Archit Sharma (s3755732)

Shivani (s3763179)

Forecasting for M3C Series

MATH1307 Final Project - Competitive

Table of Contents

[Introduction 2](#_Toc23097461)

[Objective 2](#_Toc23097462)

[Methodology 2](#_Toc23097463)

[Data Transformation 2](#_Toc23097464)

[Comparing Information Criteria on Train Set 3](#_Toc23097465)

[MASE Comparison for Train Set 3](#_Toc23097466)

[Model Fitting on Test Set 3](#_Toc23097467)

[Residual Diagnostics 3](#_Toc23097468)

[Forecasting 3](#_Toc23097469)

[Results 3](#_Toc23097470)

[IC and Best-Fit Frequency 3](#_Toc23097471)

[For Yearly Data 4](#_Toc23097472)

[For Quarterly Data 4](#_Toc23097473)

[For Monthly Data 5](#_Toc23097474)

[Computing Mean MASE Value and MASE Rank 5](#_Toc23097475)

[Fitting Model on Test Set 6](#_Toc23097476)

[Residual Analysis for Models 6](#_Toc23097477)

[Forecasting for 6, 8 and 18 Periods: 7](#_Toc23097478)

[Conclusion 7](#_Toc23097479)

[Reference 8](#_Toc23097480)

[Appendix 9](#_Toc23097481)

# Introduction

The M3C dataset (International Institute of Forecasters, 2019) consists of 997 time series which is segregated under 3 time periods i.e. yearly, quarterly and monthly and 5 primary categories which are micro-economic, macro-economic, industrial, financial, and demographic. Each of the series have different lengths and time range.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Period | Demographic | Finance | Industry | Macro | Micro | Other | TOTAL |
| Monthly | 48 | 62 | 52 | 86 | 84 | 0 | **332** |
| Quarterly | 7 | 34 | 54 | 167 | 70 | 0 | **332** |
| Yearly | 105 | 36 | 38 | 83 | 60 | 11 | **333** |

Table 1 - About the dataset

Phillips-Perron Unit Root Test, executed for the 3 periods, confirmed that only 37% of series were stationary in terms of statistical mean and variance. 63% of the time series pertaining to monthly period was observed to be stationary. Majority of the time series belonging to quarterly and yearly data was found to be non-stationary.

|  |  |  |
| --- | --- | --- |
| Period | No. of Stationary Series | % Stationary |
| Monthly | 208 | 63% |
| Quarterly | 124 | 37% |
| Yearly | 36 | 11% |
| TOTAL | **368** | **37%** |

Table 2 - Checking stationarity in the dataset

# Objective

The primary purpose of this analysis was to find the best fitting model of the given 997 time series by estimating and comparing the accuracy of different state-space models of forecasting for each time series in the dataset.

# Methodology

Each series in the dataset was bifurcated into training set (95%) and test set (5%). The following steps were performed thereafter:

## Data Transformation

Since, majority of the time series were found to be non-stationary, box-cox transformation was applied on the time series to handle the changing variance over time. The value of lambda was calculated using the BoxCox.lambda() function, and used in the following formula of transformation:

where, Z = Transformed value and Y = Original data

The new datasets were checked for any zero or negative values, post transformation.

## Comparing Information Criteria on Train Set

State-space (Error-Trend-Seasonality) models were used for the modelling purpose. The ETS models were fit to each series considering the presence of trend and seasonality in the training sets, whose information criteria values were compared. Subsequently, the model with highest model-fit frequency, along with the least AIC-BIC-HQIC values, was considered for further analysis. This task was iterated for damped and undamped Trend component using GoFVals() function.

## MASE Comparison for Train Set

The MASE values and rank were computed for the shortlisted models using MASEvalues() function.

For each of the 3 type of time series, the model with the least MASE value and MASE rank was selected as the best fit.

## Model Fitting on Test Set

The training set, containing 95% of the data, was used to forecast for the next 5% of time period using the best-fit model realized in the previous step. The quality of forecasts for each series was then checked against the 5% test set. Average of MASE values of model-fitting on train and test set was calculated for each of the 997 time series.

## Residual Diagnostics

For the chosen models, the total number of non-normal standardized residuals was computed using Shapiro-Wilk’s normality test. Similarly, the total number of standardized residuals with significant serial correlation was calculated using Ljung-Box test.

## Forecasting

Best fitted models were used to calculate the MASE values over the forecasts for 6, 8 and 18 times-ahead forecasts for yearly, quarterly and monthly periods, respectively. The forecasts obtained were converted to their non-transformed version based on the equation below:

where, Y = actual forecast and Z = transformed forecasts

# Results

Post the box-cox transformation, it was observed that the transformed datasets consisted of only positive values. Hence, the data was ready for ETS model-fitting.

## IC and Best-Fit Frequency

The models having an active seasonal component (xxA or xxM) were found to be forbidden for time series pertaining to the yearly period. Hence, in the yearly time series, non-seasonal state space models were considered as it did not have seasonality whereas for monthly and quarterly series, both seasonal and non-seasonal innovation state space models were tested. Also, the MASE value was compared with the application of damped trend models for each time series.

After taking AIC, BIC and HQIC into consideration for each period, models with the highest model-fit frequency were considered for further evaluation. The best model was selected based on the mean MASE value and MASE rank.

The following display show the fitted models along with their AIC, BIC, HQIC and mean MASE value for each period:

### For Yearly Data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trend Type | Fitted Models | AIC | BIC | HQIC | Best Fits | MASE |
| Undamped | AAN | 156 | 162 | 155 | 104 | **0.765** |
| MAN | 157 | 162 | 156 | 96 | 0.769 |
| MMN | 158 | 164 | 158 | 74 | 0.797 |
| ANN | 164 | 168 | 163 | 41 | 0.964 |
| MNN | 166 | 170 | 165 | 18 | 0.975 |
| Damped | AAdN | 157 | 164 | 156 | 171 | **0.745** |
| MMdN | 158 | 165 | 157 | 162 | 0.753 |

Table 3 - Comparison of IC and Best-Fit Frequency - Yearly Series

As per the table 3, the undamped trend models ETS(A,A,N) gave the lowest AIC, BIC and HQIC values with the highest best-fit frequency. Amongst the models applied with a damped trend, the ETS(A,Ad,N) gave the lowest IC values along with the highest best-fit frequency.

### For Quarterly Data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trend Type | Fitted Models | AIC | BIC | HQIC | Best Fits | MASE |
| Undamped | AAA | 318 | 335 | 322 | 66 | **0.431** |
| MAA | 320 | 336 | 324 | 57 | 0.433 |
| MAM | 322 | 339 | 326 | 37 | 0.445 |
| MMM | 325 | 341 | 328 | 48 | 0.454 |
| MNA | 335 | 348 | 337 | 7 | 0.493 |
| ANA | 333 | 346 | 335 | 17 | 0.498 |
| MNM | 337 | 349 | 339 | 9 | 0.504 |
| AAN | 332 | 341 | 333 | 26 | 0.693 |
| MAN | 334 | 343 | 334 | 25 | 0.694 |
| MMN | 334 | 343 | 335 | 30 | 0.697 |
| ANN | 340 | 346 | 339 | 6 | 0.763 |
| MNN | 342 | 347 | 341 | 4 | 0.766 |
| Damped | AAdA | 318 | 336 | 322 | 68 | **0.42** |
| MMdM | 320 | 338 | 324 | 77 | 0.426 |
| MAdA | 321 | 339 | 325 | 49 | 0.427 |
| MAdM | 321 | 339 | 325 | 40 | 0.428 |
| AAdN | 333 | 344 | 334 | 25 | 0.684 |
| MAdN | 334 | 345 | 336 | 33 | 0.686 |
| MMdN | 334 | 345 | 336 | 40 | 0.686 |

Table 4 - Comparison of IC and Best-Fit Frequency - Quarterly Series

As per table 4, the undamped trend models ETS(A,A,A) had the lowest AIC, BIC and HQIC values with the highest best-fit frequency. Amongst the models applied with a damped trend, the ETS(A,Ad,A) gave the lowest IC values along with the highest best-fit frequency.

### For Monthly Data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trend Type | Fitted Models | AIC | BIC | HQIC | Best Fits | MASE |
| Undamped | AAA | 436 | 483 | 452 | 76 | 0.42 |
| MAA | 440 | 486 | 455 | 64 | 0.423 |
| MAM | 452 | 498 | 468 | 27 | 0.44 |
| MMM | 461 | 507 | 476 | 37 | 0.45 |
| ANA | 476 | 517 | 489 | 24 | 0.455 |
| MNA | 482 | 523 | 495 | 15 | 0.456 |
| MNM | 504 | 545 | 518 | 3 | 0.491 |
| AAN | 469 | 483 | 472 | 29 | 0.607 |
| MAN | 470 | 484 | 473 | 27 | 0.613 |
| MMN | 471 | 485 | 473 | 18 | 0.621 |
| ANN | 478 | 486 | 478 | 6 | 0.624 |
| MNN | 481 | 489 | 481 | 6 | 0.624 |
| Damped | AAdA | 435 | 484 | 452 | 76 | 0.416 |
| MAdA | 437 | 486 | 454 | 68 | 0.416 |
| MAdM | 436 | 485 | 453 | 58 | 0.418 |
| MMdM | 436 | 486 | 453 | 59 | 0.418 |
| AAdN | 470 | 486 | 473 | 24 | 0.602 |
| MAdN | 471 | 487 | 474 | 19 | 0.603 |
| MMdN | 471 | 487 | 474 | 28 | 0.605 |

Table 5 - Comparison of IC and Best-Fit Frequency - Monthly Series

While comparing IC’s of the undamped trend models of monthly series, ETS(A,A,A) model gave the minimum IC values. Additionally, it had the maximum frequency of best-fit compared to other models. For damped trend model, ETS(A,Ad,A) model was shortlisted for further analysis as it had the least AIC, BIC and HQIC values along with the highest best-fit frequency.

## Computing Mean MASE Value and MASE Rank

The following table (table 6) compares the mean MASE value and MASE rank for the top damped and undamped trend models shortlisted in the previous step:

|  |  |  |  |
| --- | --- | --- | --- |
| Period | Model | Mean MASE | MASE Rank |
| Yearly | AAdN | 0.744544 | 331.5435 |
| AAN | 0.764695 | 633.1351 |
| Quarterly | AAdA | 0.420476 | 1060.988 |
| AAA | 0.43113 | 1682.91 |
| Monthly | AAdA | 0.416262 | 1051.753 |
| AAA | 0.420008 | 1715.527 |

Table 6 - Comparing MASE Value and MASE Rank on Training Data

The best fitted model for yearly time series was found to be **ETS(A,Ad,N)** with a MASE value of **0.744544**. Similarly, he best fitted model for quarterly and monthly series came out to be **ETS(A,Ad,A)** with the MASE value of **0.420476** and **0.416262**, respectively. Evidently, the amped trend models were found to be promising for all the 3 time periods.

## Fitting Model on Test Set

The best-fit models, chosen in the previous task, were used to forecast ahead of training set. The accuracy of models was assessed by comparing the average MASE values of training and test sets, as given in *table 7* below:

|  |  |  |  |
| --- | --- | --- | --- |
| Period | Model | MASE - Train Set | MASE – Test Set |
| Yearly | A,Ad,N | 0.7445443 | 1.165745 |
| Quarterly | A,Ad,A | 0.4204762 | 1.112868 |
| Monthly | A,Ad,A | 0.4162618 | 1.327163 |

Table 7 - MASE Values of Train Set vs. Test Set

## Residual Analysis for Models

The standardized residuals were assessed on the best-fit models selected for each period to check if the residual supported the model assumption. Non-normal distribution of standardized residuals was examined using Shapiro Wilk Normality Test. Additionally, Ljung-Box Test was applied to check for any significant serial correlation in the standardized residuals.

The following *table 8* gives a summarized view of the residual and serial correlation analysis for the three periods:

|  |  |  |  |
| --- | --- | --- | --- |
| Period | Sample Size of Period | Non-Normally Distributed Standardized Residuals | Serially Correlated Standadized Residuals |
| Yearly | 333 | 69 | 23 |
| Quarterly | 332 | 102 | 40 |
| Monthly | 332 | 185 | 23 |
| Total | - | 296 | 146 |
| % Total | - | 29.7% | 14.6% |

Table 8 - Residual Diagnostics for Selected Models

After applying best-fitted model to the yearly series, there were 69 time series in which the standardized residuals were not normally distributed, and 23 series had correlated residuals.

The residual analysis for quarterly series showed that out of 332 time series, 102 residuals were not normally distributed, along with 40 residuals with had significant serially correlated residuals.

Similarly, for monthly series, 185 series out of 332 series were not normally distributed and 146 series had correlated residuals.

## Forecasting for 6, 8 and 18 Periods:

The transformed forecasts were further obtained from the 100% values (train+test) in the data, for the 6, 8 and 18 time periods for yearly, quarterly and monthly series respectively. These forecasts were then converted to their corresponding values based on the reverse transformation mentioned in the methodology section of the report.

# Conclusion

The best-fit ETS models, chosen for forecasting for the respective groups of time series are as follows:

* Monthly Time Series – Additive Error, Additive Damped Trend, Additive Seasonality (A,Ad,A)
* Quarterly Time Series – Additive Error, Additive Damped Trend, Additive Seasonality (A,Ad,A)
* Yearly Time Series – Additive Error, Additive Damped Trend, Additive Seasonality (A,Ad,A)

The MASE values obtained for train and test set are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Period | Model | MASE - Train Set | MASE – Test Set |
| Yearly | A,Ad,N | 0.7445443 | 1.165745 |
| Quarterly | A,Ad,A | 0.4204762 | 1.112868 |
| Monthly | A,Ad,A | 0.4162618 | 1.327163 |

Table 9 - Mean MASE - Test Set vs. Train Set

With the ETS models chosen for the 997 time series, over **66%** of the standardized residuals were found to be normally distributed, with approximately **85%** of the standardized residuals showing no significant serial correlation.

# Reference

International Institute of Forecasters. (2019). M3-Competition - International Institute of Forecasters. [online] Available at: https://forecasters.org/resources/time-series-data/m3-competition/ [Accessed 27 Oct. 2019].

# Appendix

library(readxl)

library(dynlm)

library(ggplot2)

library(AER)

library(Hmisc)

library(forecast)

library(x12)

library(dLagM)

library(TSA)

library(readr)

library(dplyr)

library(tseries)

library(FitAR)

source('GoFVals.R')

source('GoFVals\_damped.R')

source('MASEvalues.R')

source('MASEvalues\_damped.R')

source('MASE.forecast.R')

source('MATH1307\_utilityFunctions.R')

## ======== Yearly data analysis ======== ##

##loading yearly data

yearly <- read\_excel("M3C\_reduced\_2019.xlsx", sheet = "M3Year")

##converting to time series and bifurcating into train and test set

year.complete<-list()

year.training<-list()

yearly.testing<-list()

for (i in 1:333) {

year.complete[[i]] = ts(t(yearly[i,6:(5+yearly$N[i])]),

start = c(yearly$`Starting Year`[i]))

year.training[[i]] = ts(t(yearly[i,6:(5+trunc(0.95\*yearly$N[i]))]),

start = c(yearly$`Starting Year`[i]))

yearly.testing[[i]] = ts(t(yearly[i,((6+trunc(0.95\*(yearly$N[i])))):(5+yearly$N[i])]),

start = c(yearly$`Starting Year`[i]+trunc(0.95\*(yearly$N[i]))))

}

##testing stationarity using PP test

yearly\_pp\_test <- 0

for (i in 1:333) {

if (pp.test(year.training[[i]], lshort = TRUE)$p.value < 0.05){

yearly\_pp\_test <- yearly\_pp\_test + 1

}

}

yearly\_pp\_test

##applying box-cox transformation to handle changing variance

year\_lambda <- list()

BC.year.complete <- list()

BC.year.train <- list()

BC.year.test <- list()

for (i in 1:333) {

year\_lambda[[i]] <- BoxCox.lambda(year.complete[[i]])

BC.year.complete[[i]] = ((year.complete[[i]]^(year\_lambda[[i]])) - 1) / year\_lambda[[i]]

BC.year.train[[i]] = ((year.training[[i]]^(year\_lambda[[i]])) - 1) / year\_lambda[[i]]

BC.year.test[[i]] = ((yearly.testing[[i]]^(year\_lambda[[i]])) - 1) / year\_lambda[[i]]

}

## checking for any 0 or negtive values

year\_neg<-0

for (i in 1:332) {

if (BC.year.train[[i]] <0)

{year\_neg = year\_neg+1}

}

year\_neg

##Checking the information criteria and frequency of best model fittings

year\_H=6

# Models with undamped trend

year\_models=c("ANN","MNN","MMN","MAN","AAN")

year.govals<-GoFVals(BC.year.train,H = year\_H,models = year\_models)

year.govals.df<-as.data.frame(year.govals)

year.GoF.Best<-year.govals.df %>% group\_by(GoF.series) %>% slice(which.min(GoF.MASE))

table(year.GoF.Best$GoF.FittedModels)

year\_IC <- year.govals$GoF %>% group\_by(FittedModels) %>% summarise(AIC = mean(AIC), BIC = mean(BIC), HQIC = mean(HQIC), MASE = mean(MASE))

## comparing IC for models with undamped trend

year\_IC

## models with damped trend

year\_models\_damped=c("AAN", "MMN")

year.govals\_damped<-GoFVals\_damped(BC.year.train,H = year\_H,models = year\_models\_damped)

year.govals.df.damped<-as.data.frame(year.govals\_damped)

year.GoF.Best.damped<-year.govals.df.damped %>% group\_by(GoF.series) %>% slice(which.min(GoF.MASE))

table(year.GoF.Best.damped$GoF.FittedModels)

year\_IC\_damped <- year.govals\_damped$GoF %>% group\_by(FittedModels) %>% summarise(AIC = mean(AIC), BIC = mean(BIC), HQIC = mean(HQIC), MASE = mean(MASE))

## comparing models with damped trend

year\_IC\_damped

## comparing mean MASE and MASE rank

year\_MASEs = year.govals$GoF$MASE

MASEvalues(data = BC.year.train, H = year\_H, model = "AAN", MASEs = year\_MASEs)

year\_MASEs\_damped = year.govals\_damped$GoF$MASE

MASEvalues\_damped(data = BC.year.train, H = year\_H, model = "AAN", MASEs = year\_MASEs\_damped)

## forcasting train over test and calculating the accuracy of model chosen

year.model=vector("list",333)

for (i in 1:333) {

year.model[[i]]<-forecast(ets(BC.year.train[[i]],model = "AAN", damped = TRUE),h=nrow(BC.year.test[[i]]))

}

## calculating mean MASE test set

year\_MASE<-vector("list", 333)

for (i in 1:333) {

year\_MASE[[i]] <- MASE.forecast(BC.year.train[[i]], BC.year.test[[i]], year.model[[i]]$mean)

}

year\_MASE\_list = 0

for (j in 1:333){

year\_MASE\_list = year\_MASE\_list + year\_MASE[[j]]

}

year.MASE <- year\_MASE\_list/333

year.MASE

## residual diagnostics

year\_non.Norm.Std.R <- 0

year\_corr.Std.R <-0

for(i in 1:333){

if (shapiro.test(year.model[[i]]$residuals)$p.value < 0.05){

year\_non.Norm.Std.R <- year\_non.Norm.Std.R + 1

}

if (Box.test(year.model[[i]]$residuals, lag = 1, type = "Ljung-Box", fitdf = 0)$p.value < 0.05){

year\_corr.Std.R <- year\_corr.Std.R + 1

}

}

## calculating percentage of non normal standard residuals

100\*year\_non.Norm.Std.R/333

## calculating percentage of correlated standard residuals

100\*year\_corr.Std.R/333

## ======== End of Yearly data analysis ======== ##

## ======== Quarterly data analysis ======== ##

##loading quarterly data

quarterly <- read\_excel("M3C\_reduced\_2019.xlsx", sheet = "M3Quart")

##converting to time series and bifurcating into train and test set

quart.complete<-list()

quarterly.training<-list()

quarterly.testing<-list()

for (i in 1:332) {

quart.complete[[i]] = ts(t(quarterly[i,6:(5+quarterly$N[i])]),

start = c(quarterly$`Starting Year`[i],quarterly$`Starting Quarter`[i]),

frequency = 4)

quarterly.training[[i]] = ts(t(quarterly[i,6:(5+trunc(0.95\*quarterly$N[i]))]),

start = c(quarterly$`Starting Year`[i],quarterly$`Starting Quarter`[i]),

frequency = 4)

quarterly.testing[[i]] = ts(t(quarterly[i,((6+trunc(0.95\*(quarterly$N[i])))):(5+quarterly$N[i])]),

start = c(trunc(quarterly$`Starting Year`[i]+(quarterly$`Starting Quarter`[i]/4)+((0.95\*(quarterly$N[i]))/4)),floor((quarterly$`Starting Quarter`[i] + (0.95\*quarterly$N[i])))%%4),

frequency = 4)

}

##testing stationarity using PP Test

quarterly\_pp\_test <- 0

for (i in 1:332) {

if (pp.test(quart.complete[[i]], lshort = TRUE)$p.value < 0.05){

quarterly\_pp\_test <- quarterly\_pp\_test + 1

}

}

quarterly\_pp\_test

##applying box-cox transformation to handle changing variance

quarter\_lambda <- list()

BC.quarter.complete <- list()

BC.quarter.train <- list()

BC.quarter.test <- list()

for (i in 1:332) {

quarter\_lambda[[i]] <- BoxCox.lambda(quart.complete[[i]])

BC.quarter.complete[[i]] = ((quart.complete[[i]]^(quarter\_lambda[[i]])) - 1) / quarter\_lambda[[i]]

BC.quarter.train[[i]] = ((quarterly.training[[i]]^(quarter\_lambda[[i]])) - 1) / quarter\_lambda[[i]]

BC.quarter.test[[i]] = ((quarterly.testing[[i]]^(quarter\_lambda[[i]])) - 1) / quarter\_lambda[[i]]

}

## checking for any 0 or negtive values

quarter\_neg<-0

for (i in 1:332) {

if (BC.quarter.train[[i]] <0)

{quarter\_neg = quarter\_neg+1}

}

quarter\_neg

##Checking the information criteria and frequency of best model fittings

quarter\_H=8

# Models with undamped trend

quarter\_models=c("ANA","MNA","MNM","ANN","MMN","MAN","MAM","MMM","AAN","MNN","MAA","AAA")

quarter.govals<-GoFVals(BC.quarter.train,H = quarter\_H,models = quarter\_models)

quarter.govals.df<-as.data.frame(quarter.govals)

quarter.GoF.Best<-quarter.govals.df %>% group\_by(GoF.series) %>% slice(which.min(GoF.MASE))

table(quarter.GoF.Best$GoF.FittedModels)

quarter\_IC <- quarter.govals$GoF %>% group\_by(FittedModels) %>% summarise(AIC = mean(AIC), BIC = mean(BIC), HQIC = mean(HQIC), MASE = mean(MASE))

## comparing models with undamped trend

quarter\_IC

# Models with damped trend

quarter\_models\_damped=c("MMN","MAN","MAM","MMM","AAN","MAA","AAA")

quarter.govals\_damped<-GoFVals\_damped(BC.quarter.train,H = quarter\_H,models = quarter\_models\_damped)

quarter.govals.df.damped<-as.data.frame(quarter.govals\_damped)

quarter.GoF.Best.damped<-quarter.govals.df.damped %>% group\_by(GoF.series) %>% slice(which.min(GoF.MASE))

table(quarter.GoF.Best.damped$GoF.FittedModels)

quarter\_IC\_damped <- quarter.govals\_damped$GoF %>% group\_by(FittedModels) %>% summarise(AIC = mean(AIC), BIC = mean(BIC), HQIC = mean(HQIC), MASE = mean(MASE))

## comparing models with damped trend

quarter\_IC\_damped

## comparing mean MASE and MASE rank

quarter\_MASEs = quarter.govals$GoF$MASE

MASEvalues(data = BC.quarter.train, H = quarter\_H, model = "AAA", MASEs = quarter\_MASEs)

quarter\_MASEs\_damped = quarter.govals\_damped$GoF$MASE

MASEvalues\_damped(data = BC.quarter.train, H = quarter\_H, model = "AAA", MASEs = quarter\_MASEs\_damped)

## forcasting train over test and calculating the accuracy of model chosen

quarter.model=vector("list",332)

for (i in 1:332) {

quarter.model[[i]]<-forecast(ets(BC.quarter.train[[i]],model = "AAA", damped = TRUE),h=nrow(BC.quarter.test[[i]]))

}

## calculating mean MASE on test set

quarter\_MASE<-vector("list", 332)

for (i in 1:332) {

quarter\_MASE[[i]] <- MASE.forecast(BC.quarter.train[[i]], BC.quarter.test[[i]], quarter.model[[i]]$mean)

}

quarter\_MASE\_list = 0

for (j in 1:332){

quarter\_MASE\_list = quarter\_MASE\_list + quarter\_MASE[[j]]

}

quarter.MASE <- quarter\_MASE\_list/332

quarter.MASE

## residual diagnostics

quarter\_non.Norm.Std.R <- 0

quarter\_corr.Std.R <-0

for(i in 1:332){

if (shapiro.test(quarter.model[[i]]$residuals)$p.value < 0.05){

quarter\_non.Norm.Std.R <- quarter\_non.Norm.Std.R + 1

}

if (Box.test(quarter.model[[i]]$residuals, lag = 1, type = "Ljung-Box", fitdf = 0)$p.value < 0.05){

quarter\_corr.Std.R <- quarter\_corr.Std.R + 1

}

}

## calculating percentage of non normal standard residuals

100\*quarter\_non.Norm.Std.R/332

## calculating percentage of correlated standard residuals

100\*quarter\_corr.Std.R/332

## ======== End of Quarterly data analysis ======== ##

## ======== Monthly data analysis ======== ##

##loading monthly data

monthly <- read\_excel("M3C\_reduced\_2019.xlsx", sheet = "M3Month")

##converting to time series and bifurcating into train and test set

month.complete<-list()

month.training<-list()

month.testing<-list()

for (i in 1:332) {

month.complete[[i]] = ts(t(monthly[i,6:(5+monthly$N[i])]),

start = c(monthly$`Starting Year`[i],monthly$`Starting Month`[i]),

frequency = 12)

month.training[[i]] = ts(t(monthly[i,6:(5+trunc(0.95\*monthly$N[i]))]),

start = c(monthly$`Starting Year`[i],monthly$`Starting Month`[i]),

frequency = 12)

month.testing[[i]] = ts(t(monthly[i,((6+trunc(0.95\*(monthly$N[i])))):(5+monthly$N[i])]),

start = c(trunc(monthly$`Starting Year`[i]+(monthly$`Starting Month`[i]/12)+((0.95\*(monthly$N[i]))/12)),floor((monthly$`Starting Month`[i] + (0.95\*monthly$N[i])))%%12),

frequency = 12)

}

##testing stationarity using PP test

month\_pp\_test <- 0

for (i in 1:332) {

if (pp.test(month.complete[[i]], lshort = TRUE)$p.value < 0.05){

month\_pp\_test <- month\_pp\_test + 1

}

}

month\_pp\_test

##applying box-cox transformation to handle changing variance

month\_lambda <- list()

BC.month.complete <- list()

BC.month.train <- list()

BC.month.test <- list()

for (i in 1:332) {

month\_lambda[[i]] <- BoxCox.lambda(month.complete[[i]])

BC.month.complete[[i]] = ((month.complete[[i]]^(month\_lambda[[i]])) - 1) / month\_lambda[[i]]

BC.month.train[[i]] = ((month.training[[i]]^(month\_lambda[[i]])) - 1) / month\_lambda[[i]]

BC.month.test[[i]] = ((month.testing[[i]]^(month\_lambda[[i]])) - 1) / month\_lambda[[i]]

}

## checking for any 0 or negtive values

month\_neg<-0

for (i in 1:332) {

if (BC.month.train[[i]] <0)

{month\_neg = month\_neg+1}

}

month\_neg

##Checking the information criteria and frequency of best model fittings

month\_H=18

# Models with undamped trend

month\_models=c("ANA","MNA","MNM","ANN","MMN","MAN","MAM","MMM","AAN","MNN","MAA","AAA")

Month.govals<-GoFVals(BC.month.train,H = month\_H,models = month\_models)

Month.govals.df<-as.data.frame(Month.govals)

Month.GoF.Best<-Month.govals.df %>% group\_by(GoF.series) %>% slice(which.min(GoF.MASE))

table(Month.GoF.Best$GoF.FittedModels)

month\_IC <- Month.govals$GoF %>% group\_by(FittedModels) %>% summarise(AIC = mean(AIC), BIC = mean(BIC), HQIC = mean(HQIC), MASE = mean(MASE))

## comparing models with undamped trend

month\_IC

# Models with damped trend

month\_models\_damped=c("MMN","MAN","MAM","MMM","AAN","MAA","AAA")

month.govals\_damped<-GoFVals\_damped(BC.month.train,H = month\_H,models = month\_models\_damped)

month.govals.df.damped<-as.data.frame(month.govals\_damped)

month.GoF.Best.damped<-month.govals.df.damped %>% group\_by(GoF.series) %>% slice(which.min(GoF.MASE))

table(month.GoF.Best.damped$GoF.FittedModels)

month\_IC\_damped <- month.govals\_damped$GoF %>% group\_by(FittedModels) %>% summarise(AIC = mean(AIC), BIC = mean(BIC), HQIC = mean(HQIC), MASE = mean(MASE))

## comparing models with damped trend

month\_IC\_damped

## comparing mean MASE and MASE rank

month\_MASEs = month.govals$GoF$MASE

MASEvalues(data = BC.month.train, H = month\_H, model = "AAA", MASEs = month\_MASEs)

month\_MASEs\_damped = month.govals\_damped$GoF$MASE

MASEvalues\_damped(data = BC.month.train, H = month\_H, model = "AAA", MASEs = month\_MASEs\_damped)

## forcasting train over test and calculating the accuracy of model chosen

month.model <- vector("list",332)

for (i in 1:332) {

month.model[[i]]<-forecast(ets(BC.month.train[[i]],model = "AAA", damped = TRUE),h=nrow(BC.month.test[[i]]))

}

## calculating mean MASE on test set

month\_MASE<-vector("list", 332)

for (i in 1:332) {

month\_MASE[[i]] <- MASE.forecast(BC.month.train[[i]], BC.month.test[[i]], month.model[[i]]$mean)

}

month\_MASE\_list = 0

for (j in 1:332){

month\_MASE\_list = month\_MASE\_list + month\_MASE[[j]]

}

month.MASE <- month\_MASE\_list/332

month.MASE

## residual diagnostics

month\_non.Norm.Std.R <- 0

month\_corr.Std.R <-0

for(i in 1:332){

if (shapiro.test(month.model[[i]]$residuals)$p.value < 0.05){

month\_non.Norm.Std.R <- month\_non.Norm.Std.R + 1

}

if (Box.test(month.model[[i]]$residuals, lag = 1, type = "Ljung-Box", fitdf = 0)$p.value < 0.05){

month\_corr.Std.R <- month\_corr.Std.R + 1

}

}

## calculating percentage of non normal standard residuals

100\*month\_non.Norm.Std.R/332

## calculating percentage of correlated standard residuals

100\*month\_corr.Std.R/332

## ======== End of Monthly data analysis ======== ##

## ======== Computing penalty value ======== ##

Total\_non.Norm.Std.R <- month\_non.Norm.Std.R + quarter\_non.Norm.Std.R + year\_non.Norm.Std.R

Total\_non.Norm.Std.R

Total\_corr.Std.R <- month\_corr.Std.R + quarter\_corr.Std.R + year\_corr.Std.R

Total\_corr.Std.R

## ======== Forecasting for 6(yearly), 8(quarterly) and 18(monthly) periods ======== ##

year.forecasts <- vector("list",333)

quarter.forecasts <- vector("list",332)

month.forecasts <- vector("list",332)

for (i in 1:333) {

for (j in 1:332) {

for (k in 1:332) {

year.forecasts[[i]]<-forecast(ets(BC.year.complete[[i]],model = "AAN", damped = TRUE),h=6)

quarter.forecasts[[j]]<-forecast(ets(BC.quarter.complete[[j]],model = "AAA", damped = TRUE),h=8)

month.forecasts[[k]]<-forecast(ets(BC.month.complete[[k]],model = "AAA", damped = TRUE),h=18)

}

}

}

final.year.forecasts <- list()

final.quarter.forecasts <- list()

final.month.forecasts <- list()

for (i in 1:333) {

for (j in 1:332) {

for (k in 1:332) {

final.year.forecasts[[i]]<-((year\_lambda[[i]])\*(year.forecasts[[i]]$mean) + 1)^(-year\_lambda[[i]])

final.quarter.forecasts[[j]]<-(quarter\_lambda[[j]]\*quarter.forecasts[[j]]$mean + 1)^(-quarter\_lambda[[j]])

final.month.forecasts[[k]]<-(month\_lambda[[k]]\*month.forecasts[[k]]$mean + 1)^(-month\_lambda[[k]])

}

}

}