TP3 - Exercice 2

EXPUESTO Ewen, AUBERTIN Candice

20 octobre 2025

L'objectif est de prédire si la quantité d'électricité générée (Total) est inférieure ou supérieure à la médiane, en utilisant un modèle de régression logistique sparse (LASSO).

1 Traitement des données

1.1 Importation des données

```
rm(list = ls())
graphics.off()
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-10
```

```
tab <- read.csv(file = "Mexico_data.csv", header = TRUE, sep = ",")
```

1.2 D'une description continue à une classification

On ajoute la variable y qui correspond à Total transformée en 1 ou 0 selon si la consommation d'électricité est supérieure ou inférieure à la médiane. Dans le tableau x des prédicteurs, on enlève la colonne "Total" ainsi que la colonne "TOY" (Time of Year) et la colonne "X0" qui correspond à la date DD-MM-YYYY.

```
median_total <- median(tab$Total)
y <- as.numeric(tab$Total > median_total)
tab$y <- y
x <- as.matrix(tab[, -which(names(tab) %in% c("Total", "TOY", "y", "X0"))])
```

Affichage des 10 premières lignes de tab, x et y :

```
head(tab, 10)
```

```
##            XO       RH     SSRD    STRD      T2M   T2Mmax    T2Mmin Covid
## 1  2019-01-01 54.88692 556483.6 1095579 13.61629 20.25525  8.480851     0
## 2  2019-01-02 61.36585 551500.0 1148553 13.64353 19.71474  9.270146     0
## 3  2019-01-03 60.81884 643224.1 1086882 13.25186 20.81150  7.662988     0
## 4  2019-01-04 55.19776 661235.1 1069402 14.09160 22.81910  7.496520     0
## 5  2019-01-05 54.77641 609757.8 1110888 15.24040 23.64965  8.615337     0
## 6  2019-01-06 63.39149 548361.9 1207957 16.86465 23.43387 12.072207     0
## 7  2019-01-07 67.53954 620992.0 1199764 17.26188 24.25773 11.968174     0
## 8  2019-01-08 65.65101 605556.2 1182445 16.97759 24.06361 11.249535     0
## 9  2019-01-09 66.31228 528989.8 1207520 16.66935 23.10667 11.913719     0
## 10 2019-01-10 69.47626 546061.0 1200844 16.02088 21.84510 11.625887     0
##    Holidays DOW TOY    Total y
## 1         1   1   1 588.3452 0
## 2         0   2   2 736.1832 0
## 3         0   3   3 793.0117 0
## 4         0   4   4 800.6797 0
## 5         0   5   5 762.7347 0
## 6         0   6   6 690.9642 0
## 7         0   0   7 774.4003 0
## 8         0   1   8 800.9579 0
## 9         0   2   9 805.3935 0
## 10        0   3  10 803.6239 0
```

```
head(x, 10)
```

```
##             RH     SSRD    STRD      T2M   T2Mmax    T2Mmin Covid Holidays DOW
##  [1,] 54.88692 556483.6 1095579 13.61629 20.25525  8.480851     0        1   1
##  [2,] 61.36585 551500.0 1148553 13.64353 19.71474  9.270146     0        0   2
##  [3,] 60.81884 643224.1 1086882 13.25186 20.81150  7.662988     0        0   3
##  [4,] 55.19776 661235.1 1069402 14.09160 22.81910  7.496520     0        0   4
##  [5,] 54.77641 609757.8 1110888 15.24040 23.64965  8.615337     0        0   5
##  [6,] 63.39149 548361.9 1207957 16.86465 23.43387 12.072207     0        0   6
##  [7,] 67.53954 620992.0 1199764 17.26188 24.25773 11.968174     0        0   0
##  [8,] 65.65101 605556.2 1182445 16.97759 24.06361 11.249535     0        0   1
##  [9,] 66.31228 528989.8 1207520 16.66935 23.10667 11.913719     0        0   2
## [10,] 69.47626 546061.0 1200844 16.02088 21.84510 11.625887     0        0   3
```

```
head(y, 10)
```

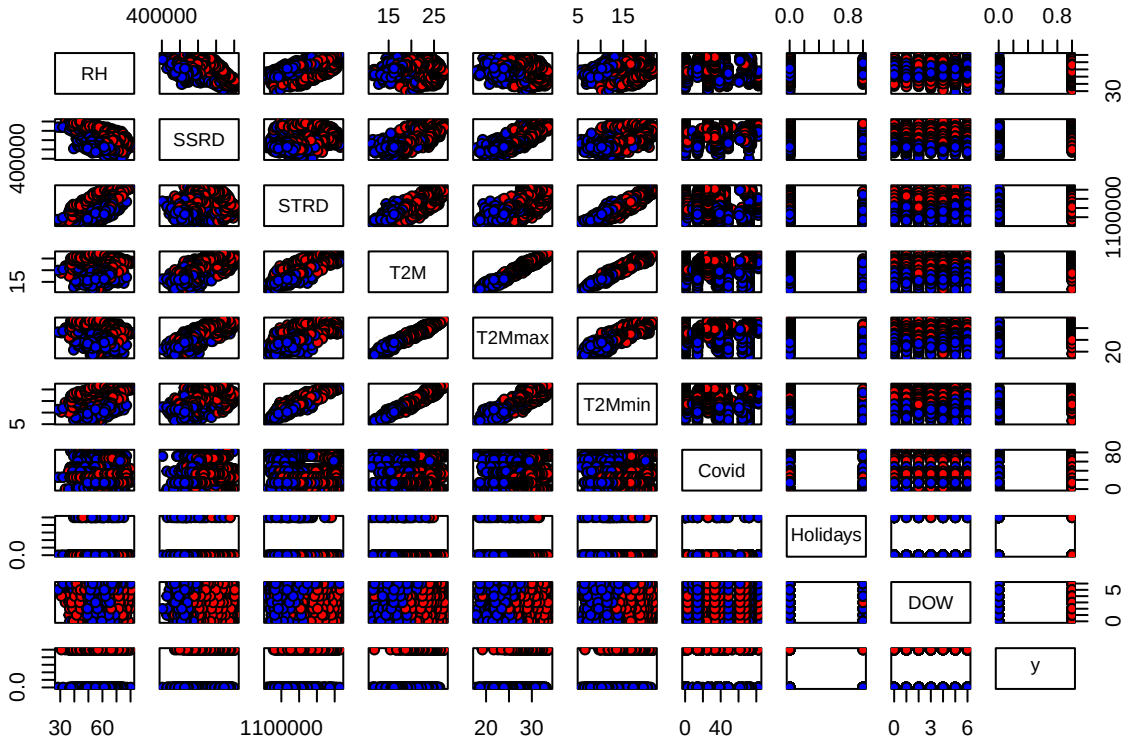
```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

1.3 Scatterplot

Scatterplot matrix des variables étudiées :

```
vars_pairs <- tab[, -c(1, which(names(tab) %in% c("Total", "TOY", "XO")))]

colors <- ifelse(tab$y == 1, "red", "blue")
pairs(vars_pairs, pch = 21, bg = colors)
```



2 Modèle logistique

Ajustement du modèle logistique :

```
res <- glm(y ~ . - TOY - X0 - Total, data = tab, family = binomial)
summary(res)
```

```
##
## Call:
## glm(formula = y ~ . - TOY - X0 - Total, family = binomial, data = tab)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.140e+00  5.731e+00   0.722 0.470048
## RH           6.161e-02  3.347e-02   1.841 0.065650 .
## SSRD         4.136e-06  1.239e-06   3.337 0.000846 ***
## STRD        -1.775e-05  6.151e-06  -2.885 0.003914 **
## T2M          5.747e-01  5.670e-01   1.014 0.310795
## T2Mmax      -4.511e-01  3.137e-01  -1.438 0.150386
## T2Mmin       8.199e-01  3.598e-01   2.279 0.022682 *
## Covid       -1.820e-02  3.426e-03  -5.312 1.08e-07 ***
## Holidays    -3.147e+00  7.636e-01  -4.121 3.76e-05 ***
## DOW         -4.320e-01  4.879e-02  -8.854  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2025.38  on 1460  degrees of freedom
## Residual deviance:  868.77  on 1451  degrees of freedom
## AIC: 888.77
##
## Number of Fisher Scoring iterations: 6
```

2.1 Résultats du modèle logistique

Récupère les coefficients du modèle logistique et les trie par ordre d'importance :

```
coeffs_data <- res$coefficients
coeffs_data <- coeffs_data[names(coeffs_data) != "(Intercept)"]

coeff_pos <- coeffs_data[coeffs_data > 0]
coeff_neg <- coeffs_data[coeffs_data < 0]

coeff_pos_sorted <- sort(coeff_pos)
coeff_neg_sorted <- sort(coeff_neg, decreasing = TRUE)
```

Importance des coefficients positifs :

```
coeff_pos_sorted
```

```
##         SSRD           RH          T2M       T2Mmin
## 4.136032e-06 6.160879e-02 5.747275e-01 8.199020e-01
```

Importance des coefficients négatifs :

```
coeff_neg_sorted
```

```
##          STRD         Covid           DOW        T2Mmax      Holidays
## -1.774482e-05 -1.819829e-02 -4.319861e-01 -4.511055e-01 -3.147058e+00
```

2.2 Performances du modèle logistique

On peut maintenant prédire les résultats avec la focntion `predict.glm`. `pred_link` et `pred_resp` représentent la prédiction du modèle à partir des mêmes données d'entraînement :

```
pred_link <- predict.glm(res, type = "link")
pred_resp <- predict.glm(res, type = "response")
```

En utilisant "link", le modèle logistique travaille linéairement (sur $]-\infty, +\infty[$) tandis qu'avec "response" il travaille avec une approche probabiliste (résultat sur $[0, 1]$).

2.2.1 Graphe des classes

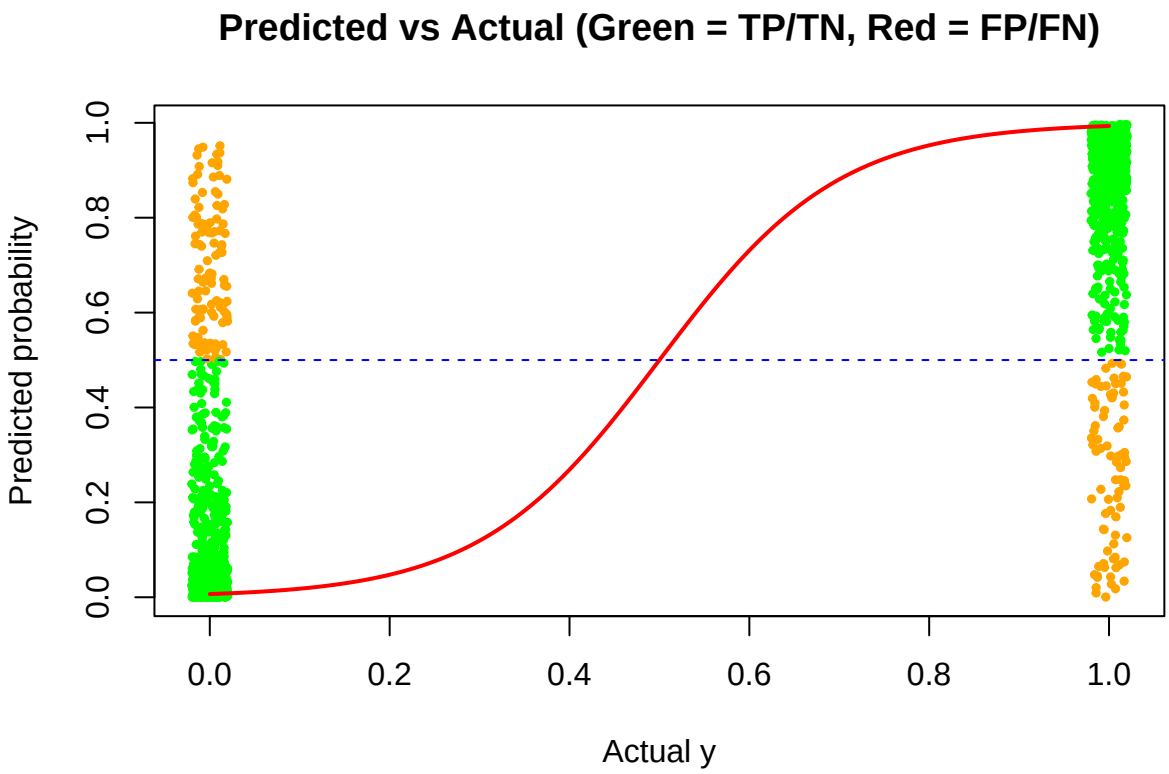
Graphe d'un modèle logistique superposé aux données classées en TN/TF/FP/FN :

```r
# Convert probabilities to predicted class
# with 0 being below the median and 1 above
pred_class <- ifelse(pred_resp >= 0.5, 1, 0)

point_colors <- ifelse(
  pred_class == y,
  "green", # True positives / True negatives
  "orange" # False positives / False negatives
)


plot(
  y + runif(length(y), -0.02, 0.02),
  pred_resp,
  xlab = "Actual y",
  ylab = "Predicted probability",
  main = "Predicted vs Actual (Green = TP/TN, Red = FP/FN)",
  col = point_colors,
  pch = 19,
  cex = 0.5
)

x_log <- seq(0, 1, length.out = 200)
y_log <- 1 / (1 + exp(-10 * (x_log - 0.5)))
lines(x_log, y_log, col = "red", lwd = 2)
abline(h = 0.5, col = "blue", lty = 2)
```



2.2.2 Matrices de confusion

Matrice de confusion :

```
pred_resp <- predict(res, type = "response")
y_pred <- ifelse(pred_resp > 0.5, 1, 0)
cm <- table(Predicted = y_pred, Actual = tab$y)
cm
```

```
##          Actual
## Predicted   0   1
##         0 618  83
##         1 113 647
```

```
tp <- cm[2, 2]
tn <- cm[1, 1]
fp <- cm[2, 1]
fn <- cm[1, 2]

accuracy <- (tp + tn) / sum(cm)
false_positive_rate <- fp / sum(cm)
false_negative_rate <- fn / sum(cm)
```

Accuracy (TP + TN) :

```
accuracy
```

```
## [1] 0.8658453
```

False positive rate :

```
false_positive_rate
```

```
## [1] 0.07734428
```

False negative rate :

```
false_negative_rate
```

```
## [1] 0.0568104
```

2.2.3 K-fold

Split data into K roughly equal parts (folds) then train on K-1 folds and test on the held-out fold. The accuracy is measured as TP + TN :

```
cv_perf <- function(data, kk = 5) {
  n <- nrow(data)
  folds <- sample(rep(1:kk, length.out = n))
  perf <- numeric(kk)
  for (k in 1:kk) {
    train <- data[folds != k, ]
    test  <- data[folds == k, ]
    fit <- glm(y ~ ., data = train, family = binomial)
    p <- predict(fit, newdata = test, type = "response")
    pred <- ifelse(p >= 0.5, 1, 0) # Convert to 0 and 1
    perf[k] <- mean(pred == test$y)
  }
  perf
}
```

Comparison K = 5 vs K = 10, larger K usually reduces bias of the performance estimate but can increase variance and computation cost :

```
input <- tab[, -c(1, which(names(tab) %in% c("Total", "TOY", "XO")))]
acc5  <- cv_perf(input, kk = 5)
acc10 <- cv_perf(input, kk = 10)
```

K = 5 accuracies (TP + TN) :

```
acc5
```

```
## [1] 0.8498294 0.8698630 0.8561644 0.8527397 0.8767123
```

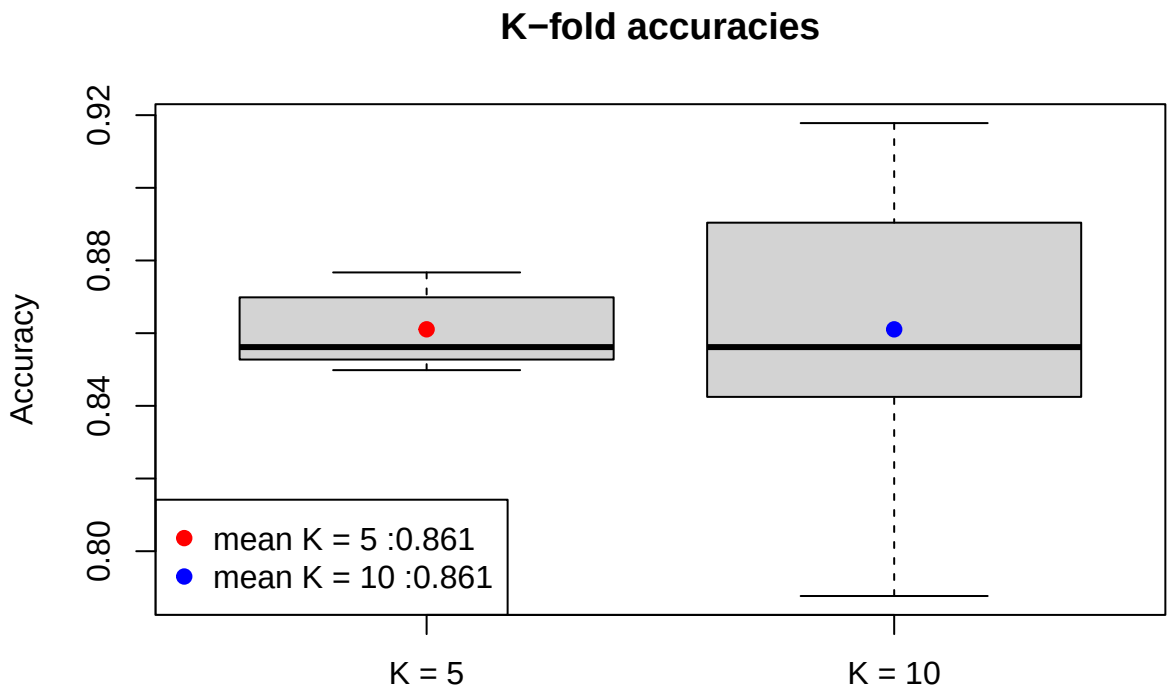
K = 10 accuracies (TP + TN) :

```
acc10
```

```
##  [1] 0.8639456 0.8630137 0.7876712 0.8493151 0.8424658 0.8904110 0.9109589
##  [8] 0.9178082 0.8424658 0.8424658
```

The boxplot of those K values shows central tendency, spread and outliers thus giving an idea of variability and stability of the estimator :

```
boxplot(acc5, acc10, names = c("K = 5", "K = 10"),
        ylab = "Accuracy", main = "K-fold accuracies")
points(1:2, c(mean(acc5), mean(acc10)), col = c("red", "blue"), pch = 19)
legend("bottomleft", legend = c(paste0("mean K = 5 :", round(mean(acc5), 3)),
                                paste0("mean K = 10 :", round(mean(acc10), 3))),
       pch = 19, col = c("red", "blue"))
```



Ainsi il y a une plus grande variabilité pour un plus grand nombre de folds comme prévu avec toutefois une précision de la prédicion à peine mieux.

3 Sélection d'un autre modèle ?

3.1 Régression logistique Forward

```r
resfor <- step(res, direction = "forward")
```

```
## Start:  AIC=888.77
## y ~ (X0 + RH + SSRD + STRD + T2M + T2Mmax + T2Mmin + Covid +
##     Holidays + DOW + TOY + Total) - TOY - X0 - Total
```

```r
resfor
```

```
##
## Call:  glm(formula = y ~ (X0 + RH + SSRD + STRD + T2M + T2Mmax + T2Mmin +
##     Covid + Holidays + DOW + TOY + Total) - TOY - X0 - Total,
##     family = binomial, data = tab)
##
## Coefficients:
## (Intercept)           RH         SSRD         STRD          T2M       T2Mmax
##   4.140e+00    6.161e-02    4.136e-06   -1.774e-05    5.747e-01   -4.511e-01
##      T2Mmin        Covid     Holidays          DOW
##   8.199e-01   -1.820e-02   -3.147e+00   -4.320e-01
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1451 Residual
## Null Deviance:       2025
## Residual Deviance: 868.8     AIC: 888.8
```

3.2 Régression logistique Backward

```r
resback <- step(res, direction = "backward")
```

```
## Start:  AIC=888.77
## y ~ (X0 + RH + SSRD + STRD + T2M + T2Mmax + T2Mmin + Covid +
##     Holidays + DOW + TOY + Total) - TOY - X0 - Total
##
##            Df Deviance    AIC
## - T2M       1   869.80 887.80
## <none>          868.77 888.77
## - T2Mmax    1   870.85 888.85
## - RH        1   872.20 890.20
## - T2Mmin    1   874.00 892.00
## - STRD      1   877.27 895.27
## - SSRD      1   880.03 898.03
## - Holidays  1   893.31 911.31
## - Covid     1   898.86 916.86
## - DOW       1   960.51 978.51
##
## Step:  AIC=887.8
## y ~ RH + SSRD + STRD + T2Mmax + T2Mmin + Covid + Holidays + DOW
##
##            Df Deviance    AIC
## - T2Mmax    1   871.17 887.17
## <none>          869.80 887.80
## - RH        1   872.48 888.48
```

```
## - STRD      1   878.12 894.12
## - SSRD      1   882.06 898.06
## - Holidays  1   894.27 910.27
## - Covid     1   899.81 915.81
## - T2Mmin    1   903.01 919.01
## - DOW       1   960.59 976.59
##
## Step:  AIC=887.17
## y ~ RH + SSRD + STRD + T2Mmin + Covid + Holidays + DOW
##
##            Df Deviance    AIC
## <none>          871.17 887.17
## - RH        1   878.43 892.43
## - STRD      1   879.14 893.14
## - SSRD      1   882.11 896.11
## - Holidays  1   895.05 909.05
## - Covid     1   900.83 914.83
## - T2Mmin    1   912.93 926.93
## - DOW       1   961.08 975.08
```

```
resback
```

```
##
## Call:  glm(formula = y ~ RH + SSRD + STRD + T2Mmin + Covid + Holidays +
##     DOW, family = binomial, data = tab)
##
## Coefficients:
## (Intercept)           RH         SSRD         STRD       T2Mmin        Covid
##   3.846e-02    7.257e-02    3.899e-06   -1.716e-05    9.830e-01   -1.805e-02
##    Holidays          DOW
##  -3.131e+00   -4.247e-01
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1453 Residual
## Null Deviance:       2025
## Residual Deviance: 871.2     AIC: 887.2
```

3.3 Régression logistique Stepwise

```
resstep <- step(res, direction = "both")
```

```
## Start:  AIC=888.77
## y ~ (X0 + RH + SSRD + STRD + T2M + T2Mmax + T2Mmin + Covid +
##     Holidays + DOW + TOY + Total) - TOY - X0 - Total
##
##            Df Deviance    AIC
## - T2M       1   869.80 887.80
## <none>          868.77 888.77
## - T2Mmax    1   870.85 888.85
## - RH        1   872.20 890.20
## - T2Mmin    1   874.00 892.00
## - STRD      1   877.27 895.27
```

```
## - SSRD      1   880.03 898.03
## - Holidays  1   893.31 911.31
## - Covid     1   898.86 916.86
## - DOW       1   960.51 978.51
##
## Step:  AIC=887.8
## y ~ RH + SSRD + STRD + T2Mmax + T2Mmin + Covid + Holidays + DOW
##
##            Df Deviance    AIC
## - T2Mmax    1   871.17 887.17
## <none>          869.80 887.80
## - RH        1   872.48 888.48
## + T2M       1   868.77 888.77
## - STRD      1   878.12 894.12
## - SSRD      1   882.06 898.06
## - Holidays  1   894.27 910.27
## - Covid     1   899.81 915.81
## - T2Mmin    1   903.01 919.01
## - DOW       1   960.59 976.59
##
## Step:  AIC=887.17
## y ~ RH + SSRD + STRD + T2Mmin + Covid + Holidays + DOW
##
##            Df Deviance    AIC
## <none>          871.17 887.17
## + T2Mmax    1   869.80 887.80
## + T2M       1   870.85 888.85
## - RH        1   878.43 892.43
## - STRD      1   879.14 893.14
## - SSRD      1   882.11 896.11
## - Holidays  1   895.05 909.05
## - Covid     1   900.83 914.83
## - T2Mmin    1   912.93 926.93
## - DOW       1   961.08 975.08
```

```
resstep
```

```
##
## Call:  glm(formula = y ~ RH + SSRD + STRD + T2Mmin + Covid + Holidays +
##     DOW, family = binomial, data = tab)
##
## Coefficients:
## (Intercept)           RH         SSRD         STRD       T2Mmin        Covid
##   3.846e-02    7.257e-02    3.899e-06   -1.716e-05    9.830e-01   -1.805e-02
##    Holidays          DOW
##  -3.131e+00   -4.247e-01
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1453 Residual
## Null Deviance:       2025
## Residual Deviance: 871.2     AIC: 887.2
```

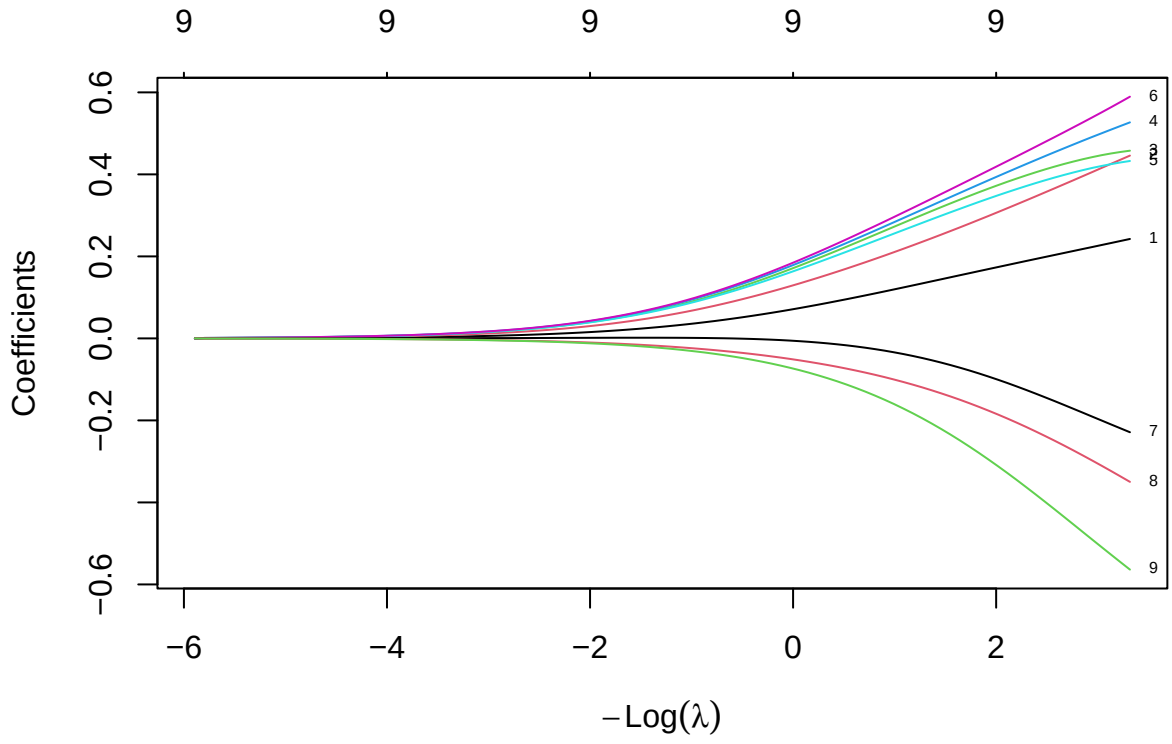
3.4 Régression logistique RIDGE

Découpage des données en deux groupes et détermination du modèle RIDGE :

```
split <- sample(c(TRUE, FALSE), nrow(tab), replace = TRUE, prob = c(0.8, 0.2))
train_tab <- tab[split, ]
test_tab <- tab[!split, ]

x_train <- as.matrix(train_tab[, -c(1, which(names(tab) %in%
                                               c("Total", "TOY", "XO", "y")))])
x_train_s <- scale(x_train)
y_train <- as.numeric(train_tab$y)
y_train_matrix <- matrix(y_train, length(y_train), 1)

rm <- glmnet(x_train_s, y_train_matrix, alpha = 0, family = "binomial")
plot(rm, xvar = "lambda", label = TRUE)
```



Détermination du λ optimal par deux méthodes avec le modèle RIDGE :

```
cv_ridge <- cv.glmnet(
  x_train_s,
  y_train_matrix,
  alpha = 0,
  family = "binomial",
  grouped = FALSE,
  nfolds = 10
```

```
)
cv_ridge$lambda.min
```

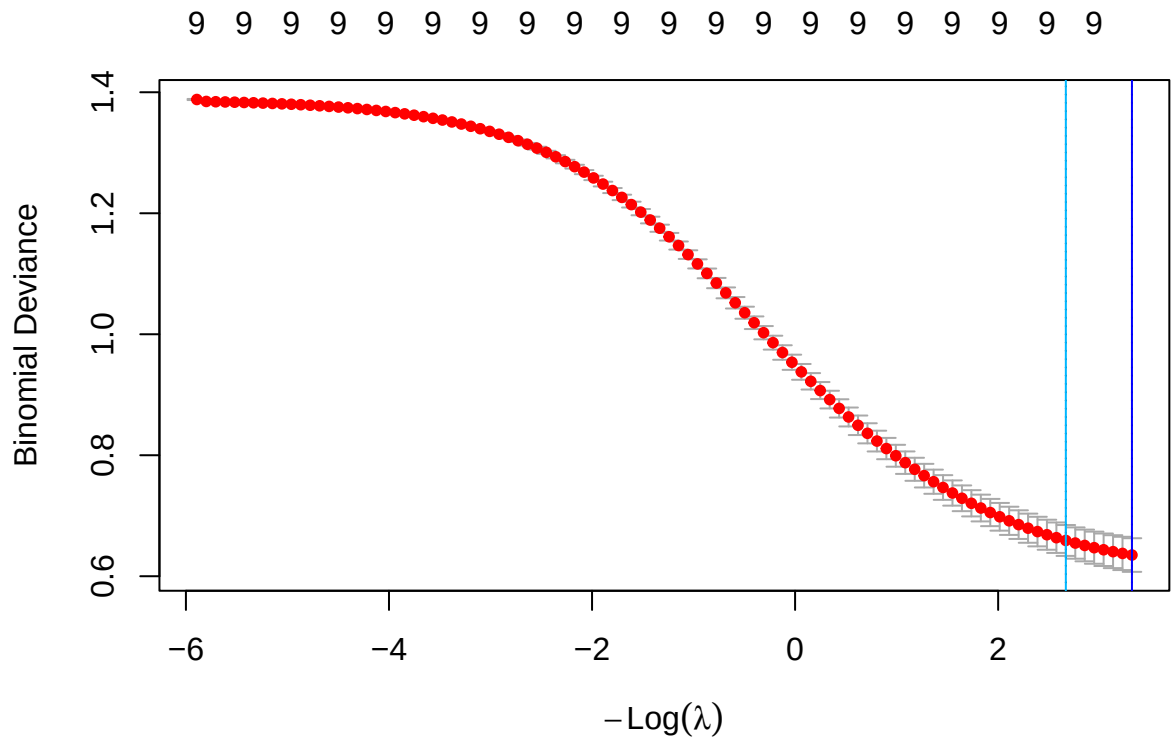
```
## [1] 0.03623211
```

```
cv_ridge$lambda.1se
```

```
## [1] 0.06948994
```

Affichage du modèle RIDGE :

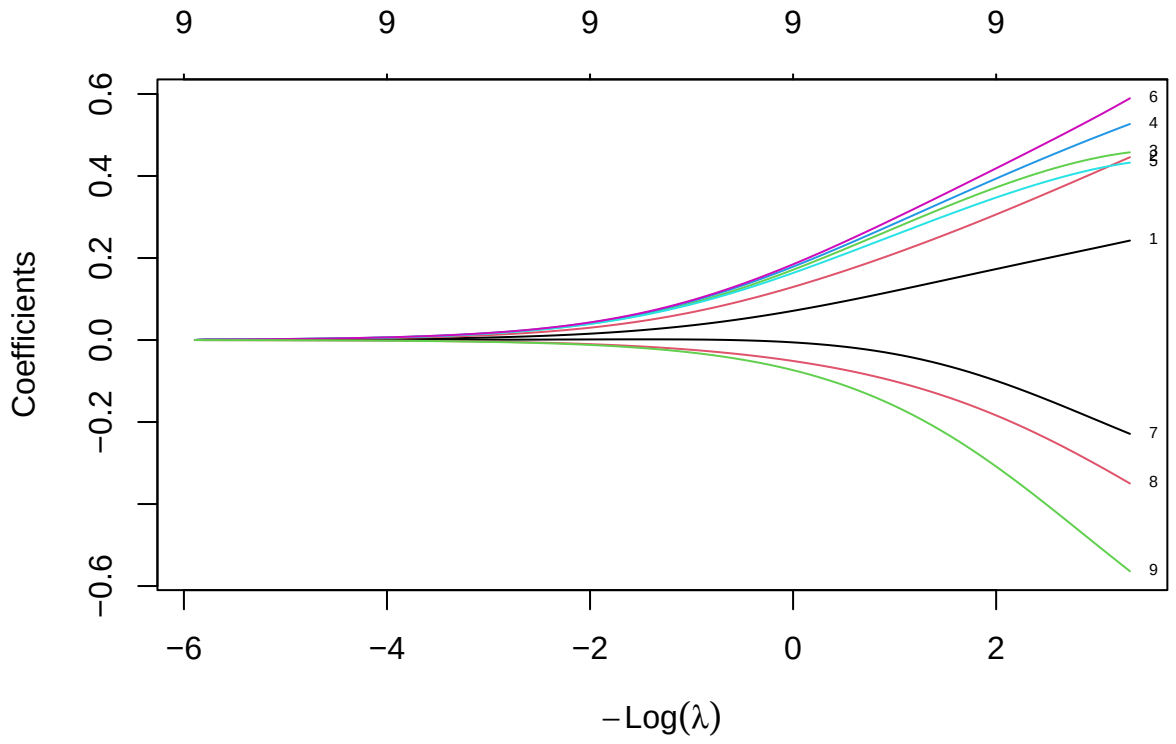
```
plot(cv_ridge)
abline(v = -log(cv_ridge$lambda.min), col = "blue")
abline(v = -log(cv_ridge$lambda.1se), col = "#00b7ff")
```



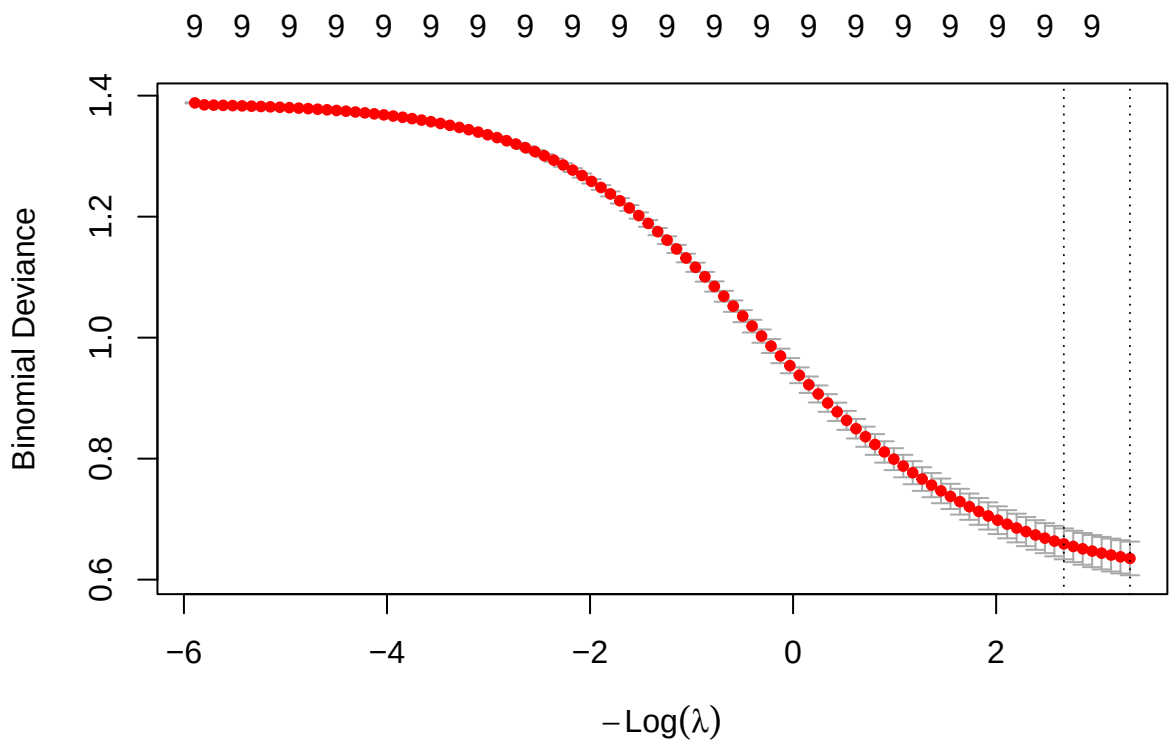
Modèles RIDGE avec le λ optimal :

```
ridge_min <- glmnet(
  x_train_s,
  y_train_matrix,
  alpha = 0,
  family = "binomial",
  lambda = cv_ridge$lambda.min
)
```

```r
# Shows coefficient shrinkage across many lambdas
plot(rm, xvar = "lambda", label = TRUE)
```

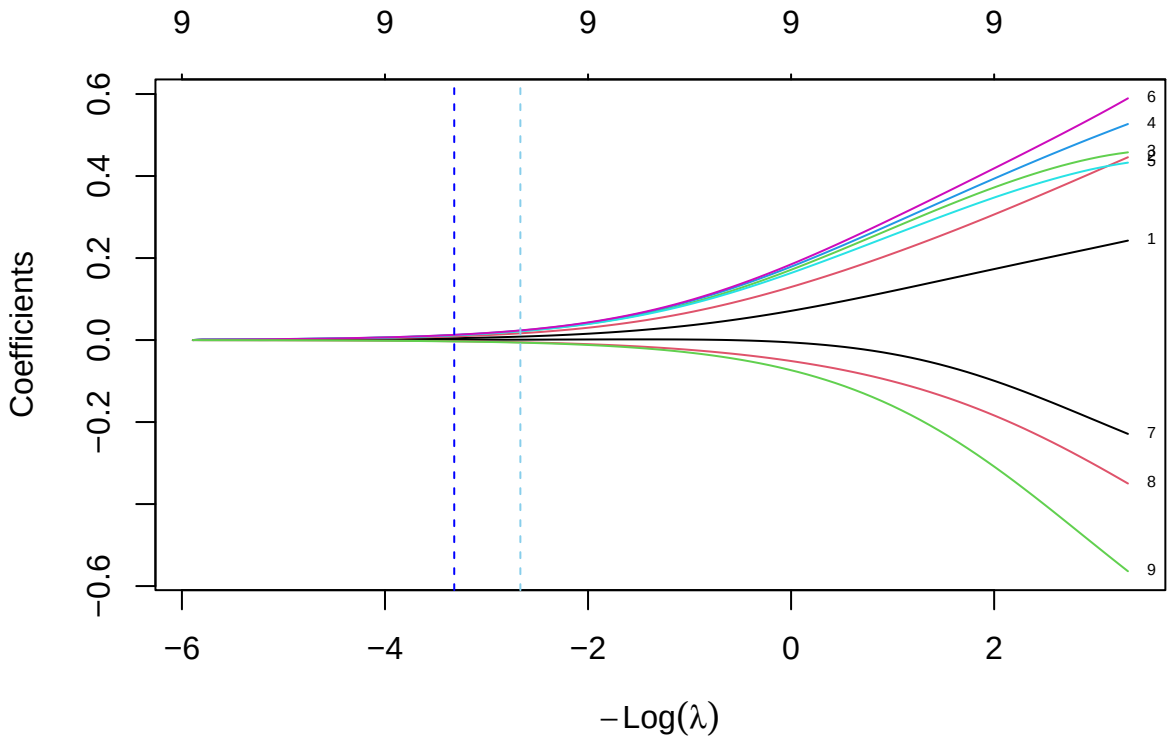



```r
# Shows cross-validation curve with optimal lambda markers
plot(cv_ridge)
```

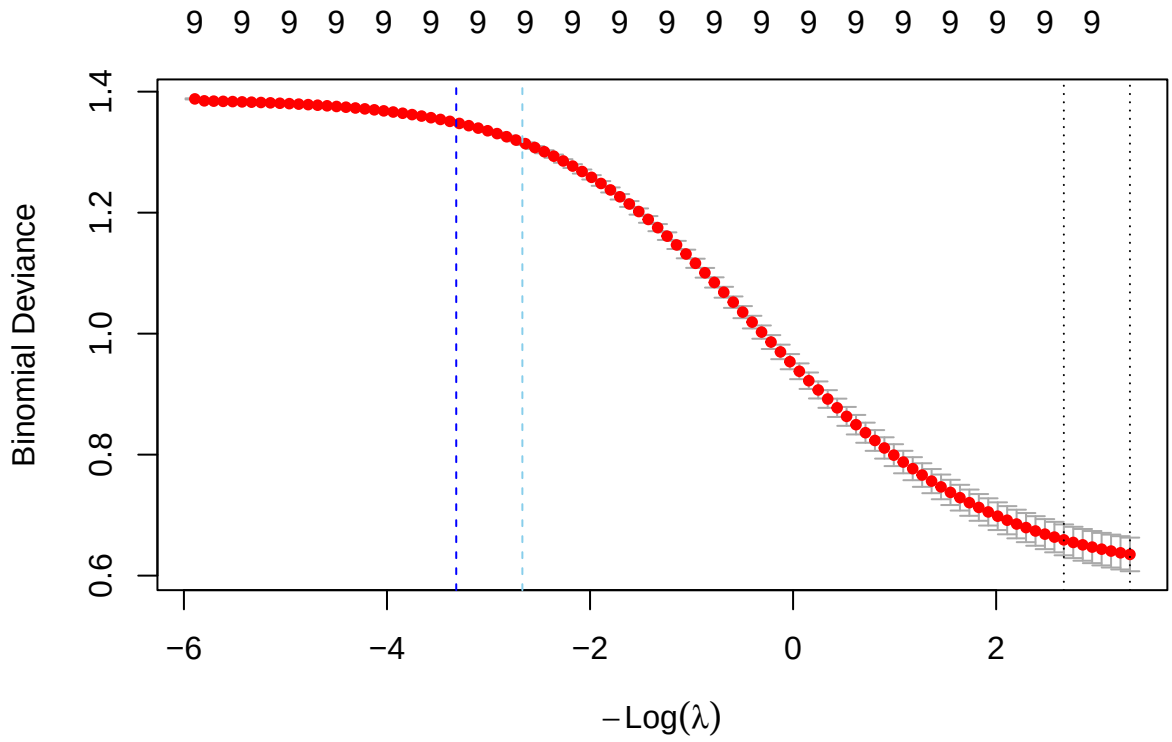



```r
ridge_1se <- glmnet(
  x_train_s,
  y_train_matrix,
  alpha = 0,
  family = "binomial",
  lambda = cv_ridge$lambda.1se
)

# Coefficient shrinkage paths
plot(rm, xvar = "lambda", label = TRUE)
abline(v = log(cv_ridge$lambda.min), col = "blue", lty = 2)
abline(v = log(cv_ridge$lambda.1se), col = "skyblue", lty = 2)
```




```
# CV curve (CV error vs lambda)
plot(cv_ridge)
abline(v = log(cv_ridge$lambda.min), col = "blue", lty = 2)
abline(v = log(cv_ridge$lambda.1se), col = "skyblue", lty = 2)
```



Calcul des variables les plus importantes pour le minimum :

```
coef_ridge_min <- coef(ridge_min, s = cv_ridge$lambda.min)
coef_ridge_min_vec <- as.numeric(coef_ridge_min)
names(coef_ridge_min_vec) <- rownames(coef_ridge_min)
important_vars_ridge_min <- sort(abs(coef_ridge_min_vec), decreasing = TRUE)
head(important_vars_ridge_min, 5)
```

```
##    T2Mmin       DOW       T2M      STRD      SSRD
## 0.5894233 0.5638151 0.5268778 0.4577130 0.4458363
```

Détermination des variables les plus importantes pour le 1 standard error :

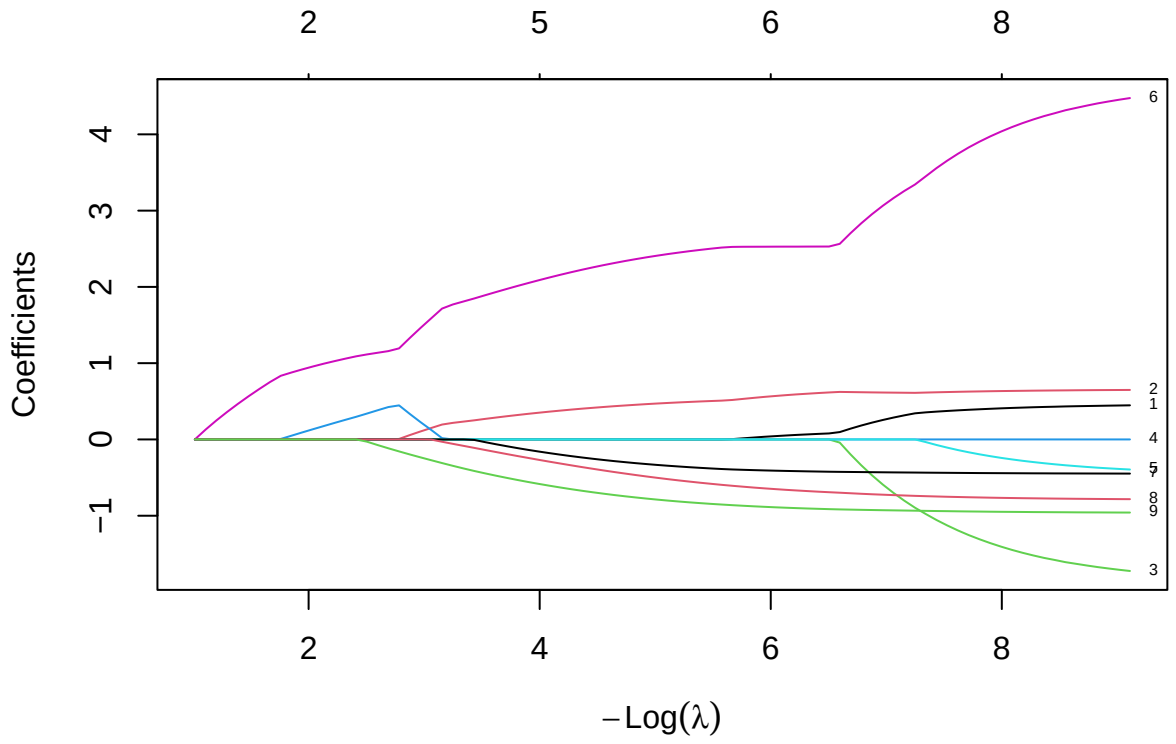
```
coef_ridge_1se <- coef(ridge_1se, s = cv_ridge$lambda.1se)
coef_ridge_1se_vec <- as.numeric(coef_ridge_1se)
names(coef_ridge_1se_vec) <- rownames(coef_ridge_1se)
important_vars_ridge_1se <- sort(abs(coef_ridge_1se_vec), decreasing = TRUE)
head(important_vars_ridge_1se, 5)
```

```
##    T2Mmin       T2M       DOW      STRD    T2Mmax
## 0.5025708 0.4636200 0.4348805 0.4252026 0.3986176
```

Conclusion : les variables les plus importantes sont T2Min, T2M, STRD, T2Max, SSRD. On obtient le même résultat avec les deux λ optimaux avec seul un ordre différent.

3.5 Régression logistique LASSO

```
lam <- glmnet(x_train_s, y_train, alpha = 1, family = "binomial")
plot(lam, xvar = "lambda", label = TRUE)
```




```
cv_lasso <- cv.glmnet(
  x_train_s,
  y_train,
  alpha = 1,
  family = "binomial",
  grouped = FALSE,
  nfolds = 10
)
```

Affichage du meilleur lambda :

```
cv_lasso$lambda.min
```

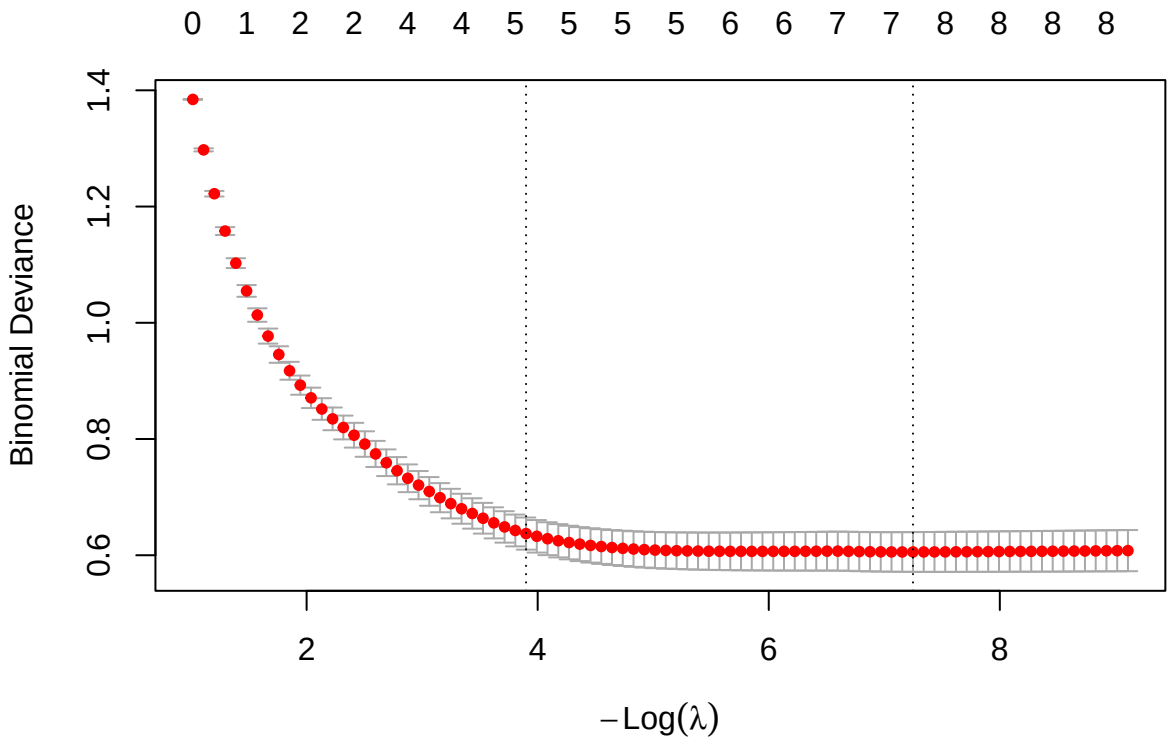
```
## [1] 0.0007112511
```

Affichage du meilleur lambda 1se :

```r
print(cv_lasso$lambda.1se)
```

```
## [1] 0.02025669
```

```r
plot(cv_lasso)
```




```r
lasso_min <- glmnet(
  x_train,
  y_train,
  alpha = 1,
  family = "binomial",
  lambda = cv_lasso$lambda.min
)

lasso_1se <- glmnet(
  x_train,
  y_train,
  alpha = 1,
  family = "binomial",
  lambda = cv_lasso$lambda.1se
)
```

Impact des coefficients pour $\lambda_{\min}$

```r
coef_lasso_min <- coef(lasso_min)
coef_lasso_min_vec <- as.numeric(coef_lasso_min)
names(coef_lasso_min_vec) <- rownames(coef_lasso_min)
important_vars_lasso_min <- sort(abs(coef_lasso_min_vec), decreasing = TRUE)
head(important_vars_lasso_min, 5)
```

```
## (Intercept)    Holidays      T2Mmin         DOW          RH
##  5.60204216  4.31877495  0.79722398  0.46557053  0.03593545
```

Impact des coefficients pour λ_{1se}

```r
coef_lasso_1se <- coef(lasso_1se, s = cv_lasso$lambda.1se)
coef_lasso_1se_vec <- as.numeric(coef_lasso_1se)
names(coef_lasso_1se_vec) <- rownames(coef_lasso_1se)
important_vars_lasso_1se <- sort(abs(coef_lasso_1se_vec), decreasing = TRUE)
head(important_vars_lasso_1se, 5)
```

```
## (Intercept)    Holidays      T2Mmin         DOW       Covid
##  8.44861877  1.41534152  0.48973061  0.27791204  0.00476857
```

4 Conclusion : comparaison des modèles

```r
# On retire la colonne qui n'est pas un chiffre
tab2 <- subset(tab, select = -X0)

K <- 10
n <- nrow(tab2)
folds <- sample(rep(1:K, length.out = n))

acc_step <- numeric(K)
acc_back <- numeric(K)
acc_for  <- numeric(K)
acc_ridge <- numeric(K)
acc_lasso <- numeric(K)

for (i in 1:K) {
  test_idx <- which(folds == i)
  train_idx <- setdiff(seq_len(n), test_idx)

  train_tab <- tab2[train_idx, ]
  test_tab  <- tab2[test_idx, ]

  # On retire la colonne résultat et les colonnes redondantes
  x_train <- as.matrix(train_tab[, -c(1, which(names(tab2) %in% c("Total", "TOY", "y")))])
  x_test  <- as.matrix(test_tab[, -c(1, which(names(tab2) %in% c("Total", "TOY", "y")))])
  x_train_s <- scale(x_train)
  x_test_s  <- scale(x_test, center = attr(x_train_s, "scaled:center"), scale = attr(x_train_s, "scaled
  y_train <- as.numeric(train_tab$y)
  y_test  <- as.numeric(test_tab$y)
```

```r
  # Calcul des modèles à partir des données test et train
  res <- glm(y ~ . - TOY - Total, data = train_tab, family = binomial)

  resstep <- step(res, direction = "both", trace = 0)
  resback <- step(res, direction = "backward", trace = 0)
  resfor  <- step(res, direction = "forward", trace = 0)

  # Tentative de prédiction
  pred_step <- predict(resstep, newdata = test_tab, type = "response")
  pred_back <- predict(resback, newdata = test_tab, type = "response")
  pred_for  <- predict(resfor,  newdata = test_tab, type = "response")

  # Calcul de la précision
  acc_step[i] <- mean((pred_step > 0.5) == y_test)
  acc_back[i] <- mean((pred_back > 0.5) == y_test)
  acc_for[i]  <- mean((pred_for  > 0.5) == y_test)

  # Et même chose pour RIDGE et LASSO
  cv_ridge <- cv.glmnet(x_train_s, y_train, alpha = 0, family = "binomial", grouped = FALSE)
  ridge_min <- glmnet(x_train_s, y_train, alpha = 0, family = "binomial", lambda = cv_ridge$lambda.min)
  ridge_pred <- predict(ridge_min, newx = x_test_s, type = "response")
  acc_ridge[i] <- mean((ridge_pred > 0.5) == y_test)

  cv_lasso <- cv.glmnet(x_train_s, y_train, alpha = 1, family = "binomial", grouped = FALSE)
  lasso_min <- glmnet(x_train_s, y_train, alpha = 1, family = "binomial", lambda = cv_lasso$lambda.min)
  lasso_pred <- predict(lasso_min, newx = x_test_s, type = "response")
  acc_lasso[i] <- mean((lasso_pred > 0.5) == y_test)
}

# Affichage des résultats
cat("Stepwise - précision moyenne :", round(mean(acc_step), 3),
    "| erreur moyenne :", round(1 - mean(acc_step), 3), "\n")
```

```
## Stepwise - précision moyenne : 0.86 | erreur moyenne : 0.14
```

```r
cat("Backward - précision moyenne :", round(mean(acc_back), 3),
    "| erreur moyenne :", round(1 - mean(acc_back), 3), "\n")
```

```
## Backward - précision moyenne : 0.86 | erreur moyenne : 0.14
```

```r
cat("Forward - précision moyenne :", round(mean(acc_for), 3),
    "| erreur moyenne :", round(1 - mean(acc_for), 3), "\n")
```

```
## Forward - précision moyenne : 0.862 | erreur moyenne : 0.138
```

```r
cat("Ridge lambda_min - précision moyenne :", round(mean(acc_ridge), 3),
    "| erreur moyenne :", round(1 - mean(acc_ridge), 3), "\n")
```

```
## Ridge lambda_min - précision moyenne : 0.863 | erreur moyenne : 0.137
```

```r
cat("LASSO lambda_min - précision moyenne :", round(mean(acc_lasso), 3),
    "| erreur moyenne :", round(1 - mean(acc_lasso), 3), "\n")
```

```
## LASSO lambda_min - précision moyenne : 0.858 | erreur moyenne : 0.142
```