Procesos Estocásticos y Series Temporales

Práctica 2: Análisis Exploratorio y Descomposición de Series Temporales

Francisco Javier Mercader Martínez

PRÁCTICA 2 PARA COMPLETAR

PROCESOS ESTOCÁSTICOS Y SERIES TEMPORALES

GRADO EN CIENCIA E INGENIERÍA DE DATOS

1. Definición y representación gráfica de series temporales

Comenzaremos importando los datos de pasajeros contenidos en el fichero "Pasajeros.csv".

```
datos_df <- read.csv2(file = "Pasajeros.csv")
class(datos_df)
```

```
## [1] "data.frame"
```

```
class(datos_df$pasajeros)
```

```
## [1] "integer"
```

La función *ts()* de R convierte un objeto a serie temporal.

```
datos_ts <- ts(datos_df$pasajeros, frequency = 12, start = c(1949,1))
datos_ts
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

```
class(datos_ts)
```

```
## [1] "ts"
```

Cargamos los datos del dataset AirPassengers directamente de R

```
datos <- AirPassengers
class(datos)
```

```
## [1] "ts"
```

```
datos
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

```
frequency(datos)
```

```
## [1] 12
```

```
start(datos)
```

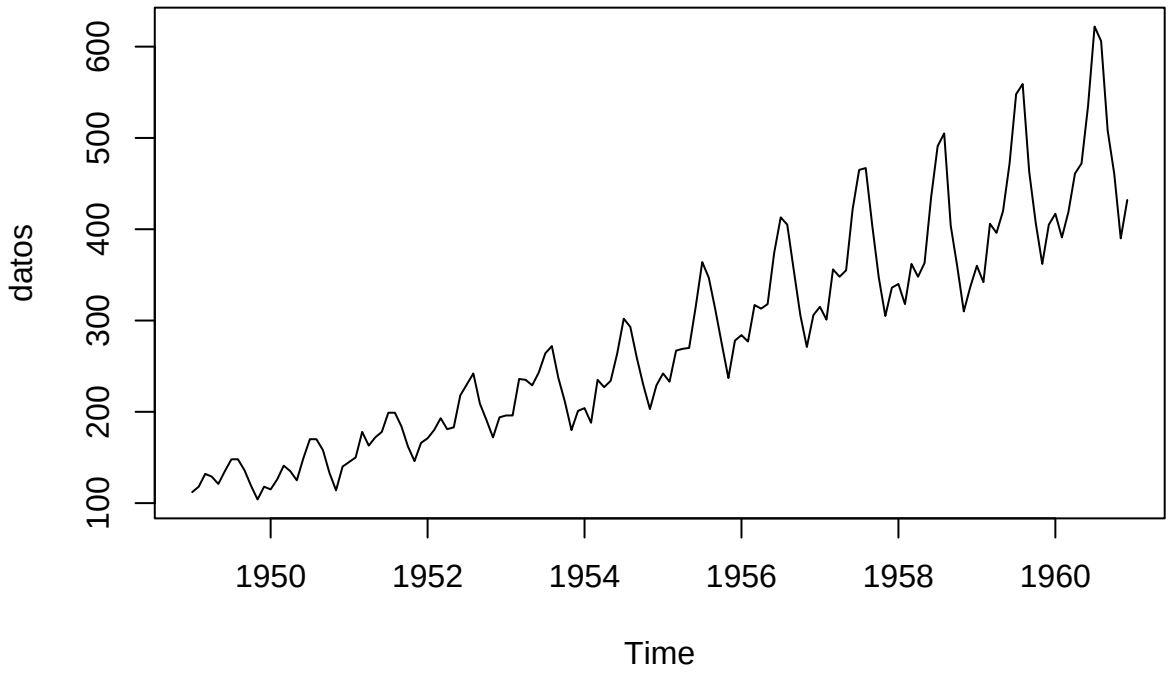
```
## [1] 1949    1
```

```
end(datos)
```

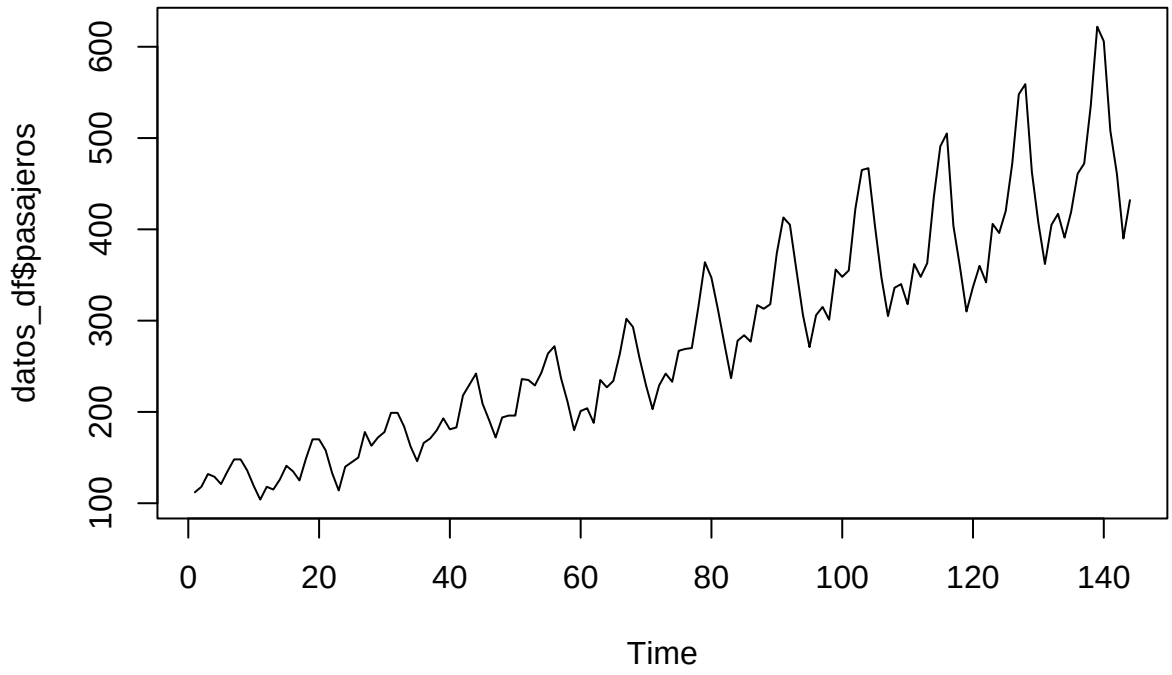
```
## [1] 1960   12
```

Regresentación gráfica de series:

```
ts.plot(datos)
```

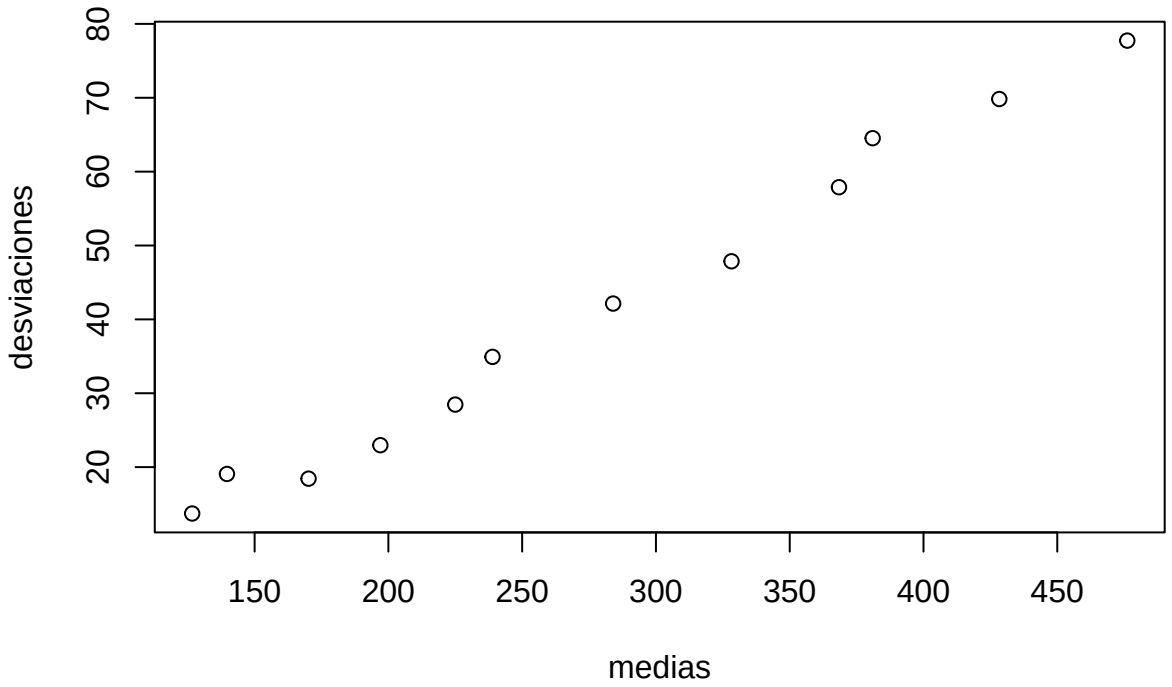



```
ts.plot(datos_df$pasajeros)
```



¿Serie con esquema aditivo o multiplicativo?

```
medias <- as.vector(aggregate(datos, FUN = mean))
desviaciones <- as.vector(aggregate(datos, FUN = sd))
plot(medias, desviaciones)
```



Concluimos por tanto que se trata de un esquema multiplicativo.

2. Algunas operaciones con series temporales

Combinación de series temporales

Unión e intersección de series temporales:

```
set.seed(135)
serie1=ts(rnorm(8), freq = 4, start = c(2020,1))
serie2=ts(rnorm(10), freq = 4, start = c(2020,4))
print("Unión")
```

```
## [1] "Unión"
```

```
#Completar esta línea
ts.union(serie1, serie2)
```

```
##             serie1      serie2
## 2020 Q1 -0.4450708          NA
## 2020 Q2 -0.4659652          NA
## 2020 Q3 -0.1044631          NA
## 2020 Q4  1.4083353  0.42282641
## 2021 Q1 -1.3160867  0.96486359
## 2021 Q2  0.2452088  1.00777310
## 2021 Q3  1.2047431  0.05038159
## 2021 Q4 -0.4407003  0.61003293
## 2022 Q1         NA -1.06719466
## 2022 Q2         NA  0.30162946
## 2022 Q3         NA -1.46547870
## 2022 Q4         NA -0.38005746
## 2023 Q1         NA -0.23219538
```

```
print("Intersección")
```

```
## [1] "Intersección"
```

```
#Completar esta línea
ts.intersect(serie1, serie2)
```

```
##             serie1     serie2
## 2020 Q4  1.4083353 0.42282641
## 2021 Q1 -1.3160867 0.96486359
## 2021 Q2  0.2452088 1.00777310
## 2021 Q3  1.2047431 0.05038159
## 2021 Q4 -0.4407003 0.61003293
```

Extracción de un tramo temporal

Extrae los valores de la serie temporal comprendidos entre una fecha de inicio y otra final.

```
window(serie1, start = c(2021,1),end = c(2021,4))
```

```
##            Qtr1       Qtr2       Qtr3       Qtr4
## 2021 -1.3160867  0.2452088  1.2047431 -0.4407003
```

Serie retardada o adelantada en el tiempo

```
serie1
```

```
##            Qtr1       Qtr2       Qtr3       Qtr4
```

```
## 2020 -0.4450708 -0.4659652 -0.1044631  1.4083353
## 2021 -1.3160867  0.2452088  1.2047431 -0.4407003
```

```
print("serie retardada dos instantes:")
```

```
## [1] "serie retardada dos instantes:"
```

```
#Completar esta línea
lag(serie1, -2)
```

```
##            Qtr1       Qtr2       Qtr3       Qtr4
## 2020                        -0.4450708 -0.4659652
## 2021 -0.1044631  1.4083353 -1.3160867  0.2452088
## 2022  1.2047431 -0.4407003
```

```
print("serie adelantada tres instantes:")
```

```
## [1] "serie adelantada tres instantes:"
```

```
#Completar esta línea
lag(serie1, 3)
```

```
##            Qtr1       Qtr2       Qtr3       Qtr4
## 2019            -0.4450708 -0.4659652 -0.1044631
## 2020  1.4083353 -1.3160867  0.2452088  1.2047431
## 2021 -0.4407003
```

Diferenciación de una serie

```
serie1
```

```
##            Qtr1       Qtr2       Qtr3       Qtr4
## 2020 -0.4450708 -0.4659652 -0.1044631  1.4083353
## 2021 -1.3160867  0.2452088  1.2047431 -0.4407003
```

```
print("serie diferenciada una vez y desfase 1:")
```

```
## [1] "serie diferenciada una vez y desfase 1:"
```

```
diff(serie1, lag = 1, differences = 1)
```

```
##             Qtr1        Qtr2        Qtr3        Qtr4
## 2020            -0.02089445  0.36150218  1.51279839
## 2021 -2.72442205  1.56129552  0.95953435 -1.64544341
```

```
print("serie diferenciada dos veces y desfase 1:")
```

```
## [1] "serie diferenciada dos veces y desfase 1:"
```

```
#Completar esta línea
diff(serie1, lag = 1, differences = 2)
```

```
##            Qtr1       Qtr2       Qtr3       Qtr4
## 2020                        0.3823966  1.1512962
## 2021 -4.2372204  4.2857176 -0.6017612 -2.6049778
```

```
print("serie diferenciada una vez y desfase 4:")
```

```
## [1] "serie diferenciada una vez y desfase 4:"
```

```r
#Completar esta línea
diff(serie1, lag = 4, differences = 1)
```

```
##            Qtr1       Qtr2       Qtr3       Qtr4
## 2021 -0.8710159  0.7111740  1.3092062 -1.8490356
```

Medias móviles de una serie

```r
library(zoo)
```

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

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

```r
serie1
```

```
##            Qtr1       Qtr2       Qtr3       Qtr4
## 2020 -0.4450708 -0.4659652 -0.1044631  1.4083353
## 2021 -1.3160867  0.2452088  1.2047431 -0.4407003
```

```r
print("media móvil centrada de orden 3:")
```

```
## [1] "media móvil centrada de orden 3:"
```

```r
rollmean(serie1,  k = 3, fill = NA, align = "center")
```

```
##              Qtr1         Qtr2         Qtr3         Qtr4
## 2020           NA -0.338499701  0.279302338 -0.004071491
## 2021  0.112485796  0.044621735  0.336417221           NA
```

3. Descomposición clásica de series temporales

3.1. Extracción de las componentes

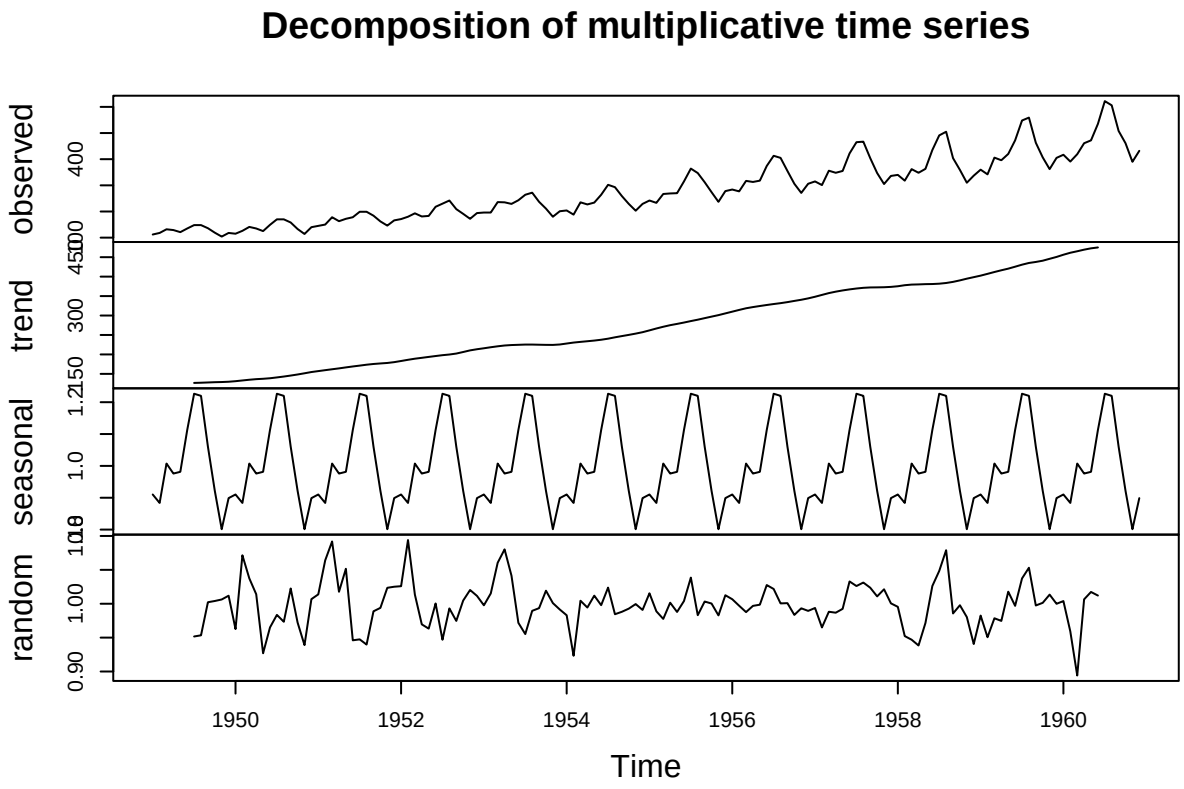
Realizar la descomposición clásica del objeto "datos":

```r
descomposicion_clasica <- decompose(datos, type = "multiplicative")
summary(descomposicion_clasica)
```

```
##          Length Class  Mode
## x        144    ts     numeric
## seasonal 144    ts     numeric
## trend    144    ts     numeric
## random   144    ts     numeric
## figure    12    -none- numeric
## type       1    -none- character
```

Representar las componentes extraídas:

```r
plot(descomposicion_clasica)
```



3.2. Predicciones: obtención de un modelo determinista

Ajuste lineal de la tendencia respecto al tiempo:

```
tiempo <- 1:length(datos)
tendencia.lm <- lm(descomposicion_clasica$trend ~ tiempo)
tendencia.lm
```

```
##
## Call:
## lm(formula = descomposicion_clasica$trend ~ tiempo)
##
## Coefficients:
## (Intercept)       tiempo
##      84.648        2.667
```

¿Cómo obtener los valores ajustados en el tramo observado y realizar predicciones en instantes futuros?

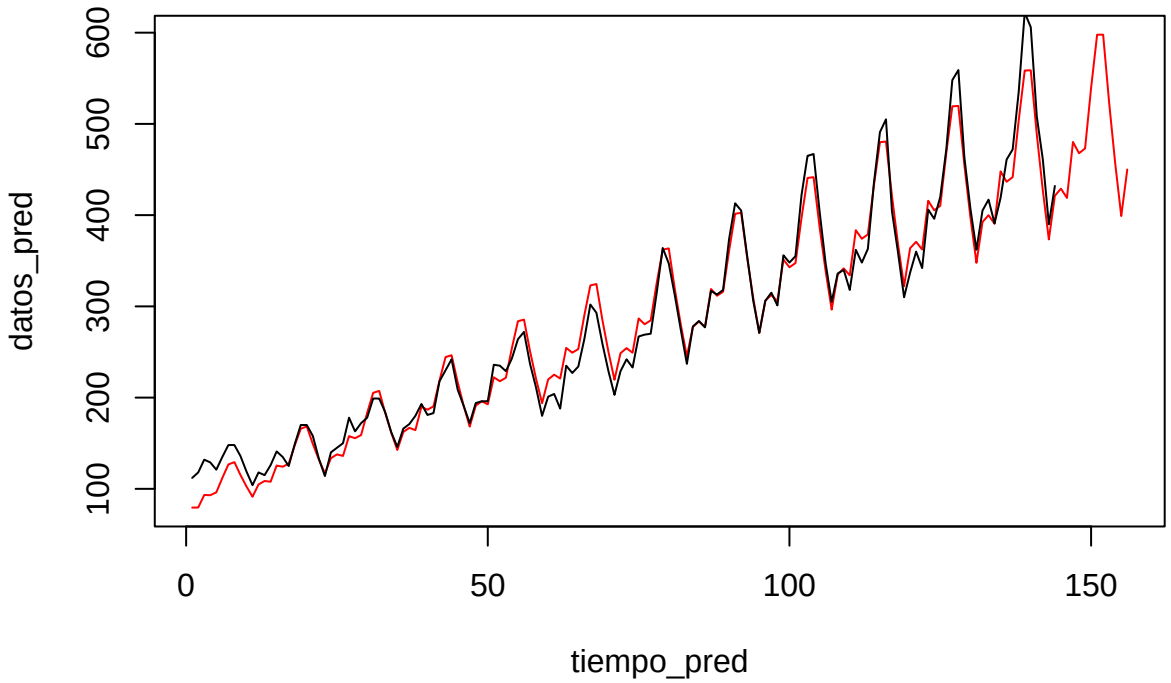
```
tiempo_pred <- 1:(length(datos) + 12)

estacional_pred <- c(descomposicion_clasica$seasonal,
                     descomposicion_clasica$seasonal[1:12])

tendencia_pred <- predict(tendencia.lm, data.frame(tiempo = tiempo_pred))
```

Las predicciones se obtienen de la siguiente forma:

```
datos_pred <- tendencia_pred * estacional_pred
plot(tiempo_pred, datos_pred, type="l", col = "red")
lines(tiempo,datos_df$pasajeros,type="l")
```



Residuos y medidas de error (bondad del ajuste)

Calculamos los residuos y medidas de error asociadas:

```
residuos <- datos - datos_pred[1:length(datos)]
MAE <- mean(abs(residuos))
RMSE <- sqrt(mean(residuos^2))
MAE
```

```
## [1] 13.45225
```

```
RMSE
```

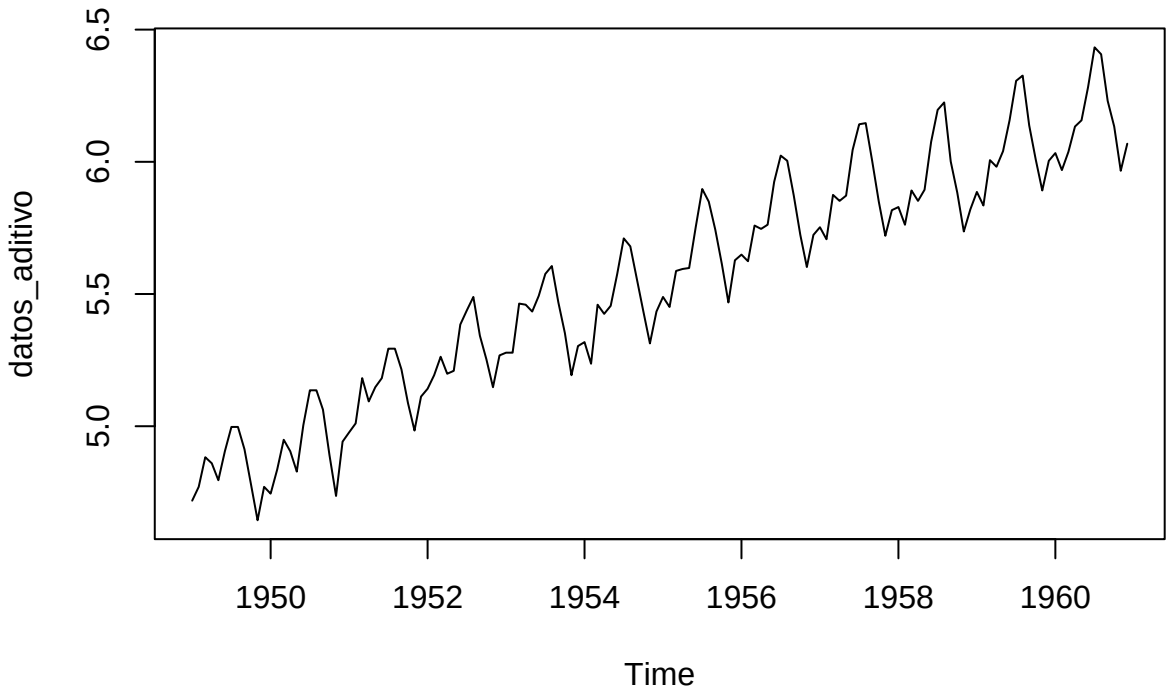
```
## [1] 17.32319
```

4. Descomposición STL de series temporales

4.1. Extracción de las componentes con STL

Transformamos los datos de AirPassengers para tener esquema aditivo:

```
datos_aditivo <- log(AirPassengers)
ts.plot(datos_aditivo)
```



Descomposicón STL:

```
STL <- stl(datos_aditivo, s.window = 7)
#str(STL)
summary(STL)
```

```
##  Call:
##  stl(x = datos_aditivo, s.window = 7)
##
##  Time.series components:
##     seasonal               trend            remainder
##  Min.   :-0.22188182   Min.   :4.809113   Min.   :-0.06875615
##  1st Qu.:-0.08857735   1st Qu.:5.201819   1st Qu.:-0.01027597
##  Median :-0.01380803   Median :5.552237   Median :-0.00104014
##  Mean   : 0.00053801   Mean   :5.541836   Mean   :-0.00019840
##  3rd Qu.: 0.07274467   3rd Qu.:5.916865   3rd Qu.: 0.01350479
##  Max.   : 0.25725445   Max.   :6.197002   Max.   : 0.06572901
##  IQR:
##       STL.seasonal STL.trend STL.remainder data
##       0.16132      0.71505   0.02378       0.69453
##     %  23.2         103.0       3.4         100.0
##
##  Weights: all == 1
##
##  Other components: List of 5
##  $ win  : Named num [1:3] 7 23 13
##  $ deg  : Named int [1:3] 0 1 1
##  $ jump : Named num [1:3] 1 3 2
##  $ inner: int 2
##  $ outer: int 0
```

Guardamos las 3 componentes extraídas:

```r
estacional_STL <- STL$time.series[, 1]
tendencia_STL <- STL$time.series[, 2]
irregular_STL <- STL$time.series[, 3]
```

Otra forma usando el paquete *forecast*.

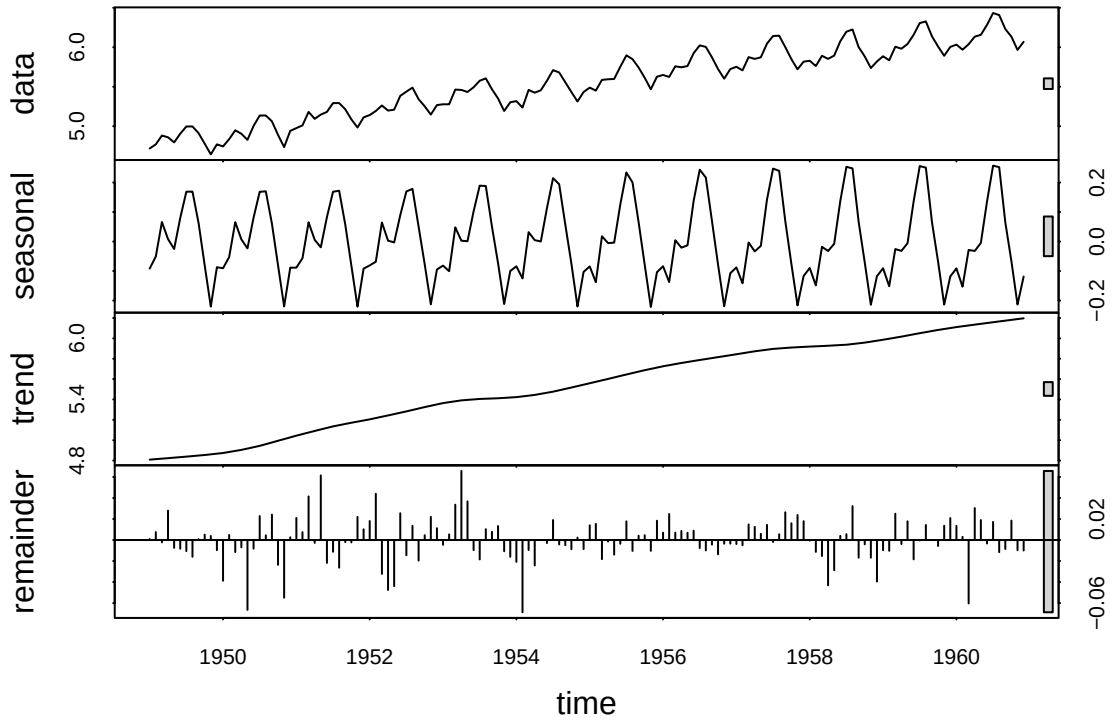
```r
library(forecast)
```

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

```r
Estacional_STL <- seasonal(STL)
Tendencia_STL <- trendcycle(STL)
Irregular_STL <- remainder(STL)
```

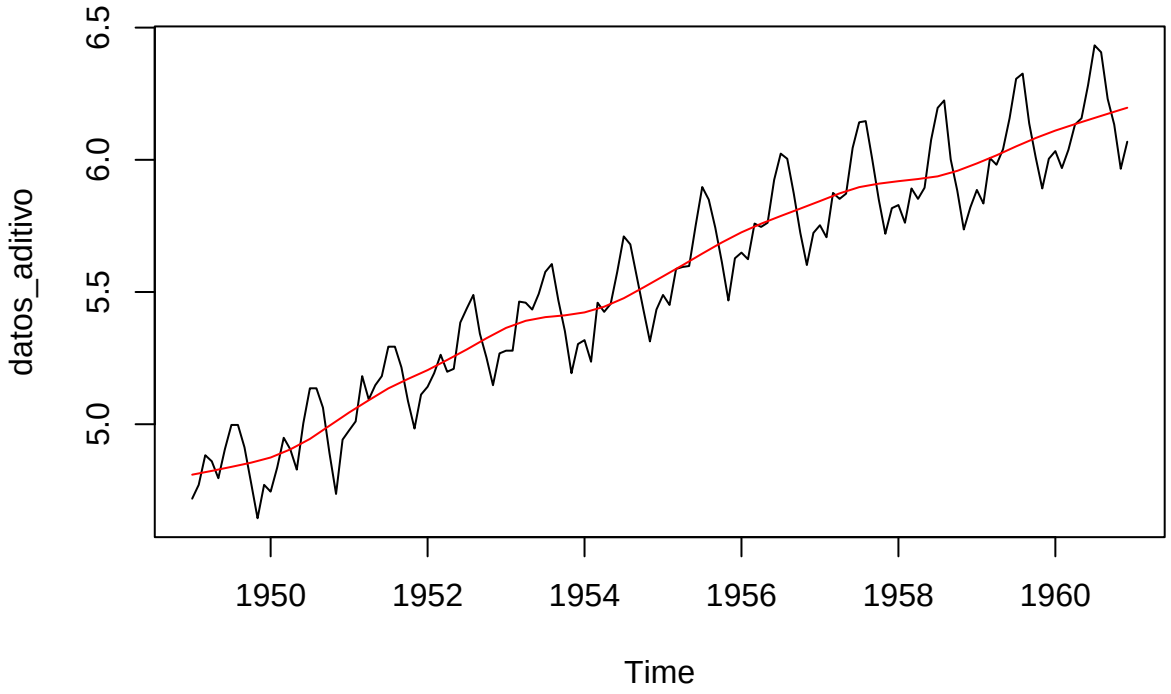
Representamos las componentes extraídas:

```r
plot(STL)
```



Representemos ahora la serie de datos aditiva junto con la tendencia extraída por STL:

```r
plot(datos_aditivo)
lines(tendencia_STL, col = "red")
```

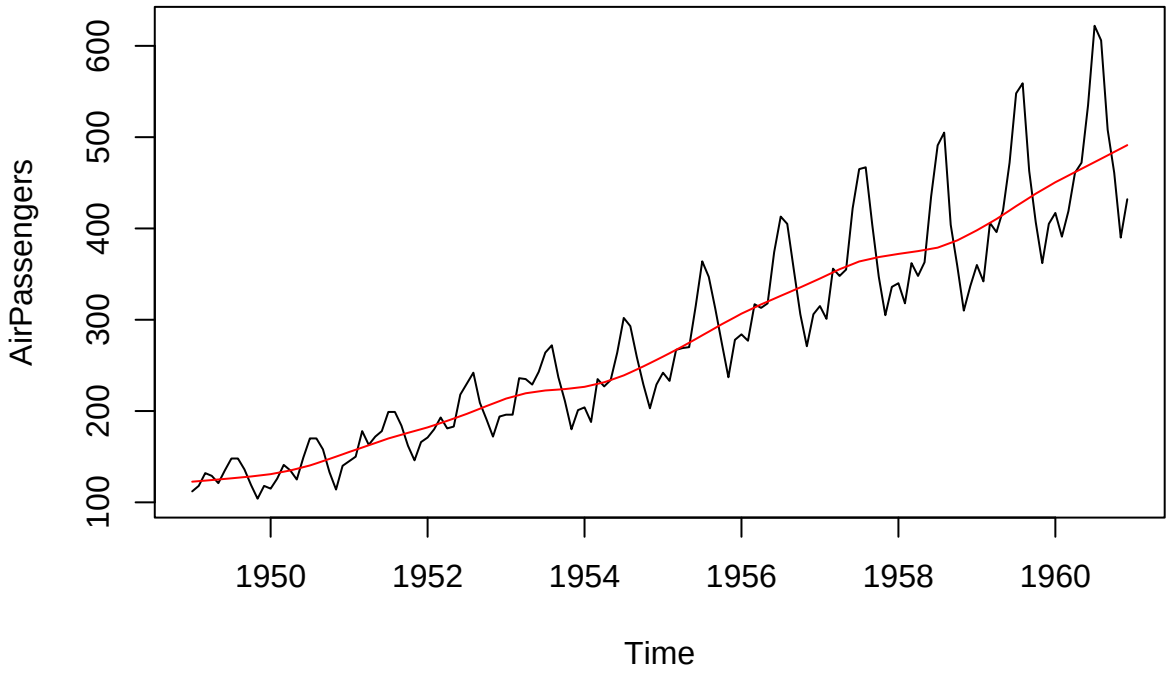


Recordemos que el método STL ha extraido las componentes del esquema aditivo. Si queremos las componentes de la serie original (esquema multiplicativo), tendremos que deshacer el cambio tomando exponenciales.

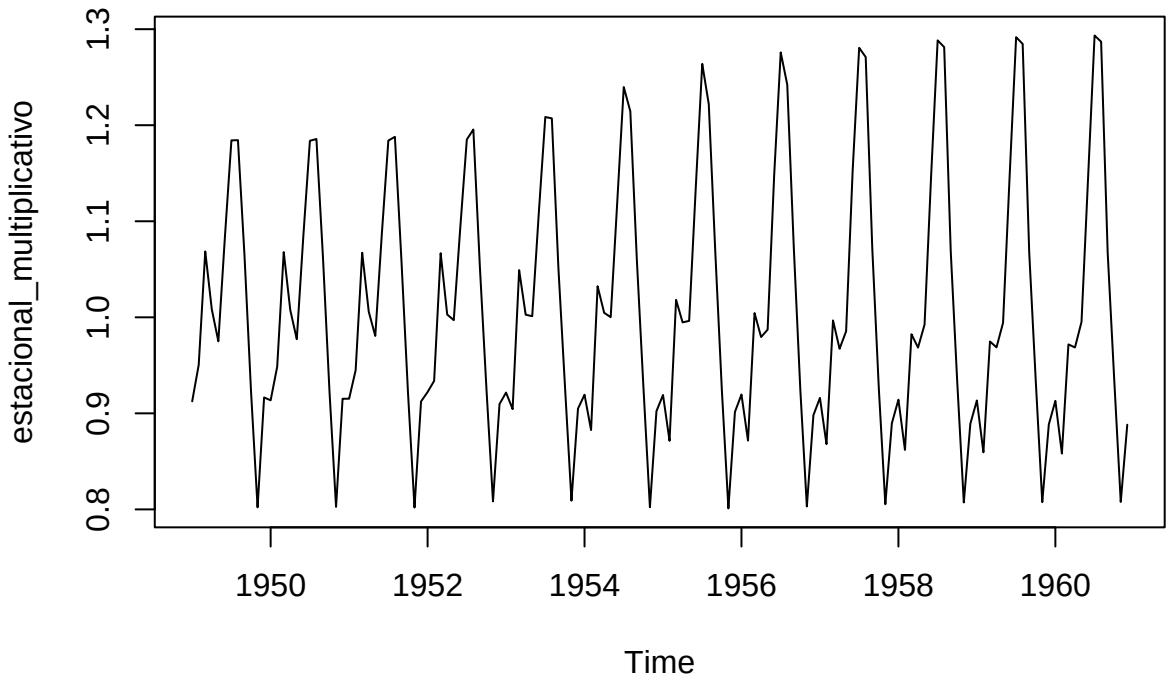
```
estacional_multiplicativo <- exp(estacional_STL)
tendencia_multiplicativo <- exp(tendencia_STL)
irregular_multiplicativo <- exp(irregular_STL)
```

Representemos la serie original (esquema multiplicativo) y sus componentes.

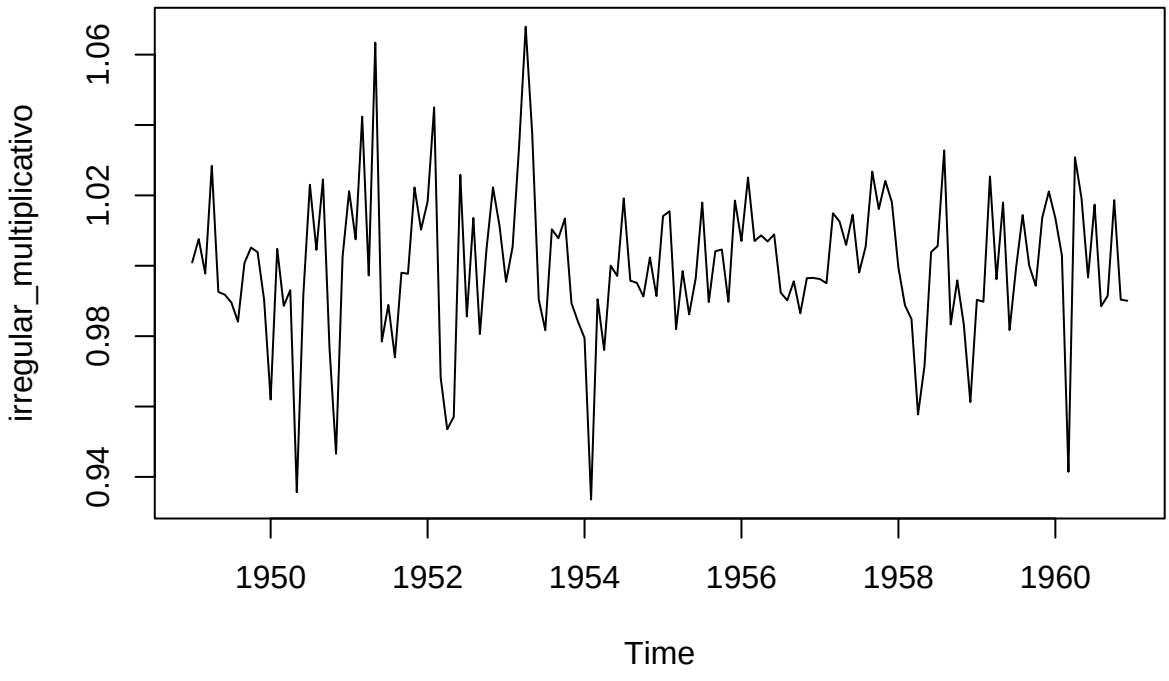
```
plot(AirPassengers)
lines(tendencia_multiplicativo, col = "red")
```



```r
ts.plot(estacional_multiplicativo)
```




```r
ts.plot(irregular_multiplicativo)
```



4.2. Predicciones con descomposición STL

Primero realizamos ajuste lineal de la tendencia respecto al tiempo:

```r
tiempo2 <- 1:length(datos_aditivo)
tendencia_aditivo.lm <- lm(tendencia_STL ~ tiempo2)
tendencia_aditivo.lm
```

```
##
## Call:
## lm(formula = tendencia_STL ~ tiempo2)
##
## Coefficients:
## (Intercept)      tiempo2
##     4.81357      0.01005
```

¿Cómo obtener los valores ajustados en el tramo observado y realizar predicciones en instantes futuros?

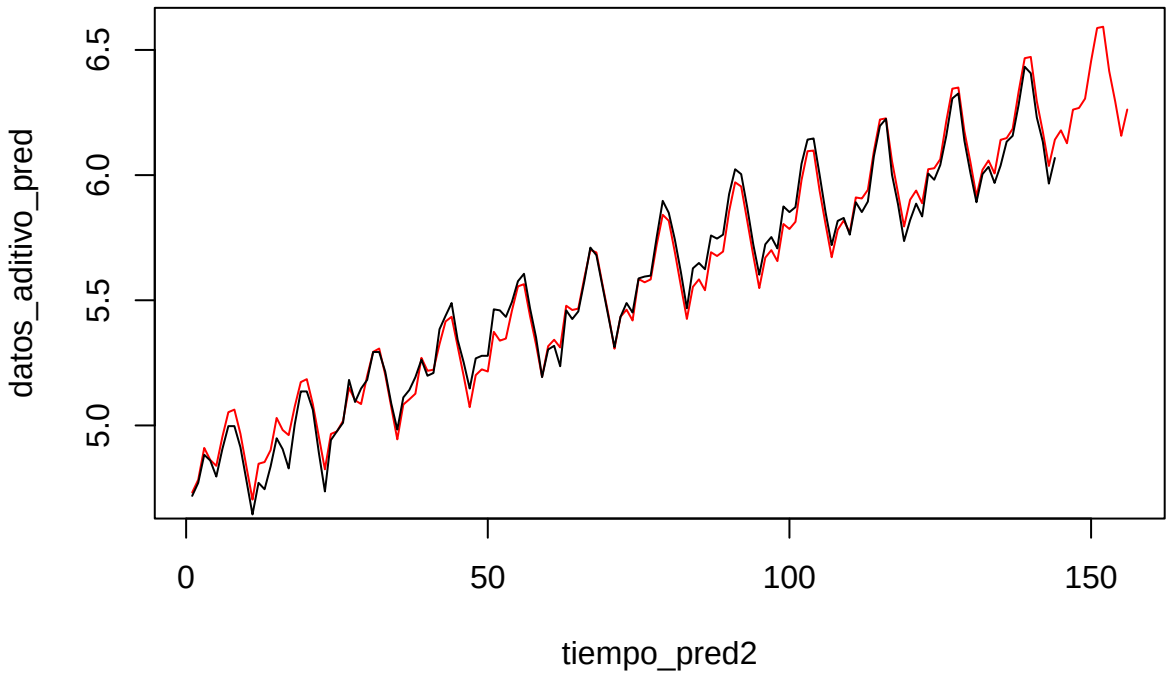
```
tiempo_pred2 <- 1:(length(datos_aditivo) + 12)

estacional_STL_pred <- c(estacional_STL, estacional_STL[133:144])

tendencia_STL_pred <-  predict(tendencia_aditivo.lm,
                               data.frame(tiempo2 = tiempo_pred2))
```

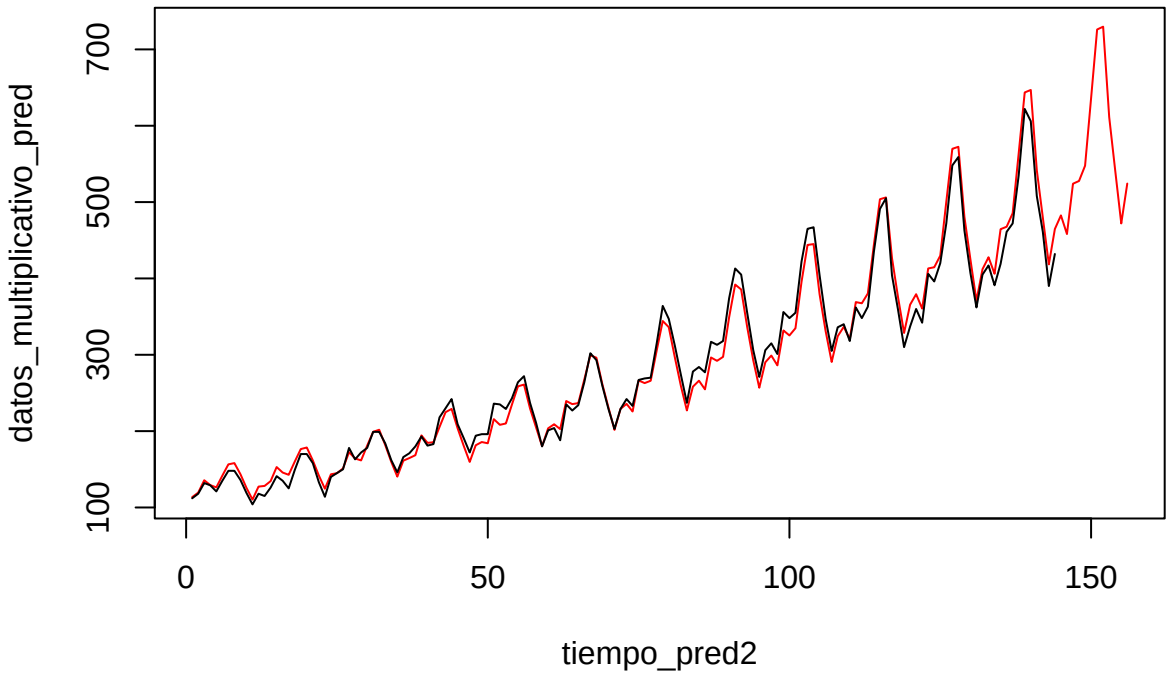
Las predicciones del esquema aditivo se obtienen de la siguiente forma:

```
datos_aditivo_pred <- tendencia_STL_pred + estacional_STL_pred
plot(tiempo_pred2, datos_aditivo_pred, type="l", col = "red")
lines(tiempo2, datos_aditivo, type="l")
```



Por tanto, las predicciones de la serie original de pasajeros (esquema multpilicativo) se obtienen tomando exponenciales.

```
datos_multiplicativo_pred <- exp(datos_aditivo_pred)
plot(tiempo_pred2, datos_multiplicativo_pred, type="l", col = "red")
lines(tiempo2, AirPassengers, type="l")
```



Residuos y medidas de error (bondad del ajuste)

```
residuos_multiplicativo <- AirPassengers - datos_multiplicativo_pred[1:length(AirPassengers)]
MAE_multiplicativo <- mean(abs(residuos_multiplicativo))
RMSE_multiplicativo <- sqrt(mean(residuos_multiplicativo ^ 2))
MAE_multiplicativo
```

```
## [1] 12.00082
```

```
RMSE_multiplicativo
```

```
## [1] 14.85867
```