Rapport TP2 - Application des Régressions Régularisées à la Consommation d'Électricité

Alexis Schneider - Kelvin Wong

19-10-2025

Contents

1 Electricity Data Set : Approche par Régularisation

1.1 Introduction et rappel du contexte (TP1) :

Dans notre précédent travail (TP1), nous avons construit un modèle de régression linéaire pour expliquer la consommation d'électricité (Total). Ce travail avait mis en lumière plusieurs défis : la nécessité d'une ingénierie des variables pour les données cycliques, la présence de forte multicolinéarité entre les prédicteurs de température et de radiation, et l'existence d'une non-linéarité dans la relation avec la température. Notre meilleure solution, le Modèle 2, reposait sur une sélection manuelle des variables et l'ajout d'un terme quadratique.

1.1.1 Validation et Comparaison aux Méthodes Classiques

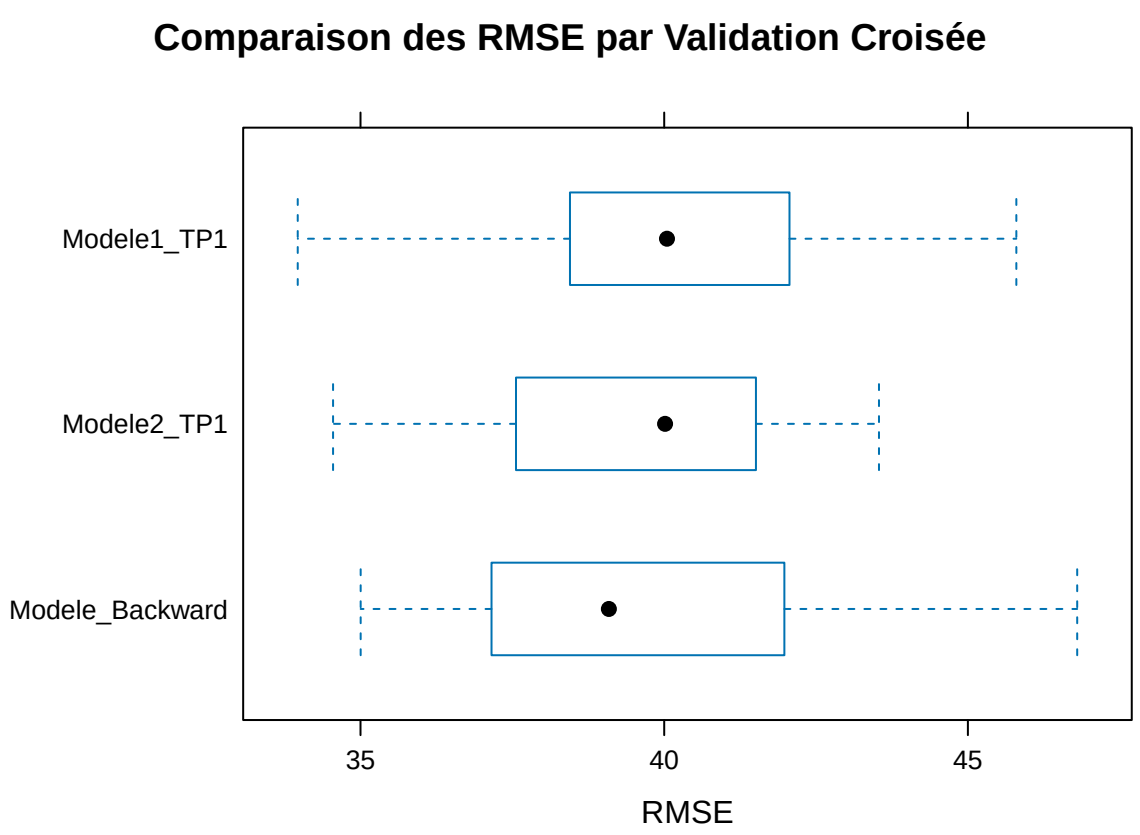
Avant d'explorer les méthodes de régularisation, nous allons établir une base de comparaison solide. Pour cela, nous allons : Valider de manière robuste les performances de nos modèles du TP1 via la validation croisée. Comparer ces performances à celles d'un modèle obtenu par une méthode de sélection algorithmique classique : la sélection descendante (backward). Cette approche, vue dans l'exercice 2 du TP, construit un modèle en partant de l'ensemble de toutes les variables disponibles, puis en retirant itérativement le prédicteur le moins pertinent (généralement sur la base du critère AIC) jusqu'à obtenir un sous-modèle optimal. Cela nous permettra de quantifier la performance de notre sélection manuelle face à cet algorithme standard, qui sert de référence classique en modélisation.

```r
# Chargement et préparation des données
data_Mexico <- read.csv("Mexico_data.csv")
data_Mexico$sin_TOY <- sin(2 * pi * data_Mexico$TOY / 365.25)
data_Mexico$cos_TOY <- cos(2 * pi * data_Mexico$TOY / 365.25)
data_Mexico$DOW <- as.factor(data_Mexico$DOW)

# Définition de la méthode de validation croisée
train_control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

# Étape 1 : Évaluation des modèles du TP1
formula1 <- Total ~ T2M + RH + SSRD + Covid + Holidays + DOW +
                    sin_TOY + cos_TOY
formula2 <- Total ~ T2M + I(T2M^2) + RH + SSRD + Covid + Holidays + DOW + sin_TOY + cos_TOY
cv_model1 <- train(formula1, data = data_Mexico, method = "lm", trControl = train_control)
cv_model2 <- train(formula2, data = data_Mexico, method = "lm", trControl = train_control)

# Étape 2 : Création et évaluation du modèle Backward
full_formula <- Total ~ T2M + I(T2M^2) + T2Mmax + T2Mmin + RH + SSRD +
                        STRD + Covid + Holidays + DOW + sin_TOY + cos_TOY
full_model_for_step <- lm(full_formula, data = data_Mexico)
backward_model <- step(full_model_for_step, direction = "backward", trace = 0)
cv_backward_model <- train(formula(backward_model), data = data_Mexico, method = "lm",
                           trControl = train_control)

# Étape 3 : Comparaison des performances prédictives (RMSE)
all_results <- resamples(list(
  Modele1_TP1 = cv_model1,
  Modele2_TP1 = cv_model2,
  Modele_Backward = cv_backward_model
))
summary(all_results)
```

```
##
## Call:
## summary.resamples(object = all_results)
##
## Models: Modele1_TP1, Modele2_TP1, Modele_Backward
## Number of resamples: 30
##
## MAE
##                      Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## Modele1_TP1     26.45703 29.30463 30.19607 30.40970 31.39117 35.15140    0
## Modele2_TP1     25.00531 28.90713 30.31532 30.34239 32.10394 34.09309    0
## Modele_Backward 26.90161 28.71814 30.10850 30.28851 31.11740 35.38499    0
##
## RMSE
##                      Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## Modele1_TP1     33.96649 38.51304 40.04568 40.17205 42.04312 45.79731    0
## Modele2_TP1     34.54825 37.60398 40.01433 39.60518 41.49796 43.53585    0
## Modele_Backward 35.00279 37.27358 39.08931 39.57959 41.82194 46.79938    0
##
## Rsquared
##                       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
```

```
## Modele1_TP1     0.7600215 0.8129245 0.8247337 0.8225806 0.8364439 0.8668790
## Modele2_TP1     0.7829501 0.8100638 0.8270499 0.8272959 0.8454041 0.8707587
## Modele_Backward 0.7735632 0.8164696 0.8295486 0.8281059 0.8449870 0.8693328
##                 NA's
## Modele1_TP1        0
## Modele2_TP1        0
## Modele_Backward    0
```

```r
bwplot(all_results, metric = "RMSE", main = "Comparaison des RMSE par Validation Croisée")
```




```r
# Étape 4 : Comparaison de la qualité d'ajustement et de la complexité (AIC/BIC)
model_comparison_df <- data.frame(
  Modele = c("Modèle 1 (TP1)", "Modèle 2 (TP1)", "Modèle Backward"),
  AIC = c(AIC(cv_model1$finalModel), AIC(cv_model2$finalModel), AIC(backward_model)),
  BIC = c(BIC(cv_model1$finalModel), BIC(cv_model2$finalModel), BIC(backward_model)),
  Nombre_Variables = c(length(coef(cv_model1$finalModel)), length(coef(cv_model2$finalModel)),
                       length(coef(backward_model)))
)

kable(model_comparison_df,
      caption = "Comparaison des modèles selon les critères AIC et BIC.",
      digits = 1)
```

Table 1: Comparaison des modèles selon les critères AIC et BIC.

Modele	AIC	BIC	Nombre_Variables
Modèle 1 (TP1)	14938.8	15018.1	14
Modèle 2 (TP1)	14893.6	14978.2	15
Modèle Backward	14890.9	14975.5	15

L'analyse de la performance prédictive via la validation croisée suggère que le Modèle 2 et le Modèle Backward sont nettement plus performants que le Modèle 1, validant l'importance de la modélisation de la non-linéarité et d'une sélection de variables pertinente. Cependant, la comparaison par RMSE seule est délicate, car les modèles analysés ne reposent pas sur le même ensemble de variables. Ainsi, une différence de performance peut provenir à la fois du choix des prédicteurs et de la structure du modèle. Pour pallier cette limite, les critères d'information AIC et BIC ont été utilisés : ils comparent la qualité d'ajustement en intégrant explicitement une pénalisation de la complexité. Cela permet une lecture plus cohérente des résultats. Selon ces critères, le Modèle Backward (AIC: 14890.9, BIC: 14975.5) possède un léger avantage théorique sur notre Modèle 2 (AIC: 14893.6, BIC: 14978.2). Notre démarche manuelle a donc produit un modèle dont la performance est statistiquement quasi-indiscernable de celle du modèle trouvé par l'algorithme de sélection descendante. Ces deux approches reposent néanmoins sur une sélection "dure" des variables : un prédicteur est soit inclus, soit totalement exclu. Cette méthode peut être suboptimale en présence de forte multicolinéarité, où plusieurs variables portent une information similaire. C'est précisément pour répondre à ces limites que nous nous tournons maintenant vers les méthodes de régularisation.

La régression Ridge nous permettra de gérer la multicolinéarité en conservant toutes les variables, mais en réduisant l'influence des prédicteurs redondants. La régression Lasso effectuera une sélection de variables plus robuste et intégrée, en pénalisant les coefficients jusqu'à les annuler, offrant ainsi un modèle parcimonieux.

1.2 Préparation des données pour la régularisation

La première étape consiste à préparer le jeu de données complet, incluant l'ingénierie des variables réalisée lors du TP1.

```r
# Définition de la variable cible Y et de la matrice des prédicteurs X
Y <- data_Mexico$Total

# Création de la matrice de design X
X_formula <- model.matrix(Total ~ T2M + T2Mmax + T2Mmin + RH + SSRD + STRD + Covid + Holidays + I(T2M^2)
                          factor(DOW) + sin_TOY + cos_TOY, data = data_Mexico)

# On retire la colonne de l'intercept car glmnet la gère par défaut
X <- X_formula[, -1]

# Centrage et réduction des données (INDISPENSABLE pour les modèles régularisés)
X_scaled <- scale(X)
```

Notre matrice de prédicteurs contient désormais 17 variables, incluant les variables corrélées et les indicatrices pour les jours de la semaine.

1.3 Méthode de Régression Ridge (L2)

Pour implémenter les régressions Ridge et Lasso, nous utilisons le package glmnet. Ce choix est motivé par sa grande efficacité de calcul, particulièrement sur de grands jeux de données, et sa flexibilité. Il permet de

gérer la régression Ridge (alpha = 0), Lasso (alpha = 1) dans un cadre unique et optimisé. Des packages plus anciens comme lars sont spécifiques au Lasso et souvent moins rapides.

La régression Ridge est particulièrement efficace pour gérer la multicolinéarité. Elle pénalise la somme des carrés des coefficients, ce qui a pour effet de "rétrécir" les coefficients des variables corrélées les uns vers les autres, sans pour autant les annuler. La fonction à minimiser est :

$$\Phi(\beta) = \|Y - X\beta\|_2^2 + \lambda\|\beta\|_2^2$$

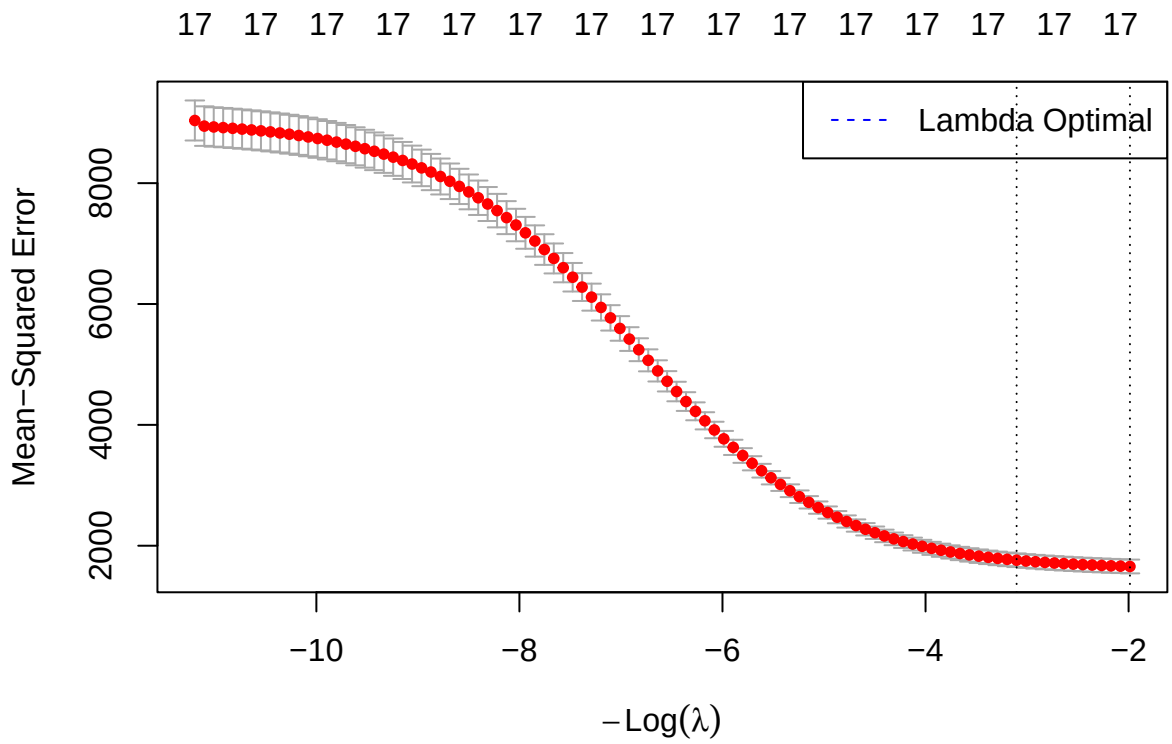
1.3.1 Recherche du paramètre optimal λ

Nous utilisons la validation croisée pour trouver la valeur de λ qui minimise l'erreur de prédiction.

```
# Effectuons la validation croisée
cv_ridge <- cv.glmnet(X_scaled, Y, alpha = 0)
lambda_optimal_ridge <- cv_ridge$lambda.min
cat("Lambda optimal pour le modèle Ridge :", lambda_optimal_ridge, "\n")
```

```
## Lambda optimal pour le modèle Ridge : 7.290536
```

```
# Utilisation du graphique standard de cv.glmnet
plot(cv_ridge)
abline(v = log(lambda_optimal_ridge), col = "blue", lty = 2, lwd = 2)
legend("topright", legend = "Lambda Optimal", col = "blue", lty = 2)
```



Ce graphe présente l'erreur quadratique moyenne (MSE) en fonction de -Log(λ). Cette convention de

visualisation est standard pour le package glmnet et oriente l'axe de telle sorte que les modèles les plus simples (associés à une forte pénalité λ) sont à gauche, et les modèles les plus complexes (associés à une faible pénalité λ) sont à droite.

Une observation pertinente est l'absence de la ligne verticale pointillée indiquant le lambda optimal sur la figure. La raison est que la valeur optimale trouvée, $\lambda \approx 7.29$, correspond à une valeur de -Log(λ) ≈ -1.99, qui se situe bien au-delà de la limite droite du graphique tracé par défaut.

Cela signifie que le minimum de l'erreur de validation croisée est atteint pour un très faible niveau de régularisation. En d'autres termes, le modèle le plus performant est celui qui se rapproche d'une régression linéaire classique (sans pénalisation). Cela constitue un premier enseignement clé : bien que la multicolinéarité entre les prédicteurs de température et de radiation soit présente, elle n'est pas suffisamment sévère pour dégrader la performance prédictive au point de nécessiter une forte régularisation. Le modèle Ridge n'a besoin que d'une pénalité très faible pour se stabiliser.

1.3.2 Analyse des coefficients

Visualisons maintenant comment les coefficients évoluent avec λ. C'est ce qu'on appelle le "chemin de régularisation".

```
# Ajuster le modèle Ridge
ridge_model <- glmnet(X_scaled, Y, alpha = 0)

# Graphique de la trajectoire
plot(ridge_model, xvar = "lambda", label = FALSE, main = "Chemin de régularisation Ridge")
```

Le graphique de la trajectoire des coefficients utilise également -Log(λ) sur l'axe des abscisses. Ainsi, se déplacer de gauche à droite sur le graphique correspond à une diminution de la pénalité lambda, ce qui augmente la complexité du modèle. On observe que les coefficients, initialement nuls pour une pénalité très forte (modèle le plus simple, à gauche), augmentent en magnitude à mesure que la pénalité diminue (vers la droite). Conformément à la théorie de la régression Ridge, les coefficients sont « rétrécis » (shrunk) vers zéro mais ne l'atteignent jamais exactement, ce qui signifie que le modèle conserve toutes les variables.

Extrayons les coefficients pour le λ optimal afin de voir les variables les plus influentes.

```
coef_ridge <- predict(ridge_model, type = "coefficients", s = lambda_optimal_ridge)
coef_ridge_matrix <- as.matrix(coef_ridge)
# Afficher les coefficients les plus importants en valeur absolue
top_coef_ridge <- head(coef_ridge_matrix[order(abs(coef_ridge_matrix[,1]), decreasing = TRUE), , drop =
print(top_coef_ridge)
```

```
##              s=7.290536
## (Intercept)  880.697684
## factor(DOW)6 -29.479458
## cos_TOY      -27.297521
## I(T2M^2)      13.687681
## sin_TOY      -13.491109
## SSRD          12.397081
## Holidays     -12.192475
## factor(DOW)2   9.570594
## STRD           9.253297
## factor(DOW)3   8.922166
## Covid         -8.821120
## T2Mmin         7.845167
```

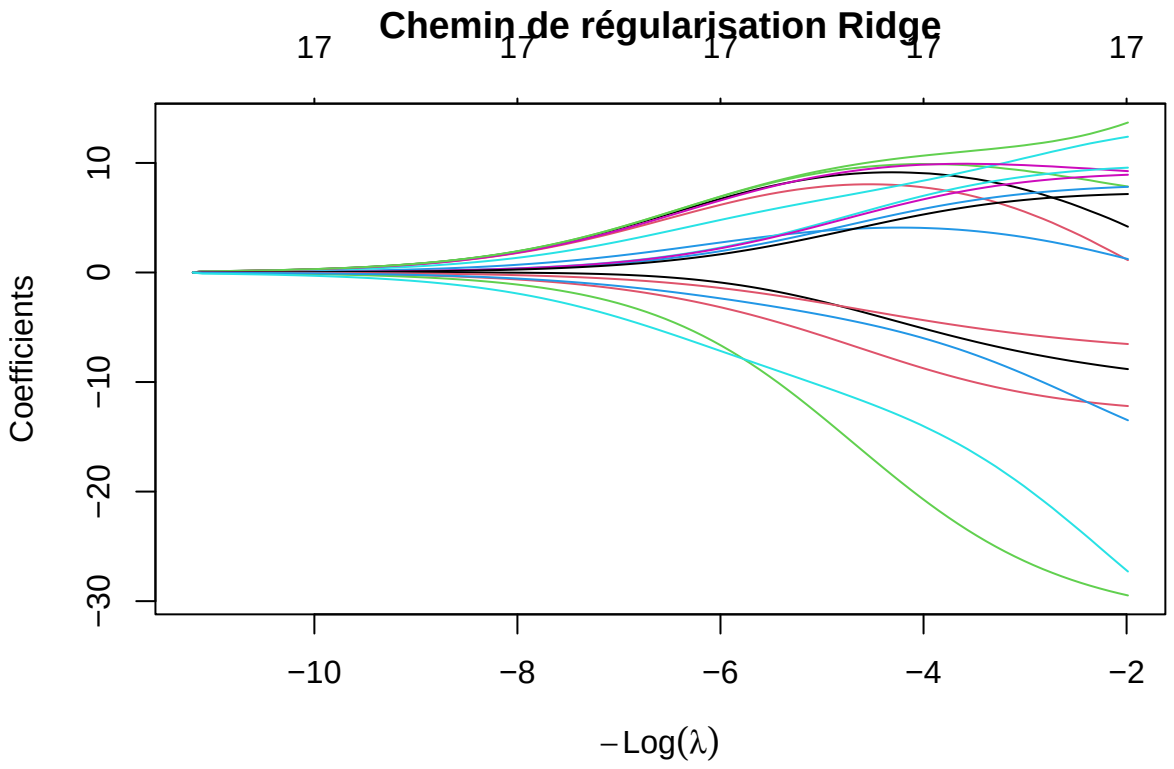


Figure 1: Trajectoire des coefficients de la régression Ridge. Les variables corrélées sont gérées collectivement.

```
## factor(DOW)1   7.819091
## factor(DOW)4   7.162708
## factor(DOW)5  -6.530065
## T2M            4.181237
## RH             1.224189
## T2Mmax         1.124499
```

On remarque que les variables que nous avions identifiées comme importantes dans le TP1 (`cos_TOY`, `factor(DOW)6`, `I(T2M^2)`) ont des coefficients élevés. Fait intéressant, Ridge a réparti le "poids" entre les variables de température (`T2M`, `T2Mmax`, `T2Mmin`) plutôt que d'en éliminer.

1.4 Méthode de Régression Lasso (L1)

La régression Lasso pénalise la somme des valeurs absolues des coefficients. Sa principale caractéristique est qu'elle peut réduire certains coefficients à exactement zéro, réalisant ainsi une sélection de variables automatique. Elle est très utile pour créer des modèles plus simples (parcimonieux). La fonction à minimiser est :

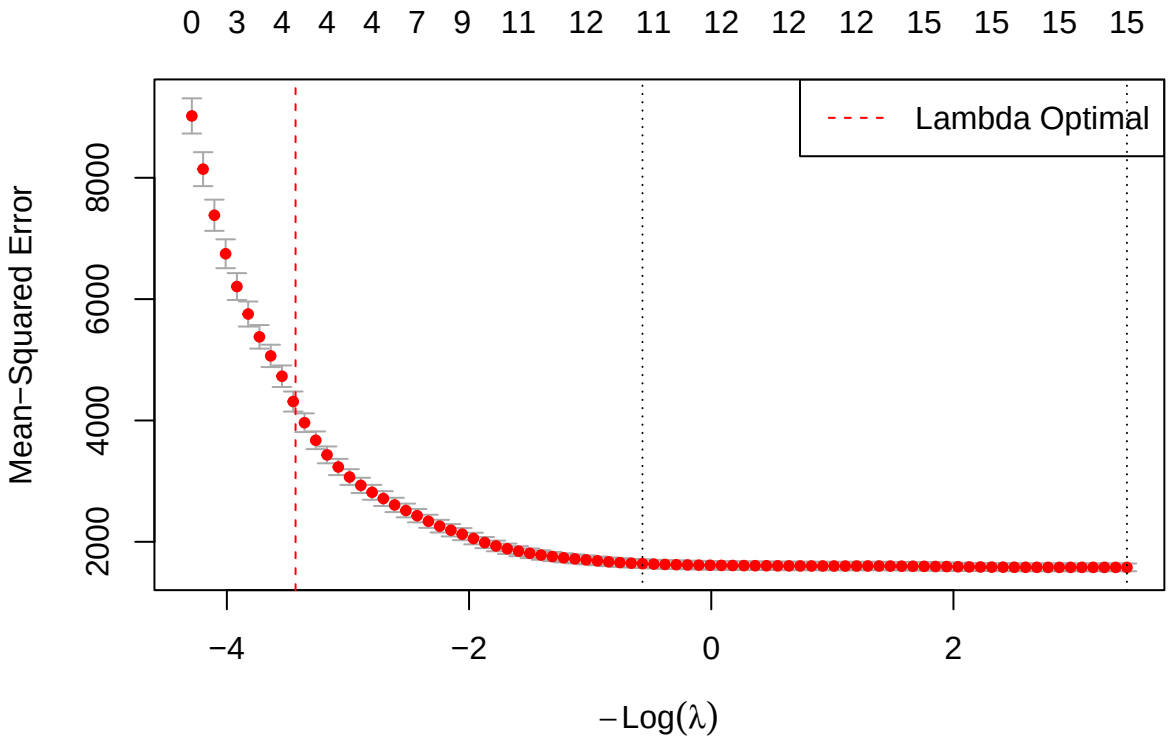
$$\Phi(\beta) = \|Y - X\beta\|_2^2 + \lambda\|\beta\|_1$$

1.4.1 Recherche du paramètre optimal λ

```r
cv_lasso <- cv.glmnet(X_scaled, Y, alpha = 1) # alpha = 1 pour Lasso

# Lambda optimal
lambda_optimal_lasso <- cv_lasso$lambda.min
cat("Lambda optimal pour le modèle Lasso :", lambda_optimal_lasso, "\n")
```

```
## Lambda optimal pour le modèle Lasso : 0.0323016
```

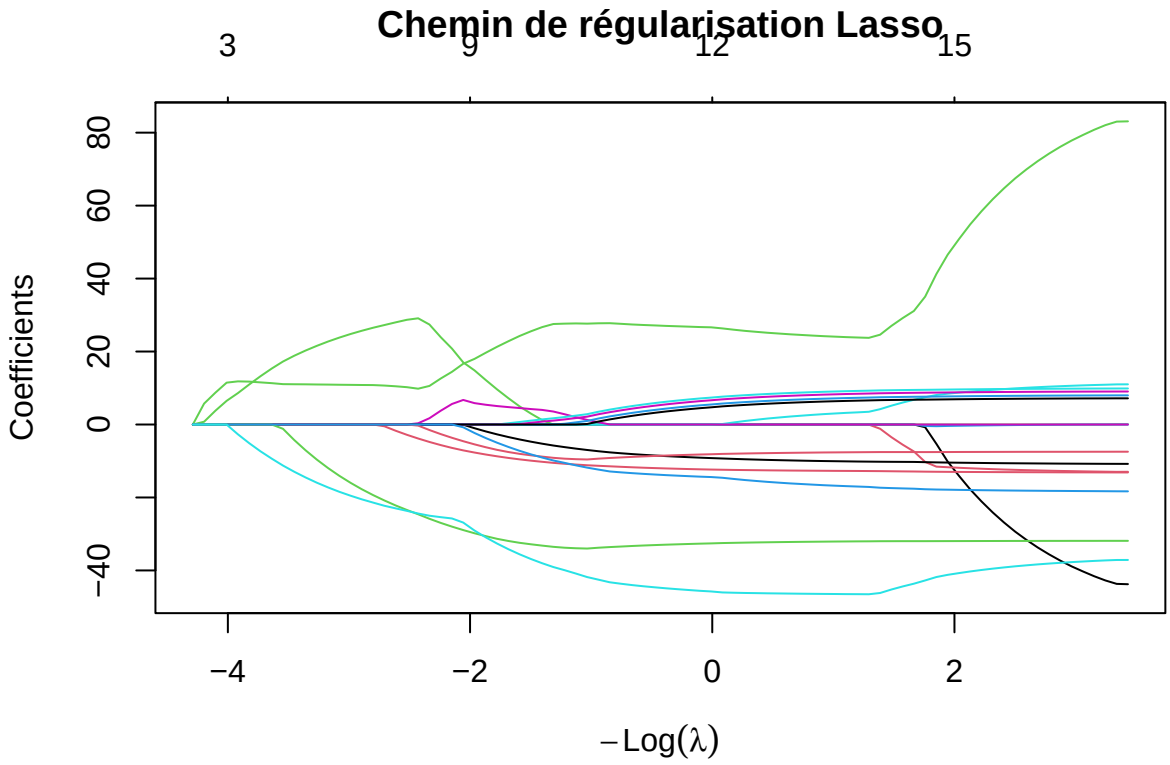
```r
# Graphique
plot(cv_lasso)
abline(v = log(lambda_optimal_lasso), col = "red", lty = 2)
legend("topright", legend = "Lambda Optimal", col = "red", lty = 2)
```



Comme pour la régression Ridge, ce graphique montre l'erreur de prédiction en fonction de -Log(λ). Se déplacer de gauche à droite équivaut à tester des modèles de plus en plus complexes (avec une pénalité lambda plus faible). La ligne pointillée rouge indique la position du lambda optimal qui minimise cette erreur. Le nombre au-dessus du graphique indique le nombre de variables non-nulles pour un lambda donné. Le nombre en haut du graphique indique le nombre de variables non-nulles pour un λ donné.

1.4.2 Analyse des coefficients et sélection de variables

```r
# Ajuster le modèle Lasso
lasso_model <- glmnet(X_scaled, Y, alpha = 1)

# Graphique de la trajectoire
plot(lasso_model, xvar = "lambda", label = FALSE, main = "Chemin de régularisation Lasso")
```



Le chemin de régularisation du Lasso est particulièrement instructif car il montre l'ordre dans lequel les variables sont éliminées du modèle. En lisant le graphique de droite à gauche (complexité décroissante), on peut identifier les variables les plus robustes comme étant celles qui "survivent" le plus longtemps avant que leur coefficient ne soit annulé.

Extrayons les coefficients non-nuls pour notre λ optimal. C'est le modèle final sélectionné par le Lasso.

```
# Extraire les coefficients pour le lambda optimal
coef_lasso <- predict(lasso_model, type = "coefficients", s = lambda_optimal_lasso)

# Variables sélectionnées (coefficients non-nuls)
variables_selectionnees <- coef_lasso[which(coef_lasso[, 1] != 0), , drop = FALSE]

cat(paste("Nombre de variables sélectionnées par le Lasso :", nrow(variables_selectionnees) - 1, "\n"))
```

```
## Nombre de variables sélectionnées par le Lasso : 15
```

```
print(variables_selectionnees)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                s=0.0323016
## (Intercept)  880.6976842
## T2M          -43.7754323
## T2Mmax       -12.9610557
## T2Mmin         0.0515399
## SSRD          11.0109957
```

```
## Covid        -10.7889015
## Holidays     -13.1392546
## I(T2M^2)      83.0952177
## factor(DOW)1   7.9876405
## factor(DOW)2   9.8789721
## factor(DOW)3   9.0452866
## factor(DOW)4   7.1779854
## factor(DOW)5  -7.4613584
## factor(DOW)6 -31.8865830
## sin_TOY      -18.3175877
## cos_TOY      -37.1299644
```

1.4.3 Interprétation Métier du Modèle Lasso

Plus qu'un simple outil de sélection, le modèle Lasso final transforme les coefficients standardisés en leviers d'analyse métier interprétables, offrant une lecture riche et structurée des moteurs de la consommation électrique :

Rythmes Saisonniers et Hebdomadaires : Le coefficient de **-37.12 pour cos_TOY** met en évidence une opposition marquée entre hiver et été, tandis que celui de **-31.88 pour factor(DOW)6 (Samedi)** reflète clairement la diminution de la consommation liée au ralentissement de l'activité le week-end.

Moteurs Météorologiques Clés : Le modèle retient à la fois l'effet linéaire négatif de **T2M (-43.77)** et l'importance du terme quadratique **I(T2M^2) (+83.09)**, indiquant une relation non linéaire où la consommation augmente aux températures extrêmes (froid ou chaud).

Impact des Événements Externes : Les variables **Holidays (-13.13)** et **Covid (-10.78)** présentent des effets négatifs mesurables, traduisant une baisse de la consommation lors des jours fériés et des périodes de restrictions sanitaires.

1.5 Comparaison Ridge vs Lasso :

Maintenant que nous avons ajusté et analysé les modèles Ridge et Lasso, nous pouvons les comparer objectivement sur la base de leurs performances prédictives et de leur structure.

1.5.1 Analyse des Métriques de Performance

Pour réaliser une évaluation complète, nous nous basons sur trois critères clés : la performance de généralisation (RMSE par validation croisée), la qualité d'ajustement (R^2 ajusté) et la parcimonie du modèle (nombre de variables).

```r
rmse_cv_ridge <- sqrt(cv_ridge$cvm[cv_ridge$lambda == cv_ridge$lambda.min])
rmse_cv_lasso <- sqrt(cv_lasso$cvm[cv_lasso$lambda == cv_lasso$lambda.min])

# Prédictions sur l'ensemble des données pour le modèle final
pred_ridge <- predict(cv_ridge, newx = X_scaled, s = "lambda.min")
pred_lasso <- predict(cv_lasso, newx = X_scaled, s = "lambda.min")

# Informations de base pour les calculs
n <- nrow(X_scaled)
k_ridge <- ncol(X_scaled) # Ridge conserve toutes les variables
k_lasso <- cv_lasso$nzero[cv_lasso$lambda == cv_lasso$lambda.min] # Lasso n'en garde qu'une partie
```

```r
# Calcul du R-carré ajusté
rsquared_ridge <- cor(Y, pred_ridge)^2
rsquared_lasso <- cor(Y, pred_lasso)^2

adj_rsquared_ridge <- 1 - (((1 - rsquared_ridge) * (n - 1)) / (n - k_ridge - 1))
adj_rsquared_lasso <- 1 - (((1 - rsquared_lasso) * (n - 1)) / (n - k_lasso - 1))

# Création du Tableau Comparatif

# Création du data.frame
metrics_df <- data.frame(
  Métrique = c("RMSE (par Validation Croisée)",
               "R² ajusté (sur données complètes)",
               "Nombre de Variables Finales"),
  Ridge = c(rmse_cv_ridge, adj_rsquared_ridge, k_ridge),
  Lasso = c(rmse_cv_lasso, adj_rsquared_lasso, k_lasso)
)

# Affichage du tableau formaté
knitr::kable(metrics_df,
      caption = "Évaluation Comparative des Modèles Ridge et Lasso.",
      digits = 4,
      align = 'lcc')
```

Table 2: Évaluation Comparative des Modèles Ridge et Lasso.

Métrique	Ridge	Lasso
RMSE (par Validation Croisée)	40.6987	39.7330
R^2 ajusté (sur données complètes)	0.8199	0.8286
Nombre de Variables Finales	17.0000	15.0000

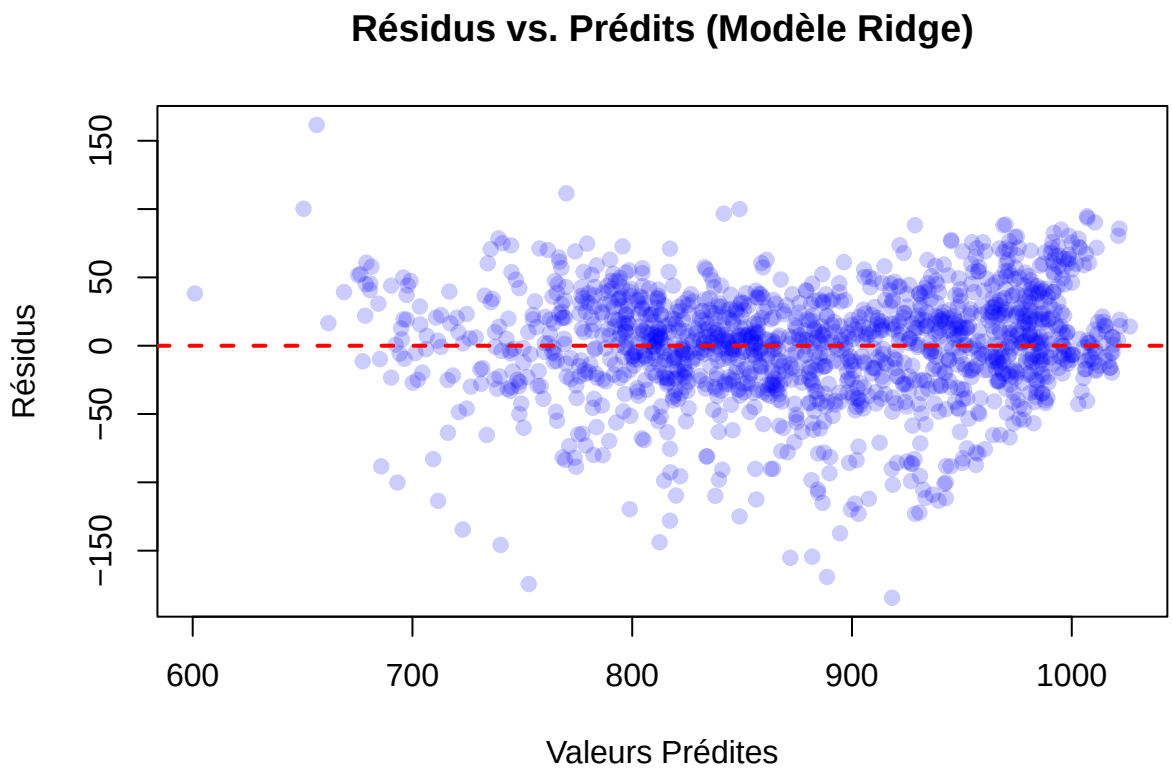
Le modèle Lasso présente un RMSE légèrement inférieur (39.77 contre 40.60), indiquant une meilleure capacité de généralisation. Cette supériorité, bien que modeste, est confirmée par un R^2 ajusté légèrement plus élevé (0.8286 pour Lasso contre 0.8199 pour Ridge), qui mesure la qualité d'ajustement du modèle final sur l'ensemble des données tout en pénalisant la complexité. De plus, le modèle Ridge a conservé les 17 variables initiales. Il les a toutes jugées utiles, mais a réduit l'influence des variables redondantes. Le modèle est performant mais reste complexe. Le modèle Lasso, lui, n'a conservé que 15 variables. Il a activement réalisé une sélection de variables en attribuant un coefficient de zéro aux prédicteurs jugés non pertinents en présence des autres (comme STRD par exemple).

1.5.2 Analyse Diagnostique des Résidus

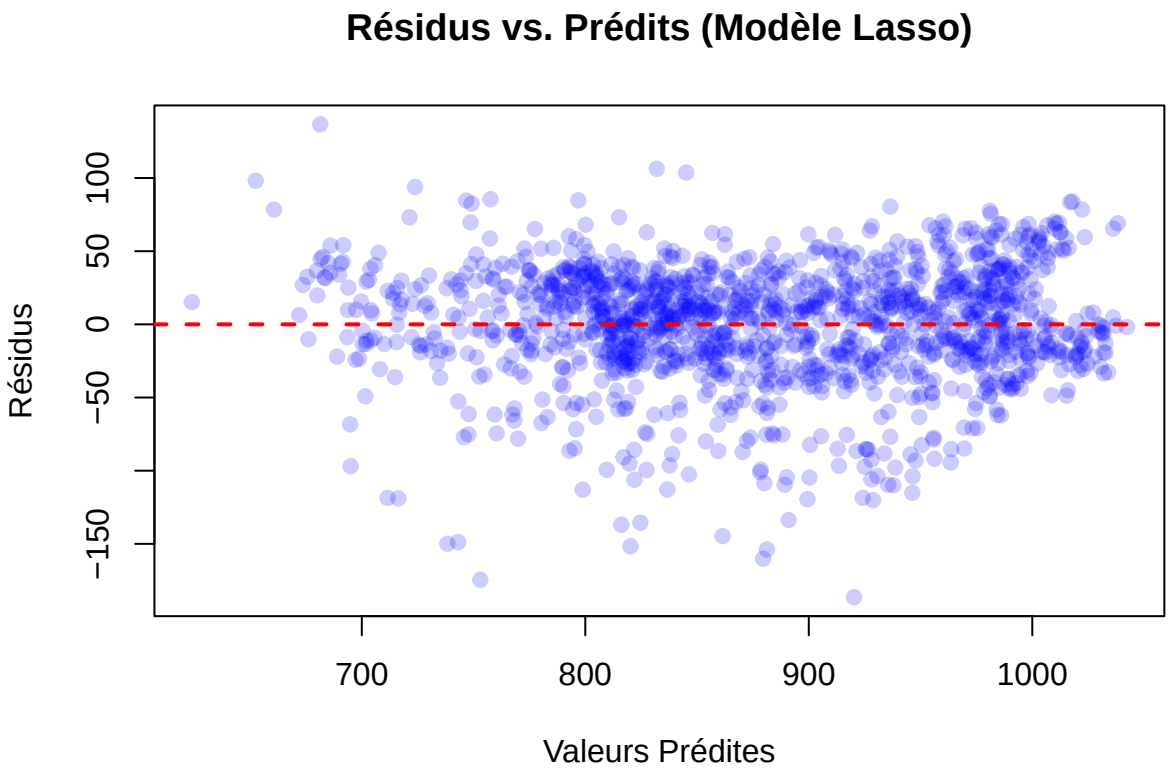
Au-delà des métriques de performance, il est essentiel d'analyser les résidus pour valider la qualité de nos modèles. Un graphique des résidus en fonction des valeurs prédites nous permet de vérifier des hypothèses clés

```r
# Étape 1 : Calculer les prédictions et les résidus

# Prédictions sur l'ensemble des données pour le lambda optimal
predictions_ridge <- predict(cv_ridge, newx = X_scaled, s = "lambda.min")
predictions_lasso <- predict(cv_lasso, newx = X_scaled, s = "lambda.min")
```

```r
# Calcul manuel des résidus = Valeurs observées (Y) - Valeurs prédites
residuals_ridge <- Y - predictions_ridge
residuals_lasso <- Y - predictions_lasso


# Étape 2 : Configuration et création des graphiques

# Graphique 1 : Résidus du modèle Ridge
plot(x = predictions_ridge, y = residuals_ridge,
     main = "Résidus vs. Prédits (Modèle Ridge)",
     xlab = "Valeurs Prédites",
     ylab = "Résidus",
     pch = 19, col = rgb(0, 0, 1, alpha = 0.2))
# Ajoute une ligne de référence à zéro
abline(h = 0, col = "red", lwd = 2, lty = 2)
```




```r
# Graphique 2 : Résidus du modèle Lasso
plot(x = predictions_lasso, y = residuals_lasso,
     main = "Résidus vs. Prédits (Modèle Lasso)",
     xlab = "Valeurs Prédites",
     ylab = "Résidus",
     pch = 19, col = rgb(0, 0, 1, alpha = 0.2))
# Ajoute une ligne de référence à zéro
abline(h = 0, col = "red", lwd = 2, lty = 2)
```



L'examen des graphiques pour les modèles Ridge et Lasso nous amène aux conclusions suivantes :

Absence de Structure Systématique : Le résultat le plus important est que, pour les deux modèles, le nuage de points est réparti de manière aléatoire et ne présente **aucune structure systématique évidente**. Contrairement au Modèle 1 du TP1 qui affichait une forme de parabole (indiquant une relation non-linéaire mal modélisée), ces graphiques montrent un "bruit blanc". Cela confirme que la structure du modèle, notamment l'inclusion du terme quadratique pour la température (I(T2M^2)), a été efficace pour capturer les dynamiques non-linéaires sous-jacentes.

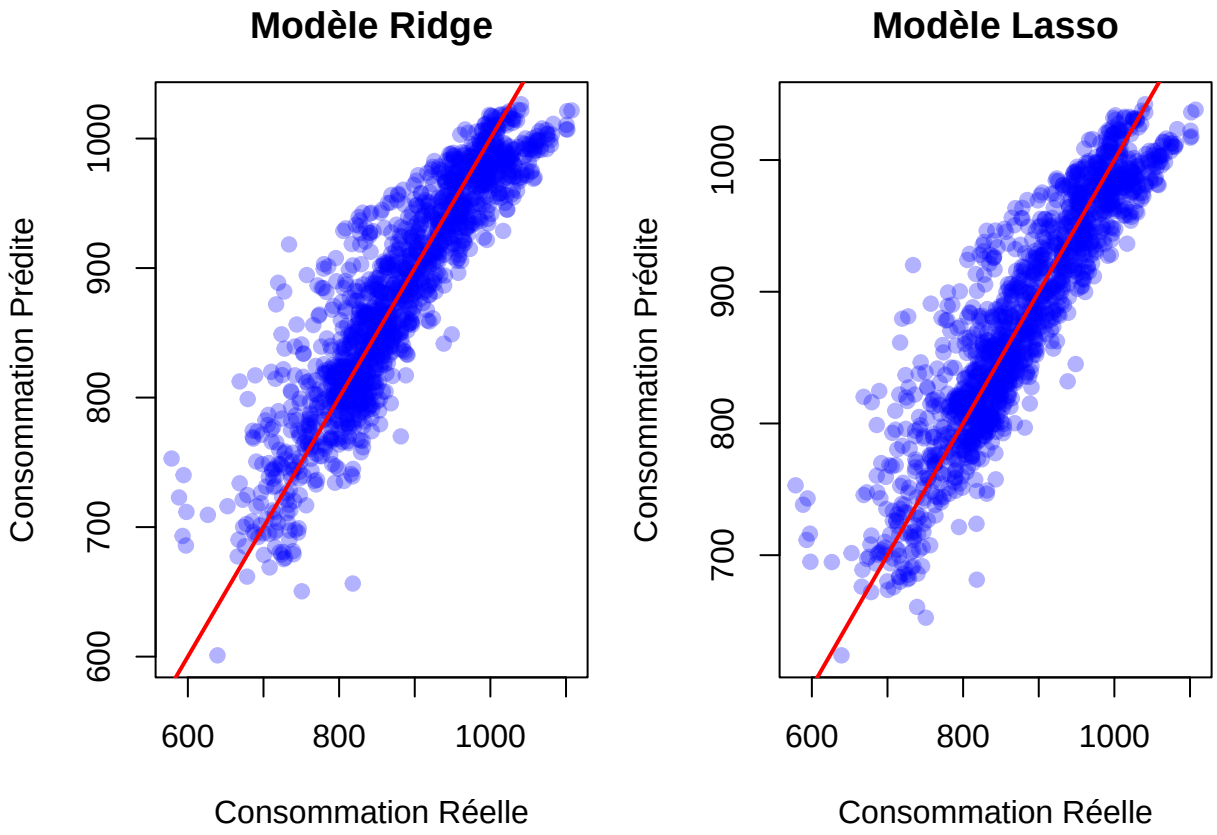
Validation de l'Hypothèse d'Homoscédasticité : L'hypothèse de variance constante des erreurs (homoscédasticité) semble **globalement respectée**. La dispersion verticale des résidus reste relativement stable sur l'ensemble des valeurs prédites.

Comparaison Directe entre Ridge et Lasso : Les deux graphiques sont **quasiment identiques**. La forme du nuage de points, sa dispersion et sa centration autour de la ligne zéro sont indiscernables entre le modèle Ridge et le modèle Lasso. Cette similarité visuelle renforce la conclusion tirée des métriques quantitatives : les deux modèles ont une performance prédictive très proche. Du point de vue du diagnostic des résidus, aucun des deux modèles ne présente un avantage ou un défaut par rapport à l'autre. En conclusion, l'analyse des résidus est très positive et valide la robustesse des deux modèles finaux. Elle confirme qu'ils sont bien spécifiés et ne souffrent pas de défauts structurels majeurs.

1.5.3 Visualisation de la Performance Prédictive

Pour synthétiser la performance globale de manière intuitive, nous pouvons visualiser les valeurs prédites par chaque modèle par rapport aux valeurs réelles observées.

```r
predictions_ridge <- predict(cv_ridge, newx = X_scaled, s = "lambda.min")
predictions_lasso <- predict(cv_lasso, newx = X_scaled, s = "lambda.min")

# Configuration de la fenêtre graphique pour afficher 2 graphiques (1 ligne, 2 colonnes)
par(mfrow = c(1, 2), mar = c(4, 4, 3, 1))

# Graphique 1 : Régression Ridge
plot(x = Y, y = predictions_ridge,
     main = "Modèle Ridge",
     xlab = "Consommation Réelle",
     ylab = "Consommation Prédite",
     pch = 19,
     col = rgb(0, 0, 1, alpha = 0.3)
)

# Ajoute la ligne de "prédiction parfaite" (y=x)
abline(a = 0, b = 1, col = "red", lwd = 2)


#  Graphique 2 : Régression Lasso
plot(x = Y, y = predictions_lasso,
     main = "Modèle Lasso",
     xlab = "Consommation Réelle",
     ylab = "Consommation Prédite",
     pch = 19,
     col = rgb(0, 0, 1, alpha = 0.3)
)

# Ajoute la ligne de "prédiction parfaite" (y=x)
abline(a = 0, b = 1, col = "red", lwd = 2)
```



Les nuages de points des deux modèles sont bien alignés sur la droite de prédiction parfaite (y=x), confirmant leur bonne qualité globale.

Le choix final entre Ridge et Lasso ne peut donc pas se faire sur la base de ce diagnostic, mais bien sur les critères de performance prédictive (léger avantage pour Lasso) et de parcimonie (avantage clair pour Lasso), comme discuté précédemment.

1.6 Conclusion Générale

Là où notre analyse précédente (TP1) nous avait contraints à une sélection de variables manuelle et itérative pour surmonter la multicolinéarité, ce travail a démontré la puissance et l'efficacité des approches régularisées pour automatiser ce processus critique.

La Régression Ridge a fourni un modèle robuste en conservant l'ensemble des prédicteurs tout en modulant leur influence. Toutefois, la **Régression Lasso s'est avérée être la plus concluante**. En opérant une sélection de variables intégrée, elle a non seulement produit un modèle plus **parcimonieux** (15 variables contre 17), mais a également affiché une **performance prédictive supérieure**, comme en témoigne son RMSE plus faible.

De manière remarquable, cette sélection algorithmique a largement **validé notre démarche manuelle du TP1**. En écartant les prédicteurs redondants, le Lasso a confirmé de manière objective leur faible apport informatif, justifiant ainsi nos intuitions initiales tout en offrant un cadre de travail plus rapide et rigoureux.

Finalement, le principal avantage de cette approche ne réside pas seulement dans l'optimisation des métriques. Il est dans la capacité du Lasso à générer un modèle final qui est non seulement performant, mais aussi **directement interprétable**. Comme l'analyse l'a montré, le modèle a permis de quantifier avec clarté l'impact des différents moteurs de la consommation, transformant un problème statistique complexe en leviers d'analyse métier clairs et actionnables.

Ce travail illustre ainsi que les méthodes de régularisation sont bien plus que des outils de prédiction : elles constituent un pont essentiel entre la robustesse statistique et la pertinence opérationnelle.