FORECASTING COMPETITIVE PROJECT
Math1307 - 303 Data Sets - 5 Students - 1 Goal

Arjun Balaji - s3629999, Irama Navarro Ruiz - s3473295, Li Xie - s3308052, Rupesh Papneja - s3637387, Sarun Sankarankutty Menon - s3642783

11 October 2017

REPORT OUTLINE

- **Introduction**
- **Data Description**
- **Methodology**
- **Findings**
- **Assessment of residuals**
- **Project Goal Summary**
- **Challenges**
- **Limitations**
- **Conclusions & Recommendations**

INTRODUCTION

In this project we build an expert system that analyses programmatically the 303 data sets provided. We use R as a programming tool for all the steps in the process. The expert system analyses each time series and runs various forecasting models from the ETS and Forecasting packages to identify a best model that has the minimum MASE. We then perform residual analysis to check for normality and its serial correlation. We then use the best model for forecasting the next few periods of the series.

DATA DESCRIPTION

The data is from the International Institute of Forecasters (IIF) which runs a competition, called M - Competitions, and is led by forecasting researcher Spyros Makridakis. The aim of the competition is to evaluate and compare the accuracy of forecasting methods over a massive collection of 3003 data series with different characteristics. Today, we will use a slightly different approach for the competition than the original M3 Competition by using a reduced version of 303 data sets.

Table 1: Summary of data type by Category

type	MICRO	INDUSTRY	MACRO	FINANCE	DEMOGRAPHIC
Y	20	27	11	11	32
Q	16	25	20	16	24
M	17	21	24	18	21

METHODOLOGY

Definition of Expert System

As previously described in the introduction, the process consists on automating functionality to run an exhaustive number of models across all series.

We perform these tasks as a 3 step process described below:

Extract, Transform and Load

First of all, we extract, transform and load the data into the several time series objects. The data has various categories including Micro, Macro, Finance, Industry and Demographic, and the data series have different intervals such as yearly, quarterly and monthly. We capture all of these details and transform them into various class objects in R. Then we convert all the series to time series objects and create training and validation data sets. Validation data sets are created from 5% of each series to validate the quality of the forecasts.

Model Building

For model building, we use loops to run all possible permutations of the models and identify the best models with the lowest MASE value. Loops capturing all exponential smoothing models and state space models are designed to avoid discriminating some possible models that could outperform if disregarded. We prefer this approach rather than inspecting the series one by one or running different tests to assess if there was different components such as trend or seasonality then running only the models that those tests resulted in.

The expert systems determines an optimal value of lambda and runs 2 sets of models one with lambda value and one without to then evaluate which one performs best. That way we automatically assess if the series requires or not transformation to reduce the variance and reach a model fit with a lower MASE.

If the best model for each specific series is an ETS model, the expert system also performs some tuning to proof if by imputing parameters it does outperform the non-tuned model; this way trying again to reduce the MASE, our goal.

Model Assessment

Lastly we identify the models with the lowest MASE value for all category types and all interval series. Then we assess the accuracy of the models and perform residual analysis to check for serial correlation. We also use Shapiro Wilk Test to test for normality and LJung Box test for autocorrelation. Lastly, we find the forecast for all and each of the series with the selected best models. For individual model assessment the expert system also allow us to obtain for each model/series combination the residual plots and the forecast for the desired series.

FINDINGS

The following table summarizes the type of models that performed best across all the series. On the table we see that the Automatic ETS model with several variations gets to be one of the best performers across all series.

Overall result set analysis. Most used models overall

The table below shows the models that within each type of series, each frequency, performs the best.

Table 2: Most used Models

description	Y	Q	M
ETS AUTO bic Damped Optimisation criteria lik	NA	2	NA
ETS AUTO bic Optimisation criteria lik	NA	1	NA
ETS AUTO bic Damped Optimisation criteria amse	NA	1	NA
ETS AUTO bic Optimisation criteria amse	1	NA	NA
ETS AUTO bic Damped Optimisation criteria mse	NA	NA	1
ETS AUTO bic Damped Optimisation criteria sigma	1	NA	1
ETS AUTO bic Damped Optimisation criteria mae	5	5	7
ETS AUTO aicc Damped Optimisation criteria mae	NA	2	3
ETS AUTO aic Damped Optimisation criteria mae	NA	1	NA
ETS AUTO bic Optimisation criteria mae	5	2	3
ETS AUTO aicc Optimisation criteria mae	NA	NA	1
SES simple Damped Exponential	2	NA	NA
Holts optimal Damped Exponential	4	NA	NA
Holts simple Exponential	2	NA	NA
Holts optimal Exponential	2	1	NA
ETS MMN Damped Optimisation criteria lik	2	NA	1
ETS AAN Damped Optimisation criteria amse	NA	NA	1
ETS MMN Damped Optimisation criteria amse	2	NA	NA
ETS AAN Damped Optimisation criteria mae	5	NA	1
ETS MMN Damped Optimisation criteria mae	12	4	4
ETS AAN Optimisation criteria mae	5	NA	NA
ETS MAN Optimisation criteria mae	2	NA	NA
ETS MMN Optimisation criteria mae	9	NA	NA
ETS ANN Optimisation criteria mae	1	NA	NA
ETS AUTO bic Damped Optimisation criteria lik Transformed using Box Cox	1	2	NA
ETS AUTO bic Optimisation criteria lik Transformed using Box Cox	NA	1	NA
ETS AUTO bic Damped Optimisation criteria amse Transformed using Box Cox	3	NA	NA
ETS AUTO bic Damped Optimisation criteria mse Transformed using Box Cox	1	2	NA
ETS AUTO bic Optimisation criteria mse Transformed using Box Cox	NA	1	NA
ETS AUTO bic Damped Optimisation criteria mae Transformed using Box Cox	19	9	7
ETS AUTO aicc Damped Optimisation criteria mae Transformed using Box Cox	NA	NA	2
ETS AUTO aic Damped Optimisation criteria mae Transformed using Box Cox	NA	1	NA
ETS AUTO bic Optimisation criteria mae Transformed using Box Cox	6	2	5
ETS AUTO aicc Optimisation criteria mae Transformed using Box Cox	1	1	NA
ETS AUTO aic Optimisation criteria mae Transformed using Box Cox	2	1	2
Holts optimal Transformed using Box Cox	1	NA	NA
ETS AAN Optimisation criteria amse Transformed using Box Cox	1	NA	NA
ETS AAN Optimisation criteria mae Transformed using Box Cox	6	2	1
Holts Winters multiplicative Damped Exponential	NA	3	NA
Holts Winters multiplicative Exponential	NA	1	1
Holts Winters additive Damped	NA	1	1
Holts Winters multiplicative Damped	NA	NA	1
ETS MAM Damped Optimisation criteria lik	NA	1	NA
ETS MAA Optimisation criteria lik	NA	NA	2
ETS AAA Damped Optimisation criteria amse	NA	2	NA
ETS MMM Damped Optimisation criteria amse	NA	1	3
ETS MAA Optimisation criteria amse	NA	NA	1

description	Y	Q	M
ETS MMM Optimisation criteria amse	NA	NA	1
ETS MAA Optimisation criteria mse	NA	NA	1
ETS MMM Damped Optimisation criteria sigma	NA	1	NA
ETS AAA Damped Optimisation criteria mae	NA	2	NA
ETS MAA Damped Optimisation criteria mae	NA	4	NA
ETS MAM Damped Optimisation criteria mae	NA	11	15
ETS MMM Damped Optimisation criteria mae	NA	13	15
ETS AAA Optimisation criteria mae	NA	1	2
ETS MAA Optimisation criteria mae	NA	2	2
ETS ANA Optimisation criteria mae	NA	NA	2
ETS MAM Optimisation criteria mae	NA	1	1
ETS MMM Optimisation criteria mae	NA	4	1
ETS MNM Optimisation criteria mae	NA	2	NA
Holts Winters additive Damped Transformed using Box Cox	NA	1	NA
ETS AAA Damped Optimisation criteria mae Transformed using Box Cox	NA	4	3
ETS AAA Optimisation criteria mae Transformed using Box Cox	NA	5	4
ETS ANA Optimisation criteria mae Transformed using Box Cox	NA	NA	5

Models with normal residuals and no serial correlation

Below we observe the number of the models that have passed both, the Normality and Non-Serial correlation tests. In this regard, we see that a total of 57 yearly series, 44 quarterly and 31 monthly series have normal residuals and no serial correlation. Also important to note, non of our yearly data models has failed both assumptions check in their residuals.

Table 3: Results by Type and Residual Check

sw.count	box.count	Y	Q	M
0	0	57	44	31
0	1	5	2	1
1	0	39	48	52
1	1	NA	7	17

Best Model Analysis

Following we provide some examples of the best models found amongst all the series.

Table 4: Minimum MASE by Type

Y	Q	M
0.1198685	0.0325648	0.019662

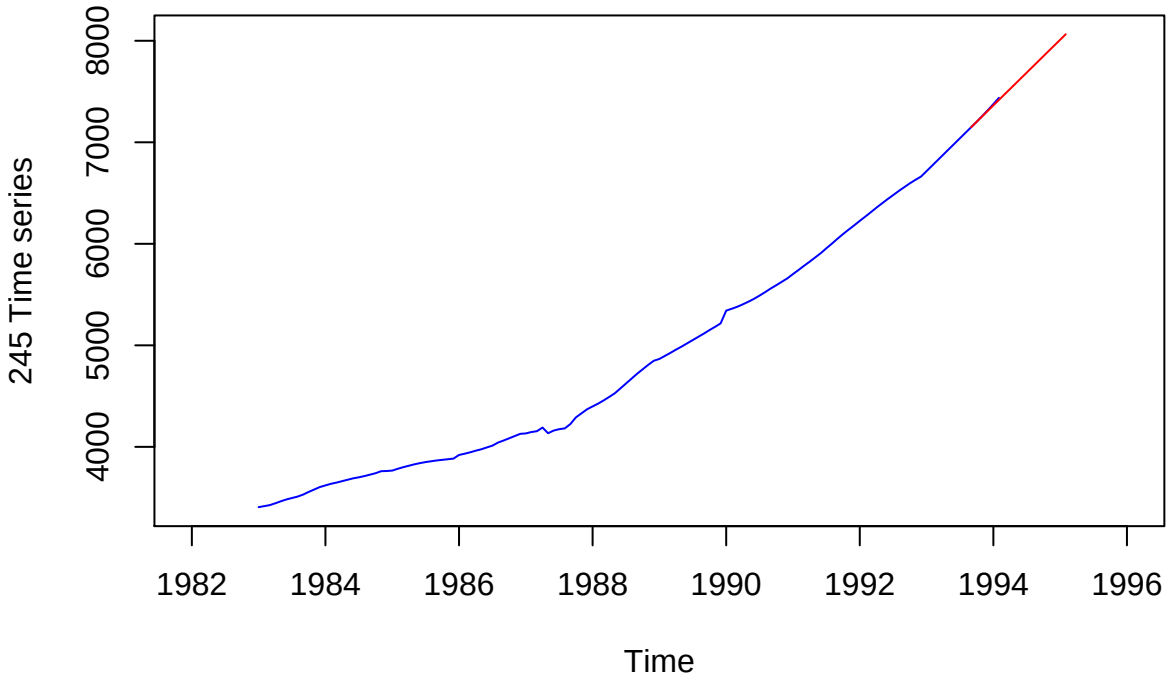
Lowest MASE monthly

MASE = 0.01966198

Corresponds to Series 245 category MACRO The model is an ETS(A,A,N) Additive Errors, Additive Trend and No Seasonality. In particular, the model has been performance tuned and has had no transformation on

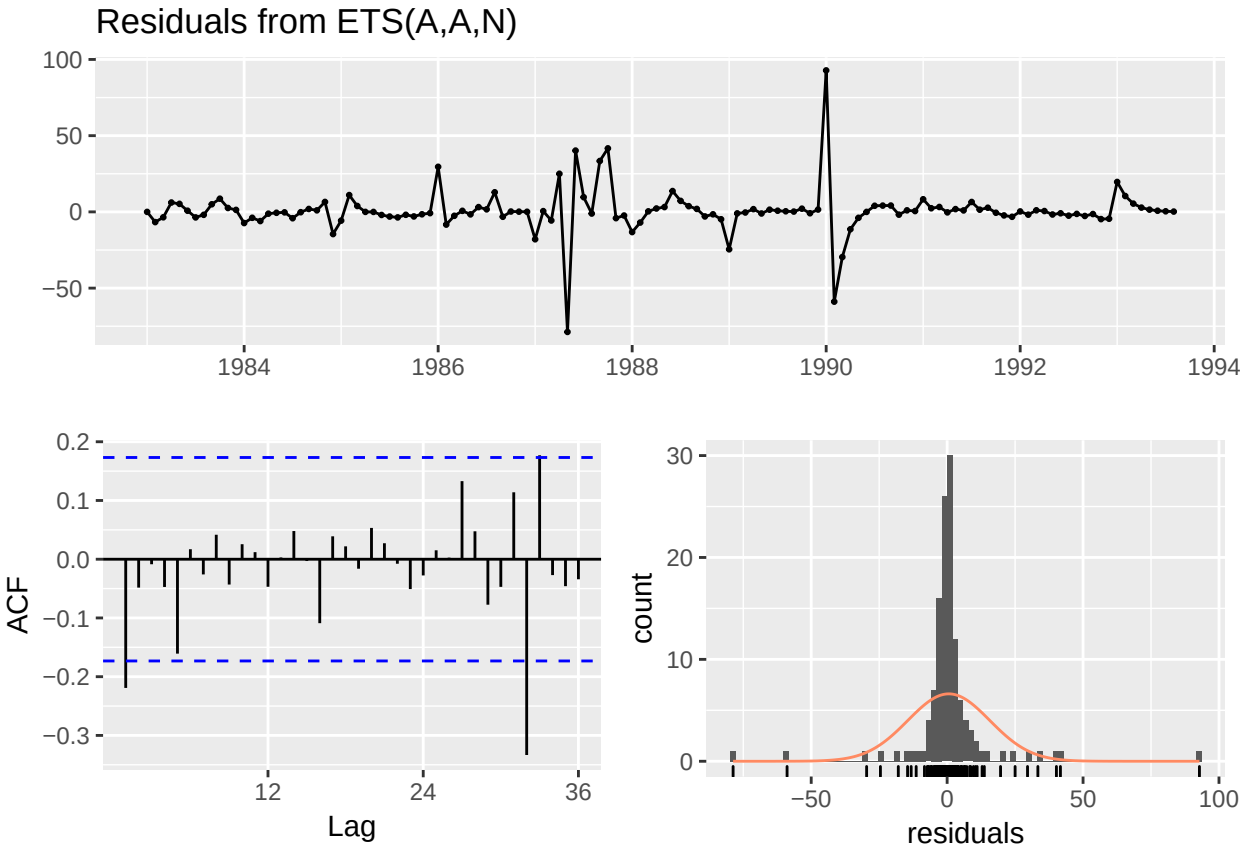
the series (ETS_AUTO_bic_mae_N).

For this example we consider all the Monthly time series data, the below plot shows the best Model based on min MASE value. This is a Non-Linear State Space Model which has best MASE. However, this model shows some residuals that seem to have serial correlation as it is confirmed by not passing the Ljung Box Test. Also if we look at the normality of the residuals we see that the model does not passes the Shapiro Wilk Test but we see visually that the data is almost normal.




```
## ETS(A,A,N)
##
## Call:
##  ets(y = tsobject, model = "ZZZ", damped = xg[i, 2], lambda = lambda,
##
##  Call:
##      opt.crit = toString(xg[i, 3]), ic = toString(xg[i, 1]))
##
##   Smoothing parameters:
##     alpha = 0.9951
##     beta  = 0.4756
##
##   Initial states:
##     l = 3389.2996
##     b = 16.7004
##
##   sigma:  15.0044
##
##      AIC     AICc      BIC
## 1324.396 1324.888 1338.656
```

```
## [1] "ETS(A,A,N)"
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,N)
## Q* = 15, df = 20, p-value = 0.7764
##
## Model df: 4.   Total lags used: 24
```

(Include point forecast and validation points to assess as an example.)

Lowest MASE quarterly.

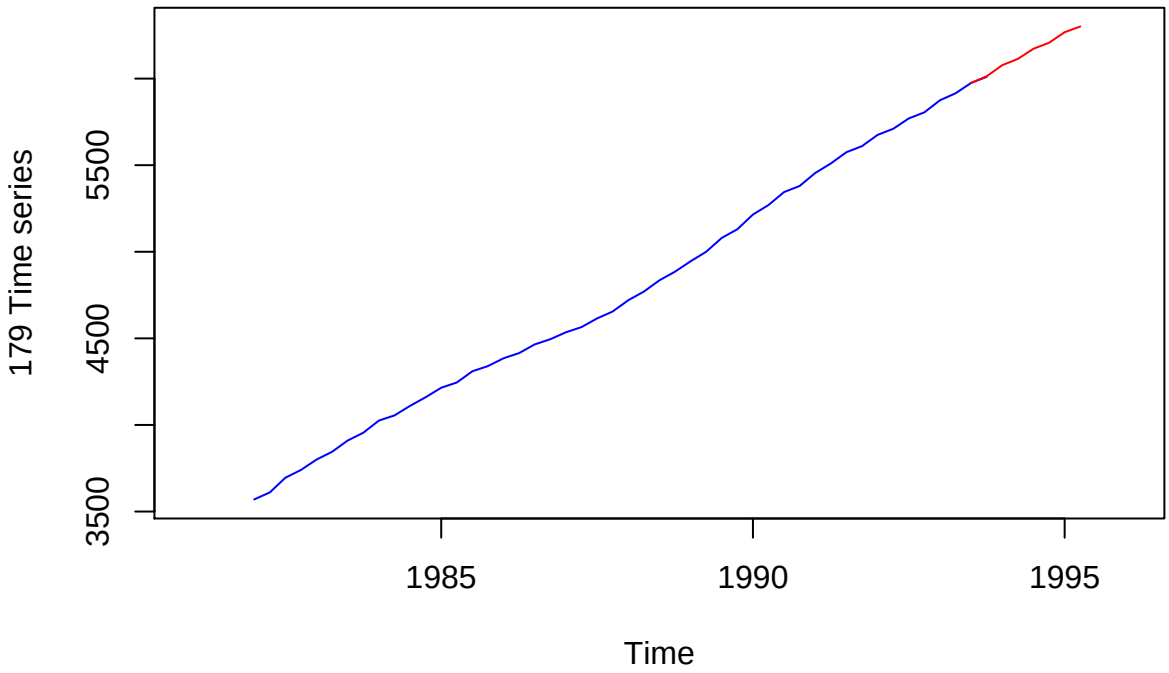
Similar to the Monthly here we look at the best model for the Quarterly data, pick the best Model.

MASE = 0.03256482

Corresponds to Series 179 category DEMOGRAPHIC The model is an ETS(A,Ad,A) Additive Errors, Additive dumped Trend and Additive Seasonality. In particular, the model has been performance tuned and has had lambda transformation on the series ETS_AUTO_bic_damped_mae_Y.

This is a Linear State Space Model which has best MASE and here the residuals are not serially correlated and the normality test is confirmed by passing the Shapiro test.

```
#Best Model Quarterly
generatePlot(179)
```



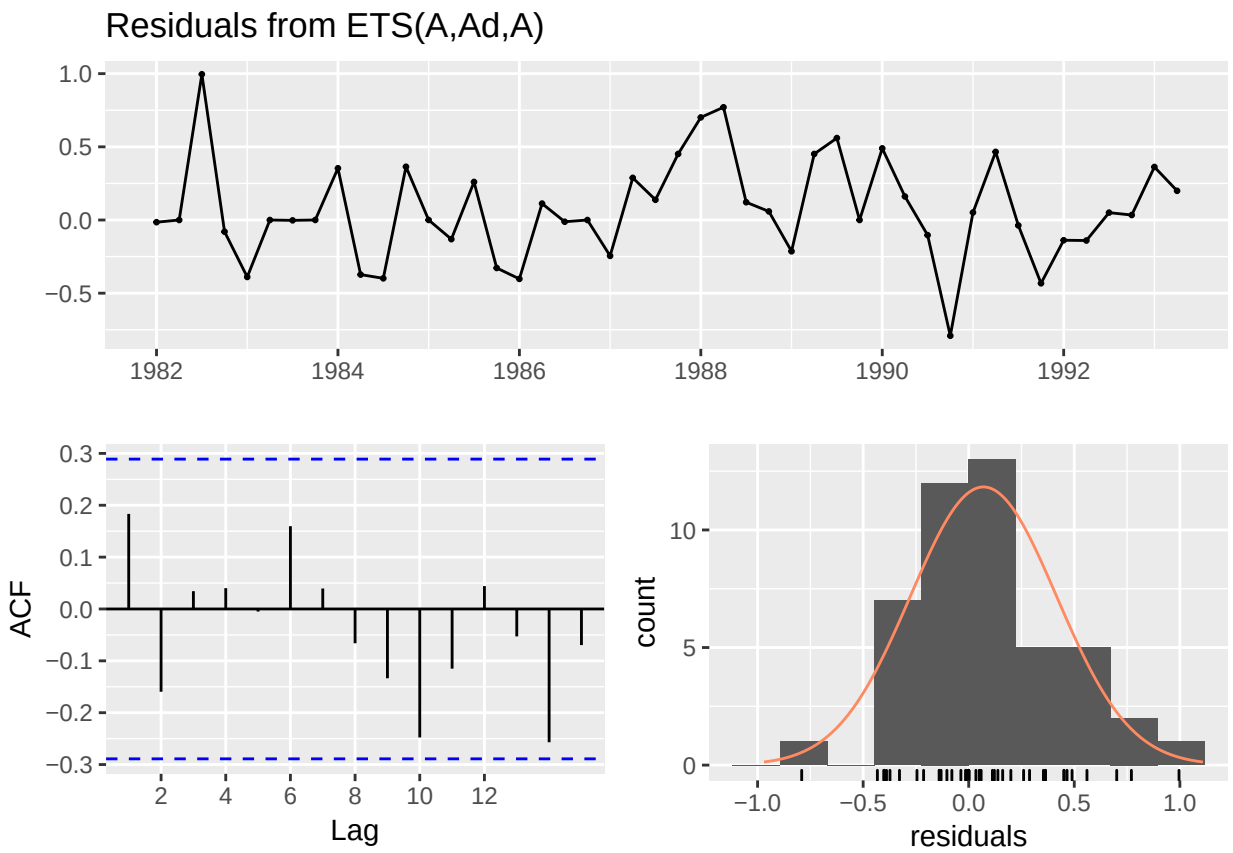

```
getModel(179)
```

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = tsobject, model = "ZZZ", damped = xg[i, 2], lambda = lambda,
##
##  Call:
##      opt.crit = toString(xg[i, 3]), ic = toString(xg[i, 1]))
##
##   Box-Cox transformation: lambda= 0.6114
##
##   Smoothing parameters:
##     alpha = 0.6451
##     beta  = 0.4209
##     gamma = 0.0836
##     phi   = 0.9796
##
##   Initial states:
##     l = 239.0195
##     b = 2.283
##     s=-0.1179 0.1114 -0.2607 0.2673
##
##   sigma:  0.3498
##
##       AIC      AICc       BIC
##  99.47932 105.76503 117.76573
```

```
getMethodForModel(179)
```

```
## [1] "ETS(A,Ad,A)"
```

```
checkresidualsforModel(179)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,Ad,A)
## Q* = 10.614, df = 3, p-value = 0.01401
##
## Model df: 9.   Total lags used: 12
```

Lowest MASE yearly.

Similar to the Monthly and Quarterly here we look at the best model for the Yearly data, pick the best Model.

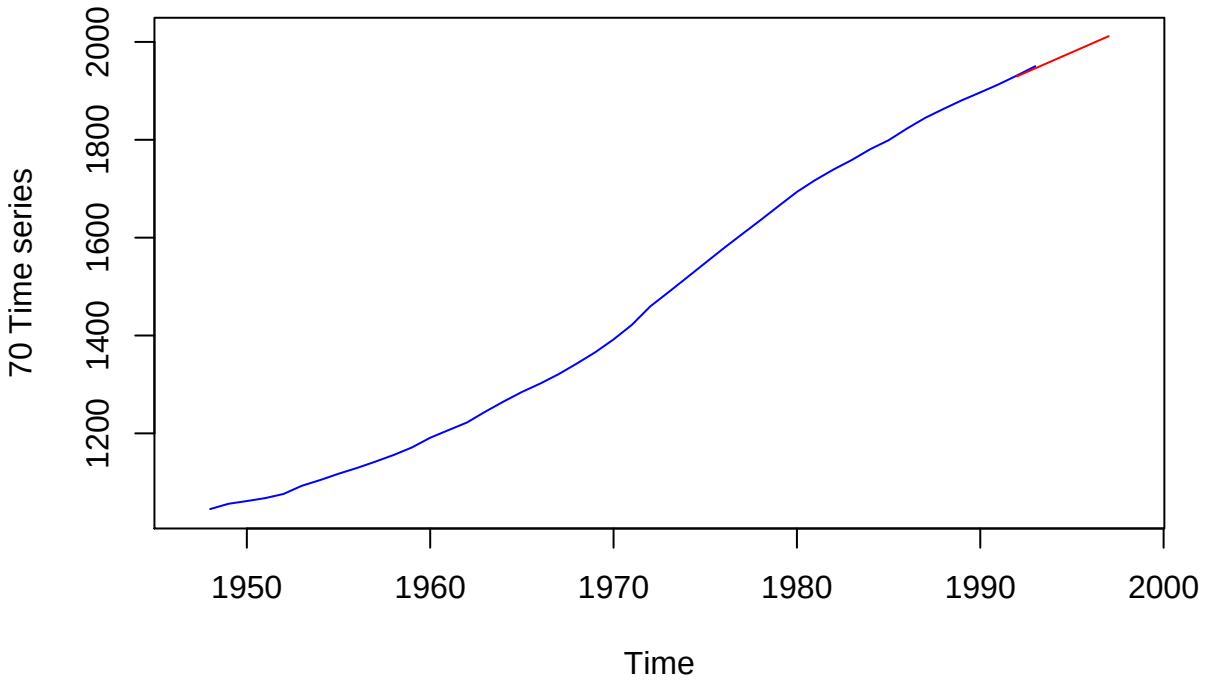
MASE = 0.11986853

Series is 70 of DEMOGRAPHIC data As in out monthly series, this model is an ETS with Additive Error, Additive trend and no seasonality. It has been tuned and the series was not previously transformed (ETS_AUTO_bic_mae_N)

ETS(A,A,N)

The residuals are all normal and they also don't have any significant serial correlation.

```
#Best Model Yearly
generatePlot(70)
```



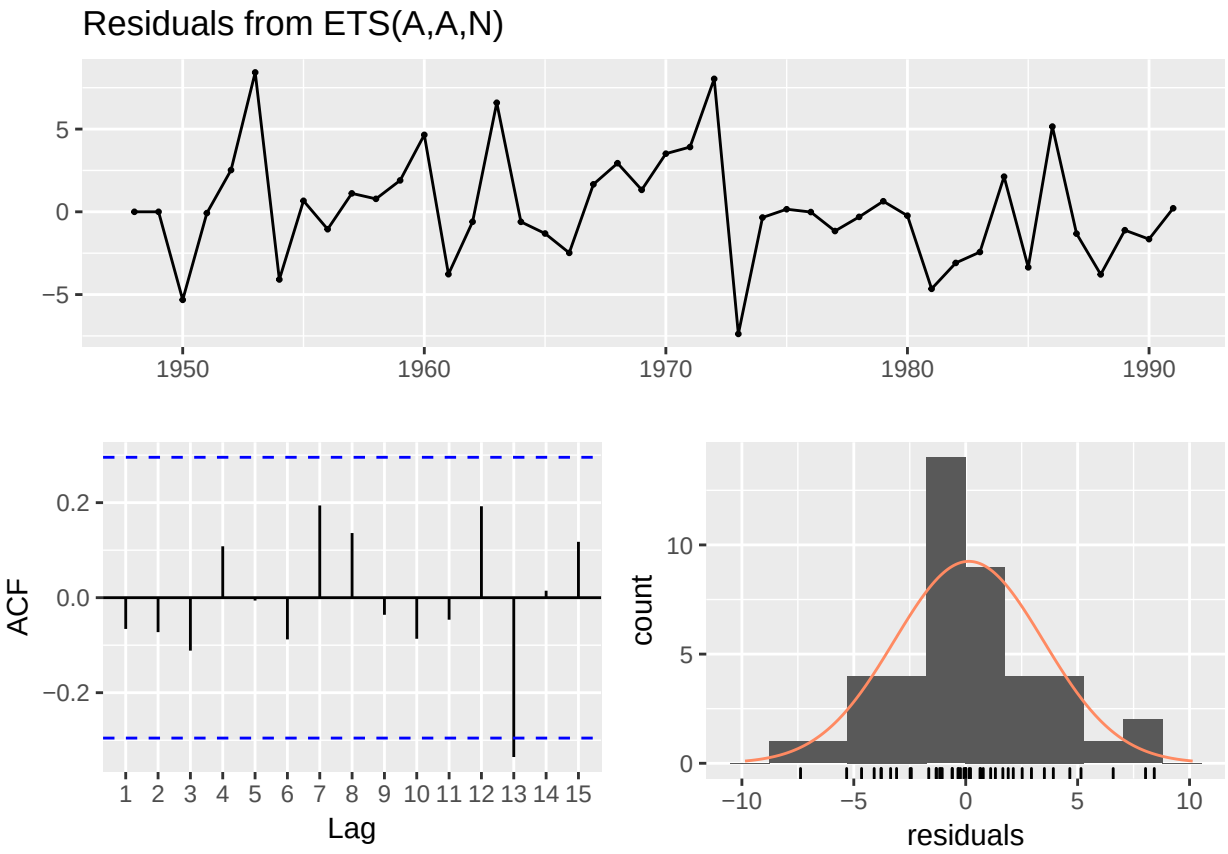

```
getModel(70)
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = tsobject, model = "ZZZ", damped = xg[i, 2], lambda = lambda,
##
##  Call:
##      opt.crit = toString(xg[i, 3]), ic = toString(xg[i, 1]))
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.8993
##
##   Initial states:
##     l = 1034.38
##     b = 10.86
##
##   sigma:  3.2982
##
##      AIC     AICc      BIC
## 281.5225 283.1014 290.4434
```

```
getMethodForModel(70)
```

```
## [1] "ETS(A,A,N)"
```

```
checkresidualsforModel(70)
```



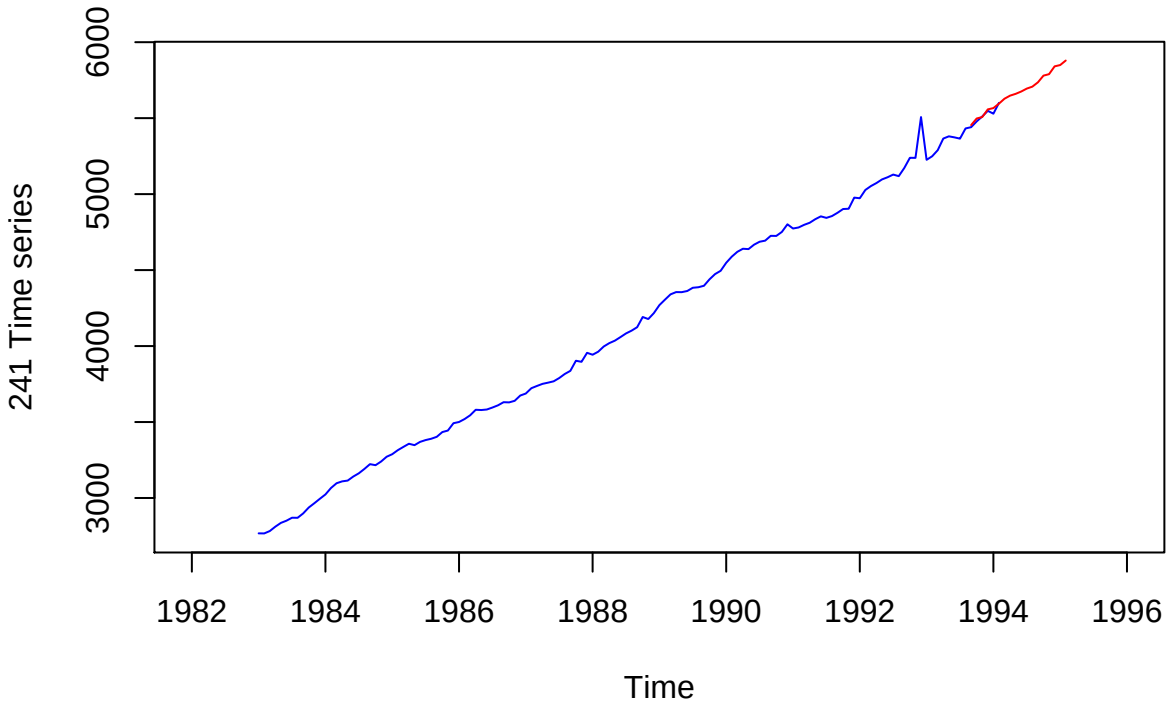

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,N)
## Q* = 5.6965, df = 6, p-value = 0.458
##
## Model df: 4.   Total lags used: 10
```

Example of model with residuals

Previously we saw the findings based on best MASE values. Lastly in our findings it is also worth showing one of the models developed for our series that was identified as not passing the normality or the serial correlation test.

As you can see here there is lot of serial correlation in the first few lags of the data and its skewed as well. This is just shown as an example.

```
generatePlot(241)
```



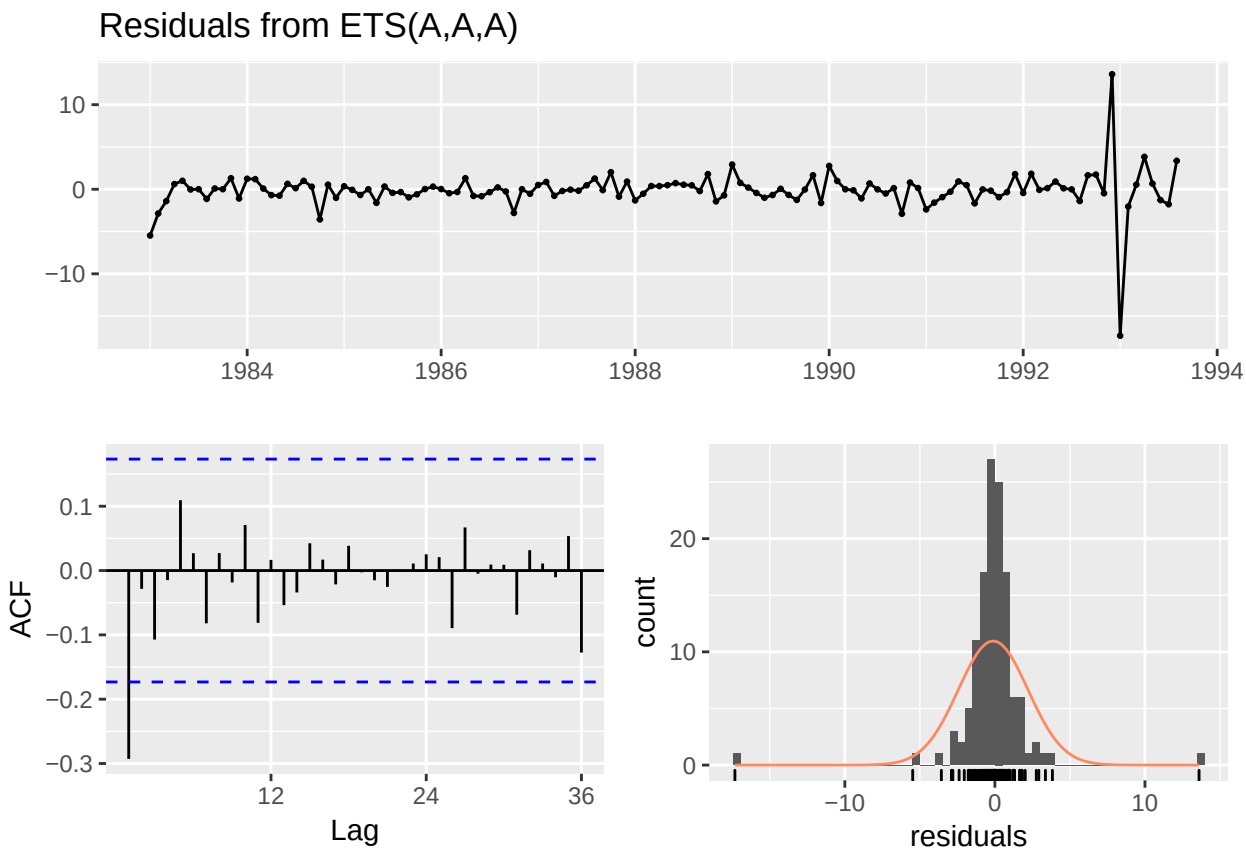

```
getModel(241)
```

```
## ETS(A,A,A)
##
## Call:
##  ets(y = tsobject, model = mt, damped = xg[i, 4], lambda = lambda,
##
##  Call:
##      opt.crit = toString(xg[i, 5]))
##
##   Box-Cox transformation: lambda= 0.6772
##
##   Smoothing parameters:
##     alpha = 0.8925
##     beta  = 0.0461
##     gamma = 1e-04
##
##   Initial states:
##     l = 318.1599
##     b = 2.1279
##     s=1.0578 -0.6247 0.2378 -1.0098 -1.3024 -0.648
##            -0.3797 0.1225 0.847 1.0578 0.5003 0.1413
##
##   sigma:  2.3197
##
##      AIC     AICc      BIC
## 870.4702 876.0338 918.9547
```

```
getMethodForModel(241)
```

```
## [1] "ETS(A,A,A)"
```

```
checkresidualsforModel(241)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,A)
## Q* = 18.819, df = 8, p-value = 0.01586
##
## Model df: 16.   Total lags used: 24
```

PROJECT GOAL SUMMARY

Lastly the table below summarizes our results obtained by the expert System, it shows the MEAN of the Minimum MASE found for each type of series and the number of models that do not satisfy each of the tests.

Table 5: Summary of Results

type	mean_mase	mean_forecast_mase	sum_sw_count	sum_box_count
Y	0.7399062	1.503683	39	5
Q	0.3605476	1.147781	55	9
M	0.3201258	1.390978	69	18

Similarly and by category we observe the Mean of the minimum MASE for each type and category.

Table 6: Summary of Results by Category

type	MICRO	INDUSTRY	MACRO	FINANCE	DEMOGRAPHIC
Y	0.6754783	0.8746450	0.4425730	0.8130020	0.7435693
Q	0.4930968	0.4175691	0.2756067	0.3784192	0.2716536
M	0.5830960	0.5068954	0.1877362	0.2184284	0.1589473

CHALLENGES

The biggest challenges encountered while developing the expert system lied in making an efficient system that ran as many models as possible in short amount of time to then be able to compare them all in terms of MASE. The expert system needed basically to achieve the lowest MASE taking into consideration the important aspects of the data without delaying the performance of the system and affecting the results.

The strategy to follow and the sequence of the steps was paramount to develop an efficient system without overlooking some of the most technical aspects of time series data At the same time, the model needed to include a smooth logic that didn't disregard the inclusion of important models or aspects of the data to find the lowest MASE. Optimization, transformations and residual check assessment needed to be smartly lined up in the system to improve the models, to include as many options as possible to try to decrease the MASE and not interfere in the logic.

LIMITATIONS

This approach focuses on running abundance of models to each series and then assess accuracy. It does not focuses on performing any visual inspection on the series as it would be too time consuming and inefficient to do it one by one for all and each of the series.

The expert system also disregards to perform any seasonality or trend checks for each of the series as it is considered the type of model selected will perform better or worst depending on the specific model characteristics. Specifically, if a model has seasonality and damped trend a model will this characteristics will indeed provide better results than other models that do not have seasonality and damped trend.

CONCLUSION & RECOMMENDATIONS

Based on all of this we choose the best model and perform forecasting. For this exercise the model is selected, residuals analysed and the forecast performed. The assessment of the best models for each of the type series confirms the forecasts are accurate.

The residuals check confirms in each of the inspected models non normality and serial correlation has been accurately identified.

Our whole exercise on a normal computer took about 40 minutes since we are running around 70 models across each of the 303 data sets. The expert system allows to reduce time the time consuming task of visually assess a big number of time series.

As further investigations a different type of forecasting models could be implemented into the expert system since it serves as a comparison tool to asses many models in many different types of series.

EXPERT MODEL

Loading the S3 classes to be used with S4 definitions.

```
.GlobalEnv$.__C__forecast
.GlobalEnv$.__C__ets

setOldClass(c("forecast", "ets"))
```

Defining the classes and methods for time series analysis and forecasting

```
setClass(Class="TClass",
         representation(
           type="character",
           frequency="numeric",
           id="numeric",
           n="numeric",
           diff_n="numeric",
           forecast_period="numeric",
           category="character",
           start_year="numeric",
           start_frequency="numeric",
           superposition="numeric",
           lambda="numeric",
           original_data="numeric",
           validation="numeric",
           ts_complete="ts",
           ts_training="ts",
           models_df="data.frame"
      )
)

setClass(Class="RClass",
         representation(
           key="character",
           lambda="character",
           mase="numeric",
           aic="numeric",
           bic="numeric",
           aicc="numeric",
           swtest.pvalue="numeric", # check shapiro-wilk test value
           swtest.count="numeric",
           boxtest.pvalue="numeric",
           boxtest.count="numeric",
           ts_forecast="ts", # forecast time series
           forecast.model="forecast",
           ets.model="ets"
         ),
         prototype = prototype(
          ets.model=structure(list(), class="ets")
        )
```

```
)

calculateMASE <- function(training,validationSet,forecastSet){
  nlen = length(training)
  error.terms = validationSet - forecastSet
  esum = 0
  for (i in 2:nlen){
    esum = esum + abs(training[i] - training[i-1] )
  }
  if (esum == 0) {
    esum = 0.0001 # code to avoid division by zero error
  }
  q.t = error.terms / (esum/(nlen-1))
  rm(nlen)
  rm(esum)
  rm(error.terms)
  return (mean(abs(q.t)))
}

checkShapiroWilkResiduals <- function(residuals){
  sw.test <- shapiro.test(residuals)
  pval <- sw.test$p.value
  ct <- 0
  if(pval < 0.05) {
    ct <- 1 #sum((residuals > 0.05) == TRUE) + sum((residuals < -0.05) == TRUE)
  }
  rm(sw.test)
  return (c(pval,ct))
}

checkBoxTestResiduals <- function(residuals){
  box.test <- Box.test(residuals, lag = 1, type = "Ljung-Box", fitdf = 0)
  pval_box <- box.test$p.value
  ct_box <- 0
  if(pval_box < 0.05) {
    ct_box <- 1 #sum((residuals > 0.05) == TRUE) + sum((residuals < -0.05) == TRUE)
  }
  rm(box.test)
  return (c(pval_box,ct_box))
}

processDataRow <- function(x, series_frequency, type) {

  id <- as.integer(x[1])
  n <- as.integer(x[2])
  forecast_period <- as.integer(x[3])
  category <- x[4]
  start_year <- as.integer(x[5])
  start_frequency <- as.integer(x[6])
  start_n <- 7

  vec <- as.vector(as.numeric(x[seq(from = start_n, to = n + start_n - 1)]))
```

```
  #checking for minimum value and calculating value that is to be added to time series object when nega

  min_x <- min(vec)
  superposition <- ifelse(min_x > 0,0,(min_x * -1 + 0.1))

  diff_n <- as.integer(n * .05)

  diff_n <- ifelse(diff_n == 1,2,diff_n)

  vec_2 <- as.vector(as.numeric(x[seq(from = start_n, to = n + start_n - diff_n - 1)])) # change formul

  vec_3 <- as.vector(as.numeric(x[seq(from = start_n + n - diff_n, to = start_n + n - 1)]))

  ts_complete <- ts(vec,start = c(start_year,start_frequency),frequency=series_frequency)
  ts_training <- ts(vec_2,start = c(start_year,start_frequency),frequency=series_frequency)

  lambda = BoxCox.lambda(ts_training)

  return (new("TClass",type=type,
              frequency=series_frequency,
              id=id,
              n=n,
              diff_n=diff_n,
              forecast_period=forecast_period,
              category=category,
              start_year=start_year,
              start_frequency=start_frequency,
              superposition=superposition,
              lambda=lambda,
              original_data=vec,
              validation=vec_3,
              ts_complete=ts_complete,
              ts_training=ts_training)
  )
}

getType <- function(sheetNumber) {
  if (sheetNumber == 1) "Y"
  else if (sheetNumber == 2) "Q"
  else if (sheetNumber == 3) "M"
}

getFrequency <- function(sheetNumber) {
  if (sheetNumber == 1) 1
  else if (sheetNumber == 2) 4
  else if (sheetNumber == 3) 12
}

getMethodName <- function(m) {
  st <- substring(text = m,5)
  st <- substring(text = st,1, nchar(st) - 1)
  st <- gsub(",","",st)
}
```

```
getAutoModels <- function(tsobject, forecasting_period, lambda) {
  lst <- list()
  ics <- c("bic", "aicc", "aic")
  dampeds <- c(TRUE, FALSE)
  lpresent <- toString(ifelse(is.null(lambda),'N','Y'))
  opt_crit = c("lik", "amse", "mse", "sigma", "mae")
  xg <- expand.grid(ics,dampeds,opt_crit)
  for(i in 1:nrow(xg)){
    fit.auto <- ets(tsobject,model="ZZZ", damped = xg[i,2], ic=toString(xg[i,1]), opt.crit=toString(xg[
    k <- ifelse(xg[i,2], paste("ETS_AUTO",toString(xg[i,1]),"damped",toString(xg[i,3]),lpresent, sep='_
    frc.ets <- forecast(fit.auto,h=forecasting_period)
    vec_sw <- checkShapiroWilkResiduals(fit.auto$residuals)
    vec_box <- checkBoxTestResiduals(fit.auto$residuals)
    lst[[k]] <- new("RClass",key=k,
              lambda=lpresent,
              mase=accuracy(fit.auto)[6],
              aic=fit.auto$aic,
              bic=fit.auto$bic,
              aicc=fit.auto$aicc,
              swtest.pvalue=vec_sw[1],
              swtest.count=vec_sw[2],
              boxtest.pvalue=vec_box[1],
              boxtest.count=vec_box[2],
              ts_forecast=frc.ets$mean,
              forecast.model=frc.ets,
              ets.model=fit.auto)
    rm(frc.ets)
    rm(k)
    rm(fit.auto)
    rm(vec_sw)
    rm(vec_box)
  }
  rm(i)
  rm(ics)
  rm(dampeds)
  rm(xg)
  rm(lpresent)
  return (lst)
}

getSesModels <- function(tsobject, forecasting_period, lambda) {
  lst <- list()
  initials <- c("simple", "optimal")
  exponentials <- c(TRUE, FALSE)
  dampeds <- c(TRUE, FALSE)
  lpresent <- toString(ifelse(is.null(lambda),'N','Y'))
  xg <- expand.grid(initials,dampeds,exponentials)
  for(i in 1:nrow(xg)){
    k <- paste('ses',toString(xg[i,1]),ifelse(xg[i,2],'d',''),ifelse(xg[i,3],'x',''),lpresent,sep='')
    tryCatch({
      fit.ses <- ses(tsobject,initial=toString(xg[i,1]), damped=xg[i,2], exponential=xg[i,3], h=forecas
      vec_sw <- checkShapiroWilkResiduals(fit.ses$residuals)
      vec_box <- checkBoxTestResiduals(fit.ses$residuals)
```

```
      lst[[k]] <- new("RClass",key=k,
              lambda=lpresent,
              mase=accuracy(fit.ses)[6],
              aic=ifelse(is.null(fit.ses$model$aic),-1,fit.ses$model$aic),
              bic=ifelse(is.null(fit.ses$model$bic),-1,fit.ses$model$bic),
              aicc=ifelse(is.null(fit.ses$model$aicc),-1,fit.ses$model$aicc),
              swtest.pvalue=vec_sw[1],
              swtest.count=vec_sw[2],
              boxtest.pvalue=vec_box[1],
              boxtest.count=vec_box[2],
              ts_forecast=fit.ses$mean,
              forecast.model=fit.ses)
      rm(fit.ses)
      rm(vec_sw)
      rm(vec_box)
    },error=function(e){
      print(paste(k,e,sep=":"))
    })
    rm(k)
  }
  rm(i)
  rm(initials)
  rm(dampeds)
  rm(exponentials)
  rm(xg)
  rm(lpresent)
  return (lst)
}

getHoltModels <- function(tsobject, forecasting_period, lambda){
  lst <- list()
  initials <- c("simple", "optimal")
  exponentials <- c(TRUE, FALSE)
  dampeds <- c(TRUE, FALSE)
  lpresent <- toString(ifelse(is.null(lambda),'N','Y'))
  xg <- expand.grid(initials,dampeds,exponentials)
  for(i in 1:nrow(xg)){
    if((xg[i,1] == "simple" && xg[i,2]==FALSE) || (xg[i,1] == "optimal")) {
      k <- paste('holt',toString(xg[i,1]),ifelse(xg[i,2],'d',''),ifelse(xg[i,3],'x',''),lpresent,sep='')
      tryCatch({
        fit.holt <- holt(tsobject, initial=toString(xg[i,1]), damped=xg[i,2], exponential=xg[i,3], h=fo
        vec_sw <- checkShapiroWilkResiduals(fit.holt$residuals)
        vec_box <- checkBoxTestResiduals(fit.holt$residuals)
        lst[[k]] <- new("RClass",key=k,
                lambda=lpresent,
                mase=accuracy(fit.holt)[6],
                aic=ifelse(is.null(fit.holt$model$aic),-1,fit.holt$model$aic),
                bic=ifelse(is.null(fit.holt$model$bic),-1,fit.holt$model$bic),
                aicc=ifelse(is.null(fit.holt$model$aicc),-1,fit.holt$model$aicc),
                swtest.pvalue=vec_sw[1],
                swtest.count=vec_sw[2],
                boxtest.pvalue=vec_box[1],
                boxtest.count=vec_box[2],
```

```
                ts_forecast=fit.holt$mean,
                forecast.model=fit.holt)
        rm(vec_sw)
        rm(vec_box)
        rm(fit.holt)
      },error=function(e){
        print(paste(k,e,sep=":"))
      })
      rm(k)
    }
  }
  rm(i)
  rm(initials)
  rm(dampeds)
  rm(lpresent)
  rm(exponentials)
  rm(xg)
  return (lst)
}

getHoltWintersModels <- function(tsobject, forecasting_period, lambda){
  lst <- list()
  seasonals <- c("additive", "multiplicative")
  exponentials <- c(TRUE, FALSE)
  dampeds <- c(TRUE, FALSE)
  lpresent <- toString(ifelse(is.null(lambda),'N','Y'))
  xg <- expand.grid(seasonals,dampeds,exponentials)
  for(i in 1:nrow(xg)){
    k <- paste('hw',toString(xg[i,1]),ifelse(xg[i,2],'d',''),ifelse(xg[i,3],'x',''),lpresent,sep='')
    tryCatch({
      fit.hw <- hw(tsobject,seasonal=toString(xg[i,1]), damped=xg[i,2], exponential=xg[i,3], h=forecasti
      vec_sw <- checkShapiroWilkResiduals(fit.hw$residuals)
      vec_box <- checkBoxTestResiduals(fit.hw$residuals)
      lst[[k]] <- new("RClass",key=k,
              lambda=lpresent,
              mase=accuracy(fit.hw)[6],
              aic=ifelse(is.null(fit.hw$model$aic),-1,fit.hw$model$aic),
              bic=ifelse(is.null(fit.hw$model$bic),-1,fit.hw$model$bic),
              aicc=ifelse(is.null(fit.hw$model$aicc),-1,fit.hw$model$aicc),
              swtest.pvalue=vec_sw[1],
              swtest.count=vec_sw[2],
              boxtest.pvalue=vec_box[1],
              boxtest.count=vec_box[2],
              ts_forecast=fit.hw$mean,
              forecast.model=fit.hw)
      rm(fit.hw)
      rm(vec_box)
      rm(vec_sw)
    },error=function(e){
      print(paste(k,e,sep=":"))
    })
    rm(k)
  }
```

```
  rm(i)
  rm(seasonals)
  rm(dampeds)
  rm(lpresent)
  rm(exponentials)
  rm(xg)
  return (lst)
}

getEtsModels <- function(tsobject, forecasting_period, lambda){
  lst <- list()
  e <- c("A","M","N")
  t <- c("A","M","N")
  s <- c("A","M","N")
  dampeds <- c(TRUE, FALSE)
  lpresent <- toString(ifelse(is.null(lambda),'N','Y'))
  opt_crit = c("lik", "amse", "mse", "sigma", "mae")
  xg <- expand.grid(e,t,s,dampeds,opt_crit)
  for(i in 1:nrow(xg)){
    mt <- toString(paste(xg[i,1],xg[i,2],xg[i,3],sep = ""))
    k <- paste('ETS(',mt,ifelse(xg[i,4],'d',''),')', toString(xg[i,5]),lpresent,sep='')
    tryCatch({
      if(mt == "NNN") {
        fit.ets <- ets(tsobject, damped = xg[i,4], lambda = lambda, opt.crit=toString(xg[i,5]))
        frc.ets <- forecast(fit.ets,h=forecasting_period)
        vec_sw <- checkShapiroWilkResiduals(fit.ets$residuals)
        vec_box <- checkBoxTestResiduals(fit.ets$residuals)
        lst[[k]] <- new("RClass",key=k,
              lambda=lpresent,
              mase=accuracy(fit.ets)[6],
              aic=fit.ets$aic,
              bic=fit.ets$bic,
              aicc=fit.ets$aicc,
              swtest.pvalue=vec_sw[1],
              swtest.count=vec_sw[2],
              boxtest.pvalue=vec_box[1],
              boxtest.count=vec_box[2],
              ts_forecast=frc.ets$mean,
              forecast.model=frc.ets,
              ets.model=fit.ets)
        rm(frc.ets)
        rm(fit.ets)
        rm(vec_sw)
        rm(vec_box)
      } else if(xg[i,1] != "N") {
        if(xg[i,3] != "N") {
          if(frequency(tsobject) != 1) {
            fit.ets <- ets(tsobject, model=mt, damped=xg[i,4], lambda = lambda, opt.crit=toString(xg[i,5
            frc.ets <- forecast(fit.ets,h=forecasting_period)
            vec_sw <- checkShapiroWilkResiduals(fit.ets$residuals)
            vec_box <- checkBoxTestResiduals(fit.ets$residuals)
            lst[[k]] <- new("RClass",key=k,
                lambda=lpresent,
```

```
                mase=accuracy(fit.ets)[6],
                aic=fit.ets$aic,
                bic=fit.ets$bic,
                aicc=fit.ets$aicc,
                swtest.pvalue=vec_sw[1],
                swtest.count=vec_sw[2],
                boxtest.pvalue=vec_box[1],
                boxtest.count=vec_box[2],
                ts_forecast=frc.ets$mean,
                forecast.model=frc.ets,
                ets.model=fit.ets)
            rm(frc.ets)
            rm(fit.ets)
            rm(vec_sw)
            rm(vec_box)
          }
        } else {
          fit.ets <- ets(tsobject, model=mt, damped=xg[i,4], lambda = lambda, opt.crit=toString(xg[i,5]))
          frc.ets <- forecast(fit.ets,h=forecasting_period)
          vec_sw <- checkShapiroWilkResiduals(fit.ets$residuals)
          vec_box <- checkBoxTestResiduals(fit.ets$residuals)
          lst[[k]] <- new("RClass",key=k,
                lambda=lpresent,
                mase=accuracy(fit.ets)[6],
                aic=fit.ets$aic,
                bic=fit.ets$bic,
                aicc=fit.ets$aicc,
                swtest.pvalue=vec_sw[1],
                swtest.count=vec_sw[2],
                boxtest.pvalue=vec_box[1],
                boxtest.count=vec_box[2],
                ts_forecast=frc.ets$mean,
                forecast.model=frc.ets,
                ets.model=fit.ets)
          rm(frc.ets)
          rm(fit.ets)
          rm(vec_sw)
          rm(vec_box)
        }
      }
    },error=function(e){
      print(paste(k,e,sep=":"))
    })
    rm(k)
    rm(mt)
}
rm(i)
rm(e)
rm(t)
rm(s)
rm(lpresent)
rm(dampeds)
rm(opt_crit)
```

```
  rm(xg)
  return (lst)
}
```

Function to run all combinations of models for a given time series

```
getAllModelsMase <- function(tsobject, forecasting_period, lambda) {
    maselst <- list()
    maselst <- append(maselst, getAutoModels(tsobject, forecasting_period, lambda))
    maselst <- append(maselst, getSesModels(tsobject, forecasting_period, lambda))
    maselst <- append(maselst, getHoltModels(tsobject, forecasting_period, lambda))
    if (frequency(tsobject) != 1) {
        maselst <- append(maselst, getHoltWintersModels(tsobject, forecasting_period,
            lambda))
    }
    maselst <- append(maselst, getEtsModels(tsobject, forecasting_period, lambda))
    return(maselst)
}
```

Reading all data and processing it to create time series objects, outputs list of class objects containing details

```
all_objects <- list()
for (i in c(1, 2, 3)) {
    df <- read_excel(path = "./data/M3C_reduced.xlsx", sheet = i, trim_ws = TRUE)
    output <- apply(df, 1, processDataRow, getFrequency(i), getType(i))
    all_objects <- append(all_objects, output)
    rm(df)
    rm(output)
}
rm(i)
```

Running all combinations of models on each time series

```
modelsSpecs <- list()
len <- length(all_objects)
for (il in 1:len) {
    element <- all_objects[[il]]
    print(all_objects[[il]]@id)
    lst <- list()
    lst <- append(lst, getAllModelsMase(element@ts_training, element@forecast_period,
        NULL))
    lst <- append(lst, getAllModelsMase(element@ts_training, element@forecast_period,
        element@lambda))
    modelsSpecs[[all_objects[[il]]@id]] <- lst
    rm(lst)
}
rm(il)
rm(element)
```

Methods for generating model description

```
concatenateWithCheck <- function(x, y) {
    if (y != "") {
        return(paste(x, y, sep = " "))
    }
    return(x)
}
getModelNameForKey <- function(key) {
    transformed <- ""
    base.model <- ""
    if (endsWith(key, "Y")) {
        transformed <- "Transformed using Box Cox"
    }
    if (startsWith(key, "ses")) {
        base.model <- "SES"
        st <- substr(key, nchar("ses") + 1, nchar(key) - 1)
        r <- concatenateWithCheck(base.model, ifelse(startsWith(st, "simple"),
            "simple", "optimal"))
        r <- concatenateWithCheck(r, ifelse(endsWith(st, "dx"), "Damped", (ifelse(endsWith(st,
            "d"), "Damped", ""))))
        r <- concatenateWithCheck(r, ifelse(endsWith(st, "dx"), "Exponential",
            (ifelse(endsWith(st, "x"), "Exponential", ""))))
        r <- concatenateWithCheck(r, transformed)
        return(r)
    } else if (startsWith(key, "ETS_AUTO")) {
        base.model <- "ETS AUTO"
        st <- substr(key, nchar("ETS_AUTO_") + 1, nchar(key) - 2)
        v <- as.vector(strsplit(st, "_")[[1]])
        if (length(v) > 2) {
            ic <- v[1]
            damped <- "Damped"
            opt <- concatenateWithCheck("Optimisation criteria", v[3])
        } else if (length(v) == 2) {
            ic <- v[1]
            damped <- ""
            opt <- concatenateWithCheck("Optimisation criteria", v[2])
        }
        r <- concatenateWithCheck(base.model, ic)
        r <- concatenateWithCheck(r, damped)
        r <- concatenateWithCheck(r, opt)
        r <- concatenateWithCheck(r, transformed)
        return(r)
    } else if (startsWith(key, "holt")) {
        base.model <- "Holts"
        st <- substr(key, nchar("holt") + 1, nchar(key) - 1)
        initials <- ifelse(startsWith(st, "simple"), "simple", "optimal")
        damped <- ifelse(endsWith(st, "dx"), "Damped", (ifelse(endsWith(st,
            "d"), "Damped", "")))
        exponential <- ifelse(endsWith(st, "dx"), "Exponential", (ifelse(endsWith(st,
            "x"), "Exponential", "")))
        r <- concatenateWithCheck(base.model, initials)
        r <- concatenateWithCheck(r, damped)
```

```
        r <- concatenateWithCheck(r, exponential)
        r <- concatenateWithCheck(r, transformed)
        return(r)
    } else if (startsWith(key, "hw")) {
        base.model <- "Holts Winters"
        st <- substr(key, nchar("hw") + 1, nchar(key) - 1)
        seasonal <- ifelse(startsWith(st, "additive"), "additive", "multiplicative")
        damped <- ifelse(endsWith(st, "dx"), "Damped", (ifelse(endsWith(st,
            "d"), "Damped", "")))
        exponential <- ifelse(endsWith(st, "dx"), "Exponential", (ifelse(endsWith(st,
            "x"), "Exponential", "")))
        r <- concatenateWithCheck(base.model, seasonal)
        r <- concatenateWithCheck(r, damped)
        r <- concatenateWithCheck(r, exponential)
        r <- concatenateWithCheck(r, transformed)
        return(r)
    } else if (startsWith(key, "ETS(")) {
        base.model <- "ETS"
        st <- substr(key, nchar("ETS(") + 1, nchar(key) - 1)
        v <- as.vector(strsplit(st, ")")[[1]])
        m <- v[1]
        damped <- ifelse(endsWith(m, "d"), "Damped", "")
        m <- ifelse(endsWith(m, "d"), substr(m, 1, nchar(m) - 1), m)
        e <- substr(m, 1, 1)
        t <- substr(m, 2, 2)
        s <- substr(m, 3, 3)
        opt <- concatenateWithCheck("Optimisation criteria", v[2])
        r <- concatenateWithCheck(base.model, m)
        r <- concatenateWithCheck(r, damped)
        r <- concatenateWithCheck(r, opt)
        r <- concatenateWithCheck(r, transformed)
        return(r)
    }
}
```

Creating a data frame for sorting and reporting purpose

```
result_df <- data.frame(id = numeric(), type = character(), frequency = numeric(),
    category = character(), isTransformed = character(), key = character(),
    description = character(), method = character(), mase = numeric(), aic = numeric(),
    aicc = numeric(), bic = numeric(), swtest.pvalue = numeric(), swtest.count = numeric(),
    boxtest.pvalue = numeric(), boxtest.count = numeric(), forecase.mase = numeric())
names_vec <- c("id", "type", "frequency", "category", "transformed", "modelKey",
    "description", "method", "mase", "aic", "aicc", "bic", "sw.pvalue", "sw.count",
    "box.pvalue", "box.count", "forecast.mase")

names(result_df) <- names_vec
for (il in 1:len) {
    lst <- modelsSpecs[[all_objects[[il]]@id]]
    df <- data.frame(id = numeric(), type = character(), frequency = numeric(),
        category = character(), isTransformed = character(), key = character(),
        description = character(), method = character(), mase = numeric(), aic = numeric(),
```

```
        aicc = numeric(), bic = numeric(), swtest.pvalue = numeric(), swtest.count = numeric(),
        boxtest.pvalue = numeric(), boxtest.count = numeric(), forecase.mase = numeric())
    names(df) <- names_vec
    for (loop in 1:length(lst)) {
        vc <- as.vector(lst[[loop]]@ts_forecast)
        vc <- vc[1:length(all_objects[[il]]@validation)]
        l <- length(grep(pattern = "^ETS", x = lst[[loop]]@key))
        mtd <- NULL
        if (l > 0) {
            mtd <- lst[[loop]]@ets.model
        } else {
            mtd <- lst[[loop]]@forecast.model
        }
        temp <- data.frame(all_objects[[il]]@id, all_objects[[il]]@type, all_objects[[il]]@frequency,
            all_objects[[il]]@category, lst[[loop]]@lambda, lst[[loop]]@key,
            getModelNameForKey(lst[[loop]]@key), mtd$method, lst[[loop]]@mase,
            lst[[loop]]@aic, lst[[loop]]@aicc, lst[[loop]]@bic, lst[[loop]]@swtest.pvalue,
            lst[[loop]]@swtest.count, lst[[loop]]@boxtest.pvalue, lst[[loop]]@boxtest.count,
            calculateMASE(as.vector(all_objects[[il]]@ts_training), all_objects[[il]]@validation,
                vc))
        names(temp) <- names_vec
        df <- rbind(df, temp)
        rm(temp)
        rm(vc)
        rm(l)
        rm(mtd)
    }
    df <- df[with(df, order(mase)), ]
    # Adding data frame for future reporting purpose
    all_objects[[il]]@models_df <- df
    # Selection logic for best model needs to go here
    result_df <- rbind(result_df, df[1, ])
    rm(df)
    rm(lst)
}
rm(il)
rm(loop)
rm(names_vec)
```

Utility functions for reporting purposes

```
generatePlot <- function(identifier) {
    vc <- c(as.vector(all_objects[[identifier]]@ts_complete), as.vector(modelsSpecs[[identifier]][[toSt
        identifier, ]$modelKey)]]@ts_forecast))
    ylv <- c((min(vc) - 1), (max(vc) - 1))
    xlv <- c(start(all_objects[[identifier]]@ts_complete)[1] - 1, end(modelsSpecs[[identifier]][[toStri
        identifier, ]$modelKey)]]@ts_forecast)[1] + 1)
    rm(vc)
    plot(all_objects[[identifier]]@ts_complete, col = c("blue"), ylab = paste(toString(identifier),
        "Time series", sep = " "), ylim = ylv, xlim = xlv)
    lines(modelsSpecs[[identifier]][[toString(result_df[result_df$id == identifier,
        ]$modelKey)]]@ts_forecast, col = c("red"))
```

```
    rm(ylv)
    rm(xlv)
}

getModel <- function(identifier) {
    k <- toString(result_df[result_df$id == identifier, ]$modelKey)
    l <- length(grep(pattern = "^ETS", x = k))
    m <- NULL
    if (l > 0) {
        m <- modelsSpecs[[identifier]][[k]]@ets.model
    } else {
        m <- modelsSpecs[[identifier]][[k]]@forecast.model
    }
    rm(l)
    rm(k)
    return(m)
}

getMethodForModel <- function(identifier) {
    return(getModel(identifier)$method)
}

checkresidualsforModel <- function(identifier) {
    checkresiduals(getModel(identifier))
}
```