Applying ARIMA-SARIMA, GARCH models for time series analysis on Seasonal and Nonseasonal datasets

VATSAL KADAKIA

2024-04-29

PART A- Seasonal Time Series

Motivation and Introduction:

A seasonal time series of Hotel Hospitality Industry Employees. This dataset contains the number of employees in thousands of persons as monthly averages from 1990-01-01 to 2018-12-01. The goal of this project is to predict and forecast the monthly number of employees.

Dataset description

The dataset titled “HotelEmployees.csv” comprises 348 records spread across two columns: ‘Date’ and ‘Employees’. Each row represents monthly data from January 1990, indicating the number of employees in the hotel industry. The ‘Date’ column, formatted as strings, provides the month and year for each entry, while the ‘Employees’ column, a float, specifies the exact count of employees during that month.

Statistically, the employee count ranges from a minimum of 1064.5 to a maximum of 2022.1, with an average of approximately 1452.5 employees, indicating fluctuations in employment numbers over the observed period. Each date in the dataset is unique, ensuring there are no duplicate entries for any month, thereby facilitating a precise monthly trend analysis of employment in the hotel sector.

Best model and summary

From ACF, PACF and Multiplicative decomposition graphs, it clearly indicates that it is a sesonal time series. It is not a financial time series so ARIMA models proved to best. Applied first differencing to make the data stationary. Tried manually to apply different combinations with seasonal orders. Compared all the AIC, BIC, loglikelihood values and SARIMA(5,1,0)X(3,0,0) was the best model with lowest BIC value. Carried out residual analysis - Shapiro Wilk test and Ljung Box test. Data was almost normal with test statistic W=0.97 and p-values also very low. Simple and straight forward execution of SARIMA models in a seasonal time series.

#Load the libraries

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.2.3
```

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

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.3
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
## Warning: package 'tibble' was built under R version 4.2.3
```

```
## Warning: package 'tidyr' was built under R version 4.2.3
```

```
## Warning: package 'readr' was built under R version 4.2.3
```

```
## Warning: package 'purrr' was built under R version 4.2.3
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
## Warning: package 'stringr' was built under R version 4.2.3
```

```
## Warning: package 'forcats' was built under R version 4.2.3
```

```
## Warning: package 'lubridate' was built under R version 4.2.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.1     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to be
come errors
```

```r
library(ggplot2)
library(stats)
library(TSA)
```

```
## Warning: package 'TSA' was built under R version 4.2.3
```

```
##
## Attaching package: 'TSA'
##
## The following object is masked from 'package:readr':
##
##     spec
##
## The following objects are masked from 'package:stats':
##
##     acf, arima
##
## The following object is masked from 'package:utils':
##
##     tar
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.2.3
```

```
## Registered S3 methods overwritten by 'forecast':
##   method       from
##   fitted.Arima TSA
##   plot.Arima   TSA
```

```
df = read_csv("C:/Users/vatsal/Downloads/HotelEmployees.csv")
```

```
## Rows: 348 Columns: 2
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): Date
## dbl (1): Employees
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
df <- df %>%
  mutate(Date = as.Date(Date, format = "%m/%d/%Y"))
str(df)
```

```
## tibble [348 × 2] (S3: tbl_df/tbl/data.frame)
##  $ Date     : Date[1:348], format: "1990-01-01" "1990-02-01" ...
##  $ Employees: num [1:348] 1064 1074 1090 1097 1109 ...
```

```
head(df)
```

Date <date>	Employees <dbl>
1990-01-01	1064.5

	Date <date>	Employees <dbl>
	1990-02-01	1074.5
	1990-03-01	1090.0
	1990-04-01	1097.4
	1990-05-01	1108.7
	1990-06-01	1123.5

6 rows

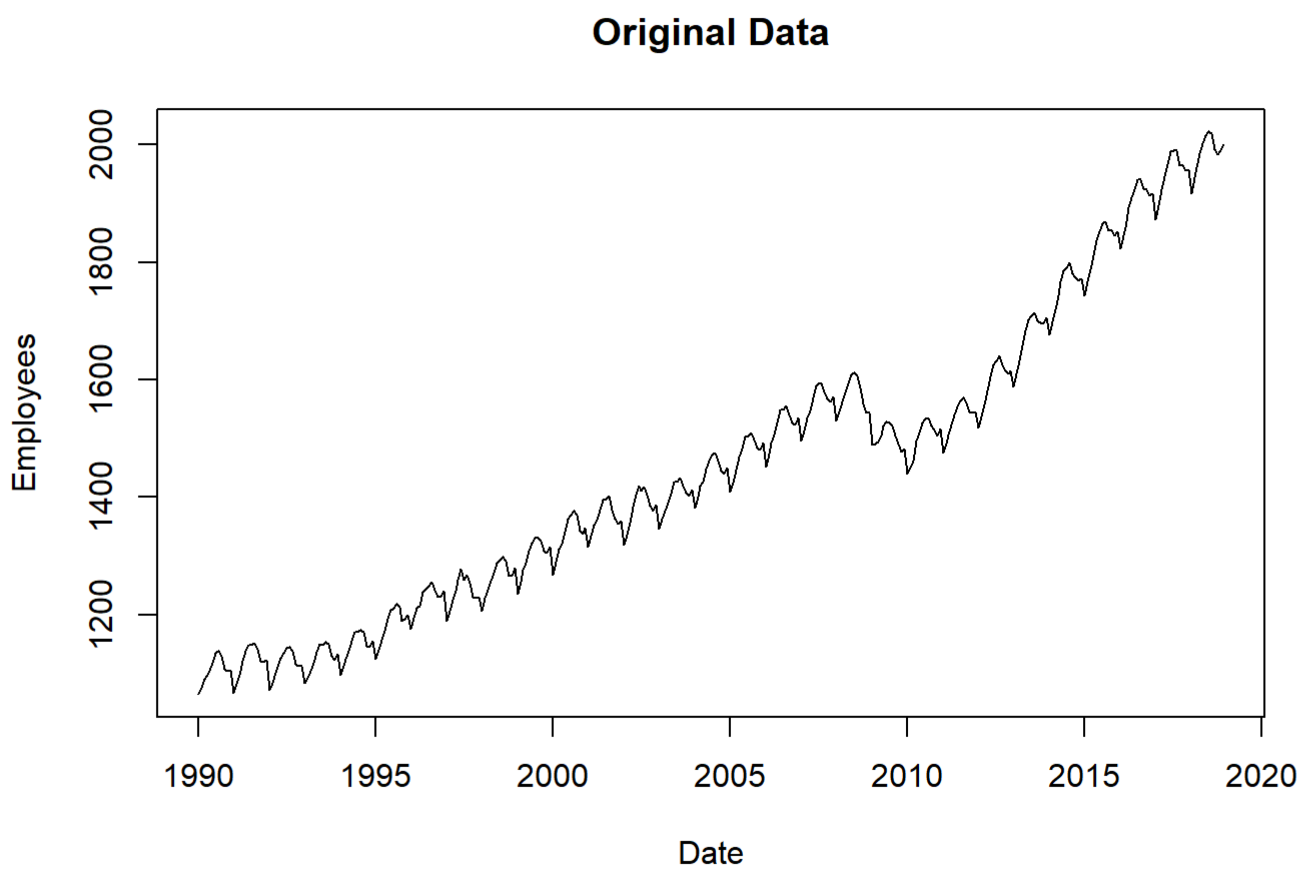
```
tail(df)
```

	Date <date>	Employees <dbl>
	2018-07-01	2022.1
	2018-08-01	2019.1
	2018-09-01	1992.5
	2018-10-01	1984.3
	2018-11-01	1990.1
	2018-12-01	2000.2

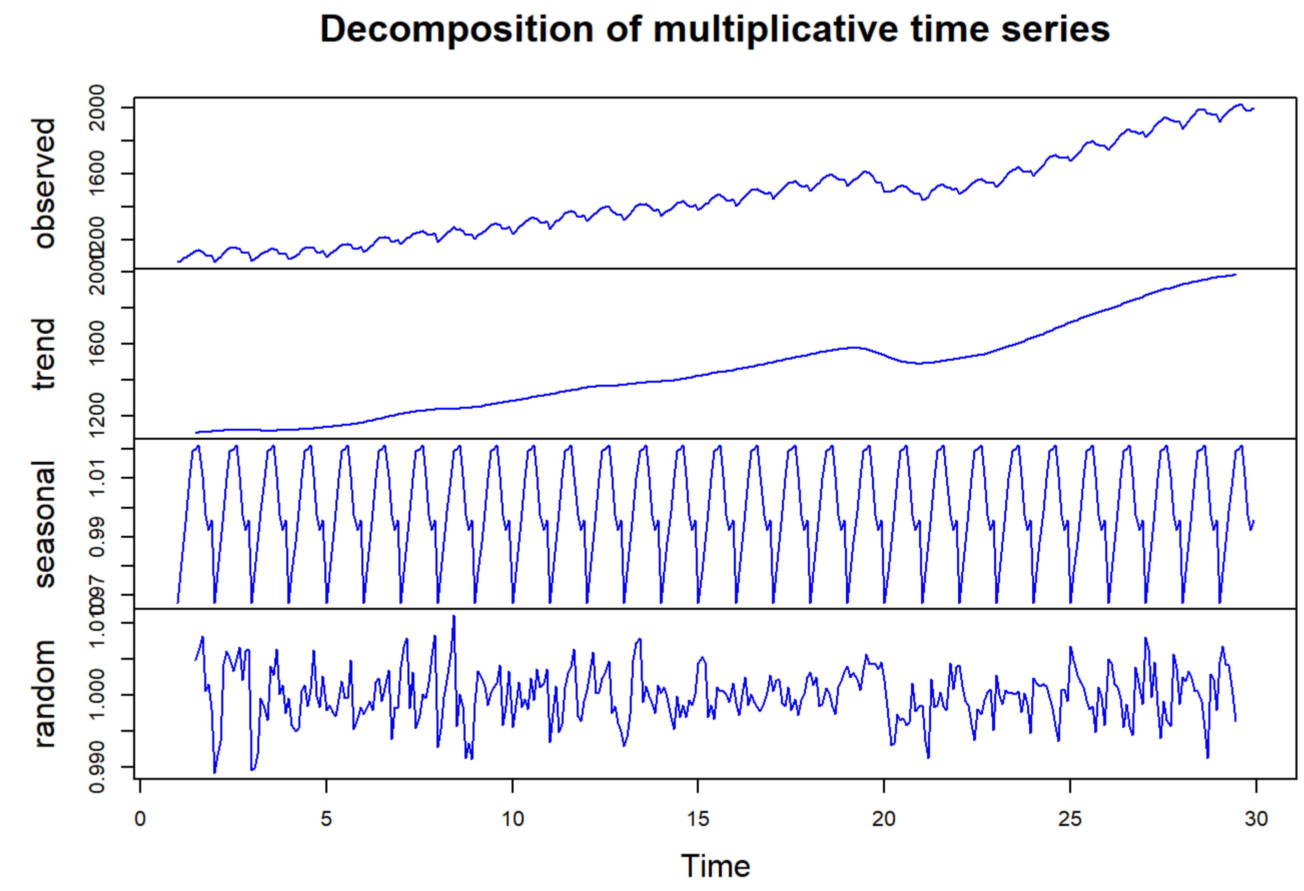
6 rows

#Original plot

```
plot(df, main='Original Data', type='l')
```



```
data = df$Employees
data_12 = ts(data, frequency = 12)
decompose_data = decompose(data_12, 'multiplicative')
plot(decompose_data, type='l', lwd=1, col='blue')
```

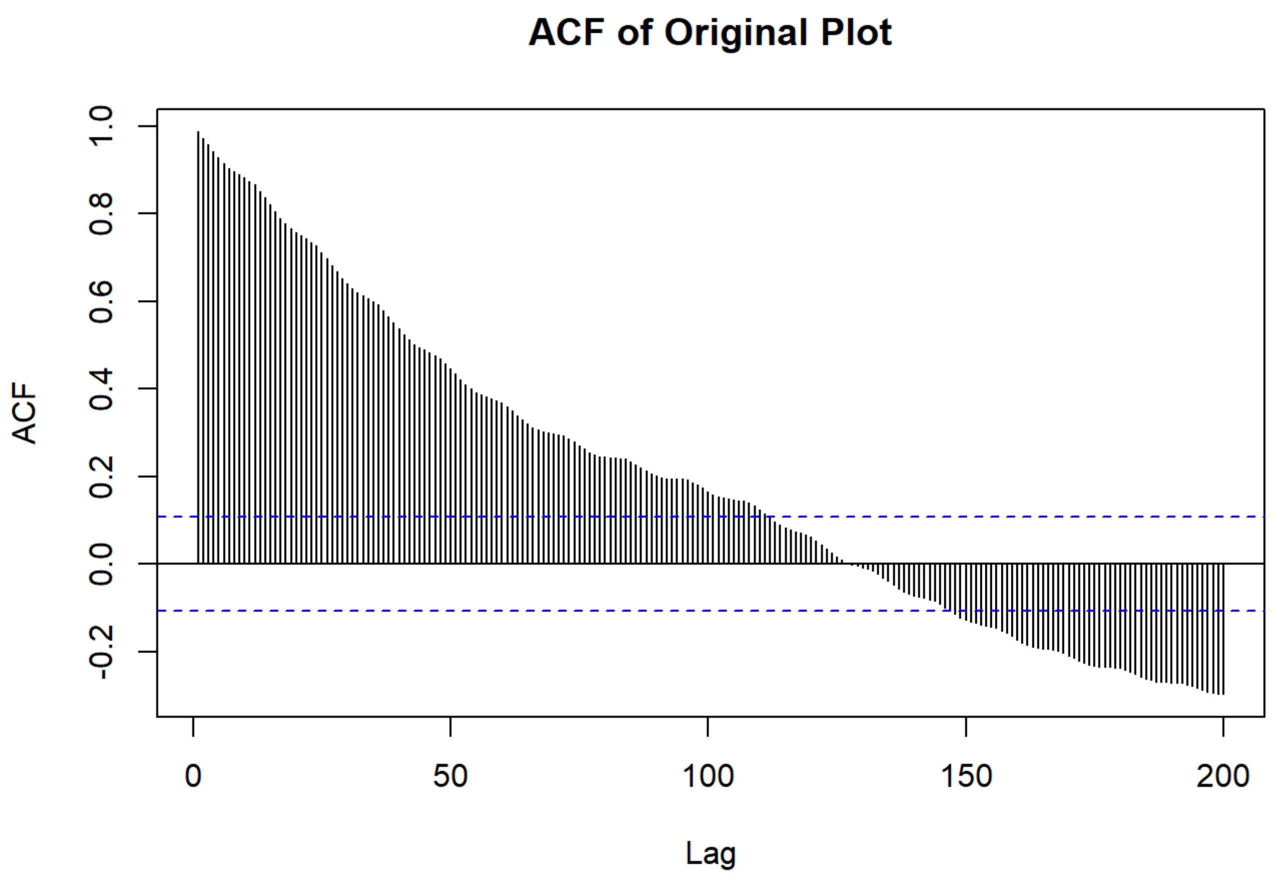


```
num_rows <- nrow(df)
train_data <- df[1:(num_rows-12),]
tail(train_data)
```

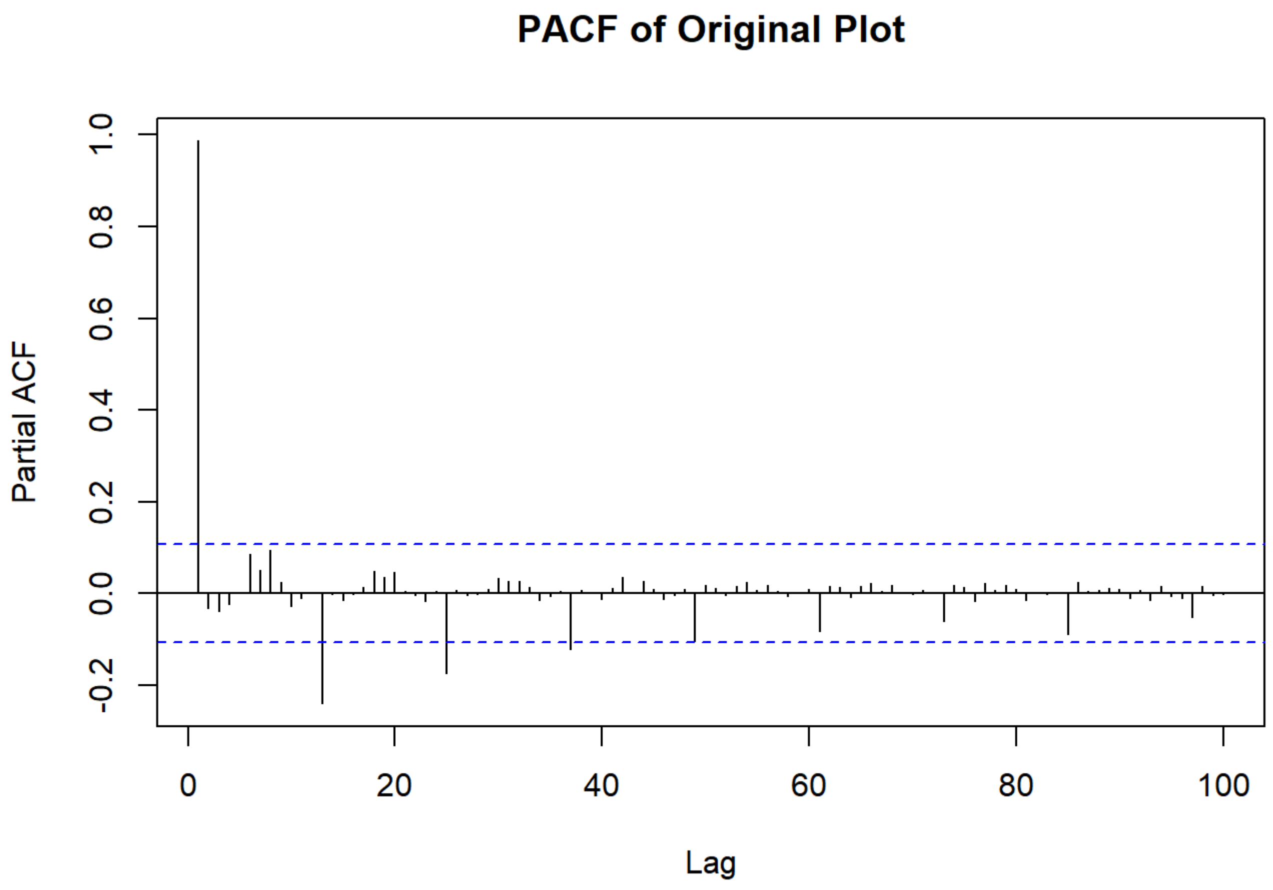
Date <date>	Employees <dbl>
2017-07-01	1990.9
2017-08-01	1989.9
2017-09-01	1964.9
2017-10-01	1965.2
2017-11-01	1956.6
2017-12-01	1957.2

6 rows

```
test_data <- tail(df, 12)

data = train_data$Employees
acf(data, lag.max = 200, main='ACF of Original Plot')
```



```
pacf(data, lag.max = 100, main='PACF of Original Plot')
```



Dickey Fuller test

```
adf_test = adf.test(data)
print(adf_test)
```

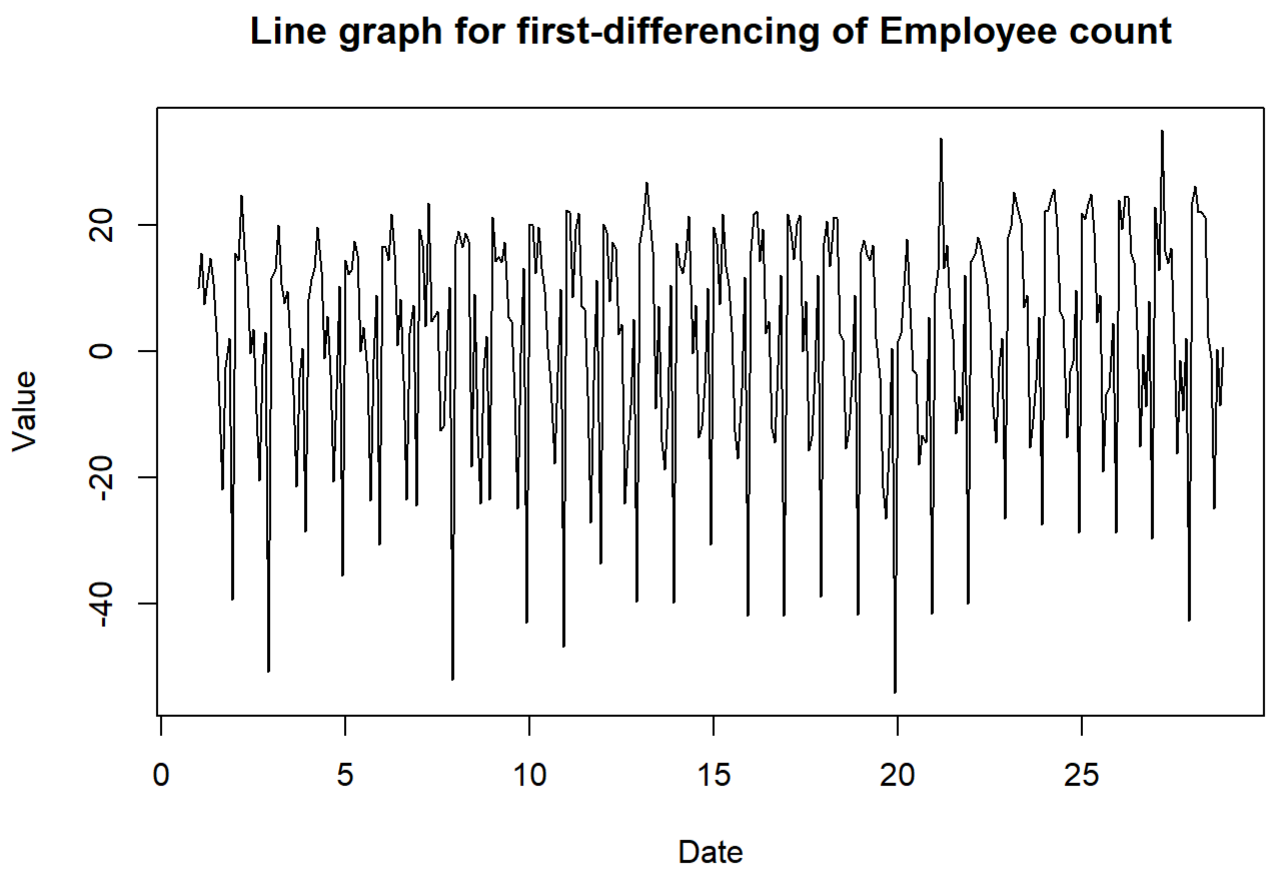
```
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -0.60192, Lag order = 6, p-value = 0.9769
## alternative hypothesis: stationary
```

```
print("Since p-value=0.9769 > 0.05, the TS is non-stationary")
```

```
## [1] "Since p-value=0.9769 > 0.05, the TS is non-stationary"
```

Differencing

```
employ_diff = diff(data)
employ_diff <- ts(employ_diff, frequency=12)
plot(employ_diff, type='l', xlab='Date', ylab='Value', main='Line graph for first-differencin
g of Employee count')
```



Differenced ACF and PACF plots

```
adf_test = adf.test(employ_diff)
```

```
## Warning in adf.test(employ_diff): p-value smaller than printed p-value
```

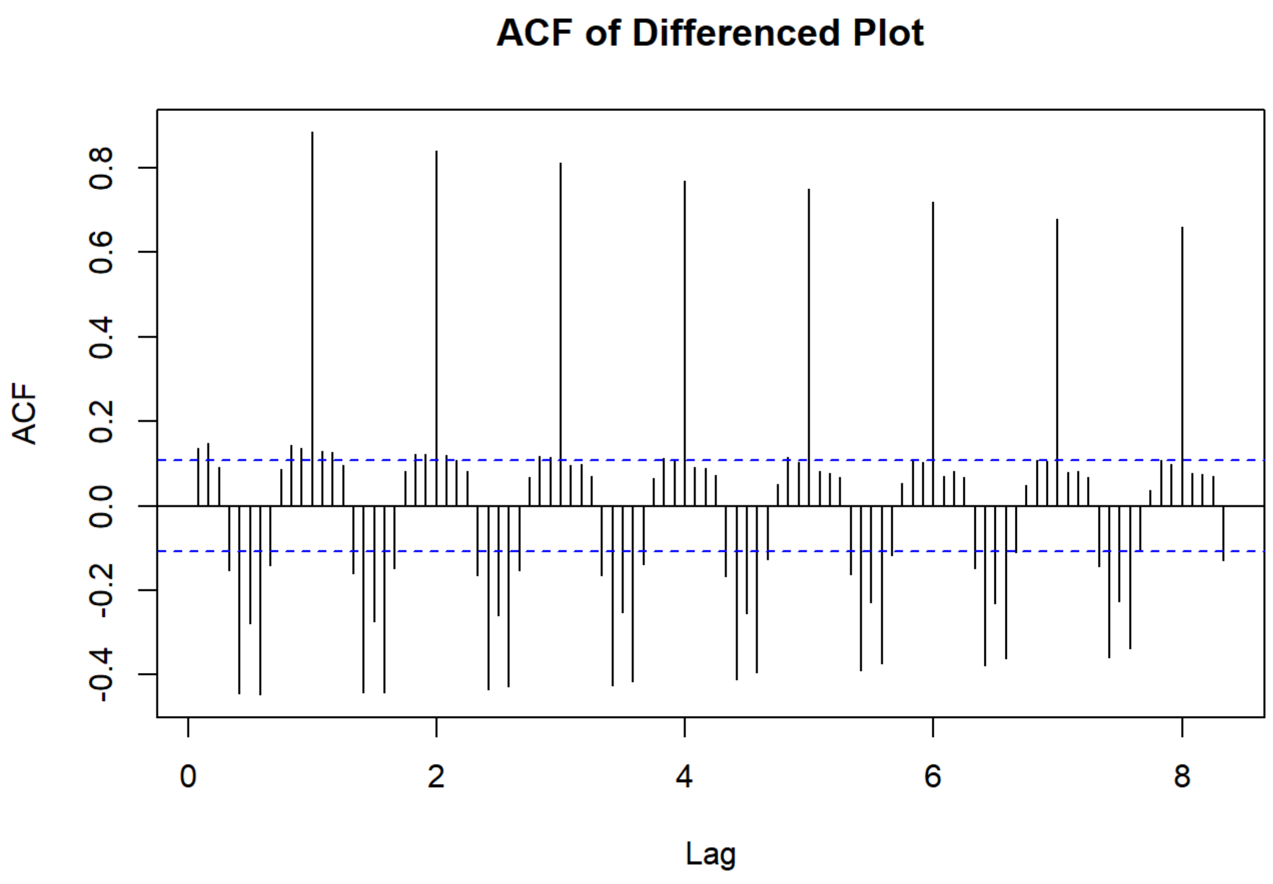
```
print(adf_test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  employ_diff
## Dickey-Fuller = -17.072, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

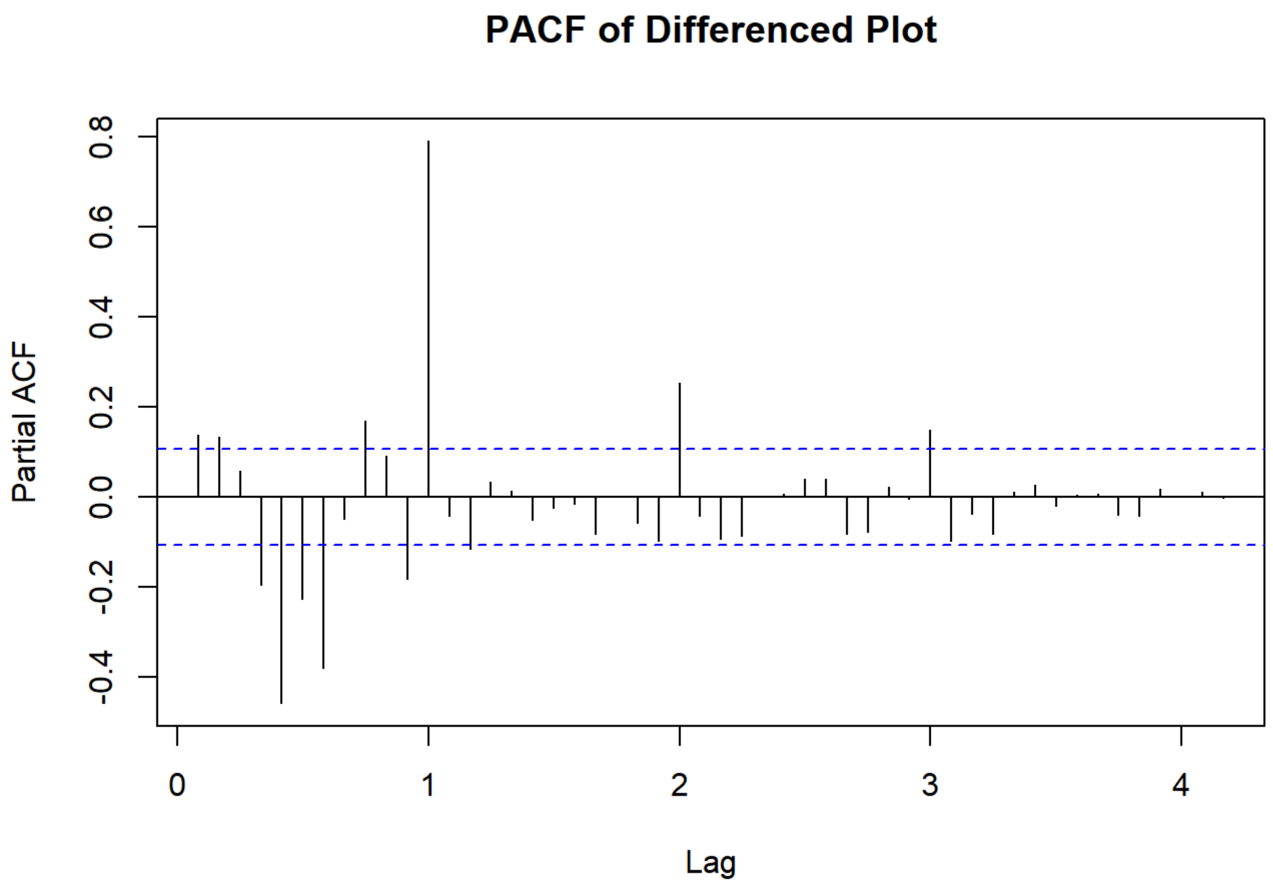
```
print("Since p-value=0.01 < 0.05, the TS is stationary")
```

```
## [1] "Since p-value=0.01 < 0.05, the TS is stationary"
```

```
acf(employ_diff, lag.max = 100, main='ACF of Differenced Plot')
```



```
pacf(employ_diff, lag.max = 50, main='PACF of Differenced Plot')
```



Running all combinations of SARIMA models

```
#
sarima110100 <- arima(employ_diff, order = c(1,1,0), seasonal = list(order = c(1,0,0),period=
12))
sarima110200 <- arima(employ_diff, order = c(1,1,0), seasonal = list(order = c(2,0,0),period=
12))
sarima110300 <- arima(employ_diff, order = c(1,1,0), seasonal = list(order = c(3,0,0),period=
12))

#
sarima210100 <- arima(employ_diff, order = c(2,1,0), seasonal = list(order = c(1,0,0),period=
12))
sarima210200 <- arima(employ_diff, order = c(2,1,0), seasonal = list(order = c(2,0,0),period=
12))
sarima210300 <- arima(employ_diff, order = c(2,1,0), seasonal = list(order = c(3,0,0),period=
12))

#
sarima310100 <- arima(employ_diff, order = c(3,1,0), seasonal = list(order = c(1,0,0),period=
12))
sarima310200 <- arima(employ_diff, order = c(3,1,0), seasonal = list(order = c(2,0,0),period=
12))
sarima310300 <- arima(employ_diff, order = c(3,1,0), seasonal = list(order = c(3,0,0),period=
12))

#
sarima410100 <- arima(employ_diff, order = c(4,1,0), seasonal = list(order = c(1,0,0),period=
12))
sarima410200 <- arima(employ_diff, order = c(4,1,0), seasonal = list(order = c(2,0,0),period=
12))
sarima410300 <- arima(employ_diff, order = c(4,1,0), seasonal = list(order = c(3,0,0),period=
12))

#
sarima510100 <- arima(employ_diff, order = c(5,1,0), seasonal = list(order = c(1,0,0),period=
12))
sarima510200 <- arima(employ_diff, order = c(5,1,0), seasonal = list(order = c(2,0,0),period=
12))
sarima510300 <- arima(employ_diff, order = c(5,1,0), seasonal = list(order = c(3,0,0),period=
12))

#
sarima610100 <- arima(employ_diff, order = c(6,1,0), seasonal = list(order = c(1,0,0),period=
12))
sarima610200 <- arima(employ_diff, order = c(6,1,0), seasonal = list(order = c(2,0,0),period=
12))
sarima610300 <- arima(employ_diff, order = c(6,1,0), seasonal = list(order = c(3,0,0),period=
12))

#
sarima710100 <- arima(employ_diff, order = c(7,1,0), seasonal = list(order = c(1,0,0),period=
12))
sarima710200 <- arima(employ_diff, order = c(7,1,0), seasonal = list(order = c(2,0,0),period=
12))
sarima710300 <- arima(employ_diff, order = c(7,1,0), seasonal = list(order = c(3,0,0),period=
12))
```

```
sarima110100
```

```
##
## Call:
## arima(x = employ_diff, order = c(1, 1, 0), seasonal = list(order = c(1, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1    sar1
##       -0.5727  0.9193
## s.e.   0.0447  0.0189
##
## sigma^2 estimated as 63.33:  log likelihood = -1178.08,  aic = 2360.16
```

```
sarima110200
```

```
##
## Call:
## arima(x = employ_diff, order = c(1, 1, 0), seasonal = list(order = c(2, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1    sar1    sar2
##       -0.5881  0.4533  0.5067
## s.e.   0.0442  0.0489  0.0497
##
## sigma^2 estimated as 47.85:  log likelihood = -1134.81,  aic = 2275.62
```

```
sarima110300
```

```
##
## Call:
## arima(x = employ_diff, order = c(1, 1, 0), seasonal = list(order = c(3, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1    sar1    sar2    sar3
##       -0.5732  0.2907  0.3438  0.3375
## s.e.   0.0448  0.0539  0.0549  0.0550
##
## sigma^2 estimated as 42.64:  log likelihood = -1117.39,  aic = 2242.78
```

```
sarima210100
```

```
##
## Call:
## arima(x = employ_diff, order = c(2, 1, 0), seasonal = list(order = c(1, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2    sar1
##       -0.7477  -0.3046  0.9231
## s.e.   0.0522   0.0523  0.0186
##
## sigma^2 estimated as 57.35:  log likelihood = -1161.91,  aic = 2329.83
```

```
sarima210200
```

```
##
## Call:
## arima(x = employ_diff, order = c(2, 1, 0), seasonal = list(order = c(2, 0, 0),
##     period = 12))
##
## Coefficients:
##          ar1      ar2    sar1    sar2
##       -0.778  -0.3212  0.4501  0.5138
## s.e.   0.052   0.0521  0.0483  0.0492
##
## sigma^2 estimated as 42.77:  log likelihood = -1116.82,  aic = 2241.65
```

```
sarima210300
```

```
##
## Call:
## arima(x = employ_diff, order = c(2, 1, 0), seasonal = list(order = c(3, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2    sar1    sar2    sar3
##       -0.7756  -0.3532  0.2659  0.3394  0.3710
## s.e.   0.0515   0.0517  0.0531  0.0533  0.0541
##
## sigma^2 estimated as 37.12:  log likelihood = -1095.59,  aic = 2201.18
```

```
sarima310100
```

```
##
## Call:
## arima(x = employ_diff, order = c(3, 1, 0), seasonal = list(order = c(1, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3    sar1
##       -0.8274  -0.4974  -0.2562  0.9322
## s.e.   0.0531   0.0646   0.0531  0.0172
##
## sigma^2 estimated as 53.38:  log likelihood = -1150.75,  aic = 2309.5
```

```
sarima310200
```

```
##
## Call:
## arima(x = employ_diff, order = c(3, 1, 0), seasonal = list(order = c(2, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3    sar1    sar2
##       -0.8544  -0.5071  -0.2391  0.4601  0.5068
## s.e.   0.0534   0.0655   0.0535  0.0487  0.0495
##
## sigma^2 estimated as 40.22:  log likelihood = -1107.14,  aic = 2224.28
```

```
sarima310300
```

```
##
## Call:
## arima(x = employ_diff, order = c(3, 1, 0), seasonal = list(order = c(3, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3    sar1    sar2    sar3
##       -0.8638  -0.5489  -0.2512  0.2739  0.3278  0.3772
## s.e.   0.0533   0.0651   0.0533  0.0532  0.0535  0.0536
##
## sigma^2 estimated as 34.65:  log likelihood = -1084.87,  aic = 2181.74
```

```
sarima410100
```

```
##
## Call:
## arima(x = employ_diff, order = c(4, 1, 0), seasonal = list(order = c(1, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4    sar1
##       -0.8945  -0.6239  -0.4657  -0.2463  0.9453
## s.e.   0.0534   0.0683   0.0691   0.0540  0.0150
##
## sigma^2 estimated as 49.86:  log likelihood = -1140.73,  aic = 2291.46
```

```
sarima410200
```

```
##
## Call:
## arima(x = employ_diff, order = c(4, 1, 0), seasonal = list(order = c(2, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4    sar1    sar2
##       -0.9150  -0.6325  -0.4517  -0.2459  0.4651  0.5079
## s.e.   0.0533   0.0692   0.0698   0.0538  0.0490  0.0496
##
## sigma^2 estimated as 37.58:  log likelihood = -1097.07,  aic = 2206.13
```

```
sarima410300
```

```
##
## Call:
## arima(x = employ_diff, order = c(4, 1, 0), seasonal = list(order = c(3, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4    sar1    sar2    sar3
##       -0.9190  -0.6670  -0.4415  -0.2181  0.2883  0.3316  0.3621
## s.e.   0.0538   0.0701   0.0703   0.0542  0.0537  0.0543  0.0544
##
## sigma^2 estimated as 32.88:  log likelihood = -1076.99,  aic = 2167.99
```

```
sarima510100
```

```
##
## Call:
## arima(x = employ_diff, order = c(5, 1, 0), seasonal = list(order = c(1, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5    sar1
##       -0.9430  -0.7126  -0.5847  -0.4134  -0.1798  0.9499
## s.e.   0.0547   0.0728   0.0772   0.0733   0.0542  0.0142
##
## sigma^2 estimated as 48.09:  log likelihood = -1135.33,  aic = 2282.65
```

```
sarima510200
```

```
##
## Call:
## arima(x = employ_diff, order = c(5, 1, 0), seasonal = list(order = c(2, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5    sar1    sar2
##       -0.9634  -0.7201  -0.5747  -0.4220  -0.1875  0.4655  0.5102
## s.e.   0.0543   0.0728   0.0775   0.0736   0.0544  0.0489  0.0496
##
## sigma^2 estimated as 36.12:  log likelihood = -1091.23,  aic = 2196.46
```

```
sarima510300
```

```
##
## Call:
## arima(x = employ_diff, order = c(5, 1, 0), seasonal = list(order = c(3, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5   sar1    sar2    sar3
##       -0.9553  -0.7379  -0.5514  -0.3700  -0.1601  0.295  0.3389  0.3493
## s.e.   0.0545   0.0734   0.0791   0.0747   0.0548  0.054  0.0547  0.0550
##
## sigma^2 estimated as 31.97:  log likelihood = -1072.79,  aic = 2161.58
```

```
sarima610100
```

```
##
## Call:
## arima(x = employ_diff, order = c(6, 1, 0), seasonal = list(order = c(1, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4     ar5      ar6    sar1
##       -0.9563  -0.7424  -0.6265  -0.4653  -0.250  -0.0735  0.9507
## s.e.   0.0555   0.0762   0.0834   0.0831   0.076   0.0559  0.0141
##
## sigma^2 estimated as 47.8:  log likelihood = -1134.46,  aic = 2282.93
```

```
sarima610200
```

```
##
## Call:
## arima(x = employ_diff, order = c(6, 1, 0), seasonal = list(order = c(2, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6    sar1    sar2
##       -0.9745  -0.7447  -0.6082  -0.4647  -0.2456  -0.0599  0.4678  0.5082
## s.e.   0.0552   0.0763   0.0834   0.0835   0.0765   0.0558  0.0490  0.0496
##
## sigma^2 estimated as 35.98:  log likelihood = -1090.65,  aic = 2197.31
```

```
sarima610300
```

```
##
## Call:
## arima(x = employ_diff, order = c(6, 1, 0), seasonal = list(order = c(3, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6    sar1    sar2
##       -0.9658  -0.7619  -0.5868  -0.4184  -0.2240  -0.0661  0.2975  0.3365
## s.e.   0.0552   0.0761   0.0845   0.0850   0.0767   0.0556  0.0539  0.0547
##         sar3
##       0.3494
## s.e.  0.0548
##
## sigma^2 estimated as 31.82:  log likelihood = -1072.08,  aic = 2162.17
```

```
sarima710100
```

```
##
## Call:
## arima(x = employ_diff, order = c(7, 1, 0), seasonal = list(order = c(1, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6      ar7    sar1
##       -0.9619  -0.7643  -0.6687  -0.5271  -0.3263  -0.1717  -0.1013  0.9487
## s.e.   0.0555   0.0770   0.0865   0.0898   0.0866   0.0775   0.0557  0.0146
##
## sigma^2 estimated as 47.4:  log likelihood = -1132.82,  aic = 2281.64
```

```
sarima710200
```

```
##
## Call:
## arima(x = employ_diff, order = c(7, 1, 0), seasonal = list(order = c(2, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6      ar7    sar1
##       -0.9818  -0.7761  -0.6675  -0.5464  -0.3493  -0.1966  -0.1389  0.4602
## s.e.   0.0548   0.0768   0.0862   0.0892   0.0865   0.0774   0.0551  0.0486
##         sar2
##       0.5147
## s.e.  0.0491
##
## sigma^2 estimated as 35.35:  log likelihood = -1087.52,  aic = 2193.04
```

```
sarima710300
```

```
##
## Call:
## arima(x = employ_diff, order = c(7, 1, 0), seasonal = list(order = c(3, 0, 0),
##     period = 12))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6      ar7    sar1
##       -0.9735  -0.7874  -0.6347  -0.4884  -0.3189  -0.1885  -0.1255  0.2919
## s.e.   0.0549   0.0765   0.0868   0.0902   0.0870   0.0772   0.0553  0.0538
##         sar2    sar3
##       0.3473  0.3434
## s.e.  0.0544  0.0548
##
## sigma^2 estimated as 31.39:  log likelihood = -1069.54,  aic = 2159.07
```

Printing BIC values and selecting the lowest

```
BIC_values <- c(BIC(sarima110100), BIC(sarima110200), BIC(sarima110300),
                BIC(sarima210100), BIC(sarima210200), BIC(sarima210300),
                BIC(sarima310100), BIC(sarima310200), BIC(sarima310300),
                BIC(sarima410100), BIC(sarima410200), BIC(sarima410300),
                BIC(sarima510100), BIC(sarima510200), BIC(sarima510300),
                BIC(sarima610100), BIC(sarima610200), BIC(sarima610300),
                BIC(sarima710100), BIC(sarima710200), BIC(sarima710300))
BIC_values
```

```
##  [1] 2373.589 2292.866 2263.835 2347.073 2262.701 2226.043 2330.552 2249.145
##  [9] 2210.417 2316.323 2234.812 2200.478 2311.328 2228.950 2197.878 2315.416
## [17] 2233.609 2202.277 2317.936 2233.152 2202.993
```

```
min_BIC <- min(BIC_values)
min_BIC
```

```
## [1] 2197.878
```

```
min_BIC_index <- which.min(BIC_values)
min_BIC_index
```

```
## [1] 15
```

Choosing the lowest BIC model for residual analysis

```
print("From the above BIC values, SARIMA(5,1,0)x(3,0,0) has the lowest value")
```

```
## [1] "From the above BIC values, SARIMA(5,1,0)x(3,0,0) has the lowest value"
```

```
sarima_model <- arima(employ_diff, order = c(5, 1, 0), seasonal = list(order = c(3, 0, 0), pe
riod = 12))
residuals = residuals(sarima510300)
```

#Residual anaylisys using Shapiro and Ljung box

```
# Shapiro-Wilk test for normality
shapiro_test <- shapiro.test(residuals)
print("Shapiro-Wilk Test for Normality:")
```

```
## [1] "Shapiro-Wilk Test for Normality:"
```

```
print(shapiro_test)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals
## W = 0.97515, p-value = 1.56e-05
```

```
# Ljung-Box test for autocorrelation in residuals
ljung_box_test <- Box.test(residuals, lag = 10, type = "Ljung-Box")
print("Ljung-Box Test for Autocorrelation:")
```

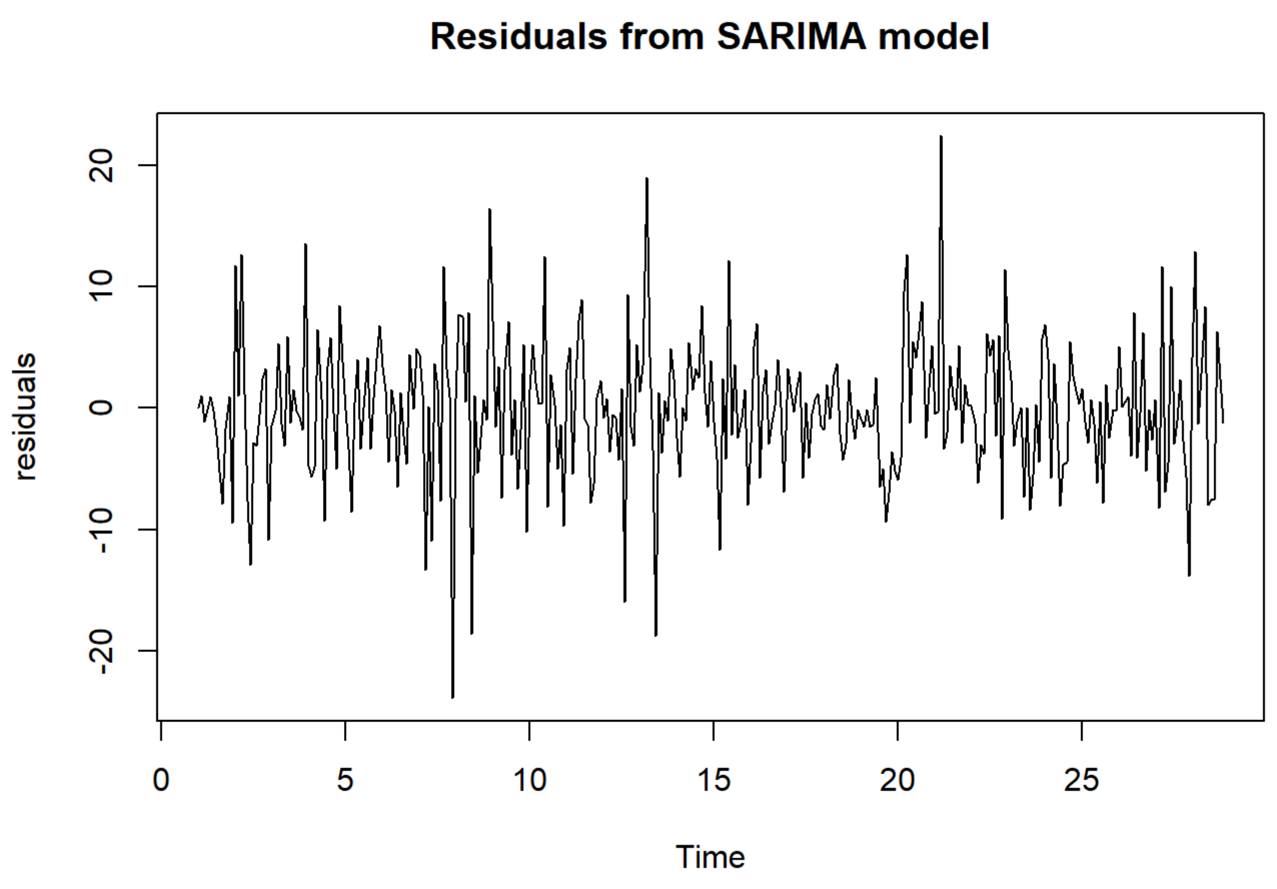
```
## [1] "Ljung-Box Test for Autocorrelation:"
```

```
print(ljung_box_test)
```

```
##
##  Box-Ljung test
##
## data:  residuals
## X-squared = 9.2147, df = 10, p-value = 0.5119
```

Predicting forecasts

```
plot(residuals, main="Residuals from SARIMA model")
```



```
predicted_values <- predict(sarima510300, n.ahead = 12)  # Predicting next 12 months
print(predicted_values)
```

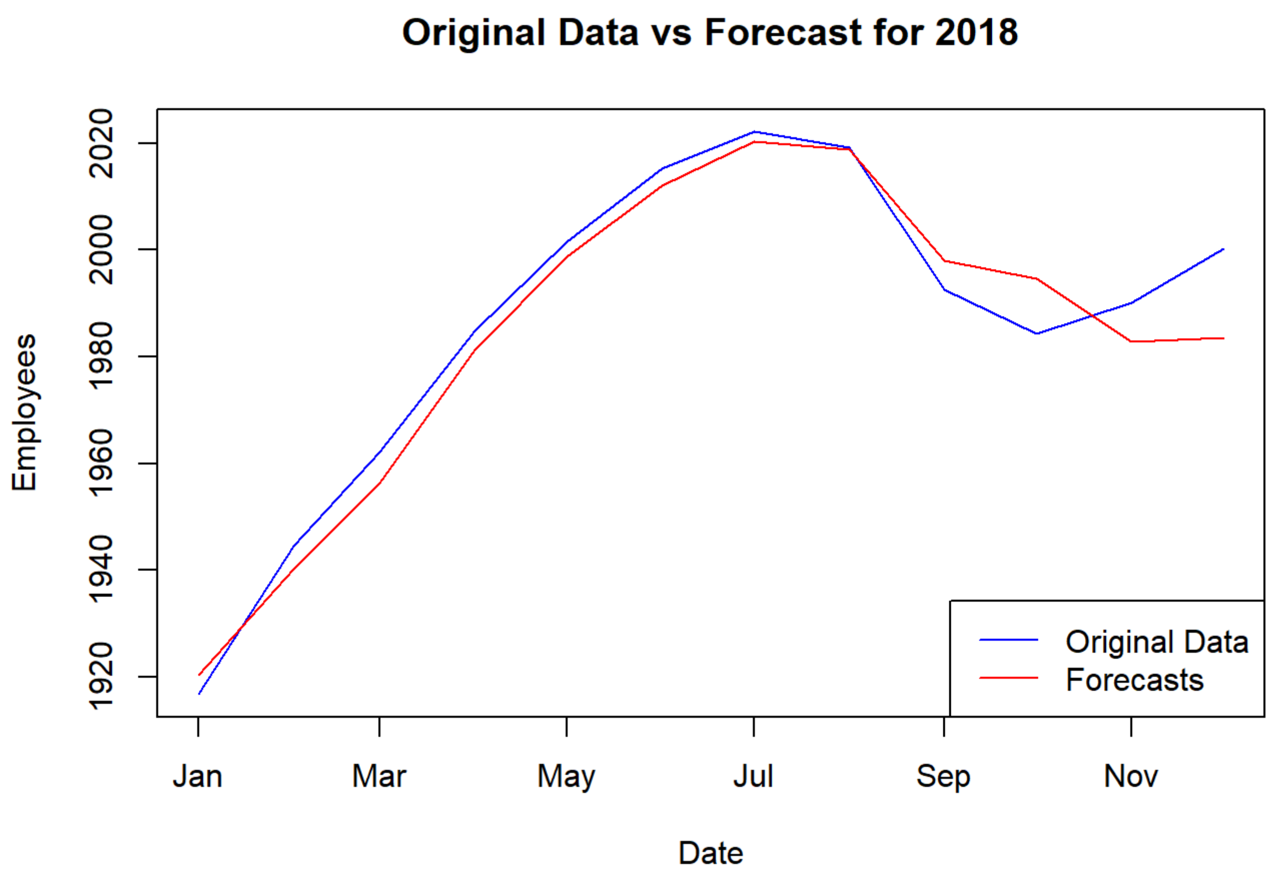
```
## $pred
##            Jan        Feb        Mar        Apr        May        Jun        Jul
## 28
## 29  19.820839  16.188842  24.968364  17.445099  13.335536   8.179975  -1.474682
##            Aug        Sep        Oct        Nov        Dec
## 28                                              -36.632007
## 29 -20.903030  -3.359250 -11.679140   0.734608
##
## $se
##          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 28
## 29 5.660166 5.794502 5.910440 6.072755 6.328143 6.574672 6.690373 6.850199
##          Sep      Oct      Nov      Dec
## 28                            5.654515
## 29 7.006101 7.172696 7.337543
```

```
original_forecasts <- diffinv(predicted_values$pred, lag = 1, differences = 1, xi = 1957)
original_forecasts <- original_forecasts[original_forecasts != 0]
original_forecasts <- original_forecasts[-1]
original_forecasts
```

```
##  [1] 1920.368 1940.189 1956.378 1981.346 1998.791 2012.127 2020.307 2018.832
##  [9] 1997.929 1994.570 1982.891 1983.625
```

Plotting forecasts for next 12 months

```
plot(test_data$Date, test_data$Employees, main = 'Original Data vs Forecast for 2018', type =
'l', col = 'blue', xlab = 'Date', ylab = 'Employees')

# Adding forecasted values to the plot (if 'original_forecasts' is aligned with 'test_data$Da
te')
lines(test_data$Date, original_forecasts, col = 'red', type = 'l')

# Adding legend
legend('bottomright', legend = c('Original Data', 'Forecasts'), col = c('blue', 'red'), lty =
1)
```



PART B- Non Seasonal Time Series

Motivation and Introduction of the Problem

The interplay between social media influence and stock market dynamics has become a significant area of interest for investors and researchers alike. In recent years, the role of platforms like Twitter in shaping public perception and potentially influencing stock prices has been underscored by several high-profile incidents. This project aims to analyze the relationship between Twitter activity and the stock prices of Twitter Inc. itself, to understand how announcements, public sentiment, and user engagement on the platform may correlate with or even predict movements in the company's stock price.

Describe the Dataset in Detail

The dataset, "twitter-stocks.csv," comprises stock price data for Twitter Inc. from its initial public offering in November 2013 up to the present. The dataset includes the following fields:

Date: Indicates the date for each stock entry. Open: The opening stock price for each day. High: The highest price reached during the trading day. Low: The lowest price during the trading day. Close: The closing price at the end of the trading day. Adj Close: Adjusted closing price accounting for any corporate actions. Volume: The total volume of stock traded during the day. This data offers a comprehensive view of the stock's performance over time, providing insights into daily fluctuations, long-term trends, and trading volume variations.

Best model and summary

First I implemented ARIMA models just to verify that ARIMA models are not a great option for a financial time series. I implemented all combinations of ARIMA models and plotted their residual analysis. Forecasting predicted a flat line because it didn't capture any dependencies. Then I applied some possible combinations of GARCH models. They were able to capture the dependencies and predicted forecasts which were also not very promising. Then finally I applied an extended version of GARCH model called GJR-GARCH. It is same as the garch but with an additional parameter which captures volatility of the data as well which crucial in financial time series. GJR-GARCH(1,1) X ARIFMA(2,0,2) was the model achieved after some manual combinations. It had the highest log likehood, smaller p-values. higher t-values with a standard distribution. Standard errors were also on the lower end. Overall it was great to apply ARIMA,ARIFMA,GARCH, GJR_GARCH models and learn all the optimal parameters and residual analysis.

```
library(stats)
library(tseries)
library(tidyverse)
library(TSA)
library(ggplot2)
library(rugarch)
```

```
## Warning: package 'rugarch' was built under R version 4.2.3
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'
```

```
## The following object is masked from 'package:purrr':
##
##     reduce
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

```
library(forecast)
library(dplyr)

df <- read.csv("C:/Users/vatsal/Downloads/twitter-stocks.csv")
df <- select(df, Date, Close)

df$Date <- as.Date(df$Date, "%Y-%m-%d")
num_rows <- nrow(df)
new_dates <- seq(from = as.Date("2013-11-01"), by = "days", length.out = num_rows)
df$Date <- new_dates
start_date <- as.Date("2013-11-01")
end_date <- as.Date("2019-12-31")
filtered_data <- subset(df, Date >= start_date & Date <= end_date)
head(filtered_data)
```

	Date <date>	Close <dbl>
1	2013-11-01	44.90
2	2013-11-02	41.65
3	2013-11-03	42.90
4	2013-11-04	41.90
5	2013-11-05	42.60
6	2013-11-06	44.69

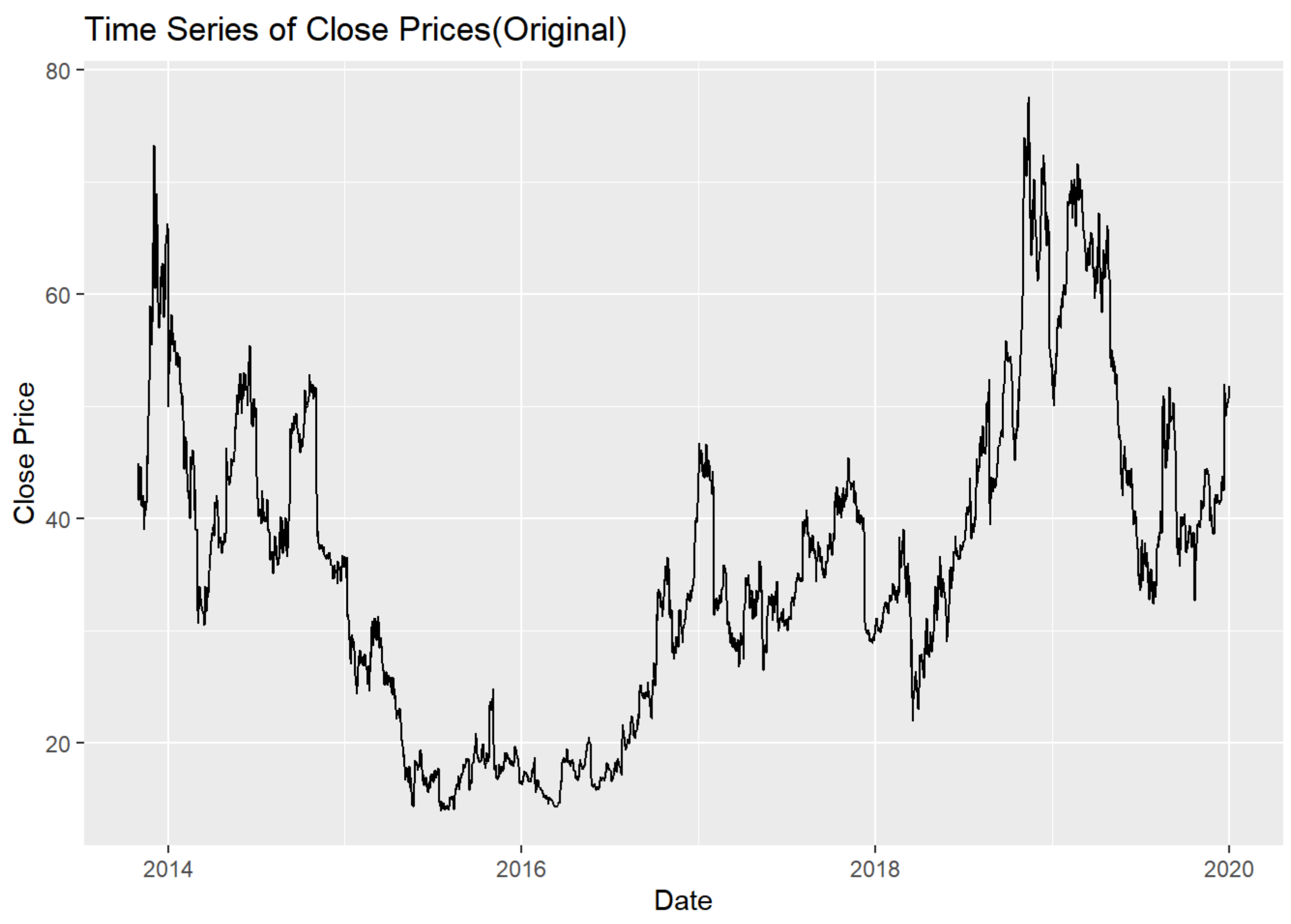
6 rows

```
tail(filtered_data)
```

	Date <date>	Close <dbl>
2247	2019-12-26	50.07
2248	2019-12-27	49.94
2249	2019-12-28	50.34
2250	2019-12-29	50.45
2251	2019-12-30	50.74
2252	2019-12-31	51.78

6 rows

```
ggplot(filtered_data, aes(x = Date, y = Close)) +
  geom_line() +
  labs(title = "Time Series of Close Prices(Original)",
       x = "Date",
       y = "Close Price")
```

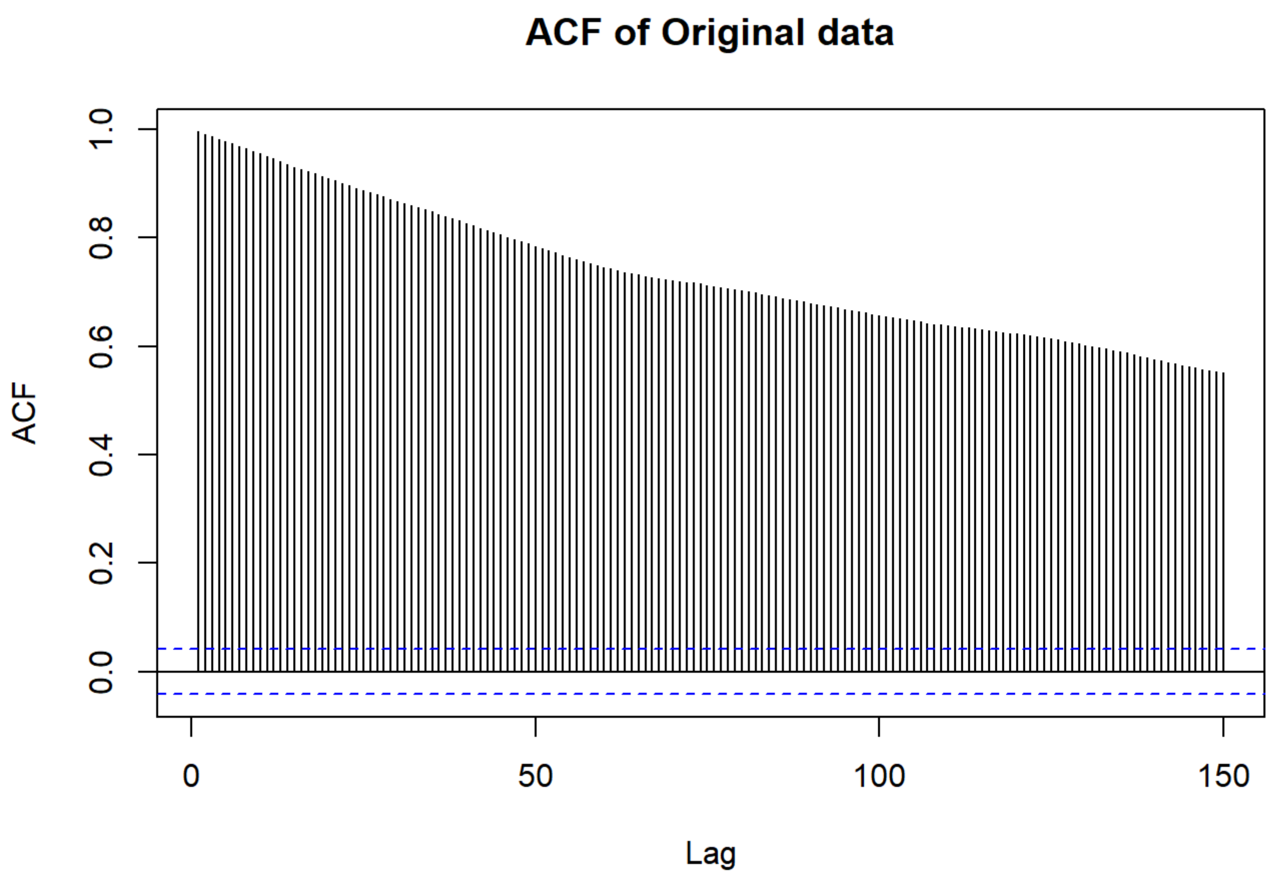


```
train_data <- subset(df, Date >= start_date & Date <= as.Date("2019-10-31"))
test_data <- subset(df, Date > as.Date("2019-10-31") & Date <= end_date)
tail(train_data)
```

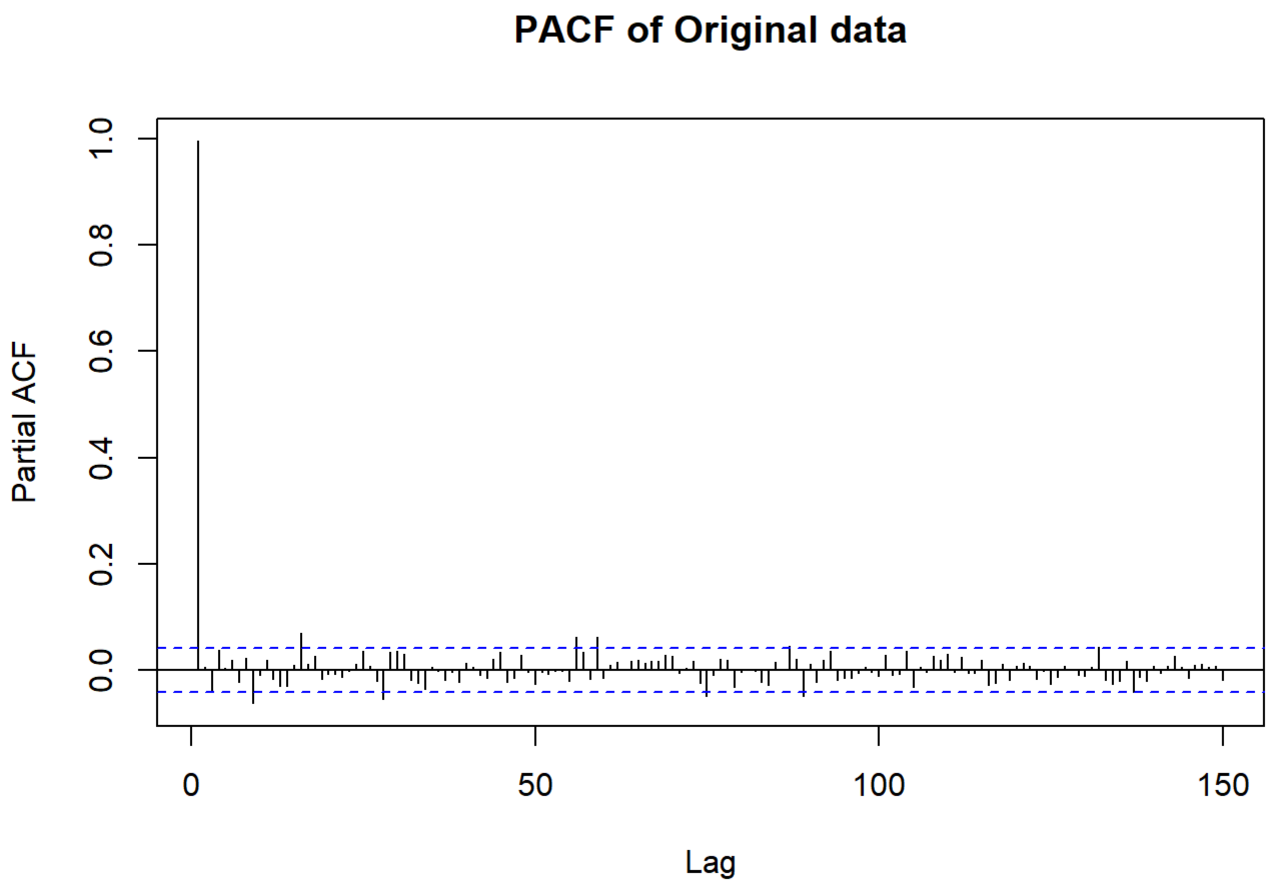
	Date <date>	Close <dbl>
2186	2019-10-26	37.74
2187	2019-10-27	38.41
2188	2019-10-28	39.49
2189	2019-10-29	39.60
2190	2019-10-30	39.52
2191	2019-10-31	39.84

6 rows

```
data1 <- train_data$Close

acf(data1, lag.max = 150, main='ACF of Original data')
```



```r
#exponentially dying

pacf(data1, lag.max = 150, main='PACF of Original data')
```



```
#Seems like AR(1)
eacf(data1)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x x x x  x  x  x
## 1 o o o o o o o x o o o  o  o  o
## 2 x o o o o o o x o o o  o  o  o
## 3 x x o o o o o x o o o  o  o  o
## 4 x x x o o o o x o o o  o  o  o
## 5 x x x x o o o x o o o  o  o  o
## 6 x x x o x o o x o o o  o  o  o
## 7 x x x o x x o x o o o  o  o  o
```

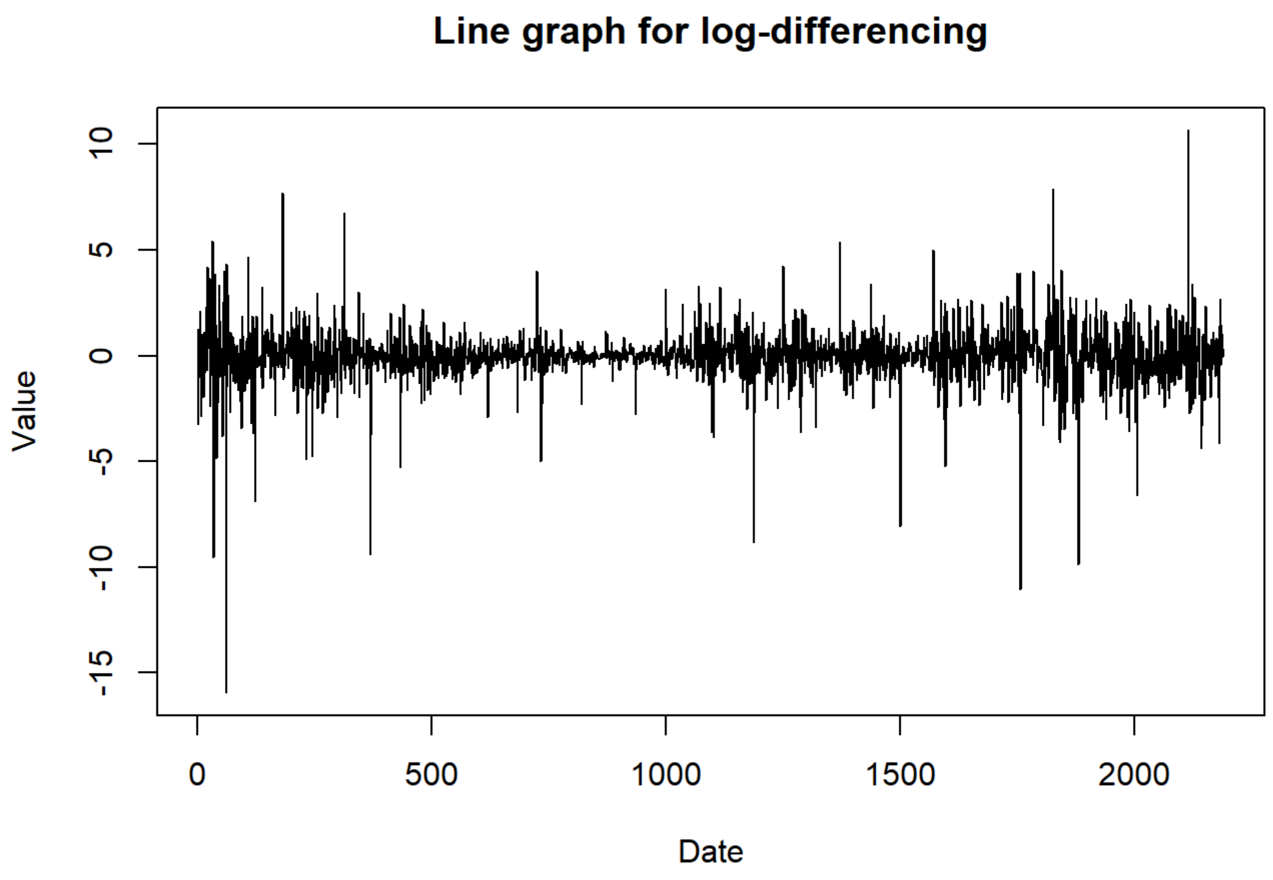
```
#Checking for stationarity
adf_test = adf.test(data1)
print(adf_test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data1
## Dickey-Fuller = -2.524, Lag order = 12, p-value = 0.3565
## alternative hypothesis: stationary
```

```
print("Since p-value=0.3565 > 0.05, the TS is non-stationary")
```

```
## [1] "Since p-value=0.3565 > 0.05, the TS is non-stationary"
```

```
#Since p-value=0.3215 > 0.05, the TS is non-stationary

close_diff <- diff(data1)
plot(close_diff, type = 'l', xlab = 'Date', ylab = 'Value', main = 'Line graph for log-differ
encing')
```



```
adf_test = adf.test(close_diff)
```

```
## Warning in adf.test(close_diff): p-value smaller than printed p-value
```

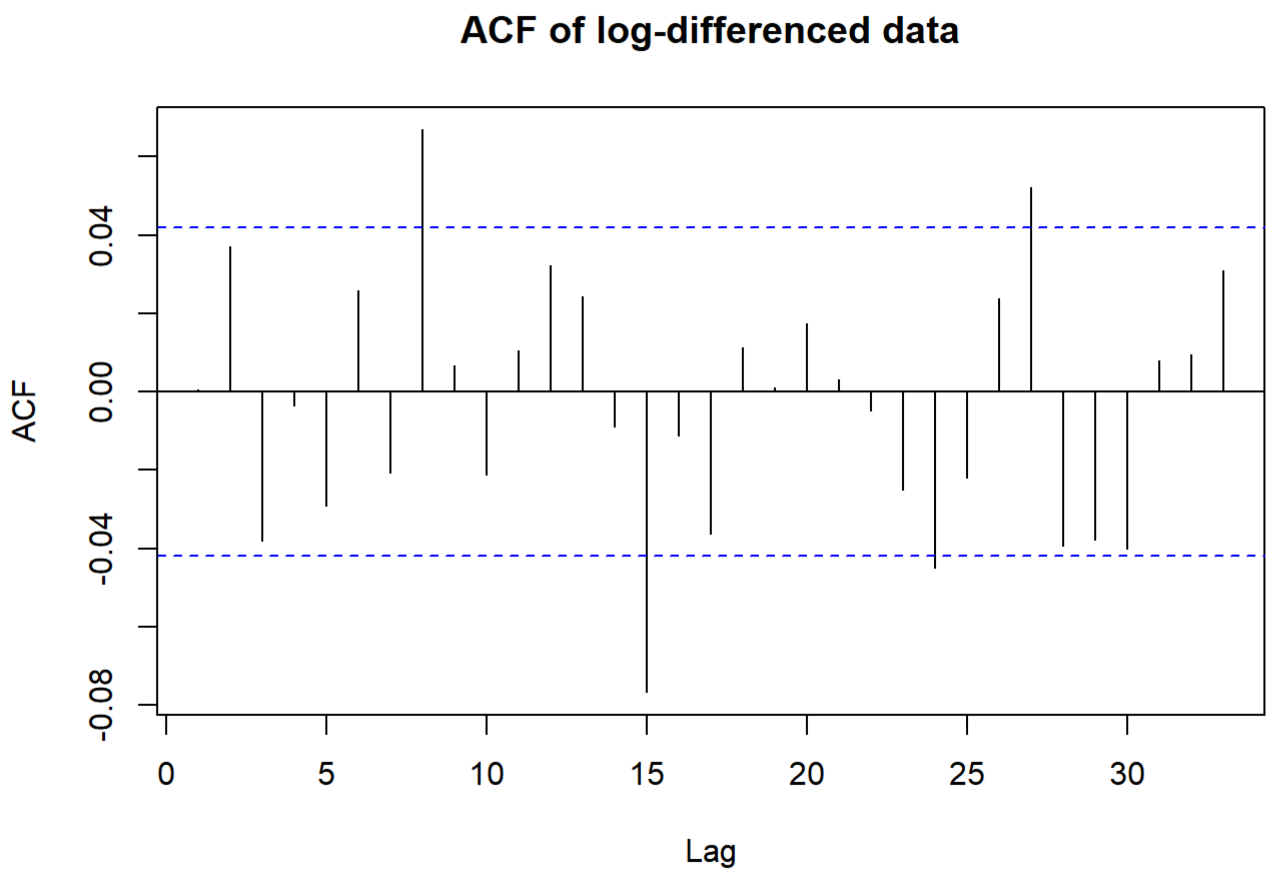
```
print(adf_test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  close_diff
## Dickey-Fuller = -11.878, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

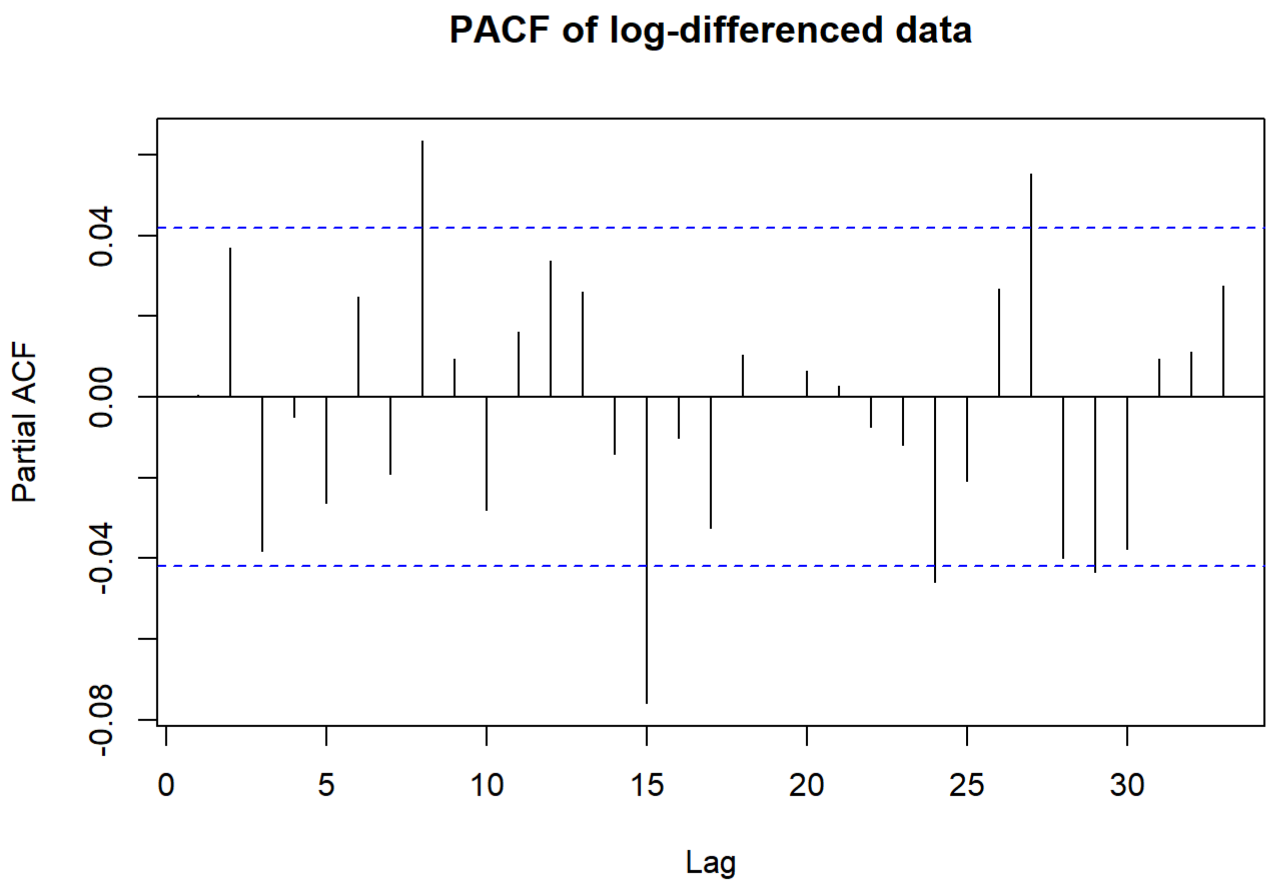
```
print("Since p-value=0.01 < 0.05, the TS is stationary")
```

```
## [1] "Since p-value=0.01 < 0.05, the TS is stationary"
```

```
# Since p-value=0.01 < 0.05, the TS is stationary

acf(close_diff, main='ACF of log-differenced data')
```



```
pacf(close_diff, main='PACF of log-differenced data')
```

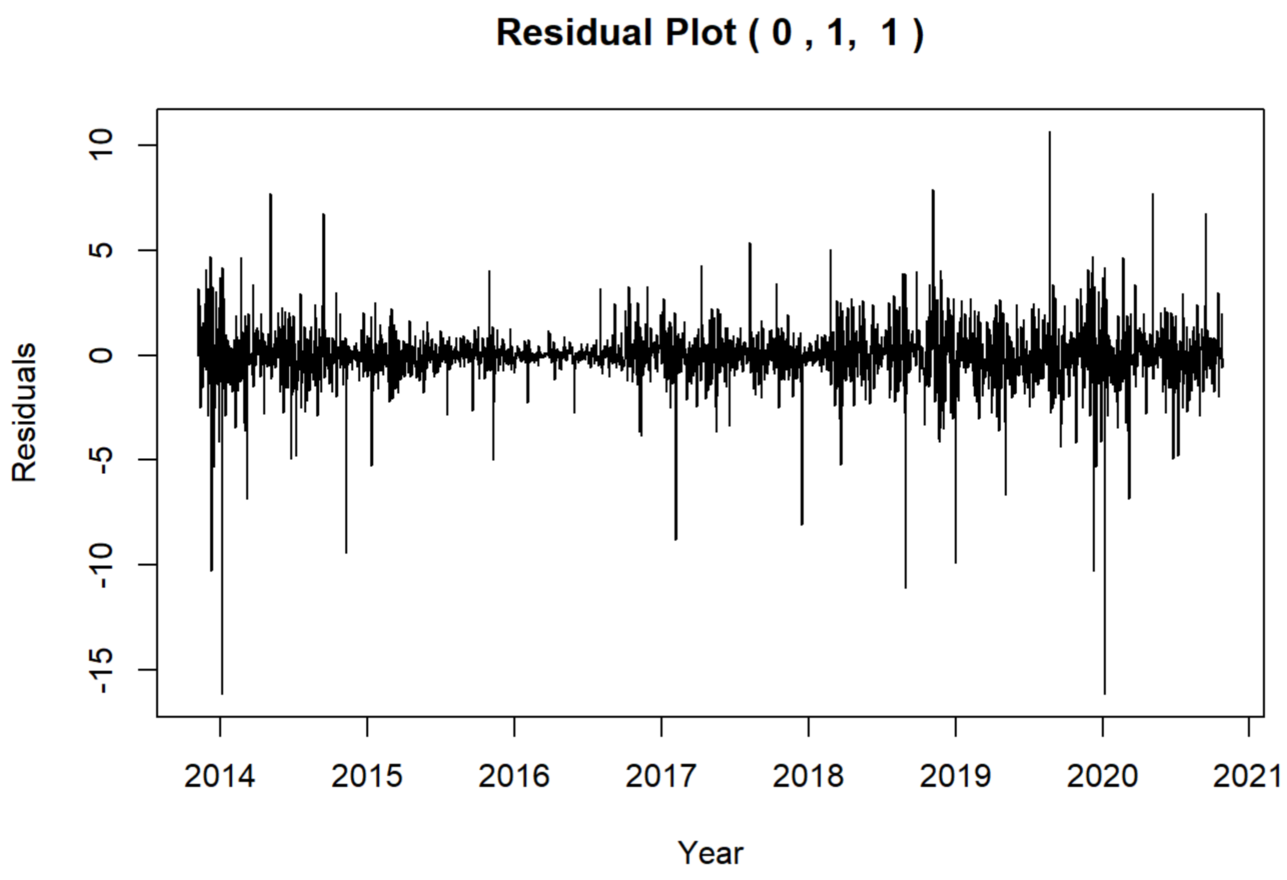


```
eacf(close_diff)
```

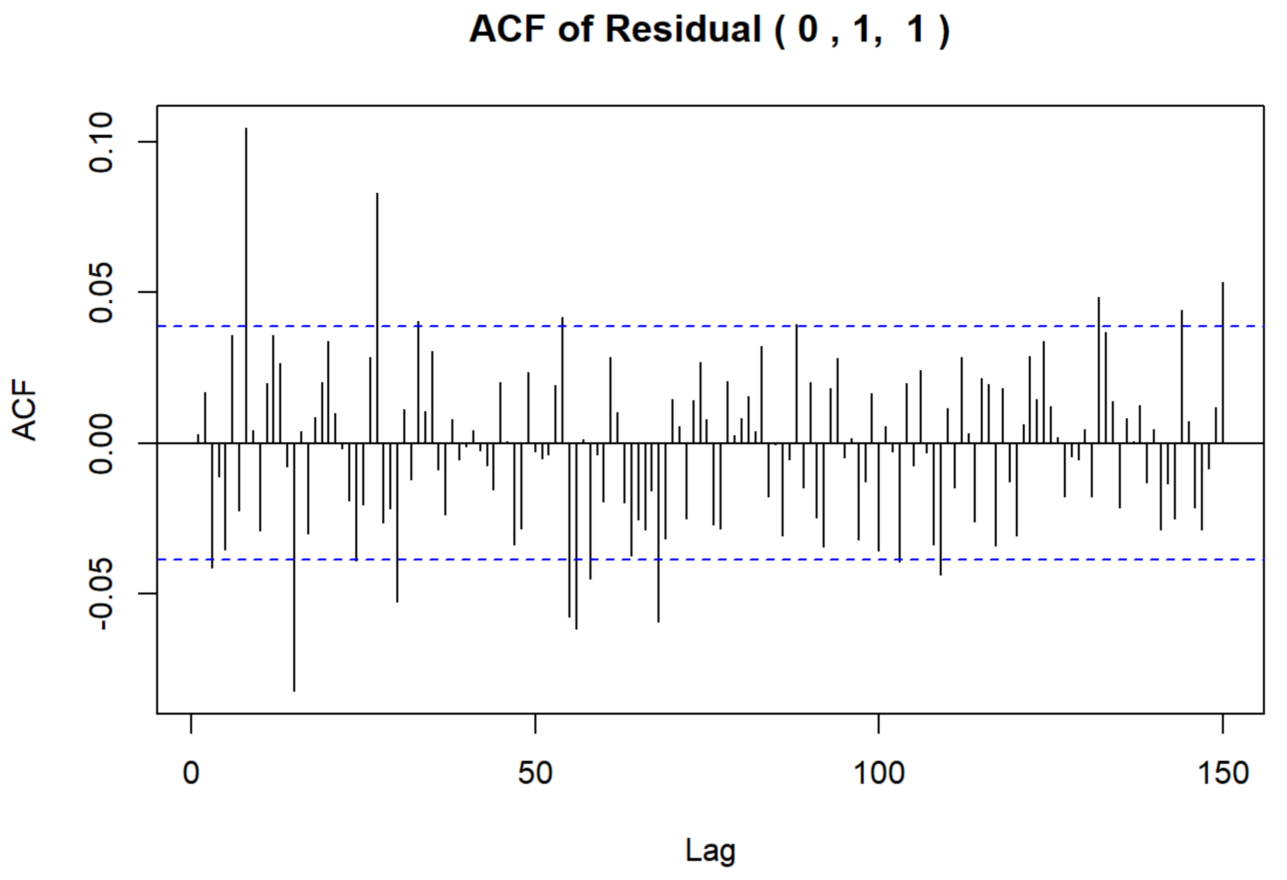
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o o o o o o o x o o o  o  o  o
## 1 o o o o o o o x o o o  o  o  o
## 2 x x o o o o o x o o o  o  o  o
## 3 x x x o o o o x o o o  o  o  o
## 4 x x x x o o o x o o o  o  o  o
## 5 x o x o x o o x o o o  o  o  o
## 6 x o x o x x o x o o o  o  o  o
## 7 x x x x x x x x o o o  o  o  o
```

```r
plot_arima_resid <- function(model) {
  resid <- resid(model)
  resid_ts <- ts(resid, start=c(2013, 311), end=c(2020,300), frequency=365)
  p <- model$arma[1]
  q <- model$arma[2]
  plot(resid_ts, xlab = "Year", ylab = 'Residuals', main=paste('Residual Plot',"(", p, ", 1,
", q, ")"))
  acf(resid_ts[1:length(resid_ts)],lag.max=150, main=paste('ACF of Residual',"(", p, ", 1, ",
q, ")"))
  pacf(resid_ts[1:length(resid_ts)],lag.max=150, main=paste('PACF of Residual',"(", p, ", 1,
", q, ")"))
}

my_df <- data.frame(model = character(),
                    AIC = numeric(),
                    BIC = numeric(),
                    Shapiro = round(numeric(),3),
                    Ljung = round(numeric(),3))
new_row <- data.frame(model='',AIC=0,BIC=0,Shapiro=0,Ljung=0)

plot_arima_residuals <- function(arima_model, my_df) {
  resid <- residuals(arima_model)
  resid_ts <- ts(resid, start=c(2013, 311), end=c(2020,300), frequency=365)
  p <- arima_model$arma[1]
  q <- arima_model$arma[2]
  order <- paste("(", p, ", 1, ", q, ")", sep = "")
  plot_arima_resid(arima_model)
  qqnorm(resid_ts, main=paste('Residuals plot',order))
  qqline(resid_ts)

  shap=shapiro.test(resid_ts)

  ljung=Box.test(resid, lag = 20, type = "Ljung-Box")

  cat("AIC:", AIC(arima_model),"\n")
  cat("BIC:", BIC(arima_model),"\n")
  model_name <- paste("ARIMA", paste(order, collapse = ','), sep = " ")
  if (!model_name %in% my_df$model) {
    new_row <- data.frame(model = model_name,
                          AIC = AIC(arima_model),
                          BIC = BIC(arima_model),
                          Shapiro = round(shap$p.value, 3),
                          Ljung = round(ljung$p.value, 3))

    my_df <- rbind(my_df, new_row)
  }
}


arima011 <- arima(x=close_diff, order=c(0,1,1))
arima110 <- arima(x=close_diff, order=c(1,1,0))
arima111 <- arima(x=close_diff, order=c(1,1,1))
arima210 <- arima(x=close_diff, order=c(2,1,0))
arima211 <- arima(x=close_diff, order=c(2,1,1))
arima012 <- arima(x=close_diff, order=c(0,1,2))
```

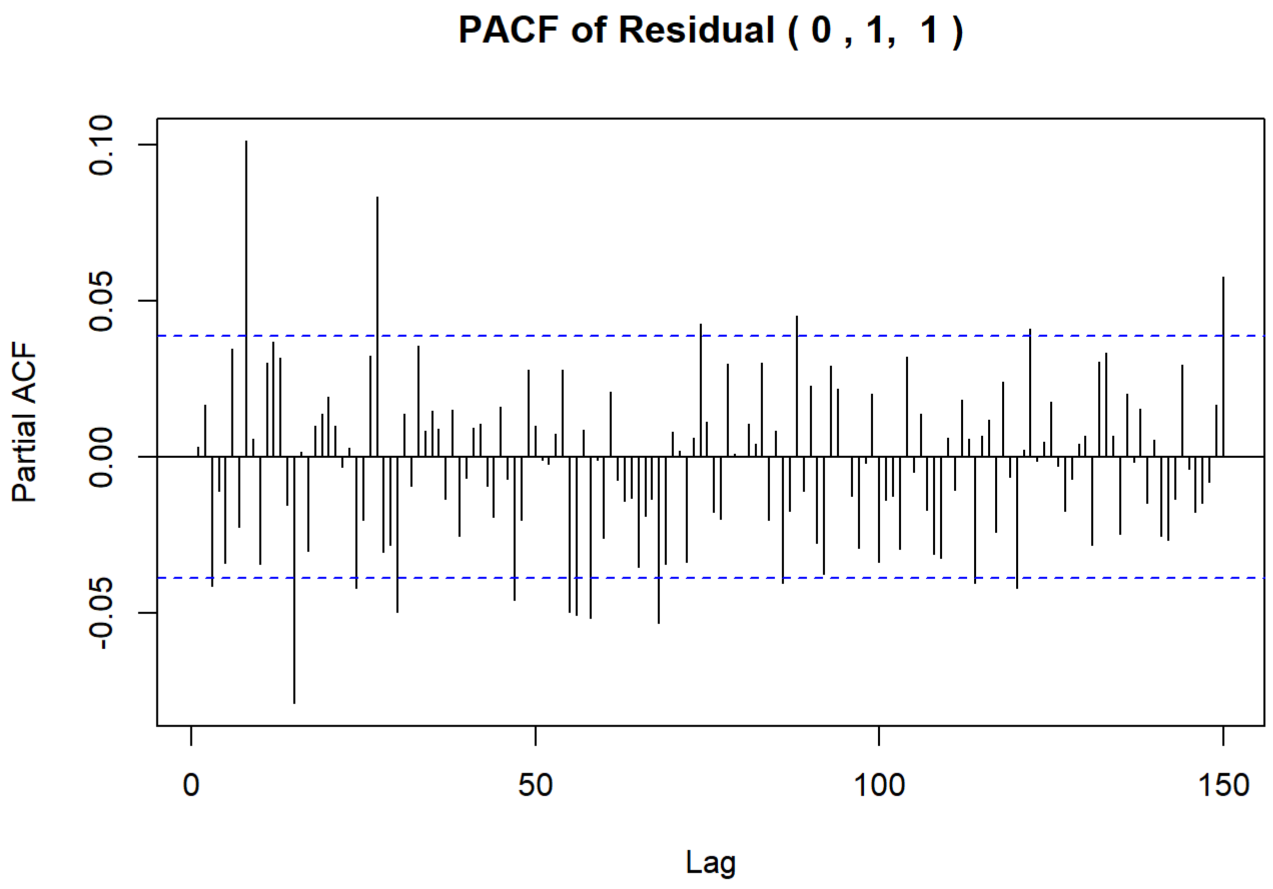
```
arima112 <- arima(x=close_diff, order=c(1,1,2))
arima212 <- arima(x=close_diff, order=c(2,1,2))
arima310 <- arima(x=close_diff, order=c(3,1,0))
arima311 <- arima(x=close_diff, order=c(3,1,1))
arima312 <- arima(x=close_diff, order=c(3,1,2))
arima013 <- arima(x=close_diff, order=c(0,1,3))
arima113 <- arima(x=close_diff, order=c(1,1,3))
arima213 <- arima(x=close_diff, order=c(2,1,3))
arima313 <- arima(x=close_diff, order=c(3,1,3))



my_df <- plot_arima_residuals(arima011, my_df)
```

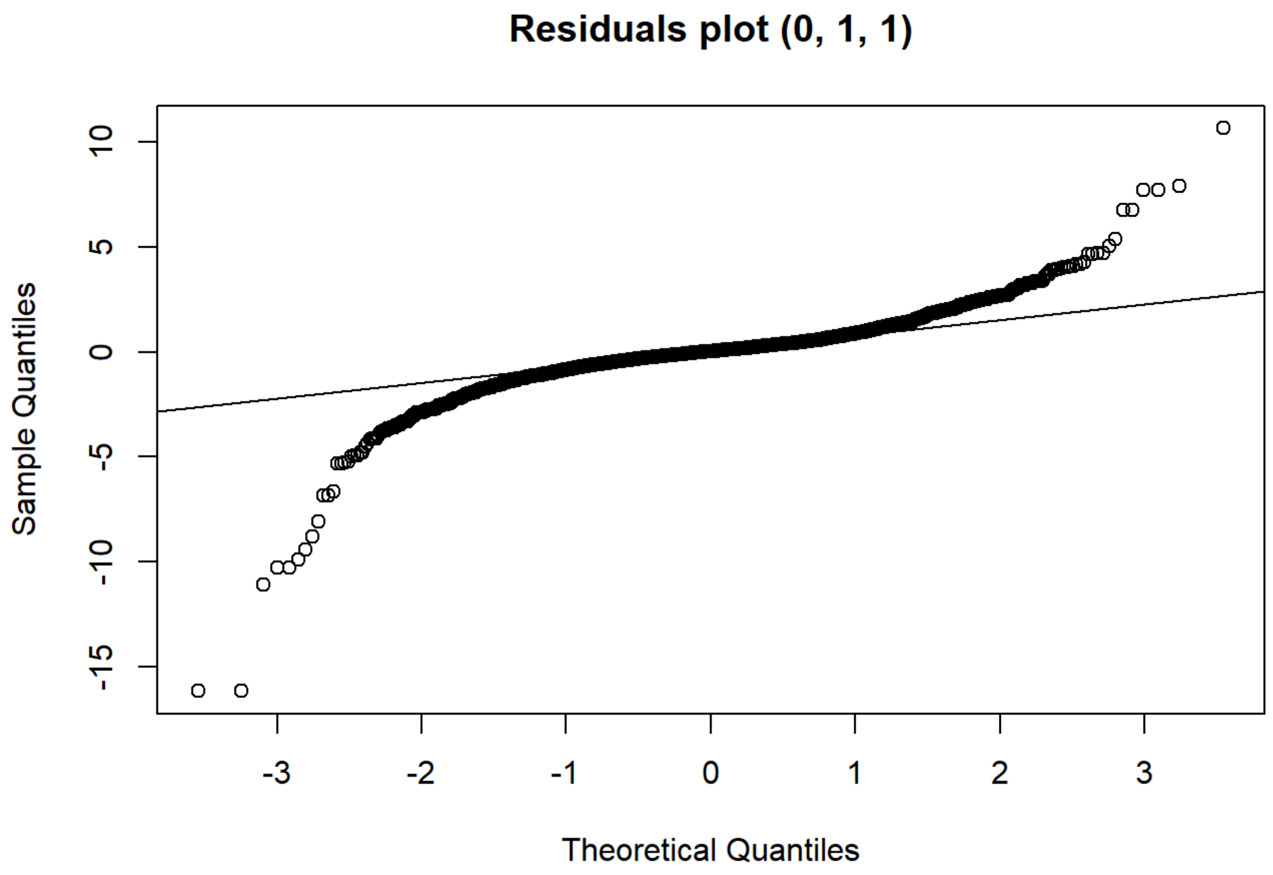


Residual Plot (0 , 1, 1)
Residuals
Year
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021



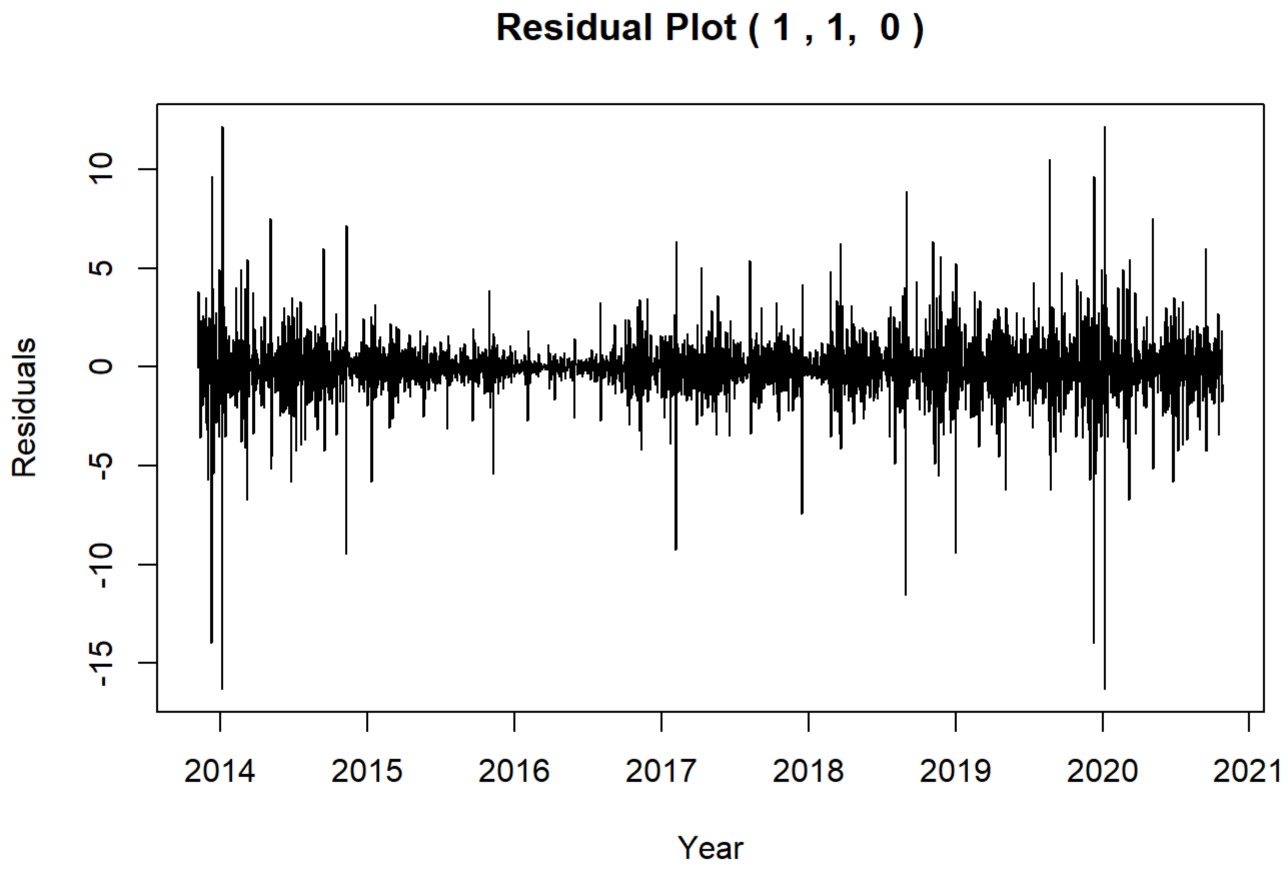
ACF of Residual (0 , 1, 1)
ACF
Lag
0.10
0.05
0.00
-0.05
0
50
100
150

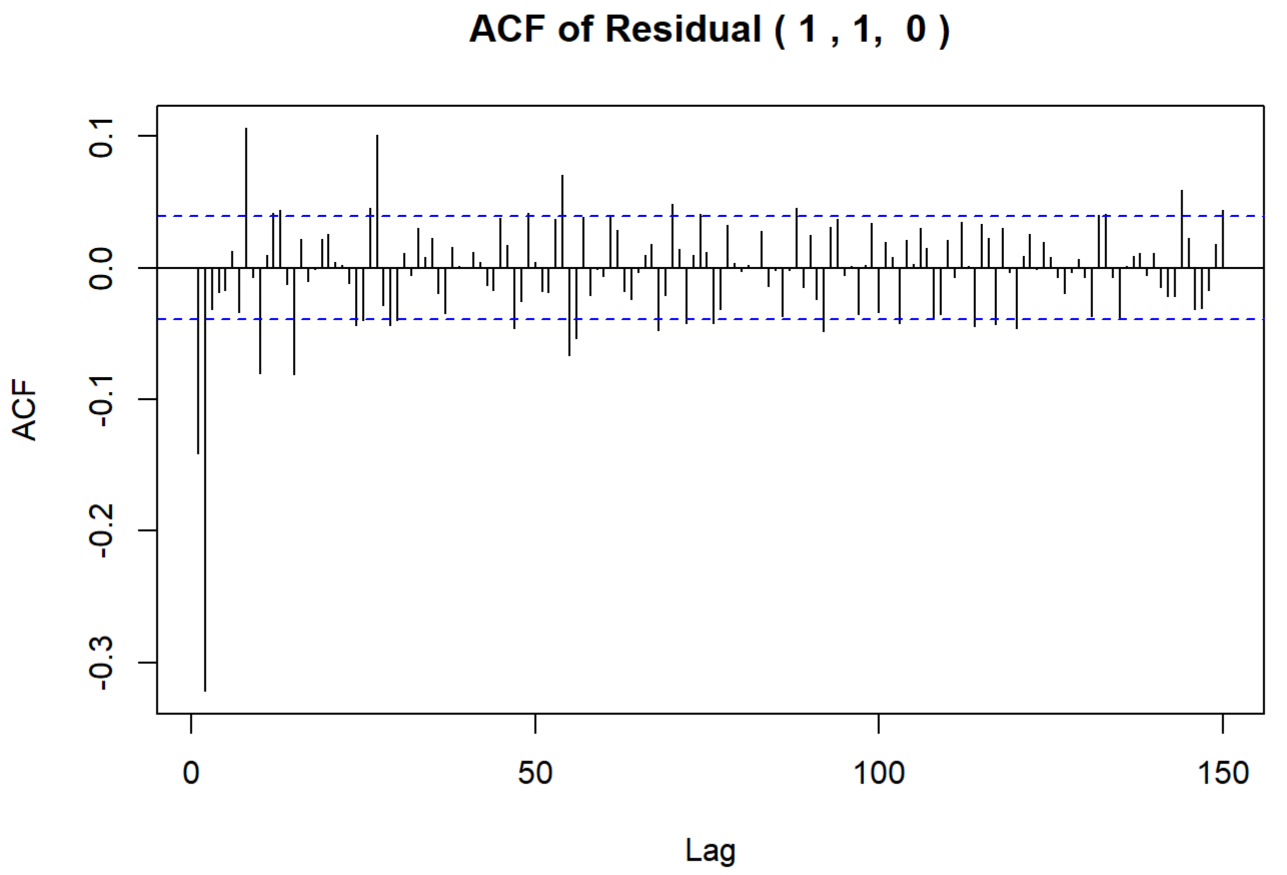


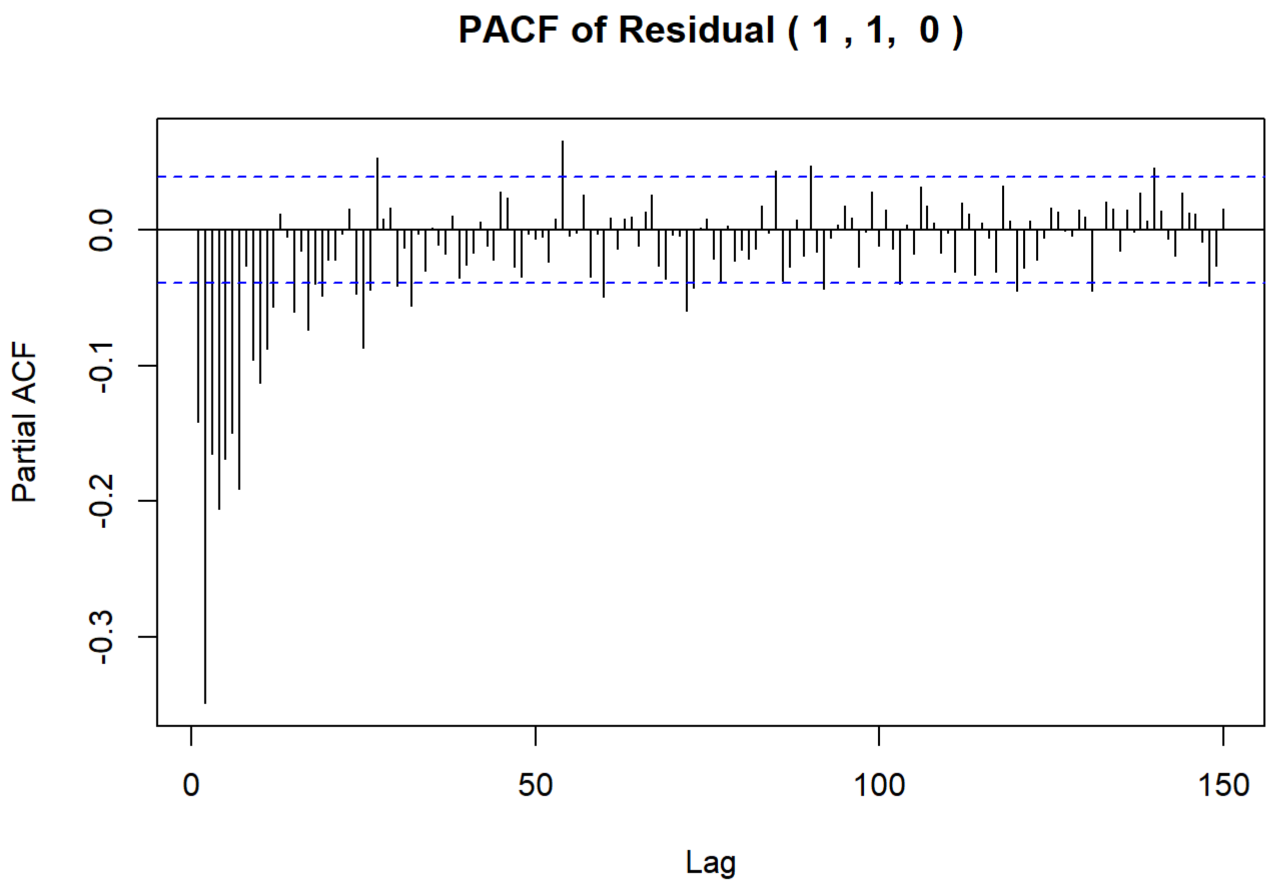


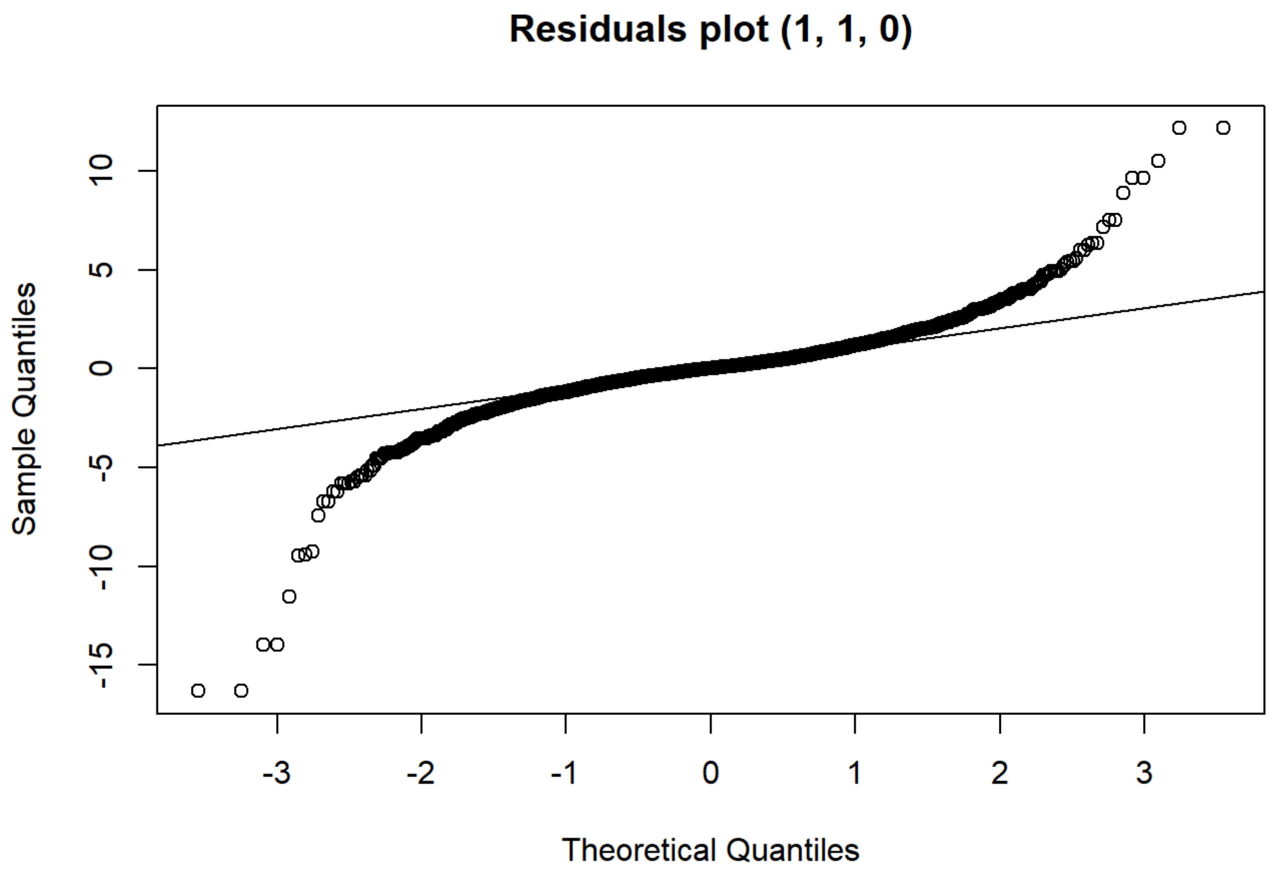
```
## AIC: 7475.113
## BIC: 7486.496
```

```
my_df <- plot_arima_residuals(arima110, my_df)
```


Residual Plot (1 , 1, 0)
Residuals
Year
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021

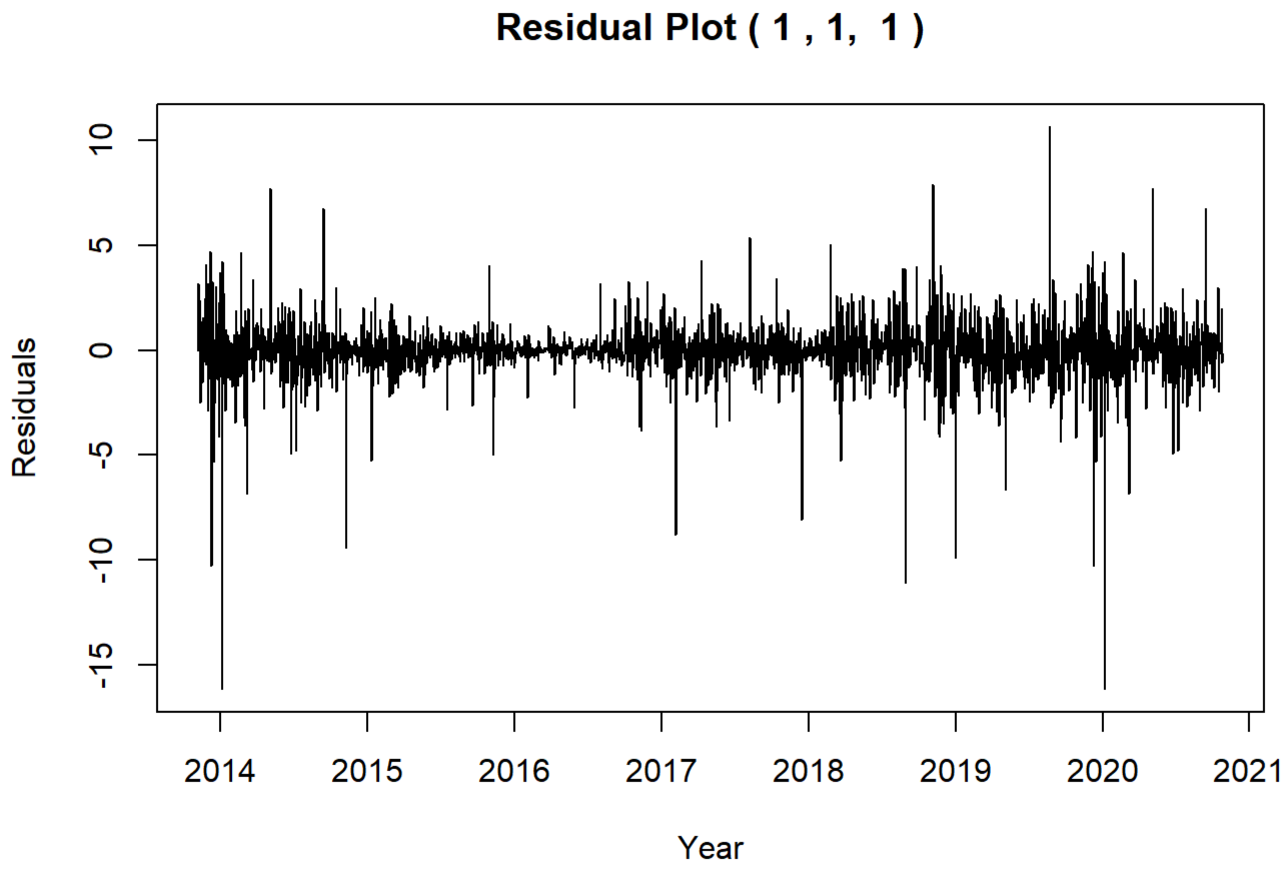

ACF of Residual (1 , 1, 0)
ACF
Lag
0.1
0.0
-0.1
-0.2
-0.3
0
50
100
150

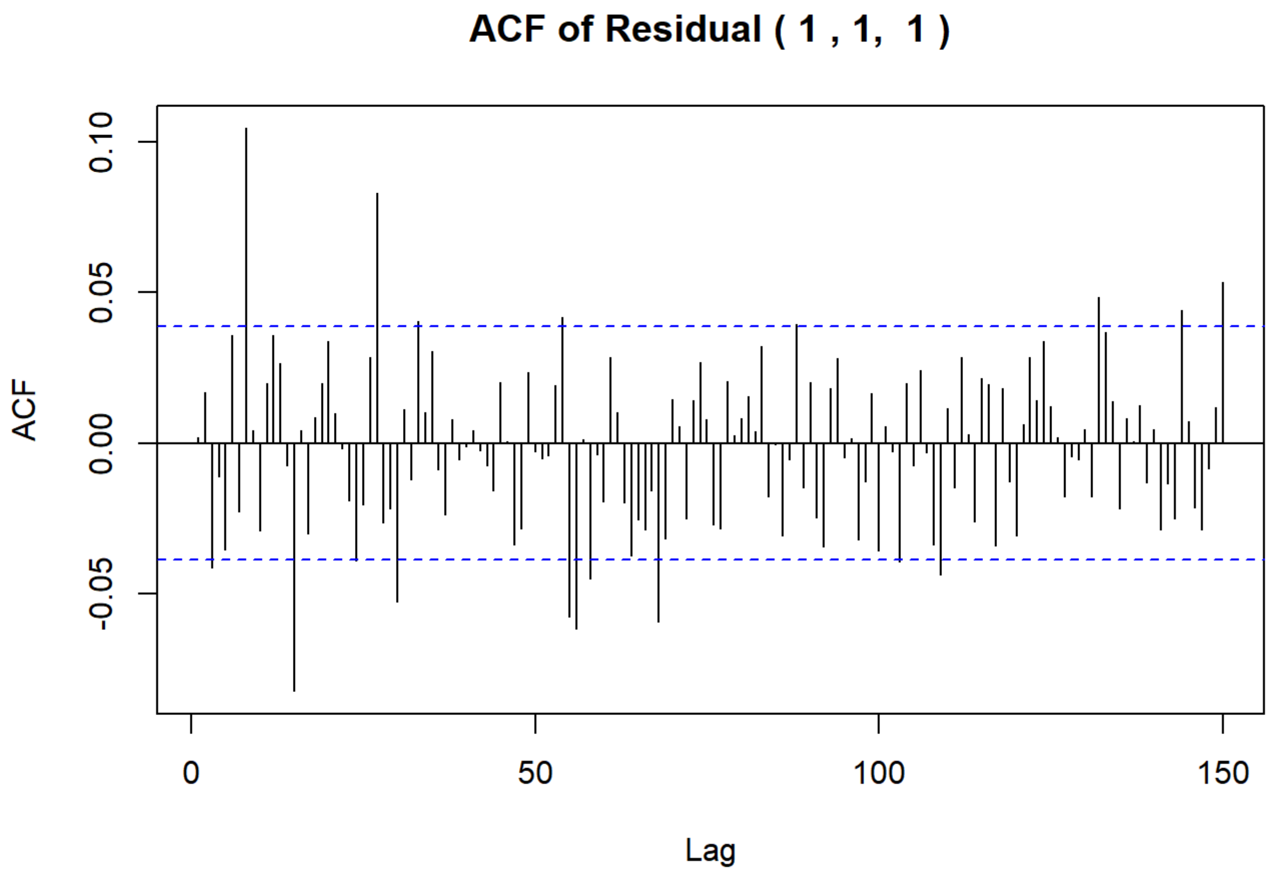


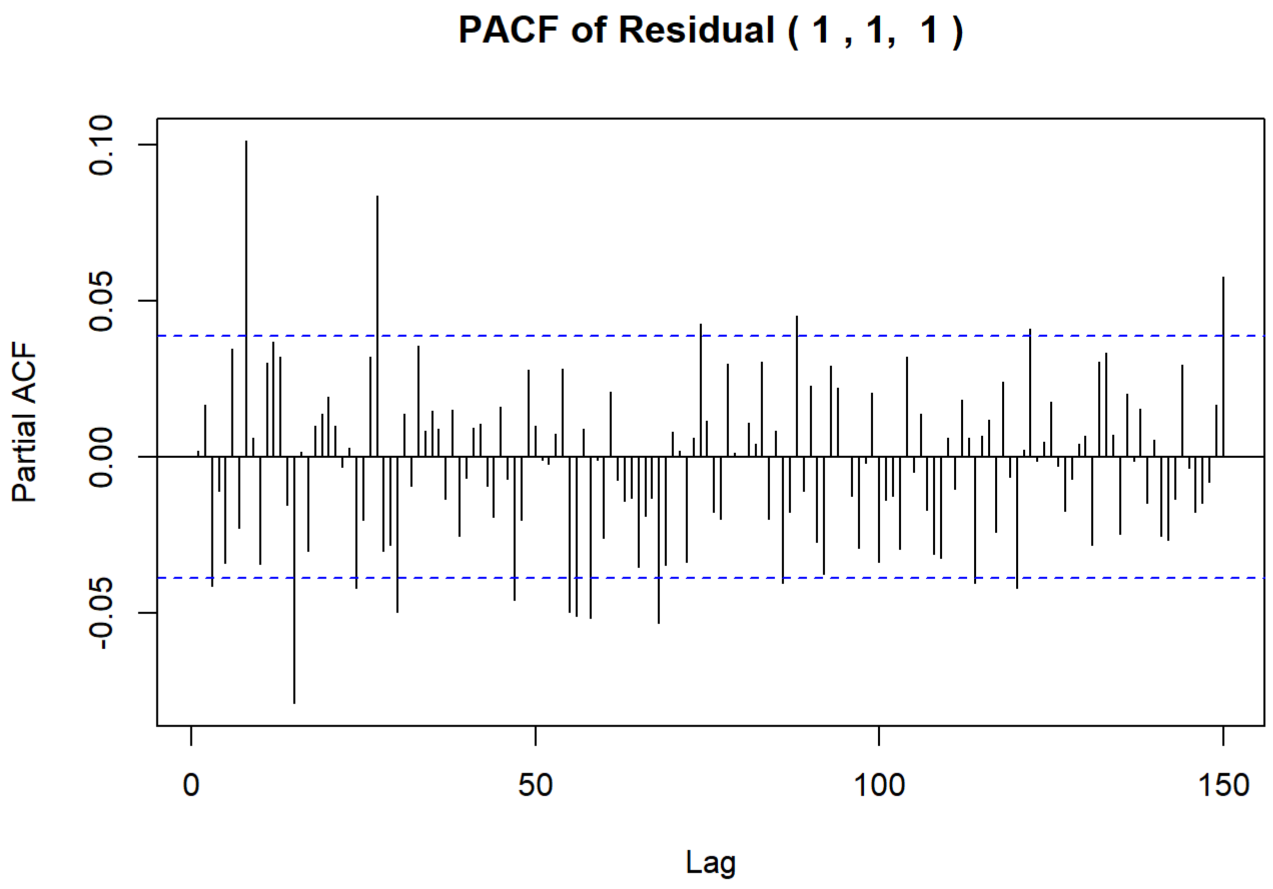


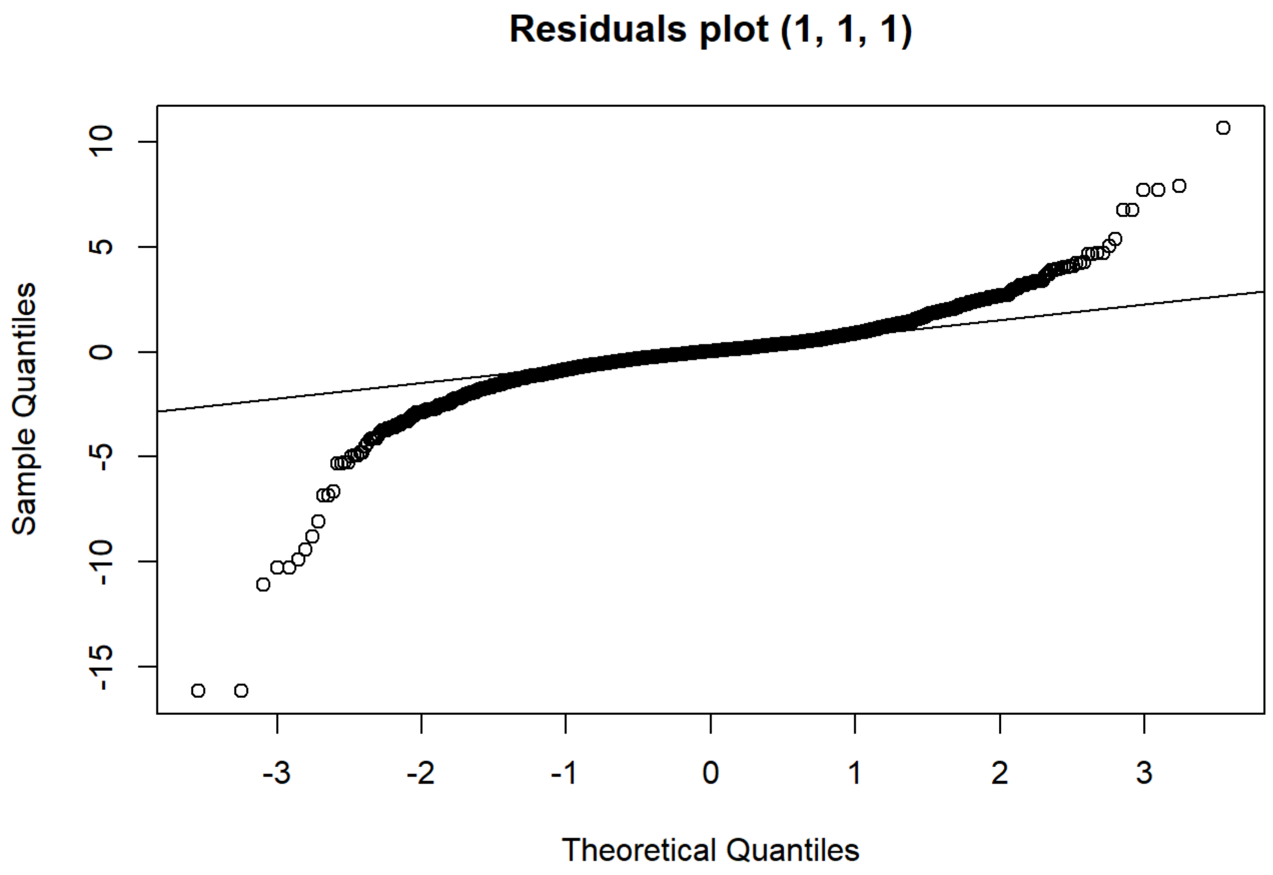
```
## AIC: 8297.696
## BIC: 8309.079
```

```
my_df <- plot_arima_residuals(arima111, my_df)
```


Residual Plot (1 , 1, 1)
Residuals
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021
Year

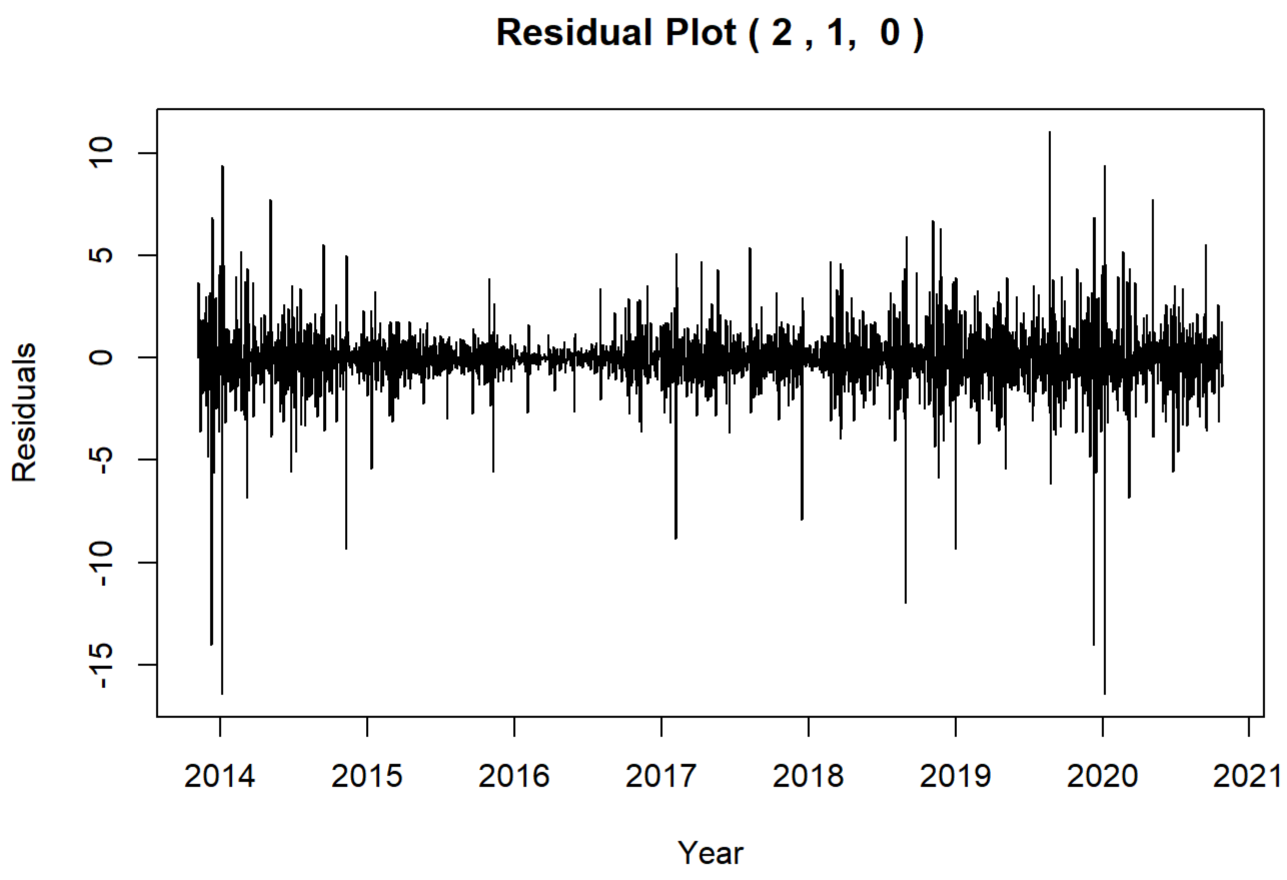

ACF of Residual (1 , 1, 1)
ACF
0.10
0.05
0.00
-0.05
0
50
100
150
Lag



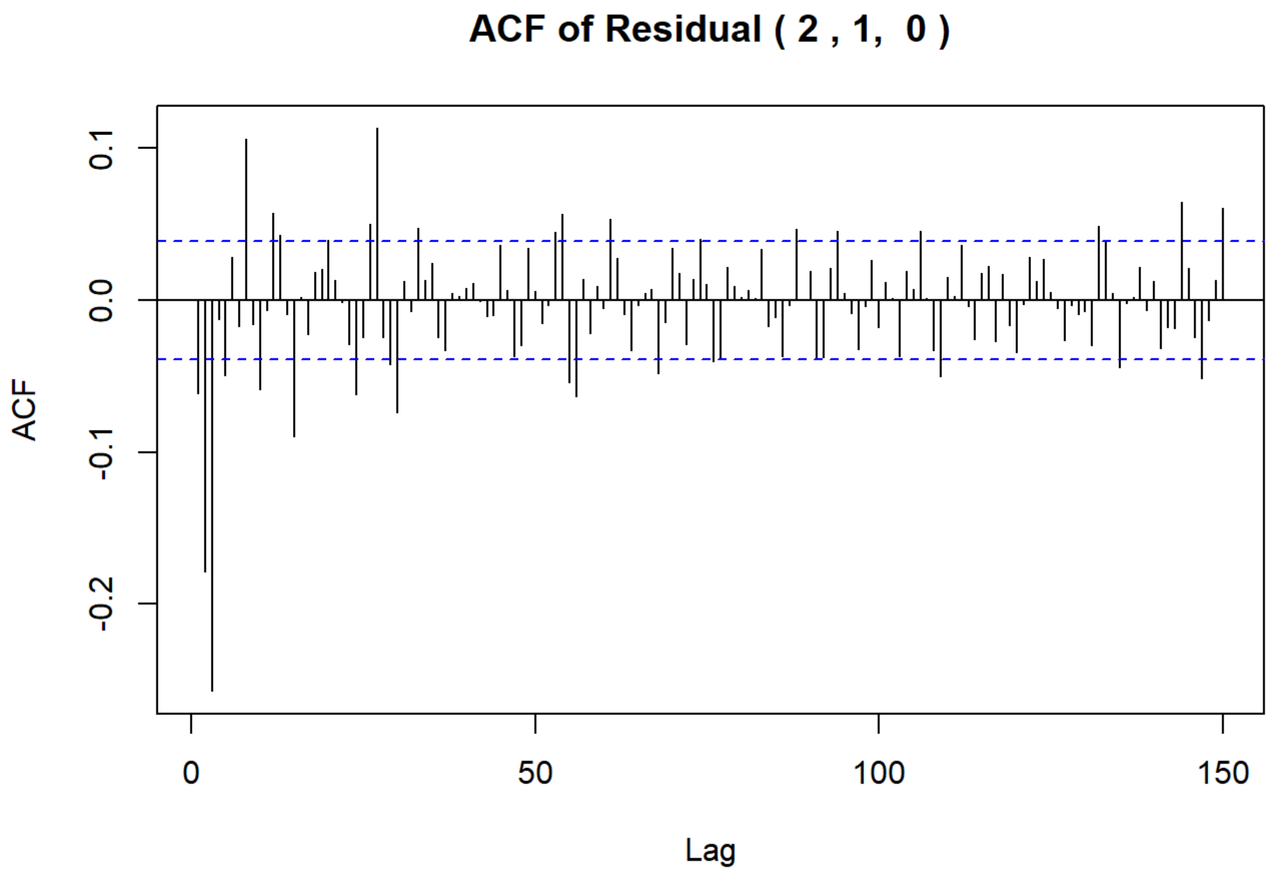


```
## AIC: 7477.111
## BIC: 7494.184
```

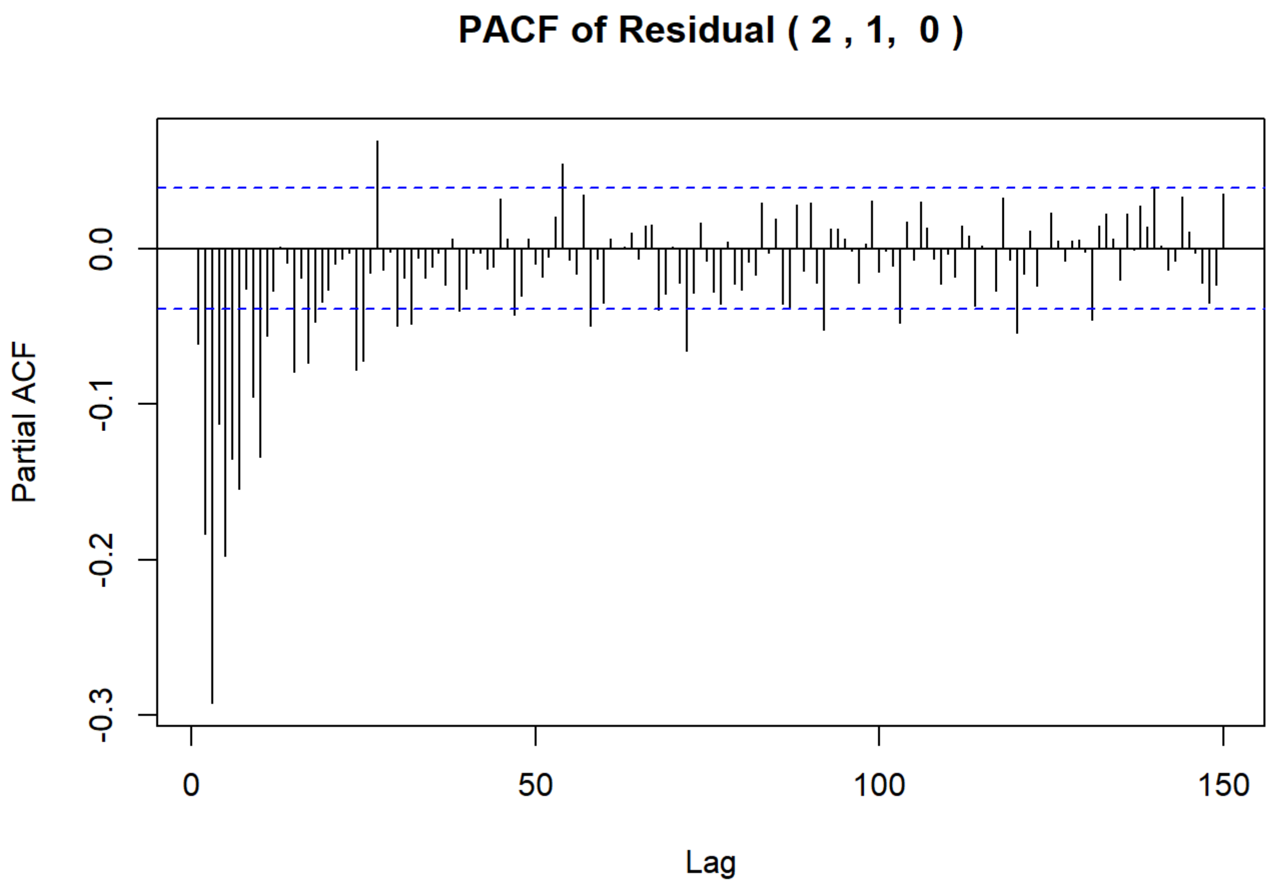
```
my_df <- plot_arima_residuals(arima210, my_df)
```

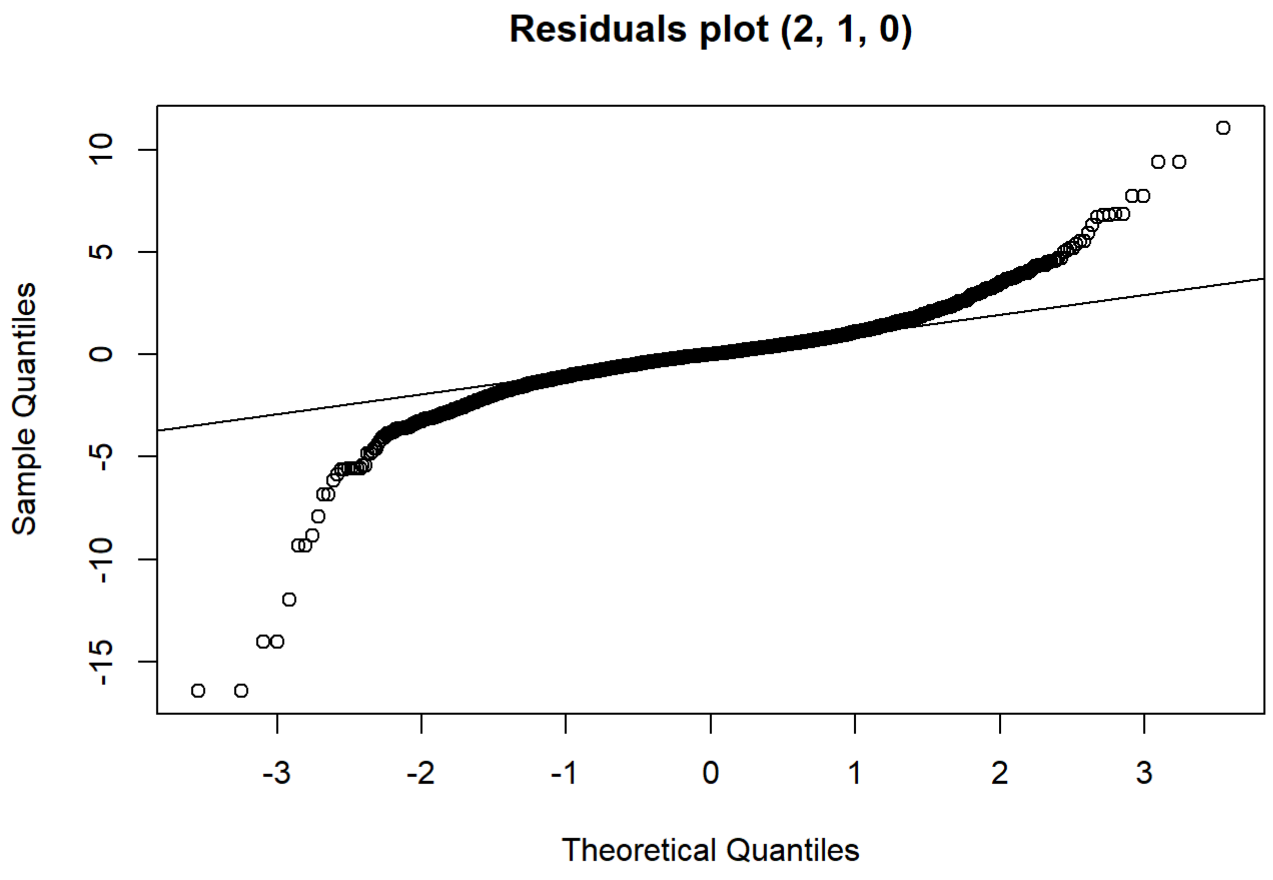


Residual Plot (2 , 1, 0)
Residuals
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021
Year



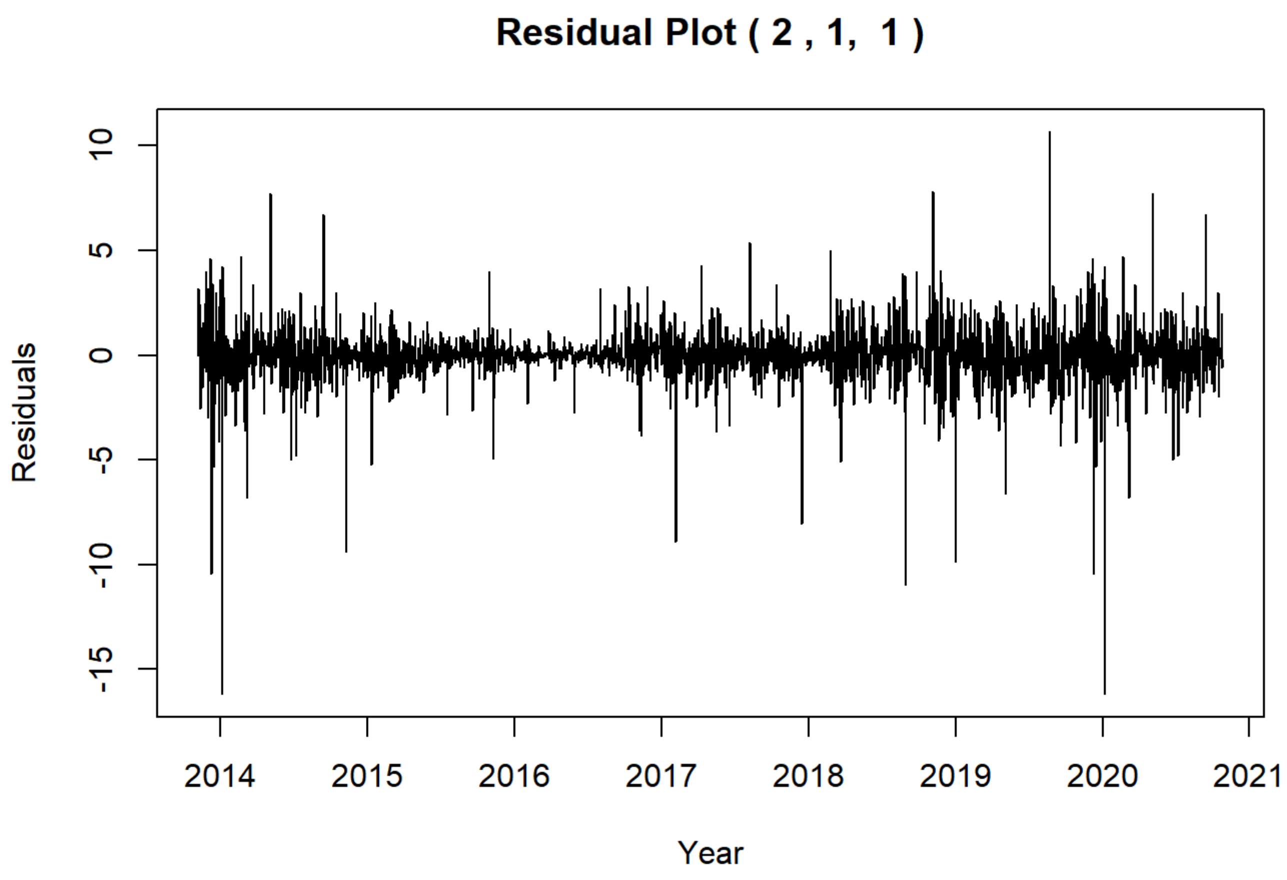
ACF of Residual (2 , 1, 0)
ACF
0.1
0.0
-0.1
-0.2
0
50
100
150
Lag

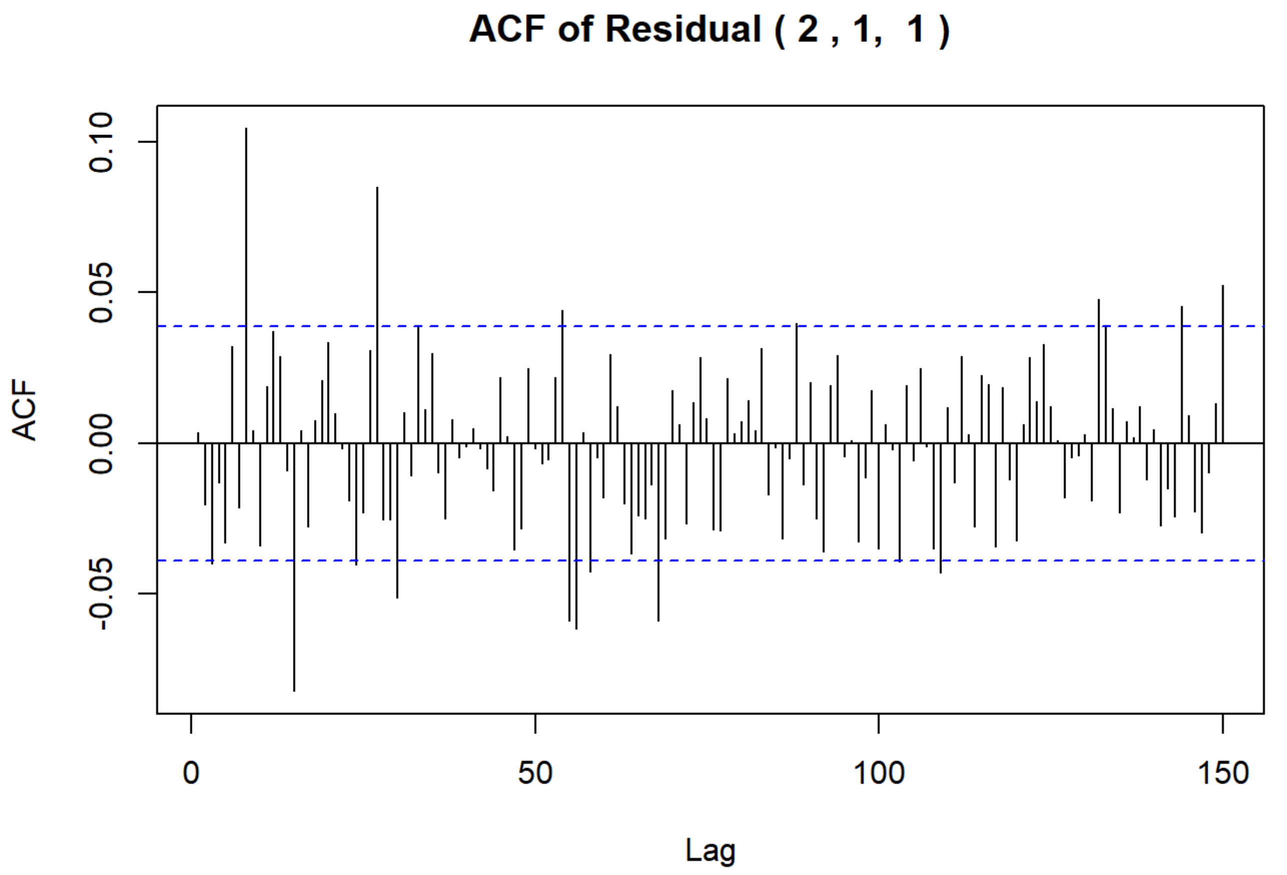


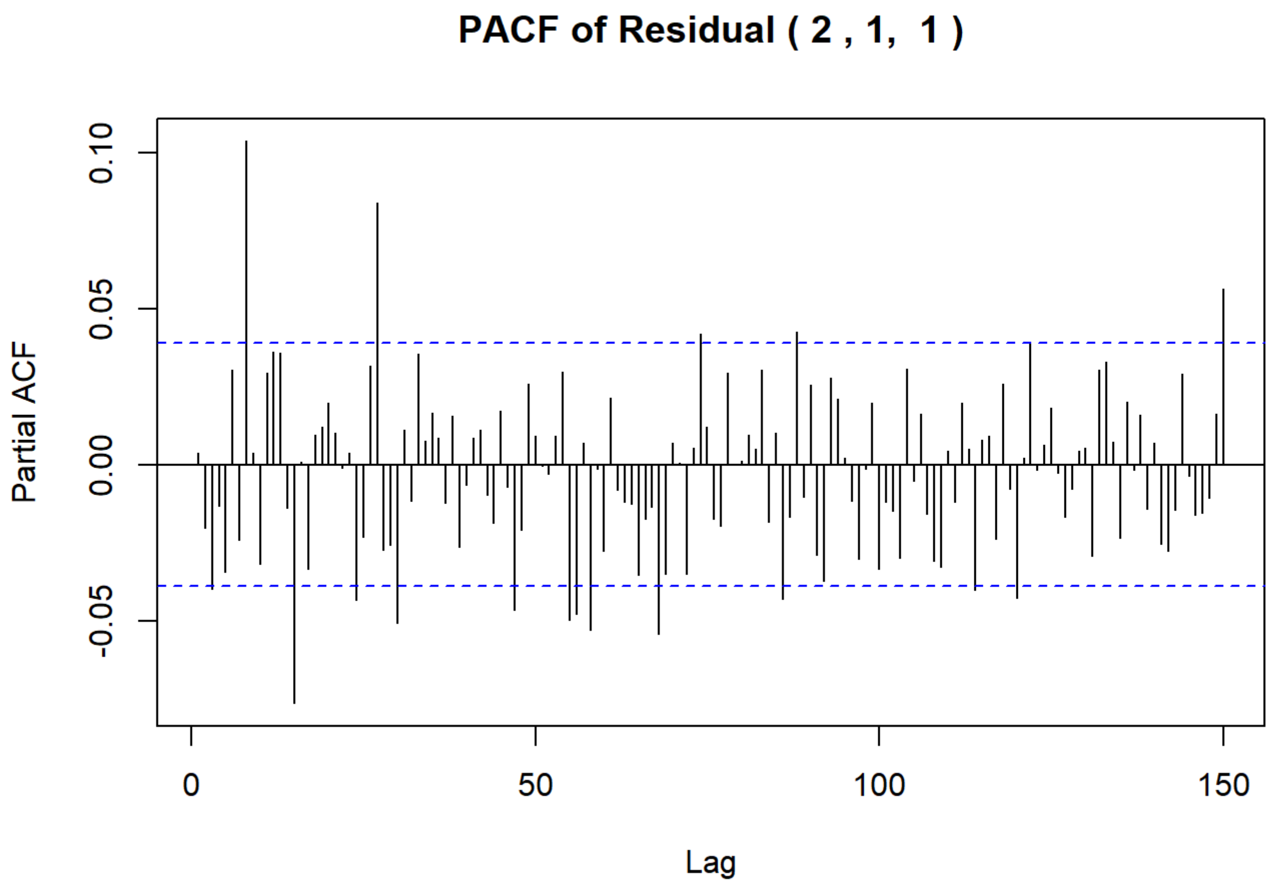


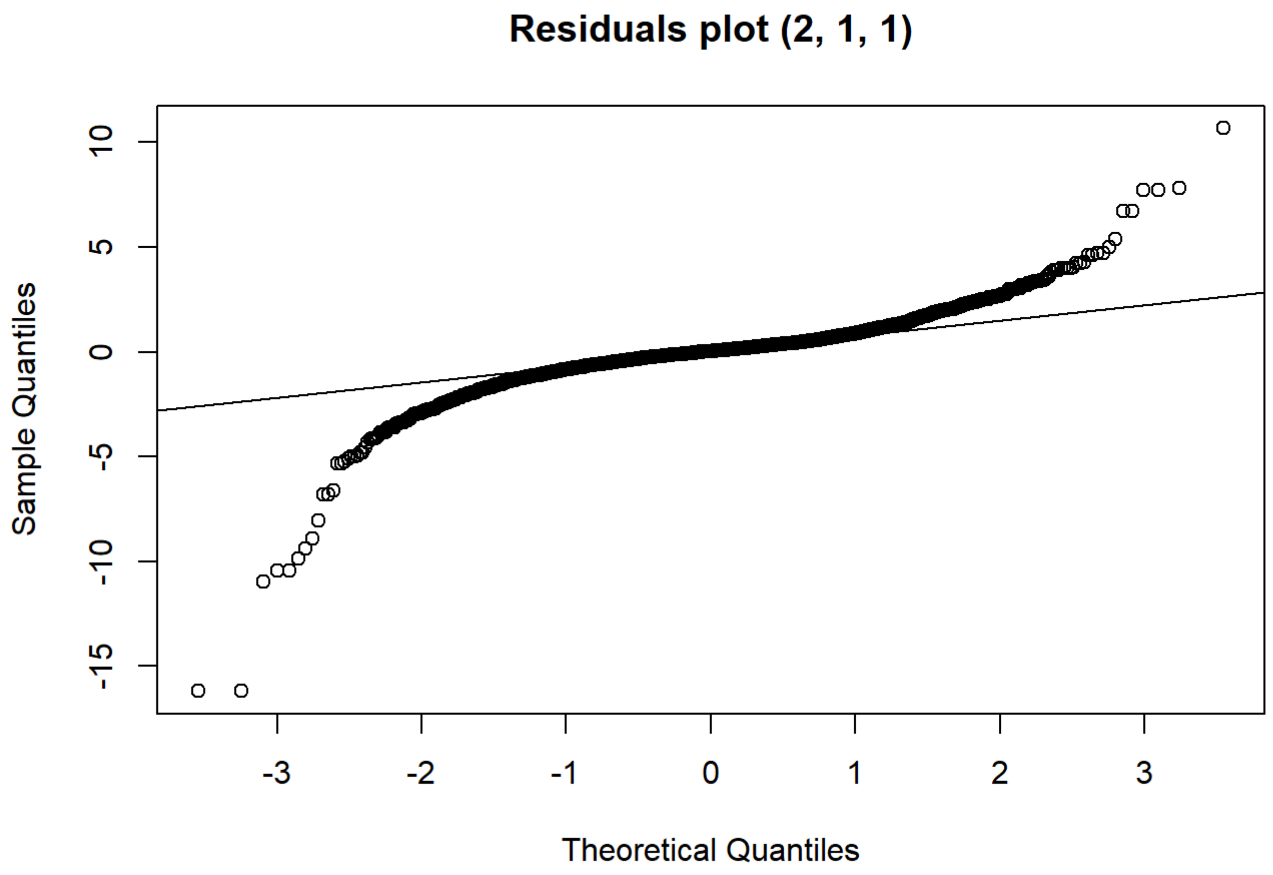
```
## AIC: 8106.817
## BIC: 8123.89
```

```
my_df <- plot_arima_residuals(arima211, my_df)
```


Residual Plot (2 , 1, 1)
Residuals
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021
Year

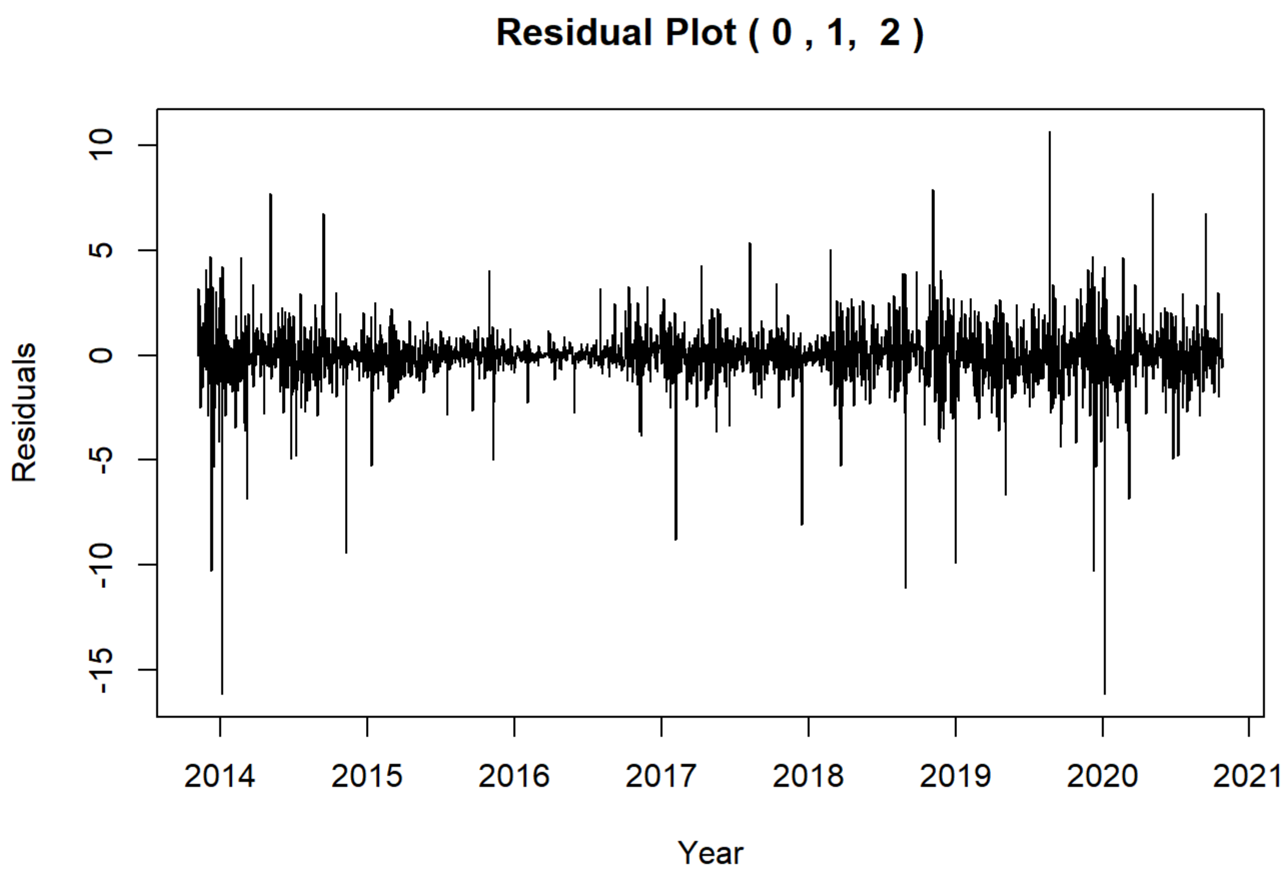

ACF of Residual (2 , 1, 1)
ACF
0.10
0.05
0.00
-0.05
0
50
100
150
Lag

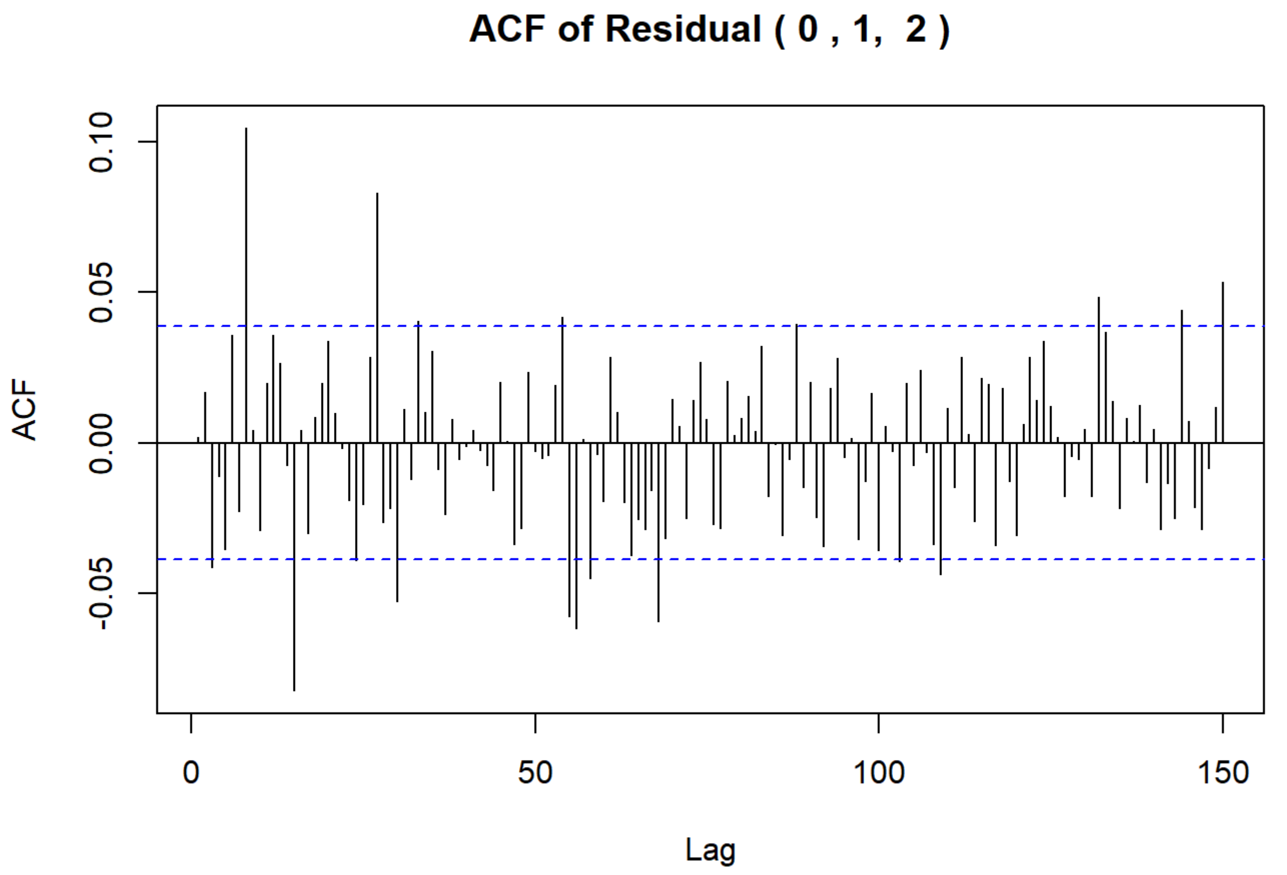


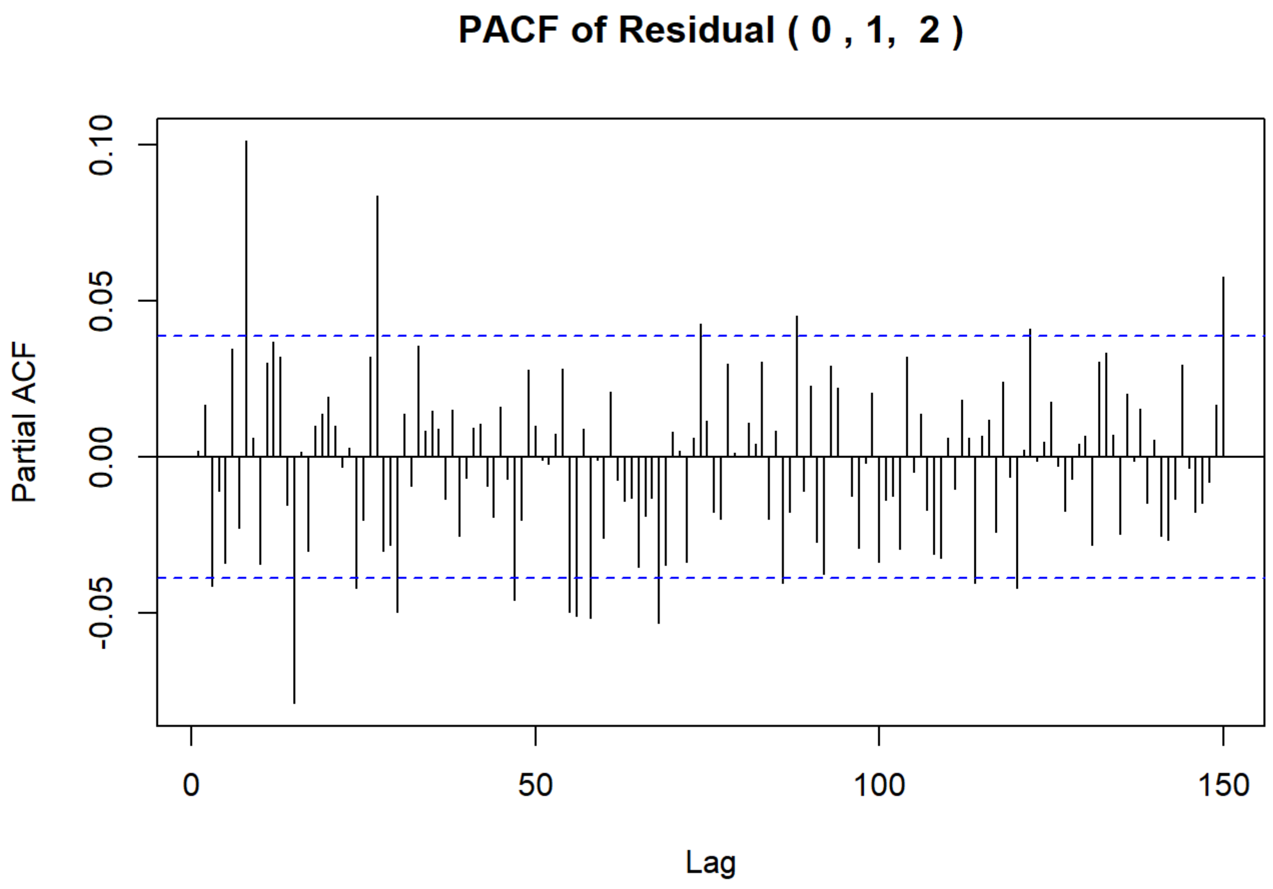


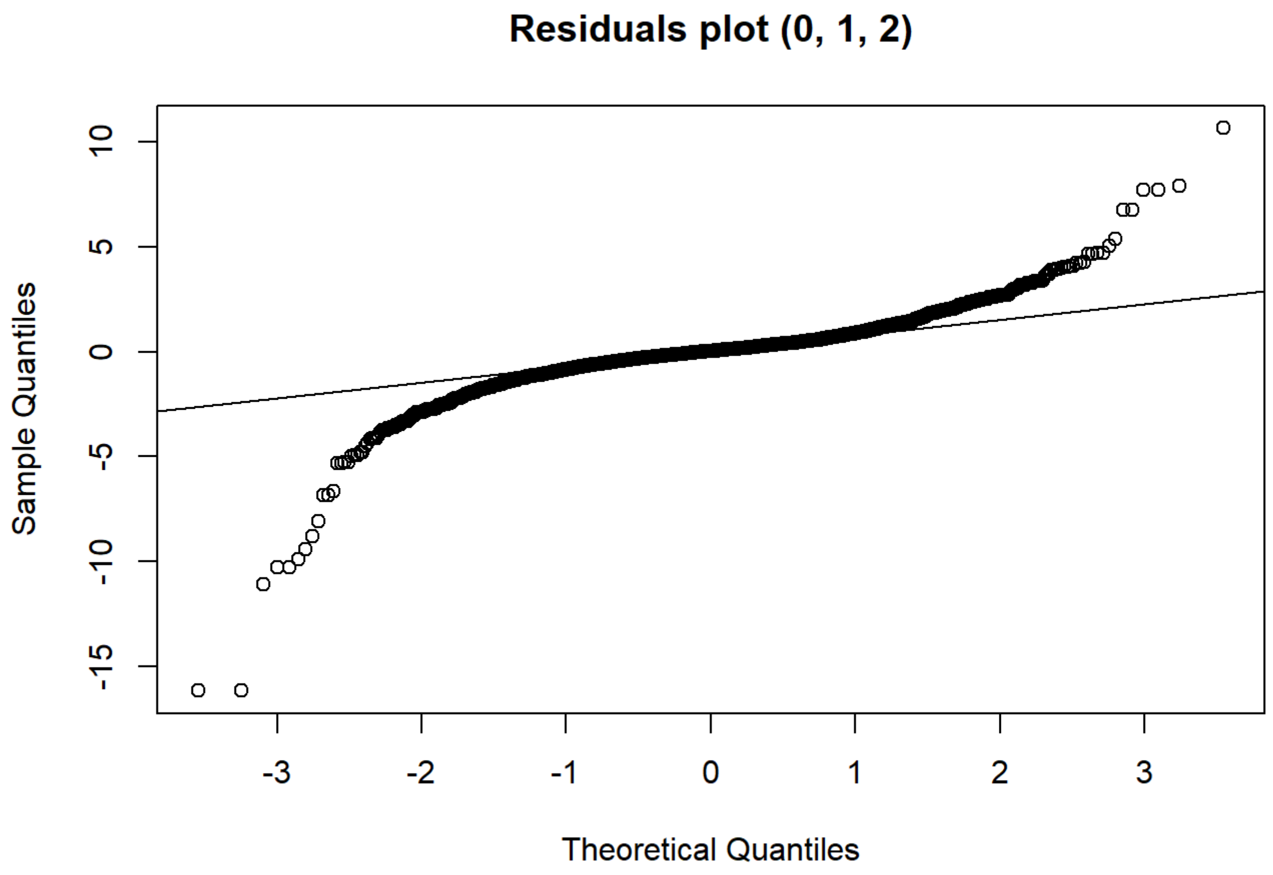
```
## AIC: 7476.025
## BIC: 7498.789
```

```
my_df <- plot_arima_residuals(arima012, my_df)
```


Residual Plot (0 , 1, 2)
Residuals
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021
Year

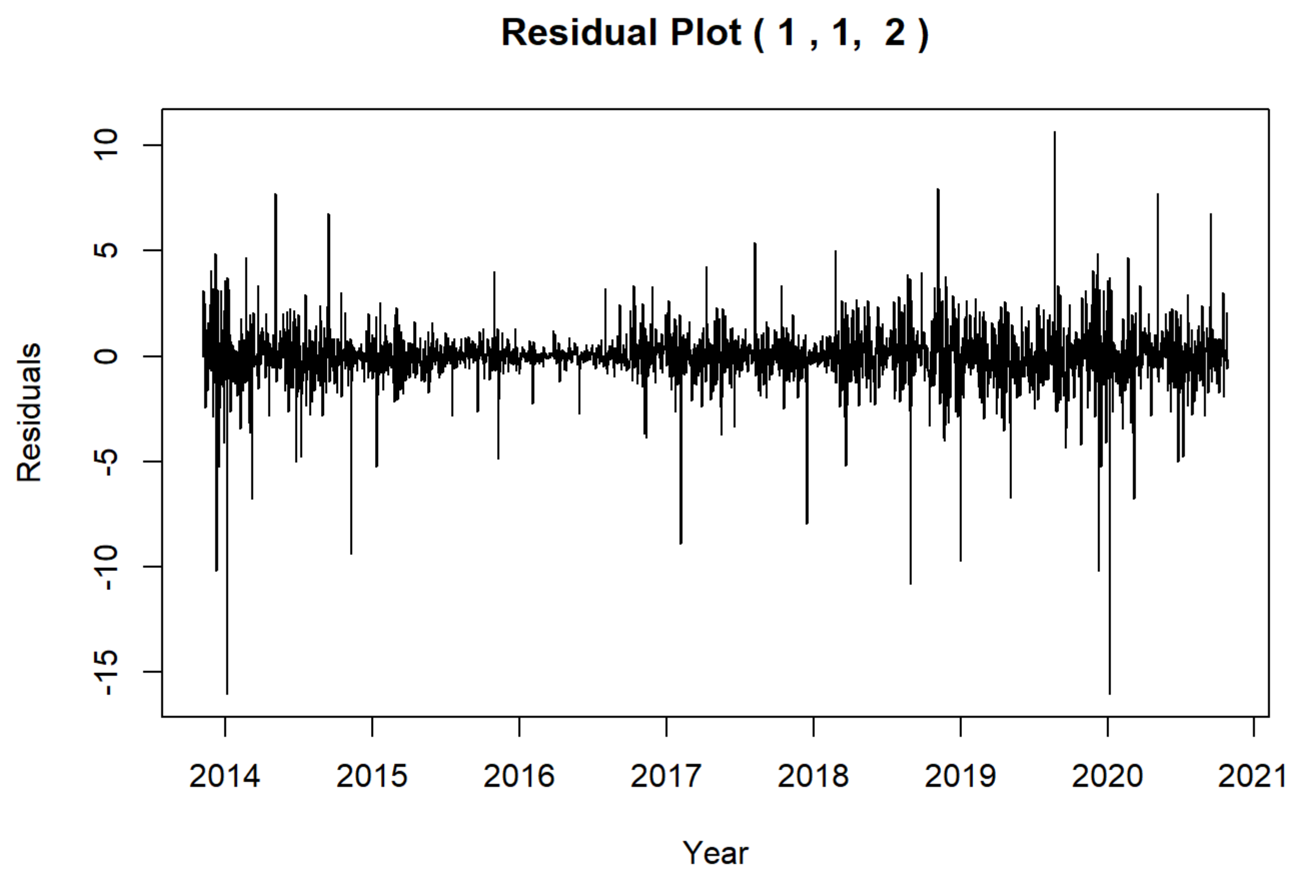

ACF of Residual (0 , 1, 2)
ACF
0.10
0.05
0.00
-0.05
0
50
100
150
Lag

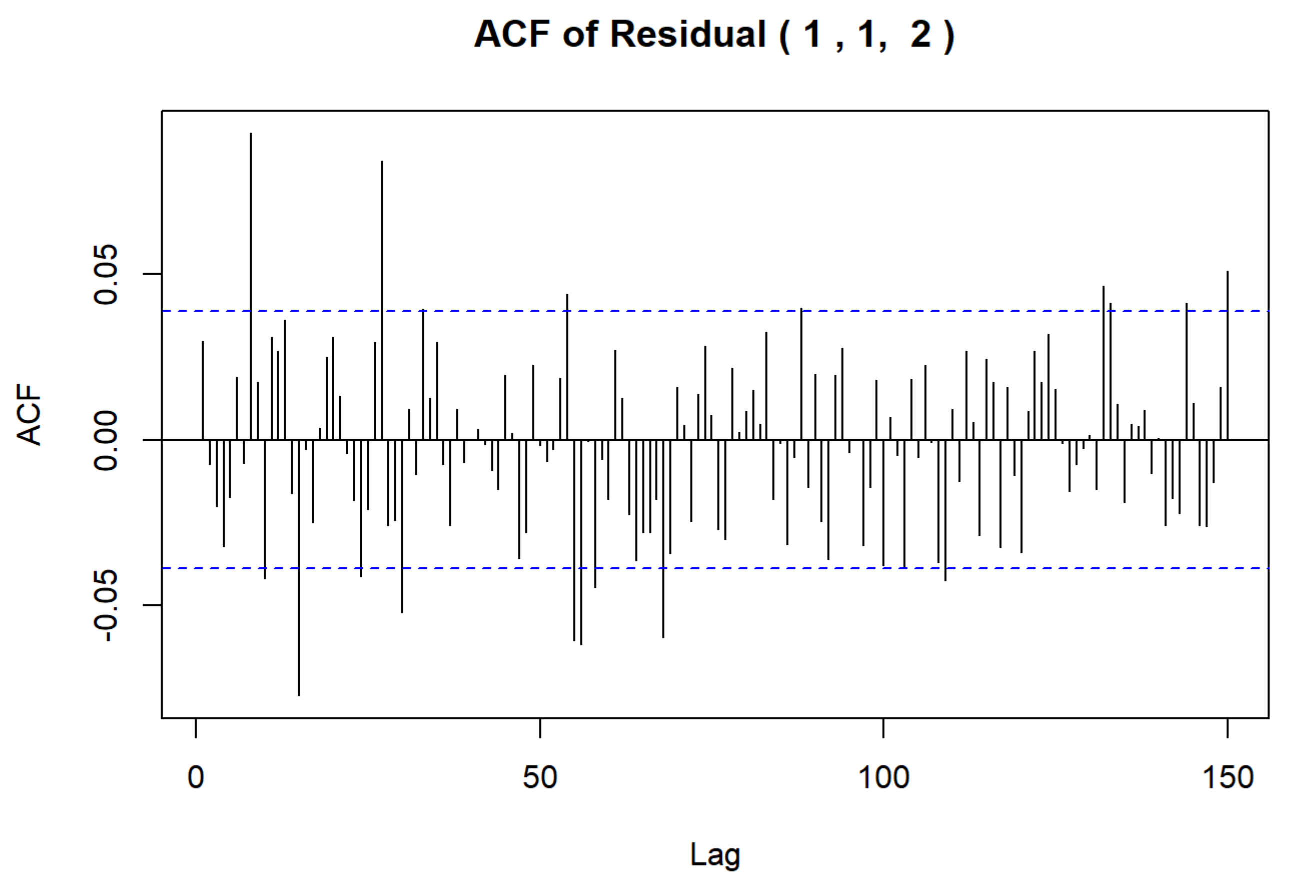


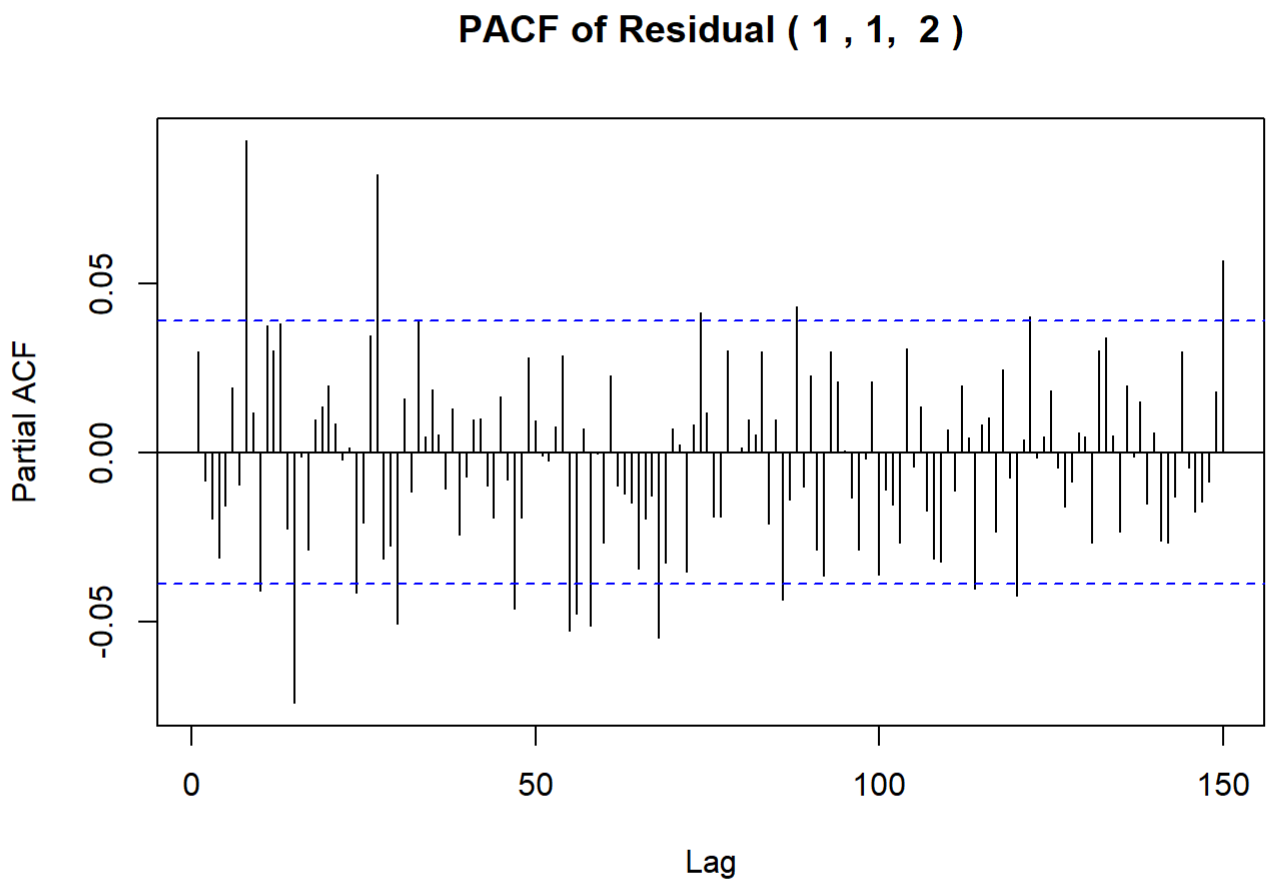


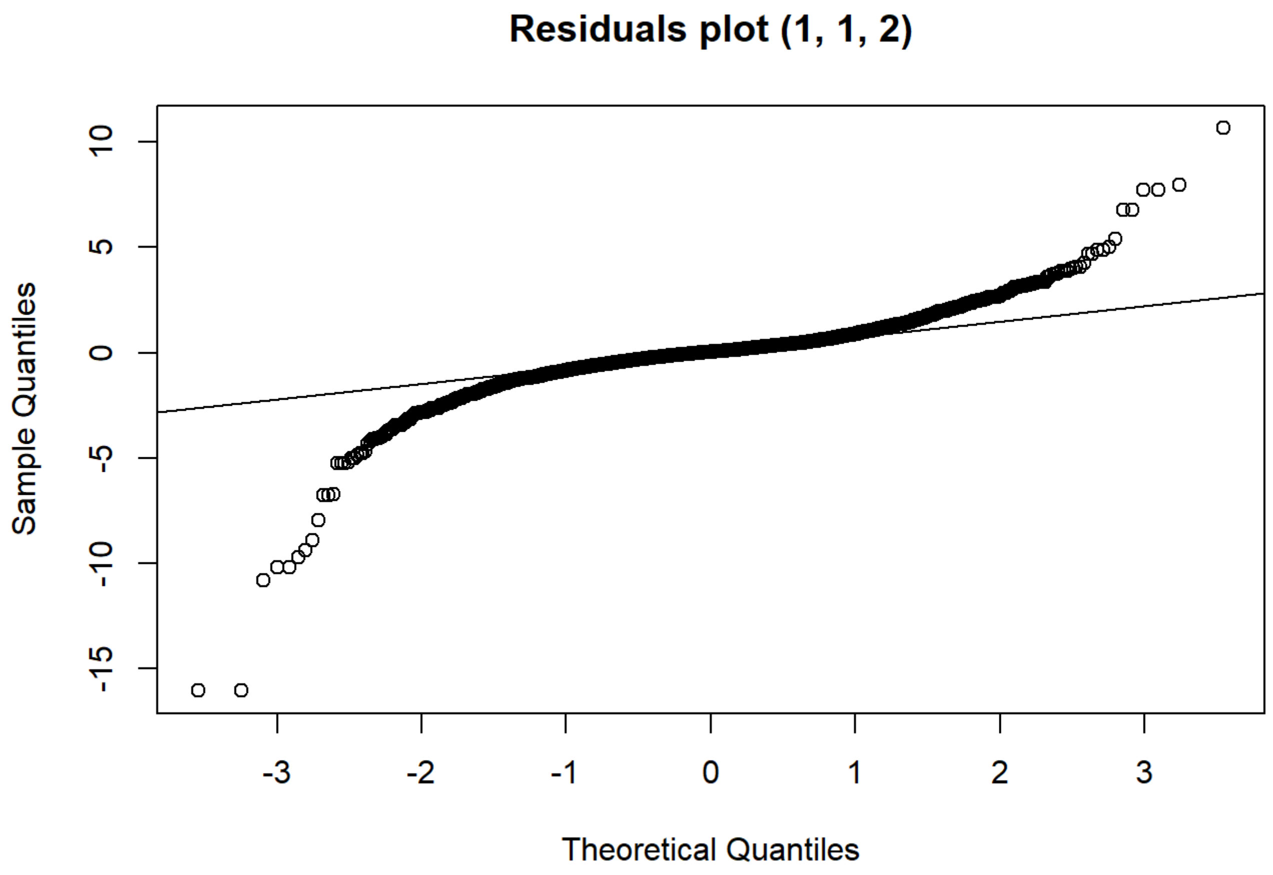
```
## AIC: 7477.111
## BIC: 7494.185
```

```
my_df <- plot_arima_residuals(arima112, my_df)
```


Residual Plot (1 , 1, 2)
Residuals
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021
Year


ACF of Residual (1 , 1, 2)
ACF
0.05
0.00
-0.05
0
50
100
150
Lag

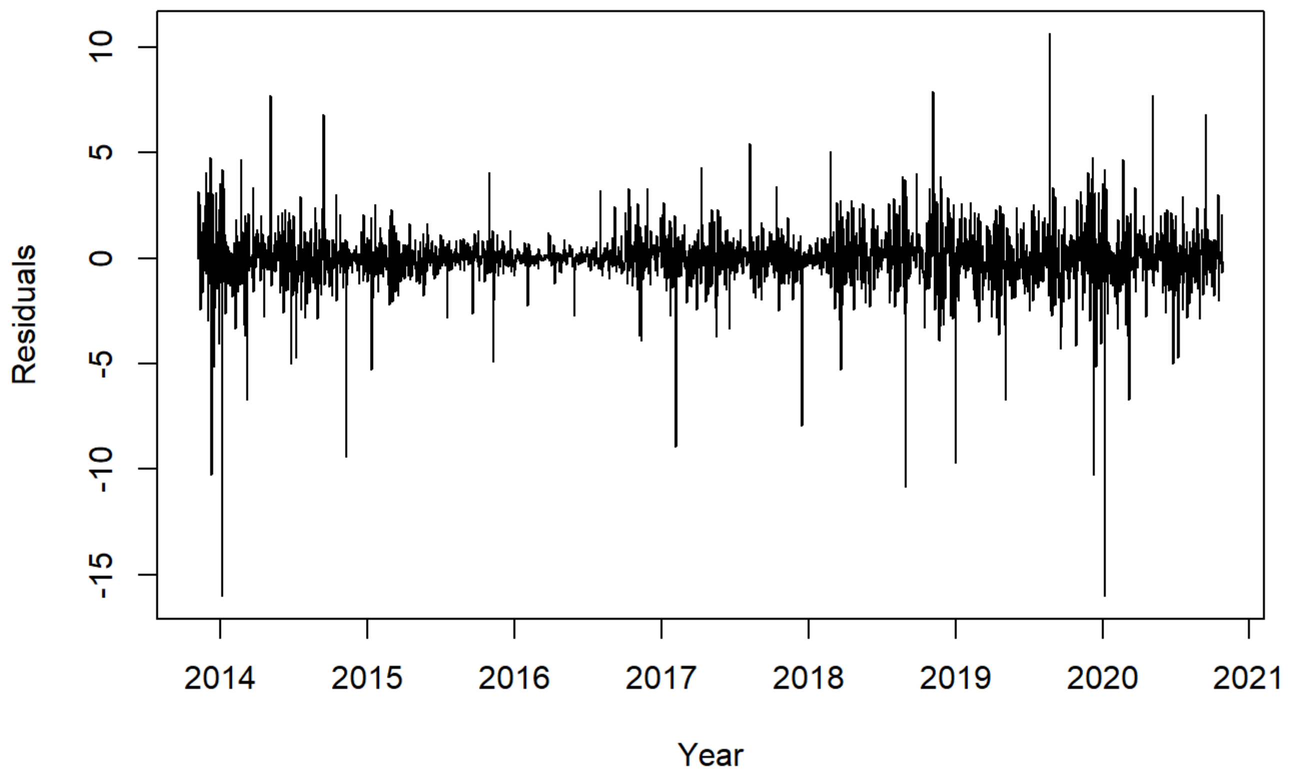




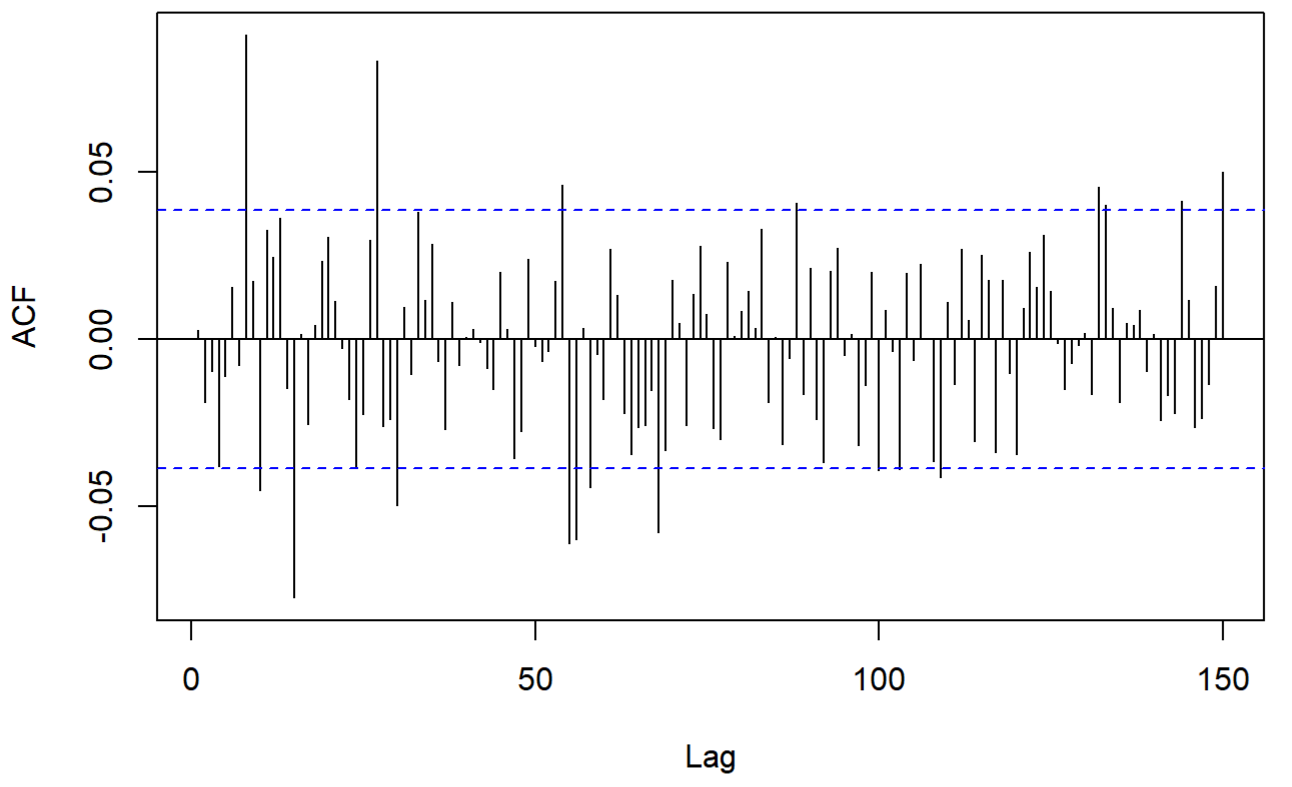
```
## AIC: 7472.119
## BIC: 7494.884
```

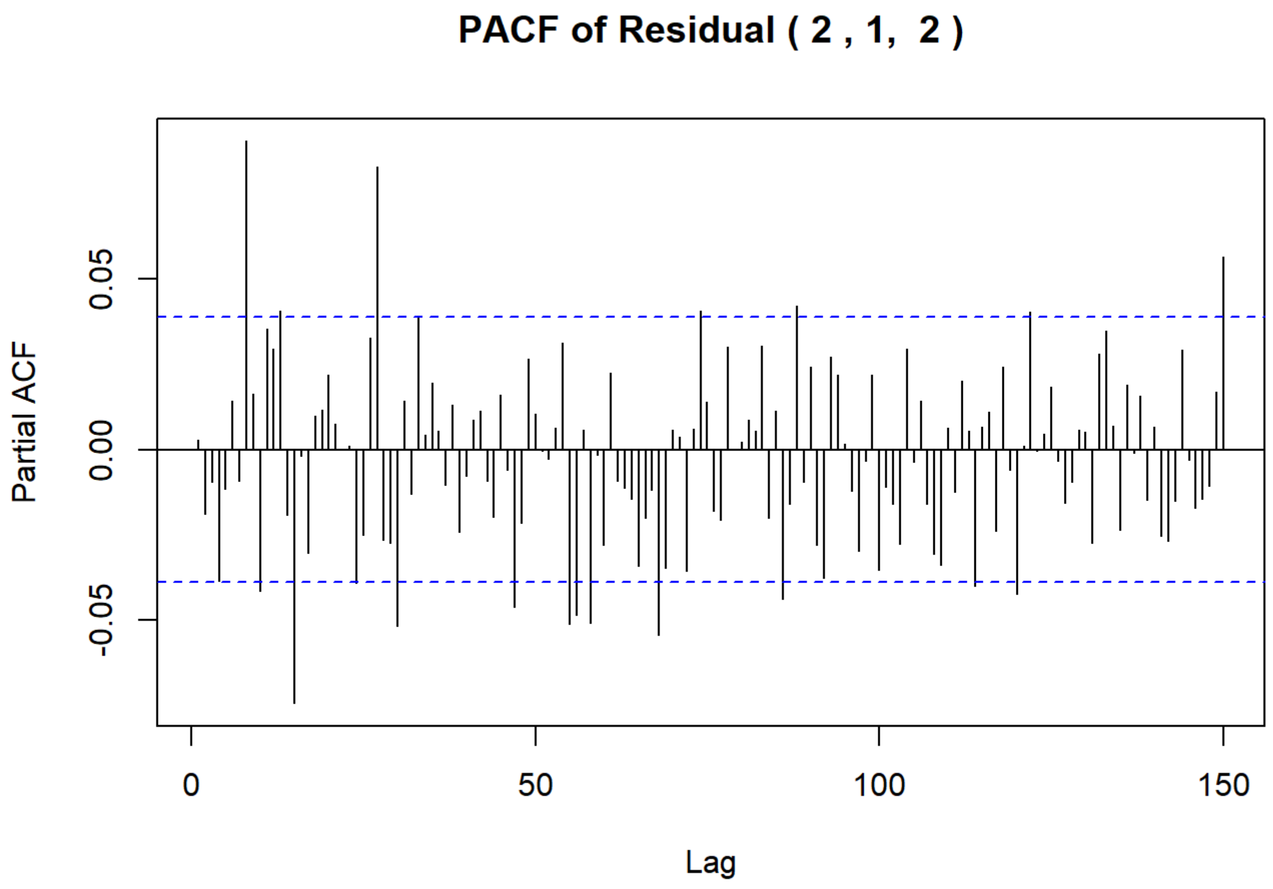
```
my_df <- plot_arima_residuals(arima212, my_df)
```

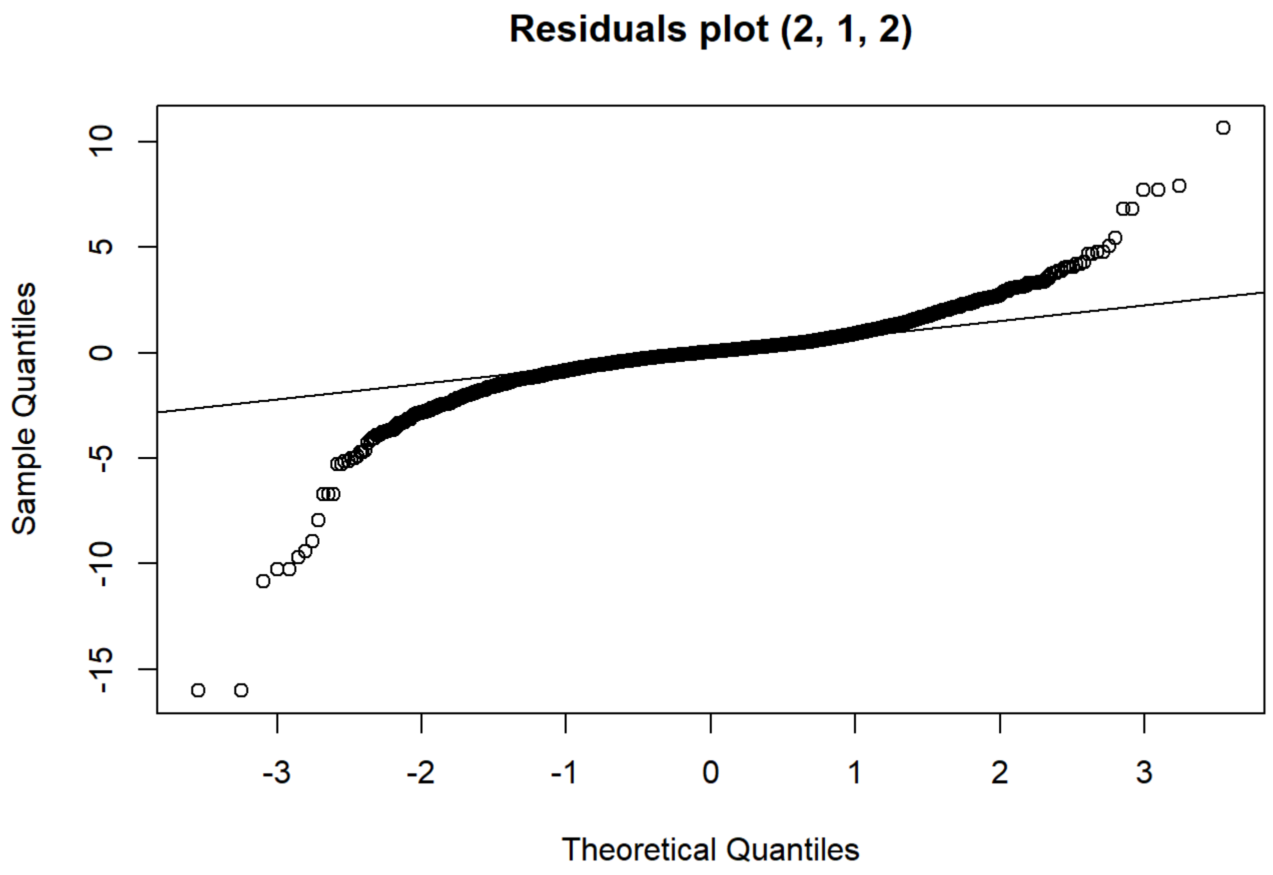
Residual Plot (2 , 1, 2)



ACF of Residual (2 , 1, 2)

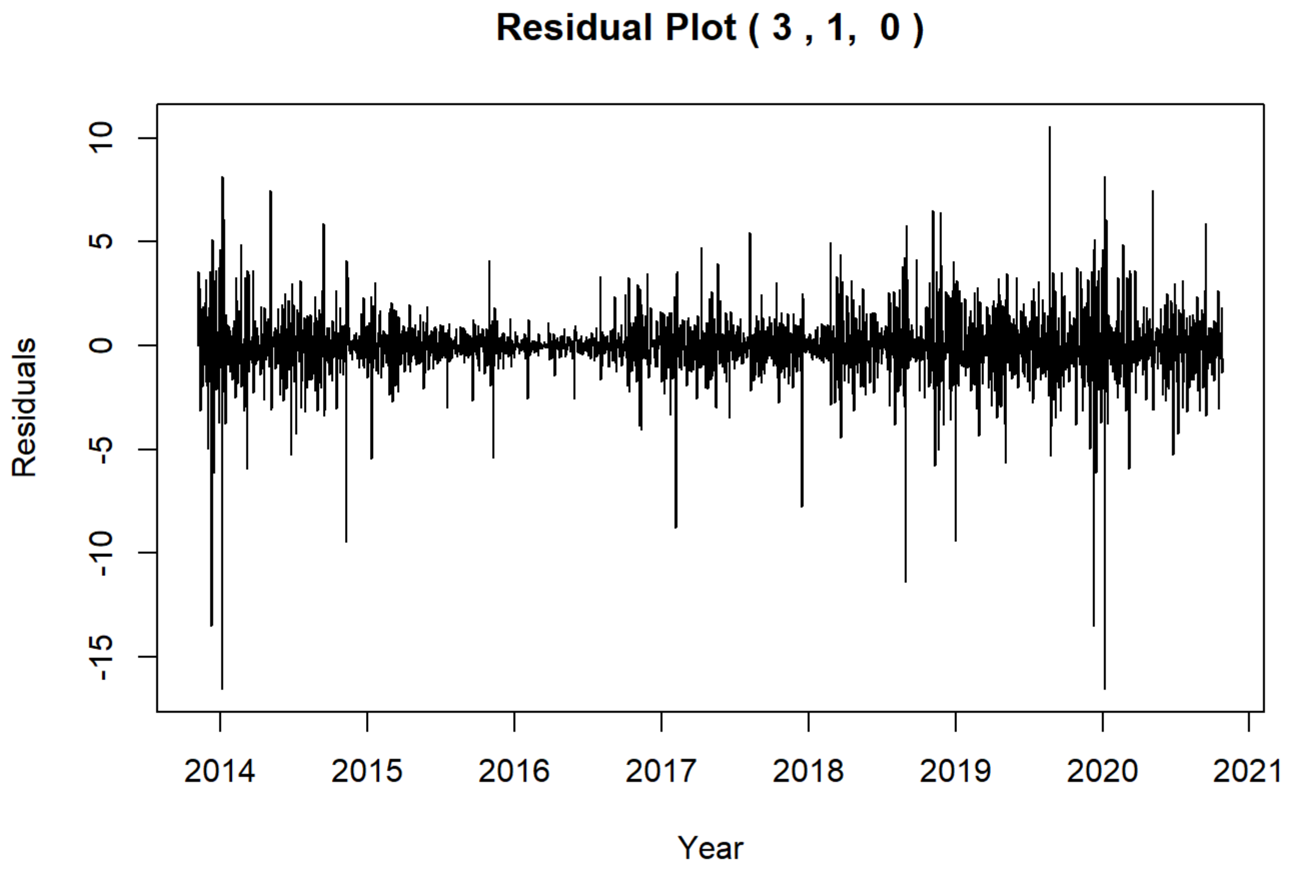




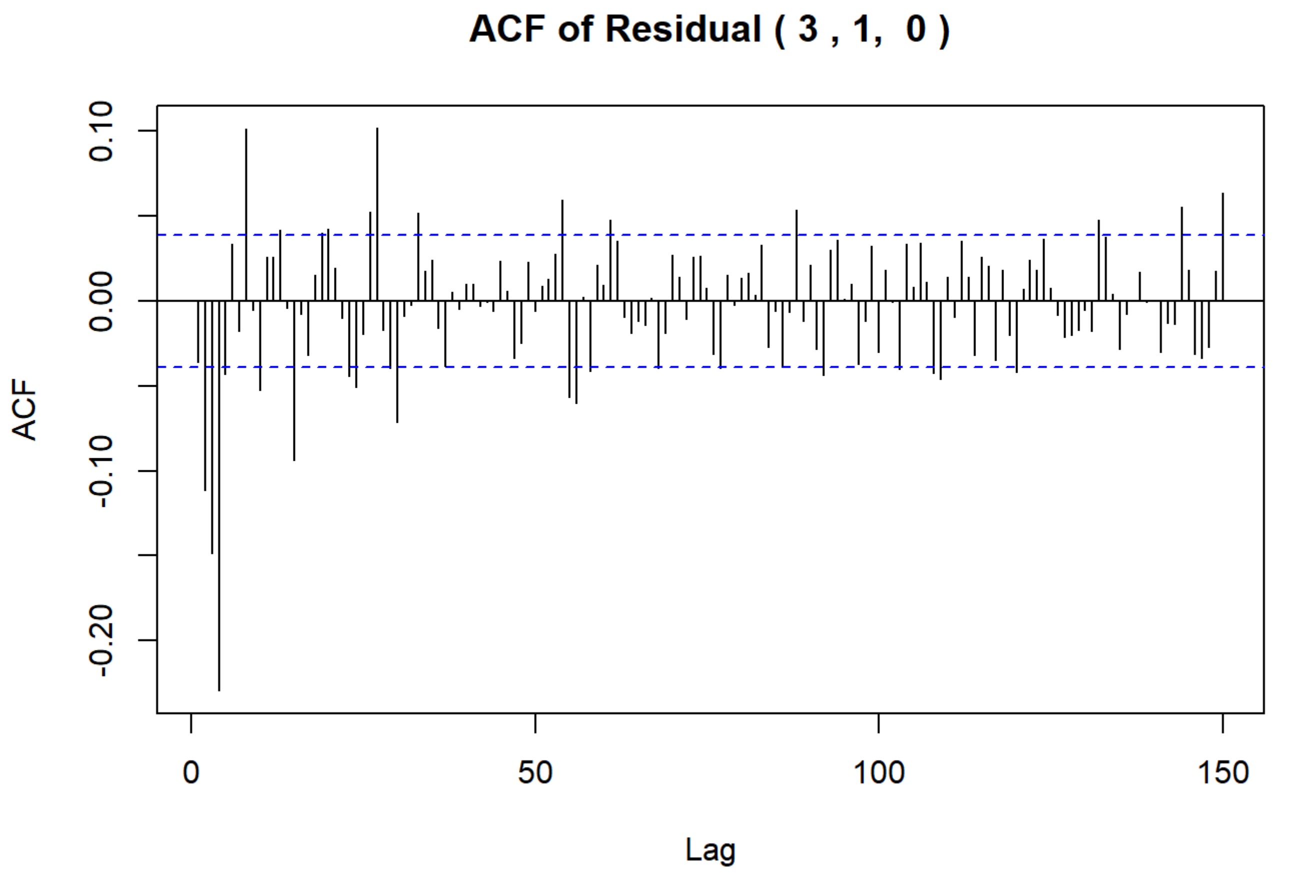


```
## AIC: 7471.717
## BIC: 7500.173
```

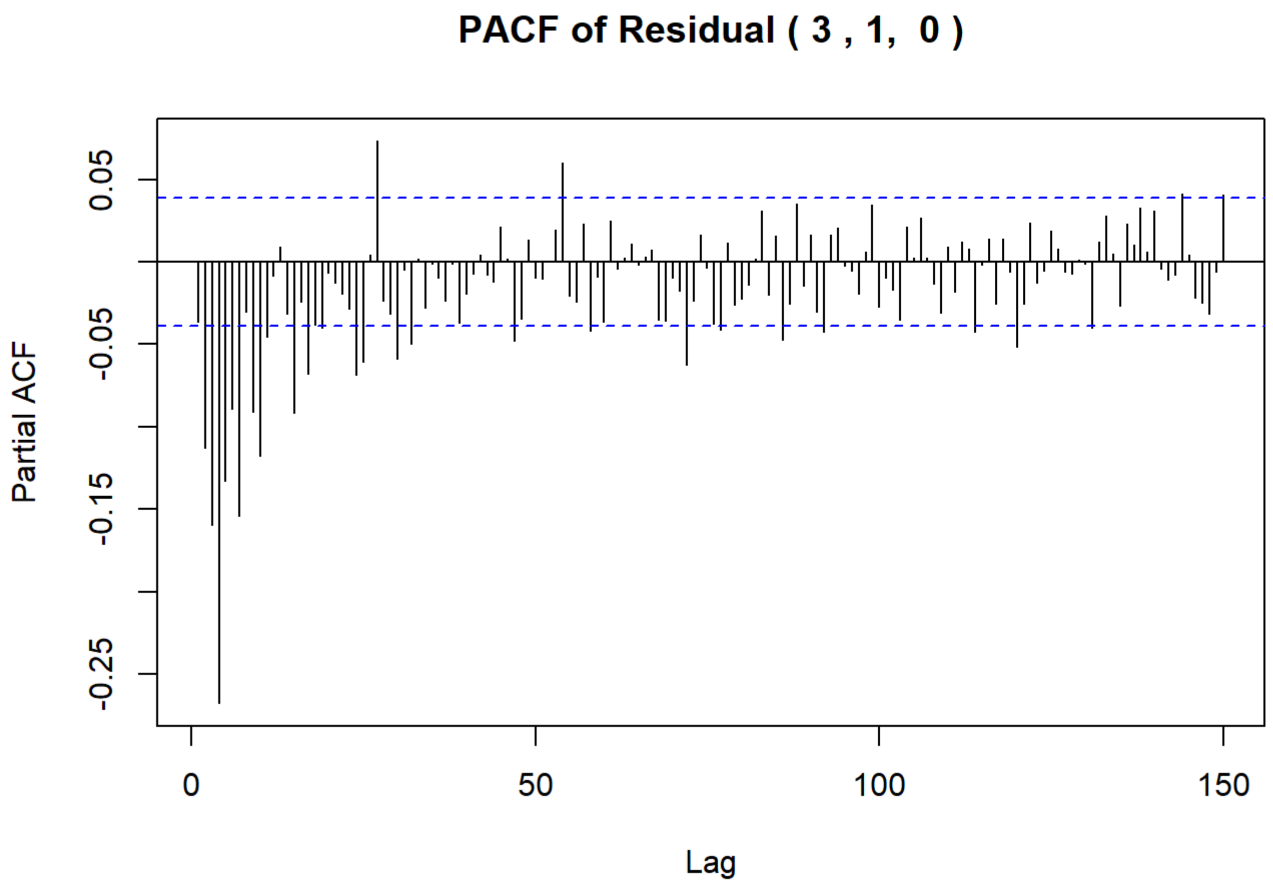
```
my_df <- plot_arima_residuals(arima310, my_df)
```

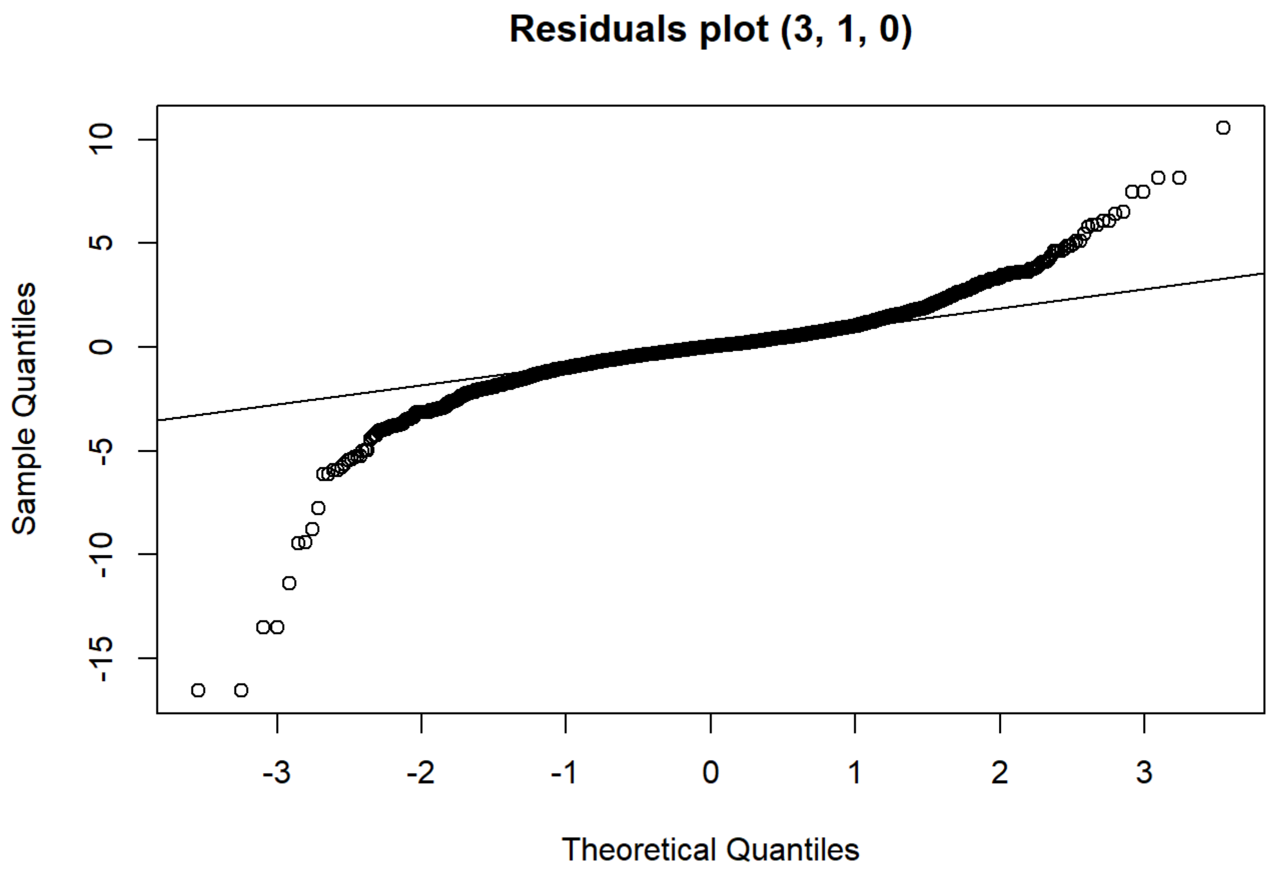


Residual Plot (3 , 1, 0)
Residuals
Year
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021



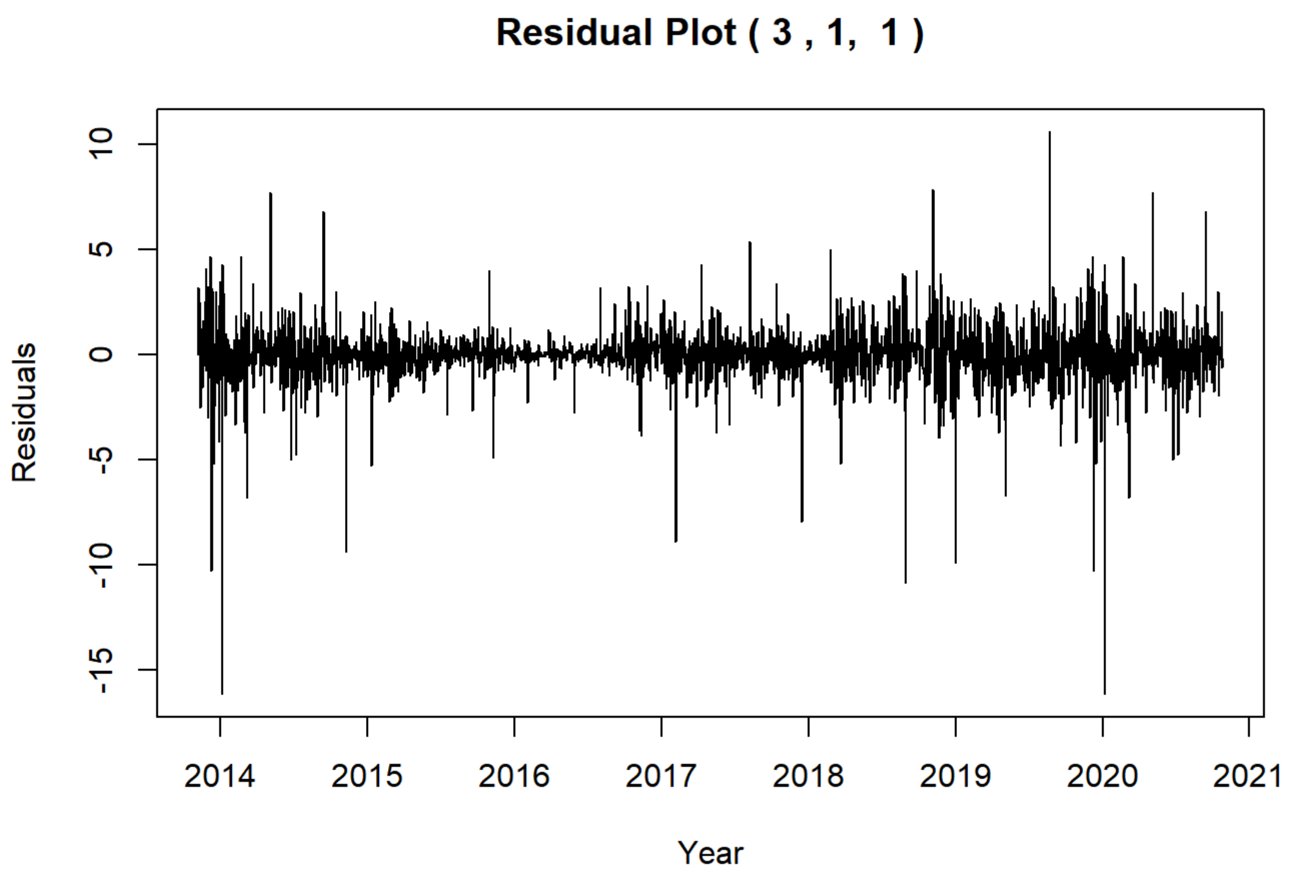
ACF of Residual (3 , 1, 0)
ACF
Lag
0.10
0.00
-0.10
-0.20
0
50
100
150



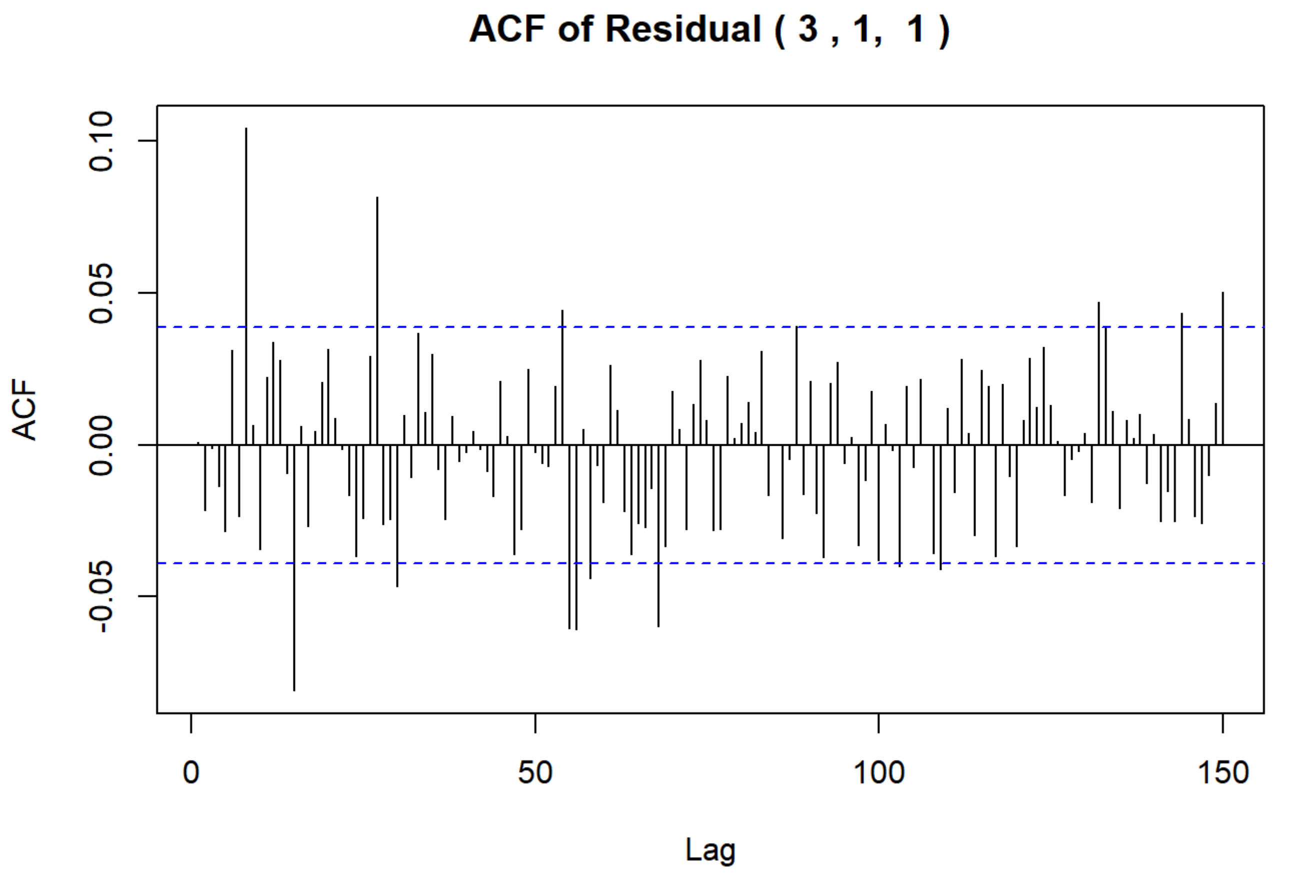


```
## AIC: 7966.435
## BIC: 7989.2
```

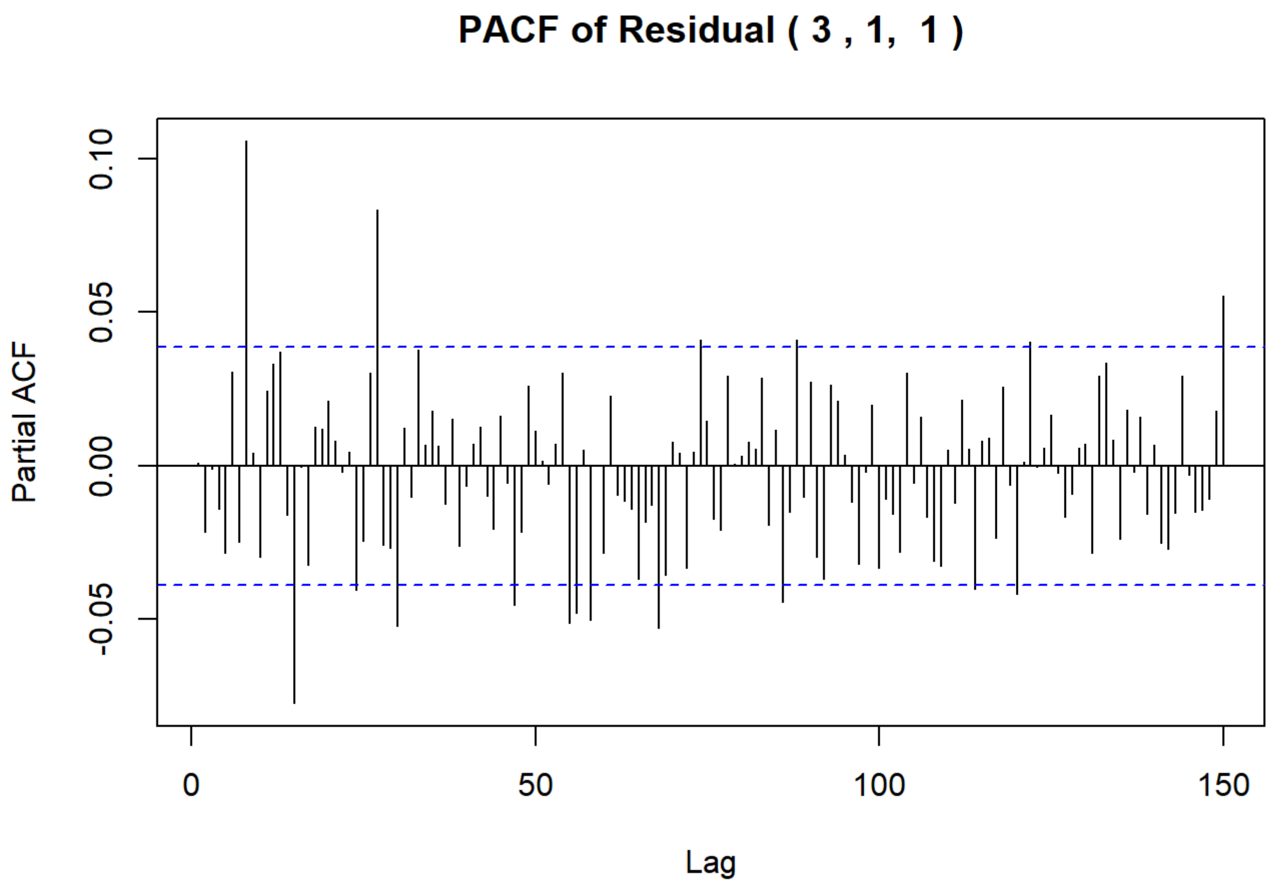
```
my_df <- plot_arima_residuals(arima311, my_df)
```

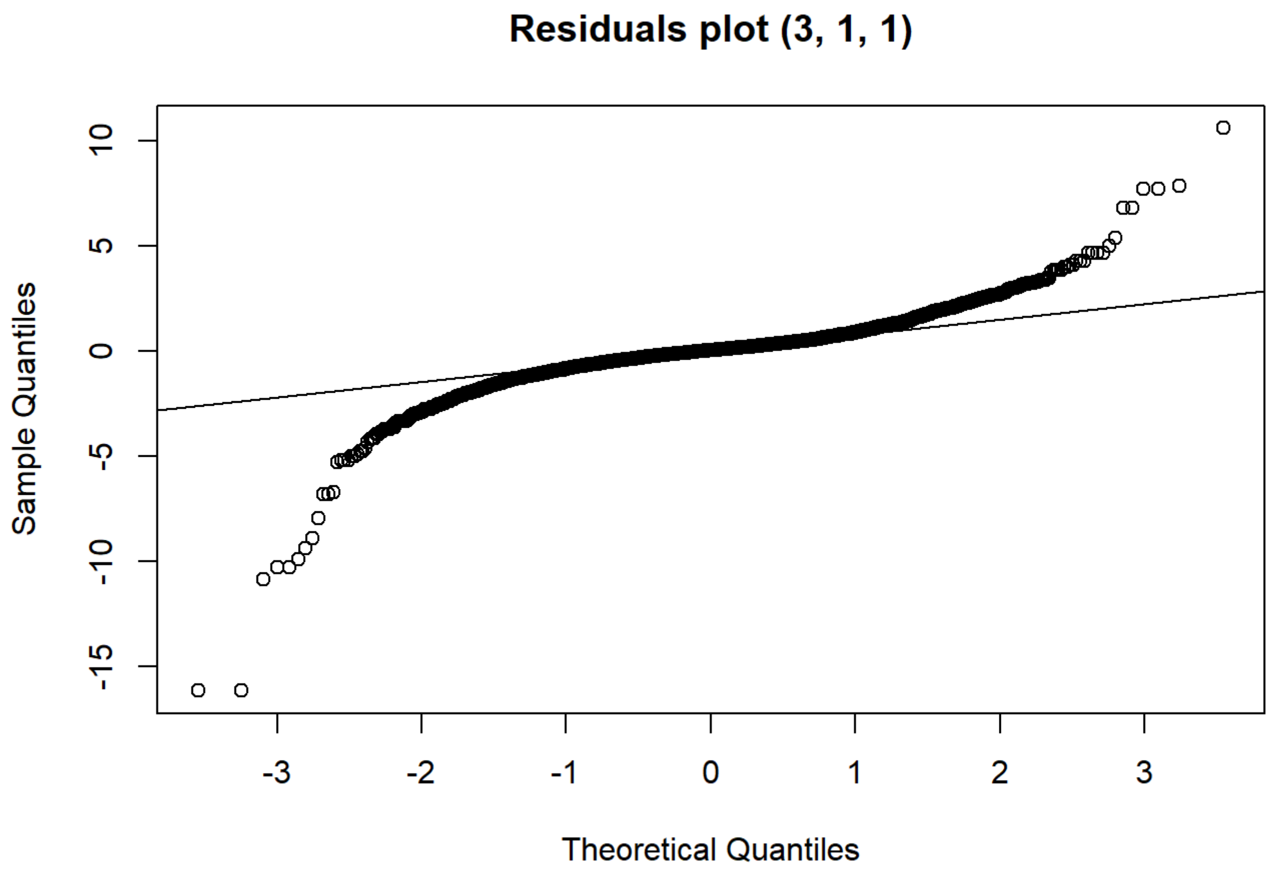


Residual Plot (3 , 1, 1)
Residuals
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021
Year



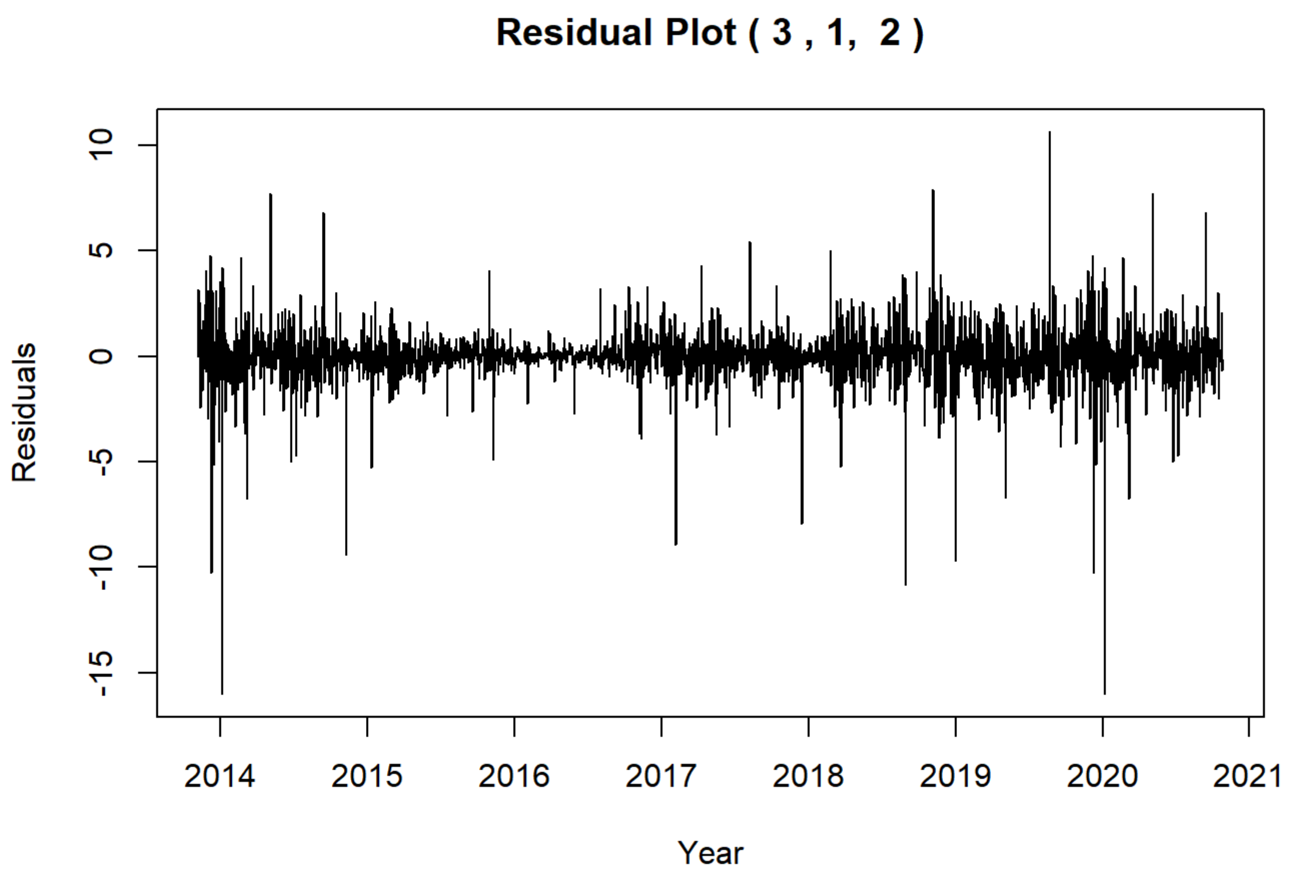
ACF of Residual (3 , 1, 1)
ACF
0.10
0.05
0.00
-0.05
0
50
100
150
Lag

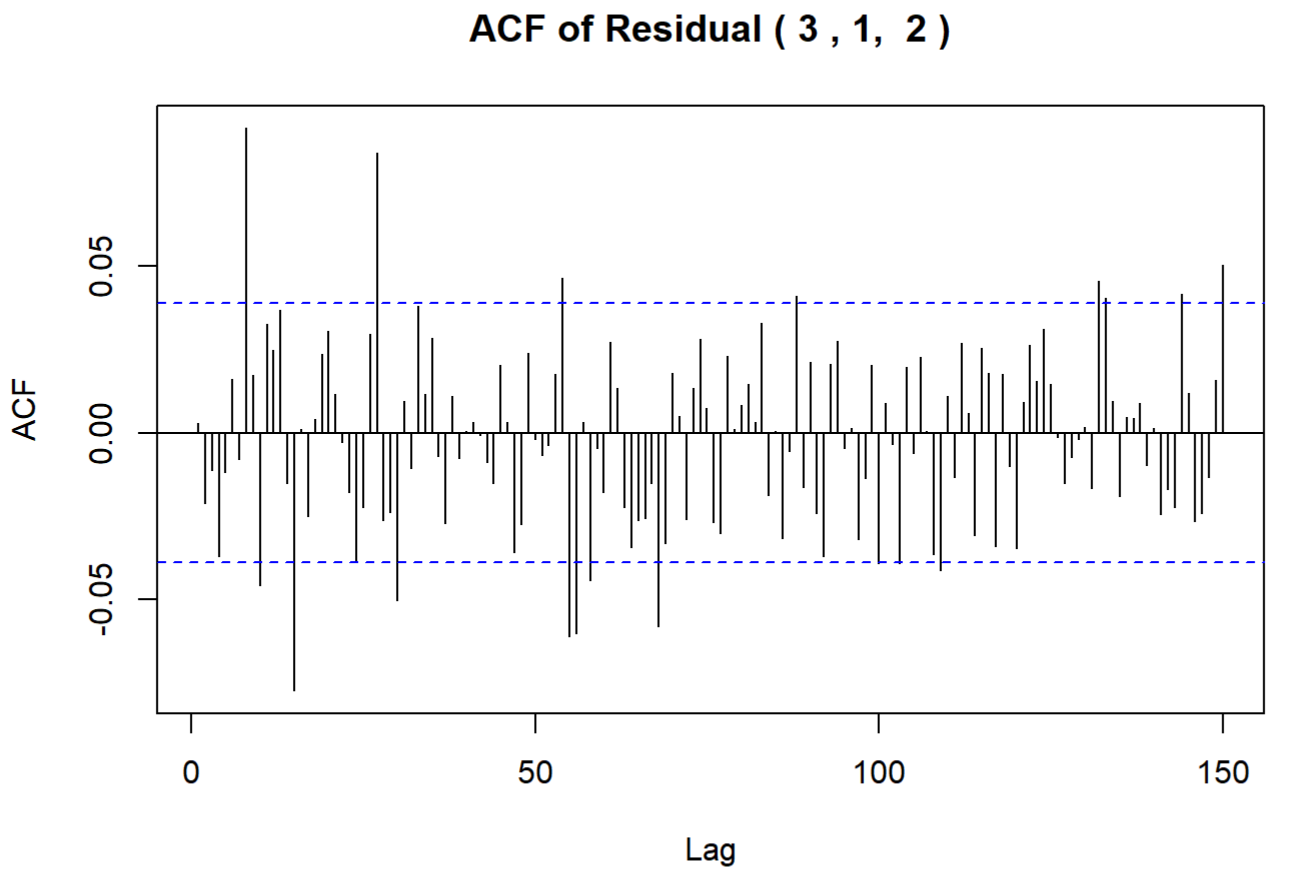


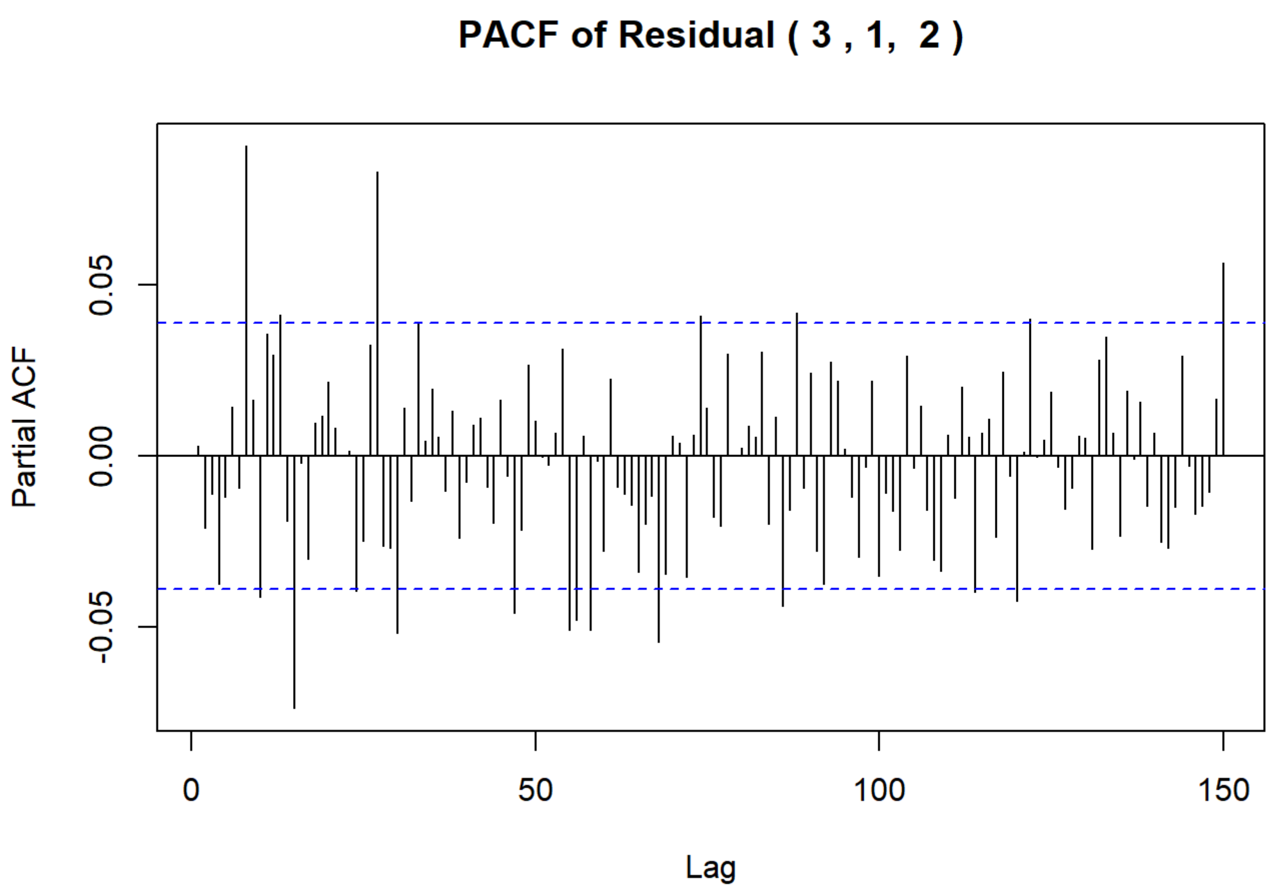


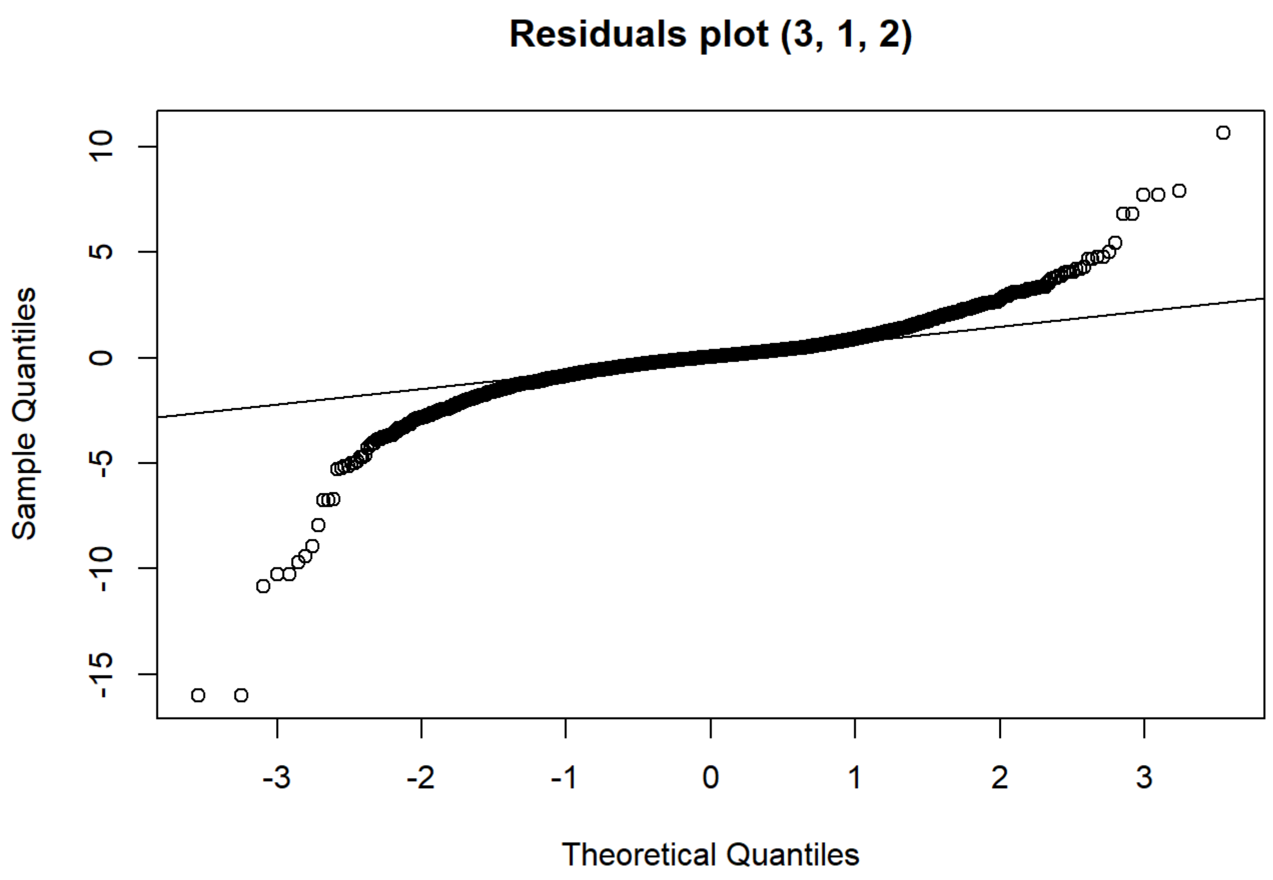
```
## AIC: 7474.882
## BIC: 7503.338
```

```
my_df <- plot_arima_residuals(arima312, my_df)
```


Residual Plot (3 , 1, 2)
Residuals
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021
Year

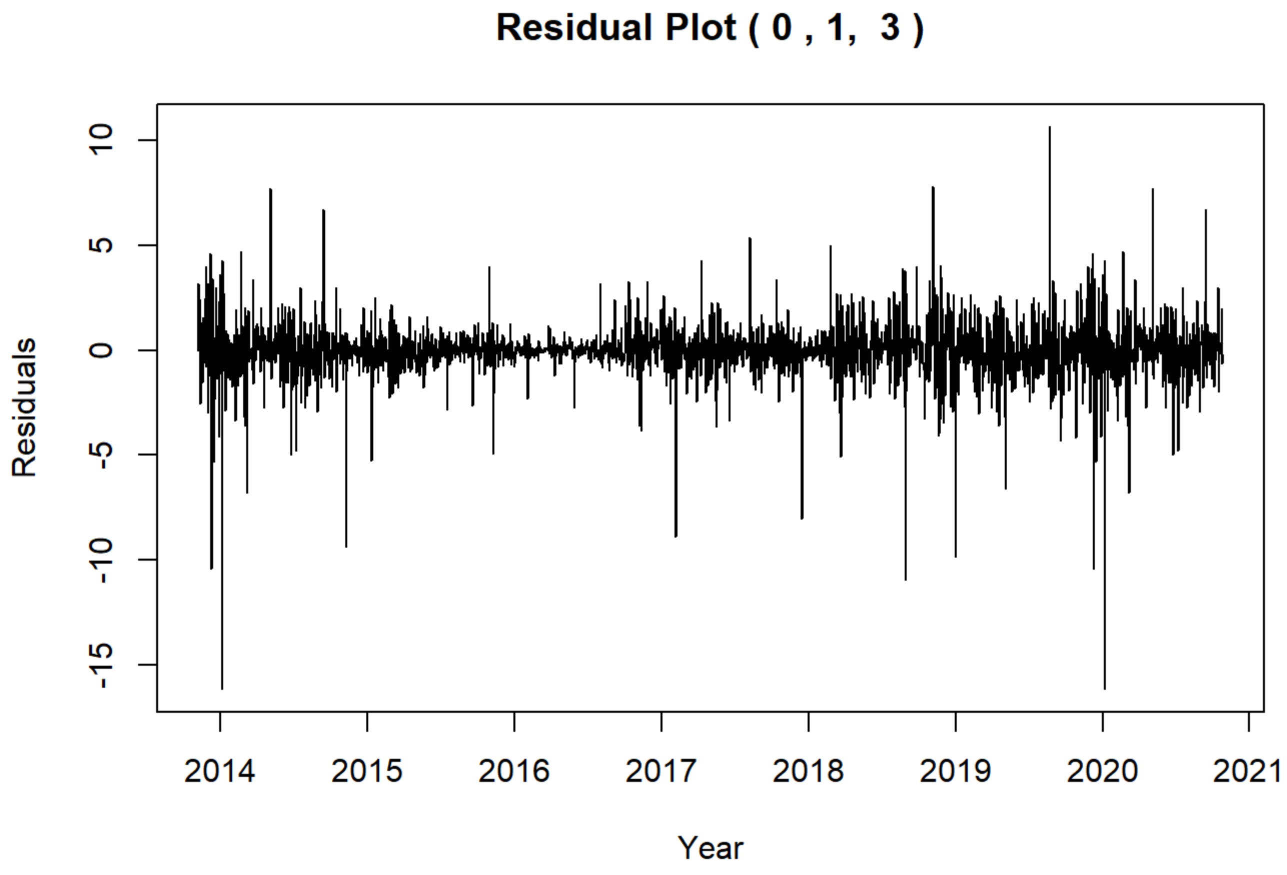

ACF of Residual (3 , 1, 2)
ACF
0.05
0.00
-0.05
0
50
100
150
Lag

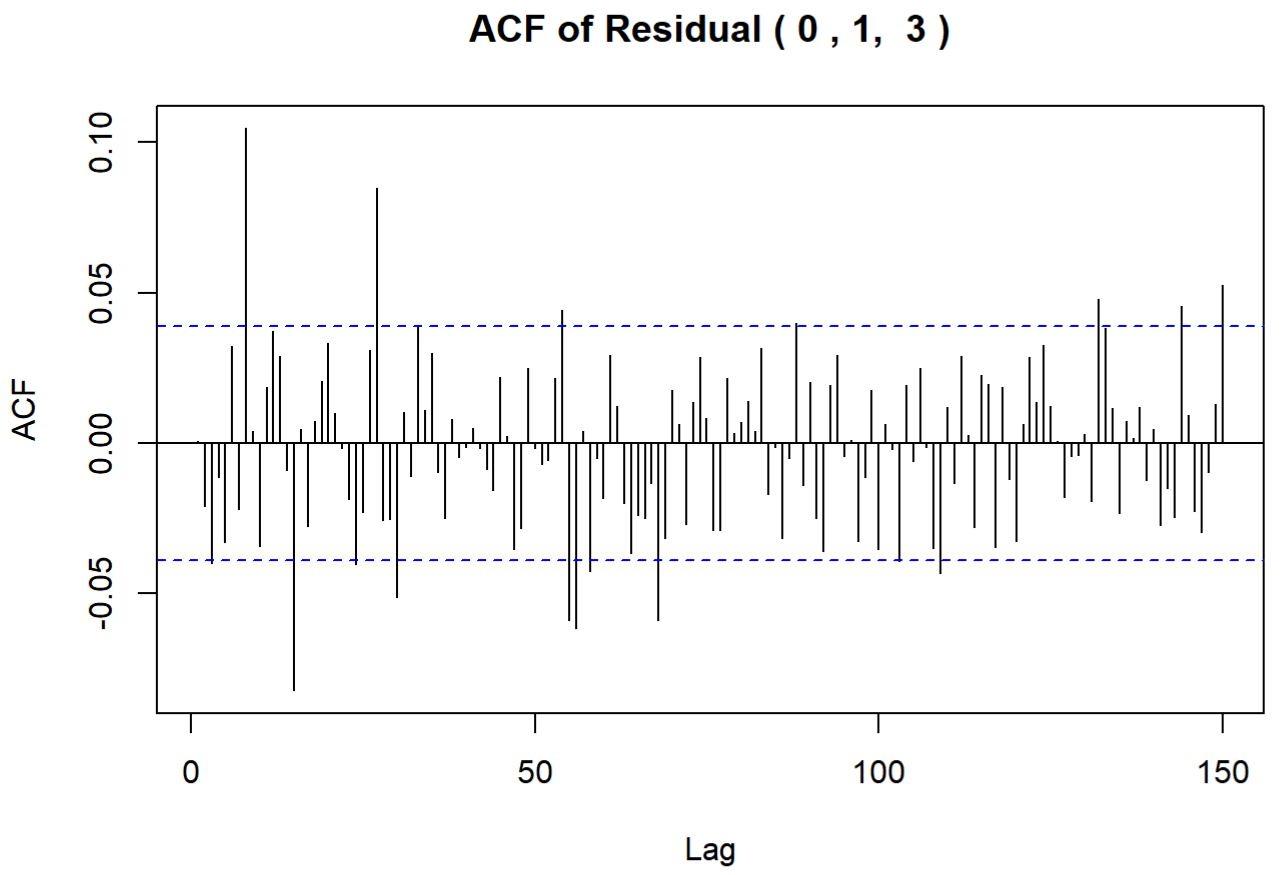


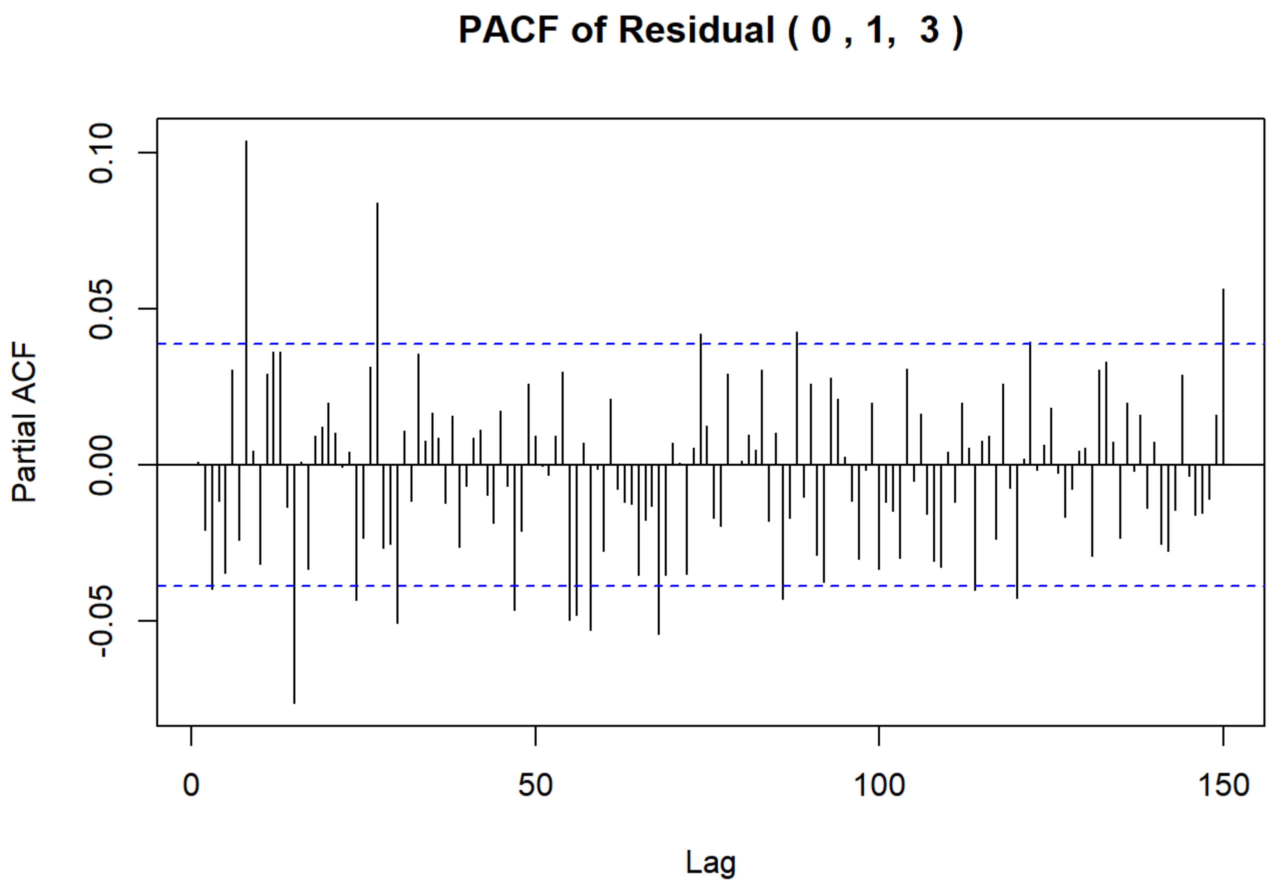


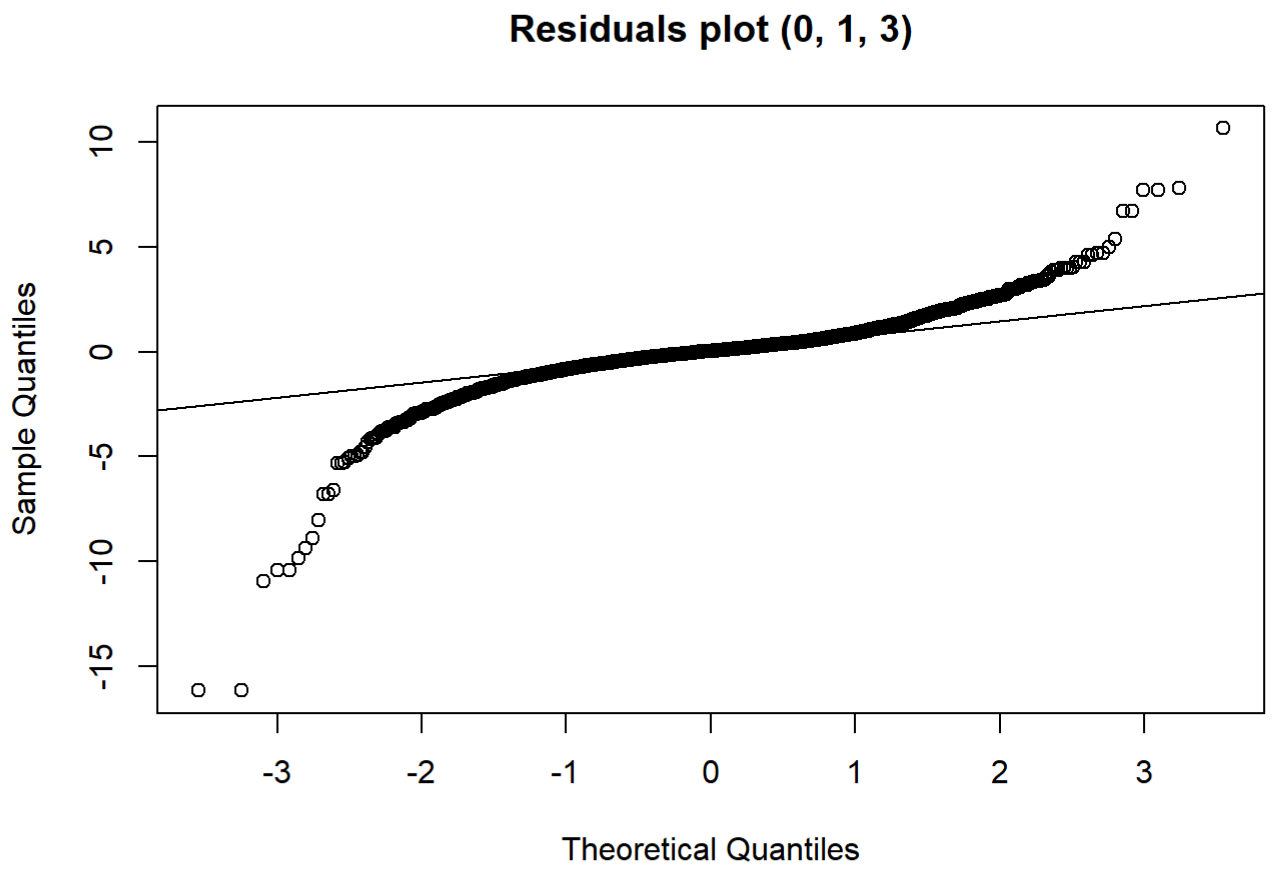
```
## AIC: 7473.692
## BIC: 7507.839
```

```
my_df <- plot_arima_residuals(arima013, my_df)
```


Residual Plot (0 , 1, 3)
Residuals
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021
Year

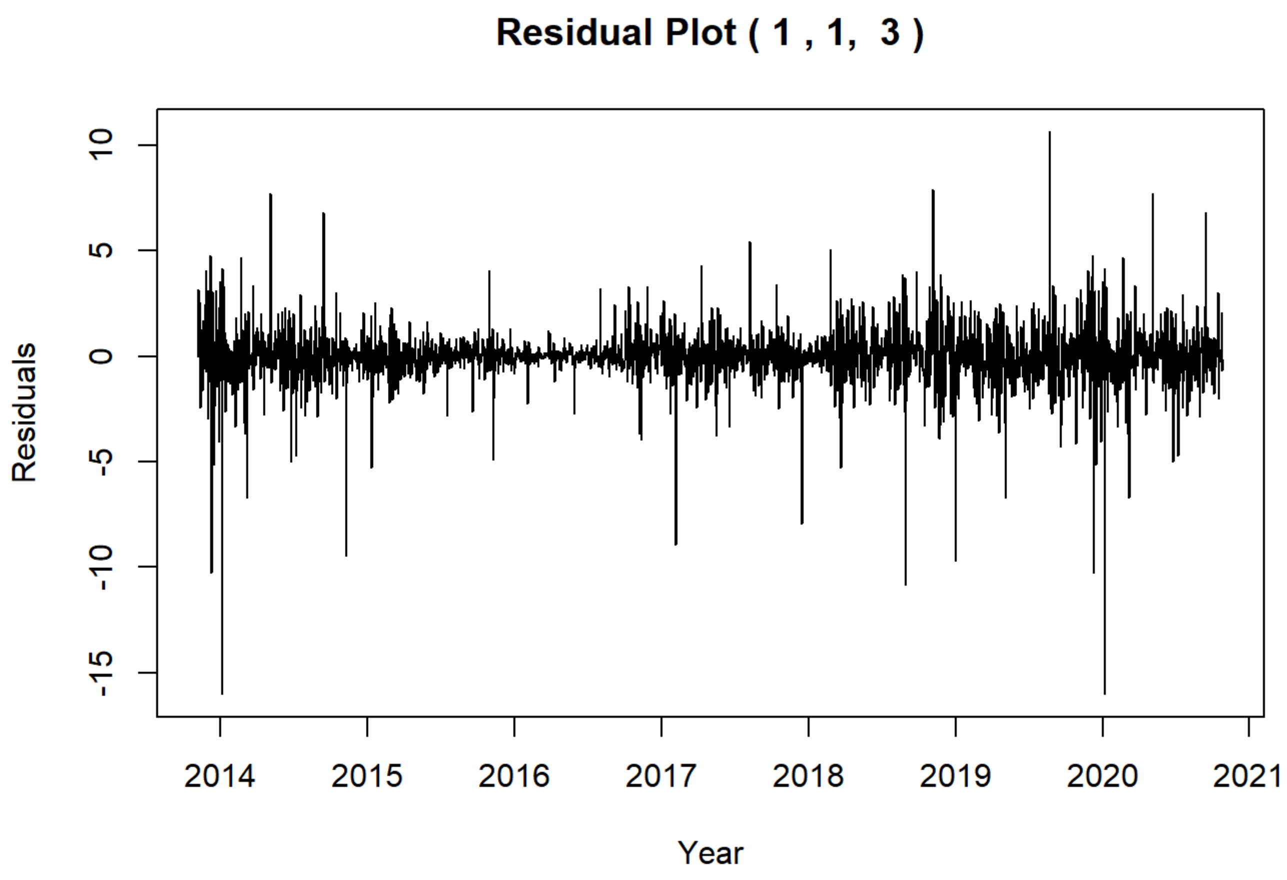

ACF of Residual (0 , 1, 3)
ACF
0.10
0.05
0.00
-0.05
0
50
100
150
Lag

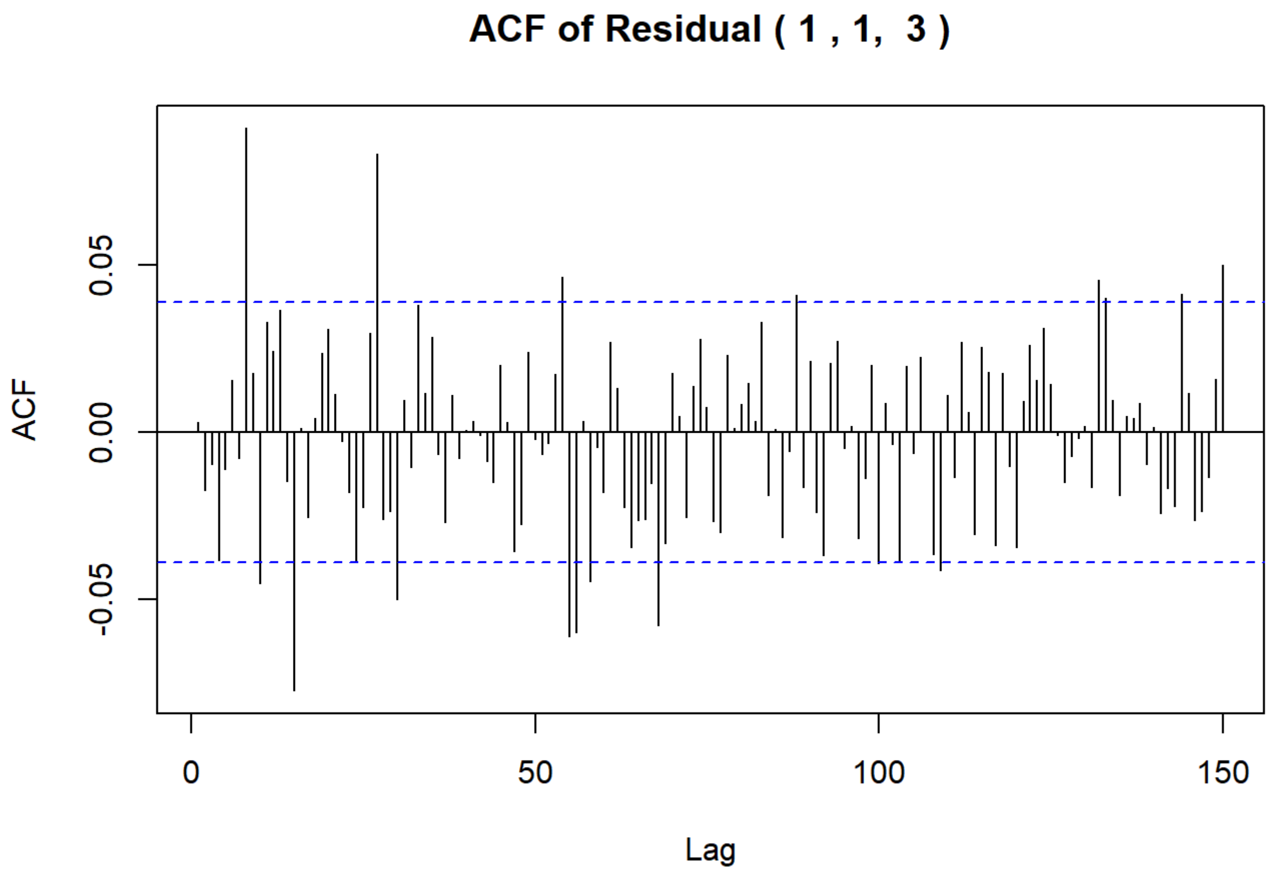


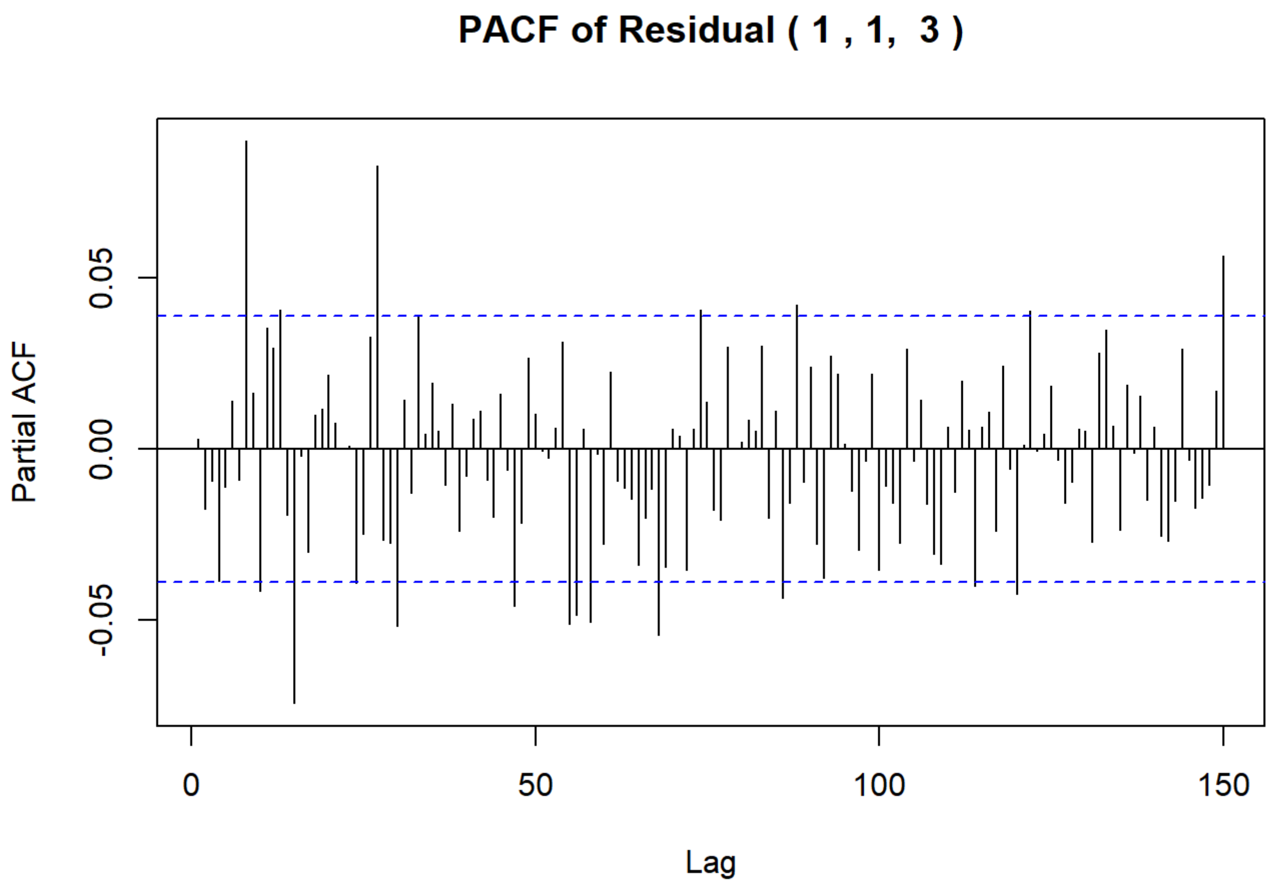


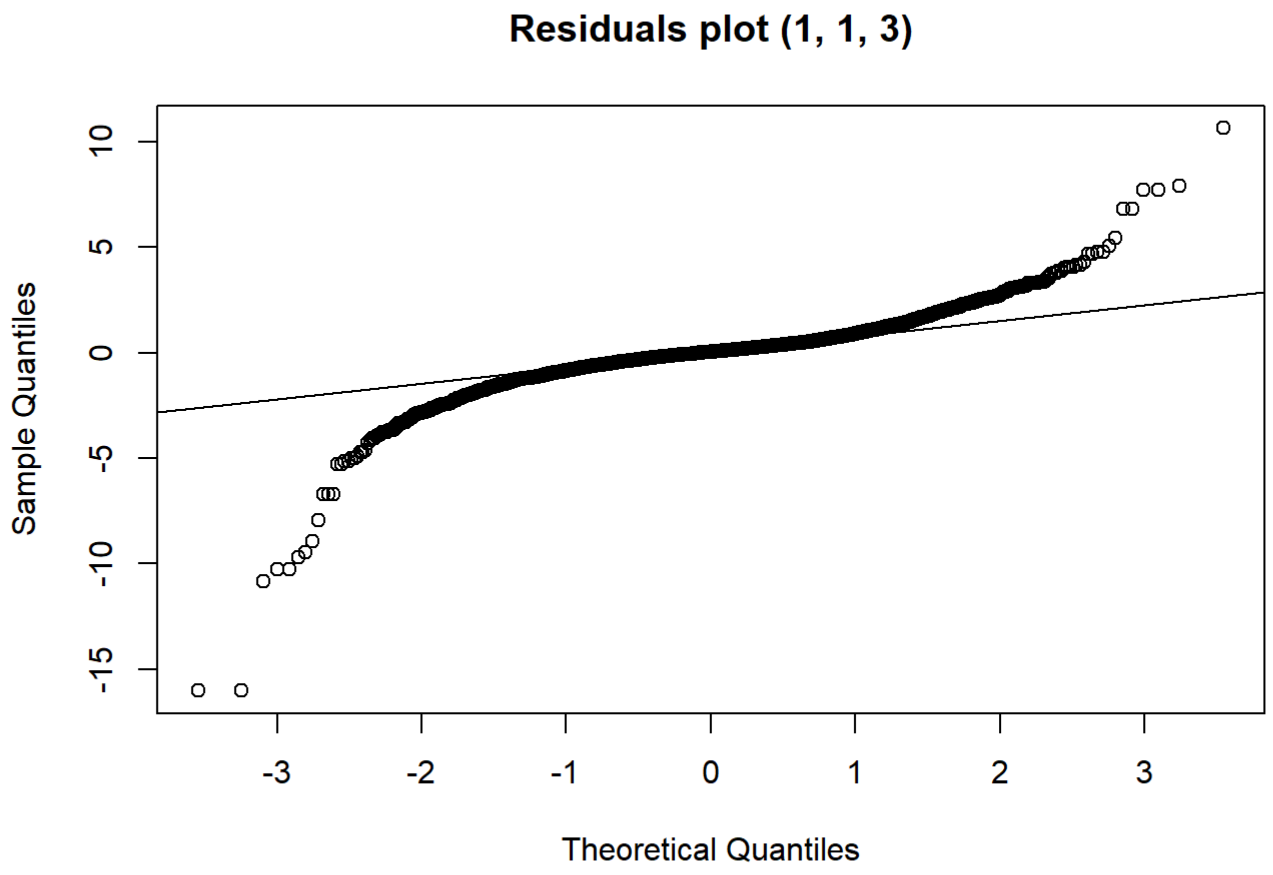
```
## AIC: 7475.967
## BIC: 7498.731
```

```
my_df <- plot_arima_residuals(arima113, my_df)
```


Residual Plot (1 , 1, 3)
Residuals
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021
Year

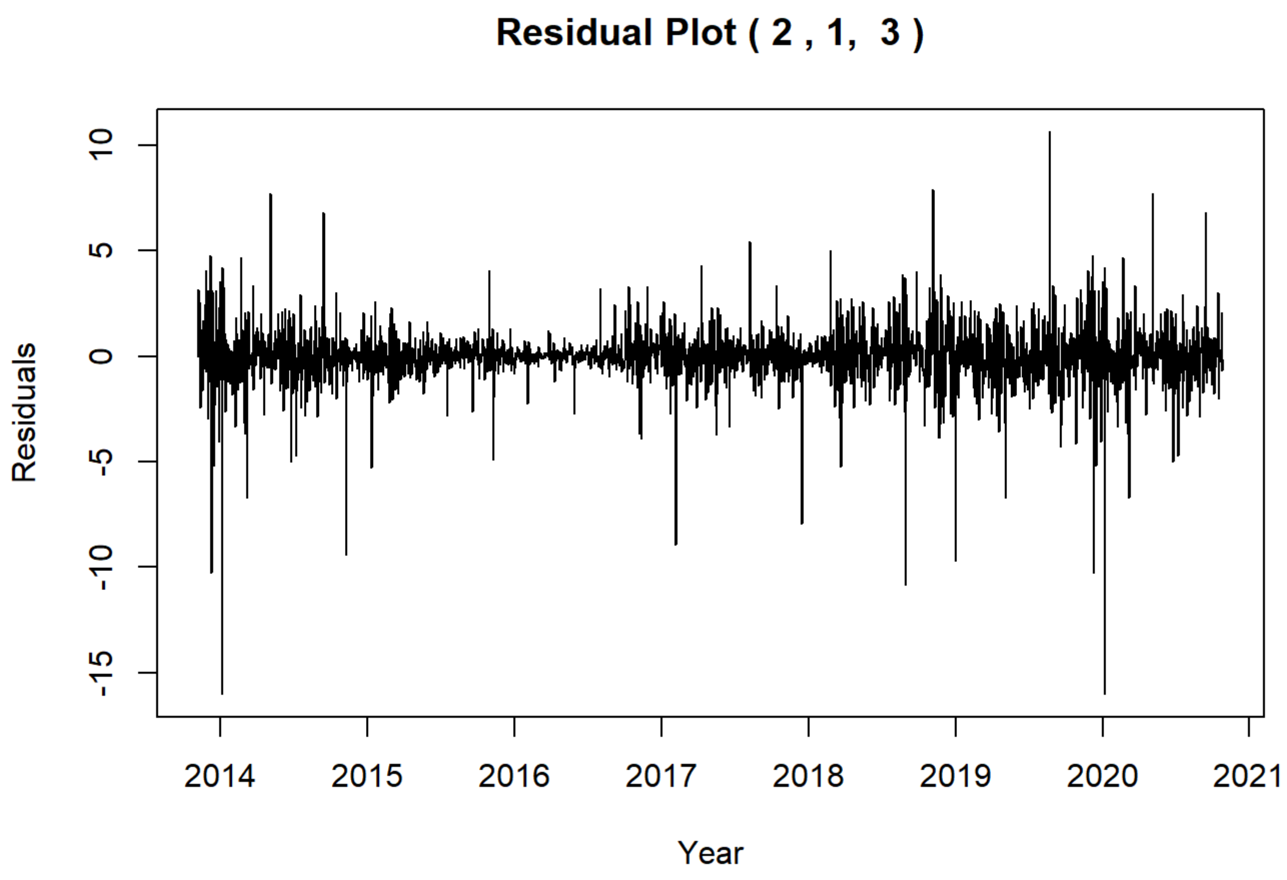

ACF of Residual (1 , 1, 3)
ACF
0.05
0.00
-0.05
0
50
100
150
Lag

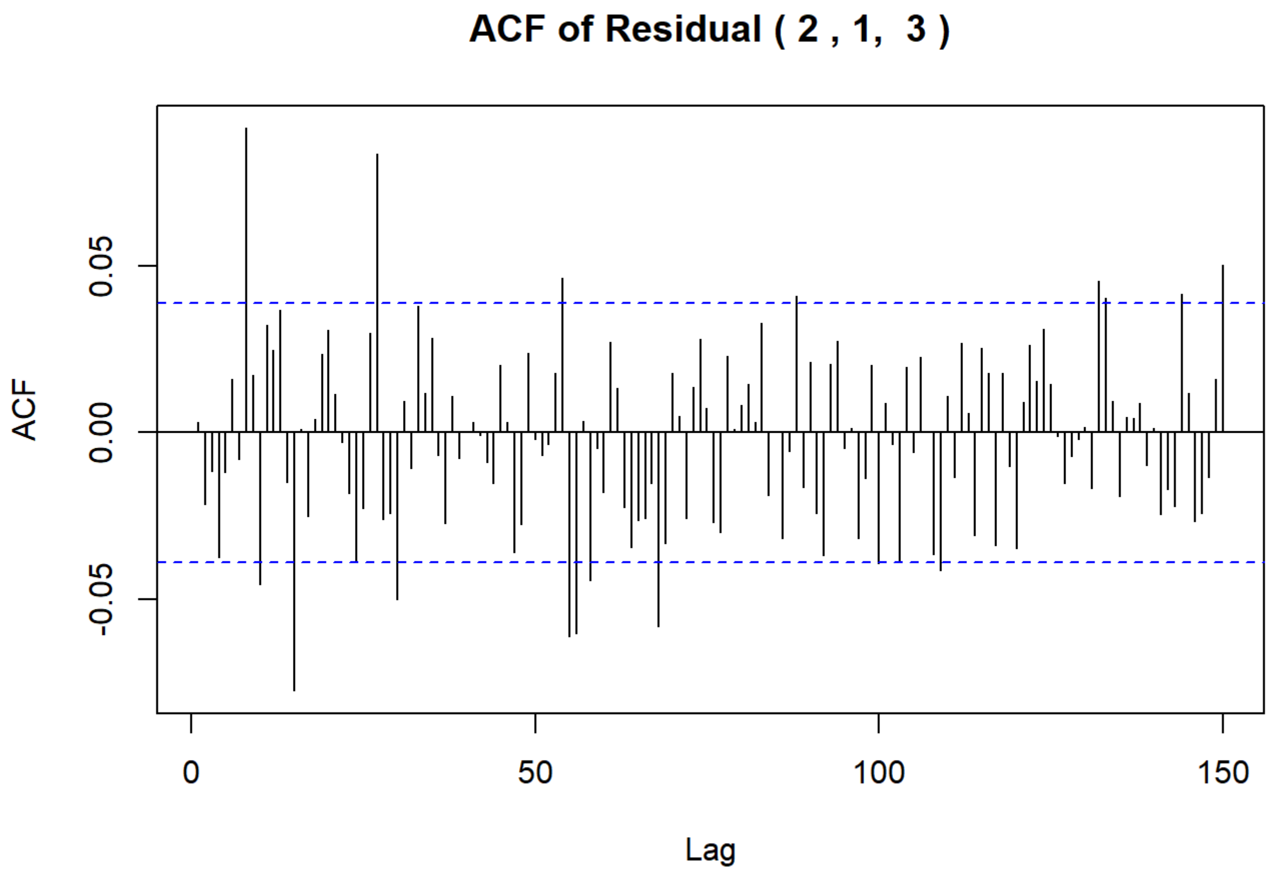


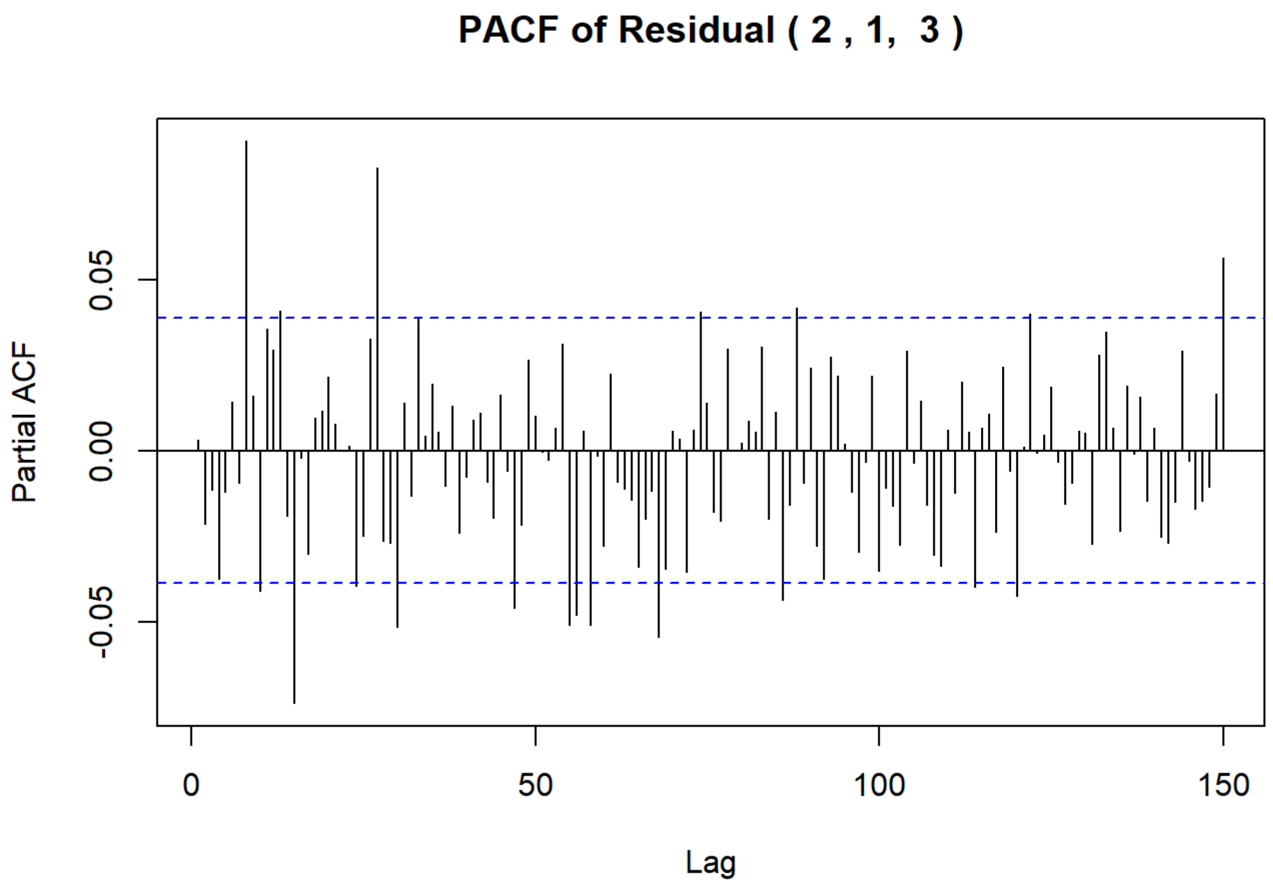


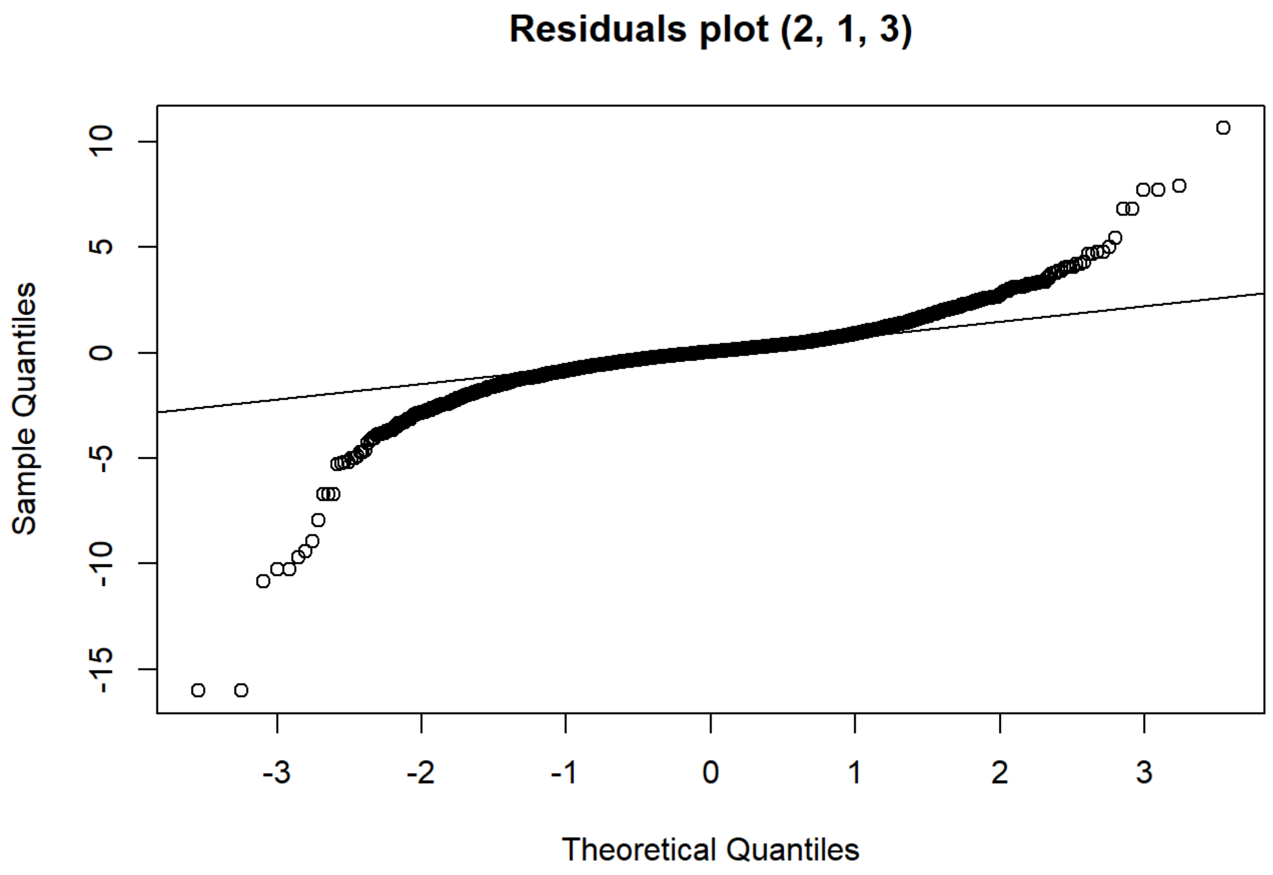
```
## AIC: 7471.742
## BIC: 7500.198
```

```
my_df <- plot_arima_residuals(arima213, my_df)
```


Residual Plot (2 , 1, 3)
Residuals
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021
Year

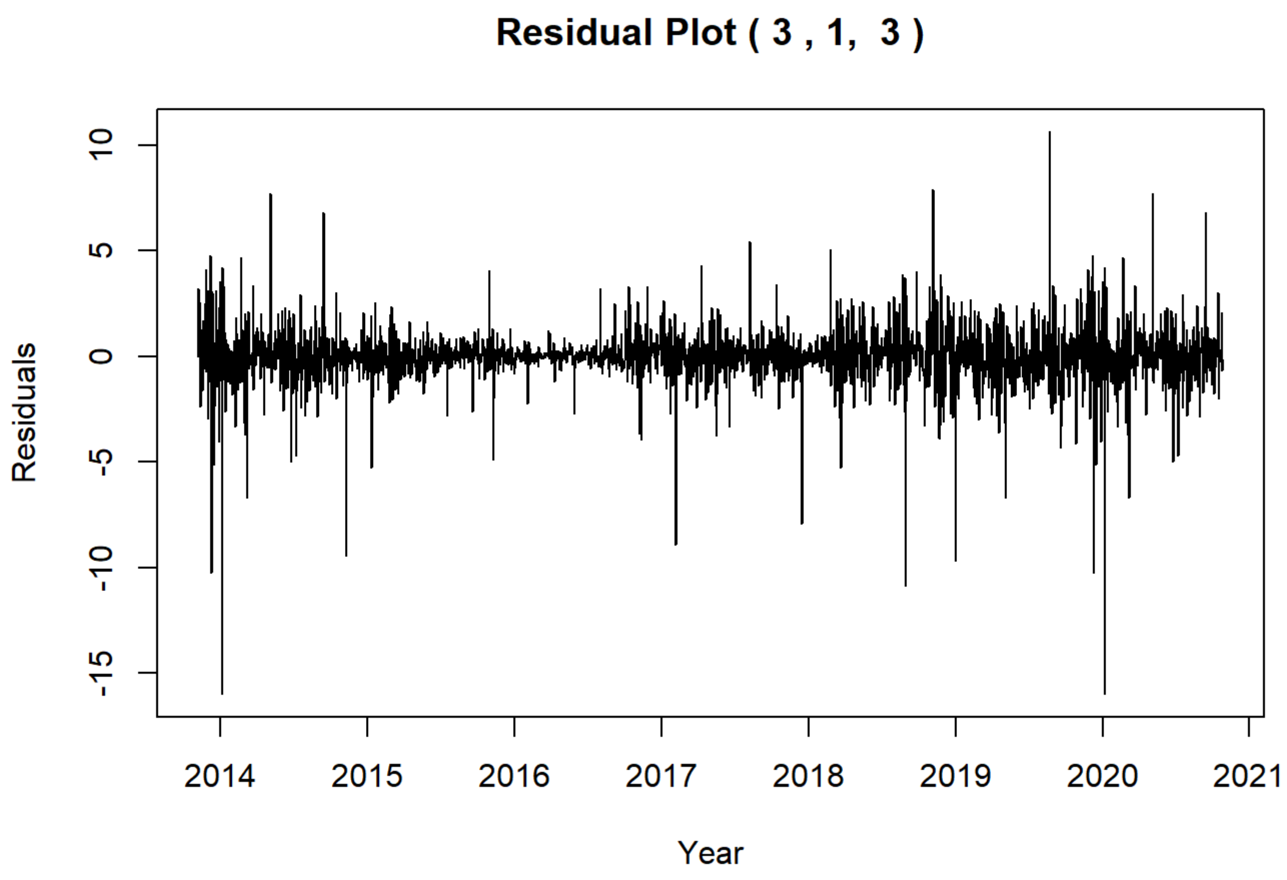

ACF of Residual (2 , 1, 3)
ACF
0.05
0.00
-0.05
0
50
100
150
Lag

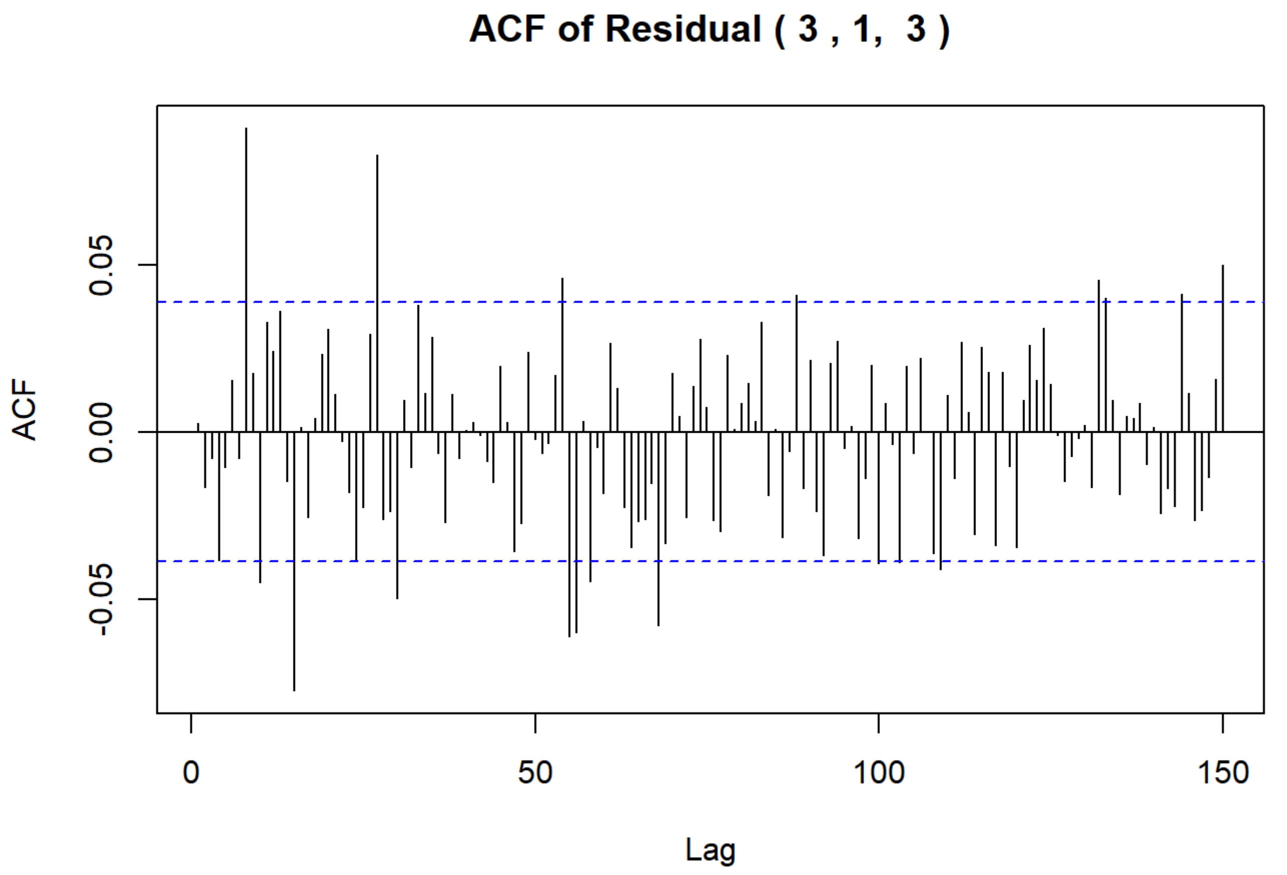


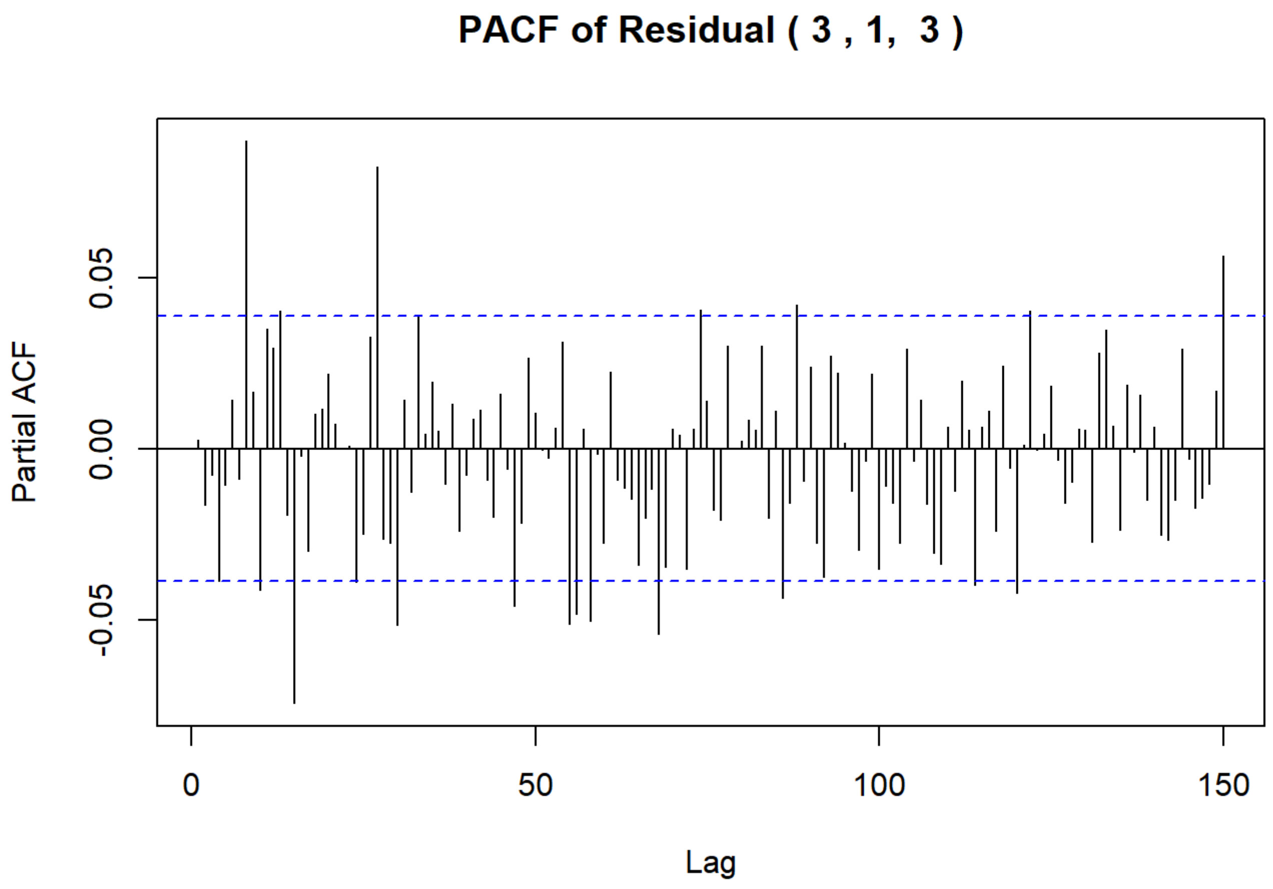


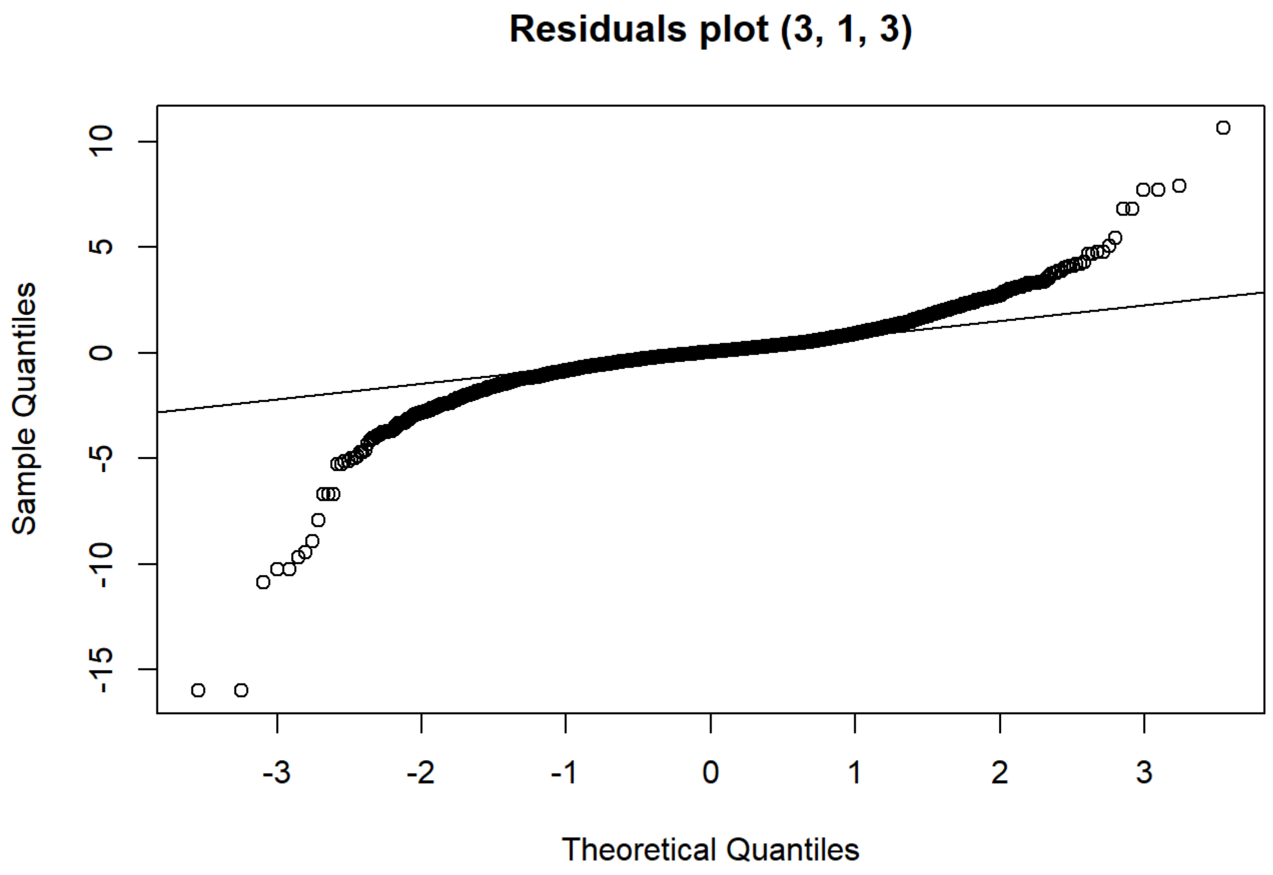
```
## AIC: 7473.721
## BIC: 7507.868
```

```
my_df <- plot_arima_residuals(arima313, my_df)
```


Residual Plot (3 , 1, 3)
Residuals
10
5
0
-5
-10
-15
2014
2015
2016
2017
2018
2019
2020
2021
Year


ACF of Residual (3 , 1, 3)
ACF
0.05
0.00
-0.05
0
50
100
150
Lag





```
## AIC: 7475.693
## BIC: 7515.532
```

```
my_df
```

model	AIC	BIC	Shapiro	Ljung
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
ARIMA (0, 1, 1)	7475.113	7486.496	0	0.002
ARIMA (1, 1, 0)	8297.696	8309.079	0	0.000
ARIMA (1, 1, 1)	7477.111	7494.184	0	0.002
ARIMA (2, 1, 0)	8106.817	8123.890	0	0.000
ARIMA (2, 1, 1)	7476.025	7498.789	0	0.006
ARIMA (0, 1, 2)	7477.111	7494.185	0	0.002
ARIMA (1, 1, 2)	7472.119	7494.884	0	0.022
ARIMA (2, 1, 2)	7471.717	7500.173	0	0.044
ARIMA (3, 1, 0)	7966.435	7989.200	0	0.000
ARIMA (3, 1, 1)	7474.882	7503.338	0	0.022

1-10 of 15 rows Previous **1** 2 Next

```
BIC_values <- c(BIC(arima011), BIC(arima110), BIC(arima111),
                BIC(arima210), BIC(arima211), BIC(arima012), BIC(arima112), BIC(arima212),
                BIC(arima310), BIC(arima311), BIC(arima312), BIC(arima013), BIC(arima113), BI
C(arima213), BIC(arima313))
BIC_values
```

```
##  [1] 7486.496 8309.079 7494.184 8123.890 7498.789 7494.185 7494.884 7500.173
##  [9] 7989.200 7503.338 7507.839 7498.731 7500.198 7507.868 7515.532
```

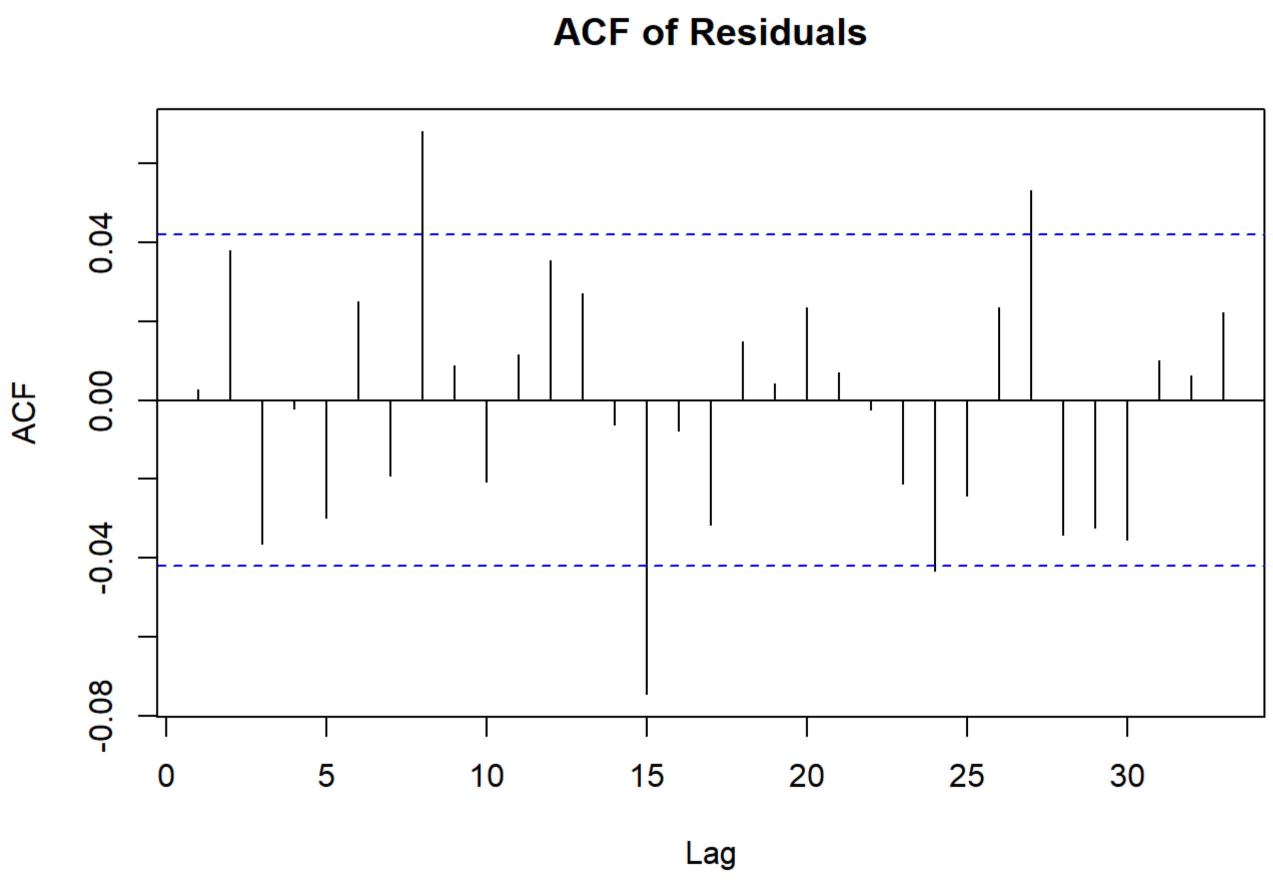
```
min_BIC <- min(BIC_values)
min_BIC
```

```
## [1] 7486.496
```

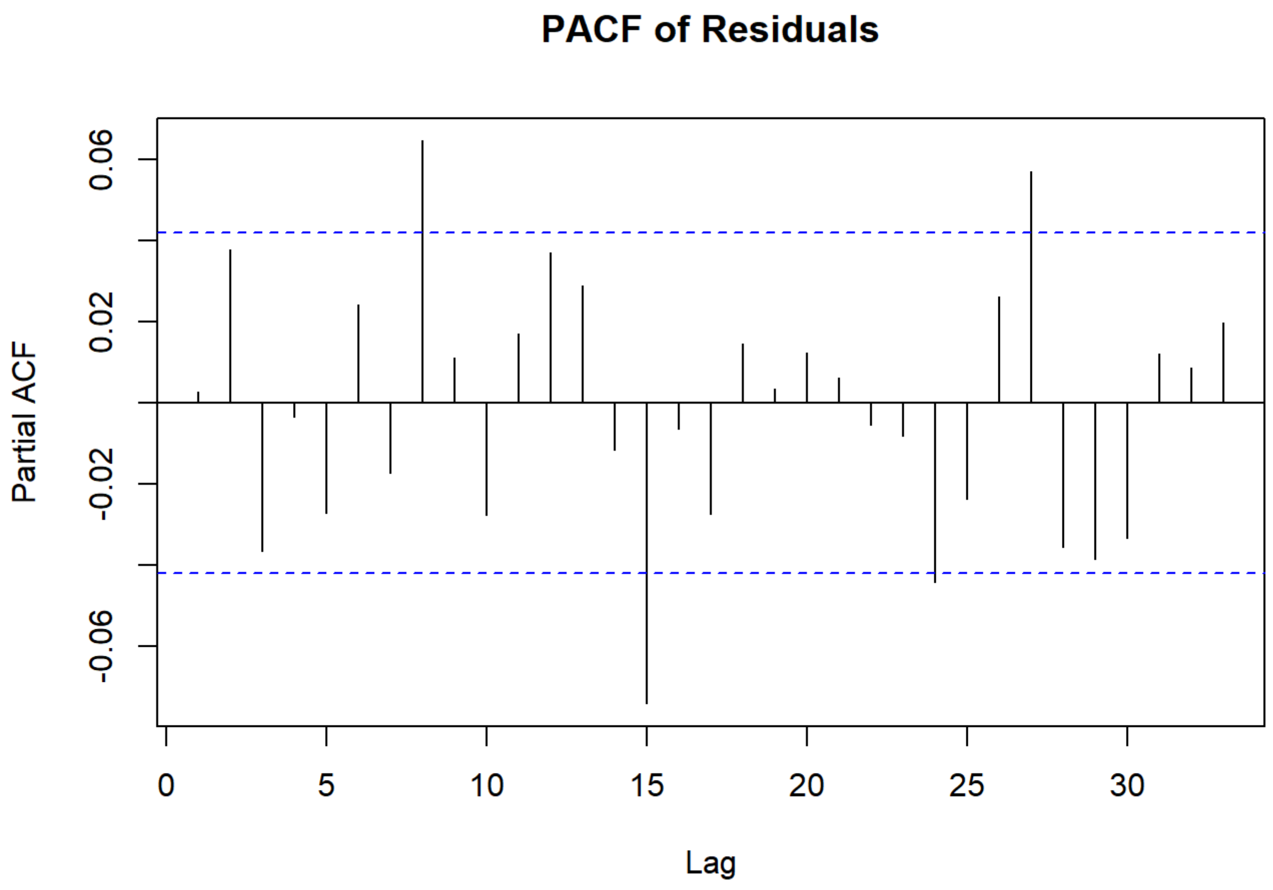
```
min_BIC_index <- which.min(BIC_values)
min_BIC_index
```

```
## [1] 1
```

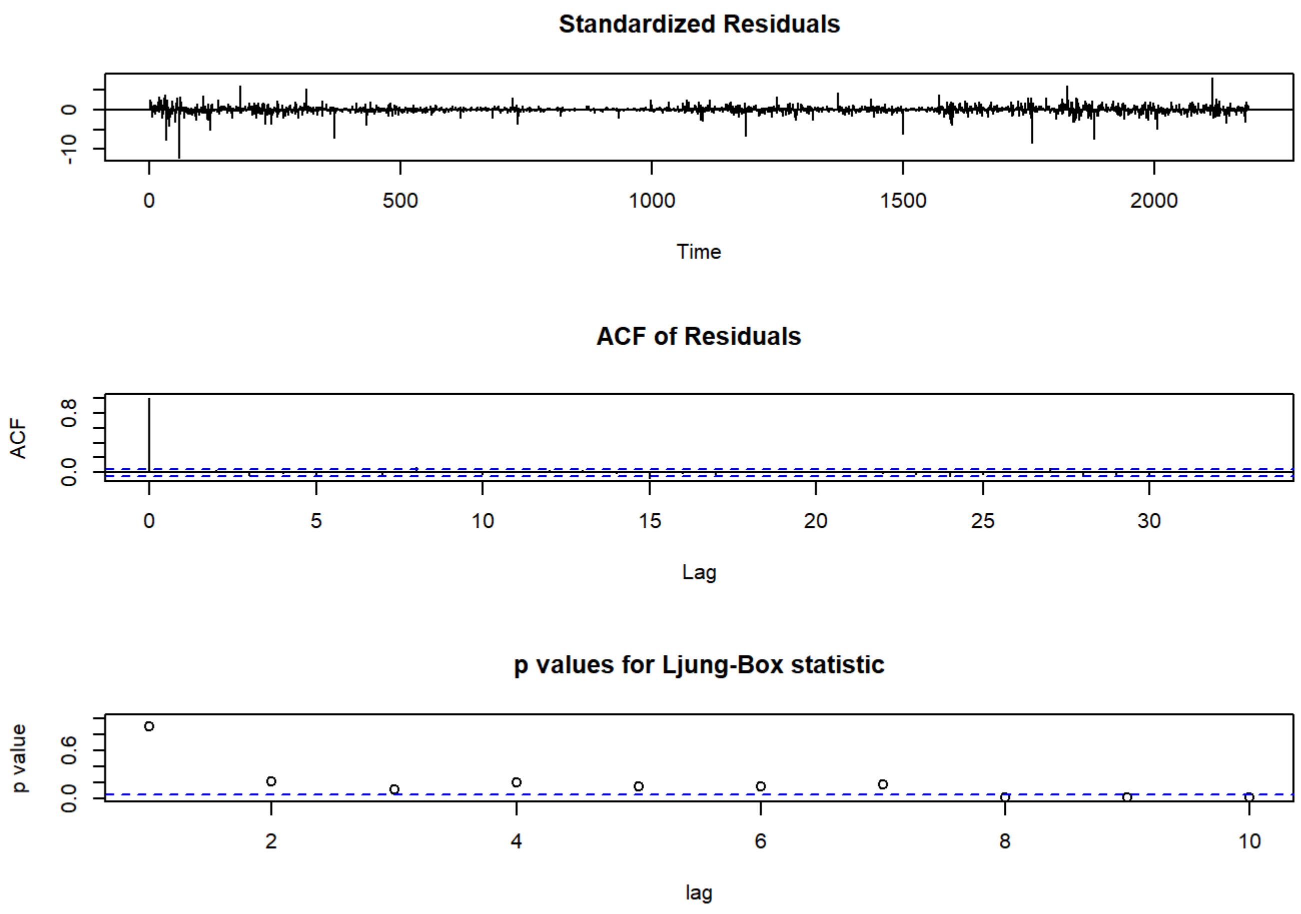
```
# Obtain the residuals from the ARIMA model
residuals <- residuals(arima011)

# Plot the ACF and PACF of residuals
acf(residuals, main = 'ACF of Residuals')
```



```
pacf(residuals, main = 'PACF of Residuals')
```



```
# Ljung-Box test for residual autocorrelation
tsdiag(arima011)
```



```
predicted_values <- predict(arima011, n.ahead = 60)
print(predicted_values)
```

```
## $pred
## Time Series:
## Start = 2191
## End = 2250
## Frequency = 1
##  [1] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
##  [6] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
## [11] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
## [16] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
## [21] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
## [26] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
## [31] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
## [36] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
## [41] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
## [46] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
## [51] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
## [56] -0.002310503 -0.002310503 -0.002310503 -0.002310503 -0.002310503
##
## $se
## Time Series:
## Start = 2191
## End = 2250
## Frequency = 1
##  [1] 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153
##  [9] 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153
## [17] 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153
## [25] 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153
## [33] 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153
## [41] 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153
## [49] 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153 1.331153
## [57] 1.331153 1.331153 1.331153 1.331153
```

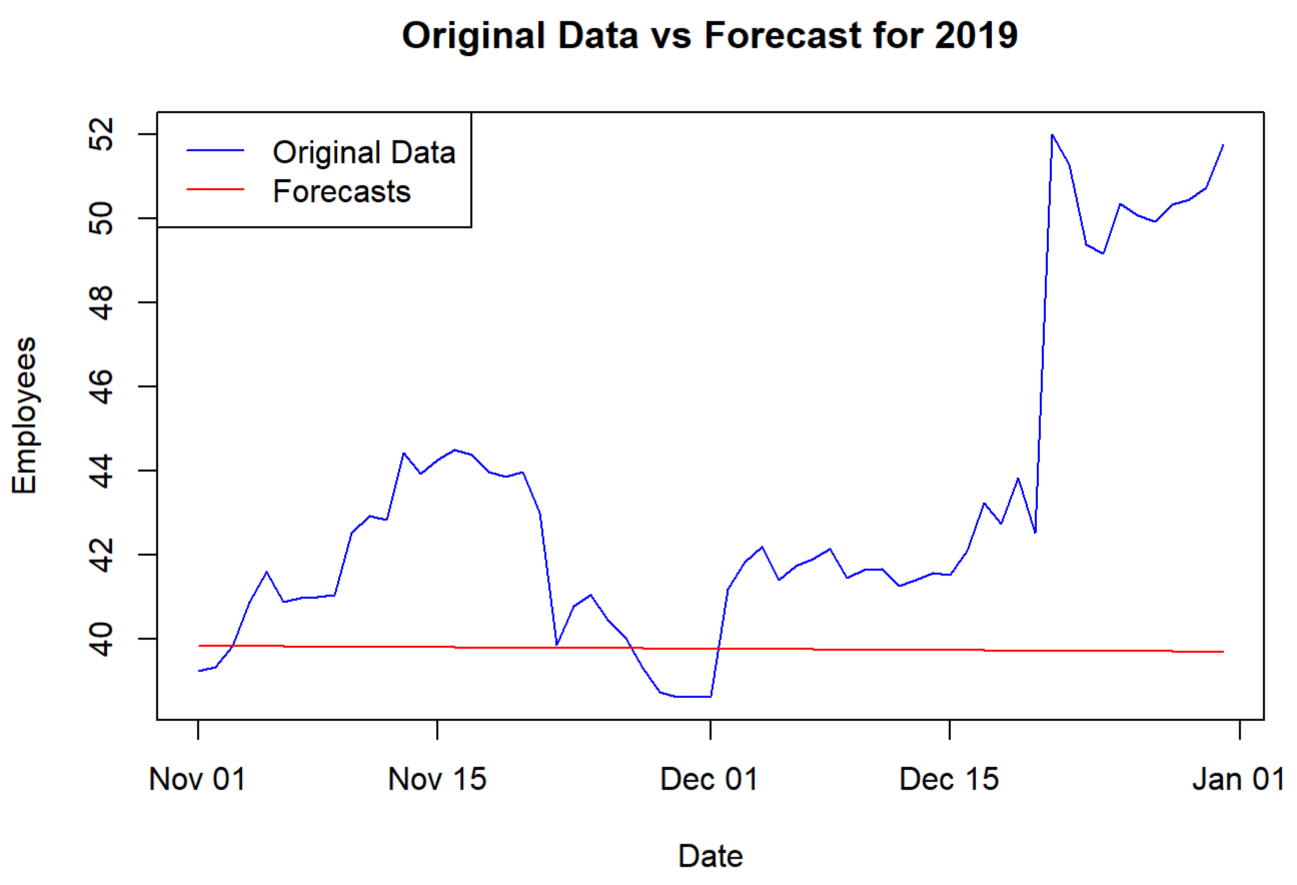
```
test_data$Close
```

```
##  [1] 39.24 39.34 39.85 40.89 41.61 40.89 40.98 41.00 41.06 42.52 42.94 42.83
## [13] 44.43 43.94 44.26 44.50 44.40 43.99 43.86 43.99 43.01 39.86 40.79 41.05
## [25] 40.46 40.04 39.32 38.75 38.62 38.63 38.65 41.20 41.85 42.19 41.41 41.74
## [37] 41.90 42.14 41.45 41.66 41.68 41.27 41.40 41.58 41.52 42.09 43.25 42.74
## [49] 43.84 42.54 52.00 51.30 49.39 49.18 50.36 50.07 49.94 50.34 50.45 50.74
## [61] 51.78
```

```
original_forecasts <- diffinv(predicted_values$pred, lag = 1, differences = 1, xi = 39.84)
original_forecasts <- original_forecasts[original_forecasts != 0]
#original_forecasts <- original_forecasts[-1]
original_forecasts
```

```
##  [1] 39.84000 39.83769 39.83538 39.83307 39.83076 39.82845 39.82614 39.82383
##  [9] 39.82152 39.81921 39.81689 39.81458 39.81227 39.80996 39.80765 39.80534
## [17] 39.80303 39.80072 39.79841 39.79610 39.79379 39.79148 39.78917 39.78686
## [25] 39.78455 39.78224 39.77993 39.77762 39.77531 39.77300 39.77068 39.76837
## [33] 39.76606 39.76375 39.76144 39.75913 39.75682 39.75451 39.75220 39.74989
## [41] 39.74758 39.74527 39.74296 39.74065 39.73834 39.73603 39.73372 39.73141
## [49] 39.72910 39.72679 39.72447 39.72216 39.71985 39.71754 39.71523 39.71292
## [57] 39.71061 39.70830 39.70599 39.70368 39.70137
```

```
plot(test_data$Date, test_data$Close, main = 'Original Data vs Forecast for 2019', type =
'l', col = 'blue', xlab = 'Date', ylab = 'Employees')

# Adding forecasted values to the plot (if 'original_forecasts' is aligned with 'test_data$Da
te')
lines(test_data$Date, original_forecasts, col = 'red', type = 'l')

# Adding legend
legend('topleft', legend = c('Original Data', 'Forecasts'), col = c('blue', 'red'), lty = 1)
```

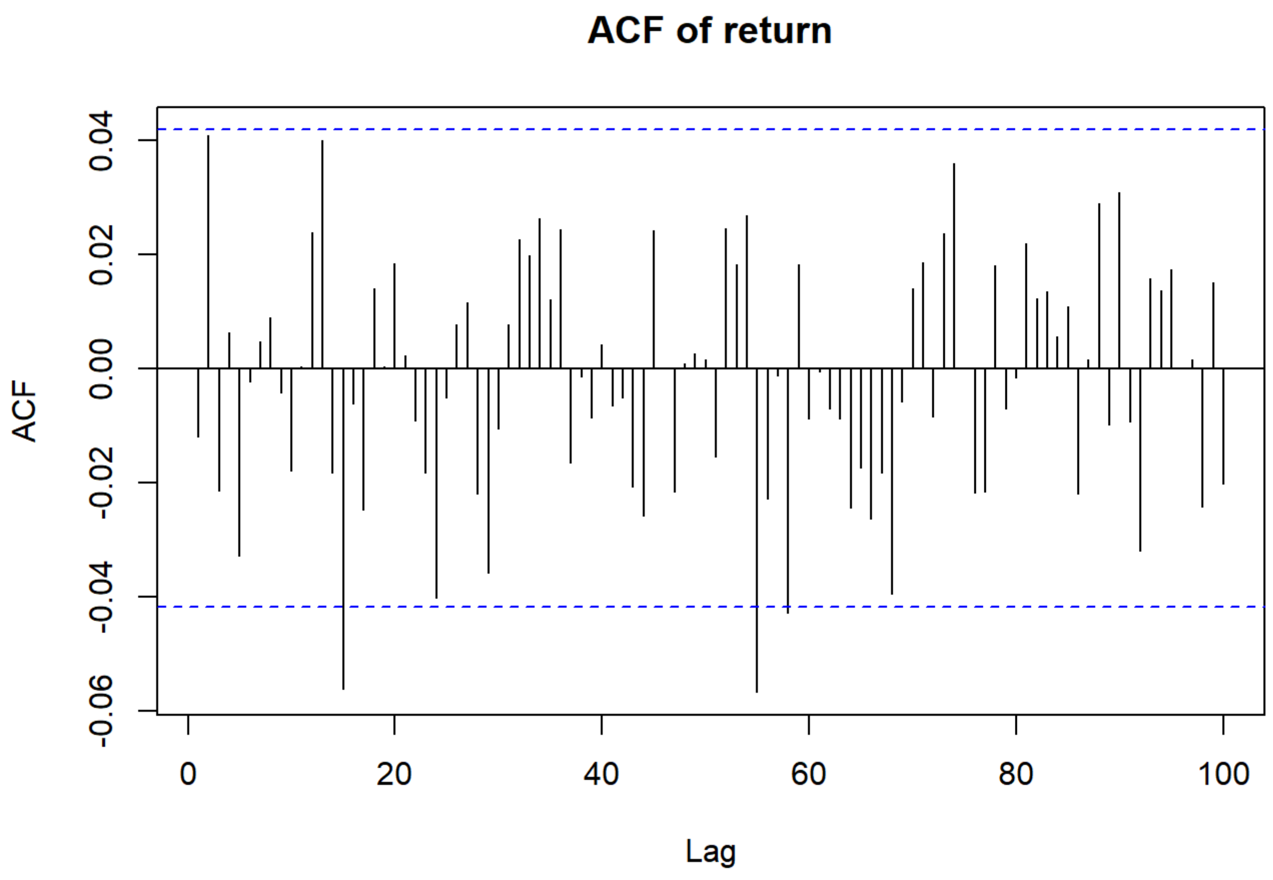


```
print("From the residual analyis, we can see that ARIMA model could not capture any dependenc
ies")
```

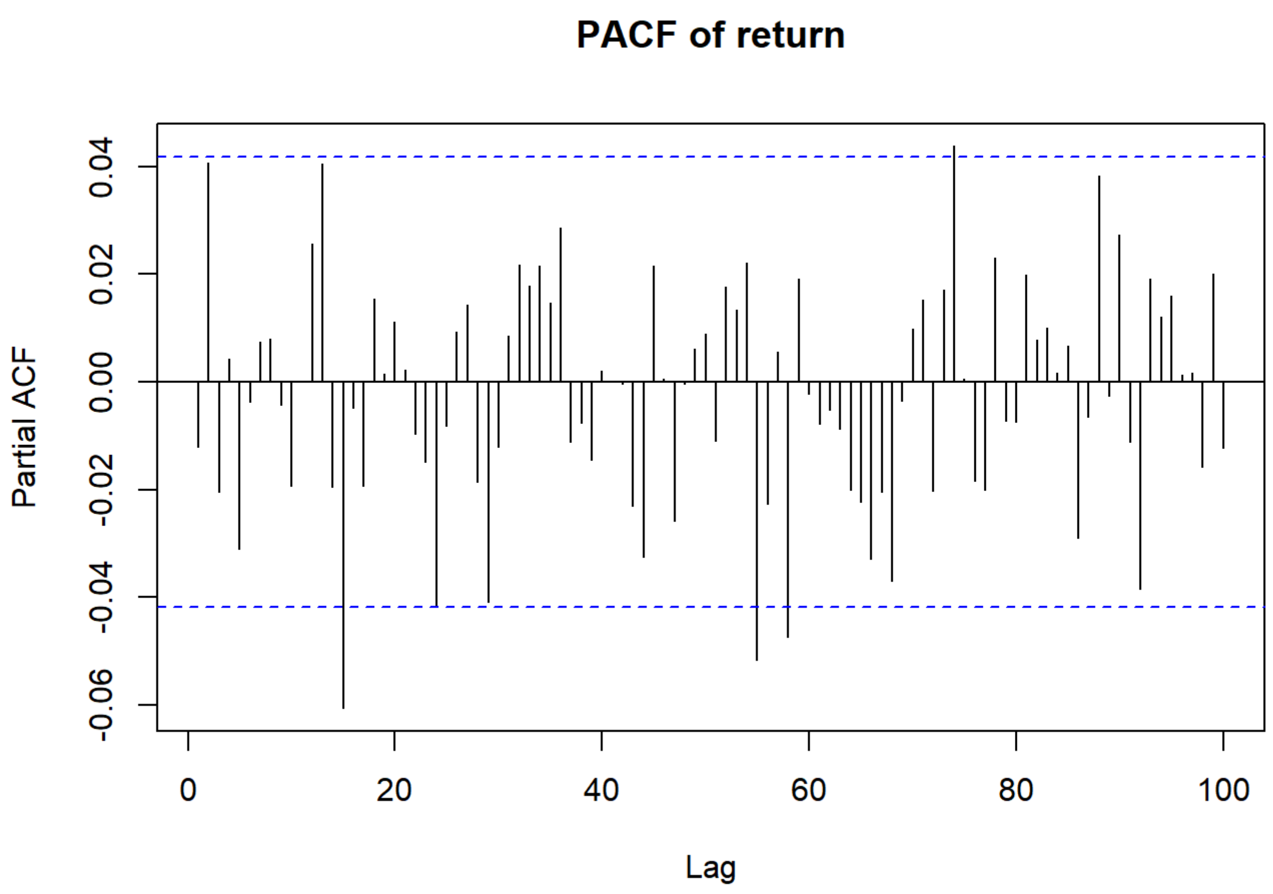
```
## [1] "From the residual analyis, we can see that ARIMA model could not capture any dependen
cies"
```

GARCH MODEL - NORMAL return

```r
return = diff(log(train_data$Close))

#Try with absolute and square of the returns
acf(return, lag.max = 100, main='ACF of return')
```



```r
pacf(return, lag.max = 100, main='PACF of return')
```



```
eacf(return)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o 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  x  o
## 2 x x o o o o o o o o o  o  o  o
## 3 x x x 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 x x x x o o o o o o  o  o  o
## 6 x x x o x x o o o o o  o  o  o
## 7 x x x x x x o o o o o  o  o  o
```

```
garch(x=return, grad='analytical', trace=FALSE)
```

```
##
## Call:
## garch(x = return, grad = "analytical", trace = FALSE)
##
## Coefficient(s):
##        a0         a1         b1
## 0.0007044  0.1560470  0.2636142
```

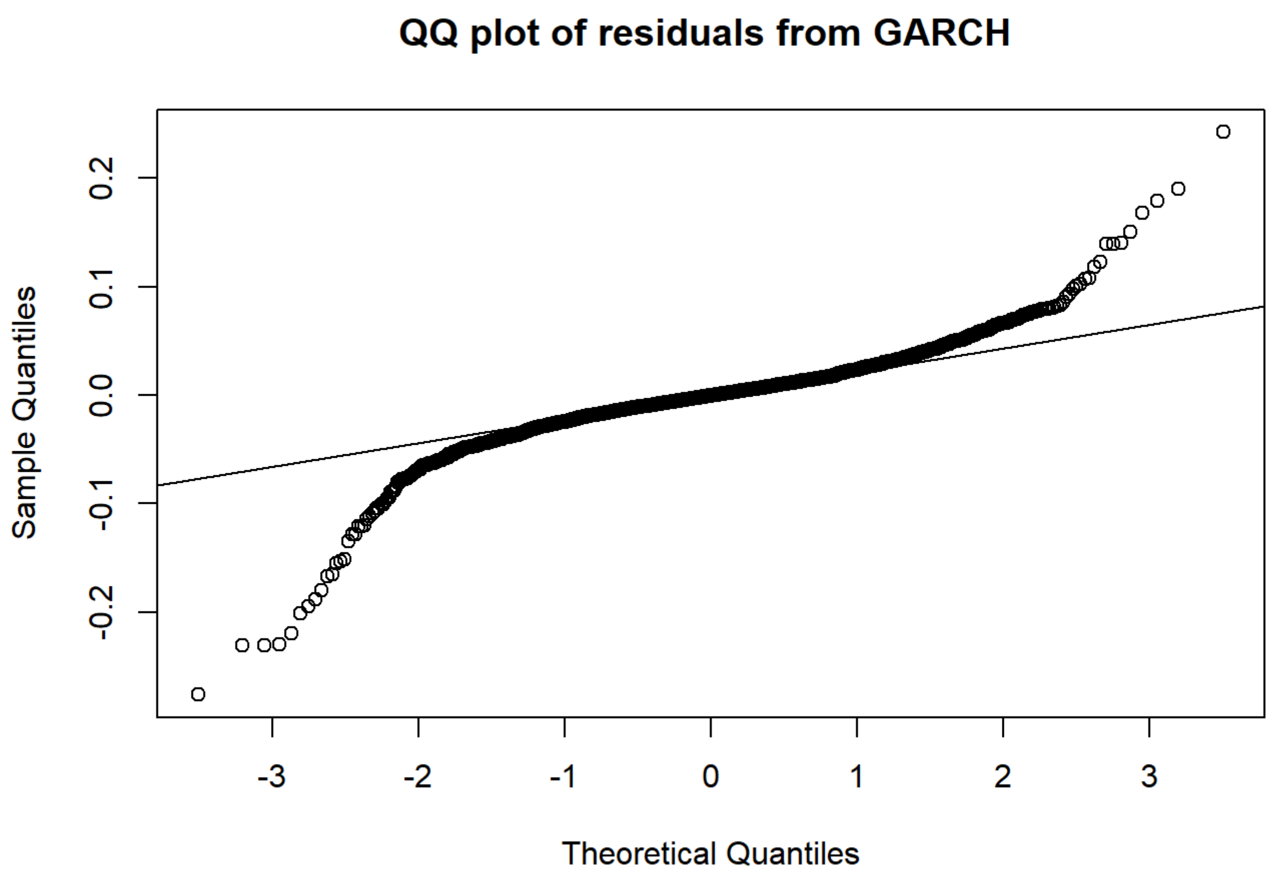
```
garch_spec <- ugarchspec(variance.model = list(garchOrder = c(2,2)),
                         mean.model = list(armaOrder = c(3,4)),
                         distribution.model = 'sstd')

garch_fit <- ugarchfit(spec = garch_spec, data = return, solver = 'hybrid')
summary(garch_fit)
```

```
##    Length     Class      Mode
##         1 uGARCHfit        S4
```

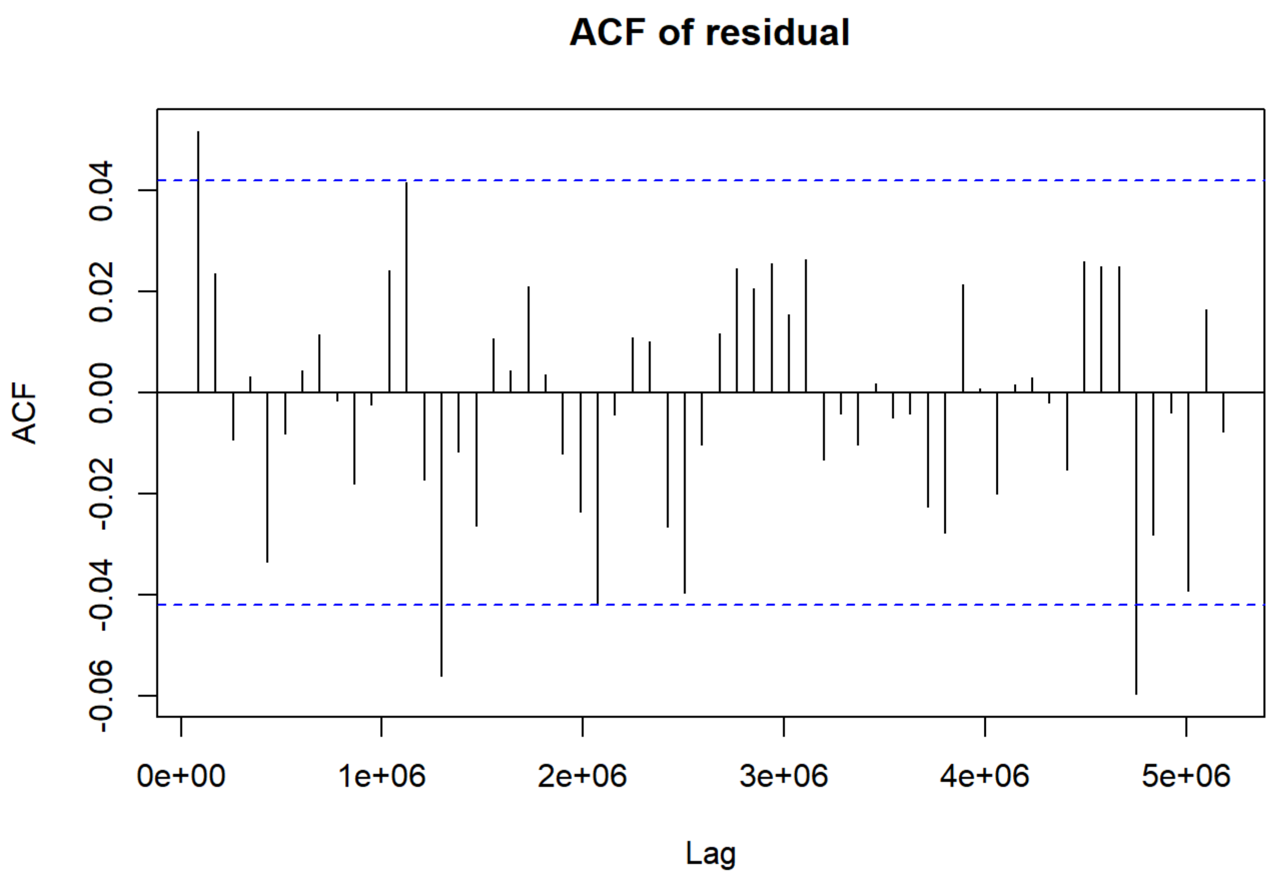
```
garch_fit
```

```
## 
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
## 
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,2)
## Mean Model   : ARFIMA(3,0,4)
## Distribution : sstd 
## 
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error     t value Pr(>|t|)
## mu      0.000928    0.000150  6.1695e+00 0.000000
## ar1     0.459871    0.000043  1.0648e+04 0.000000
## ar2    -0.698955    0.000064 -1.0967e+04 0.000000
## ar3    -0.362435    0.000037 -9.8981e+03 0.000000
## ma1    -0.525979    0.000050 -1.0430e+04 0.000000
## ma2     0.756458    0.000060  1.2643e+04 0.000000
## ma3     0.303138    0.000027  1.1223e+04 0.000000
## ma4    -0.003455    0.000012 -2.9167e+02 0.000000
## omega   0.000180    0.000026  6.9323e+00 0.000000
## alpha1  0.161748    0.042462  3.8093e+00 0.000139
## alpha2  0.000411    0.035525  1.1572e-02 0.990767
## beta1   0.663419    0.303965  2.1825e+00 0.029069
## beta2   0.033976    0.174704  1.9448e-01 0.845804
## skew    1.005468    0.022372  4.4942e+01 0.000000
## shape   3.030458    0.102257  2.9636e+01 0.000000
## 
## Robust Standard Errors:
##         Estimate  Std. Error     t value Pr(>|t|)
## mu      0.000928    0.000700  1.3258e+00 0.184922
## ar1     0.459871    0.000581  7.9113e+02 0.000000
## ar2    -0.698955    0.000086 -8.1681e+03 0.000000
## ar3    -0.362435    0.000389 -9.3169e+02 0.000000
## ma1    -0.525979    0.000799 -6.5792e+02 0.000000
## ma2     0.756458    0.000084  8.9635e+03 0.000000
## ma3     0.303138    0.000459  6.6059e+02 0.000000
## ma4    -0.003455    0.000060 -5.7532e+01 0.000000
## omega   0.000180    0.000238  7.5893e-01 0.447894
## alpha1  0.161748    0.073701  2.1947e+00 0.028189
## alpha2  0.000411    0.189655  2.1680e-03 0.998271
## beta1   0.663419    0.971826  6.8265e-01 0.494826
## beta2   0.033976    0.681867  4.9828e-02 0.960260
## skew    1.005468    0.027975  3.5941e+01 0.000000
## shape   3.030458    0.331436  9.1434e+00 0.000000
## 
## LogLikelihood : 4698.724 
## 
## Information Criteria
## ------------------------------------
##                     
## Akaike       -4.2774
## Bayes        -4.2384
## Shibata      -4.2775
## Hannan-Quinn -4.2631
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                       3.116 0.07751
## Lag[2*(p+q)+(p+q)-1][20]     9.639 0.93039
## Lag[4*(p+q)+(p+q)-1][34]    16.669 0.58395
## d.o.f=7
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.5309  0.4662
## Lag[2*(p+q)+(p+q)-1][11]    4.0526  0.7251
## Lag[4*(p+q)+(p+q)-1][19]    5.7888  0.8899
## d.o.f=4
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[5]    0.3424 0.500 2.000  0.5585
## ARCH Lag[7]    1.4754 1.473 1.746  0.6291
## ARCH Lag[9]    1.9652 2.402 1.619  0.7652
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  7.8018
## Individual Statistics:
## mu     0.02585
## ar1    0.02160
## ar2    0.04114
## ar3    0.05854
## ma1    0.02712
## ma2    0.03273
## ma3    0.06148
## ma4    0.01997
## omega  1.16325
## alpha1 0.34374
## alpha2 0.16317
## beta1  0.59015
## beta2  0.58196
## skew   0.06439
## shape  0.37051
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          3.26 3.54 4.07
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value    prob sig
## Sign Bias            1.848 0.06477   *
## Negative Sign Bias   1.546 0.12220
## Positive Sign Bias   0.695 0.48712
## Joint Effect         4.117 0.24909
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     15.83       0.6688
## 2    30     20.44       0.8788
## 3    40     36.41       0.5886
## 4    50     39.86       0.8210
##
##
## Elapsed time : 2.121947
```

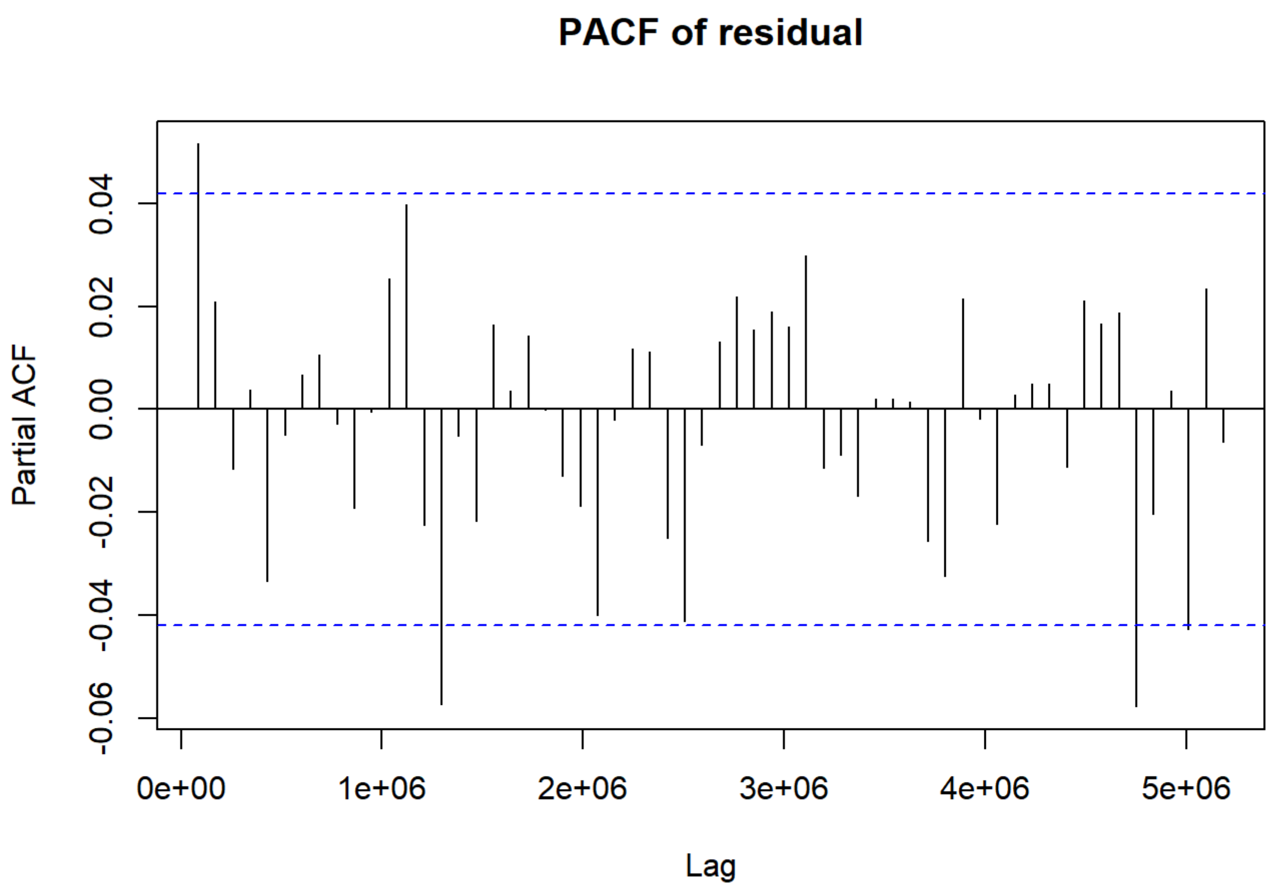
```
resid_norm = residuals(garch_fit)
qqnorm(resid_norm, main='QQ plot of residuals from GARCH ')
qqline(resid_norm)
```



```
acf(resid_norm, lag.max = 60, main='ACF of residual')
```



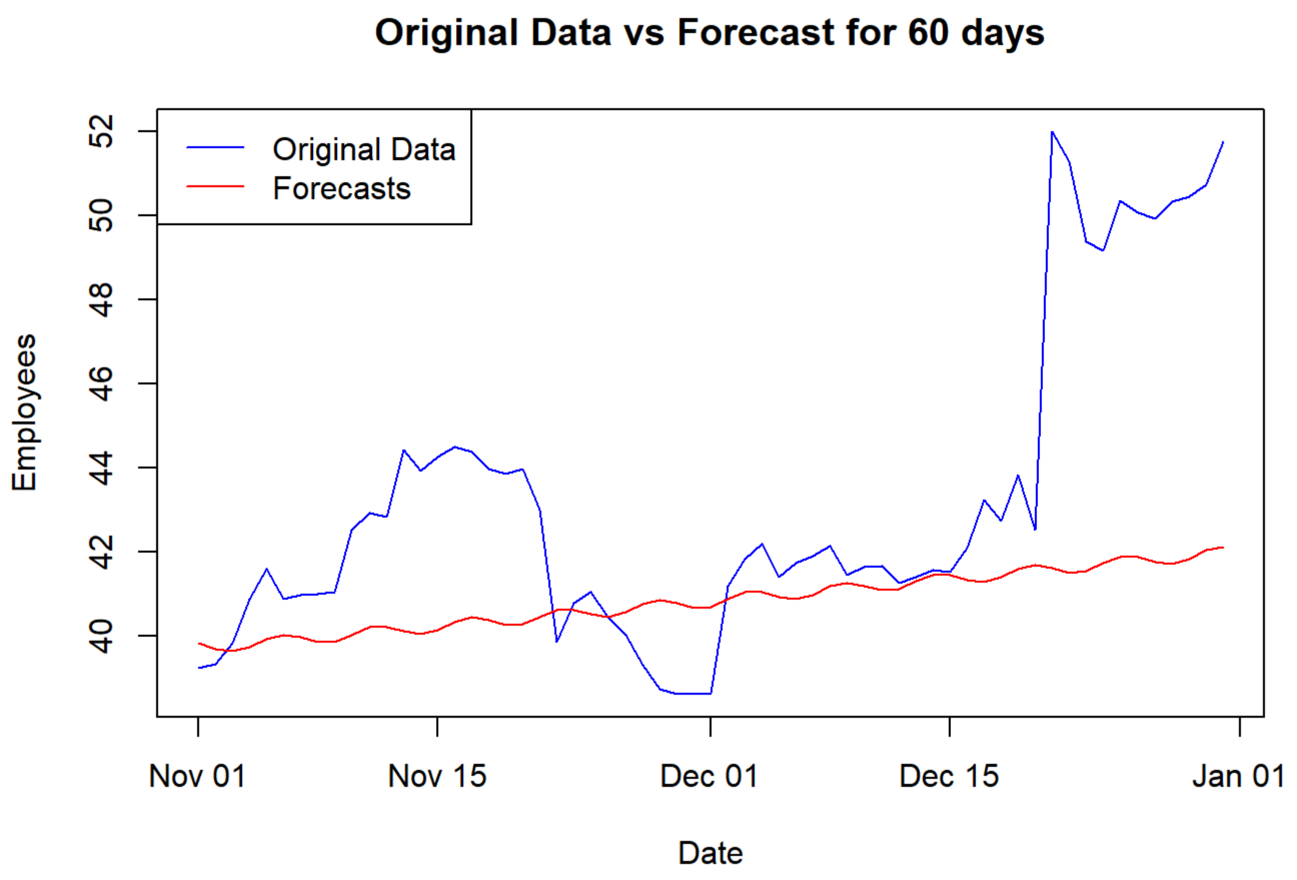
```
pacf(resid_norm, lag.max = 60, main='PACF of residual')
```



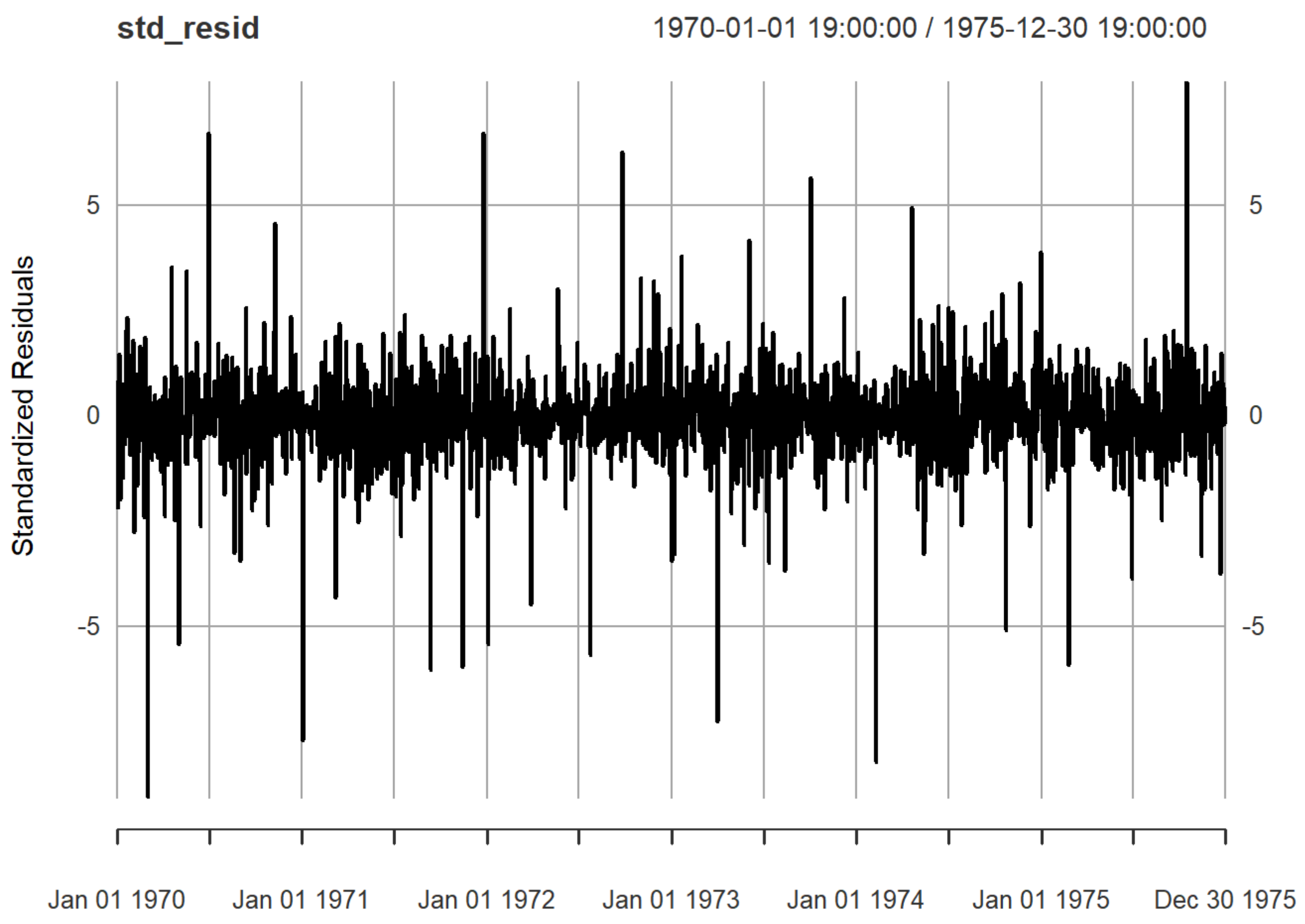
```
initial_term <- train_data$Close[length(train_data$Close)]
# Forecast next 60 days
forecasted <- ugarchforecast(garch_fit, n.ahead = 60, startMethod = "sample", initial.level =
initial_term)
# Extract the forecasted values
forecast_values <- fitted(forecasted)

predicted_diff <- cumsum(c(0, forecast_values))
predicted_log <- log(initial_term) + predicted_diff
predicted_close <- exp(predicted_log)
print(predicted_close[-1])
```

```
##  [1] 39.70713 39.64375 39.73771 39.93203 40.03817 39.97576 39.86185 39.87392
##  [9] 40.04119 40.21111 40.22714 40.11439 40.04957 40.15225 40.34593 40.44678
## [17] 40.37983 40.26839 40.28728 40.45825 40.62480 40.63481 40.52094 40.46164
## [25] 40.57045 40.76413 40.85926 40.78809 40.67933 40.70507 40.87950 41.04248
## [33] 41.04653 40.93182 40.87819 40.99299 41.18641 41.27574 41.20056 41.09473
## [41] 41.12734 41.30499 41.46419 41.46236 41.34708 41.29927 41.41992 41.61281
## [49] 41.69628 41.61730 41.51465 41.55411 41.73474 41.88996 41.88235 41.76677
## [57] 41.72492 41.85127 42.04336 42.12092
```

```
plot(test_data$Date, test_data$Close, main = 'Original Data vs Forecast for 60 days', type =
'l', col = 'blue', xlab = 'Date', ylab = 'Employees')

# Adding forecasted values to the plot (if 'original_forecasts' is aligned with 'test_data$Da
te')
lines(test_data$Date, predicted_close, col = 'red', type = 'l')

# Adding legend
legend('topleft', legend = c('Original Data', 'Forecasts'), col = c('blue', 'red'), lty = 1)
```



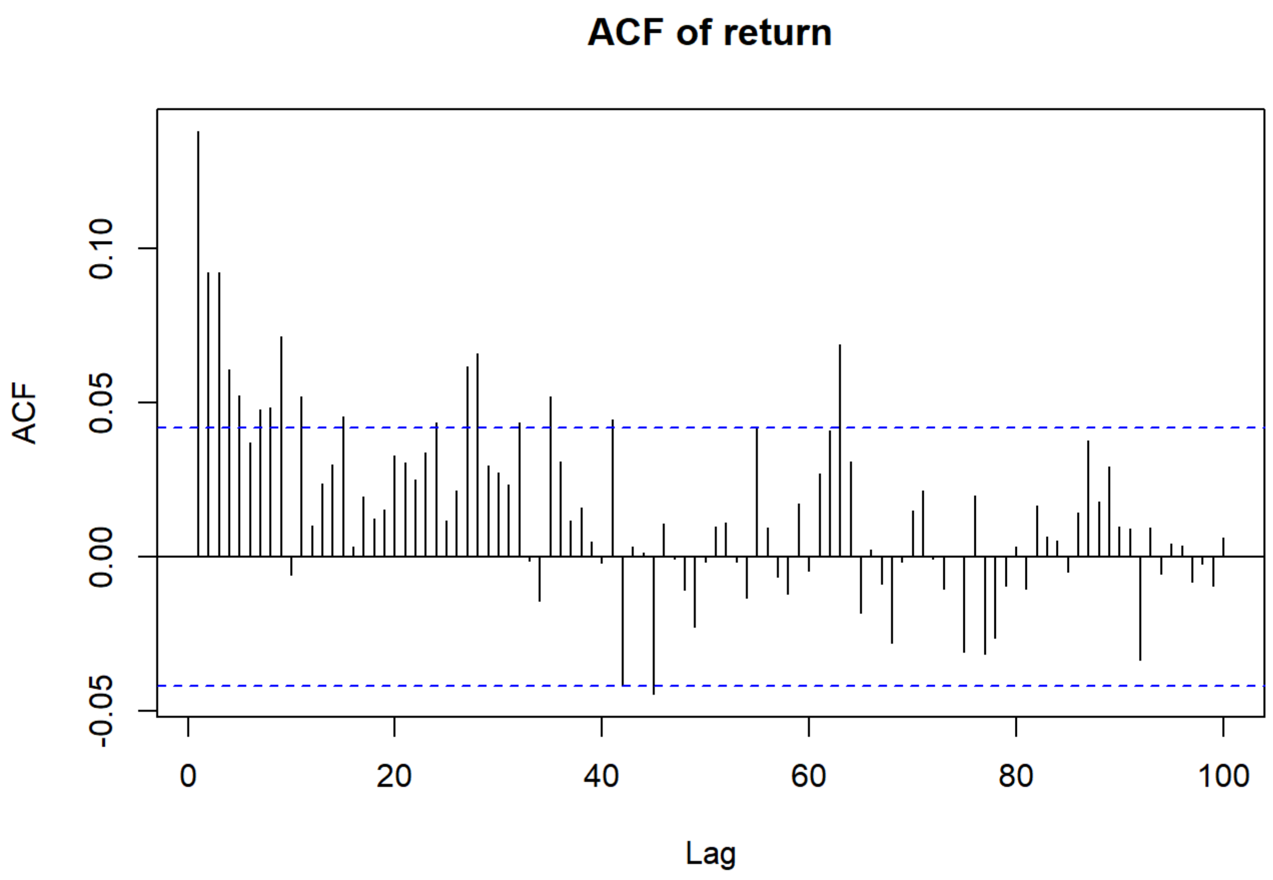
```
conditional_variance <- garch_fit@fit$sigma^2
std_resid <- resid_norm / sqrt(conditional_variance)
plot(std_resid, type = "l", ylab = "Standardized Residuals")
```



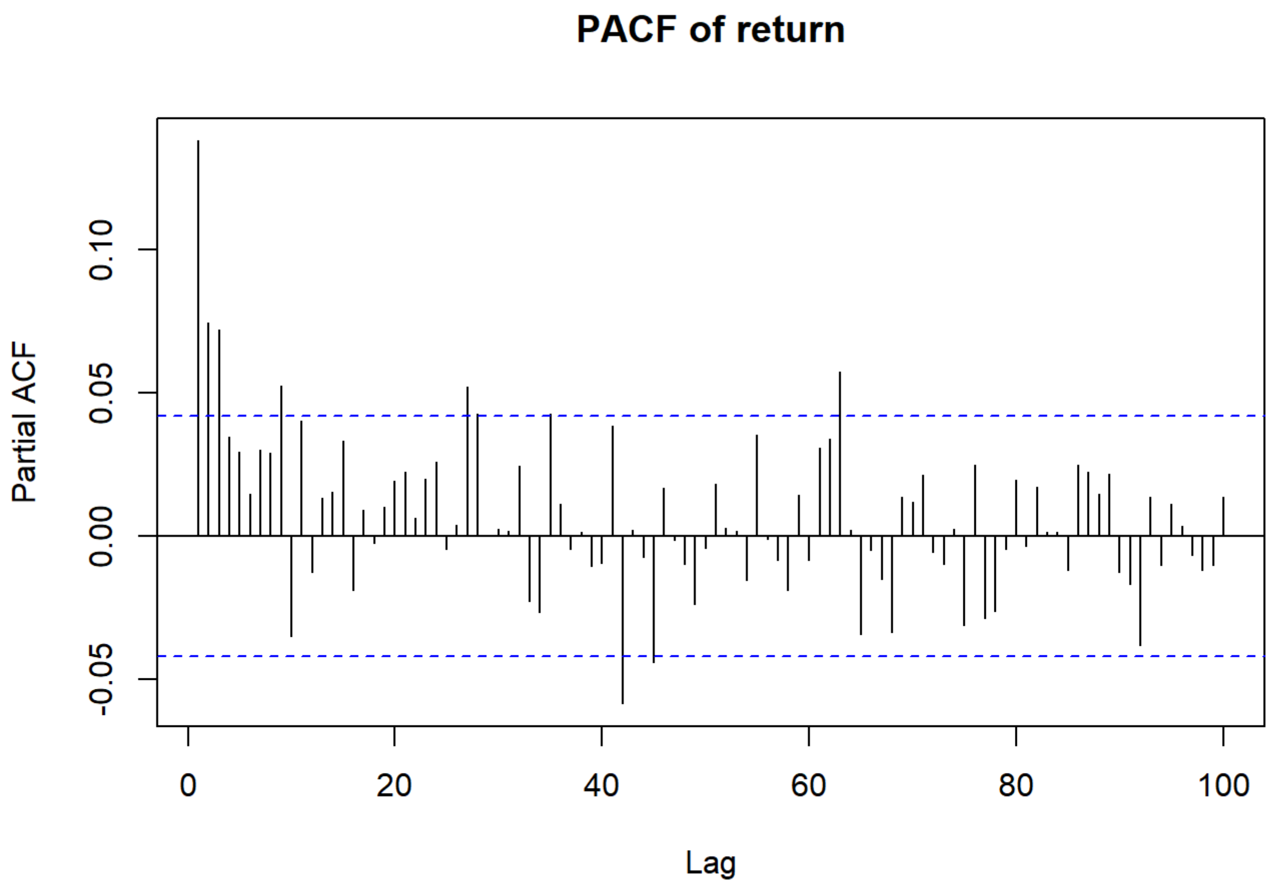
```
# Portmanteau test for squared residuals
Box.test(resid_norm, lag = 20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid_norm
## X-squared = 26.668, df = 20, p-value = 0.1449
```

```
###GARCH MODEL### abs(return)
return = abs(diff(log(train_data$Close)))

#Try with absolute and square of the returns
acf(return, lag.max = 100, main='ACF of return')
```



```
pacf(return, lag.max = 100, main='PACF of return')
```



```r
eacf(return)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x o x x x o x  o  o  o
## 1 x o o o o o o o x o x  o  o  o
## 2 x x o o o o o o o x o  o  o  o
## 3 x x x o o o o o o o o  o  o  o
## 4 x x o x o o o o o o o  o  o  o
## 5 x x x x x o o o o o o  o  o  o
## 6 x x x o x o o o o o o  o  o  o
## 7 x x o x o x o o o o o  o  o  o
```

```r
garch(x=return, grad='numerical', trace=FALSE)
```

```
##
## Call:
## garch(x = return, grad = "numerical", trace = FALSE)
##
## Coefficient(s):
##        a0         a1         b1
## 0.0007055  0.1562154  0.2625816
```

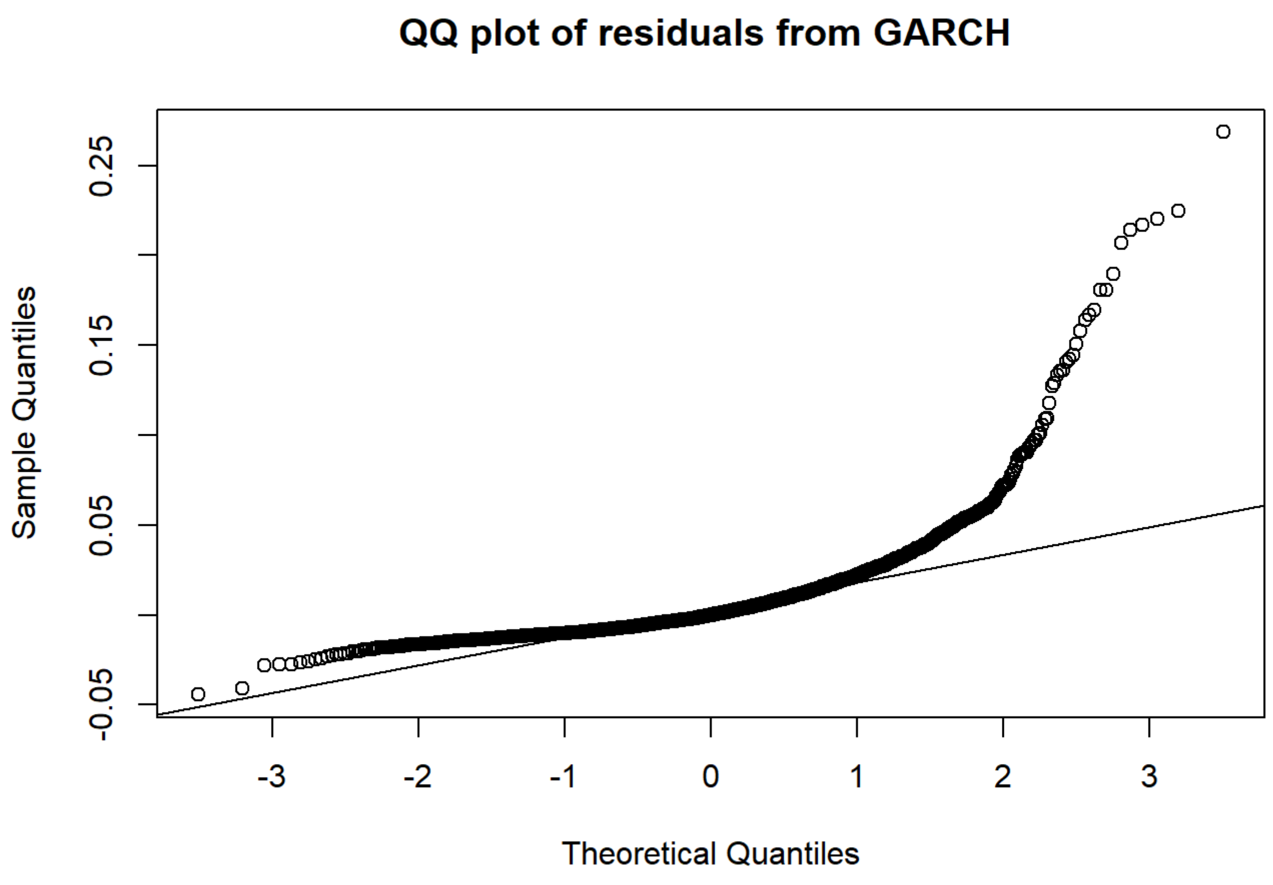
```r
garch_spec <- ugarchspec(variance.model = list(garchOrder = c(2,2)),
                         mean.model = list(armaOrder = c(3,5)),
                         distribution.model = 'std')

garch_fit <- ugarchfit(spec = garch_spec, data = return, solver = 'hybrid')
summary(garch_fit)
```

```
##    Length     Class      Mode
##         1 uGARCHfit        S4
```

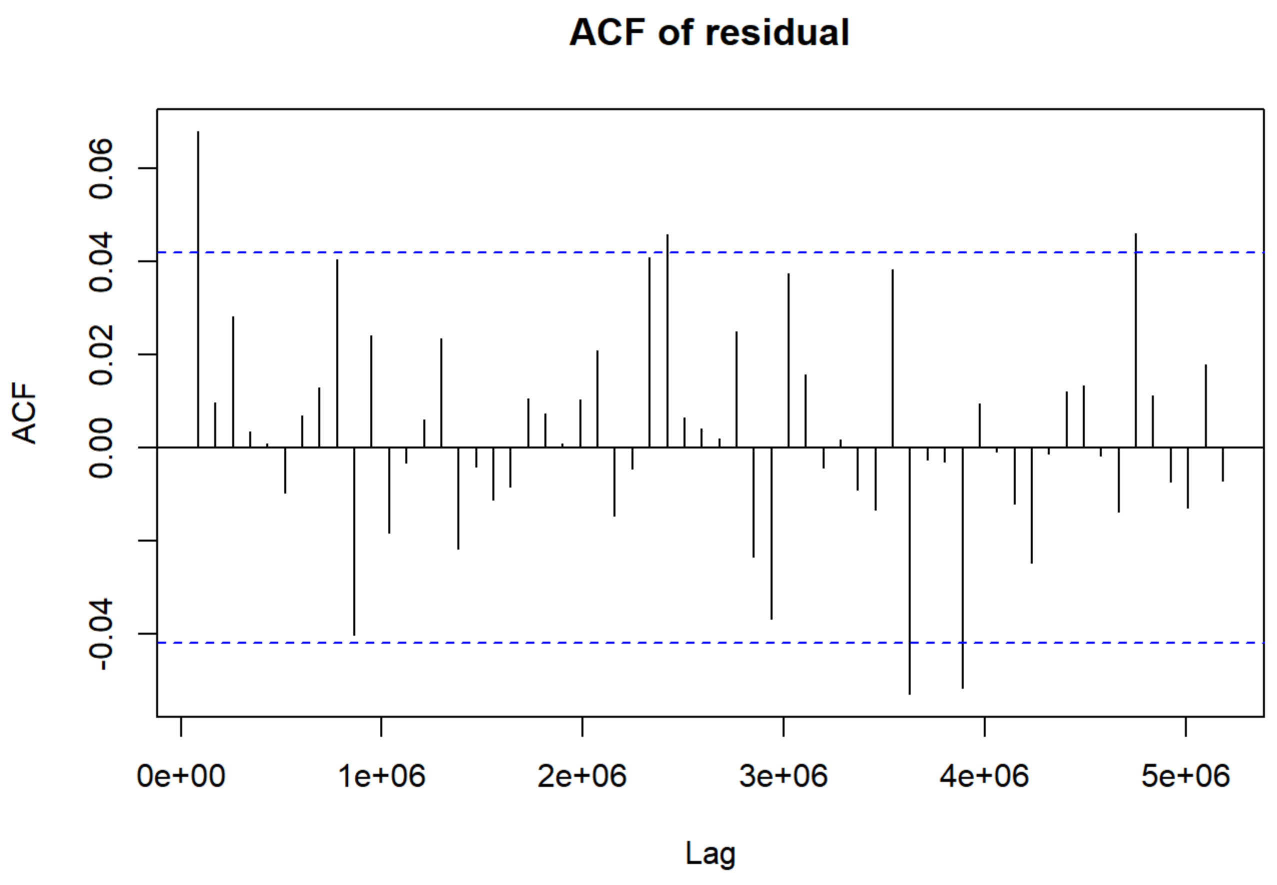
```r
garch_fit
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,2)
## Mean Model   : ARFIMA(3,0,5)
## Distribution : std
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error     t value Pr(>|t|)
## mu      0.011455    0.000852  1.3451e+01 0.000000
## ar1     2.788492    0.000769  3.6247e+03 0.000000
## ar2    -2.594705    0.000705 -3.6806e+03 0.000000
## ar3     0.805183    0.000288  2.7994e+03 0.000000
## ma1    -2.742870    0.000029 -9.3875e+04 0.000000
## ma2     2.524763    0.000023  1.0891e+05 0.000000
## ma3    -0.805629    0.000005 -1.4896e+05 0.000000
## ma4     0.032426    0.000045  7.2461e+02 0.000000
## ma5    -0.007137    0.000055 -1.2905e+02 0.000000
## omega   0.000475    0.000270  1.7586e+00 0.078642
## alpha1  0.242621    0.047933  5.0617e+00 0.000000
## alpha2  0.000001    0.103061  1.2000e-05 0.999990
## beta1   0.380498    0.772939  4.9228e-01 0.622525
## beta2   0.103455    0.438344  2.3602e-01 0.813421
## shape   2.241055    0.240554  9.3162e+00 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error     t value Pr(>|t|)
## mu      0.011455    0.050797  2.2550e-01 0.821589
## ar1     2.788492    0.040204  6.9359e+01 0.000000
## ar2    -2.594705    0.036941 -7.0240e+01 0.000000
## ar3     0.805183    0.015355  5.2437e+01 0.000000
## ma1    -2.742870    0.000495 -5.5412e+03 0.000000
## ma2     2.524763    0.000212  1.1904e+04 0.000000
## ma3    -0.805629    0.000160 -5.0223e+03 0.000000
## ma4     0.032426    0.001440  2.2520e+01 0.000000
## ma5    -0.007137    0.002060 -3.4643e+00 0.000532
## omega   0.000475    0.001973  2.4073e-01 0.809764
## alpha1  0.242621    5.501438  4.4101e-02 0.964824
## alpha2  0.000001    8.446302  0.0000e+00 1.000000
## beta1   0.380498   51.141019  7.4400e-03 0.994064
## beta2   0.103455   28.975948  3.5700e-03 0.997151
## shape   2.241055   13.384032  1.6744e-01 0.867022
##
## LogLikelihood : 5665.147
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.1600
## Bayes        -5.1210
## Shibata      -5.1600
## Hannan-Quinn -5.1457
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                       1.229  0.2676
## Lag[2*(p+q)+(p+q)-1][23]     8.464  1.0000
## Lag[4*(p+q)+(p+q)-1][39]    15.227  0.9352
## d.o.f=8
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                      0.4064  0.5238
## Lag[2*(p+q)+(p+q)-1][11]    2.5989  0.9121
## Lag[4*(p+q)+(p+q)-1][19]    4.3415  0.9679
## d.o.f=4
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[5]    0.0481 0.500 2.000  0.8264
## ARCH Lag[7]    0.6811 1.473 1.746  0.8473
## ARCH Lag[9]    2.0578 2.402 1.619  0.7473
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.2264
## Individual Statistics:
## mu     0.15980
## ar1    0.03843
## ar2    0.03812
## ar3    0.03882
## ma1    0.03781
## ma2    0.03484
## ma3    0.03381
## ma4    0.03495
## ma5    0.03783
## omega  0.42450
## alpha1 0.27384
## alpha2 0.23485
## beta1  0.35722
## beta2  0.33987
## shape  0.41464
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          3.26 3.54 4.07
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.2150 0.8298
## Negative Sign Bias  0.6384 0.5233
## Positive Sign Bias  0.7128 0.4760
## Joint Effect        1.3711 0.7123
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     630.3   1.971e-121
## 2    30     675.7   1.179e-123
## 3    40     688.3   1.127e-119
## 4    50     693.3   1.180e-114
##
##
## Elapsed time : 2.422965
```

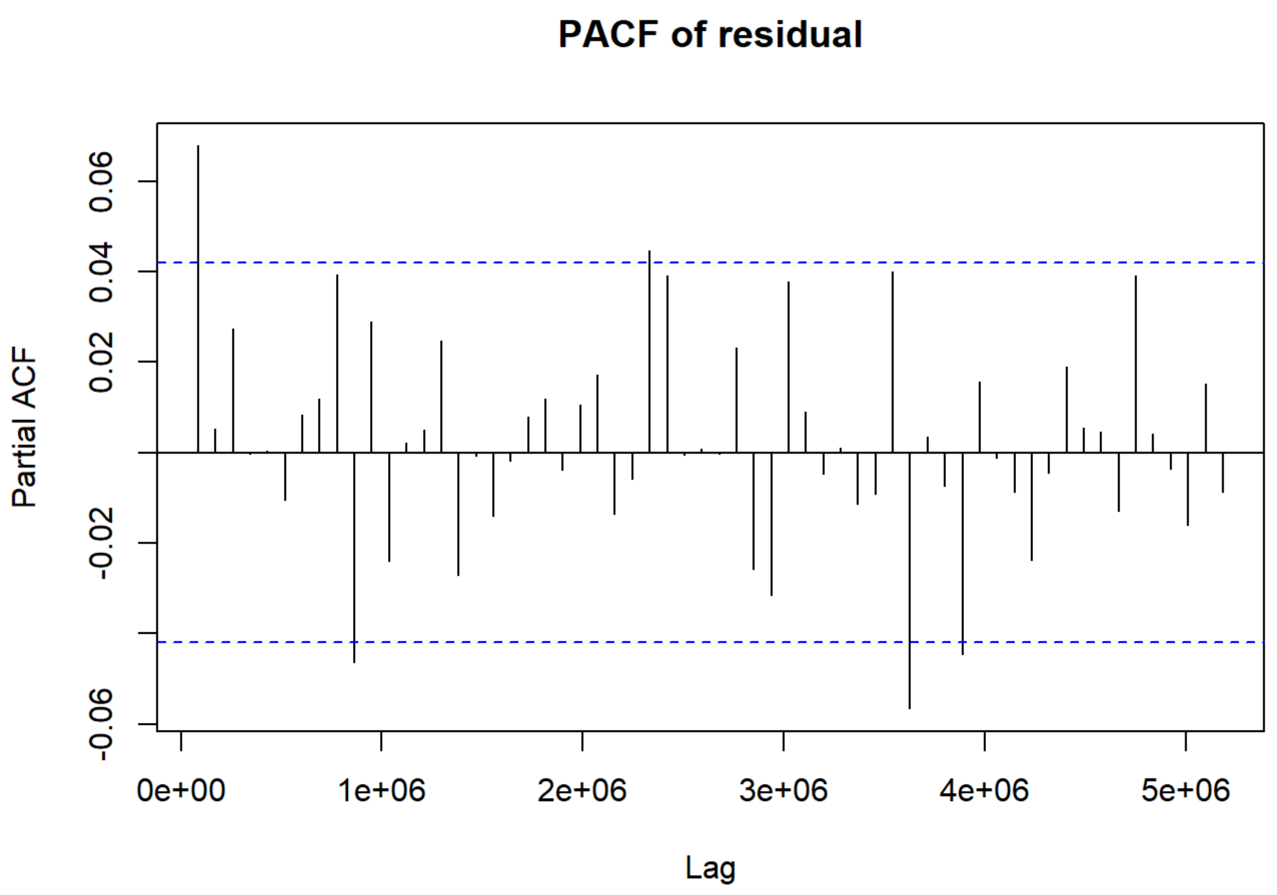
```
resid_norm = residuals(garch_fit)
qqnorm(resid_norm, main='QQ plot of residuals from GARCH ')
qqline(resid_norm)
```



```
acf(resid_norm, lag.max = 60, main='ACF of residual')
```



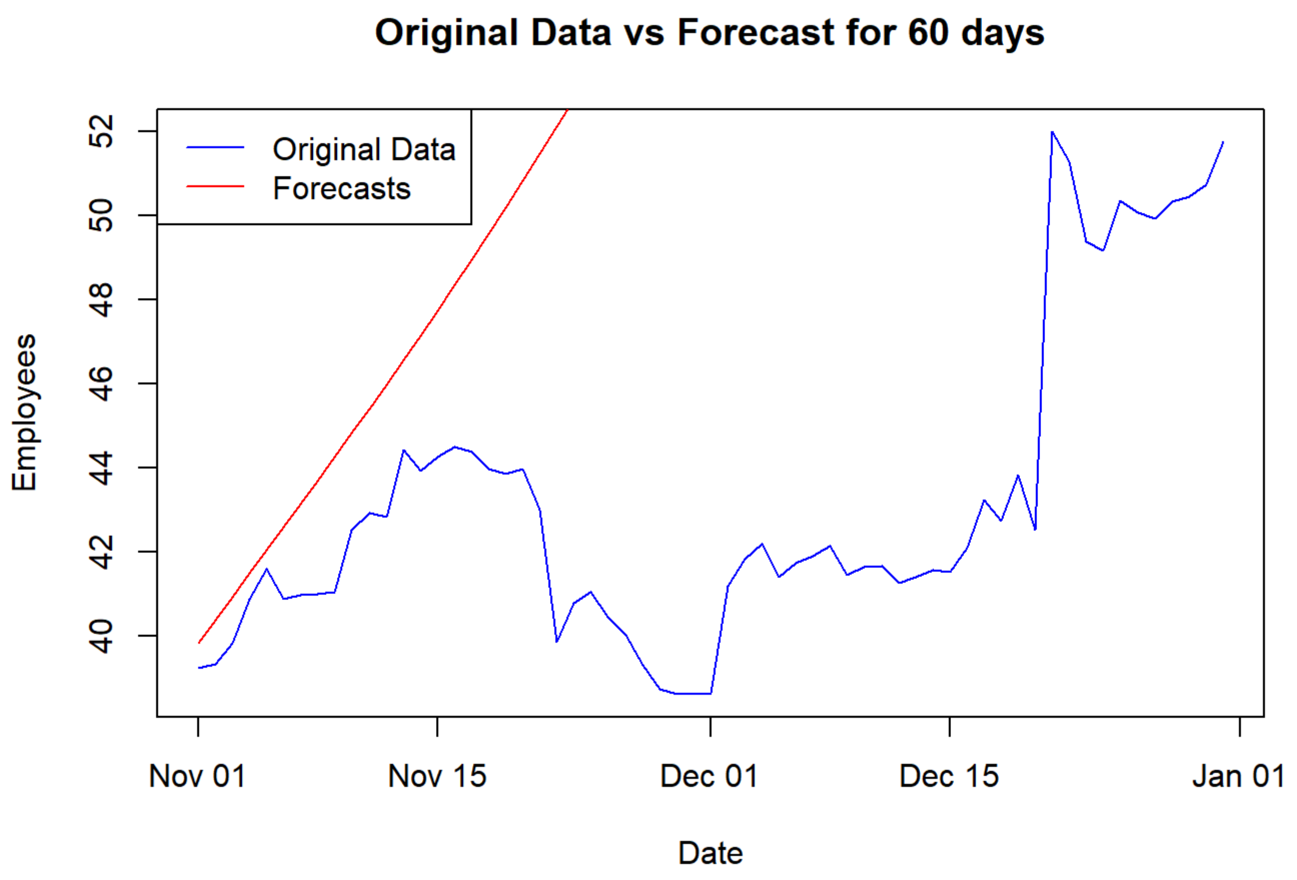
```
pacf(resid_norm, lag.max = 60, main='PACF of residual')
```



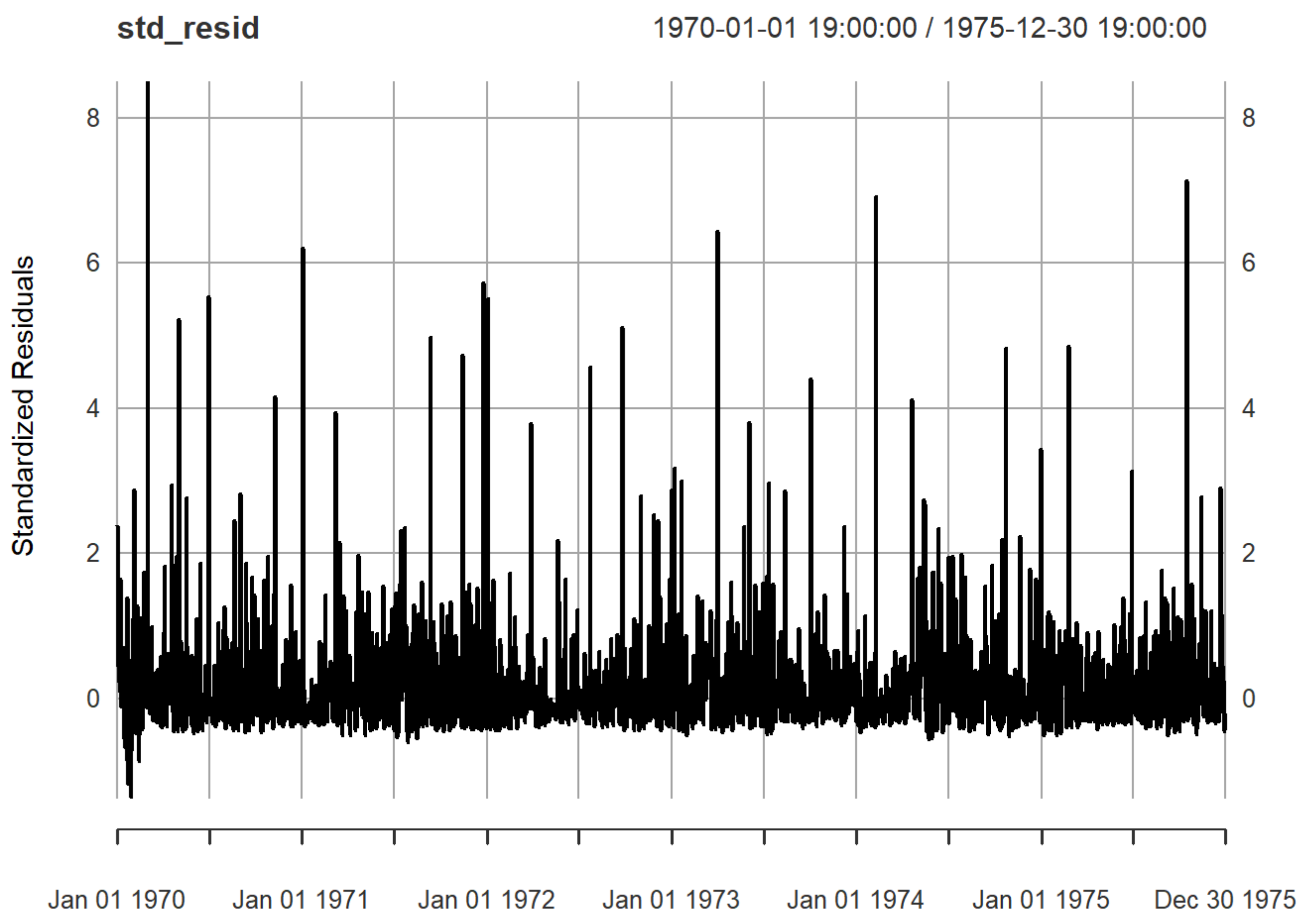
```
initial_term <- train_data$Close[length(train_data$Close)]
# Forecast next 60 days
forecasted <- ugarchforecast(garch_fit, n.ahead = 60, startMethod = "sample")
# Extract the forecasted values
forecast_values <- fitted(forecasted)
predicted_diff <- cumsum(c(0, forecast_values))
predicted_log <- log(initial_term) + predicted_diff
predicted_close <- exp(predicted_log)
print(predicted_close[-1])
```

```
##  [1] 40.39443 40.94474 41.49558 42.04616 42.59786 43.15184 43.70909 44.27042
##  [9] 44.83651 45.40793 45.98512 46.56843 47.15813 47.75441 48.35742 48.96723
## [17] 49.58388 50.20737 50.83767 51.47473 52.11847 52.76882 53.42568 54.08895
## [25] 54.75853 55.43432 56.11623 56.80417 57.49807 58.19786 58.90349 59.61492
## [33] 60.33213 61.05510 61.78386 62.51843 63.25885 64.00519 64.75752 65.51593
## [41] 66.28055 67.05150 67.82891 68.61296 69.40380 70.20163 71.00663 71.81901
## [49] 72.63899 73.46679 74.30263 75.14677 75.99943 76.86087 77.73133 78.61106
## [57] 79.50033 80.39937 81.30845 82.22781
```

```
plot(test_data$Date, test_data$Close, main = 'Original Data vs Forecast for 60 days', type =
'l', col = 'blue', xlab = 'Date', ylab = 'Employees')

# Adding forecasted values to the plot (if 'original_forecasts' is aligned with 'test_data$Da
te')
lines(test_data$Date, predicted_close, col = 'red', type = 'l')

# Adding legend
legend('topleft', legend = c('Original Data', 'Forecasts'), col = c('blue', 'red'), lty = 1)
```



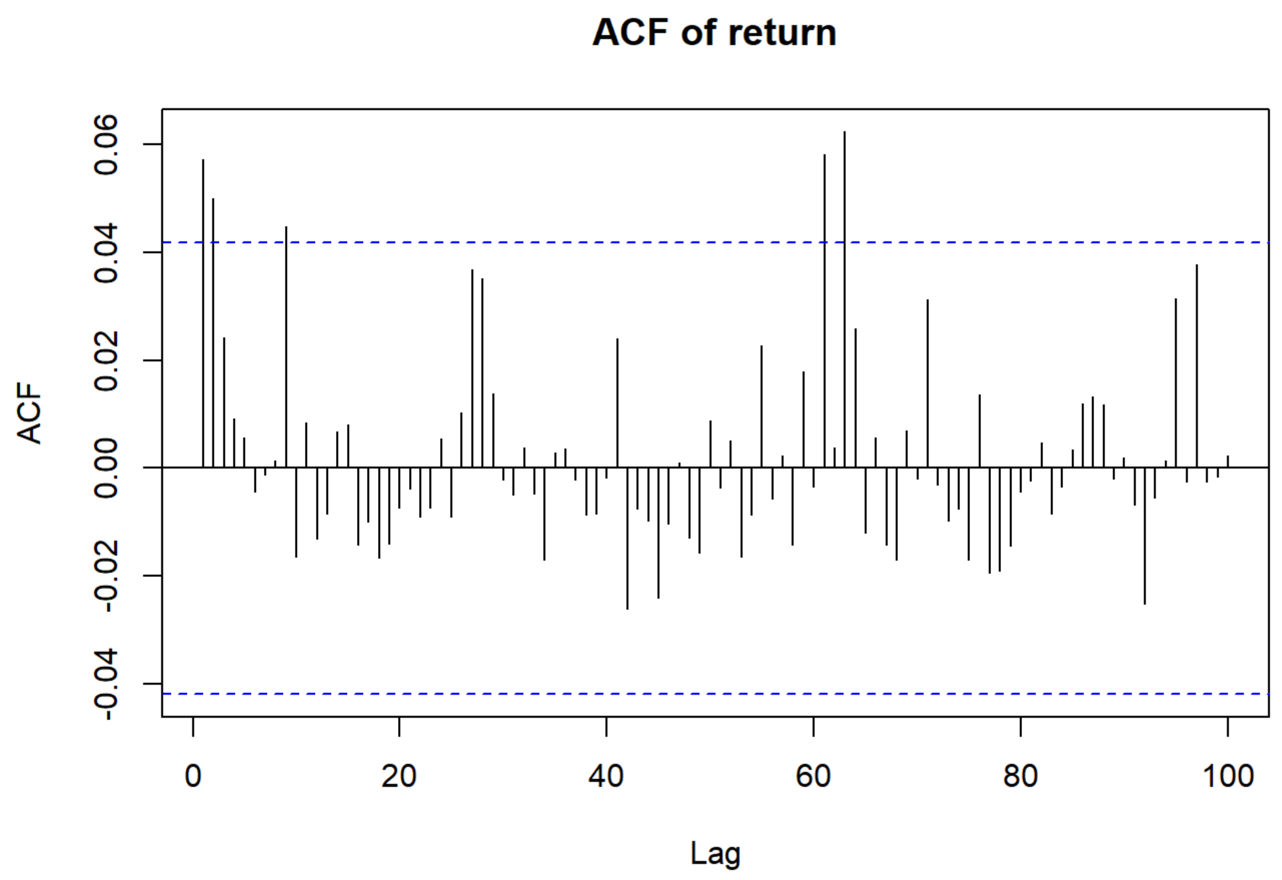
```
conditional_variance <- garch_fit@fit$sigma^2
std_resid <- resid_norm / sqrt(conditional_variance)
plot(std_resid, type = "l", ylab = "Standardized Residuals")
```



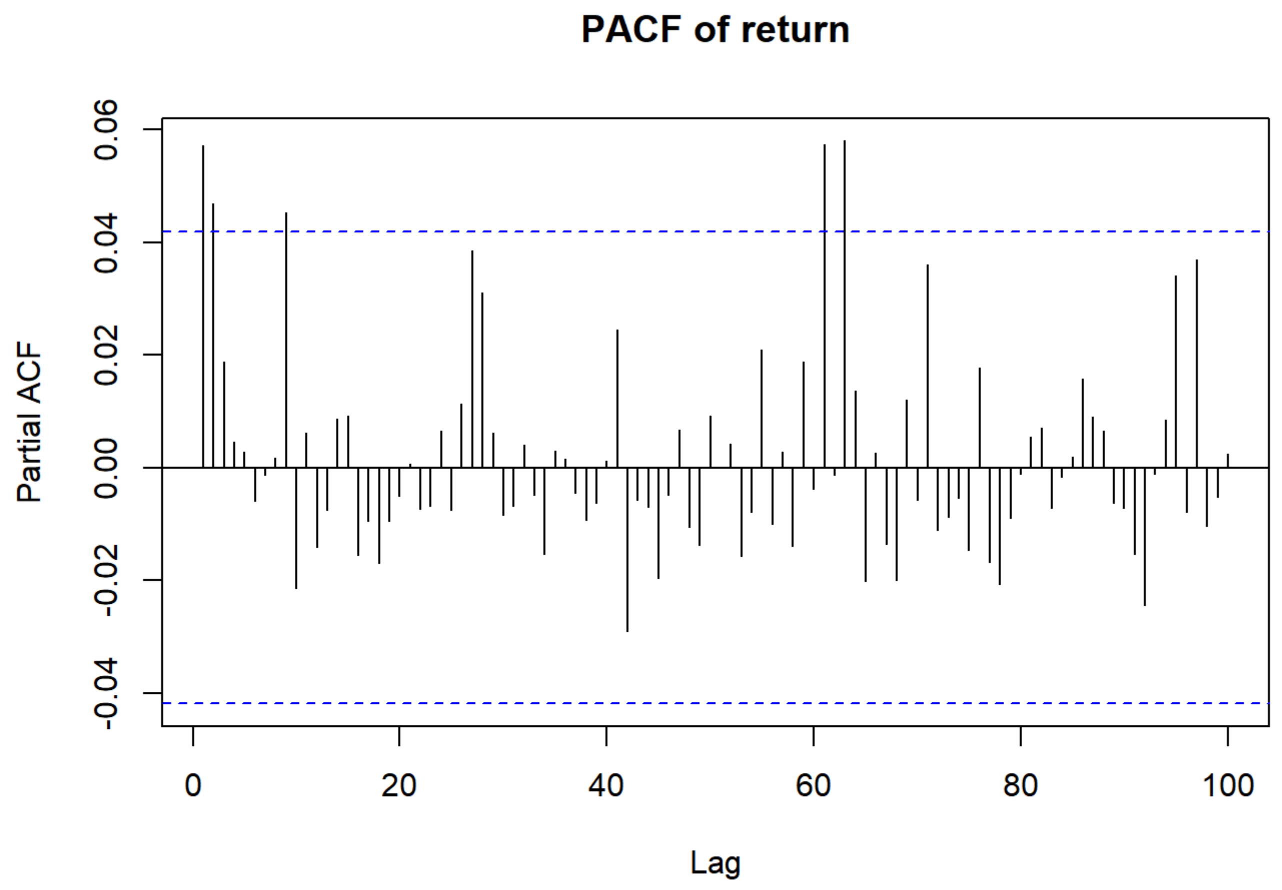
```
# Portmanteau test for squared residuals
Box.test(resid_norm, lag = 20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid_norm
## X-squared = 24.9, df = 20, p-value = 0.2053
```

```
###GARCH MODEL### (return)^2
return = diff(log(train_data$Close))^2

#Try with absolute and square of the returns
acf(return, lag.max = 100, main='ACF of return')
```



```
pacf(return, lag.max = 100, main='PACF of return')
```



```
eacf(return)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x o o o o o o x 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 x 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 x x x o o o o o o o  o  o  o
## 6 x x x o o o o o o o o  o  o  o
## 7 x o x x o o x o o o o  o  o  o
```

```
garch(x=return, grad='numerical', trace=FALSE)
```

```
##
## Call:
## garch(x = return, grad = "numerical", trace = FALSE)
##
## Coefficient(s):
##        a0         a1         b1
## 1.796e-05  5.000e-02  5.000e-02
```

```
garch_spec <- ugarchspec(variance.model = list(garchOrder = c(2,2)),
                         mean.model = list(armaOrder = c(2,2)),
                         distribution.model = 'std')

garch_fit <- ugarchfit(spec = garch_spec, data = return, solver = 'hybrid')
summary(garch_fit)
```

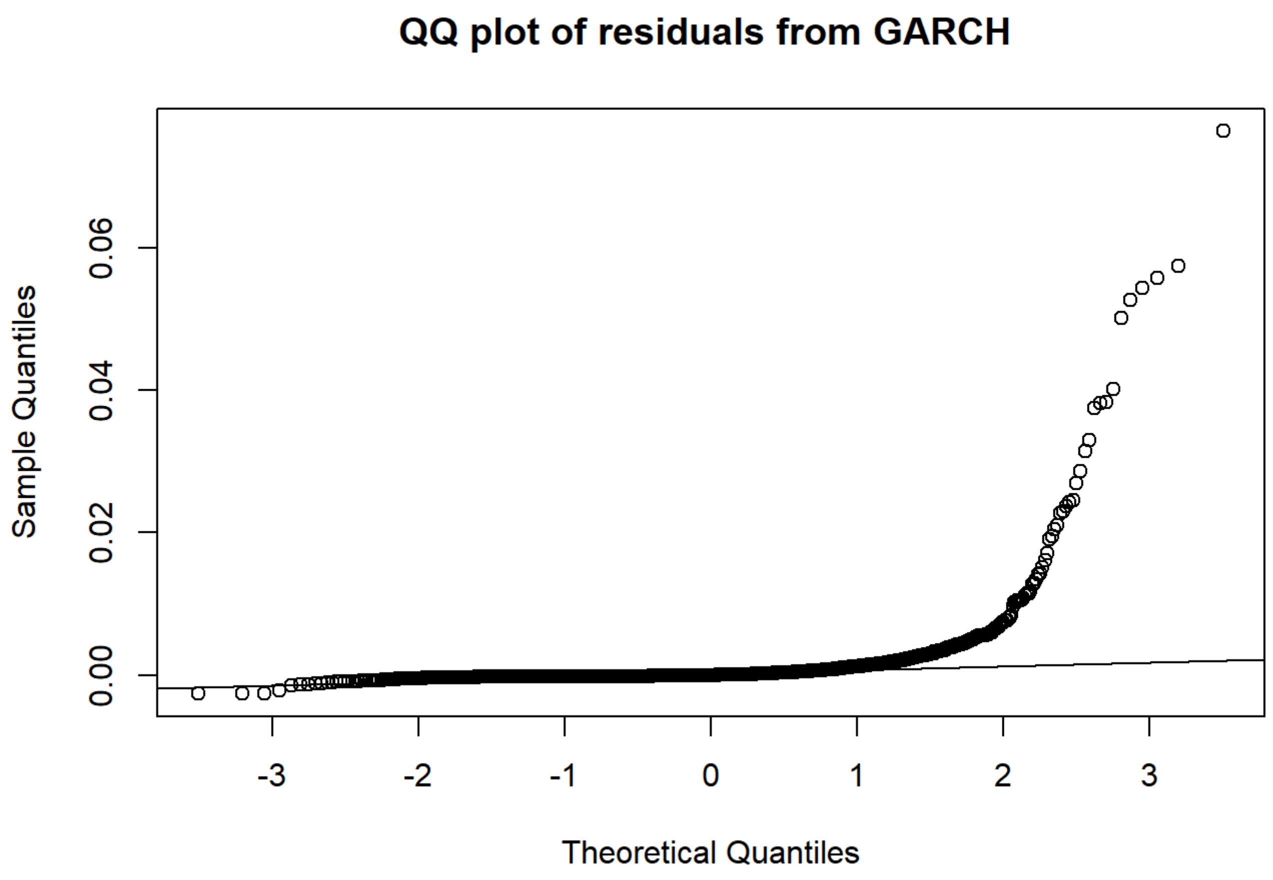
```
##    Length     Class      Mode
##         1 uGARCHfit        S4
```

```
garch_fit
```

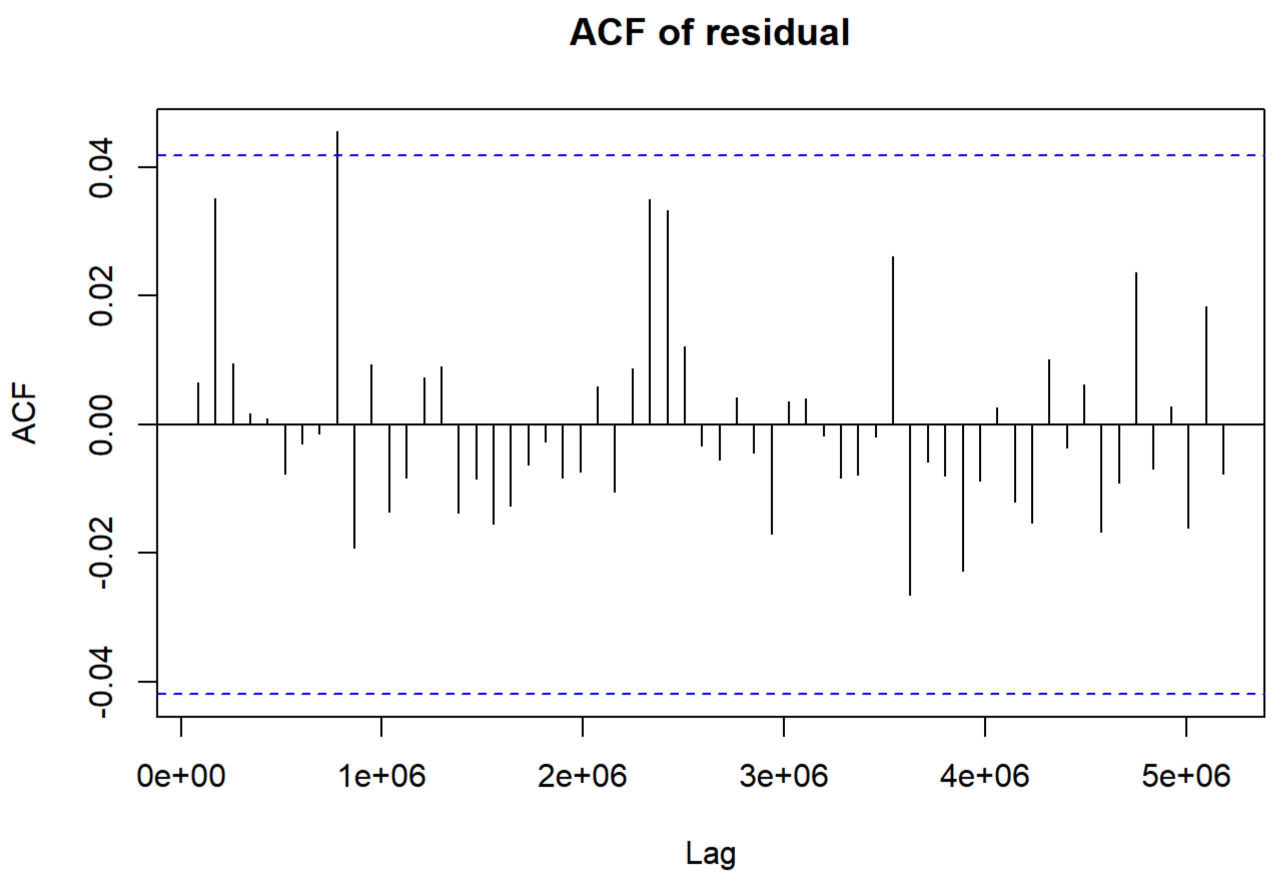
```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,2)
## Mean Model   : ARFIMA(2,0,2)
## Distribution : std
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error    t value Pr(>|t|)
## mu      0.000175    0.000010  17.430561 0.000000
## ar1     0.307328    0.253452   1.212568 0.225295
## ar2     0.174055    0.129746   1.341512 0.179754
## ma1    -0.259778    0.252649  -1.028216 0.303848
## ma2    -0.175460    0.119674  -1.466156 0.142606
## omega   0.000001    0.000000  20.942552 0.000000
## alpha1  0.315726    0.075703   4.170598 0.000030
## alpha2  0.000000    0.026681   0.000006 0.999995
## beta1   0.061222    0.049940   1.225915 0.220231
## beta2   0.070403    0.027900   2.523440 0.011621
## shape   2.100000    0.005120 410.177552 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error    t value Pr(>|t|)
## mu      0.000175    0.000009  19.715389 0.000000
## ar1     0.307328    0.153501   2.002131 0.045271
## ar2     0.174055    0.101886   1.708330 0.087575
## ma1    -0.259778    0.152802  -1.700094 0.089113
## ma2    -0.175460    0.107019  -1.639529 0.101103
## omega   0.000001    0.000000   6.570122 0.000000
## alpha1  0.315726    0.158096   1.997054 0.045819
## alpha2  0.000000    0.057880   0.000003 0.999998
## beta1   0.061222    0.065110   0.940289 0.347070
## beta2   0.070403    0.055334   1.272334 0.203254
## shape   2.100000    0.003994 525.757467 0.000000
##
## LogLikelihood : 12372.85
##
## Information Criteria
## ------------------------------------
##
## Akaike       -11.289
## Bayes        -11.261
## Shibata      -11.289
## Hannan-Quinn -11.279
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.8861  0.3465
## Lag[2*(p+q)+(p+q)-1][11]    2.9258  1.0000
## Lag[4*(p+q)+(p+q)-1][19]    4.8089  0.9960
## d.o.f=4
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.08295  0.7733
## Lag[2*(p+q)+(p+q)-1][11]   0.52661  0.9997
## Lag[4*(p+q)+(p+q)-1][19]   0.88836  1.0000
## d.o.f=4
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[5]   0.04601 0.500 2.000  0.8301
## ARCH Lag[7]   0.16023 1.473 1.746  0.9785
## ARCH Lag[9]   0.43076 2.402 1.619  0.9884
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  307.7095
## Individual Statistics:
## mu       0.2002
## ar1      0.1278
## ar2      0.4821
## ma1      0.1021
## ma2      0.4666
## omega  100.4373
## alpha1   0.1974
## alpha2   0.2747
## beta1    0.3395
## beta2    0.1200
## shape    1.7231
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.49 2.75 3.27
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.0922 0.9265
## Negative Sign Bias  0.1966 0.8441
## Positive Sign Bias  0.5339 0.5934
## Joint Effect        0.3383 0.9527
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20      2448            0
## 2    30      2827            0
## 3    40      3064            0
## 4    50      3189            0
##
```

```
##
## Elapsed time : 1.082848
```

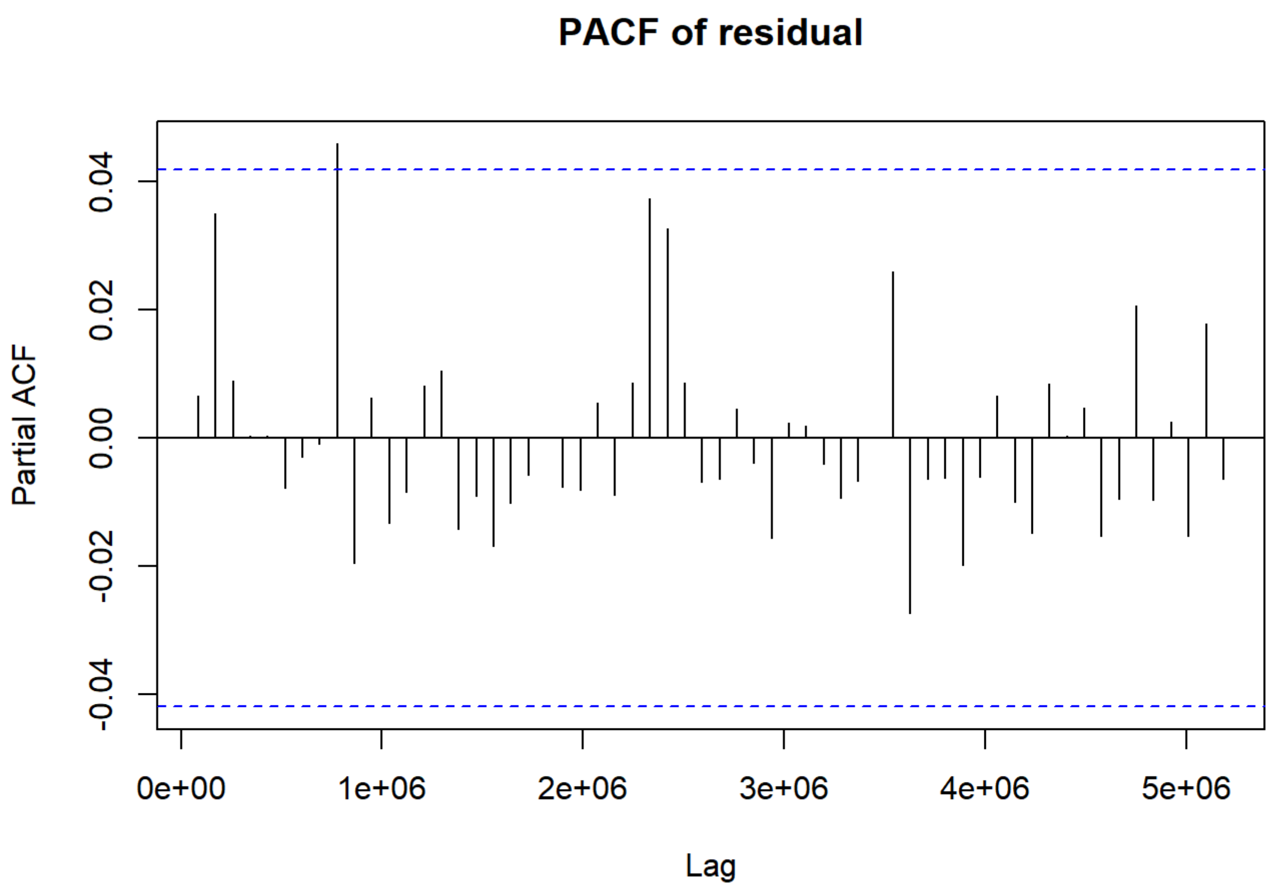
```
resid_norm = residuals(garch_fit)
qqnorm(resid_norm, main='QQ plot of residuals from GARCH ')
qqline(resid_norm)
```



```
acf(resid_norm, lag.max = 60, main='ACF of residual')
```



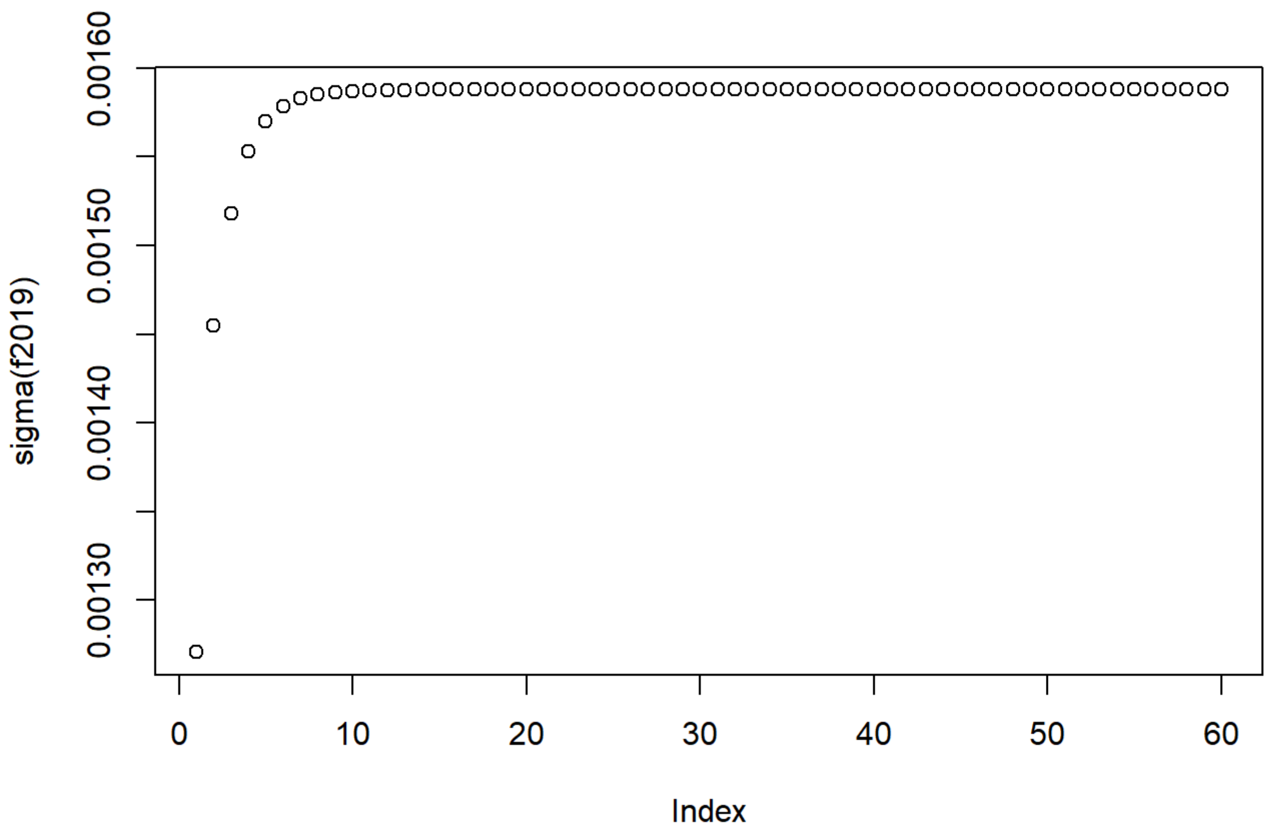
```
pacf(resid_norm, lag.max = 60, main='PACF of residual')
```



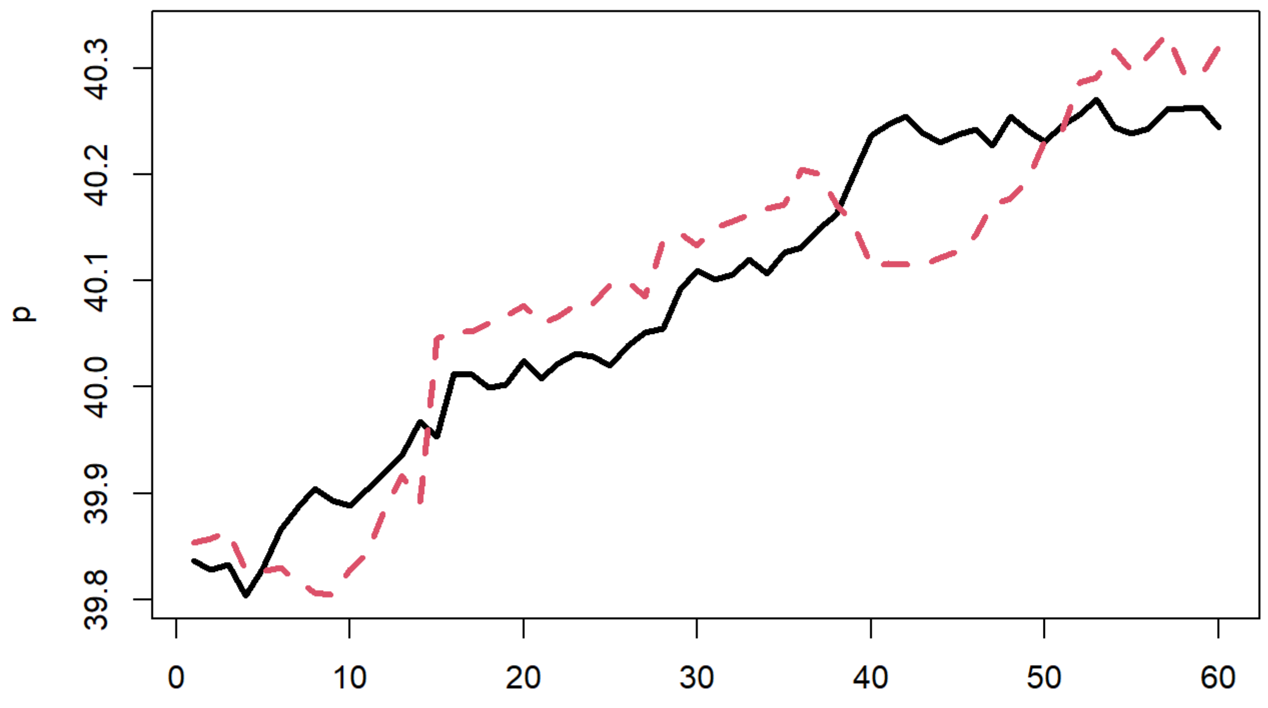
```
sfinal <- garch_spec
setfixed(sfinal) <- as.list(coef(garch_fit))
coef(garch_fit)
```

```
##            mu           ar1           ar2           ma1           ma2
##  1.745898e-04  3.073284e-01  1.740552e-01 -2.597778e-01 -1.754602e-01
##         omega        alpha1        alpha2         beta1         beta2
##  1.393437e-06  3.157256e-01  1.650788e-07  6.122200e-02  7.040305e-02
##         shape
##  2.100000e+00
```

```
f2019 <- ugarchforecast(data = return, fitORspec = sfinal, n.ahead = 60)
plot(sigma(f2019))
```



```
sim <- ugarchpath(spec = sfinal, m.sim = 2, n.sim = 1*60, rseed = 123)
p <- 39.84*apply(fitted(sim), 2, 'cumsum') + 39.84
matplot(p, type = "l", lwd = 3)
```



GJR-GARCH Model

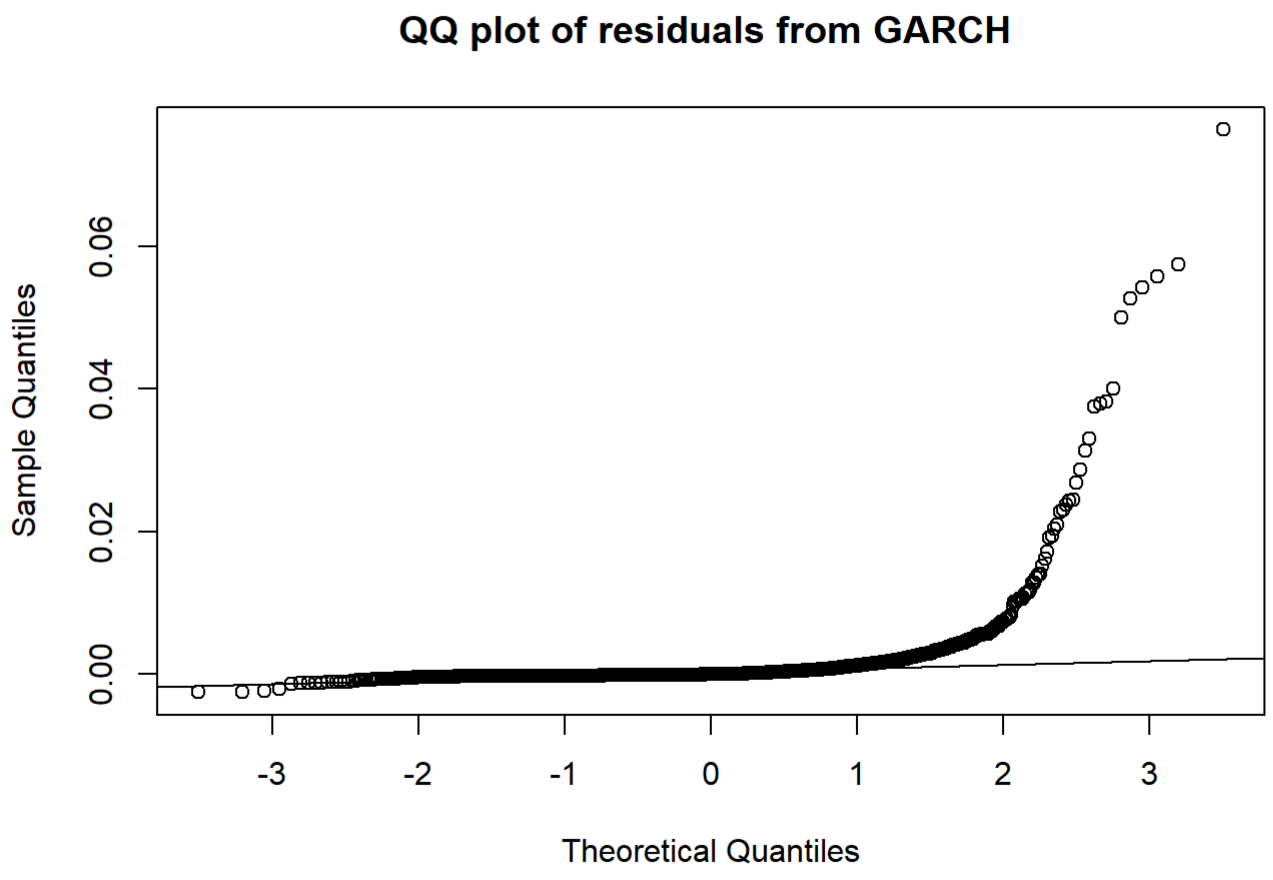
```
garch_spec <- ugarchspec(variance.model = list(model = 'gjrGARCH'),
                         mean.model = list(armaOrder = c(2,2)),
                         distribution.model = 'std')

garch_fit <- ugarchfit(spec = garch_spec, data = return, solver = 'hybrid')
summary(garch_fit)
```

```
##    Length     Class      Mode
##         1 uGARCHfit        S4
```

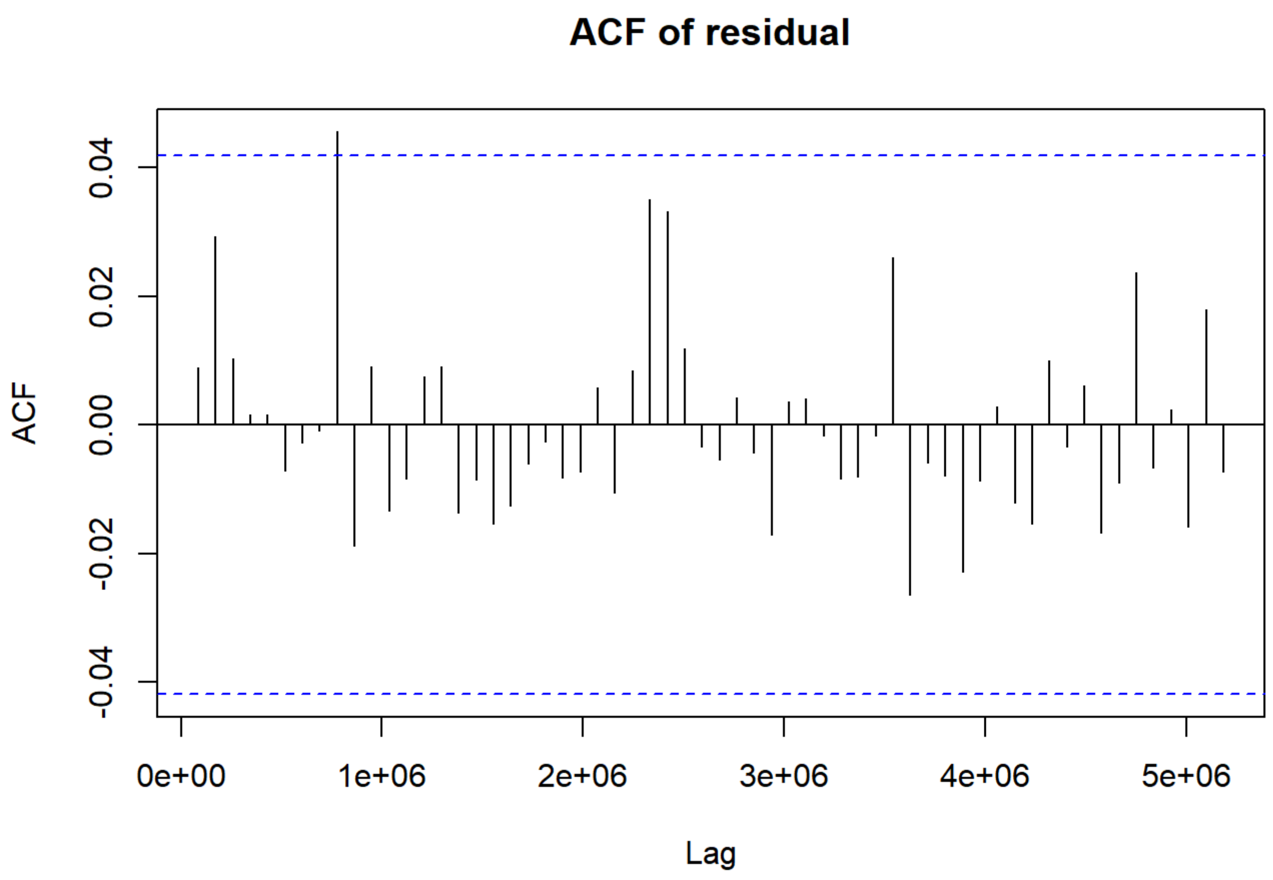
```
garch_fit
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,1)
## Mean Model   : ARFIMA(2,0,2)
## Distribution : std
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000176    0.000010  17.7343 0.000000
## ar1     0.299051    0.141868   2.1080 0.035035
## ar2     0.133908    0.076296   1.7551 0.079240
## ma1    -0.253817    0.142248  -1.7843 0.074370
## ma2    -0.128463    0.072414  -1.7740 0.076063
## omega   0.000001    0.000000 261.5065 0.000000
## alpha1  0.255672    0.073042   3.5004 0.000465
## beta1   0.241499    0.034748   6.9501 0.000000
## gamma1 -0.994294    0.239701  -4.1481 0.000034
## shape   2.100000    0.005075 413.7779 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000176    0.000011  16.2500 0.000000
## ar1     0.299051    0.047633   6.2782 0.000000
## ar2     0.133908    0.043542   3.0754 0.002102
## ma1    -0.253817    0.053985  -4.7016 0.000003
## ma2    -0.128463    0.044458  -2.8895 0.003858
## omega   0.000001    0.000000 161.2818 0.000000
## alpha1  0.255672    0.181408   1.4094 0.158725
## beta1   0.241499    0.094111   2.5661 0.010285
## gamma1 -0.994294    0.348898  -2.8498 0.004374
## shape   2.100000    0.004362 481.4290 0.000000
##
## LogLikelihood : 12370.83
##
## Information Criteria
## ------------------------------------
##
## Akaike       -11.288
## Bayes        -11.262
## Shibata      -11.288
## Hannan-Quinn -11.279
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.7567  0.3844
## Lag[2*(p+q)+(p+q)-1][11]    3.1099  1.0000
## Lag[4*(p+q)+(p+q)-1][19]    5.1302  0.9926
## d.o.f=4
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.08195  0.7747
## Lag[2*(p+q)+(p+q)-1][5]   0.23205  0.9901
## Lag[4*(p+q)+(p+q)-1][9]   0.41910  0.9991
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.07913 0.500 2.000  0.7785
## ARCH Lag[5]   0.15980 1.440 1.667  0.9747
## ARCH Lag[7]   0.24070 2.315 1.543  0.9955
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  300.055
## Individual Statistics:
## mu       0.17571
## ar1      0.07727
## ar2      0.48637
## ma1      0.07766
## ma2      0.48084
## omega  101.63582
## alpha1   0.23280
## beta1    0.27477
## gamma1   0.37373
## shape    1.71021
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.29 2.54 3.05
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.1321 0.8949
## Negative Sign Bias  0.2407 0.8098
## Positive Sign Bias  0.5131 0.6079
## Joint Effect        0.3392 0.9525
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20      2449            0
## 2    30      2868            0
## 3    40      3101            0
## 4    50      3253            0
##
##
## Elapsed time : 1.962556
```

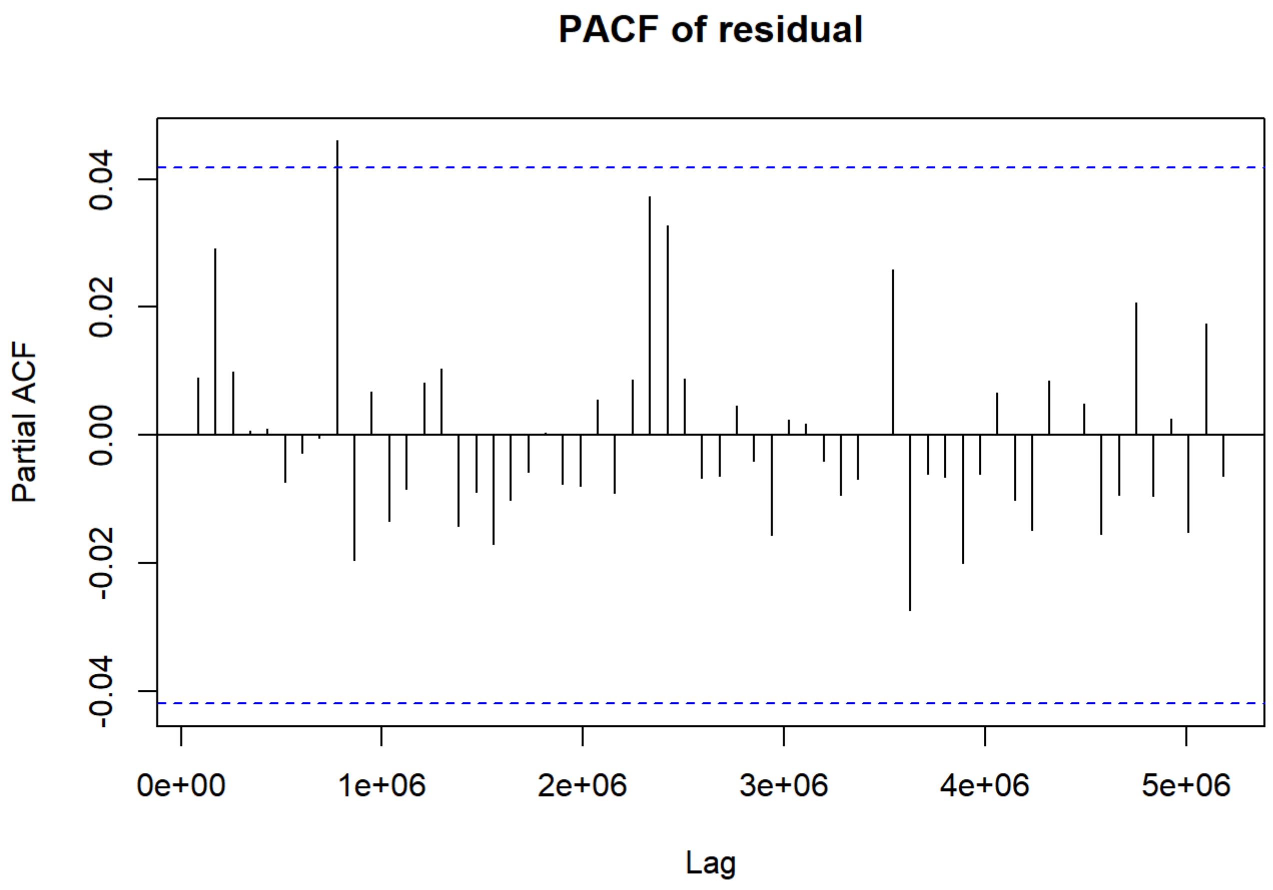
```
resid_norm = residuals(garch_fit)
qqnorm(resid_norm, main='QQ plot of residuals from GARCH ')
qqline(resid_norm)
```



```
acf(resid_norm, lag.max = 60, main='ACF of residual')
```



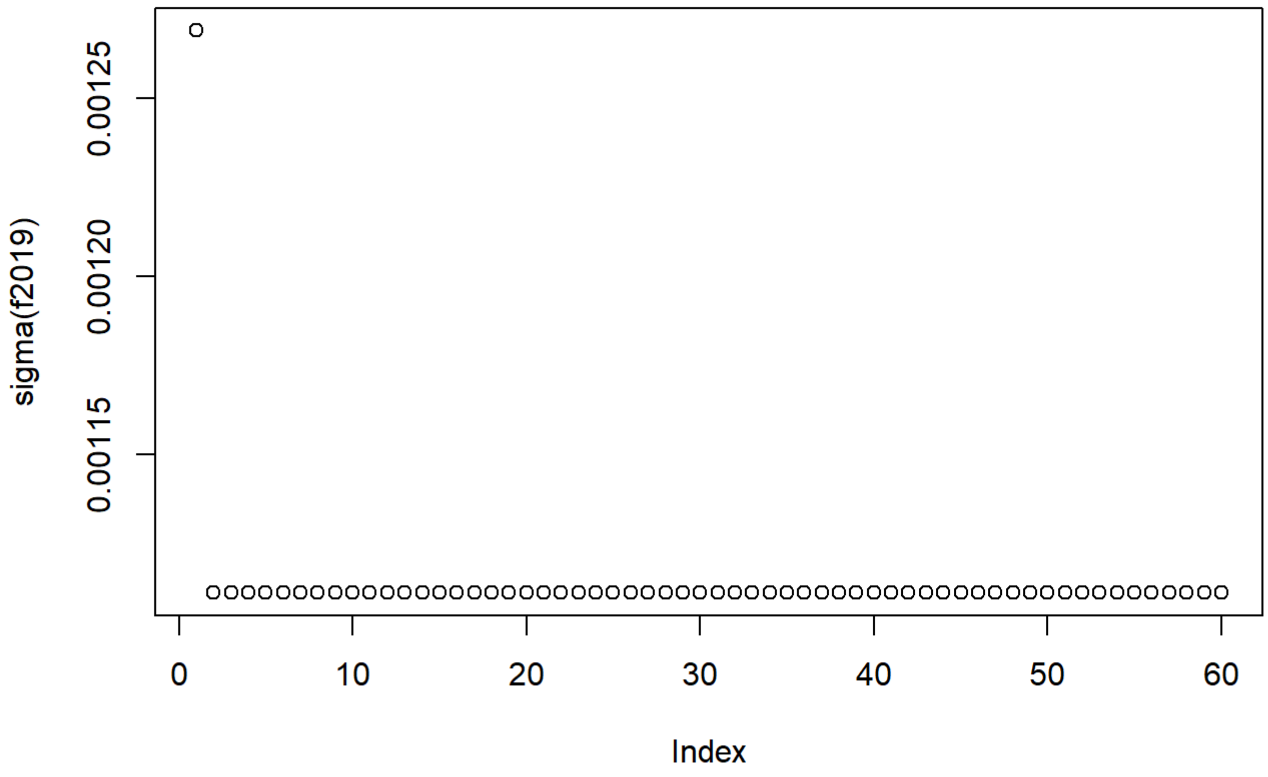
```
pacf(resid_norm, lag.max = 60, main='PACF of residual')
```



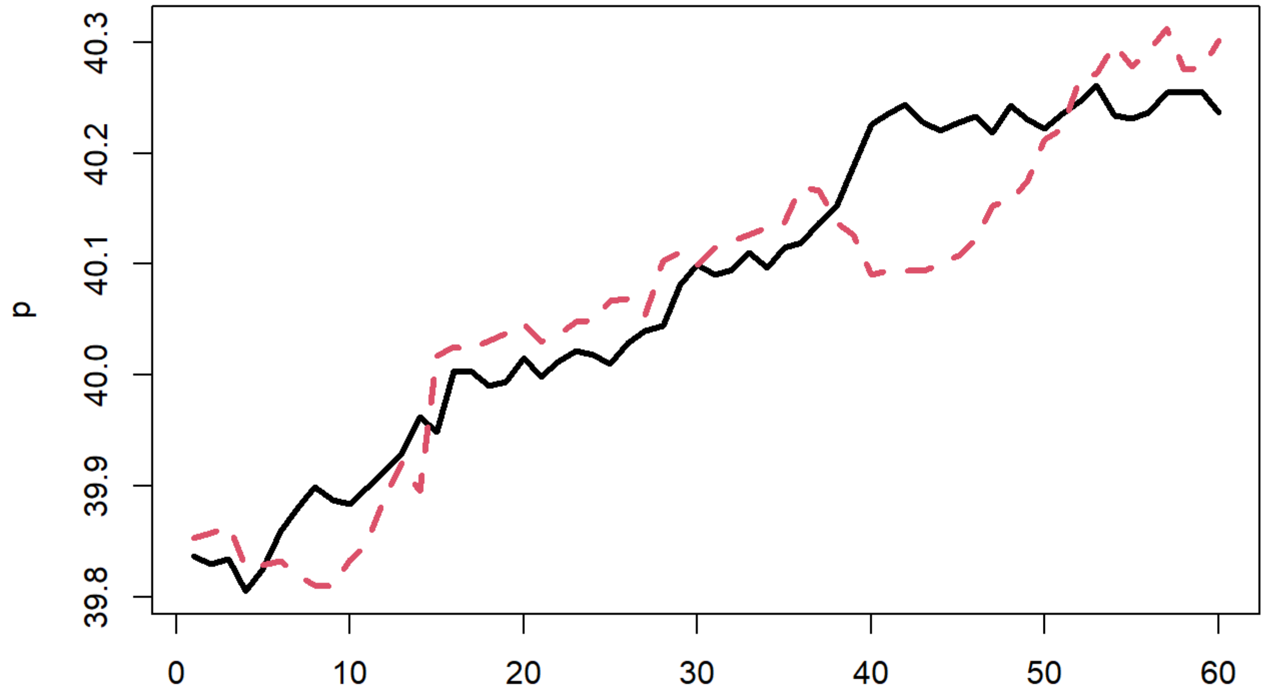
```
sfinal <- garch_spec
setfixed(sfinal) <- as.list(coef(garch_fit))
coef(garch_fit)
```

```
##            mu           ar1           ar2           ma1           ma2 
##  1.755705e-04  2.990511e-01  1.339080e-01 -2.538174e-01 -1.284631e-01 
##         omega        alpha1         beta1        gamma1         shape 
##  1.234701e-06  2.556718e-01  2.414991e-01 -9.942940e-01  2.100000e+00
```

```
f2019 <- ugarchforecast(data = return, fitORspec = sfinal, n.ahead = 60)
plot(sigma(f2019))
```



```
# The plot shows variations in forecasted volatility over time, although there's a concentrat
ion of points at a certain level of volatility, which could suggest that the model forecasts
a relatively stable volatility regime around that level for most of the time period shown.
#There are outliers which represent points in time where the forecasted volatility is signifi
cantly different from the rest of the period - notably higher in a couple of instances and lo
wer in one
sim <- ugarchpath(spec = sfinal, m.sim = 2, n.sim = 1*60, rseed = 123)
p <- 39.84*apply(fitted(sim), 2, 'cumsum') + 39.84
matplot(p, type = "l", lwd = 3)
```



Explanation for the above forecasting

The specifics of what ‘P’ stands for would depend on the actual financial series being analyzed. It could be the price of an asset, the returns on an asset, or another related financial metric. The plot appears to show that the variable exhibits increasing variance over time, which might imply that the market’s perception of risk or uncertainty regarding ‘P’ is growing

Explanation of the parameters used in the GARCH models

Conditional Variance Dynamics:

1.GARCH Model: The GARCH order specifies the number of lag terms for both ARCH and GARCH components. It’s often determined through model selection techniques, aiming for a balance between model complexity and goodness of fit.

2.Mean Model: The ARMA order specifies the number of autoregressive and moving average terms. Again, this is determined through model selection.

3.Distribution: The choice of distribution depends on the nature of the data. ‘std’ usually means standard normal distribution.

4.Estimate: These are the parameter estimates. They indicate the strength and direction of the relationship between variables.

5.Standard Error: Indicates the precision of the estimate. Smaller is better.

5.t-value: Indicates how many standard deviations the estimate is from zero. Larger (in absolute terms) is better.

6.p-value: Indicates the significance of the estimate. Smaller is better. Generally, a p-value less than 0.05 suggests the parameter is significant.

7.Robust Standard Errors: Similar to standard errors but robust to heteroskedasticity. They're more reliable when there's heteroskedasticity in the data.

8.Log-Likelihood: Measures how well the model fits the data. Higher values are better.

9.Information Criteria: Lower values are better. AIC, BIC, Hannan-Quinn, etc., are used to compare models. Lower values indicate a better trade-off between model fit and complexity.

10.Weighted Ljung-Box Test: Tests for serial correlation in the residuals. Significant p-values suggest the presence of serial correlation, which indicates that the model may not capture all the temporal dependencies in the data.

11.Weighted ARCH LM Tests: Tests for the presence of ARCH effects. Non-significant p-values suggest no ARCH effects. A non-significant p-value indicates that the model has adequately captured the volatility clustering in the data.

12.Nyblom Stability Test: Tests for parameter stability. Significant p-values suggest parameter instability. If the p-values are significant, it might indicate that the model parameters change over time, which could affect the reliability of the model.

13.Sign Bias Test: Tests for the presence of sign bias in the model residuals. Non-significant p-values indicate that the model residuals are not biased. A significant p-value might indicate that the model fails to capture some important aspect of the data.

14.Adjusted Pearson Goodness-of-Fit Test: Tests for the overall goodness of fit of the model. Lower values indicate better fit. Significant p-values indicate poor fit.

Conclusion for the project

1. In seasonal time series, I was able to learn and compare important parameters like AIC, BIC, Log Likelihood to find the best SARIMA model. SARIMA(5,1,0) with seasonal order 3 was the best model achieved.
2. Another important aspect is how to correctly read graphs of ACF, PACF and Residual Analysis. These help to predict accurate forecasting for seasonal period that we want. Shapiro-Wilk test and Ljung-Box tests are the most reliable tests that I personally prefer.Forecast for next 12 months is predicted.
3. In non-seasonal time series, ARIMA model is not suitable for a financial time series. After plotting several combinations and their residuals, forecasting predicts a flat line. Garch model captures dependencies and predicts a better forecasting but again is not promising for a stock market data point of view. I tried different combinations of models along with ARIFMA. F stands for fractional differencing but F=0 was the most appropriate result that I was achieving. Then finally tried GJR-Garch model which has an extra parameter which captures volatility as well. Also in garch models, absolute and sqaure returns are calculated to capture magnitude and their positive and negative shocks. Hence the behavior of overall performance is recognized well enough to make further predictions.
4. GJR-GARCH(1,1) X ARIFMA(2,0,2) was the best model with highest log likelihood, best S.E values, lowest p-values and better t-values. The last graph of the forecasting was achieved. I have included only the models which gave performance and results.

Links for datasets

1. https://github.com/astonglen/Time-Series-Analysis-Non-Seasonal/blob/main/twitter-stocks.csv (https://github.com/astonglen/Time-Series-Analysis-Non-Seasonal/blob/main/twitter-stocks.csv)
2. https://github.com/astonglen/Time-Series-Analysis-Seasonal/blob/main/HotelEmployees.csv (https://github.com/astonglen/Time-Series-Analysis-Seasonal/blob/main/HotelEmployees.csv)

REFERENCES

1. Safari-Katesari, H., Samadi, S. Y., & Zaroudi, S. (2020). Modelling count data via copulas. Statistics, 54(6), 1329-1355.

2. Safari-Katesari, H., & Zaroudi, S. (2020). Count copula regression model using generalized beta distribution of the second kind. Statistics, 21, 1-12.

3. Safari-Katesari, H., & Zaroudi, S. (2021). Analysing the impact of dependency on conditional survival functions using copulas. Statistics in Transition New Series, 22(1).

4. Safari Katesari, H., (2021) Bayesian dynamic factor analysis and copula-based models for mixed data, PhD dissertation, Southern Illinois University Carbondale.

5. Zaroudi, S., Faridrohani, M. R., Behzadi, M. H., & Safari-Katesari, H. (2022). Copula-based Modeling for IBNR Claim Loss Reserving. arXiv preprint arXiv:2203.12750.