CALIDAD DE AIRE

María Paula Camargo Rincón Laura Katherin Martinez Castiblanco
Yudy Vanessa Puerres Rosero

2025-10-20

```
rm(list=ls())
```

librerías

```
#install.packages("readxl")
#install.packages("knitr")
#install.packages("forecast")
#install.packages("FinTS")
#install.packages("dplyr")
#install.packages("ggplot2")
#install.packages("lubridate")
#install.packages("zoo")
#install.packages("nortest")
library(readxl) # Para leer el csv
library(knitr) # Para obtener la tabla en el pdf a partir de una tabla en r
library(forecast) # para el lambda de box-cox
```

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

```
library(tseries) # prueba de estacionariedad adf
library(FinTS)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

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

```
##
## Attaching package: 'FinTS'
```

```
## The following object is masked from 'package:forecast':
##
##     Acf
```

```
library(dplyr) # para manejo de datos
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2) # para gráficas
library(lubridate) # fechas
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(zoo) #imputación de datos faltantes
library(nortest) # para prueba kolmogorov
```

Descripción de los datos:

Base de datos

La base de datos utilizada proviene del conjunto de datos público disponible en la plataforma Kaggle (Rohan Rao, 2020). Este conjunto fue recopilado originalmente por el Central Pollution Control Board (CPCB), organismo oficial del Gobierno de la India encargado del monitoreo ambiental.

Para este estudio, se utilizó la versión diaria del conjunto de datos, enfocándose exclusivamente en la variable PM2.5 ($\mu g/m^3$) (partículas en suspensión con diámetro aerodinámico ≤ 2.5 µm) registrada en la ciudad de Delhi. La serie abarca el período comprendido entre el 1 de enero de 2015 y el 6 de diciembre de 2020, con observaciones diarias.

```
aire <- read_excel("Datos_India.xlsx")
knitr::kable(head(aire,10),
             caption = "Primeros 10 datos",
             digits = 4)  # redondea a 4 decimales
```

Table 1: Primeros 10 datos

City	Date	PM2.5	AQI_Bucket
Delhi	2015-01-01	313.22	Severe
Delhi	2015-02-01	186.18	Severe
Delhi	2015-03-01	87.18	Moderate
Delhi	2015-04-01	151.84	Very Poor
Delhi	2015-05-01	146.60	Very Poor
Delhi	2015-06-01	149.58	Very Poor
Delhi	2015-07-01	217.87	Very Poor
Delhi	2015-08-01	229.90	Very Poor
Delhi	2015-09-01	201.66	Very Poor
Delhi	2015-10-01	221.02	Very Poor

Además de PM2.5, el conjunto incluye el Índice de Calidad del Aire (AQI) y su clasificación categórica (*AQI_Bucket*), que divide la calidad del aire en seis niveles: *Good*, *Satisfactory*, *Moderate*, *Poor*, *Very Poor* y

Severe. Sin embargo, dado que el objetivo del análisis es modelar la dinámica temporal de la concentración de PM2.5, se trabajó únicamente con la variable numérica continua.

```
# Convert 'City' and 'AQI_Bucket' to factor variables
aire$City <- as.factor(aire$City)
aire$AQI_Bucket <- as.factor(aire$AQI_Bucket)

# Convert 'Date' to a date variable, specifying the format
aire$Date <- as.Date(aire$Date, format = "%d/%m/%Y")

summary(aire)
```

```
##     City           Date                PM2.5               AQI_Bucket
##  Delhi:2009   Min.   :2015-01-01   Min.   : 10.24   Good        : 21
##               1st Qu.:2016-05-17   1st Qu.: 57.09   Moderate    :519
##               Median :2017-10-01   Median : 94.62   Poor        :542
##               Mean   :2017-10-04   Mean   :117.20   Satisfactory:158
##               3rd Qu.:2019-02-15   3rd Qu.:153.03   Severe      :239
##               Max.   :2020-12-06   Max.   :685.36   Very Poor   :520
##                                    NA's   :2        NA's        : 10
```

```
str(aire)
```

```
## tibble [2,009 x 4] (S3: tbl_df/tbl/data.frame)
##  $ City      : Factor w/ 1 level "Delhi": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Date      : Date[1:2009], format: "2015-01-01" "2015-02-01" ...
##  $ PM2.5     : num [1:2009] 313.2 186.2 87.2 151.8 146.6 ...
##  $ AQI_Bucket: Factor w/ 6 levels "Good","Moderate",..: 5 5 2 6 6 6 6 6 6 6 ...
```

Evaluación de base de datos

```
# Group by Date and count the number of entries
daily_counts <- aire %>%
  group_by(Date) %>%
  summarise(count = n())

# Filter for dates with more than one entry
dates_with_duplicates <- daily_counts %>%
  filter(count > 1)


# Display the dates with duplicate entries
if (nrow(dates_with_duplicates) > 0) {
  cat("Días con mas de un valor de  PM2.5:\n")
  print(dates_with_duplicates)
} else {
  cat("No hay días con mas de un valor de PM2.5\n")
}
```

```
## No hay días con mas de un valor de PM2.5
```

El conjunto inicial contiene 2,009 observaciones. Se identificaron 2 valores faltantes en la variable `PM2.5` y 10 en `AQI_Bucket`. Dado que el AQI no se utilizará en el modelado, se centró la atención en imputar los valores ausentes de PM2.5. Se aplicó interpolación lineal mediante la función `na.approx()` del paquete `zoo`, asumiendo que la evolución de la contaminación en el corto plazo no es drástica.

```r
aire[!complete.cases(aire), ]
```

```
## # A tibble: 11 x 4
##    City  Date       PM2.5 AQI_Bucket
##    <fct> <date>     <dbl> <fct>
##  1 Delhi 2016-07-24  59.4 <NA>
##  2 Delhi 2017-06-23  44.1 <NA>
##  3 Delhi 2017-12-08  NA   Good
##  4 Delhi 2017-08-13  NA   <NA>
##  5 Delhi 2017-08-14  26.5 <NA>
##  6 Delhi 2017-08-22  46   <NA>
##  7 Delhi 2017-08-23  36.5 <NA>
##  8 Delhi 2017-08-26  62.3 <NA>
##  9 Delhi 2017-08-27  34.3 <NA>
## 10 Delhi 2017-08-28  23.8 <NA>
## 11 Delhi 2017-08-29  15.2 <NA>
```

```r
# Usar interpolación lineal para rellenar valores faltantes en 'PM2.5'
# Podemos usar la función na.approx del paquete zoo para la interpolación lineal.
aire$`PM2.5` <- na.approx(aire$`PM2.5`)

# Verificar que no haya más valores faltantes en 'PM2.5'
missing_pm25_after_imputation <- sum(is.na(aire$`PM2.5`))
cat("Número de datos faltantes en 'PM2.5' después de la imputación:", missing_pm25_after_imputation, "\n
```

```
## Número de datos faltantes en 'PM2.5' después de la imputación: 0
```

Las estadísticas resumen de la serie de PM2.5 revelan una distribución altamente asimétrica hacia la derecha, con una media de 117.10 µg/m³, una mediana de 94.49 µg/m³ y un máximo extremo de 685.36 µg/m³.

```r
# Estadísticas descriptivas
summary_stats <- aire %>%
  summarise(
    Media = mean(PM2.5, na.rm = TRUE),
    Mediana = median(PM2.5, na.rm = TRUE),
    Desv_Est = sd(PM2.5, na.rm = TRUE),
    Min = min(PM2.5, na.rm = TRUE),
    Max = max(PM2.5, na.rm = TRUE)
  )

kable(summary_stats, digits = 3, caption = "Estadísticas descriptivas de PM2.5 ")
```

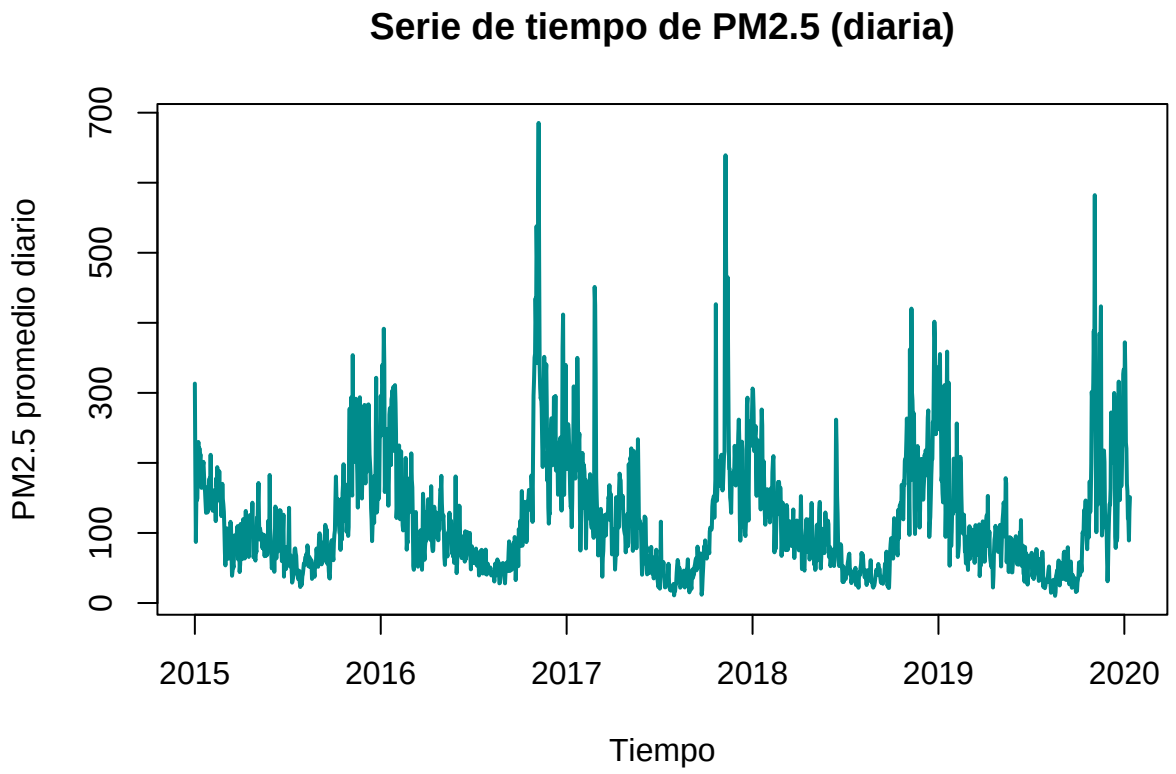
Table 2: Estadísticas descriptivas de PM2.5

Media	Mediana	Desv_Est	Min	Max
117.104	94.49	82.924	10.24	685.36

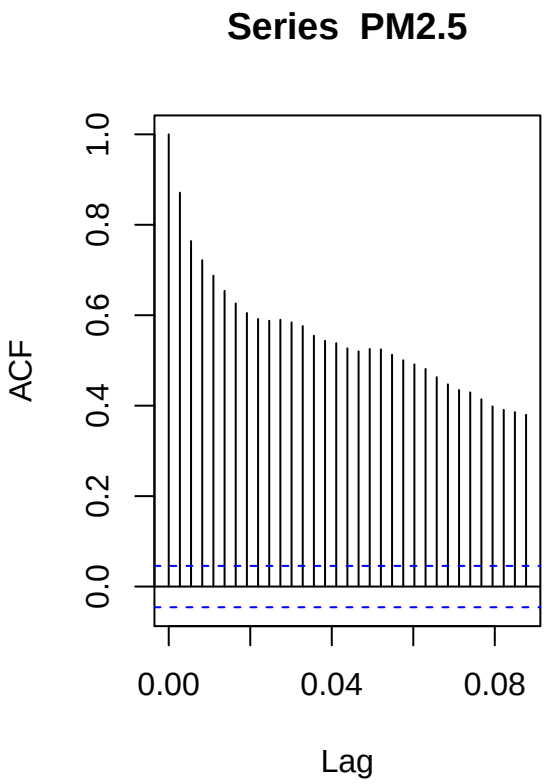
La gráfica de la serie de tiempo muestra una variabilidad, con estacionalidad clara (patrones repetidos, que parecieran ser anuales). Aunque no se observa una tendencia, la varianza no es constante puesto que los picos son más pronunciados en ciertos períodos del año, lo que sugiere heterocedasticidad.

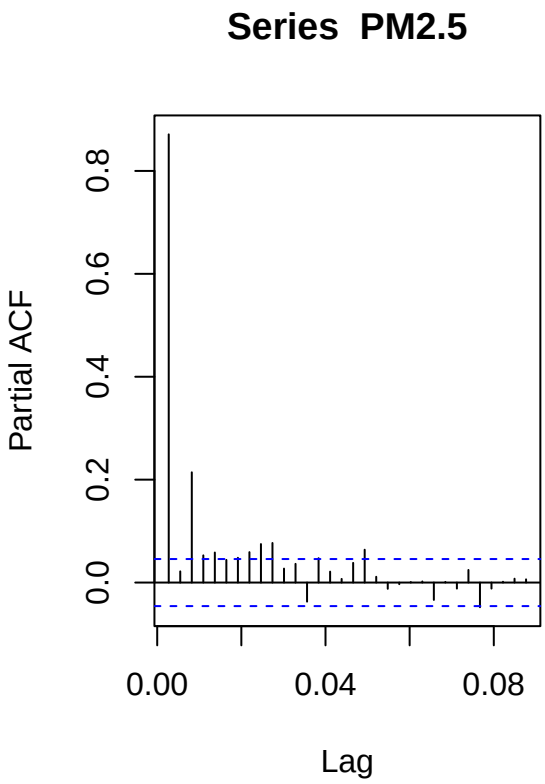
```r
# Creación de la serie
start_date <- min(aire$Date)
end_date <- max(aire$Date)
PM2.5 <- ts(aire$`PM2.5`, start = c(year(start_date), month(start_date), day(start_date)), end = c(year
```

```r
# Gráfica
ts.plot(PM2.5, col = "darkcyan", lwd = 2,
        ylab = "PM2.5 promedio diario", xlab = "Tiempo",
        main = "Serie de tiempo de PM2.5 (diaria)")
```




```r
par(mfrow=c(1,2))
acf(PM2.5) # decaimiento lento indica que la serie no es estacionaria
pacf(PM2.5)
```





Para evaluar formalmente la estacionariedad en media (la cual segun gráficos parece ser estacionaria), se usa la prueba de Dickey-Fuller (ADF). El estadístico resultante fue -4.7414 con un p-valor < 0.01, lo que permite rechazar la hipótesis nula de raíz unitaria. Por lo tanto, la serie se considera estacionaria.

```
adf_result <- adf.test(PM2.5, alternative = "stationary")
```

```
## Warning in adf.test(PM2.5, alternative = "stationary"): p-value smaller than
## printed p-value
```

```
cat("Prueba ADF:\n Estadístico =", round(adf_result$statistic, 4),
    ", p-valor =", adf_result$p.value, "\n")
```

```
## Prueba ADF:
##  Estadístico = -4.7414 , p-valor = 0.01
```

Dada la heterocedasticidad, se evaluo una transformación para estabilizar la varianza. Se estimó el parámetro de Box-Cox, obteniéndo $\lambda \approx 0.257$, cercano a cero. Por lo cual se aplicó la transformación logarítmica.

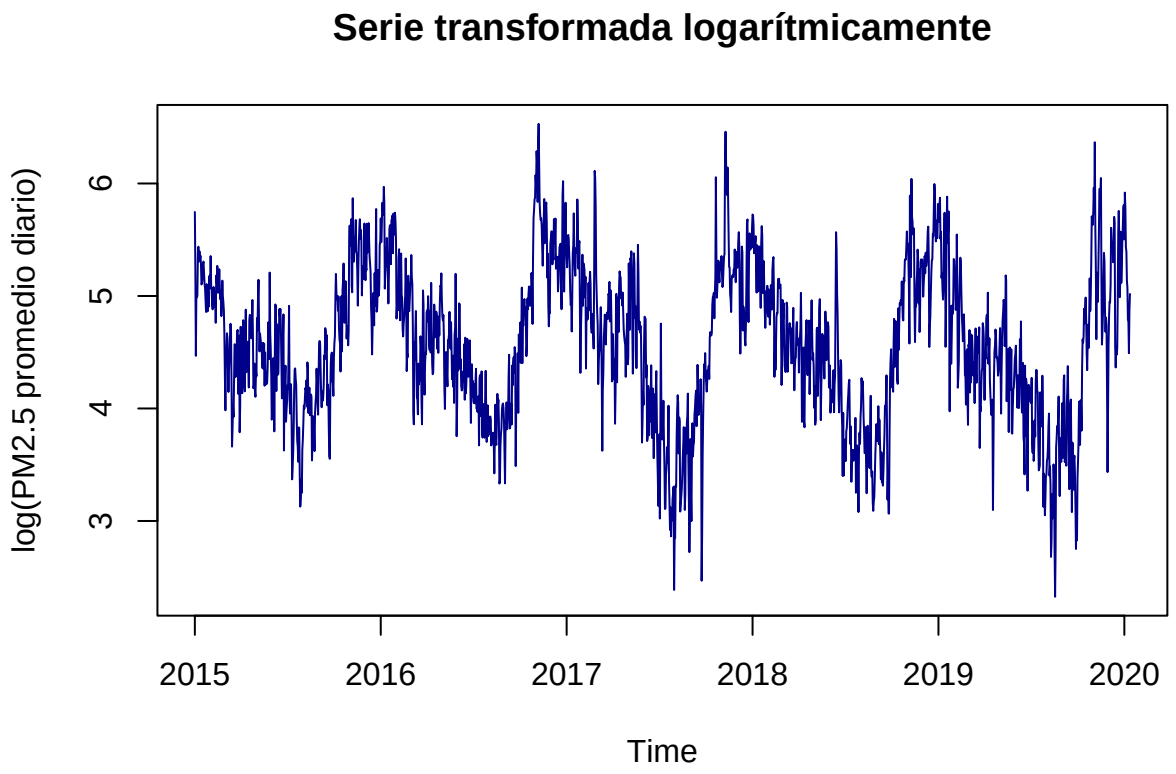
```
lambda <- BoxCox.lambda(PM2.5)
lambda
```

```
## [1] 0.2571413
```

Dada la heterocedasticidad, se evaluo una transformación para estabilizar la varianza. Se estimó el parámetro de Box-Cox, obteniéndo $\lambda = 0.257$ (cercana a 0). Por lo cual se aplicó la transformación logarítmica.

```
# Transformación logarítmica
l.PM2.5 <- log(PM2.5)
ts.plot(l.PM2.5, col = "darkblue",
        ylab = "log(PM2.5 promedio diario)",
```

```
        main = "Serie transformada logarítmicamente")
```



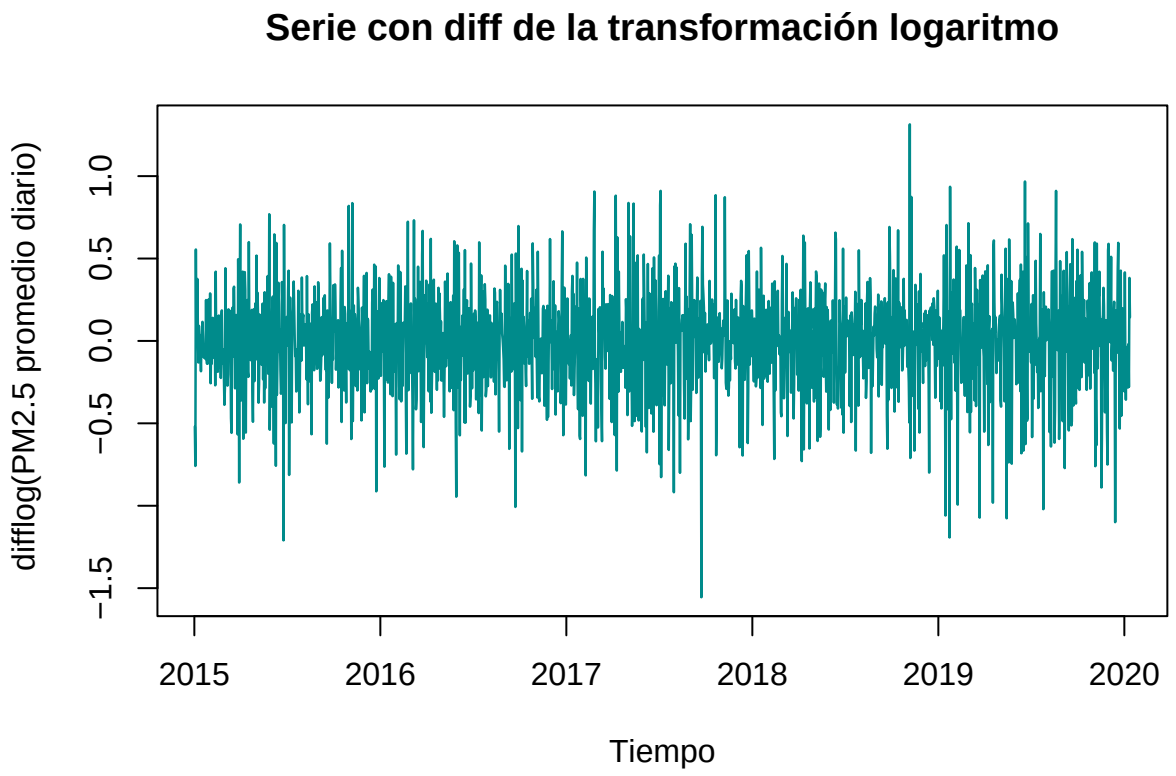
La serie transformada presenta una varianza más homogénea y mantiene la estacionariedad en media (prueba ADF con p-valor ≈ 0.027), cumpliendo así los supuestos de estacionariedad para el ajuste de modelos ARIMA.

```
adf.test(l.PM2.5,alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  l.PM2.5
## Dickey-Fuller = -3.6549, Lag order = 12, p-value = 0.02739
## alternative hypothesis: stationary
```

Luego de aplicar la transformación logarítmica a la serie de PM2.5, el test de Dickey–Fuller Aumentado (ADF) arrojó un estadístico de -3.65 con un p-valor de 0.027, lo que indicaría estacionariedad a un nivel de significancia del 5%. Sin embargo, la inspección de la función de autocorrelación (ACF) revela un decaimiento lento, señal de persistencia temporal y posible estacionalidad anual. Por tanto, la serie log-transformada aún no puede considerarse estrictamente estacionaria, y se recomienda aplicar una diferenciación regular (d = 1)

```
d.l.PM2.5 <- diff(l.PM2.5) # Retorno de PM2.5
ts.plot(d.l.PM2.5, col = "darkcyan", lwd = 1.25,
        ylab = " difflog(PM2.5 promedio diario)", xlab = "Tiempo",
        main = "Serie con diff de la transformación logaritmo")
```

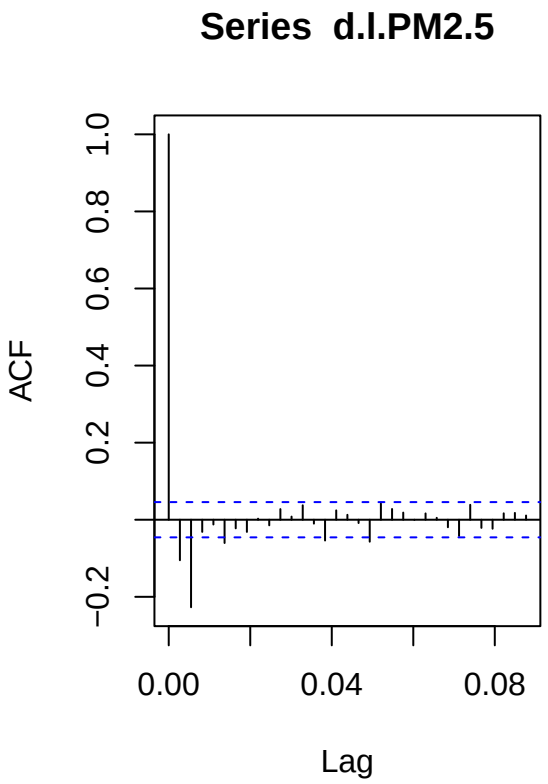


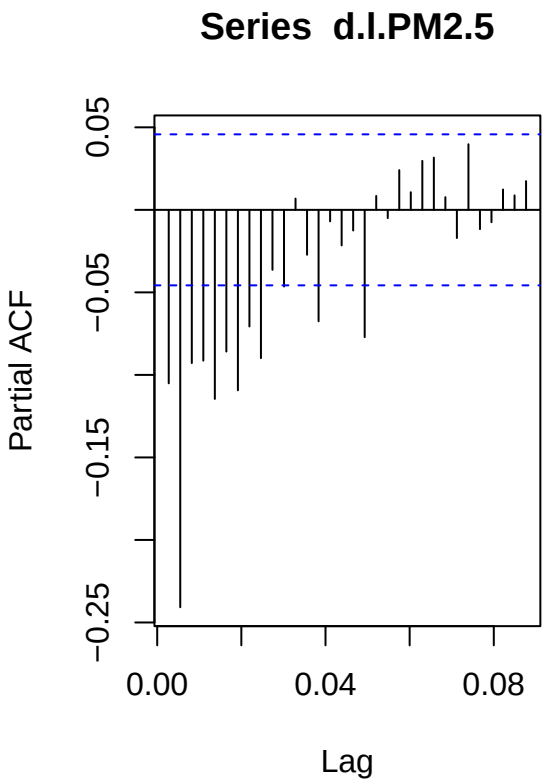

```
adf.test(d.l.PM2.5)
```

```
## Warning in adf.test(d.l.PM2.5): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  d.l.PM2.5
## Dickey-Fuller = -16.31, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

```
par(mfrow=c(1,2))
acf(d.l.PM2.5)
pacf(d.l.PM2.5)
```





Luego de aplicar la transformación logarítmica, se realizó una diferenciación de primer orden (difflog(PM2.5)) con el fin de eliminar la tendencia presente en la serie. La inspección de la serie transformada muestra oscilaciones alrededor de una media constante, sin tendencias visibles. El test de Dickey–Fuller Aumentado (ADF) arrojó un estadístico de -16.31 con un p-valor < 0.01, indicando que la serie diferenciada es estacionaria. Asimismo, la función de autocorrelación (ACF) se corta rápidamente tras el primer rezago, confirmando la ausencia de dependencia de largo plazo. En consecuencia, la serie transformada y diferenciada cumple los supuestos de estacionariedad necesarios para proceder con la identificación de un modelo ARIMA.

Luego de aplicar la transformación logarítmica, se realizó una diferenciación de primer orden ($\Delta \log(\text{PM2.5})$) con el fin de eliminar la tendencia presente en la serie. La inspección de la serie transformada muestra oscilaciones alrededor de una media constante, sin tendencias visibles. El test de Dickey–Fuller Aumentado (ADF) arrojó un estadístico de -16.31 con un p-valor < 0.01, indicando que la serie diferenciada es estacionaria. Asimismo, la función de autocorrelación (ACF) se corta rápidamente tras el primer rezago, confirmando la ausencia de dependencia de largo plazo. En consecuencia, la serie transformada y diferenciada cumple los supuestos de estacionariedad necesarios para proceder con la identificación de un modelo ARIMA.

Identificación de modelos ARIMA

Se presentan estos 2 modelos el primero para la serie transformada con log dado que, según el PACF (Función de Autocorrelación Parcial) de esta, se observa que el último rezago significativo es 2 (lo que sugiere un AR(1) o AR(2)). Por otro lado, con el ACF (Función de Autocorrelación) de la serie se observa un decaimiento lento, por lo que no se puede definir con certeza cuál sería la estructura de MA (Media Móvil). Por ello, se prueba el ARMA(2,2) (sugerido por la función `auto.arima(log(PM2.5))`), el otro modelo es un modelo ARIMA(1,1,2) (a su vez sugerido por `auto.arima($\triangle$log(PM2.5))`). Adicinalmente se prueba un modelo ARIMA(2,1,2) puesto que se puede comparar la serie sin diferenciar propuesta con la misma pero diferenciandola.

```r
auto.arima(l.PM2.5, seasonal = FALSE)
```

```
## Series: l.PM2.5
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1      ma2    mean
##       1.3806  -0.3876  -0.5955  -0.2087  4.5934
## s.e.  0.0569   0.0557   0.0584   0.0368  0.1806
##
## sigma^2 = 0.08243:  log likelihood = -312.7
## AIC=637.39   AICc=637.44   BIC=670.49
```

```r
arma2.2 <- arima(x = l.PM2.5,order = c(2,0,2))


auto.arima(d.l.PM2.5, seasonal = FALSE)
```

```
## Series: d.l.PM2.5
## ARIMA(1,0,2) with zero mean
##
## Coefficients:
##          ar1      ma1      ma2
##       0.4065  -0.6157  -0.2071
## s.e.  0.0532   0.0552   0.0371
##
## sigma^2 = 0.08278:  log likelihood = -316.68
## AIC=641.36   AICc=641.39   BIC=663.43
```

```r
arima1.1.2 <- arima(x = l.PM2.5,order = c(1,1,2))

arima2.1.2 <- arima(x = l.PM2.5, order = c(2, 1, 2))
```

Dado que no se asume normalidad de los residuos, los criterios de información (AIC, BIC) (basados en la verosimilitud) no son tan confiables. Por ello, comparamos los modelos con medidas del error en la muestra de entrenamiento, que no dependen de supuestos distribucionales.

```r
# Obtener métricas de error en la muestra de entrenamiento
acc_arma2.2    <- accuracy(arma2.2)
acc_arima2.1.2 <- accuracy(arima2.1.2)
acc_arima2.1.2 <- accuracy(arima2.1.2)

# Crear tabla comparativa
comparacion <- data.frame(
  Modelo = c("ARMA(2,2)", "ARIMA(1,1,2)", "ARIMA(2,1,2)"),
  RMSE = c(acc_arma2.2[1, "RMSE"],
           acc_arima2.1.2[1, "RMSE"],
           acc_arima2.1.2[1, "RMSE"]),
  MAE  = c(acc_arma2.2[1, "MAE"],
           acc_arima2.1.2[1, "MAE"],
           acc_arima2.1.2[1, "MAE"]),
  MAPE = c(acc_arma2.2[1, "MAPE"],
           acc_arima2.1.2[1, "MAPE"],
           acc_arima2.1.2[1, "MAPE"])
)
```

```r
# Mostrar tabla
knitr::kable(comparacion,
             caption = "Comparación de métricas de error en la muestra de entrenamiento",
             digits = 4,
             row.names = FALSE)
```

Table 3: Comparación de métricas de error en la muestra de entrenamiento

Modelo	RMSE	MAE	MAPE
ARMA(2,2)	0.2867	0.2195	5.0131
ARIMA(1,1,2)	0.2872	0.2198	5.0197
ARIMA(2,1,2)	0.2872	0.2198	5.0197

Se observa que el ARMA(2,2) es el modelo que tiene el mejor ajuste en terminos del error, seguido por el modelo ARIMA (2,1,2) a diferencia de lo sugerido por la función `auto.arima()` , por lo que se prefiere trabajar con estos dos modelos en adelante.

```r
# Nombres de parámetros (8 filas)
parametros <- c("Intercepto", "ar1", "ar2", "ma1", "ma2", "logLik", "AIC", "BIC")

# --- Modelo ARMA(2,2) ---
coef_arma <- coef(arma2.2)  # Tiene: ar1, ar2, ma1, ma2, intercept
# Aseguramos el orden: intercept, ar1, ar2, ma1, ma2
coef_arma_ordenado <- c(
  coef_arma["intercept"],
  coef_arma["ar1"],
  coef_arma["ar2"],
  coef_arma["ma1"],
  coef_arma["ma2"]
)
t_arma_ordenado <- coef_arma_ordenado / c(
  sqrt(arma2.2$var.coef["intercept", "intercept"]),
  sqrt(arma2.2$var.coef["ar1", "ar1"]),
  sqrt(arma2.2$var.coef["ar2", "ar2"]),
  sqrt(arma2.2$var.coef["ma1", "ma1"]),
  sqrt(arma2.2$var.coef["ma2", "ma2"])
)

# --- Modelo ARIMA(2,1,2) ---
# Este modelo NO tiene intercepto (porque d = 1), pero SÍ tiene ar1 y ar2
coef_arima212 <- coef(arima2.1.2)  # Debe tener: ar1, ar2, ma1, ma2
coef_arima212_ordenado <- c(
  NA,  # Intercepto
  coef_arima212["ar1"],
  coef_arima212["ar2"],
  coef_arima212["ma1"],
  coef_arima212["ma2"]
)

# Valores t para ARIMA(2,1,2)
t_arima212_ordenado <- c(
```

```
  NA,
  coef_arima212["ar1"] / sqrt(arima2.1.2$var.coef["ar1", "ar1"]),
  coef_arima212["ar2"] / sqrt(arima2.1.2$var.coef["ar2", "ar2"]),
  coef_arima212["ma1"] / sqrt(arima2.1.2$var.coef["ma1", "ma1"]),
  coef_arima212["ma2"] / sqrt(arima2.1.2$var.coef["ma2", "ma2"])
)

# --- Construir tabla ---
resumen <- data.frame(
  "Parámetro" = parametros,

  "ARMA(2,2)" = c(coef_arma_ordenado, arma2.2$loglik, arma2.2$aic, BIC(arma2.2)),
  "t-valor ARMA(2,2)" = c(t_arma_ordenado, NA, NA, NA),

  "ARIMA(2,1,2)" = c(coef_arima212_ordenado, arima2.1.2$loglik, arima2.1.2$aic, BIC(arima2.1.2)),
  "t-valor ARIMA(2,1,2)" = c(t_arima212_ordenado, NA, NA, NA),
  stringsAsFactors = FALSE
)

knitr::kable(
  resumen,
  caption = "Tabla de resumen de los modelos ARMA(2,2) y ARIMA(2,1,2)",
  digits = 3,
  row.names = FALSE
)
```

Table 4: Tabla de resumen de los modelos ARMA(2,2) y ARIMA(2,1,2)

Parámetro	ARMA.2.2.	t.valor.ARMA.2.2.	ARIMA.2.1.2.	t.valor.ARIMA.2.1.2.
Intercepto	4.593	25.438	NA	NA
ar1	1.381	24.282	0.216	1.866
ar2	-0.388	-6.963	0.134	1.671
ma1	-0.595	-10.203	-0.428	-3.849
ma2	-0.209	-5.675	-0.381	-3.794
logLik	-312.696	NA	-315.449	NA
AIC	637.392	NA	640.897	NA
BIC	670.488	NA	668.474	NA

Se presentan a continuación las formulas de los modelos con los coeficientes estimados de la serie transformada $\log(\text{PM}_{2.5})$, denotada como Z_t:.

- *Formula ARMA(2,2)*

$$\begin{aligned} Z_t &= \mu + \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + a_t + \theta_1 a_{t-1} + \theta_2 a_{t-2} \\ &= 4.593 + 1.38 Z_{t-1} - 0.388 Z_{t-2} + a_t + \theta_1 a_{t-1} + \theta_2 a_{t-2} \end{aligned}$$

- Formula ARIMA(2,1,2)

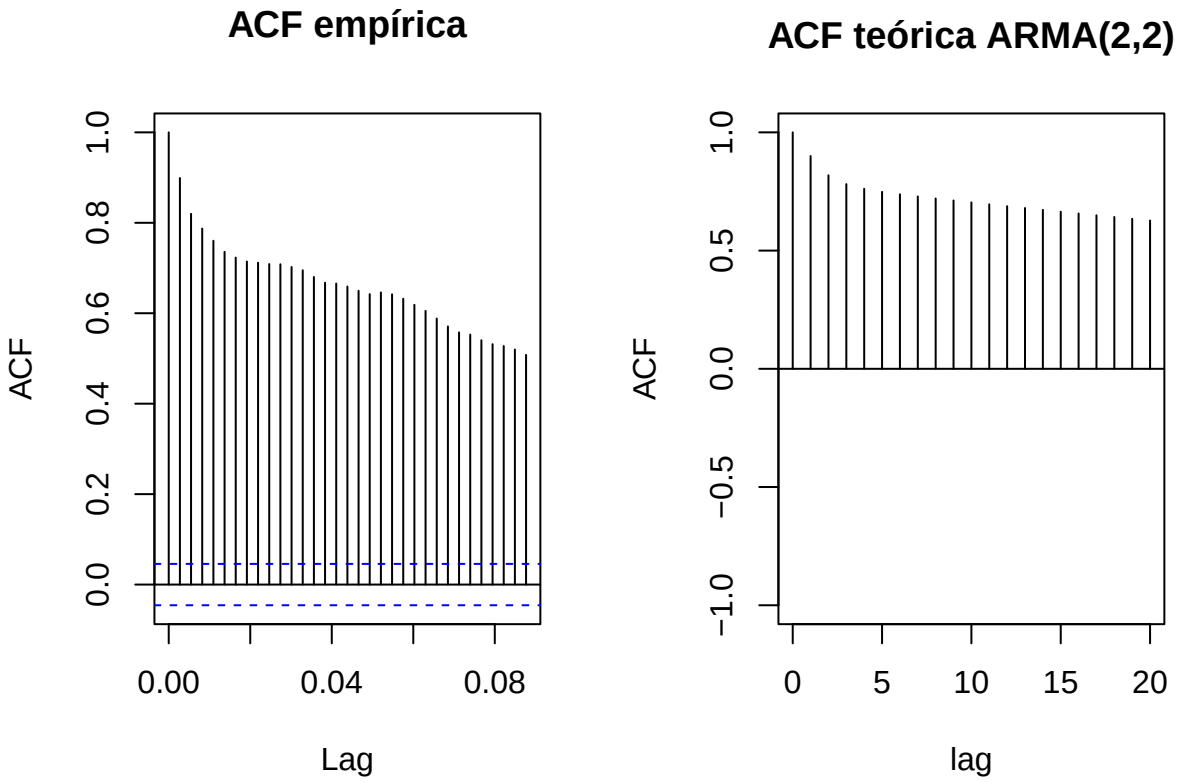
$$\begin{aligned} \nabla Z_t &= \phi_1 \nabla Z_{t-1} + \phi_2 \nabla Z_{t-2} + a_t + \theta_1 a_{t-1} + \theta_2 a_{t-2} \\ &= 0.216 \nabla Z_{t-1} + 0.134 \nabla Z_{t-2} + a_t - 0.428 a_{t-1} - 0.381 a_{t-2} \end{aligned}$$

Para el modelo ARIMA(2,1,2) parece “reducirce” a un modelo ARIMA(1,1,2)

Validación modelos

ARMA(2,2)

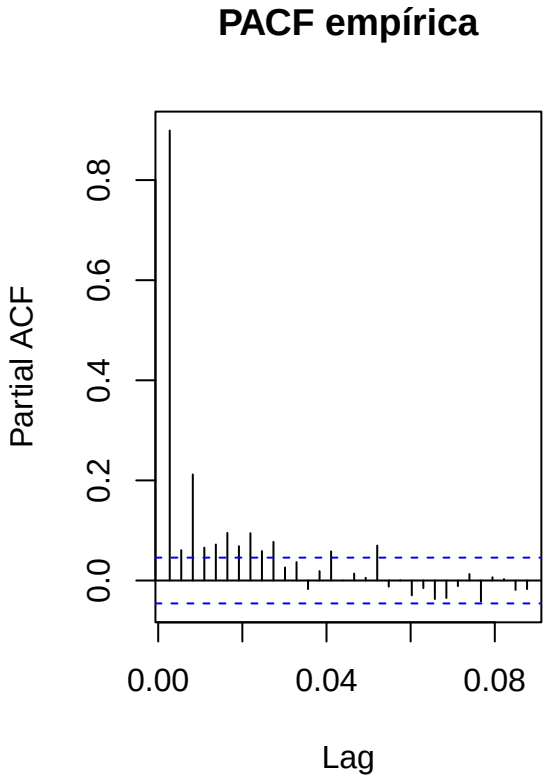
```
arma22_ar <- arma2.2$coef[grep("^ar", names(arma2.2$coef))]  # c(ar1, ar2)
arma22_ma <- arma2.2$coef[grep("^ma", names(arma2.2$coef))]  # c(ma1, ma2)

par(mfrow = c(1, 2))
# ACF
acf(l.PM2.5, main = "ACF empírica")
FinTS::plotArmaTrueacf(list(ar = arma22_ar, ma = arma22_ma),
                       main = "ACF teórica ARMA(2,2)")
```

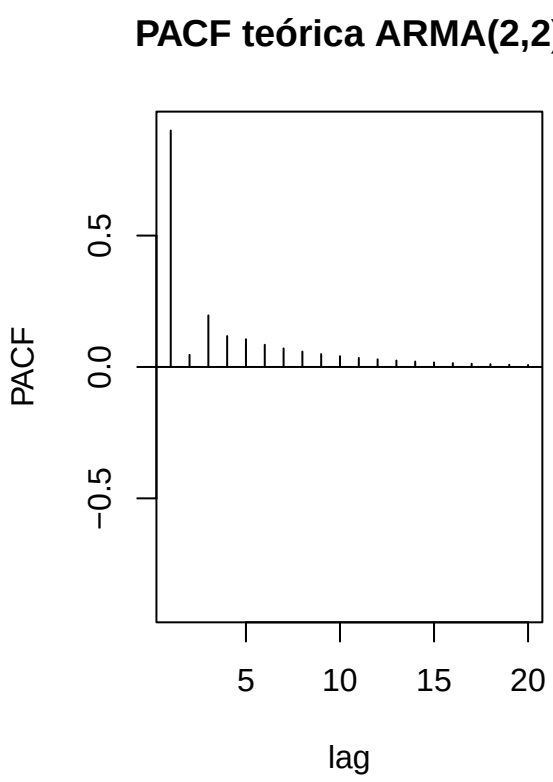



```
## $roots
## [1] 0.9885595 0.3920393
##
## $acf
##         0         1         2         3         4         5         6         7
## 1.0000000 0.8998446 0.8184919 0.7812703 0.7614109 0.7484184 0.7381776 0.7290744
##         8         9        10        11        12        13        14        15
## 0.7204754 0.7121316 0.7039448 0.6958758 0.6879085 0.6800360 0.6722551 0.6645638
##        16        17        18        19        20
## 0.6569607 0.6494447 0.6420146 0.6346696 0.6274087
##
## $periodicity
## [1] damping period
```

```
## <0 rows> (or 0-length row.names)
```

```
# PACF
pacf(l.PM2.5, main = "PACF empírica")
FinTS::plotArmaTrueacf(list(ar = arma22_ar, ma = arma22_ma),
                       pacf = TRUE,
                       main = "PACF teórica ARMA(2,2)")
```

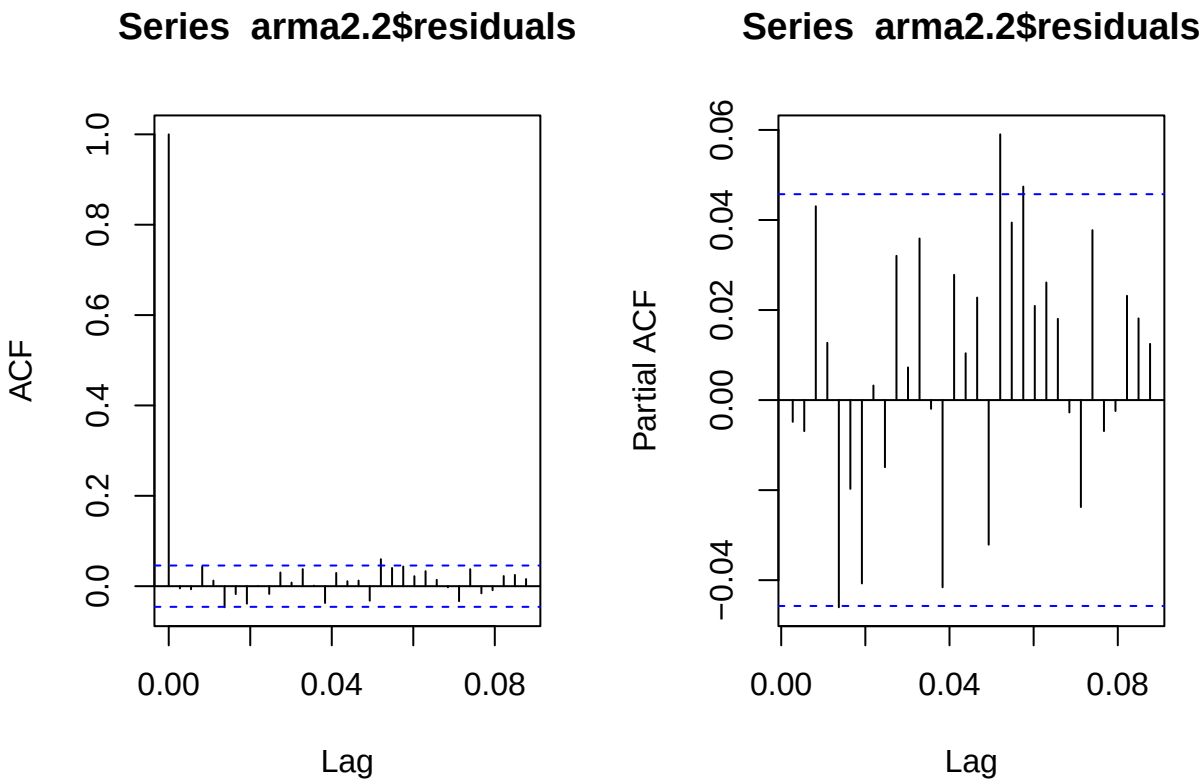





```
## $roots
## [1] 0.9885595 0.3920393
##
## $pacf
##  [1] 0.899844647 0.046097943 0.196053289 0.117481048 0.105533025 0.084416464
##  [7] 0.070596759 0.058672944 0.049094577 0.041138570 0.034540359 0.029032584
## [13] 0.024424277 0.020559553 0.017313712 0.014584670 0.012288405 0.010355236
## [19] 0.008727122 0.007355551
##
## $periodicity
## [1] damping period
## <0 rows> (or 0-length row.names)
```

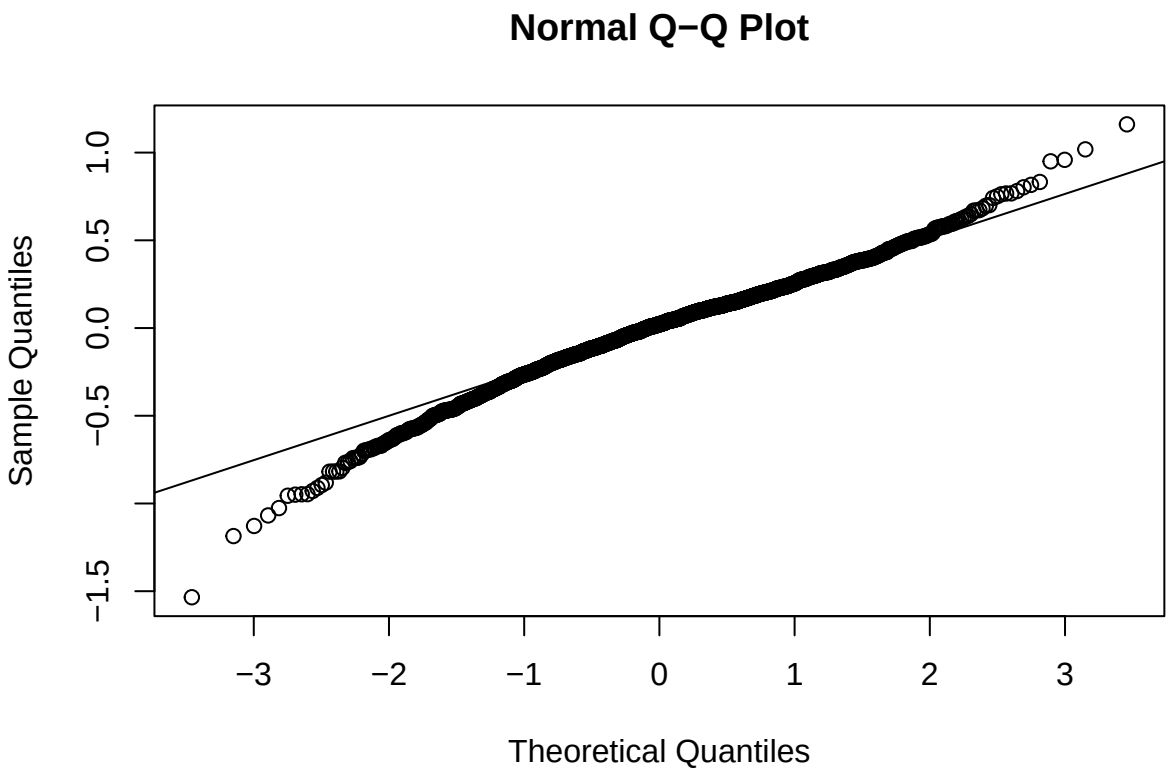
La serie logaritmica parece tener un comportamiento similar a un modelo ARMA(2,2), lo que nos puede indicar que el modelo se puede ajustar bien a los datos.

```
par(mfrow=c(1,2))
acf(arma2.2$residuals)

pacf(arma2.2$residuals)
```



Los residuales del modelo ARMA(2,2) parecen comportarse como un ruido blanco, aunque se ve un rezago en el PACF, este se encuentra muy lejos de los primeros, por lo que no se toma tanta importancia

```
qqnorm(arma2.2$residuals)
qqline(arma2.2$residuals)
```

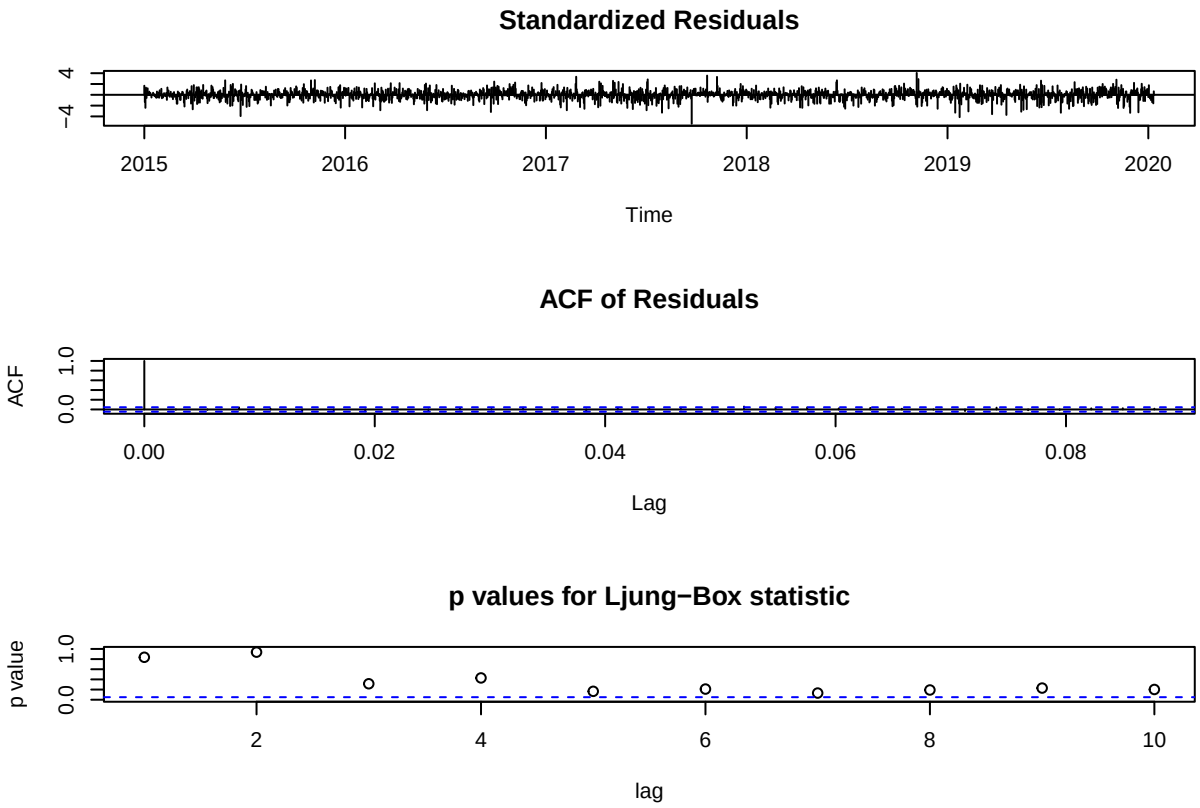


Los residuales aunque en su mayoría se encuentran alrededor de la línea, parece ser que se alejan mucho en las colas, lo que podría indicar no normalidad.

```
Box.test(arma2.2$residuals)
```

```
##
##  Box-Pierce test
##
## data:  arma2.2$residuals
## X-squared = 0.042943, df = 1, p-value = 0.8358
```

```
tsdiag(arma2.2)
```



La prueba de Ljung box muestras que los residuales son independientes ya que su p-valor aproximadamente de 0.84, mayor al nivel de significanvia 5%, no se rechaza la hipotesis nula la cual indica independencia.

```
ks.test(arma2.2$residuals, "pnorm", mean = 0, sd = sd(arma2.2$residuals))
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  arma2.2$residuals
## D = 0.03942, p-value = 0.006631
## alternative hypothesis: two-sided
```

```
shapiro.test(arma2.2$residuals)
```

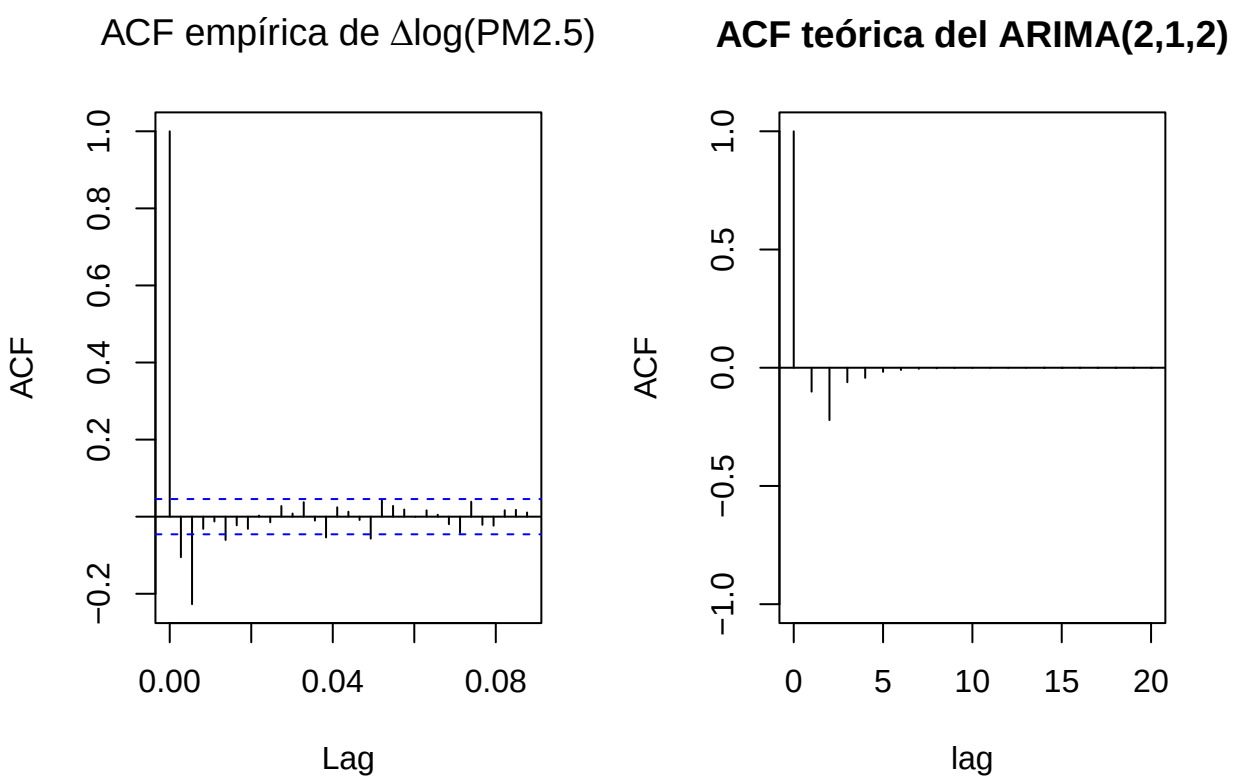
```
##
##  Shapiro-Wilk normality test
##
## data:  arma2.2$residuals
## W = 0.98539, p-value = 9.803e-13
```

Respecto a la normalidad, el p-valor fue menor al nivel de significancia por lo que se rechaza la hipotesis nula de que los residuales provienen de una distribución normal.

ARIMA(2,1,2)

```
# Extraer coeficientes del modelo ARIMA(2,1,2)
arima212_ar <- arima2.1.2$coef[grep("^ar", names(arima2.1.2$coef))]
arima212_ma <- arima2.1.2$coef[grep("^ma", names(arima2.1.2$coef))]
```

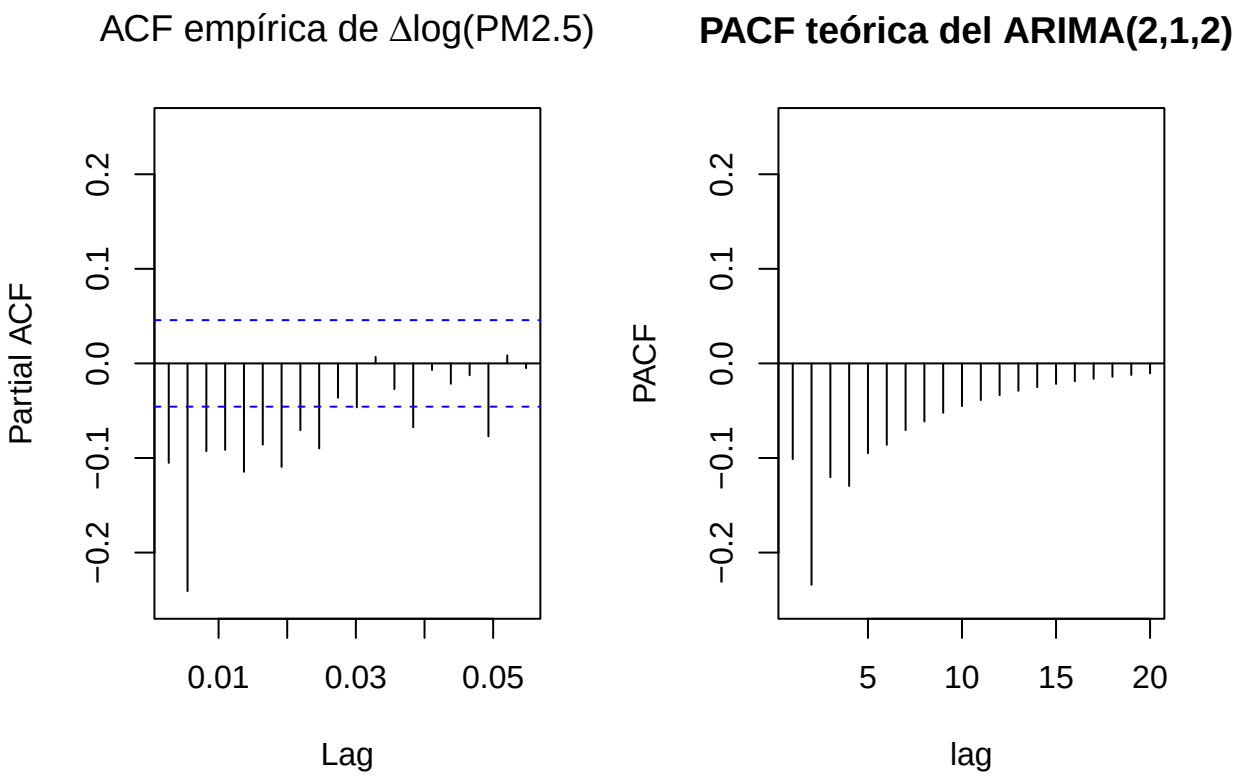
```r
# Calcular límites comunes para Y (basados en la PACF empírica)
pacf_empirica <- pacf(d.l.PM2.5, lag.max = 20, plot = FALSE)

# Gráficos con escalas sincronizadas
par(mfrow = c(1, 2))

# ACF empírica vs teórica (sobre la serie diferenciada)
acf(d.l.PM2.5, main = expression("ACF empírica de " * Delta * "log(PM2.5)"))
FinTS::plotArmaTrueacf(
  list(ar = arima212_ar, ma = arima212_ma),
  main = "ACF teórica del ARIMA(2,1,2)"
)
```




```
## $roots
## [1] -0.2733016  0.4889055
##
## $acf
##             0             1             2             3             4
##  1.000000e+00 -1.011840e-01 -2.214311e-01 -6.126147e-02 -4.279553e-02
##             5             6             7             8             9
## -1.741256e-02 -9.472496e-03 -4.368950e-03 -2.207665e-03 -1.059754e-03
##            10            11            12            13            14
## -5.234723e-04 -2.544656e-04 -1.248094e-04 -6.091075e-05 -2.980946e-05
##            15            16            17            18            19
## -1.456585e-05 -7.123554e-06 -3.482135e-06 -1.702601e-06 -8.323656e-07
##            20
```

```
## -4.069606e-07
##
## $periodicity
## [1] damping period
## <0 rows> (or 0-length row.names)
```

```
# PACF empírica vs teórica
pacf(d.l.PM2.5, lag.max = 20, main = expression("ACF empírica de " * Delta * "log(PM2.5)"), ylim = rang

FinTS::plotArmaTrueacf(
  list(ar = arima212_ar, ma = arima212_ma),
  pacf = TRUE,
  main = "PACF teórica del ARIMA(2,1,2) ",
  lag.max = 20,
  ylim = range(c(pacf_empirica$acf, -0.25, 0.25))  # ajusta según necesidad
)
```

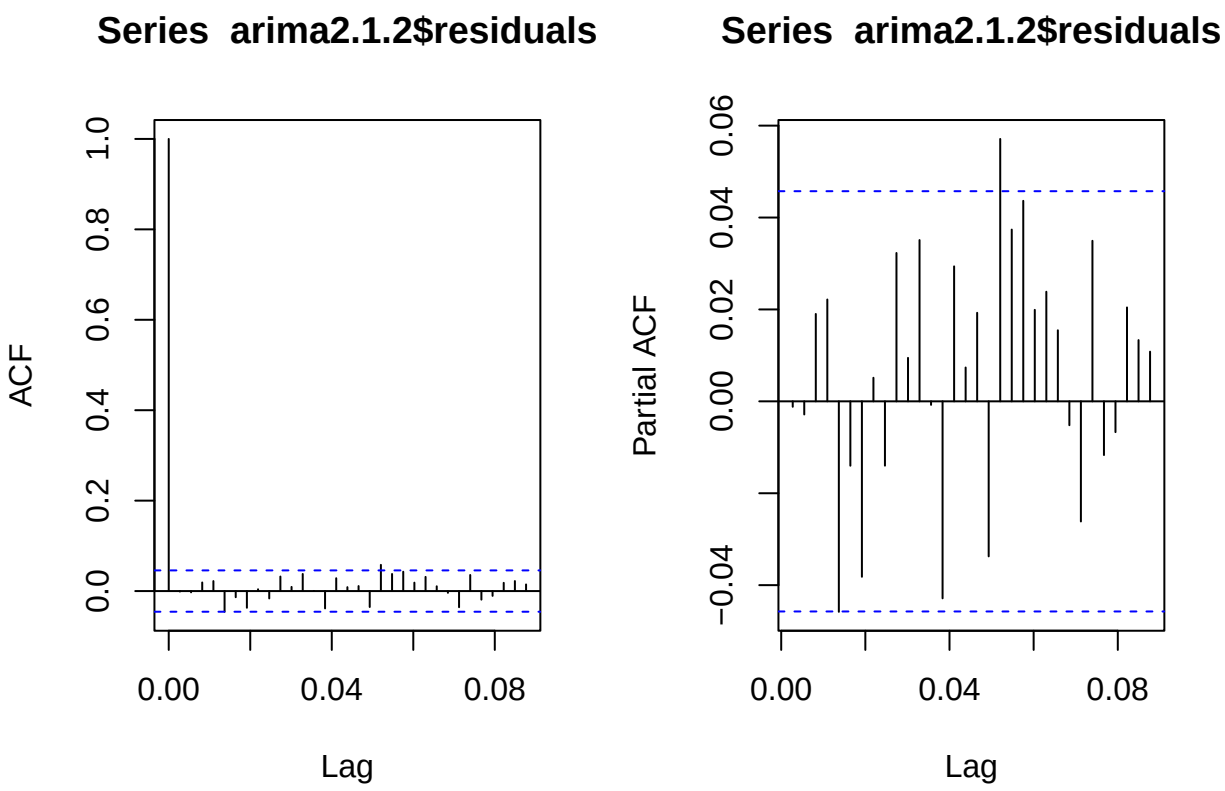



```
## $roots
## [1] -0.2733016  0.4889055
##
## $pacf
##  [1] -0.10118398 -0.23406568 -0.12035320 -0.12961799 -0.09489846 -0.08605478
##  [7] -0.07055201 -0.06144724 -0.05220309 -0.04512688 -0.03879773 -0.03353432
## [13] -0.02896123 -0.02505753 -0.02168273 -0.01877608 -0.01626270 -0.01409042
## [19] -0.01221031 -0.01058283
##
## $periodicity
```

```
## [1] damping period
## <0 rows> (or 0-length row.names)
```

La serie logaritmica diferenciada parece tener un comportamiento similar al de un modelo ARIMA(2,1,2), por lo que este podría ser un indicio de que el modelo se puede ajustar bien a los datos.

```
par(mfrow=c(1,2))
acf(arima2.1.2$residuals)
pacf(arima2.1.2$residuals)
```

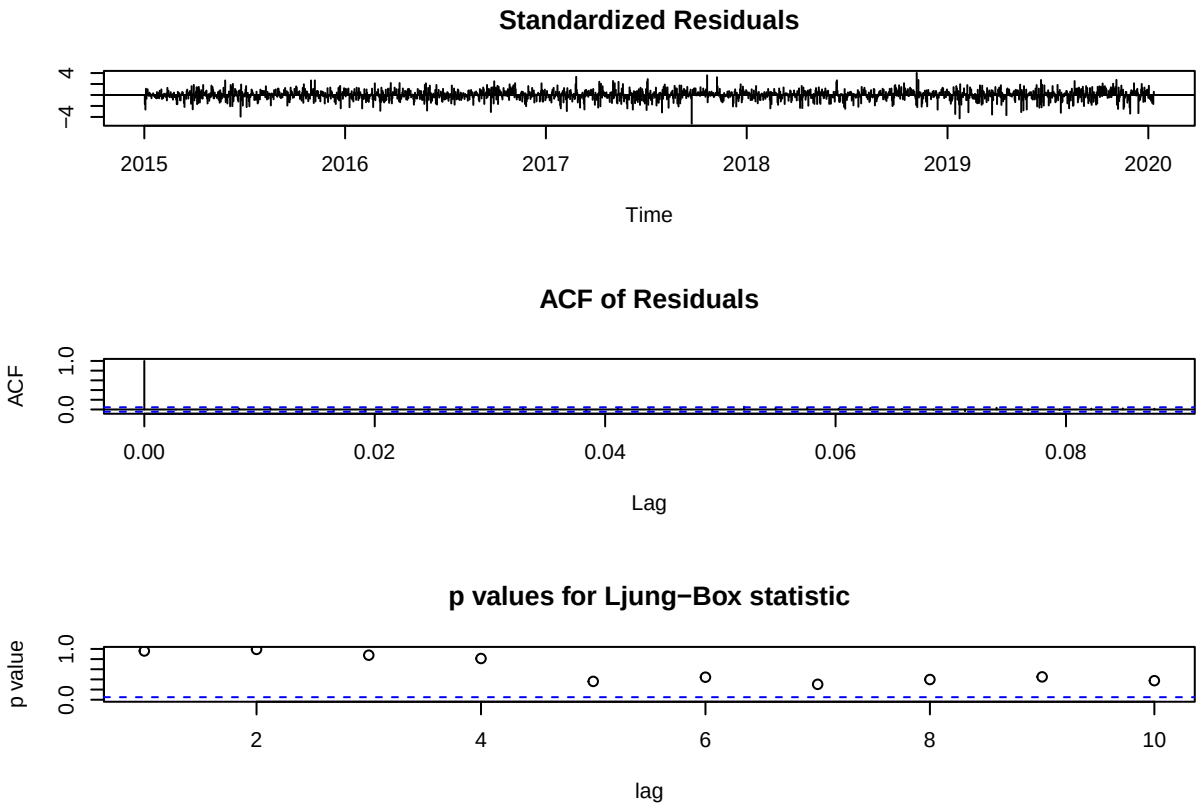


Los residuales del modelo ARIMA(2,1,2), al igual que con el anterior, parecen comportarse como un ruido blanco, e incluso tambien presentan un rezago en el PACF, con el mismo comportamiento del anterior, al ser tan lejano de los primeros valores, no se lo toma importancia.

```
Box.test(arima2.1.2$residuals)
```

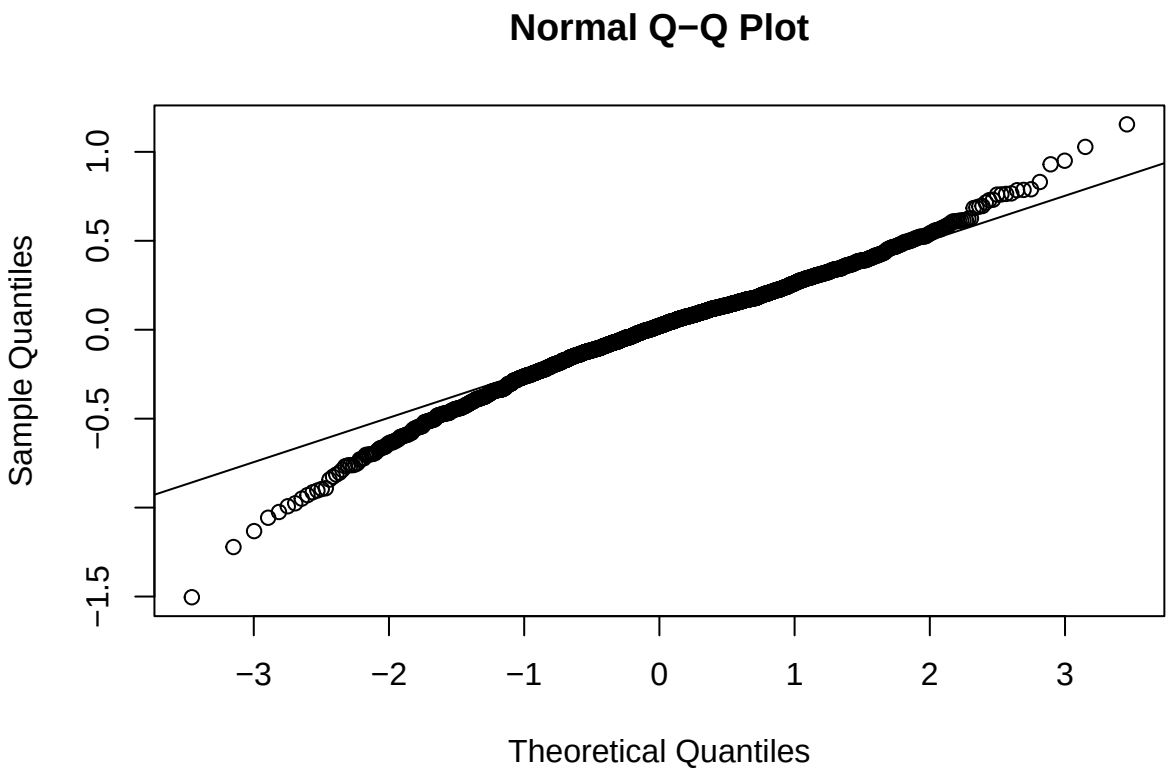
```
##
##  Box-Pierce test
##
## data:  arima2.1.2$residuals
## X-squared = 0.0025837, df = 1, p-value = 0.9595
```

```
tsdiag(arima2.1.2)
```



La prueba de Ljung box muestras que los residuales son independientes ya que su p-valor aproximadamente de 0.95, mayor al nivel de significanvia 5%, por lo que no se rechaza la hipotesis nula la cual indica independencia.

```
qqnorm(arima2.1.2$residuals)
qqline(arima2.1.2$residuals)
```



Al gual que con el modelo ARMA(2,2) parece haber problemas de normalidad ya que los datos suelen alejarse de la línea en las colas.

```
ks.test(arma2.2$residuals, "pnorm", mean = 0, sd = sd(arima2.1.2$residuals))
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  arma2.2$residuals
## D = 0.039498, p-value = 0.006483
## alternative hypothesis: two-sided
```

```
shapiro.test(arima2.1.2$residuals)
```

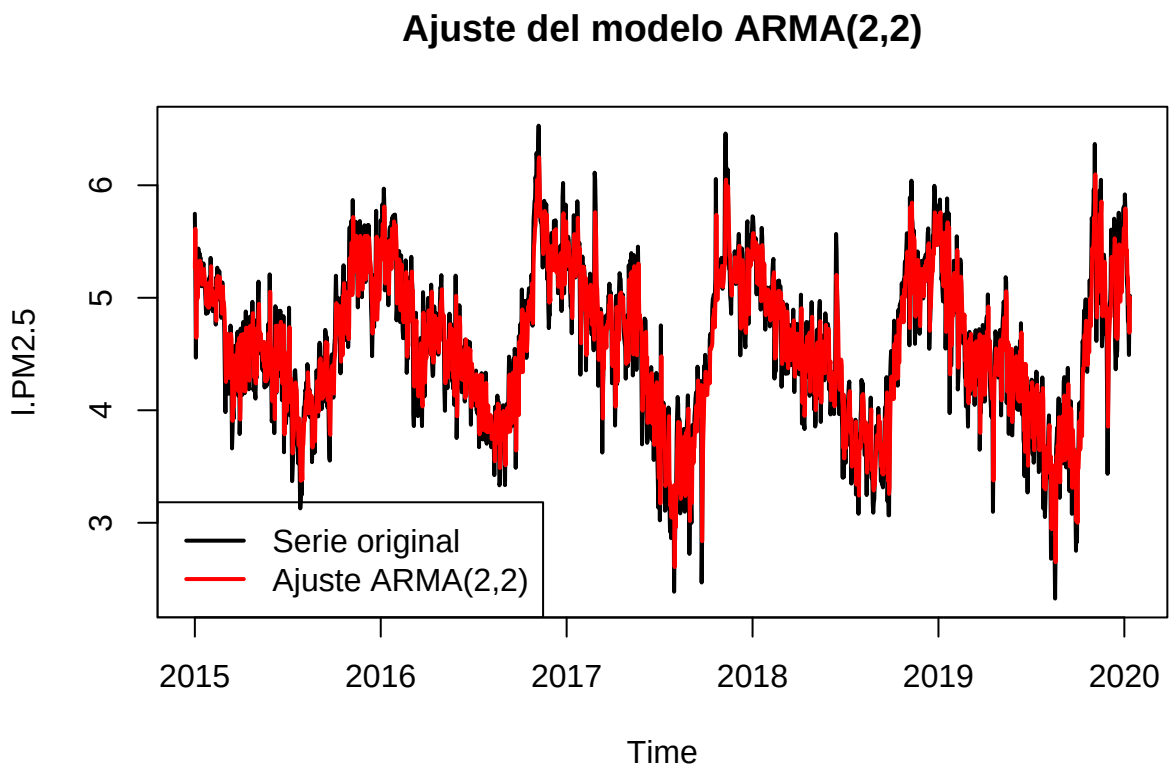
```
##
##  Shapiro-Wilk normality test
##
## data:  arima2.1.2$residuals
## W = 0.9858, p-value = 1.652e-12
```

Efectivaente, al realizar la prueba de normaildad el p-valor fue menor al nivel de significancia por lo que se rechaza la hipotesis nula de que los residuales provienen de una distribución normal.

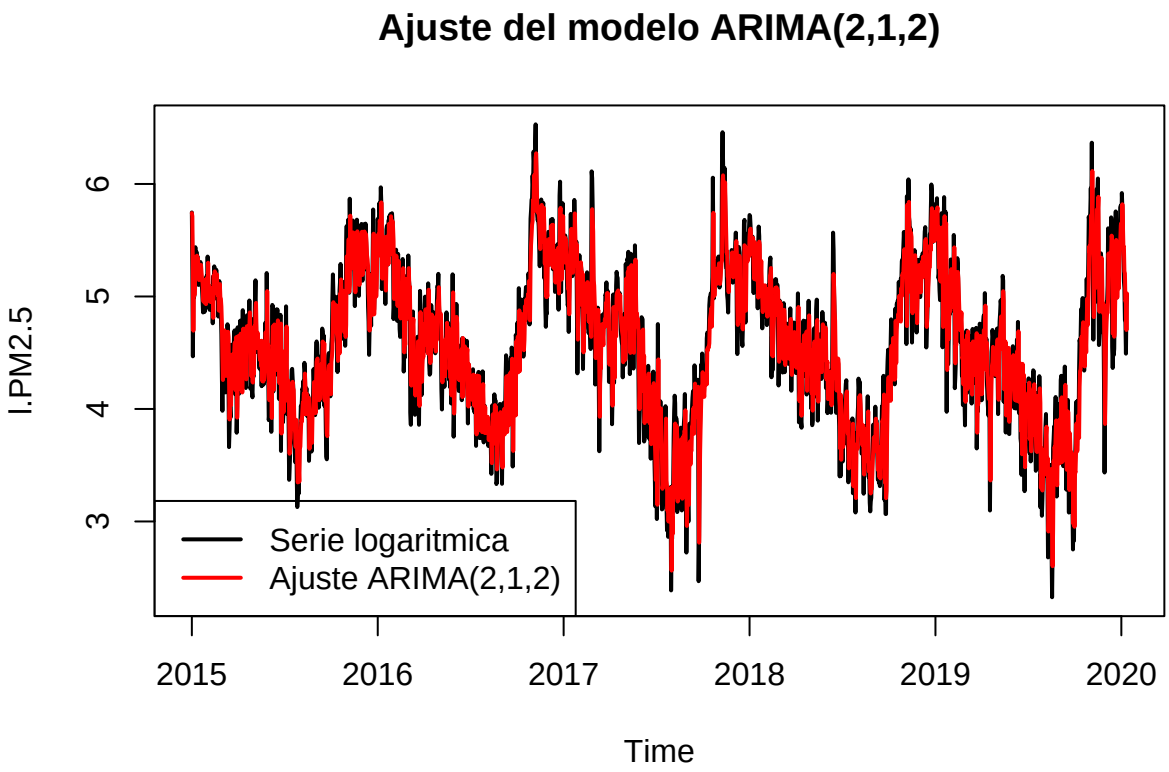
Gráficas de los modelos y la serie

```
plot(l.PM2.5, type = "l", col = "black", lwd = 2, main = "Ajuste del modelo ARMA(2,2)")
lines(fitted(arma2.2), col = "red", lwd = 2)
legend("bottomleft", legend = c("Serie original", "Ajuste ARMA(2,2)"),
```

```
       col = c("black", "red"), lwd = 2)
```




```
plot(l.PM2.5, type = "l", col = "black", lwd = 2, main = "Ajuste del modelo ARIMA(2,1,2)")
lines(fitted(arima2.1.2), col = "red", lwd = 2)
legend("bottomleft", legend = c("Serie logaritmica", "Ajuste ARIMA(2,1,2)"),
       col = c("black", "red"), lwd = 2)
```



Ambos modelos parecen ajustarse bien a los datos.

Selección del mejor modelo

Ambos modelos parecen ajustarse bien a los datos, ambos modelos carecen de normalidad, por lo que las pruebas de significancia de los parámetros no son válidas, así mismo pierden válidez los criterios AIC y BIC, sin embargo, teniendo en cuenta que ambos modelos se comportan de manera similar, y que el modelo ARMA(2,2) tiene menor RMSE, se ha seleccionado como el mejor modelo de ambos.