Code

Forecasting models for M3 series

Forecasting Project - Competitive

MUJEER M.

Introduction

The dataset we are working on, is a subset provided from the original M3 competition 3003 series from International Institute of Forecasters (IIF) which is led by forecasting researcher Spyros Makridakis. This dataset is reduced to 1000 series including yearly, quarterly, and monthly micro-economic, macro-economic, industrial, financial, and demographic time series data. Our aim is to find the best fitted model for each series and produce training and test MASE results of all 3 frequencies (yearly, quarterly, monthly).

Methodology

- We have loaded and analysed all 3 frequency series separately
- Created training set (95%) and test set (5%) for all 1000 series
- We used the GoFVals function to run ETS loop (the function didn't have any usage of H value, hence ignored this).
- Created and run a loop for all possible set of ETS and Holt's models, did a transformation where necessary
- Filtered the ETS models based on minimum MASE and HQIC results
- Created and run a loop for Shapiro test(to find the no. of non-normal residuals) and Ljung-box test(to find the no. of correlated std.residuals) based on 5% significance level for all the best models
- Please note that I have hidden all residuals plots as its consuming too many pages
- Ran the MASE.forecast function to get the training and test sample MASE
- Compared all the parameters listed in project specification to select the best modelling method for each frequency

Forecasting - Yearly Series

Setup

Loading functions

Hide

```
source("C:/Users/Mohammed/Desktop/Sem 3/Forecasting/Data Files and R Scripts/GoFVals.R")
source("C:/Users/Mohammed/Desktop/Sem 3/Forecasting/Data Files and R Scripts/MASE.forecast.R"
)
```

Importing data - Yearly, Quarterly and Monthly series

Hide

```
##Yearly data series
M3C_reduced_2019_Year <- read_excel("C:/Users/Mohammed/Desktop/Sem 3/Forecasting/Project/M3C_
reduced_2019.xlsx", sheet = "M3Year")
```

```
New names:
* `` -> ...5
```

Hide

```
##Quarterly series
M3C_reduced_2019_Quarterly <- read_excel("C:/Users/Mohammed/Desktop/Sem 3/Forecasting/Projec
t/M3C_reduced_2019.xlsx", sheet = "M3Quart")

##Monthly series
M3C_reduced_2019_Month <- read_excel("C:/Users/Mohammed/Desktop/Sem 3/Forecasting/Project/M3C
_reduced_2019.xlsx", sheet = "M3Month")
```

Ordering the datasets based on N value

Hide

```
M3C_reduced_2019_Year <- M3C_reduced_2019_Year[order(M3C_reduced_2019_Year$N),]
head(M3C_reduced_2019_Year)
```

N <dbl>	NF <dbl>	Category <chr>	Starting Year <dbl>	...5 <dbl>	1 <dbl>	2 <dbl>	3 <dbl>	4 <dbl>	5 <dbl>
20	6	MICRO	1975	1	3637.13	4086.84	4785.57	5166.65	6473.86
20	6	MICRO	1975	1	1461.57	1692.50	2193.82	2459.68	3246.80
20	6	MICRO	1975	1	48.00	96.04	288.40	351.04	421.44
20	6	MICRO	1975	1	80.17	111.61	118.57	139.16	209.91
20	6	MICRO	1975	1	773.40	939.60	1227.45	1496.30	1855.30
20	6	MICRO	1975	1	4591.48	4939.08	4898.89	4933.19	5165.89

6 rows | 1-10 of 52 columns

Hide

```
M3C_reduced_2019_Quarterly <- M3C_reduced_2019_Quarterly[order(M3C_reduced_2019_Quarterly
$N),]
head(M3C_reduced_2019_Quarterly)
```

N <dbl>	… <dbl>	Category <chr>	Starting Year <dbl>	Starting Quarter <dbl>	1 <dbl>	2 <dbl>	3 <dbl>	4 <dbl>	5 <dbl>
24	8	MACRO	1987	1	5117.5	5203.0	5208.5	5260.5	5280.0
24	8	MACRO	1987	1	5122.5	5238.5	5218.5	5251.0	5181.0

N <dbl>	… <dbl>	Category <chr>	Starting Year <dbl>	Starting Quarter <dbl>	1 <dbl>	2 <dbl>	3 <dbl>	4 <dbl>	5 <dbl>
24	8	MACRO	1987	1	5129.0	5132.5	5154.5	5176.5	5182.0
24	8	MACRO	1987	1	5736.0	5535.0	5633.5	5633.5	5453.0
24	8	MACRO	1987	1	5026.0	5187.0	5329.5	5479.0	5702.5
24	8	MACRO	1987	1	4474.0	4730.0	4674.0	5016.0	4707.0

6 rows | 1-10 of 77 columns

Hide

```
M3C_reduced_2019_Month <- M3C_reduced_2019_Month[order(M3C_reduced_2019_Month$N),]
head(M3C_reduced_2019_Month)
```

N <dbl>	NF <dbl>	Category <chr>	Starting Year <dbl>	Starting Month <dbl>	1 <dbl>	2 <dbl>	3 <dbl>	4 <dbl>	5 <dbl>
68	18	MICRO	1990	1	2640	2640	2160	4200	3360
68	18	MICRO	1990	1	1680	1920	120	1080	840
68	18	MICRO	1990	1	1140	720	4860	1200	3150
68	18	MICRO	1990	1	180	940	2040	800	1000
68	18	MICRO	1990	1	2000	1550	4450	3050	3050
68	18	MICRO	1990	1	1200	2850	1350	1500	1950

6 rows | 1-10 of 149 columns

Hide

```
NA
NA
```

We ordered all the 3 frequency series by N value i.e. the length of each series for ease of manipulating the data

Creating train and test lists based on length of each series

Hide

```
smp_size_list_Y <- list()
smp_size_test_list_Y <- list()

for(i in 1:333)
{
  smp_size_list_Y[[i]] <- floor(0.95 * M3C_reduced_2019_Year$N[[i]])
  smp_size_test_list_Y[[i]] <- floor(0.05 * M3C_reduced_2019_Year$N[[i]])
}

head(smp_size_list_Y[[i]])
```

```
[1] 44
```

Loop for 95% of Yearly series

Hide

```
train_row_year <- list()


for(i in 1:333)
{
  train_row_year[[i]] <- M3C_reduced_2019_Year[i ,1:smp_size_list_Y[[i]]+5]



}
```

Creating a list of 95% train set for yearly series

Converting list to dataframe

Hide

```


train_row_year_df <- ldply (train_row_year, data.frame)


head(train_row_year_df)
```

	X1 <dbl>	X2 <dbl>	X3 <dbl>	X4 <dbl>	X5 <dbl>	X6 <dbl>	X7 <dbl>	X8 <dbl>	X9 <dbl>
1	3637.13	4086.84	4785.57	5166.65	6473.86	8118.37	9262.07	8864.66	7852.30
2	1461.57	1692.50	2193.82	2459.68	3246.80	4748.86	5559.46	5292.42	5029.40
3	48.00	96.04	288.40	351.04	421.44	413.36	449.52	984.52	824.76
4	80.17	111.61	118.57	139.16	209.91	346.39	387.45	441.35	527.64
5	773.40	939.60	1227.45	1496.30	1855.30	2274.65	2792.20	3346.30	3764.15
6	4591.48	4939.08	4898.89	4933.19	5165.89	5206.79	5282.09	4611.29	4457.38

6 rows | 1-10 of 44 columns

Hide

```
NA
```

Possible set of models under ETS for Yearly series

Hide

```


models = c("ANN","MNN","MAN","MMN","AAN")
```

Running GoFVals function for all models

Running the 5 ETS models using GoFVals, we will consider both minimum HQIC and minimum MASE value models to see which parameter will perform better in residual analysis and auto-correlation of the series.

Converting the list output to dataframe

Hide

```
train_year_mase_df <- ldply (train_year_mase, data.frame)

str(train_year_mase_df)
```

```
'data.frame':   1665 obs. of  8 variables:
 $ .id         : chr  "GoF" "GoF" "GoF" "GoF" ...
 $ series      : int [1:1665(1d)] 1 1 1 1 1 2 2 2 2 2 ...
 $ FittedModels: chr [1:1665(1d)] "ANN" "MNN" "MAN" "MMN" ...
 $ AIC         : num [1:1665(1d)] 313 316 310 314 315 ...
 $ AICc        : num [1:1665(1d)] 315 317 315 318 320 ...
 $ BIC         : num [1:1665(1d)] 316 319 315 318 320 ...
 $ HQIC        : num [1:1665(1d)] 313 315 311 314 316 ...
 $ MASE        : num [1:1665(1d)] 0.948 0.951 0.814 0.955 0.837 ...
```

Filtering models based on minimum HQIC and minimum MASE separately for yearly series

Hide

```
train_year_mase_min_df <- train_year_mase_df %>% group_by(series) %>% filter(MASE==min(MASE))
train_year_hqic_min_df <- train_year_mase_df %>% group_by(series) %>% filter(HQIC==min(HQIC))
```

As mentioned above, we will consider both MASE and HQIC for further analysis

Running ETS loop for models selected based on minimum MASE & minimum HQIC

Hide

```
glimpse(model_train_year[[i]])
```

```
List of 19
 $ loglik    : num -290
 $ aic       : num 591
 $ bic       : num 600
 $ aicc      : num 592
 $ mse       : num 13146
 $ amse      : num 27502
 $ fit       :List of 4
  ..$ value  : num 581
  ..$ par    : num [1:4] 9.54e-01 1.00e-04 4.84e+03 1.02
  ..$ fail   : int 0
  ..$ fncount: int 441
 $ residuals : Time-Series [1:44] from 1 to 44: -0.00445 0.01149 -0.03299 0.01349 0.00905 ...
 $ fitted    : Time-Series [1:44, 1] from 1 to 44: 4937 5014 5170 5107 5276 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr "y"
 $ states    : Time-Series [1:45, 1:2] from 0 to 44: 4840 4916 5069 5007 5173 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "l" "b"
 $ par       : Named num [1:4] 9.54e-01 1.00e-04 4.84e+03 1.02
  ..- attr(*, "names")= chr [1:4] "alpha" "beta" "l" "b"
 $ m         : num 1
 $ method    : chr "ETS(M,M,N)"
 $ series    : chr "ts(t(train_row_year_df[i, ]))"
 $ components: chr [1:4] "M" "M" "N" "FALSE"
 $ call      : language ets(y = ts(t(train_row_year_df[i, ])), model = train_year_mase_min_df
$FittedModels[[i]])
 $ initstate : Named num [1:2] 4840.25 1.02
  ..- attr(*, "names")= chr [1:2] "l" "b"
 $ sigma2    : num 0.000249
 $ x         : Time-Series [1:44, 1] from 1 to 44: 4915 5071 4999 5176 5324 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr "332"
 - attr(*, "class")= chr "ets"
```

Hide

```
glimpse(model_train_year_hqic[[i]])
```

```
List of 19
 $ loglik    : num -290
 $ aic       : num 591
 $ bic       : num 600
 $ aicc      : num 592
 $ mse       : num 13146
 $ amse      : num 27502
 $ fit       :List of 4
  ..$ value  : num 581
  ..$ par    : num [1:4] 9.54e-01 1.00e-04 4.84e+03 1.02
  ..$ fail   : int 0
  ..$ fncount: int 441
 $ residuals : Time-Series [1:44] from 1 to 44: -0.00445 0.01149 -0.03299 0.01349 0.00905 ...
 $ fitted    : Time-Series [1:44, 1] from 1 to 44: 4937 5014 5170 5107 5276 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr "y"
 $ states    : Time-Series [1:45, 1:2] from 0 to 44: 4840 4916 5069 5007 5173 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "l" "b"
 $ par       : Named num [1:4] 9.54e-01 1.00e-04 4.84e+03 1.02
  ..- attr(*, "names")= chr [1:4] "alpha" "beta" "l" "b"
 $ m         : num 1
 $ method    : chr "ETS(M,M,N)"
 $ series    : chr "ts(t(train_row_year_df[i, ]))"
 $ components: chr [1:4] "M" "M" "N" "FALSE"
 $ call      : language ets(y = ts(t(train_row_year_df[i, ])), model = train_year_hqic_min_df
$FittedModels[[i]])
 $ initstate : Named num [1:2] 4840.25 1.02
  ..- attr(*, "names")= chr [1:2] "l" "b"
 $ sigma2    : num 0.000249
 $ x         : Time-Series [1:44, 1] from 1 to 44: 4915 5071 4999 5176 5324 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr "332"
 - attr(*, "class")= chr "ets"
```

Running forecast function loop for minimum HQIC and minimum MASE

Hide

```
forecast_year_hqic <- list()
forecast_year_mase <- list()

for(i in 1:333)
{

  forecast_year_hqic[[i]] <- forecast.ets(model_train_year_hqic[[i]], h =  smp_size_test_list
_Y[[i]])
  forecast_year_mase[[i]] <- forecast.ets(model_train_year[[i]], h =  smp_size_test_list_Y
[[i]])

}

head(forecast_year_hqic[[i]])
```

```
$model
ETS(A,N,N)

Call:
 ets(y = ts(t(train_row_year_df[i, ])), model = train_year_hqic_min_df$FittedModels[[i]])

  Smoothing parameters:
    alpha = 0.9396

  Initial states:
    l = 3900.822

  sigma:  1151.255

     AIC     AICc      BIC
790.7349 791.3349 796.0875

$mean
Time Series:
Start = 45
End = 46
Frequency = 1
[1] 5488.817 5488.817

$level
[1] 80 95

$x
Time Series:
Start = 1
End = 44
Frequency = 1
       333
 [1,] 3900
 [2,] 3800
 [3,] 5900
 [4,] 5300
 [5,] 3300
 [6,] 3000
 [7,] 2900
 [8,] 5500
 [9,] 4400
[10,] 4100
[11,] 4300
[12,] 6800
[13,] 5500
[14,] 5500
[15,] 6700
[16,] 5500
[17,] 5700
[18,] 5200
[19,] 4500
[20,] 3800
[21,] 3800
[22,] 3600
[23,] 3500
[24,] 4900
```

```
[25,] 5900
[26,] 5600
[27,] 4900
[28,] 5600
[29,] 8500
[30,] 7700
[31,] 7100
[32,] 6100
[33,] 5800
[34,] 7100
[35,] 7600
[36,] 9700
[37,] 9600
[38,] 7500
[39,] 7200
[40,] 7000
[41,] 6200
[42,] 5500
[43,] 5300
[44,] 5500

$upper
Time Series:
Start = 45
End = 46
Frequency = 1
        80%      95%
45 6964.209 7745.234
46 7513.316 8585.021

$lower
Time Series:
Start = 45
End = 46
Frequency = 1
        80%      95%
45 4013.424 3232.399
46 3464.317 2392.612
```

Hide

```
head(forecast_year_mase[[i]])
```

```
$model
ETS(A,N,N)

Call:
 ets(y = ts(t(train_row_year_df[i, ])), model = train_year_mase_min_df$FittedModels[[i]])

  Smoothing parameters:
    alpha = 0.9396

  Initial states:
    l = 3900.822

  sigma:  1151.255

     AIC     AICc      BIC
790.7349 791.3349 796.0875

$mean
Time Series:
Start = 45
End = 46
Frequency = 1
[1] 5488.817 5488.817

$level
[1] 80 95

$x
Time Series:
Start = 1
End = 44
Frequency = 1
       333
 [1,] 3900
 [2,] 3800
 [3,] 5900
 [4,] 5300
 [5,] 3300
 [6,] 3000
 [7,] 2900
 [8,] 5500
 [9,] 4400
[10,] 4100
[11,] 4300
[12,] 6800
[13,] 5500
[14,] 5500
[15,] 6700
[16,] 5500
[17,] 5700
[18,] 5200
[19,] 4500
[20,] 3800
[21,] 3800
[22,] 3600
[23,] 3500
[24,] 4900
```

```
[25,] 5900
[26,] 5600
[27,] 4900
[28,] 5600
[29,] 8500
[30,] 7700
[31,] 7100
[32,] 6100
[33,] 5800
[34,] 7100
[35,] 7600
[36,] 9700
[37,] 9600
[38,] 7500
[39,] 7200
[40,] 7000
[41,] 6200
[42,] 5500
[43,] 5300
[44,] 5500

$upper
Time Series:
Start = 45
End = 46
Frequency = 1
        80%      95%
45 6964.209 7745.234
46 7513.316 8585.021

$lower
Time Series:
Start = 45
End = 46
Frequency = 1
        80%      95%
45 4013.424 3232.399
46 3464.317 2392.612
```

We used the test list length created for each series to specify the H value (number of forecasts) to match the accuracy of forecasts on test set.

Checkresiduals loop for ljung-box auto-correlation test for minimum MASE and minimum HQIC models

Hide

```
glimpse(model_train_year_res[[i]])
```

```
List of 5
 $ statistic: Named num 5.11
  ..- attr(*, "names")= chr "Q*"
 $ parameter: Named num 5
  ..- attr(*, "names")= chr "df"
 $ p.value  : num 0.403
 $ method   : chr "Ljung-Box test"
 $ data.name: chr "Residuals from ETS(M,M,N)"
 - attr(*, "class")= chr "htest"
```

Hide

```
glimpse(model_train_year_hqic_res[[i]])
```

```
List of 5
 $ statistic: Named num 5.11
  ..- attr(*, "names")= chr "Q*"
 $ parameter: Named num 5
  ..- attr(*, "names")= chr "df"
 $ p.value  : num 0.403
 $ method   : chr "Ljung-Box test"
 $ data.name: chr "Residuals from ETS(M,M,N)"
 - attr(*, "class")= chr "htest"
```

Loop for shapiro-test on both minimum MASE and minimum HQIC models

Hide

```


model_train_year_hqic_res_ST <- list()
model_train_year_res_ST <- list()

for (i in 1:333)
{

  model_train_year_res_ST[[i]] <- shapiro.test(model_train_year[[i]]$residuals)

  model_train_year_hqic_res_ST[[i]] <- shapiro.test(model_train_year_hqic[[i]]$residuals)

}


head(model_train_year_hqic_res_ST)
```

```
[[1]]

	Shapiro-Wilk normality test

data:  model_train_year_hqic[[i]]$residuals
W = 0.97021, p-value = 0.7807


[[2]]

	Shapiro-Wilk normality test

data:  model_train_year_hqic[[i]]$residuals
W = 0.96705, p-value = 0.7162


[[3]]

	Shapiro-Wilk normality test

data:  model_train_year_hqic[[i]]$residuals
W = 0.93768, p-value = 0.2394


[[4]]

	Shapiro-Wilk normality test

data:  model_train_year_hqic[[i]]$residuals
W = 0.95703, p-value = 0.5153


[[5]]

	Shapiro-Wilk normality test

data:  model_train_year_hqic[[i]]$residuals
W = 0.93849, p-value = 0.2477


[[6]]

	Shapiro-Wilk normality test

data:  model_train_year_hqic[[i]]$residuals
W = 0.96091, p-value = 0.5904
```

Hide

```
head(model_train_year_res_ST)
```

```
[[1]]

    Shapiro-Wilk normality test

data:  model_train_year[[i]]$residuals
W = 0.97021, p-value = 0.7807


[[2]]

    Shapiro-Wilk normality test

data:  model_train_year[[i]]$residuals
W = 0.95991, p-value = 0.5706


[[3]]

    Shapiro-Wilk normality test

data:  model_train_year[[i]]$residuals
W = 0.93768, p-value = 0.2394


[[4]]

    Shapiro-Wilk normality test

data:  model_train_year[[i]]$residuals
W = 0.95703, p-value = 0.5153


[[5]]

    Shapiro-Wilk normality test

data:  model_train_year[[i]]$residuals
W = 0.6786, p-value = 3.124e-05


[[6]]

    Shapiro-Wilk normality test

data:  model_train_year[[i]]$residuals
W = 0.95081, p-value = 0.408
```

Loop for extracting p-values from ljung box and shapiro test for minimum MASE and minimum HQIC models

Hide

```
res_p <- list()
res_p_hqic <- list()
res_p_hqic_ST <- list()
res_p_ST <- list()

for (i in 1:333)
{

  res_p[[i]] <-  model_train_year_res[[i]]$p.value

}


for (i in 1:333)
{

  res_p_hqic[[i]] <-  model_train_year_hqic_res[[i]]$p.value

}


for (i in 1:333)
{

  res_p_ST[[i]] <-  model_train_year_res_ST[[i]]$p.value

}

for (i in 1:333)
{

  res_p_hqic_ST[[i]] <-  model_train_year_hqic_res_ST[[i]]$p.value

}

res_p_df <- ldply (res_p, data.frame)
res_p_ST_df <- ldply (res_p_ST, data.frame)
res_p__hqic_df <- ldply (res_p_hqic, data.frame)
res_p_hqic_ST_df <- ldply (res_p_hqic_ST, data.frame)


res_p_df$series <- seq.int(nrow(res_p_df))
res_p_ST_df$series <- seq.int(nrow(res_p_ST_df))
res_p__hqic_df$series <- seq.int(nrow(res_p__hqic_df))
res_p_hqic_ST_df$series <- seq.int(nrow(res_p_hqic_ST_df))

names(res_p_df)[names(res_p_df) == "X..i.."] <- "p"
names(res_p_ST_df)[names(res_p_ST_df) == "X..i.."] <- "p"
names(res_p__hqic_df)[names(res_p__hqic_df) == "X..i.."] <- "p"
names(res_p_hqic_ST_df)[names(res_p_hqic_ST_df) == "X..i.."] <- "p"


head(res_p)
```

```
[[1]]
[1] 0.002429532

[[2]]
[1] 0.03275378

[[3]]
[1] 0.1447002

[[4]]
[1] 0.002878859

[[5]]
[1] 0.4723677

[[6]]
[1] 0.7102069
```

Hide

```
head(res_p_ST)
```

```
[[1]]
[1] 0.780746

[[2]]
[1] 0.5706378

[[3]]
[1] 0.2394239

[[4]]
[1] 0.5153388

[[5]]
[1] 3.124356e-05

[[6]]
[1] 0.4079653
```

Hide

```
head(res_p_hqic)
```

```
[[1]]
[1] 0.002429532

[[2]]
[1] 0.09503312

[[3]]
[1] 0.1447002

[[4]]
[1] 0.002878859

[[5]]
[1] 0.1330733

[[6]]
[1] 0.6838321
```

Hide

```
head(res_p_hqic_ST)
```

```
[[1]]
[1] 0.780746

[[2]]
[1] 0.7162051

[[3]]
[1] 0.2394239

[[4]]
[1] 0.5153388

[[5]]
[1] 0.2476518

[[6]]
[1] 0.590419
```

Since there are too many plots and outputs to individually verify the plots for each series, we filtered out the P-values from Shapiro test and Ljung box test.

Printing pass or fail results based on 0.05 significance level for minimum MASE and minimum HQIC models

Hide

```
res_p_df$outcome <- ifelse(
  (
    res_p_df$p > 0.05
  ),
  "pass",
  "fail"
)


res_p_ST_df$outcome <- ifelse(
  (
    res_p_ST_df$p > 0.05
  ),
  "pass",
  "fail"
)

res_p__hqic_df$outcome <- ifelse(
  (
    res_p__hqic_df$p > 0.05
  ),
  "pass",
  "fail"
)


res_p_hqic_ST_df$outcome <- ifelse(
  (
    res_p_hqic_ST_df$p > 0.05
  ),
  "pass",
  "fail"
)
```

Printing pass or fail results for Shapiro-test and Ljung box test on 0.05 level of significance.

Sorting results of Shapiro-test and Ljung-box test for minimum MASE and minimum HQIC models

Hide

```
final_result_res_Mfilt <- sqldf("Select outcome,count(*) from res_p_df group by outcome");
final_result_ST_Mfilt <- sqldf("Select outcome,count(*) from res_p_ST_df group by outcome");

final_result_hqicfilt <- sqldf("Select outcome,count(*) from res_p__hqic_df group by outcome"
);
final_result_hqic_ST <- sqldf("Select outcome,count(*) from res_p_hqic_ST_df group by outcom
e");

head(final_result_res_Mfilt)
```

	outcome <chr>	count(*) <int>
1	fail	106

	outcome <chr>	count(*) <int>
2	pass	227

2 rows

Hide

```
head(final_result_ST_Mfilt)
```

	outcome <chr>	count(*) <int>
1	fail	89
2	pass	244

2 rows

Hide

```
head(final_result_hqicfilt)
```

	outcome <chr>	count(*) <int>
1	fail	86
2	pass	247

2 rows

Hide

```
head(final_result_hqic_ST)
```

	outcome <chr>	count(*) <int>
1	fail	67
2	pass	266

2 rows

Hide

```
NA
NA
```

- Not running the HW models as there is no repetitive seasonal pattern, which is the reason for models not running with frequency 1

Converting train, test and forecasts from models into vector to run MASE.forecast

Hide

```
Year_Test <- read_excel("C:/Users/Mohammed/Desktop/Sem 3/Forecasting/Project/Year_Test.xlsx")
```

```
New names:
* `` -> ...5
```

Hide

```
Year_Test <- Year_Test[order(Year_Test$N),]

Y_vec_month_test <- unlist(Year_Test[,6:7])
Y_vect_month_test <- na.omit(Y_vec_month_test)

Y_vec_train_month <- unlist(train_row_year_df[,1:44])
Y_vec_train_month <- na.omit(Y_vec_train_month)

list_fitted_train_hqic_year <- list()
forecast_Y_hqic_ets <- list()
forecast_Y_mase_ets <- list()
list_fitted_train_mase_year <- list()

for(i in 1:333)
{

##Forecast values

forecast_Y_hqic_ets[[i]] <- forecast_year_hqic[[i]]$mean
forecast_Y_mase_ets[[i]] <- forecast_year_mase[[i]]$mean

##Fitted

list_fitted_train_hqic_year[[i]] <- forecast_year_hqic[[i]]$fitted
list_fitted_train_mase_year[[i]] <- forecast_year_mase[[i]]$fitted

}


###unlisting/converting to vector, omitting NA values
Y_vect_mean_ets <- na.omit(forecast_Y_hqic_ets)
Y_vect_mean_ets <- unlist(forecast_Y_hqic_ets)
Y_vect_mean_ets_mase <- na.omit(forecast_Y_mase_ets)
Y_vect_mean_ets_mase <- unlist(forecast_Y_mase_ets)

Y_vect_fitted_ets <- na.omit(list_fitted_train_hqic_year)
Y_vect_fitted_ets <- unlist(list_fitted_train_hqic_year)
Y_vect_fitted_ets_mase <- na.omit(list_fitted_train_mase_year)
Y_vect_fitted_ets_mase <- unlist(list_fitted_train_mase_year)
```

- We extracted the forecasts(5%) based on the length of each series for both HQIC and MASE filtered models
- We extracted the fitted values after modelling on the train set for both HQIC and MASE filtered models
- Omitted any NA values from train, test and forecasts data
- Converted train, test and forecasts to vector for running the MASE.forecast function

Calculating MASE for train and test sample

Hide

```
#ETS Mase forecast for train and test based on models filtered by MASE and HQIC
MASE_ets_hqic_Y_train <- MASE.forecast(Y_vec_train_month,Y_vec_train_month,Y_vect_fitted_ets)
MASE_ets_hqic_Y_test <- MASE.forecast(Y_vec_train_month,Y_vect_month_test,Y_vect_mean_ets)
```

```
longer object length is not a multiple of shorter object length
```

Hide

```
MASE_ets_mase_Y_train <- MASE.forecast(Y_vec_train_month,Y_vec_train_month,Y_vect_fitted_ets_
mase)
MASE_ets_mase_Y_test <- MASE.forecast(Y_vec_train_month,Y_vect_month_test,Y_vect_mean_ets_mas
e)
```

```
longer object length is not a multiple of shorter object length
```

Hide

```
##MASE values
list(MASE_ets_hqic_Y_train,MASE_ets_hqic_Y_test,MASE_ets_mase_Y_train,MASE_ets_mase_Y_test)
```

```
[[1]]
[1] 1.440704

[[2]]
[1] 1.179787

[[3]]
[1] 1.439287

[[4]]
[1] 1.122732
```

Forecasting - QUARTERLY SERIES

Creating train and test lists based on length of each series

Hide

```


Q_smp_size_list <- list()
Q_smp_size_test_list <- list()

for(i in 1:332)
{
  Q_smp_size_list[[i]] <- floor(0.95 *M3C_reduced_2019_Quarterly$N[[i]])
  Q_smp_size_test_list[[i]] <- floor(0.05 * M3C_reduced_2019_Quarterly$N[[i]])
}

head(Q_smp_size_list[[i]])
```

```
[1] 68
```

Loop for 95% of quarterly series

Hide

```
Q_train_row <- list()
Q_test_row <- list()

for(i in 1:332)
{
  Q_train_row[[i]] <- M3C_reduced_2019_Quarterly[i ,1:Q_smp_size_list[[i]]+5]

}

head(Q_train_row[[i]])
```

1 <dbl>	2 <dbl>	3 <dbl>	4 <dbl>	5 <dbl>	6 <dbl>	7 <dbl>	8 <dbl>	9 <dbl>	10 <dbl>
3223.5	3674.5	3335	2914	3191.5	3705	3214	3270	3589.5	3988.5

1 row | 1-10 of 68 columns

Hide

```
NA
```

Creating a list of 95% train set for quarterly series

Converting list to dataframe

Hide

```


Q_train_row_df <- ldply (Q_train_row, data.frame)

Q_train_ts <- list()


for (i in 1:332)
{
  Q_train_ts[[i]] <- ts(t(Q_train_row_df[i,]), frequency = 4)
}

head(Q_train_row_df)
```

	X1 <dbl>	X2 <dbl>	X3 <dbl>	X4 <dbl>	X5 <dbl>	X6 <dbl>	X7 <dbl>	X8 <dbl>	X9 <dbl>
1	5117.5	5203.0	5208.5	5260.5	5280.0	5185.5	5281.0	5285.0	5291.0
2	5122.5	5238.5	5218.5	5251.0	5181.0	5114.5	5149.5	5172.5	5134.5
3	5129.0	5132.5	5154.5	5176.5	5182.0	5210.0	5218.5	5177.0	5161.0
4	5736.0	5535.0	5633.5	5633.5	5453.0	5348.0	5176.0	5062.5	5276.0
5	5026.0	5187.0	5329.5	5479.0	5702.5	5496.0	5721.0	5738.5	5623.5
6	4474.0	4730.0	4674.0	5016.0	4707.0	4822.5	4782.5	5128.5	4851.5

6 rows | 1-10 of 68 columns

Hide

```
NA
```

Possible set of models under ETS for Quarterly series

Hide

```
models = c("ANN","MNN","MAN","MMN","AAN","MMM","MAM","AAA","MAA","MNM","ANA","MNA")
```

Running GoFVals function for all models

Running all 12 ETS models using GoFVals function, we will consider both minimum HQIC and minimum MASE value models to see which parameter will perform better in residual analysis and auto-correlation of the series.

Converting the list output to dataframe

Hide

```
train_Quarter_mase_df <- ldply (train_quarter_mase, data.frame)

glimpse(train_Quarter_mase_df)
```

```
Observations: 3,984
Variables: 8
$ .id          [3m[38;5;246m<chr>[39m[23m "GoF", "GoF", "GoF", "GoF", "GoF", "GoF", "Go
F", "GoF", "GoF", "GoF", "GoF", "GoF", "G...
$ series       [3m[38;5;246m<int>[39m[23m <array[37]>
$ FittedModels [3m[38;5;246m<chr>[39m[23m "ANN", "MNN", "MAN", "MMN", "AAN", "MMM", "MA
M", "AAA", "MAA", "MNM", "ANA", "MNA", "A...
$ AIC          [3m[38;5;246m<dbl>[39m[23m <array[37]>
$ AICc         [3m[38;5;246m<dbl>[39m[23m <array[37]>
$ BIC          [3m[38;5;246m<dbl>[39m[23m <array[37]>
$ HQIC         [3m[38;5;246m<dbl>[39m[23m <array[37]>
$ MASE         [3m[38;5;246m<dbl>[39m[23m <array[37]>
```

Filtering models based on minimum HQIC and minimum MASE separately for Quarterly series

Hide

```
train_Quarter_hqic_min_df <- train_Quarter_mase_df %>% group_by(series) %>% filter(HQIC==min
(HQIC))
train_Quarter_mase_min_df <- train_Quarter_mase_df %>% group_by(series) %>% filter(MASE==min
(MASE))
```

As mentioned above, we will consider both MASE and HQIC for further analysis

Running ETS loop for models selected based on minimum HQIC and minimumm MASE

Hide

```
Q_model_train_hqic <-list()
Q_model_train_mase <-list()

for (i in 1:332)
{

  Q_model_train_hqic[[i]] <- ets(ts(t(Q_train_row_df[i,]),frequency = 4), model = train_Quart
er_hqic_min_df$FittedModels[[i]])

    Q_model_train_mase[[i]] <- ets(ts(t(Q_train_row_df[i,]),frequency = 4), model = train_Qua
rter_mase_min_df$FittedModels[[i]])

}

head(Q_model_train_hqic[[i]])
```

```
$loglik
[1] -524.8266

$aic
[1] 1063.653

$bic
[1] 1079.19

$aicc
[1] 1065.52

$mse
[1] 74352.23

$amse
[1] 158922.8
```

Hide

```
head(Q_model_train_mase[[i]])
```

```
$loglik
[1] -527.9248

$aic
[1] 1073.85

$bic
[1] 1093.825

$aicc
[1] 1076.953

$mse
[1] 72765.84

$amse
[1] 161003
```

Running forecast function loop for minimum HQIC and minimum MASE

Hide

```


forecast_quarter_hqic <- list()
forecast_quarter_mase <- list()



for(i in 1:332)
{

  forecast_quarter_hqic[[i]] <- forecast.ets(Q_model_train_hqic[[i]], h =  Q_smp_size_test_li
st[[i]])
  forecast_quarter_mase[[i]] <- forecast.ets(Q_model_train_mase[[i]], h =  Q_smp_size_test_li
st[[i]])

}

head(forecast_quarter_hqic[[i]])
```

```
$model
ETS(A,N,A)

Call:
 ets(y = ts(t(Q_train_row_df[i, ]), frequency = 4), model = train_Quarter_hqic_min_df$FittedM
odels[[i]])

  Smoothing parameters:
    alpha = 0.9985
    gamma = 1e-04

  Initial states:
    l = 3516.1772
    s = -149.7033 -76.5829 164.0299 62.2564

  sigma:  285.5654

     AIC     AICc      BIC
1063.653 1065.520 1079.190

$mean
       Qtr1     Qtr2     Qtr3
18 4491.479 4593.242 4352.650

$level
[1] 80 95

$x
     Qtr1   Qtr2   Qtr3   Qtr4
1  3223.5 3674.5 3335.0 2914.0
2  3191.5 3705.0 3214.0 3270.0
3  3589.5 3988.5 3643.5 3665.0
4  4056.0 4213.0 4028.5 3928.5
5  4272.0 3640.5 3259.0 3854.5
6  4163.0 4403.0 3969.0 3537.5
7  3250.0 2838.0 2273.0 2100.5
8  2635.0 2866.0 2835.5 3046.0
9  3496.5 3620.0 3187.5 2970.5
10 3107.0 3402.5 3154.0 3184.0
11 3412.5 3379.0 3156.5 2728.0
12 3516.0 3420.5 3990.5 4196.0
13 4372.5 4501.5 4395.0 4472.0
14 4852.0 4817.5 4531.5 4410.5
15 4777.0 4955.0 4755.5 4797.0
16 4350.5 4617.5 4640.0 4444.0
17 4429.5 4482.5 4373.5 4279.5

$upper
           80%      95%
18 Q1 4857.445 5051.177
18 Q2 5110.419 5384.197
18 Q3 4985.907 5321.132

$lower
           80%      95%
18 Q1 4125.512 3931.781
```

```
18 Q2 4076.064 3802.286
18 Q3 3719.393 3384.167
```

Hide

```
head(forecast_quarter_mase[[i]])
```

```
$model
ETS(M,A,A)

Call:
 ets(y = ts(t(Q_train_row_df[i, ]), frequency = 4), model = train_Quarter_mase_min_df$FittedM
odels[[i]])

  Smoothing parameters:
    alpha = 0.9999
    beta  = 1e-04
    gamma = 1e-04

  Initial states:
    l = 3144.6865
    b = 43.9267
    s = -151.2143 -70.1649 157.7835 63.5957

  sigma:  0.0816

     AIC     AICc      BIC
1073.850 1076.953 1093.825

$mean
       Qtr1     Qtr2     Qtr3
18 4538.077 4675.992 4491.787

$level
[1] 80 95

$x
     Qtr1   Qtr2   Qtr3   Qtr4
1  3223.5 3674.5 3335.0 2914.0
2  3191.5 3705.0 3214.0 3270.0
3  3589.5 3988.5 3643.5 3665.0
4  4056.0 4213.0 4028.5 3928.5
5  4272.0 3640.5 3259.0 3854.5
6  4163.0 4403.0 3969.0 3537.5
7  3250.0 2838.0 2273.0 2100.5
8  2635.0 2866.0 2835.5 3046.0
9  3496.5 3620.0 3187.5 2970.5
10 3107.0 3402.5 3154.0 3184.0
11 3412.5 3379.0 3156.5 2728.0
12 3516.0 3420.5 3990.5 4196.0
13 4372.5 4501.5 4395.0 4472.0
14 4852.0 4817.5 4531.5 4410.5
15 4777.0 4955.0 4755.5 4797.0
16 4350.5 4617.5 4640.0 4444.0
17 4429.5 4482.5 4373.5 4279.5

$upper
           80%      95%
18 Q1 5012.654 5263.881
18 Q2 5358.519 5719.827
18 Q3 5322.234 5761.846

$lower
           80%      95%
```

```
18 Q1 4063.500 3812.273
18 Q2 3993.465 3632.157
18 Q3 3661.340 3221.728
```

We used the test list length created for each series to specify the H value (number of forecasts) to match the accuracy of forecasts on test set.

Checkresiduals loop for ljung-box auto-correlation test for minimum MASE and minimum HQIC models

Loop for shapiro-test on both minimum MASE and minimum HQIC models

Hide

```
Q_model_train_hqic_res_ST <- list()
Q_model_train_mase_res_ST <- list()

for (i in 1:332)
{

  Q_model_train_hqic_res_ST[[i]] <- shapiro.test(Q_model_train_hqic[[i]]$residuals)
  Q_model_train_mase_res_ST[[i]] <- shapiro.test(Q_model_train_mase[[i]]$residuals)

}


head(Q_model_train_mase_res_ST[[i]])
```

```
$statistic
        W 
0.9595856 

$p.value
[1] 0.02696082

$method
[1] "Shapiro-Wilk normality test"

$data.name
[1] "Q_model_train_mase[[i]]$residuals"
```

Hide

```
head(Q_model_train_hqic_res_ST[[i]])
```

```
$statistic
        W 
0.9725243 

$p.value
[1] 0.1379219

$method
[1] "Shapiro-Wilk normality test"

$data.name
[1] "Q_model_train_hqic[[i]]$residuals"
```

Loop for extracting p-values from ljung box and shapiro test for minimum MASE and minimum HQIC models

Hide

```


Q_res_p_hqic_ST <- list()
Q_res_p_hqic <- list()
Q_res_p_mase <- list()
Q_res_p_mase_ST <- list()

for (i in 1:332)
{

  Q_res_p_hqic[[i]] <-  Q_model_train_hqic_res[[i]]$p.value
  Q_res_p_mase[[i]] <-  Q_model_train_mase_res[[i]]$p.value

}


for (i in 1:332)
{

  Q_res_p_hqic_ST[[i]] <-  Q_model_train_hqic_res_ST[[i]]$p.value
  Q_res_p_mase_ST[[i]] <-  Q_model_train_mase_res_ST[[i]]$p.value

}

Q_res_p__hqic_df <- ldply (Q_res_p_hqic, data.frame)
Q_res_p_hqic_ST_df <- ldply (Q_res_p_hqic_ST, data.frame)
Q_res_p__mase_df <- ldply (Q_res_p_mase, data.frame)
Q_res_p_mase_ST_df <- ldply (Q_res_p_mase_ST, data.frame)

Q_res_p__hqic_df$series <- seq.int(nrow(Q_res_p__hqic_df))
Q_res_p_hqic_ST_df$series <- seq.int(nrow(Q_res_p_hqic_ST_df))
Q_res_p__mase_df$series <- seq.int(nrow(Q_res_p__mase_df))
Q_res_p_mase_ST_df$series <- seq.int(nrow(Q_res_p_mase_ST_df))

names(Q_res_p__hqic_df)[names(Q_res_p__hqic_df) == "X..i.."] <- "p"
names(Q_res_p_hqic_ST_df)[names(Q_res_p_hqic_ST_df) == "X..i.."] <- "p"
names(Q_res_p__mase_df)[names(Q_res_p__mase_df) == "X..i.."] <- "p"
names(Q_res_p_mase_ST_df)[names(Q_res_p_mase_ST_df) == "X..i.."] <- "p"

head(Q_res_p_hqic[[i]])
```

```
[1] 0.2353526
```

Hide

```
head(Q_res_p_mase[[i]])
```

```
[1] 0.02919749
```

Hide

```
head(Q_res_p_hqic_ST[[i]])
```

```
[1] 0.1379219
```

Hide

```
head(Q_res_p_mase_ST[[i]])
```

```
[1] 0.02696082
```

Since there are too many plots and outputs to individually verify the plots for each series, we filtered out the P-values from Shapiro test and Ljung box test.

Printing pass or fail results based on 0.05 significance level for minimum MASE and minimum HQIC models

Hide

```
Q_res_p__hqic_df$outcome <- ifelse(
  (
    Q_res_p__hqic_df$p > 0.05
  ),
  "pass",
  "fail"
)


Q_res_p_hqic_ST_df$outcome <- ifelse(
  (
    Q_res_p_hqic_ST_df$p > 0.05
  ),
  "pass",
  "fail"
)


Q_res_p__mase_df$outcome <- ifelse(
  (
    Q_res_p__mase_df$p > 0.05
  ),
  "pass",
  "fail"
)


Q_res_p_mase_ST_df$outcome <- ifelse(
  (
    Q_res_p_mase_ST_df$p > 0.05
  ),
  "pass",
  "fail"
)
```

Printing pass or fail results for Shapiro-test and Ljung box test on 0.05 level of significance

Sorting results of Shapiro-test and Ljung-box test for minimum MASE and minimum HQIC models

Hide

```
Q_final_result_hqic <- sqldf("Select outcome,count(*) from Q_res_p__hqic_df group by outcome"
);
Q_final_result_hqic_ST <- sqldf("Select outcome,count(*) from Q_res_p_hqic_ST_df group by out
come");
Q_final_result_mase <- sqldf("Select outcome,count(*) from Q_res_p__mase_df group by outcome"
);
Q_final_result_mase_ST <- sqldf("Select outcome,count(*) from Q_res_p_mase_ST_df group by out
come");

head(Q_final_result_hqic)
```

outcome	count(*)
<chr>	<int>

	outcome <chr>	count(*) <int>
1	fail	130
2	pass	202

2 rows

Hide

```
head(Q_final_result_hqic_ST)
```

	outcome <chr>	count(*) <int>
1	fail	100
2	pass	232

2 rows

Hide

```
head(Q_final_result_mase)
```

	outcome <chr>	count(*) <int>
1	fail	186
2	pass	146

2 rows

Hide

```
head(Q_final_result_mase_ST)
```

	outcome <chr>	count(*) <int>
1	fail	124
2	pass	208

2 rows

Hide

```
NA
```

- As per above results, we will not be calculating MASE on train and test of models based on minimum MASE values

Holt’s test for both multiplicative and additive seasonality

Hide

```
Q_holt_test_mult <- list()
Q_holt_test_additive <- list()

for(i in 1:332)
{
  Q_holt_test_mult[[i]] <- hw(ts(t(Q_train_row_df[i,]),frequency = 12),damped = TRUE, seasona
l = "multiplicative", initial = "optimal",h =  Q_smp_size_test_list[[i]])
  Q_holt_test_additive[[i]] <- hw(ts(t(Q_train_row_df[i,]),frequency = 12),damped=TRUE,lambda
= "auto" ,seasonal = "additive", initial = "optimal",h =  Q_smp_size_test_list[[i]])
}


head(Q_holt_test_mult[[i]])
```

```
$model
Damped Holt-Winters' multiplicative method

Call:
 hw(y = ts(t(Q_train_row_df[i, ]), frequency = 12), h = Q_smp_size_test_list[[i]],

 Call:
     seasonal = "multiplicative", damped = TRUE, initial = "optimal")

  Smoothing parameters:
    alpha = 0.9987
    beta  = 1e-04
    gamma = 3e-04
    phi   = 0.9753

  Initial states:
    l = 3342.9285
    b = 28.1552
    s = 1.0154 1.0342 1.0893 1.0511 0.9454 0.9392
           0.9939 1.0016 0.9353 0.9686 1.0287 0.9974

  sigma:  0.0878

     AIC     AICc      BIC
1089.767 1103.726 1129.718

$mean
       Sep      Oct      Nov
6 4763.339 4941.864 4696.894

$level
[1] 80 95

$x
     Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec
1 3223.5 3674.5 3335.0 2914.0 3191.5 3705.0 3214.0 3270.0 3589.5 3988.5 3643.5 3665.0
2 4056.0 4213.0 4028.5 3928.5 4272.0 3640.5 3259.0 3854.5 4163.0 4403.0 3969.0 3537.5
3 3250.0 2838.0 2273.0 2100.5 2635.0 2866.0 2835.5 3046.0 3496.5 3620.0 3187.5 2970.5
4 3107.0 3402.5 3154.0 3184.0 3412.5 3379.0 3156.5 2728.0 3516.0 3420.5 3990.5 4196.0
5 4372.5 4501.5 4395.0 4472.0 4852.0 4817.5 4531.5 4410.5 4777.0 4955.0 4755.5 4797.0
6 4350.5 4617.5 4640.0 4444.0 4429.5 4482.5 4373.5 4279.5

$upper
           80%      95%
Sep 6 5299.614 5583.501
Oct 6 5729.308 6146.156
Nov 6 5614.628 6100.447

$lower
           80%      95%
Sep 6 4227.063 3943.176
Oct 6 4154.421 3737.573
Nov 6 3779.160 3293.340
```

Hide

```
head(Q_holt_test_additive[[i]])
```

```
$model
Damped Holt-Winters' additive method

Call:
 hw(y = ts(t(Q_train_row_df[i, ]), frequency = 12), h = Q_smp_size_test_list[[i]], 

 Call:
     seasonal = "additive", damped = TRUE, initial = "optimal", 

 Call:
     lambda = "auto")

  Box-Cox transformation: lambda= 1.9999 

  Smoothing parameters:
    alpha = 0.9971 
    beta  = 1e-04 
    gamma = 3e-04 
    phi   = 0.98 

  Initial states:
    l = 5536336.7627 
    b = 109718.5996 
    s = 151398 450995.7 1229095 616932.4 -896173.9 -1051308
           -89351.92 491076.1 -714314.1 -450418.8 379668.3 -117598.6

  sigma:  1051010

     AIC     AICc      BIC 
2189.040 2202.999 2228.991 

$mean
       Sep      Oct      Nov
6 4626.045 4762.244 4601.545

$level
[1] 80 95

$x
     Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec
1 3223.5 3674.5 3335.0 2914.0 3191.5 3705.0 3214.0 3270.0 3589.5 3988.5 3643.5 3665.0
2 4056.0 4213.0 4028.5 3928.5 4272.0 3640.5 3259.0 3854.5 4163.0 4403.0 3969.0 3537.5
3 3250.0 2838.0 2273.0 2100.5 2635.0 2866.0 2835.5 3046.0 3496.5 3620.0 3187.5 2970.5
4 3107.0 3402.5 3154.0 3184.0 3412.5 3379.0 3156.5 2728.0 3516.0 3420.5 3990.5 4196.0
5 4372.5 4501.5 4395.0 4472.0 4852.0 4817.5 4531.5 4410.5 4777.0 4955.0 4755.5 4797.0
6 4350.5 4617.5 4640.0 4444.0 4429.5 4482.5 4373.5 4279.5                            

$upper
           80%      95%
Sep 6 4908.754 5052.012
Oct 6 5146.436 5338.636
Nov 6 5082.776 5319.933

$lower
           80%      95%
Sep 6 4324.896 4156.656
```

```
Oct 6 4344.208 4105.722
Nov 6 4063.725 3747.911
```

- Created and run a loop for Holts additive and multiplicative models
- Applied damped trend for both models and applied transformation on additive models
- We tried a combination of all other parameters. However, above parameters set were providing better results

Ljung-box test to check auto-correlation in both additive and multiplicative models

Hide

```
Holt_Q_model_train_res_mult <- list()
Holt_Q_model_train_res_add <- list()

for (i in 1:332)
{

  Holt_Q_model_train_res_mult[[i]] <- Box.test(resid(Q_holt_test_mult[[i]]),type="Ljung",lag
 = 10)
  Holt_Q_model_train_res_add[[i]] <- Box.test(resid(Q_holt_test_additive[[i]]),type="Ljung",l
ag = 10)

}


Holt_Q_res_p_mul <- list()
Holt_Q_res_p_add <- list()

head(Holt_Q_model_train_res_mult)
```

```
[[1]]

	Box-Ljung test

data:  resid(Q_holt_test_mult[[i]])
X-squared = 18.159, df = 10, p-value = 0.05233


[[2]]

	Box-Ljung test

data:  resid(Q_holt_test_mult[[i]])
X-squared = 2.519, df = 10, p-value = 0.9906


[[3]]

	Box-Ljung test

data:  resid(Q_holt_test_mult[[i]])
X-squared = 39.866, df = 10, p-value = 1.789e-05


[[4]]

	Box-Ljung test

data:  resid(Q_holt_test_mult[[i]])
X-squared = 14.149, df = 10, p-value = 0.1663


[[5]]

	Box-Ljung test

data:  resid(Q_holt_test_mult[[i]])
X-squared = 16.24, df = 10, p-value = 0.09297


[[6]]

	Box-Ljung test

data:  resid(Q_holt_test_mult[[i]])
X-squared = 20.885, df = 10, p-value = 0.02191
```

Hide

```
head(Holt_Q_model_train_res_add)
```

```
[[1]]

	Box-Ljung test

data:  resid(Q_holt_test_additive[[i]])
X-squared = 10.609, df = 10, p-value = 0.3888


[[2]]

	Box-Ljung test

data:  resid(Q_holt_test_additive[[i]])
X-squared = 7.6022, df = 10, p-value = 0.6676


[[3]]

	Box-Ljung test

data:  resid(Q_holt_test_additive[[i]])
X-squared = 12.655, df = 10, p-value = 0.2436


[[4]]

	Box-Ljung test

data:  resid(Q_holt_test_additive[[i]])
X-squared = 9.0805, df = 10, p-value = 0.5245


[[5]]

	Box-Ljung test

data:  resid(Q_holt_test_additive[[i]])
X-squared = 9.9195, df = 10, p-value = 0.4476


[[6]]

	Box-Ljung test

data:  resid(Q_holt_test_additive[[i]])
X-squared = 5.215, df = 10, p-value = 0.8764
```

- We decided that lag 10 would be optimal for this series based on trail and error

Extracting p-value from Ljung box test for both additive and multiplicative models

Hide

```
for(i in 1:332)
{
  Holt_Q_res_p_mul[[i]] <- Holt_Q_model_train_res_mult[[i]]$p.value
  Holt_Q_res_p_add[[i]] <- Holt_Q_model_train_res_add[[i]]$p.value
}


Holt_Q_res_p_mul_df <- ldply (Holt_Q_res_p_mul, data.frame)
Holt_Q_res_p_add_df <- ldply (Holt_Q_res_p_add, data.frame)

Holt_Q_res_p_mul_df$series <- seq.int(nrow(Holt_Q_res_p_mul_df))
Holt_Q_res_p_add_df$series <- seq.int(nrow(Holt_Q_res_p_add_df))

names(Holt_Q_res_p_add_df)[names(Holt_Q_res_p_add_df) == "X..i.."] <- "p"
names(Holt_Q_res_p_mul_df)[names(Holt_Q_res_p_mul_df) == "X..i.."] <- "p"
```

Printing pass or fail with 0.05 significance for Ljung box test

Hide

```
Holt_Q_res_p_mul_df$outcome <- ifelse(
  (
    Holt_Q_res_p_mul_df$p > 0.05
  ),
  "pass",
  "fail"
)


Holt_Q_res_p_add_df$outcome <- ifelse(
  (
    Holt_Q_res_p_add_df$p > 0.05
  ),
  "pass",
  "fail"
)



Holt_Q_final_result_add <- sqldf("Select outcome,count(*) from Holt_Q_res_p_add_df group by o
utcome");
Holt_Q_final_result_mult <- sqldf("Select outcome,count(*) from Holt_Q_res_p_mul_df group by
 outcome");

head(Holt_Q_final_result_add)
```

	outcome <chr>	count(*) <int>
1	fail	122
2	pass	210

2 rows

Hide

```
head(Holt_Q_final_result_mult)
```

	outcome <chr>	count(*) <int>
1	fail	65
2	pass	267

2 rows

Hide

```
NA
NA
```

Multiplicative model has twice the better results than additive, let’s check if it’s the same with Shapiro test

Shapiro test to check residuals for additive and multiplicative models

Hide

```
Holt_Q_model_train_res_ST_add <- list()
Holt_Q_model_train_res_ST_mul <- list()

for (i in 1:332)
{

  Holt_Q_model_train_res_ST_mul[[i]] <- shapiro.test(Q_holt_test_mult[[i]]$residuals)
  Holt_Q_model_train_res_ST_add[[i]] <- shapiro.test(Q_holt_test_additive[[i]]$residuals)

}

Holt_Q_res_p_mul_ST <- list()
Holt_Q_res_p_add_ST <- list()

head(Holt_Q_model_train_res_ST_mul)
```

```
[[1]]

	Shapiro-Wilk normality test

data:  Q_holt_test_mult[[i]]$residuals
W = 0.98753, p-value = 0.9903


[[2]]

	Shapiro-Wilk normality test

data:  Q_holt_test_mult[[i]]$residuals
W = 0.9847, p-value = 0.9727


[[3]]

	Shapiro-Wilk normality test

data:  Q_holt_test_mult[[i]]$residuals
W = 0.9781, p-value = 0.8838


[[4]]

	Shapiro-Wilk normality test

data:  Q_holt_test_mult[[i]]$residuals
W = 0.91256, p-value = 0.0534


[[5]]

	Shapiro-Wilk normality test

data:  Q_holt_test_mult[[i]]$residuals
W = 0.97835, p-value = 0.8884


[[6]]

	Shapiro-Wilk normality test

data:  Q_holt_test_mult[[i]]$residuals
W = 0.95014, p-value = 0.3179
```

Hide

```
head(Holt_Q_model_train_res_ST_add)
```

```
[[1]]

	Shapiro-Wilk normality test

data:  Q_holt_test_additive[[i]]$residuals
W = 0.96607, p-value = 0.6205


[[2]]

	Shapiro-Wilk normality test

data:  Q_holt_test_additive[[i]]$residuals
W = 0.8931, p-value = 0.02168


[[3]]

	Shapiro-Wilk normality test

data:  Q_holt_test_additive[[i]]$residuals
W = 0.85019, p-value = 0.003421


[[4]]

	Shapiro-Wilk normality test

data:  Q_holt_test_additive[[i]]$residuals
W = 0.74317, p-value = 7.31e-05


[[5]]

	Shapiro-Wilk normality test

data:  Q_holt_test_additive[[i]]$residuals
W = 0.93553, p-value = 0.16


[[6]]

	Shapiro-Wilk normality test

data:  Q_holt_test_additive[[i]]$residuals
W = 0.98248, p-value = 0.9502
```

Extracting p-value from shapiro test for Holt’s additive and multiplicative models

Hide

```
for(i in 1:332)
{
  Holt_Q_res_p_mul_ST[[i]] <- Holt_Q_model_train_res_ST_mul[[i]]$p.value
  Holt_Q_res_p_add_ST[[i]] <- Holt_Q_model_train_res_ST_add[[i]]$p.value
}


Holt_Q_res_p_mul_ST_df <- ldply (Holt_Q_res_p_mul_ST, data.frame)
Holt_Q_res_p_add_ST_df <- ldply (Holt_Q_res_p_add_ST, data.frame)

Holt_Q_res_p_mul_ST_df$series <- seq.int(nrow(Holt_Q_res_p_mul_ST_df))
Holt_Q_res_p_add_ST_df$series <- seq.int(nrow(Holt_Q_res_p_add_ST_df))

names(Holt_Q_res_p_mul_ST_df)[names(Holt_Q_res_p_mul_ST_df) == "X..i.."] <- "p"
names(Holt_Q_res_p_add_ST_df)[names(Holt_Q_res_p_add_ST_df) == "X..i.."] <- "p"
```

Printing pass or fail results of Shapiro test based on 0.05 significance level

Hide

```
Holt_Q_res_p_mul_ST_df$outcome <- ifelse(
  (
    Holt_Q_res_p_mul_ST_df$p > 0.05
  ),
  "pass",
  "fail"
)


Holt_Q_res_p_add_ST_df$outcome <- ifelse(
  (
    Holt_Q_res_p_add_ST_df$p > 0.05
  ),
  "pass",
  "fail"
)


Holt_Q_final_result_add_ST <- sqldf("Select outcome,count(*) from Holt_Q_res_p_add_ST_df grou
p by outcome");
Holt_Q_final_result_mult_ST <- sqldf("Select outcome,count(*) from Holt_Q_res_p_mul_ST_df gro
up by outcome");

head(Holt_Q_final_result_add_ST)
```

	outcome <chr>	count(*) <int>
1	fail	83
2	pass	249

2 rows

Hide

```
head(Holt_Q_final_result_mult_ST)
```

	outcome <chr>	count(*) <int>
1	fail	108
2	pass	224

2 rows

Hide

```
NA
NA
```

- Additive model has performed better in number of normal residuals than multiplicative

Converting train, test and forecasts from models into vector to run MASE.forecast

Hide

```
Quarterly_Test <- read_excel("C:/Users/Mohammed/Desktop/Sem 3/Forecasting/Project/Quarterly_T
est.xlsx")
Quarterly_Test <- Quarterly_Test[order(Quarterly_Test$N),]

Q_vec_test <- unlist(Quarterly_Test[,6:9])
Q_vect_test <- na.omit(Q_vec_test)

Q_vec_train <- unlist(Q_train_row_df[,1:50])
Q_vec_train <- na.omit(Q_vec_train)


list_fitted_train_hqic_quarter <- list()
forecast_Q_hqic_ets <- list()
forecast_Q_holt_mult <- list()
list_fitted_train_mult_quarter <- list()
forecast_Q_holt_additive <- list()
list_fitted_train_add_quarter <- list()


for(i in 1:332)
{

##Forecast values

forecast_Q_hqic_ets[[i]] <- forecast_quarter_hqic[[i]]$mean
forecast_Q_holt_mult[[i]] <- Q_holt_test_mult[[i]]$mean
forecast_Q_holt_additive[[i]] <- Q_holt_test_additive[[i]]$mean

##Fitted

list_fitted_train_hqic_quarter[[i]] <- forecast_quarter_hqic[[i]]$fitted
list_fitted_train_mult_quarter[[i]] <- Q_holt_test_mult[[i]]$fitted
list_fitted_train_add_quarter[[i]] <- Q_holt_test_additive[[i]]$fitted

}


###unlisting/converting to vector, omitting NA values
Q_vect_mean_ets <- na.omit(forecast_Q_hqic_ets)
Q_vect_mean_ets <- unlist(forecast_Q_hqic_ets)
Q_vect_mean_mul <- unlist(forecast_Q_holt_mult)
Q_vect_mean_add <- unlist(forecast_Q_holt_additive)
Q_vect_mean_mul <- na.omit(Q_vect_mean_mul)
Q_vect_mean_add <- na.omit(Q_vect_mean_add)

Q_vect_fitted_ets <- na.omit(list_fitted_train_hqic_quarter)
Q_vect_fitted_ets <- unlist(list_fitted_train_hqic_quarter)
Q_vect_fitted_add <- na.omit(list_fitted_train_add_quarter)
Q_vect_fitted_add <- unlist(list_fitted_train_add_quarter)
Q_vect_fitted_mult <- na.omit(list_fitted_train_mult_quarter)
Q_vect_fitted_mult <- unlist(list_fitted_train_mult_quarter)
```

- We extracted the forecasts(5%) based on the length of each series for ETS (minimum HQIC), Holt’s multiplicative and additive models

- We extracted the fitted values after modelling on the train set for ETS (minimum HQIC), Holt's multiplicative and additive models
- Omitted any NA values from train, test and forecasts data
- Converted train, test and forecasts to vector for running the MASE.forecast function

Calculating MASE for train and test sample

Hide

```
#ETS, Holt Mase forecast for train and test
MASE_ets_hqic_Q_train <- MASE.forecast(Q_vec_train,Q_vec_train,Q_vect_fitted_ets)
MASE_ets_hqic_Q_test <- MASE.forecast(Q_vec_train,Q_vect_test,Q_vect_mean_ets)

MASE_holt_mult_Q_train <- MASE.forecast(Q_vec_train,Q_vec_train,Q_vect_fitted_mult)
MASE_holt_mult_Q_test <- MASE.forecast(Q_vec_train,Q_vect_test,Q_vect_mean_mul)

MASE_holt_add_Q_train <- MASE.forecast(Q_vec_train,Q_vec_train,Q_vect_fitted_add)
MASE_holt_add_Q_test <- MASE.forecast(Q_vec_train,Q_vect_test,Q_vect_mean_add)

##MASE values
list(MASE_ets_hqic_Q_train,MASE_ets_hqic_Q_test,MASE_holt_mult_Q_train,MASE_holt_mult_Q_test,
MASE_holt_add_Q_train,MASE_holt_add_Q_test)
```

```
[[1]]
[1] 1.279816

[[2]]
[1] 1.439306

[[3]]
[1] 1.284082

[[4]]
[1] 1.475372

[[5]]
[1] 1.28987

[[6]]
[1] 1.524166
```

Forecasting - MONTHLY SERIES

Creating train and test lists based on length of each seriesTrain set Month

Hide

```
M_smp_size_list <- list()
M_smp_size_test_list <- list()

for(i in 1:332)
{
  M_smp_size_list[[i]] <- floor(0.95 * M3C_reduced_2019_Month$N[[i]])
  M_smp_size_test_list[[i]] <- floor(0.05 * M3C_reduced_2019_Month$N[[i]])
}

head(M_smp_size_list[[i]])
```

```
[1] 136
```

Loop for 95% of monthly series

Hide

```
M_train_row <- list()
M_test_row <- list()

for(i in 1:332)
{
  M_train_row[[i]] <- M3C_reduced_2019_Month[i ,1:M_smp_size_list[[i]]+5]

}

head(M_train_row[[i]])
```

1 <dbl>	2 <dbl>	3 <dbl>	4 <dbl>	5 <dbl>	6 <dbl>	7 <dbl>	8 <dbl>	9 <dbl>	10 <dbl>
5830	5000	7080	6870	5830	1870	8330	7500	6460	3960

1 row | 1-10 of 136 columns

Hide

```
NA
```

Creating a list of 95% train set for quarterly series

Converting train list to dataframe

Hide

```
M_train_row_df <- ldply (M_train_row, data.frame)

M_train_ts <- list()

for (i in 1:332)
{
  M_train_ts[[i]] <- ts(t(M_train_row_df[i,]), frequency = 12)
}

head(M_train_row_df)
```

	X1 <dbl>	X2 <dbl>	X3 <dbl>	X4 <dbl>	X5 <dbl>	X6 <dbl>	X7 <dbl>	X8 <dbl>	X9 <dbl>
1	2640	2640	2160	4200	3360	2400	3600	1920	4200
2	1680	1920	120	1080	840	1440	480	720	4080
3	1140	720	4860	1200	3150	2130	1800	2010	2880
4	180	940	2040	800	1000	520	500	400	1760
5	2000	1550	4450	3050	3050	2250	2200	2450	4900
6	1200	2850	1350	1500	1950	1950	600	1650	2250

6 rows | 1-10 of 136 columns

Hide

```
NA
NA
```

Possible set of models under ETS for monthly series

Hide

```
models = c("ANN","MNN","MAN","MMN","AAN","MMM","MAM","AAA","MAA","MNM","ANA","MNA")
```

Running GoFVals function for all models

Hide

```
train_Month_mase <- list()

train_Month_mase <- GoFVals(M_train_ts,H=H,models=models)

glimpse(train_Month_mase)
```

```
List of 1
 $ GoF:'data.frame':    3984 obs. of  7 variables:
  ..$ series      : int [1:3984(1d)] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ FittedModels: chr [1:3984(1d)] "ANN" "MNN" "MAN" "MMN" ...
  ..$ AIC         : num [1:3984(1d)] 1238 1238 1240 1240 1243 ...
  ..$ AICc        : num [1:3984(1d)] 1238 1239 1242 1242 1245 ...
  ..$ BIC         : num [1:3984(1d)] 1244 1245 1253 1253 1256 ...
  ..$ HQIC        : num [1:3984(1d)] 1238 1239 1244 1244 1247 ...
  ..$ MASE        : num [1:3984(1d)] 0.651 0.648 0.676 0.7 0.651 ...
```

Running all 12 ETS models using GoFVals function, we will consider only minimum HQIC models as minimum MASE models haven't performed well in both yearly and quarterly series. We will perform residual analysis and auto-correlation on this frequency further.

Converting the list output to dataframe

Hide

```
train_Month_mase_df <- ldply (train_Month_mase, data.frame)
glimpse(train_Month_mase_df)
```

```
Observations: 3,984
Variables: 8
$ .id          [3m[38;5;246m<chr>[39m[23m "GoF", "GoF", "GoF", "GoF", "GoF", "GoF", "Go
F", "GoF", "GoF", "GoF", "GoF", "GoF", "G...
$ series       [3m[38;5;246m<int>[39m[23m <array[37]>
$ FittedModels [3m[38;5;246m<chr>[39m[23m "ANN", "MNN", "MAN", "MMN", "AAN", "MMM", "MA
M", "AAA", "MAA", "MNM", "ANA", "MNA", "A...
$ AIC          [3m[38;5;246m<dbl>[39m[23m <array[37]>
$ AICc         [3m[38;5;246m<dbl>[39m[23m <array[37]>
$ BIC          [3m[38;5;246m<dbl>[39m[23m <array[37]>
$ HQIC         [3m[38;5;246m<dbl>[39m[23m <array[37]>
$ MASE         [3m[38;5;246m<dbl>[39m[23m <array[37]>
```

Filtering models based on minimum HQIC for Monthly series

Hide

```
train_Month_hqic_min_df <- train_Month_mase_df %>% group_by(series) %>% filter(HQIC==min(HQI
C))

head(train_Month_hqic_min_df)
```

.id <chr>	series <int>	FittedModels <chr>	AIC <dbl>	AICc <dbl>	BIC <dbl>	HQIC <dbl>	MASE <dbl>
GoF	1	ANN	1237.983	1238.383	1244.459	1238.350	0.6507851
GoF	2	MAN	1172.583	1174.057	1185.536	1176.501	0.7728911
GoF	3	AAN	1216.770	1217.804	1227.564	1219.504	0.6023830
GoF	4	MAN	1225.018	1226.052	1235.812	1227.752	0.6658561

.id <chr>	series <int>	FittedModels <chr>	AIC <dbl>	AICc <dbl>	BIC <dbl>	HQIC <dbl>	MASE <dbl>
GoF	5	AAN	1286.439	1287.474	1297.234	1289.174	0.6437411
GoF	6	MNN	1190.760	1191.160	1197.237	1191.128	0.6436362

6 rows

Hide

```
NA
```

Running ETS loop for models selected based on minimum HQIC

Hide

```
M_model_train_hqic <- list()

for (i in 1:332)
{

  M_model_train_hqic[[i]] <- ets(ts(t(M_train_row_df[i,]),frequency = 12), model = train_Mont
h_hqic_min_df$FittedModels[[i]])

}

M_model_train_hqic_res <- list()
head(M_model_train_hqic[[i]])
```

```
$loglik
[1] -1346.705

$aic
[1] 2699.411

$bic
[1] 2708.149

$aicc
[1] 2699.593

$mse
[1] 2933939

$amse
[1] 2954058
```

Checkresidual loop for ljung-box auto-correlation test for monthly series

Loop for shapiro-test on minimum HQIC models

Hide

```
M_model_train_hqic_res_ST <- list()

for (i in 1:332)
{

  M_model_train_hqic_res_ST[[i]] <- shapiro.test(M_model_train_hqic[[i]]$residuals)

}

M_res_p_hqic <- list()
M_res_p_hqic_ST <- list()
head(M_model_train_hqic_res_ST[[i]])
```

```
$statistic
        W
0.9896624

$p.value
[1] 0.4099378

$method
[1] "Shapiro-Wilk normality test"

$data.name
[1] "M_model_train_hqic[[i]]$residuals"
```

Loop for extracting p-values from ljung box and shapiro test for minimum HQIC models

Hide

```
for (i in 1:332)
{

  M_res_p_hqic[[i]] <-  M_model_train_hqic_res[[i]]$p.value

}


for (i in 1:332)
{

  M_res_p_hqic_ST[[i]] <-  M_model_train_hqic_res_ST[[i]]$p.value

}

M_res_p__hqic_df <- ldply (M_res_p_hqic, data.frame)
M_res_p_hqic_ST_df <- ldply (M_res_p_hqic_ST, data.frame)

M_res_p__hqic_df$series <- seq.int(nrow(M_res_p__hqic_df))
M_res_p_hqic_ST_df$series <- seq.int(nrow(M_res_p_hqic_ST_df))

names(M_res_p__hqic_df)[names(M_res_p__hqic_df) == "X..i.."] <- "p"
names(M_res_p_hqic_ST_df)[names(M_res_p_hqic_ST_df) == "X..i.."] <- "p"

head(M_res_p__hqic_df)
```

	p <dbl>	series <int>
1	0.60269665	1
2	0.60505849	2
3	0.12568908	3
4	0.55470524	4
5	0.77354549	5
6	0.09762793	6

6 rows

Hide

```
head(M_res_p_hqic_ST_df)
```

	p <dbl>	series <int>
1	8.693511e-04	1
2	6.942509e-05	2
3	1.490086e-01	3
4	5.978780e-03	4

	p <dbl>	series <int>
5	4.826632e-03	5
6	8.596559e-03	6

6 rows

Hide

```
NA
```

Since there are too many plots and outputs to individually verify the plots for each series, we filtered out the P-values from Shapiro test and Ljung box test.

Printing pass or fail results based on 0.05 significance level for minimum HQIC models

Hide

```
M_res_p__hqic_df$outcome <- ifelse(
  (
    M_res_p__hqic_df$p > 0.05
  ),
  "pass",
  "fail"
)


M_res_p_hqic_ST_df$outcome <- ifelse(
  (
    M_res_p_hqic_ST_df$p > 0.05
  ),
  "pass",
  "fail"
)
```

Printing pass or fail results for Shapiro-test and Ljung box test on 0.05 level of significance

Sorting results of Shapiro-test and Ljung-box test for minimum HQIC models

Hide

```
M_final_result_hqic <- sqldf("Select outcome,count(*) from M_res_p__hqic_df group by outcome"
);
M_final_result_hqic_ST <- sqldf("Select outcome,count(*) from M_res_p_hqic_ST_df group by out
come");

head(M_final_result_hqic)
```

	outcome <chr>	count(*) <int>
1	fail	199

	outcome <chr>	count(*) <int>
2	pass	133

2 rows

Hide

```
head(M_final_result_hqic_ST)
```

	outcome <chr>	count(*) <int>
1	fail	166
2	pass	166

2 rows

Hide

```
NA
```

Holt test for both multiplicative and additive seasonality on Monthly series

Hide

```


rm(holt_test_mult_month)
rm(holt_test_additive_month)

holt_test_mult_month <- list()
holt_test_additive_month <- list()


for(i in 1:332)
{
  holt_test_mult_month[[i]] <- hw(ts(t(M_train_row_df[i,]),frequency = 12), seasonal = "multi
plicative", initial = "optimal",h = M_smp_size_test_list[[i]])

  holt_test_additive_month[[i]] <- hw(ts(t(M_train_row_df[i,]),frequency = 12),lambda = "aut
o" ,seasonal = "additive", initial = "optimal",h = M_smp_size_test_list[[i]])

}

head(holt_test_mult_month[[i]])
```

```
$model
Holt-Winters' multiplicative method

Call:
 hw(y = ts(t(M_train_row_df[i, ]), frequency = 12), h = M_smp_size_test_list[[i]],

 Call:
     seasonal = "multiplicative", initial = "optimal")

  Smoothing parameters:
    alpha = 0.0222
    beta  = 1e-04
    gamma = 1e-04

  Initial states:
    l = 6134.5112
    b = -6.6744
    s = 0.8557 1.0673 1.0702 0.7909 1.1213 1.0422
           1.0429 0.9585 0.9348 0.9725 1.0515 1.0921

  sigma:  0.3188

     AIC     AICc      BIC
2724.524 2729.710 2774.039

$mean
        May      Jun      Jul      Aug      Sep      Oct      Nov
12 5350.648 5816.648 5807.834 6242.441 4399.316 5946.789 5925.719

$level
[1] 80 95

$x
     Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
1   5830  5000  7080  6870  5830  1870  8330  7500  6460  3960  8750  6670
2   5000  5420  5210  4170  5210  7500  6670  6460  2500  5830  5420  7710
3   4380  7080  5210  5210  6670  6670  3910  7170  3480  7830  5650  6090
4   6300  6520  3040  5430  8700  4350  8040  4780  7390  7830  7830  3700
5   7610  6520  6090  4350  3040  9570  4780  4780  6090  4350  6960  4130
6   8700  3910  4350  6090  3480  8260  4350  6090  7610  3700  6740  4780
7   7390  6520  8260  2610  6090  6520  6300  5870  3040  5220  5430  2610
8   6960  6090  2170  7390  4130  5430  4570  5000  3040  6300  3480  6090
9   6090  5870  5000  8260  5430  3910  5650  6520  5000  6740  5430  4780
10  4350  6520  7390  7390  5220  7390  8000  8700  3910  4780  7830  3700
11  5290  6740  8700  4780  6960  3700  6960  7610  3040 10000  2830  5220
12  8700  7610  6520  3040

$upper
            80%       95%
May 12 7536.848  8694.152
Jun 12 8193.904  9452.347
Jul 12 8182.148  9439.035
Aug 12 8795.145 10146.466
Sep 12 6198.828  7151.432
Oct 12 8379.983  9668.039
Nov 12 8350.995  9634.858
```

```
$lower
            80%      95%
May 12 3164.448 2007.143
Jun 12 3439.392 2180.948
Jul 12 3433.519 2176.633
Aug 12 3689.736 2338.415
Sep 12 2599.804 1647.200
Oct 12 3513.595 2225.540
Nov 12 3500.443 2216.579
```

Hide

```
head(holt_test_additive_month[[i]])
```

```
$model
Holt-Winters' additive method

Call:
 hw(y = ts(t(M_train_row_df[i, ]), frequency = 12), h = M_smp_size_test_list[[i]],

 Call:
     seasonal = "additive", initial = "optimal", lambda = "auto")

  Box-Cox transformation: lambda= -0.0213

  Smoothing parameters:
    alpha = 0.0012
    beta  = 0.001
    gamma = 1e-04

  Initial states:
    l = 7.9059
    b = -0.0014
    s = -0.1214 0.0887 0.058 -0.1852 0.0579 0.055
           0.0343 -0.038 -0.0208 -0.0336 0.082 0.023

  sigma:  0.2836

     AIC     AICc      BIC
342.3244 347.5108 391.8395

$mean
        May      Jun      Jul      Aug      Sep      Oct      Nov
12 5162.257 5641.512 5796.128 5829.284 4363.596 5853.617 6086.193

$level
[1] 80 95

$x
     Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
1   5830  5000  7080  6870  5830  1870  8330  7500  6460  3960  8750  6670
2   5000  5420  5210  4170  5210  7500  6670  6460  2500  5830  5420  7710
3   4380  7080  5210  5210  6670  6670  3910  7170  3480  7830  5650  6090
4   6300  6520  3040  5430  8700  4350  8040  4780  7390  7830  7830  3700
5   7610  6520  6090  4350  3040  9570  4780  4780  6090  4350  6960  4130
6   8700  3910  4350  6090  3480  8260  4350  6090  7610  3700  6740  4780
7   7390  6520  8260  2610  6090  6520  6300  5870  3040  5220  5430  2610
8   6960  6090  2170  7390  4130  5430  4570  5000  3040  6300  3480  6090
9   6090  5870  5000  8260  5430  3910  5650  6520  5000  6740  5430  4780
10  4350  6520  7390  7390  5220  7390  8000  8700  3910  4780  7830  3700
11  5290  6740  8700  4780  6960  3700  6960  7610  3040 10000  2830  5220
12  8700  7610  6520  3040

$upper
            80%       95%
May 12 8001.165 10107.043
Jun 12 8751.290 11059.562
Jul 12 8993.445 11367.157
Aug 12 9045.414 11433.196
Sep 12 6752.734  8522.877
Oct 12 9083.668 11481.885
```

```
Nov 12 9448.164 11945.047

$lower
            80%      95%
May 12 3344.175 2661.821
Jun 12 3651.645 2905.310
Jul 12 3750.779 2983.790
Aug 12 3772.021 3000.599
Sep 12 2831.131 2255.264
Oct 12 3787.564 3012.879
Nov 12 3936.585 3130.812
```

- Created and run a loop for Holts additive and multiplicative models
- Applied damped trend for both models and applied transformation on additive models
- We tried a combination of all other parameters. However, above parameters set were providing better results

Ljung-box test to check auto-correlation in both additive and multiplicative models

Hide

```
Holt_M_model_train_res_mult <- list()
Holt_M_model_train_res_add <- list()

for (i in 1:332)
{

  Holt_M_model_train_res_mult[[i]] <- Box.test(resid(holt_test_mult_month[[i]]),type="Ljung",
lag = 10)
  Holt_M_model_train_res_add[[i]] <- Box.test(resid(holt_test_additive_month[[i]]),type="Ljun
g",lag = 10)

}


head(Holt_M_model_train_res_mult[[i]])
```

```
$statistic
X-squared
  15.2719

$parameter
df
10

$p.value
[1] 0.1224588

$method
[1] "Box-Ljung test"

$data.name
[1] "resid(holt_test_mult_month[[i]])"
```

Hide

```
head(Holt_M_model_train_res_add[[i]])
```

```
$statistic
X-squared
 14.87966

$parameter
df
10

$p.value
[1] 0.1365113

$method
[1] "Box-Ljung test"

$data.name
[1] "resid(holt_test_additive_month[[i]])"
```

- We decided that lag 10 would be optimal for this series based on trail and error

Extracting p-value from Ljung box test for both additive and multiplicative models

Hide

```


Holt_M_res_p_mul <- list()
Holt_M_res_p_add <- list()

for(i in 1:332)
{
  Holt_M_res_p_mul[[i]] <- Holt_M_model_train_res_mult[[i]]$p.value
  Holt_M_res_p_add[[i]] <- Holt_M_model_train_res_add[[i]]$p.value
}


Holt_M_res_p_mul_df <- ldply (Holt_M_res_p_mul, data.frame)
Holt_M_res_p_add_df <- ldply (Holt_M_res_p_add, data.frame)

Holt_M_res_p_mul_df$series <- seq.int(nrow(Holt_M_res_p_mul_df))
Holt_M_res_p_add_df$series <- seq.int(nrow(Holt_M_res_p_add_df))

names(Holt_M_res_p_mul_df)[names(Holt_M_res_p_mul_df) == "X..i.."] <- "p"
names(Holt_M_res_p_add_df)[names(Holt_M_res_p_add_df) == "X..i.."] <- "p"
```

Printing pass or fail with 0.05 significance for Ljung box test

Hide

```
Holt_M_res_p_mul_df$outcome <- ifelse(
  (
    Holt_M_res_p_mul_df$p > 0.05
  ),
  "pass",
  "fail"
)


Holt_M_res_p_add_df$outcome <- ifelse(
  (
    Holt_M_res_p_add_df$p > 0.05
  ),
  "pass",
  "fail"
)



Holt_M_final_result_add <- sqldf("Select outcome,count(*) from Holt_M_res_p_add_df group by o
utcome");
Holt_M_final_result_mult <- sqldf("Select outcome,count(*) from Holt_M_res_p_mul_df group by
 outcome");


Holt_M_model_train_res_ST_add <- list()
Holt_M_model_train_res_ST_mul <- list()

head(Holt_M_final_result_add)
```

	outcome <chr>	count(*) <int>
1	fail	114
2	pass	218

2 rows

Hide

```
head(Holt_M_final_result_mult)
```

	outcome <chr>	count(*) <int>
1	fail	148
2	pass	184

2 rows

Hide

```
NA
```

- Additive model has performed better in terms of autocorrelation of Standard residuals

Shapiro test to check residuals for additive and multiplicative models

Hide

```
Holt_M_model_train_res_ST_add <- list()
Holt_M_model_train_res_ST_mul <- list()

for (i in 1:332)
{

  Holt_M_model_train_res_ST_add[[i]] <- shapiro.test(holt_test_additive_month[[i]]$residuals)
  Holt_M_model_train_res_ST_mul[[i]] <- shapiro.test(holt_test_mult_month[[i]]$residuals)

}

head(Holt_M_model_train_res_ST_add[[i]])
```

```
$statistic
        W 
0.9661104 

$p.value
[1] 0.001832833

$method
[1] "Shapiro-Wilk normality test"

$data.name
[1] "holt_test_additive_month[[i]]$residuals"
```

Hide

```
head(Holt_M_model_train_res_ST_mul[[i]])
```

```
$statistic
        W 
0.9915047 

$p.value
[1] 0.5851096

$method
[1] "Shapiro-Wilk normality test"

$data.name
[1] "holt_test_mult_month[[i]]$residuals"
```

Extracting p-value from shapiro test for Holt’s additive and multiplicative models

Hide

```
Holt_M_res_p_mul_ST <- list()
Holt_M_res_p_add_ST <- list()

for(i in 1:332)
{
  Holt_M_res_p_mul_ST[[i]] <- Holt_M_model_train_res_ST_mul[[i]]$p.value
  Holt_M_res_p_add_ST[[i]] <- Holt_M_model_train_res_ST_add[[i]]$p.value
}


Holt_M_res_p_mul_ST_df <- ldply (Holt_M_res_p_mul_ST, data.frame)
Holt_M_res_p_add_ST_df <- ldply (Holt_M_res_p_add_ST, data.frame)

Holt_M_res_p_mul_ST_df$series <- seq.int(nrow(Holt_M_res_p_mul_ST_df))
Holt_M_res_p_add_ST_df$series <- seq.int(nrow(Holt_M_res_p_add_ST_df))

names(Holt_M_res_p_mul_ST_df)[names(Holt_M_res_p_mul_ST_df) == "X..i.."] <- "p"
names(Holt_M_res_p_add_ST_df)[names(Holt_M_res_p_add_ST_df) == "X..i.."] <- "p"
```

Printing pass or fail results of Shapiro test based on 0.05 significance level

Hide

```
Holt_M_res_p_mul_ST_df$outcome <- ifelse(
  (
    Holt_M_res_p_mul_ST_df$p > 0.05
  ),
  "pass",
  "fail"
)


Holt_M_res_p_add_ST_df$outcome <- ifelse(
  (
    Holt_M_res_p_add_ST_df$p > 0.05
  ),
  "pass",
  "fail"
)


Holt_M_final_result_add_ST_UD <- sqldf("Select outcome,count(*) from Holt_M_res_p_add_ST_df g
roup by outcome");
Holt_M_final_result_mult_ST_UD <- sqldf("Select outcome,count(*) from Holt_M_res_p_mul_ST_df
 group by outcome");


Mase_holt_mult <- list()
Mase_holt_add <- list()



head(Holt_M_final_result_add_ST_UD)
```

	outcome <chr>	count(*) <int>
1	fail	122
2	pass	210

2 rows

Hide

```
head(Holt_M_final_result_mult_ST_UD)
```

	outcome <chr>	count(*) <int>
1	fail	199
2	pass	133

2 rows

Converting train, test and forecasts from models into vector to run MASE.forecast

Hide

```
Month_Test <- read_excel("C:/Users/Mohammed/Desktop/Sem 3/Forecasting/Project/Month_Test.xls
x")
Month_Test <- Month_Test[order(Month_Test$N),]

M_vec_month_test <- unlist(Month_Test[,6:12])
M_vect_month_test <- na.omit(M_vec_month_test)
M_vec_train_month <- unlist(M_train_row_df[,1:50])
M_vec_train_month <- na.omit(M_vec_train_month)


list_fitted_train_mult_month <- list()
list_fitted_train_add_month <- list()
forecast_M_holt_mult <- list()
forecast_M_holt_additive <- list()


for(i in 1:332)
{

##Forecast values

forecast_M_holt_mult[[i]] <- holt_test_mult_month[[i]]$mean

forecast_M_holt_additive[[i]] <- holt_test_additive_month[[i]]$mean

##Fitted

list_fitted_train_mult_month[[i]] <- holt_test_mult_month[[i]]$fitted

list_fitted_train_add_month[[i]] <- holt_test_additive_month[[i]]$fitted

}

###unlisting/converting to vector, omitting NA values
M_vect_fitted_mult <- unlist(list_fitted_train_mult_month)
M_vect_fitted_mult <- na.omit(M_vect_fitted_mult)
M_vect_fitted_add <- unlist(list_fitted_train_add_month)
M_vect_fitted_add <- na.omit(M_vect_fitted_add)

M_vect_mean_mul <- unlist(forecast_M_holt_mult)
M_vect_mean_add <- unlist(forecast_M_holt_additive)
M_vect_mean_mul <- na.omit(M_vect_mean_mul)
M_vect_mean_add <- na.omit(M_vect_mean_add)
```

- We extracted the forecasts(5%) based on the length of each series for Holt’s multiplicative and additive models
- Didn’t consider ETS model as the Shapiro test and Ljung-box test results were unsatisfactory
- We extracted the fitted values after modelling on the train set for Holt’s multiplicative and additive models
- Omitted any NA values from train, test and forecasts data
- Converted train, test and forecasts to vector for running the MASE.forecast function

Calculating MASE for train and test sample

Hide

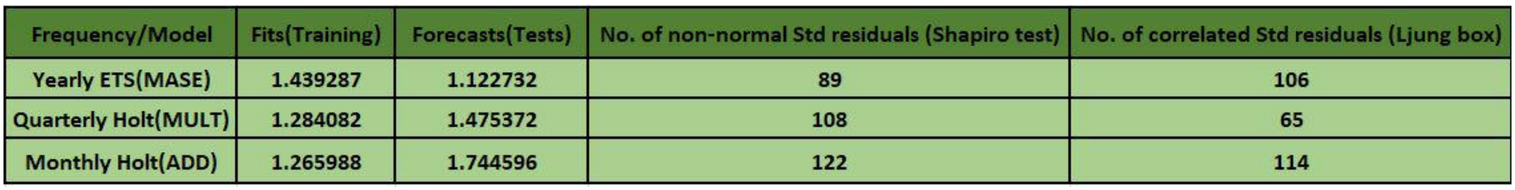
```
#Holt Mase forecast for train
MASE_holt_mult_M_train <- MASE.forecast(M_vec_train_month,M_vec_train_month,M_vect_fitted_mul
t)
MASE_holt_add_M_train <- MASE.forecast(M_vec_train_month,M_vec_train_month,M_vect_fitted_add)

#Holt Mase forecast for test
MASE_holt_mult_M_test <- MASE.forecast(M_vec_train_month,M_vect_month_test,M_vect_mean_mul)
MASE_holt_add_M_test <- MASE.forecast(M_vec_train_month,M_vect_month_test,M_vect_mean_add)

##Listing all the results
list(MASE_holt_mult_M_train, MASE_holt_add_M_train, MASE_holt_mult_M_test, MASE_holt_add_M_te
st)
```

```
[[1]]
[1] 1.259632

[[2]]
[1] 1.265988

[[3]]
[1] 1.703141

[[4]]
[1] 1.744596
```

Conclusion

After all the analysis, we can see that there is no consistency in one single model type for each frequency. If fitted model MASE is good then residuals or auto-correlation aren't providing satisfactory results and vice-versa. We could further try auto.arima or hybrid models to overcome such issues. We have considered Training, Test, Standard residuals and Auto-correlation to finalise the below models, we have also attached excel output of other model results for convenience.

Results



Frequency/Model	Fits(Training)	Forecasts(Tests)	No. of non-normal Std residuals (Shapiro test)	No. of correlated Std residuals (Ljung box)
Yearly ETS(MASE)	1.439287	1.122732	89	106
Quarterly Holt(MULT)	1.284082	1.475372	108	65
Monthly Holt(ADD)	1.265988	1.744596	122	114