Introduction

Overview

This report is concerned with performing exploratory analysis and exploring feasible ARIMA models for a dataset describing yearly Global Land Temperature Anomalies in Degrees Celsius against the base period 1901-2000. The data provided contains temperature anomalies covering the years from 1850 to 2023, with each anomaly representing the deviation of temperature in Degrees Celsius for that year compared to the mean temperature of the baseline period (1901-2000).

Objectives

- Perform descriptive analysis of the dataset.
- Shortlist ARMA and ARIMA models and analyse the models for selection using ACF-PACF, EACF, BIC etc.
- Fit the selected models and find parameter estimates
- Utilize goodness-of-fit metrics such as AIC, BIC, MSE etc. to select the optimal model.

Method and Results

Visualising the Series

In the first step, the dataset is loaded into the environment and summary obtained. The summaries show a mean anomaly value of 0.06218°C, a max of 0.91°C and a min of -0.44°C. The 1st quartile for anomalies is -0.12750°C and the 3rd quartile is 0.23°C. The median anomaly value is 0.

```
anomalies = read.csv("assignment2Data2024.csv", col.names = c("Year", "Anomaly"))
summary(anomalies)
```

```
##       Year         Anomaly
##  Min.   :1850   Min.   :-0.44000
##  1st Qu.:1893   1st Qu.:-0.12750
##  Median :1936   Median : 0.00000
##  Mean   :1936   Mean   : 0.06218
##  3rd Qu.:1980   3rd Qu.: 0.23000
##  Max.   :2023   Max.   : 0.91000
```

The anomaly column is converted to a time-series object with a frequency of 1 since it is annual data. The series is then plotted to observe any trends or seasonality in Figure 1. The initial plot demonstrates a slowly decreasing trend till the early 1900s, after which a linear increasing trend is seen. The series seems to have both AR and MA components since there are also fluctuations in the series throughout. Not much changing variance is observed.

```
anomalies.ts = ts(anomalies[,c(2)], frequency = 1, start = 1850, end = 2023)

plot(anomalies.ts, type='o', ylab='Anomalies in degrees Celsius', xlab='Year', main = "Time series plot of an
nual anomalies in degrees Celsius")
```

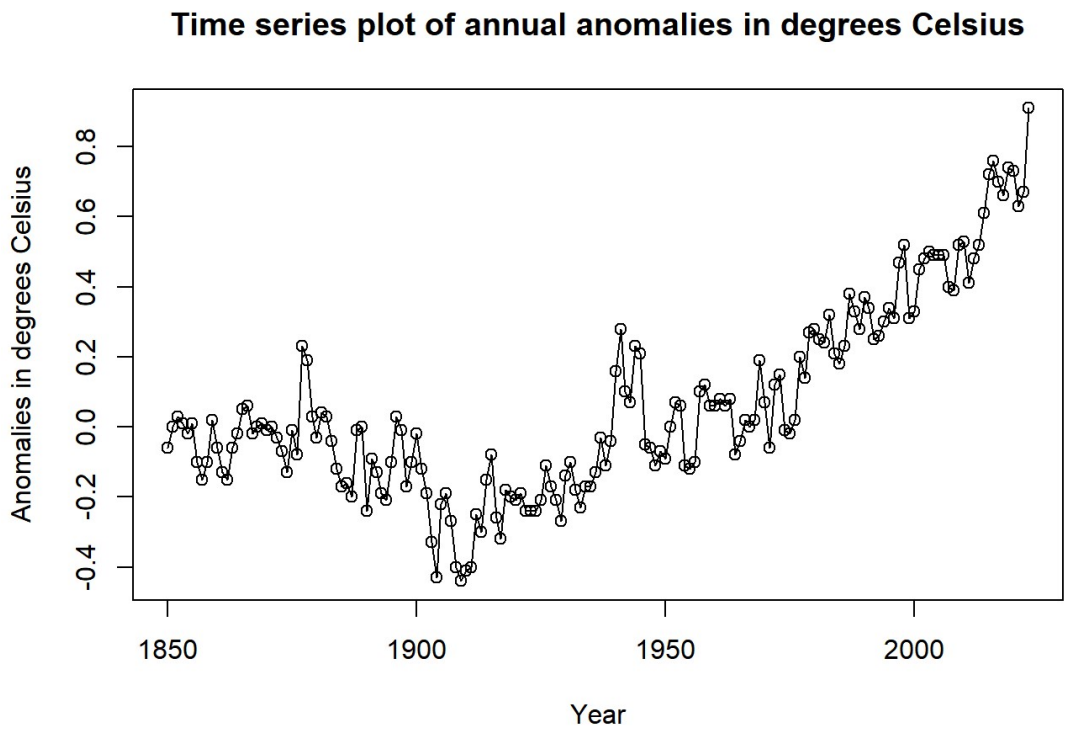

Figure 1: Plot of the annual anomalies in degrees Celsius

Verifying Stationarity and Normality

The ACF and PACF plots for the series are shown in Figure 2. The ACF plot returns a slowly decaying plot with most peaks beyond the confidence intervals and the PACF plot produces two significant peaks beyond the confidence intervals at the first and third lags. Both of these suggest non-stationarity in the series due to the autocorrelations observed at multiple lags.

```
par(mfrow=c(2,1), mar=c(3,3,3,1))
acf(anomalies.ts, main="ACF plot", lag.max = 40)
pacf(anomalies.ts, main="PACF plot", lag.max = 40)
```

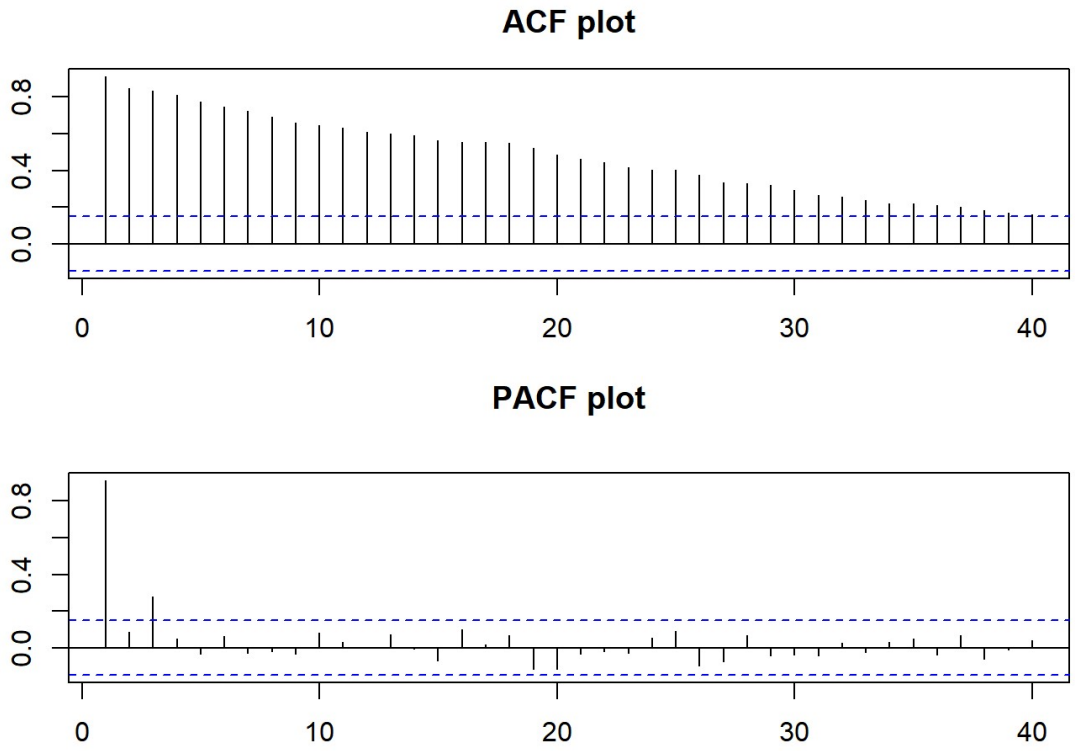

Figure 2: ACF plot of the time series anomalies data

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

Checking the stationarity using an Automated Dickey-Fuller (ADF) test reveals a p-value of 0.9044, affirming the null hypothesis at the 5% significance level. This confirms that the series is not stationary.

The QQ plot of the series in Figure 3 shows both tails deviating from the reference line suggesting visually that the data points are not normally distributed. A Shapiro-Wilk test confirms this producing a p-value <0.05, affirming the alternate hypothesis that the series is not normally distributed.

```
adf.test(anomalies.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  anomalies.ts
## Dickey-Fuller = -1.1974, Lag order = 5, p-value = 0.9044
## alternative hypothesis: stationary
```

```
qqnorm(anomalies.ts, ylab="Anomalies in degrees Celsius", xlab="Normal Scores")
qqline(anomalies.ts)
```

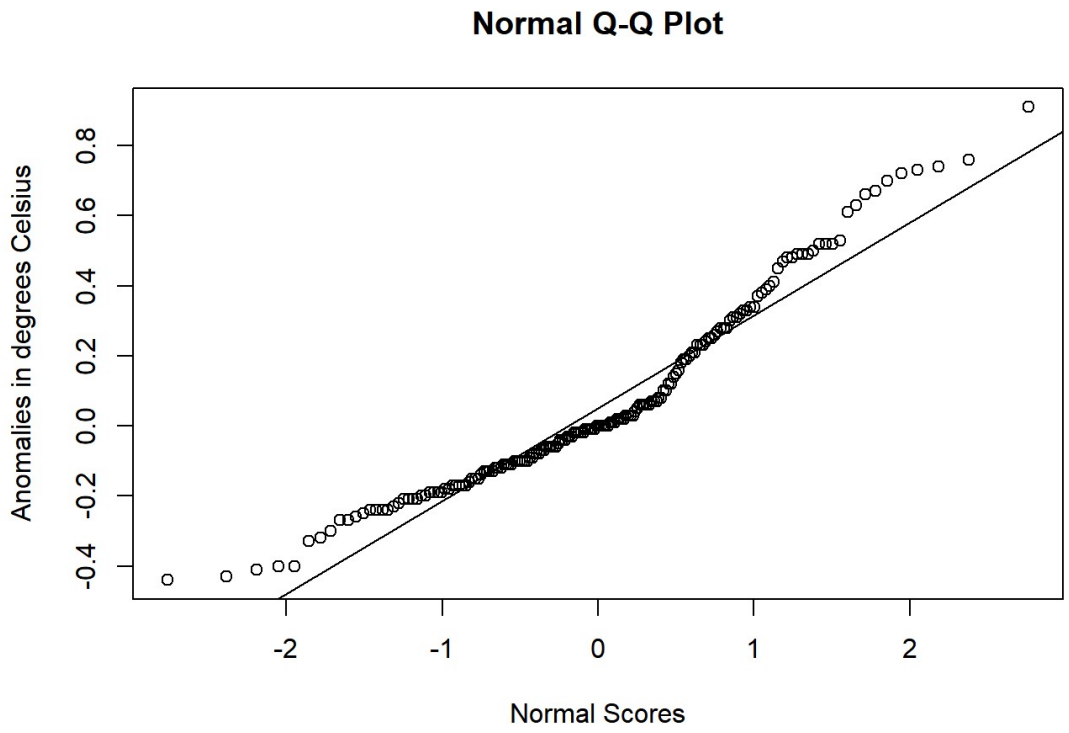

Figure 3: Q-Q plot of the time series anomalies data

```
shapiro.test(anomalies.ts)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  anomalies.ts
## W = 0.94279, p-value = 1.897e-06
```

Transforming the Series

A Box-Cox transformation may help with achieving normality in the data. A Box-Cox transformation is applied to a shifted version of the series (to ensure all values are positive), producing a lambda value of 1. A lambda value of 1 indicates no transformation, so the Box-Cox transformation is not carried out on the series.

A log transformation is then applied. The time series plot for the log transformed series is shown in Figure 4, its Q-Q plot vs the original Q-Q plot is shown in Figure 5, and ACF and PACF plots for the log transformed series are shown in Figure 6. The plot of the log transformed series does not look more stabilized compared to the original plot but has a significant dip in the early 1900s. The QQ plots compared side by side show that the Log transformed series deviates less from the reference line compared to the non-transformed series, indicating that it is more normally distributed. However overall, the transformed series is still not normally distributed. A Shapiro-Wilk test performed on the transformed series still shows a very small p-value (1.052e-15), confirming non-normal distribution. Performing an ADF test reveals that the series is still not stationary due to a high p-value (0.2184).

```
summarise_series = function(series, transform) {
  plot(series, type='o',ylab = "Anomalies in degrees Celsius", main=paste('Time series plot of ', transform,
' of anomalies series'))
  par(mfrow=c(1,2), mar=c(3,4,3,1))
  qqnorm(series, main=paste('Q-Q plot of ', transform, ' series'), ylab=paste(transform, ' anomalies'), xlab
="Normal Scores")
  qqline(series)
  qqnorm(anomalies.ts, main="Original series Q-Q plot", ylab="Anomalies in degrees Celsius", xlab="Normal Sco
res")
  qqline(anomalies.ts)
  par(mfrow=c(1,1))

  par(mfrow=c(1,2), mar=c(3,3,5,1))
  acf(series, main=paste('ACF of the ', transform, ' series.'))
  pacf(series, main=paste('PACF of the ', transform, ' series.'))
  par(mfrow=c(1,1))

  print(shapiro.test(series))
  print(adf.test(series))
}

adjusted.series = anomalies.ts+abs(min(anomalies.ts))+0.001
BC = BoxCox.ar(adjusted.series)
```

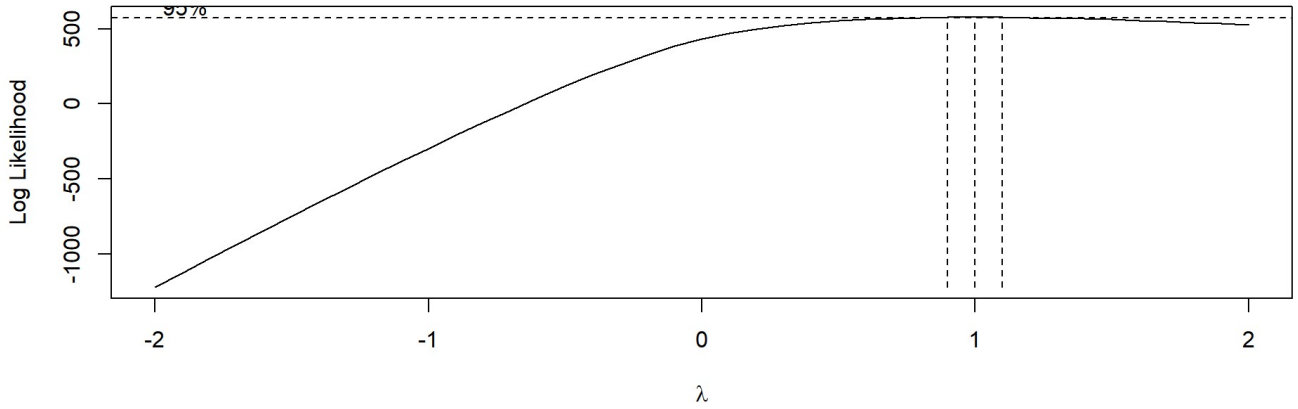

Figure 6: ACF and PACF plots for log transformed anomalies series

```
BC$ci
```

```
## [1] 0.9 1.1
```

```
lambda <- BC$lambda[which(max(BC$loglike) == BC$loglike)]
lambda
```

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

```
log.anomalies = log(adjusted.series)
summarise_series(log.anomalies, "Log")
```

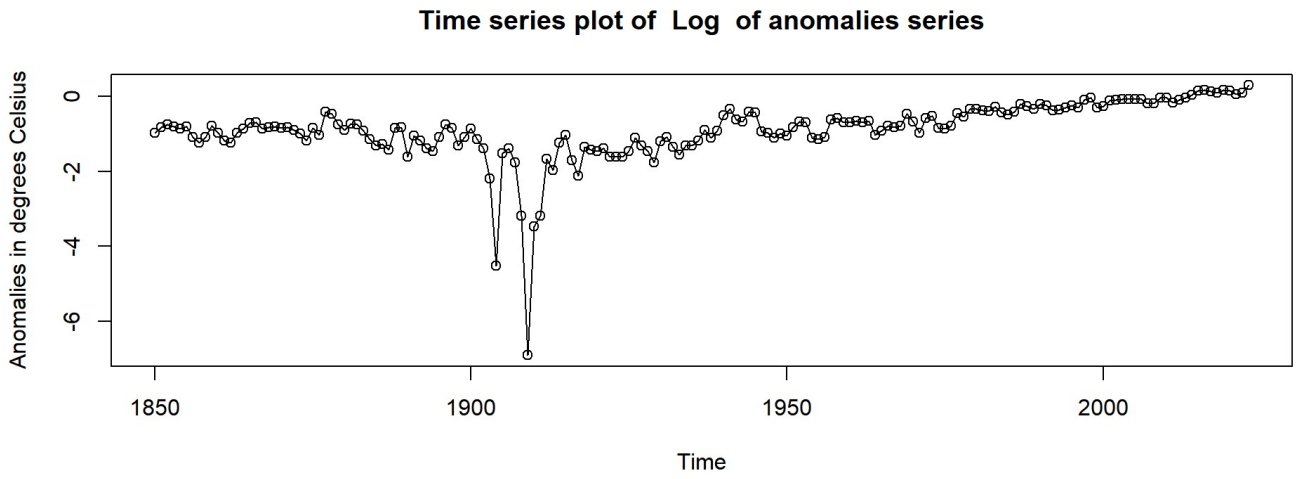

Figure 4: Time series plot of log transformed anomalies series

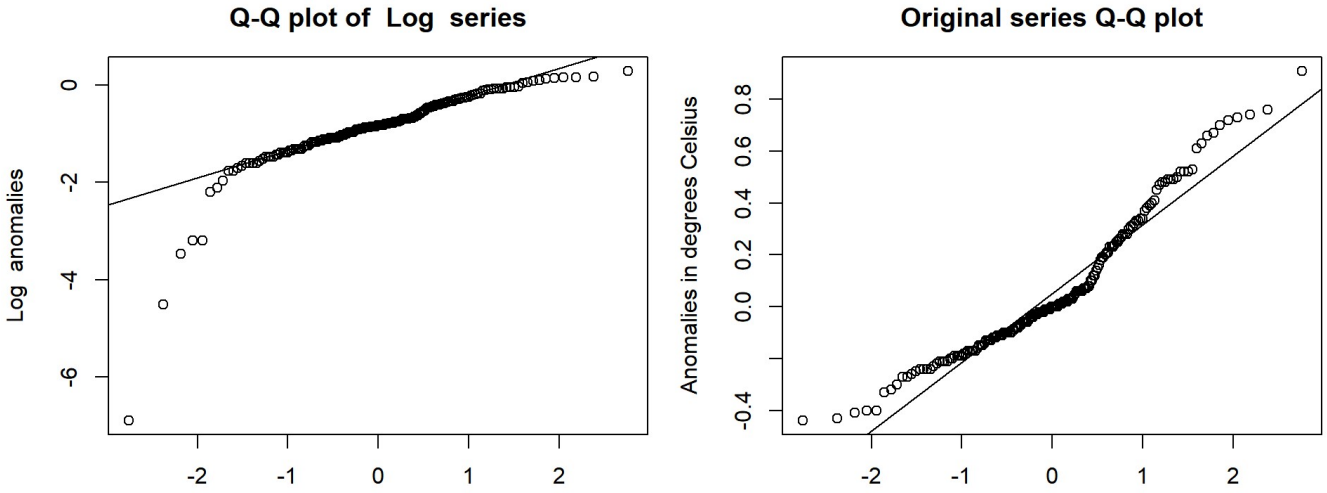

Figure 5: Q-Q plots for log transformed anomalies series vs original series

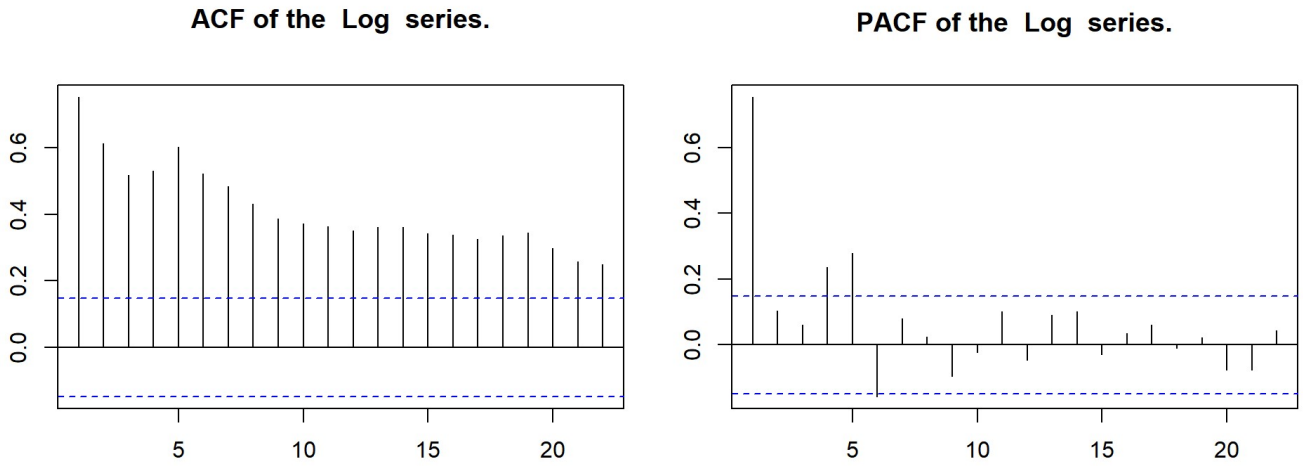

Figure 6: ACF and PACF plots for log transformed anomalies series

```
##
##  Shapiro-Wilk normality test
##
## data:  series
## W = 0.75611, p-value = 1.052e-15
##
##
##  Augmented Dickey-Fuller Test
##
## data:  series
## Dickey-Fuller = -2.8568, Lag order = 5, p-value = 0.2184
## alternative hypothesis: stationary
```

Differencing the Series

The first difference of the series is obtained to attempt at making the series stationary. The time series plot for the 1st-differenced series is shown in Figure 7, its Q-Q plot vs the original Q-Q plot is shown in Figure 8, and ACF and PACF plots for the 1st-differenced series are shown in Figure 9. The 1st difference of the anomalies series is plotted and is visually observed to have a constant mean and not much changing variance. This is confirmed with the results of the ADF test producing a p-value of 0.01, indicating that the null hypothesis is rejected and the 1st-differenced series is stationary. The Q-Q plot of the 1st-differenced series deviates less from the reference line compared to the original series and can be said to be more normal. The results of the Shapiro-Wilk test confirm normality, with a high p-value of 0.3249. The ACF plot for the 1st-differenced series shows a significant peak at the second lag and a borderline peak at the fourth lag. A couple of significant peaks are also observed at higher lags but these are ignored as they are late lags. The PACF plot is very similar to the ACF plot, with a significant peak at the second lag. Therefore, q can be assumed to be 2 and p can be taken to be 1 based on the ACF and PACF plots respectively. These p and q values suggest the modeling of ARIMA(1,1,1) and ARIMA(1,1,2)

Since the first difference of the anomalies series provided satisfactory results in terms of normality and stationarity, no more differencing is performed in order to avoid over-differencing.

```
diff.anomalies = diff(anomalies.ts, differences = 1)
summarise_series(diff.anomalies, "1st difference")
```

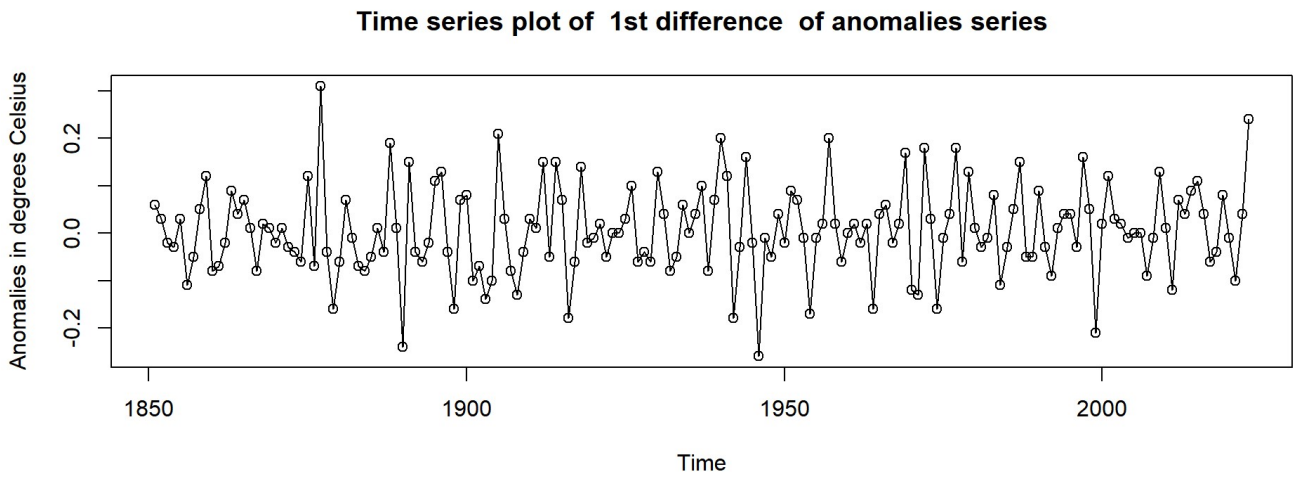

Figure 7: Time series plot of 1st-differenced anomalies series

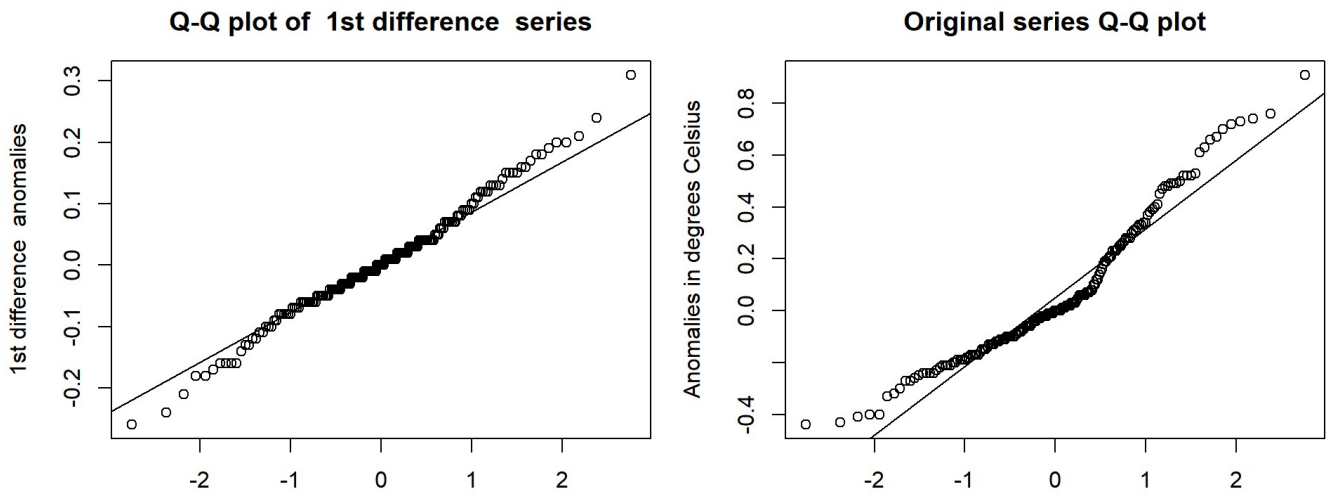

Figure 8: Q-Q plots for 1st-differenced anomalies series vs original series

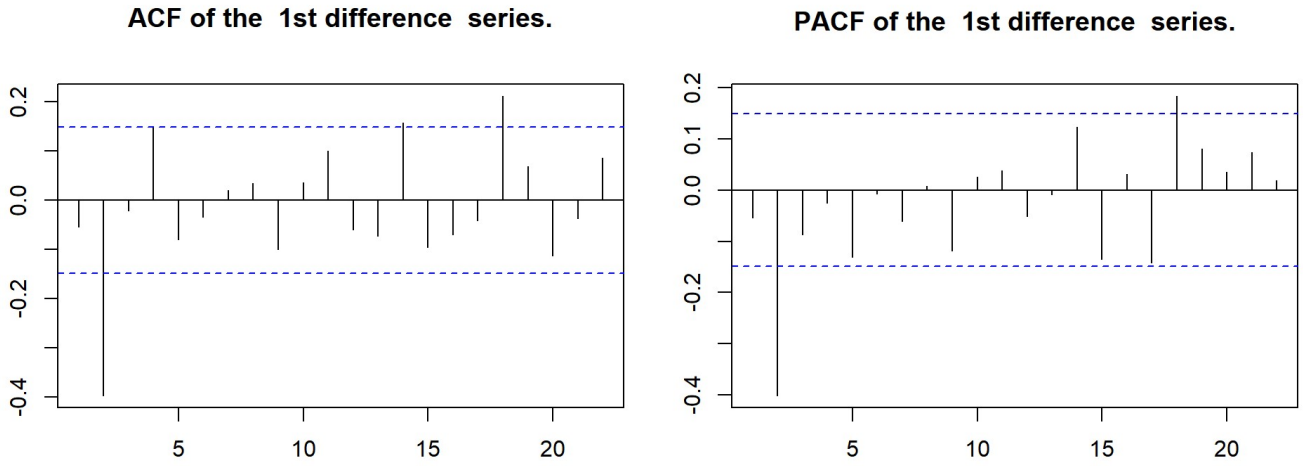

Figure 9: ACF and PACF plots for 1st-differenced anomalies series

```
##
##  Shapiro-Wilk normality test
##
## data:  series
## W = 0.99073, p-value = 0.3249
##
##
##  Augmented Dickey-Fuller Test
##
## data:  series
## Dickey-Fuller = -7.0878, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

EACF Table

The EACF table for the differenced series is shown below. The top-left ‘o’ is obtained from the EACF table and this is observed to occur at AR = 0 and MA = 2. This results in the possible models being ARIMA(0,1,2), ARIMA(1,1,2), ARIMA(1,1,3), and ARIMA(0,1,3) based on the best model and its neighbors.

```
eacf(diff.anomalies)
```

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

The BIC table for the differenced series is shown in Figure 10. Plotting the BIC table for the differenced series shows that the top 4 models all have p=2. The second best model also has error lag of 5. Based on this, the shortlisted models are ARIMA(2,1,0), ARIMA(2,1,4).

```
res = armasubsets(y=diff.anomalies, nar=5, nma=5, y.name='p', ar.method='ols')
```

```
## Reordering variables and trying again:
```

```
plot(res)
```

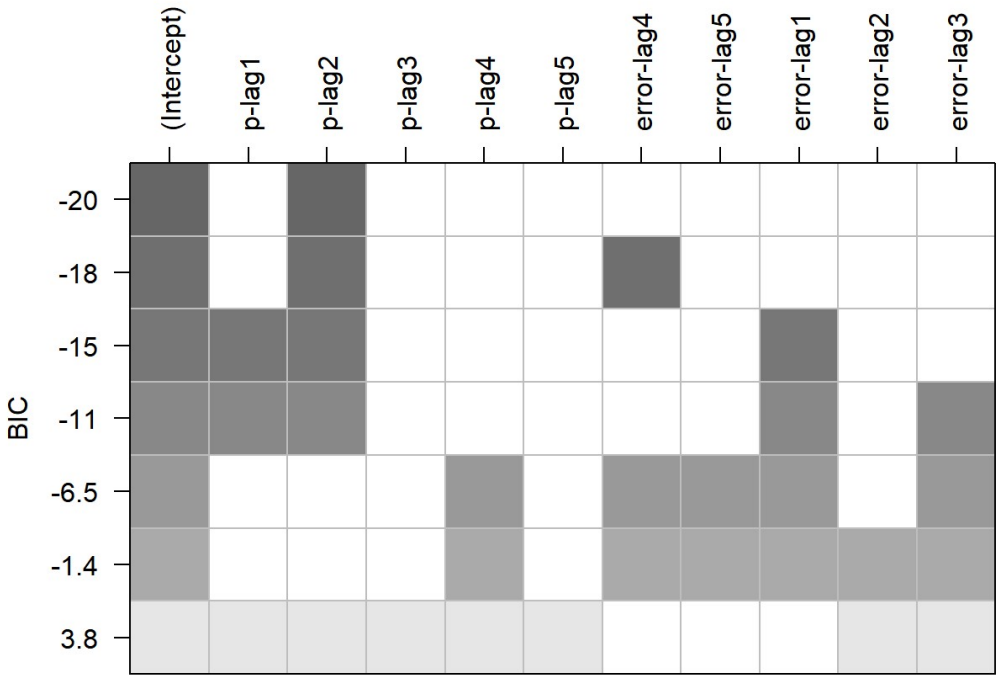

Figure 10: BIC table for 1st-differenced anomalies series

Parameter Estimation

Based on the above model selection, the final models for modeling are listed below:

- ARIMA(0,1,2)
- ARIMA(1,1,1)
- ARIMA(1,1,2)
- ARIMA(0,1,3)
- ARIMA(2,1,0)
- ARIMA(2,1,4)

ARIMA(0,1,2)

Initially, the ARIMA(0,1,2) model is fitted and the coefficients are found using the coeftest() function using the ML estimator. This produces coefficients for the MA1 and MA2 components that are both significant, with MA1 being less significant than MA2. Using the CSS estimator produces roughly similar p-values for each coefficient.

```
model.012ML = Arima(anomalies.ts, order=c(0,1,2), method='ML')
model.012ML
```

```
## Series: anomalies.ts
## ARIMA(0,1,2)
##
## Coefficients:
##           ma1      ma2
##       -0.1547  -0.3769
## s.e.   0.0674   0.0621
##
## sigma^2 = 0.007449:  log likelihood = 179.16
## AIC=-352.33   AICc=-352.18   BIC=-342.87
```

```
coeftest(model.012ML)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.154653   0.067355 -2.2961   0.02167 *
## ma2 -0.376908   0.062077 -6.0716 1.267e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.012CSS = Arima(anomalies.ts, order=c(0,1,2), method='CSS')
model.012CSS
```

```
## Series: anomalies.ts
## ARIMA(0,1,2)
##
## Coefficients:
##           ma1      ma2
##       -0.1557  -0.3801
## s.e.   0.0671   0.0621
##
## sigma^2 = 0.007453:  log likelihood = 179.31
```

```
coeftest(model.012CSS)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.155660   0.067130 -2.3188   0.02041 *
## ma2 -0.380127   0.062131 -6.1182 9.466e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ARIMA(1,1,1)

The ARIMA(1,1,1) model is fitted and the coefficients are found using the coeftest() function using the ML estimator. This produces coefficients for the AR1 and MA1 components that are both significant, with AR1 being less significant than MA2, but both are highly significant regardless. Using the CSS estimator produces roughly similar p-values for each coefficient.

```
model.111ML = Arima(anomalies.ts, order=c(1,1,1), method='ML')
model.111ML
```

```
## Series: anomalies.ts
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1      ma1
##       0.5033  -0.7849
## s.e.  0.1118   0.0736
##
## sigma^2 = 0.008072:  log likelihood = 172.28
## AIC=-338.56   AICc=-338.41   BIC=-329.1
```

```
coeftest(model.111ML)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error  z value  Pr(>|z|)
## ar1  0.50331    0.11182   4.5012 6.758e-06 ***
## ma1 -0.78487    0.07365 -10.6568 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.111CSS = Arima(anomalies.ts, order=c(1,1,1), method='CSS')
model.111CSS
```

```
## Series: anomalies.ts
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1      ma1
##       0.4833  -0.7721
## s.e.  0.1102   0.0757
##
## sigma^2 = 0.008082:  log likelihood = 171.8
```

```
coeftest(model.111CSS)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error  z value  Pr(>|z|)
## ar1  0.483274   0.110230   4.3842 1.164e-05 ***
## ma1 -0.772055   0.075674 -10.2024 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ARIMA(2,1,1) & ARIMA(2,1,2) (Neighbor Models)

Since both AR1 and MA1 coefficients of the ARIMA(1,1,1) model are highly significant, neighboring models ARIMA(2,1,1) and ARIMA(2,1,2) are added to the list of possible models to be fitted. ARIMA(2,1,1) produces a significant AR2 coefficient with a very low p-value but the AR1 and MA1 coefficients are insignificant. The ARIMA(2,1,2) model produces coefficients for AR1, AR2, MA1, and MA2 which are all insignificant, with AR2 being borderline with a p-value of 0.07585 for the ML estimator and 0.0558 for the CSS estimator. Due to the borderline p-value for AR2, the CSS-ML estimator is also used to see if it is possible to obtain a significant coefficient for AR2. However, it produces a p-value of 0.07516, mirroring the ML estimator. The CSS and ML estimators produce roughly the same p-values for all five models tested here. Neither of the neighboring models tested seem to improve upon the ARIMA(1,1,1) model due to the presence of insignificant coefficients.

```
model.211ML = Arima(anomalies.ts, order=c(2,1,1), method='ML')
model.211ML
```

```
## Series: anomalies.ts
## ARIMA(2,1,1)
##
## Coefficients:
##          ar1      ar2      ma1
##       0.1488  -0.3956  -0.2738
## s.e.  0.1944   0.0774   0.2176
##
## sigma^2 = 0.007381:  log likelihood = 180.45
## AIC=-352.9   AICc=-352.66   BIC=-340.28
```

```
coeftest(model.211ML)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1  0.148792   0.194411  0.7653    0.4441
## ar2 -0.395596   0.077386 -5.1120 3.188e-07 ***
## ma1 -0.273818   0.217564 -1.2586    0.2082
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.211CSS = Arima(anomalies.ts, order=c(2,1,1), method='CSS')
model.211CSS
```

```
## Series: anomalies.ts
## ARIMA(2,1,1)
##
## Coefficients:
##          ar1      ar2      ma1
##       0.1475  -0.3994  -0.2739
## s.e.  0.1922   0.0775   0.2146
##
## sigma^2 = 0.007358:  log likelihood = 179.91
```

```
coeftest(model.211CSS)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1  0.147460   0.192159  0.7674   0.4429
## ar2 -0.399352   0.077451 -5.1562 2.52e-07 ***
## ma1 -0.273855   0.214609 -1.2761   0.2019
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.212ML = Arima(anomalies.ts, order=c(2,1,2), method='ML')
model.212ML
```

```
## Series: anomalies.ts
## ARIMA(2,1,2)
##
## Coefficients:
##          ar1      ar2      ma1      ma2
##       0.1293  -0.2959  -0.2577  -0.1207
## s.e.  0.1741   0.1666   0.1790   0.1702
##
## sigma^2 = 0.007403:  log likelihood = 180.69
## AIC=-351.38   AICc=-351.02   BIC=-335.61
```

```
coeftest(model.212ML)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.12926    0.17412  0.7424  0.45786
## ar2 -0.29585    0.16665 -1.7753  0.07585 .
## ma1 -0.25768    0.17896 -1.4398  0.14991
## ma2 -0.12067    0.17016 -0.7092  0.47823
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.212CSS = Arima(anomalies.ts, order=c(2,1,2), method='CSS')
model.212CSS
```

```
## Series: anomalies.ts
## ARIMA(2,1,2)
##
## Coefficients:
##          ar1      ar2      ma1      ma2
##       0.1271  -0.3086  -0.2561  -0.1115
## s.e.  0.1738   0.1614   0.1792   0.1664
##
## sigma^2 = 0.007383:  log likelihood = 180.13
```

```
coeftest(model.212CSS)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.12712    0.17380  0.7314   0.4645
## ar2 -0.30861    0.16136 -1.9126   0.0558 .
## ma1 -0.25607    0.17918 -1.4291   0.1530
## ma2 -0.11150    0.16641 -0.6700   0.5028
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.212CSSML = Arima(anomalies.ts, order=c(2,1,2), method='CSS-ML')
model.212CSSML
```

```
## Series: anomalies.ts
## ARIMA(2,1,2)
##
## Coefficients:
##          ar1      ar2      ma1      ma2
##       0.1305  -0.2963  -0.2589  -0.1200
## s.e.  0.1742   0.1665   0.1791   0.1701
##
## sigma^2 = 0.007403:  log likelihood = 180.69
## AIC=-351.38   AICc=-351.02   BIC=-335.61
```

```
coeftest(model.212CSSML)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.13050    0.17423  0.7490  0.45386
## ar2 -0.29632    0.16652 -1.7795  0.07516 .
## ma1 -0.25890    0.17914 -1.4452  0.14841
## ma2 -0.12000    0.17013 -0.7053  0.48061
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ARIMA(1,1,2)

The ARIMA(1,1,2) model is fitted and the coefficients are found using the coeftest() function using the ML estimator. This produces a highly significant coefficient for only the MA2 component, while AR1 and MA1 components are both insignificant. Using the CSS estimator produces roughly similar p-values for each coefficient.

```
model.112ML = Arima(anomalies.ts, order=c(1,1,2), method='ML')
model.112ML
```

```
## Series: anomalies.ts
## ARIMA(1,1,2)
##
## Coefficients:
##          ar1      ma1      ma2
##       0.0700  -0.2077  -0.3670
## s.e.  0.1559   0.1362   0.0674
##
## sigma^2 = 0.007484:  log likelihood = 179.26
## AIC=-350.53   AICc=-350.29   BIC=-337.91
```

```
coeftest(model.112ML)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1  0.070015   0.155948  0.4490    0.6535
## ma1 -0.207702   0.136249 -1.5244    0.1274
## ma2 -0.366988   0.067420 -5.4433 5.231e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.112CSS = Arima(anomalies.ts, order=c(1,1,2), method='CSS')
model.112CSS
```

```
## Series: anomalies.ts
## ARIMA(1,1,2)
##
## Coefficients:
##          ar1      ma1      ma2
##       0.0636  -0.2027  -0.3702
## s.e.  0.1539   0.1344   0.0675
##
## sigma^2 = 0.007471:  log likelihood = 179.1
```

```
coeftest(model.112CSS)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1  0.063647   0.153936  0.4135   0.6793
## ma1 -0.202697   0.134390 -1.5083   0.1315
## ma2 -0.370220   0.067526 -5.4826 4.19e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ARIMA(0,1,3)

The ARIMA(0,1,3) model is fitted and the coefficients are found using the coeftest() function using the ML estimator. This produces a highly significant coefficient for only the MA2 component, while MA1 and MA3 components are both insignificant. Using the CSS estimator produces roughly similar p-values for each coefficient.

```
model.013ML = Arima(anomalies.ts, order=c(0,1,3), method='ML')
model.013ML
```

```
## Series: anomalies.ts
## ARIMA(0,1,3)
##
## Coefficients:
##           ma1      ma2      ma3
##       -0.1213  -0.3771  -0.0514
## s.e.   0.0869   0.0617   0.0831
##
## sigma^2 = 0.007476:  log likelihood = 179.36
## AIC=-350.71   AICc=-350.47   BIC=-338.1
```

```
coeftest(model.013ML)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.121311   0.086910 -1.3958    0.1628
## ma2 -0.377091   0.061659 -6.1158 9.609e-10 ***
## ma3 -0.051400   0.083107 -0.6185    0.5363
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.013CSS = Arima(anomalies.ts, order=c(0,1,3), method='CSS')
model.013CSS
```

```
## Series: anomalies.ts
## ARIMA(0,1,3)
##
## Coefficients:
##           ma1      ma2      ma3
##       -0.1210  -0.3803  -0.0534
## s.e.   0.0871   0.0617   0.0837
##
## sigma^2 = 0.007479:  log likelihood = 179.51
```

```
coeftest(model.013CSS)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.120979   0.087093 -1.3891    0.1648
## ma2 -0.380308   0.061678 -6.1660 7.002e-10 ***
## ma3 -0.053368   0.083661 -0.6379    0.5235
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ARIMA(2,1,0)

The ARIMA(2,1,0) model is fitted and the coefficients are found using the coeftest() function using the ML estimator. This produces a highly significant coefficient for only the AR2 component, while AR1 component is insignificant. Using the CSS estimator produces roughly similar p-values for each coefficient.

```
model.210ML = Arima(anomalies.ts, order=c(2,1,0), method='ML')
model.210ML
```

```
## Series: anomalies.ts
## ARIMA(2,1,0)
##
## Coefficients:
##           ar1      ar2
##       -0.0776  -0.4111
## s.e.   0.0708   0.0704
##
## sigma^2 = 0.007408:  log likelihood = 179.64
## AIC=-353.28   AICc=-353.14   BIC=-343.82
```

```
coeftest(model.210ML)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1 -0.077561   0.070832 -1.0950    0.2735
## ar2 -0.411121   0.070413 -5.8387 5.261e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.210CSS = Arima(anomalies.ts, order=c(2,1,0), method='CSS')
model.210CSS
```

```
## Series: anomalies.ts
## ARIMA(2,1,0)
##
## Coefficients:
##           ar1      ar2
##       -0.0787  -0.4148
## s.e.   0.0708   0.0708
##
## sigma^2 = 0.007385:  log likelihood = 179.09
```

```
coeftest(model.210CSS)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1 -0.078685   0.070804 -1.1113    0.2664
## ar2 -0.414783   0.070755 -5.8622 4.567e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ARIMA(2,1,4)

The ARIMA(2,1,4) model is fitted and the coefficients are found using the coeftest() function using the ML estimator. This produces no significant coefficients for any of the components. Using the CSS estimator produces different, but highly insignificant p-values for each coefficient. Due to the fact that the p-values are high, CSS-ML is also explored as an alternative estimator since there is a difference between CSS and ML results and it reflects the significance values obtained from CSS.

```
model.214ML = Arima(anomalies.ts, order=c(2,1,4), method='ML')
model.214ML
```

```
## Series: anomalies.ts
## ARIMA(2,1,4)
##
## Coefficients:
##           ar1      ar2     ma1      ma2      ma3     ma4
##       -0.6372  -0.2009  0.5206  -0.3100  -0.3437  0.0448
## s.e.   0.8385   0.5328  0.8369   0.4568   0.4027  0.2921
##
## sigma^2 = 0.007375:  log likelihood = 182
## AIC=-350   AICc=-349.32   BIC=-327.92
```

```
coeftest(model.214ML)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.637172   0.838528 -0.7599   0.4473
## ar2 -0.200904   0.532772 -0.3771   0.7061
## ma1  0.520573   0.836915  0.6220   0.5339
## ma2 -0.309970   0.456829 -0.6785   0.4974
## ma3 -0.343748   0.402713 -0.8536   0.3933
## ma4  0.044811   0.292062  0.1534   0.8781
```

```
model.214CSS = Arima(anomalies.ts, order=c(2,1,4), method='CSS')
model.214CSS
```

```
## Series: anomalies.ts
## ARIMA(2,1,4)
##
## Coefficients:
##           ar1      ar2     ma1      ma2      ma3     ma4
##       -0.5170  -0.1418  0.4002  -0.3611  -0.2883  0.0843
## s.e.   0.4587   0.3140  0.4544   0.2756   0.2223  0.1649
##
## sigma^2 = 0.007372:  log likelihood = 181.3
```

```
coeftest(model.214CSS)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.517027   0.458695 -1.1272   0.2597
## ar2 -0.141767   0.314000 -0.4515   0.6516
## ma1  0.400227   0.454427  0.8807   0.3785
## ma2 -0.361129   0.275577 -1.3104   0.1900
## ma3 -0.288300   0.222279 -1.2970   0.1946
## ma4  0.084291   0.164925  0.5111   0.6093
```

```
model.214CSSML = Arima(anomalies.ts, order=c(2,1,4), method='CSS-ML')
model.214CSSML
```

```
## Series: anomalies.ts
## ARIMA(2,1,4)
##
## Coefficients:
##           ar1      ar2     ma1      ma2      ma3     ma4
##       -0.6361  -0.2002  0.5196  -0.3102  -0.3428  0.0454
## s.e.   0.8313   0.5286  0.8298   0.4538   0.3988  0.2894
##
## sigma^2 = 0.007375:  log likelihood = 182
## AIC=-350   AICc=-349.32   BIC=-327.92
```

```
coeftest(model.214CSSML)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.636106   0.831280 -0.7652   0.4441
## ar2 -0.200243   0.528641 -0.3788   0.7048
## ma1  0.519642   0.829833  0.6262   0.5312
## ma2 -0.310154   0.453821 -0.6834   0.4943
## ma3 -0.342797   0.398842 -0.8595   0.3901
## ma4  0.045398   0.289425  0.1569   0.8754
```

AIC and BIC Analysis

The AIC() and BIC() functions are used to determine the AIC and BIC values for each model. The sort.score() function developed by Yong Kai Wong is utilized to sort the AIC and BIC scores in ascending order in order to aid selection of the model with the least AIC and BIC (Canvas, 2024). Model ARIMA(2,1,0) has the lowest scores for both AIC and BIC.

```
sort.score <- function(x, score = c("bic", "aic")){
  if (score == "aic"){
    x[with(x, order(AIC)),]
  } else if (score == "bic") {
    x[with(x, order(BIC)),]
  } else {
    warning('score = "x" only accepts valid arguments ("aic","bic")')
  }
}

sort.score(AIC(model.012ML, model.111ML, model.211ML, model.212ML,
               model.112ML, model.013ML, model.210ML, model.214ML),
           score = "aic")
```

```
##             df       AIC
## model.210ML  3 -353.2793
## model.211ML  4 -352.8980
## model.012ML  3 -352.3256
## model.212ML  5 -351.3795
## model.013ML  4 -350.7104
## model.112ML  4 -350.5250
## model.214ML  7 -349.9966
## model.111ML  3 -338.5562
```

```
sort.score(BIC(model.012ML, model.111ML, model.211ML, model.212ML,
               model.112ML, model.013ML, model.210ML, model.214ML),
           score = "bic")
```

```
##             df       BIC
## model.210ML  3 -343.8194
## model.012ML  3 -342.8657
## model.211ML  4 -340.2848
## model.013ML  4 -338.0972
## model.112ML  4 -337.9119
## model.212ML  5 -335.6131
## model.111ML  3 -329.0963
## model.214ML  7 -327.9236
```

Error Metric Comparison

The accuracy() function is given each fitted model to obtain the Mean Error (ME), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Percentage Error (MPE), Mean Absolute Percentage Error (MAPE), Mean Absolute Scaled Error (MASE), and Autocorrelation of errors at lag 1 (ACF1).

```
Smodel.012A = forecast::accuracy(model.012ML)[1:7]
Smodel.111A = forecast::accuracy(model.111ML)[1:7]
Smodel.211A = forecast::accuracy(model.211ML)[1:7]
Smodel.212A = forecast::accuracy(model.212ML)[1:7]
Smodel.112A = forecast::accuracy(model.112ML)[1:7]
Smodel.013A = forecast::accuracy(model.013ML)[1:7]
Smodel.210A = forecast::accuracy(model.210ML)[1:7]
Smodel.214A = forecast::accuracy(model.214ML)[1:7]

df.Smodels = data.frame(
                rbind(Smodel.012A, Smodel.111A, Smodel.211A, Smodel.212A,
                      Smodel.112A, Smodel.013A, Smodel.210A, Smodel.214A)
                )
colnames(df.Smodels) = c("ME", "RMSE", "MAE", "MPE", "MAPE", "MASE", "ACF1")
rownames(df.Smodels) = c("ARIMA(0,1,2)", "ARIMA(1,1,1)", "ARIMA(2,1,1)", "ARIMA(2,1,2)",
                         "ARIMA(1,1,2)", "ARIMA(0,1,3)", "ARIMA(2,1,0)", "ARIMA(2,1,4)")

df.Smodels
```

```
##                       ME       RMSE        MAE MPE MAPE      MASE         ACF1
## ARIMA(0,1,2) 0.010167289 0.08556209 0.06451230 NaN  Inf 0.8978783  0.003405844
## ARIMA(1,1,1) 0.010840407 0.08906817 0.06870391 NaN  Inf 0.9562169  0.091463456
## ARIMA(2,1,1) 0.008440168 0.08491914 0.06324615 NaN  Inf 0.8802561 -0.002378681
## ARIMA(2,1,2) 0.009105764 0.08479663 0.06348200 NaN  Inf 0.8835387 -0.006510229
## ARIMA(1,1,2) 0.010367057 0.08551151 0.06457613 NaN  Inf 0.8987668 -0.009921047
## ARIMA(0,1,3) 0.010497405 0.08546415 0.06461128 NaN  Inf 0.8992559 -0.022222518
## ARIMA(2,1,0) 0.007474399 0.08532577 0.06277078 Inf  Inf 0.8736400 -0.041777176
## ARIMA(2,1,4) 0.009640820 0.08413000 0.06265889 NaN  Inf 0.8720827 -0.011677925
```

The resulting values show that MPE and MAPE have either infinity or NaN instead of numeric values. This is due to the nature of the MPE equation which which has the actual data point value of the time series y_i in the denominator (MPE - Mean Percentage Error — Permetrics 2.0.0 Documentation, n.d.). The MAPE equation suffers from the same problem (MAPE - Mean Absolute Percentage Error — Permetrics 2.0.0 Documentation, n.d.). Both equations are shown below. Since the anomalies time series contains several instances of the value 0, this leads to an undefined result for both MPE and MASE.

$$\text{MPE}(y, \hat{y}) = \frac{100\%}{N} \sum_{i=0}^{N-1} \frac{y_i - \hat{y}_i}{y_i}$$

$$\text{MAPE}(y, \hat{y}) = \frac{100\%}{N} \sum_{i=0}^{N-1} \frac{|y_i - \hat{y}_i|}{|y_i|}$$

Therefore, MAPE is replaced by Symmetrical Mean Absolute Percentage Error (SMAPE), which calculates a similar metric but is instead weighted with the sum of the absolute values of the actual value y_i and predicted value $\hat{y}_i$ as shown below (SMAPE - Symmetric Mean Absolute Percentage Error — Permetrics 2.0.0 Documentation, n.d.):

$$\text{SMAPE}(y, \hat{y}) = \frac{100\%}{N} \sum_{i=0}^{N-1} \frac{2 * |y_i - \hat{y}_i|}{|y| + |\hat{y}|}$$

The SMAPE values are calculated below for each model.

```
smapes = c(smape(anomalies.ts, fitted(model.012ML)),
           smape(anomalies.ts, fitted(model.111ML)),
           smape(anomalies.ts, fitted(model.211ML)),
           smape(anomalies.ts, fitted(model.212ML)),
           smape(anomalies.ts, fitted(model.112ML)),
           smape(anomalies.ts, fitted(model.013ML)),
           smape(anomalies.ts, fitted(model.210ML)),
           smape(anomalies.ts, fitted(model.214ML))
         )
df.Smodels = cbind(df.Smodels, data.frame(SMAPE = smapes))
setDT(df.Smodels, keep.rownames = "Models")[]
```

```
##          Models          ME       RMSE        MAE MPE MAPE      MASE
## 1: ARIMA(0,1,2) 0.010167289 0.08556209 0.06451230 NaN  Inf 0.8978783
## 2: ARIMA(1,1,1) 0.010840407 0.08906817 0.06870391 NaN  Inf 0.9562169
## 3: ARIMA(2,1,1) 0.008440168 0.08491914 0.06324615 NaN  Inf 0.8802561
## 4: ARIMA(2,1,2) 0.009105764 0.08479663 0.06348200 NaN  Inf 0.8835387
## 5: ARIMA(1,1,2) 0.010367057 0.08551151 0.06457613 NaN  Inf 0.8987668
## 6: ARIMA(0,1,3) 0.010497405 0.08546415 0.06461128 NaN  Inf 0.8992559
## 7: ARIMA(2,1,0) 0.007474399 0.08532577 0.06277078 Inf  Inf 0.8736400
## 8: ARIMA(2,1,4) 0.009640820 0.08413000 0.06265889 NaN  Inf 0.8720827
##            ACF1     SMAPE
## 1:  0.003405844 0.6998716
## 2:  0.091463456 0.7385694
## 3: -0.002378681 0.6837052
## 4: -0.006510229 0.6915555
## 5: -0.009921047 0.6928430
## 6: -0.022222518 0.6873062
## 7: -0.041777176 0.6834740
## 8: -0.011677925 0.6914021
```

The error metrics are individually sorted in ascending order and just the top four models chosen to determine if there are any models which repeatedly show up in the top four. MPE is omitted due to its undefined values. ARIMA(2,1,0) is the top model for the ME, ACF1, and SMAPE error metrics. However for those metrics, ARIMA(2,1,4) still appears in the top 4 models. ARIMA(2,1,4) is the top model for RMSE, MAE, and MASE error metrics. However for those metrics, ARIMA(2,1,0) still appears in the top 4 models as well.

Despite the good performance of ARIMA(2,1,4) on these error metrics, it appears second-to-last on the AIC ranking and last in the BIC ranking. Therefore, ARIMA(2,1,0) is considered to be the best model from all the models fitted after considering all measures (AIC, BIC, and error metrics).

```
df.Smodels[order(df.Smodels$ME),][1:4,c(1,2)]
```

```
##          Models          ME
## 1: ARIMA(2,1,0) 0.007474399
## 2: ARIMA(2,1,1) 0.008440168
## 3: ARIMA(2,1,2) 0.009105764
## 4: ARIMA(2,1,4) 0.009640820
```

```
df.Smodels[order(df.Smodels$RMSE),][1:4,c(1,3)]
```

```
##          Models       RMSE
## 1: ARIMA(2,1,4) 0.08413000
## 2: ARIMA(2,1,2) 0.08479663
## 3: ARIMA(2,1,1) 0.08491914
## 4: ARIMA(2,1,0) 0.08532577
```

```
df.Smodels[order(df.Smodels$MAE),][1:4,c(1,4)]
```

```
##          Models        MAE
## 1: ARIMA(2,1,4) 0.06265889
## 2: ARIMA(2,1,0) 0.06277078
## 3: ARIMA(2,1,1) 0.06324615
## 4: ARIMA(2,1,2) 0.06348200
```

```
df.Smodels[order(df.Smodels$MASE),][1:4,c(1,7)]
```

```
##          Models      MASE
## 1: ARIMA(2,1,4) 0.8720827
## 2: ARIMA(2,1,0) 0.8736400
## 3: ARIMA(2,1,1) 0.8802561
## 4: ARIMA(2,1,2) 0.8835387
```

```
df.Smodels[order(df.Smodels$ACF1),][1:4,c(1,8)]
```

```
##          Models         ACF1
## 1: ARIMA(2,1,0) -0.041777176
## 2: ARIMA(0,1,3) -0.022222518
## 3: ARIMA(2,1,4) -0.011677925
## 4: ARIMA(1,1,2) -0.009921047
```

```
df.Smodels[order(df.Smodels$ACF1),][1:4,c(1,9)]
```

```
##          Models     SMAPE
## 1: ARIMA(2,1,0) 0.6834740
## 2: ARIMA(0,1,3) 0.6873062
## 3: ARIMA(2,1,4) 0.6914021
## 4: ARIMA(1,1,2) 0.6928430
```

Conclusion

During the investigation, a total of eight models were fitted and tested for the time series data describing yearly Global Land Temperature Anomalies in Degrees Celsius against the base period 1901-2000. Visual analysis of the series confirmed a possible combination of AR and MA components due to the trend and fluctuations seen in the series. ADF testing confirmed non-stationarity in the series and Shapiro-Wilk testing confirmed non-normality in the data. This was further supported by the ACF and PACF plots which demonstrated autocorrelations at multiple lags. Applying the Box-Cox transformation to the series provided a lambda value of 1, indicating no transformation, and was therefore abandoned. Log transforming the series provided visual improvements but further testing confirmed that the log-transformed series was still not stationary and had non-normal distribution. The first difference of the anomalies series, however, produced a stationary series with normal distribution and was chosen as the suitable transformation for the series. The ACF and PACF plots of the differenced series, in addition to the EACF table and neighboring model selection, provided a list of eight total models for fitting. These models were fitted and their parameter coefficients were discussed. Afterwards, obtaining the AIC and BIC scores, along with the ME, RMSE, MAE, MASE, ACF1, and SMAPE error metrics shortlisted the two best models to be ARIMA(2,1,0) and ARIMA(2,1,4). However, the superior performance of ARIMA(2,1,0) when considering AIC and BIC scores and its overall satisfactory performance with the other error metrics justified its selection as the most suitable model to describe the yearly Global Land Temperature Anomalies data.

References

- Canvas (2024) sort.score.R. Available at: https://rmit.instructure.com/courses/124176/files/36179115?module_item_id=5935464 (https://rmit.instructure.com/courses/124176/files/36179115?module_item_id=5935464) (accessed 1st May 2024).
- MPE - Mean Percentage Error — Permetrics 2.0.0 documentation. (n.d.). Permetrics.readthedocs.io. Retrieved May 5, 2024, from https://permetrics.readthedocs.io/en/latest/pages/regression/MPE.html (https://permetrics.readthedocs.io/en/latest/pages/regression/MPE.html)
- MAPE - Mean Absolute Percentage Error — Permetrics 2.0.0 documentation. (n.d.). Permetrics.readthedocs.io. Retrieved May 5, 2024, from https://permetrics.readthedocs.io/en/latest/pages/regression/MAPE.html (https://permetrics.readthedocs.io/en/latest/pages/regression/MAPE.html)
- SMAPE - Symmetric Mean Absolute Percentage Error — Permetrics 2.0.0 documentation. (n.d.). Permetrics.readthedocs.io. https://permetrics.readthedocs.io/en/latest/pages/regression/SMAPE.html (https://permetrics.readthedocs.io/en/latest/pages/regression/SMAPE.html)