Series temporales en R

Luis Cayuela[1] y Ana Justel[2]

Mayo de 2015

[1]Área de Biodiversidad y Conservación, Universidad Rey Juan Carlos, Departamental 1 – DI. 231, c/ Tulipán s/n. E-28933 Móstoles (Madrid), España. E-mail: luis.cayuela@urjc.es.
[2]Departamento de Matemáticas, Facultad de Ciencias, Módulo CXVI, Despacho 207-B, Universidad Autónoma de Madrid, Campus de Cantoblanco, E-28049 Madrid, España. E-mail: ana.justel@uam.es.

Series temporales en R (versión 1.2)[1]

Publicado por: Luis Cayuela y Ana Justel

Para cualquier comentario o sugerencia por favor remitirse al autor de la obra.

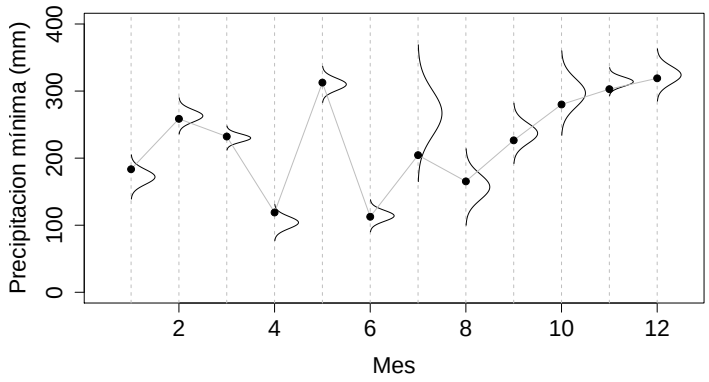
[1] Este material ha sido preparado con el editor de texto Lyx y el programa Sweave.

Índice

1. Introducción a las series temporales

1.1. ¿Qué es una serie temporal?

Las series temporales son un conjunto de observaciones que se registran en intervalos regulares de tiempo. En cada instante, la observación proviene de una variable que puede tener la misma o distinta distribución.



Datos de precipitación mensual a lo largo de 12 meses. Cada observación proviene de una variable (precipitación del mes de enero, precipitación del mes de febrero, precipitación del mes de marzo, etc.) con su propia distribución. En la gráfica se dibujan variables con distribución normal por simplicidad, pero las distribuciones pueden ser de cualquier tipo.

Ejemplos de series temporales son: las temperaturas máximas o mínimas diarias, mensuales o anuales que se recogen en las estaciones meteorológicas, el nivel del agua medio mensual en embalses o ríos, o el número de vehículos que pasan por una carretera diariamente. En las series temporales el orden de llegada de los datos es importante.

Una serie temporal puede tener interés en sí misma para el investigador (por ejemplo para comprobar la ciclicidad de un fenómeno o para ver cómo se ajustan de bien los datos a un determinado modelo teórico), pero a veces es una característica intrínseca del diseño experimental (por ejemplo, en los diseños de medidas repetidas). Los objetivos principales de las series temporales son:

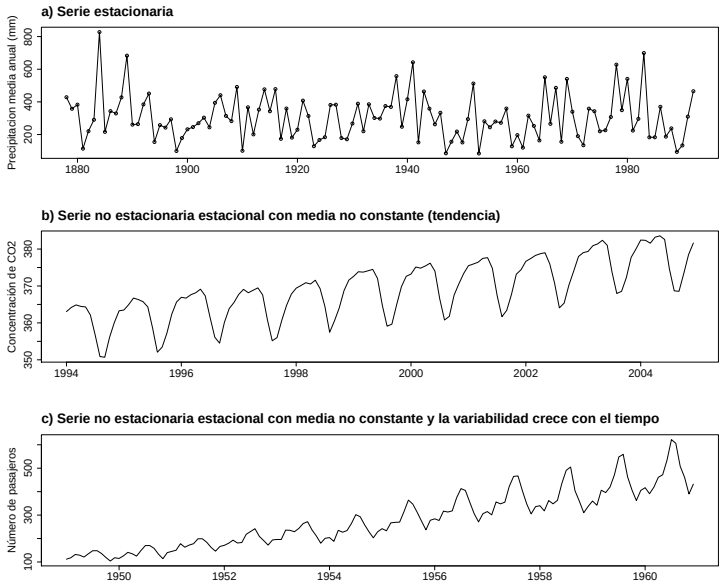
- Explicar la evolución de un fenómeno a lo largo del tiempo.
- Prever sus valores en el futuro.

El conocimiento de la estructura dinámica ayudará a producir predicciones más adecuadas de observaciones futuras y diseñar esquemas de control óptimos.

1.2. Tipos de series

Uno de los principales problemas al que nos enfrentamos en el análisis de series temporales es la falta de información sobre las distribuciones subyacentes (cada una con sus parámetros) de las variables que analizamos. Para cada variable sólo disponemos de un dato observado. Para poder superar este problema es necesario imponer una serie de condiciones a la serie. De esta forma podemos distinguir dos grandes grupos de series:

- Estacionarias - cuando los datos varían todo el tiempo alrededor del mismo valor medio, con la misma variabilidad y la correlación entre dos variables sólo depende del tiempo transcurrido entre ellas (figura a).
- No estacionarias - cuando no se cumple alguna de las condiciones de estacionariedad. Entre las series no estacionarias, las más habituales son las que:
  - Oscilan con periodicidad regular. Las oscilaciones periódicas pueden tener periodicidad anual, mensual, semanal, diaria, horaria, etc. En todos estos casos de series no estacionarias hablaremos de que la serie es **estacional**[2] (figura b).
  - Tienen un comportamiento tendencial (figura b).
  - Crece la variabilidad con el tiempo (figura c).



Serie estacionaria (a), no estacionaria con tendencia y estacionalidad (b) y no estacionaria con tendencia, estacionalidad y variabilidad que crece con el tiempo (c).

[2]En castellano surge muchas veces la confusión entre los términos estacionariedad (*stationarity* en inglés) y estacionalidad (*seasonality* en inglés).

1.3. Series temporales en R

1.3.1. Objetos de clase ts

R es un lenguaje orientado a objetos. Mediante el símbolo de asignación <- creamos objetos que quedan almacenados en la memoria virtual del ordenador y con los que iremos trabajando a lo largo de cada sesión de trabajo.

```
> objeto1 <- matrix(rnorm(15), nrow=5, ncol=3)
> str(objeto1)

 num [1:5, 1:3] 1.753 -1.151 -1.452 2.098 -0.832 ...
```

La función str() nos permite explorar la estructura de cada uno de los objetos que vamos creando en R. Un objeto de clase matrix tendra sus dimensiones [filas, columnas] definidas entre corchetes. Es conveniente utilizar con regularidad esta función para explorar la información contenida en los distintos objetos.

Existen infinidad de tipos de objetos en R. Las matrices, los arreglos de datos (data.frame), los vectores y las listas suelen contener datos de algún tipo. Para almacenar información de variables que cambian con el tiempo existe un tipo de objeto en R que permite definir las características de dicha serie para facilitar su posterior manejo y análisis. Este objeto es de clase ts y puede crearse con la función ts() del paquete stats.

Si tenemos un arreglo de datos con dos columnas, la primera representando una variable temporal, y la segunda representando una variable cualquiera, podemos convertir esta a un objeto de clase ts especificando los argumentos de la función ts() de la forma adecuada.

```
> args(ts)

function (data = NA, start = 1, end = numeric(), frequency = 1,
    deltat = 1, ts.eps = getOption("ts.eps"), class = if (nseries >
        1) c("mts", "ts", "matrix") else "ts", names = if (!is.null(dimnames(data))) colna
        seq(nseries)))
NULL
```

Tomemos por ejemplo, los valores del índice invernal de Oscilación del Atlántico Norte (NAO), un indicador global del clima. El índice NAO está fuertemente relacionado con la temperatura media invernal y con la precipitación en Europa (Hurrel 1995[3], Bojariu & Giorgi 2005[4]). Valores positivos del índice NAO representaría condiciones de alta precipitación en el

[3]Hurrell, J.W. 1995. Decadal trends in the North Atlantic Oscillation: regional temperatures and precipitation. *Science* 269: 676-679.

[4]Bojariu, R. & Giorgi, F. 2005. The North Atlantic Oscillation signal in a regional climate simulation for the European region. *Tellus series A-Dynamic Meteorology and Oceanography* 57: 641-653.

Norte de Europa, baja precipitación en el Sur de Europa y en la región Mediterránea, y un incremento de las temperaturas en el Centro y Norte de Europa. Valores negativos de este índice estarían asociados a temperaturas cálidas y condiciones húmedas en el Sur de Europa y, de forma específica, en la Península Ibérica. Vamos a leer los datos de un archivo *.txt disponible en internet. En lugar de descargarnos los datos al disco duro y leerlos localmente, vamos a leerlos directamente utilizando la URL como argumento principal de la función read.table().

```
> NAO <- read.table("http://www.escet.urjc.es/biodiversos/R/NAO.txt", header=T, sep="\t")
> str(NAO)
```

```
'data.frame':        146 obs. of  2 variables:
 $ year: int  1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 ...
 $ NAO : num  -1.02 -1.24 0.54 -1.38 2.81 1.7 -3.01 -1.01 -0.76 -0.5 ...
```

Para convertir este arreglo de datos en un objeto de clase ts, tendríamos que utilizar el siguiente código:

```
> NAO.ts <- ts(data=NAO$NAO, start=min(NAO$year), end=max(NAO$year))
> class(NAO.ts)
```

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

```
> str(NAO.ts)
```

```
 Time-Series [1:146] from 1864 to 2009: -1.02 -1.24 0.54 -1.38 2.81 1.7 -3.01 -1.01 -0.76
```

Esta sería la forma más sencilla de crear un objeto de clase ts, en donde tendríamos datos anuales de valores del índice NAO, en una serie continua de datos que comenzaría en el año 1864 y acabaría en el año 2009. Si hubiera datos faltantes hay que especificarlo de forma explícita en el archivo de datos de la serie con un espacio en blanco o un NA.

Pero también podría ser que nuestros datos representan valores mensuales. En este caso debemos especificar un nuevo argumento en la función ts(), frequency, que determinaría el número de valores que hay por año, y el argumento start como un vector numérico[5] en dónde el primer dígito haría referencia al año y el segundo dígito haría referencia al mes. Recordemos que un vector numérico puede crearse fácilmente con la función c(). Si la frecuencia que especificamos es 4 o 12 R reconoce por defecto que se trata de trimestres (*quarter*) o meses, respectivamente, y activa las etiquetas correspondientes.

```
> ts(1:10, frequency = 4, start = c(1959, 2))
```

[5]La mayor parte de las veces no es necesario definir el argumento end.

```
     Qtr1 Qtr2 Qtr3 Qtr4
1959         1    2    3
1960    4    5    6    7
1961    8    9   10
```

```
> ts(rnorm(30), frequency = 12, start = c(12, 2))
```

```
           Jan         Feb         Mar         Apr         May         Jun
12              0.38509127  0.20310778  1.50120042  0.28563632 -0.36392785
13 -0.36634099 -0.09103382 -0.93451017  1.45030750 -0.59609393 -0.51317786
14 -0.03904962 -1.04536833  1.39214187  0.05540222  2.01179814 -0.30105319
           Jul         Aug         Sep         Oct         Nov         Dec
12  0.38310963 -0.09977296 -0.03523310 -0.17866790 -0.05168806 -1.16394033
13 -0.48923138 -0.77627644  0.47964286  1.69063618 -1.52281757 -1.30057281
14  1.75745361
```

A veces vamos a tener, no una, sino varias variables, medidas a lo largo del tiempo. En este caso, podemos generar un objeto de clase mts, que es exactamente igual a los objetos de clase ts, salvo que estructura la información en columnas, correspondiendo cada columna a una variable que cambia en el tiempo. Para extraer la información de una de estas variables podemos utilizar los operadores de asignación típicamente usados para matrices y arreglos de datos. De esta forma podemos convertir fácilmente un objeto de clase mts a ts y viceversa.

```
> z <- ts(matrix(rnorm(30), 10, 3), start=c(1961, 1), frequency=12)
> class(z)
```

```
[1] "mts"    "ts"     "matrix"
```

```
> z
```

```
            Series 1    Series 2   Series 3
Jan 1961 -0.49772671 -0.80606062  2.2293974
Feb 1961  0.12344858  1.10610358 -0.7357941
Mar 1961  1.21691335  0.00199704  1.0716815
Apr 1961 -0.86789170  1.34462690 -1.0557685
May 1961 -0.97627472 -0.57289951  0.4705938
Jun 1961  0.09620355 -0.26299865  2.7862778
Jul 1961 -0.65070227  0.15286764 -1.5825161
Aug 1961 -0.12745844 -0.37221355 -0.4659964
Sep 1961  0.16157079 -1.06633034 -1.1065516
Oct 1961 -1.35177138  0.68693158 -0.4086995
```

```
> zp <- z[,2]
> class(zp)
```

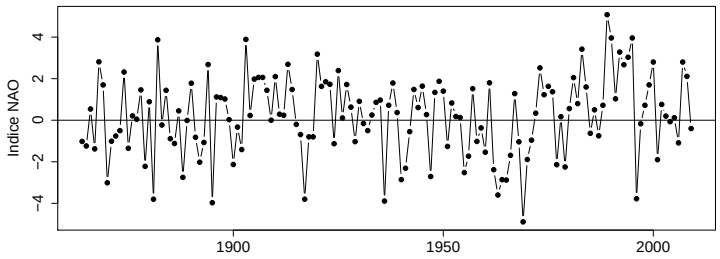
```
[1] "ts"
```

Finalmente, mencionar que hay otras clases de objetos que permiten trabajar con series temporales, si bien la clase ts es la que se utiliza más comúnmente. Las funciones its(), irts() y fts() de los paquetes its, tseries y fts respectivamente, permiten crear objetos de clase its, irts y fts para la implementación de series temporales irregulares. Se puede encontrar más información sobre clases de objetos para series temporales en CRAN Task View: Time Series Analysis.

1.3.2. Representación gráfica de series temporales en R

La representación más habitual de las series temporales es el gráfico temporal. Se representan los valores de las variables en el eje de ordenadas, consecutivamente y respetando el orden de llegada de los datos. Se suelen unir los puntos con rectas. Es importante que manejemos con cierta soltura los argumentos de las funciones gráficas para poder personalizar los gráficos que obtengamos. Para ello se puede consultar la ayuda de la función par().

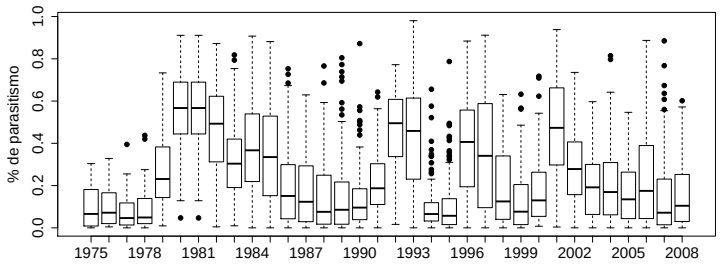
```
> par(cex.lab=1.5, cex.axis=1.5)
> plot(NAO~year, data=NAO, type="b", pch=19, ylab="Índice NAO", xlab="")
> abline(h=0)
```



Valor del índice NAO invernal desde 1864 a 2009.

En ocasiones se usan barras, áreas, etc., que suelen dificultar la apreciación de las características más relevantes de las series.

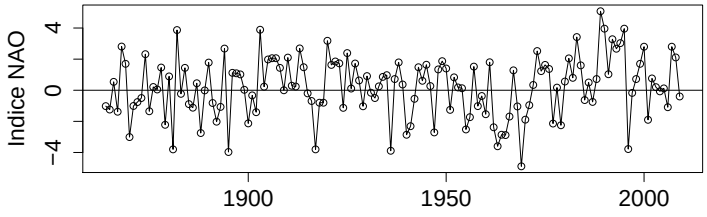
```
> parasit <- read.table("http://www.escet.urjc.es/biodiversos/R/parasitismo.txt", header=T
> par(cex.lab=1.5, cex.axis=1.5, pch=19)
> parasit$Yr2 <- as.factor(parasit$Yr)
> plot(Parasitismo~Yr2, data=parasit, type="b", ylab="% de parasitismo", xlab="")
```



Porcentaje de parasitismo en puestas de procesionaria en Mora de Rubielos, Teruel.

No obstante, estas representaciones no son necesarias si utilizamos los paquetes disponibles en R para el tratamiento y análisis de series temporales, los cuales suelen tener sus propias funciones gráficas para representación de datos temporales. La misma figura del ejemplo anterior la obtendríamos utilizando directamente un objeto de clase ts como argumento principal de la función plot().

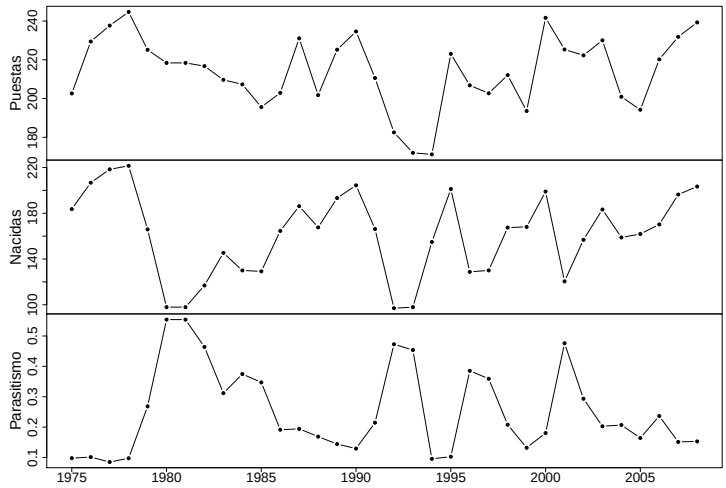
```
> par(cex.axis=1.5, cex.lab=1.5, cex=1.5)
> plot(NAO.ts, type="o", ylab="Índice NAO", xlab="")
> abline(h=0)
```



Gráfica del valor del índice NAO invernal desde 1864 a 2009 utilizando un objeto de clase ts como argumento principal de la función plot().

En realidad, la función plot(), que es una función genérica, actúa aquí como la función plot.ts(), cuya ayuda puede ser consultada para ver más detalles sobre los argumentos que pueden usarse. Si se utiliza como argumento principal un objeto de clase mts entonces lo que se obtiene es la representación gráfica de todas las variables que hay en ese objeto.

```
> parasit.mean <- aggregate(parasit[,1:3],list(parasit$Yr),mean)
> str(parasit.mean)

'data.frame':        34 obs. of  4 variables:
 $ Group.1    : int  1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 ...
 $ Puestas    : num  203 229 238 245 225 ...
 $ Nacidas    : num  184 207 218 222 166 ...
 $ Parasitismo: num  0.0977 0.1012 0.085 0.0974 0.2684 ...

> parasit.ts <- ts(parasit.mean[,2:4], start=min(parasit.mean$Group.1))


> par(cex=3, cex.lab=1.5, cex.axis=2)
> plot(parasit.ts, type="b", pch=19, xlab="", main="")
```



Promedio del tamaño de puesta (Puestas), número de larvas nacidas (Nacidas) y proporción de parasitismo en puestas de procesionaria en Mora de Rubielos, Teruel.

Especificando el argumento plot.type="single" conseguiremos que todas las series temporales queden representados en la misma gráfica, aunque esto no es siempre conveniente para la representación de nuestros datos.

1.3.3. Paquetes para el tratamiento de series temporales en R

Existen muchas funciones para el tratamiento y análisis de series temporales dentro del paquete stats. Otros paquetes interesantes son el paquete tseries y el paquete TSA, que utilizaremos en este curso. Para instalarlos, deberemos escribir en la consola el siguiente código:

```
> install.packages(c("TSA", "tseries", "forecast"), dep=T)
```

Aparecerá una ventana preguntándonos de qué repositorio queremos bajarnos los paquetes (mirror). Hemos de elegir uno que esté geográficamente cerca para que la descarga no tarde mucho. El de España funciona más o menos bien, pero también puede ser cualquiera de los que hay en Francia o Reino Unido. Una vez que aceptemos comenzará la descarga. Cuando estén los paquetes instalados, deberemos cargarlos al comienzo de cada sesión si queremos utilizar algunas de las funciones contenidas en estos paquetes.

```
> library(TSA)
> library(tseries)
> library(forecast)
```

Existe además una interfaz gráfica de R bastante nueva para el tratamiento de series temporales llamada TTAinterface. Está escrita para la versión 2.14+ de R por lo que será necesario actualizar la versión de R si tenemos instalada una versión anterior. Esta interfaz permite realizar análisis de series temporales por medio de una interfaz gráfica amigable. Para instalar esta interfaz podemos escribir en la consola el siguiente código:

```
> install.packages("TTAinterfaceTrendAnalysis", dep=T)
```

Más información sobre esta interfaz en TTAinterface.

Se puede encontrar toda una lista detallada de paquetes que contienen funciones relacionadas con el tratamiento de series temporales en CRAN Task View: Time Series Analysis. Como siempre pasa en R, pueden existir dos o más funciones dentro de distintos paquetes que hagan lo mismo o cosas parecidas, por lo que conviene tener cuidado y leer bien las páginas de ayuda de las funciones antes de usarlas. También pueden consultarse los siguientes vínculos, que contienen tutoriales e información sobre el uso de series temporales en R:

- RBloggers.
- Time Series Data Library.

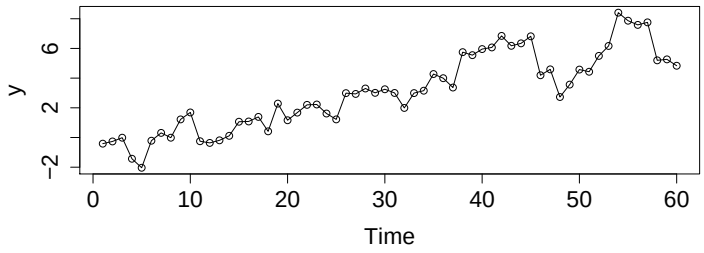
2. Modelos lineales: una primera aproximación para el análisis de series temporales

Una primera aproximación al análisis de una serie temporal es ajustar un modelo lineal. Estos modelos imponen un comportamiento determinista a la serie.

2.1. Modelar la tendencia

Tomemos como primer ejemplo, unos datos con tendencia, que intentaremos modelar con un modelo de regresión. Estos datos están contenidos en el objeto rwalk del paquete TSA.

```
> library(TSA)
> data(rwalk)
> par(cex.axis=1.5, cex.lab=1.5, cex=1.5)
> plot(rwalk, ylab="y", type="o")
```



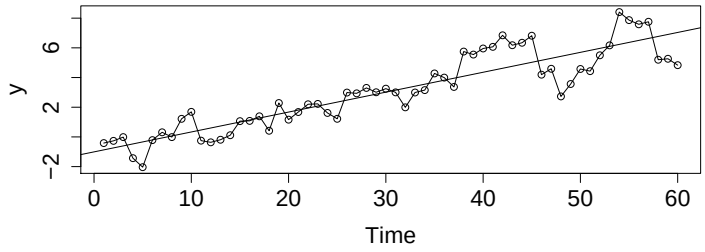
Proceso de "random walk".

Para ajustar la recta de regresión utilizaremos la función lm().

```
> lm.rwalk <- lm(rwalk~time(rwalk))
> summary(lm.rwalk)

Call:
lm(formula = rwalk ~ time(rwalk))

Residuals:
     Min       1Q   Median       3Q      Max
-2.70045 -0.79782  0.06391  0.63064  2.22128

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.007888   0.297245  -3.391  0.00126 **
time(rwalk)  0.134087   0.008475  15.822  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.137 on 58 degrees of freedom
Multiple R-squared:  0.8119,        Adjusted R-squared:  0.8086
F-statistic: 250.3 on 1 and 58 DF,  p-value: < 2.2e-16
```

La recta de regresión está definida por dos parámetros: el intercepto y la pendiente. En este caso el intercepto vale -1.008 y la pendiene 0.1341. Se puede representar la recta de regresión sobre el gráfico con la función gráfica de bajo nivel abline().

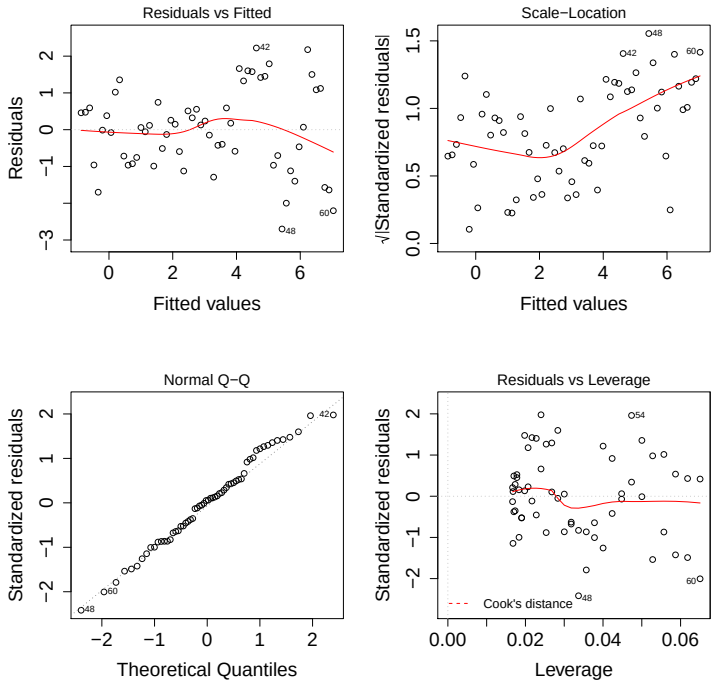
```
> par(cex.axis=1.5, cex.lab=1.5, cex=1.5)
> plot(rwalk, ylab="y", type="o")
> abline(lm.rwalk)
```



Proceso de "random walk" con un ajuste lineal de la tendencia.

¿Es el modelo adecuado? Echemos un vistazo a los gráficos de los residuos.

```
> par(mfcol=c(2,2), cex.axis=1.5, cex.lab=1.5)
> plot(lm.rwalk)
```



Residuos del modelo lineal lm.rwalk.

Aparentemente todo está bien. Pero, mostremos el gráfico de los residuos

frente al tiempo.

```
> par(cex.axis=1.5, cex.lab=1.5, cex=1.5)
> res <- residuals(lm.rwalk)
> plot(res ~ as.vector(time(rwalk)), ylab="residuos", xlab="Tiempo", type="o")
> abline(h = 0)
```

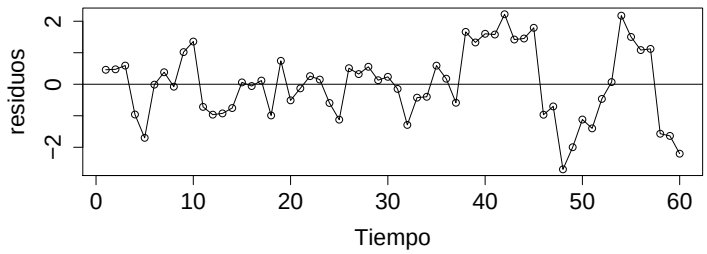


Gráfico de los residuos de un modelo lineal frente al tiempo.

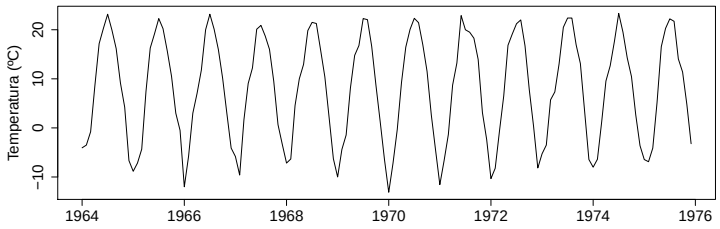
En esta gráfica, los residuos en intervalos próximos de tiempo tienden a tomar valores parecidos, algo que es raro que ocurra por azar. Otra cosa que se observa es que parece que hay más variación en el último tercio de la serie en comparación con los dos primeros tercios.

2.2. Modelar la estacionalidad

Consideremos ahora la estimación de comportamientos estacionales mediante el uso de modelos lineales. Esto se haría calculando los promedios de la variable respuesta en cada una de las estaciones. Si se tratasen de datos mensuales, estimaríamos 12 parámetros, que serían constantes que representarían el valor promedio de la variable para cada uno de los 12 meses.

Consideremos como ejemplo los datos de temperatura media mensual (convertida de grados Farenheit a grados Celsius) registrados durante un periodo de años en Dubuque, Iowa.

```
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> data(tempdub)
> plot(((tempdub-32)*5)/9, ylab="Temperatura (ºC)", xlab="")
```



Evolución temporal de la temperatura media mensual (en grados Celsius) en Dubuque, Iowa.

Podemos ajustar un modelo lineal a estos datos, una vez más, con la función lm(). Deberemos previamente obtener el nombre de cada mes para cada una de las observaciones. Esto podemos hacerlo fácilmente con la función season() del paquete TSA.

```
> month <- season(tempdub)
> lm.temp <- lm(tempdub~month)
> summary(lm.temp)

Call:
lm(formula = tempdub ~ month)

Residuals:
    Min      1Q  Median      3Q     Max
-8.2750 -2.2479  0.1125  1.8896  9.8250

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)     16.608      0.987  16.828  < 2e-16 ***
monthFebruary    4.042      1.396   2.896  0.00443 **
monthMarch      15.867      1.396  11.368  < 2e-16 ***
monthApril      29.917      1.396  21.434  < 2e-16 ***
monthMay        41.483      1.396  29.721  < 2e-16 ***
monthJune       50.892      1.396  36.461  < 2e-16 ***
monthJuly       55.108      1.396  39.482  < 2e-16 ***
monthAugust     52.725      1.396  37.775  < 2e-16 ***
monthSeptember  44.417      1.396  31.822  < 2e-16 ***
monthOctober    34.367      1.396  24.622  < 2e-16 ***
monthNovember   20.042      1.396  14.359  < 2e-16 ***
monthDecember    7.033      1.396   5.039 1.51e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 3.419 on 132 degrees of freedom
Multiple R-squared:  0.9712,        Adjusted R-squared:  0.9688
F-statistic: 405.1 on 11 and 132 DF,  p-value: < 2.2e-16
```

El intercepto sería la estimación para el mes de enero y cualquiera de los otros coeficientes hace referencia a cómo la temperatura promedio aumenta o disminuye con respecto al mes de enero. Podemos estimar un modelo sin intercepto, que es más fácilmente interpretable en este contexto, de la siguiente forma:

```
> lm.temp2 <- lm(tempdub~month-1)
> summary(lm.temp2)

Call:
lm(formula = tempdub ~ month - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-8.2750 -2.2479  0.1125  1.8896  9.8250

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
monthJanuary     16.608      0.987   16.83   <2e-16 ***
monthFebruary    20.650      0.987   20.92   <2e-16 ***
monthMarch       32.475      0.987   32.90   <2e-16 ***
monthApril       46.525      0.987   47.14   <2e-16 ***
monthMay         58.092      0.987   58.86   <2e-16 ***
monthJune        67.500      0.987   68.39   <2e-16 ***
monthJuly        71.717      0.987   72.66   <2e-16 ***
monthAugust      69.333      0.987   70.25   <2e-16 ***
monthSeptember   61.025      0.987   61.83   <2e-16 ***
monthOctober     50.975      0.987   51.65   <2e-16 ***
monthNovember    36.650      0.987   37.13   <2e-16 ***
monthDecember    23.642      0.987   23.95   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.419 on 132 degrees of freedom
Multiple R-squared:  0.9957,        Adjusted R-squared:  0.9953
F-statistic:  2569 on 12 and 132 DF,  p-value: < 2.2e-16
```

Aunque la interpretación es ligeramente distinta, los resultados son idénticos en ambos casos. Vamos a ver si hay algún patrón en los residuos:

```
> par(mfcol=c(2,1), cex.axis=1.5, cex.lab=1.5, cex=1.5)
> res <- residuals(lm.temp2)
> temp <- as.vector(time(tempdub))
> fit <- fitted(lm.temp2)
> labs <- as.vector(month)
> plot(res~temp, ylab="Residuos", xlab="Tiempo", type="o", pch=labs)
> abline(h = 0)
> labs <- as.vector(season(tempdub))
> plot(res~fit, xlab="Valores predichos", ylab="Residuos", pch=labs)
```

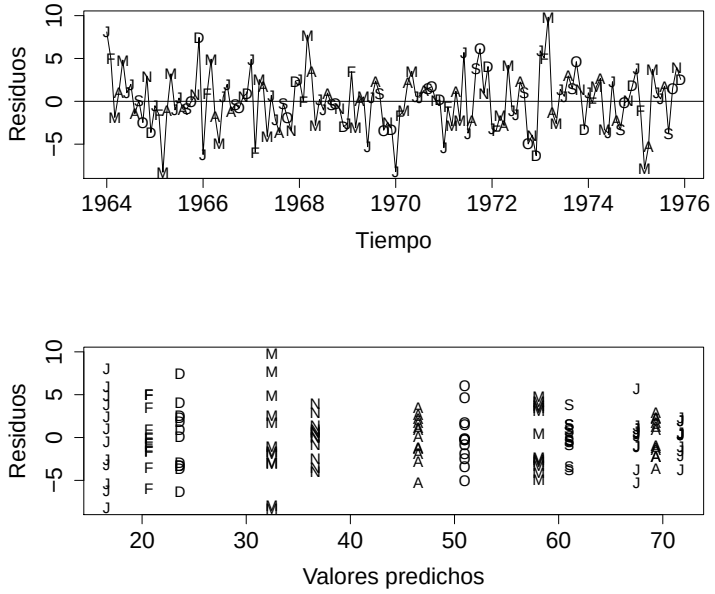


Gráfico de los residuos de un modelo lineal frente al tiempo y frente a los valores predichos (J = January, F = February, M = March, A = April, M = May, J = June, J = July, A = August, S = September, O = October, N = November, D = December).

Así hemos podido extraer la estacionalidad y tenemos una estimación de los coeficientes de estacionalidad. Podemos realizar un test específico para comprobar la independencia de los residuos, si queremos estar más seguro de ello. Para ello se puede utilizar la función runs(), que examina los residuos de forma secuencial para ver si hay patrones de algún tipo, que evidenciarían falta de independencia. La hipótesis nula en este caso es que los residuos son independientes, por lo que un *p-valor* por encima del nivel crítico de significación (típicamente $\alpha = 0{,}05$), indicaría que los residuos son independientes, como es el caso.

```
> runs(rstudent(lm.temp2))

$pvalue
[1] 0.216
```

```
$observed.runs
[1] 65

$expected.runs
[1] 72.875

$n1
[1] 69

$n2
[1] 75

$k
[1] 0
```

Datos que incluyesen tendencia y estacionalidad de forma simultánea podrían ser modelados mediante modelos lineales utilizando factores y covariables (análisis de tipo ANCOVA).

2.3. Limitaciones de los estimadores de modelos lineales

El ajuste de datos temporales mediante el uso de modelos lineales, como se ha visto en la sección anterior, siempre es posible. Sin embargo, estos modelos ofrecen varias limitaciones, entre las que destacaríamos:

1. Las tendencias deterministas que imponen los modelos lineales no son siempre buenos descriptores de la evolución de un proceso a lo largo del tiempo y condicionan la predicción.
2. Los estimadores de modelos lineales dan más peso a los valores centrales, cuando para la predicción los valores más relevantes en general deben ser los más próximos al final de la serie.
3. Con datos de series temporales frecuentemente se viola el supuesto de independencia de las observaciones. Esto afecta a las propiedades de los estimadores, si bien los modelos pueden utilizarse a nivel descriptivo.

3. Alisado exponencial

El método del alisado exponencial es una aproximación determinista al tratamiento de series temporales. Los alisados se emplean para predecir nuevos valores de la serie. Se basan en modelos paramétricos deterministas que se ajustan a la evolución de la serie. Estos modelos permiten ajustar niveles y comportamientos tendenciales y estacionales que evolucionan en el tiempo, de manera que las observaciones más recientes tienen más peso en el ajuste que las más alejadas. Aunque el alisado exponencial puede implementarse fácilmente en R sin tener un conocimiento matemático muy profundo. No siempre es un

modelo útil para explicar la evolución de una serie temporal. Tampoco se hace inferencia, ya que no se asume una estructura estocástica de las observaciones.

Hay tres tipos de alisado exponencial:

1. **Alisado exponencial simple** - Se emplea para series sin tendencia ni estacionalidad. El modelo del alisado exponencial simple depende de un parámetro, *alpha*, que modula la importancia que tienen las observaciones pasadas sobre el presente. Su valor oscila entre 0 y 1. Si *alpha* toma un valor próximo a 0 las predicciones a lo largo de la serie son muy similares entre sí, y se modifican poco con la nueva información. El caso extremo se produce cuando *alpha* es cero, lo que implicaría que la predicción es una constante a lo largo del tiempo. Si *alpha*, por el contrario, toma un valor próximo a 1 la predicción se va adaptando al último valor observado, por lo que se puede decir que los valores alejados en el tiempo no tienen mucha influencia sobre la predicción.

2. **Alisado de Holt** - Se emplea para series con tendencia y sin estacionalidad. El modelo depende de dos parámetros, *alpha* y *beta*. El parámetro *beta* modula la importancia que tienen las observaciones pasadas sobre la pendiente estimada en tiempo *t*. Al igual que para *alpha*, los valores de *beta* oscilan entre 0 y 1. Si *beta* toma un valor próximo a 0 entonces la pendiente es constante o casi constante, es decir, cambia poco a lo largo de la serie temporal. Si *beta* toma un valor próximo a 1, la predicción de la pendiente se va adaptando al último valor observado y las observaciones de las pendientes más alejadas en el tiempo no tienen apenas influencia sobre la predicción.

3. **Alisado de Holt-Winters** - Se emplea para series con tendencia y estacionalidad. Depende de tres parámetros, *alpha*, *beta* y *gamma*. El parámetro *gamma* modula la importancia que tienen las observaciones hechas en el mismo periodo de tiempos pasados sobre la predicción en tiempo *t*. *Gamma* oscila entre 0 y 1. Si *gamma* es 0 la predicción en tiempo t va a tomar un valor constante que va a depender de todas las observaciones pasadas dentro de ese mismo periodo. Si *gamma* es 1 la prediccion en tiempo t va a depender solamente de la observación hecha en tiempo t-p, siendo p la frecuencia (por ejemplo $t = 12$ para observaciones mensuales).

En R se pueden realizar los tres tipos de alisados con una única función: la función HoltWinters() del paquete stats.

```
> args(HoltWinters)
```

```
function (x, alpha = NULL, beta = NULL, gamma = NULL, seasonal = c("additive",
    "multiplicative"), start.periods = 2, l.start = NULL, b.start = NULL,
    s.start = NULL, optim.start = c(alpha = 0.3, beta = 0.1,
        gamma = 0.1), optim.control = list())
NULL
```

El argumento principal de esta función, x, son los datos con los que vamos a realizar el alisado y tiene que tratarse de un objeto de clase ts. Podemos fijar apriori el valor de los parámetros *alpha*, *beta* y *gamma* o estimarlos minimizando para ello el error cuadrático medio. Para realizar un alisado exponencial simple, deberemos especificar que beta=FALSE y gamma=FALSE. Si queremos realizar un alisado de Holt, deberemos de especificar que gamma=FALSE. Para llevar a cabo un alisado de Holt-Winter no hay que cambiar ninguno de estos argumentos. Hay básicamente dos tipos de funciones de predicción para la componente estacional (y por tanto solo aplicable al alisado de Holt-Winter): el modelo aditivo (implementado por defecto) y el modelo multiplicativo. El modelo multiplicativo sería más adecuado cuando la amplitud del ciclo va cambiando en el tiempo.

3.1. Alisado exponencial simple

Vamos a empezar implementado un alisado exponencial sobre los datos del índice NAO invernal que utilizamos anteriormente. Por si acaso no lo hemos hecho antes, volveremos a leer los datos en R.

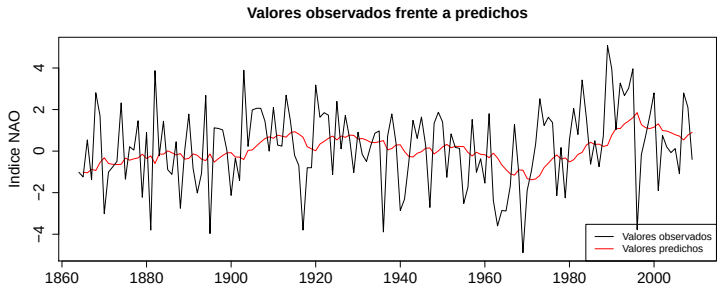
```
> NAO <- read.table("http://www.escet.urjc.es/biodiversos/R/NAO.txt", header=T, sep="\t")
> NAO.ts <- ts(data=NAO$NAO, start=min(NAO$year), end=max(NAO$year))
> alisim1 <- HoltWinters(NAO.ts, gamma=FALSE, beta=FALSE)
> alisim1

Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = NAO.ts, beta = FALSE, gamma = FALSE)

Smoothing parameters:
 alpha: 0.1020395
 beta : FALSE
 gamma: FALSE

Coefficients:
       [,1]
a 0.7714705
```

¿Cómo se interpreta este resultado? Vemos que hay dos apartados: parametros de suavizado (*Smoothing parameters*) y coeficientes (*Coefficients*). Los parámetros de suavizado son los parámetros *alpha*, *beta* y *gamma* que ya comentamos anteriormente. Al estar realizando un alisado simple, los parámetros *beta* y *gamma* no han sido estimados. El parámetro *alpha* que minimiza el error cuadrático medio es 0.102, muy próximo a cero. Esto nos indica que de acuerdo al modelo, la predicción en tiempo t es el resultado de las observaciones hechas en instantes anteriores teniendo las observaciones más alejadas en el tiempo una influencia importante sobre dicha predicción. Esto se traduce en que nuestra predicción será un valor casi constante a lo largo del

tiempo. El coeficiente $a = 0.771$ representa el valor del índice NAO en el instante *T+1*, es decir, la predicción que haría nuestro modelo en el instante posterior al último valor observado en la serie. Como nuestro modelo no tiene ni tendencia ni estacionalidad, esta predicción será aplicable a cualquier instante en el futuro, ya sea en tiempo *T+1* o en tiempo *T+50*, por lo que las predicciones a futuro son deterministas.

Para representar gráficamente los valores observados y los valores predichos por la serie podemos utilizar, una vez más, la función plot().

```
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> plot(alisim1, ylab="Índice NAO", xlab="", main="")
> title("Valores observados frente a predichos")
> labs <- c("Valores observados", "Valores predichos")
> legend("bottomright", lty=c(1,1), col=c("black", "red"), legend=labs)
```



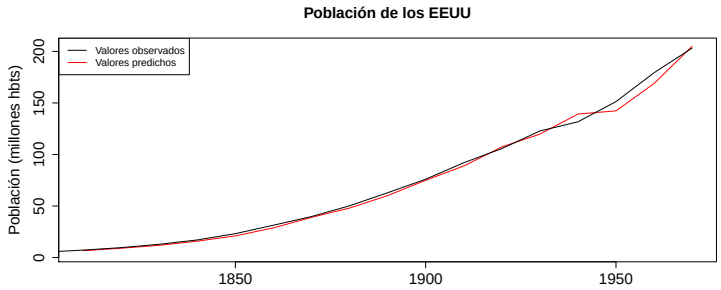
Valores observados del índice NAO invernal y predicciones utilizando un alisado exponencial simple.

Este método proporciona predicciones que explican la evolución de la serie. Aunque con errores de predicción, es más informativo que modelar utilizando para ello solamente la media de los valores observados.

3.2. Alisado de Holt

Vamos a probar ahora un modelo con tendencia para predecir los datos de crecimiento poblacional en los Estados Unidos. Estos datos se encuentran en un objeto de clase ts llamado uspop, dentro del paquete datasets. Los valores representan millones de habitantes registrados en censos realizados cada 10 años en el periodo 1790-1970.

```
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> aliholt <- HoltWinters(uspop, gamma=FALSE)
> aliholt

Holt-Winters exponential smoothing with trend and without seasonal component.

Call:
HoltWinters(x = uspop, gamma = FALSE)

Smoothing parameters:
 alpha: 1
 beta : 0.7709064
 gamma: FALSE

Coefficients:
       [,1]
a 203.20000
b  24.29044

> plot(aliholt, ylab="Población (millones hbts)", xlab="", main="")
> title("Población de los EEUU")
> labs <- c("Valores observados", "Valores predichos")
> legend("topleft", lty=c(1,1), col=c("black", "red"), legend=labs)
```

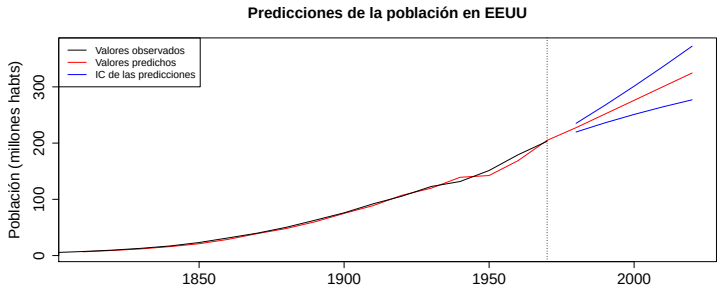


Valores observados de la población de EEUU (millones de habitantes) y predicciones utilizando un alisado de Holt para el periodo 1790-1970.

El coeficiente *alpha=1* indica que la predicción en tiempo t (en ausencia de tendencia) es función única y exclusivamente de lo observado en tiempo *t-1*. El coeficiente beta = 0.771 indica algo parecido para la pendiente, aunque en este caso, los valores anteriores a *t-1* también van a tener una ligera influencia sobre la estimación de la pendiente en tiempo t. Otra forma de verlo es que la pendiente va a ir cambiando a lo largo del tiempo y este cambio va a depender fundamentalmente de lo que se observe en los tiempos inmediatamente anteriores. El ajuste de un modelo lineal sería, por tanto, de poca ayuda en este caso. Nuestra predicción en tiempo *T+1* sería de 203.2 millones de habitantes (coeficiente a), y la tendencia para esta década vendría determinada por una pendiente de *b=24.290*.

Podríamos estar interesados en hacer predicciones más allá del rango de los valores observados. Podemos para ello utilizar la función predict(), que en realidad hace un llamamiento a la función predict.HoltWinters(). Es importante saber esto porque, mientras que para utilizar la función será suficiente con escribir predict(), para consultar la ayuda de la función tendremos que escribir ?predict.HoltWinters. La función predict() necesita como argumento principal un objeto de clase HoltWinters. Podemos especificar el número de intervalos de tiempo para el que queremos las predicciones con el argumento n.ahead. En este caso n.ahead=5 implicaría predicciones para los próximos 50 años (5 décadas) a partir de la última observación de la serie. Finalmente, podemos calcular "intervalos de predicción" con el argumento prediction.interval=TRUE. Estos intervalos se calculan utilizando la varianza de los errores de predicción calculados para toda la serie observada.

```
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> pred.aliholt <- predict(aliholt, n.ahead=5, prediction.interval=TRUE)
> plot(aliholt, pred.aliholt, ylab="Población (millones habts)", xlab="", main="")
> title("Predicciones de la población en EEUU")
> labs <- c("Valores observados", "Valores predichos", "IC de las predicciones")
> legend("topleft", lty=rep(1,3), col=c("black", "red", "blue"), legend=labs)
```



Predicciones para la población de EEUU (millones de habitantes) para las 5 próximas décadas (1970-2020) utilizando un alisado de Holt.

3.3. Alisado de Holt-Winters

En el caso de tener tendencia y estacionalidad podríamos utilizar un alisado de Holt-Winters para modelar nuestros datos. Tomemos como ejemplo una serie temporal con datos mensuales de concentraciones atmosféricas de CO_2 (en partes por millón) en los territorios noroccidentales de Canadá (cerca del Círculo Polar) para el periodo 1994-2004. Los datos están contenidos en el objeto co2 del paquete TSA[6]

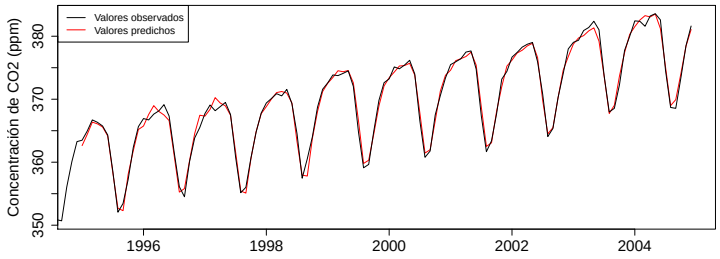
```
> data(co2, package="TSA")
> aliHW <- HoltWinters(co2)
```

[6] ¡Cuidado! Hay otros objetos con el mismo nombre (co2) en otros paquetes, por ejemplo en el paquete datasets. Para tener un control absoluto sobre el objeto que vamos a cargar utilizaremos el comando data siempre que vayamos a utilizar estos datos.

```
> aliHW

Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWinters(x = co2)

Smoothing parameters:
 alpha: 0.3610268
 beta : 0
 gamma: 0.3532495

Coefficients:
            [,1]
a   378.85957267
b     0.14625000
s1    3.87967405
s2    4.20043381
s3    4.49202098
s4    5.16172899
s5    5.39725075
s6    3.22497018
s7   -3.94239470
s8   -9.96979940
s9   -9.31512179
s10  -4.68811780
s11  -0.05222415
s12   2.53410175
```

En este caso, tenemos un efecto temporal muy marcado para la pendiente (beta = 0), lo que indicaría una pendiente constante a lo largo de toda la serie temporal. El modelo predice que en *T+1* la pendiente vale b = 0.146, aunque como beta = 0 esta sería la pendiente ajustada a lo largo de toda la serie temporal El efecto de las observaciones pasadas es intermedio para la constante (alpha = 0.361) y para la estacionalidad (gamma = 0.353). Si miramos los coeficientes s para ver el efecto de la estacionalidad en *T+1* vemos que la concentración de CO_2 predicha es máxima para el mes de mayo (s5 = 5.397) y mínima para el mes de agosto (s8 = -9.969). El gráfico resultante con las observaciones y las predicciones lo obtendríasmos con el siguiente código:

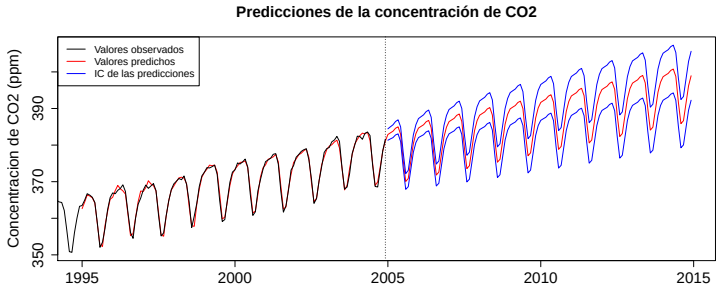
```
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> plot(aliHW, ylab="Concentración de CO2 (ppm)", xlab="", main="")
> labs <- c("Valores observados", "Valores predichos")
> legend("topleft", lty=c(1,1), col=c("black", "red"), legend=labs)
```



Valores mensuales de la concentración de CO_2 (ppm) y predicciones utilizando un alisado de Holt-Winters para el periodo 1994-2004.

Si queremos hacer predicciones a 10 años, tenemos que definir el argumento n.ahead = 120, ya que para cada año tendremos predicciones mensuales (12 meses).

```
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> pred.aliHW <- predict(aliHW, n.ahead=120, prediction.interval=TRUE)
> plot(aliHW, pred.aliHW, ylab="Concentracion de CO2 (ppm)", xlab="", main="")
> title("Predicciones de la concentración de CO2")
> labs <- c("Valores observados", "Valores predichos", "IC de las predicciones")
> legend("topleft", lty=rep(1,3), col=c("black", "red", "blue"), legend=labs)
```

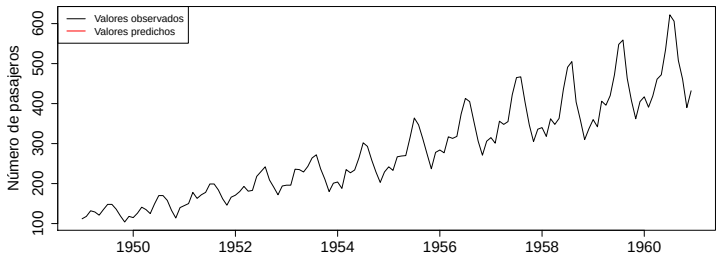


Predicciones para la concentración de CO_2 (ppm) para los 10 próximos años (2005-2015) utilizando un alisado de Holt-Winters.

Finalmente, en el caso de tener tendencia y estacionalidad, y si se observa que la amplitud del ciclo va cambiando en el tiempo, podemos utilizar un alisado de Holt-Winters con un modelo multiplicativo para la función de predicción de la componente estacional. Hasta el momento solo hemos utilizado modelos

aditivos. El modelo multiplicativo permite incorporar cambios en la amplitud del ciclo con el tiempo. El ejemplo que típicamente se utiliza para ilustrar este caso de estudio es de las líneas aéreas, en donde se observa: (1) un aumento del número de pasajeros en el tiempo (tendencia); (2) mayor número de pasajeros durante los meses de vacaciones de verano y menor durante los meses de invierno (estacionalidad); y (3) la diferencia entre el máximo y el mínimo (amplitud de ciclo) va aumentando con el tiempo.

```
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> plot(AirPassengers, ylab="Número de pasajeros", xlab="", main="")
> labs <- c("Valores observados", "Valores predichos")
> legend("topleft", lty=c(1,1), col=c("black", "red"), legend=labs)
```



Número de pasajeros en líneas aéreas internacionales para el periodo 1949-1960.

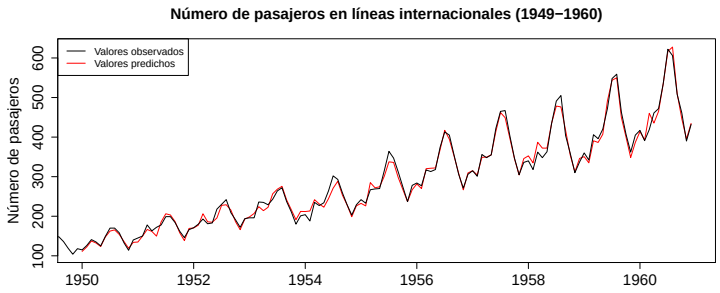
El método del alisado de Holt-Winters utilizando un modelo multiplicativo nos permite incorporar todos estos efectos en la función de predicción.

```
> aliHW2 <- HoltWinters(AirPassengers, seasonal = "mult")
> aliHW2

Holt-Winters exponential smoothing with trend and multiplicative seasonal component.

Call:
HoltWinters(x = AirPassengers, seasonal = "mult")

Smoothing parameters:
 alpha: 0.2755925
 beta : 0.03269295
 gamma: 0.8707292

Coefficients:
          [,1]
a   469.3232206
b     3.0215391
s1    0.9464611
s2    0.8829239
```

```
s3     0.9717369
s4     1.0304825
s5     1.0476884
s6     1.1805272
s7     1.3590778
s8     1.3331706
s9     1.1083381
s10    0.9868813
s11    0.8361333
s12    0.9209877
```

La representación gráfica de las predicciones del modelo se haría de la misma forma.

```
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> plot(aliHW2, ylab="Número de pasajeros", xlab="", main="")
> title(main="Número de pasajeros en líneas internacionales (1949-1960)")
> labs <- c("Valores observados", "Valores predichos")
> legend("topleft", lty=c(1,1), col=c("black", "red"), legend=labs)
```



Numero de pasajeros en líneas aéreas internacionales y predicciones utilizando un alisado de Holt-Winters con un modelo multiplicativo para la función de predicción en el periodo 1949-1960.

4. Extracción de señales

Otra forma de abordar el estudio de una serie temporal es mediante la **descomposición aditiva** de la serie. El esquema aditivo se expresaría de la siguiente forma:

$$x_t = T_t + S_t + I_t$$

$$\text{Valor observado} = \text{Tendencia} + \text{Estacionalidad} + \text{Irregular}$$

donde:

- La tendencia (T_t) representa un movimiento suave a lo largo del tiempo, que puede ser constante o variable.
- La estacionalidad (S_t) supone una oscilación dependiente de la estación.
- El componente irregular (I_t) lo componen variaciones aleatorias no explicadas por las otras componentes.

Es precisamente este componente irregular el que marca la diferencia entre los alisados exponenciales y las técnicas de descomposición de la serie temporal. Por medio de la descomposición de señales podemos modelar de forma específica la componente irregular, mientras que en los alisados se asume un carácter determinista de la serie.

Para la modelización en la descomposición aditiva se utilizan las **técnicas de extracción de señales**. Aunque existen varios métodos para llevar a cabo la extracción de señales, los pasos que se siguen de forma general son los siguientes:

1. Se extrae la tendencia y se calculan los residuos (observaciones - tendencia). Los **residuos** son la **serie sin tendencia**, que contiene la estacionalidad y el componente irregular.
2. Se estima la estacionalidad de la serie y se resta a la serie sin tendencia, se obtiene la **serie desestacionalizada**. La serie desestacionalizada no debe contener ninguna estructura aparente y debe variar en torno a un valor constante, que es el componente irregular.
3. **La predicción de la serie se realiza agregando al valor medio del componente irregular la predicción de tendencia y el componente estacional**.

4.1. Medias móviles

En R podemos utilizar la función decompose() del paquete stats para realizar la extracción de señales utilizando medias móviles. En principio, a no ser que se especifique una estructura diferente (argumento filter), R utiliza una ventana simétrica para calcular la media móvil de la pendiente en cada instante t, del siguiente modo:

$$T_t = m_t = \frac{x_{t-(k-1)/2}+\cdots+x_{t-1}+x_t+x_{t+1}+\cdots+x_{t+(k-1)/2}}{k},$$

siendo k el orden de la media móvil. Esto significa que para calcular la pendiente en tiempo t se promediarán los k valores más próximos al instante t.

Existe la opción de aplicar un filtro de medias móviles ponderado para dar mayor peso a las observaciones más próximas a t y menos peso a las observaciones más alejadas.

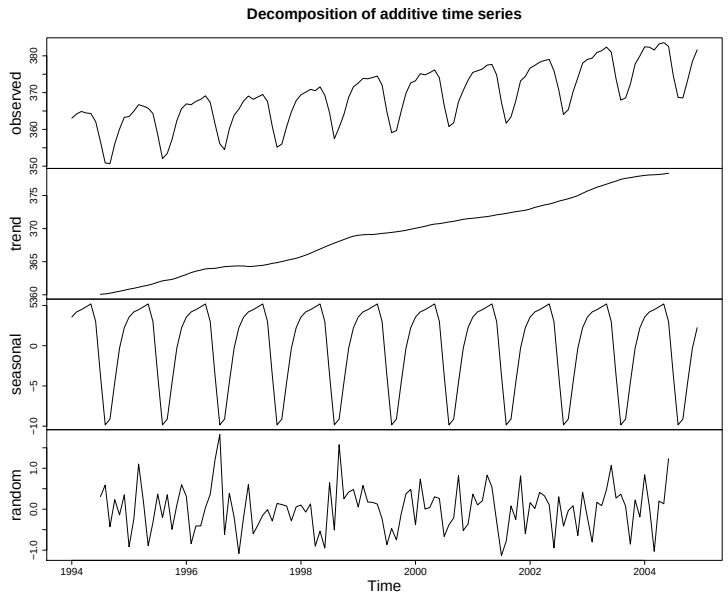
Una vez que se ha extraído la tendencia, la estacionalidad se calcula promediando las observaciones en cada estación, utilizando le metodología descrita en la sección 5.

Sigamos con el ejemplo anterior de concentraciones atmosféricas de CO_2 (en partes por millón) para el periodo 1994-2004. Vamos a realizar una descomposición aditiva sobre esta serie utilizando la función decompose(). El argumento principal que requiere esta función es un objeto de clase ts.

```
> data(co2, package="TSA")
> decomp.co2 <- decompose(co2)
> class(decomp.co2)

[1] "decomposed.ts"

> str(decomp.co2)

List of 6
 $ x       : Time-Series [1:132] from 1994 to 2005: 363 364 365 364 364 ...
 $ seasonal: Time-Series [1:132] from 1994 to 2005: 3.56 4.2 4.48 4.83 5.19 ...
 $ trend   : Time-Series [1:132] from 1994 to 2005: NA NA NA NA NA ...
 $ random  : Time-Series [1:132] from 1994 to 2005: NA NA NA NA NA ...
 $ figure  : num [1:12] 3.56 4.2 4.48 4.83 5.19 ...
 $ type    : chr "additive"
 - attr(*, "class")= chr "decomposed.ts"
```

El objeto resultante (decomp.co2) es un objeto de clase decomposed.ts que contiene los valores predichos para la estacionalidad ($seasonal), la tendencia ($trend) y la componente irregular ($random), además de los valores promedios estimados para la componente estacional ($figure). Los resultados de la descomposición pueden mostrarse gráficamente con la función plot(), que hace una llamada interna a la función plot.ts().

```
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> plot(decomp.co2)
```



Extracción de señales para la serie de concentración de CO_2(ppm) para el periodo 1994-2004 utilizando medias móviles con la función decompose().

4.2. Loess (*local polynomial regression fitting*)

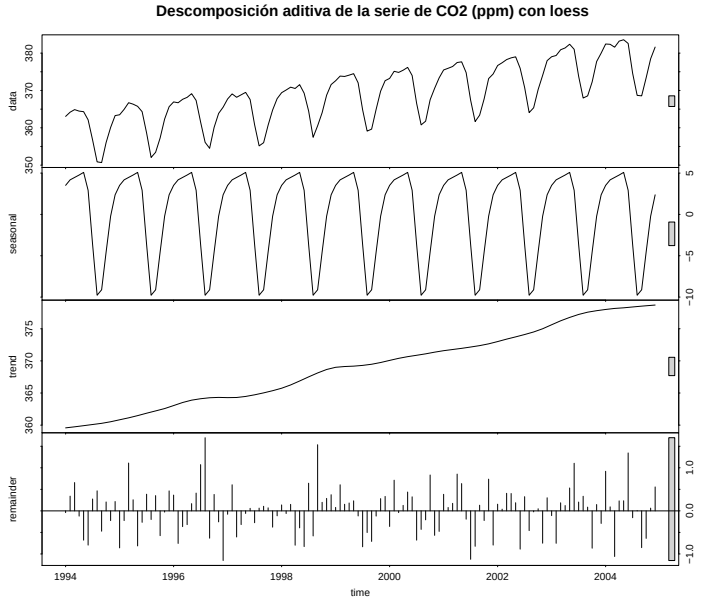
Otra forma más sofisticada de realizar la descomposición aditiva es utilizando la función stl() del paquete stats. La función stl() realiza una extracción de señales utilizando regresión polinómica local (*local polynomial regression fitting*, *loess*). La principal diferencia con medias móviles es la forma de extraer la tendencia, que es mediante regresión polinómica local. El polinomio se ajusta utilizando los mínimos cuadrados, dando más peso a los instantes cercanos al momento donde se está estimando, y menos peso a los instantes más lejanos. Muchos de los detalles de este método, tales como el grado del polinomio y el modelo de pesos, son flexibles, y se pueden modificar cambiando los argumentos establecidos por defecto. La componente estacional se extráe de la misma forma que en el caso anterior.

```
> args(stl)
```

```
function (x, s.window, s.degree = 0, t.window = NULL, t.degree = 1,
    l.window = nextodd(period), l.degree = t.degree, s.jump = ceiling(s.window/10),
    t.jump = ceiling(t.window/10), l.jump = ceiling(l.window/10),
    robust = FALSE, inner = if (robust) 1 else 2, outer = if (robust) 15 else 0,
    na.action = na.fail)
NULL
```

Es necesario especificar, al menos, dos argumentos. El primero, x, es un objeto de clase ts. El segundo, s.window, puede tomar el valor "periodic" si la serie temporal tiene una frecuencia definida (recordemos que al crear un objeto de clase ts podemos definir la frecuencia de la serie con el argumento frequency), o un número entero que indique los retardos para el cálculo de la componente estacional. Como la extracción de las componentes se basa en polinomios localmente ajustados, podemos definir el grado de los polinomios para cada una de las componentes con los argumentos s.degree, t.degree y l.degree respectivamente. Estos argumentos solo pueden tomar valores 0 o 1 y por defecto s.degree = 0, t.degree = 1 y l.degree = 1.

```
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> stl.co2 <- stl(co2, s.window="periodic")
> plot(stl.co2)
> title(main="Descomposición aditiva de la serie de CO2 (ppm) con loess", line=3)
```



Extracción de señales para la serie de concentración de CO^2 (ppm) para el periodo 1959-1997 utilizando loess (local polynomial regression fitting) con la función stl().

5. Modelos ARIMA y metodología Box-Jenkins

Existe una clase de modelos paramétricos que permite modelar series temporales estacionarias y no estacionarias: los modelos ARIMA (*Autoregressive Integrated Moving Average*). Estos modelos incluyen los modelos AR (Autoregressive), MA (Moving Averages) y ARMA (Autoregressive Moving Averages) para series estacionarias, y los modelos integrados para las series no estacionarias.

Si una serie es no estacionaria se puede transformar tomando diferencias hasta que la serie diferenciada sea estacionaria. Con esta operación se dice que la serie original es un proceso **integrado** y hablamos de modelos ARIMA. Cuando la serie también es estacional se puede utilizar el siguiente esquema multiplicativo:

$$(1-L)^d(1-L^s)^D X_t = \frac{\theta_q(B)\Theta_Q(B^S)}{\phi_p(B)\Phi_P(B^S)}a_t,$$

donde:

s es la estacionalidad.

d es el número de diferencias regulares necesarias para que la serie sea estacionaria.

D es el número de diferencias estacionales necesarias para que la serie sea estacionaria.

p y q son los ordenes de los procesos AR y MA regular, respectivamente.

P y Q son los ordenes de los procesos AR y MA estacional, respectivamente.

$\phi_p(B)$ y $\theta_q(B)$ es la estructura de los procesos AR y MA estacionarios regulares, respectivamente.

$\Phi(B^S)$ y $\Theta_Q(B^S)$ es la estructura de los procesos AR y MA estacionarios estacionales, respectivamente.

a_t es la perturbación aleatoria o innovación (ruido blanco).

Para ajustar modelos ARIMA hay que seguir una serie de pasos. Esto es lo que se conoce como metodología **Box-Jenkins**. Los pasos a seguir son:

1. Identificación del modelo ARIMA.
2. Estimación de los parámetros.
3. Validación del modelo. Si el modelo no es válido hay que volver al punto 1. Si el modelo es válido se pasa al siguiente punto.
4. Predicción de nuevos valores.

5.1. Identificación del modelo

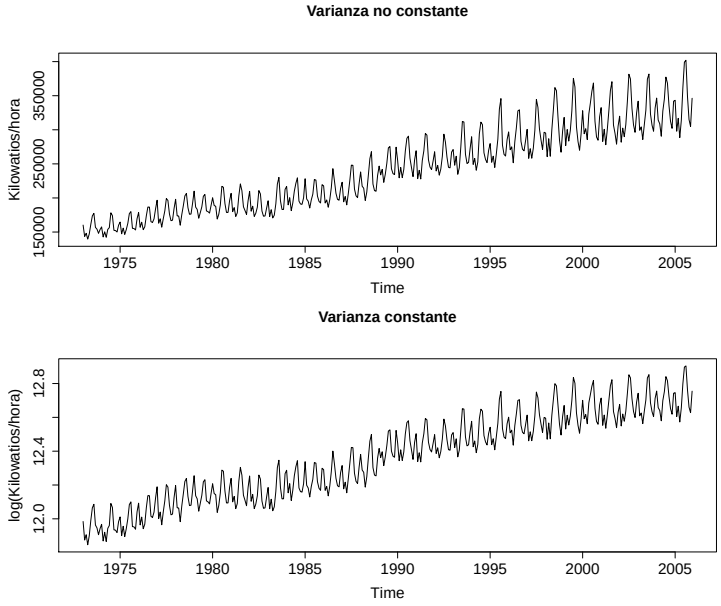
Hay que tomar tres decisiones en el siguiente orden:

- Si tomar o no logaritmos para homogeneizar la varianza.
- El número de diferencias regulares y estacionales necesarias para que la serie sea estacionaria.
- El orden de los polinomios autorregresivos y de medias móviles de la estructura regular y estacional estacionarias.

5.1.1. Transformaciones de la variable respuesta

La hipótesis de la varianza constante se exige en las condiciones de estacionariedad, pero es frecuente observar que la varianza aumenta con el nivel de la serie. En este caso la transformación con logaritmo neperiano ayuda a homogeneizar su comportamiento. Para decidir si hace falta la transformación con el *logaritmo neperiano* se utiliza el gráfico de la serie original o se agrupan los datos en bandas homogéneas, dentro de las cuales se calculan los principales estadísticos (media, varianza) que luego se representan en gráficos de caja o gráficos de rango-medias.

```
> par(mfcol=c(2,1), cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> library(TSA)
> data(electricity)
> plot(electricity, ylab="Kilowatios/hora")
> title(main="Varianza no constante", line=3)
> plot(log(electricity), ylab="log(Kilowatios/hora)")
> title(main="Varianza constante", line=3)
```



Generación de electricidad mensual (en millones de kilowatios/hora) de enero de 1973 a diciembre de 2005 en EEUU y su transformación logarítmica para homogeneizar la varianza.

Otras transformaciones de la variable respuesta también son posibles, como la familia de transformaciones Box-Cox, que incluye potencias y logaritmos.

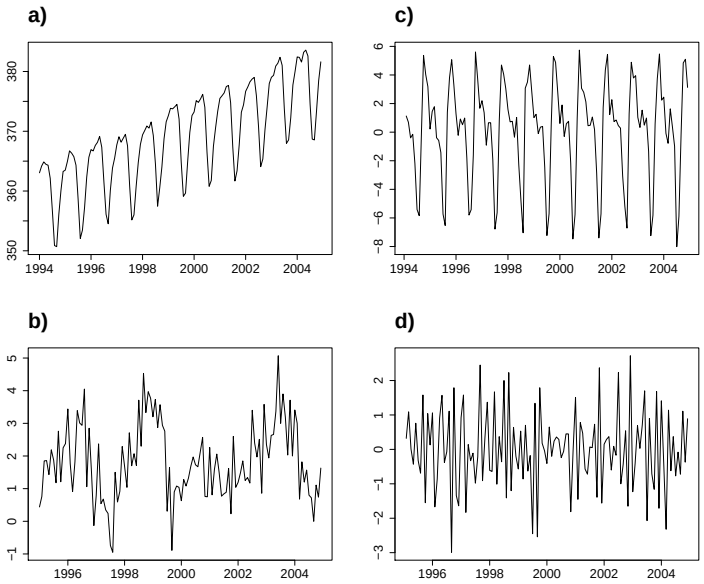
5.1.2. Diferencias regulares y estacionales que convierten la serie en estacionaria

La decisión se basa fundamentalment en la combinación de cuatro criterios:

- Gráficos de la serie.
- Desviaciones típicas de las transformaciones.
- Correlograma.
- Test de raíces unitarias, como por ejemplo el test de Dikey-Fuller aumentado.

Gráficos de la serie El gráfico de una serie estacionaria no debe presentar comportamientos tendenciales o estacionales. Si presenta alguno de estos comportamientos, podemos tomar diferencias regulares y/o estacionales para intentar convertirla en estacionaria. En R podemos utilizar la función diff() para diferenciar las series temporales. Esta función tiene un argumento lag que permite especificar los retardos. De esta manera, un lag=1 sería una diferencia para la parte regular y un lag=12 sería una diferencia para la parte estacional si tuviéramos observaciones mensuales. Si queremos aplicar una diferencia regular primero y una diferencia estacional después, utilizaremos dos veces la función diff() de forma consecutiva, como en el ejemplo que se muestra a continuación para concentraciones de CO_2.

```
> data(co2, package="TSA")
> par(mfcol=c(2,2), cex.axis=1.5, cex.lab=1.5, cex.main=2.5)
> plot(co2, ylab="", xlab="")
> title(main="a)", line=2, adj=0)
> plot(diff(co2, 12), ylab="", xlab="")
> title(main="b)", line=2, adj=0)
> plot(diff(co2, 1), ylab="", xlab="")
> title(main="c)", line=2, adj=0)
> plot(diff(diff(co2, 1), 12), ylab="", xlab="")
> title(main="d)", line=2, adj=0)
```



a) Serie no estacionaria (d=0; D=0); b) Serie no estacionaria con 1 diferencia regular (d=1; D=0); c) Serie no estacionaria con 1 diferencia estacional (d=0; D=1); d) Serie no estacionaria con una diferencia regular y una diferencia estacional (d=1; D=1).

Desviaciones típicas de las transformaciones Se seleccionará aquella diferenciación que minimice la varianza, ya que en una serie sobrediferenciada la varianza aumenta. Siguiendo con el ejemplo anterior, podemos tomar varias diferencias regulares y varias diferencias estacionales y calcular su varianza.

```
> var(diff(co2, 1))

[1] 12.03632

> var(diff(co2, 2))

[1] 38.83039
```

```
> var(diff(co2, 3))

[1] 65.73068

> var(diff(co2, 4))

[1] 83.26337

> var(diff(diff(co2, 1), 12))

[1] 1.338993

> var(diff(diff(co2, 2), 12))

[1] 1.245044

> var(diff(diff(co2, 3), 12))

[1] 1.457303

> var(diff(diff(co2, 4), 12))

[1] 1.808425
```

En este caso, la serie con dos diferencias regulares y una diferencia estacional es la que minimiza la varianza. No obstante, el gráfico de la serie diferenciada con 1 diferencia regular y 1 estacional mostraba buenas propiedades y la varianza es casi idéntica a la de la serie con dos diferencias regulares, por lo que elegiremos la primera en lugar de la segunda.

Correlograma Una herramienta muy importante de diagnóstico para examinar la dependencia es el correlograma. El correlograma es una sucesión de valores que miden el grado de dependencia lineal entre observaciones próximas. Habitualmente se representan con barras.

El **correlograma** se construye con los coeficientes de correlación calculados con todos los datos que distan entre sí uno, dos, tres, cuatro, y hasta n periodos. El correlograma es una aproximación muestral de la función de autocorrelación (ACF).

En el correlograma de una serie estacionaria, la dependencia temporal debe acercarse a cero rápidamente. Sin embargo, un decrecimiento lento de las autocorrelaciones indica que la serie **no es estacionaria**. Y un decrecimiento lento de la correlación de los retardos estacionales indica que la serie **tiene un componente estacional**. Muchas veces la no estacionariedad se observa fácilmente en el gráfico de la serie, otras

tendremos que recurrir al correlograma para ayudarnos a decidir si la serie es o no estacionaria.

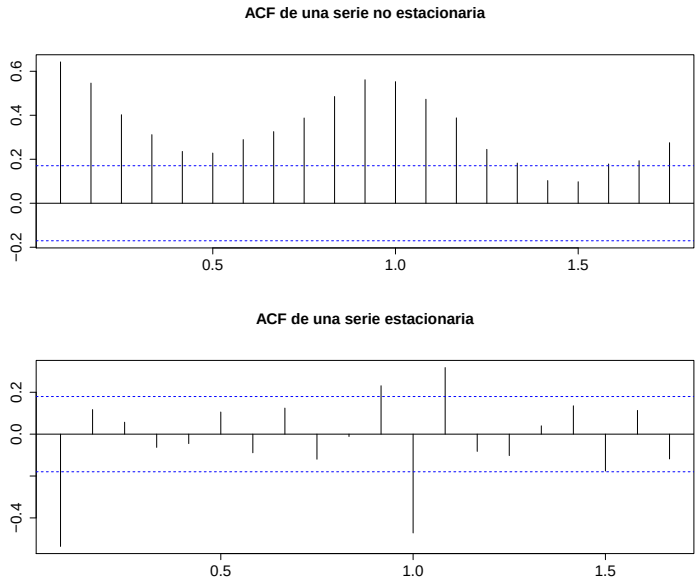


Gráfico de autocorrelación de una serie estacionaria y una serie no estacionaria.

Para calcular la autocorrelación en R podemos utilizar la función acf(), del paquete stats, cuyo principal argumento es un objeto de clase ts. Hay un argumento en esta función, plot, que especifica si queremos obtener los resultados gráficamente o no. Por defecto plot=TRUE. Hay otra función acf() dentro del paquete TSA, que en realidad es la misma función acf() del paquete stats modificada de tal forma que, por defecto, no se calcula la autocorrelación para retardos de cero (que implicarían una correlación de 1). Da exactamente lo mismo utilizar una u otra. Si tenemos los dos paquetes cargados no tendremos control sobre qué función estamos usando (por defecto se usará la función acf(), del paquete TSA). Para especificar el paquete del cual queremos extraer la función deberemos escribir el nombre del paquete::nombre de la función. Por ejemplo:

```
> TSA::acf()
> stats::acf()
```

Tomemos como ejemplo los datos de temperatura media mensual contenidos en el arreglo de datos tempdub del paquete TSA.

```
> library(TSA)
> data(tempdub)
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> acf(tempdub, ylab="", xlab="", main="")
```

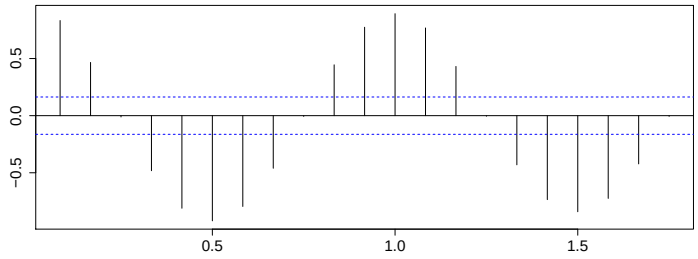


Gráfico de autocorrelación de temperaturas medias mensuales.

El gráfico de autocorrelación muestra los valores de correlación entre las observaciones en tiempo *t* y *t-1*, *t* y *t-2* y así sucesivamente. Cada barra representa un retardo. En este caso, los valores en el eje *x* representan los años y cada barra representa un mes. Hay 12 barras en cada año representando cada barra la correlación entre un mes y el mes anterior, un mes y dos meses anteriores y así sucesivamente. Vemos que hay correlaciones positivas significativas entre un mes y los meses más próximos (1 y 2 retardos) y un mes y el mismo mes al siguiente año (12 retardos y valores próximos). También hay correlaciones negativas significativas con 4, 5, 6, 7 y 8 retardos. Podemos representar más o menos retardos modificando el argumento lag.max. Si aumentamos el número de retardos a 40 o 50 veremos que se produce un decrecimiento lento de los retardos estacionales. Esto quiere decir que en *t=12* la correlación es alta, en *t=24* algo menor, en *t=36* algo menor y así sucesivamente. Las líneas discontínuas representan el valor por encima del cual la correlación es significativamente distinta de cero. Esto apunta a un proceso no estacionaria estacional, por lo que sería necesario diferenciar la serie para intentar convertirla en estacionaria.

Tomemos ahora el ejemplo de concentraciones de CO_2.

```
> par(cex.axis=1.5, cex.lab=1.5, cex.main=1.5)
> acf(co2, lag.max=48, ylab="", xlab="", main="")
```

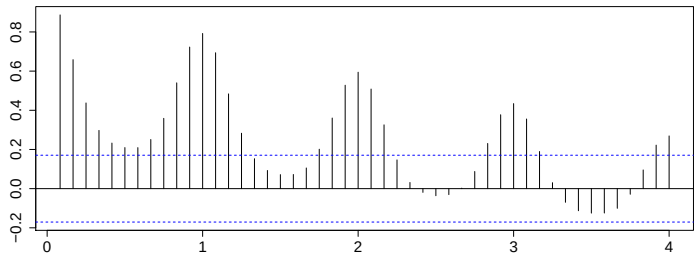


Gráfico de autocorrelación de las concentraciones mensuales de CO_2 (ppm).

En este caso, observamos que hay no solo un decrecimiento lento de la correlación de los retardos estacionales (en este caso siempre positivos), sino también un decrecimiento lento de las correlaciones alrededor de los retardos estacionales, lo que apunta a un proceso no estacionario estacional y con tendencia, algo que ya podía verse fácilmente en la gráfica de la serie temporal.

Test de raíces unitarias El test de raíces unitarias es un test para comprobar la estacionariedad de una serie temporal, y se utiliza con series previamente desestacionalizadas. Uno de estos test es el test de **Dickey-Fuller aumentado**. La hipótesis nula en este test es que la serie no es estacionaria, y por tanto habría que diferenciar la serie para que fuese estacionaria, y la hipótesis alternativa que no hay que diferenciar (aunque esto puede cambiarse con el argumento alternative). Por tanto, si el *p-valor* que obtenemos en este test supera el nivel de significación (p. ej. $\alpha = 0{,}05$), entonces tendremos que diferenciar la componente regular de la serie, es decir, asumimos que es no estacionaria. Si por el contrario, rechazamos la hipótesis nula, entonces, no habrá que diferenciar.

Sigamos con el ejemplo anterior.

```
> library(tseries)
> adf.test(diff(co2,12))

        Augmented Dickey-Fuller Test

data:  diff(co2, 12)
Dickey-Fuller = -3.0091, Lag order = 4, p-value = 0.1575
alternative hypothesis: stationary
```

En este caso, una vez que hemos desestacionalizado la serie, no rechazamos la hipótesis nula, lo que evidencia que es necesario una diferenciación en la parte regular.

5.1.3. Orden de los polinomios autorregresivos y de medias móviles de la estructura regular y estacional estacionarias

Si tenemos una serie estacionaria o una vez que hemos convertido a estacionaria una serie no estacionaria, lo siguiente es definir el orden de los polinomios autorregresivos y de medias móviles de la estructura regular y estacional de la serie. Para entender esto es necesario primero dar algunas nociones de lo que son los modelos autorregresivos (AR) y los modelos de medias móviles (MA). Para predecir el valor de la variable t en una serie estacionaria podemos utilizar la siguiente información:

- Los valores pasados de la serie: X_1, X_2,...,X_{t-1}.
- Las innovaciones pasadas: a_1, a_2,...,a_{t-1}.

Dependiendo de que para predecir X_t se utilicen los valores pasados de la serie o las innovaciones pasadas, tendremos los modelos AR o los modelos MA, respectivamente.

En los **modelos autorregresivos AR(p)** cada observación X_t depende linealmente de las anteriores, y **el número de observaciones de las que depende es el orden p**.

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + a_t,$$

dóndeϕ_j es el coeficiente que relaciona la observación en tiempo t con la observación X_{t-j}, y a_t es el ruido blanco en tiempo t.

En los **modelos de medias móviles MA(q)** cada observación X_t depende linealmente de las perturbaciones anteriores, y **el número de perturbaciones de las que depende es el orden q**.

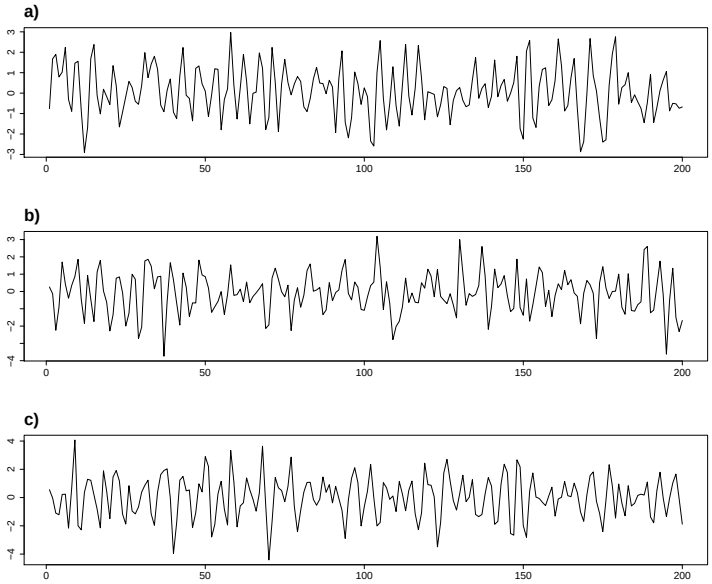
$$X_t = a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_q a_{t-q},$$

dónde a_t es el ruido blanco en tiempo t, yθ_j es el coeficiente que relaciona la observación en tiempo t con la perturbación a_{t-j}.

Finalmente, en los **modelos autorregresivos y de medias móviles ARMA(p,q)** cada observación depende linealmente de las p anteriores y de las q últimas innovaciones.

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_q a_{t-q} + a_t$$

En la práctica es muy difícil distinguir de cuántas observaciones (p) e innovaciones (q) pasadas depende la observación t. También es muy difícil distinguir si la relación de X_t con el pasado sigue un modelo AR, un modelo MA o un modelo ARMA.



Simulación de series temporales con (a) una estructura autorregresiva (AR); (b) de medias móviles (MA); y (c) autorregresiva de medias móviles (ARMA). No se pueden distinguir estos procesos a simple vista y mucho menos el orden p y q de influencia de las observaciones e innovaciones pasadas, respectivamente.

¿Qué se puede hacer entonces? Es posible identificar el modelo a partir de la estructura de autocorrelación de los datos. Para ello utilizaremos la función de correlación (ACF). Sin embargo, aunque la ACF nos permite distinguir a grandes rasgos un proceso AR de un proceso MA, no siempre es tan fácil identificar los órdenes p y q. Por este motivo también se utiliza la función de autocorrelación parcial (PACF).

La **autocorrelación parcial** de orden k es una medida de la relación lineal entre las observaciones separadas por k periodos, independientemente de los valores intermedios. Toma valores entre -1 y 1. El cero indica que no hay relación. En R la **autocorrelación parcial** la podemos obtener con la misma función `acf()` especificando que el argumento `type="partial"` o directamente utilizando la función `pacf()`.

Veamos ahora cómo se comportan la ACF y la PACF de distintos procesos autorregresivos y de medias móviles de distinto orden.

AR(1): En un proceso autorregresivo de orden 1, cada observación se puede predecir con el valor de la anterior salvo por una perturbación aleatoria.

$$X_t = \phi_1 X_{t-1} + a_t$$

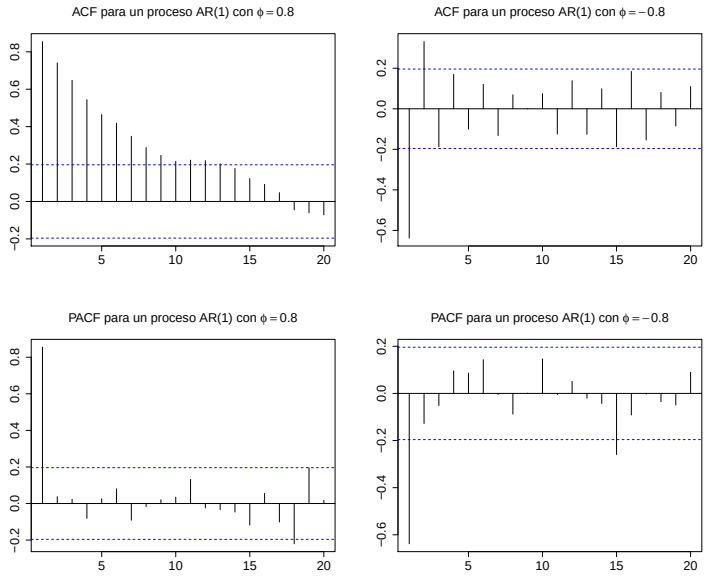
Se asume $a_t \approx N(0, \sigma_a)$ independiente (ruido blanco gaussiano).

Un AR(1) es estacionario si $-1 < \phi_1 < 1$.

En un AR(1) la ACF decrece exponencialmente. Si el coeficiente $\phi_1 > 0$ la ACF decrece monótonamente. Si por el contrario $\phi_1 < 0$, la ACF decrece alternando valores positivos y negativos.

En cuanto a la PACF solo tiene un valor no nulo. Si el coeficiente del AR es positivo, el primer valor de la PACF es positivo y el resto cero. Si el coeficiente del AR es negativo, el primer valor de la PACF es negativo y el resto cero.

```
> par(mfcol=c(2,2), cex.axis=1.5, cex.main=1.5)
> ar1.pos <- arima.sim(n= 100, model=list(ar=c(0.8)))
> ar1.neg <- arima.sim(n= 100, model=list(ar=c(-0.8)))
> tit <- expression(paste("ACF para un proceso AR(1) con ", phi == 0.8))
> acf(ar1.pos, xlab="", ylab="", main=tit)
> tit <- expression(paste("PACF para un proceso AR(1) con ", phi == 0.8))
> pacf(ar1.pos, xlab="", ylab="", main=tit)
> tit <- expression(paste("ACF para un proceso AR(1) con ", phi == -0.8))
> acf(ar1.neg, xlab="", ylab="", main=tit)
> tit <- expression(paste("PACF para un proceso AR(1) con ", phi == -0.8))
> pacf(ar1.neg, xlab="", ylab="", main=tit)
```



Ejemplo de funciones de autocorrelación (ACF) y autocorrelación parcial (PACF) para un proceso AR(1) con coeficiente ϕ positivo y negativo, respectivamente.

AR(2): El modelo incorpora las dos últimas observaciones.

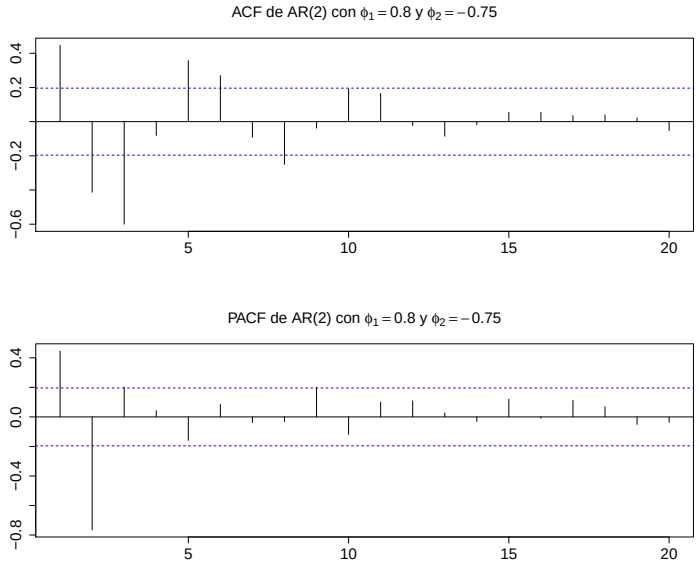
$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + a_t$$

Se asume $a_t \approx N(0, \sigma_a)$ independiente (ruido blanco gaussiano).

ϕ_1 y ϕ_2 tienen que cumplir ciertas condiciones para que AR(2) sea estacionario.

La ACF decrece rápidamente pero puede hacerlo de formas muy distintas según los valores de los parámetros. Es muy difícil distinguir entre un AR(1) y un AR(2) con la ACF. Se usa la PACF para hacer esta distinción, ya que solo tiene los dos primeros valores no nulos.

```
> par(mfcol=c(2,1), cex.axis=1.5, cex.main=1.5)
> ar2 <- arima.sim(n= 100, model=list(ar=c(0.8, -0.75)))
> tit <- expression(paste("ACF de AR(2) con ", phi[1] == 0.8, " y ", phi[2] == -0.75))
> acf(ar2, xlab="", ylab="", main=tit)
> tit <- expression(paste("PACF de AR(2) con ", phi[1] == 0.8, " y ", phi[2] == -0.75))
> pacf(ar2, xlab="", ylab="", main=tit)
```



Ejemplo de funciones de autocorrelación (ACF) y autocorrelación parcial (PACF) para un proceso AR(2) con coeficientes $\phi_1 = 0{,}8$ y $\phi_2 = -0{,}75$.

MA(1): En un proceso de medias móviles cada observación depende de las innovaciones más recientes, la actual a_t y la anterior a_{t-1}.

$$X_t = a_t - \theta_1 a_{t-1}$$

Se asume $a_t \approx N(0, \sigma_a)$ independiente (ruido blanco gaussiano).

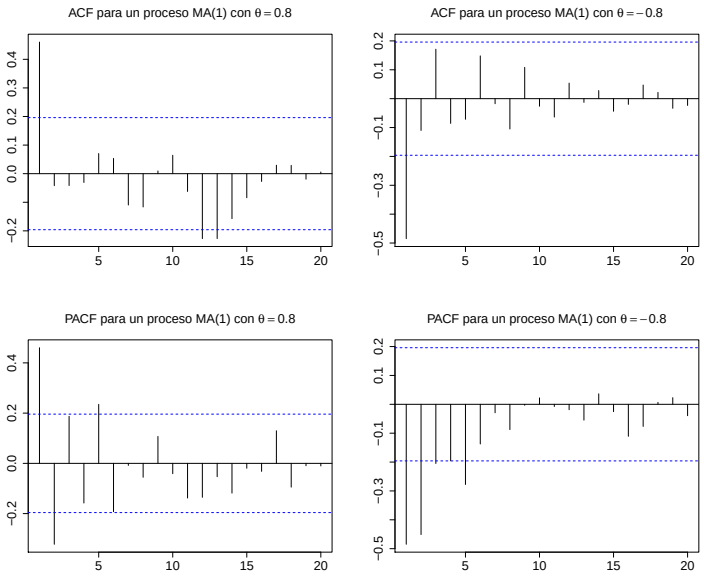
Un MA(1) es siempre estacionario, y es invertible[7] si $-1 < \theta_1 < 1$.

La ACF de un MA(1) solo tiene un valor no nulo. Si el coeficiente del MA es positivo, el primer valor de la ACF es negativo y el resto cero. Si el coeficiente del MA es negativo, el primer valor de la ACF es positivo y el resto cero.

[7]La invertibilidad de un proceso es la condición que permite estimar las innovaciones. Un proceso es invertible si se puede escribir como una combinación lineal infinita de las observaciones pasadas. Los AR son siempre invertibles.

En cuanto a la PACF, decrece exponencialmente. Si el coeficiente del MA es positivo, la PACF decrece alternando valores positivos y negativos. Si el coeficiente del MA es negativo, la PACF decrece monótonamente.

```
> par(mfcol=c(2,2), cex.axis=1.5, cex.main=1.5)
> ma1.pos <- arima.sim(n= 100, model=list(ma=c(0.8)))
> ma1.neg <- arima.sim(n= 100, model=list(ma=c(-0.8)))
> tit <- expression(paste("ACF para un proceso MA(1) con ", theta == 0.8))
> acf(ma1.pos, xlab="", ylab="", main=tit)
> tit <- expression(paste("PACF para un proceso MA(1) con ", theta == 0.8))
> pacf(ma1.pos, xlab="", ylab="", main=tit)
> tit <- expression(paste("ACF para un proceso MA(1) con ", theta == -0.8))
> acf(ma1.neg, xlab="", ylab="", main=tit)
> tit <- expression(paste("PACF para un proceso MA(1) con ", theta == -0.8))
> pacf(ma1.neg, xlab="", ylab="", main=tit)
```



Ejemplo de funciones de autocorrelación (ACF) y autocorrelación parcial (PACF) para un proceso MA(1) con coeficiente θ positivo y negativo, respectivamente.

MA(2): Cada observación depende de las innovaciones externas más recientes, la actual a_t y las dos anteriores.

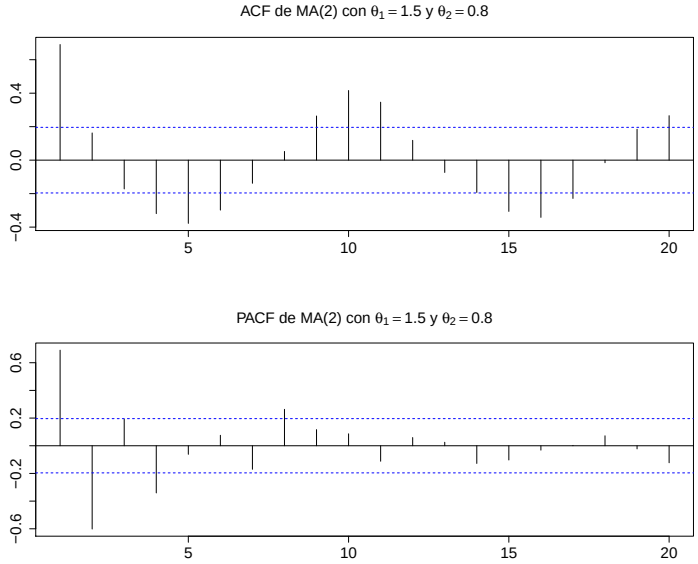
$$X_t = a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2}$$

Se asume $a_t \approx N(0, \sigma_a)$ independiente (ruido blanco gaussiano).

Un MA(2) es siempre estacionario, y es invertible si θ_1 y θ_2 cumplen ciertas condiciones.

La ACF de un proceso MA(2) solo tiene los dos primeros valores no nulos. La PACF puede presentar aspectos muy distintos.

```
> par(mfcol=c(2,1), cex.axis=1.5, cex.main=1.5)
> ma2 <- arima.sim(n= 100, model=list(ma=c(1.5,0.8)))
> tit <- expression(paste("ACF de MA(2) con ", theta[1] == 1.5, " y ", theta[2] == 0.8))
> acf(ma2, xlab="", ylab="", main=tit)
> tit <- expression(paste("PACF de MA(2) con ", theta[1] == 1.5, " y ", theta[2] == 0.8))
> pacf(ma2, xlab="", ylab="", main=tit)
```



Ejemplo de funciones de autocorrelación (ACF) y autocorrelación parcial (PACF) para un proceso MA(2) con coeficiente $\theta_1 = 1{,}5$ *y* $\theta_2 = 0{,}8$.

ARMA(p,q): Un ARMA(p,q) es una generalización de los procesos AR y MA de orden p y q, respectivamente, en donde cada observación depende linealmente de las p anteriores y de las q últimas innovaciones.

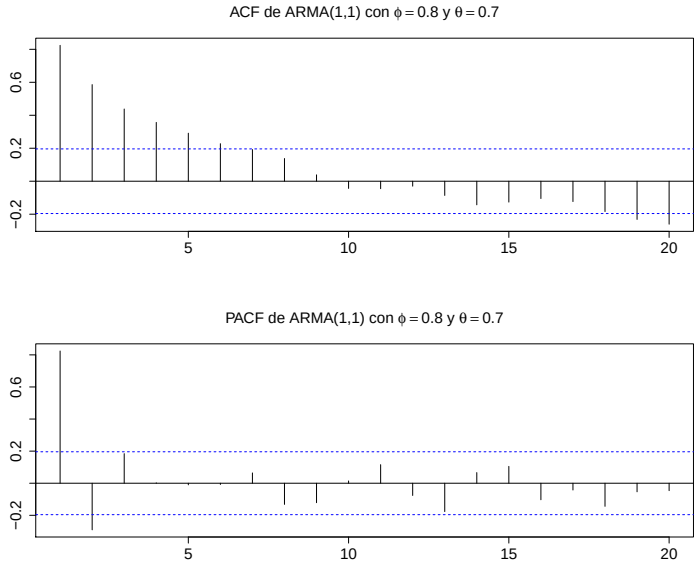
$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_q a_{t-q} + a_t$$

Se asume $a_t \approx N(0, \sigma_a)$ independiente (ruido blanco gaussiano). Aunque en la práctica esto solo ocurre si se utilizan modelos de órdenes bajos, como ARMA(1,1), ARMA(2,1), etc.

En los ARMA(p,q) la ACF tiene los q primeros valores según la estructura del MA, a partir de ahí decrece según la estructura del AR.

La PACF tiene los p primeros valores según la estructura del AR, a partir de ahí decrece según la estructura del MA.

```
> par(mfcol=c(2,1), cex.axis=1.5, cex.main=1.5)
> arma1.1 <- arima.sim(n= 100, model=list(ar=0.8, ma=0.7))
> tit <- expression(paste("ACF de ARMA(1,1) con ", phi == 0.8, " y ", theta == 0.7))
> acf(arma1.1, xlab="", ylab="", main=tit)
> tit <- expression(paste("PACF de ARMA(1,1) con ", phi == 0.8, " y ", theta == 0.7))
> pacf(arma1.1, xlab="", ylab="", main=tit)
```

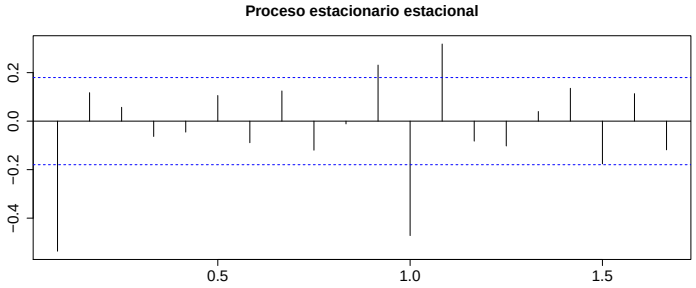


Ejemplo de funciones de autocorrelación (ACF) y autocorrelación parcial (PACF) para un proceso ARMA(1,1) con coeficiente $\phi = 0{,}8$ *y* $\theta = 0{,}7$.

Procesos estacionarios estacionales: En muchas series horarias, diarias y mensuales, las observaciones presentan dependencia con las observaciones previas, pero también con las que ocurrieron hace un día, una semana, un mes, o un año. En estas situaciones la ACF del modelo debe reflejar dependencia temporal larga, lo que nos lleva a modelos con p y q elevados (muchos parámetros).

De esta forma, se puede especificar un modelo ARMA con información de la variable y las innovaciones en los retardos estacionales (s, 2s, 3s,...), de manera equivalente a los ARMA regulares.

La estructura ARMA estacional, $ARMA(P,Q)_s$, aparece en los retardos estacionales de la ACF y la PACF, con las mismas características que en el estudio que se ha hecho de la parte regular. Si, por ejemplo, la estacionalidad es anual, debemos fijarnos en los retardos 12, 24, 36, etc.



Ejemplo de proceso estacionario estacional.

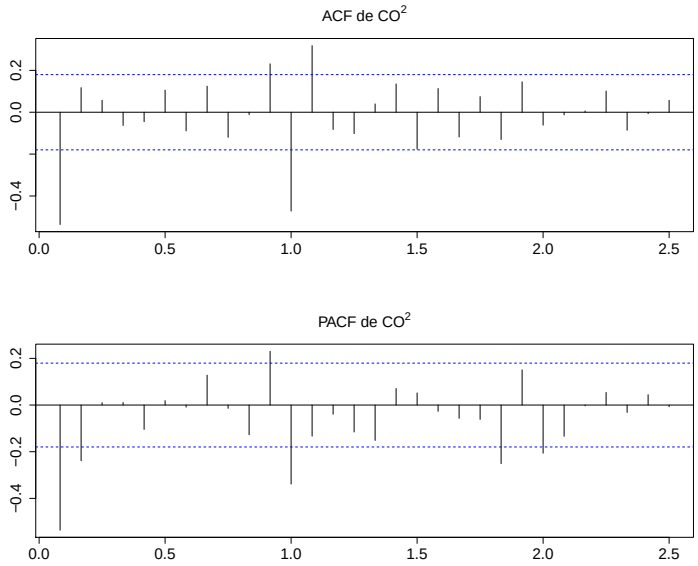
Cuando la serie presente una estructura de dependencia con el pasado reciente y también una estructura de dependencia entre retardos estacionales, los modelos ARMA y ARMA estacional multiplicativos permiten modelizar el comportamiento de la serie con el menor número de parámetros posibles (principio de parsimonia).

Siguiendo con el ejemplo de concentraciones de CO_2, vamos a ver, una vez que se ha convertido a estacionaria la serie, cual sería la estructura óptima del modelo ARMA interpretando la ACF y la PACF. Vimos en la sección 5.1.2 que necesitábamos 1 diferencia regular y 1 diferencia estacional para convertir la serie en estacionaria.

```
> co2.stat <- diff(diff(co2, 1), 12)
```

Ahora que tenemos la serie estacionaria, vamos a explorar la función de autocorrelación y de autocorrelación parcial para ver si tenemos alguna idea de cual pudiera ser la estructura óptima del modelo ARMA.

```
> par(mfcol=c(2,1), cex.axis=1.5, cex.main=1.5)
> acf(co2.stat, xlab="", ylab="", main=expression(paste("ACF de ", CO^2)), lag.max=30)
> pacf(co2.stat, xlab="", ylab="", main=expression(paste("PACF de ", CO^2)), lag.max=30)
```



Funciones de autocorrelación (ACF) y autocorrelación parcial (PACF) para la serie estacionarizada de concentraciones de CO_2.

La parte regular del correlograma muestra un primer valor significativo y negativo, que indica una estructura de tipo MA(1). También observamos un decrecimiento exponencial en la PACF, lo que confirma que se trata de un MA(1). Existe también una componente estacional. Si se tratase de una estructura autorregresiva (AR), tendríamos un decaimiento exponencial en la componente estacional (12 meses, 24 meses,...). Sin embargo, sólo observamos una correlación significativa en *t=12* en la ACF, por lo que posiblemente se trate también de un MA(1) en la parte estacional estacionaria.

Resumen de modelos para series estacionarias: Conviene tener en cuenta que:

- Los modelos AR, MA y ARMA son capaces de describir el comportamiento de los procesos estacionarios con una cantidad pequeña de parámetros.
- En los AR, la ACF y las predicciones decrecen a cero como mezcla de exponenciales y sinusoidales.
- En los MA, la ACF y las predicciones se prolongan q periodos sin estructura.
- Los modelos ARMA combinan las propiedades de los procesos AR y MA.

- La función de autocorrelación parcial sirve para determinar el orden de los procesos AR.

En la siguiente tabla se resume la dualidad entre los procesos de tipo autorregresivo y los de medias móviles:

	Estacionariedad	**Invertibilidad**	**ACF**	**PACF**	**Predicciones**
MA(q)	Siempre	Hay que comprobarlo	Se corta tras q retardos	Estructura exponencial o sinusoidal	q predicciones distintas de cero sin estructura
AR(p)	Hay que comprobarlo	Siempre	Estructura exponencial o sinusoidal	Se corta tras p retardos	Infinitas predicciones distintas de cero pero que tienden a cero exponencial o sinusoidalmente

En general, una serie temporal puede mostrar:

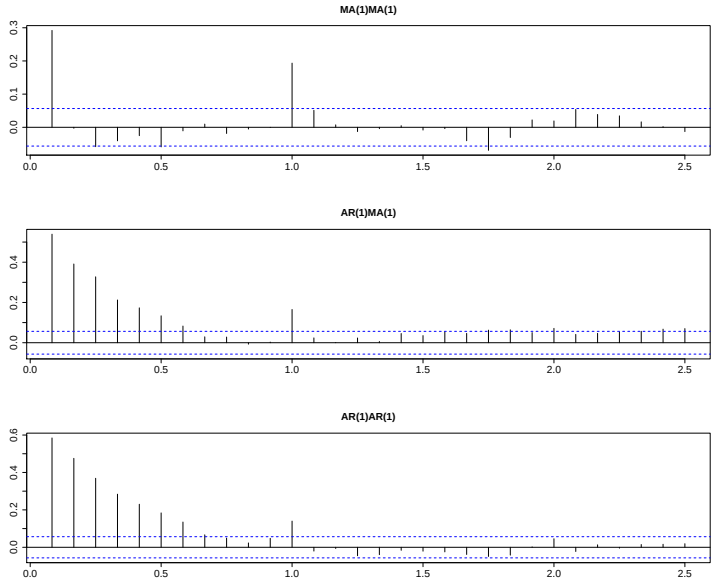
- Dependencia regular, que se representa con un modelo ARMA regular.
- Dependencia estacional, que se representa con un modelo ARMA estacional.

El modelo ARMA multiplicativo incorpora ambas dependencias con menos parámetros. Se multiplican las estructuras regular y estacional. La ACF de un modelo ARMA con dependencia regular y estacional presente las siguientes características:

- En los primeros retardos estará la ACF correspondiente a la parte regular del modelo.
- En los retardos estacionales estará la ACF correspondiente a la parte estacional del modelo.
- A la derecha e izquierda de los retardos estacionales aparecerá la estructura regular (con el mismo signo o contrario).

Veamos algunos ejemplos:

```
> par(mfcol=c(3,1), cex.axis=1.5, cex.main=1.5)
> library(CombMSC)
> ma1ma1 <- sarima.Sim(n= 100, model=list(ma=0.8), period=12, seasonal=list(ma=0.5))
> tit <- "MA(1)MA(1)"
> acf(ma1ma1, xlab="", ylab="", main=tit)
> ar1ma1 <- sarima.Sim(n= 100, model=list(ar=0.8), period=12, seasonal=list(ma=0.5))
> tit <- "AR(1)MA(1)"
> acf(ar1ma1, xlab="", ylab="", main=tit)
> ar1ar1 <- sarima.Sim(n= 100, model=list(ar=0.8), period=12, seasonal=list(ar=0.5))
> tit <- "AR(1)AR(1)"
> acf(ar1ar1, xlab="", ylab="", main=tit)
```



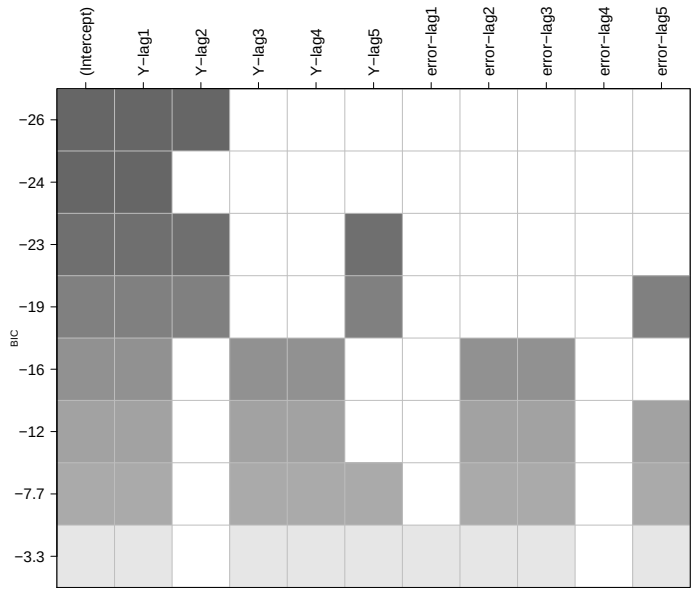
Ejemplos de correlogramas de distintos ARIMA estacionales.

Selección automática del modelo ARMA Podemos utilizar también la función armasubsets() del paquete TSA para realizar una selección automática del modelo ARMA, una vez que la serie se ha convertido a estacionaria y siempre que no tenga estacionalidad estacionaria. Esta función se aproxima a un modelo ARMA por medio de modelos de regresión donde las covariables son los retardos de la serie temporal y los retardos de las innovaciones (obtenidos a partir del ajuste de un modelo AR). Los argumentos de esta función se muestran a continuación:

```
> args(armasubsets)

function (y, nar, nma, y.name = "Y", ar.method = "ols", ...)
NULL
```

El argumento principal, y, es un objeto de clase ts. El argumento nar especifica el máximo orden del proceso autorregresivo (AR), y el argumento nma especifica el máximo orden del proceso de medias móviles (MA).

```
> library(TSA)
> par(cex.axis=1.5, cex.main=1.5)
> bestco2 <- armasubsets(co2.stat, nar=5, nma=5)
> plot(bestco2)
```



Aplicación de Criterios de Información Bayesiana (BIC) para la selección de distintos modelos ARMA para la serie estacionarizada de concentraciones de CO_2.

La Y que aparece en la parte superior de la gráfica hace referencia a la parte AR regular del modelo y el error hace referencia a la parte MA regular del modelo. El número hace referencia al orden del modelo. Los mejores modelos aparecen en la parte superior de la tabla (menor BIC, que es el más negativo). La fila superior nos indica que el submodelo del modelo ARMA(5,5) que tiene el menor BIC contiene 1 y 2 retardos de la serie temporal (y ninguno de los errores), es decir, correspondería a un AR(2).

5.2. Estimación de los parámetros

La estimación es compleja (sobre todo cuando hay estructura MA) y requiere el uso de algoritmos de búsqueda de soluciones óptimas. R nos va a dar una tabla de estimación de los coeficientes, generalmente muy sencilla de interpretar. En R podemos realizar la estimación de los parámetros con la función arima() del paquete stats. Los paquetes TSA y forecast también tienen sus propias funciones para realizar la estimación de los parámetros, arima() y

Arima(), respectivamente, si bien estas funciones hacen llamadas internas a la función arima() del paquete stats.

Para estimar los parámetros del modelo ARIMA para la serie temporal de concentración de CO_2 tendríamos que especificar la parte regular del modelo con el argumento order. Este argumento se especifica como una lista de tres componentes, p, d y q, que hacen referencia al orden del modelo AR (p), al grado de diferenciación (d), y al orden del modelo MA (q). En nuestro caso, sería p=0, d=1 y q=1. La parte estacional del modelo se especifica con el argumento seasonal, que tiene también tres componentes, P, D y Q, equivalentes a la parte regular del modelo.

```
> arima.co2 <- arima(co2, order=c(0,1,1), seasonal=list(order=c(0,1,1)))
> arima.co2

Call:
arima(x = co2, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1)))

Coefficients:
          ma1     sma1
      -0.5792  -0.8206
s.e.   0.0791   0.1137

sigma^2 estimated as 0.5446:  log likelihood = -139.54,  aic = 283.08
```

En principio ya hemos seleccionado la mejor estructura AR y MA para el modelo, por lo que no sería necesario calcular la significación de los coeficientes del modelo. Sin embargo podríamos llegar a varios mejores modelos utilizando la ACF y la PACF o el BIC/AIC. En este caso podríamos calcular la significación de los coeficientes dividiendo los coeficients por sus errores estándar para calcular el estadístico z y su p-valor asociado.

```
> (1-pnorm(abs(arima.co2$coef)/sqrt(diag(arima.co2$var.coef))))*2

         ma1         sma1
2.398082e-13 5.380141e-13
```

Si no estamos muy seguros de si hemos definido bien el orden de los polinomios autorregresivos y de medias móviles de la estructura regular y estacional estacionarias, podemos utilizar la función auto.arima() del paquete forecast. Esta función compara con criterios de información (por defecto el Criterio de Información de Akaike corregido, AICc) todos los posibles modelos con un máximo de órden p y q para la estructura regular y P y Q para la estructura estacionaria definidos por el usuario. Los argumentos max.p, max.q, max.P y max.Q hacen referencia al máximo orden para la parte regular del modelo AR y MA y para la parte estacionaria del modelo AR y MA, respectivamente. El argumento d hace referencia a las diferencias regulares que hay que tomar para estacionarizar la serie. El argumento D hace referencia a las diferencias estacionales que hay que tomar para estacionarizar la serie. Por defecto la

función realiza un procedimiento por pasos para seleccionar el mejor modelo. Esto agiliza bastante el tiempo de cálculo, aunque se puede cambiar con el argumento stepwise=FALSE para que calcule todos los posibles modelos.

```
> best.arima<-auto.arima(co2, d=1, D=1, max.p=5, max.q=5, max.P=2, max.Q=2)
> best.arima

Series: co2
ARIMA(0,1,1)(0,1,1)[12]

Coefficients:
          ma1     sma1
      -0.5792  -0.8206
s.e.   0.0791   0.1137

sigma^2 estimated as 0.5446:  log likelihood=-139.54
AIC=285.08   AICc=285.29   BIC=293.41
```

Como vemos, el resultado en este caso es idéntico. Conviene, en cualquier caso, explorar siempre la ACF y la PACF de la serie estacionarizada para verificar que los resultados obtenidos son consistentes.

Por último, los modelos se pueden hacer más complejos añadiendo regresores. Tanto la función arima() como la función auto.arima() permiten especificar un vector o una matriz de posibles regresores mediante el argumento xreg.

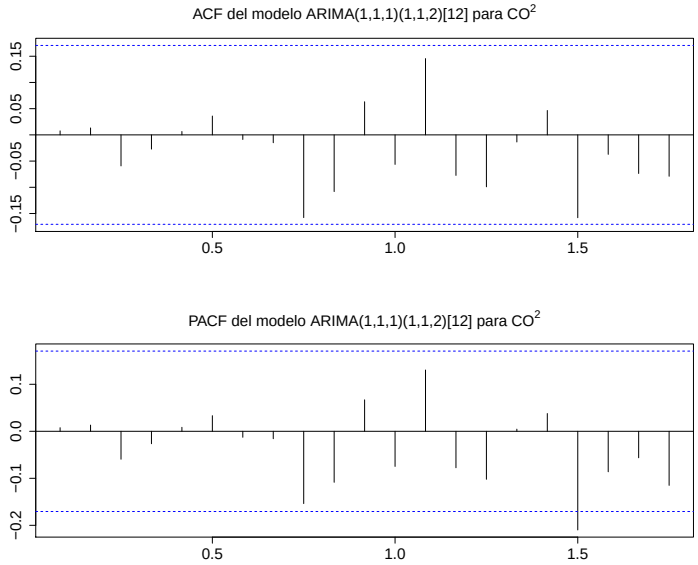
5.3. Validación del modelo

Si una serie está bien identificada, cuando se ajusta el modelo los residuos no deben tener estructura, es decir, deben parecerse a un **ruido blanco**. Un ruido blanco es una serie estacionaria en la que ninguna observación depende de las otras y, por tanto, todos los valores de la ACF y la PACF son nulos. El correlograma y el correlograma parcial deben ser muy similares y los valores no son significativamente distintos de cero.

La validación se basa en:

- La ACF y la PACF de los residuos estandarizados. La ACF y la PACF de los residuos deben ser muy parecidas, no mostrar estructura y tener casi todos los valores dentro de las bandas de confianza.

```
> par(mfcol=c(2,1), cex.axis=1.5, cex.main=1.5)
> tit <- expression(paste("ACF del modelo ARIMA(1,1,1)(1,1,2)[12] para ", CO^2))
> acf(rstandard(arima.co2), xlab="", ylab="", main=tit)
> tit <- expression(paste("PACF del modelo ARIMA(1,1,1)(1,1,2)[12] para ", CO^2))
> pacf(rstandard(arima.co2), xlab="", ylab="", main=tit)
```



Funciones de autocorrelación (ACF) y autocorrelación parcial (PACF) para los residuos estandarizados del modelo ARIMA para la serie de concentraciones de CO_2.

- El contraste de Ljung-Box-Pierce, también conocido como test de *portmanteau*. La hipótesis nula es que las primeras autocorrelaciones son nulas. La hipótesis alternativa de este contraste implica que alguna de las correlaciones es distinta de cero y, por tanto, no se puede asumir que los residuos sean ruido blanco. Podemos utilizar la función Box.test() del paquete stats o la función LB.test() del paquete TSA. El argumento principal de la función Box.test() son los residuos del modelo ARIMA, mientras que en la función LB.test() se hace una llamada directamente al modelo. La segunda modifica la primera por lo que no es esperable encontrar resultados idénticos.

```
> Box.test(rstandard(arima.co2), lag=12, type="Ljung-Box")

        Box-Ljung test

data:  rstandard(arima.co2)
X-squared = 7.1761, df = 12, p-value = 0.8458
```

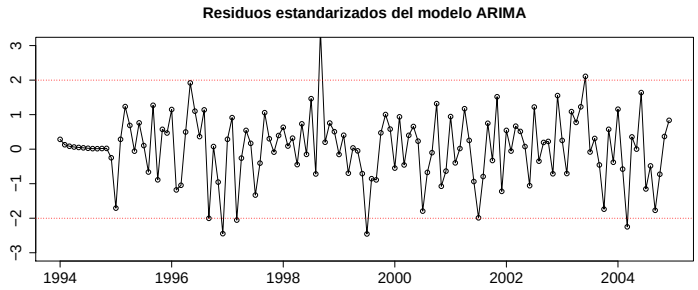
```
> LB.test(arima.co2, lag=12)
```

```
        Box-Ljung test

data:  residuals from  arima.co2
X-squared = 7.051, df = 10, p-value = 0.7206
```

En cualquiera de los dos casos no se puede rechazar la hipótesis nula, por lo que asumimos que las primeras 12 autocorrelaciones son nulas. Podemos cambiar el número de autocorrelaciones con el argumento lag en cualquiera de las dos funciones.

- La representación gráfica de la serie de residuos estandarizados. El gráfico de los residuos debe mostrar que los residuos varían en torno al cero, sin tendencias, la varianza es constante y no hay valores atípicos. Aproximadamente el 95 % de los residuos estandarizados deben estar entre -2 y 2 desviaciones típicas.

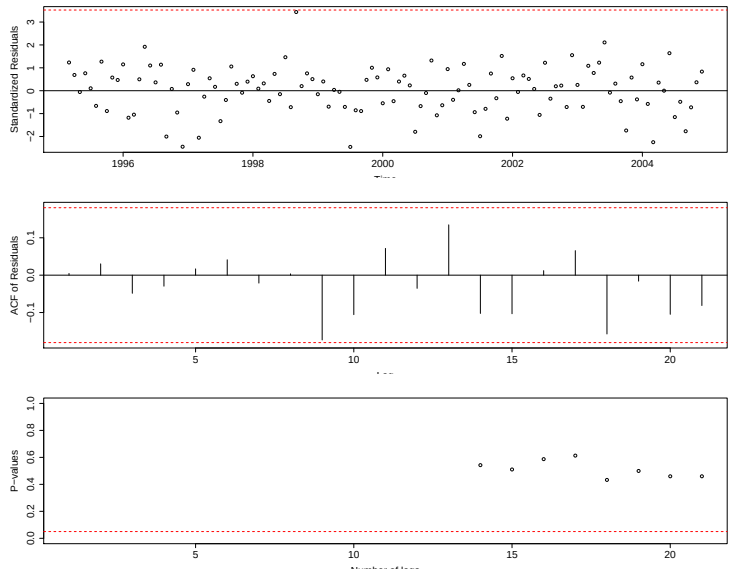
```
> par(cex.axis=1.5, cex.main=1.5, cex.lab=1.5)
> plot(rstandard(arima.co2), xlab="", ylab="", main="", type="o", ylim=c(-3,3))
> title("Residuos estandarizados del modelo ARIMA")
> abline(h=2, lty=3, col="red")
> abline(h=-2, lty=3, col="red")
```



Representación gráfica de los residuos estandarizados del modelo ARIMA para la serie de concentraciones de CO_2.

Finalmente, podemos utilizar la función tsdiag() para sacar todos estos gráficos de forma conjunta. La función tsdiag() es una función genérica y por ello puede dar problemas a la hora de obtener todos los gráficos de forma conjunta especificando únicamente el modelo ARIMA.

```
> par(cex.axis=1.5, cex.main=1.5, cex.lab=1.5)
> tsdiag(arima.co2)
```



Evaluación del modelo ARIMA con la función tsdiag().

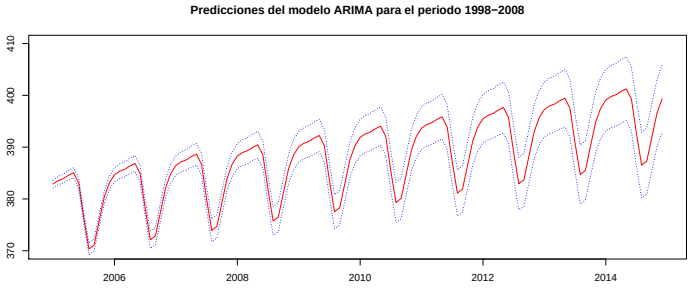
5.4. Predicción de nuevos valores

Para cada predicción tenemos el valor puntual, el error de predicción y el intervalo de predicción.

- La predicción está sujeta al menos a tres fuentes de error:
  - El carácter aleatorio de las innovaciones.
  - Fallos en la especificación del modelo.
  - Erorres en la estimación de los parámetros.
- Con un MA(1) solo se obtiene una predicción que sea distinta de la media marginal de todos los datos.
- Con un AR(1) y un ARMA(1,1) se pueden obtener infinitas predicciones distintas de la media marginal.
- Los errores de estimacrión de los parámetros no modifican sustancialmente los errores de predicción.

Para realizar las predicciones, podemos utilizar la función predict.Arima() del paquete stats.

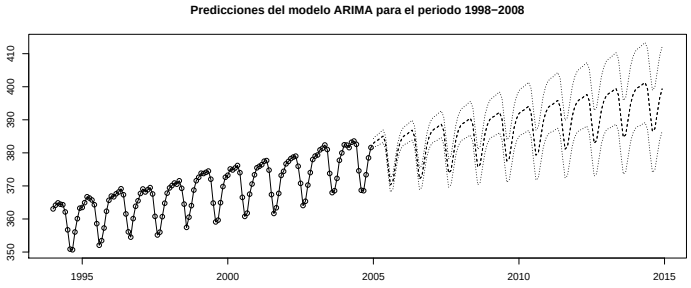
```
> pred <- predict(best.arima, 120)
> tit <- "Predicciones del modelo ARIMA para el periodo 1998-2008"
> plot(pred$pred, xlab="", ylab="", main=tit, col="red", ylim=c(370, 410))
> lines(pred$pred+pred$se, lty=3, col="blue")
> lines(pred$pred-pred$se, lty=3, col="blue")
```



Representación gráfica de las predicciones del modelo ARIMA para la serie de concentraciones de CO_2.

También podemos utilizar la función plot.Arima() del paquete TSA para sacar gráficos de las predicciones. La ventaja de utilizar esta función es que representa en el mismo gráfico la serie histórica junto con las predicciones.

```
> tit <- "Predicciones del modelo ARIMA para el periodo 1998-2008"
> plot(arima.co2, n.ahead=120, xlab="", ylab="", main=tit, type="l")
```



Representación gráfica de la serie temporal de concentraciones de CO_2 y las predicciones del modelo ARIMA.

6. Referencias

- Box, G.E.P., Jenkins, G.M. & Reinsel, G. (1996). Time series analysis: forecasting and control. Prentice-Hall.
- Cryer, J.D. & Chank, K.-S. (2008). Time series analysis with applications in R. Second Edition. Springer, USA.

- Peña Sánchez, D. (2005). Análisis de series temporales. Alianza Editorial, Madrid.
- Shumway, R.H. & Stoffer, D.S. (2006). Time series analysis and its applications with R examples. Springer, USA.