Series temporales y minería de flujos de datos

Trabajo autónomo I: Series temporales

ugr | Universidad de Granada

Máster en Ciencia de Datos e Ingeniería de los Computadores

Besay Montesdeoca Santana besayms.39@gmail.com

Fecha de entrega: viernes, 20 de mayo de 2017, 00:00

1. Parte teórica

Una serie temporal es una secuencia de datos, observaciones o valores, medidos en determinados momentos y ordenados cronológicamente. Los datos pueden estar espaciados a intervalos iguales o desiguales. Para el análisis de las series temporales se usan métodos que ayudan a interpretarlas y que permiten extraer información representativa sobre las relaciones subyacentes entre los datos de la serie o de diversas series y que permiten en diferente medida y con distinta confianza extrapolar o interpolar los datos y así predecir el comportamiento de la serie en momentos no observados, sean en el futuro (extrapolación pronóstica), en el pasado (extrapolación retrógrada) o en momentos intermedios (interpolación).
Uno de los usos más habituales de las series de datos temporales es su análisis para predicción y pronóstico (así se hace por ejemplo con los datos climáticos, las acciones de bolsa, o las series de datos demográficos).

Existen varios patrones o componentes que pertenecen a una serie temporal:

- La series temporales pueden tener **tendencia**, que indican los incrementos o decrementos a largo plazo en los datos.
- Por otro lado la serie temporal puede tener **estacionalidad**, donde los datos están afectados por un patrón estacional. Se trata de una componente causal debida a la influencia de ciertos fenómenos que se repiten de manera periódica.

Existen una serie de técnicas para modelar, analizar o incluso eliminar estas componentes para posteriormente realizar predicciones.
Para modelar la tendencia:

- Disponemos por ejemplo de la **estimación funcional**, donde aproximamos la tendencia como una función. Para ello se requiere realizar una hipótesis sobre el modelo que rige la tendencia.
- Por otro lado tenemos el **filtrado**, el cual consiste en aplicar un filtro de medias móviles para estimar dicha tendencia.
- Por último disponemos de la **diferenciación**, donde se diferencia la señal hasta que desaparezca la tendencia.

Después de aplicar algunas de estas técnicas debemos asegurarnos de que la tendencia está bien modelada. Para ello podemos mostrar los residuos ya que nos puede ayudar a percibir si los errores se distribuyen uniformemente. Un test estadístico nos puede ayudar a ver que el error se distribuye de forma normal, si es así aceptamos el modelo.

Para el cálculo de la estacionalidad:

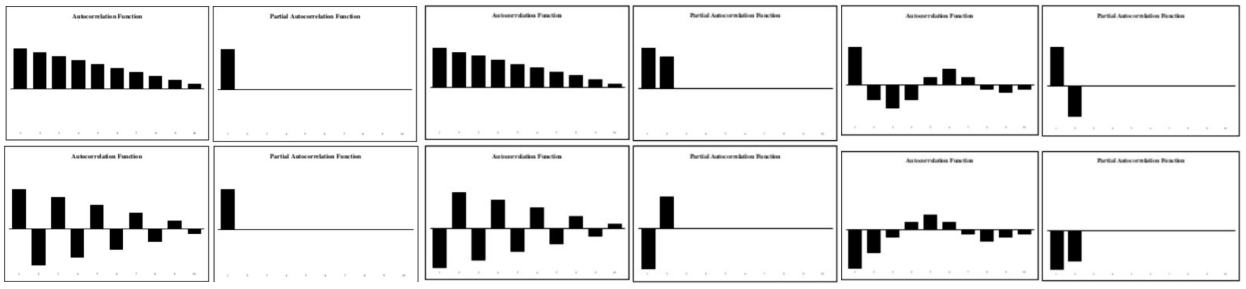
- La técnica que mejor resultados ha ofrecido es el cálculo de los **valores medios** de la serie dentro del mismo punto estacional, es decir, sea X una seria y P su periodo, calculamos la media para cada valor desde 1 hasta el periodo, de los valores de la serie X espaciados de P en P:
    - S(1)=media(X(1), X(13), X(25), X(37)...)
    - S(2)=media(X(2), X(14), X(26),X(38) ...)
    - S(3)=media(X(3), X(15), X(27),X(39) ...)
    - ...

- Otro método para eliminar estacionalidad consiste en utilizar la **diferenciación**, con un desfase d igual al periodo de la estacionalidad: $X' = X(t) - X(t - d)$

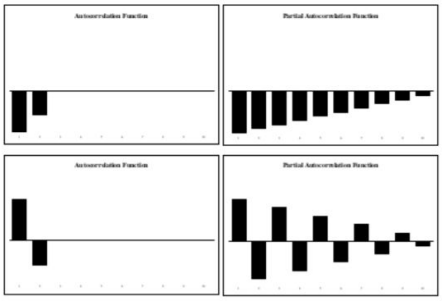
En 1970, Box y Jenkins desarrollaron un cuerpo metodológico destinado a identificar, estimar y diagnosticar modelos dinámicos de series temporales en los que la variable tiempo juega un papel fundamental. La metodología Box-Jenkins se refiere a una serie de procedimientos para identificar, ajustar y verificar los modelos de promedio móvil autorregresivo (más conocido por sus siglas en inglés ARIMA) con los datos de serie de tiempo. Los pronósticos proceden directamente de la forma del modelo ajustado, y es distinta de la mayoría de los métodos debido a que no supone un patrón particular en los datos históricos de las series que han de pronosticarse.

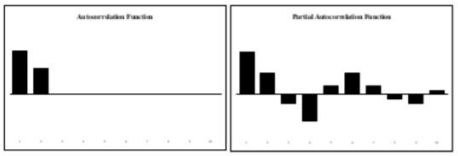
En cuanto al cálculo de los parámetros p y q de un modelo ARIMA, en principio se realiza visualizando las gráficas de los coeficientes de autocorrelación(ACF) y los coeficientes de autocorrelación parcial(PACF) de la serie. No existe una regla concreta para la obtención de los parámetros óptimos en todos los casos, pero como norma general:

Los modelos AR tiene un ACF que decrece a 0 (con diferentes posibles formas: regulares, sinusoidales, alternando +/-). El número del orden “p” (AR(p)) es tantos valores “distintos de 0 como haya en el PACF”.



Los modelos MA tiene un PACF que decrece a 0 (con diferentes posibles formas: regulares, sinusoidales, anternando +/-). El número del orden “q” (MA(q)) es tantos “valores distintos de 0” como haya en el ACF:





Un valor se considera “distinto de cero” si no está en el rango (-2/sqrt(N), 2/sqrt(N)), con N=longitud de la serie.

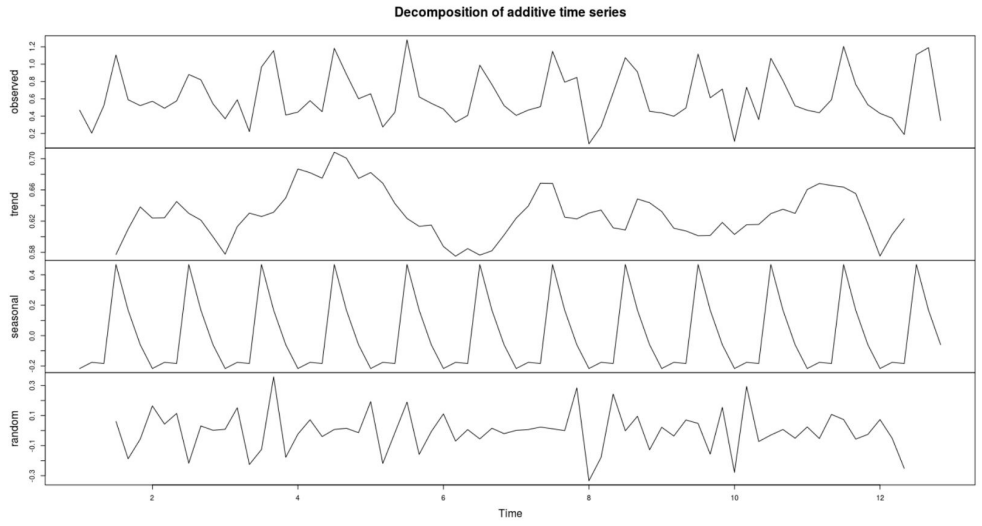
2. Parte práctica

Los pasos seguidos para conseguir una predicción final de una serie temporal descrito de forma breve son los siguientes:

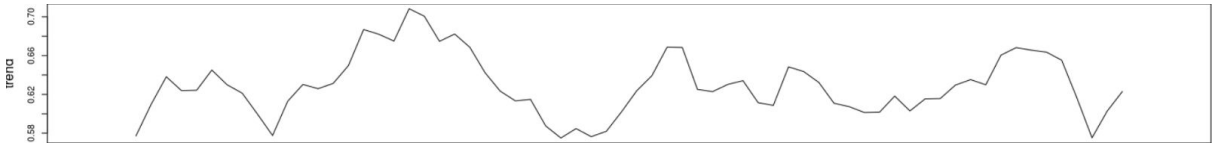
1. **Análisis de tendencia.** ¿Tiene tendencia la serie? Debemos averiguar si la serie tiene tendencia, si es así procederemos a modelarla y eliminarla.
2. **Análisis de estacionalidad.** ¿Sufre la serie de estacionalidad? Pueden existir patrones estacionales en los datos, si detectamos esto, nuevamente tenemos que modelar y eliminar este fenómeno.
3. **Estacionaridad.** ¿Es estacionaria la serie? En caso contrario, hacerla estacionaria mediante por ejemplo la diferenciación.
4. **Aplicar modelos paramétricos.** En nuestro caso, modelos autorregresivos y de medias móviles.
5. **Predicción**. Predecir en base a todas las componentes modeladas.

En primer lugar cargamos la serie en el entorno y mostramos su descomposición y gráfica ACF para hacernos una idea visual del perfil de la serie.

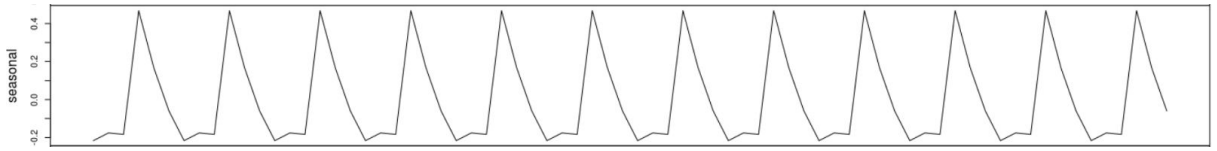
```
NPred = 6 # Valores a predecir
NTest = 6 # Valores que vamos a dejar para test

# Cargamos serie y datos reales a predecir
serie = scan("SerieTrabajoPractico.dat")

# Para averiguar la posible frecuencia de estacionalidad podríamos apoyarnos en la medida AFC
acf(serie)
# Observamos que la serie temporal posee un periodo de estacionalidad de 6 meses
serie.ts = ts(serie, frequency = 6)
# Visualizamos la descomposición
plot(decompose(serie.ts))
```

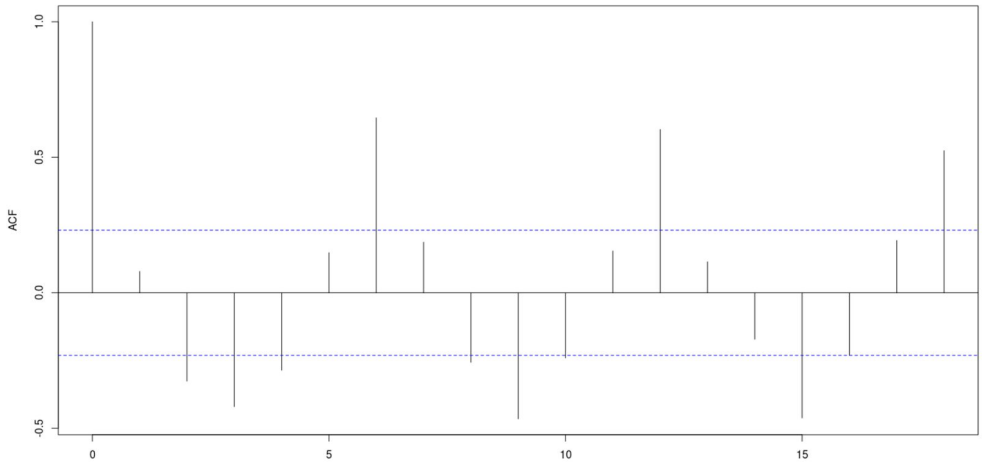


Observamos que no presenta ninguna tendencia, puesto que no se aprecia ningún incremento ni decremento constante en los datos.

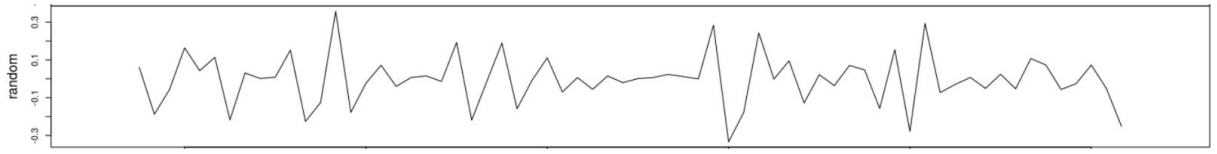


En cambio parece ser que la serie si sufre una estacionalidad. La frecuencia de la estacionalidad la podemos determinar analizando la gráfica ACF. Existe un patrón estacional que se repite cada 6 meses.





También queda patente que la serie no precisa de ningún preprocesamiento puesto que no existe una excesiva variación en la varianza en la parte irregular de la serie.



El siguiente paso será dividir los datos para ajuste y test. Cogeremos los 6 últimos valores para el test dado que también necesitaremos predecir 6 valores de la serie.

```
# Dividimos la serie en training y test (nos quedamos con los NTest últimos para el test)
serieTr = serie[1:(length(serie)-NTest)]
tiempoTr = 1:length(serieTr)
serieTs = serie[(length(serie)-NTest+1):length(serie)]
tiempoTs = (tiempoTr[length(tiempoTr)]+1):(tiempoTr[length(tiempoTr)]+NTest)
```

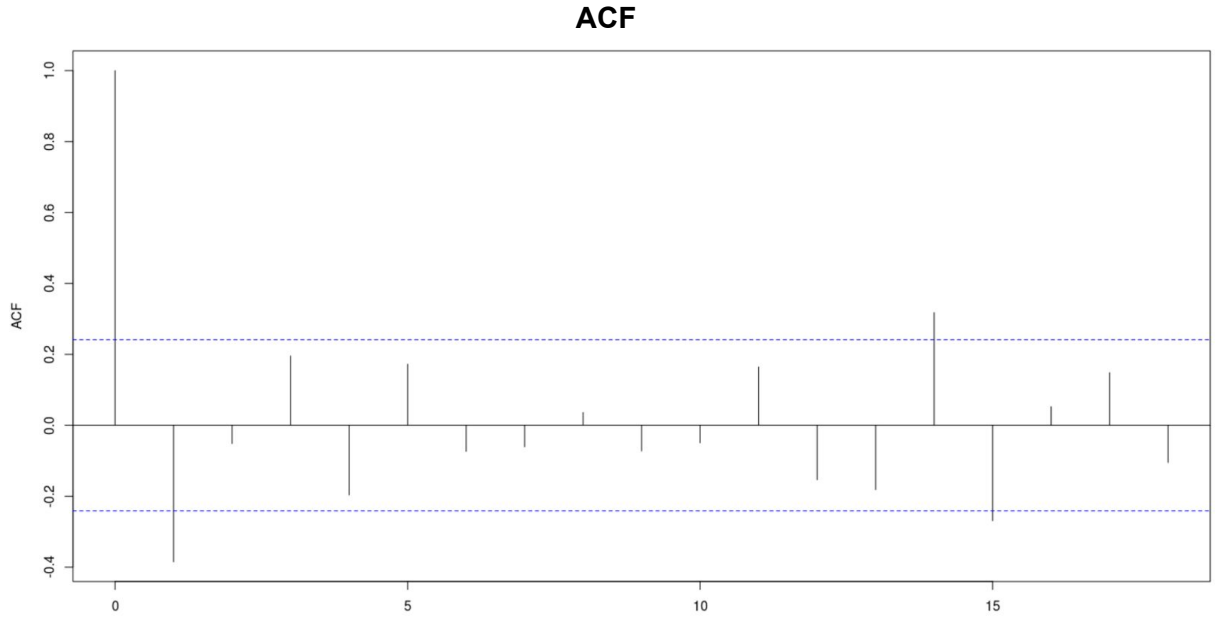
Es este punto la siguiente tarea sería modelar y eliminar la estacionalidad de la serie, para ello calculamos de los valores medios de la serie dentro del mismo punto estacional:

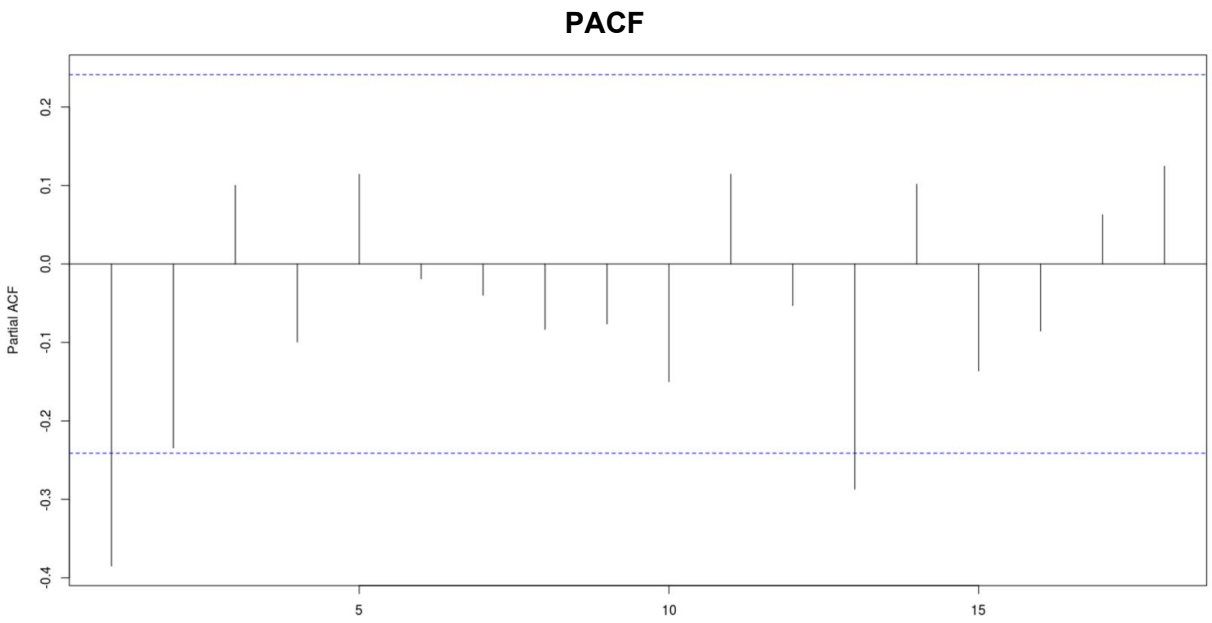
```
# Calculamos y eliminamos la estacionalidad
```

```
k = 6 # Periodo de estacionalidad = 6
estacionalidad =rep(0, k)
for (i in 1:k) {
 secuencia =seq(i, length(serieTr), by =k)
 for (j in secuencia) {
   estacionalidad[i]= estacionalidad[i] + serieTr[j]
 }
 estacionalidad[i]=estacionalidad[i]/length(secuencia)
}

# Eliminamos estacionalidad
aux = rep(estacionalidad, length(serieTr)/length(estacionalidad))
serieTr.SinEst = serieTr-aux
serieTs.SinEst = serieTs-estacionalidad
```

Una vez eliminado la estacionalidad visualizamos las gráficas de autocorrelación (ACF) y autocorrelación parcial (PACF) y realizamos el Test de Dickey-Fuller aumentado para comprobar si la serie es estacionaria.

```
acf(serieTr.SinEst)
pacf(serieTr.SinEst)

# Realizamos el test ADF
adftest = adf.test(serieTr.SinEst)
print(adftest)
```





El test ADF nos ha dado un p-value de 0.04451 por tanto se considera que la serie supera el test y es estacionaria.

El siguiente paso sería decidir qué tipo de modelo paramétrico se utilizará en la predicción en función de las gráficas de ACF y PACF.

Observando las gráficas podría tratarse de un modelo de autorregresivo de orden 1 o 13 según se quiera despreciar los los valores cercanos al umbral de error. Se podría considerar que el ACF decrece a 0 de forma quizás senoidal y por tanto el número de orden “p” es tantos “valores distintos de 0” como haya en el PACF.

Como esto no es una ciencia exacta y puede haber distintas interpretaciones de las gráficas, se modelarán y se compararán distintos modelos con el fin de determinar cuál es el que mejor rendimiento ofrecería realmente. Por ello también se probarán otros dos modelos pero esta vez de medias móviles cuyos órdenes, según las reglas que hemos visto antes serían de orden 1 y 14(donde el ACF decrece a 0 y el orden se determina con la gráfica PACF).

```
modeloAR_1= arima(serieTr.SinEst, order = c(1, 0, 0))
testSeries(modeloAR_1, NPred)

modeloAR_13 = arima(serieTr.SinEst, order = c(13, 0, 0))
testSeries(modeloAR_13, NPred)

modeloMA_1 = arima(serieTr.SinEst, order = c(0, 0, 1))
testSeries(modeloMA_1, NPred)

modeloMA_14 = arima(serieTr.SinEst, order = c(0, 0, 14))
testSeries(modeloMA_14, NPred)
```

Después de crear cada modelo ejecutamos una función donde se realizan distintos procedimientos para evaluar la precisión de las series, donde entre otras cosas se calculan los errores cuadráticos (ajuste y test) y se hace una comprobación de la bondad del ajuste mediante el tests de aleatoriedad y normalidad.

```
testSeries = function(modelo, Npred){
 # Cogemos los valores que se han ajustado de la serie
 valoresAjustados = serieTr.SinEst + modelo$residuals

 # Calculamos las predicciones
 Predicciones = predict(modelo, n.ahead = NPred)
 valoresPredichos = Predicciones$pred # Cogemos las predicciones

 # Calculamos el error cuadrático acumulado del ajuste, en ajuste y en test
 errorTr = sum((modelo$residuals)^2)
 errorTs = sum((valoresPredichos-serieTs.SinEst)^2)

 par(mfrow=c(2,2))

 # Mostramos las gráficas del ajuste y predicción en test
 plot.ts(serieTr.SinEst, xlim=c(1, tiempoTs[length(tiempoTs)]))
 lines(valoresAjustados, col = "blue")
 lines(tiempoTs, serieTs.SinEst, col = "red")
 lines(tiempoTs, valoresPredichos, col = "blue")

 valoresAjustados = valoresAjustados+aux # Estacionalidad
 valoresPredichos = valoresPredichos+estacionalidad

 plot.ts(serie)
 lines(valoresAjustados, col = "blue")
 lines(valoresPredichos, col = "red")

 # Tests para la selección del modelo y su validación
 boxtestM1 = Box.test(modelo$residuals) # Test de aleatoriedad de Box-Pierce
 JB = jarque.bera.test(modelo$residuals) # Test de normalidad de Jarque Bera
 SW = shapiro.test(modelo$residuals) # Test de normalidad de Shapiro-Wilk

 # Mostramos histograma de residuos
 hist(modelo$residuals, col = "blue", prob=T,ylim=c(0,8),xlim=c(-0.5,0.5))
 lines(density(modelo$residuals))
 par(mfrow=c(1,1))

 cat("\n.....................................................\n","Tabla
resumen\n.....................................................\n", "Errores Cuadráticos(ajuste,
test) = ", c(errorTr, errorTs),
     "\n\n Resultados test de aleatoriedad y normalidad\n", c(" Test de Box-Pierce: ",
boxtestM1$p.value,
                                                          "\n  Test de Jarque Bera: ", JB$p.value,
"\n  Test de Shapiro-Wilk: ", SW$p.value),
     "\n.....................................................\n")
}
```

Los resultados del test de cada modelo son los siguientes:

Modelo arima(1, 0, 0)

```
Tabla resumen
Errores Cuadráticos(ajuste, test) =  1.058235 0.2921816

Resultados test de aleatoriedad y normalidad
 Test de Box-Pierce:  0.448772970359382
 Test de Jarque Bera:  0.970207584030866
 Test de Shapiro-Wilk:  0.814467735913307
```

Modelo arima(13, 0, 0)

```
Tabla resumen
Errores Cuadráticos(ajuste, test) =  0.7997895 0.2753404

Resultados test de aleatoriedad y normalidad
 Test de Box-Pierce:  0.844873139613406
 Test de Jarque Bera:  0.788828592144502
 Test de Shapiro-Wilk:  0.542069172034094
```

Modelo arima(0, 0, 1)

```
Tabla resumen
Errores Cuadráticos(ajuste, test) =  1.023346 0.2921784

Resultados test de aleatoriedad y normalidad
 Test de Box-Pierce:  0.82297270586426
 Test de Jarque Bera:  0.942001922863065
 Test de Shapiro-Wilk:  0.532749917807833
```

Modelo arima(0, 0, 14)

```
Tabla resumen
Errores Cuadráticos(ajuste, test) =  0.6283986 0.395677

Resultados test de aleatoriedad y normalidad
 Test de Box-Pierce:  0.856860205278772
 Test de Jarque Bera:  0.381014029495845
 Test de Shapiro-Wilk:  0.19394369153
```

Observamos que los 4 modelos superan los test de aleatoriedad y normalidad y ofrecen errores cuadráticos similares donde destaca minuciosamente el modelo autorregresivo de orden 13.

Para comparar los modelos utilizaremos el criterio de Akaike(AIC), una medida de la calidad relativa de un modelo estadístico donde no sólo considera la bondad del ajuste, sino también la complejidad del modelo.

```
AIC(modeloAR_1, modeloAR_13, modeloMA_1, modeloMA_14)
```

	df	AIC
modeloAR_1	3	-79.32505
modeloAR_13	15	-71.54340
modeloMA_1	3	-81.48365
modeloMA_14	16	-74.84002

Viendo los resultados se optó por el modelo autorregresivo de orden 13(modelAR_13). Aunque obtiene un AIC peor con respecto a los otros modelos, debido posiblemente a que posee más grados de libertad y por tanto más complejidad, los errores cuadráticos tanto en ajuste como en test son mejores. Por esta razón este modelo será el que elegiremos en esta práctica para realizar la predicción.

Para finalizar, una vez elegido el modelo procederemos a realizar todos los pasos anteriores pero esta vez con todos los datos, con el objetivo de realizar la predicción de los 6 siguientes meses o valores de nuestra serie temporal. Para ello implementamos la siguiente función:

```
prediccionSerie = function(serie, NPred, orden) {
 serieFull = serie
 tiempo = 1:length(serieFull)
 parametros = lm (serieFull ~ tiempo  )

 # Calculamos estacionalidad
 k = 6
 estacionalidad =rep(0, k)
 for (i in 1:k) {
   secuencia =seq(i, length(serieFull), by =k)
```

```
  for (j in secuencia) {
    estacionalidad[i]= estacionalidad[i] + serieFull[j]
  }

  estacionalidad[i]=estacionalidad[i]/length(secuencia)
}
aux = rep(estacionalidad, length(serieFull)/length(estacionalidad))

#Eliminamos estacionalidad
serieFullSinEst = serieFull-aux

# Ajustamos el modelo que hemos seleccionado
modelo = arima(serieFullSinEst, order = orden)

# Obtenemos ajuste y predicción
valoresAjustados = serieFullSinEst+modelo$residuals
Predicciones = predict(modelo, n.ahead = NPred)
valoresPredichos = Predicciones$pred # Cogemos las predicciones

# Por último, deshacemos cambios
valoresAjustados = valoresAjustados+aux # Estacionalidad
valoresPredichos = valoresPredichos+estacionalidad
tiempoPred = (tiempo[length(tiempo)]+(1:NPred))

plot.ts(serie, xlim=c(1, max(tiempoPred)))
lines(valoresAjustados, col = "blue")
lines(valoresPredichos, col = "red")

# Devolvemos la predicción
return(valoresPredichos)
}
```

Valores predichos = {0.4339088 0.4677028 0.4021998 1.0751437 0.7878379 0.6190849}