Forecasting de Series Temporales

Dr. Marcelo Risk

Data Mining de Series Temporales, Maestría en Explotación de Datos y Descubrimiento de Conocimientos, FCEyN UBA

2020

Sinopsis

Introducción al Forecasting

- **Forecasting**: *es la predicción del futuro (!!!), en base al pasado...*
- **Condición fundamental**: el pasado debe ser *estacionario*, entonces debe pasar pruebas de *estacionaridad*.
- **Métodos**: forecasting lineal, STL y ARIMA.

Estacionaridad de una serie de tiempo (ST)

- Definición: una serie se considera estacional cuando su media y varianza no varían con el tiempo.
  - **ST estacionaria**: modelización con TF y ARMA, es posible forecasting.
  - **ST no estacionaria**: desestacionarla, ó utilizar métodos tiempo-frecuencia (wavelets), no es posible forecasting.
- Pruebas de estacionaridad: **Augmented Dickey-Fuller test** y **KPSS Test for Level Stationarity**.

Pruebas de estacionaridad: **Augmented Dickey-Fuller test**

- Es una prueba de la *raiz unitaria* (unit root) de un proceso estocástico (aleatorio), en el caso que la *raiz unitaria* es 1, es no estacionario.
- La H_0 la ST tiene *raiz unitaria*, y la H_1 es estacionaria.
- Función **adf.test()** en R.

Pruebas de estacionaridad: **KPSS (Kwiatkowski, Phillips, Schmidt, and Shin) Test**

- Es una prueba de estacionaridad.
- La H_0 la ST es estacionaria, y la H_1 tiene *raiz unitaria*.
- Función **kpss.test()** en R.

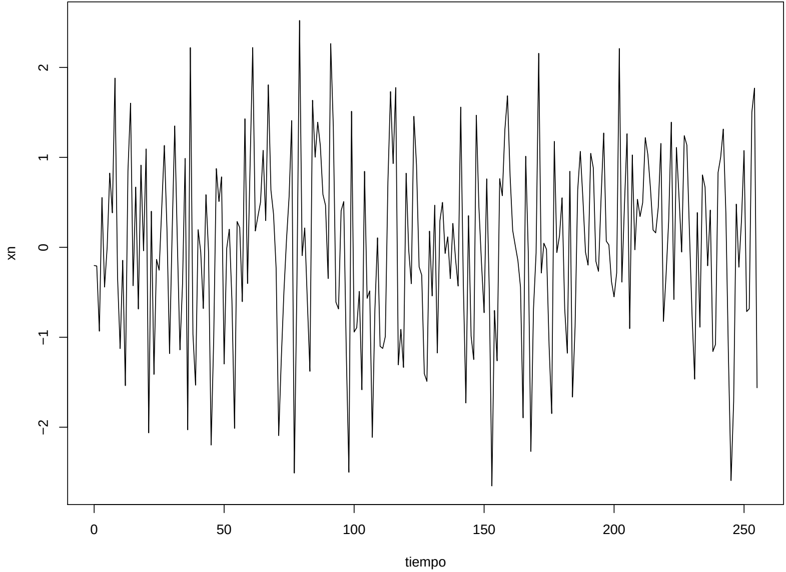
Pruebas de estacionaridad: combinación de resultados de ADF y KPSS

ADF	**KPSS**	**decisión**
$P < 0{,}05$	$P \geq 0{,}05$	estacionaria
$P < 0{,}05$	$P < 0{,}05$	indefinida
$P \geq 0{,}05$	$P \geq 0{,}05$	indefinida
$P \geq 0{,}05$	$P < 0{,}05$	no estacionaria

Ejemplo ST estacionaria

```r
library(tseries)

N = 256
tiempo = 0:(N-1)

xn <- rnorm(N)
plot(tiempo,xn,type='l')
```

Ejemplo ST estacionaria



Ejemplo ST estacionaria

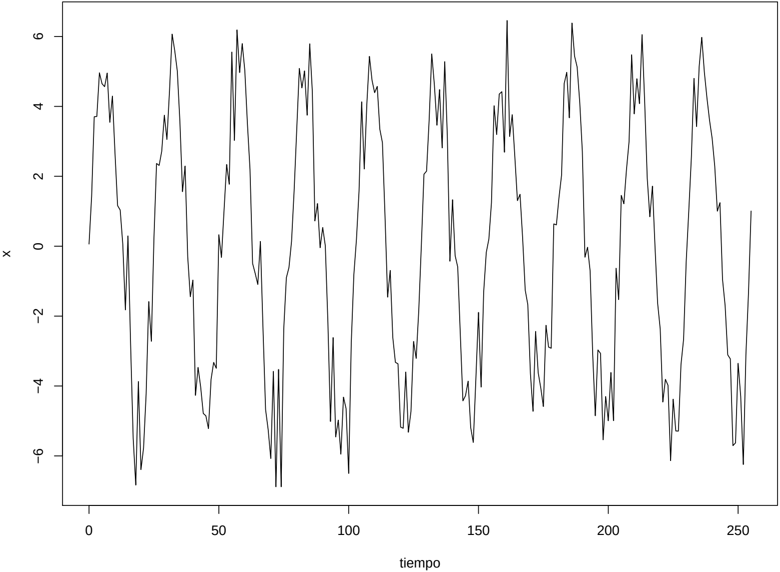
```
print(adf.test(xn))
# Augmented Dickey-Fuller Test
# data:  xn
# Dickey-Fuller = -5.8844, Lag order = 6, p-value = 0.01
# alternative hypothesis: stationary

print(kpss.test(xn))
# KPSS Test for Level Stationarity
# data:  xn
# KPSS Level = 0.10597, Truncation lag parameter = 3,
    p-value = 0.1
```

Decisión: **ST estacionaria**

Ejemplo ST estacionaria

```
N = 256
tiempo = 0:(N-1)

x <- 5*sin(10*2*pi*tiempo/N) + rnorm(N)
plot(tiempo,x,type='l')
```

Ejemplo ST estacionaria



Ejemplo ST estacionaria

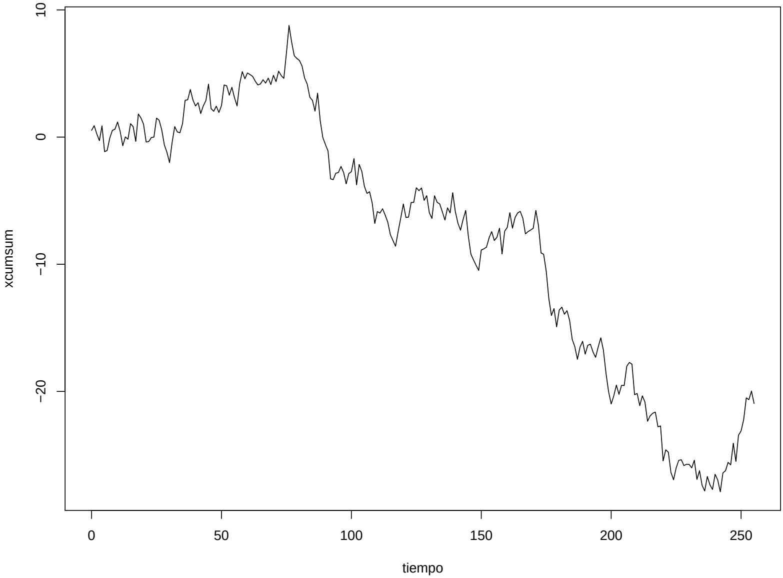
```
print(adf.test(x))
# Augmented Dickey-Fuller Test
# data:  x
# Dickey-Fuller = -11.894, Lag order = 6, p-value = 0.01
#  alternative  hypothesis:  stationary

print(kpss.test(x))
# KPSS Test for Level  Stationarity
# data:  x
# KPSS Level = 0.04288, Truncation lag parameter = 3,
    p-value = 0.1
```

Decisión: **ST estacionaria**

Ejemplo ST no estacionaria

```
N = 256
tiempo = 0:(N-1)

e <- rnorm(N)
xcumsum <- cumsum(e)
plot(tiempo,xcumsum,type='l')
```

Ejemplo ST no estacionaria



Ejemplo ST no estacionaria

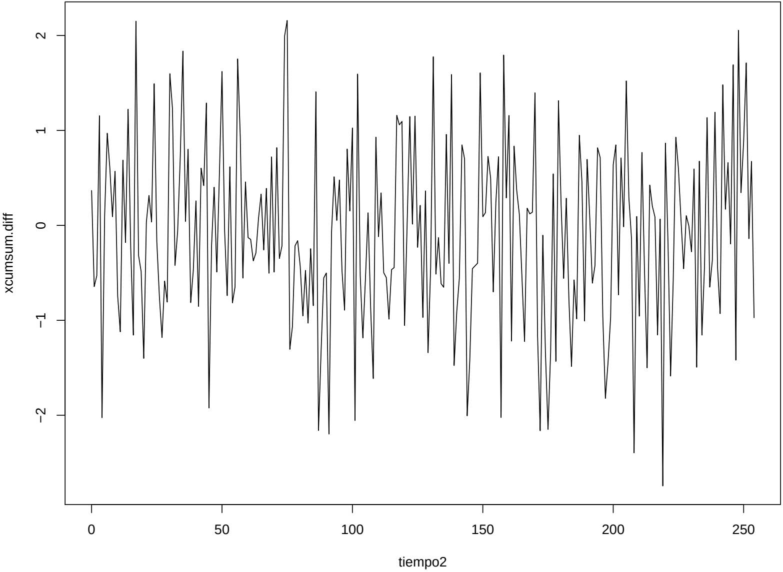
```
print(adf.test(xcumsum))
# Augmented Dickey-Fuller Test
# data:  xcumsum
# Dickey-Fuller = -2.5171, Lag order = 6, p-value = 0.3581
# alternative hypothesis: stationary

print(kpss.test(xcumsum))
# KPSS Test for Level Stationarity
# data:  xcumsum
# KPSS Level = 5.7438, Truncation lag parameter = 3, p-value
    = 0.01
```

Decisión: **ST no estacionaria**

Ejemplo ST no estacionaria, uso de **diff()**

```
xcumsum.diff <- diff(xcumsum)
tiempo2 <- tiempo
length(tiempo2) <- length(xcumsum.diff)
plot(tiempo2,xcumsum.diff,type='l')
```

Ejemplo ST no estacionaria, uso de **diff()**



Ejemplo ST no estacionaria, uso de **diff()**

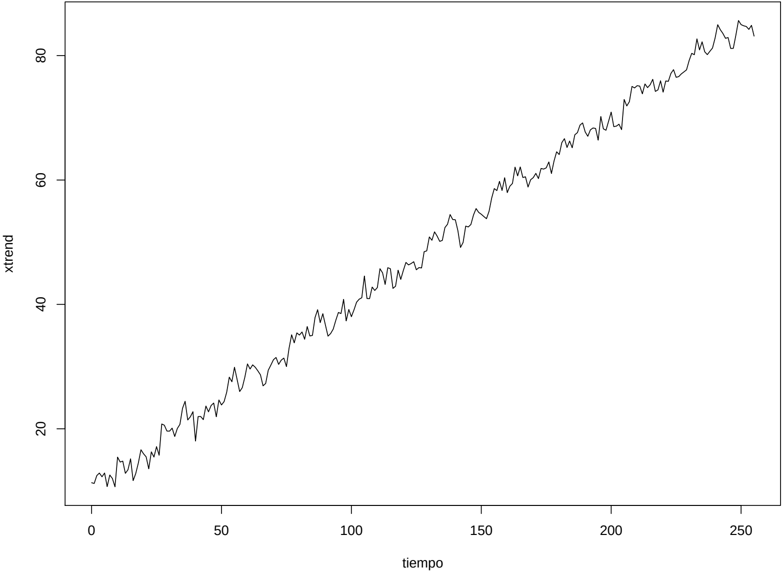
```
print(adf.test(xcumsum))
# Augmented Dickey-Fuller Test
# data:  xcumsum.diff
# Dickey-Fuller = -5.052, Lag order = 6, p-value = 0.01
#  alternative  hypothesis:  stationary

print(kpss.test(xcumsum))
# KPSS Test for Level  Stationarity
# data:  xcumsum.diff
# KPSS Level = 0.1551, Truncation lag parameter = 3, p-value
    = 0.1
```

Decisión: **ST estacionaria !!**

Ejemplo ST indefinida

```
m = 0.3
b0 = 10
xtrend <- sin(10*2*pi*tiempo/N) + sin(30*2*pi*tiempo/N) +
    rnorm(N) + m*tiempo + b0
plot(tiempo,xtrend,type='l')
```

Ejemplo ST indefinida



Ejemplo ST indefinida

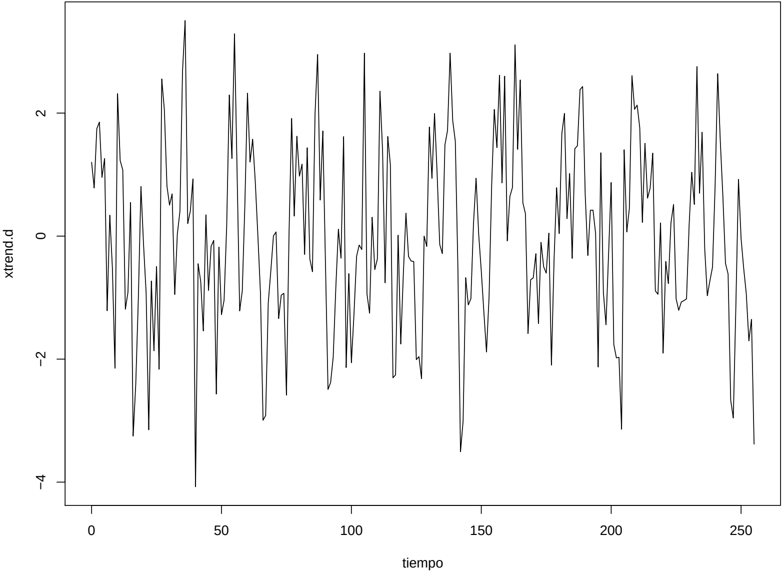
```
print(adf.test(xtrend))
# Augmented Dickey-Fuller Test
# data:  xtrend
# Dickey-Fuller = -4.7415, Lag order = 6, p-value = 0.01
#  alternative  hypothesis:  stationary

print(kpss.test(xtrend))
# KPSS Test for Level  Stationarity
# data:  xtrend
# KPSS Level = 6.4856, Truncation lag parameter = 3, p-value
    = 0.01
```

Decisión: **ST indefinida...**

Ejemplo ST indefinida, detrended

```
lm.xtrend <- lm(xtrend ~ tiempo)
print(summary(lm.xtrend))
xtrend.d <- xtrend - (lm.xtrend$coefficients[1] +
    lm.xtrend$coefficients[2]*tiempo)
plot(tiempo,xtrend.d,type='l')
```

Ejemplo ST indefinida, detrended



Ejemplo ST indefinida, detrended

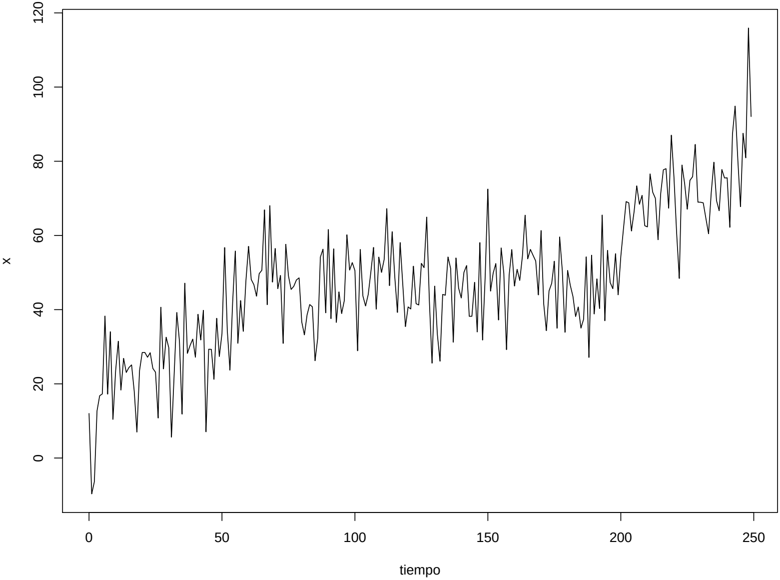
```
print(adf.test(xtrend.d))
# Augmented Dickey-Fuller Test
# data:  xtrend
# Dickey-Fuller = -4.7415, Lag order = 6, p-value = 0.01
#  alternative  hypothesis:  stationary

print(kpss.test(xtrend.d))
# KPSS Test for Level  Stationarity
# data:  xtrend
# KPSS Level = 0.047681, Truncation lag parameter = 3,
    p-value = 0.1
```

Decisión: **ST estacionaria !!**

Forecasting Lineal

```
N = 250
tiempo = 0:(N-1)

m = 0.3
b0 = 10
x <- 15*sin(2*pi*tiempo/N) + 5*sin(5*2*pi*tiempo/N) +
    rnorm(N,sd=10) + m*tiempo + b0
x <- ts(x)

plot(tiempo,x,type='l')
```

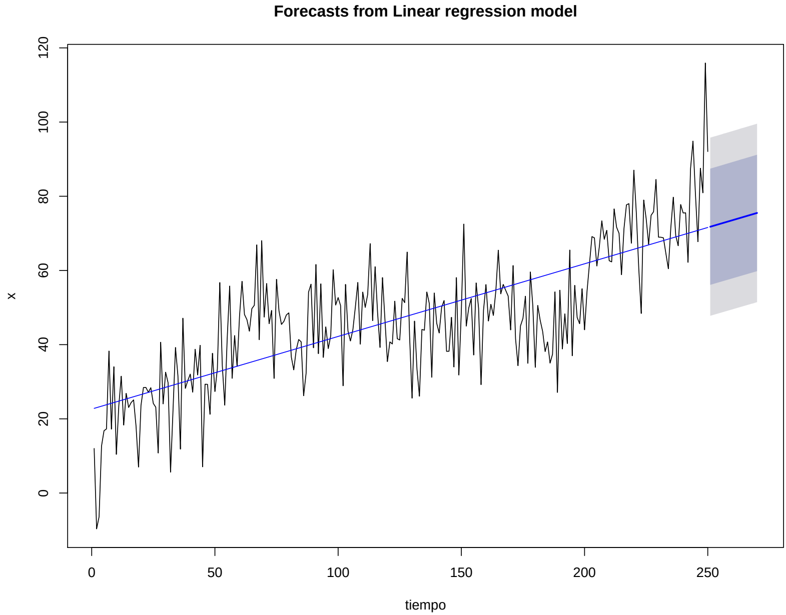
Forecasting Lineal



Forecasting Lineal

```
fit.tslm <- tslm(x ~ trend)
f <- forecast(fit.tslm, h=20,level=c(80,95))
plot(f, ylab="x", xlab="tiempo")
lines(fitted(fit.tslm),col="blue")
summary(fit.tslm)
# Coefficients:
#               Estimate Std. Error t value Pr(>|t|)
# (Intercept)   22.6260     1.5340   14.75   <2e-16 ***
# trend          0.1958     0.0106   18.48   <2e-16 ***
```

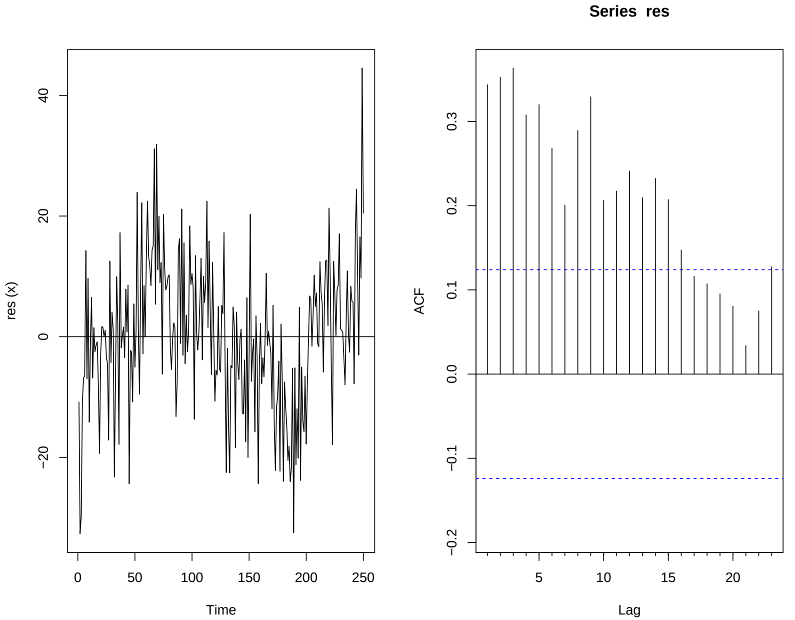
Forecasting Lineal



Forecasting Lineal

```r
par(mfrow=c(1,2))
res <- ts(resid(fit.tslm))
plot.ts(res, ylab="res (x)")
abline(0,0)
Acf(res)
```

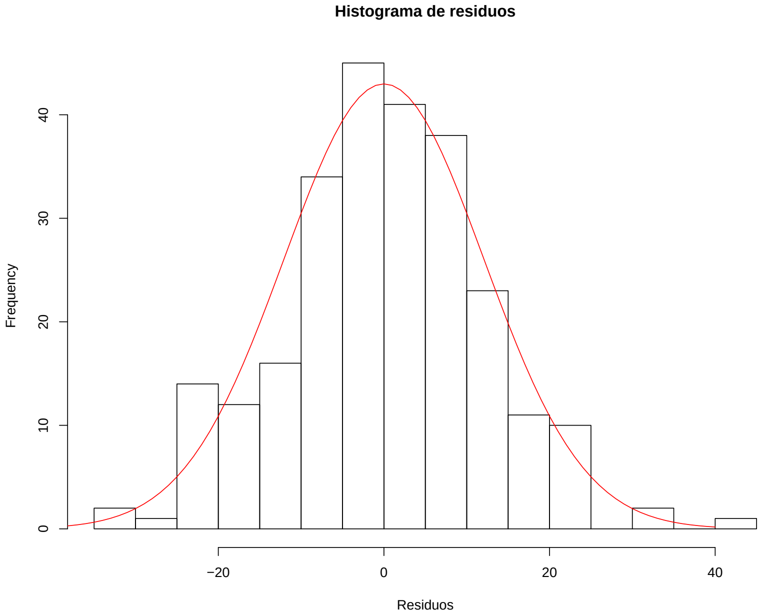
Forecasting Lineal



Forecasting Lineal

```
print(dwtest(fit.tslm, alt="two.sided"))
# Durbin-Watson test
# data:  fit.tslm
# DW = 1.2978, p-value = 1.719e-08
# alternative hypothesis: true autocorrelation is not 0

par(mfrow=c(1,1))
bins <- hist(res, breaks='FD', xlab='Residuos',
     main='Histograma de residuos')
xx <- -40:40
lines(xx, 1300*dnorm(xx,0,sd(res)), col=2)
```

Forecasting Lineal



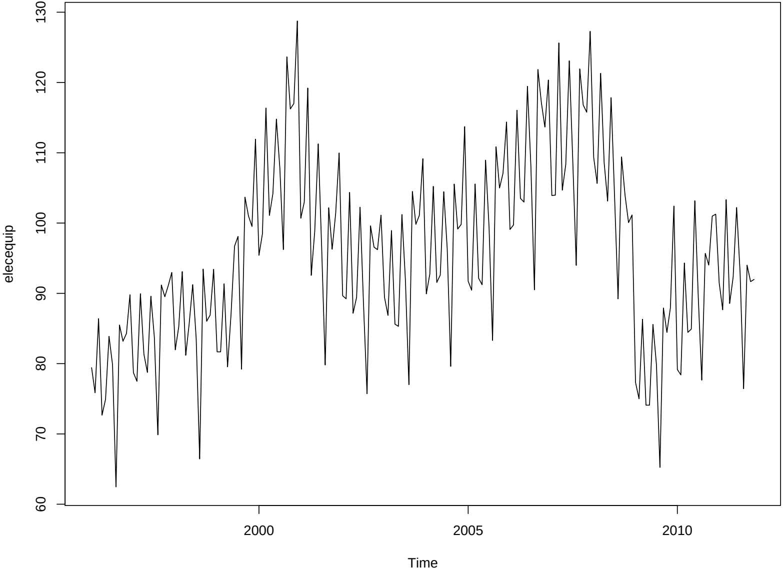
Descomposición clásica

- Descompone una ST en sus componentes estacionales, tendencia e irregular (resto).
- Utiliza medias móviles.
- Opciones componentes aditivas o multiplicativas.

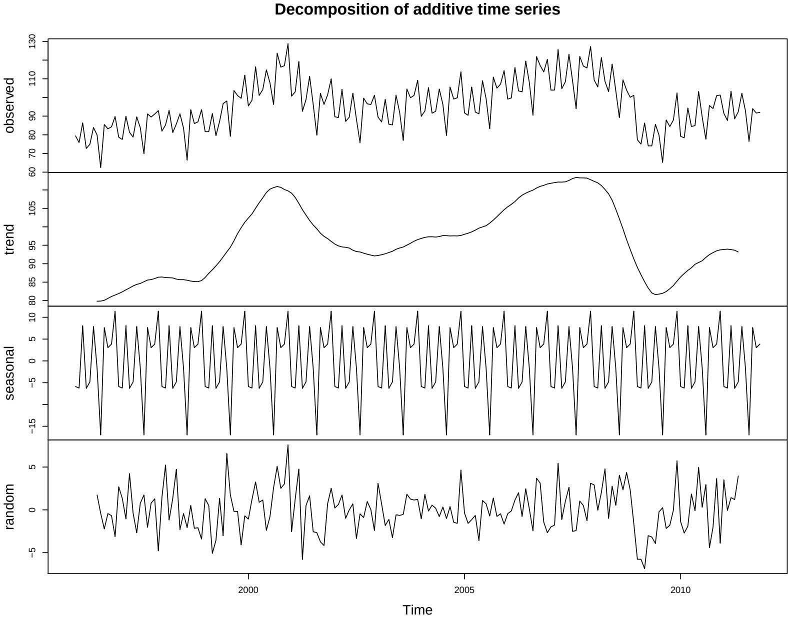
Descomposición clásica

```r
library (fpp)

data( elecequip ) # viene en el fpp
plot( elecequip )

fit .decomp <- decompose(elecequip, type='additive')
plot( fit .decomp)
```

Descomposición clásica



Descomposición clásica



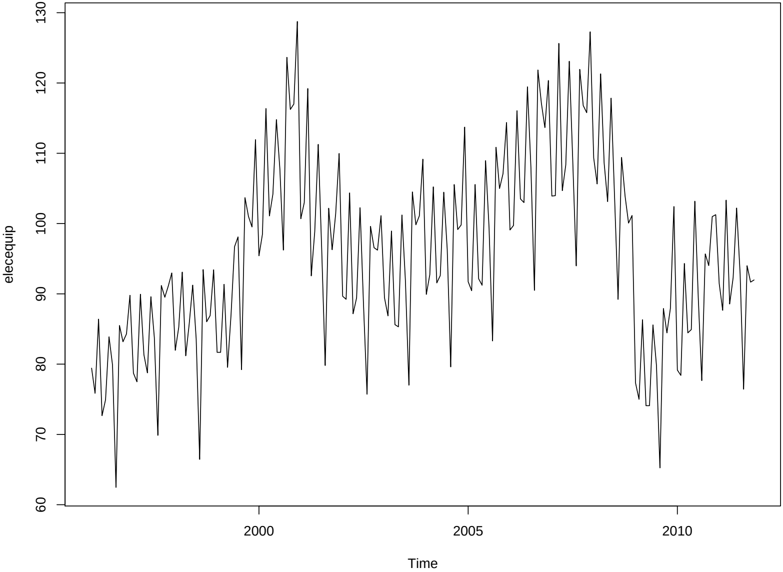
Forecasting STL (Seasonal and Trend decomposition using Loess)

- Descompone una ST en sus componentes estacionales, tendencia e irregular (resto).
- Utiliza el algoritmo **loess**, el cual realiza ajustes locales para una ventana.
- Opciones componentes aditivas o multiplicativas.

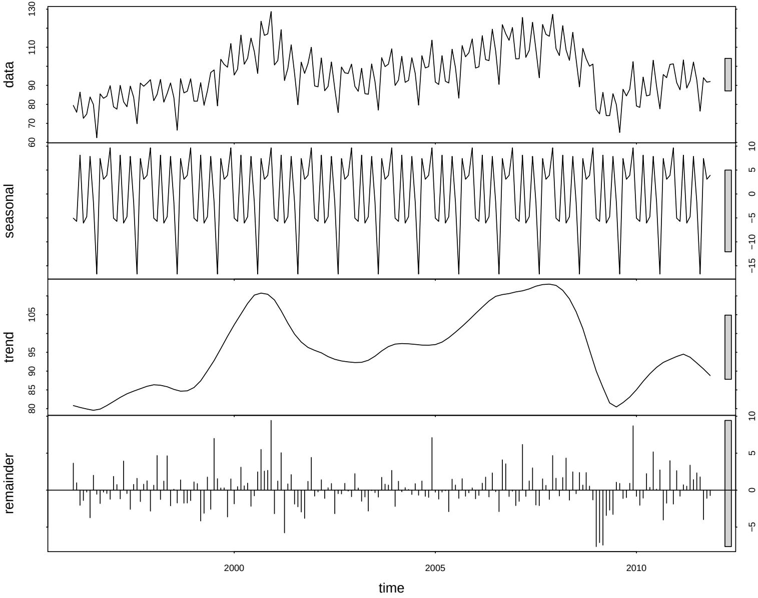
Forecasting STL

```r
library(fpp)

data(elecequip) # viene en el fpp
plot(elecequip)

fit.stl <- stl(elecequip, t.window=15, s.window="periodic",
    robust=TRUE)
plot(fit.stl)

eeadj <- seasadj(fit.stl) # remueve la componente estacional
plot(naive(eeadj), xlab="index",
     main="Forecasting de datos con estacionalidad removida")

fcast <- forecast(fit.stl, method="naive")
plot(fcast, ylab="index",
     main="Forecasting completo")
```

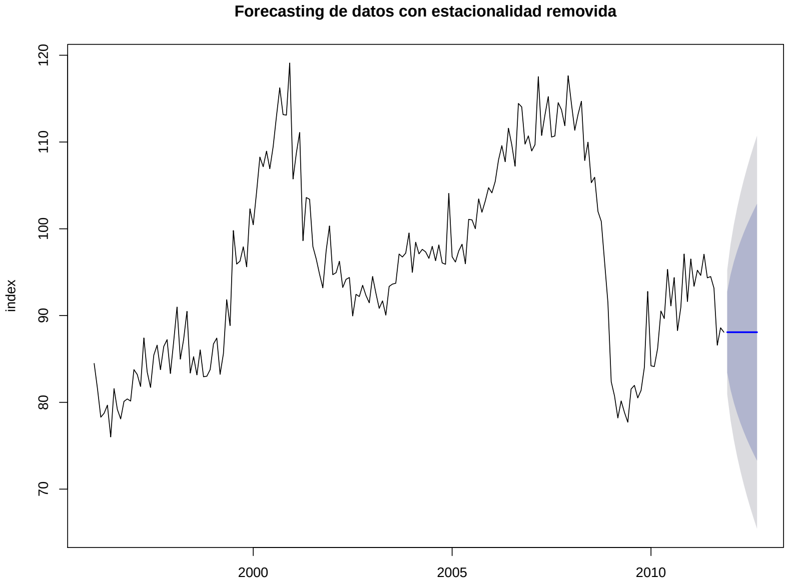
Forecasting STL



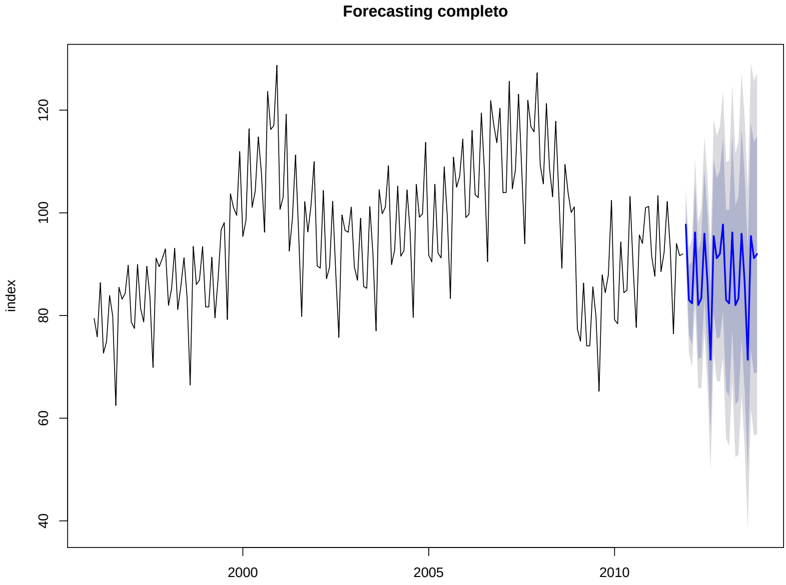
Forecasting STL



Forecasting STL



Forecasting STL



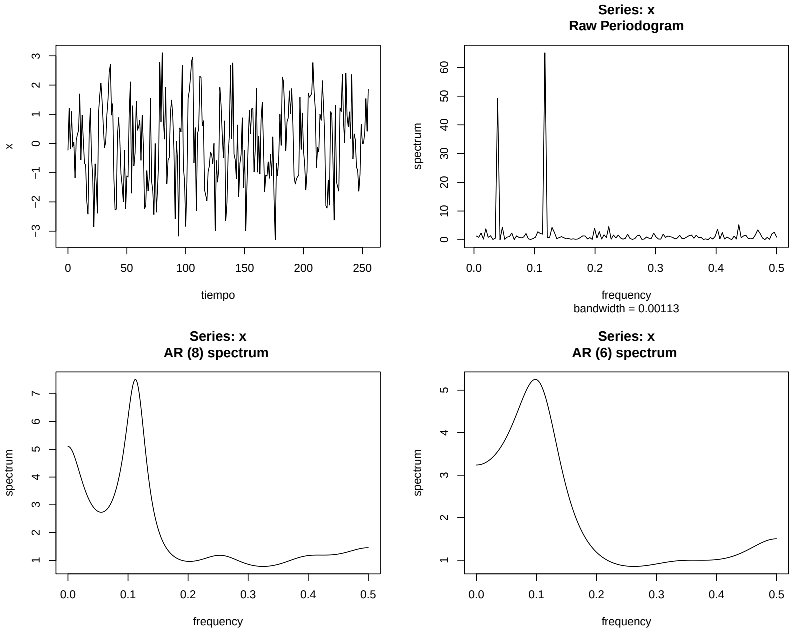
Modelos ARMA

- Modeliza una ST como la respuesta al impulso de un filtro con componentes autoregresivas (AR) y/o de media móvil (MA).
- Tres variantes:
  - puro AR, orden **p**
  - puro MA, orden **q**
  - combinado ARMA, orden **p** y orden **q**
- Ecuación ARMA: $y_n = ar_1 y_{n-1} + ar_2 y_{n-2} + ar_3 y_{n-3} + ... + ar_p y_{n-p} + ma_1 x_n + ma_2 x_{n-1} + ma_3 x_{n-3} + ... + ma_q x_{n-(q-1)}$

FFT vs modelos AR

```
N = 256
tiempo = 0:(N-1)

x <- sin(10*2*pi*tiempo/N) + sin(30*2*pi*tiempo/N) +
    rnorm(N)

par(mfrow=c(2,2))
plot(tiempo,x,type='l')
x.fft <- spec.pgram(x,plot=TRUE,taper = 0, log ='no')
x.ar8 <- spec.ar(x,plot=TRUE,log ='no',order=8)
x.ar6 <- spec.ar(x,plot=TRUE,log ='no',order=6)
```

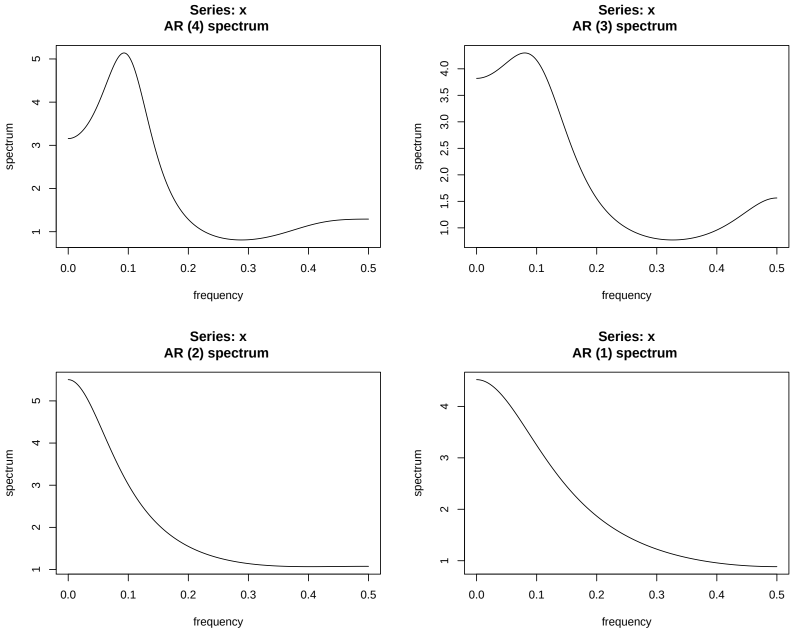
FFT vs modelos AR



FFT vs modelos AR

```
par(mfrow=c(2,2))
x.ar4 <- spec.ar(x,plot=TRUE,log ='no',order=4)
x.ar3 <- spec.ar(x,plot=TRUE,log ='no',order=3)
x.ar2 <- spec.ar(x,plot=TRUE,log ='no',order=2)
x.ar1 <- spec.ar(x,plot=TRUE,log ='no',order=1)
```

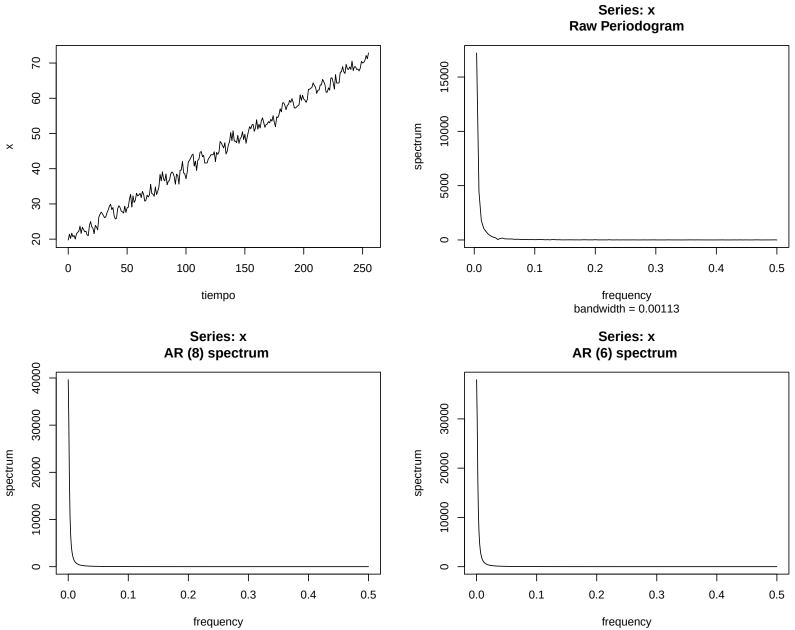
FFT vs modelos AR



FFT vs modelos AR, sumamos una tendencia

```
m = 0.1
b0 = 10
x <- x + m*tiempo + b0

par(mfrow=c(2,2))
plot(tiempo,x,type='l')
x.fft <- spec.pgram(x,plot=TRUE,taper = 0, log ='no',detrend
    = FALSE)
x.ar8 <- spec.ar(x,plot=TRUE,log ='no',order=8)
x.ar6 <- spec.ar(x,plot=TRUE,log ='no',order=6)
```

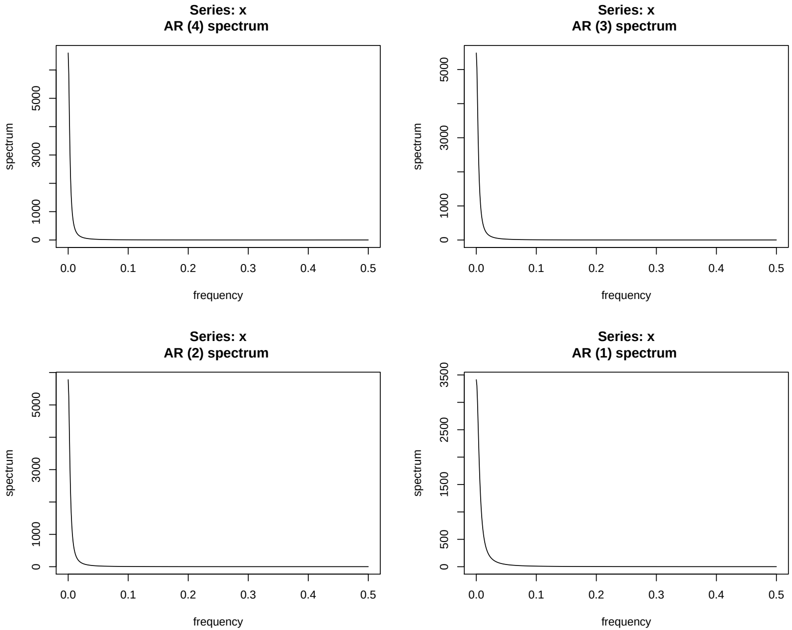
FFT vs modelos AR, sumamos una tendencia



FFT vs modelos AR, sumamos una tendencia

```
par(mfrow=c(2,2))
x.ar4 <- spec.ar(x,plot=TRUE,log ='no',order=4)
x.ar3 <- spec.ar(x,plot=TRUE,log ='no',order=3)
x.ar2 <- spec.ar(x,plot=TRUE,log ='no',order=2)
x.ar1 <- spec.ar(x,plot=TRUE,log ='no',order=1)
```

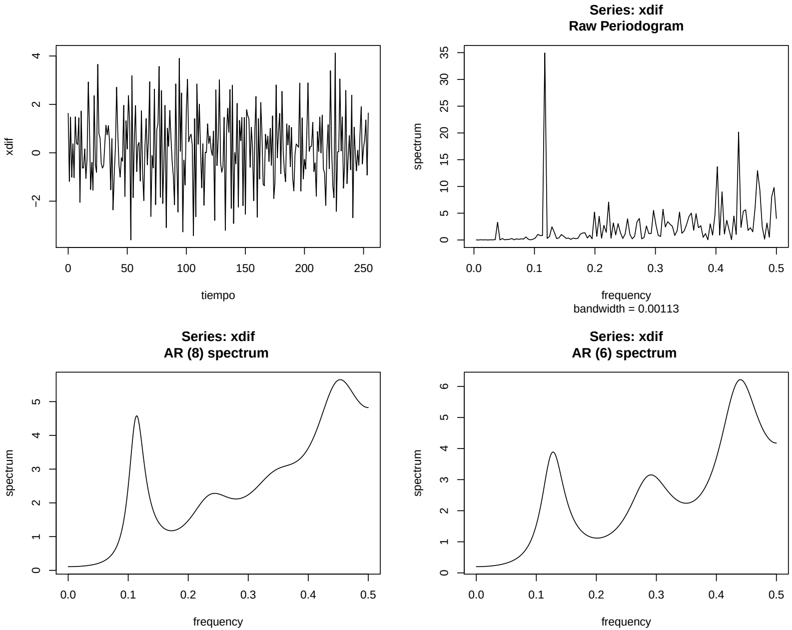
FFT vs modelos AR, sumamos una tendencia



FFT vs modelos AR, sumamos una tendencia y uso diff()

```
xdif <- diff(x)
length(tiempo) <- length(xdif)

par(mfrow=c(2,2))
plot(tiempo,xdif,type='l')
x.fft <- spec.pgram(xdif,plot=TRUE,taper = 0, log ='no',
    detrend = FALSE)
x.ar8 <- spec.ar(xdif,plot=TRUE,log ='no',order=8)
x.ar6 <- spec.ar(xdif,plot=TRUE,log ='no',order=6)
```

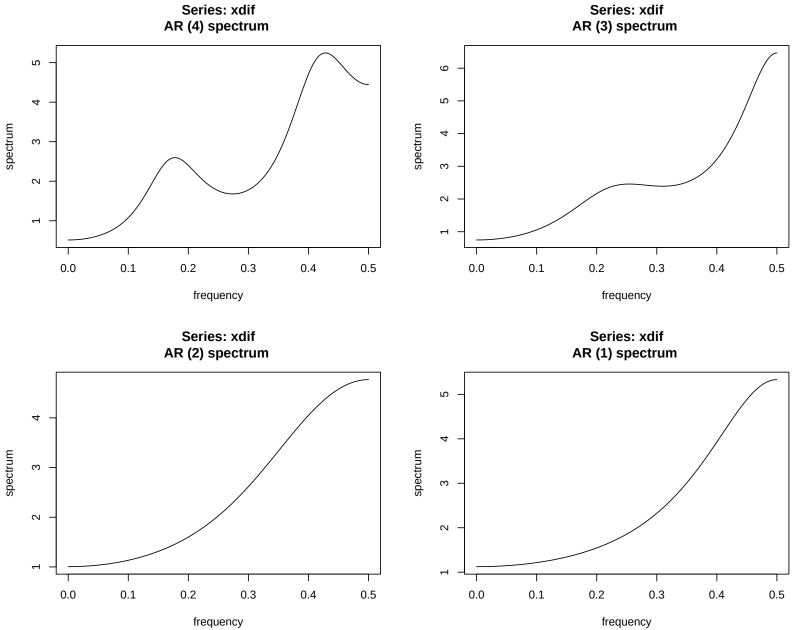
FFT vs modelos AR, sumamos una tendencia y uso diff()



FFT vs modelos AR, sumamos una tendencia y uso diff()

```
par(mfrow=c(2,2))
x.ar4 <- spec.ar(xdif, plot=TRUE,log ='no',order=4)
x.ar3 <- spec.ar(xdif, plot=TRUE,log ='no',order=3)
x.ar2 <- spec.ar(xdif, plot=TRUE,log ='no',order=2)
x.ar1 <- spec.ar(xdif, plot=TRUE,log ='no',order=1)
```

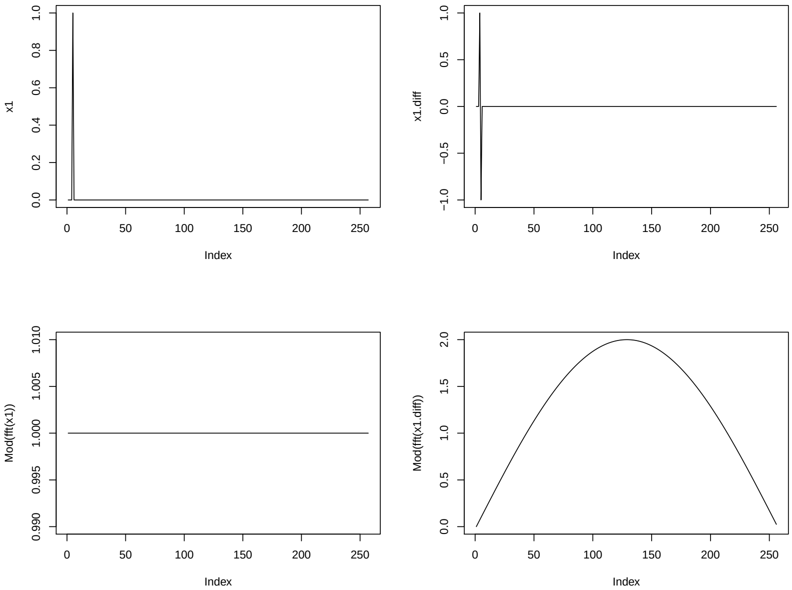
FFT vs modelos AR, sumamos una tendencia y uso diff()



Efecto del uso de diff()

```
# efecto de  diff
x1 <- rep(0,N+1)
x1[5]  <- 1
x1.diff  <- diff(x1)

par(mfrow=c(2,2))
plot(x1,type='l')
plot(x1.diff,type='l')
plot(Mod(fft(x1)),type='l')
plot(Mod(fft(x1.diff)),type='l')
```

Efecto del uso de diff()



Forecasting ARIMA

- Modeliza una ST con un ARMA (cualquiera de las tres variantes) y opción de diferenciar los datos antes, bajo la forma $ARIMA(p, d, q)$, donde p es el orden AR, d el orden de diferenciación (diff), y q el orden MA.
- Opciones:
  - puro AR: $ARIMA(p, 0, 0)$
  - puro MA: $ARIMA(0, 0, q)$
  - combinado ARMA: $ARIMA(p, 0, q)$
  - combinado ARMA con un nivel de diff: $ARIMA(p, 1, q)$

Forecasting ARIMA, determinación de los órdenes p y q

Determinación de los órdenes p y q en base a la **ACF** y la **PACF** de la ST:

	$AR(p)$	$MA(q)$	$ARMA(p,q)$
ACF	disminuye gradualmente	corta en q	disminuye gradualmente
PACF	corta en p	disminuye gradualmente	disminuye gradualmente

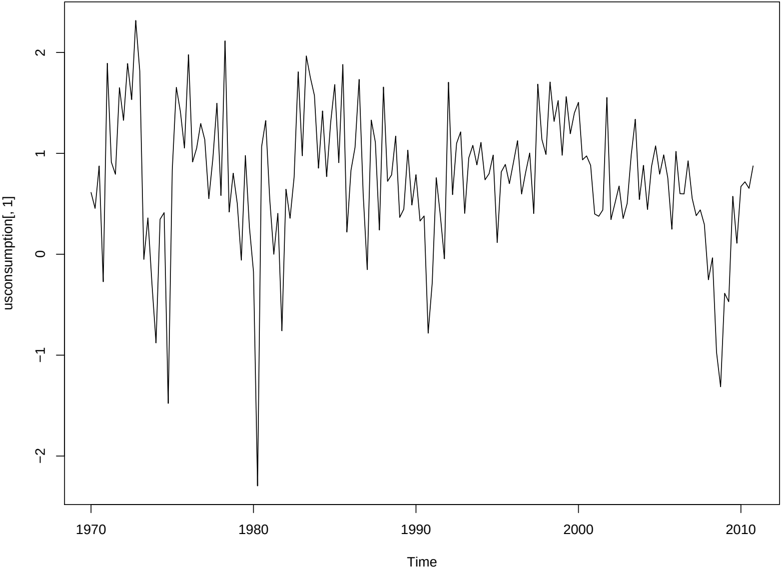
Forecasting ARIMA, determinación de los órdenes p y q

- Determinación del modelo de acuerdo a medidas relativas, en todos los casos se opta por el modelo con la menor medida relativa:
  - **Criterio de información de Akaike (AIC)**
  - **AIC corregido (AICc)**
  - **Criterio de información Bayesiano (BIC)**

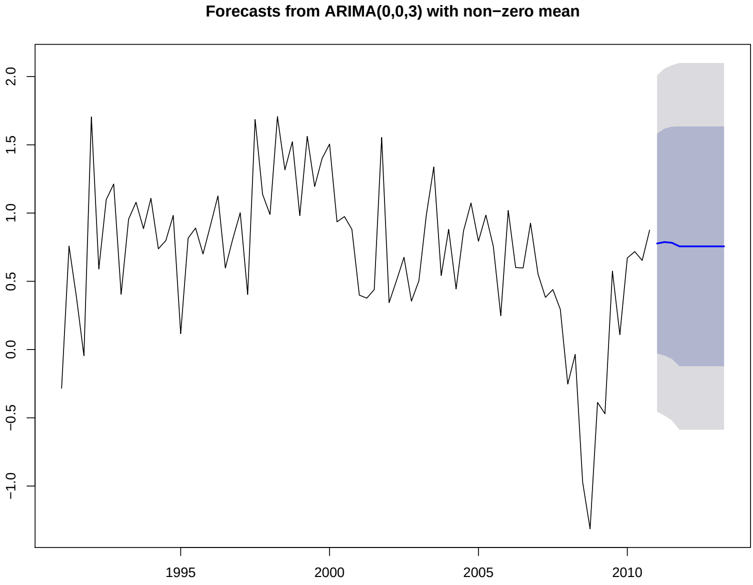
Forecasting ARIMA

```
data(usconsumption) # viene en el fpp
plot(usconsumption[,1])

fit.arima <- auto.arima(usconsumption[,1],seasonal=FALSE)
print(fit.arima)
# Series: usconsumption[, 1]
# ARIMA(0,0,3) with non-zero mean
#
# Coefficients:
#            ma1     ma2     ma3  intercept
#         0.2542  0.2260  0.2695     0.7562
# s.e.    0.0767  0.0779  0.0692     0.0844
#
# sigma^2 estimated as 0.3953:  log likelihood=-154.73
# AIC=319.46  AICc=319.84  BIC=334.96
plot(forecast(fit.arima,h=10),include=80)
```

Forecasting ARIMA



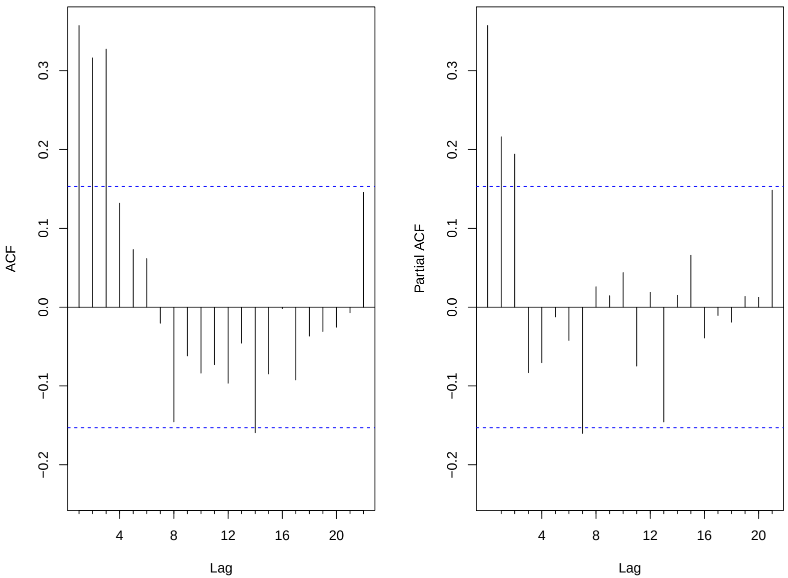
Forecasting ARIMA



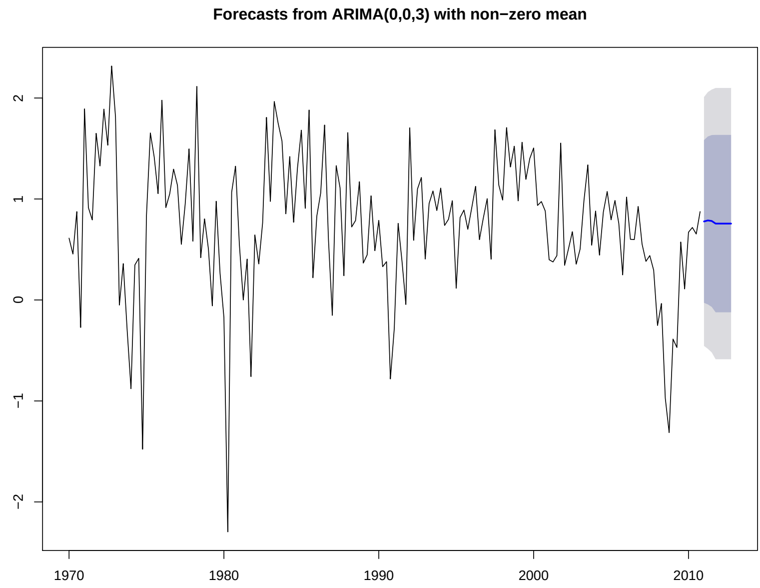
Forecasting ARIMA

```
par(mfrow=c(1,2))
Acf(usconsumption[,1], main="")
Pacf(usconsumption[,1], main="")
fit.arima2 <- Arima(usconsumption[,1], order=c(0,0,3))
print(fit.arima2)
# Series: usconsumption[, 1]
# ARIMA(0,0,3) with non-zero mean
#
# Coefficients:
#          ma1     ma2     ma3  intercept
#       0.2542  0.2260  0.2695     0.7562
# s.e.  0.0767  0.0779  0.0692     0.0844
#
# sigma^2 estimated as 0.3953:  log likelihood=-154.73
# AIC=319.46  AICc=319.84  BIC=334.96
#
plot(forecast(fit.arima2))
```

Forecasting ARIMA



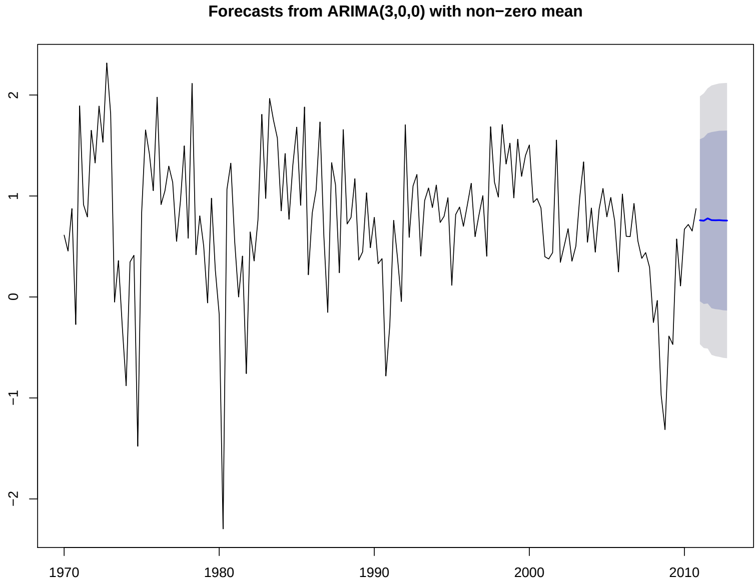
Forecasting ARIMA



Forecasting ARIMA

```r
fit.arima3 <- Arima(usconsumption[,1], order=c(3,0,0))
print(fit.arima3)
# Series: usconsumption[, 1]
# ARIMA(3,0,0) with non-zero mean
#
# Coefficients:
#          ar1     ar2     ar3  intercept
#       0.2366  0.1603  0.1909     0.7533
# s.e.  0.0763  0.0774  0.0759     0.1153
#
# sigma^2 estimated as 0.3921:  log likelihood=-154.08
# AIC=318.16  AICc=318.54  BIC=333.66
#
plot(forecast(fit.arima3))
```

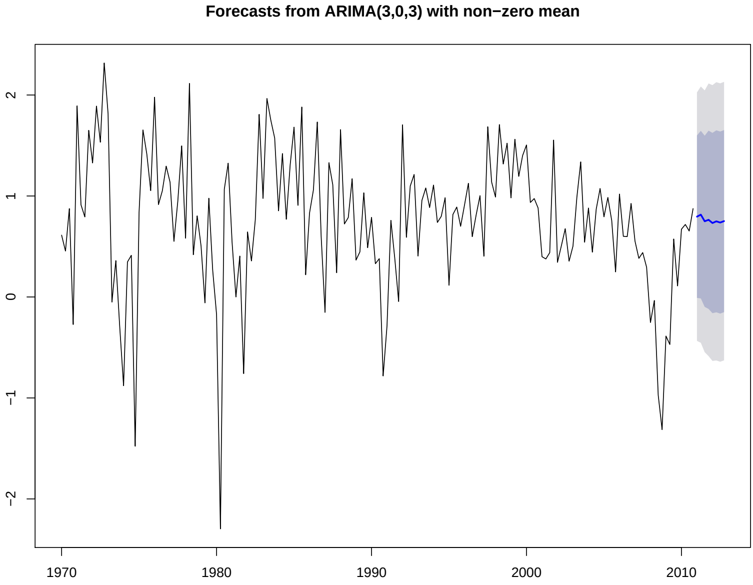
Forecasting ARIMA



Forecasting ARIMA

```
fit.arima4 <- Arima(usconsumption[,1], order=c(3,0,3))
print(fit.arima4)
# Series: usconsumption[, 1]
# ARIMA(3,0,3) with non-zero mean
#
# Coefficients:
#          ar1     ar2      ar3      ma1      ma2     ma3
    intercept
#       0.5487  0.4810  -0.4132  -0.2950  -0.4055  0.4796
   0.7545
# s.e.  0.3340  0.2159   0.2513   0.3181   0.2074  0.1650
   0.0968
#
# sigma^2 estimated as 0.3939:  log likelihood=-152.96
# AIC=321.92  AICc=322.84  BIC=346.71
#
plot(forecast(fit.arima4))
```

Forecasting ARIMA



Forecasting ARIMA

```
fit.arima5 <- Arima(usconsumption[,1], order=c(3,1,3))
print(fit.arima5)
# Series: usconsumption[, 1]
# ARIMA(3,1,3)
#
# Coefficients:
#           ar1     ar2     ar3      ma1      ma2     ma3
#       -0.2747  0.4921  0.2266  -0.4611  -0.7180  0.1791
# s.e.   0.2610  0.2169  0.1046   0.2650   0.3289  0.2025
#
# sigma^2 estimated as 0.3996:  log likelihood=-155.33
# AIC=324.66  AICc=325.39  BIC=346.32#
#
plot(forecast(fit.arima5))
```

Forecasting ARIMA