Ejercicio Guiado - Series Temporales

Néstor Rodriguez Vico

06 April, 2019

- Néstor Rodríguez Vico
- nrv23@correo.ugr.es

- Ejercicio guiado. Curso 2018-2019

Carga de librerías:

```
library(tseries)
```

```
NPred = 12
NTest = 12
serie <- scan("pasajeros_1949_1959.dat")
serie.ts <- ts(serie, frequency = 12)
```

Analisis inicial de la serie

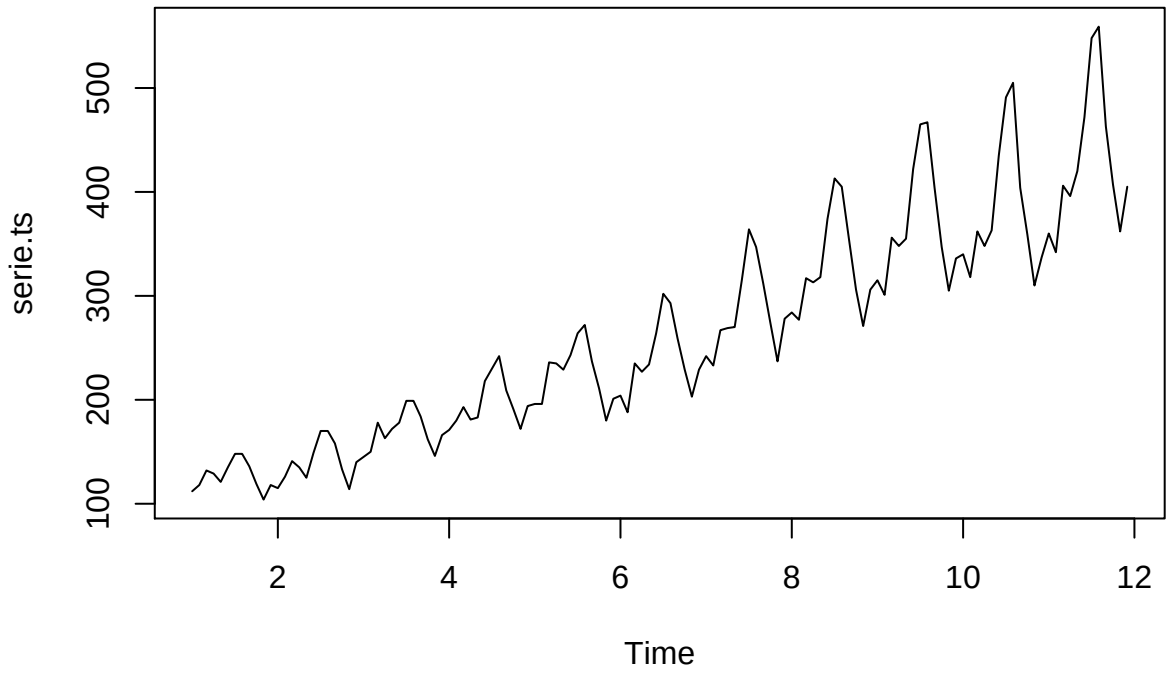
Veamos la serie en sí:

```
serie.ts
```

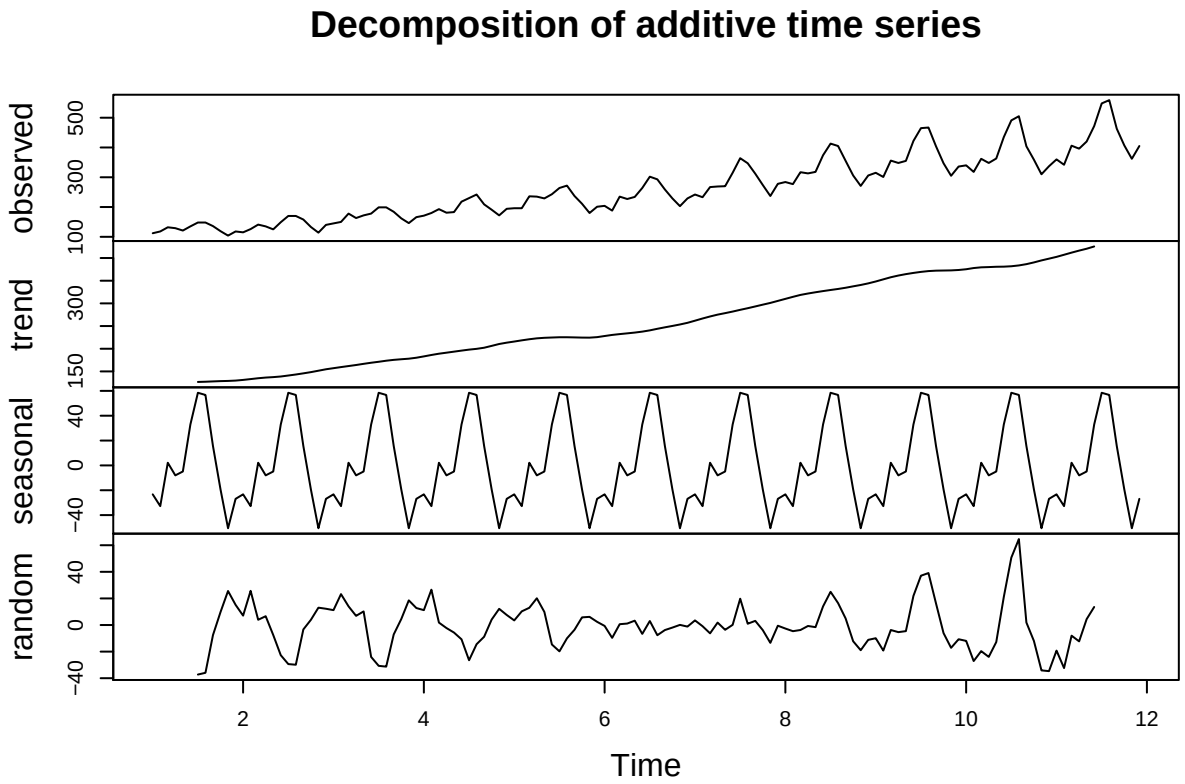
```
##    Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1  112 118 132 129 121 135 148 148 136 119 104 118
## 2  115 126 141 135 125 149 170 170 158 133 114 140
## 3  145 150 178 163 172 178 199 199 184 162 146 166
## 4  171 180 193 181 183 218 230 242 209 191 172 194
## 5  196 196 236 235 229 243 264 272 237 211 180 201
## 6  204 188 235 227 234 264 302 293 259 229 203 229
## 7  242 233 267 269 270 315 364 347 312 274 237 278
## 8  284 277 317 313 318 374 413 405 355 306 271 306
## 9  315 301 356 348 355 422 465 467 404 347 305 336
## 10 340 318 362 348 363 435 491 505 404 359 310 337
## 11 360 342 406 396 420 472 548 559 463 407 362 405
```

Vamos a pintar algunos gráficos para ver la serie:

```
plot(serie.ts)
```

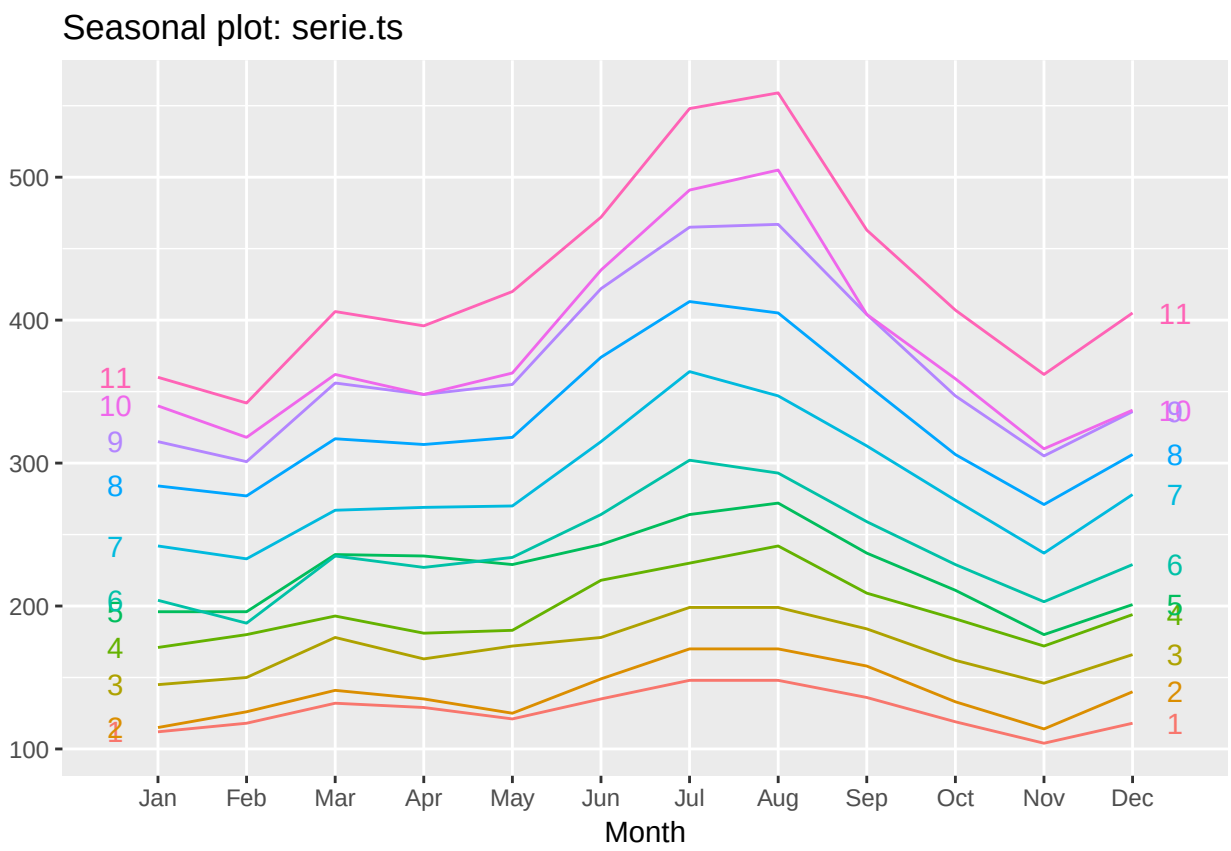



```
plot(decompose(serie.ts))
```



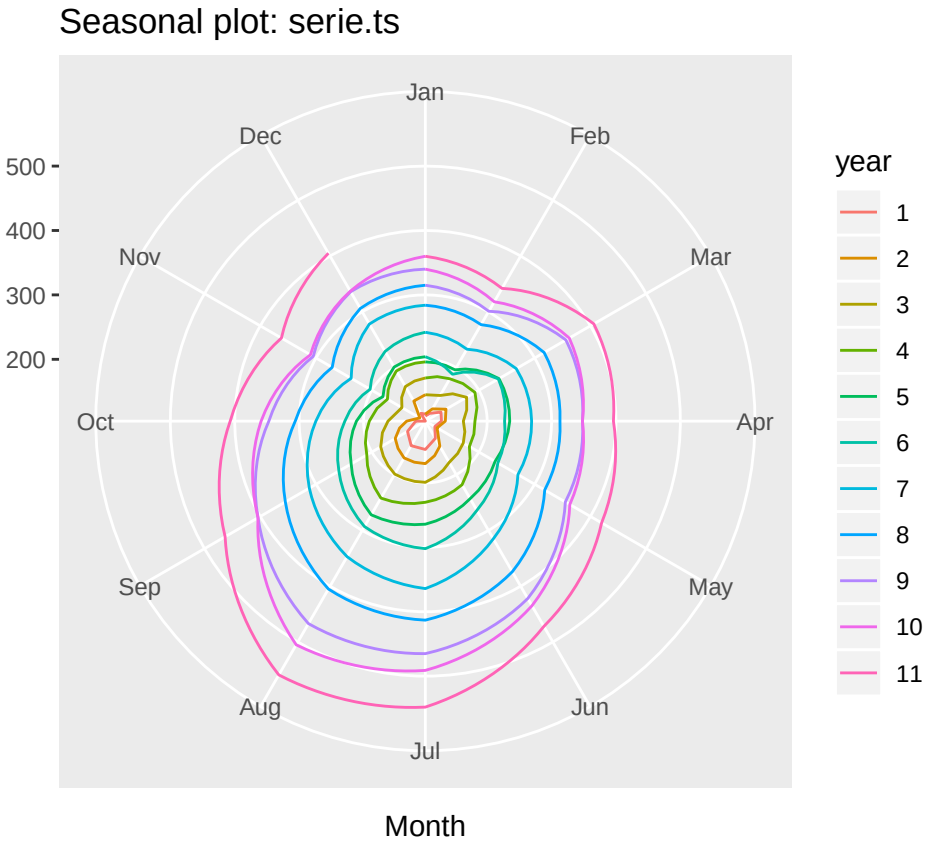
Como podemos ver, hay variación en la varianza. Podemos usar el parquete forecast junto con el método ggseasonplot para ver más información de los datos (idea sacada de la página web https://otexts.com/fpp2/graphics.html). Podemos pintar la serie en cada año:

```
forecast::ggseasonplot(serie.ts, year.labels = T, year.labels.left = T)
```



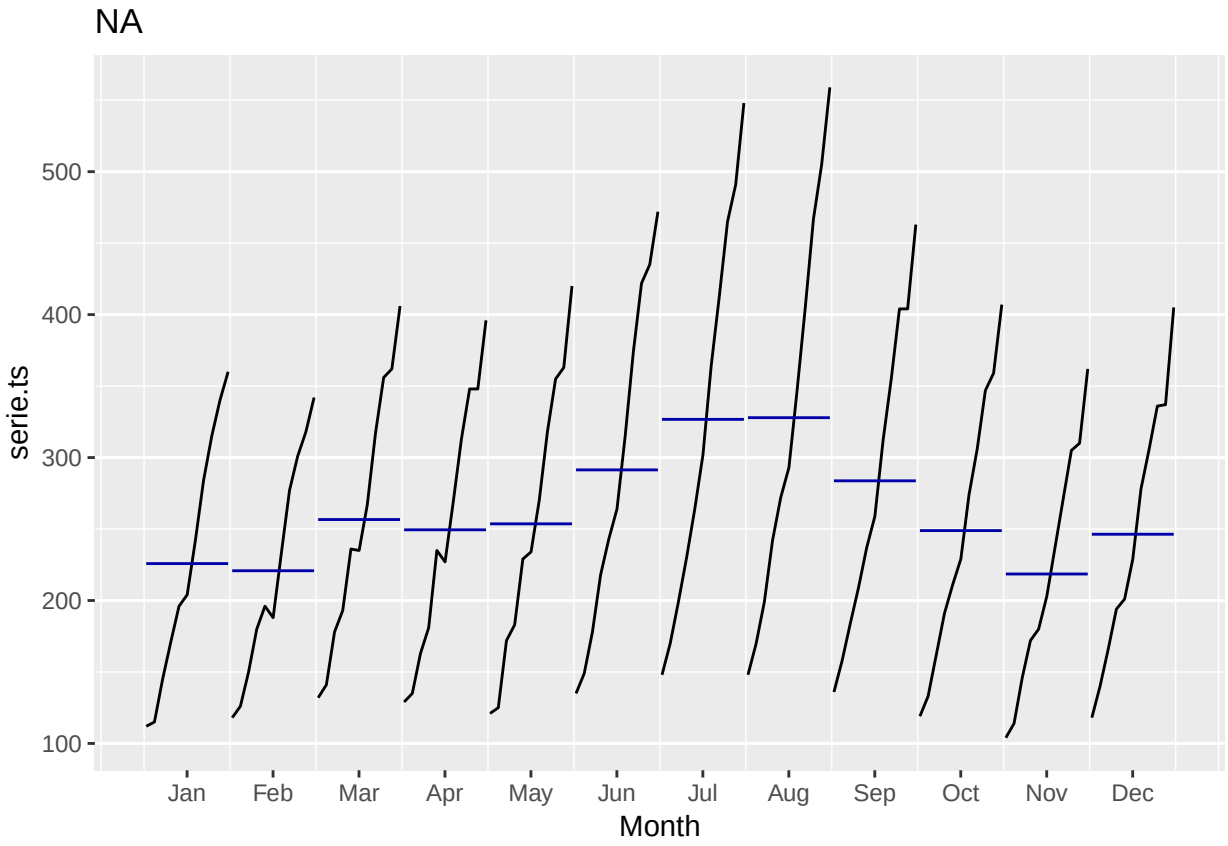
En la gráfica podemos ver como hay una tendencia creciente, ya que cada años aumenta el número de pasajeros.

```
forecast::ggseasonplot(serie.ts, polar=TRUE)
```



En este segundo gráfico también podemos ver la tendencia creciente del número de pasajeros por año. Podemos ver más información de la serie con el siguiente comando:

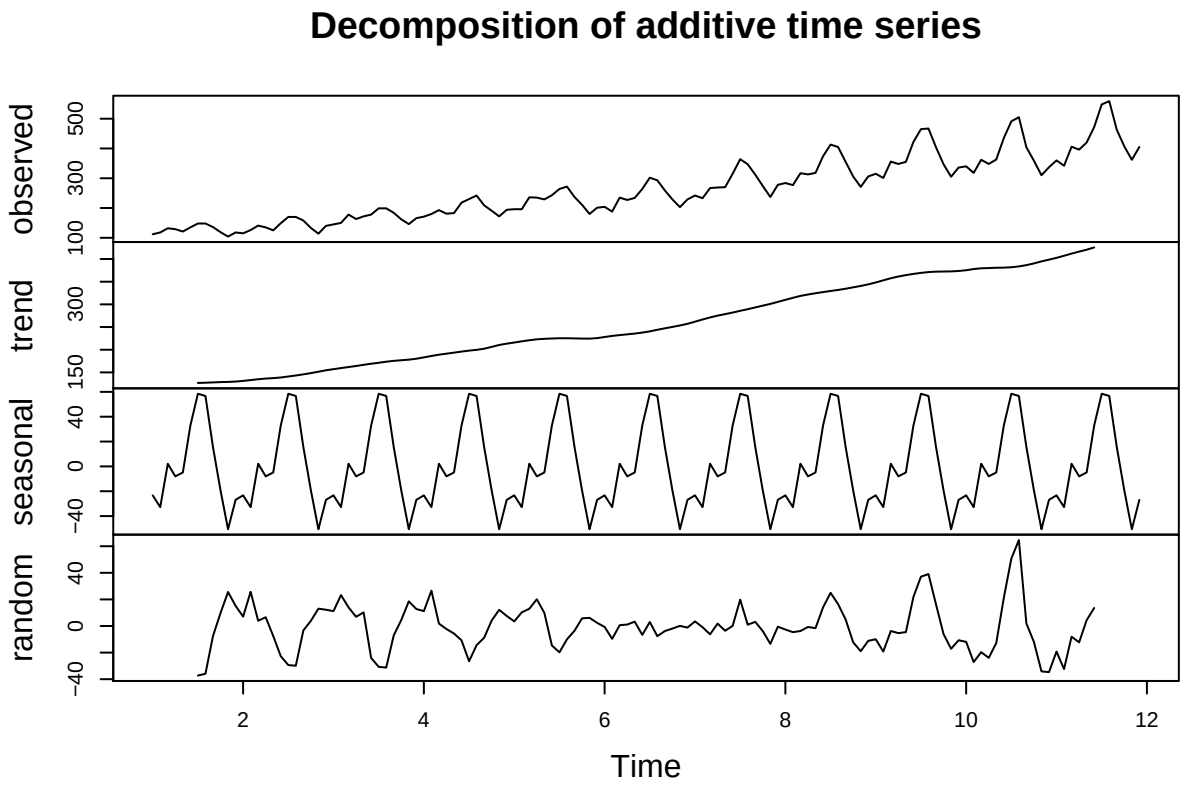
```
forecast::ggsubseriesplot(serie.ts)
```



En este gráfico podemos ver los valores de la serie para cada mes junto con la media.

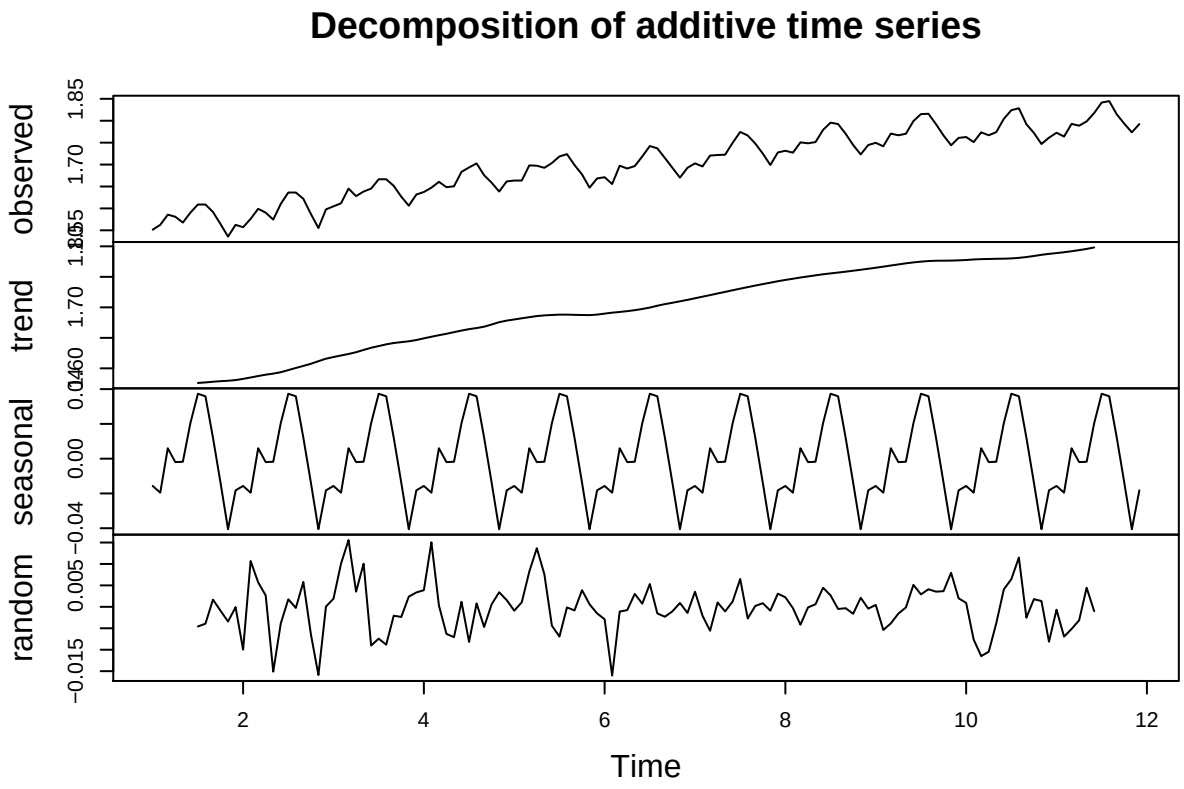
Preprocesamiento

```
plot(decompose(serie.ts))
```



Como podemos ver, la sreie tiene bastante varianza, así que será necesario reducirla con una transformación logarítmica:

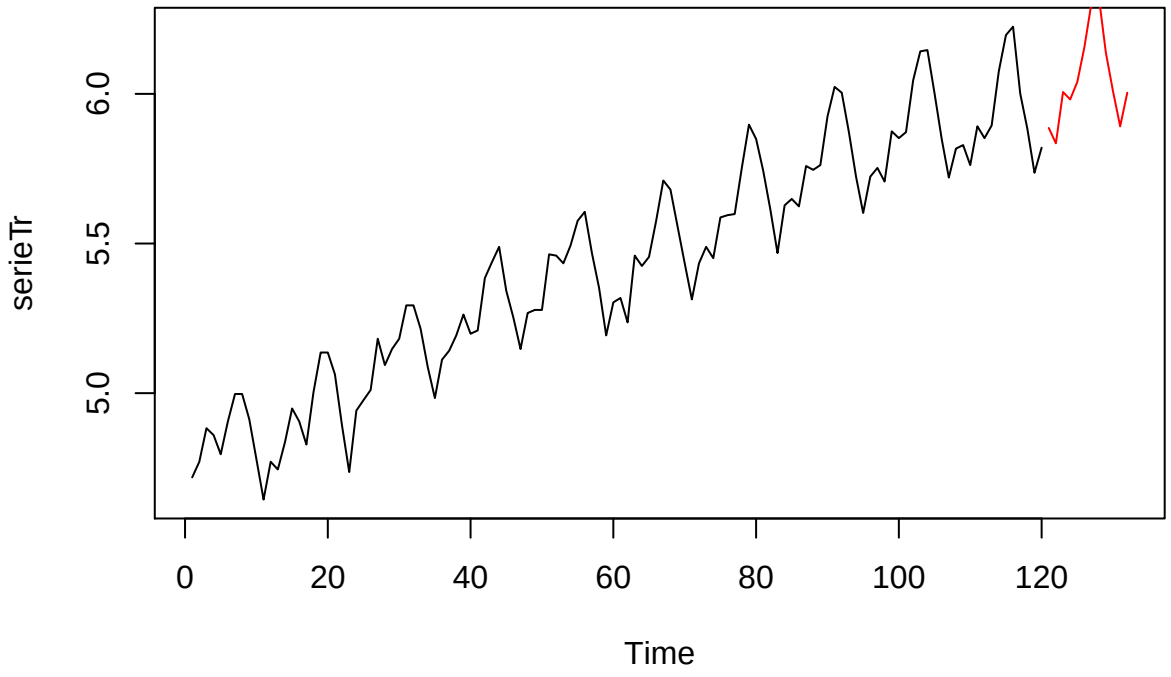
```
serie.ts <- log(serie.ts)
serie.log <- log(serie)
plot(decompose(log(serie.ts)))
```



Particionamiento en train y test

Solucionado el problema de la varianza. Vemos que hay tendencia y estacionalidad. El siguiente paso será dividir los datos para ajuste y test. Cogeremos, por ejemplo, los 12 últimos valores para el test dado que también necesitaremos predecir 12 valores de la serie.

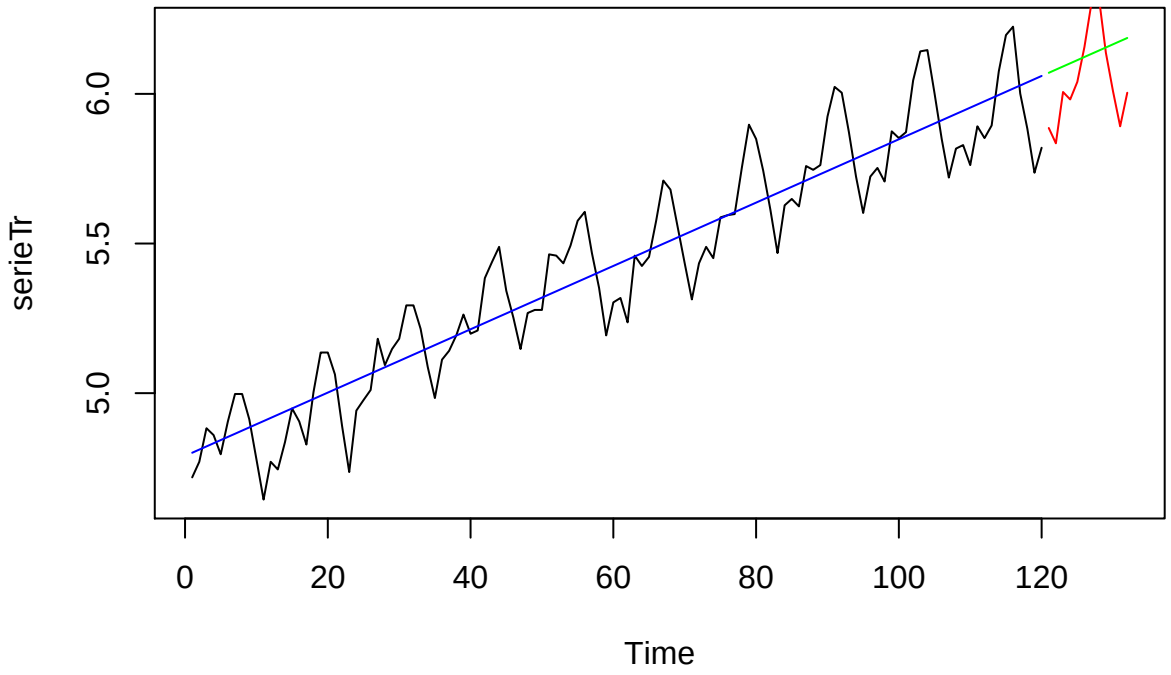
```
serieTr <- serie.log[1:(length(serie.log)-NTest)]
tiempoTr <- 1:length(serieTr)
serieTs <- serie.log[(length(serie.log)-NTest+1):length(serie)]
tiempoTs <- (tiempoTr[length(tiempoTr)]+1):(tiempoTr[length(tiempoTr)]+NTest)

plot.ts(serieTr, xlim = c(1, tiempoTs[length(tiempoTs)]))
lines(tiempoTs, serieTs, col = "red")
```



Tendencia

A continuación, modelaremos la tendencia. Debemos modelar primero la tendencia ya que puede darse el caso de que no seamos capaces de detectar estacionalidad si no quitamos primero la tendencia. Para modelar la tendencia vamos a usar una aproximaxión funcional. Asumiremos un comportamiento lineal en este caso como hipótesis.

```
parametros.H1 <- lm(serieTr ~ tiempoTr)

#Calculamos la estimación de la tendencia.
TendEstimadaTr.H1 <- parametros.H1$coefficients[1]+tiempoTr*parametros.H1$coefficients[2]

TendEstimadaTs.H1 <- parametros.H1$coefficients[1]+tiempoTs*parametros.H1$coefficients[2]

#Mostramos en la misma figura la serie y la tendencia estimada
plot.ts(serieTr, xlim=c(1, tiempoTs[length(tiempoTs)]))
lines(tiempoTr, TendEstimadaTr.H1, col="blue")
lines(tiempoTs, serieTs, col="red")
lines(tiempoTs, TendEstimadaTs.H1, col="green")
```



Comprobamos que la hipótesis de tendencia lineal es válida. Para ello se aplica un T-test, asumiendo normalidad en los datos (u otro test no paramétrico si los datos no son normales), que compare los residuos del ajuste con los errores del modelo en test. En nuestro caso, todos los tests de normalidad dan p-value>0.05. Asumimos normalidad. También el T-test da un p-value>0.05. No existen diferencias significativas en los datos.

```
#Test de normalidad de Jarque Bera
jarque.bera.test(parametros.H1$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  parametros.H1$residuals
## X-squared = 1.755, df = 2, p-value = 0.4158
```

```
jarque.bera.test((TendEstimadaTs.H1 - serieTs))
```

```
##
##  Jarque Bera Test
##
## data:  (TendEstimadaTs.H1 - serieTs)
## X-squared = 0.87957, df = 2, p-value = 0.6442
```

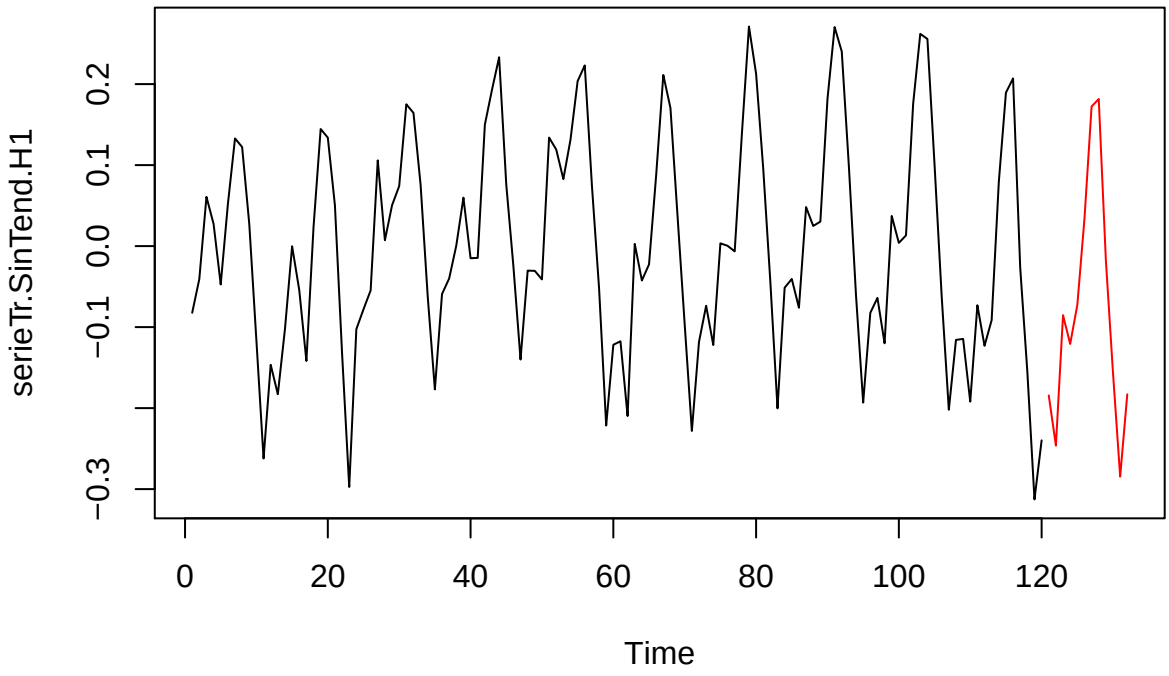
```
#Test de Student
t.test(c(parametros.H1$residuals, TendEstimadaTs.H1-serieTs))
```

```
##
##  One Sample t-test
##
## data:  c(parametros.H1$residuals, TendEstimadaTs.H1 - serieTs)
## t = 0.61219, df = 131, p-value = 0.5415
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.01628561  0.03088222
## sample estimates:
```

```
##   mean of x
## 0.007298304
```

Como podemos ver, no se encuentran diferencias significativas tanto en el proceso de entrenamiento (p-value 0.4158) como en el proceso de test (0.6442), así que podemos decir que una aproximación con un modelo lineal es interesante. El test de Student no muestra diferenias significativas, así que podemos confirmar que el modelo lineal puede ser usado. A continuación eliminamos la tendencia en la serie:

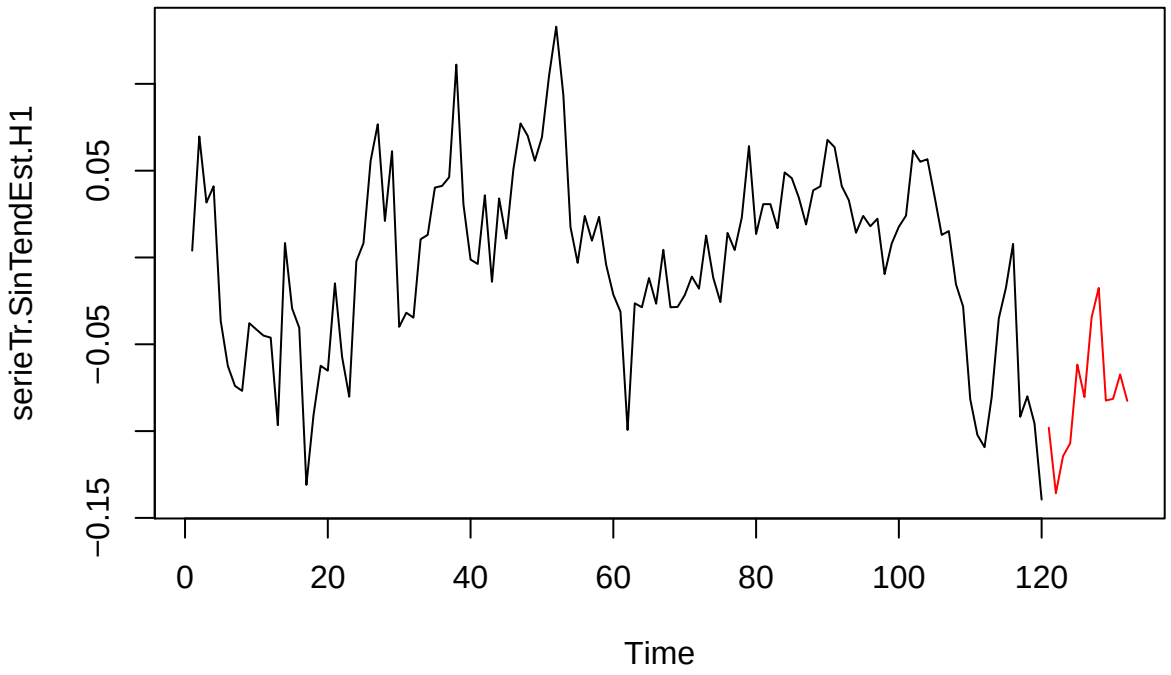
```
serieTr.SinTend.H1 <- serieTr - TendEstimadaTr.H1
serieTs.SinTend.H1 <- serieTs - TendEstimadaTs.H1
plot.ts(serieTr.SinTend.H1, xlim = c(1, tiempoTs[length(tiempoTs)]))
lines(tiempoTs, serieTs.SinTend.H1, col = "red")
```



Estacionalidad

El siguiente paso es eliminar la estacionalidad. Para ello, vamos a usar un modelo de diferenciación. Al comienzo, asumimos una estacionalidad anual (12 meses/valores de la serie) al crear el objeto ts (diapositiva 5): serie.ts<- ts(serie, frequency=12). Para eliminar la estacionalidad, podemos hacer uso de las salidas de la función "decompose":

```
#Calculamos y elimnamos la estacionalidad
k <- 12 #Asumimos periodo de estacionalidad k = 12
estacionalidad.H1 <- decompose(serie.ts)$seasonal[1:k]

#Eliminamos estacionalidad para el modelo
aux <- rep(estacionalidad.H1, length(serieTr)/length(estacionalidad.H1))
serieTr.SinTendEst.H1 <- serieTr.SinTend.H1 - aux
serieTs.SinTendEst.H1 <- serieTs.SinTend.H1 - estacionalidad.H1
plot.ts(serieTr.SinTendEst.H1, xlim=c(1, tiempoTs[length(tiempoTs)]))
lines(tiempoTs, serieTs.SinTendEst.H1, col = "red")
```



Con la serie sin tendencia ni estacionalidad, debemos comprobar si es estacionaria antes de hipotetizar modelos de predicción. Usamos el Test de Dickey-Fuller aumentado. Diferenciamos la serie tras aplicar el test, que falla. Volvemos a aplicar el test a la serie diferenciada (esta vez sí se pasa el test)

```
#Comprobamos el test de Dickye-Fuller aumentado para la estacionaridad
adf.test(serieTr.SinTendEst.H1)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  serieTr.SinTendEst.H1
## Dickey-Fuller = -1.8407, Lag order = 4, p-value = 0.6427
## alternative hypothesis: stationary
```

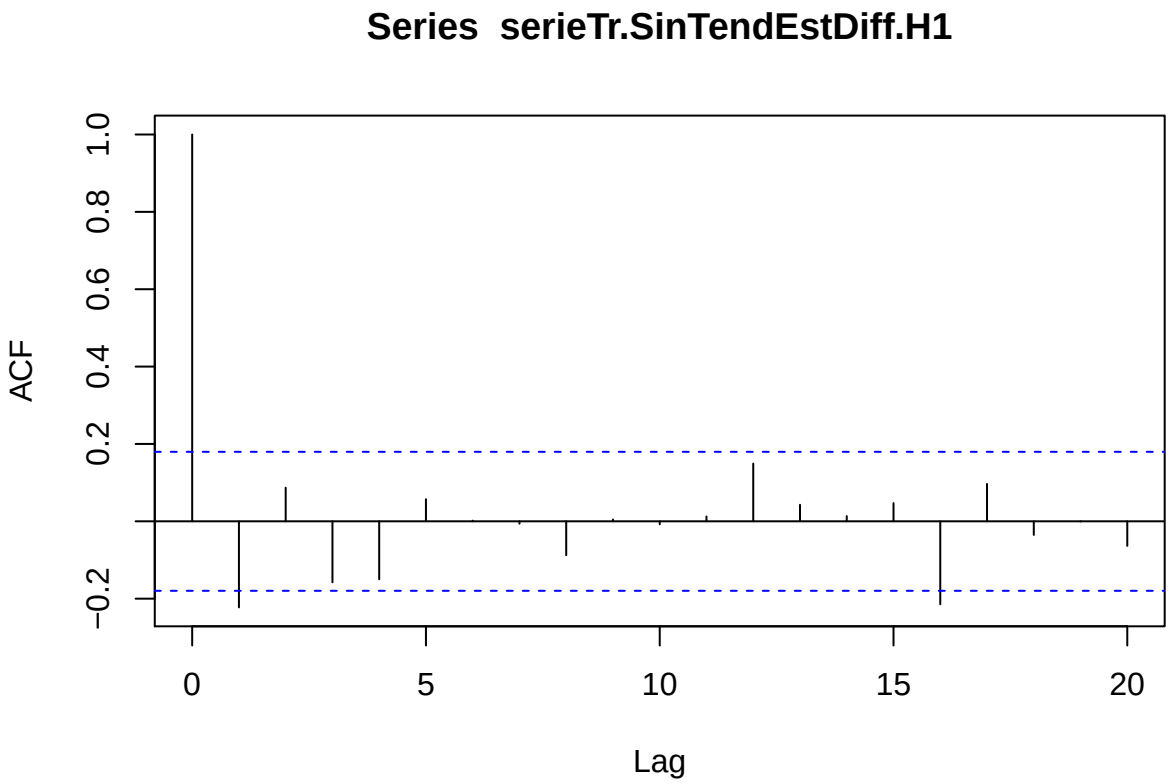
```
#Como no se supera (valor>0.05), diferenciamos la serie
serieTr.SinTendEstDiff.H1 <- diff(serieTr.SinTendEst.H1)
serieTs.SinTendEstDiff.H1 <- diff(serieTs.SinTendEst.H1)

#Volvemos a aplicar el test
adf.test(serieTr.SinTendEstDiff.H1)
```

```
## Warning in adf.test(serieTr.SinTendEstDiff.H1): p-value smaller than
## printed p-value
```

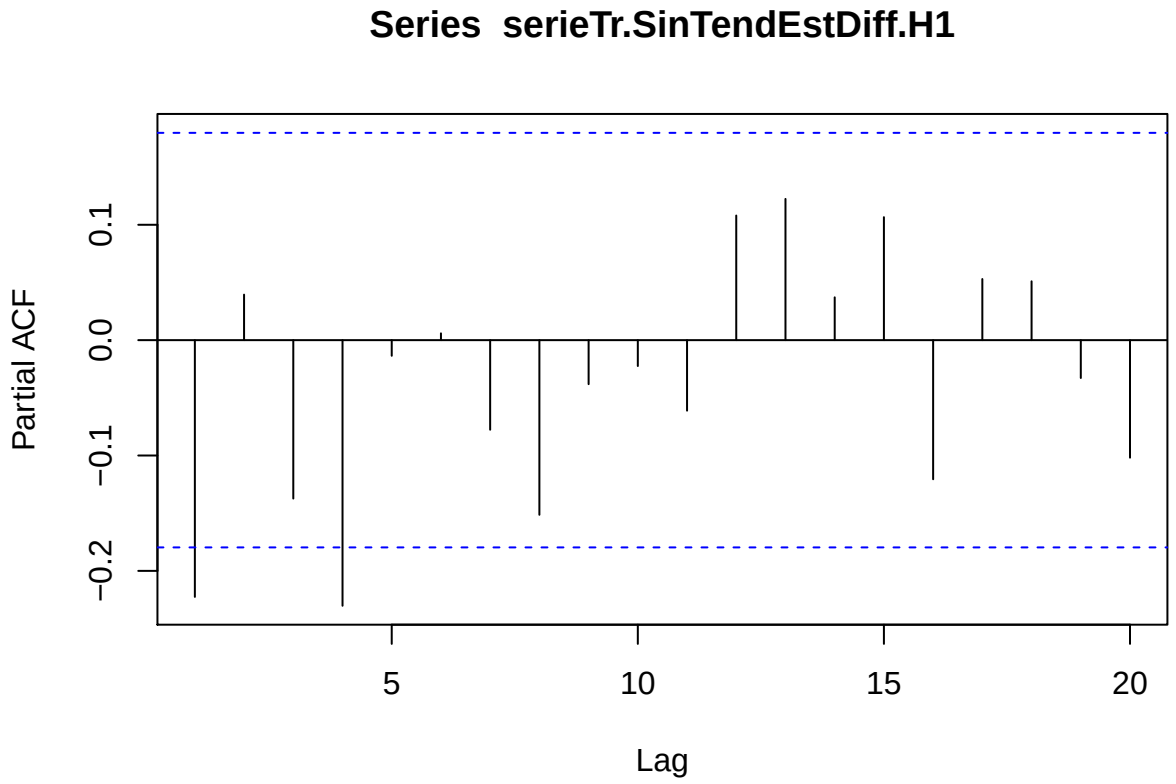
```
##
##  Augmented Dickey-Fuller Test
##
## data:  serieTr.SinTendEstDiff.H1
## Dickey-Fuller = -6.2153, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

Obtenemos un p-value de 0.6427, así que el test adf falla y, por lo tanto, la serie no es estacionaria. Podemos ver que, tras aplicar diferenciación, la serie que tenemos si es estacionaria. Con una serie ya estacionaria, vamos a visualizar ACF y PACF para poder dar hipótesis de modelos.

```
acf(serieTr.SinTendEstDiff.H1)
```




```
pacf(serieTr.SinTendEstDiff.H1)
```



Visualizando ACF y PACF, podríamos estar ante un modelo AR(4), un modelo MA(1). La intensidad de los picos es muy baja, por lo que podríamos estar cerca, también, de un error de distribución normal con media cero. Vamos a asumir un modelo AR(4). Sabemos que es 4 ya que es el último valor que pasa el umbral en el PACF. Como hemos diferenciado 1 instante de tiempo, podríamos incluir la diferencia dentro del modelo ajustando un ARIMA(4, 1, 0). También calculamos error de ajuste y test (para comparar con otros modelos, si deseamos probar con más).

```
# Ajustamos el modelo
modelo.H1 <- arima(serieTr.SinTendEst.H1, order = c(4, 1, 0))
valoresAjustados.H1 <- serieTr.SinTendEst.H1 + modelo.H1$residuals

#Calculamos las predicciones
Predicciones.H1 <- predict(modelo.H1, n.ahead = NPred)
valoresPredichos.H1 <- Predicciones.H1$pred

#Calculamos el error cuadrático acumulado del ajuste, en ajuste y en test
errorTr.H1 <- sum((modelo.H1$residuals)^2)
errorTr.H1
```

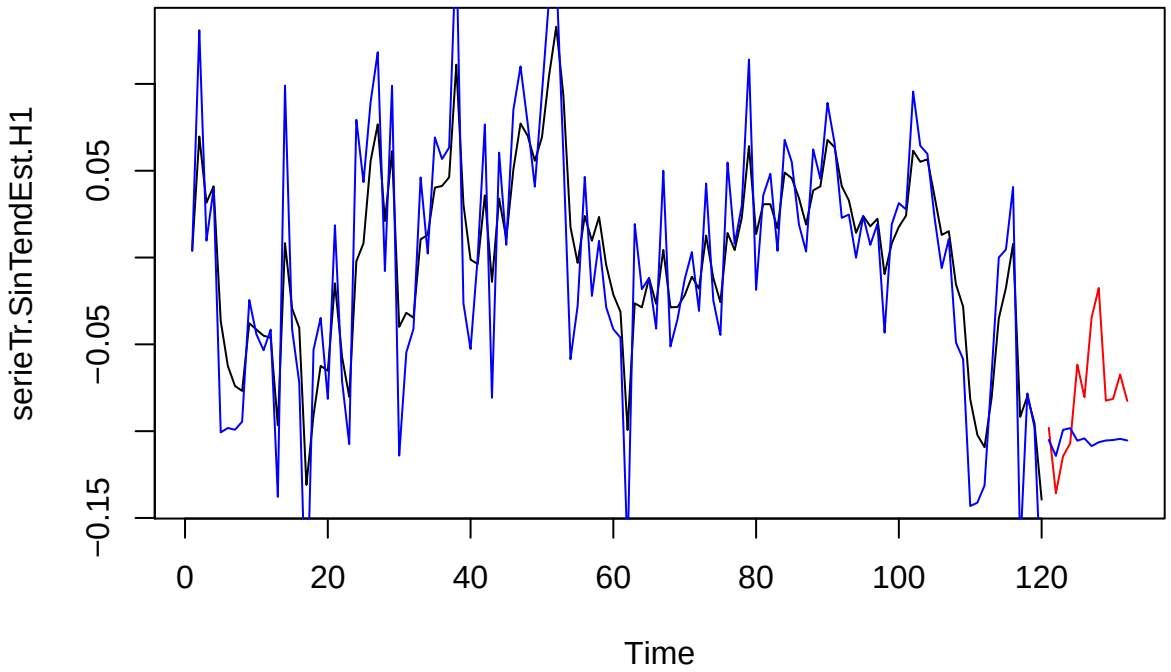
```
## [1] 0.1344656
```

```
errorTs.H1 <- sum((valoresPredichos.H1-serieTs.SinTendEst.H1)^2)
errorTs.H1
```

```
## [1] 0.01965443
```

Ilustramos los resultados a continuación:

```
# Mostramos las gráficas del ajuste y predicción en test
plot.ts(serieTr.SinTendEst.H1, xlim=c(1, tiempoTs[length(tiempoTs)]))
lines(valoresAjustados.H1, col = "blue")
lines(tiempoTs, serieTs.SinTendEst.H1, col = "red")
lines(tiempoTs, valoresPredichos.H1, col = "blue")
```



Finalmente, validamos el modelo:

- Test de Box-Pierce para aleatoriedad de residuos (lo pasa).
- Tests de Jarque Bera y Shapiro-Wilk para normalidad de residuos (los pasa)
- Mostramos histograma y función de densidad para confirmación gráfica.

```
#Tests para la seccion del modelo y su validacion
Box.test(modelo.H1$residuals)
```

```
##
##  Box-Pierce test
##
## data:  modelo.H1$residuals
## X-squared = 0.005349, df = 1, p-value = 0.9417
```

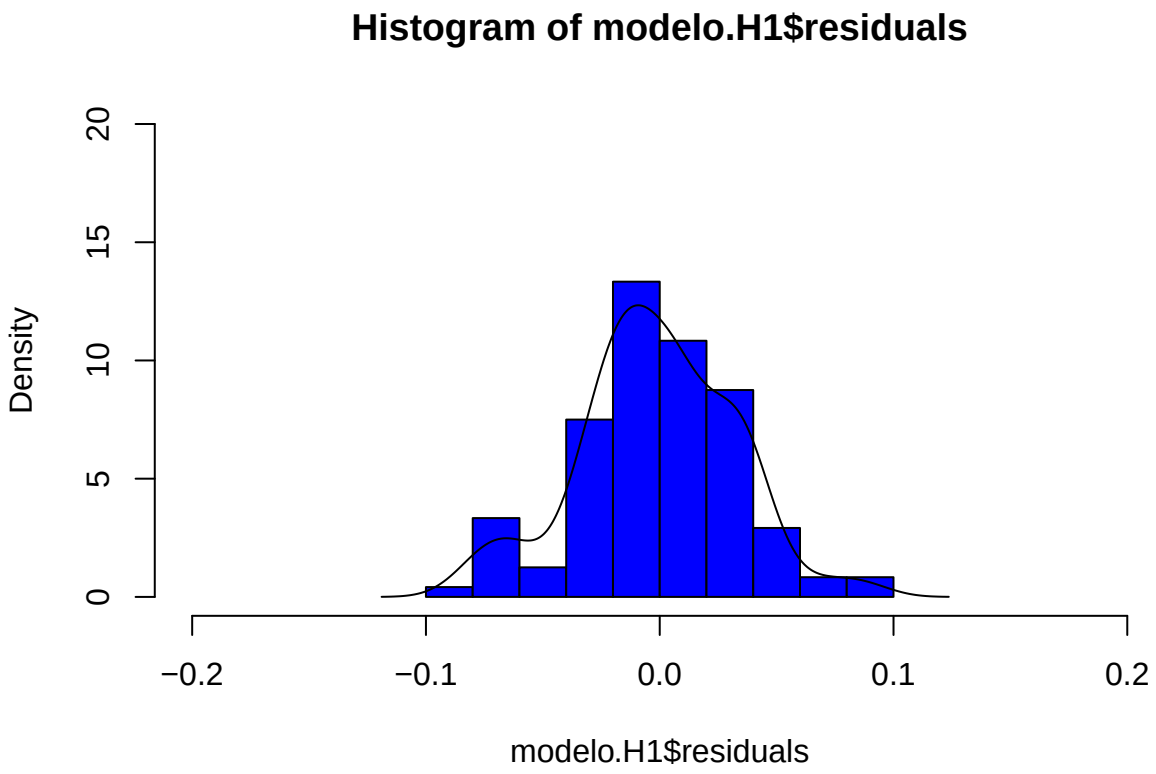
```
#Test de normalidad de Jarque Bera
jarque.bera.test(modelo.H1$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  modelo.H1$residuals
## X-squared = 0.39988, df = 2, p-value = 0.8188
```

```
#Test de normalidad de Shaphiro-Wilk
shapiro.test(modelo.H1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  modelo.H1$residuals
## W = 0.9867, p-value = 0.2904
```

```
hist(modelo.H1$residuals, col = "blue", prob = T, ylim=c(0, 20), xlim=c(-0.2, 0.2))
lines(density(modelo.H1$residuals))
```



Al pasar el test de Box podemos afirmar que los los residuos son aleatorios y, por lo tanto, nuestro modelo es válido. Con el test de Jarque Bera podemos afirmar la normalidad de los residuos, lo cual también lo podemos afirmar con el test de Shaphiro-Wilk.

Selección de modelo

Si se hubiesen ejecutado varios modelos (varios modelos ARIMA con distintos parámetros, un modelo AR o un modelo MA) podríamos calcular el error cuadrático acumulado del ajuste y quedarnos con el modelo que cometa un menor error.

Modelo final

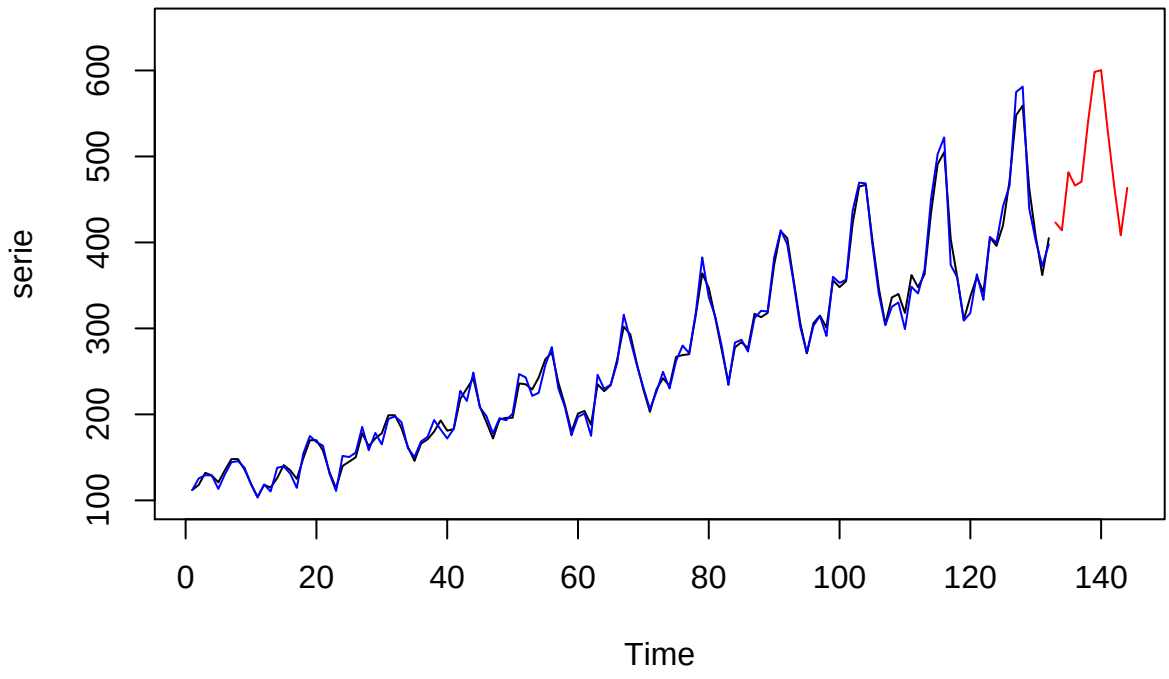
Una vez que hemos validado todo el modelo, volvemos a seguir los pasos iniciales, sin dividir la serie en ajuste y test, para hacer la predicción de los meses de 1960.

```
serieEntera <- serie.log #Cogemos toda la serie
tiempo <- 1:length(serieEntera)
parametros <- lm(serieEntera ~ tiempo) #Ajustamos modelo de tendencia
TendEstimada <- parametros$coefficients[1] + tiempo * parametros$coefficients[2]
serieSinTend <- serieEntera - TendEstimada
aux <- ts(serieEntera, frequency = 12)
aux <- decompose(aux)$seasonal
estacionalidad <- as.numeric(aux[1:12])
aux <- rep(estacionalidad, length(serieSinTend) / length(estacionalidad))
serieSinTendEst <- serieSinTend - aux
modelo <- arima(serieSinTendEst, order = c(4, 1, 0))
valoresAjustados <- serieSinTendEst + modelo$residuals
```

```
Predicciones <- predict(modelo, n.ahead = NPred)
valoresPredichos <- Predicciones$pred #Cogemos las predicciones
```

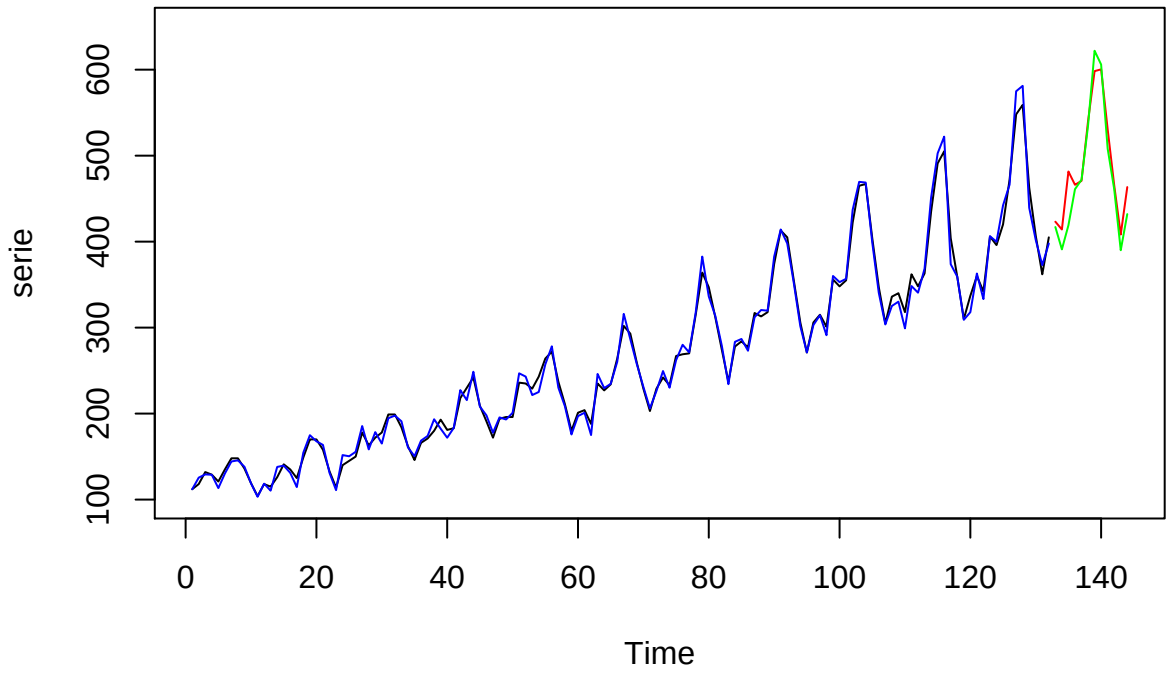
Por último, deshacemos los cambios realizamos para calcular las predicciones reales:

```
#Por ultimo, deshacemos cambios
 #Estacionalidad
valoresAjustados <- valoresAjustados + aux
valoresPredichos <- valoresPredichos + estacionalidad

valoresAjustados <- valoresAjustados + TendEstimada
tiempoPred <- (tiempo[length(tiempo)]+(1:NPred))
TendEstimadaPred <- parametros$coefficients[1] + tiempoPred*parametros$coefficients[2]
valoresPredichos <- valoresPredichos + TendEstimadaPred

 #Transformación log de los datos
valoresAjustados <- exp(valoresAjustados)
valoresPredichos <- exp(valoresPredichos)

plot.ts(serie, xlim=c(1, max(tiempoPred)), ylim=c(100, 650))
lines(valoresAjustados, col = "blue")
lines(valoresPredichos, col = "red")
```



Si tuviésemos disponibles los datos correctos de la predicción, podríamos compararlos con los resultados para validar el modelo.

```
plot.ts(serie, xlim=c(1, max(tiempoPred)), ylim=c(100, 650))
lines(valoresAjustados, col = "blue")
lines(valoresPredichos, col = "red")

#Cargamos los valores reales de predicción y los mostramos
predReales <- scan("pasajeros_1960.predict")
lines(tiempoPred, predReales, col = "green")
```




```
#Calculamos el error de predicción
ErrorMedio <- sum(abs(predReales - valoresPredichos))
ErrorMedio
```

```
## [1] 209.5667
```