SUMMARY SESSION 6

Victor Miguel Terrón Macias

30/1/2021

EJEMPLOS DE SERTIES DE TIEMPO Y TECNICAS DESCRIPTIVAS

Las series de tiempo son analizadas para entender el pasado y para predecir el futuro, permitiendo a los administradores o creadores de políticas tomar decisiones informadas propiamente.

Cuando una variable es medida secuencialmente en el tiempo durante o en un intervalo fijo, conocido como el intervalo de muestreo, los datos que resultan forman una serie de tiempo.

El número de reservas de pasajeros internacionales (en miles) por mes en una aerolínea (Pan Am) en los Estados Unidos fue obtenida de la Administración de Aviación Federal para el periodo 1949-1960 (Brown, 1963). La compañía usó los datos para predecir la demanda futura antes de ordenar nuevos aviones y tripulación de entrenamiento.

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

Figure 1: Imagen

Tendencia. - En general, un cambio sistemático en una serie de tiempo que no parece ser periódica es conocida como una tendencia.

Variación estacional. - Un patrón que se repite dentro de cada año es conocido como variación estacional, aunque el término es aplicado más generalmente para patrones que se repiten dentro de cualquier periodo fijo.

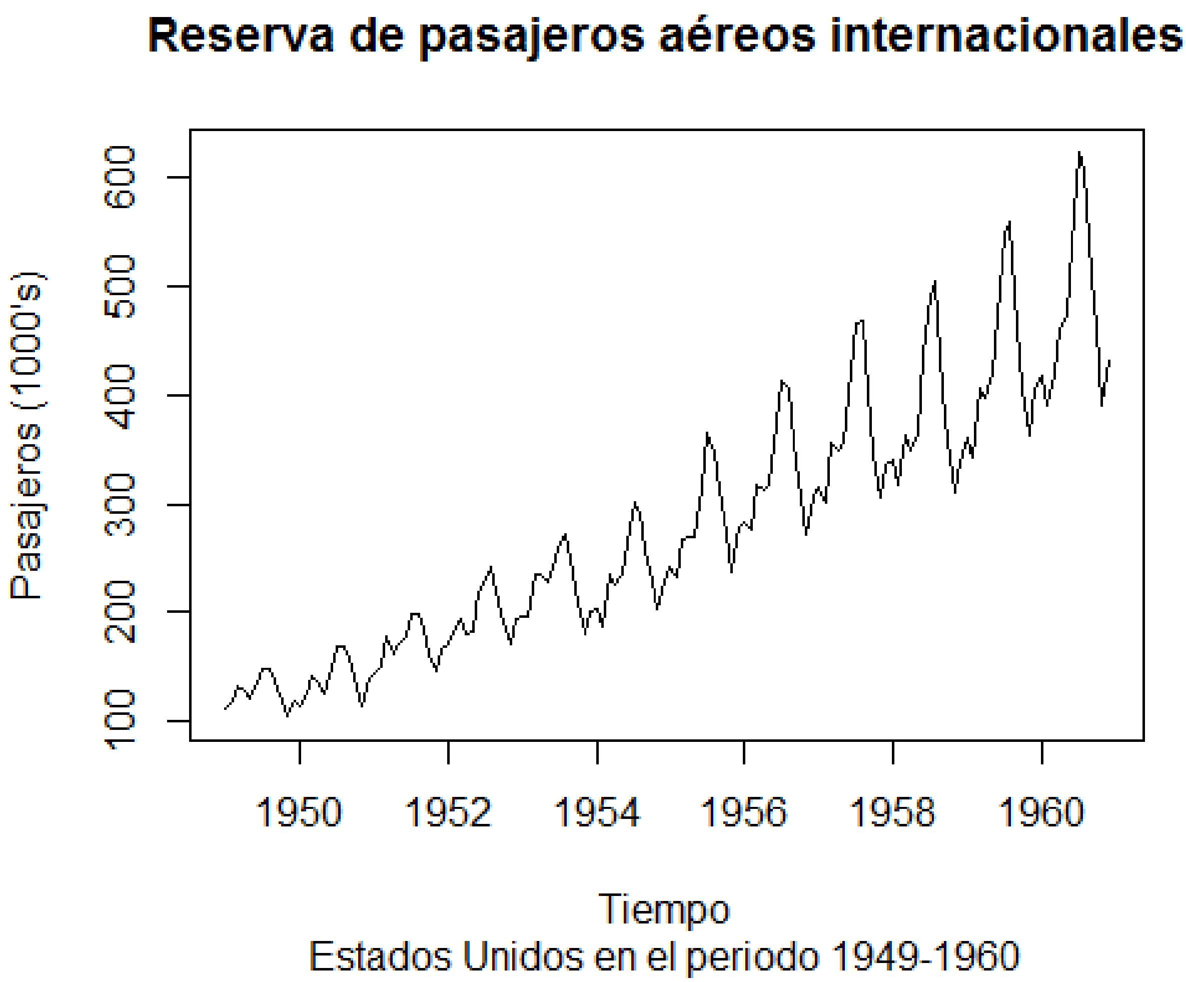


Figure 2: Imagen

Ciclos. - A veces podemos afirmar que hay ciclos en una serie de tiempo que no corresponden a algún periodo natural fijo.

Los pronósticos confían en la extrapolación, y los pronósticos están generalmente basados en la suposición de que las tendencias presentes continuarán. No podemos verificar esta suposición en alguna forma empírica, pero si podemos identificar causas probables para una tendencia, podemos justificar extrapolarla, por algunos pocos pasos en el tiempo al menos.

Un pronóstico es un valor futuro que se predice, y el número de pasos de tiempo hacia el futuro es el tiempo de espera (k).

Hay varias formas para estimar la tendencia mt al tiempo t, pero un procedimiento relativamente simple, el cual está disponible en R y no asume ninguna forma específica es calcular un promedio móvil centrado en xt.

La longitud de pronósticos promedio móvil es elegida para promediar los efectos estacionales, los cuales pueden ser estimados después.

Un segundo algoritmo de suavizamiento ofrecido por R es stl. Este usa una técnica de regresión ponderada localmente conocida como loess.

Procedimientos de suavizamiento tales como el promedio móvil centrado y loess no requieren un modelo predeterminado, pero no producen una fórmula que pueda ser extrapolada para dar pronósticos

En R, la función decompose estima tendencias y efectos estacionales usando un método de promedio móvil.

MODELOS ESTADÍSTICOS BÁSICOS, MODELOS ESTACIONARIOS Y PREDICCIÓN

La función de autocovarianza sólo identifica un único proceso MA(q) si la condición de que el proceso sea invertible es impuesta.

MODELOS NO ESTACIONARIOS Y PREDICCIÓN

SESION 6. SERIES DE TIEMPO

OBJETIVO

Obtener predicciones de ventas, de la demanda de productos o servicios, que ayuden a las organizaciones a ajustar presupuestos, justificar decisiones de marketing, o tener un mejor conocimiento sobre la evolución del negocio.

En esta sesión estudiaremos temas relacionados con los siguientes puntos:

- Datos que provienen de una variable que es medida secuencialmente en el tiempo durante o en un intervalo fijo (series de tiempo)
- Análisis de series de tiempo y ajustes de modelos para entender el pasado y predecir el futuro

EJEMPLO1. Ejemplos de series de tiempo y técnicas descriptivas

OBJETIVO

Entender el concepto de serie de tiempo y tomar consciencia sobre los problemas que podrían ayudar a resolver al analizarlas.

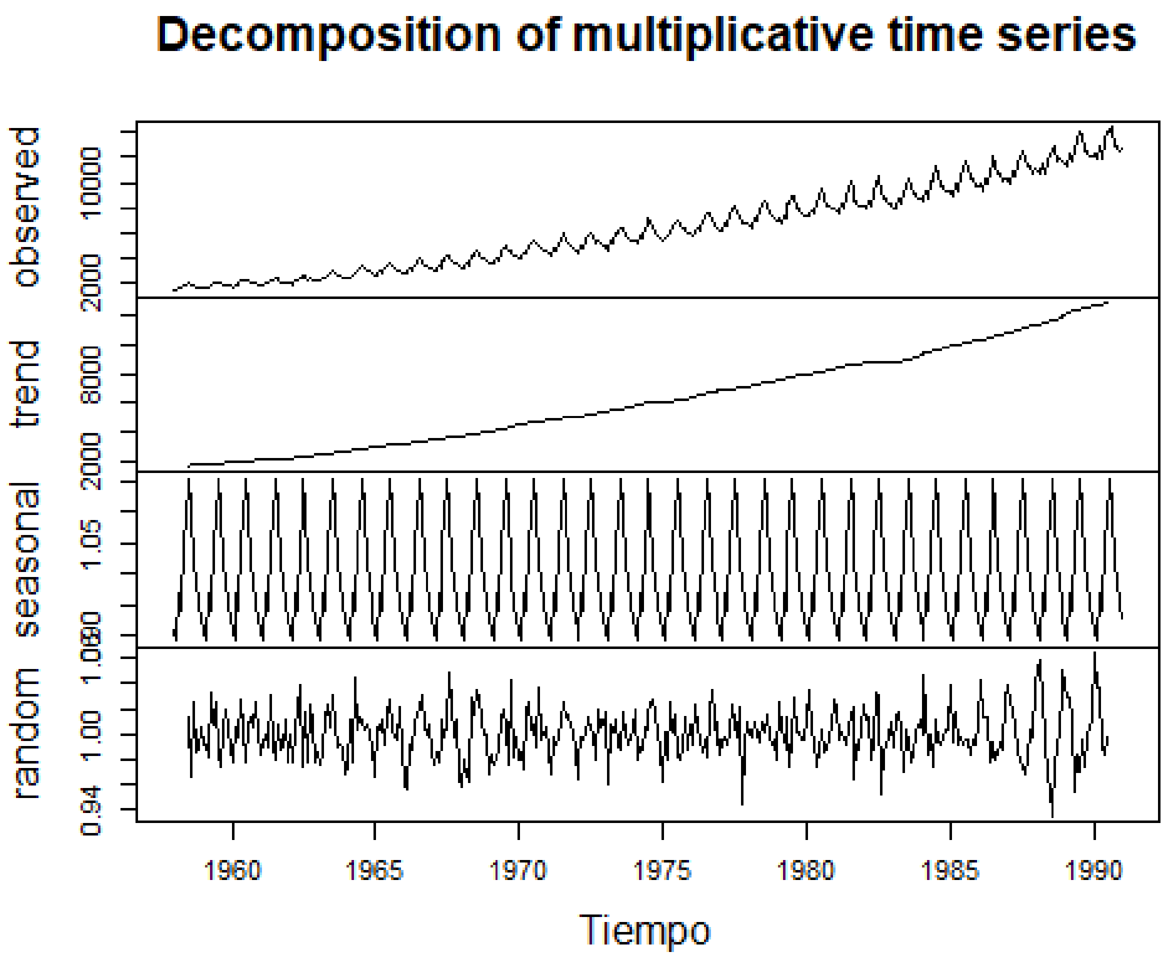


Figure 3: Imagen

Ruido Blanco. Una serie de tiempo $\{w_t : t = 1,\ 2,\ \cdots,\ n\}$ es ruido blanco discreto si las variables $w_1, w_2,\ \cdots, w_n$ son *independientes* e *idénticamente* distribuidas con una media de cero.

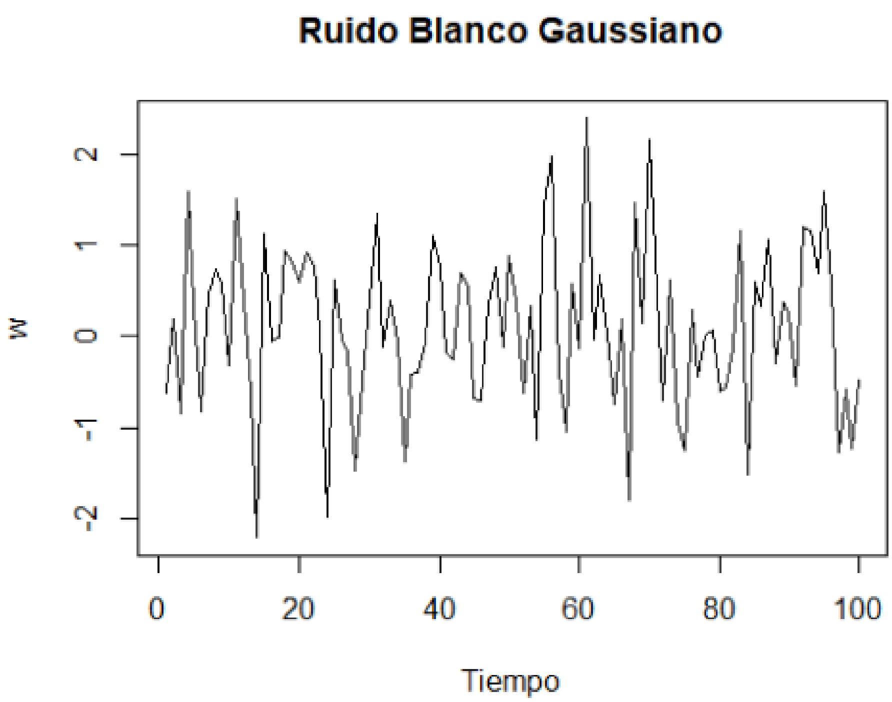


Figure 4: Imagen

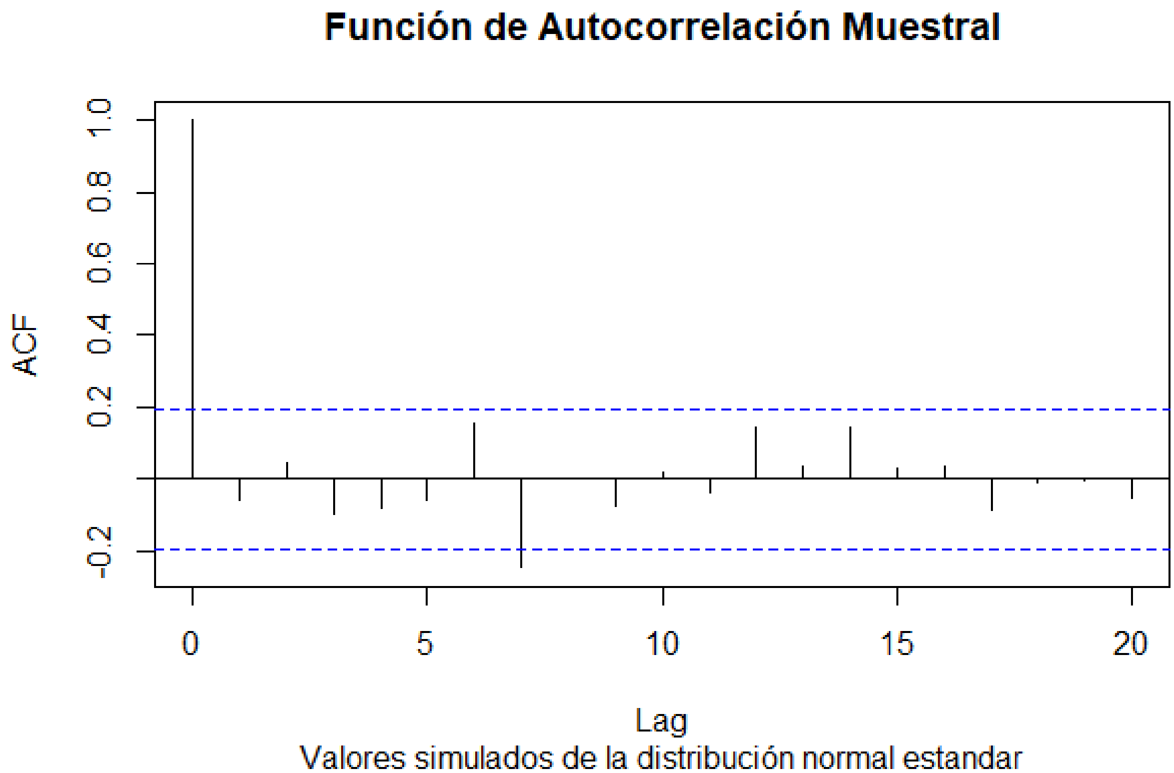


Figure 5: Imagen

Caminata Aleatoria. Sea $\{x_t\}$ una serie de tiempo. Entonces $\{x_t\}$ es una caminata aleatoria si

$$x_t = x_{t-1} + w_t$$

Donde $\{w_t\}$ es una serie de ruido blanco.

Operador de rezago. El *operador de rezago* **B** se define como

$$\mathbf{B}\, x_t = x_{t-1}$$

Operador diferencia. El *operador diferencia* ∇ se define como

$$\nabla x_t = x_t - x_{t-1}$$

Es decir,

$$\nabla x_t = (1 - B)x_t$$

Figure 6: Imagen

DESARROLLO

En este ejemplo se hará la visualización y descomposición de series de tiempo, además de poder determinar la tendencia de nuestros datos a lo largo de un periodo de tiempo determinado, las series de tiempo son recolectadas y sirven para estudios de tipo retrospectivo, además también se pueden realizar predicciones a futuro. A continuación verás aplicaciones que te serán de mucha utilidad.

Técnicas descriptivas: gráficas, tendencias y variación estacional El primer dataset corresponde a las ventas mensuales de un filtro de aceite en diversos concesionarios para equipo de construcción.

library(TSA) data(oilfilters); plot(oilfilters, type = "o", ylab = "Ventas", xlab = "Tiempo", main = "Ventas Mesuales") plot(oilfilters, type = "l", ylab = "Ventas", xlab = "Tiempo", main = "Ventas Mensuales de Filtro de Aceite", sub = "Símbolos Especiales") points(y = oilfilters, x = time(oilfilters), pch = as.vector(season(oilfilters))) Ahora utilizaremos el dataset AirPAssengers, el cual contiene datos de serie de tiempo correspondiente a pasajeros aéreos.

data(AirPassengers) AP <- AirPassengers AP Clase de un objeto

class(AP)

start(AP); end(AP); frequency(AP)

summary(AP)

plot(AP, ylab = "Pasajeros (1000's)", xlab = "Tiempo", main = "Reserva de pasajeros aéreos internacionales", sub = "Estados Unidos en el periodo 1949-1960") layout(1:2) plot(aggregate(AP), xlab = "Tiempo", main = "Reserva de pasajeros aéreos internacionales", sub = "Estados Unidos en el periodo 1949-1960")

boxplot(AP ~ cycle(AP), xlab = "Boxplot de valores estacionales", sub = "Estados Unidos en el periodo

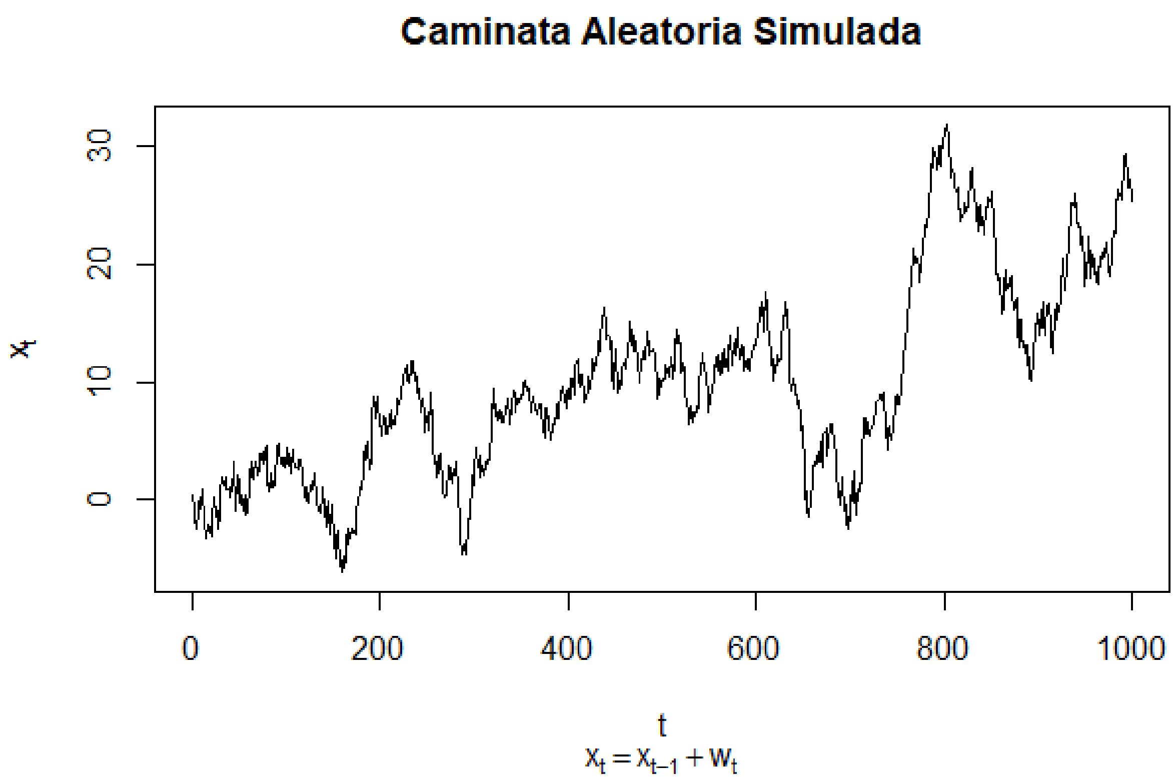


Figure 7: Imagen

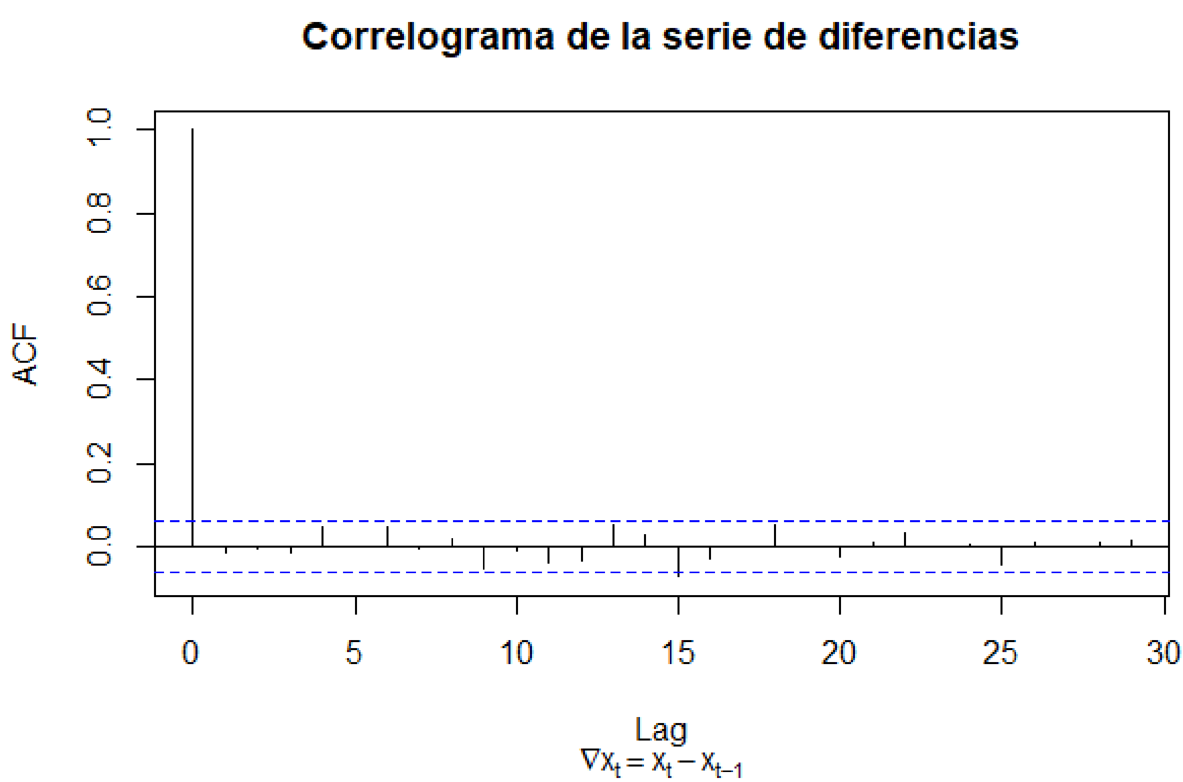


Figure 8: Imagen

Si una serie sigue una caminata aleatoria, la serie de diferencias será ruido blanco.

La serie $\{x_t\}$ es un **proceso autorregresivo de orden p**, abreviado como **AR(p)**, si

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \cdots + \alpha_p x_{t-p} + w_t$$

Dónde $\{w_t\}$ es ruido blanco y las α_i son los parámetros del modelo con $\alpha_p \neq 0$ para un proceso de orden p.

Es de utilidad notar los siguientes puntos:

(a) La caminata aleatoria es el caso especial AR(1) con $\alpha_1 = 1$

(b) El modelo es una regresión de x_t sobre términos pasados de la misma serie; de ahí el uso del término autorregresivo.

(c) Una predicción al tiempo t está dada por

$$\hat{x}_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \cdots + \alpha_p x_{t-p}$$

Figure 9: Imagen

1949-1960", main = "Reserva de pasajeros aéreos internacionales") dev.off() Algunos datos en https://github.com/AtefOuni/ts/tree/master/Data

Series de Tiempo Múltiple Serie de producción de electricidad, cerveza y chocolate

CBE <- read.csv("cbe.csv", header = TRUE) CBE[1:4,] class(CBE)

Elec.ts <- ts(CBE[, 3], start = 1958, freq = 12) Beer.ts <- ts(CBE[, 2], start = 1958, freq = 12) Choc.ts <- ts(CBE[, 1], start = 1958, freq = 12)

Electricidad <- Elec.ts Cerveza <- Beer.ts Chocolate <- Choc.ts

plot(cbind(Electricidad, Cerveza, Chocolate), main = "Producción de Chocolate, Cerveza y Electricidad", xlab = "Tiempo", sub = "Enero de 1958 - Diciembre de 1990") Serie de temperatura global

Global <- scan("global.txt") Global.ts <- ts(Global, st = c(1856, 1), end = c(2005, 12), fr = 12) Global.annual <- aggregate(Global.ts, FUN = mean) plot(Global.ts, xlab = "Tiempo", ylab = "Temperatura en °C", main = "Serie de Temperatura Global", sub = "Serie mensual: Enero de 1856 a Diciembre de 2005") plot(Global.annual, xlab = "Tiempo", ylab = "Temperatura en °C", main = "Serie de Temperatura Global", sub = "Serie anual de temperaturas medias: 1856 a 2005") New.series <- window(Global.ts, start = c(1970, 1), end = c(2005, 12)) New.time <- time(New.series) plot(New.series, xlab = "Tiempo", ylab = "Temperatura en °C", main = "Serie de Temperatura Global", sub = "Serie mensual: Enero de 1970 a Diciembre de 2005"); abline(reg = lm(New.series ~ New.time)) Descomposición de series CBE <- read.csv("cbe.csv", header = TRUE) Elec.ts <- ts(CBE[, 3], start = 1958, freq = 12) Modelo Aditivo

Elec.decom.A <- decompose(Elec.ts)

plot(Elec.decom.A, xlab = "Tiempo", sub = "Descomposición de los datos de producción de electricidad") Componentes

Tendencia <- Elec.decom.A$trend Estacionalidad < -Elec.decom.A$seasonal Aleatorio <- Elec.decom.A$random

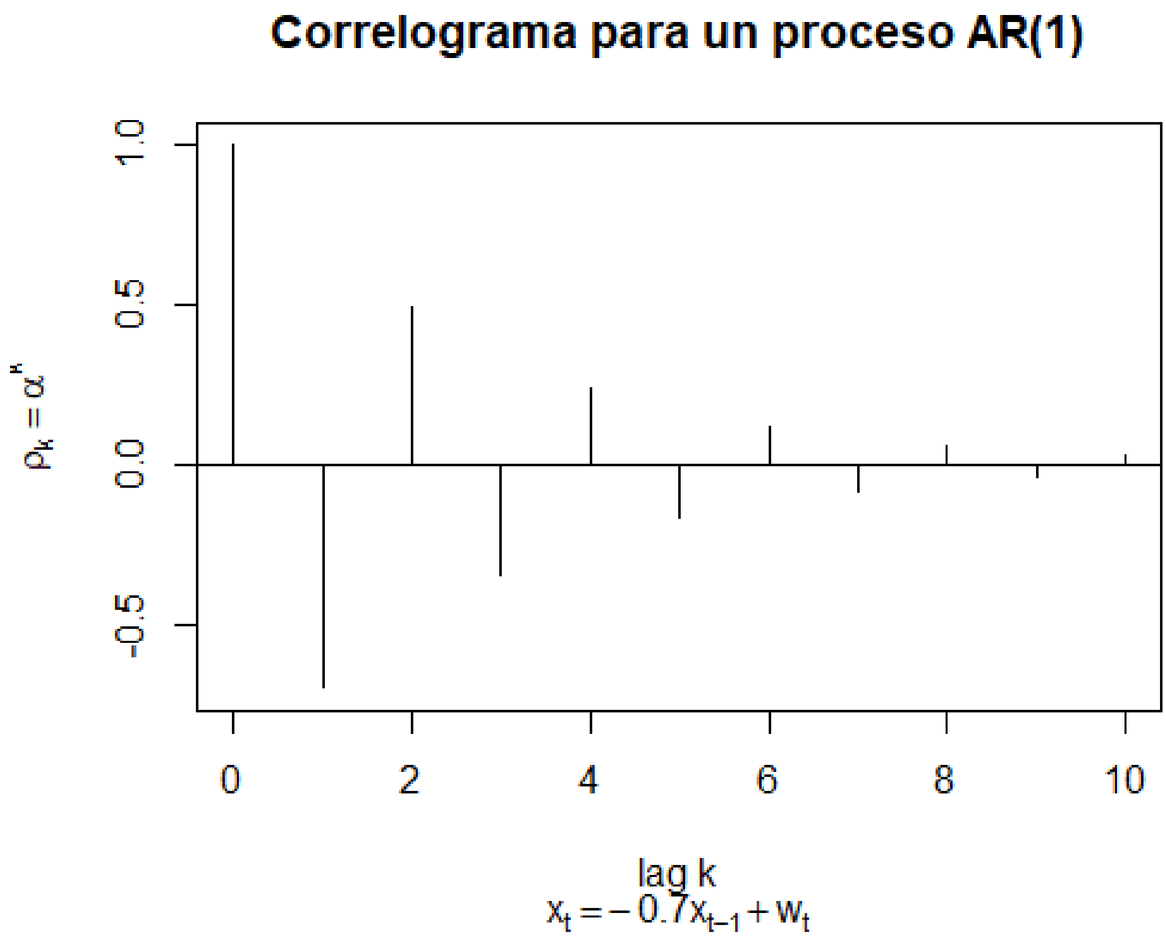


Figure 10: Imagen

Autocorrelación Parcial. La autocorrelación parcial en el retraso k es la correlación que resulta después de eliminar el efecto de cualquier correlación debido a los términos en retrasos más cortos.

Un proceso AR(p) tiene un correlograma de autocorrelaciones parciales que son cero después del retraso p.

Un **proceso de promedios móviles de orden q**, abreviado como **MA(q)**, es una combinación lineal del término de ruido blanco actual y los q términos pasados de ruido blanco más recientes y se define como

$$x_t = w_t + \beta_1 w_{t-1} + \cdots + \beta_q w_{t-q}$$

Dónde $\{w_t\}$ es ruido blanco con media cero y varianza σ_w^2.

Debido a que un proceso MA consiste de una suma finita de términos de ruido blanco estacionarios, estos son estacionarios y así tienen una media y autocovarianza invariante en el tiempo.

Un proceso MA es **invertible** si puede expresarse como un proceso autorregresivo estacionario de orden infinito sin un término de error. Por ejemplo, el proceso MA $x_t = w_t - \beta w_{t-1}$ puede expresarse como

$$w_t = x_t + \beta x_{t-1} + \beta^2 x_{t-2} + \cdots$$

Siempre que $|\beta| < 1$, lo cual se requiere para la convergencia.

Figure 11: Imagen

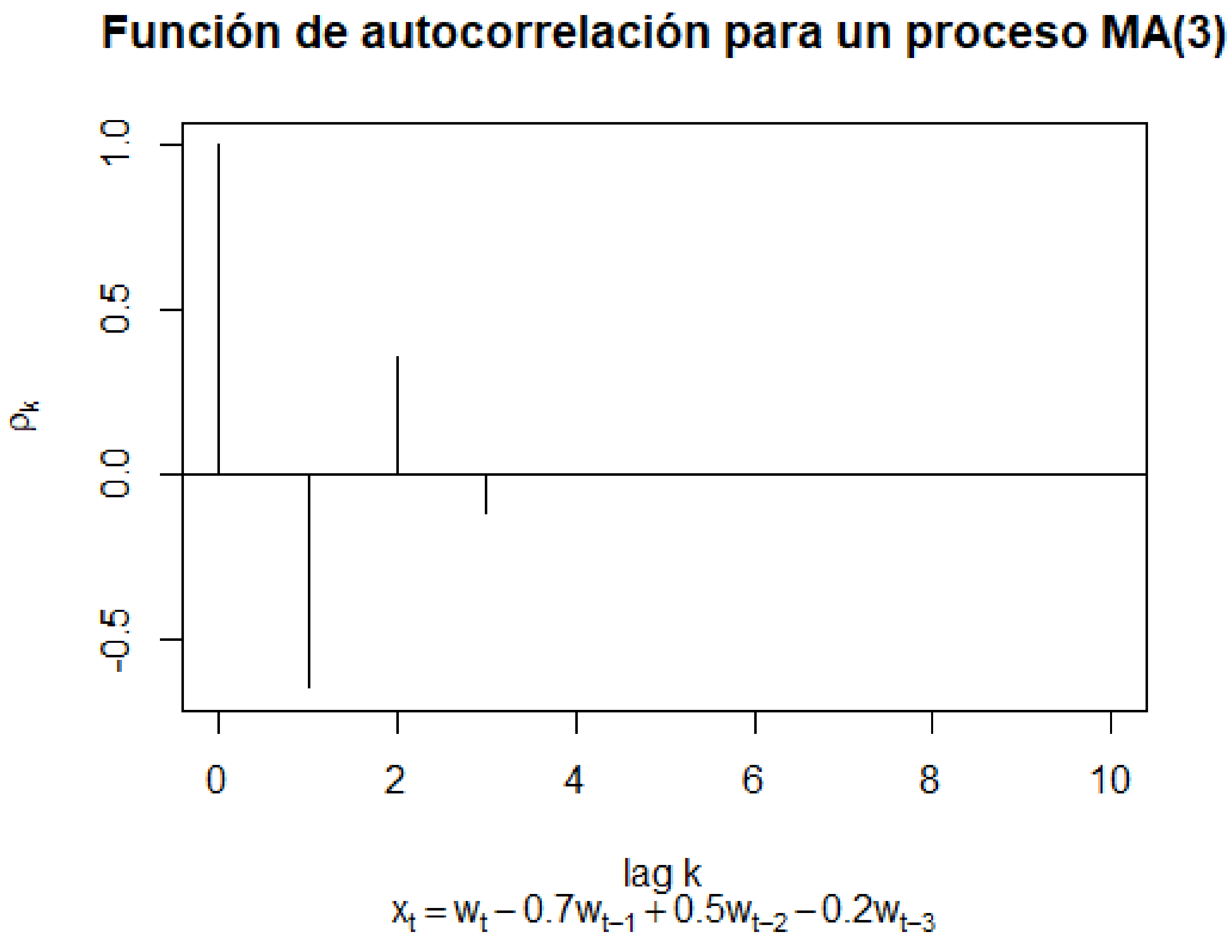


Figure 12: Imagen

Una serie de tiempo $\{x_t\}$ sigue un proceso (ARMA) autorregresivo de promedios móviles de órden (p, q), denotado ARMA(p, q), cuando

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \cdots + \alpha_p x_{t-p} + w_t + \beta_1 w_{t-1} + \beta_2 w_{t-2} + \cdots + \beta_q w_{t-q}$$

Donde $\{w_t\}$ es ruido blanco.

Es de utilidad notar algunos puntos acerca de un proceso ARMA(p, q):

(a) El modelo AR(p) es el caso especial ARMA(p, 0).

(b) El modelo MA(q) es el caso especial ARMA(0, q).

(c) *Parsimonia de parámetros.* Cuando se ajusta a los datos, un modelo ARMA frecuentemente será más eficiente en parámetros (es decir, requiere menos parámetros) que un único modelo MA o AR.

Figure 13: Imagen

Diferenciar una serie $\{x_t\}$ puede eliminar tendencias, tanto si estas tendencias son estocásticas, como en una caminata aleatoria, o determinísticas, como en el caso de una tendencia lineal.

Una serie $\{x_t\}$ es **integrada** de orden d, denotada como **I**(d), si después de aplicar d veces el operador diferencia a $\{x_t\}$, esta nueva serie diferenciada resulta ser ruido blanco $\{w_t\}$.

Es decir,

$$\nabla^d x_t = w_t$$

Como

$$\nabla^d \equiv (1 - B)^d$$

Una serie $\{x_t\}$ es **integrada de orden d** si

$$(1 - B)^d x_t = w_t$$

Modelo ARIMA. Una serie de tiempo $\{x_t\}$ sigue un proceso ARIMA(p, d, q), si después de aplicar d veces el operador diferencia ∇ a $\{x_t\}$, la nueva serie que resulta es un proceso ARMA(p, q).

Un **modelo ARIMA estacional** usa diferenciación a un retraso igual al número de estaciones (s) para eliminar efectos estacionales aditivos.

Figure 14: Imagen

El siguiente modelo

$$x_t = x_{t-1} + \alpha x_{t-12} - \alpha x_{t-13} + w_t$$

Que también se puede escribir como

$$\left(1 - \alpha B^{12}\right)(1 - B)\, x_t = w_t$$

Es un modelo

$$ARIMA(0,\ 1,\ 0)(1,\ 0,\ 0)_{12}$$

Debemos tener presentes que diferenciar a un retraso s eliminará una tendencia lineal, así que tendremos como elección si incluimos diferenciar a un retraso 1 o no.

Podemos elegir un modelo de mejor ajuste usando un criterio apropiado tal como el AIC. Una vez que un modelo de mejor ajuste ha sido encontrado, debemos de verificar que el correlograma de los residuales corresponda al correlograma de ruido blanco.

Figure 15: Imagen

ts.plot(cbind(Tendencia, Tendencia + Estacionalidad), xlab = "Tiempo", main = "Datos de Producción de Electricidad", ylab = "Producción de electricidad", lty = 1:2, sub = "Tendencia con efectos estacionales aditivos sobrepuestos")

Tendencia[20] + Estacionalidad[20] + Aleatorio[20] Elec.ts[20] Modelo Multiplicativo

Elec.decom.M <- decompose(Elec.ts, type = "mult")

plot(Elec.decom.M, xlab = "Tiempo", sub = "Descomposición de los datos de producción de electricidad") Componentes

Trend <- Elec.decom.M$trend Seasonal < -Elec.decom.M$seasonal Random <- Elec.decom.M\$random

ts.plot(cbind(Trend, Trend*Seasonal), xlab = "Tiempo", main = "Datos de Producción de Electricidad", ylab = "Producción de electricidad", lty = 1:2, sub = "Tendencia con efectos estacionales multiplicativos sobrepuestos")

Trend[7]*Seasonal[7]*Random[7] Elec.ts[7]

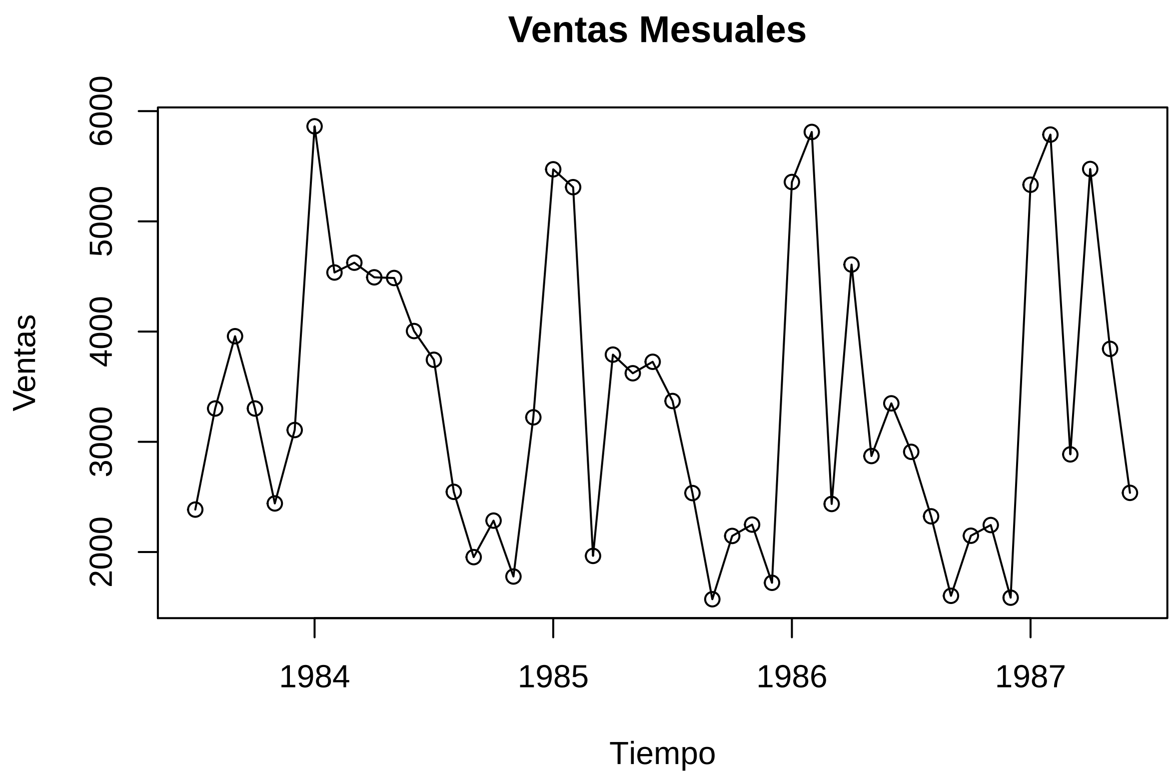
Trend[100]*Seasonal[100]*Random[100] Elec.ts[100]

```r
#### Técnicas descriptivas: gráficas, tendencias y variación estacional

library(TSA)
```

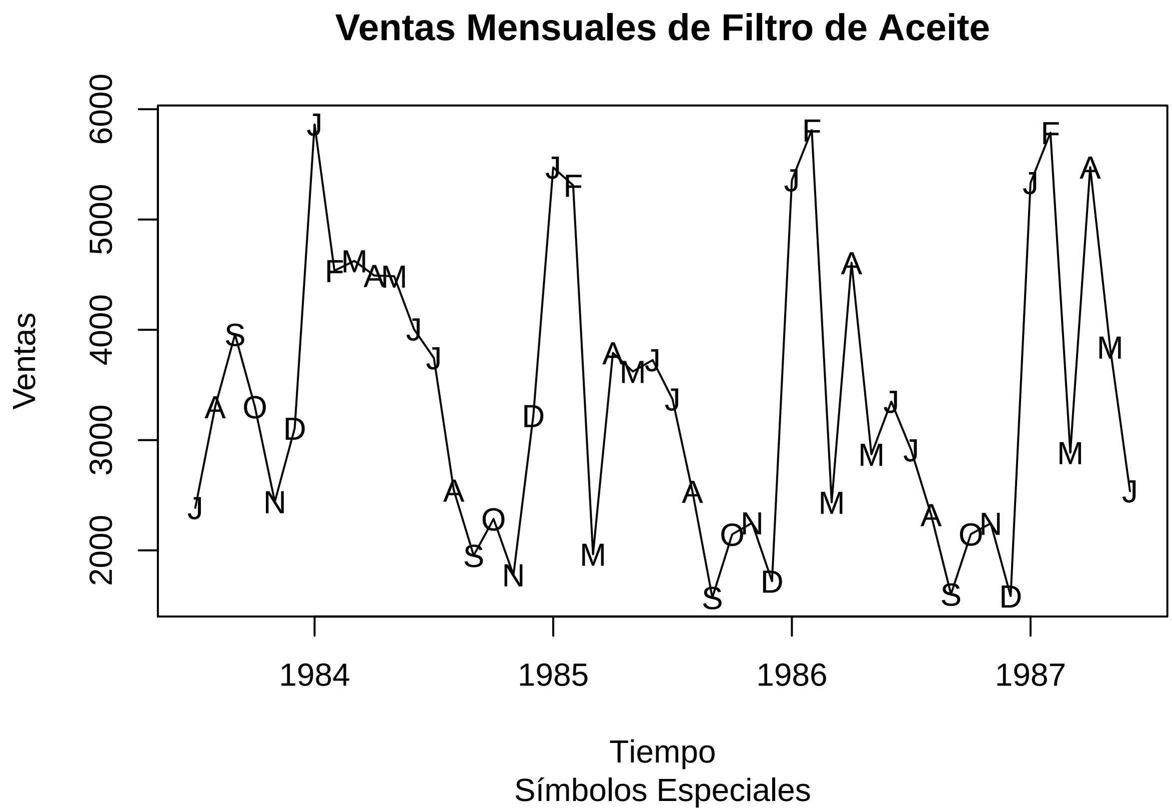
```
Attaching package: 'TSA'

The following objects are masked from 'package:stats':

    acf, arima
```

```
The following object is masked from 'package:utils':

    tar
```

```
data(oilfilters); plot(oilfilters, type = "o", ylab = "Ventas", xlab = "Tiempo", main = "Ventas Mesuales
```




```
plot(oilfilters, type = "l", ylab = "Ventas", xlab = "Tiempo",
                 main = "Ventas Mensuales de Filtro de Aceite",
                 sub = "Símbolos Especiales")
points(y = oilfilters, x = time(oilfilters),
pch = as.vector(season(oilfilters)))
```




```r
data(AirPassengers)
AP <- AirPassengers
AP
```

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

```r
# Clase de un objeto

class(AP)
```

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

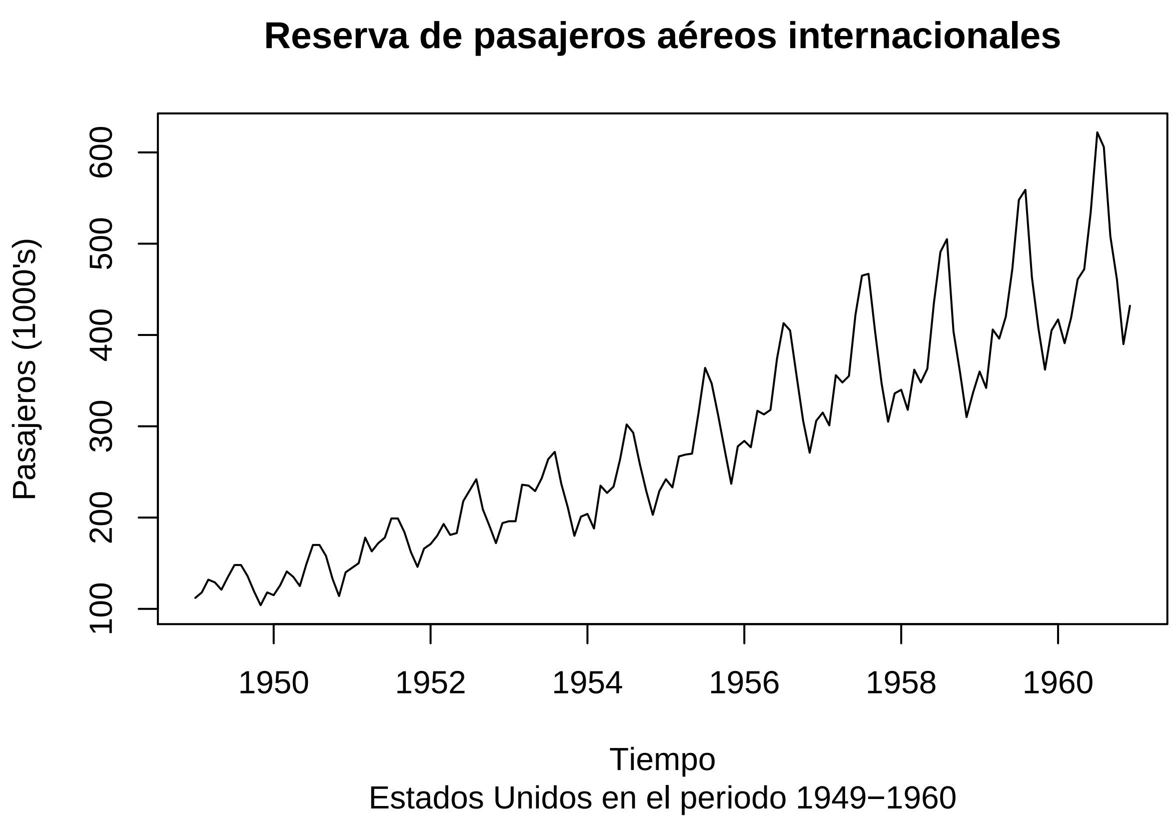
```r
start(AP); end(AP); frequency(AP)
```

```
[1] 1949    1

[1] 1960   12

[1] 12
```

```r
summary(AP)
```

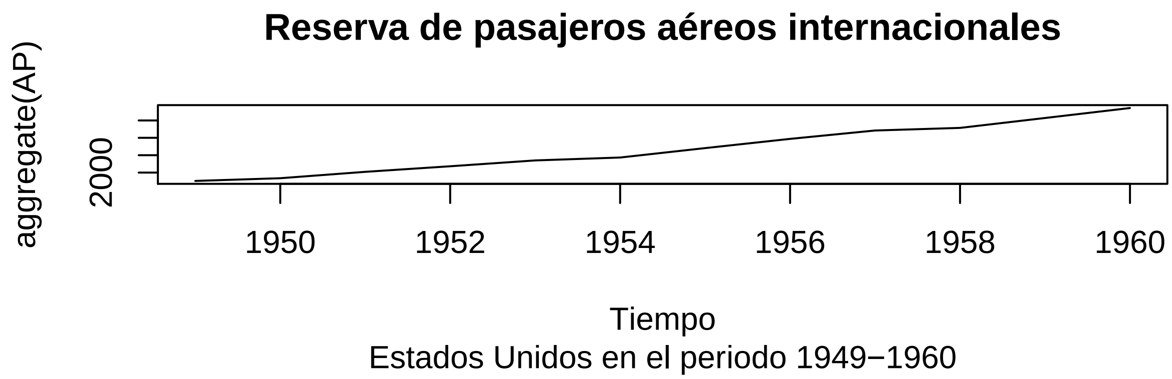
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  104.0   180.0   265.5   280.3   360.5   622.0
```

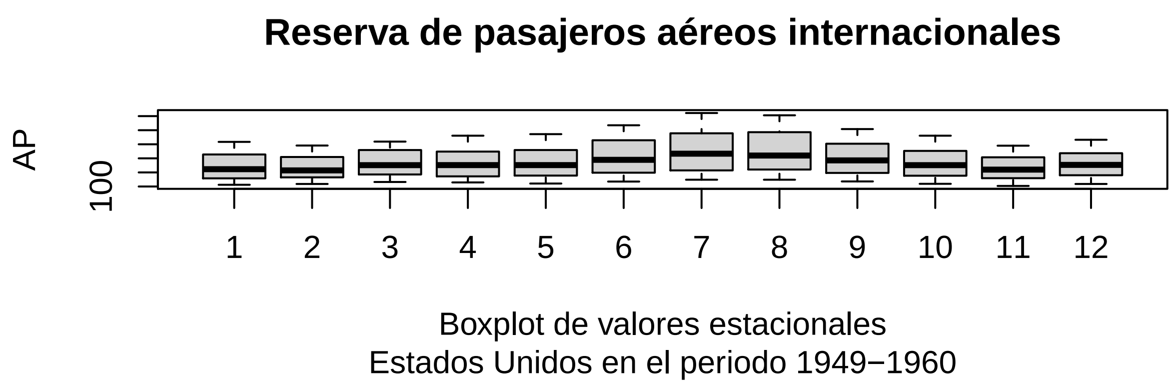
```r
plot(AP, ylab = "Pasajeros (1000's)", xlab = "Tiempo",
     main = "Reserva de pasajeros aéreos internacionales",
     sub = "Estados Unidos en el periodo 1949-1960")
```




```r
###############################################

layout(1:2)
plot(aggregate(AP), xlab = "Tiempo",
     main = "Reserva de pasajeros aéreos internacionales",
     sub = "Estados Unidos en el periodo 1949-1960")
```

```r
boxplot(AP ~ cycle(AP),
        xlab = "Boxplot de valores estacionales",
        sub = "Estados Unidos en el periodo 1949-1960",
        main = "Reserva de pasajeros aéreos internacionales")
```






```r
dev.off()
```

```
null device
          1
```

```r
################################################

# https://github.com/AtefOuni/ts/tree/master/Data

# Series de Tiempo Múltiple

# Serie de producción de electricidad, cerveza y chocolate

CBE <- read.csv("cbe.csv", header = TRUE)
CBE[1:4,]
```

```
  choc beer elec
1 1451 96.3 1497
2 2037 84.4 1463
```

```
3 2477 91.2 1648
4 2785 81.9 1595
```

```r
class(CBE)
```

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

```r
Elec.ts <- ts(CBE[, 3], start = 1958, freq = 12)
Beer.ts <- ts(CBE[, 2], start = 1958, freq = 12)
Choc.ts <- ts(CBE[, 1], start = 1958, freq = 12)

Electricidad <- Elec.ts
Cerveza <- Beer.ts
Chocolate <- Choc.ts

plot(cbind(Electricidad, Cerveza, Chocolate),
     main = "Producción de Chocolate, Cerveza y Electricidad",
     xlab = "Tiempo",
     sub = "Enero de 1958 - Diciembre de 1990")

################################################

# Serie de Temperatura Global

Global <- scan("global.txt")
Global.ts <- ts(Global, st = c(1856, 1), end = c(2005, 12), fr = 12)
Global.annual <- aggregate(Global.ts, FUN = mean)
plot(Global.ts, xlab = "Tiempo", ylab = "Temperatura en Â°C", main = "Serie de Temperatura Global",
     sub = "Serie mensual: Enero de 1856 a Diciembre de 2005")
plot(Global.annual, xlab = "Tiempo", ylab = "Temperatura en Â°C", main = "Serie de Temperatura Global",
     sub = "Serie anual de temperaturas medias: 1856 a 2005")

################################################

New.series <- window(Global.ts, start = c(1970, 1), end = c(2005, 12))
New.time <- time(New.series)
plot(New.series, xlab = "Tiempo", ylab = "Temperatura en Â°C", main = "Serie de Temperatura Global",
     sub = "Serie mensual: Enero de 1970 a Diciembre de 2005"); abline(reg = lm(New.series ~ New.time))


#### Descomposición de series

CBE <- read.csv("cbe.csv", header = TRUE)
Elec.ts <- ts(CBE[, 3], start = 1958, freq = 12)

# Modelo Aditivo

Elec.decom.A <- decompose(Elec.ts)

plot(Elec.decom.A, xlab = "Tiempo",
     sub = "Descomposición de los datos de producción de electricidad")

# Componentes
```

```r
Tendencia <- Elec.decom.A$trend
Estacionalidad <- Elec.decom.A$seasonal
Aleatorio <- Elec.decom.A$random

ts.plot(cbind(Tendencia, Tendencia + Estacionalidad),
        xlab = "Tiempo", main = "Datos de Producción de Electricidad",
        ylab = "Producción de electricidad", lty = 1:2,
        sub = "Tendencia con efectos estacionales aditivos sobrepuestos")

Tendencia[20] + Estacionalidad[20] + Aleatorio[20]
```

```
[1] 2016
```

```r
Elec.ts[20]
```

```
[1] 2016
```

```r
###

# Modelo Multiplicativo

Elec.decom.M <- decompose(Elec.ts, type = "mult")

plot(Elec.decom.M, xlab = "Tiempo",
     sub = "Descomposición de los datos de producción de electricidad")

# Componentes

Trend <- Elec.decom.M$trend
Seasonal <- Elec.decom.M$seasonal
Random <- Elec.decom.M$random

ts.plot(cbind(Trend, Trend*Seasonal), xlab = "Tiempo", main = "Datos de Producción de Electricidad",
        ylab = "Producción de electricidad", lty = 1:2,
        sub = "Tendencia con efectos estacionales multiplicativos sobrepuestos")

Trend[7]*Seasonal[7]*Random[7]
```

```
[1] 1994
```

```r
Elec.ts[7]
```

```
[1] 1994
```

```r
Trend[100]*Seasonal[100]*Random[100]
```

```
[1] 3033
```

```
Elec.ts[100]
```

```
[1] 3033
```

```
# J. Cryer & K. Chan. (2008). Time Series Analysis With Applications
# in R. 233 Spring Street, New York, NY 10013, USA: Springer
# Science+Business Media, LLC.

# P. Cowpertwait & A. Metcalfe. (2009). Introductory Time Series with R.
# 233 Spring Street, New York, NY 10013, USA: Springer Science+Business
# Media, LLC.
```

EJEMPLO 2. Modelos estocásticos básicos, modelos estacionarios y predicción

OBJETIVO

Estudiar algunos modelos estocásticos básicos, modelos estacionarios y realizar predicciones de algunas series de tiempo.

DESARROLLO

En el desarrollo de este ejemplo se van a tratar tópicos como: ruido blanco, caminatas aleatorias, operadores de rezago y diferencia.

Ruido Blanco y simulación en R

set.seed(1) w <- rnorm(100) plot(w, type = "l", xlab = "") title(main ="Ruido Blanco Gaussiano", xlab ="Tiempo") Para ilustrar mediante simulación como las muestras pueden diferir de sus poblaciones subyacentes considere lo siguiente

x <- seq(-3, 3, length = 1000) hist(rnorm(100), prob = T, ylab = "", xlab ="", main ="") points(x, dnorm(x), type ="l") title(ylab ="Densidad", xlab ="Valores simulados de la distribución normal estandar", main ="Comparación de una muestra con su población subyacente") set.seed(2) acf(rnorm(100), main ="") title(main ="Función de Autocorrelación Muestral", sub ="Valores simulados de la distribución normal estandar") Caminata aleatoria y simulación en R

x <- w <- rnorm(1000) for(t in 2:1000) x[t] <- x[t-1] + w[t] plot(x, type = "l", main = "Caminata Aleatoria Simulada", xlab = "t", ylab = expression(x[t]), sub = expression(x[t]==x[t-1]+w[t])) acf(x, main = "") title(main ="Correlograma para la caminata aleatoria simulada", sub = expression(x[t]==x[t-1]+w[t])) Modelos ajustados y gráficas de diágnostico, series de caminatas aleatorias simuladas. El correlograma de las series de diferencias puede usarse para evaluar si una serie dada puede modelarse como una caminata aleatoria

acf(diff(x), main = "") title(main ="Correlograma de la serie de diferencias", sub = expression(nabla*x[t]==x[t]-x[t-1])) Modelos AR(p), MA(q) y ARMA(p, q) Modelos AR(p)

Correlograma de un proceso AR(1)

rho <- function(k, alpha) alpha^k plot(0:10, rho(0:10, 0.7), type = "h", ylab = "", xlab ="") title(main ="Correlograma para un proceso AR(1)", ylab = expression(rho[k] == alpha^k), xlab ="lag k", sub = expression(x[t]==0.7*x[t-1]+w[t])) plot(0:10, rho(0:10, -0.7), type = “h”, ylab = "“, xlab =”“) title(main =”Correlograma para un proceso AR(1)“, ylab = expression(rho[k] == alpha^k), xlab =”lag k", sub = expression(x[t]==-0.7*x[t-1]+w[t])) abline(h = 0) Simulación en R

Un proceso AR(1) puede ser simulado en R como sigue:

set.seed(1) x <- w <- rnorm(100) for(t in 2:100) x[t] <- 0.7 * x[t-1] + w[t] plot(x, type = “l”, xlab = "“, ylab =”“) title(main =”Proceso AR(1) simulado“, xlab =”Tiempo“, ylab = expression(x[t]), sub = expression(x[t]==0.7*x[t-1]+w[t])) acf(x, main = "") title(main = "Correlograma del proceso AR(1) simulado", sub = expression(x[t]==0.7*x[t-1]+w[t])) pacf(x, main =”“) title(main =”Correlograma Parcial del proceso AR(1) simulado", sub = expression(x[t]==0.7*x[t-1]+w[t])) Modelos Ajustados

Ajuste de modelos a series simuladas

x.ar <- ar(x, method = “mle”) x.ar$order x.ar$ar x.ar$ar + c(-2, 2) * sqrt(x.ar$asy.var) Serie de temperatura global: Ajuste de un modelo AR

Global <- scan(“global.txt”) Global.ts <- ts(Global, st = c(1856, 1), end = c(2005, 12), fr = 12) Global.annual <- aggregate(Global.ts, FUN = mean) plot(Global.ts, xlab = “Tiempo”, ylab = “Temperatura en °C”, main = “Serie de Temperatura Global”, sub = “Serie mensual: Enero de 1856 a Diciembre de 2005”) plot(Global.annual, xlab = “Tiempo”, ylab = “Temperatura en °C”, main = “Serie de Temperatura Global”, sub = “Serie anual de temperaturas medias: 1856 a 2005”) mean(Global.annual) Global.ar <- ar(Global.annual, method = “mle”) Global.ar$order Global.ar$ar acf(Global.ar$res[-(1 : Global.ar$order)], lag = 50, main = "“) title(main =”Correlograma de la serie de residuales“, sub =”Modelo AR(4) ajustado a la serie de temperaturas globales anuales") Modelos MA(q)

Ejemplos en R: Correlograma y Simulación

Función en R para calcular la Función de Autocorrelación

rho <- function(k, beta){ q <- length(beta) - 1 if(k > q) ACF <- 0 else { s1 <- 0; s2 <- 0 for(i in 1:(q-k+1)) s1 <- s1 + beta[i]*beta[i + k] for(i in 1:(q+1)) s2 <- s2 + beta[i]^2 ACF <- s1/s2} ACF} Correlograma para un proceso MA(3)

beta <- c(1, 0.7, 0.5, 0.2) rho.k <- rep(1, 10) for(k in 1:10) rho.k[k] <- rho(k, beta) plot(0:10, c(1, rho.k), ylab = expression(rho[k]), xlab = “lag k”, type = “h”, sub = expression(x[t] == w[t] + 0.7*w[t-1] + 0.5*w[t-2] + 0.2*w[t-3]), main = “Función de autocorrelación para un proceso MA(3)”) abline(0, 0) Correlograma para otro proceso MA(3)

beta <- c(1, -0.7, 0.5, -0.2) rho.k <- rep(1, 10) for(k in 1:10) rho.k[k] <- rho(k, beta) plot(0:10, c(1, rho.k), ylab = expression(rho[k]), xlab = “lag k”, type = “h”, sub = expression(x[t] == w[t] - 0.7*w[t-1] + 0.5*w[t-2] - 0.2*w[t-3]), main = “Función de autocorrelación para un proceso MA(3)”) abline(0, 0) Simulación de un proceso MA(3)

set.seed(1) b <- c(0.8, 0.6, 0.4) x <- w <- rnorm(1000) for(t in 4:1000){ for(j in 1:3) x[t] <- x[t] + b[j]*w[t-j]
}

plot(x, type = “l”, ylab = expression(x[t]), xlab = “Tiempo t”, sub = expression(x[t] == w[t] + 0.8*w[t-1] + 0.6*w[t-2] + 0.4*w[t-3]), main = “Serie de tiempo simulada de un proceso MA(3)”) acf(x, main = "“) title(main =”Correlograma para un proceso MA(3) simulado", sub = expression(x[t] == w[t] + 0.8*w[t-1] + 0.6*w[t-2] + 0.4*w[t-3])) Ajuste de modelos MA

x.ma <- arima(x, order = c(0, 0, 3)) x.ma Modelos ARMA(p, q)

Simulación y ajuste

set.seed(1) x <- arima.sim(n = 10000, list(ar = -0.6, ma = 0.5)) plot(x[1:100], type = “l”, xlab = "“) title(main =”Serie simulada“, xlab =”Tiempo", sub = expression(x[t] == -0.6*x[t-1] + w[t] + 0.5*w[t-1])) coef(arima(x, order = c(1, 0, 1))) Predicción Serie de producción de electricidad

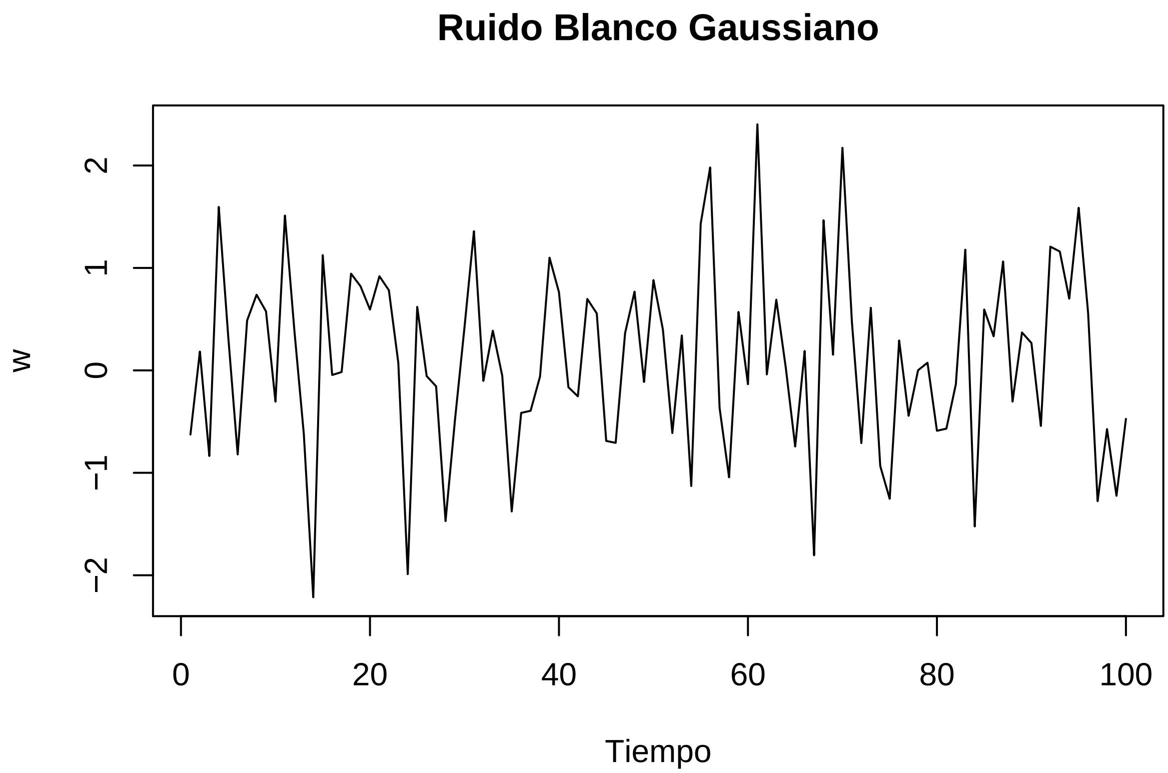
CBE <- read.csv(“cbe.csv”, header = TRUE) Elec.ts <- ts(CBE[, 3], start = 1958, freq = 12) plot(Elec.ts, xlab = "“, ylab =”“) title(main =”Serie de Producción de Electricidad“, xlab =”Tiempo“, ylab =”Producción de electricidad“) plot(log(Elec.ts), xlab =”“, ylab =”“) title(main =”Serie-log de Producción de

Electricidad“, xlab =”Tiempo“, ylab =”Log de Producción de electricidad“) Time <- 1:length(Elec.ts) Imth <- cycle(Elec.ts) Elec.lm <- lm(log(Elec.ts) ~ Time + I(Time^2) + factor(Imth)) acf(resid(Elec.lm), main =”“) title(main =”Correlograma de la serie de residuales del modelo de regresión“, sub =”Serie de producción de electricidad“) plot(resid(Elec.lm), type =”l“, main =”“, xlab =”“, ylab =”“) title(main =”Serie de residuales del modelo de regresión ajustado“, sub =”Serie de producción de electricidad“, xlab =”Tiempo“, ylab =”Residuales") Código para encontrar el mejor modelo ARMA(p, q) considerando el AIC (Akaike Information Criterion)

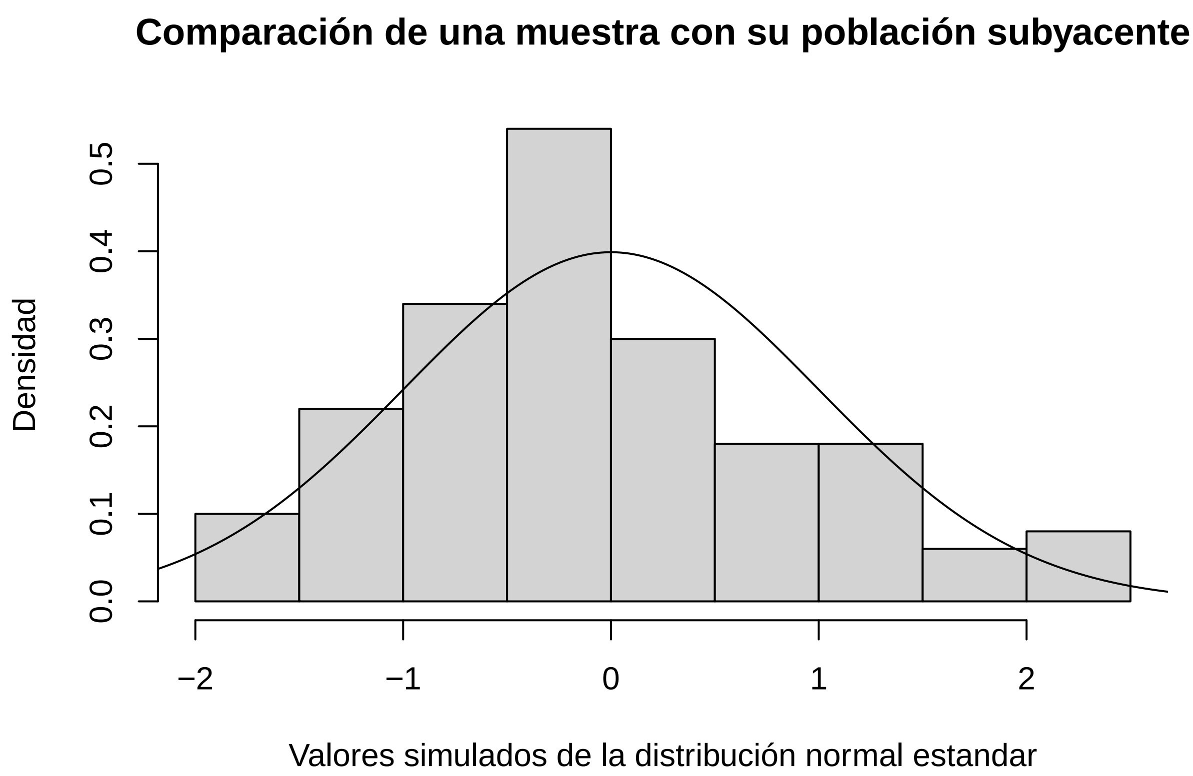
best.order <- c(0, 0, 0) best.aic <- Inf for(i in 0:2)for(j in 0:2){ model <- arima(resid(Elec.lm), order = c(i, 0, j)) fit.aic <- AIC(model) if(fit.aic < best.aic){ best.order <- c(i, 0, j) best.arma <- arima(resid(Elec.lm), order = best.order) best.aic <- fit.aic } }

best.order acf(resid(best.arma), main = "“) title(main =”Serie de residuales del modelo ARMA(2, 0) ajustado“, sub =”Serie de residuales del modelo de regresión ajustado a los datos de electricidad“) new.time <- seq(length(Elec.ts)+1, length = 36) new.data <- data.frame(Time = new.time, Imth = rep(1:12, 3)) predict.lm <- predict(Elec.lm, new.data) predict.arma <- predict(best.arma, n.ahead = 36) elec.pred <- ts(exp(predict.lm + predict.arma$pred), start = 1991, freq = 12) ts.plot(cbind(Elec.ts, elec.pred), lty = 1:2, col = c(”blue“,”red“), xlab =”Tiempo“, ylab =”Producción de electricidad“, main =”Predicción de los datos de producción de electricidad“, sub =”Predicción de 36 meses")

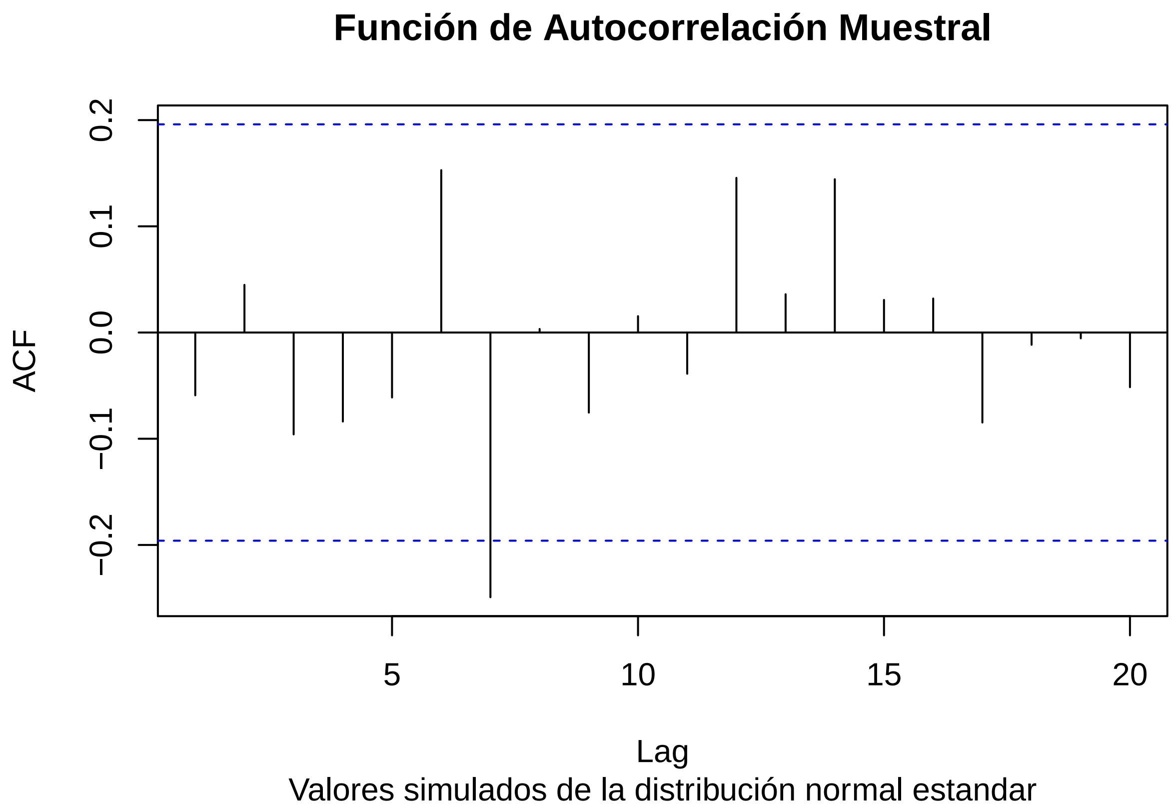
```r
# Ejemplo 2. Modelos estocásticos básicos, modelos estacionarios y predicción

# Ruido Blanco y simulación en R

set.seed(1)
w <- rnorm(100)
plot(w, type = "l", xlab = "")
title(main = "Ruido Blanco Gaussiano", xlab = "Tiempo")
```



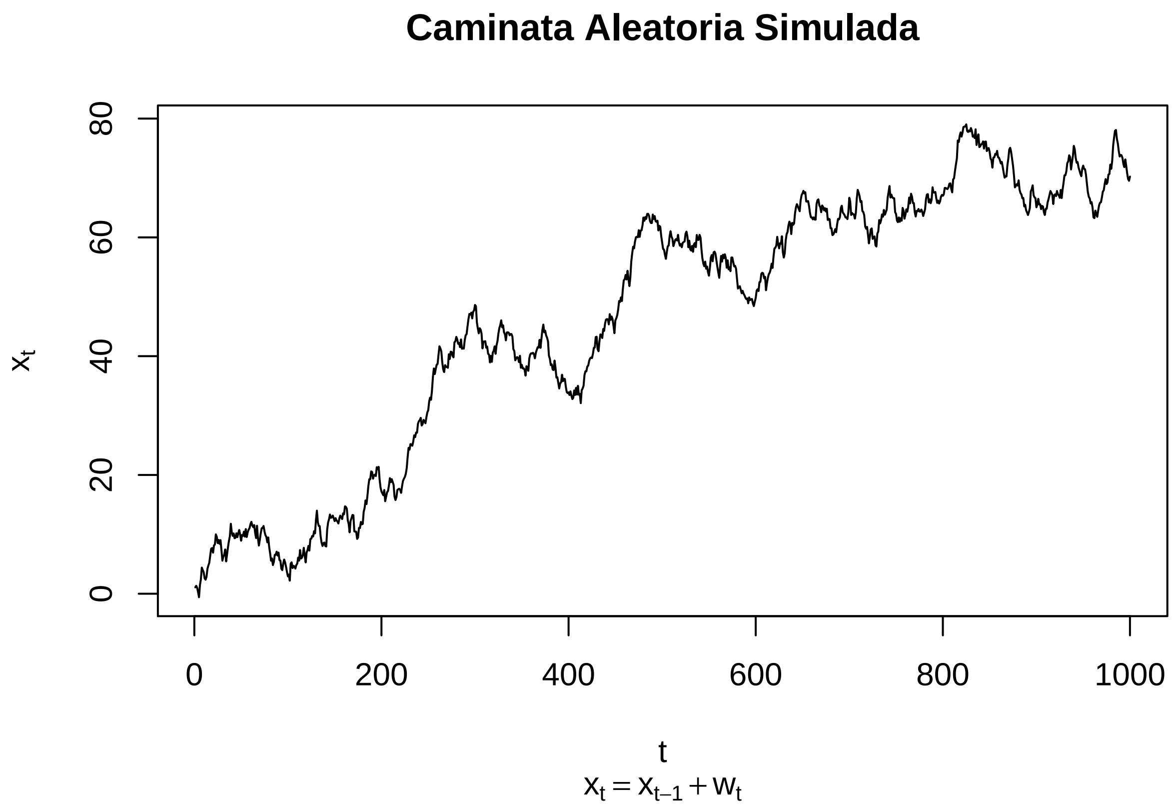

```r
###

# Para ilustrar mediante simulación como las muestras pueden diferir
# de sus poblaciones subyacentes considere lo siguiente

x <- seq(-3, 3, length = 1000)
hist(rnorm(100), prob = T, ylab = "", xlab = "", main = "")
points(x, dnorm(x), type = "l")
title(ylab = "Densidad", xlab = "Valores simulados de la distribución normal estandar",
      main = "Comparación de una muestra con su población subyacente")
```



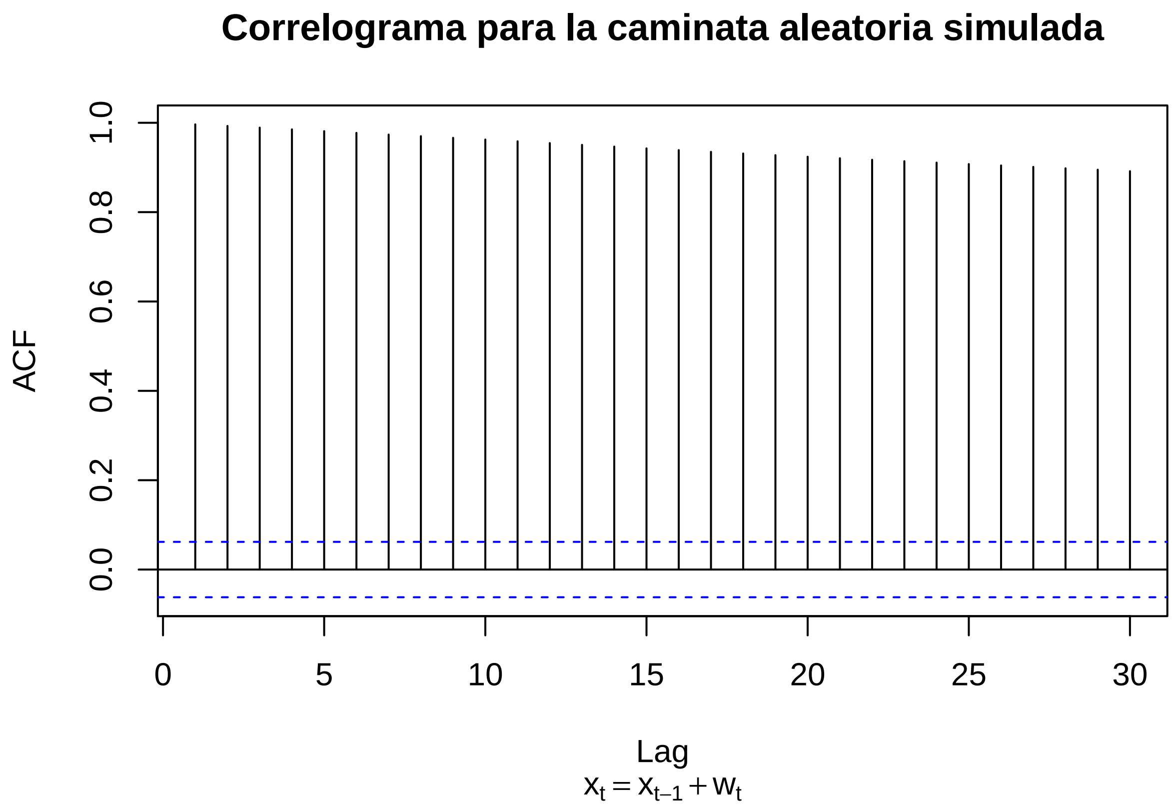

```
###

set.seed(2)
acf(rnorm(100), main = "")
title(main = "Función de Autocorrelación Muestral",
      sub = "Valores simulados de la distribución normal estandar")
```



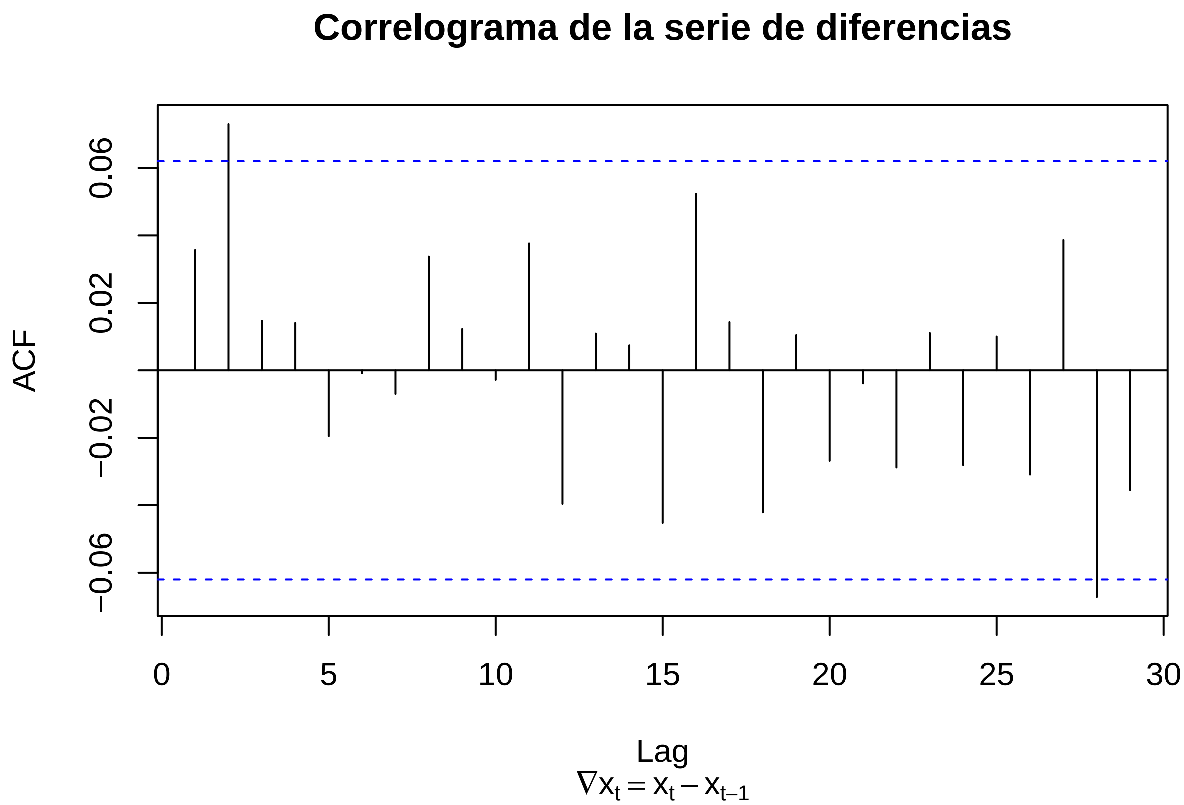

```r
###

              # Caminata Aleatoria
              # Simulación en R

x <- w <- rnorm(1000)
for(t in 2:1000) x[t] <- x[t-1] + w[t]
plot(x, type = "l", main = "Caminata Aleatoria Simulada",
     xlab = "t", ylab = expression(x[t]),
     sub = expression(x[t]==x[t-1]+w[t]))
```



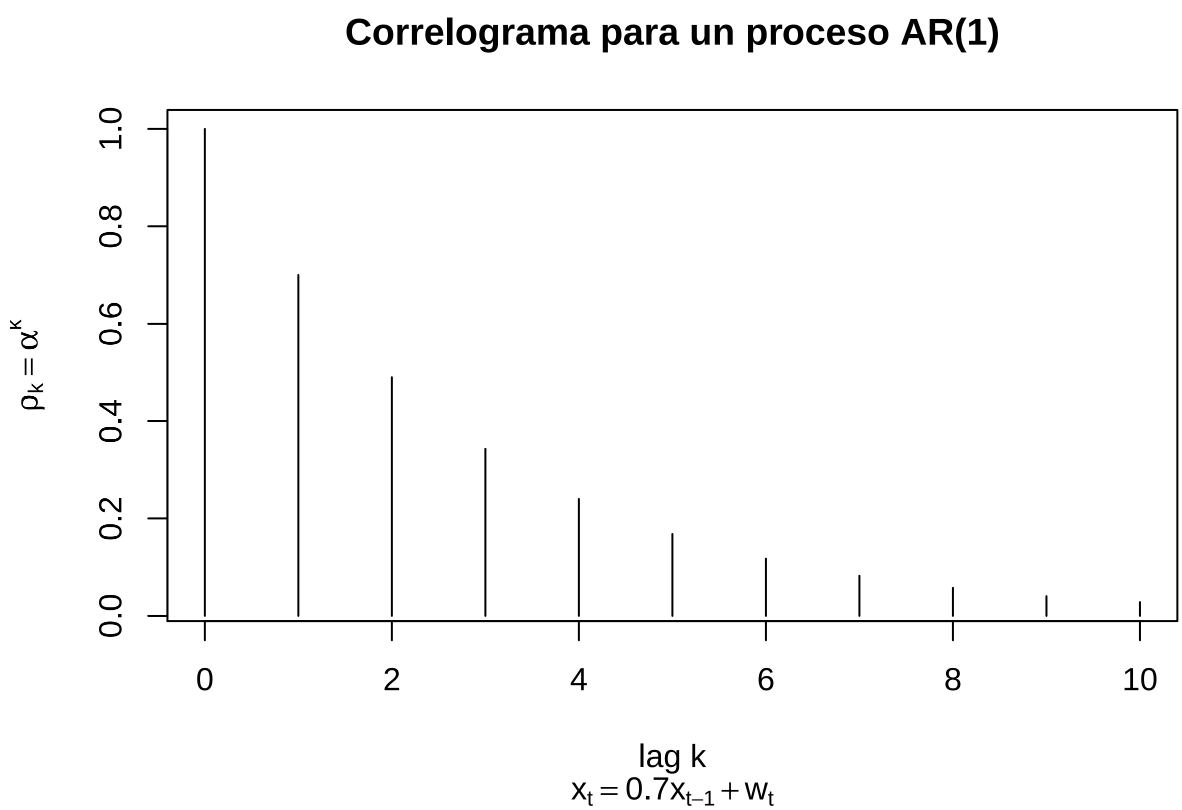

```
acf(x, main = "")
title(main = "Correlograma para la caminata aleatoria simulada",
      sub = expression(x[t]==x[t-1]+w[t]))
```



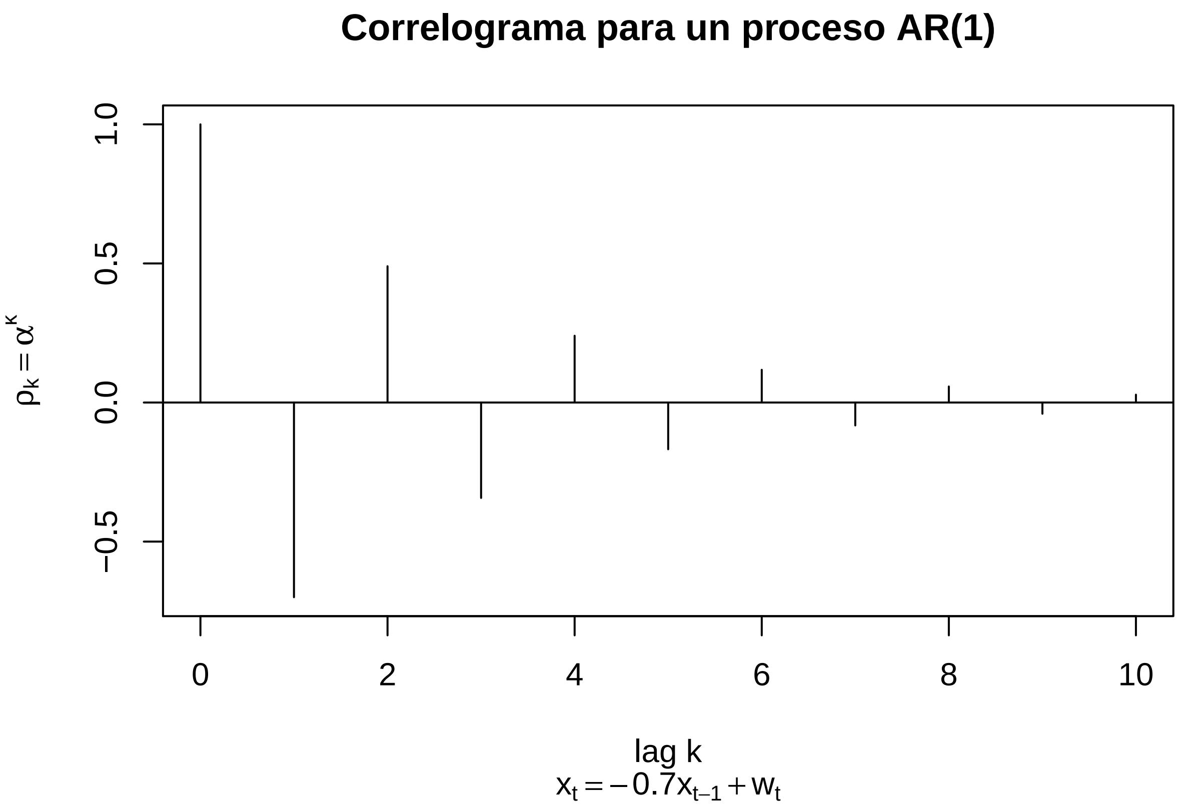

```
###

        # Modelos Ajustados y gráficas de diagnóstico
      # Series de caminatas aleatorias simuladas

# El correlograma de las series de diferencias puede usarse para evaluar si una serie dada
# puede modelarse como una caminata aleatoria

acf(diff(x), main = "")
title(main = "Correlograma de la serie de diferencias",
      sub = expression(nabla*x[t]==x[t]-x[t-1]))
```



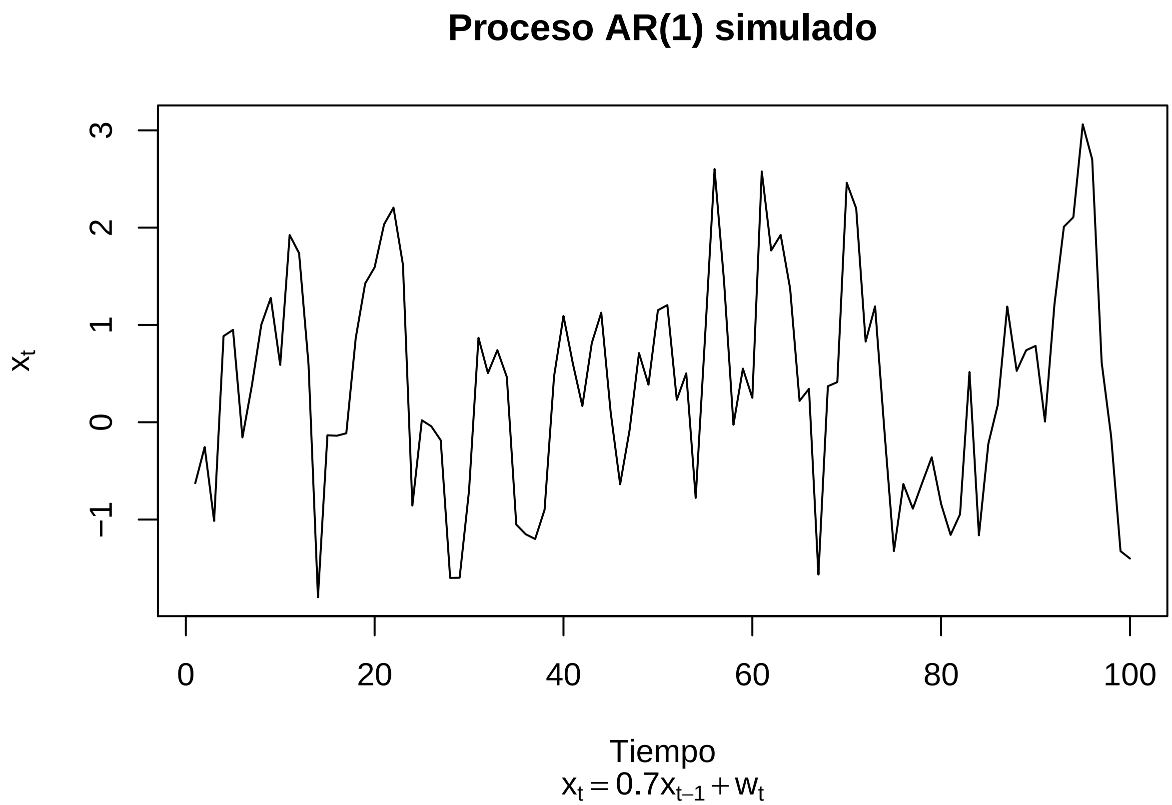

```r
#### Modelos AR(p), MA(q) y ARMA(p, q)

# Modelos AR(p)

# Correlograma de un proceso AR(1)

rho <- function(k, alpha) alpha^k
plot(0:10, rho(0:10, 0.7), type = "h", ylab = "", xlab = "")
title(main = "Correlograma para un proceso AR(1)",
      ylab = expression(rho[k] == alpha^k),
      xlab = "lag k",
      sub = expression(x[t]==0.7*x[t-1]+w[t]))
```



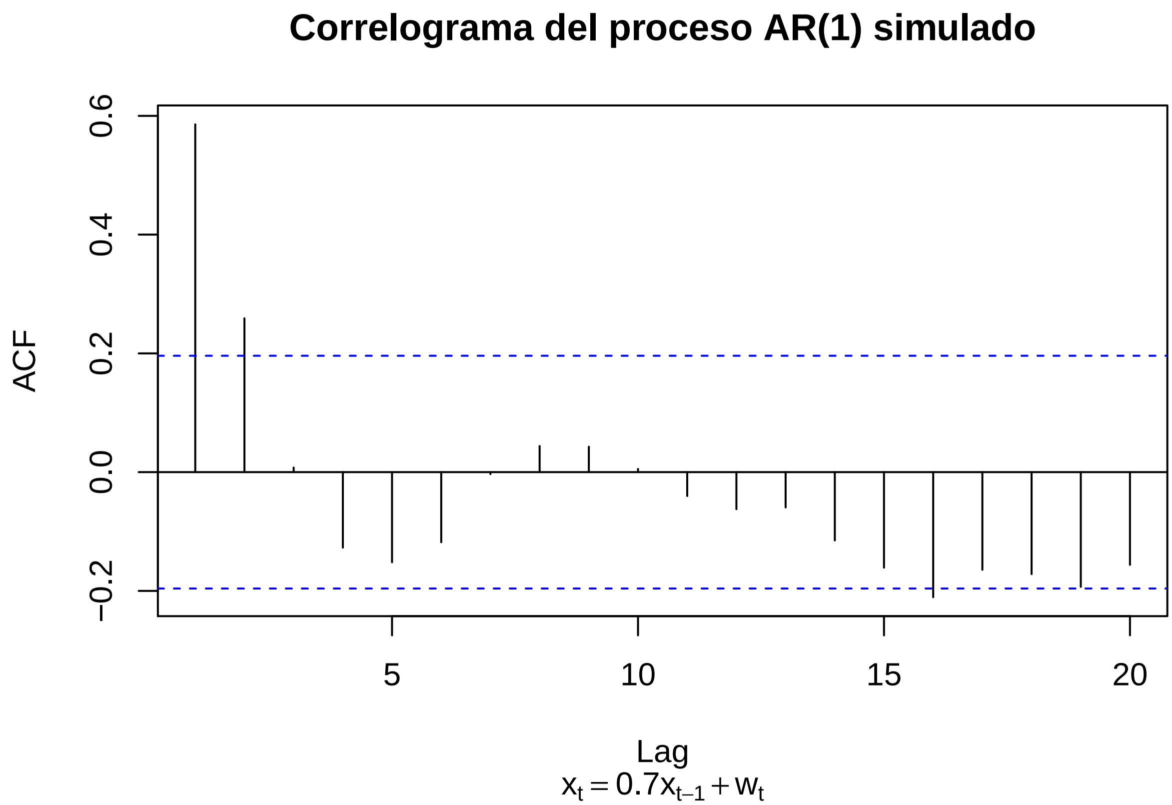

```
plot(0:10, rho(0:10, -0.7), type = "h", ylab = "", xlab = "")
title(main = "Correlograma para un proceso AR(1)",
      ylab = expression(rho[k] == alpha^k),
      xlab = "lag k",
      sub = expression(x[t]==-0.7*x[t-1]+w[t]))
abline(h = 0)
```



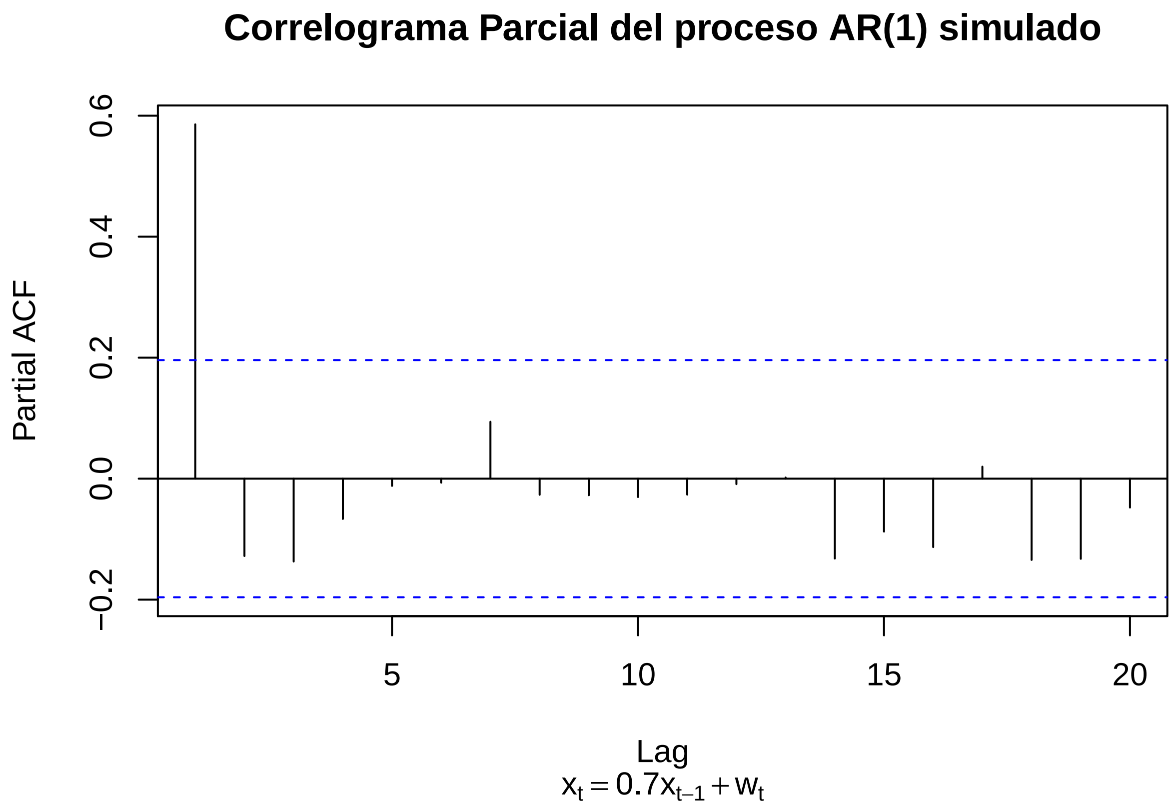

```
###

# Simulación en R

# Un proceso AR(1) puede ser simulado en R como sigue:

set.seed(1)
x <- w <- rnorm(100)
for(t in 2:100) x[t] <- 0.7 * x[t-1] + w[t]
plot(x, type = "l", xlab = "", ylab = "")
title(main = "Proceso AR(1) simulado",
      xlab = "Tiempo",
      ylab = expression(x[t]),
      sub = expression(x[t]==0.7*x[t-1]+w[t]))
```




```
#

acf(x, main = "")
title(main = "Correlograma del proceso AR(1) simulado",
      sub = expression(x[t]==0.7*x[t-1]+w[t]))
```




```r
#

pacf(x, main = "")
title(main = "Correlograma Parcial del proceso AR(1) simulado",
      sub = expression(x[t]==0.7*x[t-1]+w[t]))
```




```r
###

# Modelos Ajustados

# Ajuste de modelos a series simuladas

x.ar <- ar(x, method = "mle")
x.ar$order
```

```
[1] 1
```

```r
x.ar$ar
```

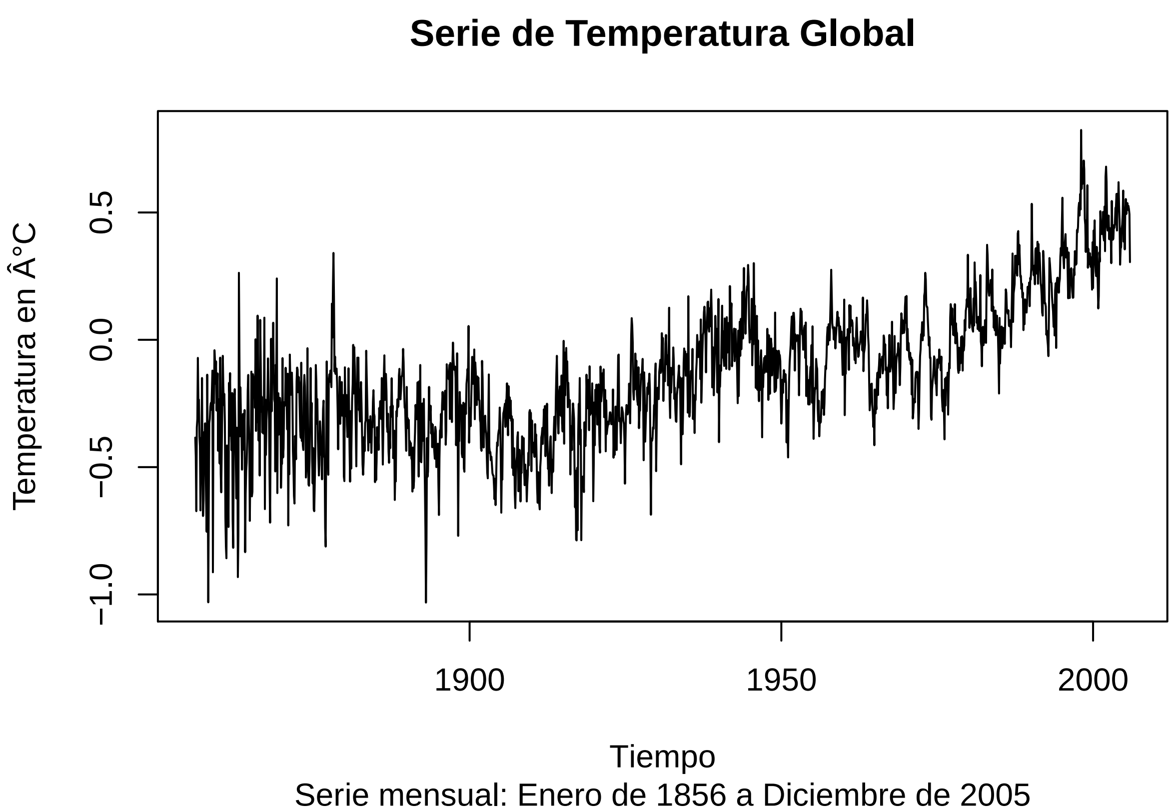
```
[1] 0.6009459
```

```r
x.ar$ar + c(-2, 2)*sqrt(x.ar$asy.var)
```

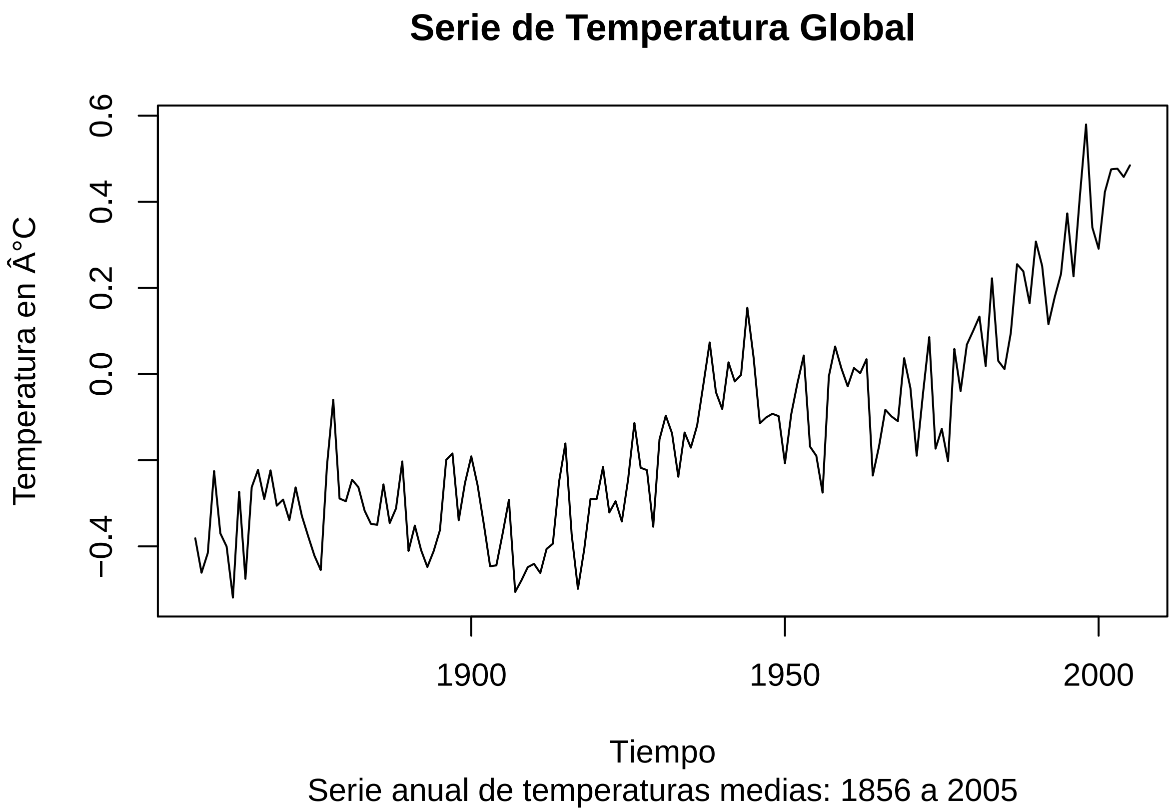
```
Warning in c(-2, 2) * sqrt(x.ar$asy.var): Recycling array of length 1 in vector-array arithmetic is dep
  Use c() or as.vector() instead.
```

```
[1] 0.4404031 0.7614886
```

```r
# Serie de temperatura global: Ajuste de un modelo AR

Global <- scan("global.txt")
Global.ts <- ts(Global, st = c(1856, 1), end = c(2005, 12), fr = 12)
Global.annual <- aggregate(Global.ts, FUN = mean)
plot(Global.ts, xlab = "Tiempo", ylab = "Temperatura en Â°C",
     main = "Serie de Temperatura Global",
     sub = "Serie mensual: Enero de 1856 a Diciembre de 2005")
```




```r
plot(Global.annual, xlab = "Tiempo", ylab = "Temperatura en Â°C",
     main = "Serie de Temperatura Global",
     sub = "Serie anual de temperaturas medias: 1856 a 2005")
```




```r
#

mean(Global.annual)
```

```
[1] -0.1382628
```

```r
Global.ar <- ar(Global.annual, method = "mle")
```

```
Warning in arima0(x, order = c(i, 0L, 0L), include.mean = demean): possible
convergence problem: optim gave code = 1
```

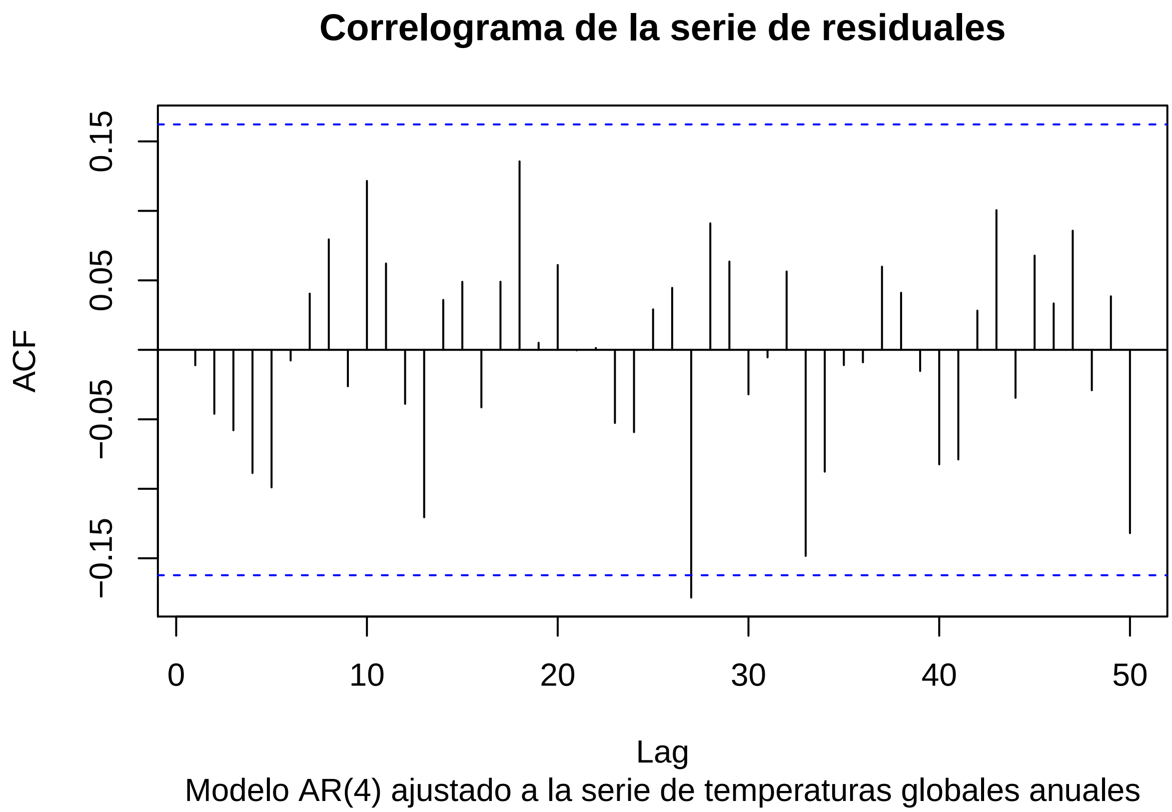
```r
Global.ar$order
```

```
[1] 4
```

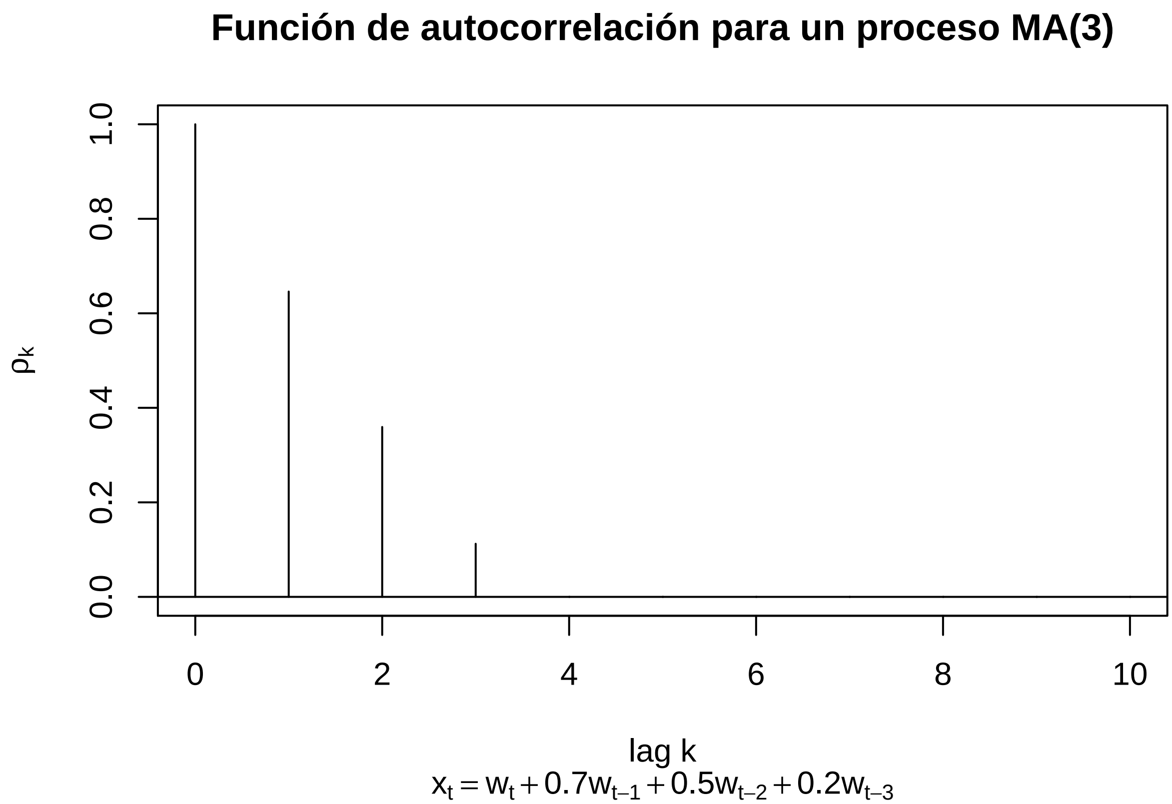
```r
Global.ar$ar
```

```
[1] 0.58762026 0.01260254 0.11116731 0.26763656
```

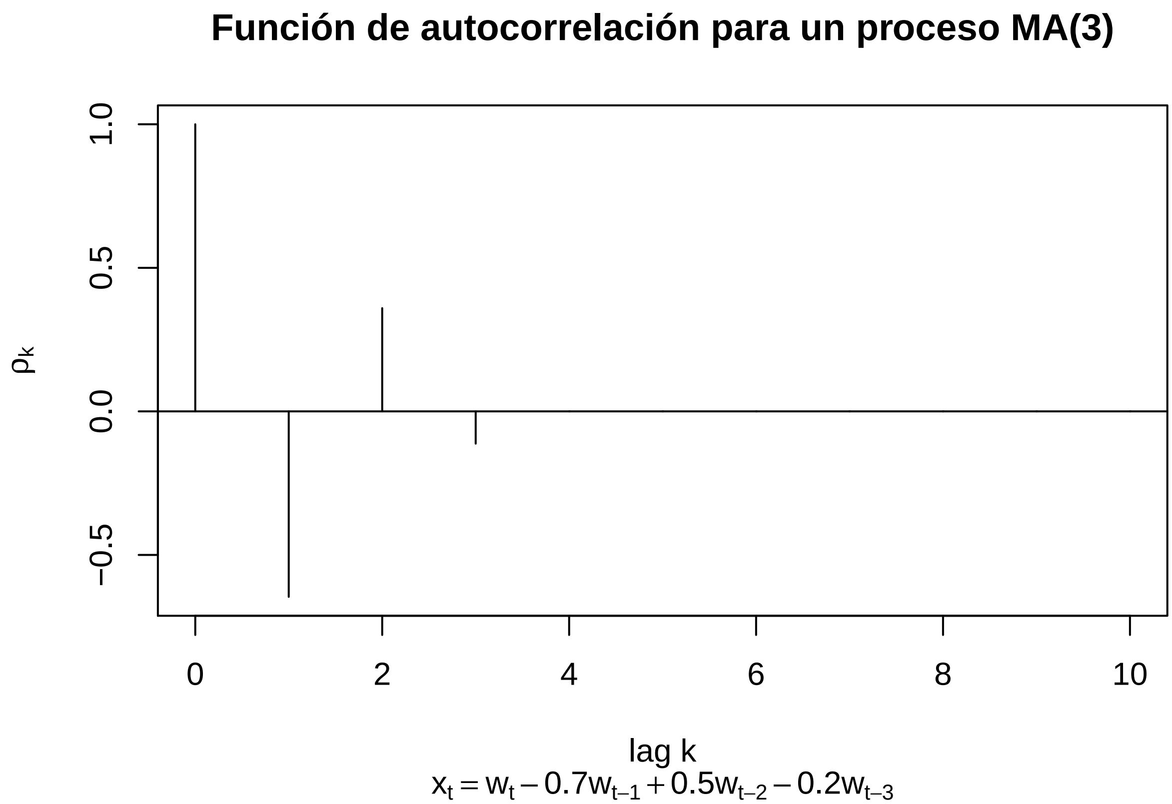
```r
acf(Global.ar$res[-(1:Global.ar$order)], lag = 50, main = "")
title(main = "Correlograma de la serie de residuales",
      sub = "Modelo AR(4) ajustado a la serie de temperaturas globales anuales")
```



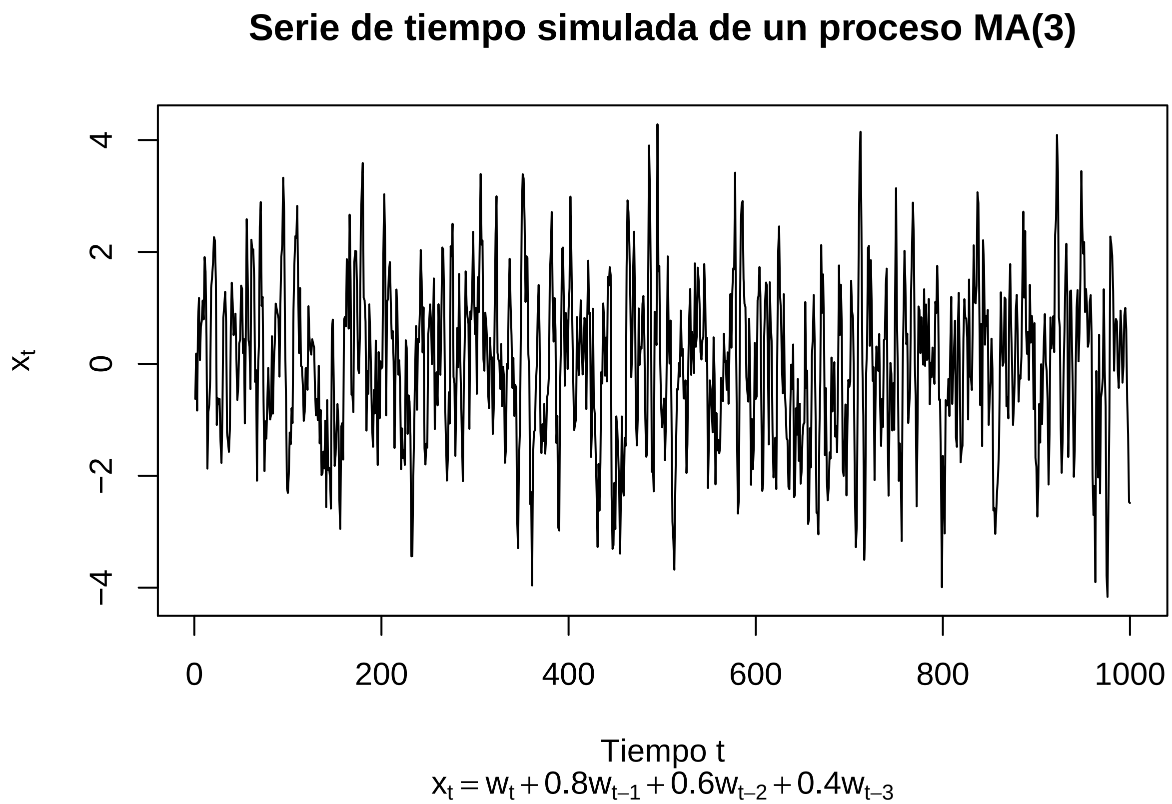

```
################################################################################################

# Modelos MA(q)

# Ejemplos en R: Correlograma y Simulación

# Función en R para calcular la Función de Autocorrelación

rho <- function(k, beta){
  q <- length(beta) - 1
  if(k > q) ACF <- 0 else {
    s1 <- 0; s2 <- 0
    for(i in 1:(q-k+1)) s1 <- s1 + beta[i]*beta[i + k]
    for(i in 1:(q+1)) s2 <- s2 + beta[i]^2
    ACF <- s1/s2}
  ACF}

# Correlograma para un proceso MA(3)

beta <- c(1, 0.7, 0.5, 0.2)
rho.k <- rep(1, 10)
for(k in 1:10) rho.k[k] <- rho(k, beta)
plot(0:10, c(1, rho.k), ylab = expression(rho[k]), xlab = "lag k", type = "h",
     sub = expression(x[t] == w[t] + 0.7*w[t-1] + 0.5*w[t-2] + 0.2*w[t-3]),
     main = "Función de autocorrelación para un proceso MA(3)")
abline(0, 0)
```



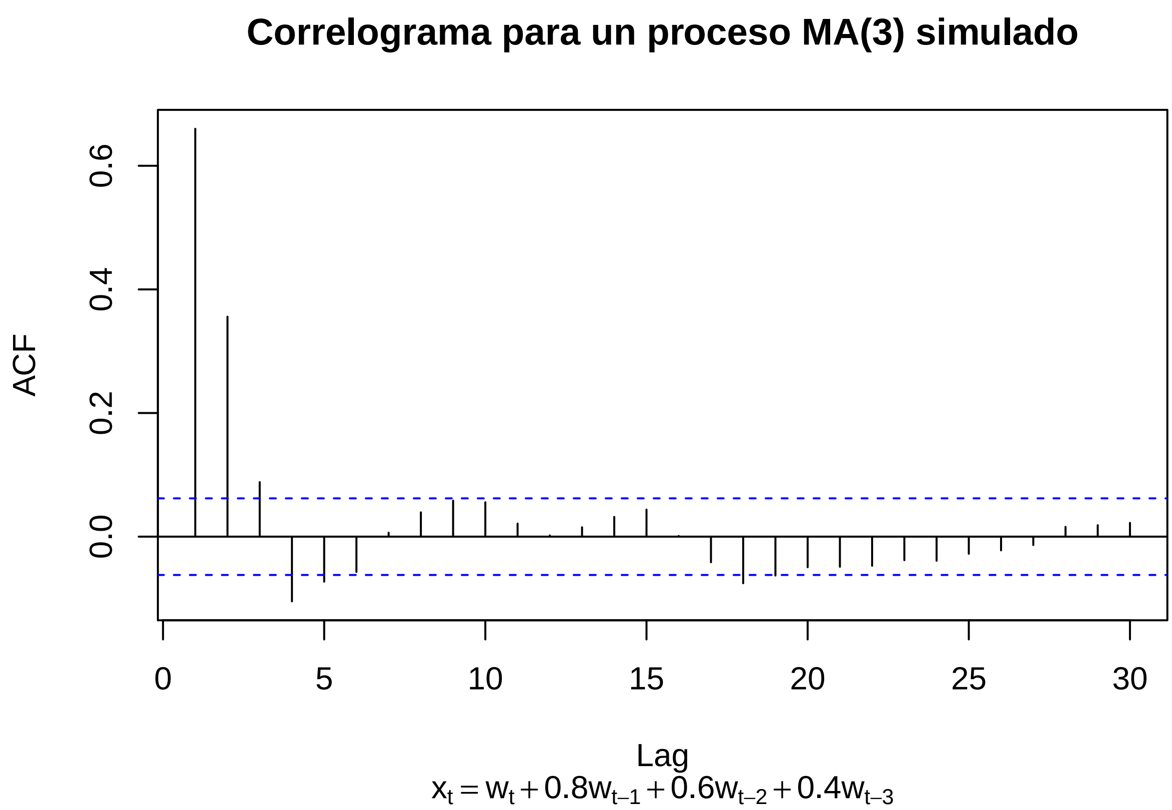

```r
# Correlograma para otro proceso MA(3)

beta <- c(1, -0.7, 0.5, -0.2)
rho.k <- rep(1, 10)
for(k in 1:10) rho.k[k] <- rho(k, beta)
plot(0:10, c(1, rho.k), ylab = expression(rho[k]), xlab = "lag k", type = "h",
     sub = expression(x[t] == w[t] - 0.7*w[t-1] + 0.5*w[t-2] - 0.2*w[t-3]),
     main = "Función de autocorrelación para un proceso MA(3)")
abline(0, 0)
```




```r
########################################################################################################

# Simulación de un proceso MA(3)

set.seed(1)
b <- c(0.8, 0.6, 0.4)
x <- w <- rnorm(1000)
for(t in 4:1000){
  for(j in 1:3) x[t] <- x[t] + b[j]*w[t-j]
}

plot(x, type = "l", ylab = expression(x[t]), xlab = "Tiempo t",
     sub = expression(x[t] == w[t] + 0.8*w[t-1] + 0.6*w[t-2] + 0.4*w[t-3]),
     main = "Serie de tiempo simulada de un proceso MA(3)")
```




```
###

acf(x, main = "")
title(main = "Correlograma para un proceso MA(3) simulado",
      sub = expression(x[t] == w[t] + 0.8*w[t-1] + 0.6*w[t-2] + 0.4*w[t-3]))
```

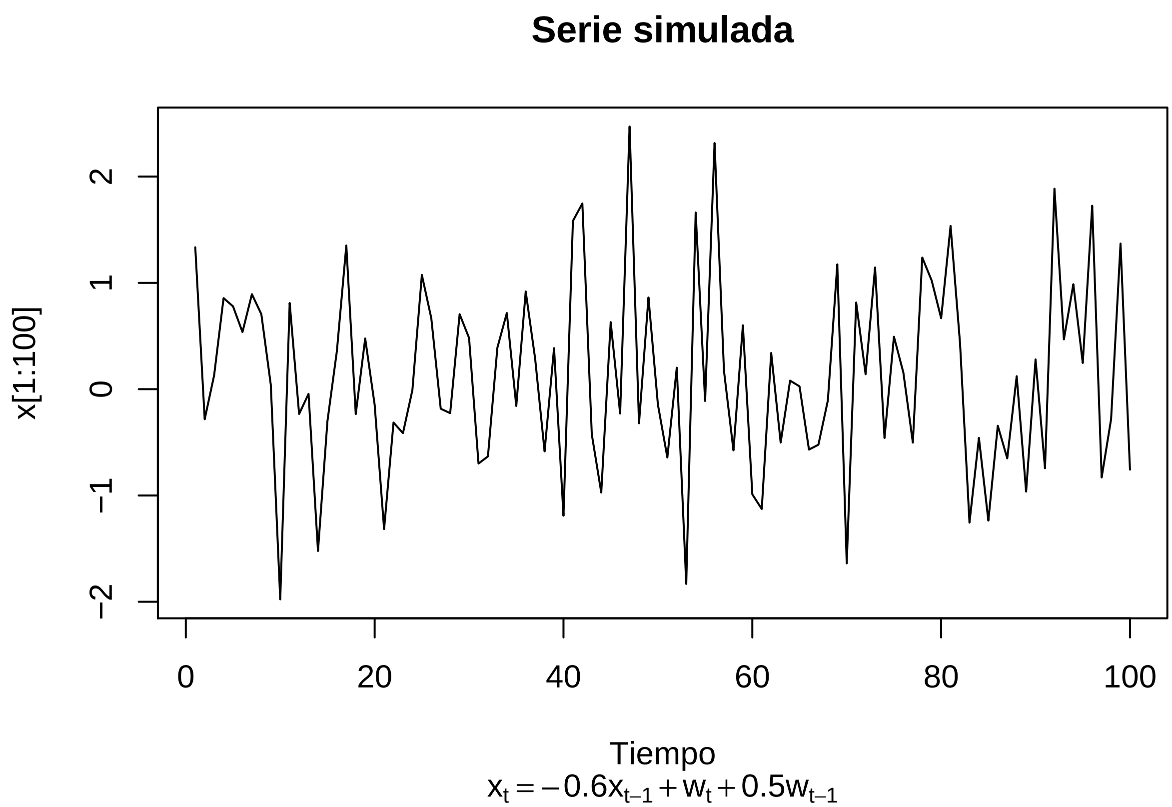



```r
##########################################################################################

# Ajuste de modelos MA

x.ma <- arima(x, order = c(0, 0, 3))
x.ma
```

```
Call:
arima(x = x, order = c(0, 0, 3))

Coefficients:
         ma1     ma2     ma3  intercept
      0.7898  0.5665  0.3959    -0.0322
s.e.  0.0307  0.0351  0.0320     0.0898

sigma^2 estimated as 1.068:  log likelihood = -1452.41,  aic = 2912.83
```

```r
##########################################################################################

# Modelos ARMA(p, q)


# Simulación y Ajuste
```

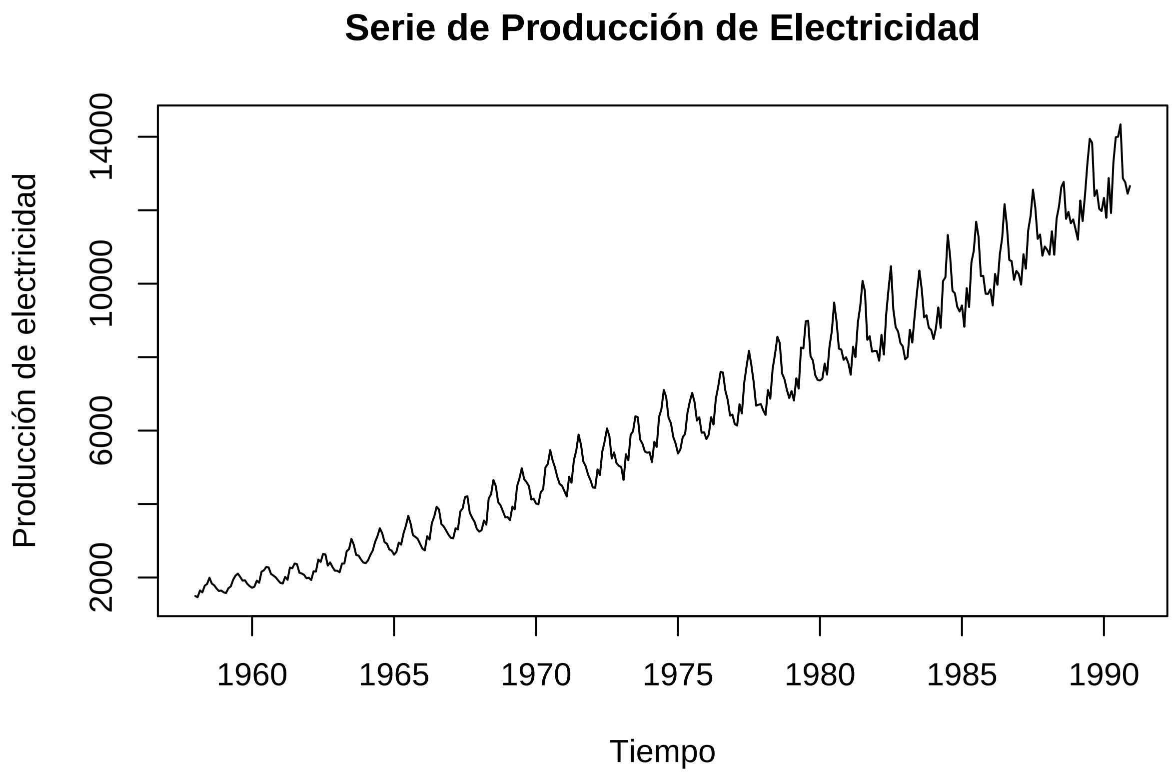
```r
set.seed(1)
x <- arima.sim(n = 10000, list(ar = -0.6, ma = 0.5))
plot(x[1:100], type = "l", xlab = "")
title(main = "Serie simulada", xlab = "Tiempo",
      sub = expression(x[t] == -0.6*x[t-1] + w[t] + 0.5*w[t-1]))
```



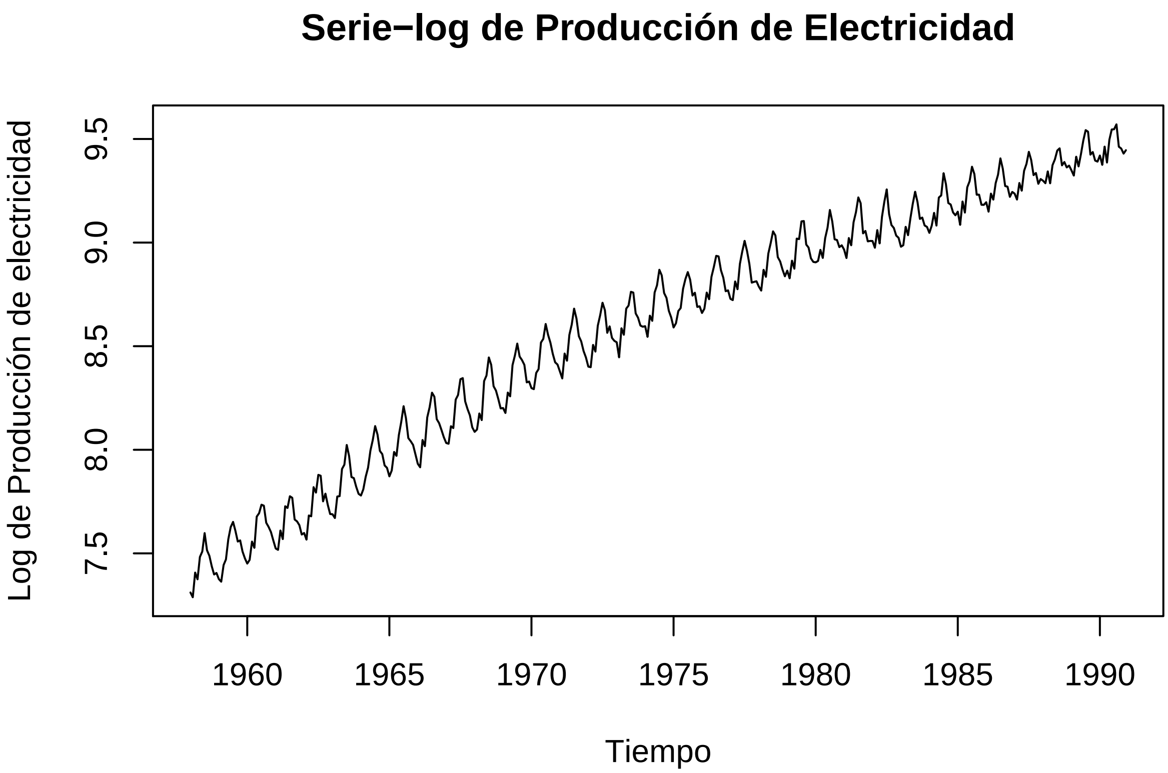

```r
#

coef(arima(x, order = c(1, 0, 1)))
```

```
         ar1          ma1    intercept 
-0.596966371  0.502703368 -0.006571345 
```

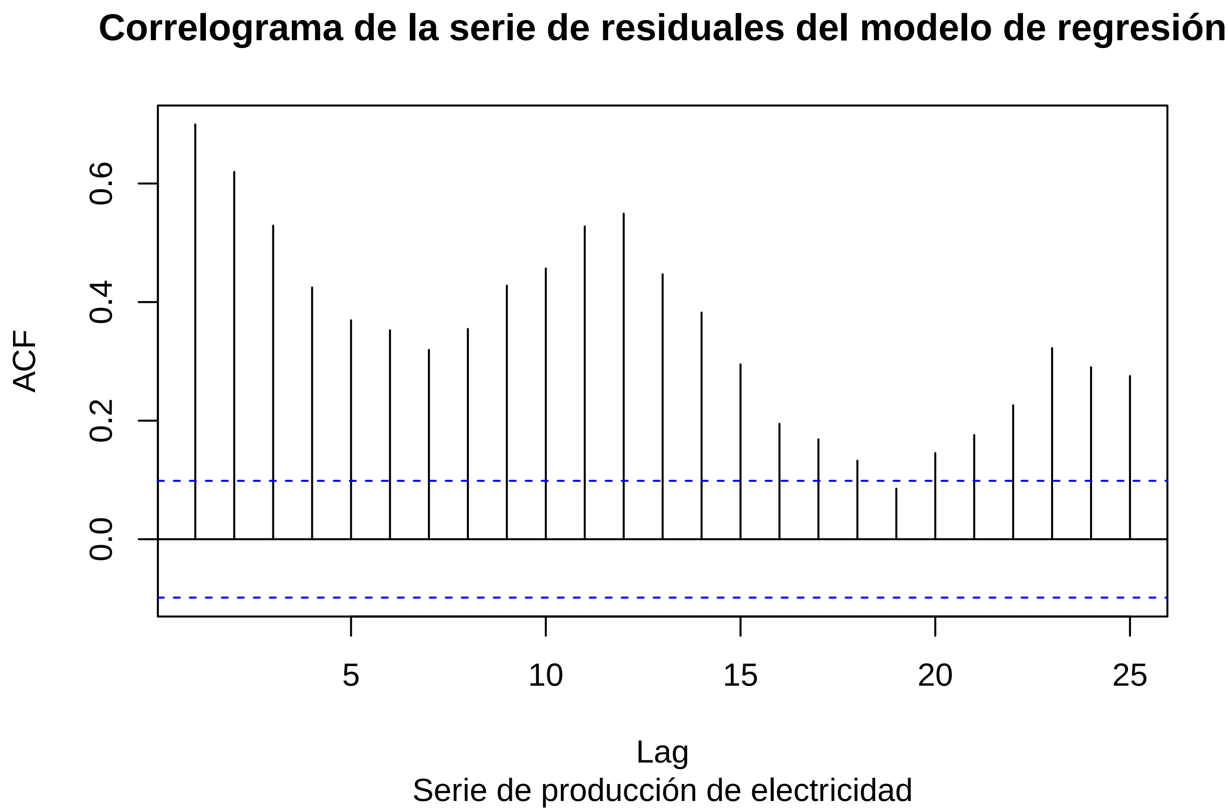
```r
#### Predicción

# Serie de producción de electricidad

CBE <- read.csv("cbe.csv", header = TRUE)
Elec.ts <- ts(CBE[, 3], start = 1958, freq = 12)
plot(Elec.ts, xlab = "", ylab = "")
title(main = "Serie de Producción de Electricidad",
      xlab = "Tiempo",
      ylab = "Producción de electricidad")
```



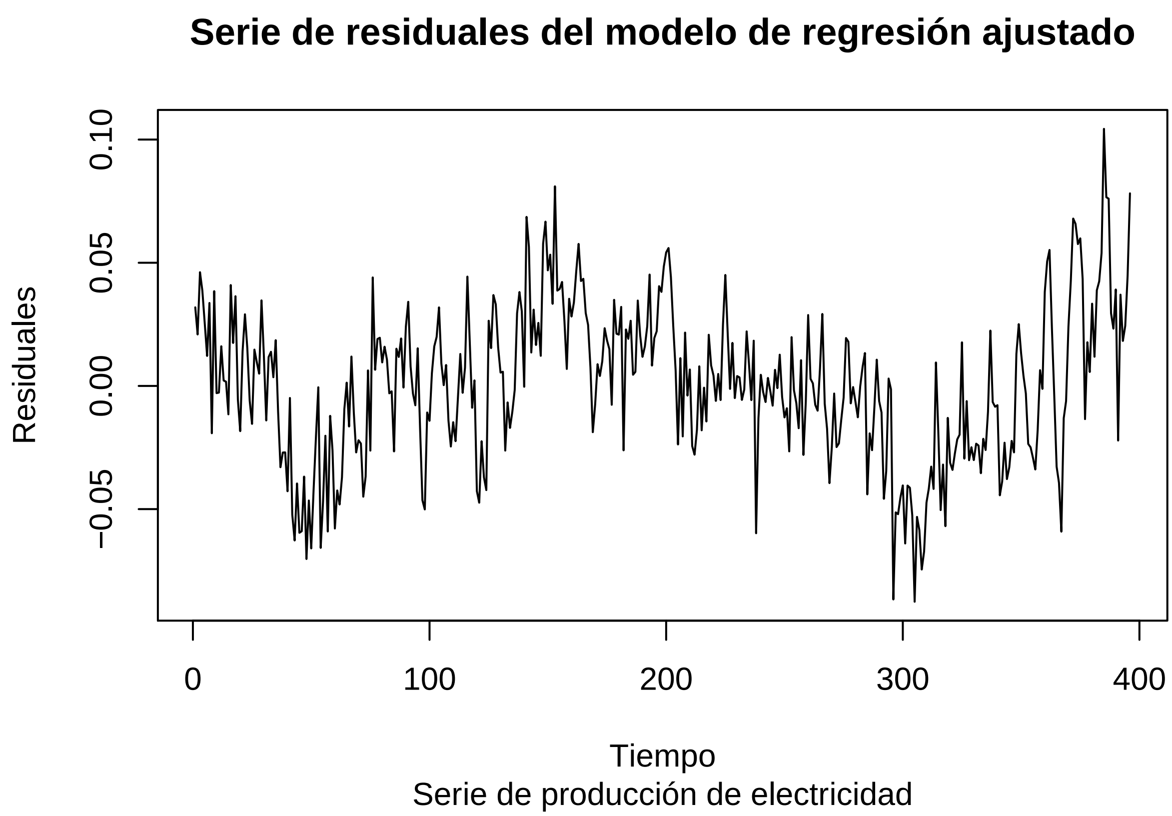

```
#

plot(log(Elec.ts), xlab = "", ylab = "")
title(main = "Serie-log de Producción de Electricidad",
      xlab = "Tiempo",
      ylab = "Log de Producción de electricidad")
```




```
#

Time <- 1:length(Elec.ts)
Imth <- cycle(Elec.ts)
Elec.lm <- lm(log(Elec.ts) ~ Time + I(Time^2) + factor(Imth))

#

acf(resid(Elec.lm), main = "")
title(main = "Correlograma de la serie de residuales del modelo de regresión",
      sub = "Serie de producción de electricidad")
```



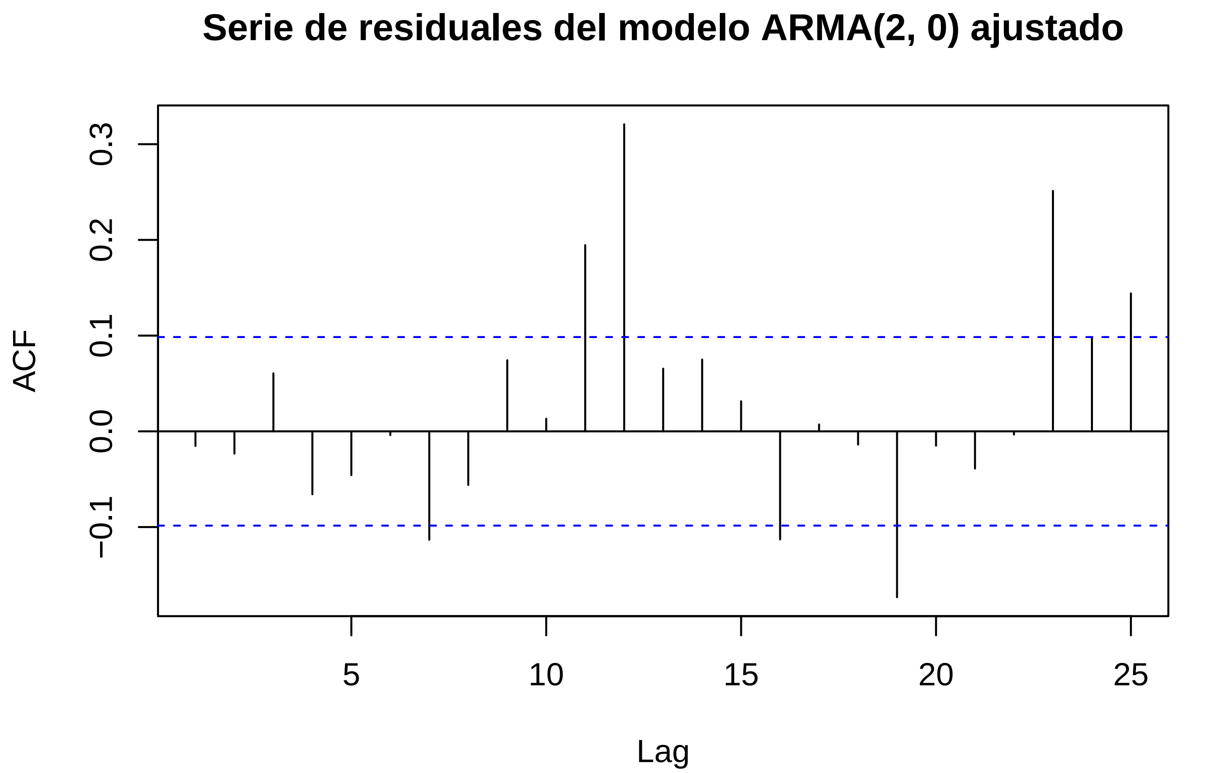

```r
#

plot(resid(Elec.lm), type = "l", main = "", xlab = "", ylab = "")
title(main = "Serie de residuales del modelo de regresión ajustado",
      sub = "Serie de producción de electricidad",
      xlab = "Tiempo",
      ylab = "Residuales")
```




```
###

# Código para encontrar el mejor modelo ARMA(p, q) considerando el AIC
# (Akaike Information Criterion)

best.order <- c(0, 0, 0)
best.aic <- Inf
for(i in 0:2)for(j in 0:2){
  model <- arima(resid(Elec.lm), order = c(i, 0, j))
  fit.aic <- AIC(model)
  if(fit.aic < best.aic){
    best.order <- c(i, 0, j)
    best.arma <- arima(resid(Elec.lm), order = best.order)
    best.aic <- fit.aic
  }
}

best.order
```

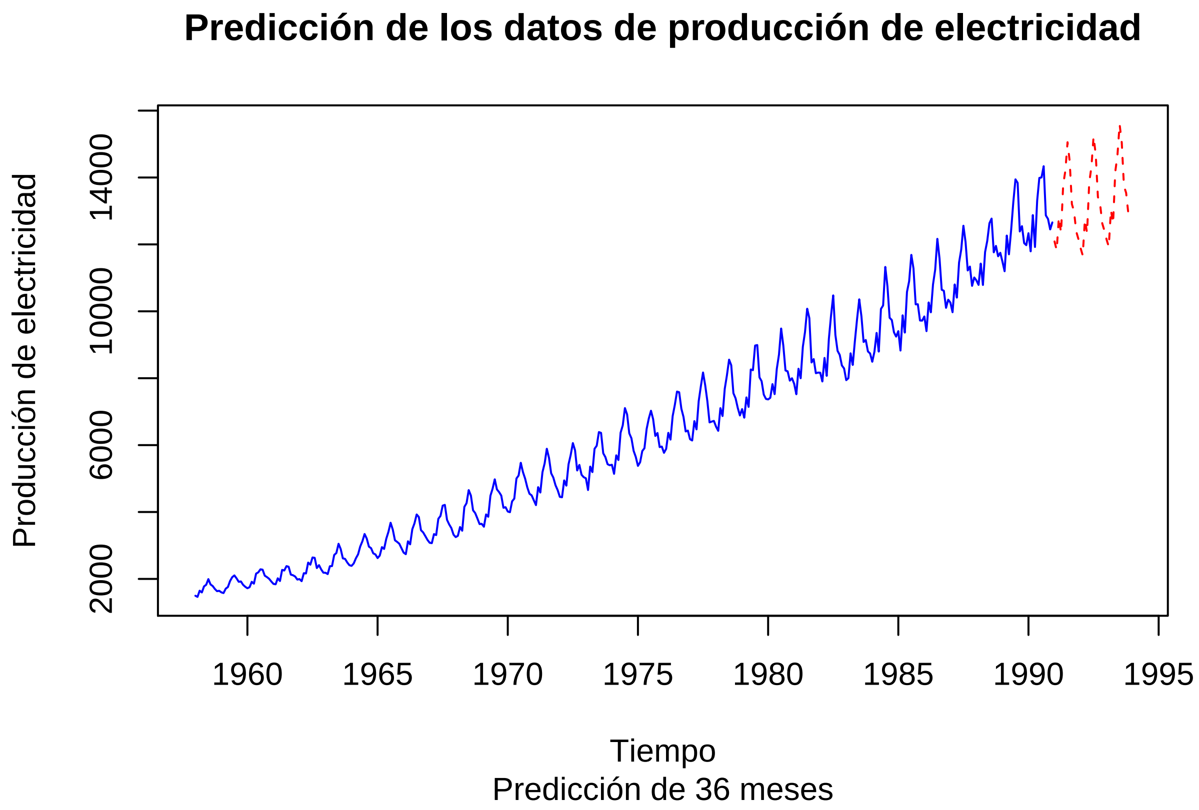
```
[1] 2 0 0
```

```
#

acf(resid(best.arma), main = "")
title(main = "Serie de residuales del modelo ARMA(2, 0) ajustado",
      sub = "Serie de residuales del modelo de regresión ajustado a los datos de electricidad")
```



Serie de residuales del modelo de regresión ajustado a los datos de electricidad

```
###

new.time <- seq(length(Elec.ts)+1, length = 36)
new.data <- data.frame(Time = new.time, Imth = rep(1:12, 3))
predict.lm <- predict(Elec.lm, new.data)
predict.arma <- predict(best.arma, n.ahead = 36)
elec.pred <- ts(exp(predict.lm + predict.arma$pred), start = 1991, freq = 12)

#

ts.plot(cbind(Elec.ts, elec.pred), lty = 1:2,
        col = c("blue", "red"), xlab = "Tiempo",
        ylab = "Producción de electricidad",
        main = "Predicción de los datos de producción de electricidad",
        sub = "Predicción de 36 meses")
```



RETO 1. PROCESO AR

OBJETIVO

Observar algunas características de una serie de tiempo que proviene de un proceso AR(1) y ajustar un modelo.

DESARROLLO

Reto 1. Simulación de un proceso AR(1)

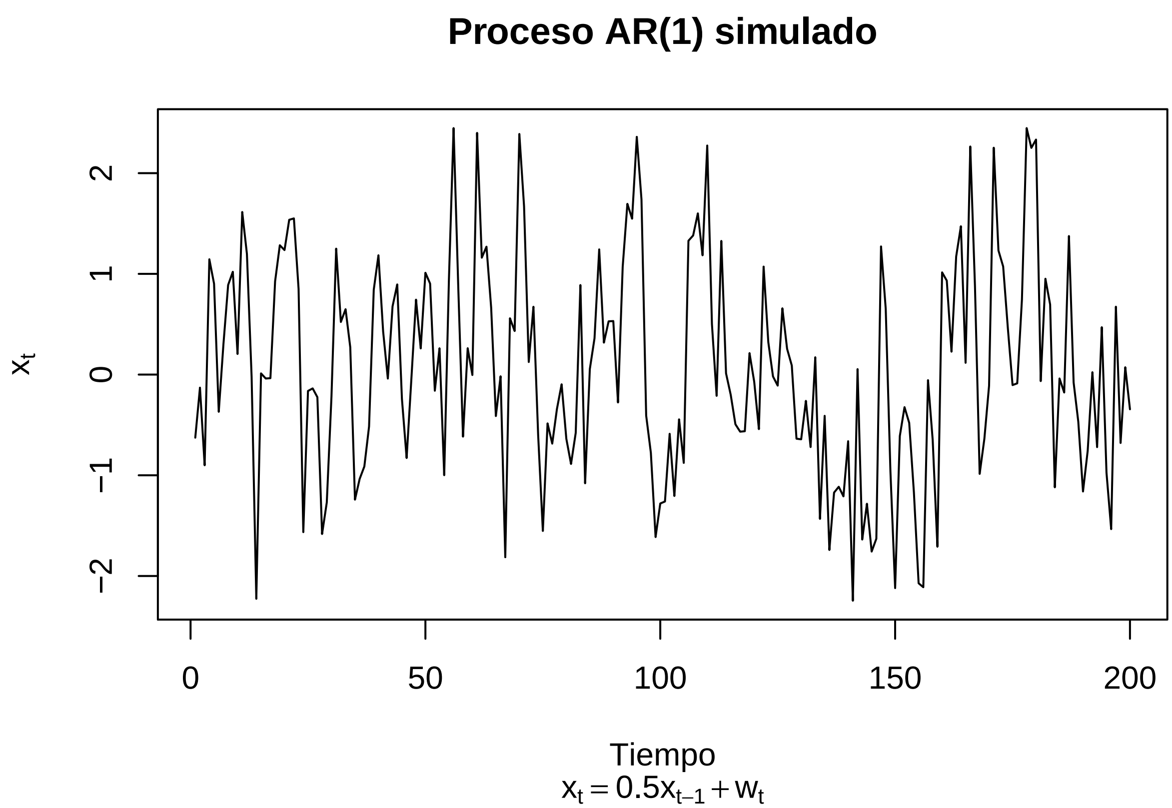
Simula un proceso AR(1) de la forma $x[t] = 0.5 * x[t-1] + w[t]$ para $t = 1, 2, ..., 200$ y muestra gráficamente la serie de tiempo obtenida

Obtén el correlograma y el correlograma parcial del proceso AR(1) simulado

Ajusta un modelo autorregresivo a la serie simulada utilizando la función ar, observa el orden del modelo y el parámetro estimado (los parámetros estimados)

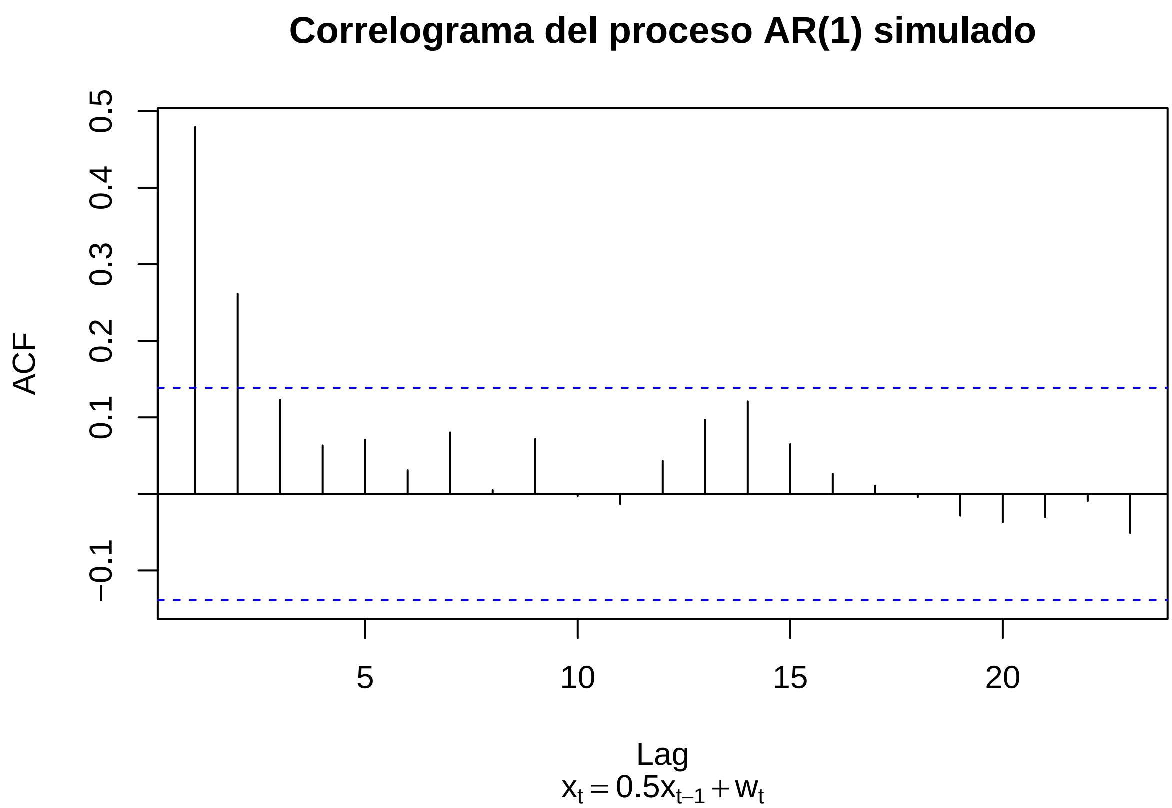
```
# Reto 1. Proceso AR(1)

# 1. Simule un proceso AR(1) de la forma x[t] = 0.5 * x[t-1] + w[t] para
# t = 1, 2, ..., 200 y muestre gráficamente la serie de tiempo obtenida
```

```r
# 2. Obtenga el correlograma y el correlograma parcial del proceso AR(1)
# simulado

# 3. Ajuste un modelo autorregresivo a la serie simulada utilizando la
# función ar, observe el orden del módelo y el parámetro
# estimado (los paramétros estimados)

# **Solución**

# Simulación en R

# Un proceso AR(1) puede ser simulado en R como sigue:

set.seed(1)
x <- w <- rnorm(200)
for(t in 2:200) x[t] <- 0.5 * x[t-1] + w[t]
plot(x, type = "l", xlab = "", ylab = "")
title(main = "Proceso AR(1) simulado",
      xlab = "Tiempo",
      ylab = expression(x[t]),
      sub = expression(x[t]==0.5*x[t-1]+w[t]))
```

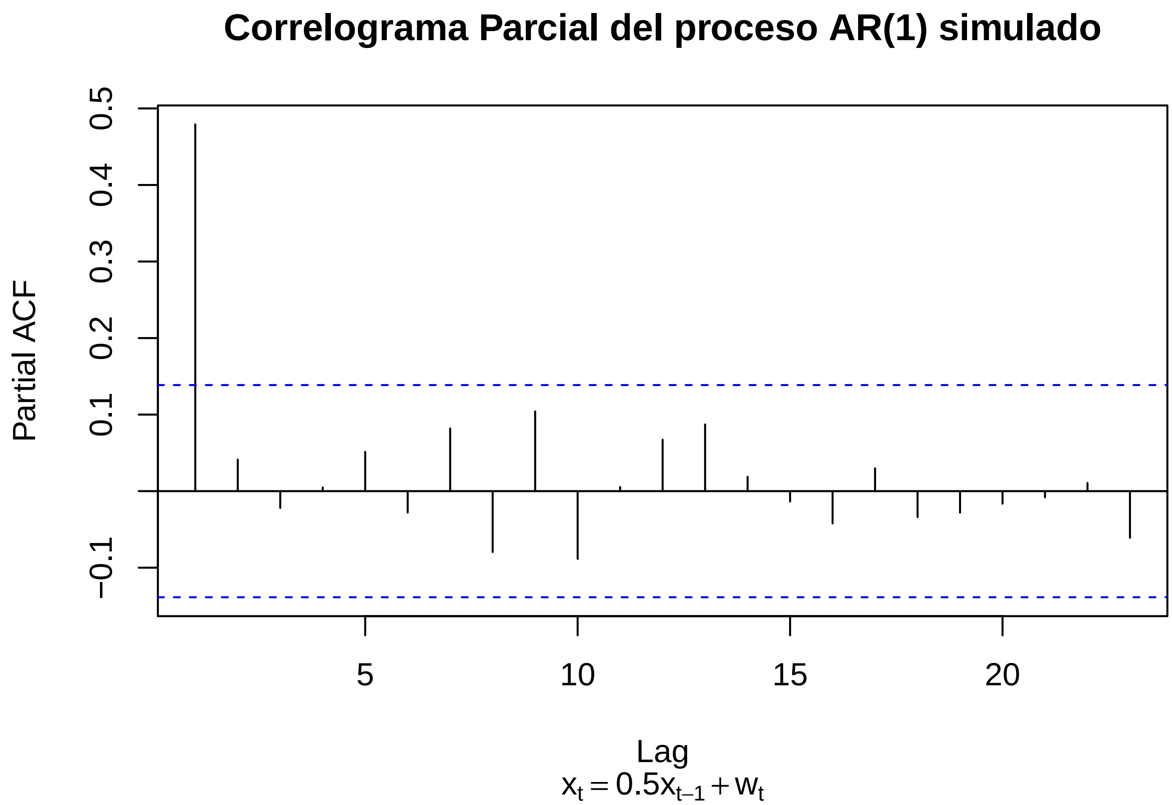



```r
#

acf(x, main = "")
```

```r
title(main = "Correlograma del proceso AR(1) simulado",
      sub = expression(x[t]==0.5*x[t-1]+w[t]))
```




```r
#

pacf(x, main = "")
title(main = "Correlograma Parcial del proceso AR(1) simulado",
      sub = expression(x[t]==0.5*x[t-1]+w[t]))
```




```r
###

# Modelos Ajustados

# Ajuste de modelos a series simuladas

x.ar <- ar(x, method = "mle")
x.ar$order
```

```
[1] 1
```

```r
x.ar$ar
```

```
[1] 0.4782263
```

EJEMPLO 3. MODELOS NO ESTACIONARIOS Y PREDICCIÓN

DESARROLLO

Tomamos datos de https://github.com/AtefOuni/ts/tree/master/Data

Serie de producción de electricidad de Australia

CBE <- read.csv("cbe.csv", header = TRUE) Elec.ts <- ts(CBE[, 3], start = 1958, freq = 12) plot(Elec.ts, xlab = "", ylab ="") title(main ="Serie de Producción de Electricidad Australiana", ylab ="Producción de electricidad (GWh)", xlab ="Tiempo") plot(diff(Elec.ts), xlab ="", ylab ="") title(main ="Serie Diferenciada de Producción de Electricidad Australiana", xlab ="Tiempo", ylab ="Dif Serie", sub ="Gráfica de la serie diferenciada de primer órden") plot(diff(log(Elec.ts)), xlab ="", ylab ="") title(main ="Serie de log dif de Producción de Electricidad Australiana", xlab ="Tiempo", ylab ="Dif log-Serie", sub ="Gráfica de la serie log-transformada diferenciada de primer órden") Simulación y ajuste

A continuación, simulamos datos de un modelo ARIMA(1, 1, 1) y luego ajustamos un modelo a la serie simulada para recuperar los parámetros estimados.

set.seed(1) x <- w <- rnorm(1000) for(i in 3:1000) x[i] <- 0.5*x[i-1] + x[i-1] - 0.5*x[i-2] + w[i] + 0.3*w[i-1] plot(x, type = "l", main = "Serie simulada de un modelo ARIMA(1, 1, 1)", xlab = "Tiempo", ylab = expression(x[t]), sub = expression(x[t] == 0.5*x[t-1] + x[t-1] - 0.5*x[t-2] + w[t] + 0.3*w[t-1])) arima(x, order = c(1, 1, 1)) Simulación con la función arima.sim

x <- arima.sim(model = list(order = c(1, 1, 1), ar = 0.5, ma = 0.3), n = 1000) arima(x, order = c(1, 1, 1)) Serie de producción de cerveza

CBE <- read.csv("cbe.csv", header = TRUE) Beer.ts <- ts(CBE[, 2], start = 1958, freq = 12) plot(Beer.ts, xlab = "", ylab ="") title(main ="Serie de Producción de Cerveza en Australia", ylab ="Producción de Cerveza (Megalitros)", xlab ="Mes") Beer.ima <- arima(Beer.ts, order = c(0, 1, 1)) Beer.ima acf(resid(Beer.ima), main ="") title(main ="Autocorrelaciones para los Residuales del Ajuste", sub = expression(x[t]==x[t-1]+w[t]-0.33*w[t-1])) Beer.1991 <- predict(Beer.ima, n.ahead = 12) sum(Beer.1991$pred) Modelos Arima estacionales Procedimiento de ajuste Serie de producción de electricidad de Australia

CBE <- read.csv("cbe.csv", header = TRUE) Elec.ts <- ts(CBE[, 3], start = 1958, freq = 12) plot(Elec.ts, xlab = "", ylab ="") title(main ="Serie de Producción de Electricidad Australiana", ylab ="Producción de electricidad (GWh)", xlab ="Tiempo") plot(log(Elec.ts), xlab ="", ylab ="") title(main ="Log de Serie de Producción de Electricidad Australiana", ylab ="Log de Producción de electricidad (GWh)", xlab ="Tiempo") Elec.AR <- arima(log(Elec.ts), order = c(1, 1, 0), seas = list(order = c(1, 0, 0), 12))

Elec.MA <- arima(log(Elec.ts), order = c(0, 1, 1), seas = list(order = c(0, 0, 1), 12))

AIC(Elec.AR) AIC(Elec.MA) Función para buscar un buen modelo

get.best.arima <- function(x.ts, maxord = c(1, 1, 1, 1, 1, 1)){ best.aic <- 1e8 n <- length(x.ts) for(p in 0:maxord[1])for(d in 0:maxord[2])for(q in 0:maxord[3]) for(P in 0:maxord[4])for(D in 0:maxord[5])for(Q in 0:maxord[6]) { fit <- arima(x.ts, order = c(p, d, q), seas = list(order = c(P, D, Q), frequency(x.ts)), method = "CSS") fit.aic <- -2*fit$loglik + (log(n) + 1) * length(fit$coef) if(fit.aic < best.aic){ best.aic <- fit.aic best.fit <- fit best.model <- c(p, d, q, P, D, Q) } } list(best.aic, best.fit, best.model) } Nuevo ajuste a los datos de la serie transformada de producción de electricidad

best.arima.elec <- get.best.arima(log(Elec.ts), maxord = c(2, 2, 2, 2, 2, 2))

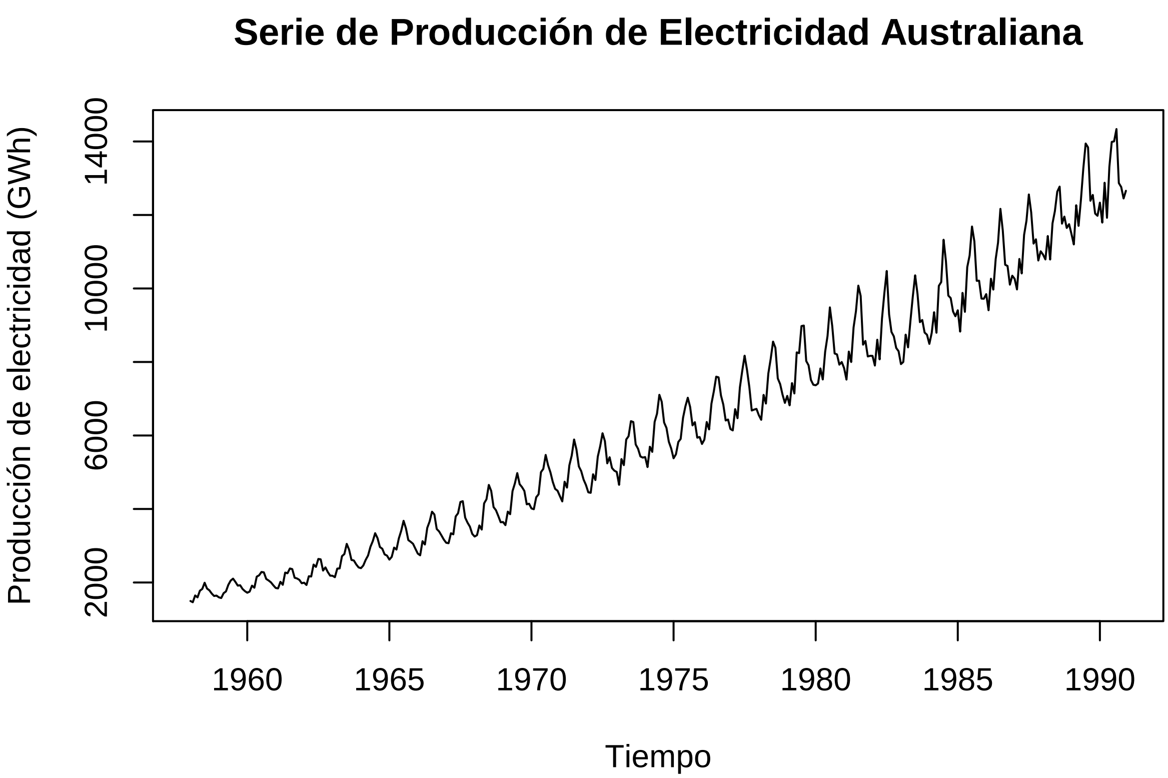
best.fit.elec <- best.arima.elec[[2]] # Modelo best.arima.elec[[3]] # Tipo de modelo (órdenes) best.fit.elec best.arima.elec[[1]] # AIC ACF para residuales del ajuste

acf(resid(best.fit.elec), main = "") title(main ="Correlograma de los residuales del ajuste") Predicción

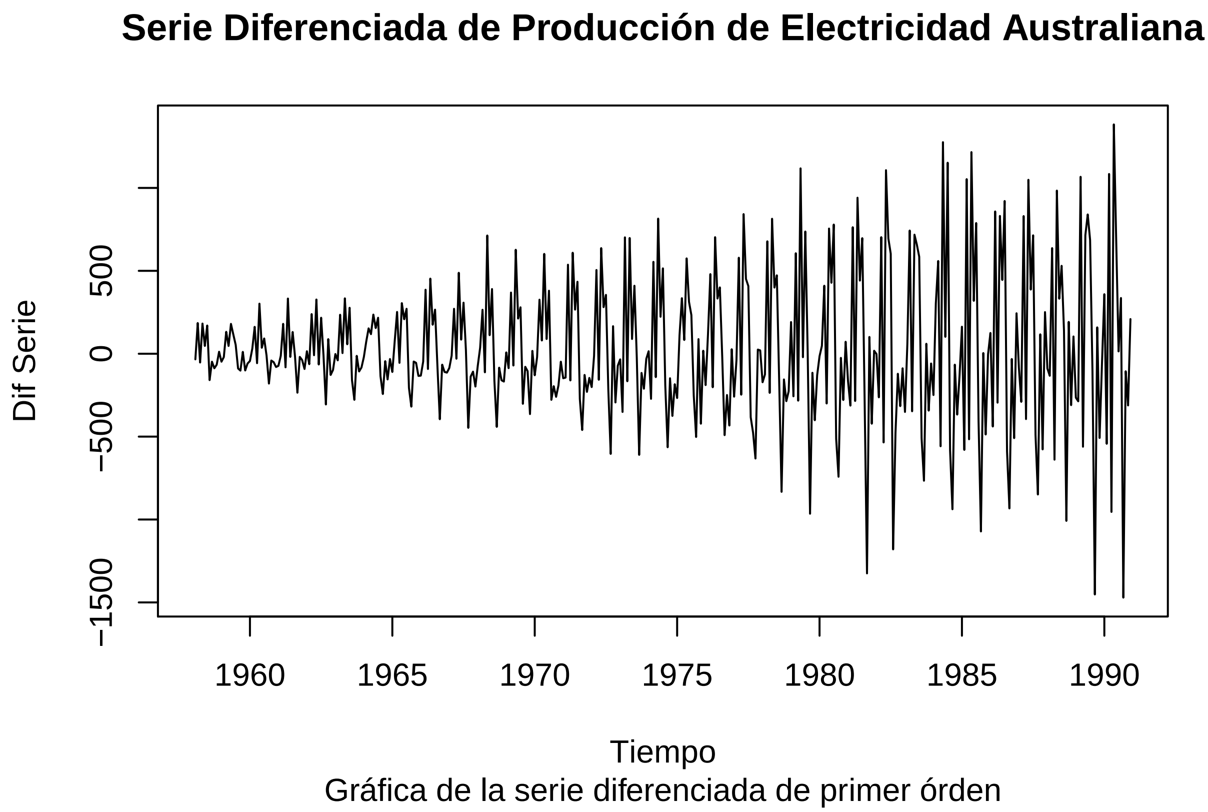
pr <- predict(best.fit.elec, 12)$pred ts.plot(cbind(window(Elec.ts, start = 1981), exp(pr)), col = c("blue", "red"), xlab = "") title(main ="Predicción para la serie de producción de electricidad", xlab ="Mes", ylab ="Producción de electricidad (GWh)") Inspirado en la siguiente bibliografía:

```
# Ejemplo 3. Modelos no estacionarios y predicción

# https://github.com/AtefOuni/ts/tree/master/Data

# Serie de Producción de Electricidad de Australia
```

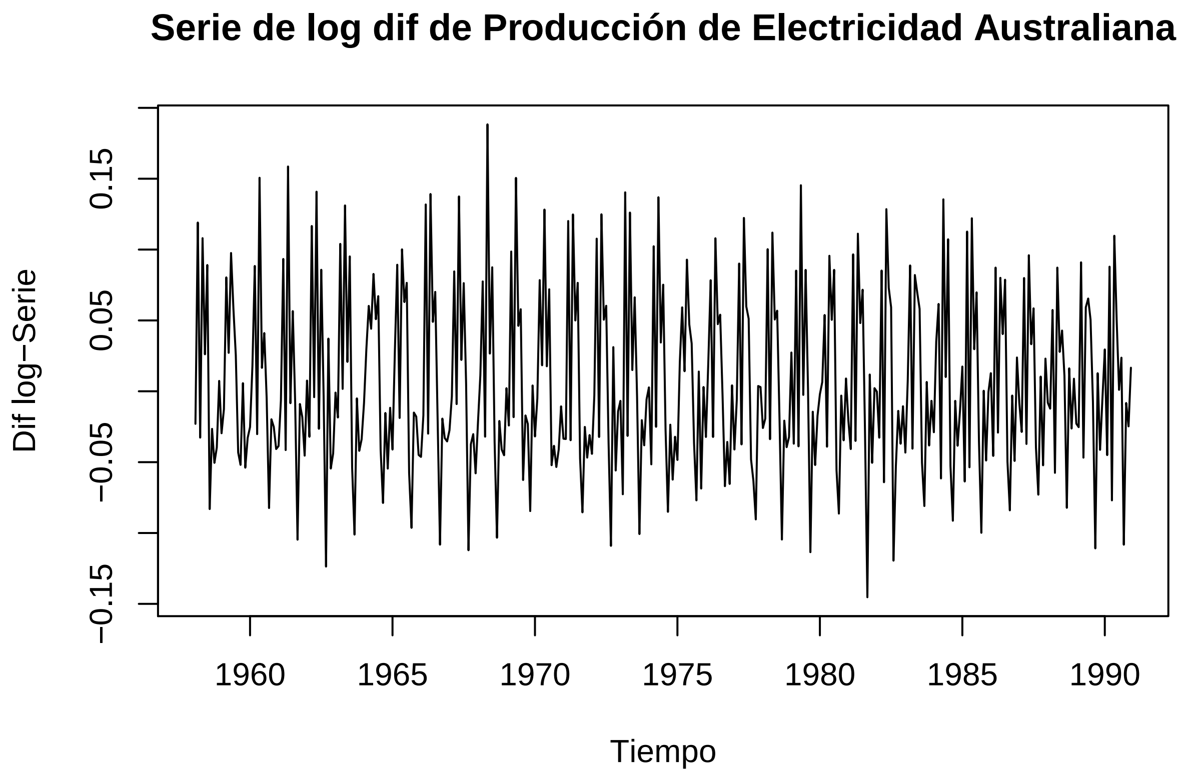
```r
CBE <- read.csv("cbe.csv", header = TRUE)
Elec.ts <- ts(CBE[, 3], start = 1958, freq = 12)
plot(Elec.ts, xlab = "", ylab = "")
title(main = "Serie de Producción de Electricidad Australiana",
      ylab = "Producción de electricidad (GWh)",
      xlab = "Tiempo")
```




```r
###

plot(diff(Elec.ts), xlab = "", ylab = "")
title(main = "Serie Diferenciada de Producción de Electricidad Australiana",
      xlab = "Tiempo", ylab = "Dif Serie",
      sub = "Gráfica de la serie diferenciada de primer órden")
```

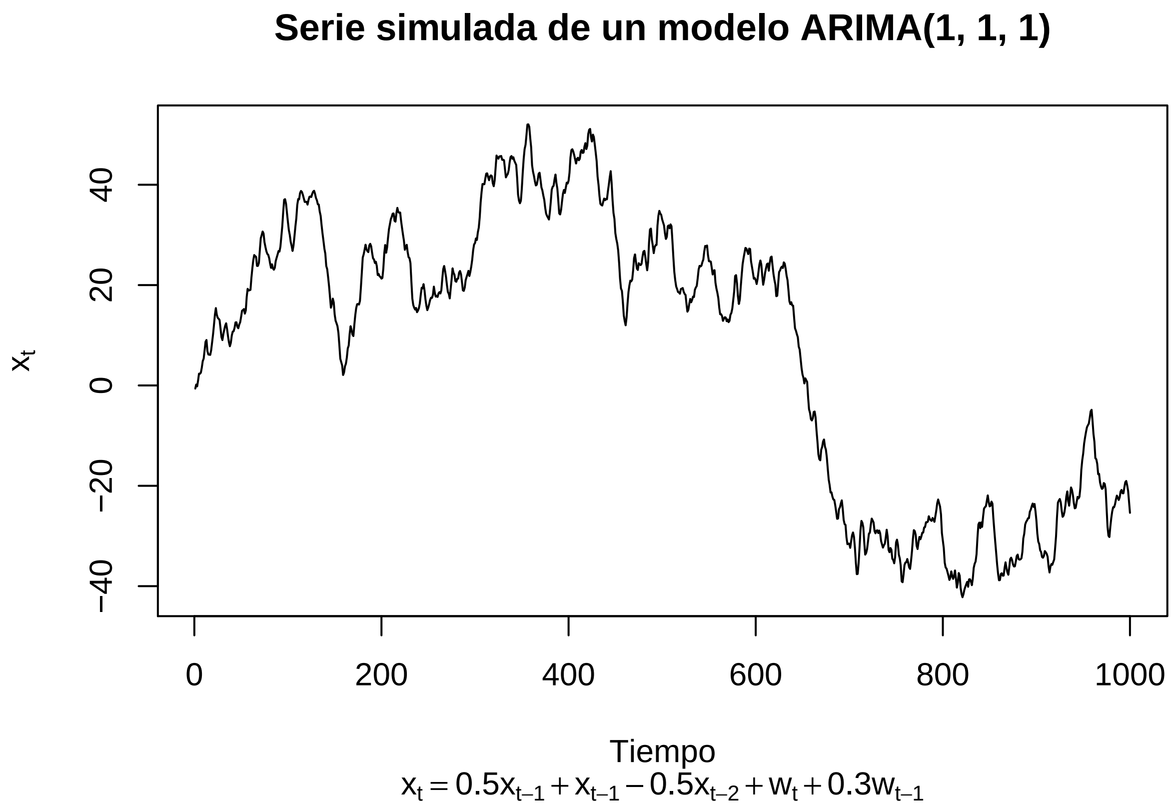



```
###

plot(diff(log(Elec.ts)), xlab = "", ylab = "")
title(main = "Serie de log dif de Producción de Electricidad Australiana",
      xlab = "Tiempo", ylab = "Dif log-Serie",
      sub = "Gráfica de la serie log-transformada diferenciada de primer órden")
```



Gráfica de la serie log−transformada diferenciada de primer órden

```
#######################################################################################################

                      # Simulación y ajuste

# A continuación, simulamos datos de un modelo ARIMA(1, 1, 1) y luego ajustamos un modelo a la serie si
# para recuperar los parámetros estimados.

set.seed(1)
x <- w <- rnorm(1000)
for(i in 3:1000) x[i] <- 0.5*x[i-1] + x[i-1] - 0.5*x[i-2] + w[i] + 0.3*w[i-1]

###

plot(x, type = "l",
     main = "Serie simulada de un modelo ARIMA(1, 1, 1)",
     xlab = "Tiempo",
     ylab = expression(x[t]),
     sub = expression(x[t] == 0.5*x[t-1] + x[t-1] - 0.5*x[t-2] + w[t] + 0.3*w[t-1]))
```



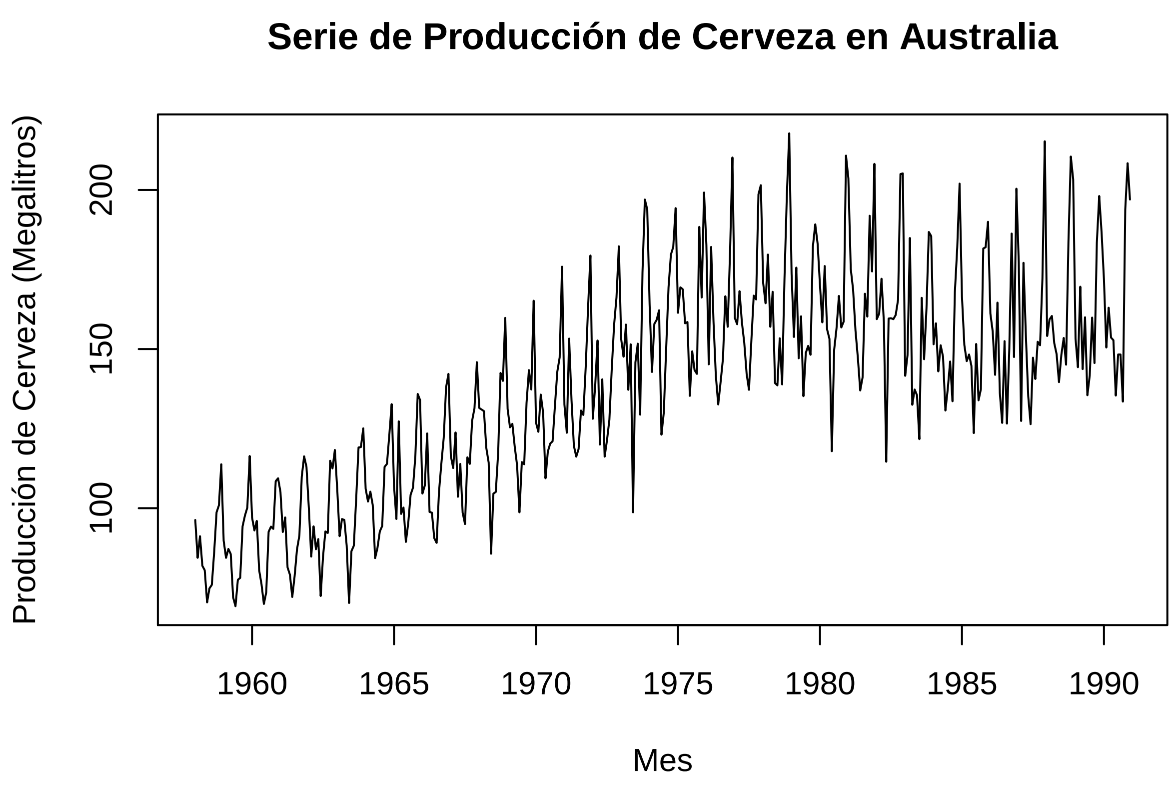

```r
###

arima(x, order = c(1, 1, 1))
```

```
Call:
arima(x = x, order = c(1, 1, 1))

Coefficients:
         ar1     ma1
      0.4235  0.3308
s.e.  0.0433  0.0450

sigma^2 estimated as 1.067:  log likelihood = -1450.13,  aic = 2904.26
```

```r
###

      # Simulación con la función arima.sim

x <- arima.sim(model = list(order = c(1, 1, 1), ar = 0.5, ma = 0.3), n = 1000)

###

arima(x, order = c(1, 1, 1))
```

```
Call:
arima(x = x, order = c(1, 1, 1))

Coefficients:
         ar1     ma1
      0.5567  0.2502
s.e.  0.0372  0.0437

sigma^2 estimated as 1.079:  log likelihood = -1457.45,  aic = 2918.91
```

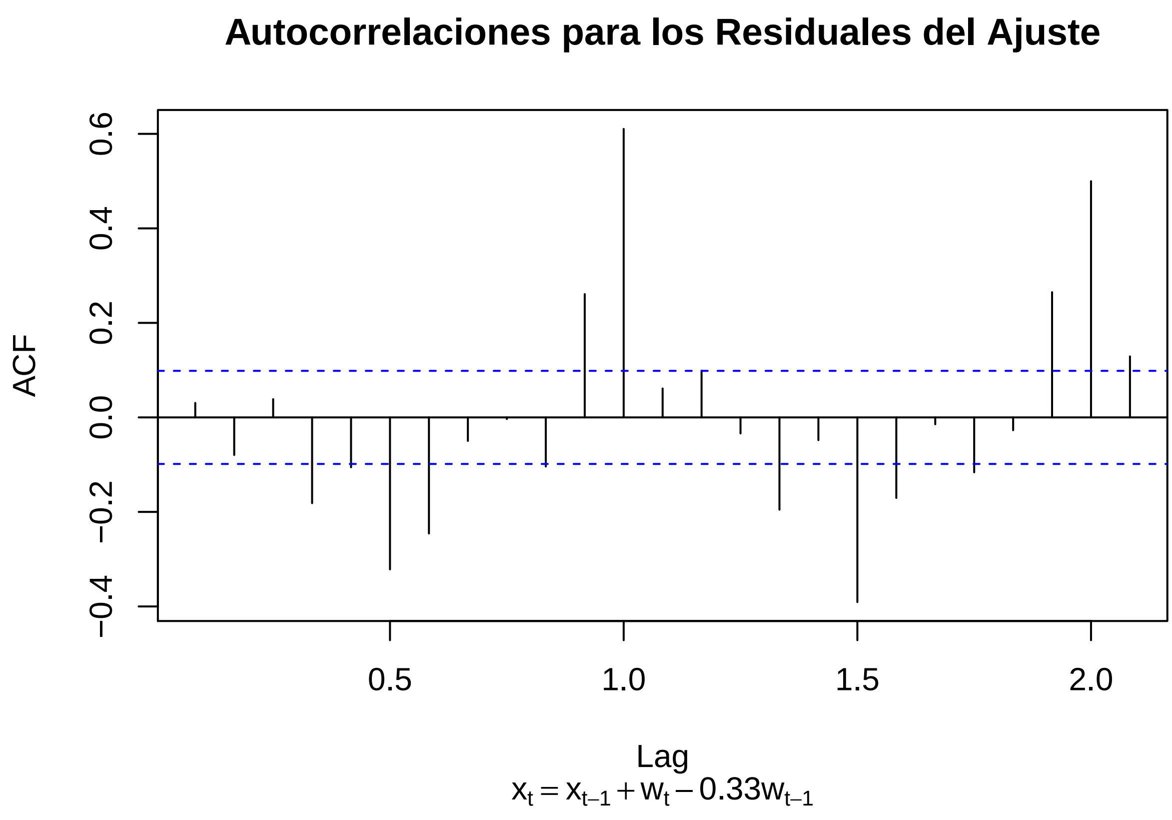
```
####################################################################################################

    # Serie de producción de cerveza

CBE <- read.csv("cbe.csv", header = TRUE)
Beer.ts <- ts(CBE[, 2], start = 1958, freq = 12)
plot(Beer.ts, xlab = "", ylab = "")
title(main = "Serie de Producción de Cerveza en Australia",
      ylab = "Producción de Cerveza (Megalitros)",
      xlab = "Mes")
```




```
###

Beer.ima <- arima(Beer.ts, order = c(0, 1, 1))
Beer.ima
```

```
Call:
arima(x = Beer.ts, order = c(0, 1, 1))

Coefficients:
          ma1
      -0.3334
s.e.   0.0558

sigma^2 estimated as 360.4:  log likelihood = -1723.27,  aic = 3448.53
```

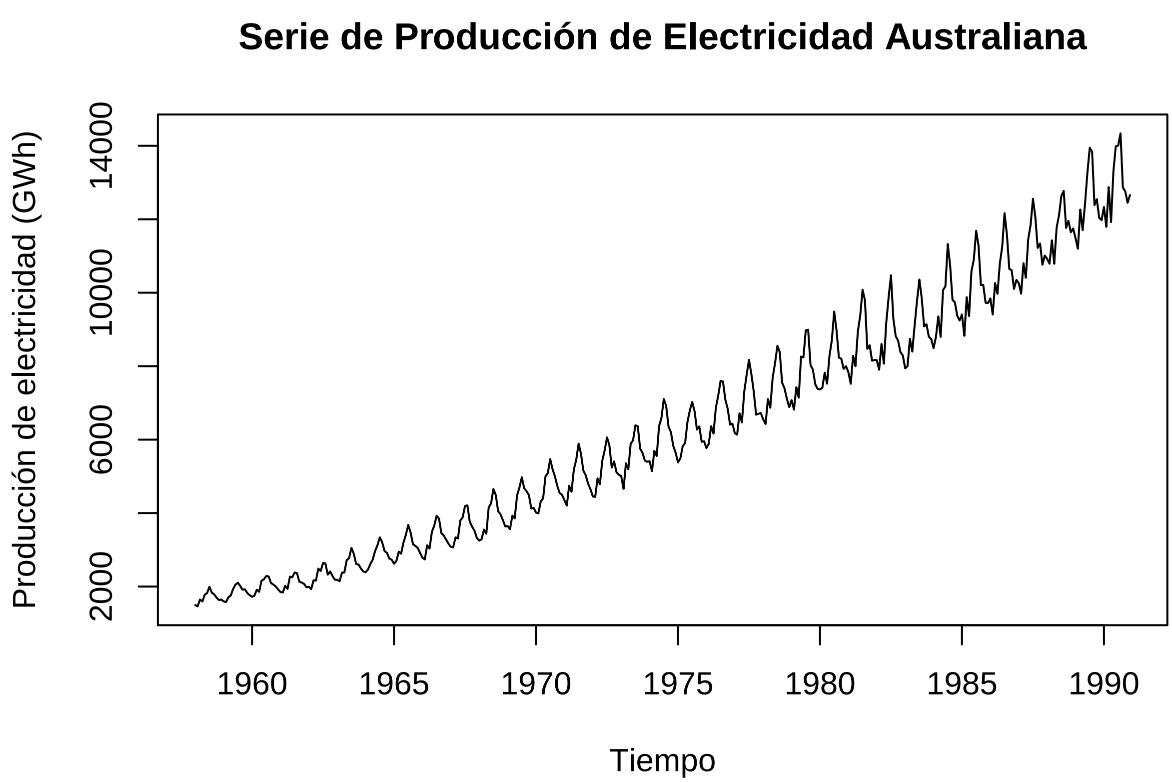
```
###

acf(resid(Beer.ima), main = "")
title(main = "Autocorrelaciones para los Residuales del Ajuste",
      sub = expression(x[t]==x[t-1]+w[t]-0.33*w[t-1]))
```



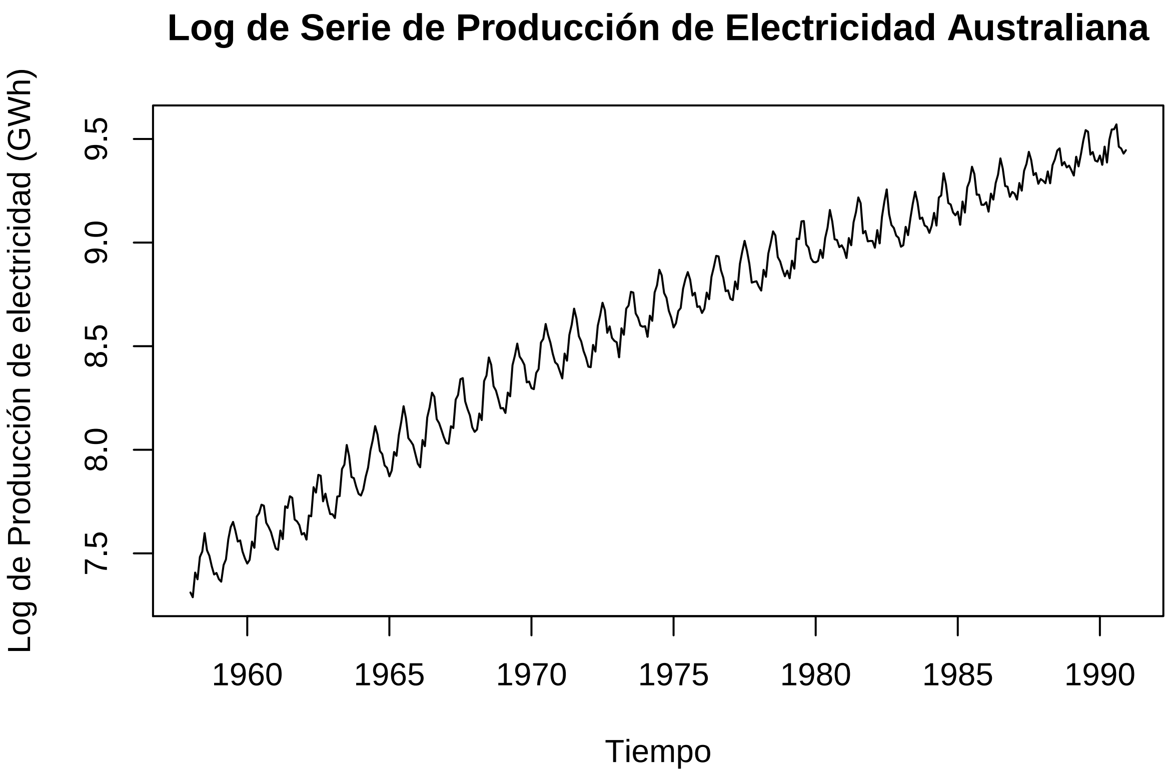

```
###

Beer.1991 <- predict(Beer.ima, n.ahead = 12)
sum(Beer.1991$pred)
```

```
[1] 2365.412
```

```r
#### Modelos Arima estacionales

# Procedimiento de ajuste
# Serie de Producción de Electricidad de Australia

CBE <- read.csv("cbe.csv", header = TRUE)
Elec.ts <- ts(CBE[, 3], start = 1958, freq = 12)
plot(Elec.ts, xlab = "", ylab = "")
title(main = "Serie de Producción de Electricidad Australiana",
      ylab = "Producción de electricidad (GWh)",
      xlab = "Tiempo")
```




```r
###

plot(log(Elec.ts), xlab = "", ylab = "")
title(main = "Log de Serie de Producción de Electricidad Australiana",
      ylab = "Log de Producción de electricidad (GWh)",
      xlab = "Tiempo")
```




```
###

Elec.AR <- arima(log(Elec.ts), order = c(1, 1, 0),
                 seas = list(order = c(1, 0, 0), 12))

Elec.MA <- arima(log(Elec.ts), order = c(0, 1, 1),
                 seas = list(order = c(0, 0, 1), 12))


AIC(Elec.AR)
```

```
[1] -1764.741
```

```
AIC(Elec.MA)
```

```
[1] -1361.586
```

```
###

# Función para buscar un buen modelo

get.best.arima <- function(x.ts, maxord = c(1, 1, 1, 1, 1, 1)){
  best.aic <- 1e8
  n <- length(x.ts)
```

```r
  for(p in 0:maxord[1])for(d in 0:maxord[2])for(q in 0:maxord[3])
    for(P in 0:maxord[4])for(D in 0:maxord[5])for(Q in 0:maxord[6])
    {
      fit <- arima(x.ts, order = c(p, d, q),
                   seas = list(order = c(P, D, Q),
                               frequency(x.ts)), method = "CSS")
      fit.aic <- -2*fit$loglik + (log(n) + 1)*length(fit$coef)
      if(fit.aic < best.aic){
        best.aic <- fit.aic
        best.fit <- fit
        best.model <- c(p, d, q, P, D, Q)
      }
    }
  list(best.aic, best.fit, best.model)
}

# Nuevo ajuste a los datos de la serie transformada de producción
# de electricidad

best.arima.elec <- get.best.arima(log(Elec.ts),
                                  maxord = c(2, 2, 2, 2, 2, 2))
```

```
Warning in stats::arima(x = x, order = order, seasonal = seasonal, xreg =
xreg, : possible convergence problem: optim gave code = 1

Warning in stats::arima(x = x, order = order, seasonal = seasonal, xreg =
xreg, : possible convergence problem: optim gave code = 1

Warning in stats::arima(x = x, order = order, seasonal = seasonal, xreg =
xreg, : possible convergence problem: optim gave code = 1
```

```r
best.fit.elec <- best.arima.elec[[2]]  # Modelo
best.arima.elec[[3]] # Tipo de modelo (órdenes)
```

```
[1] 0 1 1 2 0 2
```

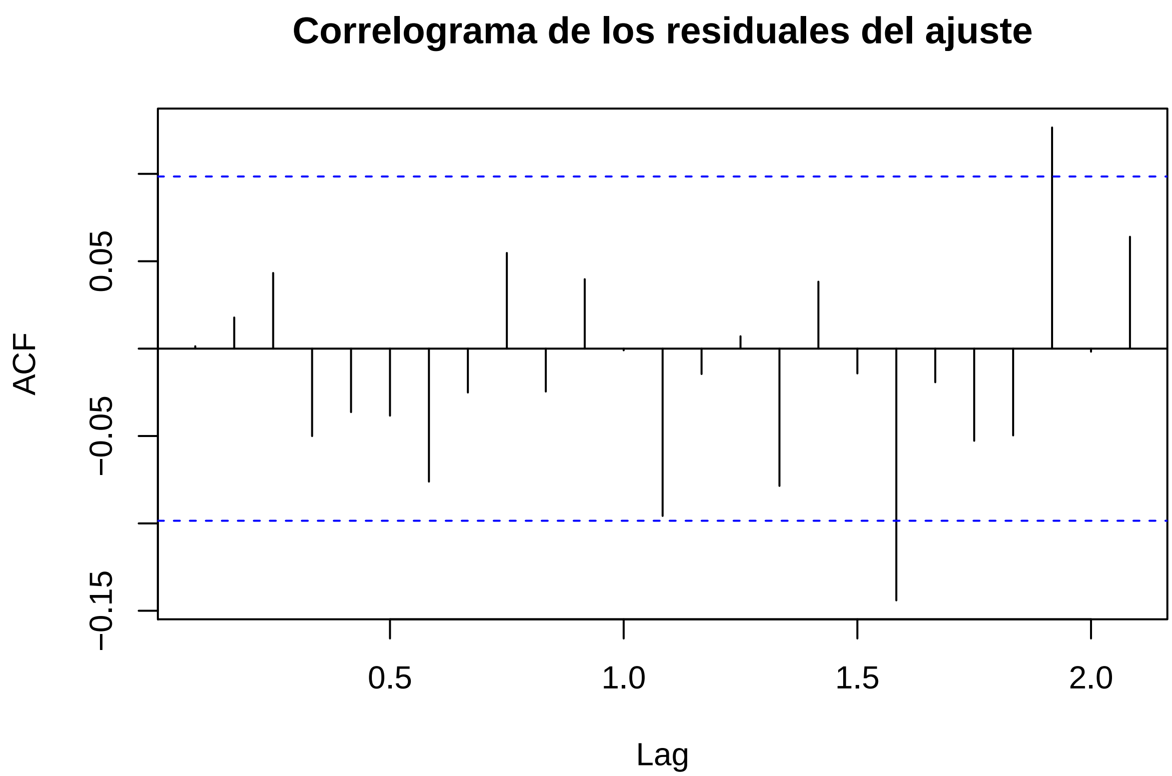
```r
best.fit.elec
```

```
Call:
arima(x = x.ts, order = c(p, d, q), seasonal = list(order = c(P, D, Q), frequency(x.ts)),
    method = "CSS")

Coefficients:
          ma1    sar1    sar2     sma1     sma2
      -0.6566  0.7315  0.2557  -0.3324  -0.3051
s.e.   0.0420  0.1270  0.1259   0.1279   0.0958

sigma^2 estimated as 0.0004161:  part log likelihood = 976.97
```

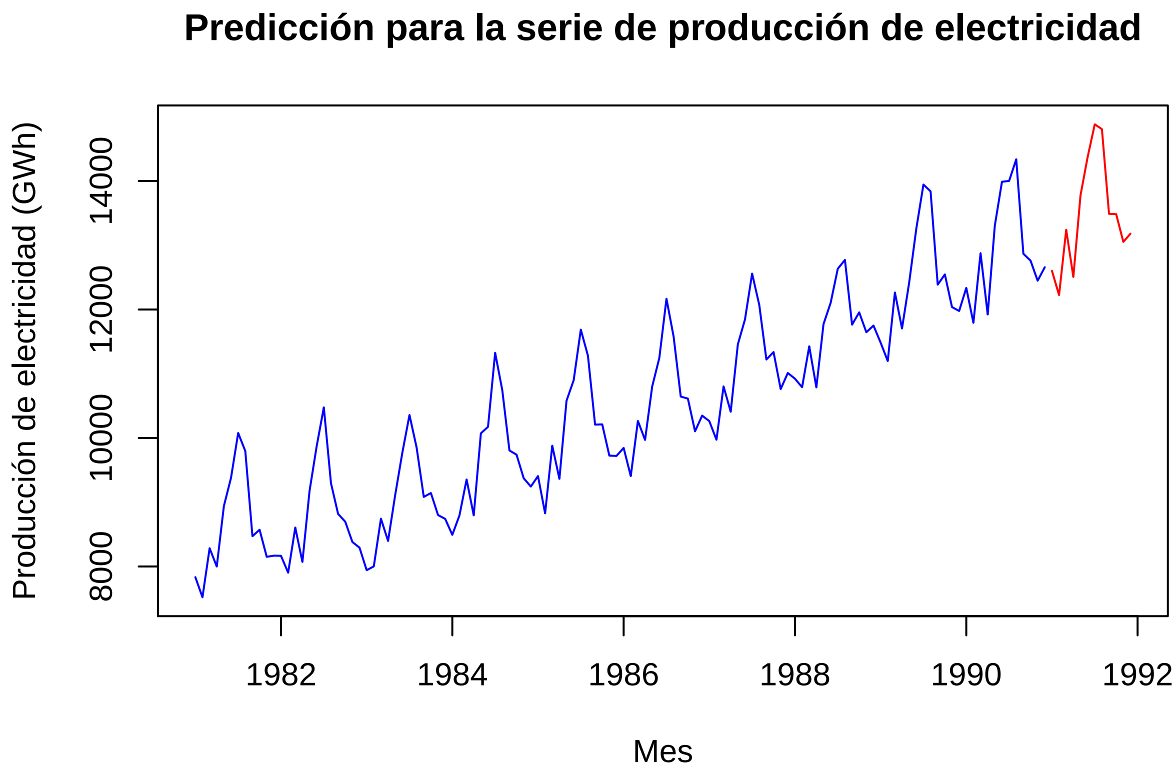
```r
best.arima.elec[[1]] # AIC
```

```
[1] -1919.025
```

```r
###

# ACF para residuales del ajuste

acf(resid(best.fit.elec), main = "")
title(main = "Correlograma de los residuales del ajuste")
```




```r
###
# Predicción

pr <- predict(best.fit.elec, 12)$pred
ts.plot(cbind(window(Elec.ts, start = 1981),
              exp(pr)), col = c("blue", "red"), xlab = "")
title(main = "Predicción para la serie de producción de electricidad",
      xlab = "Mes",
      ylab = "Producción de electricidad (GWh)")
```



RETO 2. SIMULACION DE UN PROCESO ARIMA (1,1,1)

OBJETIVO

Ajustar un modelo a una serie de tiempo simulada, observar que tan adecuado es el ajuste y realizar algunas predicciones.

DESARROLLO

1. Realiza la siguiente simulación con las siguientes características: n = 1000 valores de un proceso ARIMA(1, 1, 1) con parámetros ar = 0.6 y ma = 0.2

2.Ajusta un modelo Arima a la serie simulada para estimar los parámetros y observe las estimaciones de los parámetros

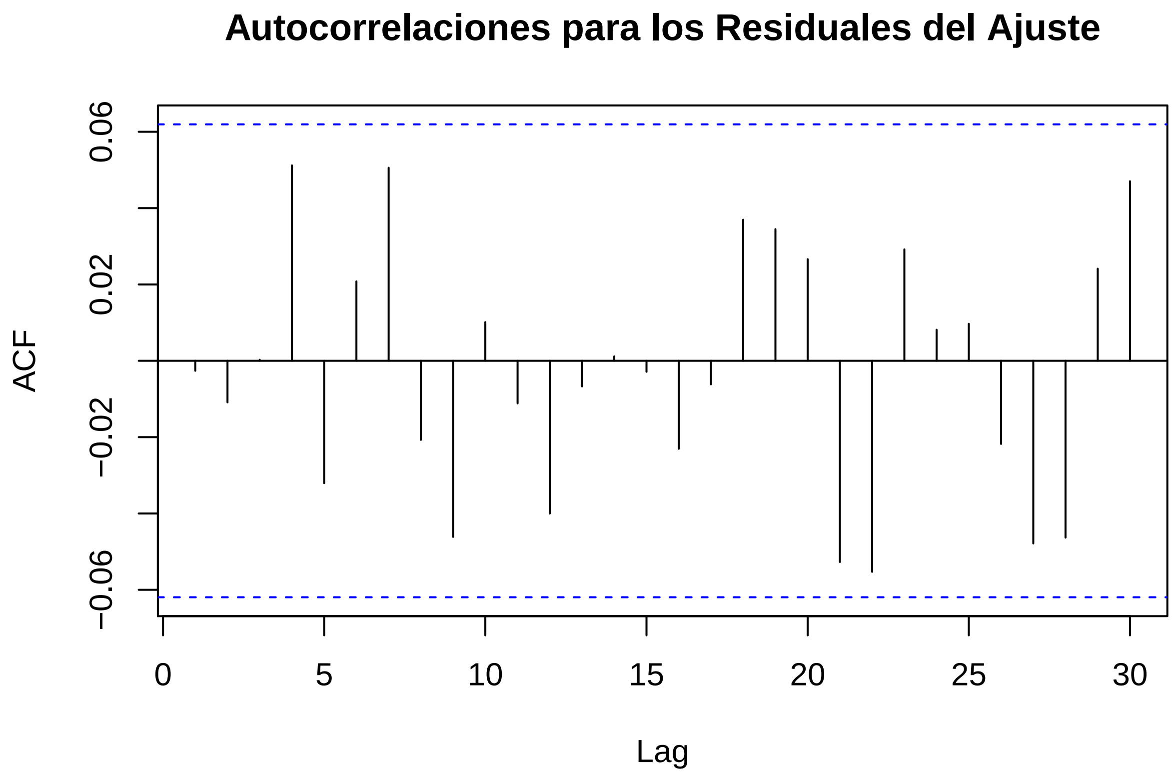
3. Obtén el correlograma de los residuales del ajuste
4. Realiza tres predicciones con ayuda del modelo ajustado y la función *predict*

```
# Reto 2. Proceso ARIMA(1, 1, 1)

# 1. Simula n = 1000 valores de un proceso ARIMA(1, 1, 1) con parámetros
# ar = 0.6 y ma = 0.2
```

```r
# 2. Ajusta un modelo Arima a la serie simulada para estimar los
# parámetros y observa las estimaciones de los parámetros

# 3. Obtén el correlograma de los residuales del ajuste

# 4. Realiza tres predicciones con ayuda del modelo ajustado y la función
# `predict`

# **Solución**

# A continuación, simulamos datos de un modelo ARIMA(1, 1, 1)

set.seed(9)
x <- arima.sim(model = list(order = c(1, 1, 1),
                            ar = 0.6, ma = 0.2), n = 1000)

# ajustamos un modelo a la serie simulada para recuperar los parámetros
# estimados

fit <- arima(x, order = c(1, 1, 1))

# observamos las estimaciones de los parámetros

coefficients(fit)
```

```
      ar1       ma1
0.5694063 0.2042077
```

```r
# obtenemos el correlograma de los residuales del ajuste

acf(resid(fit), main = "")
title(main = "Autocorrelaciones para los Residuales del Ajuste")
```




```
# Llevamos a cabo tres predicciones con el modelo ajustado y la función 'predict'

pred <- predict(fit, n.ahead = 3)
pred$pred
```

```
Time Series:
Start = 1002
End = 1004
Frequency = 1
[1] 27.23524 28.00449 28.44251
```

RETO 3. GRAFICA DE SERIES DE TIEMPO

OBJETIVO

Utilizar las funciones *ts* y *ts.plot* para crear series de tiempo en `R`

DESARROLLO

Con el conjunto de datos soccer.csv realiza lo siguiente

Crea un data frame para el Barcelona que indique el número de goles anotados en cada fecha que ha jugado.

Obtén un data frame que indique el promedio de goles anotados en cada mes que ha jugado

Crea una serie de tiempo mensual para el número promedio de goles anotados por el Barcelona

Realiza los pasos 1 a 3 para el Real Madrid

Muestra en una misma imagen las gráficas de las series de tiempo anteriores

```
# RETO 3 SESION 6
# Reto 3. Gráfica de series de tiempo

# Objetivo

# Utilizar las funciones ‘ts‘ y ‘ts.plot‘ para crear series de tiempo en ‘R‘.

# Requisitos

# Tener instalado R y RStudio
# Haber trabajado con el Prework y el Work

# Desarrollo

# Con el conjunto de datos soccer.csv realiza lo siguiente

# 1. Crea un data frame para el Barcelona que indique el número de goles
# anotados en cada fecha que ha jugado.

# 2. Obtén un data frame que indique el promedio de goles anotados en cada
# mes que ha jugado

# 3. Crea una serie de tiempo mensual para el número promedio de goles
# anotados por el Barcelona

# 4. Realiza los pasos 1 a 3 para el Real Madrid

# 5. Muestra en una misma imagen las gráficas de las series de tiempo
# anteriores

# **Solución**
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
data <- read.csv("soccer.csv")
data <- mutate(data, date = as.Date(date, "%Y-%m-%d"))

# Anotaciones y fechas como local
```

```r
d1 <- data %>% select(date, home.team, home.score) %>%
  filter(home.team == "Barcelona") %>%
  rename(team = home.team, score = home.score)

# Anotaciones y fechas como visitante

d2 <- data %>% select(date, away.team, away.score) %>%
  filter(away.team == "Barcelona") %>%
  rename(team = away.team, score = away.score)

# data frame de anotaciones y fechas

d <- rbind(d1, d2)

# data frame de promedio de anotaciones en cada mes

d <- mutate(d, Ym = format(date, "%Y-%m"))
barca <- d %>% group_by(Ym) %>% summarise(goles = mean(score))

# Creación de la serie de tiempo

# A partir de agosto 2017

(barca <- ts(barca$goles, start = c(1, 1), end = c(3, 5), # Hasta diciembre de 2019
             frequency = 10))
```

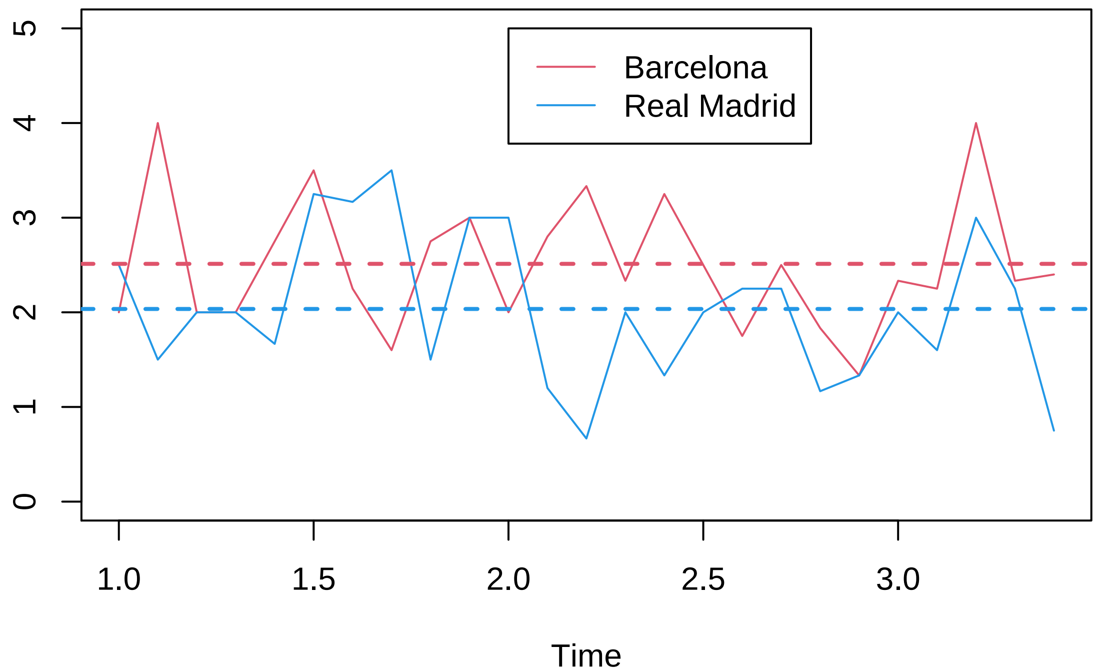
```
Time Series:
Start = c(1, 1)
End = c(3, 5)
Frequency = 10
 [1] 2.000000 4.000000 2.000000 2.000000 2.750000 3.500000 2.250000 1.600000
 [9] 2.750000 3.000000 2.000000 2.800000 3.333333 2.333333 3.250000 2.500000
[17] 1.750000 2.500000 1.833333 1.333333 2.333333 2.250000 4.000000 2.333333
[25] 2.400000
```

```r
######

# Anotaciones y fechas como local

d1 <- data %>% select(date, home.team, home.score) %>%
  filter(home.team == "Real Madrid") %>%
  rename(team = home.team, score = home.score)

# Anotaciones y fechas como visitante

d2 <- data %>% select(date, away.team, away.score) %>%
  filter(away.team == "Real Madrid") %>%
  rename(team = away.team, score = away.score)

# data frame de anotaciones y fechas

d <- rbind(d1, d2)
```

```r
# data frame de promedio de anotaciones en cada mes

d <- mutate(d, Ym = format(date, "%Y-%m"))
realM <- d %>% group_by(Ym) %>% summarise(goles = mean(score))

# Creación de la serie de tiempo

(realM <- ts(realM$goles, start = c(1, 1), # A partir de agosto 2017
             frequency = 10, end = c(3, 5))) # Hasta diciembre de 2019
```

```
Time Series:
Start = c(1, 1)
End = c(3, 5)
Frequency = 10
 [1] 2.5000000 1.5000000 2.0000000 2.0000000 1.6666667 3.2500000 3.1666667
 [8] 3.5000000 1.5000000 3.0000000 3.0000000 1.2000000 0.6666667 2.0000000
[15] 1.3333333 2.0000000 2.2500000 2.2500000 1.1666667 1.3333333 2.0000000
[22] 1.6000000 3.0000000 2.2500000 0.7500000
```

```r
######

# Gráficas de series de tiempo

ts.plot(cbind(barca, realM), col = c(2, 4), ylim = c(0, 5))
abline(h = mean(barca), lwd = 2, col = 2, lty = 2)
abline(h = mean(realM), lwd = 2, col = 4, lty = 2)
legend(x = 2, y = 5,
       legend = c("Barcelona", "Real Madrid"),
       col = c(2, 4), lty = c(1, 1))
```



POSTWORK SESION 6

OBJETIVO

Aprender a crear una serie de tiempo en R

DESARROLLO

Importa el conjunto de datos match.data.csv a R y realiza lo siguiente:

Agrega una nueva columna sumagoles que contenga la suma de goles por partido.

Obtén el promedio por mes de la suma de goles.

Crea la serie de tiempo del promedio por mes de la suma de goles hasta diciembre de 2019.

Grafica la serie de tiempo.

```
# Postwork 6

# Importe el conjunto de datos match.data.csv a 'R':

# 1. Agregue una nueva columna 'sumagoles' que contenga la suma de goles por partido
# 2. Obtenga el promedio por mes de la suma de goles
# 3. Creé la serie de tiempo del promedio por mes de la suma de goles hasta diciembre de 2019
```

```r
# 4. Grafique la serie de tiempo

# Solución

# Primero cargamos el paquete que utilizaremos para manipular los datos

library(dplyr)

# Establecemos nuestro directorio de trabajo que deberá contener el archivo csv que importaremos a `R`

# Importamos los datos a `R`

data <- read.csv("match.data.csv")

# 1. # 2.

# Agregamos la columna `sumagoles` y obtenemos el promedio por mes
# de la suma de goles

nd <- data %>%
  mutate(date = as.Date(date, "%Y-%m-%d"),
         sumagoles = home.score + away.score) %>%
  mutate(Ames = format(date, "%Y-%m")) %>%
  group_by(Ames) %>%
  summarise(promgoles = mean(sumagoles))

(nd <- as.data.frame(nd))
```

```
      Ames promgoles
1  2010-08  2.200000
2  2010-09  2.425000
3  2010-10  3.025641
4  2010-11  2.902439
5  2010-12  2.733333
6  2011-01  3.000000
7  2011-02  2.325000
8  2011-03  2.400000
9  2011-04  2.930233
10 2011-05  2.957447
11 2011-08  3.000000
12 2011-09  2.525000
13 2011-10  2.420000
14 2011-11  2.833333
15 2011-12  2.900000
16 2012-01  2.550000
17 2012-02  3.050000
18 2012-03  2.981818
19 2012-04  2.854545
20 2012-05  2.700000
21 2012-08  3.000000
22 2012-09  2.871795
23 2012-10  2.838710
24 2012-11  2.829268
```

25	2012-12	2.794872
26	2013-01	3.025000
27	2013-02	2.750000
28	2013-03	2.657895
29	2013-04	3.023810
30	2013-05	2.725000
31	2013-06	3.800000
32	2013-08	2.920000
33	2013-09	2.711111
34	2013-10	2.850000
35	2013-11	3.166667
36	2013-12	3.125000
37	2014-01	2.902439
38	2014-02	2.500000
39	2014-03	2.474576
40	2014-04	2.769231
41	2014-05	2.387097
42	2014-08	2.400000
43	2014-09	2.650000
44	2014-10	2.903226
45	2014-11	2.631579
46	2014-12	2.400000
47	2015-01	2.555556
48	2015-02	2.780488
49	2015-03	2.200000
50	2015-04	2.633333
51	2015-05	3.225000
52	2015-08	1.750000
53	2015-09	2.650000
54	2015-10	3.055556
55	2015-11	2.441176
56	2015-12	2.435897
57	2016-01	3.204082
58	2016-02	2.714286
59	2016-03	3.025000
60	2016-04	2.727273
61	2016-05	2.880000
62	2016-08	2.850000
63	2016-09	2.878049
64	2016-10	3.384615
65	2016-11	2.600000
66	2016-12	2.862069
67	2017-01	2.675000
68	2017-02	2.880952
69	2017-03	2.948718
70	2017-04	2.985294
71	2017-05	3.250000
72	2017-08	2.300000
73	2017-09	2.800000
74	2017-10	2.914286
75	2017-11	3.033333
76	2017-12	2.128205
77	2018-01	3.200000
78	2018-02	2.604167

```
79  2018-03  2.513514
80  2018-04  2.296296
81  2018-05  3.312500
82  2018-08  2.260870
83  2018-09  2.644444
84  2018-10  2.375000
85  2018-11  2.709677
86  2018-12  2.526316
87  2019-01  2.756098
88  2019-02  2.300000
89  2019-03  2.725000
90  2019-04  2.550000
91  2019-05  2.966667
92  2019-08  2.269231
93  2019-09  2.590909
94  2019-10  2.564103
95  2019-11  2.742857
96  2019-12  2.777778
97  2020-01  2.133333
98  2020-02  2.590909
99  2020-03  2.375000
100 2020-06  2.415094
101 2020-07  2.263158
```

```
(nd <- nd %>% filter(Ames != "2013-06"))
```

```
       Ames promgoles
1   2010-08  2.200000
2   2010-09  2.425000
3   2010-10  3.025641
4   2010-11  2.902439
5   2010-12  2.733333
6   2011-01  3.000000
7   2011-02  2.325000
8   2011-03  2.400000
9   2011-04  2.930233
10  2011-05  2.957447
11  2011-08  3.000000
12  2011-09  2.525000
13  2011-10  2.420000
14  2011-11  2.833333
15  2011-12  2.900000
16  2012-01  2.550000
17  2012-02  3.050000
18  2012-03  2.981818
19  2012-04  2.854545
20  2012-05  2.700000
21  2012-08  3.000000
22  2012-09  2.871795
23  2012-10  2.838710
24  2012-11  2.829268
25  2012-12  2.794872
26  2013-01  3.025000
27  2013-02  2.750000
```

28	2013-03	2.657895
29	2013-04	3.023810
30	2013-05	2.725000
31	2013-08	2.920000
32	2013-09	2.711111
33	2013-10	2.850000
34	2013-11	3.166667
35	2013-12	3.125000
36	2014-01	2.902439
37	2014-02	2.500000
38	2014-03	2.474576
39	2014-04	2.769231
40	2014-05	2.387097
41	2014-08	2.400000
42	2014-09	2.650000
43	2014-10	2.903226
44	2014-11	2.631579
45	2014-12	2.400000
46	2015-01	2.555556
47	2015-02	2.780488
48	2015-03	2.200000
49	2015-04	2.633333
50	2015-05	3.225000
51	2015-08	1.750000
52	2015-09	2.650000
53	2015-10	3.055556
54	2015-11	2.441176
55	2015-12	2.435897
56	2016-01	3.204082
57	2016-02	2.714286
58	2016-03	3.025000
59	2016-04	2.727273
60	2016-05	2.880000
61	2016-08	2.850000
62	2016-09	2.878049
63	2016-10	3.384615
64	2016-11	2.600000
65	2016-12	2.862069
66	2017-01	2.675000
67	2017-02	2.880952
68	2017-03	2.948718
69	2017-04	2.985294
70	2017-05	3.250000
71	2017-08	2.300000
72	2017-09	2.800000
73	2017-10	2.914286
74	2017-11	3.033333
75	2017-12	2.128205
76	2018-01	3.200000
77	2018-02	2.604167
78	2018-03	2.513514
79	2018-04	2.296296
80	2018-05	3.312500
81	2018-08	2.260870

```
82  2018-09  2.644444
83  2018-10  2.375000
84  2018-11  2.709677
85  2018-12  2.526316
86  2019-01  2.756098
87  2019-02  2.300000
88  2019-03  2.725000
89  2019-04  2.550000
90  2019-05  2.966667
91  2019-08  2.269231
92  2019-09  2.590909
93  2019-10  2.564103
94  2019-11  2.742857
95  2019-12  2.777778
96  2020-01  2.133333
97  2020-02  2.590909
98  2020-03  2.375000
99  2020-06  2.415094
100 2020-07  2.263158
```

```
(nd <- nd[1:95,])
```

```
      Ames promgoles
1  2010-08  2.200000
2  2010-09  2.425000
3  2010-10  3.025641
4  2010-11  2.902439
5  2010-12  2.733333
6  2011-01  3.000000
7  2011-02  2.325000
8  2011-03  2.400000
9  2011-04  2.930233
10 2011-05  2.957447
11 2011-08  3.000000
12 2011-09  2.525000
13 2011-10  2.420000
14 2011-11  2.833333
15 2011-12  2.900000
16 2012-01  2.550000
17 2012-02  3.050000
18 2012-03  2.981818
19 2012-04  2.854545
20 2012-05  2.700000
21 2012-08  3.000000
22 2012-09  2.871795
23 2012-10  2.838710
24 2012-11  2.829268
25 2012-12  2.794872
26 2013-01  3.025000
27 2013-02  2.750000
28 2013-03  2.657895
29 2013-04  3.023810
30 2013-05  2.725000
31 2013-08  2.920000
```

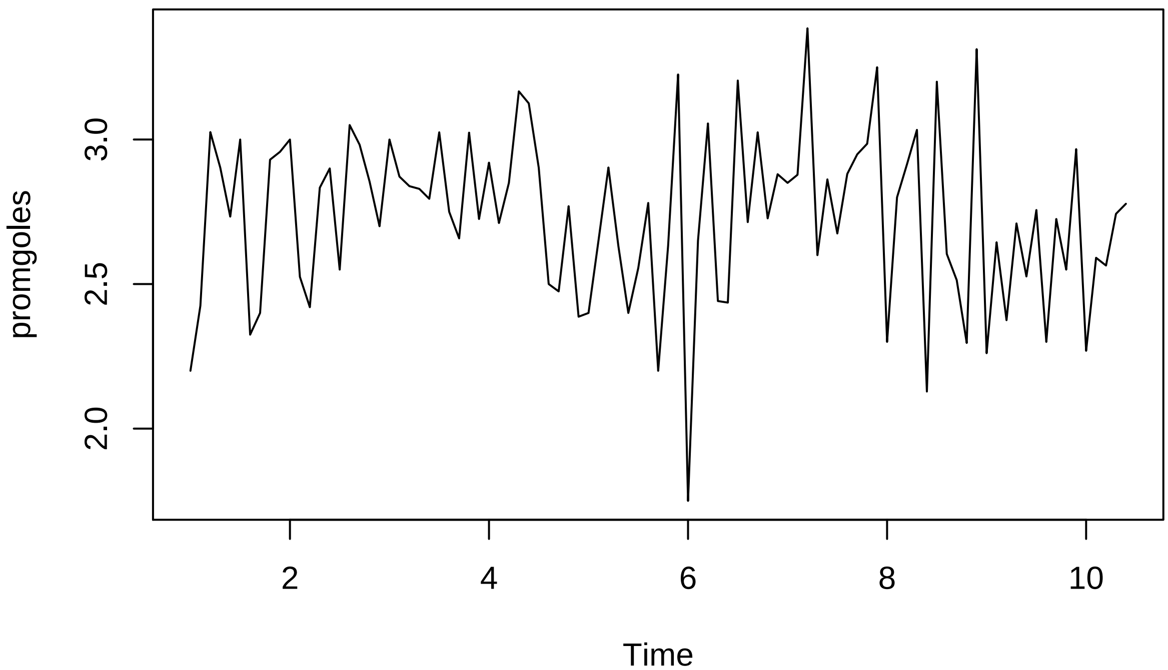
32	2013-09	2.711111
33	2013-10	2.850000
34	2013-11	3.166667
35	2013-12	3.125000
36	2014-01	2.902439
37	2014-02	2.500000
38	2014-03	2.474576
39	2014-04	2.769231
40	2014-05	2.387097
41	2014-08	2.400000
42	2014-09	2.650000
43	2014-10	2.903226
44	2014-11	2.631579
45	2014-12	2.400000
46	2015-01	2.555556
47	2015-02	2.780488
48	2015-03	2.200000
49	2015-04	2.633333
50	2015-05	3.225000
51	2015-08	1.750000
52	2015-09	2.650000
53	2015-10	3.055556
54	2015-11	2.441176
55	2015-12	2.435897
56	2016-01	3.204082
57	2016-02	2.714286
58	2016-03	3.025000
59	2016-04	2.727273
60	2016-05	2.880000
61	2016-08	2.850000
62	2016-09	2.878049
63	2016-10	3.384615
64	2016-11	2.600000
65	2016-12	2.862069
66	2017-01	2.675000
67	2017-02	2.880952
68	2017-03	2.948718
69	2017-04	2.985294
70	2017-05	3.250000
71	2017-08	2.300000
72	2017-09	2.800000
73	2017-10	2.914286
74	2017-11	3.033333
75	2017-12	2.128205
76	2018-01	3.200000
77	2018-02	2.604167
78	2018-03	2.513514
79	2018-04	2.296296
80	2018-05	3.312500
81	2018-08	2.260870
82	2018-09	2.644444
83	2018-10	2.375000
84	2018-11	2.709677
85	2018-12	2.526316

```
86 2019-01  2.756098
87 2019-02  2.300000
88 2019-03  2.725000
89 2019-04  2.550000
90 2019-05  2.966667
91 2019-08  2.269231
92 2019-09  2.590909
93 2019-10  2.564103
94 2019-11  2.742857
95 2019-12  2.777778
```

```r
# A partir de agosto de 2010

# 3. Creamos la serie de tiempo en `R`

(promgoles <- ts(nd$promgoles, start = 1,
                frequency = 10))
```

```
Time Series:
Start = c(1, 1)
End = c(10, 5)
Frequency = 10
 [1] 2.200000 2.425000 3.025641 2.902439 2.733333 3.000000 2.325000 2.400000
 [9] 2.930233 2.957447 3.000000 2.525000 2.420000 2.833333 2.900000 2.550000
[17] 3.050000 2.981818 2.854545 2.700000 3.000000 2.871795 2.838710 2.829268
[25] 2.794872 3.025000 2.750000 2.657895 3.023810 2.725000 2.920000 2.711111
[33] 2.850000 3.166667 3.125000 2.902439 2.500000 2.474576 2.769231 2.387097
[41] 2.400000 2.650000 2.903226 2.631579 2.400000 2.555556 2.780488 2.200000
[49] 2.633333 3.225000 1.750000 2.650000 3.055556 2.441176 2.435897 3.204082
[57] 2.714286 3.025000 2.727273 2.880000 2.850000 2.878049 3.384615 2.600000
[65] 2.862069 2.675000 2.880952 2.948718 2.985294 3.250000 2.300000 2.800000
[73] 2.914286 3.033333 2.128205 3.200000 2.604167 2.513514 2.296296 3.312500
[81] 2.260870 2.644444 2.375000 2.709677 2.526316 2.756098 2.300000 2.725000
[89] 2.550000 2.966667 2.269231 2.590909 2.564103 2.742857 2.777778
```

```r
# 4. Graficamos la serie de tiempo

ts.plot(promgoles)
```



promgoles
2.0
2.5
3.0
2
4
6
8
10
Time