Analyse Fonctionnelle

CARLU Jonas, AFANE Fouad, LANCON Cindy

2026-02-24

Introduction

Ce projet a pour objectif d'étudier les profils de consommation électrique de $n = 6\,291$ entreprises, observées sur $D = 672$ instants de mesure espacés de 30 minutes sur deux semaines consécutives.

Contrairement à une analyse classique où nous traitons chaque instant de mesure serait traité comme une variable indépendante, nous considérons ici que chaque ligne du jeu de données correspond à une fonction de consommation évoluant dans le temps. Nous adaptons donc une approche d'Analyse de Données Fonctionnelles (FDA). L'analyse reposera sur trois étapes principales :

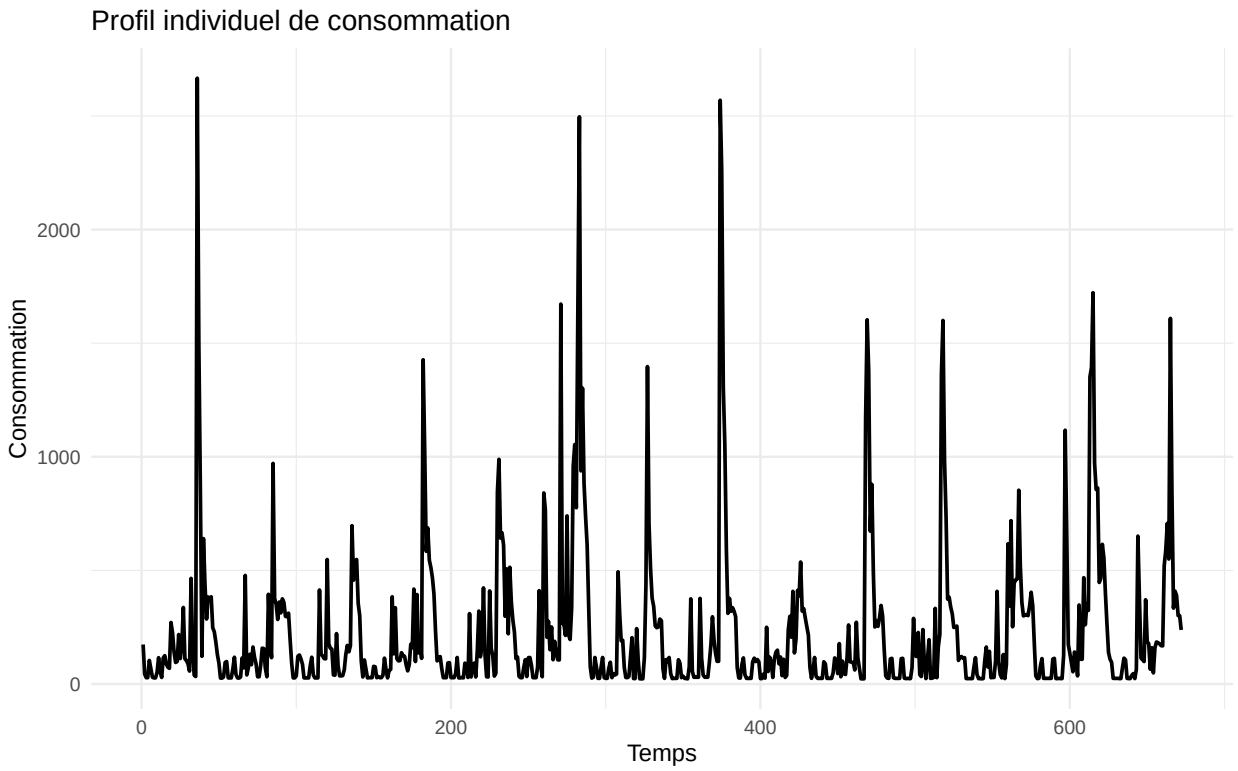
- Reconstruction des courbes continues par lissage B-splines pénalisé.
- Mise en oeuvre d'une Analyse en Composantes Principales Fonctionnelle (ACP fonctionnelle), afin de résumer la variabilité des courbes et d'identifier les principaux modes de variation des profils de consommation.
- Classification des entreprises par clustering, afin de regrouper les consommateurs présentant des comportements similaires.

```
#Chargement des données
chemin <- "donnees/smart278co.Rdata"
load(chemin)
table <- smart.278co
df <- as.data.frame(table)

#Nb de lignes
n <- nrow(table)
#Nb de cols
p <- ncol(table)
#indices des mesures
time <- 1:p
```

Exploration des données

Avant toute modélisation, on peut représenter des exemples des données brutes :

```
#Afficher le profil du premier individu
ggplot() +
  geom_line(aes(x = time, y = as.numeric(table[1, ])), linewidth = 0.8) +
  theme_minimal() +
  labs(x = "Temps", y = "Consommation", title = "Profil individuel de consommation")
```

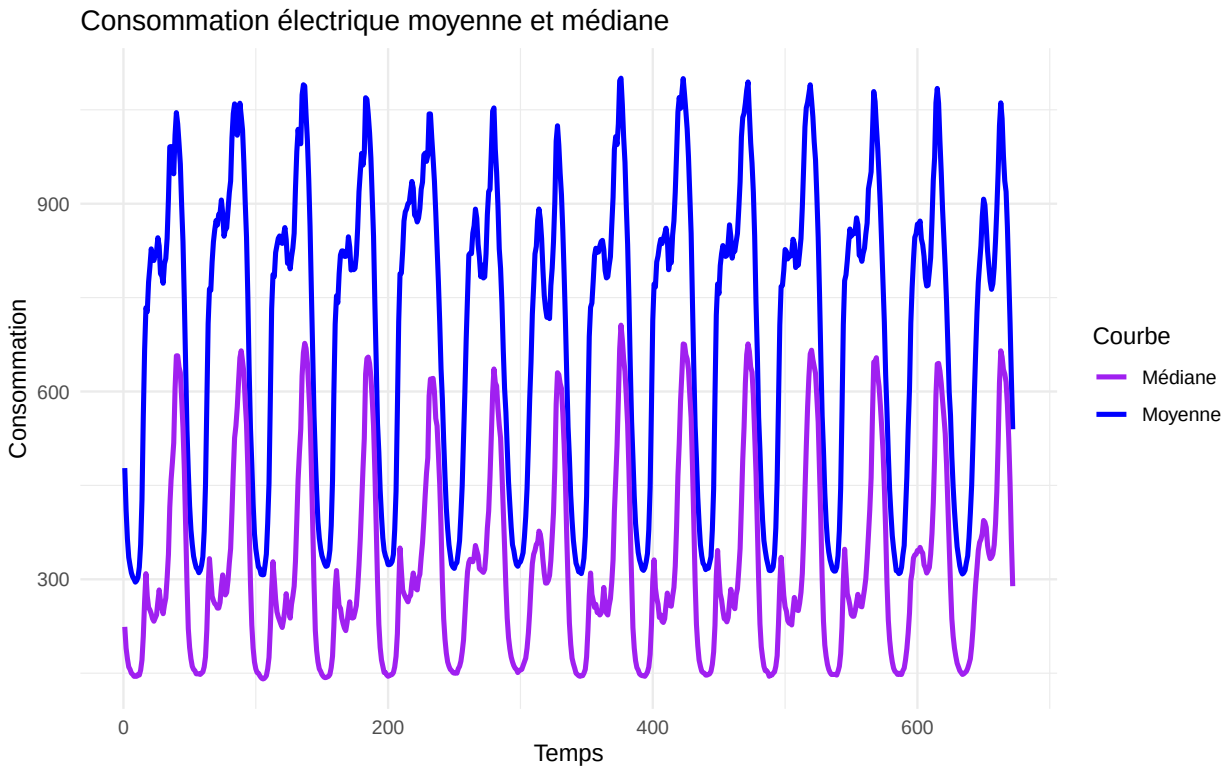


Les profils présentent une structure périodique marquée correspondant aux cycles journaliers. On observe des pics de consommation récurrents, probablement liés aux usages domestiques (matin, soir, weekends), ainsi qu'une variabilité notable entre individus.

Afin de synthétiser cette information, nous calculons ensuite la courbe moyenne et la courbe médiane, plus robuste aux profils atypiques.

```
#Courbe moyenne et mediane
f_mean <- colMeans(table,  na.rm=T)
f_median <- apply(table, 2, median, na.rm=T)
```

```
ggplot() +
  geom_line() +
  theme_minimal() + labs(x = "Temps", y = "Consommation")+
  #courbe moyenne
  geom_line(aes( x=time, y=f_mean, color="Moyenne"),
            linewidth = 1.1)+
  #courbe mediane
  geom_line(aes(x=time, y=f_median, color="Médiane"),
            linewidth = 1.1)+
  #Couleurs
  scale_color_manual(values = c("Moyenne"="blue", "Médiane"="purple"))+
  theme_minimal()+
  labs(title = "Consommation électrique moyenne et médiane", color="Courbe")
```



La courbe moyenne met clairement en évidence une structure périodique régulière.On observe clairement quatorze cycles journaliers successifs.

Chaque cycle présente une dynamique similaire : une phase de faible consommation nocturne suivie de pics plus marqués en journée, probablement associés aux usages domestiques (chauffage, cuisson, éclairage).

La stabilité de la forme moyenne suggère que la variabilité observée entre individus porte principalement sur l'amplitude des pics plutôt que sur la structure temporelle globale.

Vérification des valeurs manquantes

```
sum(is.na(table))
```

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

Aucune valeur manquante n'est détectée dans la matrice de données. Il n'est donc pas nécessaire d'effectuer une étape d'imputation ou de reconstruction préalable.

Lissage semi-paramétrique

L'objectif de cette partie est de passer des observations discrètes Y_i à des courbes continues $\hat{f}_i$, afin de pouvoir appliquer par la suite l'ACP fonctionnelle et le clustering.

Pour réaliser cette transformation, deux alternatives méthodologiques se présentent : l'approche historique de l'Analyse de Données Fonctionnelles (stricte projection sur une base de B-splines avec optimisation du paramètre de lissage par Validation Croisée) et l'approche par splines pénalisées (optimisation automatique par la méthode REML).

Alternative 1 : Base de B-splines et Validation Croisée

Les données de consommation de chaque entreprise i sont observées aux instants discrétisés $t_1, ..., t_D$, avec D= 672 points qui sont équidistants espacés de 30 minutes sur deux semaines.

L'objectif de cette partie est donc de passer des observations discrètes et bruitées Y_i, aux véritables courbes continues $\hat{f}_i$. En effet, les données brutes comportent du bruit de mesure qui empêche de comparer les individus point par point.

Pour chaque entreprise i, on pose le modèle :

$$Y_i(t_d) = f_i(t_d) + \epsilon_i(t_d), d = 1, ..., D$$

où ϵ_i est le bruit et f_i est estimée dans une base de B-splines $B_1(.), ..., B_q(.)$, de dimension q = K+m, avec :

- K = nombre de noeuds intérieurs équidistants sur [1 : 672]
- m-1 = degré des polynômes par morceaux.

La courbe s'écrit donc :

$$f_i(t) = \sum_{k=1}^{q} \beta_k B_k(t)$$

Les coefficients β_k sont estimés en minimisant l'erreur quadratique pénalisée :

$$\sum_{i,t} (Y_i(t) - f(t))^2 \; + \; \lambda \int (f''(t))^2 \, dt$$

où :

- $\lambda \geq 0$ est le paramètre de lissage,
- le terme $\int (f''(t))^2 dt$ pénalise la rugosité de la fonction.

```
K <- 35 # Nombre de noeuds
degree <- 3 # Degré des polynômes, m-1
m <- 4 # Ordre (degré+1)
q <- K + degree + 1 # nbasis = q = K+m

# Objet basisfd : notre base
base_bspline <- create.bspline.basis(
  rangeval = c(1, p), # Intervalle de définition de la base : [1:672]
  nbasis = q,         # Nb total de fonctions de base
  norder = m          # Ordre des polynômes
)
```

Soit δ le paramètre de lissage utilisé dans la reconstruction d'une courbe Y_i. Il peut être sélectionné en minimisant le critère de validation croisée généralisée (GCV).

Nous calculons donc le δ^* optimal pour chaque individu séparément en balayant une grille de valeurs. Une approche individuelle est plus rigoureuse qu'un δ commun car les entreprises peuvent avoir des profils de consommation très différents.

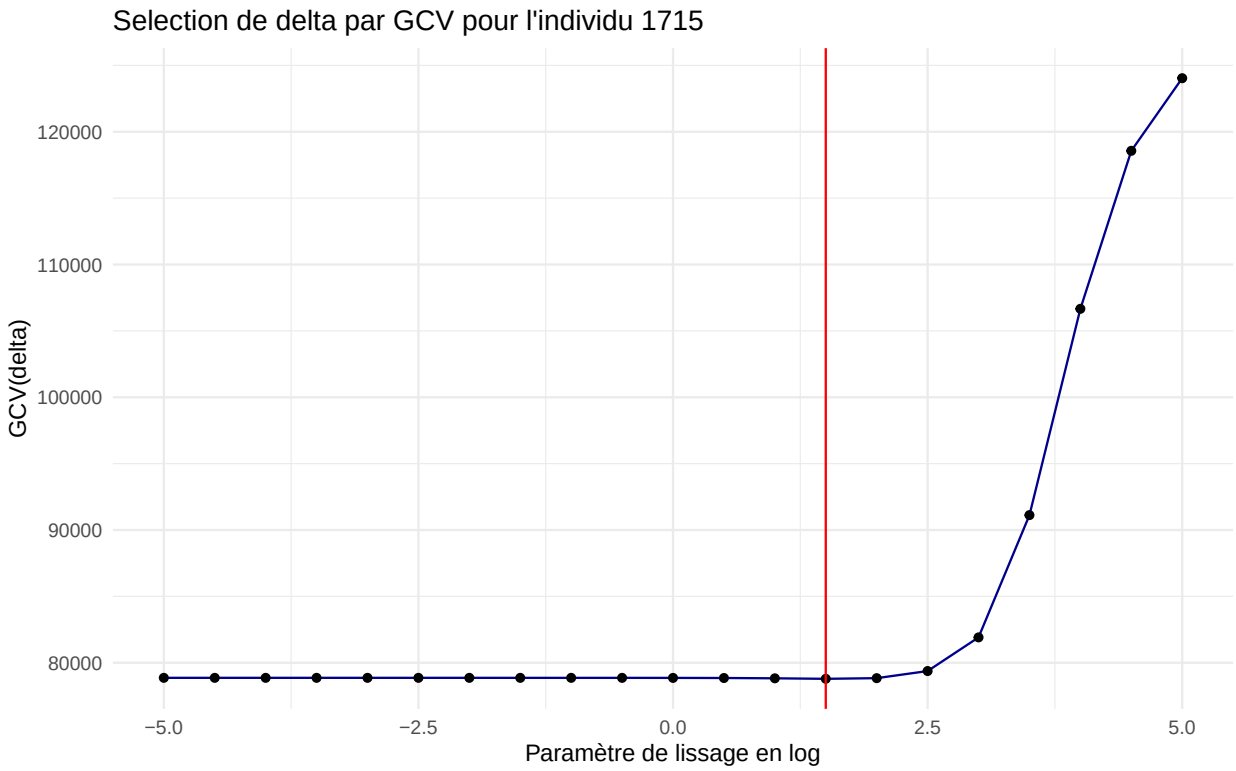
```r
#Grille des paramètre de lissage à tester
delta_grid <- 10^seq(-5, 5, by=0.5)


calcul_gcv <- function(y_i, delta){
  res <- fdPar(base_bspline, Lfdobj = 2, lambda = delta)
  mean(smooth.basis(time, y_i, res)$gcv)
}


delta_opt_i <- function(i){
  y_i <- as.numeric(smart.278co[i, ])
  gcv_val <- sapply(delta_grid, calcul_gcv, y_i = y_i)
  delta_grid[which.min(gcv_val)]
}
```

Vérification de l'optimisation sur un individu de test :

```r
set.seed(368)
index_sample <- sample(1:n, 3)
i <- index_sample[2]

y_i <- as.numeric(smart.278co[i, ])
delta_opt_i_test <- delta_opt_i(i)

cat("Le delta optimal pour l'individu : ",i," est ", delta_opt_i_test)
```

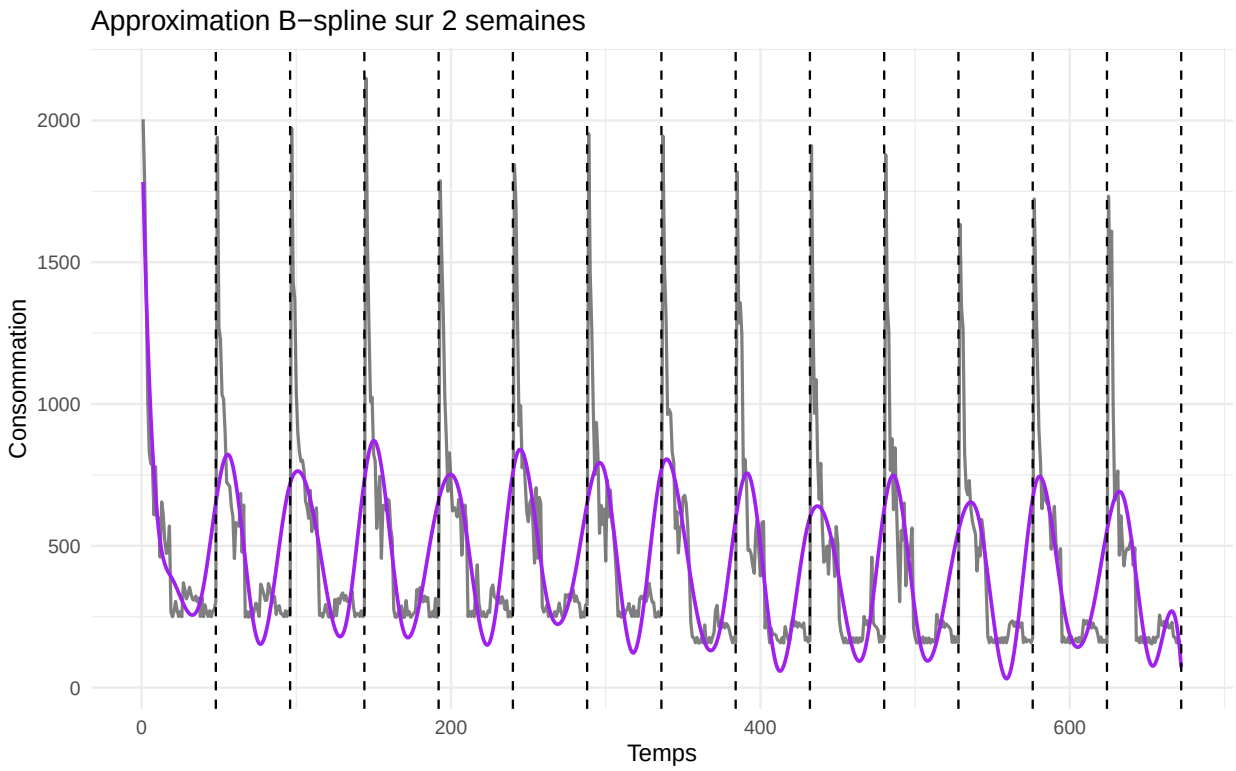
```
## Le delta optimal pour l'individu :  1715  est  31.62278
```

```r
gcv_val <- sapply(delta_grid, calcul_gcv, y_i = y_i)

#dataframe pour le plot
df_plot_test1 <- data.frame( delta=delta_grid, gcv = gcv_val)

ggplot( df_plot_test1, aes(x=log10(delta), y=gcv)) +
  geom_line(color="darkblue") +
  geom_point(color="black")+
  #Delta optimal
  geom_vline(xintercept = log10(delta_opt_i_test), color="red")+
  theme_minimal()+

  labs( title = paste("Selection de delta par GCV pour l'individu", i), x="Paramètre de lissage en log"
```



Visualisation de la courbe lissée :

```r
res_i <- fdPar(base_bspline, Lfdobj = 2, lambda = delta_opt_i_test)
smooth_i <- smooth.basis(time, y_i, res_i)
y_lisse <- as.numeric(eval.fd(time, smooth_i$fd))

df_i_fda <- data.frame(temps = time, y = y_i, lisse = y_lisse)

# Plot global sur 2 semaines
ggplot(df_i_fda, aes(x = temps)) +
  geom_line(aes(y = y), linewidth = 0.7, alpha = 0.5) +
  geom_line(aes(y = lisse), color = "purple", linewidth = 0.8) +
  geom_vline(xintercept = seq(48, 672, by = 48), linewidth = 0.5, linetype = "dashed") +
  theme_minimal() +
  labs(title = "Approximation B-spline sur 2 semaines", x = "Temps", y = "Consommation")
```



Application à l'ensemble de la base de données : (pour approche 2 PCA)

```
deltas_res <- sapply(1:n, delta_opt_i)
```

```
deltas_res <- readRDS("donnees/deltas_res.rds")
lisser_i <- function(i){
  y_i <- as.numeric(smart.278co[i, ])
  res <- fdPar(base_bspline, Lfdobj = 2, lambda = deltas_res[i])
  smooth.basis(time, y_i, res)$fd
}

#retourne list de 6291objets fd séparés -> pca.fd a besoin d'un seul objet
fd_list <- lapply(1:n, lisser_i)

#Extraire les coefs
extraire_coefs <- function(fd_i){
  as.numeric(fd_i$coefs)
}

coefs_mat <- sapply(fd_list, extraire_coefs)

#Reconstruire en un seul objet fd
fd_final <- fd(coefs_mat, base_bspline)
```

Alternative 2 : Splines Pénalisées et modèle GAM

La méthode décrite ici provient du livre *Functional Data Analysis with R*. Le modèle est identique à l'Alternative 1. La seule différence réside dans la sélection du paramètre de lissage λ : au lieu d'une

recherche manuelle par GCV, celui-ci est estimé automatiquement par la méthode REML (Restricted Maximum Likelihood) intégrée à gam(). Cette approche est plus rapide et évite de définir une grille de valeurs.

On utilise la fonction *gam*() pour ajuster les courbes lisses dans lequel on spécifie un terme de lissage utilisant des splines cubiques avec un maximum de 40 noeuds possibles.

Choix du nombre de noeuds K

(à compléter et justifier -> Règle de ruppert K =min(0.25*D, 35) ?)

k = 40 = q -> K = q-m = 36

Ou K = 84 (Fouad) : Le choix du nombre de fonctions de base est dicté par la nécessité de capturer les variations intra-journalières (pics de consommation du matin et du soir) sur une période de 14 jours.La consommation électrique présente au minimum deux événements majeurs par jour.

Exemple sur un individu

```
# on extrait le premier individu pour la démo
ind_1 <- as.numeric(table[1,])
df_1 <- data.frame(Temps=time, Var=ind_1)

# modèle de lissage
modele_gam <- gam(Var ~ s(Temps, bs = "cr", k = 40),
                  data = df_1,
                  method = "REML")

summary(modele_gam)
```

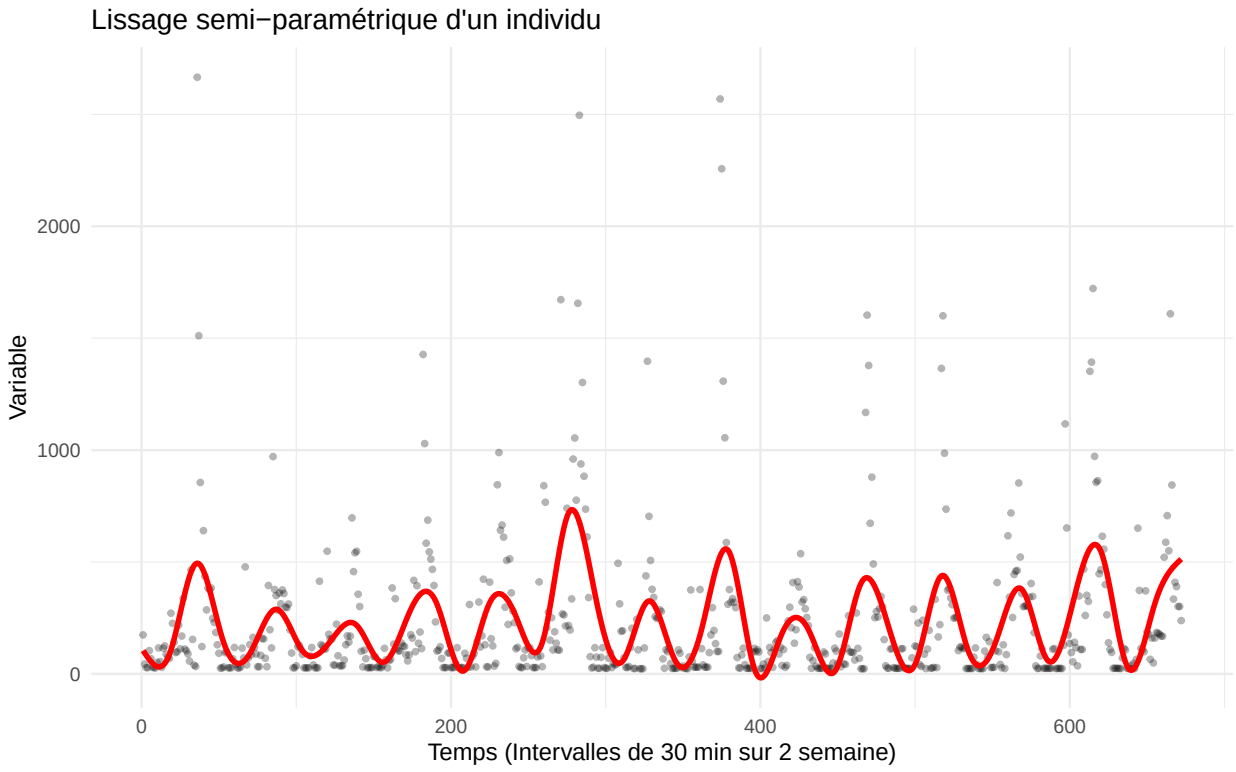
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Var ~ s(Temps, bs = "cr", k = 40)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   228.31      11.01   20.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df     F p-value
## s(Temps) 34.32  37.64 6.624  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.271   Deviance explained = 30.8%
## -REML = 4818.9  Scale est. = 81510     n = 672
```

Contrairement à la première alternative où le paramètre de lissage optimal devait être recherché manuellement via une grille de valeurs et la validation croisée généralisée (GCV), la fonction *gam*() automatise ce processus.

En spécifiant *method* = "REML", l'algorithme reformule le problème de lissage sous la forme d'un modèle mixte. Le paramètre de pénalité λ est alors estimé directement, comme un ratio entre la variance du signal et la variance du bruit.

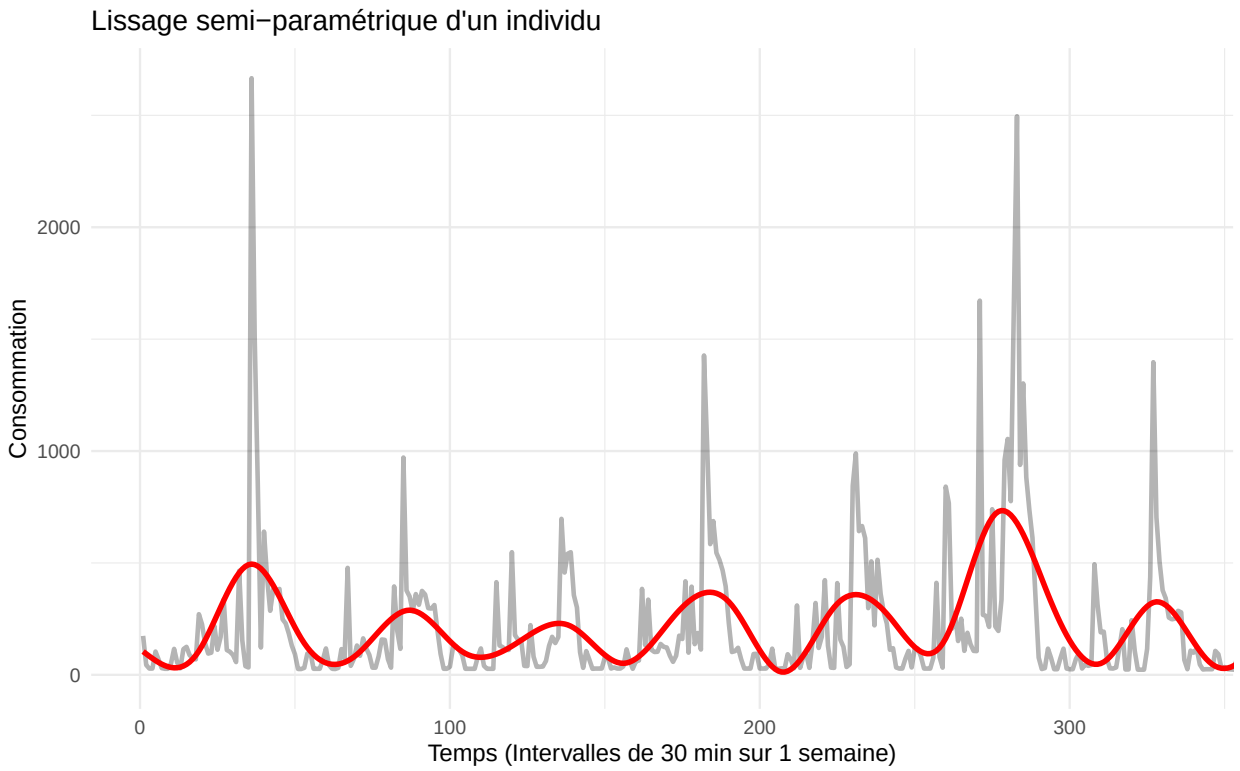
Le degré de lissage optimal est ainsi obtenu en une seule étape, sans risque de sous-lissage.

```
# prédiction de la courbe lisse
df_1$Courbe_Lisse <- predict(modele_gam, newdata = df_1)

# données brutes VS signal estimé
ggplot(df_1, aes(x = Temps)) +
  geom_point(aes(y = Var), alpha = 0.3, color = "black", size = 1) +
  geom_line(aes(y = Courbe_Lisse), color = "red", linewidth = 1.2) +
  theme_minimal() +
  labs(title = "Lissage semi-paramétrique d'un individu",
       x = "Temps (Intervalles de 30 min sur 2 semaine)",
       y = "Variable")
```

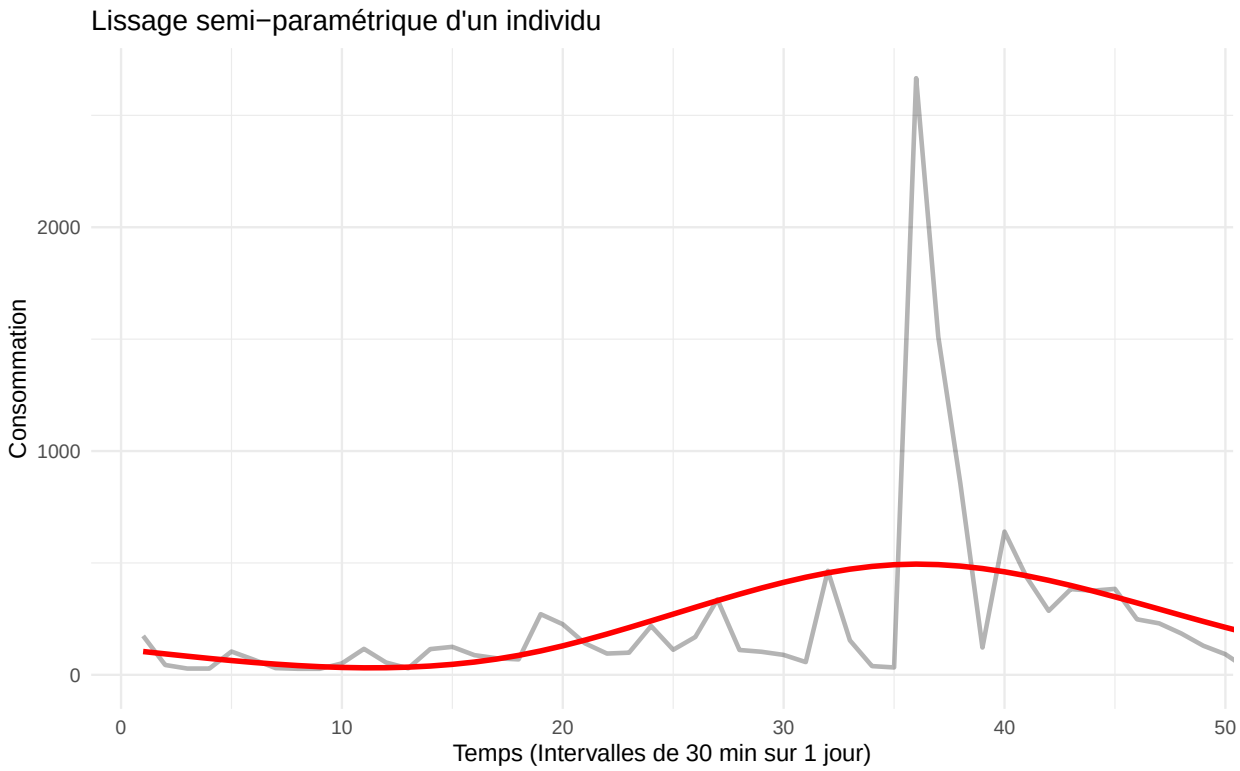



```
# summary(modele_gam)
```

```
# données brutes VS signal estimé : 1 semaine
ggplot(df_1, aes(x = Temps)) +
  geom_line(aes(y = Var), alpha = 0.3, color = "black", size = 1) +
  geom_line(aes(y = Courbe_Lisse), color = "red", linewidth = 1.2) +
  coord_cartesian(xlim = c(1, 336)) +
  theme_minimal() +
  labs(title = "Lissage semi-paramétrique d'un individu",
       x = "Temps (Intervalles de 30 min sur 1 semaine)",
       y = "Consommation")
```




```r
# données brutes VS signal estimé : 1 jour
ggplot(df_1, aes(x = Temps)) +
  geom_line(aes(y = Var), alpha = 0.3, color = "black", size = 1) +
  geom_line(aes(y = Courbe_Lisse), color = "red", linewidth = 1.2) +
  theme_minimal() +
  coord_cartesian(xlim = c(1, 48)) +
  labs(title = "Lissage semi-paramétrique d'un individu",
       x = "Temps (Intervalles de 30 min sur 1 jour)",
       y = "Consommation")
```



Les grandes vagues dessinées par la courbe rouge représentent le rythme quotidien de consommation du premier individu. On remarque 14 pics et 14 creux. on en déduit que les creux correspondent aux périodes de faible consommation (la nuit) et les pics les périodes de haute consommation (milieu de journée).

Après avoir testé sur un individu et visualiser la qualité du lissage, on applique à l'ensemble des n = 6291 individu.

Application à tous les individus (pour approche 2 PCA)

(attention certains ont des consommations nulles tout le long des deux semaines, pose problème pour cette approche, on décide donc d'enlever les individus problématiques).

```
#Individu ayant un consommation nulle
#Renvoie True si la variance de l'individu i = 0 / conso constante
verif_conso <- function (i){
  y_i <- as.numeric(smart.278co[i, ])

  var(y_i)==0 || any(!is.finite(y_i))
}

probleme_conso <- sapply(1:n, verif_conso)
idx_pb <- which(probleme_conso)
idx_ok <- which(!probleme_conso)
cat("Nombre d'ind avec conso constantes :", length(idx_pb),"\n")
```

```
## Nombre d'ind avec conso constantes : 16
```

```r
cat("Nombre d'ind ok :", length(idx_ok))
```

```
## Nombre d'ind ok : 6275
```

```r
#Fonction pour lisser chaque courbe de chaque individu
lisser_gam_i <- function(i){

  y_i <- as.numeric(smart.278co[i, ])

  df_i <- data.frame(Temps = time, Y=y_i)

  mod <- gam(Y~ s(Temps, bs = "cr", k = 40),
                 data = df_i,
                 method = "REML")

  #Recuperer le lambda optimal trouvé par REML
  lambda_opt <- mod$sp
  res <- fdPar(base_bspline, Lfdobj = 2, lambda = lambda_opt)
  smooth.basis(time, y_i, res)$fd
}

#Matrice pxn des courbes lissées
fd_list <- lapply(idx_ok, lisser_gam_i)
coefs   <- sapply(fd_list, function(fd_i) as.numeric(fd_i$coefs))
fd_final <- fd(coefs, base_bspline)
#Ligne executée qu'une fois
saveRDS(fd_final, file = "donnees/fd_final.rds") #15-20 min
```

```r
#chargement du lissage
fd_final <- readRDS("donnees/fd_final.rds")
```

Analyse en Composantes Principales Fonctionnelle (ACPF)

L'Analyse en Composantes Principales Fonctionnelle (FPCA) permet de résumer la variabilité des courbes $Y_i(t)$ autour de leur moyenne fonctionnelle $\mu(t)$.

Le principe consiste à rechercher des fonctions propres $\phi_k(t)$ maximisant la variance projetée :

$$Y_i(t) = \mu(t) + \sum_{k=1}^{\infty} \xi_{ik}\phi_k(t)$$

où :

- $\mu(t)$ est la courbe moyenne de la population,
- $\phi_k(t)$ sont les fonctions propres (composantes principales fonctionnelles)
- ξ_{ik} sont les scores principaux de l'individu i sur la k-ième composante.

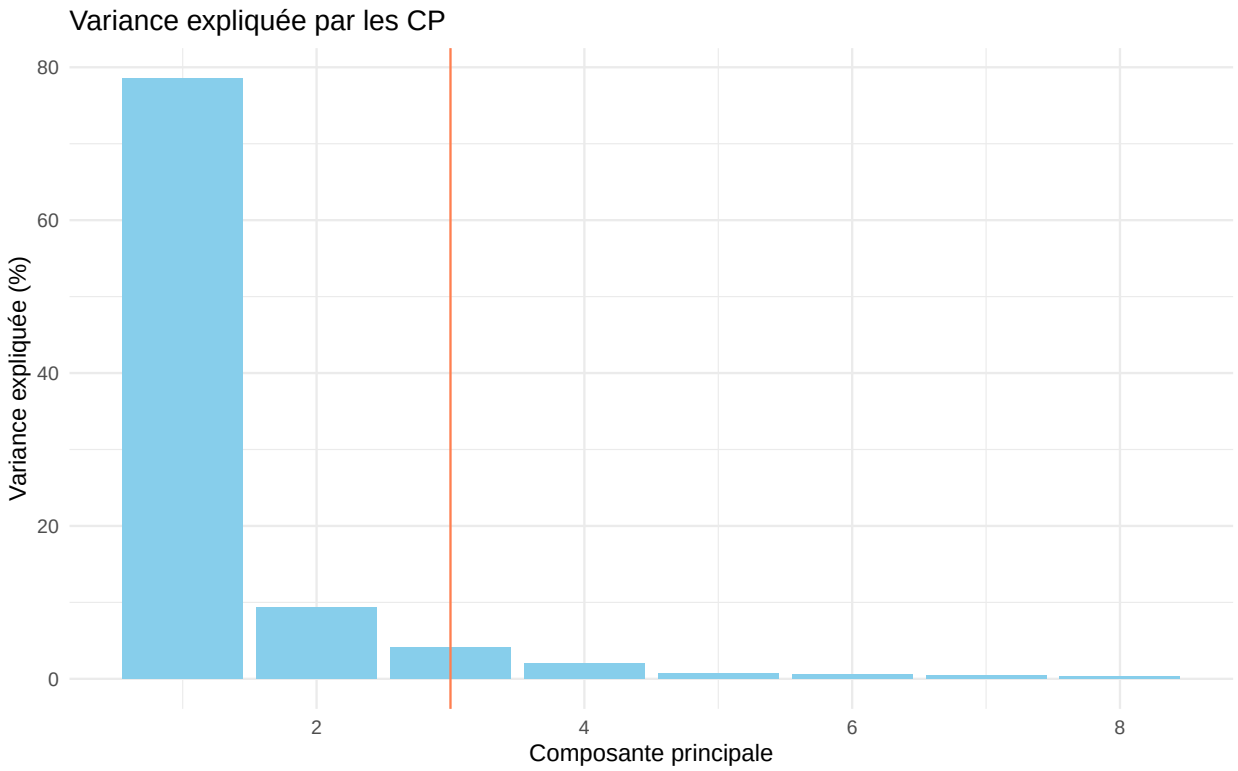
Approche 1 : pca.fd() - package fda

Cette approche consiste à d'abord lisser individuellement les courbes (section précédente), puis l'ACPF est appliquée sur l'objet fd en utilisant pca.fd(). L'opérateur de covariance est estimé à partir des coefficients B-splines des courbes lissées. (prends du temps)

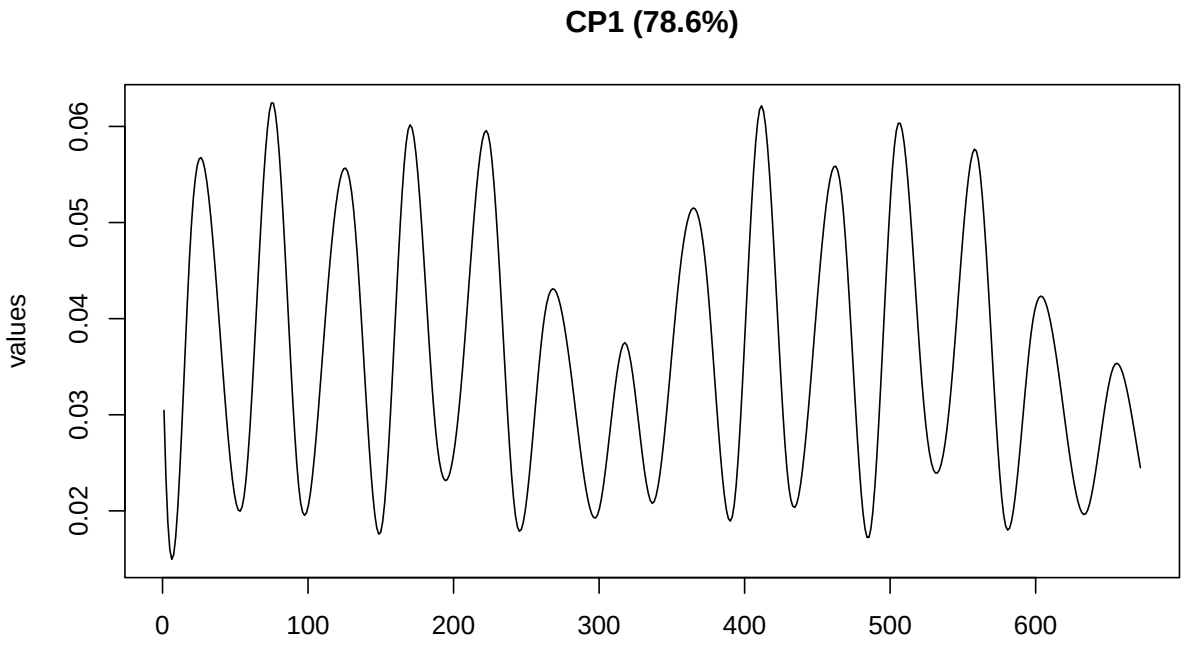
```r
# PCA fonctionnelle
pca_res <- pca.fd(fd_final, nharm = 10)

# Variance expliquée
var <- pca_res$varprop *100
var_cum <- cumsum(pca_res$varprop*100)

data.frame(CP  = 1:10,
           prop = round(var[1:10], 2),
           cum  = round(var_cum[1:10], 2))
```

```
##    CP  prop   cum
## 1   1 78.57 78.57
## 2   2  9.40 87.97
## 3   3  4.09 92.06
## 4   4  2.04 94.09
## 5   5  0.73 94.82
## 6   6  0.55 95.37
## 7   7  0.50 95.87
## 8   8  0.35 96.22
## 9   9  0.32 96.54
## 10 10  0.29 96.83
```

```r
#Nb de CP qui permettent 90% de variance expliquée
nb_cp <- min(which(var_cum >= 90))

#DF pour simplifier le plot
df_var <- data.frame(
  CP = 1:8,
  prop = var[1:8]
)

#Plot de la variance expliquée
ggplot(df_var, aes(x=CP, y=prop))+
  geom_col(fill="skyblue")+
  geom_vline(xintercept=nb_cp, color="coral")+
  theme_minimal()+
  labs(title = "Variance expliquée par les CP",
       x = "Composante principale", y="Variance expliquée (%)")
```

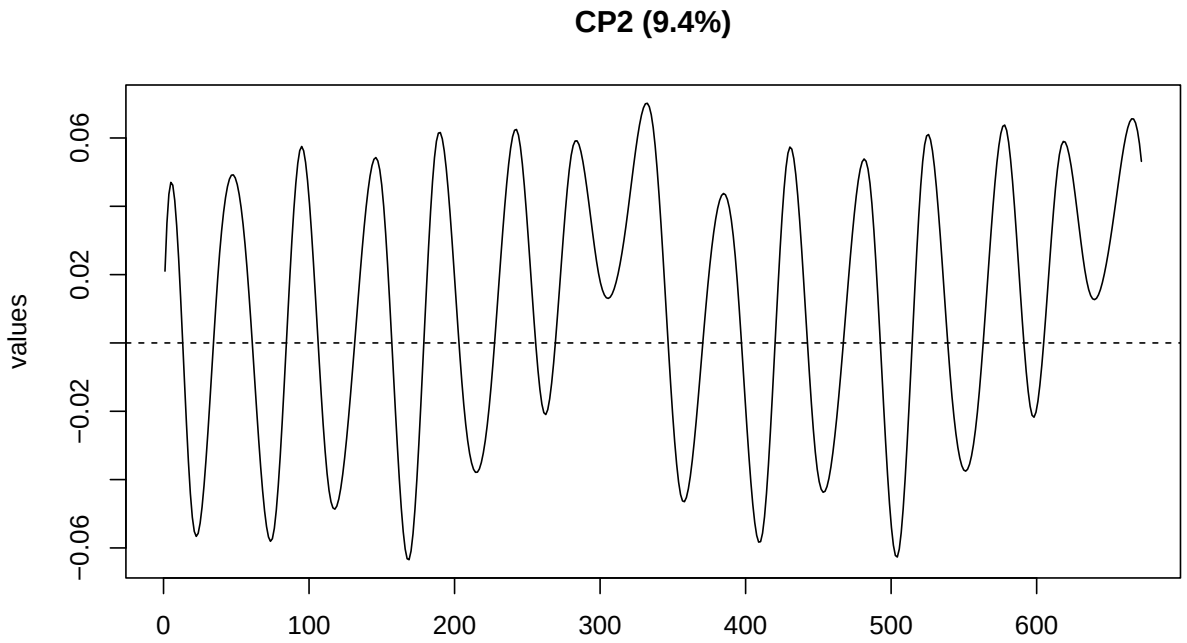



```
plot(pca_res$harmonics[1], main = paste0("CP1 (", round(var[1], 1), "%)"))
```



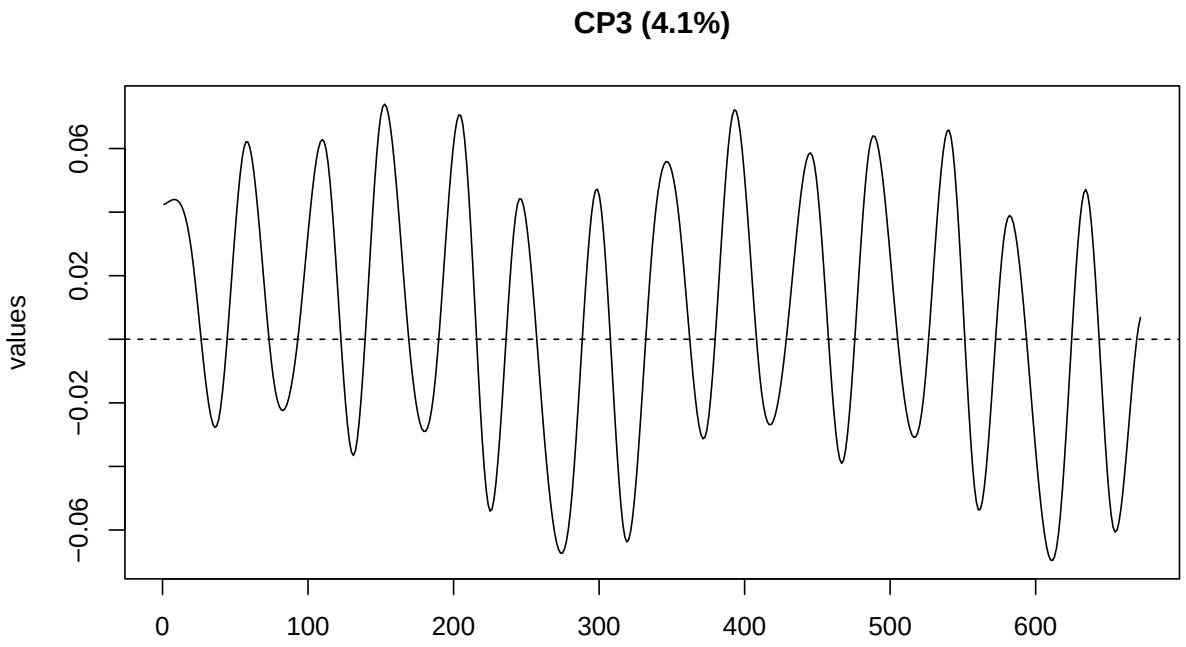

```
## [1] "done"
```

```r
plot(pca_res$harmonics[2], main = paste0("CP2 (", round(var[2], 1), "%)"))
```




```
## [1] "done"
```

```r
plot(pca_res$harmonics[3], main = paste0("CP3 (", round(var[3], 1), "%)"))
```




```
## [1] "done"
```

CP1 : niveau global consommation, CP2 : contraste jour/nuit

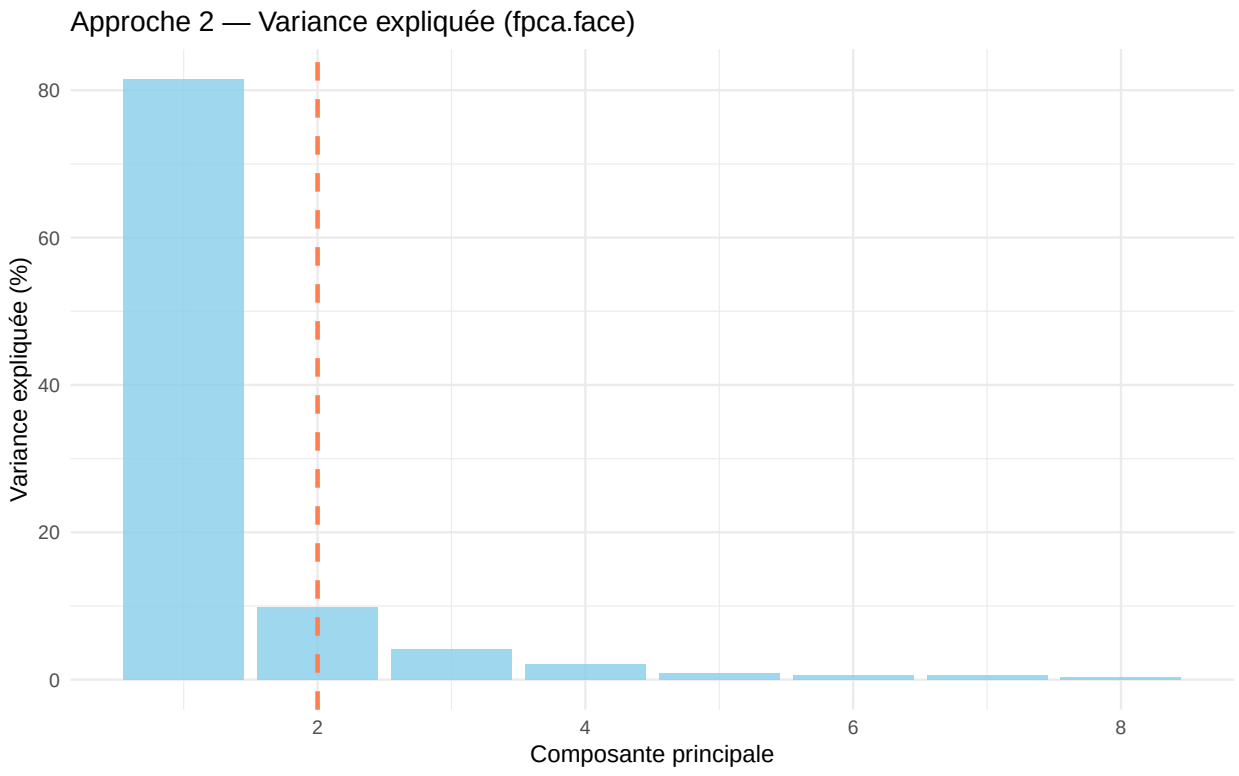
Approche 2 : fpca.face() — package refund

Approche, proposée par Xiao et al. (2016)

On utilise l'algorithme $fpca.face()$ du package $refund$ qui estime et lisse la matrice de covariance globale (au lieu de lisser les individus un par un). A partir de cette matrice, il extrait les fonctions propres (les composantes) et les scores de chaque individu. On extraie ici les deux premières composantes (PC1 et PC2), que lon multiplie par $\sqrt{p}$ (c'est le bouquin qui dit qu'il faut le faire, jai pas trop compris pourquoi...) On lui demande 50 noeuds ($knots = 50$) et de retenir les composantes expliquant 95% de la variance ($pve = 0.95$).

```
W <- as.matrix(table)

modele_fpca <- fpca.face(Y = W, argvals = time, knots = 50, pve = 0.95)
# summary(modele_fpca)

#on extraie les éléments de la FCPA
courbe_moyenne <- modele_fpca$mu
fonctions_propres <- modele_fpca$efunctions
scores_individuels <- modele_fpca$scores
valeurs_propres <- modele_fpca$evalues

variance_expliquee <- valeurs_propres / sum(valeurs_propres)
```

```r
# on extrait les fonctions propres
pc1 <- fonctions_propres[, 1] * sqrt(p)
pc2 <- fonctions_propres[, 2] * sqrt(p)

data_plot <- data.frame(Temps = time, PC1 = pc1)
data_plot2 <- data.frame(Temps = time, PC2 = pc2)


# Variance expliquée
var2     <- modele_fpca$evalues / sum(modele_fpca$evalues) * 100
var_cum2 <- cumsum(var2)
nb_cp2 <- min(which(var_cum2 >= 90))
df_var2 <- data.frame(
  CP   = 1:length(var2),
  prop = var2
)

ggplot(df_var2, aes(x = CP, y = prop)) +
  geom_col(fill = "skyblue", alpha = 0.8) +
  geom_vline(xintercept = nb_cp2, color = "coral", linetype = "dashed", linewidth = 1) +
  theme_minimal() +
  labs(title = "Approche 2 - Variance expliquée (fpca.face)",
       x = "Composante principale", y = "Variance expliquée (%)")
```

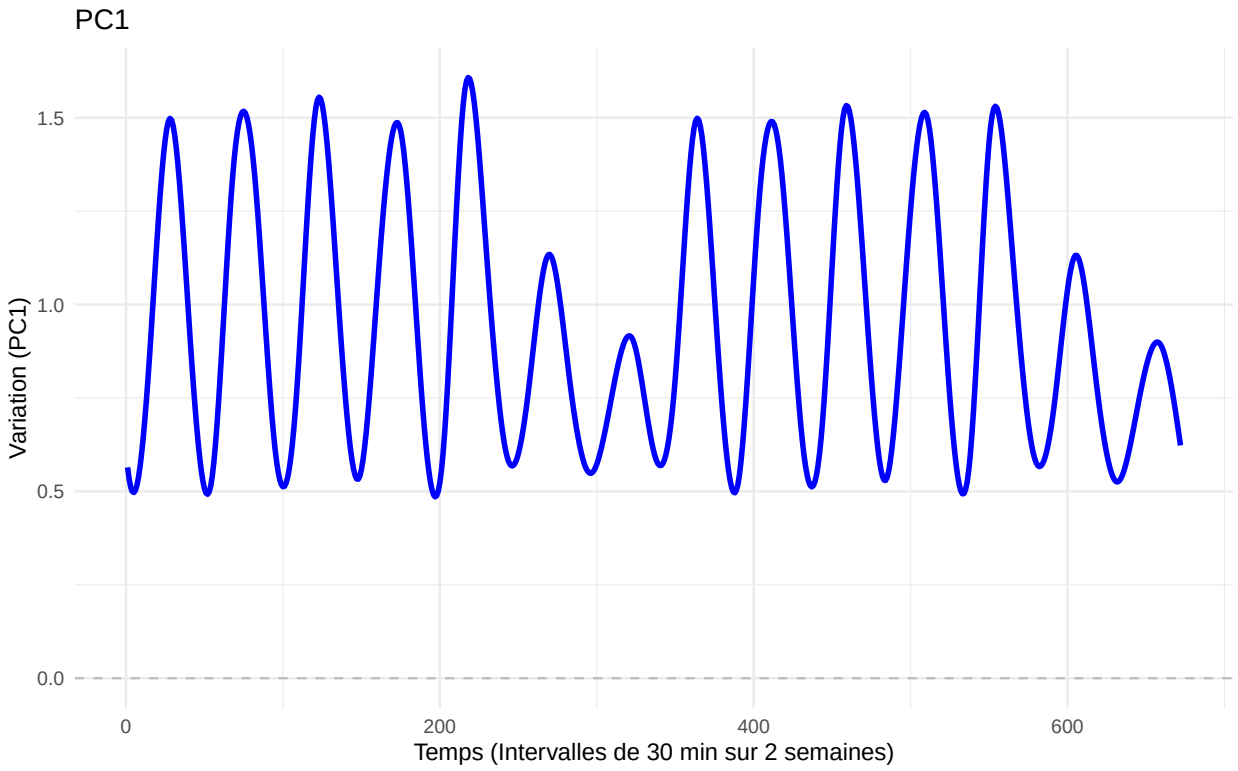



```r
df_var2
```

```
##   CP       prop
```

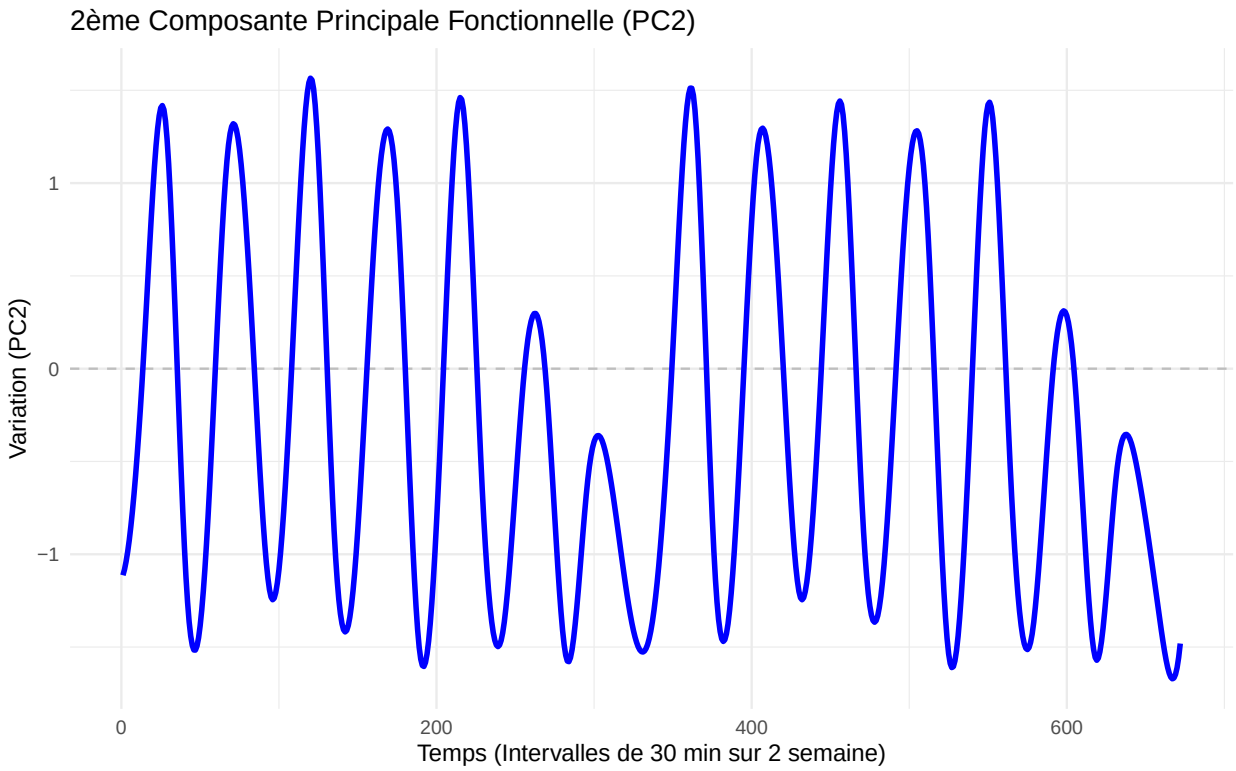
```
## 1  1 81.5122791
## 2  2  9.9030870
## 3  3  4.0723509
## 4  4  2.1297840
## 5  5  0.8201363
## 6  6  0.6269994
## 7  7  0.5553995
## 8  8  0.3799638
```

```r
# visualisation de la PC1
ggplot(data_plot, aes(x = Temps)) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray") +
  geom_line(aes(y = PC1), color = "blue", linewidth = 1.2) +
  theme_minimal() +
  labs(title = "PC1",
       x = "Temps (Intervalles de 30 min sur 2 semaines)",
       y = "Variation (PC1)")
```



On observe une courbe qui ondule et qui reste strictement positive.

```r
# visualisation de la PC2
ggplot(data_plot2, aes(x = Temps)) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray") +
  geom_line(aes(y = PC2), color = "blue", linewidth = 1.2) +
  theme_minimal() +
  labs(title = "2ème Composante Principale Fonctionnelle (PC2)",
       x = "Temps (Intervalles de 30 min sur 2 semaine)",
       y = "Variation (PC2)")
```



On observe là aussi une courbe qui ondule mais cette fois qui prend des valeurs négatives ou nulles.

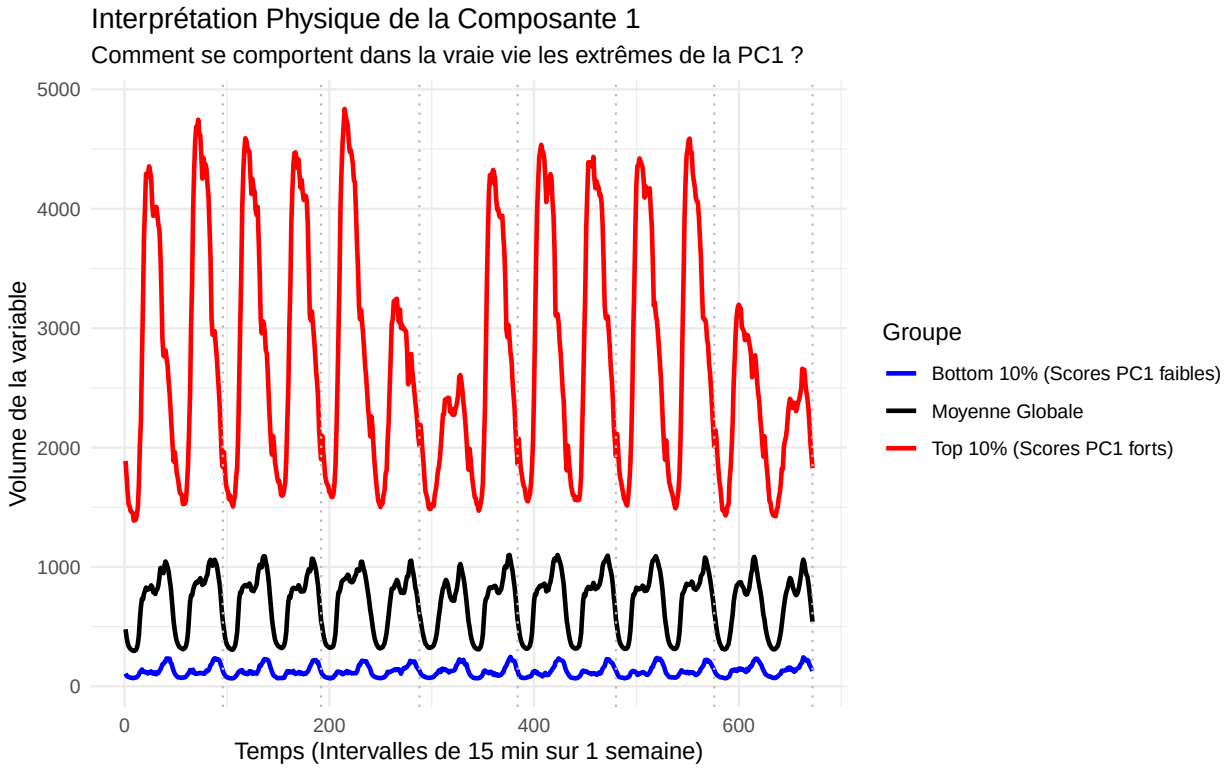
(Jonas :)

Interpréation de la PC1

La courbe de la première composante est assez abstraite. On cherche à comprendre ce que la PC1 mesure physiquement sur notre jeu. Pour essayer de traduire, on isole les comportements extrêmes de la population : les individus dans le top 10% des scores de la PC1 et ceux dans le bottom 10% de la PC1. On calcule ensuite la vraie courbe moyenne de ces deux groupes pour observer comment la PC1 affecte la forme globale de la productio sur une semaine.

```
#score de la composante 1
scores_pc1 <- modele_fpca$scores[, 1]

seuil_haut <- quantile(scores_pc1, 0.90)
seuil_bas <- quantile(scores_pc1, 0.10)

#individus extrêmes
index_haut <- which(scores_pc1 >= seuil_haut)
index_bas <- which(scores_pc1 <= seuil_bas)

#vraies moyennes pour groupes extrêmes
courbe_moyenne_top10 <- colMeans(W[index_haut, ])
courbe_moyenne_bot10 <- colMeans(W[index_bas, ])

#dataframe pour graphique
df_interpretation <- data.frame(
```

```
  Temps = rep(time, 3),
  Activite = c(courbe_moyenne_top10, courbe_moyenne_bot10, courbe_moyenne),
  Groupe = rep(c("Top 10% (Scores PC1 forts)",
                 "Bottom 10% (Scores PC1 faibles)",
                 "Moyenne Globale"), each = 672)
)

ggplot(df_interpretation, aes(x = Temps, y = Activite, color = Groupe)) +
  geom_line(linewidth = 1) +
  scale_color_manual(values = c("Top 10% (Scores PC1 forts)" = "red",
                                "Bottom 10% (Scores PC1 faibles)" = "blue",
                                "Moyenne Globale" = "black")) +
  geom_vline(xintercept = seq(96, 672, by = 96), color = "gray", linetype = "dotted") +
  theme_minimal() +
  labs(title = "Interprétation Physique de la Composante 1",
       x = "Temps (Intervalles de 15 min sur 1 semaine)",
       y = "Volume de la variable",
       subtitle = "Comment se comportent dans la vraie vie les extrêmes de la PC1 ?")
```



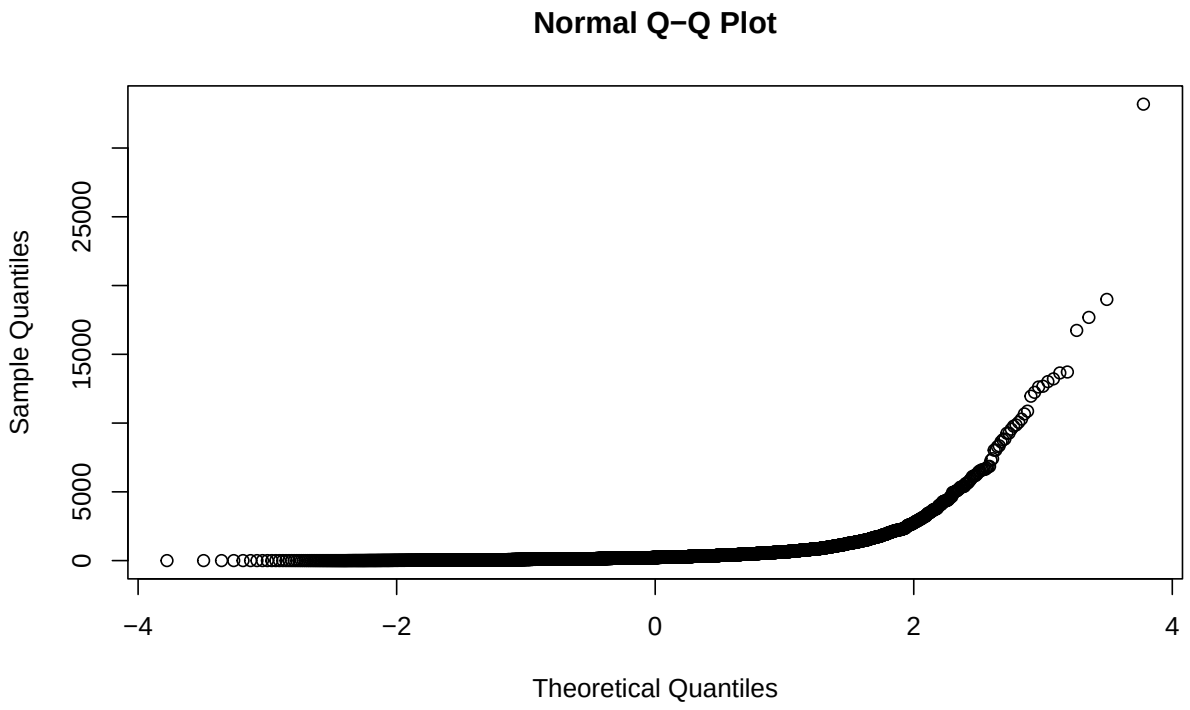
On déduit que la première composante agit comme un multiplicateur global sur les deux semaines. On constate que les individus du top 10% consomment massivement plus d'énergie que la moyenne. Leurs pics journaliers sont également bien plus importants. A l'inverse, la consommation du Bottom 10% semble presque plate sur l'échelle du réseau.

GFCPA

Normalité

La FCPA décrite dans le premier point suppose que les données suivent une loi normale (d'après le chapite 3.2). Nous allons vérifier cette hypothèse sur le premier instant temporel :

```
qqnorm(df[,1])
```



On voit que les points ne s'alignent pas sur une droite. Ainsi, nos données de consommation ne sont pas gaussiennes sur le premier instant d'où l'utilité de la méthod

Préparation et méthode de binning (fenêtrage)

Lobjectif est ici de préparer les données pour les algorithmes de modèles mixtes. En premier lieu nous convertissons notre data frame en format long i.e une ligne par observation temporelle par individu. Ensuite, nous appliquons la méthode du Binning : au lieu de calculer un modèle mixte pour chacun des 672 points sur 6291 individus (ce quiserait trop complexe), on regroupe les intervalles de 30mn en blocs de 4 heures (8 intervalles).

```
p <- 672
n <- nrow(W)

df_long <- data.frame(
  ID = as.factor(rep(1:n, times = p)),
  Temps = rep(1:p, each = n),
```

```
  Production = as.vector(W)
)

# binning
taille_fenetre <- 8

df_long <- df_long %>%
  mutate(Fenetre = floor((Temps - 1) / taille_fenetre) * taille_fenetre + (taille_fenetre / 2))

centres_fenetres <- unique(df_long$Fenetre)
```

Modèles mixtes locaux

Pour chaque bloc de 4 heures on isole les données et on ajuste un modèle mixte grâce à la fonction $glmer()$. Puisque nous traitons des données semblables à des données de "comptage", nous utilisons la loi de Poisson avec un lien logarithmique ($family = poisson(link = "log")$). La valeur extraite pour chaque centrale est son effet aléatoire i.e son écart latent par rapport à la moyenne à la moyenne du réseau durant ces 4 heures.

Pour la fenêtre temporelle j, on modélise la production Y_{ij} de la centrale i par :

$Y_{ij} \sim \mathcal{P}(\mu_{ij})$

$\log(\mu_{ij}) = \beta_0 + b_{ij}$

Où μ_{ij} est la consommation moyenne théorique de la centrale i durant la fenêtre temporelle j, β_0 est l'intercepte fixe (la moyenne globale) et b_{ij} est l'effet aléatoire (la déviation latente propre à la centrale i).

Pour extraire les effets aléatoires de chaque modèle on utilise la fonction $ranef()$.

```
# matrice qui stocke les effets aléatoires
matrice_effets_locaux <- matrix(NA, nrow = n, ncol = length(centres_fenetres))

for (j in 1:length(centres_fenetres)) {
  fenetre_actuelle <- centres_fenetres[j]
  data_locale <- subset(df_long, Fenetre == fenetre_actuelle)

  modele_local <- glmer(Production ~ 1 + (1 | ID),
                        family = poisson(link = "log"),
                        data = data_locale)

  matrice_effets_locaux[, j] <- ranef(modele_local)$ID[[1]]
}
```

Nous allons maintenant lisser les signaux latents obtenus par fenêtres pour retrouver des profils de consommation continus sur les deux semaines entières. Pour cela, nous appliquons la FCPA classique sur la matrice des effets aléatoires pour lisser les sauts entre les fenêtres de 4h. Une fois les fonctions propres ϕ_k extraites nous les interpolons pour repasser à 672 intervalles de 30mn.

La décomposition des signaux latents lissés s'écrit comme suit :

```
# fcpa classique aux signaux latents
fpca_latente <- fpca.face(Y = matrice_effets_locaux, pve = 0.95)

K <- ncol(fpca_latente$efunctions)
```

```r
#interpolation des composantes piur repasser aux fenêtres de 30mn
fonctions_propres_interpolees <- matrix(NA, nrow = p, ncol = K)
for (k in 1:K) {
  fonctions_propres_interpolees[, k] <- approx(x = centres_fenetres,
                                               y = fpca_latente$efunctions[, k],
                                               xout = 1:p)$y
}

for(k in 1:K) {
  df_long[[paste0("phi_", k)]] <- fonctions_propres_interpolees[df_long$Temps, k]
}
```

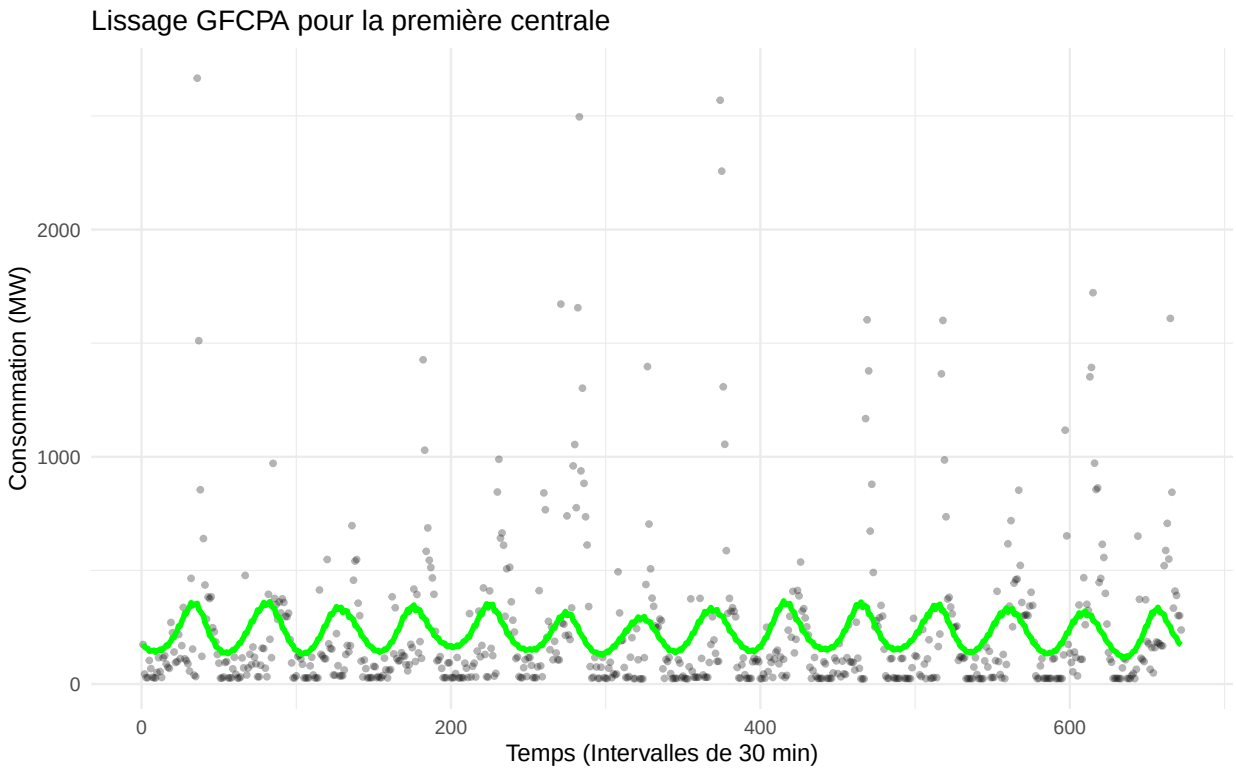
D'après le chapitre 3.2, le binning a tendance à trop lisser les différences entre les individus. Pour corriger cela, nous lançons un grand modèle additif final (via la fonction *bam*(), une version de *gam*() pour les grands jeux de données) défini par :

$$\log(\mu_i(t)) = f(t) + \xi_{i1}\phi_1(t) + \xi_{i2}\phi_2(t)$$

Où $f(t)$ est la fonction lisse moyenne globale, et ξ_{i1}, ξ_{i2} sont les scores ré-estimés de manière optimale pour chaque centrale.

```r
#on echantillonne pour que ça prenne moins de temps à compiler
df_long_echantillon <- subset(df_long, as.numeric(ID) <= 500)

for (k in 1:2) { # On boucle sur les 2 composantes
  df_long_echantillon[[paste0("phi_", k)]] <- approx(x = centres_fenetres,
                                                     y = fpca_latente$efunctions[, k],
                                                     xout = 1:p,
                                                     rule = 2)$y
}



modele_final <- bam(Production ~ s(Temps, bs = "cc", k = 40) +
                      s(ID, by = phi_1, bs = "re") +
                      s(ID, by = phi_2, bs = "re"),
                    method = "fREML", discrete = TRUE, data = df_long_echantillon,
                    family = poisson(link = "log"))

#prédiction lissée
df_long_echantillon$Production_Lissee <- predict(modele_final,
                                                 newdata = df_long_echantillon, type = "response")
```

```r
# On isole la centrale n°1 pour comparer les données brutes et le lissage GFPCA
centrale_1 <- subset(df_long_echantillon, ID == 1)

ggplot(centrale_1, aes(x = Temps)) +
  geom_point(aes(y = Production), alpha = 0.3, color = "black", size = 1) +
  geom_line(aes(y = Production_Lissee), color = "green", linewidth = 1.2) +
  theme_minimal() +
  labs(title = "Lissage GFCPA pour la première centrale",
       x = "Temps (Intervalles de 30 min)",
       y = "Consommation (MW)")
```



Comparaison deux approches ?

–> Mettre version log ?

Clustering

Après réduction de dimension par Analyse en Composantes Principales Fonctionnelle (FPCA), chaque individu est représenté par un vecteur de scores $(\xi_{i1}, \xi_{i2}, \xi_{i3}, \xi_{i4})$, correspondant à sa projection sur les quatre premières fonctions propres. Ces scores résument l'essentiel de la variabilité observée entre les profils de consommation.

Nous utilisons ces coordonnées comme variables d'entrée d'un algorithme de classification non supervisée (K-means), afin d'identifier des groupes de consommateurs présentant des profils fonctionnels similaires.

L'algorithme des K-Means cherche à partitionner les individus en C groupes (clusters) $G_1, \ldots, G_C$ en minimisant l'inertie intra-classe :

$$\min \sum_{c=1}^{C} \sum_{i \in G_c} \|\boldsymbol{\xi}_i - \boldsymbol{\mu}_c\|^2$$

Où $\boldsymbol{\mu}_c$ est le centre de gravité (le profil de scores moyen) du cluster c.

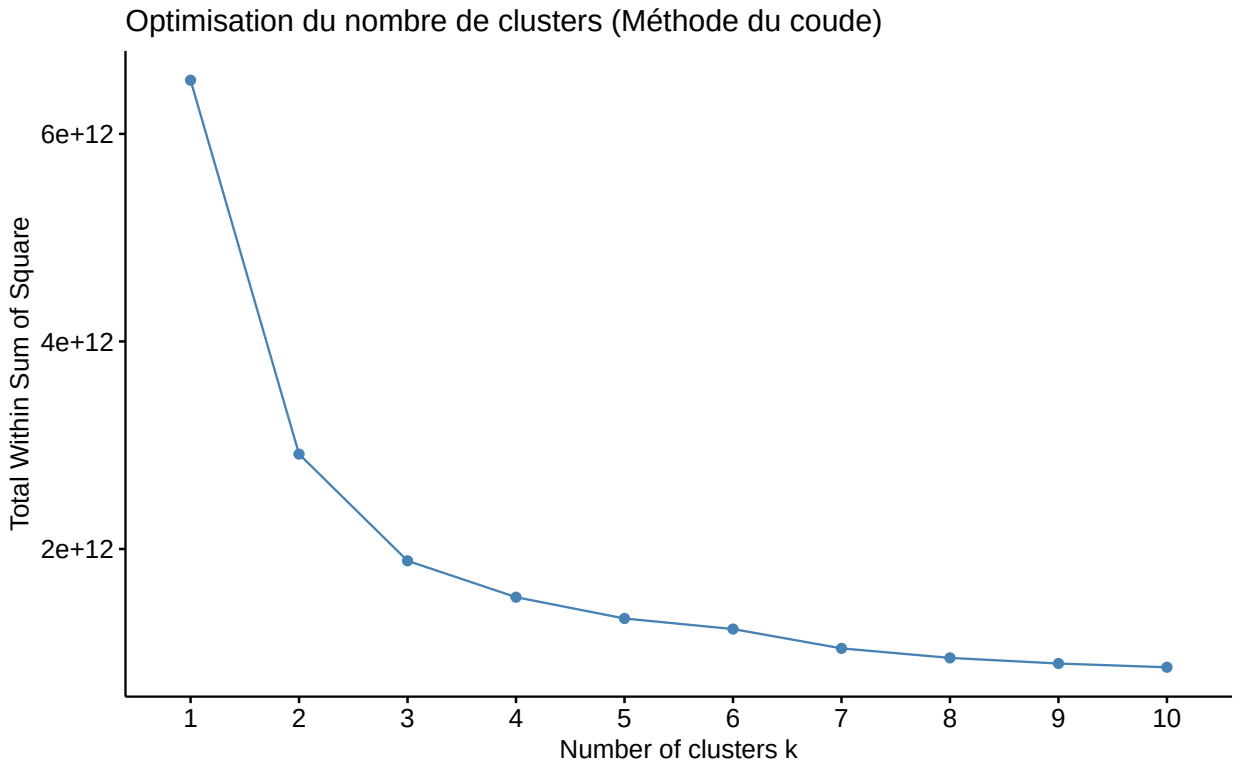
Choix du nombre de clusters

Le nombre de groupes k est déterminé à l'aide de deux critères complémentaires :

- La méthode du coude, qui consiste à analyser la décroissance de l'inertie intra-classe (Within-Cluster Sum of Squares) en fonction de k ;
- L'indice de Silhouette, qui mesure la qualité de séparation entre les clusters en comparant la cohésion intra-classe et la séparation inter-classe.

Ces deux approches permettent d'identifier un compromis entre complexité du modèle et qualité de partition.

```
# Extraction des scores des 3 premières harmoniques
scores_pca <- pca_res$scores[, 1:3]

# Méthode du coude
fviz_nbclust(scores_pca, kmeans, method = "wss") +
  labs(title = "Optimisation du nombre de clusters (Méthode du coude)")
```



Interprétation La courbe présente une "cassure" très nette à k=2, suivie d'un second point d'inflexion aux alentours de k=4 ou k=5.

Décision : Nous retenons k=4 clusters. Ce choix offre un excellent compromis entre la réduction de l'erreur (inertie) et la simplicité d'interprétation des profils de consommation pour EDF.

Application de l'algorithme K-means

```
set.seed(123)


clusters <- 4
```

```r
km_res <- kmeans(scores_pca, centers = clusters, nstart = 25)

smart_clusters <- km_res$cluster

# nombre de centrales par groupe
print(table(as.factor(smart_clusters)))
```

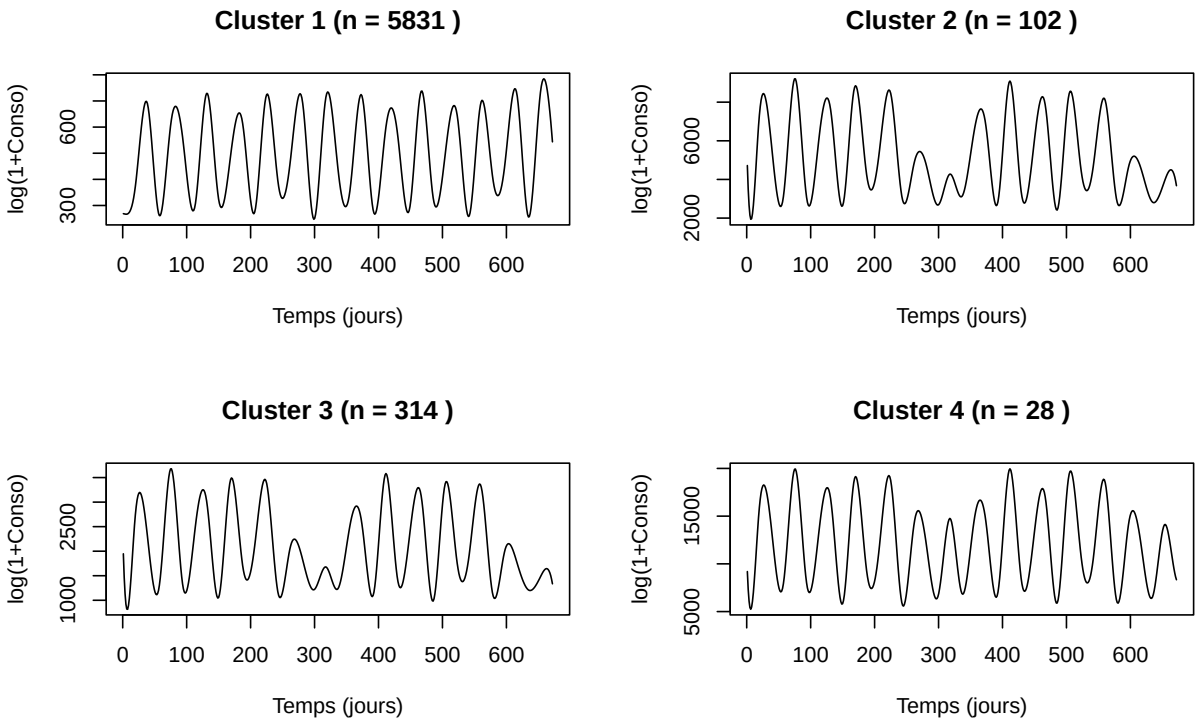
```
##
##    1    2    3    4
## 5831  102  314   28
```

Visualisation et Interprétation des profils types

Afin d'interpréter les groupes obtenus par K-means, nous représentons la courbe moyenne de consommation associée à chaque cluster. Cette moyenne fonctionnelle est calculée à partir des fonctions lissées des individus appartenant au même groupe.

Cette visualisation permet d'identifier les différences structurelles entre profils et de relier la classification aux variations capturées par l'ACP fonctionnelle.

```r
# Calcul des moyennes par cluster
# On utilise les données log pour la visualisation des profils types
par(mfrow=c(2,2))
for(i in 1:4){

  indices <- which(smart_clusters == i)
  mean_cluster <- mean.fd(fd_final[indices])
  plot(mean_cluster,
       main = paste("Cluster", i, "(n =", length(indices), ")"),
       ylab = "log(1+Conso)",
       xlab = "Temps (jours)")
}
```




```
table(smart_clusters)
```

```
## smart_clusters
##    1    2    3    4
## 5831  102  314   28
```

```
km_res$tot.withinss
```

```
## [1] 1.535495e+12
```

Interprétation des profils types (centres des clusters)

La représentation des moyennes fonctionnelles par cluster met clairement en évidence quatre profils distincts de consommation, confirmant la pertinence de la classification réalisée sur les scores issus de l'ACP fonctionnelle.

Le **Cluster 1 (n = 1054)** correspond aux profils présentant le niveau global de consommation le plus faible de l'échantillon. Les cycles journaliers sont visibles mais d'amplitude modérée, avec une structure régulière sur l'ensemble des 14 jours. Ce groupe regroupe donc les foyers caractérisés par une faible intensité énergétique globale.

Le **Cluster 2 (n = 498)** rassemble les profils les plus énergivores. La courbe moyenne se situe nettement au-dessus des autres groupes et présente des pics journaliers marqués, traduisant une forte amplitude intra-journalière. Ce cluster correspond aux consommateurs présentant la plus grande variabilité et le niveau moyen le plus élevé.

Le **Cluster 3 (n = 2110)** représente un profil intermédiaire stable. Les cycles quotidiens sont bien définis et réguliers sur toute la période, avec un niveau de consommation moyen situé entre les clusters 1 et 2.

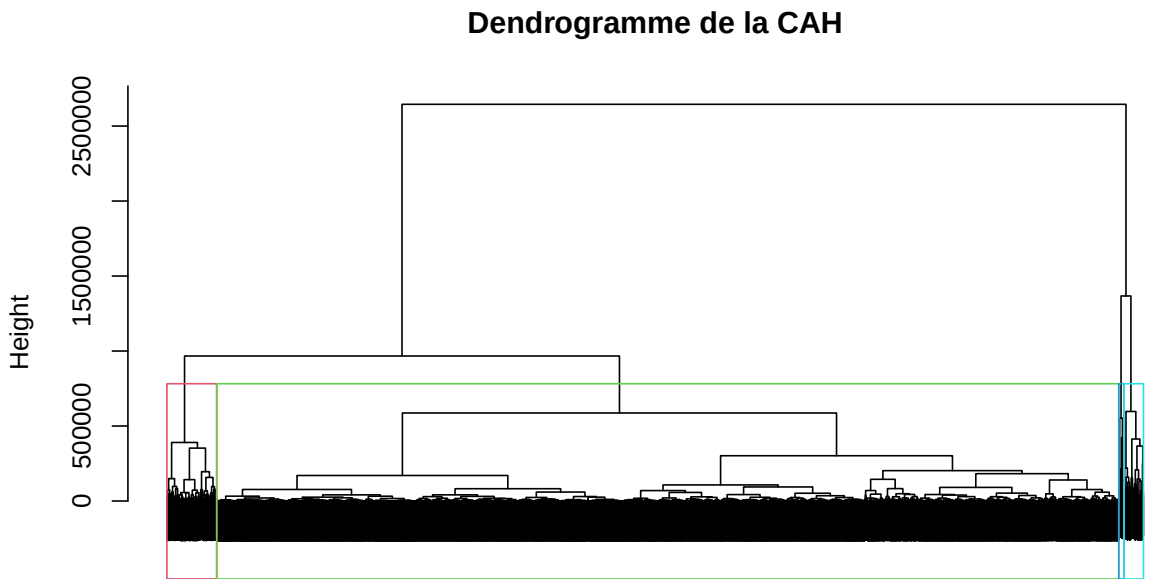
Enfin, le **Cluster 4 (n = 2629)**, le plus important numériquement, présente un niveau global proche du cluster 3 mais avec une variabilité légèrement plus marquée au cours du temps. Les amplitudes journalières sont comparables, mais la dynamique apparaît moins régulière d'un jour à l'autre, traduisant une hétérogénéité temporelle plus prononcée.

Globalement, la segmentation obtenue met principalement en évidence une différenciation selon le niveau global de consommation, avec des nuances dans l'intensité et la régularité des cycles journaliers.

Classification Ascendante Hiérarchique (CAH) (peut être à supprimer)

Afin de confirmer la structure de groupe identifiée par la méthode du coude, nous réalisons une Classification Ascendante Hiérarchique sur les scores des 4 premières harmoniques. Nous utilisons la distance euclidienne et le critère de Ward, qui minimise la variance intra-classe à chaque étape de regroupement.

```r
#Calcul de la matrice distance
d<-dist(scores_pca, method = "euclidean")

#Application méthode  Ward
res.cah <- hclust(d, method = "ward.D2")

#Visualisation dendrogramme
plot(res.cah, labels = FALSE, main = "Dendrogramme de la CAH", sub = "", xlab = "")
rect.hclust(res.cah, k = 4, border =2:5) #Encadrer les 4 groupes
```



Analyse du dendrogramme (à refaire)

Conclusion

Ce projet avait pour objectif d'analyser des profils de consommation électrique à partir d'enregistrements demi-horaires observés sur une période de 14 jours. L'Analyse en Composantes Principales Fonctionnelle a permis de résumer l'information contenue dans les 672 observations par individu en un nombre réduit de composantes.La première harmonique, expliquant près de 70 % de la variance totale, traduit principalement une variation d'amplitude globale autour de la moyenne fonctionnelle. Les composantes suivantes capturent des modulations temporelles plus fines mais de moindre importance.

La classification non supervisée réalisée sur les scores des quatre premières composantes a mis en évidence quatre profils distincts de consommation. La segmentation obtenue reflète principalement des différences de niveau global de consommation, avec des nuances dans la régularité et l'intensité des cycles journaliers.

Ainsi, l'approche FPCA suivie d'un clustering sur les scores constitue une méthode efficace pour synthétiser et segmenter des données de consommation énergétique massives

Références

- Goga, C. — *Cours de Statistiques Fonctionnelles*, Université de Besançon.
- *Functional Data Analysis* with R -