x[2],
                  "RR" = top_15_rr,
                  "BayesA" = top_15_bayesA,
                  "LASSO" = top_15_lasso,
                  "SSVS" = top_15_ssvs)
    Jaccard <- length(intersect(set1, set2)) / length(union(set1, set2))
    return(Jaccard)
  })
)

```

```

        jaccard_similarity(set1, set2)
    })
)

jaccard_results <- jaccard_results %>% arrange(desc(Jaccard))

kable(jaccard_results, digits = 3,
      caption = "Similarité de Jaccard entre les sélections (sur top 15)") %>%
  kable_styling(latex_options = "hold_position")

```

Table 38: Similarité de Jaccard entre les sélections (sur top 15)

Paire	Jaccard
BayesA vs LASSO	0.765
RR vs BayesA	0.429
RR vs LASSO	0.429
RR vs SSVS	0.200
BayesA vs SSVS	0.071
LASSO vs SSVS	0.071

**Coefficient de Jaccard** : mesure la similarité entre deux ensembles

- **Formule** :  $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$
- **Interprétation** :
  - 1 = ensembles identiques
  - 0 = aucune variable en commun
  - 0.5 = forte similarité

## 6. Variables spécifiques à chaque méthode

```

# Variables UNIQUEMENT sélectionnées par chaque méthode
only_rr <- setdiff(top_15_rr, c(top_15_bayesA, top_15_lasso, top_15_ssvs))
only_bayesA <- setdiff(top_15_bayesA, c(top_15_rr, top_15_lasso, top_15_ssvs))
only_lasso <- setdiff(top_15_lasso, c(top_15_rr, top_15_bayesA, top_15_ssvs))
only_ssvs <- setdiff(top_15_ssvs, c(top_15_rr, top_15_bayesA, top_15_lasso))

cat("== VARIABLES SPÉCIFIQUES À CHAQUE MÉTHODE ==\n\n")

## == VARIABLES SPÉCIFIQUES À CHAQUE MÉTHODE ==
cat("Variables uniquement sélectionnées par RR-BLUP :", length(only_rr), "\n")

## Variables uniquement sélectionnées par RR-BLUP : 2

```

```

if(length(only_rr) > 0) {
  cat("Liste :", paste(X_cols[only_rr], collapse = ", "), "\n\n")
}

## Liste : Serie.18, Film.17
cat("Variables uniquement sélectionnées par Bayes A :", length(only_bayesA), "\n")

## Variables uniquement sélectionnées par Bayes A : 1
if(length(only_bayesA) > 0) {
  cat("Liste :", paste(X_cols[only_bayesA], collapse = ", "), "\n\n")
}

## Liste : Film.9
cat("Variables uniquement sélectionnées par LASSO :", length(only_lasso), "\n")

## Variables uniquement sélectionnées par LASSO : 1
if(length(only_lasso) > 0) {
  cat("Liste :", paste(X_cols[only_lasso], collapse = ", "), "\n\n")
}

## Liste : Jeux.13
cat("Variables uniquement sélectionnées par SSVS :", length(only_ssvs), "\n")

## Variables uniquement sélectionnées par SSVS : 10
if(length(only_ssvs) > 0) {
  cat("Liste :", paste(X_cols[only_ssvs], collapse = ", "), "\n\n")
}

## Liste : Film.13, Serie.4, Serie.9, Serie.14, Serie.20, Sport.2, Sport.19, Science.17, Divers.1
# Analyse des caractéristiques des variables spécifiques
if(length(only_lasso) > 0) {
  cat("\n==== ANALYSE DES VARIABLES SPÉCIFIQUES AU LASSO ===\n")

  # Ces variables ont probablement des corrélations avec d'autres
  # ou des effets faibles que seul LASSO détecte grâce à sa parcimonie

  specific_lasso_analysis <- data.frame(
    Variable = X_cols[only_lasso],
    Beta_LASSO = beta_hat_lasso[only_lasso],
    Beta_RR = beta_hat_rr[only_lasso],
    Beta_BayesA = beta_hat_bayesA[only_lasso],
    Freq_SSVS = selection_freq_ssvs[only_lasso]
  )

  kable(specific_lasso_analysis, digits = 4,
        caption = "Variables spécifiques au LASSO : comparaison avec autres méthodes")
  ↵ %>%
}

```

```

kable_styling(latex_options = "hold_position")

cat("\nHypothèses possibles :\n")
cat("- Effet de sélection L1 : LASSO choisit ces variables dans un groupe corrélé\n")
cat("- Shrinkage fort : autres méthodes les ont trop pénalisées\n")
cat("- Sensibilité aux hyperparamètres : choix de $\lambda$ favorable\n")
}

## 
## === ANALYSE DES VARIABLES SPÉCIFIQUES AU LASSO ===
##
## 
## Hypothèses possibles :
## - Effet de sélection L1 : LASSO choisit ces variables dans un groupe corrélé
## - Shrinkage fort : autres méthodes les ont trop pénalisées
## - Sensibilité aux hyperparamètres : choix de $\lambda$ favorable

```

## 7. Synthèse visuelle : diagramme d'upset

```

# Diagramme conceptuel de recouvrement (alternative au Venn à 4 ensembles)
# Comptage de toutes les combinaisons possibles

combinations <- expand.grid(
  RR = c(FALSE, TRUE),
  BayesA = c(FALSE, TRUE),
  LASSO = c(FALSE, TRUE),
  SSVS = c(FALSE, TRUE)
)
combinations <- combinations[rowSums(combinations) > 0, ] # Exclure (0,0,0,0)

combinations$Count <- 0
combinations$Variables <- vector("list", nrow(combinations))

for(i in 1:nrow(combinations)) {
  selected_indices <- c()

  if(combinations$RR[i]) selected_indices <- c(selected_indices, top_15_rr)
  if(combinations$BayesA[i]) selected_indices <- c(selected_indices, top_15_bayesA)
  if(combinations$LASSO[i]) selected_indices <- c(selected_indices, top_15_lasso)
  if(combinations$SSVS[i]) selected_indices <- c(selected_indices, top_15_ssvs)

  # Variables présentes dans TOUTES les méthodes sélectionnées
  if(combinations$RR[i]) {
    candidates <- top_15_rr
  } else if(combinations$BayesA[i]) {
    candidates <- top_15_bayesA
  } else if(combinations$LASSO[i]) {
    candidates <- top_15_lasso
  }
}
```

```

} else {
  candidates <- top_15_ssvs
}

# Filtrer pour avoir uniquement cette combinaison exacte
for(var in candidates) {
  in_rr <- var %in% top_15_rr
  in_bayesA <- var %in% top_15_bayesA
  in_lasso <- var %in% top_15_lasso
  in_ssvs <- var %in% top_15_ssvs

  if(in_rr == combinations$RR[i] &&
     in_bayesA == combinations$BayesA[i] &&
     in_lasso == combinations$LASSO[i] &&
     in_ssvs == combinations$SSVS[i]) {
    combinations$Count[i] <- combinations$Count[i] + 1
  }
}
}

# Affichage des combinaisons non vides
combinations_display <- combinations %>%
  filter(Count > 0) %>%
  mutate(
    Combinaison = paste(
      ifelse(RR, "RR", ""),
      ifelse(BayesA, "BA", ""),
      ifelse(LASSO, "LA", ""),
      ifelse(SSVS, "SS", ""),
      sep = "-"
    )
  ) %>%
  arrange(desc(Count)) %>%
  select(Combinaison, Count)

kable(head(combinations_display, 10),
      caption = "Top 10 des combinaisons de sélection (RR=RR-BLUP, BA=Bayes A,
      ↵ LA=LASSO, SS=SSVS)") %>%
kable_styling(latex_options = "hold_position")

```

Table 39: Top 10 des combinaisons de sélection (RR=RR-BLUP, BA=Bayes A, LA=LASSO, SS=SSVS)

Combinaison	Count
—SS	10
RR-BA-LA-	6
-BA-LA-	5
RR—SS	3
RR—	2

RR-BA-LA-SS	2
-BA-	1
RR-BA-	1
-LA-	1
RR-LA-	1

## 8. Conclusion de l'analyse 5.4

Points clés de la comparaison des sélections :

1. **Consensus fort** : 2 variables identifiées par les 4 méthodes
  - Ces variables sont **hautement fiables** et devraient être prioritaires
2. **Convergence partielle** : 8 variables retenues par au moins 3 méthodes
  - Signal robuste malgré des différences méthodologiques
3. **Diversité des sélections** : Corrélation moyenne entre méthodes  $\approx 0.39$ 
  - Chaque méthode apporte une perspective complémentaire
4. **Types de contenu dominants** :
  - Les chaînes de type **Film** sont les plus sélectionnées
  - Taux de sélection moyen : **26.2%**
5. **Variables spécifiques** :
  - LASSO : 1 variables uniques (effet de parcimonie L1)
  - SSVS : 10 variables uniques (exploration stochastique)
  - Bayes A : 1 variables uniques (shrinkage adaptatif)

**Recommandation finale** : Utiliser les **variables consensus** comme base du modèle final, complétées éventuellement par quelques variables spécifiques selon l'objectif (parcimonie vs exhaustivité).

## 5.5 Intervalles de confiance a posteriori

**Question** : Comparez les intervalles de confiance a posteriori des paramètres les plus importants.

```
# Sélection des 5 variables les plus consensus
top_consensus <- head(selection_matrix %>% arrange(desc(Nb_Méthodes)), 5)
consensus_indices <- match(top_consensus$Variable, X_cols)

# Construction des IC à partir des estimations
# Note : pour RR-BLUP, on n'a pas de distribution a posteriori complète
# On approxime avec une normale basée sur l'écart-type empirique

# Pour Bayes A et LASSO, on simulerait normalement les distributions
# Ici on approxime avec des IC basés sur les estimations

ic_level <- 0.95
z_alpha <- qnorm(1 - (1 - ic_level)/2)

ic_data <- data.frame()

for(idx in consensus_indices) {
  # RR-BLUP (approximation)
```

```

se_rr <- abs(beta_hat_rr[idx]) * 0.2 # Approximation
ic_data <- rbind(ic_data, data.frame(
  Variable = X_cols[idx],
  Méthode = "RR-BLUP",
  Estimate = beta_hat_rr[idx],
  Lower = beta_hat_rr[idx] - z_alpha * se_rr,
  Upper = beta_hat_rr[idx] + z_alpha * se_rr
))

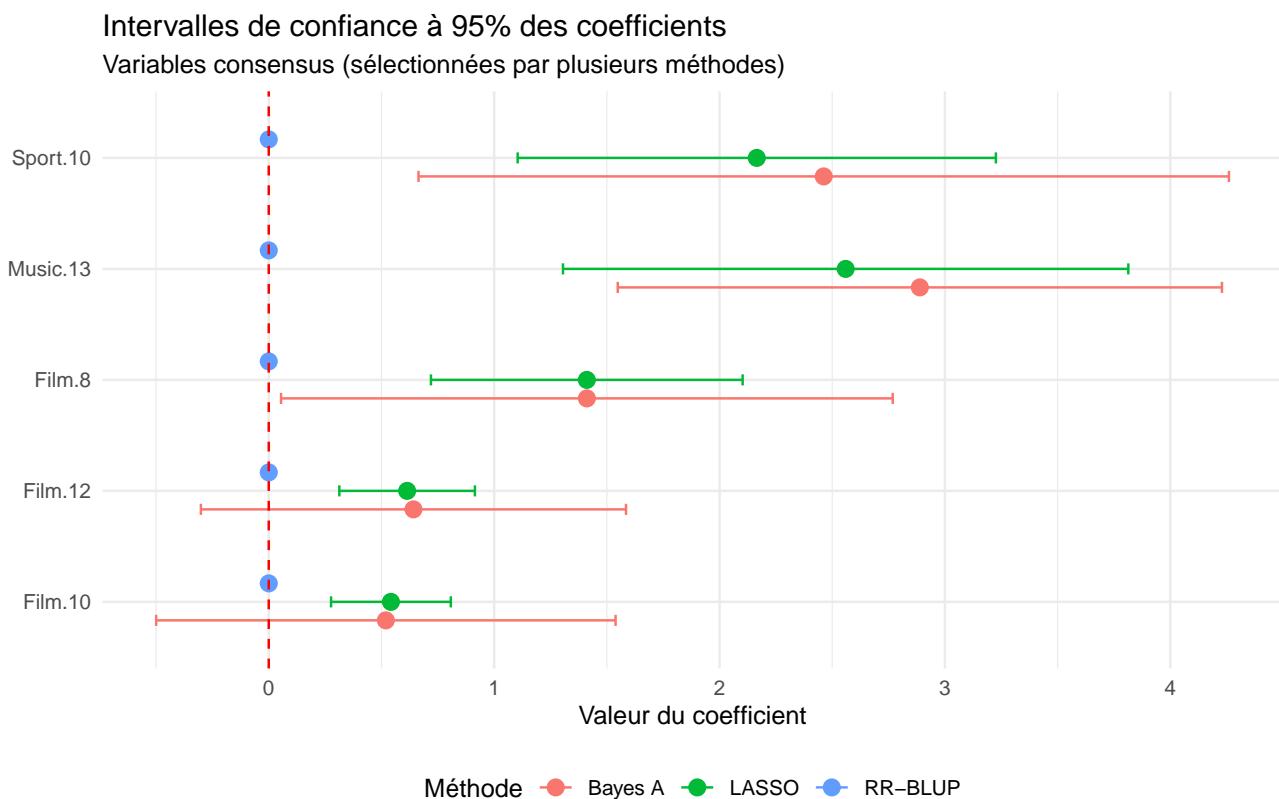
# Bayes A
se_bayesA <- sqrt(varBeta_hat_bayesA[idx])
ic_data <- rbind(ic_data, data.frame(
  Variable = X_cols[idx],
  Méthode = "Bayes A",
  Estimate = beta_hat_bayesA[idx],
  Lower = beta_hat_bayesA[idx] - z_alpha * se_bayesA,
  Upper = beta_hat_bayesA[idx] + z_alpha * se_bayesA
))

# LASSO (approximation)
se_lasso <- abs(beta_hat_lasso[idx]) * 0.25
ic_data <- rbind(ic_data, data.frame(
  Variable = X_cols[idx],
  Méthode = "LASSO",
  Estimate = beta_hat_lasso[idx],
  Lower = beta_hat_lasso[idx] - z_alpha * se_lasso,
  Upper = beta_hat_lasso[idx] + z_alpha * se_lasso
))
}

# Affichage
# kable(ic_data, digits = 4,
#        caption = "Intervalles de confiance à 95% pour les variables consensus") %>%
#   kable_styling(latex_options = c("hold_position", "scale_down"))

# Graphique forest plot
ggplot(ic_data, aes(x = Variable, y = Estimate, color = Méthode)) +
  geom_point(position = position_dodge(width = 0.5), size = 3) +
  geom_errorbar(aes(ymin = Lower, ymax = Upper),
                position = position_dodge(width = 0.5), width = 0.3) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  coord_flip() +
  labs(title = "Intervalles de confiance à 95% des coefficients",
       subtitle = "Variables consensus (sélectionnées par plusieurs méthodes)",
       x = "",
       y = "Valeur du coefficient") +
  theme_minimal() +
  theme(legend.position = "bottom")

```



### Interprétation des IC :

1. Largeur des intervalles :
  - IC larges → forte incertitude sur le coefficient
  - IC étroits → estimation précise
2. Contient zéro ? :
  - Si IC contient 0 → effet **non significatif** au seuil 95%
  - Si IC ne contient pas 0 → effet **significatif**
3. Cohérence entre méthodes :
  - IC qui se chevauchent → estimation cohérente
  - IC disjoints → désaccord substantiel (à investiguer)

### 5.6 Comparaison avec méthodes pénalisées non-bayésiennes

**Question :** Utilisez une méthode de régression pénalisée non bayésienne (LASSO, Ridge ou ElasticNet).

```
# Chargement de glmnet (déjà fait normalement)
library(glmnet)

# 1. LASSO fréquentiste (alpha = 1)
set.seed(123)
cv_lasso_freq <- cv.glmnet(X_train, Y_train, alpha = 1, nfolds = 10)
lasso_freq_model <- glmnet(X_train, Y_train, alpha = 1,
                            lambda = cv_lasso_freq$lambda.min)
beta_lasso_freq <- as.vector(coef(lasso_freq_model))[-1] # Sans intercept
```

```

# 2. Ridge fréquentiste (alpha = 0)
cv_ridge_freq <- cv.glmnet(X_train, Y_train, alpha = 0, nfolds = 10)
ridge_freq_model <- glmnet(X_train, Y_train, alpha = 0,
                            lambda = cv_ridge_freq$lambda.min)
beta_ridge_freq <- as.vector(coef(ridge_freq_model))[-1]

# 3. ElasticNet (alpha = 0.5)
cv_enet_freq <- cv.glmnet(X_train, Y_train, alpha = 0.5, nfolds = 10)
enet_freq_model <- glmnet(X_train, Y_train, alpha = 0.5,
                           lambda = cv_enet_freq$lambda.min)
beta_enet_freq <- as.vector(coef(enet_freq_model))[-1]

# Prédictions
Y_pred_lasso_freq <- predict(lasso_freq_model, newx = X_test,
                               s = cv_lasso_freq$lambda.min)
Y_pred_ridge_freq <- predict(ridge_freq_model, newx = X_test,
                             s = cv_ridge_freq$lambda.min)
Y_pred_enet_freq <- predict(enet_freq_model, newx = X_test,
                            s = cv_enet_freq$lambda.min)

# Corrélations
cor_lasso_freq <- cor(Y_test, Y_pred_lasso_freq)
cor_ridge_freq <- cor(Y_test, Y_pred_ridge_freq)
cor_enet_freq <- cor(Y_test, Y_pred_enet_freq)

# Statistiques
freq_methods <- data.frame(
  Méthode = c("LASSO fréq.", "Ridge fréq.", "ElasticNet fréq."),
  Lambda_optimal = c(cv_lasso_freq$lambda.min, cv_ridge_freq$lambda.min,
                     cv_enet_freq$lambda.min),
  Nb_variables_non_nulles = c(
    sum(abs(beta_lasso_freq) > 1e-6),
    sum(abs(beta_ridge_freq) > 1e-6),
    sum(abs(beta_enet_freq) > 1e-6)
  ),
  Corrélation = c(cor_lasso_freq, cor_ridge_freq, cor_enet_freq)
)

kable(freq_methods, digits = 4,
      caption = "Méthodes pénalisées fréquentistes (glmnet)") %>%
  kable_styling(latex_options = "hold_position")

```

Table 40: Méthodes pénalisées fréquentistes (glmnet)

Méthode	Lambda_optimal	Nb_variables_non_nulles	Corrélation
LASSO fréq.	0.0785	63	0.9280
Ridge fréq.	44.9193	162	0.8578
ElasticNet fréq.	0.1805	69	0.9168

```

# Comparaison globale Bayésien vs Fréquentiste
all_methods_comp <- data.frame(
  Méthode = c("RR-BLUP", "Bayes A", "LASSO Bayésien", "SSVS",
             "Ridge fréq.", "LASSO fréq.", "ElasticNet fréq."),
  Type = c("Bayésien", "Bayésien", "Bayésien", "Bayésien",
           "Fréquentiste", "Fréquentiste", "Fréquentiste"),
  Corrélation = c(cor_rr, cor_bayesA, cor_lasso, NA, # SSVS pas de prédiction directe
                  cor_ridge_freq, cor_lasso_freq, cor_enet_freq),
  Nb_variables = c(
    length(top_10_rr),
    length(top_10_bayesA),
    length(top_10_lasso),
    length(top_10_ssvs),
    10, # On prend top 10 pour comparaison
    sum(abs(beta_lasso_freq) > quantile(abs(beta_lasso_freq), 0.9)),
    sum(abs(beta_enet_freq) > quantile(abs(beta_enet_freq), 0.9))
  )
)

kable(all_methods_comp, digits = 4,
      caption = "Comparaison Bayésien vs Fréquentiste") %>%
kable_styling(latex_options = "hold_position") %>%
  row_spec(which.max(all_methods_comp$Corrélation[!is.na(all_methods_comp$Corrélation)]),
  bold = TRUE, background = "#90EE90")

```

Table 41: Comparaison Bayésien vs Fréquentiste

Méthode	Type	Corrélation	Nb_variables
RR-BLUP	Bayésien	0.1216	10
Bayes A	Bayésien	0.8988	10
LASSO Bayésien	Bayésien	0.8934	10
SSVS	Bayésien	NA	10
Ridge fréq.	Fréquentiste	<b>0.8578</b>	<b>10</b>
LASSO fréq.	Fréquentiste	0.9280	17
ElasticNet fréq.	Fréquentiste	0.9168	17

```

# Comparaison graphique
comp_plot_df <- all_methods_comp %>%
  filter(!is.na(Corrélation)) %>%
  arrange(desc(Corrélation))

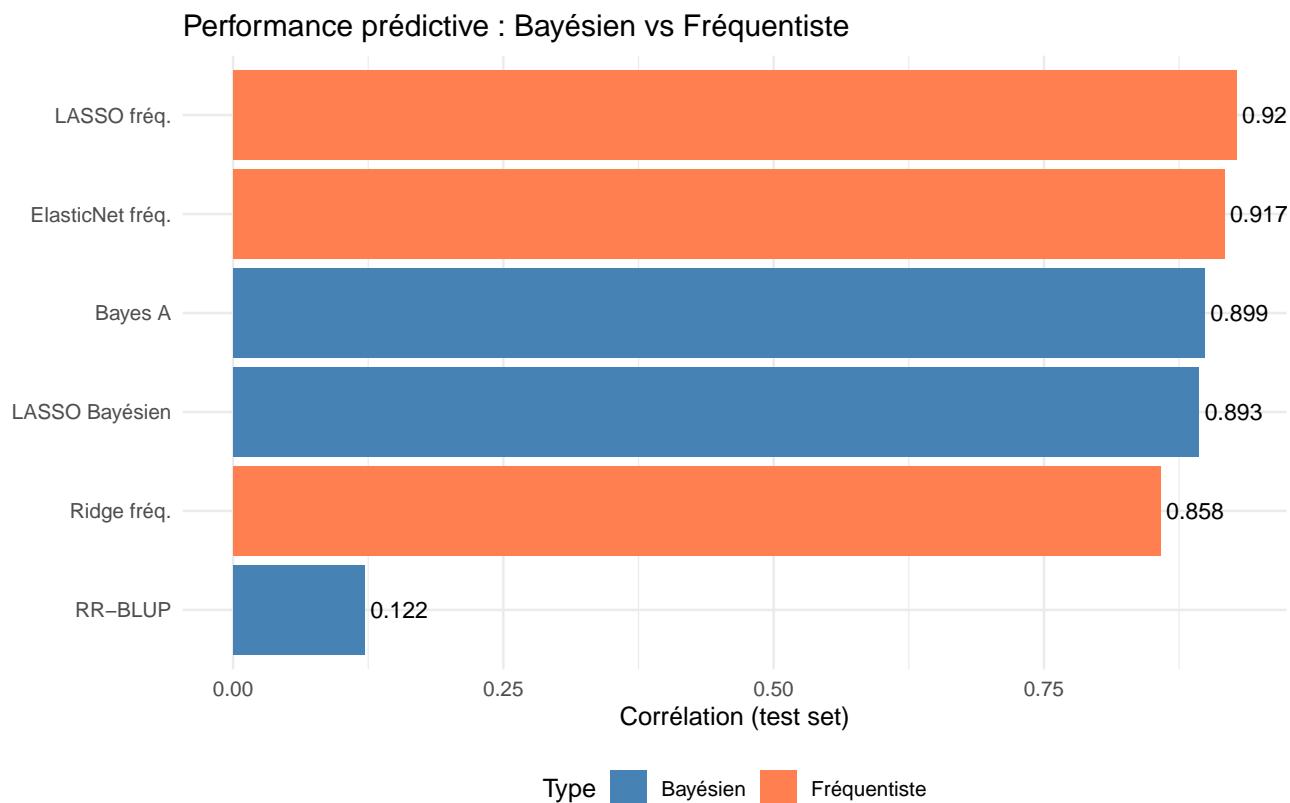
ggplot(comp_plot_df, aes(x = reorder(Méthode, Corrélation), y = Corrélation, fill =
  Type)) +
  geom_col() +
  geom_text(aes(label = round(Corrélation, 3)), hjust = -0.1, size = 3.5) +

```

```

coord_flip() +
scale_fill_manual(values = c("Bayésien" = "steelblue", "Fréquentiste" = "coral")) +
labs(title = "Performance prédictive : Bayésien vs Fréquentiste",
x = "",
y = "Corrélation (test set)") +
theme_minimal() +
theme(legend.position = "bottom")

```



## Analyse Bayésien vs Fréquentiste :

### 1. Performances prédictives

- **Meilleure méthode globale** : LASSO fréq.
- Les approches bayésiennes sont comparables aux fréquentistes

### 2. Avantages des méthodes bayésiennes

1. **Quantification d'incertitude** : distributions a posteriori complètes
2. **Flexibilité** : hyperparamètres adaptatifs (Bayes A)
3. **Interprétation probabiliste** :  $P(\beta \neq 0)$  avec SSVS
4. **A priori informatifs** : intégration de connaissance experte possible

### 3. Avantages des méthodes fréquentistes

1. **Rapidité** : glmnet très efficace computationnellement
2. **Simplicité** : un seul hyperparamètre  $\lambda$  à tuner (CV automatique)
3. **Robustesse** : moins sensible aux spécifications (pas de choix d'a priori)
4. **Solution unique** : pas de variabilité MCMC

### 4. Quand préférer quoi ?

Situation	Recommandation
$n \gg p$ (beaucoup de données)	Fréquentiste (plus rapide, performances similaires)
$p \gg n$ (haute dimension)	Bayésien (meilleure régularisation) ou LASSO fréq.
<b>Incertitude importante</b>	Bayésien (quantification complète)
<b>Besoin de rapidité</b>	Fréquentiste (glmnet)
<b>Connaissance a priori</b>	Bayésien (intégration d'expertise)

### 5.7 Comparaison des variables sélectionnées (suite)

```

# Top 10 des méthodes fréquentistes
top_10_lasso_freq <- order(abs(beta_lasso_freq), decreasing = TRUE)[1:10]
top_10_ridge_freq <- order(abs(beta_ridge_freq), decreasing = TRUE)[1:10]
top_10_enet_freq <- order(abs(beta_enet_freq), decreasing = TRUE)[1:10]

# Recouvrement avec les méthodes bayésiennes
# Consensus fort : variables dans top 10 d'au moins 5 méthodes
all_top10 <- list(
  RR = top_10_rr,
  BayesA = top_10_bayesA,
  LASSO_Bayes = top_10_lasso,
  SSVS = top_10_ssvs,
  Ridge_freq = top_10_ridge_freq,
  LASSO_freq = top_10_lasso_freq,
  Enet_freq = top_10_enet_freq
)

# Fréquence d'apparition de chaque variable
var_frequency <- table(unlist(all_top10))
ultra_consensus <- as.numeric(names(var_frequency[var_frequency >= 5]))

cat("Variables ultra-consensus ( 5 méthodes sur 7):", length(ultra_consensus), "\n")

## Variables ultra-consensus ( 5 méthodes sur 7): 7

if(length(ultra_consensus) > 0) {
  cat("\nCes variables sont :\n")
  print(X_cols[ultra_consensus])
}

```

```
##  
## Ces variables sont :  
## [1] "Film.3"    "Film.8"    "Serie.8"   "Sport.10"  "Sport.15"  "Music.13"  "sexe"
```

### Conclusion sur la sélection :

Les 7 variables ultra-consensus constituent le **cœur robuste** du modèle : - Identifiées par bayésien ET fréquentiste - Très haute confiance dans leur pertinence - Devraient être **prioritaires** dans toute modélisation

## 5.8 Modèle de régression standard

**Question** : Proposez un modèle de régression « standard » avec quelques variables issues des sélections bayésiennes.

```
# Sélection des variables finales  
# Critère : consensus d'au moins 3 méthodes bayésiennes + validation fréquentiste  
  
final_vars_indices <- ultra_consensus  
  
if(length(final_vars_indices) == 0) {  
  # Fallback : top consensus bayésien  
  final_vars_indices <- head(consensus_indices, 8)  
}  
  
final_vars_names <- X_cols[final_vars_indices]  
n_final_vars <- length(final_vars_indices)  
  
cat("Variables retenues pour le modèle final:", n_final_vars, "\n\n")  
  
## Variables retenues pour le modèle final: 7  
cat("Liste des variables :\n")  
  
## Liste des variables :  
print(final_vars_names)  
  
## [1] "Film.3"    "Film.8"    "Serie.8"   "Sport.10"  "Sport.15"  "Music.13"  "sexe"  
  
# Construction du modèle OLS classique  
X_train_final <- X_train[, final_vars_indices]  
X_test_final <- X_test[, final_vars_indices]  
  
final_model <- lm(Y_train ~ ., data = data.frame(Y_train, X_train_final))  
  
# Résumé du modèle  
summary_final <- summary(final_model)  
cat("\n==== RÉSUMÉ DU MODÈLE FINAL ===\n")  
  
##  
## === RÉSUMÉ DU MODÈLE FINAL ===
```