Series Temporales: Trabajo Final

Juanjo Sierra

22 de abril de 2019

Paquetes a cargar

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

```
library(tseries)
library(dplyr)
```

```
##
## 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
```

Problema 1

Teniendo los datos de una estación meteorológica (ubicada en Loja) desde mayo de 2013 hasta febrero de 2018, se pide predecir los valores de temperatura máxima mensuales para los meses de marzo y abril de 2018.

En primer lugar, se leen los datos de la estación meteorológica escogida.

```
datosEstacion = read.csv("5582A.csv", sep = ";")
summary(datosEstacion)
```

```
##      Id              Fecha           Tmax           HTmax
##  5582A:1726   2013-05-07:   1   Min.   : 5.80          : 122
##               2013-05-08:   1   1st Qu.:16.38   15:20  :  92
##               2013-05-09:   1   Median :23.80   15:50  :  83
##               2013-05-10:   1   Mean   :24.00   16:00  :  82
##               2013-05-11:   1   3rd Qu.:31.02   15:40  :  78
##               2013-05-12:   1   Max.   :44.30   15:10  :  75
##               (Other)   :1720   NA's   :122     (Other):1194
##       Tmin           HTmin           Tmed           Racha
##  Min.   :-3.70          : 122   Min.   : 2.10   Min.   :14
##  1st Qu.: 5.50   07:20  : 121   1st Qu.:11.00   1st Qu.:27
##  Median :11.40   07:10  : 109   Median :17.60   Median :31
##  Mean   :11.11   07:40  :  99   Mean   :17.56   Mean   :32
##  3rd Qu.:16.30   07:30  :  97   3rd Qu.:23.43   3rd Qu.:36
##  Max.   :27.30   08:00  :  89   Max.   :35.00   Max.   :73
##  NA's   :122     (Other):1089   NA's   :122     NA's   :131
##      HRacha          Vmax           HVmax          TPrec
##          : 131   Min.   : 4.00          : 131   Min.   : 0.000
##  17:50   :  47   1st Qu.:11.00   17:00  :  36   1st Qu.: 0.000
##  17:10   :  32   Median :14.00   17:40  :  33   Median : 0.000
##  15:40   :  31   Mean   :13.82   15:40  :  31   Mean   : 1.069
```

```
##  16:50  :  31   3rd Qu.:16.00   16:50  :  29   3rd Qu.: 0.100
##  18:10  :  31   Max.   :32.00   16:20  :  26   Max.   :63.600
##  (Other):1423   NA's   :131     (Other):1440   NA's   :132
##      Prec1             Prec2             Prec3             Prec4
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.2668   Mean   : 0.2618   Mean   : 0.2543   Mean   : 0.2758
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :31.3000   Max.   :26.7000   Max.   :21.2000   Max.   :32.5000
##  NA's   :88        NA's   :100       NA's   :77        NA's   :83
```

Como solamente vamos a trabajar con la temperatura máxima y necesitamos agruparlas por fecha, nos quedamos únicamente con esas dos columnas.

```
# Nos quedamos con los valores de fecha y temperatura máxima
datosEstacion = datosEstacion[,2:3]
head(datosEstacion)
```

```
##        Fecha Tmax
## 1 2013-05-07 27.8
## 2 2013-05-08 29.3
## 3 2013-05-09 27.1
## 4 2013-05-10 26.6
## 5 2013-05-11 26.8
## 6 2013-05-12 26.1
```

Como hemos comprobado antes en el summary que sí que hay valores perdidos (122), vamos a eliminar las instancias en las que haya un NA para poder calcular correctamente cuál es la temperatura máxima en cada mes.

```
# Elimino aquellos datos que sean NA
datosEstacion = datosEstacion[-which(is.na(datosEstacion$Tmax)),]

# Comprobamos la nueva dimensionalidad de los datos
# y confirmamos que ya no hay datos perdidos
dim(datosEstacion)
```

```
## [1] 1604    2
```

```
anyNA(datosEstacion)
```

```
## [1] FALSE
```

Hemos reducido la dimensionalidad de los datos de 1726 a 1604, pero ya no hay ningún valor perdido y podemos agrupar los datos por mes, obteniendo la temperatura máxima en cada uno.

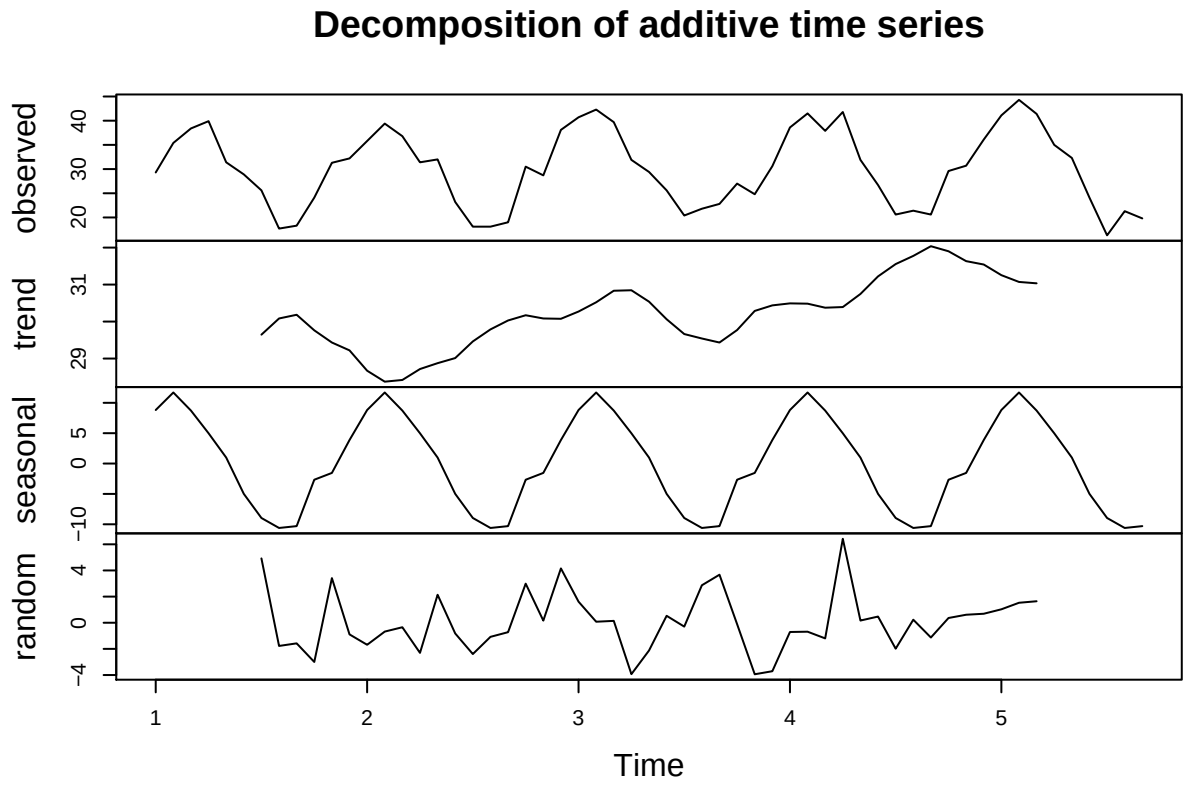
Para obtener dichos datos máximos utilizo la librería `dplyr`.

```
# Almacenamos la variable Fecha como tipo Date
datosEstacion$Fecha = as.Date(datosEstacion$Fecha)

# Convertimos el dataset en dos columnas (Fecha y Mes),
# agrupamos por mes y año y para todos los valores nos
# quedamos con el valor máximo de TMax
datosTmax = datosEstacion %>%
mutate(Mes = format(Fecha, "%m"), Año = format(Fecha, "%Y")) %>%
group_by(Año, Mes) %>%
summarise(Tmax = max(Tmax))
```

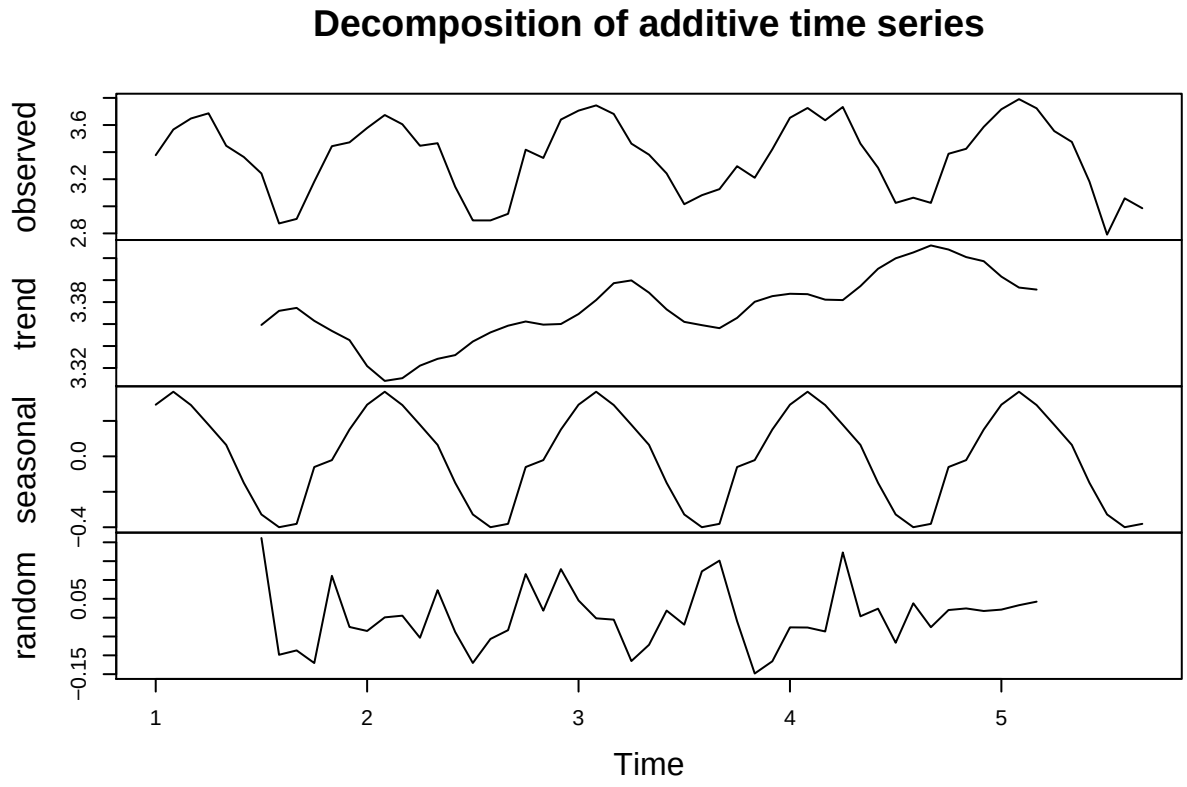
Ahora podemos trabajar con la columna TMax como nuestra serie temporal. Podemos crear el objeto "Serie Temporal" con la librería `tseries`. Usando `plot` y `decompose` se pueden echar un vistazo general al aspecto de nuestros datos. Incluimos un valor de `frequency` de 12 porque es lo que estimamos que es el periodo de la estacionalidad (de año en año y tenemos valores mensuales).

```
# Observamos la tendencia y estacionalidad con el decompose
# y utilizamos frecuencia 12 porque asumimos que tienen
# estacionalidad anual
serie = datosTmax$Tmax
serie.ts = ts(serie, frequency=12)
plot(decompose(serie.ts))
```



Buscando que la serie sea estacionaria, implicando eso que no varíe ni en media ni en varianza, vamos a realizar una transformación logarítmica para conseguir suavizar los valores.

```
# Realizamos una transformación logarítmica
# y volvemos a mostrar los valores del decompose
serie.ts.log = log(serie.ts)
serie.log = log(serie)
plot(decompose(serie.ts.log))
```



Ya podemos empezar a trabajar con los datos. Como primer paso, vamos a asumir que todo el conjunto de train es el total de datos que tenemos y que la predicción a realizar estará compuesta por 2 valores (marzo y abril de 2018).

```
nPred = 2
serie.train = serie.log
tiempo.train = 1:length(serie.train)
tiempo.pred = (length(tiempo.train)+1):(length(tiempo.train)+nPred)
```

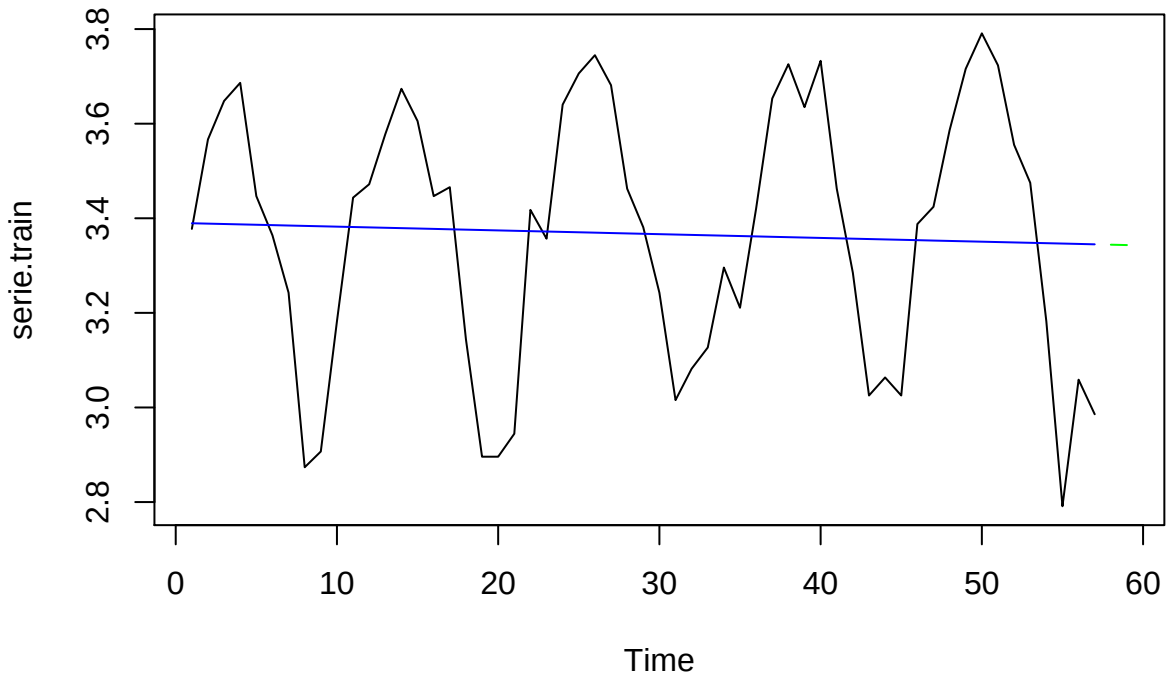
A continuación tenemos que estimar la tendencia. Vamos a utilizar un modelo lineal simple dado que basándonos en la gráfica puede generalizar aceptablemente. Construimos el modelo lineal con la función `lm`.

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

tendencia.train = parametros.lm$coefficients[1]+tiempo.train*parametros.lm$coefficients[2]
tendencia.pred = parametros.lm$coefficients[1]+tiempo.pred*parametros.lm$coefficients[2]
```

En la siguiente gráfica mostramos la tendencia estimada en la misma gráfica que la serie temporal.

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



Para validar que el modelo es correcto, dado que no se puede afirmar con la gráfica que hemos obtenido, utilizaremos el test de Jarque Bera sobre los residuos que han quedado de generar el modelo lineal.

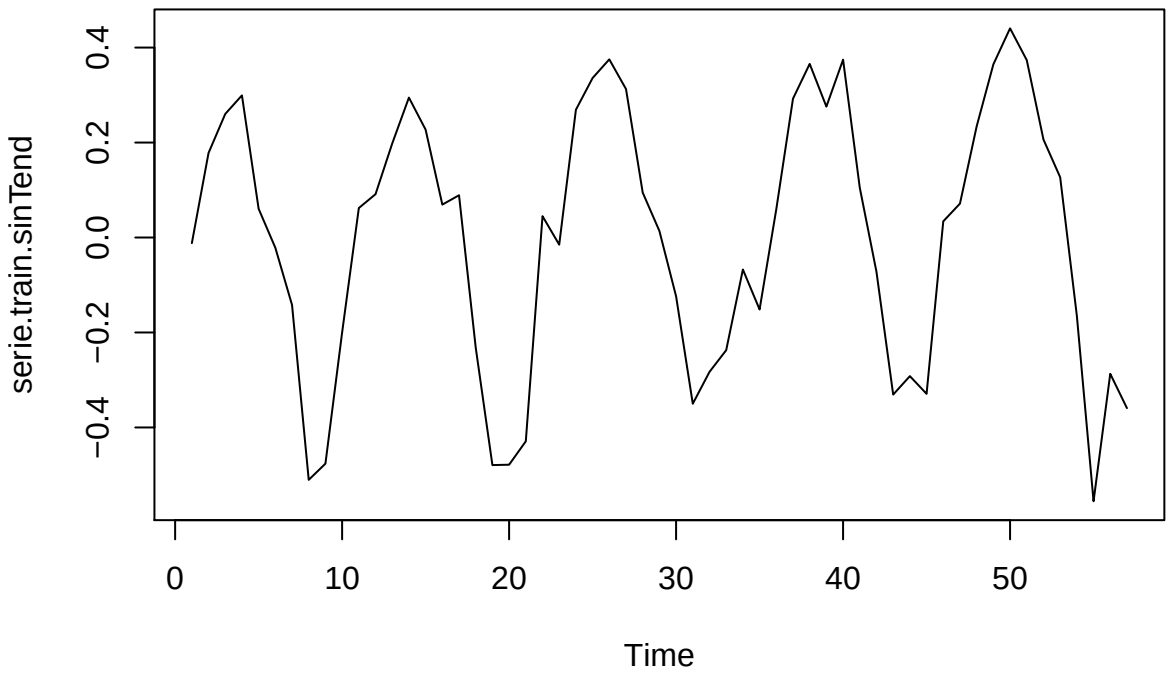
```
JB.train = jarque.bera.test(parametros.lm$residuals)
JB.train
```

```
##
##  Jarque Bera Test
##
## data:  parametros.lm$residuals
## X-squared = 3.3983, df = 2, p-value = 0.1828
```

El test de Jarque Bera da un p-value de 0.18, que está por encima de 0.05. Por este motivo, no tenemos suficiente confianza como para afirmar que los residuos no sigan una distribución normal, y por tanto asumimos que sí que la siguen.

El siguiente paso es eliminar la tendencia a la serie. Comprobaremos en una gráfica qué aspecto tiene una vez eliminada esa tendencia.

```
serie.train.sinTend = serie.train - tendencia.train
plot.ts(serie.train.sinTend, xlim=c(1, tiempo.train[length(tiempo.train)]))
```



Se puede observar que la gráfica tiene el mismo aspecto, solamente se ha trasladado en el eje Y, ubicándose ahora en torno al 0.

El siguiente paso es eliminar la estacionalidad del modelo. Inicialmente supusimos una estacionalidad de frecuencia 12 (anual).

```
k = 12
estacionalidad = decompose(serie.ts.log)$seasonal[1:k]
estacionalidad
```

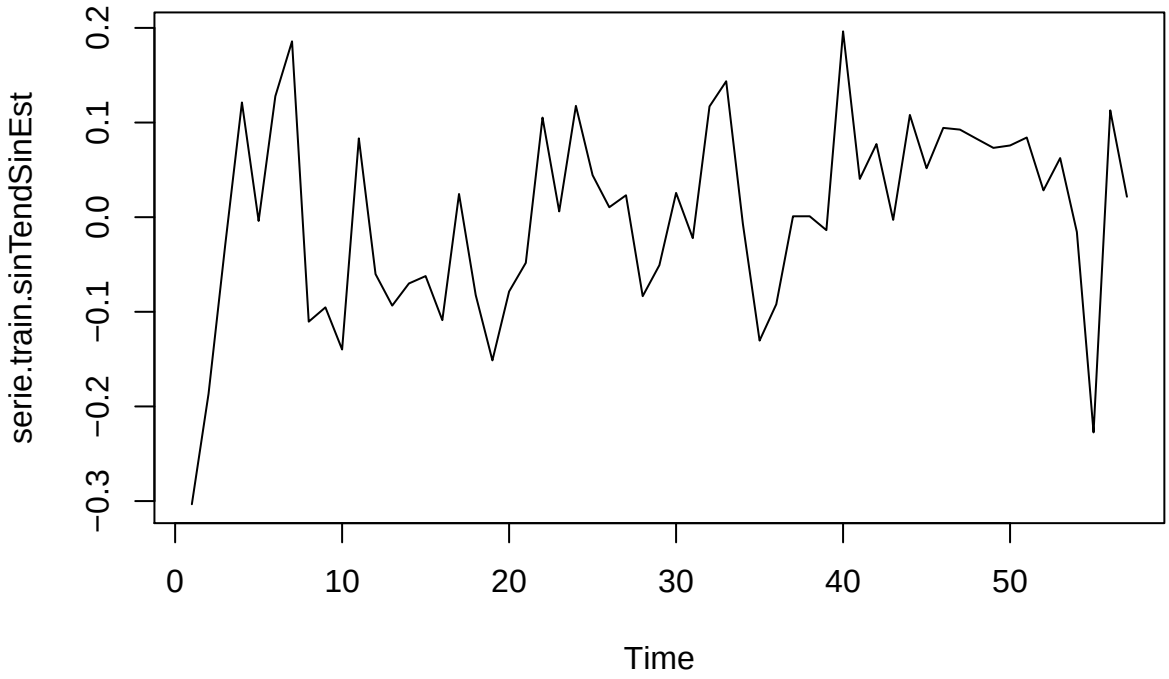
```
##  [1]  0.29147951  0.36470197  0.28941527  0.17808018  0.06450275
##  [6] -0.14945909 -0.32790964 -0.39990923 -0.38087543 -0.06025461
## [11] -0.02116616  0.15139447
```

Aquí vemos los 12 valores que componen la estacionalidad de la serie. Para eliminar la estacionalidad hay que restar estos valores de forma periódica a lo largo de la serie.

```
# Aprovecho el reciclaje de R para no tener que crear una variable auxiliar
serie.train.sinTendSinEst = serie.train.sinTend - estacionalidad
```

```
## Warning in serie.train.sinTend - estacionalidad: longitud de objeto mayor
## no es múltiplo de la longitud de uno menor
```

```
plot.ts(serie.train.sinTendSinEst, xlim=c(1, tiempo.train[length(tiempo.train)]))
```


serie.train.sinTendSinEst
−0.3
−0.2
−0.1
0.0
0.1
0.2
0
10
20
30
40
50
Time