enercoop

L'énergie militante

[ENERCOOP]2

Présentation, Problématique, Plan

Présentation (Lien Enercoop**)**

- Fournisseur français d'électricité d'origine renouvelable
- Activité fonctionnant exclusivement par des accords de gré à gré avec des petits producteurs français d'énergie renouvelable.
- Fonctionnement coopératif

1 - https://midipyrenees.enercoop.fr/actualites/les-chiffres-cles-de-la-cooperative-avril-2017

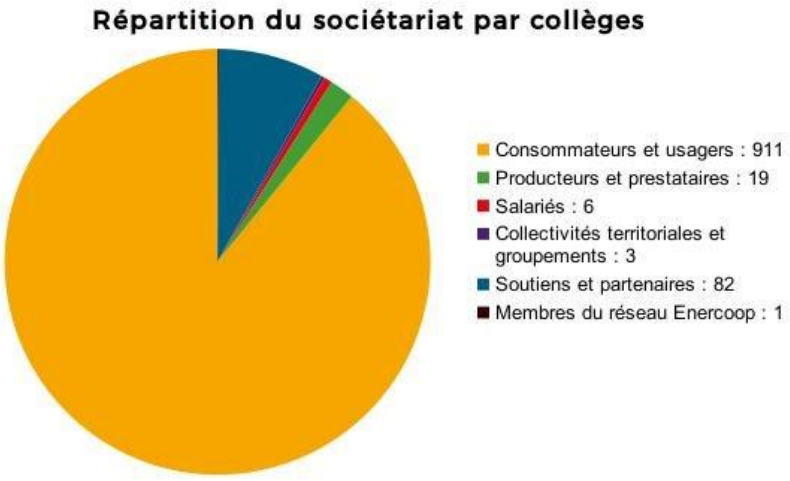
Problématique

Les sources d'énergies renouvelables sont intermittentes :

- Non disponible en permanence
- Forte variation de la disponibilité sans possibilité de contrôle

La demande en électricité des utilisateurs varie au cours du temps, et dépend de paramètres comme la météo (température, luminosité, etc.)

Objectif : Mettre en adéquation l'offre et la demande en prédisant la demande en électricité.

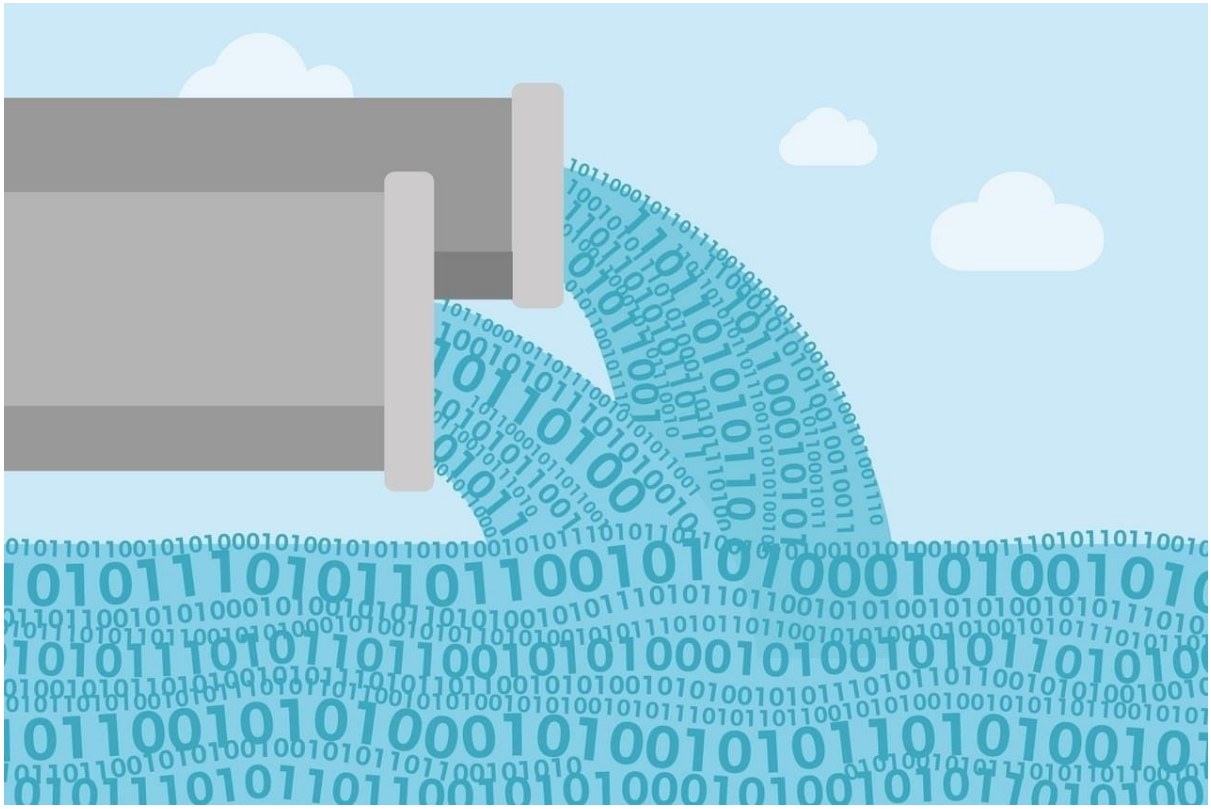


2 - https://midipyrenees.enercoop.fr/actualites/les-chiffres-cles-de-la-cooperative-avril-2017

Plan :

1. Sources et retraitement des données
2. Mission 1 : Corrigez les données de consommation mensuelles de l'effet température (dues au chauffage électrique) en utilisant une régression linéaire.
3. Mission 2 : Effectuez une désaisonnalisation de la consommation que vous aurez obtenue après correction, grâce aux moyennes mobiles.
4. Mission 3 : Effectuez une prévision de la consommation (corrigée de l'effet température) sur un an, en utilisant la méthode de Holt Winters (lissage exponentiel) puis la méthode ARMA sur la série temporelle désaisonnalisée.

Sources et retraitement des données



Importation des données énergie

Source : [Réseau de Transport de l'Electricité (RTE)]

RTE distribue l'électricité en France et produit des données sur son réseau accessibles via l'outil Eco2mix

```
df_energie <- as.data.frame(read.xls('../fichiers_csv/energie.xls'))
colnames(df_energie)
df_energie$Mois <- as.vector(df_energie$Mois)
df_energie$Qualite <- as.vector(df_energie$Qualite)
df_energie$Territoire <- as.vector(df_energie$Territoire)
```

3 - Import des données et transformation des colonnes en vecteur pour faciliter la manipulation

```
df_energie <- df_energie %>%
  select(Mois, Territoire, Consommation.totale) %>%
  filter(Territoire == "France")
```

4 - On ne s'intéresse qu'aux données de la France entière

```
mois = rep(c('JAN','FEV','MAR','AVR','MAI','JUIN','JUIL','AOUT','SEP','OCT','NOV','DEC'),8)
annee = c(rep('2010',12),rep('2011',12),rep('2012',12),rep('2013',12),rep('2014',12),rep('2015',12),rep('2016',12)
```

```
df_energie <- df_energie[1:96,]
df_energie <- cbind.data.frame(df_energie,mois,annee)
df_energie$mois <- as.vector(df_energie$mois)
df_energie$annee <- as.vector(df_energie$annee)
```

5 - On prépare le data.frame pour un pivot :

- *Création des variables mois et année*
- *Ajout des variables au data.frame de base*
- *Transformation des variables en vecteur pour faciliter la manipulation*

```
df_energie <- df_energie[,c("annee", "mois", "Consommation.totale", "Mois","Territoire")]
```

```
df_energie <- cast(df_energie,annee ~ mois,value='Consommation.totale',sum)
df_energie <- df_energie[,c("annee", 'JAN','FEV','MAR','AVR','MAI','JUIN','JUIL','AOUT','SEP','OCT','NOV','DEC')]
```

*6 - **Pivot : Utiliser une valeurs unique située dans une colonne pour construire le nouveau data.frame***

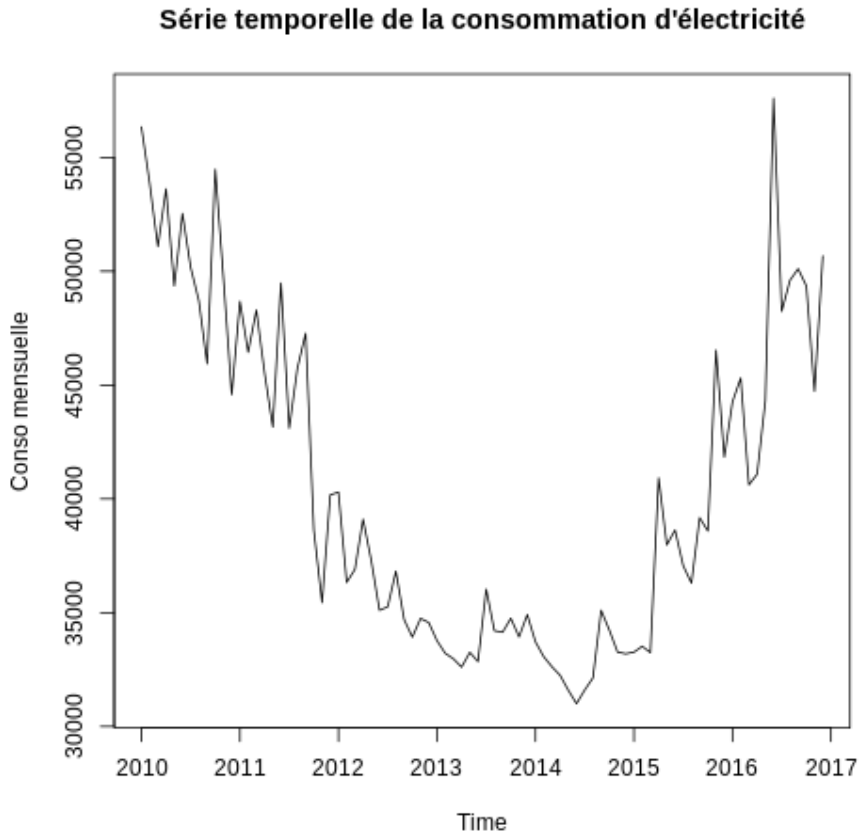
```
df_energie <- df_energie[1:7,]
```

```
df_energie
```

annee	JAN	FEV	MAR	AVR	MAI	JUIN	JUIL	AOUT	SEP	OCT	NOV	DEC
2010	56342	48698	48294	38637	37284	34567	36031	33069	35104	40918	46532	57600
2011	53873	45937	45543	35442	35107	33771	34204	32625	34230	37977	41837	48241
2012	51086	54476	43156	40176	35257	33219	34141	32247	33269	38628	44260	49602
2013	53619	49639	49480	40292	36821	32973	34751	31591	33195	37063	45310	50108
2014	49359	44580	43104	36331	34695	32608	33935	31004	33266	36301	40617	49350
2015	52536	48676	45739	36898	33927	33256	34912	31603	33521	39170	41085	44727
2016	50161	46440	47270	39102	34746	32852	33718	32132	33245	38590	44293	50670

7 - On fera abstraction de l'année 2017 (voir valeur manquante df_base_meteo)

Affichage du data.frame



Importation des données météo

Source : [Simulateur DJU CEGIBAT]

CEGIBAT :

Simulateur de DJU : Cet outil, réalisé en partenariat avec Météo France, permet de calculer les degrés jour (DJ ou DJU) chauffage ou climatisation sur une période, une station météo et un seuil de température donnés.

DJU : Le **degré jour** est une valeur représentative de l'écart entre la température d'une journée donnée et un seuil de température préétabli (18 °C dans le cas des DJU ou Degré Jour Unifié). Sommés sur une période, ils permettent de calculer les besoins de chauffage et de climatisation d'un bâtiment.

```
files_to_read <- list.files(path = '../fichiers_csv/meteo_data',full.names = TRUE)
```

```
liste_meteo <- lapply(files_to_read,function(i){
    list_csv <- read.xls(i)


})
```

```
df_meteo <- lapply(liste_meteo,function(i){
    a <- data.frame(i,stringsAsFactors = FALSE)

station <- as.vector(a[1,'X.1'])

mat <- as.matrix(a)
mat <- mat[-c(1,2,3,4,5,6,7),]
colnames(mat) <- mat[1,]
mat <- mat[-1,]
colnames(mat)[1] <- 'Annee'
nb_row <- length(mat[,'Annee'])
mat<- cbind(rep(station,nb_row),mat)
colnames(mat)[1] <- 'Station'
as.data.frame(mat,stringsAsFactors = FALSE)
mat[1:7,] #Valeur manquantes en 2017
})
```

8 -

- *Récupération de tous les fichiers xls par station*
- *Suppression des valeurs non pertinentes*
- *Résultat obtenu : Liste de data.frame par station*

```
df_base_meteo <- ldply(df_meteo, rbind)
df_base_meteo <- rename(df_base_meteo,c('Station' = 'station','Annee'='annee','JAN'='JAN','F\xc9V'='FEV','M
```

```
df_base_meteo[] <- as.vector(unlist(df_base_meteo, use.names = FALSE))
df_base_meteo[-2] <- as.numeric(unlist(df_base_meteo[-2], use.names = FALSE))
```

```
Warning message in eval(expr, envir, enclos):
“NAs introduits lors de la conversion automatique”
```

9 -

- *Empilement des data.frames pour en avoir qu'un seul*
- *Conversion en vecteur de la colonne 'stations'*
- *Conversion en vecteurs numériques des colonnes mensuelles*


```
df_base_meteo[] <- as.vector(unlist(df_base_meteo, use.names = FALSE))
df_base_meteo[-2] <- as.numeric(unlist(df_base_meteo[-2], use.names = FALSE))
```

```
Warning message in eval(expr, envir, enclos):
“NAs introduits lors de la conversion automatique”
```

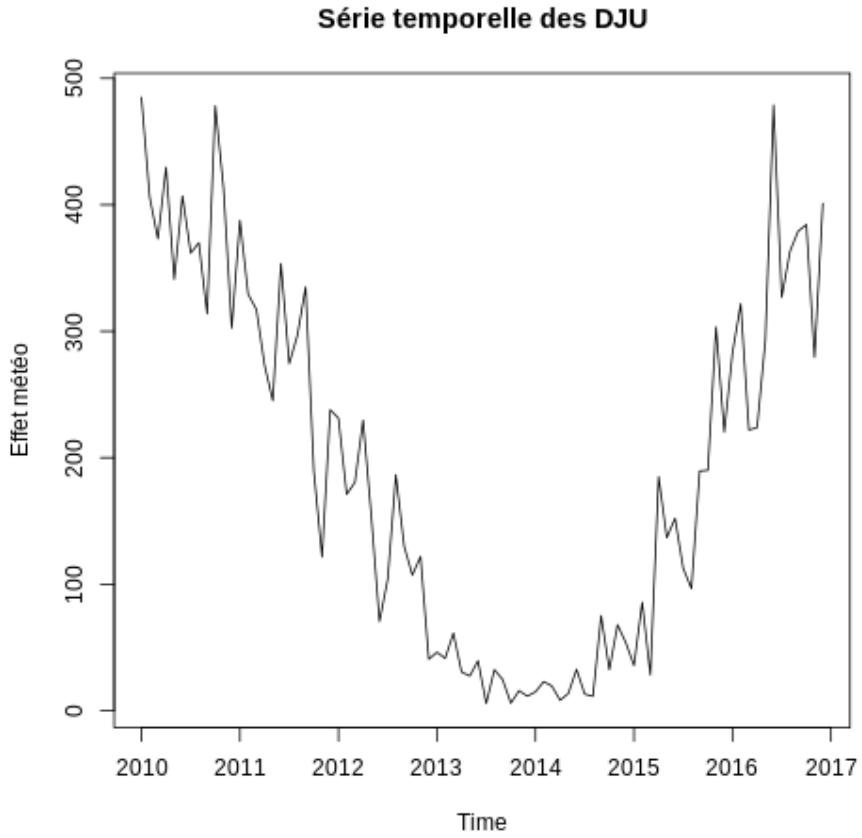

```
df_base_meteo <- ddply(df_base_meteo,~annee,summarise, JAN = mean(JAN),
      FEV = mean(FEV),
      MAR = mean(MAR),
      AVR = mean(AVR),
MAI = mean(MAI),
JUN = mean(JUN),
JUI = mean(JUI),
AOU = mean(AOU),
SEP = mean(SEP),
OCT = mean(OCT),
NOV = mean(NOV),
DEC = mean(DEC))
```

10 - Moyenne DJU en France par mois et par an

```
df_base_meteo
```

annee	JAN	FEV	MAR	AVR	MAI	JUN	JUI	AOU	SEP	OCT	NOV	DEC
2010	484.9267	369.9756	317.2278	189.6422	154.08889	40.69889	5.656667	22.85444	75.07222	184.95667	303.3189	478.8556
2011	406.2789	313.8867	273.3411	121.5056	70.45333	46.10111	32.323333	19.43889	32.31667	136.76111	220.3189	326.8511
2012	373.1278	478.2522	245.1367	237.8778	103.84111	41.35444	24.552222	8.21000	67.81889	152.04111	282.9667	363.1200
2013	429.6233	413.0800	353.4844	231.2144	186.51889	61.25778	5.921111	13.47556	53.80667	112.49667	321.7444	379.1200
2014	341.1244	302.5311	274.3767	171.0933	130.10000	30.49000	15.743333	32.52667	35.64222	96.57444	221.9400	384.3467
2015	406.9656	387.6867	296.7800	180.4811	107.18111	27.36556	11.420000	13.06667	85.59333	189.47000	223.8689	279.3678
2016	361.8989	329.3178	334.9833	229.5767	121.88111	39.40111	14.776667	11.26111	28.29556	190.01556	292.3822	401.0267

11 - Affichage des données météo



Correction de l'effet météo

1. On effectue la régression linéaire
2. On calcule la série corrigée de l'effet météo.

```
## Note : Colinéarité = En algèbre linéaire, deux vecteurs u et v d'un espace vectoriel E sont colinéaires s'il existe un
reg=lm(log_df_energie_ts~log_df_base_meteo_ts)
summary(reg)

## Affichage des coefficients
reg$coefficients

## Coefficients initiaux
a=reg$coefficients
```

```
Call:
lm(formula = log_df_energie_ts ~ log_df_base_meteo_ts)

Residuals:
      Min        1Q    Median        3Q       Max
-0.153079 -0.072196 -0.003129  0.067329  0.260510

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         10.023935   0.041358  242.37   <2e-16 ***
log_df_base_meteo_ts  0.119856   0.008432   14.21   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.09386 on 82 degrees of freedom
Multiple R-squared:  0.7113,    Adjusted R-squared:  0.7078
F-statistic:   202 on 1 and 82 DF,  p-value: < 2.2e-16
```

(Intercept)	10.0239347476463
log_df_base_meteo_ts	0.119855935210034

cem = Correction Effet météo

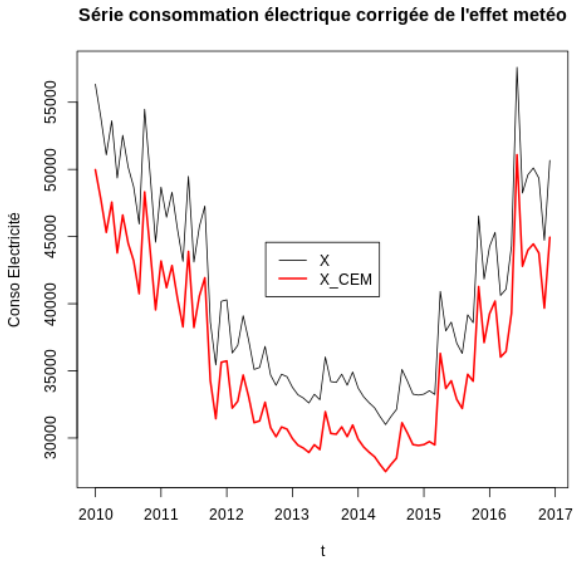
12 - On effectue la régression linéaire

```
log_df_energie_ts_cem=log_df_energie_ts-a[2]


df_energie_ts_cem=exp(log_df_energie_ts_cem)

ts.plot(df_energie_ts,df_energie_ts_cem,xlab='t',ylab='Conso Electricité',col=c(1,2),lwd=c(1,2),main = "Série consommation
legend('center',legend=c("X","X_CEM"),col=c(1,2),lwd=c(1,2))
```

13 - On calcule la série corrigée de l'effet météo.



14 - On calcule la série corrigée de l'effet météo.

Correction de la saisonnalité

Nous allons utilisé les moyennes mobiles : Une **moyenne mobile** *M* est une combinaison linéaire d'instants passés et futurs de la série temporelle.

La désaisonnalisation s'effectue en deux phases comprenant chacune 4 étapes :

1. Estimation de la tendance
2. Estimation de la somme composante saisonnière-perturbation
3. Estimation de la composante saisonnière
4. Estimation de la série corrigée des variations saisonnières

---

Phase 1 :

Etape 1 : Primo-estimation de la tendance

```
m2_12=function(x){
  y=(1/12)*stats::filter(x,c(0.5,rep(1,times=11),0.5))
  return(y)
}
t1=m2_12(log_df_energie_ts_cem)
```

Etape 2 : Primo-estimation de la somme composante saisonnière-perturbation

```
sig1=log_df_energie_ts_cem-t1
```

Etape 3 : Primo-estimation de la composante saisonnière

```
m3=function(x){

  y=(1/3)*stats::filter(x,rep(1,times=3))

  return(y)

}
s1=m3(m3(sig1))
shat1=s1-m2_12(s1)
```

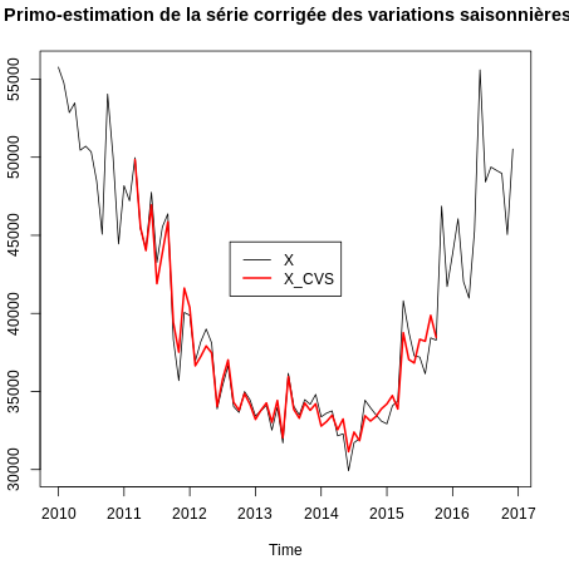
Etape 4 : Primo-estimation de la série corrigée des variations saisonnières

```
log_df_energie_ts_cem_cvs1=log_df_energie_ts_cem-shat1
df_energie_ts_cem_cvs1=exp(log_df_energie_ts_cem_cvs1)

ts.plot(df_energie_ts_cem,df_energie_ts_cem_cvs1,col=c(1,2),lwd=c(1,2))

legend("center",legend=c("X","X_CVS"),col=c(1,2),lwd=c(1,2))
```

15 - Méthodologie de la phase 1



16 - Affichage de l'estimation après la phase 1

Phase 2 :

Etape 1 : Estimation définitive de la tendance

```
m13h=function(x){
  y=(1/16796)*stats::filter(x,c(-325,-468,0,1100,2475,3600,4032,3600,2475,1100,0,-468,-325))
  return(y)
}
t2=m13h(df_energie_ts_cem_cvs1)
```

Etape 2 : Estimation définitive de la somme composante saisonnière-perturbation (2)

```
sig2=log_df_energie_ts_cem-t2
```

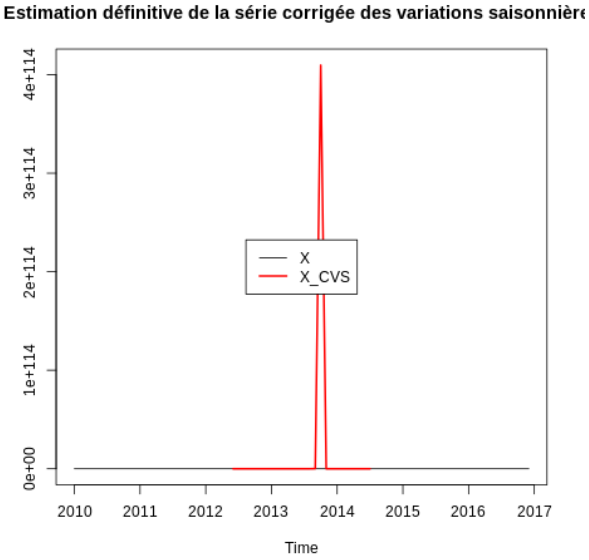
Etape 3 : Estimation définitive de la composante saisonnière

```
m5=function(x){
  y=(1/5)*stats::filter(x,rep(1,times=5))
  return(y)
}
s2=m3(m5(sig2))
shat2=s2-m2_12(s2)
```

Estimation définitive de la série corrigée des variations saisonnières

```
log_df_energie_ts_cem_cvs2=log_df_energie_ts_cem-shat2
df_energie_ts_cem_cvs2=exp(log_df_energie_ts_cem_cvs2)
```

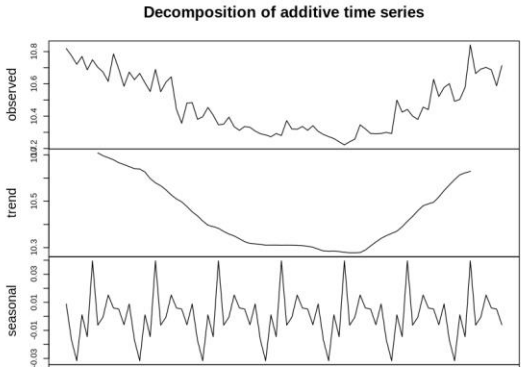
17 - Méthodologie phase 2



18 - Affichage de l'estimation définitive

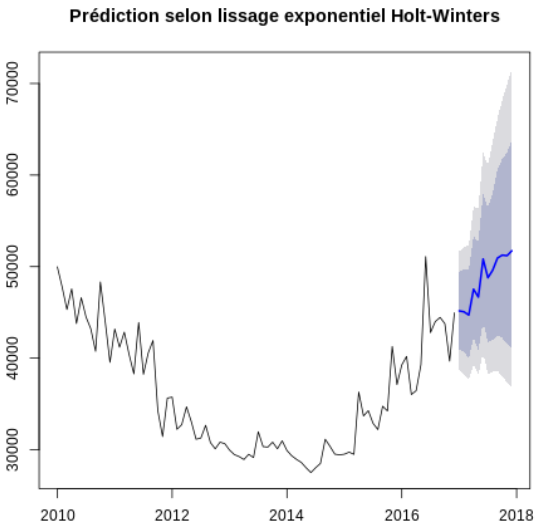
```
##  OU Decomp Y
decomp_y=decompose(log_df_energie_ts_cem,type='additive')
plot(decomp_y)

## Correction effet saisonniers
x_cvs=exp(log_df_energie_ts_cem-decomp_y$seasonal)
ts.plot(df_energie_ts_cem,x_cvs,xlab='t',ylab='Conso électrique',col=c(1,2),lwd=c(1,2))
legend('topleft',legend=c("X","X_CVS"),col=c(1,2),lwd=c(1,2))
```



19 - En désaisonnalisant avec les moyennes mobiles, nous avons perdu de l'information. Par conséquent, on préférera désaisonnaliser via la fonction décompose.

Prédictions

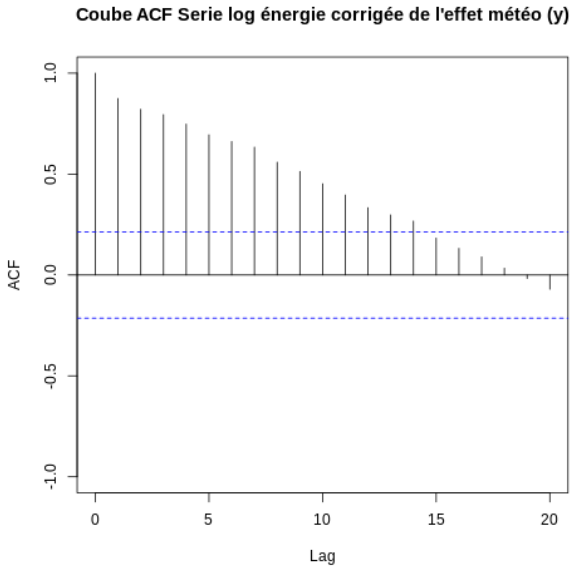


20 - Modélisation : Holt-Winters

La prévision obtenue par lissage exponentiel simple ne tient pas compte de la saisonalité. La méthode de Holt-Winters est plus efficace que les lissages exponentiels simple ou double et tient compte d'une éventuelle tendance et saisonnalité

Modélisation : ARMA

Affichage de la courbe ACF de la série

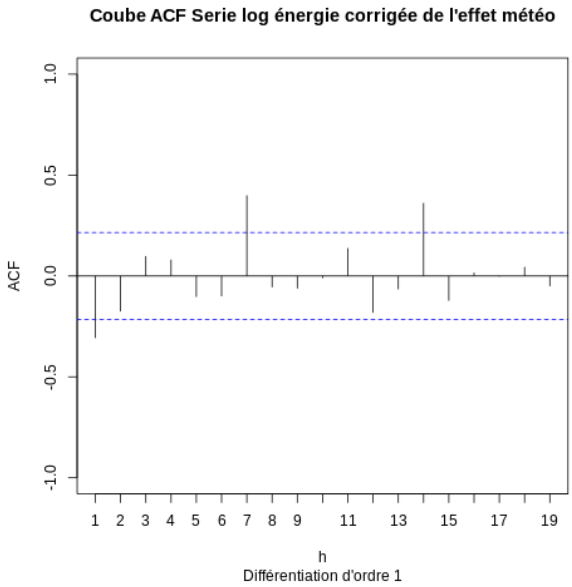


21 - En statistiques, on aime bien avoir des valeurs indépendantes -- or les séries temporelles ont des valeurs naturellement corrélées. Pour observer les corrélation on utilise une courbe d'autocorrélation (ACF)Il y a trois observations possibles :

- *l'ACF qui est presque nulle correspond à des données non corrélées (processus stationnaire)*
- *l'ACF tantôt positive, tantôt négative à des données périodiques*
- *Décroissance lente de la courbe ACF (processus non stationnaire)*

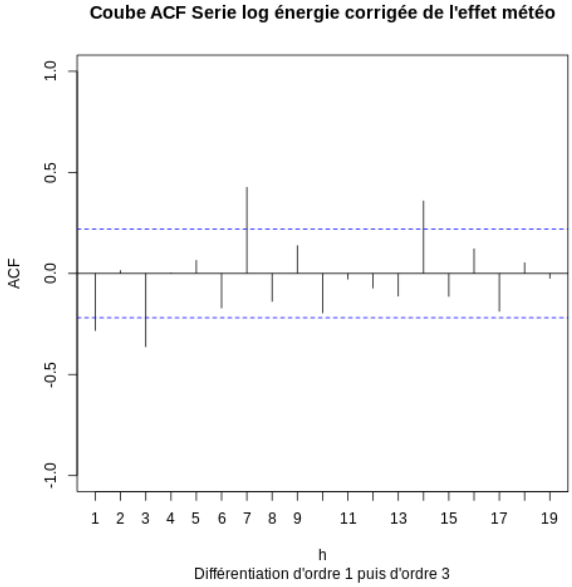
La sortie ACF présente une décroissance lente vers 0, ce qui traduit un problème de non-stationnarité.

Stationnarisation du processus par différentiation d'ordre 1



La sortie ACF de la série ainsi différenciée n'est toujours pas stationnaire. Nous allons ajouter une nouvelle différentiation d'ordre 3.

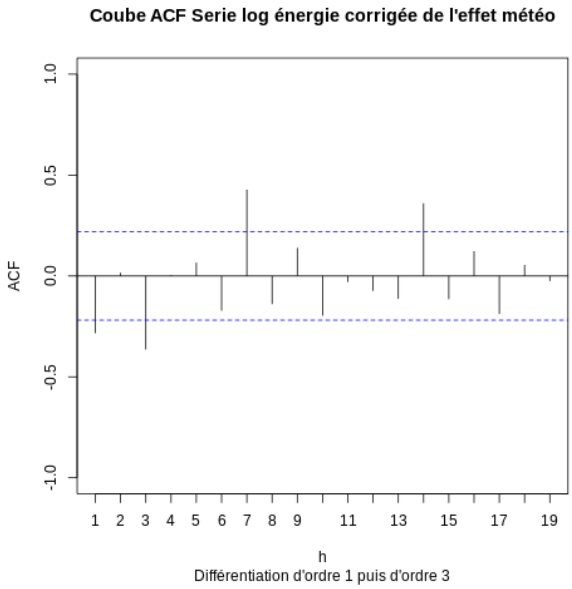
Stationnarisation du processus par différentiation d'ordre 1 et d'ordre 3



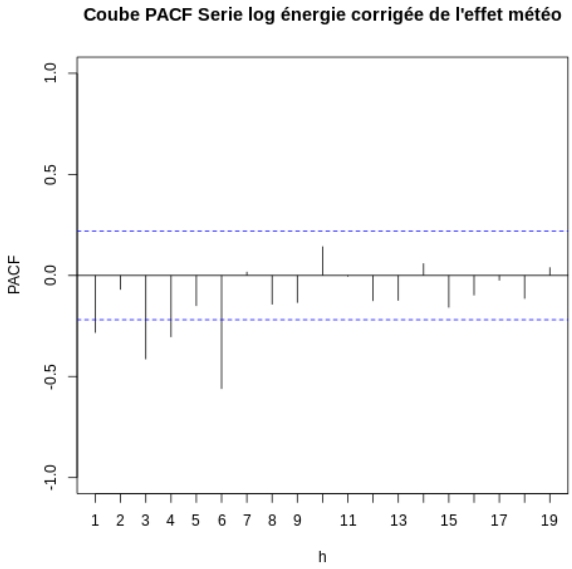
La sortie ACF de la série doublement différenciée semble pouvoir être interprétée comme un autocorrélogramme simple empirique. On identifiera donc un modèle ARMA sur la série $(I-B)(I-B3)\ln(Xt)$.

Identification, estimation et validation de modèles en s'appuyant sur les corrélogrammes simples et partiels estimés

---



22 - MA = 3



23 - AR = 6

Modèles potentiels

t-stat : Significativité d'un coefficient

H0 : Coefficient = 0

H1 : Coeffificient différent de 0

box-test : La statistique Ljung-Box Q permet de déterminer si une série d'observations dans le temps sont aléatoires et indépendantes. Si les observations ne sont pas indépendantes, une observation

peut être corrélée avec une autre observation k unités de temps après, établissant ainsi une relation appelée autocorrélation. L'autocorrélation peut nuire à l'exactitude d'un modèle de prévision basé sur le temps, tel qu'un diagramme de série chronologique, et conduire à une mauvaise interprétation des données.

H0 : Données indépendantes (bruit blanc faible)

H1 : Données qui ne sont pas indépendantes (bruit blanc fort)

Modèle 1 On estime en premier lieu un modèle ARMA(6,3,3) au vu des autocorrélogrammes empiriques simples et partiels.

```
model1=Arima(y,order=c(6,3,3),include.mean = FALSE,method = "CSS-ML")
summary(model1)

t_stat(model1)

Box.test.2(model1$residuals,nlag = c(6,9,12,15,18,20),type="Ljung-Box",decim=5)
```

```
Series: y
ARIMA(6,3,3)

Coefficients:
          ar1      ar2      ar3      ar4      ar5      ar6      ma1     ma2
      -0.6626  -0.7005  -0.6648  -0.6284  -0.5810  -0.5085  -1.8473  0.9982
s.e.   0.3426   0.1809   0.1804   0.1493   0.1197   0.1178   0.3920  0.8457
          ma3
      -0.1509
s.e.   0.4649

sigma^2 estimated as 0.003674:  log likelihood=109.26
AIC=-198.52   AICc=-195.37   BIC=-174.57

Training set error measures:
                      ME       RMSE        MAE         MPE      MAPE      MASE
Training set -0.001067881 0.05611399 0.04058518 -0.01093686 0.3868557 0.2960009
                     ACF1
Training set -0.01156882
```

	ar1	ar2	ar3	ar4	ar5	ar6	ma1	ma2	ma3
t.stat	-1.933766	-4.315245	-3.685570	-4.207834	-4.852105	-4.316714	-4.712207	1.180362	-0.324595
p.val	0.053142	0.000016	0.000228	0.000026	0.000001	0.000016	0.000002	0.237856	0.745487

Retard	p-value
6	0.98557
9	0.99737
12	0.96415
15	0.83994
18	0.93641

24 - Modèle 1 :

Tous les coeffiencents de notre modèle ne sont pas significatifs (> à 0.05). Nous allons essayer un nouveau modèle en retirant le coefficient le moins significatif (ma3).

Modèle 2

```
model2=Arima(y,order=c(6,3,2),include.mean = FALSE,method = "CSS-ML")
summary(model2)

t_stat(model2)

Box.test.2(model2$residuals,nlag = c(6,9,12,15,18,20),type="Ljung-Box",decim=5)
```

```
Series: y
ARIMA(6,3,2)

Coefficients:
          ar1      ar2      ar3      ar4      ar5      ar6      ma1     ma2
      -0.7648  -0.8159  -0.6915  -0.6326  -0.5787  -0.5173  -1.7291  0.7291
s.e.   0.1189   0.1475   0.1664   0.1591   0.1294   0.1078   0.1342  0.1305

sigma^2 estimated as 0.003614:  log likelihood=109.2
AIC=-200.4   AICc=-197.86   BIC=-178.85

Training set error measures:
                      ME      RMSE        MAE          MPE      MAPE      MASE
Training set -0.0002009644 0.056046 0.04012383 -0.002606068 0.3823312 0.2926361
                     ACF1
Training set -0.02565172
```

	ar1	ar2	ar3	ar4	ar5	ar6	ma1	ma2
t.stat	-6.432957	-5.531827	-4.154823	-3.976077	-4.472846	-4.799091	-12.88381	5.58499
p.val	0.000000	0.000000	0.000033	0.000070	0.000008	0.000002	0.00000	0.00000

Retard	p-value
6	0.97987
9	0.99795
12	0.94181
15	0.80047
18	0.91003
20	0.94665

25 - Modèle 2 : On retire l'ordre du paramètre le moins significatif du modèle 1

t-test : Ce modèle ayant des paramètres non significatifs, on en teste un second.

Box-cox : Les p-valeurs ne sont pas inférieures au niveau de test de 5% donc on ne rejette pas l'hypothèse nulle qui est : Le résidu suit un bruit blanc

Test de normalité du résidu

H0: Le résidu suit une loi normale
H1 : Le résidu ne suit pas une loi normale

```
shapiro.test(model2$residuals)
```

```
        Shapiro-Wilk normality test

data:  model2$residuals
W = 0.96396, p-value = 0.01885
```

26 - Cependant, Le test de normalité du résidu n'est pas concluant

Modèle 3

```
model3=Arima(y,order=c(6,3,1),include.mean = FALSE,method = "CSS-ML")
summary(model3)

t_stat(model3)

Box.test.2(model3$residuals,nlag = c(6,9,12,15,18,20),type="Ljung-Box",decim=5)
```

```
Series: y
ARIMA(6,3,1)

Coefficients:
         ar1      ar2      ar3      ar4      ar5      ar6      ma1
      -1.1942  -1.3041  -1.1521  -0.9287  -0.6985  -0.5346  -1.0000
s.e.   0.0956   0.1469   0.1797   0.1789   0.1442   0.0975   0.0344

sigma^2 estimated as 0.00445:  log likelihood=102.05
AIC=-188.11   AICc=-186.11   BIC=-168.95

Training set error measures:
                      ME       RMSE        MAE         MPE      MAPE      MASE
Training set -0.001358168 0.06261106 0.04575614 -0.01448254 0.4357222 0.3337143
                   ACF1
Training set -0.161944
```

	ar1	ar2	ar3	ar4	ar5	ar6	ma1
t.stat	-12.49316	-8.877176	-6.412403	-5.191793	-4.843365	-5.483582	-29.08815
p.val	0.00000	0.000000	0.000000	0.000000	0.000001	0.000000	0.00000

Retard	p-value
6	0.31746
9	0.26253
12	0.12012
15	0.06215
18	0.13442
20	0.20971

27 - Modèle 3 : On retire l'ordre du paramètre le moins significatif du modèle 2

t-test : Ce modèle ayant des paramètres significatifs, on utilisera ce modèle pour nos predictions

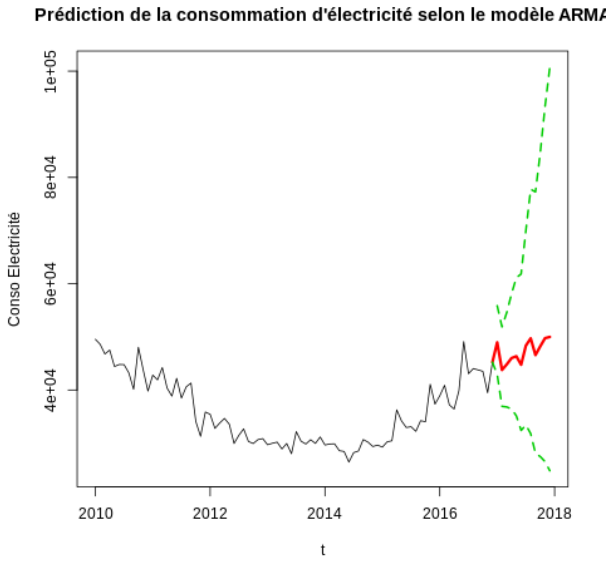
Box-cox : Les p-valeurs ne sont pas inférieures au niveau de test de 5% donc on ne rejette pas l'hypothèse nulle qui est : Le résidu suit un bruit blanc

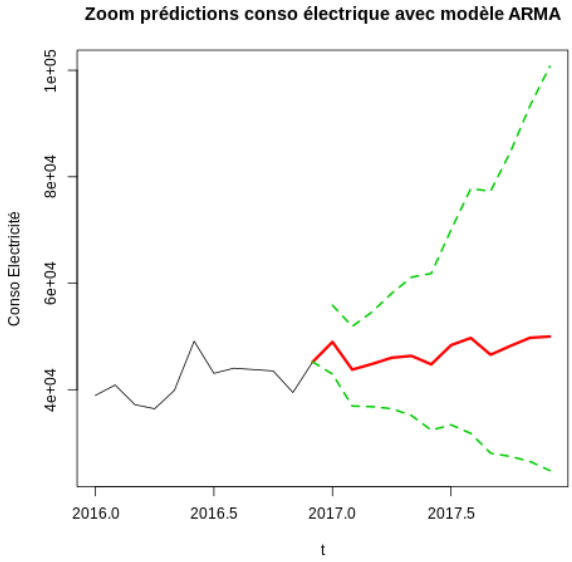
```
shapiro.test(model3$residuals)
```

```
        Shapiro-Wilk normality test

data:  model3$residuals
W = 0.97341, p-value = 0.07927
```

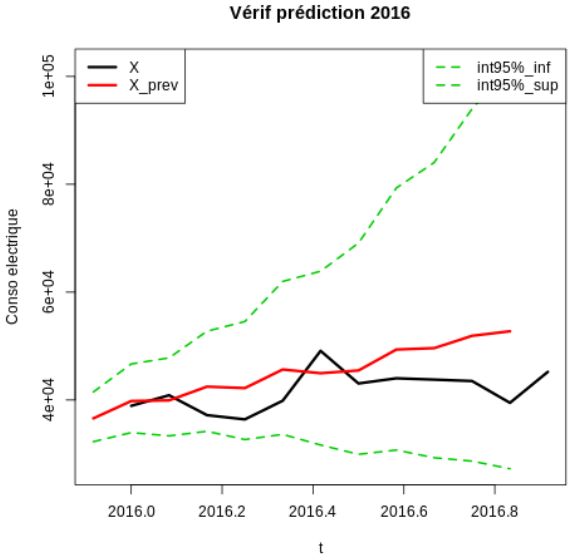
28 - Tous les tests sont concluants





Conclusion : Analyse à posteriori

Nous allons vérifier l'efficacité du modèle en tronquant la série de l'année 2016, qu'on cherche ensuite à prévoir à partir de l'historique 2010-2015.



29 - Test shapiro non concluant : On rejette fortement l'hypothèse de normalité. Cependant compte tenu de la faible probabilité d'obtenir un résidu qui suive une loi normale nous acceptons ce test.

Le RMSE est un indicateur qui mesure comment est éparpillé le résidu (les erreurs de prédiction). On doit le minimiser.

```
rmse=sqrt(mean((x_a_prevoir-pred_tronc)^2))
rmse
```

6243.35850500749

Le MAPE mesure à quel point les predictions diffère des valeurs réelles. On doit le minimiser également. Une différence de 12% semble acceptable.

```
mape=mean(abs(1-pred_tronc/x_a_prevoir))*100
mape
```

12.8795785843695