

Analyse des données de consommations d'énergie en Inde

Environnement de travail

```
library(tidyverse)
library(ggplot2)
library(readxl)
library(tidyr)
library(fda)
```

I. Notre base de données

Notre étude porte sur l'analyse de la consommation d'énergie dans les 33 Etats d'Inde. Il est à noter que l'Inde est le troisième producteur et le troisième consommateur d'électricité au monde.

Nous avons choisi ce sujet car il est au coeur des problématiques sociétales actuelles. En effet, dans les politiques actuelles, la question de l'environnement a une part importante. Notamment, la gestion de l'énergie pour les pays avec une forte croissance de population comme l'Inde. Notre étude peut mettre en avant les comportements de consommation des différents Etats et inciter à réfléchir sur la problématique de l'optimisation de cette consommation.

Ce projet nous permet d'appliquer nos connaissances à des thématiques concrètes dans le monde professionnel.

I.A. Importation des données

Dans un premier temps, nous allons importer les données.

```
data <- read.csv("long_data_2.csv", stringsAsFactors = T, sep = ";")

colnames(data) <- c("Etat", "Regions", "latitude", "longitude", "Date", "Consommation") # renommage des

data <- data[,-c(3,4)]
summary(data)
```

| ## | Etat | Regions | Date | Consommation |
|----|-------------------|---------|----------|----------------------|
| ## | Andhra Pradesh | : 498 | ER :2490 | 01/01/2020 00:00: 33 |
| ## | Arunachal Pradesh | : 498 | NER:3486 | 01/02/2020 00:00: 33 |
| ## | Assam | : 498 | NR :4482 | 01/03/2020 00:00: 33 |
| ## | Bihar | : 498 | SR :2988 | 01/04/2020 00:00: 33 |
| ## | Chandigarh | : 498 | WR :2988 | 01/05/2020 00:00: 33 |
| ## | Chhattisgarh | : 498 | | 01/06/2020 00:00: 33 |
| ## | (Other) | :13446 | (Other) | :16236 |

```
# View(data)
```

```
dim(data)
```

```
## [1] 16434 4
```

Concernant notre base de données, elle décrit la consommation d'énergie dans **33 Etats** d'Inde entre 2019 et 2020.

Nous avons 16 434 lignes, soit **498 points de mesure** par pays (498×33). Les données sont journalières, exprimées en Mega Units.

Grâce au résumé statistique ci-dessus, nous pouvons voir que les valeurs de nos données de consommation d'énergie sont assez étendues : elles sont comprises entre 0.3 et 522.1 Mega Units. La consommation moyenne en Inde est de 103.1 Mega Units en comparaison avec la médiane qui est de 64.5 Mega Units. Ainsi, la moyenne est très influencée par les valeurs extrêmes de certains Etats de notre jeu de données.

```
data$Etat[which.max(data$Consommation)]
```

```
## [1] Maharashtra
## 33 Levels: Andhra Pradesh Arunachal Pradesh Assam Bihar ... West Bengal
```

```
data$Etat[which.min(data$Consommation)]
```

```
## [1] Sikkim
## 33 Levels: Andhra Pradesh Arunachal Pradesh Assam Bihar ... West Bengal
```

En effet, certaines valeurs journalières sont très faibles (inférieures à 1) alors que d'autres sont 500 fois plus élevées. Par exemple, le Maharashtra qui présente certaines valeurs très importantes en opposition avec le Sikkim.

I.B. Choix de la période étudiée

Nous allons transformer notre variable Date au format date pour poursuivre l'étude correctement.

```
data$Date <- as.Date(data$Date, format = "%d/%m/%Y %H:%M")
str(data) # Vérification
```

```
## 'data.frame': 16434 obs. of 4 variables:
## $ Etat : Factor w/ 33 levels "Andhra Pradesh",...: 25 11 26 7 31 32 12 13 5 6 ...
## $ Regions : Factor w/ 5 levels "ER","NER","NR",...: 3 3 3 3 3 3 3 3 5 ...
## $ Date : Date, format: "2019-01-02" "2019-01-02" ...
## $ Consommation: num 119.9 130.3 234.1 85.8 313.9 ...
```

Pour simplifier notre analyse, nous allons nous concentrer seulement sur l'année 2019 :

```
data <- data |>
  filter(Date <= as.Date("2020-01-02"))
```

On vérifie à présent que les données que l'on va analyser correspondent bien à la période choisie.

```
min(data$Date)
```

```
## [1] "2019-01-02"
```

```
max(data$Date)
```

```
## [1] "2020-01-02"
```

Notre étude porte sur la période du 02 Janvier 2019 au 02 Janvier 2020.

Nous avons pu observer que les données du 1er de chaque mois sont manquantes mais cela n'impactera pas notre analyse.

I.C. Normalisation des données

Au départ, nous avons voulu normaliser les données par le nombre d'habitants de chaque Etat pour que les consommations aient le même impact dans notre étude. Cependant, cette méthode s'est révélée problématique car nous avons obtenu des valeurs très proches de zéro, ce qui aurait compliqué les calculs.

Nous avons donc choisi de réduire les données de consommation, en soustrayant la moyenne, pour continuer notre analyse avec données moins disproportionnées.

```
moyenne_par_etat <- tapply(data$Consommation, data$Etat, mean)

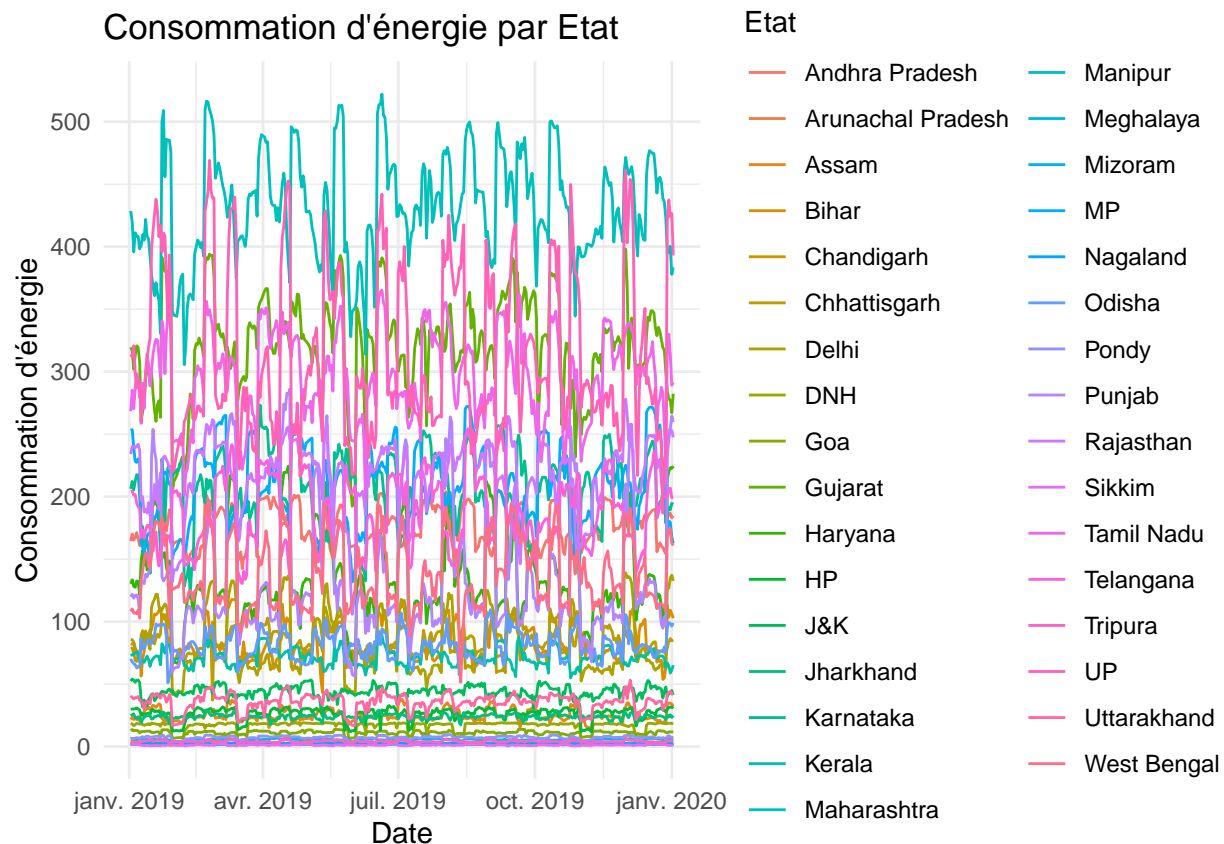
normalized_energy <- data
for(i in 1:nrow(normalized_energy)){
  normalized_energy$Consommation[i] <- normalized_energy$Consommation[i] - moyenne_par_etat[normalized_
}
head(normalized_energy)
```

Suite à cette normalisation, nous pouvons observer une diminution importante des valeurs. Mais nous obtenons des valeurs de consommations négatives, ce qui pose problème pour l'interprétation de nos analyses. De plus, mettre les données en valeur absolue n'aurait pas de sens dans notre contexte. Nous avons donc choisi travailler avec les valeurs initiales.

I.D. Représentations graphiques

Nous allons tout d'abord représenter simplement nos données pour pouvoir faire ressortir les informations importantes.

```
ggplot(data, aes(x = Date, y = Consommation, color = Etat)) +
  geom_line() +
  labs(x = "Date", y = "Consommation d'énergie", title = "Consommation d'énergie par Etat") +
  theme_minimal()
```



Ce graphique nous montre que nos données sont très irrégulières. Cependant, la lecture des valeurs est assez compliquée à cause du nombre important d'Etats et des fluctuations, c'est pourquoi la représentation selon les

régions d'Inde nous a paru pertinente.

[Représentation des données en différentes parties].{underline}

Nous avons fixé des seuils par rapport à la moyenne de consommations des Etats pour réaliser ces graphiques.

[Calcul de la moyenne par Etat].{underline}

On calcule d'abord la moyenne par Etat et on ajoute cette moyenne a un nouveau dataframe :

```
moyenne_par_etat <- tapply(data$Consommation, data$Etat, mean)
moyenne_par_etat <- data.frame(Etat = names(moyenne_par_etat), Moyenne = moyenne_par_etat)

data_complete <- merge(data, moyenne_par_etat, by = "Etat")
# knitr::kable(data_complete, caption = " Tableau des données avec la moyenne des consommations par Eta
```

[Séparation de notre jeu de données].{underline}

On sépare les données en 3 catégories d'Etats caractérisant leur consommation d'énergie :

```
data_inf <- data_complete |>
  filter(Moyenne <= 50)
# data_inf

data_moy <- data_complete |>
  filter(Moyenne > 50 & Moyenne < 200)
# data_moy

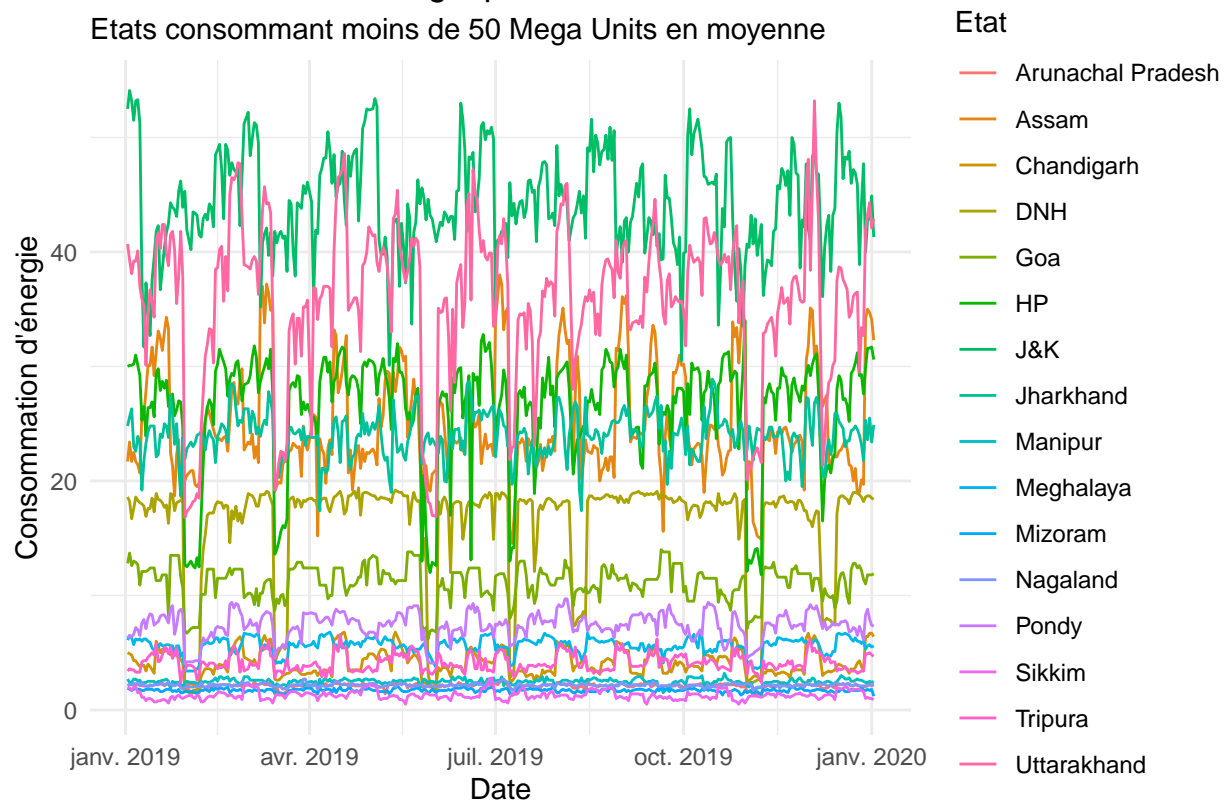
data_sup <- data_complete |>
  filter(Moyenne >= 200)
# data_sup
```

[Représentations graphiques].{underline}

```
ggplot(data_inf, aes(x = Date, y = Consommation, color = Etat)) +
  geom_line() +
  labs(x = "Date", y = "Consommation d'énergie", title = "Consommation d'énergie par Etat", subtitle = "Inde", theme_minimal())
```

Consommation d'énergie par Etat

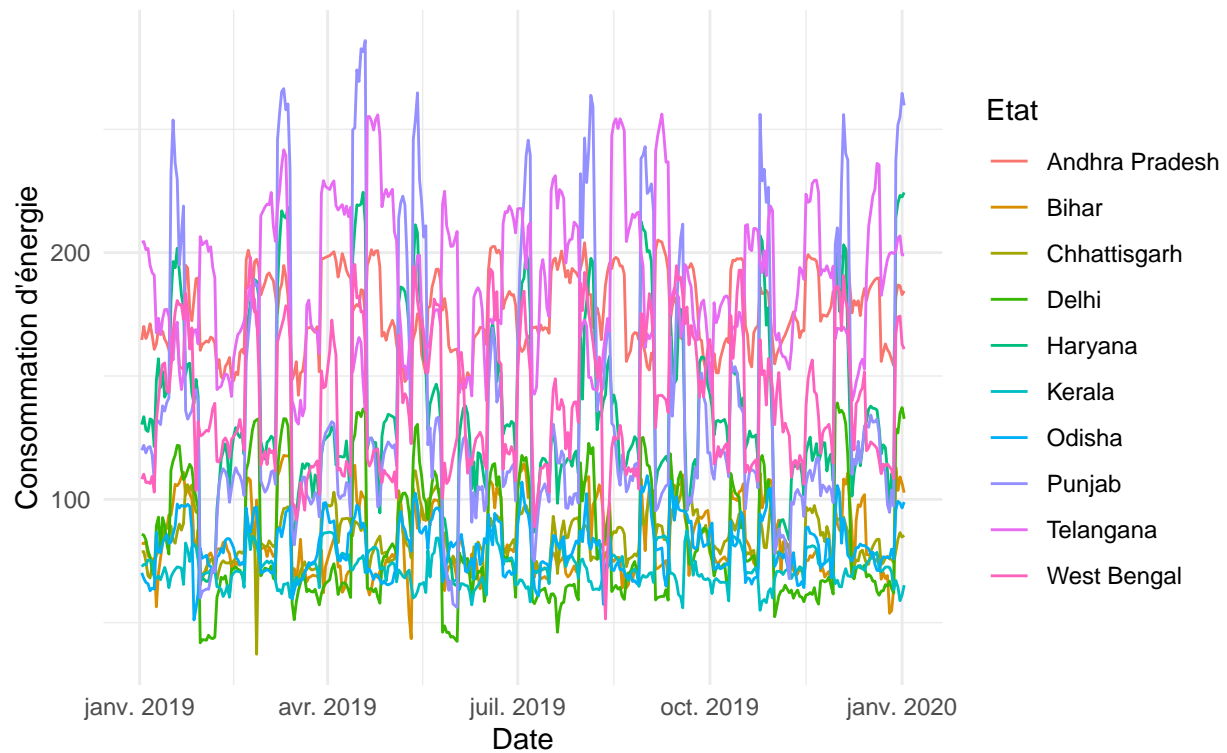
Etats consommant moins de 50 Mega Units en moyenne



```
ggplot(data_moy, aes(x = Date, y = Consommation, color = Etat)) +  
  geom_line() +  
  labs(x = "Date", y = "Consommation d'énergie", title = "Consommation d'énergie par Etat", subtitle = "  
  theme_minimal()
```

Consommation d'énergie par Etat

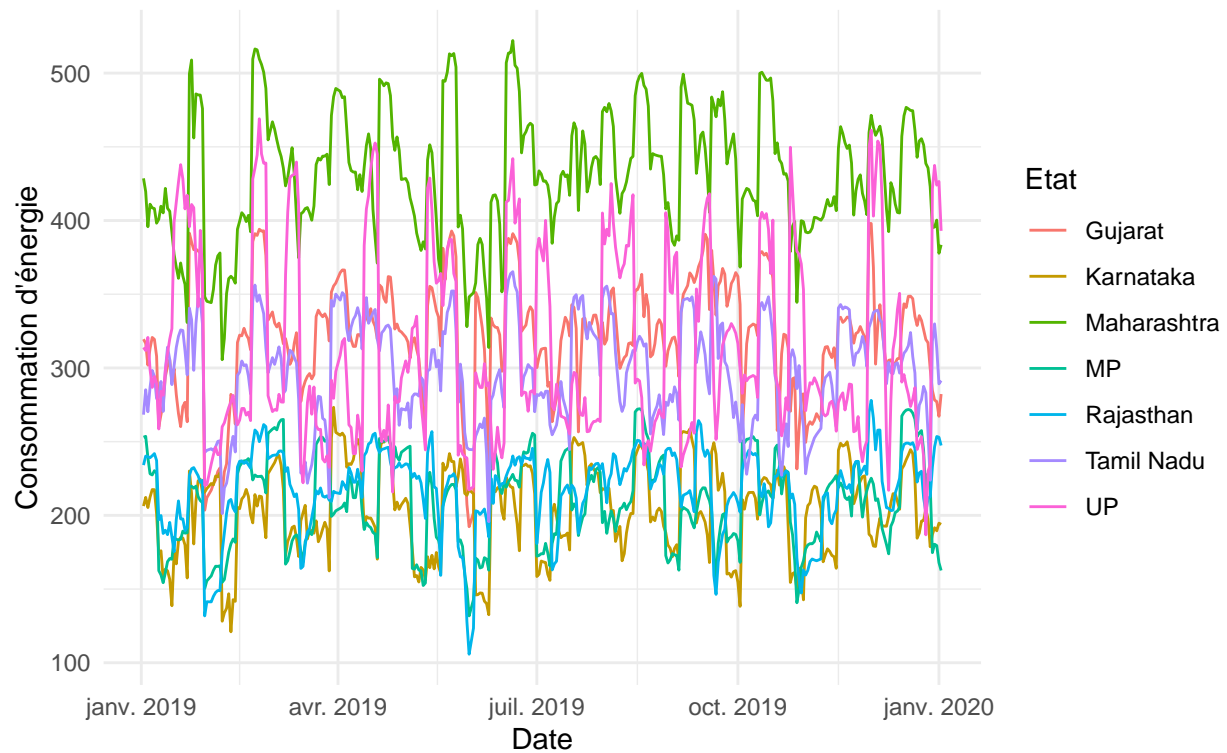
Etats consommant entre 50 et 200 Mega Units en moyenne



```
ggplot(data_sup, aes(x = Date, y = Consommation, color = Etat)) +  
  geom_line() +  
  labs(x = "Date", y = "Consommation d'énergie", title = "Consommation d'énergie par Etat", subtitle = "  
  theme_minimal()
```

Consommation d'énergie par Etat

Etats consommant plus de 200 Mega Units en moyenne



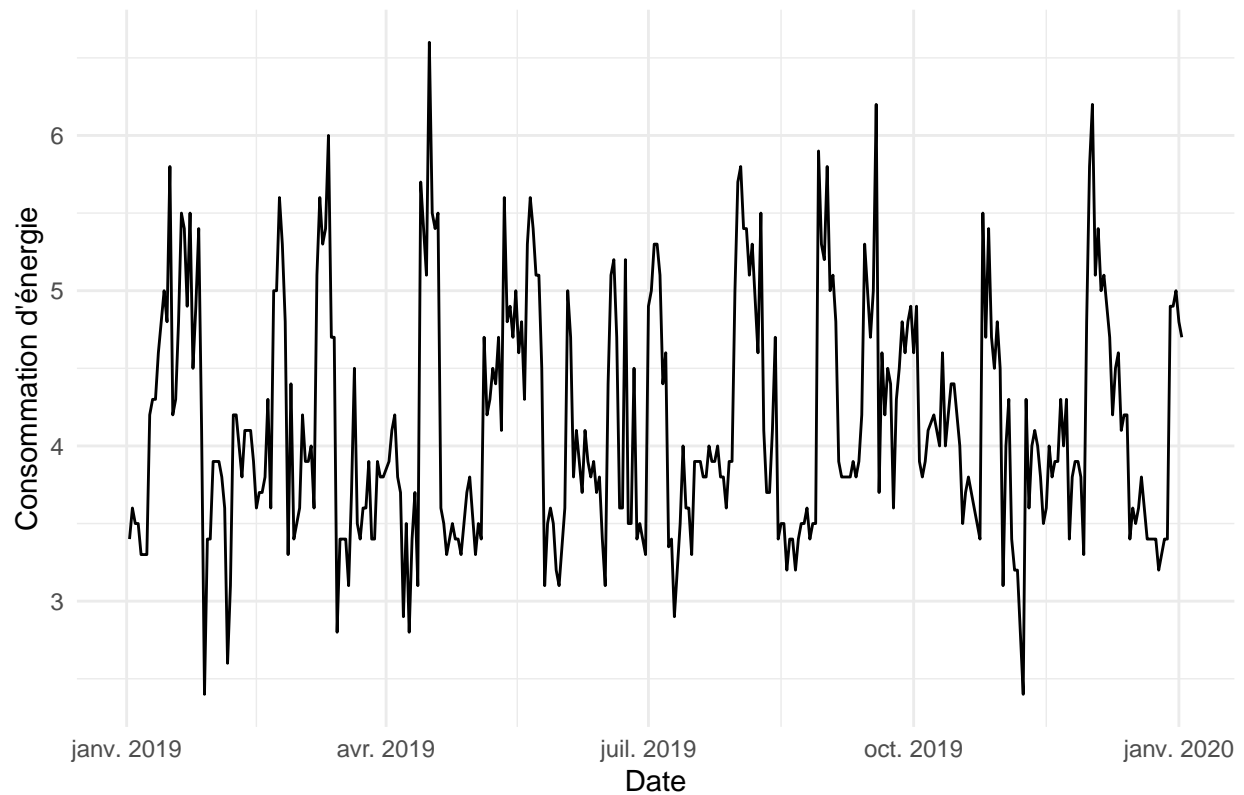
Grâce à ces représentations, il est plus facile d'observer et d'étudier l'allure de nos données.

Pour la plupart des Etats, leur consommation varie beaucoup tout au long de l'année, et, en quelques jours leur consommation peut doubler ou bien diminuer subitement.

Cependant certains Etats comme le Tripura possède une consommation très faible, et peu de variations dans son signal sont observées :

```
data |>
  filter(Etat=="Tripura") |>
  ggplot(aes(x = Date, y = Consommation)) +
  geom_line() +
  labs(x = "Date", y = "Consommation d'énergie", title = "Consommation d'énergie du Tripura") +
  theme_minimal()
```

Consommation d'énergie du Tripura



Sa consommation varie entre 2.4 et 6.6 mega Units.

```
min(data$Consommation[data$Etat=="Tripura"])
```

```
## [1] 2.4
```

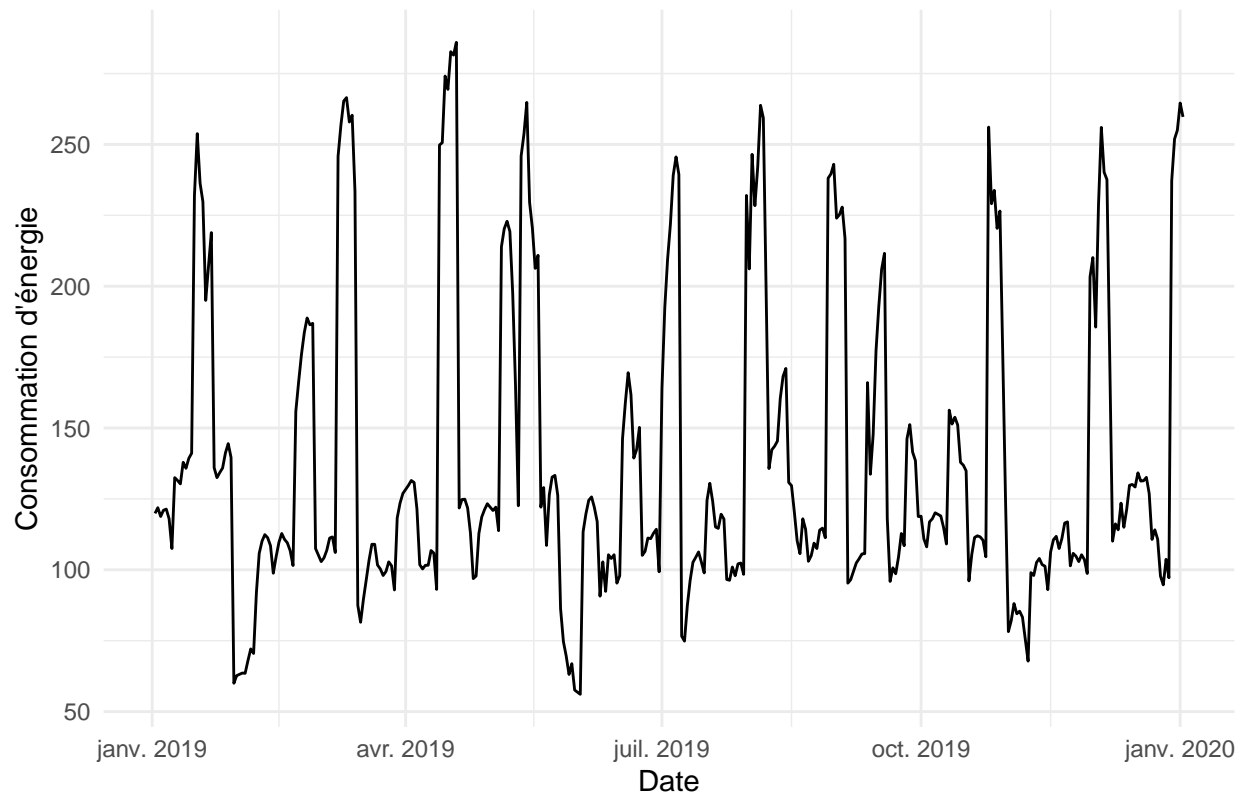
```
max(data$Consommation[data$Etat=="Tripura"])
```

```
## [1] 6.6
```

A l'opposition du Punjab qui lui possède une très grande variation dans sa consommation :

```
data |>
  filter(Etat=="Punjab") |>
  ggplot(aes(x = Date, y = Consommation)) +
  geom_line() +
  labs(x = "Date", y = "Consommation d'énergie", title = "Consommation d'énergie du Punjab") +
  theme_minimal()
```


Consommation d'énergie du Punjab



```
min(data$Consommation[data$Etat=="Punjab"])
```

```
## [1] 56.1
```

```
max(data$Consommation[data$Etat=="Punjab"])
```

```
## [1] 286
```

Tableau des moyennes de consommation par Etat

```
knitr::kable(tapply(data$Consommation, data$Etat, mean),
  "simple",
  caption = "Tableau des moyennes des consommation par Etat",
  col.names = "Moyenne")
```

Table 1: Tableau des moyennes des consommation par Etat

| | Moyenne |
|-------------------|------------|
| Andhra Pradesh | 175.891573 |
| Arunachal Pradesh | 2.108146 |
| Assam | 25.156601 |
| Bihar | 83.878652 |
| Chandigarh | 4.132584 |
| Chhattisgarh | 84.220646 |
| Delhi | 82.659831 |
| DNH | 16.415590 |
| Goa | 11.183427 |
| Gujarat | 321.564747 |

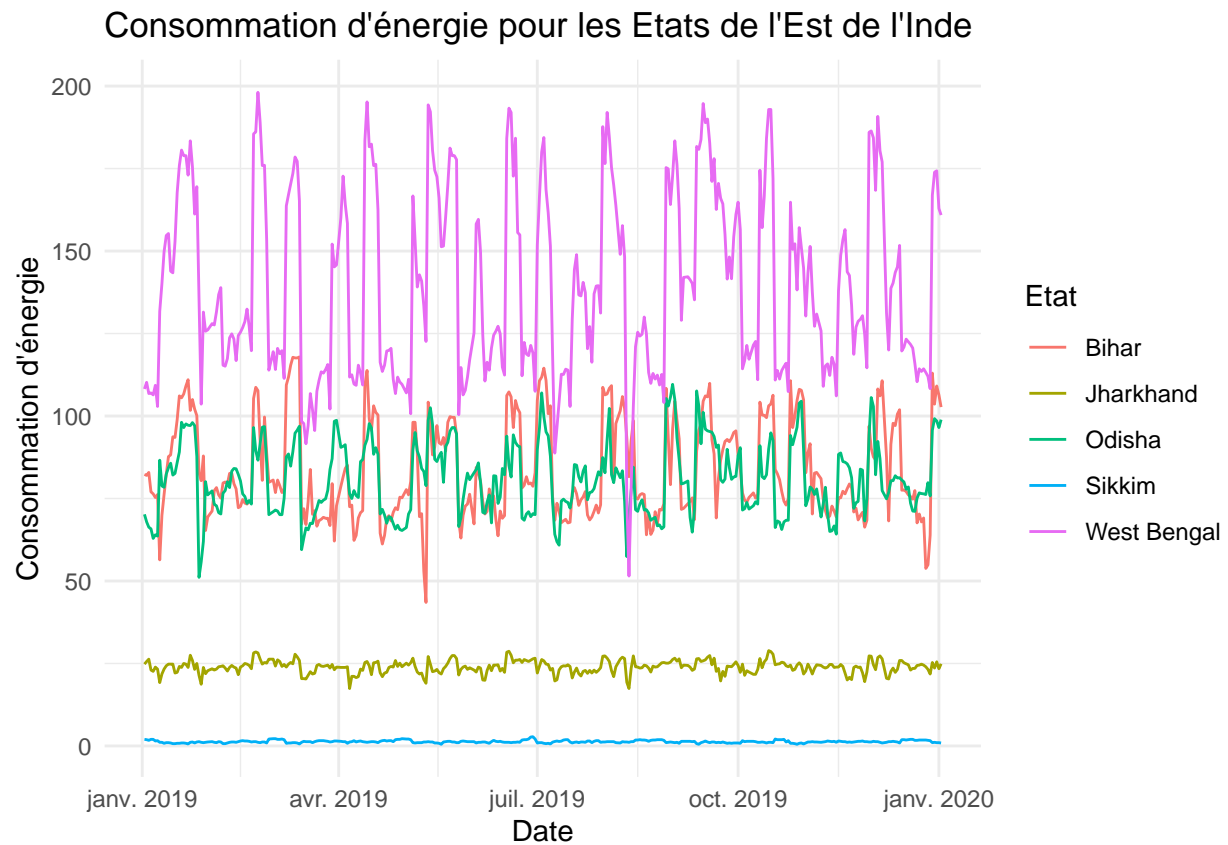
| | Moyenne |
|-------------|------------|
| Haryana | 137.518258 |
| HP | 26.582303 |
| J&K | 44.331180 |
| Jharkhand | 23.998596 |
| Karnataka | 203.602247 |
| Kerala | 71.824298 |
| Maharashtra | 431.561517 |
| Manipur | 2.494101 |
| Meghalaya | 5.635674 |
| Mizoram | 1.712500 |
| MP | 208.722331 |
| Nagaland | 2.167416 |
| Odisha | 80.974438 |
| Pondy | 7.407865 |
| Punjab | 139.669522 |
| Rajasthan | 218.367977 |
| Sikkim | 1.296348 |
| Tamil Nadu | 297.655758 |
| Telangana | 188.360534 |
| Tripura | 4.145084 |
| UP | 314.962781 |
| Uttarakhand | 36.104494 |
| West Bengal | 139.527107 |

Représentation selon la région

L'Inde est divisée en 5 régions : le Nord (NR), Nord-Est (NER), le Sud (SR), l'Est (ER) et l'Ouest (WR). Nous allons tracer les consommations en fonction des régions (une couleur par région)./ De la même manière que les représentations précédentes, nous ne pouvons représenter les régions sur le même graphique. Sans diviser la représentation, il est difficile de bien observer les différences entre les régions.

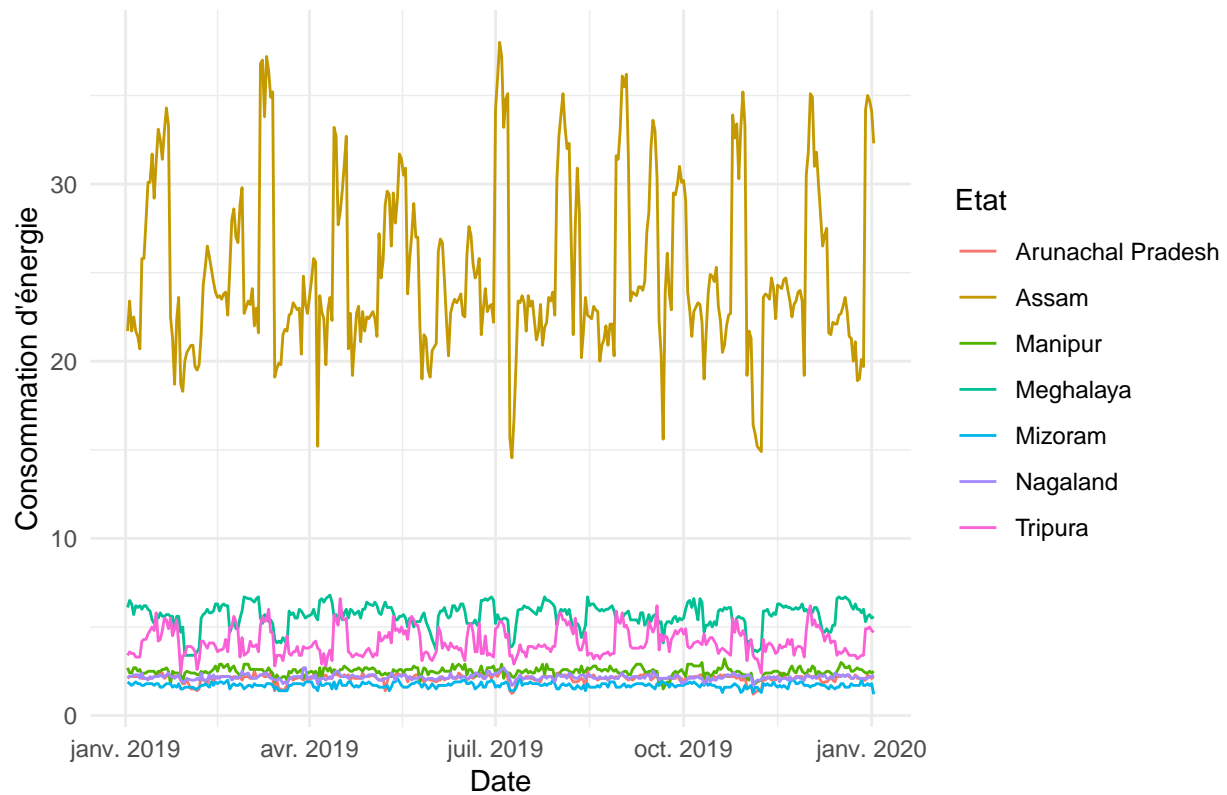
Nous décidons donc de représenter une région par graphique :

```
data |>
  filter(Regions=="ER") |>
  ggplot(aes(x = Date, y = Consommation, color = Etat)) +
  geom_line() +
  labs(x = "Date", y = "Consommation d'énergie", title = "Consommation d'énergie pour les Etats de l'Es",
  theme_minimal()
```



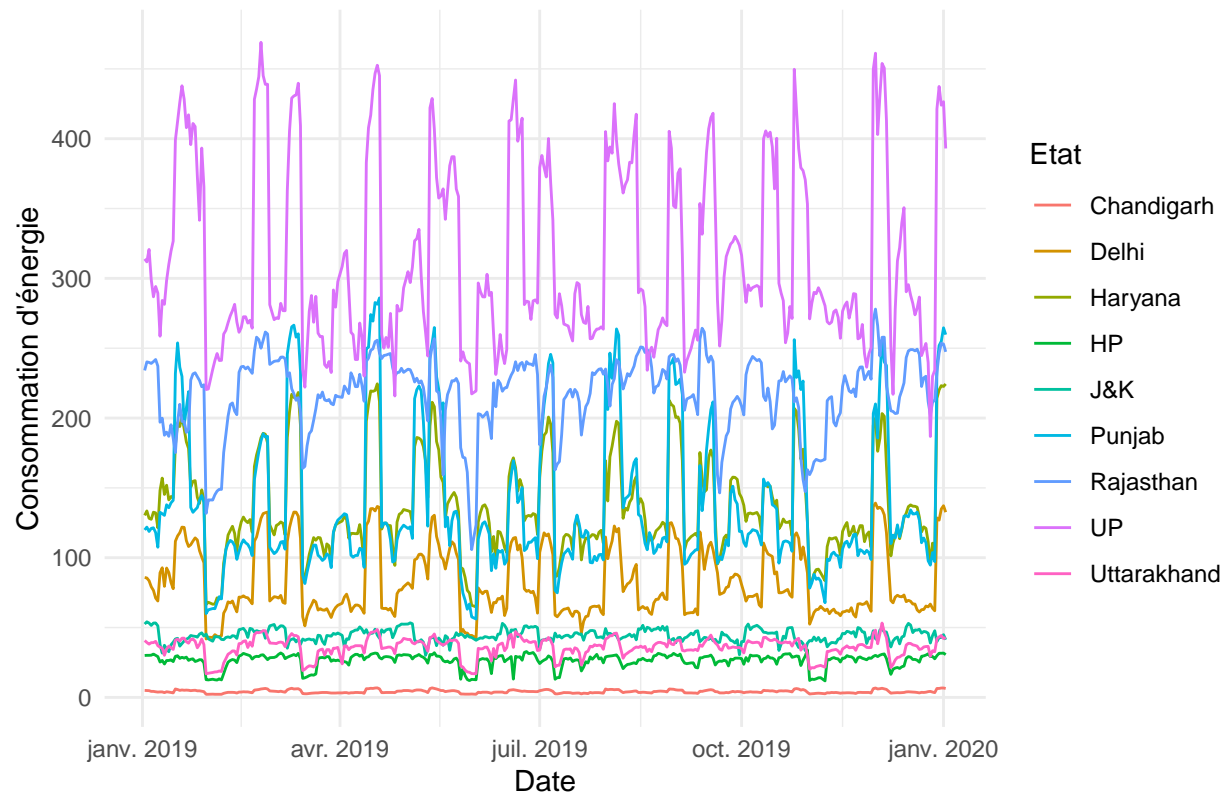
```
data |>
  filter(Regions=="NER") |>
  ggplot(aes(x = Date, y = Consommation,color = Etat)) +
  geom_line() +
  labs(x = "Date", y = "Consommation d'énergie", title = "Consommation d'énergie pour les Etats du Nord")
  theme_minimal()
```

Consommation d'énergie pour les Etats du Nord Est de l'Inde



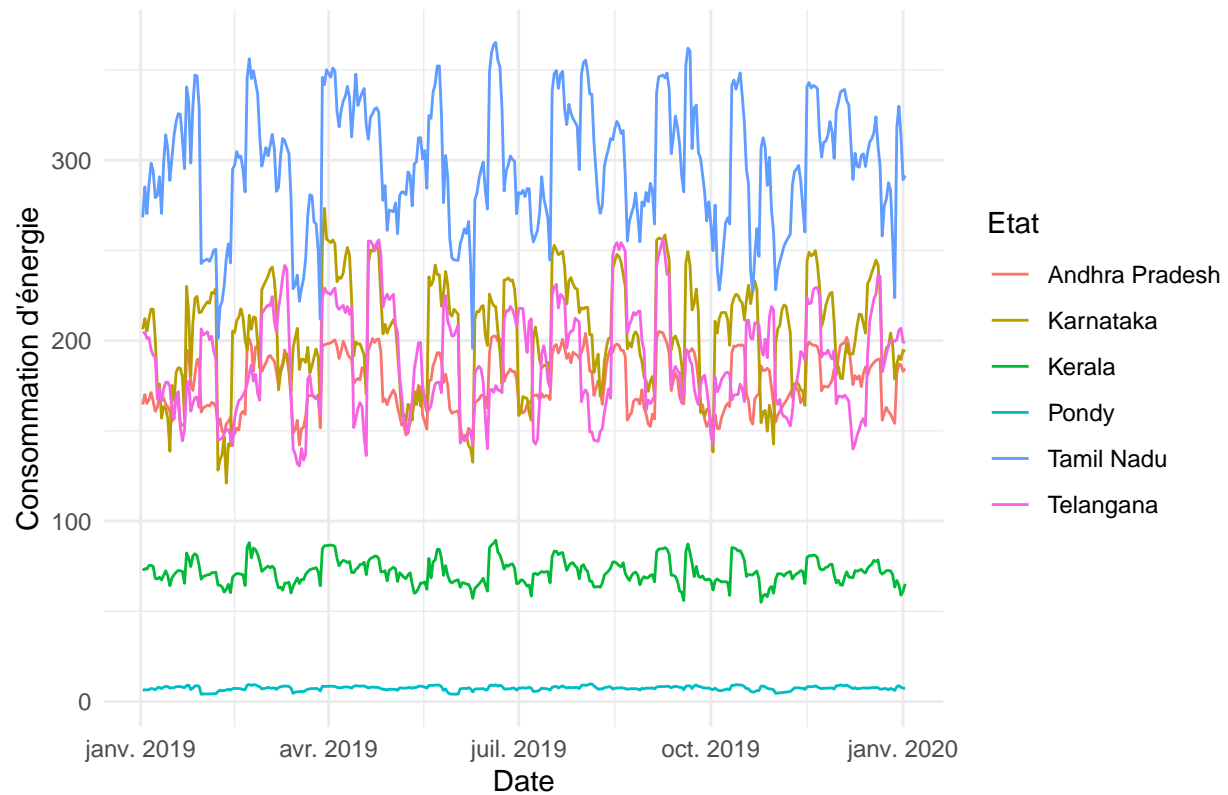
```
data |>
  filter(Regions=="NR") |>
  ggplot(aes(x = Date, y = Consommation,color = Etat)) +
  geom_line() +
  labs(x = "Date", y = "Consommation d'énergie", title = "Consommation d'énergie pour les Etats du Nord Est de l'Inde") +
  theme_minimal()
```

Consommation d'énergie pour les Etats du Nord de l'Inde



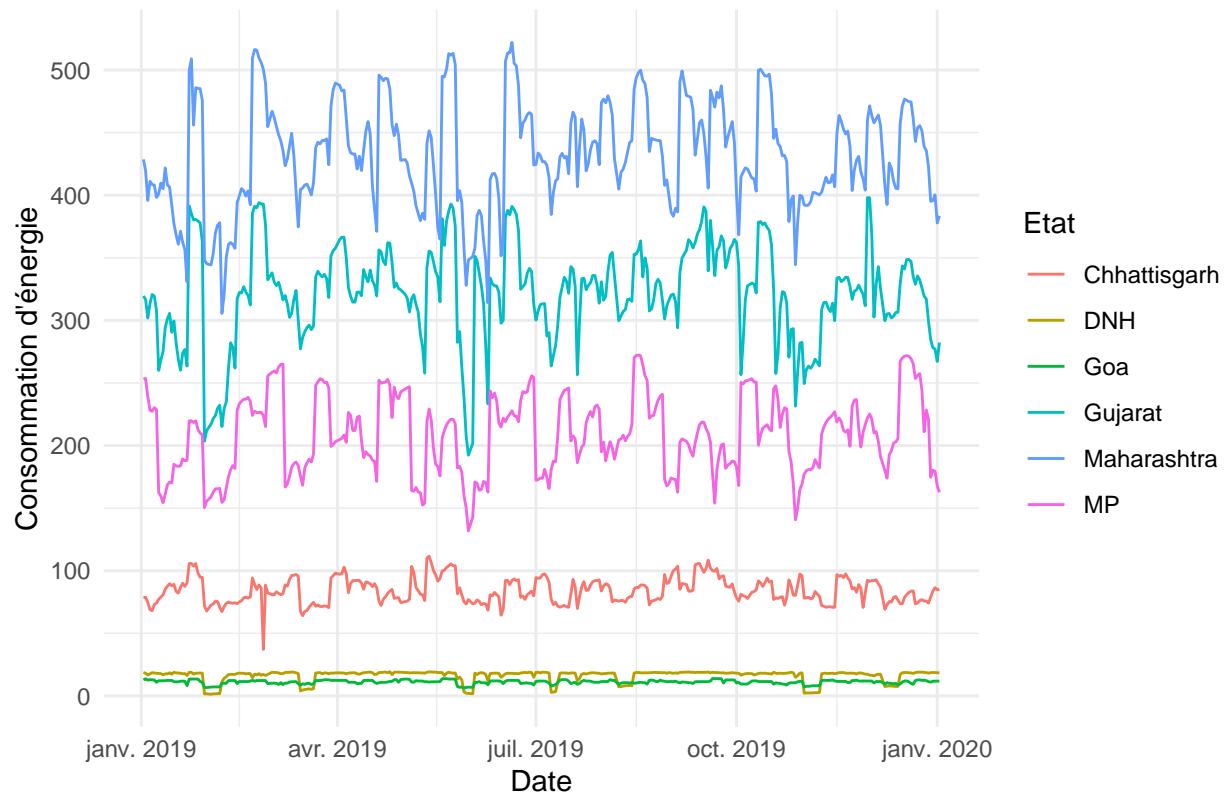
```
data |>
  filter(Regions=="SR") |>
  ggplot(aes(x = Date, y = Consommation,color = Etat)) +
  geom_line() +
  labs(x = "Date", y = "Consommation d'énergie", title = "Consommation d'énergie pour les Etats du Sud de l'Inde") +
  theme_minimal()
```

Consommation d'énergie pour les Etats du Sud de l'Inde



```
data |>
  filter(Regions=="WR") |>
  ggplot(aes(x = Date, y = Consommation,color = Etat)) +
  geom_line() +
  labs(x = "Date", y = "Consommation d'énergie", title = "Consommation d'énergie pour les Etats de l'Ou")
  theme_minimal()
```

Consommation d'énergie pour les Etats de l'Ouest de l'Inde



Au sein de chaque région de l'Inde, nous observons de forts contrastes de consommation d'énergie. Nous pouvons noter une différence d'échelle entre les graphiques. Par exemple, dans le Sud de l'Inde, Pondy a une faible consommation d'énergie (entre 4 et 10 Mega Units) à l'opposé de Tamil Nadu où celle-ci varie entre 200 et 370 Mega Units.

Il serait intéressant de consulter l'avis d'un économiste ou politicien pour analyser et comprendre ces grandes disparités au sein des régions de l'Inde.

Nous construisons le tableau des moyennes de consommation par région :

```
knitr::kable(tapply(data$Consommation, data$Regions, mean),
              "simple", col.names = "moyenne",
              caption = "Tableau des moyennes des consommation d'énergie par région")
```

Table 2: Tableau des moyennes des consommation d'énergie par région

| | moyenne |
|-----|------------|
| ER | 65.935028 |
| NER | 6.202789 |
| NR | 111.592104 |
| SR | 157.457046 |
| WR | 178.944710 |

Nous observons une hétérogénéité en termes de consommations entre les régions de l'Inde. En effet, le Nord-Est a une consommation très faible par rapport à l'Ouest. Peut-être qu'un expert pourrait aussi nous donner les causes de ces écarts de consommation.

II. Ondelettes

Puisque nos signaux sont irréguliers, notre choix de lissage s'est d'abord tourné vers les ondelettes car elles ont des bonnes propriétés de compressions et sont adaptées à ce type de données.

Pour pouvoir poursuivre cette étude, nous avons du transformer notre dataframe de départ pour avoir chaque Etat en colonne avec la valeur de consommation en ligne pour chaque point de mesure.

```
data_long <- data.frame("Etat" = data$Etat, "Date" = data$Date, "Consommation" = data$Consommation)

data_long <- data_long |>
  pivot_wider(names_from = "Etat", values_from = "Consommation")

data_long <- as.data.frame(data_long)
knitr::kable(head(data_long), "simple", caption = "Première ligne du tableau")
```

| Date | Punjab | Haryana | Rajasthan | Delhi | UP | Uttarakhand | HP | J&K | Chandigarh | Chhattisgarh | Goa |
|------------|--------|---------|-----------|-------|-------|-------------|------|------|------------|--------------|------|
| 2019-01-02 | 119.9 | 130.3 | 234.1 | 85.8 | 313.9 | 40.7 | 30.0 | 52.5 | 5.0 | 78.7 | 10.0 |
| 2019-01-03 | 121.9 | 133.5 | 240.2 | 85.5 | 311.8 | 39.3 | 30.1 | 54.1 | 4.9 | 78.8 | 10.0 |
| 2019-01-04 | 118.8 | 128.2 | 239.8 | 83.5 | 320.7 | 38.1 | 30.1 | 53.2 | 4.8 | 74.8 | 10.0 |
| 2019-01-05 | 121.0 | 127.5 | 239.1 | 79.2 | 299.0 | 39.2 | 30.2 | 51.5 | 4.3 | 69.0 | 10.0 |
| 2019-01-06 | 121.4 | 132.6 | 240.4 | 76.6 | 286.8 | 39.2 | 31.0 | 53.2 | 4.3 | 68.1 | 10.0 |
| 2019-01-07 | 118.0 | 132.1 | 241.9 | 71.1 | 294.2 | 40.1 | 30.1 | 53.3 | 4.0 | 73.1 | 10.0 |

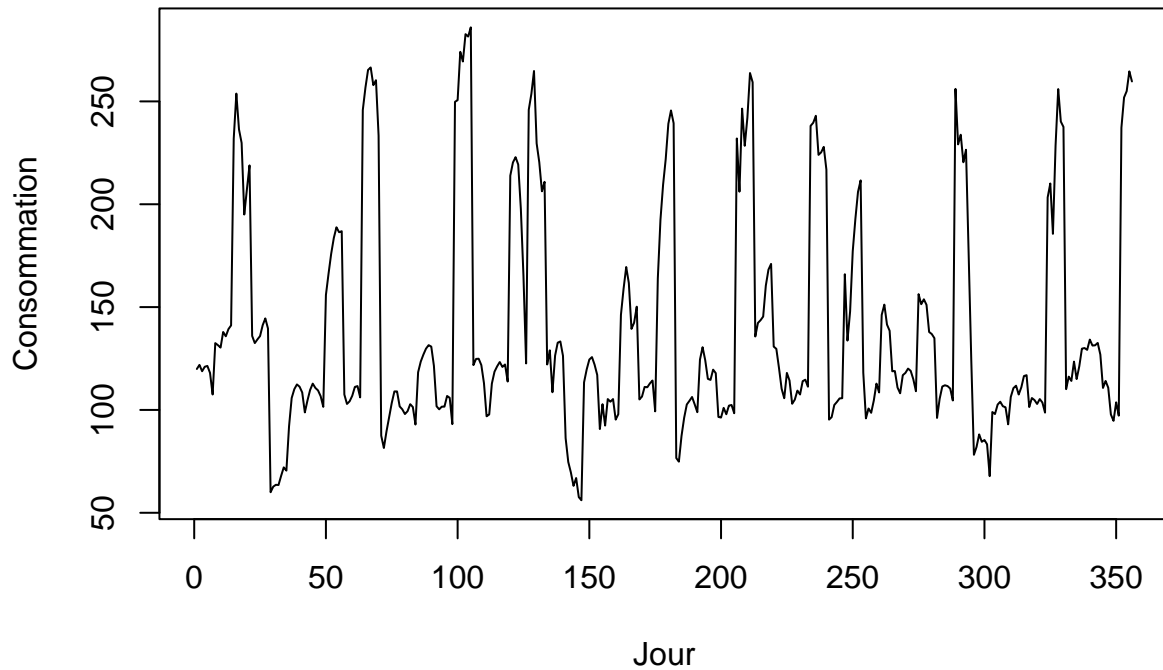
Le problème ici est que la longueur des courbes de consommation n'est pas de la forme 2^J . Ces courbes sont de longueur 356, on symétrise le signal à la fin pour atteindre une longueur de 512. On se ramènera à la fin au signal de départ de taille 356.

Nous représentons le signal pour un Etat (ici le Punjab) :

```
i = 2
y <- data_long[,i]
# y

plot(1:356,y,type="l",main="Signal de départ", ylab = "Consommation", xlab="Jour")
```


Signal de départ

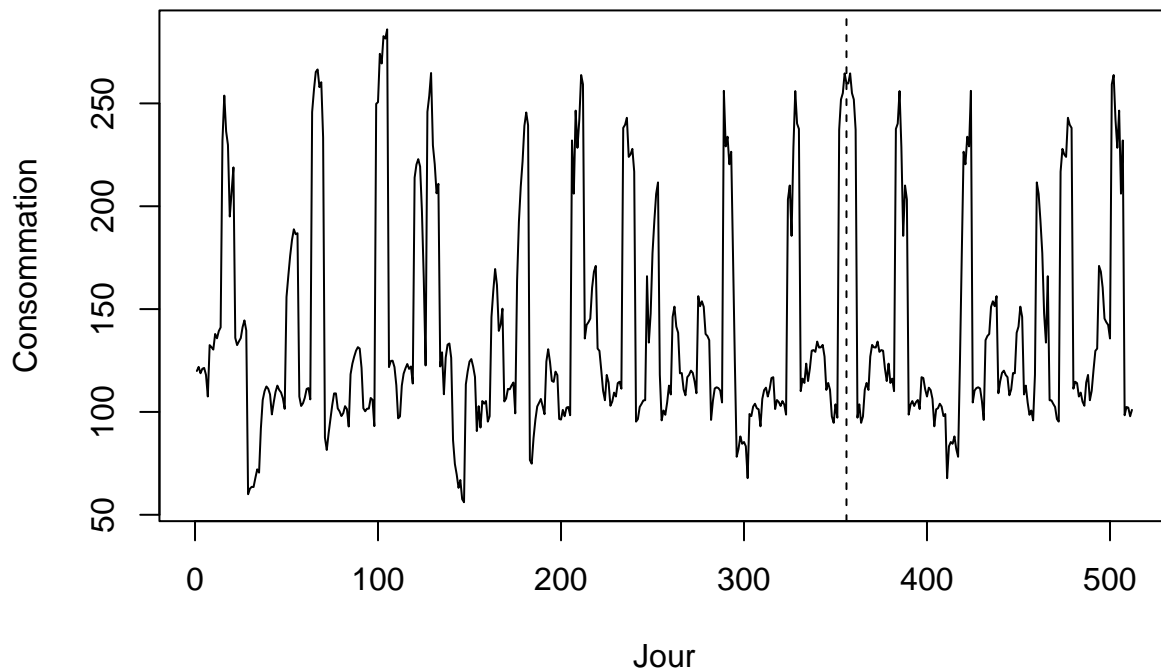


```
# Créer un signal symétrisé
ysym = c(y, rev(tail(y, n= 512-length(y)))) # on choisit 512 pour avoir une puissance de 2

# Tracer le signal symétrisé
plot(ysym, type = "l", main = "Signal symétrisé", ylab = "Consommation", xlab = "Jour")

# Ajouter une ligne verticale à la position
# axe de symétrie : 361 jours
abline(v = 356, lty = 2)
```

Signal symétrisé



La méthode des ondelettes est complexe et nous ne disposons pas des connaissances nécessaires pour faire une analyse très approfondie. De plus, il n'est pas possible de réaliser une ACP avec les ondelettes.

Nous nous tournons donc vers le lissage par moindres carrés pénalisés à l'aide des bases de splines pour continuer notre étude.

III. Lissage par moindres carrés pénalisés - Base de Spline

L'utilisation d'une base de spline pénalisée nous a semblé être adaptée à nos signaux. Effectivement, cette méthode présente une grande flexibilité de modélisation pour les données plus ou moins irrégulières. Cette approche permet de capturer des comportements non linéaires, de gérer les valeurs aberrantes, de réduire la dimension de l'espace des caractéristiques, et de s'adapter à diverses structures de données. La pénalisation va nous permet d'éviter un sur-ajustement de nos données.

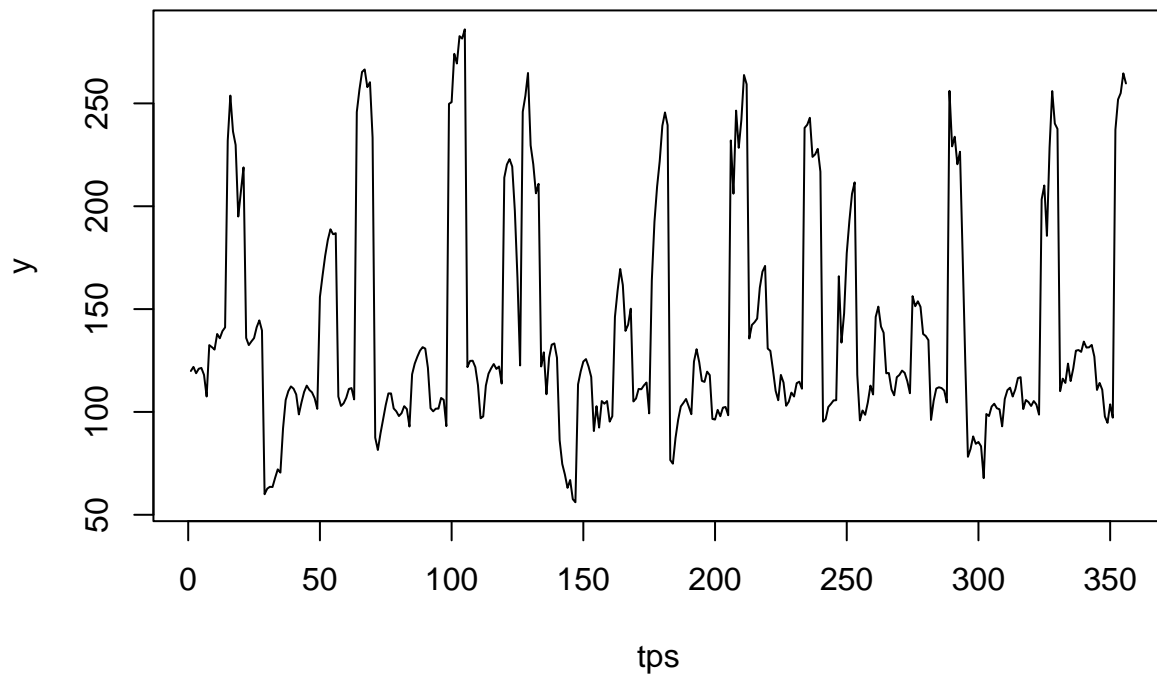
III.A. Lissage d'une trajectoire (pour un Etat)

Nous allons d'abord effectuer le lissage sur un seul Etat : le Punjab, pour ensuite généraliser à l'ensemble des Etats.

Représentation de la consommation d'énergie

```
i = 2
y <- data_long[, 2]
tps <- 1:356
plot(tps,y,type="l", main = "Consommation d'énergie du Punjab")
```

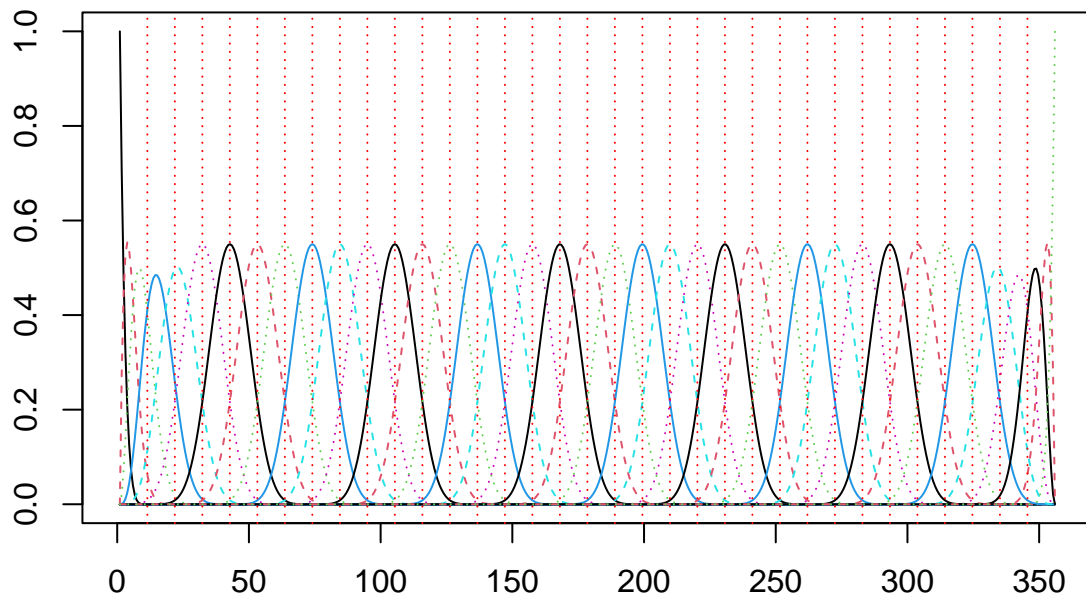
Consommation d'énergie du Punjab



Lissage par moindres carrés pénalisés sur la première courbe

```
splbasis = create.bspline.basis(rangeval = c(1,356), # 1 à 356 jours
                                norder=6,
                                breaks=seq(1,356,length=35))
# définition une base de B-spline entre 1 et 356 (les dates)
# ordre 6

plot(splbasis)
```



```
summary(splbasis)
```

```
##
## Basis object:
##
##   Type:   bspline
##
##   Range:  1 to 356
##
##   Number of basis functions: 39
```

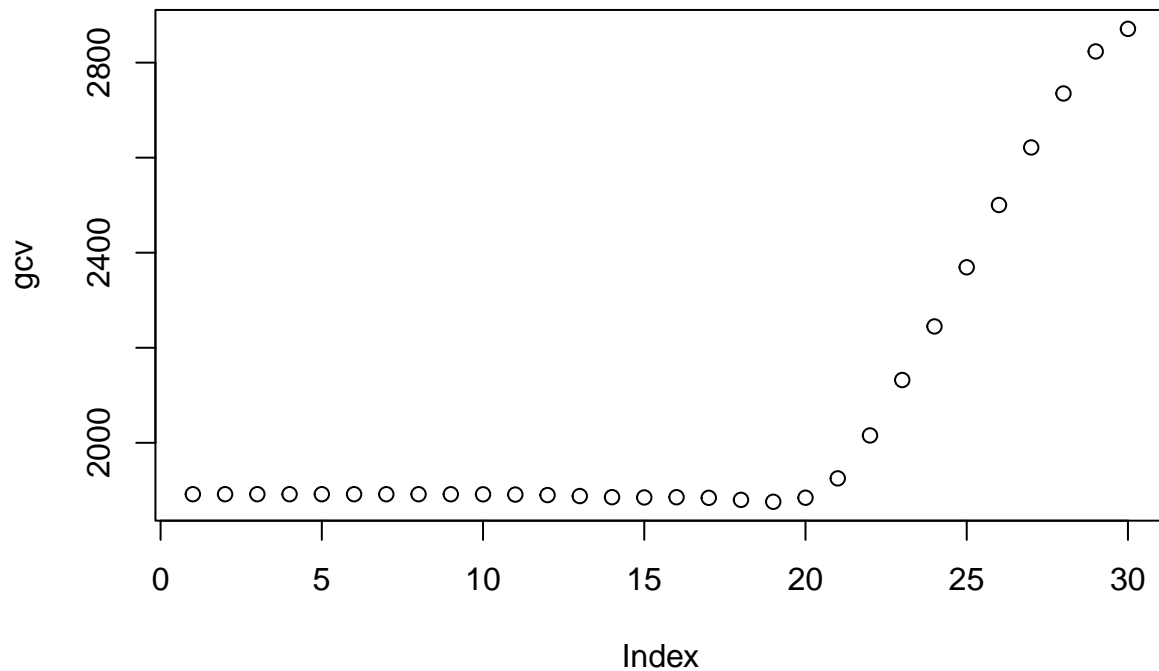
Il s'agit d'une base de type "bspline" ce qui signifie qu'elle utilise les fonctions B-splines pour modéliser les données. Dans cette base, il y a 39 fonctions de base, utilisées pour construire la représentation fonctionnelle des données.

La plage des données s'étend entre 1 et 356, correspondant à nos temps de mesure.

Nous allons rechercher la valeur de lambda optimale pour trouver le meilleur lissage de nos données.

```
i = 2
y = data_long[,i]
gcv = 1:30 # grille de lambda
for (i in 1:30){
  lambda = exp(i-10)
  fdpar= fdPar(splbasis,Lfdobj =4,lambda=lambda)
  smoothdata = smooth.basis(tps,y,fdParobj = fdpar)
  gcv[i] = smoothdata$gcv
}
```

```
plot(gcv)
```



```
which.min(gcv)
```

```
## [1] 19
```

Nous obtenons une valeur de lambda égale à 19 pour l'Etat du Punjab.

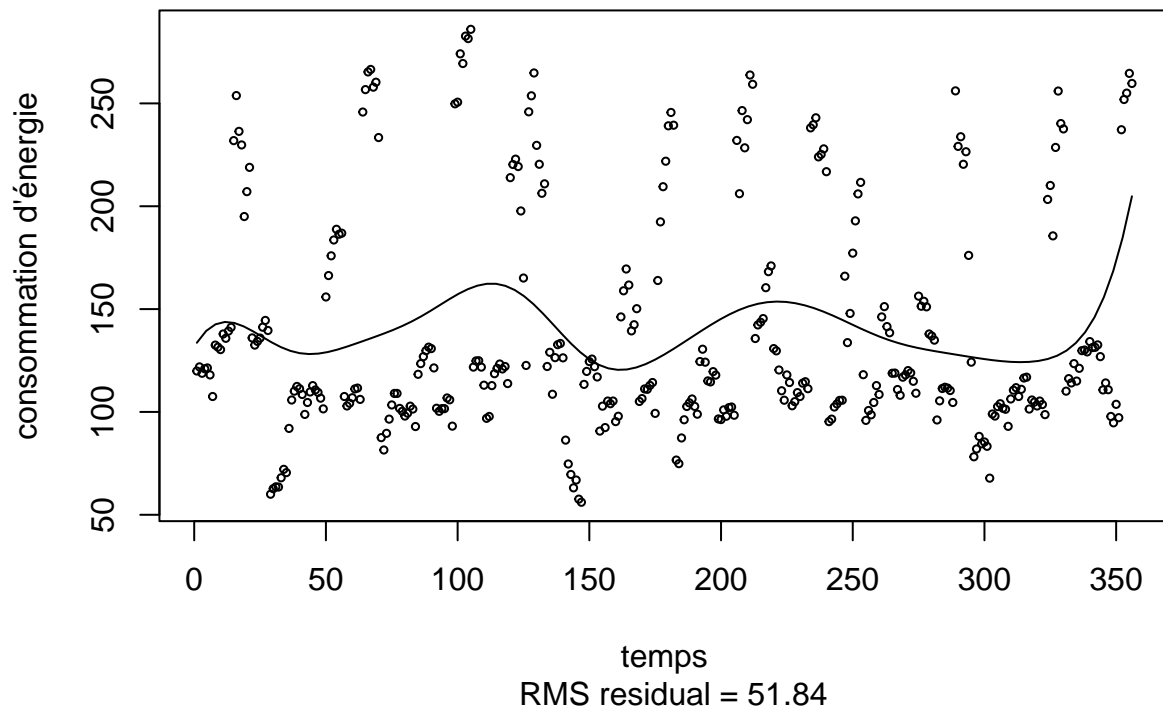
Nous évaluons notre base de spline avec cette valeur optimale de lambda.

```
lambda = exp(which.min(gcv)-10)
fdparTemp = fdPar(splbasis,Lfdobj = 4,lambda=lambda)
smoothdata = smooth.basis(tps,y,fdParobj = fdpar)
```

Nous représentons sur un même graphe les observations et la fonction estimée reconstruite \hat{f} .

```
plotfit.fd(y,# valeur de consommation pour un pays
            tps, # les dates
            smoothdata$fd, # nos données estimées
            # 1 à 356
            pch=20,
            cex=0.5,
            main="les observations et la fonction estimée \n reconstruite ^f",
            ylab = "consommation d'énergie",
            xlab="temps")
```

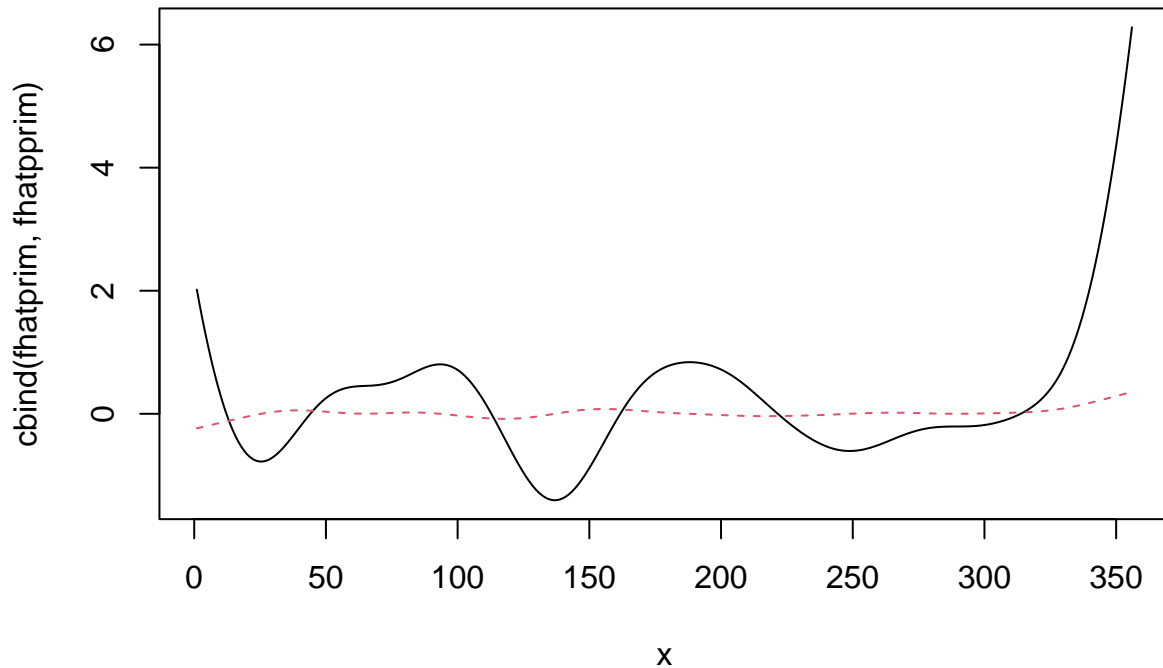
les observations et la fonction estimée reconstruite \hat{f}



Nous obtenons une fonction estimée reconstruite lisse pour l'Etat du Punjab. Elle est beaucoup moins irrégulière que nos données de base. Elle prend globalement en compte les variations de consommation d'énergie du Punjab sur une l'année 2019.

Nous représentons la dérivée première et la dérivée seconde sur le même graphique :

```
fhatprim = eval.fd(tps,smoothdata$fd,Lfdobj=1)
fhatpprim = eval.fd(tps,smoothdata$fd,Lfdobj=2)
matplot(tps,cbind(fhatprim,fhatpprim),type="l")
```



La dérivée première, en noir, correspondant à la vitesse de croissance, n'est pas très lisse. Elle indique que par moment dans l'année, la consommation d'énergie diminue ou augmente brusquement. Par exemple au niveau du 20ème et 140ème jour, sa consommation d'énergie décroît rapidement. A la fin de l'année, celle-ci explose.

Les dérivées obtenues sont régulières (on a pris une base d'ordre plus élevé, égale à 6) et le lissage est contraint par la rugosité de la dérivée seconde avec une pénalité sur la dérivée d'ordre 6.

III.B. Pour l'ensemble des Etats

Nous allons généraliser notre démarche pour tous les Etats.

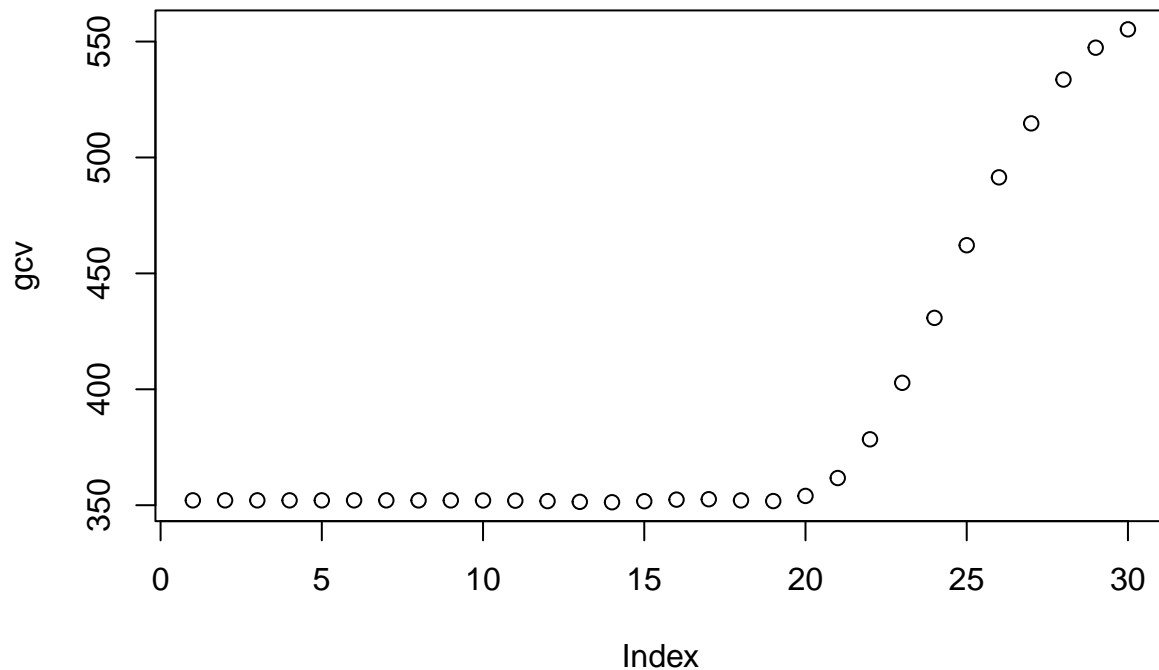
Lissage par moindre carrés pénalisés

On fait l'ajustement pour différentes valeurs de lambda pour minimiser le critère de pénalité.

On veut la fonction la plus proche des points et la moins oscillante grâce à la pénalité.

```
# Initialisation d'une grille de lambda
gcv = rep(0,30)

for (i in 1:30){
  lambda = exp(i-10)
  fdpar = fdPar(splbasis,Lfdobj = 4,lambda=lambda)
  smoothdata = smooth.basis(tps,as.matrix(data_long[,c(-1)]), fdParobj = fdpar)
  gcv[i] = mean(smoothdata$gcv)
}
plot(gcv)
```



```
which.min(gcv)
```

```
## [1] 14
```

Nous avons observé que la valeur optimale de lambda est la 14ème valeur, correspondant au minimum. Cette valeur nous permettra d'être proche des données observées.

Nous avons choisi de paramétrer le nombre de coupures à 35 car les signaux sont très irréguliers et nous souhaitons être proche, mais raisonnablement, du signal d'origine. Ce choix est arbitraire car il peut engendrer du sur-apprentissage si le nombre de coupures est trop élevée par rapport au signal de départ.

Nous allons recalculer la pénalité et refaire le lissage avec la valeur de lambda minimale :

```
lambda = exp(which.min(gcv)-10)
fdpar = fdPar(splbasis,Lfdobj = 4,lambda=lambda) # calcul pénalité
smoothdata = smooth.basis(tps,as.matrix(data_long[,c(-1)]),fdParobj = fdpar) # lissage
```

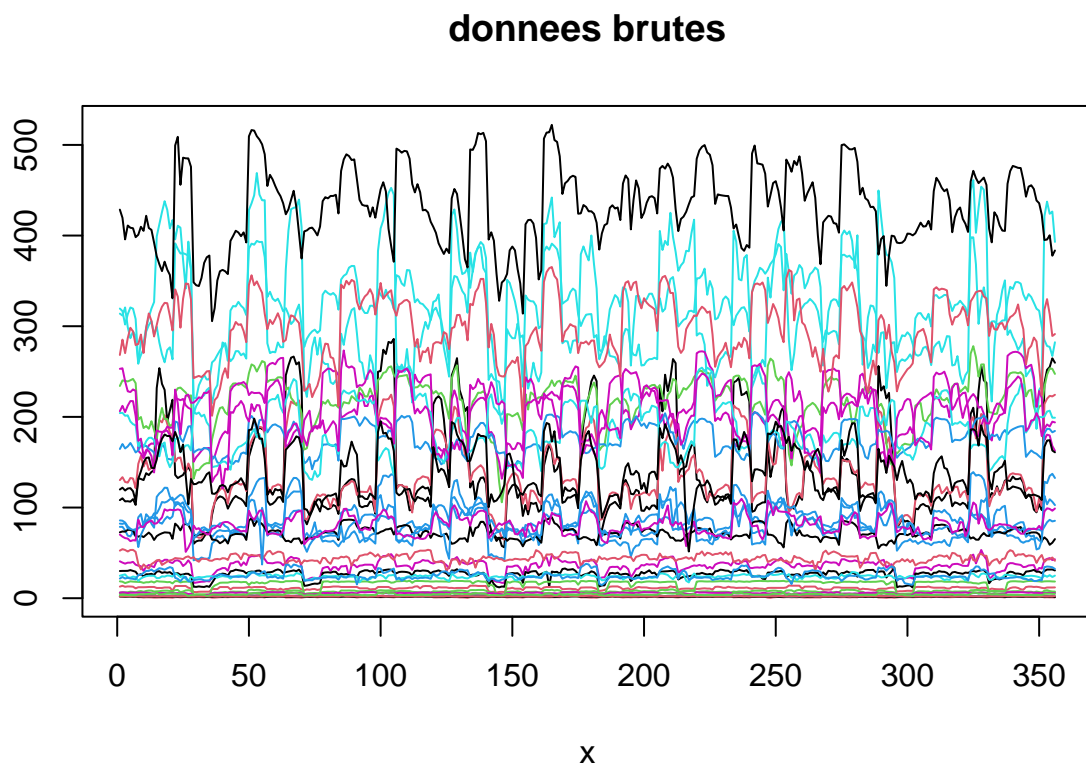
Nous évaluons notre objet fonctionnel à chaque point de temps :

```
fhatsmooth <- eval.fd(tps, smoothdata$fd)
```

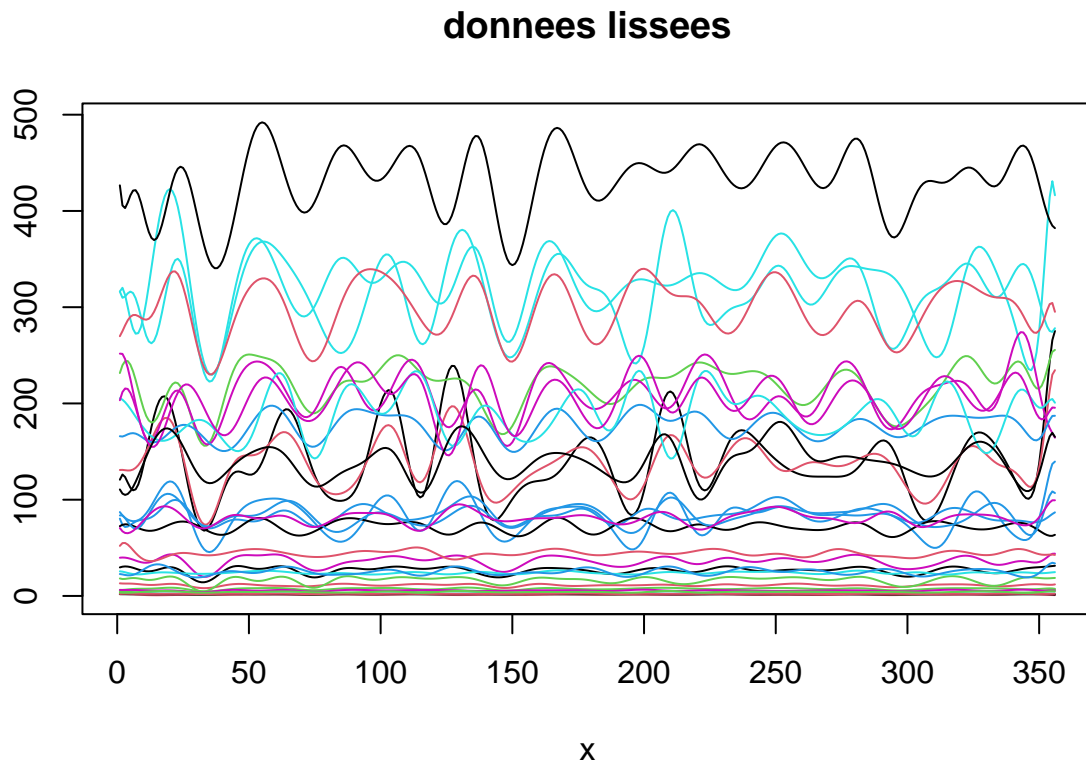
Représentation pour l'ensemble des Etats

Nous représentons les observations et la fonction estimée reconstruite \hat{f} .


```
matplot(tps,data_long[,c(-1)],type="l",lty=1,ylab="",main="donnees brutes")
```



```
matplot(tps,fhatsmooth,type="l",lty=1,ylab="",main="donnees lissees")
```



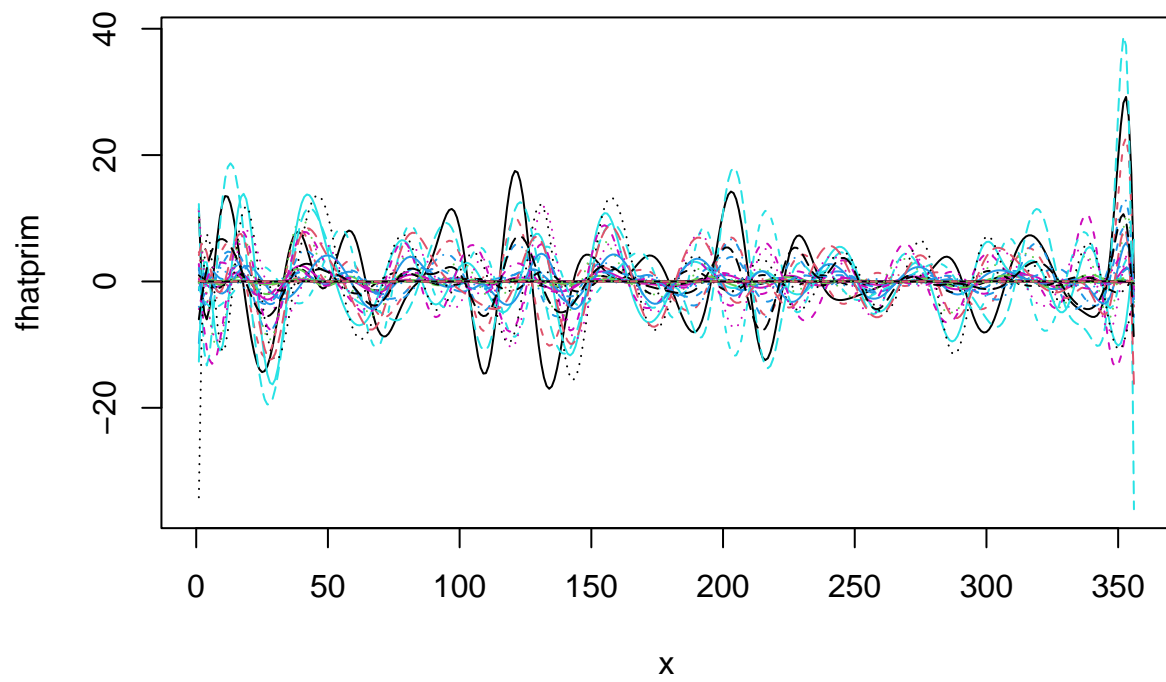
Nous obtenons des courbes lissées, moins irrégulières, qui explique bien la tendance de la consommation d'énergie pour chacun des Etats.

Lors de la création de notre base de spline, nous avons choisi 35 coupures, ce qui biaise forcément notre analyse. Avec ces courbes lissées, les analyses ne sont pas exhaustives pour éviter le sur-apprentissage. Avec cette représentation, nous pouvons supposer qu'il y a plus de variabilité en hiver (première partie de l'année) qu'en été. Il faudrait un expert pour nous aiguiller sur ce sujet.

Nous représentons maintenant la dérivée première et seconde pour étudier les vitesses de croissance et les accélérations de croissance.

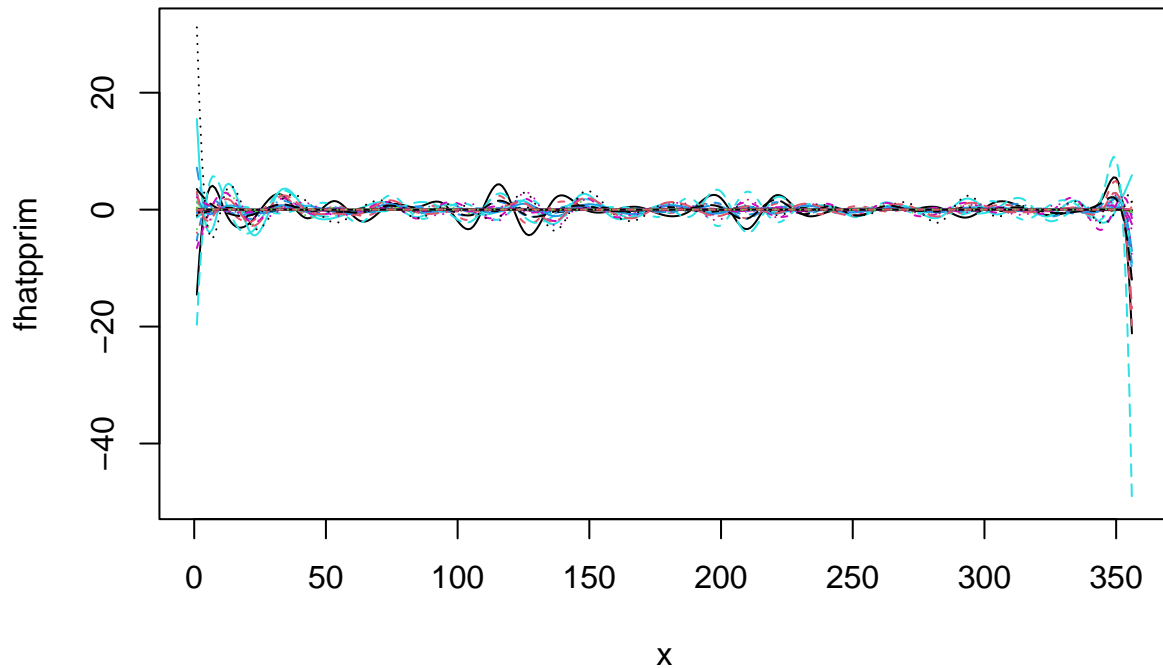
```
fhatprim = eval.fd(tps,smoothdata$fd,Lfdobj=1) # dérivé première
fhatpprim = eval.fd(tps,smoothdata$fd,Lfdobj=2) # dérivé seconde
matplot(tps,
        fhatprim,
        type="l",
        main = "Dérivée première")
```

Dérivée première



```
matplotlib(tps, fhatpprim, type = "l",  
           main = "Dérivée seconde")
```

Dérivée seconde



Les dérivés premières montrent des fluctuations, ce qui implique des différences de vitesse de consommation dans les différents Etats.

IV. Statistiques exploratoires

Nous allons réaliser quelques statistiques exploratoires pour pouvoir appréhender au mieux les données et détecter les pays avec un comportement atypique.

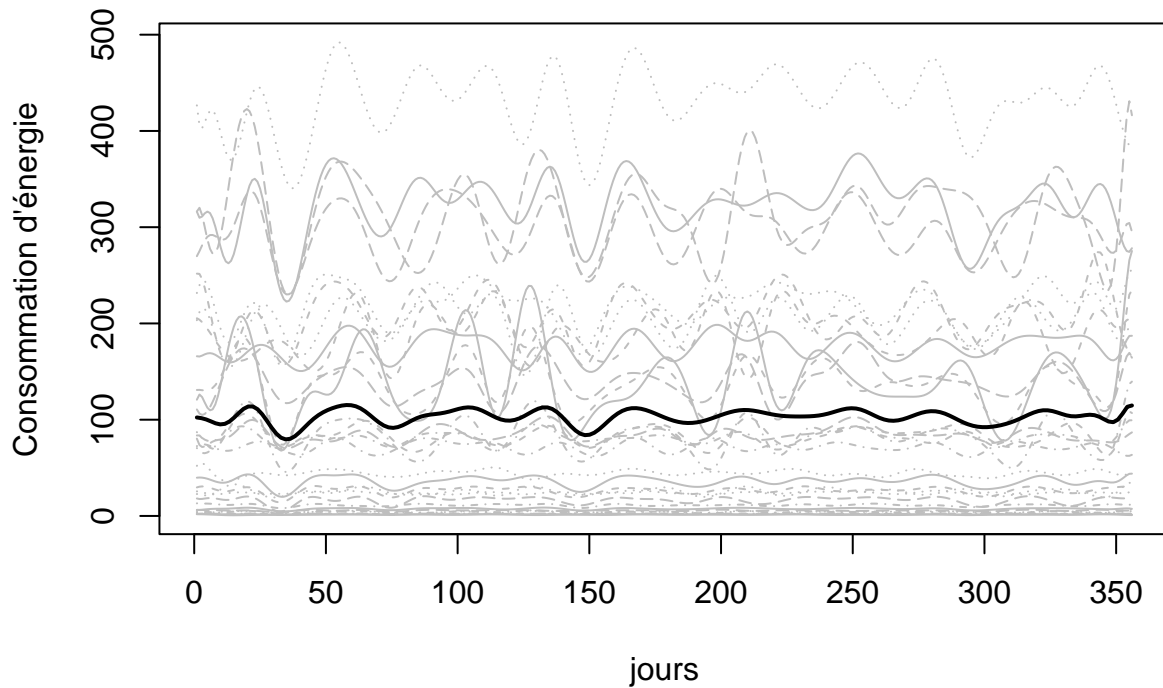
IV.A. Moyenne fonctionnelle

Afin de visualiser les tendances, nous allons représenter la moyenne de notre objet fonctionnel.

```
mean_smooth_data = mean.fd(smoothdata$fd)
# mean_smooth_data

matplot(fhatsmooth,col="gray",type="l",
        xlab="jours",
        ylab="Consommation d'énergie",
        main = "Consommation d'énergie des Etats d'Inde, \n avec tracé de la moyenne"
)
lines(mean_smooth_data,lwd=2) # moyenne
```

Consommation d'énergie des Etats d'Inde, avec tracé de la moyenne



Grâce à ce graphique, nous pouvons observer la courbe moyenne de la consommation d'énergie dans les différents Etats d'Inde. Les valeurs de consommations sont situées aux alentours de 100 Méga Units. Cette courbe est influencée par les valeurs extrêmes car nous pouvons observer un grand nombre de courbes avec des valeurs plus faibles de consommation.

IV.B. Ecart-type fonctionnel

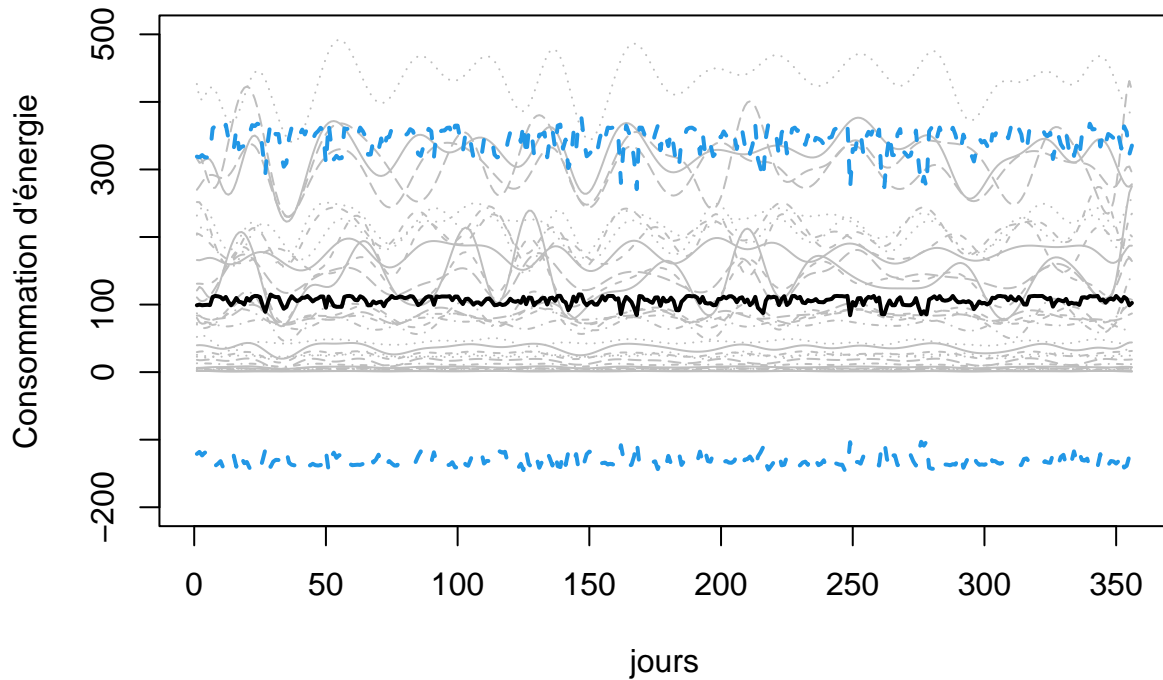
La représentation de l'écart-type fonctionnel permet de caractériser la variabilité de nos signaux sur la période étudiée.

Nous allons pouvoir identifier s'il existe des périodes de stabilité ou d'instabilité entre le début et la fin de l'année 2019.

Aussi, nous allons pouvoir comparer les différents Etats pour analyser d'éventuels similarités ou différences.

```
sd_smooth_data= sd.fd(smoothdata$fd)
```

```
matplot(fhatsmooth,
        col="gray",type="l",
        xlab="jours",
        ylab="Consommation d'énergie",
        ylim = c(-200,500))
fnmoy = eval.fd(y,mean_smooth_data)
fnsd = eval.fd(y,sd_smooth_data)
lines(fnmoy,lwd=2)
lines(fnmoy+2*fnsd,lwd=2,col=4,lty=2)
lines(fnmoy-2*fnsd,lwd=2,col=4,lty=2)
```



Ce graphique nous permet de visualiser les consommations d'énergie (données lissées) des différents Etats en Inde. La courbe moyenne, en noir, nous permet d'avoir une idée de la tendance centrale. La plage de variabilité est définie par les pointillés en bleus, correspondant à la moyenne plus (ou moins) 2 fois l'écart-type. La majorité des pays présentent donc des données lissées proches de la moyenne. Nous pouvons identifier quelques Etats atypiques.

IV.C. Surface de covariance et corrélation

Afin d'explorer les relations et les dépendances entre nos différents Etats.

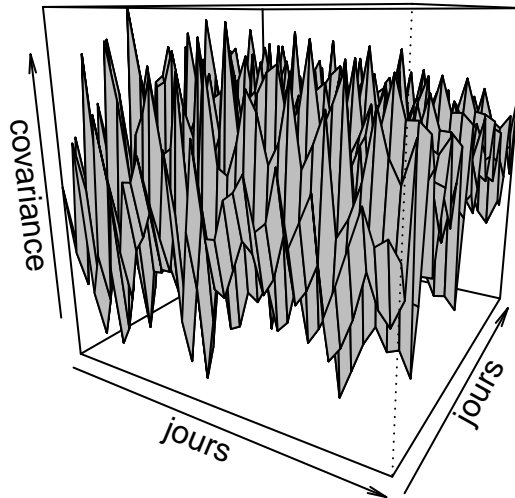
IV.C.1. Covariance

Dans un premier temps, nous allons mesurer comment les consommations des Etats évoluent conjointement en fonction du temps.

```
cov = var.fd(smoothdata$fd)
surfcov = eval.bifd(seq(1,356,length=35),seq(1,356,length=35),cov)
```

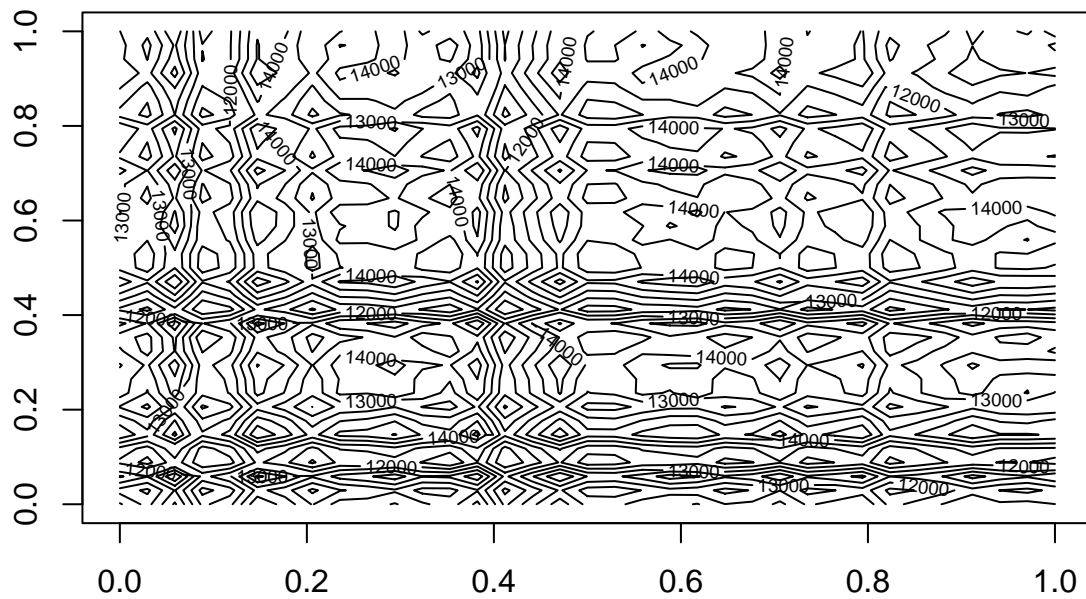
Pour le tracé d'une surface, il est préférable de choisir un nombre modéré de points dans la grille d'évaluation (la représentation est rapidement illisible dans le cas contraire). Ici, nous choisissons arbitrairement 30 points répartis entre 1 et 356.

```
persp(surfcov,col="gray",theta=30,xlab="jours",ylab="jours",zlab="covariance")
```



L'analyse de cette représentation est assez compliquée car elle permet de visualiser la covariance entre les différentes consommations des Etats selon les jours, mais aucune tendance claire ne se dégage.

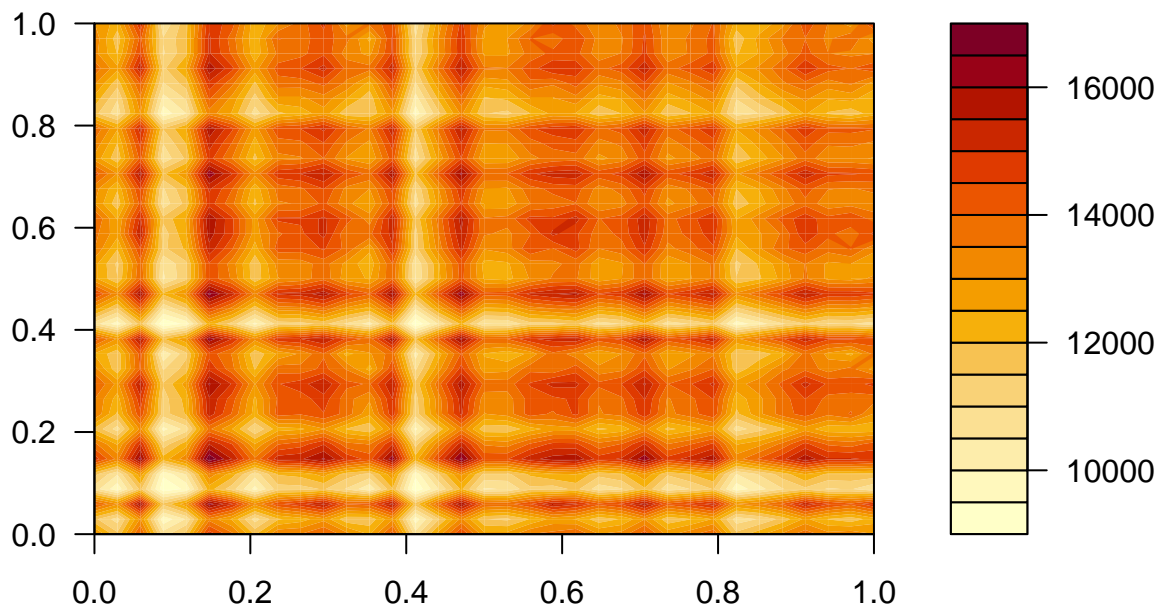
```
contour(surfcov)
```



De même, cette représentation devraient pouvoir nous aider pour identifier la covariance entre les consommations d'énergie des Etats selon les jours mais il est presque illisible.

IV.C.2. Représentation graphique en couleur

```
filled.contour(surfcov)
```

En général, on peut voir que la covariance est “moyenne” entre le début et la fin de l’année pour chaque Etat. Nous pouvons observer des moments où la covariance est moins importante, autour du mois d’Avril (0.4) et de Septembre (0.8).

IV.C.3. Corrélation

L’analyse de la corrélation nous permet de détecter si nous avons des effets d’un jour à l’autre sur la consommation d’Energie dans les divers Etats d’Inde.

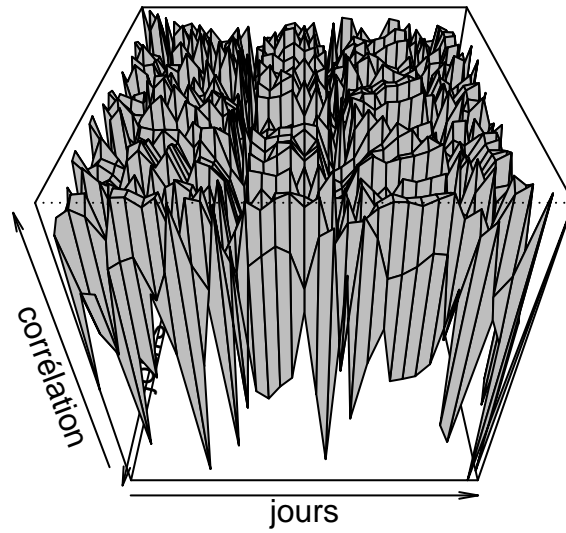
Nous construisons d’abord la matrice de corrélation :

```
cor = cor.fd(seq(1,356,length=35),smoothdata$fd)
# cor
```

Cette manipulation nous permet de récupérer la matrice de corrélation empirique évaluée (sur les données lissées) en chaque point de temps (jour).

Nous allons représenter graphiquement la matrice de corrélation :

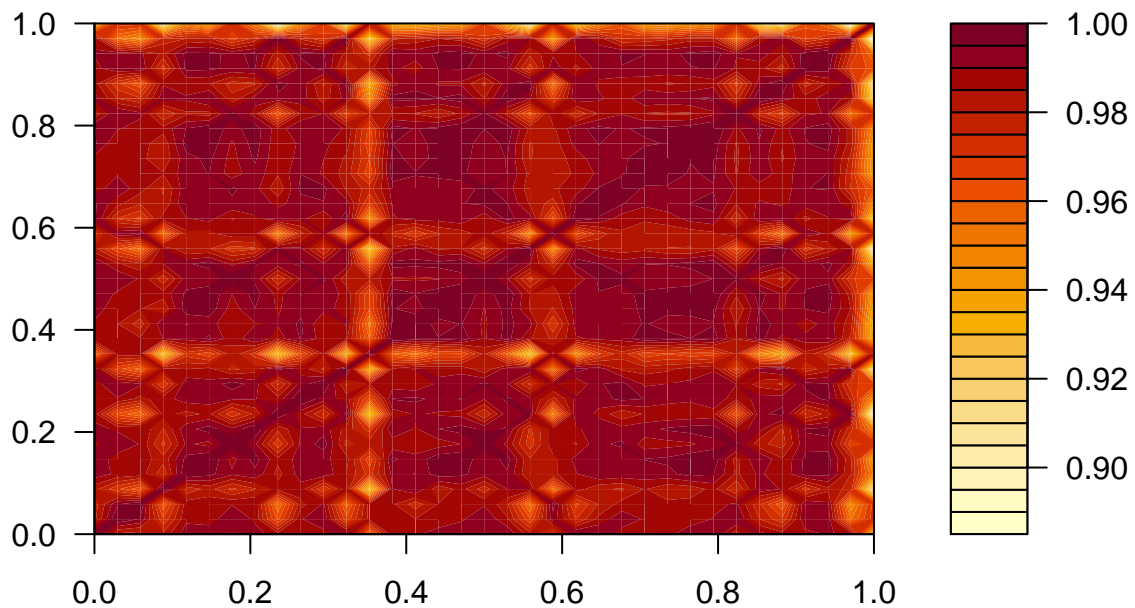
```
persp(cor,col="gray",theta=90,
      phi=40,xlab="jours",
      ylab="jours",zlab="corrélacion")
```



La corrélation est difficilement interprétable sur le graphe, nous allons essayer une autre méthode.

IV.C.4. Représentation graphique en couleur

```
filled.contour(cor)
```



Nous pouvons observer une corrélation égale à 1 sur la diagonale car la consommation d'un jour est forcément corrélée avec elle-même.

De plus, nous pouvons voir que les corrélations sont globalement fortes d'un jour à l'autre, ce qui signifie que la consommation de la veille influence celle du lendemain.

IV.D. ACP fonctionnelle

```
ACPF = pca.fd(smoothdata$fd, nharm=4, centerfns = TRUE)
ACPF$varprop
```

```
## [1] 0.987179812 0.008252207 0.001454582 0.001198720
```

```
cumsum(ACPF$varprop)
```

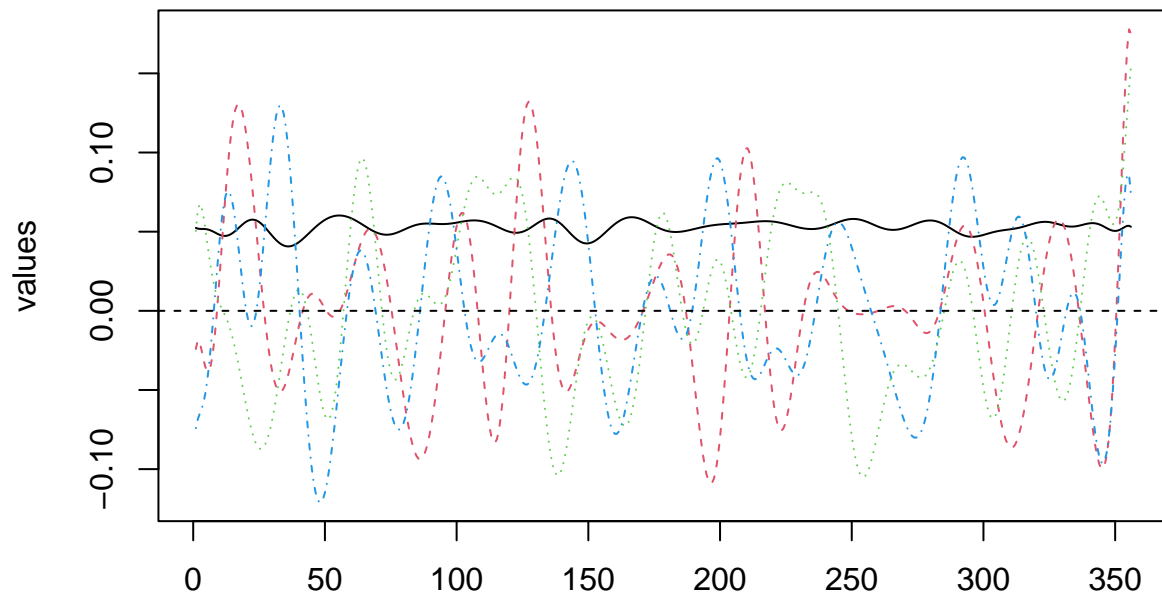
```
## [1] 0.9871798 0.9954320 0.9968866 0.9980853
```

La première composante explique environ 99% de la variabilité.

IV.D.1. Représentations des composantes principales

Nous représentons en premier lieu les composantes principales obtenues

```
plot(ACPF$harmonics)
```



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

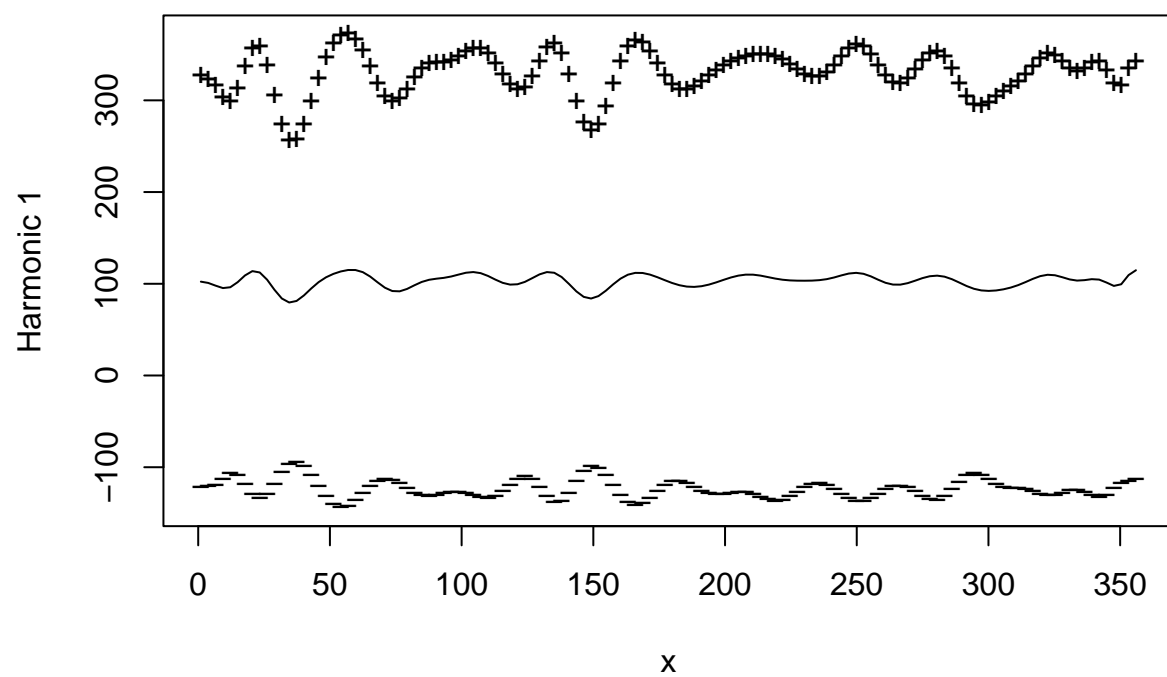
La première composante (en noir) est tout le temps positive. Globalement, elle a quelques variations, environ au 30ème, 140ème et 280ème jour de l'année. Elle a donc tendance à être plus élevée en hiver.

Pour une interprétation plus aisée, on représente la moyenne augmentée ou diminuée des premières fonctions propres.

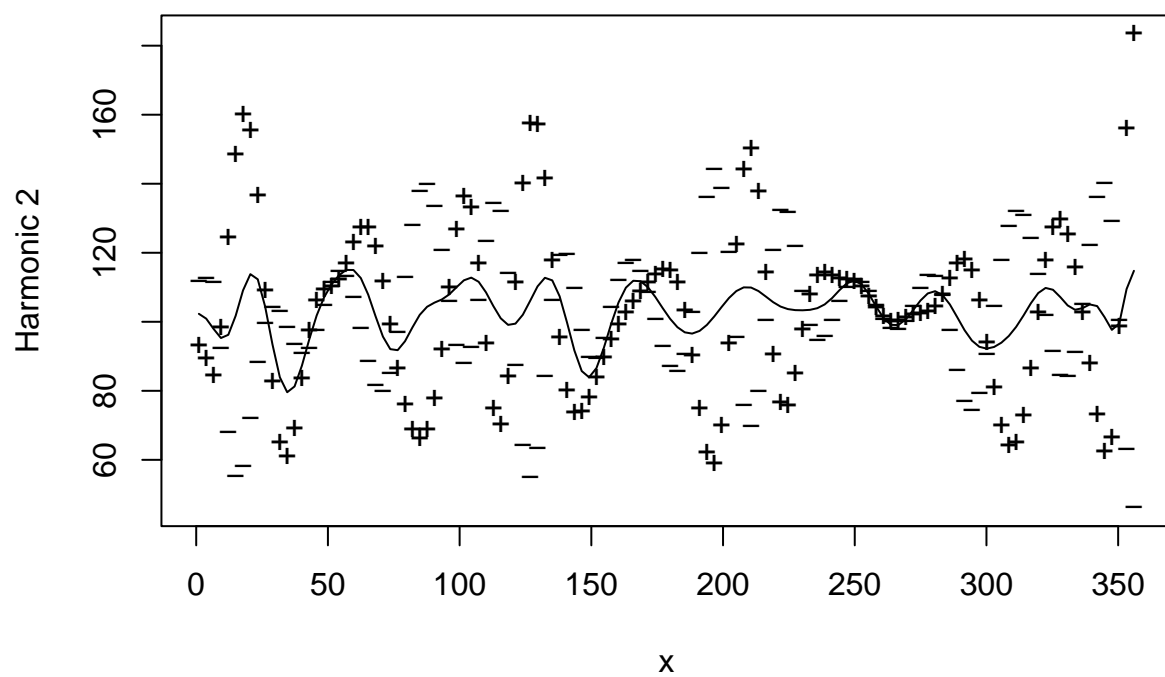
On peut utiliser la fonction `plot.pca.fd` pour cette représentation.

```
plot.pca.fd(ACPF)
```

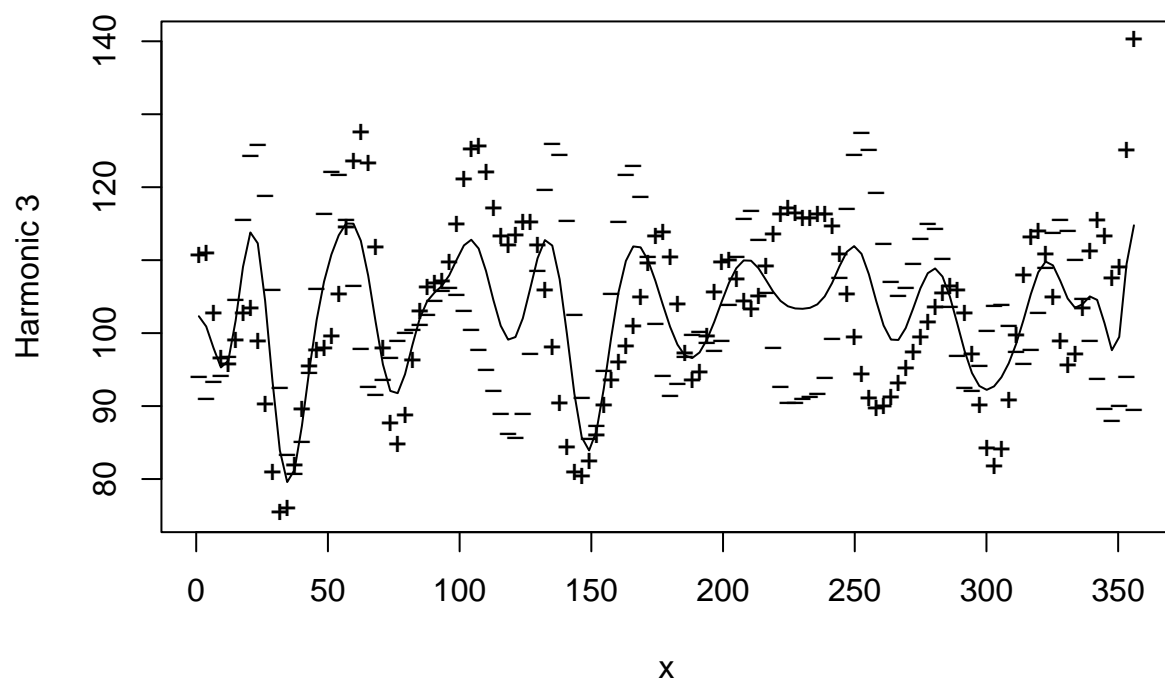
PCA function 1 (Percentage of variability 98.7)



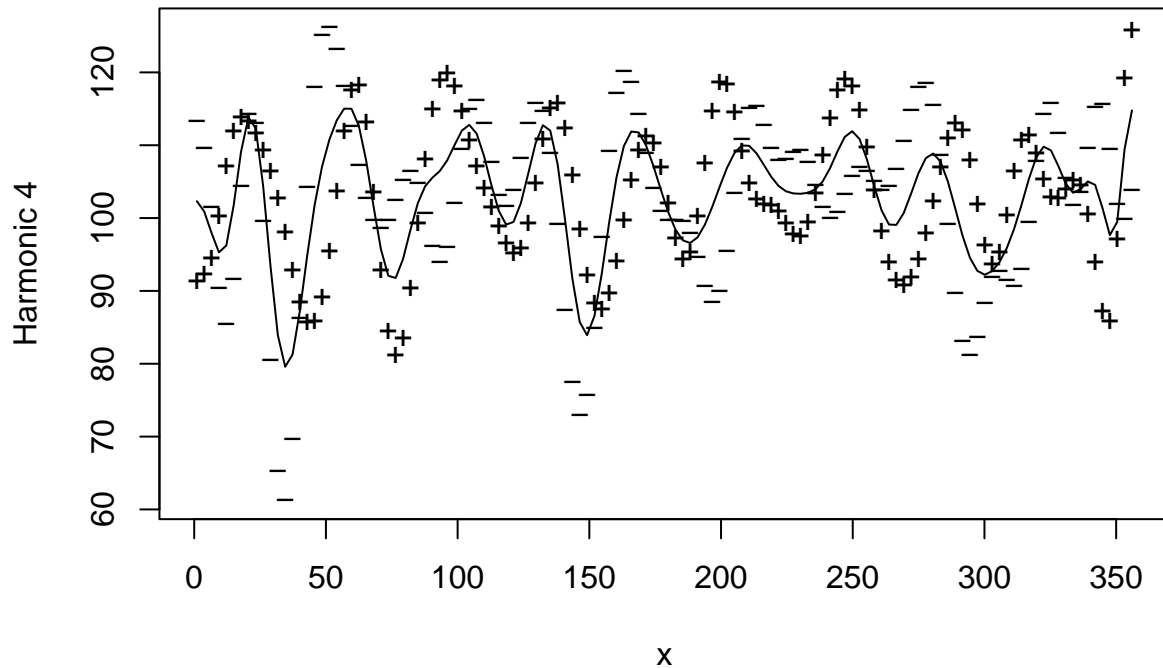
PCA function 2 (Percentage of variability 0.8)



PCA function 3 (Percentage of variability 0.1)



PCA function 4 (Percentage of variability 0.1)



Pour la première fonction propre:

- La courbe en noir représente la moyenne
- La courbe avec des “-” représente la moyenne - α * la première composantes principales
- La courbe avec des “+” représente la moyenne + α * la première composantes principales

Comme évoqué précédemment, nous pouvons supposer qu’il y a plus de variabilité en hiver qu’en été.

IV.D.2. Représentation des individus

Représentation des individus dans le premier plan factoriel

Les scores des individus sont stockés dans la matrice `scores` de `ACPF`. Nous allons les représenter sur le premier plan factoriel.

```
head(ACPF$scores)
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,]  694.4987 574.65950  211.59702  -1.507536
## [2,]  652.9611 341.00037  133.02690  -38.626858
## [3,] 2182.0635 -35.48669  162.31803 -138.399753
## [4,] -380.1501 242.24678  -11.92453  -35.326778
## [5,] 4004.6725 595.76303 -146.66314   12.226365
## [6,] -1259.7014  -5.22891   1.44213  -34.825769
```

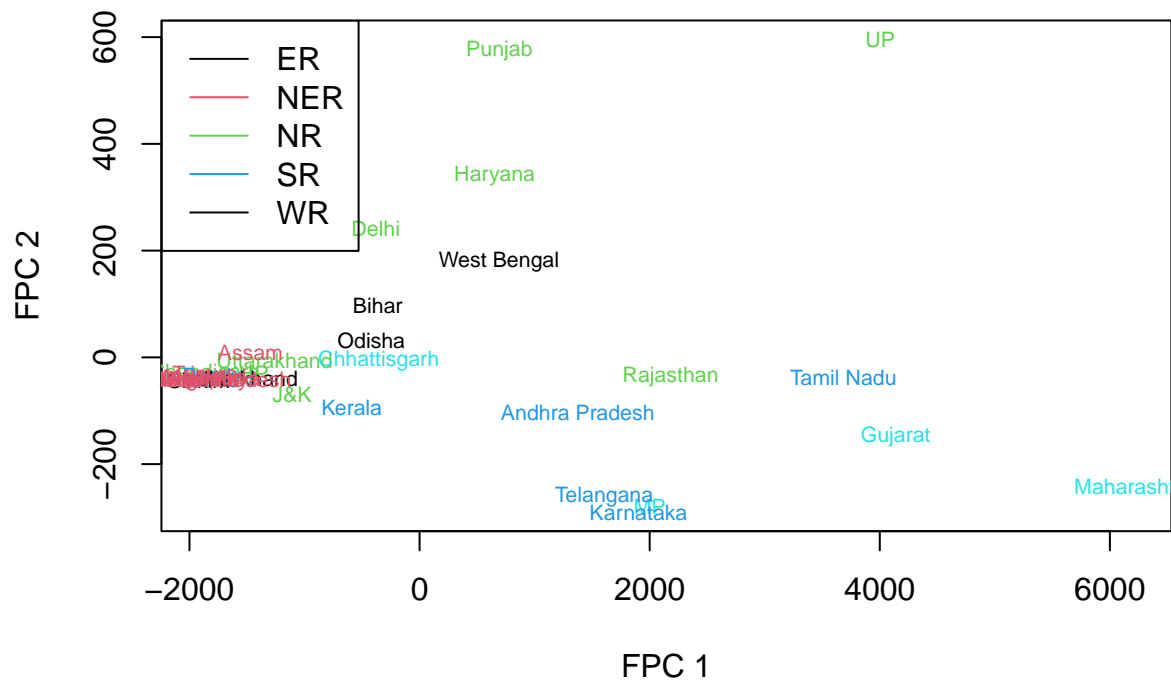
```
plot(ACPF$scores[,1],
     ACPF$scores[,2],
     pch=20,
```



```

xlab="FPC 1",
ylab="FPC 2",
type="n")
nomsvilles = colnames(data_long[,c(-1)])
regions = as.factor(data$Regions)
text(ACPF$scores[,1],
     ACPF$scores[,2],
     labels=nomsvilles,cex=0.7,col=as.numeric(regions))
legend("topleft",legend = levels(regions),col=1:4,lty=1)

```



Aucune région ne semble se distinguer concernant les consommations d'énergie des Etats. Cependant, nous observons quelques exceptions comme la région NR (Nord) où quelques Etats présentent des valeurs assez élevées sur le second axe factoriel contrairement à la région SR (Sud) où les valeurs sont plutôt faibles.

Pour la première composante principale (abscisse), les Etats concentrés à droite du graphique présentent des valeurs de consommations d'énergie élevées. Par exemple, Maharashtra et l'Etat du Gujarat sont des grands consommateurs d'énergies. Pour les petites valeurs, les Etats présentent des consommations plus faibles. Cette observation est en adéquation avec les premières représentations graphiques que nous avons réalisées au début de ce rapport.

Pour la deuxième composante principale (ordonnée), pour les Etats situés en haut du graphique, cela indique que leur consommation d'énergie est très variable selon les périodes de l'année. L'Etat du Punjab et Uttar Pradesh (UP) présentent beaucoup d'écarts de valeurs. Pour les petites valeurs, cela signifie que les Etats ont des consommations d'énergie plus stables entre le début et la fin de la période étudiée.

V. Conclusion

VI. Sources

- base de données : https://www.kaggle.com/datasets/twinkle0705/state-wise-power-consumption-in-india?fbclid=IwAR3tNVrXpTrRlO_HVOoa53bcaTR8aR7QU_Nlbe4AhhmfdK9ERa5Xxid1yk