US. Unemployment rate forecast using Time Series

Bhargavi Katta
Department of Mathematical Sciences, Stevens Institute of Technology, Hoboken, NJ
Project Supervisor: Dr. Hadi Safari Katesar

Abstract

This project aims to forecast the US unemployment rate using time series analysis techniques. We collected monthly data on the unemployment rate from January 2000 to March 2022 and applied a variety of methods to model and forecast the series. Specifically, we explored the use of univariate models, such. as. AARIMA and also tested for stationarity, autocorrelation, and other statistical properties of the data to inform our modeling choices.

Non Seasonal DATASET: US Unemployment rate

Data Source:" https://tradingeconomics.com/united-states/unemployment-rate (https://tradingeconomics.com/united-states/unemployment-rate) "

Data Description: Data has unemployment rates of US for every month from 2000 to 2022

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

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

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
setwd("/Users/gopalrao000/Desktop/Spring-2023/Time series/project")
```

```
data=read.csv("seas unemployment.csv")
```

Data

```
##     Year Month Value
## 1   2000     1   4.0
## 2   2000     2   4.1
## 3   2000     3   4.0
## 4   2000     4   3.8
## 5   2000     5   4.0
## 6   2000     6   4.0
## 7   2000     7   4.0
## 8   2000     8   4.1
## 9   2000     9   3.9
## 10  2000    10   3.9
## 11  2000    11   3.9
## 12  2000    12   3.9
## 13  2001     1   4.2
## 14  2001     2   4.2
## 15  2001     3   4.3
## 16  2001     4   4.4
## 17  2001     5   4.3
## 18  2001     6   4.5
## 19  2001     7   4.6
```

```
## 20  2001     8   4.9
## 21  2001     9   5.0
## 22  2001    10   5.3
## 23  2001    11   5.5
## 24  2001    12   5.7
## 25  2002     1   5.7
## 26  2002     2   5.7
## 27  2002     3   5.7
## 28  2002     4   5.9
## 29  2002     5   5.8
## 30  2002     6   5.8
## 31  2002     7   5.8
## 32  2002     8   5.7
## 33  2002     9   5.7
## 34  2002    10   5.7
## 35  2002    11   5.9
## 36  2002    12   6.0
## 37  2003     1   5.8
## 38  2003     2   5.9
## 39  2003     3   5.9
## 40  2003     4   6.0
## 41  2003     5   6.1
## 42  2003     6   6.3
## 43  2003     7   6.2
## 44  2003     8   6.1
## 45  2003     9   6.1
## 46  2003    10   6.0
## 47  2003    11   5.8
## 48  2003    12   5.7
## 49  2004     1   5.7
## 50  2004     2   5.6
## 51  2004     3   5.8
## 52  2004     4   5.6
## 53  2004     5   5.6
## 54  2004     6   5.6
## 55  2004     7   5.5
## 56  2004     8   5.4
## 57  2004     9   5.4
## 58  2004    10   5.5
## 59  2004    11   5.4
## 60  2004    12   5.4
## 61  2005     1   5.3
## 62  2005     2   5.4
## 63  2005     3   5.2
## 64  2005     4   5.2
## 65  2005     5   5.1
## 66  2005     6   5.0
```

```
## 67  2005     7   5.0
## 68  2005     8   4.9
## 69  2005     9   5.0
## 70  2005    10   5.0
## 71  2005    11   5.0
## 72  2005    12   4.9
## 73  2006     1   4.7
## 74  2006     2   4.8
## 75  2006     3   4.7
## 76  2006     4   4.7
## 77  2006     5   4.6
## 78  2006     6   4.6
## 79  2006     7   4.7
## 80  2006     8   4.7
## 81  2006     9   4.5
## 82  2006    10   4.4
## 83  2006    11   4.5
## 84  2006    12   4.4
## 85  2007     1   4.6
## 86  2007     2   4.5
## 87  2007     3   4.4
## 88  2007     4   4.5
## 89  2007     5   4.4
## 90  2007     6   4.6
## 91  2007     7   4.7
## 92  2007     8   4.6
## 93  2007     9   4.7
## 94  2007    10   4.7
## 95  2007    11   4.7
## 96  2007    12   5.0
## 97  2008     1   5.0
## 98  2008     2   4.9
## 99  2008     3   5.1
## 100 2008     4   5.0
## 101 2008     5   5.4
## 102 2008     6   5.6
## 103 2008     7   5.8
## 104 2008     8   6.1
## 105 2008     9   6.1
## 106 2008    10   6.5
## 107 2008    11   6.8
## 108 2008    12   7.3
## 109 2009     1   7.8
## 110 2009     2   8.3
## 111 2009     3   8.7
## 112 2009     4   9.0
## 113 2009     5   9.4
```

```
## 114 2009     6   9.5
## 115 2009     7   9.5
## 116 2009     8   9.6
## 117 2009     9   9.8
## 118 2009    10  10.0
## 119 2009    11   9.9
## 120 2009    12   9.9
## 121 2010     1   9.8
## 122 2010     2   9.8
## 123 2010     3   9.9
## 124 2010     4   9.9
## 125 2010     5   9.6
## 126 2010     6   9.4
## 127 2010     7   9.4
## 128 2010     8   9.5
## 129 2010     9   9.5
## 130 2010    10   9.4
## 131 2010    11   9.8
## 132 2010    12   9.3
## 133 2011     1   9.1
## 134 2011     2   9.0
## 135 2011     3   9.0
## 136 2011     4   9.1
## 137 2011     5   9.0
## 138 2011     6   9.1
## 139 2011     7   9.0
## 140 2011     8   9.0
## 141 2011     9   9.0
## 142 2011    10   8.8
## 143 2011    11   8.6
## 144 2011    12   8.5
## 145 2012     1   8.3
## 146 2012     2   8.3
## 147 2012     3   8.2
## 148 2012     4   8.2
## 149 2012     5   8.2
## 150 2012     6   8.2
## 151 2012     7   8.2
## 152 2012     8   8.1
## 153 2012     9   7.8
## 154 2012    10   7.8
## 155 2012    11   7.7
## 156 2012    12   7.9
## 157 2013     1   8.0
## 158 2013     2   7.7
## 159 2013     3   7.5
## 160 2013     4   7.6
```

```
## 161 2013     5   7.5
## 162 2013     6   7.5
## 163 2013     7   7.3
## 164 2013     8   7.2
## 165 2013     9   7.2
## 166 2013    10   7.2
## 167 2013    11   6.9
## 168 2013    12   6.7
## 169 2014     1   6.6
## 170 2014     2   6.7
## 171 2014     3   6.7
## 172 2014     4   6.2
## 173 2014     5   6.3
## 174 2014     6   6.1
## 175 2014     7   6.2
## 176 2014     8   6.1
## 177 2014     9   5.9
## 178 2014    10   5.7
## 179 2014    11   5.8
## 180 2014    12   5.6
## 181 2015     1   5.7
## 182 2015     2   5.5
## 183 2015     3   5.4
## 184 2015     4   5.4
## 185 2015     5   5.6
## 186 2015     6   5.3
## 187 2015     7   5.2
## 188 2015     8   5.1
## 189 2015     9   5.0
## 190 2015    10   5.0
## 191 2015    11   5.1
## 192 2015    12   5.0
## 193 2016     1   4.8
## 194 2016     2   4.9
## 195 2016     3   5.0
## 196 2016     4   5.1
## 197 2016     5   4.8
## 198 2016     6   4.9
## 199 2016     7   4.8
## 200 2016     8   4.9
## 201 2016     9   5.0
## 202 2016    10   4.9
## 203 2016    11   4.7
## 204 2016    12   4.7
## 205 2017     1   4.7
## 206 2017     2   4.6
## 207 2017     3   4.4
```

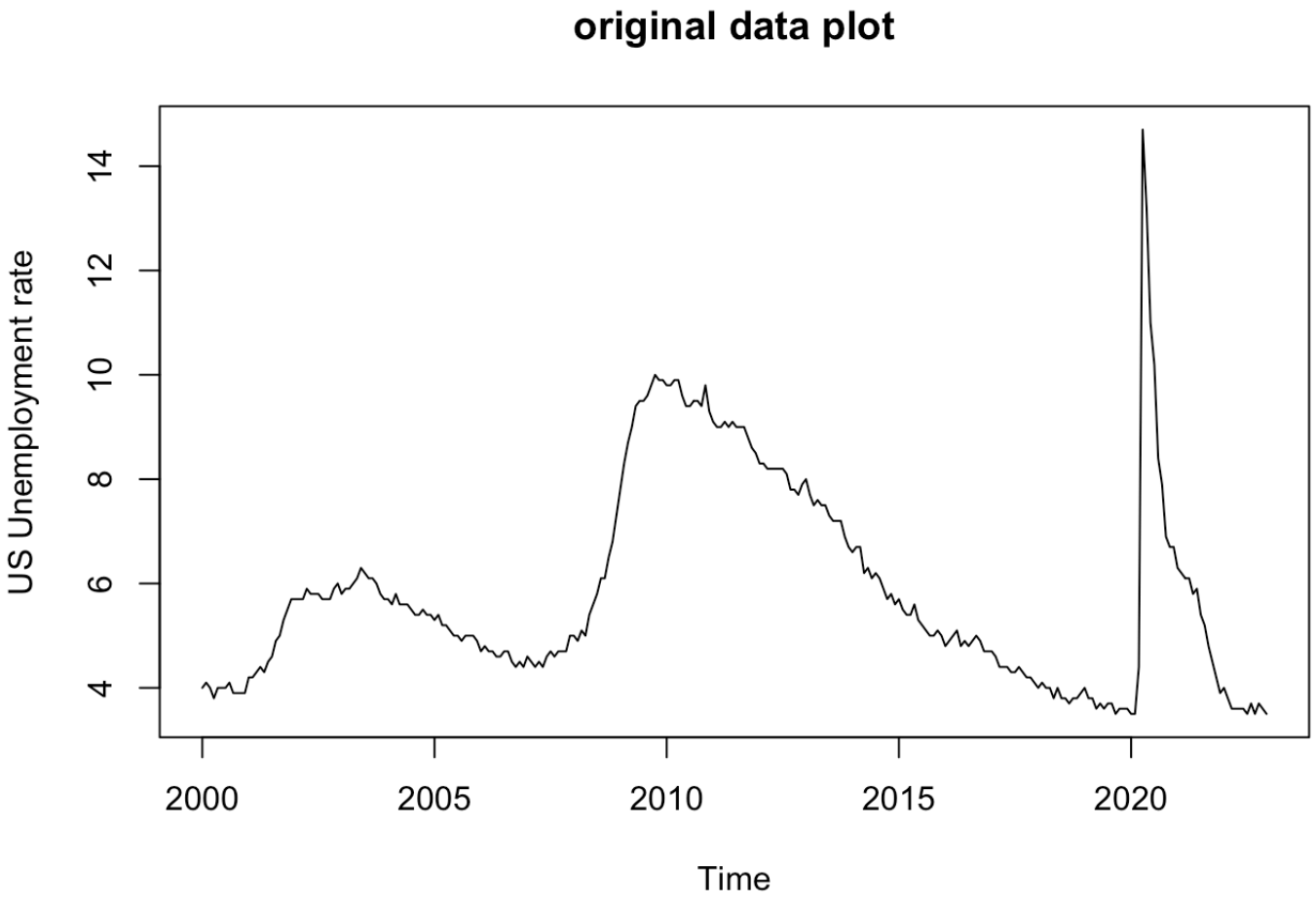
```
## 208 2017     4   4.4
## 209 2017     5   4.4
## 210 2017     6   4.3
## 211 2017     7   4.3
## 212 2017     8   4.4
## 213 2017     9   4.3
## 214 2017    10   4.2
## 215 2017    11   4.2
## 216 2017    12   4.1
## 217 2018     1   4.0
## 218 2018     2   4.1
## 219 2018     3   4.0
## 220 2018     4   4.0
## 221 2018     5   3.8
## 222 2018     6   4.0
## 223 2018     7   3.8
## 224 2018     8   3.8
## 225 2018     9   3.7
## 226 2018    10   3.8
## 227 2018    11   3.8
## 228 2018    12   3.9
## 229 2019     1   4.0
## 230 2019     2   3.8
## 231 2019     3   3.8
## 232 2019     4   3.6
## 233 2019     5   3.7
## 234 2019     6   3.6
## 235 2019     7   3.7
## 236 2019     8   3.7
## 237 2019     9   3.5
## 238 2019    10   3.6
## 239 2019    11   3.6
## 240 2019    12   3.6
## 241 2020     1   3.5
## 242 2020     2   3.5
## 243 2020     3   4.4
## 244 2020     4  14.7
## 245 2020     5  13.2
## 246 2020     6  11.0
## 247 2020     7  10.2
## 248 2020     8   8.4
## 249 2020     9   7.9
## 250 2020    10   6.9
## 251 2020    11   6.7
## 252 2020    12   6.7
## 253 2021     1   6.3
## 254 2021     2   6.2
```

```
## 255 2021     3   6.1
## 256 2021     4   6.1
## 257 2021     5   5.8
## 258 2021     6   5.9
## 259 2021     7   5.4
## 260 2021     8   5.2
## 261 2021     9   4.8
## 262 2021    10   4.5
## 263 2021    11   4.2
## 264 2021    12   3.9
## 265 2022     1   4.0
## 266 2022     2   3.8
## 267 2022     3   3.6
## 268 2022     4   3.6
## 269 2022     5   3.6
## 270 2022     6   3.6
## 271 2022     7   3.5
## 272 2022     8   3.7
## 273 2022     9   3.5
## 274 2022    10   3.7
## 275 2022    11   3.6
## 276 2022    12   3.5
## 277 2023     1   3.4
## 278 2023     2   3.6
## 279 2023     3   3.5
```

```
data.ts <- ts(data$Value,start=c(2000,1), end=c(2022,12), frequency=12)
data.ts
```

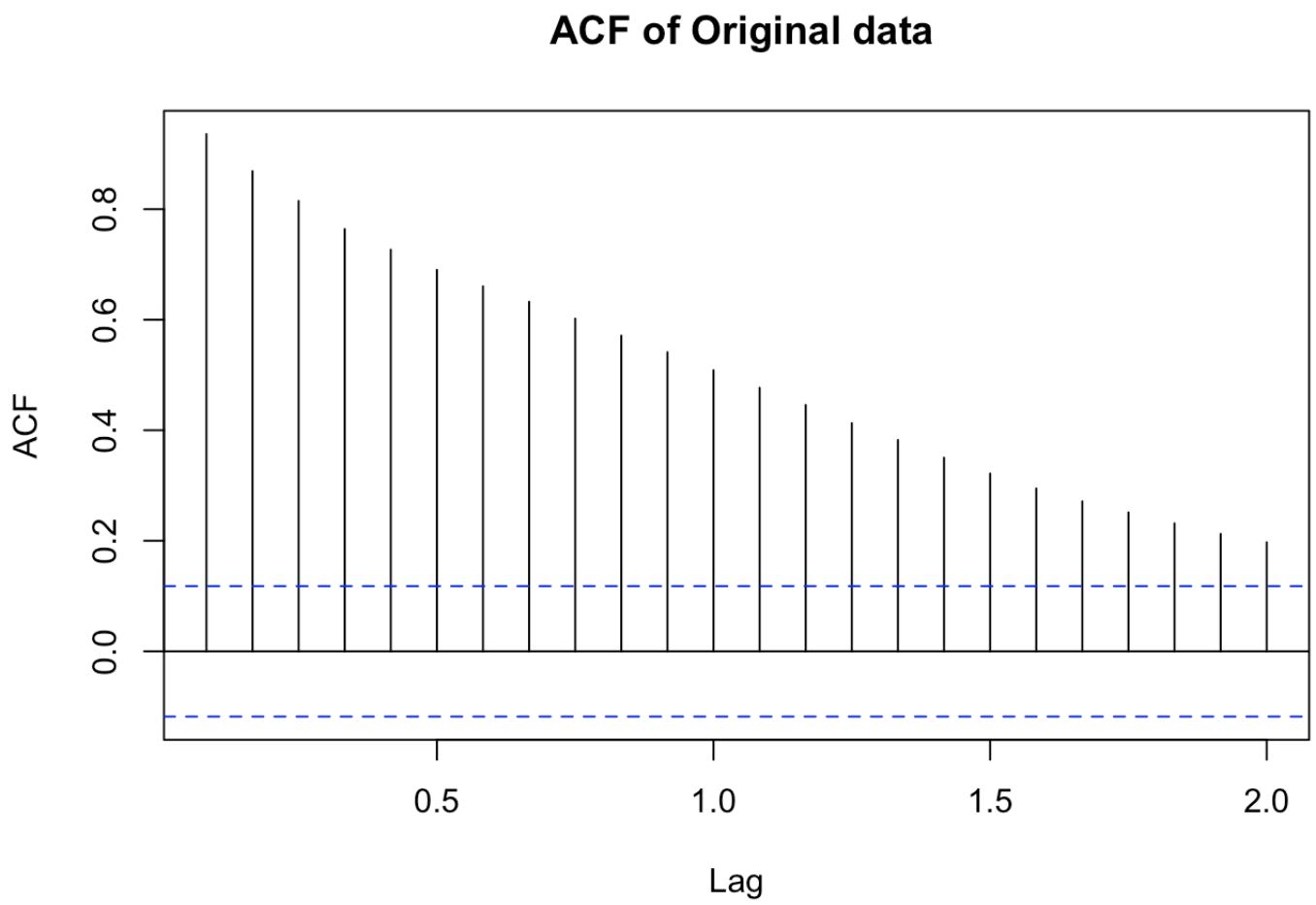
```
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 2000  4.0  4.1  4.0  3.8  4.0  4.0  4.0  4.1  3.9  3.9  3.9  3.9
## 2001  4.2  4.2  4.3  4.4  4.3  4.5  4.6  4.9  5.0  5.3  5.5  5.7
## 2002  5.7  5.7  5.7  5.9  5.8  5.8  5.8  5.7  5.7  5.7  5.9  6.0
## 2003  5.8  5.9  5.9  6.0  6.1  6.3  6.2  6.1  6.1  6.0  5.8  5.7
## 2004  5.7  5.6  5.8  5.6  5.6  5.6  5.5  5.4  5.4  5.5  5.4  5.4
## 2005  5.3  5.4  5.2  5.2  5.1  5.0  5.0  4.9  5.0  5.0  5.0  4.9
## 2006  4.7  4.8  4.7  4.7  4.6  4.6  4.7  4.7  4.5  4.4  4.5  4.4
## 2007  4.6  4.5  4.4  4.5  4.4  4.6  4.7  4.6  4.7  4.7  4.7  5.0
## 2008  5.0  4.9  5.1  5.0  5.4  5.6  5.8  6.1  6.1  6.5  6.8  7.3
## 2009  7.8  8.3  8.7  9.0  9.4  9.5  9.5  9.6  9.8 10.0  9.9  9.9
## 2010  9.8  9.8  9.9  9.9  9.6  9.4  9.4  9.5  9.5  9.4  9.8  9.3
## 2011  9.1  9.0  9.0  9.1  9.0  9.1  9.0  9.0  9.0  8.8  8.6  8.5
## 2012  8.3  8.3  8.2  8.2  8.2  8.2  8.2  8.1  7.8  7.8  7.7  7.9
## 2013  8.0  7.7  7.5  7.6  7.5  7.5  7.3  7.2  7.2  7.2  6.9  6.7
## 2014  6.6  6.7  6.7  6.2  6.3  6.1  6.2  6.1  5.9  5.7  5.8  5.6
## 2015  5.7  5.5  5.4  5.4  5.6  5.3  5.2  5.1  5.0  5.0  5.1  5.0
## 2016  4.8  4.9  5.0  5.1  4.8  4.9  4.8  4.9  5.0  4.9  4.7  4.7
## 2017  4.7  4.6  4.4  4.4  4.4  4.3  4.3  4.4  4.3  4.2  4.2  4.1
## 2018  4.0  4.1  4.0  4.0  3.8  4.0  3.8  3.8  3.7  3.8  3.8  3.9
## 2019  4.0  3.8  3.8  3.6  3.7  3.6  3.7  3.7  3.5  3.6  3.6  3.6
## 2020  3.5  3.5  4.4 14.7 13.2 11.0 10.2  8.4  7.9  6.9  6.7  6.7
## 2021  6.3  6.2  6.1  6.1  5.8  5.9  5.4  5.2  4.8  4.5  4.2  3.9
## 2022  4.0  3.8  3.6  3.6  3.6  3.6  3.5  3.7  3.5  3.7  3.6  3.5
```

```
plot(data.ts,ylab="US Unemployment rate", main="original data plot")
```



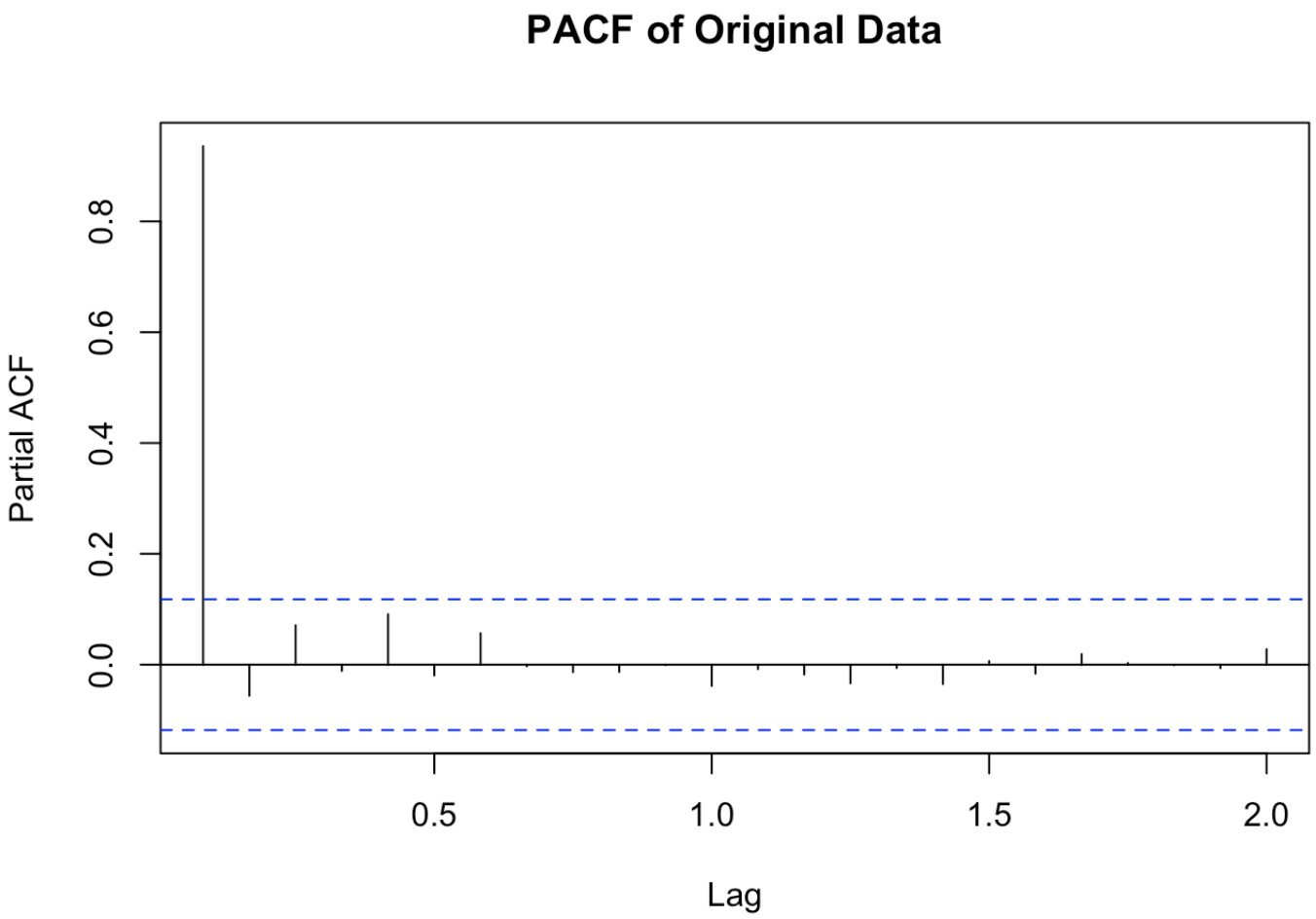
ACF Plot

```
acf(data.ts,main=" ACF of Original data")
```



PACF Plot

```
pacf(data.ts, main=" PACF of Original Data ")
```



Staionary Test

```
adf.test(data.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data.ts
## Dickey-Fuller = -2.2171, Lag order = 6, p-value = 0.4846
## alternative hypothesis: stationary
```

Null Hypothesis: Data is not stationary Alternate hypothesis: Data is Stationary P. value > 0.05, failed to reject null hypothesis Hence data is not stationary

```
ndiffs(data.ts)
```

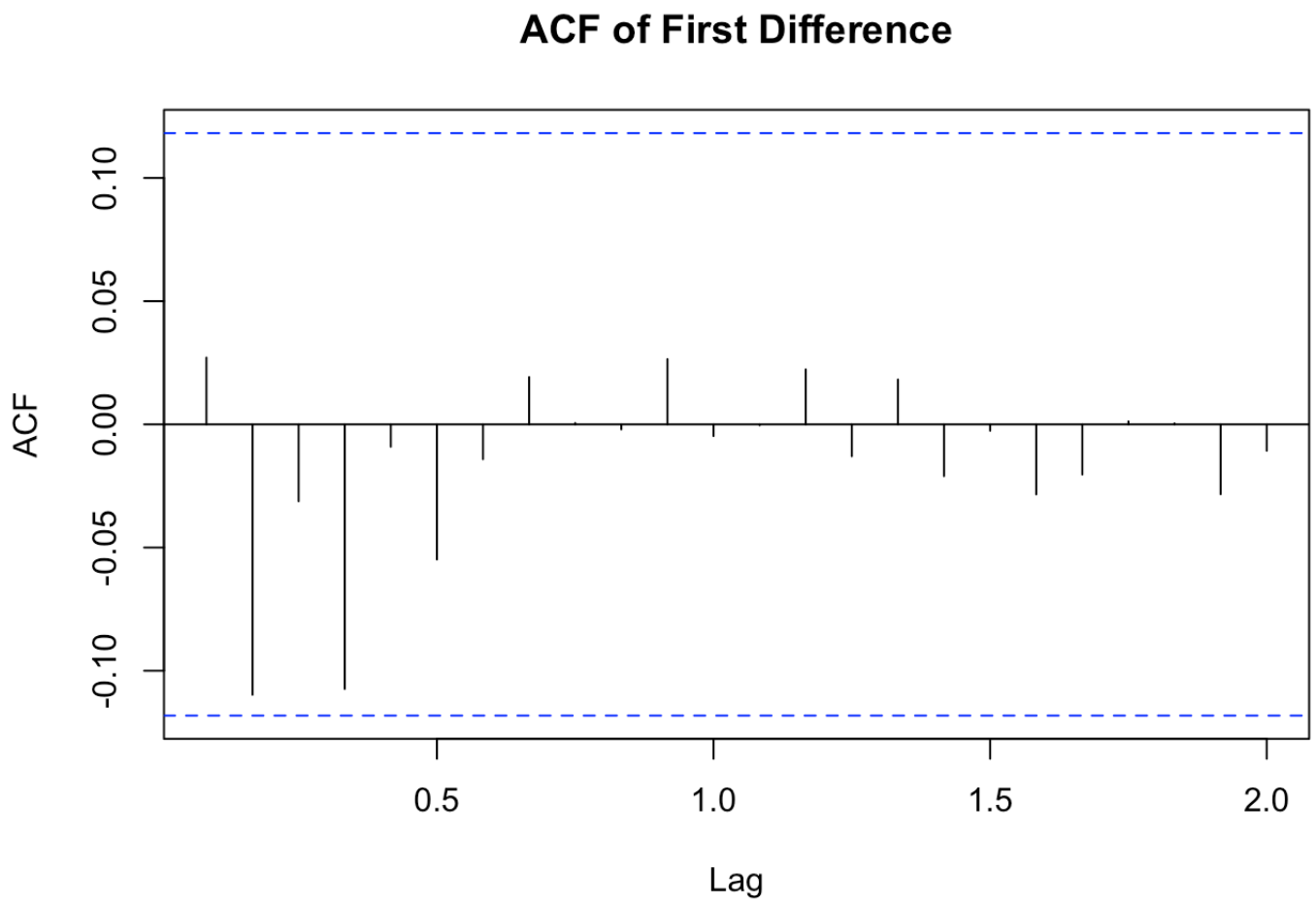
```
## [1] 1
```

```
nsdiffs(data.ts)
```

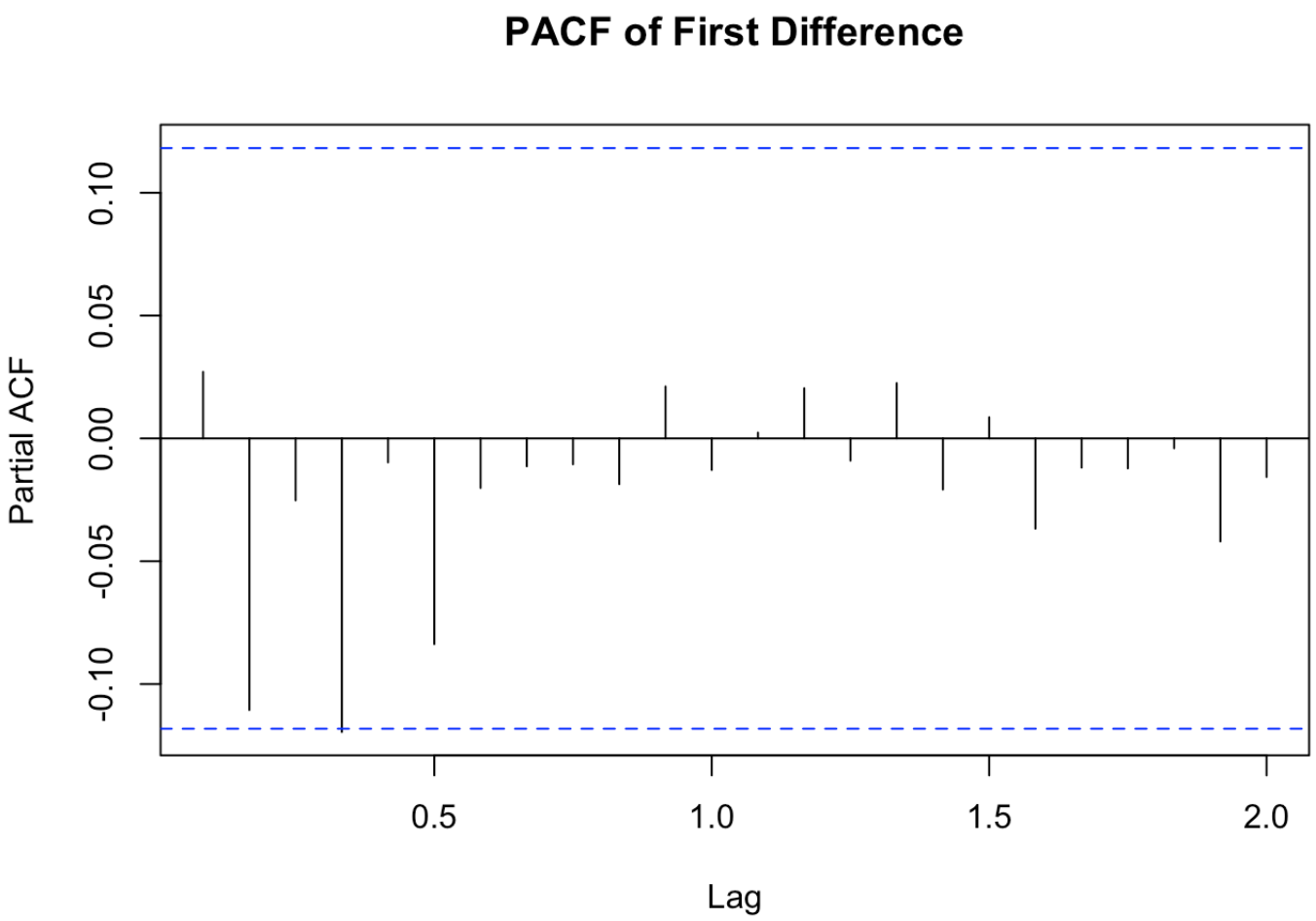
```
## [1] 0
```

Attempt to make the data Stationary

```
new_data <- diff(data.ts, differences = 1)
acf(new_data, main = "ACF of First Difference")
```




```
pacf(new_data, main = "PACF of First Difference")
```




```
adf.test(new_data)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  new_data
## Dickey-Fuller = -7.4355, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

After first differencing, data is stationary.

KPSS Test

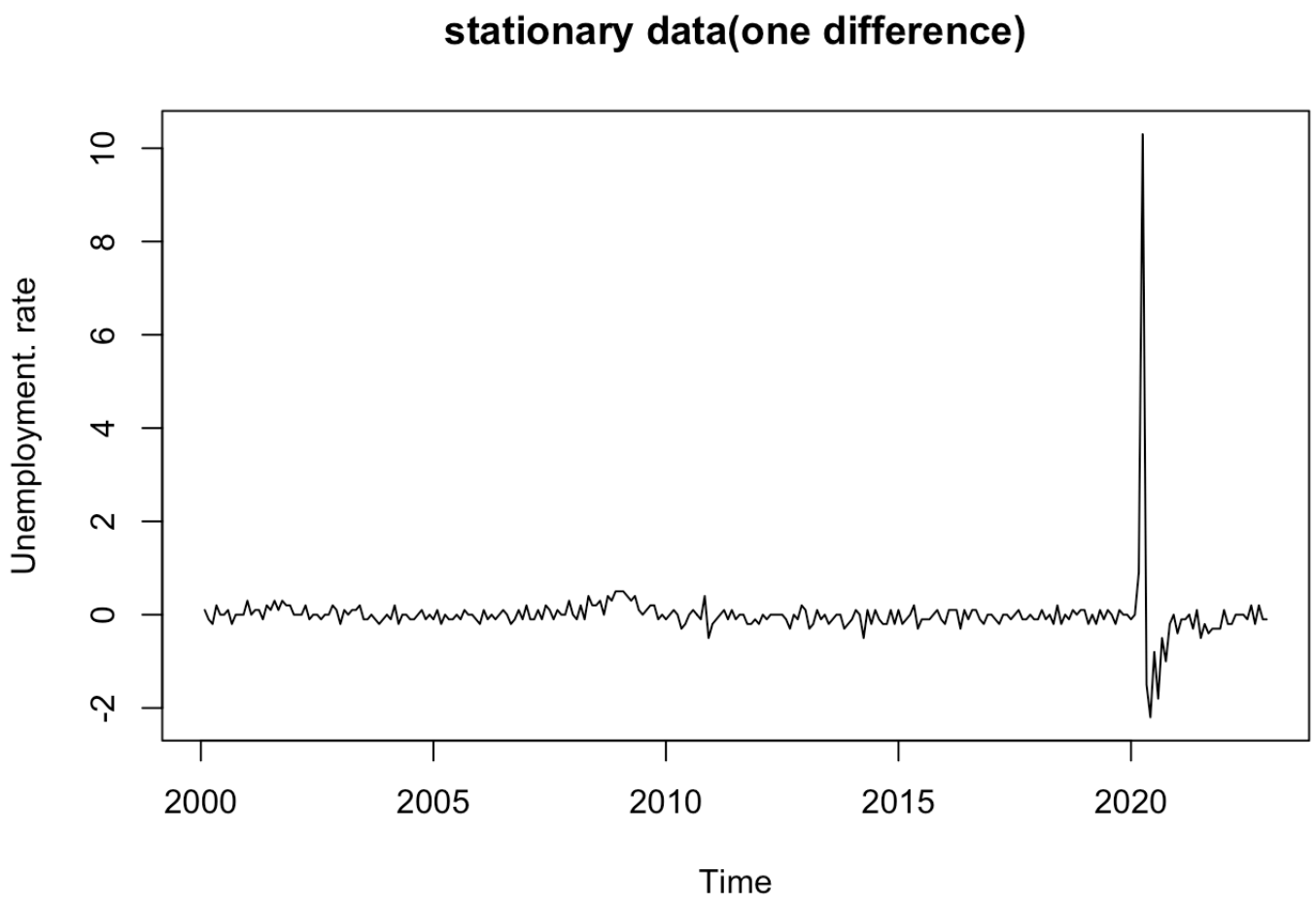
```
kpss.test(new_data)
```

```
## Warning in kpss.test(new_data): p-value greater than printed p-value
```

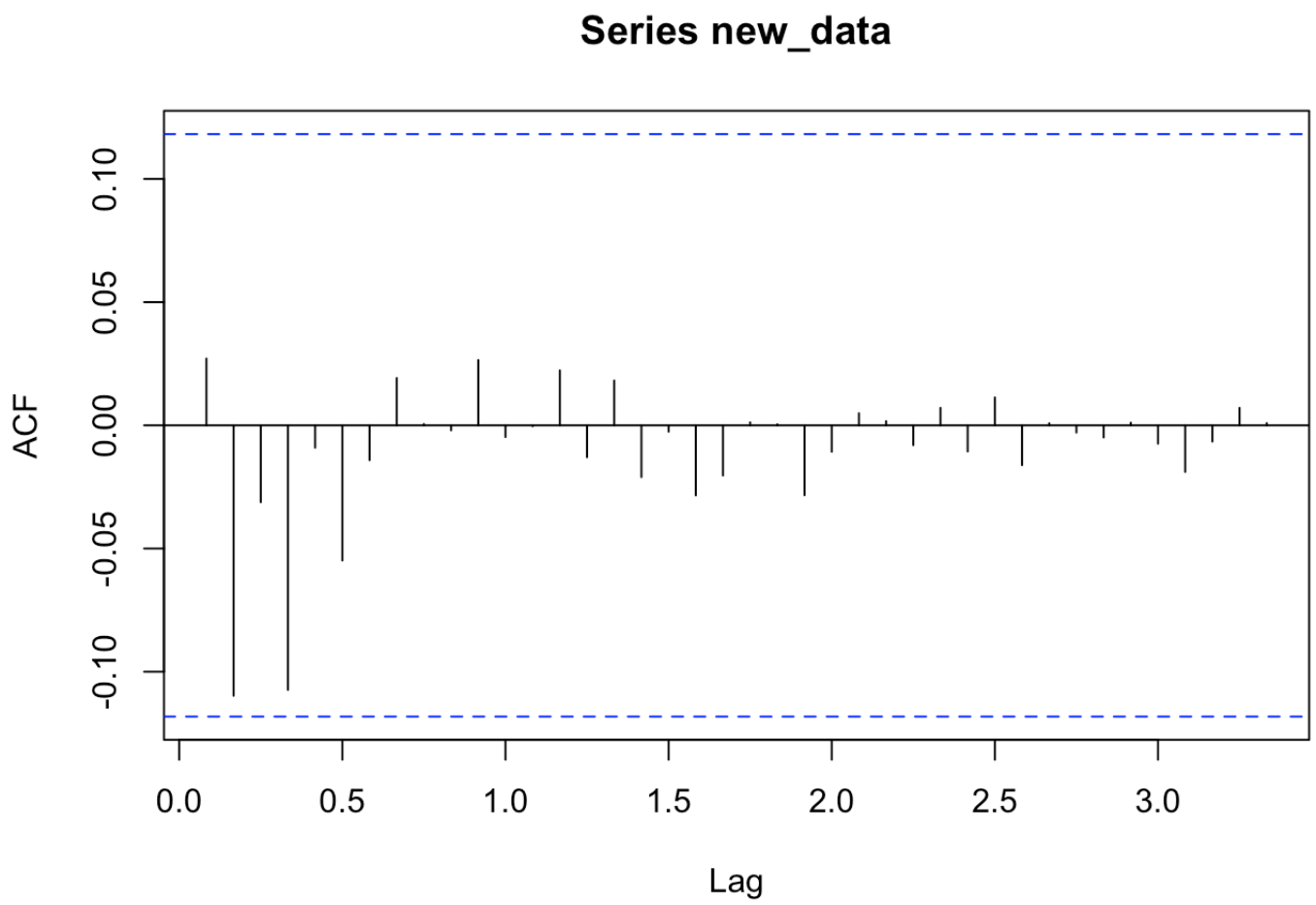
```
##
##  KPSS Test for Level Stationarity
##
## data:  new_data
## KPSS Level = 0.082695, Truncation lag parameter = 5, p-value = 0.1
```

Here as the p-value is greater than 0.05, the data is stationary.

```
plot(new_data ,ylab="Unemployment. rate", main="stationary data(one difference)")
```

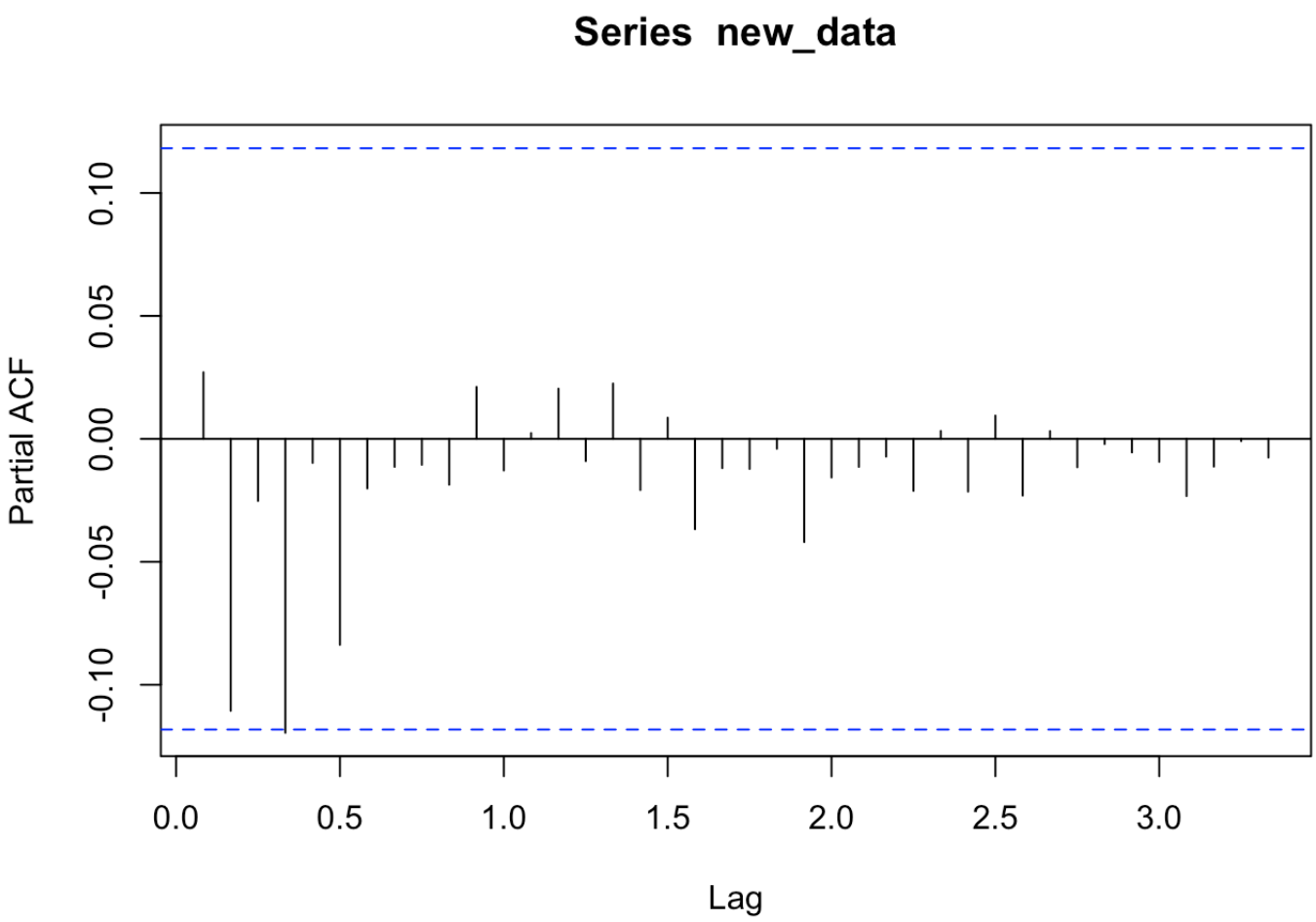



```
acf(new_data,lag.max=40)
```



Above ACF plot is white noise

```
pacf(new_data,lag.max=40)
```



Above PACF plot looks like almost whitenoise and 4th lag seems to have slight significance

```
eacf(new_data)
```

```
## 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  o  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 o x x o o o o o o o o  o  o  o
## 5 o x x x x o o o o o o  o  o  o
## 6 x o o x x o o o o o o  o  o  o
## 7 x o o o x o o o o o o  o  o  o
```

From the EACF plot possible beeter model is arma(0,0) but we will see how different models are capturing the patterns

```
fit <- auto.arima(new_data)
```

```
summary(fit)
```

```
## Series: new_data
## ARIMA(0,0,0) with zero mean
##
## sigma^2 = 0.4602:  log likelihood = -283.49
## AIC=568.98   AICc=568.99   BIC=572.6
##
## Training set error measures:
##                        ME     RMSE       MAE MPE MAPE      MASE      ACF1
## Training set -0.001818182 0.678367 0.1887273 100  100 0.6053082 0.0270969
```

Modeling

```
# load necessary libraries
library(forecast)
library(dplyr)

# create a dataframe to store the model results
model_results <- data.frame(model = character(), AIC = numeric(), BIC = numeric(), si
gma2 = numeric(), loglik = numeric())

# create and fit ARIMA models with different parameters
model1 <- arima(new_data, order = c(1,1,0))
model2 <- arima(new_data, order = c(0,1,1))
model3 <- arima(new_data, order = c(1,1,1))
model4 <- arima(new_data, order = c(2,1,1))
model5 <- arima(new_data, order = c(1,1,2))
model6 <- arima(new_data, order = c(2,1,2))


# store the results for each model in the dataframe
model_results <- model_results %>% add_row(model = "Model 1", AIC = AIC(model1), BIC
= BIC(model1), sigma2 = summary(model1)$sigma2, loglik = as.numeric(logLik(model1)))
model_results <- model_results %>% add_row(model = "Model 2", AIC = AIC(model2), BIC
= BIC(model2), sigma2 = summary(model2)$sigma2, loglik = as.numeric(logLik(model2)))
model_results <- model_results %>% add_row(model = "Model 3", AIC = AIC(model3), BIC
= BIC(model3), sigma2 = summary(model3)$sigma2, loglik = as.numeric(logLik(model3)))
model_results <- model_results %>% add_row(model = "Model 4", AIC = AIC(model4), BIC
= BIC(model4), sigma2 = summary(model4)$sigma2, loglik = as.numeric(logLik(model4)))
model_results <- model_results %>% add_row(model = "Model 5", AIC = AIC(model5), BIC
= BIC(model5), sigma2 =
summary(model5)$sigma2, loglik = as.numeric(logLik(model5)))
model_results <- model_results %>% add_row(model = "Model 6", AIC = AIC(model6), BIC
= BIC(model6), sigma2 = summary(model6)$sigma2, loglik = as.numeric(logLik(model6)))


# view the results
model_results
```

```
##     model      AIC      BIC    sigma2    loglik
## 1 Model 1 696.5734 703.7997 0.7327316 -346.2867
## 2 Model 2 575.5307 582.7569 0.4618582 -285.7653
## 3 Model 3 577.2718 588.1111 0.4615250 -285.6359
## 4 Model 4 576.1288 590.5813 0.4558861 -284.0644
## 5 Model 5 576.2267 590.6792 0.4564143 -284.1133
## 6 Model 6 577.4710 595.5366 0.4549605 -283.7355
```

```
model_results_sorted <- model_results %>% arrange(AIC)
# view the sorted results
model_results_sorted
```

```
##     model      AIC      BIC    sigma2    loglik
## 1 Model 2 575.5307 582.7569 0.4618582 -285.7653
## 2 Model 4 576.1288 590.5813 0.4558861 -284.0644
## 3 Model 5 576.2267 590.6792 0.4564143 -284.1133
## 4 Model 3 577.2718 588.1111 0.4615250 -285.6359
## 5 Model 6 577.4710 595.5366 0.4549605 -283.7355
## 6 Model 1 696.5734 703.7997 0.7327316 -346.2867
```

From the above, AIC and BIC values Model2 i.e arima(0,1,1) is the good for for the data and we can also test Model4 i.e(2,1,1) model

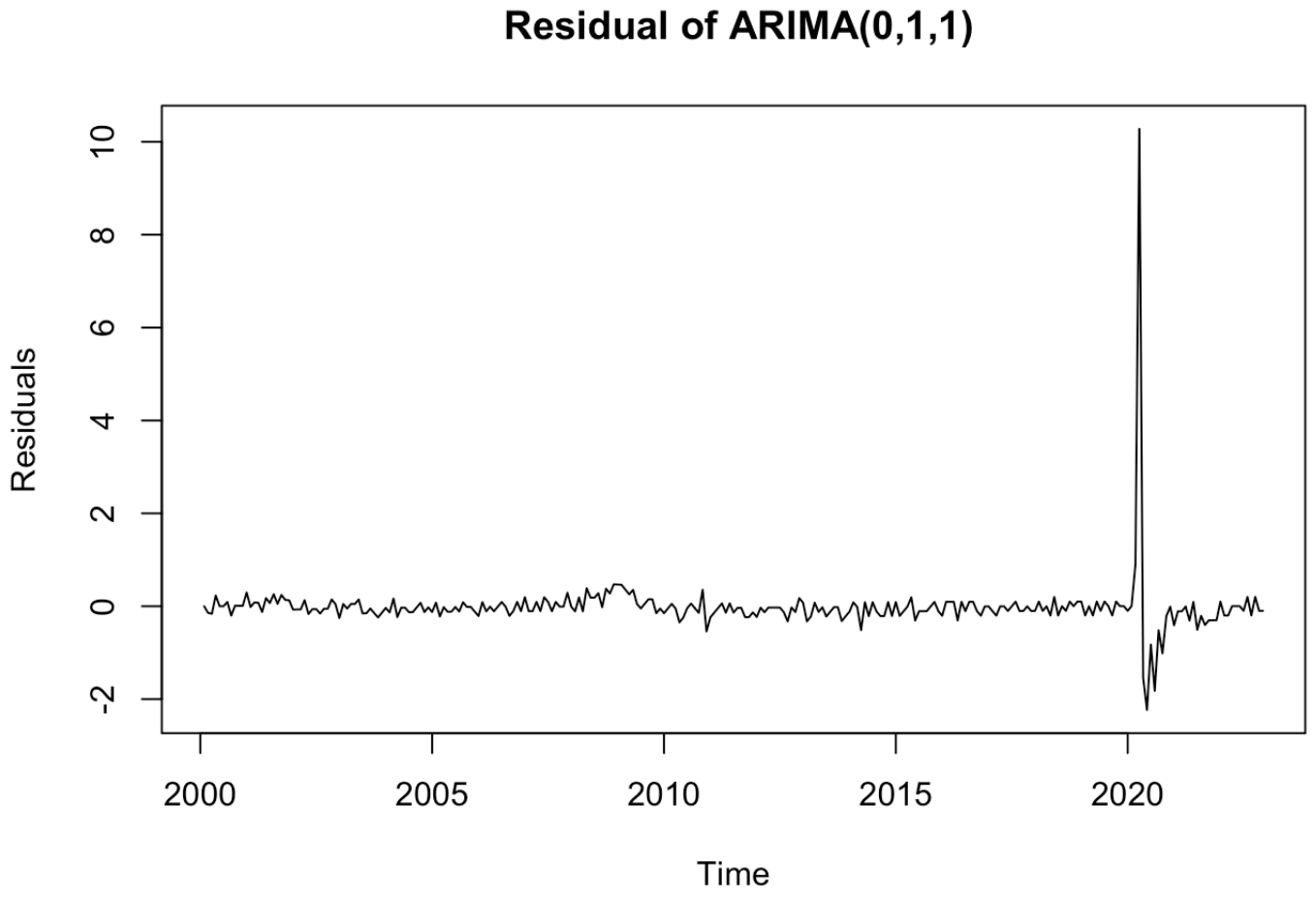
Arima(0,1,1)

```
model <- Arima(new_data, order = c(0, 1, 1))
```

Residual Analysis

```
residuals <- residuals(model)
```

```
plot(residuals, type = "l", xlab = "Time", ylab = "Residuals",main=" Residual of ARIM
A(0,1,1)")
```



Stationary test on residuals

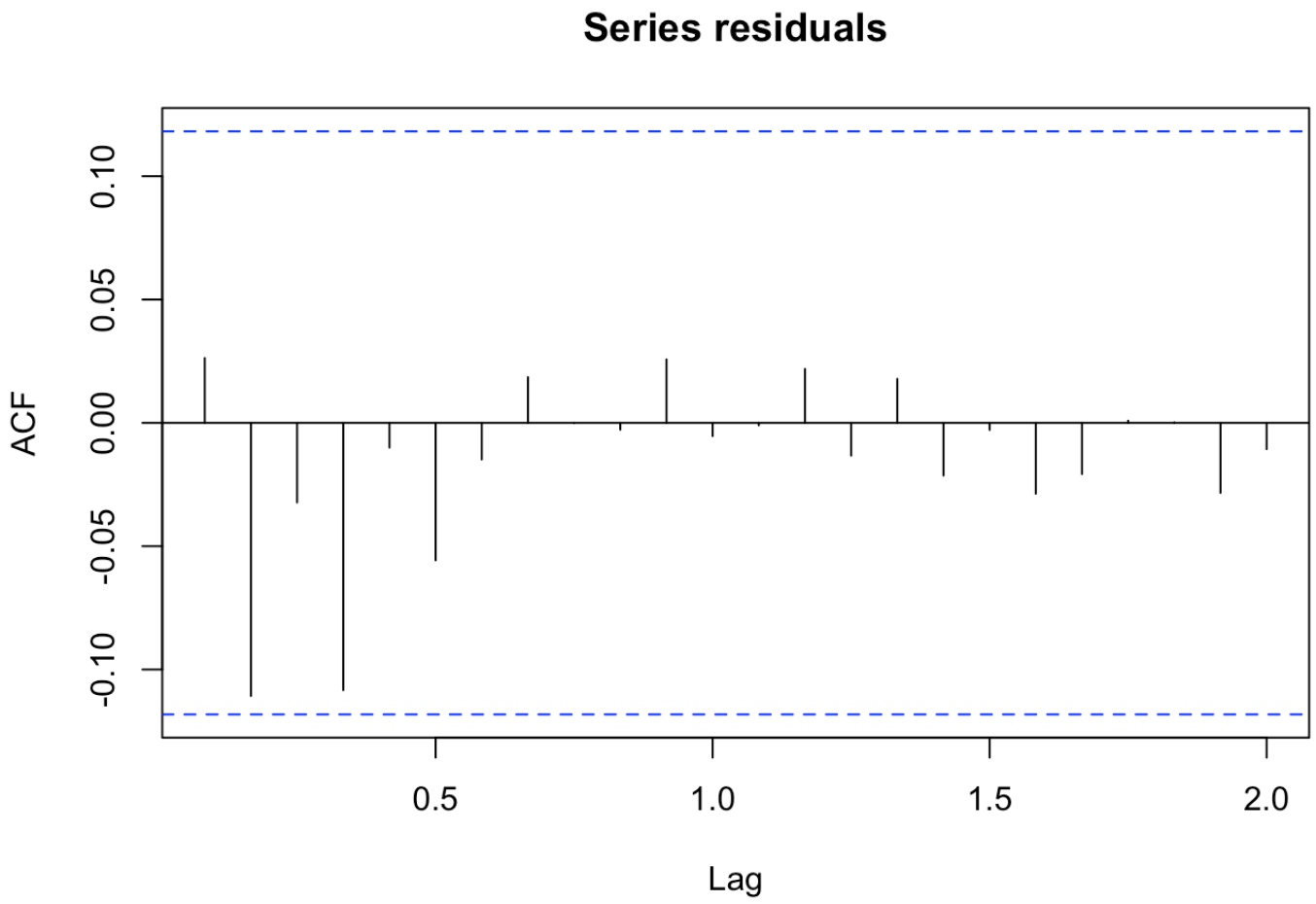
```
adf.test(residuals)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  residuals
## Dickey-Fuller = -7.4291, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

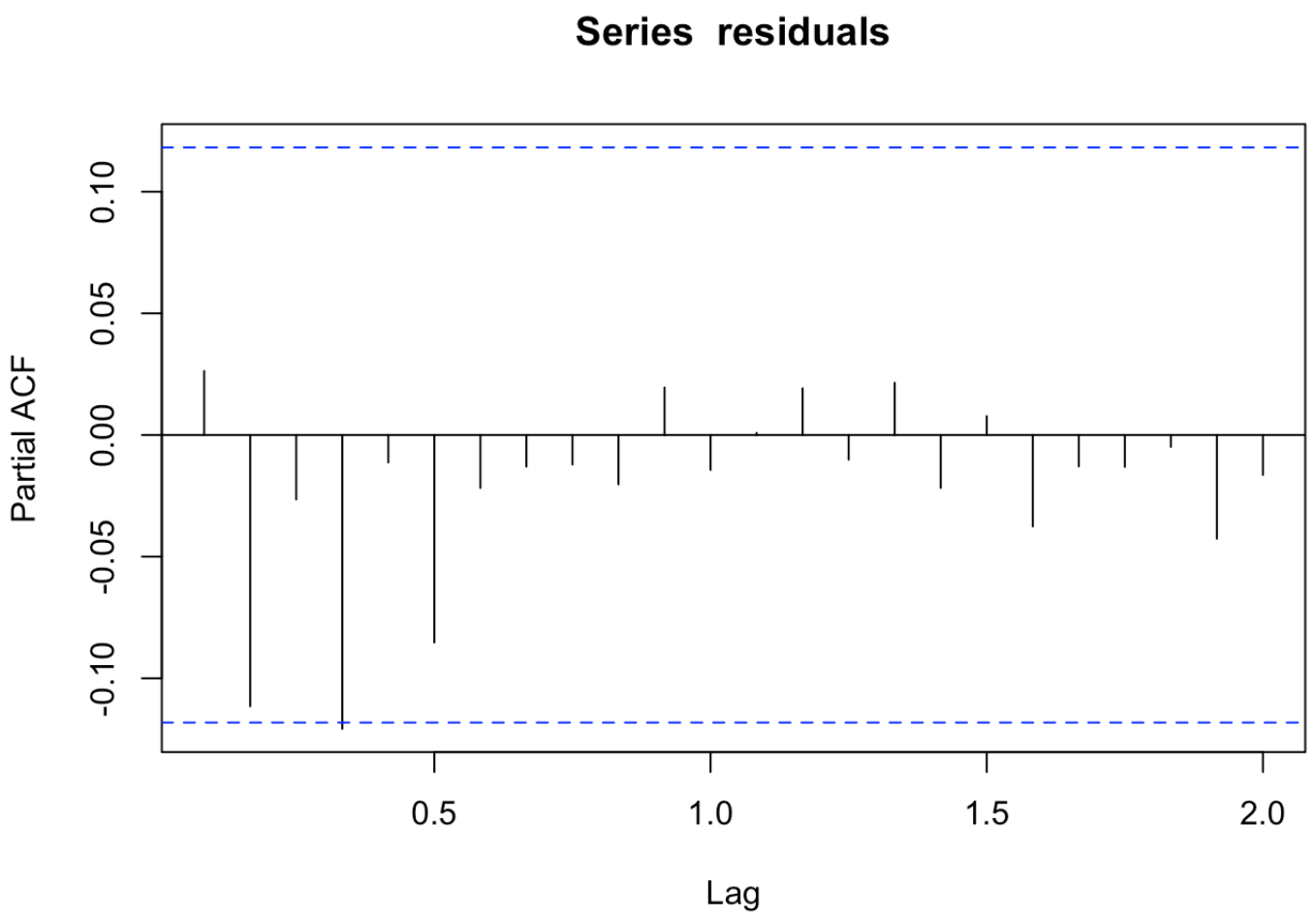
ACF of residuals

```
acf(residuals)
```

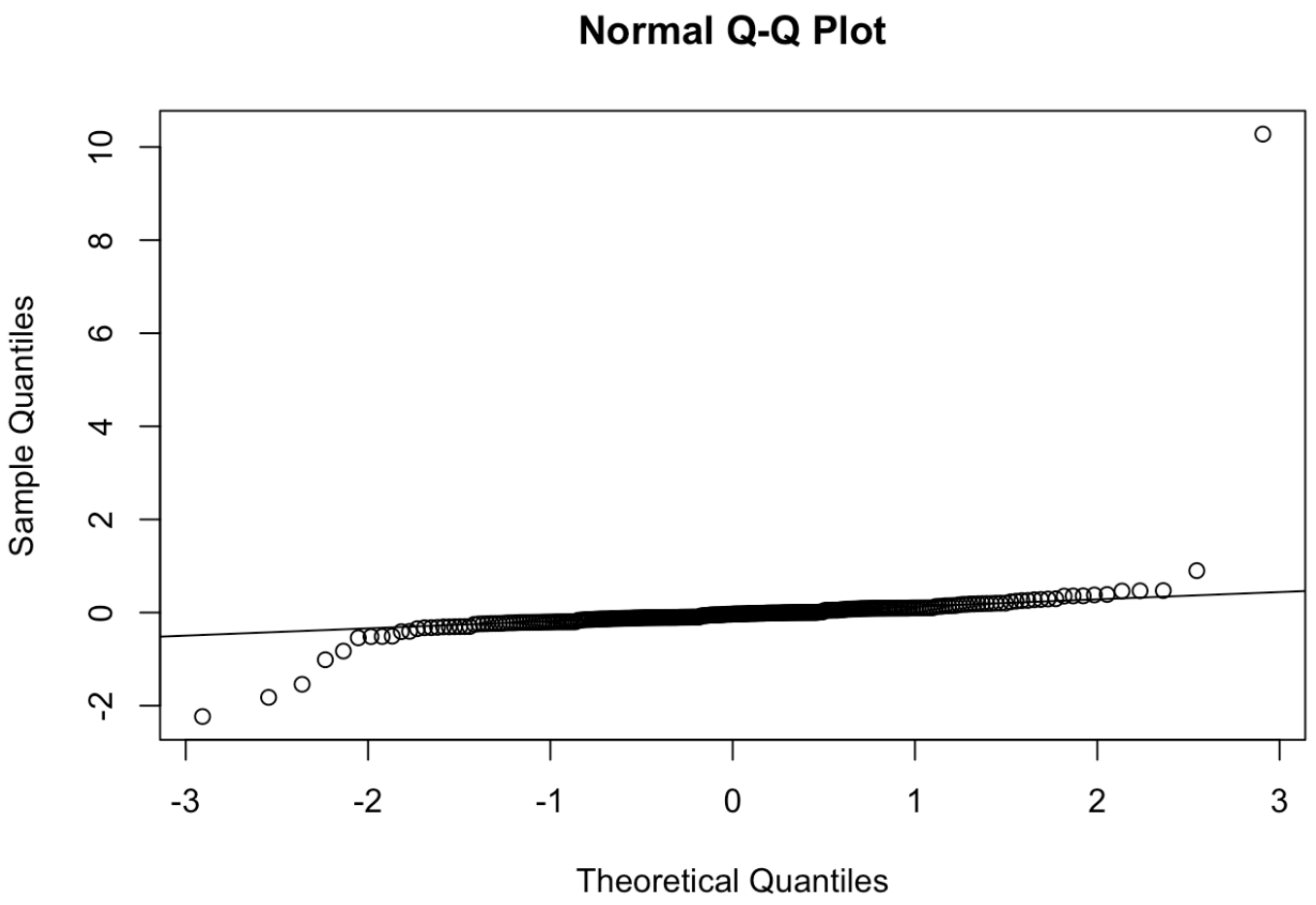


PACF of residuals

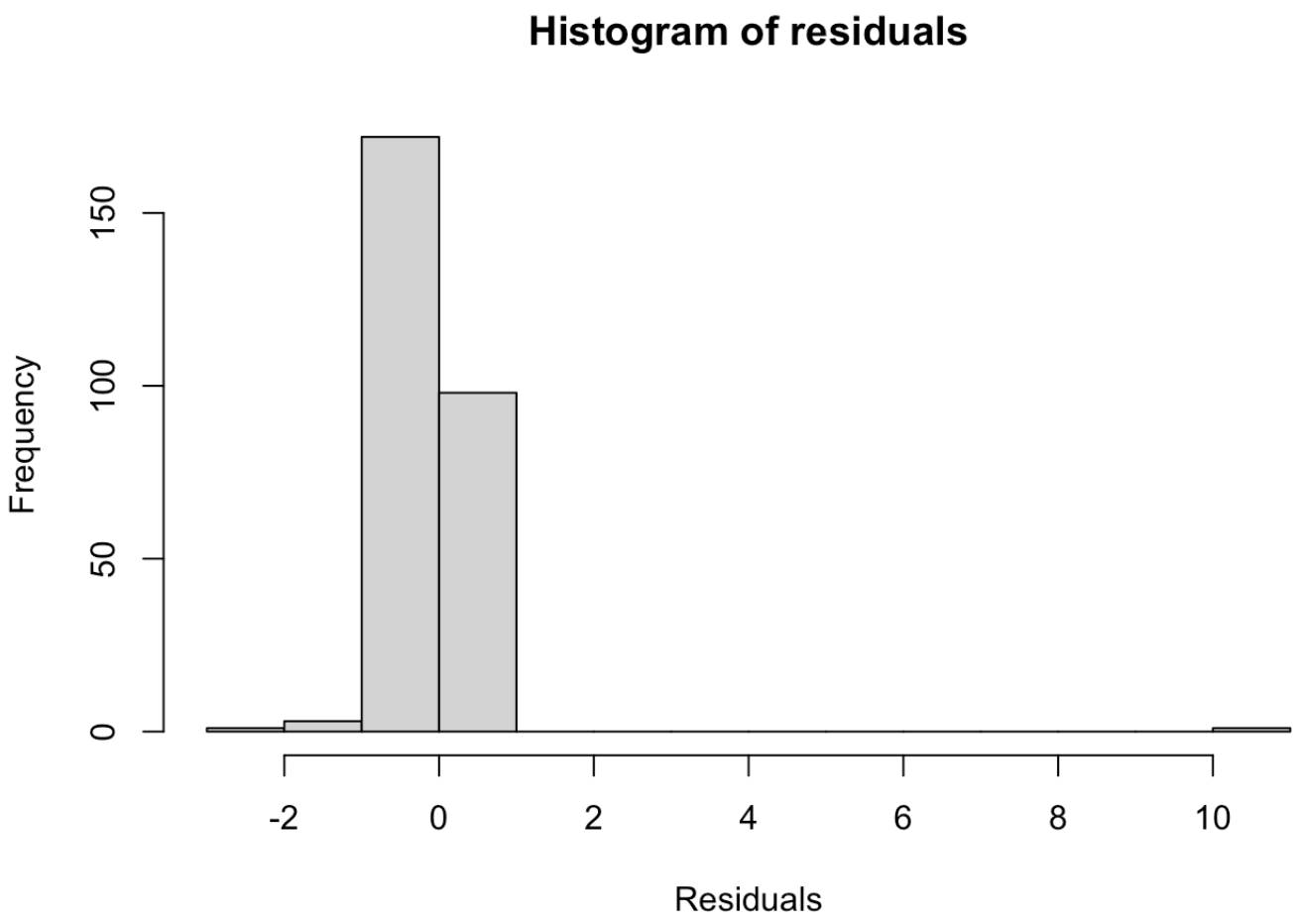
```
pacf(residuals)
```




```
qqnorm(residuals)
qqline(residuals)
```




```
hist(residuals, xlab='Residuals')
```



Ljung-Box test :

The Ljun-Box test is a hypothesis test that checks if a time series contains an autocorrelation. The null Hypothesis H0 is that the residuals are independently distributed. The alternative hypothesis is that the residuals are not independently distributed and exhibit a serial correlation.

The Ljung-Box test uses the following hypotheses: H0: The residuals are independently distributed. HA: The residuals are not independently distributed; they exhibit serial correlation

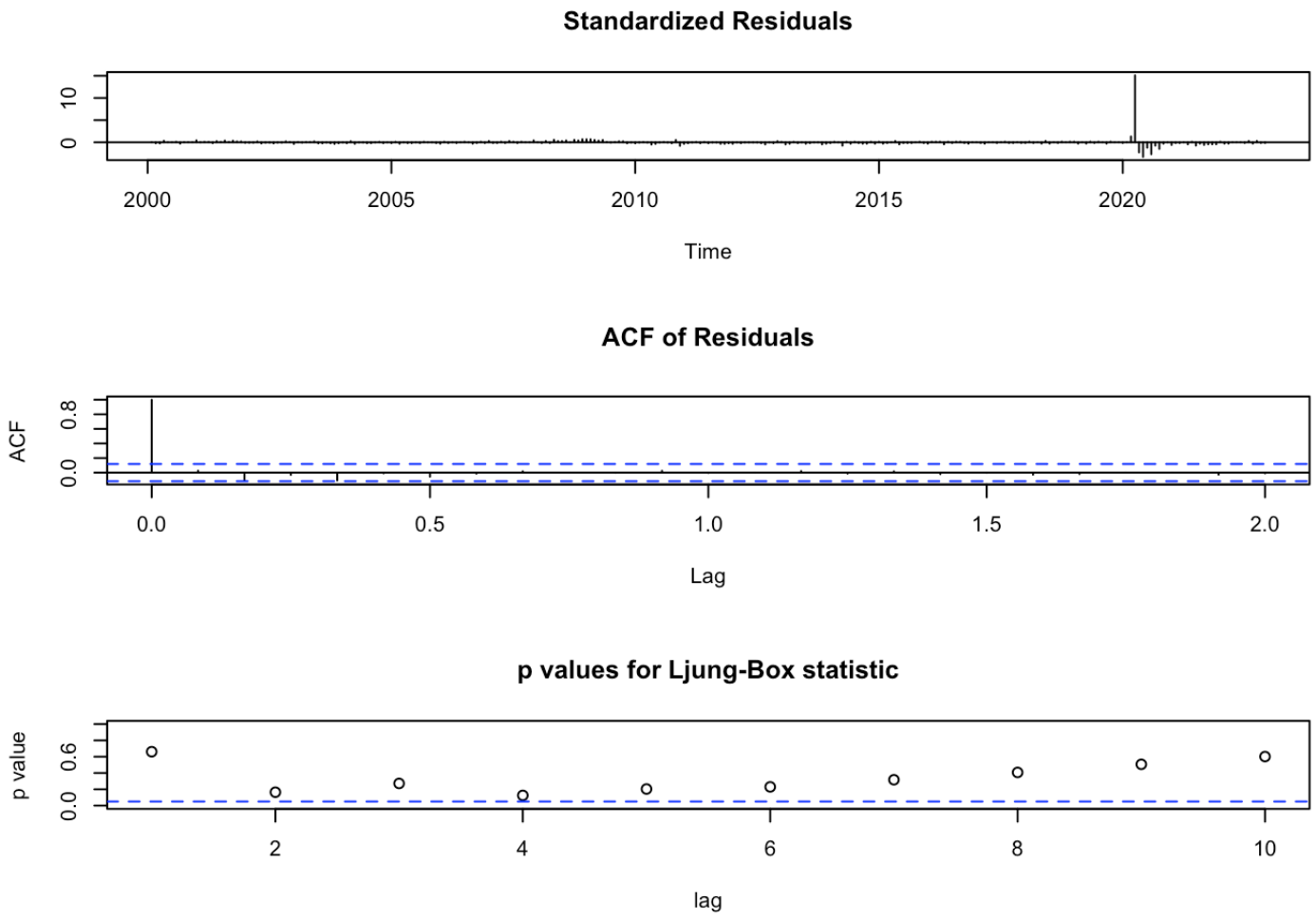
```
ljung_box_test = Box.test(residuals, type="Ljung-Box")
ljung_box_test
```

```
##
##  Box-Ljung test
##
## data:  residuals
## X-squared = 0.192, df = 1, p-value = 0.6613
```

P Value is > 0.05, failed to reject null Hypothesis.

Hence,The residuals are independently distributed.

```
tsdiag(model)
```



Forecaasting:

```
library(forecast)

# Use the model to forecast future values
forecast_values <- forecast(model, h = 6)

# Print the forecast values
print(forecast_values)
```

```
##          Point Forecast     Lo 80     Hi 80     Lo 95    Hi 95
## Jan 2023   -0.001818182 -0.875941 0.8723047 -1.338674 1.335037
## Feb 2023   -0.001818182 -0.875941 0.8723047 -1.338674 1.335037
## Mar 2023   -0.001818182 -0.875941 0.8723047 -1.338674 1.335037
## Apr 2023   -0.001818182 -0.875941 0.8723047 -1.338674 1.335037
## May 2023   -0.001818182 -0.875941 0.8723047 -1.338674 1.335037
## Jun 2023   -0.001818182 -0.875941 0.8723047 -1.338674 1.335037
```

```
# Create a data frame with one column containing the point forecasts
df <- data.frame(forecast_values$mean)

# Print the data frame
print(df)
```

```
##   forecast_values.mean
## 1         -0.001818182
## 2         -0.001818182
## 3         -0.001818182
## 4         -0.001818182
## 5         -0.001818182
## 6         -0.001818182
```

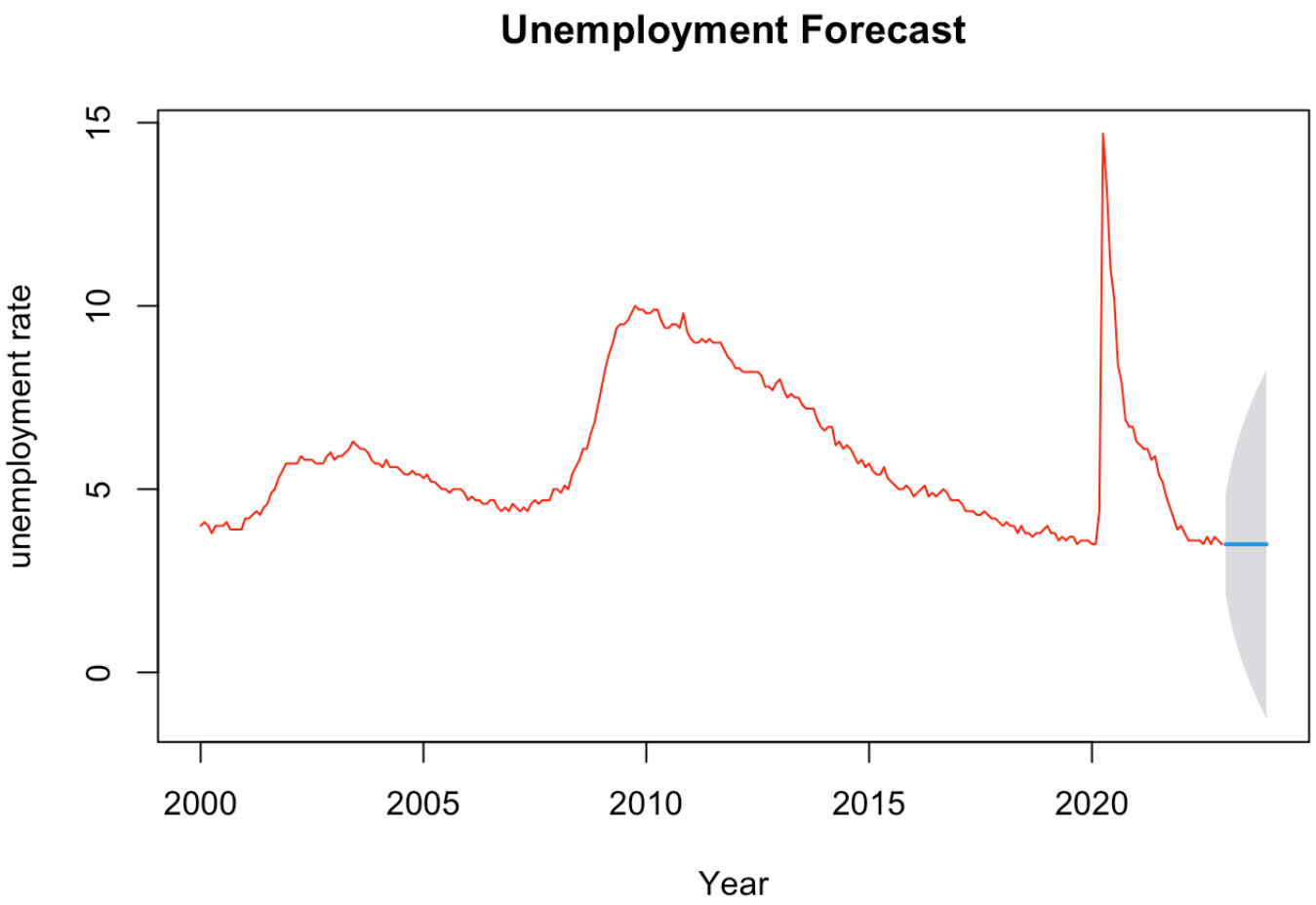
```
# Create time series object with start and end dates
data.ts <- ts(data.ts, start = c(2000, 1), end = c(2022, 12), frequency = 12)

# Fit the ARIMA model
Forecast_model <- Arima(data.ts, order = c(0, 1, 1))

# Forecast
rate_forecast <- forecast(Forecast_model, h = 12, level = c(95))
print(rate_forecast)
```

```
##          Point Forecast      Lo 95    Hi 95
## Jan 2023       3.496678  2.1653053 4.828051
## Feb 2023       3.496678  1.5810835 5.412273
## Mar 2023       3.496678  1.1373451 5.856011
## Apr 2023       3.496678  0.7647554 6.228601
## May 2023       3.496678  0.4372090 6.556147
## Jun 2023       3.496678  0.1414881 6.851868
## Jul 2023       3.496678 -0.1302007 7.123557
## Aug 2023       3.496678 -0.3829094 7.376265
## Sep 2023       3.496678 -0.6201348 7.613491
## Oct 2023       3.496678 -0.8444160 7.837772
## Nov 2023       3.496678 -1.0576657 8.051022
## Dec 2023       3.496678 -1.2613673 8.254723
```

```
# Plot the forecast
plot(rate_forecast, ylab = "unemployment rate", xlab = "Year", main = "Unemployment F
orecast", col = "red")
```



Actual Values: 2023 1 3.4
2023 2 3.6
2023 3 3.5

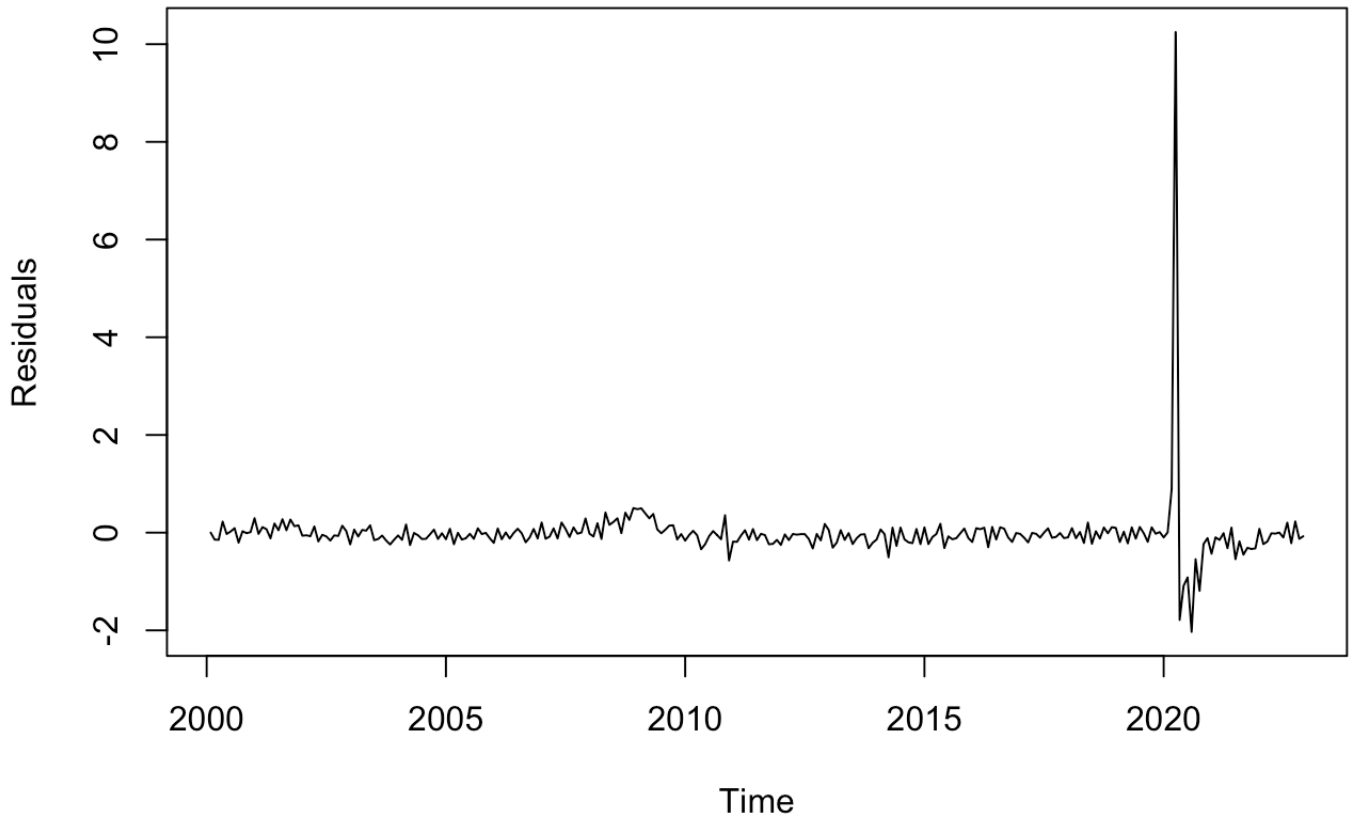
Model4 : Arima((2,1,1))

```
model4 <- Arima(new_data, order = c(2, 1, 1))
```

Residual Analysis

```
residuals4 <- residuals(model4)
```

```
plot(residuals4, type = "l", xlab = "Time", ylab = "Residuals")
```



Stationarity check on residuals

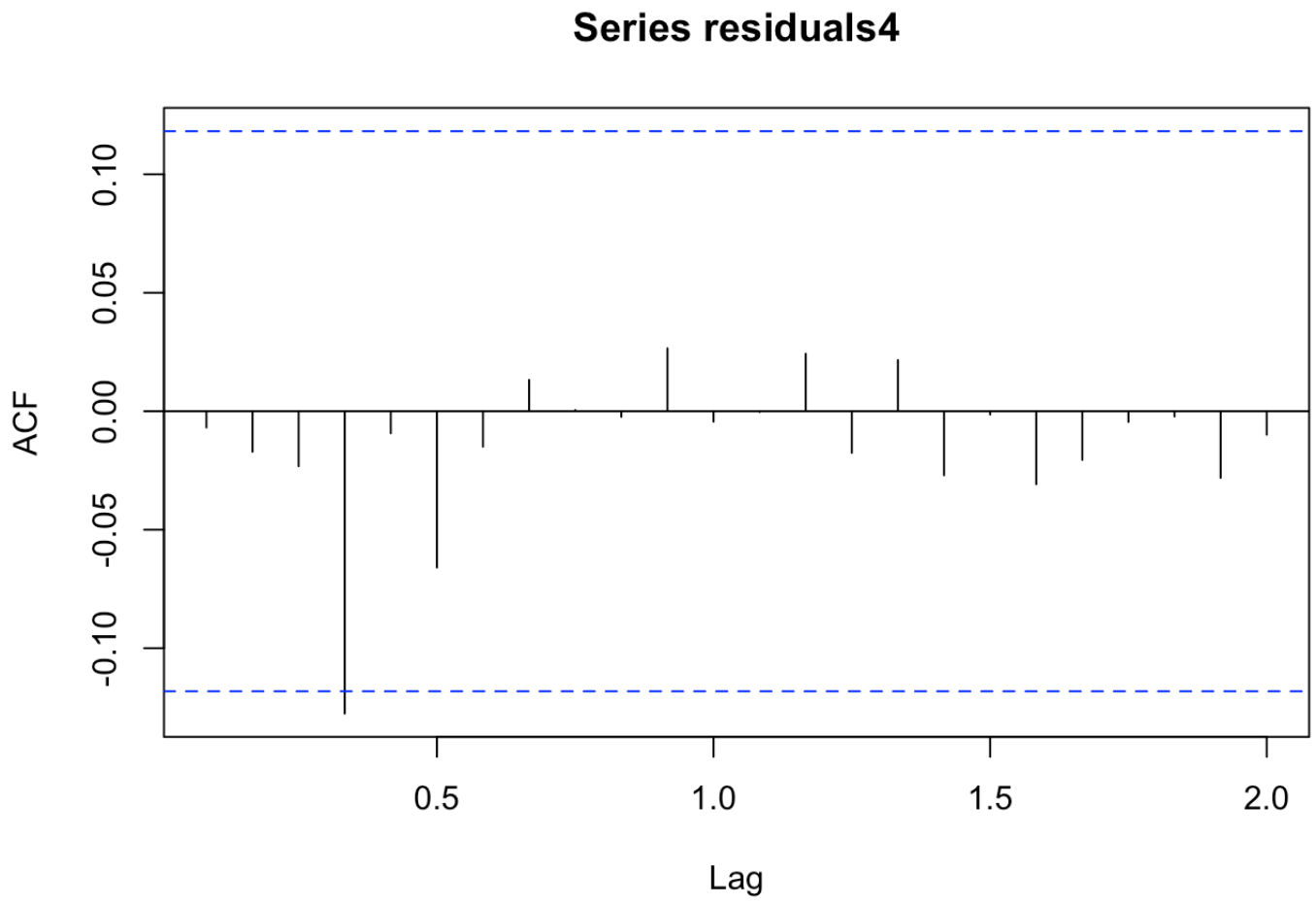
```
adf.test(residuals4)
```

```
## Warning in adf.test(residuals4): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  residuals4
## Dickey-Fuller = -7.2712, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

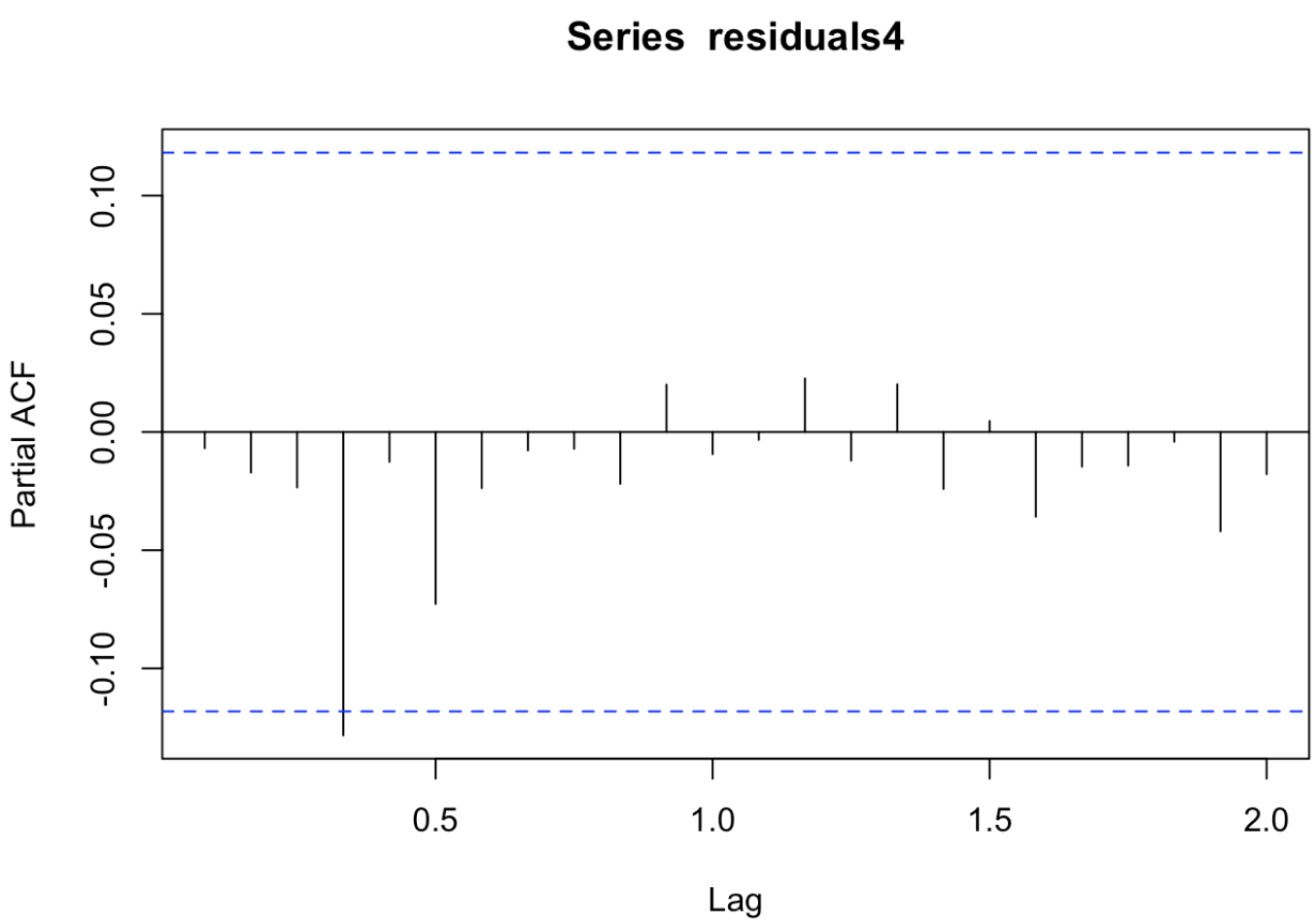
ACF of Residuals

```
acf(residuals4)
```



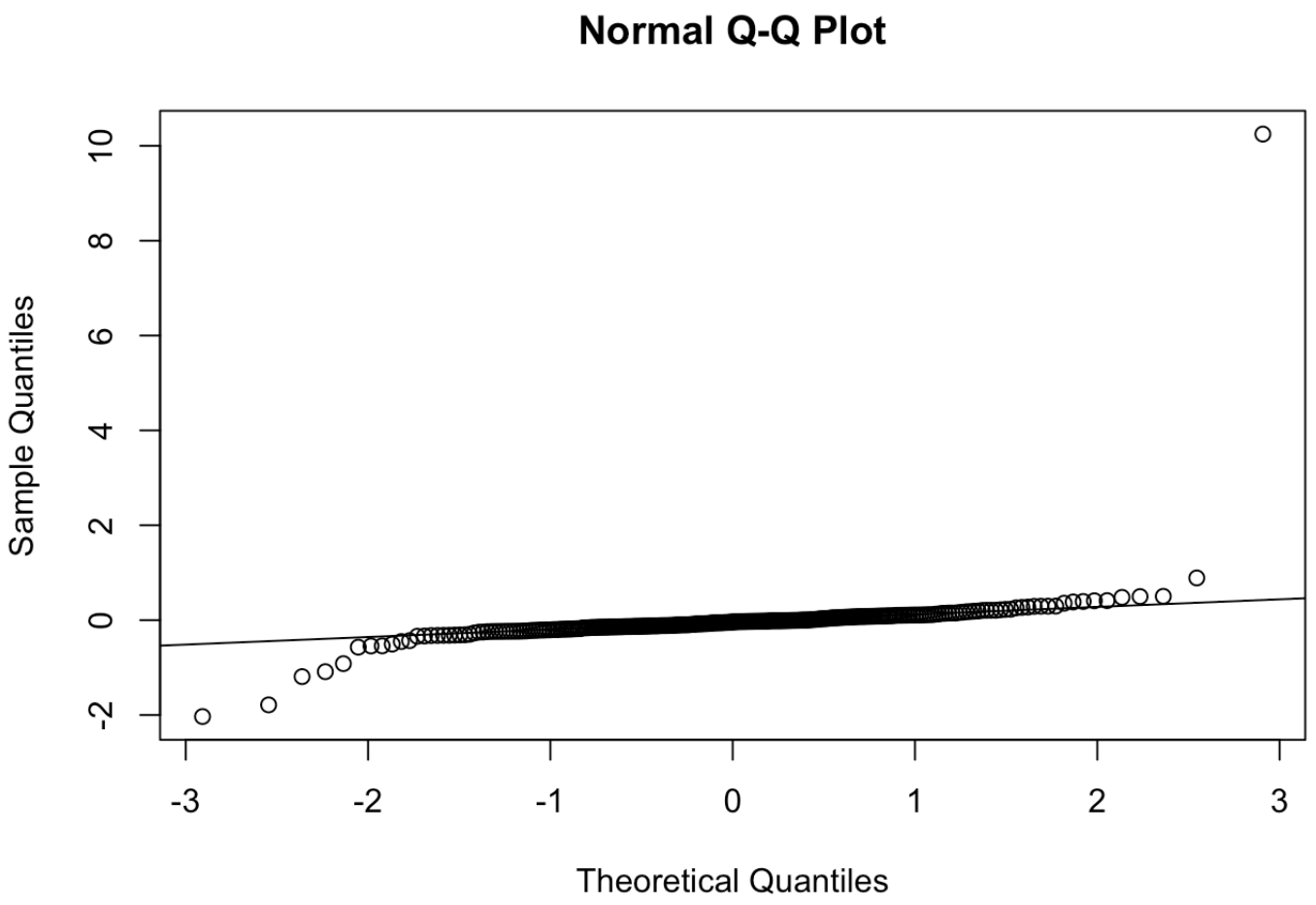
PACF of Residuals

```
pacf(residuals4)
```



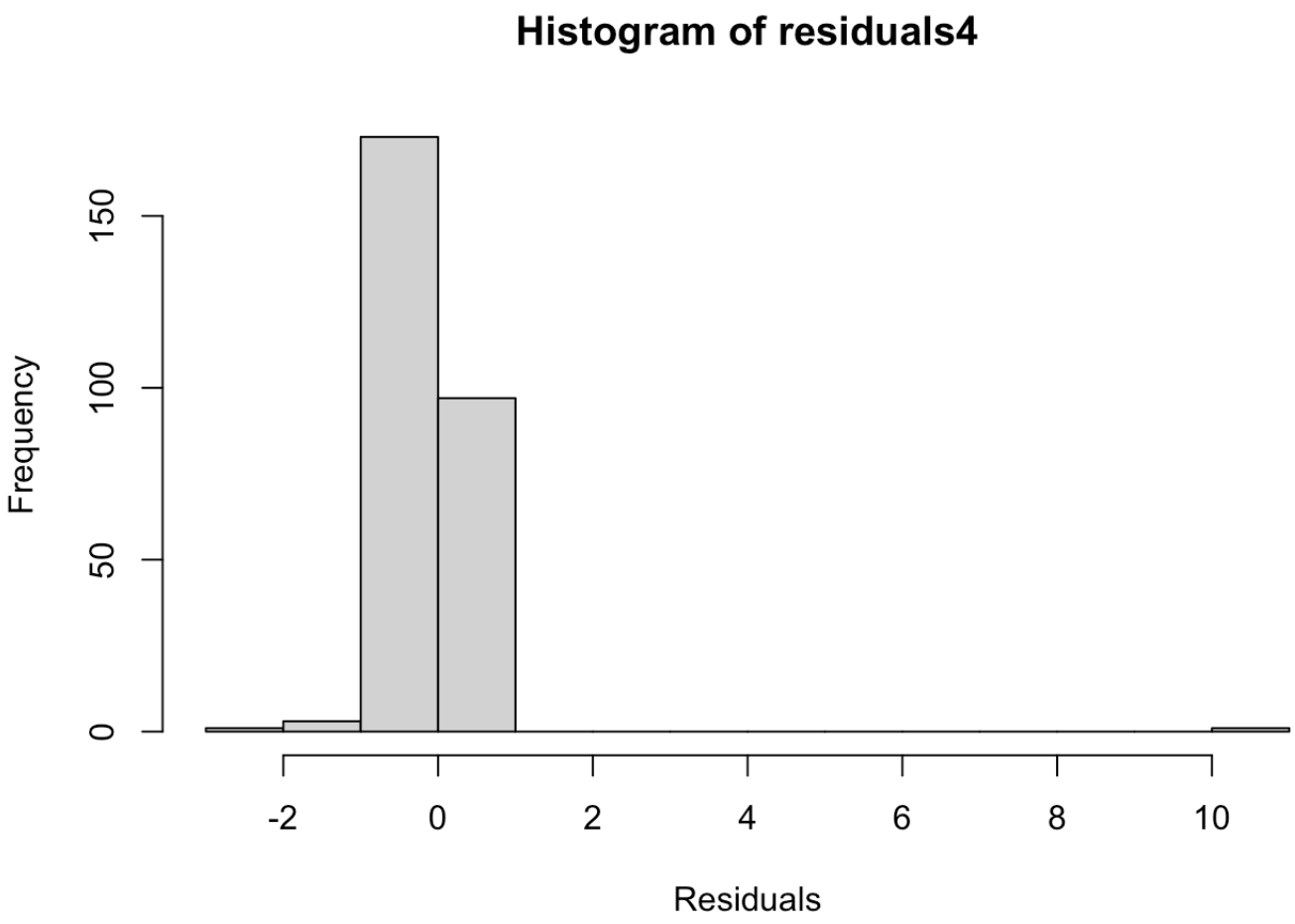
QQ Plot of Residual

```
qqnorm(residuals4)
qqline(residuals4)
```



Histogram of Residual

```
hist(residuals4, xlab='Residuals')
```

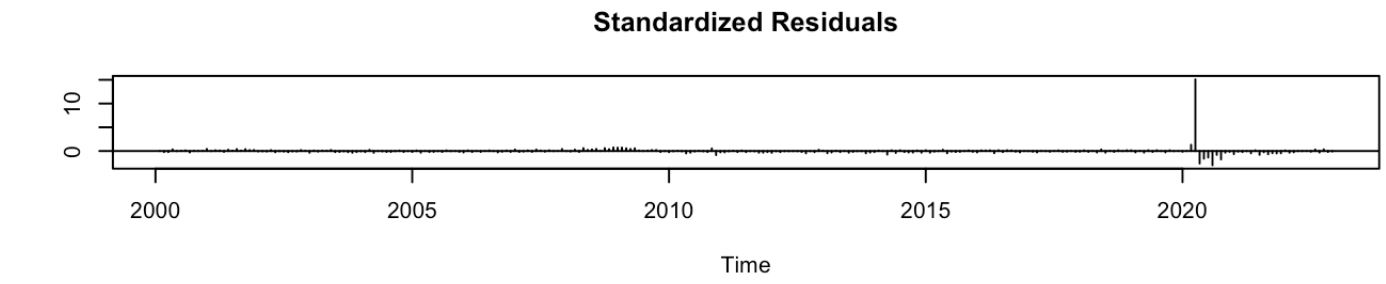


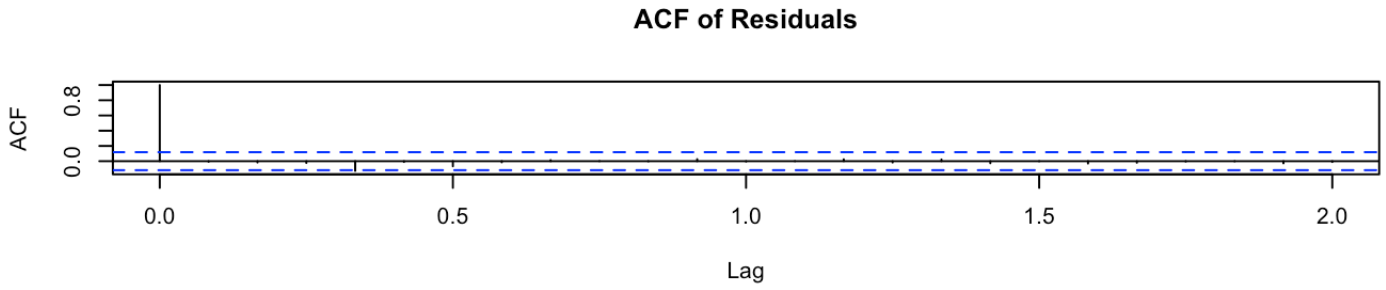
L jung Box test of Residuals

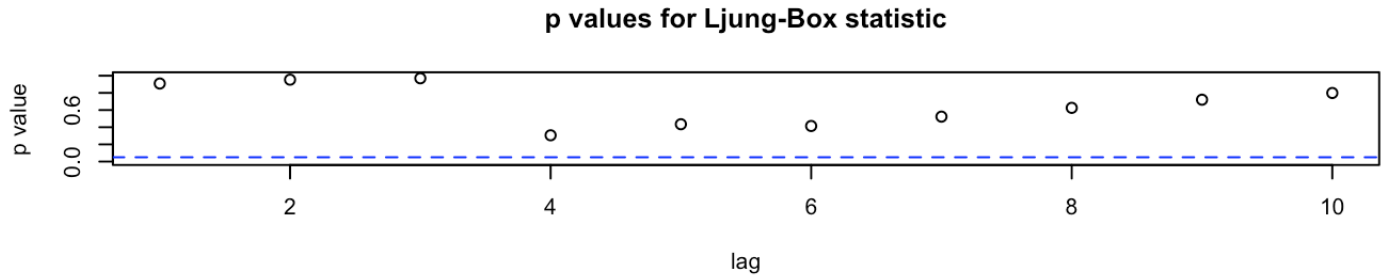
```
ljung_box_test = Box.test(residuals4, lag=20, type="Ljung-Box")
ljung_box_test
```

```
##
##  Box-Ljung test
##
## data:  residuals4
## X-squared = 7.4279, df = 20, p-value = 0.995
```

```
tsdiag(model4)
```







Forecasting:

```
library(forecast)

# Use the model to forecast future values
forecast_values4 <- forecast(model4, h = 12)

# Print the forecast values
print(forecast_values4)
```

```
##          Point Forecast      Lo 80     Hi 80     Lo 95    Hi 95
## Jan 2023    0.005458835 -0.8661949 0.8771126 -1.327621 1.338538
## Feb 2023    0.008964180 -0.8632760 0.8812044 -1.325012 1.342940
## Mar 2023   -0.002160215 -0.8788907 0.8745703 -1.343004 1.338683
## Apr 2023   -0.002903615 -0.8796351 0.8738279 -1.343749 1.337941
## May 2023   -0.001742570 -0.8785584 0.8750733 -1.342717 1.339232
## Jun 2023   -0.001624739 -0.8784444 0.8751949 -1.342605 1.339355
## Jul 2023   -0.001744579 -0.8785615 0.8750723 -1.342720 1.339231
## Aug 2023   -0.001761122 -0.8785776 0.8750553 -1.342736 1.339214
## Sep 2023   -0.001748898 -0.8785657 0.8750679 -1.342724 1.339227
## Oct 2023   -0.001746728 -0.8785636 0.8750701 -1.342722 1.339229
## Nov 2023   -0.001747959 -0.8785648 0.8750689 -1.342724 1.339228
## Dec 2023   -0.001748231 -0.8785650 0.8750686 -1.342724 1.339227
```

```
# Create a data frame with one column containing the point forecasts
df <- data.frame(forecast_values4$mean)

# Print the data frame
#print(df)
```

```
df$Point.Forecast
```

```
## NULL
```

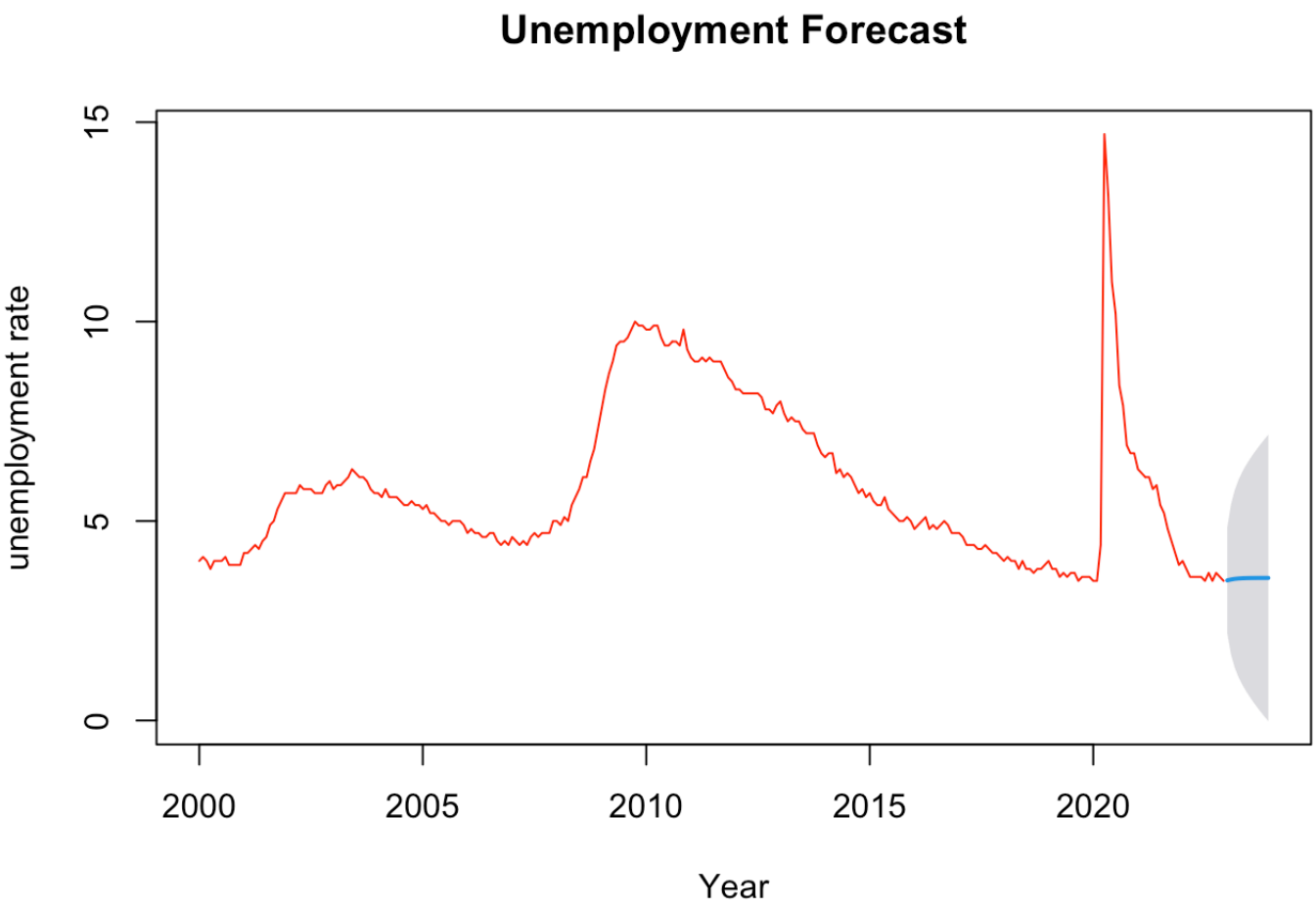
```
# Create time series object with start and end dates
data.ts <- ts(data.ts, start = c(2000, 1), end = c(2022, 12), frequency = 12)

# Fit the ARIMA model
Forecast_model4 <- Arima(data.ts, order = c(2, 1, 1))

# Forecast
rate_forecast4 <- forecast(Forecast_model4, h = 12, level = c(95))
print(rate_forecast4)
```

```
##          Point Forecast       Lo 95    Hi 95
## Jan 2023       3.512653  2.19262835 4.832677
## Feb 2023       3.534554  1.65764135 5.411466
## Mar 2023       3.550283  1.32904585 5.771520
## Apr 2023       3.560050  1.09411595 6.025983
## May 2023       3.565859  0.90623679 6.225482
## Jun 2023       3.569266  0.74451807 6.394015
## Jul 2023       3.571255  0.59881006 6.543699
## Aug 2023       3.572413  0.46380521 6.681020
## Sep 2023       3.573087  0.33654032 6.809634
## Oct 2023       3.573480  0.21525713 6.931702
## Nov 2023       3.573708  0.09884812 7.048568
## Dec 2023       3.573841 -0.01342717 7.161109
```

```
# Plot the forecast
plot(rate_forecast4, ylab = "unemployment rate", xlab = "Year", main = "Unemployment
Forecast", col = "red")
```



Forecasat of. the unemployment rate of the year 2023

```
rate_forecast4
```

```
##          Point Forecast       Lo 95    Hi 95
## Jan 2023       3.512653  2.19262835 4.832677
## Feb 2023       3.534554  1.65764135 5.411466
## Mar 2023       3.550283  1.32904585 5.771520
## Apr 2023       3.560050  1.09411595 6.025983
## May 2023       3.565859  0.90623679 6.225482
## Jun 2023       3.569266  0.74451807 6.394015
## Jul 2023       3.571255  0.59881006 6.543699
## Aug 2023       3.572413  0.46380521 6.681020
## Sep 2023       3.573087  0.33654032 6.809634
## Oct 2023       3.573480  0.21525713 6.931702
## Nov 2023       3.573708  0.09884812 7.048568
## Dec 2023       3.573841 -0.01342717 7.161109
```

Actual values: 2023 1 3.4
2023 2 3.6
2023 3 3.5

Conclusion:

In this project, we analyzed the US unemployment rate using time series analysis. Based on our analysis, we found that the ARIMA(2, 1, 1) model was a good fit for the data, and we used this model to make forecasts for future unemployment rates. However, it is important to note that the unemployment rate is affected by various factors, such as recessions, natural disasters, and human attacks, which can impact the accuracy of our forecasts. Therefore, when making policy decisions or planning for the future, it is important to take these factors into consideration and adjust our forecasts accordingly.