Earth's Weakening Magnetic Field

Jun Xian Lu 1006300888

2023-04-10

Contents

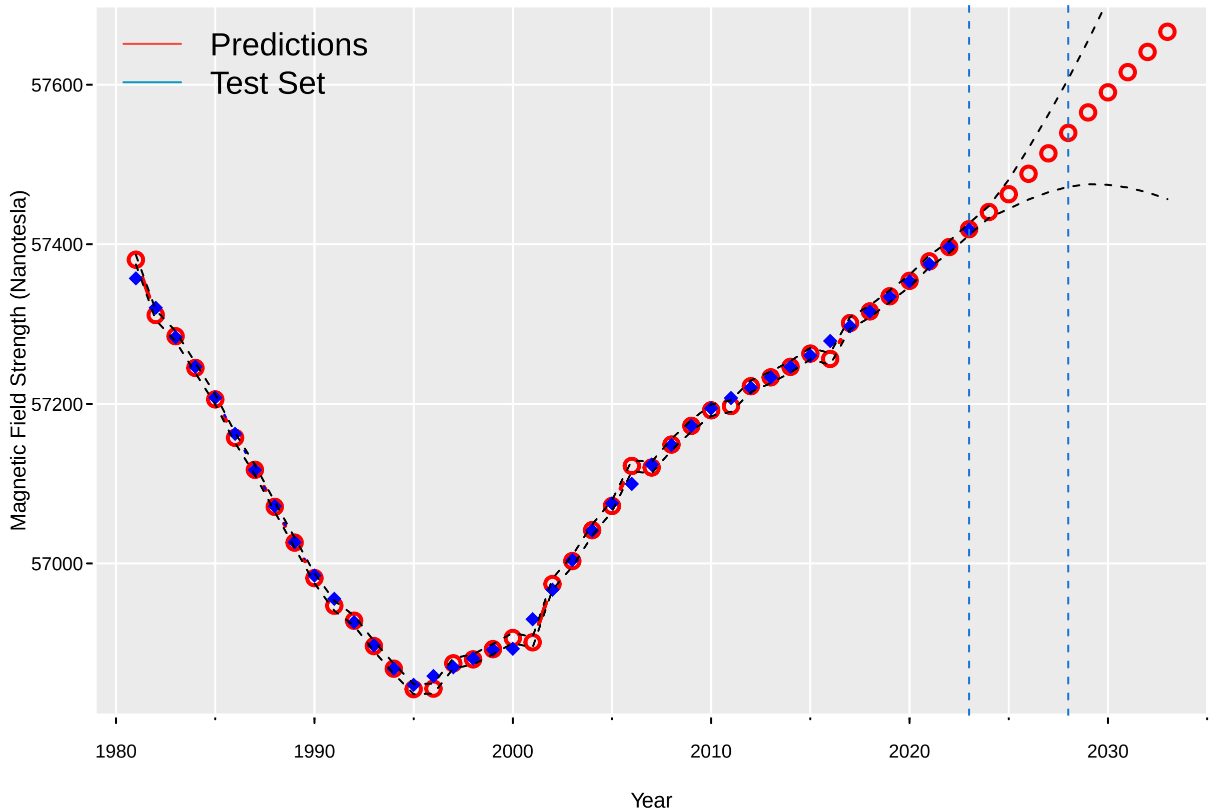
1 Abstract

The goal of this project is to provide an insight on Earth's change in geomagnetic field strength for the next decade. There is sufficient evidence to conclude that the north magnetic field strength will increase until the year 2028. On the other hand, there is a lack of evidence to make any conclusive statement about the south magnetic field strength.

1.1 North Geomagnetic Pole

The model is a multiplicative seasonal ARIMA model given as $ARIMA(1,1,6) \times (3,1,0)_5$ with parameters given below and the forecasted time series.

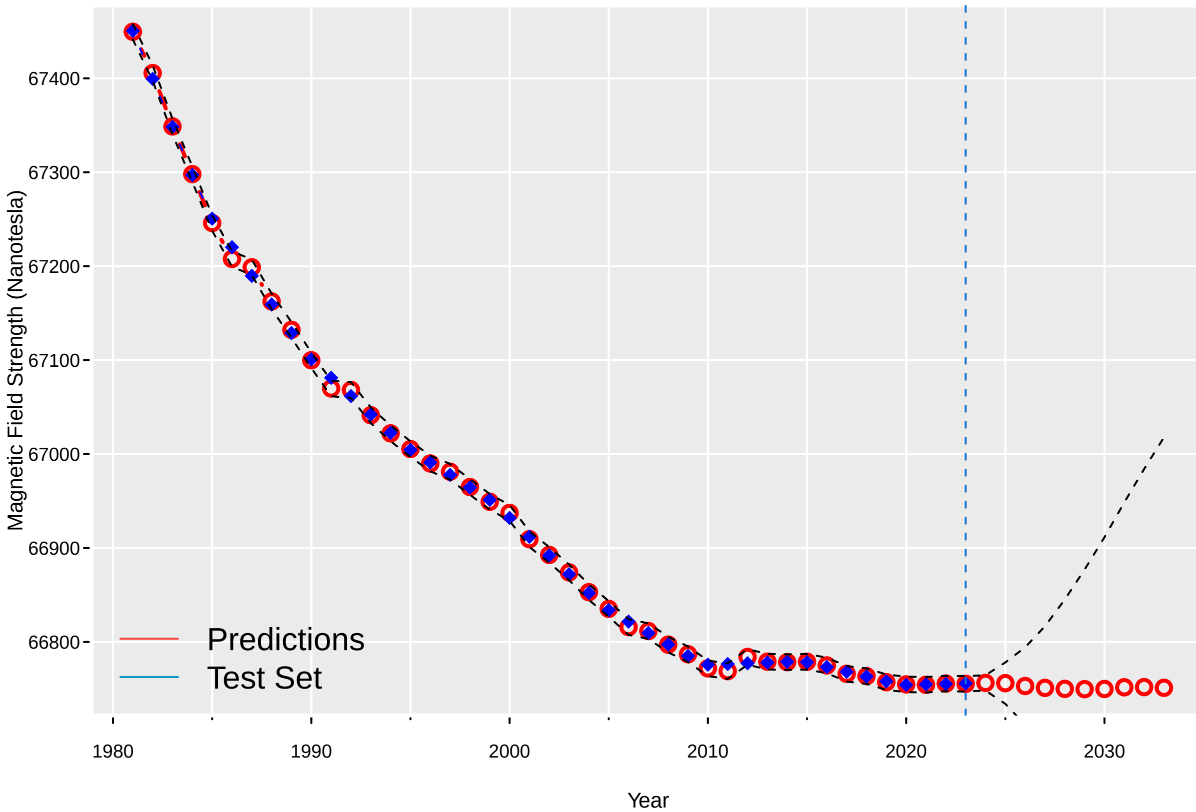
```
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4      ma5      ma6    sar1    sar2
##       0.9443  0.4430  0.1600  0.1544  0.1530  -0.8448  -0.2848  0.1269  0.0332
## s.e.  0.0447  0.0692  0.0851  0.0863  0.0863   0.0854   0.0568  0.0589  0.0569
##          sar3    xmean
##       -0.1434  -1.8933
## s.e.   0.0581   2.1436
##
## sigma^2 estimated as 9.022:  log likelihood = -979.91,  aic = 1983.82
```



1.2 South Geomagnetic Pole

The model is $ARIMA(7,1,8)$ with parameters given below and the forecasted time series.

```
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     ar2      ar3     ar4      ar5     ar6     ar7     ma1     ma2
##       -0.0932  0.5431  -0.1236  0.0919  -0.5064  0.1098  0.6199  1.6026  1.3643
## s.e.   0.1034  0.0969   0.0716  0.0734   0.0582  0.1008  0.0964  0.1054  0.1188
##          ma3     ma4     ma5     ma6     ma7     ma8     xmean
##       1.4698  1.5214  1.6573  1.5147  0.8925  0.3441  -10.1381
## s.e.  0.1155  0.0991  0.1019  0.1321  0.1068  0.0607    6.4896
##
## sigma^2 estimated as 17.35:  log likelihood = -1116.1,  aic = 2266.2
```



2 Introduction

Earth's magnetic field is generated by its molten iron core. The core allow free movement of electrons which generates electric currents that in turn generates the magnetic field. The magnetic field constantly changes, however, Earth's magnetic field strength overall has been decreasing for the last few centuries, this is also evident through the data set used for this project. In particular, the data set has yearly magnetic field strength starting the year 1590 to 2023 of two different locations, one is near the north pole and the other is near the south pole. Earth's magnetic field shield the Earth from crucial solar wind and solar storm which could disrupt our communication systems and electronics. Moreover, it could severely impact life on the planet since the charged particles and radiation that come from the Sun could impact weather patterns and cause health problems. Recent studies has indicated that Earth's magnetic field is undergoing significant changes which prompts the following questions: will Earth's magnetic field continue to weaken in the upcoming decade? If so, by how much?

Note that it is recommend to download and view the PDF file of this report since the figures are labeled and cross-referenced for better navigation.

3 North Geomagnetic Pole

The time series of the north geomagnetic pole was plotted using the training set (Figure 1). It appears that it is non-stationary.

3.1 Model Specification

To verify the non-stationary behavior of north geomagnetic pole data, the mean level plot (Figure 2) and both ACF and PACF plots (Figure 3) were plotted. ADF tests and KPSS tests were performed (see Section 7.1.1) and indeed, all indicates a non-stationary behavior.

In an attempt to transform the data to a stationary one, first order differencing along with logarithmic transformation, square root transformation, and reciprocal transformation were applied, however, all gave the same result that the data is still non-stationary suggesting further differencing may be required. Figure 4 gives the time series of the first order difference, Figure 5 gives both the ACF and PACF plots, and Section 7.1.2 gives the ADF tests and KPSS tests for the first order differencing.

As a result, second order differencing was employed and this time, the times series (Figure 6) and both ACF and PACF plots (Figure 7) indicates a possible stationary behavior. To verify stationarity, ADF and KPSS tests (Section 7.1.3) were used and it gave sufficient evidence that it may be stationary, namely, p-values less than 0.05 for all 3 different ADF tests and p-values slightly greater than 0.01 for both KPSS tests. Furthermore, it seems there is a seasonal pattern from the ACF, in addition, the p-values from the KPSS tests are quite close to 0.01 so perhaps taking an additional seasonal difference might improve the model.

This led to adding an additional seasonal difference at lag 10 (tried lag 5, 10, 15, 20, but lag 10 seems to be the best from ACF/PACF and KPSS tests) to the already second order differenced data. The time series (Figure 8) does not appear to be vastly different from previous time series without the seasonal difference, however, both the ACF and the PACF plots (Figure 9) gave stronger evidence of stationarity. Despite all this, this is not the transformation selected since the ADF and KPSS tests (Section 7.1.4) suggests that it may be overdifferenced. The 3 different ADF tests all gives a p-value of 0.01 and both the KPSS tests give a p-value of 0.1.

After experimenting with many different candidate transformations while taking in the consideration of overdifferencing and the appeared seasonality from previous transformations, a transformation that takes the seasonal differencing at lag 5 and a first order non-seasonal differencing came out on top. The times series of this transformation (Figure 10) appears stationary with no evidence of overdifferencing. The mean level plot (Figure 11) shows that the mean may not be constant but the deviation from constant mean does not appear to be large. The ACF and PACF plots (Figure 12) appears to exhibit ARIMA properties since tailing off, in addition, it has a clear signature of seasonal effect. Despite failing to reject the null hypothesis of ADF test of data having constant and trend (p-value 0.216), this does not necessarily mean it is non-stationary since the KPSS test for trend gives a p-value of 0.01817, hence the decision to reject the null hypothesis is made at $\alpha = 1\%$ (Section 7.1.5). Therefore, together the KPSS test for trend and ADF test for constant (p-value 0.066 hence reject at $\alpha = 10\%$), it is concluded that the underlying transformation is stationary.

3.2 Fitting and Diagnostics

EACF was (Section 7.1.6) plotted with data using transformation that takes the seasonal differencing at lag 5 and a first order non-seasonal differencing. AIC and BIC values (Section 7.1.7) from candidate models were calculated and sorted. Standard residuals plot, ACF of residuals plot, QQ plot, and Ljung-Box test were performed to each of candidate models, once again, using the transformed data. Ultimately, the three best performing models are ARIMA(1,0,6), ARIMA(2,0,6), and ARIMA(5,0,5), however, ARIMA(1,0,6) was chosen since it has fewer parameters (Principle of Parsimony). The standardized residual plots, ACF of residuals plot and Ljung-Box test of ARIMA(1,0,6) (Figure 13) confirms existence of uncorrelated errors, but the QQ plot clearly reject the normality of error. This is fine since it is difficult to have the normality of error assumption, this however will affect the prediction interval which will be further discussed in the discussion (5).

To determine the seasonal parameters, the residual plot and ACF of residuals plot were further examine, this led to first increasing the MA terms, then the AR terms, and finally, repeating the steps and reasoning in the above paragraph, the best model is given as $ARIMA(1, 1, 6) \times (3, 1, 0)_5$ (Figure 16). Note that the parameters of this model is the true model based off the original (non-transformed) data. This model has the lowest standard error of estimates and relatively low AIC and BIC values compare to other candidate models (Section 7.1.9), not to mention it has fewer parameters compare to more complex models. Unfortunately, this model still does not pass the normality of error (Shapiro test 7.1.8).

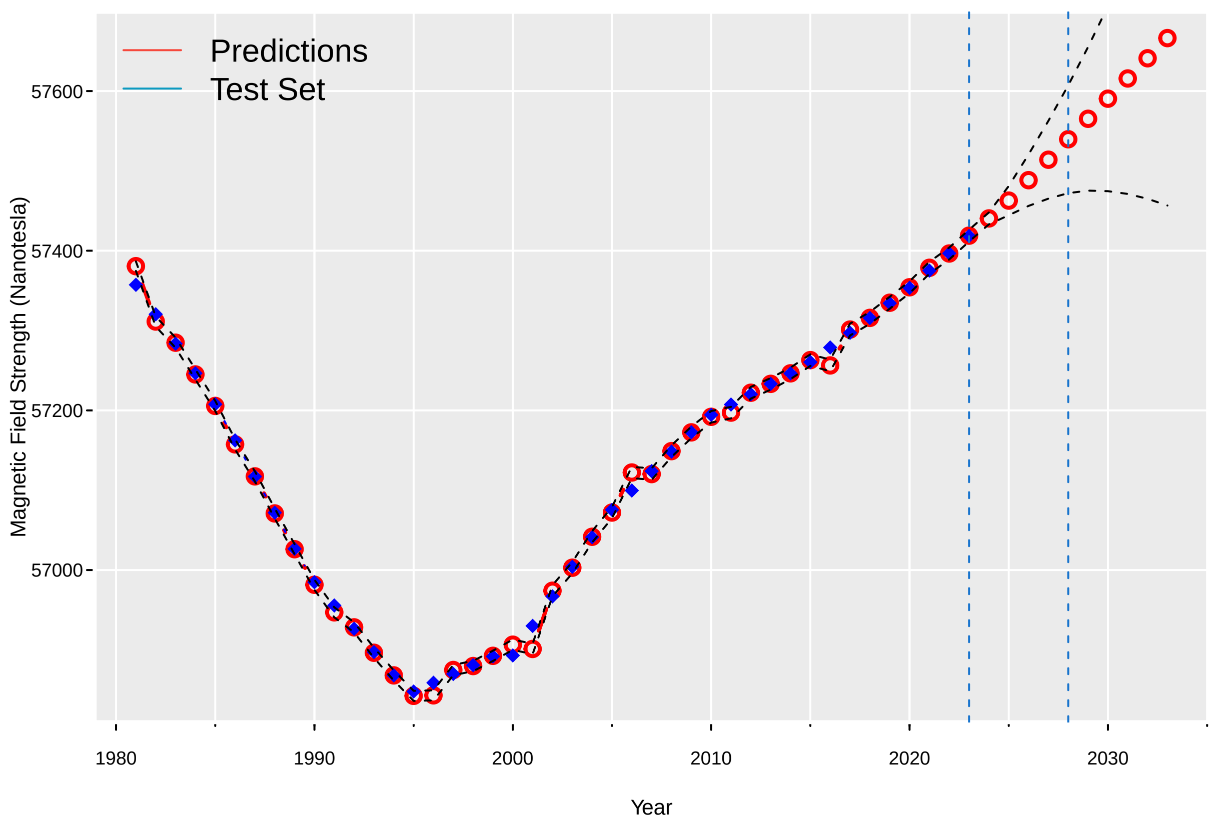
3.3 Forecasting

The best model given as $ARIMA(1, 1, 6) \times (3, 1, 0)_5$ (Section 1.1) was used to predict and compare with the test data. The prediction was done using the estimated parameters of the model and all of the training data to forecast the magnetic field strength for next year. This method is repeated until it reaches the end of the test data, the root mean square error (RMSE below) is then computed to test how well the model is performing. The RMSE is 8.77 which indicates the model is doing very well comparing to the scale of magnetic field strength which ranges from 56000 to 58000.

```
RMSEN = RMSE(data$North_Geomagnetic_Pole[(nTrain+1):n],x.test.forecast.N10[1:nTest])
RMSEN
```

```
## [1] 8.772111
```

To obtain useful information, further forecasting, namely the next 10 years (from 2024 to 2023) of north geomagnetic pole strength was forecasted the same way using this model. This time instead of using the training data, the entire data was used.



The first blue vertical dotted line indicates the year 2023 where our data set ends and the next 10 years of north geomagnetic field strength was plotted with its prediction interval. The prediction interval makes sense since it gets wider as the lead time increases. From the plot, it seems that the north geomagnetic field strength will increase for at least the next 5 years until 2028 (second blue dotted line). After 2028 the prediction interval gets so large that it becomes difficult to make any conclusion at a reasonable confidence.

4 South Geomagnetic Pole

The time series of the south geomagnetic pole was plotted using the training set (Figure 1). It appears that it is also non-stationary with greater evidence of seasonal effect.

4.1 Model Specification

Likewise with the north geomagnetic pole data, the mean level plot (Figure 17) and both ACF and PACF plots (Figure 18) were plotted and analyzed. ADF tests and KPSS tests (Section 7.3.1) were performed and all indicates a non-stationary behaviour.

Unlike the north geomagnetic pole data, the south geomagnetic pole data can be transformed to a stationary one once the first order non-seasonal difference is applied. This stionary characteristic is evident through the time series plot of first order difference (Figure 19), ACF and PACF plots (Figure 20), and both ADF and KPSS tests (Section 7.3.2). In particular, the ACF is tailing off and the p-value for KPSS test level is 0.1, p-value for KPSS test trend is 0.023 which can be both rejected at $\alpha = 1\%$. Together with the ADF tests, there is sufficient evidence to conclude the transformed data is stationary and the desired model consisting of a constant/drift term.

4.2 Fitting and Diagnostics

EACF was (Section 7.3.3) plotted with data using the non-seasonal first order difference. AIC and BIC values (Section 7.3.4) from candidate models were calculated and sorted. Standard residuals plot, ACF of residuals plot, QQ plot, and Ljung-Box test were performed to each of candidate models using the transformed data. Ultimately, the three best performing models are ARIMA(7,0,10), ARIMA(7,0,8), and ARIMA(6,0,8). Despite ARIMA(6,0,8) having 1 less parameter than ARIMA(7,0,8), ARIMA(7,0,8) was chosen since ARIMA(6,0,8) has a spike in the ACF of residual plots and smaller p-values for Ljung-Box test (Figure 21), suggesting that the errors may be uncorrelated. On the other hand, the standardized residuals plot, ACF of residuals plot and Ljung-Box test of ARIMA(7,0,8) suggest that the errors are uncorrelated (Figure 22), however, the QQ Plot (Figure 22) and Shapiro test (7.3.5) both suggests the non-normality of errors, just like the models from north geomagnetic pole.

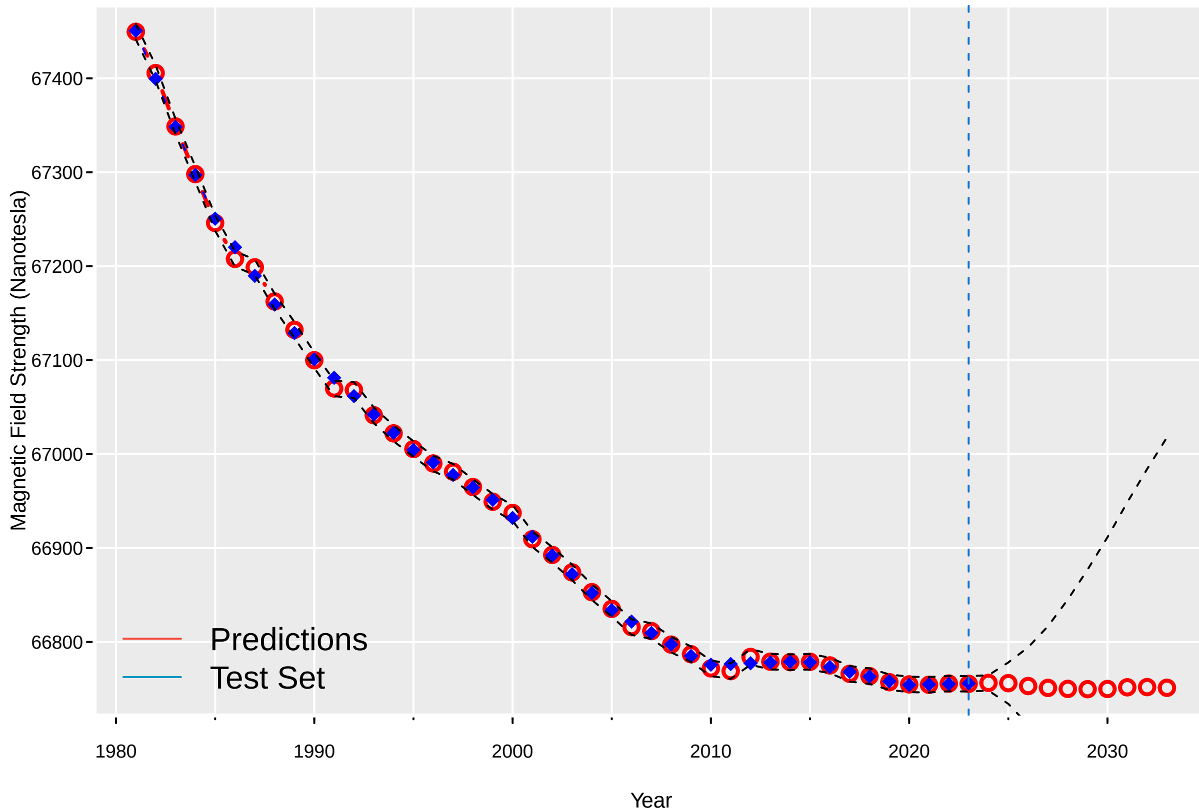
4.3 Forecasting

The best model given as $ARIMA(7,1,8)$ (Section 1.2) was used to predict and compare with the test data. The prediction was done identical to the method presented in section 3.3. The RMSE is 4.10 which is roughly half the RMSE for the north geomagnetic pole model. This highlights the model selected is even better at predictions than the north geomagnetic pole model.

```
RMSES = RMSE(data$South_Geomagnetic_Pole[(nTrain+1):n],x.test.forecast.S[1:nTest])
RMSES
```

```
## [1] 4.10712
```

Below is the plot when the model is used to forecast the south geomagnetic pole strength for the next 10 years (from 2024 to 2023).



The first blue vertical dotted line indicates the year 2023 where our data set ends and the next 10 years of south geomagnetic field strength was plotted with its prediction interval. The prediction interval however, suggests that the south geomagnetic field strength could increase, decrease, or neither, even with small lead values such as two years ahead in 2025. Therefore, there isn't sufficient evidence to conclude anything based on the forecast.

Wide prediction interval starting at early lead values may be an indication that the selected model does not have the normality of error assumption. This will be further discussed below.

5 Discussion

The only thing that is conclusive from this project is that the north magnetic field strength will increase until the year 2028. The lack of conclusive results may be the consequence of both selected models failing the assumption that the errors are normally distributed.

Recall that the prediction interval is derived using the assumption that the white noise terms in ARIMA model arise independently from a normal distribution.

$$\hat{Y}_t(l) \pm z_{1-\alpha/2}\sqrt{Var(e_t(l))} \tag{1}$$

Since the selected models do not pass the tests for normality of errors, the prediction intervals may not be representative of the true prediction interval which is the limitation of the chosen models. Furthermore, seasonal ARIMA model or a periodic model may be better fit for the south geomagnetic pole data since the time series depict a strong seasonal effect (1). Some attempts with the seasonal ARIMA model were made to the south geomagnetic pole data but none gave significant difference to the chosen ARIMA model while periodic models were not attempted.

6 Bibliography

1. Government of Canada, N. R. C. (2019, March 1). Government of Canada / gouvernement du Canada. Government of Canada, Natural Resources Canada, Canadian Hazards Information Service. Retrieved April 9, 2023, from https://geomag.nrcan.gc.ca/mag_fld/sec-en.php
2. Buis, Alan. NASA. (2021, November 16). Earth's magnetosphere: Protecting our planet from harmful space energy – climate change: Vital signs of the planet. NASA. Retrieved April 9, 2023, from https://climate.nasa.gov/news/3105/earths-magnetosphere-protecting-our-planet-from-harmful-space-energy/
3. O'Callaghan, JONATHAN.(2018, December 7). Earth's magnetic poles could start to flip. what happens then? Horizon Magazine. Retrieved April 9, 2023, from https://ec.europa.eu/research-and-innovation/en/horizon-magazine/earths-magnetic-poles-could-start-flip-what-happens-then
4. Holt, Chris. (2021, September 14). When north goes south: Is Earth's magnetic field flipping? Astronomy.com. Retrieved April 9, 2023, from https://astronomy.com/news/2021/09/when-north-goes-south-is-earths-magnetic-field-flipping
5. Stoffer, D. S., & Poison, N. (n.d.).R you kidding. Retrieved April 9, 2023, from https://nickpoison.github.io/

7 Appendices

All the codes used in this report (including the report itself) to generate figures/plots/results can be downloaded using the link below

https://github.com/Faelu/Project

7.1 North Geomagnetic Pole Tests

7.1.1 ADF_KPSS_N1

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -1.8216
##   P VALUE:
##     0.06945
##
## Description:
##  Mon Apr 10 00:29:33 2023 by user: jason
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
```

```
##     Dickey-Fuller: -2.1348
##   P VALUE:
##     0.2623
##
## Description:
##  Mon Apr 10 00:29:33 2023 by user: jason
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -2.8459
##   P VALUE:
##     0.22
##
## Description:
##  Mon Apr 10 00:29:33 2023 by user: jason
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$North_Geomagnetic_Pole[1:nTrain]
## Dickey-Fuller = -2.8459, Lag order = 17, p-value = 0.22
## alternative hypothesis: stationary
```

```
## Warning in kpss.test(data$North_Geomagnetic_Pole[1:nTrain], null = "Level"):
## p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  data$North_Geomagnetic_Pole[1:nTrain]
## KPSS Level = 5.9459, Truncation lag parameter = 5, p-value = 0.01
```

```
## Warning in kpss.test(data$North_Geomagnetic_Pole[1:nTrain], null = "Trend"):
## p-value smaller than printed p-value
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  data$North_Geomagnetic_Pole[1:nTrain]
## KPSS Trend = 0.67056, Truncation lag parameter = 5, p-value = 0.01
```

7.1.2 ADF_KPSS_N2

```
##
## Title:
##  Augmented Dickey-Fuller Test
```

```
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -2.2892
##   P VALUE:
##     0.02246
##
## Description:
##  Mon Apr 10 00:29:33 2023 by user: jason
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -2.7524
##   P VALUE:
##     0.07005
##
## Description:
##  Mon Apr 10 00:29:33 2023 by user: jason
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -2.693
##   P VALUE:
##     0.2845
##
## Description:
##  Mon Apr 10 00:29:33 2023 by user: jason
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diffN1
## Dickey-Fuller = -2.693, Lag order = 17, p-value = 0.2845
## alternative hypothesis: stationary
```

```
## Warning in kpss.test(diffN1, null = "Level"): p-value smaller than printed
## p-value
```

```
##
```

```
##  KPSS Test for Level Stationarity
##
## data:  diffN1
## KPSS Level = 1.1898, Truncation lag parameter = 5, p-value = 0.01

## Warning in kpss.test(diffN1, null = "Trend"): p-value smaller than printed
## p-value

##
##  KPSS Test for Trend Stationarity
##
## data:  diffN1
## KPSS Trend = 1.0953, Truncation lag parameter = 5, p-value = 0.01
```

7.1.3 ADF_KPSS_N3

```
## Warning in adfTest(diffN2, type = "nc", lags = p_max_lag): p-value smaller than
## printed p-value

##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -3.3641
##   P VALUE:
##     0.01
##
## Description:
##  Mon Apr 10 00:29:33 2023 by user: jason

##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -3.4423
##   P VALUE:
##     0.01032
##
## Description:
##  Mon Apr 10 00:29:33 2023 by user: jason

##
## Title:
##  Augmented Dickey-Fuller Test
```

```
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -3.5816
##   P VALUE:
##     0.03489
##
## Description:
##  Mon Apr 10 00:29:33 2023 by user: jason
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diffN2
## Dickey-Fuller = -3.5816, Lag order = 17, p-value = 0.03489
## alternative hypothesis: stationary
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diffN2
## KPSS Level = 0.69819, Truncation lag parameter = 5, p-value = 0.01371
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  diffN2
## KPSS Trend = 0.17089, Truncation lag parameter = 5, p-value = 0.02926
```

7.1.4 ADF_KPSS_N4

```
## Warning in adfTest(diffN2L10, type = "nc", lags = p_max_lag): p-value smaller
## than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -4.7028
##   P VALUE:
##     0.01
##
## Description:
##  Mon Apr 10 00:29:34 2023 by user: jason
```

```
## Warning in adfTest(diffN2L10, type = "c", lags = p_max_lag): p-value smaller
## than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -4.6921
##   P VALUE:
##     0.01
##
## Description:
##  Mon Apr 10 00:29:34 2023 by user: jason
```

```
## Warning in adfTest(diffN2L10, type = "ct", lags = p_max_lag): p-value smaller
## than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -4.7018
##   P VALUE:
##     0.01
##
## Description:
##  Mon Apr 10 00:29:34 2023 by user: jason
```

```
## Warning in adf.test(diffN2L10, k = p_max_lag): p-value smaller than printed
## p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diffN2L10
## Dickey-Fuller = -4.7018, Lag order = 17, p-value = 0.01
## alternative hypothesis: stationary
```

```
## Warning in kpss.test(diffN2L10, null = "Level"): p-value greater than printed
## p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diffN2L10
## KPSS Level = 0.080272, Truncation lag parameter = 5, p-value = 0.1
```

```
## Warning in kpss.test(diffN2L10, null = "Trend"): p-value greater than printed
## p-value
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  diffN2L10
## KPSS Trend = 0.04447, Truncation lag parameter = 5, p-value = 0.1
```

7.1.5 ADF_KPSS_N5

```
## Warning in adfTest(diffNL5D1, type = "nc", lags = p_max_lag): p-value smaller
## than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -2.7097
##   P VALUE:
##     0.01
##
## Description:
##  Mon Apr 10 00:29:34 2023 by user: jason
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -2.7718
##   P VALUE:
##     0.06688
##
## Description:
##  Mon Apr 10 00:29:34 2023 by user: jason
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
```

```
##     Dickey-Fuller: -2.8553
##   P VALUE:
##     0.216
##
## Description:
##  Mon Apr 10 00:29:34 2023 by user: jason
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diffNL5D1
## Dickey-Fuller = -2.8553, Lag order = 17, p-value = 0.216
## alternative hypothesis: stationary
```

```
## Warning in kpss.test(diffNL5D1, null = "Level"): p-value smaller than printed
## p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diffNL5D1
## KPSS Level = 0.88349, Truncation lag parameter = 5, p-value = 0.01
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  diffNL5D1
## KPSS Trend = 0.19421, Truncation lag parameter = 5, p-value = 0.01817
```

7.1.6 N_EACF

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

7.1.7 N_AIC_BIC

```
diffNL5D1.aic=matrix(0,10,10)
diffNL5D1.bic = matrix(0,10,10)
for (i in 0:9) for (j in 0:9){
```

```
  diffNL5D1.fit = arima(diffNL5D1, order = c(i,0,j), method = "ML", include.mean = TRUE)
  diffNL5D1.aic[i+1,j+1] = diffNL5D1.fit$aic
  diffNL5D1.bic[i+1,j+1] = BIC(diffNL5D1.fit)
}
diffNL5D1.aic_vec = sort(unmatrix(diffNL5D1.aic, byrow = FALSE))[1:20]
diffNL5D1.bic_vec = sort(unmatrix(diffNL5D1.bic, byrow = FALSE))[1:20]
diffNL5D1.aic_vec
```

```
##    r8:c7    r8:c8    r9:c8   r10:c9    r9:c9   r9:c10  r10:c10   r7:c10
## 1970.646 1971.024 1972.513 1974.390 1974.560 1978.575 1979.353 1979.814
##    r4:c7    r5:c8    r2:c7   r2:c10    r6:c5    r4:c6    r5:c6    r8:c9
## 1983.841 1985.590 1985.967 1986.475 1986.495 1986.504 1986.668 1986.763
##    r9:c6    r9:c7    r2:c8    r4:c8
## 1987.125 1987.423 1987.633 1987.754
```

```
diffNL5D1.bic_vec
```

```
##    r2:c5    r2:c7    r2:c6    r3:c5    r1:c7    r1:c6    r4:c6    r2:c8
## 2019.534 2023.546 2025.259 2025.303 2025.816 2025.988 2028.037 2029.165
##    r3:c7    r4:c7    r4:c5    r5:c5    r3:c6    r1:c8    r8:c7    r6:c5
## 2029.308 2029.327 2030.400 2031.015 2031.162 2031.643 2031.944 2031.980
##    r5:c6    r2:c9   r1:c10    r3:c8
## 2032.154 2033.579 2034.767 2035.354
```

7.1.8 N_Shapiro

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(diffNL5D1.fit_55)
## W = 0.66271, p-value < 2.2e-16
```

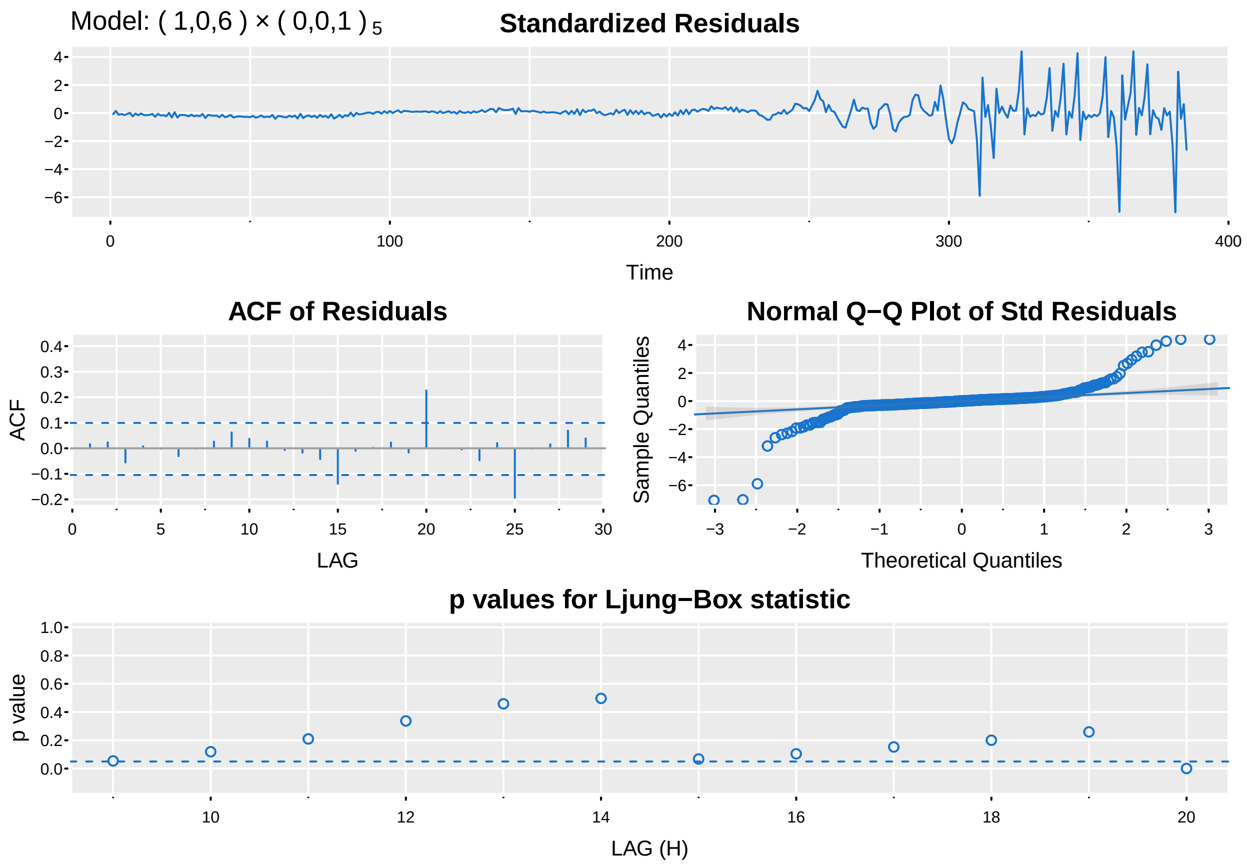
7.1.9 N_AIC_BIC_ALL

```
diffNL5D1.fit_S1 = sarima(diffNL5D1,1,0,6,0,0,1,5, gg = TRUE, col = 4)
```

```
## initial  value 2.352973
## iter   2 value 1.651019
## iter   3 value 1.475085
## iter   4 value 1.295902
## iter   5 value 1.248902
## iter   6 value 1.184541
## iter   7 value 1.183703
## iter   8 value 1.147947
## iter   9 value 1.142581
## iter  10 value 1.134367
## iter  11 value 1.119696
## iter  12 value 1.115916
## iter  13 value 1.110204
```

```
## iter  14 value 1.106559
## iter  15 value 1.105584
## iter  16 value 1.104311
## iter  17 value 1.103421
## iter  18 value 1.102701
## iter  19 value 1.101779
## iter  20 value 1.101077
## iter  21 value 1.100577
## iter  22 value 1.100544
## iter  23 value 1.099287
## iter  24 value 1.098841
## iter  25 value 1.098545
## iter  26 value 1.097941
## iter  27 value 1.097208
## iter  28 value 1.097092
## iter  29 value 1.097075
## iter  30 value 1.096705
## iter  30 value 1.096705
## iter  31 value 1.096703
## iter  32 value 1.096702
## iter  33 value 1.096700
## iter  34 value 1.096696
## iter  35 value 1.096695
## iter  35 value 1.096695
## iter  35 value 1.096695
## final  value 1.096695
## converged
## initial  value 1.148948
## iter   2 value 1.142339
## iter   3 value 1.140233
## iter   4 value 1.139871
## iter   5 value 1.139037
## iter   6 value 1.138551
## iter   7 value 1.138330
## iter   8 value 1.138305
## iter   9 value 1.138303
## iter  10 value 1.138303
## iter  11 value 1.138302
## iter  12 value 1.138300
## iter  13 value 1.138294
## iter  14 value 1.138282
## iter  15 value 1.138263
## iter  16 value 1.138246
## iter  17 value 1.138232
## iter  18 value 1.138199
## iter  19 value 1.138138
## iter  20 value 1.137991
## iter  21 value 1.137735
## iter  22 value 1.137716
## iter  23 value 1.137707
## iter  24 value 1.137644
## iter  25 value 1.137639
## iter  26 value 1.137593
## iter  27 value 1.137572
```

```
## iter  28 value 1.137416
## iter  29 value 1.137293
## iter  30 value 1.137201
## iter  31 value 1.137169
## iter  32 value 1.137141
## iter  33 value 1.137083
## iter  34 value 1.136945
## iter  35 value 1.136735
## iter  36 value 1.136414
## iter  37 value 1.135668
## iter  38 value 1.135413
## iter  39 value 1.135151
## iter  40 value 1.135140
## iter  41 value 1.135129
## iter  42 value 1.135083
## iter  43 value 1.135077
## iter  44 value 1.135073
## iter  45 value 1.135062
## iter  46 value 1.135056
## iter  47 value 1.135028
## iter  48 value 1.135000
## iter  49 value 1.134963
## iter  50 value 1.134947
## iter  51 value 1.134943
## iter  52 value 1.134942
## iter  53 value 1.134942
## iter  54 value 1.134940
## iter  55 value 1.134938
## iter  56 value 1.134934
## iter  57 value 1.134932
## iter  58 value 1.134931
## iter  58 value 1.134931
## final  value 1.134931
## converged
```

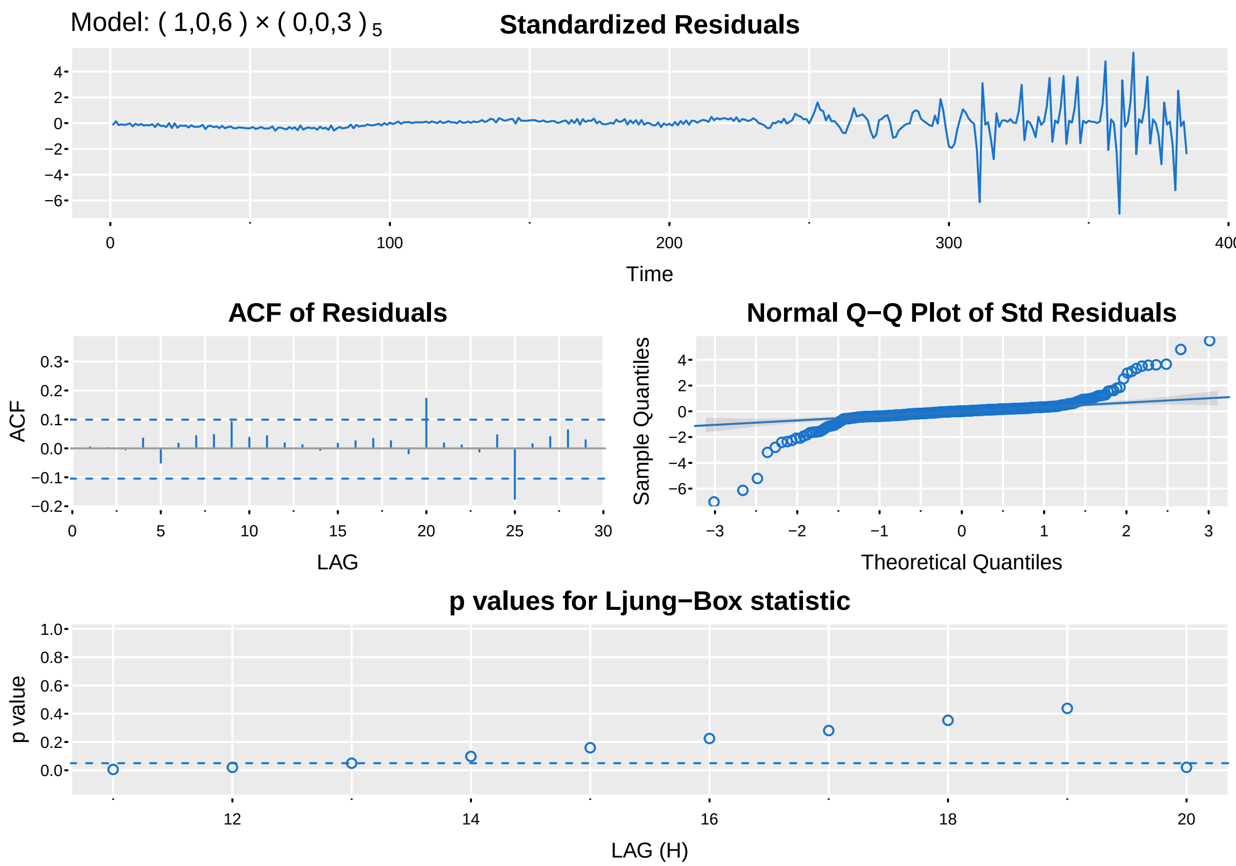



```
diffNL5D1.fit_S2 = sarima(diffNL5D1,1,0,6,0,0,3,5, gg = TRUE, col = 4)
```

```
## initial  value 2.352973
## iter   2 value 1.599254
## iter   3 value 1.466948
## iter   4 value 1.317826
## iter   5 value 1.293723
## iter   6 value 1.253760
## iter   7 value 1.242887
## iter   8 value 1.229747
## iter   9 value 1.184388
## iter  10 value 1.126612
## iter  11 value 1.120952
## iter  12 value 1.102907
## iter  13 value 1.101120
## iter  14 value 1.099271
## iter  15 value 1.091253
## iter  16 value 1.090043
## iter  17 value 1.089831
## iter  18 value 1.088393
## iter  19 value 1.086850
## iter  20 value 1.085236
## iter  21 value 1.084544
## iter  22 value 1.084094
## iter  23 value 1.083226
```

```
## iter  24 value 1.082677
## iter  25 value 1.081749
## iter  26 value 1.081109
## iter  27 value 1.079423
## iter  28 value 1.079370
## iter  29 value 1.079314
## iter  30 value 1.079303
## iter  31 value 1.078987
## iter  32 value 1.078486
## iter  33 value 1.078042
## iter  34 value 1.077687
## iter  35 value 1.077672
## iter  36 value 1.077669
## iter  36 value 1.077669
## iter  37 value 1.077648
## iter  38 value 1.077581
## iter  39 value 1.077525
## iter  39 value 1.077525
## iter  39 value 1.077525
## final  value 1.077525
## converged
## initial  value 1.131095
## iter   2 value 1.124769
## iter   3 value 1.122157
## iter   4 value 1.121834
## iter   5 value 1.121114
## iter   6 value 1.120772
## iter   7 value 1.120562
## iter   8 value 1.120504
## iter   9 value 1.120483
## iter  10 value 1.120479
## iter  11 value 1.120478
## iter  12 value 1.120476
## iter  13 value 1.120474
## iter  14 value 1.120470
## iter  15 value 1.120463
## iter  16 value 1.120457
## iter  17 value 1.120454
## iter  18 value 1.120451
## iter  19 value 1.120447
## iter  20 value 1.120438
## iter  21 value 1.120414
## iter  22 value 1.120365
## iter  23 value 1.120310
## iter  24 value 1.120252
## iter  25 value 1.120251
## iter  26 value 1.120249
## iter  27 value 1.120247
## iter  28 value 1.120239
## iter  29 value 1.120238
## iter  30 value 1.120238
## iter  31 value 1.120237
## iter  32 value 1.120237
## iter  33 value 1.120237
```

```
## iter  34 value 1.120236
## iter  35 value 1.120236
## iter  36 value 1.120236
## iter  36 value 1.120236
## iter  36 value 1.120236
## final  value 1.120236
## converged
```



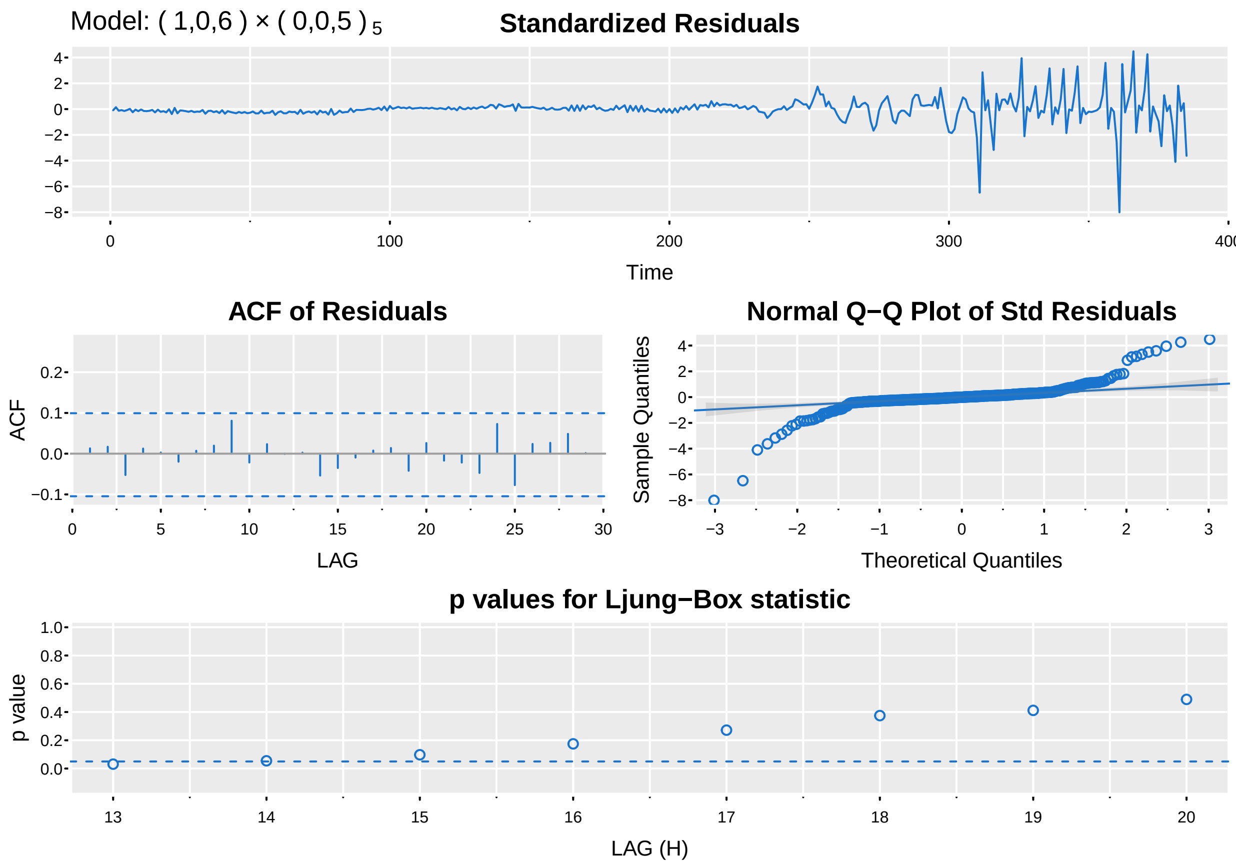

```
diffNL5D1.fit_S3 = sarima(diffNL5D1,1,0,6,0,0,5,5, gg = TRUE, col = 4)
```

```
## initial  value 2.352973
## iter   2 value 1.551905
## iter   3 value 1.417133
## iter   4 value 1.303916
## iter   5 value 1.195191
## iter   6 value 1.175445
## iter   7 value 1.156742
## iter   8 value 1.133940
## iter   9 value 1.129478
## iter  10 value 1.121695
## iter  11 value 1.101225
## iter  12 value 1.091007
## iter  13 value 1.089754
## iter  14 value 1.083232
## iter  15 value 1.072735
```

```
## iter  16 value 1.056172
## iter  17 value 1.050095
## iter  18 value 1.049433
## iter  19 value 1.047051
## iter  20 value 1.041048
## iter  21 value 1.039715
## iter  22 value 1.038697
## iter  23 value 1.038437
## iter  24 value 1.038309
## iter  25 value 1.038299
## iter  26 value 1.038282
## iter  27 value 1.038265
## iter  28 value 1.038250
## iter  29 value 1.038244
## iter  30 value 1.038242
## iter  31 value 1.038239
## iter  32 value 1.038235
## iter  33 value 1.038229
## iter  34 value 1.038218
## iter  35 value 1.038218
## iter  36 value 1.038218
## iter  37 value 1.038218
## iter  38 value 1.038217
## iter  39 value 1.038214
## iter  40 value 1.038206
## iter  41 value 1.038204
## iter  42 value 1.038201
## iter  43 value 1.038199
## iter  44 value 1.038193
## iter  45 value 1.038176
## iter  46 value 1.038138
## iter  47 value 1.038068
## iter  48 value 1.037954
## iter  49 value 1.037774
## iter  50 value 1.037742
## iter  51 value 1.037730
## iter  52 value 1.037712
## iter  53 value 1.037703
## iter  54 value 1.037696
## iter  55 value 1.037691
## iter  56 value 1.037688
## iter  57 value 1.037687
## iter  58 value 1.037687
## iter  58 value 1.037687
## iter  59 value 1.037687
## iter  59 value 1.037687
## iter  59 value 1.037687
## final  value 1.037687
## converged
## initial  value 1.077269
## iter   2 value 1.073298
## iter   3 value 1.072847
## iter   4 value 1.072649
## iter   5 value 1.072529
```

```
## iter   6 value 1.072336
## iter   7 value 1.072154
## iter   8 value 1.072121
## iter   9 value 1.072103
## iter  10 value 1.072091
## iter  11 value 1.072076
## iter  12 value 1.072051
## iter  13 value 1.072024
## iter  14 value 1.072007
## iter  15 value 1.072003
## iter  16 value 1.072002
## iter  17 value 1.072002
## iter  18 value 1.072001
## iter  19 value 1.071999
## iter  20 value 1.071994
## iter  21 value 1.071982
## iter  22 value 1.071952
## iter  23 value 1.071893
## iter  24 value 1.071827
## iter  25 value 1.071652
## iter  26 value 1.071650
## iter  27 value 1.071498
## iter  28 value 1.071449
## iter  29 value 1.071449
## iter  30 value 1.071445
## iter  31 value 1.071442
## iter  32 value 1.071440
## iter  33 value 1.071439
## iter  34 value 1.071435
## iter  35 value 1.071434
## iter  36 value 1.071432
## iter  37 value 1.071428
## iter  38 value 1.071414
## iter  39 value 1.071386
## iter  40 value 1.071321
## iter  41 value 1.071261
## iter  42 value 1.071063
## iter  43 value 1.070989
## iter  44 value 1.070722
## iter  45 value 1.070586
## iter  46 value 1.070556
## iter  47 value 1.070516
## iter  48 value 1.070483
## iter  49 value 1.070476
## iter  50 value 1.070461
## iter  51 value 1.070443
## iter  52 value 1.070421
## iter  53 value 1.070398
## iter  54 value 1.070387
## iter  55 value 1.070385
## iter  56 value 1.070385
## iter  56 value 1.070385
## iter  56 value 1.070385
## final  value 1.070385
```

```
## converged
```



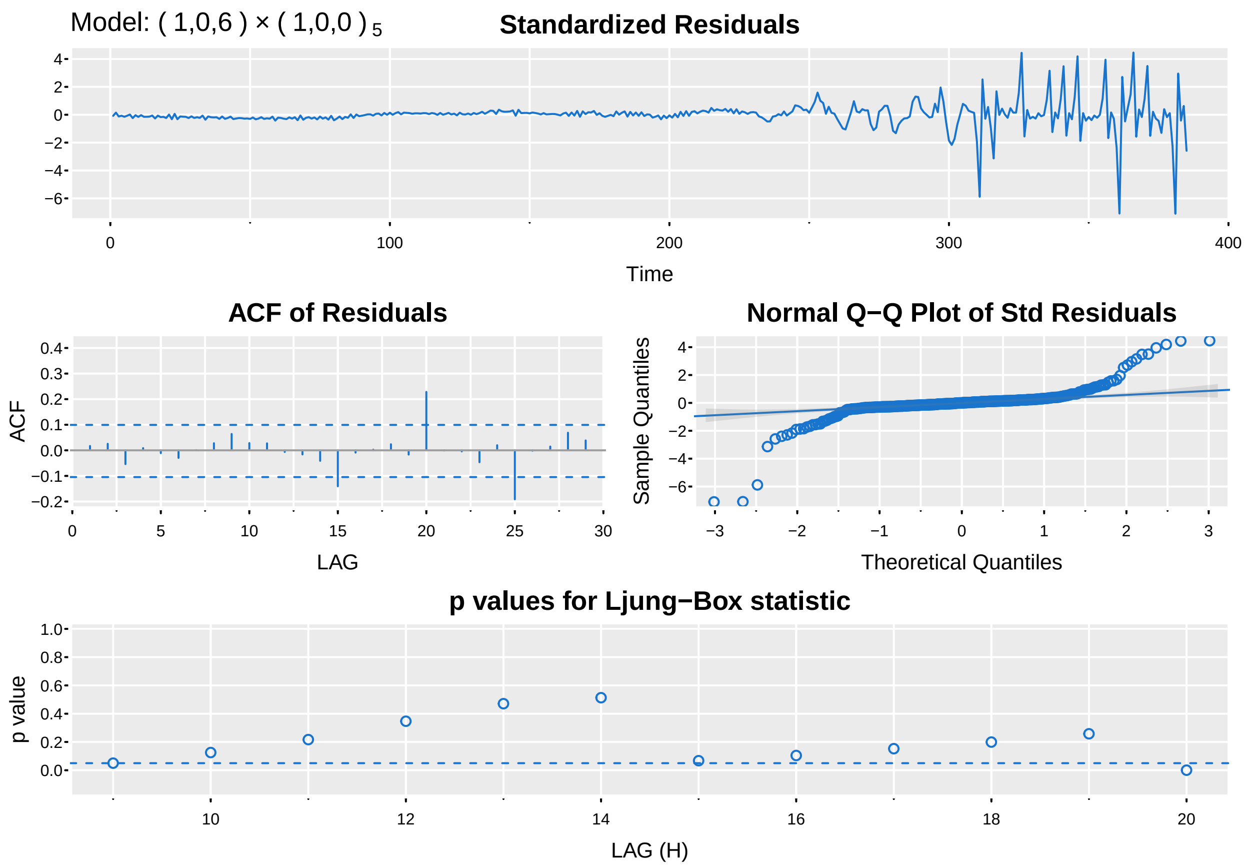

```
diffNL5D1.fit_S4 = sarima(diffNL5D1,1,0,6,1,0,0,5, gg = TRUE, col = 4)
```

```
## initial  value 2.359416
## iter   2 value 1.623644
## iter   3 value 1.469564
## iter   4 value 1.290698
## iter   5 value 1.235473
## iter   6 value 1.230362
## iter   7 value 1.204635
## iter   8 value 1.180880
## iter   9 value 1.170136
## iter  10 value 1.155512
## iter  11 value 1.151805
## iter  12 value 1.139280
## iter  13 value 1.138595
## iter  14 value 1.137975
## iter  15 value 1.135548
## iter  16 value 1.135160
## iter  17 value 1.131213
## iter  18 value 1.129097
## iter  19 value 1.126976
## iter  20 value 1.125304
## iter  21 value 1.124567
```

```
## iter  22 value 1.124534
## iter  23 value 1.124518
## iter  24 value 1.123998
## iter  25 value 1.123734
## iter  26 value 1.122349
## iter  27 value 1.119907
## iter  28 value 1.119377
## iter  29 value 1.115902
## iter  30 value 1.115217
## iter  31 value 1.114029
## iter  32 value 1.112230
## iter  33 value 1.111785
## iter  34 value 1.110644
## iter  35 value 1.109731
## iter  36 value 1.108437
## iter  37 value 1.107105
## iter  38 value 1.106016
## iter  39 value 1.105974
## iter  40 value 1.105826
## iter  41 value 1.104607
## iter  42 value 1.104590
## iter  43 value 1.104463
## iter  44 value 1.104385
## iter  45 value 1.103295
## iter  46 value 1.102780
## iter  47 value 1.101927
## iter  48 value 1.101498
## iter  49 value 1.100875
## iter  50 value 1.099481
## iter  51 value 1.098462
## iter  52 value 1.097901
## iter  53 value 1.097756
## iter  54 value 1.096877
## iter  55 value 1.096848
## iter  55 value 1.096848
## iter  56 value 1.096602
## iter  57 value 1.096569
## iter  58 value 1.096527
## iter  59 value 1.095787
## iter  60 value 1.095769
## iter  61 value 1.095764
## iter  62 value 1.095743
## iter  63 value 1.095733
## iter  63 value 1.095733
## iter  64 value 1.095727
## iter  65 value 1.095720
## iter  65 value 1.095720
## iter  66 value 1.095719
## iter  66 value 1.095719
## iter  67 value 1.095719
## iter  67 value 1.095719
## iter  67 value 1.095719
## final  value 1.095719
## converged
```

```
## initial  value 1.155717
## iter   2 value 1.153874
## iter   3 value 1.151922
## iter   4 value 1.150609
## iter   5 value 1.147011
## iter   6 value 1.145352
## iter   7 value 1.143553
## iter   8 value 1.142770
## iter   9 value 1.142248
## iter  10 value 1.140739
## iter  11 value 1.138151
## iter  12 value 1.137072
## iter  13 value 1.136621
## iter  14 value 1.136608
## iter  15 value 1.136606
## iter  16 value 1.136606
## iter  17 value 1.136605
## iter  18 value 1.136603
## iter  19 value 1.136596
## iter  20 value 1.136578
## iter  21 value 1.136566
## iter  22 value 1.136564
## iter  23 value 1.136560
## iter  24 value 1.136548
## iter  25 value 1.136544
## iter  26 value 1.136542
## iter  27 value 1.136541
## iter  28 value 1.136521
## iter  29 value 1.136469
## iter  30 value 1.136389
## iter  31 value 1.136281
## iter  32 value 1.136164
## iter  33 value 1.135312
## iter  34 value 1.134726
## iter  35 value 1.134644
## iter  36 value 1.134555
## iter  37 value 1.134503
## iter  38 value 1.134456
## iter  39 value 1.134421
## iter  40 value 1.134418
## iter  41 value 1.134417
## iter  42 value 1.134412
## iter  43 value 1.134408
## iter  44 value 1.134408
## iter  45 value 1.134407
## iter  46 value 1.134407
## iter  47 value 1.134407
## iter  48 value 1.134406
## iter  49 value 1.134406
## iter  50 value 1.134406
## iter  51 value 1.134405
## iter  52 value 1.134405
## iter  53 value 1.134405
## iter  53 value 1.134405
```

```
## iter  53 value 1.134405
## final  value 1.134405
## converged
```

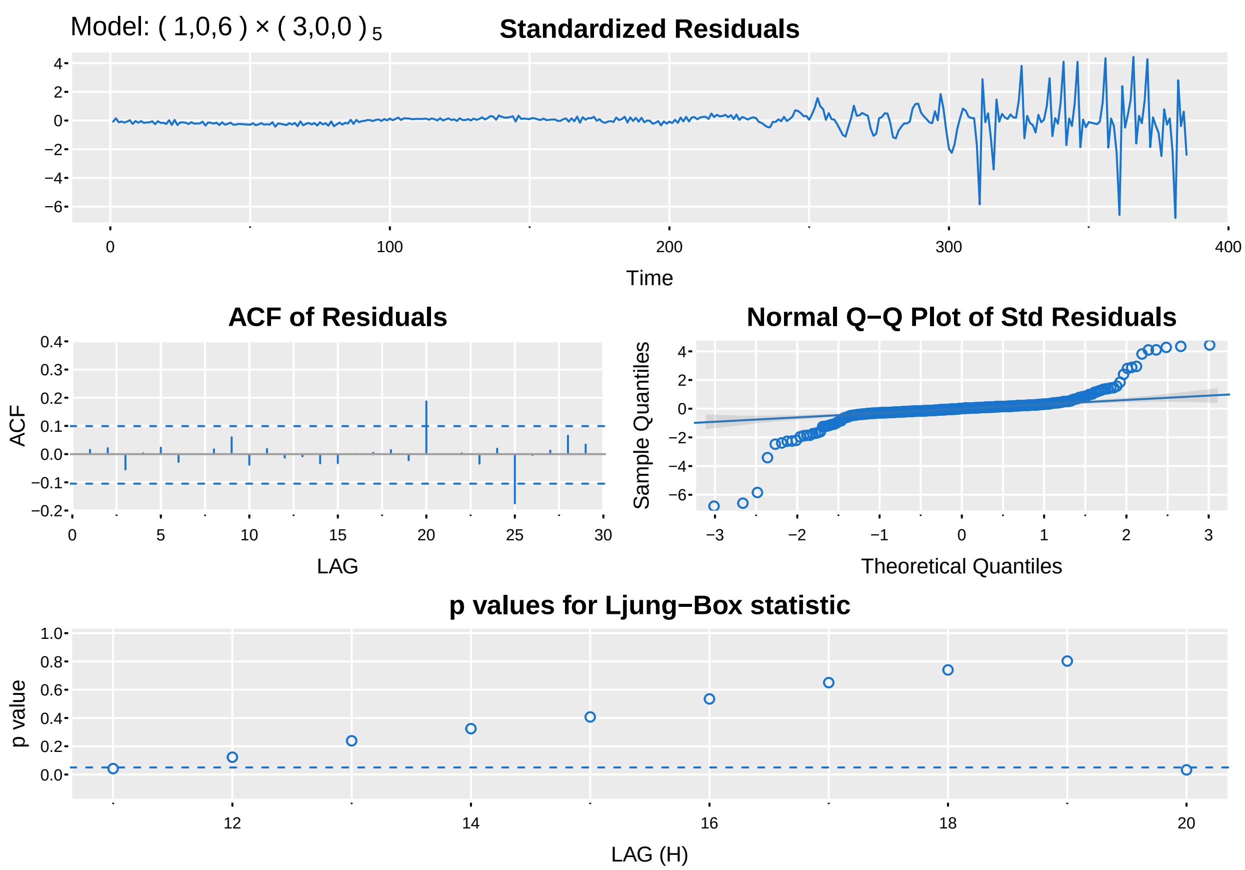



```
diffNL5D1.fit_S5 = sarima(diffNL5D1,1,0,6,3,0,0,5, gg = TRUE, col = 4)
```

```
## initial  value 2.371341
## iter   2 value 1.621285
## iter   3 value 1.496923
## iter   4 value 1.354090
## iter   5 value 1.271548
## iter   6 value 1.209786
## iter   7 value 1.203302
## iter   8 value 1.195402
## iter   9 value 1.178559
## iter  10 value 1.176489
## iter  11 value 1.173813
## iter  12 value 1.166562
## iter  13 value 1.160901
## iter  14 value 1.157469
## iter  15 value 1.155296
## iter  16 value 1.154148
## iter  17 value 1.151894
## iter  18 value 1.145744
## iter  19 value 1.143171
```

```
## iter  20 value 1.142766
## iter  21 value 1.142019
## iter  22 value 1.140584
## iter  23 value 1.136864
## iter  24 value 1.135906
## iter  25 value 1.135737
## iter  26 value 1.135648
## iter  27 value 1.135589
## iter  28 value 1.135237
## iter  29 value 1.134000
## iter  30 value 1.132382
## iter  31 value 1.131926
## iter  32 value 1.131098
## iter  33 value 1.130833
## iter  34 value 1.130471
## iter  35 value 1.130234
## iter  36 value 1.129432
## iter  37 value 1.129191
## iter  38 value 1.128798
## iter  39 value 1.127744
## iter  40 value 1.127501
## iter  41 value 1.126631
## iter  42 value 1.126226
## iter  43 value 1.125955
## iter  44 value 1.125856
## iter  45 value 1.125845
## iter  46 value 1.125826
## iter  47 value 1.125800
## iter  48 value 1.125736
## iter  49 value 1.125685
## iter  50 value 1.125667
## iter  51 value 1.125663
## iter  52 value 1.125663
## iter  52 value 1.125663
## iter  52 value 1.125663
## final  value 1.125663
## converged
## initial  value 1.127560
## iter   2 value 1.126704
## iter   3 value 1.126626
## iter   4 value 1.126617
## iter   5 value 1.126465
## iter   6 value 1.126455
## iter   7 value 1.126406
## iter   8 value 1.126372
## iter   9 value 1.126368
## iter  10 value 1.126361
## iter  11 value 1.126343
## iter  12 value 1.126317
## iter  13 value 1.126293
## iter  14 value 1.126284
## iter  15 value 1.126283
## iter  16 value 1.126283
## iter  17 value 1.126283
```

```
## iter  18 value 1.126283
## iter  19 value 1.126283
## iter  20 value 1.126282
## iter  21 value 1.126282
## iter  22 value 1.126282
## iter  22 value 1.126282
## iter  22 value 1.126282
## final  value 1.126282
## converged
```

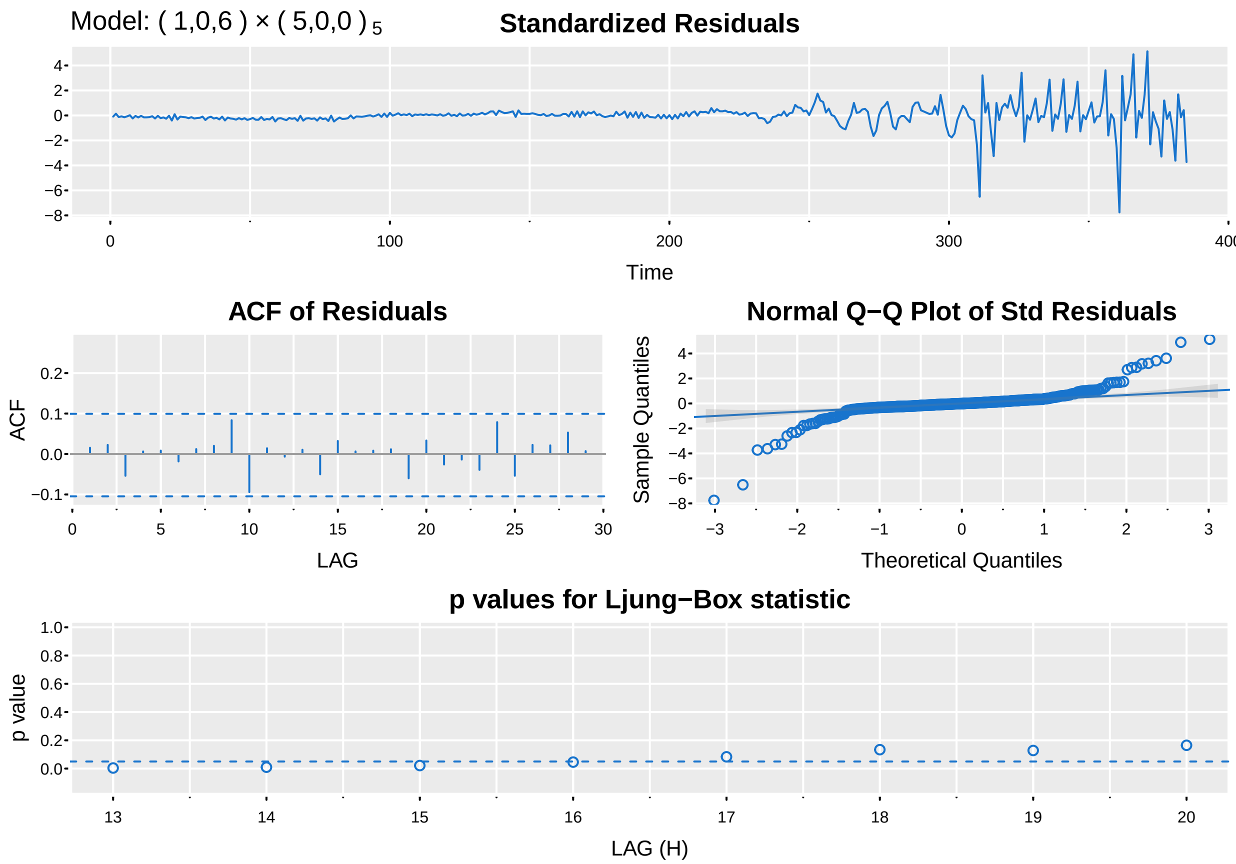



```
diffNL5D1.fit_S6 = sarima(diffNL5D1,1,0,6,5,0,0,5, gg = TRUE, col = 4)
```

```
## initial  value 2.381593
## iter   2 value 1.582389
## iter   3 value 1.457767
## iter   4 value 1.324028
## iter   5 value 1.221426
## iter   6 value 1.177718
## iter   7 value 1.160135
## iter   8 value 1.155454
## iter   9 value 1.148034
## iter  10 value 1.130966
## iter  11 value 1.122324
## iter  12 value 1.112563
## iter  13 value 1.091689
```

```
## iter  14 value 1.088862
## iter  15 value 1.086782
## iter  16 value 1.085431
## iter  17 value 1.085027
## iter  18 value 1.083104
## iter  19 value 1.082460
## iter  20 value 1.082076
## iter  21 value 1.080799
## iter  22 value 1.079919
## iter  23 value 1.078216
## iter  24 value 1.077117
## iter  25 value 1.076607
## iter  26 value 1.076134
## iter  27 value 1.076058
## iter  28 value 1.075870
## iter  29 value 1.075215
## iter  30 value 1.073687
## iter  31 value 1.073085
## iter  32 value 1.072363
## iter  33 value 1.071918
## iter  34 value 1.070929
## iter  35 value 1.070824
## iter  36 value 1.070792
## iter  37 value 1.070669
## iter  38 value 1.070656
## iter  39 value 1.070644
## iter  40 value 1.070638
## iter  41 value 1.070636
## iter  42 value 1.070206
## iter  43 value 1.069781
## iter  44 value 1.069754
## iter  45 value 1.069750
## iter  46 value 1.069699
## iter  47 value 1.069654
## iter  48 value 1.069640
## iter  49 value 1.069636
## iter  50 value 1.069629
## iter  51 value 1.069612
## iter  52 value 1.069604
## iter  53 value 1.069601
## iter  54 value 1.069601
## iter  55 value 1.069601
## iter  56 value 1.069601
## iter  57 value 1.069601
## iter  58 value 1.069601
## iter  59 value 1.069600
## iter  60 value 1.069600
## iter  60 value 1.069600
## iter  60 value 1.069600
## final  value 1.069600
## converged
## initial  value 1.072804
## iter   2 value 1.066797
## iter   3 value 1.066301
```

```
## iter   4 value 1.065786
## iter   5 value 1.064929
## iter   6 value 1.064275
## iter   7 value 1.064181
## iter   8 value 1.064056
## iter   9 value 1.064014
## iter  10 value 1.063999
## iter  11 value 1.063989
## iter  12 value 1.063970
## iter  13 value 1.063943
## iter  14 value 1.063907
## iter  15 value 1.063880
## iter  16 value 1.063869
## iter  17 value 1.063867
## iter  18 value 1.063866
## iter  19 value 1.063865
## iter  20 value 1.063865
## iter  21 value 1.063864
## iter  22 value 1.063862
## iter  23 value 1.063858
## iter  24 value 1.063854
## iter  25 value 1.063850
## iter  26 value 1.063849
## iter  27 value 1.063849
## iter  27 value 1.063849
## iter  27 value 1.063849
## final  value 1.063849
## converged
```



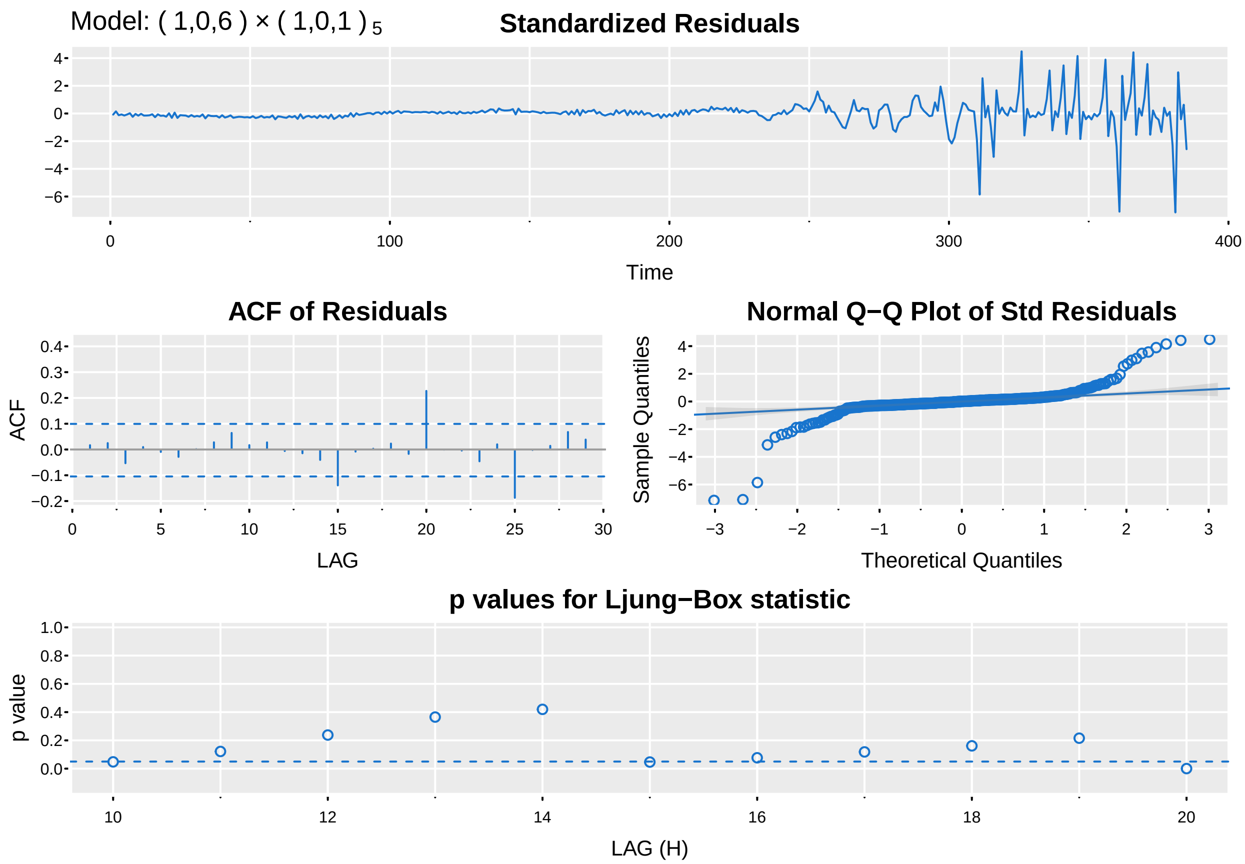

```
diffNL5D1.fit_S7 = sarima(diffNL5D1,1,0,6,1,0,1,5, gg = TRUE, col = 4)
```

```
## initial  value 2.359416
## iter   2 value 1.864375
## iter   3 value 1.609842
## iter   4 value 1.336221
## iter   5 value 1.320490
## iter   6 value 1.279999
## iter   7 value 1.220238
## iter   8 value 1.208193
## iter   9 value 1.187656
## iter  10 value 1.170222
## iter  11 value 1.162289
## iter  12 value 1.153137
## iter  13 value 1.140381
## iter  14 value 1.137886
## iter  15 value 1.135245
## iter  16 value 1.134534
## iter  17 value 1.131776
## iter  18 value 1.131621
## iter  19 value 1.130018
## iter  20 value 1.129040
## iter  21 value 1.125552
## iter  22 value 1.122434
## iter  23 value 1.121928
```

```
## iter  24 value 1.120490
## iter  25 value 1.118294
## iter  26 value 1.117620
## iter  27 value 1.117255
## iter  28 value 1.116750
## iter  29 value 1.116615
## iter  30 value 1.116024
## iter  31 value 1.115768
## iter  32 value 1.115244
## iter  33 value 1.114746
## iter  34 value 1.112902
## iter  35 value 1.111594
## iter  36 value 1.111461
## iter  37 value 1.111257
## iter  38 value 1.110928
## iter  39 value 1.110281
## iter  40 value 1.109220
## iter  41 value 1.107543
## iter  42 value 1.107041
## iter  43 value 1.106417
## iter  44 value 1.105821
## iter  45 value 1.105165
## iter  46 value 1.104270
## iter  47 value 1.103730
## iter  48 value 1.103400
## iter  49 value 1.102496
## iter  50 value 1.102240
## iter  51 value 1.101801
## iter  52 value 1.100294
## iter  53 value 1.099505
## iter  54 value 1.098802
## iter  55 value 1.098181
## iter  56 value 1.098093
## iter  57 value 1.098006
## iter  58 value 1.097439
## iter  59 value 1.097419
## iter  59 value 1.097419
## iter  60 value 1.096930
## iter  61 value 1.096928
## iter  62 value 1.096871
## iter  63 value 1.096825
## iter  64 value 1.096796
## iter  65 value 1.096762
## iter  66 value 1.096761
## iter  66 value 1.096761
## iter  67 value 1.096761
## iter  68 value 1.096760
## iter  68 value 1.096760
## iter  68 value 1.096760
## final  value 1.096760
## converged
## initial  value 1.155220
## iter   2 value 1.153635
## iter   3 value 1.151998
```

```
## iter   4 value 1.150707
## iter   5 value 1.147330
## iter   6 value 1.144543
## iter   7 value 1.143134
## iter   8 value 1.142495
## iter   9 value 1.141859
## iter  10 value 1.140132
## iter  11 value 1.137633
## iter  12 value 1.136450
## iter  13 value 1.136097
## iter  14 value 1.136075
## iter  15 value 1.136074
## iter  16 value 1.136073
## iter  17 value 1.136073
## iter  18 value 1.136073
## iter  19 value 1.136071
## iter  20 value 1.136066
## iter  21 value 1.136054
## iter  22 value 1.136037
## iter  23 value 1.136025
## iter  24 value 1.136004
## iter  25 value 1.135980
## iter  26 value 1.135967
## iter  27 value 1.135959
## iter  28 value 1.135950
## iter  29 value 1.135944
## iter  30 value 1.135943
## iter  31 value 1.135937
## iter  32 value 1.135922
## iter  33 value 1.135894
## iter  34 value 1.135855
## iter  35 value 1.135806
## iter  36 value 1.135750
## iter  37 value 1.135690
## iter  38 value 1.135643
## iter  39 value 1.135547
## iter  40 value 1.135413
## iter  41 value 1.135143
## iter  42 value 1.135052
## iter  43 value 1.134888
## iter  44 value 1.134847
## iter  45 value 1.134810
## iter  46 value 1.134779
## iter  47 value 1.134772
## iter  48 value 1.134764
## iter  49 value 1.134721
## iter  50 value 1.134664
## iter  51 value 1.134446
## iter  52 value 1.134330
## iter  53 value 1.134275
## iter  54 value 1.134265
## iter  55 value 1.134264
## iter  56 value 1.134264
## iter  57 value 1.134264
```

```
## iter  58 value 1.134264
## iter  59 value 1.134263
## iter  60 value 1.134263
## iter  61 value 1.134262
## iter  62 value 1.134262
## iter  63 value 1.134262
## iter  63 value 1.134262
## iter  63 value 1.134262
## final  value 1.134262
## converged
```




```
c(diffNL5D1.fit_S1$fit$aic,diffNL5D1.fit_S2$fit$aic,diffNL5D1.fit_S3$fit$aic,diffNL5D1.fit_S4$fit$aic,d
```

```
## [1] 1986.480 1979.164 1944.779 1986.074 1983.820 1939.746 1987.964
```

```
diffNL5D1.fit_S1$fit
```

```
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
```

```
##          ar1     ma1     ma2     ma3     ma4      ma5      ma6    sma1    xmean
##       0.9261  0.4650  0.1892  0.1832  0.1819  -0.8159  -0.2780  0.1021  -1.8742
## s.e.  0.0615  0.0833  0.1116  0.1132  0.1133   0.1122   0.0618  0.0529   2.1021
##
## sigma^2 estimated as 9.202:  log likelihood = -983.24,  aic = 1986.48
```

```
diffNL5D1.fit_S2$fit
```

```
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4     ma5      ma6    sma1    sma2
##       0.3526  1.0696  1.0510  1.0523  1.0523  0.0522  -0.0173  0.2802  0.0165
## s.e.  0.4435  0.4497  0.6311  0.6344  0.6368  0.6369   0.1910  0.0599  0.0540
##          sma3    xmean
##       -0.2053  -1.5306
## s.e.   0.0664   1.3463
##
## sigma^2 estimated as 8.91:  log likelihood = -977.58,  aic = 1979.16
```

```
diffNL5D1.fit_S3$fit
```

```
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ma1     ma2     ma3    ma4      ma5      ma6    sma1    sma2
##       0.8609  0.5376  0.3020  0.2971  0.295  -0.7030  -0.2372  0.1489  0.0145
## s.e.  0.1186  0.1345  0.1906  0.1926  0.193   0.1916   0.0783  0.0661  0.0600
##          sma3    sma4     sma5    xmean
##       -0.0160  0.3215  -0.1466  -1.7412
## s.e.   0.0703  0.0605   0.0761   2.0158
##
## sigma^2 estimated as 8.042:  log likelihood = -958.39,  aic = 1944.78
```

```
diffNL5D1.fit_S4$fit
```

```
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4      ma5      ma6    sar1    xmean
```

```
##       0.9226  0.4677  0.1916  0.1853  0.1840  -0.8138  -0.2786  0.1172  -1.8495
## s.e.  0.0685  0.0895  0.1220  0.1237  0.1237   0.1226   0.0639  0.0597   2.0877
##
## sigma^2 estimated as 9.197:  log likelihood = -983.04,  aic = 1986.07
```

```
diffNL5D1.fit_S5$fit
```

```
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4      ma5      ma6    sar1    sar2
##       0.9443  0.4430  0.1600  0.1544  0.1530  -0.8448  -0.2848  0.1269  0.0332
## s.e.  0.0447  0.0692  0.0851  0.0863  0.0863   0.0854   0.0568  0.0589  0.0569
##          sar3    xmean
##       -0.1434  -1.8933
## s.e.   0.0581   2.1436
##
## sigma^2 estimated as 9.022:  log likelihood = -979.91,  aic = 1983.82
```

```
diffNL5D1.fit_S6$fit
```

```
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4      ma5      ma6    sar1     sar2
##       0.8987  0.5059  0.2466  0.2417  0.2385  -0.7589  -0.2600  0.1599  -0.0245
## s.e.  0.1012  0.1203  0.1748  0.1769  0.1772   0.1758   0.0781  0.0555   0.0545
##          sar3    sar4     sar5    xmean
##       -0.0658  0.3224  -0.3041  -1.6938
## s.e.   0.0577  0.0561   0.0615   1.8681
##
## sigma^2 estimated as 7.891:  log likelihood = -955.87,  aic = 1939.75
```

```
diffNL5D1.fit_S7$fit
```

```
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4      ma5      ma6    sar1     sma1
##       0.9188  0.4710  0.1970  0.1906  0.1894  -0.8084  -0.2766  0.2227  -0.1046
```

```
## s.e.  0.0760  0.0958  0.1325  0.1343  0.1344   0.1332   0.0663  0.3380   0.3377
##         xmean
##       -1.8602
## s.e.   2.0809
##
## sigma^2 estimated as 9.198:  log likelihood = -982.98,  aic = 1987.96
```

7.2 North Geomagnetic Pole Figures

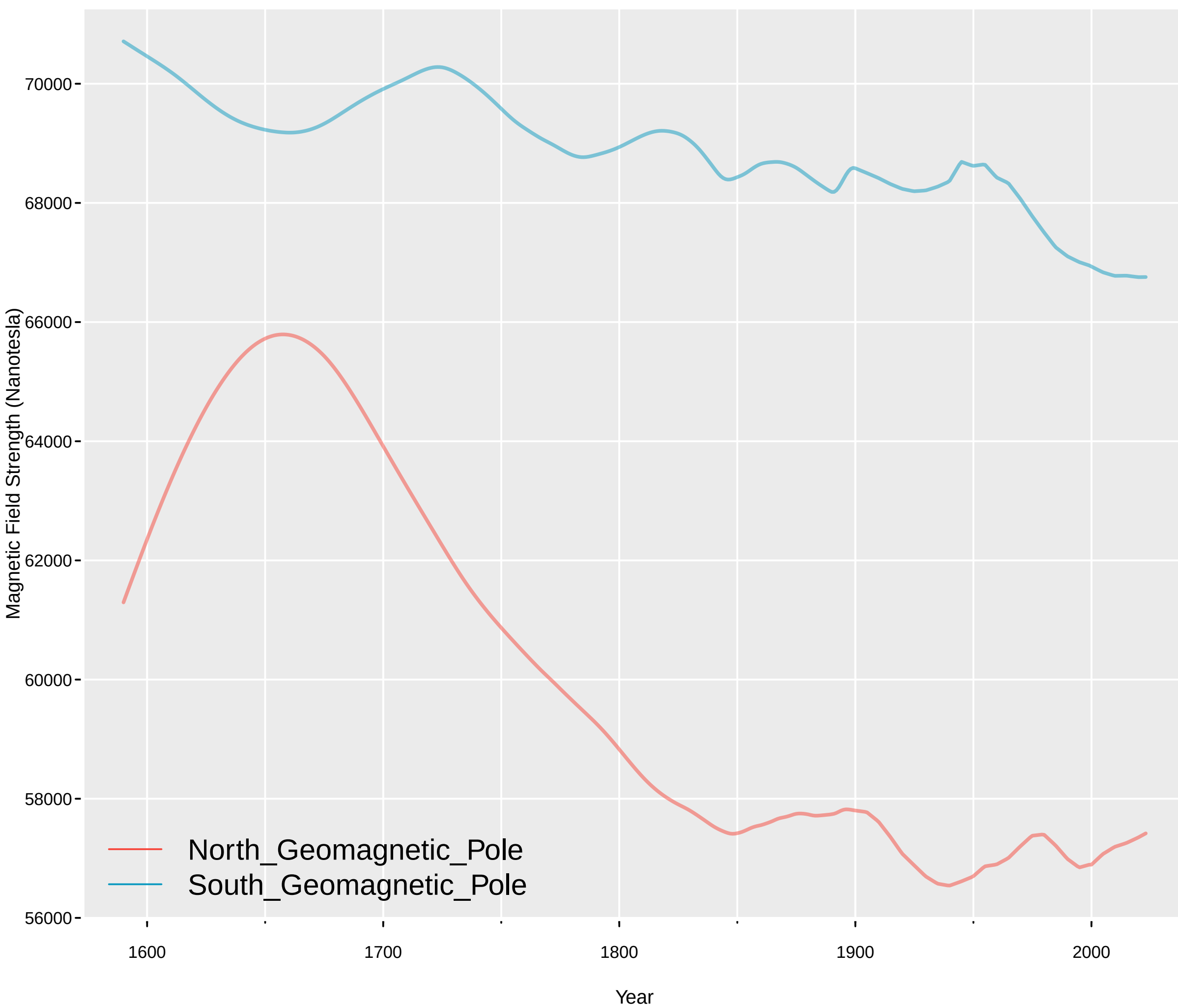


Figure 1: Time series of North and South Geomagnetic Pole

```
## initial  value 2.352973
## iter   2 value 1.464565
## iter   3 value 1.370134
## iter   4 value 1.271337
## iter   5 value 1.230414
## iter   6 value 1.211991
## iter   7 value 1.196550
```

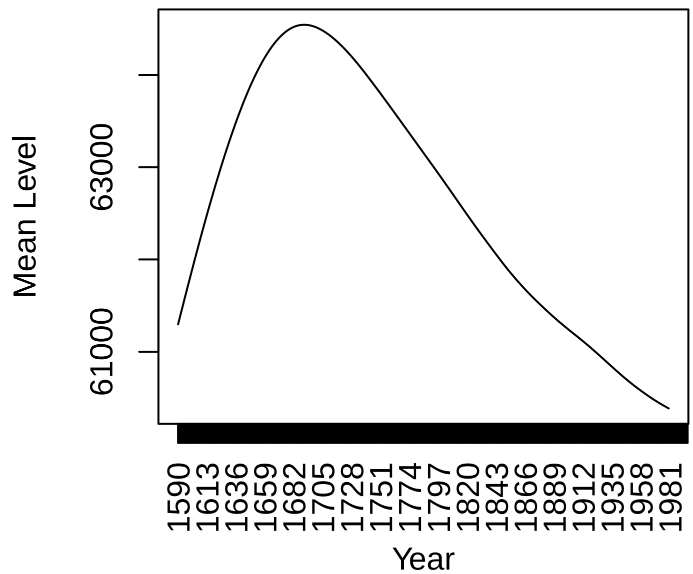


Figure 2: Mean Level Plot of North Geomagnetic Pole

```
## iter   8 value 1.187439
## iter   9 value 1.168967
## iter  10 value 1.163370
## iter  11 value 1.158261
## iter  12 value 1.152506
## iter  13 value 1.147359
## iter  14 value 1.144933
## iter  15 value 1.142278
## iter  16 value 1.141413
## iter  17 value 1.125010
## iter  18 value 1.124694
## iter  19 value 1.121367
## iter  20 value 1.119897
## iter  21 value 1.119132
## iter  22 value 1.118438
## iter  23 value 1.114132
## iter  24 value 1.113277
## iter  25 value 1.112317
## iter  26 value 1.111240
## iter  27 value 1.110670
## iter  28 value 1.110330
## iter  29 value 1.109373
## iter  30 value 1.108886
## iter  31 value 1.107764
## iter  32 value 1.107224
## iter  33 value 1.106758
```

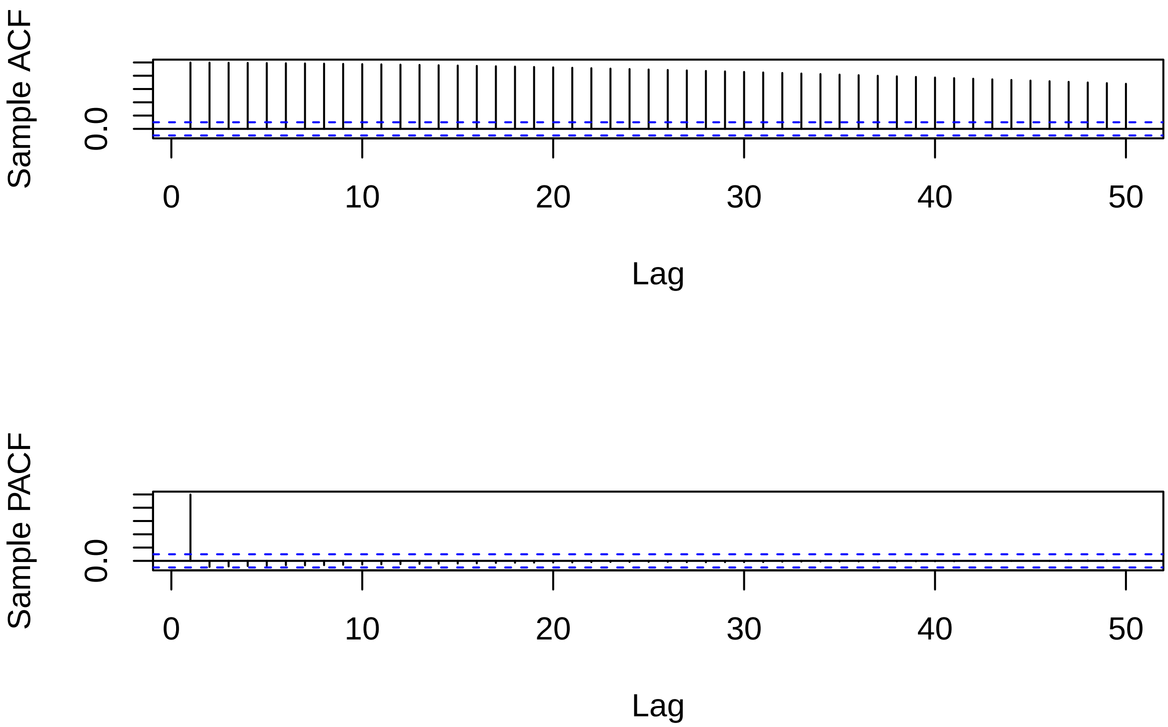


Figure 3: ACF and PACF of North Geomagnetic Pole

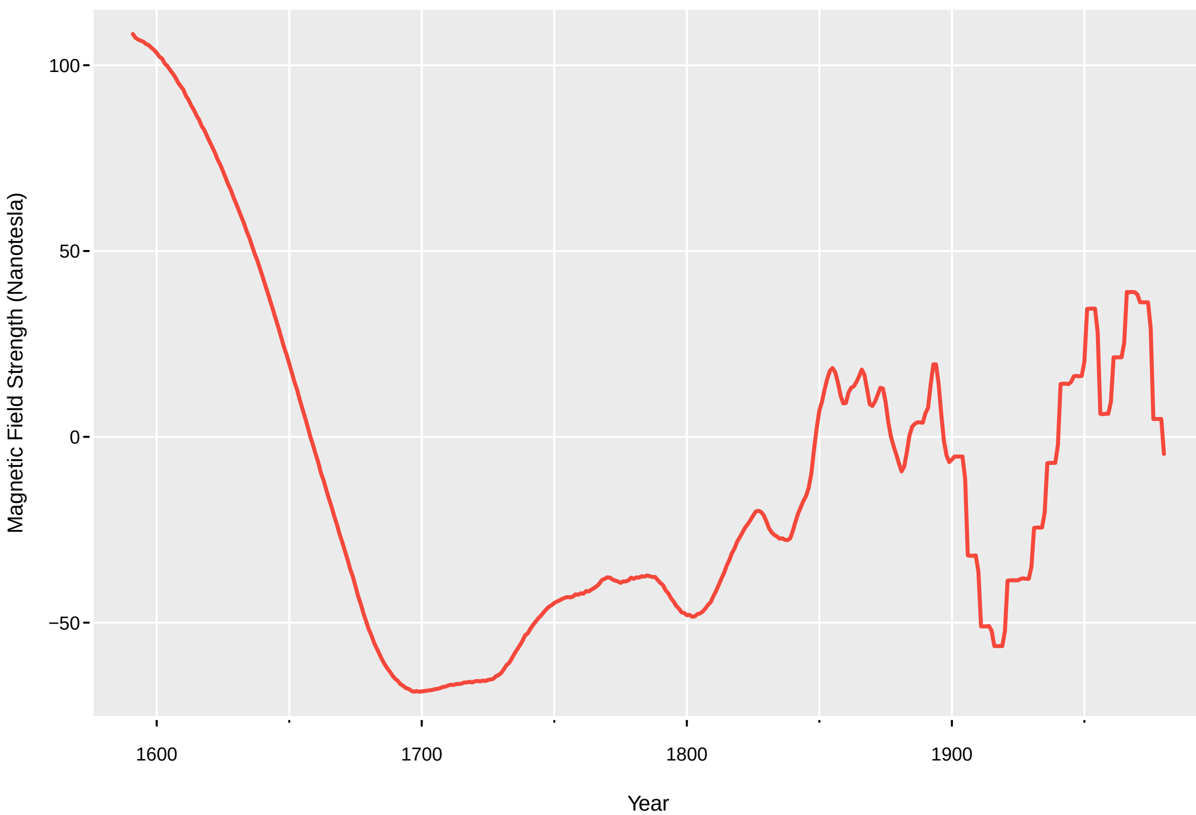


Figure 4: Time Series of North Geomagnetic Pole First Order Difference

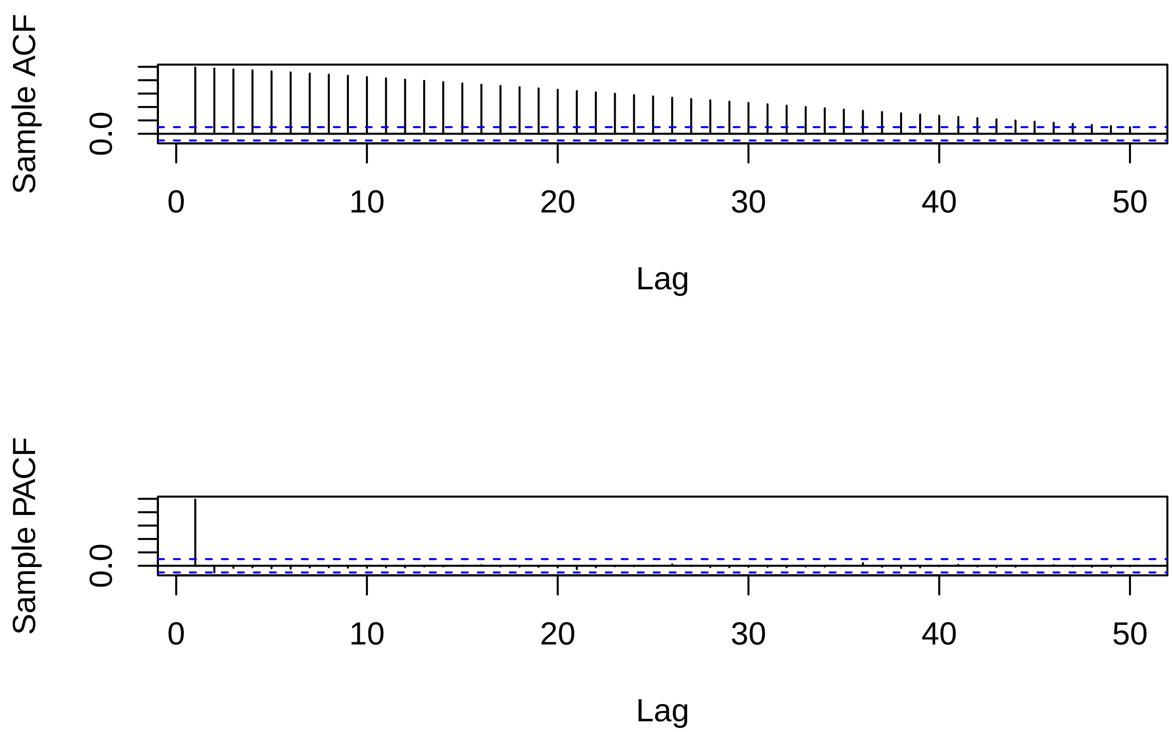


Figure 5: ACF and PACF of North Geomagnetic Pole Taking First Order Difference

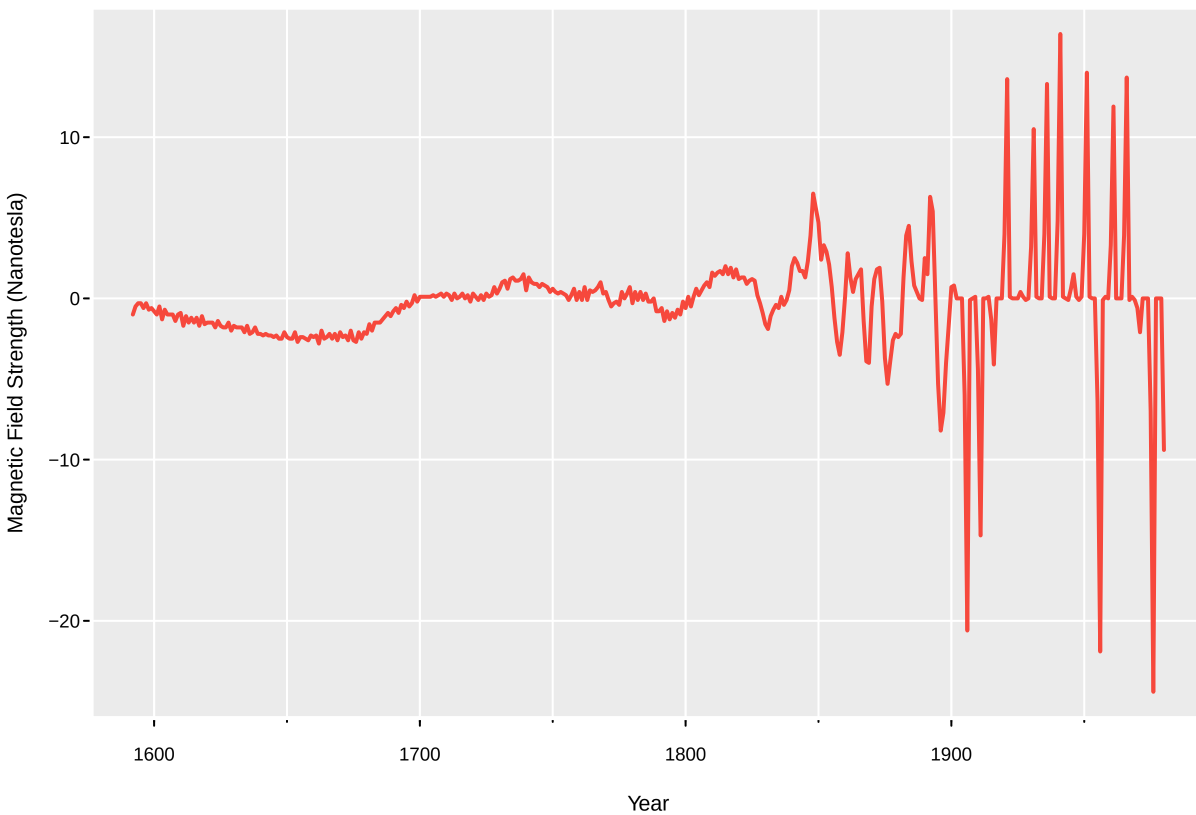


Figure 6: Time Series of North Geomagnetic Pole Second Order Difference

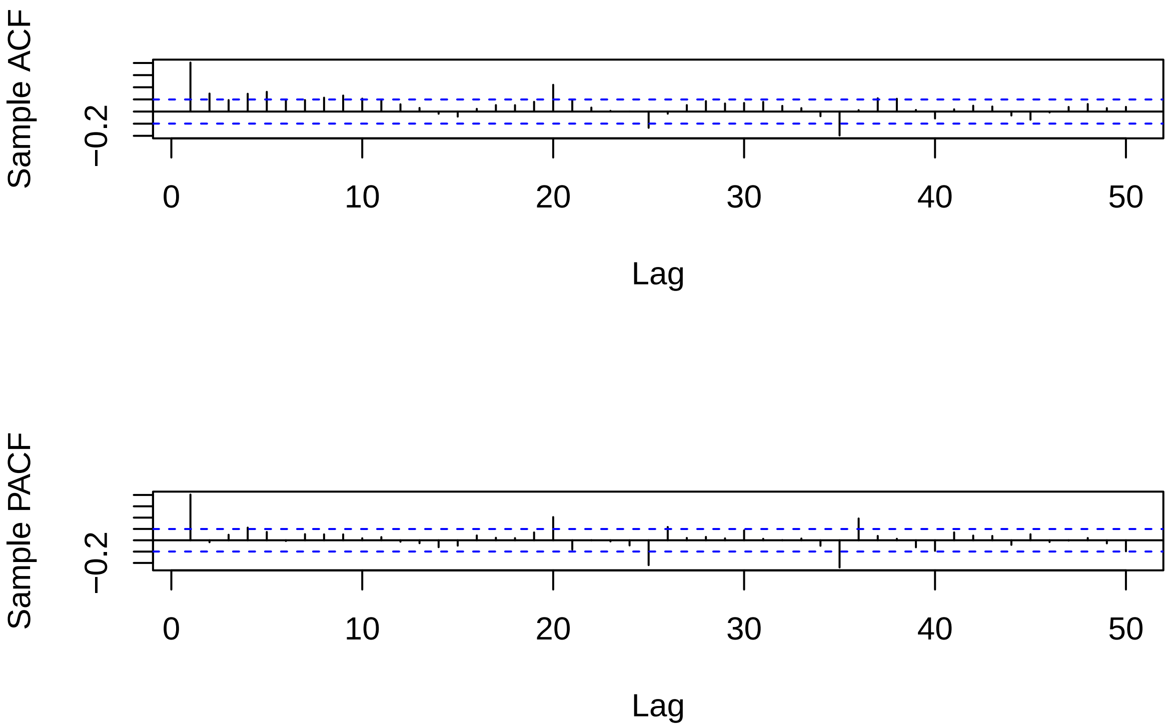


Figure 7: ACF and PACF of North Geomagnetic Pole Taking Second Order Difference

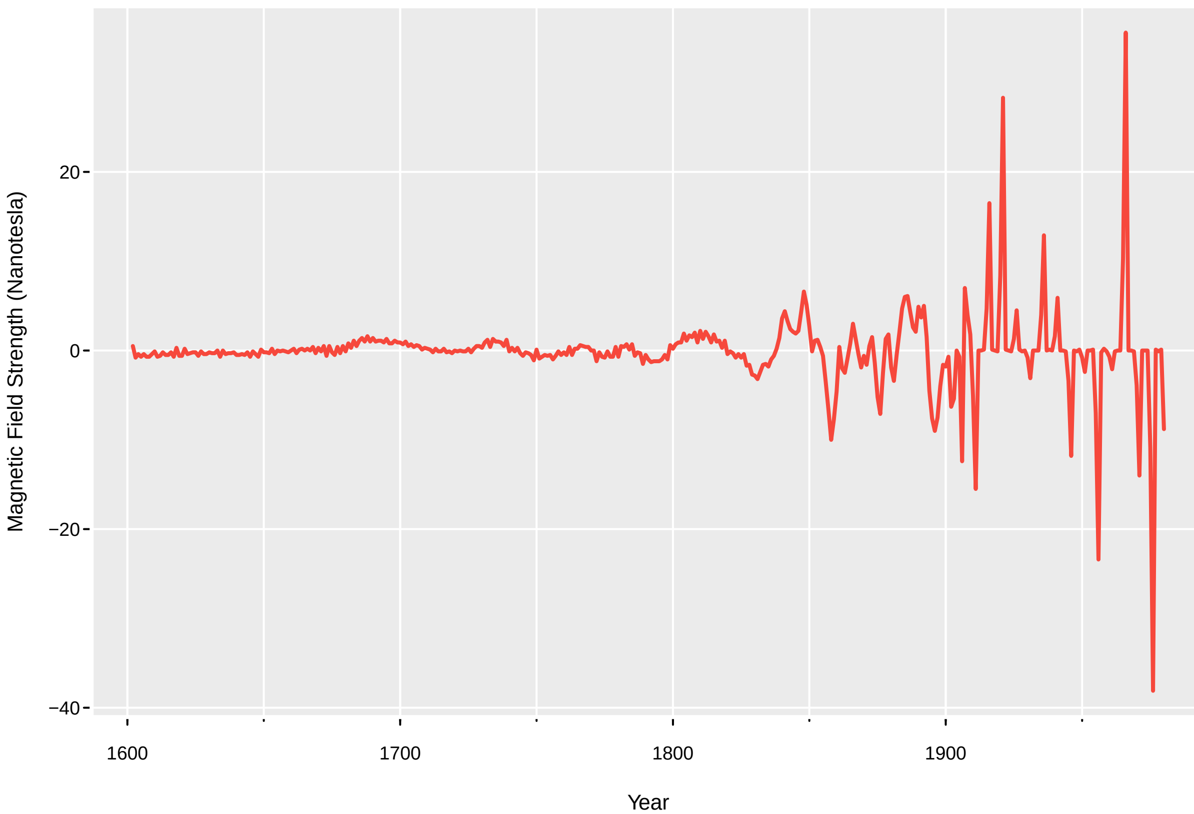


Figure 8: Time Series of North Geomagnetic Pole Second Order Difference and an Additional Seasonal Difference at Lag 10

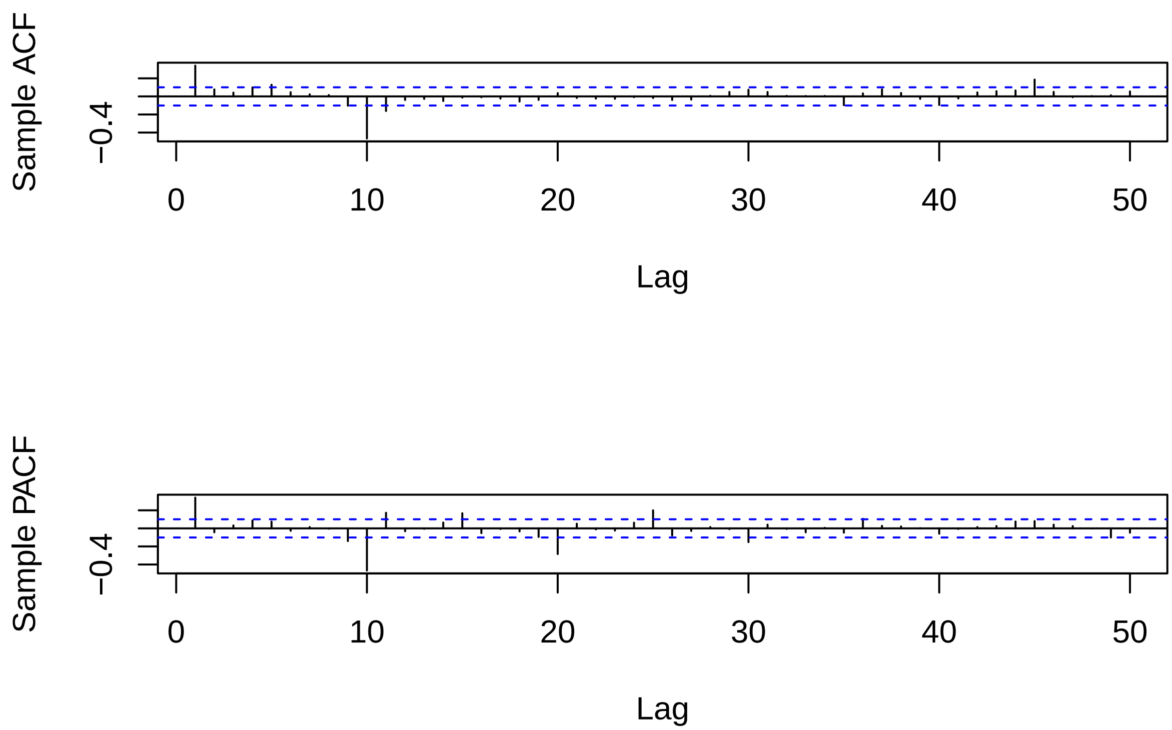


Figure 9: ACF and PACF of North Geomagnetic Pole Taking Second Order Difference and Seasonal Difference at Lag 10

Figure 10: Time Series of North Geomagnetic Pole First Order Seasonal Difference at Lag 5 With a First Order Difference

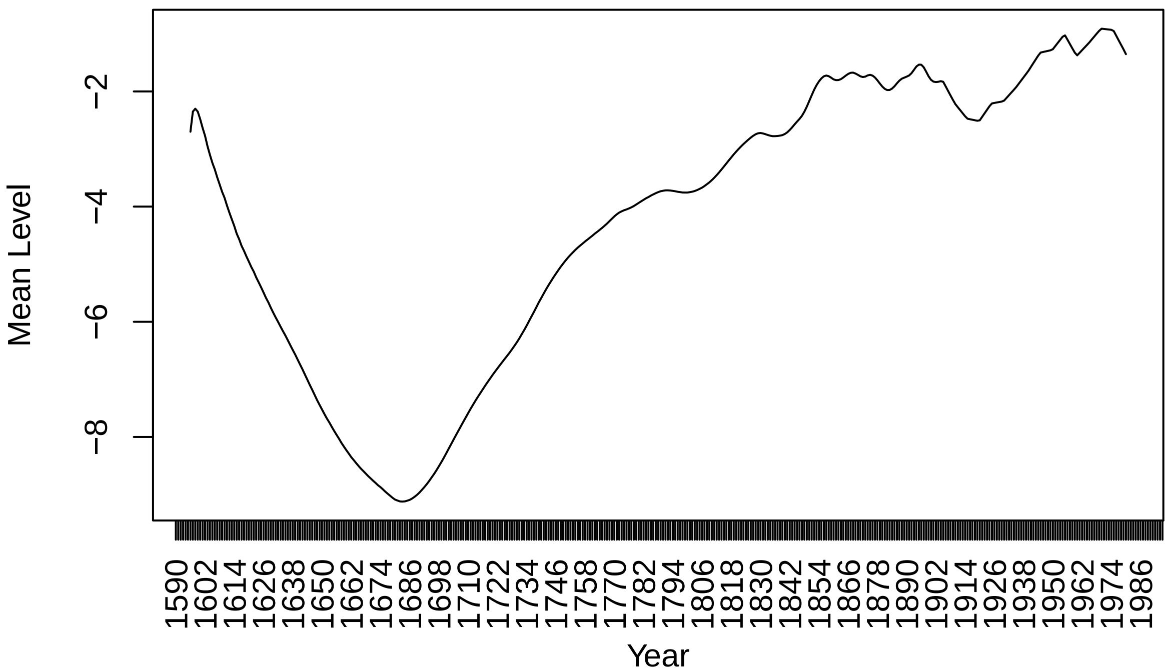


Figure 11: Mean Level Plot of North Geomagnetic Pole First Order Seasonal Difference at Lag 5 With a First Order Difference

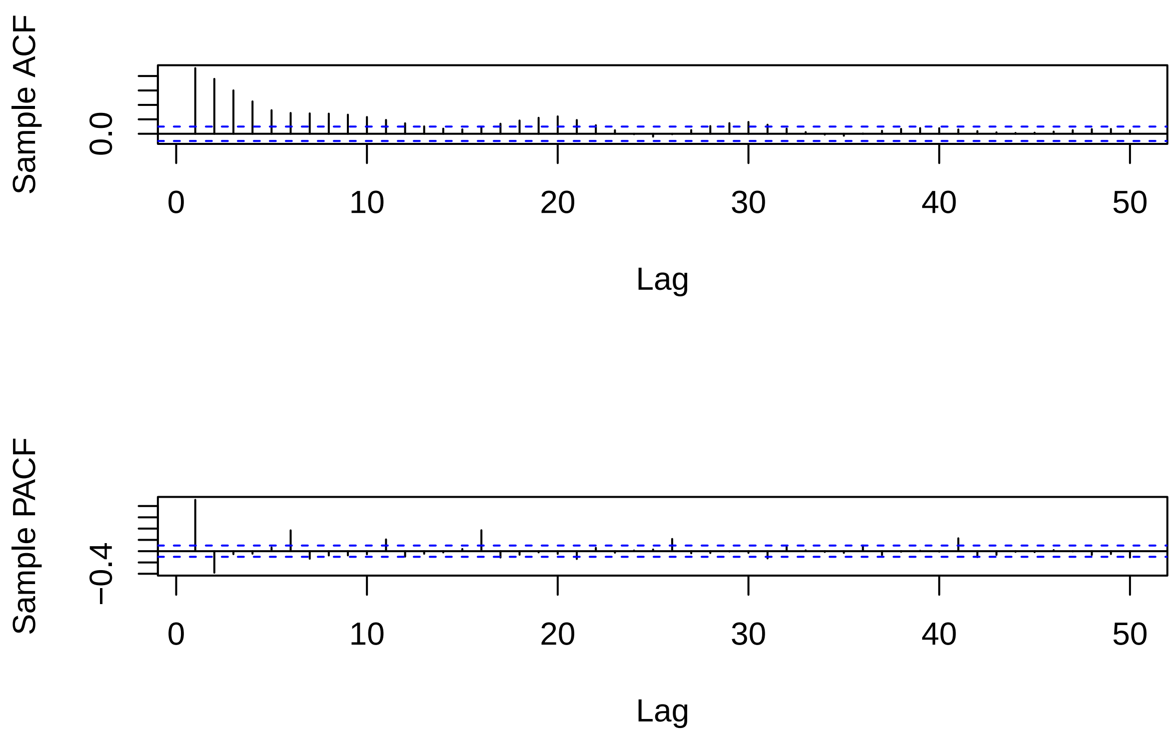


Figure 12: ACF and PACF of North Geomagnetic Pole First Order Seasonal Difference at Lag 5 With a First Order Difference

```
## iter  34 value 1.106724
## iter  35 value 1.106707
## iter  35 value 1.106707
## iter  36 value 1.106702
## iter  36 value 1.106702
## iter  37 value 1.106702
## iter  38 value 1.106700
## iter  38 value 1.106700
## iter  39 value 1.106688
## iter  40 value 1.106688
## iter  40 value 1.106688
## iter  40 value 1.106688
## final  value 1.106688
## converged
## initial  value 1.158778
## iter   2 value 1.152264
## iter   3 value 1.148452
## iter   4 value 1.148332
## iter   5 value 1.148069
## iter   6 value 1.147961
## iter   7 value 1.147782
## iter   8 value 1.147741
## iter   9 value 1.147721
## iter  10 value 1.147399
## iter  11 value 1.146585
## iter  12 value 1.146190
## iter  13 value 1.145655
## iter  14 value 1.143649
## iter  15 value 1.140935
## iter  16 value 1.140208
## iter  17 value 1.140106
## iter  18 value 1.139935
## iter  19 value 1.139855
## iter  20 value 1.139847
## iter  21 value 1.139839
## iter  22 value 1.139809
## iter  23 value 1.139799
## iter  24 value 1.139765
## iter  25 value 1.139699
## iter  26 value 1.139587
## iter  27 value 1.139499
## iter  28 value 1.139464
## iter  29 value 1.139460
## iter  30 value 1.139460
## iter  31 value 1.139460
## iter  32 value 1.139460
## iter  33 value 1.139460
## iter  34 value 1.139460
## iter  35 value 1.139460
## iter  35 value 1.139460
## final  value 1.139460
## converged

## $fit
```

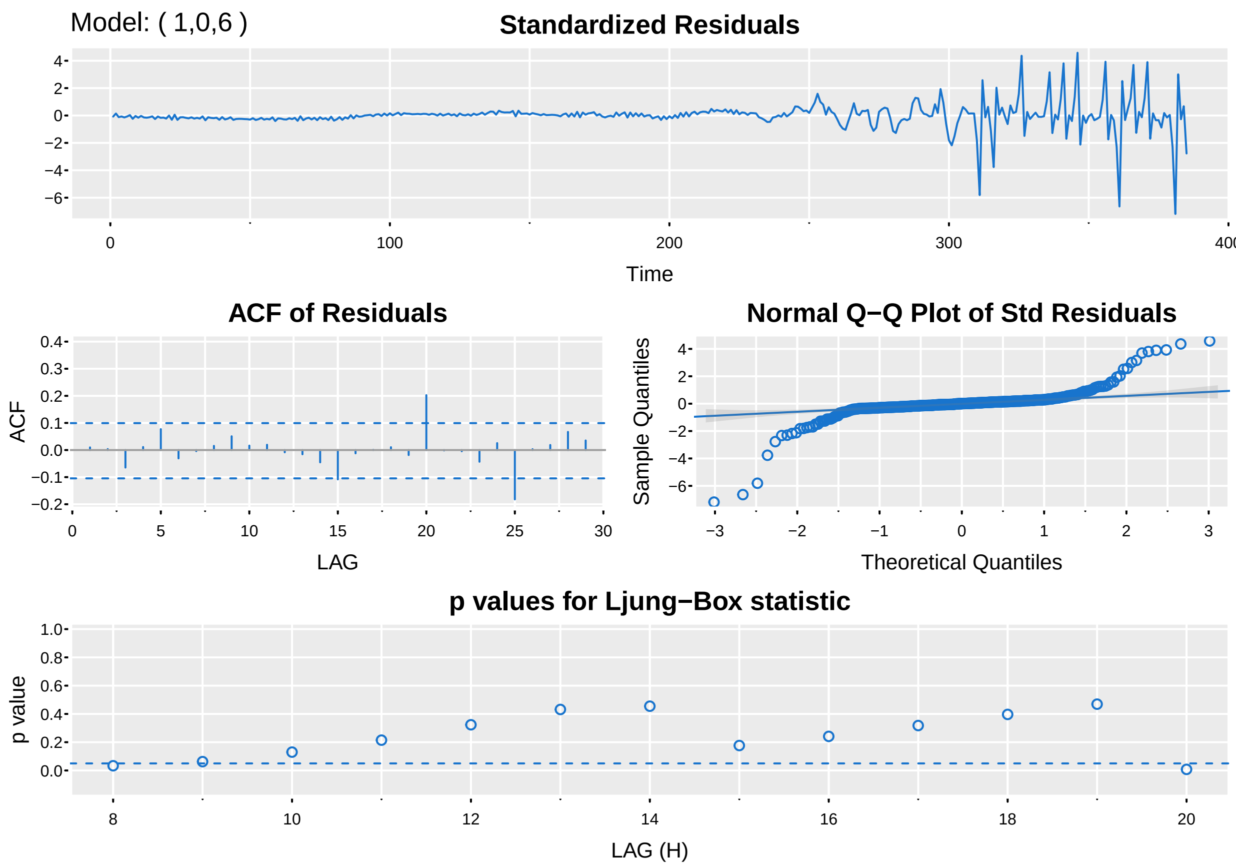


Figure 13: Standardized Residuals Plot, ACF of Residuals, QQ Plot, and Ljung-Box Test for ARIMA(1,0,6)

```
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4      ma5      ma6    xmean
##       0.9293  0.4612  0.2008  0.1970  0.1954  -0.8029  -0.2613  -1.8812
## s.e.  0.0483  0.0725  0.0912  0.0923  0.0923   0.0915   0.0578   2.1354
##
## sigma^2 estimated as 9.268:  log likelihood = -984.98,  aic = 1987.97
##
## $degrees_of_freedom
## [1] 377
##
## $ttable
##       Estimate     SE t.value p.value
## ar1     0.9293 0.0483 19.2425  0.0000
## ma1     0.4612 0.0725  6.3591  0.0000
## ma2     0.2008 0.0912  2.2026  0.0282
## ma3     0.1970 0.0923  2.1333  0.0335
## ma4     0.1954 0.0923  2.1167  0.0349
## ma5    -0.8029 0.0915 -8.7720  0.0000
## ma6    -0.2613 0.0578 -4.5237  0.0000
## xmean  -1.8812 2.1354 -0.8810  0.3789
##
## $AIC
## [1] 5.16355
##
## $AICc
## [1] 5.164545
##
## $BIC
## [1] 5.255964
```

```
## initial  value 2.354271
## iter   2 value 1.770303
## iter   3 value 1.644378
## iter   4 value 1.461712
## iter   5 value 1.253526
## iter   6 value 1.220849
## iter   7 value 1.219529
## iter   8 value 1.210336
## iter   9 value 1.200909
## iter  10 value 1.170406
## iter  11 value 1.165847
## iter  12 value 1.156434
## iter  13 value 1.155880
## iter  14 value 1.150054
## iter  15 value 1.145717
## iter  16 value 1.140832
## iter  17 value 1.139656
## iter  18 value 1.139314
```

```
## iter  19 value 1.126701
## iter  20 value 1.125969
## iter  21 value 1.124373
## iter  22 value 1.123907
## iter  23 value 1.122302
## iter  24 value 1.122103
## iter  25 value 1.121675
## iter  26 value 1.121600
## iter  27 value 1.121500
## iter  28 value 1.121325
## iter  29 value 1.121243
## iter  30 value 1.121193
## iter  31 value 1.121077
## iter  32 value 1.120659
## iter  33 value 1.120088
## iter  34 value 1.118999
## iter  35 value 1.118866
## iter  36 value 1.118859
## iter  37 value 1.117440
## iter  38 value 1.116579
## iter  39 value 1.116217
## iter  40 value 1.115794
## iter  41 value 1.113605
## iter  42 value 1.113446
## iter  43 value 1.112997
## iter  44 value 1.110283
## iter  45 value 1.109574
## iter  46 value 1.109410
## iter  47 value 1.108662
## iter  48 value 1.107414
## iter  49 value 1.106337
## iter  50 value 1.104786
## iter  51 value 1.102641
## iter  52 value 1.100770
## iter  53 value 1.100227
## iter  54 value 1.099623
## iter  55 value 1.097840
## iter  56 value 1.097704
## iter  57 value 1.097529
## iter  58 value 1.095619
## iter  59 value 1.095412
## iter  59 value 1.095412
## iter  60 value 1.094905
## iter  61 value 1.094849
## iter  62 value 1.094724
## iter  62 value 1.094724
## iter  63 value 1.092899
## iter  64 value 1.092521
## iter  65 value 1.092459
## iter  66 value 1.092392
## iter  67 value 1.092391
## iter  68 value 1.092336
## iter  68 value 1.092336
## iter  69 value 1.092083
```

```
## iter  70 value 1.092082
## iter  71 value 1.092044
## iter  72 value 1.092038
## iter  72 value 1.092038
## iter  73 value 1.092009
## iter  74 value 1.092008
## iter  75 value 1.091983
## iter  76 value 1.091983
## iter  76 value 1.091983
## iter  77 value 1.091968
## iter  78 value 1.091966
## iter  79 value 1.091953
## iter  79 value 1.091953
## iter  80 value 1.091891
## iter  81 value 1.091891
## iter  82 value 1.091867
## iter  83 value 1.091864
## iter  83 value 1.091864
## iter  84 value 1.091854
## iter  85 value 1.091854
## iter  86 value 1.091835
## iter  86 value 1.091835
## iter  87 value 1.091825
## iter  88 value 1.091824
## iter  89 value 1.091806
## iter  89 value 1.091806
## iter  90 value 1.091794
## iter  91 value 1.091794
## iter  92 value 1.091776
## iter  92 value 1.091776
## iter  93 value 1.091763
## iter  94 value 1.091763
## iter  95 value 1.091745
## iter  95 value 1.091745
## iter  96 value 1.091732
## iter  97 value 1.091732
## iter  98 value 1.091715
## iter  98 value 1.091715
## iter  99 value 1.091701
## iter  99 value 1.091701
## iter 100 value 1.091700
## final  value 1.091700
## stopped after 100 iterations
## initial  value 2.351694
## iter   2 value 1.523582
## iter   3 value 1.394673
## iter   4 value 1.255550
## iter   5 value 1.248226
## iter   6 value 1.226937
## iter   7 value 1.221091
## iter   8 value 1.204756
## iter   9 value 1.189816
## iter  10 value 1.178820
## iter  11 value 1.176248
```

```
## iter  12 value 1.172374
## iter  13 value 1.165532
## iter  14 value 1.162295
## iter  15 value 1.160494
## iter  16 value 1.159863
## iter  17 value 1.159349
## iter  18 value 1.158579
## iter  19 value 1.152045
## iter  20 value 1.150586
## iter  21 value 1.148980
## iter  22 value 1.148896
## iter  23 value 1.148776
## iter  24 value 1.148482
## iter  25 value 1.148180
## iter  26 value 1.148066
## iter  27 value 1.147797
## iter  28 value 1.147314
## iter  29 value 1.146799
## iter  30 value 1.146344
## iter  31 value 1.146098
## iter  32 value 1.146057
## iter  33 value 1.145987
## iter  34 value 1.145756
## iter  35 value 1.144989
## iter  36 value 1.144150
## iter  37 value 1.143811
## iter  38 value 1.143631
## iter  39 value 1.143319
## iter  40 value 1.143028
## iter  41 value 1.142581
## iter  42 value 1.142521
## iter  43 value 1.142265
## iter  44 value 1.142076
## iter  45 value 1.141749
## iter  46 value 1.140793
## iter  47 value 1.140666
## iter  48 value 1.140613
## iter  49 value 1.140568
## iter  50 value 1.140439
## iter  51 value 1.140228
## iter  52 value 1.139893
## iter  53 value 1.139532
## iter  54 value 1.139321
## iter  55 value 1.139241
## iter  56 value 1.139227
## iter  57 value 1.139219
## iter  58 value 1.139219
## iter  59 value 1.139218
## iter  60 value 1.139218
## iter  61 value 1.139218
## iter  62 value 1.139218
## iter  63 value 1.139217
## iter  64 value 1.139217
## iter  65 value 1.139215
```

```
## iter  66 value 1.139213
## iter  67 value 1.139213
## iter  68 value 1.139212
## iter  69 value 1.139212
## iter  70 value 1.139212
## iter  71 value 1.139212
## iter  72 value 1.139212
## iter  72 value 1.139212
## iter  72 value 1.139212
## final  value 1.139212
## converged
```

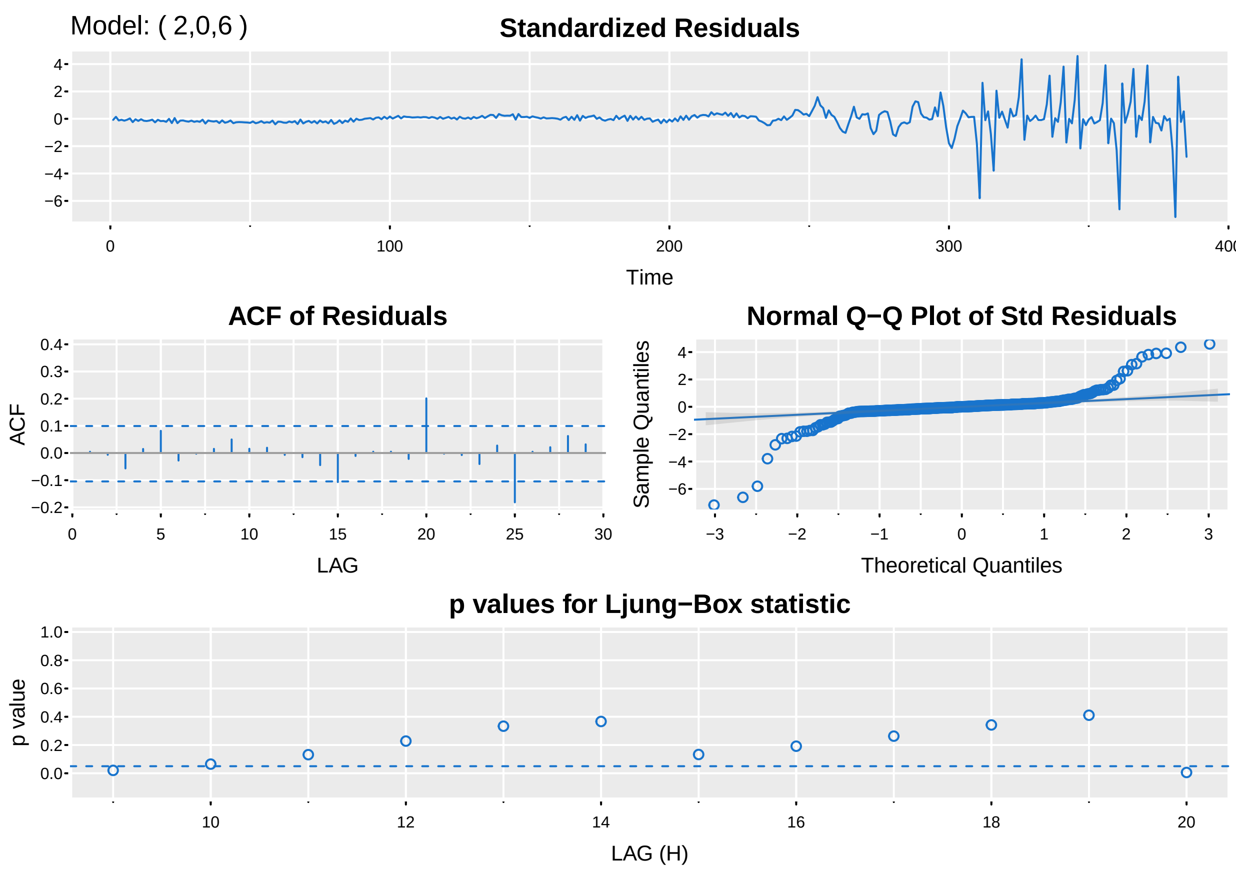


Figure 14: Standardized Residuals Plot, ACF of Residuals, QQ Plot, and Ljung-Box Test for ARIMA(2,0,6)

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2     ma1     ma2     ma3     ma4      ma5      ma6
##       1.0139  -0.0738  0.3826  0.1708  0.1692  0.1678  -0.8310  -0.2116
## s.e.  0.1947   0.1664  0.1908  0.1046  0.1023  0.1022   0.1023   0.1281
```

```
##        xmean
##      -1.8726
## s.e.   2.1543
##
## sigma^2 estimated as 9.262:  log likelihood = -984.89,  aic = 1989.78
##
## $degrees_of_freedom
## [1] 376
##
## $ttable
##       Estimate     SE t.value p.value
## ar1     1.0139 0.1947  5.2080  0.0000
## ar2    -0.0738 0.1664 -0.4438  0.6574
## ma1     0.3826 0.1908  2.0058  0.0456
## ma2     0.1708 0.1046  1.6333  0.1033
## ma3     0.1692 0.1023  1.6542  0.0989
## ma4     0.1678 0.1022  1.6419  0.1014
## ma5    -0.8310 0.1023 -8.1252  0.0000
## ma6    -0.2116 0.1281 -1.6511  0.0996
## xmean  -1.8726 2.1543 -0.8692  0.3853
##
## $AIC
## [1] 5.168249
##
## $AICc
## [1] 5.169496
##
## $BIC
## [1] 5.270931
```

```
## initial  value 2.358147
## iter   2 value 2.079059
## iter   3 value 1.602288
## iter   4 value 1.491845
## iter   5 value 1.402516
## iter   6 value 1.330362
## iter   7 value 1.309536
## iter   8 value 1.286400
## iter   9 value 1.190454
## iter  10 value 1.181462
## iter  11 value 1.160133
## iter  12 value 1.154323
## iter  13 value 1.149085
## iter  14 value 1.139939
## iter  15 value 1.137242
## iter  16 value 1.124899
## iter  17 value 1.124881
## iter  18 value 1.121727
## iter  19 value 1.119648
## iter  20 value 1.119232
## iter  21 value 1.118608
## iter  22 value 1.118027
## iter  23 value 1.117776
## iter  24 value 1.117653
```

```
## iter  25 value 1.117612
## iter  26 value 1.117606
## iter  27 value 1.117604
## iter  28 value 1.117600
## iter  29 value 1.117576
## iter  30 value 1.117520
## iter  31 value 1.117369
## iter  32 value 1.117105
## iter  33 value 1.116627
## iter  34 value 1.116078
## iter  35 value 1.115566
## iter  36 value 1.115275
## iter  37 value 1.115163
## iter  38 value 1.115154
## iter  39 value 1.115138
## iter  40 value 1.115132
## iter  41 value 1.115088
## iter  42 value 1.115072
## iter  43 value 1.115038
## iter  44 value 1.115017
## iter  45 value 1.115016
## iter  46 value 1.115012
## iter  47 value 1.115010
## iter  48 value 1.114996
## iter  49 value 1.114982
## iter  50 value 1.114964
## iter  51 value 1.114957
## iter  52 value 1.114956
## iter  52 value 1.114956
## iter  52 value 1.114956
## final  value 1.114956
## converged
## initial  value 1.140123
## iter   2 value 1.139063
## iter   3 value 1.138461
## iter   4 value 1.138195
## iter   5 value 1.137405
## iter   6 value 1.137335
## iter   7 value 1.137246
## iter   8 value 1.136907
## iter   9 value 1.136643
## iter  10 value 1.135744
## iter  11 value 1.135043
## iter  12 value 1.134776
## iter  13 value 1.134736
## iter  14 value 1.134712
## iter  15 value 1.134662
## iter  16 value 1.134574
## iter  17 value 1.134423
## iter  18 value 1.134253
## iter  19 value 1.134137
## iter  20 value 1.134097
## iter  21 value 1.134076
## iter  22 value 1.134057
```

```
## iter  23 value 1.134042
## iter  24 value 1.134038
## iter  24 value 1.134038
## final  value 1.134038
## converged
```

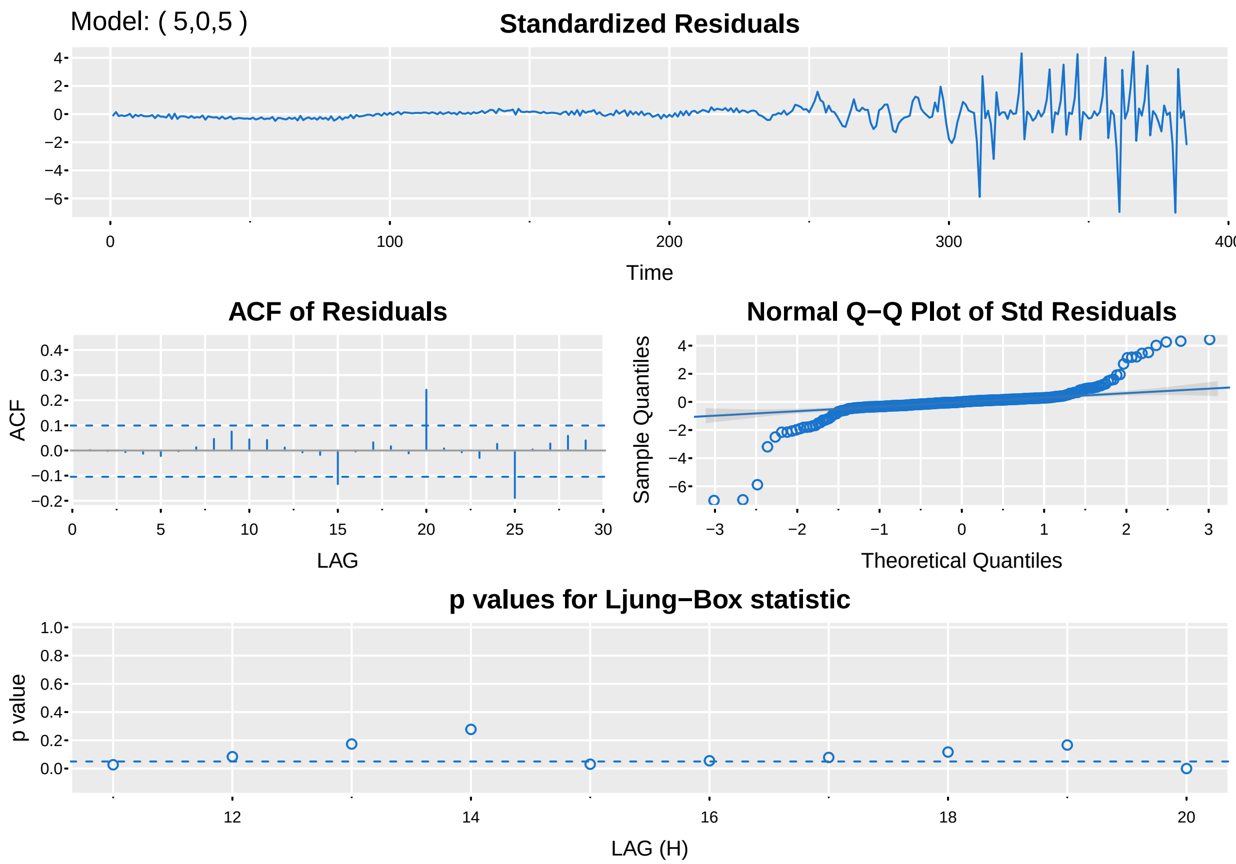


Figure 15: Standardized Residuals Plot, ACF of Residuals, QQ Plot, and Ljung-Box Test for ARIMA(5,0,5)

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     ar2      ar3     ar4     ar5     ma1     ma2     ma3     ma4
##       -0.0348  0.1392  -0.0160  0.0675  0.1748  1.4471  1.4459  1.4479  1.4441
## s.e.   0.4086  0.1748   0.0563  0.0556  0.0609  0.4120  0.4140  0.4144  0.4109
##          ma5    xmean
##       0.4417  -1.6348
## s.e.  0.4055   1.6514
##
## sigma^2 estimated as 9.196:  log likelihood = -982.9,  aic = 1989.79
##
```

```
## $degrees_of_freedom
## [1] 374
##
## $ttable
##       Estimate     SE t.value p.value
## ar1    -0.0348 0.4086 -0.0853  0.9321
## ar2     0.1392 0.1748  0.7960  0.4266
## ar3    -0.0160 0.0563 -0.2846  0.7761
## ar4     0.0675 0.0556  1.2148  0.2252
## ar5     0.1748 0.0609  2.8688  0.0044
## ma1     1.4471 0.4120  3.5125  0.0005
## ma2     1.4459 0.4140  3.4921  0.0005
## ma3     1.4479 0.4144  3.4937  0.0005
## ma4     1.4441 0.4109  3.5142  0.0005
## ma5     0.4417 0.4055  1.0893  0.2767
## xmean  -1.6348 1.6514 -0.9899  0.3228
##
## $AIC
## [1] 5.168291
##
## $AICc
## [1] 5.17013
##
## $BIC
## [1] 5.291509
```

```
## initial  value 2.371341
## iter   2 value 1.621285
## iter   3 value 1.496923
## iter   4 value 1.354090
## iter   5 value 1.271548
## iter   6 value 1.209786
## iter   7 value 1.203302
## iter   8 value 1.195402
## iter   9 value 1.178559
## iter  10 value 1.176489
## iter  11 value 1.173813
## iter  12 value 1.166562
## iter  13 value 1.160901
## iter  14 value 1.157469
## iter  15 value 1.155296
## iter  16 value 1.154148
## iter  17 value 1.151894
## iter  18 value 1.145744
## iter  19 value 1.143171
## iter  20 value 1.142766
## iter  21 value 1.142019
## iter  22 value 1.140584
## iter  23 value 1.136864
## iter  24 value 1.135906
## iter  25 value 1.135737
## iter  26 value 1.135648
## iter  27 value 1.135589
## iter  28 value 1.135237
```

```
## iter  29 value 1.134000
## iter  30 value 1.132382
## iter  31 value 1.131926
## iter  32 value 1.131098
## iter  33 value 1.130833
## iter  34 value 1.130471
## iter  35 value 1.130234
## iter  36 value 1.129432
## iter  37 value 1.129191
## iter  38 value 1.128798
## iter  39 value 1.127744
## iter  40 value 1.127501
## iter  41 value 1.126631
## iter  42 value 1.126226
## iter  43 value 1.125955
## iter  44 value 1.125856
## iter  45 value 1.125845
## iter  46 value 1.125826
## iter  47 value 1.125800
## iter  48 value 1.125736
## iter  49 value 1.125685
## iter  50 value 1.125667
## iter  51 value 1.125663
## iter  52 value 1.125663
## iter  52 value 1.125663
## iter  52 value 1.125663
## final  value 1.125663
## converged
## initial  value 1.127560
## iter   2 value 1.126704
## iter   3 value 1.126626
## iter   4 value 1.126617
## iter   5 value 1.126465
## iter   6 value 1.126455
## iter   7 value 1.126406
## iter   8 value 1.126372
## iter   9 value 1.126368
## iter  10 value 1.126361
## iter  11 value 1.126343
## iter  12 value 1.126317
## iter  13 value 1.126293
## iter  14 value 1.126284
## iter  15 value 1.126283
## iter  16 value 1.126283
## iter  17 value 1.126283
## iter  18 value 1.126283
## iter  19 value 1.126283
## iter  20 value 1.126282
## iter  21 value 1.126282
## iter  22 value 1.126282
## iter  22 value 1.126282
## iter  22 value 1.126282
## final  value 1.126282
## converged
```

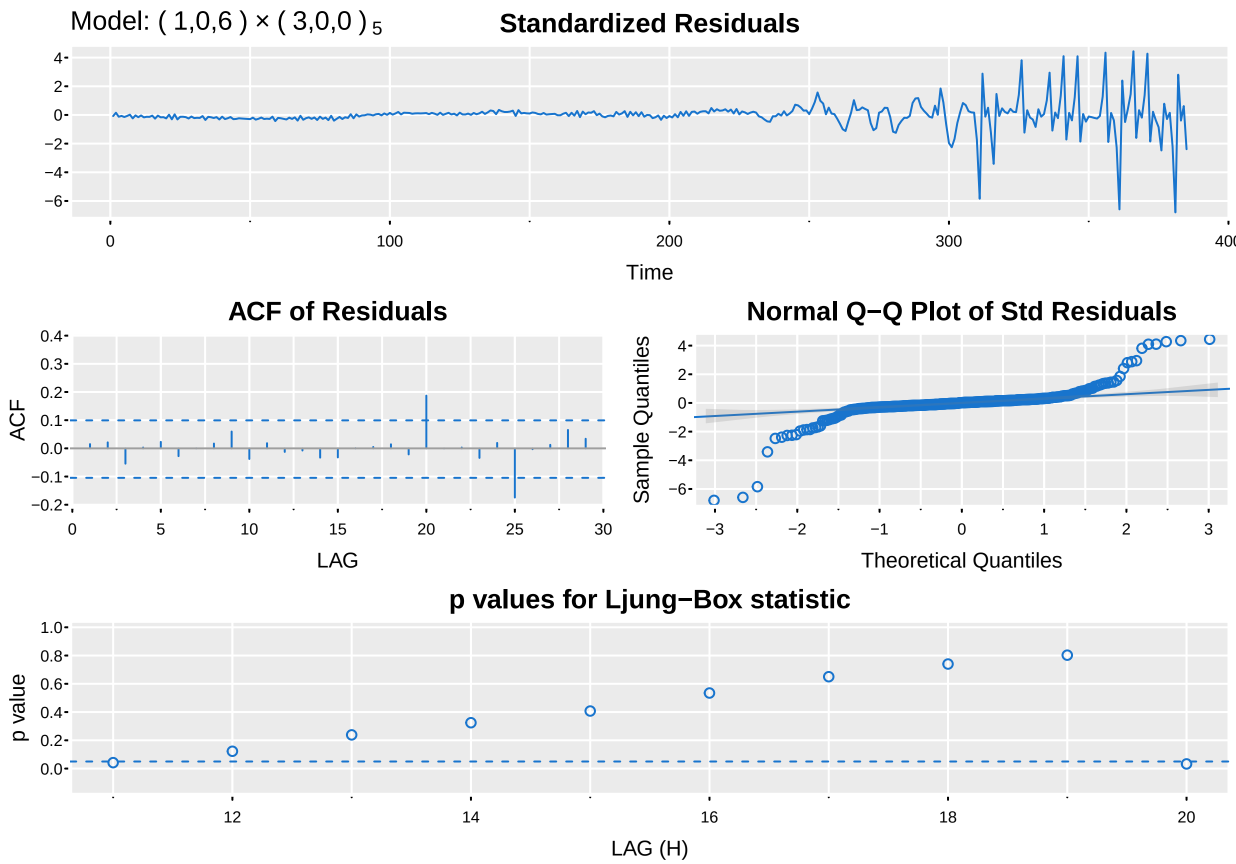


Figure 16: Standardized Residuals Plot, ACF of Residuals, QQ Plot, and Ljung-Box Test for SARIMA(1,0,6)(3,0,0)_5

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4      ma5      ma6    sar1    sar2
##       0.9443  0.4430  0.1600  0.1544  0.1530  -0.8448  -0.2848  0.1269  0.0332
## s.e.  0.0447  0.0692  0.0851  0.0863  0.0863   0.0854   0.0568  0.0589  0.0569
##          sar3    xmean
##       -0.1434  -1.8933
## s.e.   0.0581   2.1436
##
## sigma^2 estimated as 9.022:  log likelihood = -979.91,  aic = 1983.82
##
## $degrees_of_freedom
## [1] 374
##
## $ttable
##       Estimate     SE t.value p.value
```

```
## ar1     0.9443 0.0447 21.1369  0.0000
## ma1     0.4430 0.0692  6.4058  0.0000
## ma2     0.1600 0.0851  1.8803  0.0609
## ma3     0.1544 0.0863  1.7904  0.0742
## ma4     0.1530 0.0863  1.7719  0.0772
## ma5    -0.8448 0.0854 -9.8930  0.0000
## ma6    -0.2848 0.0568 -5.0131  0.0000
## sar1    0.1269 0.0589  2.1555  0.0318
## sar2    0.0332 0.0569  0.5829  0.5603
## sar3   -0.1434 0.0581 -2.4667  0.0141
## xmean  -1.8933 2.1436 -0.8832  0.3777
##
## $AIC
## [1] 5.152779
##
## $AICc
## [1] 5.154617
##
## $BIC
## [1] 5.275997
```

7.3 South Geomagnetic Pole Tests

7.3.1 ADF_KPSS_S1

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -1.3465
##   P VALUE:
##     0.1872
##
## Description:
##  Mon Apr 10 00:31:17 2023 by user: jason
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -0.7349
##   P VALUE:
##     0.7846
##
## Description:
```

```
##  Mon Apr 10 00:31:17 2023 by user: jason
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -2.9055
##   P VALUE:
##     0.1948
##
## Description:
##  Mon Apr 10 00:31:17 2023 by user: jason
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$South_Geomagnetic_Pole[1:nTrain]
## Dickey-Fuller = -2.9055, Lag order = 17, p-value = 0.1948
## alternative hypothesis: stationary
```

```
## Warning in kpss.test(data$South_Geomagnetic_Pole[1:nTrain], null = "Level"):
## p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  data$South_Geomagnetic_Pole[1:nTrain]
## KPSS Level = 5.0584, Truncation lag parameter = 5, p-value = 0.01
```

```
## Warning in kpss.test(data$South_Geomagnetic_Pole[1:nTrain], null = "Trend"):
## p-value smaller than printed p-value
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  data$South_Geomagnetic_Pole[1:nTrain]
## KPSS Trend = 0.2705, Truncation lag parameter = 5, p-value = 0.01
```

7.3.2 ADF_KPSS_S2

```
## Warning in adfTest(diffS1, type = "nc", lags = p_max_lag_S): p-value smaller
## than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
```

```
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -3.0274
##   P VALUE:
##     0.01
##
## Description:
##  Mon Apr 10 00:31:17 2023 by user: jason

##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -3.3121
##   P VALUE:
##     0.01653
##
## Description:
##  Mon Apr 10 00:31:17 2023 by user: jason

##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 17
##   STATISTIC:
##     Dickey-Fuller: -3.2558
##   P VALUE:
##     0.07864
##
## Description:
##  Mon Apr 10 00:31:17 2023 by user: jason

##
##  Augmented Dickey-Fuller Test
##
## data:  diffS1
## Dickey-Fuller = -3.2558, Lag order = 17, p-value = 0.07864
## alternative hypothesis: stationary

## Warning in kpss.test(diffS1, null = "Level"): p-value greater than printed
## p-value

##
##  KPSS Test for Level Stationarity
```

```
##
## data:  diffS1
## KPSS Level = 0.17975, Truncation lag parameter = 5, p-value = 0.1
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  diffS1
## KPSS Trend = 0.17941, Truncation lag parameter = 5, p-value = 0.02372
```

7.3.3 S_EACF

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

7.3.4 S_AIC_BIC

```
diffS1.aic=matrix(0,10,10)
diffS1.bic = matrix(0,10,10)
for (i in 0:9) for (j in 0:9){
  diffS1.fit = arima(diffS1, order = c(i,0,j), method = "ML", include.mean = TRUE)
  diffS1.aic[i+1,j+1] = diffS1.fit$aic
  diffS1.bic[i+1,j+1] = BIC(diffS1.fit)
}
diffS1.aic_vec = sort(unmatrix(diffS1.aic, byrow = FALSE))[1:20]
diffS1.bic_vec = sort(unmatrix(diffS1.bic, byrow = FALSE))[1:20]
diffS1.aic_vec
```

```
##    r8:c9   r7:c10    r9:c9   r8:c10   r9:c10    r7:c9  r10:c10    r7:c8
## 2264.202 2265.598 2266.040 2267.914 2268.240 2268.360 2270.057 2270.492
##    r8:c7    r9:c7   r10:c7    r9:c8    r8:c8    r7:c7   r10:c5   r10:c8
## 2270.602 2271.021 2277.741 2278.324 2280.196 2282.222 2283.232 2284.095
##   r10:c6   r10:c9    r9:c5   r10:c4
## 2286.053 2286.176 2287.719 2287.953
```

```
diffS1.bic_vec
```

```
##    r8:c1    r7:c8    r8:c7    r8:c9    r7:c9   r7:c10    r9:c1    r9:c7
## 2330.159 2331.985 2332.095 2333.627 2333.818 2335.022 2336.072 2336.480
```

```
##    r7:c4    r8:c2    r7:c3    r7:c2    r8:c4    r9:c9    r7:c7    r9:c2
## 2336.815 2337.018 2337.640 2338.582 2338.979 2339.431 2339.748 2339.773
##   r10:c1   r8:c10    r9:c4    r8:c3
## 2340.983 2341.305 2341.550 2342.711
```

7.3.5 S_Shapiro

```
##
##  Shapiro-Wilk normality test
##
## data:  diffS1.fit_22$residuals
## W = 0.60984, p-value < 2.2e-16
```

7.4 South Geomagnetic Pole Figures

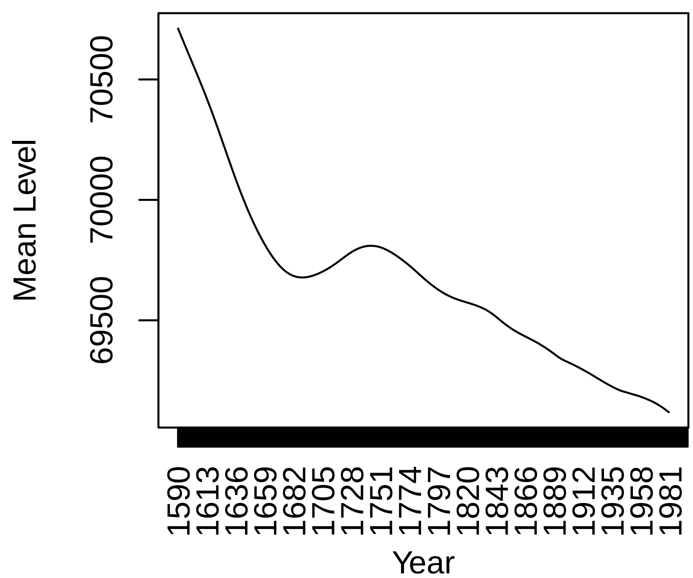


Figure 17: Mean Level Plot of South Geomagnetic Pole

```
## initial  value 3.227280
## iter   2 value 3.030625
## iter   3 value 2.535786
## iter   4 value 2.317536
## iter   5 value 2.155372
## iter   6 value 1.884370
## iter   7 value 1.783309
## iter   8 value 1.655134
## iter   9 value 1.623155
```

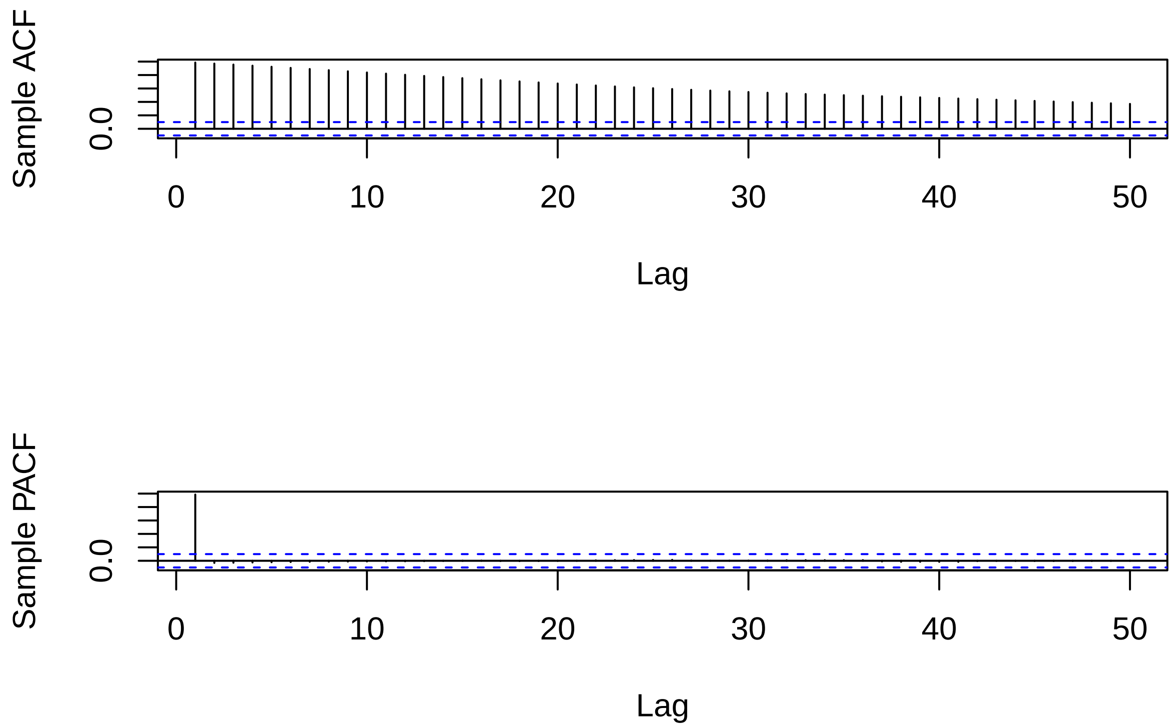


Figure 18: ACF and PACF of South Geomagnetic Pole

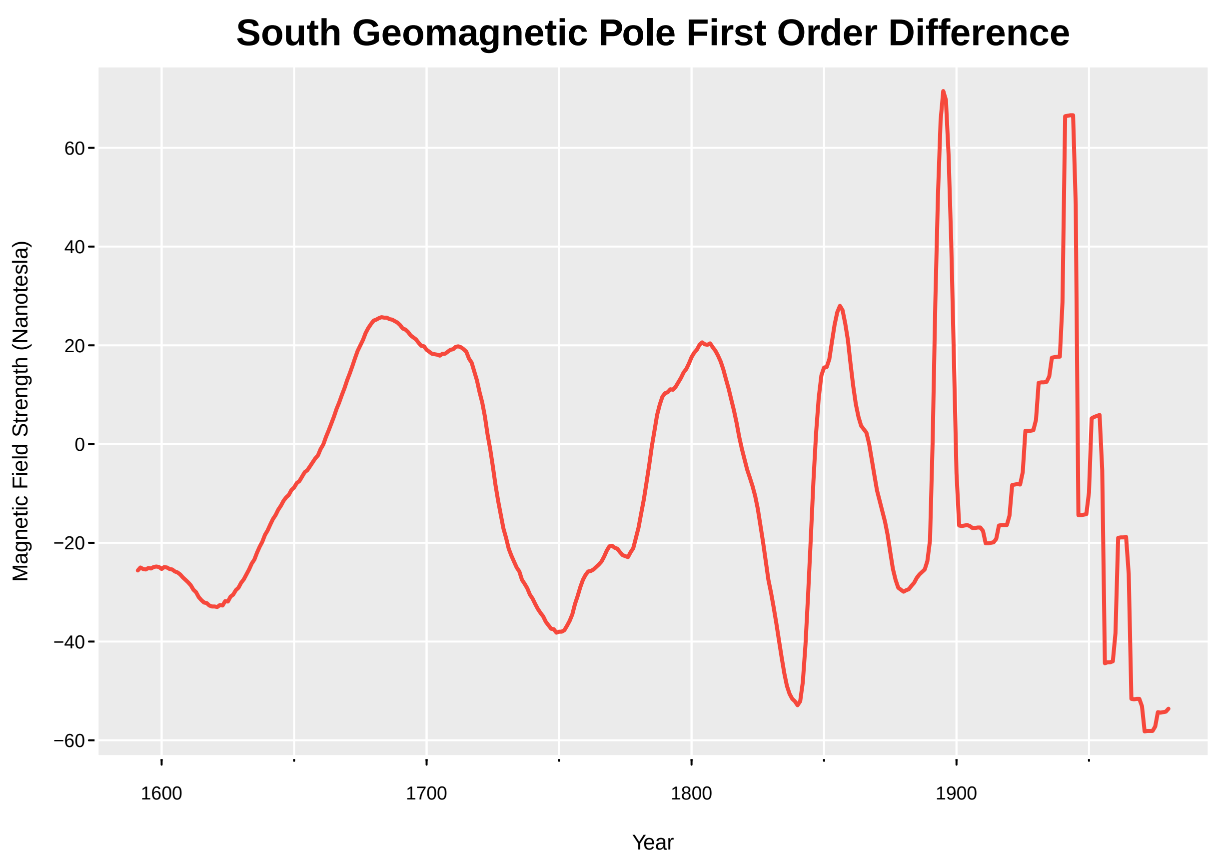


Figure 19: Time Series of South Geomagnetic Pole Taking First Order Non-Seasonal Difference

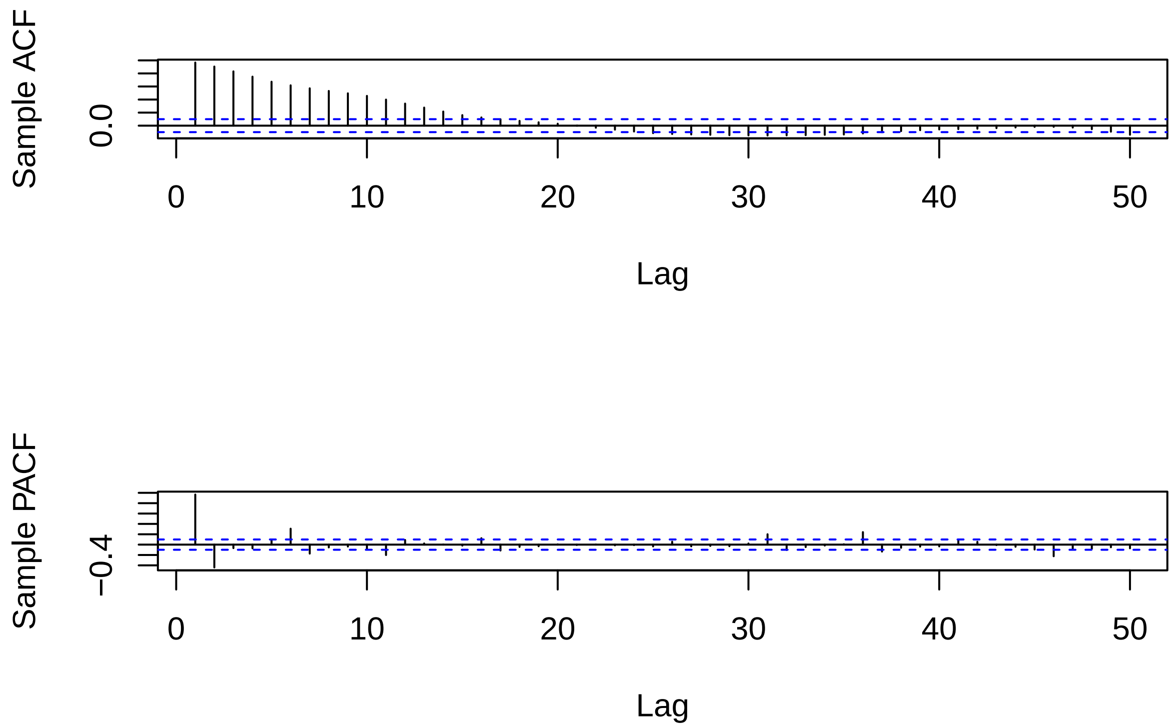


Figure 20: ACF and PACF of South Geomagnetic Pole Taking First Order Non-Seasonal Difference

```
## iter  10 value 1.546794
## iter  11 value 1.523308
## iter  12 value 1.500793
## iter  13 value 1.492767
## iter  14 value 1.489112
## iter  15 value 1.486878
## iter  16 value 1.486732
## iter  17 value 1.485600
## iter  18 value 1.485365
## iter  19 value 1.484996
## iter  20 value 1.483969
## iter  21 value 1.482959
## iter  22 value 1.480807
## iter  23 value 1.478997
## iter  24 value 1.477745
## iter  25 value 1.477418
## iter  26 value 1.476290
## iter  27 value 1.475677
## iter  28 value 1.475319
## iter  29 value 1.475061
## iter  30 value 1.474957
## iter  31 value 1.474933
## iter  32 value 1.474894
## iter  33 value 1.474887
## iter  34 value 1.474884
## iter  35 value 1.474881
## iter  36 value 1.474879
## iter  37 value 1.474876
## iter  38 value 1.474874
## iter  39 value 1.474868
## iter  40 value 1.474865
## iter  41 value 1.474856
## iter  42 value 1.474847
## iter  43 value 1.474839
## iter  44 value 1.474831
## iter  45 value 1.474823
## iter  46 value 1.474818
## iter  47 value 1.474815
## iter  48 value 1.474814
## iter  49 value 1.474813
## iter  50 value 1.474811
## iter  51 value 1.474807
## iter  52 value 1.474797
## iter  53 value 1.474772
## iter  54 value 1.474736
## iter  55 value 1.474681
## iter  56 value 1.474551
## iter  57 value 1.474530
## iter  58 value 1.474510
## iter  59 value 1.474458
## iter  60 value 1.474284
## iter  61 value 1.474263
## iter  62 value 1.474237
## iter  63 value 1.474205
```

```
## iter  64 value 1.474180
## iter  65 value 1.474151
## iter  66 value 1.474130
## iter  67 value 1.474114
## iter  68 value 1.473948
## iter  69 value 1.473171
## iter  70 value 1.472852
## iter  71 value 1.471852
## iter  72 value 1.470671
## iter  73 value 1.470585
## iter  74 value 1.470192
## iter  75 value 1.469164
## iter  76 value 1.467991
## iter  77 value 1.467000
## iter  78 value 1.466338
## iter  79 value 1.465836
## iter  80 value 1.465515
## iter  81 value 1.465038
## iter  82 value 1.464908
## iter  83 value 1.464476
## iter  84 value 1.463597
## iter  85 value 1.462554
## iter  86 value 1.459871
## iter  87 value 1.457713
## iter  88 value 1.454495
## iter  89 value 1.451626
## iter  90 value 1.450934
## iter  91 value 1.449040
## iter  92 value 1.448341
## iter  93 value 1.447299
## iter  94 value 1.447294
## iter  95 value 1.446606
## iter  96 value 1.445941
## iter  97 value 1.445475
## iter  98 value 1.444955
## iter  99 value 1.444502
## iter 100 value 1.443927
## final  value 1.443927
## stopped after 100 iterations
## initial  value 3.223083
## iter   2 value 2.469999
## iter   3 value 1.597443
## iter   4 value 1.583039
## iter   5 value 1.572064
## iter   6 value 1.548684
## iter   7 value 1.542101
## iter   8 value 1.520808
## iter   9 value 1.510434
## iter  10 value 1.505749
## iter  11 value 1.495436
## iter  12 value 1.493734
## iter  13 value 1.491659
## iter  14 value 1.490909
## iter  15 value 1.489737
```

```
## iter  16 value 1.488891
## iter  17 value 1.488517
## iter  18 value 1.488166
## iter  19 value 1.487785
## iter  20 value 1.486840
## iter  21 value 1.485440
## iter  22 value 1.483049
## iter  23 value 1.480641
## iter  24 value 1.479807
## iter  25 value 1.477869
## iter  26 value 1.476690
## iter  27 value 1.475941
## iter  28 value 1.475096
## iter  29 value 1.474543
## iter  30 value 1.473654
## iter  31 value 1.473170
## iter  32 value 1.472589
## iter  33 value 1.471678
## iter  34 value 1.471597
## iter  35 value 1.470432
## iter  36 value 1.470204
## iter  37 value 1.469823
## iter  38 value 1.469562
## iter  39 value 1.468990
## iter  40 value 1.468706
## iter  41 value 1.468215
## iter  42 value 1.468043
## iter  43 value 1.467690
## iter  44 value 1.467142
## iter  45 value 1.466677
## iter  46 value 1.466300
## iter  47 value 1.465815
## iter  48 value 1.465238
## iter  49 value 1.464648
## iter  50 value 1.463938
## iter  51 value 1.462505
## iter  52 value 1.462185
## iter  53 value 1.461286
## iter  54 value 1.458462
## iter  55 value 1.456669
## iter  56 value 1.454962
## iter  57 value 1.454076
## iter  58 value 1.453825
## iter  59 value 1.453277
## iter  60 value 1.452886
## iter  61 value 1.452401
## iter  62 value 1.451929
## iter  63 value 1.451693
## iter  64 value 1.451657
## iter  65 value 1.451619
## iter  66 value 1.451514
## iter  67 value 1.451391
## iter  68 value 1.451258
## iter  69 value 1.451212
```

```
## iter  70 value 1.450998
## iter  71 value 1.450979
## iter  72 value 1.450904
## iter  73 value 1.450868
## iter  74 value 1.450851
## iter  75 value 1.450835
## iter  76 value 1.450826
## iter  77 value 1.450809
## iter  78 value 1.450794
## iter  79 value 1.450782
## iter  80 value 1.450775
## iter  81 value 1.450771
## iter  82 value 1.450767
## iter  83 value 1.450762
## iter  84 value 1.450759
## iter  85 value 1.450758
## iter  86 value 1.450756
## iter  87 value 1.450755
## iter  88 value 1.450755
## iter  89 value 1.450754
## iter  90 value 1.450754
## iter  91 value 1.450754
## iter  92 value 1.450754
## iter  93 value 1.450754
## iter  94 value 1.450754
## iter  95 value 1.450754
## iter  95 value 1.450754
## iter  95 value 1.450754
## final  value 1.450754
## converged
```

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2      ar3     ar4      ar5     ar6     ma1     ma2     ma3
##       0.6598  -0.0458  -0.0015  0.0171  -0.5651  0.6809  0.8658  0.8449  0.9406
## s.e.  0.1065   0.1112   0.0639  0.1214   0.0914  0.0493  0.1285  0.1142  0.1025
##          ma4     ma5     ma6     ma7     ma8    xmean
##       1.0117  1.1670  0.7639  0.4559  0.2232  -9.9797
## s.e.  0.1534  0.1479  0.1046  0.1016  0.1063   5.9339
##
## sigma^2 estimated as 17.66:  log likelihood = -1119.18,  aic = 2270.36
##
## $degrees_of_freedom
## [1] 375
##
## $ttable
##       Estimate     SE t.value p.value
```

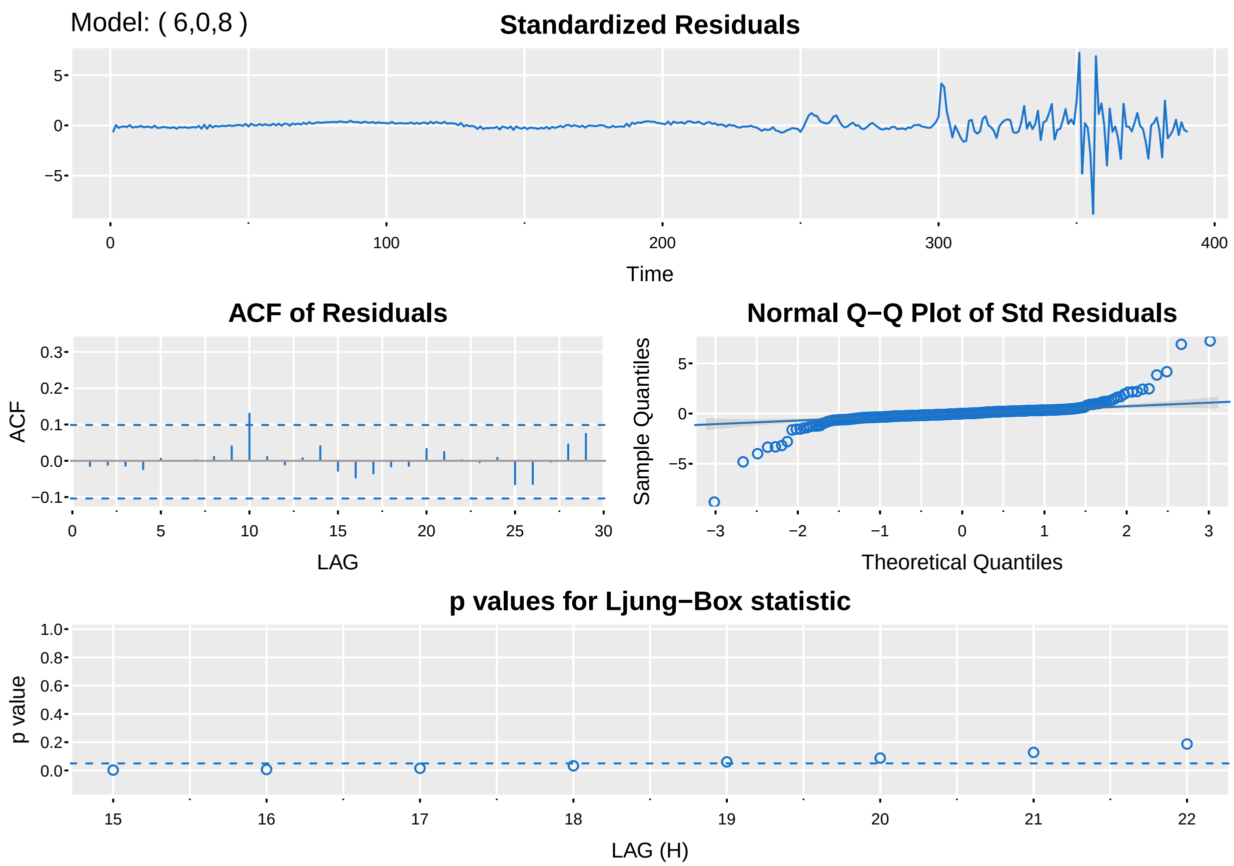


Figure 21: Standardized Residuals Plot, ACF of Residuals, QQ Plot, and Ljung-Box Test for ARIMA(6,0,8)

```
## ar1      0.6598 0.1065  6.1946  0.0000
## ar2     -0.0458 0.1112 -0.4121  0.6805
## ar3     -0.0015 0.0639 -0.0237  0.9811
## ar4      0.0171 0.1214  0.1411  0.8879
## ar5     -0.5651 0.0914 -6.1798  0.0000
## ar6      0.6809 0.0493 13.8036  0.0000
## ma1      0.8658 0.1285  6.7394  0.0000
## ma2      0.8449 0.1142  7.3968  0.0000
## ma3      0.9406 0.1025  9.1787  0.0000
## ma4      1.0117 0.1534  6.5960  0.0000
## ma5      1.1670 0.1479  7.8880  0.0000
## ma6      0.7639 0.1046  7.3016  0.0000
## ma7      0.4559 0.1016  4.4858  0.0000
## ma8      0.2232 0.1063  2.1000  0.0364
## xmean  -9.9797 5.9339 -1.6818  0.0934
##
## $AIC
## [1] 5.821436
##
## $AICc
## [1] 5.824727
##
## $BIC
## [1] 5.98415
```

```
## initial  value 3.228014
## iter   2 value 3.068223
## iter   3 value 2.625824
## iter   4 value 2.272138
## iter   5 value 2.001111
## iter   6 value 1.890752
## iter   7 value 1.808364
## iter   8 value 1.763334
## iter   9 value 1.663816
## iter  10 value 1.542788
## iter  11 value 1.522895
## iter  12 value 1.506727
## iter  13 value 1.497904
## iter  14 value 1.492781
## iter  15 value 1.488108
## iter  16 value 1.484397
## iter  17 value 1.482146
## iter  18 value 1.479424
## iter  19 value 1.478069
## iter  20 value 1.476811
## iter  21 value 1.476409
## iter  22 value 1.475429
## iter  23 value 1.471912
## iter  24 value 1.470119
## iter  25 value 1.467335
## iter  26 value 1.460841
## iter  27 value 1.460452
## iter  28 value 1.459459
## iter  29 value 1.458030
```

```
## iter  30 value 1.456695
## iter  31 value 1.450315
## iter  32 value 1.445866
## iter  33 value 1.443840
## iter  34 value 1.442276
## iter  35 value 1.441839
## iter  36 value 1.441217
## iter  37 value 1.440770
## iter  38 value 1.440624
## iter  39 value 1.440465
## iter  40 value 1.440156
## iter  41 value 1.439685
## iter  42 value 1.439123
## iter  43 value 1.438533
## iter  44 value 1.437882
## iter  45 value 1.437584
## iter  46 value 1.437198
## iter  47 value 1.436571
## iter  48 value 1.435930
## iter  49 value 1.434683
## iter  50 value 1.433899
## iter  51 value 1.432714
## iter  52 value 1.432051
## iter  53 value 1.430956
## iter  54 value 1.430538
## iter  55 value 1.429837
## iter  56 value 1.429519
## iter  57 value 1.429116
## iter  58 value 1.429023
## iter  59 value 1.428967
## iter  60 value 1.428963
## iter  61 value 1.428961
## iter  62 value 1.428958
## iter  63 value 1.428942
## iter  64 value 1.428924
## iter  65 value 1.428918
## iter  66 value 1.428909
## iter  67 value 1.428899
## iter  68 value 1.428897
## iter  69 value 1.428896
## iter  70 value 1.428896
## iter  71 value 1.428895
## iter  72 value 1.428895
## iter  73 value 1.428895
## iter  74 value 1.428895
## iter  75 value 1.428895
## iter  75 value 1.428895
## iter  75 value 1.428895
## final  value 1.428895
## converged
## initial  value 1.451198
## iter   2 value 1.451132
## iter   3 value 1.450918
## iter   4 value 1.449700
```

```
## iter   5 value 1.449119
## iter   6 value 1.448689
## iter   7 value 1.448433
## iter   8 value 1.448263
## iter   9 value 1.448078
## iter  10 value 1.447974
## iter  11 value 1.447814
## iter  12 value 1.447555
## iter  13 value 1.447361
## iter  14 value 1.446832
## iter  15 value 1.446342
## iter  16 value 1.445845
## iter  17 value 1.445251
## iter  18 value 1.444499
## iter  19 value 1.443674
## iter  20 value 1.443254
## iter  21 value 1.443120
## iter  22 value 1.443019
## iter  23 value 1.442969
## iter  24 value 1.442945
## iter  25 value 1.442929
## iter  26 value 1.442905
## iter  27 value 1.442885
## iter  28 value 1.442869
## iter  29 value 1.442863
## iter  30 value 1.442861
## iter  31 value 1.442861
## iter  32 value 1.442861
## iter  33 value 1.442861
## iter  34 value 1.442860
## iter  34 value 1.442860
## final  value 1.442860
## converged
```

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     ar2      ar3     ar4      ar5     ar6     ar7     ma1     ma2
##       -0.0932  0.5431  -0.1236  0.0919  -0.5064  0.1098  0.6199  1.6026  1.3643
## s.e.   0.1034  0.0969   0.0716  0.0734   0.0582  0.1008  0.0964  0.1054  0.1188
##          ma3     ma4     ma5     ma6     ma7     ma8     xmean
##       1.4698  1.5214  1.6573  1.5147  0.8925  0.3441  -10.1381
## s.e.  0.1155  0.0991  0.1019  0.1321  0.1068  0.0607    6.4896
##
## sigma^2 estimated as 17.35:  log likelihood = -1116.1,  aic = 2266.2
##
## $degrees_of_freedom
## [1] 374
```

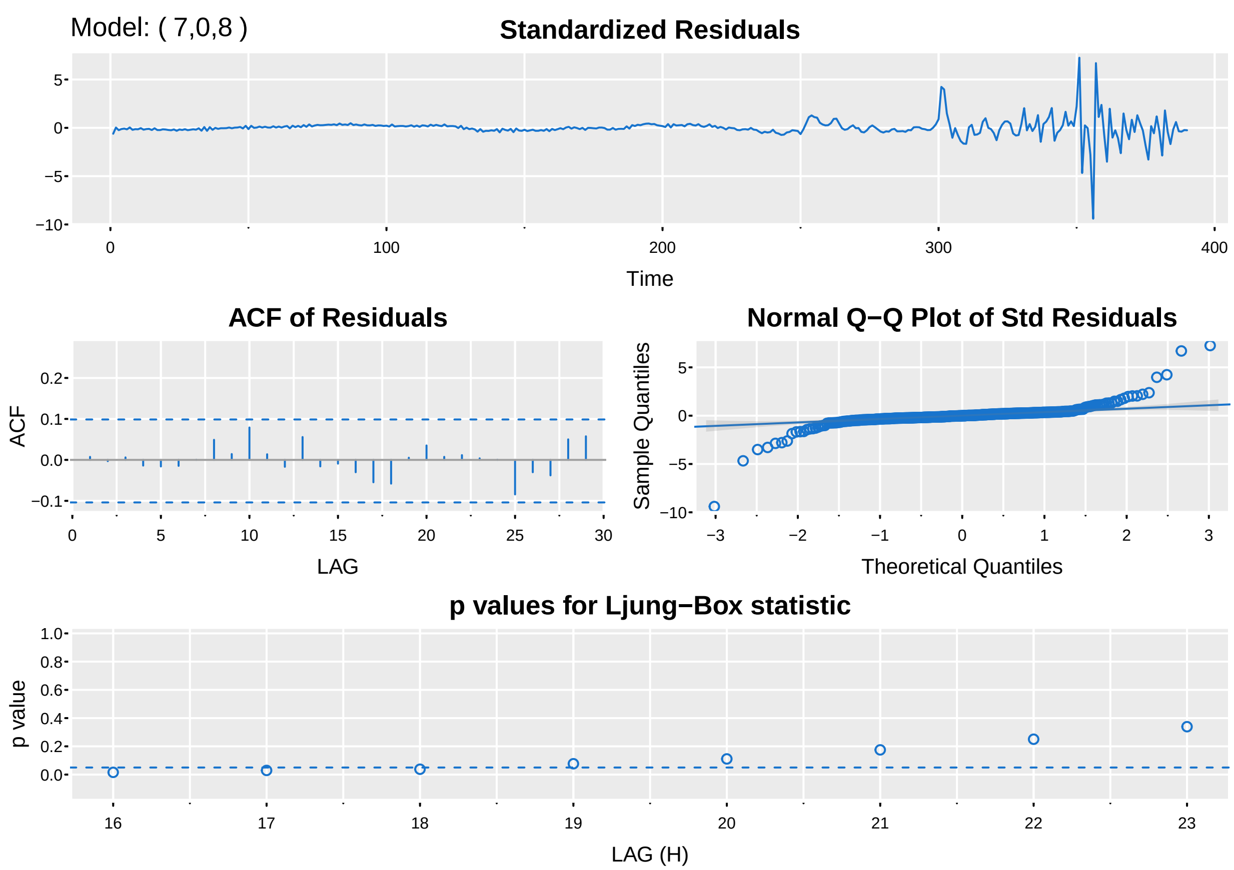


Figure 22: Standardized Residuals Plot, ACF of Residuals, QQ Plot, and Ljung-Box Test for ARIMA(7,0,8)

```
##
## $ttable
##       Estimate     SE t.value p.value
## ar1    -0.0932 0.1034 -0.9006  0.3684
## ar2     0.5431 0.0969  5.6028  0.0000
## ar3    -0.1236 0.0716 -1.7265  0.0851
## ar4     0.0919 0.0734  1.2513  0.2116
## ar5    -0.5064 0.0582 -8.6960  0.0000
## ar6     0.1098 0.1008  1.0890  0.2769
## ar7     0.6199 0.0964  6.4288  0.0000
## ma1     1.6026 0.1054 15.2076  0.0000
## ma2     1.3643 0.1188 11.4793  0.0000
## ma3     1.4698 0.1155 12.7219  0.0000
## ma4     1.5214 0.0991 15.3492  0.0000
## ma5     1.6573 0.1019 16.2634  0.0000
## ma6     1.5147 0.1321 11.4692  0.0000
## ma7     0.8925 0.1068  8.3595  0.0000
## ma8     0.3441 0.0607  5.6694  0.0000
## xmean -10.1381 6.4896 -1.5622  0.1191
##
## $AIC
## [1] 5.810777
##
## $AICc
## [1] 5.814517
##
## $BIC
## [1] 5.983661
```