Forecasting Google Trends with VAR and ARIMA Models

Pasquale Gravante

2024-06-24

INTROUDCTION

This project aims to forecast Google Trends data for search terms using Vector Autoregression (VAR) and Autoregressive Integrated Moving Average (ARIMA) models. The analysis will involve data collection, preprocessing, stationarity testing, model selection, diagnostics, and forecasting. By the end of this project, we will compare the performance of VAR and ARIMA models in terms of their forecasting accuracy.

For the first part of the project, the data of the last 5 years for the "AI" search term is going to be used and then also the data for the "Chat GPT" search term is going to be used.

FIRST TIME SERIES

DATA COLLECTION AND PRE-PROCESSING

```r
library(gtrendsR)
library(vars)
library(forecast)
library(ggplot2)
library(urca)
library(xts)
library(readr)
library(tidyverse)
library(lmtest)
library(tseries)
library(fUnitRoots)
```

```r
library(quantmod)
library(kableExtra)
library(lubridate)
library(formattable)
```

The necessary libraries are loaded, and the data is imported from a CSV file. The first few rows of the dataset are displayed to ensure it has been loaded correctly.

```r
# Load the CSV file
trend1_data <- read_csv("AItrend.csv")

# Check the data
head(trend1_data)
```

```
# A tibble: 6 x 2
  Week          AI
  <date>     <dbl>
1 2019-05-26    14
2 2019-06-02    14
3 2019-06-09    14
4 2019-06-16    14
5 2019-06-23    14
6 2019-06-30    14
```

PRE-PROCESSING

The 'Week' column is converted to Date format for time series analysis. The structure of the dataset is checked to confirm the conversion.

```r
# Convert the 'week' column to Date type
trend1_data$Week <- as.Date(trend1_data$Week, format = "%Y-%m-%d")

# Check the data types to ensure the date conversion was successful
str(trend1_data)
```

```
spc_tbl_ [262 x 2] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Week: Date[1:262], format: "2019-05-26" "2019-06-02" ...
 $ AI  : num [1:262] 14 14 14 14 14 14 13 13 13 13 ...
 - attr(*, "spec")=
```

```
  .. cols(
  ..   Week = col_date(format = ""),
  ..   AI = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

The dataset is then converted into an xts object, which is a common format for time series data since it facilitates time-based indexing and operations.

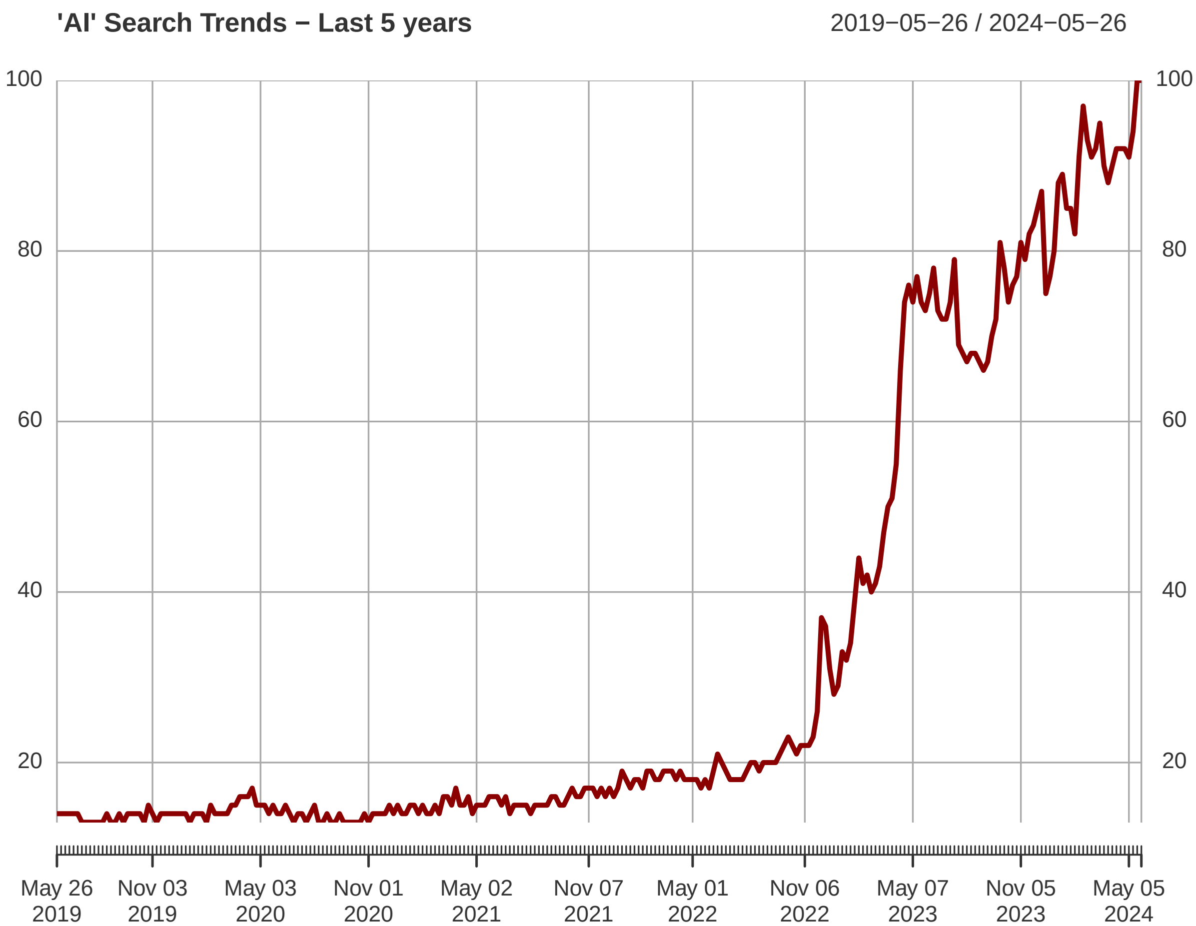
```
# Convert the data frame to xts object
trend1_xts <- xts(trend1_data$AI, order.by = trend1_data$Week)

# View the first few rows of xts object
head(trend1_xts)
```

```
           [,1]
2019-05-26   14
2019-06-02   14
2019-06-09   14
2019-06-16   14
2019-06-23   14
2019-06-30   14
```

Visualization

The time series data is visualized to observe trends, seasonal patterns, and any apparent anomalies in the 'AI' search trends over the last five years.

```
plot(trend1_xts,
     type = "l",
     col  = "darkred",
     lwd  = 3,
     main = "'AI' Search Trends - Last 5 years")
```



DETRENDING AND STATIONARITY

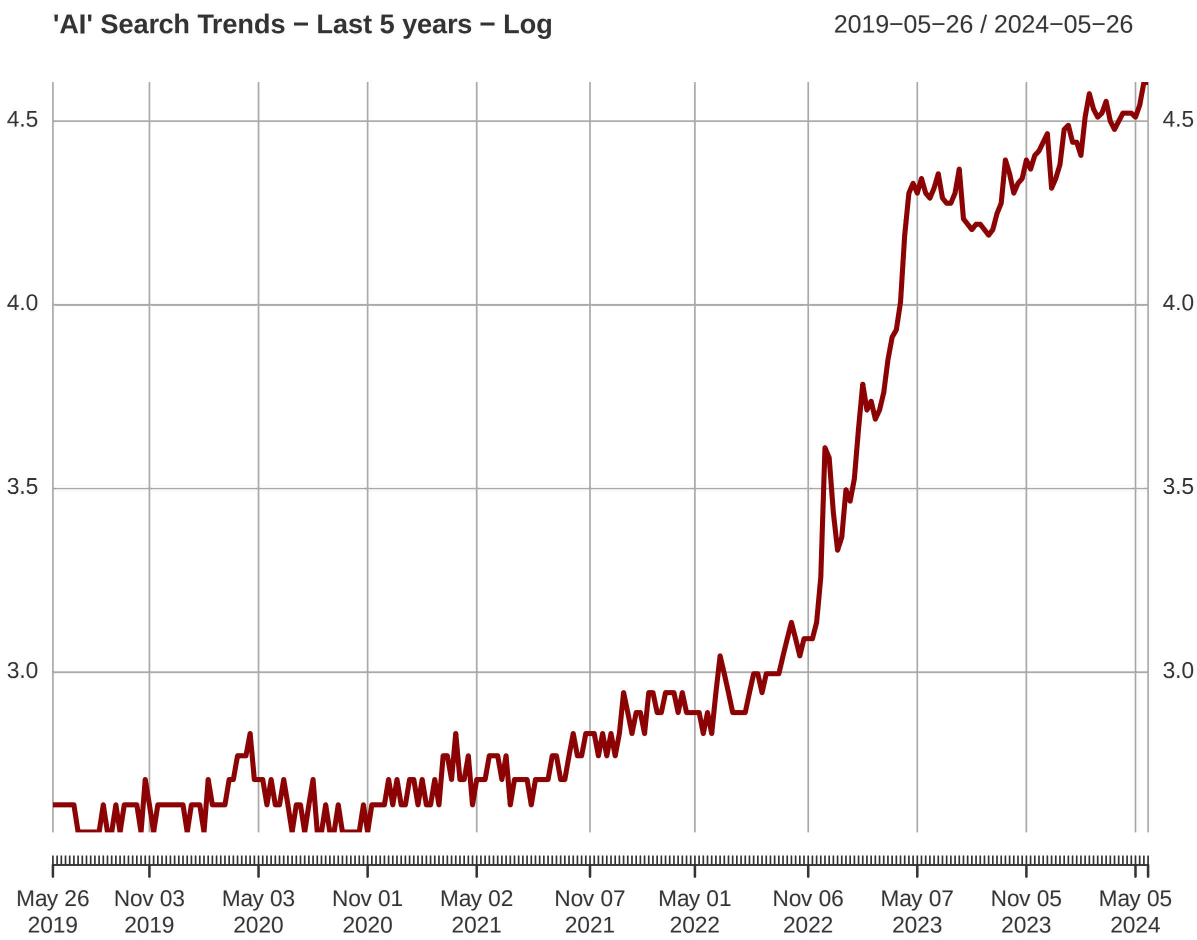
In this section the stationarity assumption is addressed, since it is crucial in time series modeling. Non-stationary data can lead to unreliable and spurious results.

Transformations

Log transformation is applied to stabilize the variance.

```
# Log transformation
log_trend1_xts <- log(trend1_xts)
plot(log_trend1_xts,
     type = "l",
```

```
     col  = "darkred",
     lwd  = 3,
     main = "'AI' Search Trends - Last 5 years - Log")
```

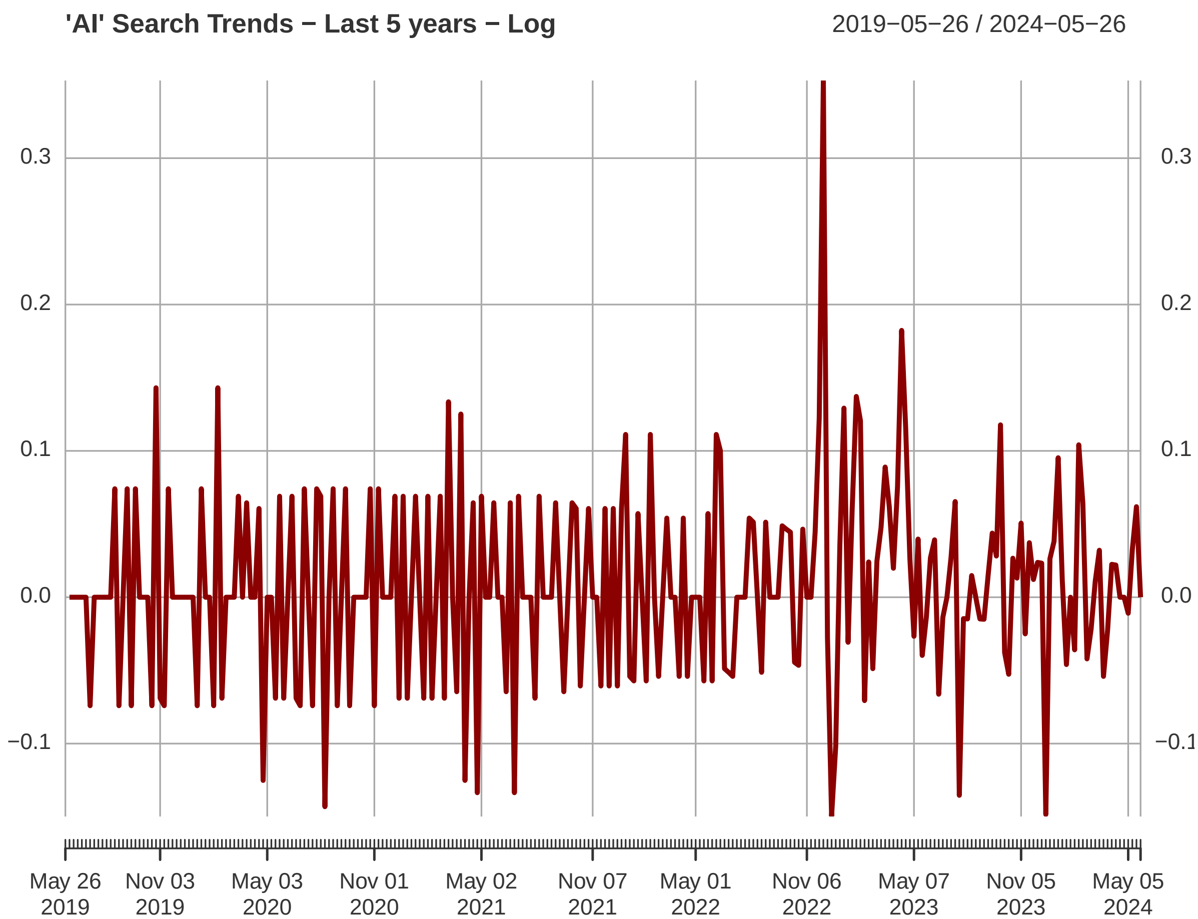


Differencing

Differencing is used to remove trends and achieve stationarity.

```
diff_log_trend1_xts <- diff(log_trend1_xts, differences = 1)
plot(diff_log_trend1_xts,
     type = "l",
     col  = "darkred",
     lwd  = 3,
```

```r
    main = "'AI' Search Trends - Last 5 years - Log")
```



The graph suggests that stationarity has been achieved but it has to be checked through the relative tests.

ADF Test for stationarity

The Augmented Dickey-Fuller (ADF) test checks for the presence of a unit root in the series, which indicates non-stationarity.

```r
# Apply the ADF test to the differenced series
diff_log_trend1_xts <- diff_log_trend1_xts[-1,]
adf_test <- ur.df(diff_log_trend1_xts, type = "drift", lags = 0)
```

```r
summary(adf_test)
```

```
############################################### 
# Augmented Dickey-Fuller Test Unit Root Test # 
############################################### 

Test regression drift 


Call:
lm(formula = z.diff ~ z.lag.1 + 1)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.16257 -0.03395 -0.00875  0.04063  0.36327 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.008746   0.003855   2.269   0.0241 *  
z.lag.1     -1.156534   0.061490 -18.809   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06171 on 258 degrees of freedom
Multiple R-squared:  0.5783,	Adjusted R-squared:  0.5766 
F-statistic: 353.8 on 1 and 258 DF,  p-value: < 2.2e-16


Value of test-statistic is: -18.8086 176.8808 

Critical values for test statistics: 
      1pct  5pct 10pct
tau2 -3.44 -2.87 -2.57
phi1  6.47  4.61  3.79
```

The test statistic value (-18.8086) is far below the critical values at all common significance levels (1%, 5%, and 10%), strongly indicating the rejection of the null hypothesis of a unit root. This result confirms that the time series is stationary.

PP Test for stationarity

The Phillips-Perron (PP) test also checks for a unit root, accounting for serial correlation in the error terms.

```r
pp.test <- ur.pp(diff_log_trend1_xts,   # tested series
                 type = c("Z-tau"),     # standardization of the test
                  ↪  statistic needed
                 model = c("constant")) # constant deterministic
                  ↪  component
# which means we assume that any trends in the data are stochastic
summary(pp.test)
```

```
##################################
# Phillips-Perron Unit Root Test #
##################################

Test regression with intercept


Call:
lm(formula = y ~ y.l1)

Residuals:
     Min       1Q   Median       3Q      Max
-0.16257 -0.03395 -0.00875  0.04063  0.36327

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.008746   0.003855   2.269   0.0241 *
y.l1        -0.156534   0.061490  -2.546   0.0115 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06171 on 258 degrees of freedom
Multiple R-squared:  0.0245,    Adjusted R-squared:  0.02072
F-statistic: 6.481 on 1 and 258 DF,  p-value: 0.01149


Value of test-statistic, type: Z-tau  is: -19.3793
```

```
         aux. Z statistics
Z-tau-mu            2.3374

Critical values for Z statistics:
                     1pct      5pct     10pct
critical values -3.457006 -2.872754 -2.572697
```

The Z-tau test statistic value (-19.3793) is far below the critical values at all common significance levels (1%, 5%, and 10%), strongly indicating the rejection of the null hypothesis of a unit root. This confirms the results of the ADF test and we can say that the series is stationary.

KPSS Test

The KPSS is like the other 2 tests but the hypothesis are inverted.

```r
kpss.test <- ur.kpss(diff_log_trend1_xts,
                     type = c("mu")) # constant deterministic
summary(kpss.test)
```

```
#######################
# KPSS Unit Root Test #
#######################

Test is of type: mu with 5 lags.

Value of test-statistic is: 0.5171

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

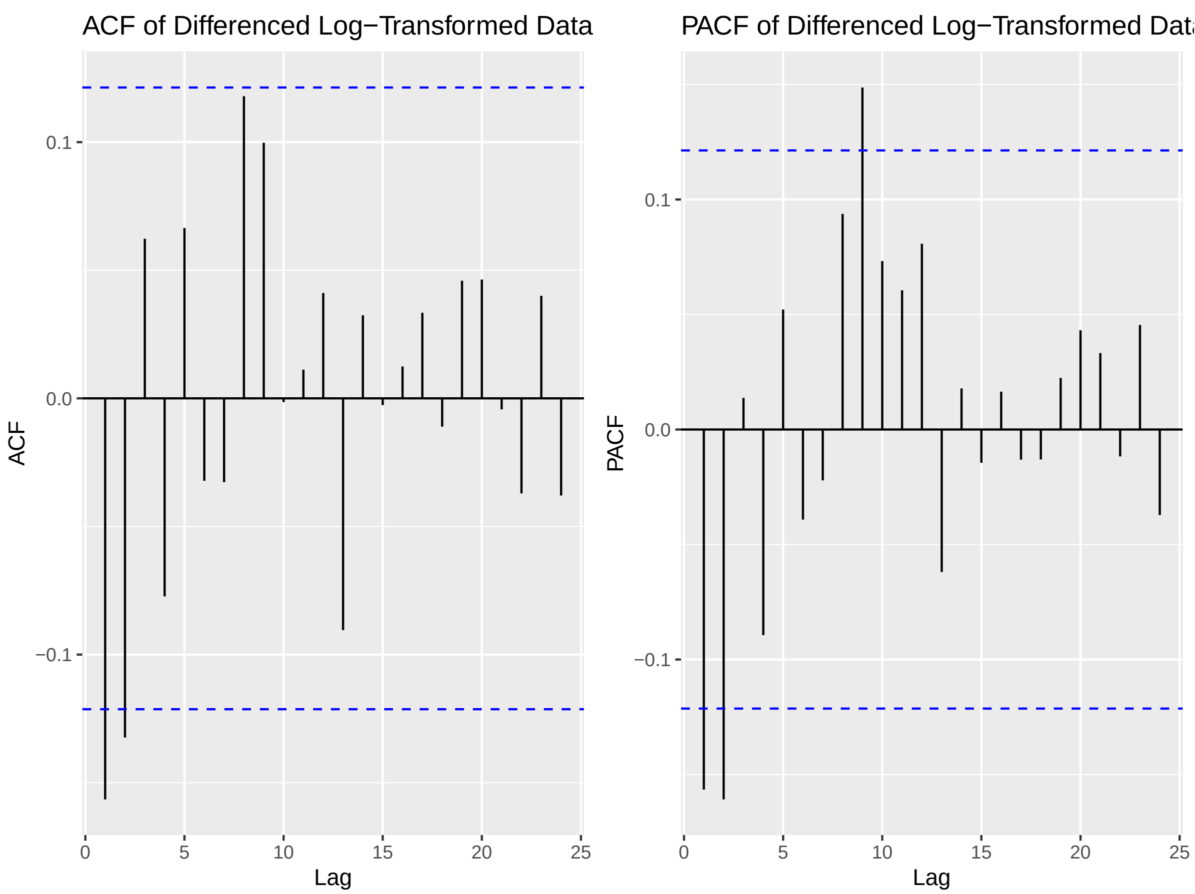
In this case the test-statistic of **0.51** is lower than than 2.5% critical value (0.574) so we **cannot reject the null** about **stationarity** of the first differences at **2.5% and 1% significance level**. (however we reject it at 5% level).

By looking at the graph and from the results of the tests we can conclude that ***AI ~ I (1).***

ACF and PACF plots

ACF shows the correlation between the time series and its lagged versions, while PACF gives the partial correlation of a stationary time series with its own lagged values. Their plots help in identifying the appropriate AR and MA terms for ARIMA modeling.

```
# ACF and PACF plots of the differenced series
acf_plot <- ggAcf(diff_log_trend1_xts, main = "ACF of Differenced
↪   Log-Transformed Data")
pacf_plot <- ggPacf(diff_log_trend1_xts, main = "PACF of Differenced
↪   Log-Transformed Data")

gridExtra::grid.arrange(acf_plot, pacf_plot, ncol = 2)
```



As shown by the plots, we can see some significant spikes at lag 1 and 2 for the ACF plot and some at lag 1, 2 and 9 in the PACF plot.

MODELING

A grid search is conducted to identify the optimal ARIMA parameters based on AIC and BIC values. The results help in selecting the best combination of p, d, and q.

Grid search for ARIMA parameters

```
# Define the parameter grid
p_values <- c(1, 2, 9)
d_values <- c(1)
q_values <- c(1, 2, 8)

# Initialize an empty list to store results
results <- list()

# Perform grid search WITHOUT CONSTANT
for (p in p_values) {
  for (d in d_values) {
    for (q in q_values) {
      # Fit the ARIMA model
      model <- tryCatch(
        {
          Arima(log_trend1_xts, order = c(p, d, q))
        },
        error = function(e) NULL
      )

      # Check if model fitting was successful
      if (!is.null(model)) {
        # Extract AIC and BIC
        aic <- AIC(model)
        bic <- BIC(model)

        # Store the results
        results <- rbind(results, data.frame(p = p, d = d, q = q, AIC =
↪ aic, BIC = bic))
      }
    }
  }
}
```

```r
# Convert results to a data frame
results_df <- as.data.frame(results)

# Print the results
print(results_df)
```

```
  p d q       AIC       BIC
1 1 1 1 -708.0181 -697.3245
2 1 1 2 -709.9838 -695.7257
3 1 1 8 -712.0973 -676.4521
4 2 1 1 -709.8640 -695.6060
5 2 1 2 -708.0825 -690.2599
6 2 1 8 -710.5610 -671.3512
7 9 1 1 -712.6472 -673.4375
8 9 1 2 -710.6835 -667.9092
9 9 1 8 -709.7928 -645.6314
```

Above we find the table with the respective AIC and BIC values for each model (without including a constant).

Now let's do the same thing by including a constant in the model and compare the results:

```r
# Define the parameter grid
p_values <- c(1, 2, 9)
d_values <- c(1)
q_values <- c(1, 2, 8)

# Initialize an empty list to store results
results <- list()

# Perform grid search WITHOUT CONSTANT
for (p in p_values) {
  for (d in d_values) {
    for (q in q_values) {
      # Fit the ARIMA model
      model <- tryCatch(
        {
          Arima(log_trend1_xts, order = c(p, d, q), include.constant =
           ↪  TRUE)
        },
```

```
        error = function(e) NULL
      )

      # Check if model fitting was successful
      if (!is.null(model)) {
        # Extract AIC and BIC
        aic <- AIC(model)
        bic <- BIC(model)

        # Store the results
        results <- rbind(results, data.frame(p = p, d = d, q = q, AIC =
↪   aic, BIC = bic))
      }
    }
  }
}

# Convert results to a data frame
results_df <- as.data.frame(results)

# Print the results
print(results_df)
```

```
  p d q       AIC       BIC
1 1 1 1 -713.2871 -699.0290
2 1 1 2 -715.0502 -697.2276
3 1 1 8 -714.1144 -674.9047
4 2 1 1 -713.9599 -696.1373
5 2 1 2 -713.0656 -691.6785
6 2 1 8 -712.8204 -670.0462
7 9 1 1 -713.1831 -670.4089
8 9 1 2 -711.1839 -664.8451
```

According to the table we can see that values are generally lower in this case, implying that drift should be included in the model.

Let's now compare the best models in terms of AIC / BIC, that are ARIMA (1,1,1) since it has the best BIC, ARIMA (1,1,2) since it has the best AIC and ARIMA (2,1,1) since it has good values for both.

Model diagnostics

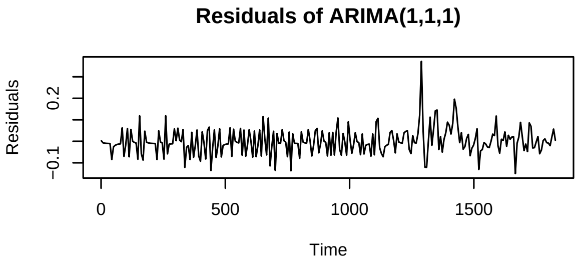
In this section the residuals are explored to make sure they don't present autocorrelation (i.e. they are white noise).

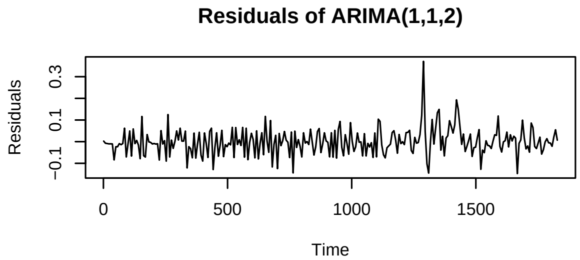
```
# Fit three different ARIMA models
model1 <- Arima(log_trend1_xts, order = c(1, 1, 1), include.constant =
↪  TRUE)
model2 <- Arima(log_trend1_xts, order = c(1, 1,2), include.constant =
↪  TRUE)
model3 <- Arima(log_trend1_xts, order = c(2, 1, 1), include.constant =
↪  TRUE)

# Extract residuals
residuals_model1 <- residuals(model1)
residuals_model2 <- residuals(model2)
residuals_model3 <- residuals(model3)
```

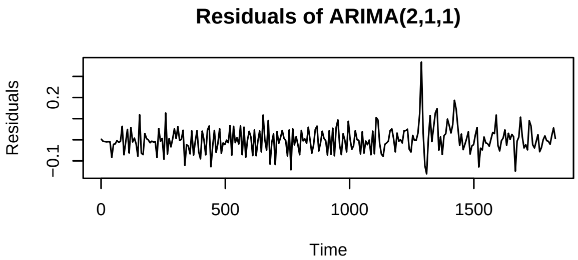
ARIMA models have been ran and their respective residuals extracted.

Let's see their plots:

```
# Plot Residuals
par(mfrow = c(3, 2))
plot(residuals_model1, main = "Residuals of ARIMA(1,1,1)", ylab =
↪  "Residuals")
plot(residuals_model2, main = "Residuals of ARIMA(1,1,2)", ylab =
↪  "Residuals")
plot(residuals_model3, main = "Residuals of ARIMA(2,1,1)", ylab =
↪  "Residuals")
```



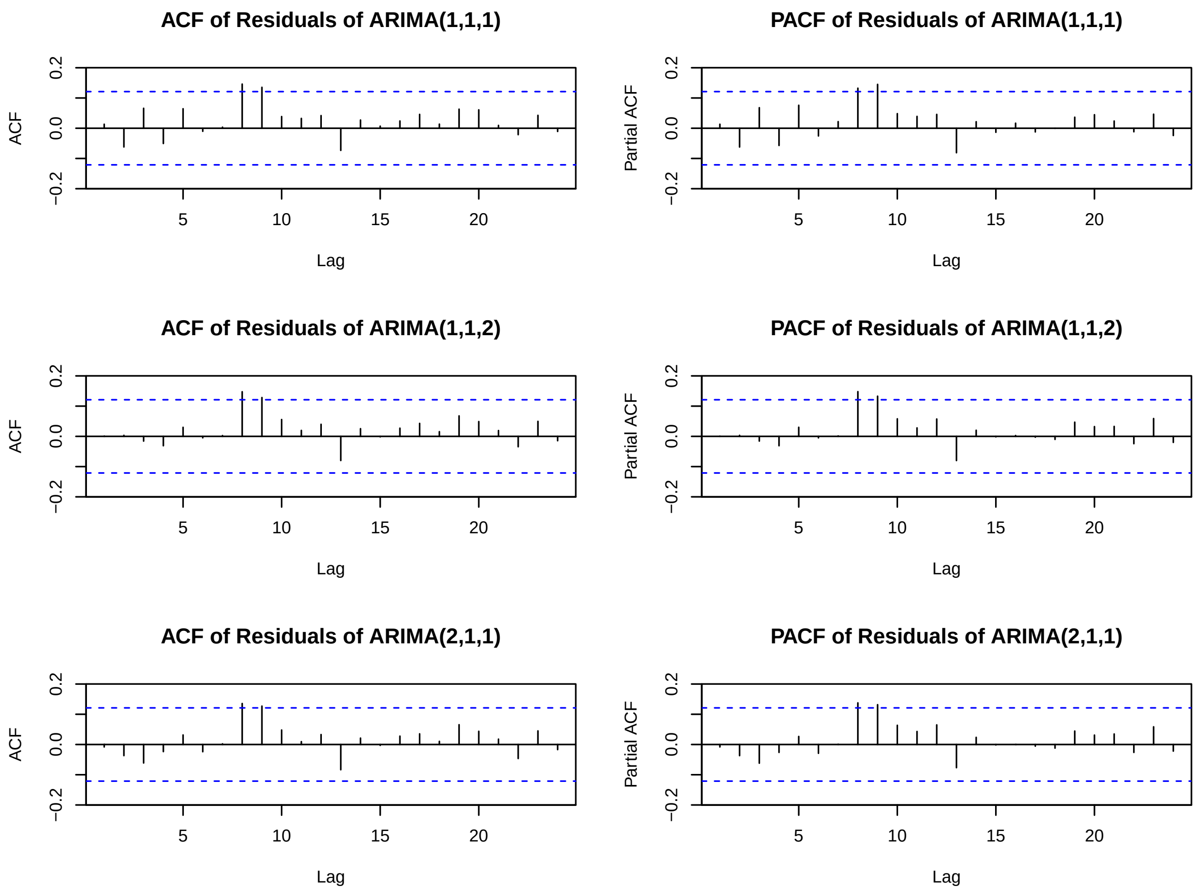




Residuals from different ARIMA models are plotted to visually inspect for white noise characteristics. Ideally, the residuals should appear random with no discernible patterns.

ACF and PACF plots of the residuals are also examined to check for any remaining autocorrelation:

```r
# ACF and PACF of Residuals
par(mfrow = c(3, 2))
Acf(residuals_model1, main = "ACF of Residuals of ARIMA(1,1,1)")
Pacf(residuals_model1, main = "PACF of Residuals of ARIMA(1,1,1)")
Acf(residuals_model2, main = "ACF of Residuals of ARIMA(1,1,2)")
Pacf(residuals_model2, main = "PACF of Residuals of ARIMA(1,1,2)")
Acf(residuals_model3, main = "ACF of Residuals of ARIMA(2,1,1)")
Pacf(residuals_model3, main = "PACF of Residuals of ARIMA(2,1,1)")
```



For each model, we can see some spikes at lag 8 and 9, so to choose the best one between the three, some tests have to performed:

- **Ljung-box test**: Checks the null hypothesis that the residuals are independently distributed (i.e. they are not autocorrelated).
- **Breusch-Godfrey test**: Checks the null hypothesis that there is no serial correlation in the residuals (i.e. they are white noise).

For each test, we would like the p-value to be greater than 0.05

We select number of **lags = 5** according to Ljung and Tsay (ln(T) = 5.5)

```
# Perform tests on residuals to check for white noise
lb_test_model1 <- Box.test(residuals_model1, lag = 5, type =
 ↪  "Ljung-Box")
```

```r
lb_test_model2 <- Box.test(residuals_model2, lag = 5, type =
  "Ljung-Box")
lb_test_model3 <- Box.test(residuals_model3, lag = 5, type =
  "Ljung-Box")
bg_test_model1 <- bgtest(residuals_model1 ~ fitted(model1), order = 5)
bg_test_model2 <- bgtest(residuals_model2 ~ fitted(model2), order = 5)
bg_test_model3 <- bgtest(residuals_model3 ~ fitted(model3), order = 5)
```

```r
# Create a table to compare results
comparison_table <- data.frame(
  Model = c("ARIMA(1,1,1)", "ARIMA(1,1,2)", "ARIMA(2,1,1)"),
  AIC = c(AIC(model1), AIC(model2), AIC(model3)),
  BIC = c(BIC(model1), BIC(model2), BIC(model3)),
  LjungBox_p_value = c(lb_test_model1$p.value, lb_test_model2$p.value,
    lb_test_model3$p.value),
  BG_statistic = c(bg_test_model1$p.value, bg_test_model2$p.value,
    bg_test_model3$p.value)
)

# Print the comparison table
print(comparison_table)
```

```
         Model       AIC       BIC LjungBox_p_value BG_statistic
1 ARIMA(1,1,1) -713.2871 -699.0290        0.5442502    0.4715572
2 ARIMA(1,1,2) -715.0502 -697.2276        0.9894031    0.9781981
3 ARIMA(2,1,1) -713.9599 -696.1373        0.8770163    0.7930028
```

What we can say from the comparison table is:

- For the Ljung-Box test, all three models (ARIMA(1,1,1), ARIMA(1,1,2), ARIMA(2,1,1)) have p-values well above 0.05 (0.5442502, 0.9894031, 0.8770163 respectively), suggesting that none of the models have significant autocorrelation in their residuals.
- For the BG test, p-values for the models are 0.4715572, 0.9781981, and 0.7930028 respectively. This suggests that for all of the three models the residuals are white noise.
- **ARIMA (1,1,2)** has the highest p-values for both tests and the best AIC value, so it is selected as the model that is going to be used for the forecasts.

Automatic model diagnostics

Now that the best model has been selected manually, some automatic model selection functions can be used and then compare the models' performances.

Auto AIC

Models with all combinations of p from 1 to 9 and q from 1 to 9 are analyzed and rated according to their AIC value:

```
arima.best.AIC <-
  auto.arima(log_trend1_xts,
             d = 1,              # parameter d of ARIMA model
             max.p = 9,          # Maximum value of p
             max.q = 9,          # Maximum value of q
             max.order = 18,     # maximum p+q
             start.p = 1,        # Starting value of p in stepwise
              ↪  procedure
             start.q = 1,        # Starting value of q in stepwise
              ↪  procedure
             ic = "aic",         # Information criterion to be used in
              ↪  model selection.
             stepwise = FALSE,   # if FALSE considers all models
             allowdrift = TRUE,  # include a constant
             trace = TRUE)       # show summary of all models considered
```

```
Fitting models using approximations to speed things up...

ARIMA(0,1,0)                    : -698.4329
ARIMA(0,1,0) with drift         : -700.243
ARIMA(0,1,1)                    : -703.1706
ARIMA(0,1,1) with drift         : -707.2513
ARIMA(0,1,2)                    : -703.8547
ARIMA(0,1,2) with drift         : -709.4507
ARIMA(0,1,3)                    : -703.6541
ARIMA(0,1,3) with drift         : -708.457
ARIMA(0,1,4)                    : -702.1104
ARIMA(0,1,4) with drift         : -707.312
ARIMA(0,1,5)                    : -701.705
ARIMA(0,1,5) with drift         : -706.5377
```

```
ARIMA(0,1,6)                    : -699.9709
ARIMA(0,1,6) with drift         : -704.6477
ARIMA(0,1,7)                    : -699.218
ARIMA(0,1,7) with drift         : -703.4153
ARIMA(0,1,8)                    : -707.3909
ARIMA(0,1,8) with drift         : -709.9724
ARIMA(0,1,9)                    : -707.22
ARIMA(0,1,9) with drift         : -709.1223
ARIMA(1,1,0)                    : -700.5758
ARIMA(1,1,0) with drift         : -703.7308
ARIMA(1,1,1)                    : -701.511
ARIMA(1,1,1) with drift         : -706.8663
ARIMA(1,1,2)                    : -703.5201
ARIMA(1,1,2) with drift         : -708.5851
ARIMA(1,1,3)                    : -701.5996
ARIMA(1,1,3) with drift         : -706.5911
ARIMA(1,1,4)                    : -699.6244
ARIMA(1,1,4) with drift         : -704.8335
ARIMA(1,1,5)                    : -708.2386
ARIMA(1,1,5) with drift         : -708.8586
ARIMA(1,1,6)                    : -706.3513
ARIMA(1,1,6) with drift         : -706.9195
ARIMA(1,1,7)                    : -704.7301
ARIMA(1,1,7) with drift         : -705.4502
ARIMA(1,1,8)                    : -705.955
ARIMA(1,1,8) with drift         : -707.8649
ARIMA(1,1,9)                    : -704.2214
ARIMA(1,1,9) with drift         : -706.116
ARIMA(2,1,0)                    : -702.6465
ARIMA(2,1,0) with drift         : -707.5998
ARIMA(2,1,1)                    : -702.3992
ARIMA(2,1,1) with drift         : -706.5471
ARIMA(2,1,2)                    : -700.6169
ARIMA(2,1,2) with drift         : -705.696
ARIMA(2,1,3)                    : -706.728
ARIMA(2,1,3) with drift         : -703.738
ARIMA(2,1,4)                    : -705.2976
ARIMA(2,1,4) with drift         : -703.1931
ARIMA(2,1,5)                    : -705.4142
ARIMA(2,1,5) with drift         : -706.0213
ARIMA(2,1,6)                    : Inf
ARIMA(2,1,6) with drift         : Inf
ARIMA(2,1,7)                    : Inf
```

```
ARIMA(2,1,7) with drift        : -702.5663
ARIMA(2,1,8)                   : -703.4342
ARIMA(2,1,8) with drift        : -705.6162
ARIMA(2,1,9)                   : Inf
ARIMA(2,1,9) with drift        : -704.1108
ARIMA(3,1,0)                   : -700.0285
ARIMA(3,1,0) with drift        : -704.6663
ARIMA(3,1,1)                   : -699.5123
ARIMA(3,1,1) with drift        : -703.8845
ARIMA(3,1,2)                   : -697.6481
ARIMA(3,1,2) with drift        : -702.8165
ARIMA(3,1,3)                   : -705.4334
ARIMA(3,1,3) with drift        : Inf
ARIMA(3,1,4)                   : Inf
ARIMA(3,1,4) with drift        : -702.0234
ARIMA(3,1,5)                   : Inf
ARIMA(3,1,5) with drift        : Inf
ARIMA(3,1,6)                   : Inf
ARIMA(3,1,6) with drift        : Inf
ARIMA(3,1,7)                   : Inf
ARIMA(3,1,7) with drift        : Inf
ARIMA(3,1,8)                   : -700.5921
ARIMA(3,1,8) with drift        : -702.8043
ARIMA(3,1,9)                   : -698.7332
ARIMA(3,1,9) with drift        : -701.695
ARIMA(4,1,0)                   : -698.0541
ARIMA(4,1,0) with drift        : -703.78
ARIMA(4,1,1)                   : -697.0899
ARIMA(4,1,1) with drift        : -702.6024
ARIMA(4,1,2)                   : Inf
ARIMA(4,1,2) with drift        : Inf
ARIMA(4,1,3)                   : Inf
ARIMA(4,1,3) with drift        : Inf
ARIMA(4,1,4)                   : -707.0912
ARIMA(4,1,4) with drift        : -707.793
ARIMA(4,1,5)                   : Inf
ARIMA(4,1,5) with drift        : Inf
ARIMA(4,1,6)                   : Inf
ARIMA(4,1,6) with drift        : Inf
ARIMA(4,1,7)                   : Inf
ARIMA(4,1,7) with drift        : Inf
ARIMA(4,1,8)                   : Inf
ARIMA(4,1,8) with drift        : Inf
```

```
ARIMA(4,1,9)                    : Inf
ARIMA(4,1,9) with drift         : Inf
ARIMA(5,1,0)                    : -696.6702
ARIMA(5,1,0) with drift         : -701.506
ARIMA(5,1,1)                    : -694.7047
ARIMA(5,1,1) with drift         : -699.5596
ARIMA(5,1,2)                    : -700.9894
ARIMA(5,1,2) with drift         : Inf
ARIMA(5,1,3)                    : Inf
ARIMA(5,1,3) with drift         : Inf
ARIMA(5,1,4)                    : Inf
ARIMA(5,1,4) with drift         : Inf
ARIMA(5,1,5)                    : Inf
ARIMA(5,1,5) with drift         : Inf
ARIMA(5,1,6)                    : Inf
ARIMA(5,1,6) with drift         : Inf
ARIMA(5,1,7)                    : Inf
ARIMA(5,1,7) with drift         : Inf
ARIMA(5,1,8)                    : -695.2927
ARIMA(5,1,8) with drift         : -698.1751
ARIMA(5,1,9)                    : -693.2953
ARIMA(5,1,9) with drift         : -696.3847
ARIMA(6,1,0)                    : -695.1703
ARIMA(6,1,0) with drift         : -700.8807
ARIMA(6,1,1)                    : -693.1896
ARIMA(6,1,1) with drift         : -699.0504
ARIMA(6,1,2)                    : -696.2897
ARIMA(6,1,2) with drift         : -697.785
ARIMA(6,1,3)                    : Inf
ARIMA(6,1,3) with drift         : Inf
ARIMA(6,1,4)                    : Inf
ARIMA(6,1,4) with drift         : -698.1418
ARIMA(6,1,5)                    : Inf
ARIMA(6,1,5) with drift         : Inf
ARIMA(6,1,6)                    : Inf
ARIMA(6,1,6) with drift         : Inf
ARIMA(6,1,7)                    : Inf
ARIMA(6,1,7) with drift         : Inf
ARIMA(6,1,8)                    : -694.501
ARIMA(6,1,8) with drift         : -697.8074
ARIMA(6,1,9)                    : Inf
ARIMA(6,1,9) with drift         : Inf
ARIMA(7,1,0)                    : -692.1834
```

```
ARIMA(7,1,0) with drift         : -698.1525
ARIMA(7,1,1)                    : -700.4668
ARIMA(7,1,1) with drift         : -696.2813
ARIMA(7,1,2)                    : -701.3723
ARIMA(7,1,2) with drift         : -698.7333
ARIMA(7,1,3)                    : -699.6977
ARIMA(7,1,3) with drift         : Inf
ARIMA(7,1,4)                    : Inf
ARIMA(7,1,4) with drift         : Inf
ARIMA(7,1,5)                    : Inf
ARIMA(7,1,5) with drift         : Inf
ARIMA(7,1,6)                    : Inf
ARIMA(7,1,6) with drift         : Inf
ARIMA(7,1,7)                    : Inf
ARIMA(7,1,7) with drift         : Inf
ARIMA(7,1,8)                    : Inf
ARIMA(7,1,8) with drift         : Inf
ARIMA(7,1,9)                    : Inf
ARIMA(7,1,9) with drift         : Inf
ARIMA(8,1,0)                    : -692.9176
ARIMA(8,1,0) with drift         : -697.6092
ARIMA(8,1,1)                    : -701.8554
ARIMA(8,1,1) with drift         : -702.9817
ARIMA(8,1,2)                    : -700.8535
ARIMA(8,1,2) with drift         : -702.6568
ARIMA(8,1,3)                    : -698.8542
ARIMA(8,1,3) with drift         : -700.6602
ARIMA(8,1,4)                    : -697.0867
ARIMA(8,1,4) with drift         : -699.2619
ARIMA(8,1,5)                    : -695.2056
ARIMA(8,1,5) with drift         : Inf
ARIMA(8,1,6)                    : -694.4045
ARIMA(8,1,6) with drift         : -695.6406
ARIMA(8,1,7)                    : Inf
ARIMA(8,1,7) with drift         : Inf
ARIMA(8,1,8)                    : Inf
ARIMA(8,1,8) with drift         : Inf
ARIMA(8,1,9)                    : Inf
ARIMA(8,1,9) with drift         : Inf
ARIMA(9,1,0)                    : -697.4957
ARIMA(9,1,0) with drift         : -700.5683
ARIMA(9,1,1)                    : -700.1773
ARIMA(9,1,1) with drift         : -701.5039
```

```
ARIMA(9,1,2)                    : -698.199
ARIMA(9,1,2) with drift         : Inf
ARIMA(9,1,3)                    : -696.1991
ARIMA(9,1,3) with drift         : -697.6981
ARIMA(9,1,4)                    : -694.9972
ARIMA(9,1,4) with drift         : -696.5029
ARIMA(9,1,5)                    : -694.1092
ARIMA(9,1,5) with drift         : Inf
ARIMA(9,1,6)                    : Inf
ARIMA(9,1,6) with drift         : Inf
ARIMA(9,1,7)                    : Inf
ARIMA(9,1,7) with drift         : Inf
ARIMA(9,1,8)                    : Inf
ARIMA(9,1,8) with drift         : Inf
ARIMA(9,1,9)                    : Inf
ARIMA(9,1,9) with drift         : Inf

Now re-fitting the best model(s) without approximations...




Best model: ARIMA(0,1,8) with drift
```

According to the automatic function, best model is ARIMA(0,1,8) with drift.

Let's test the significance of its coefficients:

```
coeftest(arima.best.AIC)
```

```
z test of coefficients:

       Estimate Std. Error z value  Pr(>|z|)    
ma1  -0.2106522  0.0611543 -3.4446 0.0005719 ***
ma2  -0.1520129  0.0623873 -2.4366 0.0148261 *  
ma3   0.0696407  0.0648367  1.0741 0.2827809    
ma4  -0.0788677  0.0634919 -1.2422 0.2141736    
ma5   0.1021360  0.0641732  1.5916 0.1114821    
ma6  -0.0258656  0.0641125 -0.4034 0.6866237    
ma7   0.0051393  0.0613435  0.0838 0.9332318    
ma8   0.1854018  0.0642254  2.8867 0.0038926 ** 
```

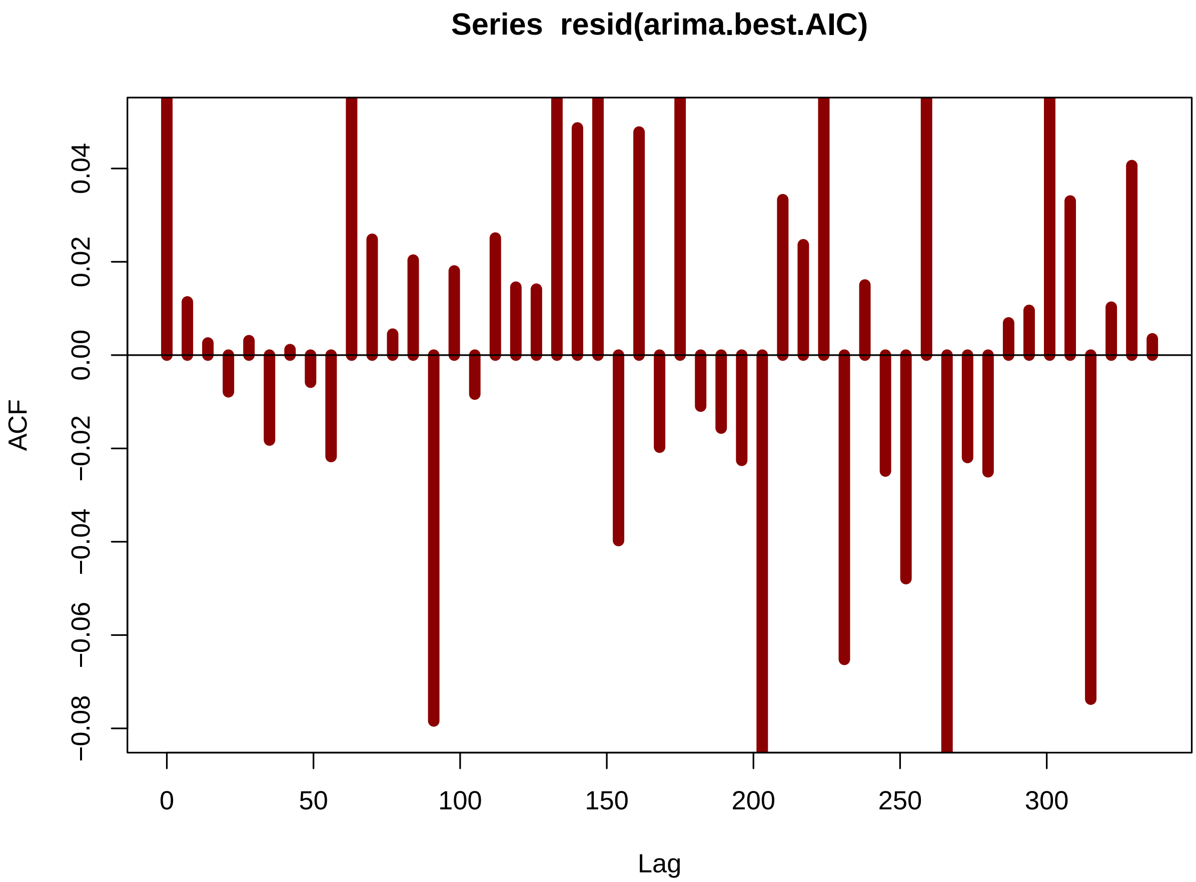
```
drift  0.0074155  0.0032582  2.2759 0.0228510 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- ma1, ma2, and ma8 are significant while ma3 to ma7 are not and might indicate some over-parametrization. Moreover, the drift term is also significant, suggesting it should be included in the model.

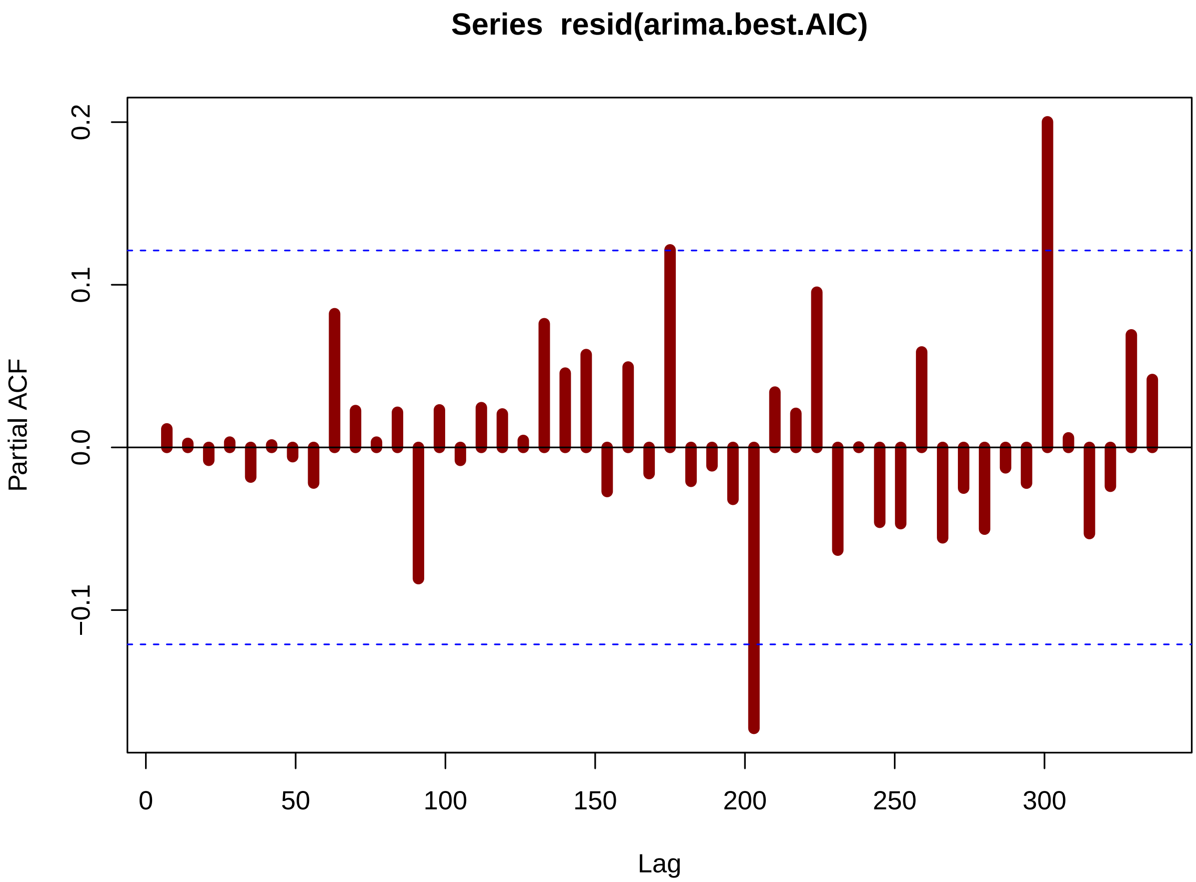
Model diagnostics

Let's again look at the ACF and PACF plots:

```r
acf(resid(arima.best.AIC),
    lag.max = 48,
    lwd = 7,
    col = "darkred",
    na.action = na.pass,
    ylim = c(-0.08, 0.05))
```




```r
pacf(resid(arima.best.AIC),
     lag.max = 48,
     lwd = 7,
     col = "darkred",
     na.action = na.pass)
```



Residuals look overall good although showing 2 significant spikes, so let's check the statistics.

Performing the Ljung-Box test and the BG test for the residuals:

```
Box.test(resid(arima.best.AIC), type = "Ljung-Box", lag =  5)
```

```

	Box-Ljung test

data:  resid(arima.best.AIC)
X-squared = 0.14463, df = 5, p-value = 0.9996
```

```
bgtest(arima.best.AIC$residuals ~ fitted(arima.best.AIC), order = 5)
```

```
    Breusch-Godfrey test for serial correlation of order up to 5

data:  arima.best.AIC$residuals ~ fitted(arima.best.AIC)
LM test = 0.20204, df = 5, p-value = 0.9991
```

For both tests the p.value is 0.999 meaning that residuals are not autocorrelated and they are white noise.

Let's now compare AIC and BIC values for both the automatic and the manual model:

```r
AIC(arima.best.AIC, model2)
```

```
               df       AIC
arima.best.AIC 10 -715.1920
model2          5 -715.0502
```

```r
BIC(arima.best.AIC, model2)
```

```
               df       BIC
arima.best.AIC 10 -679.5468
model2          5 -697.2276
```

As the table shows, the automatic model is just slighlty better in terms of AIC but the manual model is way better in BIC.

Auto BIC

Let's now do the same procedure by selecting the best model in terms of BIC:

```r
arima.best.BIC <-
  auto.arima(log_trend1_xts,
             d = 1,              # parameter d of ARIMA model
             max.p = 9,          # Maximum value of p
             max.q = 9,          # Maximum value of q
             max.order = 18,     # maximum p+q
             start.p = 1,        # Starting value of p in stepwise
              ↪  procedure
             start.q = 1,        # Starting value of q in stepwise
              ↪  procedure
```

```r
               ic = "bic",        # Information criterion to be used in
                ↪  model selection.
               stepwise = FALSE,  # if FALSE considers all models
               allowdrift = TRUE, # include a constant
               trace = TRUE)      # show summary of all models considered
```

```
Fitting models using approximations to speed things up...

ARIMA(0,1,0)                    : -694.8684
ARIMA(0,1,0) with drift         : -693.114
ARIMA(0,1,1)                    : -696.0416
ARIMA(0,1,1) with drift         : -696.5577
ARIMA(0,1,2)                    : -693.1612
ARIMA(0,1,2) with drift         : -695.1927
ARIMA(0,1,3)                    : -689.396
ARIMA(0,1,3) with drift         : -690.6344
ARIMA(0,1,4)                    : -684.2878
ARIMA(0,1,4) with drift         : -685.9249
ARIMA(0,1,5)                    : -680.3179
ARIMA(0,1,5) with drift         : -681.586
ARIMA(0,1,6)                    : -675.0193
ARIMA(0,1,6) with drift         : -676.1316
ARIMA(0,1,7)                    : -670.7019
ARIMA(0,1,7) with drift         : -671.3346
ARIMA(0,1,8)                    : -675.3102
ARIMA(0,1,8) with drift         : -674.3272
ARIMA(0,1,9)                    : -671.5748
ARIMA(0,1,9) with drift         : -669.9126
ARIMA(1,1,0)                    : -693.4467
ARIMA(1,1,0) with drift         : -693.0372
ARIMA(1,1,1)                    : -690.8175
ARIMA(1,1,1) with drift         : -692.6082
ARIMA(1,1,2)                    : -689.262
ARIMA(1,1,2) with drift         : -690.7625
ARIMA(1,1,3)                    : -683.777
ARIMA(1,1,3) with drift         : -685.204
ARIMA(1,1,4)                    : -678.2373
ARIMA(1,1,4) with drift         : -679.8819
ARIMA(1,1,5)                    : -683.2869
ARIMA(1,1,5) with drift         : -680.3425
```

```
ARIMA(1,1,6)                    : -677.8351
ARIMA(1,1,6) with drift         : -674.8388
ARIMA(1,1,7)                    : -672.6494
ARIMA(1,1,7) with drift         : -669.805
ARIMA(1,1,8)                    : -670.3098
ARIMA(1,1,8) with drift         : -668.6551
ARIMA(1,1,9)                    : -665.0116
ARIMA(1,1,9) with drift         : -663.3417
ARIMA(2,1,0)                    : -691.9529
ARIMA(2,1,0) with drift         : -693.3418
ARIMA(2,1,1)                    : -688.1411
ARIMA(2,1,1) with drift         : -688.7245
ARIMA(2,1,2)                    : -682.7943
ARIMA(2,1,2) with drift         : -684.3088
ARIMA(2,1,3)                    : -685.3409
ARIMA(2,1,3) with drift         : -678.7864
ARIMA(2,1,4)                    : -680.3459
ARIMA(2,1,4) with drift         : -674.6769
ARIMA(2,1,5)                    : -676.8981
ARIMA(2,1,5) with drift         : -673.9406
ARIMA(2,1,6)                    : Inf
ARIMA(2,1,6) with drift         : Inf
ARIMA(2,1,7)                    : Inf
ARIMA(2,1,7) with drift         : -663.3566
ARIMA(2,1,8)                    : -664.2245
ARIMA(2,1,8) with drift         : -662.842
ARIMA(2,1,9)                    : Inf
ARIMA(2,1,9) with drift         : -657.772
ARIMA(3,1,0)                    : -685.7705
ARIMA(3,1,0) with drift         : -686.8437
ARIMA(3,1,1)                    : -681.6897
ARIMA(3,1,1) with drift         : -682.4974
ARIMA(3,1,2)                    : -676.261
ARIMA(3,1,2) with drift         : -677.8648
ARIMA(3,1,3)                    : -680.4817
ARIMA(3,1,3) with drift         : Inf
ARIMA(3,1,4)                    : Inf
ARIMA(3,1,4) with drift         : -669.9427
ARIMA(3,1,5)                    : Inf
ARIMA(3,1,5) with drift         : Inf
ARIMA(3,1,6)                    : Inf
ARIMA(3,1,6) with drift         : Inf
ARIMA(3,1,7)                    : Inf
```

```
ARIMA(3,1,7) with drift         : Inf
ARIMA(3,1,8)                    : -657.8178
ARIMA(3,1,8) with drift         : -656.4656
ARIMA(3,1,9)                    : -652.3944
ARIMA(3,1,9) with drift         : -651.7917
ARIMA(4,1,0)                    : -680.2315
ARIMA(4,1,0) with drift         : -682.3929
ARIMA(4,1,1)                    : -675.7027
ARIMA(4,1,1) with drift         : -677.6508
ARIMA(4,1,2)                    : Inf
ARIMA(4,1,2) with drift         : Inf
ARIMA(4,1,3)                    : Inf
ARIMA(4,1,3) with drift         : Inf
ARIMA(4,1,4)                    : -675.0105
ARIMA(4,1,4) with drift         : -672.1478
ARIMA(4,1,5)                    : Inf
ARIMA(4,1,5) with drift         : Inf
ARIMA(4,1,6)                    : Inf
ARIMA(4,1,6) with drift         : Inf
ARIMA(4,1,7)                    : Inf
ARIMA(4,1,7) with drift         : Inf
ARIMA(4,1,8)                    : Inf
ARIMA(4,1,8) with drift         : Inf
ARIMA(4,1,9)                    : Inf
ARIMA(4,1,9) with drift         : Inf
ARIMA(5,1,0)                    : -675.2831
ARIMA(5,1,0) with drift         : -676.5543
ARIMA(5,1,1)                    : -669.7531
ARIMA(5,1,1) with drift         : -671.0435
ARIMA(5,1,2)                    : -672.4733
ARIMA(5,1,2) with drift         : Inf
ARIMA(5,1,3)                    : Inf
ARIMA(5,1,3) with drift         : Inf
ARIMA(5,1,4)                    : Inf
ARIMA(5,1,4) with drift         : Inf
ARIMA(5,1,5)                    : Inf
ARIMA(5,1,5) with drift         : Inf
ARIMA(5,1,6)                    : Inf
ARIMA(5,1,6) with drift         : Inf
ARIMA(5,1,7)                    : Inf
ARIMA(5,1,7) with drift         : Inf
ARIMA(5,1,8)                    : -645.3894
ARIMA(5,1,8) with drift         : -644.7073
```

```
ARIMA(5,1,9)                    : -639.8275
ARIMA(5,1,9) with drift         : -639.3524
ARIMA(6,1,0)                    : -670.2186
ARIMA(6,1,0) with drift         : -672.3645
ARIMA(6,1,1)                    : -664.6734
ARIMA(6,1,1) with drift         : -666.9697
ARIMA(6,1,2)                    : -664.209
ARIMA(6,1,2) with drift         : -662.1398
ARIMA(6,1,3)                    : Inf
ARIMA(6,1,3) with drift         : Inf
ARIMA(6,1,4)                    : Inf
ARIMA(6,1,4) with drift         : -655.3676
ARIMA(6,1,5)                    : Inf
ARIMA(6,1,5) with drift         : Inf
ARIMA(6,1,6)                    : Inf
ARIMA(6,1,6) with drift         : Inf
ARIMA(6,1,7)                    : Inf
ARIMA(6,1,7) with drift         : Inf
ARIMA(6,1,8)                    : -641.0332
ARIMA(6,1,8) with drift         : -640.7751
ARIMA(6,1,9)                    : Inf
ARIMA(6,1,9) with drift         : Inf
ARIMA(7,1,0)                    : -663.6672
ARIMA(7,1,0) with drift         : -666.0718
ARIMA(7,1,1)                    : -668.3861
ARIMA(7,1,1) with drift         : -660.6361
ARIMA(7,1,2)                    : -665.7271
ARIMA(7,1,2) with drift         : -659.5236
ARIMA(7,1,3)                    : -660.488
ARIMA(7,1,3) with drift         : Inf
ARIMA(7,1,4)                    : Inf
ARIMA(7,1,4) with drift         : Inf
ARIMA(7,1,5)                    : Inf
ARIMA(7,1,5) with drift         : Inf
ARIMA(7,1,6)                    : Inf
ARIMA(7,1,6) with drift         : Inf
ARIMA(7,1,7)                    : Inf
ARIMA(7,1,7) with drift         : Inf
ARIMA(7,1,8)                    : Inf
ARIMA(7,1,8) with drift         : Inf
ARIMA(7,1,9)                    : Inf
ARIMA(7,1,9) with drift         : Inf
ARIMA(8,1,0)                    : -660.8369
```

```
ARIMA(8,1,0) with drift         : -661.964
ARIMA(8,1,1)                    : -666.2102
ARIMA(8,1,1) with drift         : -663.772
ARIMA(8,1,2)                    : -661.6438
ARIMA(8,1,2) with drift         : -659.8825
ARIMA(8,1,3)                    : -656.0799
ARIMA(8,1,3) with drift         : -654.3215
ARIMA(8,1,4)                    : -650.7479
ARIMA(8,1,4) with drift         : -649.3587
ARIMA(8,1,5)                    : -645.3023
ARIMA(8,1,5) with drift         : Inf
ARIMA(8,1,6)                    : -640.9367
ARIMA(8,1,6) with drift         : -638.6083
ARIMA(8,1,7)                    : Inf
ARIMA(8,1,7) with drift         : Inf
ARIMA(8,1,8)                    : Inf
ARIMA(8,1,8) with drift         : Inf
ARIMA(8,1,9)                    : Inf
ARIMA(8,1,9) with drift         : Inf
ARIMA(9,1,0)                    : -661.8505
ARIMA(9,1,0) with drift         : -661.3586
ARIMA(9,1,1)                    : -660.9676
ARIMA(9,1,1) with drift         : -658.7296
ARIMA(9,1,2)                    : -655.4248
ARIMA(9,1,2) with drift         : Inf
ARIMA(9,1,3)                    : -649.8604
ARIMA(9,1,3) with drift         : -647.7948
ARIMA(9,1,4)                    : -645.0939
ARIMA(9,1,4) with drift         : -643.0351
ARIMA(9,1,5)                    : -640.6414
ARIMA(9,1,5) with drift         : Inf
ARIMA(9,1,6)                    : Inf
ARIMA(9,1,6) with drift         : Inf
ARIMA(9,1,7)                    : Inf
ARIMA(9,1,7) with drift         : Inf
ARIMA(9,1,8)                    : Inf
ARIMA(9,1,8) with drift         : Inf
ARIMA(9,1,9)                    : Inf
ARIMA(9,1,9) with drift         : Inf

Now re-fitting the best model(s) without approximations...
```

```
 Best model: ARIMA(0,1,1) with drift
```

In this case ARIMA(0,1,1) with drift is selected.

Diagnostics

Significancy of the coefficients is tested:

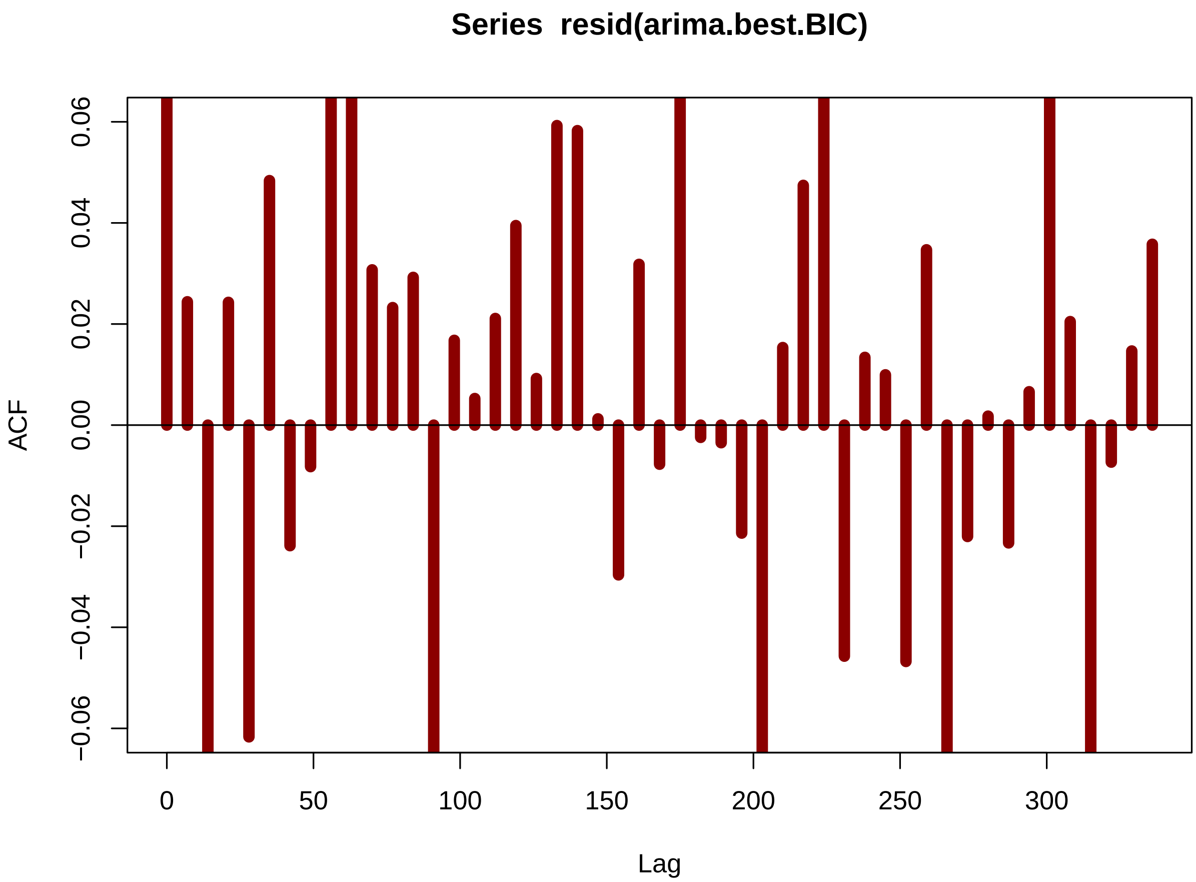
```r
coeftest(arima.best.BIC)
```

```
z test of coefficients:

        Estimate Std. Error z value Pr(>|z|)
ma1   -0.2134061  0.0685149 -3.1147 0.001841 **
drift  0.0075367  0.0029766  2.5320 0.011341 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

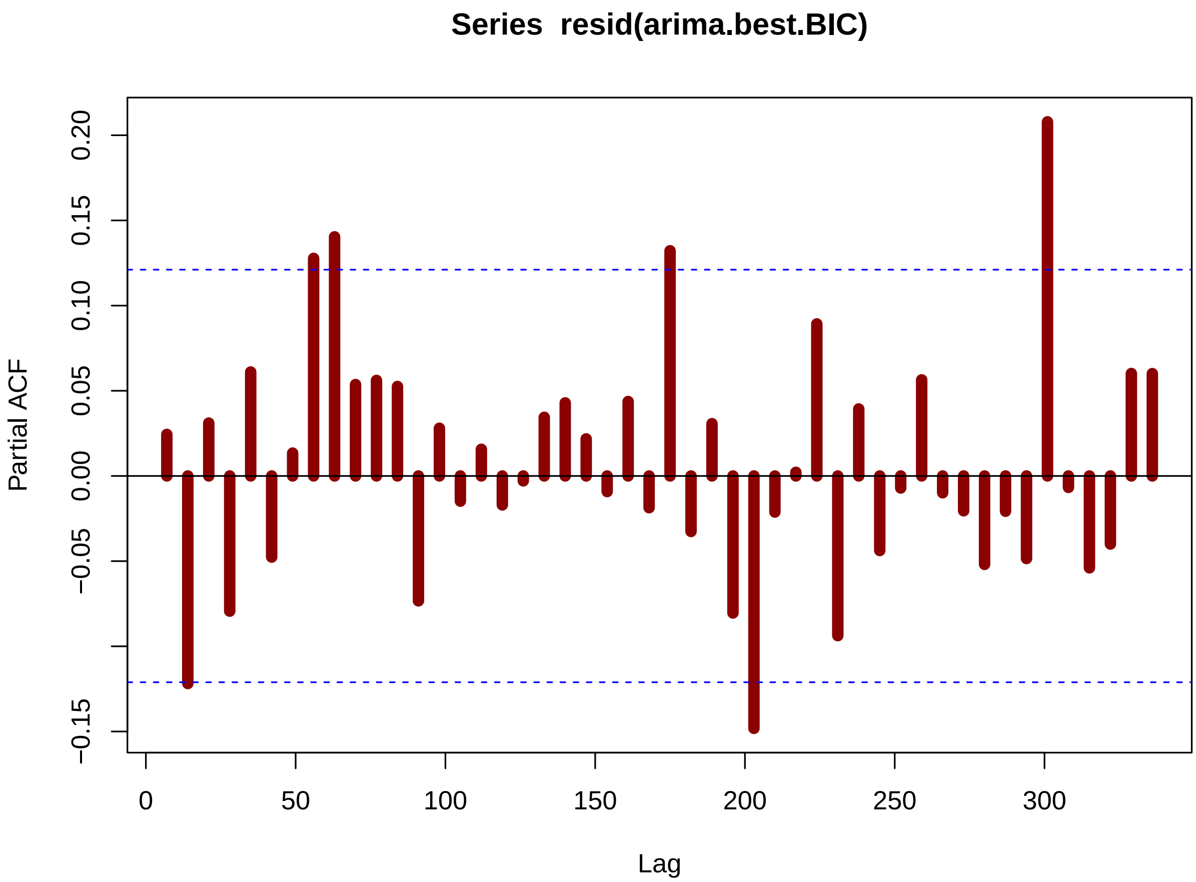
As shown by the results, both the ma1 and the drift are significant at 5% level.

Let's now inspect the ACF and PACF plots:

```r
par(mfrow=c(1,1))
acf(resid(arima.best.BIC),
    lag.max = 48,
    lwd = 7,
    col = "darkred",
    na.action = na.pass,
    ylim = c(-0.06, 0.06))
```




```
pacf(resid(arima.best.BIC),
     lag.max = 48,
     lwd = 7,
     col = "darkred",
     na.action = na.pass)
```



For this model way more spikes are observable, suggesting a check on the autocorrelation of the residuals:

```
Box.test(resid(arima.best.BIC), type = "Ljung-Box", lag =  5)
```

```
	Box-Ljung test

data:  resid(arima.best.BIC)
X-squared = 5.872, df = 5, p-value = 0.3189
```

```
bgtest(arima.best.BIC$residuals ~ fitted(arima.best.BIC), order = 5)
```

```
	Breusch-Godfrey test for serial correlation of order up to 5

data:  arima.best.BIC$residuals ~ fitted(arima.best.BIC)
LM test = 7.4087, df = 5, p-value = 0.192
```

According to the tests there is no significant autocorrelation in the residuals, even though the values are pretty low compared to the other models, suggesting there might be some issues.

Let's also check its values of BIC/AIC compared with the other models:

```r
BIC(arima.best.BIC, arima.best.AIC, model2)
```

```
               df       BIC
arima.best.BIC  3 -702.0548
arima.best.AIC 10 -679.5468
model2          5 -697.2276
```

```r
AIC(arima.best.BIC, arima.best.AIC, model2)
```

```
               df       AIC
arima.best.BIC  3 -712.7484
arima.best.AIC 10 -715.1920
model2          5 -715.0502
```

Also in this case, the values of BIC is higher for the automatic model but the value of AIC is higher for the manual one.

FORECASTING

This section is going to be about forecasting, where the performance of predicting the next values of two different models are compared.

Manual model

First of all, data is split in an "in sample" period and an "out-of-sample" period that is going to be used to test the model performances.

```
# Define the split point
split_point <- floor(0.9 * length(log_trend1_xts))

# Split the data into training and testing sets
train_data <- window(log_trend1_xts, end =
 ↪  index(log_trend1_xts)[split_point])
test_data <- window(log_trend1_xts, start =
 ↪  index(log_trend1_xts)[split_point + 1])
```

Then, the model is ran on the in-sample period:

```
manual_model <- Arima(train_data, order = c(1, 1, 2), include.constant =
 ↪  TRUE)
```

Let's now forecast the values for the out-of-sample period:

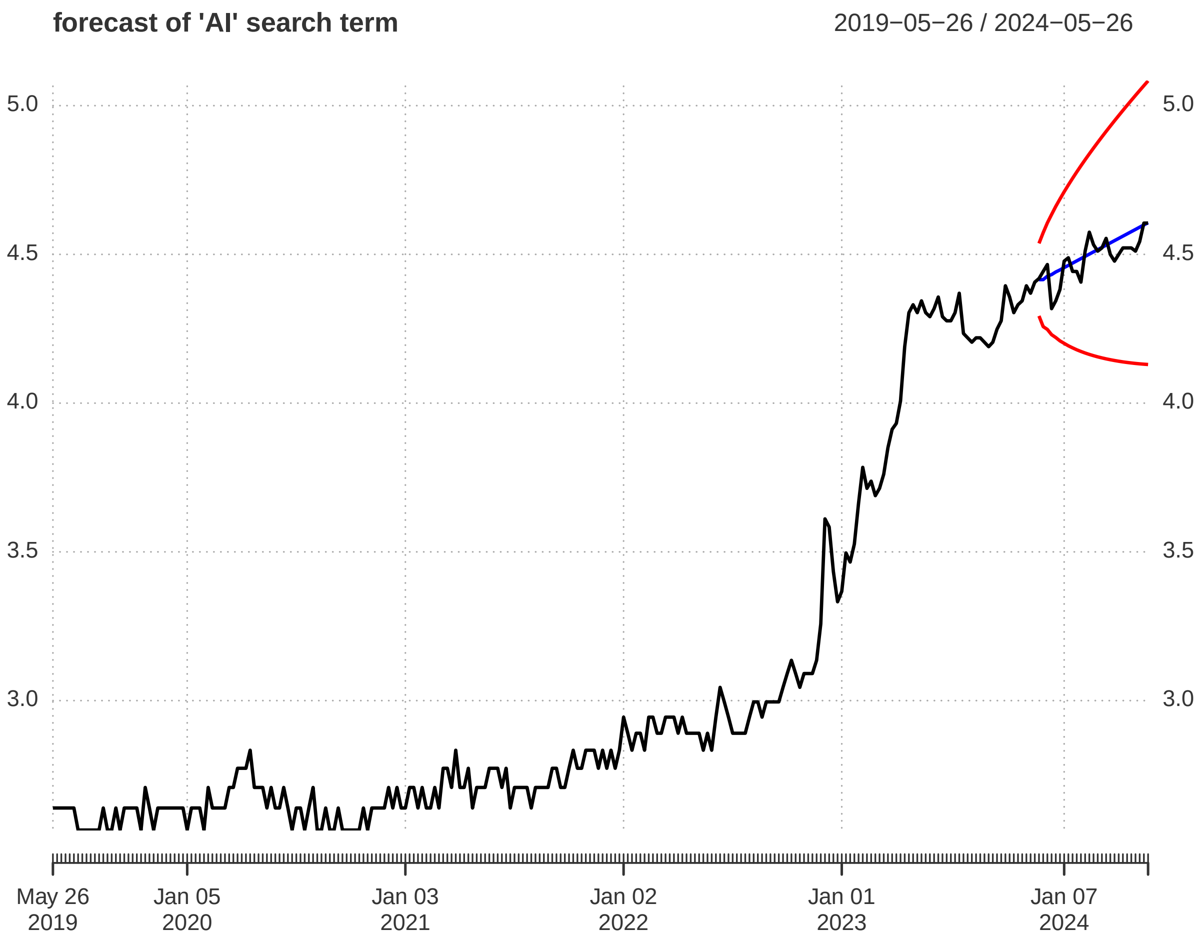
```
# Generate out-of-sample forecasts for the length of the test set
forecast_horizon <- length(test_data)
out_of_sample_forecast <- forecast(manual_model, h = forecast_horizon)
forecasts_data <- data.frame(f_mean  =
 ↪  as.numeric(out_of_sample_forecast$mean),
                             f_lower =
                              ↪  as.numeric(out_of_sample_forecast$lower[,
                              ↪  2]),
                             f_upper =
                              ↪  as.numeric(out_of_sample_forecast$upper[,
                              ↪  2]))
```

Now that the forecasts are generated, the data is manipulated a bit to make sure a graph can be shown:

```
AI.oos <- test_data
# Ensure names match for consistency
names(AI.oos) <- "V1"
colnames(train_data) <- "V1"
AI2 <- rbind(train_data[, "V1"], AI.oos)
forecasts_xts <- xts(forecasts_data, order.by = index(AI.oos))
forecasted_data <- merge(AI2, forecasts_xts)
```

Let's now see the graph of the time series with the forecasted values:

```r
plot(forecasted_data[, c("V1", "f_mean", "f_lower", "f_upper")],
     major.ticks = "years",
     grid.ticks.on = "years",
     grid.ticks.lty = 3,
     main = "forecast of 'AI' search term",
     col = c("black", "blue", "red", "red"))
```



Just from the graph the predictions look pretty good, but some measures have to be analyzed. The data has been transformed back to normal and error measures analyzed:

- **MAE** (Mean Absolute Error)
- **MSE** (Mean Square Error)
- **MAPE** (Mean Absolute Percentage Error)

- **AMAPE** (Adjusted Mean Absolute Percentage Error)

```r
eval_data_oos <- tail(forecasted_data, length(test_data))
eval_data_oos$V1 <- exp(eval_data_oos$V1)
eval_data_oos$f_mean <- exp(eval_data_oos$f_mean)
eval_data_oos$f_lower <- exp(eval_data_oos$f_lower)
eval_data_oos$f_upper <- exp(eval_data_oos$f_upper)
eval_data_oos$mae   <-  abs(eval_data_oos$V1 - eval_data_oos$f_mean)
eval_data_oos$mse   <-  (eval_data_oos$V1 - eval_data_oos$f_mean) ^ 2
eval_data_oos$mape  <-  abs((eval_data_oos$V1 -
↪   eval_data_oos$f_mean)/eval_data_oos$V1)
eval_data_oos$amape <-  abs((eval_data_oos$V1 -
↪   eval_data_oos$f_mean)/(eval_data_oos$V1 + eval_data_oos$f_mean))
eval_data_oos
```

```
            V1   f_mean  f_lower   f_upper       mae         mse        mape
2023-11-26  83  82.68320 73.19259  93.40442 0.3167997  0.10036202 0.003816863
2023-12-03  85  82.68854 70.60995  96.83330 2.3114628  5.34286019 0.027193680
2023-12-10  87  83.66206 69.94611 100.06762 3.3379394 11.14183972 0.038367120
2023-12-17  75  84.10251 68.72737 102.91724 9.1025071 82.85563541 0.121366761
2023-12-24  77  84.84834 68.06671 105.76744 7.8483445 61.59651181 0.101926552
2023-12-31  80  85.43113 67.30648 108.43649 5.4311327 29.49720221 0.067889159
2024-01-07  88  86.11260 66.74871 111.09397 1.8874021  3.56228679 0.021447751
2024-01-14  89  86.74658 66.20154 113.66757 2.2534247  5.07792301 0.025319379
2024-01-21  85  87.41478 65.74648 116.22437 2.4147757  5.83114148 0.028409125
2024-01-28  85  88.07161 65.32361 118.74127 3.0716086  9.43477945 0.036136572
2024-02-04  82  88.74260 64.95325 121.24488 6.7426019 45.46268004 0.082226852
2024-02-11  91  89.41355 64.61488 123.72976 1.5864466  2.51681271 0.017433479
2024-02-18  97  90.09246 64.31209 126.20724 6.9075431 47.71415141 0.071211784
2024-02-25  93  90.77491 64.03675 128.67742 2.2250932  4.95103989 0.023925734
2024-03-03  91  91.46342 63.78821 131.14584 0.4634247  0.21476248 0.005092579
2024-03-10  92  92.15666 63.56257 133.61401 0.1566630  0.02454331 0.001702859
2024-03-17  95  92.85544 63.35838 136.08511 2.1445639  4.59915426 0.022574357
2024-03-24  90  93.55935 63.17336 138.56082 3.5593509 12.66897910 0.039548344
2024-03-31  88  94.26869 63.00611 141.04324 6.2686895 39.29646793 0.071235108
2024-04-07  90  94.98336 62.85510 143.53390 4.9833571 24.83384838 0.055370635
2024-04-14  92  95.70347 62.71918 146.03434 3.7034701 13.71569097 0.040255110
2024-04-21  92  96.42903 62.59721 148.54587 4.4290274 19.61628338 0.048141602
2024-04-28  92  97.16009 62.48826 151.06972 5.1600938 26.62656798 0.056087976
2024-05-05  91  97.89670 62.39145 153.60699 6.8966980 47.56444288 0.075787890
2024-05-12  94  98.63889 62.30603 156.15872 4.6388893 21.51929351 0.049349886
2024-05-19 100  99.38671 62.23131 158.72584 0.6132941  0.37612968 0.006132941
```

```
2024-05-26 100 100.14019 62.16667 161.30924 0.1401928  0.01965402 0.001401928
                  amape
2023-11-26 0.0019120807
2023-12-03 0.0137842623
2023-12-10 0.0195587668
2023-12-17 0.0572115881
2023-12-24 0.0484919667
2023-12-31 0.0328301729
2024-01-07 0.0108401239
2024-01-14 0.0128220122
2024-01-21 0.0140056190
2024-01-28 0.0177476169
2024-02-04 0.0394898625
2024-02-11 0.0087933891
2024-02-18 0.0369204788
2024-02-25 0.0121077098
2024-03-03 0.0025398226
2024-03-10 0.0008507052
2024-03-17 0.0114160331
2024-03-24 0.0193907361
2024-03-31 0.0343925745
2024-04-07 0.0269394891
2024-04-14 0.0197304297
2024-04-21 0.0235050163
2024-04-28 0.0272789767
2024-05-05 0.0365104210
2024-05-12 0.0240807517
2024-05-19 0.0030759028
2024-05-26 0.0007004730
```

Let's save the average errors for the out-of-sample evaluation data:

```r
perf_manual_model = colMeans(eval_data_oos[, c("mae", "mse", "mape",
 ↪  "amape")])
```

Values will be analyzed and compared to the ones of the automatic model.

Automatic model

As an automatic model to compare, ARIMA(0,1,8) is chosen due to its better behavior of the residuals and similar AIC/BIC values.

```
automatic_model <- Arima(train_data, order = c(0, 1, 8),
 ↪  include.constant = TRUE)
```

Let's now forecast the values for the out-of-sample period:

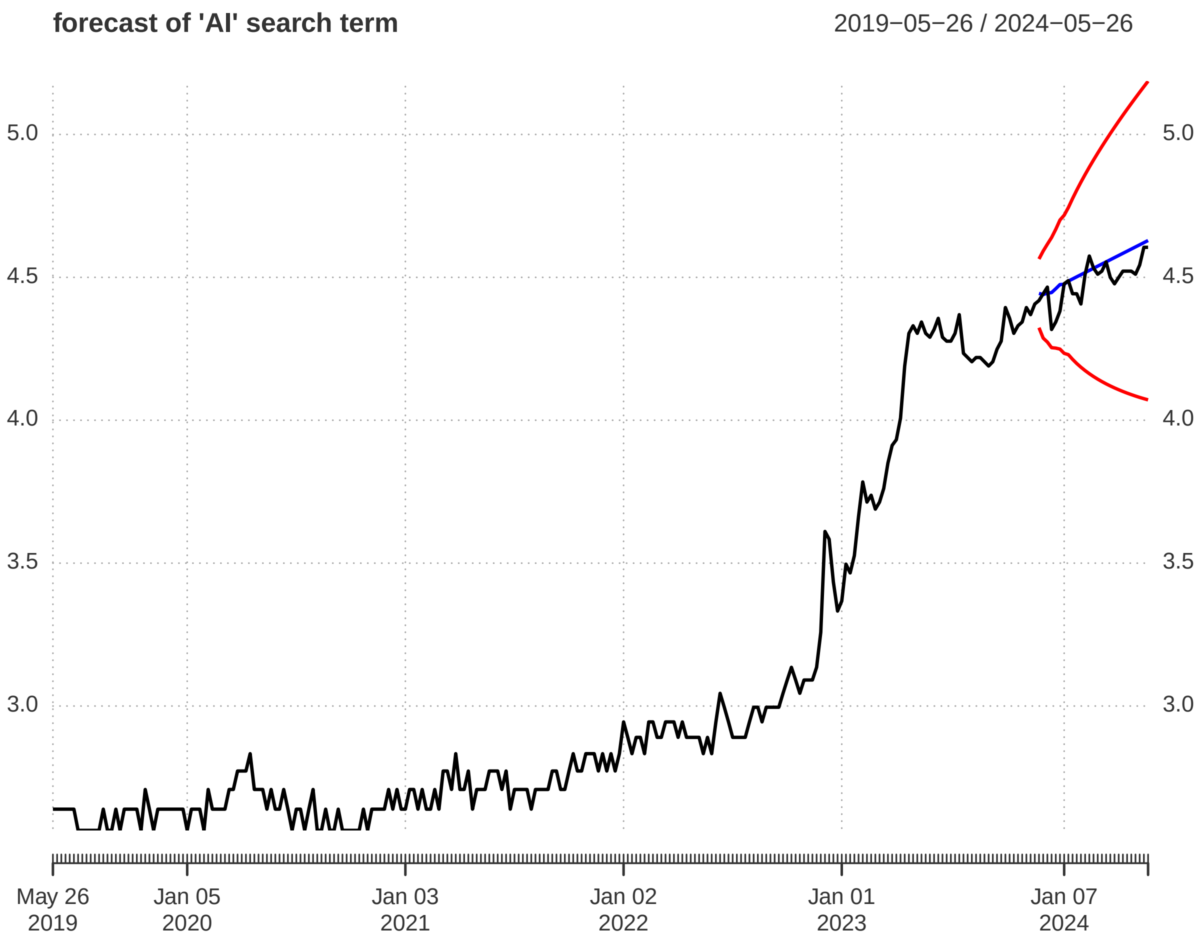
```
# Generate out-of-sample forecasts for the length of the test set
out_of_sample_forecast_auto <- forecast(automatic_model, h =
 ↪  forecast_horizon)

forecasts_data_auto <- data.frame(f_mean  =
 ↪  as.numeric(out_of_sample_forecast_auto$mean),
                             f_lower =
                              ↪  as.numeric(out_of_sample_forecast_auto$lower[,
                              ↪  2]),
                             f_upper =
                              ↪  as.numeric(out_of_sample_forecast_auto$upper[,
                              ↪  2]))
```

Now that the forecasts are generated, the data is manipulated a bit to make sure a graph can be shown:

```
forecasts_auto_xts <- xts(forecasts_data_auto,
                          order.by = index(AI.oos))
forecasted_data_auto <- merge(AI2, forecasts_auto_xts)
```

Let's now see the graph of the time series with the forecasted values by the automatic model:

```
plot(forecasted_data_auto[, c("V1", "f_mean", "f_lower", "f_upper")],
     major.ticks = "years",
     grid.ticks.on = "years",
     grid.ticks.lty = 3,
     main = "forecast of 'AI' search term",
     col = c("black", "blue", "red", "red"))
```



Also in this case, the prediction don't look far from the actual values, but we can't tell just by looking at the graph which of the two models is performing better.

So, again the error measures are computed and saved to be compared with the other model:

```
eval_data_oos_auto <- tail(forecasted_data_auto, length(test_data))
eval_data_oos_auto$V1 <- exp(eval_data_oos_auto$V1)
eval_data_oos_auto$f_mean <- exp(eval_data_oos_auto$f_mean)
eval_data_oos_auto$f_lower <- exp(eval_data_oos_auto$f_lower)
eval_data_oos_auto$f_upper <- exp(eval_data_oos_auto$f_upper)
eval_data_oos_auto$mae    <-  abs(eval_data_oos_auto$V1 -
 ↪  eval_data_oos_auto$f_mean)
eval_data_oos_auto$mse    <-  (eval_data_oos_auto$V1 -
 ↪  eval_data_oos_auto$f_mean) ^ 2
```

```r
eval_data_oos_auto$mape  <-  abs((eval_data_oos_auto$V1 -
 ↪  eval_data_oos_auto$f_mean)/eval_data_oos_auto$V1)
eval_data_oos_auto$amape <-  abs((eval_data_oos_auto$V1 -
 ↪  eval_data_oos_auto$f_mean)/(eval_data_oos_auto$V1 +
 ↪  eval_data_oos_auto$f_mean))
```

```r
perf_auto_model = colMeans(eval_data_oos_auto[, c("mae", "mse", "mape",
 ↪  "amape")])
```

Performance comparison

Let's now compare the error measures of the two models to select the best one:

```r
# Combine the performance metrics into a single data frame
combined_perf <- data.frame(
  ARIMA_1_1_2 = perf_manual_model,
  ARIMA_0_1_8 = perf_auto_model
)

# Print the combined table
print(combined_perf)
```

```
      ARIMA_1_1_2 ARIMA_0_1_8
mae    3.65165914  4.53356390
mse   19.48744607 31.70184775
mape   0.04219822  0.05237733
amape  0.02062693  0.02519301
```

- **Overall Performance**: The ARIMA(1,1,2) model outperforms the ARIMA(0,1,8) model in all error metrics (MAE, MSE, MAPE, and AMAPE).
- **Model Choice**: Based on these error measures, the ARIMA(1,1,2) model appears to be a better choice for forecasting, as it consistently provides more accurate predictions with smaller errors.
- **Error Distribution**: The significant difference in MSE suggests that the ARIMA(1,1,2) model not only has smaller average errors but also fewer large errors compared to the ARIMA(0,1,8) model.

These results support the preference for the ARIMA(1,1,2) model due to its superior accuracy and robustness in predicting the data.

SECOND TIME SERIES

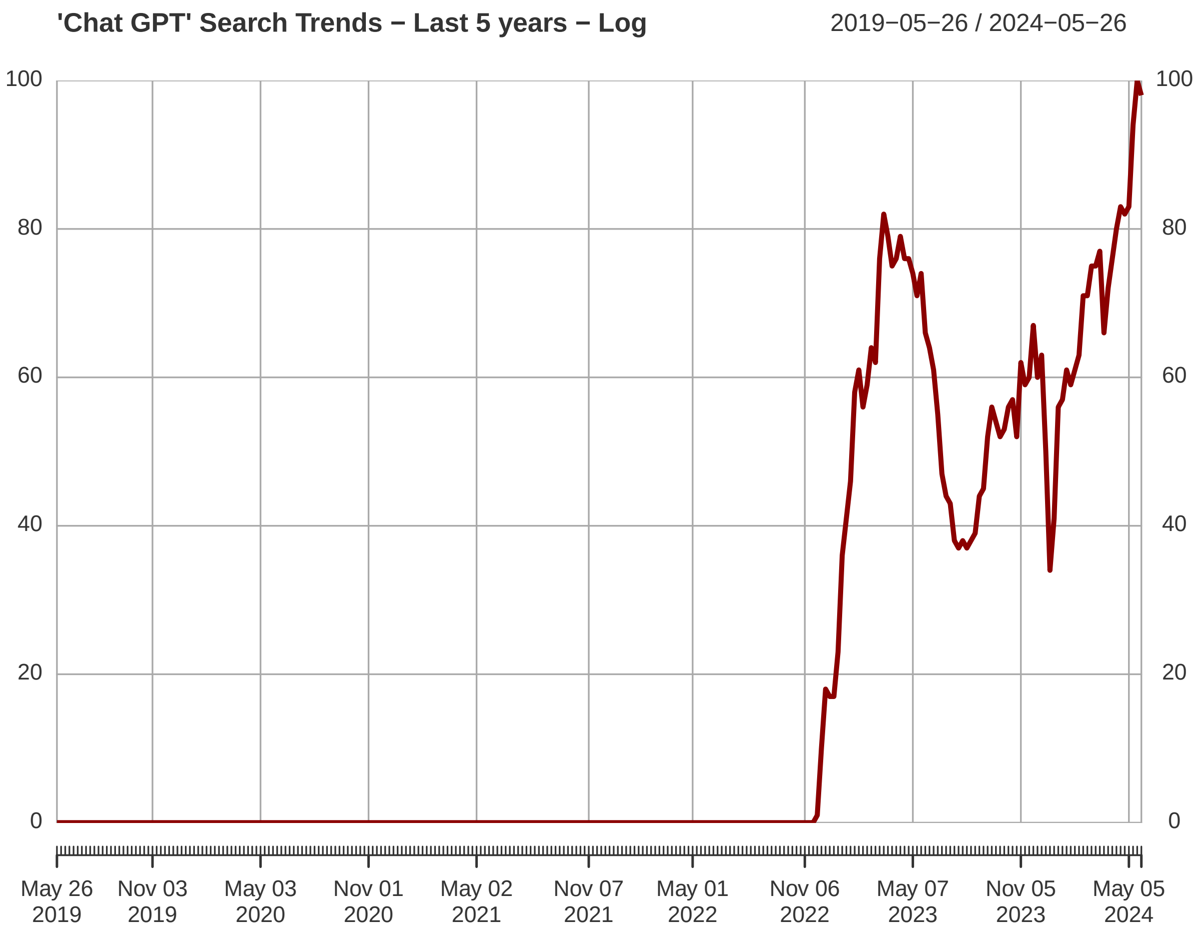
Now are going to load the second time series by using the "gtrends" package, and it is about the searches of the phrase "Chat GPT".

Loading data and transformations

```
# Define the search term
search_term <- "Chat GPT"

# Calculate the current date and the date 5 years ago
end_date <- as.Date("2024-05-26")
start_date <- end_date %m-% years(5)

# Format the date range
time_frame <- paste0(start_date, " ", end_date)

# Fetch the trend data
trend_data <- gtrends(search_term, time = time_frame)

# Extract the interest over time data
interest_over_time <- trend_data$interest_over_time

gpt_data = interest_over_time[,1:2]

gpt_data$date <- as.Date(gpt_data$date, format = "%Y-%m-%d")
```

```
gpt_data$hits <- as.numeric(gsub("<1", "0", gpt_data$hits))
gpt_data$hits <- as.numeric(gpt_data$hits)
```

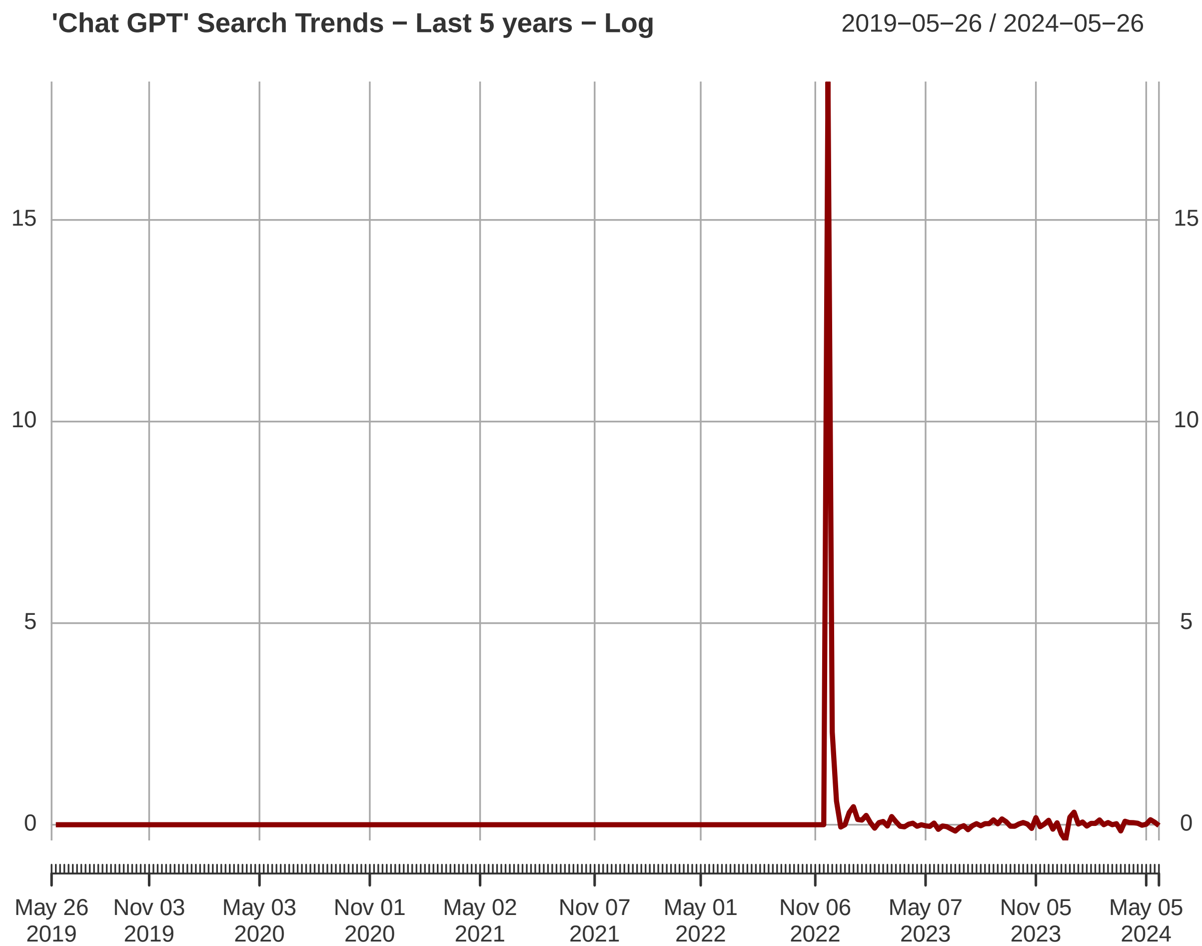
```
# Convert the data frame to xts object
gpt_xts <- xts(gpt_data$hits, order.by = gpt_data$date)
plot(gpt_xts,
     type = "l",
     col  = "darkred",
     lwd  = 3,
     main = "'Chat GPT' Search Trends - Last 5 years - Log")
```




```
# Log transformation
gpt_xts <- gpt_xts + 1e-8
log_gpt_xts <- log(gpt_xts)
```

Checking integration order

```
diff_log_gpt_xts <- diff(log_gpt_xts, differences = 1)
plot(diff_log_gpt_xts,
type = "l",
col = "darkred",
lwd = 3,
main = "'Chat GPT' Search Trends - Last 5 years - Log")
```




```r
diff_log_gpt_xts <- diff_log_gpt_xts[-1,]
gpt_adf_test <- ur.df(diff_log_gpt_xts, type = "drift", lags = 0)
summary(gpt_adf_test)
```

```
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression drift


Call:
lm(formula = z.diff ~ z.lag.1 + 1)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-0.4351 -0.0777 -0.0777 -0.0777 18.3430

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.07766    0.07129   1.089    0.277
z.lag.1     -0.87782    0.06179 -14.206   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.146 on 258 degrees of freedom
Multiple R-squared:  0.4389,    Adjusted R-squared:  0.4367
F-statistic: 201.8 on 1 and 258 DF,  p-value: < 2.2e-16


Value of test-statistic is: -14.2062 100.9084

Critical values for test statistics:
      1pct  5pct 10pct
tau2 -3.44 -2.87 -2.57
phi1  6.47  4.61  3.79
```

The test statistic is far below the critical value, so we can reject the null-hypothesis of non-stationarity.

Let's check more tests:

```
gpt_pp.test <- ur.pp(diff_log_gpt_xts,   # tested series
                 type = c("Z-tau"),     # standardization of the test
                  ↪  statistic needed
                 model = c("constant")) # constant deterministic
                  ↪  component
# which means we assume that any trends in the data are stochastic
summary(gpt_pp.test)
```

```
##################################
# Phillips-Perron Unit Root Test #
##################################
```

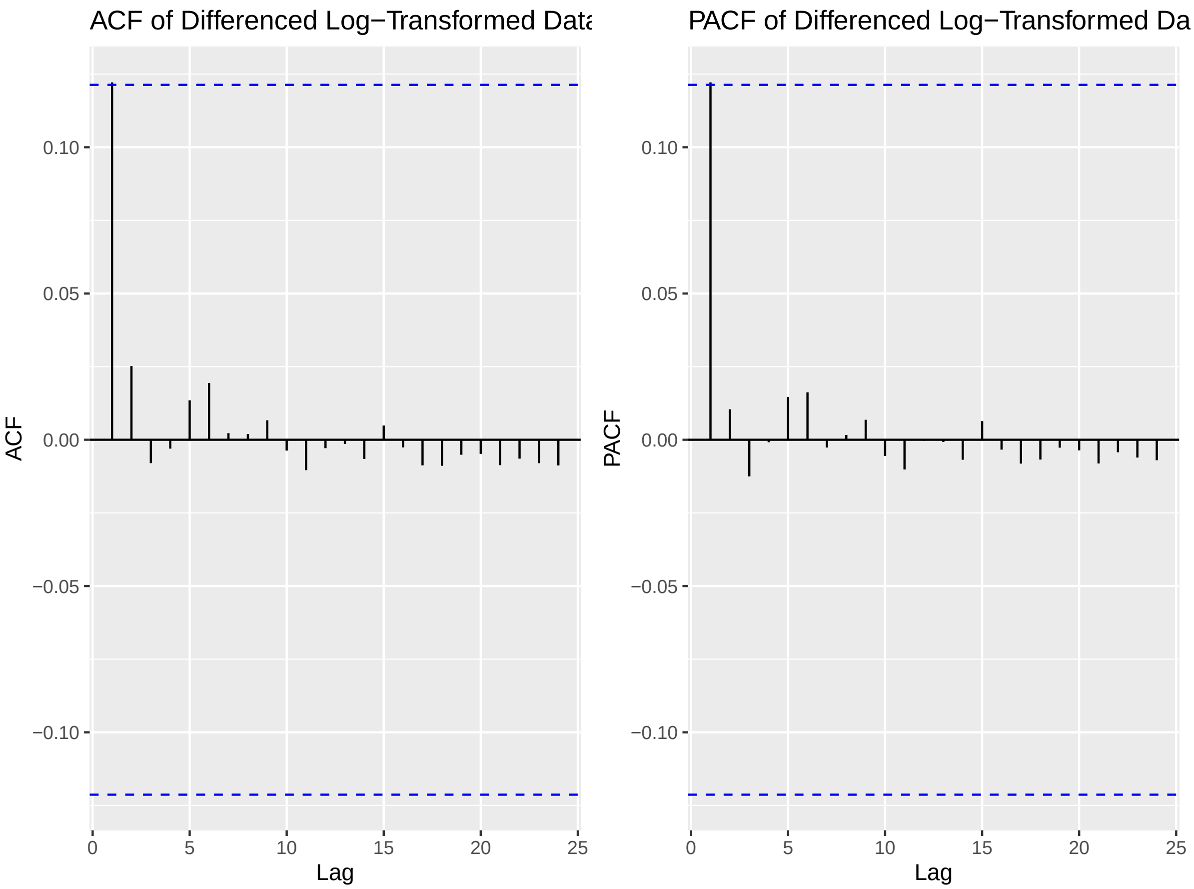
```
Test regression with intercept


Call:
lm(formula = y ~ y.l1)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.4351 -0.0777 -0.0777 -0.0777 18.3430 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)  0.07766    0.07129   1.089   0.2770  
y.l1         0.12218    0.06179   1.977   0.0491 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.146 on 258 degrees of freedom
Multiple R-squared:  0.01493,	Adjusted R-squared:  0.01111 
F-statistic:  3.91 on 1 and 258 DF,  p-value: 0.04907


Value of test-statistic, type: Z-tau  is: -14.2098 

         aux. Z statistics
Z-tau-mu            1.0897

Critical values for Z statistics: 
                     1pct      5pct     10pct
critical values -3.457006 -2.872754 -2.572697
```

This test also confirms the non stationarity, so we can say that the series is $\sim I(1)$.

ACF - PACF Evaluations

```r
# ACF and PACF plots of the differenced series
acf_plot <- ggAcf(diff_log_gpt_xts, main = "ACF of Differenced
  Log-Transformed Data")
pacf_plot <- ggPacf(diff_log_gpt_xts, main = "PACF of Differenced
  Log-Transformed Data")
```

```r
gridExtra::grid.arrange(acf_plot, pacf_plot, ncol = 2)
```



From the plots we can see AR(1) and MA(1).

Let's now check ACF and PACF for the residuals:

ARIMA

Fitting two models

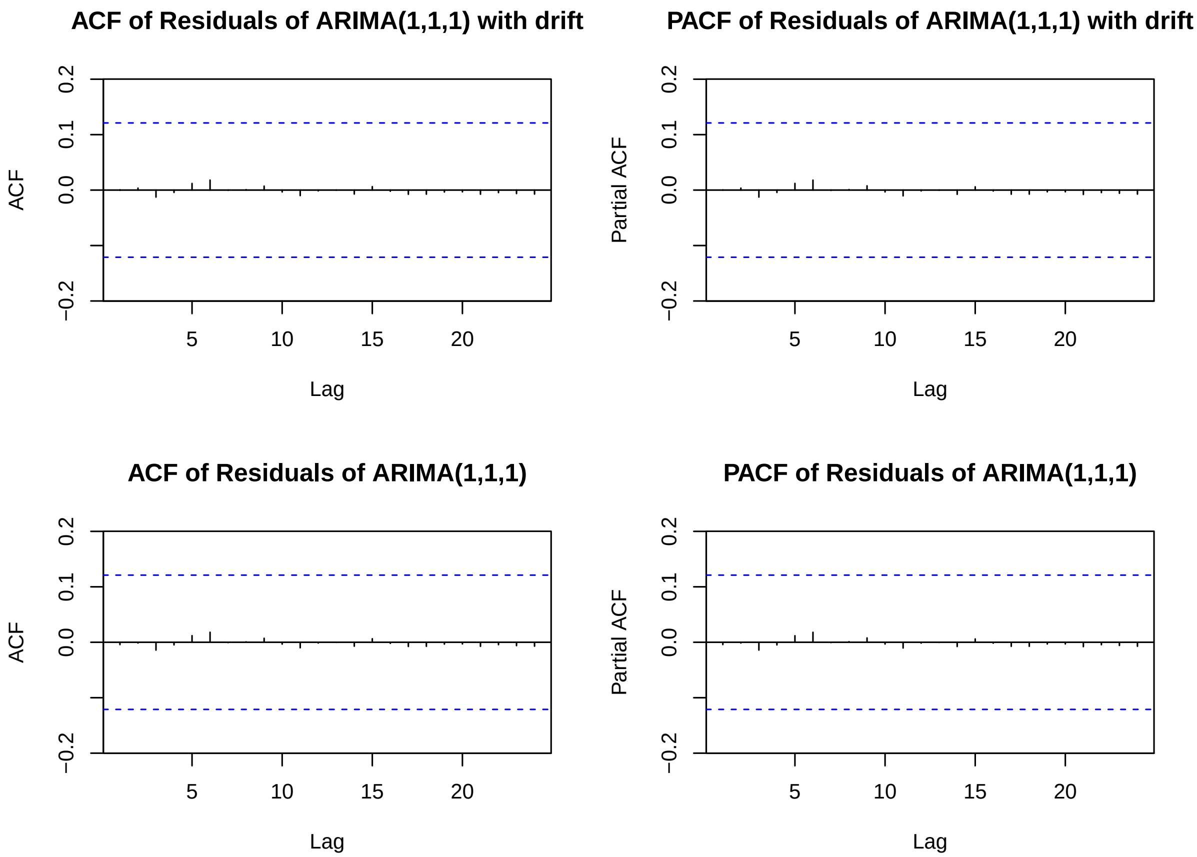
We are going to compare ARIMA (1,1,1) with drift and without it:

```r
# Fit two different ARIMA models - constant and no constant
ts2_model1 <- Arima(log_gpt_xts, order = c(1, 1, 1), include.constant =
↪   TRUE)
ts2_model2 <- Arima(log_gpt_xts, order = c(1, 1, 1), include.constant =
↪   FALSE)
residuals_ts2_model1 <- residuals(ts2_model1)
residuals_ts2_model2 <- residuals(ts2_model2)
```

Diagnostics

Let's check their respective ACF and PACF plots:

```r
par(mfrow = c(2, 2))
Acf(residuals_ts2_model1, main = "ACF of Residuals of ARIMA(1,1,1) with
↪   drift")
Pacf(residuals_ts2_model1, main = "PACF of Residuals of ARIMA(1,1,1)
↪   with drift")
Acf(residuals_ts2_model2, main = "ACF of Residuals of ARIMA(1,1,1)")
Pacf(residuals_ts2_model2, main = "PACF of Residuals of ARIMA(1,1,1)")
```



From the plots it looks like there is no autocorrelation in the residuals for both models, let's now check with the tests:

```
# Perform tests on residuals to check for white noise
lb_test_ts2_model1 <- Box.test(residuals_ts2_model1, lag = 1, type =
↪   "Ljung-Box")
lb_test_ts2_model2 <- Box.test(residuals_ts2_model2, lag = 1, type =
↪   "Ljung-Box")
bg_test_ts2_model1 <- bgtest(residuals_ts2_model1 ~ fitted(ts2_model1),
↪   order = 1)
bg_test_ts2_model2 <- bgtest(residuals_ts2_model2 ~ fitted(ts2_model2),
↪   order = 1)
```

After comparing lags = 5 (rule of thumb) and lags = 1 (intuition), 1 was selected since it makes more sense in the context of application.

```r
# Create a table to compare results
comparison_table_2 <- data.frame(
  Model = c("ARIMA(1,1,1) with drift", "ARIMA(1,1,1)"),
  AIC = c(AIC(ts2_model1), AIC(ts2_model2)),
  BIC = c(BIC(ts2_model1), BIC(ts2_model2)),
  LjungBox_p_value = c(lb_test_ts2_model1$p.value,
   ↪  lb_test_ts2_model2$p.value), BG_statistic =
   ↪  c(bg_test_ts2_model1$p.value, bg_test_ts2_model2$p.value))

# Print the comparison table
print(comparison_table_2)
```

```
                    Model      AIC      BIC LjungBox_p_value BG_statistic
1 ARIMA(1,1,1) with drift 816.8274 831.0855        0.9962941    0.9573842
2            ARIMA(1,1,1) 815.9925 826.6861        0.9460117    0.9858002
```

None of the models shows autocorrelation in the residuals, ARIMA (1,1,1) is selected for lower AIC and BIC values.

AUTO-ARIMA

Automatic model selection

Let's look for the best models in terms of AIC and BIC:

```r
ts2_arima.best.AIC <-
  auto.arima(log_gpt_xts,
             d = 1,              # parameter d of ARIMA model
             max.p = 2,          # Maximum value of p
             max.q = 2,          # Maximum value of q
             max.order = 4,     # maximum p+q
             start.p = 0,        # Starting value of p in stepwise
              ↪  procedure
             start.q = 0,        # Starting value of q in stepwise
              ↪  procedure
             ic = "aic",         # Information criterion to be used in
              ↪  model selection.
             stepwise = FALSE,  # if FALSE considers all models
             allowdrift = TRUE, # include a constant
```

```r
               trace = TRUE)      # show summary of all models considered
```

```
Fitting models using approximations to speed things up...

ARIMA(0,1,0)                    : 816.0132
ARIMA(0,1,0) with drift         : 816.4792
ARIMA(0,1,1)                    : 813.9386
ARIMA(0,1,1) with drift         : 814.6977
ARIMA(0,1,2)                    : 815.6699
ARIMA(0,1,2) with drift         : 816.5003
ARIMA(1,1,0)                    : 814.7477
ARIMA(1,1,0) with drift         : 815.5497
ARIMA(1,1,1)                    : 816.6958
ARIMA(1,1,1) with drift         : 817.5255
ARIMA(1,1,2)                    : 818.6703
ARIMA(1,1,2) with drift         : 819.4913
ARIMA(2,1,0)                    : 817.695
ARIMA(2,1,0) with drift         : 818.5224
ARIMA(2,1,1)                    : 819.6817
ARIMA(2,1,1) with drift         : 820.5105
ARIMA(2,1,2)                    : 821.6732
ARIMA(2,1,2) with drift         : 822.4687

Now re-fitting the best model(s) without approximations...




Best model: ARIMA(0,1,1)
```

```r
ts2_arima.best.BIC <-
  auto.arima(log_gpt_xts,
             d = 1,               # parameter d of ARIMA model
             max.p = 2,           # Maximum value of p
             max.q = 2,           # Maximum value of q
             max.order = 4,     # maximum p+q
             start.p = 0,         # Starting value of p in stepwise
              ↪  procedure
```

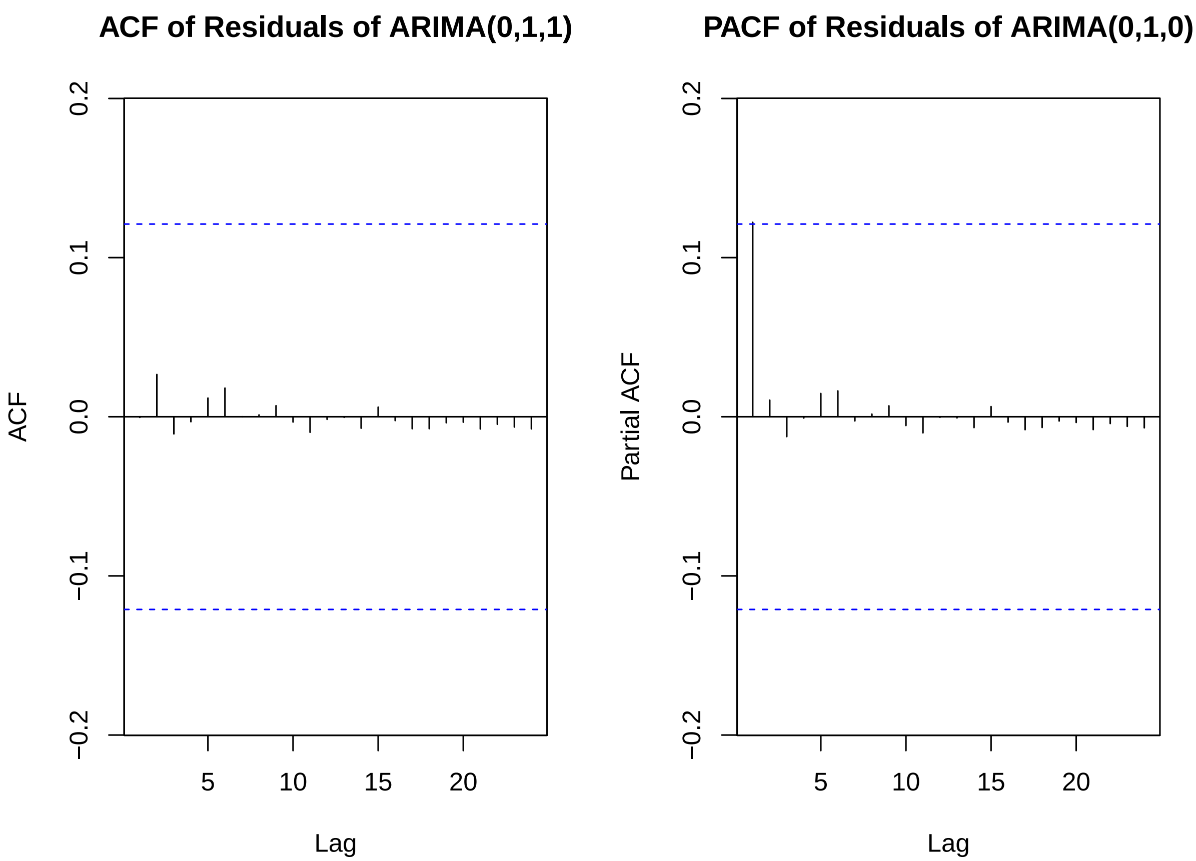
```r
                 start.q = 0,        # Starting value of q in stepwise
                  ↪  procedure
                 ic = "bic",         # Information criterion to be used in
                  ↪  model selection.
                 stepwise = FALSE,  # if FALSE considers all models
                 allowdrift = TRUE, # include a constant
                 trace = TRUE)      # show summary of all models considered
```

```
Fitting models using approximations to speed things up...

ARIMA(0,1,0)                    : 819.5778
ARIMA(0,1,0) with drift         : 823.6083
ARIMA(0,1,1)                    : 821.0676
ARIMA(0,1,1) with drift         : 825.3913
ARIMA(0,1,2)                    : 826.3635
ARIMA(0,1,2) with drift         : 830.7584
ARIMA(1,1,0)                    : 821.8767
ARIMA(1,1,0) with drift         : 826.2433
ARIMA(1,1,1)                    : 827.3893
ARIMA(1,1,1) with drift         : 831.7836
ARIMA(1,1,2)                    : 832.9284
ARIMA(1,1,2) with drift         : 837.3139
ARIMA(2,1,0)                    : 828.3885
ARIMA(2,1,0) with drift         : 832.7805
ARIMA(2,1,1)                    : 833.9398
ARIMA(2,1,1) with drift         : 838.3331
ARIMA(2,1,2)                    : 839.4958
ARIMA(2,1,2) with drift         : 843.8558

Now re-fitting the best model(s) without approximations...




Best model: ARIMA(0,1,0)
```

From these automatic selection methods, ARIMA (0,1,1) and ARIMA (0,1,0) were selected.

Automatic models diagnostics

```
ts2_arima.best.AIC_res = residuals(ts2_arima.best.AIC)
ts2_arima.best.BIC_res = residuals(ts2_arima.best.BIC)
par(mfrow = c(1, 2))
Acf(ts2_arima.best.AIC_res, main = "ACF of Residuals of ARIMA(0,1,1)")
Pacf(ts2_arima.best.BIC_res, main = "PACF of Residuals of ARIMA(0,1,0)")
```



ARIMA (0,1,1) shows no autocorrelation in the residuals while ARIMA (0,1,0) shows just one spike at lag 1.

Let's continue with the tests:

```
lb_test_ts2_model3 <- Box.test(ts2_arima.best.AIC_res, lag = 1, type =
 ↪  "Ljung-Box")
```

```r
lb_test_ts2_model4 <- Box.test(ts2_arima.best.BIC_res, lag = 1, type =
  "Ljung-Box")
bg_test_ts2_model3 <- bgtest(ts2_arima.best.AIC_res ~
  fitted(ts2_arima.best.AIC), order = 1)
bg_test_ts2_model4 <- bgtest(ts2_arima.best.BIC_res ~
  fitted(ts2_arima.best.BIC), order = 1)
```

```r
# Create a table to compare results
comparison_table_3 <- data.frame(
  Model = c("ARIMA(0,1,1)", "ARIMA(0,1,0)"),
  AIC = c(AIC(ts2_arima.best.AIC), AIC(ts2_arima.best.BIC)),
  BIC = c(BIC(ts2_arima.best.AIC), BIC(ts2_arima.best.BIC)),
  LjungBox_p_value = c(lb_test_ts2_model3$p.value,
   lb_test_ts2_model4$p.value), BG_statistic =
   c(bg_test_ts2_model3$p.value, bg_test_ts2_model4$p.value))

# Print the comparison table
print(comparison_table_3)
```

```
         Model      AIC      BIC LjungBox_p_value BG_statistic
1 ARIMA(0,1,1) 814.2354 821.3645       0.99479494   0.96827473
2 ARIMA(0,1,0) 816.2953 819.8598       0.04666743   0.04370179
```

The tests confirm the presence of autocorrelation for the ARIMA (0,1,0) when considering one lag.

For this reason, after checking for the significance of its coefficient we are going to select ARIMA(0,1,1)

```r
coeftest(ts2_arima.best.AIC)
```

```
z test of coefficients:

    Estimate Std. Error z value Pr(>|z|)
ma1 0.121174   0.059452  2.0382  0.04153 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we can see the moving average term is significant at 5% level.

FORECASTING

Manual selection

Let's now compare ARIMA(1,1,1) and ARIMA(0,1,1) models in terms of forecasting:

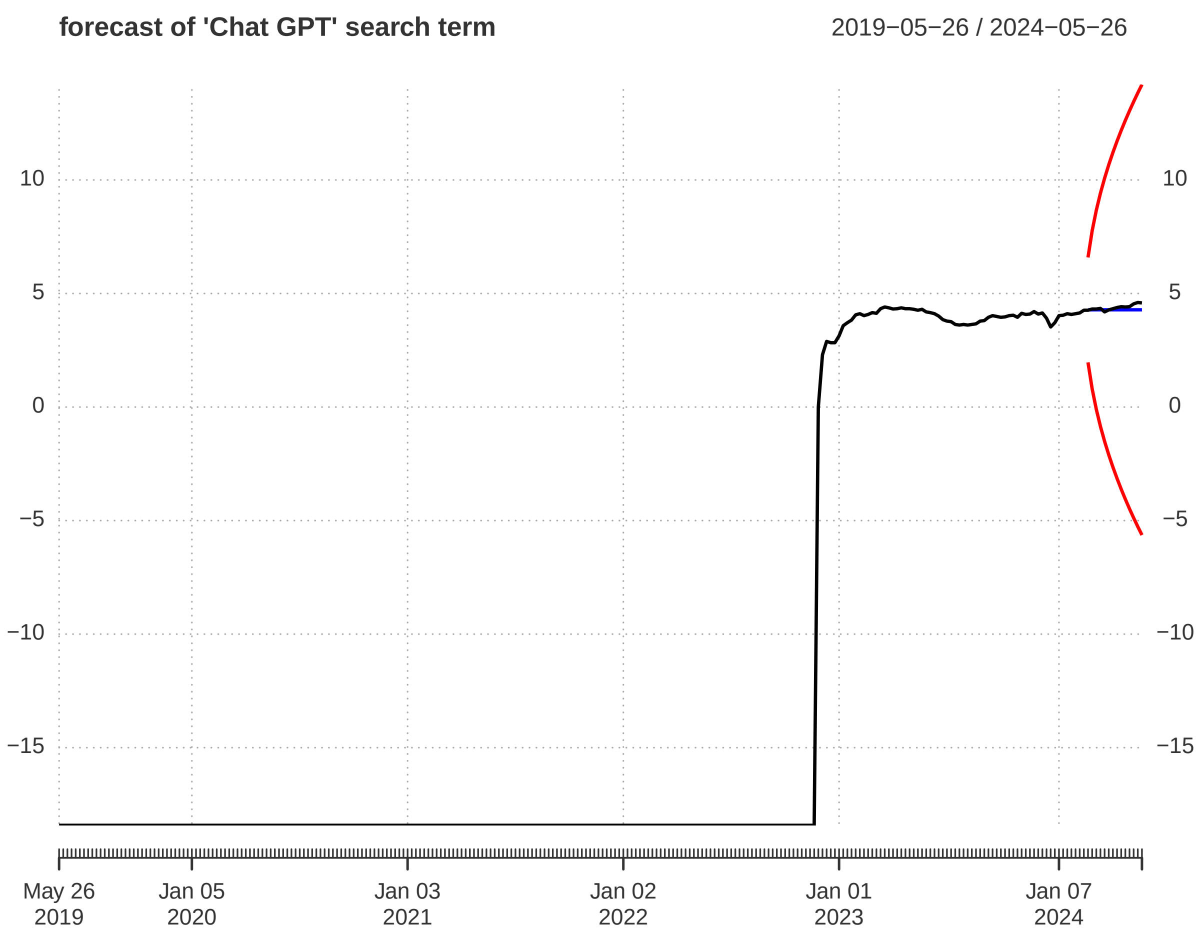
```
# Split the data into training and testing sets
split_point <- floor(0.95 * length(log_gpt_xts))
ts2_train_data <- window(log_gpt_xts, end =
↪  index(log_gpt_xts)[split_point])
ts2_test_data <- window(log_gpt_xts, start =
↪  index(log_gpt_xts)[split_point + 1])
```

```
ts2_manual_model <- Arima(ts2_train_data, order = c(1, 1, 1))
ts2_auto_model <- Arima(ts2_train_data, order = c(0,1,1))
```

```
# Generate out-of-sample forecasts for the length of the test set
forecast_horizon <- length(ts2_test_data)
ts2_out_of_sample_forecast <- forecast(ts2_manual_model, h =
↪  forecast_horizon)
ts2_forecasts_data <- data.frame(f_mean  =
↪  as.numeric(ts2_out_of_sample_forecast$mean),
                             f_lower =
                              ↪  as.numeric(ts2_out_of_sample_forecast$lower[,
                              ↪  2]),
                             f_upper =
                              ↪  as.numeric(ts2_out_of_sample_forecast$upper[,
                              ↪  2]))
```

```
gpt.oos <- ts2_test_data
# Ensure names match for consistency
names(gpt.oos) <- "V1"
colnames(ts2_train_data) <- "V1"
gpt2 <- rbind(ts2_train_data[, "V1"], gpt.oos)
ts2_forecasts_xts <- xts(ts2_forecasts_data, order.by = index(gpt.oos))
ts2_forecasted_data <- merge(gpt2, ts2_forecasts_xts)
```

```r
plot(ts2_forecasted_data[, c("V1", "f_mean", "f_lower", "f_upper")],
     major.ticks = "years",
     grid.ticks.on = "years",
     grid.ticks.lty = 3,
     main = "forecast of 'Chat GPT' search term",
     col = c("black", "blue", "red", "red"))
```




```r
eval_data_oos <- tail(ts2_forecasted_data, length(ts2_test_data))
eval_data_oos$V1 <- exp(eval_data_oos$V1)
eval_data_oos$f_mean <- exp(eval_data_oos$f_mean)
eval_data_oos$f_lower <- exp(eval_data_oos$f_lower)
eval_data_oos$f_upper <- exp(eval_data_oos$f_upper)
eval_data_oos$mae    <-   abs(eval_data_oos$V1 - eval_data_oos$f_mean)
```

```r
eval_data_oos$mse    <-  (eval_data_oos$V1 - eval_data_oos$f_mean) ^ 2
eval_data_oos$mape  <-  abs((eval_data_oos$V1 -
↪  eval_data_oos$f_mean)/eval_data_oos$V1)
eval_data_oos$amape <-  abs((eval_data_oos$V1 -
↪  eval_data_oos$f_mean)/(eval_data_oos$V1 + eval_data_oos$f_mean))
eval_data_oos
```

```
            V1   f_mean     f_lower      f_upper        mae         mse
2024-02-25  71 72.10376 7.157822941     726.3315  1.1037582   1.2182821
2024-03-03  75 72.35574 2.234929701    2342.5139  2.6442550   6.9920846
2024-03-10  75 72.41285 0.906070756    5787.2097  2.5871457   6.6933228
2024-03-17  77 72.42578 0.426150767   12309.0076  4.5742240  20.9235255
2024-03-24  66 72.42870 0.220575112   23782.9025  6.4286985  41.3281649
2024-03-31  72 72.42936 0.122108159   42962.0114  0.4293595   0.1843496
2024-04-07  76 72.42951 0.071079923   73804.7194  3.5704910  12.7484061
2024-04-14  80 72.42954 0.043034354  121903.5060  7.5704572  57.3118224
2024-04-21  83 72.42955 0.026896776  195043.4445 10.5704496 111.7344041
2024-04-28  82 72.42955 0.017261030  303923.9292  9.5704478  91.5934718
2024-05-05  83 72.42955 0.011328466  463084.7773 10.5704474 111.7343592
2024-05-12  94 72.42955 0.007580028  692087.1290 21.5704474 465.2841993
2024-05-19 100 72.42955 0.005158294 1017010.6833 27.5704473 760.1295665
2024-05-26  98 72.42955 0.003563058 1472342.0819 25.5704473 653.8477769
                  mape       amape
2024-02-25 0.015545890 0.007712992
2024-03-03 0.035256734 0.017944703
2024-03-10 0.034495276 0.017550340
2024-03-17 0.059405507 0.030612015
2024-03-24 0.097404523 0.046440504
2024-03-31 0.005963326 0.002972799
2024-04-07 0.046980145 0.024055129
2024-04-14 0.094630715 0.049665288
2024-04-21 0.127354814 0.068007979
2024-04-28 0.116712779 0.061972904
2024-05-05 0.127354789 0.068007964
2024-05-12 0.229472844 0.129607074
2024-05-19 0.275704473 0.159893979
2024-05-26 0.260922932 0.150035290
```

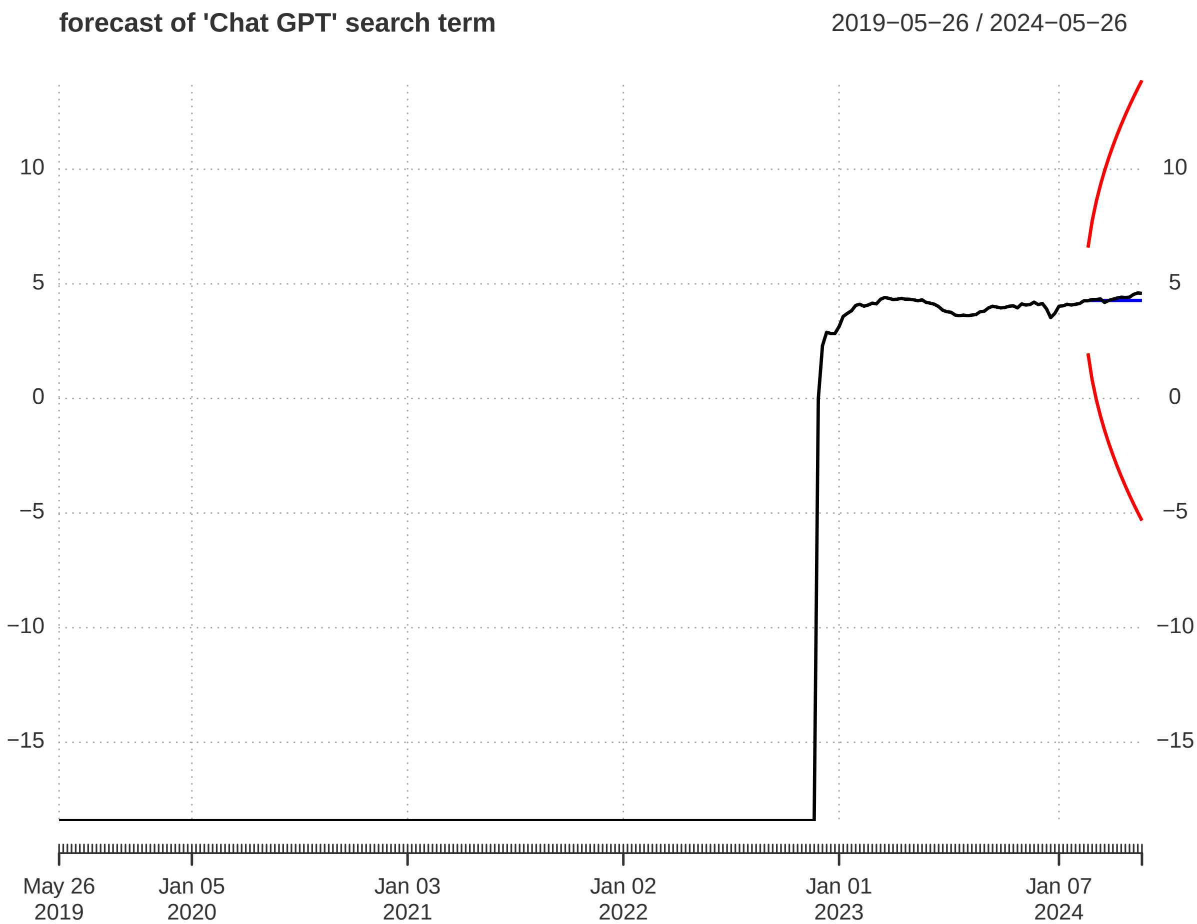
```r
ts2_perf_manual_model = colMeans(eval_data_oos[, c("mae", "mse", "mape",
↪  "amape")])
```

Automatic selection

```
# Generate out-of-sample forecasts for the length of the test set
forecast_horizon <- length(ts2_test_data)
ts2_out_of_sample_forecast <- forecast(ts2_auto_model, h =
 ↪  forecast_horizon)
ts2_forecasts_data_auto <- data.frame(f_mean  =
 ↪  as.numeric(ts2_out_of_sample_forecast$mean),
                             f_lower =
                              ↪  as.numeric(ts2_out_of_sample_forecast$lower[,
                              ↪  2]),
                             f_upper =
                              ↪  as.numeric(ts2_out_of_sample_forecast$upper[,
                              ↪  2]))
```

```
ts2_forecasts_auto_xts <- xts(ts2_forecasts_data_auto,
                     order.by = index(gpt.oos))
ts2_forecasted_data_auto <- merge(gpt2, ts2_forecasts_auto_xts)
```

```
plot(ts2_forecasted_data_auto[, c("V1", "f_mean", "f_lower",
 ↪  "f_upper")],
     major.ticks = "years",
     grid.ticks.on = "years",
     grid.ticks.lty = 3,
     main = "forecast of 'Chat GPT' search term",
     col = c("black", "blue", "red", "red"))
```




```
eval_data_oos <- tail(ts2_forecasted_data_auto, length(ts2_test_data))
eval_data_oos$V1 <- exp(eval_data_oos$V1)
eval_data_oos$f_mean <- exp(eval_data_oos$f_mean)
eval_data_oos$f_lower <- exp(eval_data_oos$f_lower)
eval_data_oos$f_upper <- exp(eval_data_oos$f_upper)
eval_data_oos$mae   <-  abs(eval_data_oos$V1 - eval_data_oos$f_mean)
eval_data_oos$mse   <-  (eval_data_oos$V1 - eval_data_oos$f_mean) ^ 2
eval_data_oos$mape  <-  abs((eval_data_oos$V1 -
↪  eval_data_oos$f_mean)/eval_data_oos$V1)
eval_data_oos$amape <-  abs((eval_data_oos$V1 -
↪  eval_data_oos$f_mean)/(eval_data_oos$V1 + eval_data_oos$f_mean))
eval_data_oos
```

```
            V1   f_mean     f_lower     f_upper         mae          mse
2024-02-25  71 72.00684 7.174120497    722.7345  1.006836196 1.013719e+00
```

```
2024-03-03  75 72.00684 2.252243337    2302.1422  2.993163804 8.959030e+00
2024-03-10  75 72.00684 0.954494376    5432.1792  2.993163804 8.959030e+00
2024-03-17  77 72.00684 0.467276408   11096.1828  4.993163804 2.493168e+01
2024-03-24  66 72.00684 0.250147593   20727.7008  6.006836196 3.608208e+01
2024-03-31  72 72.00684 0.142535920   36376.6863  0.006836196 4.673358e-05
2024-04-07  76 72.00684 0.085106877   60923.2135  3.993163804 1.594536e+01
2024-04-14  80 72.00684 0.052718327   98352.5980  7.993163804 6.389067e+01
2024-04-21  83 72.00684 0.033645282  154107.3280 10.993163804 1.208497e+02
2024-04-28  82 72.00684 0.022013706  235534.3726  9.993163804 9.986332e+01
2024-05-05  83 72.00684 0.014711248  352450.3364 10.993163804 1.208497e+02
2024-05-12  94 72.00684 0.010012491  517851.5895 21.993163804 4.836993e+02
2024-05-19 100 72.00684 0.006924373  748802.0354 27.993163804 7.836172e+02
2024-05-26  98 72.00684 0.004856957 1067537.7247 25.993163804 6.756446e+02
                   mape        amape
2024-02-25 1.418079e-02 7.040476e-03
2024-03-03 3.990885e-02 2.036071e-02
2024-03-10 3.990885e-02 2.036071e-02
2024-03-17 6.484628e-02 3.350963e-02
2024-03-24 9.101267e-02 4.352564e-02
2024-03-31 9.494717e-05 4.747133e-05
2024-04-07 5.254163e-02 2.697959e-02
2024-04-14 9.991455e-02 5.258424e-02
2024-04-21 1.324478e-01 7.092051e-02
2024-04-28 1.218679e-01 6.488779e-02
2024-05-05 1.324478e-01 7.092051e-02
2024-05-12 2.339698e-01 1.324835e-01
2024-05-19 2.799316e-01 1.627445e-01
2024-05-26 2.652364e-01 1.528948e-01
```

```r
ts2_perf_auto_model = colMeans(eval_data_oos[, c("mae", "mse", "mape",
 ↪  "amape")])
```

Comparison

```r
# Combine the performance metrics into a single data frame
ts2_combined_perf <- data.frame(
  ARIMA_1_1_1 = ts2_perf_manual_model,
  ARIMA_0_1_1 = ts2_perf_auto_model
)
```

```r
# Print the combined table
print(ts2_combined_perf)
```

```
       ARIMA_1_1_1  ARIMA_0_1_1
mae      9.59507686   9.85323646
mse    167.26598112 174.59323412
mape     0.10908605   0.11202213
amape    0.05960564   0.06137572
```

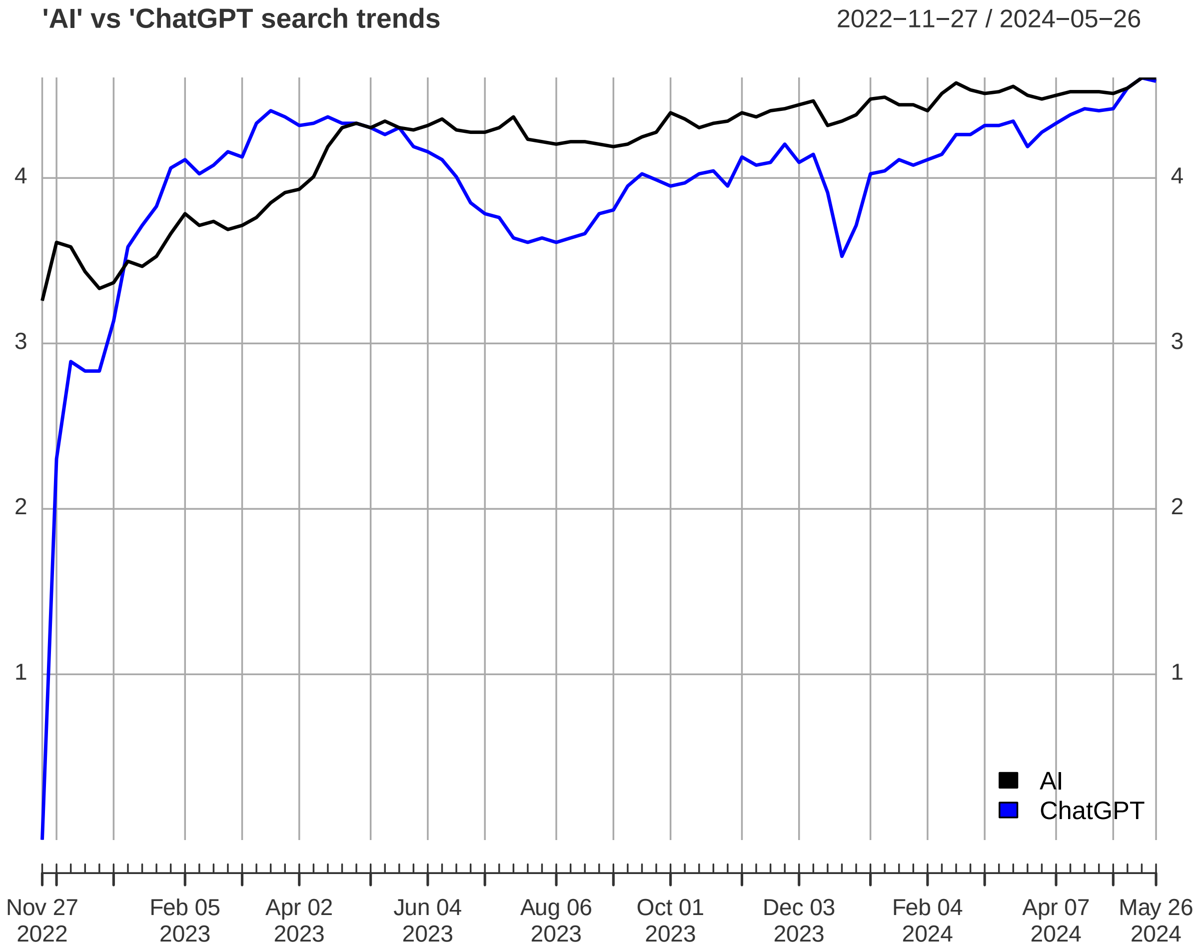
As we can see from the table, it looks like ARIMA (1,1,1) performs slightly better, so also in this case the manual model proved to be better.

TESTING COINTEGRATION

First of all, we are going to shorten the period considered up to the date when ChatGPT went out.

```r
series <- merge.xts(log_trend1_xts, log_gpt_xts)
colnames(series) <- c("AI", "ChatGPT")
```

```r
plot(series[184:262, 1:2],
     col = c("black", "blue"),
     major.ticks = "months",
     grid.ticks.on = "months",
     grid.ticks.lty = 7,
     main = "'AI' vs 'ChatGPT search trends",
     legend.loc = "bottomright")
```



The cointegrating vector is then estimated:

```
model.coint <- lm(AI ~ ChatGPT, data = series)
summary(model.coint)
```

```
Call:
lm(formula = AI ~ ChatGPT, data = series)

Residuals:
     Min       1Q   Median       3Q      Max
-0.79375 -0.10428  0.00319  0.14704  0.39216

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.94165    0.02053  191.98   <2e-16 ***
ChatGPT      0.06505    0.00132   49.27   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2196 on 260 degrees of freedom
Multiple R-squared:  0.9033,    Adjusted R-squared:  0.9029
F-statistic:  2428 on 1 and 260 DF,  p-value: < 2.2e-16
```

Testing the stationarity of the residuals:

```r
adf.test(residuals(model.coint))
```

```

    Augmented Dickey-Fuller Test

data:  residuals(model.coint)
Dickey-Fuller = -3.5227, Lag order = 6, p-value = 0.04104
alternative hypothesis: stationary
```

Since the p-value is 0.04, we reject the null about non-stationarity, so we can conclude that the two series are **cointegrated**.

Johansen cointegration test

We can also check it by performing the Johansen Test:

```r
# Convert the relevant columns to a matrix
series_matrix <- as.matrix(series[, c("ChatGPT", "AI")])

# Perform the Johansen cointegration test
johansen_test <- ca.jo(series_matrix, type = "trace", ecdet = "none", K
↪  = 5)

# Summary of the test results
summary(johansen_test)
```

```
######################
# Johansen-Procedure #
######################

Test type: trace statistic , with linear trend

Eigenvalues (lambda):
[1] 0.0874720246 0.0007934656

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 1 |  0.20  6.50  8.18 11.65
r = 0  | 23.73 15.66 17.95 23.52

Eigenvectors, normalised to first column:
(These are the cointegration relations)

           ChatGPT.l5     AI.l5
ChatGPT.l5    1.00000   1.00000
AI.l5       -12.39133 -26.45684

Weights W:
(This is the loading matrix)

            ChatGPT.l5         AI.l5
ChatGPT.d -0.040607903 -3.328168e-03
AI.d       0.004530529 -8.673299e-05
```

Null Hypothesis r=0:

- The test statistic is 23.55, which is above the 5% critical value of 14.90. This suggests that we can reject the null hypothesis of no cointegrating relationships at the 5% level, indicating the presence of one cointegrating relationship between ChatGPT and AI.

Null Hypothesis r<=1:

- The test statistic is 0.20, which is well below the 5% critical value of 8.18. This means we fail to reject the null hypothesis that there is at most 1 cointegrating relationship.

We can conclude that the two time series are cointegrated and there is only one cointegrating vector.

GRANGER CAUSALITY

In this section we want to check if either of the series is Granger caused by the other one:

```
diff_series <- diff(series)
```

```
granger_test_result <- grangertest(AI ~ ChatGPT, order = 5, data=
 ↪  diff_series)
print(granger_test_result)
```

```
Granger causality test

Model 1: AI ~ Lags(AI, 1:5) + Lags(ChatGPT, 1:5)
Model 2: AI ~ Lags(AI, 1:5)
  Res.Df Df      F    Pr(>F)
1    245
2    250 -5 11.825 2.943e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output of the Granger causality test shows that the p-value is close to 0, which is lower than the typical significance level of 0.05. This indicates that we reject the null hypothesis of no granger causality, suggesting that **AI** search **does** Granger-cause **Chat GPT** searches.

In other words, including lagged values of AI searches time series improves the predictive ability of the model over including only the ones of ChatGPT searches time series.

```
granger_test_result_reverse <- grangertest(ChatGPT ~ AI, order = 5, data
 ↪  = diff_series)
print(granger_test_result_reverse)
```

```
Granger causality test

Model 1: ChatGPT ~ Lags(ChatGPT, 1:5) + Lags(AI, 1:5)
Model 2: ChatGPT ~ Lags(ChatGPT, 1:5)
  Res.Df Df      F Pr(>F)
1    245
2    250 -5 0.2445 0.9423
```

Given the p-value higher than 0.05, we fail to reject the null hypothesis, meaning that the past values of **Chat GPT** searches **do not Granger-cause AI** searches.

VAR MODEL

Selecting lags

First of all, the optimal number of lags has to be found:

```
VARselect(series, # input data for VAR
          lag.max = 6)     # maximum lag
```

```
$selection
AIC(n)  HQ(n)  SC(n) FPE(n)
     5      3      2      5

$criteria
                 1            2            3           4           5
AIC(n) -5.283402284 -5.417711694 -5.443040351 -5.44967753 -5.48101290
HQ(n)  -5.249983755 -5.362014146 -5.365063784 -5.34942194 -5.35847830
SC(n)  -5.200312188 -5.279228200 -5.249163460 -5.20040724 -5.17634922
FPE(n)  0.005075145  0.004437333  0.004326428  0.00429794  0.00416555
                 6
AIC(n) -5.455769282
HQ(n)  -5.310955656
SC(n)  -5.095712198
FPE(n)  0.004272337
```

```
VARselect(series, lag.max = 6) %>%
  .$criteria %>%
  t() %>%
  as_tibble() %>%
  mutate(nLags = 1:nrow(.)) %>%
  select(nLags, everything()) %>%
  kbl(digits = 3) %>%
  kable_classic("striped", full_width = F)
```

Since 2,3 and 5 lags are optimal, 5 are going to be selected.

nLags	AIC(n)	HQ(n)	SC(n)	FPE(n)
1	-5.283	-5.250	-5.200	0.005
2	-5.418	-5.362	-5.279	0.004
3	-5.443	-5.365	-5.249	0.004
4	-5.450	-5.349	-5.200	0.004
5	-5.481	-5.358	-5.176	0.004
6	-5.456	-5.311	-5.096	0.004

Model

```r
series.var5 <- VAR(series, p = 5) # order of VAR model
summary(series.var5)
```

```
VAR Estimation Results:
=========================
Endogenous variables: AI, ChatGPT
Deterministic variables: const
Sample size: 257
Log Likelihood: -1.956
Roots of the characteristic polynomial:
1.003 0.9109 0.6321 0.6321 0.573 0.573 0.5611 0.5611 0.4603 0.4603
Call:
VAR(y = series, p = 5)


Estimation results for equation AI:
===================================
AI = AI.l1 + ChatGPT.l1 + AI.l2 + ChatGPT.l2 + AI.l3 + ChatGPT.l3 + AI.l4 + ChatGPT.l4 + AI.l

            Estimate Std. Error t value Pr(>|t|)
AI.l1       0.661396   0.063066  10.487  < 2e-16 ***
ChatGPT.l1  0.022964   0.002998   7.661 4.25e-13 ***
AI.l2       0.106046   0.075941   1.396 0.163849
ChatGPT.l2 -0.018539   0.004497  -4.123 5.12e-05 ***
AI.l3       0.233649   0.074797   3.124 0.002000 **
ChatGPT.l3 -0.008931   0.004652  -1.920 0.056001 .
AI.l4      -0.107777   0.072572  -1.485 0.138796
ChatGPT.l4 -0.001905   0.004668  -0.408 0.683505
AI.l5       0.052842   0.058258   0.907 0.365276
ChatGPT.l5  0.010856   0.003307   3.283 0.001176 **
```

```
const       0.234435   0.066732   3.513 0.000527 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 0.05395 on 246 degrees of freedom
Multiple R-Squared: 0.9944, Adjusted R-squared: 0.9942
F-statistic:  4376 on 10 and 246 DF,  p-value: < 2.2e-16


Estimation results for equation ChatGPT:
========================================
ChatGPT = AI.l1 + ChatGPT.l1 + AI.l2 + ChatGPT.l2 + AI.l3 + ChatGPT.l3 + AI.l4 + ChatGPT.l4 +

            Estimate Std. Error t value Pr(>|t|)
AI.l1       1.320837   1.360791   0.971    0.333
ChatGPT.l1  1.100036   0.064682  17.007   <2e-16 ***
AI.l2      -0.722181   1.638603  -0.441    0.660
ChatGPT.l2 -0.126916   0.097029  -1.308    0.192
AI.l3       0.045849   1.613917   0.028    0.977
ChatGPT.l3 -0.008105   0.100367  -0.081    0.936
AI.l4       0.812773   1.565899   0.519    0.604
ChatGPT.l4  0.017738   0.100720   0.176    0.860
AI.l5      -0.866039   1.257050  -0.689    0.492
ChatGPT.l5 -0.026689   0.071348  -0.374    0.709
const      -2.334692   1.439896  -1.621    0.106
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 1.164 on 246 degrees of freedom
Multiple R-Squared: 0.9878, Adjusted R-squared: 0.9874
F-statistic:  1999 on 10 and 246 DF,  p-value: < 2.2e-16



Covariance matrix of residuals:
             AI ChatGPT
AI      0.00291 0.01204
ChatGPT 0.01204 1.35504

Correlation matrix of residuals:
            AI ChatGPT
```

```
AI      1.0000  0.1917
ChatGPT 0.1917  1.0000
```

The roots of the characteristic polynomial are within or very close to the unit circle (1.002, 0.9122, etc.), indicating that the VAR model is stable.

Equation for AI:

- Significant lags: **AI.l1, ChatGPT.l1, ChatGPT.l2, AI.l3, ChatGPT.l5**, and **const,** indicating a good predictive power of ChatGPT on AI.
- High R^2 (0.9944) indicates that the model explains 99.44% of the variance in AI.

Equation for ChatGPT:

- Significant lags: only **ChatGPT.l1,** while the other being non significant; this indicates limited predictive power of AI on ChatGPT.

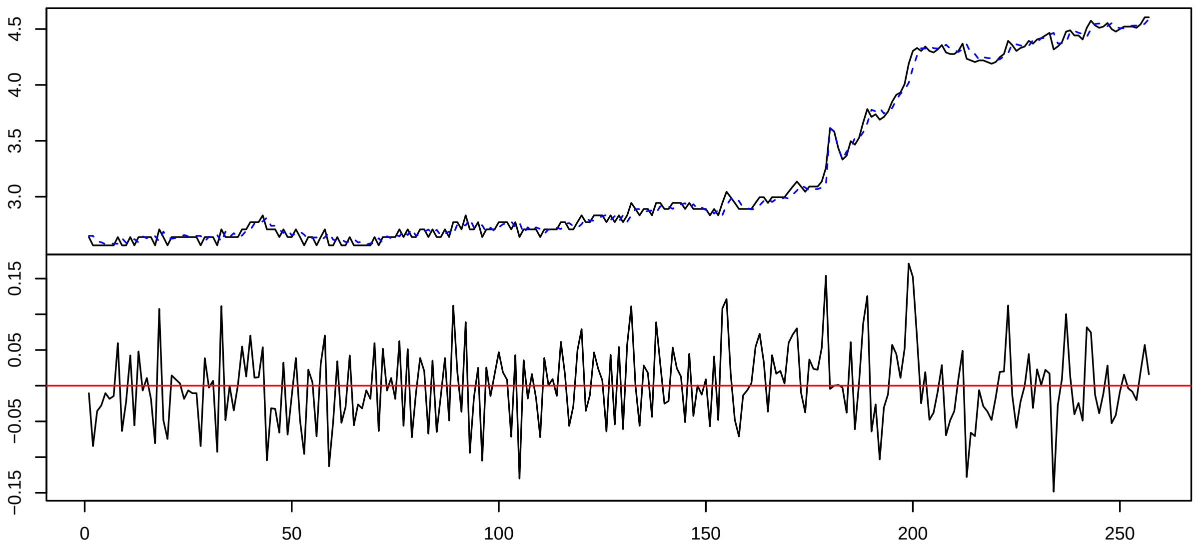
Residuals:

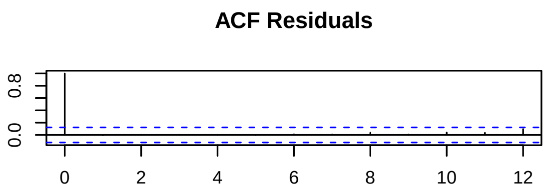
- The residuals show some correlation (0.1912), but this is not particularly strong.

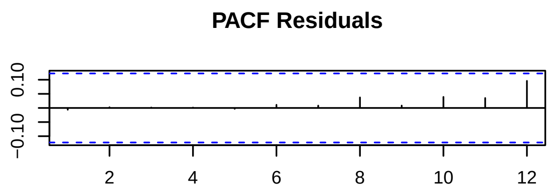
Diagnostics

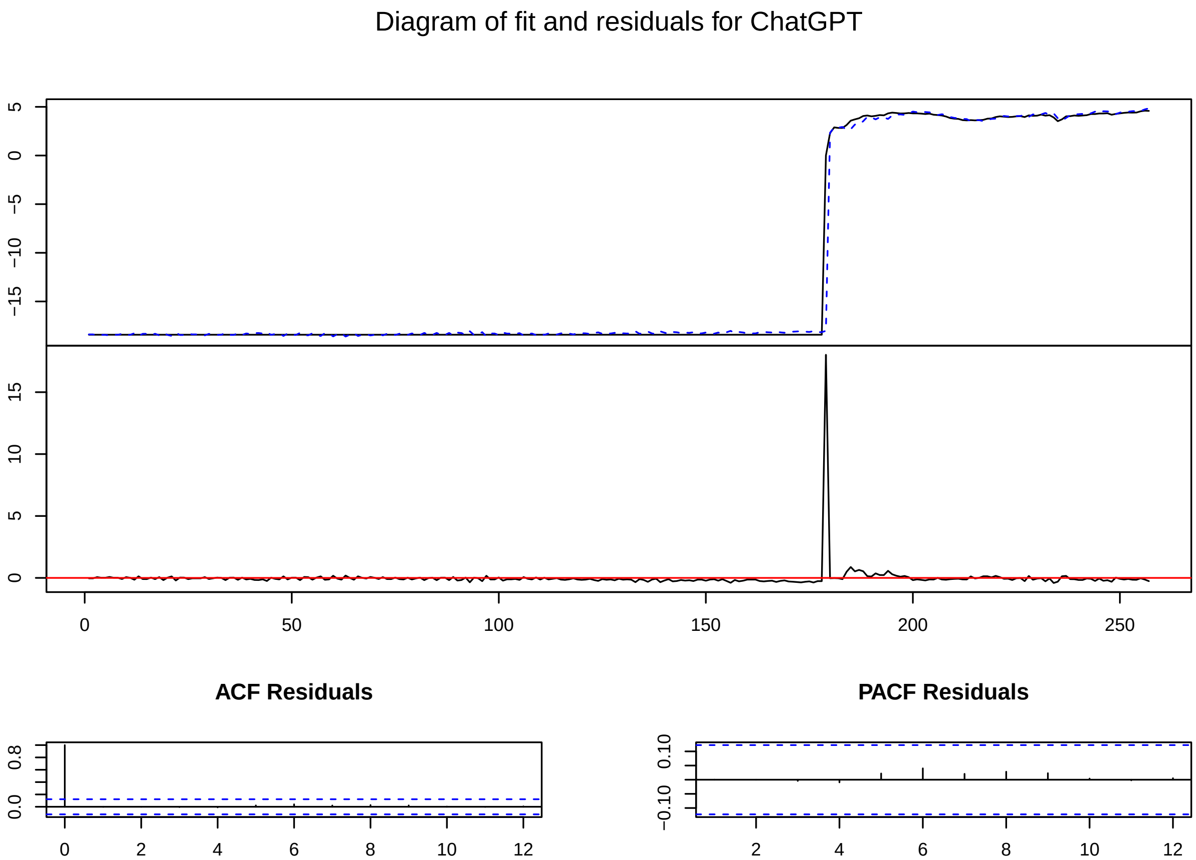
```
plot(series.var5)
```

Diagram of fit and residuals for AI









For both equations we see significant spikes at lag 0 of the ACF plot. This autocorrelation at lag 0 indicates that the current values of each variable strongly depend on their previous values, which can happen in time series data where variables exhibit persistence.

Let's check the residuals formally by using **Portmanteau test**:

```
serial.test(series.var5)
```

```

	Portmanteau Test (asymptotic)

data:  Residuals of VAR object series.var5
Chi-squared = 36.008, df = 44, p-value = 0.7988
```

Since the P-value is 0.77, we fail to reject the null of no autocorrelation.

Also with the BG test:

```r
serial.test(series.var5, type = "BG")
```

```
	Breusch-Godfrey LM test

data:  Residuals of VAR object series.var5
Chi-squared = 13.791, df = 20, p-value = 0.8409
```

Also in this case we fail to reject the null hypothesis since the p-value is > 0.05.

Information Criteria

Let's compare with the information criteria with models having the other suggested number of lags (2 and 3):

```r
series.var2 = VAR(series, p=2)
series.var3 = VAR(series, p=3)
```

```r
AIC(series.var2, series.var3, series.var5)
```

```
            df      AIC
series.var2 10 58.78234
series.var3 14 53.97275
series.var5 22 47.91246
```

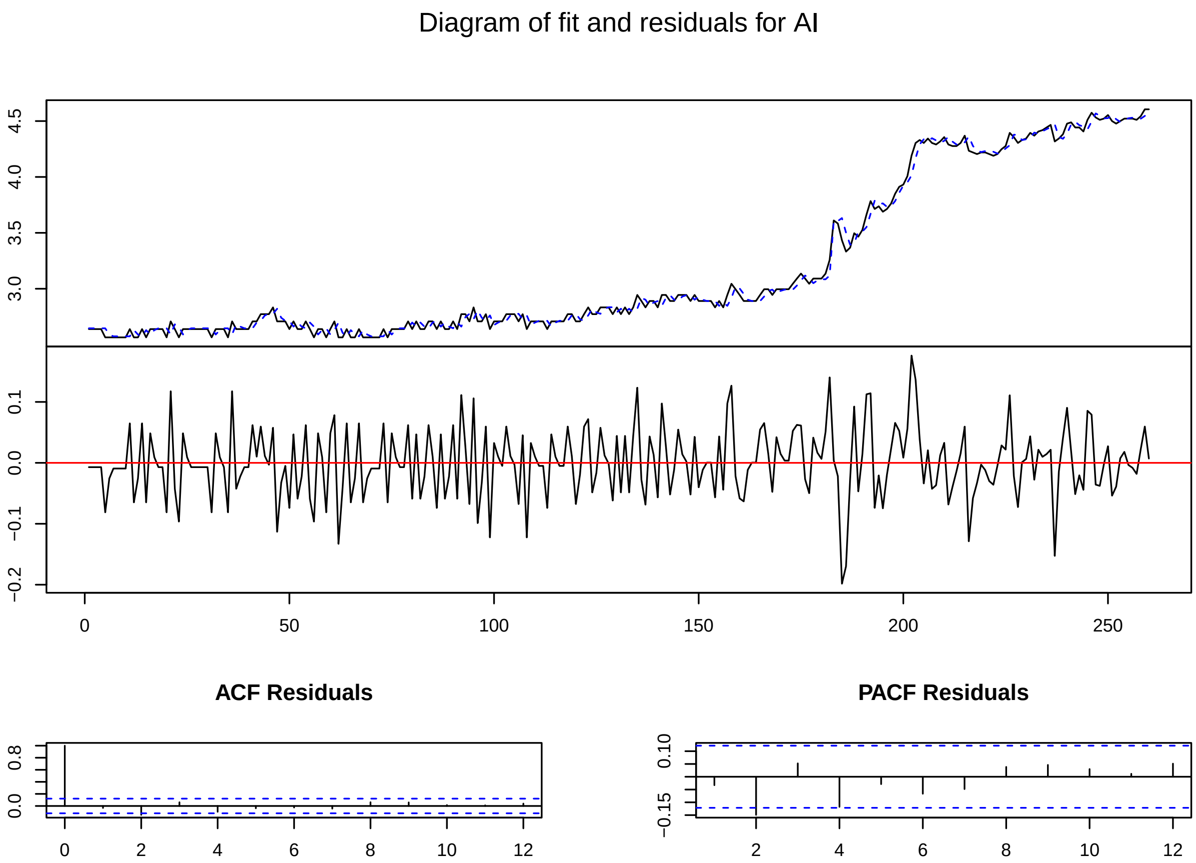
```r
BIC(series.var2, series.var3, series.var5)
```

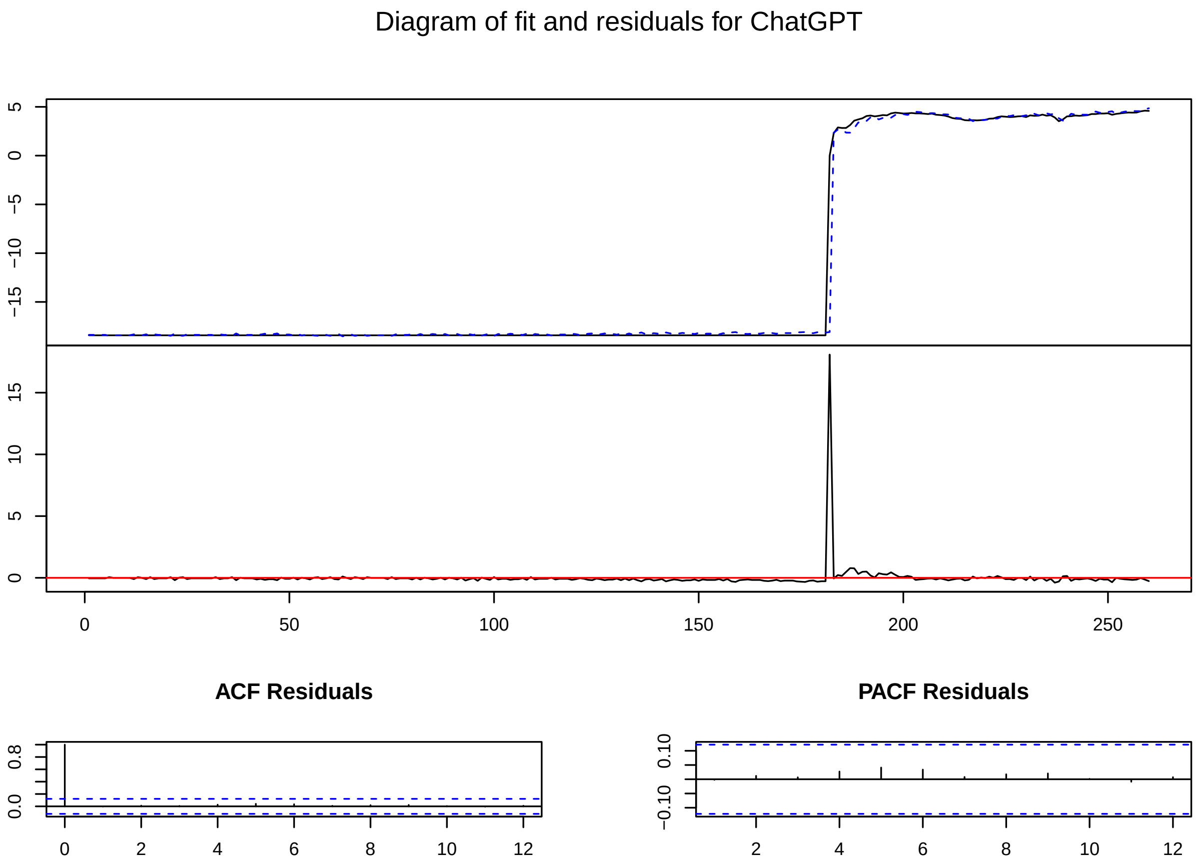
```
            df       BIC
series.var2 10  94.38916
series.var3 14 103.76834
series.var5 22 125.99213
```

AIC suggests 5 or 3 lags, BIC suggests 2 or 3 lags.

Since BIC is slightly prefered in the context of VAR, let's check the residuals of the model with 2 lags:

```r
plot(series.var2)
```





A spike at lag 2 of the PACF plot can be seen; let's test it formally:

```r
serial.test(series.var2)
```

```
    Portmanteau Test (asymptotic)

data:  Residuals of VAR object series.var2
Chi-squared = 63.769, df = 56, p-value = 0.2221
```

```r
serial.test(series.var2, type = "BG")
```

```
    Breusch-Godfrey LM test
```

```
data:  Residuals of VAR object series.var2
Chi-squared = 40.967, df = 20, p-value = 0.003762
```

As we can see from the Breusch-Godfrey test, in this case the residuals show autocorrelation, so the model with 5 lags is still prefered.

The VAR model with 3 lags is also checked:

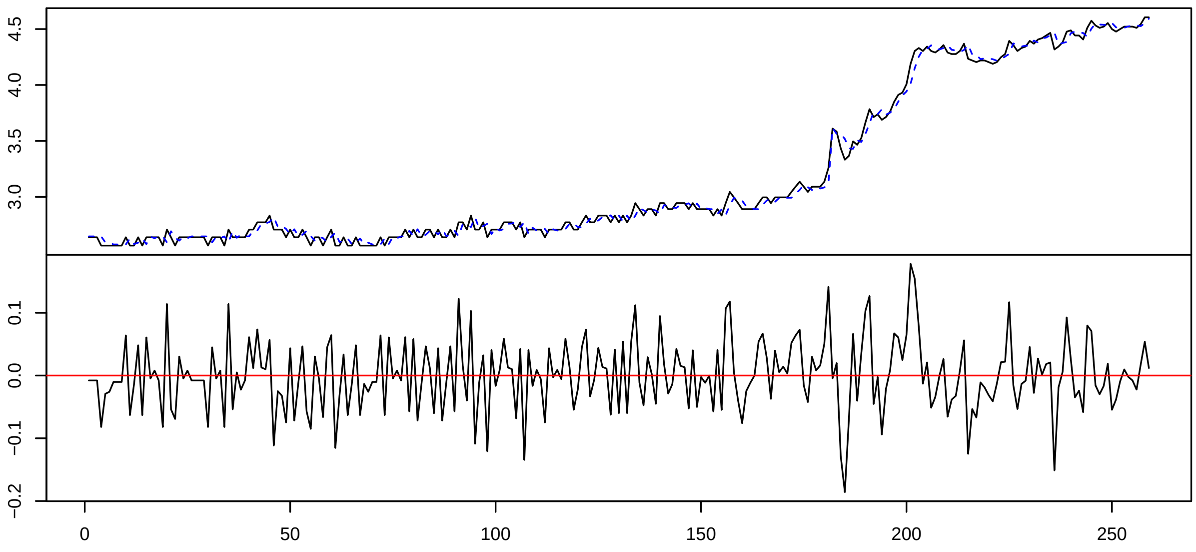
```
summary(series.var3)
```

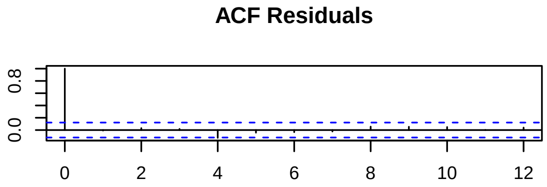
```
VAR Estimation Results:
=========================
Endogenous variables: AI, ChatGPT
Deterministic variables: const
Sample size: 259
Log Likelihood: -12.986
Roots of the characteristic polynomial:
1.003 0.9384 0.4494 0.4494 0.2324 0.09186
Call:
VAR(y = series, p = 3)


Estimation results for equation AI:
===================================
AI = AI.l1 + ChatGPT.l1 + AI.l2 + ChatGPT.l2 + AI.l3 + ChatGPT.l3 + const

             Estimate Std. Error t value Pr(>|t|)
AI.l1       0.7127668  0.0621562  11.467  < 2e-16 ***
ChatGPT.l1  0.0222271  0.0030903   7.193 7.24e-12 ***
AI.l2       0.0433975  0.0740655   0.586 0.558444
ChatGPT.l2 -0.0200309  0.0046093  -4.346 2.01e-05 ***
AI.l3       0.2126966  0.0586075   3.629 0.000344 ***
ChatGPT.l3  0.0005411  0.0033523   0.161 0.871892
const       0.1404485  0.0657533   2.136 0.033644 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 0.05573 on 252 degrees of freedom
Multiple R-Squared: 0.9939, Adjusted R-squared: 0.9938
F-statistic:  6863 on 6 and 252 DF,  p-value: < 2.2e-16
```

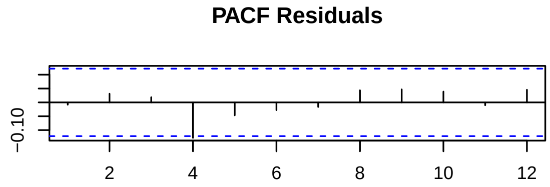
```
Estimation results for equation ChatGPT:
========================================
ChatGPT = AI.l1 + ChatGPT.l1 + AI.l2 + ChatGPT.l2 + AI.l3 + ChatGPT.l3 + const

           Estimate Std. Error t value Pr(>|t|)
AI.l1       1.18011    1.28485   0.918    0.359
ChatGPT.l1  1.10196    0.06388  17.250   <2e-16 ***
AI.l2      -0.77895    1.53103  -0.509    0.611
ChatGPT.l2 -0.12215    0.09528  -1.282    0.201
AI.l3       0.13245    1.21149   0.109    0.913
ChatGPT.l3 -0.01875    0.06930  -0.271    0.787
const      -2.08162    1.35920  -1.532    0.127
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 1.152 on 252 degrees of freedom
Multiple R-Squared: 0.9878, Adjusted R-squared: 0.9876
F-statistic:  3414 on 6 and 252 DF,  p-value: < 2.2e-16



Covariance matrix of residuals:
              AI ChatGPT
AI      0.003106 0.01093
ChatGPT 0.010927 1.32724

Correlation matrix of residuals:
            AI ChatGPT
AI      1.0000  0.1702
ChatGPT 0.1702  1.0000
```

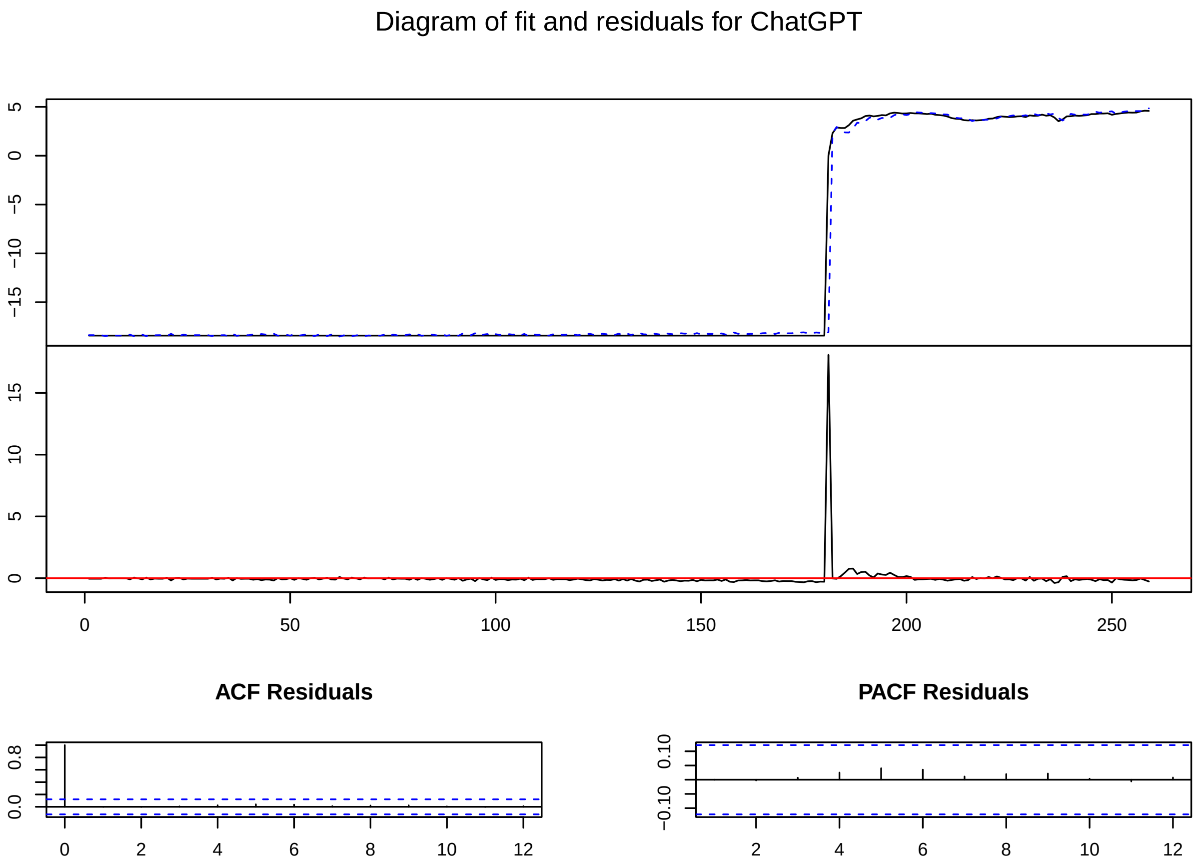
```r
plot(series.var3)
```

Diagram of fit and residuals for AI









Also in this case a significant spike at lag 4 of the PACF plot is detected.

Let's use the tests:

```r
serial.test(series.var3)
```

```
    Portmanteau Test (asymptotic)

data:  Residuals of VAR object series.var3
Chi-squared = 54.782, df = 52, p-value = 0.3695
```

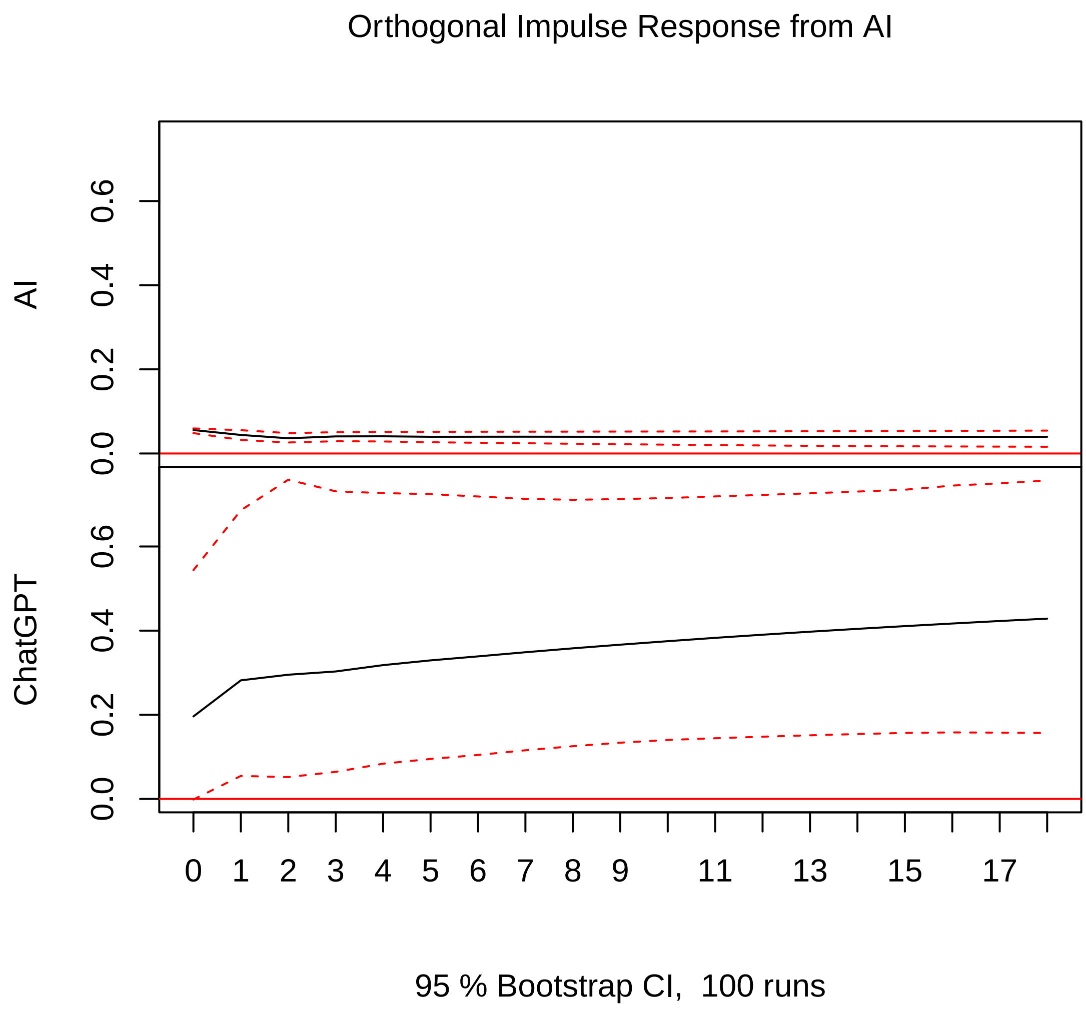
```r
serial.test(series.var3, type = "BG")
```

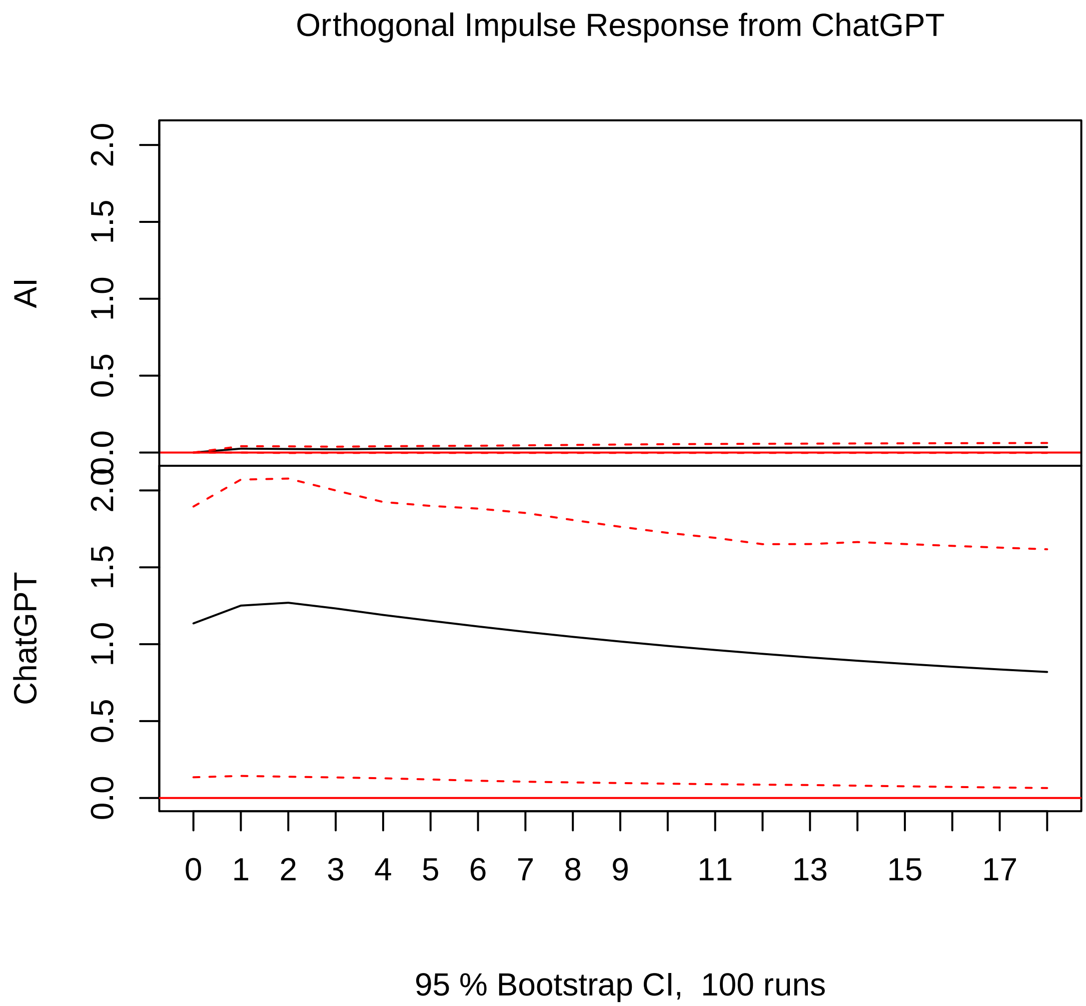
```
    Breusch-Godfrey LM test

data:  Residuals of VAR object series.var3
Chi-squared = 30.073, df = 20, p-value = 0.06868
```

At 5% significance level, there is no evidence of autocorrelation in the residuals, so the VAR model with 3 lags is going to be selected because of its lower BIC and also less parameters.

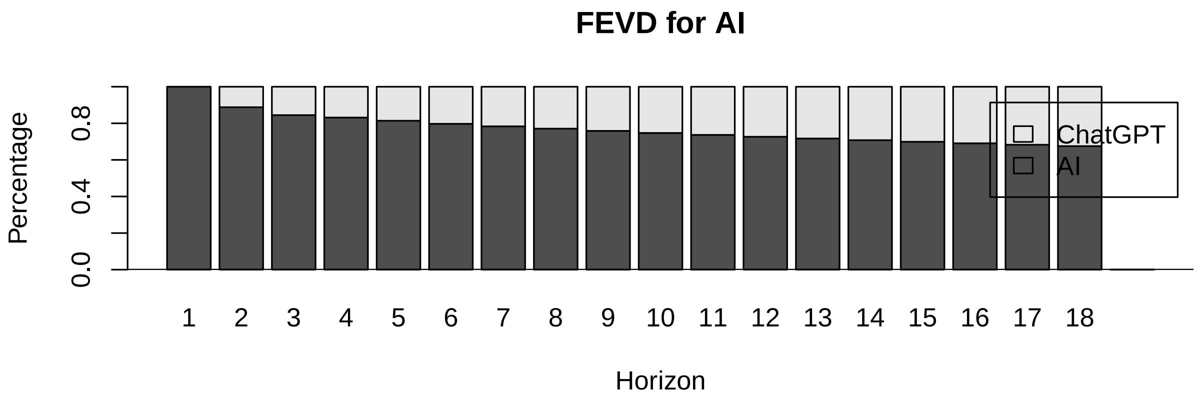
Impulse response functions

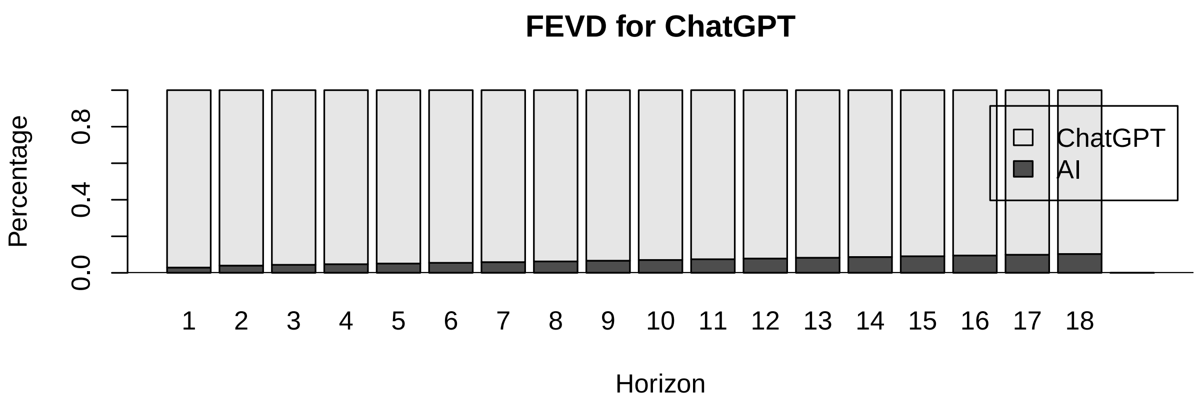
```
plot(irf(series.var3, n.ahead = 18))
```






```r
plot(fevd(series.var3, n.ahead = 18))
```





Impulse response function for AI:

- **Impact on AI:** When there is a sudden spike in searches for "AI" (e.g., a significant news event related to AI), searches for "AI" itself see an immediate positive effect. This means that interest in "AI" increases sharply in response to relevant events, but the effect stabilizes quickly, indicating that the spike in interest doesn't last long.
- **Impact on ChatGPT:** A sudden spike in searches for "AI" gradually influences searches for "ChatGPT". The impact starts small and grows over time, suggesting that as people search for "AI", they increasingly also search for "ChatGPT". This could indicate that interest in AI-related developments or news gradually leads to more curiosity and searches about specific AI applications like ChatGPT.

Impulse response function from ChatGPT:

- **Impact on ChatGPT:** When there is a sudden spike in searches for "ChatGPT" (e.g., a significant news event or release related to ChatGPT), searches for "ChatGPT" itself increase sharply but this effect gradually declines. This suggests that interest in "ChatGPT" spikes in response to specific events but fades over time as the initial excitement wears off.

- **Impact on AI:** A sudden spike in searches for "ChatGPT" has a negligible effect on searches for "AI". This indicates that increased interest in ChatGPT does not lead to a significant increase in interest in the broader topic of AI. People searching for ChatGPT may already be aware of AI or may not feel the need to search for AI specifically.

Interpretation of FEVD:

FEVD for AI:

- The forecast error variance decomposition for "AI" shows that most of the variability in searches for "AI" can be explained by its own past search behavior. Over time, a small portion of the variability can be explained by searches for "ChatGPT", indicating that interest in "AI" is primarily self-driven with some influence from the increasing interest in "ChatGPT".

FEVD for ChatGPT:

- The forecast error variance decomposition for "ChatGPT" indicates that almost all of the variability in searches for "ChatGPT" is explained by its own past search behavior, with very little influence from searches for "AI". This suggests that interest in "ChatGPT" is largely self-sustained and independent of the broader interest in AI.

VAR FORECASTING

Let's now evaluate the forecasting power of the model:

```
series.var3.forecast <- predict(series.var3,
                                n.ahead = 14,
                                ci = 0.95) # 95% confidence interval
```

Results are stored in a data frame:

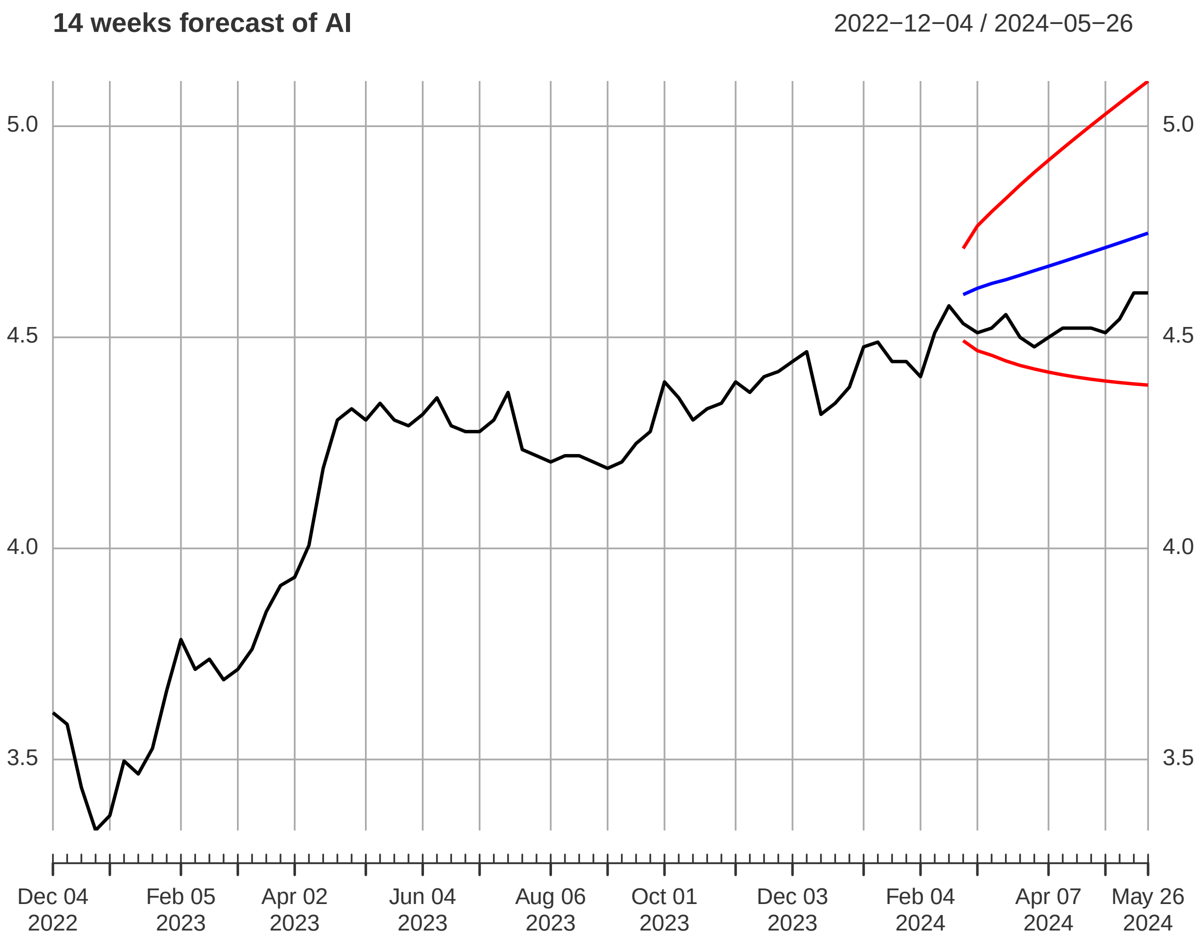
```
gpt_var_forecast <- xts(series.var3.forecast$fcst$ChatGPT[,-4],
                    # we exclude the last column with CI
                    tail(index(series), 14))
ai_var_forecast <- xts(series.var3.forecast$fcst$AI[,-4],
                    # we exclude the last column with CI
                    tail(index(series), 14))
```

```
names(gpt_var_forecast) <- c("gpt_fore", "gpt_lower", "gpt_upper")
names(ai_var_forecast) <- c("ai_fore", "ai_lower", "ai_upper")
```

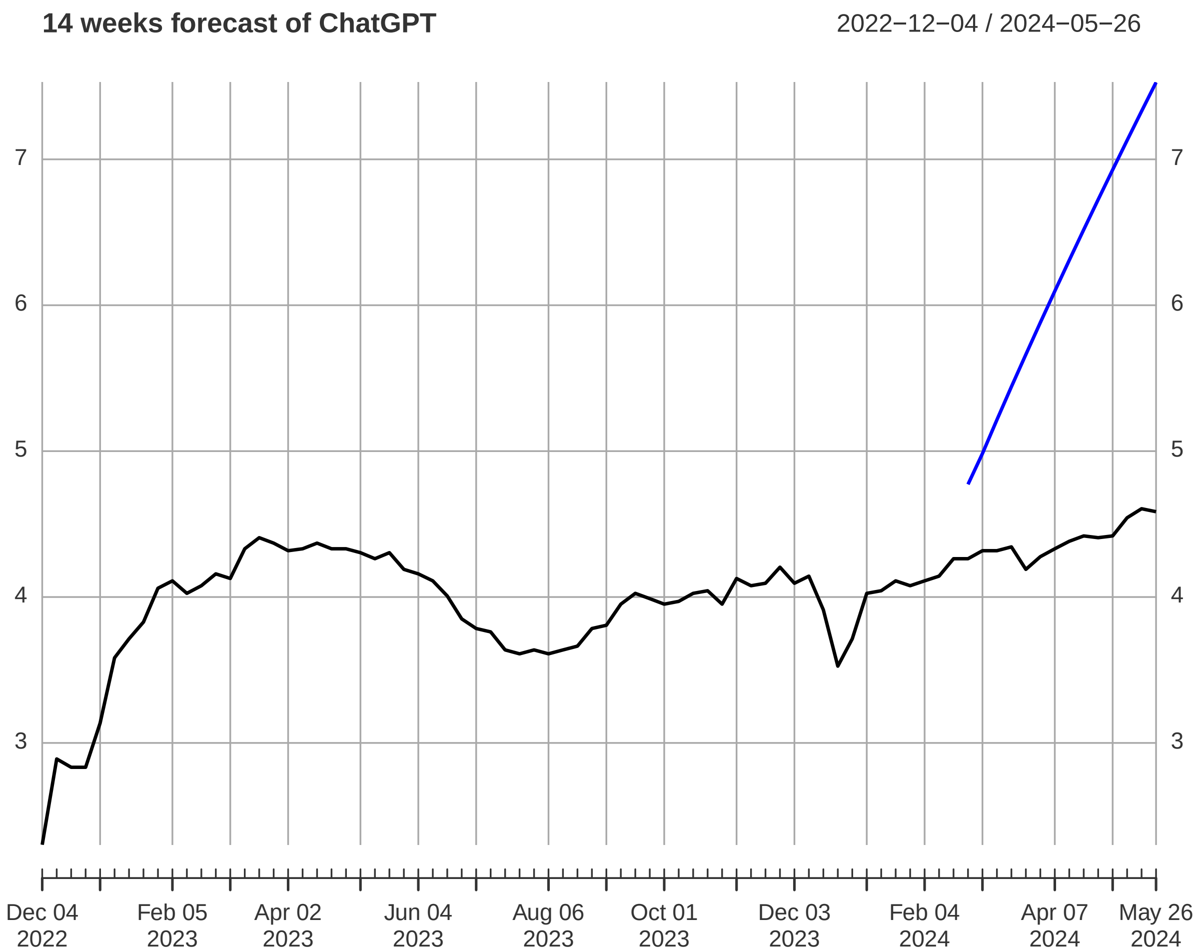
```
ai_gpt_forecasts <- merge(series,
                 gpt_var_forecast,
                 ai_var_forecast)
```

Let's now inspect them visually:

```
plot(ai_gpt_forecasts["2022-12-04/", c("AI", "ai_fore",
                        "ai_lower", "ai_upper")],
     major.ticks = "months",
     grid.ticks.on = "months",
     grid.ticks.lty = 7,
     main = "14 weeks forecast of AI",
     col = c("black", "blue", "red", "red"))
```




```
plot(ai_gpt_forecasts["2022-12-04/", c("ChatGPT", "gpt_fore")],
     major.ticks = "months",
     grid.ticks.on = "months",
     grid.ticks.lty = 7,
     main = "14 weeks forecast of ChatGPT",
     col = c("black", "blue"))
```



From the plots it is already pretty clear that the VAR model is performing worse than the ARIMA alternatives, but let's compare the error metrics anyway.

```
ai_gpt_forecasts$AI <- exp(ai_gpt_forecasts$AI)
ai_gpt_forecasts$ai_fore <- exp(ai_gpt_forecasts$ai_fore)
ai_gpt_forecasts$ai_lower <- exp(ai_gpt_forecasts$ai_lower)
ai_gpt_forecasts$ai_upper <- exp(ai_gpt_forecasts$ai_upper)
ai_gpt_forecasts$mae.ai   <-  abs(ai_gpt_forecasts$AI -
↪   ai_gpt_forecasts$ai_fore)
ai_gpt_forecasts$mse.ai   <-  (ai_gpt_forecasts$AI -
↪   ai_gpt_forecasts$ai_fore) ^ 2
ai_gpt_forecasts$mape.ai  <-  abs((ai_gpt_forecasts$AI -
↪   ai_gpt_forecasts$ai_fore)/ai_gpt_forecasts$AI)
```

```r
ai_gpt_forecasts$amape.ai <-  abs((ai_gpt_forecasts$AI -
↪  ai_gpt_forecasts$ai_fore) / (ai_gpt_forecasts$AI +
↪  ai_gpt_forecasts$ai_fore))


ai_gpt_forecasts$ChatGPT <- exp(ai_gpt_forecasts$ChatGPT)
ai_gpt_forecasts$gpt_fore <- exp(ai_gpt_forecasts$gpt_fore)
ai_gpt_forecasts$gpt_lower <- exp(ai_gpt_forecasts$gpt_lower)
ai_gpt_forecasts$gpt_upper <- exp(ai_gpt_forecasts$gpt_upper)
ai_gpt_forecasts$mae.gpt   <-  abs(ai_gpt_forecasts$ChatGPT -
↪  ai_gpt_forecasts$gpt_fore)
ai_gpt_forecasts$mse.gpt   <-  (ai_gpt_forecasts$ChatGPT -
↪  ai_gpt_forecasts$gpt_fore) ^ 2
ai_gpt_forecasts$mape.gpt  <-  abs((ai_gpt_forecasts$ChatGPT -
↪  ai_gpt_forecasts$gpt_fore)/ai_gpt_forecasts$ChatGPT)
ai_gpt_forecasts$amape.gpt <-  abs((ai_gpt_forecasts$ChatGPT -
↪  ai_gpt_forecasts$gpt_fore) / (ai_gpt_forecasts$ChatGPT +
↪  ai_gpt_forecasts$gpt_fore))
```

```r
var_metrics = colMeans(ai_gpt_forecasts[,9:16], na.rm = TRUE)
```

FINAL COMPARISON

Let's now compare the models' forecasting power, by starting with **ChatGPT** time series:

```r
ts2_combined_perf$VAR <- (var_metrics[5:8])
ts2_combined_perf
```

```
        ARIMA_1_1_1  ARIMA_0_1_1          VAR
mae      9.59507686   9.85323646 5.962454e+02
mse    167.26598112 174.59323412 6.274502e+05
mape     0.10908605   0.11202213 6.799866e+00
amape    0.05960564   0.06137572 6.660865e-01
```

- The table clearly shows that ARIMA(1,1,1) is the best at forecasting future values compared to the others.
- VAR model has substantially higher errors across all metrics, suggesting that it is not well-suited for the given data.

Let's also examine **AI** time series performances table:

```
combined_perf$VAR <- (var_metrics[1:4])
combined_perf
```

```
      ARIMA_1_1_2 ARIMA_0_1_8          VAR
mae    3.65165914  4.53356390  14.43073561
mse   19.48744607 31.70184775 224.24582515
mape   0.04219822  0.05237733   0.15589074
amape  0.02062693  0.02519301   0.07190771
```

- The table shows that ARIMA(1,1,2) is the best model at forecasting future values compared to the others.
- Also in this case, VAR model has higher errors across all metrics, suggesting that it is not well-suited for the given data.

Justification and Motivation for Results

1. **ARIMA Model Strengths:**
   - **Univariate Nature:** ARIMA models are designed for univariate time series data, making them ideal when the primary goal is to predict the future values of a single series without considering cross-series interactions.
   - **Strong Performance on Trend Data:** Google Trends data often exhibit clear patterns and trends, which ARIMA models can effectively capture and extrapolate.
2. **VAR Model Weaknesses:**
   - **Complexity and Data Requirements:** VAR models are more complex and require more data to accurately estimate the relationships between multiple time series. If these relationships are weak or not properly captured, the model's performance can degrade significantly.
   - **Overfitting Risks:** With more parameters to estimate, VAR models are prone to overfitting, especially if the underlying data does not strongly support the assumed interdependencies.

Final conclusions

This project demonstrates the importance of model selection and validation in time series forecasting. While ARIMA models have shown strong performance with Google Trends data, the use of VAR models requires careful consideration of data characteristics and model specifications. The results obtained justify prioritizing ARIMA models for similar forecasting tasks, ensuring reliable and accurate predictions.