CAPÍTULO 5

Descomposición, Espacio de Estados y Redes neuronales

Índice

Este capítulo introduce varios modelos alternos a los paramétricos para la tendencia y la estacionalidad. Son modelos no paramétricos. Las estimaciones se basan en algoritmos aplicados a los datos, sin asumir un modelo paramétrico fijo. Una característica de los modelos no paramétricos es que son mucho más flexibles para modelar la tendencia y la estacionalidad porque las estimaciones se basan en una ventana (intervalo) de datos móvil. Por tanto, cabe denominarlos estimadores locales de tendencia y estacionalidad.

5.1. Métodos de descomposición

Página web: discusión extensa de descomposición
`http://course1.winona.edu/bdeppa/FIN20335/Handouts/Time_Series_Decomposition.html`

Varias librerías y funciones en R realizan la descomposición de T_t, S_t, ε_t, por ejemplo,

1. El algoritmo STL, con base en regresión Loess, incluye pronósticos.
2. El método Prophet de Facebook, en R. Incluye pronósticos.
3. El método híbrido Loess mas estacionalidad es una propuesta del curso, sin referencias. Incluye pronósticos.
4. La función `decompose()`.
5. La librería `timsac` tiene la función `decomp()` que realiza la descomposición incluyendo una componente autoregresiva y otra para fechas de intervenciones, $Y_t = T_t + S_t + R_t + TA_t + \varepsilon_t$.
6. La función `ma.filter()` de la librería `rmaf` (Refined Moving Average Filter, Qiu [2015]).

5.1.1. Regresión Local Loess

La Regresión Loess, denominada en R inicialmente Lowess por sus siglas en inglés: *Locally weighted scatterplot smoothing*, es un modelo de regresión no paramétrica

que regresa y_i versus x_i, pero no asume un modelo global fijo, es decir, no asume un intercepto y una pendiente fijas, sino variables, de manera local. Local significa una ventana móvil que contiene un número determinado de datos de la variable explicativa. La motivación para utilizar Loess es que se puede estimar la tendencia de la serie T_t sin utilizar un modelo global paramétrico, que en muchos casos resulta muy restrictivo.

La descripción del algoritmo de estimación Loess es el siguiente. Suponga datos bivariados: $(x_i, y_i),\ i = 1, \ldots, n$. El objetivo de Loess es calcular una función de regresión local $g(x)$ de forma que,

$$y_i = g(x_i) + \varepsilon_i, \quad i = 1, \ldots, n. \tag{5.1}$$

con $g(x_i)$ el análogo de $a + bx_i$ en $y_i = a + bx_i + \varepsilon_i$. Suponga $x_{(1)} \leq x \leq x_{(n)}$, entonces $g(x)$ se calcula así:

1. Se escoge q con $1 \leq q \leq n$.
2. Se escogen los q valores x_i más cercanos a x.
3. Defina $\lambda_q(x)$ la distancia máxima entre x y los q x_i escogidos. Luego
$$0 \leq \frac{|x_i - x|}{\lambda_q(x)} \leq 1.$$
4. Defina $w(x) = \begin{cases} (1 - x^3)^3 & ,\ 0 \leq x \leq 1 \\ 0 & ,\ x > 1 \end{cases}$
5. Defina $w_i(x) = w\left(\dfrac{|x_i - x|}{\lambda_q(x)}\right)$, para cada x_i escogido.
6. Ajuste $y_i = a + bx_i$ ó $y_i = a + bx_i + cx_i^2$ con MCO, tomando solamente los q valores más cercanos a x, ponderando cada x_i con $w_i(x)$.
7. Defina $g(x)$ como el valor estimado en x mediante la regresión local anterior.

En series de tiempo se toma, (t_i, y_i), con $x_i = t_i$. En cada t se estima la tendencia T_t con $\widehat{T}_t = \widehat{g}(t)$. En R se puede implementar la regresión Loess mediante la función `loess()`.

```
T = length(y)
t = seq(1,T)
yw = loess(y ~ t, span = 0.75,degree = 1,
control = loess.control(surface = "direct"))
```

El parámetro `span = alfa` es un número $\alpha \in (0, 1)$, que determina el ancho de la ventana en términos de un porcentaje de datos, $q = \lfloor \alpha n \rfloor$. Mientras más cercano a 1 más suaviza porque se asemeja a una regresión lineal ó cuadrática global.

El parámetro `degree = 1,2` determina si la regresión local es lineal ó cuadrática.

Una característica importante y útil de la función `predict()` es que admite objetos generados con `loess()` para calcular pronósticos. En el Ejemplo siguiente se muestra cómo implementar estos procedimientos.

Ejemplo 5.1.1. *Se consideran los datos del DANE sobre estadísticas de edificación, correspondientes a número de unidades aprobadas para vivienda, mensuales, en 88 municipios en el período 2009/01 - 2017/06.*

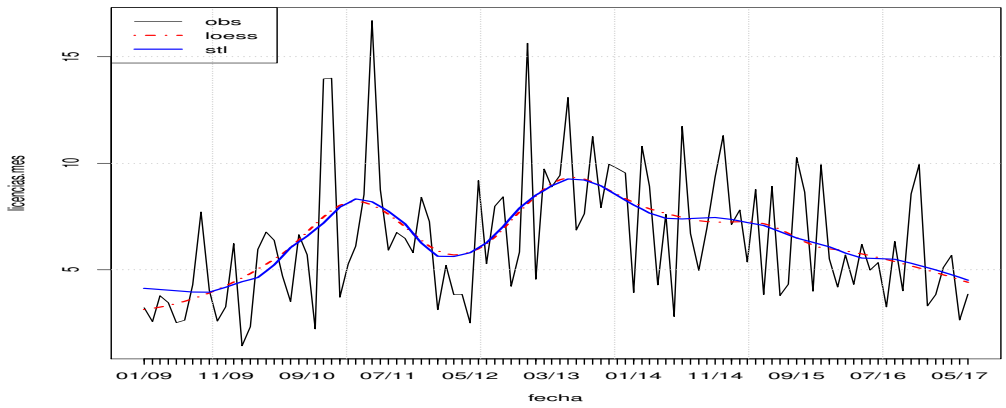

Figura 5.1: Suavizamiento Loess de número de unidades aprobadas para vivienda, mensuales, en 88 municipios en el período 2009/01 - 2017/06

Prueba Mann-Kendall para detectar tendencia monótona

Si se tiene una serie de tiempo $\{Y_t, t = 1, 2, \ldots, T\}$ se define la prueba de Mann-Kendall para las hipótesis nula H_0 : los datos Y_t son iid, versus la alterna H_1 : los

datos tienen una tendencia monótona. El estadístico Mann-Kendall se define como

$$S = \sum_{k=1}^{T-1} \sum_{j=k+1}^{T} signo(Y_j - Y_k). \tag{5.2}$$

donde $signo(x) = \pm 1$ si $x \gtrless 0$, y $signo(0) = 0$. Bajo H_0, S se distribuye aproximadamente Normal, con media cero. Un valor positivo de S indica una tendencia creciente, y uno negativo una tendencia decreciente. La prueba está implementada en la librería `kendall`.

Este es el código en R para suavizamiento con Loess, ver Figura 5.3.

Código R 5.1.1.

```
# Analisis de número de unidades aprobadas para vivienda
# 88 municipios 2009/01 - 2017/06
# leer
G = read.table("series_histo_88_mpio_jun17.prn",
header = TRUE, stringsAsFactors = FALSE)
attach(G)

# definir como serie de tiempo
y = ts(G$Totalviv,frequency=12,start=c(1986,1))
ts.plot(y)

# Puede asumirse una tendencia global?
# detectar tendencia monotona con la prueba Mann-Kendall
# Ho: no hay tendencia, Ha: tendencia monotona
library(Kendall)
 MannKendall(y)
tau = 0.179, 2-sided pvalue =0.0078129
# como valor p menor de 0.05 rechaza la nula
#---------------utilizar loess para estimar tendencia
m = 12
n = length(y)
yi = ts(y[1:(n-m)],frequency=12)
yf = ts(y[(n-m+1):n],frequency=12)
```

```
t = seq(1,(n-m))
yw = loess(yi ~ t, span=0.5,
control = loess.control(surface = "direct"))
yest.yw = yw$fitted
```

5.1.2. Método de descomposición STL

La descomposición STL es un procedimiento para estimar las tres componentes del modelo $Y_t = T_t + S_t + \varepsilon_t$, desarrollado por Cleveland et al. [1990], con base en Regresión Loess. Consiste de una secuencia de dos aplicaciones iteradas de Regresión Loess. Para aplicar este método se debe especificar el período (frequency=s) de la componente estacional, mediante la función `ts(x,frequency=s)`, donde, por ejemplo, $s = 4, 12, 52, 360$.

La estimación de la componente estacional $\widehat{S}_t$ por este método no es equivalente a la que se obtiene mediante los modelos paramétricos de componentes estacionales ó trigonométricas.

Además, este método permite calcular pronósticos con la función `forecast` de la líbrería del mismo nombre. Se aplicará para obtener las componentes de la serie Y_t y realizar un pronóstico.

El método está implementado en la función de R `slt()`. Una re-implementación del algoritmo `stl` en C++ y R está en `stl.plus`.

Ejemplo 5.1.2. *Al aplicar el método STL a los datos del Ejemplo (5.1.1) se obtiene la descomposición en la Figura (5.2). El tercer panel es la tendencia estimada por Loess. El segundo la componente estacional. El tercero es la componente del error. La componente estacional tiene un rango estimado* $[-2, 6] + 6.8 = [4.8, 12.8]$*, (se añade* $min(Y_t) = 6.83$ *para colocar la serie a nivel de la serie* Y_t*), y la* Y_t *un rango* $[6.8, 31]$*. La componente estacional tiene un rango que abarca una parte importante del rango de la serie* Y_t*.*

```
#---------descomposicion  y pronostico con stl
```

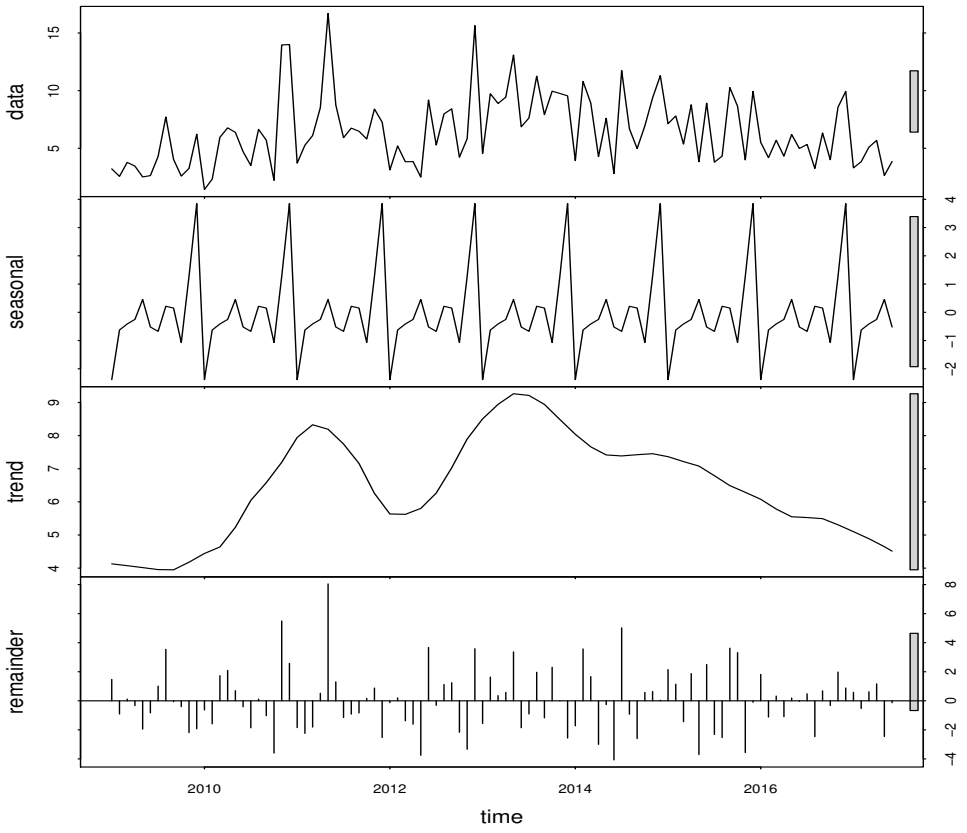

Figura 5.2: Descomposición STL número de unidades aprobadas para vivienda, mensuales, en 88 municipios en el período 2009/01 - 2017/06

```
y.stl = stl(yi,"per")
plot(y.stl)
St.stl = y.stl$time.series[,1]
Tt.stl = y.stl$time.series[,2]
#----------pronostico
pr.stl = forecast(y.stl,method="ets",h=m)$mean
#---------------grafica pronosticos
plot(tt,yf,type='b',lwd=1,ylim=c(8,21))
lines(tt,pr.y,col='red',lwd=1)
lines(tt,pr.stl,col='blue',lwd=1)
legend("topleft", c("Obs","hibrido:Loess+Ind","STL"),
```

```
pch = c(1, 3, 5),col = c("black","red","blue"))
#--------------compara calidad pronosticos
pr.y = ts(pr.y,frequency=12,start=c(2016,06))
pr.stl = ts(pr.stl,frequency=12,start=c(2016,06))
yf = ts(yf,frequency=12,start=c(2016,06))
M = rbind(accuracy(pr.y,yf),accuracy(pr.stl,yf))
rownames(M) = c("hibrido:Loess+Ind","STL")
(M)
```

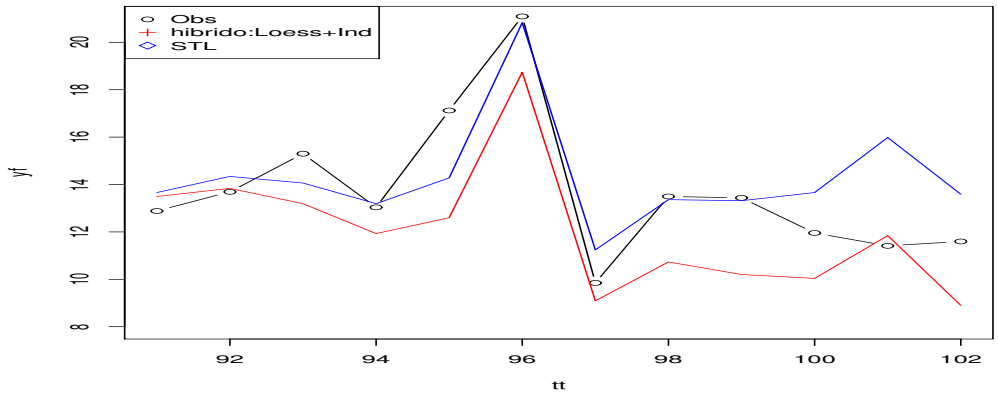

Figura 5.3: Pronóstico modelo híbrido Loess + Indicadoreas, serie de número de unidades aprobadas para vivienda, mensuales, en 88 municipios en el período 2009/01 - 2017/06

Tabla 5.1: Calidad de pronósticos para los modelos Loess+Estacional versus STL

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
hibrido:Loess+Ind	1.69	2.27	1.89	11.82	13.41	0.04	0.78
STL	-0.55	1.84	1.32	-5.82	10.46	0.37	0.64

5.1.3. Prophet

En el sitio web ([1]): "Facebook Prophet es una herramienta desarrollada por Facebook para pronosticar objetos o datos de series temporales. Ayuda a las empresas a conocer

[1]https://www.geeksforgeeks.org/time-series-analysis-using-facebook-prophet-in-r-programming

el comportamiento de sus productos al pronosticar precios, ventas o el clima".

De la ayuda: "Implementa un procedimiento para pronosticar datos de series de tiempo basado en un modelo aditivo en el que las tendencias no lineales se ajustan con datos anuales, semanales y estacionalidad diaria, más efectos festivos. Funciona mejor con series de tiempo que tienen fuertes efectos estacionales y varias estaciones completas de datos históricos. Prophet es robusto a los datos faltantes y a los cambios en la tendencia y, por lo general, maneja bien los valores atípicos".

El modelo lo introdujeron Taylor and Letham [2018], y utiliza un modelo aditivo generalizado ó GAM, con tres componentes: tendencia, estacional y efecto calendario dado por días especiales. Se combinan en un modelo similar al aditivo de componentes:

$$Y_t = T_t + S_t + h_t + \epsilon_t \tag{5.3}$$

donde T_t es una función seccionalmente lineal, o también una curva logística, para modelar la tendencia media.

La componente estacional es S_t, y h_t es el efecto calendario que consiste en fechas irregulares correspondientes a fiestas, etc. Y ϵ_t es el residuo aleatorio.

5.1.4. Método híbrido Loess y componente estacional

En el modelo de componentes con variables indicadoras (3.16), pág 68,

$$Y_t = \beta_0 + \sum_{j=1}^{k} \beta_j t^j + \sum_{j=1}^{s-1} \delta_j I_j(t) + \varepsilon_t,$$

la tendencia y la estacionalidad son modelos paramétricos globales, lo que quiere decir que sus parámetros no cambian con t. Para introducir una mayor flexibilidad se propone estimar la tendencia mediante Loess, y conservar la parte paramétrica de la componente estacional. Se tendría un modelo semi-paramétrico, denominado híbrido, de la forma

$$Y_t = g(t) + \sum_{j=1}^{s} \delta_j I_j(t) + \varepsilon_t, \tag{5.4}$$

donde $g(t)$ es la estimación con Loess de la tendencia que reemplaza a $\beta_0+\sum_{j=1}^{k}\beta_j t^j$. Nótese que se puede estimar el conjunto total de coeficientes estacionales δ_j, porque no hay, estrictamente, un problema de multicolinealidad.

Debido a que el modelo híbrido (5.4) es un modelo aditivo, el procedimiento de estimación se puede realizar en dos pasos. Estimar por Loess la tendencia, obteniendo $\widehat{T}_t = \widehat{g}(t)$, y luego estimar la componente estacional y el error estructural mediante regresión lineal con respecto a la variable $Y_t - \widehat{T}_t$. El código R 5.1.2 siguiente muestra cómo realizar el procedimiento. En la Figura 5.3 se muestran los resultados.

Código R 5.1.2.

```
#-------------pronostico modelo híbrido
#             pronostico con Loess
tt = seq((n-m+1),n,1)
pr.yw = predict(yw, data.frame(t = tt))
#-------------pronostico de la parte estacional
dummy=function(s,n,i){
# s: estaciones, n: numero de datos, i: estación de inicio
A=diag(s)
for(j in 1:(floor(n/s))){
A=rbind(A,diag(s))}
A=A[i:(n+i-1),]
return(A)}

Itt = dummy(12,m,7)
pr.st = predict(mod1,data.frame(It=I(Itt)))
pr.y = pr.yw+pr.st
```

5.2. Modelos de Espacio de Estados

La idea de los modelos de Espacio de Estados es que son modelos con coeficientes variables. En el modelo de componentes con variables indicadoras (3.16), pág 68,

$$Y_t = \beta_0 + \sum_{j=1}^{k}\beta_j t^j + \sum_{j=1}^{s}\delta_j I_j(t) + \varepsilon_t,$$

la tendencia y la estacionalidad son modelos paramétricos globales, lo que quiere decir que sus parámetros no cambian con t. Para introducir una mayor flexibilidad se pueden permitir coeficientes variables con t. Por ejemplo, tomando $k = 1$ en el modelo anterior, y haciendo depender cada parámetro del tiempo t, se tendría un modelo que se denomina de coeficientes variables:

$$Y_t = \beta_{0,t} + \beta_{1,t}t + \sum_{j=1}^{s} \delta_{j,t} I_j(t) + \varepsilon_t, \tag{5.5}$$

Pero para definir el modelo de espacio de estados se usan dos ecuaciones. Si $\underline{\mu}_t$ es un vector que contiene los coeficientes variando con el tiempo, se asume que cumplen una ecuación recursiva, la ecuación de estados, de la forma

$$\underline{\mu}_t = T_t \, \underline{\mu}_{t-1} + \underline{d}_t + R_t \, \underline{\eta}_t, \tag{5.6}$$

Y otra ecuación en donde se relaciona la serie observada con los estados, la ecuación observada, en forma vectorial

$$\underline{Y}_t = Z_t \, \underline{\mu}_t + \underline{c}_t + \underline{\epsilon}_t, \tag{5.7}$$

Finalmente, se indica cómo se inicia la ecuación de los estados

$$\underline{\mu}_0 \sim N_m(\underline{\mu}_0, P_0). \tag{5.8}$$

La formulación es vectorial, pero aplica a casos escalares, ver [Helske, 2017, pag. 2]. Las variables que intervienen en el modelo son

- $\underline{Y}_t \in \mathbb{R}^p$, $\underline{\mu_t} \in \mathbb{R}^m$, $\underline{\epsilon_t} \in \mathbb{R}^p$, $\underline{\eta_t} \in \mathbb{R}^q$
- Las matrices $Z_t \in \mathbb{R}^{p \times m}$, $T_t \in \mathbb{R}^{m \times m}$, $R_t \in \mathbb{R}^{m \times q}$ pueden ser independientes de t.
- $P_0 \in \mathbb{R}^{m \times m}$ es una matriz de varianza-covarianza.
- Los errores $\underline{\epsilon_t} \sim iidN_p(0, H)$, $\eta_t \sim iidN_q(0, Q)$, se asumen Normales multivariados, independientes entre ellos.
- Las matrices $H \in \mathbb{R}^{p \times p}$, $Q \in \mathbb{R}^{q \times q}$ son matrices de varianzas-covarianzas.

Existen varias librerías en R para estimar modelos de Espacio de Estados en su forma general, como la librería `dlm`. Ejemplos de modelos de espacio de estados en ([2])

El modelo anterior es general. Se exponen varios casos particulares de las ecuaciones latente y observada, (5.7) y (5.6).

5.2.1. Modelo de espacio de estados Holt-Winters

El modelo Holt-Winters fué introducido por Winters [1960]. Discusiones adicionales sobre diferentes formas de las recursiones aditiva y multiplicativas Holt-Winters se pueden encontrar en Ord et al. [1997], Harvey [2006, pag. 358]

El modelo Holt-Winters se puede ver como un algoritmo para estimar recursivamente los coeficientes variables en

$$Y_t = \beta_{0,t} + \beta_{1,t}t + \sum_{j=1}^{s} \delta_{j,t} I_j(t) + \varepsilon_t,$$

Llamando μ_t es el nivel (`level`), y β_t el gradiente de la tendencia lineal, se definen como

$$\begin{aligned} \mu_t &= \beta_{0,t}, \\ \beta_t &= \beta_{1,t}t, \\ S_t &= \sum_{j=1}^{s} \delta_{j,t} I_j(t), \end{aligned}$$

a partir de unos valores iniciales $\mu_0, \beta_0, S_{0,j} = \delta_{j,0}, j = 1, 2, \ldots, s$.

La versión de espacio de estados Holt-Winters consiste en el sistema de ecuaciones recursivas para μ_t, β_t y S_t, ver la exposición de R.J.Hyndman en ([3])

$$Y_t = \mu_{t-1} + \beta_{t-1} + S_{t-s} + \varepsilon_t, \tag{5.9a}$$

$$\mu_t = \mu_{t-1} + \beta_{t-1} + \alpha\varepsilon_t, \tag{5.9b}$$

$$\beta_t = \beta_{t-1} + \beta\varepsilon_t, \tag{5.9c}$$

$$S_t = S_{t-s} + \gamma\varepsilon_t,\ t = 2, \ldots, T, \tag{5.9d}$$

[2] https://github.com/pteetor/StateSpaceModels.

[3] https://robjhyndman.com/talks/ABS1.pdf

donde $\epsilon_t \sim iidN(0, \sigma_\epsilon^2)$. Los parámetros son μ_0, y las desviaciones $\sigma_\epsilon > 0$, $\alpha > 0$, $\beta > 0$, $\gamma > 0$. Es posible que algunas de éstas tengan estimaciones casi cero. Sin la componente estacional S_t y S_{t-s}, en el sistema anterior, se denomina el modelo de Holt.

El valor suavizado de la serie en $t = 1, 2, \ldots, T$ es

$$\widehat{Y}_t = Y_t = \mu_{t-1} + \beta_{t-1} + S_{t-s}.$$

El pronóstico para un horizonte $j = 1, 2, \ldots, m$

$$\widehat{Y}_{T+j} = \mu_T + j\beta_T + S_{T-s+j}$$

La programación en R del método Holt-Winters se puede hacer con la librería `forecast` mediante la función `hw`

```
#---------modelo Holt-Winters en forecast
mod1 = hw(yi)
summary(mod1)
(coef1 = mod1$model$par[1:3])
yi.hat1 = mod1$model$fitted
#---------
source("medidas.yest.r")
(A=medidas.yest(yi.hat1,yi,3))
#---------pronosticos
ypron = forecast(mod1,h=m)
```

Otra versión del modelo es BSM (*Basic Structural Model*), que se define como (ver la ayuda de la función `StructTS`)

$$Y_t = \mu_t + S_t + \epsilon_t, \tag{5.10a}$$

$$\mu_t = \mu_{t-1} + \beta_{t-1} + \alpha\varepsilon_t, \tag{5.10b}$$

$$\beta_t = \beta_{t-1} + \beta\varepsilon_t, \tag{5.10c}$$

$$S_t = S_{t-1} + \ldots + S_{t-s+1} + \gamma\varepsilon_t, \tag{5.10d}$$

Este modelo se puede estimar por máxima verosimilitud con la instrucción

```
mod.bsm <- StructTS(y, type = "BSM")
print(mod.bsm$coef)
```

El Modelo de Espacio de Estados, `ETS(A,A,A)` definido en Hyndman et al. [2008, pag. 45, sec. 3.4.3], es el modelo Holt-Winters (5.9).

La estimación de los parámetros se realiza mediante la función `ets()` de la librería `forecast`.

```
m1 = ets(y,model="AAA",damped=FALSE)
yhat1 = m1$fitted
#---- h pronosticos
yf.ets = forecast(m1,h)$mean
```

En la presentación mencionada de R. Hyndman se introduce una taxonomía de modelos de espacios de estados. Ver también Hyndman et al. [2002] y Hyndman et al. [2008].

5.2.2. Modelos de componentes Holt-Winters

Hay una variante alterna del modelo, denominada de componentes, ver ([4]), utilizada en varios textos.

$$Y_t = \mu_t + \beta_t + S_t \tag{5.11a}$$
$$\mu_t = \alpha(Y_t - S_{t-s}) + (1-\alpha)(\mu_{t-1} + \beta_{t-1}), \tag{5.11b}$$
$$\beta_t = \beta(\mu_t - \mu_{t-1}) + (1-\beta)\beta_{t-1}, \tag{5.11c}$$
$$S_t = \gamma(Y_t - \mu_t) + (1-\gamma)S_{t-s}, \tag{5.11d}$$

En el modelo (5.9) los parámetros son desviaciones estándar. En el modelo (5.11) son constantes α, β, γ, cada una en $(0, 1)$, y son los parámetros de suavizamiento (*tunnig parameters*). Se escogen de manera que se minimize el MSE. O también, por tanteo. Según Abraham and Ledolter [2000, pag. 169],

> "Si las componentes de tendencia y estacionalidad cambian rápidamente, las constantes deben tomarse cercanas a 1.0. Si las componentes son estables y varían lentamente, se escogen cercanas a 0.0".

[4] https://robjhyndman.com/talks/ABS1.pdf

Se puede realizar la estimación de este modelo con la función `HoltWinters` de la librería `stats`.

```
# ---------ejemplo programación de Holt-Winters
# y es un objeto 'ts', con período s
m = HoltWinters(y)
(c(m$alpha,m$beta,m$gamma))
# la tendencia, componente estacional y y estimada
Tt = m$fitted[,2] + m$fitted[,3]
St = m$fitted[,4]
Yt.hat = m$fitted[,1]
# y 12 pronosticos se calculan como
ypron = predict(m,12, prediction.interval = TRUE)
plot(m, p)
```

5.2.3. Modelo de nivel y tendencia local

El modelo de tendencia lineal local (*local linear trend model*), asume que se observa Y_t, pero la serie tiene un nivel local μ_t más una tendencia β_t. La tendencia es una marcha aleatoria.

$$Y_t = \mu_{t-1} + \beta_{t-1} + \varepsilon_t, \tag{5.12a}$$

$$\mu_t = \mu_{t-1} + \beta_{t-1} + \alpha\varepsilon_t, \tag{5.12b}$$

$$\beta_t = \beta_{t-1} + \beta\varepsilon_t, \tag{5.12c}$$

con $\epsilon_t \sim iidN(0, \sigma^2)$. Los parámetros son μ_0, $\alpha > 0$, $\beta > 0$ y σ^2. Este modelo se puede estimar con la instrucción

```
mod.t <- StructTS(y, type = "trend")
print(mod.t$coef)
```

Un procedimiento de estimación Bayesiana del modelo de tendencia local está en la librería `bsts`.

```
library(bsts)
```

```
ss <- AddLocalLinearTrend(list(),yt)
mod2.m <- bsts(yt,state.specification = ss,niter = 4000)
pred2 <- predict(mod2.m, horizon = 6)
print(mod2.m)
plot(mod2.m,"components")
summary(mod2.m, burn  = 80)

par(mfrow=c(2,2))
hist(mod2.m$sigma.trend.level,100)
points(mean(mod2.m$sigma.trend.level),0,
pch=20,col='red',cex=2.5)

hist(mod2.m$sigma.trend.slope,100)
points(mean(mod2.m$sigma.trend.slope),0,
pch=20,col='red',cex=2.5)

hist(mod2.m$sigma.obs,100)
points(mean(mod2.m$sigma.obs),0,
pch=20,col='red',cex=2.5)

(M=rbind(M,c(mean(mod2.m$sigma.trend.level),
mean(mod2.m$sigma.trend.slope)
,mean(mod2.m$sigma.obs))))
```

El modelo de tendencia local (*local level model*), asume un nivel local μ_t pero no tendencia. El nivel local es una marcha aleatoria.

$$Y_t = \mu_{t-1} + \varepsilon_t, \tag{5.13a}$$

$$\mu_t = \mu_{t-1} + \alpha\varepsilon_t, \tag{5.13b}$$

con $\epsilon_t \sim iidN(0, \sigma^2)$, μ_0, $\sigma > 0$, $\alpha > 0$. Este modelo se puede estimar con la instrucción

```
mod.loc <- StructTS(y, type = "level")
print(mod.loc$coef)
```

5.2.4. Modelo de tendencia local amortiguada

El modelo de tendencia lineal local amortiguada (*damped local linear level model*), es una modificación del método Holt, que consiste en introducir un parámetro autoregresivo de amortiguamiento $\theta \in (0, 1)$ en la componente del gradiente de la tendencia β_t,

labeldamped.loc.lin.level

$$Y_t = \mu_{t-1} + \theta\beta_{t-1} + S_{t-s} + \varepsilon_t, \tag{5.14a}$$
$$\mu_t = \mu_{t-1} + \theta\beta_{t-1} + \alpha\varepsilon_t, \tag{5.14b}$$
$$\beta_t = \theta\beta_{t-1} + \beta\varepsilon_t, \tag{5.14c}$$

con $\epsilon_t \sim iidN(0, \sigma^2)$, los parámetros son μ_0, $\sigma > 0$, $\alpha > 0$, $\beta > 0$.

Este modelo se puede estimar con la instrucción

5.2.5. Modelo de tendencia local y estacionalidad amortiguados

Según R. Hydnman: "Often the single most accurate forecasting method for seasonal data", en la presentación en ([5]). Es una modificación del modelo Holt-Winters (5.9), en donde se multiplica el gradiente de la tendencia β_t por un coeficiente autoregresivo $\theta \in (0, 1)$.

$$Y_t = \mu_{t-1} + \theta\beta_{t-1} + S_{t-s} + \varepsilon_t, \tag{5.15a}$$
$$\mu_t = \mu_{t-1} + \theta\beta_{t-1} + \alpha\varepsilon_t, \tag{5.15b}$$
$$\beta_t = \theta\beta_{t-1} + \beta\varepsilon_t, \tag{5.15c}$$
$$S_t = S_{t-s} + \gamma\varepsilon_t, \tag{5.15d}$$

Se puede implementar una estimación Bayesiana del modelo con la librería `bayesforecast`.

De la ayuda de la función `stan_ssm`:

"De forma predeterminada, la función ssm() genera un modelo de nivel local (o un ets(."A","N","N") o un modelo de suavizado exponencial del paquete de `forecast`).

[5] https://robjhyndman.com/talks/ABS1.pdf

Si la tendencia se establece como `trend=TRUE`, entonces se define un modelo ssm de tendencia local (un modelo equivalente ets(.ᴬ",.ᴬ","N") o Holt del paquete de `forecast`).

Para modelos de tendencia amortiguada, `damped=TRUE`.

Si estacional se establece en VERDADERO, se define un modelo de nivel local estacional (un modelo ets(.ᴬ","N",.ᴬ") equivalente del paquete de `forecast`).

Para un método de Holt-Winters (ets(.ᴬ",.ᴬ",.ᴬ")) establezca Tendencia y estacional en VERDADERO".

```
#---------------------------bayesforecast
# stan_Hw: Fitting a Holt-Winters state-space model
library(bayesforecast)
mod6.m  = stan_ssm(yi,trend = TRUE,
seasonal = TRUE,damped = TRUE,
iter = 500,chains = 1)
ypron6 = forecast(mod6.m,h=m)$mean
```

Ejemplo 5.2.1. *Con el Ejemplo 5.1.1, de los datos del DANE sobre estadísticas de edificación, correspondientes a número de unidades aprobadas para vivienda, mensuales, en 88 municipios en el período 2009/01 - 2017/06. Se ajustó los tres modelos estructurales a los datos. El modelo BSM resultó claramente superior tanto en calidad de ajuste como en calidad de pronósticos, como lo muestran las Tablas 5.4 y 5.5.*

Tabla 5.2: Medidas de calidad de ajuste, tres modelos estructurales

	R2-ajus	MSE	logAIC	logBIC
local	0.492	11.768	2.498	2.581
tendencia	0.484	11.946	2.524	2.635
bsm	0.725	6.378	1.907	2.046

Ejemplo 5.2.2. *Con el Ejemplo 5.1.1, de los datos del DANE sobre estadísticas de edificación, correspondientes a número de unidades aprobadas para vivienda, mensuales, en 88 municipios en el período 2009/01 - 2017/06. Se ajustó los tres modelos estructurales a los datos. El modelo BSM resultó claramente superior tanto*

Tabla 5.3: Medidas de calidad de pronósticos, tres modelos estructurales

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
local	-0.898	2.991	2.499	-10.526	18.767	0.035	0.850
tendencia	-0.751	2.913	2.392	-9.346	17.808	0.012	0.813
bsm	-0.746	2.303	1.728	-7.473	13.194	0.243	0.775

en calidad de ajuste como en calidad de pronósticos, como lo muestran las Tablas 5.4 y 5.5.

Tabla 5.4: Medidas de calidad de ajuste, tres modelos estructurales

	R2-ajus	MSE	logAIC	logBIC
local	0.492	11.768	2.498	2.581
tendencia	0.484	11.946	2.524	2.635
bsm	0.725	6.378	1.907	2.046

Tabla 5.5: Medidas de calidad de pronósticos, tres modelos estructurales

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
local	-0.898	2.991	2.499	-10.526	18.767	0.035	0.850
tendencia	-0.751	2.913	2.392	-9.346	17.808	0.012	0.813
bsm	-0.746	2.303	1.728	-7.473	13.194	0.243	0.775

Ejemplo 5.2.3. *Continuando con el Ejemplo 5.1.1, de los datos del DANE sobre estadísticas de edificación, correspondientes a número de unidades aprobadas para vivienda, mensuales, en 88 municipios en el período 2009/01 - 2017/06. Se puede observar una comparación entre Loess y Holt-Winters en la Figura (5.4)*

5.3. Introducción a Redes neuronales

5.3.1. Redes neuronales como modelos estadísticos

Hay varias discusiones sobre las redes neuronales y su relación con varios tipos de modelos estadísticos. En especial, la relación con regresión noparamétrica y con series de tiempo no lineales.

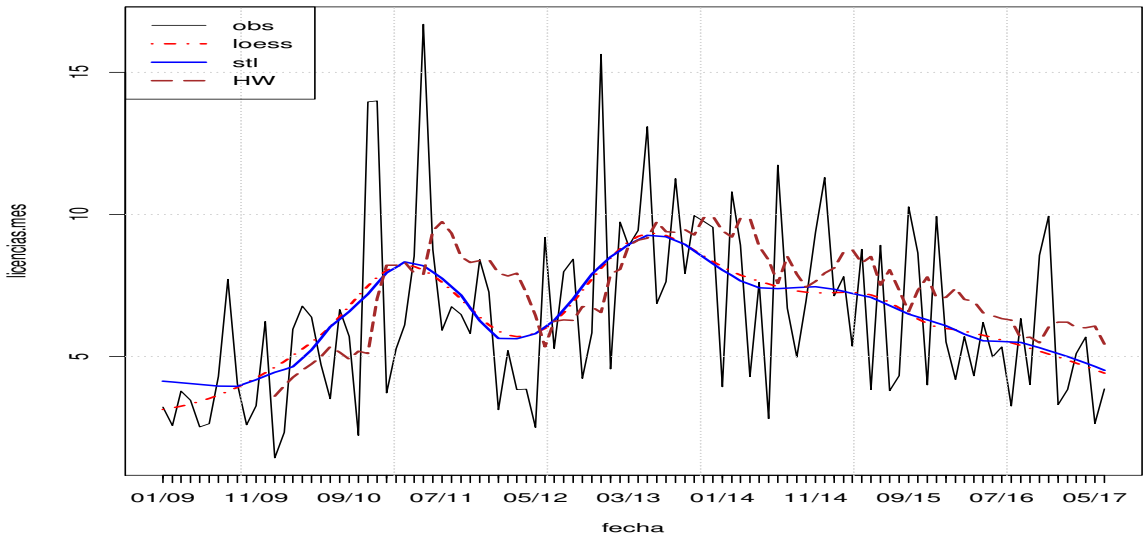

Figura 5.4: Estimación de la tendencia con los Métodos Loess, STL y Holt-Winters

En Warner and Misra [1996] los autores muestran que un modelo de regresión lineal múltiple se puede reformular como una red neuronal perceptrón multi-capa, en donde los coeficientes beta de la regresión son los pesos dentro la neurona y la función de activación es la identidad. La diferencia está en que en la regresión lineal el problema de estimación de parámetros por mínimos cuadrados tiene solución cerrada mientras que en la red neuronal la solución se obtiene por un procedimiento iterativo denominado backpropagation.

También extienden la comparación y muestran que cualquier modelos de regresión lineal generalizada GLM se puede mapear en una red neuronal.

En los inicios de la teoría de redes neuronales se consideró que una diferencia entre éstas y la regresión lineal múltiple era la interpretabilidad del modelo ya que las concebía como modelos caja negra. Con desarrollos recientes esto ya no es cierto pues existen métodos para medir la importancia de las variables predictoras y métodos bootstrap para calcular significación de los parámetros, ver Olden and Jackson [2002]. Sin embargo, cuando se comparan con modelos de regresión, las redes neuronales se utilizan más para realizar pronósticos. Ver por ejemplo, la discusión de R. Tibshirani al artículo de Cheng and Titterington [1994].

También, en desarrollos recientes como las redes NARX, las redes neuronales se pueden interpretar como modelos de regresión no lineales.

Y también son similares a los métodos de suavizamiento, por ejemplo, regresión polinómica local, splines, entre otros, en donde no es posible obtener una forma funcional del modelo.

Lo cual es otra ventaja tanto de los métodos de suavizamiento como de redes neuronales por cuanto no asumen un modelo apriori, sino que permiten mucha flexibilidad.

Las relaciones entre otros modelos estadísticos y las redes neuronales en el área de clasificación se examinan en Cheng and Titterington [1994]. En White [1989] se comprueba que las metodologías de aprendizaje de redes neuronales son inherentemente, métodos estadísticos.

Algunas referencias para esta sección. Textos: Mandic and Chambers [2001] y Bianchi et al. [2017c]. Artículos: Bianchi et al. [2017a], Bianchi et al. [2017b] y Montúfar et al. [2014].

5.3.2. Redes neuronales perceptrón multicapa, MLP

Esta clase de redes neuronales es la más básica. Se conoce también como redes ffnn, por sus siglas en inglés *feedforward neural network*, que puede traducirse como redes con pre-alimentación.

En ([6]) se explica el término prealimentacion (*Feed-forward*) "describe un tipo de sistema que reacciona a los cambios en su entorno, normalmente para mantener algún estado concreto del sistema...responde a las alteraciones de manera predefinida, en contraste con los sistemas retroalimentados".

Una definición de MLP está en Montúfar et al. [2014]. Utiliza el concepto de composición de dos funciones f, g

$$g \circ f(x) = g(f(x)).$$

Una red neuronal se define como una cadena de composiciones de dos tipos de funciones: f_l es una función lineal de pre-activación y g_l es una función no lineal

[6] https://es.wikipedia.org/wiki/Prealimentacion

de activación, donde $l = 1, 2, \ldots, L$ indica el nivel de la cadena de composiciones. Cada nivel se denomina una capa. En cada capa actúa la composición $g_l \circ f_l$. Al final de la cadena de composiciones se aplica una función de activación f_{n_L}, que define la MLP con L capas, como una función de la forma

$$\begin{aligned} &F\colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_L} \\ &\underline{x} \to F(\underline{x}; \underline{\theta}) = f_{n_L} \circ g_L \circ f_L \circ \cdots \circ g_1 \circ f_1(\underline{x}_1) \end{aligned} \tag{5.16}$$

El parámetro $\underline{\theta}$ se compone de matrices de pesos de entrada (*input weight matrices*), $\mathbf{W}_l \in \mathbb{R}^{k \cdot n_l \times n_{l-1}}$ y vectores de sesgo $\underline{b}_l \in \mathbb{R}^{k \cdot n_l}$, $k \geq 1$, para cada capa $l \in \{1, 2, \ldots, L\}$.

A partir de un vector inicial $\underline{x}_1 \in \mathbb{R}^{n_1}$ la cadena genera iterativamente L vectores denominados activaciones. Las componentes de estos vectores se denominan unidades en la capa.

Dada $\underline{x}_{l-1} \in \mathbb{R}^{n_{l-1}}$ con las unidades en la $(l-1)$-ésima capa, la capa de pre-activación l está dada por

$$\begin{aligned} &f_l\colon \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{k \cdot n_l} \\ &\underline{x}_{l-1} \mapsto f_l(\underline{x}_{l-1}) = \mathbf{W}_l\, \underline{x}_{l-1} + \underline{b}_l, \end{aligned} \tag{5.17}$$

Se pasa de la activación $\underline{x}_{l-1}$ a la activación $\underline{x}_l$, aplicando a la cantidad anterior la función de activación g_l

$$\underline{x}_l = g_l(\mathbf{W}_l\, \underline{x}_{l-1} + \underline{b}_l), \tag{5.18}$$

que se puede expresar como una composición

$$\begin{aligned} &g_l \circ f_l\colon \mathbb{R}^{n_{l-1}} \xrightarrow{f_l} \mathbb{R}^{k \cdot n_l} \xrightarrow{g_l} \mathbb{R}^{n_l} \\ &\underline{x}_{l-1} \mapsto \underline{x}_l = g_l \circ f_l(\underline{x}_{l-1}) = g_l(\mathbf{W}_l\, \underline{x}_{l-1} + \underline{b}_l), \end{aligned} \tag{5.19}$$

Se abrevia $g_l \circ f_l$ por h_l.

$$\underline{x}_l = g_l(f_l(\underline{x}_{l-1})) = h_l(\underline{x}_{l-1}). \tag{5.20}$$

La activación $\underline{x}_l$ es la salida de la capa l, y es un vector $\mathbf{x}_l = [\mathbf{x}_{l,1}, \ldots, \mathbf{x}_{l,n_l}]^\top$ de activaciones $\mathbf{x}_{l,i}$ de las unidades $i \in \{1, 2, \ldots, n_l\}$ en esa capa. La activación de la unidad i-ésima en la capa l-ésima está dada por

$$\underline{x}_{l,i} = g_{l,i}(f_{l,i}(\underline{x}_{l-1})).$$

Hay una activación inicial y una final

$$\underline{x}_1 = \mathbf{W}_0 \underline{x} \in \mathbb{R}^{n_1}$$
$$\underline{x}_L = h_L(\underline{x}_{L-1})$$

Las funciones de activación g_l son no lineales de tipo seccionalmente lineales, logístico y tangente hiperbólicas.

Una red MLP con una capa es de la forma

$$\begin{aligned} F &: \mathbb{R}^{n_0} \to \mathbb{R}^{n_1} \\ \underline{x} &\to F(\underline{x}; \underline{\theta}) = f_{n_L}(h_1(\mathbf{W}_0 \underline{x})) \end{aligned} \tag{5.21}$$

5.3.3. Aplicación a series de tiempo

En las aplicaciones a series de tiempo se asume que un modelo no lineal se expresa de manera general como

$$\begin{aligned} X_t &= F(\underline{X}_{t-1:p}; \underline{\theta}) + \epsilon_t, t = 1, 2, \ldots, T. \\ \underline{X}_{t-1:p} &= (X_{t-1}, \ldots, X_{t-p})'. \end{aligned}$$

donde $F : \mathbb{R}^p \to \mathbb{R}$ es una función no lineal, general, $\underline{\theta}$ es el vector de parámetros de la red y ϵ_t es un término de error definido como

$$\epsilon_t = X_t - \mathbb{E}(X_t \mid X_{t-1:p}).$$

Una red neuronal es una aproximación de la esperanza condicional

$$\mathbb{E}(X_t \mid \underline{X}_{t-1:p}) = F(\underline{X}_{t-1:p}).$$

La definición de red MLP con una capa es la siguiente. Defina la función f_1 tal que, para cierta matriz y vectores, y estado inicial $\underline{X}_{t-1:p}$

$$\begin{aligned} f_1(\underline{X}_{t-1:p}) &= \mathbf{W}_1 \underline{X}_{t-1:p} + \mathbf{b}_1 \\ \mathbf{W}_1 &\in \mathbb{R}^{n_1 \times p}, \quad \mathbf{b}_1 \in \mathbb{R}^{n_1}. \end{aligned}$$

Luego, cada componente de $f_1(\underline{X}_{t-1:p})$ está dada por

$$f_1(\underline{X}_{t-1:p})_j = b_j + \sum_{k=1}^{p} [\mathbf{W}_1]_{j,k} X_{t-k}, j = 1, 2, \ldots, n_1,$$

Y se define el siguiente estado de la red como

$$h_1(\underline{X}_{t-1:p}) = g_1(\mathbf{W}_1 \underline{X}_{t-1:p} + \mathbf{b}_1) \in \mathbb{R}^{n_1},$$

La salida es el valor estimado de X_t

$$X_t = \beta_0 + \sum_{j=1}^{n_1} \beta_j h_{1,j}(\underline{X}_{t-1:p}) + \epsilon_t$$

Una red con varias capas itera este resultado formando una cadena de L capas.

La expresión anterior corresponde a una red neuronal de una capa oculta (hidden layer), con n_1 nodos en la capa oculta y p nodos de entrada. Se puede interpretar como un modelo de regresión no lineal dado por

$$X_t = \beta_0 + \sum_{j=1}^{n_1} \beta_j g_1 \left(b_j + \sum_{k=1}^{p} [\mathbf{W}_1]_{j,k} X_{t-k} \right) + \varepsilon_t, \tag{5.22}$$

Para el caso de $p = 2, n_1 = 1$, por ejemplo, con $g_1 = (1 + e^x)^{-1}$, se tiene,

$$X_t = \beta_0 + \frac{\beta_1}{1 + e^{b_1 + \omega_1 X_{t-1} + \omega_2 X_{t-2}}} + \varepsilon_t,$$

5.4. Redes neuronales autoregresivas

Una red MLP con una componente autoregresiva, con $q = 1$ nodos en la capa oculta y p nodos de entrada, se puede definir por

$$X_t = \beta_0 + \sum_{k=1}^{p} [\mathbf{W}_2]_{j,k} X_{t-k} + \sum_{j=1}^{n_1} \beta_j g_1 \left(b_j + \sum_{k=1}^{p} [\mathbf{W}_1]_{j,k} X_{t-k} \right) + \varepsilon_t, \tag{5.23}$$

El modelo permite incorporar estacionalidad, mediante rezagos adicionales estacionales, cambiando

$$b_j + \sum_{k=1}^{p} [\mathbf{W}_1]_{j,k} X_{t-k}$$

por

$$\beta_j + \sum_{k=1}^{p} [\mathbf{W}_1]_{j,k} X_{t-k} + \sum_{k=1}^{P} [\mathbf{W}_1^{(s)}]_{j,k} X_{t-ks},$$

con $p \geq 1$, $P \geq 1$. El modelo (5.23) se indica por NNAR(p, q). Cuando se incluyen P rezagos estacionales de la forma $Y_{t-s}, Y_{t-2s}, \ldots, Y_{t-Ps}$, donde s es el período de la componente estacional, se indica por NNAR$(p, P, q)[s]$. Ejemplos

- NNAR(2,1): un nodo en la capa, 2 nodos de entrada

$$Y_t = \beta_0 + \beta_1 (1 + e^{b_1 + \omega_1 Y_{t-1} + \omega_2 Y_{t-2}})^{-1} + \varepsilon_t,$$

- NNAR(3,2): dos nodos en la capa y p=3 nodos de entrada

$$\begin{aligned} Y_t = \beta_0 &+ \beta_1 (1 + e^{\beta_{1,0} + \beta_{1,1} Y_{t-1} + \beta_{1,2} Y_{t-2} + \beta_{1,3} Y_{t-3}})^{-1} \\ &+ \beta_2 (1 + e^{\beta_{2,0} + \beta_{2,1} Y_{t-1} + \beta_{2,2} Y_{t-2} + \beta_{2,3} Y_{t-3}})^{-1} + \varepsilon_t, \end{aligned}$$

- NNAR(2,1,1)[12]: un nodo en la capa, 2 nodos de entrada auto regresivos, 1 nodo de entrada estacional

$$Y_t = \beta_0 + \beta_1 (1 + e^{\beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \delta_1 Y_{t-12}})^{-1} + \varepsilon_t,$$

Los modelos $NNAR(p, m, q)[s]$ están implementados en la librería `forecast` con la función `nnetar(y, p, P = 1, size)`, donde $size = q$ es el número de nodos en la capa oculta. Es posible utilizarla de la forma `nnetar(y)`, en cuyo caso los órdenes $p, P, size$ se escogen con valores por defecto, que son $P = 1$, p se escoge mediante un modelo autoregresivo de orden p, AR(p), que mejor ajuste según el AIC, y $q = (P + p + 1)/2$.

La función incluye la transformación Box-Cox

$$Y_t^{(\lambda)} = \begin{cases} (Y_t^{\lambda} - 1)/\lambda, \ \lambda \neq 0, \\ \log(Y_t), \ \lambda = 0. \end{cases}$$

que se aplica antes de estimar el modelo. Colocando `nnetar(y, p, P = 1, size,lambda=0)`, se excluye su aplicación.

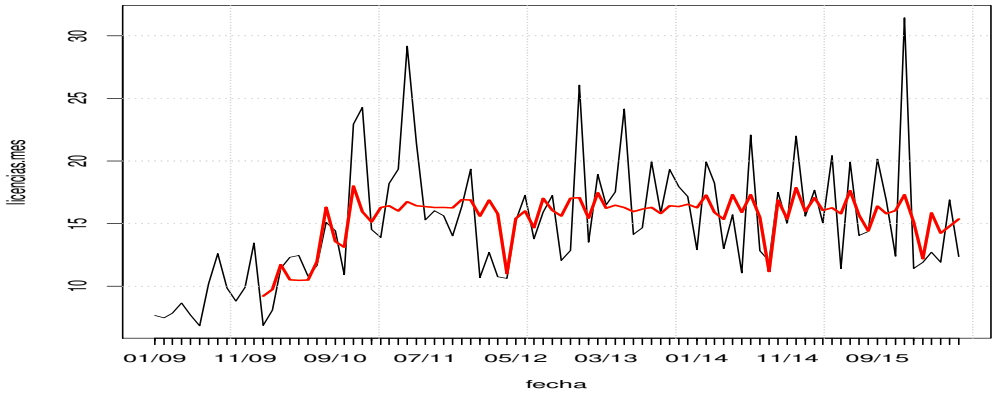

Figura 5.5: Ajuste con NNAR(3,1,2)[12], serie de número de unidades aprobadas para vivienda, mensuales, en 88 municipios en el período 2009/01 - 2017/06

Ejemplo 5.4.1. *Con el Ejemplo 5.1.1, de los datos del DANE sobre estadísticas de edificación, correspondientes a número de unidades aprobadas para vivienda, mensuales, en 88 municipios en el período 2009/01 - 2017/06. Se ajustó un modelo NNAR a los datos.*

```
require(forecast)
y.nnar = nnetar(yi,lambda=0)
print(y.nnar)
Series: yi
Model:  NNAR(3,1,2)[12]
sigma^2 estimated as 0.04706
#----------valores ajustados
yhat.nnar= fitted(y.nnar)
```

Los pronósticos se calculan con la función `forecast`.

```
pr.nnar = forecast(y.nnar,h=m)$mean
pr.nnar = ts(pr.nnar,frequency=12,start=c(2016,06))
```

	R2-ajus	MSE	logAIC	logBIC
local	0.492	11.768	2.498	2.581
tendencia	0.484	11.946	2.524	2.635
bsm	0.725	6.378	1.907	2.046
NNAR	0.280	14.578	2.831	3.223

5.4.1. Redes recurrentes LSTM

En Bianchi et al. [2017c] se señala que es difícil capturar dependencias a largo plazo usando una NNAR porque los gradientes (estocásticos) tienden a hacerse cero o explotar con series largas cuando se aplica el procedimiento de estimación de los pesos mediante regresión no lineal, denominado backpropagation.

Dos modelos particulares, la Memoria a Largo Corto Plazo (LSTM) (*Long Short-Term Memory*) y la red unidad recurrente cerrada (GRU) (*Gated recurrent unit*), han sido propuestos para resolver los dos problemas mencionados.

La red neuronal autorregresiva LSTM es ampliamente utilizada hoy en día debido a su rendimiento superior en el modelado preciso de datos de series correlacionadas, tanto a corto como a largo plazo.

La definición de LSTM introduce tres funciones de activación, $\sigma(), g_1(), g_2()$

$$\begin{aligned}
\text{puente borrado}: &\ \sigma_f[t] = \sigma\left(\mathbf{W}_f \underline{X}_{t-1:p} + \mathbf{R}_f X_{t-1} + \mathbf{b}_f\right), \\
\text{puente actualiza}: &\ \sigma_u[t] = \sigma\left(\mathbf{W}_u \underline{X}_{t-1:p} + \mathbf{R}_u X_{t-1} + \mathbf{b}_u\right), \\
\text{estado candidato}: &\ \tilde{\mathbf{h}}[t] = g_1\left(\mathbf{W}_h \underline{X}_{t-1:p} + \mathbf{R}_h X_{t-1} + \mathbf{b}_h\right), \\
\text{celda estado}: &\ \mathbf{h}[t] = \sigma_u[t] \times \tilde{\mathbf{h}}[t] + \sigma_f[t] \times \mathbf{h}[t-1], \\
\text{puente salida}: &\ \sigma_o[t] = \sigma\left(\mathbf{W}_o \underline{X}_{t-1:p} + \mathbf{R}_o X_{t-1} + \mathbf{b}_o\right), \\
\text{salida}: &\ X_t = \sigma_o[t] \times g_2(\mathbf{h}[t]).
\end{aligned} \tag{5.24}$$

La estimación requiere la librería `Keras`, escrita en Python, con la función `layer_lstm`, y la librería `Tensorflow` para el cálculo de gradientes.