

ASML Stock Price Analysis

Project of Time Series Analysis for Economic and Financial Data

Daniele Tambone

2026-01-14

Table of contents

Preface	3
Data Source	3
Data Loading	3
1 Exploratory Data Analysis	5
1.1 Adjusted closing prices	5
1.2 Log returns	7
1.3 Distribution analysis	10
1.3.1 Graphical representation	10
1.3.2 Synthetic indicators	15
1.3.3 Test of normality	17
2 Return Analysis	20
2.1 ACF and PACF Analysis	20
2.2 Models	23
2.2.1 AR(1)	23
2.2.2 MA(1)	25
2.2.3 ARMA(1,1)	26
2.2.4 AR(2)	27
2.2.5 MA(2)	28
2.2.6 ARMA(1,2)	29
2.2.7 ARMA(2,1)	31
2.2.8 ARMA(2,2)	32
2.2.9 AR(3)	33
2.2.10 MA(3)	34
2.2.11 ARMA(1,3)	36
2.2.12 ARMA(2,3)	37
2.2.13 ARMA(3,1)	38
2.2.14 ARMA(3,2)	40
2.2.15 ARMA(3,3)	41
2.3 AIC vs BIC	42
2.4 Forecast	45
2.5 Conclusion	51
3 Volatility analysis	52

Preface

[GitHub Repository](#)

[View Slides](#)

[Download PDF](#)

This project aims to conduct an in-depth time series analysis of the stock prices for **ASML Holding NV (ASML)**, a global leader in the manufacturing of lithography systems for the semiconductor industry. The primary objective is to explore the dynamics of historical prices, volatility, and underlying trends of the stock.

Data Source

We retrieve historical daily data using the `quantmod` R package, focusing on the **Adjusted Closing Price** to account for dividends and stock splits.

- **Ticker:** ASML
- **Source:** [Yahoo Finance](#)
- **Start Date:** 2000-01-01
- **End Date:** 2025-12-31

Data Loading

The dataset structure is shown below:

```
datatable(
  tail(asml, 100),
  rownames = FALSE,
  options = list(
    pageLength = 10,
    order = list(list(0, 'desc'))
)
)
```

Show entries Search:

Date	ASML.Open	ASML.High	ASML.Low	ASML.Close	ASML.Volume	ASML.Adjusted
2025-12-30	1084.579956054688	1086.06005859375	1070.420043945312	1072.140014648438	807300	1072.140014648438
2025-12-29	1064.579956054688	1073.650024414062	1061.069946289062	1066	424300	1066
2025-12-26	1066.25	1076.089965820312	1063.079956054688	1072.75	363800	1072.75
2025-12-24	1059.819946289062	1066.800048828125	1057.43994140625	1065.52001953125	229800	1065.52001953125
2025-12-23	1057.93994140625	1064.75	1055.68994140625	1061.839965820312	543600	1061.839965820312
2025-12-22	1066	1067.160034179688	1050	1056.97998046875	703700	1056.97998046875
2025-12-19	1042.56005859375	1062.680053710938	1042.56005859375	1056.02001953125	2319600	1056.02001953125
2025-12-18	1047.969970703125	1051.81005859375	1035.150024414062	1036.31005859375	1645400	1036.31005859375
2025-12-17	1060.77001953125	1065.119995117188	1010.010009765625	1015.429992675781	2140900	1015.429992675781
2025-12-16	1081.530029296875	1088.06005859375	1065.93994140625	1076.050048828125	956300	1076.050048828125

Showing 1 to 10 of 100 entries Previous ... Next

1 Exploratory Data Analysis

1.1 Adjusted closing prices

Let us plot the adjusted closing prices...

```
fig_asml <- plot_ly(data = asml, x = ~Date, y = ~ASML.Adjusted,
  type = 'scatter', mode = 'lines', name = 'ASML Adjusted',
  line = list(color = 'darkblue', width = 1.5))

fig_asml <- layout(
  fig_asml,
  title = "ASML Stock Price Evolution",
  xaxis = list(
    title = "Date",
    rangeslider = list(visible = TRUE),
    rangeselector = list(
      buttons = list(
        list(count=1, label="1m", step="month", stepmode="backward"),
        list(count=6, label="6m", step="month", stepmode="backward"),
        list(count=1, label="YTD", step="year", stepmode="todate"),
        list(count=1, label="1y", step="year", stepmode="backward"),
        list(step="all")
      )
    )
  )
)

fig_asml
```

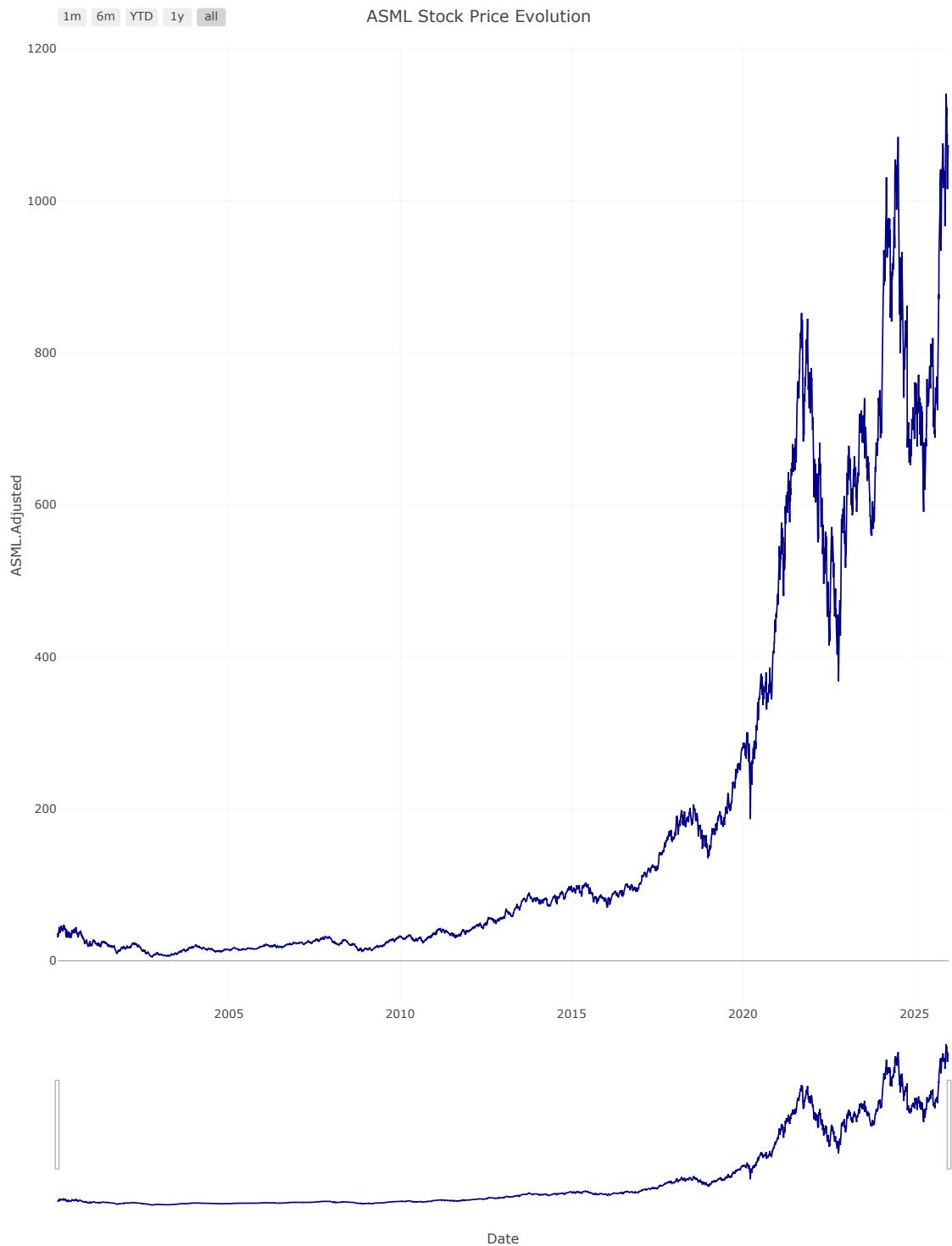


Figure 1.1: ASML Daily Adjusted Closing Prices

The graph clearly shows that the price series is **non-stationary**: the mean is not constant (it exhibits trends), and the variance appears to fluctuate depending on the price level. Consequently, standard statistical inference cannot be directly applied to prices.

1.2 Log returns

To obtain a stationary process, we transform prices into **log-returns**. Let P_t be the price at time t . The simple net return is defined as:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

However, in financial econometrics, **log-returns** (r_t) are preferred due to their time-additivity property. The log-return is defined as the natural logarithm of the gross return:

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) = \ln(P_t) - \ln(P_{t-1})$$

We compute the log-returns for ASML in R:

```
log_ret_vec <- diff(log(asml$ASML.Adjusted))
log_ret_vec <- log_ret_vec[is.finite(log_ret_vec)]

log_returns <- data.frame(
  Date = asml>Date[-1],
  LogReturns = as.numeric(log_ret_vec)
)
```

Let's visualize the log-returns of ASML:

```
fig_log <- plot_ly(data = log_returns, x = ~Date, y = ~LogReturns, type = 'scatter',
  mode = 'lines', name = 'Log Returns', line = list(color = 'darkred', width = 1))

fig_log <- layout(
  fig_log,
  title = "ASML Log>Returns",
  xaxis = list(title = "Date"),
  yaxis = list(title = "Log Return"),
  shapes = list(
    list(
      type = "line",
      x0 = min(log_returns>Date),
      x1 = max(log_returns>Date),
      y0 = 0,
```

```
    y1 = 0,
    line = list(color = "black", width = 1)
)
)
)

fig_log
```

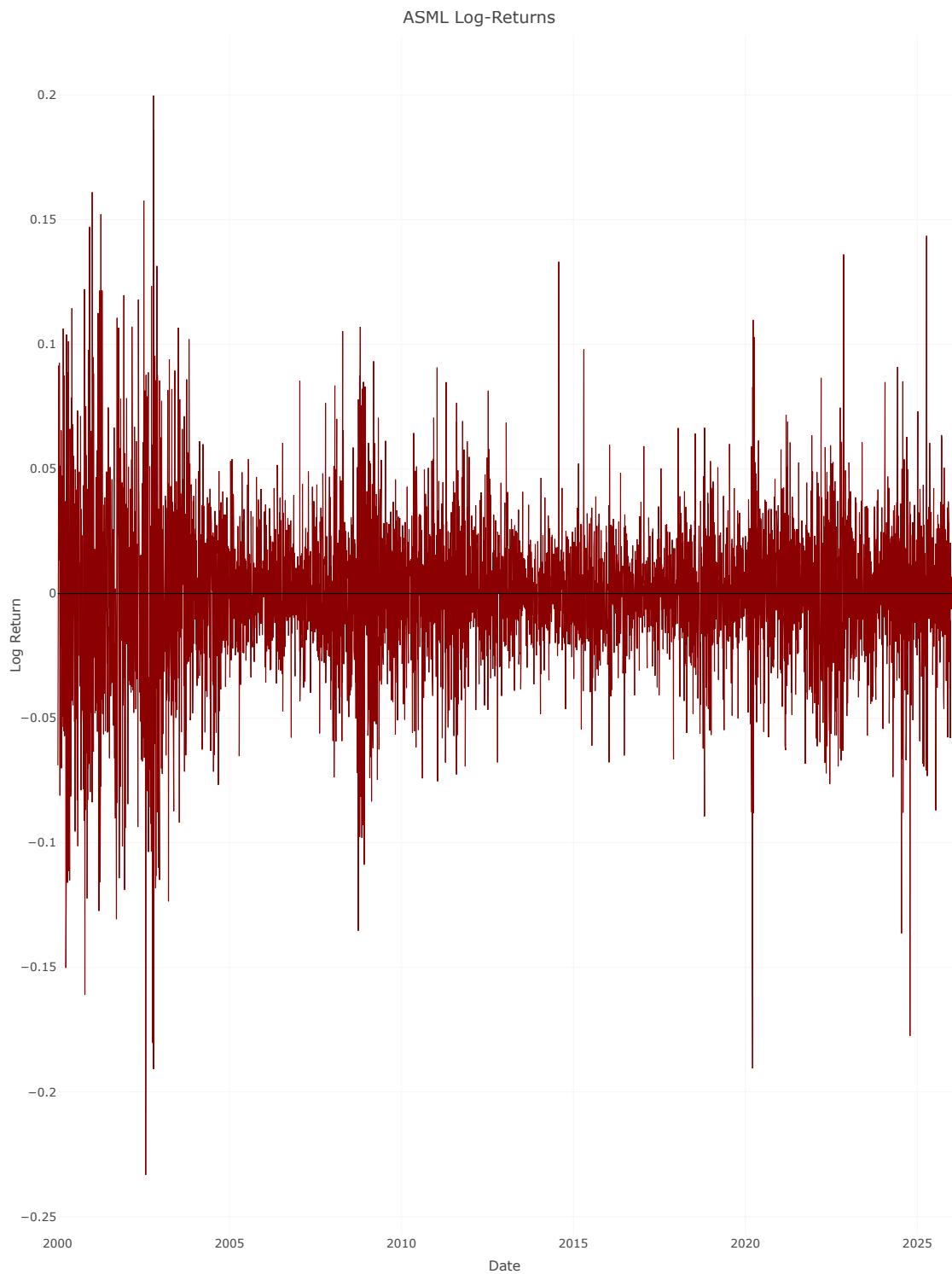


Figure 1.2: ASML Daily Log-Returns

Unlike prices, the log-returns oscillate around a constant mean (close to zero). This behavior suggests that the return series is **stationary**, satisfying the conditions for further econometric analysis.

1.3 Distribution analysis

We analyze the distribution of log-returns to check for deviations from normality (e.g., fat tails or asymmetry).

1.3.1 Graphical representation

1.3.1.1 Histogram

The graphical representation using a histogram of the frequency distribution of returns observed over a given sample period provides an initial indication of the characteristics of the probability distribution that generated them.

```
mu_ret <- mean(log_returns$LogReturns)
sd_ret <- sd(log_returns$LogReturns)

t_fit_fGarch <- stdFit(log_returns$LogReturns * 100)
t_params <- t_fit_fGarch$par

fig_dist <- plot_ly(data = log_returns, x = ~LogReturns,
  type = "histogram", name = "Log Returns", histnorm = "probability density",
  marker = list(color = "lightgray", line = list(color = "gray", width = 1)),
  opacity = 0.7)

x_seq <- seq(min(log_returns$LogReturns), max(log_returns$LogReturns), length.out = 500)

y_norm <- dnorm(x_seq, mean = mu_ret, sd = sd_ret)
y_t <- dstd(
  x_seq,
  mean = t_params["mean"]/100,
  sd = t_params["sd"]/100,
  nu = t_params["nu"])
)

fig_dist <- fig_dist %>%
  add_lines(
    x = x_seq,
    y = y_norm,
    name = "Normal Distribution",
    line = list(color = "#E74C3C", width = 2, dash = "dash"),
```

```
    inherit = FALSE
) %>%

add_lines(
  x = x_seq,
  y = y_t,
  name = "Student-t",
  line = list(color = "#2E86C1", width = 2),
  inherit = FALSE
) %>%

layout(
  title = "Distribution of ASML Log-Returns",
  xaxis = list(title = "Log Return"),
  yaxis = list(title = "Density"),
  legend = list(x = 0.8, y = 0.9),
  hovermode = "x unified"
)

fig_dist
```

Distribution of ASML Log-Returns

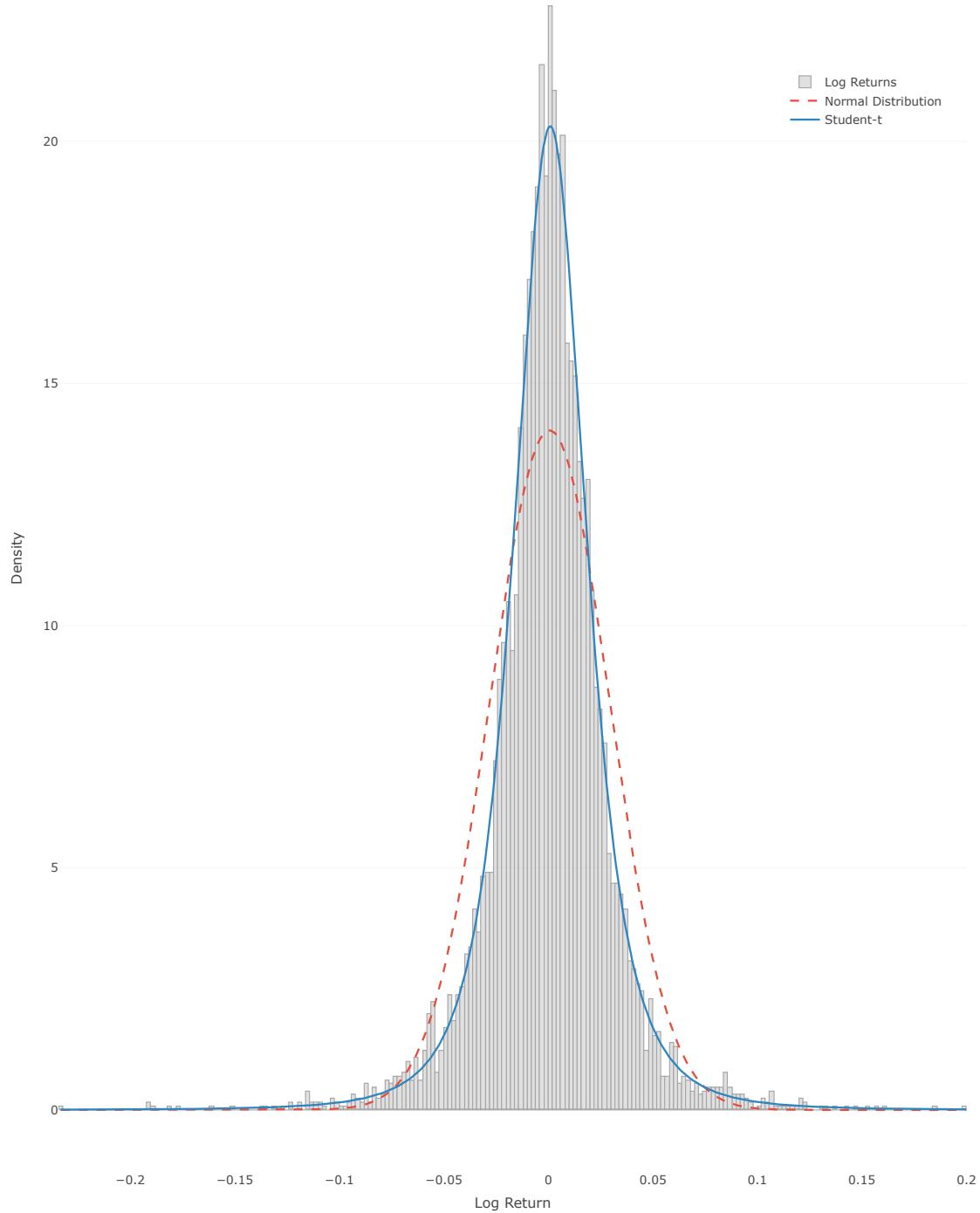


Figure 1.3: Histogram of ASML Log-Returns

The histogram indicates that the distribution of log-returns is closer to a Student-t distribution than to a Normal distribution.

1.3.1.2 Q-Q plot

```
vec_ret <- log_returns$LogReturns

qq_vals <- qnorm(vec_ret, plot.it = FALSE)
qq_data <- data.frame(
  Theoretical = qq_vals$x,
  Sample = qq_vals$y
)

y <- quantile(vec_ret, c(0.25, 0.75), names = FALSE)
x <- qnorm(c(0.25, 0.75))
slope <- diff(y)/diff(x)
int <- y[1] - slope * x[1]

fig_qq <- plot_ly(data = qq_data, x = ~Theoretical, y = ~Sample, type = 'scatter',
  mode = 'markers', marker = list(size = 3, color = '#2E86C1', opacity = 0.6),
  name = "Returns")

fig_qq <- fig_qq %>%
  add_lines(
    x = ~Theoretical,
    y = ~Theoretical * slope + int,
    line = list(color = "#E74C3C", width = 2, dash = "dash"),
    name = "Normal Reference",
    inherit = FALSE
  ) %>%

  layout(
    title = "Q-Q Plot: Normal vs Empirical",
    xaxis = list(title = "Theoretical Quantiles (Normal)"),
    yaxis = list(title = "Sample Quantiles (ASML)")
  )

fig_qq
```

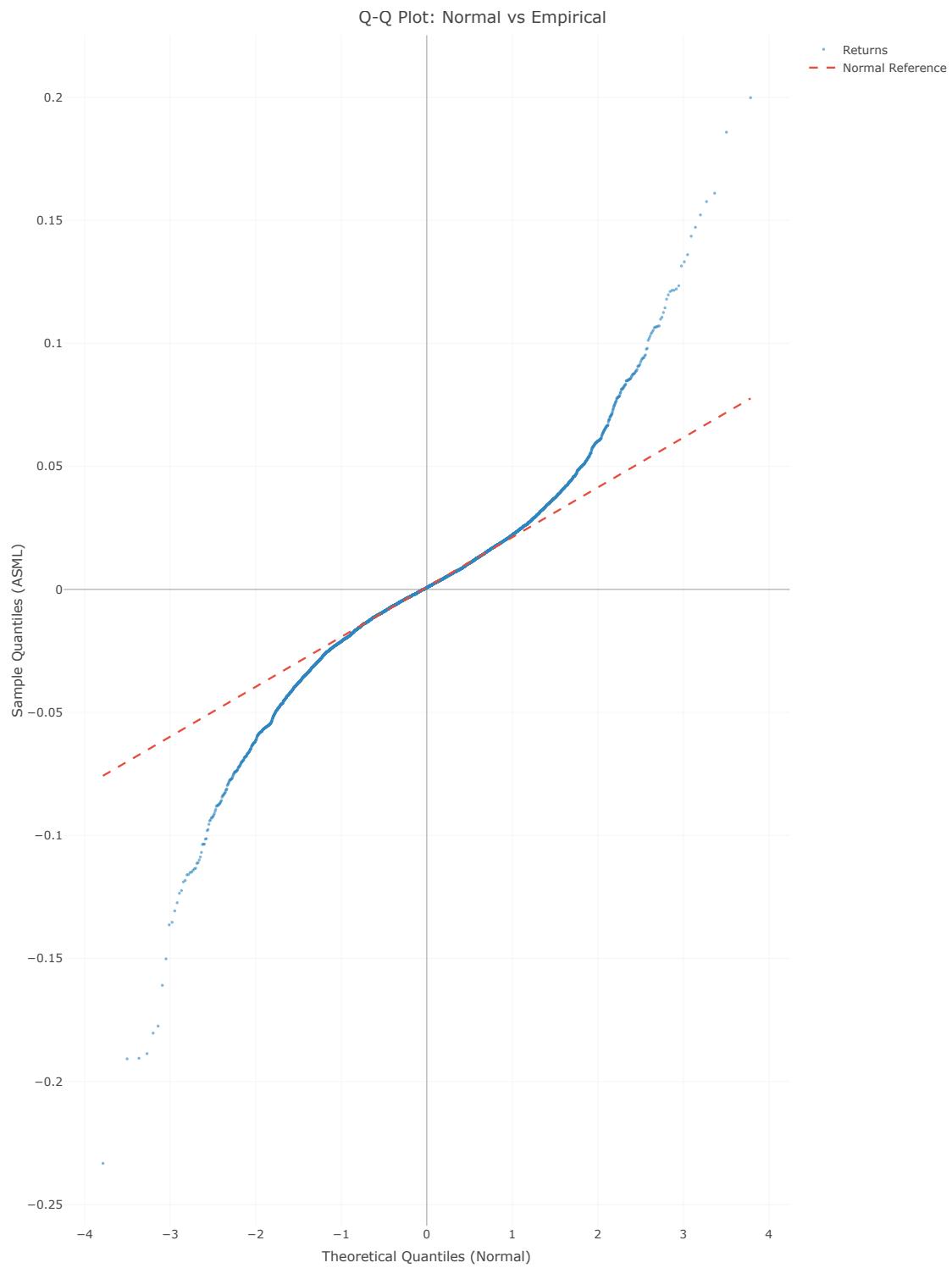


Figure 1.4: Q-Q Plot of ASML Log-Returns

The **Q-Q plot** confirms the findings from the histogram. While the central observations (the body of the distribution) align well with the theoretical normal line (in red), the extreme values at both ends deviate significantly, forming an '*S-shape*'. This provides clear visual evidence of fat tails (leptokurtosis), reinforcing the stylized fact that extreme market movements occur more frequently than predicted by a Gaussian model.

1.3.2 Synthetic indicators

```
desc_stats <- data.frame(
  Metric = c("Mean", "Std. Dev.", "Skewness", "Kurtosis"),
  Value = c(
    mean(log_returns$LogReturns),
    sd(log_returns$LogReturns),
    skewness(log_returns$LogReturns),
    kurtosis(log_returns$LogReturns)
  )
)

datatable(
  desc_stats,
  options = list(dom = 't', paging = FALSE),
  rownames = FALSE
) %>% formatRound('Value', digits = 5)
```

Metric	Value
Mean	0.00052
Std. Dev.	0.02843
Skewness	-0.18021
Kurtosis	8.67864

As we can see, the distribution has a mean close to zero, slight negative skewness, and significant leptokurtosis (fat tails). These characteristics align with the well-known **stylized facts** of financial log-returns, indicating a departure from normality.

1.3.3 Test of normality

To formally test the normality hypothesis, we employ the **Jarque-Bera test**, which is based on skewness and kurtosis matching a normal distribution.

H_0 : The data is normally distributed (Skewness=0, Kurtosis=3)

H_1 : The data is not normally distributed

```
jb_test <- jarque.bera.test(log_returns$LogReturns)

jb_res <- data.frame(
  Test = "Jarque-Bera",
  Statistic = round(jb_test$statistic, 2),
  P_Value = ifelse(jb_test$p.value < 0.001, "< 0.001", round(jb_test$p.value, 4)),
  Result = ifelse(jb_test$p.value < 0.05, "Reject H0", "Fail to Reject H0")
)
datatable(jb_res, options = list(dom = 't'), rownames = FALSE)
```

Test	Statistic	P_Value	Result
Jarque-Bera	8818.65	< 0.001	Reject H0

Since the p-value is virtually zero (<0.001), we strongly reject the null hypothesis of normality. This confirms that ASML returns are not normally distributed.

2 Return Analysis

The objective of this chapter is to identify the optimal ARMA(p,q) model to describe the dynamics of ASML log-returns. To determine the most suitable stochastic process, we adopt a systematic approach structured in the following stages:

1. **Exploratory Analysis:** Analysis of autocorrelation functions (**ACF/PACF**) to assess temporal dependency and detect potential seasonality or distinct patterns.
2. **Model Selection (Grid Search):** Estimation of various order combinations, residual analysis (plots and **Ljung-Box test**), and selection of the “winning” models based on the minimization of **AIC** (Akaike Information Criterion) and **BIC** (Bayesian Information Criterion).
3. **Out-of-Sample Validation:** Comparison of the predictive performance of the selected models against a benchmark (historical mean) using a test set withheld from the estimation process.

2.1 ACF and PACF Analysis

Analyzing correlograms is the preliminary step to identify potential orders for the **AR (AutoRegressive)** and **MA (Moving Average)** components.

```
acf_res <- acf(log_returns$LogReturns, plot = FALSE, lag.max = 20)
pacf_res <- pacf(log_returns$LogReturns, plot = FALSE, lag.max = 20)

ci <- qnorm((1 + 0.95)/2)/sqrt(length(log_returns$LogReturns))

df_acf <- data.frame(
  lag = as.numeric(acf_res$lag)[-1],
  acf = as.numeric(acf_res$acf)[-1])

df_pacf <- data.frame(
  lag = as.numeric(pacf_res$lag),
  pacf = as.numeric(pacf_res$acf))

fig_acf <- plot_ly(df_acf, x = ~lag, y = ~acf, type = 'bar', name = 'ACF',
  marker = list(color = '#1f77b4')) %>%
  add_segments(x = min(df_acf$lag), xend = max(df_acf$lag), y = ci, yend = ci,
  line = list(color = 'red', dash = 'dash', width = 1), showlegend = FALSE) %>%
  add_segments(x = min(df_acf$lag), xend = max(df_acf$lag), y = -ci, yend = -ci,
```

```

line = list(color = 'red', dash = 'dash', width = 1), showlegend = FALSE) %>%
layout(yaxis = list(title = "ACF"))

fig_pacf <- plot_ly(df_pacf, x = ~lag, y = ~pacf, type = 'bar', name = 'PACF',
marker = list(color = '#ff7f0e')) %>%
add_segments(x = min(df_pacf$lag), xend = max(df_pacf$lag), y = ci, yend = ci,
line = list(color = 'red', dash = 'dash', width = 1), showlegend = FALSE) %>%
add_segments(x = min(df_pacf$lag), xend = max(df_pacf$lag), y = -ci, yend = -ci,
line = list(color = 'red', dash = 'dash', width = 1), showlegend = FALSE) %>%
layout(yaxis = list(title = "PACF"))

subplot(fig_acf, fig_pacf, nrows = 1, margin = 0.05, titleY = TRUE) %>%
layout(title = "Autocorrelation & Partial Autocorrelation", margin = list(t = 50))

```

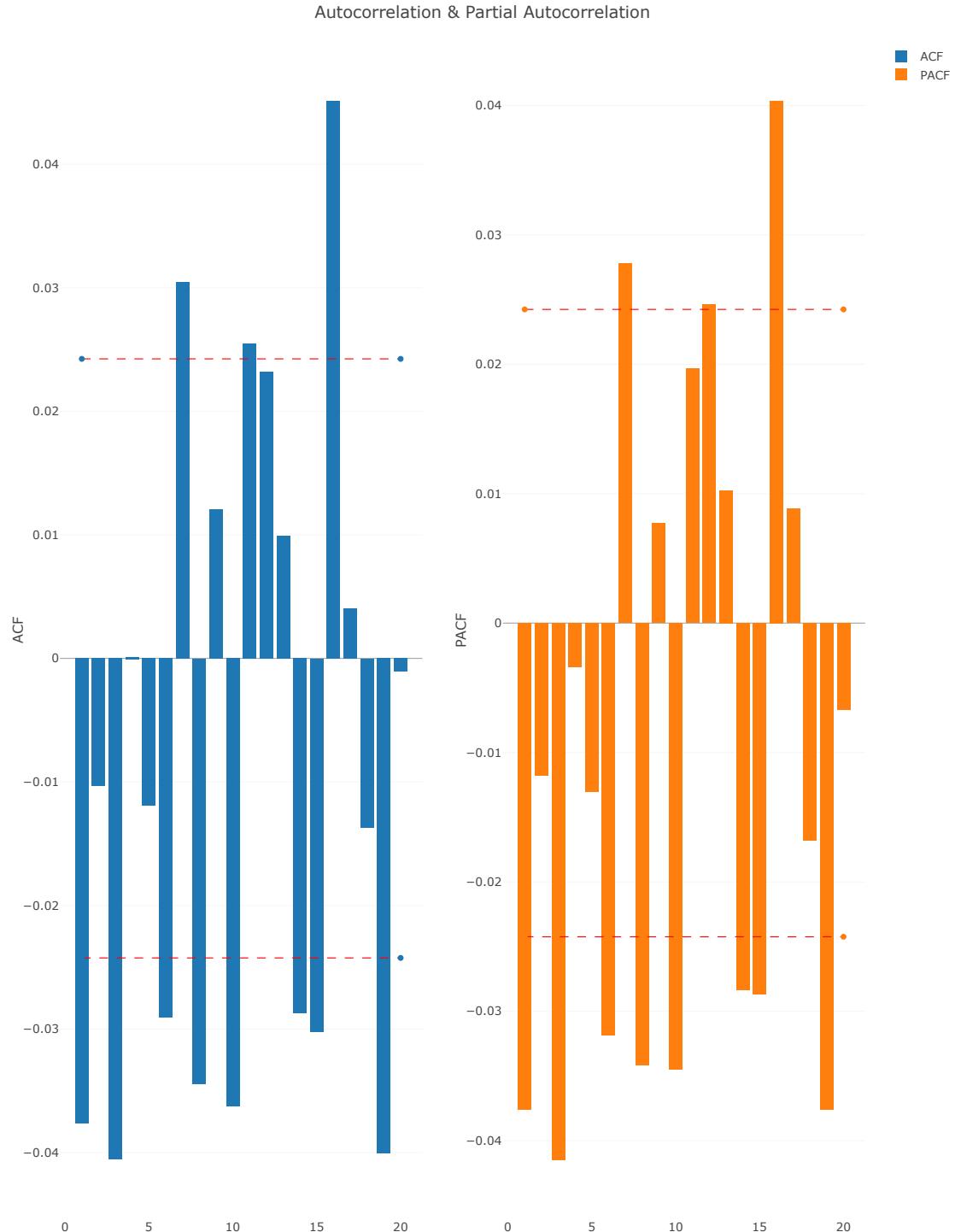


Figure 2.1: Log-RetURNS ACF and PACF

Observations: The autocorrelation at lag-1 is statistically significant, yet its magnitude is limited (approximately -0.04). Neither the ACF nor the PACF exhibits a clear decay or a sharp cutoff. This ambiguity makes it difficult to visually identify a pure AR or MA process, suggesting instead a mixed ARMA structure.

A classical iterative approach (“Box-Jenkins”) would proceed by analyzing the **residuals**:

- If the residuals’ ACF shows a sharp cutoff → add an MA term.
- If the residuals’ PACF shows a sharp cutoff → add an AR term.
- If both plots decay slowly or are unclear → increase both orders.

However, given the ambiguity of the observed patterns and considering that high-order lags (> 3) are uncommon in financial contexts, we opted for a more transparent and systematic approach. We will perform a comprehensive search, testing all possible combinations within a limited range.

2.2 Models

To assess **the goodness of fit** for each configuration, we analyze two fundamental diagnostic outputs:

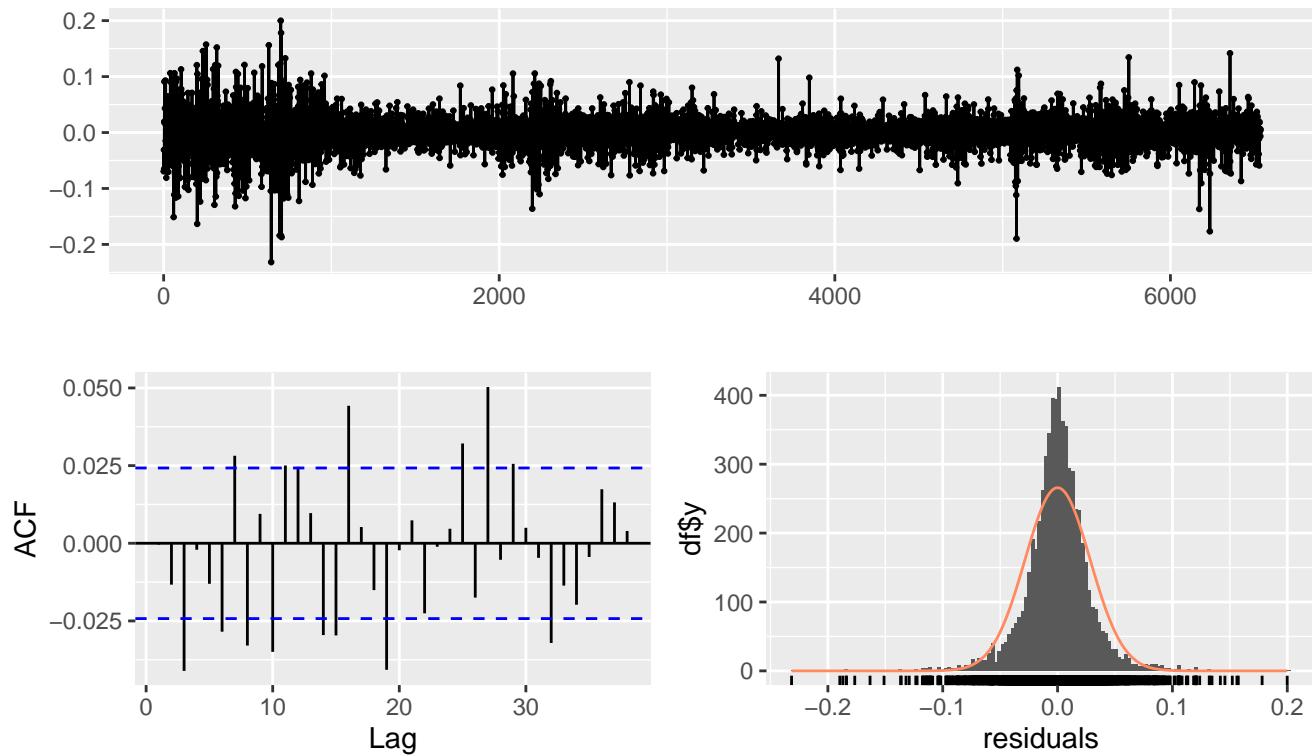
- **Residual Analysis (checkresiduals):** This function allows us to verify the absence of autocorrelation in the residuals. In addition to a visual inspection of the plots, the **Ljung-Box Test** is performed.
 - H_0 (Null Hypothesis): The residuals are independently distributed (White Noise).
 - Interpretation: A p-value > 0.05 indicates that there is insufficient evidence to reject the null hypothesis. Therefore, the model has adequately captured the temporal structure of the data.
- **Parameter Significance (coeftest):** This reports the estimated coefficients (*ar* and *ma*) along with their Standard Errors. Using the z-test, we verify whether the coefficients are statistically different from zero (p-value < 0.05).

```
set.seed(123)
```

2.2.1 AR(1)

```
arma_10 <- Arima(log_returns$LogReturns, order=c(1,0,0))
checkresiduals(arma_10)
```

Residuals from ARIMA(1,0,0) with non-zero mean



Ljung-Box test

```
data: Residuals from ARIMA(1,0,0) with non-zero mean
Q* = 39.515, df = 9, p-value = 9.299e-06
```

Model df: 1. Total lags used: 10

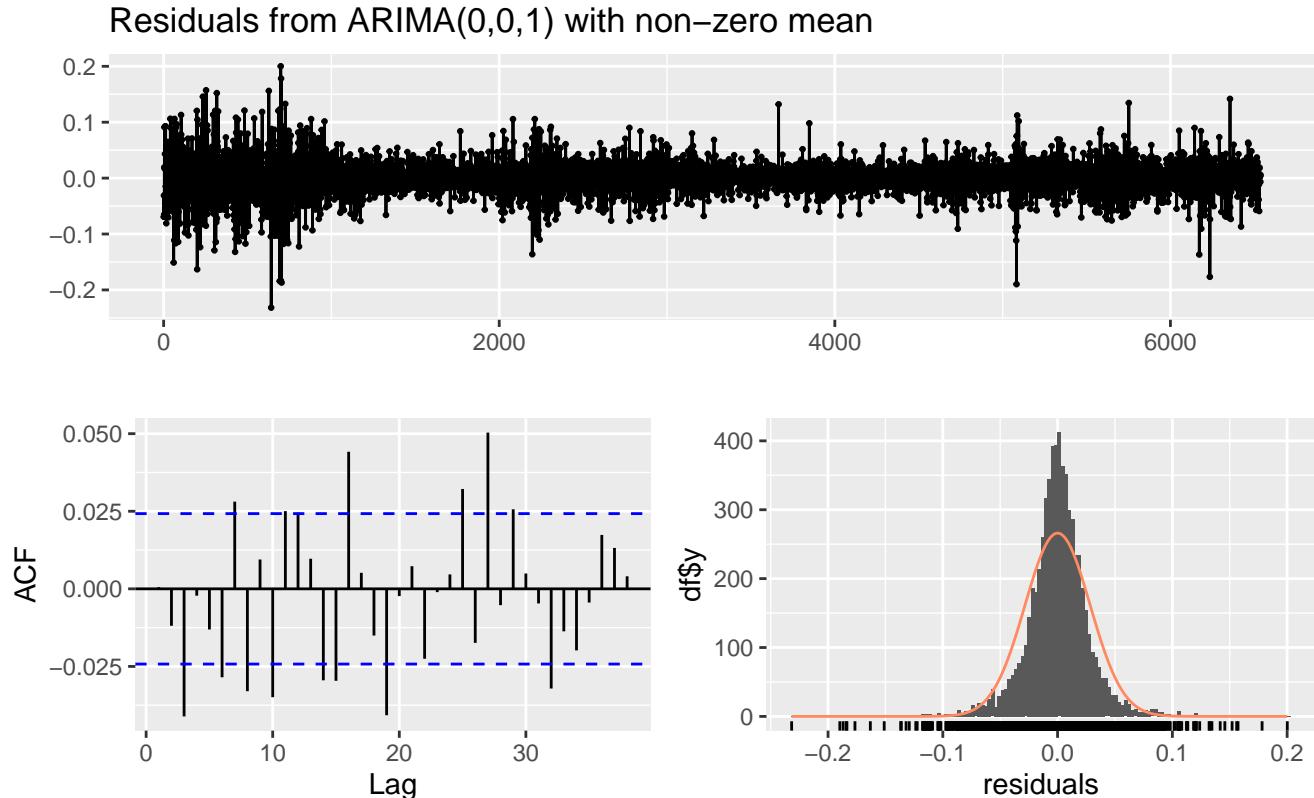
```
coefest(arma_10)
```

z test of coefficients:

	Estimate	Std. Error	z value	Pr(> z)
ar1	-0.03766313	0.01236477	-3.0460	0.002319 **
intercept	0.00051722	0.00033887	1.5263	0.126937
<hr/>				
Signif. codes:	0 ***	0.001 **	0.01 *	0.05 .

2.2.2 MA(1)

```
arma_01 <- Arima(log_returns$LogReturns, order=c(0,0,1))
checkresiduals(arma_01)
```



Ljung-Box test

```
data: Residuals from ARIMA(0,0,1) with non-zero mean
Q* = 39.316, df = 9, p-value = 1.01e-05
```

```
Model df: 1. Total lags used: 10
```

```
coeftest(arma_01)
```

z test of coefficients:

	Estimate	Std. Error	z value	Pr(> z)
ma1	-0.03870342	0.01255525	-3.0826	0.002052 **

```

intercept  0.00051675  0.00033802  1.5288  0.126326
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

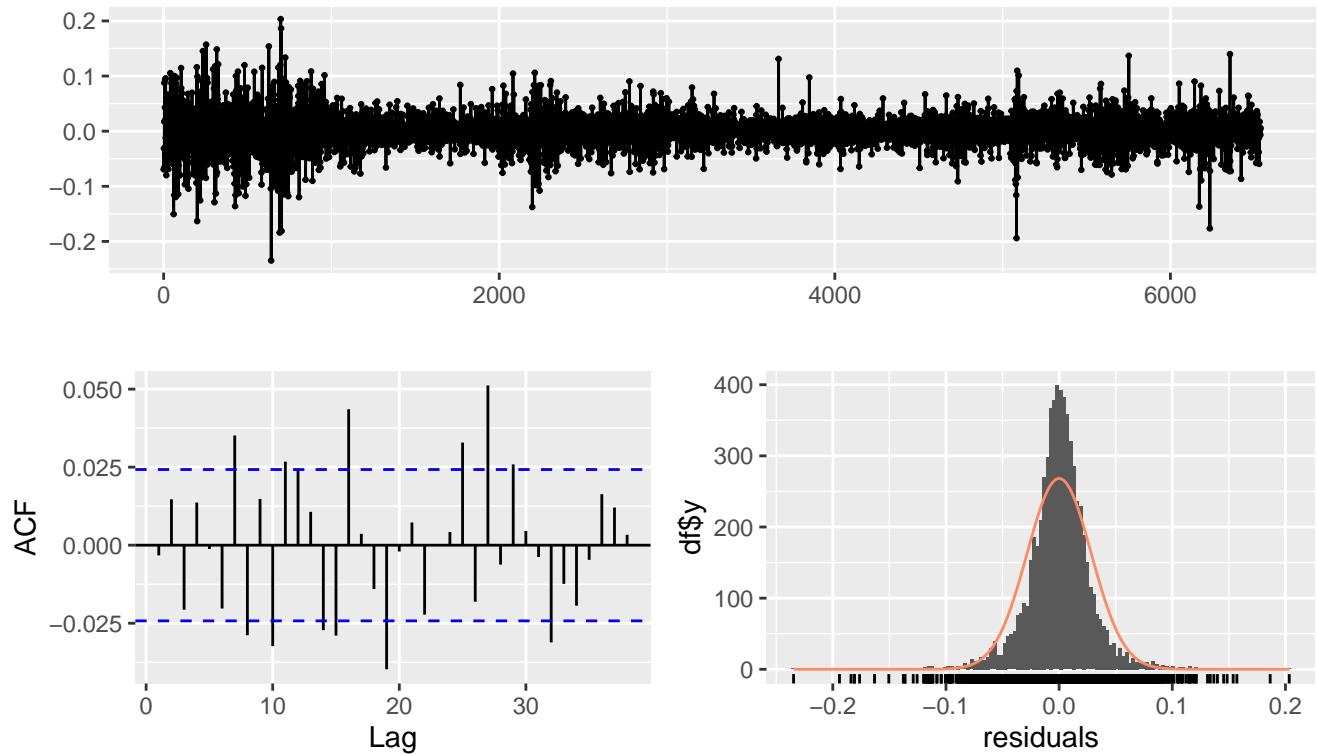
2.2.3 ARMA(1,1)

```

arma_11 <- Arima(log_returns$LogReturns, order=c(1,0,1))
checkresiduals(arma_11)

```

Residuals from ARIMA(1,0,1) with non-zero mean



Ljung-Box test

```

data: Residuals from ARIMA(1,0,1) with non-zero mean
Q* = 30.017, df = 8, p-value = 0.0002099

```

```

Model df: 2.  Total lags used: 10

```

```

coeftest(arma_11)

```

```
z test of coefficients:
```

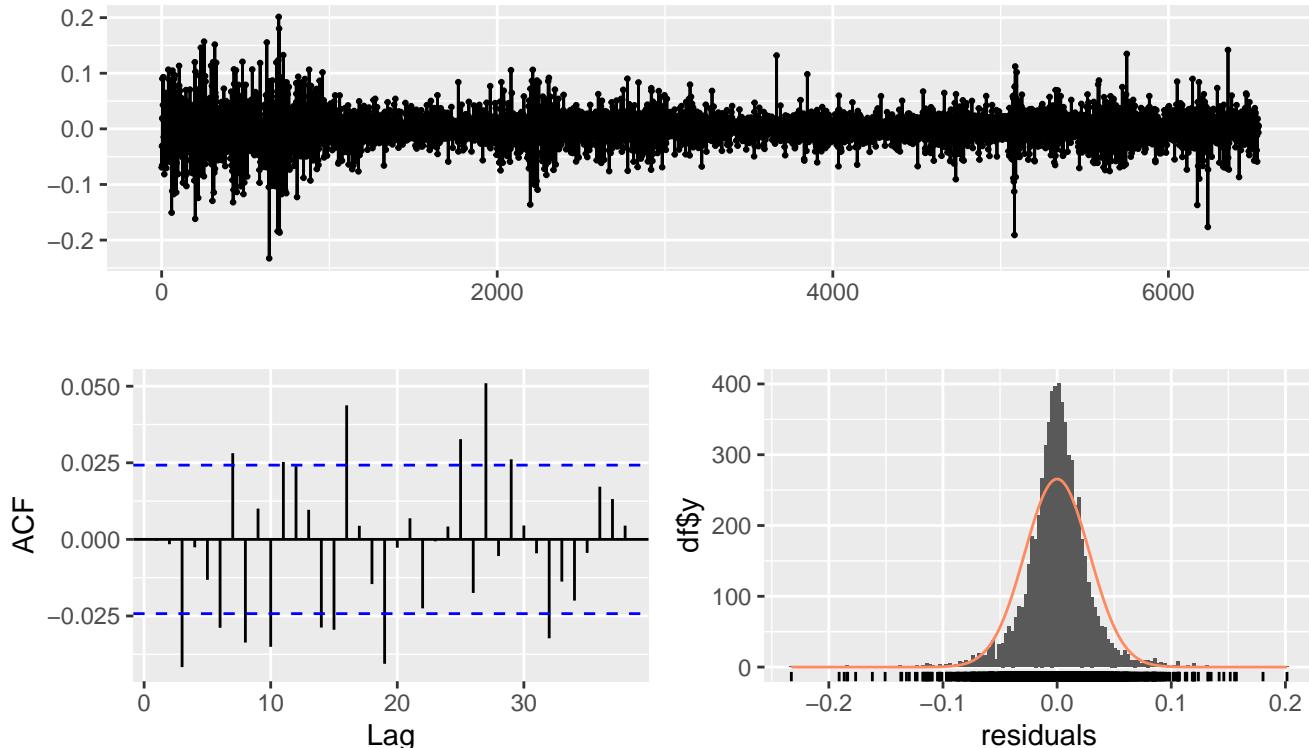
	Estimate	Std. Error	z value	Pr(> z)
ar1	0.74835795	0.09285002	8.0599	7.638e-16 ***
ma1	-0.78494073	0.08686041	-9.0368	< 2.2e-16 ***
intercept	0.00049196	0.00030037	1.6379	0.1015

Signif. codes:	0	'***'	0.001 '**'	0.01 '*'
			0.05 '.'	0.1 ' '
			1	

2.2.4 AR(2)

```
arma_20 <- Arima(log_returns$LogReturns, order=c(2,0,0))
checkresiduals(arma_20)
```

Residuals from ARIMA(2,0,0) with non-zero mean



Ljung-Box test

```
data: Residuals from ARIMA(2,0,0) with non-zero mean
Q* = 39.327, df = 8, p-value = 4.275e-06
```

Model df: 2. Total lags used: 10

```
coefest(arma_20)
```

z test of coefficients:

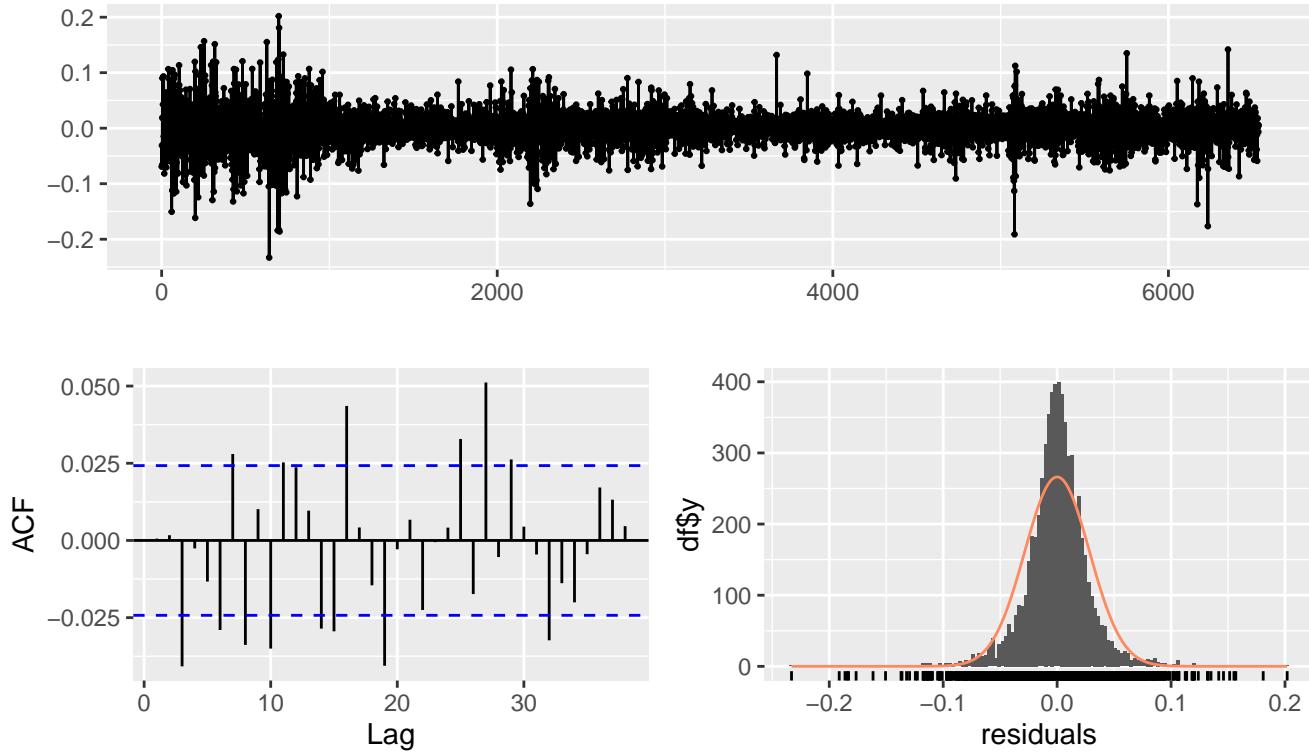
	Estimate	Std. Error	z value	Pr(> z)
ar1	-0.03811204	0.01237289	-3.0803	0.002068 **
ar2	-0.01178780	0.01237261	-0.9527	0.340725
intercept	0.00051767	0.00033491	1.5457	0.122176

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

2.2.5 MA(2)

```
arma_02 <- Arima(log_returns$LogReturns, order=c(0,0,2))
checkresiduals(arma_02)
```

Residuals from ARIMA(0,0,2) with non-zero mean



Ljung-Box test

```
data: Residuals from ARIMA(0,0,2) with non-zero mean  
Q* = 38.929, df = 8, p-value = 5.066e-06
```

Model df: 2. Total lags used: 10

```
coefest(arma_02)
```

z test of coefficients:

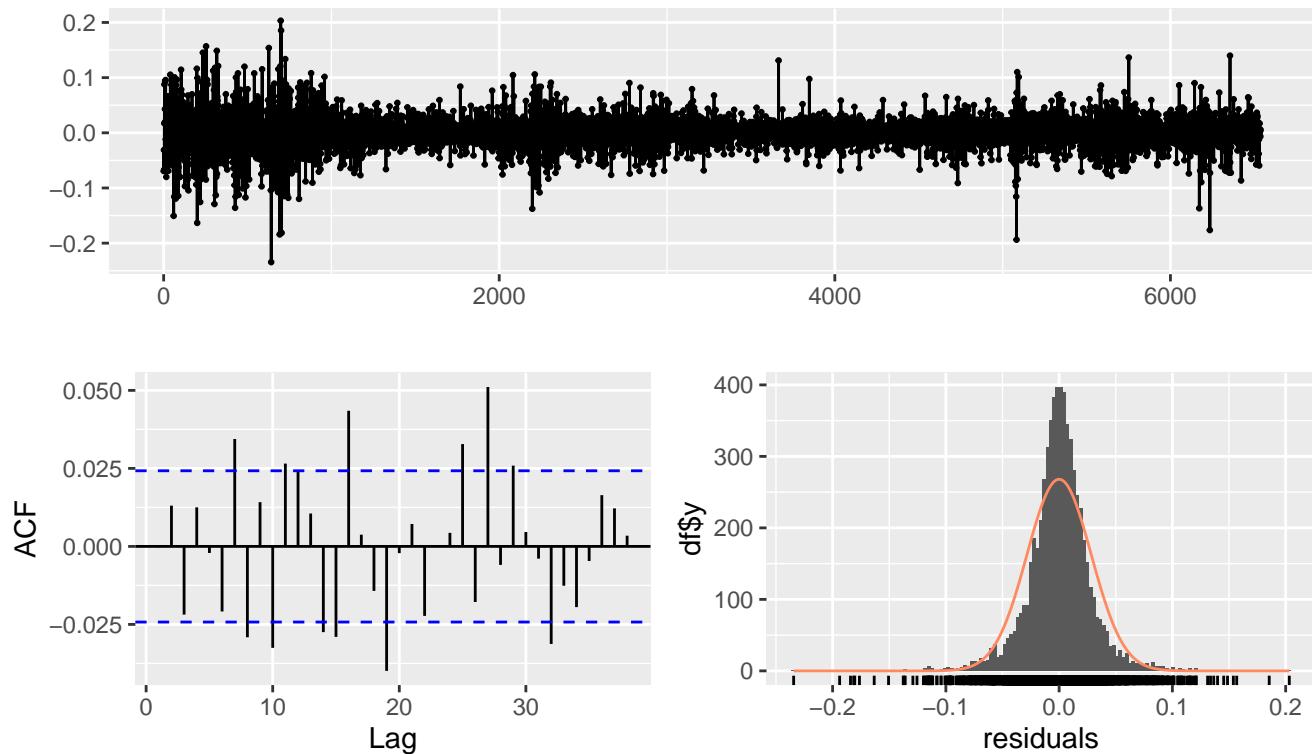
	Estimate	Std. Error	z value	Pr(> z)
ma1	-0.03931345	0.01239570	-3.1715	0.001516 **
ma2	-0.01363525	0.01243057	-1.0969	0.272680
intercept	0.00051676	0.00033299	1.5519	0.120694

Signif. codes:	0 ***	0.001 **	0.01 *	0.05 .
	'	'	'	'
	1			

2.2.6 ARMA(1,2)

```
arma_12 <- Arima(log_returns$LogReturns, order=c(1,0,2))  
checkresiduals(arma_12)
```

Residuals from ARIMA(1,0,2) with non-zero mean



Ljung-Box test

```
data: Residuals from ARIMA(1,0,2) with non-zero mean
Q* = 29.688, df = 7, p-value = 0.0001083
```

Model df: 3. Total lags used: 10

```
coefest(arma_12)
```

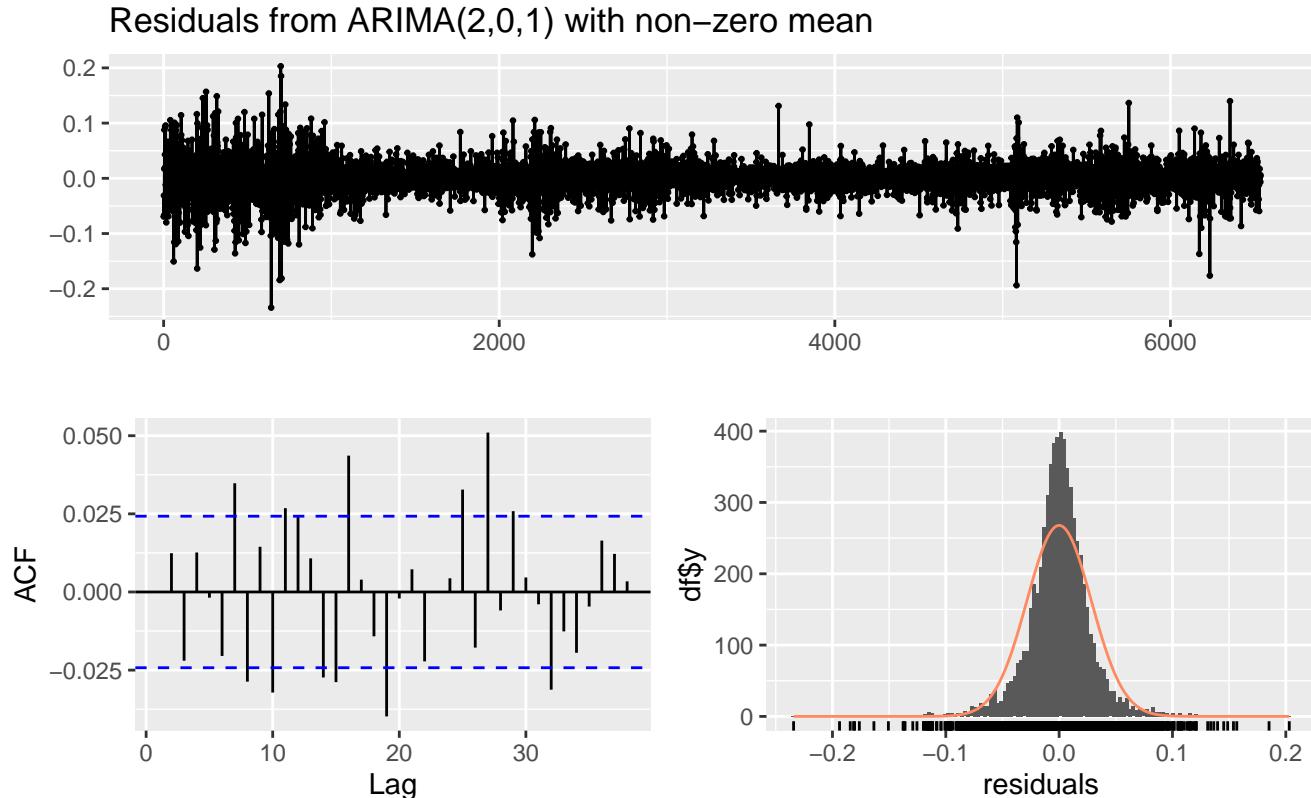
z test of coefficients:

	Estimate	Std. Error	z value	Pr(> z)
ar1	0.74593520	0.12744730	5.8529	4.831e-09 ***
ma1	-0.78546932	0.12794838	-6.1390	8.307e-10 ***
ma2	0.00378391	0.01582148	0.2392	0.81098
intercept	0.00052830	0.00030199	1.7494	0.08023 .

Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'
	1			

2.2.7 ARMA(2,1)

```
arma_21 <- Arima(log_returns$LogReturns, order=c(2,0,1))
checkresiduals(arma_21)
```



Ljung-Box test

```
data: Residuals from ARIMA(2,0,1) with non-zero mean
Q* = 29.435, df = 7, p-value = 0.0001205
```

Model df: 3. Total lags used: 10

```
coeftest(arma_21)
```

z test of coefficients:

	Estimate	Std. Error	z value	Pr(> z)
ar1	0.75205133	0.10514843	7.1523	8.535e-13 ***

```

ar2      0.00504381  0.01520674  0.3317   0.74013
ma1     -0.79207112  0.10452900 -7.5775 3.522e-14 ***
intercept 0.00052286  0.00030084  1.7380   0.08221 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

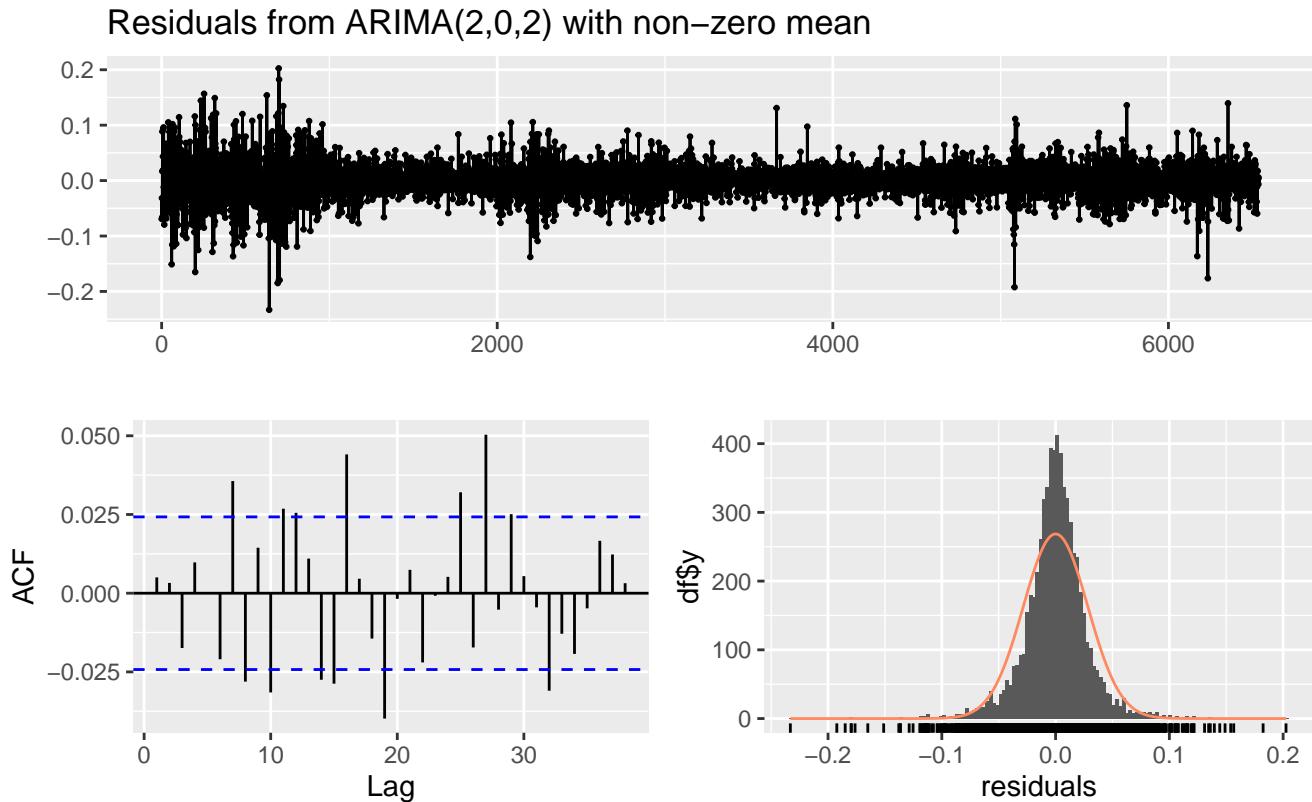
```

2.2.8 ARMA(2,2)

```

arma_22 <- Arima(log_returns$LogReturns, order=c(2,0,2))
checkresiduals(arma_22)

```



Ljung-Box test

```

data: Residuals from ARIMA(2,0,2) with non-zero mean
Q* = 27.067, df = 6, p-value = 0.0001407

```

```

Model df: 4.  Total lags used: 10

```

```
coeftest(arma_22)
```

z test of coefficients:

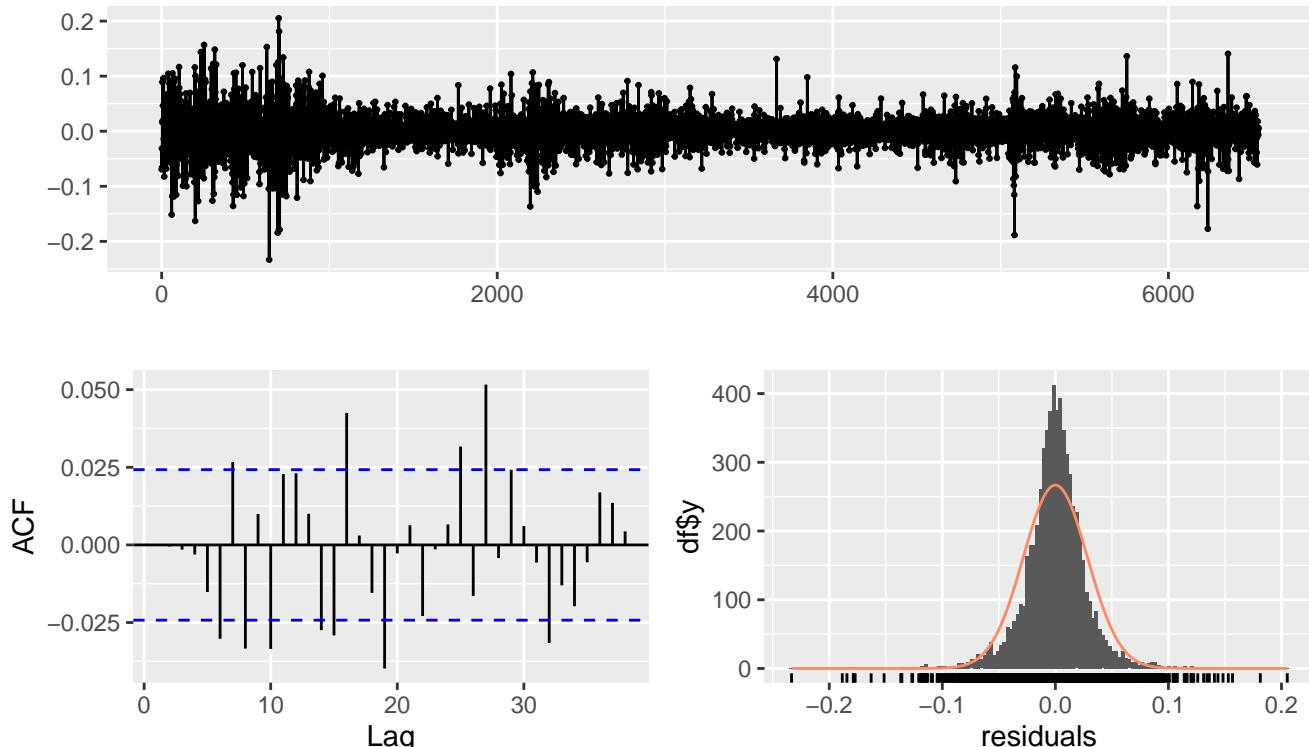
	Estimate	Std. Error	z value	Pr(> z)
ar1	0.13889874	0.20809305	0.6675	0.5044632
ar2	0.49269066	0.13448637	3.6635	0.0002488 ***
ma1	-0.18349708	0.20967841	-0.8751	0.3815001
ma2	-0.50222812	0.14274632	-3.5183	0.0004343 ***
intercept	0.00052958	0.00029978	1.7666	0.0772984 .

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

2.2.9 AR(3)

```
arma_30 <- Arima(log_returns$LogReturns, order=c(3,0,0))  
checkresiduals(arma_30)
```

Residuals from ARIMA(3,0,0) with non-zero mean



Ljung-Box test

```
data: Residuals from ARIMA(3,0,0) with non-zero mean  
Q* = 27.506, df = 7, p-value = 0.0002701
```

Model df: 3. Total lags used: 10

```
coeftest(arma_30)
```

z test of coefficients:

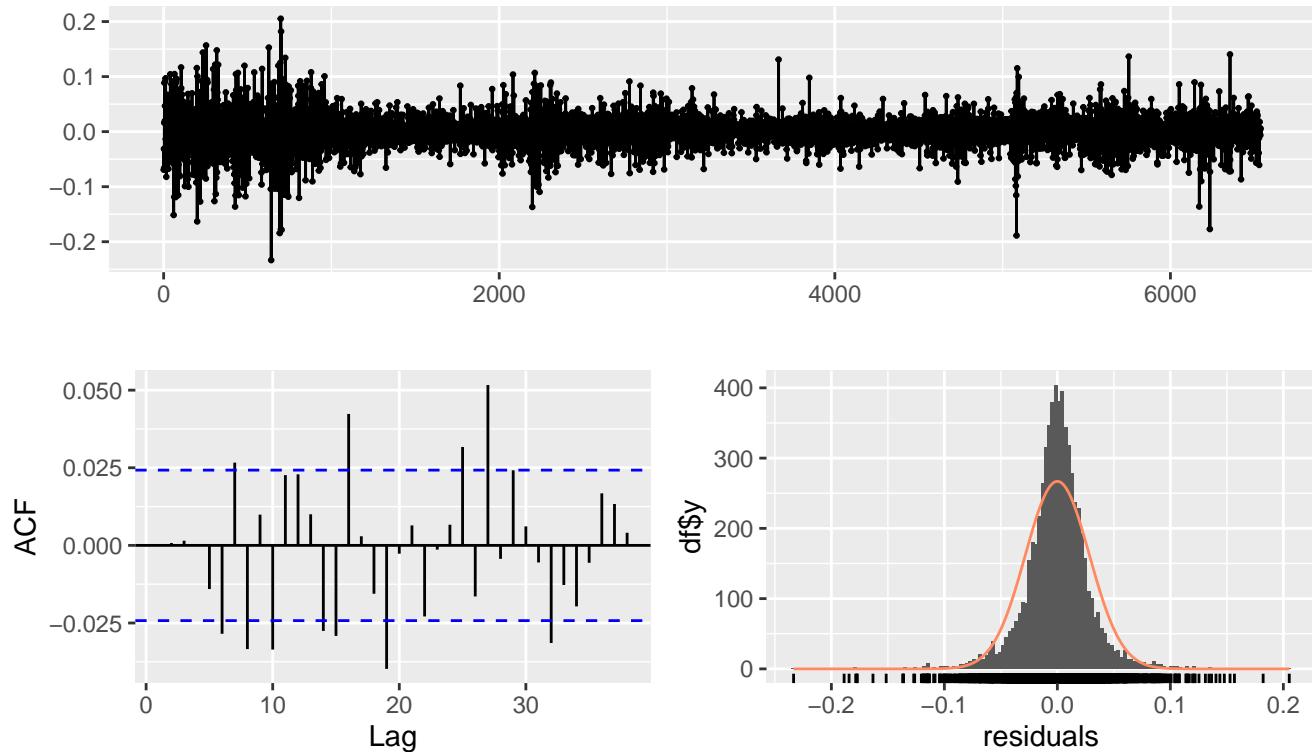
	Estimate	Std. Error	z value	Pr(> z)							
ar1	-0.03863768	0.01236319	-3.1252	0.0017767 **							
ar2	-0.01340447	0.01237126	-1.0835	0.2785792							
ar3	-0.04157126	0.01236735	-3.3614	0.0007756 ***							
intercept	0.00051836	0.00032127	1.6135	0.1066419							

Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	'. '	0.1	' '	1

2.2.10 MA(3)

```
arma_03 <- Arima(log_returns$LogReturns, order=c(0,0,3))  
checkresiduals(arma_03)
```

Residuals from ARIMA(0,0,3) with non-zero mean



Ljung-Box test

```
data: Residuals from ARIMA(0,0,3) with non-zero mean
Q* = 26.596, df = 7, p-value = 0.000394
```

Model df: 3. Total lags used: 10

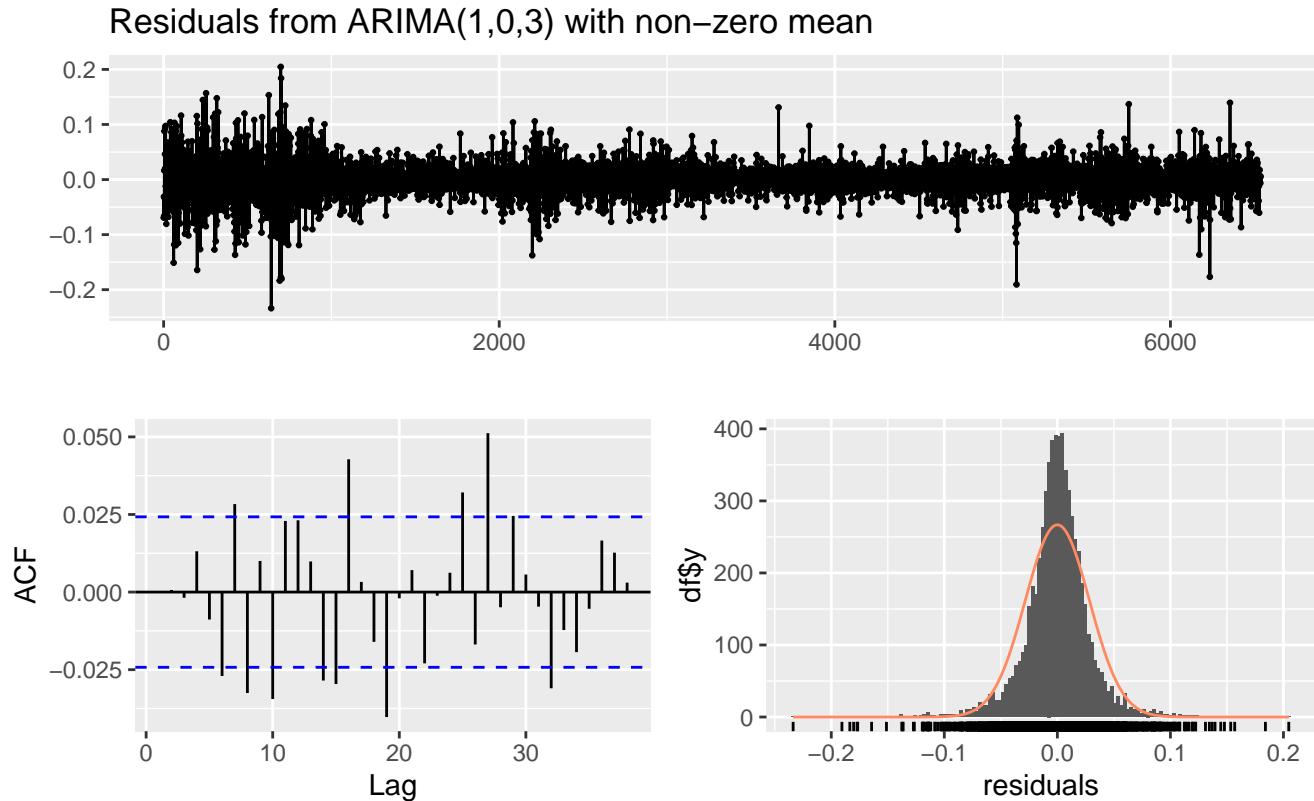
```
coeftest(arma_03)
```

z test of coefficients:

	Estimate	Std. Error	z value	Pr(> z)
ma1	-0.03884109	0.01237078	-3.1397	0.0016909 **
ma2	-0.01337143	0.01235618	-1.0822	0.2791791
ma3	-0.04358355	0.01269756	-3.4324	0.0005982 ***
intercept	0.00051686	0.00031767	1.6270	0.1037266
<hr/>				
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1				

2.2.11 ARMA(1,3)

```
arma_13 <- Arima(log_returns$LogReturns, order=c(1,0,3))
checkresiduals(arma_13)
```



Ljung–Box test

```
data: Residuals from ARIMA(1,0,3) with non-zero mean
Q* = 27.07, df = 6, p-value = 0.0001405
```

```
Model df: 4. Total lags used: 10
```

```
coeftest(arma_13)
```

```
Warning in sqrt(diag(se)): NaNs produced
```

```
z test of coefficients:
```

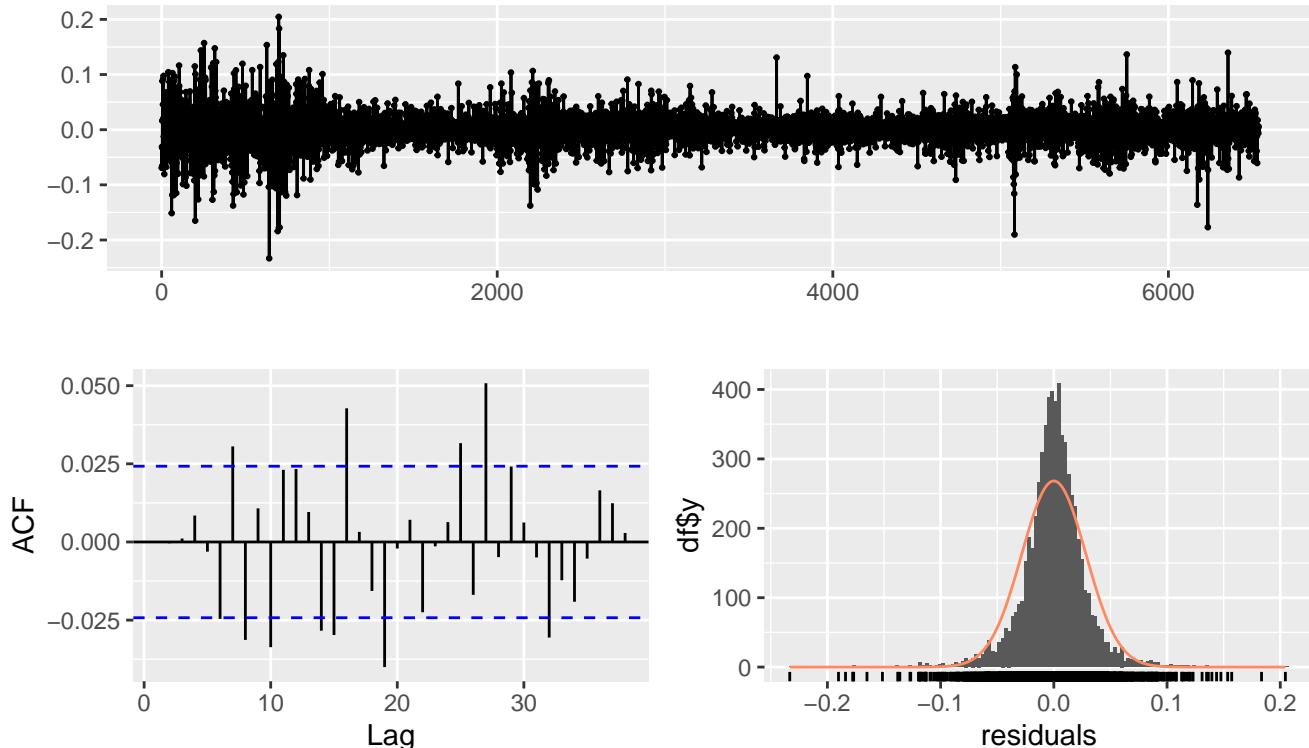
	Estimate	Std. Error	z value	Pr(> z)
ar1	0.33895398	NaN	NaN	NaN
ma1	-0.37787655	NaN	NaN	NaN
ma2	-0.00033565	NaN	NaN	NaN
ma3	-0.03534996	NaN	NaN	NaN
intercept	0.00051530	0.00031168	1.6533	0.09827 .

Signif. codes:	0 *** 0.001 ** 0.01 * 0.05 . 0.1 ' ' 1			

2.2.12 ARMA(2,3)

```
arma_23 <- Arima(log_returns$LogReturns, order=c(2,0,3))
checkresiduals(arma_23)
```

Residuals from ARIMA(2,0,3) with non-zero mean



Ljung-Box test

```
data: Residuals from ARIMA(2,0,3) with non-zero mean
Q* = 25.221, df = 5, p-value = 0.0001263
```

```
Model df: 5. Total lags used: 10
```

```
coefest(arma_23)
```

```
z test of coefficients:
```

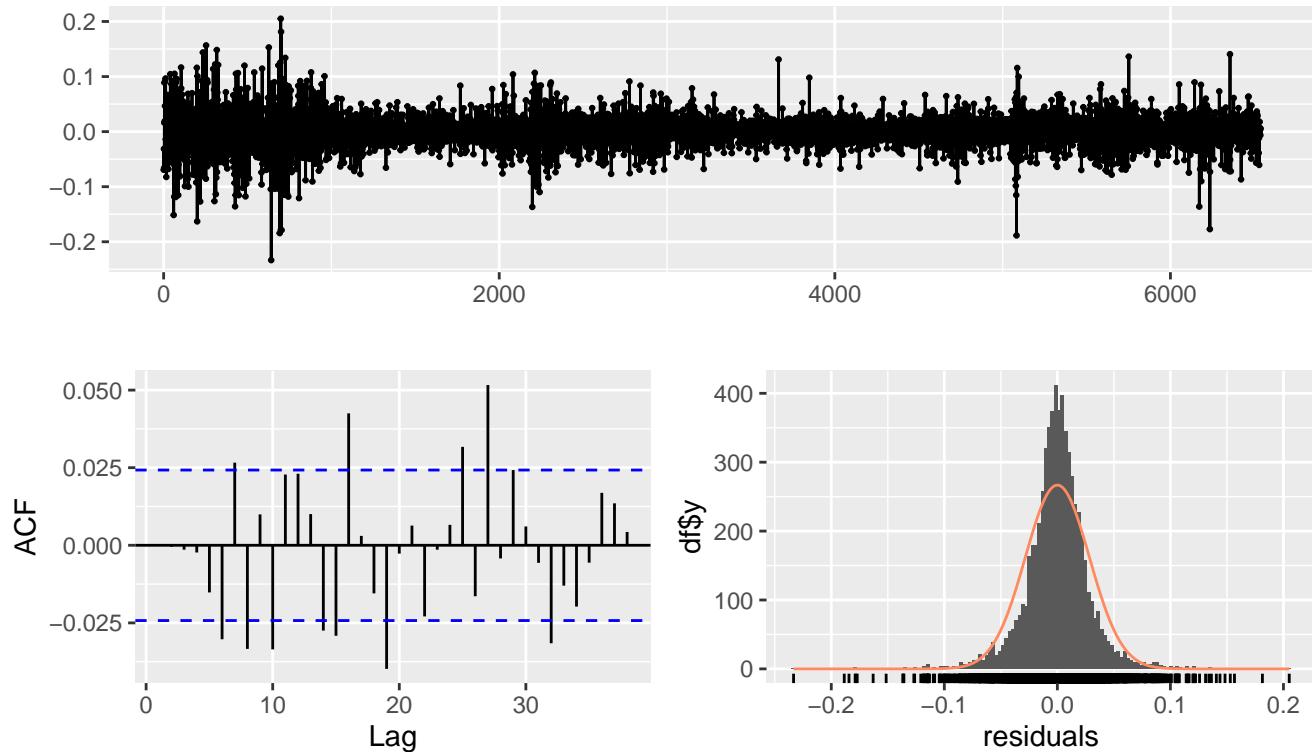
	Estimate	Std. Error	z value	Pr(> z)
ar1	0.12929122	0.26476513	0.4883	0.62532
ar2	0.22589447	0.19486231	1.1593	0.24635
ma1	-0.16827064	0.26487284	-0.6353	0.52524
ma2	-0.23309652	0.19068064	-1.2224	0.22154
ma3	-0.03296964	0.01859007	-1.7735	0.07614 .
intercept	0.00050770	0.00030819	1.6474	0.09948 .

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

2.2.13 ARMA(3,1)

```
arma_31 <- Arima(log_returns$LogReturns, order=c(3,0,1))
checkresiduals(arma_31)
```

Residuals from ARIMA(3,0,1) with non-zero mean



Ljung-Box test

```
data: Residuals from ARIMA(3,0,1) with non-zero mean
Q* = 27.501, df = 6, p-value = 0.0001167
```

```
Model df: 4. Total lags used: 10
```

```
coefest(arma_31)
```

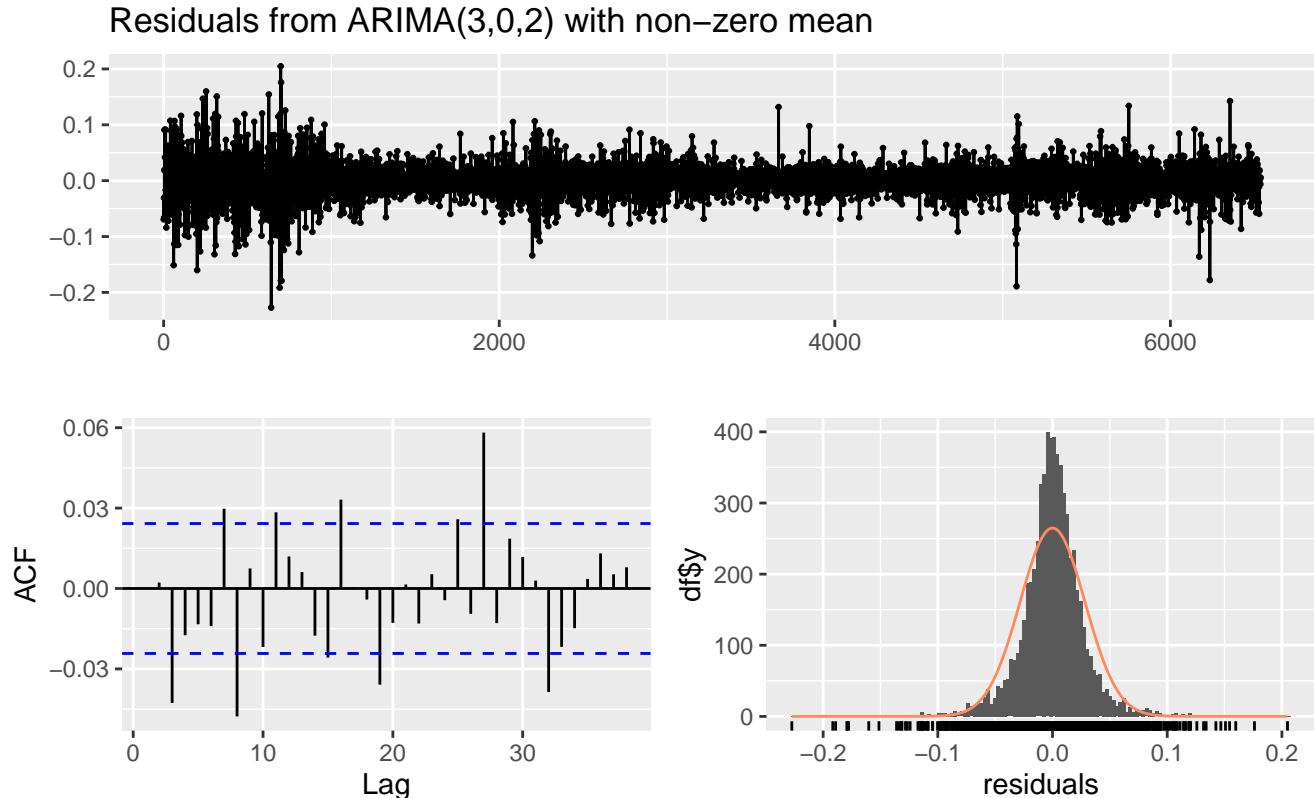
```
z test of coefficients:
```

	Estimate	Std. Error	z value	Pr(> z)
ar1	-0.01946401	0.55212413	-0.0353	0.971878
ar2	-0.01274590	0.02435663	-0.5233	0.600763
ar3	-0.04139616	0.01342386	-3.0838	0.002044 **
ma1	-0.01920240	0.55290607	-0.0347	0.972295
intercept	0.00051900	0.00032097	1.6170	0.105887

Signif. codes:	0 ***	0.001 **	0.01 *	0.05 .
	'	'	'	'

2.2.14 ARMA(3,2)

```
arma_32 <- Arima(log_returns$LogReturns, order=c(3,0,2))
checkresiduals(arma_32)
```



Ljung-Box test

```
data: Residuals from ARIMA(3,0,2) with non-zero mean
Q* = 40.61, df = 5, p-value = 1.125e-07
```

```
Model df: 5. Total lags used: 10
```

```
coeftest(arma_32)
```

z test of coefficients:

	Estimate	Std. Error	z value	Pr(> z)
ar1	0.03149305	0.02356085	1.3367	0.1813309

```

ar2      -0.96653560  0.01833468 -52.7163 < 2.2e-16 ***
ar3      -0.03500277  0.01292549 -2.7080  0.0067682 **
ma1      -0.06983280  0.01998214 -3.4948  0.0004745 ***
ma2       0.95647723  0.02188440 43.7059 < 2.2e-16 ***
intercept 0.00053136  0.00033628  1.5801  0.1140757
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

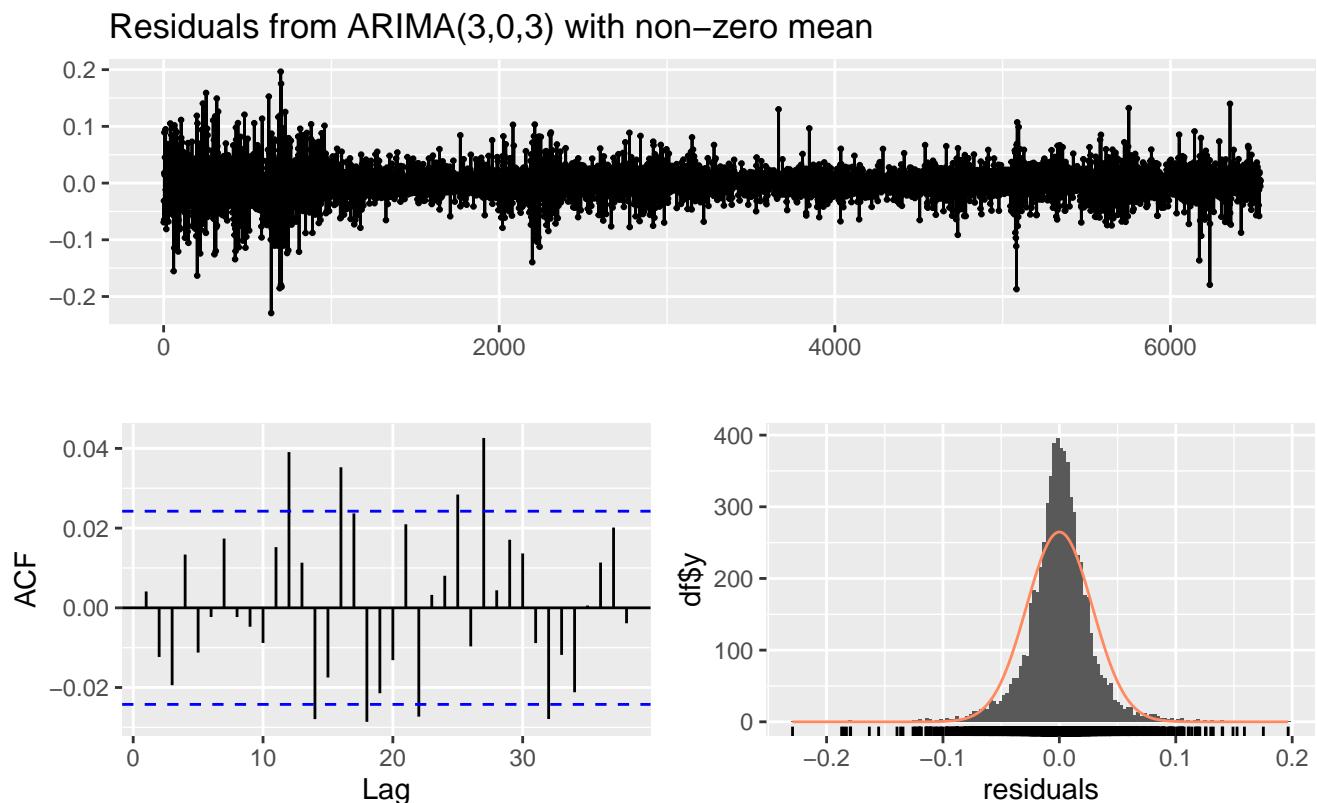
```

2.2.15 ARMA(3,3)

```

arma_33 <- Arima(log_returns$LogReturns, order=c(3,0,3))
checkresiduals(arma_33)

```



Ljung-Box test

```

data: Residuals from ARIMA(3,0,3) with non-zero mean
Q* = 8.2696, df = 4, p-value = 0.08219

```

```

Model df: 6.  Total lags used: 10

```

```
coefest(arma_33)
```

z test of coefficients:

	Estimate	Std. Error	z value	Pr(> z)							
ar1	-0.94053284	0.06896629	-13.6376	< 2.2e-16 ***							
ar2	0.69776654	0.11017141	6.3335	2.397e-10 ***							
ar3	0.82856695	0.06357149	13.0336	< 2.2e-16 ***							
ma1	0.90137254	0.06674361	13.5050	< 2.2e-16 ***							
ma2	-0.73818759	0.09584391	-7.7020	1.340e-14 ***							
ma3	-0.82510117	0.05905981	-13.9706	< 2.2e-16 ***							
intercept	0.00050774	0.00028632	1.7733	0.07618 .							

Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	'..'	0.1	' '	1

As observed from the outputs, the simpler models (such as AR(1) and MA(1)) yield very low p-values for the Ljung-Box test (< 0.05). This indicates that significant autocorrelation remains within the residuals, implying that these models have failed to capture all the relevant information.

Conversely, as we increase complexity towards an **ARMA(3,3)**, the p-value rises above the critical 5% threshold. Consequently, we fail to reject the null hypothesis: the residuals of this model behave as **White Noise**. This suggests that the ARMA(3,3) is mathematically effective in capturing the patterns of the time series, although its high complexity warrants caution regarding the risk of **overfitting**.

Additionally, the **ARMA(1,3)** model exhibits instability in parameter estimation (incalculable Std. Error), pointing to potential **over-parameterization**.

2.3 AIC vs BIC

We visualize the trends of the information criteria for all estimated models.

```
models_list <- list(  
  "AR(1)" = arma_10, "MA(1)" = arma_01, "ARMA(1,1)" = arma_11,  
  "AR(2)" = arma_20, "MA(2)" = arma_02, "ARMA(1,2)" = arma_12,  
  "ARMA(2,1)" = arma_21, "ARMA(2,2)" = arma_22,  
  "AR(3)" = arma_30, "MA(3)" = arma_03, "ARMA(1,3)" = arma_13,  
  "ARMA(2,3)" = arma_23, "ARMA(3,1)" = arma_31, "ARMA(3,2)" = arma_32,  
  "ARMA(3,3)" = arma_33  
)  
  
metrics_df <- data.frame(  
  Model = names(models_list),  
  AIC = sapply(models_list, AIC),
```

```

    BIC = sapply(models_list, BIC)
  )

metrics_df$Model <- factor(metrics_df$Model, levels = metrics_df$Model)

min_aic_val <- min(metrics_df$AIC)
min_bic_val <- min(metrics_df$BIC)

plot_aic <- plot_ly(metrics_df, x = ~Model, y = ~AIC, name = 'AIC',
  type = 'scatter', mode = 'lines+markers',
  line = list(color = '#1f77b4', width = 2),
  marker = list(size = 8, color = '#1f77b4')) %>%
  layout(title = "AIC Trend", yaxis = list(title = "AIC"))

best_aic <- metrics_df[metrics_df$AIC == min_aic_val, ]
plot_aic <- plot_aic %>%
  add_trace(data = best_aic, x = ~Model, y = ~AIC,
  type = 'scatter', mode = 'markers', name = 'Best AIC',
  marker = list(color = 'red', size = 12, symbol = 'star'),
  showlegend = FALSE
)

plot_bic <- plot_ly(metrics_df, x = ~Model, y = ~BIC, name = 'BIC',
  type = 'scatter', mode = 'lines+markers',
  line = list(color = '#ff7f0e', width = 2),
  marker = list(size = 8, color = '#ff7f0e')) %>%
  layout(title = "BIC Trend", yaxis = list(title = "BIC"))

best_bic <- metrics_df[metrics_df$BIC == min_bic_val, ]
plot_bic <- plot_bic %>%
  add_trace(
    data = best_bic, x = ~Model, y = ~BIC,
    type = 'scatter', mode = 'markers', name = 'Best BIC',
    marker = list(color = 'red', size = 12, symbol = 'star'),
    showlegend = FALSE
)

final_plot <- subplot(plot_aic, plot_bic, nrows = 2, shareX = TRUE, titleY = TRUE) %>%
  layout(
    title = "Model Selection Criteria: AIC & BIC Trends",
    hovermode = "x unified",
    margin = list(t = 50)
  )

final_plot

```

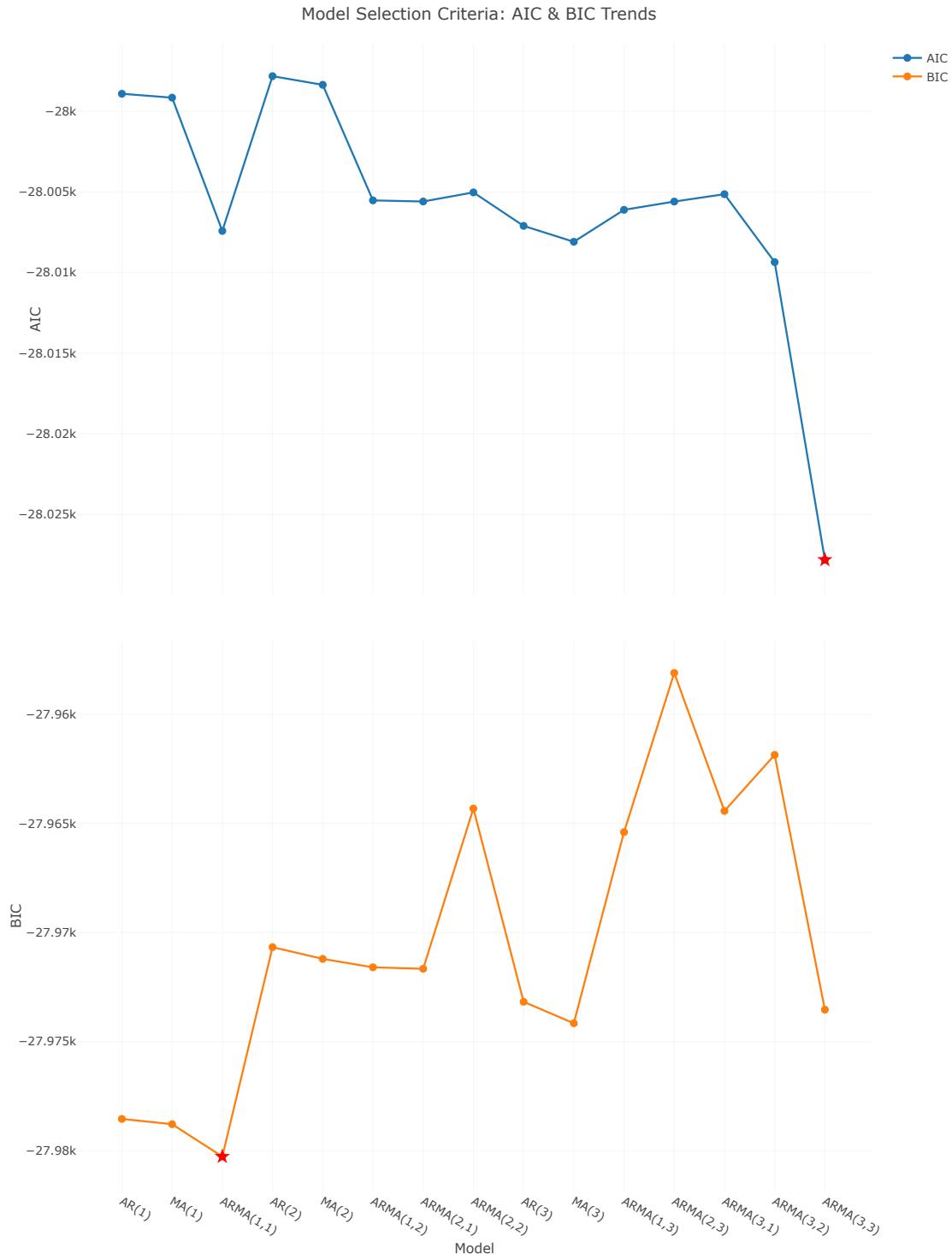


Figure 2.2: Comparison between AIC and BIC

The graphical analysis of the information criteria highlights a typical divergence found in financial time series modeling.

As evidenced by the plot, the **AIC** criterion, which prioritizes **goodness of fit**, identifies the **ARMA(3,3)** as the optimal model (minimum value). Conversely, the **BIC** criterion, which penalizes model complexity more severely to prevent overfitting, suggests a much more **parsimonious** model: the **ARMA(1,1)**.

Faced with these conflicting indications (a complex model vs a simple model), it is not possible to determine a priori which one is superior. To resolve this ambiguity and assess which model generalizes better on unobserved data, we will proceed with an **Out-of-Sample validation**, computing forecasts and comparing error metrics against actual data.

2.4 Forecast

Since the information criteria (AIC and BIC) suggest different models, we proceed with an Out-of-Sample validation to test the actual predictive capacity on data not used for estimation.

We split the time series into two subsets:

- **Training Set** (80): Used to estimate the model parameters.
- **Test Set** (20): Used to compare forecasts against actual data (h-step-ahead forecast).

```
n_total <- nrow(log_returns)
n_train <- floor(0.80 * n_total)

train_data <- log_returns$LogReturns[1:n_train]
test_data <- log_returns$LogReturns[(n_train + 1):n_total]

set.seed(123)

mod_winner_bic <- Arima(train_data, order=c(1,0,1))
mod_winner_aic <- Arima(train_data, order=c(3,0,3))
mod_naive <- Arima(train_data, order=c(0,0,0))

h_steps <- length(test_data)

fc_bic <- forecast(mod_winner_bic, h=h_steps)
fc_aic <- forecast(mod_winner_aic, h=h_steps)
fc_naive <- forecast(mod_naive, h=h_steps)
```

The following plot displays the time series trend and the forecasts generated by the three models over the test period.

Note

Given the stationary nature and high volatility of log-returns, it is expected that medium-to-long-term forecasts will tend to converge rapidly towards the unconditional mean of the process.

```
dates_train <- log_returns$Date[1:n_train]
dates_test  <- log_returns$Date[(n_train + 1):n_total]

fig_fc <- plot_ly(
  x = dates_train,
  y = train_data,
  type = 'scatter',
  mode = 'lines',
  name = 'Train (80%)',
  line = list(color = 'darkgrey', width = 1)
)

fig_fc <- add_trace(
  fig_fc,
  x = dates_test,
  y = test_data,
  name = 'Actual Test Data (20%)',
  line = list(color = 'black', width = 1, dash = 'dot')
)

fig_fc <- add_trace(
  fig_fc,
  x = dates_test,
  y = as.numeric(fc_bic$mean),
  name = 'Forecast ARMA(1,1)',
  line = list(color = 'blue', width = 1.5)
)

fig_fc <- add_trace(
  fig_fc,
  x = dates_test,
  y = as.numeric(fc_aic$mean),
  name = 'Forecast ARMA(3,3)',
  line = list(color = 'red', width = 1.5)
)

fig_fc <- layout(
  fig_fc,
  title = "Forecast Validation: Models vs Actuals",
  xaxis = list(title = "Date"),
```

```
yaxis = list(title = "Log Return"),
legend = list(orientation = "h", x = 0.1, y = -0.2),
shapes = list(
  list(
    type = "line",
    x0 = min(dates_test),
    x1 = min(dates_test),
    y0 = min(log_returns$LogReturns),
    y1 = max(log_returns$LogReturns),
    line = list(color = "gray", width = 1, dash = "dash")
  )
)
)

fig_fc
```

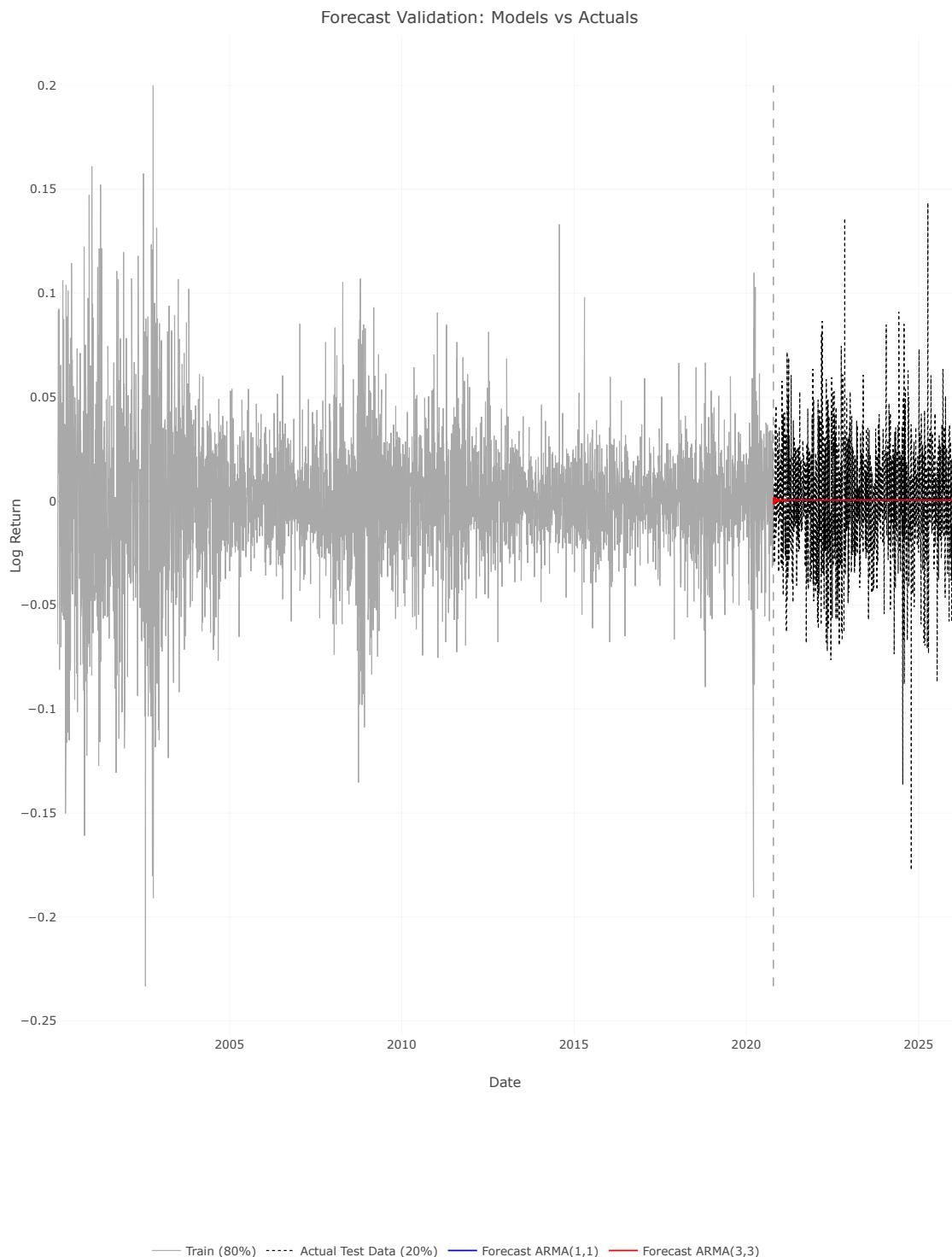


Figure 2.3: Interactive Forecast Validation: Train vs Test vs Models

We evaluate the predictive performance using:

- **MAE (Mean absolute Error)**
- **RMSE (Root Mean Squared Error)**
- **Theil's U2 index:** The ratio between the model's RMSE and the Naive model's RMSE.
 - $U < 1$: The model outperforms the benchmark.
 - $U \approx 1$: The model performs equivalently to the benchmark.
 - $U > 1$: The model performs worse than the benchmark.

```
get_metrics_robust <- function(forecast_obj, actuals) {  
  vec_pred <- as.numeric(forecast_obj$mean)  
  vec_obs <- as.numeric(actuals)  
  
  acc <- accuracy(vec_pred, vec_obs)  
  
  return(c(RMSE = acc[1, "RMSE"], MAE = acc[1, "MAE"]))  
}  
  
res_bic <- get_metrics_robust(fc_bic, test_data)  
res_aic <- get_metrics_robust(fc_aic, test_data)  
res_naive <- get_metrics_robust(fc_naive, test_data)  
  
u_bic <- res_bic["RMSE"] / res_naive["RMSE"]  
u_aic <- res_aic["RMSE"] / res_naive["RMSE"]  
u_naive <- 1.0  
  
validation_table <- data.frame(  
  Model = c("ARMA(1,1) [BIC Winner]", "ARMA(3,3) [AIC Winner]", "Naive (Benchmark)"),  
  RMSE = c(res_bic["RMSE"], res_aic["RMSE"], res_naive["RMSE"]),  
  MAE = c(res_bic["MAE"], res_aic["MAE"], res_naive["MAE"]),  
  Theil_U2 = c(u_bic, u_aic, u_naive)  
)  
  
# knitr::kable(  
#   validation_table,  
#   caption = "Final ranking based on prediction error (test set)",  
#   digits = 5  
# )  
  
datatable(validation_table, options = list(dom = 't'), rownames = FALSE)
```

Model		RMSE	MAE	Theil_U2
ARMA(1,1) [BIC Winner]		0.02595539895906081	0.01907374426769277	0.9999414872296027
ARMA(3,3) [AIC Winner]		0.02595231722608405	0.01907020977949925	0.9998227623099478
Naive (Benchmark)		0.02595691777023052	0.01907580742779326	1

Figure 2.4: Comparison of out-of-sample error metrics

As highlighted in the table, the **Theil's U2 index** for both ARMA models is extremely close to 1. This indicates that the statistical models, despite their complexity, fail to significantly outperform the simple forecast based on the historical mean (Naive Benchmark).

! Important

This result is consistent with the **Efficient Market Hypothesis (EMH)**, according to which past returns contain little to no useful information for predicting future short-term returns, making the series resemble a Random Walk.

2.5 Conclusion

The analysis conducted in this chapter has highlighted the limitations of stationary linear models in forecasting the direction of ASML returns.

Although information criteria (AIC) identified the ARMA(3,3) as a mathematical structure capable of “whitening” the in-sample residuals, the *Out-of-Sample* validation delivered an unequivocal verdict: **a Theil's U index close to 1 confirms that no ARMA model systematically outperforms the Naive benchmark.** This empirical result corroborates the hypothesis that, regarding the **conditional mean**, returns follow a process closely resembling a Random Walk, rendering trading strategies based solely on past autocorrelation futile.

However, the unpredictability of the mean does not imply a total absence of structure within the series. Visual inspection of the log-returns plot reveals evident **volatility clustering**: periods of high volatility tend to be followed by high volatility, and periods of calm by calm (a phenomenon known as *conditional heteroskedasticity*).

By assuming constant variance over time (homoskedasticity), ARMA models fail to capture this fundamental characteristic of financial markets. Therefore, in the next chapter, we will shift our focus from predicting the *sign* of returns to modeling their **magnitude** (risk) by introducing the **GARCH** (Generalized AutoRegressive Conditional Heteroskedasticity) family of models.

3 Volatility analysis