Identificación, Estimación, Diagnóstico ARIMA

CG

15/10/2020

Algunas librerias útiles

```
library(tseries)
library(astsa)
library(MASS)
#____________________________

library(forecast)
library(fBasics)
library(car)
#library(fArma)
library(urca)
library(TSA)
library(ggplot2)
```

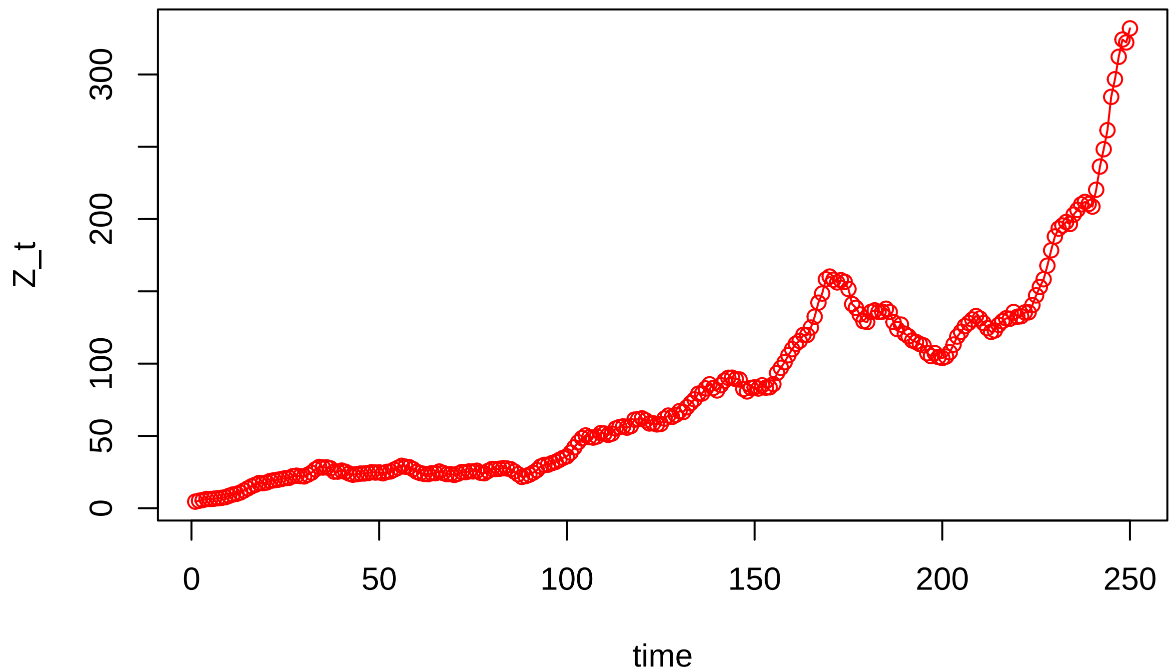
Carga de los datos

```
Y = ts(scan("ejemplo_1.txt"))
z <- Y[1:242]
head(z)
```

```
## [1] 4.576712 5.181197 5.674699 6.399020 6.381147 6.602814
```

Siempre el primer paso en el análisis de una serie de tiempo, consiste en la gráfica de los datos

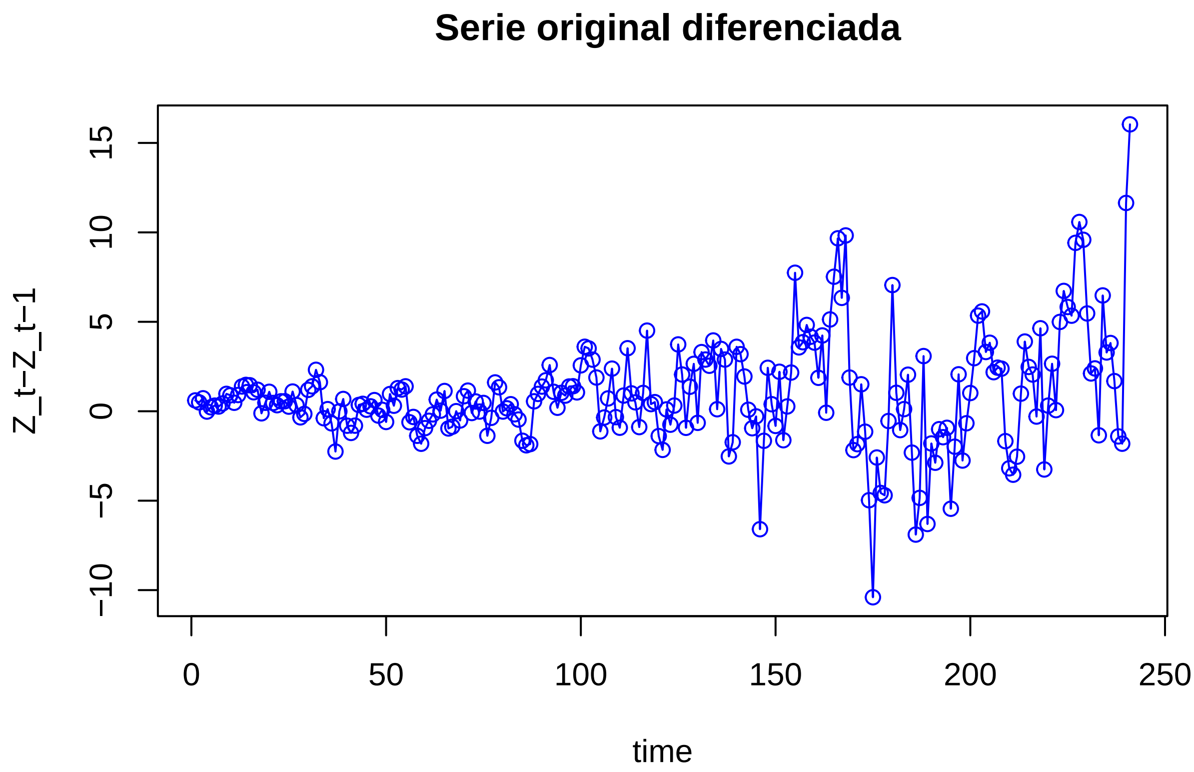
```
plot.ts(Y, ylab = "Z_t",  xlab = "time", type = "o",col= 'red', lwd = 1)
```



Las primeras observaciones son:

- No se trata de un proceso estacionario.
- Posiblemente se trate de un proceso integrado (raices unitarias).
- Posiblemente la varianza de la series no es homogenea. Pero esto no es evidente de la gráfica de la serie. Para corroborar esto mejor se diferencia la serie

```
plot.ts(diff(z),main = " Serie original diferenciada",ylab = "Z_t-Z_t-1",  xlab = "time", type = "o",col
```



Transformación Box-Cox

Como la varianza de la serie no es homogénea, se puede estimar el parámetro λ de la transformación Box-Cox utilizando la librería `car`. En este caso se trata del método incondicional, que es independiente del modelo empleado para modelar los datos

```
(tBoxCox=powerTransform(z))
```

```
## Estimated transformation parameter
##         z
## 0.2352666
```

```
summary(tBoxCox)
```

```
## bcPower Transformation to Normality
##   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## z    0.2353        0.33       0.0855        0.385
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                            LRT df      pval
## LR test, lambda = (0) 10.02208  1 0.0015468
##
## Likelihood ratio test that no transformation is needed
##                            LRT df       pval
## LR test, lambda = (1) 85.25552  1 < 2.22e-16
```

La librería `forecast` también proporciona métodos robustos para la estimación de la transformación de Box-Cox

```
BoxCox.lambda(z, method=c("loglik"), lower=-2, upper=2)
```

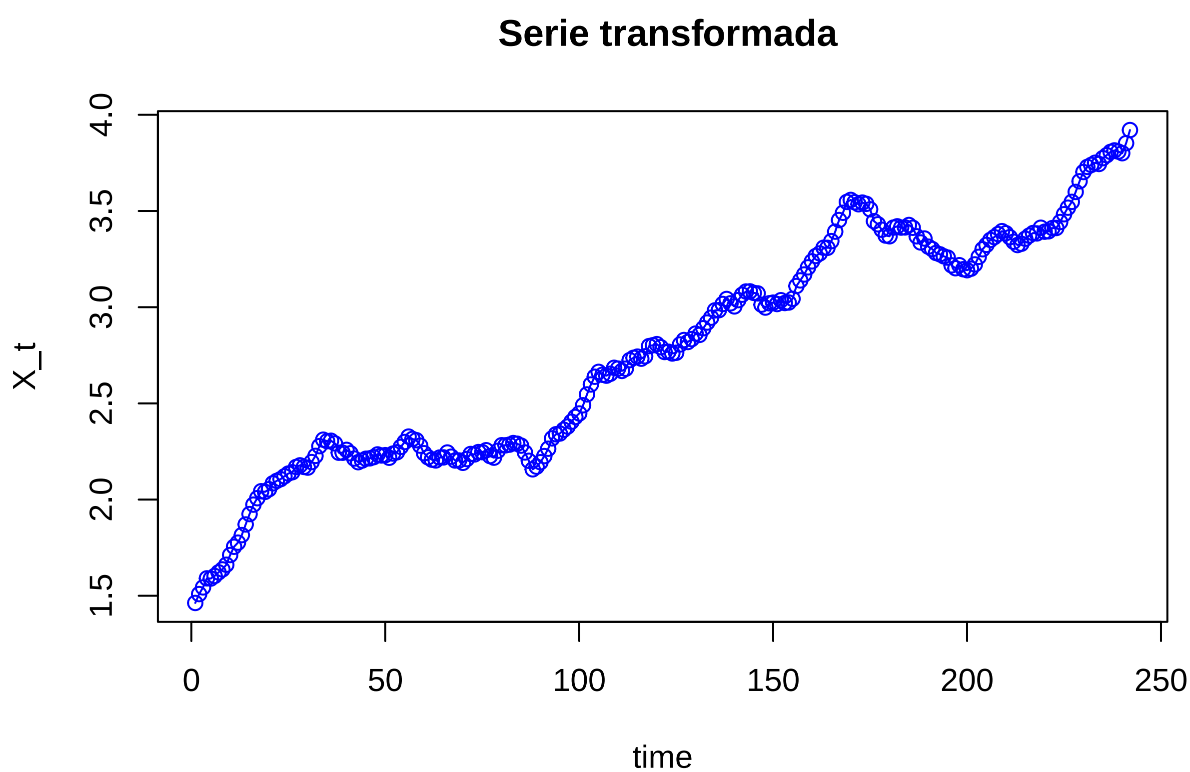
```
## [1] 0.25
```

```
BoxCox.lambda(z, method=c("guerrero"))
```

```
## [1] 0.239439
```

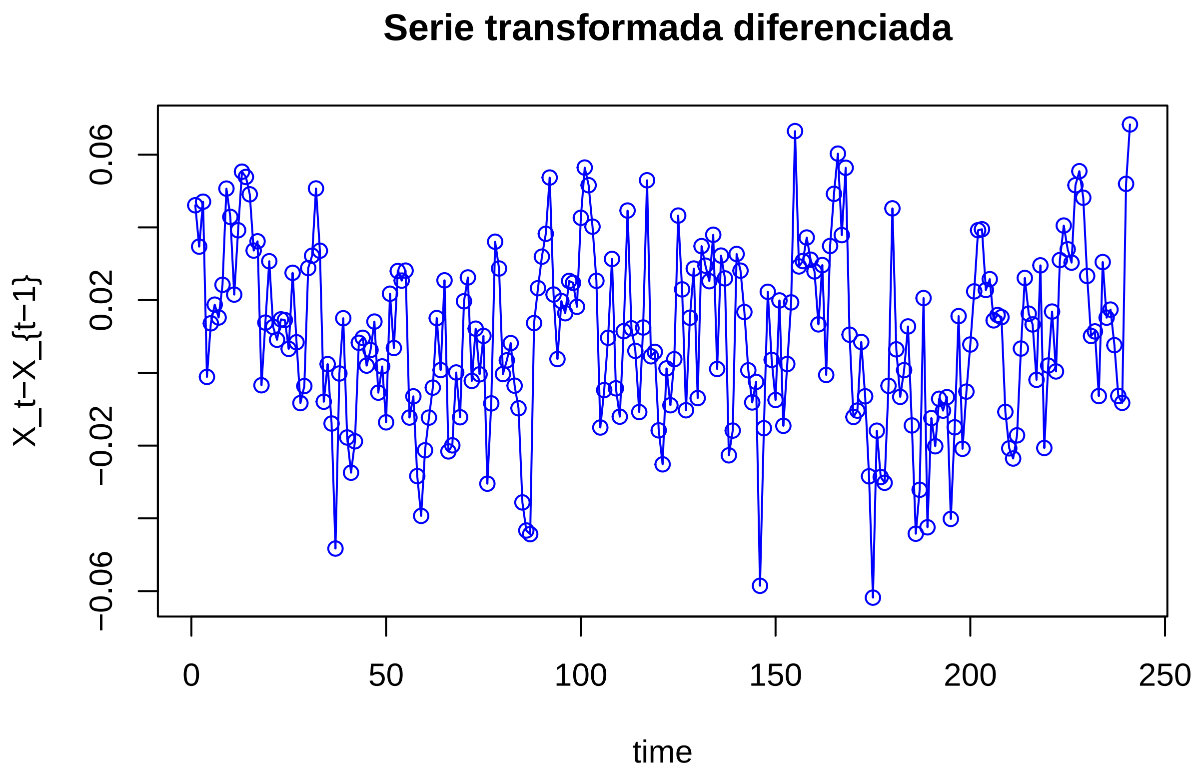
Vamos entonces a transformar la serie, utilizando la raiz cuarta $X_t = T(Z_t) = Z_t^{1/4}$

```
plot.ts(z^.25 ,main = " Serie transformada",ylab = "X_t",  xlab = "time", type = "o",col='blue', lwd = 1
```



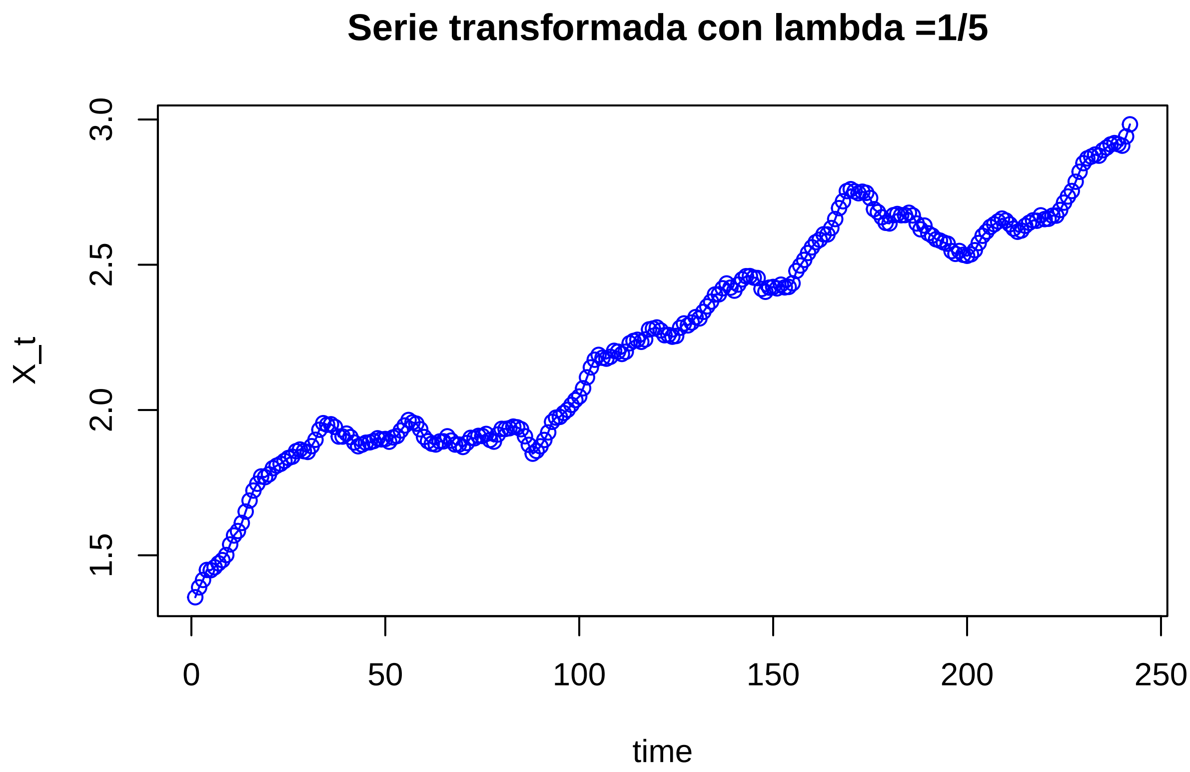
Observe en la serie diferenciada que el problema de la varianza ha mejorado

```
plot.ts(diff(z^.25) ,main = " Serie transformada diferenciada",ylab = "X_t-X_{t-1}",  xlab = "time", typ
```



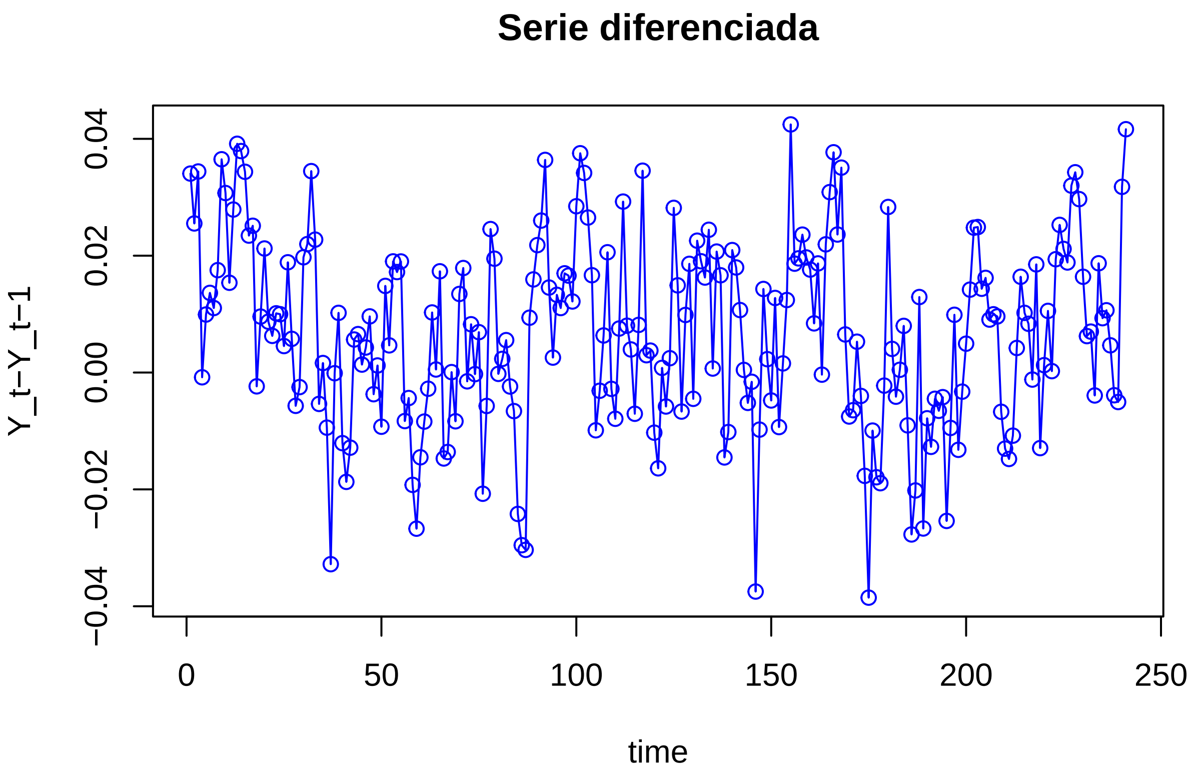
Observe los resultados que se obtendrían, si en vez de la raiz cuarta utilizaramos la raiz quinta $1/5 = 0.2$. Es deir en este caso tomaremos $Y_t = Z_t^{1/5}$

```
plot.ts(z^.2 ,main = " Serie transformada con lambda =1/5",ylab = "X_t",  xlab = "time", type = "o",col=
```



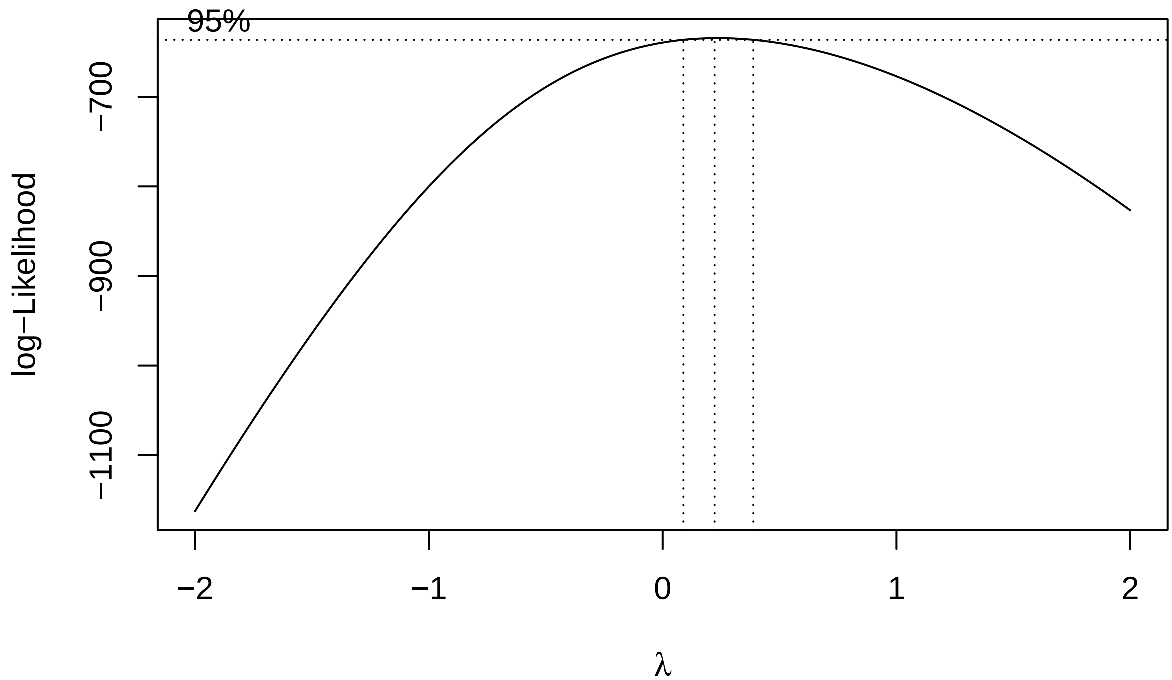
Observese que no se aprecian, mayores diferencias en relación a utilizar $\lambda = 1/4$,

```
plot.ts(diff(z^.2) ,main = " Serie diferenciada ",ylab = "Y_t-Y_t-1",  xlab = "time", type = "o",col='bl
```

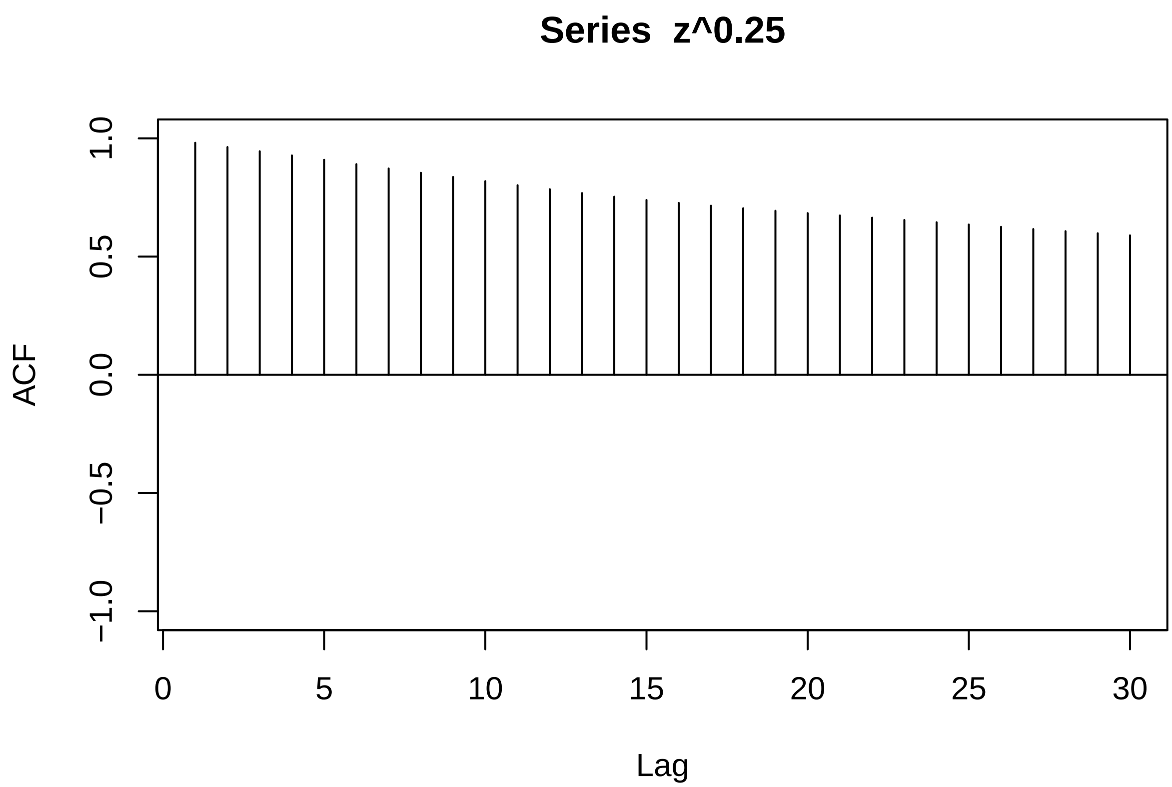


Continuaremos utlizando $\lambda = 0.25$ y cuando cada vez que nos refiramos a los datos o "la serie' ', nos estaremos refiriendo a los datos tranformados. El análisis se centrará entonces en la serie transformada $X_t = Z_t^{1/4}$. Este es el moemnto para gráficar correlogramas, en busqueda de evidencia de la necesidad de diferenciar la serie

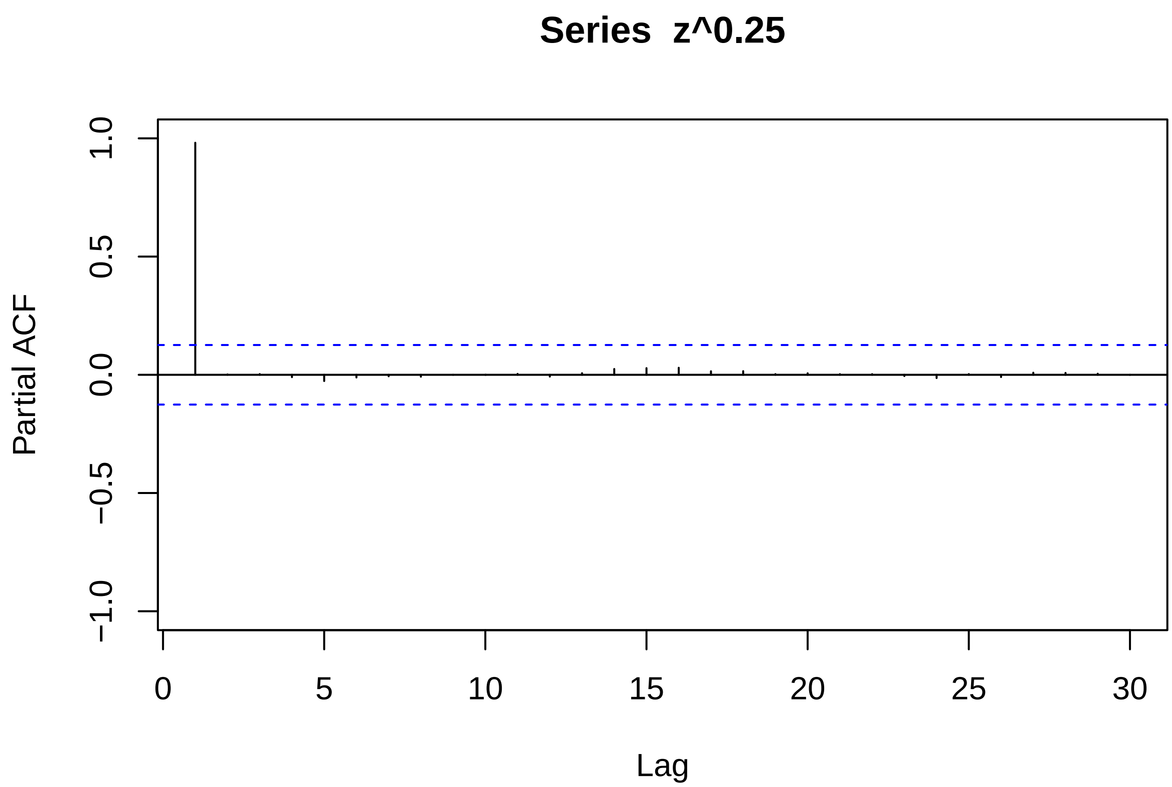
```
bc <- boxcox(lm(z~1))  # Libreria MASS
```




```
Acf(z^.25, lag.max=30, ci=0,ylim=c(-1,1))
```




```
pacf(z^.25, lag.max=30, ylim=c(-1,1))
```



Recuerdese que el hecho de que la función de autocorrelación ρ_k (ACF) decaiga lentamente y se observe un pico alto en el primer rezago de la función de autocorrelación parcial (PACF) constituyen una señal de que es posible que sea necesario diferenciar la serie a fin de volverla estacionaria.

Para corroborar esto, también podemos utilizar la prueba de raices unitarias

Prueba de raices unitarias - Test de Dickey - Fuller

La siguiente función `df` para ennvocar el test de Dickey - Fuller, se toma de la librería `urca`

```
(maxlag=floor(12*(length(z)/100)^(1/4)))
```

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

```
ru_tz=ur.df(z^.25, type = c("trend"), lags=maxlag, selectlags = c("BIC"))
summary(ru_tz)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
```

```
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.059692 -0.014165  0.000981  0.013517  0.052152
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.512e-02  1.703e-02   2.063   0.0403 *
## z.lag.1     -1.825e-02  9.253e-03  -1.973   0.0498 *
## tt           1.608e-04  7.636e-05   2.105   0.0364 *
## z.diff.lag   5.488e-01  5.650e-02   9.713   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02056 on 223 degrees of freedom
## Multiple R-squared:  0.302,  Adjusted R-squared:  0.2927
## F-statistic: 32.17 on 3 and 223 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -1.9728 4.1291 2.2349
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
```

```r
ru_tz=ur.df(z^.25, type = c("trend"), lags=maxlag, selectlags = c("AIC"))
summary(ru_tz)
```

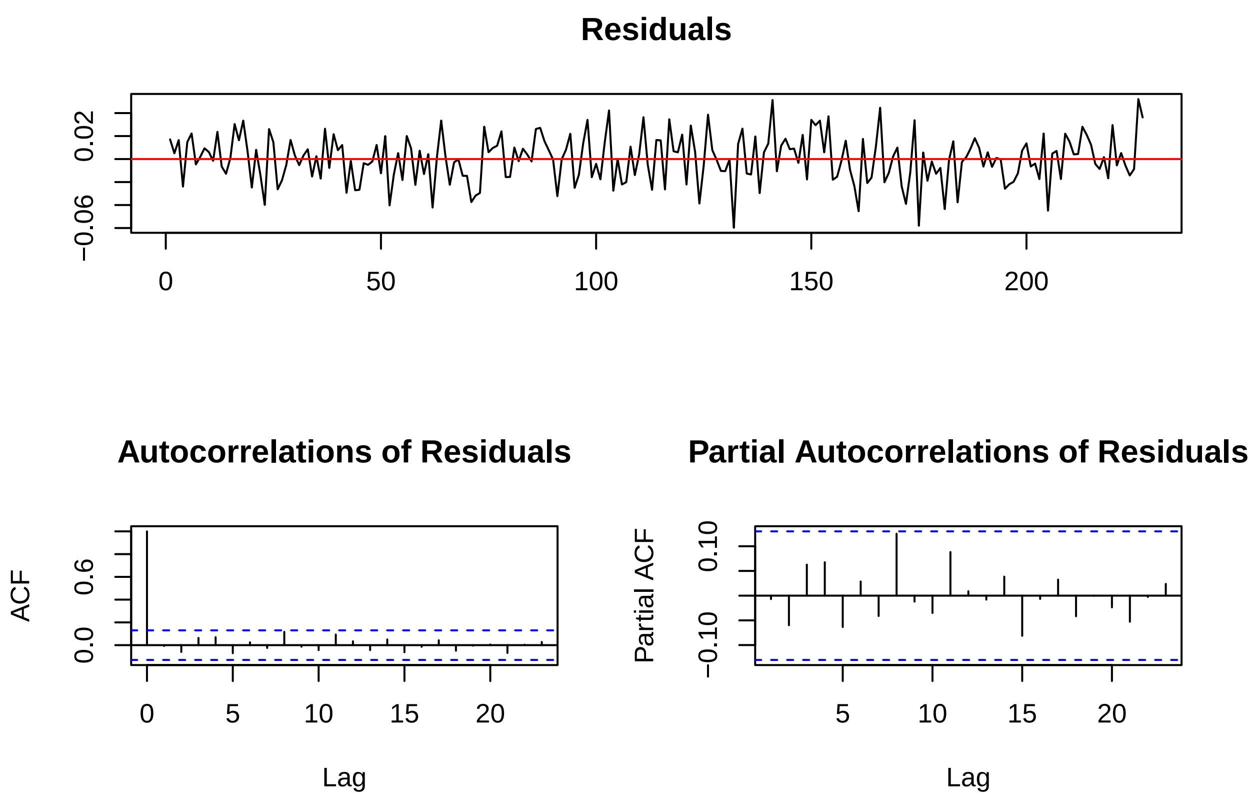
```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.059692 -0.014165  0.000981  0.013517  0.052152
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.512e-02  1.703e-02   2.063   0.0403 *
## z.lag.1     -1.825e-02  9.253e-03  -1.973   0.0498 *
## tt           1.608e-04  7.636e-05   2.105   0.0364 *
## z.diff.lag   5.488e-01  5.650e-02   9.713   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02056 on 223 degrees of freedom
## Multiple R-squared:  0.302,  Adjusted R-squared:  0.2927
```

```
## F-statistic: 32.17 on 3 and 223 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -1.9728 4.1291 2.2349
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
```

Enseguida se procede a validar la ecuación de regresión empleada en el test DF.

La función `auto.arima` realiza una busqueda automática de los ordenes (p, d, q) de un modelo para un conjunto de datos dado

```
resid=ru_tz@testreg$residuals              # residuales del modelo ADF
  plot(ru_tz)
```




```
  auto.arima(resid, max.p=5, max.q=5)
```

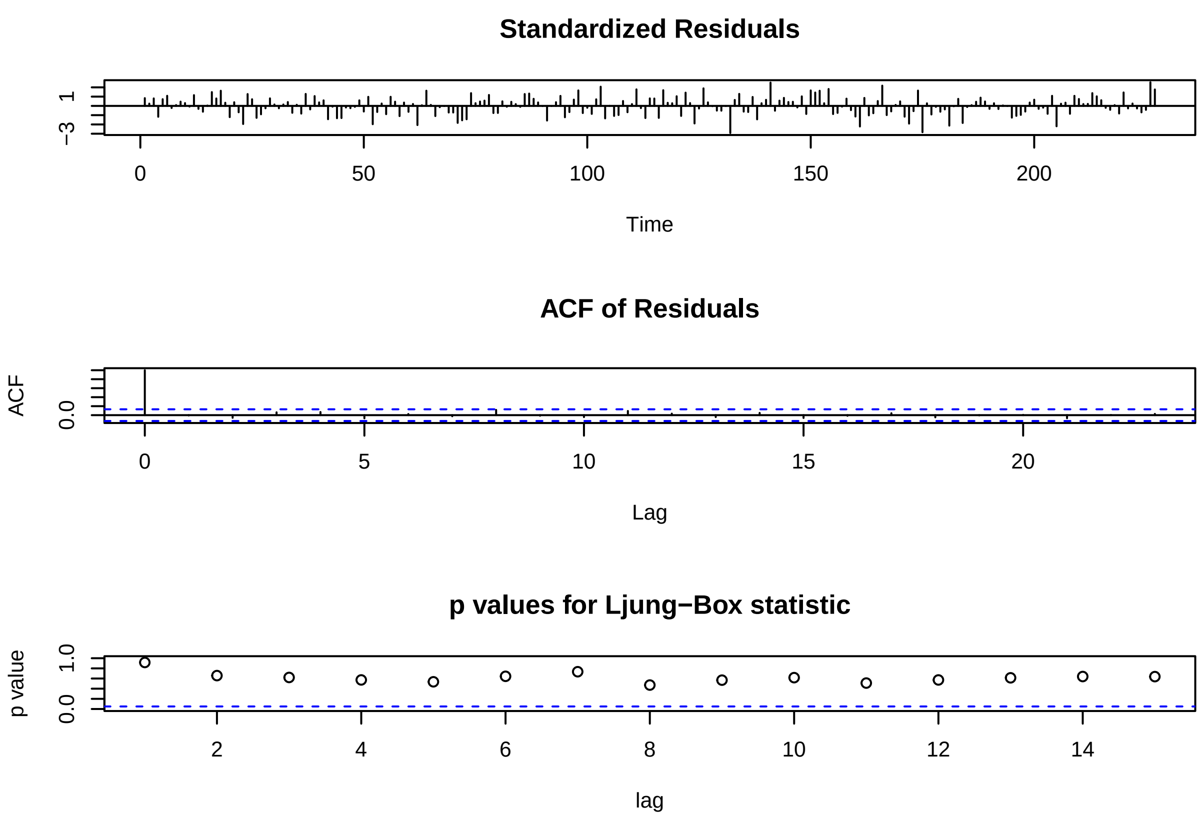
```
## Series: resid
## ARIMA(0,0,0) with zero mean
##
## sigma^2 estimated as 0.0004155:  log likelihood=561.63
## AIC=-1121.25   AICc=-1121.23   BIC=-1117.83
```

```
 # busqueda "autom?tica"
  cheq=Arima(resid, c(0,0,0), include.constant=TRUE)
```

```r
summary(cheq)
```

```
## Series: resid
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
##         mean
##       0.0000
## s.e.  0.0014
##
## sigma^2 estimated as 0.0004173:  log likelihood=561.63
## AIC=-1119.25   AICc=-1119.2   BIC=-1112.4
##
## Training set error measures:
##                          ME       RMSE        MAE MPE MAPE      MASE
## Training set -2.411395e-19 0.02038286 0.01625419 100  100 0.7094741
##                      ACF1
## Training set -0.006937537
```

```r
tsdiag(cheq, gof.lag=15)
```



Ahora se procede a indagar si hay más de una raiz unitaria

```r
ru_dif_tz=ur.df(diff(z^.25), type = c("drift"), lags=maxlag, selectlags = c("BIC"))
summary(ru_dif_tz)
```

```
##
```

```
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.061461 -0.013262  0.001426  0.013768  0.052195
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.004130   0.001488   2.775  0.00599 **
## z.lag.1     -0.474135   0.065388  -7.251 6.73e-12 ***
## z.diff.lag   0.017779   0.067289   0.264  0.79186
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02074 on 223 degrees of freedom
## Multiple R-squared:  0.2295, Adjusted R-squared:  0.2226
## F-statistic: 33.21 on 2 and 223 DF,  p-value: 2.367e-13
##
##
## Value of test-statistic is: -7.251 26.2907
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.46 -2.88 -2.57
## phi1  6.52  4.63  3.81
```

En vista de que el parámetro del t'ermino ΔY_{t-1} donde $y_t = \Delta z_t$ (recuerdese que se está investigando la existencia de más de una raiz unitaria). Se debe reespecificar el modelo con `lags` $= 0$

```
ru_dif_tz=ur.df(diff(z^.25), type = c("drift"), lags=0)
summary(ru_dif_tz)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.061658 -0.013579  0.001501  0.013519  0.052043
##
```

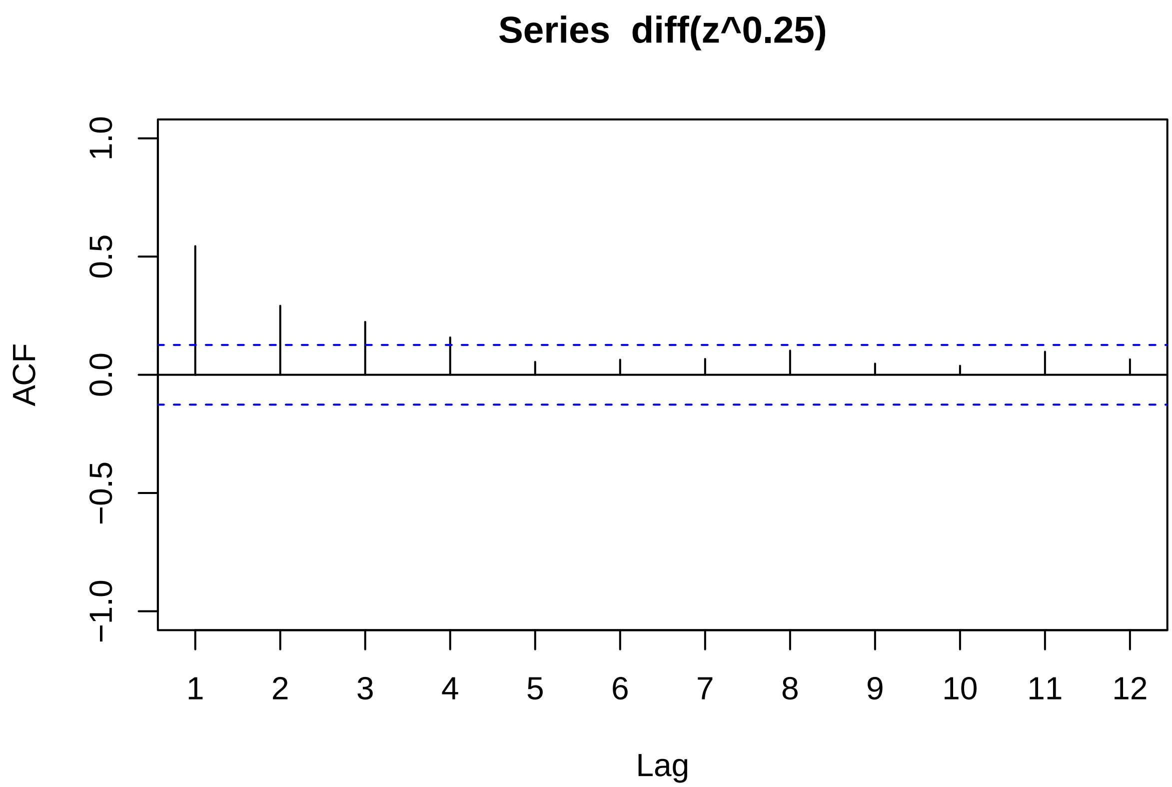
```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.004504   0.001436   3.136  0.00193 **
## z.lag.1     -0.443053   0.054404  -8.144 2.16e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02061 on 238 degrees of freedom
## Multiple R-squared:  0.2179, Adjusted R-squared:  0.2146
## F-statistic: 66.32 on 1 and 238 DF,  p-value: 2.157e-14
##
##
## Value of test-statistic is: -8.1438 33.1631
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.46 -2.88 -2.57
## phi1  6.52  4.63  3.81
```

La ecuación debe validarse como se hizo anteriormente. Luego de esto puede concluirce que solo hay una raiz unitaria y que por lo tanto solo es necesario diferenciar los datos una vez

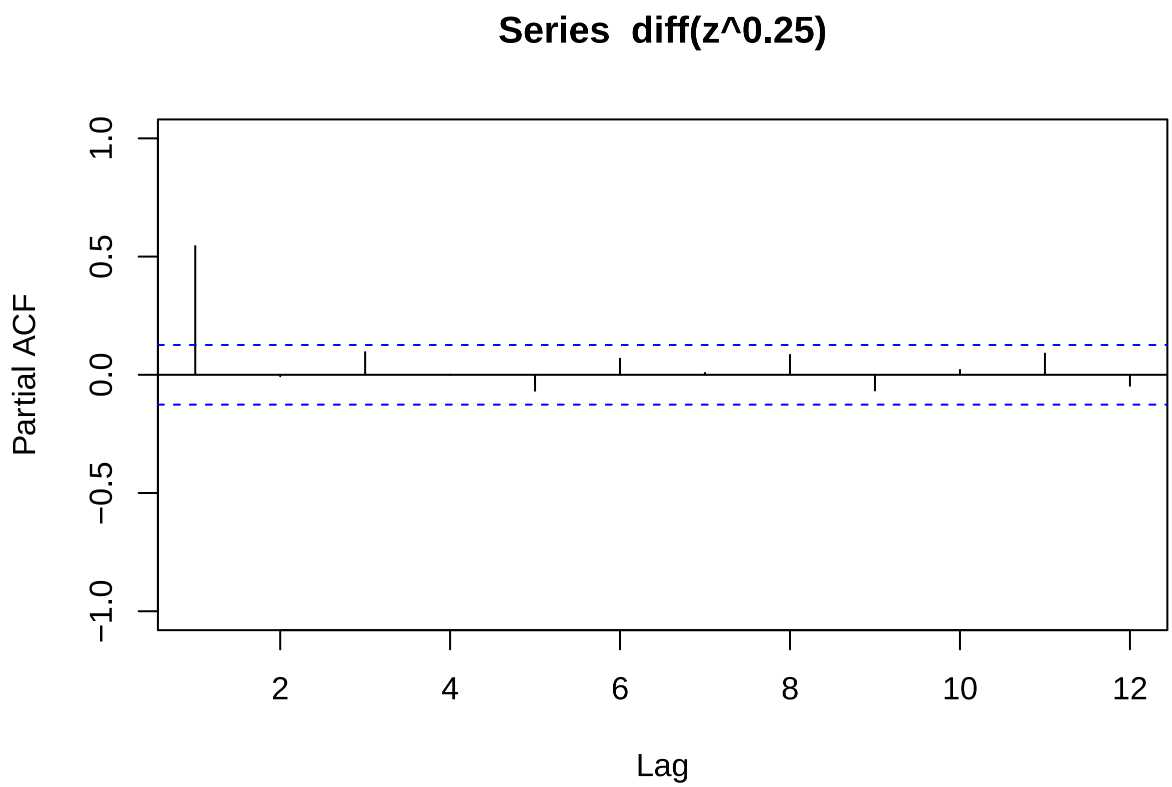
Determinación de los ordenes p y q

A continuación se ilustran los correlogramas de la serie diferenciada $\Delta z_t^{0.25}$

```
Acf(diff(z^.25), lag.max=12, ylim=c(-1,1))
```




```
pacf(diff(z^.25), lag.max=12, ylim=c(-1,1))
```




```r
eacf(z^0.25)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x x x x  x  x  x
## 1 x x x x o o o o o o o  o  o  o
## 2 o x o o x o o o o o o  o  o  o
## 3 x x o o x o o o o o o  o  o  o
## 4 x x o o o o o o o o o  o  o  o
## 5 x o x o x o o o o o o  o  o  o
## 6 x o x x x o o o o o o  o  o  o
## 7 x x x x x o o o o o o  o  o  o
```

Vamos entonces a contemplar un modelo como:

$$(1 - \phi B)(1 - B)z_t^{0.25} = \theta_0 + a_t, \tag{1}$$

procedemos a realizar una estimación de máxima verosimilitud, tras la cual podemos verificar entre otras detalles la significancia de θ_0

```r
mod1_CSS_ML=Arima(z, c(1, 1, 0), include.drift=TRUE, lambda=.25, method = c("CSS-ML"))
summary(mod1_CSS_ML)
```

```
## Series: z
## ARIMA(1,1,0) with drift
## Box Cox transformation: lambda= 0.25
##
## Coefficients:
```

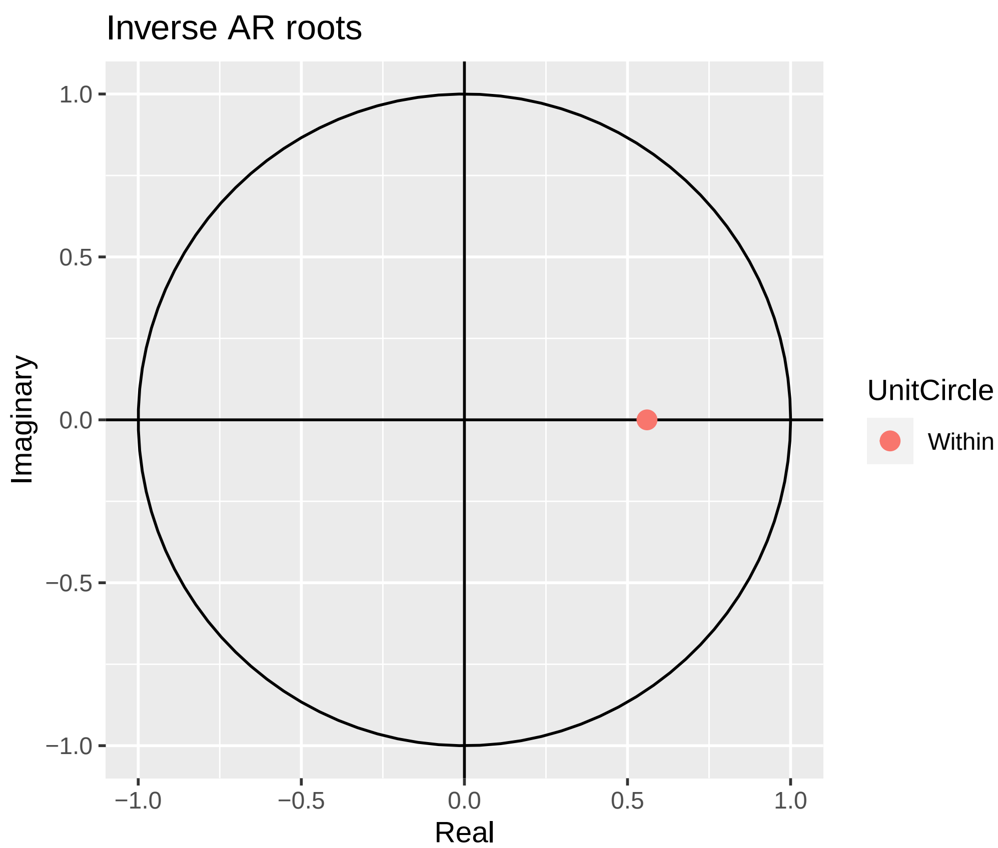
```
##          ar1   drift
##       0.5597  0.0428
## s.e.  0.0543  0.0120
##
## sigma^2 estimated as 0.006825:  log likelihood=259.81
## AIC=-513.63   AICc=-513.52   BIC=-503.17
##
## Training set error measures:
##                        ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.0252449 2.439518 1.646874 -0.0198455 2.461948 0.7704152
##                   ACF1
## Training set 0.0297623
```

```
res1_CSS_ML=residuals(mod1_CSS_ML)
```

Etapa de diagnósticos

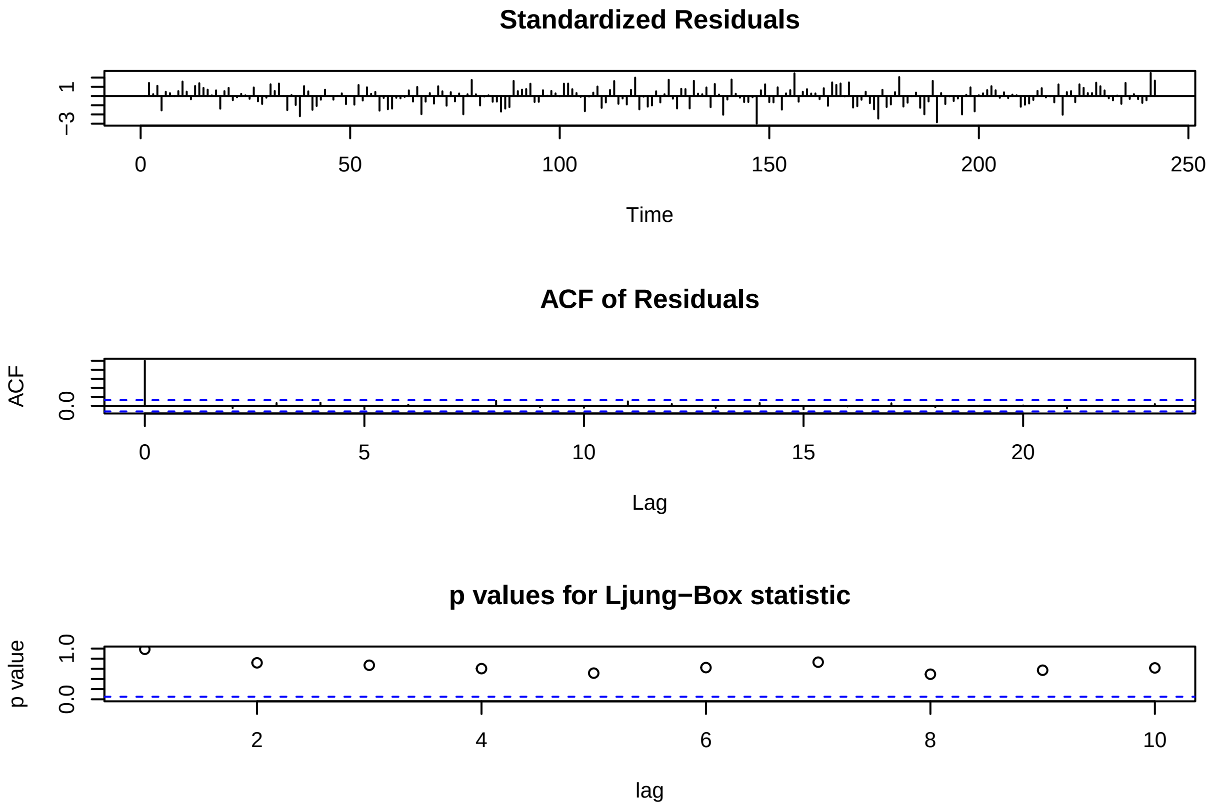
Vamos a validar la estacionaridad del polinomio autoregresivo $(1 - \phi B)$ calculando las raices por ejemplo por medio de la función `armaRoots()`

```
autoplot(mod1_CSS_ML)
```



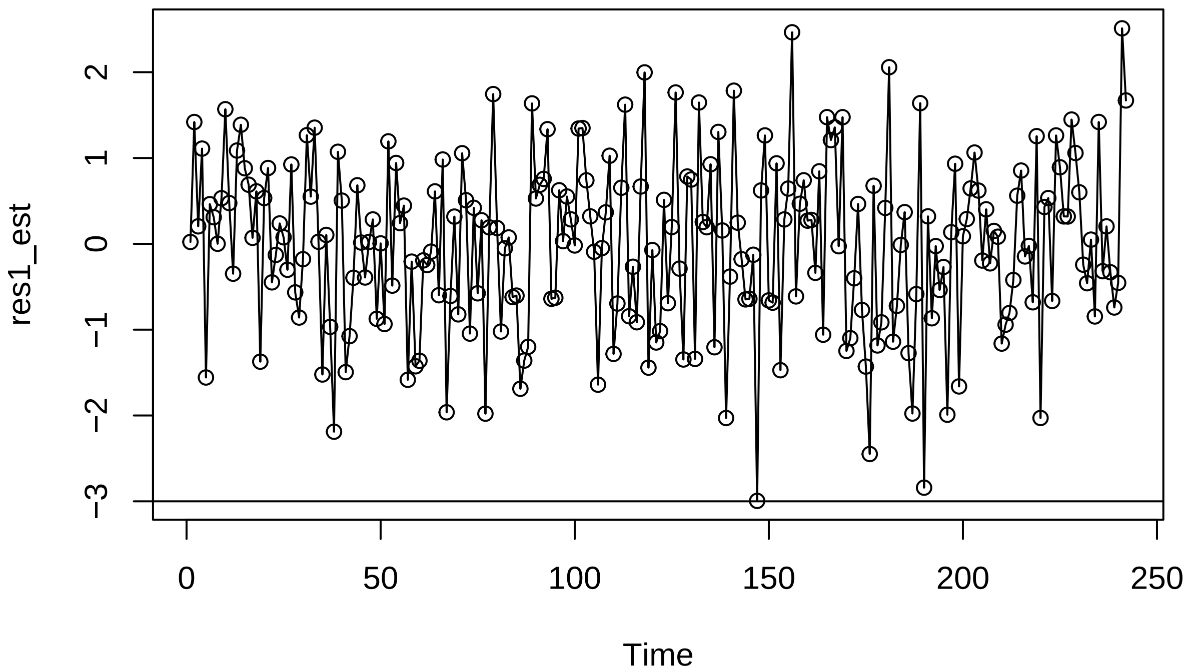
Analísis de los residuales $\hat{a}_t$ del modelo

```
tsdiag(mod1_CSS_ML)
```



Acá vamos a chequear la presencia de observaciones atípicas con un gráfico de los residuales estandarizados

```
res1_est=res1_CSS_ML/(mod1_CSS_ML$sigma2^.5)
plot.ts(res1_est, type="o")
abline(a=-3, b=0)
abline(a=3, b=0)
```



Bajo la hipótesis de normalidad el número esperado A de observaciones atípicas es

$$A \approx [\text{leng}(z) \cdot 2\phi(-0.9982)] \tag{2}$$

```
(Nobs_Esp=round(length(z)*2*pnorm(-3, mean = 0, sd = 1, lower.tail = TRUE)))
```

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

Se detectan las observaciones atípicas

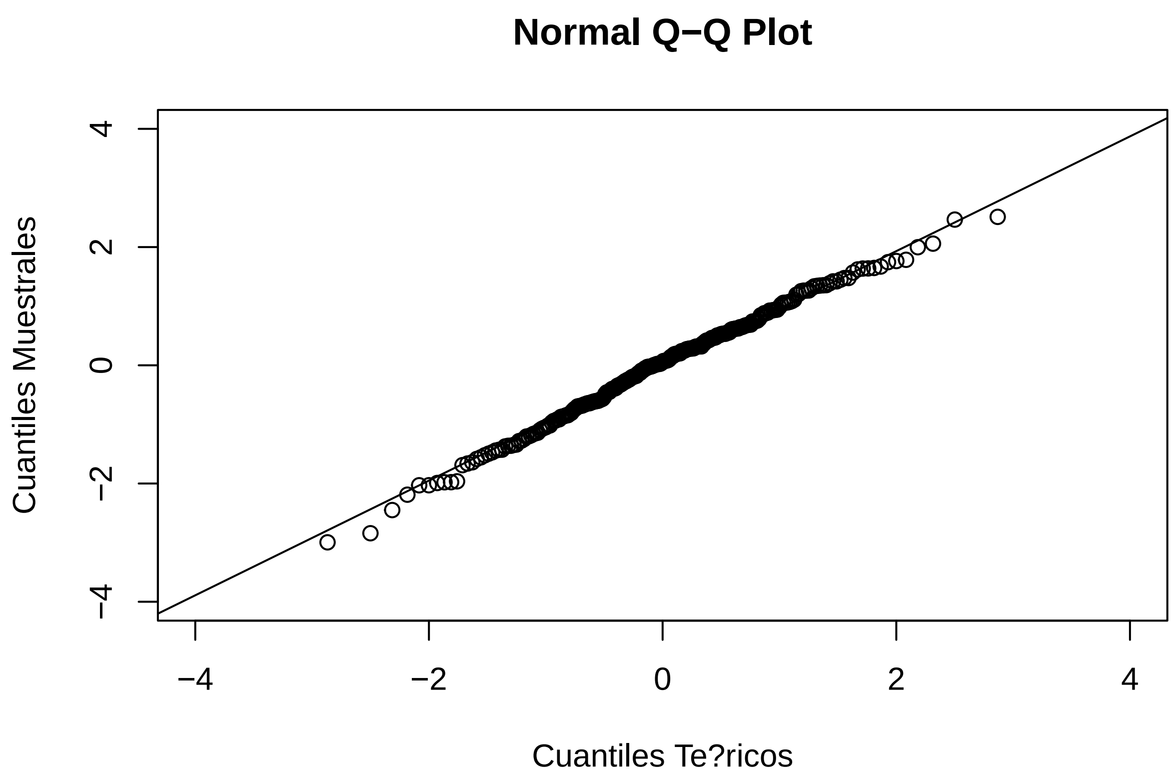
```
ind=(abs(res1_est)>3.0)
sum(ind)
```

```
## [1] 0
```

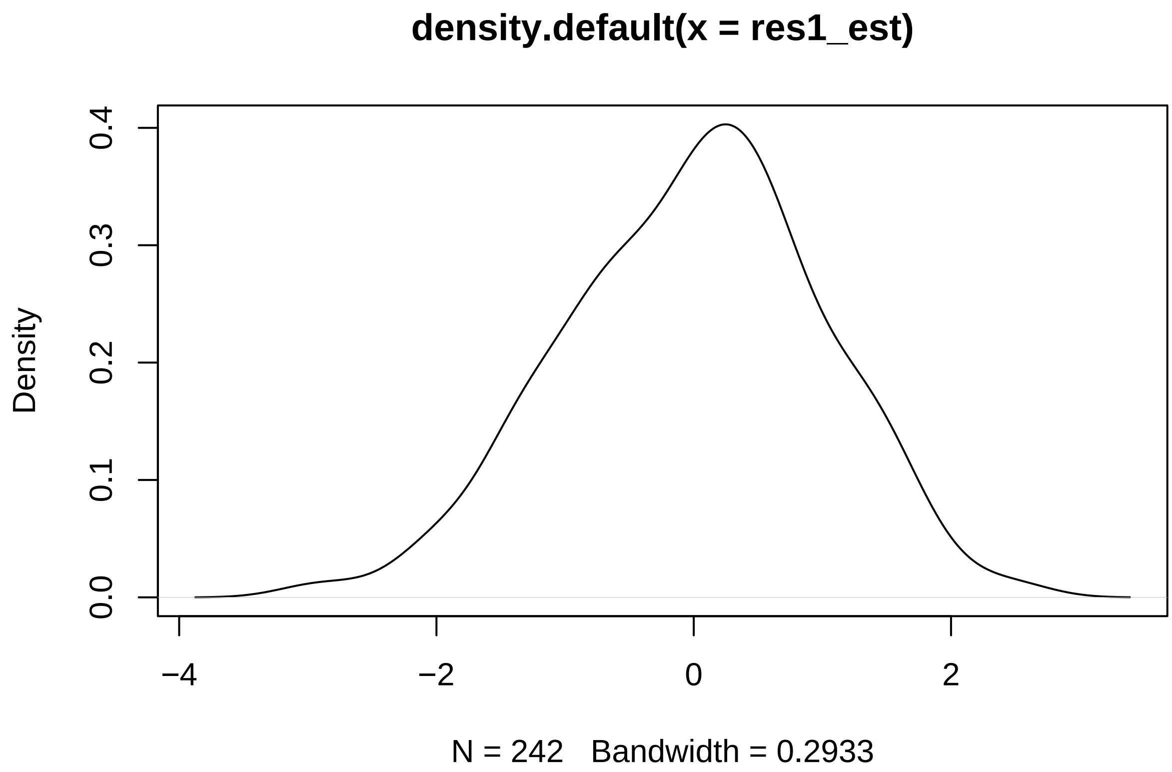
```
grupo=cbind(res1_est, ind)
```

Se verifica la normalidad de los residuales con un q-q plot

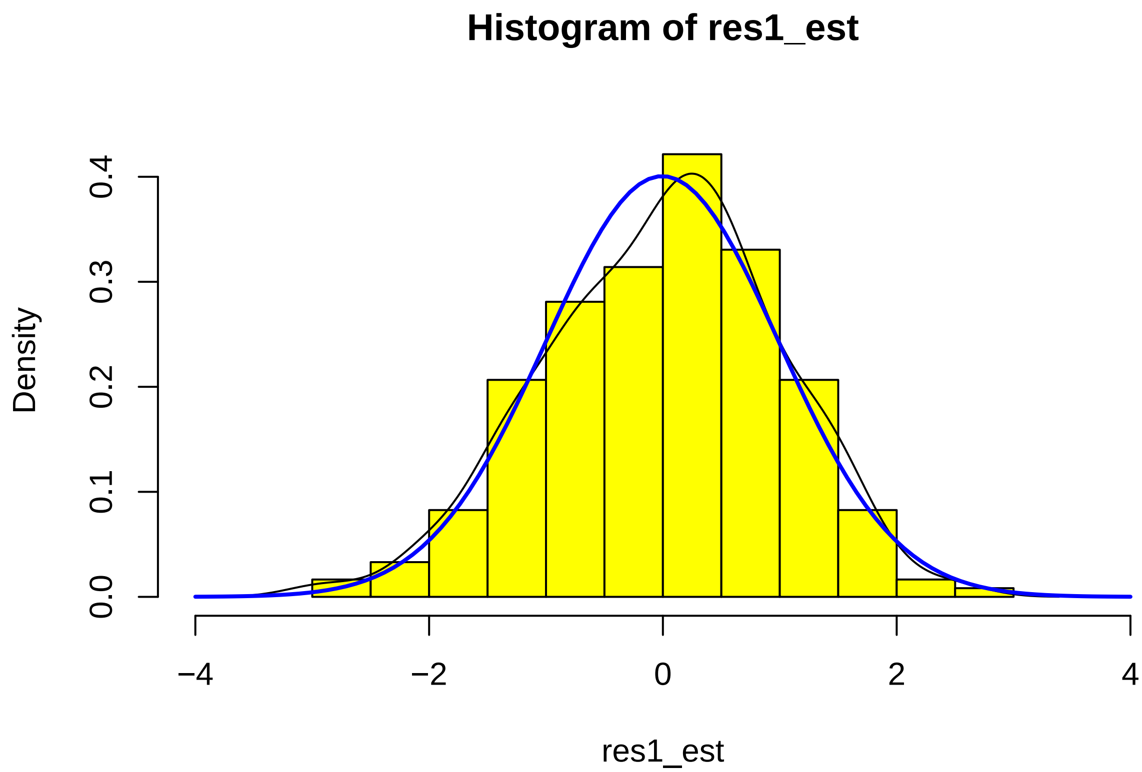
```
qqnorm(res1_est,  xlab = "Cuantiles Te?ricos", ylab = "Cuantiles Muestrales",
       xlim=c(-4,4), ylim=c(-4,4))
qqline(res1_est)
```




```
plot(density(res1_est))
```




```
mu<-mean(res1_est)
sigm<-sd(res1_est)
x<-seq(-4,4,length=100)
y<-dnorm(x,mu,sigm)
hist(res1_est,prob=T,ylim=c(0,.45),xlim=c(-4,4),col="yellow")
lines(density(res1_est))
lines(x,y,lwd=2,col="blue")
```



Por el momente se concluye que no hay un desvio fuerte de la normalidad. Se procede a realizar algunos test

Test de Normalidad

```
shapiro.test(res1_est)                # prueba de Shapiro-Wilks
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res1_est
## W = 0.99472, p-value = 0.5671
```

```
normalTest(res1_est, method=("ks"))  # En la librer?a fBasics: puede realizar otras pruebas
```

```
##
## Title:
##  One-sample Kolmogorov-Smirnov test
##
## Test Results:
##   STATISTIC:
##     D: 0.0426
##   P VALUE:
##     Alternative Two-Sided: 0.7719
##     Alternative       Less: 0.4153
##     Alternative    Greater: 0.7855
```

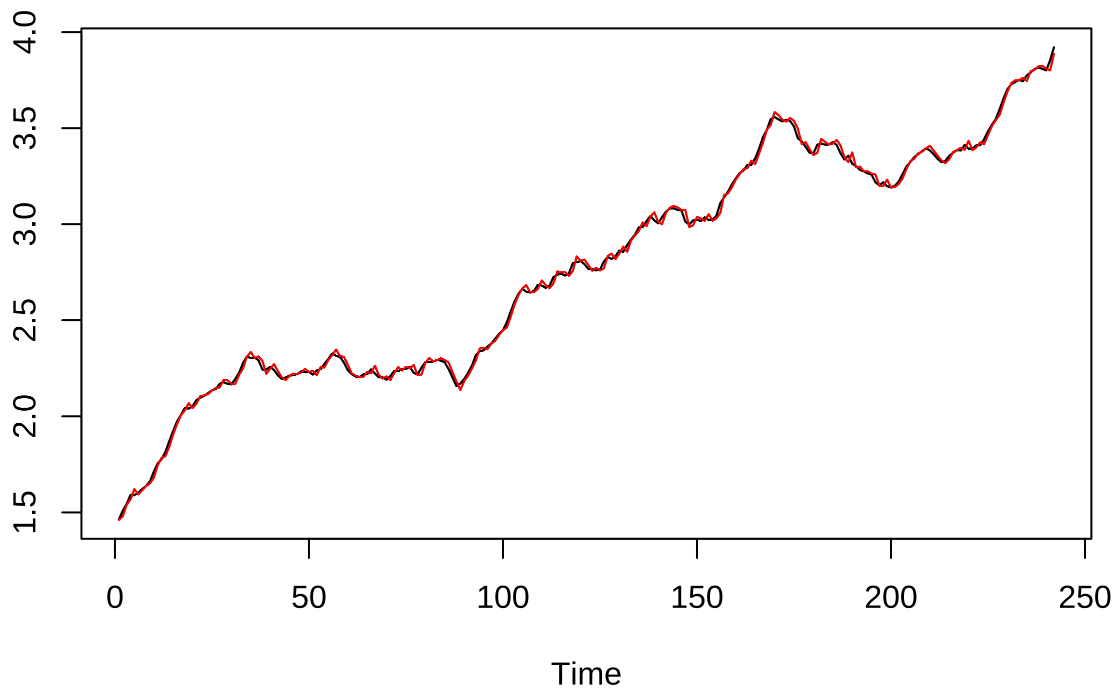
```
                                              # "ks" for Kolmogorov-Smirnov one?sample test,
                                              # "sw" for the Shapiro-Wilk test,
                                              # "jb" for the Jarque-Bera Test,
                                              # "da" for the D'Agostino Test. The default value is "ks"
```

Valores Ajustados y Observados

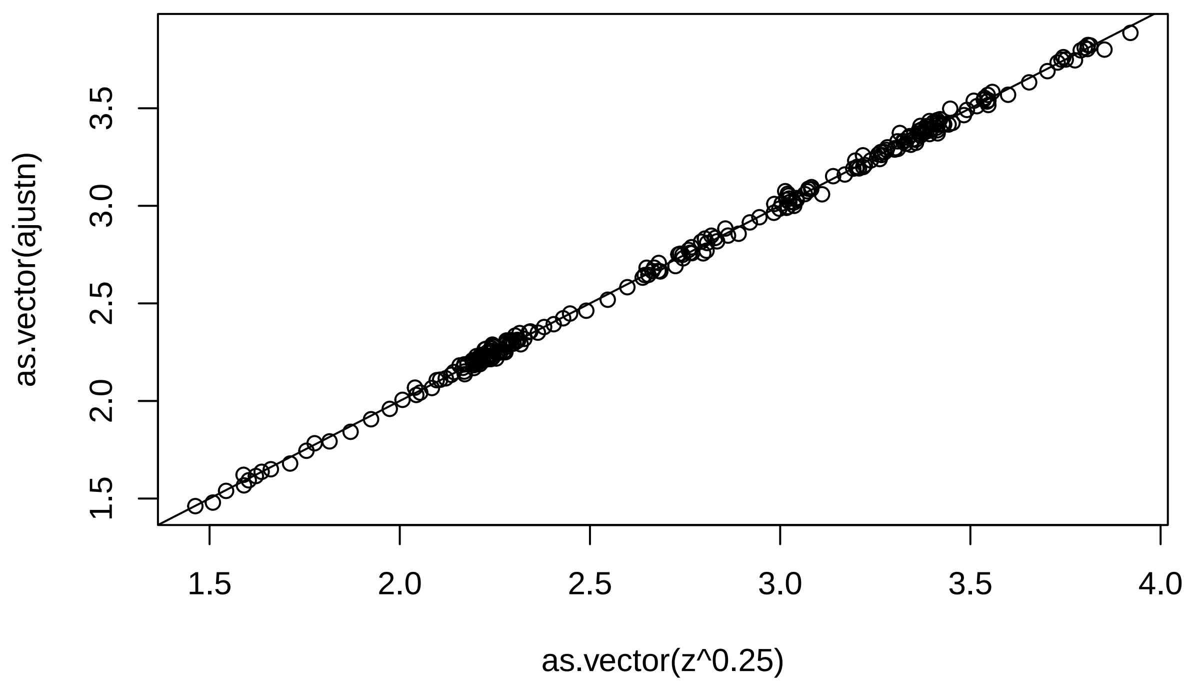
```
mod1_CSS_MLn=Arima(z^0.25, c(1, 1, 0), include.drift=TRUE, method = c("CSS-ML"))
summary(mod1_CSS_MLn)
```

```
## Series: z^0.25
## ARIMA(1,1,0) with drift
##
## Coefficients:
##          ar1   drift
##       0.5597  0.0107
## s.e.  0.0543  0.0030
##
## sigma^2 estimated as 0.0004265:  log likelihood=593.91
## AIC=-1181.82   AICc=-1181.72   BIC=-1171.37
##
## Training set error measures:
##                         ME       RMSE        MAE        MPE      MAPE      MASE
## Training set -0.0001029398 0.02052452 0.01644226 0.00434231 0.6152176 0.7604143
##                       ACF1
## Training set -0.0008894088
```

```
res1_CSS_MLn=residuals(mod1_CSS_MLn)
ajustn=z^0.25-residuals(mod1_CSS_MLn)
# gr?fico para los valores ajustados y los valores observados
ts.plot(z^0.25,ajustn)   # gr?fico de las series contra el tiempo
lines(z^0.25, col="black")
lines(ajustn, col="red")
```




```
plot(as.vector(z^.25),as.vector(ajustn), type="p")
abline(0,1)
```




```
(mod_Evalpron=Arima(z, c(1, 1, 0), include.drift=TRUE, lambda=.25, method = c("CSS-ML")))
```

```
## Series: z
## ARIMA(1,1,0) with drift
## Box Cox transformation: lambda= 0.25
##
## Coefficients:
##          ar1   drift
##       0.5597  0.0428
## s.e.  0.0543  0.0120
##
## sigma^2 estimated as 0.006825:  log likelihood=259.81
## AIC=-513.63   AICc=-513.52   BIC=-503.17
```

```
summary(mod_Evalpron)
```

```
## Series: z
## ARIMA(1,1,0) with drift
## Box Cox transformation: lambda= 0.25
##
## Coefficients:
##          ar1   drift
##       0.5597  0.0428
## s.e.  0.0543  0.0120
##
## sigma^2 estimated as 0.006825:  log likelihood=259.81
## AIC=-513.63   AICc=-513.52   BIC=-503.17
```

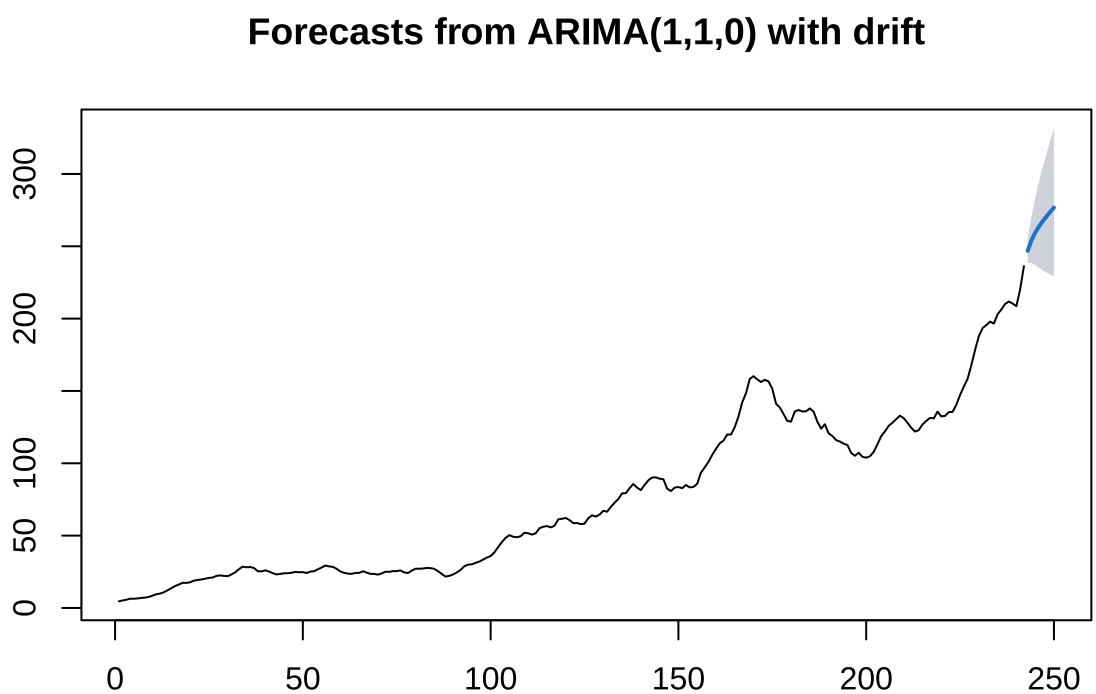
```
##
## Training set error measures:
##                        ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.0252449 2.439518 1.646874 -0.0198455 2.461948 0.7704152
##                     ACF1
## Training set 0.0297623
```

Acá se calculan los pronósticos

```
(z_pred <- forecast(mod_Evalpron, h=8, level=c(90), lambda=mod1_CSS_ML$lambda, fan=FALSE))
```

```
##     Point Forecast    Lo 90    Hi 90
## 243       246.8249 238.4714 255.3960
## 244       254.0624 238.4161 270.4665
## 245       259.3976 237.0206 283.3229
## 246       263.6505 235.2087 294.5970
## 247       267.2966 233.3958 304.7629
## 248       270.6084 231.7542 314.1546
## 249       273.7407 230.3423 323.0008
## 250       276.7817 229.1668 331.4573
```

```
v <- seq(length(Y),length(Y)-2)
plot(forecast(z_pred))
```



Evaluación de los pronósticos

```
(real=ts(Y[243:250]))
```

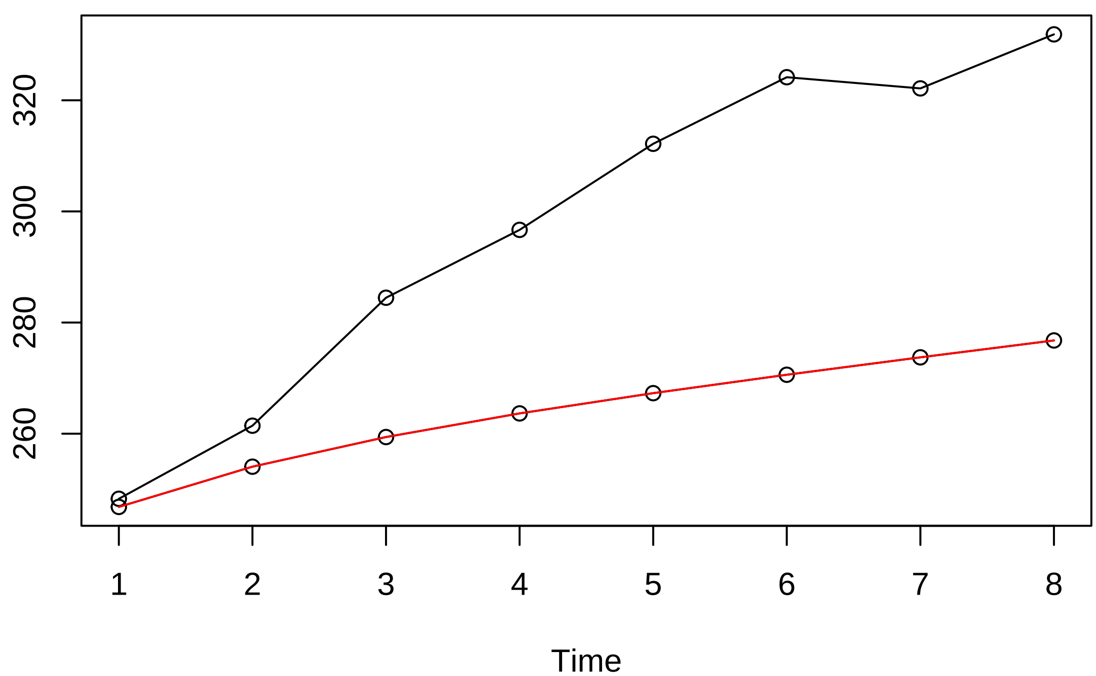
```
## Time Series:
```

```
## Start = 1
## End = 8
## Frequency = 1
## [1] 248.2866 261.4380 284.4658 296.6774 312.1698 324.1530 322.1355 331.8610
```

```r
cbind(real, ts(z_pred$mean))
```

```
## Time Series:
## Start = 1
## End = 8
## Frequency = 1
##       real ts(z_pred$mean)
## 1 248.2866        246.8249
## 2 261.4380        254.0624
## 3 284.4658        259.3976
## 4 296.6774        263.6505
## 5 312.1698        267.2966
## 6 324.1530        270.6084
## 7 322.1355        273.7407
## 8 331.8610        276.7817
```

```r
ts.plot(real, ts(z_pred$mean),type="o")
lines(ts(z_pred$mean), col="red")
```




```r
(recm=(mean((real-ts(z_pred$mean))^2))^.5)
```

```
## [1] 38.78167
```

```r
(recmp=(mean(((real-ts(z_pred$mean))/real)^2))^.5)
```

```
## [1] 0.1219904
```

```r
(eam=mean(abs(real-ts(z_pred$mean))))
```

```
## [1] 33.60304
```

```r
(eamp=mean(abs((real-ts(z_pred$mean))/real)))
```

```
## [1] 0.1073346
```