Rapport de TP - Modèles de Régression Régularisée

Tommy MORALES, Aslan MARTIN

2025-11-09

Table des matières

1 Introduction

Ce rapport, réalisé dans le cadre du cours de Modèles de Régression Régularisée, se concentre sur l'analyse d'un jeu de données concernant la production d'électricité au Mexique. L'objectif est de construire un modèle de régression logistique permettant d'expliquer et de prédire la production énergétique journalière en fonction de variables météorologiques et calendaires. Une attention particulière sera portée à l'identification et au traitement des problèmes de multicolinéarité afin de garantir la robustesse et l'interprétabilité du modèle final. Dans le cadre de cette analyse, nous posons l'hypothèse que la production électrique journalière est équivalente de la demande. Cette simplification est nécessaire pour l'interprétation des résultats, et elle est aussi cohérente: Une centrale ne produit que ce qu'elle arrivera à vendre, et le stockage d'énergie coûte beaucoup d'argent.

2 Amélioration des variables

2.1 Formalisation du problème et corrélations initiales du dataframe

```
cor <- cor(df)
corrplot(cor)
```

On repère immédiatement que TOY et DOW sont sous-exploitées: Leurs corrélations à *Total* est très faible. Il semble pourtant intuitif que ces variables introduisent des informations exploitables. Par exemple, la production en été devrait être remarquablement plus haute que celle en hiver du fait de la climatisation.

```
# On divise le dataframe en 2, selon le cas > et < à la médiane
med <- median(df$Total)
df$Total <- df$Total > med
plot(df$Total==TRUE)
```

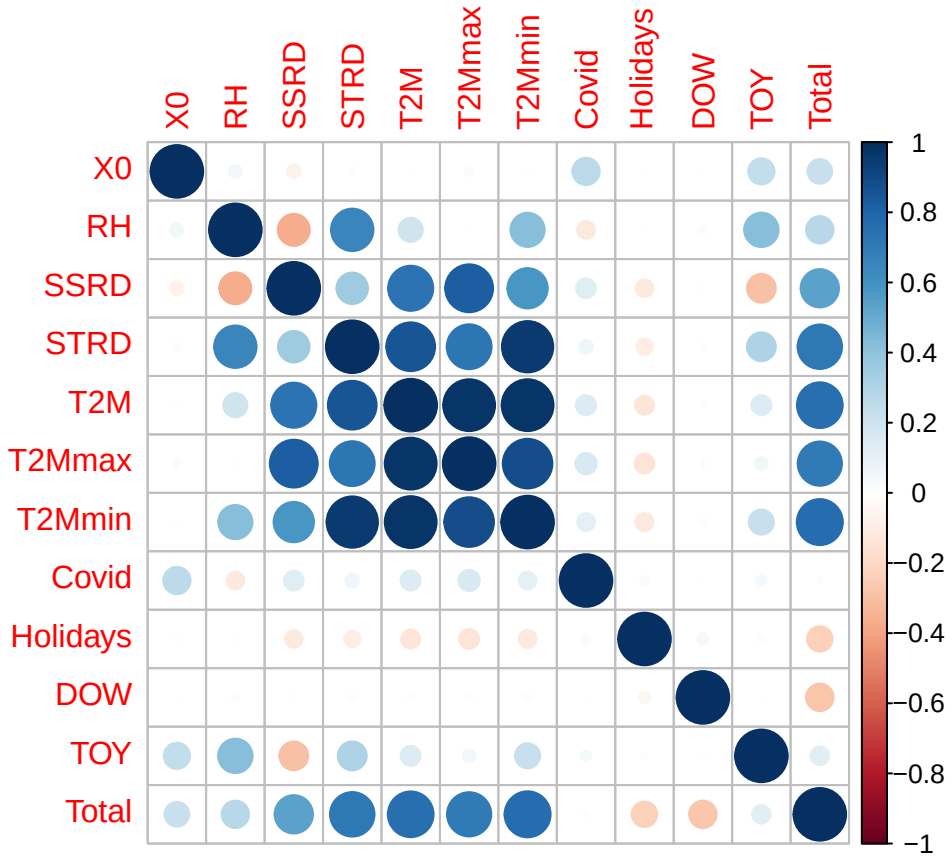
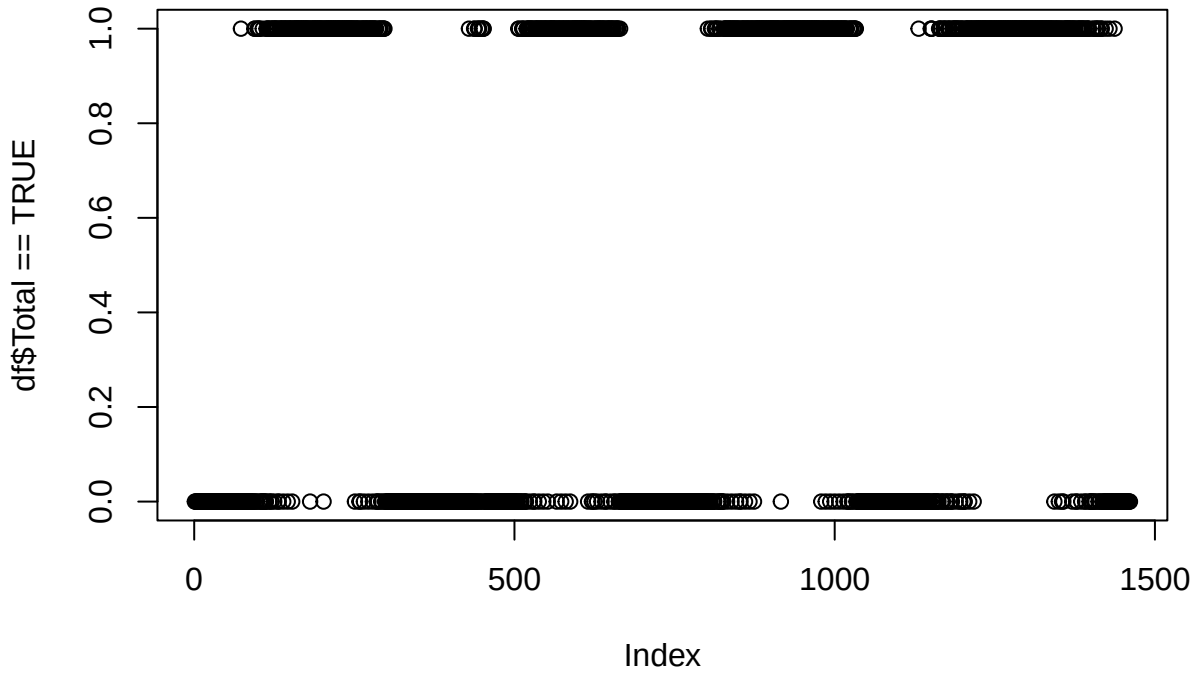


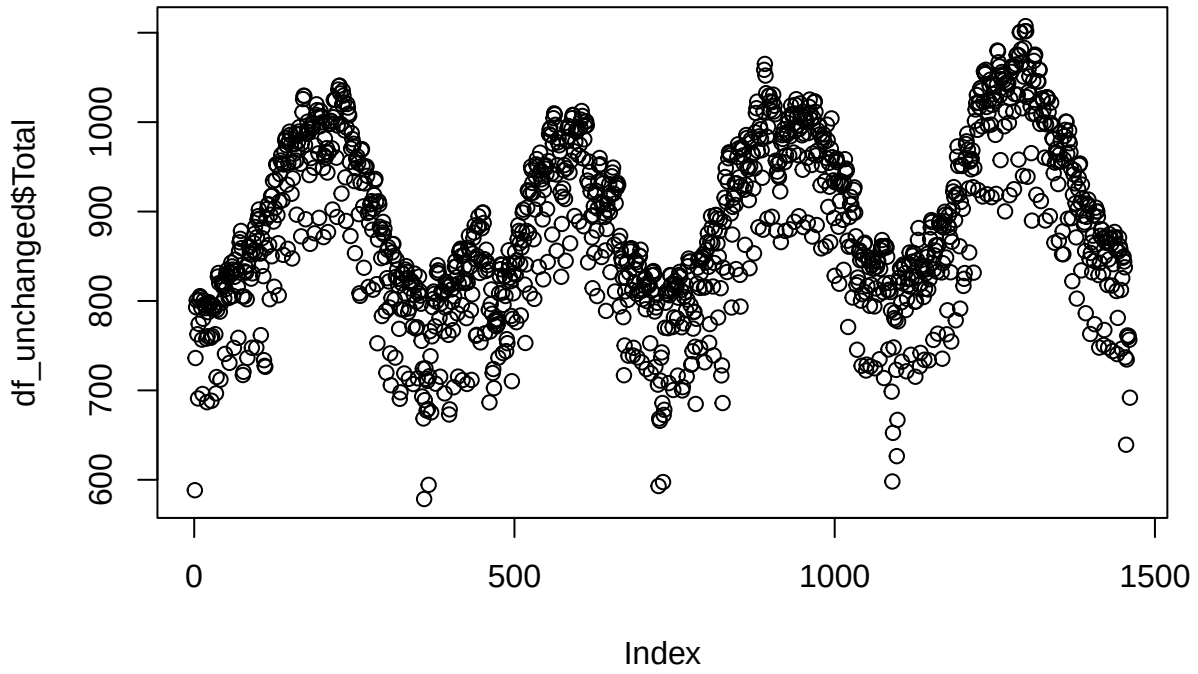
FIGURE 1 – Corrplot initial.



Un patterne périodique semble apparaître. Cela semble suggérer une très forte autocorrélation de *Total*.

2.1.1 Variables cycliques

```
plot(df_unchanged$Total)
```



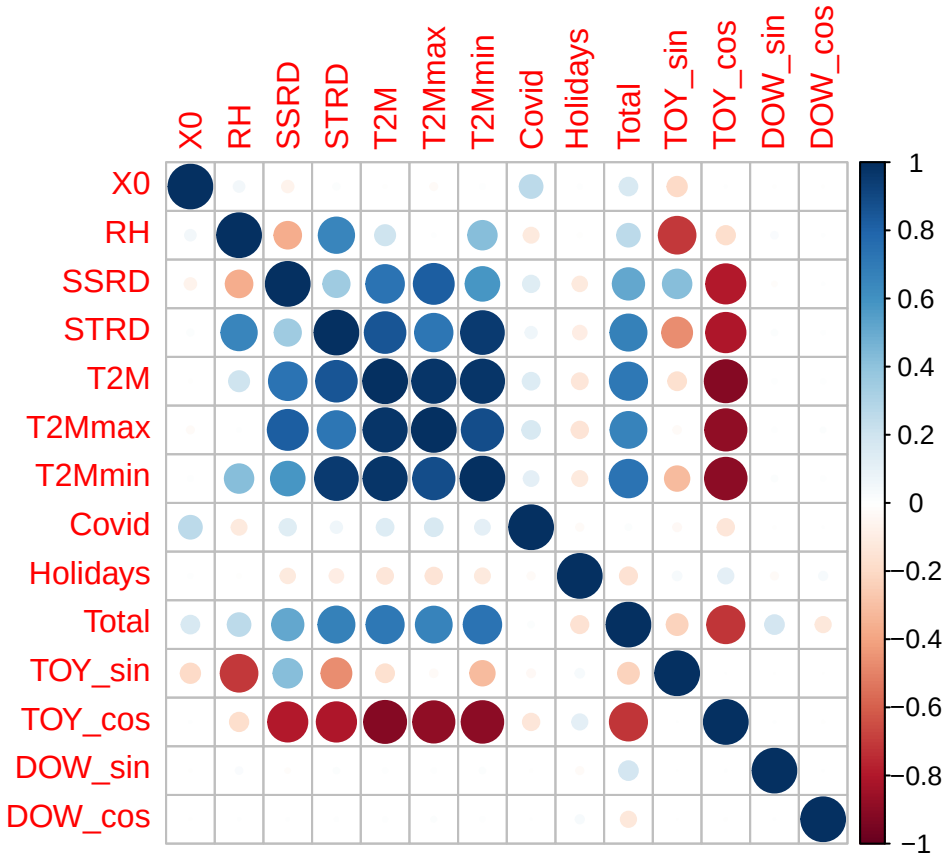
Une allure sinusoïdale de *Total* selon le temps apparait. Cependant, *DOW* et *TOY* sont discontinus, ce qui empêche un modèle linéaire en ses coefficients de représenter adéquatement leur lien avec *Total*. Pour cette raison, nous allons capturer cette relation non-linéaire par le biais de transformations trigonométriques.

```
df$TOY_sin <- sin(2*3.1415*df$TOY/365) # La période du signal est de 1 an
df$TOY_cos <- cos(2*3.1415*df$TOY/365)
```

```r
df$TOY <- NULL

df$DOW_sin <- sin(2*3.1415*df$DOW/7)
df$DOW_cos <- cos(2*3.1415*df$DOW/7)
df$DOW <- NULL

cor <- cor(df)
corrplot(cor)
```

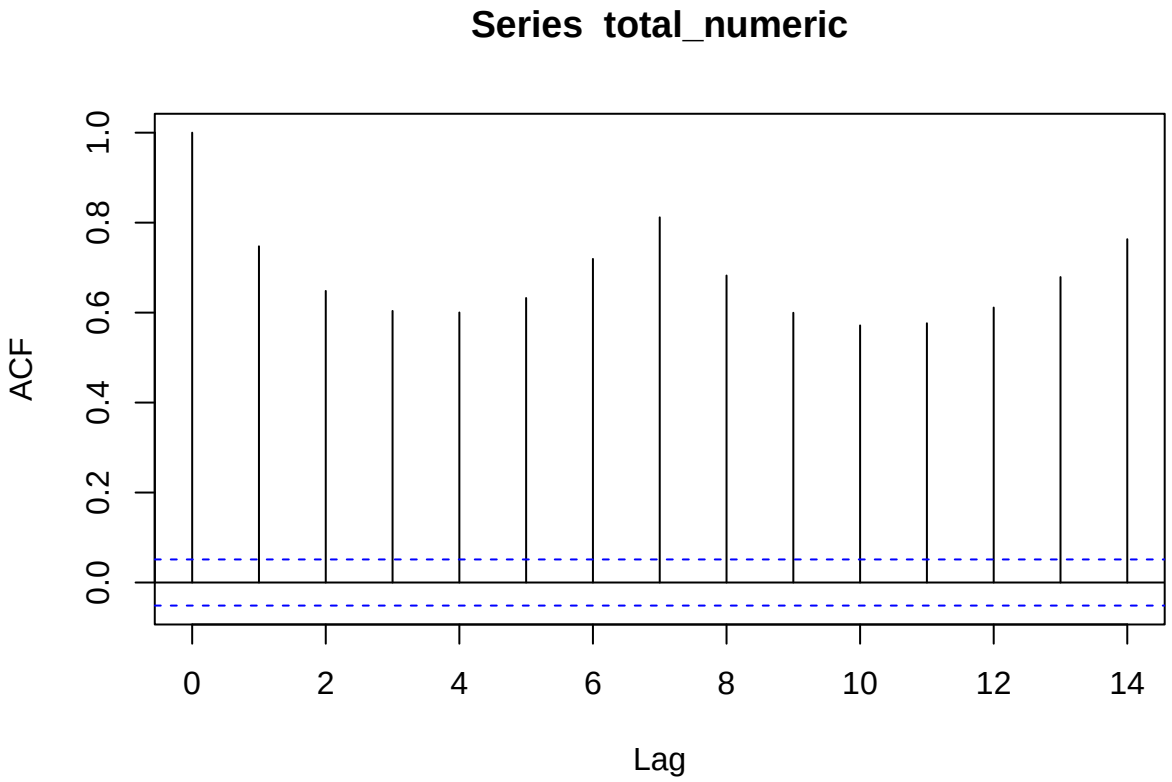


Comme attendu: La corrélation avec TOY devient évidente. Notre dataframe est maintenant nettement plus exploitable pour des modèles régressifs.

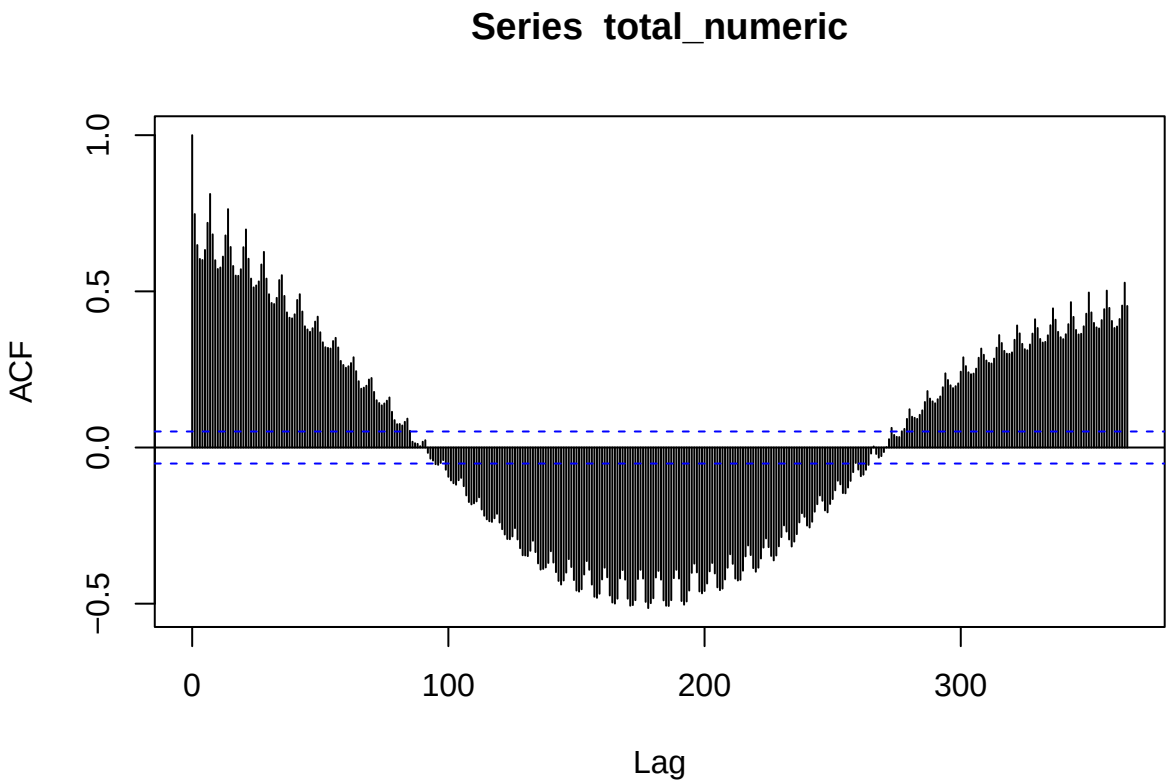
2.1.2 Autocorrélation

Pour démontrer et évaluer l'aspect autocorrelé de *Total*, nous allons plotter l'ACF de *Total*.

```r
total_numeric <- as.numeric(df$Total)
acf(total_numeric, lag.max = 14)
```




```
acf(total_numeric, lag.max = 365)
```



On observe une allure de signal modulé:

— Une enveloppe de pseudo-période de un an

— Une porteuse de période une semaine

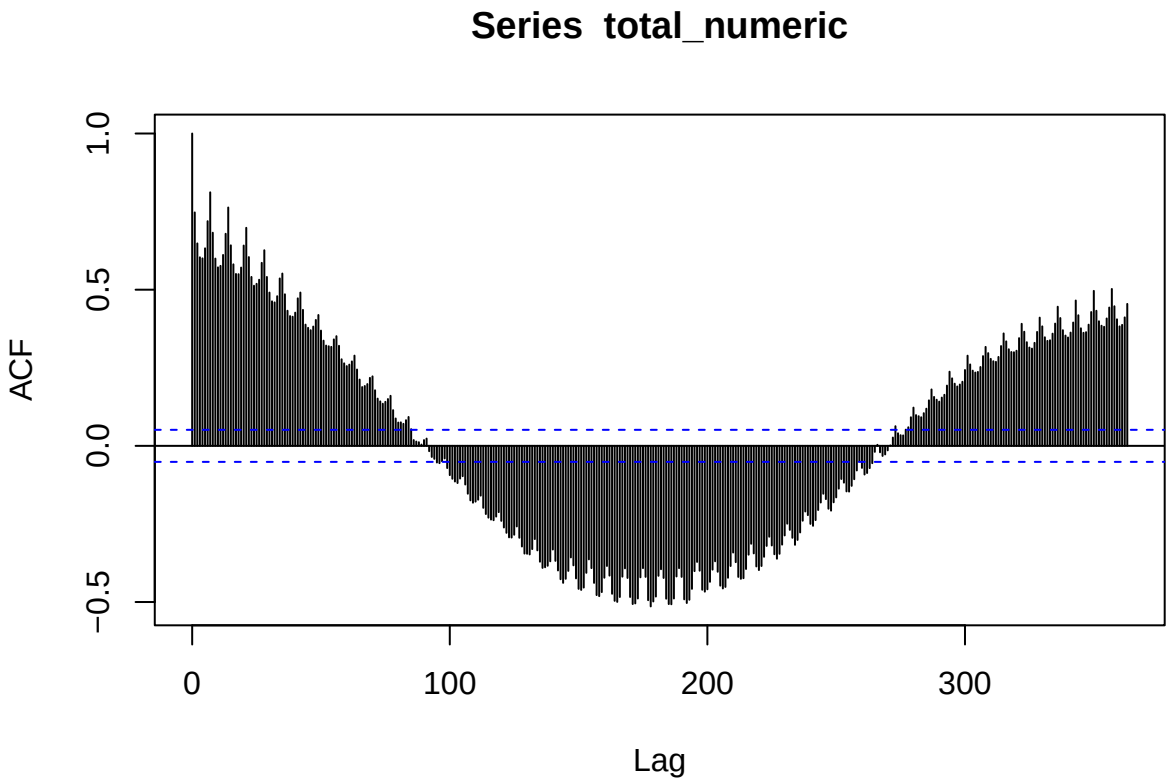
Cette observation est parfaitement attendue: Les besoins en énergie sont sensiblement similaires chaque année pour une même période, et nos besoins en énergie dépendent du jour de la semaine (Par exemple, week-end ou non).

Actuellement, l'aspect périodique à l'année est déjà pris en compte via TOY.

Cependant, l'influence des jours précédents ne l'est pas, donc nous devons rajouter des variables d'autocrrelation récente $Total_{n-k}$ à nos variables.

De plus, bien que DOW capture la périodicité hebdomadaire, l'ACF révèle que $Total_{n-7k}$ est plus influant que $Total_{n-7(k-1)-i}$ pour k faible et $i \in [0, 6]$. En d'autres termes, si nous sommes un lundi, alors le lundi qui le précède nous donne nettement plus d'information qu'un lundi quelconque, ou qu'un jour de la semaine précédente. En moyenne, $|cor(Total_{n-7}, Total_n)| > |cor(Total_{n-1}, Total_n)|$, et pour une même saison (voir graphe suivant: Cette relation n'est plus vérifiée au delà d'une saison), $|cor(Total_{n-7k}, Total_n)| > |cor(Total_{n-7p}, Total_n)| \implies k < p$

```
# Pour démontrer la dépendance saisonnale de la relation, nous plottons la somme cummulée
# inverse de l'acf.
# Ce graphique nous montrera les 'lags' les plus utiles pour interpréter Total.
acf_ini <- acf(total_numeric, lag.max = 363)
```

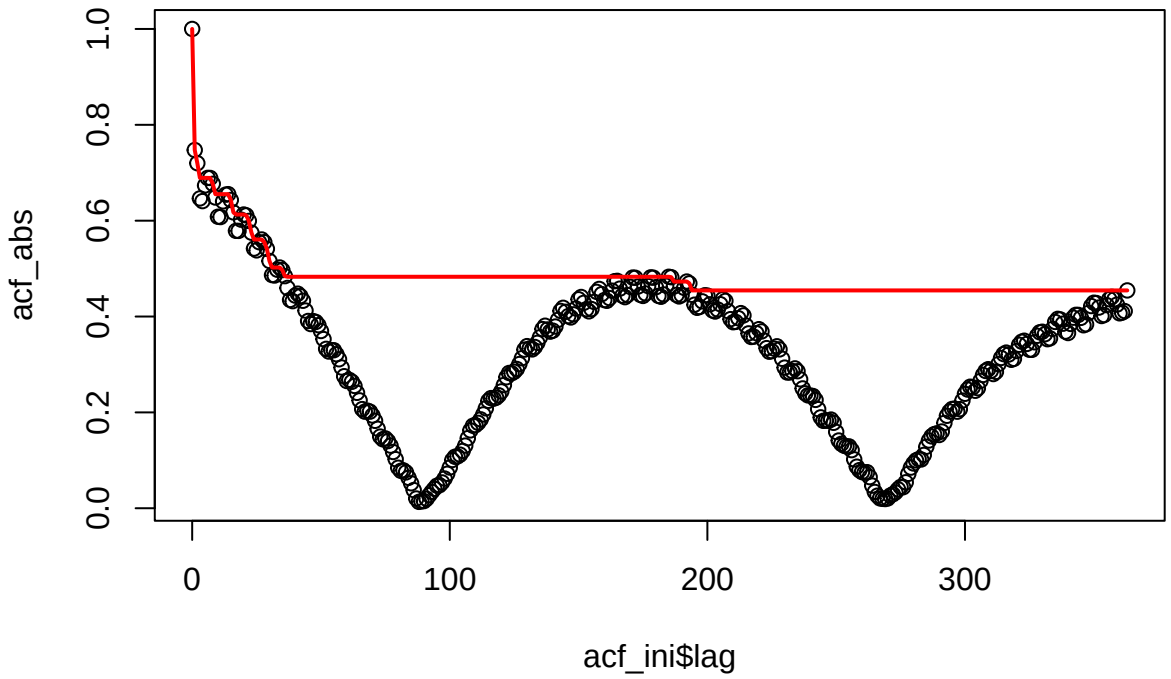



```
acf_abs_ini <- abs(acf_ini$acf)
acf_abs_ini[365/2]
```

```
## [1] 0.4161587
```

```
acf_abs <- rollmean(abs(acf_abs_ini), k=5, , fill = NA) # Nous sommes obligé de prendre
#une moyenne glissante, sans quoi acf_abs considérait exclusivement des outliers
na_indices <- is.na(acf_abs) # Toutes premières 4 valeurs
acf_abs[na_indices] <- acf_abs_ini[na_indices] # On redonne les valeurs initiales aux 4
# premières valeurs
```

```r
maximum_cumule_acf <- rev(cummax(rev(acf_abs)))
plot(acf_ini$lag, acf_abs)
lines(acf_ini$lag, maximum_cumule_acf, col = "red", lwd = 2)
```



Nous devons donc également ajouter $Total_{n-7k}$ pour quelques k à notre dataframe.

Comme annoncé précédemment, on repère une périodicité de la valeur absolue de la corrélation toutes les 2 saisons (Hiver↔Eté ou Automne↔Printemps, et Saison_i↔Saison_i).

Cependant, le signal étant amorti, l'acf absolue démontre que l'information de ces périodes ne mérite pas l'ajout d'une nouvelle variable autocorréllée (Par exemple, $cor(Total, Total_{n-7}) \approx 0.81$, $cor(Total, Total_{n-1}) \approx 0.78$, mais $cor(Total, Total_{n-365/2}) \approx 0.42$). En effet, hors-réchauffement climatique et évènement mondial majeur (ex: épidémie mondiale majeure), $X_{n-(365/2)k}$ n'ajoute pas d'information qui ne soit portée par TOY_cos et TOY_sin.

Enfin, de manière intuitive, la routine dépend de la saison. Si lors d'une saison, le comportement du climat est nettement plus imprévisible, alors la routine hebdomadaire viendrait à être brisée, et $Total_{n-7}$ ne serait plus un bon prédicteur. Nous tenterons donc de confronter nos variables hebdomadaires (ou routinières) à nos variables annuelles.

2.1.3 Création des variables autorégressives

Suite à l'analyse de l'ACF, nous enrichissons notre dataframe avec des variables autorégressives pour capturer les dépendances temporelles. Le code ci-dessous met en oeuvre la création de lags journaliers (jusqu'à 6 jours) et hebdomadaires (7 et 14 jours).

```r
# Ajout manuel du premier lag
idx <- c(NA, 1:(nrow(df)-1))
df$Total_prec <- df$Total[idx]
df$Total_prec[1] <- 0.5

# Fonction pour automatiser la création des lags consécutifs
autoregression_k <- function (dafr, rang) {
  for (rank in 2:rang){
```

```
    colonne <- paste("Total_prec_", rank, sep = "")
    idx <- c(rep(x=NA, times=rank), 1:(nrow(df)-rank))
    dafr[[colonne]] <- df$Total[idx]
  }
  dafr
}

# Fonction pour ajouter des lags spécifiques (hebdomadaires)
autoregression_k_fix <- function (dafr, rang) {
  colonne <- paste("Total_prec_", rang, sep = "")
  idx <- c(rep(x=NA, times=rang), 1:(nrow(df)-rang))
  dafr[[colonne]] <- df$Total[idx]
  dafr
}

# Application des fonctions pour construire le dataframe enrichi
df_final <- autoregression_k(df, 6)
df_final <- autoregression_k_fix(df_final, 7)
df_final <- autoregression_k_fix(df_final, 14)
df_final <- na.omit(df_final)
```

La création de ces variables décalées introduit des valeurs manquantes (NA) au début du jeu de données. Cela nous force à supprimer ces lignes, afin d'éviter d'appauvrir le modèle en leur imposant des valeurs aléatoires.

3 Première Modélisation et Évaluation

Avec notre jeu de données enrichi, nous procédons à la construction et à l'évaluation des modèles. La démarche sera incrémentale : nous allons d'abord construire un modèle de référence avec les lags journaliers, puis l'améliorer en intégrant les lags hebdomadaires et les interactions pour valider notre hypothèse. Enfin, nous comparerons ces approches à un modèle régularisé par Lasso. Pour chaque étape, les données sont séparées en un ensemble d'entraînement (70%) et de test (30%). On ne peut pas ce permettre de prendre des lignes aléatoirement du fait de la dépendance autocorrellée (Il serait absurde d'accéder à des données futures).

```
# On définit la fonction de séparation des données ici pour l'utiliser par la suite.
set_split <- function(dafr, percent){
  train <- floor(percent/100 * nrow(dafr))

  list(train = dafr[1:train, ], test = dafr[(train+1):nrow(dafr), ])
}
```

3.1 Modèle 1 : Régression Stepwise avec lags journaliers

Notre premier modèle sert de référence. Il intègre toutes les variables météorologiques et calendaires, ainsi que les lags autorégressifs des 6 jours précédents, pour capturer la dynamique à court terme. La fonction *step* va sélectionner itérativement le meilleur sous-ensemble de variables en se basant sur l'AIC.

```
# Dataframe avec lags journaliers
df_6 <- na.omit(autoregression_k(df, 6))

# Séparation des données
df_6_sample <- set_split(df_6, 70)
df_6_train <- df_6_sample$train
df_6_test  <- df_6_sample$test
```

```r
# Construction du modèle GLM et sélection stepwise
resall <- glm(Total~., data=df_6_train, family=binomial)
resstep <- step(resall, direction='both', trace = 0)
print(resstep)
```

```
##
## Call:  glm(formula = Total ~ X0 + RH + SSRD + STRD + T2Mmin + Covid +
##     Holidays + DOW_sin + DOW_cos + Total_prec_2 + Total_prec_3 +
##     Total_prec_4 + Total_prec_5 + Total_prec_6, family = binomial,
##     data = df_6_train)
##
## Coefficients:
##      (Intercept)                X0                RH              SSRD
##       -6.398e+01         4.030e-03         9.441e-02         4.114e-06
##             STRD            T2Mmin             Covid          Holidays
##       -3.939e-05         1.623e+00        -4.159e-02        -5.744e+00
##          DOW_sin           DOW_cos  Total_prec_2TRUE  Total_prec_3TRUE
##        4.610e+00        -3.903e+00         8.765e-01         2.794e+00
## Total_prec_4TRUE  Total_prec_5TRUE  Total_prec_6TRUE
##        2.984e+00         1.659e+00         9.166e-01
##
## Degrees of Freedom: 1017 Total (i.e. Null);  1003 Residual
## Null Deviance:       1411
## Residual Deviance: 290.3     AIC: 320.3
```

```r
# Évaluation du modèle de référence
predictions_prob <- predict(resstep, newdata = df_6_test, type = "response")
predictions_class <- predictions_prob > 0.5
matrice_confusion <- table(Observé = df_6_test$Total, Prédit = predictions_class)
accuracy_6 <- sum(diag(matrice_confusion)) / sum(matrice_confusion)

print(matrice_confusion)
```

```
##        Prédit
## Observé FALSE TRUE
##   FALSE   175   31
##   TRUE      8  223
```

```r
print(paste("Accuracy (Modèle de référence - lags journaliers):", accuracy_6))
```

```
## [1] "Accuracy (Modèle de référence - lags journaliers): 0.910755148741419"
```

Ce premier modèle atteint déjà une précision élevée, confirmant la forte prédictibilité du signal à court terme.

3.2 Modèle 2 : Amélioration par ajout des lags hebdomadaires

Nous testons maintenant notre hypothèse selon laquelle les lags hebdomadaires apportent une information supplémentaire pertinente. Nous reprenons la démarche précédente sur le dataframe le plus complet.

```r
# Dataframe enrichi avec les lags hebdomadaires
df_6_2 <- autoregression_k(df, 6)
df_6_2 <- autoregression_k_fix(df_6_2, 7)
df_6_2 <- na.omit(autoregression_k_fix(df_6_2, 14))

# Séparation des données
df_6_sample_2 <- set_split(df_6_2, 70)
```

```r
df_6_2_train <- df_6_sample_2$train
df_6_2_test  <- df_6_sample_2$test

# Construction du modèle GLM complexe avec sélection stepwise
resall <- glm(df_6_2_train$Total~., data=df_6_2_train, family=binomial)
resstep <- step(resall, direction='both', trace = 0)
print(resstep)
```

```
##
## Call:  glm(formula = df_6_2_train$Total ~ X0 + RH + STRD + T2M + T2Mmax +
##     Covid + Holidays + TOY_sin + DOW_sin + DOW_cos + Total_prec_3 +
##     Total_prec_4 + Total_prec_5 + Total_prec_7 + Total_prec_14,
##     family = binomial, data = df_6_2_train)
##
## Coefficients:
##       (Intercept)                X0                RH              STRD
##         -6.317e+01          4.295e-03          1.807e-01         -5.011e-05
##               T2M             T2Mmax             Covid          Holidays
##          3.216e+00         -1.302e+00         -4.346e-02         -5.772e+00
##           TOY_sin            DOW_sin            DOW_cos   Total_prec_3TRUE
##          6.438e-01          3.325e+00         -2.813e+00          2.027e+00
##  Total_prec_4TRUE   Total_prec_5TRUE   Total_prec_7TRUE  Total_prec_14TRUE
##          2.304e+00          1.322e+00          1.871e+00          1.099e+00
##
## Degrees of Freedom: 1011 Total (i.e. Null);  996 Residual
## Null Deviance:       1403
## Residual Deviance: 257.4     AIC: 289.4
```

On observe que le modèle préserve des variables très corréllées (Ex: T2Mmax et T2M). Donc la méthode *step* est nécessairement moins interprétable qu'une régression Lasso.

```r
# Évaluation du modèle amélioré
predictions_prob <- predict(resstep, newdata = df_6_2_test, type = "response")
predictions_class <- predictions_prob > 0.5
matrice_confusion <- table(Observé = df_6_2_test$Total, Prédit = predictions_class)
accuracy <- sum(diag(matrice_confusion)) / sum(matrice_confusion)

print(matrice_confusion)
```

```
##        Prédit
## Observé FALSE TRUE
##   FALSE   172   34
##   TRUE      8  221
```

```r
print(paste("Accuracy (Modèle amélioré - lags hebdos):", accuracy))
```

```
## [1] "Accuracy (Modèle amélioré - lags hebdos): 0.903448275862069"
```

La comparaison des précisions (0.903 contre 0.911) met en doûte notre suspicion : l'ajout des variables autorégressives hebdomadaires n'améliore pas nécessairement la performance du modèle. Cependant, il est possible que ce résultat soit justifié par l'aspect 'greedy' de la fonction step. L'ajout de nouvelles variable pourrait avoir introduit de l'overfitting, baissant ainsi nos résultats. Le fait que le modèle ait préservé toutes les variables est un fort indicateur d'overfitting. Pour vérifier ou réfuter notre hypothèse, nous la testerons sur une régression Lasso.

3.3 Modèle 3 : Amélioration par ajout des intéractions saisonales

Nous testons maintenant notre hypothèse selon laquelle les lags hebdomadaires et'intéraction des lags hebdomadaires avec la saisonnalité apportent une information supplémentaire pertinente, ce qui était notre hypothèse la moins certaine (celle-ci se basait exclusivement sur notre intuition). Nous reprenons la démarche précédente sur le dataframe le plus complet.

```
# Dataframe enrichi avec les lags hebdomadaires
df_6_2 <- autoregression_k(df, 6)
df_6_2 <- autoregression_k_fix(df_6_2, 7)
df_6_2 <- na.omit(autoregression_k_fix(df_6_2, 14))

# Séparation des données
df_6_sample_2 <- set_split(df_6_2, 70)
df_6_2_train <- df_6_sample_2$train
df_6_2_test  <- df_6_sample_2$test

# Construction du modèle GLM complexe avec sélection stepwise
resall <- glm(df_6_2_train$Total~X0 + RH + SSRD + STRD + T2M + T2Mmax + T2Mmin + Covid + Holidays +
                TOY_sin * Total_prec_7 * Total_prec_14 + TOY_cos * Total_prec_7 * Total_prec_14 +
                DOW_sin + DOW_cos + Total_prec + Total_prec_2 + Total_prec_3 + Total_prec_4 +
                Total_prec_5 + Total_prec_6,
              data=df_6_2_train, family=binomial)
resstep <- step(resall, direction='both', trace = 0)
print(resstep)
```

```
##
## Call:  glm(formula = df_6_2_train$Total ~ X0 + RH + STRD + T2M + T2Mmax +
##     Covid + Holidays + TOY_sin + Total_prec_7 + Total_prec_14 +
##     TOY_cos + DOW_sin + DOW_cos + Total_prec_3 + Total_prec_4 +
##     Total_prec_5 + Total_prec_6 + TOY_sin:Total_prec_7 + TOY_sin:Total_prec_14 +
##     Total_prec_7:Total_prec_14 + Total_prec_14:TOY_cos, family = binomial,
##     data = df_6_2_train)
##
## Coefficients:
##                          (Intercept)                                   X0
##                           -5.208e+01                            3.971e-03
##                                   RH                                 STRD
##                            1.665e-01                           -5.766e-05
##                                  T2M                                T2Mmax
##                            4.050e+00                           -1.802e+00
##                                Covid                              Holidays
##                           -4.056e-02                           -5.511e+00
##                              TOY_sin                    Total_prec_7TRUE
##                            1.710e+00                            3.525e+00
##                   Total_prec_14TRUE                              TOY_cos
##                            2.055e+00                            2.018e+00
##                              DOW_sin                              DOW_cos
##                            3.207e+00                           -3.212e+00
##                    Total_prec_3TRUE                    Total_prec_4TRUE
##                            2.328e+00                            2.003e+00
##                    Total_prec_5TRUE                    Total_prec_6TRUE
##                            1.190e+00                            1.205e+00
##           TOY_sin:Total_prec_7TRUE           TOY_sin:Total_prec_14TRUE
##                           -1.236e+00                           -8.723e-01
```

```
## Total_prec_7TRUE:Total_prec_14TRUE           Total_prec_14TRUE:TOY_cos
##                          -3.341e+00                           -2.789e+00
##
## Degrees of Freedom: 1011 Total (i.e. Null);  990 Residual
## Null Deviance:       1403
## Residual Deviance: 232.1     AIC: 276.1
```

```
# Évaluation du modèle amélioré
predictions_prob <- predict(resstep, newdata = df_6_2_test, type = "response")
predictions_class <- predictions_prob > 0.5
matrice_confusion <- table(Observé = df_6_2_test$Total, Prédit = predictions_class)
accuracy <- sum(diag(matrice_confusion)) / sum(matrice_confusion)

print(matrice_confusion)
```

```
##        Prédit
## Observé FALSE TRUE
##   FALSE   175   31
##   TRUE     12  217
```

```
print(paste("Accuracy (Modèle amélioré - lags hebdos + interactions):", accuracy))
```

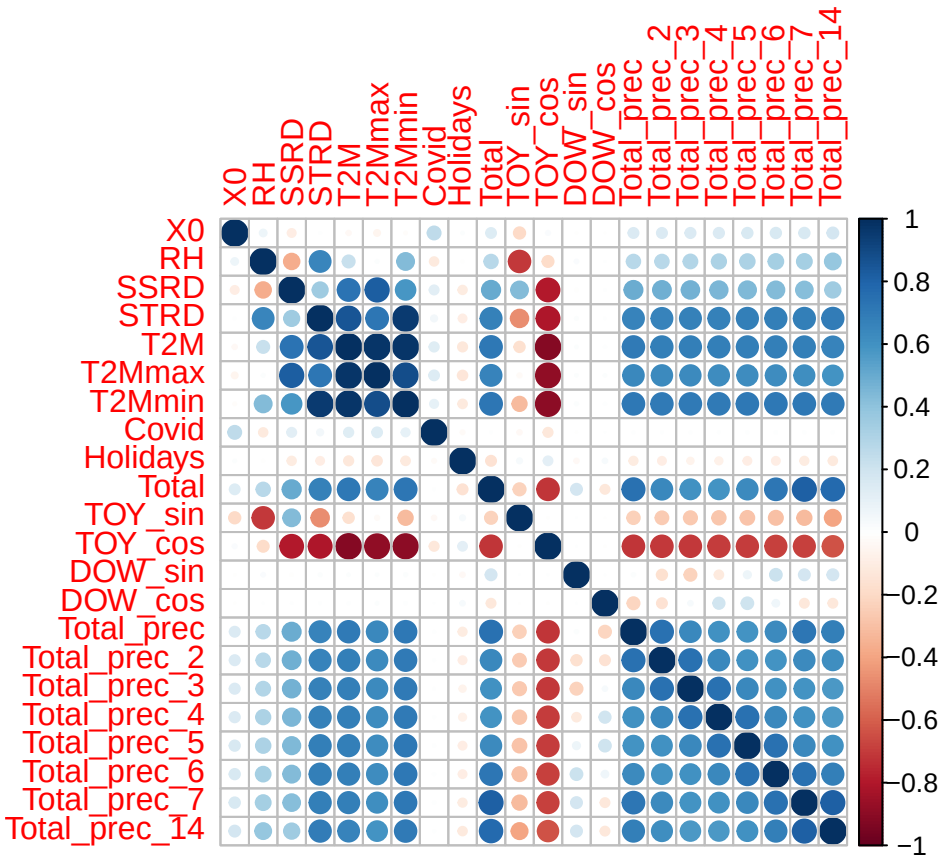
```
## [1] "Accuracy (Modèle amélioré - lags hebdos + interactions): 0.901149425287356"
```

La qualité du modèle est redescendue. Là encore, nous confronteront notre hypothèse à la regression Lasso pour vérifier ou inférer notre hypothèse. Le fait que le modèle ait préservé autant (toutes?) ces variables renforce l'hypothèse d'overfitting.

3.4 Modèle 4 : Sélection par Régression Lasso

Enfin, nous appliquons une régularisation Lasso sur l'ensemble des variables.

```
corrplot(cor(df_6_2))
```



Le corrplot le démontre, de nombreuses variables sont corréllées, ce qui limite l'interprétabilité et la robustesse de notre modèle. (Ex: T2M et T2Mmin ou Total_prec et Total_prec_2)

```
# Utilisation des mêmes données d'entraînement/test que le modèle 2
df_6_2_train <- df_6_sample_2$train
df_6_2_test  <- df_6_sample_2$test

# Préparation des matrices pour glmnet
x_train <- model.matrix(Total ~ ., data = df_6_2_train)
y_train <- df_6_2_train$Total
x_test <- model.matrix(Total ~ ., data = df_6_2_test)
y_test <- df_6_2_test$Total

# Validation croisée et ajustement du modèle Lasso
cv_lasso <- cv.glmnet(x_train, y_train, family = "binomial")

# Évaluation du modèle Lasso
predictions_prob <- predict(cv_lasso, x_test, type = "response")
predictions_class <- predictions_prob > 0.5
matrice_confusion <- table(Observé = y_test, Prédit = predictions_class)
accuracy_lasso <- sum(diag(matrice_confusion)) / sum(matrice_confusion)

print(matrice_confusion)
```

```
##        Prédit
## Observé FALSE TRUE
##   FALSE   184   22
```

```
##   TRUE     15  214
```

```
print(coef(cv_lasso, s = "lambda.min"))
```

```
## 23 x 1 sparse Matrix of class "dgCMatrix"
##                       lambda.min
## (Intercept)       -7.511740e+01
## (Intercept)        .
## X0                 3.099064e-03
## RH                 .
## SSRD               4.915576e-06
## STRD               .
## T2M                3.902381e-01
## T2Mmax             .
## T2Mmin             1.542121e-01
## Covid             -3.202970e-02
## Holidays          -4.583845e+00
## TOY_sin            .
## TOY_cos            .
## DOW_sin            2.483296e+00
## DOW_cos           -1.783328e+00
## Total_prec         5.213394e-01
## Total_prec_2TRUE   2.529341e-02
## Total_prec_3TRUE   1.350580e+00
## Total_prec_4TRUE   1.422856e+00
## Total_prec_5TRUE   6.059843e-01
## Total_prec_6TRUE   2.771468e-01
## Total_prec_7TRUE   1.523553e+00
## Total_prec_14TRUE  1.001915e+00
```

```
print(paste("Accuracy (Lasso):", accuracy_lasso))
```

```
## [1] "Accuracy (Lasso): 0.914942528735632"
```

Ce premier modèle a préservé chaque variable auto-corrélée, ce qui semble indiquer que notre hypothèse hebdomadaire était en réalité correcte. Cependant, pour nous en assurer, nous allons observer la précision du modèle sans ces variables.

```
# Utilisation des mêmes données d'entraînement/test que le modèle 2
df_6_train <- df_6_sample$train
df_6_test  <- df_6_sample$test

# Préparation des matrices pour glmnet
x_train <- model.matrix(Total ~ ., data = df_6_train)
y_train <- df_6_train$Total
x_test <- model.matrix(Total ~ ., data = df_6_test)
y_test <- df_6_test$Total

# Validation croisée et ajustement du modèle Lasso
cv_lasso <- cv.glmnet(x_train, y_train, family = "binomial")

# Évaluation du modèle Lasso
predictions_prob <- predict(cv_lasso, x_test, type = "response")
predictions_class <- predictions_prob > 0.5
matrice_confusion <- table(Observé = y_test, Prédit = predictions_class)
accuracy_lasso <- sum(diag(matrice_confusion)) / sum(matrice_confusion)
```

```r
print(matrice_confusion)
```

```
##        Prédit
## Observé FALSE TRUE
##   FALSE   181   25
##   TRUE     16  215
```

```r
print(coef(cv_lasso, s = "lambda.min"))
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                       lambda.min
## (Intercept)      -7.709949e+01
## (Intercept)       .
## X0                3.240166e-03
## RH                .
## SSRD              6.508399e-06
## STRD              .
## T2M               8.471612e-02
## T2Mmax            .
## T2Mmin            4.348797e-01
## Covid            -3.370158e-02
## Holidays         -4.302203e+00
## TOY_sin          -4.461366e-01
## TOY_cos           .
## DOW_sin           3.591943e+00
## DOW_cos          -2.741275e+00
## Total_prec        4.932061e-01
## Total_prec_2TRUE  3.667272e-01
## Total_prec_3TRUE  2.149404e+00
## Total_prec_4TRUE  2.167923e+00
## Total_prec_5TRUE  1.023209e+00
## Total_prec_6TRUE  6.559572e-01
```

```r
print(paste("Accuracy (Lasso):", accuracy_lasso))
```

```
## [1] "Accuracy (Lasso): 0.906178489702517"
```

Nous observons une nette chutte de la précision (0.915 vs 0.906). Cela confirme finalement notre hypothèse: Les variables hebdomadaires ajoutent une quantité non-négligeable d'information.

Procédons de la même façon pour notre seconde hypothèse:

```r
# Utilisation des mêmes données d'entraînement/test que le modèle 2
df_6_2_train <- df_6_sample_2$train
df_6_2_test  <- df_6_sample_2$test

# Préparation des matrices pour glmnet
x_train <- model.matrix(Total ~ X0 + RH + SSRD + STRD + T2M + T2Mmax + T2Mmin + Covid + Holidays +
                TOY_sin * Total_prec_7 * Total_prec_14 + TOY_cos * Total_prec_7 * Total_prec_14 +
                DOW_sin + DOW_cos + Total_prec + Total_prec_2 + Total_prec_3 + Total_prec_4 +
                Total_prec_5 + Total_prec_6, data = df_6_2_train)
y_train <- df_6_2_train$Total
x_test <- model.matrix(Total ~ X0 + RH + SSRD + STRD + T2M + T2Mmax + T2Mmin + Covid + Holidays +
                TOY_sin * Total_prec_7 * Total_prec_14 + TOY_cos * Total_prec_7 * Total_prec_14 +
                DOW_sin + DOW_cos + Total_prec + Total_prec_2 + Total_prec_3 + Total_prec_4 +
                Total_prec_5 + Total_prec_6, data = df_6_2_test)
```

```r
y_test <- df_6_2_test$Total

# Validation croisée et ajustement du modèle Lasso
cv_lasso <- cv.glmnet(x_train, y_train, family = "binomial")

# Évaluation du modèle Lasso
predictions_prob <- predict(cv_lasso, x_test, type = "response")
predictions_class <- predictions_prob > 0.5
matrice_confusion <- table(Observé = y_test, Prédit = predictions_class)
accuracy_lasso <- sum(diag(matrice_confusion)) / sum(matrice_confusion)

print(matrice_confusion)
```

```
##        Prédit
## Observé FALSE TRUE
##   FALSE   184   22
##   TRUE     14  215
```

```r
print(coef(cv_lasso, s = "lambda.min"))
```

```
## 30 x 1 sparse Matrix of class "dgCMatrix"
##                                                lambda.min
## (Intercept)                                  -5.637749e+01
## (Intercept)                                   .
## X0                                            3.339712e-03
## RH                                            4.118196e-02
## SSRD                                          2.381844e-06
## STRD                                         -3.201036e-05
## T2M                                           3.848422e-01
## T2Mmax                                        .
## T2Mmin                                        1.143648e+00
## Covid                                        -3.520480e-02
## Holidays                                     -4.954810e+00
## TOY_sin                                       9.315754e-01
## Total_prec_7TRUE                              2.425706e+00
## Total_prec_14TRUE                             1.815365e+00
## TOY_cos                                       1.284758e+00
## DOW_sin                                       2.758801e+00
## DOW_cos                                      -2.318169e+00
## Total_prec                                    4.928299e-01
## Total_prec_2TRUE                              2.131394e-01
## Total_prec_3TRUE                              1.621957e+00
## Total_prec_4TRUE                              1.440935e+00
## Total_prec_5TRUE                              7.654217e-01
## Total_prec_6TRUE                              8.729868e-01
## TOY_sin:Total_prec_7TRUE                     -8.143557e-01
## TOY_sin:Total_prec_14TRUE                    -5.167329e-01
## Total_prec_7TRUE:Total_prec_14TRUE           -2.304594e+00
## Total_prec_7TRUE:TOY_cos                     -8.075110e-01
## Total_prec_14TRUE:TOY_cos                    -1.608826e+00
## TOY_sin:Total_prec_7TRUE:Total_prec_14TRUE -4.582459e-01
## Total_prec_7TRUE:Total_prec_14TRUE:TOY_cos  .
```

```r
print(paste("Accuracy (Lasso):", accuracy_lasso))
```

```
## [1] "Accuracy (Lasso): 0.917241379310345"
```

Ce résultat est extrêmement intéressant: * Les variables disparues précédemment sont réapparues. Donc comme nous l'avions hypothétisé, en confrontant *TOY* cos et *TOY* cos avec nos variables hebdomadaires, nous avons donné une nouvelle manière d'interpréter le reste de nos variables, ce qui a permis d'obtenir un nouveau record de précision.

Nous observons donc que notre hypothèse était aussi correcte, et nous arrivons à une précision finale de 0.917.

De plus, une annalyse des lambdas de notre dernier modèle révèle que nos variables autocorrellées sont prépondérantes. Par exemple, Total_prec_7 est la troisième variable la plus importante (ce qui était attendu, car l'ACF la présentait comme la variable la plus correllée à Total). Cela est d'ailleurs très intuitif: Il est probable qu'un individu passe un jour 'j' de la même manière cette semaine que la précédente, et de ce fait, il consommera autant. Dès lors, il est attendu que Total_prec_7 soit un prédicteur avancé.

Ceux-ci confirment aussi l'intérêt de notre transformation trigonométrique des variables cycliques, car DOW_sin est maintenant la seconde variable la plus importante pour notre meilleur modèle.

4 Conclusion

La comparaison finale des stratégies de modélisation a montré la supériorité du modèle Lasso avec corrélations hebdomadaires et termes d'intéraction avec les variables cycliques, qui a atteint une précision de **0.917** sur l'échantillon de test. Ce modèle a dépassé la regression Lasso, tout en maintenant sa robustesse.

Il a aussi nettement dépassé la méthode *step*, démontrant ainsi les limites de la robustesse de celle-ci.

Plus encore: Le fait que l'ajout de variables utiles mais fortement corrélées ait fait chuter le score de la méthode *step* a démontré que cette approche est très instable, et sensible à l'overfitting.