TIME SERIES ANALYSIS OF SOLAR POWER

Modeling and Forecasting Short-Term Solar Power Variability with ARMA-GARCH Models

Yulin He

S4072826

CONTENT

1. Introduction **4**

2. Methods **4**

3. Results **6**

3.1 Exploratory Data Analysis (EDA) *6*

3.2 Stationarity Testing *10*

3.3 Data Transformation *11*

3.3 Model Identification – ARIMA Part *13*

3.4 Model Estimation and Residual Diagnostics – ARIMA Part *15*

3.5 Model Selection (AIC/BIC) – ARIMA Part *25*

3.6 Accuracy Evaluation -ARIMA Part *26*

3.7 Best Model – ARIMA Part *26*

3.8 Overparameterized Model: ARIMA(1, 0, 3) *26*

3.9 Model Identification – GARCH Part *28*

3.10 Model Estimation and Residual Diagnostics – ARMA(1,2)+GARCH(p,q) *32*

3.10 Model Selection (AIC/BIC) – ARIMA+GARCH *40*

4. Forecast **41**

4.1 A 10-day ahead forecast *41*

4.2 Conversion of forecasted log returen to actual values *42*

4.3 Actual Solar Power Forecast *42*

5. Discussion **43**

6. Conclusion **44**

Reference **45**

Use of Generative AI Tools in Completing the Assignment **45**

R codes **45**

MATH1318 Time Series Analysis / MATH2204 Time Series and Forecasting Final Project Report

Declaration of contributions:

No	Name of Team Member	Contribution to the project
1	Yulin He	100%
2		
3		
4		
5		
6		
	Sum:	1

1. INTRODUCTION

Solar power is a key renewable energy source in Australia’s transition to a low-carbon future. However, its generation is highly variable and weather-dependent, which poses challenges for grid stability and energy planning. Analyzing solar power generation data helps improve short-term forecasting, supports efficient grid operations, and informs long-term policy decisions.

This report presents a time series data analysis focusing on solar power generation. The dataset was sourced from the Australian Energy Market Operator (AEMO) official website. The original dataset contains over 7.3 million records collected every 4 seconds, starting from August 1st, 2019.

The goal of this project is to forecast daily solar energy output and evaluate the dynamics of its return volatility for the upcoming 15 days. Modeling volatility helps identify periods of potential instability in energy generation, which is critical for risk management. Meanwhile, accurate forecasting of daily output supports better capacity planning and resource allocation by energy providers.

2. METHODS

To conduct this anlysis, I utilized Rstudio along with TSA, fUnitRoots, tseries and forecast packages. The methodology followed five key steps:

(1) Exploratory Data Analysis (EDA)

Since the original dataset was too large to process efficiently in R, I first aggregated the solar power readings into daily totals, converting units from kilowatts (kW) to kilowatt-hours (kWh). This resulted in a manageable time series of 343 daily observations, which was then structured as a time series object for further analysis.

(2) Stationarity Testing

To evaluate the stationarity of the daily solar output series, I applied several tests: ACF, PACF, Augmented Dickey-Fuller (ADF), Phillips-Perron (PP), and KPSS test. These tests revealed both trend and heteroskedasticity in the series. To stabilize the mean and variance, I applied a log transformation followed by first differencing, resulting in a stationary return series suitable for ARMA-GARCH modeling.

(3) Model Identification

Using ACF-PACF plot, and the McLeod-Li test, I identified potential ARMA models for the mean structure and assessed conditional heteroskedasticity. To account for time-varying variance, I applied an ARMA+GARCH framework to model both autocorrelation and volatility clustering in the series.

(4) Model Estimation

The ARMA order was specified first, and residuals were then analyzed to determine the GARCH order. Using Extended Autocorrelation Function (EACF) table and Bayesian Information Criterion (BIC) to propose a set of candidate ARIMA and GARCH models based on the observed autocorrelation patterns. Candidate ARMA and GARCH models were estimated using both Maximum Likelihood (ML) and Conditional Sum of Squares (CSS) methods.

(5) Model Selection

Model performance was evaluated based on both the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) values. Models with the lowest AIC and BIC values were selected as the best-fitting candidates.

To futher assess model performance, I used the forecast package to calculate additional accuracy measures, including Mean Error (ME), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Percentage Error (MPE), Mean Absolute Percentage Error (MAPE), Mean Absolute Scaled Error (MASE), and the first-lag autocorrelation of residuals (ACF1).

(6) Model Diagnostics

For each candidate ARMA-GARCH model, I conducted a comprehensive residual analysis. Several diagnostic plots were generated:

a. The time series plot of standardized residuals shows random fluctuations around zero, with no visible trend or structural change.

b. The histogram of residuals appears roughly symmetric but slightly deviates from the ideal bell shape.

c. The Q-Q plot indicates heavy tails on both ends, suggesting that the residuals deviate from a normal distribution, especially in the extremes.

d. The ACF and PACF plots of residuals show that most autocorrelations fall within the 95% confidence bounds, indicating no strong residual autocorrelation.

e. The Ljung-Box test p-value plot further confirms this. Most p-values remain above the 0.05 threshold across lags, indicating that the residuals behave like white noise. The Ljung-Box test is used to check whether the residuals from a time series model are independently distributed. Specifically, it tests the null hypothesis that the residuals are not autocorrelated up to a certain lag. If the p-values are consistently above 0.05, we fail to reject the null hypothesis, suggesting that the model has adequately captured the time-dependent structure in the data.

(7) Forecasting

Once the best-fitting model was selected, I generated 15-step-ahead forecasts for the daily log returns of solar energy output. To convert the forecasted log returns from the GARCH model back into actual electricity generation values (in kWh), I applied an exponential transformation and reconstructed the predicted output based on the last observed actual value. This allowed me to obtain both the expected future production and forecast intervals that reflect estimated volatility—providing useful information for both capacity planning and risk management in solar power operations.

3. RESULTS

3.1 Exploratory Data Analysis (EDA)

The time series plot in **Figure 1** reveals the solar power generation pattern over a single day with bell-shaped curve. Power output remians at zero during the early and late hours, rises sharply in the morning, stays relatively stable at peak levels around midday, and then drops off in the late afternoon. Minor high-frequency fluctuations observed around the peak period may be attributed to transient environmental factors such as cloud cover or inverter response delays.

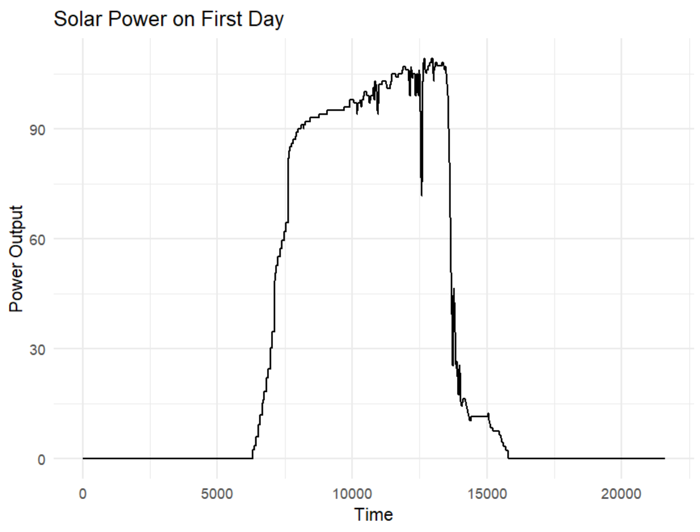


Figure 1. Solar Power Output on the First Day

After aggregating the solar power readings into daily totals and converting the units from kilowatts (kW) to kilowatt-hours (kWh), the resulting time series (**Figure 2)** exhibits 343 daily solar energy output from August 2019 to July 2020, ranging approximately from 200 kWh to 1200 kWh. The following five features are evident:

Trend: A clear rise-and-fall pattern is observed. Output increases from August to December 2019, then declines gradually into mid-2020. This likely reflects seasonal solar irradiance changes, with higher output during the summer months and lower in winter.

Seasonality: While the time span covers only one year, it’s not possible to determine whether this is a consistent long-term trend or simply a seasonal fluctuation.

Variability: The volatility is not constant over time. Fluctuations are narrower in late 2019 but become more erratic and extreme in early to mid-2020, indicating potential time-varying variance.

Autoregressive Behavior: The series shows strong temporal dependence. Daily values appear closely related to those of preceding days, suggesting autoregressive dynamics.

Change Point: There is no abrupt structal break in the series, but the variaility increases significantly after March 2020, which may imply a latent change in system dynamic or external influences.

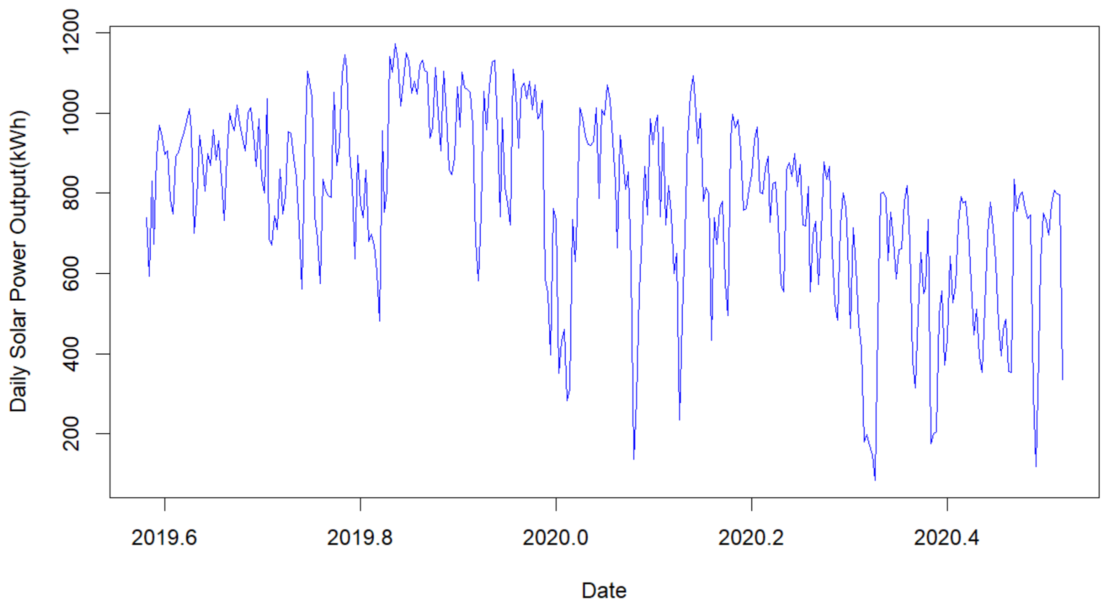


Figure 2. Daily Total Solar Power Outpout (kWh).

The summary statistics shows that daily solar output ranges from 85.66 to 1173.21 kWh, with a median of 802.28 kWh and a mean of 781.78 kWh. The standard devidation is 226.76.

```
> print(summary_stats)
       Min     Max     Mean   Median       SD
1 85.65778 1173.21 781.7845 802.2768 226.7553
```

```
> head(ts_daily)
Time Series:
Start = c(2019, 213)
End = c(2019, 218)
Frequency = 365
[1] 739.5803 592.2064 829.3783
[4] 671.8799 891.0257 969.0506
```

The scatter plot (**Figure 3**) shows a strong positive relationship between daily solar energy outputs on consecutive days, with a lag-1 correlation coefficient of **0.763**, indicating substantial autocorrelation.

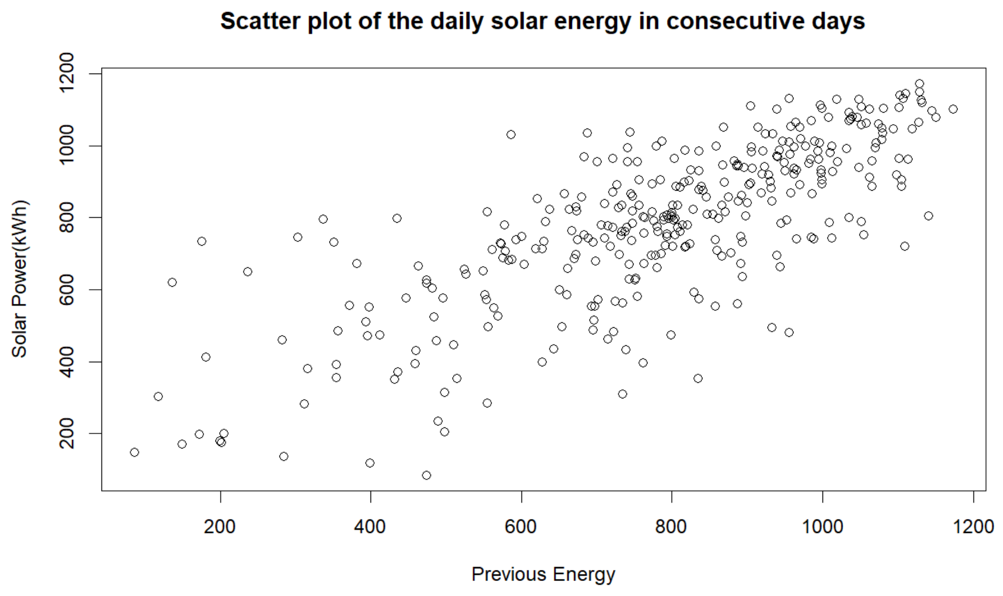


Figure 3. Scatter plot of daily solar power in consequtive days.

The Q-Q plot (**Figure 4**) exhibits heavy tails at both ends, indicating deviations from normality. This visual assessment is supported by the Shapiro-Wilk test, which returns a p-value close to zero ($p < 0.001$), providing strong evidence that the daily solar power data do not follow a normal distribution.

```
> shapiro.test(ts_daily)

	Shapiro-Wilk normality test

data:  ts_daily
W = 0.95623, p-value = 1.383e-08
```

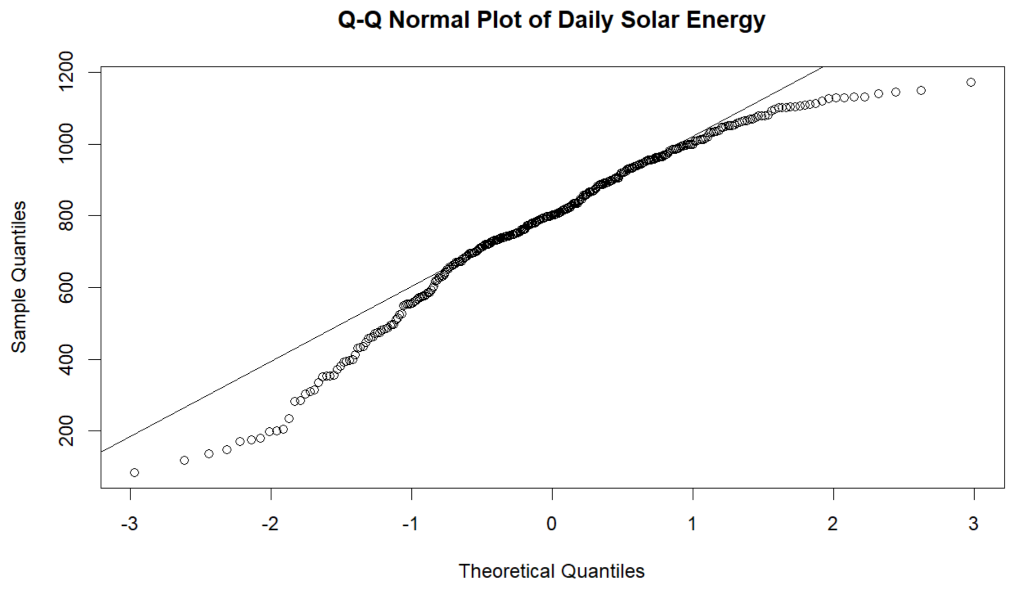
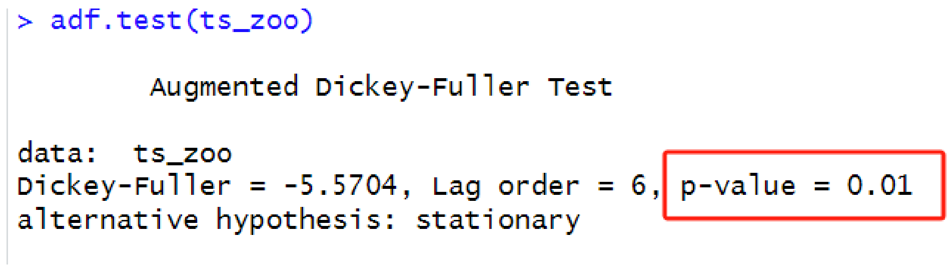


Figure 4. Q-Q plot of daily solar power.

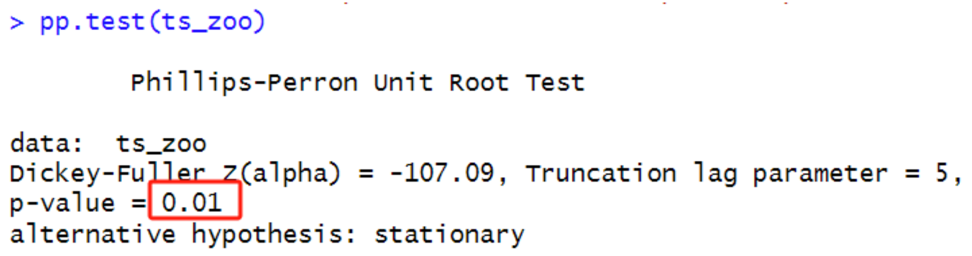
3.2 Stationarity Testing

To evaluate the stationarity of the time series, I conducted two unit root tests: the ADF test and the PP test. The null hypothesis for both tests assumes that "H_0: The process is difference non-stationary (the process is non-stationary but becomes stationary after first differencing), while the alternative hypothesis H_A: The process is stationary."

Both tests returned a p-value of 0.01, allowing us to reject the null hypothesis at the 5% significant level and conclude that the time series is stationary.



```
> adf.test(ts_zoo)

        Augmented Dickey-Fuller Test

data:  ts_zoo
Dickey-Fuller = -5.5704, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary
```



```
> pp.test(ts_zoo)

        Phillips-Perron Unit Root Test

data:  ts_zoo
Dickey-Fuller Z(alpha) = -107.09, Truncation lag parameter = 5,
p-value = 0.01
alternative hypothesis: stationary
```

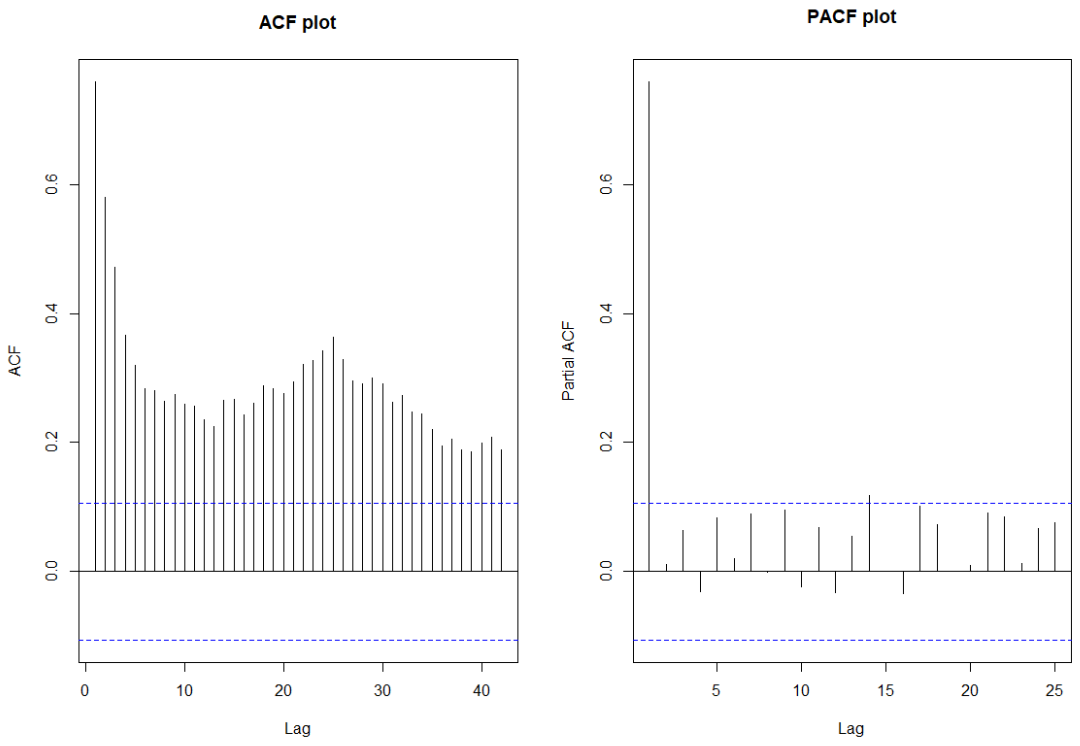


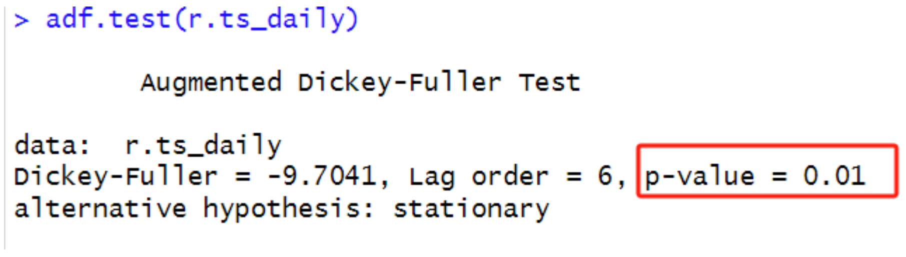
Figure 5. ACF and PACF plot of daily solar power.

To further investigate the time series structure and guide model selection, we examined the ACF and PACF plots. **Figure 5** shwos that the ACF decays slowly, indicating strong autocorrelation and suggesting the presence of long-term dependencies or a remaining trend component. The PACF cuts off quickly after lag 1, which is typical of an autoregressive (AR) process.

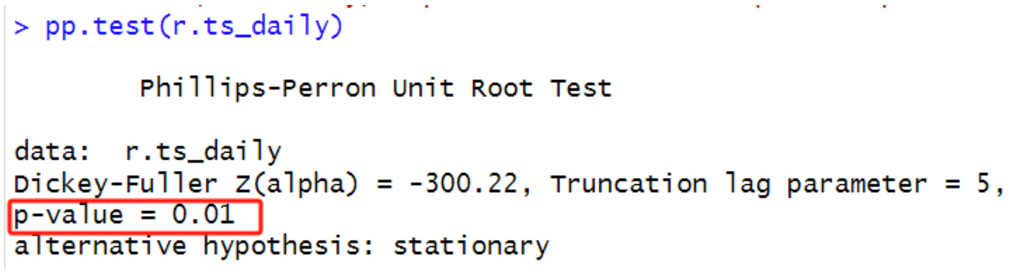
In addition, the original time series exhibits clear volatility clustering, implying that the variance changes over time. Taken together, these findings confirm that while the series is stationarity, it displays both temporal dependence and heteroskedasticity. Therefore, an **ARMA-GARCH** model is considered appropriated to simultaneously model the autocorrelation in the mean and the conditional heteroskedasticity in the variance.

3.3 Data Transformation

To stabilize variance and prepare the data for time series modeling, a log-differencing transformation was applied to the original daily solar output series. This process converts the level data into log returns, making the series more suitable for ARMA and GARCH models, which assume both stationarity and constant variance in their underlying assumptions.



```
> adf.test(r.ts_daily)

        Augmented Dickey-Fuller Test

data:  r.ts_daily
Dickey-Fuller = -9.7041, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary
```



```
> pp.test(r.ts_daily)

        Phillips-Perron Unit Root Test

data:  r.ts_daily
Dickey-Fuller Z(alpha) = -300.22, Truncation lag parameter = 5,
p-value = 0.01
alternative hypothesis: stationary
```

Although the original series was already stationary, we performed unit root testing again to validate the stationarity of the transformed series. Both the ADF and PP test returned p-values of 0.01, confirming that the log-differenced series remains stationary and is appropriate for subsequent modeling.

As shown in **Figure 6**, the transformed return series fluctuates around a constant mean, with visible volatility clustering—periods of low and high variation tend to alternate over time, further supporting the use of GARCH-type models.

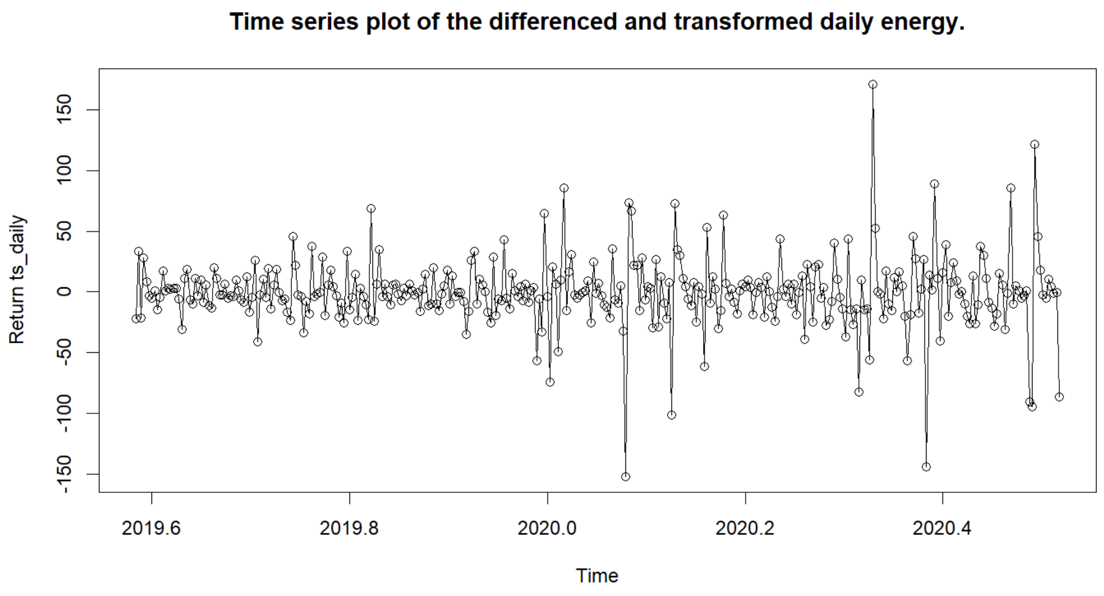


Figure 6. Time series plot after applying log-differenced transformation of daily solar power.

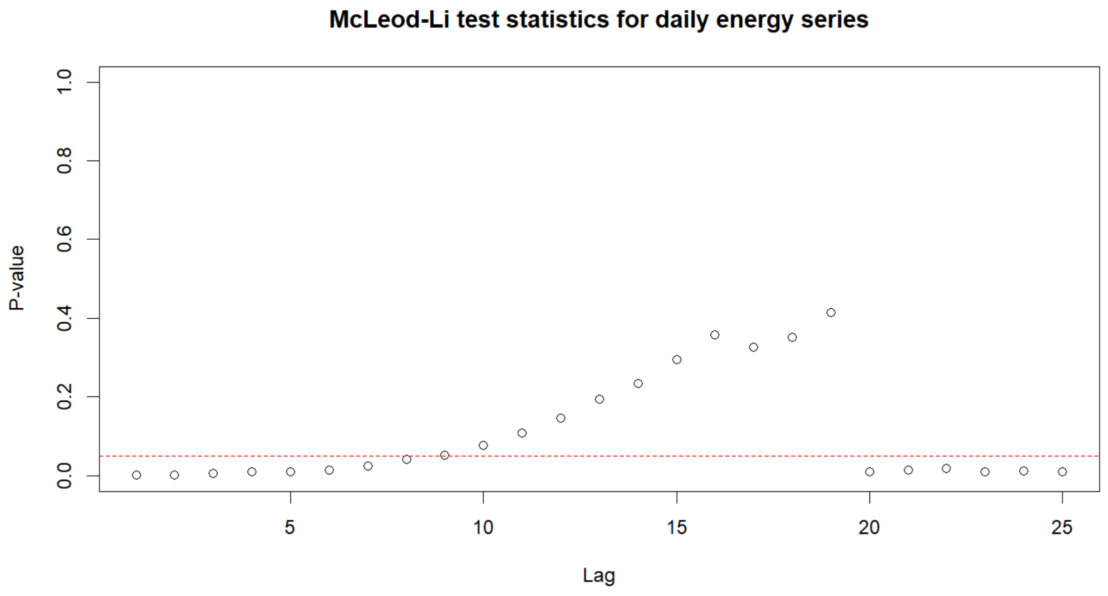


Figure 7. McLeod.Li.test to the squared residuals of the energy return series.

To check whether there's evidence of conditional heteroskedasticity, **I applied the McLeod-Li test** to the squared residuals of the energy return series. From the plot in **Figure 7**, several p-values fall below the 0.05 significance threshold, especially at early lags. That suggests the presence of ARCH effects in the data, which supports the use of GARCH-type models.

```
> shapiro.test(r.ts_daily)

	Shapiro-Wilk normality test

data:  r.ts_daily
W = 0.86243, p-value < 2.2e-16
```

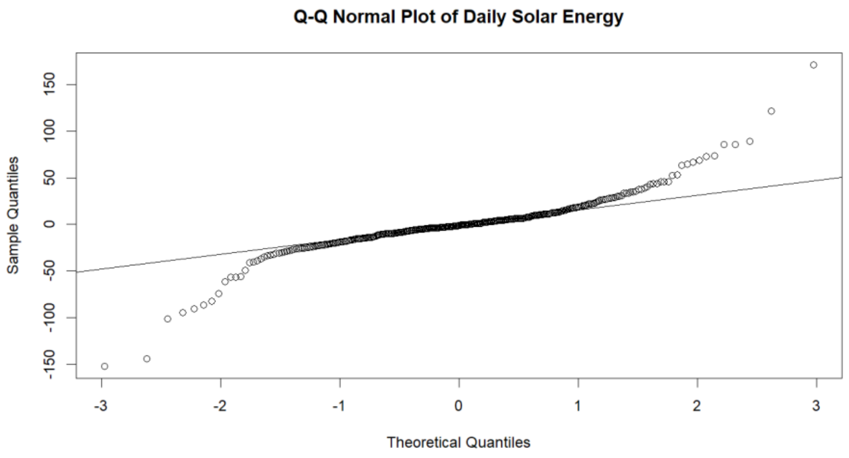


Figure 8. Q-Q plot of log-differenced transformation of daily solar power

Despite the log-differenced transformation, the series remains non-normal according to the Q-Q plot (**Figure 8**) and the Shapiro-Wilk test (p-value < 0.001), which may affect model assumptions and should be accounted for when estimating and interpreting volatility models.

3.3 Model Identification – ARIMA Part

In this study, model specification tools including ACF, PACF, EACF (extended autocorrelation function) to guide the preliminary identification of candidate ARIMA models. After selecting suitable ARMA orders, I then examined the standardized residuals of the fitted ARMA models to determine appropriate GARCH specifications, in line with the two-step modeling strategy typically used in ARMA-GARCH frameworks.

(1) ACF and PACF Plots

The model selection process began by identifying the appropriate ARMA orders, using the ACF-PACF patterns of the log-differenced return series. As shown in **Figure 9**, the PACF plot shows a long tail of significant lags, while the ACF decays quickly after lag 1 suggesting the presence of a MR component of order 1.

Since a log-differencing transformation was applied to the original time series to stabilize variance and remove trend, this transformation effectively incorporates one order of differencing ($d = 1$). Therefore, when fitting an ARIMA(p, d, q) model to the transformed data, the differencing component can be omitted, and the model can be specified as an ARMA(p, q) process, i.e., ARIMA with d=0. So, reasonable candidate models include: **{ARMA(1,1), ARMA(2,1), ARMA(3,1), ARMA(4,1)}**

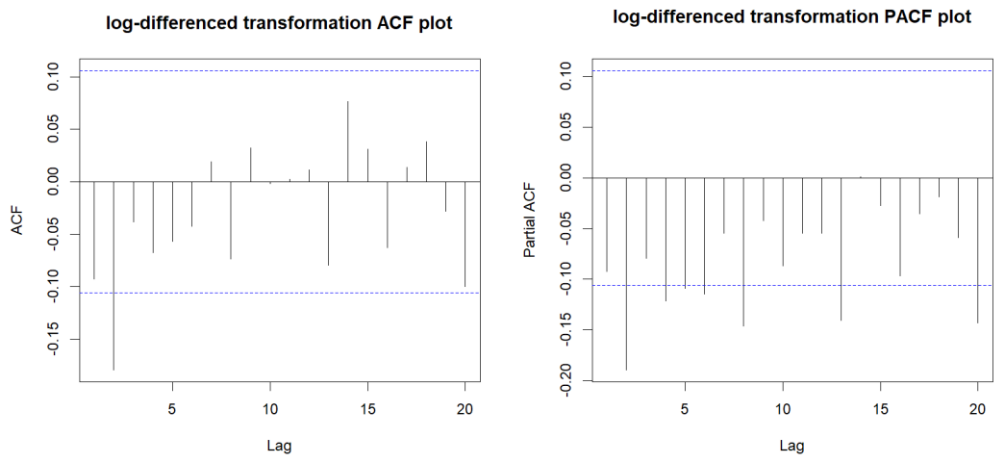


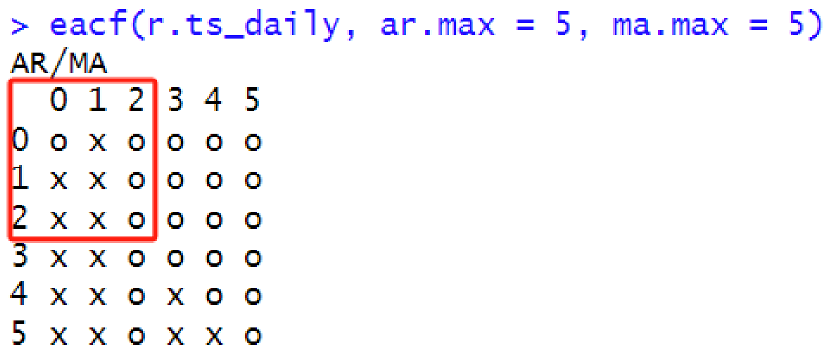
Figure 9. ACF and PACF plot of the data after applying log- differenced transformation.

(2) EACF

The EACF was used to assist in model identification. Giving the size of the series, the maximum number of AR and MR parameters was restricted to five. From the EACF table (**Figure 10**), the top-left 'o' symbol appears at the intersections of AR = 0 and MA = 2. Then following the vertex downward, additional 'o' symbols are AR = 1 and AR = 2.

Based on this EACF structure, the set of candidate models includes:

{ARMA (0, 2) ARIMA (1, 2) ARIMA (2, 2)}



```
> eacf(r.ts_daily, ar.max = 5, ma.max = 5)
AR/MA
  0 1 2 3 4 5
0 o x o o o o
1 x x o o o o
2 x x o o o o
3 x x o o o o
4 x x o x o o
5 x x o x x o
```

Figure 10. EACF table of ARIMA

(3) BIC table

All models within the specified range of AR and MA orders were fitted to the series and residual analysis was conducted using the Bayesian Information Criterion (BIC) table. The relative magnitude of the BIC values is represented by color intensity. Darker shades indicate lower BIC values, suggesting better model performance. Based on **Figure 11**, I selected the top three rows with the lowest BIC values, identifying the candidate models as:

{ARMA (0, 1), ARIMA (0, 2), ARIMA (2, 1), ARIMA(2,2), ARIMA(2,5)}.

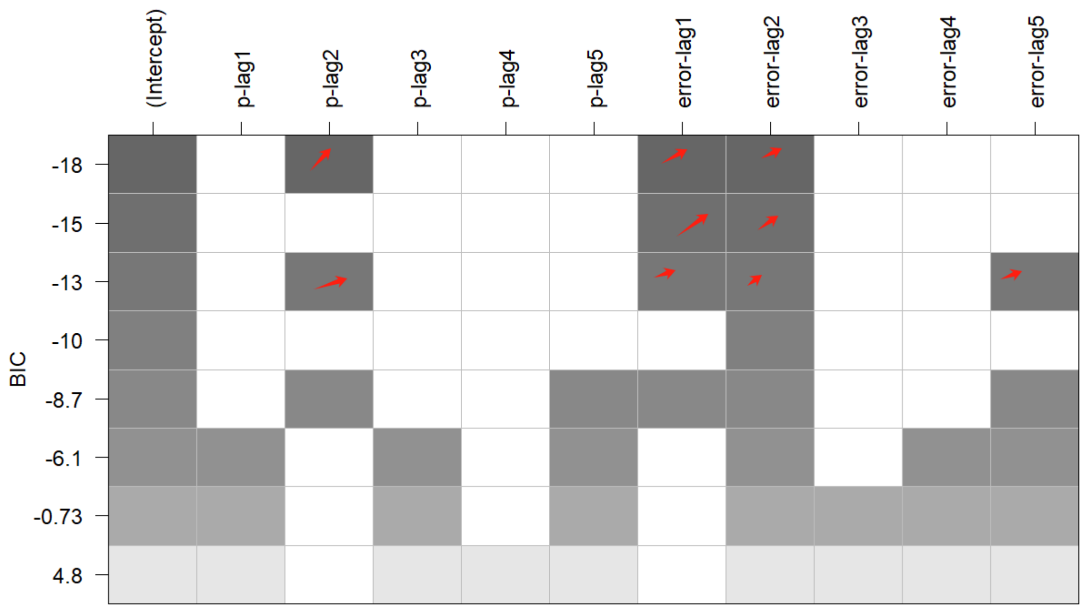


Figure 11. Bayesian Information Criterion (BIC) table.

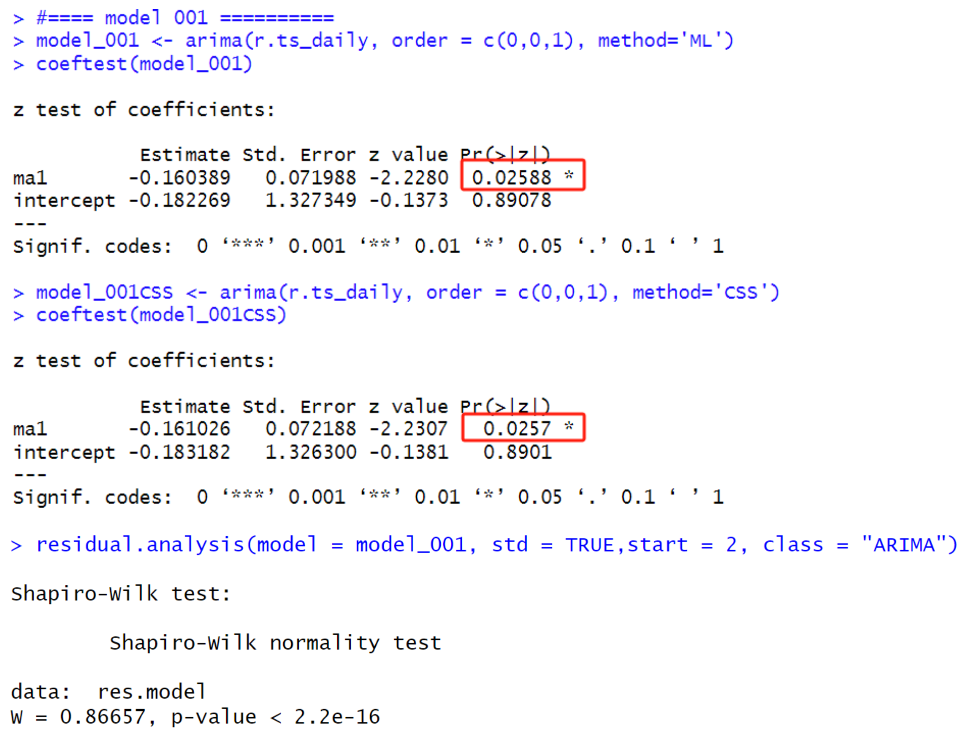
Overall set of possible nine models becomes: **{ARMA(0, 1), ARMA(0, 2), ARMA(1, 1), ARMA(1, 2), ARMA(2, 1), ARMA(2, 2), ARMA(2, 5), ARMA(3, 1), ARMA(4, 1) }.**

3.4 Model Estimation and Residual Diagnostics – ARIMA Part

I used **log-differenced transformated data** to fit ARIMA models and performed parameter estimation and siginificance testing. Both Conditional Sum of Squares (CSS) and Maximum Likelihood (ML) approaches were applied to ensure robustness of the model fitting process.

- **ARMA(0, 1)**

The estimated MA(1) coefficient is statistically significant at the 5% level under both ML and CSS estimation methods (p-values ≈ 0.025), indicating that the moving average component is an important contributor to the model. The residual diagnostics in **Figure 12** suggest that the model is not fully adequate. While the residuals appear roughly centered around zero, the Ljung-Box test p-values fall below the 0.05 threshold at many lags, indicating that autocorrelation remains in the residuals.



```
> #==== model 001 ==========
> model_001 <- arima(r.ts_daily, order = c(0,0,1), method='ML')
> coeftest(model_001)

z test of coefficients:

           Estimate Std. Error z value Pr(>|z|)
ma1       -0.160389   0.071988 -2.2280  0.02588 *
intercept -0.182269   1.327349 -0.1373  0.89078
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> model_001css <- arima(r.ts_daily, order = c(0,0,1), method='CSS')
> coeftest(model_001css)

z test of coefficients:

           Estimate Std. Error z value Pr(>|z|)
ma1       -0.161026   0.072188 -2.2307   0.0257 *
intercept -0.183182   1.326300 -0.1381   0.8901
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> residual.analysis(model = model_001, std = TRUE,start = 2, class = "ARIMA")

Shapiro-Wilk test:

        Shapiro-Wilk normality test

data:  res.model
W = 0.86657, p-value < 2.2e-16
```

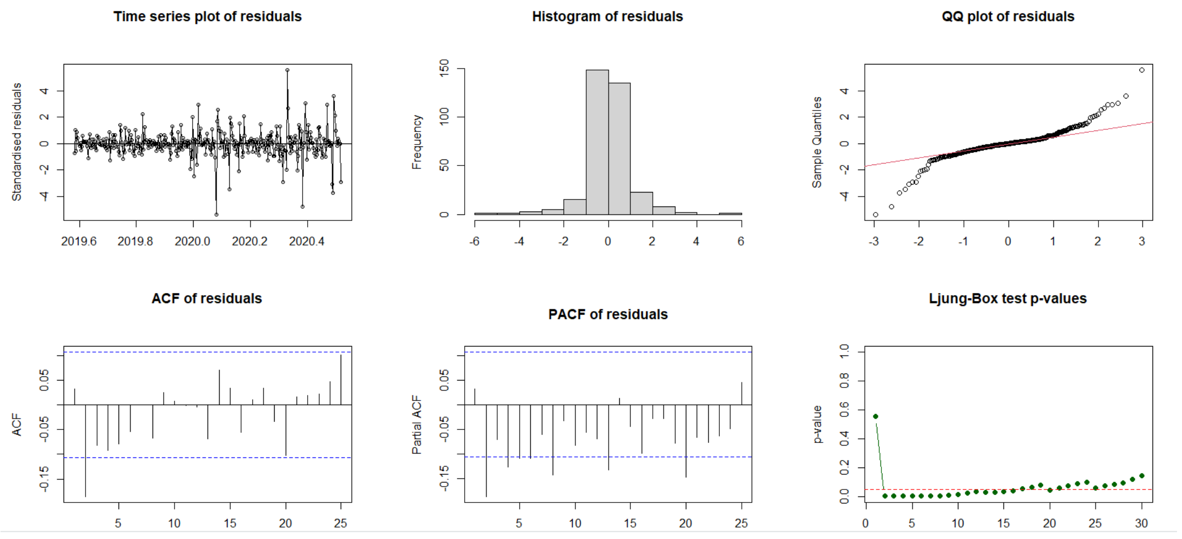
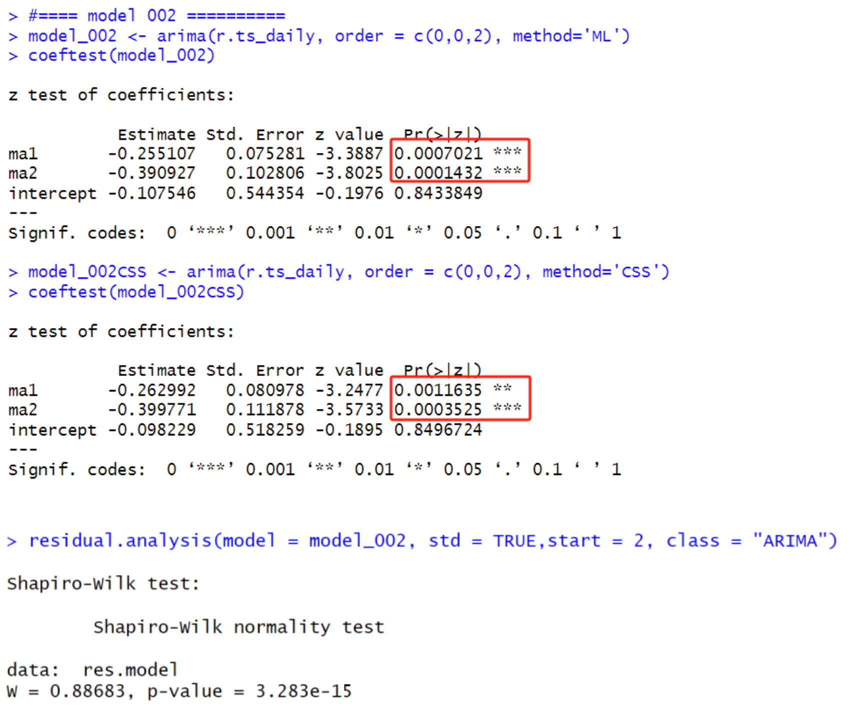


Figure 12. Residual Diagnostics of ARMA(0, 1)

- **ARMA(0, 2)**

The ARMA(0,2) model yields two highly significant MA terms, with p-values < 0.01 for both MA(1) and MA(2) under both ML and CSS estimation methods. This indicates a strong moving average structure in the return series.

In **Figure 13**, while the standardized residuals fluctuate around zero, the Q-Q plot indicates non-normality, and most Ljung-Box test p-values fall below the 0.05 threshold. This suggests that the residuals still exhibit autocorrelation, and the model has not fully captured the temporal dependence in the data.



```
> #==== model 002 ==========
> model_002 <- arima(r.ts_daily, order = c(0,0,2), method='ML')
> coeftest(model_002)

z test of coefficients:

           Estimate Std. Error z value  Pr(>|z|)    
ma1       -0.255107   0.075281 -3.3887 0.0007021 ***
ma2       -0.390927   0.102806 -3.8025 0.0001432 ***
intercept -0.107546   0.544354 -0.1976 0.8433849    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> model_002CSS <- arima(r.ts_daily, order = c(0,0,2), method='CSS')
> coeftest(model_002CSS)

z test of coefficients:

           Estimate Std. Error z value  Pr(>|z|)    
ma1       -0.262992   0.080978 -3.2477 0.0011635 ** 
ma2       -0.399771   0.111878 -3.5733 0.0003525 ***
intercept -0.098229   0.518259 -0.1895 0.8496724    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


> residual.analysis(model = model_002, std = TRUE,start = 2, class = "ARIMA")

Shapiro-Wilk test:

        Shapiro-Wilk normality test

data:  res.model
W = 0.88683, p-value = 3.283e-15
```

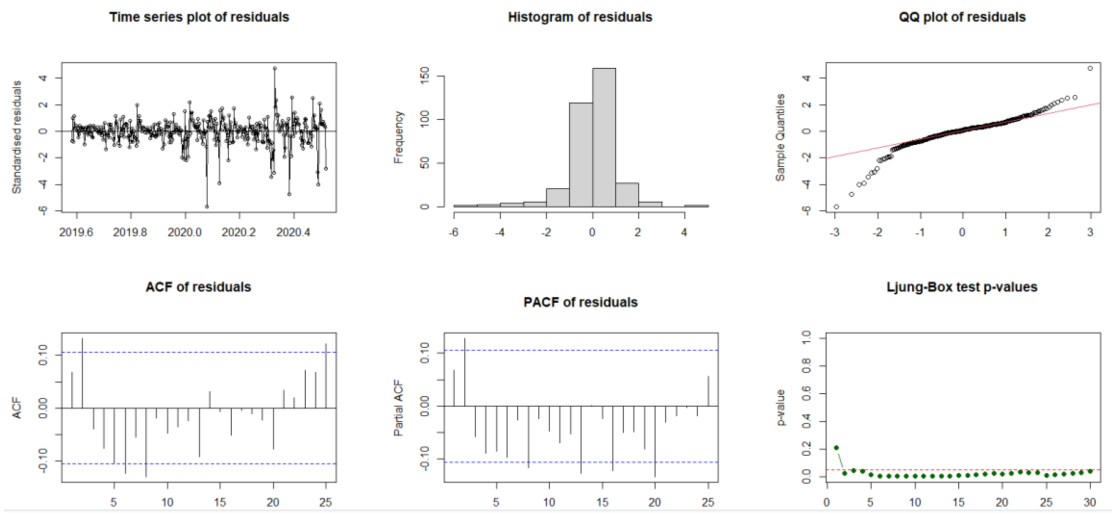
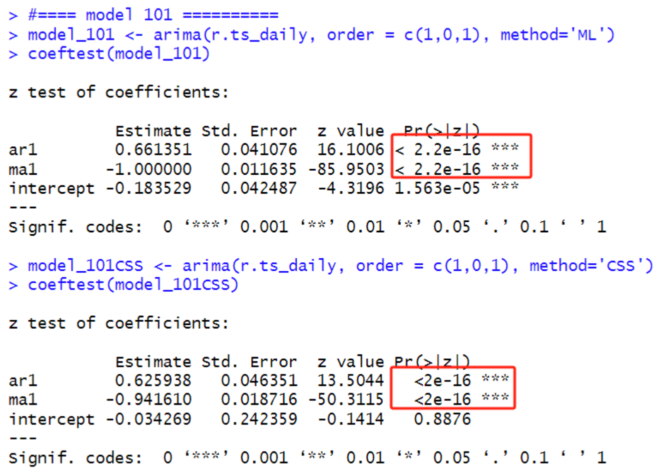


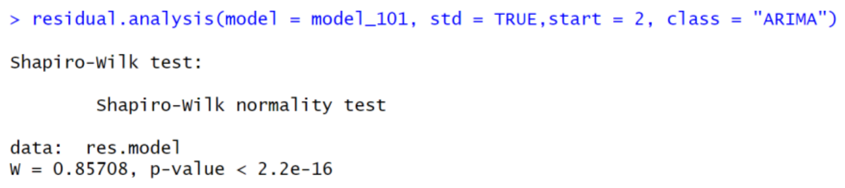
Figure 13. Residual Diagnostics of ARMA(0, 2)

- **ARMA(1, 1)**

The **ARMA(1,1)** model yields highly significant parameter estimates for both the AR(1) and MA(1) terms under both ML and CSS estimation methods (p-values < 2e-16). As shown in **Figure 14**, the residual diagnostics support the model adequacy. The standardized residuals fluctuate randomly around zero, and the ACF/PACF plots show no strong autocorrelation. Most importantly, the Ljung-Box test p-values remain consistently above the 0.05 threshold across lags, suggesting that the residuals resemble white noise. This indicates that the ARMA(1,1) model provides a well-specified and statistically sound fit to the transformed time series.



```
> #==== model 101 ==========
> model_101 <- arima(r.ts_daily, order = c(1,0,1), method='ML')
> coeftest(model_101)

z test of coefficients:

           Estimate Std. Error  z value  Pr(>|z|)    
ar1        0.661351   0.041076  16.1006 < 2.2e-16 ***
ma1       -1.000000   0.011635 -85.9503 < 2.2e-16 ***
intercept -0.183529   0.042487  -4.3196 1.563e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> model_101CSS <- arima(r.ts_daily, order = c(1,0,1), method='CSS')
> coeftest(model_101CSS)

z test of coefficients:

           Estimate Std. Error  z value Pr(>|z|)    
ar1        0.625938   0.046351  13.5044   <2e-16 ***
ma1       -0.941610   0.018716 -50.3115   <2e-16 ***
intercept -0.034269   0.242359  -0.1414   0.8876    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```



```
> residual.analysis(model = model_101, std = TRUE,start = 2, class = "ARIMA")

Shapiro-Wilk test:

        Shapiro-Wilk normality test

data:  res.model
W = 0.85708, p-value < 2.2e-16
```

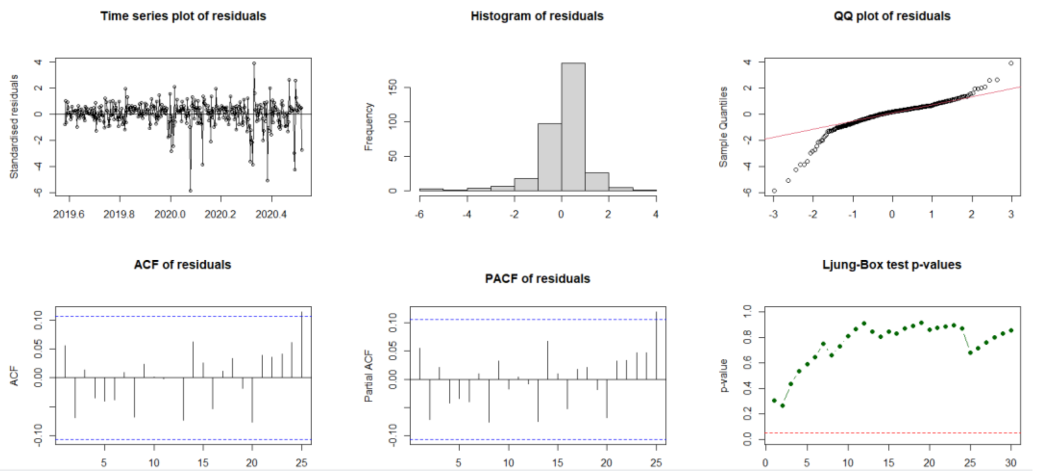
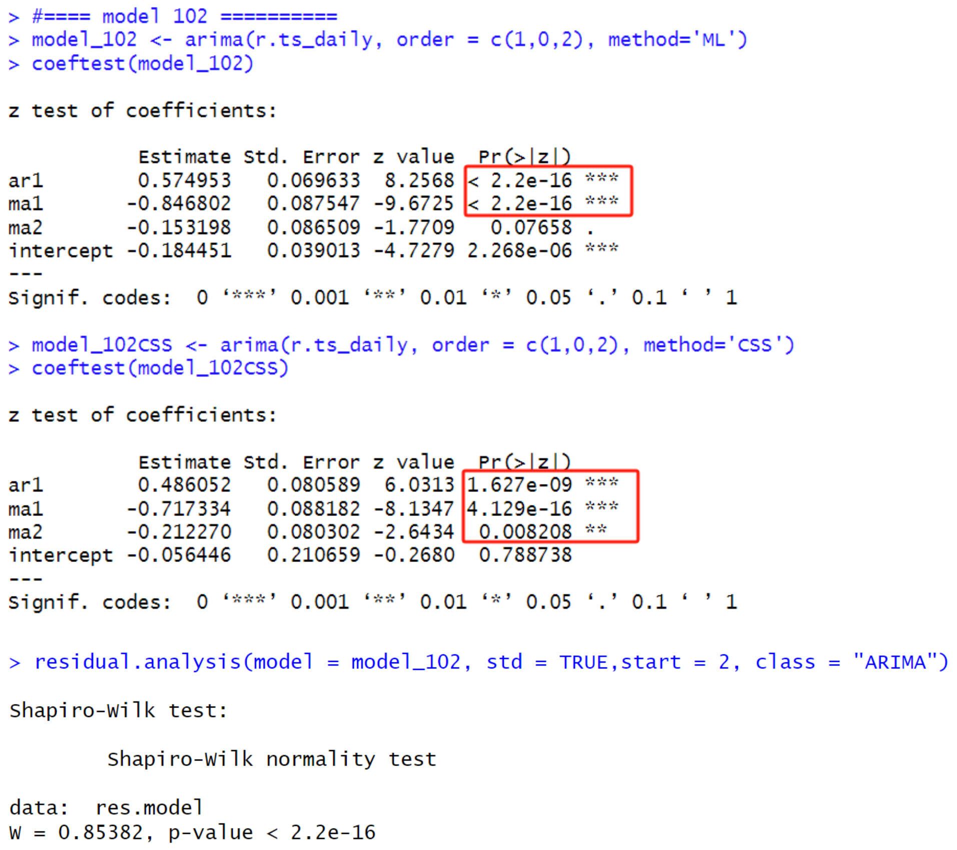


Figure 14. Residual Diagnostics of ARMA(1, 1)

- **ARMA(1, 2)**

The ARMA(1,2) model produces results similar to ARMA(1,1), with **significant AR(1) and MA(1) terms**, and mixed significance for MA(2). Residual diagnostics (**Figure 15**) show no remaining autocorrelation and Ljung-Box p-values are consistently above 0.05



```
> #==== model 102 ==========
> model_102 <- arima(r.ts_daily, order = c(1,0,2), method='ML')
> coeftest(model_102)

z test of coefficients:

           Estimate Std. Error z value  Pr(>|z|)    
ar1        0.574953   0.069633  8.2568 < 2.2e-16 ***
ma1       -0.846802   0.087547 -9.6725 < 2.2e-16 ***
ma2       -0.153198   0.086509 -1.7709   0.07658 .  
intercept -0.184451   0.039013 -4.7279 2.268e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> model_102css <- arima(r.ts_daily, order = c(1,0,2), method='CSS')
> coeftest(model_102css)

z test of coefficients:

           Estimate Std. Error z value  Pr(>|z|)    
ar1        0.486052   0.080589  6.0313 1.627e-09 ***
ma1       -0.717334   0.088182 -8.1347 4.129e-16 ***
ma2       -0.212270   0.080302 -2.6434  0.008208 ** 
intercept -0.056446   0.210659 -0.2680  0.788738    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> residual.analysis(model = model_102, std = TRUE,start = 2, class = "ARIMA")

Shapiro-Wilk test:

        Shapiro-Wilk normality test

data:  res.model
W = 0.85382, p-value < 2.2e-16
```

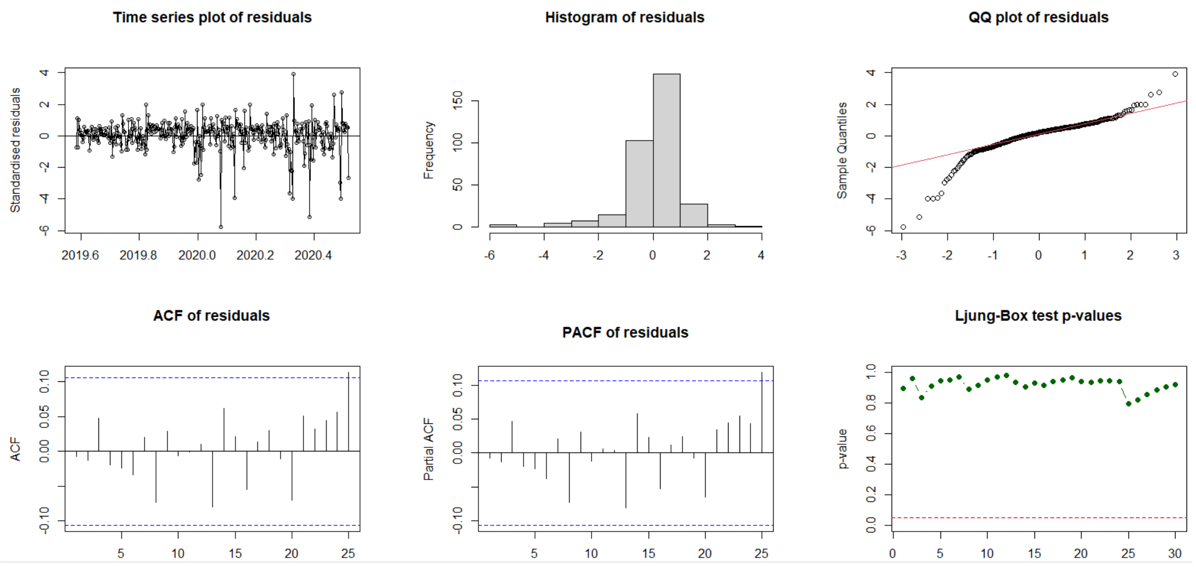
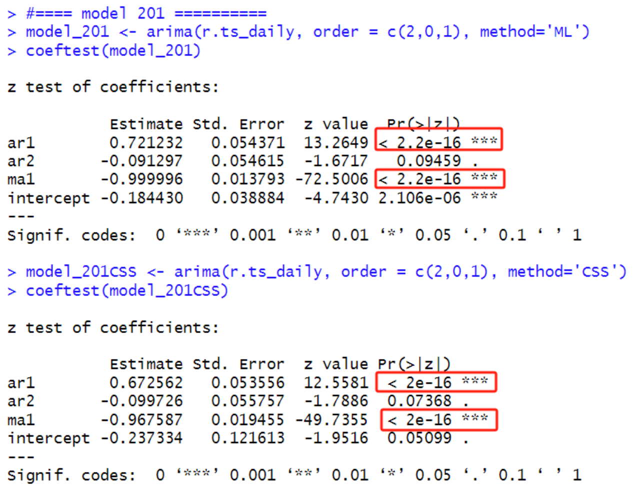


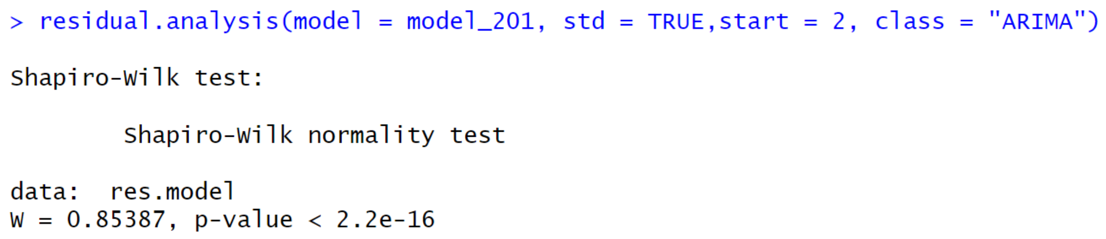
Figure 15. Residual Diagnostics of ARMA(1, 2)

- **ARMA(2, 1)**

The ARMA(2,1) model yields significant AR(1) and MA(1) terms, while the AR(2) term is not statistically significant (p > 0.05) under both ML and CSS estimation. Residual diagnostics (**Figure 16**) show no notable autocorrelation, and Ljung-Box p-values remain consistently above 0.05 across lags, indicating that the residuals resemble white noise. Overall, the model fits the data well, but the added AR(2) term may be unnecessary.



```
> #==== model 201 ==========
> model_201 <- arima(r.ts_daily, order = c(2,0,1), method='ML')
> coeftest(model_201)

z test of coefficients:

           Estimate Std. Error  z value  Pr(>|z|)    
ar1        0.721232   0.054371  13.2649 < 2.2e-16 ***
ar2       -0.091297   0.054615  -1.6717   0.09459 .  
ma1       -0.999996   0.013793 -72.5006 < 2.2e-16 ***
intercept -0.184430   0.038884  -4.7430 2.106e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> model_201css <- arima(r.ts_daily, order = c(2,0,1), method='CSS')
> coeftest(model_201css)

z test of coefficients:

           Estimate Std. Error  z value Pr(>|z|)    
ar1        0.672562   0.053556  12.5581  < 2e-16 ***
ar2       -0.099726   0.055757  -1.7886  0.07368 .  
ma1       -0.967587   0.019455 -49.7355  < 2e-16 ***
intercept -0.237334   0.121613  -1.9516  0.05099 .  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```



```
> residual.analysis(model = model_201, std = TRUE,start = 2, class = "ARIMA")

Shapiro-Wilk test:

        Shapiro-Wilk normality test

data:  res.model
W = 0.85387, p-value < 2.2e-16
```

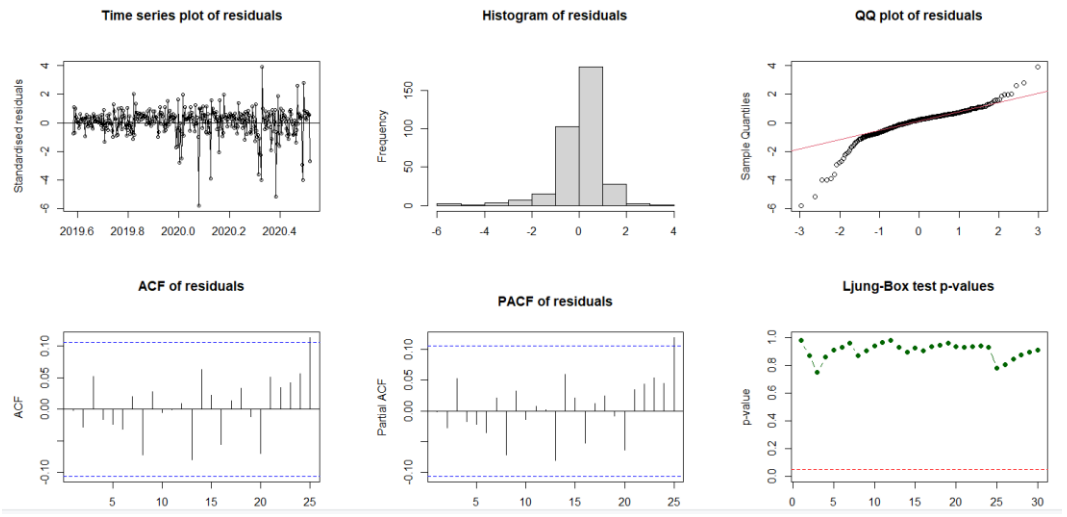
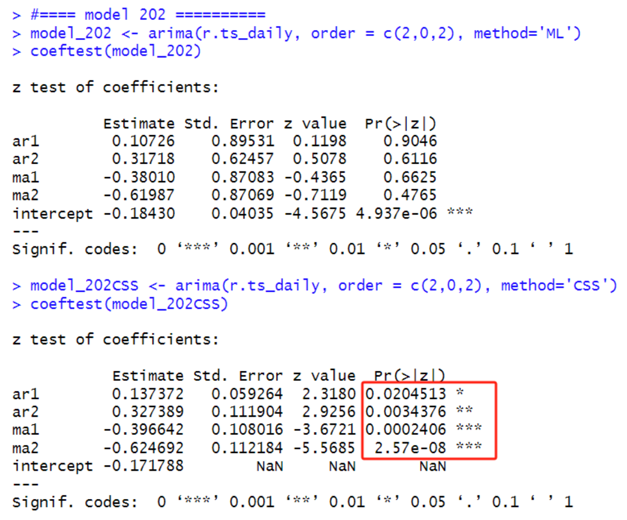


Figure 16. Residual Diagnostics of ARMA(2, 1)

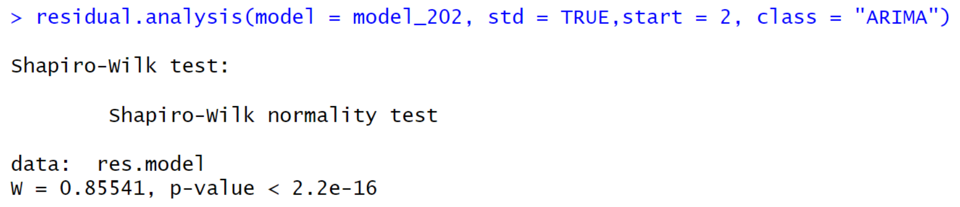
- **ARMA(2, 2)**

The ARMA(2,2) model shows mixed results. Under ML estimation, none of the parameters are significant, while under CSS estimation, all coefficients are statistically significant ($p < 0.05$).

Residual diagnostics (**Figure 17**) indicate no major autocorrelation, and Ljung-Box p-values remain well above 0.05 across lags, suggesting white-noise residuals. However, due to inconsistent significance across methods, the model’s stability may be questionable compared to simpler alternatives like ARMA(1,1) or ARMA(1,2).



```
> #==== model 202 ==========
> model_202 <- arima(r.ts_daily, order = c(2,0,2), method='ML')
> coeftest(model_202)

z test of coefficients:

          Estimate Std. Error z value  Pr(>|z|)
ar1        0.10726    0.89531  0.1198    0.9046
ar2        0.31718    0.62457  0.5078    0.6116
ma1       -0.38010    0.87083 -0.4365    0.6625
ma2       -0.61987    0.87069 -0.7119    0.4765
intercept -0.18430    0.04035 -4.5675 4.937e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> model_202css <- arima(r.ts_daily, order = c(2,0,2), method='CSS')
> coeftest(model_202css)

z test of coefficients:

           Estimate Std. Error z value  Pr(>|z|)
ar1        0.137372   0.059264  2.3180 0.0204513 *
ar2        0.327389   0.111904  2.9256 0.0034376 **
ma1       -0.396642   0.108016 -3.6721 0.0002406 ***
ma2       -0.624692   0.112184 -5.5685  2.57e-08 ***
intercept -0.171788        NaN     NaN       NaN
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```



```
> residual.analysis(model = model_202, std = TRUE,start = 2, class = "ARIMA")

Shapiro-Wilk test:

	Shapiro-Wilk normality test

data:  res.model
W = 0.85541, p-value < 2.2e-16
```

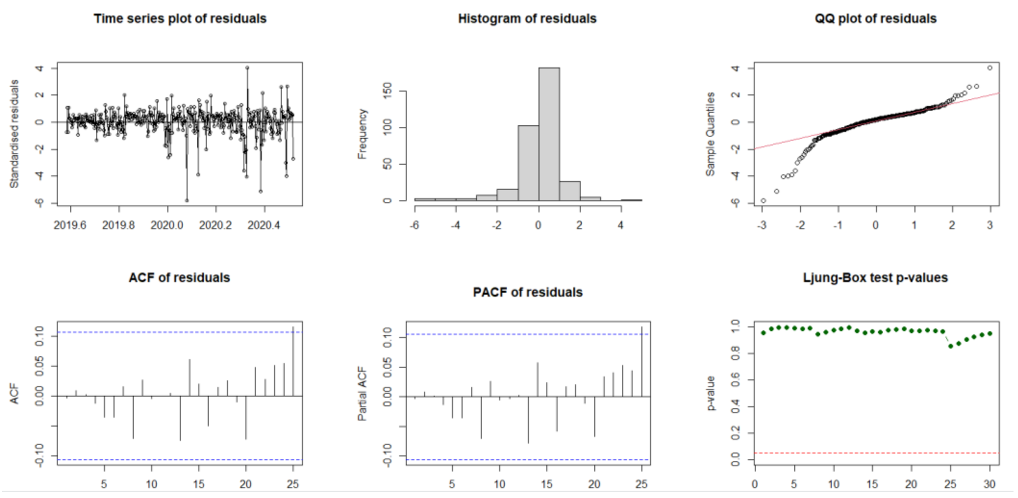
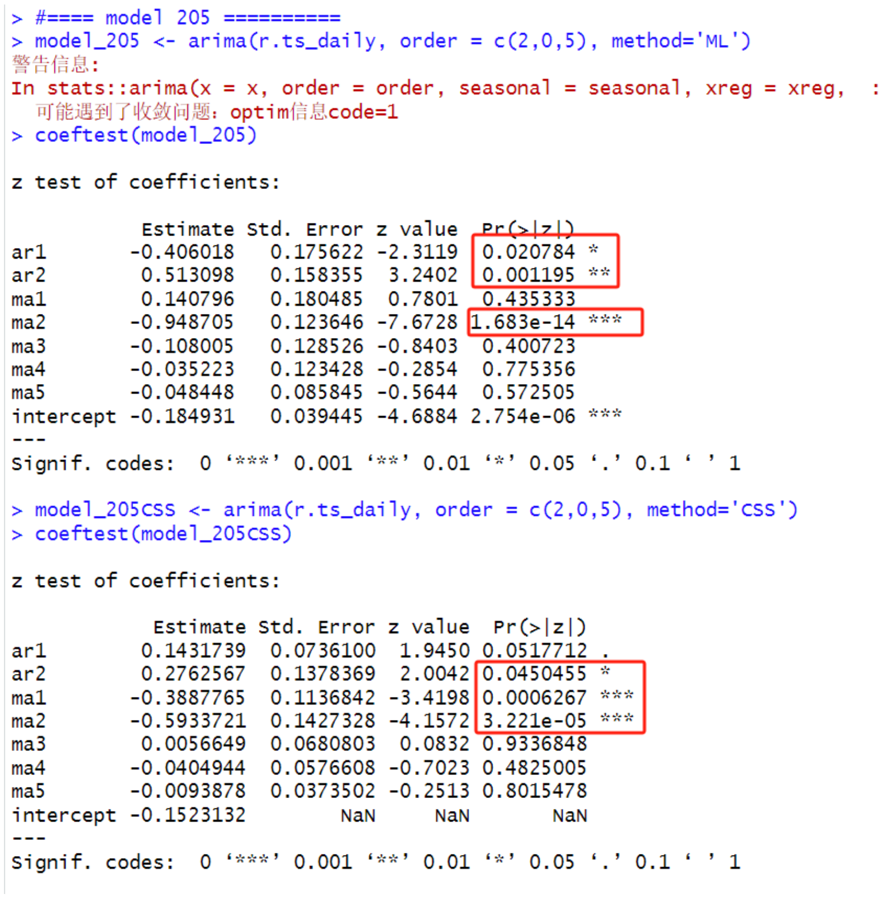


Figure 17. Residual Diagnostics of ARMA(2, 2)

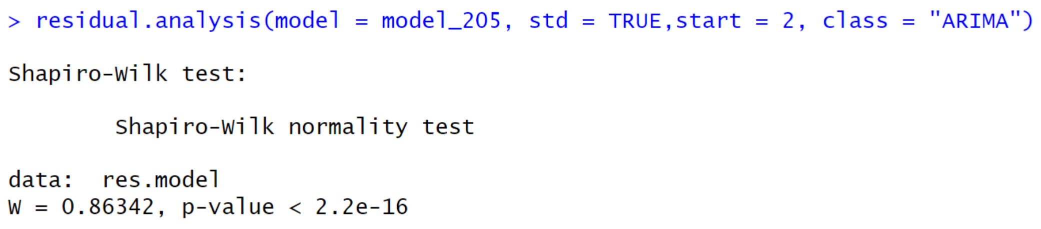
- **ARMA(2, 5)**

The ARMA(2,5) model includes multiple parameters, but only a subset (e.g., AR(2), MA(2)) are statistically significant under both ML and CSS methods. Most higher-order MA terms (MA(3) to MA(5)) are insignificant, suggesting potential overfitting.

Although residual diagnostics (**Figure 18**) show white noise behavior and the Ljung-Box test indicates no residual autocorrelation, the inclusion of many insignificant terms reduces model parsimony. A simpler model like ARMA(1,1) or ARMA(2,2) may offer a more interpretable and efficient alternative.



```
> #==== model 205 ==========
> model_205 <- arima(r.ts_daily, order = c(2,0,5), method='ML')
警告信息:
In stats::arima(x = x, order = order, seasonal = seasonal, xreg = xreg,  :
  可能遇到了收敛问题: optim信息code=1
> coeftest(model_205)

z test of coefficients:

           Estimate Std. Error z value  Pr(>|z|)    
ar1       -0.406018   0.175622 -2.3119  0.020784 *  
ar2        0.513098   0.158355  3.2402  0.001195 ** 
ma1        0.140796   0.180485  0.7801  0.435333    
ma2       -0.948705   0.123646 -7.6728 1.683e-14 ***
ma3       -0.108005   0.128526 -0.8403  0.400723    
ma4       -0.035223   0.123428 -0.2854  0.775356    
ma5       -0.048448   0.085845 -0.5644  0.572505    
intercept -0.184931   0.039445 -4.6884 2.754e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> model_205CSS <- arima(r.ts_daily, order = c(2,0,5), method='CSS')
> coeftest(model_205CSS)

z test of coefficients:

            Estimate Std. Error z value  Pr(>|z|)    
ar1        0.1431739  0.0736100  1.9450 0.0517712 .  
ar2        0.2762567  0.1378369  2.0042 0.0450455 *  
ma1       -0.3887765  0.1136842 -3.4198 0.0006267 ***
ma2       -0.5933721  0.1427328 -4.1572 3.221e-05 ***
ma3        0.0056649  0.0680803  0.0832 0.9336848    
ma4       -0.0404944  0.0576608 -0.7023 0.4825005    
ma5       -0.0093878  0.0373502 -0.2513 0.8015478    
intercept -0.1523132        NaN     NaN       NaN    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



```
> residual.analysis(model = model_205, std = TRUE,start = 2, class = "ARIMA")

Shapiro-Wilk test:

	Shapiro-Wilk normality test

data:  res.model
W = 0.86342, p-value < 2.2e-16
```

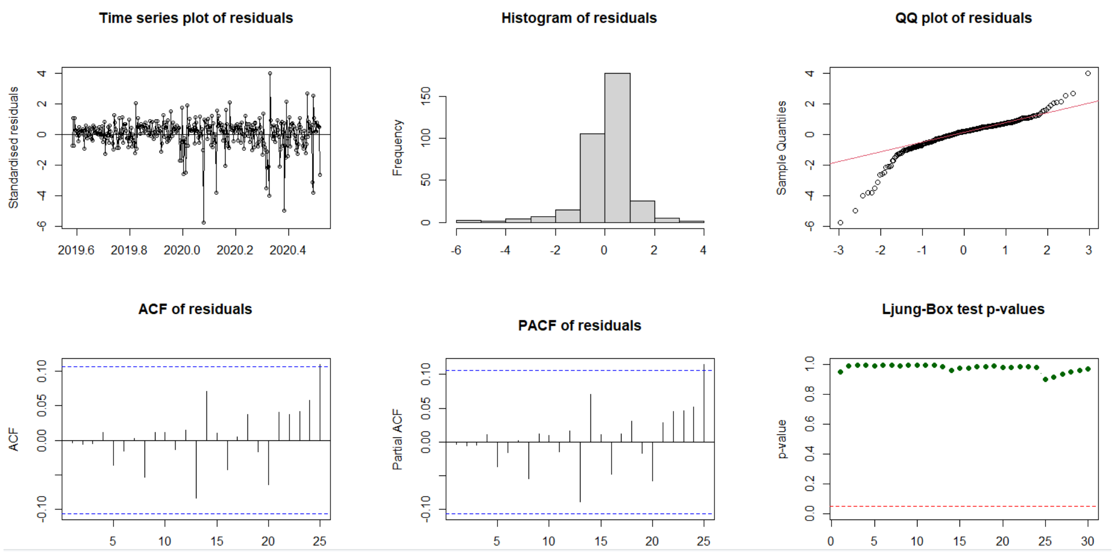
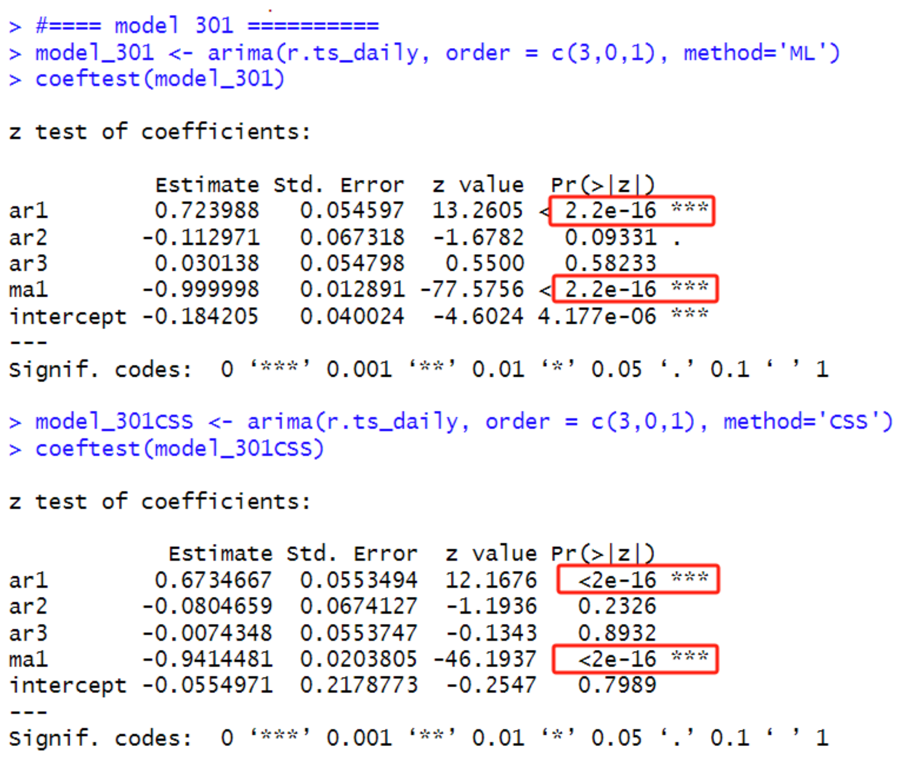


Figure 18. Residual Diagnostics of ARMA(2, 5)

- **ARMA(3, 1)**

The ARMA(3,1) model includes a significant AR(1) and MA(1) term ($p < 0.001$), while the AR(2) and AR(3) terms are not statistically significant under either ML or CSS estimation, suggesting they may be unnecessary. Residual diagnostics (**Figure 19**) show no remaining autocorrelation, and Ljung-Box p-values remain well above 0.05, indicating white-noise residuals.



```
> #==== model 301 ==========
> model_301 <- arima(r.ts_daily, order = c(3,0,1), method='ML')
> coeftest(model_301)

z test of coefficients:

           Estimate Std. Error  z value  Pr(>|z|)    
ar1        0.723988   0.054597  13.2605 < 2.2e-16 ***
ar2       -0.112971   0.067318  -1.6782   0.09331 .  
ar3        0.030138   0.054798   0.5500   0.58233    
ma1       -0.999998   0.012891 -77.5756 < 2.2e-16 ***
intercept -0.184205   0.040024  -4.6024 4.177e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> model_301CSS <- arima(r.ts_daily, order = c(3,0,1), method='CSS')
> coeftest(model_301CSS)

z test of coefficients:

            Estimate Std. Error  z value Pr(>|z|)    
ar1        0.6734667  0.0553494  12.1676   <2e-16 ***
ar2       -0.0804659  0.0674127  -1.1936   0.2326    
ar3       -0.0074348  0.0553747  -0.1343   0.8932    
ma1       -0.9414481  0.0203805 -46.1937   <2e-16 ***
intercept -0.0554971  0.2178773  -0.2547   0.7989    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> residual.analysis(model = model_301, std = TRUE,start = 2, class = "ARIMA")

Shapiro-Wilk test:

        Shapiro-Wilk normality test

data:  res.model
W = 0.85418, p-value < 2.2e-16
```

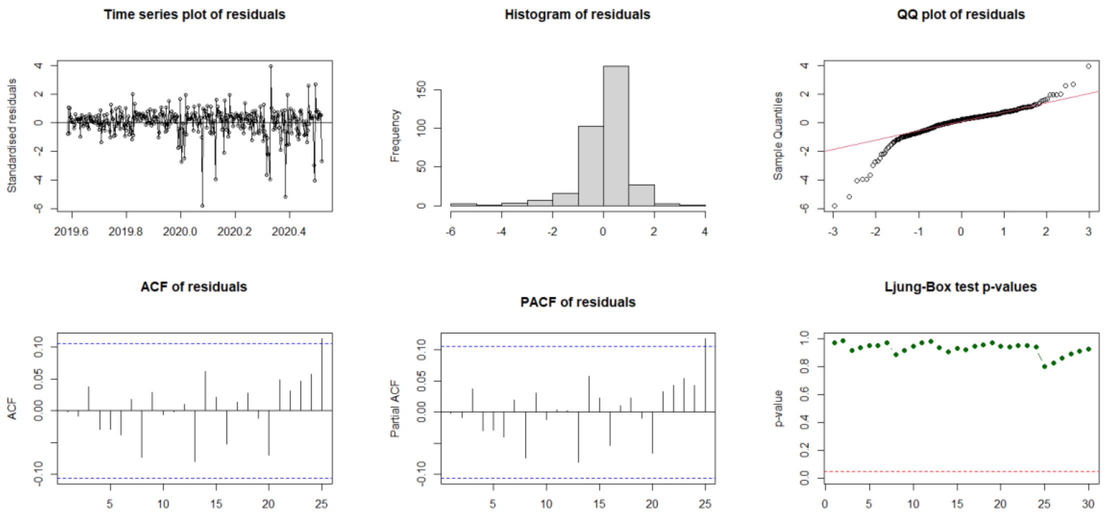


Figure 19. Residual Diagnostics of ARMA(3, 1)

- **ARIMA(4, 1)**

The ARMA(4,1) model yields similar results to ARMA(3,1).

```
> #==== model 401 ==========
> model_401 <- arima(r.ts_daily, order = c(4,0,1), method='ML')
> coeftest(model_401)

z test of coefficients:

           Estimate Std. Error  z value  Pr(>|z|)    
ar1        0.726009   0.054484  13.3251 < 2.2e-16 ***
ar2       -0.120957   0.067503  -1.7919   0.07315 .  
ar3        0.077988   0.067480   1.1557   0.24780    
ar4       -0.066233   0.054755  -1.2096   0.22643    
ma1       -0.999806   0.015229 -65.6497 < 2.2e-16 ***
intercept -0.185130   0.037560  -4.9289 8.268e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> model_401css <- arima(r.ts_daily, order = c(4,0,1), method='CSS')
> coeftest(model_401css)

z test of coefficients:

           Estimate Std. Error  z value Pr(>|z|)    
ar1        0.681513   0.053698  12.6917  < 2e-16 ***
ar2       -0.098652   0.068896  -1.4319  0.15217    
ar3        0.031826   0.066999   0.4750  0.63477    
ar4       -0.076870   0.056262  -1.3663  0.17185    
ma1       -0.963165   0.022659 -42.5068  < 2e-16 ***
intercept -0.231155   0.126138  -1.8326  0.06687 .  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> residual.analysis(model = model_401, std = TRUE,start = 2, class = "ARIMA")

Shapiro-Wilk test:

        Shapiro-Wilk normality test

data:  res.model
W = 0.85721, p-value < 2.2e-16
```

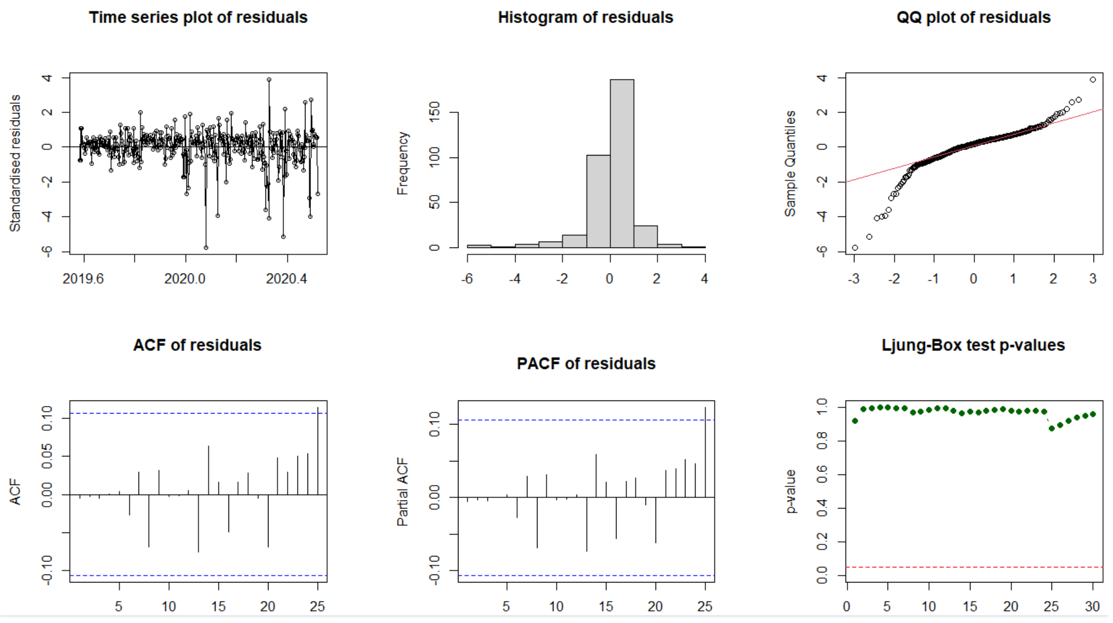


Figure 20. Residual Diagnostics of ARMA(4, 1)

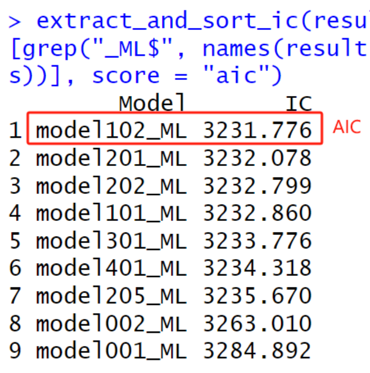
Across all fitted models, simpler structures such as ARMA(1,1), ARMA(1,2), and ARMA(2,1) consistently exhibit stable estimation and significant parameters, while more complex models like ARMA(2,5), ARMA(3,1), and ARMA(4,1) tend to include several insignificant terms, indicating possible overfitting without substantial improvement in residual behavior.

3.5 Model Selection (AIC/BIC) – ARIMA Part

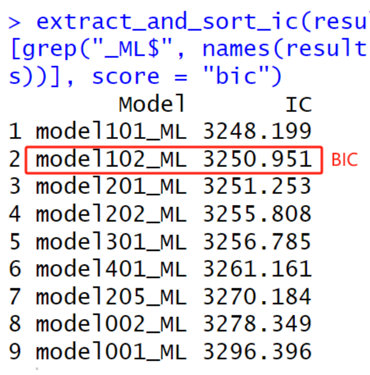
To determine the best-fitting model, I compared the values of the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) across all fitted ARIMA models. As shown in **Table 1**, ARIMA(1, 0, 2) achieved the lowest AIC (3131.776) and second-lowest BIC (3250.951) among all candidate models. Notably, this model also has statistically significant coefficients for both AR and MA terms, further supporting its reliability.

Therefore, based on consistent evidence from AIC, BIC, and parameter significance, **ARIMA(1, 0, 2)** is selected as the best-fitting model for the log-differenced transformed series.

Table 1. AIC and BIC comparision tables for ARIMA models



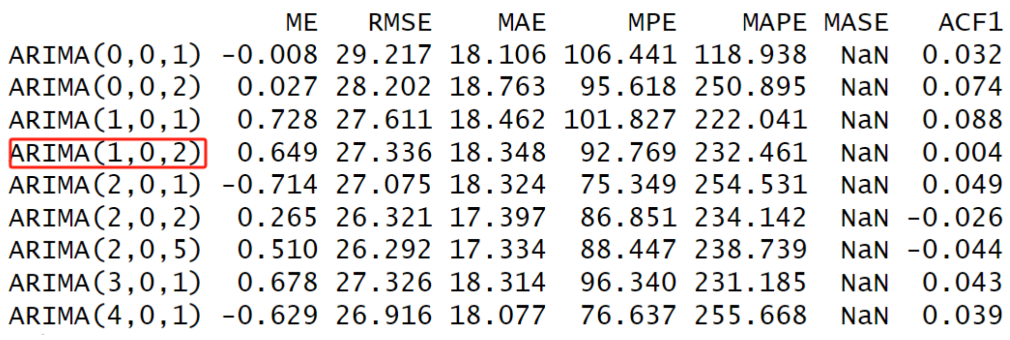
```
> extract_and_sort_ic(results
[grep("_ML$", names(result
s))], score = "aic")
         Model       IC
1 model102_ML 3231.776  AIC
2 model201_ML 3232.078
3 model202_ML 3232.799
4 model101_ML 3232.860
5 model301_ML 3233.776
6 model401_ML 3234.318
7 model205_ML 3235.670
8 model002_ML 3263.010
9 model001_ML 3284.892
```



```
> extract_and_sort_ic(results
[grep("_ML$", names(result
s))], score = "bic")
         Model       IC
1 model101_ML 3248.199
2 model102_ML 3250.951  BIC
3 model201_ML 3251.253
4 model202_ML 3255.808
5 model301_ML 3256.785
6 model401_ML 3261.161
7 model205_ML 3270.184
8 model002_ML 3278.349
9 model001_ML 3296.396
```

3.6 Accuracy Evaluation -ARIMA Part

Table 2. Model accuracy comparison table.



```
                 ME   RMSE    MAE     MPE    MAPE MASE   ACF1
ARIMA(0,0,1) -0.008 29.217 18.106 106.441 118.938  NaN  0.032
ARIMA(0,0,2)  0.027 28.202 18.763  95.618 250.895  NaN  0.074
ARIMA(1,0,1)  0.728 27.611 18.462 101.827 222.041  NaN  0.088
ARIMA(1,0,2)  0.649 27.336 18.348  92.769 232.461  NaN  0.004
ARIMA(2,0,1) -0.714 27.075 18.324  75.349 254.531  NaN  0.049
ARIMA(2,0,2)  0.265 26.321 17.397  86.851 234.142  NaN -0.026
ARIMA(2,0,5)  0.510 26.292 17.334  88.447 238.739  NaN -0.044
ARIMA(3,0,1)  0.678 27.326 18.314  96.340 231.185  NaN  0.043
ARIMA(4,0,1) -0.629 26.916 18.077  76.637 255.668  NaN  0.039
```

To further evaluate model performance, several accuracy metrics were calculated, including ME, RMSE, MAE, MPE, MAPE, and ACF1. As shown in Table 2, the ARIMA(1,0,2) model does not achieve the lowest error in any single metric. However, its performance consistently falls within a favorable range across all indicators—neither the best nor the worst among the candidate models. This balanced and stable performance supports its selection as a robust and reliable model.

3.7 Best Model – ARIMA Part

Among the candidate models, **ARIMA(1,0,2)** stands out as the most suitable choice for the log-differenced transformed series.

3.8 Overparameterized Model: ARIMA(1, 0, 3)

- **ARIMA(1, 0, 3)**

```
> model_103 = arima(r.ts_daily,order=c(1,0,3), method='ML')
> coeftest(model_103)

z test of coefficients:

           Estimate Std. Error z value  Pr(>|z|)    
ar1        0.618293   0.099750  6.1984 5.703e-10 ***
ma1       -0.889758   0.112349 -7.9196 2.383e-15 ***
ma2       -0.157474   0.083036 -1.8965    0.0579 .  
ma3        0.047233   0.086087  0.5487    0.5832    
intercept -0.184278   0.039980 -4.6093 4.041e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> model_103_css = arima(r.ts_daily,order=c(1,0,3), method='CSS')
> coeftest(model_103_css)

z test of coefficients:

           Estimate Std. Error z value  Pr(>|z|)    
ar1        0.431399   0.132686  3.2513  0.001149 ** 
ma1       -0.665459   0.134094 -4.9626 6.955e-07 ***
ma2       -0.213466   0.085358 -2.5008  0.012390 *  
ma3       -0.046445   0.080013 -0.5805  0.561597    
intercept -0.062345   0.202120 -0.3085  0.757735    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> residual.analysis(model = model_103_css)

Shapiro-Wilk test:

        Shapiro-Wilk normality test

data:  res.model
W = 0.8665, p-value < 2.2e-16
```

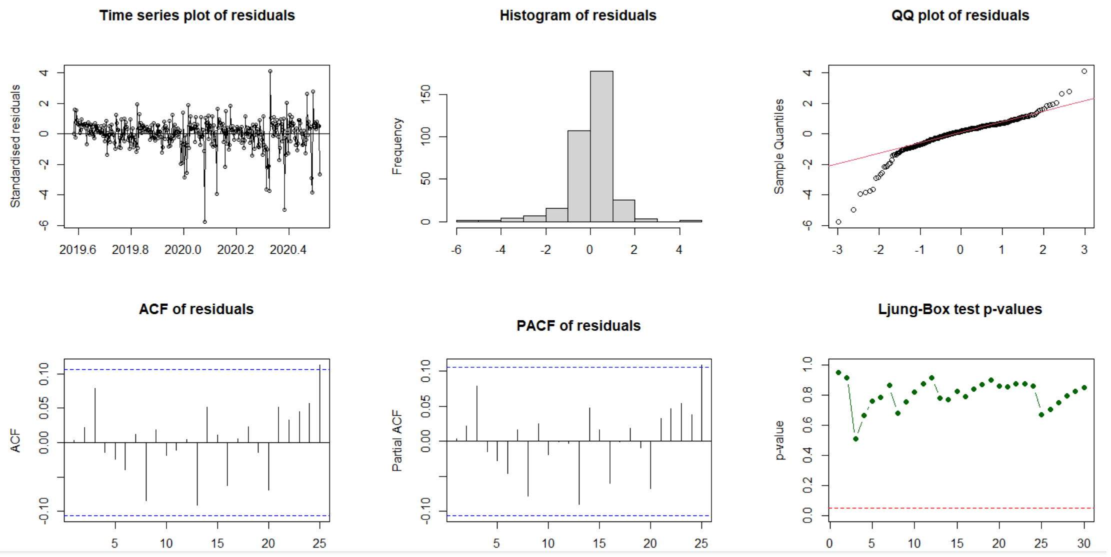
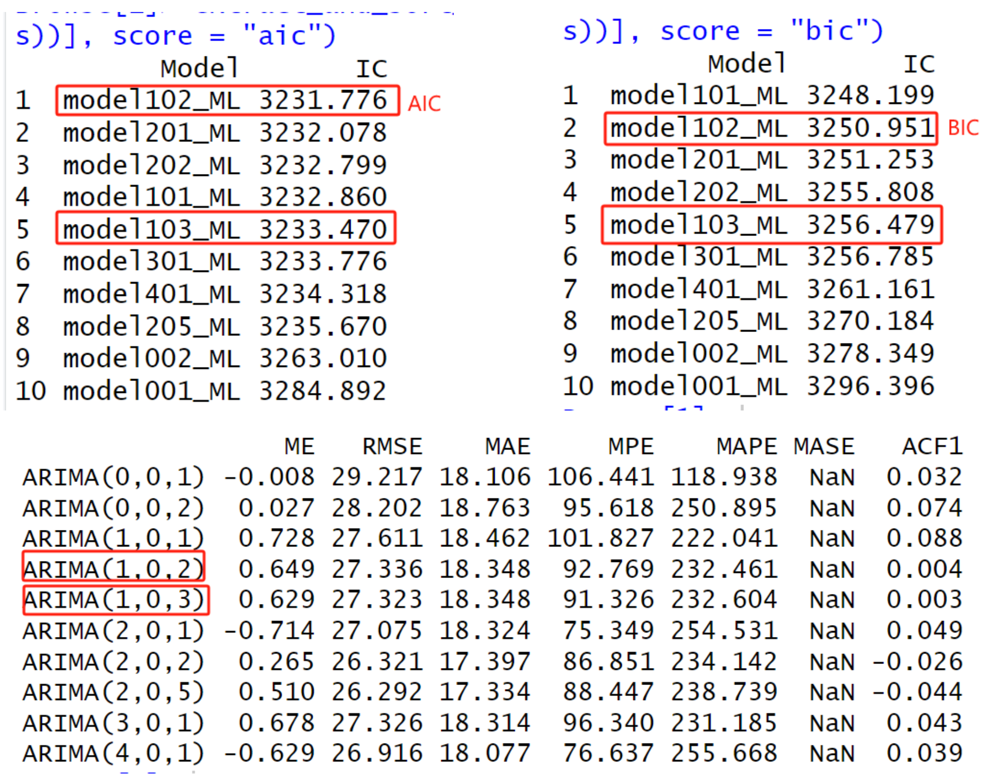


Figure 21. Residual Diagnostics of ARMA(1, 3).

Table 3. AIC/BIC score and model accuracy comparison table including ARMA(1,3).



```
s))], score = "aic")
         Model        IC
1  model102_ML  3231.776
2  model201_ML  3232.078
3  model202_ML  3232.799
4  model101_ML  3232.860
5  model103_ML  3233.470
6  model301_ML  3233.776
7  model401_ML  3234.318
8  model205_ML  3235.670
9  model002_ML  3263.010
10 model001_ML  3284.892
```

```
s))], score = "bic")
         Model        IC
1  model101_ML  3248.199
2  model102_ML  3250.951
3  model201_ML  3251.253
4  model202_ML  3255.808
5  model103_ML  3256.479
6  model301_ML  3256.785
7  model401_ML  3261.161
8  model205_ML  3270.184
9  model002_ML  3278.349
10 model001_ML  3296.396
```

```
                  ME   RMSE    MAE     MPE    MAPE MASE   ACF1
ARIMA(0,0,1) -0.008 29.217 18.106 106.441 118.938  NaN  0.032
ARIMA(0,0,2)  0.027 28.202 18.763  95.618 250.895  NaN  0.074
ARIMA(1,0,1)  0.728 27.611 18.462 101.827 222.041  NaN  0.088
ARIMA(1,0,2)  0.649 27.336 18.348  92.769 232.461  NaN  0.004
ARIMA(1,0,3)  0.629 27.323 18.348  91.326 232.604  NaN  0.003
ARIMA(2,0,1) -0.714 27.075 18.324  75.349 254.531  NaN  0.049
ARIMA(2,0,2)  0.265 26.321 17.397  86.851 234.142  NaN -0.026
ARIMA(2,0,5)  0.510 26.292 17.334  88.447 238.739  NaN -0.044
ARIMA(3,0,1)  0.678 27.326 18.314  96.340 231.185  NaN  0.043
ARIMA(4,0,1) -0.629 26.916 18.077  76.637 255.668  NaN  0.039
```

ARIMA(1,0,3) is an overparameterized extension of the more parsimonious ARIMA(1,0,2) model. While both models show comparable performance in terms of RMSE, MAE, and ACF1, ARIMA(1,0,3) does not offer substantial improvement in predictive accuracy. Moreover, although it ranks close in AIC and BIC (e.g., 5th vs. 1st in AIC), it introduces additional complexity by including an extra MA term. From a model selection perspective, this added complexity is not justified by a meaningful gain in fit. Residual diagnostics (**Figure 21**) also indicate well-behaved residuals for both models, with no major improvement in ARIMA(1,0,3). Therefore, ARIMA(1,0,2) remains the preferred model due to its better balance of simplicity and performance.

3.9Model Identification – GARCH Part

To capture volatility clustering in the residuals of the precipitation series, I employed a Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model. I first fitted an ARMA(1,2) model to account for autocorrelation in the mean structure of the series. Since the GARCH(p,q) model is mathematically analogous to an ARMA(max(p,q), q) model applied to the squared or absolute residuals of the original return series, I first examined the ACF, PACF, and EACF plots of both the squared and absolute log-differenced returns.

(1) ACF-PACF for absolute return series

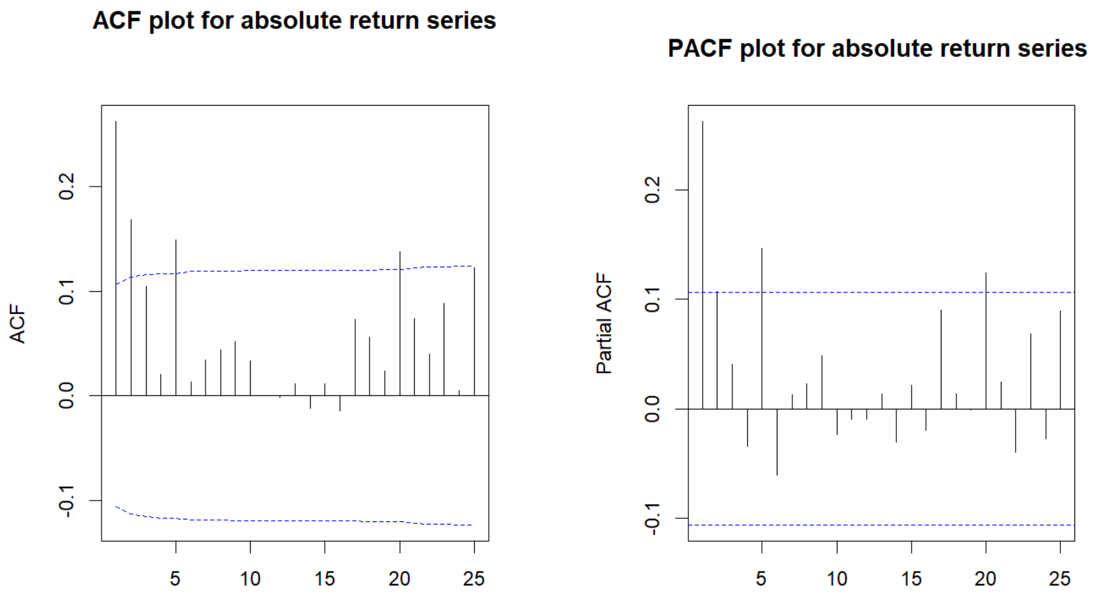


Figure 22. ACF and PACF plot for absolute return series.

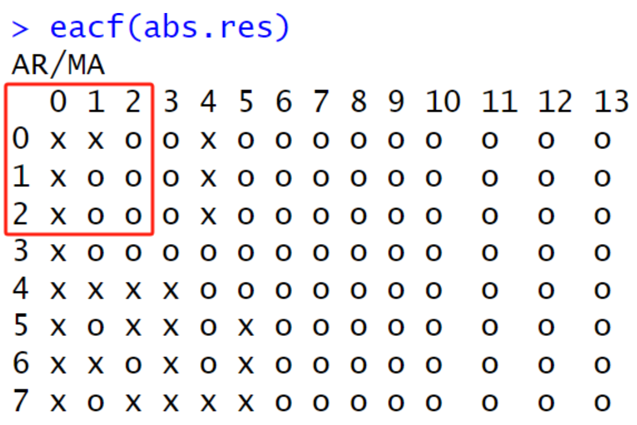
Derivation Process:

- *From PACF ==> max(p,q) = 2*
- *From ACF ==> q=0, 1, 2, 3*
- *max(p,q) = 2 and q=0 ==> max(p, 0) = 2 ==> p = 2*
- *max(p,q) = 2 and q=1 ==> max(p, 1) = 2 ==> p = 2*
- *max(p,q) = 2 and q=2 ==> max(p, 2) = 2 ==> p = 0, 1, 2*
- *max(p,q) = 2 and q=3 ==> max(p, 3) = 2 ==> Doesn't make any sense.*

GARCH(2,0), GARCH(2,1), GARCH(0,2), GARCH(1,2), GARCH(2,2)

From the ACF and PACF plots of the absolute return series (**Figure 22**), I observed five candidate models: GARCH(2,0), GARCH(2,1), GARCH(0,2), GARCH(1,2), GARCH(2,2).

(2) EACF of absolute return series



```
> eacf(abs.res)
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x o o x o o o o o o  o  o  o
1 x o o o x o o o o o o  o  o  o
2 x o o o x o o o o o o  o  o  o
3 x o o o o o o o o o o  o  o  o
4 x x x x o o o o o o o  o  o  o
5 x o x x o x o o o o o  o  o  o
6 x x o x o x o o o o o  o  o  o
7 x o x x x x o o o o o  o  o  o
```

Figure 23. EACF table for absolute return series.

Derivation Process:

Top-left-o is at (0,2), (1,1), (1,2), (2,1), (2,2)

- *max(p,q) = 0 and q=2 ==> max(p,2)=0 ==> Doesn't make any sense.*
- *max(p,q) = 1 and q=1 ==> max(p,1)=1 ==> p=0 or p=1*
- *max(p,q) = 1 and q=2 ==> max(p,2)=1 ==> Doesn't make any sense.*
- *max(p,q) = 2 and q=1 ==> max(p,1)=2 ==> p=2*
- *max(p,q) =2 and q=2 ==> max(p,2)=2 ==> p=0, p=1, p=2*

GARCH(0,1), GARCH(1,1), GARCH(2,1), GARCH(0,2), GARCH(1,2), GARCH(2,2)

Based on the EACF table (**Figure 23**) of the absolute residuals, I observed six potential models include: GARCH(0,1), GARCH(1,1), GARCH(2,1), GARCH(0,2), GARCH(1,2), GARCH(2,2).

(3) ACF & PACF of square return series

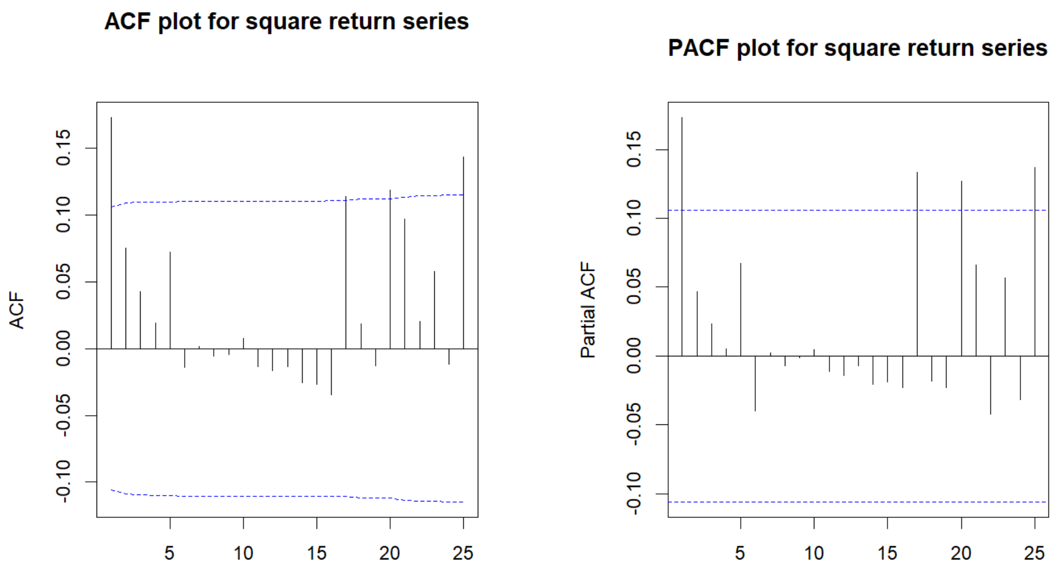


Figure 24. ACF-PACF plot for square return series.

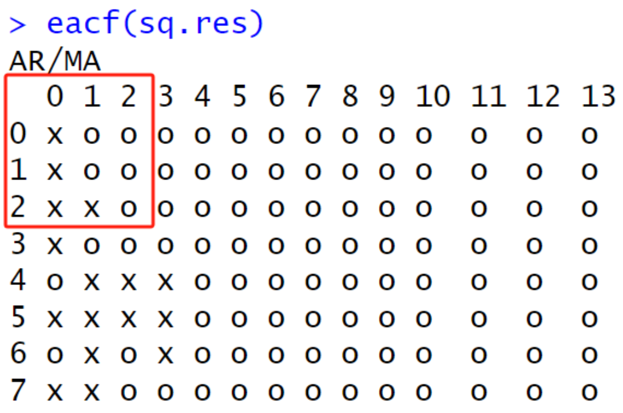
Derivation Process:

- *From PACF, max(p,q) = 1*
- *From ACF, q = 1*
- *max(p,q) = max(p,1) = 1 ==> p=0, 1*

GARCH(0,1), GARCH(1,1)

From the ACF and PACF plots of the square return series (**Figure 24**), I observed two candidate models: GARCH(0,1), GARCH(1,1).

(4) EACF of square return series



```
> eacf(sq.res)
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x o o o o o o o o o o  o  o  o
1 x o o o o o o o o o o  o  o  o
2 x x o o o o o o o o o  o  o  o
3 x o o o o o o o o o o  o  o  o
4 o x x x o o o o o o o  o  o  o
5 x x x x o o o o o o o  o  o  o
6 o x o x o o o o o o o  o  o  o
7 x x o o o o o o o o o  o  o  o
```

Figure 25. EACF table for square return series.

Derivation Process:

Top-left-o is at (0,1), (0,2), (1,1), (1,2), (2,1), (2,2)

- *max(p,q) = 0 and q=1 ==> max(p,1)=0 ==> Doesn't make any sense.*
- *max(p,q) = 0 and q=2 ==> max(p,2)=0 ==> Doesn't make any sense.*
- *max(p,q) = 1 and q=1 ==> max(p,1)=1 ==> p=0 or p=1*
- *max(p,q) = 1 and q=2 ==> max(p,2)=1 ==> Doesn't make any sense.*
- *max(p,q) = 2 and q=1 ==> max(p,1)=2 ==> p=2*
- *max(p,q) =2 and q=2 ==> max(p,2)=2 ==> p=0, p=1, p=2*

GARCH(0,1), GARCH(1,1), GARCH(2,1), GARCH(0,2), GARCH(1,2), GARCH(2,2)

Based on the EACF table (**Figure 25**) of the absolute residuals, I observed six potential models include: GARCH(0,1), GARCH(1,1), GARCH(2,1), GARCH(0,2), GARCH(1,2), GARCH(2,2).

All in all, seven candidate GARCH models were identified:

{GARCH(0,1), GARCH(0,2), GARCH(1,1), GARCH(1,2), GARCH(2,0), GARCH(2,1), GARCH(2,2)}.

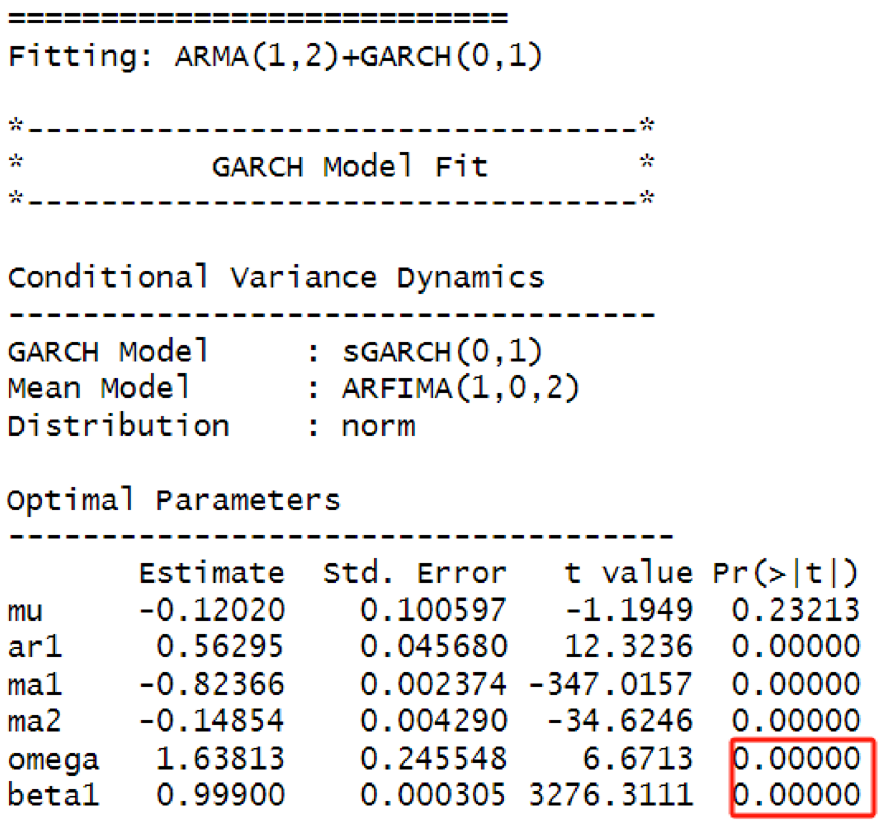
3.10 Model Estimation and Residual Diagnostics – ARMA(1,2)+GARCH(p,q)

All seven ARMA(1,2)+GARCH(p,q) models were successfully estimated using the ugarchfit() function from the **rugarch** package in R.

Across most models, at least one alpha and one beta term are statistically significant, suggesting the presence of both immediate volatility response and long-term volatility persistence. However, in higher-order models, such as GARCH(2,2), some coefficients are not significant, indicating potential overfitting and suggesting that simpler models may offer comparable performance with greater parsimony.

All fitted models exhibit similar residual characteristics. The Ljung-Box test results consistently show that autocorrelation in the residuals is not statistically significant across different lags. Additionally, the time series plots of the standardized residuals reveal no visible patterns or structure, indicating that the models have effectively captured the serial dependence in the data. These findings suggest that the residuals behave like white noise, and no substantial trend remains unexplained.

- **ARMA(1, 2) + GARCH (0, 1)**



```
==============================
Fitting: ARMA(1,2)+GARCH(0,1)

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model      : sGARCH(0,1)
Mean Model       : ARFIMA(1,0,2)
Distribution     : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value Pr(>|t|)
mu      -0.12020    0.100597   -1.1949  0.23213
ar1      0.56295    0.045680   12.3236  0.00000
ma1     -0.82366    0.002374 -347.0157  0.00000
ma2     -0.14854    0.004290  -34.6246  0.00000
omega    1.63813    0.245548    6.6713  0.00000
beta1    0.99900    0.000305 3276.3111  0.00000
```

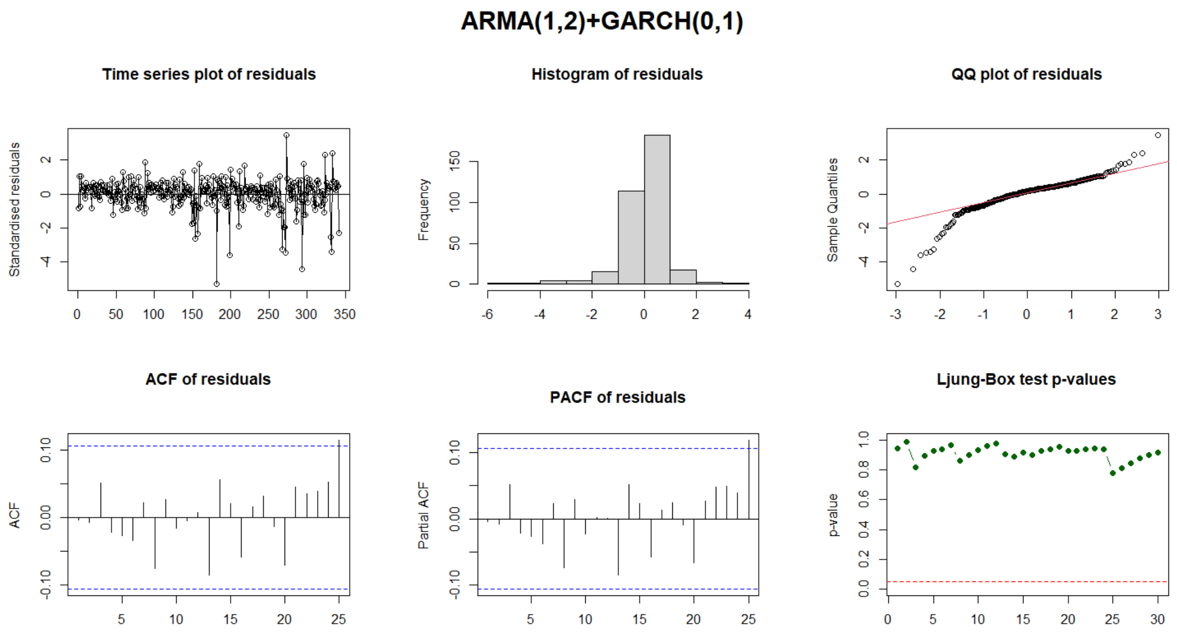
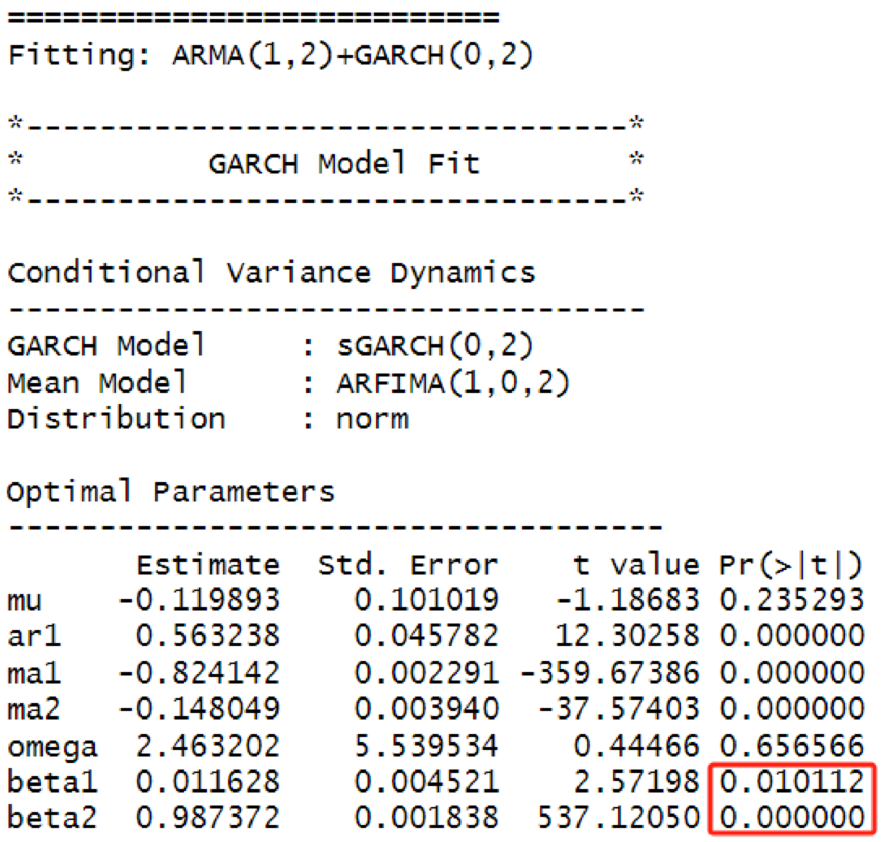


Figure 26. Residual Diagnostics of GARCH(0, 1)

- **ARMA(1, 2) + GARCH (0, 2)**



```
============================
Fitting: ARMA(1,2)+GARCH(0,2)

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(0,2)
Mean Model      : ARFIMA(1,0,2)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error    t value Pr(>|t|)
mu     -0.119893    0.101019   -1.18683 0.235293
ar1     0.563238    0.045782   12.30258 0.000000
ma1    -0.824142    0.002291 -359.67386 0.000000
ma2    -0.148049    0.003940  -37.57403 0.000000
omega   2.463202    5.539534    0.44466 0.656566
beta1   0.011628    0.004521    2.57198 0.010112
beta2   0.987372    0.001838  537.12050 0.000000
```

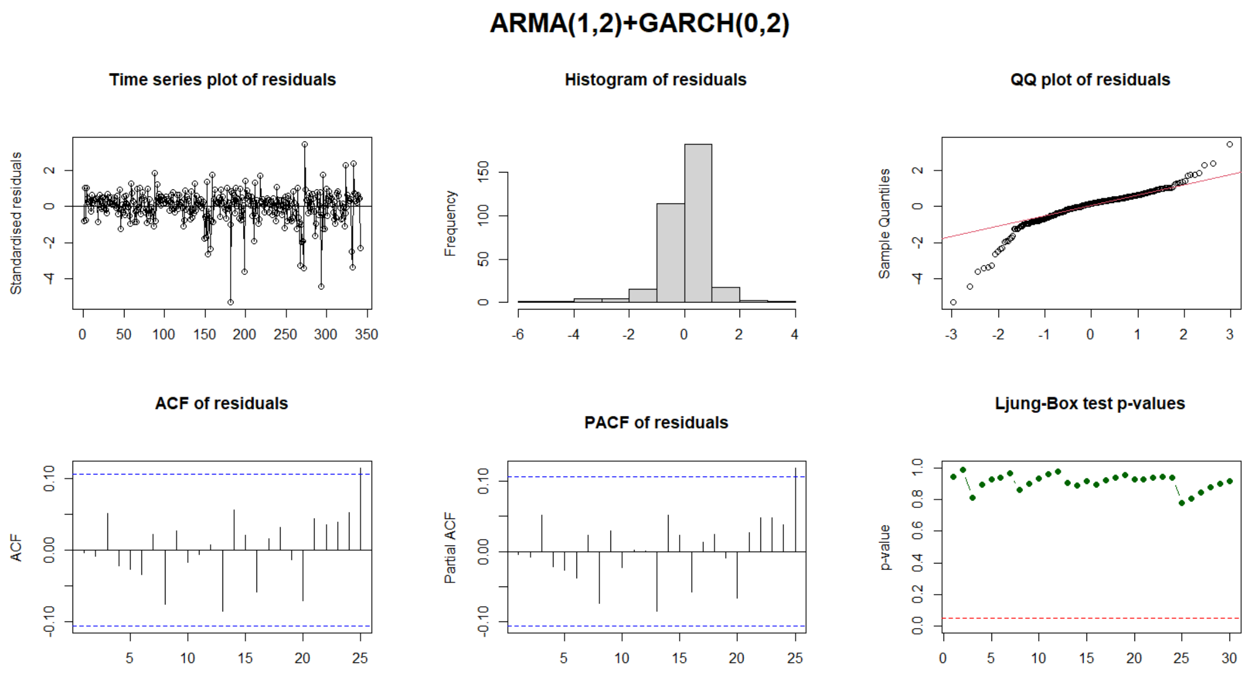
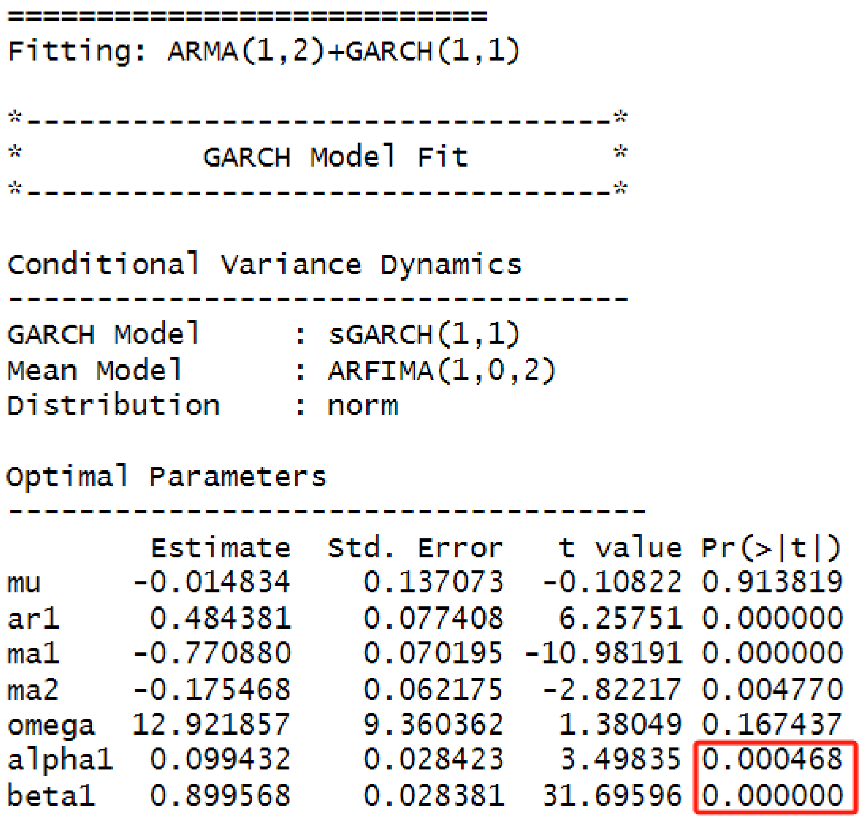


Figure 27. Residual Diagnostics of GARCH(0, 2)

- **ARMA(1, 2) + GARCH (1, 1)**



```
============================
Fitting: ARMA(1,2)+GARCH(1,1)

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(1,1)
Mean Model      : ARFIMA(1,0,2)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value Pr(>|t|)
mu     -0.014834    0.137073  -0.10822 0.913819
ar1     0.484381    0.077408   6.25751 0.000000
ma1    -0.770880    0.070195 -10.98191 0.000000
ma2    -0.175468    0.062175  -2.82217 0.004770
omega  12.921857    9.360362   1.38049 0.167437
alpha1  0.099432    0.028423   3.49835 0.000468
beta1   0.899568    0.028381  31.69596 0.000000
```

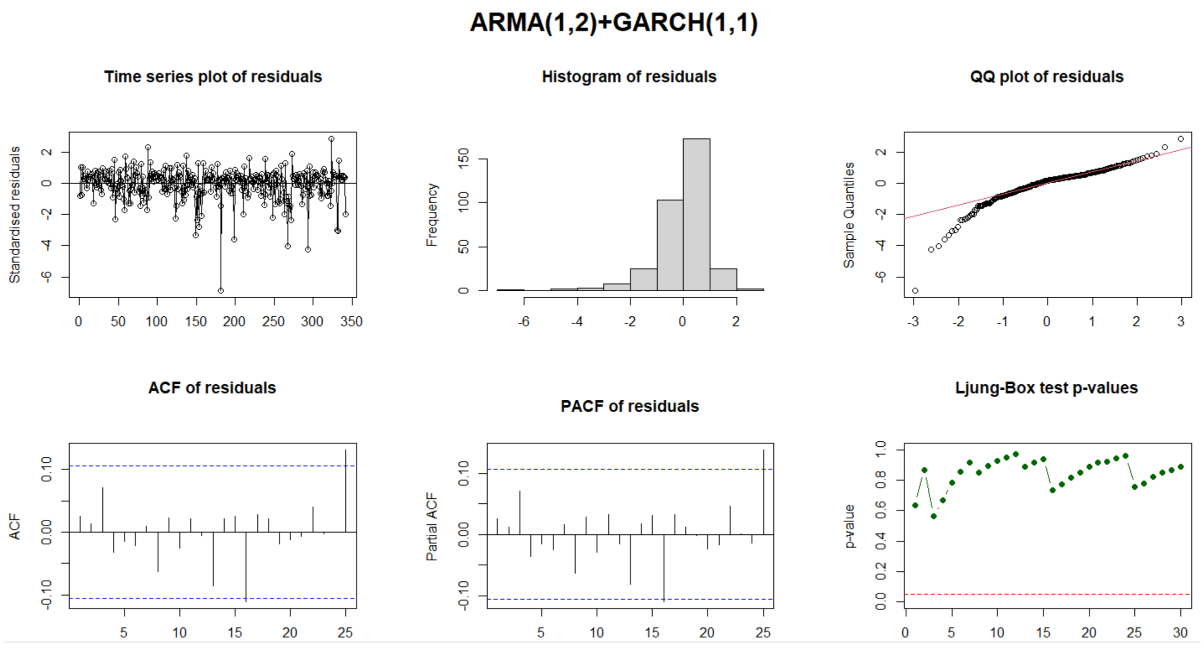


Figure 28. Residual Diagnostics of GARCH(1, 1)

- **ARMA(1, 2) + GARCH (1, 2)**

```
============================
Fitting: ARMA(1,2)+GARCH(1,2)

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model      : sGARCH(1,2)
Mean Model       : ARFIMA(1,0,2)
Distribution     : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error    t value Pr(>|t|)
mu      0.120012    0.106150   1.130587 0.258229
ar1     0.559747    0.052463  10.669441 0.000000
ma1    -0.885678    0.009999 -88.572396 0.000000
ma2    -0.079584    0.008743  -9.102908 0.000000
omega  30.861917   23.749913   1.299454 0.193788
alpha1  0.255254    0.087475   2.918012 0.003523
beta1   0.000000    0.026861   0.000001 0.999999
beta2   0.743746    0.108731   6.840236 0.000000
```

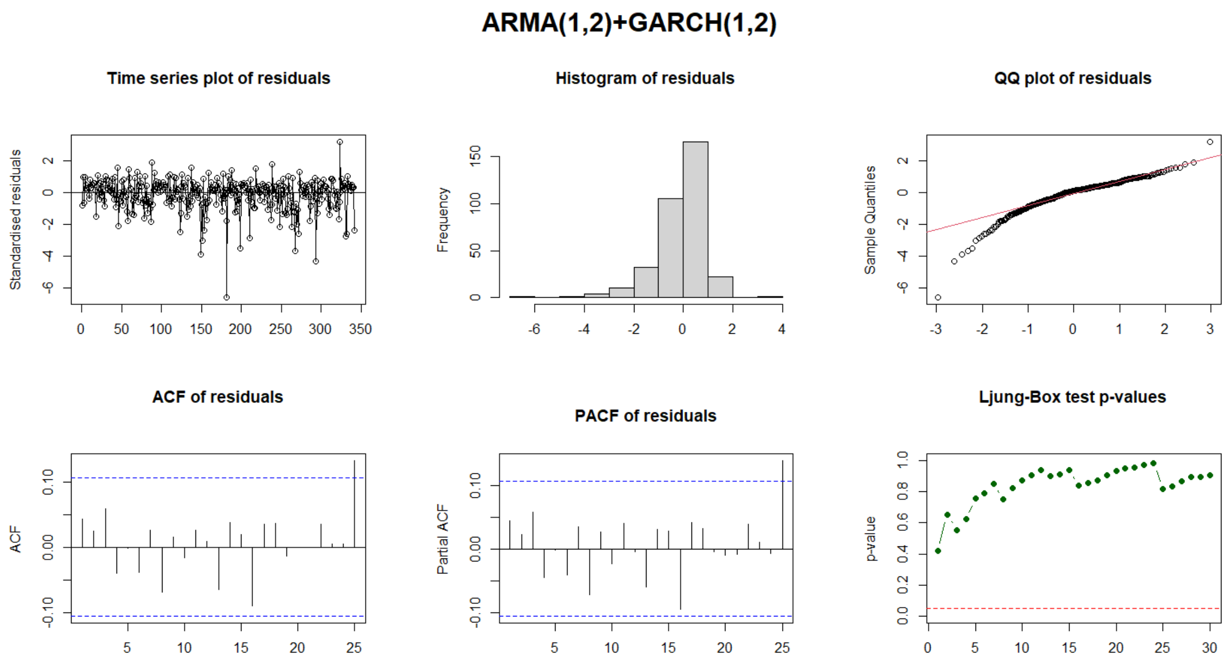


Figure 29. Residual Diagnostics of GARCH(1, 2)

- **ARMA(1, 2) + GARCH (2, 0)**

```
===========================
Fitting: ARMA(1,2)+GARCH(2,0)

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(2,0)
Mean Model      : ARFIMA(1,0,2)
Distribution    : norm

Optimal Parameters
------------------------------------
         Estimate  Std. Error   t value Pr(>|t|)
mu       0.134827    0.086155   1.56493 0.117599
ar1      0.363037    0.069780   5.20257 0.000000
ma1     -0.726430    0.009901 -73.37103 0.000000
ma2     -0.235239    0.009593 -24.52200 0.000000
omega  307.754091   46.554639   6.61060 0.000000
alpha1   0.791113    0.174113   4.54367 0.000006
alpha2   0.086233    0.101474   0.84981 0.395429
```

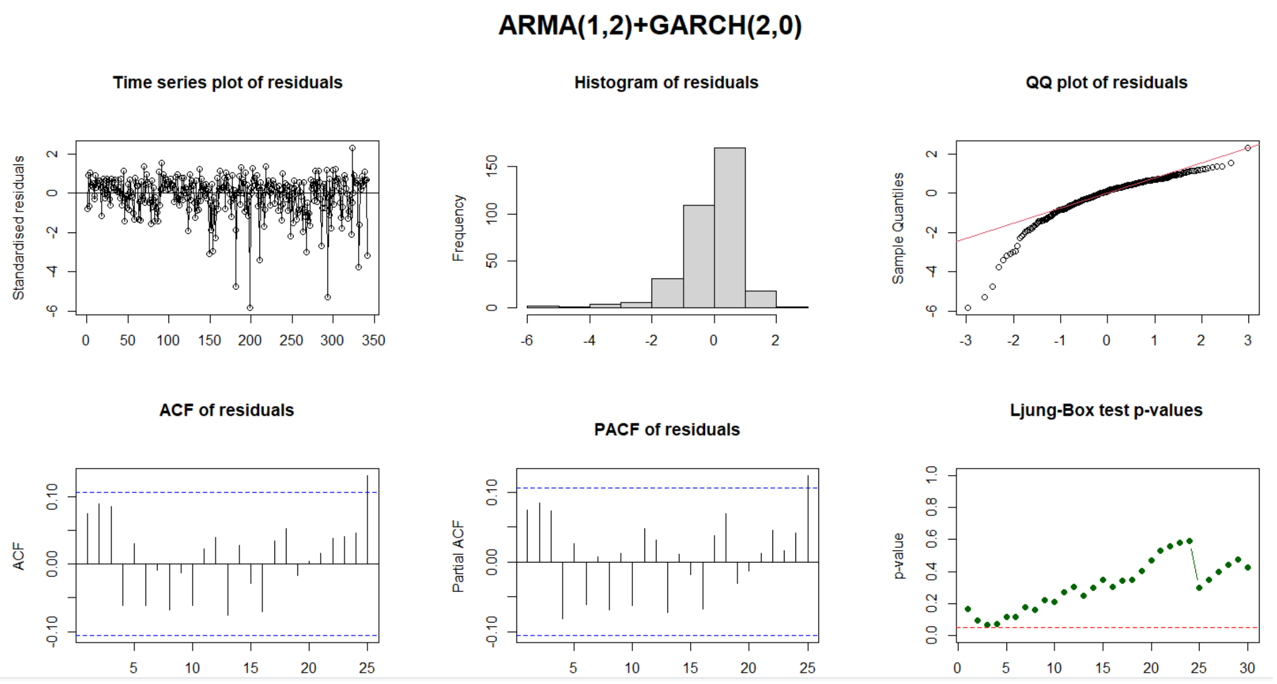
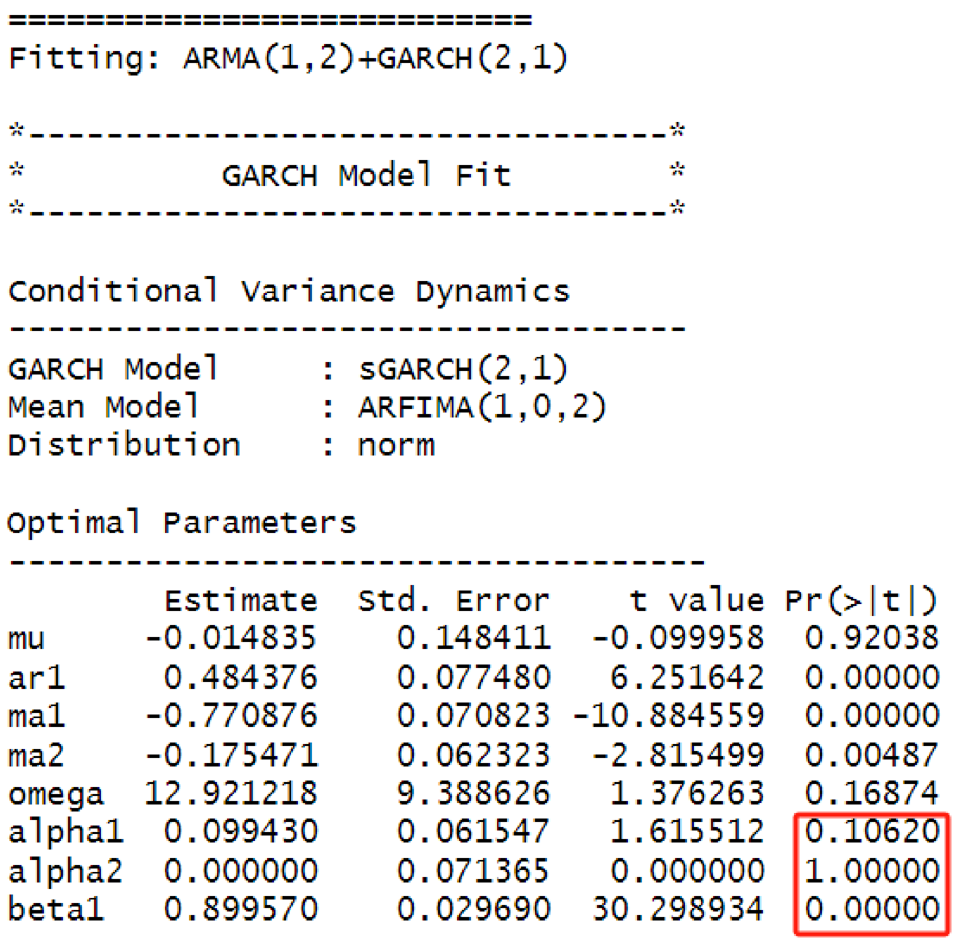


Figure 30. Residual Diagnostics of GARCH(2, 0)

- **ARMA(1, 2) + GARCH (2, 1)**



```
==============================
Fitting: ARMA(1,2)+GARCH(2,1)

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(2,1)
Mean Model      : ARFIMA(1,0,2)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value Pr(>|t|)
mu     -0.014835    0.148411 -0.099958  0.92038
ar1     0.484376    0.077480  6.251642  0.00000
ma1    -0.770876    0.070823 -10.884559 0.00000
ma2    -0.175471    0.062323 -2.815499  0.00487
omega  12.921218    9.388626  1.376263  0.16874
alpha1  0.099430    0.061547  1.615512  0.10620
alpha2  0.000000    0.071365  0.000000  1.00000
beta1   0.899570    0.029690 30.298934  0.00000
```

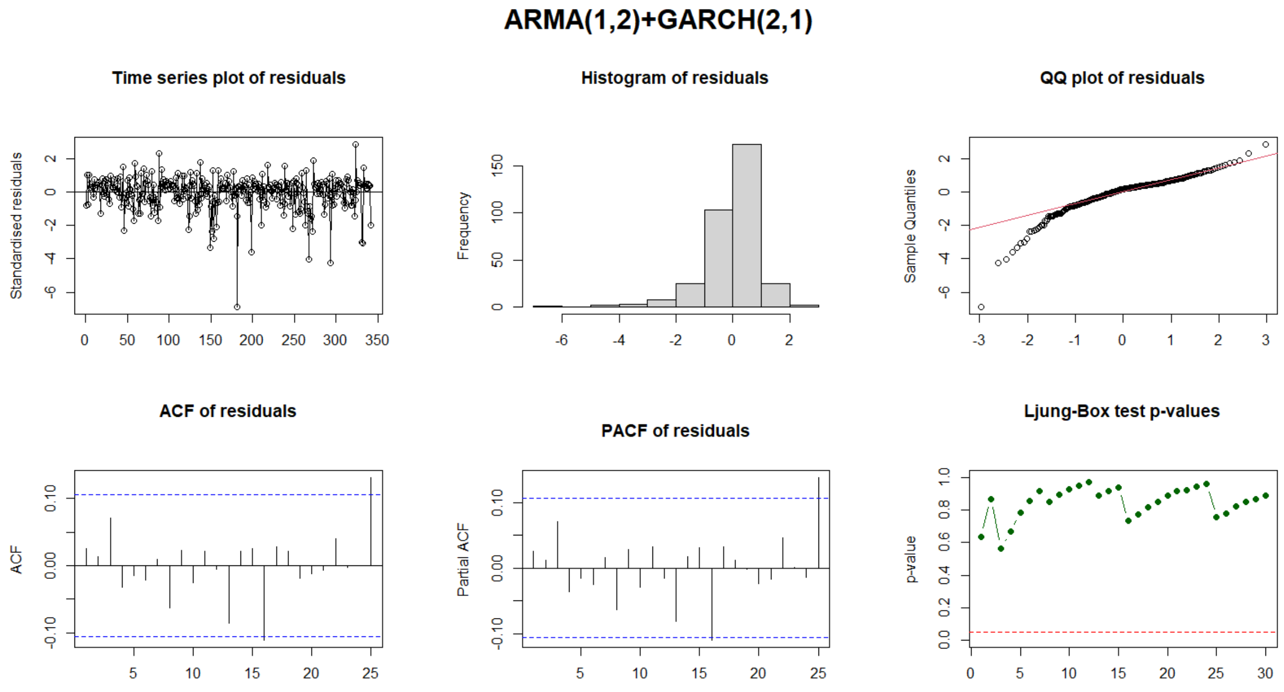


Figure 31. Residual Diagnostics of GARCH(2, 1)

- **ARMA(1, 2) + GARCH (2, 2)**

```
============================
Fitting: ARMA(1,2)+GARCH(2,2)

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model      : sGARCH(2,2)
Mean Model       : ARFIMA(1,0,2)
Distribution     : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error    t value Pr(>|t|)
mu       0.12002    0.071765   1.672353 0.094455
ar1      0.55975    0.045442  12.318052 0.000000
ma1     -0.88568    0.010035 -88.263124 0.000000
ma2     -0.07958    0.009296  -8.560368 0.000000
omega   30.86715    6.942655   4.446014 0.000009
alpha1   0.25528    0.118122   2.161135 0.030685
alpha2   0.00000    0.047031   0.000001 0.999999
beta1    0.00000    0.092946   0.000000 1.000000
beta2    0.74372    0.179597   4.141073 0.000035
```

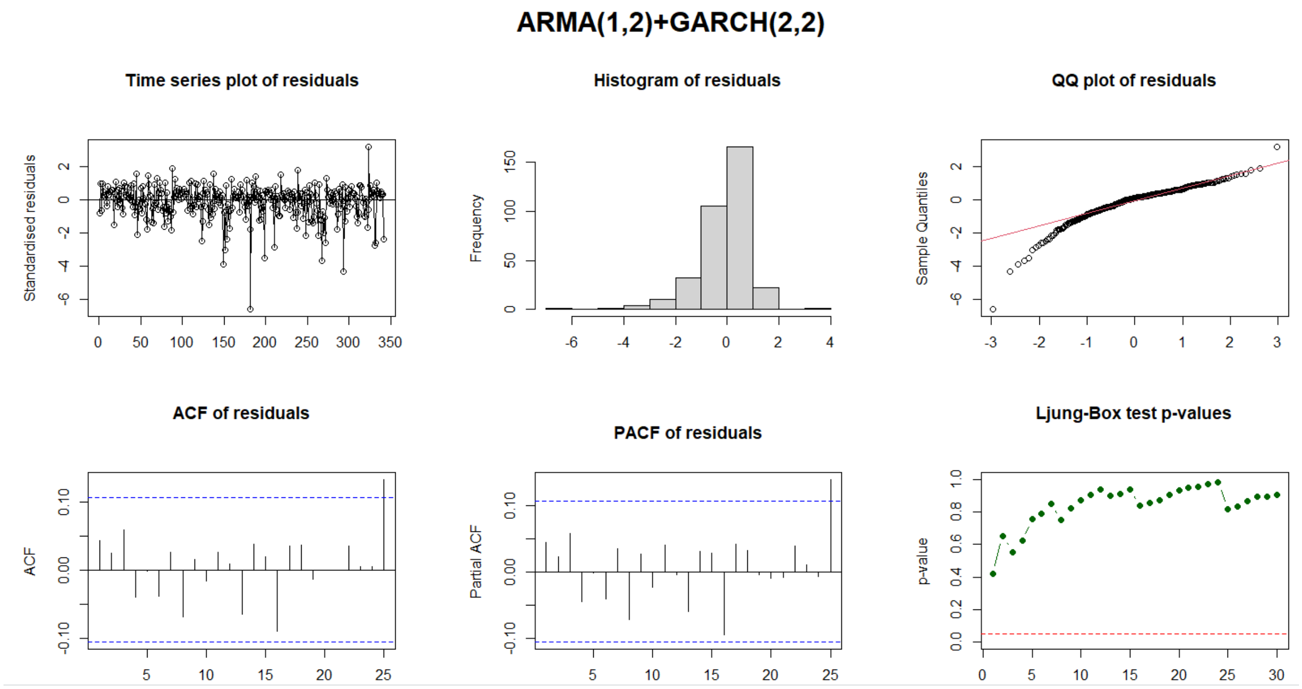
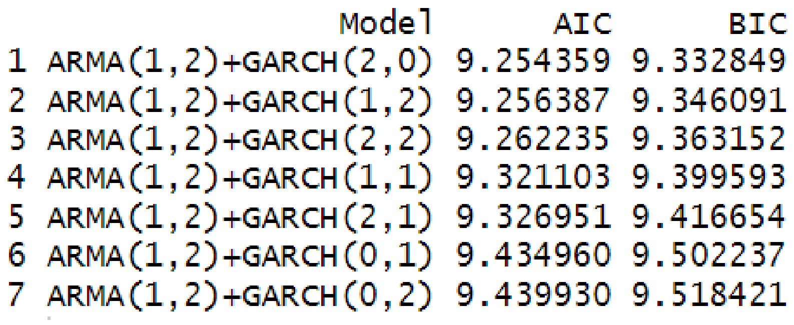


Figure 32. Residual Diagnostics of GARCH(2, 2)

3.10 Model Selection (AIC/BIC) – ARIMA+GARCH

Table 4. AIC and BIC comparision tables for ARIMA(1, 2)+GARCH(p,q) models



```
                      Model      AIC      BIC
1 ARMA(1,2)+GARCH(2,0) 9.254359 9.332849
2 ARMA(1,2)+GARCH(1,2) 9.256387 9.346091
3 ARMA(1,2)+GARCH(2,2) 9.262235 9.363152
4 ARMA(1,2)+GARCH(1,1) 9.321103 9.399593
5 ARMA(1,2)+GARCH(2,1) 9.326951 9.416654
6 ARMA(1,2)+GARCH(0,1) 9.434960 9.502237
7 ARMA(1,2)+GARCH(0,2) 9.439930 9.518421
```

Model selection was further guided by comparing AIC and BIC values across all candidate ARMA(1,2)+GARCH(p,q) models. As shown in **Table 4**, the model **ARMA(1,2)+GARCH(2,0)** achieved the lowest AIC (9.254) and BIC (9.333), indicating the best overall fit among the seven candidates.

To further investigate the dynamic volatility structure captured by the GARCH model, I plotted the return series along with two conditional standard deviation bands (±2 SD) (**Figure 33**). This plot shows how volatility changes over time. The bands get wider during early 2020, meaning the volatility increased, and then they narrow again, showing calmer periods.

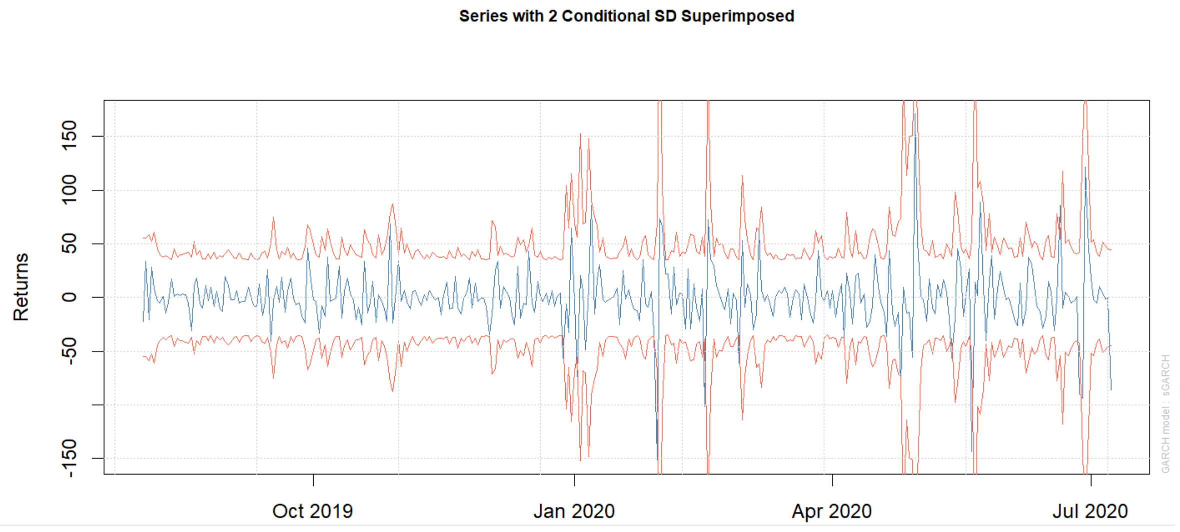


Figure 33. Series with ±2 standard deviations superimposed.

4. FORECAST

4.1 A 10-day ahead forecast

From the **Figure 34**, the forecasted return values are shown in red, while the shaded pink area represents the 95% confidence band derived from the conditional volatility estimates. This visualization offers insight into the short-term predictive performance of the ARMA(1,2)+GARCH(2,0) model, as well as the uncertainty associated with future return behavior.

```
*------------------------------------*
*       GARCH Model Forecast         *
*------------------------------------*
Model: sGARCH
Horizon: 10
Roll Steps: 0
Out of Sample: 0

0-roll forecast [T0=2020-07-08]:
      Series Sigma
T+1  17.3455 66.19
T+2  23.2231 64.93
T+3   8.5167 63.41
T+4   3.1778 62.07
T+5   1.2395 60.84
T+6   0.5359 59.74
T+7   0.2804 58.74
T+8   0.1877 57.84
T+9   0.1540 57.02
T+10  0.1418 56.29
```

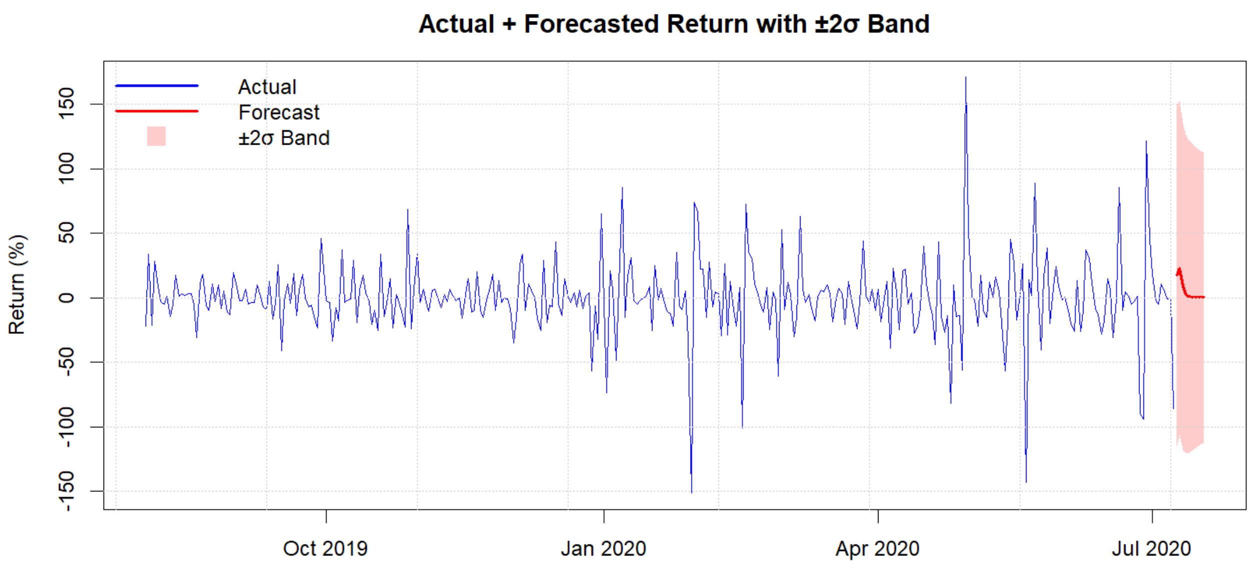


Figure 34. A 10-day ahead return forecast using the ARMA(1,2) +GARCH(2,0) model.

I observe that the forecasted returns remain relatively stable and close to zero, suggesting no major directional shifts are expected in the immediate future. However, the widening of the ±2σ band over time reflects increasing uncertainty in the forecasts, which is typical in GARCH-based models due to the compounding effect of forecasted variance.

4.2 Conversion of forecasted log returen to actual values

Since the GARCH model produces forecasts on the log return scale, it is necessary to transform these predictions back to the original scale of electricity generation to make them interpretable. To achieve this, I used an exponential transformation. Starting from the last observed actual value in the original time series (denoted as last_value), I cumulatively applied the forecasted log returns using the following formula:

$$\begin{tikzcd} \boxed{\text{last\_value}} \arrow[r] & \boldsymbol{Forecast_1} = \text{last\_value} \times \exp(\text{log-retur}_{\text{n}}) \end{tikzcd}$$

$$\boldsymbol{Forecast_t} = \text{Forecast}_{t-1} \times \exp(\text{log-retur}_t)$$

This approach restores the predicted series from log scale back to the original units of electricity generation, enabling interpretable and actionable insights.

4.3 Actual Solar Power Forecast

The forecast plot (**Figure 35**) illustrates the predicted solar power output for the next 10 days using the ARMA(1,2)+GARCH(2,0) model. The resulting curve (in blue) shows a smooth and modest upward trend in electricity output. The absence of sharp fluctuations in the forecasted values reflects a period of relatively low volatility, as estimated by the GARCH component of the model.

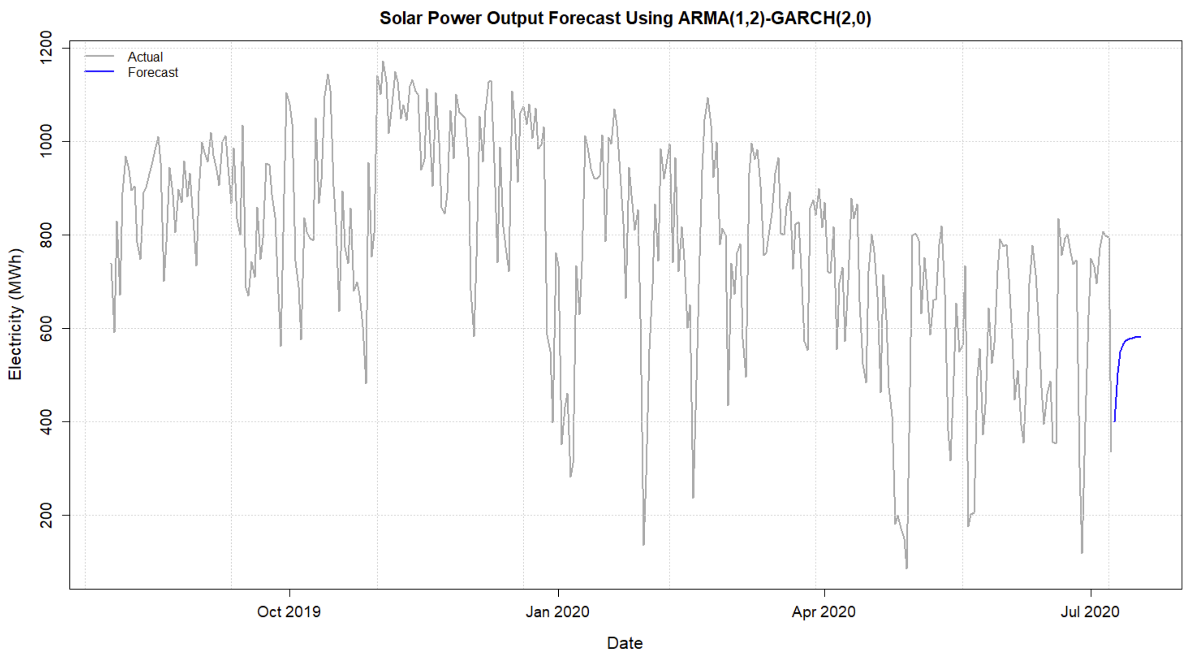


Figure 35.Actual Solar Power Forecast Using ARMA(1,2) +GARCH(1,2)

5. DISCUSSION

Throughout this analysis of solar output and the ARIMA-GARCH model selection process, serval key insights emerged:

(1) Data suitability.

The dataset used in this analysis consists of daily solar power generation values, which are appropriate for modeling short-term fluctuations and volatility in returns. However, a key limitation is that the model does not explicitly incorporate seasonal components, such as recurring weather patterns or annual solar cycles. Since solar output is inherently influenced by seasonal trends (e.g., daylight hours and cloud cover), excluding these factors may reduce the model's long-term explanatory power and lead to biased forecasts during transitional periods (e.g., summer to winter).

(2) Model Efeectiveness

The ARMA(1,2)+GARCH(2,0) framework proved effective in modeling the time series. The ARMA component successfully captured the short-term autocorrelation in the mean structure, while the GARCH component addressed the presence of volatility clustering in the residuals. Residual diagnostic plots confirmed the adequacy of the model, with no significant autocorrelation remaining and relatively stable residuals.

The AIC/BIC comparisons showed that simpler GARCH structures such as GARCH(2,0) offered better trade-offs between fit and complexity, while avoiding overfitting.

(3) Forecast Interpretability and Performance.

The model produces realistic and interpretable predictions. Although external factors such as cloud cover and seasonality were not explicitly included, the model still captured short-term dynamics effectively. These forecasts can provide valuable insights for short-term operational planning

(4) Directions for Future Work.

Future improvements could include incorporating variables such as temperature, cloud cover, or solar irradiance, which could significantly enhance the forecast accuracy. Additionally, exploring more flexible volatility models such as EGARCH or GJR-GARCH, or switching to machine learning-based hybrid models, may offer improved performance in capturing asymmetries and nonlinear behavior in the data.

6. CONCLUSION

In this report, a log return transformation was applied to the daily solar power generation data to stabilize variance and capture short-term fluctuations. An ARMA(1,2) model was selected to model the mean structure of the series, effectively capturing short-term autocorrelation patterns. After fitting and evaluating multiple candidate GARCH models, ARMA(1,2)+GARCH(2,0) was selected as the best-performing specification based on AIC, BIC, and residual diagnostics. The chosen model was then used to generate a 10-day ahead forecast. The predicted returns were transformed back into actual electricity generation values using an exponential transformation. The resulting forecast exhibited a stable and realistic trend that aligned well with recent observations. Overall, the model demonstrated strong capability in capturing short-term dynamics and offered interpretable, actionable forecasts that can support operational planning in solar energy systems.

The model developed in this study can be applied in various ways to support short-term solar energy forecasting and operational decision-making. Its ability to capture both the expected generation levels and the associated volatility makes it particularly useful for day-ahead forecasting, battery storage optimization, and grid reliability planning.

REFERENCE

Sun, H., Yan, D., Zhao, N., & Zhou, J. (2015). Empirical investigation on modeling solar radiation series with ARMA–GARCH models. *Energy Conversion and Management, 92*, 385–395. https://doi.org/10.1016/j.enconman.2014.12.072

Atique, S., Noureen, S., Roy, V., Subburaj, V., Bayne, S., & Macfie, J. (2019). Forecasting of total daily solar energy generation using ARIMA: A case study. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 114–119). IEEE. https://doi.org/10.1109/CCWC.2019.8666481

USE OF GENERATIVE AI TOOLS IN COMPLETING THE ASSIGNMENT

In completing this assignment, I used ChatGPT to enhance my work in three key areas:

1. Coding Assistance

ChatGPT helped identify and fix coding errors and improve code efficiency during data analysis in RStudio.

2. Refining Explanations

I consulted ChatGPT for guidance on describing my results clearly and effectively.

3. Improving Language and Structure

I used ChatGPT to refine the structure, flow, and coherence of my report and speech content.

To ensure accuracy, I cross-referenced insights with course notes, research papers, and I confirmed model diagnostics using R outputs. I also carefully reviewed AI-generated content to ensure precision and relevance.

R CODES

```
# upload the following libraries
library(TSA)
library(fUnitRoots)
library(forecast)
library(tseries)
```

```
    library(lmtest)
    library(zoo)
    library(rugarch)
    # ====================================
    # Descriptive Statistic analysis
    # ====================================

    df1<-read.csv("C:/Users/Yulin He/Documents/Datascience/9.Time Series
Analysis/Final Project/daily_total_energy.csv")

    #summary statistics
    summary_stats <- data.frame(
      Min = min(ts_daily),
      Max = max(ts_daily),
      Mean = mean(ts_daily),
      Median = median(ts_daily),
      SD = sd(ts_daily)
    )

    print(summary_stats)
    #check days
    length(df1$total_kwh)
    #check class
    class(df1)
    #transform date and keep the right order
    df1$date <- as.Date(df1$date)
    ts_zoo <- zoo(df1$total_kwh, order.by = df1$date)
    plot(ts_zoo, type = "l",
         xlab = "Date", ylab = "Energy (kWh)",
         main = "Daily Solar Energy Production",
         col = "blue")

    # Convert data to time series object
    ts_daily <- ts(coredata(ts_zoo), start = c(2019, as.numeric(format(start(ts_zoo),
"%j"))), frequency = 365)
    plot(ts_daily, type = "l",
         xlab = "Date", ylab = "Daily Solar Power Output(kWh)",
         col = "blue")

    class(ts_daily)
    head(ts_daily)
```

```
# Autocorrelation Analysis
# First-order autocorrelation
y<-ts_daily
x<- zlag(ts_daily)
index <- 2:length(x)
cor(y[index],x[index]) #0.7630352
# Lag plot for first-order autocorrelation
plot(y[index], x[index],
    ylab = "Solar Power(kWh)",
    xlab = "Previous Energy",
    main = "Scatter plot of the daily solar energy in consecutive days")

#Normally distribution
qqnorm(ts_daily, main="Q-Q Normal Plot of Daily Solar Energy")
qqline(ts_daily)
#Shapiro-Wilk normality test
shapiro.test(ts_daily)

#===Stationarity Testing===
# ADF test to check whether solar.ts is stationary or not
adf.test(ts_daily)
pp.test(ts_daily)
#p-value = 0.01, stationarity

# ACF & PACF plot
par(mfrow=c(1,2))
acf(ts_daily,main = "ACF plot",lag.max = 100)
pacf(ts_daily, main = 'PACF plot',lag.max = 100)
par(mfrow=c(1,1))

##changing variance
r.ts_daily=diff(log(ts_daily))*100
plot(r.ts_daily, ylab='Return ts_daily', xlab='Time', type='o',
    main = "Time series plot of the differenced and transformed daily
energy.")

adf.test(r.ts_daily)
pp.test(r.ts_daily)
McLeod.Li.test(y=r.ts_daily,main="McLeod-Li test statistics for daily energy
series")

qqnorm(r.ts_daily, main="Q-Q Normal Plot of Daily Solar Energy")
```

```
	qqline(r.ts_daily)
	shapiro.test(r.ts_daily)


	#We will first specify the orders of ARMA part.
	#Then we will use the residuals of ARMA part to specify the orders of GARCH
part.
	#We will use returns series to specify the orders of ARMA part following the
model specification steps.
	# ===================================
	# Model Identification - ARMA Part
	# ===================================
	# ACF & PACF plot
	par(mfrow=c(1,2))
	acf(as.numeric(r.ts_daily),main = "log-differenced transformation ACF
plot",lag.max = 20)
	pacf(as.numeric(r.ts_daily), main = 'log-differenced transformation PACF
plot',lag.max = 20)
	par(mfrow=c(1,1))
	#{ARMA(1,1), ARMA(2,1), ARMA(3,1), ARMA(4,1)}  can be identified

	#EACF
	eacf(r.ts_daily, ar.max = 5, ma.max = 5)
	#{ARMA(0,2) ARIMA(1,2) ARIMA(2,2)}

	# BIC table
	par(mfrow=c(1,1))
	res = armasubsets(y=r.ts_daily,nar=5,nma=5,y.name='p',ar.method='ols')
	plot(res)

	#{ARMA(0,1) ARIMA(0,2) ARIMA(2,1) ARIMA(2,2)  ARIMA(2,5)}

	#totally: {ARMA(0,1), ARMA(0,2), ARMA(1,1), ARMA(1,2), ARMA(2,1),
ARMA(2,2), ARMA(2,5), ARMA(3,1), ARMA(4,1)}

	#====residual diagnostics====
	residual.analysis <- function(model, std = TRUE, start = 2,
	                              class = c("ARIMA","GARCH","ARMA-
GARCH","garch","fGARCH","rugarch"),
	                              max.lag = 30, title = NULL) {
	  library(TSA)
	  class <- match.arg(class)
```

```
        if (class == "ARIMA") {
          res.model <- if (std) rstandard(model) else residuals(model)
        } else if (class %in% c("GARCH", "garch")) {
          res.model <- model$residuals[start:model$n.used]
        } else if (class == "ARMA-GARCH") {
          res.model <- model@fit$residuals
        } else if (class == "fGARCH") {
          res.model <- model@residuals
        } else if (class == "rugarch") {
          res.model <- residuals(model, standardize = std)
          res.model <- as.numeric(res.model)
        }else {
          stop("The argument 'class' must be one of the predefined types.")
        }

        # Ljung-Box test p-values
        lbq.pvals <- sapply(1:max.lag, function(lag) Box.test(res.model, lag = lag,
type = "Ljung-Box")$p.value)

        par(mfrow = c(2, 3))

        # Diagnostic plots
        plot(res.model, type = 'o', ylab = 'Standardised residuals', main = "Time
series plot of residuals")
        abline(h = 0)
        hist(res.model, main = "Histogram of residuals", col = "lightgray")
        qqnorm(res.model, main = "QQ plot of residuals")
        qqline(res.model, col = 2)
        acf(as.numeric(res.model), main = "ACF of residuals")
        pacf(as.numeric(res.model), main = "PACF of residuals")

        # Ljung-Box p-value plot only
        plot(1:max.lag, lbq.pvals, type = "b", pch = 16, col = "darkgreen",
             main = "Ljung-Box test p-values", xlab = "Lag", ylab = "p-value",
             ylim = c(0, 1))
        abline(h = 0.05, col = "red", lty = 2)

        if (!is.null(title)) {
          mtext(title, outer = TRUE, line = -1.5, cex = 1.5, font = 2)
        }

        par(mfrow = c(1, 1))
```

```
    # Shapiro-Wilk test
    cat("\nShapiro-Wilk test:\n")
    print(shapiro.test(res.model))
  }


  #=====build a model fitting function====
  fit_arima_models <- function(ts_data, orders, std = TRUE, start = 2) {

    #using ML and CSS methods to fit ARIMA model
    #print the coeftest results
    results <- list()

    for (order in orders) {
      order_str <- paste(order, collapse = "")

      cat("\n===== ARIMA(", paste(order, collapse = ","), ") with ML
=====\n")
      model_ml <- arima(ts_data, order = order, method = "ML", include.mean
= TRUE)
      print(coeftest(model_ml))
      results[[paste0("model", order_str, "_ML")]] <- model_ml

      cat("\n===== ARIMA(", paste(order, collapse = ","), ") with CSS
=====\n")
      model_css <- arima(ts_data, order = order, method = "CSS",
include.mean = TRUE)
      print(coeftest(model_css))
      results[[paste0("model", order_str, "_CSS")]] <- model_css

      cat("Residual diagnostics for ARIMA(", paste(order, collapse = ","), ")
CSS:\n")
      residual.analysis(model = model_css, std = std, start = start, class =
"ARIMA",
                    title = paste0("Residual Diagnostics for ARMA(", order[1], ",",
order[3], ")"))
    }

    return(results)
  }
```

```
orders_to_test <-
list(c(0,0,1),c(0,0,2),c(1,0,1),c(1,0,2),c(2,0,1),c(2,0,2),c(2,0,5),c(3,0,1),c(4,0,1))

results <- fit_arima_models(r.ts_daily, orders_to_test)

# ===================================
# Comparised AIC and BIC
# ===================================
#AIC/BIC SCORE
extract_and_sort_ic <- function(model_list, score = c("aic", "bic")) {
  score <- match.arg(score)

  ic_values <- sapply(names(model_list), function(name) {
    model <- model_list[[name]]
    val <- tryCatch(
      if (score == "aic") AIC(model) else BIC(model),
      error = function(e) NA
    )
    return(val)
  })
  # delete NA models
  ic_values <- ic_values[!is.na(ic_values)]
  # order the results
  df <- data.frame(Model = names(ic_values), IC = ic_values)
  df <- df[order(df$IC), ]
  rownames(df) <- NULL
  return(df)
}

extract_and_sort_ic(results[grep("_ML$", names(results))], score = "aic")
extract_and_sort_ic(results[grep("_ML$", names(results))], score = "bic")

#model_102 is the best
# ===================================
# Accuracy Evaluation
# ===================================
model_001_css = Arima(r.ts_daily,order=c(0,0,1), method='CSS')
model_002_css = Arima(r.ts_daily,order=c(0,0,2), method='CSS')
model_101_css = Arima(r.ts_daily,order=c(1,0,1), method='CSS')
model_102_css = Arima(r.ts_daily,order=c(1,0,2), method='CSS')
model_201_css = Arima(r.ts_daily,order=c(2,0,1), method='CSS')
```

```
    model_202_css = Arima(r.ts_daily,order=c(2,0,2), method='CSS')
    model_205_css = Arima(r.ts_daily,order=c(2,0,5), method='CSS')
    model_301_css = Arima(r.ts_daily,order=c(3,0,1), method='CSS')
    model_401_css = Arima(r.ts_daily,order=c(4,0,1), method='CSS')
    # get accuracy results
    Smodel_001_css <- accuracy(model_001_css)[1:7]
    Smodel_002_css <- accuracy(model_002_css)[1:7]
    Smodel_101_css <- accuracy(model_101_css)[1:7]
    Smodel_102_css <- accuracy(model_102_css)[1:7]
    Smodel_201_css <- accuracy(model_201_css)[1:7]
    Smodel_202_css <- accuracy(model_202_css)[1:7]
    Smodel_205_css <- accuracy(model_205_css)[1:7]
    Smodel_301_css <- accuracy(model_301_css)[1:7]
    Smodel_401_css <- accuracy(model_401_css)[1:7]
    # cpllect the results in one data frame
    df.Smodels <- data.frame(
      rbind(Smodel_001_css, Smodel_002_css, Smodel_101_css, Smodel_102_css,
          Smodel_201_css, Smodel_202_css, Smodel_205_css, Smodel_301_css,
Smodel_401_css))

    colnames(df.Smodels) <- c("ME", "RMSE", "MAE", "MPE", "MAPE", "MASE",
"ACF1")
    rownames(df.Smodels) <- c("ARIMA(0,0,1)", "ARIMA(0,0,2)",
"ARIMA(1,0,1)","ARIMA(1,0,2)",
                              "ARIMA(2,0,1)", "ARIMA(2,0,2)", "ARIMA(2,0,5)",
"ARIMA(3,0,1)", "ARIMA(4,0,1)")
    round(df.Smodels,  digits = 3)

    #ARMA(1,2) fitted to the energy return series is the best model

    #Overfitting models are ARMA(2,2) and ARMA(1,3)
    model_103 = arima(r.ts_daily,order=c(1,0,3), method='ML')
    coeftest(model_103)

    model_103_css = arima(r.ts_daily,order=c(1,0,3), method='CSS')
    coeftest(model_103_css)
    residual.analysis(model = model_103_css)

    orders_to_test <-
list(c(0,0,1),c(0,0,2),c(1,0,1),c(1,0,2),c(1,0,3),c(2,0,1),c(2,0,2),c(2,0,5),c(3,0,1),c(4,0,1))
    results <- fit_arima_models(r.ts_daily, orders_to_test)
    extract_and_sort_ic(results[grep("_ML$", names(results))], score = "aic")
    extract_and_sort_ic(results[grep("_ML$", names(results))], score = "bic")


    model_103_css = Arima(r.ts_daily,order=c(1,0,3), method='CSS')
    Smodel_103_css <- accuracy(model_103_css)[1:7]
    # cllect the results in one data frame
    df.Smodels <- data.frame(
      rbind(Smodel_001_css, Smodel_002_css, Smodel_101_css, Smodel_102_css,
Smodel_103_css,
            Smodel_201_css, Smodel_202_css, Smodel_205_css, Smodel_301_css,
Smodel_401_css))

    colnames(df.Smodels) <- c("ME", "RMSE", "MAE", "MPE", "MAPE", "MASE",
"ACF1")
    rownames(df.Smodels) <- c("ARIMA(0,0,1)",
"ARIMA(0,0,2)","ARIMA(1,0,1)","ARIMA(1,0,2)","ARIMA(1,0,3)","ARIMA(2,0,1)","ARIM
A(2,0,2)","ARIMA(2,0,5)","ARIMA(3,0,1)","ARIMA(4,0,1)")
    round(df.Smodels,  digits = 3)

    #===================================
    #Model Identification - GARCH Part
    #====================================
    #==== use the residuals of ARMA(1,2) model to identify the orders of
GARCH.====
    model_102 <- arima(r.ts_daily, order = c(1,0,2), method='ML')
    m12residuals = model_102$residuals
    abs.res = abs(m12residuals)
    sq.res = m12residuals^2
    par(mfrow=c(1,2))
    acf(as.numeric(abs.res), ci.type = "ma", main="ACF plot for absolute return
series")
    pacf(as.numeric(abs.res), main="PACF plot for absolute return series ")
    par(mfrow=c(1,1))
    eacf(abs.res)
    #GARCH(0,1) GARCH(1,1) GARCH(2,1) GARCH(0,2) , GARCH(0,2) ,
GARCH(2,2)
    par(mfrow=c(1,2))
    acf(as.numeric(sq.res), ci.type = "ma",main = "ACF plot for square return
series")
    pacf(as.numeric(sq.res), main = "PACF plot for square return series")
    par(mfrow=c(1,1))
    eacf(sq.res)
    #Overall:{GARCH(0,1), GARCH(0,2), GARCH(1,1), GARCH(1,2), GARCH(2,0),
GARCH(2,1),GARCH(2,2)}

    #====fit these models using ugarchfit() function from rugarch package.
====
    #====fit ARIMA(1,0,2)+GARCH(p,q)====
    fit_arma12_garch_models_minimal <- function(data,
                                          garch_orders = list(c(0,1), c(0,2), c(1,1), c(1,2),
c(2,0), c(2,1), c(2,2)),
                                          plot_path = "garch_diagnostics") {
      dir.create(plot_path, showWarnings = FALSE)

      model_names <- c()
      aic_vec <- c()
      bic_vec <- c()

      for (order in garch_orders) {
        p <- order[1]
        q <- order[2]
        model_name <- paste0("ARMA(1,2)+GARCH(", p, ",", q, ")")
        cat("\n==============================\nFitting:", model_name,
"\n")

        spec <- ugarchspec(
          variance.model = list(model = "sGARCH", garchOrder = c(p, q)),
          mean.model = list(armaOrder = c(1, 2), include.mean = TRUE),
          distribution.model = "norm"
        )

        fit <- tryCatch({
          ugarchfit(spec, data = data)
        }, error = function(e) {
          warning(paste("Failed to fit", model_name, ":", e$message))
          return(NULL)
        })

        if (!is.null(fit)) {
          show(fit)


          cat("Residual diagnostics for", model_name, ":\n")
          par(mfrow = c(3, 2), oma = c(0, 0, 2, 0))
          residual.analysis(fit, class = "rugarch")
          mtext(model_name, outer = TRUE, cex = 1.3, font = 2)
          par(mfrow = c(1, 1))  # reset

          #AIC / BIC
          ic <- infocriteria(fit)
          aic_vec <- c(aic_vec, ic[1])
          bic_vec <- c(bic_vec, ic[2])
          model_names <- c(model_names, model_name)
        }
      }

      result_df <- data.frame(Model = model_names, AIC = aic_vec, BIC =
bic_vec)
      result_df <- result_df[order(result_df$AIC), ]
      rownames(result_df) <- NULL
      return(result_df)
    }


    result_table <- fit_arma12_garch_models_minimal(r.ts_daily)
    print(result_table)

    # ==================================
    #ARMA(1,2)+GARCH(2,0) forecasting.
    #==================================
    library(xts)
    r.xts <- xts(r.ts_daily, order.by = as.Date(df1$date[-1]))
    spec <- ugarchspec(
      variance.model = list(model = "sGARCH",garchOrder = c(2, 0)),
      mean.model = list(armaOrder = c(1, 2), include.mean = TRUE),
      distribution.model = "norm")

    model_12_20_fit <- ugarchfit(spec = spec, data = r.xts,
                                 solver = "hybrid",
                                 solver.control = list(trace=0))

    par(mfrow=c(1,1))
    plot(model_12_20_fit, which = 1)

    # 10 day ahead forecast
```

```
frc <- ugarchforecast(model_12_20_fit,n.ahead=10,data=r.ts_daily)
frc
class(frc)
plot(frc, which = 1) ## Forecasted return series + conditional SD bands
plot(frc, which = 3) ## Forecasted conditional sigma (volatility)

#====better visualization====
plot_garch_forecast_with_ci <- function(
    return_past,
    forecast_object,
    date_vector,
    save_path = NULL
) {
  # date
  dates_past <- as.Date(date_vector)[-1]
  n_forecast <- length(forecast_object@forecast$seriesFor)
  dates_future <- seq(from = tail(dates_past, 1) + 1, by = "day", length.out =
n_forecast)
  dates_all <- c(dates_past, dates_future)

  # data
  returns_full <- c(return_past, rep(NA, n_forecast))
  forecast_returns <- c(rep(NA, length(return_past)),
forecast_object@forecast$seriesFor)
  sigma_forecast <- forecast_object@forecast$sigmaFor
  upper <- forecast_object@forecast$seriesFor + 2 * sigma_forecast
  lower <- forecast_object@forecast$seriesFor - 2 * sigma_forecast

  upper_full <- c(rep(NA, length(return_past)), upper)
  lower_full <- c(rep(NA, length(return_past)), lower)

  par(bg = "white", mar = c(4, 4, 3, 2))
  plot(dates_all, returns_full, type = "l", col = "blue", lwd = 1,
     ylab = "Return (%)", xlab = "Date",
     main = "Actual + Forecasted Return with ±2σ Band",
     ylim = range(c(returns_full, upper, lower), na.rm = TRUE))

  polygon(
    x = c(dates_future, rev(dates_future)),
    y = c(lower, rev(upper)),
    col = rgb(1, 0, 0, 0.2), border = NA)
  lines(dates_all, forecast_returns, col = "red", lwd = 2)
  legend("topleft", legend = c("Actual", "Forecast", "±2σ Band"),
         col = c("blue", "red", rgb(1, 0, 0, 0.2)), lty = c(1, 1, NA),
         lwd = c(2, 2, NA), pch = c(NA, NA, 15), pt.cex = 2, bty = "n")

  grid()

  if (!is.null(save_path)) dev.off()
}


plot_garch_forecast_with_ci(
  return_past = r.ts_daily,
  forecast_object = frc,
  date_vector = df1$date
)


# All the transformations:To get the actual forecasts, we need to take the
tansformation and differencing back.
#r.ts_daily = diff(log(ts_daily))*100
last_value <- tail(ts_daily, 1)  #last observation
r_forecast <- frc@forecast$seriesFor  # return
r_forecast

n <- length(r_forecast)
x_pred <- numeric(n)
x_pred[1] <- last_value * exp(r_forecast[1] / 100)

for (i in 2:n) {
  x_pred[i] <- x_pred[i - 1] * exp(r_forecast[i] / 100)
}
print(x_pred)


# x-index
dates <- as.Date(df1$date)
last_date <- tail(dates, 1)
future_dates <- seq(from = last_date + 1, by = "day", length.out = n)
time_index <- c(dates, future_dates)

# original data + predicted data
ts_full <- c(ts_daily, rep(NA, n))
```

```
pred_full <- c(rep(NA, length(ts_daily)), x_pred)

par(bg = "white", mar = c(4, 4, 3, 2), cex.axis = 1.2, cex.lab = 1.3, cex.main =
1.4)
plot(time_index, ts_full, type = "l", col = "darkgrey", lwd = 2,
    ylab = "Electricity (MWh)", xlab = "Date",
    main = "Solar Power Output Forecast Using ARMA(1,2)-GARCH(2,0)",
    ylim = range(c(ts_full, pred_full), na.rm = TRUE))
lines(time_index, pred_full, col = "blue", lwd = 2)
legend("topleft", legend = c("Actual", "Forecast"),
       col = c("darkgrey", "blue"), lty = 1, lwd = 2, bty = "n")
grid(col = "lightgray", lty = "dotted")
box()
```