Documentación sobre autoajuste de modelos ARIMAX

Ana Xiangning Pereira Ezquerro

Versión 13 julio, 2022

Índice

1 Función de auto-ajuste de modelos ARIMAX (`auto.fit.arima` en `auto_fit_arima.R`)

Descripción: Obtiene el ajuste de un modelo válido para una serie temporal y, opcionalmente, una o varias variables regresoras. En el ajuste obtenido todos los parámetros son estadísticamente significativos y se verifica que se cumplen las hipótesis de independencia y media nula sobre sus residuos. Este ajuste es escogido por un criterio de información que se introduce como argumento.

Devuelve:

a. Ajuste para la serie temporal (objeto `Arima`) si existe y se puede optimizar.
b. `NA` en caso de que no exista o no se pueda optimizar.
c. Si `plot_result = TRUE` y se ha conseguido ajustar un modelo válido para la serie, devuelve un objeto de tipo `list` donde se encuentra el ajuste (`$ajuste`), el gráfico de la serie (`$fig_serie`) y el gráfico de los residuos del ajuste (`$fig_residuals`).

```
auto.fit.arima(serie, xregs = NULL, seasonal = TRUE, ic = c("aicc", "aic", "bic"),
               d = NA, D = NA, alpha = 0.05, show_info = TRUE, plot_result = FALSE)
```

Argumentos:

- `serie` [`ts`]: Serie temporal sobre la que se quiere obtener un ajuste válido de un modelo ARIMAX.
- `xregs` [`ts`]: Se pueden introducir series de tiempo que actuarán como variables regresoras sobre `serie`. Por defecto, `xregs=NULL`, i.e. no hay variables regresoras.
- `ic` [`character`]: Criterio de información para escoger modelos.
  - `"aicc"`: Criterio de Información de Akaike Corregido (por defecto).
  - `"aic"`: Criterio de Información de Akaike.
  - `"bic"`: Criterio de Información Bayesiano.
- `d` [`numeric`]: Orden de diferenciación regular de `serie` sobre el que se limita la búsqueda de modelos. Si no se introduce ningún valor el valor máximo de la búsqueda es `d=4`.
- `D` [`numeric`]: Orden de de diferenciación estacional de `serie` sobre el que se limita la búsqueda de modelos. Si no se introduce ningún valor el valor máximo de la úsqueda es `D=3`.
- `alpha` [`numeric`]: Valor entre 0 y 1 que indica el nivel de significación de los tests para chequear:
  - La significación de los parámetros de los ajustes.
  - La validez del modelo a partir del test de independencia de residuos y el test de media nula de los residuos.
- `show_info` [`boolean`]: Indica si se muestra la información de la búsqueda del mejor ajuste o no. Por defecto `TRUE`.
- `plot_results` [`boolean`]: Indica si se deben devolver los gráficos de la serie temporal y los residuos del modelo obtenido. Por defecto `FALSE`.

Consideraciones:

- Para chequear la independencia de residuos se utiliza el contraste de Ljung-Box (`Box.test`). El

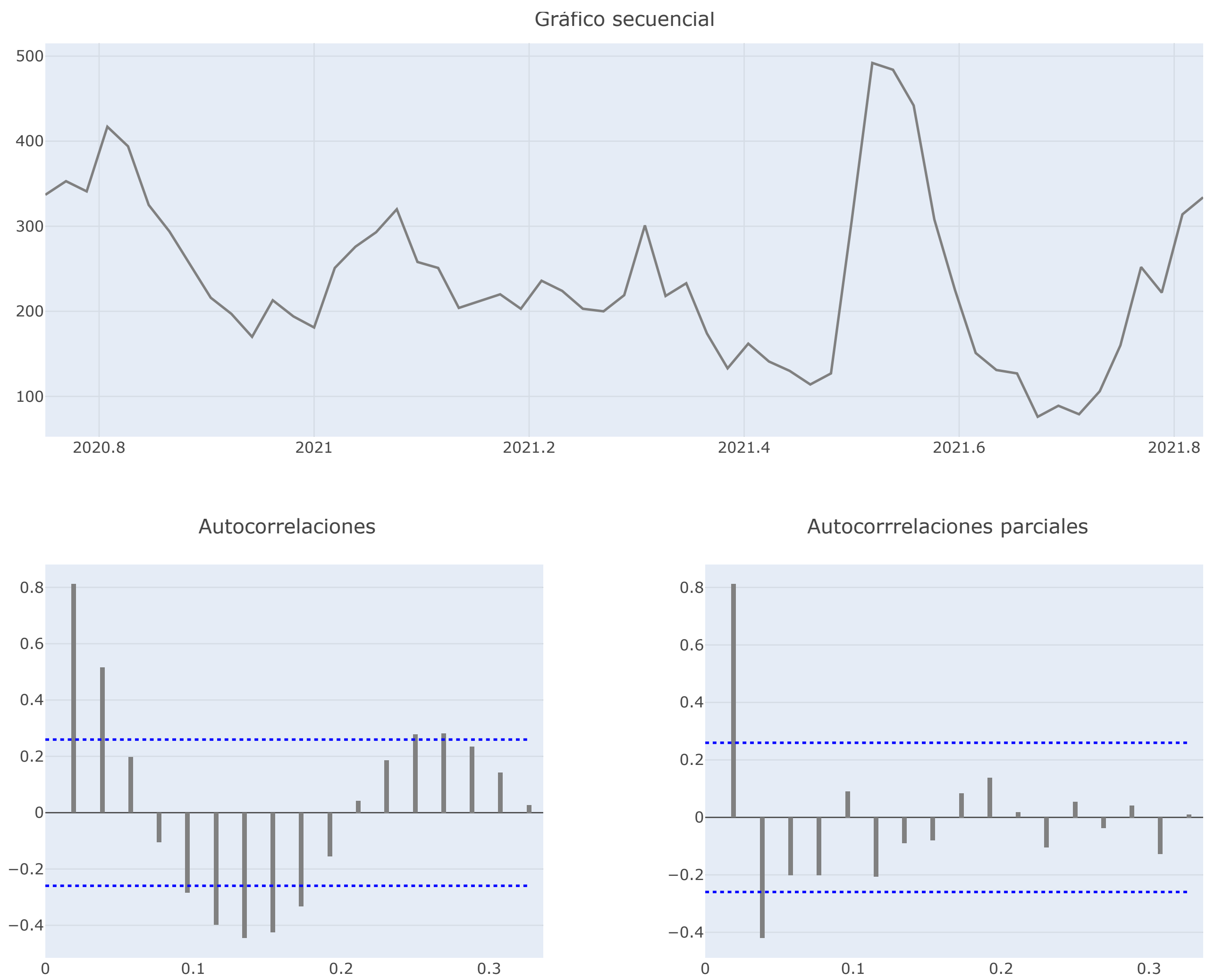
número de retardos se escoge en base a la estacionalidad de la serie (si la hay) y la longitud de la misma (función `ljungbox_lag`).
- Para chequear la media nula de los residuos se utiliza el `t.test`.
- Para chequear la normalidad de los residuos se utilizar el test de Jarque-Bera (`jarque.bera.test`) y el de Shapiro-Wilks (`shapiro.test`).
- Los modelos considerados tendrán siempre un orden de diferenciación regular igual o inferior a 3 ($d \leq 3$) y un orden de diferenciación estacional menor o igual a 2 ($D \leq 2$).

Ejemplo de uso: Evolución de la gripe en Cataluña.

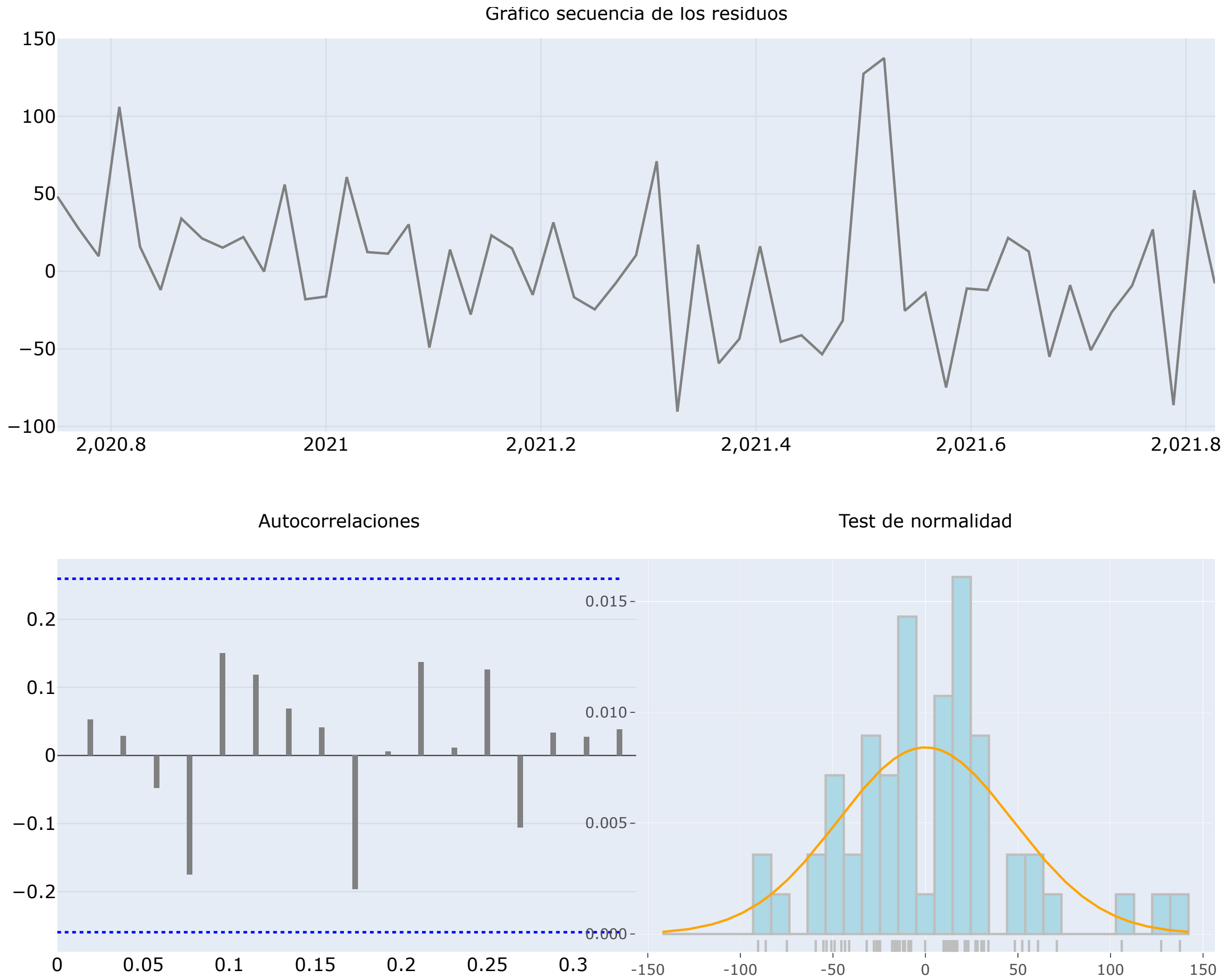
```r
dat <- read.csv("data/evolucion_gripe_covid.csv")
gripe <- ts(dat$sdgripal, start=c(2020, 40), frequency=52)
result_gripe <- auto.fit.arima(gripe, plot_result = TRUE)
```

```
-----------------------------------------------------------------------------------------
Series: serie
ARIMA(2,0,1) with non-zero mean

Coefficients:
         ar1      ar2      ma1      mean
      1.7011  -0.8606  -0.7378  233.5599
s.e.  0.1023   0.0832   0.1715   10.7434

sigma^2 = 2245:  log likelihood = -299.83
AIC=609.66   AICc=610.83   BIC=619.87
-----------------------------------------------------------------------------------------
Falla la hipótesis de normalidad sobre los residuos.
El modelo es válido pero los intervalos de predicción basados en la
dist. asintótica no son válidos
-----------------------------------------------------------------------------------------
|                                      MODELO FINAL                                     |
-----------------------------------------------------------------------------------------
Series: serie
ARIMA(2,0,1) with non-zero mean

Coefficients:
         ar1      ar2      ma1      mean
      1.7011  -0.8606  -0.7378  233.5599
s.e.  0.1023   0.0832   0.1715   10.7434

sigma^2 = 2245:  log likelihood = -299.83
AIC=609.66   AICc=610.83   BIC=619.87
```

```r
display(result_gripe$fig_serie, "serie gripe", width=1000, height=800)
```




```
display(result_gripe$fig_residuals, "residuals gripe", width=1000, height=800)
```



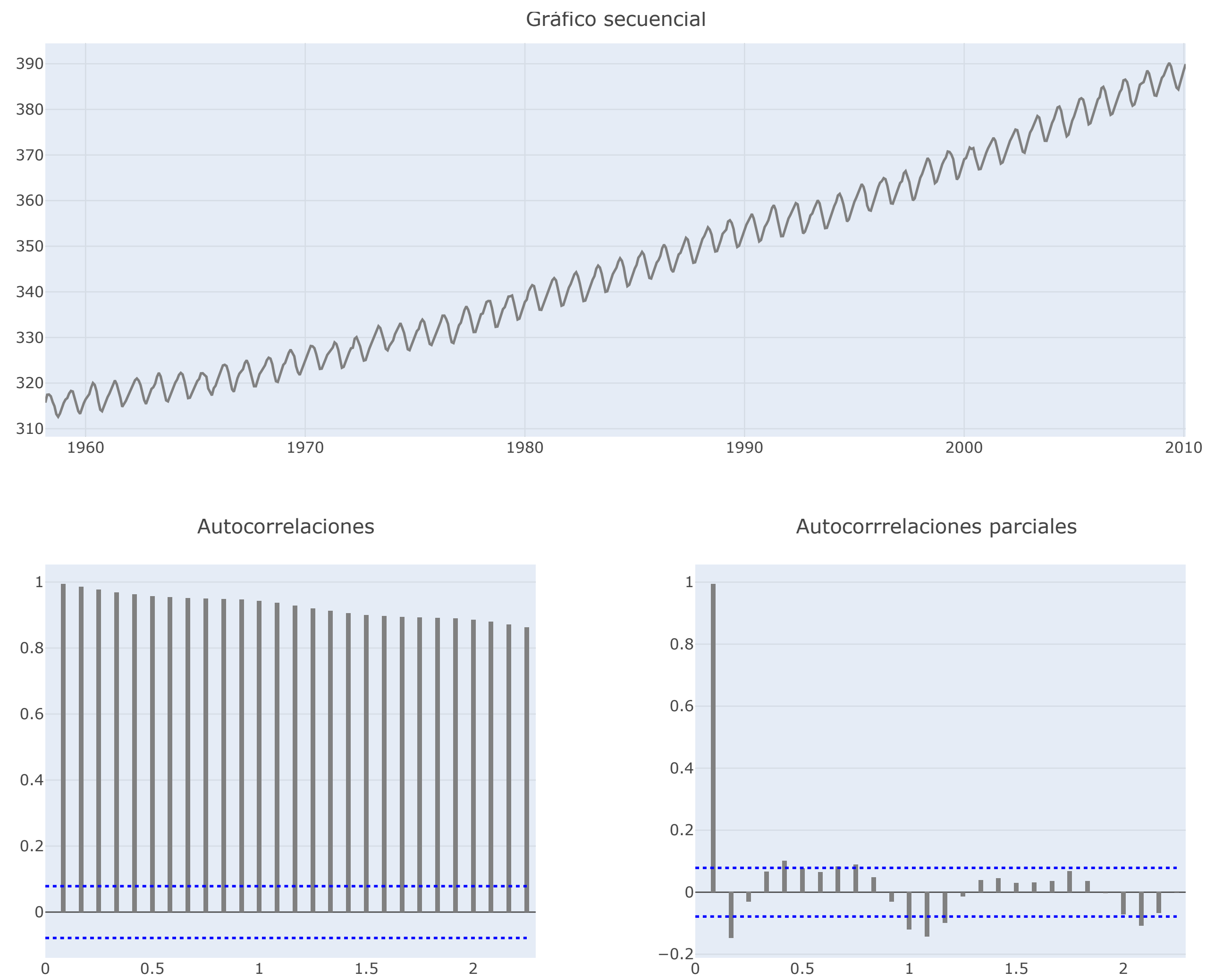
Ejemplo de uso: Nivel mensual de dióxido de carbono (Co2) medido en el Observatorio de Mauna Loa (Hawaii). La serie comienza en Marzo de 1958.

```
co2 <- ts(scan('data/co2MaunaLoa.dat'), start=c(1958, 3), frequency=12)
result_co2 <- auto.fit.arima(co2, ic="aicc", plot_result=TRUE)
```

```
-----------------------------------------------------------------------------------------
Series: serie
ARIMA(1,1,1)(0,1,1)[12]

Coefficients:
         ar1      ma1     sma1
      0.1645  -0.5210  -0.8684
s.e.  0.1048   0.0909   0.0208

sigma^2 = 0.09136:  log likelihood = -135.78
AIC=279.57   AICc=279.63   BIC=297.23
-----------------------------------------------------------------------------------------
Es necesario retirar del modelo el parámetro: ar1
-----------------------------------------------------------------------------------------
Series: serie
ARIMA(0,1,1)(0,1,1)[12]

Coefficients:
```

```
          ma1     sma1
      -0.3783  -0.8684
s.e.   0.0415   0.0209

sigma^2 = 0.09155:  log likelihood = -136.93
AIC=279.87   AICc=279.91   BIC=293.11
------------------------------------------------------------------------------------------
Falla la hipótesis de normalidad sobre los residuos.
El modelo es válido pero los intervalos de predicción basados en la
dist. asintótica no son válidos
------------------------------------------------------------------------------------------
|                                      MODELO FINAL                                      |
------------------------------------------------------------------------------------------
Series: serie
ARIMA(0,1,1)(0,1,1)[12]

Coefficients:
          ma1     sma1
      -0.3783  -0.8684
s.e.   0.0415   0.0209

sigma^2 = 0.09155:  log likelihood = -136.93
AIC=279.87   AICc=279.91   BIC=293.11
```

```r
display(result_co2$fig_serie, "serie co2", width=1000, height=800)
```




```
display(result_co2$fig_residuals, "residuals co2", width=1000, height=800)
```

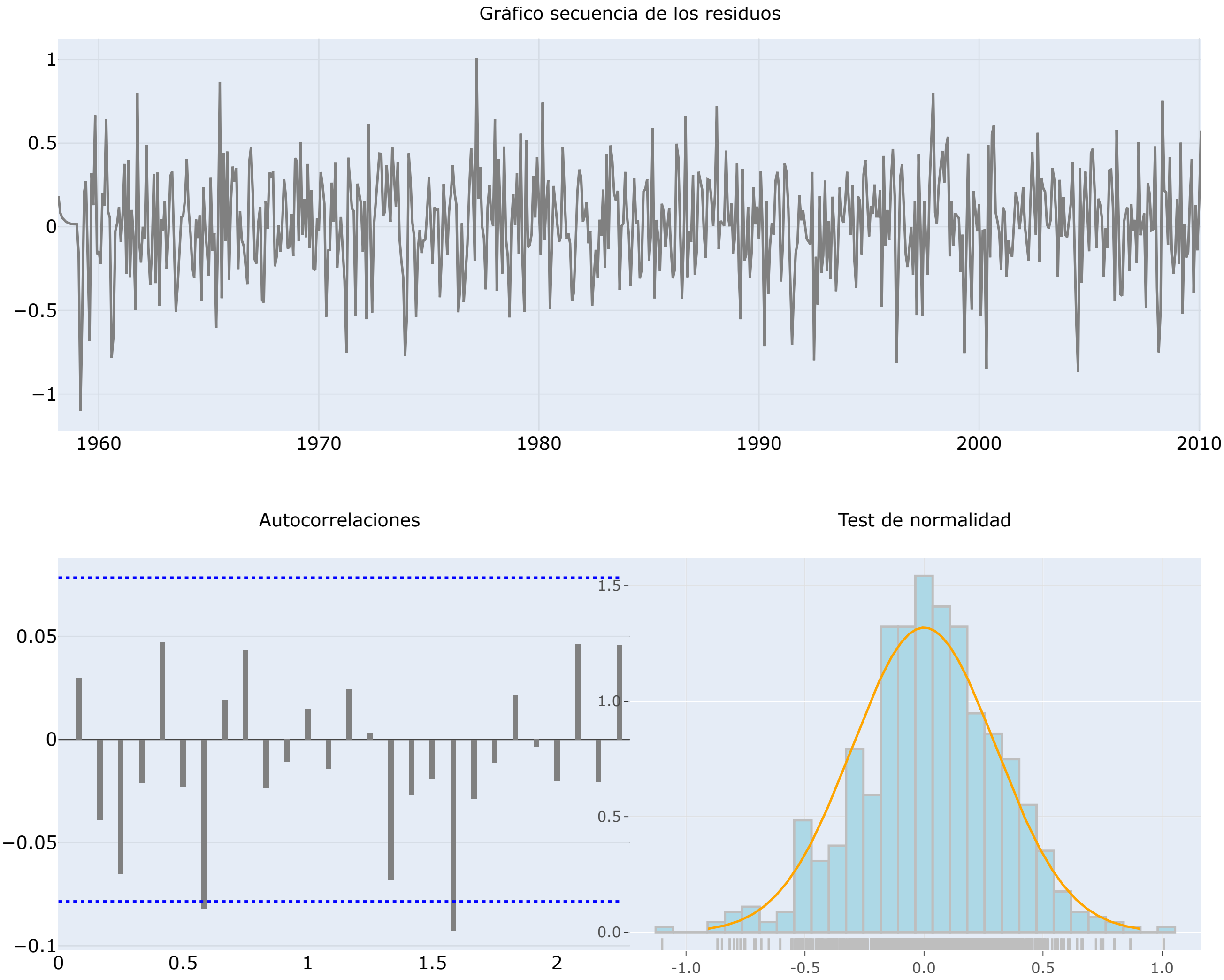


Gráfico secuencia de los residuos
1
0.5
0
−0.5
−1
1960
1970
1980
1990
2000
2010
Autocorrelaciones
0.05
0
−0.05
−0.1
0
0.5
1
1.5
2
Test de normalidad
1.5
1.0
0.5
0.0
-1.0
-0.5
0.0
0.5
1.0

2 Función de selección automática de múltiples variables y retardos en modelos ARIMAX (`auto.fit.arima.regression` en `automatic_selection.R`)

Descripción: Método de selección las variables regresoras y sus respectivos retardos (óptimos) para una serie de tiempo en base al método propuesto por Cryer y Chan (2008).

Devuelve:

a. Un objeto de tipo `list` donde se almacena el ajuste de un modelo válido de todas las variables regresoras (`$ajuste`, objeto `Arima`) que se han seleccionado para modelar la variable respuesta y el número de diferenciaciones regulares que se han aplicado sobre los datos para que los errores del ajuste sean estacionarios (`$ndiff`).
b. `NA` en caso de que no se haya podido ajustar ningún modelo (incluso uno sin variables regresoras).

```
auto.fit.arima.regression(serie, xregs, ic = c("aicc", "aic", "bic"),
                          alpha = 0.05, stationary_method='auto.arima',
                          show_info = TRUE, ndiff=0)
```

Argumentos:

- `serie` [ts]: Serie temporal que funciona como variable respuesta en el modelo de regresión dinámico sobre el que se realiza la selección de variables regresoras.
- `xregs` [`data.frame`]: Dataframe con las series temporales que actuarán como variables regresoras de `serie`. Es importante que los nombres de las columnas tengan un significado de cara a identificar las variables regresoras.
- `alpha` [numeric]: Valor entre 0 y 1 que indica el nivel de significación de los tests para chequear:
  - La significación de los parámetros de los ajustes.
  - La validez del modelo a partir del test de independencia de residuos y el test de media nula de residuos.
  - La selección de retardos óptimos.
  - La comprobación de tendencia de las series.
- `stationary_method` [character]: Método utilizado para chequear la estacionariedad de una serie temporal en las fases de preblanqueado (técnica usada para eliminar la correlación espuria entre dos series). Si `stationary_method = 'auto.arima'`, se utiliza la función `forecast::auto.arima` para ajustar un modelo ARIMA(p,d,q) y chequear si $d > 0$ (si se cumple esta condición se asume que la serie no es estacionaria). Si `stationary_method = 'adf.test'` se usa el test Dickey-Fuller (`tseries::adf.test`) para chequear la estacionariedad de una serie temporal.
- `show_info` [boolean]: Indica si se muestra la información de la selección de variables o no.
- `ndiff` [numeric]: Parámetro interno del programa (no utilizar) para diferenciar todas las variables cuando no se pueda ajustar un modelo válido con errores estacionarios y mantener un registro del número de diferenciaciones que se están realizando. Nótese que cuando, en la salida de la

función, el valor de `$ndiff` es mayor a 0, se han aplicado `ndiff` diferencias a los datos (tanto a la variable respuesta como a las regresoras) y por tanto el modelo que se devuelve en `$ajuste` se trata de un modelo de diferencias, no sobre los datos originales.

Nota: No se mostrará la información del ajuste de cada modelo para cada variable regresora.

Ejemplo de uso: Logaritmo de las ventas semanales y el precio de patatas fritas *Bluebird* de NUeva Zelanda. El período de observación es de 104 semanas (desde el 20 de Septiembre de 1988 hasta el 10 de Septiembre de 2000).

```
load("data/patatas.dat")
Y <- patatas[,1]
X <- patatas[,2]
ajuste_patatas <- auto.fit.arima.regression(Y, data.frame(X=X))
```

```
Se ha probado con la variable X [ic=-71.4371307570267, lag=0]
Se ha añadido la variable regresora X [aicc=-71.4371307570267]
Series: serie
Regression with ARIMA(0,0,4) errors

Coefficients:
      ma1     ma2  ma3     ma4  intercept     xreg
        0  0.2884    0  0.5416    15.8559  -2.4682
s.e.    0  0.0794    0  0.1167     0.1909   0.1100

sigma^2 = 0.02728:  log likelihood = 41.02
AIC=-72.05   AICc=-71.44   BIC=-58.83
------------------------------------------------------------------------------------------
No se añaden más variables

------------------------------------------------------------------------------------------
|                  Histórico de variables añadidas al modelo (ndiff=0)                   |
------------------------------------------------------------------------------------------
 var lag                ic
   X   0 -71.4371307570267
------------------------------------------------------------------------------------------
Series: serie
Regression with ARIMA(0,0,4) errors

Coefficients:
      ma1     ma2  ma3     ma4  intercept     xreg
        0  0.2884    0  0.5416    15.8559  -2.4682
s.e.    0  0.0794    0  0.1167     0.1909   0.1100

sigma^2 = 0.02728:  log likelihood = 41.02
AIC=-72.05   AICc=-71.44   BIC=-58.83
```

Ejemplo de uso: Serie temporal sobre el *stock* de Microsoft.

```
microsoft <- read.csv('data/microsoft-stock.csv')
close_price <- ts(microsoft$Close)   # variable respuesta
regresoras <- as.data.frame(
    lapply(microsoft[, c('Open', 'High', 'Low', 'Volume')], ts))
```

```r
elapsed_time <- system.time(
    ajuste <- auto.fit.arima.regression(close_price, regresoras)
)
```

```
Se ha probado con la variable Open [ic=5924.58405199719, lag=0]
No se ha podido ajustar un modelo para High
No se ha podido ajustar un modelo para Low
No se ha podido ajustar un modelo para Volume
Se ha añadido la variable regresora Open [aicc=5924.58405199719]
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    xreg
         0.0645  0.9997
s.e.     0.0947  0.0008

sigma^2 = 2.954:  log likelihood = -2959.28
AIC=5924.57   AICc=5924.58   BIC=5940.53
------------------------------------------------------------------------------------------
Saltamos Open
Se ha probado con la variable High [ic=5921.57447994466, lag=-1]
Se ha probado con la variable Low [ic=5924.33558667892, lag=-1]
No se ha podido ajustar un modelo para Volume
Se ha añadido la variable regresora High [aicc=5921.57447994466]
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    Open    High
         0.0748  0.9461  0.0531
s.e.     0.0947  0.0239  0.0237

sigma^2 = 2.946:  log likelihood = -2956.77
AIC=5921.55   AICc=5921.57   BIC=5942.83
------------------------------------------------------------------------------------------
Saltamos Open
Saltamos High
Se ha probado con la variable Low [ic=5923.5637198765, lag=-1]
No se ha podido ajustar un modelo para Volume
No se añaden más variables

------------------------------------------------------------------------------------------
|                  Histórico de variables añadidas al modelo (ndiff=0)                   |
------------------------------------------------------------------------------------------
  var lag               ic
 Open   0 5924.58405199719
 High  -1 5921.57447994466
------------------------------------------------------------------------------------------
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
```

```
      intercept    Open    High
         0.0748  0.9461  0.0531
s.e.     0.0947  0.0239  0.0237

sigma^2 = 2.946:  log likelihood = -2956.77
AIC=5921.55   AICc=5921.57   BIC=5942.83
```

```r
print(elapsed_time)  # tiempo secuencial
```

```
   user  system elapsed
 256.88    1.50  258.48
```

```r
eval(parse("parallel.R", encoding="UTF-8"))
elapsed_time <- system.time(
    ajuste <- auto.fit.arima.regression(close_price, regresoras, show_info=F)
)
```

```
------------------------------------------------------------------------------------------
|                  Histórico de variables añadidas al modelo (ndiff=0)                    |
------------------------------------------------------------------------------------------
  var lag              ic
 Open   0 5924.58405199719
 High  -1 5921.57447994466
------------------------------------------------------------------------------------------
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    Open    High
         0.0748  0.9461  0.0531
s.e.     0.0947  0.0239  0.0237

sigma^2 = 2.946:  log likelihood = -2956.77
AIC=5921.55   AICc=5921.57   BIC=5942.83
```

```r
print(elapsed_time)
```

```
   user  system elapsed
  31.98    0.31  183.36
```

Ejemplo de uso: Modelización de la serie de tiempo de muertes en España debido al COVID19, considerando como posibles variables regresoras:

- Los casos confirmados y curados en España.
- Los casos confirmados y muertes en Francia.
- Los casos confirmados y muertes en Inglaterra.

```r
confirmed <- read.csv("data/covid-global-confirmed-bycountry.csv")
deaths <- read.csv("data/covid-global-deaths-bycountry.csv")
recovered <- read.csv("data/covid-global-recovered-bycountry.csv")

confirmed_spain <- ts(confirmed$Spain, frequency=7)
deaths_spain <- ts(deaths$Spain, frequency=7)
```

```r
recovered_spain <- ts(recovered$Spain, frequency=7)

confirmed_france <- ts(confirmed$France, frequency=7)
confirmed_england <- ts(confirmed$United.Kingdom, frequency=7)

deaths_france <- ts(deaths$France, frequency=7)
deaths_england <- ts(deaths$United.Kingdom, frequency=7)

regresoras <- data.frame(confirmed_spain, recovered_spain)

ajuste <- auto.fit.arima.regression(deaths_spain, regresoras)
```

```
Se ha probado con la variable confirmed_spain [ic=5853.12673996867, lag=0]
Se ha probado con la variable recovered_spain [ic=5907.4070401008, lag=-1]
Se ha añadido la variable regresora confirmed_spain [aicc=5853.12673996867]
Series: serie
Regression with ARIMA(0,2,1)(1,0,1)[7] errors

Coefficients:
          ma1    sar1     sma1    xreg
      -0.7844  0.9026  -0.7783  0.0074
s.e.   0.0285  0.0761   0.1152  0.0011

sigma^2 = 31436:  log likelihood = -2921.49
AIC=5852.99   AICc=5853.13   BIC=5873.46
------------------------------------------------------------------------------------------
Se ha probado con la variable recovered_spain [ic=5847.9488914573, lag=-1]
Se ha añadido la variable regresora recovered_spain [aicc=5847.9488914573]
Series: serie
Regression with ARIMA(1,1,1)(1,0,1)[7] errors

Coefficients:
         ar1      ma1    sar1     sma1  confirmed_spain  recovered_spain
      0.9588  -0.7734  0.9008  -0.7712           0.0075           0.0303
s.e.  0.0170   0.0363  0.0725   0.1123           0.0011           0.0185

sigma^2 = 30901:  log likelihood = -2916.85
AIC=5847.69   AICc=5847.95   BIC=5876.35
------------------------------------------------------------------------------------------
No se ha podido encontrar un modelo válido con errores estacionarios
Se aplica una diferenciación regular (ndiff=1) y se vuelve a llamar a la función
------------------------------------------------------------------------------------------
Se ha probado con la variable confirmed_spain [ic=5853.12674283582, lag=0]
Se ha probado con la variable recovered_spain [ic=5907.40705113048, lag=-1]
Se ha añadido la variable regresora confirmed_spain [aicc=5853.12674283582]
Series: serie
Regression with ARIMA(0,1,1)(1,0,1)[7] errors

Coefficients:
          ma1    sar1     sma1    xreg
      -0.7844  0.9026  -0.7783  0.0074
s.e.   0.0285  0.0761   0.1152  0.0011
```

```
sigma^2 = 31436:  log likelihood = -2921.49
AIC=5852.99   AICc=5853.13   BIC=5873.46
------------------------------------------------------------------------------------------
Se ha probado con la variable recovered_spain [ic=5840.74083575298, lag=-1]
Se ha añadido la variable regresora recovered_spain [aicc=5840.74083575298]
Series: serie
Regression with ARIMA(0,1,1)(1,0,1)[7] errors

Coefficients:
          ma1    sar1     sma1  confirmed_spain  recovered_spain
      -0.8003  0.9147  -0.8004           0.0074           0.0279
s.e.   0.0294  0.0684   0.1056           0.0010           0.0184

sigma^2 = 31410:  log likelihood = -2914.27
AIC=5840.55   AICc=5840.74   BIC=5865.1
------------------------------------------------------------------------------------------
No se añaden más variables

------------------------------------------------------------------------------------------
|                  Histórico de variables añadidas al modelo (ndiff=1)                   |
------------------------------------------------------------------------------------------
             var lag               ic
 confirmed_spain   0 5853.12674283582
 recovered_spain  -1 5840.74083575298
------------------------------------------------------------------------------------------
Series: serie
Regression with ARIMA(1,0,1)(1,0,1)[7] errors

Coefficients:
         ar1      ma1    sar1     sma1  confirmed_spain  recovered_spain
      0.9588  -0.7735  0.9014  -0.7722           0.0075           0.0305
s.e.  0.0170   0.0363  0.0722   0.1120           0.0011           0.0185

sigma^2 = 30901:  log likelihood = -2916.85
AIC=5847.69   AICc=5847.95   BIC=5876.35
```

```r
save(ajuste, file='ajuste covid.RData')
```

3 Funciones auxiliares

3.1 Ajuste de los coeficientes de un modelo (`fit.coefficients()` de `auto_fit_arima.R`)

Descripción: Elimina de forma incremental los coeficientes no significativos en un modelo.

Devuelve: Ajuste de un modelo donde todos sus coeficientes son significativamente distintos de cero.

```
fit.coefficients(ajuste, alpha=0.05, show_info=T)
```

Argumentos:

- `ajuste` [`Arima`]: Ajuste de un modelo ARIMA sobre el que se deben eliminar los coeficientes no significativos.
- `alpha` [`numeric`]: Valor entre 0 y 1 que especifica el nivel de significación para retirar parámetros del modelo. Por defecto es 5%.
- `show_info` [`boolean`]: Indica si se debe mostrar información sobre los parámetros que se van retirando del ajuste o no. Por defecto, va mostrando esta información en consola.

3.2 Ajuste de un ARIMA vía múltiples optimizadores (`fit.model()` de `auto_fit_arima.R`)

Descripción: Ajuste de un modelo ARIMA dados sus órdenes sobre una serie temporal, manejando posibles errores de optimización y probando con otros métodos en caso de que el que viene dado por defecto provoque errores. Los optimizadores con los que prueba son, en este orden: `BFGS`, `Nelder-Mead`, `CG`, `L-BFGS-B`, `SANN` y `Brent`.

Devuelve: Modelo ARIMA para los parámetros y serie temporal dada o `NA` en caso de que no haya sido posible ajustar ningún modelo por problemas de optimización.

```
fit.model(serie, orders, xregs=NULL, fixed=NULL)
```

Argumentos:

- `serie` [`Arima`]: Serie temporal sobre la que se ajusta el modelo ARIMA.
- `orders` [`list`]: Objeto de tipo lista donde se especifica información sobre los órdenes regulares y estacionales del modelo. El formato es el siguiente:
  - `orders$regular = c(p, d, q)` [`numeric`]: Especifica los órdenes regulares.
  - `orders$seasonal = c(P, D, Q)` [`numeric`]: Especifica los órdenes estacionales.
  - `orders$include_constant` [`boolean`]: Especifica si se debe incluir la media en un ajuste sin diferencias.
- `xregs` [`ts`]: Matriz de posibles variables regresoras.
- `fixed`: Vector de valores fijos para los coeficientes del modelo ARIMA que se quiere ajustar.

3.3 Selección del retardo óptimo (`select.optimal.lag()` de `automatic_selection.`

Descripción: Selección del retardo significativo y óptimo de dos series (asumiendo que una funciona como variable explicativa y otra como variable respuesta en un modelo de regresión con componente temporal). Esta selección se realiza siguiendo el procedimiento descrito por Cryer y Chan (2008) usando las funciones `tseries::adf.test()` o `auto.arima` para chequear estacionariedad, `seastests::isSeasonal` para chequear presencia de estacionalidad y `TSA::prewhiten()` para aplicar el preblanqueado sobre las dos series.

Devuelve: El retardo óptimo de las dos series o `NA` en caso de que ningún retardo sea significativo.

```
select.optimal.lag(serie, xreg, alpha=0.05, max_lag=NA)
```

Argumentos:

- `serie` [`ts`]: Serie temporal que funciona como variable respuesta.
- `xreg` [`ts`]: Variable regresora de `serie`.
- `alpha` [`numeric`]: Valor entre 0 y 1 que indica el nivel de significación para aceptar o no la hipótesis nulas en los contrastes de signficación, estacionariedad y estacionalidad.
- `max_lag` [`numeric` o `NA`]: Opcionalmente, se puede añadir un valor que limite el valor del retardo óptimo tal que su valor absoluto siempre sea menor que `max_lag`.
- `method` [`character`]: Selecciona el método para chequear estacionairedad sobre ambas series. Cuando se fija como `adf.test` se usa el test Dickey-Fuller y cunado se fija como `auto.arima` se ajusta un modelo ARIMA(p,d,q) con la función `forecast::auto.arima()` y se comprueba si $d > 0$.

4 Predicciones puntuales a horizonte h e intervalos de confianza (`forecasting_model()` de `forecasting.R`)

Una vez obtenido un ajuste del modelo de regresión dinámica, el objetivo es realizar predicciones a horizonte h con sus correspondientes intervalos de predicción. Partiendo del objeto `Arima` obtenido de la función `auto.fit.arima.regression` (con información acerca de las variables introducidas y sus retardos y las diferenciaciones aplicadas para conseguir errores estacionarios).

Para realizar predicciones a horizonte h a partir de un modelo de regresión es necesario tener predicciones de los valores de las variables regresoras a ese mismo horizonte. Estos valores se pueden obtener, para cada variable, a partir de:

- En caso de que el retardo r con el que se ha introducido la variable en el modelo sea igual o mayor a h, con los valores "sobrantes" de la serie original.
- En caso de que $0 \leq r < h$, ajustando un modelo ARIMA sobre la variable regresora y prediciendo $h - r$ valores.

Descripción: A partir del ajuste de un ARIMAX realiza predicciones puntuales a horizonte h de cada variable regresora para introducirlas en las predicciones puntuales de la variable respuesta.

Devuelve: Objeto `forecast` con las predicciones puntuales y los intervalos de confianza.

```
forecast_model(ajuste, h, mode='bootstrap', levels=c(80, 90))
```

Argumentos:

- `ajuste` [`Arima`]: Ajuste de un modelo de regresión con series temporales sobre el que se quieren hacer predicciones puntuales e intervalos de confianza.
- `h` [`numeric`]: Valor horizonte de las predicciones.
- `mode` [`character`]: Modo de realizar las predicciones: basadas en normalidad sobre los residuos (`norm`) o a través de *bootstrap* (`bootstrap`). Por defecto se realizan a través de bootstrap.
  - Si `mode='norm'` se intentarán realizar todas las predicciones (incluso las de las variables regresoras) basadas en normalidad, siempre y cuando se cumpla esa condición (chequeada con los tests de Jarque Bera y Shapiro Wilks).
- `levels` [`vector`]: Vector numérico de los niveles a los que se quieren hacer los intervalos de predicción.

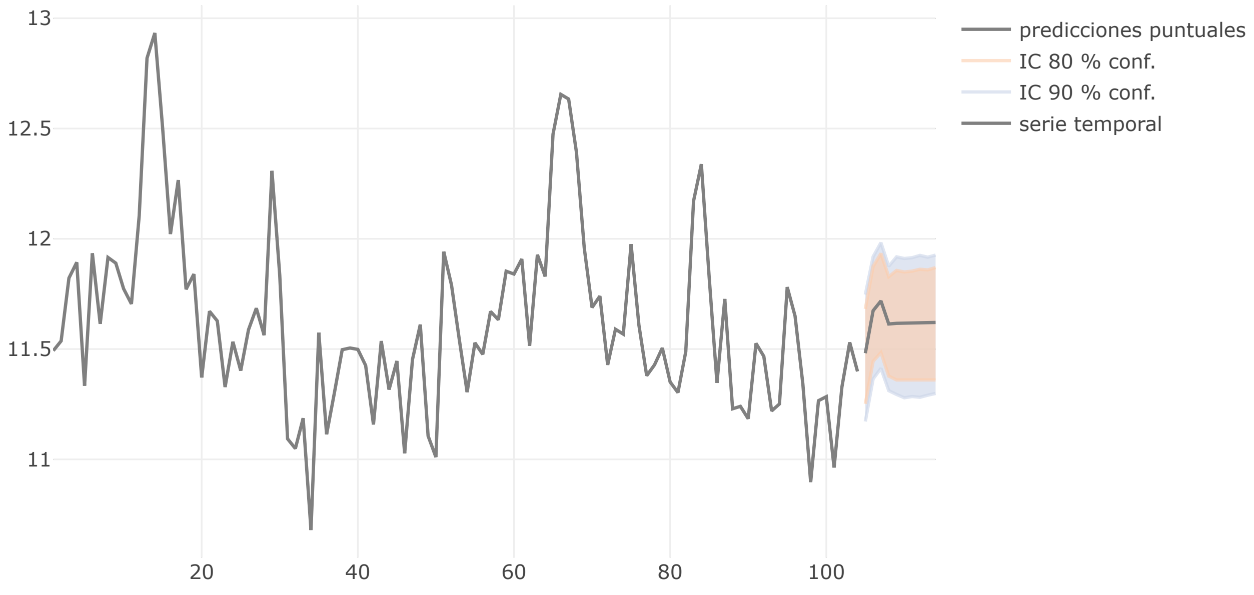
Ejemplo de uso:

```
load("data/patatas.dat")
Y <- ts(patatas[,1])
X <- ts(patatas[,2])
ajuste_patatas <- auto.fit.arima.regression(Y, data.frame(X=X))
```

```
Se ha probado con la variable X [ic=-71.4371307570267, lag=0]
Se ha añadido la variable regresora X [aicc=-71.4371307570267]
```

```
Series: serie
Regression with ARIMA(0,0,4) errors

Coefficients:
      ma1     ma2  ma3     ma4  intercept     xreg
        0  0.2884    0  0.5416    15.8559  -2.4682
s.e.    0  0.0794    0  0.1167     0.1909   0.1100

sigma^2 = 0.02728:  log likelihood = 41.02
AIC=-72.05   AICc=-71.44   BIC=-58.83
------------------------------------------------------------------------------------------
No se añaden más variables

------------------------------------------------------------------------------------------
|                  Histórico de variables añadidas al modelo (ndiff=0)                   |
------------------------------------------------------------------------------------------
 var lag                ic
   X   0 -71.4371307570267
------------------------------------------------------------------------------------------
Series: serie
Regression with ARIMA(0,0,4) errors

Coefficients:
      ma1     ma2  ma3     ma4  intercept     xreg
        0  0.2884    0  0.5416    15.8559  -2.4682
s.e.    0  0.0794    0  0.1167     0.1909   0.1100

sigma^2 = 0.02728:  log likelihood = 41.02
AIC=-72.05   AICc=-71.44   BIC=-58.83
```

```r
# Calculamos las predicciones puntuales
preds <- forecast_model(Y, data.frame(X=X), ajuste_patatas, h=10, mode='bootstrap')
display(plot_forecast(preds), name='preds_patatas')
```



5 Comprobación con ejemplos

5.1 Evolución de la gripe en Cataluña

```r
# Carga de datos
cataluna <- read.csv("data/evolucion_gripe_covid.csv")
str(cataluna)
```

```
'data.frame':	57 obs. of  39 variables:
 $ fecha               : chr  "2020-40" "2020-41" "2020-42" "2020-43" ...
 $ sdgripal            : int  337 353 341 417 394 325 294 254 216 197 ...
 $ sarscov2            : int  71 133 133 218 220 169 161 135 87 60 ...
 $ sdgripal.edad04     : int  52 35 40 65 48 49 43 33 32 28 ...
 $ sarscov2.edad04     : int  6 5 7 13 7 8 14 3 2 1 ...
 $ sdgripal.edad1544   : int  89 115 113 138 129 88 77 61 60 49 ...
 $ sarscov2.edad1544   : int  21 47 50 82 81 51 39 33 26 11 ...
 $ sdgripal.edad4564   : int  37 66 48 80 75 72 42 44 40 49 ...
 $ sarscov2.edad4564   : int  15 30 22 51 55 43 25 23 19 22 ...
 $ sdgripal.edad514    : int  139 103 108 105 108 87 78 51 44 54 ...
 $ sarscov2.edad514    : int  25 38 39 55 57 48 46 25 16 20 ...
 $ sdgripal.edad65     : int  20 34 32 29 34 29 54 65 40 17 ...
 $ sarscov2.edad65     : int  4 13 15 17 20 19 37 51 24 6 ...
 $ pob04               : int  5450 5450 5450 5450 5441 5438 5443 5436 5425 5426 ...
 $ pob514              : int  14270 14270 14270 14270 14262 14268 14266 14248 14242 14235 ...
 $ pob1544             : int  18428 18428 18428 18428 18431 18402 18386 18334 18328 18311 ...
 $ pob4564             : int  13825 13825 13825 13825 13841 13848 13854 13816 13817 13821 ...
 $ pob65               : int  8991 8991 8991 8991 8998 8995 8997 8982 8988 8987 ...
 $ pob                 : int  60964 60964 60964 60964 60973 60951 60946 60816 60800 60780 ...
 $ sdgripal.BARCELONA  : int  95 93 92 123 125 94 86 59 61 58 ...
 $ sarscov2.BARCELONA  : int  14 27 39 50 51 40 36 18 15 10 ...
 $ sdgripal.CANTALUNYA : int  37 49 50 56 35 34 31 23 21 26 ...
 $ sarscov2.CANTALUNYA : int  11 21 20 27 26 21 23 19 13 12 ...
 $ sdgripal.GIRONA     : int  16 27 30 36 28 24 17 12 23 9 ...
 $ sarscov2.GIRONA     : int  4 11 12 25 21 16 14 6 8 3 ...
 $ sdgripal.LLEIDA     : int  27 27 37 24 31 32 26 16 18 15 ...
 $ sarscov2.LLEIDA     : int  5 11 13 10 18 18 14 8 11 9 ...
 $ sdgripal.METR_NORD  : int  24 29 24 34 29 29 27 31 20 12 ...
 $ sarscov2.METR_NORD  : int  8 16 10 26 19 15 17 16 7 5 ...
 $ sdgripal.METR_SUD   : int  20 23 14 24 32 22 20 8 9 13 ...
 $ sarscov2.METR_SUD   : int  3 9 5 20 15 11 5 1 0 6 ...
 $ sdgripal.PIRINEU    : int  11 16 14 10 10 12 39 72 32 29 ...
 $ sarscov2.PIRINEU    : int  1 8 9 3 7 6 32 57 26 7 ...
 $ sdgripal.TARRAGONA  : int  35 48 44 67 62 44 28 18 22 18 ...
 $ sarscov2.TARRAGONA  : int  10 11 12 29 34 18 11 8 7 3 ...
 $ sdgripal.TERRES_EBRE: int  21 14 15 11 8 16 4 4 4 5 ...
 $ sarscov2.TERRES_EBRE: int  7 9 9 7 5 9 3 0 0 1 ...
 $ sdgripal.VALLES     : int  51 27 21 32 34 18 16 10 6 12 ...
 $ sarscov2.VALLES     : int  8 10 4 21 24 15 6 2 0 4 ...
```

El dataset `evolucion_gripe_covid.csv` contiene información sobre la evolución de la gripe y el COVID19 en las distintas áreas sanitarias de Cataluña y en toda la comunidad a lo largo del tiempo. Cada dato recogido representa el número de casos confirmados (de gripe y COVID19) en una semana (desde la 40ª semana de 2020 hasta la 46ª semana de 2021).

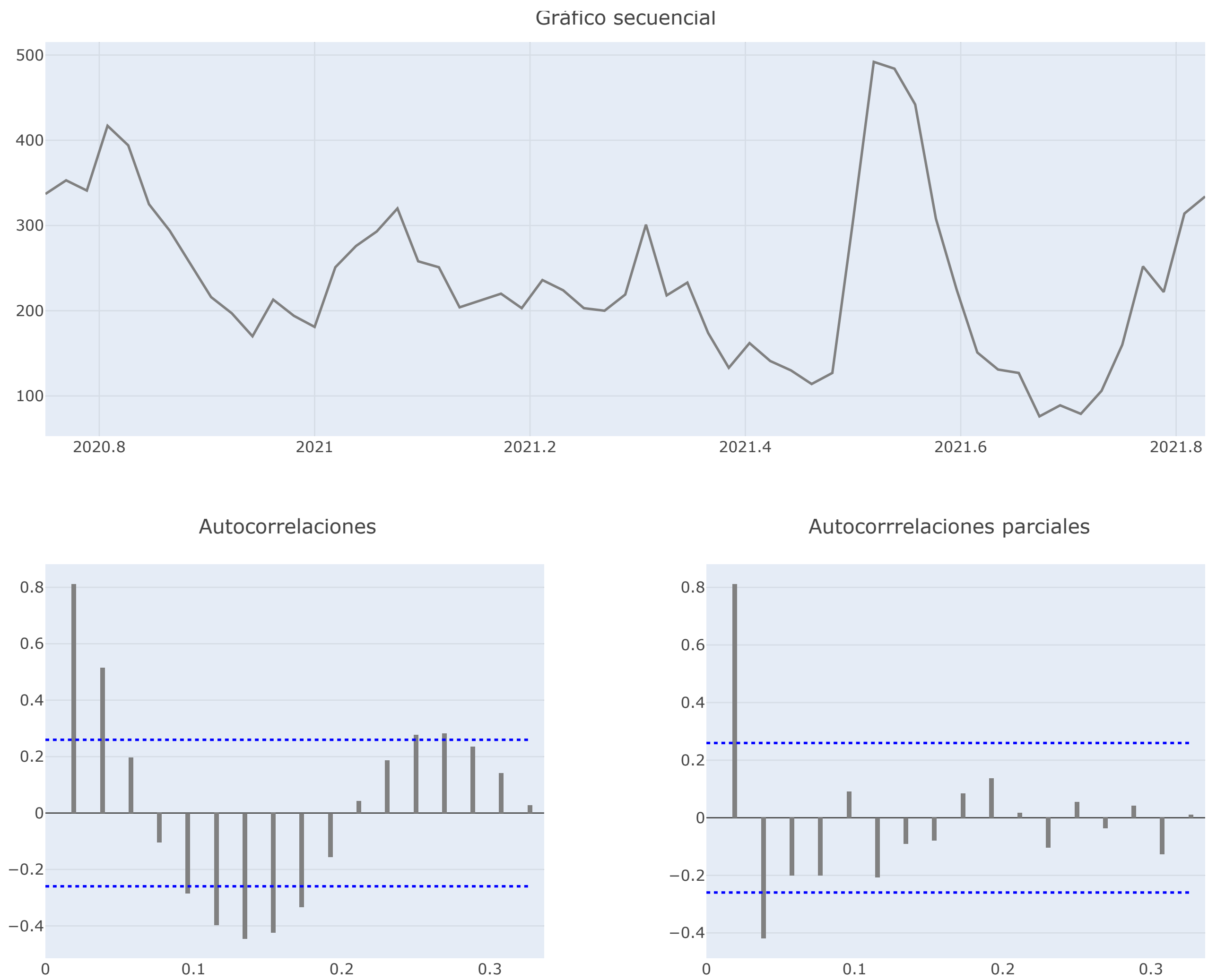
Vamos a intentar modelizar la evolución de la gripe con un ARIMA a través de los siguientes métodos:

- Usando la función `auto.arima` y ajustando los coeficientes para obtener un ajuste válido.
- Usando la función `auto.fit.arima` que realiza todo el proceso.

```
# Los datos ya están ordenados temporalmente
gripe <- ts(cataluna$sdgripal, start=c(2020, 40), frequency=52)
```

Analizamos el gráfico secuencial y la fas y fap muestral:

```
display(result_gripe$fig_serie, "serie gripe", width=1000, height=800)
```



A continuación, usamos la función `auto.arima`:

```
ajuste <- auto.arima(gripe, stepwise=FALSE, approximation=FALSE, trace=TRUE)
```

```
 ARIMA(0,0,0)           with zero mean     : 795.7079
 ARIMA(0,0,0)           with non-zero mean : 686.4374
 ARIMA(0,0,1)           with zero mean     : 731.2285
```

```
 ARIMA(0,0,1)            with non-zero mean : 641.3142
 ARIMA(0,0,2)            with zero mean     : 697.1511
 ARIMA(0,0,2)            with non-zero mean : 628.1215
 ARIMA(0,0,3)            with zero mean     : Inf
 ARIMA(0,0,3)            with non-zero mean : 616.4541
 ARIMA(0,0,4)            with zero mean     : Inf
 ARIMA(0,0,4)            with non-zero mean : 616.4276
 ARIMA(0,0,5)            with zero mean     : 647.1342
 ARIMA(0,0,5)            with non-zero mean : 617.7512
 ARIMA(1,0,0)            with zero mean     : 628.8489
 ARIMA(1,0,0)            with non-zero mean : 624.0469
 ARIMA(1,0,1)            with zero mean     : 625.7308
 ARIMA(1,0,1)            with non-zero mean : 618.9305
 ARIMA(1,0,2)            with zero mean     : 625.3825
 ARIMA(1,0,2)            with non-zero mean : 616.9578
 ARIMA(1,0,3)            with zero mean     : 627.3056
 ARIMA(1,0,3)            with non-zero mean : 615.7902
 ARIMA(1,0,4)            with zero mean     : Inf
 ARIMA(1,0,4)            with non-zero mean : 618.3407
 ARIMA(2,0,0)            with zero mean     : 624.5081
 ARIMA(2,0,0)            with non-zero mean : 614.2113
 ARIMA(2,0,1)            with zero mean     : 626.7644
 ARIMA(2,0,1)            with non-zero mean : 610.832
 ARIMA(2,0,2)            with zero mean     : 627.6806
 ARIMA(2,0,2)            with non-zero mean : 613.1555
 ARIMA(2,0,3)            with zero mean     : 628.5378
 ARIMA(2,0,3)            with non-zero mean : 618.1807
 ARIMA(3,0,0)            with zero mean     : 626.7033
 ARIMA(3,0,0)            with non-zero mean : 613.0392
 ARIMA(3,0,1)            with zero mean     : 628.9774
 ARIMA(3,0,1)            with non-zero mean : 613.1275
 ARIMA(3,0,2)            with zero mean     : 627.6101
 ARIMA(3,0,2)            with non-zero mean : 615.5654
 ARIMA(4,0,0)            with zero mean     : 627.7034
 ARIMA(4,0,0)            with non-zero mean : 614.977
 ARIMA(4,0,1)            with zero mean     : Inf
 ARIMA(4,0,1)            with non-zero mean : 615.6291
 ARIMA(5,0,0)            with zero mean     : 627.1917
 ARIMA(5,0,0)            with non-zero mean : 617.5599



 Best model: ARIMA(2,0,1)            with non-zero mean
```

Y comprobamos que el mejor modelo (siguiendo el AICc) es un ARIMA(2, 0, 1) con media.

```
ajuste
```

```
Series: gripe
ARIMA(2,0,1) with non-zero mean

Coefficients:
         ar1      ar2      ma1      mean
      1.7011  -0.8606  -0.7378  233.5599
s.e.  0.1023   0.0832   0.1715   10.7434
```

```
sigma^2 = 2245:  log likelihood = -299.83
AIC=609.66   AICc=610.83   BIC=619.87
```

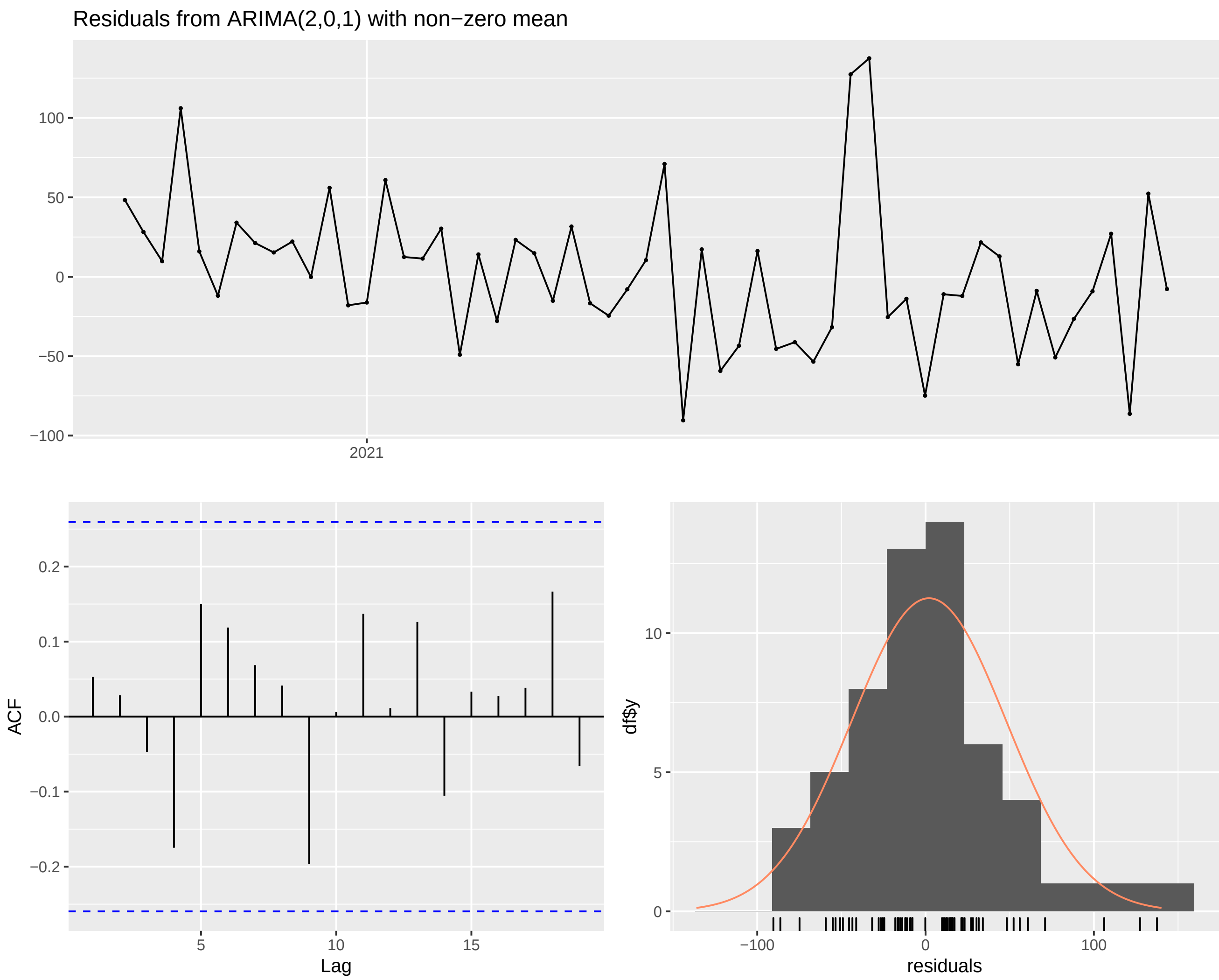
A continuación, comprobamos qué parámetros *no* son significativos:

```
alpha <- 0.05; stat <- qnorm(1-0.05/2)
abs(ajuste$coef) < stat*sqrt(diag(ajuste$var.coef))
```

```
      ar1       ar2       ma1 intercept
    FALSE     FALSE     FALSE     FALSE
```

En este caso, **todos** los parámetros son significativos y por tanto se trata de un ajuste válido. Finalmente, realizamos el análisis de residuos para chequear las hipótesis de independencia y media nula.

```
checkresiduals(ajuste)
```




```
    Ljung-Box test

data:  Residuals from ARIMA(2,0,1) with non-zero mean
Q* = 9.1953, df = 7, p-value = 0.2389

Model df: 4.   Total lags used: 11
```

```r
t.test(ajuste$residuals, mu=0)
```

```

	One Sample t-test

data:  ajuste$residuals
t = 0.32866, df = 56, p-value = 0.7436
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -10.21437  14.22381
sample estimates:
mean of x 
 2.004717 
```

El test de independencia de Ljung-Box y el test de media nula nos dicen que los residuos sí son independientes y tienen media cero, por tanto se puede considerar que el ajuste es válido para modelizar la evolución de la gripe.

El objetivo de la función `auto.fit.arima` es realizar todo este proceso de forma automática. El resultado que nos devuelva debe ser el mismo que el que hemos obtenido haciendo los cálculos paso a paso:

```r
ajuste <- auto.fit.arima(gripe)
```

```
------------------------------------------------------------------------------------------
Series: serie 
ARIMA(2,0,1) with non-zero mean 

Coefficients:
         ar1      ar2      ma1      mean
      1.7011  -0.8606  -0.7378  233.5599
s.e.  0.1023   0.0832   0.1715   10.7434

sigma^2 = 2245:  log likelihood = -299.83
AIC=609.66   AICc=610.83   BIC=619.87
------------------------------------------------------------------------------------------
Falla la hipótesis de normalidad sobre los residuos.
El modelo es válido pero los intervalos de predicción basados en la
dist. asintótica no son válidos
------------------------------------------------------------------------------------------
|                                     MODELO FINAL                                       |
------------------------------------------------------------------------------------------
Series: serie 
ARIMA(2,0,1) with non-zero mean 

Coefficients:
         ar1      ar2      ma1      mean
      1.7011  -0.8606  -0.7378  233.5599
s.e.  0.1023   0.0832   0.1715   10.7434

sigma^2 = 2245:  log likelihood = -299.83
AIC=609.66   AICc=610.83   BIC=619.87
```

Adicionalmente, la función `auto.fit.arima` nos avisa de que los residuos no siguen una distribución normal, por tanto tendremos que tener cuidado al hacer predicciones sobre la serie.

5.2 Selección de covariables para predecir el precio de cierre en el *stock* de Microsoft

Cargamos la base de datos y separamos la variable respuesta (`response`) del conjunto de variables regresoras (`Open`, `High`, `Low`, `Volume`).

```
microsoft <- read.csv('data/microsoft-stock.csv')
str(microsoft)
```

```
'data.frame':	1511 obs. of  6 variables:
 $ Date  : chr  "4/1/2015 16:00:00" "4/2/2015 16:00:00" "4/6/2015 16:00:00" "4/7/2015 16:00:00"
 $ Open  : num  40.6 40.7 40.3 41.6 41.5 ...
 $ High  : num  40.8 40.7 41.8 41.9 41.7 ...
 $ Low   : num  40.3 40.1 40.2 41.3 41 ...
 $ Close : num  40.7 40.3 41.5 41.5 41.4 ...
 $ Volume: int  36865322 37487476 39223692 28809375 24753438 25723861 28022002 30276692 2424438
```

```
close <- ts(microsoft$Close)
open <- ts(microsoft$Open)
high <- ts(microsoft$High)
low <- ts(microsoft$Low)
volume <- ts(microsoft$Volume)
```

Inicialización: Selección de un retardo máximo permitido para las variables regresoras que se van añadiendo al modelo.

Asumiendo que se dispone de un conjunto de series temporales (del cual una de ellas funciona como variable respuesta y el resto como candidatas a variables regresoras) de tamaño T, en cada iteración del algoritmo de añade una nueva variable regresora con retardo r y se compara su criterio de información con un modelo de regresión más sencillo que no tiene esa variable (el modelo anterior del algoritmo). Para que estas comparaciones iterativas se realicen con el mismo número de datos, es necesario recortar todas las series por el áximo retardo permitido (este valor se denomina como `max_lag` en el código).

La función `get.maximum.lag()` aproxima este valor como el máximo (en valor absoluto) de los retardos significativos de todas las variables candidatas a regresoras, sin la restricción de que este sea menor o igual a 0. Formalmente:

$$\mathcal{R} = \max_{i=1\cdots m} \left\{ \left| \arg\max_{k \in \mathbb{Z}} \{ |\rho_k(X_i, Y)| \} \right| \right\}$$

donde $\rho_k(X,Y)$ es la correlación entre dos procesos preblanqueados X e Y en el retardo k.

Para obtener estos retardos para cada variable se debe aplicar el **proceso de preblanqueado** entre cada par (X_i, Y) para eliminar la correlación espuria y obtener los retardos donde se produce una correlación significativa. Realizamos este proceso para la variable regresora `Open`:

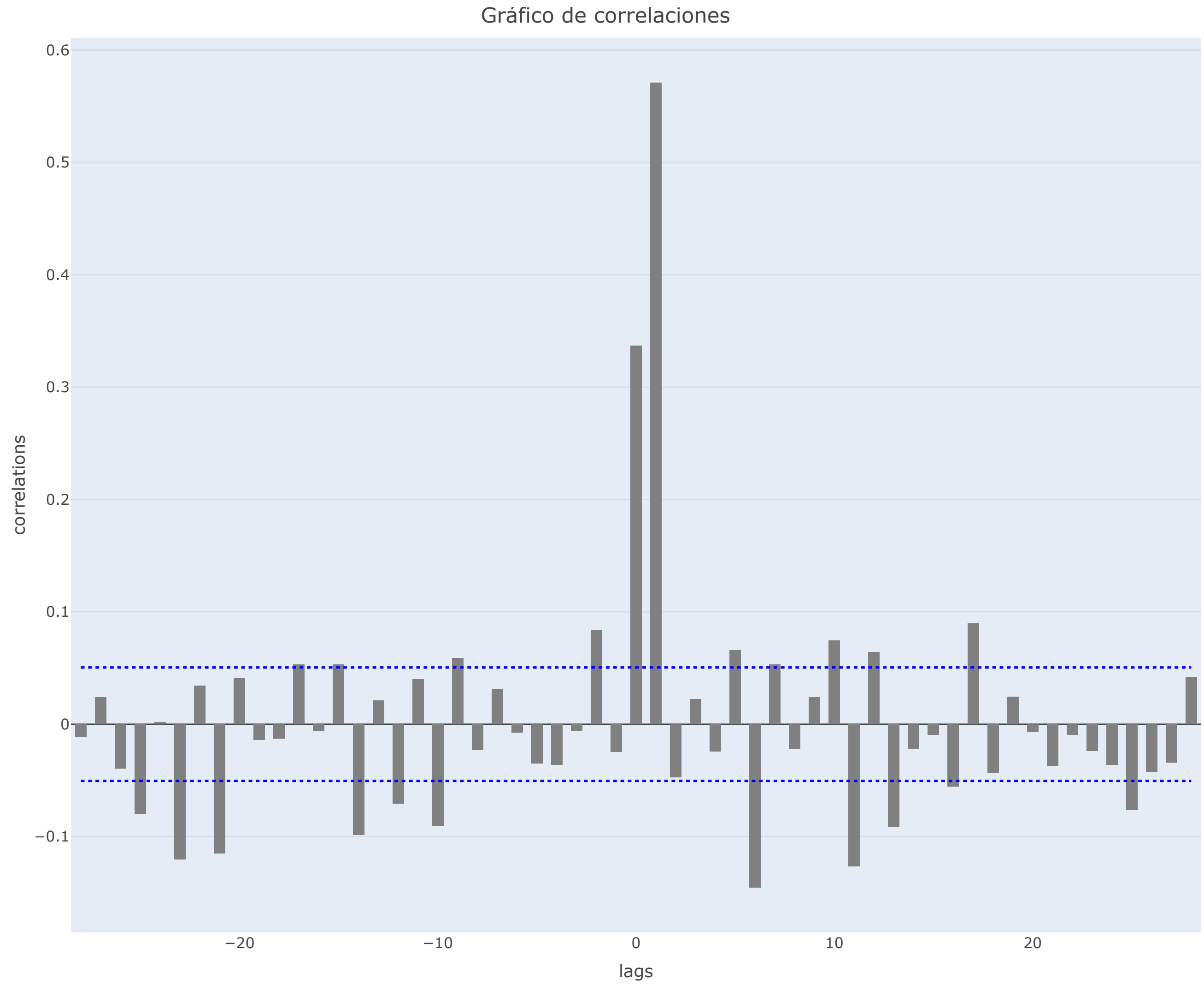
```r
open_diff <- open; close_diff <- close

# Chequear estacionariedad con el adf.test y diferenciar hasta eliminar la
# estacionariedad en ambas
adf.test(open_diff); adf.test(close_diff)
```

```
	Augmented Dickey-Fuller Test

data:  open_diff
Dickey-Fuller = -1.6497, Lag order = 11, p-value = 0.7266
alternative hypothesis: stationary

	Augmented Dickey-Fuller Test

data:  close_diff
Dickey-Fuller = -1.6209, Lag order = 11, p-value = 0.7388
alternative hypothesis: stationary
```

```r
open_diff <- diff(open_diff); close_diff <- diff(close_diff)

# Aplicando una única diferenciación ya se obtienen dos variables sin estacionariedad
adf.test(open_diff); adf.test(close_diff)
```

```
	Augmented Dickey-Fuller Test

data:  open_diff
Dickey-Fuller = -12.319, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary

	Augmented Dickey-Fuller Test

data:  close_diff
Dickey-Fuller = -11.873, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary
```

```r
# Realizamos el preblanqueado
series_prewhiten <- TSA::prewhiten(open_diff, close_diff, plot=F)

corr_plot <- plot_prewhiten(series_prewhiten)
display(corr_plot, "fig-open0", width=1000, height=800)
```




```
cat(paste0('Correlación máxima con la variable Open en el retardo: ',
           series_prewhiten$ccf$lag[which.max(abs(series_prewhiten$ccf$acf))]))
```

```
Correlación máxima con la variable Open en el retardo: 1
```

Como podemos apreciar en el gráfico, la mayor correlación significativa (en valor absoluto) se encuentra en $\text{lag} = 1$. A la hora de añadir variables al modelo, este retardo no sería válido para introducir esta variable regresora (pues es mayor a 0). No obstante, para fijar un valor para $\mathcal{R}$ sí se tiene en cuenta.

Si repetimos este proceso para las otras 3 variables:

```
# Preblanqueado con la variable high
high_diff <- high; close_diff <- close
adf.test(high_diff); adf.test(close_diff)
```

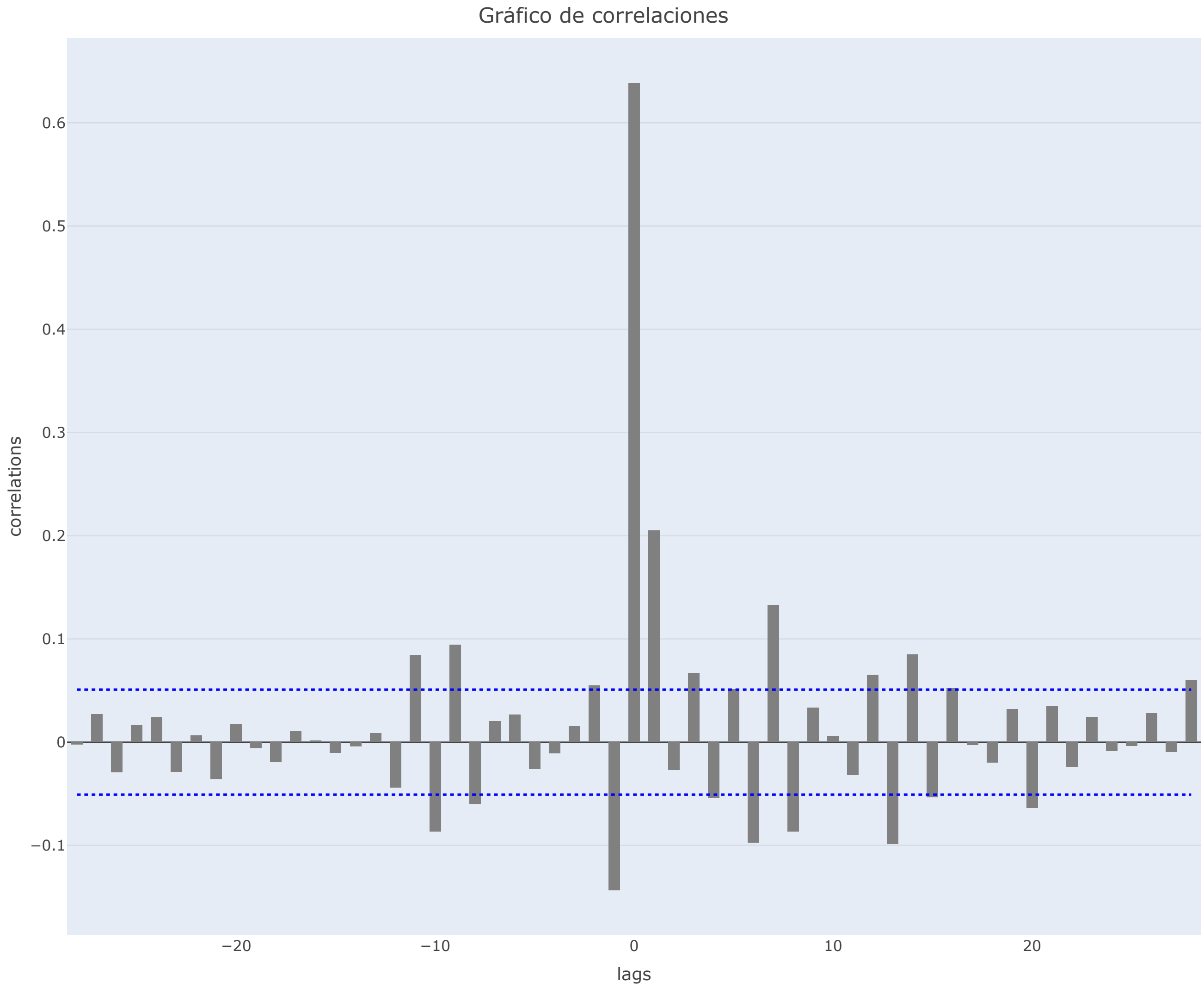
```
    Augmented Dickey-Fuller Test

data:  high_diff
Dickey-Fuller = -1.3936, Lag order = 11, p-value = 0.835
alternative hypothesis: stationary


    Augmented Dickey-Fuller Test
```

```
data:  close_diff
Dickey-Fuller = -1.6209, Lag order = 11, p-value = 0.7388
alternative hypothesis: stationary
```

```
high_diff <- diff(high_diff); close_diff <- diff(close_diff)
adf.test(high_diff); adf.test(close_diff)
```

```

    Augmented Dickey-Fuller Test

data:  high_diff
Dickey-Fuller = -12.332, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary


    Augmented Dickey-Fuller Test

data:  close_diff
Dickey-Fuller = -11.873, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary
```

```
series_prewhiten <- TSA::prewhiten(high_diff, close_diff, plot=F)
corr_plot <- plot_prewhiten(series_prewhiten)
display(corr_plot, "fig-high0", width=1000, height=800)
```




```r
cat(paste0('Correlación máxima con la variable High en el retardo: ',
           series_prewhiten$ccf$lag[which.max(abs(series_prewhiten$ccf$acf))], '\n'))
```

```
Correlación máxima con la variable High en el retardo: 0
```

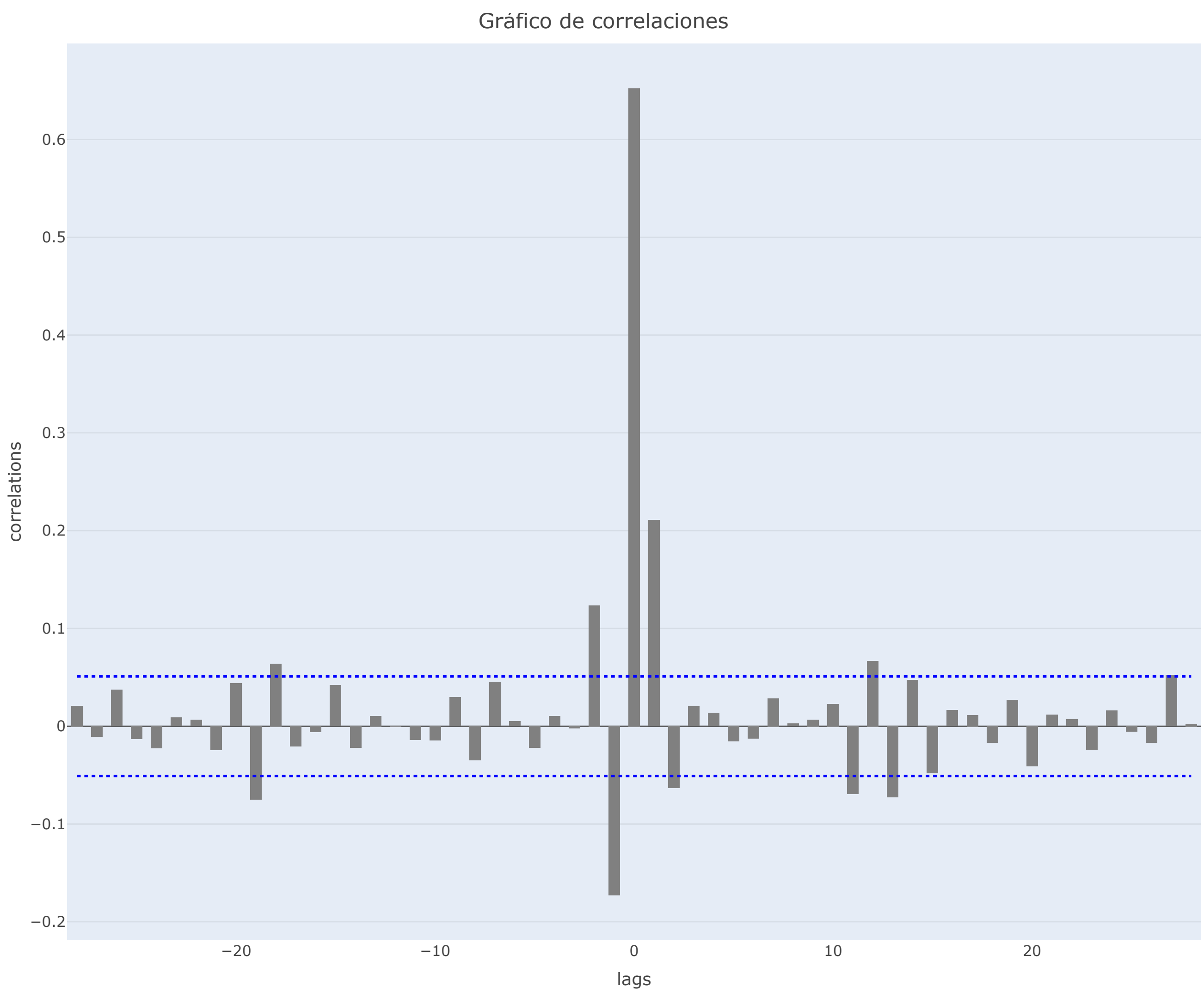
```r
# Preblanqueado con la variable low
low_diff <- low; close_diff <- close
adf.test(low_diff); adf.test(close_diff)
```

```

	Augmented Dickey-Fuller Test

data:  low_diff
Dickey-Fuller = -1.7039, Lag order = 11, p-value = 0.7037
alternative hypothesis: stationary


	Augmented Dickey-Fuller Test

data:  close_diff
Dickey-Fuller = -1.6209, Lag order = 11, p-value = 0.7388
alternative hypothesis: stationary
```

```r
low_diff <- diff(low_diff); close_diff <- diff(close_diff)
adf.test(low_diff); adf.test(close_diff)
```

```
    Augmented Dickey-Fuller Test

data:  low_diff
Dickey-Fuller = -11.633, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary


    Augmented Dickey-Fuller Test

data:  close_diff
Dickey-Fuller = -11.873, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary
```

```
series_prewhiten <- TSA::prewhiten(low_diff, close_diff, plot=F)
corr_plot <- plot_prewhiten(series_prewhiten)
display(corr_plot, "fig-low0", width=1000, height=800)
```




```
cat(paste0('Correlación máxima con la variable Low en el retardo: ',
           series_prewhiten$ccf$lag[which.max(abs(series_prewhiten$ccf$acf))], '\n'))
```

```
Correlación máxima con la variable Low en el retardo: 0
```

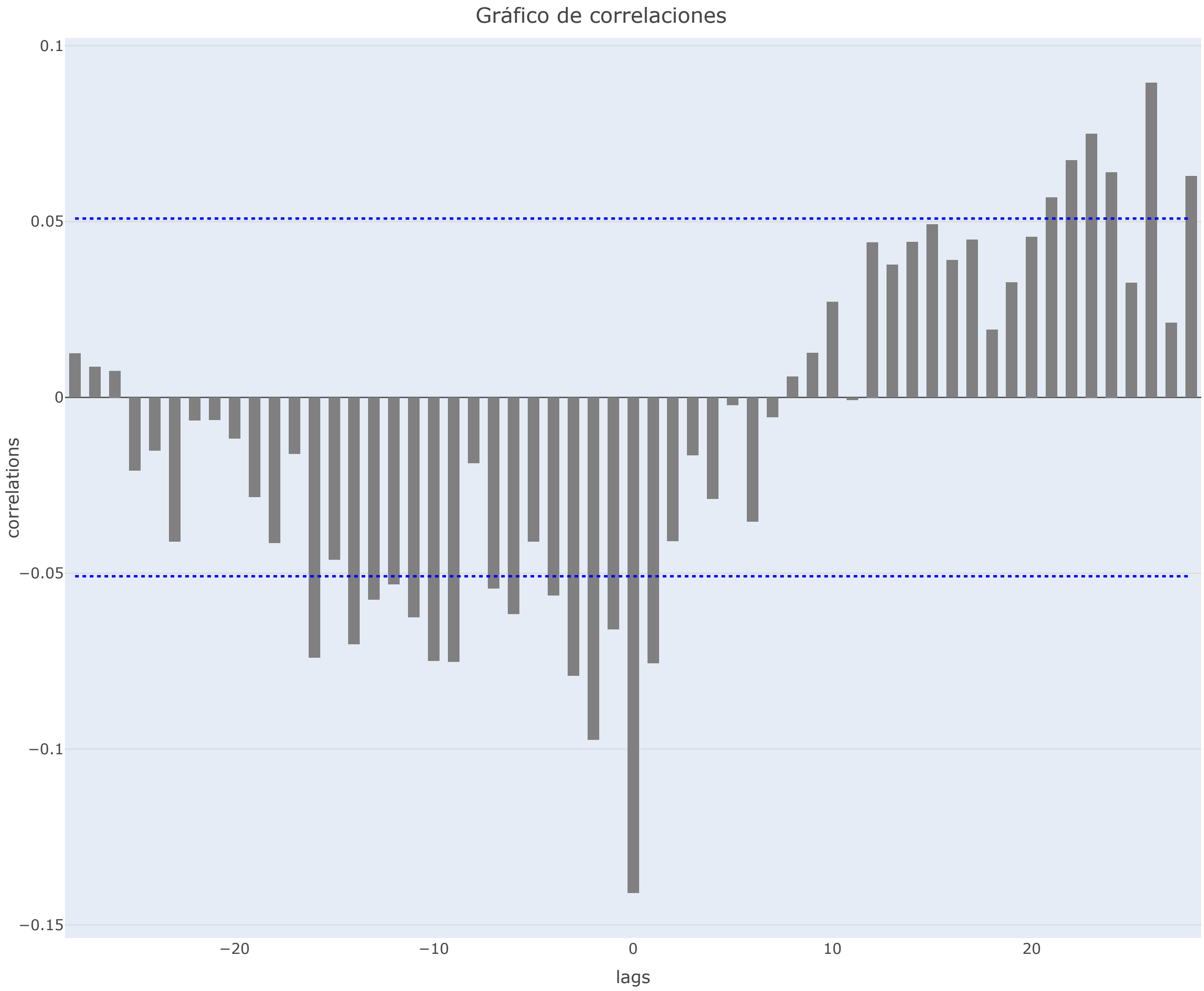
```r
# Preblanqueado con la variable volume
volume_diff <- volume; close_diff <- close
adf.test(volume_diff); adf.test(close_diff)
```

```
	Augmented Dickey-Fuller Test

data:  volume_diff
Dickey-Fuller = -6.2957, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary

	Augmented Dickey-Fuller Test

data:  close_diff
Dickey-Fuller = -1.6209, Lag order = 11, p-value = 0.7388
alternative hypothesis: stationary
```

```r
volume_diff <- diff(volume_diff); close_diff <- diff(close_diff)
adf.test(volume_diff); adf.test(close_diff)
```

```
	Augmented Dickey-Fuller Test

data:  volume_diff
Dickey-Fuller = -16.344, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary

	Augmented Dickey-Fuller Test

data:  close_diff
Dickey-Fuller = -11.873, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary
```

```r
series_prewhiten <- TSA::prewhiten(volume_diff, close_diff, plot=F)
corr_plot <- plot_prewhiten(series_prewhiten)
display(corr_plot, "fig-volume0", width=1000, height=800)
```




```
cat(paste0('Correlación máxima con la variable Volume en el retardo: ',
           series_prewhiten$ccf$lag[which.max(abs(series_prewhiten$ccf$acf))], '\n'))
```

```
Correlación máxima con la variable Volume en el retardo: 0
```

En el resto de variables, la máxima correlación (en términos absolutos) se produce en $k = 0$, por lo que $\mathcal{R} = 1$. Tendremos por tanto que recortar las series un total de $\mathcal{R}$ valores para tener el mismo número de datos con los que comparar los modelos.

Nota: Las series se recortan a la hora de ajustar el modelo con la función `auto.fit.arima()`, pero se utilizan todos los datos para escoger el retardo en el que se produce la máxima correlación (función `select.optimal.lag()`).

Fase 1: En la fase de inicialización ya tenemos los retardos óptimos calculados (todos iguales a 0). A continuación, procedemos a ajustar un modelo ARIMA (sin la restricción de estacionariedad sobre sus residuos) para cada variable candidata a regresora. Podemos ajustar este modelo a mano (utilizando la función `stats::auto.arima` y retirando los coeficientes no significativos), pero para este caso utilizaremos la función ya descrita anteriormente, `auto.fit.arima`, que realiza todo este proceso de forma automática.

Nota: En este caso utilizaremos como criterio de información el AICc.

```r
max_lag <- 1

# función para recortar max_lag valores de las series temporales
cut <- function(x) {return(window(x, start=(start(x)[1]+max_lag)))}

close_cut <- cut(close)
open_cut <- cut(open)
high_cut <- cut(high)
low_cut <- cut(low)
volume_cut <- cut(volume)



# Ajuste de un modelo con la variable open
ajuste_open <- auto.fit.arima(close_cut, xregs=cbind(open=open_cut), show_info=F)
ajuste_open
```

```
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    xreg
         0.0645  0.9997
s.e.     0.0947  0.0008

sigma^2 = 2.954:  log likelihood = -2959.28
AIC=5924.57   AICc=5924.58   BIC=5940.53
```

```r
# Ajuste de un modelo con la variable high
ajuste_high <- auto.fit.arima(close_cut, xregs=cbind(high=high_cut), show_info=F)
ajuste_high
```

```
[1] NA
```

```r
# Ajuste de un modelo con la variable low
ajuste_low <- auto.fit.arima(close_cut, xregs=cbind(low=low_cut), show_info=F)
ajuste_low
```

```
[1] NA
```

```r
# Ajuste de un modelo con la variable volume
ajuste_volume <- auto.fit.arima(close_cut, xregs=cbind(volume=volume_cut), show_info=F)
ajuste_volume
```

```
[1] NA
```

Los resultados obtenidos en esta primera iteración son los siguientes:

- El mejor modelo para la variable `Open` es un ARIMA(0,0,0) con AICc=5924.58 y retardo nulo.
- No se puede encontrar un modelo para la variable `High` con retardo nulo.
- No se puede encontrar un modelo para la variable `Low` con retardo nulo.
- No se puede encontrar un modelo para la variable `Volume` con retardo nulo.

Como el "mejor" criterio de información se obtiene al añadir la variable Open al modelo, se toma dicho ajuste y se calculan los residuos. Estos residuos pasarán a ser la variable respuesta y se repetirá el proceso en busca de nuevas variables a añadir al modelo.

```
response <- residuals(ajuste_open, type='regression')
```

Fase 2: Búsqueda de variables a añadir al modelo tomando como variable respuesta los residuos del modelo anterior.

Aplicamos de nuevo el preblanqueado a las variables que no se han introducido en el modelo (i.e. todas excepto Open):

```
alpha <- 0.05       # nivel de significación
trend <- function(x) {
    return(suppressWarnings(adf.test(na.remove(x))$p.value) >= alpha)
}

# Preblanqueado para la variable high
high_diff <- high; response_diff <- response
while (any(apply(cbind(high_diff, response_diff), 2, trend))) {
    high_diff <- diff(high_diff)
    response_diff <- diff(response_diff)
}
series_prewhiten <- TSA::prewhiten(high_diff, response_diff, plot=F)
lags <- series_prewhiten$ccf$lag[series_prewhiten$ccf$lag <= 0]
corrs <- abs(series_prewhiten$ccf$acf[series_prewhiten$ccf$lag <= 0])
cat(paste0('Correlación máxima con la variable high en el retardo: ',
           lags[which.max(corrs)], '\n'))
```

```
Correlación máxima con la variable high en el retardo: -1
```

```
# Preblanqueado con la variable low
low_diff <- low; response_diff <- response
while (any(apply(cbind(low_diff, response_diff), 2, trend))) {
    low_diff <- diff(low_diff)
    response_diff <- diff(response_diff)
}
series_prewhiten <- TSA::prewhiten(low_diff, response_diff, plot=F)
lags <- series_prewhiten$ccf$lag[series_prewhiten$ccf$lag <= 0]
corrs <- abs(series_prewhiten$ccf$acf[series_prewhiten$ccf$lag <= 0])
cat(paste0('Correlación máxima con la variable low en el retardo: ',
           lags[which.max(corrs)], '\n'))
```

```
Correlación máxima con la variable low en el retardo: -1
```

```
# Preblanqueado con la variable volume
volume_diff <- volume; response_diff <- response
while (any(apply(cbind(volume_diff, response_diff), 2, trend))) {
    volume_diff <- diff(volume_diff)
```

```r
    response_diff <- diff(response_diff)
}
series_prewhiten <- TSA::prewhiten(volume_diff, response_diff, plot=F)
lags <- series_prewhiten$ccf$lag[series_prewhiten$ccf$lag <= 0]
corrs <- abs(series_prewhiten$ccf$acf[series_prewhiten$ccf$lag <= 0])
cat(paste0('Correlación máxima con la variable volume en el retardo: ',
           lags[which.max(corrs)], '\n'))
```

```
Correlación máxima con la variable volume en el retardo: 0
```

En esta segunda iteración vemos que los retardos óptimos se producen en valores $k \neq 0$. Teniendo esto en cuenta debemos "alterar" las series correspondientes a las variables regresoras para construir el modelo ARIMAX.

```r
high_lagged <- lag(high, -1)
low_lagged <- lag(low, -1)
head(high_lagged)
```

```
Time Series:
Start = 2
End = 7
Frequency = 1
[1] 40.76 40.74 41.78 41.91 41.69 41.62
```

```r
head(low_lagged)
```

```
Time Series:
Start = 2
End = 7
Frequency = 1
[1] 40.31 40.12 40.18 41.31 41.04 41.25
```

Como podemos comprobar ahora las variables que han sido retardadas tienen su inicio en el tiempo $t = 3$ (uno más que sin retardar). Para construir un ARIMAX con la variable respuesta no se pueden introducir de esta forma en la función `auto.arima`, pues esta no tiene en cuenta el inicio de cada serie temporal. La forma correcta de introducirlas es construyendo una matriz para que coincidan los tiempos del par de series temporales:

```r
data_high <- cbind(close_cut, open=open_cut,
                   high=window(high_lagged, start=start(close_cut)))
head(data_high)
```

```
Time Series:
Start = 2
End = 7
Frequency = 1
  close_cut  open  high
2     40.29 40.66 40.76
3     41.55 40.34 40.74
4     41.53 41.61 41.78
5     41.42 41.48 41.91
6     41.48 41.25 41.69
```

```
7     41.72 41.63 41.62
```

```r
data_low <- cbind(close_cut, open=open_cut,
                  low=window(low_lagged, start=start(close_cut)))
head(data_low)
```

```
Time Series:
Start = 2
End = 7
Frequency = 1
  close_cut  open   low
2     40.29 40.66 40.31
3     41.55 40.34 40.12
4     41.53 41.61 40.18
5     41.42 41.48 41.31
6     41.48 41.25 41.04
7     41.72 41.63 41.25
```

```r
data_volume <- cbind(close_cut, open=open_cut, volume=volume_cut)
head(data_volume)
```

```
Time Series:
Start = 2
End = 7
Frequency = 1
  close_cut  open   volume
2     40.29 40.66 37487476
3     41.55 40.34 39223692
4     41.53 41.61 28809375
5     41.42 41.48 24753438
6     41.48 41.25 25723861
7     41.72 41.63 28022002
```

Una vez retardadas las variables procedemos a ampliar el modelo inicial:

```r
ajuste_high <- auto.fit.arima(data_high[, c(1)],     # variable respuesta
                              data_high[, -c(1)],    # variables regresoras
                              show_info=F)
ajuste_high
```

```
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    open    high
         0.0748  0.9461  0.0531
s.e.     0.0947  0.0239  0.0237

sigma^2 = 2.946:  log likelihood = -2956.77
AIC=5921.55   AICc=5921.57   BIC=5942.83
```

```r
ajuste_low <- auto.fit.arima(data_low[, c(1)], xregs=data_low[, -c(1)],
                             show_info=F)
ajuste_low
```

```
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    open     low
         0.0522  0.9609  0.0394
s.e.     0.0950  0.0258  0.0262

sigma^2 = 2.951:  log likelihood = -2958.15
AIC=5924.31   AICc=5924.34   BIC=5945.59
```

```
ajuste_volume <- auto.fit.arima(data_volume[, c(1)], xregs=data_volume[, -c(1)],
                               show_info=F)
ajuste_volume
```

```
[1] NA
```

Los resultados de la segunda iteración son los siguientes:

- El mejor modelo para la variable `High` y `Open` es un ARIMA(0,0,0) con AICc=5921.57 y retardo $k = -1$ (se ha mejorado el modelo anterior).
- El mejor modelo para la variable `Low` y `Open` es un ARIMA(0,0,0) con AICc=5924.34 y retardo $k = -1$ (se ha mejorado el modelo anterior).
- No se puede encontrar un modelo para la variable `Volume` y `Open` con retardo nulo.

A partir de esto se toma el modelo con las variables regresoras `High` y `Open` y se calculan sus residuos.

```
response <- residuals(ajuste_high, type='regression')
head(response)
```

```
Time Series:
Start = 2
End = 7
Frequency = 1
[1] -0.4169381  1.1468740 -0.1298797 -0.1237890  0.1654925  0.0496928
```

Estos residuos serán utilizados para la siguiente iteración para escoger nuevos retardos con los que añadir más variables regresoras.

Nota: Los residuos tienen valores nulos. Para aplicar el `adf.test` será necesario eliminarlos.

Fase 3: Búsqueda de variables a añadir al modelo tomando como variable respuesta los residuos del modelo anterior.

Volvemos a aplicar preblanqueado con los residuos del modelo anterior y las covariables que aún no han sido añadidas al modelo:

```
# Preblanqueado con la variable low
low_diff <- low; response_diff <- response
while (any(apply(cbind(low_diff, response_diff), 2, trend))) {
    low_diff <- diff(low_diff)
    response_diff <- diff(response_diff)
```

```
}
series_prewhiten <- TSA::prewhiten(low_diff, response_diff, plot=F)
lags <- series_prewhiten$ccf$lag[series_prewhiten$ccf$lag <= 0]
corrs <- abs(series_prewhiten$ccf$acf[series_prewhiten$ccf$lag <= 0])
cat(paste0('Correlación máxima con la variable low en el retardo: ',
           lags[which.max(corrs)], '\n'))
```

```
Correlación máxima con la variable low en el retardo: -1
```

```
# Preblanqueado con la variable volume
volume_diff <- volume; response_diff <- response
while (any(apply(cbind(volume_diff, response_diff), 2, trend))) {
    volume_diff <- diff(volume_diff)
    response_diff <- diff(response_diff)
}
series_prewhiten <- TSA::prewhiten(volume_diff, response_diff, plot=F)
lags <- series_prewhiten$ccf$lag[series_prewhiten$ccf$lag <= 0]
corrs <- abs(series_prewhiten$ccf$acf[series_prewhiten$ccf$lag <= 0])
cat(paste0('Correlación máxima con la variable volume en el retardo: ',
           lags[which.max(corrs)], '\n'))
```

```
Correlación máxima con la variable volume en el retardo: 0
```

Volvemos a construir las matrices con las series temporales que actuarán como regresoras en el nuevo modelo:

```
data_low <- cbind(close_cut, open=open_cut,
                  high=window(high_lagged, start=start(close_cut)),
                  low=window(low_lagged, start=start(close_cut)))
head(data_low)
```

```
Time Series:
Start = 2
End = 7
Frequency = 1
  close_cut  open  high   low
2     40.29 40.66 40.76 40.31
3     41.55 40.34 40.74 40.12
4     41.53 41.61 41.78 40.18
5     41.42 41.48 41.91 41.31
6     41.48 41.25 41.69 41.04
7     41.72 41.63 41.62 41.25
```

```
data_volume <- cbind(close_cut, open=open_cut,
                     high=window(high_lagged, start=start(close_cut)),
                     volume_cut)
head(data_volume)
```

```
Time Series:
Start = 2
End = 7
```

```
Frequency = 1
  close_cut  open  high volume_cut
2     40.29 40.66 40.76   37487476
3     41.55 40.34 40.74   39223692
4     41.53 41.61 41.78   28809375
5     41.42 41.48 41.91   24753438
6     41.48 41.25 41.69   25723861
7     41.72 41.63 41.62   28022002
```

Y ajustamos los dos modelos:

```
ajuste_low <- auto.fit.arima(data_low[, c(1)], xregs=data_low[, -c(1)], show_info=F)
ajuste_low
```

```
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    open    high     low
         0.0726  0.9439  0.0502  0.0052
s.e.     0.0957  0.0277  0.0301  0.0332

sigma^2 = 2.948:  log likelihood = -2956.76
AIC=5923.52   AICc=5923.56   BIC=5950.12
```

```
ajuste_volume <- auto.fit.arima(data_volume[, c(1)], xregs=data_volume[, -c(1)],
                                show_info=F)
ajuste_volume
```

```
[1] NA
```

- El mejor modelo para la variable `Low`, `High`, `Open` es un ARIMA(0,0,0) con AICc=5923.56 y retardo $k = -1$ (no se ha mejorado el modelo anterior).
- No se puede encontrar un modelo para la variable `Volume`, `High`, `Open` con retardo nulo.

Como al añadir el modelo `Low` no se ha mejorado el criterio de información escogido (el AICc) no se añade la variable `Low` al modelo y se detiene el algoritmo de adición de variables regresoras.

Como el modelo anterior (`ajuste_high`) tiene errores estacionarios se asume como modelo final. Si tuviera errores estacionarios (modelizables por un ARIMA con $d > 0$) entonces se tendría que intentar ajustar un modelo ARIMAX con errores estacionarios y, en caso de no ser posible

```
ajuste_high
```

```
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    open    high
         0.0748  0.9461  0.0531
s.e.     0.0947  0.0239  0.0237

sigma^2 = 2.946:  log likelihood = -2956.77
```

```
AIC=5921.55   AICc=5921.57   BIC=5942.83
```

Podemos comprobar que utilizando la función `auto.fit.arima.regression` se automatiza todo este proceso y se tienen los mismos resultados:

```r
microsoft <- read.csv('data/microsoft-stock.csv')
close <- ts(microsoft$Close)
regresoras <- as.data.frame(
    lapply(microsoft[, c('Open', 'High', 'Low', 'Volume')], ts))
auto.fit.arima.regression(serie=close, xregs=regresoras, ic='aicc', stationary_method='adf.test
```

```
Se ha probado con la variable Open [ic=5924.58405199719, lag=0]
No se ha podido ajustar un modelo para High
No se ha podido ajustar un modelo para Low
No se ha podido ajustar un modelo para Volume
Se ha añadido la variable regresora Open [aicc=5924.58405199719]
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    xreg
         0.0645  0.9997
s.e.     0.0947  0.0008

sigma^2 = 2.954:  log likelihood = -2959.28
AIC=5924.57   AICc=5924.58   BIC=5940.53
------------------------------------------------------------------------------------------
Se ha probado con la variable High [ic=5921.57447994466, lag=-1]
Se ha probado con la variable Low [ic=5924.33558667892, lag=-1]
No se ha podido ajustar un modelo para Volume
Se ha añadido la variable regresora High [aicc=5921.57447994466]
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    Open    High
         0.0748  0.9461  0.0531
s.e.     0.0947  0.0239  0.0237

sigma^2 = 2.946:  log likelihood = -2956.77
AIC=5921.55   AICc=5921.57   BIC=5942.83
------------------------------------------------------------------------------------------
Se ha probado con la variable Low [ic=5923.5637198765, lag=-1]
No se ha podido ajustar un modelo para Volume
No se añaden más variables

------------------------------------------------------------------------------------------
|                  Histórico de variables añadidas al modelo (ndiff=0)                   |
------------------------------------------------------------------------------------------
  var lag               ic
 Open   0 5924.58405199719
 High  -1 5921.57447994466
------------------------------------------------------------------------------------------
Series: serie
```

```
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    Open    High
         0.0748  0.9461  0.0531
s.e.     0.0947  0.0239  0.0237

sigma^2 = 2.946:  log likelihood = -2956.77
AIC=5921.55   AICc=5921.57   BIC=5942.83
```

```
Series: serie
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept    Open    High
         0.0748  0.9461  0.0531
s.e.     0.0947  0.0239  0.0237

sigma^2 = 2.946:  log likelihood = -2956.77
AIC=5921.55   AICc=5921.57   BIC=5942.83
```

Referencias

Cryer, Jonathan D y Kung-Sik Chan (2008). *Time series analysis: with applications in R*. Vol. 2. Springer.