1. Metodología Box-Jenkins para estimar modelos de series de tiempo.

La metodología Box-Jenkins es una serie de pasos que ayudan a estimar un modelo de Series de Tiempo tipo ARIMA para el pronóstico. La metodología fue usada originalmente con datos de un horno en gas y fue usado para pronosticar el compartimiento futuro usando Procesos Auto-regresivos con Medias Móviles, originalmente, la metodología se compone de tres pasos iterativos entre ellos:

- **Identificación y selección del modelo:**
  En este proceso, resulta necesario identificar los posibles procesos que componen a la serie de tiempo de interés (por ejemplo, si contiene una tendencia, o si contiene efectos externos, o si contiene un componente estacional). Es necesario comprobar que la serie de tiempo sea o no estacionaria, hacer las transformaciones necesarias para tener una **serie estacionaria** tanto en media como en varianza, analizar si la serie presenta o no un comportamiento estacional (para estimar si se debe de ocupar un modelo estacional o no), y con ayuda de las gráficas de correlogramas, tratar de inferir que tipo de proceso y orden de modelo utilizar para representar a la serie.

- **Estimación de parámetros usando para tener coeficientes que mejor ajusten el modelo ARIMA seleccionado.**
  Normalmente los procesos de optimización que se usan para generar los coeficientes son los de Máxima Verosimilitud. En este paso también resulta importante realizar las pruebas de estabilidad de los parámetros estimados del modelo, debido a que un problema de estabilidad podría arrojar resultados explosivos que resulten en pronósticos ineficientes.

- **Comprobar si el modelo estimado se ajusta a las especificaciones de un proceso univariado estacionario:**
  Generalmente en este paso es necesario comprobar, a partir de los supuestos sobre los residuos, si el modelo arroja residuos que se comporten de manera aleatoria (es decir, con una distribución normal, sin problemas de auto-correlación serial, y con una varianza constante). Este paso es el más riguroso de todos, ya que es donde podremos afirmar si el modelo estimado es bueno o no para el pronóstico de la serie. En este punto, si los resultados se alejan de los deseados para una variable aleatoria, es necesario regresar desde la identificación y selección del modelo, o incluso desde la delimitación de los datos seleccionados para el análisis, con la finalidad de arreglar los problemas que podría ocasionar que los residuos no estén "limpios" para pasar las pruebas correspondientes.

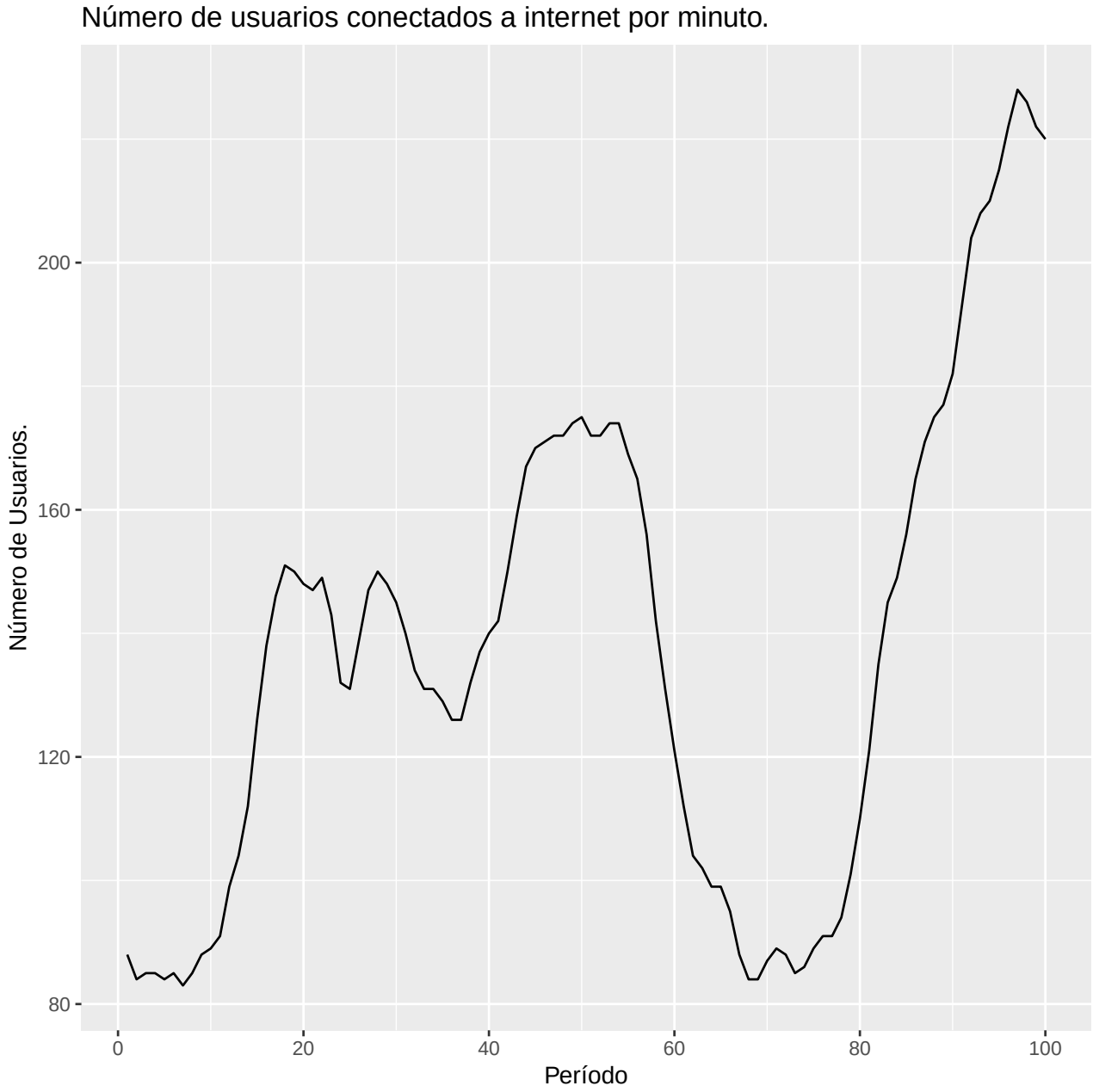
Como tal una serie de tiempo puede ser representada por muchos procesos estocásticos diferentes, de tal forma en que tendremos que seleccionar los modelos que sean más "cercanos a la realidad". Una práctica muy común al recurrir a modelos ARIMA es generar distintas propuestas de modelos, y con base en el cálculo de los indicadores de los residuos (la diferencia entre los datos pronosticados con los datos reales) podemos encontrar los modelos que, en términos de la diferencia entre los pronósticos y los datos reales, cumplen con tener las menores diferencias posibles. Entre estos indicadores encontraremos los *Criterios de información*.

1.1. Estimación de modelos ARIMA: Análisis de la tendencia y el Ruido Blanco

```
library(tseries)

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

library(forecast)
library(ggplot2)
data("WWWusage")#cargamos base de datos
Usuarios <- WWWusage #Serie de usuarios conectados a internet por minuto
```

Gráfica

```
autoplot(Usuarios)+
  labs(title = "Número de usuarios conectados a internet por minuto.",
       x = "Período", y="Número de Usuarios." )+theme_gray()
```



A simple vista podemos contrastar que la serie tien importantes caídas y alzas en sus niveles; además, aunque sea muy leve, existe una tendencia estocástica que ocasiona que la serie no se mantenga en una media constante.
Siguiendo la metodología de Box-Jenkins, lo primero sería realizar las pruebas de estacionariedad para comprobar que la serie sea o no una serie estacionaria. Las pruebas más comunes para este tipo de contrastes son:

- **Prueba de Dickey-Fuller Aumentado (ADF):** Elimina la auto-correlación e indica si una serie es estacionaria o no.
- **Prueba de Phillips-Perron:** Es una modificación de la prueba de Dickey-Fuller. Esta prueba corrige la auto-correlación y heterocedasticidad (varianza NO constante) en los errores para comprobar la existencia de estacionariedad de la serie.

Recordemos...
Una prueba de hipótesis, en el campo de la Estadística, se comprende como una regla o una validación en donde se específica si se puede aceptar o rechazar una afirmación sobre una población o fenómeno, dependiendo de la evidencia proporcionada por una muestra de datos. Estas se basan en dos hipótesis:

- H_0 **Hipótesis nula (simple)**: Es el enunciado que se buscará comprobar
- H_a **Hipótesis alternativa (compuesta):** Es el resultado de que la nula se rechace de acuerdo con los datos proporcionados.

Con base en los datos proporcionados es que se puede realizar la prueba, y así determinar si se puede rechazar o no la hipótesis nula; la forma de determinar esto es con base en los diferentes estadísticos, o bien, usando el P-value de la prueba. La forma más sencilla de saber que es el P-value es verlo como la probabilidad de rechazar la hipótesis nula cuando esta es verdadera. Por lo tanto:

- Si p-valor$< \alpha$, entonces se puede rechazar la hipotessi nula.
- Si p-valor$> \alpha$, hay evidencia para que no se rechace la hipótesis nula.

Para realizar la prueba *Dickey-Fuller Aumentada* sobre la serie de prueba que estamos ocupando, podemos utilizar el siguiente código:

```
adf.test(Usuarios , alternative = "stationary", k=0)

## Warning in adf.test(Usuarios, alternative = "stationary", k = 0): p-value greater than
printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  Usuarios
## Dickey-Fuller = -0.15634, Lag order = 0, p-value = 0.99
## alternative hypothesis: stationary
```

Con un valor del p-value del 0.99, muchísimo mayor al $\alpha = 0.05$ nivel de significancia de la prueba, se acepta la hipótesis nula y rechazamos la hipótesis alterna, lo que contrasta que la serie presenta un comportamiento **NO estacionario.** Esto indica que existe una posible tendencia estocástica sobre la serie. Para comparar los resultados de la prueba *Dickey-Fuller aumentada*, utilizaremos la prueba ***Phillips-Perron***, y comprobaremos si esta igual señala la existencia de un comportamiento NO estacionario utilizando el siguiente código:

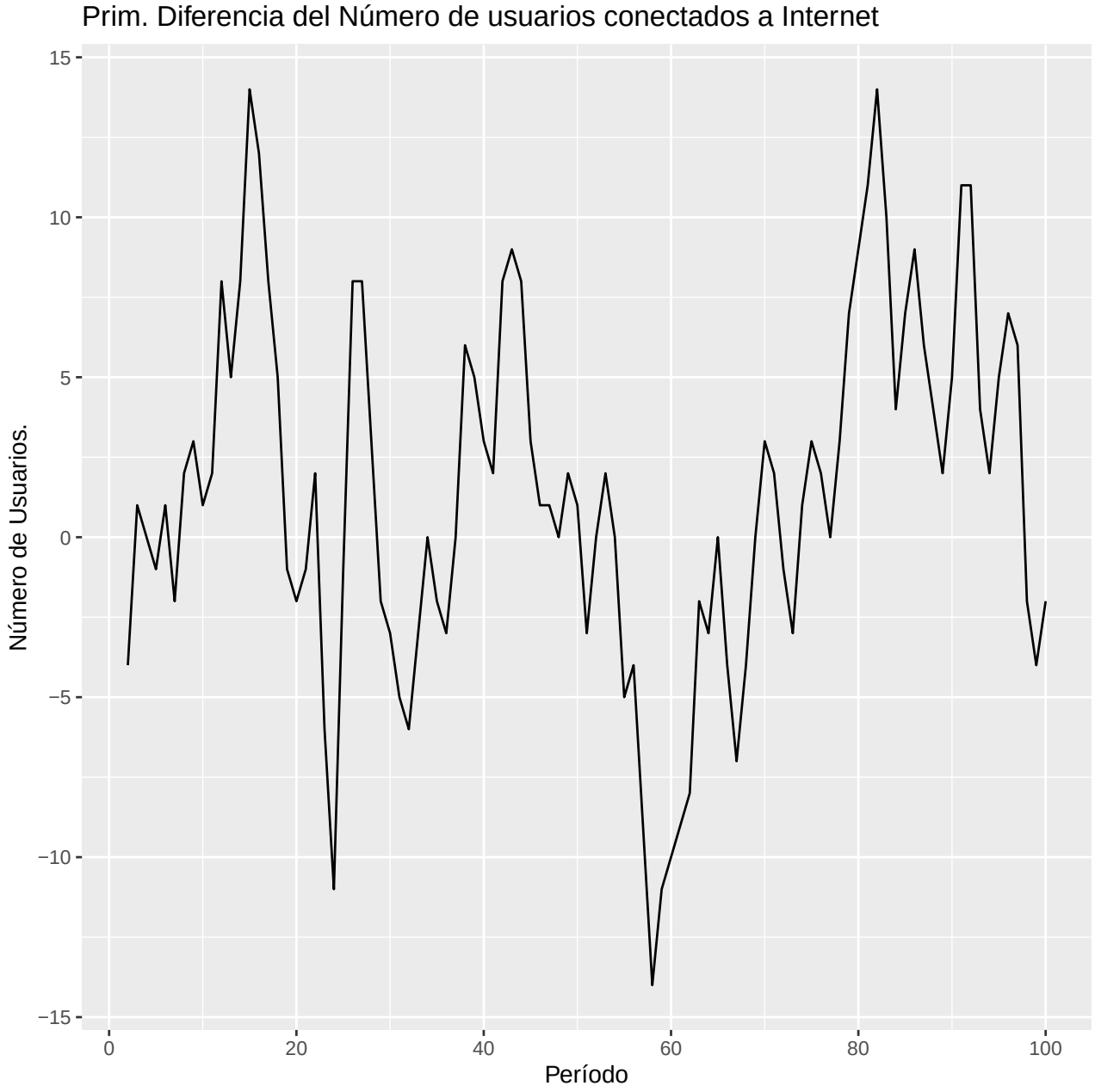
```
pp.test(Usuarios , alternative = "stationary")

##
##  Phillips-Perron Unit Root Test
##
## data:  Usuarios
## Dickey-Fuller Z(alpha) = -2.6214, Truncation lag parameter = 3, p-value
## = 0.9499
## alternative hypothesis: stationary
```

Con un P-value mayor al 0.05 nivel de significancia, al igual que la prueba *Dickey-Fuller Aumentada*, la prueba *Phillips-Perron* contrasta que la serie NO es una serie estacionaria. Por lo que es un buen candidato para presentar una tendencia estocástica que impide el cumplimiento de los supuestos de estacionariedad de la serie. Como ya se había menciondad, el proceso más normal para volver estacionaria una serie de tiempo es *diferenciarla*, y de esta manera eliminar el efecto de la tendencia estocástica, dejando solo los procesos AR y MA.

Ojo: Siempre hay que considerar que una serie puede NO ser estacionaria por falta de una media constante, o porque existe un importante factor que impide que la serie tenga una varianza constante: el método más común para solucionar el supuesto de la media es diferenciando, y el método que se ocupa para resolver el problema del supuesto de la varianza es ocupar una transformación algorítmica o transformación Box-Cox, por el momento supondremos que el problema de la serie es sobre su supuesto en la media.

Para visualizar la serie diferenciada del número de usuarios conectados a internet cada minuto utilizando la función *diff()* (por default la función utiliza el rezago = 1, para un mayor número necesario especificar con la opción *lag*)

```
autoplot(diff(Usuarios , lag = 1))+
  labs(title = "Prim. Diferencia del Número de usuarios conectados a Internet",
       x="Período", y="Número de Usuarios.")+theme_gray()
```



Al aplicar la primera diferencia de la serie original, podemos contrastar que la transformación permitió que la serie se mantuviera en una media constante cercana a 0. Esto podría ayudar a que la serie sea ahora estacionaria en media, lo que podría solucionar el problema del factor integrado de la serie. Para confirmar esto, volveremos a realizar las pruebas *Dickey-Fuller aumentada* y *Phillips-Perron*. Para realizar la prueba *Dickey-Fuller Aumentada* ocupamos el siguiente código:

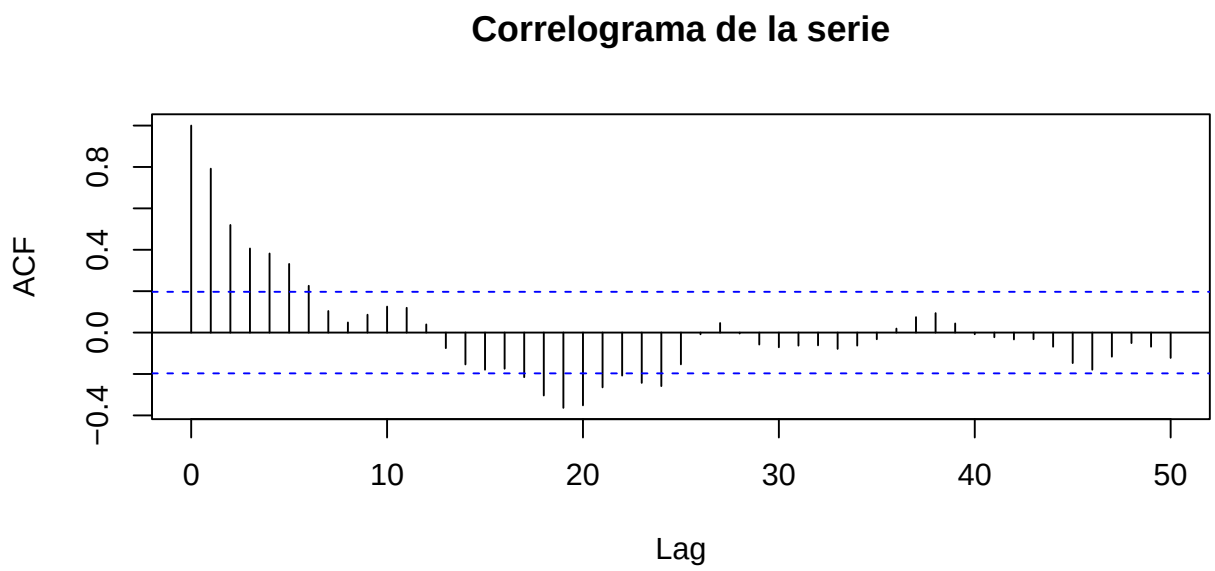
```
adf.test(diff(Usuarios),alternative = "stationary", k=0 )

##
##  Augmented Dickey-Fuller Test
##
## data:  diff(Usuarios)
## Dickey-Fuller = -3.2941, Lag order = 0, p-value = 0.07624
## alternative hypothesis: stationary
```

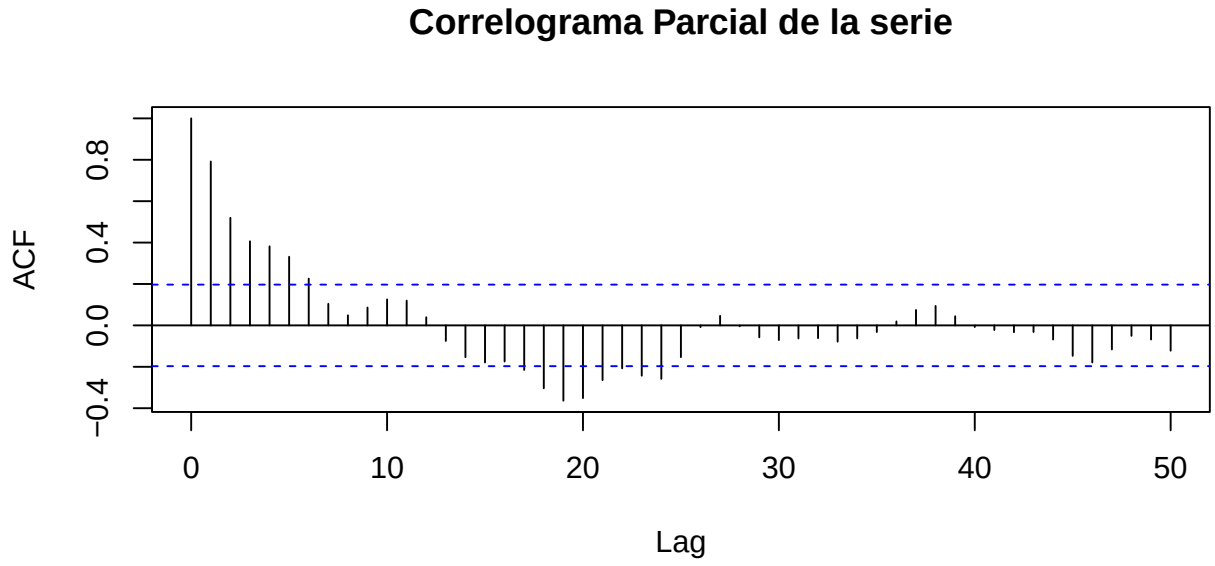
Con n valor del p-valor menor de 0.0762, bastante cercano al 0.05 nivel de significancia podemos confirmar o rechazar la hipótesis nula de NO estacionariedad. En este caso de decisión del autor rechazarla o no, ya que sigue siendo por un poco mayor al 0.05 nivel de significancia, sin embargo, esto es suponiendo que consideramos un nivel de confianza del 95 % . Para efectos de nuestros ejemplos, aceptaremos la hipótesis alterna, rechazando la hipótesis nula, por lo que existe evidencia estadística de que la serie sea una serie estacionaria. Para comparar dichas conclusiones, ahora utilizaremos la prueba de Phillips-Perron para contrastar estacionariedad utilizando el siguiente código:

```
pp.test(diff(Usuarios),alternative = "stationary" )

##
##  Phillips-Perron Unit Root Test
##
## data:  diff(Usuarios)
## Dickey-Fuller Z(alpha) = -20.405, Truncation lag parameter = 3, p-value
## = 0.05406
## alternative hypothesis: stationary
```

Algo interesante que podemos notar de la prueba de *Phillips-Perron* es que, si se elimina el efecto de heterocedasticidad, la serie se encuentra a ser más estacionaria que en la prueba de *Dickey-Fuller Aumentada*; esto da evidencia de que, si es posible, una transformación Box-Cox o logarítmica podría ayudar a modelar de mejor manera la serie. Para efectos didácticos del ejercicio, seguiremos suponiendo que la serie diferenciada en un rezago es estacionaria, por lo que podemos proceder con las siguientes pasos correspondientes pruebas que señala la metodología Box-Jenkins. Una vez que comprobamos que la primera diferencia de la serie es estacionaria, podemos realizar el análisis de correlaciones, y de esta manera poder identificar la existencia de un posible proceso AR, MA o ARMA. Para evitar realizar por separado los correlogramas, podemos utilizar la función *layout()*, y de esta manera tener ambos correlogramas en la misma imagen. Los criterios dentro del layout(1:2) indican que queremos que la imagen se componga de una sola columna, y dos filas.

```
layout(1:2)
acf(diff(Usuarios, lag = 1), 50, main = "Correlograma de la serie")
acf(diff(Usuarios, lag = 1), 50, main = "Correlograma Parcial de la serie")
```





Utilizando los correlogramas simple y parcia, podemos confirmar que el resultado de la primera diferenciación de la serie de "Usuarios" se puede considerar como un proceso estacionario, y que, a su vez, este se puede modelar a partir de un proceso AR y MA; es decir, un proceso ARMA estacionario, ya que en ambos correlogramas existen rezagos que traspasan las bandas de correlación.
Dentro de la metodologúa estándar de estimación de modelos ARMA, el proceso se suele realizar a partir de candidatos posibles que puedan modelar adecuadamente el comportamiento estocástico de la serie. Para este ejemplo, hemos propuesto unos 5 candidatos a modelos ARIMA, los cuales son:

- ARIMA(1,1,1)
- ARIMA(1,1,0)
- ARIMA (2,1,1)
- ARIMA (1,1,2)
- ARIMA (0,1,1)

No hay que perder de vista que los posibles candidatos a modelar el proceso estocástico se suelen elaborar a partir de comparar los correlogramas hipotéticos con los correlogramas reales, y de esta manera encontrar los mejores candidatos posible. Sin embargo, otra posible propuesta es elaborar

una buena cantidad de candidatos posibles (lo cual resulta ser bastante fácil utilizando el programa *R*, a diferencia de otros programas estadísticos) y comparar cual de esos candidatos tiene el menor criterio de información. Nosotros realizaremos este último proceso debido a la complejidad de simular varios procesos AR, MA y/o ARMA para la obtención de los correlogramas hipotéticos. El código que ocuparemos para estimar los procesos ARIMA seleccionados sobre la serie de "Usuarios" es el siguiente:

```
#Calculamos los modelos ARIMA propuestos:
ModA_ARIMA =arima(Usuarios, order = c(1,1,1))
ModB_ARIMA =arima(Usuarios, order = c(1,1,0))
ModC_ARIMA =arima(Usuarios, order = c(2,1,1))
ModD_ARIMA =arima(Usuarios, order = c(1,1,2))
ModE_ARIMA =arima(Usuarios, order = c(0,1,1))
```

El código que ocuparemos para comparar los criterios de información de los modelos propuestos es:

```
#Comparamos los modelos estimados para seleccionar el que contenga el menor valor en el Criterio
ModA_ARIMA$aic

## [1] 514.2995

ModB_ARIMA$aic

## [1] 529.2378

ModC_ARIMA$aic

## [1] 516.2914

ModD_ARIMA$aic

## [1] 516.2519

ModE_ARIMA$aic

## [1] 549.8055
```

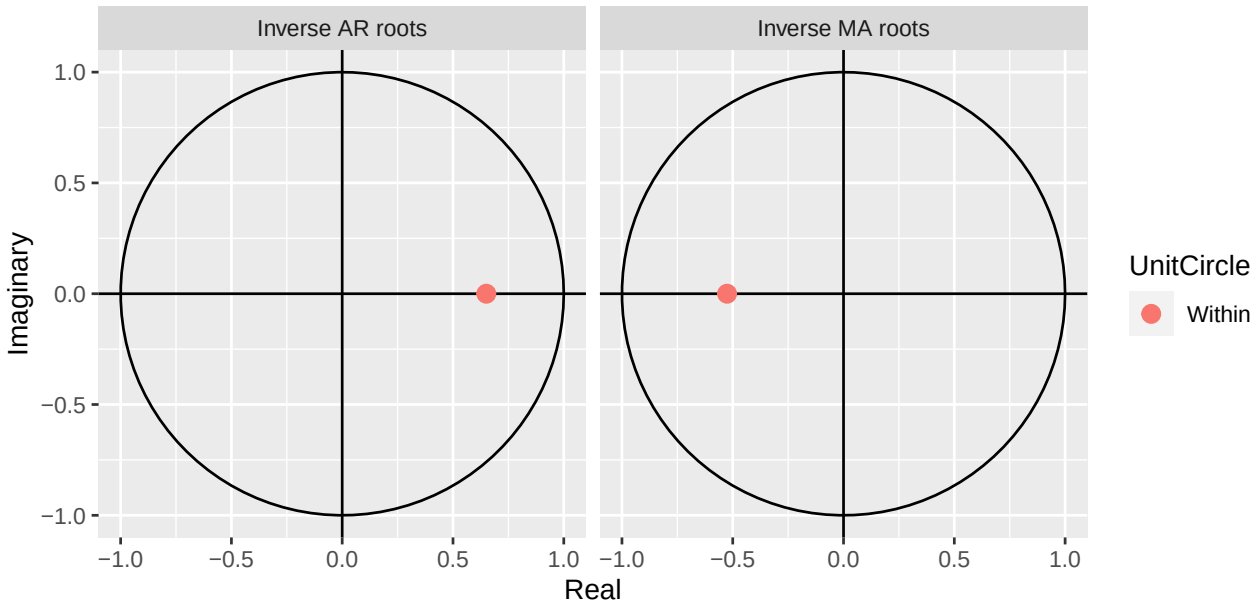
Podemos observar que al hacer la comparación de modelos utilizando el **Criterio de Información de Akaike**, el modelo que cuenta con el menor valor es el modelo ARIMA(1,1,1), por lo cual ese es el modelo que ocuparemos para realizar las pruebas sobre los supuestos y, en el caso de que los supuestos se cumplan, el modelo que podremos ocupar para realizar el pronóstico de la serie.
Podemos ocupar la función *summary()* sobe el modelo seleccionado para observar las estimaciones de los parámetros del modelo, utilizando el siguiente código:

```
summary(ModA_ARIMA)

##
## Call:
## arima(x = Usuarios, order = c(1, 1, 1))
##
## Coefficients:
##          ar1     ma1
##       0.6504  0.5256
## s.e.  0.0842  0.0896
##
```

```
## sigma^2 estimated as 9.793:  log likelihood = -254.15,  aic = 514.3
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.3035616 3.113754 2.405275 0.2805566 1.917463 0.5315228
##                      ACF1
## Training set -0.01715517
```

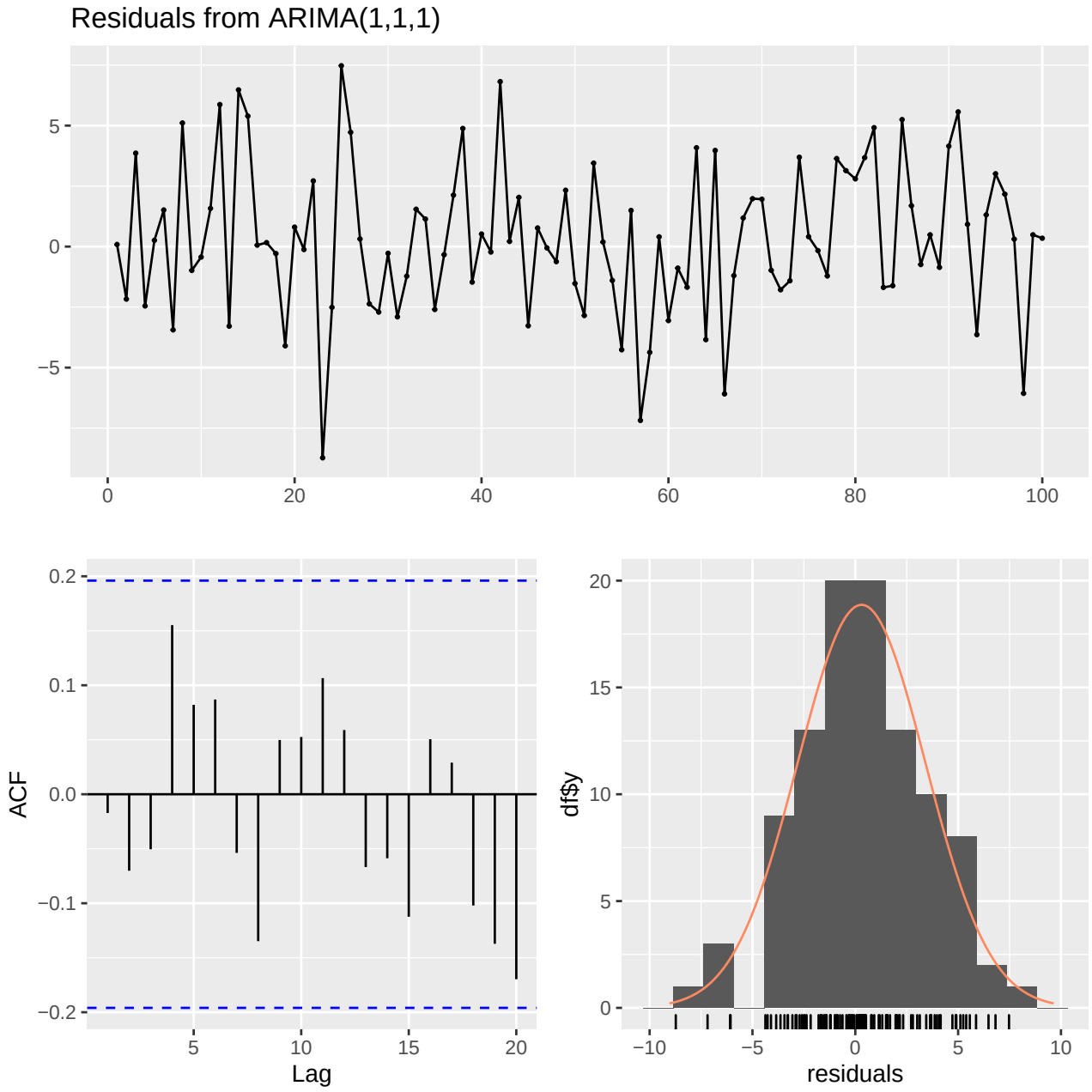
El parámetro del modelo proceso AR es igual a 0.6504, y el parámetro del proceso MA es igual a 0.52. Por otra parte, la función *summary()* también nos permite calcular los valores de algunos indicadores del error sobre el **Training Set**. Por el momento, no nos harán falta revisarlos, por lo que procederemos con la realización de nuestro ejercicio.
Un supuesto fundamental para el uso de modelos ARIMA para el pronóstico de series de tiempo es que los parámetros estimados, tanto del proceso AR como del proceso MA del modelo estimado, no presente comportamientos explosivos que podrían ocasionar que el pronóstico sea erróneo. Por suerte, la función *arima()* cuenta con default con el cálculo de las raíces unitarias del modelo, por lo que lo único que se requiere para observar los resultados es utilizar la función *autoplot()* sobre el modelo seleccionado, justo como realizamos en el siguiente código:

```
autoplot(ModA_ARIMA)+theme_gray()
```



Las raíces, tanto del proceso AR como al proceso MA del modelo propuesto se encuentran dentro de los círculos unitarios, por lo que los pronósticos convergen a la media, de tal manera que no presentan un comportamiento explosivo que podría afectar a la certeza de nuestro pronóstico. Una vez comprobado este supuesto, podremos continuar con realizar los supuestos sobre los errores de las estimaciones del modelo. El cumplimiento de los supuestos básicos sobre los residuos es de suma importancia para el uso de pronósticos con modelos de series de tiempo, debido a que lo que se busca es que no exista (o que al menos no de manera importante) ninguna variable determinista que pueda afectar al modelo, dejando "limpia" esa variable, y por lo tanto, lo que sobre sea una variable puramente aleatoria. Para esto, tendremos que contrastar con unas últimas pruebas si estos errores se distribuyen o no de manera normal. como una primera impresión, podemos ocupar la función *checkresiduals()* con este código:

```
checkresiduals(ModA_ARIMA)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)
## Q* = 7.8338, df = 8, p-value = 0.4499
##
## Model df: 2.   Total lags used: 10

mean(ModA_ARIMA$residuals)

## [1] 0.3035616
```

Al utilizar la función, podemos apreciar una imagen que cuenta con 3 gráficas útiles:

1. Gráfica de los residuos: Podemos contrastar que, en efecto, los errores se mantienen en una media constate igual a cero, y dentro de unos intervalos de varianza entre 5 y -5.

2. Histograma junto con una gráfica de distribución: Observamos que la serie tiene un comportamiento ligeramente normal, ya que no presenta un importante sesgo ni una fuerte curtosis.

3. Correlograma simple: Es posible contrastar que no existen importantes comportamientos correlacionados entre los rezagos de la serie, de tal forma que cumplen con el supuesto de **NO** autocorrelación de los errores.

No hay que olvidar que este es solo un acercamiento al cumplimiento de los supuestos de los errores de la serie, el cual no podrá sustituir al cumplimiento de las pruebas formales para contrastar el cumplimiento de los supuestos de manera individual. Para contrastar el cumplimiento del supuesto de **NO** autocorrelación, realizamos la prueba "Ljung-Box" sobre los errores, utilizando el siguiente código:

```
Box.test(ModA_ARIMA$residuals , lag = 20, type = "Ljung-Box")

##
##  Box-Ljung test
##
## data:  ModA_ARIMA$residuals
## X-squared = 19.736, df = 20, p-value = 0.4745
```

Al realizar la prueba Ljung-Box sobre los residuos del modelo, con un p-valor mucho mayor al 0.05 nivel de significancia, podemos aceptar la hipótesis nula de que la serie no tiene problemas de auto-correlación. Para realizar la prueba de normalidad, vamos a proponer dos pruebas estadísticas:

- Prueba Jarque-Bera
- Prueba Shapiro

El uso de mabas pruebas, al igual que el uso de las pruebas de raíces unitarias, servirán para confirmar de mejor manera el cumplimiento o el incumplimiento del supuesto de normalidad de la serie. Lo anterior lo hacemos de la siguiente manera:

```
library(tseries) #Es para poder hacer la prueba JARQUE-BERA
jarque.bera.test(ModA_ARIMA$residuals)

##
##  Jarque Bera Test
##
## data:  ModA_ARIMA$residuals
## X-squared = 0.13243, df = 2, p-value = 0.9359
```

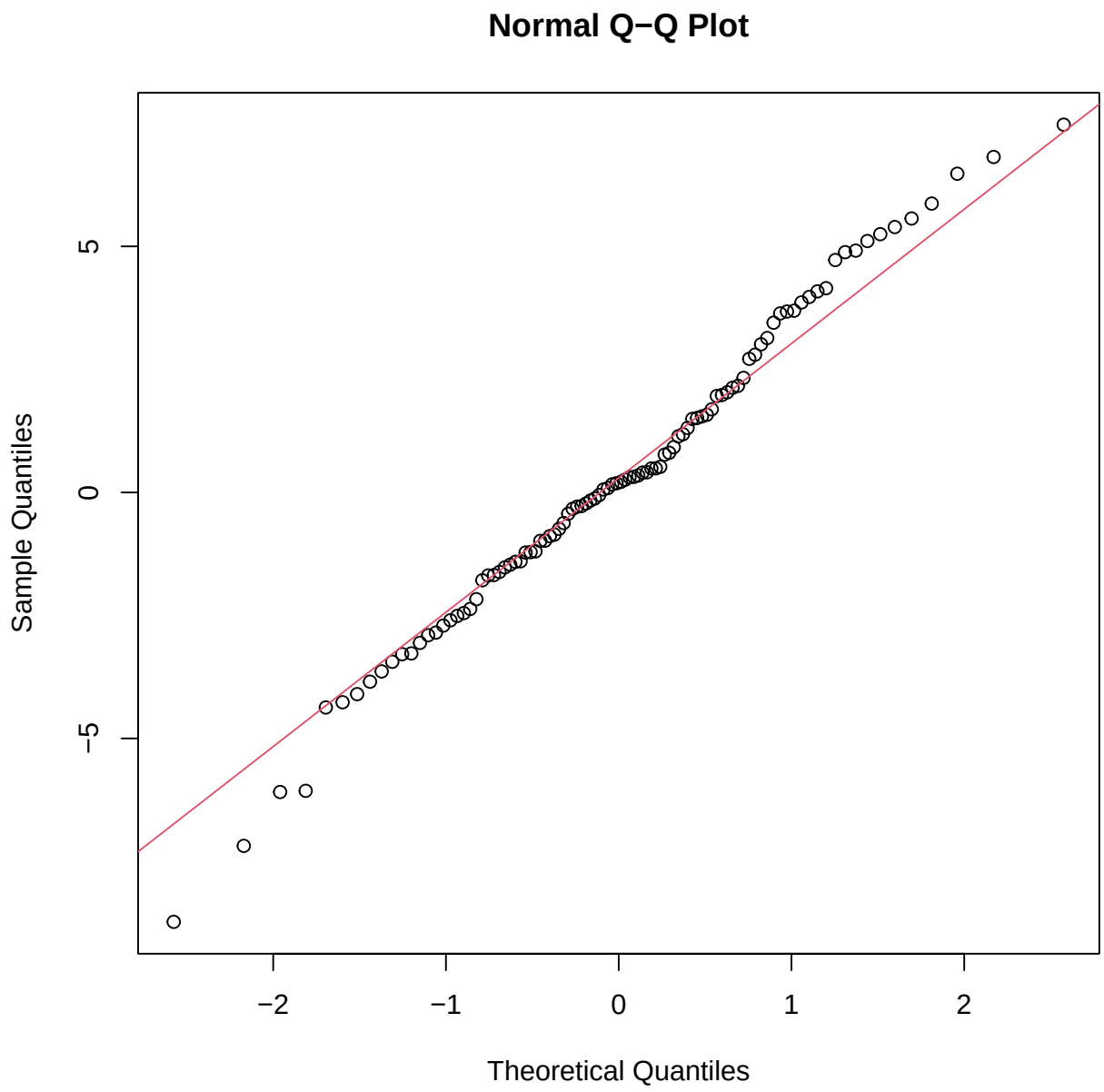
El resultado de la prueba, al igual que otros contrastes que hemos mencionado, se puede confirmar observando el p-valor obtenido. de acuerdo con los resultados de la prueba, con un p-valor mayor al 0.05, se acepta la hipótesis nula de normalidad sobre los residuos del modelo seleccionado. Confirmamos que el modelo tiene residuos que se distribuyen de manera normal. Por otra parte, procedemos a confirmar el supuesto de normalidad sobre los residuos utilizando de manera paralela la prueba de *Shapiro* utilizando el siguiente código:

```
shapiro.test(ModA_ARIMA$residuals)

##
##  Shapiro-Wilk normality test
##
## data:  ModA_ARIMA$residuals
## W = 0.99057, p-value = 0.7107
```

El p-valor de la prueba, con un valor de 0.71, mucho mayor al nivel de significancia del 0.05, se acepta como verdadera la hipótesis nula de normalidad sobre los residuos del modelo ARIMA (1,1,1). Por último, podemos realizar un tercer contraste de normalidad con la Q-Q plot sobre los resiudos, la

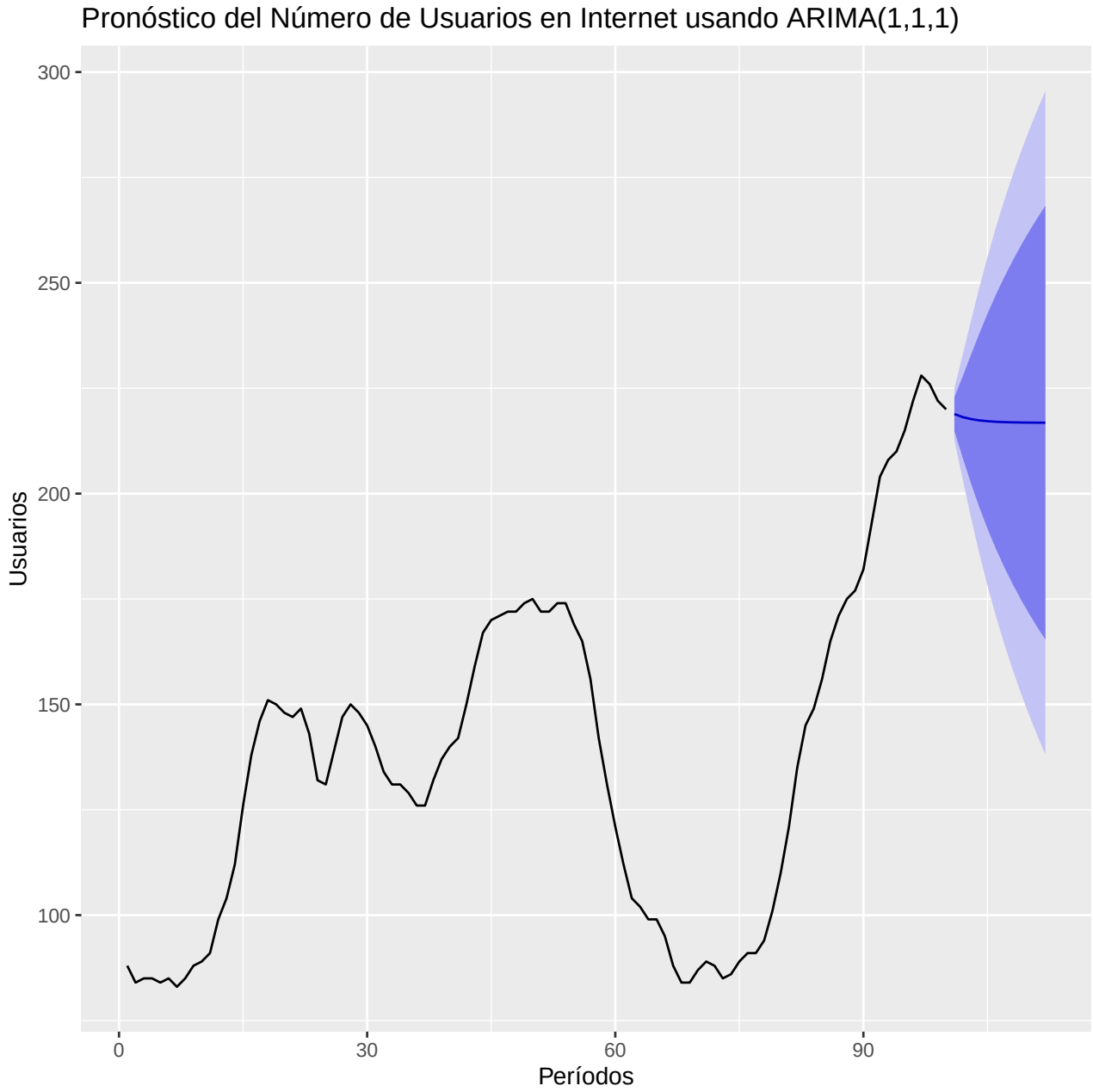
cual nos permite visualizar si los residuos siguen la distribución de una variable simulada normal. Para esto utilizaremos el siguiente código:

```
qqnorm(ModA_ARIMA$residuals)
qqline(ModA_ARIMA$residuals, col = "2")
```



Podemos observar que la gran mayoría de puntos se alinean dentro de los datos simulados normales, de tal manera que se confirma que los residuos de la serie presentan una distribución norma, con una media igual a cero y una varianza constante. Con estas pruebas podemos confirmar que los pronósticos que arrojara el modelo serán los más eficientes posibles. Por último, podemos realizar el pronóstico de la serie "Usuarios" de Internet utilizandoo el modelo ARMA seleccionado. Para realizar el modelo, tenemos que utilizar la función *forecast()* que se encuenta dentro de la paquetería **"forecast"**. Debemos de especificar dentro de la función el modelo ARIMA que desarrollamos y que revisamos que cumpliera los puestos de estabilidad y de los residuos. Se deberá especificar el número de períodos que se busca pronosticar considerando la periodicidad de la serie; es decir, si buscamos pronosticar a 1 año la serie, debemos especificar 12 períodos, ya que existen 12 meses dentro del año. Procedente a eso, utilizaremos la función *autoplot()* para visualizar la gráfica sobre el pronósico que realizamos, así como los intervalos de confianza del pronóstico. El código que ocuparemos para realizar el pronóstico es el siguiente:

```
pronostico <- forecast(ModA_ARIMA, 12)
autoplot(pronostico)+
labs(title = "Pronóstico del Número de Usuarios en Internet usando ARIMA(1,1,1)",
     x="Períodos", fill = "Niv.Conf.")+theme_gray()
```



De acuerdo con los resultados del modelo, se pronostica que habrá un leve decrecimiento en el número de usuarios, seguido de un comportamiento constante después de unas 3 observaciones o meses posteriores.También podemos apreciar los intervalos de confianza de la serie (Tanto un 90 % como un 85 % de confianza), los cuales nos ayudarpan como un segundo criterio de posibles valores a futuros de la serie (Ya que significa que el valor real de la serie no podrá superar esos intervalos con 85 % o un 90 % de probabilidad.