TPE 3 INFO 308

Jeudi le 10 juin 2021

Membres du groupe

BAKANG EKOSSO SALMA RIHANA 17T2229 rihana.bakang@facsciences-uy1.cm

DJIEMBOU TIENTCHEU VICTOR NICO 17T2051 nico.djiembou@facsciences-uy1.cm

KENFACK TEMGOUA VANESSA 17J2871 vanessa.kenfack@facsciences-uy1.cm

TITTI URSULA SOREL 17L2969 sorel.titti@facsciences-uy1.cm

SOMMAIRE:

Table de matiere:

I : Rediger un tutoriel pour l'installation de R sur votre machine

II : En utilisant la serie temporelle **AIrPassengers** du cour effectuer :

1 : Une caracterisation complete

2 : Un lissage exponentiel simple

3 : Un lissage exponentiel double

4 : Un lissage exponentiel saisonnier additif

5 : Un lissage exponentiel saisonnier multifiplicatif

III : Sur le meme jeu de donnees donnees, appliquer la methode de difference pour identifier la periode de la saisonnalite et le degre du polynome de la tendance

Installation de R sur votre machine

Sous Mac OS

Les étapes ci-dessous permettront d'installer Homebrew, R et RStudio. Ouvrez un terminal et exécutez les commandes suivantes.

1. Installez Homebrew.

   usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

Cette commande va télécharger, installer et configurer Homebrew. La première fois qu'une commande brew est exécutée, elle demande un mot de passe.

2. Installez R avec les commandes suivantes :

   brew tap homebrew/science # adds another source for applications.

   brew install r # runs installer of r

La première commande est nécessaire si vous souhaitez ajouter des applications de type scientifique/mathématique. La deuxième commande gère toute l'installation de R lui-même. Depuis la version 3.25.2017, elle installe la version . Cela peut prendre un certain temps car Homebrew télécharge puis construit les outils, bien que cela tende à être plus rapide et plus facile que l'installation manuelle.

3. Installez RStudio avec la commande suivante :

   brew cask install rstudio

Essayez de naviguer vers RStudio pour vérifier l'installation.

4. Facultatif Configurer RStudio pour travailler à partir de la ligne de commande.

   echo "alias rstudio='open -a RStudio'" » ~/.bash_profile

   source ~/.bash_profile

Cela configure votre profil bash pour lancer RStudio lorsque rstudio est entré dans la ligne de commande (c'est-à-dire rstudio exemple.rmd). Elle ajoute un alias à votre ~/.bash_profile pour que vous puissiez le supprimer si vous le souhaitez.

Sous Ubuntu

Voici la marche à suivre, en ligne de commande :

1. Comme toujours, mettez à jour vos dépôts et vos logiciels :

   sudo apt update

   sudo apt -y upgrade

2. Installez le package qui permet d'ajouter des dépôts depuis https :

   sudo apt install apt-transport-https software-properties-common

3. Ajoutez le dépôt CRAN à votre liste de dépôts, et la clé GPG associée :

   sudo apt-key adv –keyserver keyserver.ubuntu.com –recv-keys E298A3A825C0D65DFD57\CBB6517166

   sudo add-apt-repository 'deb https://cloud.r-project.org/bin/linux/ubuntu bionic-cran35/'

4. Mettez à jour la liste des dépôts :

   sudo apt update

5. Installez le package r-base

   sudo apt install r-base

6. Installer RStudio

- Allez sur la page de téléchargement de RStudio Desktop (version Open Source) : rstudio.com/products/rstudio/download ;
- Téléchargez le fichier d'installation compatible avec votre système d'exploitation (Mac, Windows ou Linux) et installez-le ;
- Pour les utilisateurs de Linux Ubuntu : téléchargez le fichier .deb et ouvrez-le avec la Logithèque.

Sous Windows

1. Téléchargez la dernière version de R : cran.r-project.org/bin/windows/base
2. Enregistrez le fichier d'installation et exécutez-le ;
3. Vérifiez que l'installation a été effectuée en cliquant sur la magnifique icône qui s'affiche sur le bureau et dans le menu Démarrer.

Serie temporelle AirPassengers

NB: Dans chaque cas, les donnees doivent etre decoupees en donnees d'apprentisage et du test, et le RMSE doit etre calculer sur donnees de test de chaque model

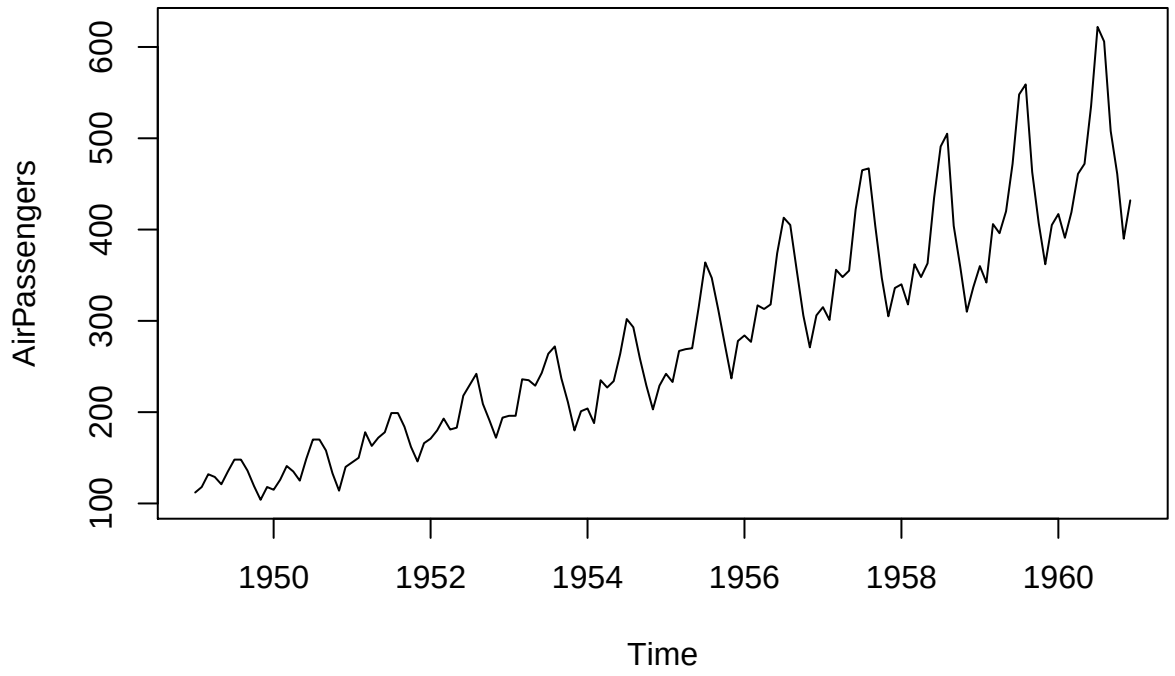
1. **Importation de AirPassengers**

```
data("AirPassengers")
str(AirPassengers)
```

```
##  Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 148 148 136 119 ...
```

2. **Affichage de la serie temporelle de AirPassengers**

```
plot(AirPassengers)
```



3. **Description de AirPassengers**

```
summary(AirPassengers)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   104.0   180.0   265.5   280.3   360.5   622.0
```

4. **Decomposer en donnees d'apprentissage et de test**

```
airpassengers_app <- ts(AirPassengers[1:115],start=c(1949,1),end=c(1957,12),freq=12)
airpassengers_test <- ts(AirPassengers[116:144],start=c(1958,1),end=c(1960,6),freq=12)
airpassengers_test
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1958 505 404 359 310 337 360 342 406 396 420 472 548
## 1959 559 463 407 362 405 417 391 419 461 472 535 622
## 1960 606 508 461 390 432 505
```

Caracterisation

1. **Moyenne de AirPassengers**

```
mean(AirPassengers)
```

```
## [1] 280.2986
```

```
mean(airpassengers_app)
```

```
## [1] 230.8981
```

```
mean(airpassengers_test)
```

```
## [1] 442.4667
```

2. **Variance de AirPassengers**

```
var(AirPassengers)
```

```
## [1] 14391.92
```

```
var(airpassengers_app)
```

```
## [1] 7322.709
```

```
var(airpassengers_test)
```

```
## [1] 6243.982
```

3. **Ecart-Type de AirPassengers**

```
sd(AirPassengers)
```

```
## [1] 119.9663
```

```
sd(airpassengers_app)
```

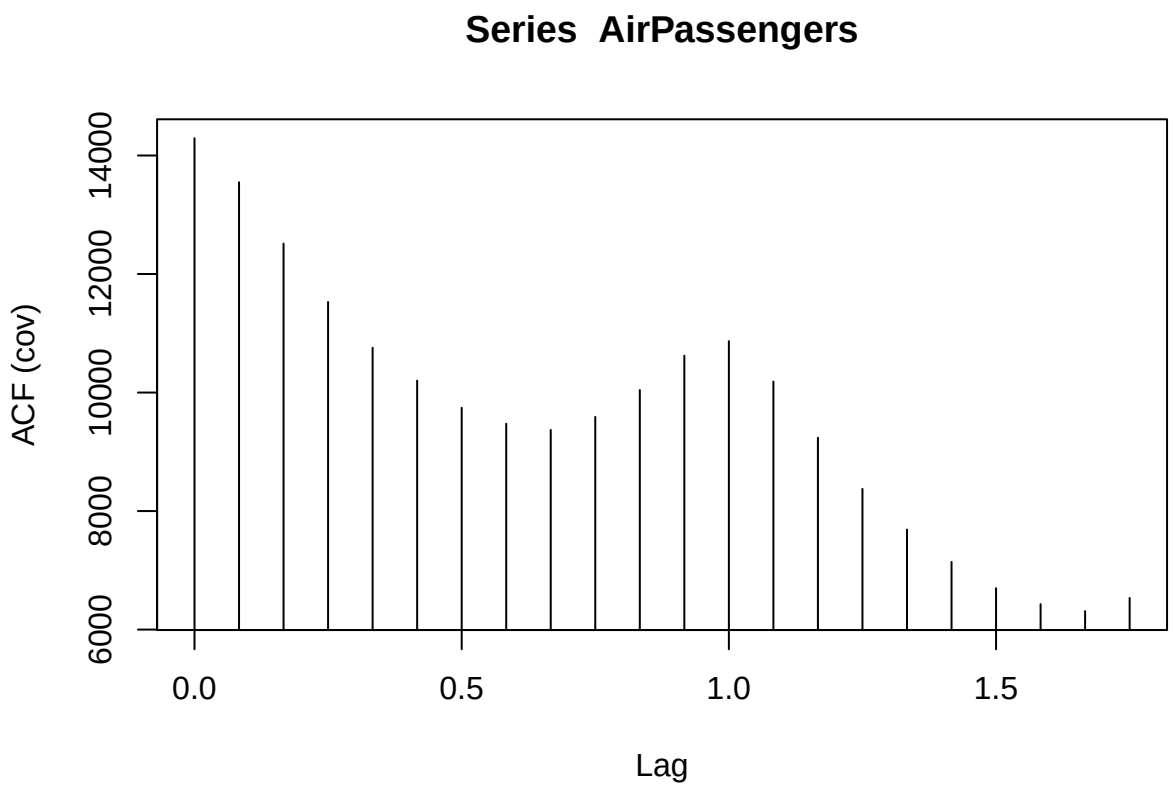
```
## [1] 85.57283
```

```
sd(airpassengers_test)
```

```
## [1] 79.01887
```

4. **Auto-Covariance de AirPassengers**

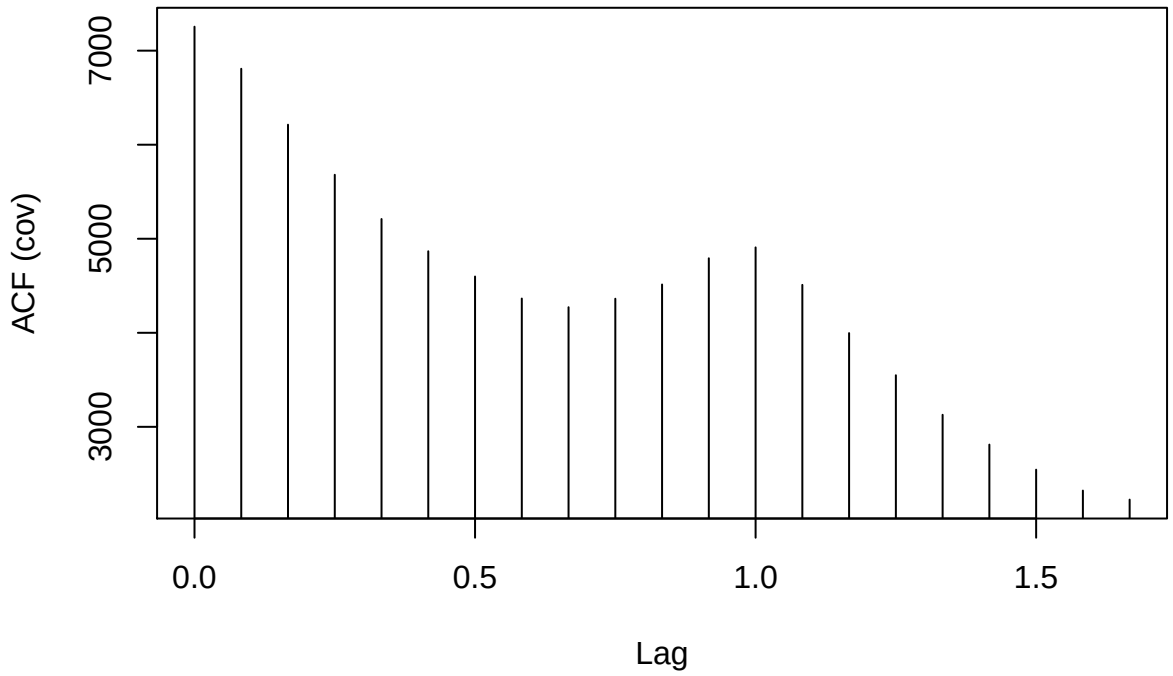
```
tmp=acf(AirPassengers,type="cov",plot = TRUE)
```




```
print(tmp)
```

```
##
## Autocovariances of series 'AirPassengers', by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500
##  14292  13549  12514  11529  10757  10201   9743   9474   9370   9589
## 0.8333 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833
##  10043  10622  10868  10185   9238   8374   7688   7142   6699   6430
## 1.6667 1.7500
##   6312   6535
```

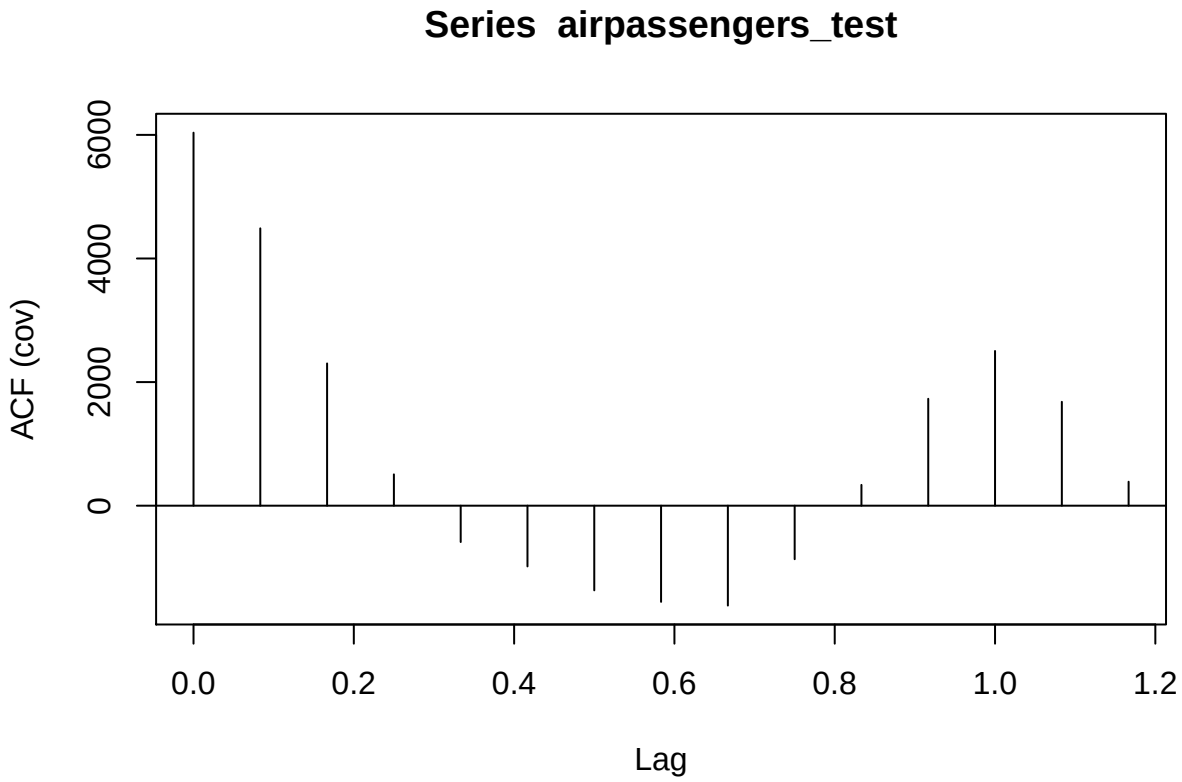
```
tmp=acf(airpassengers_app,type="cov",plot = TRUE)
```




```
print(tmp)
```

```
##
## Autocovariances of series 'airpassengers_app', by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500
##   7255   6808   6213   5680   5210   4867   4598   4364   4272   4362
## 0.8333 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833
##   4514   4793   4908   4510   3997   3549   3128   2811   2545   2322
## 1.6667
##   2226
```

```
tmp=acf(airpassengers_test,type="cov",plot = TRUE)
```

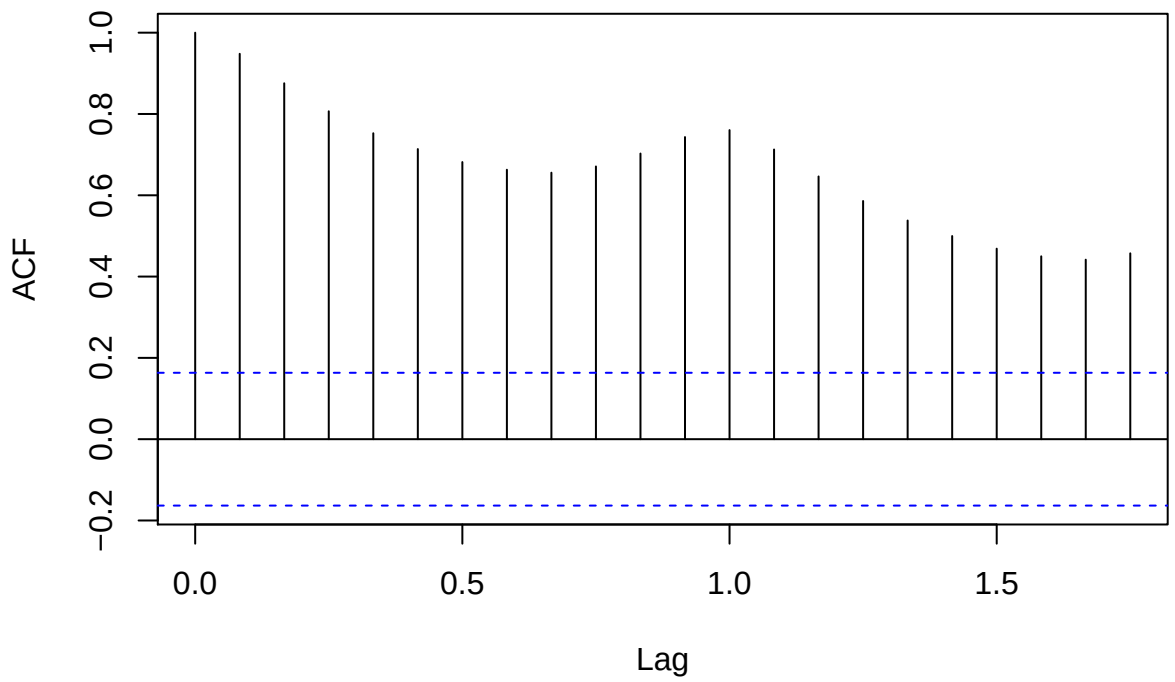



```
print(tmp)
```

```
##
## Autocovariances of series 'airpassengers_test', by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500
##   6036   4488   2302    507   -588   -982  -1370  -1556  -1616   -866
## 0.8333 0.9167 1.0000 1.0833 1.1667
##    337   1730   2504   1681    388
```

5. **Auto-Correlation de AirPassengers**

```
tmp=acf(AirPassengers,type="cor",plot = TRUE)
```

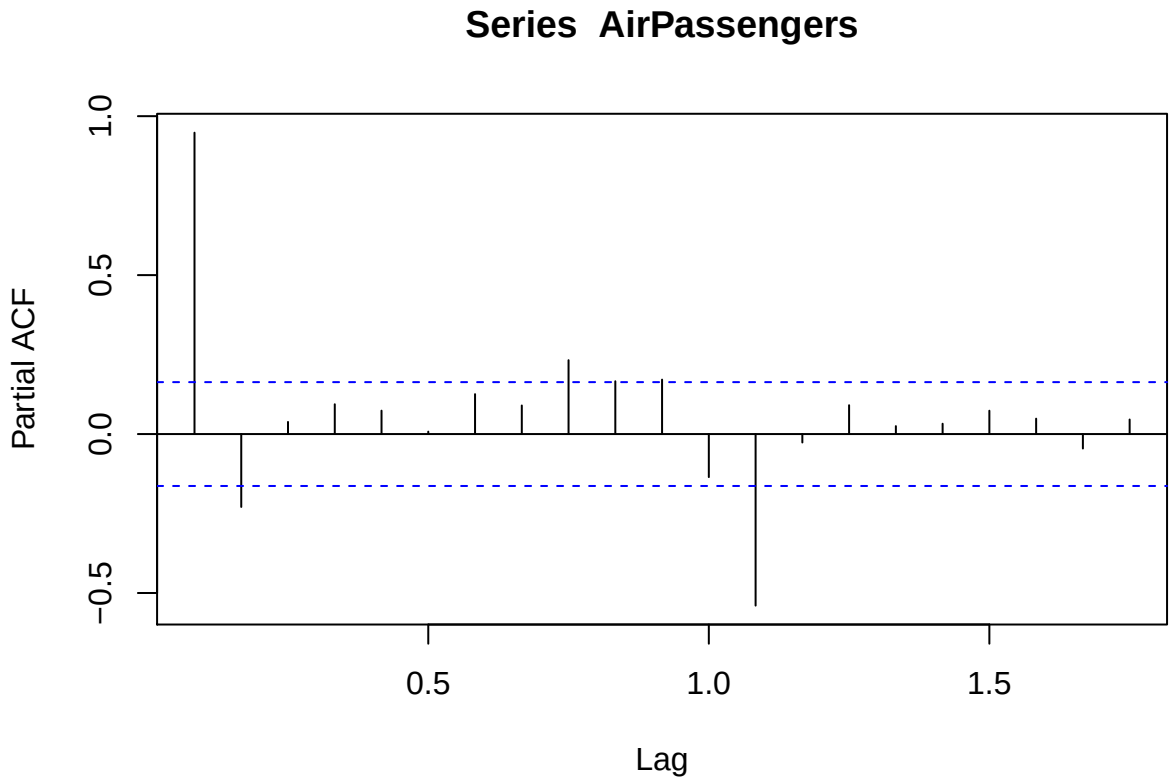



```
print(tmp)
```

```
##
## Autocorrelations of series 'AirPassengers', by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500
##  1.000  0.948  0.876  0.807  0.753  0.714  0.682  0.663  0.656  0.671
## 0.8333 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833
##  0.703  0.743  0.760  0.713  0.646  0.586  0.538  0.500  0.469  0.450
## 1.6667 1.7500
##  0.442  0.457
```

6. **Auto-Correlation Partielle de AirPassengers**

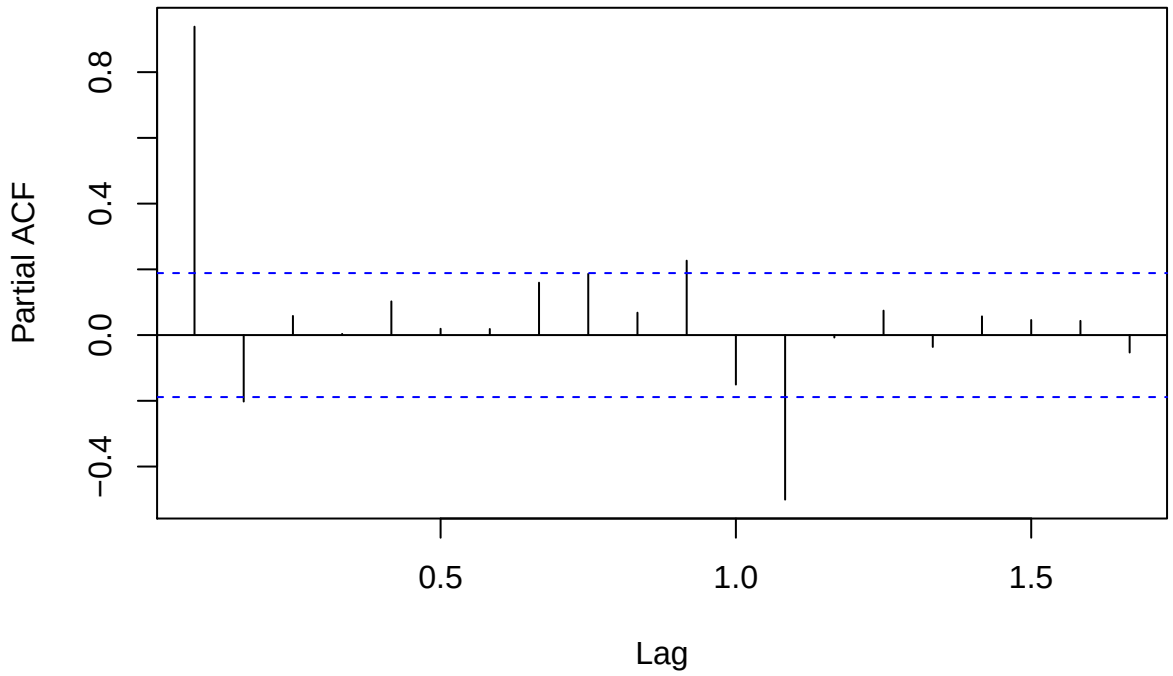
```
tmp=pacf(AirPassengers,plot = TRUE)
```




```
print(tmp)
```

```
##
## Partial autocorrelations of series 'AirPassengers', by lag
##
## 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333
##  0.948 -0.229  0.038  0.094  0.074  0.008  0.126  0.090  0.232  0.166
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
##  0.171 -0.135 -0.540 -0.027  0.091  0.025  0.033  0.073  0.048 -0.046
## 1.7500
##  0.046
```

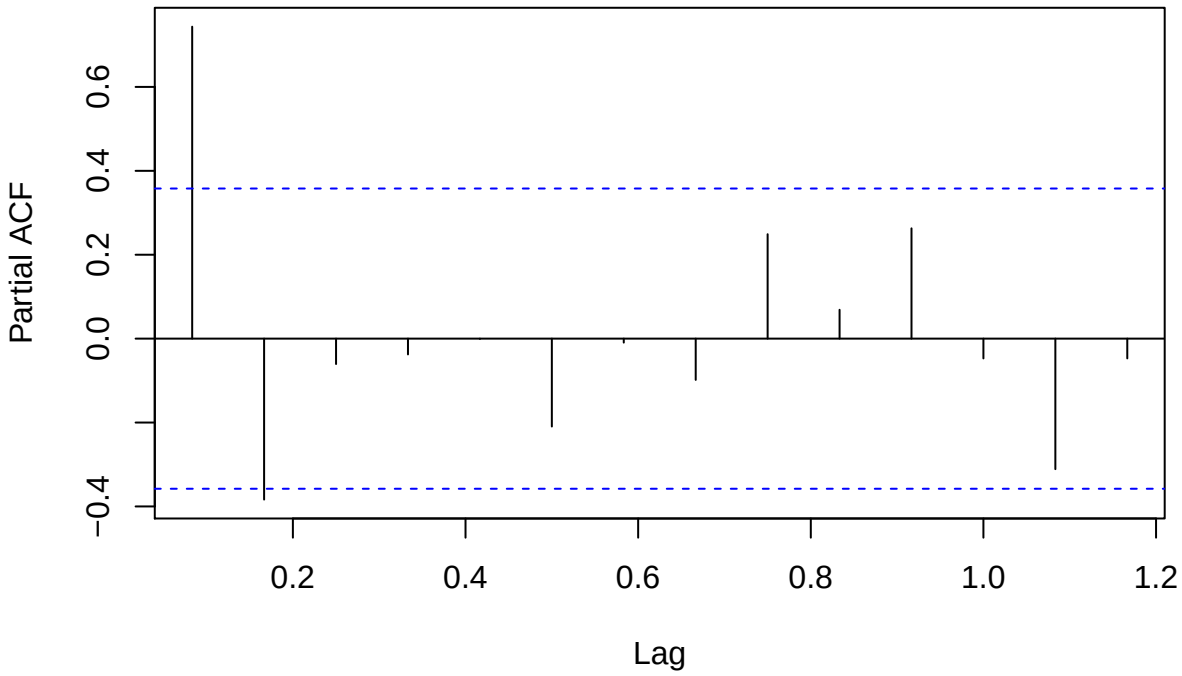
```
tmp=pacf(airpassengers_app,plot = TRUE)
```




```
print(tmp)
```

```
##
## Partial autocorrelations of series 'airpassengers_app', by lag
##
## 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333
##  0.938 -0.202  0.058  0.004  0.103  0.019  0.019  0.159  0.186  0.068
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
##  0.226 -0.151 -0.501 -0.007  0.074 -0.036  0.057  0.046  0.043 -0.053
```

```
tmp=pacf(airpassengers_test,plot = TRUE)
```

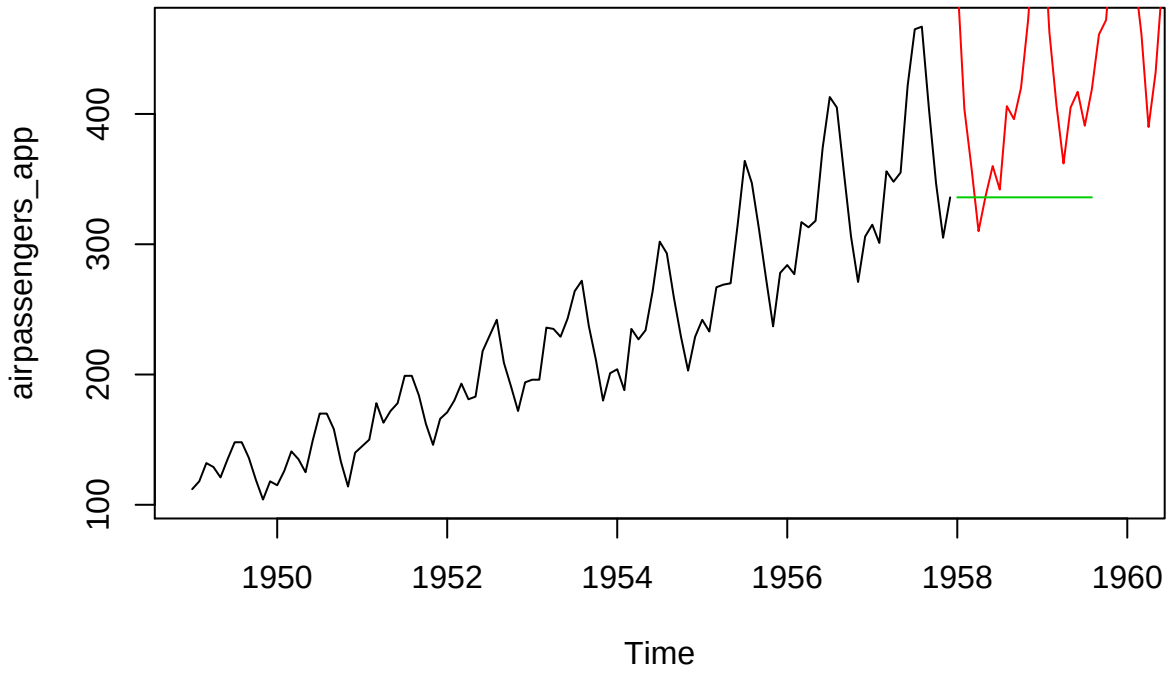



```
print(tmp)
```

```
##
## Partial autocorrelations of series 'airpassengers_test', by lag
##
## 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333
##  0.744 -0.384 -0.061 -0.038 -0.001 -0.210 -0.010 -0.098  0.249  0.069
## 0.9167 1.0000 1.0833 1.1667
##  0.263 -0.047 -0.311 -0.047
```

Lissage exponentiel simple ou LSE

```
plot(airpassengers_app,xlim=c(1949,1960))
lines(airpassengers_test,col=2)
LES=HoltWinters(airpassengers_app,alpha=NULL,beta=FALSE,gamma=FALSE)
p1<-predict(LES,n.ahead=20)
lines(p1,col=3)
```




```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

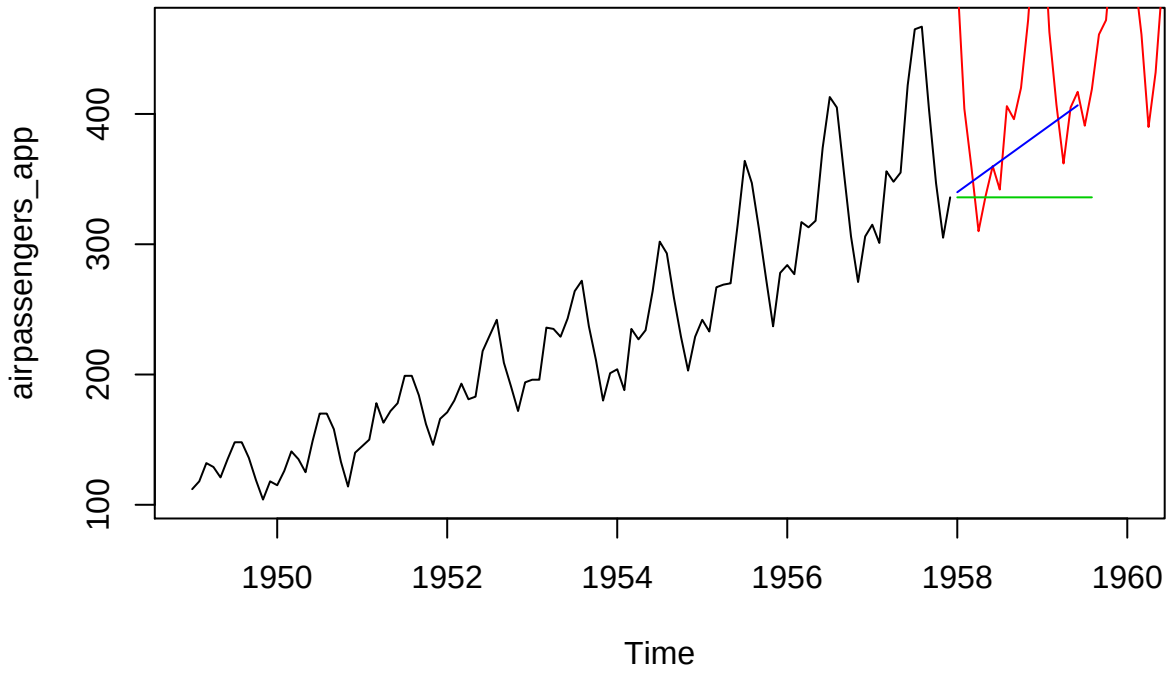
```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
caret::RMSE(p1,airpassengers_test)
```

```
## [1] 101.6898
```

Lissage exponentiel double

```
LES2=HoltWinters(airpassengers_app,alpha=NULL,beta=NULL,gamma=FALSE)
p2<-predict(LES2,n.ahead=18)
plot(airpassengers_app,xlim=c(1949,1960))
lines(airpassengers_test,col=2)
lines(p1,col=3)
lines(p2,col=4)
```

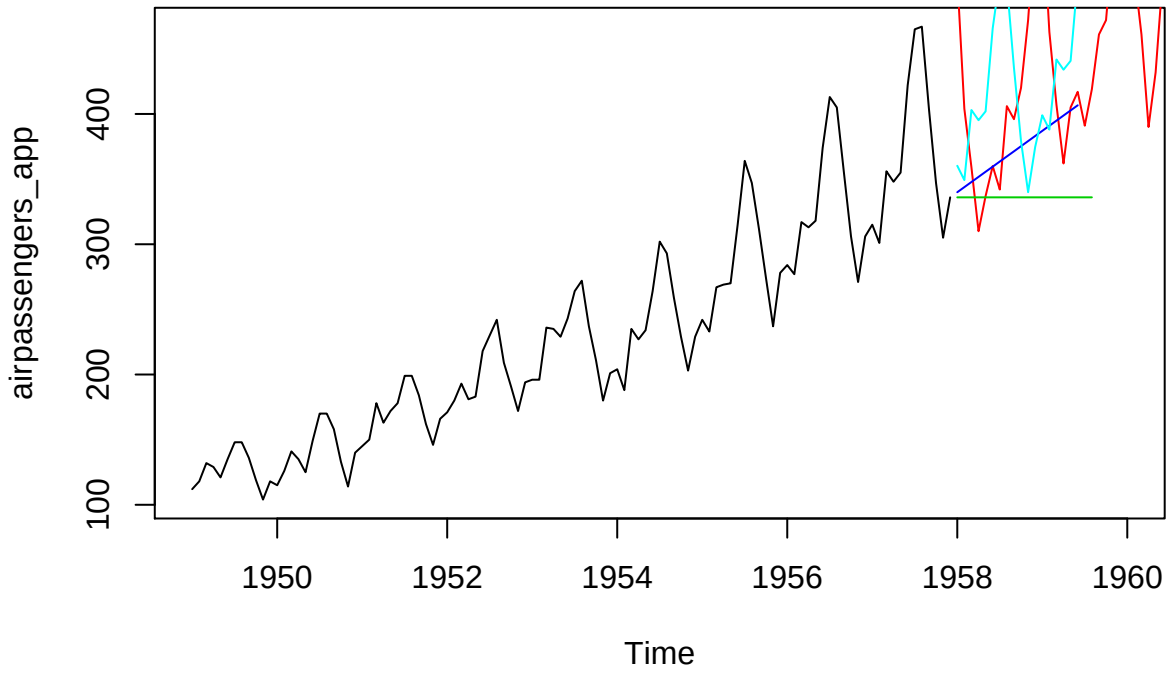



```
caret::RMSE(airpassengers_test,p2)
```

```
## [1] 78.11632
```

Lissage exponentiel saisonnier additif

```
LES3=HoltWinters(airpassengers_app,alpha=NULL,beta=NULL,gamma=NULL)
p3<-predict(LES3,n.ahead=18)
plot(airpassengers_app,xlim=c(1949,1960))
lines(airpassengers_test,col=2)
lines(p1,col=3)
lines(p2,col=4)
lines(p3,col=5)
```

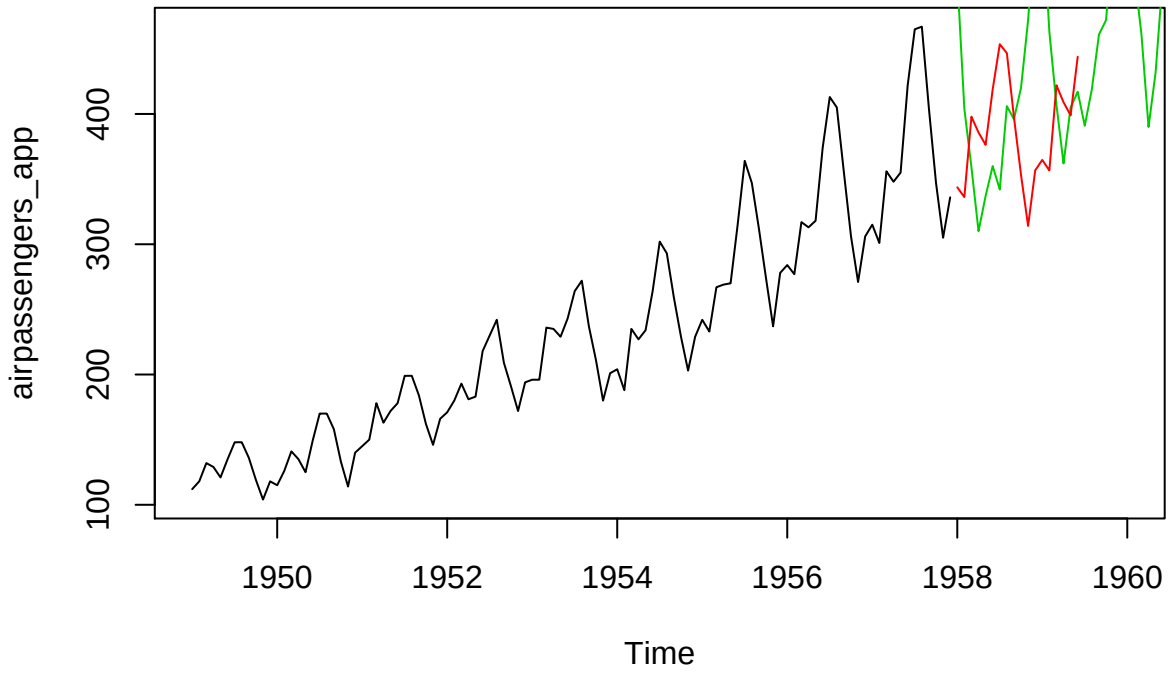



```
caret::RMSE(airpassengers_test,p3)
```

```
## [1] 100.3429
```

Lissage exponentiel saisonnier multifiplicatif

```
LES=HoltWinters(airpassengers_app,alpha=NULL,beta=NULL,gamma=NULL,seasonal = "multi")
plot(airpassengers_app,xlim=c(1949,1960))
lines(airpassengers_test,col=3)
p<-predict(LES,n.ahead=18)
lines(p,col=2)
```

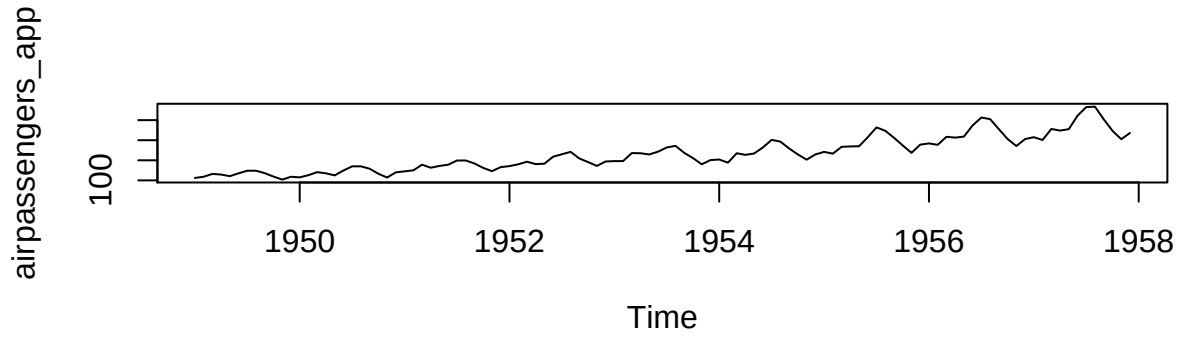


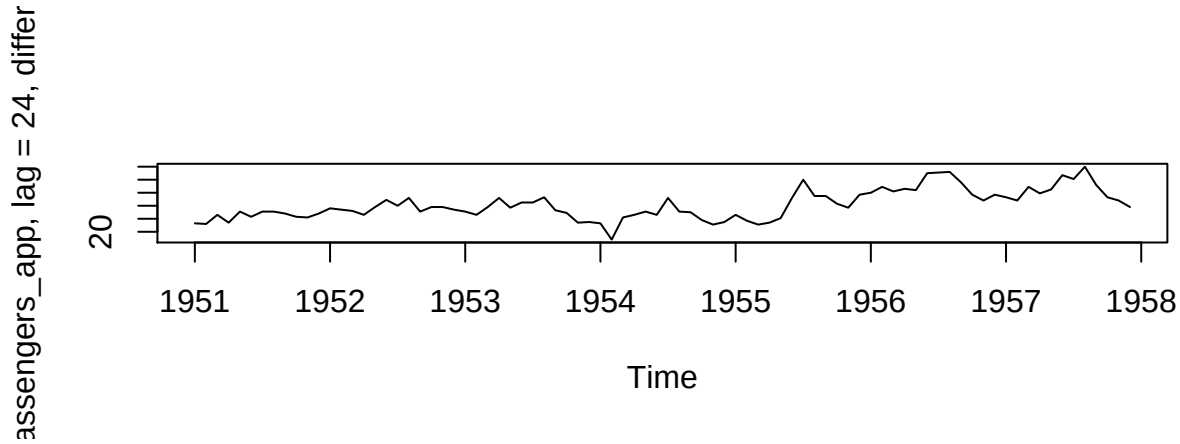

```
caret::RMSE(airpassengers_test,p)
```

```
## [1] 98.67568
```

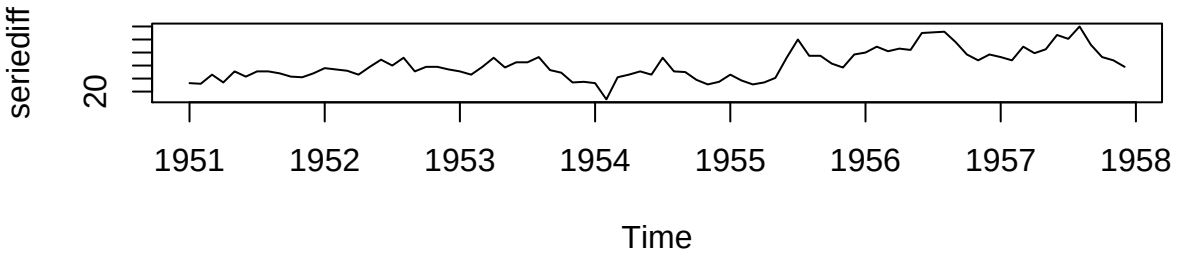
Methode de difference

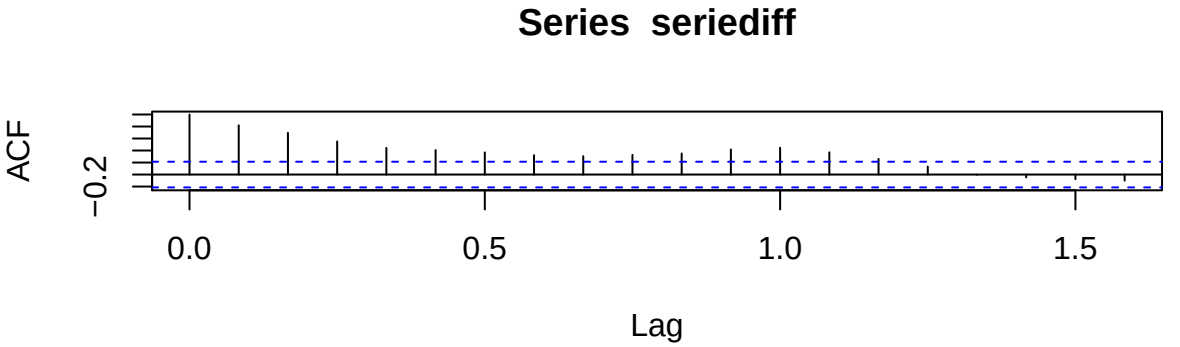
```
par(mfrow=c(2,1))
plot(airpassengers_app) # nous constatons que nous avons une periode de 24 mois, une tendance approxima
plot(diff(airpassengers_app,lag=24,differences=1)) # nous observons une saisonnalite tres legere mais p
```



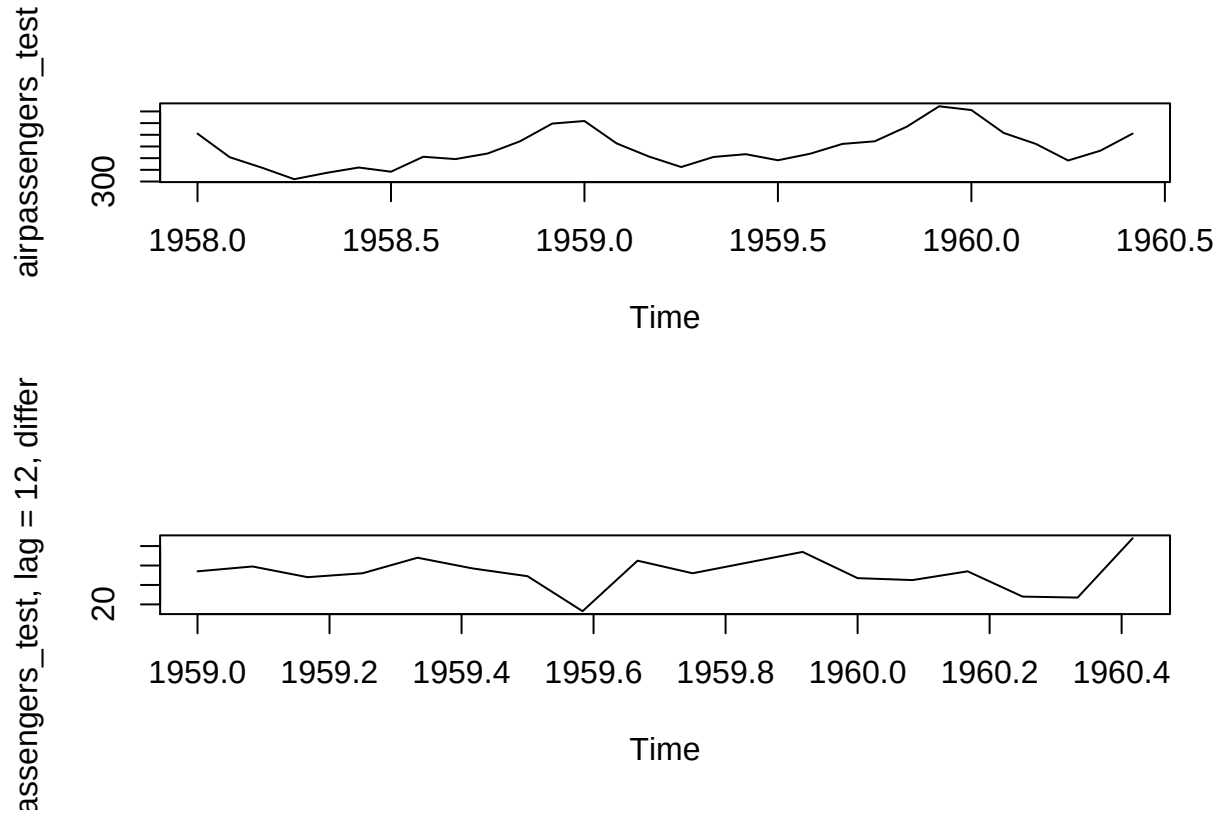



```
par(mfrow=c(2,1))
seriediff=diff(airpassengers_app,lag = 24,differences = 1)
plot(seriediff) # Cela semble à l'oeil approximativement stationnaire
acf(seriediff) # nous ne pouvons plus parler de tendance, ni de saisonnalite donc nous avons un bruit b
```

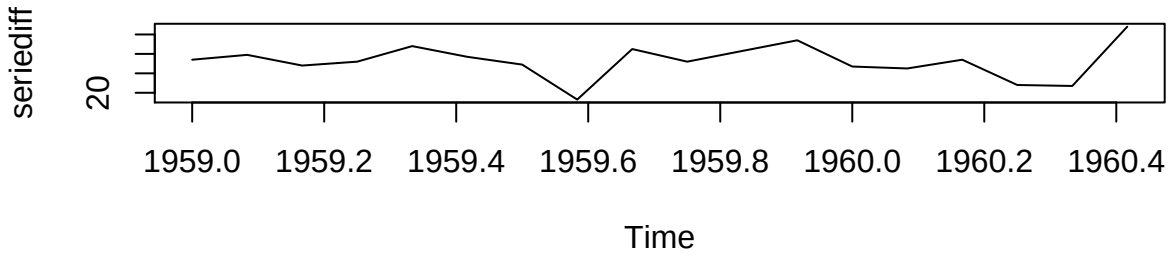


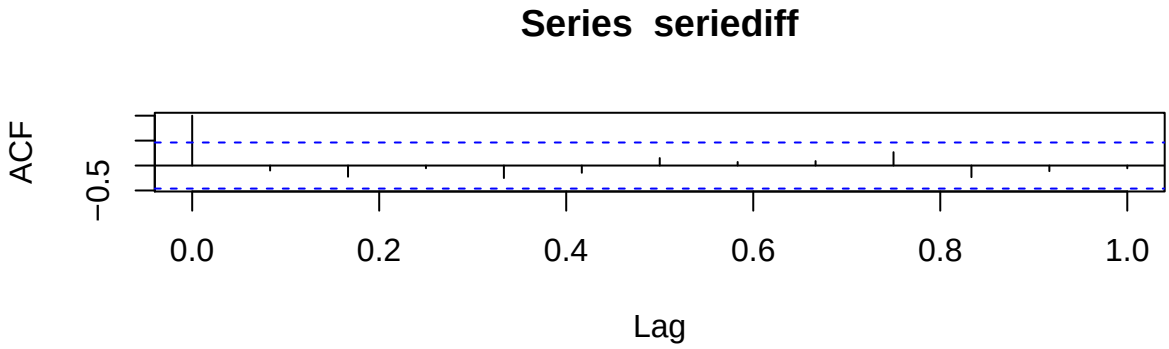


```r
par(mfrow=c(2,1))
plot(airpassengers_test) # nous constatons que nous avons une periode de 12 mois, une tendance approxim
plot(diff(airpassengers_test,lag=12,differences=1)) # Cela semble à l'oeil approximativement stationnai
```




```r
par(mfrow=c(2,1))
seriediff=diff(airpassengers_test,lag = 12,differences = 1)
plot(seriediff) # Cela semble à l'oeil approximativement stationnaire
acf(seriediff) # nous ne pouvons plus parler de tendance, ni de saisonnalite donc nous avons un bruit b
```





pour le jeu d'apprentissage, nous avons:

- **saisonnalite** = 24 mois
- **tendance** == lineaire (degre k=1)

pour le jeu d'apprentissage, nous avons:

- **saisonnalite** = 12 mois
- **tendance** == lineaire (degre k=1)