# Series de tiempo

**Equipo 3**

**2025-04-01**

## Actividad - Series de Tiempo

**Obtención de datos:** La empresa seleccionada fue **Tesla (TSLA)**

```
inicio <- "2013-01-01"
fin <- "2025-02-28"

getSymbols("TSLA", from = inicio, to = fin)
```

```
## [1] "TSLA"
```

Precios de cierre ajustados:

```
tsla_ajustados <- TSLA$TSLA.Adjusted
tsla_df <- data.frame(
  fecha = index(tsla_ajustados),
  precio = as.numeric(tsla_ajustados)
)
```

```
datatable(tsla_df,
          options = list(pageLength = 10,
                         autoWidth = TRUE,
                         dom = 'Bfrtip',
          caption = "Tabla de precios ajustados de TSLA"))
```

### Paso 1: Visualizar

*Grafica la Serie de Tiempo, así como su descomposición en tendencia, estacionalidad y errores.*

```
ggplot(tsla_df, aes(x = fecha, y = precio)) +
  geom_line(color = "#E31937") +
  labs(
    title = "Serie de precios ajustados de TSLA (2013-2025)",
    x = "Fecha",
    y = "Precio ajustado (USD)"
  ) +
  theme_minimal()
```

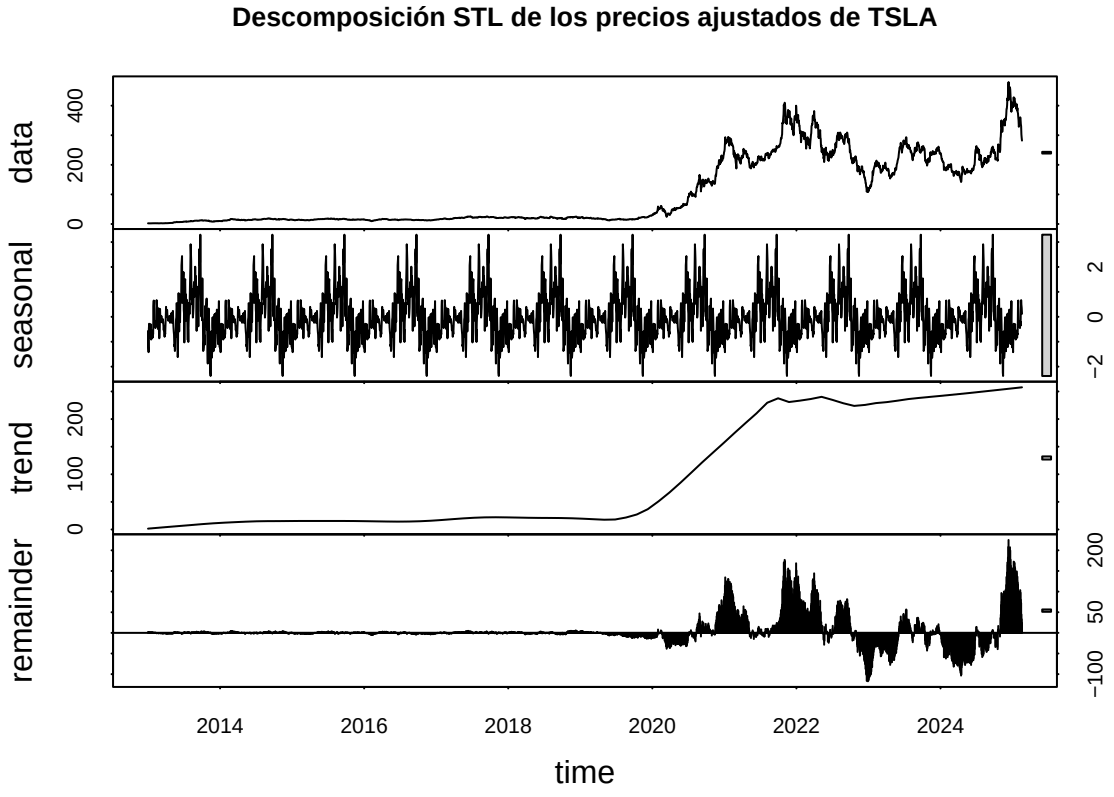

**Visualización general**

**Transformar a serie de tiempo** Usamos frequency=252 asumiendo ~252 días de trading por año.

```
tsla_ts <- ts(tsla_df$precio, frequency = 252, start = c(2013, 1))
```

**Descomposición de la serie** En 2020 todo cambió, el precio explotó y empezó a moverse de forma muy diferente Curiosamente, los ciclos estacionales siguen el mismo patrón aunque el precio se multiplicó por 400. Después de 2020, cuando el precio se dispara hacia arriba lo hace con más fuerza que cuando cae. La tendencia muestra dos grandes oleadas de crecimiento (2020 y 2024) con un período de estabilidad entre medias. Las fluctuaciones diarias no aumentaron proporcionalmente con el precio, lo que sugiere que Tesla dejó de comportarse como las acciones normales.

```
# s.window="periodic" fuerza la estacionalidad a ser periódica (anual en este caso)
tsla_stl <- stl(tsla_ts, s.window = "periodic", robust = TRUE)

# Graficar la descomposición
plot(tsla_stl, main = "Descomposición STL de los precios ajustados de TSLA")
```




**Paso 2: Estacionariza la serie**

*Encuentra el valor d y verifica que la d-ésima diferencia sea un proceso estacionario, también revisa si esta tiene o no una tendencia determinística.*

```
adf_test_original <- adf.test(tsla_ts)
```

**Prueba de Dickey-Fuller aumentada para verificar estacionariedad en la serie original**

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag    ADF p.value
##  [1,]   0 -0.264   0.568
##  [2,]   1 -0.245   0.573
##  [3,]   2 -0.283   0.563
##  [4,]   3 -0.282   0.563
##  [5,]   4 -0.345   0.545
##  [6,]   5 -0.282   0.563
##  [7,]   6 -0.345   0.545
##  [8,]   7 -0.544   0.484
```

```
##  [9,]   8 -0.436   0.518
## Type 2: with drift no trend
##       lag   ADF p.value
##  [1,]   0 -1.11   0.662
##  [2,]   1 -1.10   0.668
##  [3,]   2 -1.13   0.656
##  [4,]   3 -1.13   0.656
##  [5,]   4 -1.19   0.635
##  [6,]   5 -1.13   0.656
##  [7,]   6 -1.19   0.635
##  [8,]   7 -1.38   0.569
##  [9,]   8 -1.27   0.605
## Type 3: with drift and trend
##       lag   ADF p.value
##  [1,]   0 -2.57   0.335
##  [2,]   1 -2.55   0.344
##  [3,]   2 -2.60   0.325
##  [4,]   3 -2.60   0.325
##  [5,]   4 -2.67   0.294
##  [6,]   5 -2.60   0.323
##  [7,]   6 -2.67   0.293
##  [8,]   7 -2.91   0.195
##  [9,]   8 -2.78   0.248
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
print(adf_test_original)
```

```
## $type1
##       lag        ADF   p.value
##  [1,]   0 -0.2643526 0.5678736
##  [2,]   1 -0.2452868 0.5733592
##  [3,]   2 -0.2828662 0.5625469
##  [4,]   3 -0.2817939 0.5628555
##  [5,]   4 -0.3452188 0.5446071
##  [6,]   5 -0.2821166 0.5627626
##  [7,]   6 -0.3446750 0.5447635
##  [8,]   7 -0.5443982 0.4842317
##  [9,]   8 -0.4360714 0.5184672
##
## $type2
##       lag       ADF   p.value
##  [1,]   0 -1.112565 0.6619239
##  [2,]   1 -1.095168 0.6680822
##  [3,]   2 -1.129933 0.6557759
##  [4,]   3 -1.129021 0.6560988
##  [5,]   4 -1.187880 0.6352636
##  [6,]   5 -1.129743 0.6558432
##  [7,]   6 -1.187484 0.6354039
##  [8,]   7 -1.376227 0.5685922
##  [9,]   8 -1.273347 0.6050099
##
## $type3
##       lag       ADF   p.value
```

```
##  [1,]   0 -2.572451 0.3347574
##  [2,]   1 -2.549495 0.3444231
##  [3,]   2 -2.596366 0.3246880
##  [4,]   3 -2.596183 0.3247650
##  [5,]   4 -2.669993 0.2936872
##  [6,]   5 -2.601126 0.3226838
##  [7,]   6 -2.670709 0.2933855
##  [8,]   7 -2.905042 0.1947191
##  [9,]   8 -2.778841 0.2478564
```

Los resultados de la prueba ADF muestran que la serie original **no es estacionaria** (todos los p-valores > 0.05), por lo tanto debemos diferenciarla.

```r
tsla_diff1 <- diff(tsla_ts)
adf_test_diff1 <- adf.test(tsla_diff1)
```

**Primera diferencia**

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag   ADF p.value
##  [1,]   0 -55.8    0.01
##  [2,]   1 -38.5    0.01
##  [3,]   2 -31.5    0.01
##  [4,]   3 -26.5    0.01
##  [5,]   4 -24.4    0.01
##  [6,]   5 -21.6    0.01
##  [7,]   6 -18.4    0.01
##  [8,]   7 -18.2    0.01
##  [9,]   8 -15.7    0.01
## Type 2: with drift no trend
##       lag   ADF p.value
##  [1,]   0 -55.8    0.01
##  [2,]   1 -38.5    0.01
##  [3,]   2 -31.5    0.01
##  [4,]   3 -26.5    0.01
##  [5,]   4 -24.4    0.01
##  [6,]   5 -21.6    0.01
##  [7,]   6 -18.4    0.01
##  [8,]   7 -18.2    0.01
##  [9,]   8 -15.7    0.01
## Type 3: with drift and trend
##       lag   ADF p.value
##  [1,]   0 -55.8    0.01
##  [2,]   1 -38.5    0.01
##  [3,]   2 -31.5    0.01
##  [4,]   3 -26.5    0.01
##  [5,]   4 -24.4    0.01
##  [6,]   5 -21.6    0.01
```

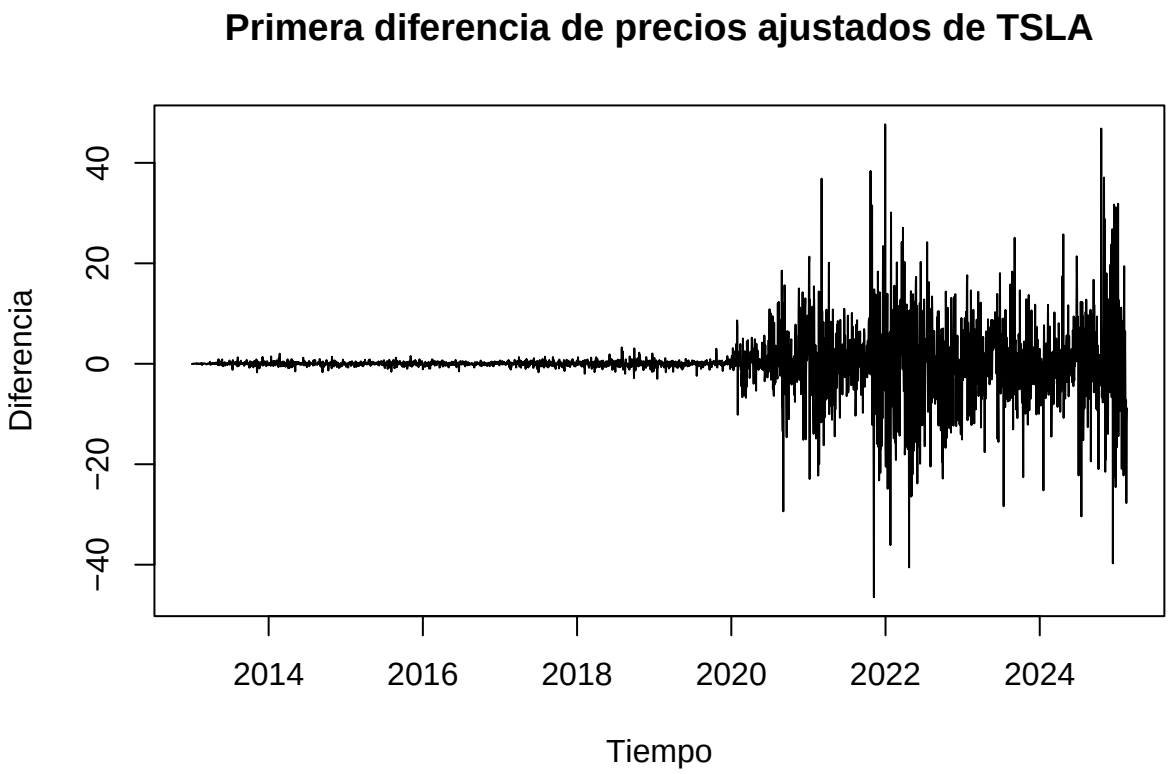
```
##  [7,]   6 -18.4    0.01
##  [8,]   7 -18.2    0.01
##  [9,]   8 -15.7    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

La prueba ADF para la primera diferencia muestra p-valores de 0.01 en todos los casos, lo que nos permite rechazar contundentemente la hipótesis nula de no estacionariedad. Por lo tanto, **la primera diferencia es estacionaria (d=1)**.

```
print(adf_test_diff1)
```

```
## $type1
##       lag       ADF p.value
##  [1,]   0 -55.80913    0.01
##  [2,]   1 -38.48076    0.01
##  [3,]   2 -31.46718    0.01
##  [4,]   3 -26.45169    0.01
##  [5,]   4 -24.40114    0.01
##  [6,]   5 -21.60911    0.01
##  [7,]   6 -18.36710    0.01
##  [8,]   7 -18.20688    0.01
##  [9,]   8 -15.72375    0.01
##
## $type2
##       lag       ADF p.value
##  [1,]   0 -55.81363    0.01
##  [2,]   1 -38.48840    0.01
##  [3,]   2 -31.47717    0.01
##  [4,]   3 -26.46285    0.01
##  [5,]   4 -24.41457    0.01
##  [6,]   5 -21.62311    0.01
##  [7,]   6 -18.37965    0.01
##  [8,]   7 -18.22170    0.01
##  [9,]   8 -15.73656    0.01
##
## $type3
##       lag       ADF p.value
##  [1,]   0 -55.80790    0.01
##  [2,]   1 -38.48549    0.01
##  [3,]   2 -31.47549    0.01
##  [4,]   3 -26.46164    0.01
##  [5,]   4 -24.41400    0.01
##  [6,]   5 -21.62233    0.01
##  [7,]   6 -18.37773    0.01
##  [8,]   7 -18.22044    0.01
##  [9,]   8 -15.73382    0.01
```

```
plot(tsla_diff1, main = "Primera diferencia de precios ajustados de TSLA",
     ylab = "Diferencia", xlab = "Tiempo")
```




Si tsla_diff1 todavía tuviera una tendencia determinística, significaría que su media no es constante, sino que cambia linealmente con el tiempo, ignorar esto llevaría a un mal modelo.

```
time_trend <- 1:length(tsla_diff1)
trend_model <- lm(tsla_diff1 ~ time_trend)
summary(trend_model)
```

**Verificamos si existe tendencia determinística en la serie diferenciada**

```
##
## Call:
## lm(formula = tsla_diff1 ~ time_trend)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.608  -0.549  -0.024   0.488  47.537
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.194e-02  2.130e-01   0.056    0.955
## time_trend  5.201e-05  1.206e-04   0.431    0.666
##
## Residual standard error: 5.886 on 3055 degrees of freedom
```

```
## Multiple R-squared:  6.083e-05,  Adjusted R-squared:  -0.0002665
## F-statistic: 0.1859 on 1 and 3055 DF,  p-value: 0.6664
```

Observamos que el coeficiente asociado a la variable de tiempo (time_trend) no es estadísticamente significativo por su p-value (0.666), sugiriendo que **no hay evidencia de una tendencia lineal determinística** en la serie de las primeras diferencias de los precios.

1. Detectamos no estacionariedad en la serie original (tsla_ts) con ADF.
2. Aplicamos la primera diferencia (d=1) y el ADF confirmó que la serie resultante (tsla_diff1) sí es estacionaria (p=0.01).
3. Verificamos ausencia de tendencia determinística en la serie estacionaria (tsla_diff1) con el modelo de regresión (p=0.666).
4. **La d es igual a 1.**

### Paso 3: Identifica los valor $p$ y $q$

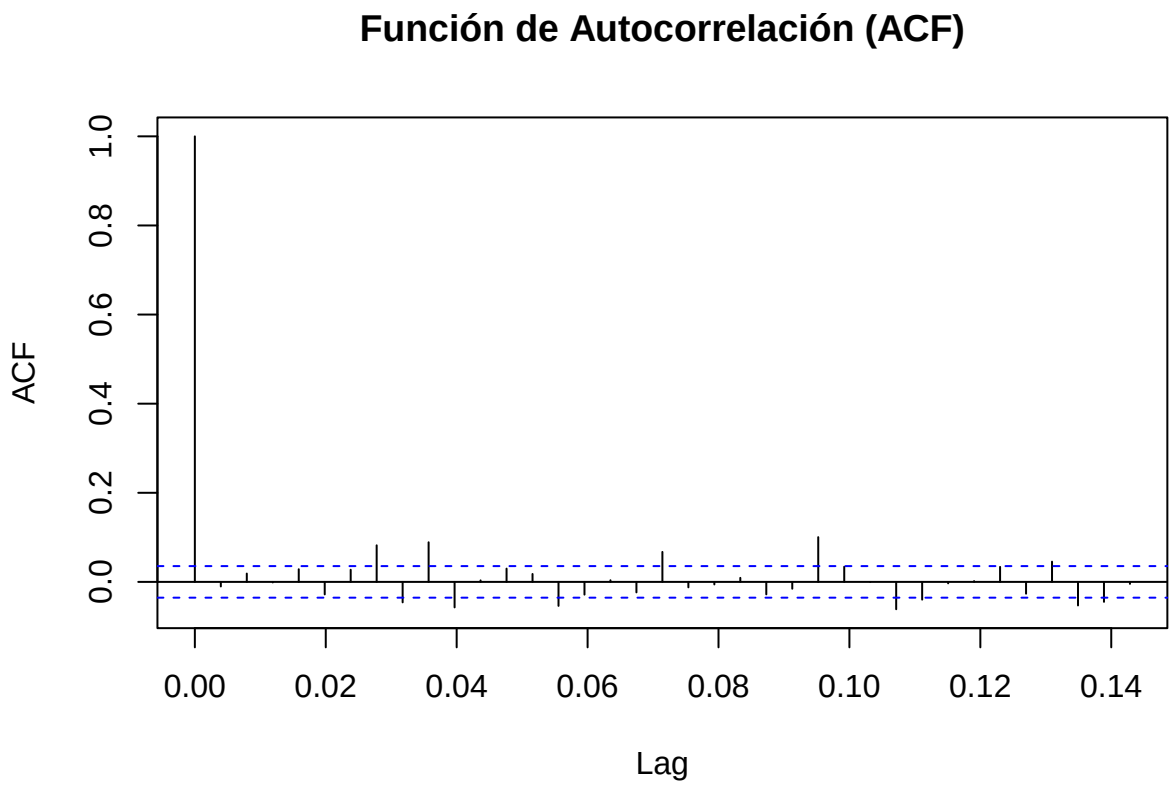
*Estima la función de autocorrelación y la función de autocorrelación parcial. A partir de sus correlogramas estima los valores de $p$ y $q$.*

```r
acf_tsla <- acf(tsla_diff1, lag.max = 36, plot = FALSE)
acf_tsla
```

**Función de autocorrelación (ACF)**

```
##
## Autocorrelations of series 'tsla_diff1', by lag
##
## 0.00000 0.00397 0.00794 0.01190 0.01587 0.01984 0.02381 0.02778 0.03175 0.03571
##   1.000  -0.010   0.019  -0.001   0.028  -0.028   0.027   0.082  -0.046   0.089
## 0.03968 0.04365 0.04762 0.05159 0.05556 0.05952 0.06349 0.06746 0.07143 0.07540
##  -0.057   0.003   0.030   0.018  -0.054  -0.029   0.004  -0.024   0.067  -0.012
## 0.07937 0.08333 0.08730 0.09127 0.09524 0.09921 0.10317 0.10714 0.11111 0.11508
##  -0.006   0.009  -0.028  -0.015   0.100   0.034   0.000  -0.061  -0.040  -0.003
## 0.11905 0.12302 0.12698 0.13095 0.13492 0.13889 0.14286
##   0.002   0.034  -0.026   0.045  -0.053  -0.045  -0.004
```

```r
plot(acf_tsla, main = "Función de Autocorrelación (ACF)")
```

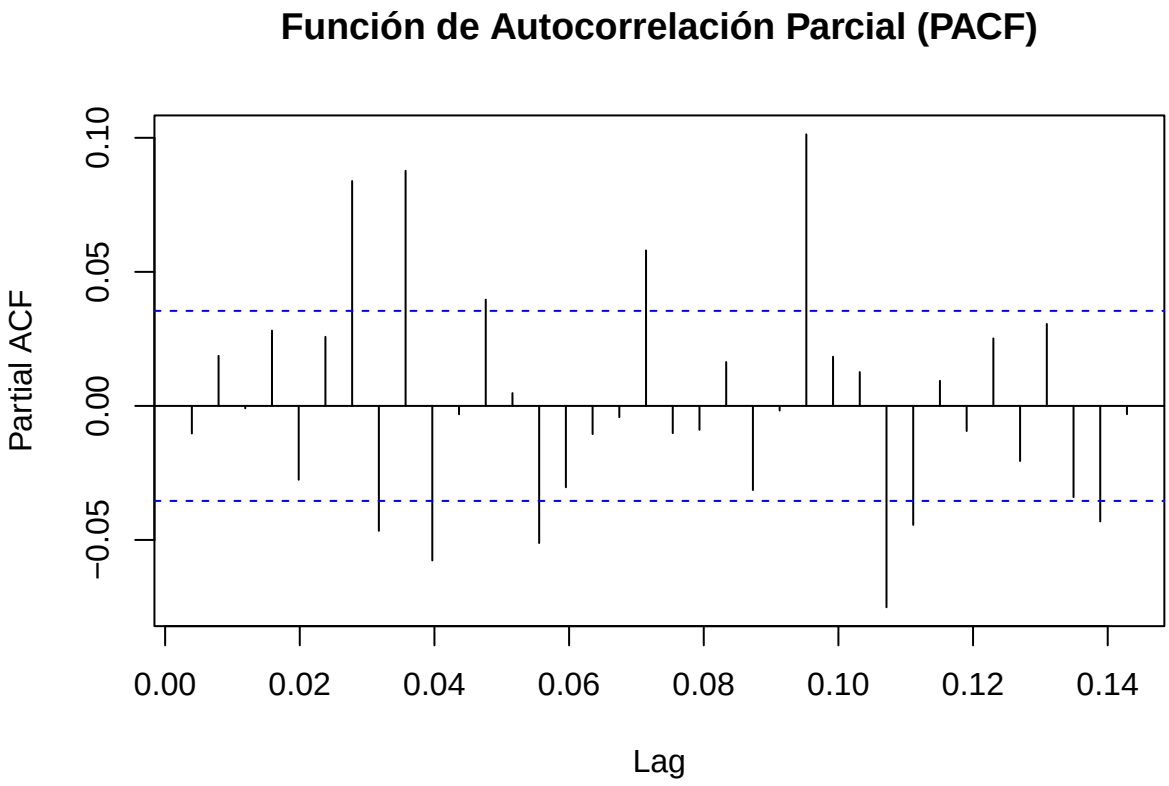



```
pacf_tsla <- pacf(tsla_diff1, lag.max = 36, plot = FALSE)
pacf_tsla
```

**Función de autocorrelación parcial (PACF)**

```
##
## Partial autocorrelations of series 'tsla_diff1', by lag
##
## 0.00397 0.00794 0.01190 0.01587 0.01984 0.02381 0.02778 0.03175 0.03571 0.03968
##  -0.010   0.019  -0.001   0.028  -0.028   0.026   0.084  -0.047   0.088  -0.058
## 0.04365 0.04762 0.05159 0.05556 0.05952 0.06349 0.06746 0.07143 0.07540 0.07937
##  -0.003   0.040   0.005  -0.051  -0.030  -0.010  -0.004   0.058  -0.010  -0.009
## 0.08333 0.08730 0.09127 0.09524 0.09921 0.10317 0.10714 0.11111 0.11508 0.11905
##   0.016  -0.031  -0.002   0.101   0.018   0.013  -0.075  -0.044   0.009  -0.009
## 0.12302 0.12698 0.13095 0.13492 0.13889 0.14286
##   0.025  -0.021   0.031  -0.034  -0.043  -0.003
```

```
plot(pacf_tsla, main = "Función de Autocorrelación Parcial (PACF)")
```




Basándonos en las gráficas de ACF y PACF, vamos a probar varios modelos ARIMA para encontrar el mejor ajuste con los criterios de AIC, BIC y LogLikelihood:

```
mostrar_metricas <- function(modelo, nombre) {
  cat("Modelo:", nombre, "\n")
  cat("AIC:", AIC(modelo), "\n")
  cat("BIC:", BIC(modelo), "\n")
  cat("Log-Likelihood:", logLik(modelo), "\n\n")
  return(data.frame(
    Modelo = nombre,
    AIC = AIC(modelo),
    BIC = BIC(modelo),
    LogLik = as.numeric(logLik(modelo))
  ))
}
```

```
modelos <- list()
resultados <- data.frame(Modelo = character(), AIC = numeric(), BIC = numeric(), LogLik = numeric())
```

```
# Modelo ARIMA(1,1,1) - modelo simple balanceado
modelo_111 <- Arima(tsla_ts, order = c(1, 1, 1))
modelos[["ARIMA(1,1,1)"]] <- modelo_111
resultados <- rbind(resultados, mostrar_metricas(modelo_111, "ARIMA(1,1,1)"))
```

**Modelos simples (p,q entre 1 y 3)**

```
## Modelo: ARIMA(1,1,1)
## AIC: 19517.89
## BIC: 19535.97
## Log-Likelihood: -9755.947
```

```
# Modelo ARIMA(2,1,2) - modelo simple con dos rezagos
modelo_212 <- Arima(tsla_ts, order = c(2, 1, 2))
modelos[["ARIMA(2,1,2)"]] <- modelo_212
resultados <- rbind(resultados, mostrar_metricas(modelo_212, "ARIMA(2,1,2)"))
```

```
## Modelo: ARIMA(2,1,2)
## AIC: 19517.34
## BIC: 19547.47
## Log-Likelihood: -9753.671
```

```
# Modelo ARIMA(3,1,3) - modelo simple con tres rezagos
modelo_313 <- Arima(tsla_ts, order = c(3, 1, 3))
modelos[["ARIMA(3,1,3)"]] <- modelo_313
resultados <- rbind(resultados, mostrar_metricas(modelo_313, "ARIMA(3,1,3)"))
```

```
## Modelo: ARIMA(3,1,3)
## AIC: 19524.3
## BIC: 19566.48
## Log-Likelihood: -9755.15
```

```
# Modelos adicionales simples
# Modelo ARIMA(1,1,0) - modelo AR simple
modelo_110 <- Arima(tsla_ts, order = c(1, 1, 0))
modelos[["ARIMA(1,1,0)"]] <- modelo_110
resultados <- rbind(resultados, mostrar_metricas(modelo_110, "ARIMA(1,1,0)"))
```

```
## Modelo: ARIMA(1,1,0)
## AIC: 19515.89
## BIC: 19527.94
## Log-Likelihood: -9755.945
```

```
# Modelo ARIMA(0,1,1) - modelo MA simple
modelo_011 <- Arima(tsla_ts, order = c(0, 1, 1))
modelos[["ARIMA(0,1,1)"]] <- modelo_011
resultados <- rbind(resultados, mostrar_metricas(modelo_011, "ARIMA(0,1,1)"))
```

```
## Modelo: ARIMA(0,1,1)
## AIC: 19515.9
## BIC: 19527.95
## Log-Likelihood: -9755.95
```

```r
# Modelo ARIMA(7,1,0) - identificado por pico en rezago 7 del PACF
modelo_710 <- Arima(tsla_ts, order = c(7, 1, 0))
modelos[["ARIMA(7,1,0)"]] <- modelo_710
resultados <- rbind(resultados, mostrar_metricas(modelo_710, "ARIMA(7,1,0)"))
```

### Modelos basados en ACF y PACF

```
## Modelo: ARIMA(7,1,0)
## AIC: 19497.92
## BIC: 19546.12
## Log-Likelihood: -9740.958
```

```r
# Modelo ARIMA(0,1,7) - identificado por pico en rezago 7 del ACF
modelo_017 <- Arima(tsla_ts, order = c(0, 1, 7))
modelos[["ARIMA(0,1,7)"]] <- modelo_017
resultados <- rbind(resultados, mostrar_metricas(modelo_017, "ARIMA(0,1,7)"))
```

```
## Modelo: ARIMA(0,1,7)
## AIC: 19493.03
## BIC: 19541.23
## Log-Likelihood: -9738.513
```

```r
# Modelo ARIMA(9,1,0) - identificado por pico en rezago 9 del PACF
modelo_910 <- Arima(tsla_ts, order = c(9, 1, 0))
modelos[["ARIMA(9,1,0)"]] <- modelo_910
resultados <- rbind(resultados, mostrar_metricas(modelo_910, "ARIMA(9,1,0)"))
```

```
## Modelo: ARIMA(9,1,0)
## AIC: 19471.32
## BIC: 19531.57
## Log-Likelihood: -9725.659
```

```r
# Modelo ARIMA(0,1,9) - identificado por pico en rezago 9 del ACF
modelo_019 <- Arima(tsla_ts, order = c(0, 1, 9))
modelos[["ARIMA(0,1,9)"]] <- modelo_019
resultados <- rbind(resultados, mostrar_metricas(modelo_019, "ARIMA(0,1,9)"))
```

```
## Modelo: ARIMA(0,1,9)
## AIC: 19474.39
## BIC: 19534.64
## Log-Likelihood: -9727.195
```

```r
# Modelo ARIMA(7,1,7) - combinación de efectos AR y MA en rezago 7
modelo_717 <- Arima(tsla_ts, order = c(7, 1, 7))
modelos[["ARIMA(7,1,7)"]] <- modelo_717
resultados <- rbind(resultados, mostrar_metricas(modelo_717, "ARIMA(7,1,7)"))
```

```
## Modelo: ARIMA(7,1,7)
## AIC: 19438.24
## BIC: 19528.62
## Log-Likelihood: -9704.122
```

```r
# Modelo ARIMA(9,1,9) - combinación de efectos AR y MA en rezago 9
modelo_919 <- Arima(tsla_ts, order = c(9, 1, 9))
modelos[["ARIMA(9,1,9)"]] <- modelo_919
resultados <- rbind(resultados, mostrar_metricas(modelo_919, "ARIMA(9,1,9)"))
```

```
## Modelo: ARIMA(9,1,9)
## AIC: 19437.91
## BIC: 19552.39
## Log-Likelihood: -9699.954
```

```r
# Modelo ARIMA(24,1,0) - por pico más alto en PACF (rezago 24)
modelo_2410 <- Arima(tsla_ts, order = c(24, 1, 0))
modelos[["ARIMA(24,1,0)"]] <- modelo_2410
resultados <- rbind(resultados, mostrar_metricas(modelo_2410, "ARIMA(24,1,0)"))
```

```
## Modelo: ARIMA(24,1,0)
## AIC: 19427.67
## BIC: 19578.3
## Log-Likelihood: -9688.835
```

```r
# Modelo ARIMA(0,1,24) - por pico más alto en ACF (rezago 24)
modelo_0124 <- Arima(tsla_ts, order = c(0, 1, 24))
modelos[["ARIMA(0,1,24)"]] <- modelo_0124
resultados <- rbind(resultados, mostrar_metricas(modelo_0124, "ARIMA(0,1,24)"))
```

```
## Modelo: ARIMA(0,1,24)
## AIC: 19431.44
## BIC: 19582.07
## Log-Likelihood: -9690.72
```

```r
print(resultados)
```

**Resultados**

```
##           Modelo      AIC      BIC    LogLik
## 1   ARIMA(1,1,1) 19517.89 19535.97 -9755.947
## 2   ARIMA(2,1,2) 19517.34 19547.47 -9753.671
## 3   ARIMA(3,1,3) 19524.30 19566.48 -9755.150
## 4   ARIMA(1,1,0) 19515.89 19527.94 -9755.945
## 5   ARIMA(0,1,1) 19515.90 19527.95 -9755.950
## 6   ARIMA(7,1,0) 19497.92 19546.12 -9740.958
## 7   ARIMA(0,1,7) 19493.03 19541.23 -9738.513
## 8   ARIMA(9,1,0) 19471.32 19531.57 -9725.659
## 9   ARIMA(0,1,9) 19474.39 19534.64 -9727.195
## 10  ARIMA(7,1,7) 19438.24 19528.62 -9704.122
## 11  ARIMA(9,1,9) 19437.91 19552.39 -9699.954
## 12 ARIMA(24,1,0) 19427.67 19578.30 -9688.835
## 13 ARIMA(0,1,24) 19431.44 19582.07 -9690.720
```

```r
top_5_aic <- resultados[order(resultados$AIC), ][1:5, ]
top_5_bic <- resultados[order(resultados$BIC), ][1:5, ]
top_5_loglik <- resultados[order(-resultados$LogLik), ][1:5, ]
```

Top 5 modelos según **AIC**: 1. **ARIMA(24,1,0)** 2. ARIMA(0,1,24) 3. ARIMA(9,1,9) 4. **ARIMA(7,1,7)** 5. ARIMA(9,1,0)

```r
print(top_5_aic)
```

```
##           Modelo      AIC      BIC    LogLik
## 12 ARIMA(24,1,0) 19427.67 19578.30 -9688.835
## 13 ARIMA(0,1,24) 19431.44 19582.07 -9690.720
## 11  ARIMA(9,1,9) 19437.91 19552.39 -9699.954
## 10  ARIMA(7,1,7) 19438.24 19528.62 -9704.122
## 8   ARIMA(9,1,0) 19471.32 19531.57 -9725.659
```

Top 5 modelos según **BIC**: 1. ARIMA(1,1,0) 2. ARIMA(0,1,1) 3. **ARIMA(7,1,7)** 4. ARIMA(9,1,0) 5. ARIMA(0,1,9)

```r
print(top_5_bic)
```

```
##          Modelo      AIC      BIC    LogLik
## 4  ARIMA(1,1,0) 19515.89 19527.94 -9755.945
## 5  ARIMA(0,1,1) 19515.90 19527.95 -9755.950
## 10 ARIMA(7,1,7) 19438.24 19528.62 -9704.122
## 8  ARIMA(9,1,0) 19471.32 19531.57 -9725.659
## 9  ARIMA(0,1,9) 19474.39 19534.64 -9727.195
```

Top 5 modelos según **LogLik**: 1. **ARIMA(24,1,0)** 2. ARIMA(0,1,24) 3. ARIMA(9,1,9) 4. **ARIMA(7,1,7)** 5. ARIMA(9,1,0)

```r
print(top_5_loglik)
```

```
##           Modelo      AIC      BIC    LogLik
## 12 ARIMA(24,1,0) 19427.67 19578.30 -9688.835
## 13 ARIMA(0,1,24) 19431.44 19582.07 -9690.720
## 11  ARIMA(9,1,9) 19437.91 19552.39 -9699.954
## 10  ARIMA(7,1,7) 19438.24 19528.62 -9704.122
## 8   ARIMA(9,1,0) 19471.32 19531.57 -9725.659
```

Basado en los resultados, los 2 mejores modelos para seleccionar son: 1. **ARIMA(24,1,0)**: Tiene el mejor AIC (19427.67) y el mejor LogLik (-9688.835). Captura los patrones mensuales en los datos de Tesla. 2. **ARIMA(7,1,7)**: Tiene un AIC muy competitivo (19438.24, solo 10 puntos peor que el mejor) y el tercer mejor BIC (19528.62) Captura patrones semanales. También podríamos considerar ARIMA(9,1,9) que está arriba en AIC y LogLik pero solo por muy poco, mientras que en BIC es peor por una diferencia muy grande de 23.77.

Es interesante observar que aunque ARIMA(1,1,0) tiene el mejor BIC, su AIC y LogLik son mucho peores, lo que sugiere que es demasiado simple y no captura adecuadamente la dinámica de los datos. Los dos modelos recomendados ofrecen el mejor balance entre los 3 criterios que consideramos relevantes.

**Paso 4: Ajusta el Modelo ARIMA**

*Ajusta un modelo $ARIMA(p, d, q)$ con los valores encontrados en los pasos previos. Revisa sus residuales, comparalos con el proceso estacionario (ruido blanco) que encontraste en el paso 2 y muestra su valor AIC.*

```
modelo_2410 <- Arima(tsla_ts, order = c(24, 1, 0))
modelo_717 <- Arima(tsla_ts, order = c(7, 1, 7))
```

**Ajustamos los dos mejores modelos seleccionados**

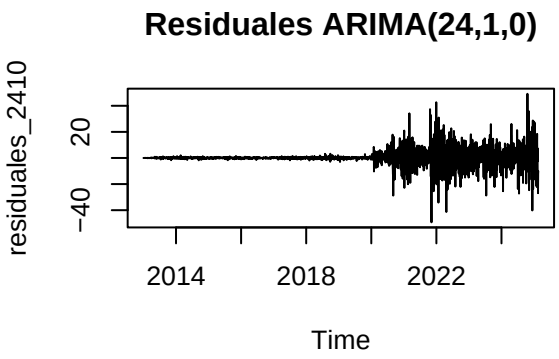
```
residuales_2410 <- residuals(modelo_2410)
residuales_717 <- residuals(modelo_717)
```

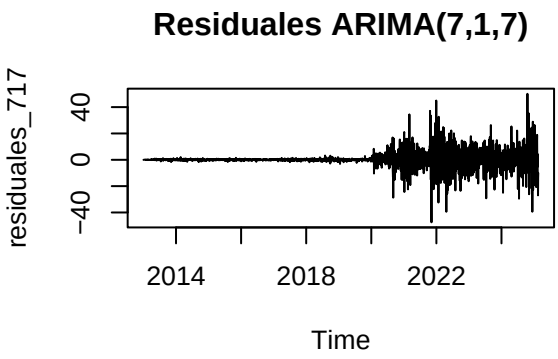
**Extraemos residuales**

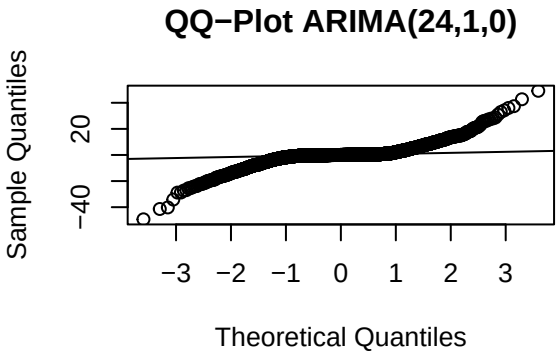
**Análisis de residuales**

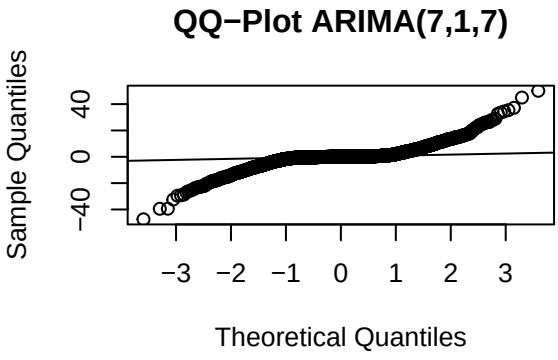
- Ambos modelos muestran patrones muy similares. Se observa un aumento notable en la volatilidad después de 2020, coincidiendo con la pandemia y el gran crecimiento de Tesla. **Esto indica heterocedasticidad (varianza no constante) en ambos modelos**, lo que sugiere que los precios de Tesla se volvieron mucho más volátiles.
- Igualmente las distribuciones de residuales de ambos modelos se desvían de la normalidad, con colas pesadas (más valores extremos que en una distribución normal), **aunque es algo típico en series financieras**.

```
par(mfrow = c(2, 2))
plot(residuales_2410, main = "Residuales ARIMA(24,1,0)")
plot(residuales_717, main = "Residuales ARIMA(7,1,7)")
qqnorm(residuales_2410, main = "QQ-Plot ARIMA(24,1,0)")
qqline(residuales_2410)
qqnorm(residuales_717, main = "QQ-Plot ARIMA(7,1,7)")
qqline(residuales_717)
```




```
par(mfrow = c(1, 1))
```

**Test de Ljung-Box para autocorrelación en residuales**

- ARIMA(24,1,0): p-value = 1
- ARIMA(7,1,7): p-value = 0.588

Ambos p-valores son mayores a 0.05, lo que indica que **no hay autocorrelación significativa en los residuales**. El modelo ARIMA(24,1,0) muestra un mejor comportamiento de ruido blanco.

```
Box.test(residuales_2410, type = "Ljung-Box", lag = 20)
```

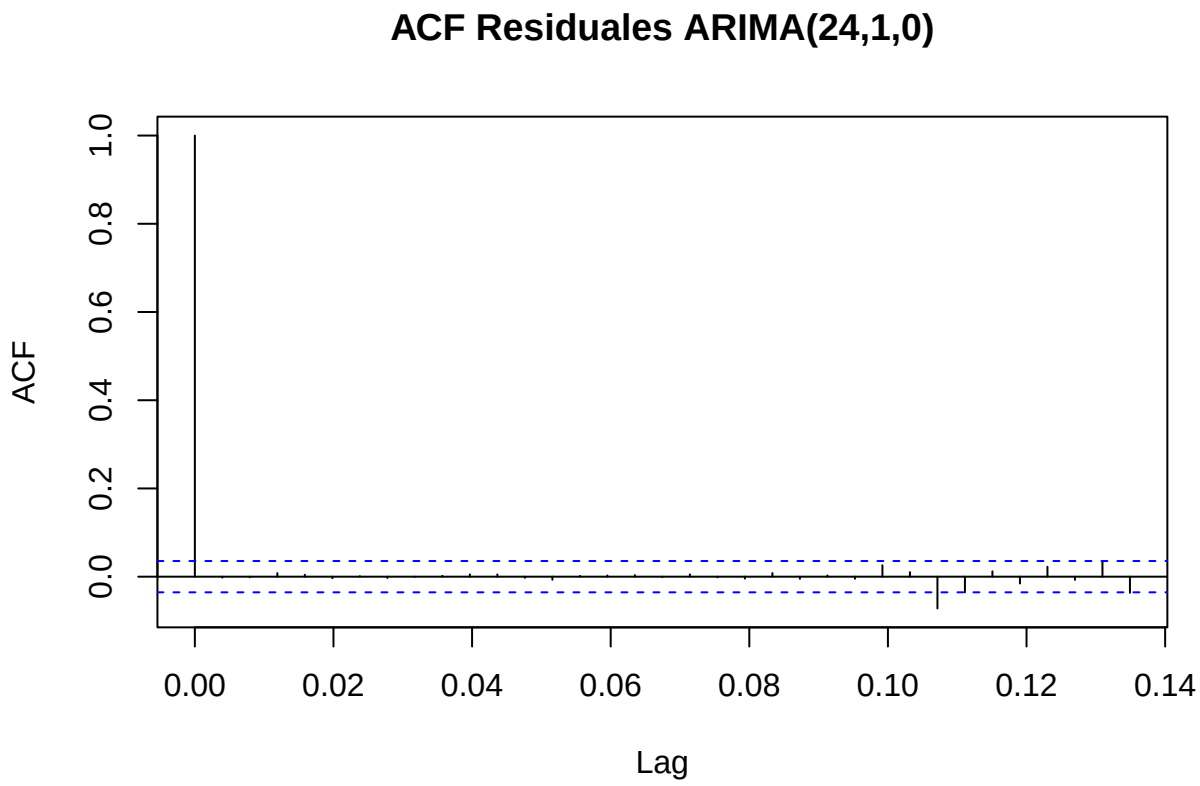
```
##
##  Box-Ljung test
##
## data:  residuales_2410
## X-squared = 0.92334, df = 20, p-value = 1
```

```
Box.test(residuales_717, type = "Ljung-Box", lag = 20)
```

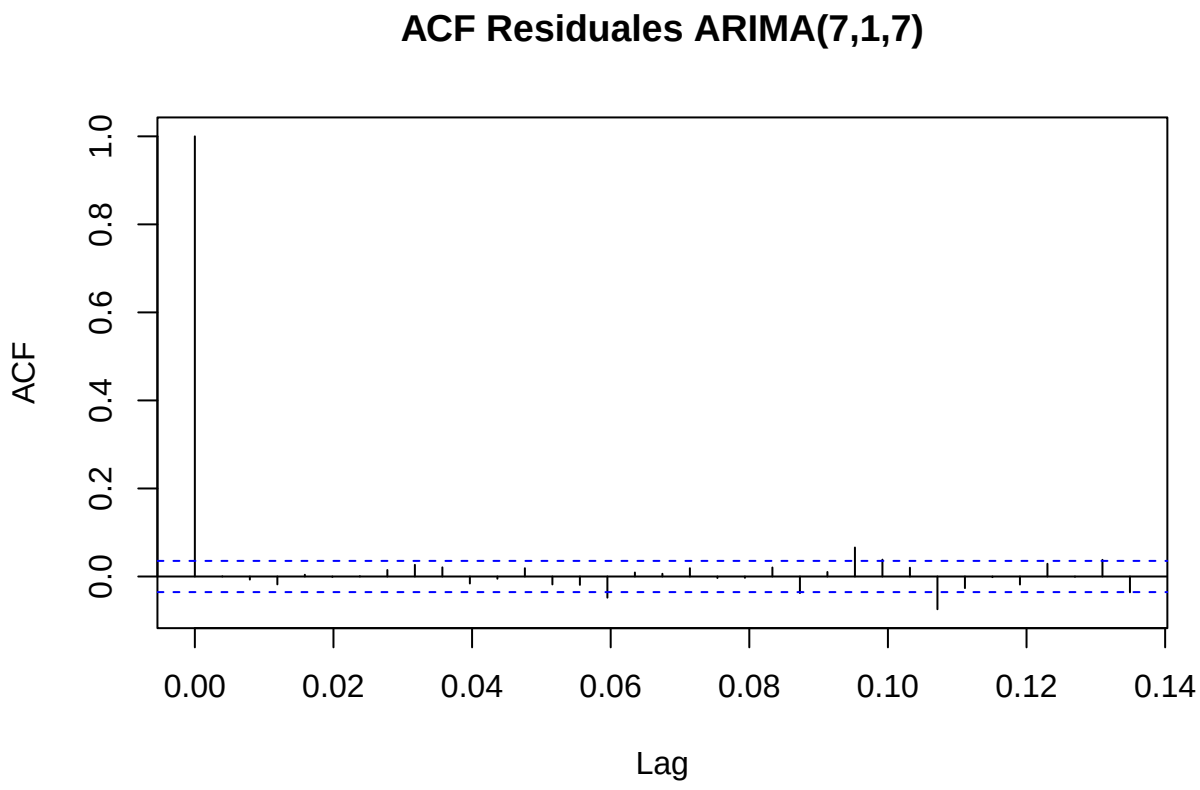
```
##
##  Box-Ljung test
##
## data:  residuales_717
## X-squared = 17.991, df = 20, p-value = 0.588
```

**ACF de residuales** Los correlogramas confirman la ausencia de autocorrelación significativa en ambos modelos. Casi todos los valores están dentro de las bandas de confianza, lo que indica que los modelos han capturado adecuadamente la estructura temporal de la serie.

```
acf(residuales_2410, main = "ACF Residuales ARIMA(24,1,0)")
```




```
acf(residuales_717, main = "ACF Residuales ARIMA(7,1,7)")
```




**Valores AIC, BIC y LogLik**

- ARIMA(24,1,0)
  - AIC: 19427.67
  - BIC: 19578.3
  - LogLik: -9688.835

Este modelo tiene mejor AIC y LogLik, indicando un mejor ajuste a los datos, pero con más parámetros.

```
cat("AIC:", AIC(modelo_2410), "\n")
```

```
## AIC: 19427.67
```

```
cat("BIC:", BIC(modelo_2410), "\n")
```

```
## BIC: 19578.3
```

```
cat("LogLik:", logLik(modelo_2410), "\n\n")
```

```
## LogLik: -9688.835
```

- ARIMA(7,1,7)

- AIC: 19438.24
- BIC: 19528.62
- LogLik: -9704.122

Este modelo tiene mejor BIC, indicando mejor equilibrio entre ajuste y parsimonia (menos parámetros).

```
cat("AIC:", AIC(modelo_717), "\n")
```

```
## AIC: 19438.24
```

```
cat("BIC:", BIC(modelo_717), "\n")
```

```
## BIC: 19528.62
```

```
cat("LogLik:", logLik(modelo_717), "\n\n")
```

```
## LogLik: -9704.122
```

En conjunto, ambos modelos son válidos, con el ARIMA(24,1,0) mostrando un mejor ajuste puro y el ARIMA(7,1,7) ofreciendo una mejor relación entre ajuste y complejidad.

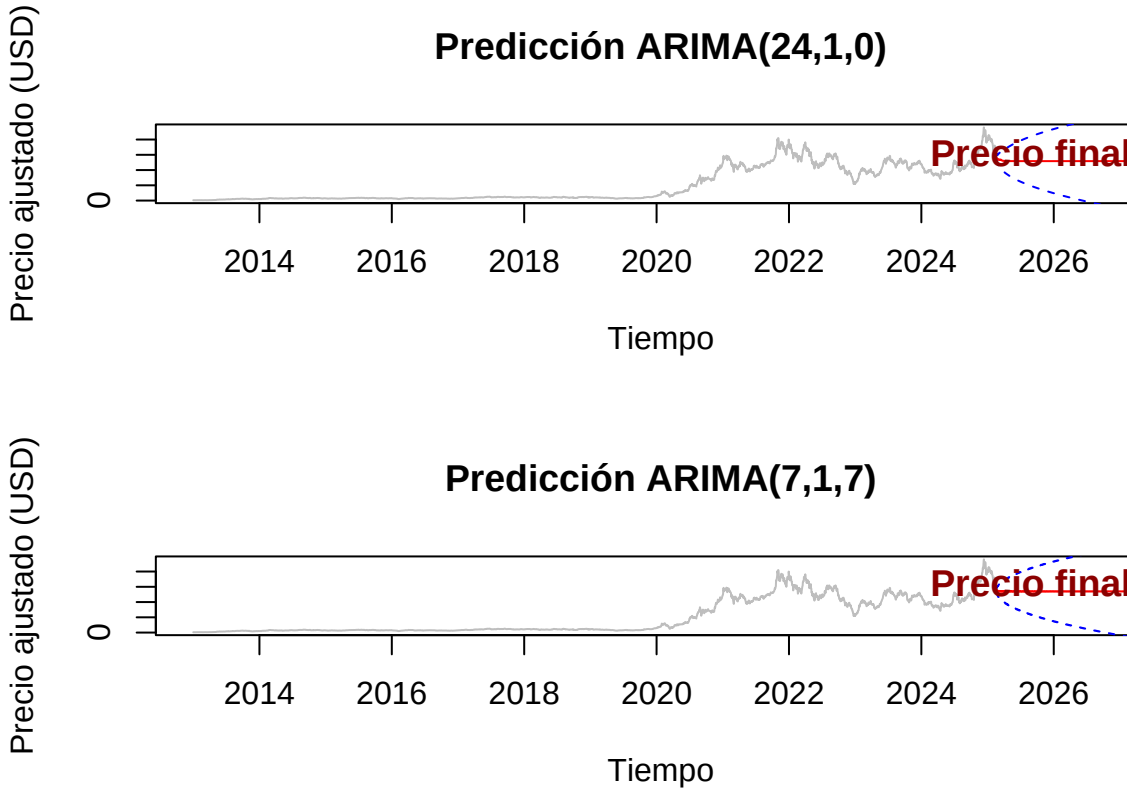
### Paso 5: Proyecciones

*Realiza predicciones para los próximos 2 años (la frecuencia de la serie es diaria) y muestra estas gráficamente. Para esto puedes utilizar el comando 'predict' o 'forecast'.*

```
# Número de días para 2 años (aproximadamente 504 días de trading)
dias_prediccion <- 504

# Realizamos predicciones con ambos modelos
prediccion_2410 <- predict(modelo_2410, n.ahead = dias_prediccion)
prediccion_717 <- predict(modelo_717, n.ahead = dias_prediccion)

# Creamos secuencia de fechas futuras para el eje x
ultima_fecha <- time(tsla_ts)[length(tsla_ts)]
fechas_futuras <- seq(ultima_fecha, by = 1/252, length.out = dias_prediccion)

# Obtenemos los últimos precios predichos
ultimo_precio_2410 <- prediccion_2410$pred[dias_prediccion]
ultimo_precio_717 <- prediccion_717$pred[dias_prediccion]
```

### Predicción a 2 años

```r
par(mfrow = c(2, 1))

# Gráfica para ARIMA(24,1,0)
plot(time(tsla_ts), tsla_ts, type = "l", col = "gray",
     xlim = c(min(time(tsla_ts)), max(fechas_futuras)),
     main = "Predicción ARIMA(24,1,0)",
     xlab = "Tiempo", ylab = "Precio ajustado (USD)")
lines(fechas_futuras, prediccion_2410$pred, col = "red")
lines(fechas_futuras, prediccion_2410$pred + 2*prediccion_2410$se, col = "blue", lty = 2)
lines(fechas_futuras, prediccion_2410$pred - 2*prediccion_2410$se, col = "blue", lty = 2)
# Añadimos texto con el último precio predicho
text(max(fechas_futuras) - 0.2, ultimo_precio_2410 + 50,
     labels = paste("Precio final: $", round(ultimo_precio_2410, 2)),
     col = "darkred", cex = 1.2, font = 2, bg = "white")

# Gráfica para ARIMA(7,1,7)
plot(time(tsla_ts), tsla_ts, type = "l", col = "gray",
     xlim = c(min(time(tsla_ts)), max(fechas_futuras)),
     main = "Predicción ARIMA(7,1,7)",
     xlab = "Tiempo", ylab = "Precio ajustado (USD)")
lines(fechas_futuras, prediccion_717$pred, col = "red")
lines(fechas_futuras, prediccion_717$pred + 2*prediccion_717$se, col = "blue", lty = 2)
lines(fechas_futuras, prediccion_717$pred - 2*prediccion_717$se, col = "blue", lty = 2)
# Añadimos texto con el último precio predicho
text(max(fechas_futuras) - 0.2, ultimo_precio_717 + 50,
     labels = paste("Precio final: $", round(ultimo_precio_717, 2)),
     col = "darkred", cex = 1.2, font = 2, bg = "white")
```




**Visualización a 2 años**

```
par(mfrow = c(1, 1))
```

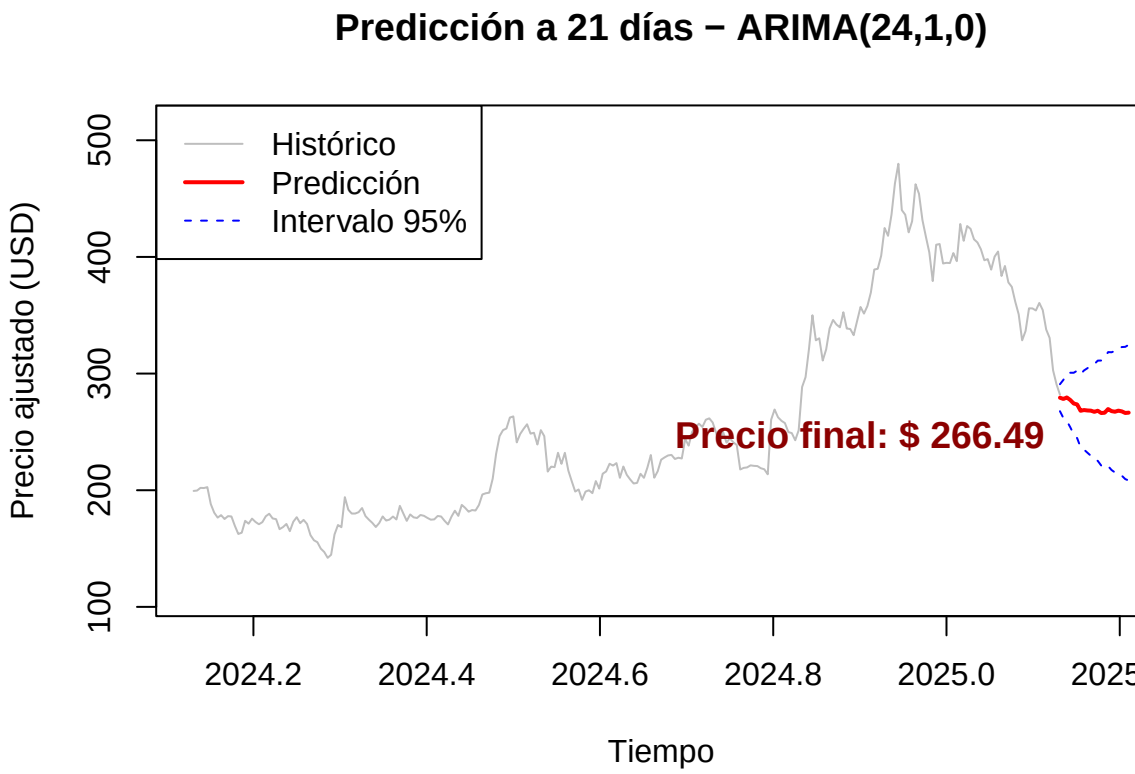
Aún siendo un buen modelo, este tipo de ARIMA no es adecuado para predecir a largo plazo, sería más adecuado utilizar modelos más complejos y considerando variables exógenas. Para visualizarlos mejor, mostramos aquí las predicciones para los próximos 21 días:

```
# Número de días para 1 mes de trading (aproximadamente 21 días)
dias_prediccion <- 21

# Realizamos predicciones con ambos modelos
prediccion_2410 <- predict(modelo_2410, n.ahead = dias_prediccion)
prediccion_717 <- predict(modelo_717, n.ahead = dias_prediccion)

# Preparamos los datos del último año y las fechas
tiempo_ts <- time(tsla_ts)
indice_ultimo_anio <- length(tsla_ts) - 252
datos_ultimo_anio <- window(tsla_ts, start = tiempo_ts[indice_ultimo_anio])

# Creamos secuencia de tiempo para predicciones
ultima_fecha <- tiempo_ts[length(tsla_ts)]
tiempo_prediccion <- seq(ultima_fecha, by = 1/252, length.out = dias_prediccion)
tiempo_ultimo_anio <- tiempo_ts[indice_ultimo_anio:length(tsla_ts)]
```

```r
# Calculamos límites del eje Y
y_min <- min(c(datos_ultimo_anio, prediccion_2410$pred - 2*prediccion_2410$se,
               prediccion_717$pred - 2*prediccion_717$se))
y_max <- max(c(datos_ultimo_anio, prediccion_2410$pred + 2*prediccion_2410$se,
               prediccion_717$pred + 2*prediccion_717$se))
y_margin <- (y_max - y_min) * 0.1

# Obtenemos los últimos precios predichos
ultimo_precio_2410 <- prediccion_2410$pred[dias_prediccion]
ultimo_precio_717 <- prediccion_717$pred[dias_prediccion]

# Visualizamos las predicciones
par(mfrow = c(2, 1), mar = c(5, 4, 4, 2) + 0.1)
```

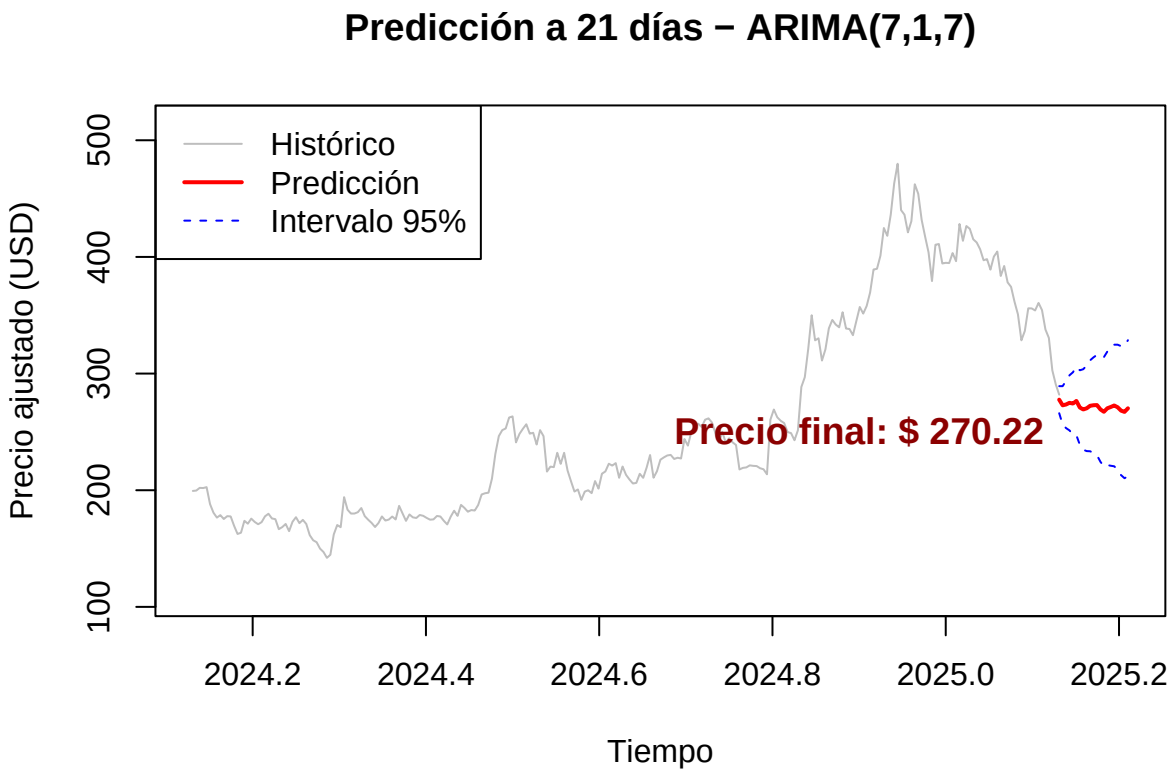
**Predicción a 1 mes**

```r
# Gráfica para ARIMA(24,1,0)
plot(tiempo_ultimo_anio, datos_ultimo_anio, type = "l", col = "gray",
     xlim = c(min(tiempo_ultimo_anio), max(tiempo_prediccion)),
     ylim = c(y_min - y_margin, y_max + y_margin),
     main = "Predicción a 21 días - ARIMA(24,1,0)",
     xlab = "Tiempo", ylab = "Precio ajustado (USD)")
lines(tiempo_prediccion, prediccion_2410$pred, col = "red", lwd = 2)
lines(tiempo_prediccion, prediccion_2410$pred + 2*prediccion_2410$se, col = "blue", lty = 2)
lines(tiempo_prediccion, prediccion_2410$pred - 2*prediccion_2410$se, col = "blue", lty = 2)
legend("topleft",
       legend = c("Histórico", "Predicción", "Intervalo 95%"),
       col = c("gray", "red", "blue"),
       lty = c(1, 1, 2),
       lwd = c(1, 2, 1))
# Añadimos texto con el último precio predicho
text(2024.9, ultimo_precio_2410 - 20,
     labels = paste("Precio final: $", round(ultimo_precio_2410, 2)),
     col = "darkred", cex = 1.2, font = 2, bg = "white")
```




**Visualización a 1 mes**

```
# Gráfica para ARIMA(7,1,7)
plot(tiempo_ultimo_anio, datos_ultimo_anio, type = "l", col = "gray",
     xlim = c(min(tiempo_ultimo_anio), max(tiempo_prediccion)),
     ylim = c(y_min - y_margin, y_max + y_margin),
     main = "Predicción a 21 días - ARIMA(7,1,7)",
     xlab = "Tiempo", ylab = "Precio ajustado (USD)")
lines(tiempo_prediccion, prediccion_717$pred, col = "red", lwd = 2)
lines(tiempo_prediccion, prediccion_717$pred + 2*prediccion_717$se, col = "blue", lty = 2)
lines(tiempo_prediccion, prediccion_717$pred - 2*prediccion_717$se, col = "blue", lty = 2)
legend("topleft",
       legend = c("Histórico", "Predicción", "Intervalo 95%"),
       col = c("gray", "red", "blue"),
       lty = c(1, 1, 2),
       lwd = c(1, 2, 1))
# Añadimos texto con el último precio predicho
text(2024.9, ultimo_precio_717 - 20,
     labels = paste("Precio final: $", round(ultimo_precio_717, 2)),
     col = "darkred", cex = 1.2, font = 2, bg = "white")
```




```
par(mfrow = c(1, 1))
```