Untitled

Daniel Felipe Puentes Rocha

2024-11-09

La idea es hacer una serie de tiempo a partir de la volatilidad de las acciones de NU Holdings; para ello se utilizó una base de datos que va desde el 08/12/2021 hasta 18/10/2024, con una franja de tiempo diaria:

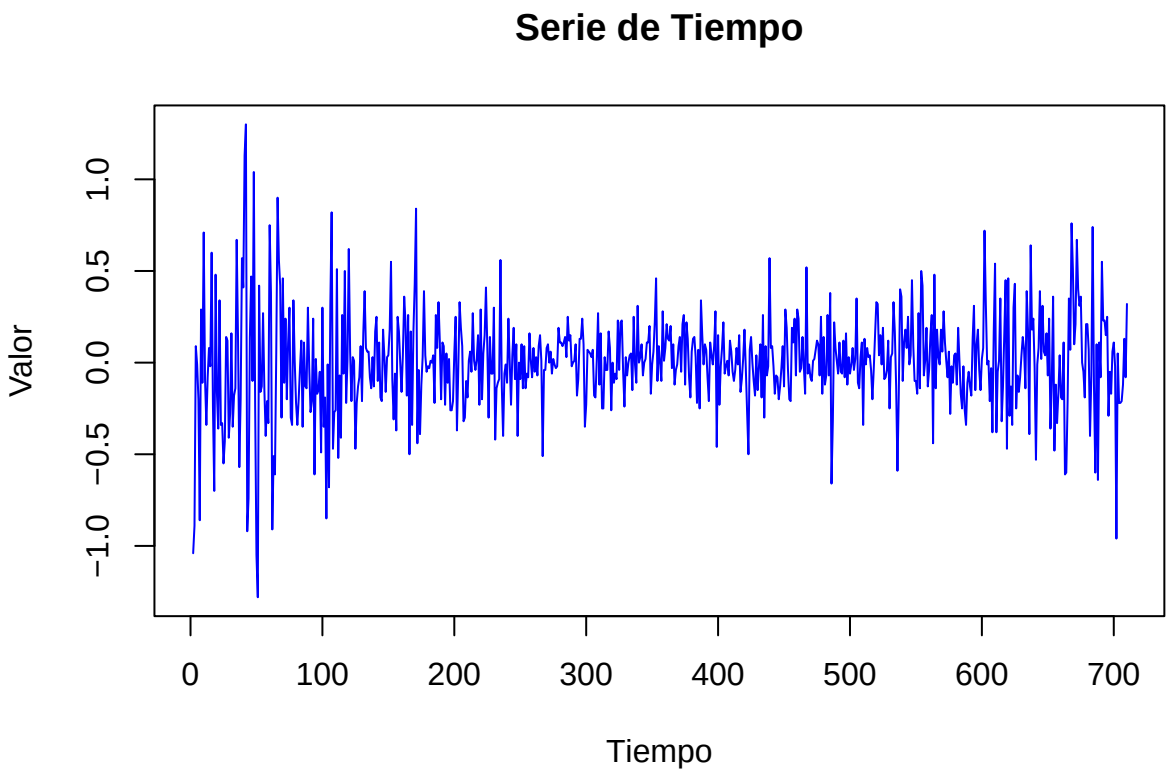
```
## Rows: 718 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Date, Vol., Change %
## dbl (4): Price, Open, High, Low
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## Cargando paquete requerido: zoo
```

```
##
## Adjuntando el paquete: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```


```
plot.ts(diff(serie), col="blue", main="Serie de Tiempo", xlab="Tiempo", ylab="Valor")
```



A pesar que la media está al rededor de cero, no hay varianza constante

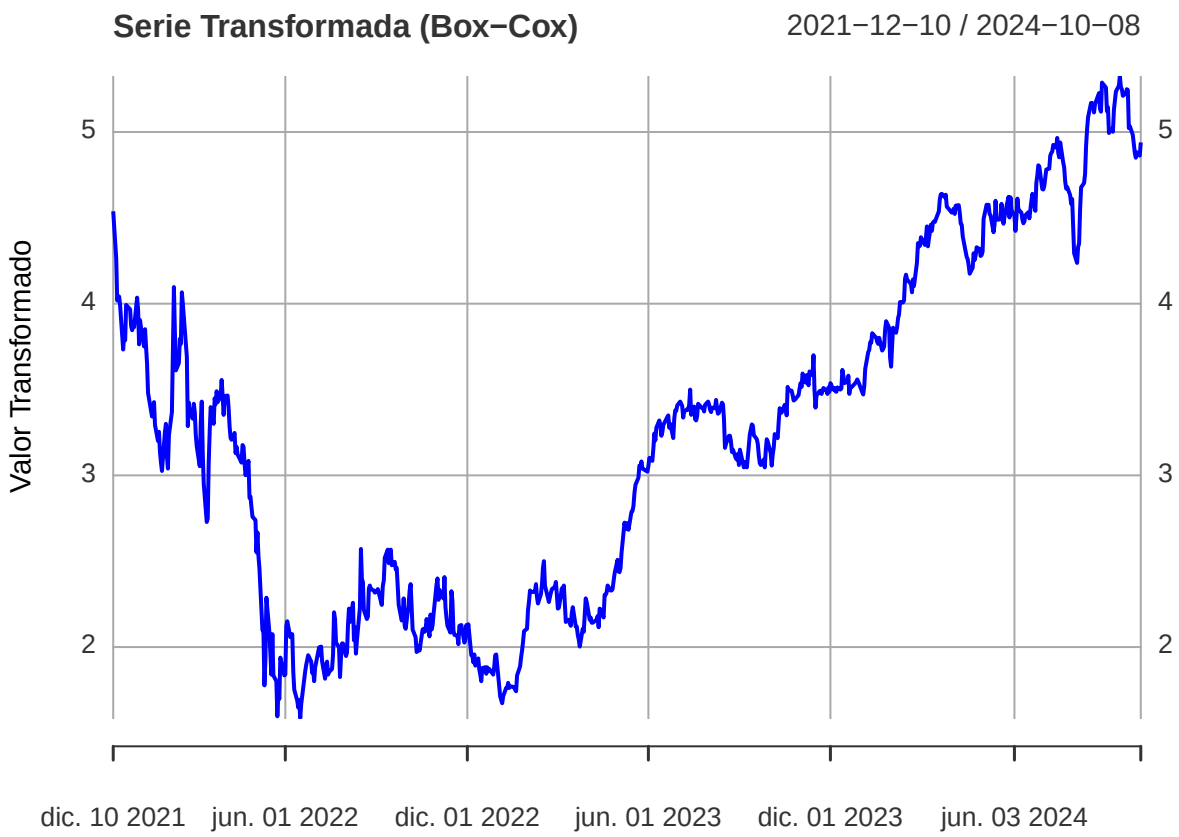
```
library(forecast)
```

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

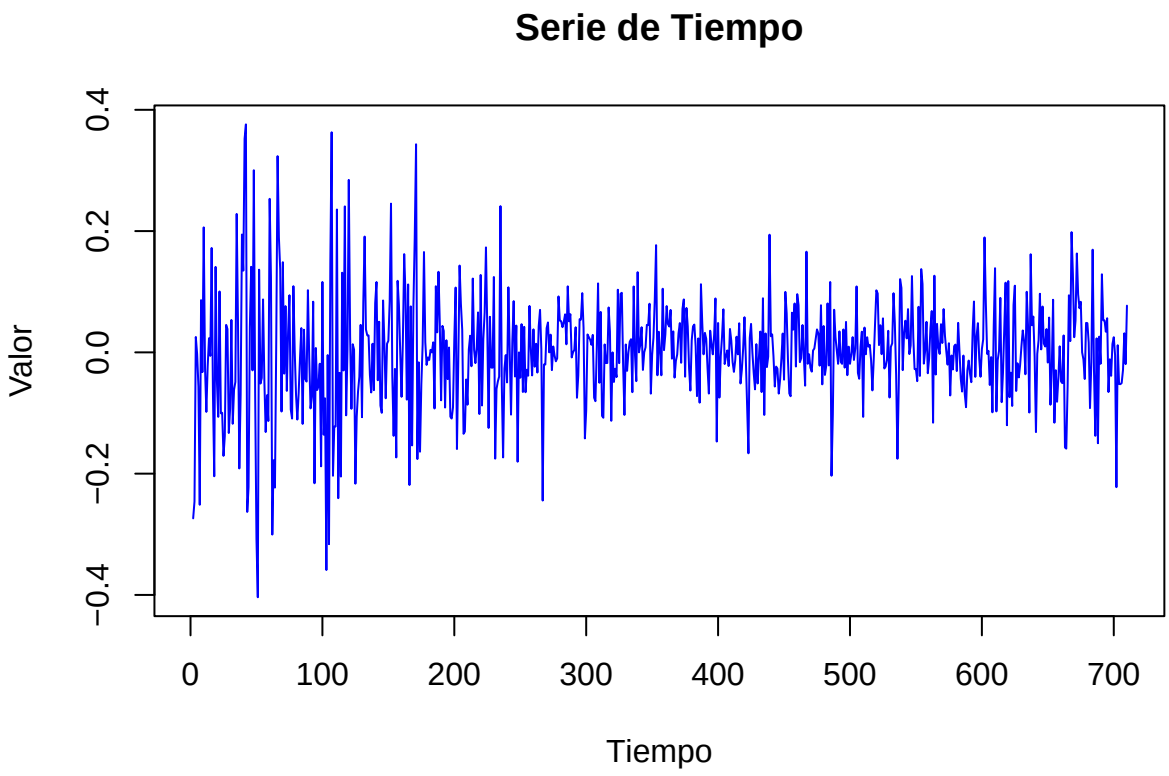
```
lambda_optimo <- round(BoxCox.lambda(serie, method=c("guerrero"), lower=-2, upper=2), 2)
lambda_optimo
```

```
## [1] 0.45
```

```
serie_transformada <- BoxCox(serie, lambda_optimo)
plot(serie_transformada, type="l", col="blue", main="Serie Transformada (Box-Cox)",
xlab="Tiempo", ylab="Valor Transformado")
```




```
plot.ts(diff(serie_transformada), col="blue", main="Serie de Tiempo", xlab="Tiempo", ylab="Valor")
```



Aunque sigue sin haber varianza constance, se estabiliza un poco su variancion

```
library(urca)
df_prueba <- ur.df(serie, type = "trend", lags = 1)
summary(df_prueba)
```

```
##
## ###############################################
## # 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
## -1.12777 -0.13760 -0.01147  0.12919  1.33190
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.431e-02  2.733e-02   0.890 0.374010
## z.lag.1     -1.475e-02  4.724e-03  -3.122 0.001871 **
## tt           2.604e-04  7.179e-05   3.627 0.000308 ***
## z.diff.lag   5.695e-02  3.704e-02   1.537 0.124659
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2719 on 704 degrees of freedom
## Multiple R-squared:  0.02325,    Adjusted R-squared:  0.01908
## F-statistic: 5.585 on 3 and 704 DF,  p-value: 0.0008664
##
##
## Value of test-statistic is: -3.1218 4.6168 6.8618
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

El valor del estadistico de prueba es de -3.1218 el cual es mayor al α del 1%, 5% y 10%, por tanto no se puede rechazar H_0 y se concluye que la serie tiene raices una raiz unitaria, por tanto no es estacionaria.

Con esto, hay que realizar nuevamente la prueba de Dickey Fuller, pero con la serie diferenciada

```
serie_diferenciada <- na.omit(diff(serie))
df_prueba_diff <- ur.df(serie_diferenciada, type = "none", lags = 1)
summary(df_prueba_diff)
```

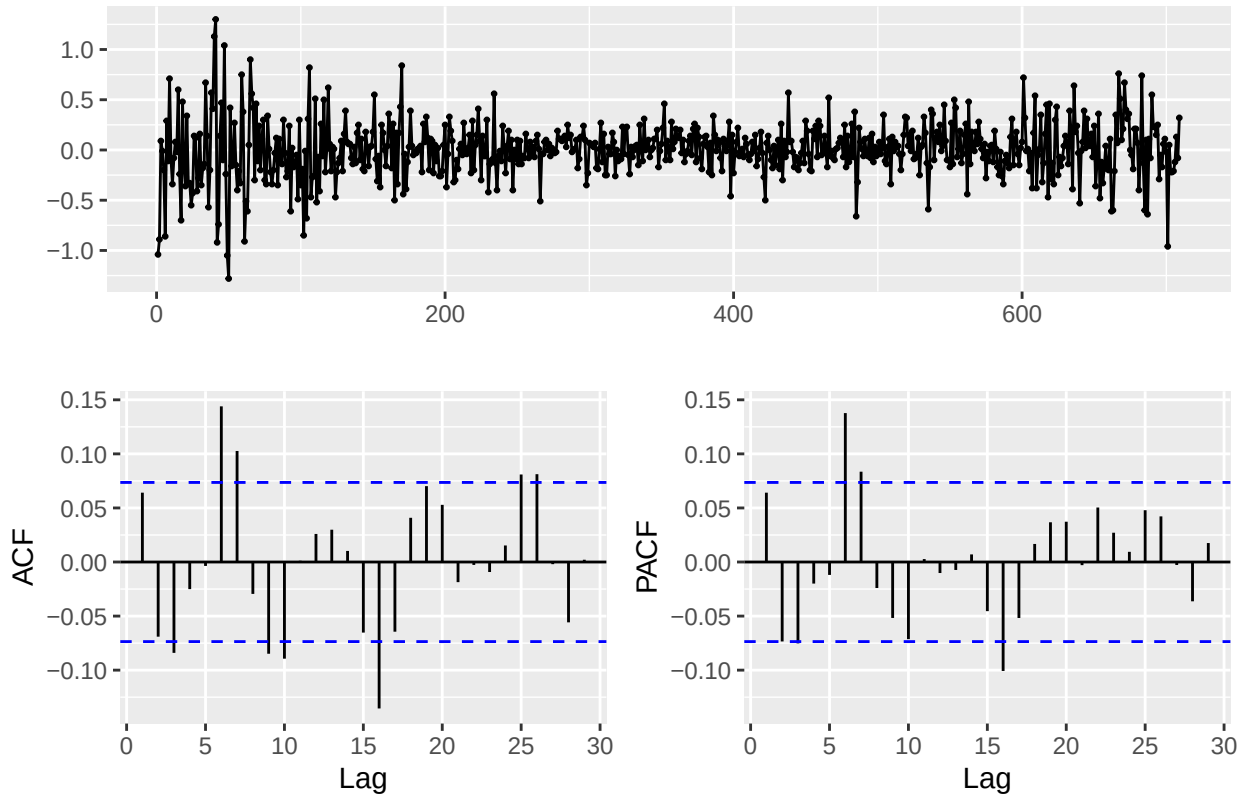
```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2417 -0.1381  0.0081  0.1359  1.2698
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -1.01943    0.05081  -20.06   <2e-16 ***
## z.diff.lag  0.07247    0.03698    1.96   0.0504 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2719 on 705 degrees of freedom
## Multiple R-squared:  0.482,  Adjusted R-squared:  0.4805
## F-statistic:   328 on 2 and 705 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -20.0629
##
## Critical values for test statistics:
##       1pct  5pct 10pct
```

```
## tau1 -2.58 -1.95 -1.62
```

Esta vez, el estadistico de prueba es mucho menor a los valores criticos en cualquier nivel de significacia, por lo que se puede rechazar H_0 y concluir que ya no presencia de raices unitarias y por tanto la serie es estacionaria

Hecho esto, se puede hacer una identificación del modelo

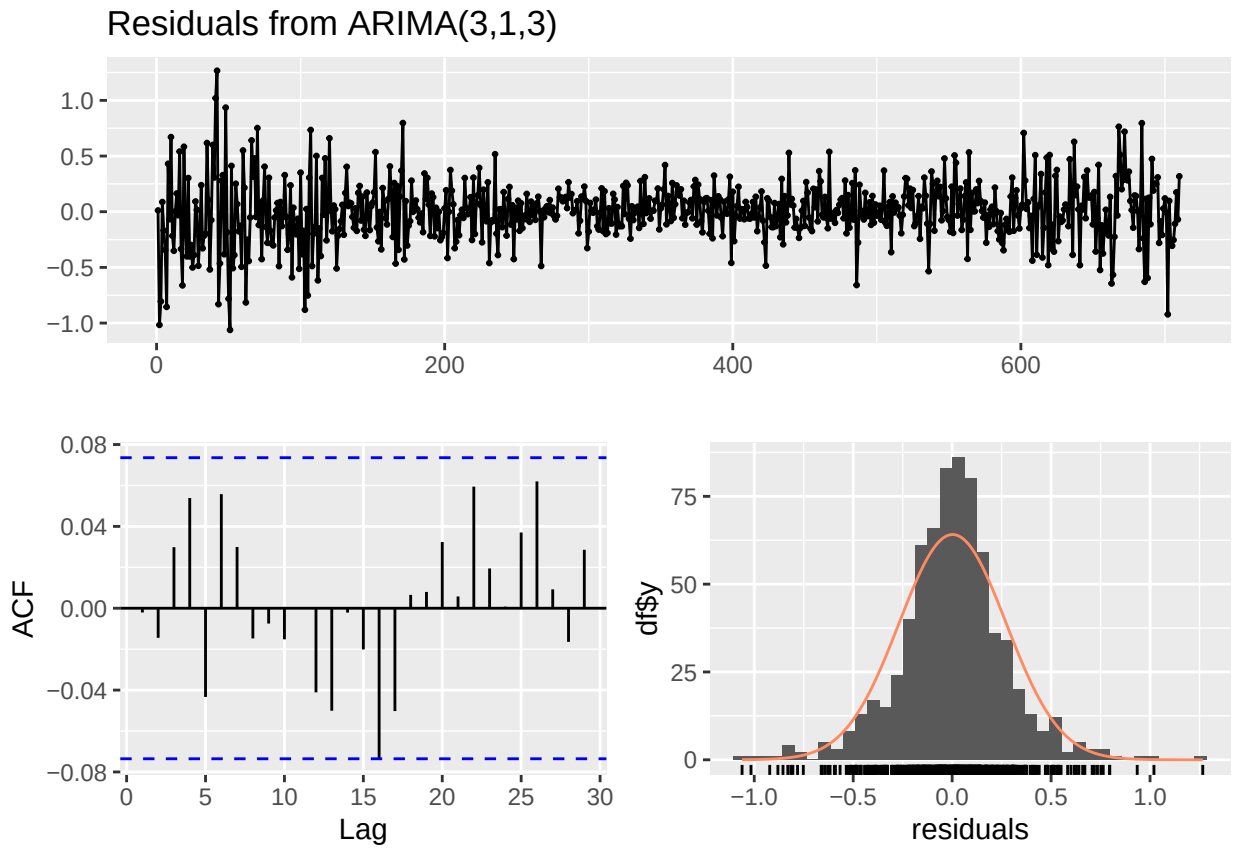
```
ggtsdisplay(serie_diferenciada)
```



Tanto el ACF como el PACF muestran picos significativos despues de los primeros 3 rezagos, lo que podria indicar que un modelo ARIMA(3,1,3) podría ser adecuado

Ahora, hay que verificar sus residuales:

```
modelo <- Arima(serie, order = c(3,1,3))

checkresiduals(modelo)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,1,3)
## Q* = 7.4565, df = 4, p-value = 0.1136
##
## Model df: 6.   Total lags used: 10
```

El test de Ljung-Box valida que hay ruido blanco en el modelo

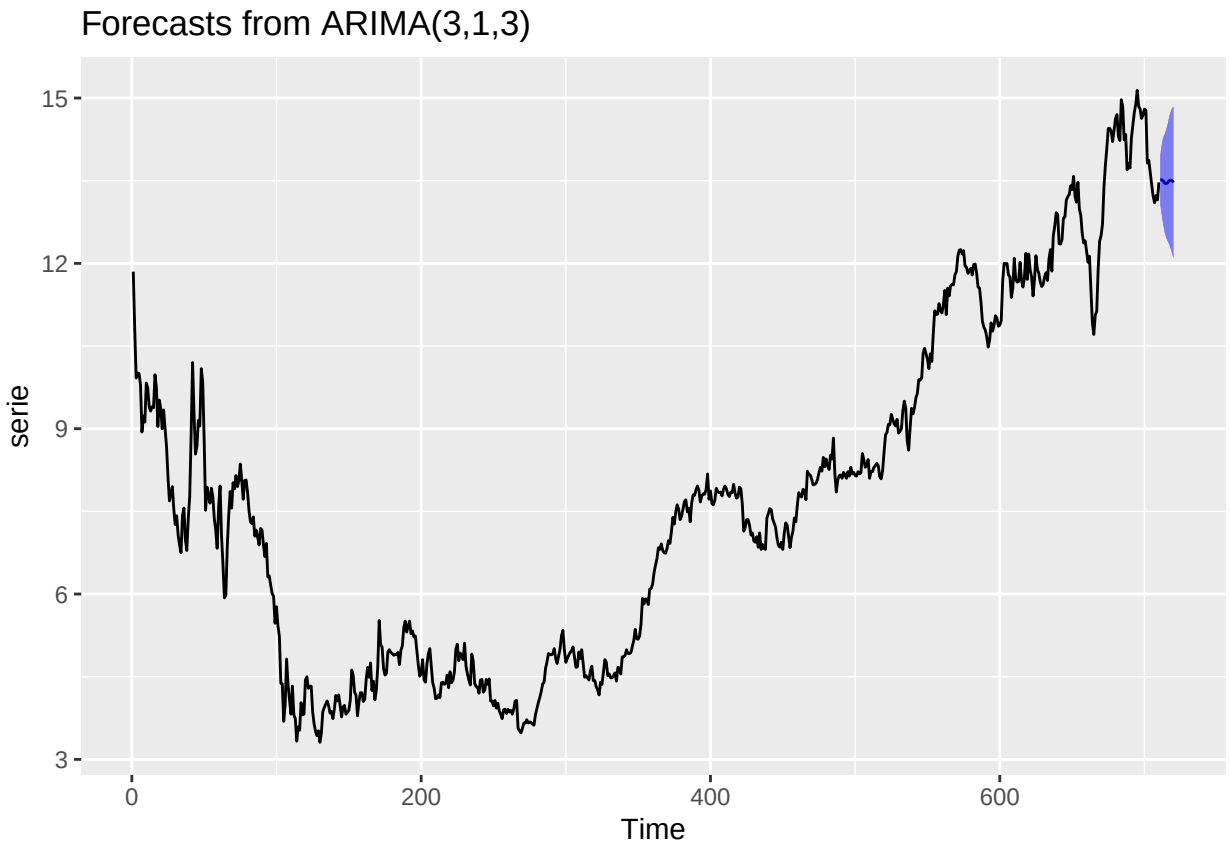
```
tseries::jarque.bera.test(modelo$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  modelo$residuals
## X-squared = 135.53, df = 2, p-value < 2.2e-16
```

Por otro lado, el test Jarque Bera muestra que los residuales no siguen una distribución normal, lo que dificulta posibles inferencias

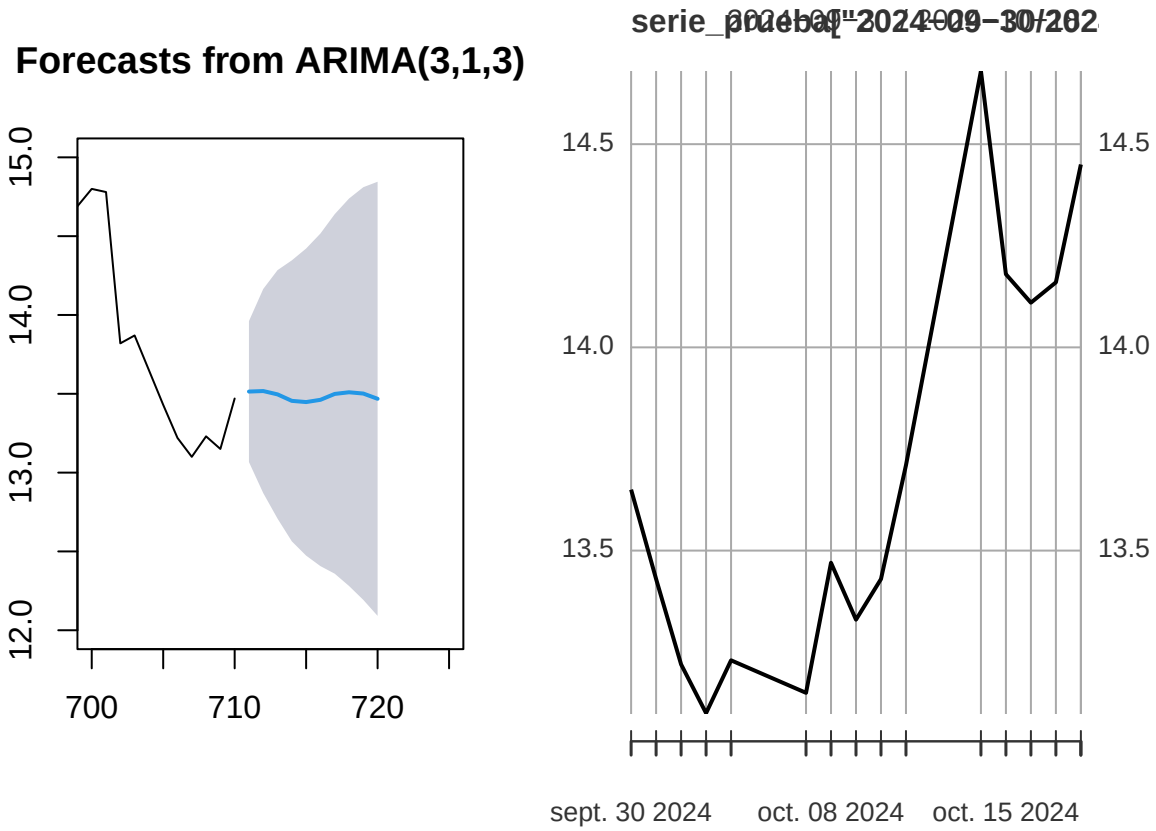
Teniendo el modelo se aplican pronósticos:

```
pronostico_modelo1 = forecast(modelo, h = 10, level = c(90), fan = FALSE)
autoplot(pronostico_modelo1) + autolayer(pronostico_modelo1)
```



Ampliando un poco mas para ver el pronostico

```
library(ggplot2)
par(mfrow=c(1,2))
plot(pronostico_modelo1, xlim=c(700,725), ylim=c(12,15))
plot(serie_prueba["2024-09-30/2024-10-18"])
```



Ahora, creando un modelo ARIMA con diferentes combinaciones de p, d y q para encontrar el mejor modelo y posteriormente comparar sus métricas:

```
# La serie está en formato xts, se convierte a ts
serie <- as.ts(serie)

# Dividiendo la serie en entrenamiento y prueba
n_entrenamiento <- floor(length(serie)*0.8)
conjunto_entrenamiento <- serie[1:n_entrenamiento]
conjunto_prueba <- serie[(n_entrenamiento + 1):length(serie)]

# Creando una función para calcular el MAPE, RMSE y MAE
calcular_error <- function(real, pronosticado){
  error <- real - pronosticado
  mape <- mean(abs(error/real))
  rmse <- sqrt(mean(error^2))
  mae <- mean(abs(error))
  return(c(mape, rmse, mae))
}

# Comparando metricas de los modelos
lista_modelos <- list(c(0,1,0), c(1,1,1), c(2,1,1), c(2,1,2), c(1,1,2),
                      c(3,1,1), c(3,1,2), c(3,1,3), c(4,1,1), c(4,1,2),
                      c(4,1,3), c(4,1,4), c(5,1,1), c(5,1,2), c(5,1,3),
                      c(5,1,4), c(5,1,5))
resultados <- data.frame(Modelo = character(), MAPE_entrenamiento = numeric(),
                         RMSE_entrenamiento = numeric(),
```

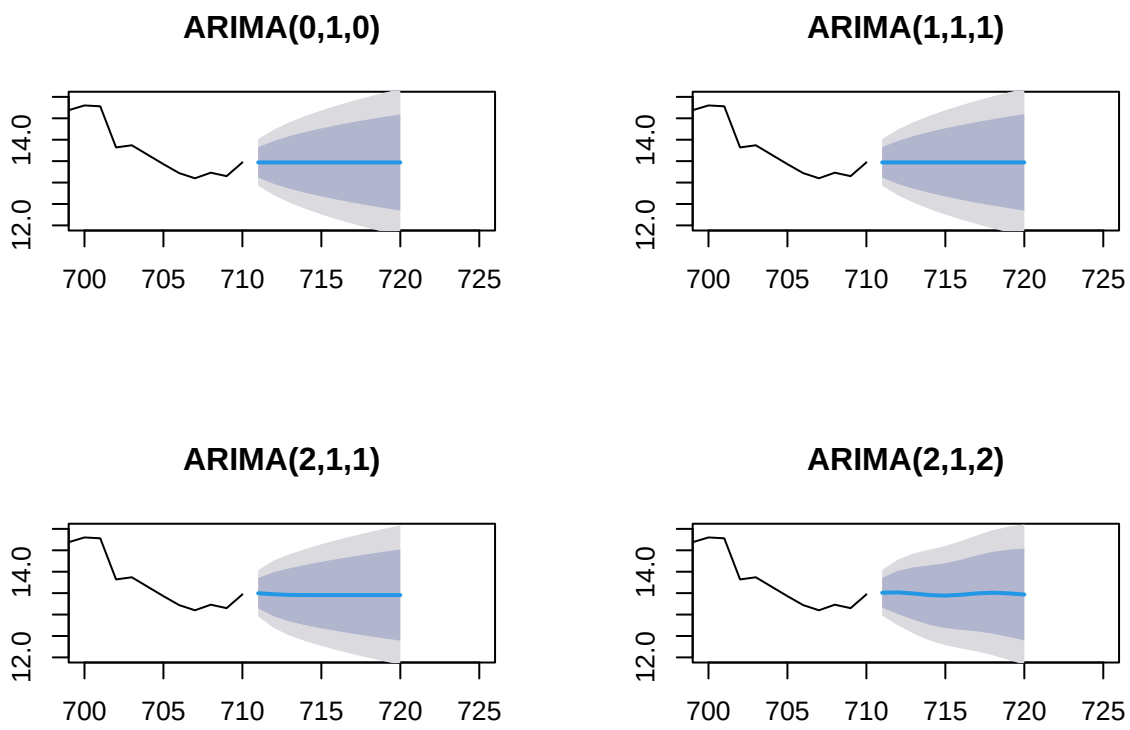
```
                        MAE_entrenamiento = numeric(), MAPE_prueba = numeric(),
                        RMSE_prueba = numeric(),
                        MAE_prueba = numeric(), stringsAsFactors = FALSE)

for (i in lista_modelos){
  modelo <- Arima(conjunto_entrenamiento, order = i)

  ajuste_entrenamiento <- fitted(modelo)
  metricas_entrenamiento <- calcular_error(conjunto_entrenamiento,
                                           ajuste_entrenamiento)

  pronostico_prueba <- forecast(modelo, h = length(conjunto_prueba))
  metricas_prueba <- calcular_error(conjunto_prueba, pronostico_prueba$mean)

  resultados <- rbind(
    resultados,
    data.frame(Modelo = paste("ARIMA(",paste(i, collapse = ","), ")", sep = ""),
               MAPE_entrenamiento = metricas_entrenamiento[1],
               RMSE_entrenamiento = metricas_entrenamiento[2],
               MAE_entrenamiento = metricas_entrenamiento[3],
               MAPE_prueba = metricas_prueba[1],
               RMSE_prueba = metricas_prueba[2],
               MAE_prueba = metricas_prueba[3]))
}

resultados
```

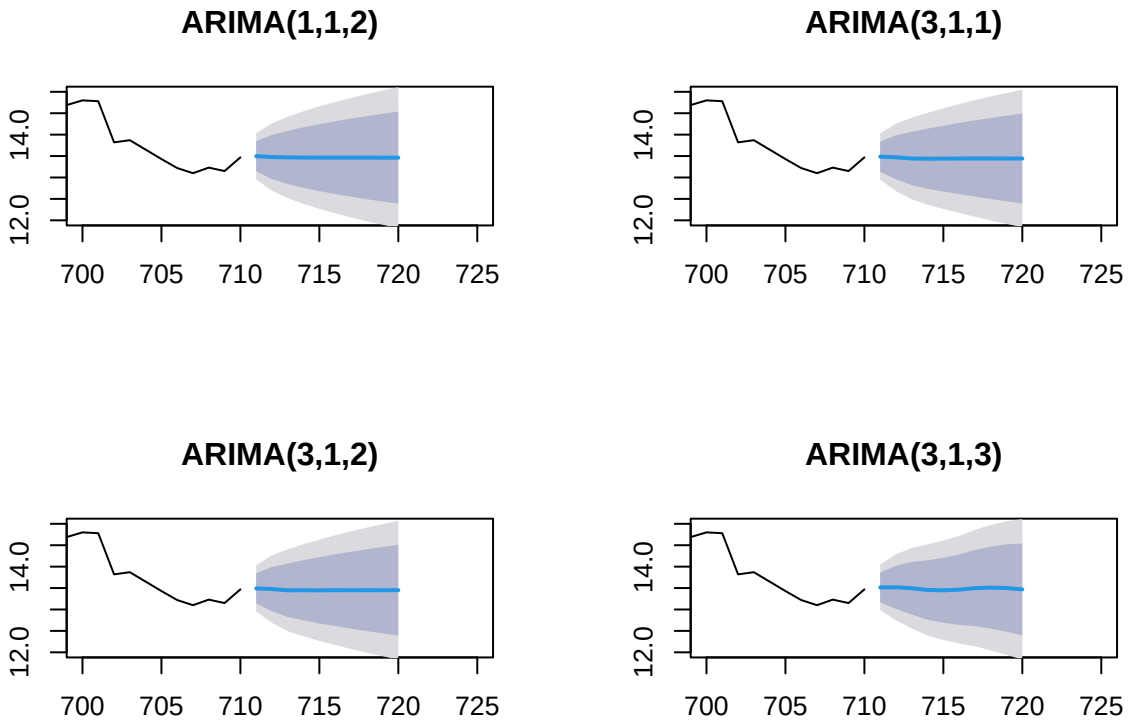
```
##          Modelo MAPE_entrenamiento RMSE_entrenamiento MAE_entrenamiento
## 1  ARIMA(0,1,0)         0.03099713          0.2732503         0.1899857
## 2  ARIMA(1,1,1)         0.03095649          0.2729604         0.1895113
## 3  ARIMA(2,1,1)         0.03108209          0.2697134         0.1898674
## 4  ARIMA(2,1,2)         0.03095450          0.2637547         0.1886676
## 5  ARIMA(1,1,2)         0.03104658          0.2703664         0.1896904
## 6  ARIMA(3,1,1)         0.03111841          0.2689581         0.1904863
## 7  ARIMA(3,1,2)         0.03110567          0.2690607         0.1904327
## 8  ARIMA(3,1,3)         0.03080753          0.2632806         0.1881053
## 9  ARIMA(4,1,1)         0.03110720          0.2688290         0.1903891
## 10 ARIMA(4,1,2)         0.03094943          0.2637173         0.1885725
## 11 ARIMA(4,1,3)         0.03094763          0.2637102         0.1885579
## 12 ARIMA(4,1,4)         0.03069157          0.2601880         0.1868172
## 13 ARIMA(5,1,1)         0.03097103          0.2676913         0.1892944
## 14 ARIMA(5,1,2)         0.03096092          0.2635784         0.1885327
## 15 ARIMA(5,1,3)         0.03090653          0.2633098         0.1881084
## 16 ARIMA(5,1,4)         0.03081148          0.2631988         0.1881538
## 17 ARIMA(5,1,5)         0.03069562          0.2620206         0.1874537
##    MAPE_prueba RMSE_prueba MAE_prueba
## 1   0.08571886    1.536971   1.149577
## 2   0.08654513    1.548334   1.160747
## 3   0.08677786    1.551256   1.163851
## 4   0.08730715    1.557997   1.170908
## 5   0.08674393    1.550886   1.163406
## 6   0.08696192    1.553672   1.166319
## 7   0.08732753    1.558535   1.171229
## 8   0.08710450    1.555230   1.168173
```

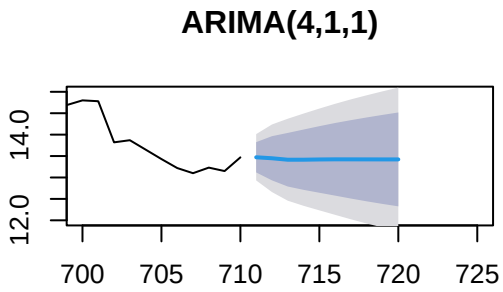
```
## 9   0.08687723    1.552539   1.165180
## 10  0.08723149    1.556982   1.169891
## 11  0.08719820    1.556539   1.169445
## 12  0.08922288    1.581745   1.196332
## 13  0.08218273    1.485313   1.101120
## 14  0.08693930    1.553099   1.165969
## 15  0.08500058    1.526067   1.139702
## 16  0.08619673    1.542651   1.155902
## 17  0.08838494    1.571769   1.185292
```

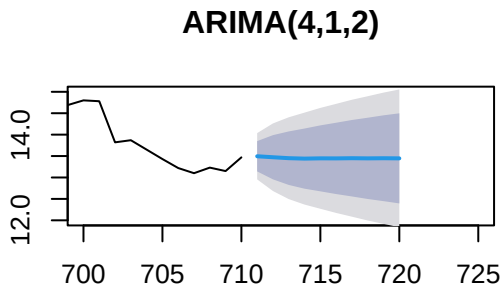
Encontrados los posibles modelos, se procede a graficar los resultados:

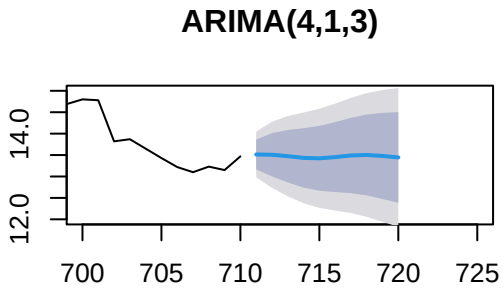
```
# Graficando los resultados el pronostico de todos los modelos
par(mfrow=c(2,2))
for (i in 1:length(lista_modelos)) {
  orden <- lista_modelos[[i]]
  modelo <- Arima(serie, order = orden)
  pronostico <- forecast(modelo, h = 10)
  plot(pronostico, main = paste("ARIMA(", paste(orden, collapse = ","), ")",
                                sep = ""), xlim=c(700,725), ylim=c(12,15))
}
```

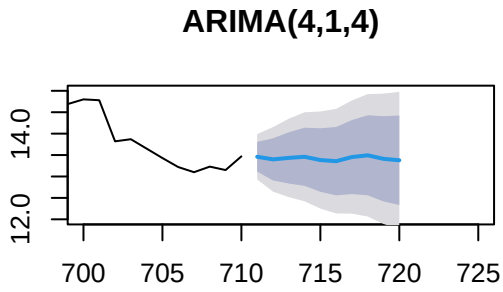


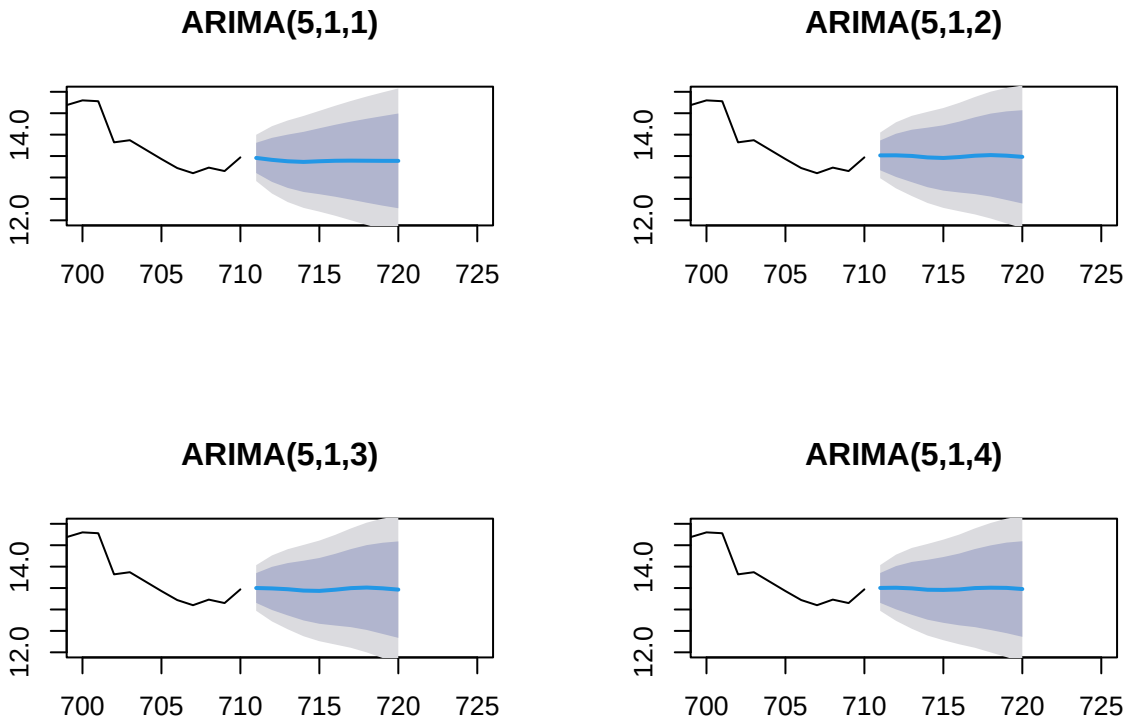

ARIMA(1,1,2)
ARIMA(3,1,1)
ARIMA(3,1,2)
ARIMA(3,1,3)
12.0
14.0
700
705
710
715
720
725

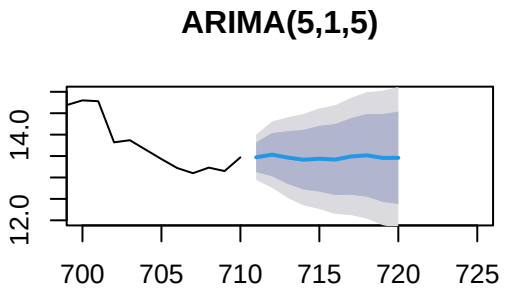

ARIMA(4,1,1)
14.0
12.0
700
705
710
715
720
725


ARIMA(4,1,2)
14.0
12.0
700
705
710
715
720
725


ARIMA(4,1,3)
14.0
12.0
700
705
710
715
720
725


ARIMA(4,1,4)
14.0
12.0
700
705
710
715
720
725


ARIMA(5,1,1)
ARIMA(5,1,2)
ARIMA(5,1,3)
ARIMA(5,1,4)
12.0
14.0
700
705
710
715
720
725


ARIMA(5,1,5)
14.0
12.0
700
705
710
715
720
725