Examen Statistiques II

Projet Séries chronologiques

Submitted To:
Mme Anissa Rabhi

Submitted By :
Sami BEN BRAHIM
2A MIndS -FINANCE-

List of Figures

Contents

Introduction

Une série temporelle est une suite de valeurs numériques représentant la variation d'une certaine grandeur/valeur au cours du temps. En s'appuyant sur des concepts probabilistes et statistiques, on parvient par l'intermédiaire de la série chronologique à comprendre et analyser l'évolution du prix de pétrole, du taux de change, de la vitesse du vent,... Non seulement cela mais aussi on peut prévoir leur comportement futur en se basant sur certaines méthodes de "Forecasting", et c'est ce qu'on va faire dans ce projet en manipulant des données relatives à l'évolution d'une certaine quantité.
Donc tout d'abord, on va faire le prétraitement des données pour comprendre et anayser l'évolution de notre série temporelle. Ensuite, on va estimer la tendance et la saisonnalité de notre série. Enfin, on fait le forecasting pour prévoir l'évolution de la quantité considérée dans les deux prochaines années.

Chapter 1

Observation et prétraitement des données

1 Transformation des données

On dispose d'un fichier word qui contient les données de l'évolution mensuelle d'une certaine quantité.

2.445096 3.413556 0.1216148 1.015122 2.767924 1.879017 0.6799043 0.2633145
-0.595518 1.101397 -0.1284823 1.214477 0.3319922 3.253486 2.4693 0.5531992
4.073041 1.579469 1.944064 2.05827 -1.730466 -3.990143 -1.601067 3.4643
5.151589 5.485229 3.287411 0.8666735 2.623148 1.272351 0.4044382 1.119419
4.783418 3.128788 3.408668 0.7838095 3.379215 4.857688 4.487959 4.686121
3.76127 2.179099 1.796065 1.285345 4.042392 6.204218 7.000369 0.1730169
0.7202729 6.754407 9.69569 8.366198 2.355795 0.3741808 0.779934 6.45218
9.010877 8.880639 5.264182 2.26852 6.679158 11.39282 11.38154 7.069186
4.75259 6.76327 9.684003 9.648772 4.991009 2.259617 1.666322 4.492107
10.63804 8.489658 10.21704 8.91845 8.630397 7.841018 7.357964 7.649252
5.540967 6.594242 7.649124 9.921665 14.01545 10.57084 8.095099 5.156471
9.402515 10.5937 10.17743 9.342241 6.049854 4.834511 7.168277 8.161612
13.21406 14.50754 9.387707 8.174664 9.676791 11.40462 13.26704 9.621463
8.975399 9.906933 10.46896 10.9326 14.22168 15.51673 11.23763 9.302331
7.354789 10.19111 12.43395 11.8468 9.853657 7.194033 9.830435 12.22011

Figure 1.1: Les données avant le prétraitement

Pour faciliter la lecture et la compréhension des données, on va tout d'abord convertir le fichier word en un fichier csv. Une fois converti, on se rend compte que les données sont rangées horizontalement. Donc il suffit de faire un *copier-coller* en cochant l'option "transpose" pour obtenir une colonne contenant les 120 observations mensuelles. La figure ci-après permet d'observer une partie du dataset étudié.

Quantity
2.445096
3.413556
0.1216148
1.015122
2.767924
1.879017
0.6799043
0.2633145
-0.595518
1.101397
-0.1284823
1.214477
0.3319922
3.253486
2.4693
0.5531992
4.073041
1.579469
1.944064

Figure 1.2: Les données après le pré-traitement

2 Prétraitement des données

2.1 Importation des données

Le premier reflexe d'un statisticien est de procéder à l'analyse descriptive du jeu de données considéré pour structurer et représenter l'information contenue dans les données.

Alors, on commence par importer le dataset "*dataexamen*" en exécutant le code suivant :

```
library(readr)
quantity <- as.data.frame(read_csv("dataexamen.csv"))
view(quantity)
```

Figure 1.3: importation des données

La fonction as.data.frame() transforme le fichier de sortie de la fonction read_csv(qui est un objet de type tibble) en objet de type data.frame.
On applique ensuite les trois fonctions classiques lors de l'exploration d'un jeu de données: str, summary et hist.

2.2 Str, summary et hist

```
> str(quantity)
'data.frame':   120 obs. of  1 variable:
 $ Quantity: num  2.445 3.414 0.122 1.015 2.768 ...
 - attr(*, "spec")=
  .. cols(
  ..   Quantity = col_double()
  .. )
```

Figure 1.4: Résultat de la fonction "str"

120 observations de la variable *quantite* de type double forment le dataset "dataexamen".

```
> summary(quantity)
    Quantity
 Min.   :-3.990
 1st Qu.: 2.239
 Median : 5.795
 Mean   : 5.902
 3rd Qu.: 9.457
 Max.   :15.517
```

Figure 1.5: Résultat de la fonction "summary"

On remarque que la moyenne des quantités est supérieure à la médiane. Donc on peut conclure que plus que 50% des quantités sont inférieures à la moyenne évaluée à 5.902

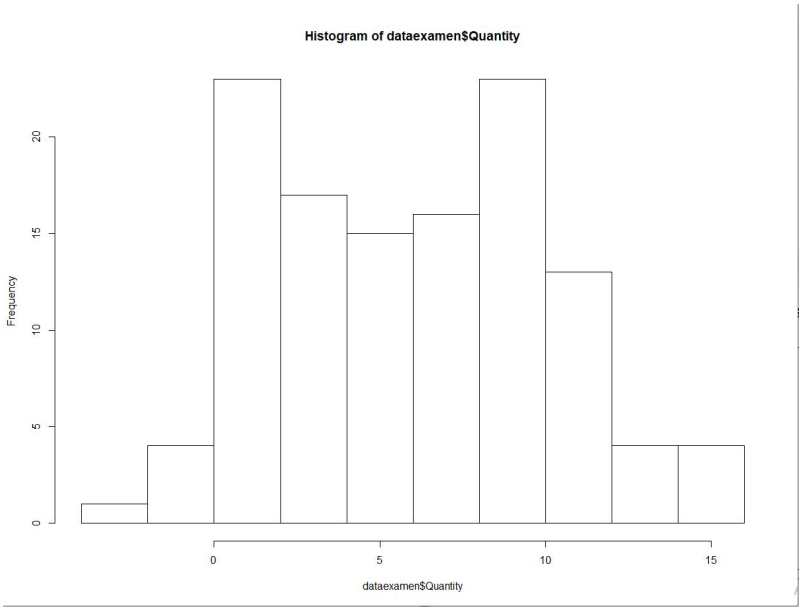

Figure 1.6: L'histogramme des quantités

L'histogramme montre qu'environ 50% des quantités sont localisées dans l'union des deux intervalles $[0, 2]$ et $[8, 10]$, et que les valeurs négatives sont les moins récurrentes. (Les valeurs négatives expriment certainement une perte ou un déficit).

2.3 Le chronogramme

On commence par transformer la série en objet de classe ts (time series) à l'aide de la fonction ts(), on choisit 2004 comme l'année du début des observations,

```
quantity <- ts(quantity,start=2004,frequency=12)
```

Figure 1.7: La fonction *ts*

puis on trace le chronogramme de la série

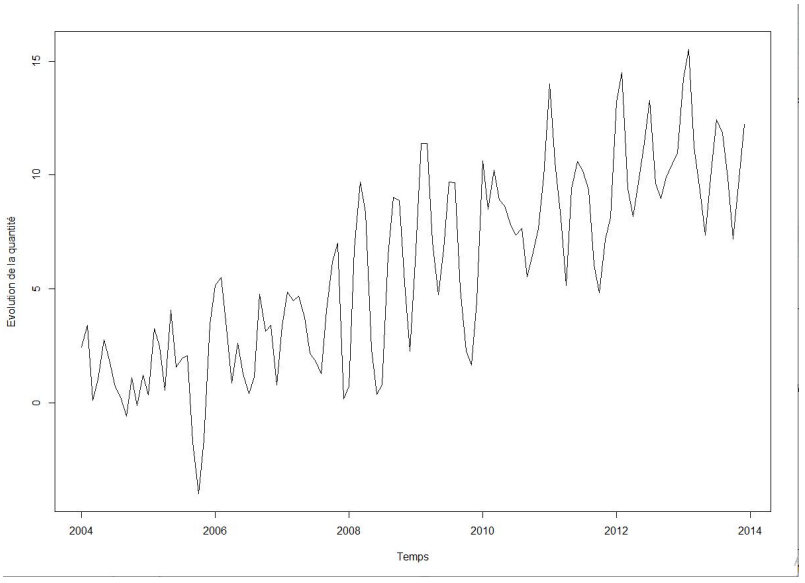

Figure 1.8: le chronogramme de la série

On peut remarquer une tendance à la croissance de la série, avec un motif qui se répète(à partir de 2006), révélant peut-être la présence d'une composante saisonnière. D'après le ts.plot(), on peut déduire que la variation de la quantité étudiée est saisonnière. Il paraît qu'on peut décrire cette variation en utilisant un modèle additif puisque les fluctuations saisonnières sont quasiment de même taille et ne dépendent pas du niveau la série temporelle, ainsi que les fluctuations aléatoires semblent constantes (à partir de 2006).

2.4 Month-plot

Pour confirmer le résultat précédent, on peut regarder le tracé de la série par mois (month-plot).

```
monthplot(quantity)
```

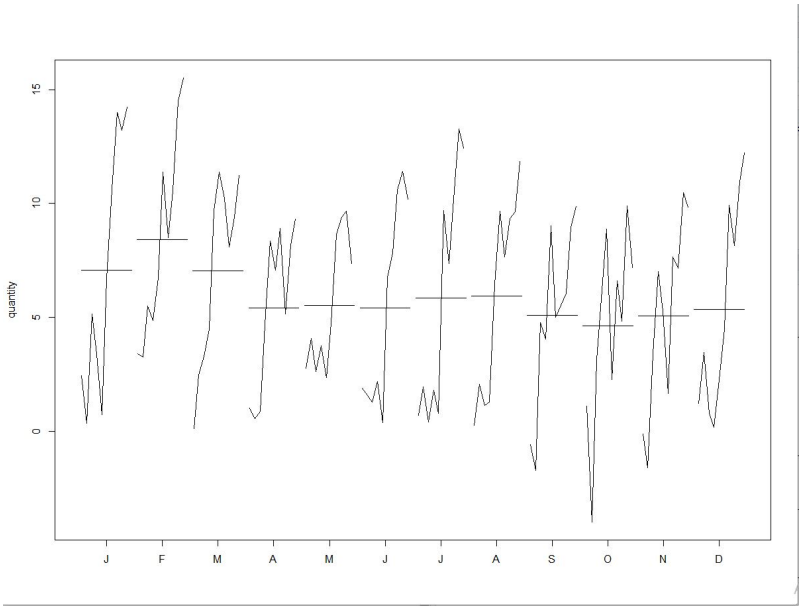

Figure 1.9: le plot de la série par mois

En absence d'effet saisonnier, les 12 chronogrammes mensuels seraient à peu près identiques, ce qui n'est pas le cas ici. D'ailleurs, on remarque que le pic des valeurs de la quantité est atteint en hiver, plus précisement aux deux mois janvier et février. Entre Mars et Juin, la quantité est instable : elle décroit de Mars à Avril puis croit d'Avril jusqu'à Juin. Puis, la chute est remarquable pendant la période Juillet-Octobre.

2.5 Etude des autocorrélations

L'autocorrélation (ou l'autocovariance) d'une série fait référence au fait que dans une série temporelle ou spatiale, la mesure d'un phénomène à un instant t peut être corrélée aux mesures précédentes aux temps t−1, t−2, t−3, etc. ou aux mesures suivantes à t+1, t+2, t+3, etc. Donc le rôle de l'autocorrélation est de détecter des régularités, des profils répétés.

On peut calculer l'autocorrélation de la série (pour différents décalages k) de la manière suivante:

result_acf=acf(quantity) .

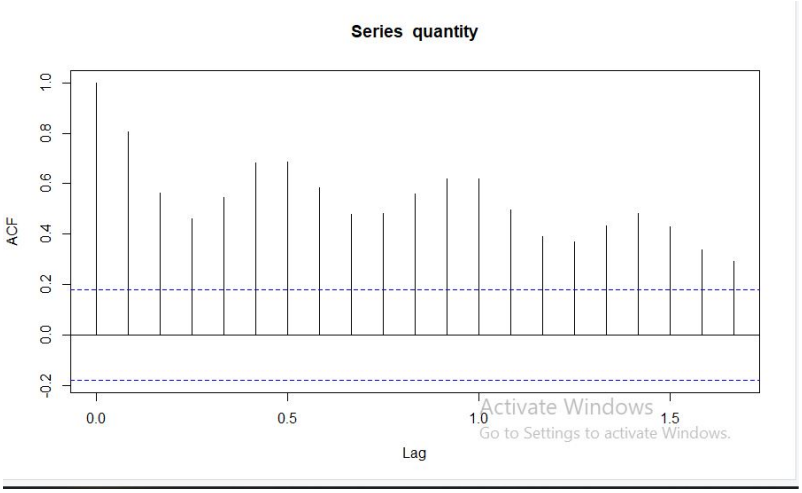

Figure 1.10: ACF

```
> print(data.frame(result_acf$lag,result_acf$acf)[1:10,])
   result_acf.lag result_acf.acf
1      0.00000000      1.0000000
2      0.08333333      0.7896766
3      0.16666667      0.5531448
4      0.25000000      0.4570073
5      0.33333333      0.5357914
6      0.41666667      0.6701556
7      0.50000000      0.6729905
8      0.58333333      0.5729767
9      0.66666667      0.4746498
10     0.75000000      0.4745295
```

Figure 1.11: les 10 premières valeurs d'ACF

L'auto-corrélation est particulièrement forte pour les lags 0.1 (acf=0.8) et 0.5 (acf=0.58). On vérifie ci-après que la série est très corrélée avec elle-même pour un lag égal à 0.1 :

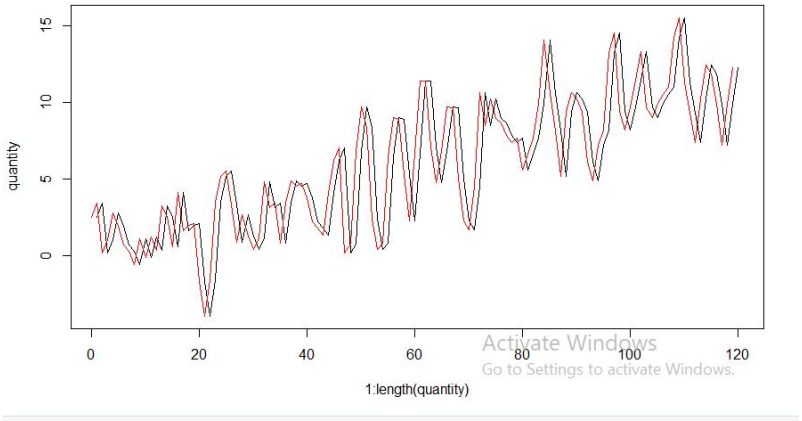

Figure 1.12: Vérification de l'auto-corrélation de la série pour un lag égal à 0.1

2.6 Autocorrélation partielle

On poursuit avec l'autocorrélation, mais cette fois-ci on va prendre en considération une mesure dérivée de l'autocorrélation : l'autocorrélation partielle. En effet, L'autocorrélation partielle *pacf* permet de mesurer l'autocorrélation d'un signal pour un décalage k "indépendamment" des autocorrélations pour les décalages inférieurs.

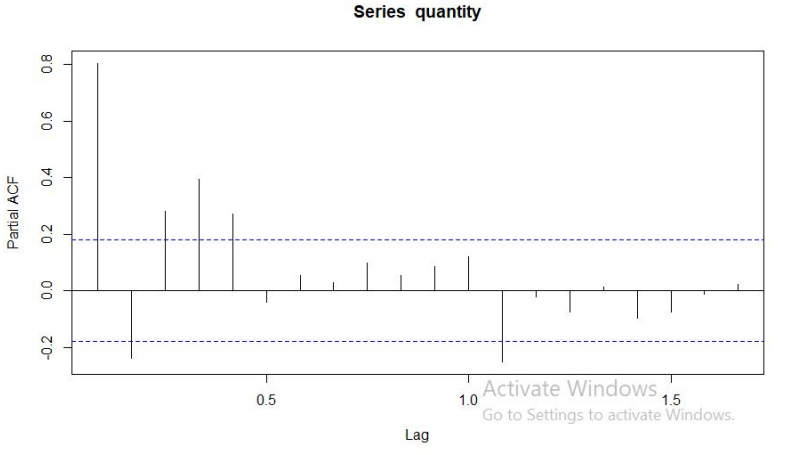

Figure 1.13: pacf

```
> print(data.frame(result_pacf$lag,result_pacf$acf)[1:10,])
   result_pacf.lag result_pacf.acf
1       0.08333333      0.80424539
2       0.16666667     -0.23847820
3       0.25000000      0.27936729
4       0.33333333      0.39264200
5       0.41666667      0.27197592
6       0.50000000     -0.04149778
7       0.58333333      0.05402443
8       0.66666667      0.02951925
9       0.75000000      0.09728280
10      0.83333333      0.05406557
```

Figure 1.14: Résultats pacf

On remarque d'après le graphe et le tableau que les pacf observées aux lags 0.4 et 0.5 sont faibles. Donc l'importance de l'autocorrélation en ces deux points est due à un effet résiduel de l'autocorrélation pour des lags inférieurs.

Chapter 2

Estimation de la tendance et de la saisonnalité

1 Estimation par la fonction "decompose"

En plus de la tendance et des composantes aléatoires, une série chronologique saisonnière a une comosante saisonnière. La décomposition d'une série temporelle saisonnière consiste en la séparation de la série en ses trois composantes.

Comme la série semble avoir à la fois une tendance et une composante saisonnière, nous pouvons nous aider des fonctions decompose() et stl() de R pour les modéliser.

```
fit1 <- decompose(quantity)
plot(fit1)
```

Figure 2.1: Code de decompose()

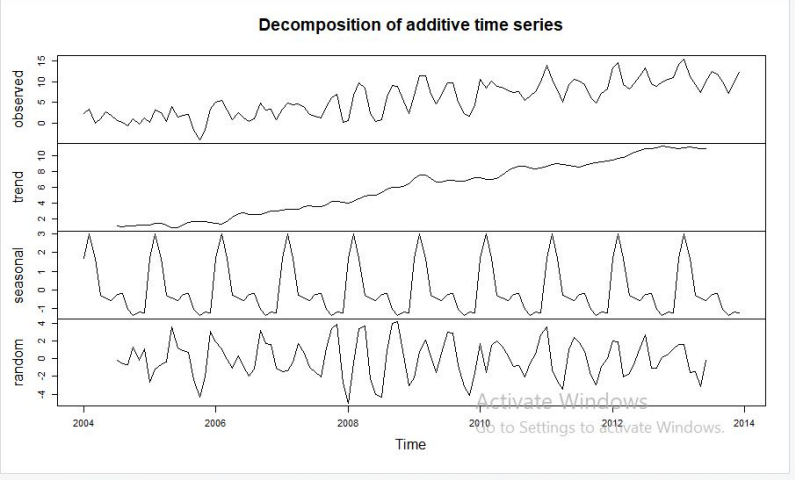

Figure 2.2: Décompostion en séries chronologiques additives

La fonction *decompose* donne une tendance croissante de la série de 2006 jusqu'à 2013, puis une stabilisation à partir de 2013, ainsi qu'un motif périodique. Mais au niveau des résidus, on remarque que leur variance est importante par rapport à la variation des valeurs de la série et qu'ils dépendent du temps. Ce qui suggère que le modèle additif n'est pas le modèle adéquat.

On peut observer également la qualité de l'estimation en superposant les estimateurs de la tendance et de la saisonnalité au chronogramme.

```
plot(quantity,xlab='Temps',ylab="Evolution de la quantité",main='decompose( ) avec modèle additif')
points(fit1$trend,type='l',col=2)
points(fit1$trend+fit1$seasonal,type='l',col='purple')
legend('topleft',c(expression(X[t]),expression(m[t]),expression(m[t]+s[t])),col=c (1,2,'purple'),lty=1)
```

Figure 2.3: code de superposition des estimateurs de la tendance et de la saisonnalité au chronogramme(décomp additive)

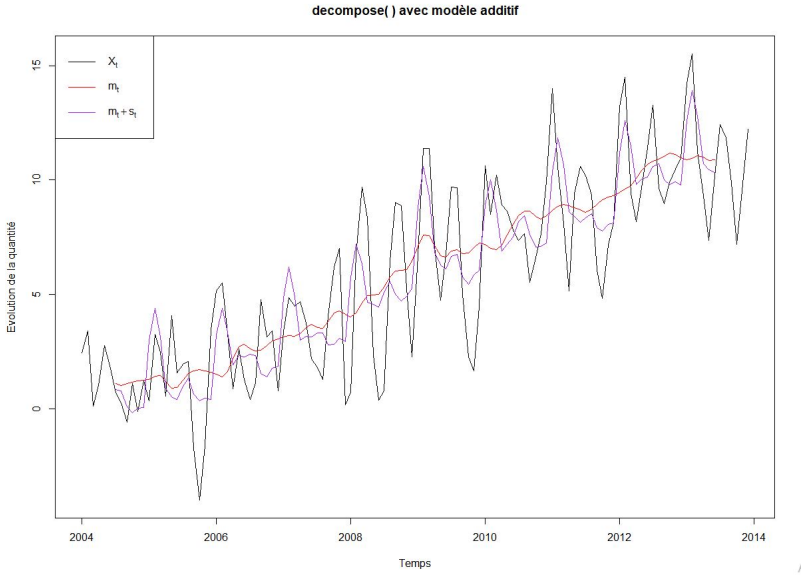

Figure 2.4: superposition des estimateurs de la tendance et de la saisonnalité au chronogramme (décomp additive)

On voit que le modèle sur-estime quelques petites valeurs, mais généralement ce modèle représente une sous-estimation de la variation de la quantité qui varie. Donc le modèle additif est loin d'être le modèle adéquat à notre dataset. C'est pourquoi on va tester l'adéquation du modèle multiplicatif à la variation des valeurs.

```
fit2 <- decompose(quantity,type='multiplicative')
plot(fit2)
```

Figure 2.5: Code de la décomposition en série multiplicative

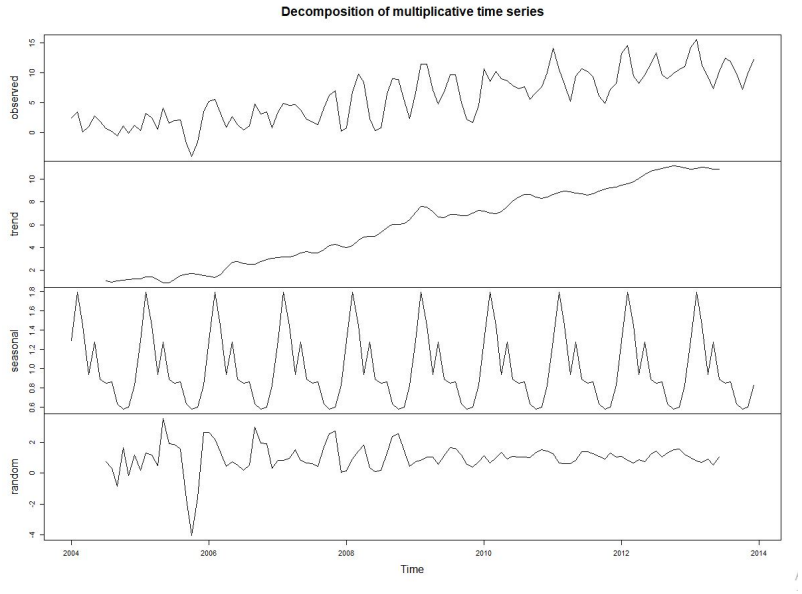

Figure 2.6: Décomposition en série multiplicative

Le modèle multiplicatif nous donne plus ou moins la même tendance et la même saisonnalité. Mais au niveau des résidus, on remarque que leur variance diminue considérablement et qu'ils ne dépendent presque plus du temps à partir de 2009. Donc on peut conclure que ce modèle est mieux adapté pour la série.

```
plot(quantity,xlab='Temps',ylab="Evolution de la quantité",main='decompose( ) avec modèle multiplicatif')
points(fit2$trend,type='l',col=2)
points(fit2$trend*fit2$seasonal,type='l',col='purple')
legend('topright',c(expression(X[t]),expression(m[t]),expression(m[t]*s[t])),col=c (1,2,'purple'),lty=1)
```

Figure 2.7: code de superposition des estimateurs de la tendance et de la saisonnalité(décomposition multiplicative)

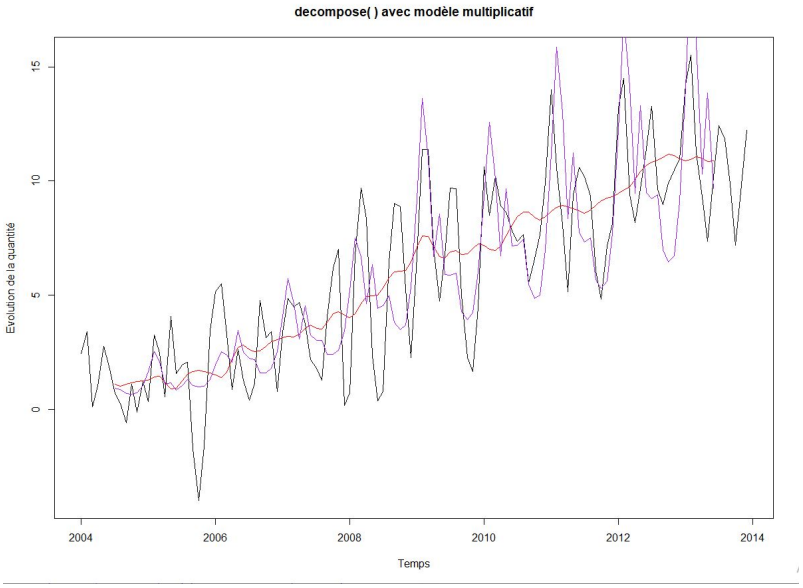

Figure 2.8: superposition des estimateurs de la tendance et de la saisonnalité(décomposition multiplicative)

Le tracé ci-dessus confirme cette idée, car la fonction $m_t * s_t$ semble être plus proche de la série que le tracé précédent. Mais cela n'empêche qu'il y a des erreurs de sur-estimation.
En résumé, si on veut décider entre la décomposition additive et la décomposition multiplicative, on doit choisir la deuxième.

```
plot(fit2$figure,type='l',xlab='mois',ylab='motif périodique')
```

Figure 2.9: Code du traçage de la courbe de variation de la composante saisonnière

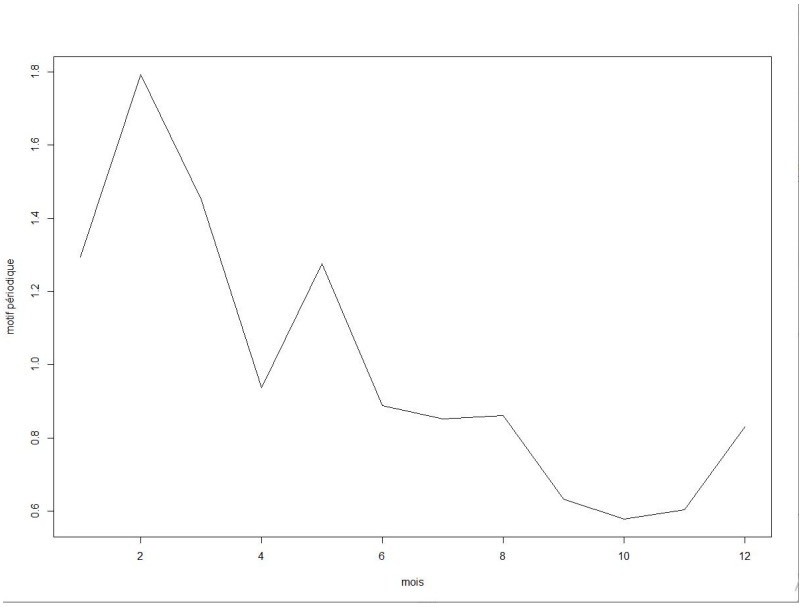

Figure 2.10: Variation de la composante saisonnière

L'observation du motif de variation de la saisonnalité montre que le pic est atteint en février, que les valeurs à la saison d'hiver(jan-fev-mar) sont les plus grandes. Puis à partir de juin, il se passe une diminution considérable des valeurs jusqu'à atteindre le minimum global en octobre.

2 Estimation par la fonction "stl"

```
quantity <- ts(quantity,start=2004,frequency=12,end=2014)
fit3 <- stl(quantity,s.window=12)
plot(fit3)
```

Figure 2.11: code d'exécution de la fonction "stl"

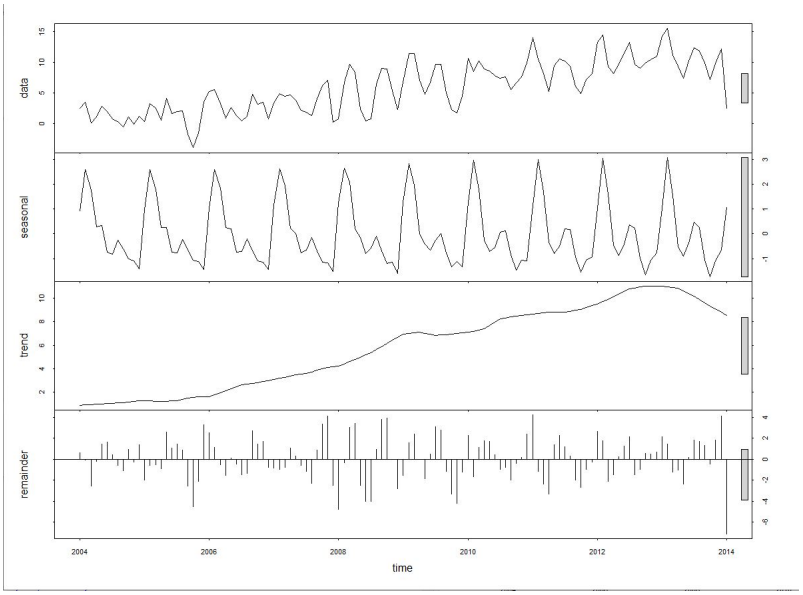

Figure 2.12: stl plot

Le résultat de la fonction stl() nous donne une tendance et une saisonnalité d'allure similaire, mais avec une amplitude croissante et non constante pour la saisonnalité. L'observation des résidus indique également une variance dépendant du temps ce qui n'est pas très satisfaisant.

```
plot(quantity,xlab='Temps',ylab="Evolution de la quantité",main='stl')
points(fit3$time.series[,2],type='l',col=2)
points(fit3$time.series[,2]+fit3$time.series[,1],type='l',col='purple')
legend('topleft',c(expression(X[t]),expression(m[t]),expression(m[t]+s[t])),col=c (1,2,'purple'),lty=1)
```

Figure 2.13: code de superposition du chronogramme et des estimateurs de la tendance et de la saisonnalité par la fonction stl()

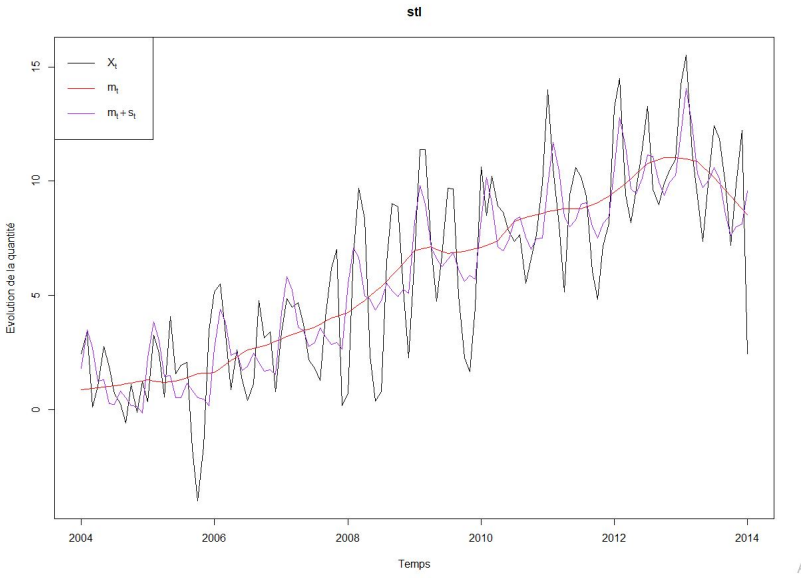

Figure 2.14: superposition du chronogramme et des estimateurs de la tendance et de la saisonnalité par la fonction stl()

Pour le cas étudié, la fonction stl() est plus performante car l'estimateur qu'elle donne est plus précis et proche des données réelles.

Chapter 3

Prédiction

On va tester différentes méthodes de lissage exponentiel. Pour évaluer leur performance prédictive, nous allons estimer les paramètres du modèle sur la série allant de janvier 2004 jusqu'à décembre 2012, celle-ci sera nommée "train". Et on va garder les observations de l'année 2013 pour les comparer avec les prévisions, on le nomme "test".
On importe évidemment le package nécessaire pour faire les prédictions "Forecast".

```
library("forecast")

train <- window(quantity,start=2004,end=c(2012,12))
test <- window(quantity,start=2013)
```

Figure 3.1: Préparation des données de training et de test

1 Lissage exponentiel

On commence par le lissage exponentiel simple. La commande suivante estime les paramètres du modèle de lissage exponentiel simple avec erreur additive.

```
fitLES <- ets(train,model="ANN")
```

Figure 3.2: Lissage exponentiel simple LES

On exécute la fonction *summary* pour voir la performance de la méthode.

summary(fitLES)

```
> summary(fitLES)
ETS(A,N,N)

Call:
 ets(y = train, model = "ANN")

  Smoothing parameters:
    alpha = 0.1676

  Initial states:
    l = 1.6214

  sigma:  2.6729

     AIC     AICc      BIC
722.0115 722.2423 730.0579

Training set error measures:
                    ME     RMSE      MAE       MPE     MAPE      MASE   ACF1
Training set 0.4756222 2.647994 2.086283 -50.77842 120.7613 0.8186975 0.4366
```

Figure 3.3: summary(fitLES)

L'exécution de la commande donne une estimation par maximum de vraisemblance des paramètres du modèle (α, l, σ), les valeurs des critères AIC, AICc et BIC ainsi que différentes mesures d'erreurs. Ces critères sont importants pour juger sur la précision et la performance d'une méthode de Forcasting. Par exemple, plus AIC est petit plus le forecasting est meilleur. On va prendre en considération ces coefficients ainsi que la courbe de Forecasting afin de juger sur sa qualité.
Notre but est de trouver la meilleure méthode de Forecasting parmi les trois méthodes envisageables: LES, LED et Holt-Winters, afin de prévoir l'évolution des valeurs dans les deux prochaines années.

La fonction *forecast()* permet de prédire à l'aide du modèle généré par la fonction ets().

```
predLES <- forecast(fitLES,h=36)
plot(predLES)
points(test,type='l',col='red',lwd=2)
legend('top',c("Valeurs observées","Prédictions"),col=c("red","blue"),lty=rep(1,2),lwd = rep(2,2))
```

Figure 3.4: code de prédiction par LES

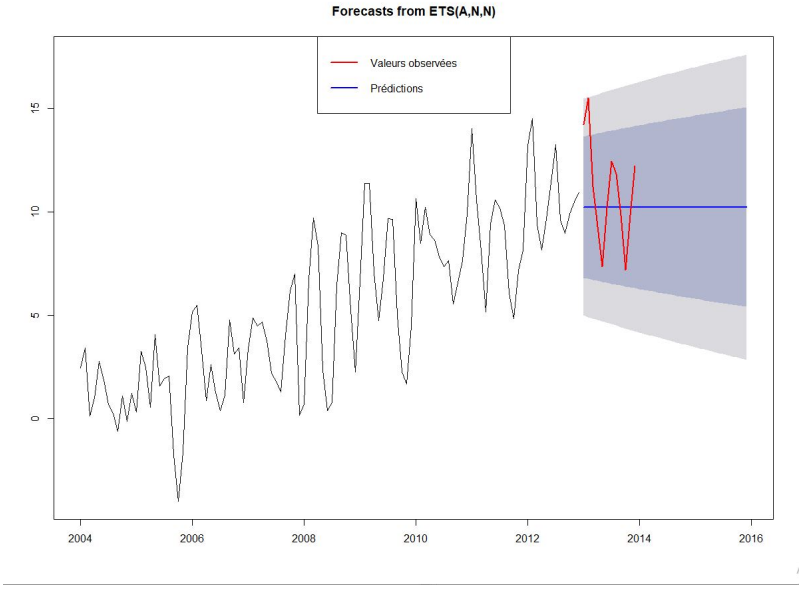

Figure 3.5: Plot de la prédiction par LES

On remarque que le modèle est trop basique dans notre cas pour prédire à l'horizon 12 et que l'intervalle de confiance à 80%, bien que très large, ne contient pas les vraies valeurs de la série.

2 Lissage exponentiel double

```
fitLED <- ets(train,model="AAN")
summary(fitLED)
```

Figure 3.6: Lissage Exponentiel double LED

```
> summary(fitLED)
ETS(A,A,N)

Call:
 ets(y = train, model = "AAN")

  Smoothing parameters:
    alpha = 1e-04
    beta  = 1e-04

  Initial states:
    l = -0.2235
    b = 0.1015

  sigma:  2.5223

     AIC     AICc      BIC
711.4300 712.0183 724.8407

Training set error measures:
                     ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
Training set 0.01719258 2.475137 1.939479 -50.91675 113.7163 0.7610888 0.4599995
```

Figure 3.7: Summary LED

```
predLED <- forecast(fitLED,h=36)
plot(predLED)
points(test,type='l',col='red',lwd=2)
legend('top',c("Valeurs observées","Prédictions"),col=c("red","blue"),lty=rep(1,2),lwd = rep(2,2))
```

Figure 3.8: code de Prédiction par LED

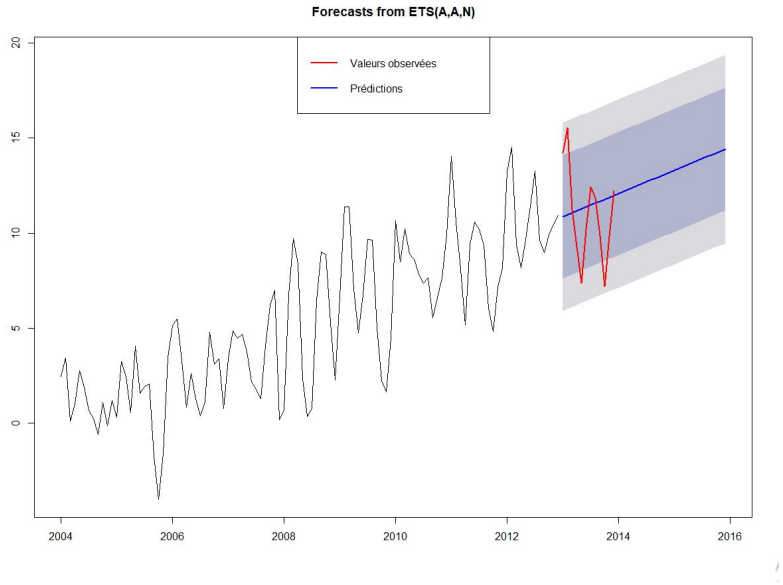

Figure 3.9: plot de Prédiction par LED

Le modèle étant plus général, l'erreur moyenne est nécessairement inférieure mais cette méthode ne semble pas préférable à la précédente, le paramètre estimé est très petit, l'AIC, l'AICc et le BIC sont plus grands et l'erreur moyenne relative est à peu près du même ordre.

3 Méthode de Holt-Winters

On prend maintenant en compte la composante saisonnière.

```
fitHW <- ets(quantity,model="AAA")
summary(fitHW)
```

Figure 3.10: Méthode Holt-Winters

```
> summary(fitHW)
ETS(A,A,A)

Call:
 ets(y = quantity, model = "AAA")

  Smoothing parameters:
    alpha = 0.0476
    beta  = 1e-04
    gamma = 1e-04

  Initial states:
    l = 0.5192
    b = 0.0953
    s = -1.3015 -1.2122 -1.5634 -0.9168 -0.067 0.031
           -0.5504 -0.2838 -0.089 1.7609 2.9668 1.2255

  sigma:  2.3205

     AIC     AICc      BIC
793.3511 799.3511 840.7384

Training set error measures:
                     ME     RMSE     MAE       MPE     MAPE      MASE      ACF1
Training set -0.0231747 2.160235 1.77308 -64.99214 109.5012 0.7316085 0.4055803
```

Figure 3.11: Summary Holt-Winters

```
predHW <- forecast(fitHW,h=24)
plot(predHW)
points(test,type='l',col='red',lwd=2)
legend('top',c("Valeurs observées","Prédictions"),col=c("red","blue"),lty=rep(1,2),lwd = rep(2,2))
```

Figure 3.12: Code du Forecasting

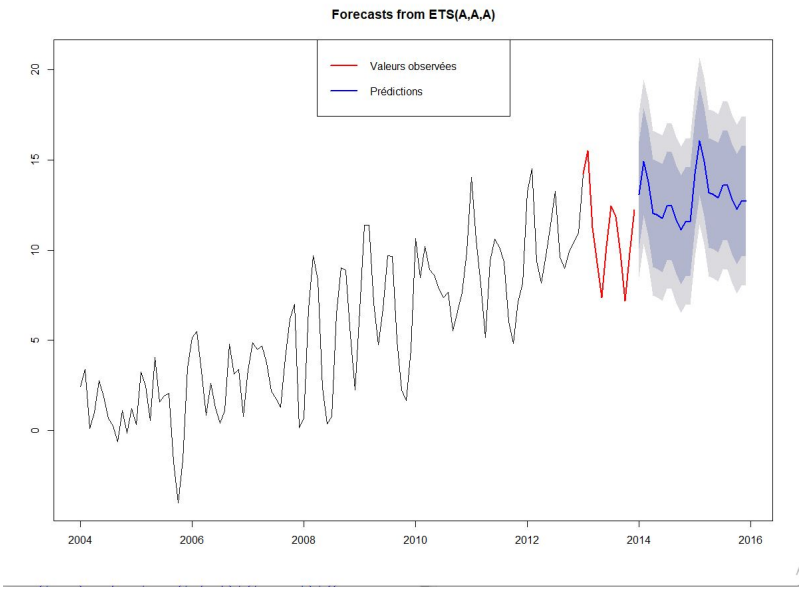

Figure 3.13: Forecasting des quantités au cours de la période 2014-2015

C'est bien remarquable que la qualité de forecasting s'est beaucoup améliorée en appliquant la méthode Holt-Winters : c'est la méthode la plus convenable.

Conclusion

Les deux méthodes de lissage exponentiel simple et double ne prennent pas en compte la saisonnalité. Par conséquent, elles donnent des prédictions très éloignées de la réalité pour la série. Par contre, la méthode de Holt-Winters donne des prédictions logiques et proches de la réalité. En effet, grâce à cette méthode, on a pu prévoir la variation de la quantité étudiée au cours des deux prochaines années (2014-2015).