Series Temporales: Trabajo Guiado

Juanjo Sierra

19 de abril de 2019

Paquetes a cargar

Importamos los paquetes que necesitamos para resolver los problemas planteados para la práctica.

```
library(tseries)
```

Problema

Dadas las cifras de pasajeros que vuelan con una aerolínea por año (en miles, anotadas mensualmente) entre los años 1949 y 1959, se pide estimar la cantidad de pasajeros que esa misma aerolínea tendrá cada mes durante el año 1960.

Se leen los datos de los pasajeros que nos proporciona la aerolínea.

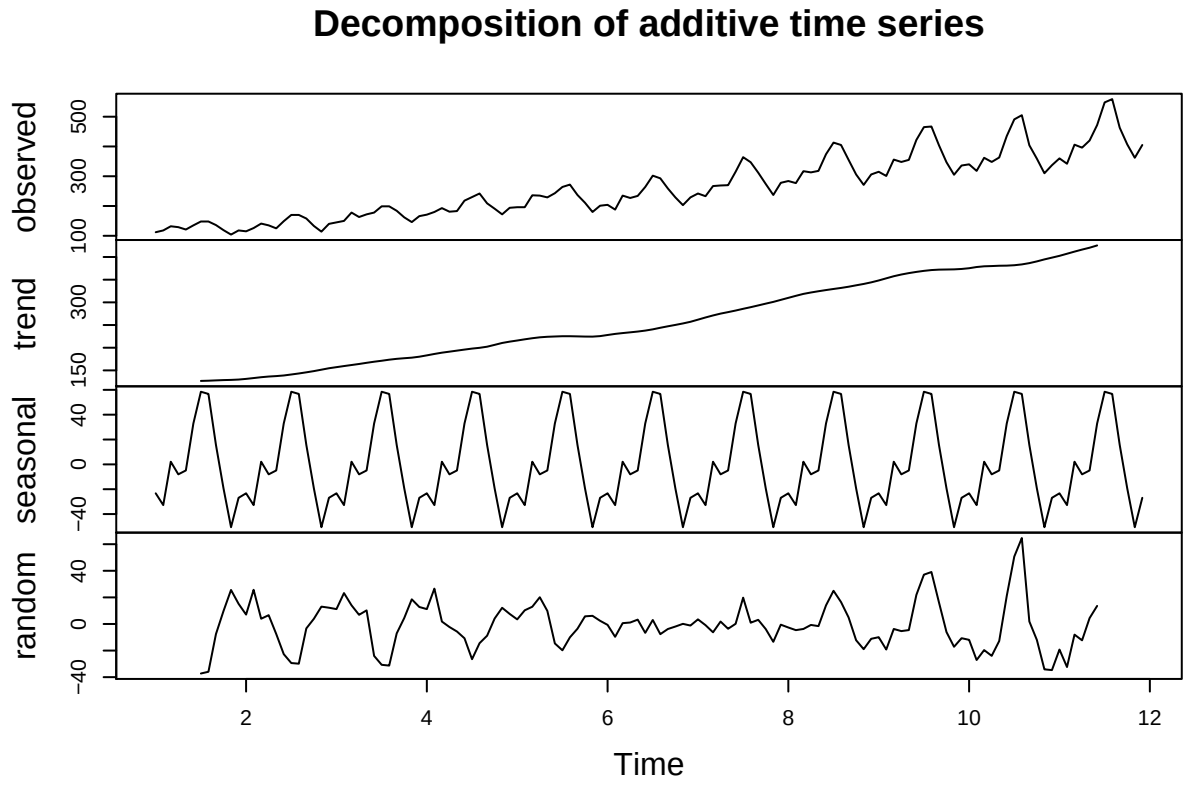
```
serie = scan("Datos/pasajeros_1949_1959.dat")
```

Defino el número de datos que quiero usar para test y para predecir.

```
nTest = 12
nPred = 12
```

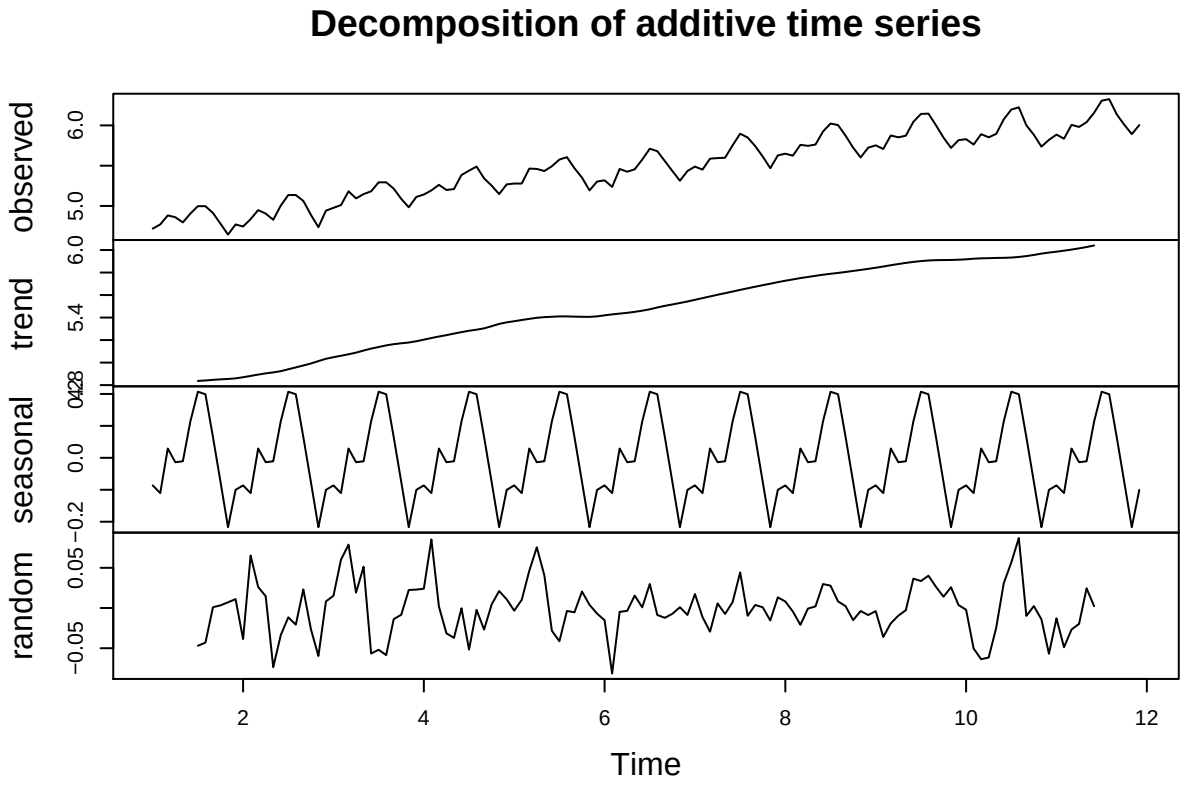
Creo el objeto "Serie temporal" con la librería `tseries` y su función `ts`. La frecuencia será 12 porque suponemos que la estacionalidad es anual. Utilizando la función `plot` junto a `descompose` se pueden ver la tendencia, la estacionalidad y lo que queda de la serie una vez eliminadas la tendencia y la estacionalidad.

```
serie.ts = ts(serie, frequency=12)
plot(decompose(serie.ts))
```



Vamos a construir la serie logarítmica (tanto el objeto ts como la serie original) para tratar de reducir la varianza en la serie resultante de restar la tendencia y estacionalidad. Buscamos que la serie sea estacionaria, y eso implica que no varíe ni en media ni en varianza.

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



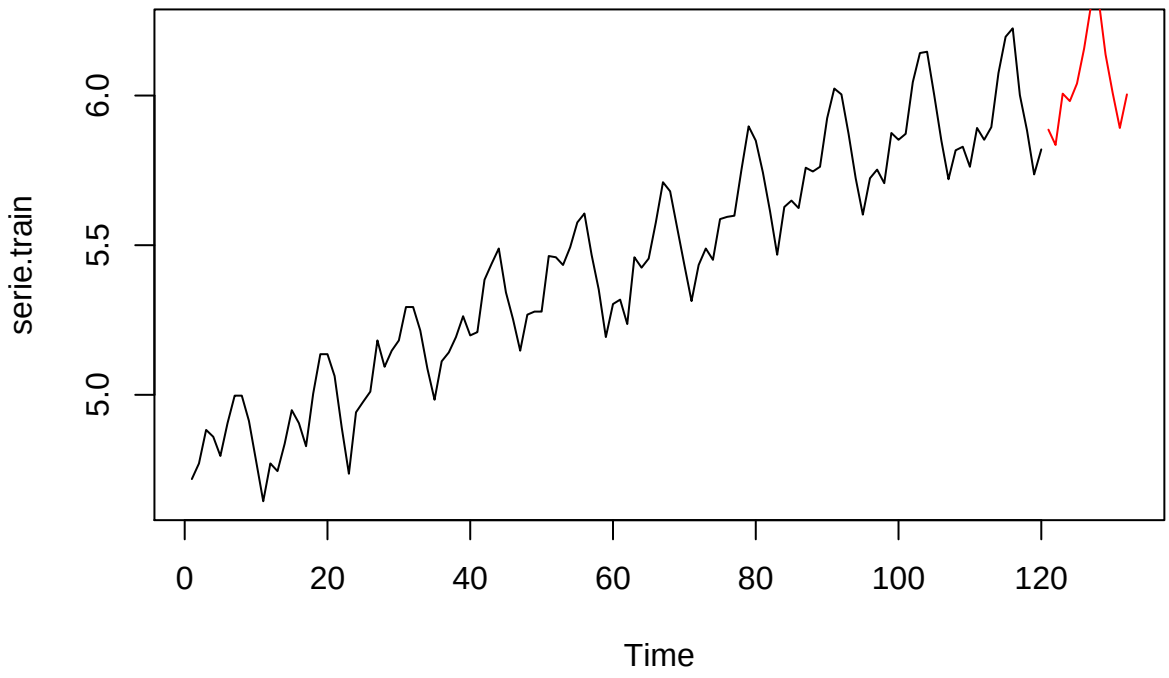
Vamos a seleccionar los datos de train y test del problema, entendiendo test como validación previa a la predicción del año 1960.

```
serie.train = serie.log[1:(length(serie.log) - nTest)]
tiempo.train = 1:length(serie.train)

serie.test = serie.log[(length(serie.log) - nTest+1):length(serie.log)]
tiempo.test = (length(serie.train)+1):length(serie.log)
```

A continuación los mostramos en una gráfica, resaltando en rojo los datos que se están tomando como test.

```
plot.ts(serie.train, xlim=c(1, tiempo.test[length(tiempo.test)]))
lines(tiempo.test, serie.test, col="red")
```



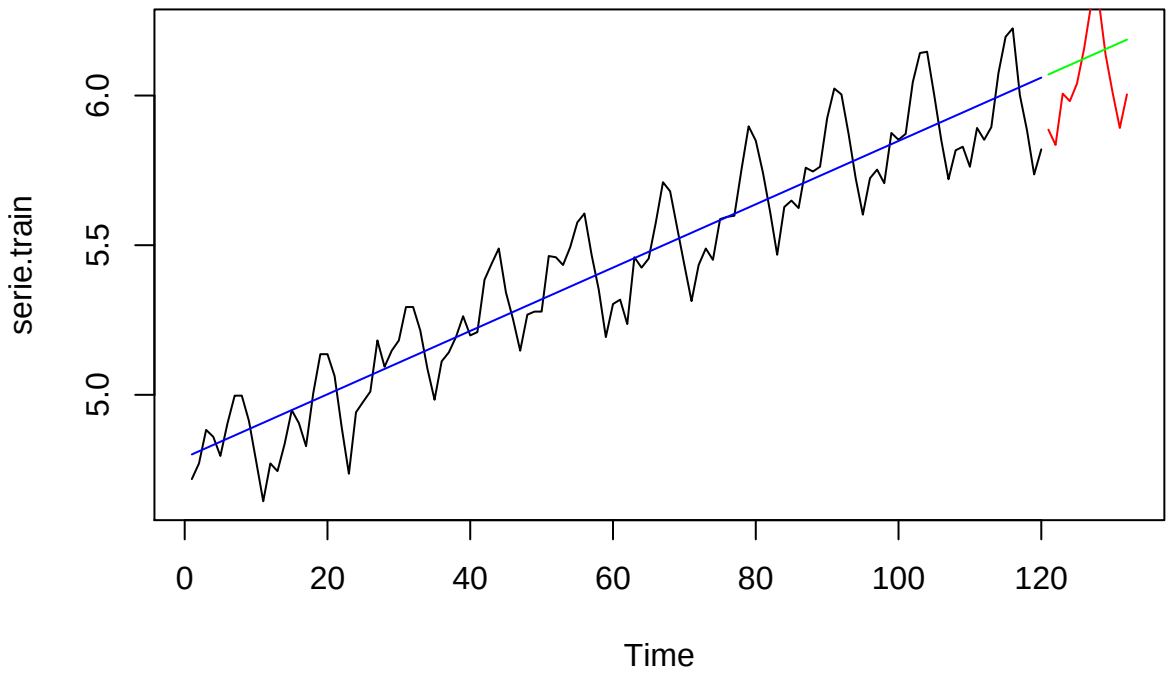
Lo siguiente que se va a hacer es estimar la tendencia. Se va a utilizar un modelo lineal en principio dado que por el aspecto de la serie temporal da la sensación de que puede generalizar bien.

```
parametros.H1 = lm(serie.train ~ tiempo.train)

tendencia.train.H1 = parametros.H1$coefficients[1]+tiempo.train*parametros.H1$coefficients[2]
tendencia.test.H1 = parametros.H1$coefficients[1]+tiempo.test*parametros.H1$coefficients[2]
```

Y a continuación, mostramos la tendencia estimada en la misma gráfica que la serie temporal, para comprobar cómo se ajusta.

```
plot.ts(serie.train, xlim=c(1, tiempo.test[length(tiempo.test)]))
lines(tiempo.test, serie.test, col="red")
lines(tiempo.train, tendencia.train.H1, col="blue")
lines(tiempo.test, tendencia.test.H1, col="green")
```



Vamos a validar que el modelo es correcto, para apoyarnos en algo más que una hipótesis. Utilizaremos el test de Jarque Bera sobre los residuos que han quedado de generar el modelo lineal y los que quedan en el test.

```
JB.train = jarque.bera.test(parametros.H1$residuals)
JB.test = jarque.bera.test(tendencia.test.H1-serie.test)

JB.train
```

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

```
JB.test
```

```
##
##  Jarque Bera Test
##
## data:  tendencia.test.H1 - serie.test
## X-squared = 0.87957, df = 2, p-value = 0.6442
```

Como el p-value resultante del test de Jarque Bera es superior a 0.05 no se puede afirmar con suficiente confianza que los datos no sigan una distribución normal, por lo que consideramos normales los residuos del modelo lineal para train y para test.