Universidade Vigo

Implementación integrada de modelos de predicción semiparamétricos de series de tiempo

Proyecto Fin de Máster

Máster en Técnicas Estadísticas

Autor:
Borja Lafuente Rego

Directores:
Wenceslao González Manteiga
Manuel Febrero Bande

Universidade de Santiago de Compostela
Curso 2011–2012

Índice general

Capítulo 1

Introducción

Trabajar con datos obtenidos de forma secuencial en el tiempo es muy común. Por ejemplo, en meteorología, todos los días se observa la temperatura, los índices de precipitación, y la velocidad del viento, en economía se estudia la evolución diaria de activos financieros, etc.

El propósito del análisis de series temporales es modelizar el mecanismo estocástico que da lugar a la serie observada y predecir los valores futuros basados en la historia de esa serie.

Una primera aproximación a la modelización de series de tiempo la dan los modelos autorregresivos y de medias móviles, ARMA(p,q),

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \tag{1.1}$$

donde c, $\phi_1, ..., \phi_p$ y $\theta_1, ..., \theta_q$ son parámetros desconocidos que hay que estimar y $\epsilon_t \sim IID(0, \sigma^2)$.

Otra forma de escribir estos modelos sería

$$\phi(B)X_t = c + \theta(B)\epsilon_t, \tag{1.2}$$

donde

$$\begin{aligned} \phi(B) &= 1 - \phi_1 B - ... - \phi_p B^p, \\ \theta(B) &= 1 + \theta_1 B + ... + \theta_q B^q \end{aligned}$$

siendo B el operador retardo tal que $BX_t = X_{t-1}$.

Casos particulares de este modelo serían los procesos autorregresivos AR(p) y los procesos de medias móviles MA(q).

En la Figura 1.1, se muestra la representación gráfica de un proceso ARMA(2,1). Si atendemos al gráfico de autocorrelaciones parciales, observamos que sólo los dos primeros retardos son significativamente distintos de cero, mientras que en el gráfico de autocorrelaciones simples, sólo lo sería el primer retardo.

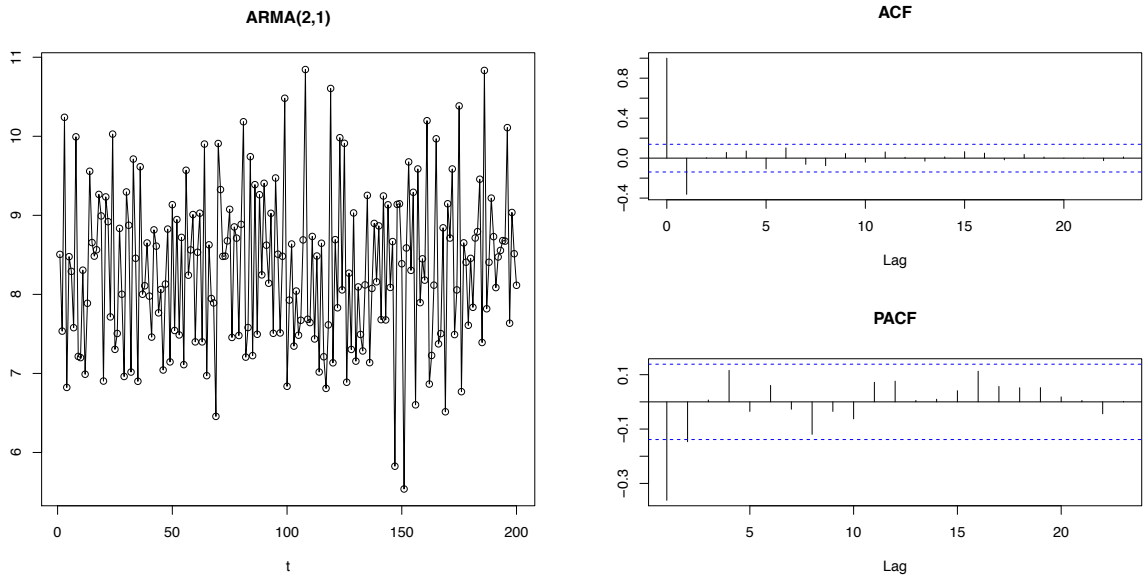

Figura 1.1: Proceso ARMA(2,1), acompañado de sus correspondientes gáficos de autocorrelaciones simples y parciales.

Estos modelos no tienen tendencia, pero en la vida real, los datos suelen presentar esta propiedad. Es por eso que surge un nuevo modelo que generaliza al anterior, los procesos autorregresivos y de medias móviles integrados, ARIMA(p,d,q),

$$\phi(B)(1-B)^d X_t = c + \theta(B)\epsilon_t. \tag{1.3}$$

Por tanto, un proceso ARMA, es un proceso ARIMA(p,d,q) al que se le han aplicado d diferencias regulares.

En la Figura 1.2 tenemos un ejemplo de un proceso ARIMA(1,1,1). Se puede observar en su representación gráfica que presenta una tendencia positiva clara, ya que al aumentar t, el valor X_t aumenta en media. A su lado se muestra la serie diferenciada, en la que se puede apreciar que ha desaparecido la tendencia.

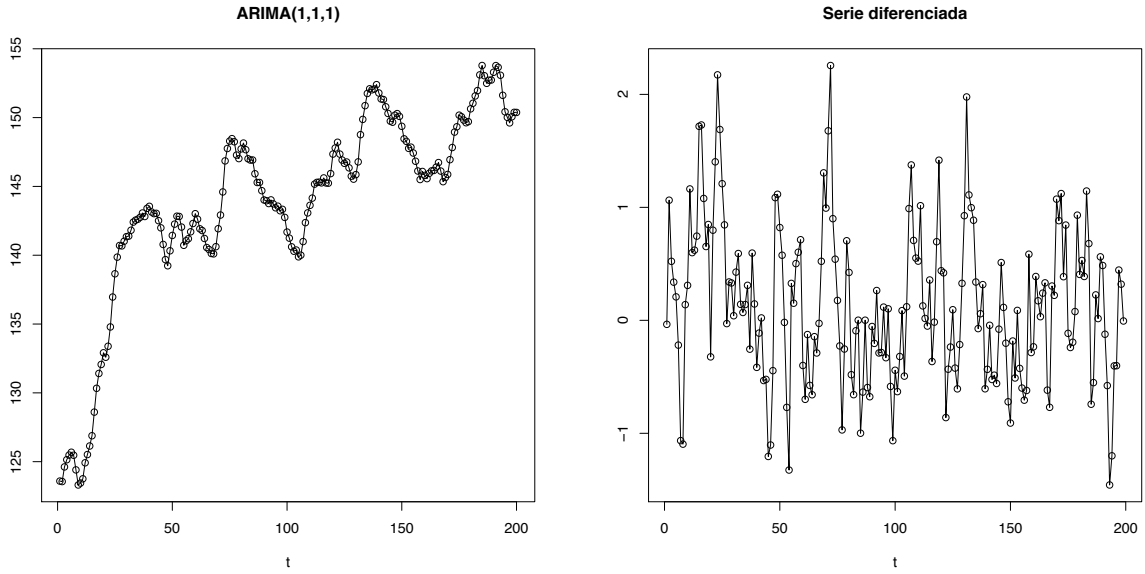

Figura 1.2: Proceso ARIMA(1,1,1) y su serie diferenciada.

Estos dos tipos de procesos, son modelos lineales de series temporales. En los últimos años ha habido mucho interés en el análisis de series temporales no lineales. La mayoría de modelos y métodos utilizados se basan en modelos paramétricos no lineales o modelos no paramétricos. Los modelos paramétricos no lineales pueden ser muy restrictivos en muchos casos, mientras que los modelos no paramétricos son más flexibles.

Este trabajo se centra en estudiar dos modelos de series temporales no lineales paramétricos (TAR y BM) y dos modelos semiparamétricos (FAR y AFAR).

La estructura de este documento será la siguiente:

En los capítulos 2 y 3, se presentan respectivamente 2 modelos paramétricos no lineales y 2 modelos semiparamétricos de series temporales, tratando para cada uno de ellos estimación y predicción. Además, se incluirá un estudio de simulación.

En el capítulo 4 se probará la validez de los modelos vistos en los capítulos 2 y 3 aplicándolos a dos bases distintas de datos reales.

Por último, se incluye un anexo con todo el código de **R** utilizado para realizar este trabajo. Este código es la principal aportación ya que es íntegramente de programación propia.

Capítulo 2

Modelos parámetricos no lineales

2.1. Modelos de Umbral Autorregresivos

Los modelos TAR fueron introducidos por Tong (1990). Estos modelos están motivados por varias características no lineales comúnmente observadas en la práctica, tales como la asimetría en la disminución y aumento de los patrones de un proceso.

Los modelos de umbral autorregresivos (TAR) asumen diferentes formas lineales en distintas regiones del espacio de estados. La división del espacio de estados viene determinada por una variable umbral, X_{t-d}, con $d \geq 1$.

2.1.1. Modelo

Un modelo TAR con $k \geq 2$ regiones, viene definido por

$$X_t = \sum_{i=1}^{k} \left[b_{i0} + b_{i1} X_{t-1} + \ldots + b_{i,p_i} X_{t-p_i} + \sigma_i \epsilon_t \right] I\left(X_{t-d} \in A_i \right), \tag{2.1}$$

con $\epsilon_t \sim IID(0,1)$, d, p_1,..., p_k enteros positivos desconocidos, σ_i y b_{ij} parámetros desconocidos y $\{A_i\}$ es una partición de la recta real tal que $\cup A_i = (-\infty, \infty)$ y $A_i \cap A_j = \emptyset$ con $i \neq j$.

En este trabajo se van a considerar particiones de la recta real del tipo $A_i = (r_{i-1}, r_i]$ con $-\infty = r_0 < r_1 < ... < r_k = \infty$. Comúnmente, a los r_i se les denota por umbrales.

Ejemplo

Tomemos como ejemplo el siguiente modelo TAR

$$X_t = \begin{cases} 0.5 + 0.6X_{t-1} + \epsilon_{1,t} & X_{t-1} \leq 0 \\ -0.5 - X_{t-1} + \epsilon_{2,t} & X_{t-1} > 0 \end{cases} \tag{2.2}$$

con $\epsilon_{1,t} \sim \mathcal{N}(0, 0.3^2)$ y $\epsilon_{2,t} \sim \mathcal{N}(0, 0.5^2)$ y comparémoslo con el siguiente modelo AR(1)

$$X_t = 0.5 + 0.6X_{t-1} + \epsilon_t \tag{2.3}$$

con $\epsilon_t \sim \mathcal{N}(0, 0.3^2)$.

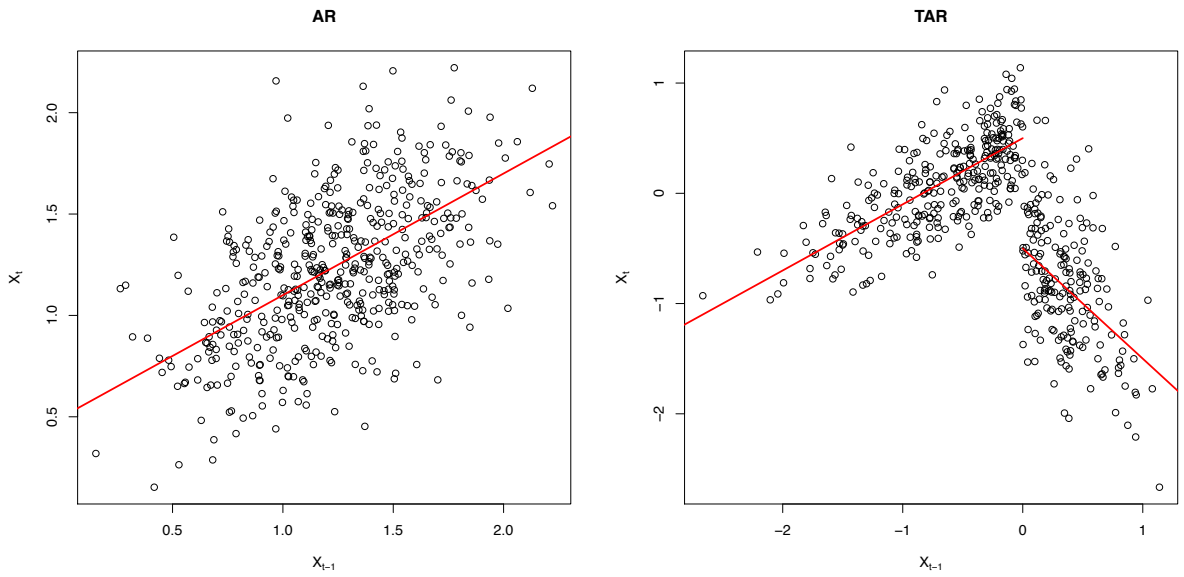

Figura 2.1: Diagramas de dispersión de un modelo AR y un modelo TAR incluyendo las rectas de regresión para cada componente.

En la Figura 2.1 se puede ver como al representar X_t frente a X_{t-1} en el caso del TAR los puntos adoptan dos formas lineales distintas dependiendo de si X_{t-1} es menor o mayor que cero, mientras que en el proceso AR todos los puntos siguen la misma forma lineal independientemente del valor de X_{t-1}.

2.1.2. Estimación e identificación del modelo

Sean X_1,...,X_T valores observados de un modelo TAR con k dado. Suponemos que la partición $\{A_i\}$ y los órdenes p_i son conocidos y denotamos por $\mathbf{b}_i = (b_{i0}, b_{i1}, ..., b_{ip_i})^\tau$.

Basándose en estas observaciones, se estiman los valores de b_{ij}, σ_i y d minimizando

$$\sum_{i=1}^{k} L\left(\mathbf{b}_i, d; A_i\right), \tag{2.4}$$

siendo

$$L\left(\mathbf{b}_i, d; A_i\right) = \sum_{\substack{X_{t-d}\in A_i \\ p<t\leq T}} \left\{X_t - \left(b_{i0} + b_{i1}X_{t-1} + \ldots + b_{i,p_i}X_{t-p_i}\right)\right\}^2. \tag{2.5}$$

Un estimador para las varianzas σ_i^2 vendría definido de la siguiente manera

$$\hat{\sigma_i}^2 = \frac{1}{T_i} L\left(\hat{\mathbf{b}}_i, \hat{d}; A_i\right), \tag{2.6}$$

con T_i el número de elementos en el conjunto $\left\{t : p < t \leq T,\ X_{t-\hat{d}} \in A_i\right\}$.

En la práctica, la partición $\{A_i\}$ es desconocida y, como ya se comentó con anterioridad, se supone del tipo $A_i = (r_{i-1}, r_i]$ con $-\infty = r_0 < r_1 < ... < r_k = \infty$. Para determinarla, se escogen una colección de particiones $\{A_{il}\}$ con $l = 1, .., L$ el número de particiones seleccionadas. Definimos

$$\mathbf{L}(\{A_{il}\}) = \sum_{1<i\leq k} L\left(\mathbf{b}_i, d; A_{il}\right). \tag{2.7}$$

Se tomará la partición $\left\{\hat{A}_i\right\}$ que minimice la ecuación (2.7). En la práctica se suelen tomar particiones pequeñas con $k = 2, 3, 4$.

Si denotamos por $\hat{A}_i, \hat{\mathbf{b}}_i, \hat{d}$ a los estimadores de mínimos cuadrados que minimizan (2.4), se obtiene

$$\hat{\sigma_i}^2 = \frac{1}{T_i} L\left(\hat{\mathbf{b}}_i, \hat{d}; \hat{A}_i\right), \quad i = 1, ..., k. \tag{2.8}$$

Por último, la estimación de los ordenes $\{p_i\}$ se realizaría mediante criterio AIC definido como sigue

$$AIC(\{p_i\}) = \sum_{i=1}^{k} \left[T_i \log\left\{\hat{\sigma}^2(p_i)\right\} + 2(p_i + 1)\right].$$

Se tomarían los órdenes p_i que minimicen la función AIC. Para la estimación de los $\{p_i\}$ también se podrían adaptar los criterios BIC y AICC.

2.1.3. Estudio de simulación

Para probar la efectividad de este modelo, se va a realizar un estudio de simulación. Consideremos el siguiente modelo TAR

$$X_t = \begin{cases} 0.62 + 1.25X_{t-1} - 0.43X_{t-2} + \epsilon_{1,t} & X_{t-2} \leq 3.25 \\ 2.25 + 1.52X_{t-1} - 1.24X_{t-2} + \epsilon_{2,t} & X_{t-2} > 3.25 \end{cases}, \tag{2.9}$$

donde $\epsilon_{1,t} \sim \mathcal{N}(0, 0.2^2)$ y $\epsilon_{2,t} \sim \mathcal{N}(0, 0.4^2)$.

Se realizaron 400 simulaciones del modelo (2.9) de tamaño 600. En la Figura 2.2 se muestra los diagramas de cajas con las estimaciones de los diversos parámetros del modelo. Los puntos rojos marcan el valor real del parámetro.

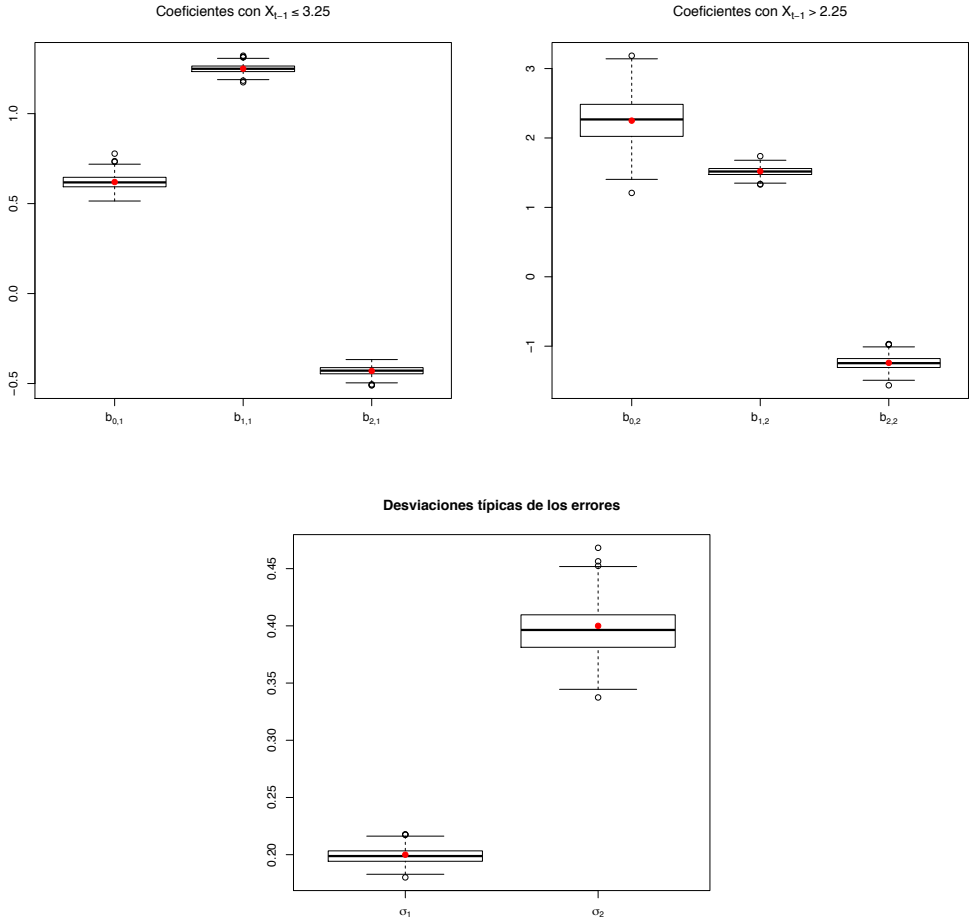

Figura 2.2: Boxplots de los parámetros de las 400 series simuladas.

Para probar la efectividad de este modelo frente a un modelo más clásico como es un ARMA, se realizó para las 400 series simuladas la predicción de los siguientes 20 datos mediante los dos métodos y se calculó su error de predicción de estas dos formas

$$EP1 = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{X}_i - X_i\right|,$$
$$EP2 = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{X}_i - X_i\right)^2.$$

En la Figura 2.3 se puede ver que el modelo TAR comete un menor error a la hora de predecir los valores futuros. Sin embargo, cabe destacar que el modelo ARMA funciona razonablemente bien, ya que la diferencia entre los dos no es muy grande.

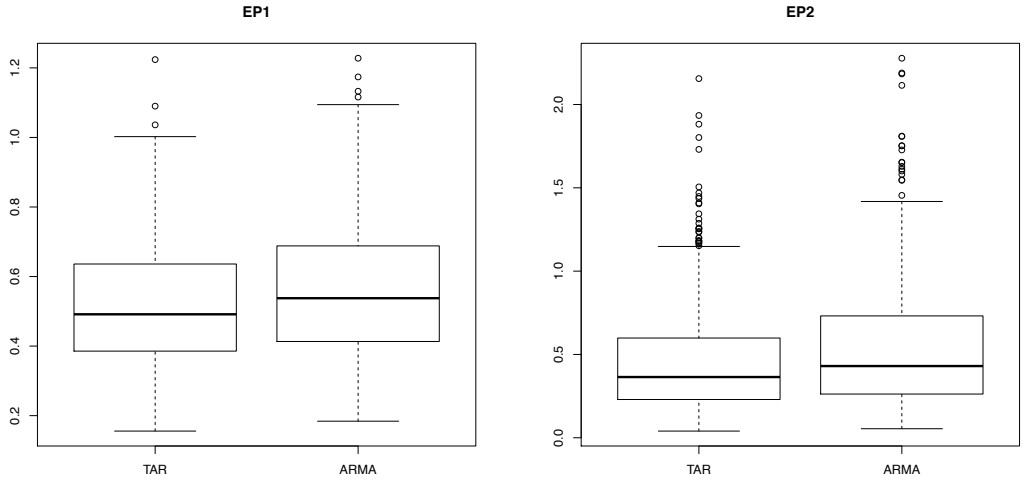

Figura 2.3: Boxplots de los errores de los modelos TAR y ARMA.

A continuación, de las 400 simulaciones, se escoge la que obtuvo un menor error de predicción de tipo 1. Para esta serie, en la Figura 2.4 se muestra la representación de los 100 últimos puntos. Además incluye la predicción a horizonte 10 con los intervalos de confianza bootstrap a nivel $\alpha = 0.05$.

Para el cálculo de los intervalos de confianza bootstrap se le sumó un termino de error $\eta \sim \mathcal{N}(0, \hat{\sigma}_i)$ a cada predicción repitiendo este proceso B veces y se toma como extremo inferior el cuantil de orden α y como extremo superior el cuantil de orden $1 - \alpha$.

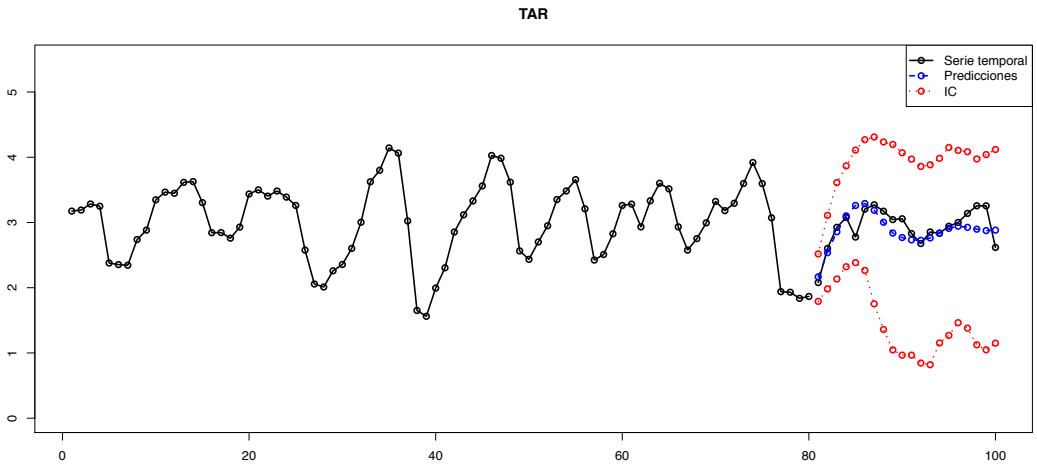

Figura 2.4: Valores reales frente a predicciones.

Por otra parte, en la Figura 2.5 tenemos la representación del gráfico de residuos y de los p–valores del test de Ljung–Box. Además en el test de media cero se obtiene un p–valor de 1 y el test de Shapiro–Wilk de 0.5973, por lo que aceptamos que los residuos siguen una distribución normal con media cero.

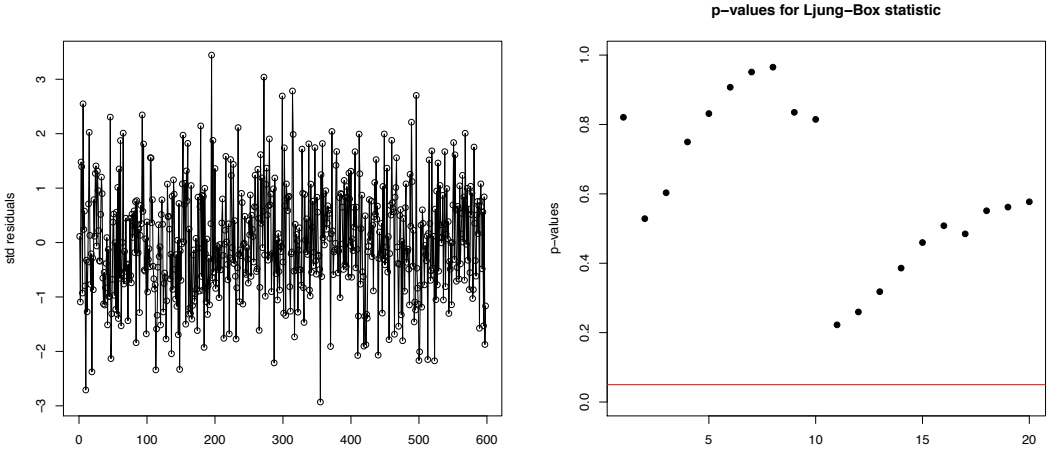

Figura 2.5: Residuos estandarizados y test de Ljung–Box.

Por último, se va a calcular la cobertura de los intervalos de confianza bootstrap empleados. Para esto, se calcularon los intervalos de confianza al 5 % para las 400 series simuladas hasta horizante 5. Se generaron 100 continuaciones de cada una de las series simuladas y se promedió cuantas veces el intervalo de confianza contenía a las continuaciones. Los resultados obtenidos se muestran en la siguiente tabla.

	1	2	3	4	5
Cobertura	0.9433	0.9453	0.9451	0.9448	0.9465
Error estándar	0.0281	0.0292	0.0285	0.0282	0.0267

Tabla 2.1: Coberturas de los intervalos de confianza para las predicciones.

Como se puede ver, el modelo funciona bastante bien tanto a la hora de la estimación del modelo, como a la hora de la predicción.

2.2. Modelos Bilineales

Los modelos bilineales fueron introducidos por Granger y Anderson (1978). Son una generalización de los modelos ARMA a los que se le añade una parte de interacción entre los instantes pasados y las innovaciones.

2.2.1. Modelo

El modelo es de la forma

$$X_t = \sum_{j=1}^{p} b_j X_{t-j} + \sum_{k=1}^{q} a_k \epsilon_{t-k} + \sum_{j=1}^{P}\sum_{k=1}^{Q} c_{jk} X_{t-j}\epsilon_{t-k} + \epsilon_t \tag{2.10}$$

con $\epsilon_t \sim IID(0, \sigma^2)$ y b_j, a_k y c_{jk} parámetros desconocidos. Al modelo (2.10) se le denota por BL(p,q,P,Q).

Estos modelos son útiles para captar picos ocasionales en series de tiempo. Por tanto, pueden ser útiles para el modelado de datos sismológicos, tales como registros de explosiones y terremotos.

Ejemplo

Usemos como ejemplo el siguiente modelo BL(2,0,2,1)

$$X_t = 0.5X_{t-1} - 0.3X_{t-2} + 0.6\epsilon_{t-1}X_{t-1} + 0.5\epsilon_{t-1}X_{t-2} + \epsilon_t \tag{2.11}$$

con $\epsilon_t \sim \mathcal{N}(0, 1)$.

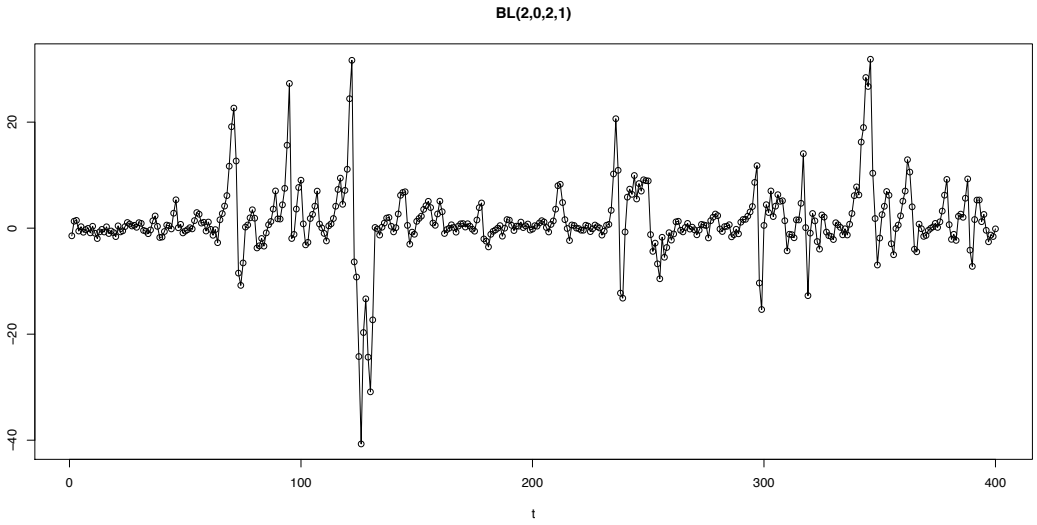

Figura 2.6: Serie de 400 datos generados a partir del modelo bilineal (2.11).

En la Figura 2.6 podemos ver representado este ejemplo. En él se observa como, en determinados instantes, aparecen picos que se corresponden con episodios atípicos en la serie temporal.

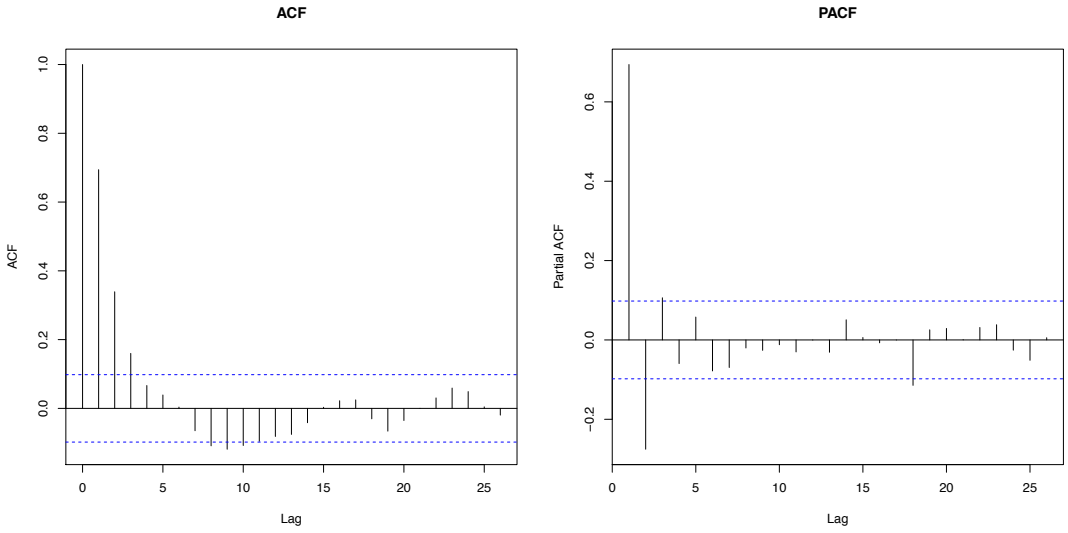

Figura 2.7: Gráficos de autocorrelaciones simples y parciales del modelo (2.11).

Mirando el gráfico de autocorrelaciones simples se observa que éste cae bastante rápido, siendo significativamente igual a cero a partir del retardo 4. Por otra parte, en el gráfico de autocorrelaciones parciales sólo son significativamente distintos de cero los 3 primeros retardos. Guiándose por estas gráficas, cumple las propiedades de un ARMA(3,3).

2.2.2. Estimación e identificación del modelo

Para ajustar un modelo bilineal tenemos que calcular el orden de (p,q,P,Q) y estimar los parámetros del modelo. El rendimiento de estos procedimientos en el contexto de modelos bilineales no está claro debido a la falta de teoría asintótica para la estimación de máxima verosimilitud.

Dado el orden (p,q,P,Q) del modelo bilineal, bajo la hipótesis de que los residuos ϵ_t siguen una distribución normal, se pueden estimar los parámetros a_j, b_k, c_{jk} y σ^2 por el método de máxima verosimilitud como se explica a continuación:

Sean $X_1, ..., X_T$ las observaciones de un proceso BL(p,q,P,Q) con $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$. Llamamos $\theta = \left(\theta_1^T, \theta_2^T\right)^T$ con

$$\theta_1 = (b_1, ..., b_p, a_1, ..., a_q) \;\; y \;\; \theta_2 = (c_{11}, ..., c_{1Q}, c_{21}, ..., c_{PQ})$$

La función de log–verosimilitud puede ser aproximada por

$$l(\theta, \sigma^2) = -\frac{T - p'}{2}\log\sigma^2 - \frac{1}{\sigma^2}\sum_{t=\max(p',q')+1}^{T} \hat{\epsilon}_t(\theta)^2 \tag{2.12}$$

$p' = \max\{p, P\}$ y $q' = \max\{q, Q\}$. $\hat{\epsilon}_{p'}(\theta)$, $\hat{\epsilon}_{p'+1}(\theta)$,... se calculan recursivamente tomando como cero los valores iniciales $\hat{\epsilon}_{p'-1}(\theta)$,...,$\hat{\epsilon}_{p'-q'}(\theta)$.

Por último, el cálculo de los órdenes de (p,q,P,Q) se realizaría mediante criterio AIC, BIC o AICC.

2.2.3. Estudio de simulación

Para el estudio de simulación consideremos el siguiente modelo bilineal BL(2,1,1,1)

$$X_t = 0.6X_{t-1} - 0.8X_{t-2} + \epsilon_t + 0.5\epsilon_{t-1} + 0.8X_{t-1}\epsilon_{t-1} \tag{2.13}$$

donde $\epsilon_t \sim \mathcal{N}(0, 0.2^2)$.

Se realizaron 400 simulaciones del modelo (2.13) de tamaño 400. En la Figura 2.9 se muestra los diagramas de cajas con las estimaciones de los diversos parámetros del modelo. Los puntos rojos marcan el valor real del parámetro.

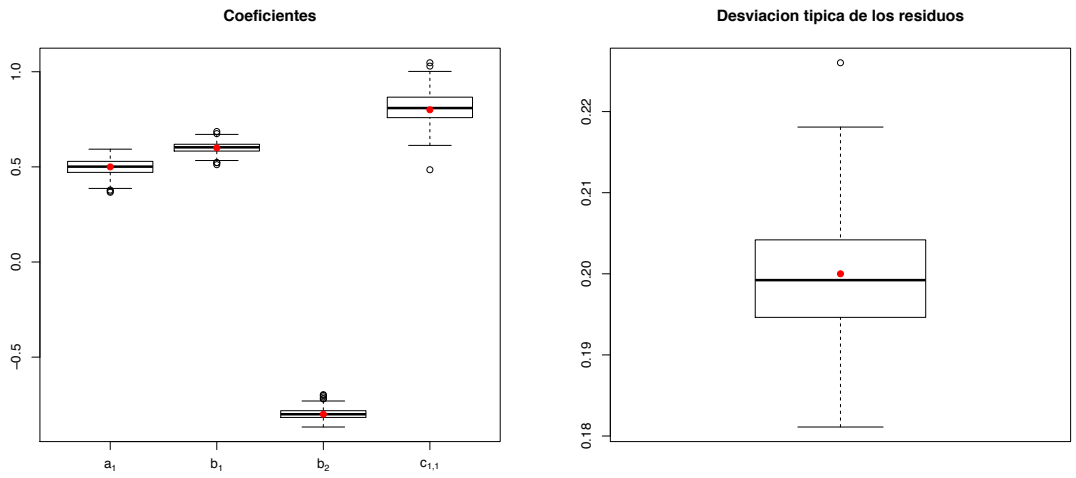

Figura 2.8: Boxplots de los parámetros de las 400 series simuladas.

Igual que en el caso del TAR, se va a probar la efectividad de este modelo frente a un modelo ARMA. Se realizó para las 400 series simuladas la predicción de los siguientes 20 datos mediante los dos métodos y se calculó su error de predicción EP1 y EP2.

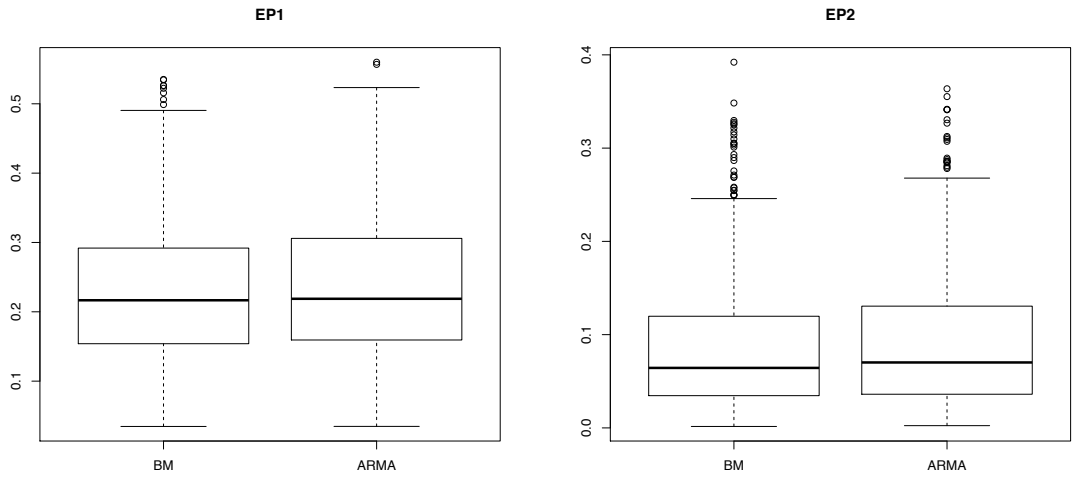

Figura 2.9: Boxplots de los errores de los modelos BM y ARMA.

En la Figura 2.9 se puede ver que el modelo bilineal comete un menor error a la hora de predecir los valores futuros. El modelo ARMA funciona bastante bien, ya que la diferencia entre los dos no es muy grande. Esto es lógico, debido a que el modelo bilineal es muy similar a un proceso ARMA a excepción de la parte de interacción que, en este caso, no es muy grande.

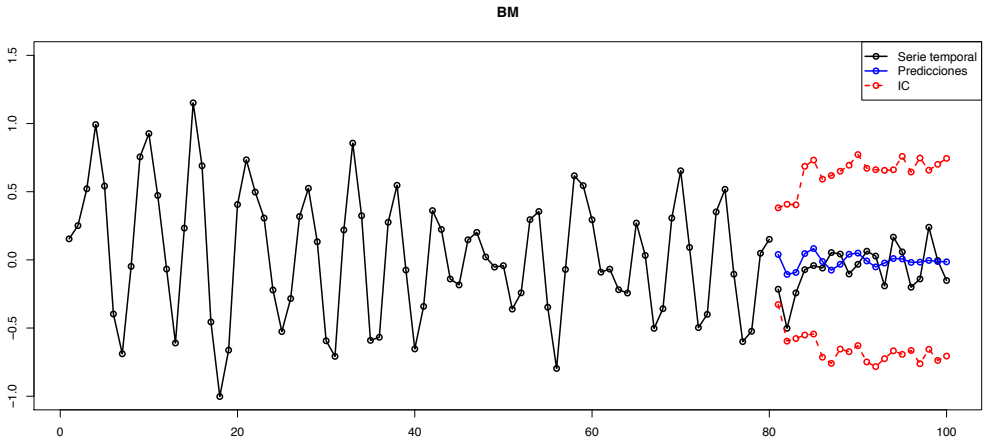

Figura 2.10: Predicciones de los últimos 20 puntos.

De las 400 series generadas, escogemos aquella que tiene un menor error de predicción de tipo 1 y calculamos las predicciones a horizonte 20 con los intervalos de confianza bootstrap al 95 %. En la Figura 2.10 se representan estas predicciones frente a los valores reales.

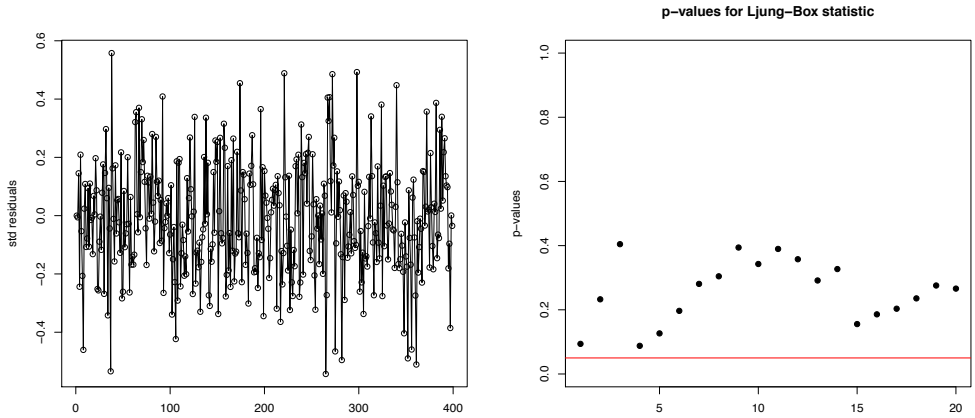

Figura 2.11: Residuos estandarizados y test de Ljung–Box.

Se incluye además un estudio de los residuos. En primer lugar cabe señalar que superan tanto el test de media cero como el test de normalidad de Shapiro–Wilk con p–valores 0.1412 y 0.4825 respectivamente. Además en la Figura 2.11 se tiene el gráfico de los residuos estandarizados y el test de Ljung–Box.

Para terminar, se va a calcular la cobertura de los intervalos de confianza bootstrap empleados. Los resultados obtenidos se muestran en la siguiente tabla,

	1	2	3	4	5
Cobertura	0.9340	0.9536	0.9518	0.9216	0.9680
Error estándar	0.0538	0.0398	0.0301	0.0673	0.0225

Tabla 2.2: Coberturas de los intervalos de confianza para las predicciones.

Por tanto, el modelo bilineal funciona razonablemente bien para el ajuste y predicción de valores futuros.

Capítulo 3

Modelos semiparámetricos

Los modelos de series temporales paramétricos son una herramienta fundamental para el análisis de series temporales cuando los modelos están correctamente especificados.

Una alternativa a éstos es usar métodos no parámétricos. El modelo más natural sería el no parámetrico saturado. Este modelo no impone ninguna forma particular en la función de autorregresión.

Entre los modelos paramétricos y los no paramétricos, se encuentran los modelos semiparamétricos, que son aquellos en los que se les impone cierta forma a las funciones de autorregresión. En particular en este capítulo nos centraremos en estudiar los modelos autorregresivos con coeficiente funcional (FAR) y los modelos autorregrasivos con coeficiente funcional adaptado (AFAR).

3.1. Modelos Autorregresivos con Coeficiente Funcional

El modelo FAR fue introducido por primera vez por Chen y Tsay (1993). Tiene la forma de un proceso autorregresivo sin constante en el que los parámetros del modelo son funciones evaluadas en la variable X_{t-d}.

3.1.1. Modelo

Se dice que una serie temporal X_t sigue un modelo FAR(p,d) si satisface,

$$X_t = f_1(X_{t-d})X_{t-1} + \ldots + f_p(X_{t-d})X_{t-p} + \epsilon_t \tag{3.1}$$

donde $\{\epsilon\}_t \sim IID(0, \sigma^2)$ y son independientes de $X_{t-1}, X_{t-2}, ...$ Las funciones $f_1(\cdot), ..., f_p(\cdot)$ son desconocidas.

El principal objetivo de la especificación del modelo es inferir las funciones $f_i(\cdot)$ a partir de los datos, es decir, estimar la funciones $f_i(\cdot)$ en los puntos X_{t-d}. Además de esto, también habría que estimar el valor de la varianza de los residuos, σ^2.

Todos los modelos paramétricos y no paramétricos pueden ser considerados como modelos de regresión, por tanto, se pueden aplicar esas técnicas a modelos de series temporales.

Introducimos la siguiente notación:

$$Y \equiv X_t,\ X_i \equiv X_{t-i},\ U \equiv X_{t-d}.$$

De esta forma, la $(t–p)$–ésima observación es:

$$Y_t = X_t,\ X_{t1} = X_{t-1}, \ldots, X_{tp} = X_{t-p},\ U_t = X_{t-d}, \quad t = p+1, \ldots, T.$$

Para facilitar la notación, renombraremos los subíndices

$$\{(Y_i, X_{i1}, \ldots, X_{ip}, U_i)\,, i = 1, ..., n\}$$

con $n = T - p$.

Ejemplo

Un modelo EXPAR es de la forma

$$X_t = \sum_{i=1}^{p} \left[\alpha_i + (\beta_i + \gamma_i X_{t-d}) \exp\left(-\theta_i X_{t-d}^2\right)\right] X_{t-i} + \epsilon_t, \tag{3.2}$$

donde $\theta_i \geq 0$ para $i = 1, ..., p$ y $\epsilon_t \sim \text{IID}(0, \sigma^2)$.

Si definimos

$$f_i(u) = \alpha_i + (\beta_i + \gamma_i u) \exp\left(-\theta_i u^2\right) \quad i = 1, ..., p,$$

entonces el modelo (3.2) se puede reescribir como un modelo FAR(p, d)

$$X_t = \sum_{i=1}^{p} f_i(X_{t-d}) X_{t-i} + \epsilon_t. \tag{3.3}$$

Por lo tanto, todo modelo EXPAR se puede poner en términos de un modelo FAR(p, d).

3.1.2. Estimación de las funciones de coeficientes

Las funciones de coeficientes se pueden estimar empleando regresión lineal local. Dado u_0 y u en un entorno de u_0, tenemos por la fórmula de Taylor

$$f_j(u) \approx f_j(u_0) + f_j'(u_0)\,(u-u_0) \equiv a_j + b_j\,(u-u_0)\,. \tag{3.4}$$

Para obtener un estimador de $f_j(u_0)$, basta con minimizar con respecto a $\{a_j\}$ y $\{b_j\}$

$$\sum_{i=1}^{n}\left[Y_i - \sum_{j=1}^{p}\{a_j + b_j(U_i-u_0)\}\,X_{ij}\right]^2 K_h(U_i-u_0), \tag{3.5}$$

con $K_h(\cdot) = h^{-1}K(\cdot/h)$, siendo $K(\cdot)$ una función núcleo y h su parámetro de suavizado. Sean $(\hat{a}_j, \hat{b}_j)$ el estimador de mínimos cuadrados. Por tanto,

$$\hat{f}_j(u_0) = \hat{a}_j \quad \text{y} \quad \hat{f'}_j(u_0) = \hat{b}_j.$$

Para obtener el estimador lineal local, comenzamos definiendo las siguientes matrices,

$$\mathbf{W} = \begin{pmatrix} K_h(U_1-u_0) & 0 & \cdots & 0 \\ 0 & K_h(U_2-u_0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & K_h(U_n-u_0) \end{pmatrix}_{n\times n},$$

$$\tilde{\mathbf{X}} = \begin{pmatrix} \mathbf{X}_1^T & \mathbf{X}_1^T(U_1-u_0) \\ \mathbf{X}_2^T & \mathbf{X}_2^T(U_2-u_0) \\ \vdots & \vdots \\ \mathbf{X}_n^T & \mathbf{X}_n^T(U_n-u_0) \end{pmatrix}_{n\times 2p}.$$

donde $\mathbf{X}_i = (X_{i-1}, ..., X_{i-p})^T$.

Tomando $\mathbf{Y} = (Y_1, Y_2, ..., Y_n)^T$, el problema de regresión lineal local puede expresarse matricialmente

$$(\mathbf{Y} - \tilde{\mathbf{X}}\boldsymbol{\beta})^T\mathbf{X}(\mathbf{Y} - \tilde{\mathbf{X}}\boldsymbol{\beta}),$$

siendo $\boldsymbol{\beta} = (a_1, ..., a_p, b_1, ..., b_p)^T$.

El estimador lineal local viene dado por

$$\hat{\boldsymbol{\beta}} = (\tilde{\mathbf{X}}^T\mathbf{W}\tilde{\mathbf{X}})^{-1}(\tilde{\mathbf{X}}^T\mathbf{W}\mathbf{Y}),$$

de donde se obtiene que:

$$\hat{f}_j = \hat{a}_j = e_{j,2p}^T\hat{\boldsymbol{\beta}},$$

con $e_{j,2p}$ el vector de dimensión $2p$ con un 1 en la posición j–ésima y 0 en el resto.

3.1.3. Estimación del parámetro de suavizado

Como suele suceder en los métodos de suavizado, en el problema de regresión lineal local anterior surge un inconveniente, encontrar el valor óptimo del parámetro ventana h.

Para obtener el h óptimo, emplearemos el método de validación cruzada modificado. Para ello tomamos Q y m dos enteros positivos tales que $n > mQ$. Para cada subserie q de longitud $n - qm$ $(q = 1, ..., Q)$, estimamos las funciones f_j y con ellas calculamos los errores de predicción de la siguiente sección, de longitud m, de la serie temporal. Se escogerá como h aquel que minimice el error medio de predicción (Average Prediction Error) definido como

$$APE(h) = Q^{-1} \sum_{q=1}^{Q} APE_q(h), \tag{3.6}$$

donde

$$APE_q(h) = \frac{1}{m} \sum_{i=n-qm+1}^{n-qm+m} \left[Y_i - \sum_{j=1}^{p} \hat{f}_{j,q}(U_i) X_{ij} \right]^2,$$

con $\hat{f}_{j,q}$ las funciones estimadas usando la q–ésima subserie.

3.1.4. Estudio de simulación

Para el estudio de simulación consideremos el siguiente modelo FAR(2,1)

$$X_t = f_1(X_{t-1})X_{t-1} + f_2(X_{t-1})X_{t-2} + \epsilon_t, \tag{3.7}$$

con $\epsilon_t \sim \mathcal{N}(0, 0.2^2)$, siendo

$$f_1(u) = 0.138 + (0.316 + 0.982u)e^{-3.89u^2}$$
$$f_2(u) = -0.437 - (0.659 + 1.260u)e^{-3.89u^2}.$$

Para la estimación del parámetro de suavizado se toma $Q = 4$ y $m = [0.1n]$, con n el tamaño de la serie. Se evaluará la función APE(h) en la rejilla $h_j = a^j h_0$ $(j = 1, ..., 20)$ siendo $a = 1.2$ y $h_0 = 1.2^{-20}$.

Como función núcleo se toma el núcleo de Epanechnikov

$$K(u) = \frac{3}{4}\left(1 - u^2\right) 1_{|u| \leq 1},$$

es decir

$$K_h(u) = \frac{3}{4h}\left(1 - \left(\frac{u}{h}\right)^2\right) 1_{|\frac{u}{h}|\leq 1}.$$

Se generaron 400 series de tamaño 500 y 400 series de tamaño 250. La representación de los boxplots del valor de h obtenido para cada serie puede verse en la Figura 3.1. Se observa que al aumentar el tamaño de la muestra, los valores del parámetro de suavizado tienen menor varianza y el valor de la mediana se reduce.

Tomando como h óptimo el valor medio de todos los h estimados para cada muestra, se tiene $h = 0.4165$ para las muestras de tamaño 500 y $h = 0.5219$ para las de tamaño 250.

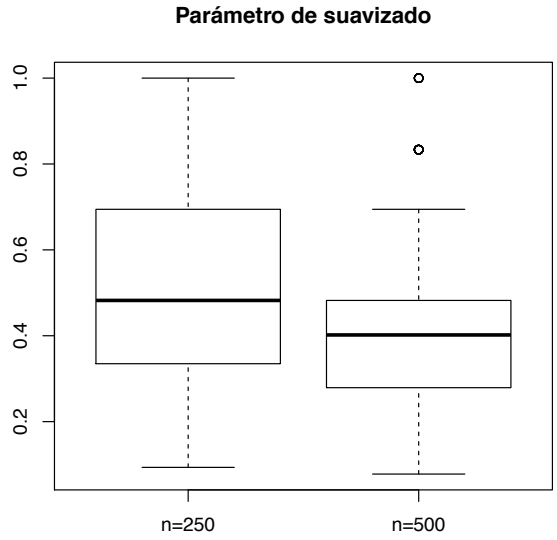

Figura 3.1: Valores estimados de h para tamaños de muestra 250 y 500.

A partir de este punto, trabajaremos solo con la muestra de series de tamaño 500.

Se define el error cuadrático medio (Average Square Error) como

$$ASE = \sum_{j=1}^{p}\left[\frac{1}{n_{grid}}\sum_{k=1}^{n_{grid}}\left[\hat{f}_j(z_k) - f_j(z_k)\right]^2\right],$$

Calculando el ASE para las 400 series de tamaño 500 se obtiene el boxplot que podemos ver en la Figura 3.2. La mediana de los errores está en torno a 0.013, siento el primer cuartil 0.008 y el tercero 0.022.

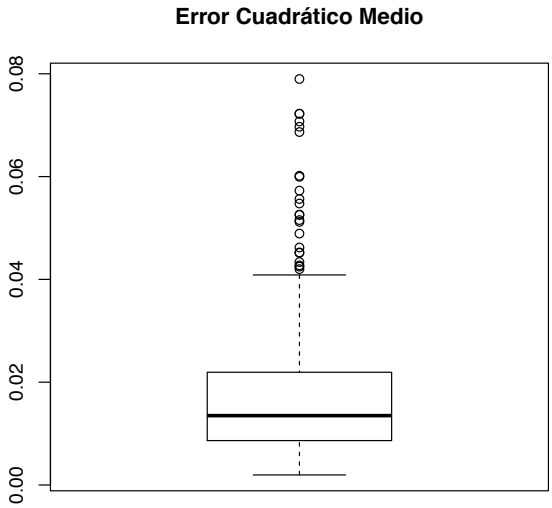

Figura 3.2: Error cuadrático medio de la estimación de las funciones para cada muestra.

Escogemos como muestra aquella que obtuvo un menor error cuadrático medio, y representamos las estimaciones de f_1 y f_2 en la Figura 3.3. Se puede ver que la estimación de ambas funciones es bastante buena.

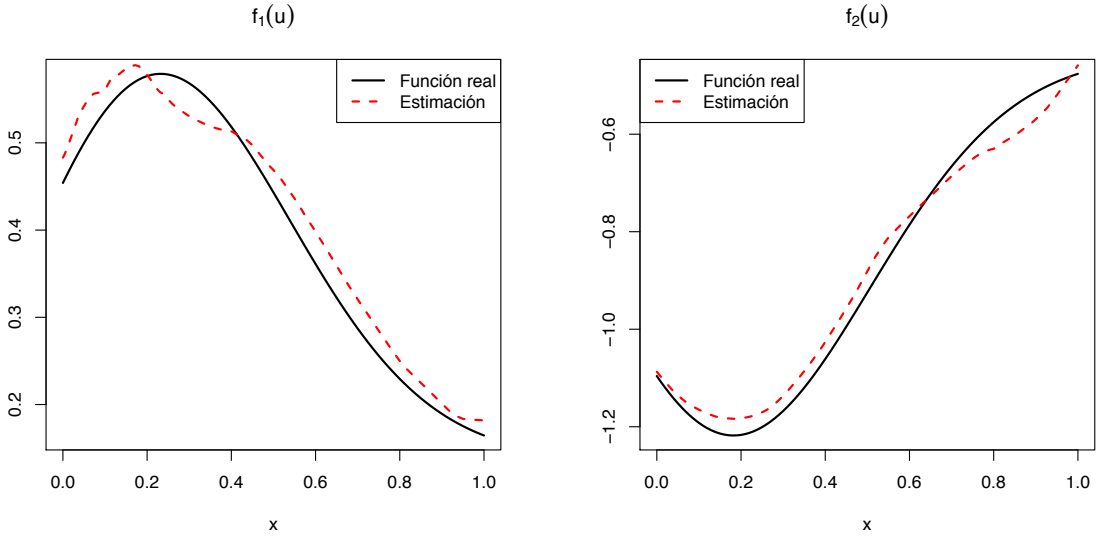

Figura 3.3: Estimaciones de f_1 y f_2 para la muestra con menor ASE.

A continuación, en la Figura 3.5 se muestra la representación de las predicciones de los 20 valores siguientes de la serie y sus correspondientes intervalos de confianza al 95 %.

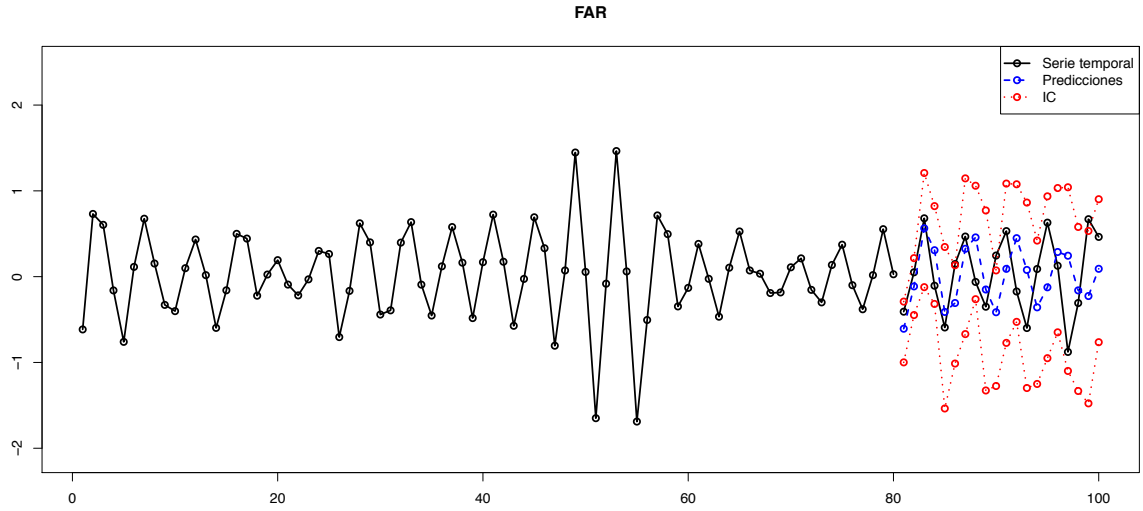

Figura 3.4: Predicción de los 20 últimos valores de la serie.

La representación de los residuos junto con los p–valores del test de Ljung–Box pueden verse en la Figura 3.4. Además el p–valor para el test de media cero es 0.9958 y el p–valor para el test de normalidad de Shapiro–Wilk es 0.2901.

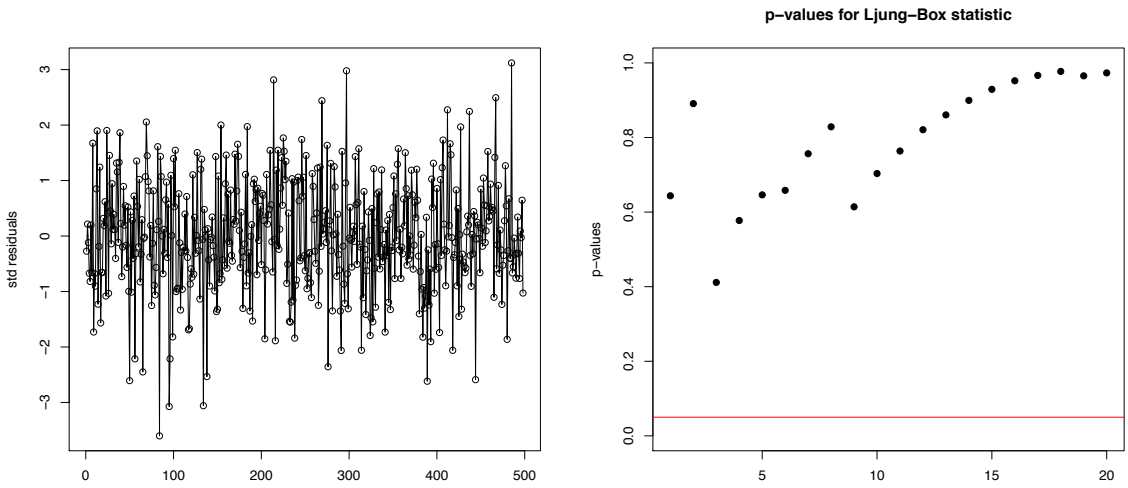

Figura 3.5: Gráfico de residuos estandarizados y test de Ljung–Box.

Por último, se incluye en la tabla 3.1 la cobertura de los intervalos al 95 % para las predicciones hasta horizonte 5.

	1	2	3	4	5
Cobertura	0.9492	0.9472	0.9438	0.9436	0.9486
Error estándar	0.0879	0.0624	0.0647	0.0662	0.0413

Tabla 3.1: Coberturas de los intervalos de confianza para las predicciones.

Se puede ver que este modelo funciona muy bien para estos datos simulados. Realiza una buena estimación de las funciones f_j y predice muy bien valores futuros.

3.2. Modelos Autorregresivos con Coeficiente Funcional Adaptado

Una posible generalización del modelo FAR sería considerar que la funciones estén evaluadas en una combinación lineal de p instantes anteriores en lugar de estar evaluadas en la variable con retardo d. Este modelo se conoce como Modelo autorregresivo con coeficiente funcional adaptado (AFAR) y se puede consultar en Fan y Yao (2003).

3.2.1. Modelo

Diremos que una serie temporal X_t sigue un modelo AFAR$(p, \boldsymbol{\beta})$ si cumple que

$$X_t = g_0(\boldsymbol{\beta}^T \mathbf{X}_{t-1}) + \sum_{j=1}^{p-1} g_j(\boldsymbol{\beta}^T \mathbf{X}_{t-1}) X_{t-j} + \epsilon_t, \tag{3.8}$$

con $\mathbf{X}_{t-1} = (X_{t-1}, ..., X_{t-p})$ y $\boldsymbol{\beta}$ una dirección en el espacio p–dimensional $\mathbb{R}^p$ tal que $\|\boldsymbol{\beta}\| = 1$.

Los residuos ϵ_t son variables aleatorias independientes e idénticamente distribuidas de media cero y varianza σ^2. Además, se asume que ϵ_t es independiente de $\mathbf{X}_{t-1}$.

Ejemplo

Todo modelo TAR se puede reescribir como un modelo AFAR. Tomemos como ejemplo el siguiente proceso TAR,

$$X_t = \begin{cases} 1 + 0.5X_{t-1} + 0.2X_{t-2} + \epsilon_{1,t} & X_{t-1} \leq -5 \\ 0.5 + 0.6X_{t-1} + \epsilon_{2,t} & -5 < X_{t-1} \leq 5 \\ -1 - 0.2X_{t-1} + 0.8X_{t-2} + \epsilon_{3,t} & X_{t-1} > 5 \end{cases}. \tag{3.9}$$

Denotamos las funciones

$$g_0(u) = 1I(u \leq -5) + 0.5I(-5 < u \leq 5) - 1I(u > 5),$$
$$g_1(u) = 0.5I(u \leq -5) + 0.6I(-5 < u \leq 5) - 0.2I(u > 5),$$
$$g_2(u) = 0.2I(u \leq -5) + 0.8I(u > 5).$$

Por tanto, el modelo AFAR se escribiría de la siguiente forma

$$X_t = g_0(\boldsymbol{\beta}^T \mathbf{X}_{t-1}) + g_1(\boldsymbol{\beta}^T \mathbf{X}_{t-1}) X_{t-1} + g_2(\boldsymbol{\beta}^T \mathbf{X}_{t-1}) X_{t-2} + \epsilon_t, \tag{3.10}$$

donde $\boldsymbol{\beta} = (1, 0, 0)$ y $\mathbf{X}_{t-1} = (X_{t-1}, X_{t-2}, X_{t-3})$.

3.2.2. Estimación

En este modelo nos encontramos con varios elementos a estimar: la dirección $\boldsymbol{\beta}$, las funciones de coeficientes g_i y el parámetro de suavizado h. A continuación se muestra la estimación de estos parámetros uno a uno, para finalizar este punto con un algoritmo que permita calcularlos todos conjuntamente.

Estimador lineal local dado $\boldsymbol{\beta}$

Dado $\boldsymbol{\beta}$, las funciones g_j se obtienen mediante regresión lineal local en un entorno de $\boldsymbol{\beta}^T \mathbf{X} \approx z$. Es decir, hay que minimizar

$$\sum_{t=1}^{n} \left[Y_t - \sum_{j=0}^{p-1} \left\{ b_j + c_j \left(\boldsymbol{\beta}^T \mathbf{X}_t - z \right) \right\} X_{tj} \right]^2 K_h \left(\boldsymbol{\beta}^T \mathbf{X}_t - z \right) w \left(\boldsymbol{\beta}^T \mathbf{X}_t \right),$$

con respecto a $\{b_j\}$ y $\{c_j\}$. La función de peso $w(\cdot)$ se introduce para atenuar el efecto frontera.

Para obtener el estimador lineal local, comenzamos definiendo las siguientes matrices,

$$\mathbf{W}(z) = \begin{pmatrix} K_h(\boldsymbol{\beta}^T \mathbf{X}_1 - z) w(\boldsymbol{\beta}^T \mathbf{X}_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & K_h(\boldsymbol{\beta}^T \mathbf{X}_n - z) w(\boldsymbol{\beta}^T \mathbf{X}_n) \end{pmatrix}_{n \times n},$$

$$\tilde{\mathbf{X}}(z) = \begin{pmatrix} \mathbf{U}_1^T & (\boldsymbol{\beta}^T \mathbf{X}_1 - z) \mathbf{U}_1^T \\ \mathbf{U}_2^T & (\boldsymbol{\beta}^T \mathbf{X}_2 - z) \mathbf{U}_1^T \\ \vdots & \vdots \\ \mathbf{U}_n^T & (\boldsymbol{\beta}^T \mathbf{X}_n - z) \mathbf{U}_1^T \end{pmatrix}_{n \times 2p},$$

donde $\mathbf{U}_i = (1, X_{i-1}, ..., X_{i-p+1})^T$ y $\mathbf{X}_i = (X_{i-1,...,X_{i-p}})$.

Tomando $\mathbf{Y} = (Y_1, Y_2, ..., Y_n)^T$, el problema de regresión lineal local puede expresarse matricialmente

$$(\mathbf{Y} - \tilde{\mathbf{X}}\boldsymbol{\theta})^T \mathbf{X} (\mathbf{Y} - \tilde{\mathbf{X}}\boldsymbol{\theta}),$$

siendo $\boldsymbol{\theta} = (a_0, ..., a_{p-1}, b_0, ..., b_{p-1})^T$.

El estimador lineal local viene dado por

$$\hat{\boldsymbol{\theta}} = (\tilde{X}(z)^T W(z) \tilde{X}(z))^{-1} (\tilde{X}(z)^T W(z) Y).$$

Si se define

$$\hat{\boldsymbol{\theta}} = \left(\hat{a}_0, ..., \hat{a}_{p-1}, \hat{b}_0, ..., \hat{b}_{p-1}\right),$$

se obtiene que:

$$\hat{g}_j(z) = \hat{a}_j,$$
$$\hat{\dot{g}}_j(z) = \hat{b}_j.$$

Estimador de $\boldsymbol{\beta}$ dadas las funciones de coeficientes

Se estimará $\boldsymbol{\beta}$ minimizando la función

$$R(\boldsymbol{\beta}) = \frac{1}{n} \sum_{t=1}^{n} \left\{ Y_t - \sum_{j=0}^{p-1} \hat{g}_j(\boldsymbol{\beta}^T \mathbf{X}_t) X_{tj} \right\}^2 w(\boldsymbol{\beta}^T \mathbf{X}_t), \tag{3.11}$$

con $X_{t0} = 1$.

Minimizar la función $R(\boldsymbol{\beta})$ tiene un coste computacional muy elevado. Para acelerar este proceso, se empleará el método de Newton–Raphson.

Si $\hat{\boldsymbol{\beta}}$ es el valor que hace mínimo la función (3.11), se tiene que $\dot{R}(\hat{\boldsymbol{\beta}}) = 0$, siendo $\dot{R}(\cdot)$ la derivada de $R(\cdot)$. Luego para cada $\boldsymbol{\beta}^{(0)}$ en un entorno de $\hat{\boldsymbol{\beta}}$

$$0 = \dot{R}\left(\hat{\boldsymbol{\beta}}\right) \approx \dot{R}\left(\boldsymbol{\beta}^{(0)}\right) + \ddot{R}\left(\boldsymbol{\beta}^{(0)}\right)\left(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{(0)}\right),$$

con $\ddot{R}(\cdot)$ la matriz Hessiana de $R(\cdot)$.

Teniendo esto es cuenta, la estimación de β se hará mediante un proceso iterativo:

$$\boldsymbol{\beta}^{(1)} = \boldsymbol{\beta}^{(0)} - \left[\ddot{R}\left(\boldsymbol{\beta}^{(0)}\right)\right]^{-1} \dot{R}\left(\boldsymbol{\beta}^{(0)}\right),$$

siendo $\boldsymbol{\beta}^{(0)}$ un estimador inicial que se irá actualizando por el de $\boldsymbol{\beta}^{(1)}$ en cada iteración, hasta que se cumpla un criterio de parada.

Selección del parámetro de suavizado

Para estimar el parámetro ventana, se empleará una modificación del método de validación cruzada generalizada (GCV). El método GCV selecciona h minimizando

$$\mathrm{GCV}(h) = \frac{1}{n\left[1 - n^{-1}\mathrm{tr}(\mathbf{H}(h))\right]^2} \sum_{t=1}^{n} \left[Y_t - \hat{Y}_t\right]^2 w(\boldsymbol{\beta}^T \mathbf{X}_t), \tag{3.12}$$

con $\mathbf{H}(h)$ tal que $(\hat{Y}_1, ..., \hat{Y}_n)^T = \mathbf{H}(h)Y$.

Bajo condiciones de regularidad se tiene que

$$\mathrm{GCV}(h) = a_0 + a_1 h^4 + \frac{a_2}{nh} + o_p(h^4 + n^{-1}h^{-1}).$$

Por tanto, la ventana óptima sería

$$h_{\mathrm{opt}} = \left(\frac{a_2}{4na_1}\right)^{1/5}.$$

Los coeficientes a_0, a_1 y a_2 se estimarán de $\{\mathrm{GCV}(h_k)\}$ mediante regresión por mínimos cuadrados.

Algoritmo

A continuación se muestra el algoritmo utilizado para estimar todos los parámetros comentados con anterioridad. Este algoritmo fue propuesto por Fan y Yao (2003). Consta de los siguientes pasos:

- **Paso 1:** Estandarizar el conjunto de datos $\{\mathbf{X}_t\}$ de tal forma que tenga media 0 y matriz de varianzas–covarianzas la identidad $\mathbf{I}_p$. Dar un valor inicial $\boldsymbol{\beta}_0$.
- **Paso 2:** Para cada h_k, $k = 1, ..., q$ repetir los dos siguientes pasos hasta que dos valores sucesivos de $R(\boldsymbol{\beta})$ difieran de forma insignificante:
  - Dado $\boldsymbol{\beta}$ estimar las funciones $g_j(\cdot)$.
  - Con las $g_j(\cdot)$ estimadas, recalcular $\boldsymbol{\beta}$.
- **Paso 3:** Para $k = 1, ..., q$ se calcula el $\mathrm{GCV}(h_k)$ con $\boldsymbol{\beta}$ el estimado en el paso anterior. Se define $\hat{h} = \{\hat{a}_2/(4n\hat{a}_1)\}^{1/5}$ si $\hat{a}_1$ y $\hat{a}_2$ son positivos y $\hat{h} = \mathrm{argmin}_{h_k}\mathrm{GCV}(h_k)$ en caso contrario.
- **Paso 4:** Para $h = \hat{h}$ repetir el **Paso 2** hasta que se cumpla el criterio de convergencia.

3.2.3. Estudio de simulación

En este apartado, se va a probar la efectividad del método propuesto para la estimación del parámetro de suavizado h y del vector de dirección $\boldsymbol{\beta}$. Para ello, consideremos el siguiente modelo AFAR

$$X_t = -X_{t-2}\exp\left(-\frac{X_{t-2}^2}{2}\right) + \frac{1}{1+X_{t-2}^2}\cos\left(1.5X_{t-2}\right)X_{t-1} + \epsilon_t$$

con $\epsilon_t \sim \mathcal{N}\left(0, 0.5^2\right)$ y siendo

$$g_0(z) = -z\exp\left(-\frac{z^2}{2}\right) \quad \text{y} \quad g_1(z) = \frac{\cos(1.5z)}{1+z^2}.$$

El modelo sigue la forma de un AFAR con $p = 2$ y $\boldsymbol{\beta} = (0, 1)$.

En el estudio de simulación se tuvieron en cuenta los siguientes puntos:

- Para la estimación del parámetro ventana h se probó entre $h_j = 0.2 \times 1.2^{j-1}$ con $j = 1, ..., 15$. Estos valores cubren el intervalo desde 0.2 hasta 2.57.
- Por otra parte, la función de pesos utilizada es $w(z) = I(|z| \leq 1.5 + \delta)$, $\delta \geq 0$. En este caso se tomo $\delta = 0.001$.
- Además, como en el caso del FAR, se empleará el núcleo de Epanechnikov.
- La estimación de las funciones g_j se hizo tomando una rejilla de 500 puntos equidistantes en el intervalo [-1.5,1.5].
- Para el estudio de simulación se generaron 200 series de tamaño 300 siguiendo el modelo anterior.
- Para el cálculo de los errores de predicción a la hora de estimar las funciones g_j se empleará la siguiente fórmula,

$$PE = \frac{1}{pn_{grid}}\sum_{j=0}^{p-1}\sum_{k=1}^{n_{grid}}\left|\hat{g}_j(z_k) - g_j(z_k)\right|.$$

Para estudiar el funcionamiento del algoritmo a la hora de estimar la dirección $\boldsymbol{\beta}$ se consideró $\left|\boldsymbol{\beta}^T\hat{\boldsymbol{\beta}}\right|$, es decir, el valor absoluto del coseno de los ángulos entre $\boldsymbol{\beta}$ y $\hat{\boldsymbol{\beta}}$. En la Figura 3.6 se puede ver el boxplot de estos cosenos. La mediana tiene un valor de 0.9397, muy cercano a 1.

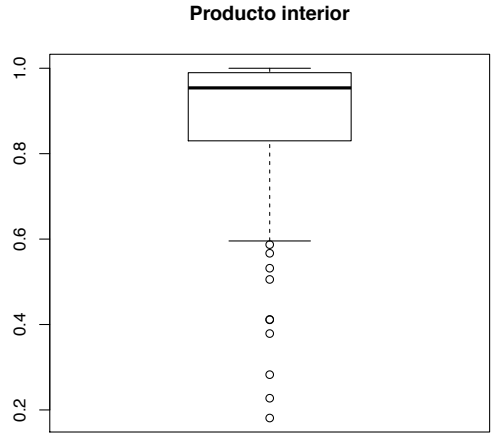

Figura 3.6: Valor absoluto del producto interior $\boldsymbol{\beta}^T \hat{\boldsymbol{\beta}}$.

En la Figura 3.7 se muestran dos gráficos. En el de la izquierda, tenemos los boxplots con las estimaciones de h. Se puede apreciar que el valor de h disminuye si utilizamos el valor real de $\boldsymbol{\beta}$ para su estimación. En el gráfico de la derecha se representan los boxplots de los errores de estimación de las funciones g_j. Como es lógico, el error se reduce considerablemente si usamos el valor real de $\boldsymbol{\beta}$ a la hora de estimar las funciones del modelo.

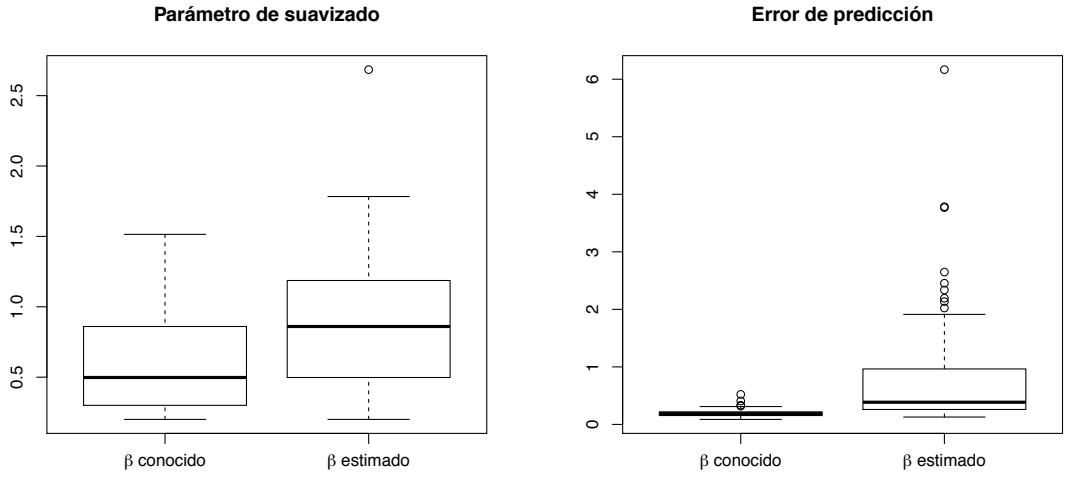

Figura 3.7: Estimación del parámetro h y errores de predicción.

A partir de este punto, trabajaremos con la serie que obtuvo un menor error de predicción para el cálculo de las funciones g_0 y g_1. La representación de estas estimaciones puede verse en la Figura 3.8.

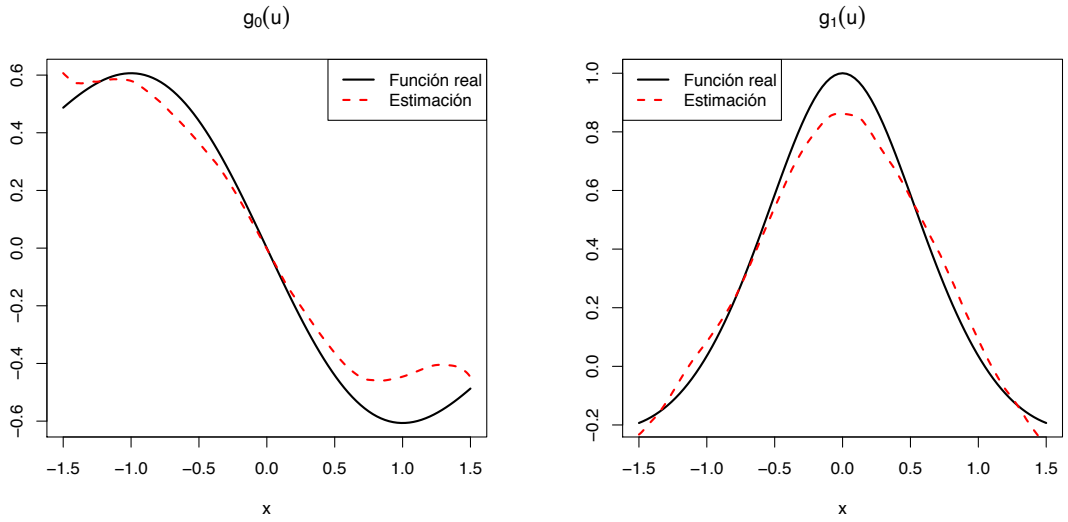

Figura 3.8: Estimaciones de g_0 y g_1 para la muestra con menor error.

La representación de los últimos 100 puntos de esta serie junto con las predicciones para los últimos 20 puntos y sus intervalos de confianza al 95 % pueden verse en la Figura 3.9

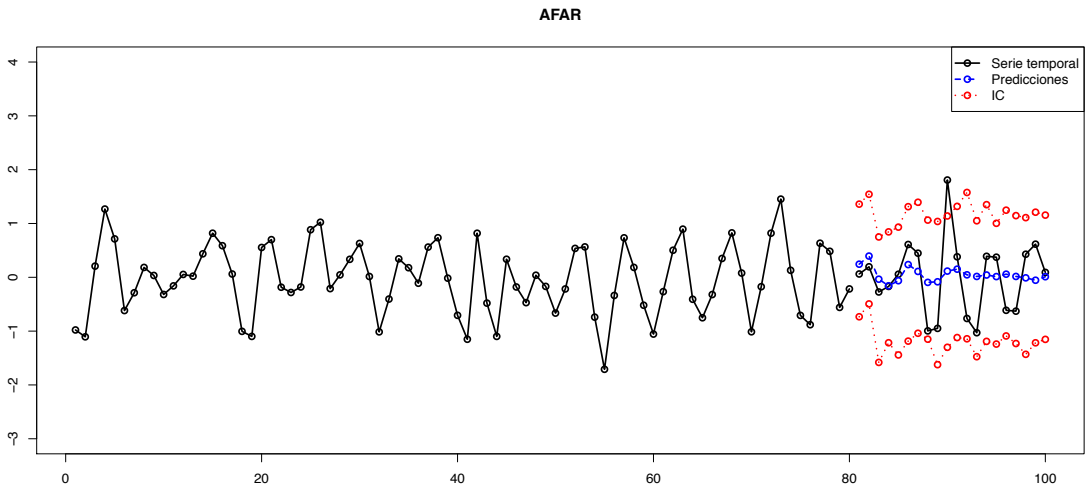

Figura 3.9: Predicciones e intervalos de confianza.

Se incluye además un estudio de residuos. Para empezar, en la Figura 3.10 se muestra la representación de los residuos estandarizados y los p–valores del test de Ljung–Box. El p–valor para el test de media cero es 0.870. Además los residuos pasan el test de normalidad de Shapiro–Wilk con un p–valor de 0.383.

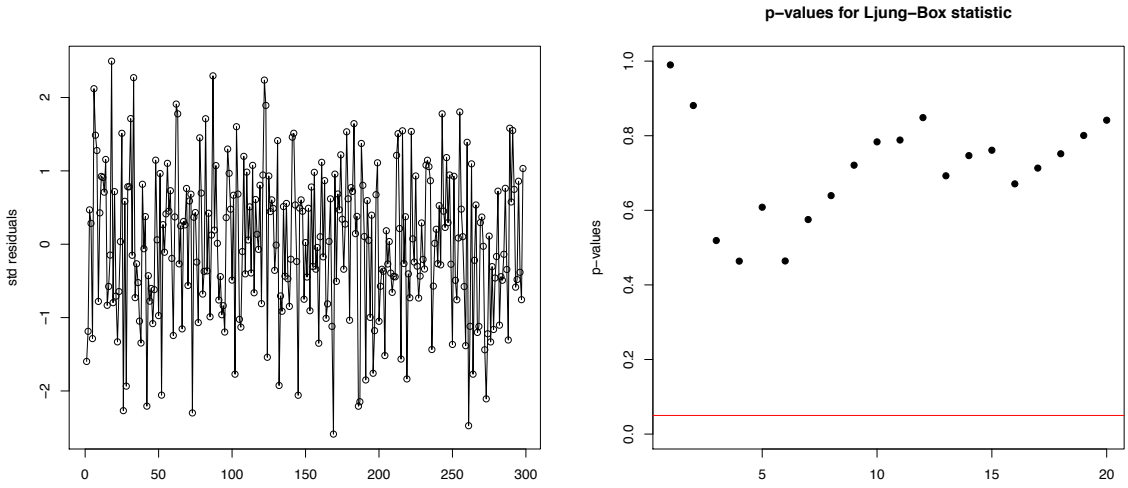

Figura 3.10: Residuos estandarizados y test de Ljung–Box.

Por último, se incluye en la Tabla 3.2 la cobertura de los intervalos al 95 % para las predicciones hasta horizonte 5.

	1	2	3	4	5
Cobertura	0.9565	0.9527	0.9541	0.9558	0.9572
Error estándar	0.0482	0.0459	0.0396	0.0362	0.0343

Tabla 3.2: Coberturas de los intervalos de confianza para las predicciones.

En este caso, el modelo también funciona razonablemente bien. Aproxima bastante bien las funciones del modelo pero, en este caso, las predicciones no son tan buenas como los modelos anteriores.

Capítulo 4

Aplicación a datos reales

En este capítulo se van a aplicar los modelos comentados en este documento al estudio de dos bases de datos reales distintas, una de carácter medioambiental (velocidad del viento) y otra de carácter económico (índice de producción industrial de EEUU).

4.1. Velocidad del viento

Comenzamos analizando unos datos de velocidad media diaria del viento tomados en una estación meteorológica situada en el aeropuerto de Santiago de Compostela. Estos datos fueron obtenidos de la base de datos del AEMET y comprenden desde 1 de noviembre de 2003 hasta el 1 de enero de 2005. Tenemos por tanto una serie temporal de 428 datos. A continuación se puede ver la representación de la serie temporal,

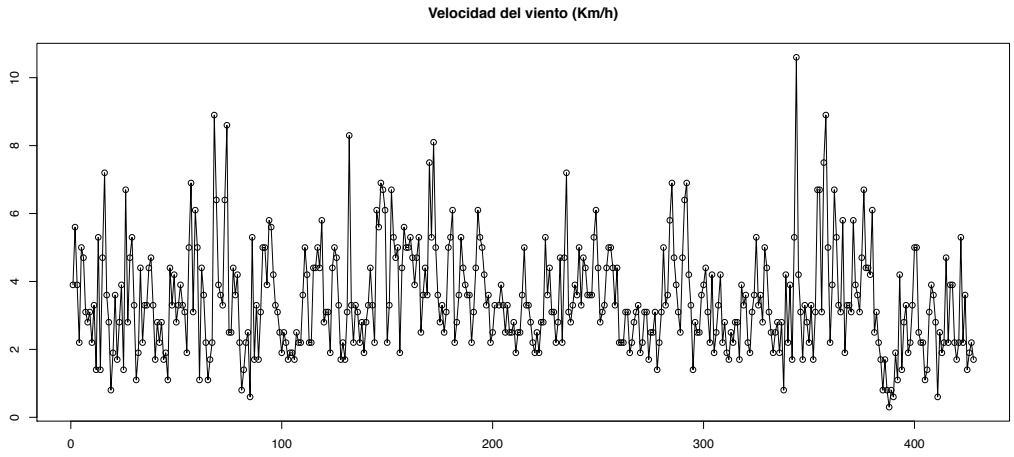

Figura 4.1: Velocidad media del viento en el aeropuerto de Santiago de Compostela.

TAR

Tratemos de modelizar estos datos mediante un TAR. Tras probar con varias particiones, se escogió mediante criterio AIC la que mejor se adaptaba a estos datos, y esta fue $\{(-\infty, 2.315], (2.315, \infty)\}$ con $d = 2$. Se probó también con otros valores de d obteniéndose peores resultados.

Una vez estimada la partición, el mejor modelo que ajusta los datos es aquel con $p_1 = p_2 = 1$. El modelo obtenido es

$$X_t = \begin{cases} 1.5198 + 0.5613X_{t-1} & X_{t-2} \leq 2.315 \\ 2.3688 + 0.3268X_{t-1} & X_{t-2} > 2.315 \end{cases}. \tag{4.1}$$

El número de observaciones en cada región es respectivamente 117 y 311. En la Figura 4.2 se muestran los 250 primeros valores de la serie temporal frente a sus valores ajustados por el modelo TAR estimado. En ella se puede ver que el modelo ajusta razonablemente bien a la serie original, si bien es cierto que tiende a infraestimar los picos más elevados.

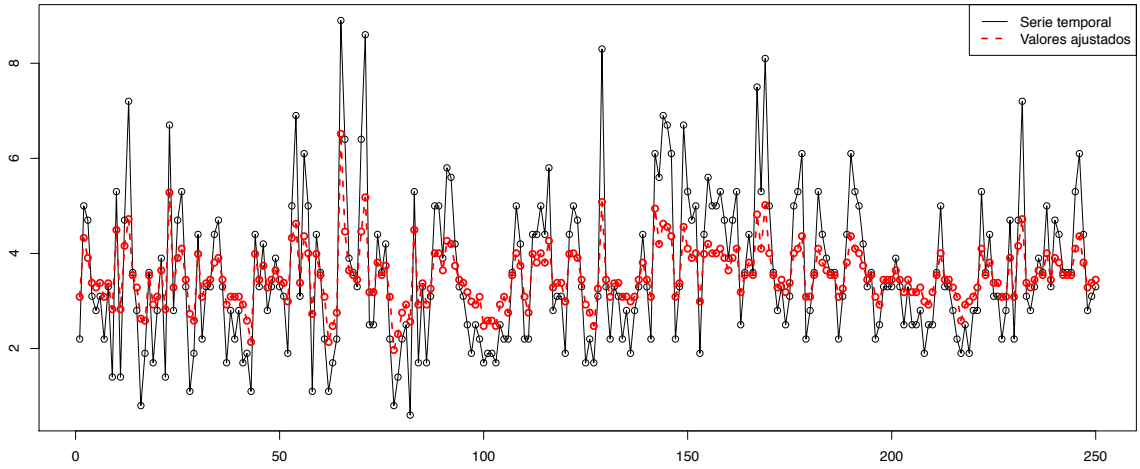

Figura 4.2: Valores reales frente a ajustados.

Las desviaciones típicas residuales estimadas de cada una de las regiones son $\hat{\sigma}_1^2 = 1.677$ y $\hat{\sigma}_2^2 = 1.323$, mientras que la desviación típica residual total es $\hat{\sigma} = 1.429$. En la Figura 4.3 están representados los residuos asociados a este modelo y los p–valores del test de Ljung–Box. Además el p–valor para el test de media 0 es igual a 1.

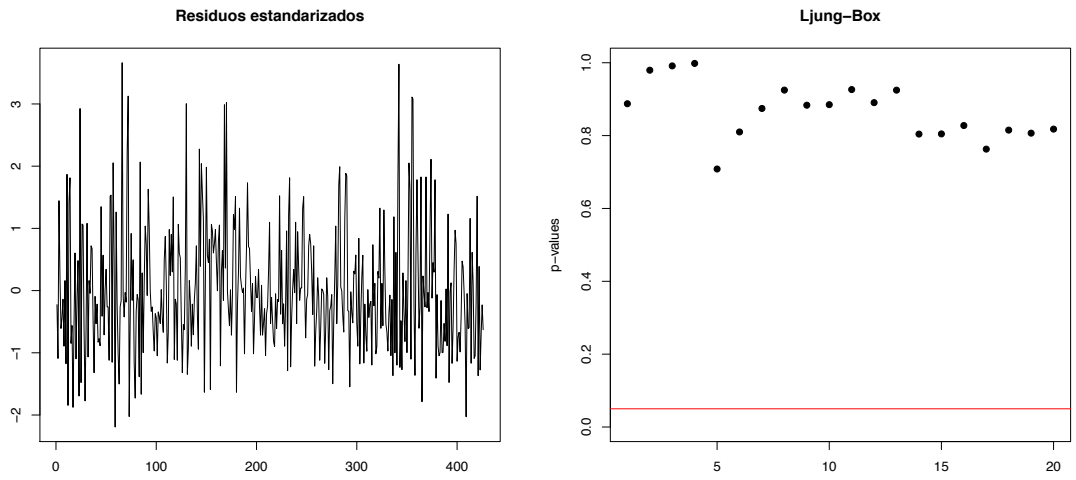

Figura 4.3: Gráfico de residuos y test de Ljung–Box.

Por último se incluye una gráfica en la que se representa los últimos 100 puntos de la serie temporal con la predicción de los 10 puntos siguientes acompañados de sus intervalos de confianza al 95 %

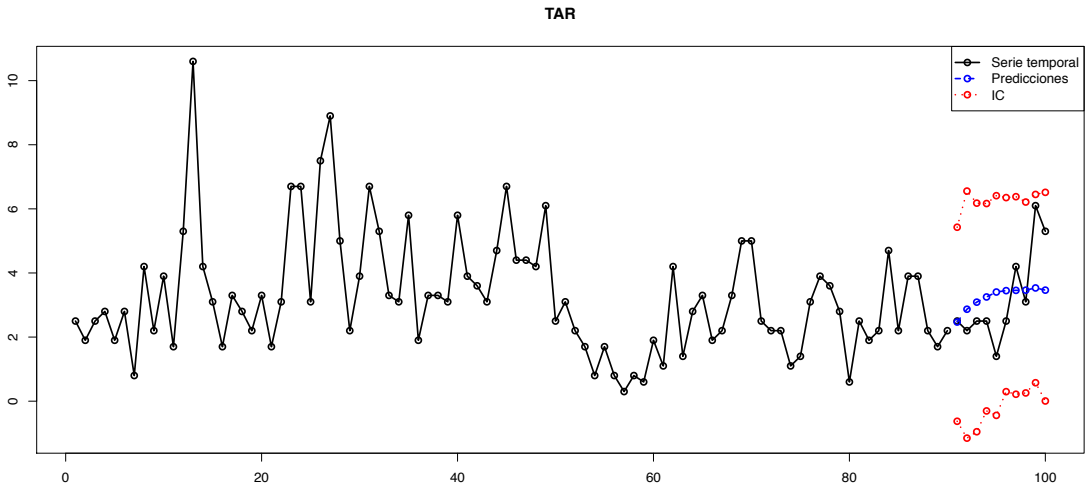

Figura 4.4: Predicciones e intervalos de confianza.

BM

Si intentamos modelizar estos datos mediante un modelo bilinear, obtenemos que, mediante criterio AIC, los mejores órdenes son

p	q	P	Q	AIC
3	1	1	2	730.45
4	1	1	2	746.75
3	1	2	1	747.41
1	1	1	2	747.64

Tabla 4.1: Mejores órdenes según criterio AIC.

Por lo tanto, el mejor modelo bilineal es BL(3,1,1,2). Para estos órdenes, el modelo obtenido sería el siguiente,

$$X_t = 1.3695X_{t-1} - 0.3317X_{t-2} - 0.0378X_{t-3} - 0.9966\epsilon_{t-1} + \\ + 0.0421X_{t-1}\epsilon_{t-1} - 0.0116X_{t-1}\epsilon_{t-2}.$$

En la Figura 4.5 se muestran los valores ajustados de los 250 primeros elementos de la serie según el modelo BL(3,1,1,2) frente a sus valores reales. Se observa que el modelo ajusta más o menos igual que el TAR.

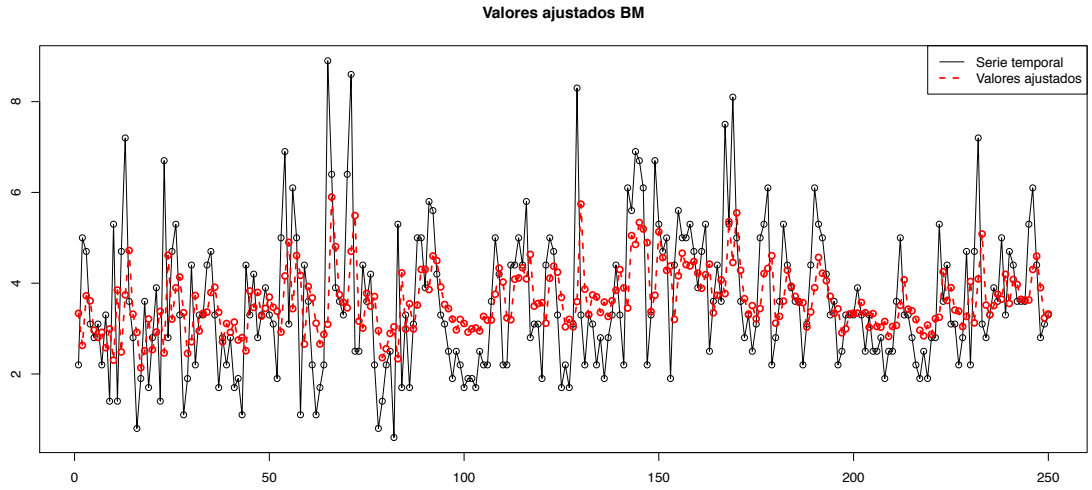

Figura 4.5: Valores reales frente a ajustados.

Los residuos obtenidos de este modelo están representados en la Figura 4.6 junto con los p–valores del test de Ljung–Box. La desviación típica estimada es $\hat{\sigma} = 1.3559$. Por otra parte, se obtuvo un p–valor de 0.8892 para el test de media 0. Pero como se puede observar por el q-q plot, los residuos no son normales. Además, el p–valor para el test de Shapiro–Wilk es 1.09×10^{-8} por lo que este modelo no sería válido, ya que para la estimación de los parámetros se exige normalidad en los residuos.

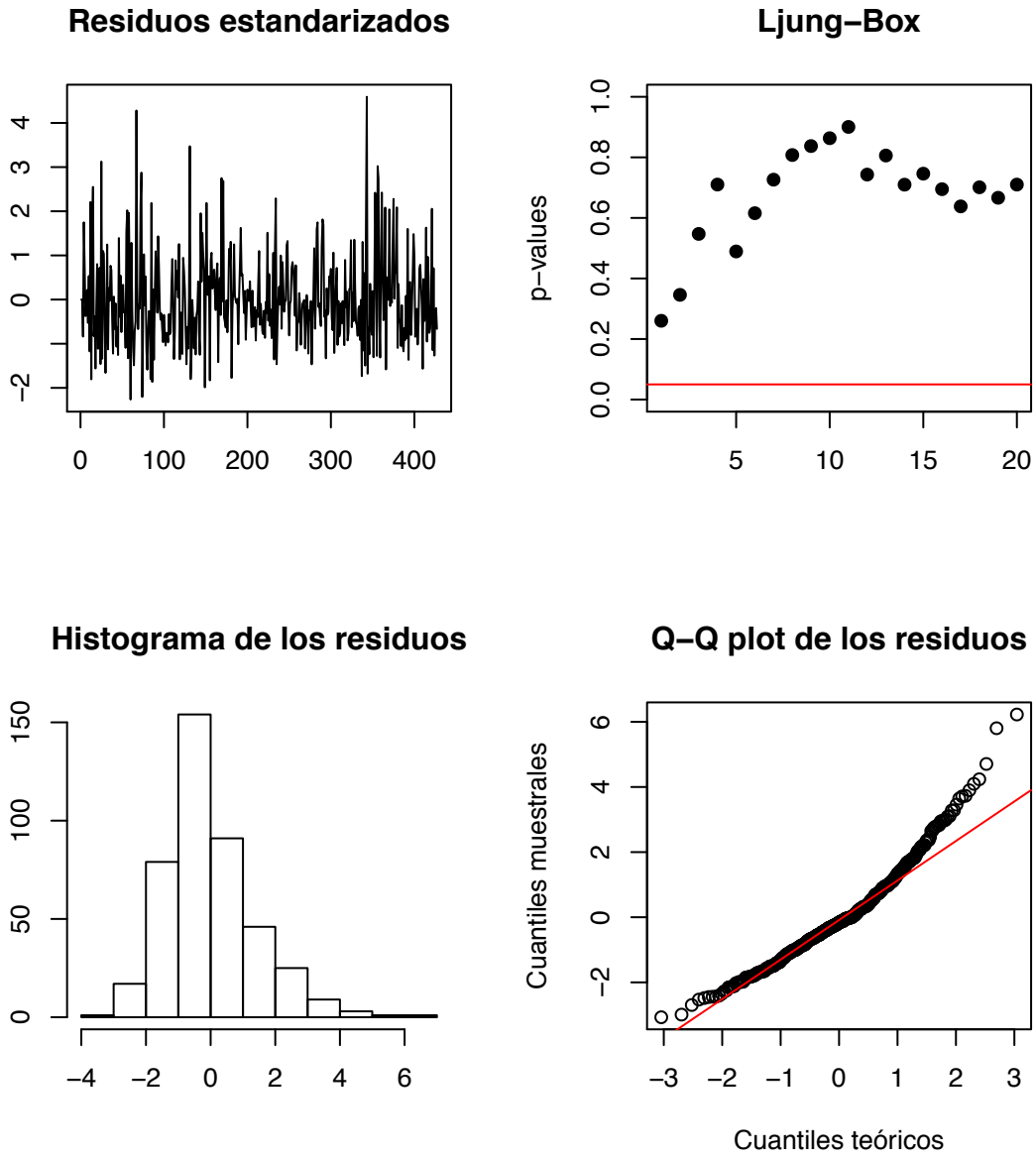

Figura 4.6: Gráficos de residuos.

Para terminar, se representan las predicciones de los 10 siguientes puntos de la serie con sus intervalos de confianza al 95 %. Se ve que el intervalo se va ampliando de manera muy rápida a medida que avanzamos en las predicciones.

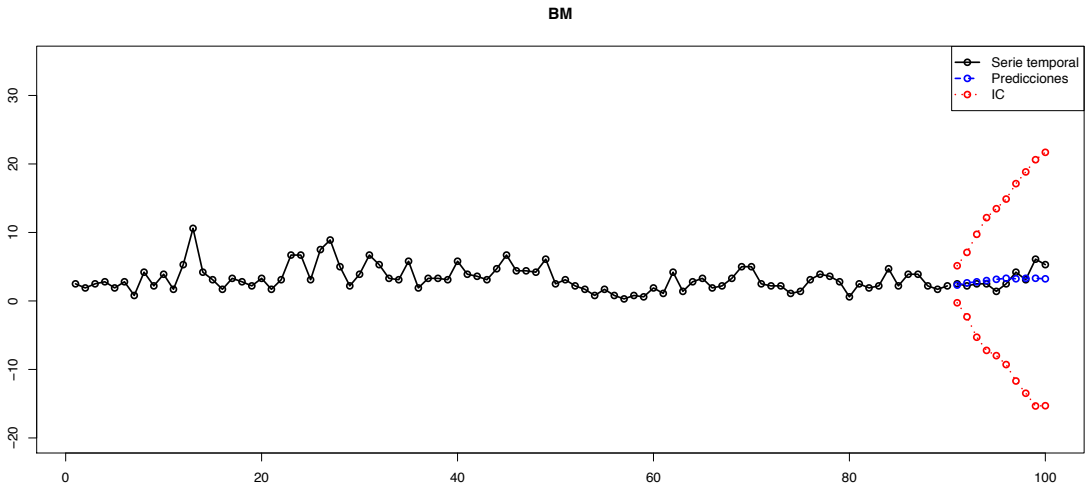

Figura 4.7: Predicciones e intervalos de confianza.

FAR

Para el modelo FAR se calculó el valor del parámetro d para varios valores de p, tomando como posibles valores de h 20 puntos equidistantes en el intervalo [1,3], obteniéndose los siguientes resultados:

p	1	2	3	4	5	6	7	8
d	1	2	2	3	3	5	2	1
APE	2.203	2.055	2.025	2.126	2.153	2.270	2.306	2.343

Tabla 4.2: d óptimo para cada posible modelo.

De todos los modelos estimados, nos decantamos por el FAR(3,2), ya que es el que tiene menor error medio de predicción (APE). Para este modelo el valor estimado del parámetro ventana es $h = 2.3684$. Por tanto, estaremos trabajando con un modelo de la forma

$$X_t = f_1(X_{t-2})X_{t-1} + f_2(X_{t-2})X_{t-2} + f_3(X_{t-2})X_{t-3}.$$

A continuación, se muestran los valores estimados de las funciones f_1, f_2 y f_3

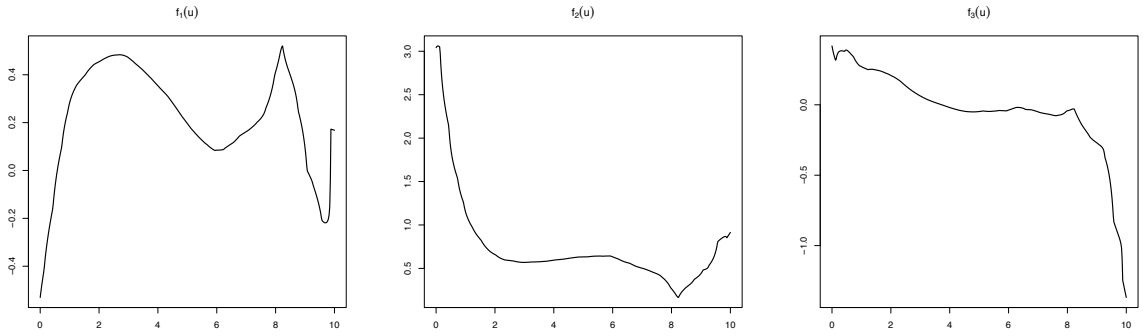

Figura 4.8: Estimaciones de las funciones f_1, f_2 y f_3.

Para este modelo, se calculan los valores ajustados de los 250 elementos de la serie y se representan en la Figura 4.9 frente a los valores reales.

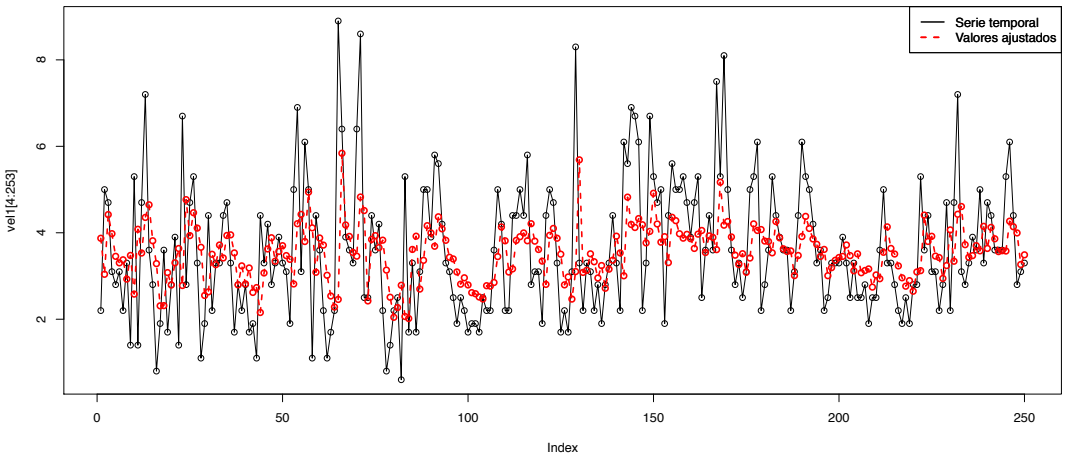

Figura 4.9: Valores reales frente a ajustados.

La representación de los residuos estandarizados y de los p–valores del test de Ljung–Box puede verse en la Figura 4.10. Estos residuos tienen una desviación típica $\hat{\sigma} = 1.3917$. En el test de media cero se obtiene un p–valor de 0.993.

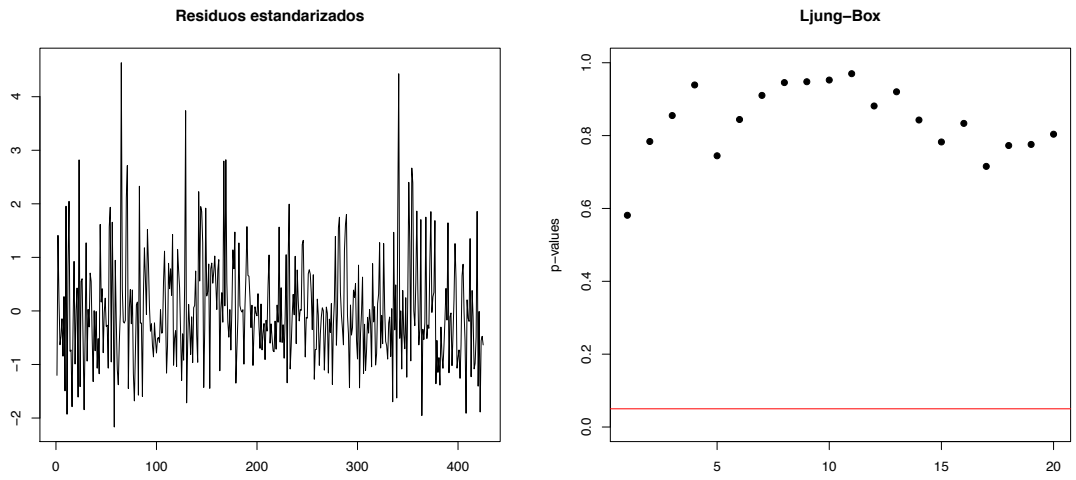

Figura 4.10: Gráfico de residuos y test de Ljung–Box.

La representación de las predicciones de los 10 siguientes puntos según el modelo FAR junto con los intervalos de confianza para las predicciones puede verse en la Figura 4.11.

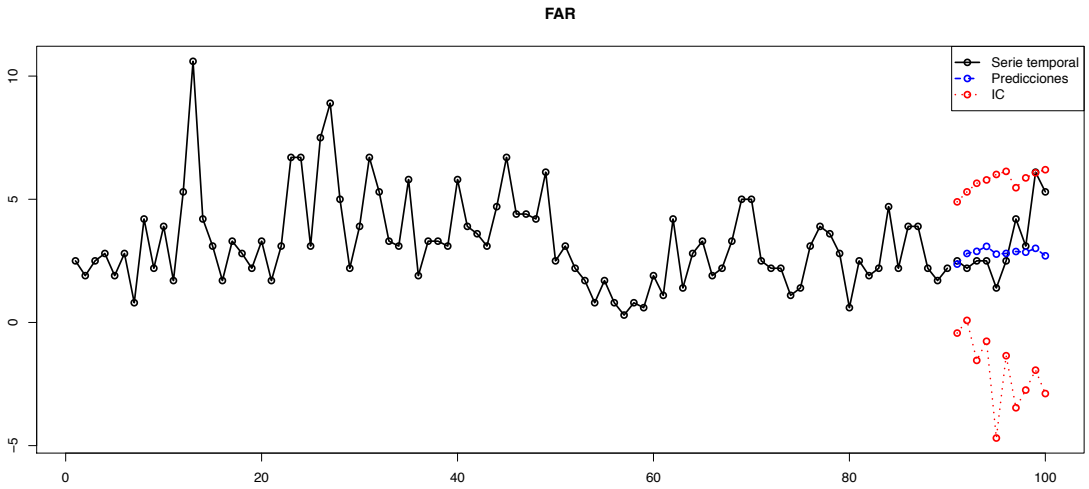

Figura 4.11: Predicciones e intervalos de confianza.

AFAR

En el modelo AFAR para estimar los valores de h y $\boldsymbol{\beta}$ se probó con $p = 2, 3, 4$. Los resultados están recogidos en la siguiente tabla

p	2	3	4
h	3.4349	2.5555	4.3954
β	$(0.999, 0.041)^T$	$(0.951, -0.061, 0.303)^T$	$(0.129, -0.604, 0.357, 0.700)^T$
GCV	1.9895	2.0018	2.0243

Tabla 4.3: h y β óptimosβ para cada posible modelo.

Escogemos el modelo que tiene un menor GCV, es decir, nos quedamos con un modelo AFAR de la forma

$$X_t = g_0(\boldsymbol{\beta}^T \mathbf{X}_{t-1}) + g_1(\boldsymbol{\beta}^T \mathbf{X}_{t-1}) X_{t-1}$$

siendo $\mathbf{X}_{t-1} = (X_{t-1}, X_{t-2})$ y $\boldsymbol{\beta} = (0.999, 0.041)$, con $h = 3.4349$.

En la Figura 4.12 están representadas las estinaciones de g_0 y g_1. La función $\hat{g}_0$ toma valores en el intervalo $(-90.6, 6.8)$ mietras que la función $\hat{g}_1$ toma valores en $(-5.8, 9.8)$.

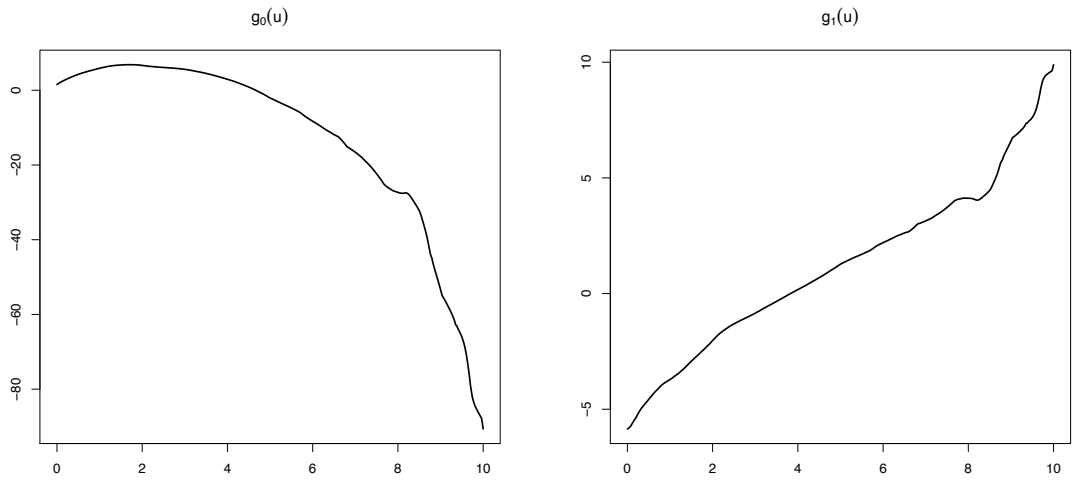

Figura 4.12: Estimaciones de las funciones g_0, g_1.

Para este modelo se calculan los valores ajustados del modelo para los 250 primeros puntos de la serie obteniéndose como resultado lo mostrado en la Figura 4.13.

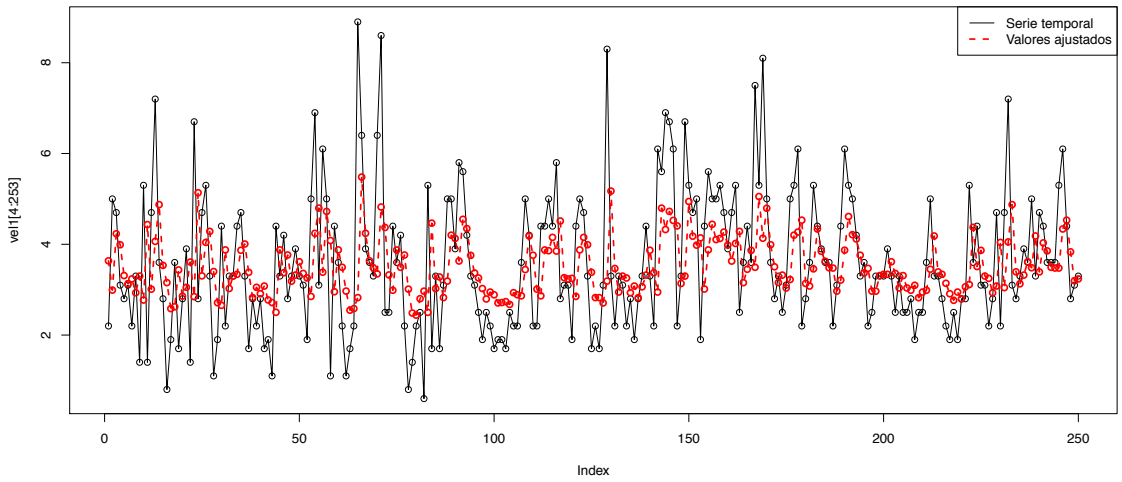

Figura 4.13: Valores reales frente a ajustados.

Como en todos los casos anteriores, se muestra el gráfico de residuos y los p–valores del test de Ljung–Box asociado a este modelo. Estos residuos tienen una desviación típica estimada de $\hat{\sigma} = 1.3917$. El p–valor para el test de media cero es 0.993.

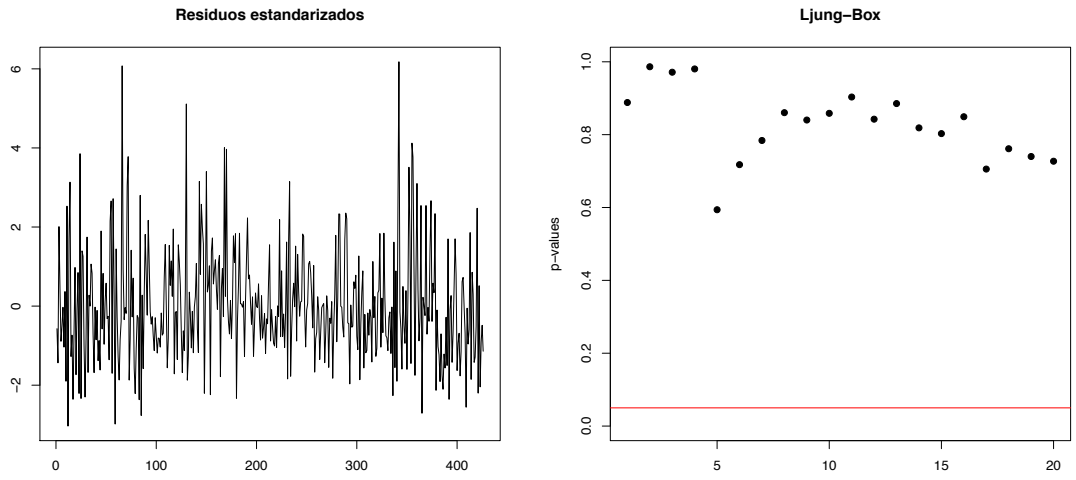

Figura 4.14: Gráfico de residuos y test de Ljung–Box.

Finalmente, se presentan las predicciones de los 10 siguientes valores de la serie temporal acompañados de los intervalos de confianza bootstrap al 95 %.

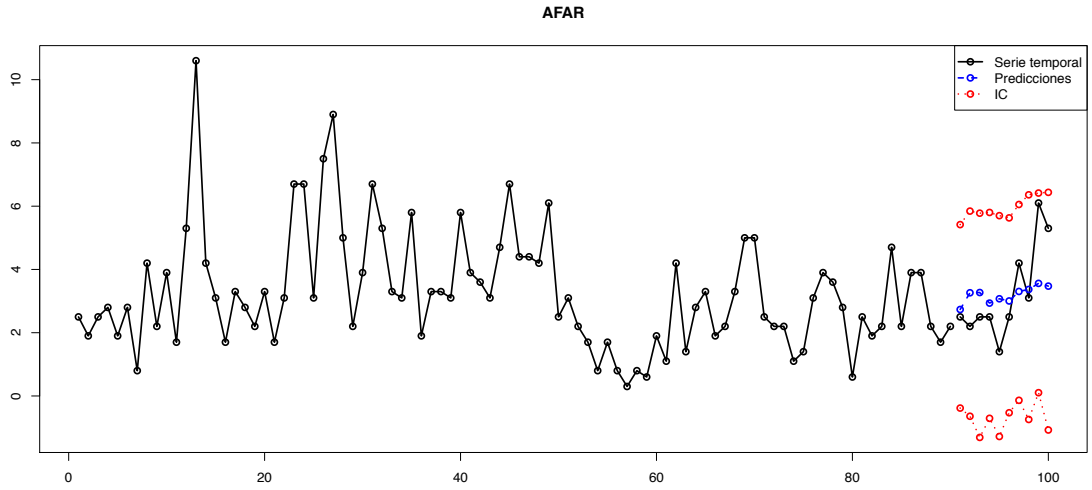

Figura 4.15: Predicciones e intervalos de confianza.

Comparación de modelos

Un criterio para decidir cual de estos modelos es el mejor para emplear con estos datos, es comparar sus errores de predicción. Para esto predecimos los 10 siguientes valores de la serie y calculamos la media de los errores en valor absoluto. Los resultados se muestran en la siguiente tabla,

	TAR	BM	FAR	AFAR
Error	1.08362	1.04434	1.03467	1.17046

Tabla 4.4: Error medio de predicción.

Por tanto, el modelo que comete un menor error a la hora de predecir valores futuros es en FAR.

En cuanto a los intervalos de confianza, se puede ver claramente que, en el caso del modelo bilineal, la longitud de los intervalos crece muy rápidamente a medida que avanzamos en las predicciones. Esto no pasa en el resto de modelos, en los que parece que la longitud de los intervalos es más estable.

4.2. Índice de producción industrial de EEUU

A continuación analizaremos otra base de datos. En este caso trabajaremos con la serie temporal del índice de producción industrial en Estados Unidos. Los datos fueron tomados mensualmente desde enero de 1966 hasta mayo de 1999. Esta base de datos se encuentra en el paquete `fEcofin` de **R** bajo el nombre `IP.dat`. En este paquete, además se encuentran otras muchas bases de datos de series temporales financieras.

Como se puede observar en la Figura 4.16, esta serie presenta tendencia. Para suprimir esta tendencia, procederemos a diferenciarla con retardo 1. El resultado puede verse también en la Figura 4.16.

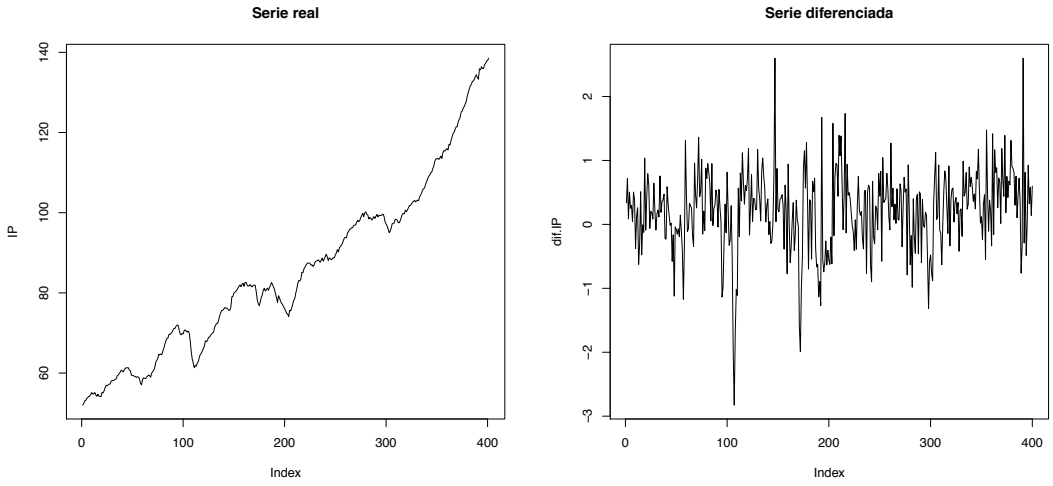

Figura 4.16: Series real y diferenciada.

Durante todo este apartado, trabajaremos con la serie diferenciada para eliminar la tendencia.

TAR

Comencemos tratando de modelizar estos datos según un prodeso TAR. Mediante criterio AIC la partición de la recta real que mejor funciona para esta base de datos fue $\{(\infty, -0.221], (-0.221, \infty)\}$ con $d = 2$. Se probó también con otros valores de d obteniéndose peores resultados.

El modelo obtenido una vez fueron estimados los parámetros fue

$$X_t = \begin{cases} -0.0027 + 0.3279X_{t-1} & X_{t-2} \leq -0.221 \\ 0.2050 + 0.2492X_{t-1} & X_{t-2} > -0.221 \end{cases} \tag{4.2}$$

En este caso, el número de observaciones en cada región es 82 y 318 respectivamente. En la Figura 4.17 se muestan los 250 primeros valores de la serie temporal frente a sus valores ajustados por el modelo TAR estimado.

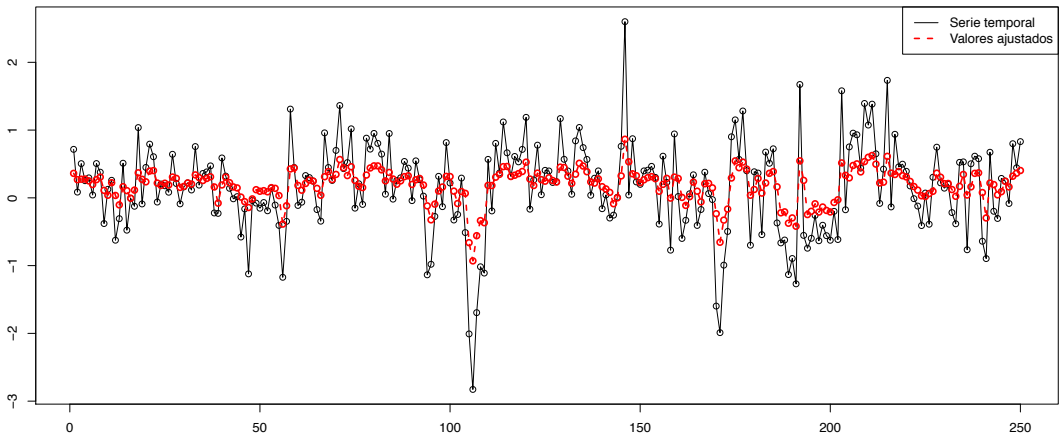

Figura 4.17: Valores ajustados.

Se procede a realizar un estudio de los residuos del modelo. En la Figura 4.18 se tiene la representación de los residuos estandarizados y los p–valores del test de Ljung–Box. Además el p–valores para el test de media cero es 1, por lo que los residuos pasan el contraste.

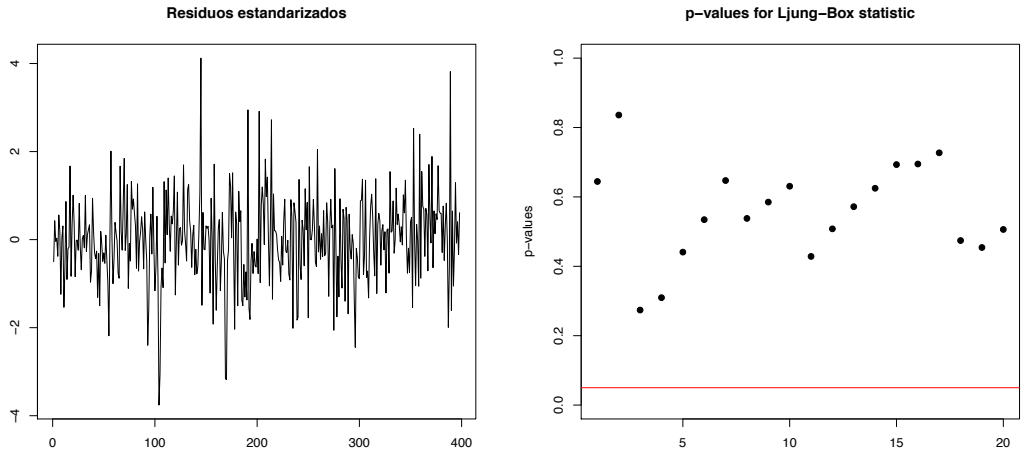

Figura 4.18: Gráfico de residuos y test de Ljung–Box.

A continuación se representa la predicción de los siguientes 12 datos de la serie temporal y sus intervalos de confianza puntuales al 95 %.

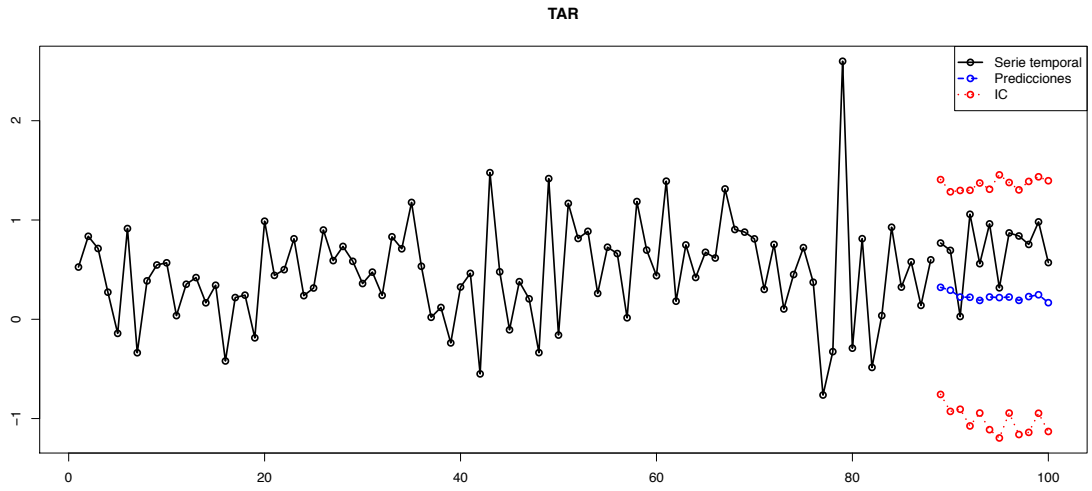

Figura 4.19: Predicciones e intervalos de confianza.

BM

Para ajustar estos datos mediante un modelo bilineal, se comienza por buscar los órdenes óptimos mediante criterio AIC. Éstos se muestran en la Tabla 4.5

p	q	P	Q	AIC
2	1	2	2	-29.88
1	2	2	2	-29.03
2	2	2	2	-26.56
3	2	2	2	-23.86

Tabla 4.5: Mejores órdenes según criterio AIC.

Por lo tanto, el mejor modelo bilineal es BL(2,1,2,2). Para estos órdenes, el modelo obtenido sería el siguiente,

$$X_t = 1.1629X_{t-1} - 0.2069X_{t-2} - 0.8669\epsilon_{t-1} - 0.1795X_{t-1}\epsilon_{t-1} + \\ + 0.0623X_{t-1}\epsilon_{t-2} - 0.0046X_{t-2}\epsilon_{t-1} + 0.2056X_{t-2}\epsilon_{t-2}.$$

En la Figura 4.20 se muestran los valores ajustados de los 250 primeros elementos de la serie según el modelo BL(2,1,2,2) frente a sus valores reales. El modelo se ajusta más o menos igual que el TAR.

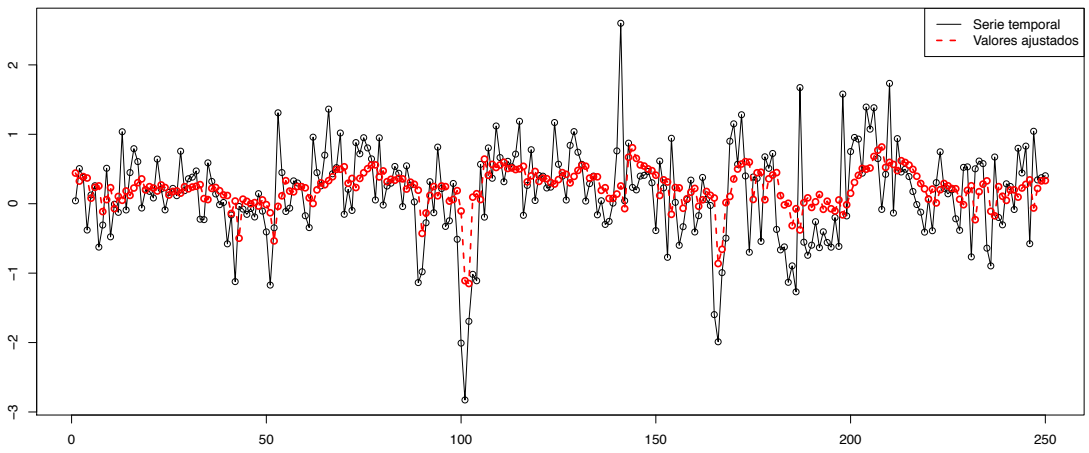

Figura 4.20: Valores ajustados frente a reales.

En cuanto a los residuos, en la Figura 4.21 se pueden ver su representación gráfica y la de los p–valores del test de Ljung–Box. Por otra parte, el p–valor para el test de media cero es 0.3017.

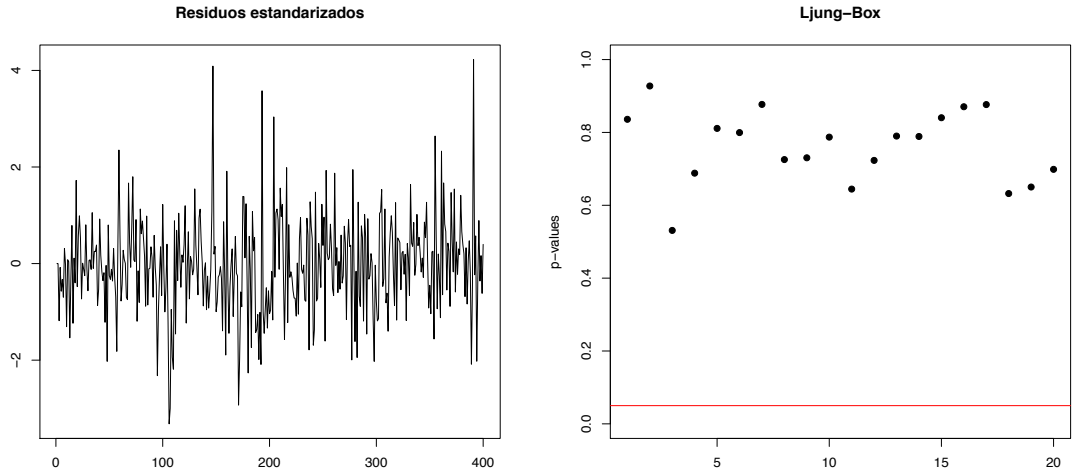

Figura 4.21: Gráfico de residuos y test de Ljung–Box.

Sin embargo, los residuos no son normales, como puede verse en la Figura 4.22. Además el p–valor para el test de Shapiro–Wilk es 2.2×10^{-4}, por lo que este modelo no sería

valido ya que la estimación de los parámetros se hace bajo la hipótesis de que los residuos sean variables aleatorias normales.

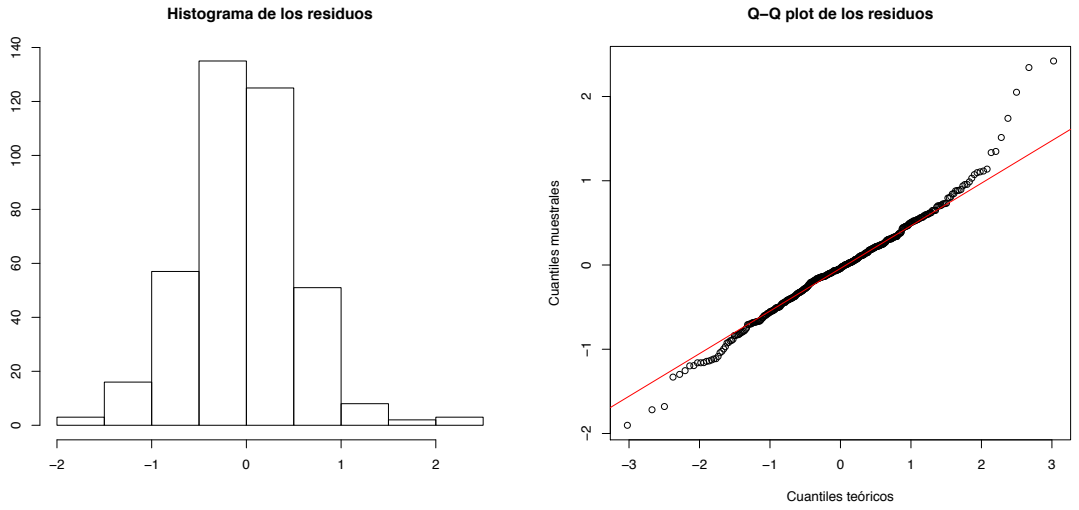

Figura 4.22: Histograma de los reesiduos y Q–Q plot.

Por último, se representa la predicción de los siguientes 12 datos de la serie temporal y sus intervalos de confianza puntuales al 95 % aunque, como ya se comentó con anterioridad, este modelo no sería válido.

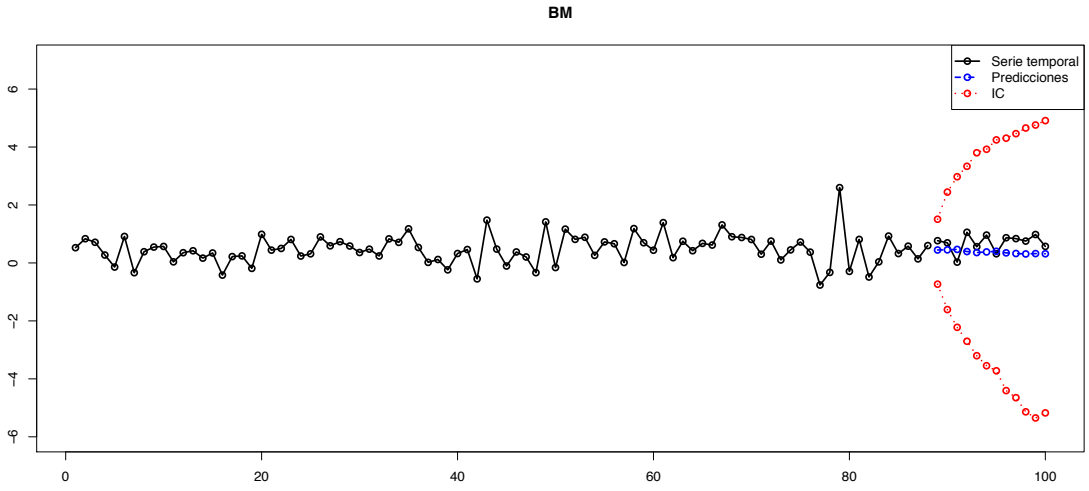

Figura 4.23: Predicciones e intervalos de confianza.

FAR

Para el modelo FAR se estimó el valor óptimo de d para distintos valores de p y tomando como posibles h una rejilla de 21 puntos equidistantes en el intervalo [0.15,2], obteniéndose los siguientes resultados:

p	1	2	3	4	5	6	7
d	1	1	1	4	5	5	5
APE	0.3407	0.3281	0.3197	0.3212	0.3084	0.3065	0.3091

Tabla 4.6: d óptimo para cada posible modelo.

Por tanto, el modelo FAR que mejor se ajusta a nuestros datos es el FAR(6,5), ya que es el que tiene un menor APE. Para este modelo, valor el h óptimo estimado es de 1.63. Por lo que estaremos trabajando con un modelo de la forma

$$X_t = \sum_{j=1}^{6} f_j(X_{t-5})X_{t-j}$$

La estimación de las funciones f_j puede verse en la Figura 4.24

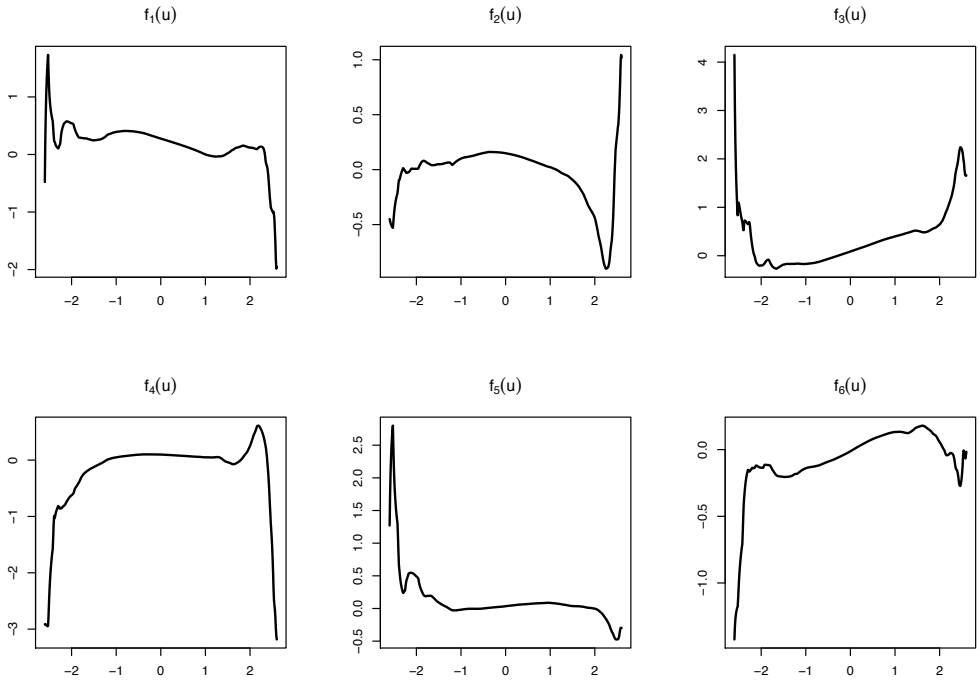

Figura 4.24: Estimaciones de las funciones f_j.

Para este modelo, se calculan los valores ajustados de los 250 primeros puntos de la serie.

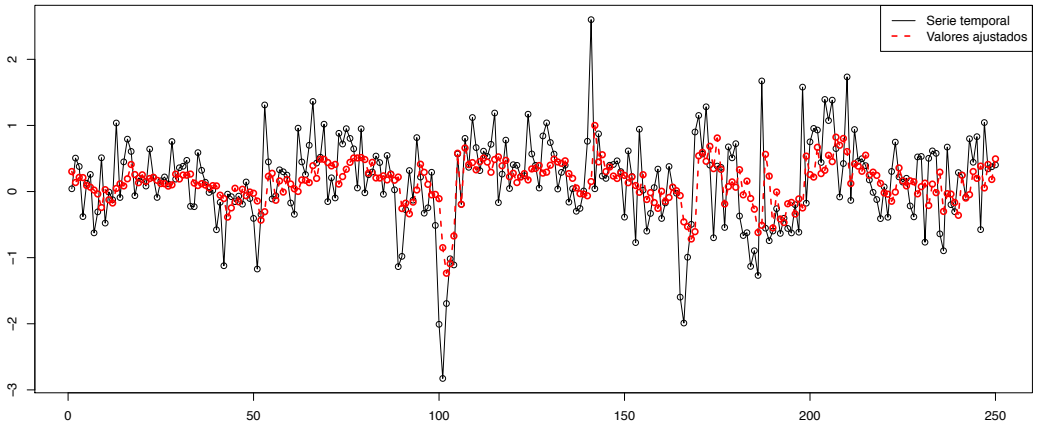

Figura 4.25: Valores ajustados frente a reales.

Por último, en el estudio de residuos, el p–valor del test de media cero es 0.2258. Además, en el gráfico siguiente se representan los residuos estandarizados y los p–valores para el test de Ljung–Box.

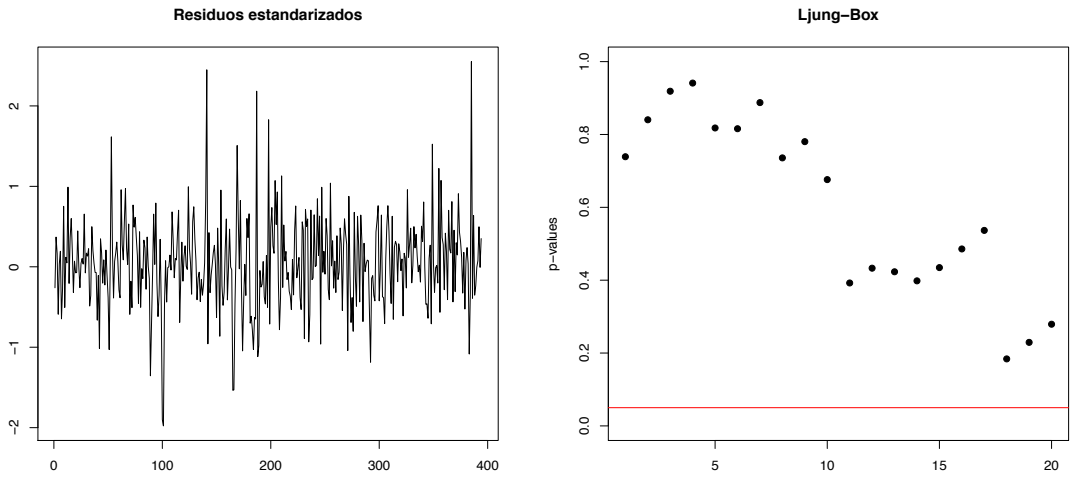

Figura 4.26: Gráfico de residuos y test de Ljung–Box.

Finalmente, se presentan las predicciones de los 12 siguientes valores de la serie temporal acompañados de los intervalos de confianza bootstrap al 95 %.

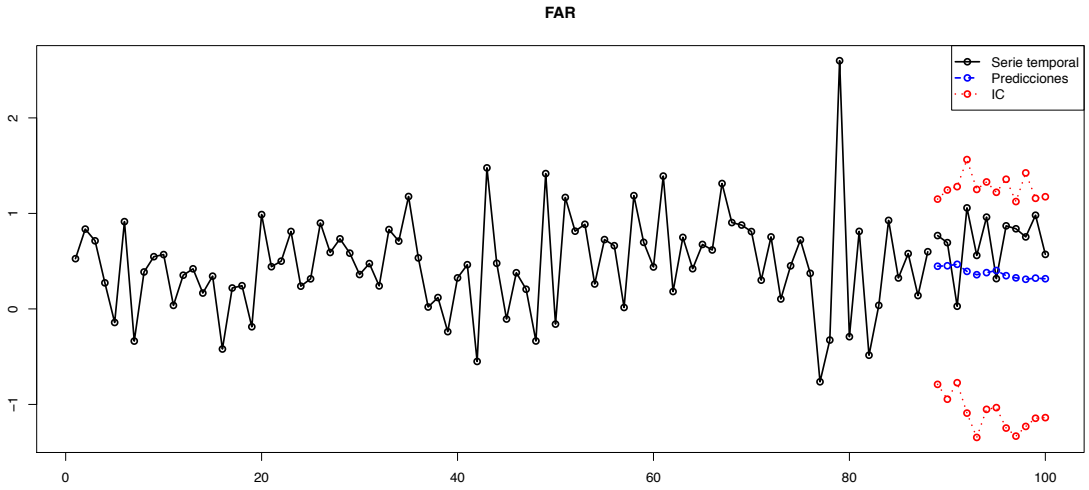

Figura 4.27: Predicciones e intervalos de confianza.

AFAR

Para el modelo AFAR se estimaron los valores de h y $\boldsymbol{\beta}$ para $p = 2, 3, 4$. Los resultados están recogidos en la siguiente tabla.

p	2	3	4
h	0.1511	2.6685	2.0121
β	$(0.405, 0.914)^T$	$(0.844, 0.5, -0.191)^T$	$(0.387, -0.667, 0.624, 0.118)^T$
GCV	0.3607	0.3347	0.3436

Tabla 4.7: h y β óptimos para cada posible modelo.

Seleccionamos pues el modelo con $p = 3$ ya que es el que minimiza el GCV. El modelo quedaría de la forma

$$X_t = g_0(\boldsymbol{\beta}^T \mathbf{X}_{t-1}) + g_1(\boldsymbol{\beta}^T \mathbf{X}_{t-1}) X_{t-1} + g_2(\boldsymbol{\beta}^T \mathbf{X}_{t-1}) X_{t-2}$$

siendo $\mathbf{X}_{t-1} = (X_{t-1}, X_{t-2}, X_{t-3})$ y $\boldsymbol{\beta} = (0.844, 0.5, -0.191)$, con $h = 2.6685$.

En la Figura 4.28 están representadas las estimaciones de las funciones g_j.

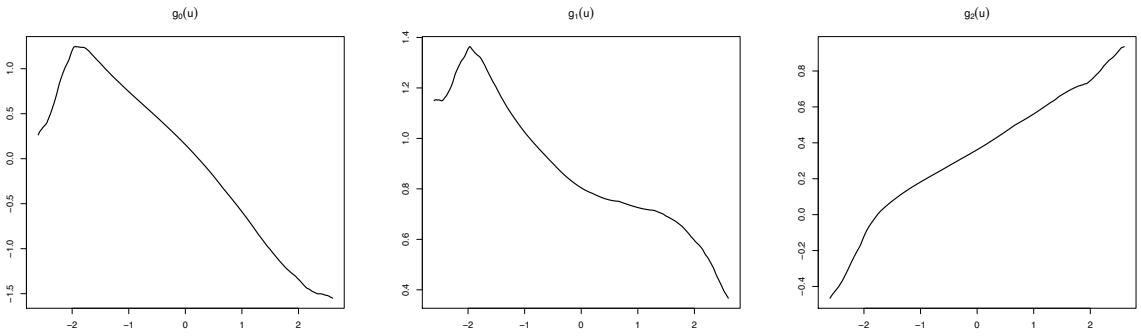

Figura 4.28: Estimaciones de las funciones g_0, g_1 y g_2.

Para este modelo se calculan los valores ajustados del modelo para los 250 primeros puntos de la serie obteniéndose como resultado lo mostrado en la Figura 4.29.

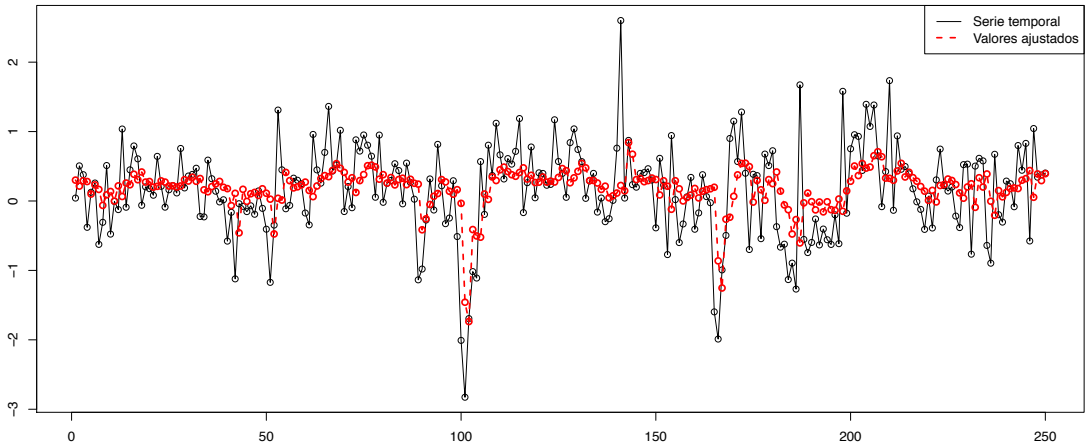

Figura 4.29: Valores reales frente a ajustados.

Como en todos los casos anteriores, se muestra el gráfico de residuos y los p–valores del test de Ljung–Box asociado a este modelo. Estos residuos tienen una desviación típica estimada de $\hat{\sigma} = 1.3917$. El p–valor para el test de media cero es 0.9482.

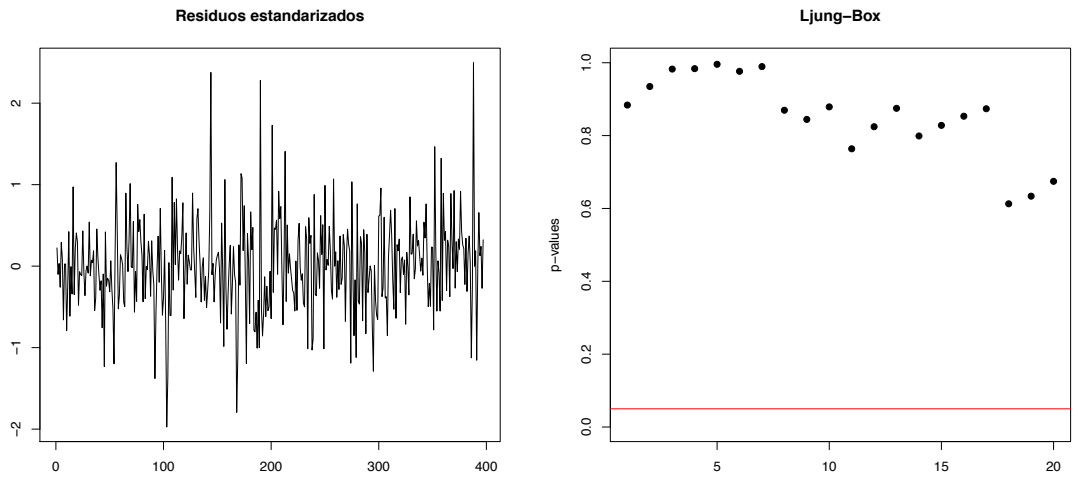

Figura 4.30: Gráfico de residuos y test de Ljung–Box.

Por último se incluye una gráfica en la que se representa los últimos 100 puntos de la serie temporal con la predicción de los 12 puntos siguientes acompañados de sus intervalos de confianza al 95 %

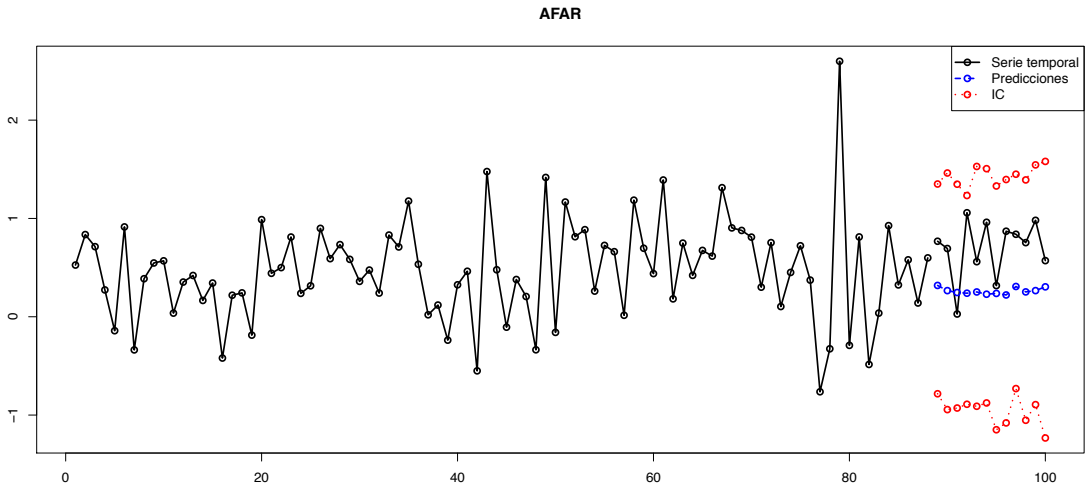

Figura 4.31: Predicción e intervalos de confianza.

Comparación de modelos

Un criterio para decidir cual de estos modelos es el mejor para emplear con estos datos, es comparar sus errores de predicción. Para esto predecimos los 12 siguientes valores de la serie y calculamos la media de los errores en valor absoluto. Los resultados se muestran en la siguiente tabla,

	TAR	BM	FAR	AFAR
Error	0.49247	0.37490	0.55783	0.47464

Tabla 4.8: Error medio de predicción.

Por tanto, el modelo que comete un menor error a la hora de predecir valores futuros es el bilineal, pero, como ya vimos antes, este modelo no es válido por no cumplir la hipótesis de normalidad de los residuos. Luego el mejor modelo sería el AFAR.

En cuanto a los intervalos de confianza bootstrap, al igual que con los datos de viento, el modelo bilineal es el que ofrece unos intervalos con mayor longitud, mientras que en el resto de modelos tienen una longitud similar.

Apéndice A

Código desarrollado

En este anexo se muestran todas las funciones de programación propia utilizadas para la estimación y predicción de los modelos tratados en este documento.

TAR

Comenzamos con la funciones programadas para el modelo TAR.

La función `tar` estima todos los parámetros del modelo y calcula los residuos y los valores ajustados por el modelo. Entre sus argumentos, cabe destacar que la partición será introducida en forma de vector (p.e. `part=c(1,2)` se referirá a la partición $(-\infty, 1] \cup (1, 2] \cup (2, \infty)$). El código es el siguiente:

```
tar=function(x,d,part,ord.part,include.mean=TRUE){
    part=c(-Inf,part,Inf)
    k=length(part)
    n=length(x)
    p=max(ord.part)
    res=Inf
    dat<-cbind(x)
    h=max(d,ord.part)
    for(l in 1:h){
        dat=cbind(dat,cbind(c(x[-c(1:l)],rep(NA,l))))
    }
    L=numeric(k-1)
    coeff=list()
    sigma=numeric(k-1)
    Ti=numeric(k-1)
    resid1=0
    resid2=0
```

```
FIT=0
for (i in 1:(k-1)){
    if (ord.part[i]==0) {
        xnew=x[(d+1):n]
        dat2=xnew[part[i]<x[1:(n-d)] & x[1:(n-d)]<=part[i+1]]
        label=1:(length(x)-d)
        names(dat2)=label[part[i]<x[1:(n-d)] & x[1:(n-d)]<=part[i+1]]
        L[i]=sum((dat2-mean(dat2))^2,na.rm =TRUE)
        coeff[[i]]=mean(dat2)
        names(coeff[[i]])="Intercept"
        Ti[i]=length(dat2)
        sigma[i]=sqrt(mean((dat2-mean(dat2))^2))
        resid1=c(resid1,(dat2-mean(dat2)/sigma[i]))
        resid2=c(resid2,dat2-mean(dat2))
        fit=rep(mean(dat2),length(dat2))
        names(fit)=label[part[i]<x[1:(n-d)] & x[1:(n-d)]<=part[i+1]]
        FIT=c(FIT,fit)
    }
    else{
        name=c("X")
        for (j in 1:(ord.part[i])) name=c(name, paste("X",j,sep=""))
        dat1=cbind(dat[,d+1],dat[,h:(h-ord.part[i]+1)])
        dat1=dat[,(h+1):(h-ord.part[i]+1)]
        dat1=data.frame(dat1)
        names(dat1)=name
        fit=lm(data=dat1[part[i]<dat[,h-d+1] & dat[,h-d+1]<=part[i+1],])
        L[i]=sum(residuals(fit)^2,na.rm =TRUE)
        coeff[[i]]=coef(fit)
        Ti[i]=length(residuals(fit))
        sigma[i]=sqrt(mean(residuals(fit)^2))
        resid1=c(resid1,residuals(fit)/sigma[i])
        resid2=c(resid2,residuals(fit))
        FIT=c(FIT,fit$fitted.values)
    }
}
ind1=sort(as.numeric(names(resid1)),index.return=TRUE)$ix
resid1=resid1[-1]
residual1=resid1[ind1]
ind2=sort(as.numeric(names(resid2)),index.return=TRUE)$ix
resid2=resid2[-1]
residual2=resid2[ind2]
FIT=FIT[-1]
```

```
    fitted.values=FIT[ind1]
    Lik<-sum(L)
    fit=list("coeff"=coeff,"d"=d, "Lik"=Lik,"fitted.values"=fitted.values,
        "res"=residual2,"std.res"=residual1,"sigma"=sigma,"Ti"=Ti,
        "x"=x,"ord.part"=ord.part,"part"=part)
    class(fit)<- "tar"
    return(fit)
}
```

La segunda función de este modelo es la función `best.tar`. Esta función devuelve una matriz con los mejores órdenes hasta distancia `dist.max.crit`. Se puede escoger entre los métodos AIC, AICC o BIC:

```
best.tar=function(x,ord.max=3,d=1,part=0,crit="AIC",dist.max.crit=2){
    n=length(part)+1
    T=length(x)
    perm=permutations(ord.max+1,n,v=0:ord.max,repeats.allowed=TRUE)
    P=dim(perm)[1]
    if (crit=="AIC")  factor=2
    else if (crit=="BIC")  factor=log(T)
    pb=txtProgressBar(style=3)
    iind = 0
    nit=dim(perm)[2]
    crit.val=rbind(rep(0,n+1))
    for (i in 1:P){
        setTxtProgressBar(pb, iind/nit)
        fit=tar(x=x,d=d,part=part,ord.part=perm[i,])
        if (crit=="AICC")  factor=2*T/(T-(sum(perm[i,])+dim(perm)[1])-2)
        fit.crit=sum(log((fit$sigma)^2))+factor*(sum(perm[i,])+dim(perm)[1])
        if(res.test(fit$res)$mu>=0.05 & sum(res.test(fit$res)$ind>=0.05)==20){
            crit.val=rbind(crit.val,c(perm[i,], fit.crit))
        }
        iind=iind+1
    }
    if(dim(crit.val)[1]==2){max.dist=crit.val[-1,]}
    else{
        crit.val=data.frame(crit.val[-1,])
        min.dist=crit.val[,n+1]-min(crit.val[,n+1])
        row.ok=rep(FALSE,length=length(min.dist))
        row.ok[min.dist<= dist.max.crit & min.dist>=0]=TRUE
        crit.val.ok=crit.val[row.ok,]
```

```
        max.dist=crit.val.ok[order(crit.val.ok[,n+1]),]
        max.dist=data.frame(max.dist)
    }
    param=0
    for(j in 1:n){param=c(param,paste("P",j,sep=""))}
    param=param[-1]
if (crit=="AIC") names(max.dist)=c(param, "AIC")
else if (crit=="AICC") names(max.dist)=c(param, "AICC")
else names(max.dist)=c(param, "BIC")
setTxtProgressBar(pb, 1)
cat("\n")
return(list("max.dist"=max.dist))
}
```

Por último, para este modelo, tenemos la función `predict.tar` a la que se le pasa como argumentos el elemento de la clase "tar" obtenido de la función `tar`, el nivel de significación para los intervalos de confianza y el número máximo de valores a predecir. El código es:

```
predict.tar=function(data,alpha=0.05,n.ahead=5){
    K=length(data$part)
    d=data$d
    mat.pred=matrix(0,nrow=500,ncol=n.ahead)
    for(k in 1:500){
        x=data$x
        n=length(x)
        for(i in 1:n.ahead){
        for (j in 1:(K-1)){
            if(data$part[j]<x[n-d+1] & x[n-d+1]<=data$part[j+1]){
                if(data$ord.part[j]==0) {mat.pred[k,i]=data$coeff[[j]][1]
                    +rnorm(1,mean=0,sd=data$sigma[j])}
                else{
                    mat.pred[k,i]=data$coeff[[j]][1]
                    +data$coeff[[j]][-1]%*%x[n:(n-data$ord.part[j]+1)]
                    +rnorm(1,mean=0,sd=data$sigma[j])}}}
            x=c(x,mat.pred[k,i])
            n=n+1
        }
    }
    dat.mean=apply(mat.pred,2,mean)
    dat.sd=apply(mat.pred,2,sd)
```

```
    mat.pred.ord=apply(mat.pred,2,sort)
    quantili=mat.pred.ord[floor(dim(mat.pred)[1]*(alpha/2)),]
    quantils=mat.pred.ord[floor(dim(mat.pred)[1]*(1-(alpha/2))),]
    CI=cbind(quantili,quantils)
    fit=list("pred"=dat.mean,"se"=dat.sd,"mat.pred"=mat.pred,
    "quantili"=quantili,"quantils"=quantils,"CI"=CI)
    return(fit)
}
```

BM

La primera función para el modelo bilineal es `bm`. Es la encargada de estimar los parámetros del modelo y calcular los valores ajustados y los residuos. Es la función más extensa, debido a que hay que diferenciar varios casos a la hora de la estimación de los parámetros. Esta función llama a su vez a otras dos que son `loglik` y `res.bm`. La primera es la función a minimizar para obtener las estimaciones de los parámetros mientras que la segunda calcula los residuos y los valores ajustados. Ambas funciones son también de programación propia, pero su código se omite para no extender demasiado este apéndice.

```
bm=function(x,ord.param,method){
    p=ord.param[1]
    q=ord.param[2]
    P=ord.param[3]
    Q=ord.param[4]
    pP=max(p,P)
    qQ=max(q,Q)
    N=length(x)
    eps=numeric()
    eps[1:qQ]=0
    dat=cbind(x)
    for (j in 1:pP){
        dat=cbind(dat,cbind(c(x[-c(1:j)],rep(NA,j))))
    }
    if(P==0|Q==0){
        fit=arima(x,order = c(p,0,q),include.mean=FALSE)
        A=opt$par[(p+1):(p+q)]
        B=fit$coef[1:p]
        Sigma=sqrt(fit$sigma)
        res=fit$residuals
        log.lik=fit$loglik
        res=list("a"=A,"b"=B,"sigma"=Sigma,"res"=res,"loglik"=log.lik,
            "dat"=dat,"fitted.values"=fit$fitted.values,
            "método"=method,"ord.param"=ord.param,"x"=x)
    }else{
        if(p==0 && q!=0){
            logl1=function(y){
                a=y[1:q]
                b=0
                c=y[(p+q+1):(p+q+P*Q)]
                sigma=y[p+q+P*Q+1]
                loglik(a,b,c,sigma,ord.param,x=dat)
```

```
        }
        fit=arima(x, order=c(0,0,q),include.mean=FALSE,
            optim.control=list(maxit=500))
        par=c(fit$coef[1:q],rep(0,P*Q),fit$sigma)
        opt=optim(par,logl1,method=method)
        A=opt$par[1:q]
        C=opt$par[(p+q+1):(p+q+P*Q)]
        Sigma=opt$par[p+q+P*Q+1]
        fit=res.bm(x=dat,ord.param=ord.param,a=A,b=0,c=C,sigma=Sigma)
        res=fit$eps
        fitted.values=fit$fitted.values
        log.lik=-loglik(A,0,C,Sigma,ord.param,dat)
        res=list("a"=A,"b"=0,"c"=C,"sigma"=Sigma,"res"=res,
            "loglik"=log.lik,"fitted.values"=fitted.values,"dat"=dat,
            "método"=method,"ord.param"=ord.param,"x"=x)
    }
    if(p!=0 && q==0){
        logl2=function(y){
            a=0
            b=y[1:p]
            c=y[(p+q+1):(p+q+P*Q)]
            sigma=y[p+q+P*Q+1]
            loglik(a,b,c,sigma,ord.param,x=dat)
        }
        fit=arima(x, order=c(p,0,0),include.mean=FALSE,
            optim.control=list(maxit=500))
        par=c(fit$coef[1:p],rep(0,P*Q),fit$sigma)
        opt=optim(par,logl2,method=method)
        B=opt$par[1:p]
        C=opt$par[(p+q+1):(p+q+P*Q)]
        Sigma=opt$par[p+q+P*Q+1]
        fit=res.bm(x=dat,ord.param=ord.param,a=0,b=B,c=C,sigma=Sigma)
        res=fit$eps
        fitted.values=fit$fitted.values
        log.lik=-loglik(0,B,C,Sigma,ord.param,dat)
        res=list("a"=0,"b"=B,"c"=C,"sigma"=Sigma,"res"=res,
            "loglik"=log.lik,"fitted.values"=fitted.values,
            "dat"=dat,"método"=method,"ord.param"=ord.param,"x"=x)
    }
    if(p==0 && q==0){
        logl2=function(y){
            a=0
```

```
                b=0
                c=y[1:(P*Q)]
                sigma=y[P*Q+1]
                loglik(a,b,c,sigma,ord.param,x=dat)
            }
            fit=arima(x, order=c(0,0,0),include.mean=FALSE,
                optim.control=list(maxit=500))
            par=c(rep(0,P*Q),fit$sigma)
            opt=optim(par,logl2,method=method)
            C=opt$par[1:(P*Q)]
            Sigma=opt$par[P*Q+1]
            fit=res.bm(x=dat,ord.param=ord.param,a=0,b=0,c=C,sigma=Sigma)
            res=fit$eps
            fitted.values=fit$fitted.values
            log.lik=-loglik(0,0,C,Sigma,ord.param,dat)
            res=list("a"=0,"b"=0,"c"=C,"sigma"=Sigma,"res"=res,
                "loglik"=log.lik,"fitted.values"=fitted.values,
                "dat"=dat,"método"=method,"ord.param"=ord.param,"x"=x)
}
        if(p!=0 && q!=0){
            logl4=function(y){
                a=y[1:q]
                b=y[(q+1):(p+q)]
                c=y[(p+q+1):(p+q+P*Q)]
                sigma=y[p+q+P*Q+1]
                loglik(a,b,c,sigma,ord.param,x=dat)
            }
            fit=arima(x, order=c(p,0,q),include.mean=FALSE,
                optim.control=list(maxit=500))
            par=c(fit$coef[(p+1):(p+q)],fit$coef[1:p],rep(0,P*Q),fit$sigma)
            opt=optim(par,logl4,method=method)
            A=opt$par[1:q]
            B=opt$par[(q+1):(p+q)]
            C=opt$par[(p+q+1):(p+q+P*Q)]
            Sigma=opt$par[p+q+P*Q+1]
            fit=res.bm(x=dat,ord.param=ord.param,a=A,b=B,c=C,sigma=Sigma)
            res=fit$eps
            fitted.values=fit$fitted.values
            log.lik=-loglik(A,B,C,Sigma,ord.param,dat)
            res=list("a"=A,"b"=B,"c"=C,"sigma"=Sigma,"res"=res,
                "loglik"=log.lik,"fitted.values"=fitted.values,
                "dat"=dat,"método"=method,"ord.param"=ord.param,"x"=x)
```

```
        }
    }
    class(res)="bm"
    return(res)
}
```

La siguiente función, `best.tar`, es la encargada de calcular los valores óptimos de los parámetros p, P, q y Q. Funciona de manera similar a la función `best.tar`. El código sería:

```
best.bm=function(x,ord.max,method,crit="AIC",dist.max.crit=2){
    T <- length(x)
    if (crit=="AIC")  factor <- 2
    else if (crit=="BIC")  factor <- log(T)
    pb=txtProgressBar(style=3)
    iind = 0
    nit=(ord.max[1]+1)*(ord.max[2]+1)*(ord.max[3])*(ord.max[4])
    crit.val=rbind(rep(0,5))
    for (p in 0:ord.max[1]) for (q in 0:ord.max[2])
    for (P in 1:ord.max[3]) for (Q in 1:ord.max[4])
    {
        setTxtProgressBar(pb, iind/nit)
        fit=bm(x,ord.param=c(p,q,P,Q),method)
        if (crit=="AICC")  factor=2*T/(T-p+q+P*Q-2)
        fit.crit=-2*fit$loglik+factor*(p+q+P*Q+1)
        if(res.test(fit$res)$mu>=0.05
            & sum(res.test(fit$res)$ind>=0.05)==20){
            crit.val=rbind(crit.val,c(p,q,P,Q,fit.crit))
        }
        iind<-iind+1
    }
    if(dim(crit.val)[1]==2){
        max.dist=crit.val[-1,]
    }else{
        crit.val=data.frame(crit.val[-1,])
        min.dist=crit.val[,5]-min(crit.val[,5])
        row.ok=rep(FALSE,length=length(min.dist))
        row.ok[min.dist<= dist.max.crit & min.dist>=0]=TRUE
        crit.val.ok=crit.val[row.ok,]
        max.dist=crit.val.ok[order(crit.val.ok[,5]),]
        max.dist=data.frame(max.dist)
    }
```

```
    if (crit=="AIC") names(max.dist)=c("p", "q","P","Q", "AIC")
    else if (crit=="AICC") names(max.dist)=c("p", "q","P","Q", "AICC")
    else names(max.dist)=c("p", "q","P","Q", "BIC")
    setTxtProgressBar(pb, 1)
    cat("\n")
    result=structure(list(max.dist=max.dist))
    return(result)
}
```

La última función para el modelo bilineal realiza la predicción de valores futuros. En este caso recibirá como argumentos un elemento de la clase "bm" obtenido de la función `bm`, el nivel de significación para los intervalos de confianza y el horizonte máximo al que calculará la predicción. El código:

```
predict.bm=function(bm,alpha=0.05,n.ahead=5){
    a=bm$a
    b=bm$b
    c=bm$c
    res=bm$res
    sigma=bm$sigma
    p=bm$ord.param[1]
    q=bm$ord.param[2]
    P=bm$ord.param[3]
    Q=bm$ord.param[4]
    pP=max(p,P)
    qQ=max(q,Q)
    c=matrix(c,byrow=TRUE,nrow=P,ncol=Q)
    mat.pred=matrix(0,nrow=500,ncol=n.ahead)
    pb=txtProgressBar(style=3)
    iind=1
    for(j in 1:500){
        setTxtProgressBar(pb, iind/500)
        x=bm$x
        eps=bm$res
        N=length(x)
        n=length(eps)
        for(i in 1:n.ahead){
            sum1=b%*%x[N:(N-length(b)+1)]
            sum2=x[N:(N-dim(c)[1]+1)]%*%c%*%eps[n:(n-dim(c)[2]+1)]
            sum3=a%*%eps[n:(n-length(a)+1)]
            mat.pred[j,i]=sum1+sum2+sum3+rnorm(1,mean=0,sd=sigma)
            x=c(x,mat.pred[j,i])
```

```
            eps=c(eps,sample(res,replace=TRUE)[1])
            N=N+1
            n=n+1
        }
        iind=iind+1
    }
    dat.mean=apply(mat.pred,2,mean)
    dat.sd=apply(mat.pred,2,sd)
    mat.pred.ord=apply(mat.pred,2,sort)
    quantili=mat.pred.ord[floor(dim(mat.pred)[1]*(alpha/2)),]
    quantils=mat.pred.ord[floor(dim(mat.pred)[1]*(1-(alpha/2))),]
    CI=cbind(quantili,quantils)
    cat("\n")
    fit=list("pred"=dat.mean,"se"=dat.sd,"mat.pred"=mat.pred,
        "quantili"=quantili,"quantils"=quantils,"CI"=CI)
    return(fit)
}
```

FAR

A continuación, pasamos a comentar las funciones programadas para el modelo FAR.

En primer lugar, la función `f.est` calcula la estimación de las f_j en el punto u_0 indicado. Cabe destacar que en esta función, el argumento `x` es la matriz con columnas $(X_{t-p}, ..., X_{t-1}, X_t)$. A continuación se muestra el código:

```
f.est=function(x,p,d,u0,Kh,h){
    T=dim(x)[1]
    w=numeric(T)
    for (i in 1:T){w[i]=Kh(x=x[i,p+1-d]-u0,h=h)}
    W=diag(w)
    Y=x[,(p+1)]
    XX=cbind(x[,p:1],x[,p:1]*(x[,p+1-d]-u0))
    if (det(t(XX)%*%W%*%XX)<=1e-12){
        beta=solve(t(XX)%*%W%*%XX+0.0001/(sqrt(T))
            *diag(rep(1,2*p)))%*%t(XX)%*%W%*%Y
    }
    else{beta=solve(t(XX)%*%W%*%XX)%*%t(XX)%*%W%*%Y}
    return(list("a"=beta[1:p], "b"=beta[(p+1):(2*p)]))
}
```

La función `far` es la encargada de calcular los valores ajustados y los residuos del modelo. Para esto, llama a la función `f.est`. El código:

```
far=function(x,p,d,Kh,h){
    T=length(x)
    pd=max(p,d)
    dat<-cbind(x)
    for (j in 1:pd){
        dat=cbind(dat,cbind(c(x[-c(1:j)],rep(NA,j))))
    }
    dat=dat[1:(T-pd),]
    n.eval=dim(dat)[1]

    pb=txtProgressBar(style=3)
    iind=1

    A=matrix(0,nrow=n.eval,ncol=p)
    for(I in 1:n.eval){
```

```
        setTxtProgressBar(pb, iind/n.eval)
        A[I,]=f.est(x=dat,p=p,d=d,u0=dat[I,pd-d+1],Kh=Kh,h=h)$a
        iind=iind+1
    }
    A=data.frame(A)
    cat("\n")

    fitted.values=rowSums(A*dat[,p:1])
    res=dat[,(pd+1)]-fitted.values
    sigma=sqrt(mean(res^2,na.rm=TRUE))
    std.res=(res-mean(res,na.rm=TRUE))/sigma

    fit=list("A"=A,"dat"=dat,"x"=x,"fitted.values"=fitted.values,"res"=res,
    "std.res"=std.res,"sigma"=sigma,"p"=p,"d"=d,"Kh"=Kh,"h"=h)
    class(fit)<-"far"
    return(fit)
}
```

La función `APEh` es la encargada de calcular $APE_h(q)$.

```
APEh=function(h,x,p,d,m,Q,Kh){
    T=dim(x)[1]
    Sum=numeric(m)
    apeh=numeric(Q)
    for (q in 1:Q){
        j=1
        for (i in (T-q*m+1):(T-q*m+m)){
            Sum[j]=(x[i,p+1]-f.est(x=x[1:(T-q*m),],p=p,d=d,
                 u0=x[i-d,p+1],Kh=Kh,h=h)$a%*%x[i,p:1])^2
            j=j+1
        }
        apeh[q]=mean(Sum,na.rm=TRUE)
    }
    return(apeh)
}
```

Para el cálculo del h óptimo, se programó la función `h.opt` que, como es lógico, llama a la función `APEh`. Esta función, calculará el valor de h de una rejilla de posibles valores que le pasaremos como argumento. Ademas de devolver el valor óptimo de h, también devuelve el valor del $APE(h)$ correspondiente. El código es el siguiente:

```
h.opt.far=function(x,p,d,Q,m,Kh,hj){
    T=length(x)-p
    dat<-cbind(x)
    for (j in 1:p){
        dat=cbind(dat,cbind(c(x[-c(1:j)],rep(NA,j))))
    }
    dat=dat[1:T,]
    pb=txtProgressBar(style=3)
    iind=1
    APE=numeric(length(hj))
    for(i in 1:length(hj)){
        setTxtProgressBar(pb, iind/length(hj))
        APE[i]=mean(APEh(x=dat,h=hj[i],p=p,m=m,d=d,Q=Q,Kh=Kh))
        iind=iind+1
    }
    pos=sort(APE,index.return=TRUE)$ix[1]
    hopt=hj[sort(APE,index.return=TRUE)$ix][1]
    cat("\n")
    return(list("h.opt"=hopt,"pos"=pos))
}
```

Por último, como en los modelos anteriores, la función `predict.far` se encarga de calcular las estimaciones de valores futuros de la serie. El código:

```
predict.far=function(far,alpha=0.05,n.ahead=5){
    x=far$x
    d=far$d
    p=far$p
    sigma=far$sigma
    dat=far$dat
    T=length(x)
    mat.pred=matrix(0,nrow=100,ncol=n.ahead)
    pb=txtProgressBar(style=3)
    iind=1
    for(j in 1:100){
        setTxtProgressBar(pb, iind/100)
        x=far$x
```

```
        for (i in 1:n.ahead){
            T=length(x)
            A=f.est(dat,p=p,d=d,u0=x[T-d+1],Kh=far$Kh,h=far$h)$a
            mat.pred[j,i]=A%*%c(x[T:(T-p+1)])+rnorm(1,mean=0,sd=sigma)
            x=c(x,mat.pred[j,i])
        }
        iind=iind+1
    }
    dat.mean=apply(mat.pred,2,mean)
    dat.sd=apply(mat.pred,2,sd)
    mat.pred.ord=apply(mat.pred,2,sort)
    quantili=mat.pred.ord[floor(dim(mat.pred)[1]*(alpha/2)),]
    quantils=mat.pred.ord[floor(dim(mat.pred)[1]*(1-(alpha/2))),]
    CI=cbind(quantili,quantils)
    cat("\n")
    fit=list("pred"=dat.mean,"se"=dat.sd,"mat.pred"=mat.pred,
        "quantili"=quantili,"quantils"=quantils,"CI"=CI)
    return(fit)
}
```

AFAR

Para terminar, se muestran las funciones programadas para el modelo AFAR.

En primer lugar, `g.est.afar` estima el valor de las funciones g_j en un punto `z` dado. El código es el siguiente:

```
g.est.afar=function(z,x,p,beta,Kh,h,w){
    T=dim(x)[1]
    ww=numeric(T)
    U=cbind(rep(1,T),x[,2:p])
    for (i in 1:T){
        ww[i]=Kh(beta%*%x[i,-1]-z,h=h)*w(beta%*%x[i,-1])
    }
    W=diag(ww)
    Y=x[,1]
    s=x[,-1]%*%beta-rep(z,T)
    XX=cbind(U,U*cbind(s,s))
    if (det(t(XX)%*%W%*%XX)<=1e-12){
        AUX=solve(t(XX)%*%W%*%XX+0.0001/(sqrt(T))*diag(rep(1,2*p)))%*%t(XX)%*%W
    }
    else AUX=solve(t(XX)%*%W%*%XX)%*%t(XX)%*%W
    g.est=AUX%*%Y
    return(list("a"=g.est[1:p], "b"=g.est[(p+1):(2*p)],"AUX"=AUX))
}
```

La función `afar` calcula los valores ajustados por el modelo y sus correspondientes residuos. Todo esto lo devuelve como un objeto de la clase "afar". El código es:

```
afar=function(x,beta=NULL,p,Kh,w,h){
    if(is.null(beta)) {
        beta=Beta.hat(x=x,p=p,Kh=Kh,w=w,h=h)$beta
    }
    N=length(x)
    dat=cbind(x)
    for (j in 1:p){
        dat=cbind(dat,cbind(c(x[-c(1:j)],rep(NA,j))))
    }
    dat=dat[1:(N-p),(p+1):1]
    T=dim(dat)[1]
    G=matrix(0,nrow=T,ncol=p)
    H.hat=matrix(0,nrow=T,ncol=T)
```

```
    for(I in 1:T){
        fit=g.est.afar(z=beta%*%dat[I,2:(p+1)],x=dat,
            p=p,beta=beta,Kh=Kh,h=h,w=w)
        G[I,]=fit$a
        H.hat[I,]=c(1,dat[I,2:p],rep(0,p))%*%fit$AUX
    }
    diag.H.hat=diag(H.hat)
    G=data.frame(G)
    fitted.values=rowSums(G*cbind(rep(1,T),dat[,2:p]))
    res=dat[,1]-fitted.values
    sigma=sqrt(mean(res^2,na.rm=TRUE))
    std.res=(res-mean(res,na.rm=TRUE))/sigma
    fit=list("dat"=dat,"x"=x,"beta"=beta,"fitted.values"=fitted.values,
        "res"=res,"std.res"=std.res,"sigma"=sigma,"p"=p,"Kh"=Kh,"h"=h,
        "diag.H.hat"=diag.H.hat)
    class(fit)<-"afar"
    return(fit)
}
```

La función `Beta.hat` computa la estimación de la dirección $\boldsymbol{\beta}$ para la serie de tiempo `x`.

```
Beta.hat=function(x,p,Kh,w,h){
    T=length(x)
    dat<-cbind(x)
    for (j in 1:p){
        dat=cbind(dat,cbind(c(x[-c(1:j)],rep(NA,j))))
    }
    dat=dat[1:(T-p),(p+1):1]
    # Standarized Data
    c.dat=dat[,-1]-matrix(apply(dat[,-1],2,mean),ncol=dim(dat[,-1])[2],
        nrow=dim(dat[,-1])[1],byrow=TRUE)
    std.dat=c.dat%*%solve(chol(cov(dat[,-1])))
    dat=cbind(dat[,1],std.dat)
    # Beta 0:
    fit1=lm(data=data.frame(dat))
    beta0=fit1$coef[2:(p+1)]
    betaold=norma(beta0)
    tol=0.001;ind=1;para=0
    while(ind<=30 && para==0){
        R.beta=function(beta){r.beta(beta,x=dat,p=p,Kh=Kh,w=w,h=h)}
        deri=genD(R.beta,betaold)$D
```

```
        der=deri[1:p]
        tri=triang(deri[(p+1):length(deri)],p)
        hess=tri+(t(tri-diag(diag(tri))))
        betanew=betaold-solve(hess)%*%der
        betanew=as.numeric(norma(betanew))
        if (abs(r.beta(beta=betanew,x=dat,p=p,Kh=Kh,w=w,h=h)-
            r.beta(beta=betaold,x=dat,p=p,Kh=Kh,w=w,h=h))<tol){para=1}
        else {
            betaold=betanew
            ind=ind+1
        }
    }
    fit=list("beta"=betanew,"ind"=ind,"x"=x,"p"=p,"Kh"=Kh,"h"=h)
    return(fit)
}
```

`h.opt.afar` calcula el valor del parámetro de suavizado óptimo para una serie de tiempo `x`. Es interesante señalar que uno de los argumentos que se le pueden pasar es el valor de $\boldsymbol{\beta}$. En caso de que no se le de ese valor, la función lo estimaría conjuntamente con el h. El código:

```
h.opt.afar=function(x,p,beta=NULL,Kh,w,hj){
    n=length(x)
    GCV=numeric(length(hj))
    pb=txtProgressBar(style=3)
    iind=1
    for(i in 1:length(hj)){
        setTxtProgressBar(pb, iind/length(hj))
        fit=afar(x=x,beta=beta,p=p,Kh=Kh,w=w,h=hj[i])
        GCV[i]=1/(n*(1-sum(fit$diag.H.hat)/n)^2)*
            sum((x[(p+1):n]-fit$fitted.values)^2)
        iind=iind+1}
    X1=hj^4
    X2=1/(n*hj)
    a=as.numeric(lm(GCV~X1+X2)$coef)
    if(a[2]>0 && a[3]>0){
        hopt=(a[3]/(4*n*a[2]))^(1/5)
    }else{
        hopt=hj[sort(GCV,index.return=TRUE)$ix][1]}
    cat("\n")
    return(list("h.opt"=hopt,"GCV"=GCV))
}
```

Para terminar, tenemos la función `predict.afar` que, como en el resto de modelos, realiza la predicción de valores futuros de la serie. El código es el siguiente:

```
predict.afar=function(afar,n.ahead=5,alpha=0.05){
    x=afar$x
    d=afar$d
    p=afar$p
    sigma=afar$sigma
    dat=afar$dat
    beta=afar$beta
    T=length(x)
    mat.pred=matrix(0,nrow=100,ncol=n.ahead)
    pb=txtProgressBar(style=3)
    iind=1
    for(j in 1:100){
        setTxtProgressBar(pb, iind/100)
        x=afar$x
        T=length(x)
        for (i in 1:n.ahead){
            G=g.est.afar(z=beta%*%x[T:(T-p+1)],x=dat,beta=beta,p=p,
                Kh=afar$Kh,h=afar$h,w=afar$w)$a
            mat.pred[j,i]=G[1]+G[2:p]%*%x[T:(T-p+2)]
                +rnorm(1,mean=0,sd=afar$sigma)
            x=c(x,mat.pred[j,i])
            T=length(x)
        }
        iind=iind+1
    }
    cat("\n")
    mat.pred.ord=apply(mat.pred,2,sort)
    quantili=mat.pred.ord[floor(dim(mat.pred)[1]*(alpha/2)),]
    quantils=mat.pred.ord[floor(dim(mat.pred)[1]*(1-(alpha/2))),]
    pred=apply(mat.pred,2,mean)
    std=apply(mat.pred,2,sd)
    return(list("pred"=pred,"se"=std,"mat.pred"=mat.pred,"quantili"=quantili,
        "quantils"=quantils))
}
```

Bibliografía

[1] Cai, Z., Fan, J. y Yao, Q. (2000). Functional–coefficient regression models for nonlinear time series. *Journal of the American Statistical Association*, **95**, 941–956.

[2] Chan, K.S. y Crier J.D. (2008). *Time Series Analysis with Applications in R.* Springer, New York.

[3] Chen, R. y Tsay, R.S. (1993). Functional–coefficient autoregressive models. *Journal of the American Statistical Association*, **88**, 298–308.

[4] Fan, J. y Yao, Q. (2003). *Nonlinear Time Series: Nonparametric and Parametric Methods*. Springer.

[5] Granger, C.W.J. y Anderson, A.P. (1978a). *An Introduction to Bilinear Models*. Van derhoeck & Ruprecht, Gottingen.

[6] Subba Rao, T. (1981). On the theory of bilinear models. *Journal of the Royal Statistical Society*, Series B, **43**, 224–255.

[7] Tong, H. (1990). *Non-linear Time Series: A Dynamical Systems Approach*, Oxford University Press, Oxford.

[8] Tsay, R.S. (2010). *Analysis of Financial Time Series* (3ª edición). Wiley, New York.

[9] Wei, W.W.S. (2006). *Time Series Analysis. Univariate and Multivariate Methods* (2ª edición). Pearson.