# Forecast package et séries temporelles

## *Fait par Arnaud et Nina*

### 20 Novembre 2020

## Présentation

Le package “Forecast” est fait de méthodes et outils pour afficher et analyser des prévisions de séries temporelles univariées, y compris le lissage exponentiel via des modèles d’espace d’états et la modélisation automatique ARIMA. Tout au long de cette présentation nous allons utiliser

### Importation des packages désirés

library(fpp2)

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

## -- Attaching packages ---------------------------------------------- fpp2 2.4 --

## v ggplot2 3.3.2 v fma 2.4   
## v forecast 8.13 v expsmooth 2.3

### Importation des données

groupe\_st = read.csv("C:/Users/jfurs/r/groupest.csv",sep = ";", header =T)

### Montrer les données de la dataframe

attach(groupe\_st)  
str(groupe\_st)

## 'data.frame': 254 obs. of 8 variables:  
## $ Month : chr "Jan-00" "Feb-00" "Mar-00" "Apr-00" ...  
## $ Paris...Charles.de.Gaulle: chr "3 223 328" "3 289 676" "3 891 206" "4 221 430" ...  
## $ Paris...Orly : chr "1 935 261" "1 942 750" "2 204 640" "2 266 448" ...  
## $ Total : chr "5 158 589" "5 232 426" "6 095 846" "6 487 878" ...  
## $ X : logi NA NA NA NA NA NA ...  
## $ X.1 : logi NA NA NA NA NA NA ...  
## $ X.2 : logi NA NA NA NA NA NA ...  
## $ X.3 : logi NA NA NA NA NA NA ...

### Modification de l’intitulé des variables

names(groupe\_st)[1]="Mois"  
names(groupe\_st)[2]="CDG"  
names(groupe\_st)[3]="ORLY"

### Vérification du type de données

class(Total)

## [1] "character"

### Changement du type de données

Ici, on veut changer le type de données car on veut travailler avec des chiffres et si les données sont de type “character”, ne pourrons pas afficher les graphiques proprement.

groupe\_st$Total=sub(" ","",groupe\_st$Total)  
groupe\_st$Total <- as.numeric(groupe\_st$Total)

### Création de la série temporelle

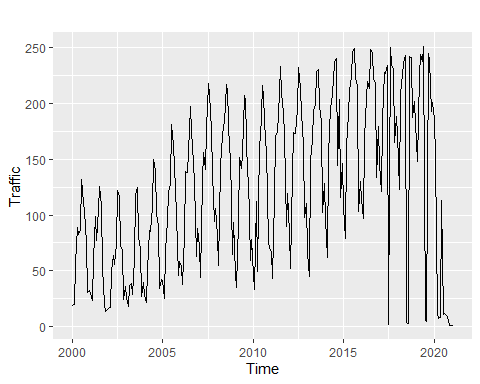
Une série temporelle est une collection de données obtenue de manière séquentielle au cours du temps. Il y a donc typiquement deux variables associées : une variable quantitative dont les données sont dépendantes et une variable ”temps”.

Nous trouverons donc par exemple : des données météorologiques, des données financières, des phénomènes ondulatoires (son, signal neuronique, etc), des suivis de populations…

Y <- ts(groupe\_st[,4],start=c(2000,1), frequency=12)

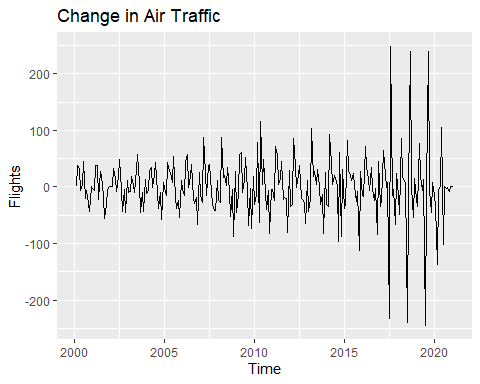
### Représentation graphique de ma data

autoplot(Y) + ylab("Traffic")



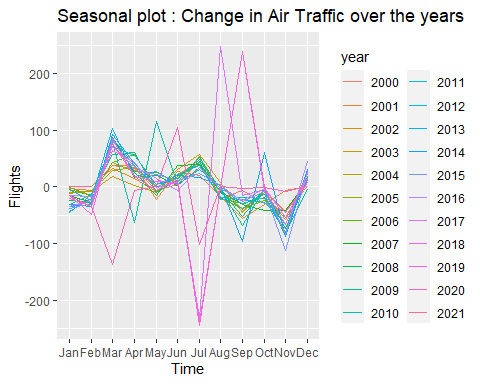
### Vérification plus précise de la différence

DY <-diff(Y)  
autoplot(DY) + ggtitle("Change in Air Traffic") + ylab("Flights")



### Vérification de la saisonnalité

ggseasonplot(DY) + ggtitle("Seasonal plot : Change in Air Traffic over the years") + ylab("Flights") + xlab("Time")



## Après avoir vu les différents facteurs d’évolution de la courbe, il est temps de faire les prédictions.

## Il existe plusieurs méthodes pour faire des prédictions : Naive, ETS (Exponential Smoothing algorithm) et le modèle ARIMA. Nous allons donc voir ces 3 modèles de prédiction.

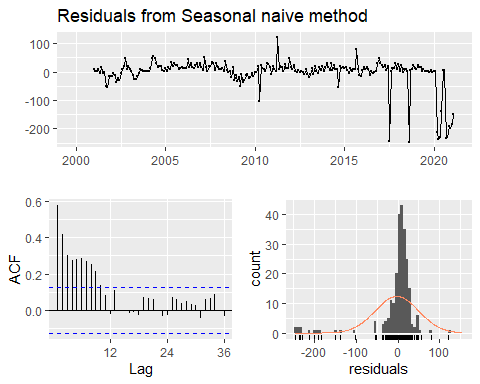
### La “naive model” méthode

I s’agit d’une technique d’estimation dans laquelle les chiffres réels de la dernière période sont utilisés comme prévision de cette période, sans les ajuster ni tenter d’établir des facteurs de causalité. Il n’est utilisé qu’à des fins de comparaison avec les prévisions générées par les meilleures techniques (sophistiquées).

fit\_n <- snaive(Y)  
print(summary(fit\_n))

##   
## Forecast method: Seasonal naive method  
##   
## Model Information:  
## Call: snaive(y = Y)   
##   
## Residual sd: 51.9895   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set -2.698347 51.98947 25.90496 -435.2016 457.3396 1 0.5774185  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Mar 2021 13 -53.62719 79.62719 -88.89749 114.8975  
## Apr 2021 7 -59.62719 73.62719 -94.89749 108.8975  
## May 2021 8 -58.62719 74.62719 -93.89749 109.8975  
## Jun 2021 113 46.37281 179.62719 11.10251 214.8975  
## Jul 2021 11 -55.62719 77.62719 -90.89749 112.8975  
## Aug 2021 12 -54.62719 78.62719 -89.89749 113.8975  
## Sep 2021 10 -56.62719 76.62719 -91.89749 111.8975  
## Oct 2021 9 -57.62719 75.62719 -92.89749 110.8975  
## Nov 2021 1 -65.62719 67.62719 -100.89749 102.8975  
## Dec 2021 1 -65.62719 67.62719 -100.89749 102.8975  
## Jan 2022 1 -65.62719 67.62719 -100.89749 102.8975  
## Feb 2022 1 -65.62719 67.62719 -100.89749 102.8975  
## Mar 2022 13 -81.22507 107.22507 -131.10481 157.1048  
## Apr 2022 7 -87.22507 101.22507 -137.10481 151.1048  
## May 2022 8 -86.22507 102.22507 -136.10481 152.1048  
## Jun 2022 113 18.77493 207.22507 -31.10481 257.1048  
## Jul 2022 11 -83.22507 105.22507 -133.10481 155.1048  
## Aug 2022 12 -82.22507 106.22507 -132.10481 156.1048  
## Sep 2022 10 -84.22507 104.22507 -134.10481 154.1048  
## Oct 2022 9 -85.22507 103.22507 -135.10481 153.1048  
## Nov 2022 1 -93.22507 95.22507 -143.10481 145.1048  
## Dec 2022 1 -93.22507 95.22507 -143.10481 145.1048  
## Jan 2023 1 -93.22507 95.22507 -143.10481 145.1048  
## Feb 2023 1 -93.22507 95.22507 -143.10481 145.1048  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Mar 2021 13 -53.62719 79.62719 -88.89749 114.8975  
## Apr 2021 7 -59.62719 73.62719 -94.89749 108.8975  
## May 2021 8 -58.62719 74.62719 -93.89749 109.8975  
## Jun 2021 113 46.37281 179.62719 11.10251 214.8975  
## Jul 2021 11 -55.62719 77.62719 -90.89749 112.8975  
## Aug 2021 12 -54.62719 78.62719 -89.89749 113.8975  
## Sep 2021 10 -56.62719 76.62719 -91.89749 111.8975  
## Oct 2021 9 -57.62719 75.62719 -92.89749 110.8975  
## Nov 2021 1 -65.62719 67.62719 -100.89749 102.8975  
## Dec 2021 1 -65.62719 67.62719 -100.89749 102.8975  
## Jan 2022 1 -65.62719 67.62719 -100.89749 102.8975  
## Feb 2022 1 -65.62719 67.62719 -100.89749 102.8975  
## Mar 2022 13 -81.22507 107.22507 -131.10481 157.1048  
## Apr 2022 7 -87.22507 101.22507 -137.10481 151.1048  
## May 2022 8 -86.22507 102.22507 -136.10481 152.1048  
## Jun 2022 113 18.77493 207.22507 -31.10481 257.1048  
## Jul 2022 11 -83.22507 105.22507 -133.10481 155.1048  
## Aug 2022 12 -82.22507 106.22507 -132.10481 156.1048  
## Sep 2022 10 -84.22507 104.22507 -134.10481 154.1048  
## Oct 2022 9 -85.22507 103.22507 -135.10481 153.1048  
## Nov 2022 1 -93.22507 95.22507 -143.10481 145.1048  
## Dec 2022 1 -93.22507 95.22507 -143.10481 145.1048  
## Jan 2023 1 -93.22507 95.22507 -143.10481 145.1048  
## Feb 2023 1 -93.22507 95.22507 -143.10481 145.1048

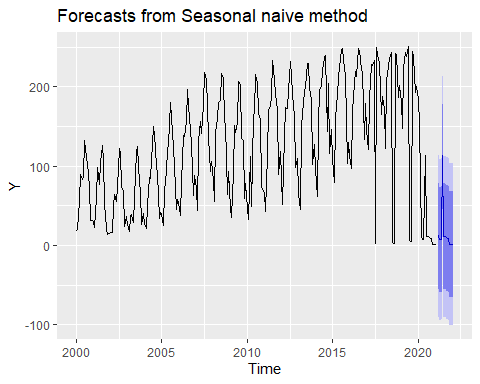
checkresiduals(fit\_n) #residual SD : 51,85



##   
## Ljung-Box test  
##   
## data: Residuals from Seasonal naive method  
## Q\* = 266.41, df = 24, p-value < 2.2e-16  
##   
## Model df: 0. Total lags used: 24

### Représentation graphique

fcts\_n <- forecast(fit\_n, h=12)  
autoplot(fcts\_n)



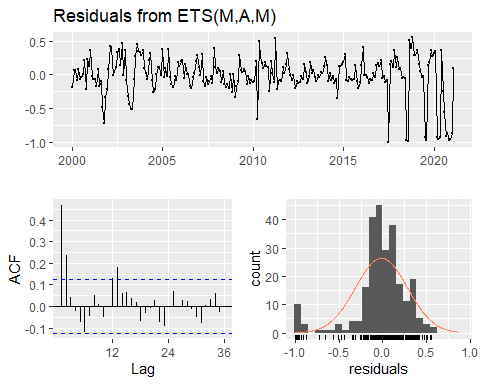
### La “ets model” méthode

La fonction .ETS est l’une des fonctions statistiques qui est utilisée pour prédire une valeur future en fonction des valeurs existantes (historiques) à l’aide de la version AAA de l’algorithme de lissage exponentiel (Exponential Smoothing, ETS).

fit\_ets <- ets(Y)  
print(summary(fit\_ets))

## ETS(M,A,M)   
##   
## Call:  
## ets(y = Y)   
##   
## Smoothing parameters:  
## alpha = 0.2609   
## beta = 0.0084   
## gamma = 0.5065   
##   
## Initial states:  
## l = 72.1534   
## b = 0.0706   
## s = 0.3896 0.3658 1.3022 1.243 1.5966 1.9702  
## 1.3038 1.1068 1.3608 0.7657 0.2735 0.322  
##   
## sigma: 0.3049  
##   
## AIC AICc BIC   
## 3175.156 3177.749 3235.290   
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set -5.174503 39.48924 22.8407 -166.6325 180.9981 0.8817116 0.2806337  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set -5.174503 39.48924 22.8407 -166.6325 180.9981 0.8817116 0.2806337

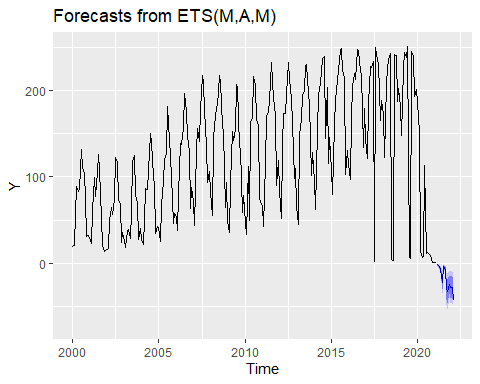
checkresiduals(fit\_ets) # residual SD = 0.3098



##   
## Ljung-Box test  
##   
## data: Residuals from ETS(M,A,M)  
## Q\* = 99.948, df = 8, p-value < 2.2e-16  
##   
## Model df: 16. Total lags used: 24

### Représentation graphique

fcts\_ets <- forecast(fit\_ets, h=12)  
autoplot(fcts\_ets)

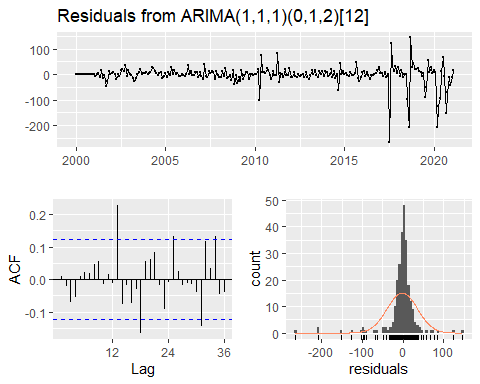


### La “arima model” méthode

fit\_arima <- auto.arima(Y)  
print(summary(fit\_arima))

## Series: Y   
## ARIMA(1,1,1)(0,1,2)[12]   
##   
## Coefficients:  
## ar1 ma1 sma1 sma2  
## 0.3721 -0.8514 -0.3476 -0.1948  
## s.e. 0.0841 0.0447 0.0753 0.0789  
##   
## sigma^2 estimated as 1489: log likelihood=-1222.36  
## AIC=2454.72 AICc=2454.98 BIC=2472.14  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -2.378156 37.26782 18.20019 -143.4614 173.9147 0.7025754  
## ACF1  
## Training set 0.009955806  
## ME RMSE MAE MPE MAPE MASE  
## Training set -2.378156 37.26782 18.20019 -143.4614 173.9147 0.7025754  
## ACF1  
## Training set 0.009955806

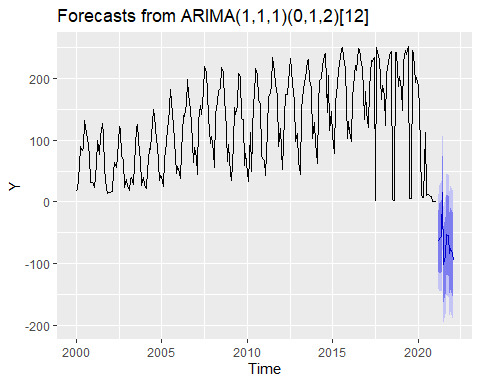
checkresiduals(fit\_arima) # residual SD = sqrt(1488) = 38,57



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(1,1,1)(0,1,2)[12]  
## Q\* = 34.954, df = 20, p-value = 0.02035  
##   
## Model df: 4. Total lags used: 24

### Représentation graphique

fcts\_arima <- forecast(fit\_arima,h=12)  
autoplot(fcts\_arima)



# THE END