Final Project TS

Roy Madpis (319091526) And Michael Kobaivanov (206814485)

```
knitr::opts_chunk$set(echo = TRUE)
#libraries
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.1

## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'purrr' was built under R version 4.3.1

## Warning: package 'dplyr' was built under R version 4.3.1

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(cowplot) #for multiple plots together
```

```
##
## Attaching package: 'cowplot'
##
## The following object is masked from 'package:lubridate':
##
##     stamp
```

```
library(timetk) #for generating tibble time series
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.1

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(readxl) #for reading excel files (xls, xlsx)
```

```
## Warning: package 'readxl' was built under R version 4.3.1
```

```r
library(lubridate) #for time manupulations
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(DT) #for html interactive table - datatable()
```

```
## Warning: package 'DT' was built under R version 4.3.1
```

```r
library(ggpubr) #to plot multiple plots in the same output
```

```
##
## Attaching package: 'ggpubr'
##
## The following object is masked from 'package:forecast':
##
##     gghistogram
##
## The following object is masked from 'package:cowplot':
##
##     get_legend
```

```r
library(corrplot) #for correlation matrix
```

```
## corrplot 0.92 loaded
```

```r
#for performance evaluation
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'
##
## The following object is masked from 'package:base':
##
##     Recall
```

```r
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following objects are masked from 'package:MLmetrics':
##
##     MAE, RMSE
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(tsibble)
```

```
##
## Attaching package: 'tsibble'
##
## The following object is masked from 'package:zoo':
##
##     index
##
## The following object is masked from 'package:lubridate':
##
##     interval
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, union
```

```r
library(feasts) #for using the features function
```

```
## Loading required package: fabletools
```

```
## Warning: package 'fabletools' was built under R version 4.3.3
```

```
##
## Attaching package: 'fabletools'
##
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
##
## The following objects are masked from 'package:MLmetrics':
##
##     MAE, MAPE, MSE, RMSE
```

```r
library(fable) #for the ARIMA Function

library(sjmisc) #for transposing the dataframe
```

```
## Warning: package 'sjmisc' was built under R version 4.3.1
```

```
##
## Attaching package: 'sjmisc'
##
## The following object is masked from 'package:purrr':
##
##     is_empty
##
## The following object is masked from 'package:tidyr':
##
##     replace_na
##
## The following object is masked from 'package:tibble':
##
##     add_case
```

```r
library(ggfortify) #for acf confidence intervals
```

```
## Registered S3 methods overwritten by 'ggfortify':
##   method                 from
##   autoplot.Arima         forecast
##   autoplot.acf           forecast
##   autoplot.ar            forecast
##   autoplot.bats          forecast
##   autoplot.decomposed.ts forecast
##   autoplot.ets           forecast
##   autoplot.forecast      forecast
##   autoplot.glmnet        parsnip
##   autoplot.stl           forecast
##   autoplot.ts            forecast
##   fitted.ar              forecast
##   fortify.ts             forecast
##   residuals.ar           forecast
```

```r
library(nnfor) #for mlp function for forecasting time sereis with ML
```

```
## Warning: package 'nnfor' was built under R version 4.3.3
```

```
## Loading required package: generics
##
## Attaching package: 'generics'
##
## The following object is masked from 'package:caret':
##
##     train
##
## The following object is masked from 'package:lubridate':
##
##     as.difftime
##
## The following object is masked from 'package:dplyr':
##
##     explain
```

```
##
## The following objects are masked from 'package:base':
##
##     as.difftime, as.factor, as.ordered, intersect, is.element, setdiff,
##     setequal, union
##
## Registered S3 method overwritten by 'greybox':
##   method     from
##   print.pcor lava
## Registered S3 methods overwritten by 'tsutils':
##   method          from
##   print.nemenyi   greybox
##   summary.nemenyi greybox
```

```r
### more on the mlp function in this package can be found in the following link: https://rdrr.io/cran/n

library(smooth) #for the CES model
```

```
## Warning: package 'smooth' was built under R version 4.3.3

## Loading required package: greybox

## Warning: package 'greybox' was built under R version 4.3.3

## Package "greybox", v2.0.2 loaded.
##
##
## Attaching package: 'greybox'
##
## The following objects are masked from 'package:fabletools':
##
##     MAE, MAPE, MASE, ME, MPE, MSE, RMSSE
##
## The following object is masked from 'package:tsibble':
##
##     measures
##
## The following object is masked from 'package:caret':
##
##     MAE
##
## The following objects are masked from 'package:MLmetrics':
##
##     MAE, MAPE, MSE
##
## The following object is masked from 'package:lubridate':
##
##     hm
##
## The following object is masked from 'package:tidyr':
##
##     spread
##
## This is package "smooth", v4.1.0
```

```r
library(forecTheta) #for theta forecasting method
```

```
## Warning: package 'forecTheta' was built under R version 4.3.3
```

```
## Loading required package: tseries
```

```
## Warning: package 'tseries' was built under R version 4.3.1
```

```r
#################### Data Location:
#Change the folder location to the location in your local computer where the data is stored:
data_folder <- "Dataset/"

#In each question provide the name of the data-file that is used in the question:
file_name <- "roy" ######change this!

data_location <-  paste(data_folder,file_name,sep="")
```

This is the Final project in Time Series course,

We present here: 1. The functions we have built for this project (Predefined functions for Tasks 1-3) 2. The 10 Times series we chose (Tasks 1 + 2) 3. The article part (Task 3)

- Note that as was discussed in class, we didn't implement the RNN method in R, but in Python, so in the zip file you can find the python code that we used to generate the predictions via the RNN method. We read the csv files with the predictions to this R Markdown.

Predefing Helper Functions

We begin with declaring the functions that we developed for completing Tasks 1-3:

Function that calcualtes the performance of the classifiaction model:

```r
library(MLmetrics)
library(caret)

Madpis_classification_performance <-
    function(y_pred_train=NA, y_true_train=NA, y_pred_test=NA, y_true_test=NA){

   #train performance
   if (!is.na(y_pred_train)){
      accuracy_train <- round(MLmetrics::Accuracy(y_pred_train, y_true_train),3)
      precision_train <- round(MLmetrics::Precision(y_pred_train, y_true_train),3)
      recall_train <- round(MLmetrics::Recall(y_pred_train, y_true_train),3)
      f1_score_train <- round(MLmetrics::F1_Score(y_pred_train,y_true_train), 3)
      #confusion matrix - train
      CM_train <- caret::confusionMatrix(as.factor(y_pred_train), as.factor(y_true_train))
   } else{
      accuracy_train <-NA; precision_train <- NA; recall_train <- NA; f1_score_train <-NA;CM_train <- NA
   }
```

```
  #test performance
  if (!is.na(y_pred_test)){
    accuracy_test <- round(MLmetrics::Accuracy(y_pred_test, y_true_test), 3)
    precision_test <- round(MLmetrics::Precision(y_pred_test, y_true_test), 3)
    recall_test <- round(MLmetrics::Recall(y_pred_test, y_true_test), 3)
    f1_score_test <- round(MLmetrics::F1_Score(y_pred_test,y_true_test), 3)
  #confusion matrix - test
    CM_test <- caret::confusionMatrix(as.factor(y_pred_test), as.factor(y_true_test))
  } else{
    accuracy_test <- NA; precision_test <- NA; recall_test <- NA; f1_score_test <- NA ; CM_test <- NA
  }

  #performance table
  model_performance_tib <- tibble("type" = c("Train", "Test"),
                      "Accuracy" = c(accuracy_train, accuracy_test),
                      "Precision"=c(precision_train, precision_test),
                      "Recall" = c(recall_train, recall_test),
                      "F1_Score" = c(f1_score_train, f1_score_test))
  #The output:
  output <- list()
  output$model_performance_tib <- model_performance_tib
  output$CM_train <- CM_train
  output$CM_test <- CM_test

  return (output)
}

### Using the function
#output_performance <-  Madpis_classification_performance(
#   y_pred_train = y_pred_train,
#   y_true_train = y_true_train,
#   y_pred_test = y_pred_test,
#   y_true_test = y_true_test)

#output_performance$CM_train
#output_performance$CM_test
#output_performance$model_performance_tib
```

Madpis_preprocess

The following function enables getting as input a tibble with time series with first column containing the time-index and k more column of k different time series. In argument `data_col_name` provide the column name you wish to separate and in argument `new_data_col_name` provide the new column name. The function will automatically omit all rows with NAs in the new data frame.

```
Madpis_preprocess <-
    function(ts_data_tib, data_col_name, new_data_col_name = "value", label = "None"){

  if (label != "None"){
    final_df <- ts_data_tib %>% select(c("Date", data_col_name, label))
    colnames(final_df) <- c(colnames(final_df)[1],new_data_col_name, label)
```

```
  } else{
    final_df <- ts_data_tib %>% select(c("Date", data_col_name))
    colnames(final_df) <- c(colnames(final_df)[1],new_data_col_name)
  }

  final_df <- na.omit(final_df)
  return(final_df)
}
```

Plotting

Madpis_plot_ts

Madpis_plot_ts is a function that enables quickly plotting: 1. The time plot of the series 2. The Decomposition graph 3. The seasonal graph 4. The ACF, PACF

The function gets as input: 1. ts_data_tib = time series tibble 2. time series object (ts) = by default = "Auto", with this default value, the function will take the time series tibble table and will make from it a ts object. In order for this to happen you must insert in `data_freq` value. note that you also need to provide in argument `column_name_with_value` the name of the column in `ts_data_tib` that holds the time series data.

3. **column_name_with_value** = The name of the column in `ts_data_tib` that contains the time series data, by default = "value"

4. **data_freq** = can hold one of the following supported possibilities:

- "Y" / "y" / "Yearly" / "year" / "Year"
- "Q" / "q" / "Quarterly" / "Quarter" / "quarter"
- "M" / "m" / "Monthly" / "Month" / "month"

5. title_for_time_plot
6. subtitle_for_time_plot –> if it is "Auto" then you need to pass to the argument `data_freq` one of the following supported possibilities:

- "Y" / "y" / "Yearly" / "year" / "Year"
- "Q" / "q" / "Quarterly" / "Quarter" / "quarter"
- "M" / "m" / "Monthly" / "Month" / "month" And the function will automatically create a subtitle for you.

7. plot_decompose = Binary - TRUE by default - if you wish to plot a decomposition graph

8. plot_seasonal = Binary - TRUE by default - if you wish to plot a seasonality graph

```
#Function to plot the series:

Madpis_plot_ts <- function(ts_data_tib, ts_data = "Auto",
                           column_name_with_value = "value",
                           data_freq = "",
                           title_for_time_plot = "Time plot",
                           subtitle_for_time_plot = "Auto",
                           plot_decompose = TRUE,
```

```
                         plot_seasonal = TRUE,
                         plot_trend_lines = TRUE
                         ){

####################### subtitle_for_time_plot and data_freq
if (data_freq == "Q" || data_freq == "q" || data_freq == "Quarterly" || data_freq == "quarter" || da
   number_of_periods_in_year <- 4
   number_of_years <- floor(nrow(ts_data_tib)/number_of_periods_in_year)
   subtitle_auto <- paste0(number_of_years, " Years of Quarterly data")
} else if (data_freq == "M" || data_freq == "m" || data_freq == "Monthly" || data_freq == "month" ||
   number_of_periods_in_year <- 12
   number_of_years <- floor(nrow(ts_data_tib)/number_of_periods_in_year)
   subtitle_auto <- paste0(number_of_years, " Years of Monthly data")
}   else if (data_freq == "Y" || data_freq == "y" || data_freq == "Yearly" || data_freq == "year" ||
   number_of_periods_in_year <- 1
   number_of_years <- floor(nrow(ts_data_tib)/number_of_periods_in_year)
   subtitle_auto <- paste0(number_of_years, " Years of yearly data")
} else{
   subtitle_auto = ""
}

if (subtitle_for_time_plot == "Auto"){
   subtitle_for_time_plot = subtitle_auto
}

if (ts_data == "Auto"){
   ts_data <- ts(ts_data_tib[[column_name_with_value]],
              frequency = number_of_periods_in_year)

}

#plotting the time series
time_plot <- ggplot(ts_data_tib) +
geom_point(mapping = aes(x=Date, y=value))+
geom_line(mapping = aes(x=Date, y=value))+
labs(title = title_for_time_plot,
     subtitle = subtitle_for_time_plot)
print(time_plot)
### decompose plot - to spot tend and seasonality
if (plot_decompose == TRUE){
   plot(decompose(ts_data))
   plot(stl(ts_data, s.window = "periodic"))
}


### Seasonal plot - to spot seasonality
if (plot_seasonal == TRUE){
   seasonal_plot <- ggseasonplot(ts_data, xlab = "Time")
   print(seasonal_plot)
}

#ACF, PACF
Acf(ts_data, main = "ACF") ; Pacf(ts_data, main = "PACF")
```

```r
  ### plotting trend lines can help understand the trend in the data
  if (plot_trend_lines == TRUE){
    ts_data_linear <- tslm(ts_data ~ trend)
    ts_data_exponential <- tslm(ts_data ~ trend + I(trend^2))

    plot(ts_data, xlab = "Time", ylab = "y", main = "Trend line plot")
    lines(ts_data_linear$fitted, lwd = 2, col="forestgreen")
    lines(ts_data_exponential $fitted, lwd = 2, col = "blue")
    abline(h=mean(ts_data), col="orange") #The level (average) line
    #legend:
    level1 <- paste("Level=", as.character(round(mean(ts_data),1)))

    legend("bottomright", legend=c("The Time-Series","Linear fit","Quadratic fit", level1), lwd = 3, c

  }

}
```

Madpis_Residuals_plots - Plot time plot, histogram and acf for the residuals

This function enables plotting the residuals of a given model. It gets as input:

- **model** = the trained model object
- **nValid** = the length og the validation period
- **train_ts** = the train_ts object
- **validation_ts** = the validation_ts object
- **level** = the confidence level to calculate for the model
- **bins_for_hist** = Number of bins for the histogram of train residuals

```r
Madpis_Residuals_plots <- function(model, nValid, train_ts, validation_ts, level = c(0.2, 0.4, 0.6, 0.8)

  if (class(model)!="forecast"){
    lm_pred <- forecast::forecast(model, h = nValid, level = level)
  }else{
    lm_pred <- model
}
  ### generating the train residuals tibble
  train_residuals <- tk_tbl(train_ts - lm_pred$fitted, rename_index = "Date") %>% mutate( group = "Tra

  colnames(train_residuals)[2] <- "value"

  ### generating the Validation residuals tibble
  validation_residuals <- tk_tbl(validation_ts - lm_pred$mean, rename_index= "Date") %>%mutate(group =

  ### plotting the residuals
  plot1 <- ggplot() +
   #plot train residuals
    geom_line(data = train_residuals, mapping=aes(x=Date, y=value, color = "Train Res")) +
```

```
    #plot Validation residuals
    geom_line(data=validation_residuals, mapping=aes(x=Date, y=value,
                                         color="Validation Res")) +
    labs(title = "Train and Validation Residuals", x = "")+
    scale_color_manual(name = "Legened",
                       values = c("Train Res" = "tomato3",
                                  "Validation Res" = "orange"))

  #histogram of train residuals
  subtitle_for_hist_res <- paste0("With ", bins_for_hist, " bins")
  plot2 <- ggplot() + geom_histogram(data = train_residuals, mapping=aes(x=value), bins = bins_for_his
    labs(title = "Histogram of Train Residuals",
         subtitle = subtitle_for_hist_res,
         x = "", y = "")


  ###acf plot

  ic_alpha <-  function(alpha, acf_res){
    return(qnorm((1 + (1 - alpha))/2)/sqrt(acf_res$n.used))
    }

  ggplot_acf_pacf <-  function(acf_object, label, alpha= 0.05){
    df_= with(acf_object, data.frame(lag, acf))

    # IC alpha
    lim1= ic_alpha(alpha, acf_object)
    lim0= -lim1

    plot_acf <- ggplot(data = df_, mapping = aes(x = lag, y = acf)) +
      geom_hline(aes(yintercept = 0)) +
      geom_segment(mapping = aes(xend = lag, yend = 0)) +
      labs(title = "ACF plot for the train residuals") +
      geom_hline(aes(yintercept = lim1), linetype = 2, color = 'blue') +
      geom_hline(aes(yintercept = lim0), linetype = 2, color = 'blue')

    return(plot_acf)
    }
  acf_data <- acf(na.remove(train_residuals$value), plot = F)
  plot3 <- ggplot_acf_pacf(acf_data)


  #bacf <- acf(train_residuals$value, plot = FALSE)
  #bacfdf <- with(bacf, data.frame(lag, acf))

 # plot3 <- ggplot(data = bacfdf, mapping = aes(x = lag, y = acf)) +
  #    geom_hline(aes(yintercept = 0)) +
  #    geom_segment(mapping = aes(xend = lag, yend = 0)) + labs(title = "ACF plot for the train residu


  ### plot all the 3 plots together
  ggdraw() +
    draw_plot(plot1, 0, .5, 1, .5) +
```

```r
      draw_plot(plot2, 0, 0, .5, .5) +
      draw_plot(plot3, .5, 0, .5, .5)
}
### use of function:
#Madpis_Residuals_plots(model = lm_model,
#                        nValid = nValid,
#                        train_ts = train_ts,
#                        validation_ts = validation_ts,
#                        level = c(0.2, 0.4, 0.6, 0.8),
#                        bins_for_hist = 7)
```

Madpis_Accuracy & Madpis_Accuracy_list_models Functions

Madpis_Accuracy_list_models function enables getting a list of fitted models and returning a table with the scores of all the models on both the train and test set. It gets as input a list of fitted models, a tsibble object containing the training and the test set and nValid number representing the number of periods in the test set (the forecast horizon).

```r
Madpis_Accuracy <- function(fit_model_object, tsibble_train_test, nValid){
   return(bind_rows(fit_model_object %>% forecast::accuracy(),
             fit_model_object %>% forecast(h=nValid) %>% forecast::accuracy(tsibble_train_test)))
}

Madpis_Accuracy_list_models <- function(list_models, tsibble_train_test, nValid){

   accuracy_vector <- vector()
   for (j in 1:length(list_models)){
      if (j ==1){
         accuracy_vector_final <- Madpis_Accuracy(
         fit_model_object = list_models[[j]],
         tsibble_train_test = tsibble_train_test,
         nValid = nValid)
      } else{
         accuracy_vector_i <- Madpis_Accuracy(
         fit_model_object = list_models[[j]],
         tsibble_train_test = tsibble_train_test,
         nValid = nValid)
         accuracy_vector_final <- bind_rows(accuracy_vector_final, accuracy_vector_i)
      }
}
   return(accuracy_vector_final)

}
```

Madpis_kpss_test

The Madpis_kpss_test allows to perform the KPSS Test and report on its outcome; we can use this function to find how many differencing should be done to remove trend and seaasonality and use that to build an ARIMA model (this will enable to decide the parameters for the ARIMA(p,d,q)(P,D,Q)).

```r
Madpis_kpss_test <- function(tsibble_data, col_name,
 alpha = 0.05, verbose = TRUE){
 output <- list()
 tsibble_data$value <- tsibble_data[[col_name]]

 #KPSS Test
 kpss_test <- tsibble_data %>% features(value, unitroot_kpss)
 kpss_test$H0 <- "The data is stationary" #This is the H0 of the KPSS Test
 kpss_test$Action_Item <- ifelse(kpss_test$kpss_pvalue < alpha, "Reject H0", "Accept H0")

 ### using unitroot_ndiffs to determine the number of times we need to perform differencing for station
 num_diff_for_stationary <- tsibble_data %>%
 features(value, unitroot_ndiffs)
 ### using unitroot_nsdiffs to determine the number of times we need to perform seasonal differencing f
 num_seasonal_diff_for_stationary <- tsibble_data %>% features(value, unitroot_nsdiffs)
 kpss_test$num_diff_for_stationary <- num_diff_for_stationary$ndiffs
 kpss_test$num_seasonal_diff_for_stationary <- num_seasonal_diff_for_stationary$nsdiffs
 if (verbose == TRUE){
 if (kpss_test$kpss_pvalue < alpha){
 print1 <- paste0("KPSS Test p-value is lower than ", alpha, " Thus we need to reject
H0: The data is **NOT Stationary** --> We need to perform differencing to transform the data
into stationary, In fact we need to perform the differencing ",
num_diff_for_stationary$ndiffs, " Times and perform seasonal differencing
",num_seasonal_diff_for_stationary$nsdiffs," times to get stationary data")
 print(print1)
 } else{
 print1 <- paste0("KPSS Test p-value is higher than ", alpha, " Thus we need to Accept
H0: The data is **Stationary**")
 print(print1)
 }
 }

 output$tsibble_data <- tsibble_data
 output$kpss_test <- kpss_test
 return(output)
}
```

Function Madpis_create_lm_pred_3_models

- This function belongs to the part 3 of building the models (:

This function gets as input 3 models pred objects (model pred object is the outcome of using the `predict` function on a model from the `forecast` package) and it yields an arithmetic average (by default). You can change the weights of the average.

```r
#model1 <- model_4_SES_pred
#model2 <- model_6_damped_pred
#model3 <- model_5a_HW_pred

Madpis_create_lm_pred_3_models <- function(model1, model2, model3, weight_model1 = 1/3, weight_model2 =

  all_models_mean <- weight_model1*model1$mean + weight_model2*model2$mean + weight_model3*model3$mean
```

```
  all_models_upper <- tk_tbl(weight_model1*model1$upper) + tk_tbl(weight_model2*model2$upper) + tk_tbl
  all_models_upper$index <- tk_tbl(weight_model1*model1$upper)$index
  all_models_upper <-  all_models_upper %>% rename(
                                  "Hi 20" = "20%",
                                  "Hi 40" = "40%",
                                  "Hi 60" = "60%",
                                  "Hi 80" = "80%",
                                  "Date" = "index")

  all_models_lower <- tk_tbl(weight_model1*model1$lower) + tk_tbl(weight_model2*model2$lower) + tk_tbl
  all_models_lower$index <- tk_tbl(weight_model1*model1$lower)$index
  all_models_lower <-  all_models_lower %>% rename(
                                  "Lo 20" = "20%",
                                  "Lo 40" = "40%",
                                  "Lo 60" = "60%",
                                  "Lo 80" = "80%",
                                  "Date" = "index")

  all_models_fitted <- weight_model1*model1$fitted + weight_model2*model2$fitted + weight_model3*model3

  all_models_residuals <- weight_model1*model1$residuals + weight_model2*model2$residuals + weight_mode

  all_models_methods <- paste0("Method model 1: ", model1$method, " Method model 2: ", model2$method,

  all_models_pred <- list()
  all_models_pred$mean <- all_models_mean
  all_models_pred$x <- model1$x
  all_models_pred$upper <- all_models_upper
  all_models_pred$lower <-  all_models_lower
  all_models_pred$fitted <-  all_models_fitted
  all_models_pred$residuals <- all_models_residuals
  all_models_pred$method <- all_models_methods
  class(all_models_pred) <- "comb_pred"
  return(all_models_pred)
}
```

Madpis_deseasonalize_time_series

Function to get the seasonal value of the time series, we also have here another function that take a model_pred object, adjusing the seasonality value (using the output from the function `Madpis_deseasonalize_time_series` and create a model_pred object that can be processed by the function Madpis_deal_with_model_pred)

```
Madpis_deseasonalize_time_series <- function(train_ts, nValid, seasonalize_type = "additive"){
  input <- train_ts
  fh <- nValid
  freq_ts <- frequency(train_ts)

  ### seasonal adjustments: (additive!!)
  #In this code we are handling the case we want to de-seasonalize the time series and get a variable 

  if (seasonalize_type == "additive"){
```

```
      Decmpose <- decompose(train_ts, type = "additive")
      deseasonalize_train_ts <- train_ts - Decmpose$seasonal
   }else if (seasonalize_type == "multiplicative"){
      Decmpose <- decompose(train_ts, type = "multiplicative")
      deseasonalize_train_ts <- train_ts / Decmpose$seasonal

   }

    #SIout takes the nValid variable, lets say it equal to 8, and produces 8 numbers, those 8 numbers a
    SIout <- head(rep(Decmpose$seasonal[(length(Decmpose$seasonal)-freq_ts+1):length(Decmpose$seasonal)]


 output_all <- list()
 output_all$deseasonalize_train_ts <- deseasonalize_train_ts
 output_all$seasonal_adj_for_train_period <-  Decmpose$seasonal
 output_all$seasonal_adj_for_validation_period <- SIout

return(output_all)
}

##### The following function will use the output of the previous function and make a "model_pred" objec
Madpis_create_lm_pred_for_seasonal_adj_model <- function(model_pred, seasonal_adj_for_train_period, seas


   if (seasonalize_type == "additive"){
      fitted_adj <- model_pred$fitted + seasonal_adj_for_train_period
      mean_adj <- model_pred$mean + seasonal_adj_for_validation_period
      upper_adj <- model_pred$upper + seasonal_adj_for_validation_period
      lower_adj <- model_pred$lower + seasonal_adj_for_validation_period
      x <- model_pred$x
   }else if (seasonalize_type == "multiplicative"){
      fitted_adj <- model_pred$fitted * seasonal_adj_for_train_period
      mean_adj <- model_pred$mean*seasonal_adj_for_validation_period
      upper_adj <- model_pred$upper*seasonal_adj_for_validation_period
      lower_adj <- model_pred$lower*seasonal_adj_for_validation_period
      x <- model_pred$x
}
   upper_adj <- tk_tbl(upper_adj)

   #upper_adj$index <- tk_tbl(model_pred)$index
   upper_adj <-  upper_adj %>% rename(
                                  "Hi 20" = "20%",
                                  "Hi 40" = "40%",
                                  "Hi 60" = "60%",
                                  "Hi 80" = "80%",
                                  "Date" = "index")

   lower_adj <- tk_tbl(lower_adj)
   #lower_adj$index <- tk_tbl(model_pred)$index
   lower_adj <-  lower_adj %>% rename(
                                  "Lo 20" = "20%",
                                  "Lo 40" = "40%",
                                  "Lo 60" = "60%",
```

```
                                     "Lo 80" = "80%",
                                     "Date" = "index")


   all_models_pred <- list()
   #here we store the "original" un adjustd values - so if we wish we can compare them to the adjusted
   all_models_pred$not_adj_values <- list()
   all_models_pred$not_adj_values$mean_not_adj <- model_pred$mean
   all_models_pred$not_adj_values$x_not_adj <- model_pred$x
   all_models_pred$not_adj_values$upper_not_adj <- model_pred$upper
   all_models_pred$not_adj_values$lower_not_adj <- model_pred$lower
   all_models_pred$not_adj_values$fitted_not_adj <- model_pred$fitted

#Here we store the adjusted values
   all_models_pred$mean <- mean_adj
   all_models_pred$x <- x
   all_models_pred$upper <- upper_adj
   all_models_pred$lower <-  lower_adj
   all_models_pred$fitted <-  fitted_adj
   class(all_models_pred) <- "deseasonalize_pred"
   return(all_models_pred)
}
```

Madpis_deal_with_model_pred

The following function gets as input the **model pred object** and creates the `training pred tib` table + the `validation pred tib` table. It will add a column for the label representing the name of the model.

Moreover The following function gets as input (besides the the lm_pred object and model name), `train_ts_tib`, `validation_ts_tib`, and option to print the accuracy table. The function enables **calculating the accuracy score for a given model**.

Note: If you don't provide the `validation_ts_tib` table to the function, it will **NOT** calculate the accuracy score.

- The function is designated to work also for the "special" models, i.e: MLP, comb, SCUM, RNN, Theta

```
Madpis_deal_with_model_pred <- function(lm_pred, train_ts_tib, validation_ts_tib = FALSE, freq_data = 4
   #print(validation_ts_tib) #roymadpis
   output <- list()

   training_model_label = paste0("Training ", model_name, " forecast")
   validation_model_label = paste0("Validation ", model_name, " forecast")

#### handling case of thetaModel
   #print(class(lm_pred)) #roymadpis
   if (any(class(lm_pred)=="thetaModel")){
      training_pred_tib <- tk_tbl(lm_pred$fitted, rename_index="Date") %>%
      mutate(group = training_model_label)

      validation_pred_tib_0 <- tk_tbl(lm_pred$mean, rename_index="Date") %>%
      mutate(group = validation_model_label)
   ### The following tibble will contain both the point-forecasts and the prediction intervals
```

```r
    validation_pred_tib <- tk_tbl(lm_pred$upper, rename_index="Date") %>%
    full_join(tk_tbl(lm_pred$lower, rename_index="Date"), by = "Date") %>%
    mutate(Date=validation_pred_tib_0$Date, value = validation_pred_tib_0$value, group = validation_mo


    ### changing the column names + reordering them
    validation_pred_tib <- validation_pred_tib %>% rename("Hi 20" = "Hi 0.2",
                               "Hi 40" = "Hi 0.4",
                               "Hi 60" = "Hi 0.6",
                               "Hi 80" = "Hi 0.8",
                               "Lo 20" = "Lo 0.2",
                               "Lo 40" = "Lo 0.4",
                               "Lo 60" = "Lo 0.6",
                               "Lo 80" = "Lo 0.8"
                               ) %>% select(c("Date", "Lo 20", "Hi 20",
                                          "Lo 40", "Hi 40", "Lo 60",
                                          "Hi 60", "Lo 80", "Hi 80",
                                          "value", "group"))



    ### if the model is mlp:
  }else if(any(class(lm_pred)[1] =="forecast.net") || any(class(lm_pred) == "SCUM") || any(class(lm_pre
  training_pred_tib <- tk_tbl(lm_pred$fitted, rename_index="Date") %>%
  mutate(group = training_model_label)

  validation_pred_tib <- tk_tbl(lm_pred$mean, rename_index="Date") %>%
  mutate(group = validation_model_label)
  validation_pred_tib_0 <- validation_pred_tib
    if (any(class(lm_pred) == "SCUM") || any(class(lm_pred)=="RNN")){
      ### add the date column to the train and validation tables
      training_pred_tib <- training_pred_tib %>% mutate("Date" = lm_pred$train_dates) %>% select(c("I

      validation_pred_tib <- validation_pred_tib %>% mutate("Date" = lm_pred$test_dates) %>% select(c
      validation_pred_tib_0 <- validation_pred_tib
    }

  # if the model is combination of models - from function madpis_create_lm_pred_3_models
  }else if (any(class(lm_pred) == "comb_pred") || any(class(lm_pred) == "deseasonalize_pred")){
  training_pred_tib <- tk_tbl(lm_pred$fitted, rename_index="Date") %>%
  mutate(group = training_model_label)

  validation_pred_tib_0 <- tk_tbl(lm_pred$mean, rename_index="Date") %>%
  mutate(group = validation_model_label)

    if ("lower" %in% names(lm_pred)){
      validation_pred_tib <- full_join(lm_pred$upper, lm_pred$lower)
      validation_pred_tib <- full_join(validation_pred_tib, validation_pred_tib_0) %>% mutate(group =
    }else{
    validation_pred_tib <- mutate(validation_pred_tib_0, group = validation_model_label)
  }

  } else{
    training_pred_tib <- tk_tbl(lm_pred$fitted, rename_index="Date") %>%
```

```
      mutate(group = training_model_label)

    validation_pred_tib_0 <- tk_tbl(lm_pred$mean, rename_index="Date") %>%
      mutate(group = validation_model_label)

  ### The following tibble will contain both the point-forecasts and the prediction intervals

    validation_pred_tib <- tk_tbl(lm_pred, rename_index="Date") %>%
      mutate(Date=validation_pred_tib_0$Date, value = `Point Forecast`,
             group = validation_model_label)
}
  colnames(training_pred_tib)[2] <- "value" #change the name of the second column to "value"

  #drop the point forecast column from the validation tib
  drop <- c("Point Forecast")
  validation_pred_tib <-  validation_pred_tib[,!(names(validation_pred_tib) %in% drop)]

  ##################### Calculating Accuracy
  #print(validation_ts_tib) #roymadpis
  #view(validation_ts_tib)
  if (typeof(validation_ts_tib) != "logical"){
  #if (validation_ts_tib != FALSE){
   year_validation <- as.integer(substring(train_ts_tib$Date[nrow(train_ts_tib)], 1, 2))

  quarter_validation <- as.integer(substring(train_ts_tib$Date[nrow(train_ts_tib)], 5, 6))
  if (is.na(quarter_validation)){
      quarter_validation <- as.integer(substring(train_ts_tib$Date[nrow(train_ts_tib)], 4, 5))
  }
  if (freq_data == 4){
    if (quarter_validation<=3){
    year_validation <- year_validation
    quarter_validation <- quarter_validation + 1
  }else{ #quarter_validation == 4
    year_validation <- year_validation
    year_validation <- year_validation+1
    quarter_validation <- 1
  }


  }else if (freq_data == 12){
      if (quarter_validation<=11){
    year_validation <- year_validation
    quarter_validation <- quarter_validation + 1
  }else{ #quarter_validation == 12
    year_validation <- year_validation
    year_validation <- year_validation+1
    quarter_validation <- 1
  }
  }

  start_date_validation <- c(year_validation, quarter_validation)

  ts_validation_accuracy <- ts(validation_ts_tib$value, frequency = freq_data, start = start_date_vali
```

```
# if the model is combination of models - from function madpis_create_lm_pred_3_models
if (any(class(lm_pred) == "comb_pred") || any(class(lm_pred) == "deseasonalize_pred") || any(class(lm
   library(Metrics)
   #generating a tibble that includes the train fitted values and the true train values - thats for

   train_ped_tib_for_accuracy <- training_pred_tib %>% rename("fitted" = "value") %>% mutate("value"

   train_ped_tib_for_accuracy <- na.omit(train_ped_tib_for_accuracy)

   RMSE_test <- MLmetrics::RMSE(y_pred = lm_pred$mean, ts_validation_accuracy)
   RMSE_train <- MLmetrics::RMSE(y_pred = train_ped_tib_for_accuracy$fitted, train_ped_tib_for_accura

   MAE_test <- MLmetrics::MAE(y_pred = lm_pred$mean, ts_validation_accuracy)
   MAE_train <- MLmetrics::MAE(y_pred = train_ped_tib_for_accuracy$fitted, train_ped_tib_for_accuracy

   MAPE_test <- MLmetrics::MAPE(y_pred = lm_pred$mean, ts_validation_accuracy)*100
   MAPE_train <- MLmetrics::MAPE(y_pred = train_ped_tib_for_accuracy$fitted, train_ped_tib_for_accura

   MASE_test <- Metrics::mase(lm_pred$mean, ts_validation_accuracy)
   MASE_train <- Metrics::mase(train_ped_tib_for_accuracy$fitted, train_ped_tib_for_accuracy$value)

   accuracy_tibble <- tibble("Model" = c(model_name, model_name),
          "Set" = c("Trainin set", "Test Set"),
          "ME" = c(NA, NA),
          "RMSE" = c(RMSE_train, RMSE_test),
          "MAE" = c(MAE_train, MAE_test),
          "MPE" = c(NA,NA),
          "MAPE" = c(MAPE_train, MAPE_test),
          "MASE" = c(MASE_train,MASE_test),
          "ACF1" = c(NA,NA),
          "Theil's U" = c(NA,NA))
   }else{
      accuracy_tibble <- as_tibble(forecast::accuracy(lm_pred, ts_validation_accuracy)) %>% mutate("

}

if (verbose == TRUE) {print(accuracy_tibble)}

output$accuracy_tibble <- accuracy_tibble

}


training_pred_tib <- mutate(training_pred_tib, "model" = model_name)
validation_pred_tib <- mutate(validation_pred_tib, "model" = model_name)

output$training_pred_tib <- training_pred_tib
output$validation_pred_tib_0 <- validation_pred_tib_0
output$validation_pred_tib <- validation_pred_tib

return(output)
```

```
}
```

Other functions:

Step 1 – Reading the data

Madpis_Initial_read_data

The following function will enable reading the train + test data. We need to call this function only one time!

After that we will need to call the function: ## Madpis_Second_read_data That function selects the specific time-series we want to use and performs several manipulations on it.

```
Madpis_Initial_read_data <- function(data_folder, data_freq = "Q", num_periods_in_year = 4){

   output <- list()

   if (data_freq == "Q" || data_freq == "q"){
      file_name <- "Quarterly-train.csv"
      data_location <-  paste(data_folder,file_name,sep="")
      data_quarter_train <- read_csv(data_location)

      #Quarterly - Teat
      file_name <- "Quarterly-test.csv"
      data_location <-  paste(data_folder,file_name,sep="")
      data_quarter_test <- read_csv(data_location)

      #Change the names in column V1 - we want each dataset to have a distinct name

      num_of_datasets_quarterly <- length(data_quarter_train$V1)
      dataset_names <- vector() #vector containing the new datasets names
      for (i in 1:num_of_datasets_quarterly){
         dataset_names[i] <- paste0("Data_set_", i)
      }

      data_quarter_train$V1 <- dataset_names
      data_quarter_test$V1 <- dataset_names

      data_quarter_train_test <- inner_join(data_quarter_train, data_quarter_test, by = "V1")

      ### Transposing the table so now each column represents a different data set, and each row repres

      ### Quarterly train transpose
      transpose_df <- sjmisc::rotate_df(data_quarter_train, rn = "Quarter")
      transpose_df[[1,1]] <- "Quarter"
      colnames(transpose_df) <- transpose_df[1,]
      transpose_df <- transpose_df[-c(1),]
      data_quarter_train_t <- transpose_df

      ### Quarterly Test transpose
      transpose_df <- sjmisc::rotate_df(data_quarter_test, rn = "Quarter")
      transpose_df[[1,1]] <- "Quarter"
      colnames(transpose_df) <- transpose_df[1,]
```

```
    transpose_df <- transpose_df[-c(1),]

    data_quarter_test_t <- transpose_df

    ### Quarterly Train&Test transpose
    transpose_df <- sjmisc::rotate_df(data_quarter_train_test, rn = "Quarter")
    transpose_df[[1,1]] <- "Quarter"
    colnames(transpose_df) <- transpose_df[1,]
    transpose_df <- transpose_df[-c(1),]

    data_quarter_train_test_t <- transpose_df

    output$data_quarter_train <-  data_quarter_train
    output$data_quarter_train_t <- data_quarter_train_t
    output$data_quarter_test <-  data_quarter_test
    output$data_quarter_test_t <-  data_quarter_test_t
    output$data_quarter_train_test <- data_quarter_train_test
    output$data_quarter_train_test_t <-  data_quarter_train_test_t

    output$data_freq <- data_freq
    output$num_periods_in_year <- num_periods_in_year

  }
  return(output)
}
```

Step 2 – Choose time-series + Preprocess the data + Plot

Madpis_Second_read_data

This function gets as inpput the output of `Madpis_Initial_read_data`, selects one timeseries from it (as the user specify in the argument data_set_to_load) and perfroms several manipualtions on it.

The reason we seperate the `Madpis_Initial_read_data` and `Madpis_Second_read_data` is time-saving considerations: the first function reads all the time-sereis data, and we don't want to read it 10 times…

- **plot_dataset** –> If TRUE then it will call the function `Madpis_plot_ts` and plot the time-series you specified in `data_set_to_load`

```
Madpis_Second_read_data <- function(Madpis_Initial_read_data_output,
                                    data_set_to_load = "Data_set_100",
                                    plot_dataset = TRUE,
                                    plot_decompose = TRUE, plot_seasonal = TRUE, plot_trend_lines = TRUI
                                    ){

    output <- list()
    output$Madpis_Initial_read_data_output <-  Madpis_Initial_read_data_output
    output$data_set_to_load <- data_set_to_load

    data_freq <- Madpis_Initial_read_data_output$data_freq
    num_periods_in_year <- Madpis_Initial_read_data_output$num_periods_in_year
```

```r
    data_quarter_train <- Madpis_Initial_read_data_output$data_quarter_train
    data_quarter_train_t <- Madpis_Initial_read_data_output$data_quarter_train_t
    data_quarter_test <- Madpis_Initial_read_data_output$data_quarter_test
    data_quarter_test_t <- Madpis_Initial_read_data_output$data_quarter_test_t
    data_quarter_train_test <- Madpis_Initial_read_data_output$data_quarter_train_test
    data_quarter_train_test_t <- Madpis_Initial_read_data_output$data_quarter_train_test_t

  ### selecting the specific time-series to load
  fin_data_quarter_train <- select(data_quarter_train_t, data_set_to_load) %>% na.omit()
  fin_data_quarter_train  <- as_tibble(sapply(fin_data_quarter_train, function(x) as.numeric(x))) #con

  fin_data_quarter_test <- select(data_quarter_test_t, data_set_to_load) %>% na.omit()
  fin_data_quarter_test  <- as_tibble(sapply(fin_data_quarter_test, function(x) as.numeric(x))) #conve


  output$fin_data_quarter_train <- fin_data_quarter_train
  output$fin_data_quarter_test <- fin_data_quarter_test

  #generating a time-series object for the quarterly time series

  ts_data_quarterly_train <- ts(data = fin_data_quarter_train, freq = num_periods_in_year)

  ts_data_quarterly_test <- ts(data = fin_data_quarter_test, freq = num_periods_in_year)

  #converting the time-series object to tibble
  ts_data_tib_quarterly_train <- tk_tbl(ts_data_quarterly_train, rename_index = "Date") %>% mutate(labe

  ts_data_tib_quarterly_test <- tk_tbl(ts_data_quarterly_test, rename_index = "Date") %>% mutate(label


  output$ts_data_quarterly_train <- ts_data_quarterly_train
  output$ts_data_quarterly_test <- ts_data_quarterly_test
  output$ts_data_tib_quarterly_train <- ts_data_tib_quarterly_train
  output$ts_data_tib_quarterly_test <- ts_data_tib_quarterly_test

  #generating tsibbles
  tsibble_quarterly_train <-  na.omit(as_tsibble(ts_data_quarterly_train))
  tsibble_quarterly_test <- na.omit(as_tsibble(ts_data_quarterly_test))

  output$tsibble_quarterly_train <- tsibble_quarterly_train
  output$tsibble_quarterly_test  <- tsibble_quarterly_test

  ### Using Madpis Preprocess function
  ### Madpis_preprocess

#The following function will enable getting as input a tibble with time series with first column contai

  # train tibble

  DS_Q_10 <- Madpis_preprocess(ts_data_tib_quarterly_train,
                              data_col_name = data_set_to_load,
                              new_data_col_name = "value",
                              label = "label")
```

```
  # test tibble
  DS_Q_10_test <- Madpis_preprocess(ts_data_tib_quarterly_test,
                                    data_col_name = data_set_to_load,
                                    new_data_col_name = "value",
                                    label = "label")

  # train + test tibble
  DS_Q_10_train_test <- rbind(DS_Q_10, DS_Q_10_test)

  # train + test tsibble (!)
  DS_Q_10_tsibble_train_test <- as_tsibble(ts(DS_Q_10_train_test$value, frequency = num_periods_in_year
  DS_Q_10_tsibble_train_test$label <- DS_Q_10_train_test$label

  output$DS_Q_10 <- DS_Q_10
  output$DS_Q_10_test <- DS_Q_10_test
  output$DS_Q_10_tsibble_train_test <- DS_Q_10_tsibble_train_test


  if (plot_dataset == TRUE){
    title_for_time_plot <- paste0("Time plot for ", data_set_to_load)

    #title_for_time_plot <- "Time plot for Data set 10"
    #data_freq <- "Q"
    #ts_data_tib --> DS_Q_10

    Madpis_plot_ts(ts_data_tib = DS_Q_10,
                   ts_data = "Auto",
                   column_name_with_value = "value",
                   data_freq = data_freq,
                   title_for_time_plot = title_for_time_plot,
                   subtitle_for_time_plot = "Auto",
                   plot_decompose = plot_decompose,
                   plot_seasonal = plot_seasonal,
                   plot_trend_lines = plot_trend_lines)

  }

  return (output)
  }


#loading the data - Quarterly
#Quarterly - train
```

Step 3 – Models - Madpis_Third_models

The following function will train all the relevant models, get the predictions, evaluate the performance and plot the predictions.

Defining all the Prediction models + the 4 variables:

- train_ts

- validation_ts (= fin_data_quarter_test)
- train_ts_tib (=DS_Q_10)
- validation_ts_tib (=DS_Q_10_test)
- nValid (=nrow(validation_ts_tib))

```
Madpis_Third_models <- function(Madpis_Second_read_data_output,
                                include_RNN = TRUE,
                                plot_all_models = TRUE,
                                print_accuracy_all_models = TRUE,
                                show_top_3_models = TRUE,
                                plot_MLP = FALSE,
                                write_datasets_for_RNN = TRUE,
                                dir_name_save_dataset ="datasets_for_RNN"

){

   output <- list()
   output$Madpis_Second_read_data_output <-  Madpis_Second_read_data_output
   output$Madpis_Initial_read_data_output <- Madpis_Second_read_data_output$Madpis_Initial_read_data_ou


   train_ts <- Madpis_Second_read_data_output$ts_data_quarterly_train
   validation_ts <- Madpis_Second_read_data_output$fin_data_quarter_test
   train_ts_tib <- Madpis_Second_read_data_output$DS_Q_10
   validation_ts_tib <- Madpis_Second_read_data_output$DS_Q_10_test #roymadpis
   nValid <- nrow(validation_ts_tib)
   data_set_to_load <- Madpis_Second_read_data_output$data_set_to_load


   ### Define all the models
    #### SCUM
   point_forecast_train <- array(NA, c(4, length(train_ts)))
   point_forecast_test <- array(NA, c(4, nValid))

   #1. Exponential smoothing
   sec3_ets_model <- ets(train_ts, model = "ZZZ")
   sec3_ets_model_pred <- forecast::forecast(sec3_ets_model, h = nValid,
                                             level = c(0.2, 0.4, 0.6, 0.8))
   #Saving the point forecasts of the training period and the validation period
   point_forecast_train_ets <- sec3_ets_model_pred$fitted
   point_forecast_train[1,] <- point_forecast_train_ets

   point_forecast_test_ets <- sec3_ets_model_pred$mean
   point_forecast_test[1,] <- point_forecast_test_ets

   #save the train and test "dates"
   train_dates <- tk_tbl(sec3_ets_model_pred$fitted, rename_index="Date")$Date
   test_dates <- tk_tbl(sec3_ets_model_pred$mean, rename_index="Date")$Date

   #2. Complex exponential smoothing - CES
   #library(smooth)
   sec3_ces_model_pred <- smooth::auto.ces(train_ts, h = nValid)
   point_forecast_train_ces <- sec3_ces_model_pred$fitted
   point_forecast_train[2,] <- point_forecast_train_ces
```

```
point_forecast_test_ces <- sec3_ces_model_pred$forecast
point_forecast_test[2,] <- point_forecast_test_ces

#3.  Automatic autoregressive integrated moving average (ARIMA)
sec3_ARIMA_model <- auto.arima(train_ts)
sec3_ARIMA_model_pred <- forecast::forecast(sec3_ARIMA_model, h = nValid, level = c(0.2,0.4,0.6,0.8))
point_forecast_train_Arima <- sec3_ARIMA_model_pred$fitted
point_forecast_train[3,] <- point_forecast_train_Arima

point_forecast_test_Arima <- sec3_ARIMA_model_pred$mean
point_forecast_test[3,] <- point_forecast_test_Arima

#4. Dynamic optimized theta (DOTM) (using dotm())
sec3_DOTM_model_pred <- dotm(train_ts, h=nValid, level = c(0.2,0.4,0.6,0.8))
point_forecast_train_DOTM <- sec3_DOTM_model_pred$fitted
point_forecast_train[4,] <- point_forecast_train_DOTM

point_forecast_test_DOTM <- sec3_DOTM_model_pred$mean
point_forecast_test[4,] <- point_forecast_test_DOTM

######### Performing median on the point_forecast_train and point_forecast_test arrays - on the colu

median_train_forecast <- apply(point_forecast_train, MARGIN = 2, median)
median_test_forecast <- apply(point_forecast_test, MARGIN = 2, median)

#median(point_forecast_test[,1]) #senity ceck
########## Create a list with the new model's output

model_SCUM <- list()
model_SCUM$method <- "Median of 4 models: ETS, CES, ARIMA, DOTM"
model_SCUM$mean <- median_test_forecast #enter the validation's forecast
model_SCUM$fitted <- median_train_forecast #enter the train's forecast
model_SCUM$train_dates <- train_dates
model_SCUM$test_dates <- test_dates
class(model_SCUM) <-  "SCUM" #change the class of this object for `Madpis_deal_with_model_pred` func

#################################################################
#Model 1 - Naive
naive_pred <- naive(train_ts, h = nValid, level = c(0.2, 0.4, 0.6, 0.8))
#################################################################
#Model 2 - Snaive
snaive_pred <- snaive(train_ts, h = nValid, level = c(0.2, 0.4, 0.6, 0.8))
#################################################################
#Model 3 - Like Naïve 1 but the data are seasonally adjusted (multiplicately), if needed**

#deseasonalize the data - in additive way
deseasonalize_from_madpis <- Madpis_deseasonalize_time_series(train_ts = train_ts, nValid = nValid, 
deseasonalize_train_ts <- deseasonalize_from_madpis$deseasonalize_train_ts
seasonal_adj_for_train_period <- deseasonalize_from_madpis$seasonal_adj_for_train_period
seasonal_adj_for_validation_period <- deseasonalize_from_madpis$seasonal_adj_for_validation_period
## train a naive model with the de-seasonalize data
model3_pred <- naive(deseasonalize_train_ts, h=nValid, level = c(0.2, 0.4, 0.6, 0.8))
```

```r
### transform the model_pred object we have just got by re-seasonalizing the outcome - both for the tra

  model3_pred_fin <- Madpis_create_lm_pred_for_seasonal_adj_model(
    model_pred = model3_pred,
    seasonal_adj_for_train_period = seasonal_adj_for_train_period,
    seasonal_adj_for_validation_period = seasonal_adj_for_validation_period,
    seasonalize_type = "additive")

  ##################################################################
  #Model 4 - SES = Simple Exponential Smoothing (for no trend and no seasonality)
  model_4_SES <- ets(y = train_ts, model = "ANN")
  model_4_SES_pred <- forecast::forecast(model_4_SES, h = nValid, level = c(0.2, 0.4, 0.6, 0.8))
  ##################################################################
  #Model 5 - Holt Winter - Exponential smoothing, linear trend, seasonal as per Naïve 2
  model_5a_HW <- ets(y = train_ts, model = "ZZA")
  model_5b_HW <- ets(y = train_ts, model = "ZZM")

  model_5a_HW_pred <- forecast::forecast(model_5a_HW, h = nValid, level = c(0.2, 0.4, 0.6, 0.8))

  model_5b_HW_pred <- forecast::forecast(model_5b_HW, h = nValid, level = c(0.2, 0.4, 0.6, 0.8))
  ##################################################################
    #Model 6 - **Damped - Similar to Holt but with damped extrapolation**
  #????????????????????????????
  model_6_Damped <-  ets(y = train_ts, damped = TRUE, model = "ZZZ")
  model_6_damped_pred <- forecast::forecast(model_6_Damped, h = nValid, level = c(0.2, 0.4, 0.6, 0.8))

  ##################################################################
  # Model 7 - **Theta - TBD**
  model_7_Theta_normal <- thetaf(train_ts, level = c(0.2, 0.4, 0.6, 0.8))
  #dotm(y = train_ts, h = nValid, level = c(0.2, 0.4, 0.6, 0.8))
  model_7_Theta <- forecast::forecast(model_7_Theta_normal, h = nValid)
##################################################################
  # Model 8 - **comb - Simple arithmetic average of SES, Holt and Damped. The reference benchmark **
  comb_model_pred <-  Madpis_create_lm_pred_3_models(
    model1 = model_4_SES_pred,
                           model2 = model_5a_HW_pred,
                           model3 = model_6_damped_pred,
                           weight_model1 = 1/3,
                           weight_model2 = 1/3, weight_model3 = 1/3)
  ##################################################################

  #Model 9 - **MLP** - A perceptron of a very basic architecture and parameterization. Some preprocess
  #mlp(y, m = frequency(y), hd = NULL, reps = 20, comb = c("median",
  #  "mean", "mode"), lags = NULL, keep = NULL, difforder = NULL,
  #  outplot = c(FALSE, TRUE), sel.lag = c(TRUE, FALSE),
  #  allow.det.season = c(TRUE, FALSE), det.type = c("auto", "bin",
  #  "trg"), xreg = NULL, xreg.lags = NULL, xreg.keep = NULL,
  #  hd.auto.type = c("set", "valid", "cv", "elm"), hd.max = NULL,
  #  model = NULL, retrain = c(FALSE, TRUE), ...)

  model_9_MLP <- nnfor::mlp(y = train_ts, m = num_periods_in_year)

  model_9_MLP_pred <-  forecast::forecast(model_9_MLP, h = nValid, level = c(0.2, 0.4, 0.6, 0.8))
```

```r
  if (plot_MLP == TRUE){
    plot(model_9_MLP)
    plot(model_9_MLP_pred)
  }

  ################################################################
  #Model 10 - **RNN** - A recurrent network of a very basic architecture and parameterization. Some pr

#### Using the `Madpis_deal_with_model_pred` function on all the models:

  output_i <- Madpis_deal_with_model_pred(lm_pred = naive_pred, model_name = "naive", train_ts_tib = tr

  training_pred_tib_naive <- output_i$training_pred_tib
  validation_pred_tib_0_naive <- output_i$validation_pred_tib_0
  validation_pred_tib_naive <- output_i$validation_pred_tib
  accuracy_naive <- output_i$accuracy_tibble


  if (include_RNN == TRUE){
    dataset_number <- strsplit(data_set_to_load, split = "_")[[1]][3]
    file_name_train <- paste("train_with_predictions_",dataset_number, ".csv", sep = "")

    train_with_predictions_RNN <- read_csv(file.path(dir_name_save_dataset, file_name_train))

    file_name_test <- paste("test_with_predictions_",dataset_number, ".csv", sep = "")

    test_with_predictions_RNN <- read_csv(file.path(dir_name_save_dataset, file_name_test))

    model_RNN <- list()
    model_RNN$mean <- test_with_predictions_RNN$predictions #enter the validation's forecast
    model_RNN$fitted <- train_with_predictions_RNN$predictions #enter the train's forecast
    model_RNN$train_dates <- train_dates
    model_RNN$test_dates <- test_dates
    class(model_RNN) <- "RNN"

    output_i <- Madpis_deal_with_model_pred(lm_pred = model_RNN, model_name = "RNN", train_ts_tib = tr
    training_pred_tib_RNN <- output_i$training_pred_tib
    validation_pred_tib_0_RNN <- output_i$validation_pred_tib_0
    validation_pred_tib_RNN <- output_i$validation_pred_tib
    accuracy_RNN <- output_i$accuracy_tibble

  }else{ #if NOT evaluate RNN then just put there the naive forecast
    training_pred_tib_RNN <- training_pred_tib_naive
    validation_pred_tib_0_RNN <- validation_pred_tib_0_naive
    validation_pred_tib_RNN <- validation_pred_tib_naive
    accuracy_RNN <- accuracy_naive
  }


  ################################################################
  # Model 11 - ETS - Automatic choice of best exponential smoothing using information criteria (e.g. A
  model_11_ets <- ets(y = train_ts, model = "ZZZ")
  model_11_ets_pred <- forecast::forecast(model_11_ets, h = nValid, level = c(0.2, 0.4, 0.6, 0.8))
```

```r
  #################################################################
  # Model 12 - ARIMA - Automatic choice of best ARIMA using information criteria (e.g. AICc)

  model_12_arima <- auto.arima(train_ts)
  model_12_arima_pred <- forecast::forecast(model_12_arima, h = nValid, level = c(0.2, 0.4, 0.6, 0.8))

  #################################################################
  # Model 13 - ARIMA - our ARIMA
  tsibble_data <- train_ts %>% as_tsibble()

  # use the kpss test to get the nubmer of regular and seasonal differencing need to perform in order
  ouput_madpis_hypothesis <- Madpis_kpss_test(tsibble_data = tsibble_data,
  col_name = "value", alpha = 0.05)

  order_d <- ouput_madpis_hypothesis$kpss_test$num_diff_for_stationary
  #calculate the "p" in the ARIMA(p,q,d) by finding the last integer lag that it's acf value is above
  acf_train_ts <- acf(train_ts, plot = FALSE)
  conf_interval_acf <- ggfortify:::confint.acf(acf_train_ts)
  order_p <- max(acf_train_ts$lag[acf_train_ts$acf>conf_interval_acf])

  seasonal_D <-  ouput_madpis_hypothesis$kpss_test$num_seasonal_diff_for_stationary


  model_13_arima <- Arima(y = train_ts, order = c(order_p, order_d, 0), seasonal = c(0,seasonal_D, 0),
  model_13_arima_pred <- forecast::forecast(model_13_arima, h = nValid, level = c(0.2, 0.4, 0.6, 0.8))

#################################################################



#### Using the `Madpis_deal_with_model_pred` function on all the models:

  #num_periods_in_year <- 4
  #Model 1 - naive dataframe
  output_i <- Madpis_deal_with_model_pred(lm_pred = naive_pred, model_name = "naive", train_ts_tib = t

  training_pred_tib_naive <- output_i$training_pred_tib
  validation_pred_tib_0_naive <- output_i$validation_pred_tib_0
  validation_pred_tib_naive <- output_i$validation_pred_tib
  accuracy_naive <- output_i$accuracy_tibble

  ### IMPORTNAT! CHANGE THE VALUES IN THE DATE COLUMN IN validation_ts_tib!!!
  validation_ts_tib$Date <-validation_pred_tib_naive$Date

  #Model 2 - snaive dataframe
  output_i <- Madpis_deal_with_model_pred(lm_pred = snaive_pred, model_name = "snaive", train_ts_tib =
  training_pred_tib_snaive <- output_i$training_pred_tib
  validation_pred_tib_0_snaive <- output_i$validation_pred_tib_0
  validation_pred_tib_snaive <- output_i$validation_pred_tib
  accuracy_snaive <- output_i$accuracy_tibble

  #Model 3 - model3_pred
  #model3_pred_fin
  output_i <- Madpis_deal_with_model_pred(lm_pred = model3_pred_fin, model_name = "model3", train_ts_t
```

```r
training_pred_tib_model3 <- output_i$training_pred_tib
validation_pred_tib_0_model3 <- output_i$validation_pred_tib_0
validation_pred_tib_model3 <- output_i$validation_pred_tib
accuracy_model3 <- output_i$accuracy_tibble

#model_4_SES_pred dataframe
output_i <- Madpis_deal_with_model_pred(lm_pred = model_4_SES_pred, model_name = "SES", train_ts_tib
training_pred_tib_SES <- output_i$training_pred_tib
validation_pred_tib_0_SES <- output_i$validation_pred_tib_0
validation_pred_tib_SES <- output_i$validation_pred_tib
accuracy_SES <- output_i$accuracy_tibble


#Model 5a - model_5a_HW_pred dataframe
output_i <- Madpis_deal_with_model_pred(lm_pred = model_5a_HW_pred, model_name = "HA A", train_ts_til
training_pred_tib_HA_a <- output_i$training_pred_tib
validation_pred_tib_0_HA_a <- output_i$validation_pred_tib_0
validation_pred_tib_HA_a <- output_i$validation_pred_tib
accuracy_HW_a <- output_i$accuracy_tibble

#Model 5b - model_5b_HW_pred dataframe
output_i <- Madpis_deal_with_model_pred(lm_pred = model_5b_HW_pred, model_name = "HA B", train_ts_til
training_pred_tib_HA_b <- output_i$training_pred_tib
validation_pred_tib_0_HA_b <- output_i$validation_pred_tib_0
validation_pred_tib_HA_b <- output_i$validation_pred_tib
accuracy_HW_b <- output_i$accuracy_tibble


#Model 6 - model_6_Damped dataframe
output_i <- Madpis_deal_with_model_pred(lm_pred = model_6_damped_pred, model_name = "Damped", train_
training_pred_tib_damped <- output_i$training_pred_tib
validation_pred_tib_0_damped <- output_i$validation_pred_tib_0
validation_pred_tib_damped <- output_i$validation_pred_tib
accuracy_Damped <- output_i$accuracy_tibble

#Model 7 - model_7_Theta dataframe
output_i <- Madpis_deal_with_model_pred(lm_pred = model_7_Theta, model_name = "Theta", train_ts_tib =
training_pred_tib_theta <- output_i$training_pred_tib
validation_pred_tib_0_theta <- output_i$validation_pred_tib_0
validation_pred_tib_theta <- output_i$validation_pred_tib
accuracy_Theta <- output_i$accuracy_tibble

#Model 8 - model_8_comb_pred dataframe (Simple arithmetic average of SES, Holt and Damped. The refer

output_i <- Madpis_deal_with_model_pred(lm_pred = comb_model_pred, model_name = "comb", train_ts_tib
training_pred_tib_comb <- output_i$training_pred_tib
validation_pred_tib_0_comb <- output_i$validation_pred_tib_0
validation_pred_tib_comb <- output_i$validation_pred_tib
accuracy_Comb <- output_i$accuracy_tibble

#Model 9 - model_9_MLP_pred dataframe
output_i <- Madpis_deal_with_model_pred(lm_pred = model_9_MLP_pred, model_name = "MLP", train_ts_tib
training_pred_tib_mlp <- output_i$training_pred_tib
```

```r
    #validation_pred_tib_0_mlp <- output_i$validation_pred_tib_0
    validation_pred_tib_mlp <- output_i$validation_pred_tib
    accuracy_MLP <- output_i$accuracy_tibble

    #Model 10 - model_10_RNN_pred dataframe
    #output_i <- Madpis_deal_with_model_pred(lm_pred = model_10_RNN_pred, model_name = "RNN", train_ts_t
    #training_pred_tib_RNN <- output_i$training_pred_tib
    #validation_pred_tib_0_RNN <- output_i$validation_pred_tib_0
    #validation_pred_tib_RNN <- output_i$validation_pred_tib
    #accuracy_RNN <- output_i$accuracy_tibble

#### see previous in the code - in the models definitions

    #Model 11 - model_11_ets_pred dataframe
    output_i <- Madpis_deal_with_model_pred(lm_pred = model_11_ets_pred, model_name = "ETS", train_ts_tib
    training_pred_tib_ETS <- output_i$training_pred_tib
    validation_pred_tib_0_ETS <- output_i$validation_pred_tib_0
    validation_pred_tib_ETS <- output_i$validation_pred_tib
    accuracy_ETS <- output_i$accuracy_tibble

    #Model 12 - model_12_arima_pred dataframe
    output_i <- Madpis_deal_with_model_pred(lm_pred = model_12_arima_pred, model_name = "AutoArima", trai
    training_pred_tib_autoArima <- output_i$training_pred_tib
    validation_pred_tib_0_autoArima <- output_i$validation_pred_tib_0
    validation_pred_tib_autoArima <- output_i$validation_pred_tib
    accuracy_AutoArima <- output_i$accuracy_tibble


    #Model 13 - model_13_arima_pred
    output_i <- Madpis_deal_with_model_pred(lm_pred = model_13_arima_pred, model_name = "Arima", train_ts
    training_pred_tib_Arima <- output_i$training_pred_tib
    validation_pred_tib_0_Arima <- output_i$validation_pred_tib_0
    validation_pred_tib_Arima <- output_i$validation_pred_tib
    accuracy_arima <- output_i$accuracy_tibble

    #Model 14 - SCUM
    output_i <- Madpis_deal_with_model_pred(lm_pred = model_SCUM, model_name = "SCUM", train_ts_tib = tra
    training_pred_tib_SCUM <- output_i$training_pred_tib
    validation_pred_tib_0_SCUM <- output_i$validation_pred_tib_0
    validation_pred_tib_SCUM <- output_i$validation_pred_tib
    accuracy_SCUM <- output_i$accuracy_tibble


    ####################################################
    ###### all together

    training_pred_tib_all <- rbind(training_pred_tib_naive,
                                   training_pred_tib_snaive,
                                   training_pred_tib_model3,
                                   training_pred_tib_SES,
                                   training_pred_tib_HA_a,
                                   training_pred_tib_HA_b,
                                   training_pred_tib_damped,
```

```r
                                  training_pred_tib_theta,
                                  training_pred_tib_comb,
                                  training_pred_tib_mlp,
                                  training_pred_tib_RNN,
                                  training_pred_tib_ETS,
                                  training_pred_tib_autoArima,
                                  training_pred_tib_Arima,
                                  training_pred_tib_SCUM)


Validation_pred_tib_all <- rbind(validation_pred_tib_naive,
                                    validation_pred_tib_snaive,
                                    validation_pred_tib_model3,
                                    validation_pred_tib_SES,
                                    validation_pred_tib_HA_a,
                                    validation_pred_tib_HA_b,
                                    validation_pred_tib_damped,
                                    validation_pred_tib_theta,
                                    validation_pred_tib_comb,
                                    validation_pred_tib_ETS,
                                    validation_pred_tib_autoArima,
                                    validation_pred_tib_Arima
                                    )

#adding the MLP validation which doesn't have prediction intervals
Validation_pred_tib_all <- full_join(Validation_pred_tib_all, validation_pred_tib_mlp, by = c("Date"
#adding the SCUM validation which doesn't have prediction intervals
Validation_pred_tib_all <- full_join(Validation_pred_tib_all, validation_pred_tib_SCUM, by = c("Date
#adding the RNN validation which doesn't have prediction intervals
Validation_pred_tib_all <- full_join(Validation_pred_tib_all, validation_pred_tib_RNN, by = c("Date"


accuracy_all <- rbind(accuracy_naive,
                       accuracy_snaive,
                       accuracy_model3,
                       accuracy_SES,
                       accuracy_HW_a, accuracy_HW_b,
                       accuracy_Damped, accuracy_Theta,
                       accuracy_Comb, accuracy_MLP,
                       accuracy_ETS, accuracy_AutoArima, accuracy_arima,
                      accuracy_SCUM, accuracy_RNN)


## combining the train and validation
train_validation_pred_tib_all <- full_join(training_pred_tib_all, Validation_pred_tib_all, by = c("Da

### adding the real data of train and validation:
train_validation_pred_tib_all <- full_join(train_validation_pred_tib_all, train_ts_tib %>% rename("g

train_validation_pred_tib_all <- full_join(train_validation_pred_tib_all, validation_ts_tib %>% renam
, by = c("Date","group", "value", "model"))

#############################################################
```

```
output$train_ts <- train_ts
output$validation_ts <-  validation_ts
output$train_ts_tib <-  train_ts_tib
output$validation_ts_tib <-  validation_ts_tib
output$nValid <- nValid

output$training_pred_tib_all <- training_pred_tib_all
output$Validation_pred_tib_all <- Validation_pred_tib_all
output$accuracy_all <- accuracy_all
output$train_validation_pred_tib_all <- train_validation_pred_tib_all

#############################################################

if (plot_all_models == TRUE){
   paste0("Number of unique models: ", length(unique(accuracy_all$Model)))
   plot_models <- ggplot(train_validation_pred_tib_all)+
geom_line(mapping=aes(x=Date, y=value, color = group))
   output$plot_models <- plot_models

   print(plot_models)
}
if (print_accuracy_all_models == TRUE){
   print(accuracy_all)
}
if(show_top_3_models == TRUE){
   Best_accuracy_models <- accuracy_all %>% filter(Set == "Test Set") %>% arrange(RMSE)
   print(Best_accuracy_models%>%head(3))
### Top 3 models according to RMSE:
# Adding "Test" and "Train" strings so we would retrieve also this data from the united tibble (of a
   top_3_models_RMSE <-  c(Best_accuracy_models$Model[1:3], "Test","Train")

   plot_top3 <- ggplot(train_validation_pred_tib_all %>% filter(model %in% top_3_models_RMSE))+
   geom_line(mapping=aes(x=Date, y=value, color = group)) +labs(title = "Top 3 Models based on RMSE")

   output$plot_top3 <- plot_top3
   output$Best_accuracy_models <- Best_accuracy_models%>%head(3)
   print(plot_top3)
}

#Creating the dir to Save the datasets for the RNN model
#+ Wrtiting the training and validation sets to csv to later read them via python for fitting an RNN

if (write_datasets_for_RNN == TRUE){
   #dir_name_save_dataset <- "datasets_for_RNN"

   if (file.exists(dir_name_save_dataset)) {
    cat("The folder already exists")
   } else {
    dir.create(dir_name_save_dataset)
   }

   ### write train data
   file_name <- paste(data_set_to_load,"_train.csv", sep = "")
```

```
        file_path <- file.path(dir_name_save_dataset, file_name)
        write.csv(train_ts_tib, file_path)

        ###write validation data
        file_name <- paste(data_set_to_load,"_test.csv", sep = "")
        file_path <- file.path(dir_name_save_dataset, file_name)
        write.csv(validation_ts_tib, file_path)

    }

    return(output)

}
```

Generating the (SCUM) model for Part 3:

We chose the article: "A simple combination of univariate models" by Fotios Petropoulos and Ivan Svetunkov. We are going to implement their modeling approach here and use it to create forecasts for the datasets.

The approach: a median combination of the point forecasts and prediction intervals of four models:

1. **Exponential Smoothing** (using ETS)
2. **Complex exponential smoothing** (CES - produces non-linear trends with a slope that depends on the data characteristics. There are both non-seasonal and seasonal versions of this model. The former allows one to slide between the level and the trend without the need for a dichotomic selection of components that is appropriate for the time series. The latter captures the type of seasonality (additive or multiplicative) and produces the appropriate forecasts, once again without the need to switch between the two option. The combination of these two models allows us to capture complex dynamics in the data. (using auto.ces())
3. **Automatic autoregressive integrated moving average** (=using auto.arima())
4. **Dynamic optimized theta** (DOTM) (using dotm())

Up until now we only decalred the functions that we are going to use in order to answer Tasks 1-3. The Following Code and Explanations are answering all three tasks,

Tasks 1 and 2

1 - Reading the whole data:

We need to run this function only one time. This will load the data into R.

```
#################### Data Location:
#Change the folder location to the location in your local computer where the data is stored:
data_folder <- "data_for_final_project/"
data_freq <-  "Q"
num_periods_in_year <- 4

###########################################
```

```r
output_Inital_Read_Data <-
    Madpis_Initial_read_data(data_folder = data_folder,
                             data_freq = data_freq,
                             num_periods_in_year = num_periods_in_year)
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## Rows: 24000 Columns: 867
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (1): V1
## dbl (737): V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16...
## lgl (129): V739, V740, V741, V742, V743, V744, V745, V746, V747, V748, V749,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 24000 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): V1
## dbl (8): V2, V3, V4, V5, V6, V7, V8, V9
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

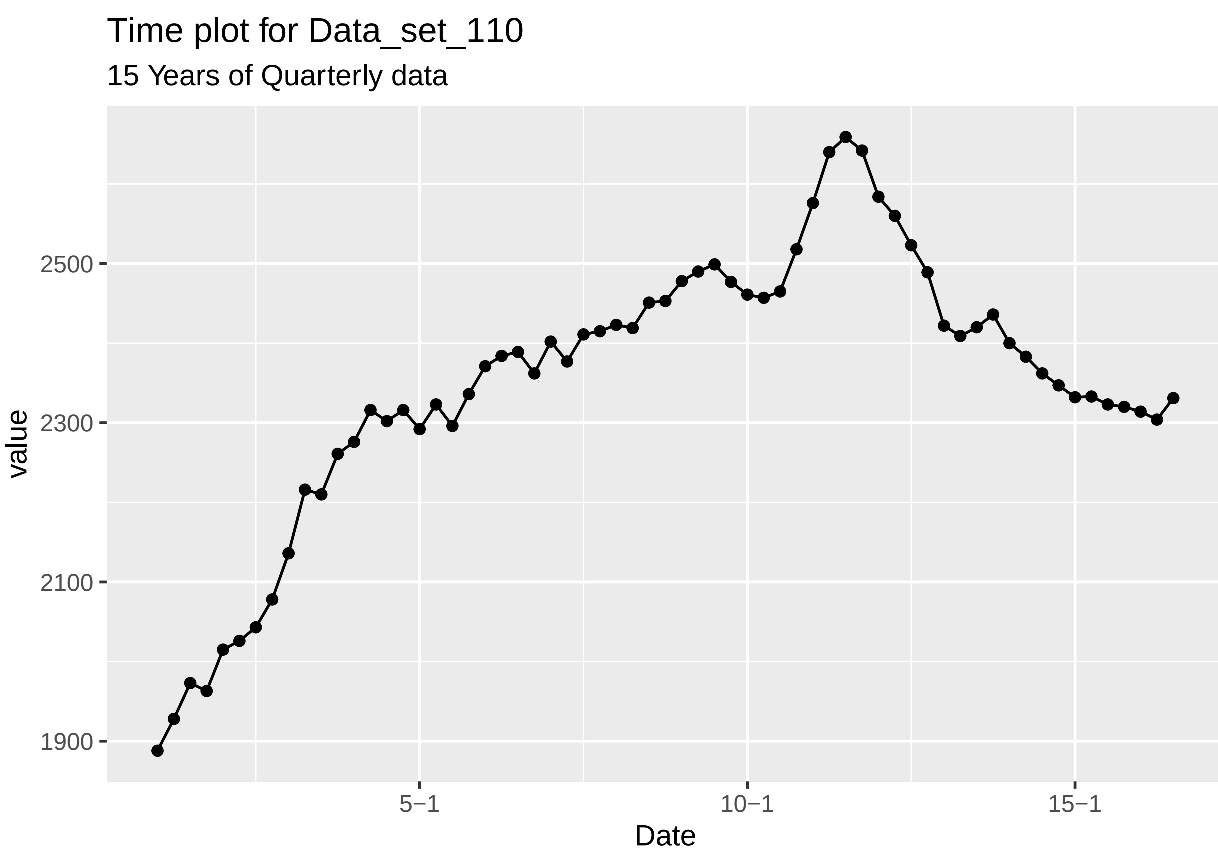
Dataset #1 - Evaluating dataset - Q110

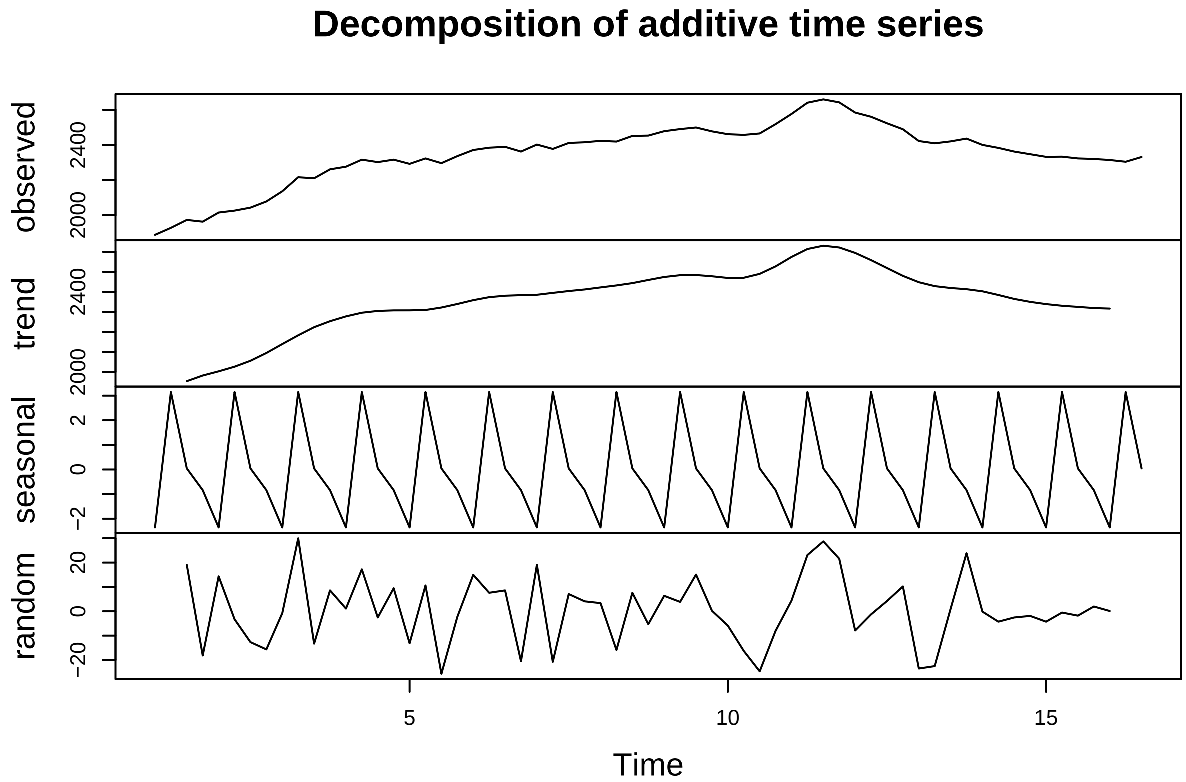
Task 1- Select dataset Q-110 and preprocess it + Plot the data

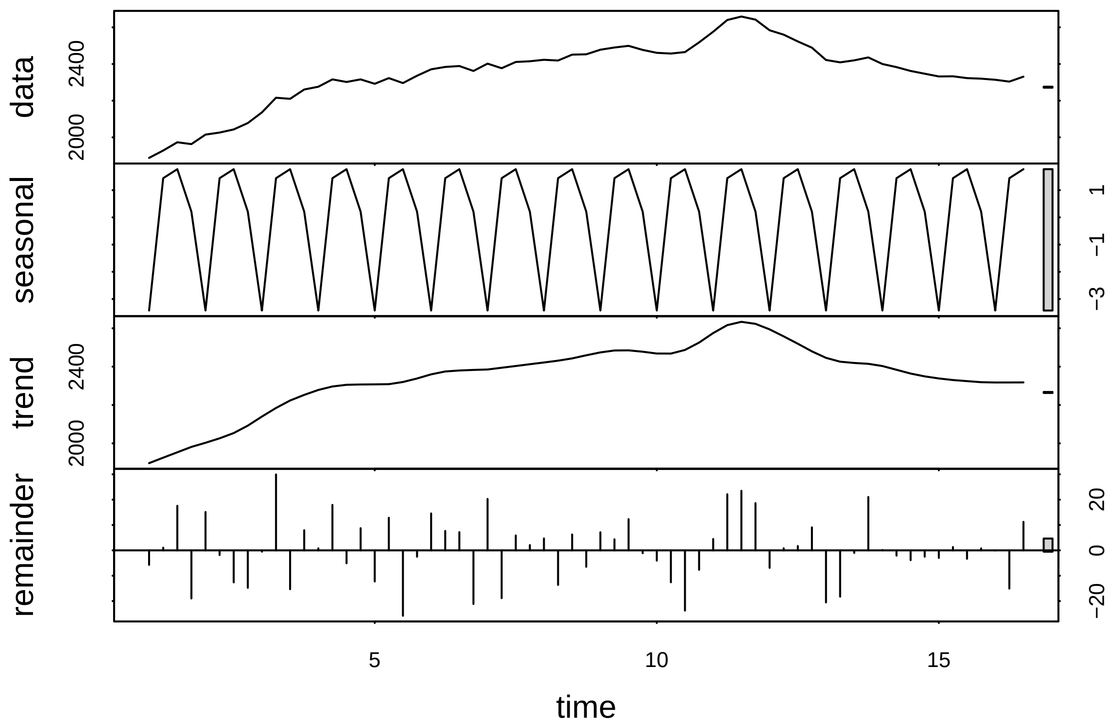
```r
data_set_to_load <- c("Data_set_110")

output_Second_Read_Data <- Madpis_Second_read_data(
   Madpis_Initial_read_data_output = output_Inital_Read_Data,
   data_set_to_load = data_set_to_load,
   plot_dataset = TRUE,
   plot_decompose = TRUE, plot_seasonal = TRUE, plot_trend_lines = TRUE)
```

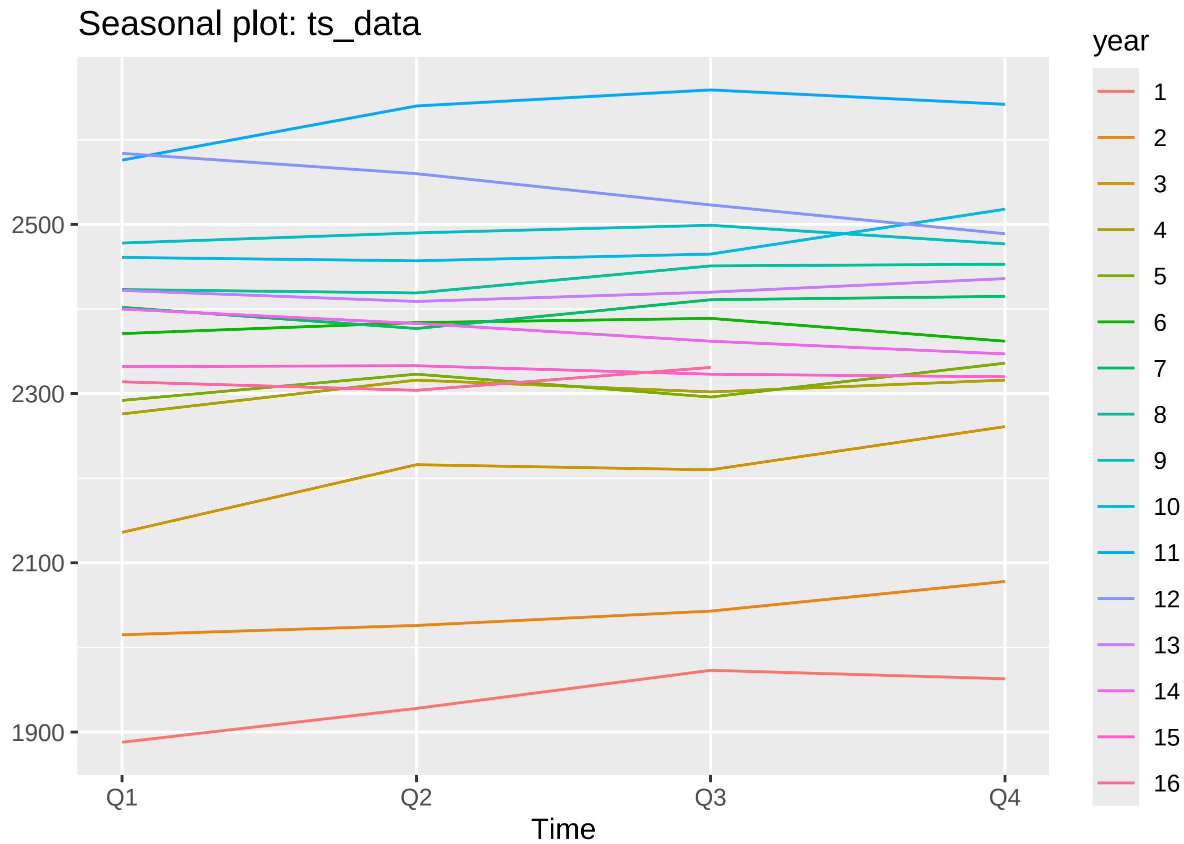
```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(data_set_to_load)
##
##   # Now:
##   data %>% select(all_of(data_set_to_load))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(data_col_name)
##
##   # Now:
##   data %>% select(all_of(data_col_name))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: The `trans` argument of `continuous_scale()` is deprecated as of ggplot2 3.5.0.
## i Please use the `transform` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

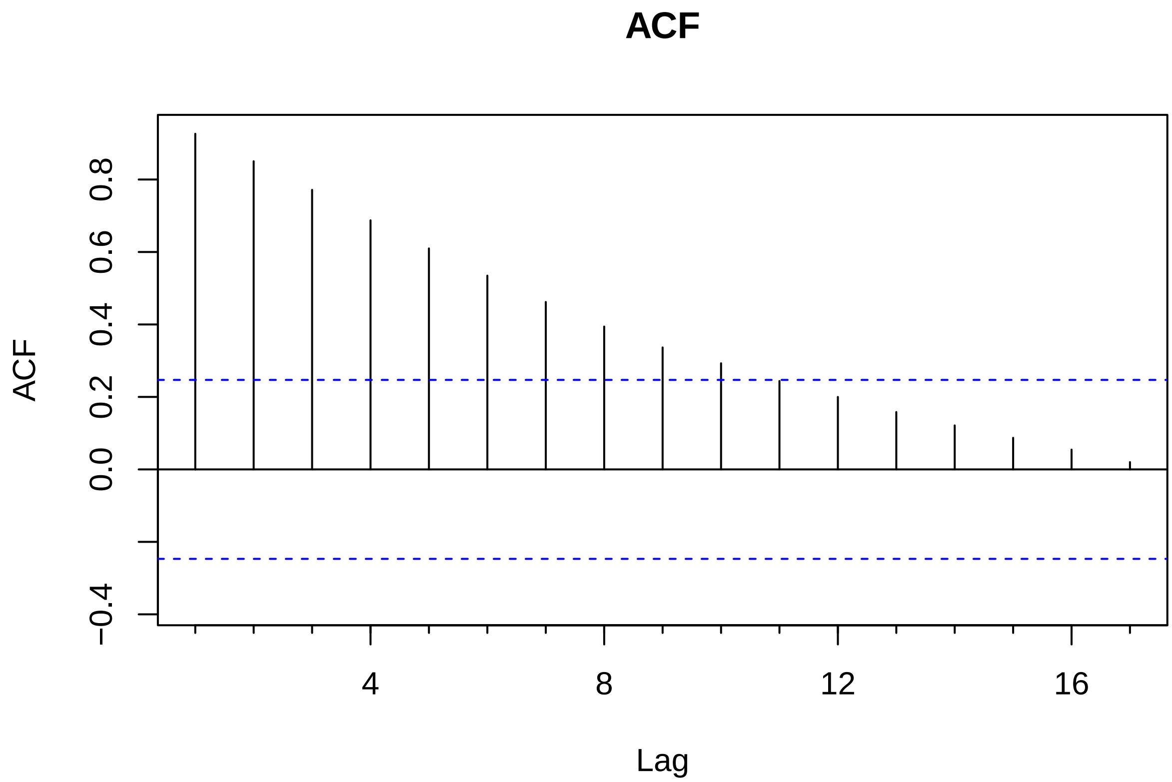


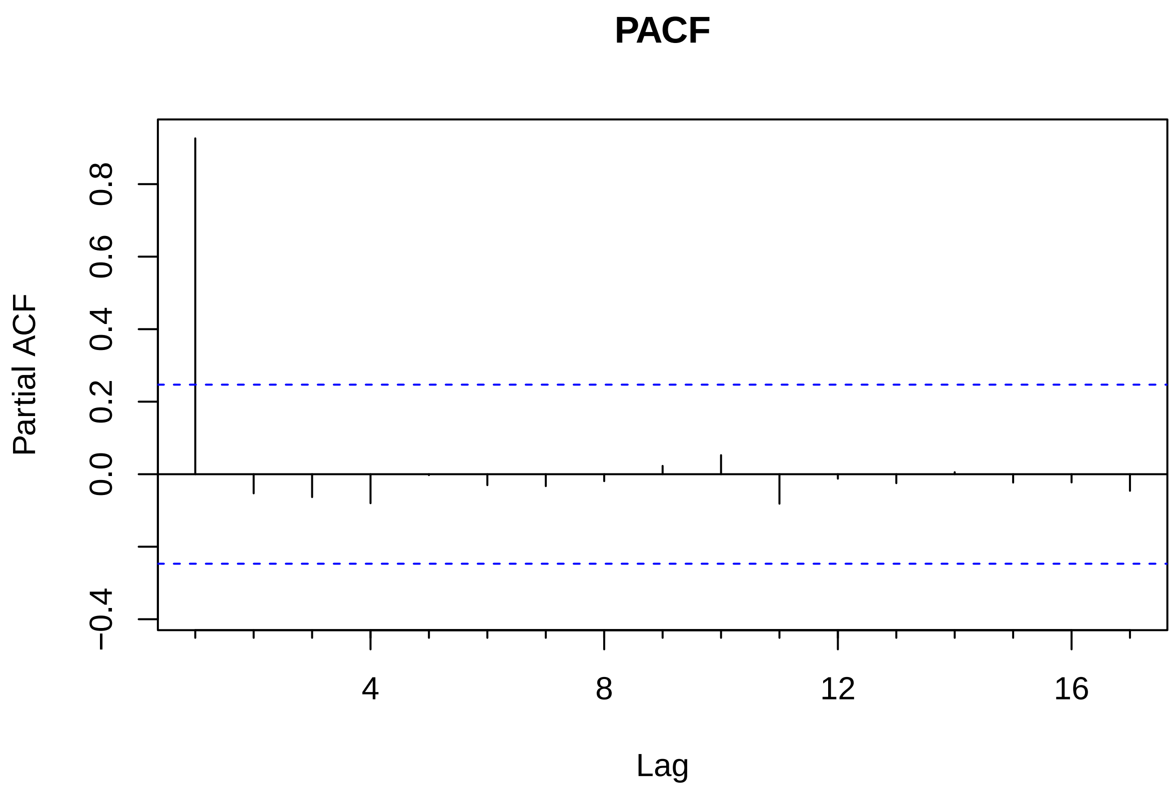

Decomposition of additive time series
observed
2000
2400
trend
2000
2400
seasonal
−2
0
2
random
−20
0
20
5
10
15
Time

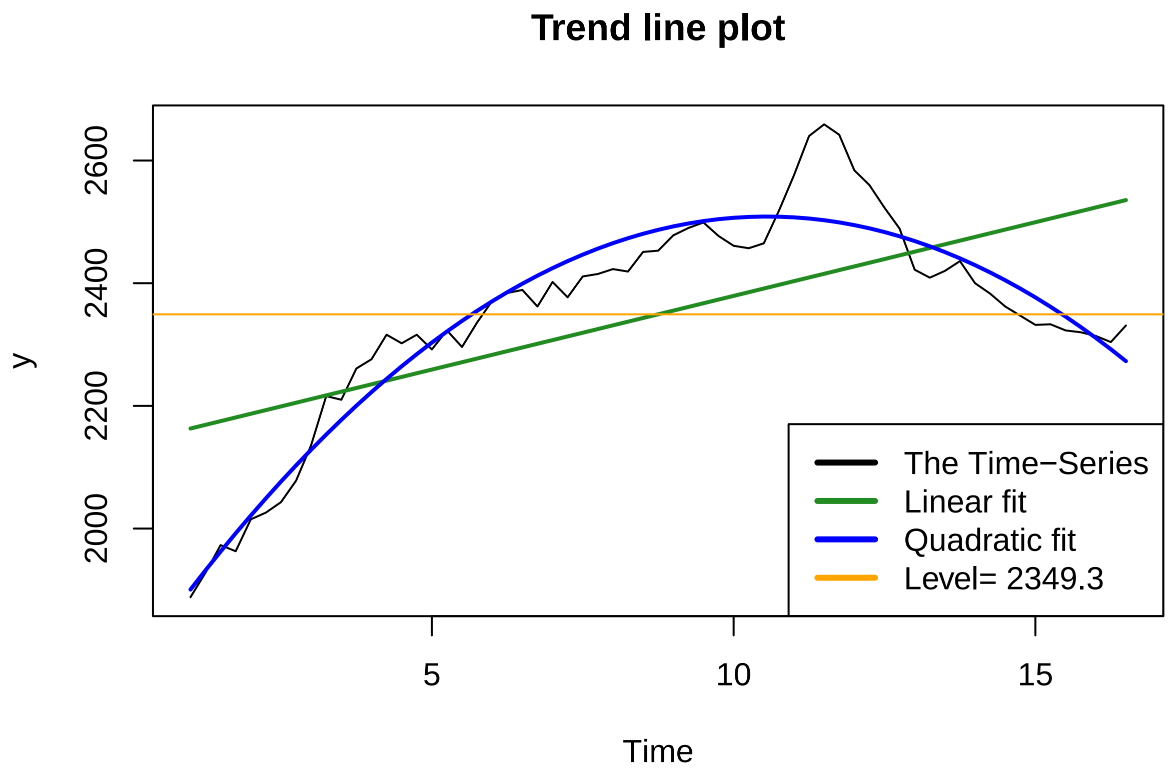

data
2000
2400
seasonal
1
−1
−3
trend
2000
2400
remainder
20
0
−20
5
10
15
time



Seasonal plot: ts_data
year
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
2500
2300
2100
1900
Q1
Q2
Q3
Q4
Time


ACF
ACF
0.8
0.6
0.4
0.2
0.0
−0.4
4
8
12
16
Lag


PACF
Partial ACF
0.8
0.6
0.4
0.2
0.0
−0.4
4
8
12
16
Lag




```
#tsibble_data <- output_Second_Read_Data$tsibble_quarterly_train
```

Notes on the time-series:

- We can see that there is **15 years of quarterly data**
- From the **Time plot** we can see that for the first 11 years there is an increase trend, afterwords - a decrease trend.
- From the **Decomposition graph** and the **seasonality graph** we can see there is seasonality in the series
- From the **ACF** and **Pacf** plots we can understand there **is an autocorrelation in the series**, the **Pacf** plot reveals that most of that autocorrelation originate from the 1st lag autocorrelation.

Task 2- Define the models + fit them + get the performance and plot

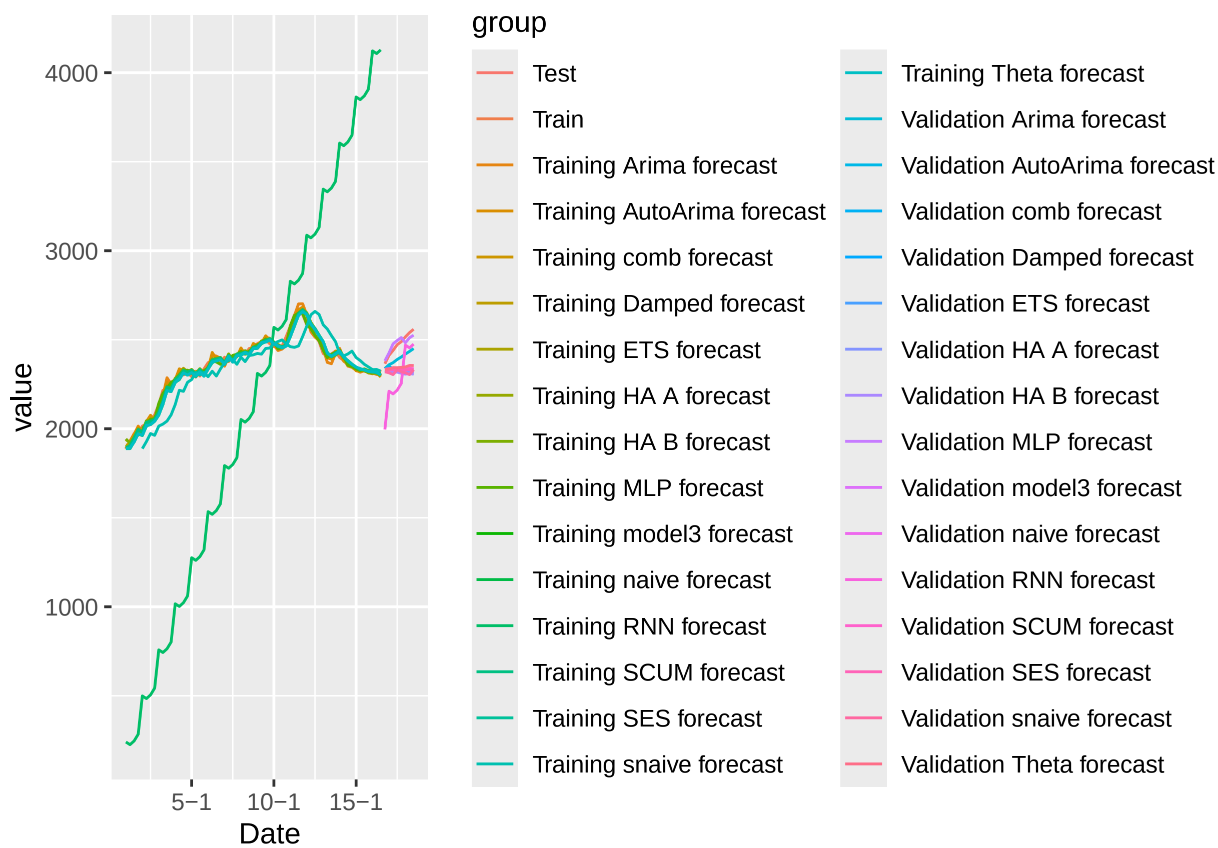
```
Madpis_Second_read_data_output <- output_Second_Read_Data

Madpis_Third_output <- Madpis_Third_models(
   Madpis_Second_read_data_output = output_Second_Read_Data,
   include_RNN = T,
   plot_all_models = TRUE,
   print_accuracy_all_models = TRUE,
```

```r
    show_top_3_models = TRUE,
    plot_MLP = FALSE, write_datasets_for_RNN = F,
    dir_name_save_dataset = "datasets_for_RNN")
```

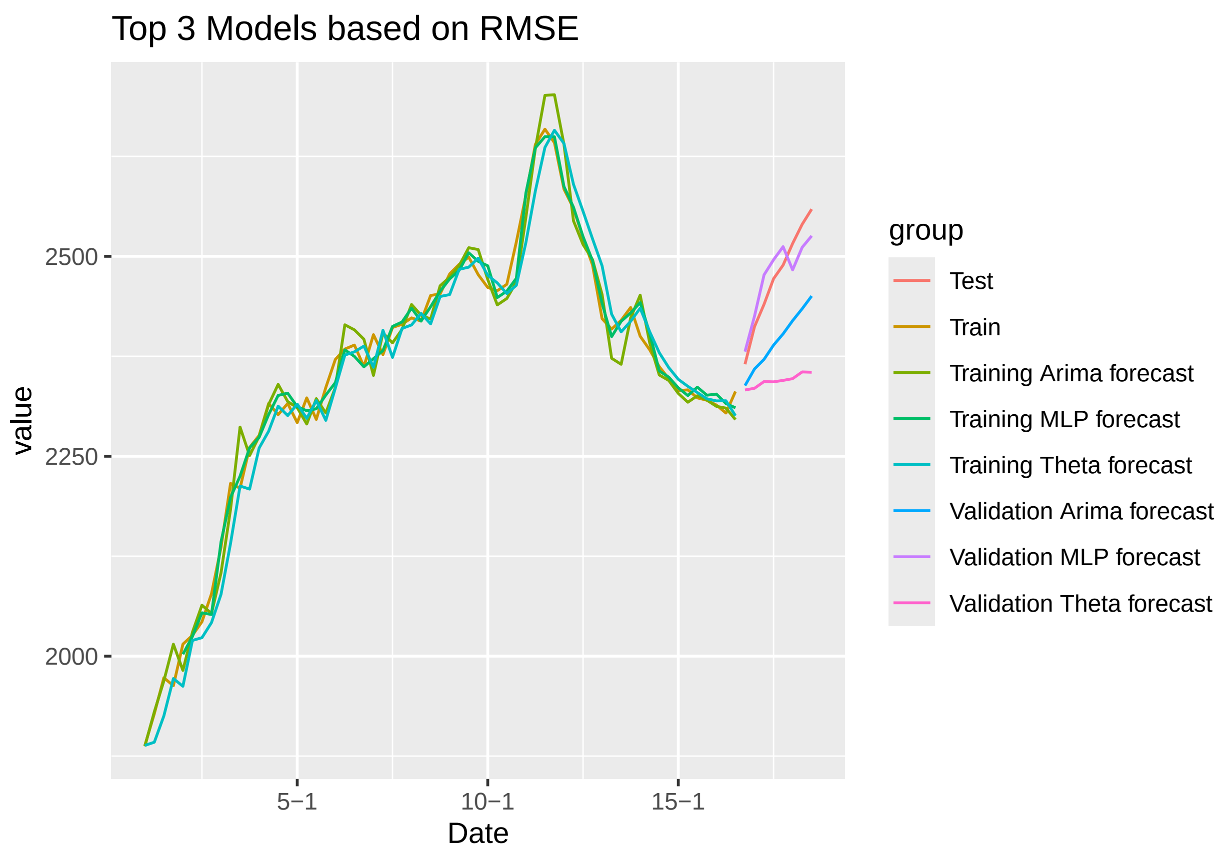
```
## New names:
## Rows: 63 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## New names:
## Rows: 8 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## Attaching package: 'Metrics'
## The following object is masked from 'package:smooth':
##
## accuracy
## The following object is masked from 'package:greybox':
##
## accuracy
## The following object is masked from 'package:generics':
##
## accuracy
## The following object is masked from 'package:fabletools':
##
## accuracy
## The following objects are masked from 'package:caret':
##
## precision, recall
## The following object is masked from 'package:forecast':
##
## accuracy
## * ‘‘ -> ‘...1‘
```

```
## [1] "KPSS Test p-value is lower than 0.05 Thus we need to reject\nH0: The data is **NOT Stationary**
```

```
## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
```

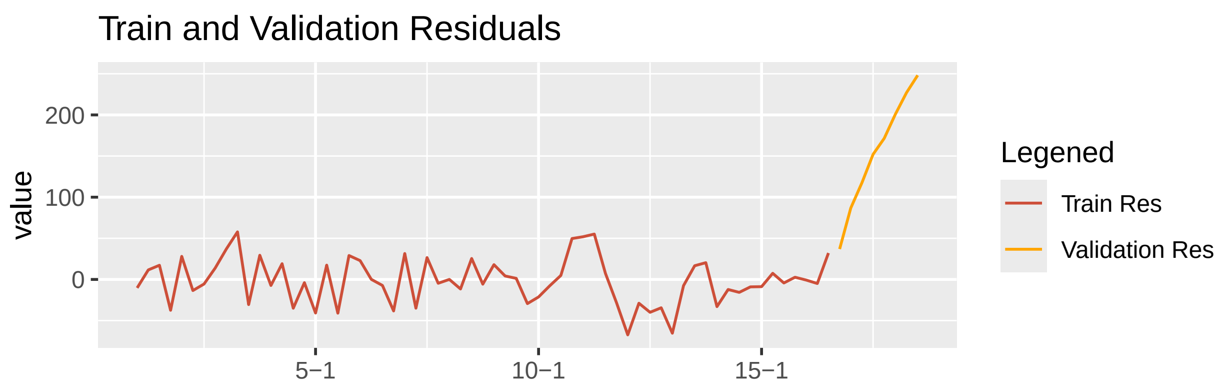


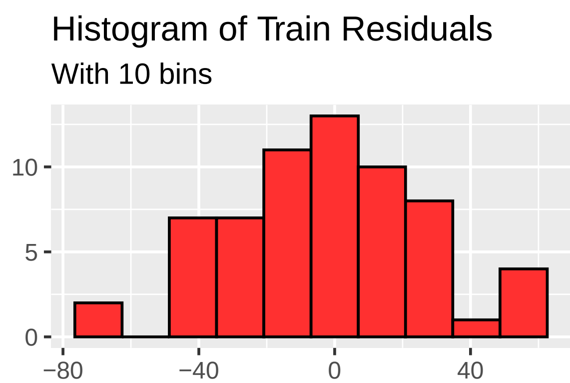

```
## # A tibble: 30 x 10
##    Model  Set            ME  RMSE   MAE    MPE  MAPE   MASE   ACF1 `Theil's U`
##    <chr>  <chr>       <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>  <dbl>       <dbl>
##  1 naive  Trainin set  7.15  31.3  25.3  0.331 1.08   0.352  0.331       NA
##  2 naive  Test Set   143.   156.  143.   5.73  5.73   1.99   0.588        5.54
##  3 snaive Trainin set  25.7  89.8  71.8  1.14  3.04   1      0.885       NA
##  4 snaive Test Set   157.   168.  157.   6.28  6.28   2.18   0.587        5.99
##  5 model3 Trainin set NA     31.2  25.5 NA     1.09   1.00  NA           NA
##  6 model3 Test Set    NA    156.  143.  NA     5.73  47.4   NA           NA
##  7 SES    Trainin set  7.03  31.1  24.9  0.326 1.06   0.346  0.327       NA
##  8 SES    Test Set   143.   156.  143.   5.73  5.73   1.99   0.588        5.54
##  9 HA A   Trainin set -2.42  28.5  23.3 -0.103 0.994  0.324  0.160       NA
## 10 HA A   Test Set   152.   166.  152.   6.10  6.10   2.12   0.579        5.90
## # i 20 more rows
## # A tibble: 3 x 10
##   Model Set            ME  RMSE   MAE    MPE  MAPE  MASE  ACF1 `Theil's U`
##   <chr> <chr>       <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>       <dbl>
## 1 MLP   Test Set  -2.16  27.2  26.0 -0.107  1.05 0.362 0.542       0.956
## 2 Arima Test Set  78.5   82.8  78.5  3.15   3.15 1.09  0.548       2.94
## 3 Theta Test Set 130.   141.  130.   5.18   5.18 1.80  0.579       5.00
```

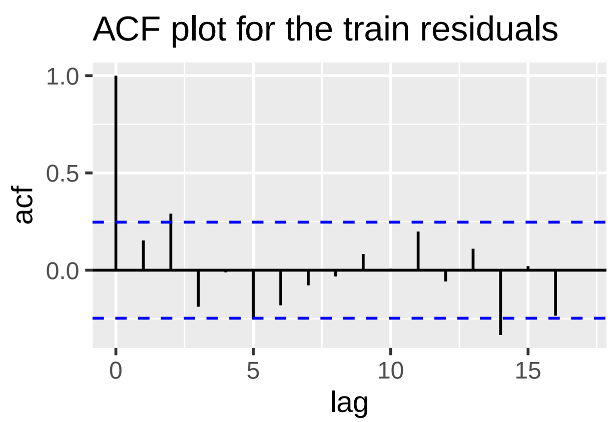


If we want we can also investigate the residuals: Lets take for example the ETS model and explore its residuals:

```
model_for_example <-  ets(y = Madpis_Third_output$train_ts, model = "ZZZ")
lm_pred <- forecast::forecast(model_for_example, h = Madpis_Third_output$nValid, c(0.2, 0.4, 0.6, 0.8))

Madpis_Residuals_plots(model = model_for_example,
                       nValid = Madpis_Third_output$nValid,
                       train_ts = Madpis_Third_output$train_ts,
                       validation_ts = Madpis_Third_output$validation_ts[[1]],
                       level = c(0.2, 0.4, 0.6, 0.8),
                       bins_for_hist = 10)
```







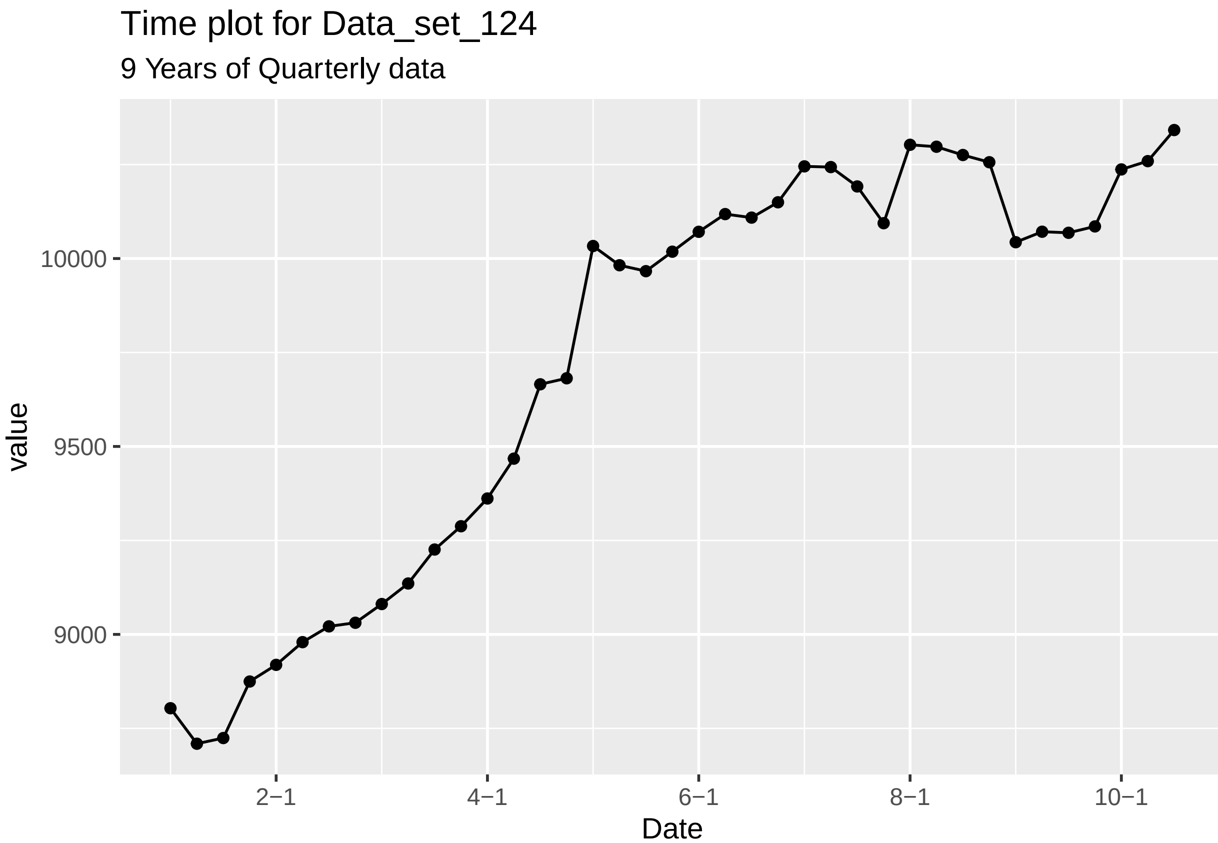
Note that we can run the above on each of the implemented models.

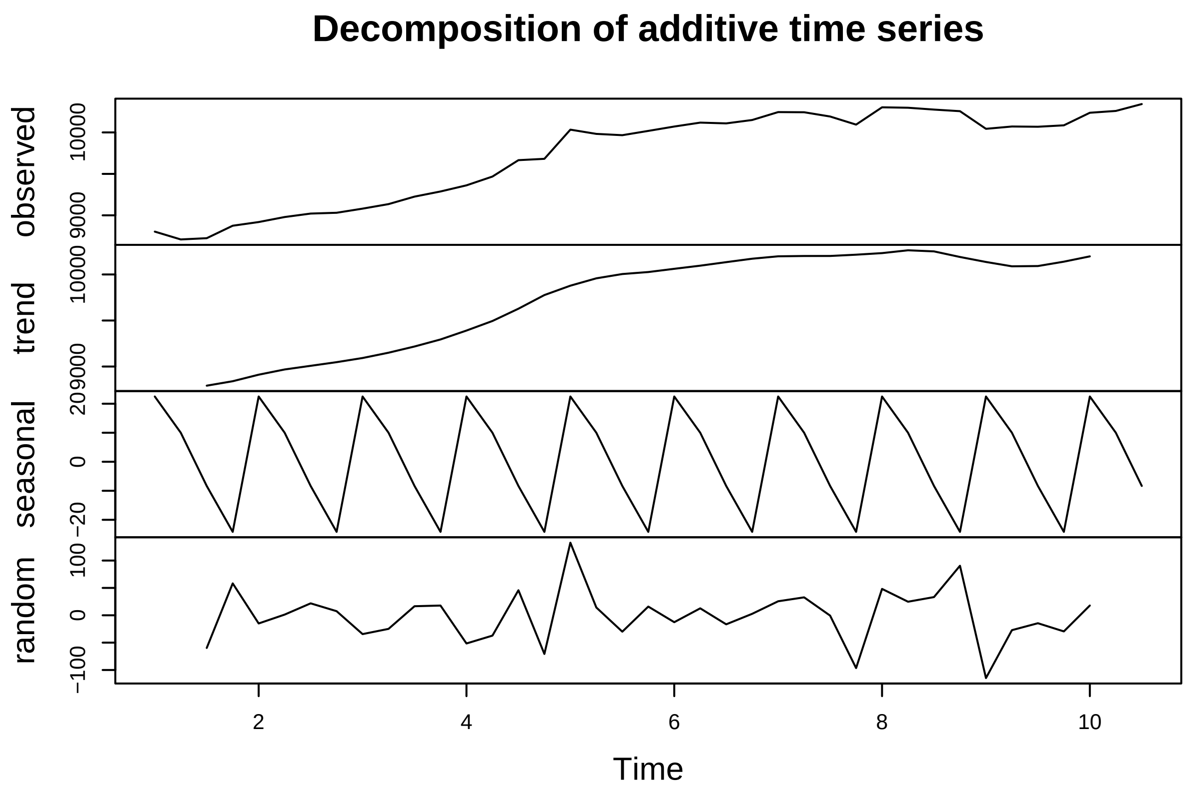
```
#Madpis_Second_read_data_output <- output_Second_Read_Data
#dir_name_save_dataset <- "datasets_for_RNN"
```

Dataset #2 - Evaluate data set Q124:

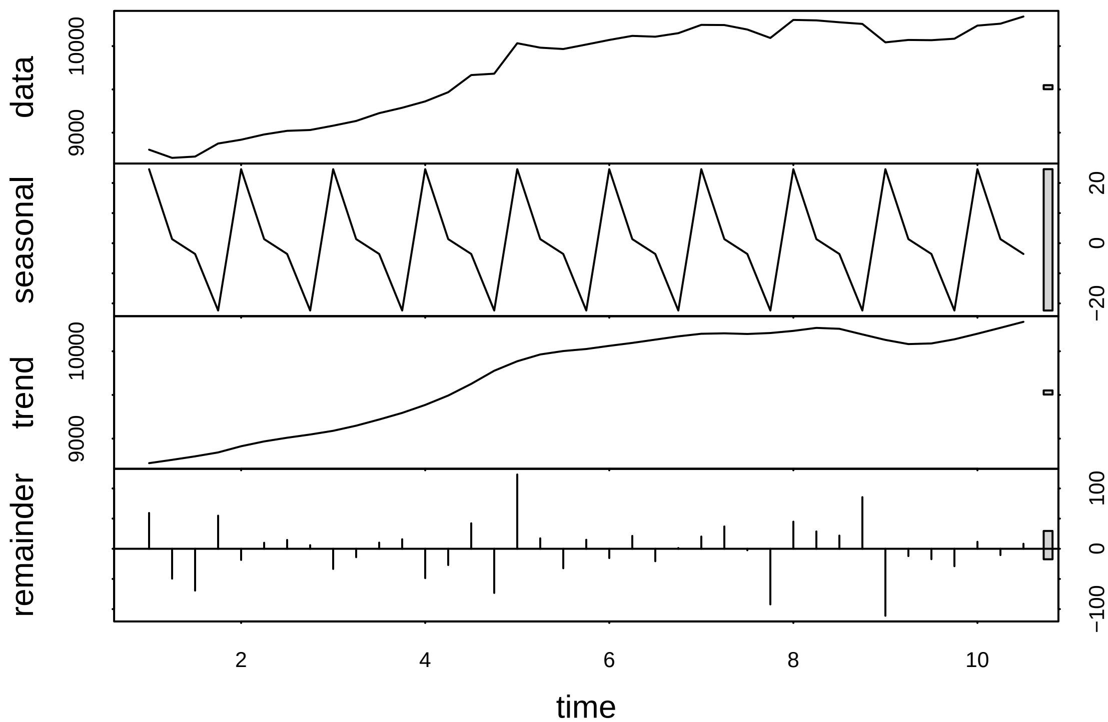
Task 1- Select dataset Q-124 and preprocess it + Plot the data

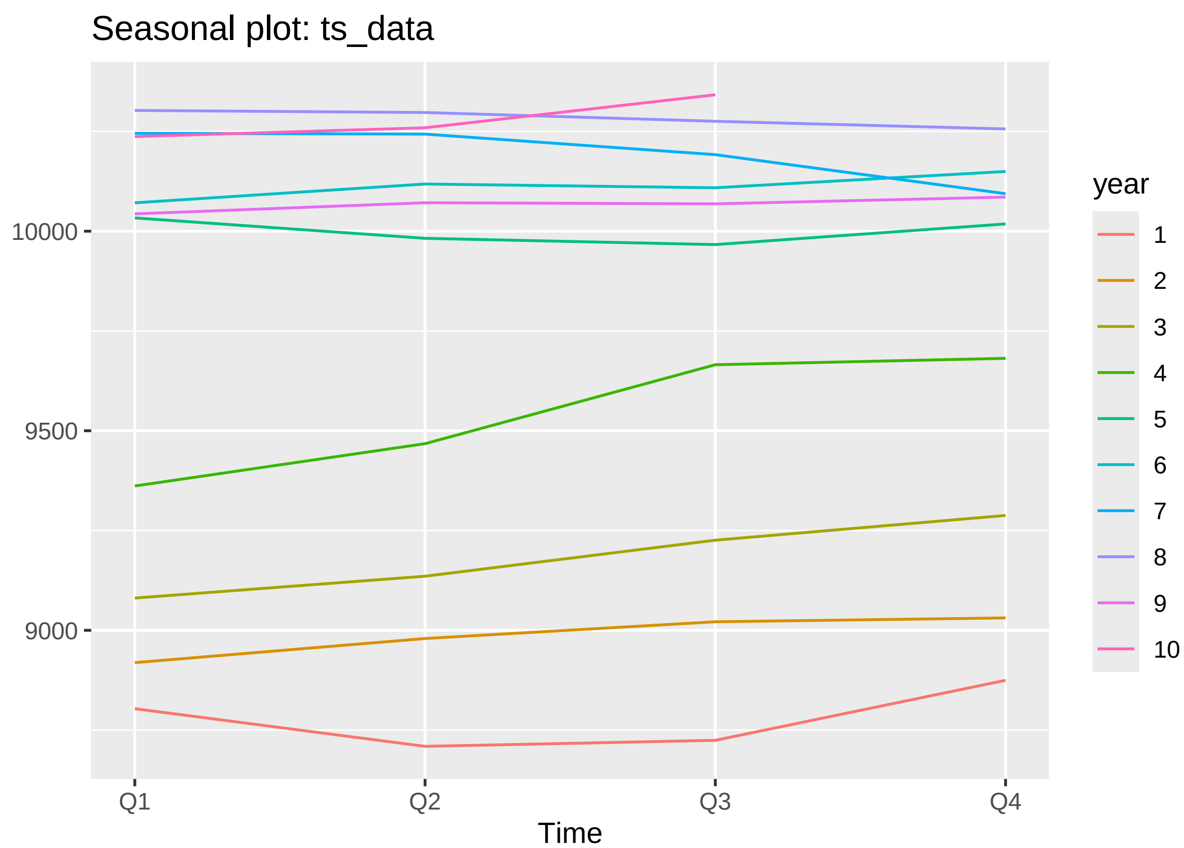
```
data_set_to_load <- c("Data_set_124")

output_Second_Read_Data <- Madpis_Second_read_data(
   Madpis_Initial_read_data_output = output_Inital_Read_Data,
   data_set_to_load = data_set_to_load,
   plot_dataset = TRUE,
   plot_decompose = TRUE, plot_seasonal = TRUE)
```

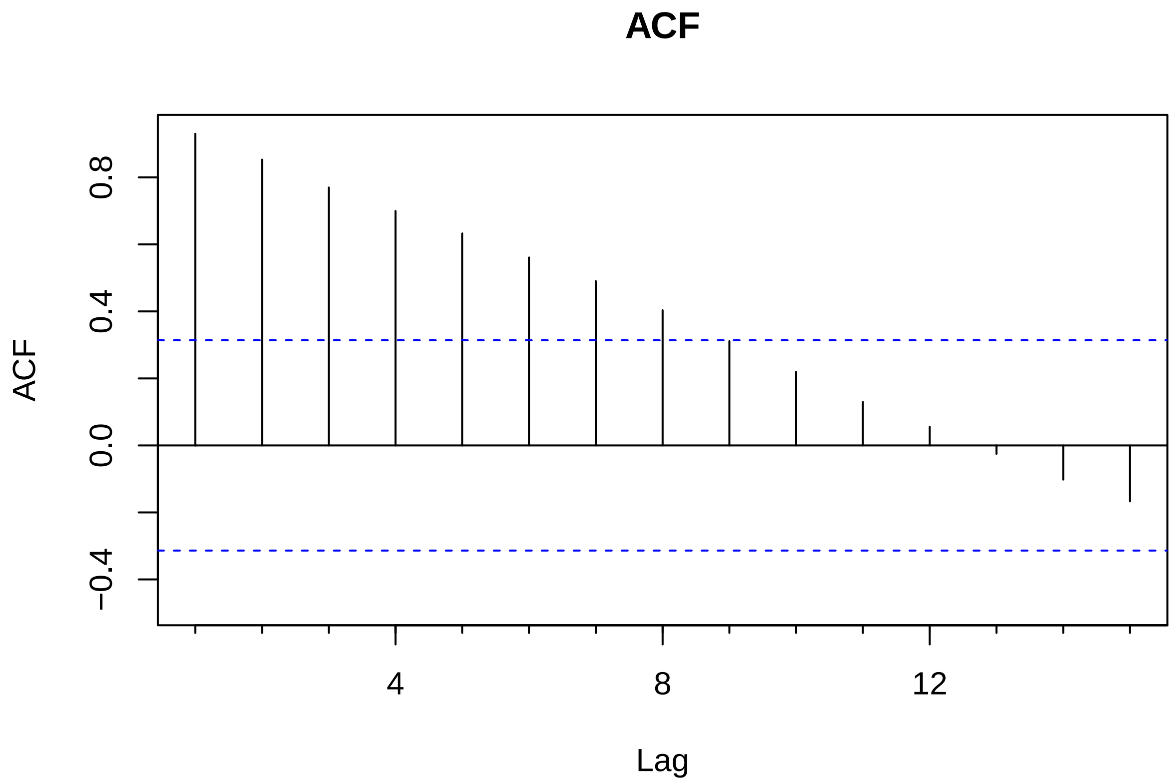

Time plot for Data_set_124
9 Years of Quarterly data
value
10000
9500
9000
2−1
4−1
6−1
8−1
10−1
Date

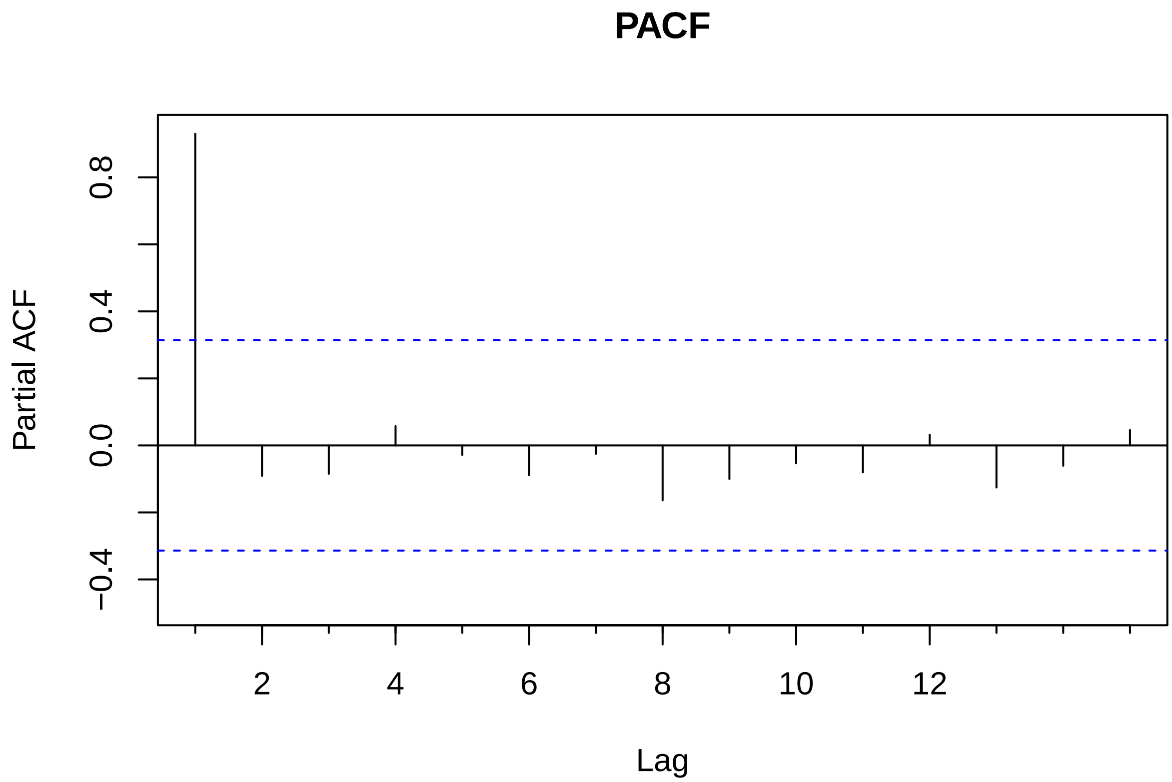


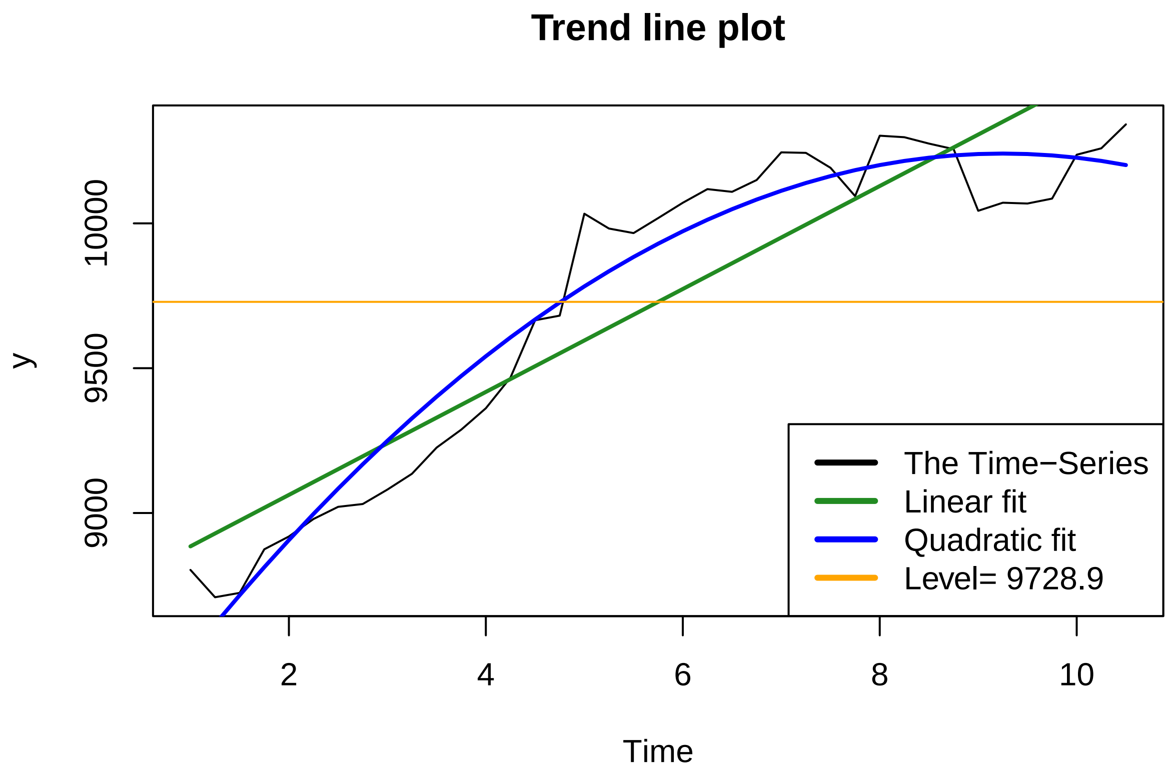
Decomposition of additive time series
observed
trend
seasonal
random
Time
10000
9000
10000
9000
20
0
−20
100
0
−100
2
4
6
8
10


data
seasonal
trend
remainder
10000
9000
20
0
−20
10000
9000
100
0
−100
2
4
6
8
10
time


Seasonal plot: ts_data
10000
9500
9000
Q1
Q2
Q3
Q4
Time
year
1
2
3
4
5
6
7
8
9
10


ACF
ACF
0.8
0.4
0.0
−0.4
4
8
12
Lag


PACF
Partial ACF
0.8
0.4
0.0
−0.4
2
4
6
8
10
12
Lag




```
#tsibble_data <- output_Second_Read_Data$tsibble_quarterly_train
```

Notes on the time-series:

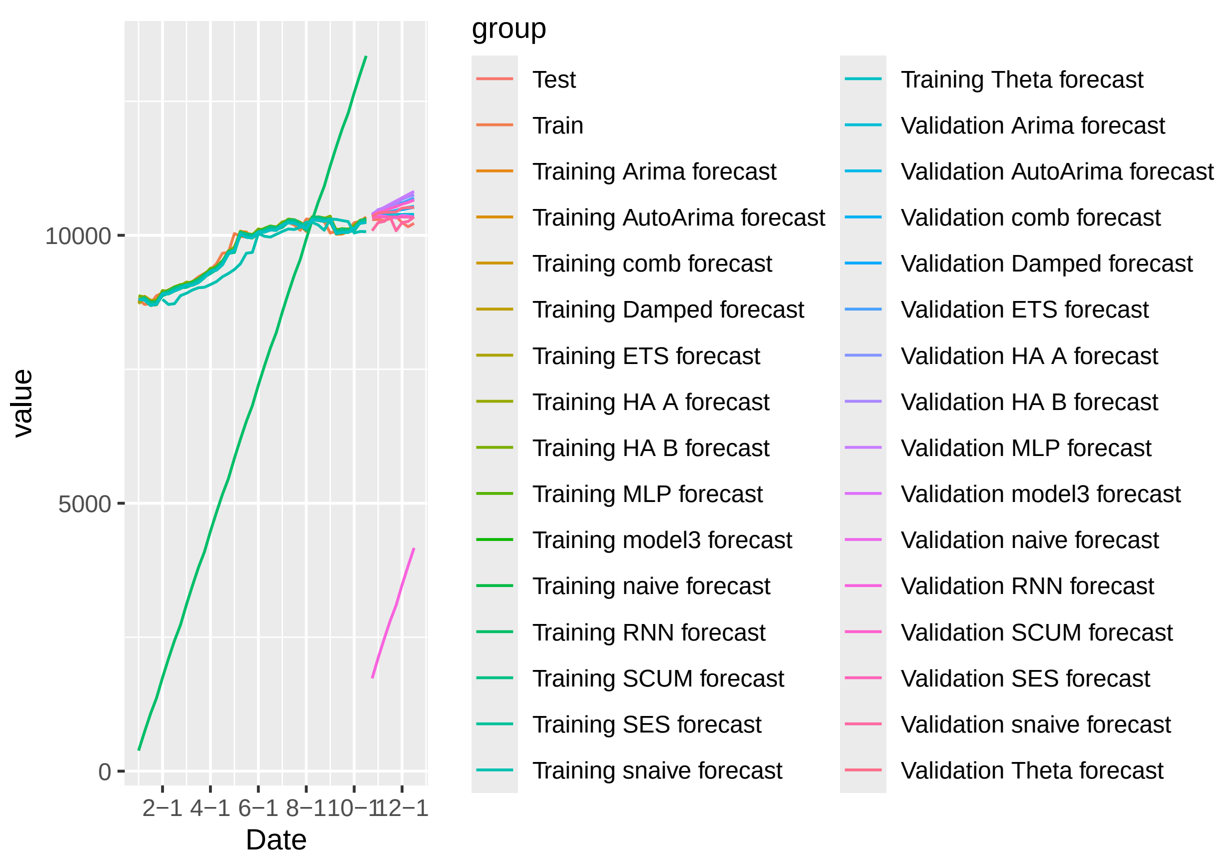
- We can see that there is **9 years of quarterly data**
- From the **Time plot** we can see that there is an overall increase trend
- From the **Decomposition graph** and the **seasonality graph** we can see there is seasonality in the series
- From the **ACF** and **Pacf** plots we can understand there **is an autocorrelation in the series**, the **Pacf** plot reveals that most of that autocorrelation originate from the 1st lag autocorrelation.

Task 2- Define the models + fit them + get the performance and plot

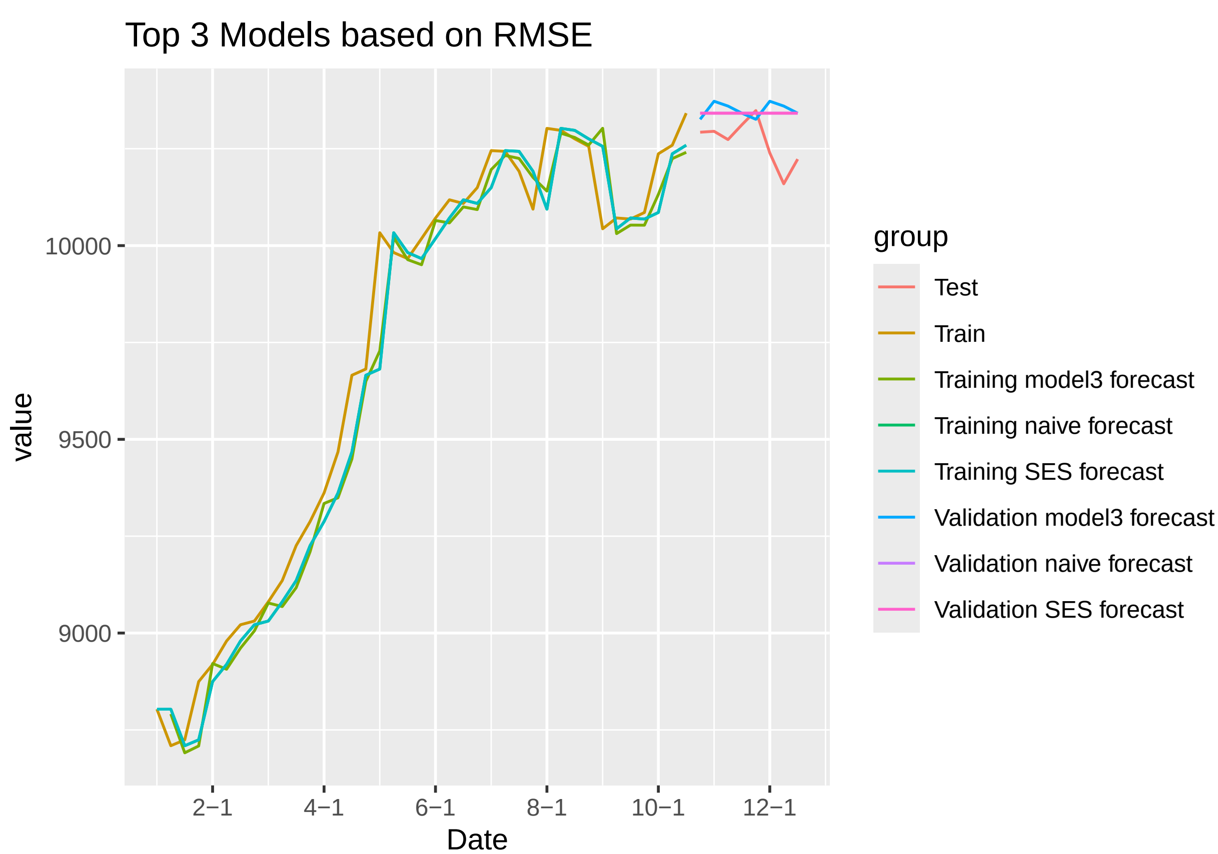
```
#Madpis_Second_read_data_output <- output_Second_Read_Data

Madpis_Third_output <- Madpis_Third_models(
   Madpis_Second_read_data_output = output_Second_Read_Data,
   include_RNN = T,
   plot_all_models = TRUE,
   print_accuracy_all_models = TRUE,
   show_top_3_models = TRUE,
```

```
    plot_MLP = FALSE, write_datasets_for_RNN = F,
    dir_name_save_dataset = "datasets_for_RNN"
)
```

```
## New names:
## Rows: 39 Columns: 8
## -- Column specification
## ---------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## New names:
## Rows: 8 Columns: 8
## -- Column specification
## ---------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘

## [1] "KPSS Test p-value is lower than 0.05 Thus we need to reject\nH0: The data is **NOT Stationary**

## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
```



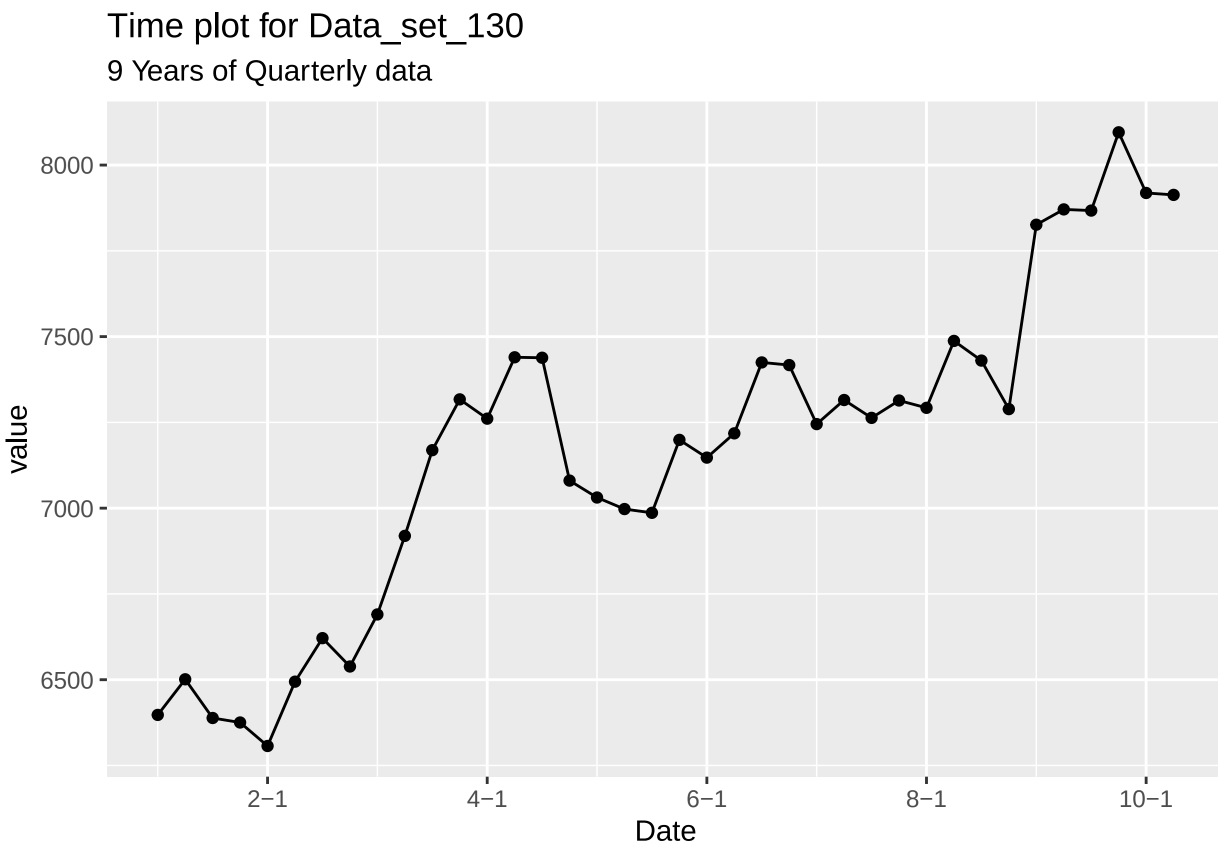
```
## # A tibble: 30 x 10
##    Model  Set            ME  RMSE   MAE    MPE  MAPE  MASE    ACF1 `Theil's U`
##    <chr>  <chr>       <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>       <dbl>
##  1 naive  Trainin set  40.5 101.   71.2  0.418 0.731 0.326 -0.0465       NA
##  2 naive  Test Set    -73.8  92.2  75.5 -0.722 0.738 0.346  0.421         1.61
##  3 snaive Trainin set 166.  257.  219.   1.72  2.24  1      0.741        NA
##  4 snaive Test Set     37.1 133.   99.1  0.357 0.964 0.454  0.0590        1.95
##  5 model3 Trainin set  NA    98.1  67.8 NA     0.698 0.946 NA            NA
##  6 model3 Test Set     NA   105.   87.8 NA     0.858 3.60  NA            NA
##  7 SES    Trainin set  39.4 100.   69.4  0.408 0.712 0.318 -0.0299       NA
##  8 SES    Test Set    -73.8  92.2  75.5 -0.722 0.738 0.346  0.421         1.61
##  9 HA A   Trainin set -11.8  89.7  61.6 -0.121 0.632 0.282 -0.0119       NA
## 10 HA A   Test Set   -313.  352.  313.  -3.06  3.06  1.43   0.612         6.22
## # i 20 more rows
## # A tibble: 3 x 10
##   Model  Set         ME  RMSE   MAE    MPE  MAPE  MASE   ACF1 `Theil's U`
##   <chr>  <chr>    <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>       <dbl>
## 1 SES    Test Set -73.8  92.2  75.5 -0.722 0.738 0.346  0.421        1.61
## 2 naive  Test Set -73.8  92.2  75.5 -0.722 0.738 0.346  0.421        1.61
## 3 model3 Test Set  NA   105.   87.8 NA     0.858 3.60  NA           NA
```

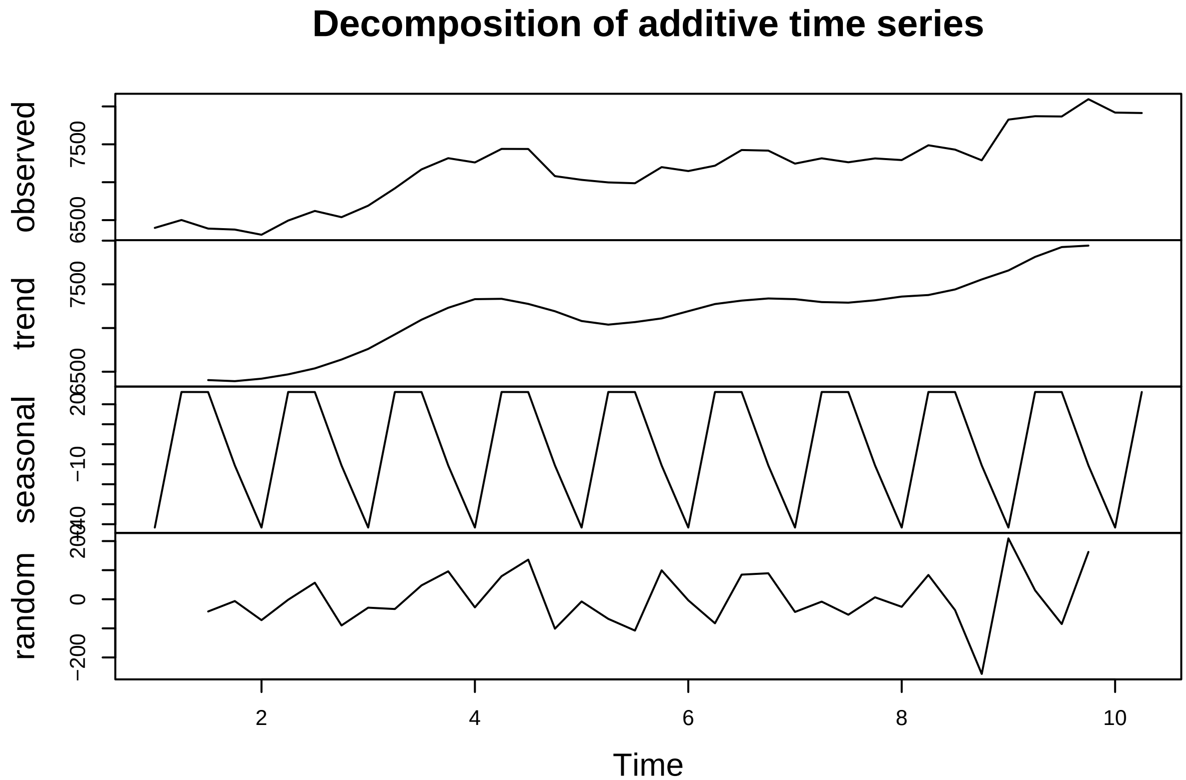


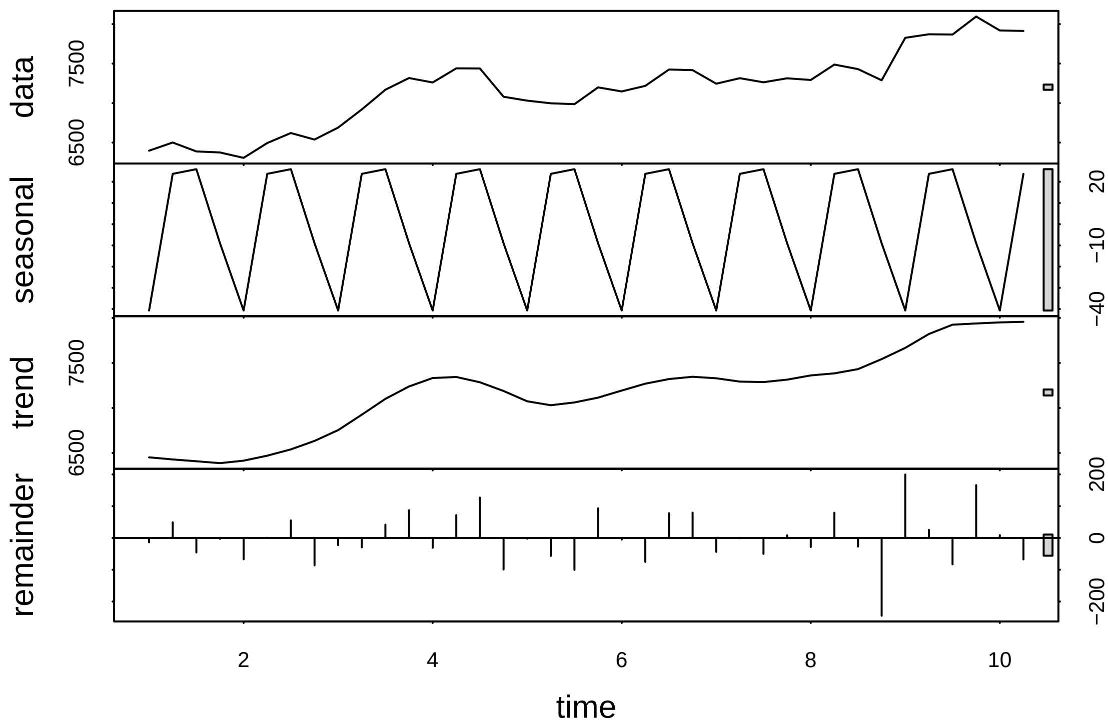
Dataset #3- Evaluate data set Q130:

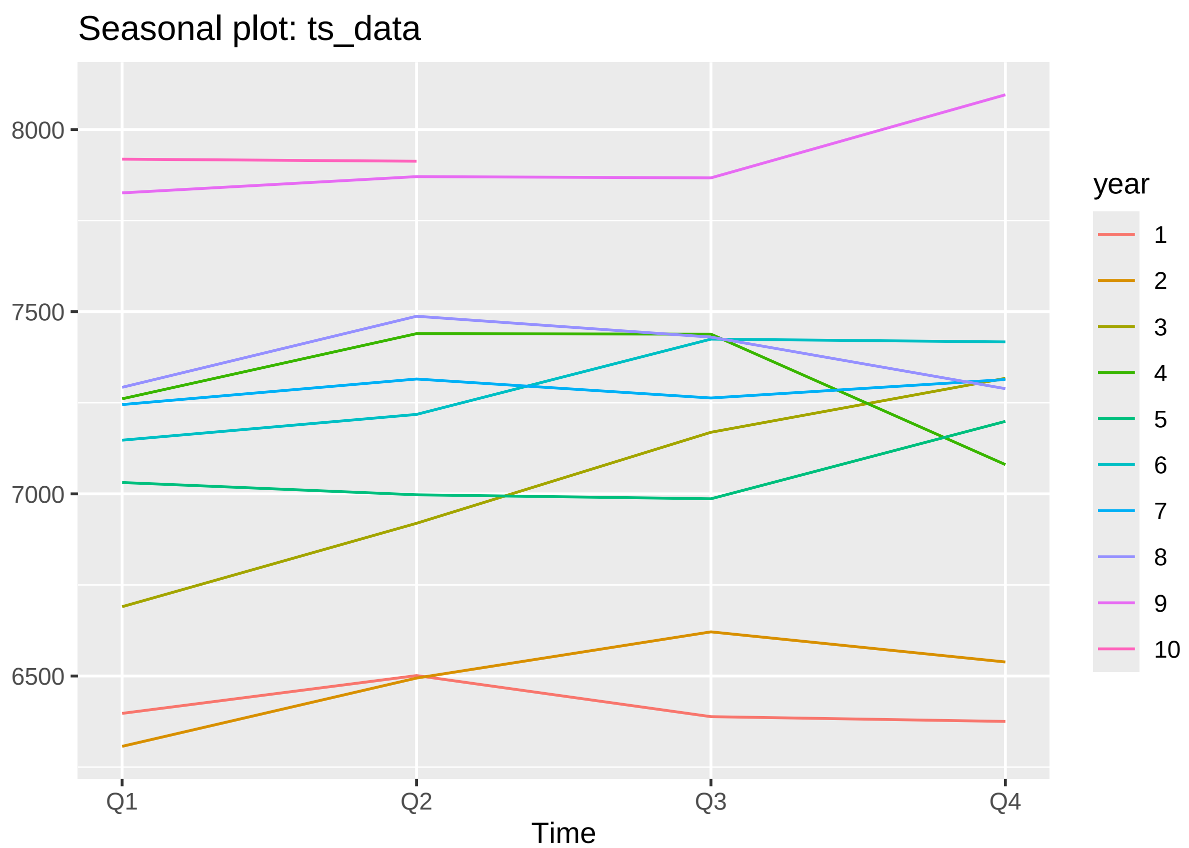
Task 1- Select dataset Q-130 and preprocess it + Plot the data

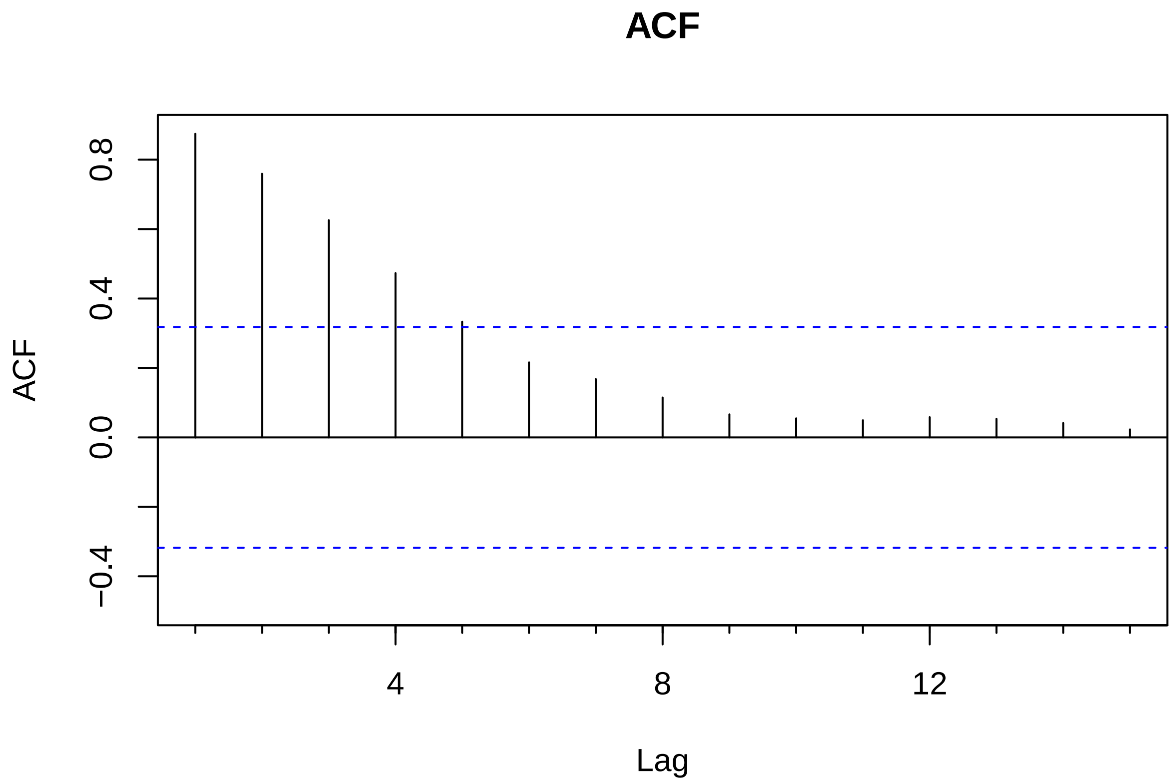
```
data_set_to_load <- c("Data_set_130")

output_Second_Read_Data <- Madpis_Second_read_data(
   Madpis_Initial_read_data_output = output_Inital_Read_Data,
   data_set_to_load = data_set_to_load,
   plot_dataset = TRUE,
   plot_decompose = TRUE, plot_seasonal = TRUE)
```



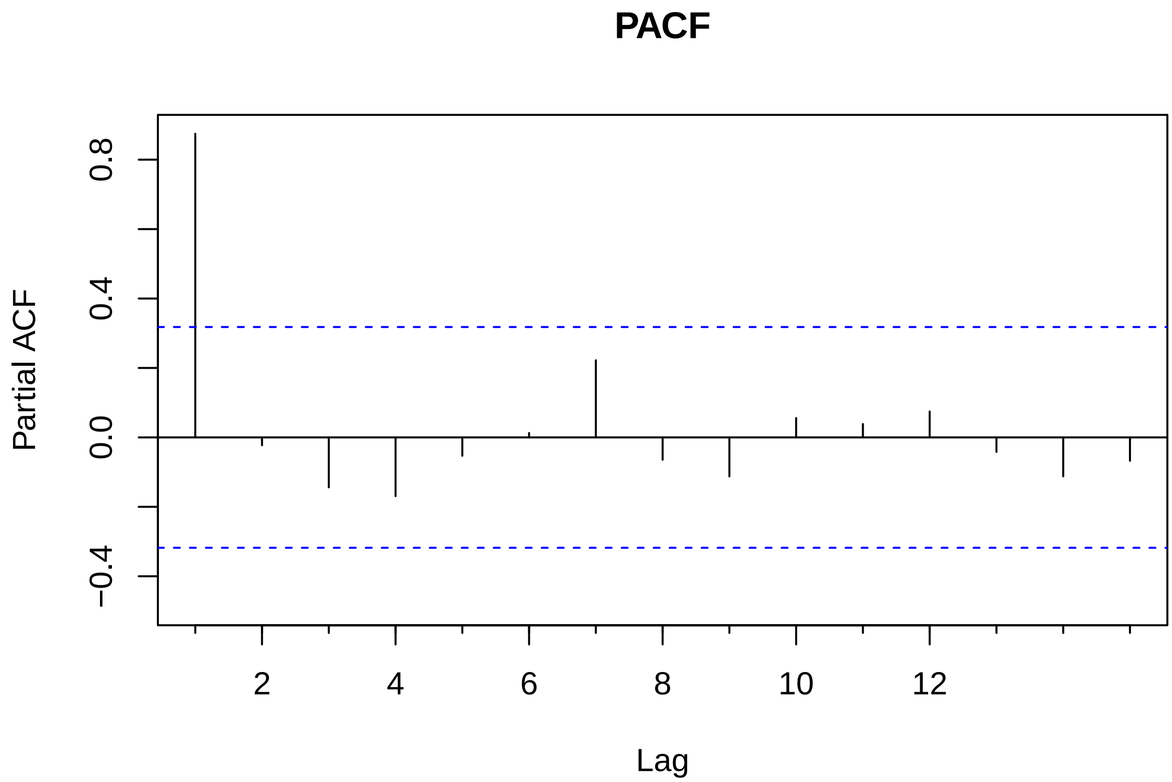

Decomposition of additive time series
observed
trend
seasonal
random
Time
6500
7500
-200
0
200
-10
2
4
6
8
10

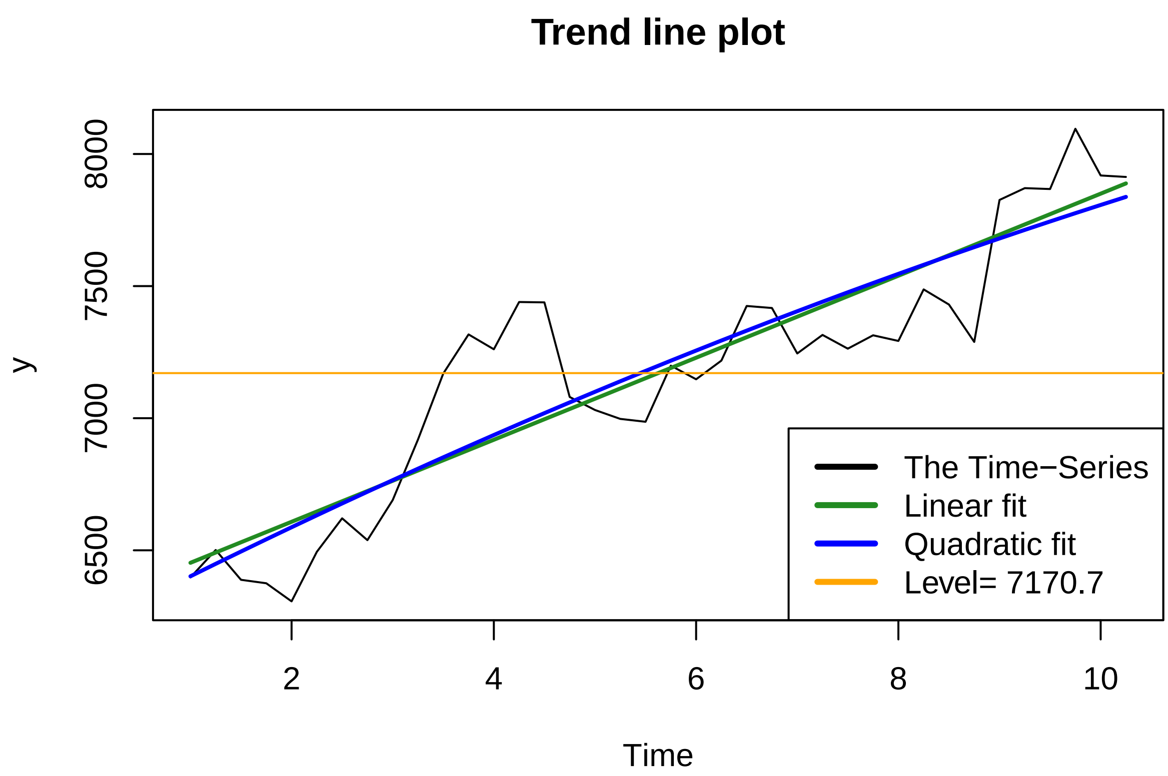

data
seasonal
trend
remainder
6500
7500
6500
7500
20
−10
−40
200
0
−200
2
4
6
8
10
time


Seasonal plot: ts_data
8000
7500
7000
6500
Q1
Q2
Q3
Q4
Time
year
1
2
3
4
5
6
7
8
9
10



ACF
0.8
0.4
0.0
−0.4
ACF
4
8
12
Lag


PACF
Partial ACF
0.8
0.4
0.0
−0.4
2
4
6
8
10
12
Lag




```
#tsibble_data <- output_Second_Read_Data$tsibble_quarterly_train
```

Notes on the time-series:

- We can see that there is **9 years of quarterly data**
- From the **Time plot** we can see that there is an overall increase trend
- From the **Decomposition graph** and the **seasonality graph** we can see there is seasonality in the series
- From the **ACF** and **Pacf** plots we can understand there **is an autocorrelation in the series**, the **Pacf** plot reveals that most of that autocorrelation originate from the 1st lag autocorrelation.

Task 2- Define the models + fit them + get the performance and plot

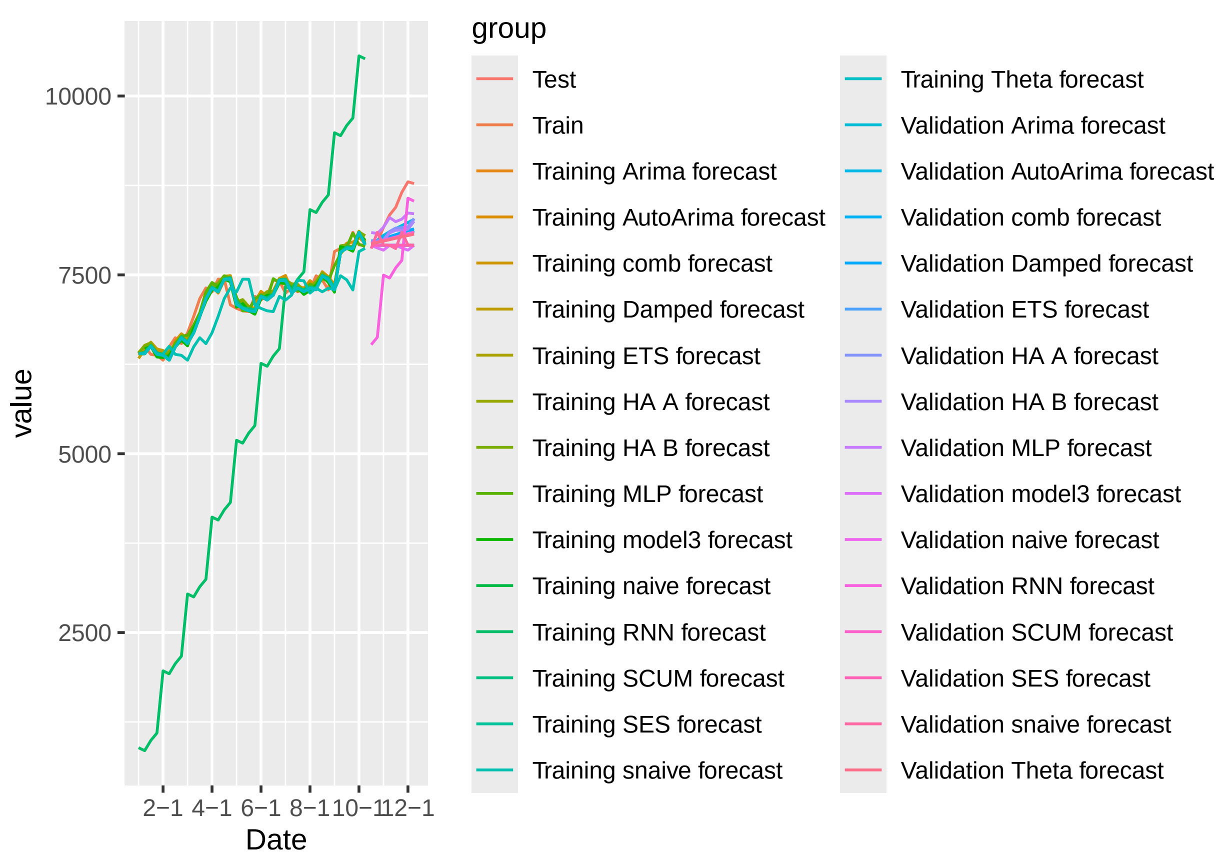
```
#Madpis_Second_read_data_output <- output_Second_Read_Data

Madpis_Third_output <- Madpis_Third_models(
   Madpis_Second_read_data_output = output_Second_Read_Data,
   include_RNN = T,
   plot_all_models = TRUE,
   print_accuracy_all_models = TRUE,
   show_top_3_models = TRUE,
```

```r
    plot_MLP = FALSE, write_datasets_for_RNN = F,
    dir_name_save_dataset = "datasets_for_RNN"
)
```

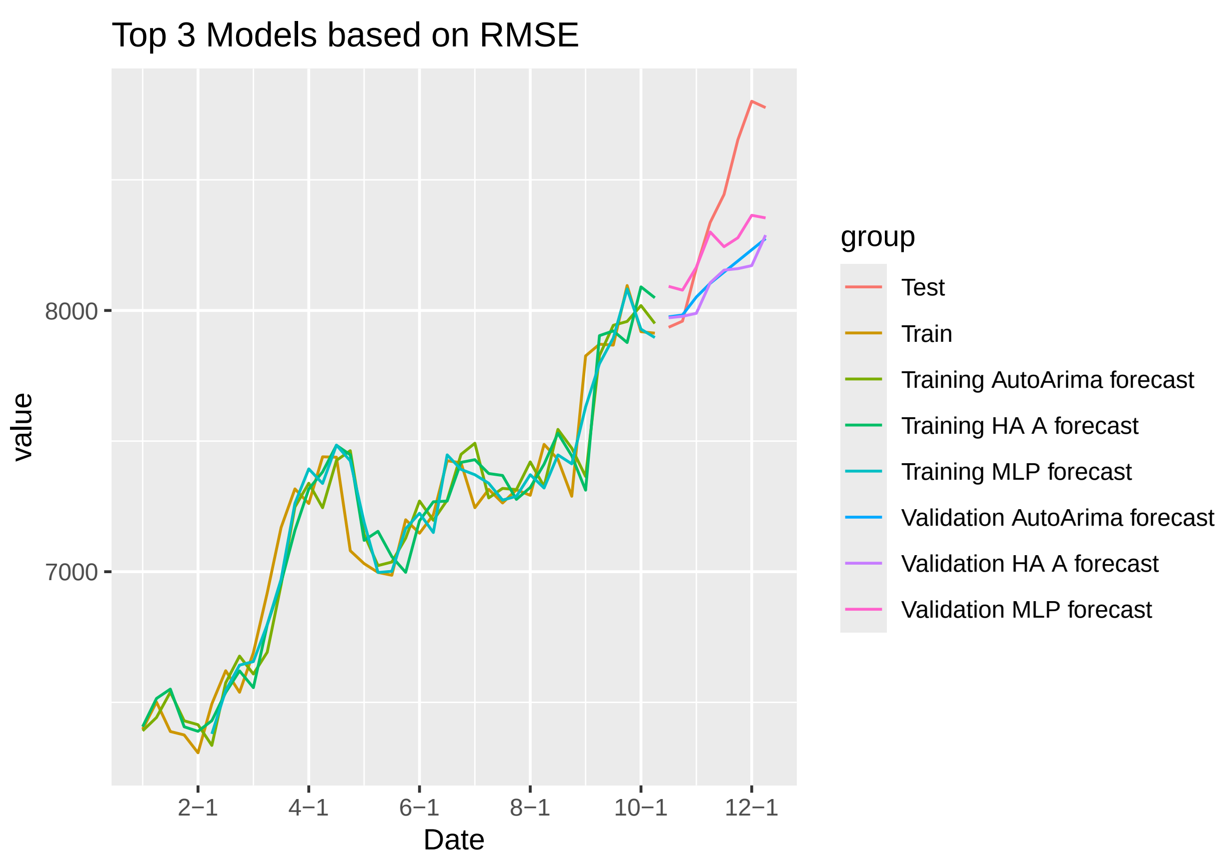
```
## New names:
## Rows: 38 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## New names:
## Rows: 8 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
## [1] "KPSS Test p-value is lower than 0.05 Thus we need to reject\nH0: The data is **NOT Stationary**
```

```
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
```



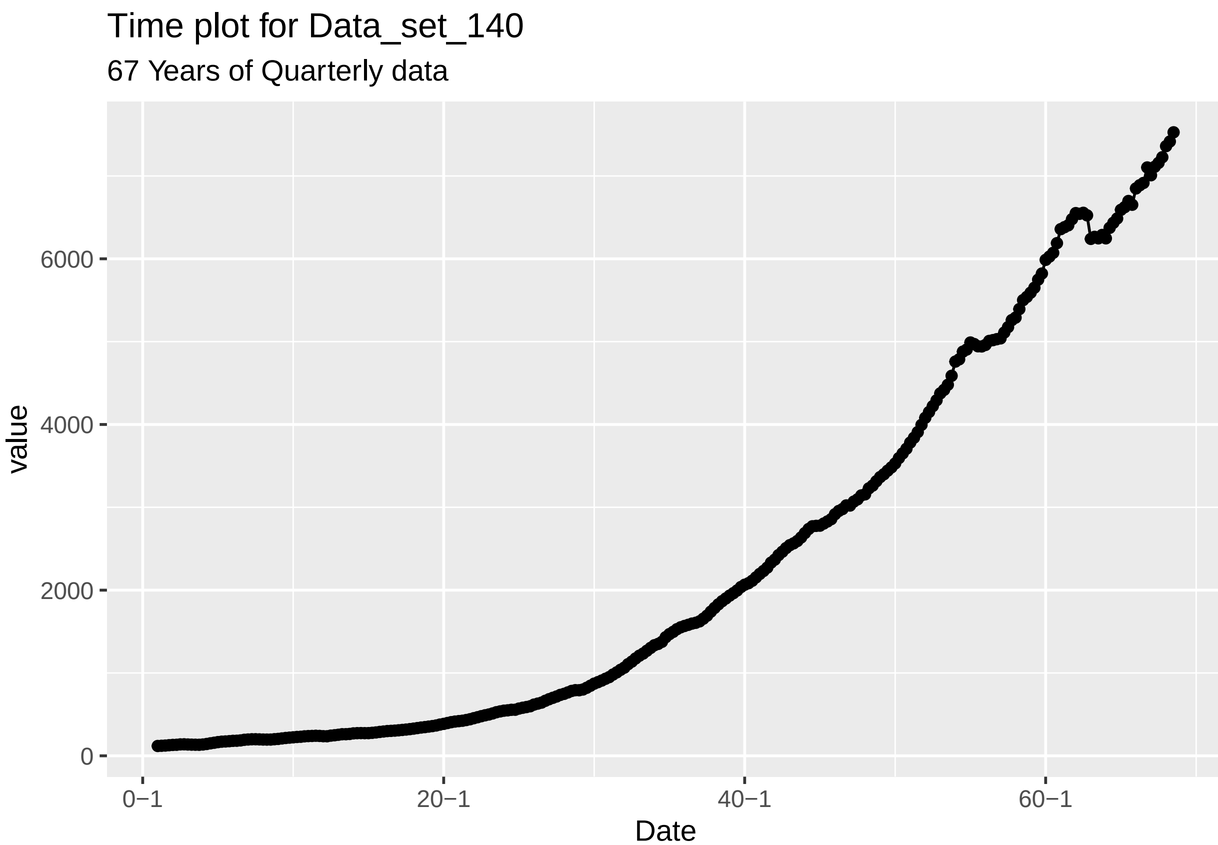
```
## # A tibble: 30 x 10
##    Model  Set            ME  RMSE   MAE    MPE  MAPE   MASE     ACF1 `Theil's U`
##    <chr>  <chr>       <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>    <dbl>       <dbl>
##  1 naive  Trainin s~  41.0   163.  121.  0.550  1.67  0.426 -0.0783         NA
##  2 naive  Test Set   470.    571.  470.  5.47   5.47  1.66   0.698          4.03
##  3 snaive Trainin s~ 180.    352.  283.  2.40   3.87  1      0.635          NA
##  4 snaive Test Set   435.    551.  469.  5.04   5.47  1.66   0.637          3.88
##  5 model3 Trainin s~  NA     158.  109. NA      1.50  0.950 NA              NA
##  6 model3 Test Set    NA     595.  497. NA      5.78 12.8   NA              NA
##  7 SES    Trainin s~  40.4   161.  118.  0.542  1.63  0.415 -0.0689         NA
##  8 SES    Test Set   470.    571.  470.  5.47   5.47  1.66   0.698          4.03
##  9 HA A   Trainin s~  -6.93  150.  114. -0.120  1.57  0.402 -0.00596        NA
## 10 HA A   Test Set   281.    362.  295.  3.26   3.43  1.04   0.677          2.55
## # i 20 more rows
## # A tibble: 3 x 10
##   Model     Set         ME  RMSE   MAE   MPE  MAPE  MASE  ACF1 `Theil's U`
##   <chr>     <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>       <dbl>
## 1 MLP       Test Set  149.  271.  219.  1.68  2.55 0.772 0.702        1.87
## 2 AutoArima Test Set  264.  344.  280.  3.05  3.25 0.988 0.706        2.42
## 3 HA A      Test Set  281.  362.  295.  3.26  3.43 1.04  0.677        2.55
```

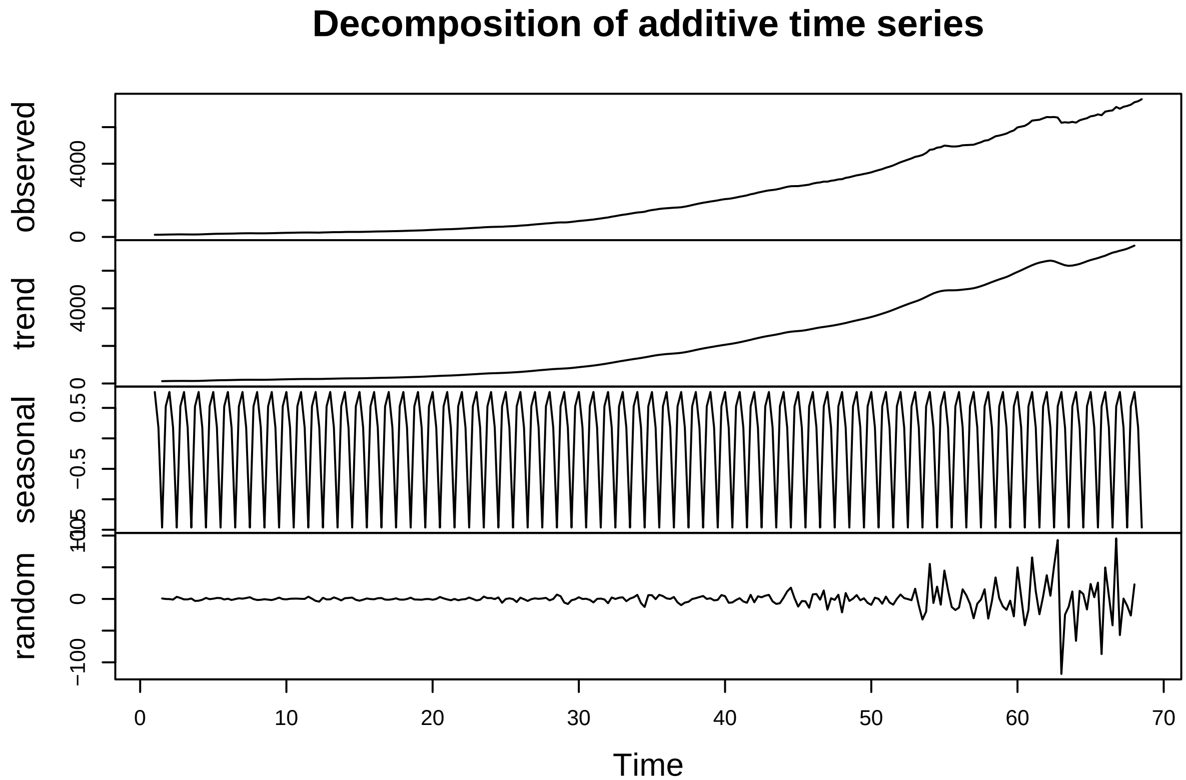


Dataset #4- Evaluate data set Q140:

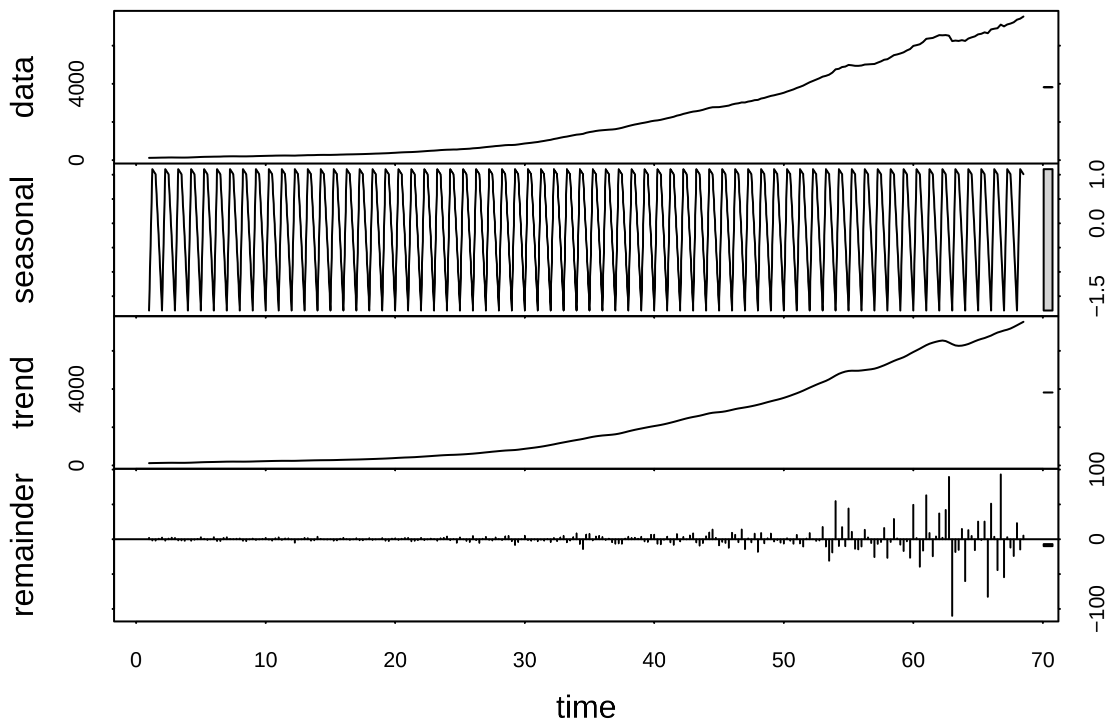
Task 1- Select dataset Q-140 and preprocess it + Plot the data

```
data_set_to_load <- c("Data_set_140")

output_Second_Read_Data <- Madpis_Second_read_data(
   Madpis_Initial_read_data_output = output_Inital_Read_Data,
   data_set_to_load = data_set_to_load,
   plot_dataset = TRUE,
   plot_decompose = TRUE, plot_seasonal = TRUE)
```

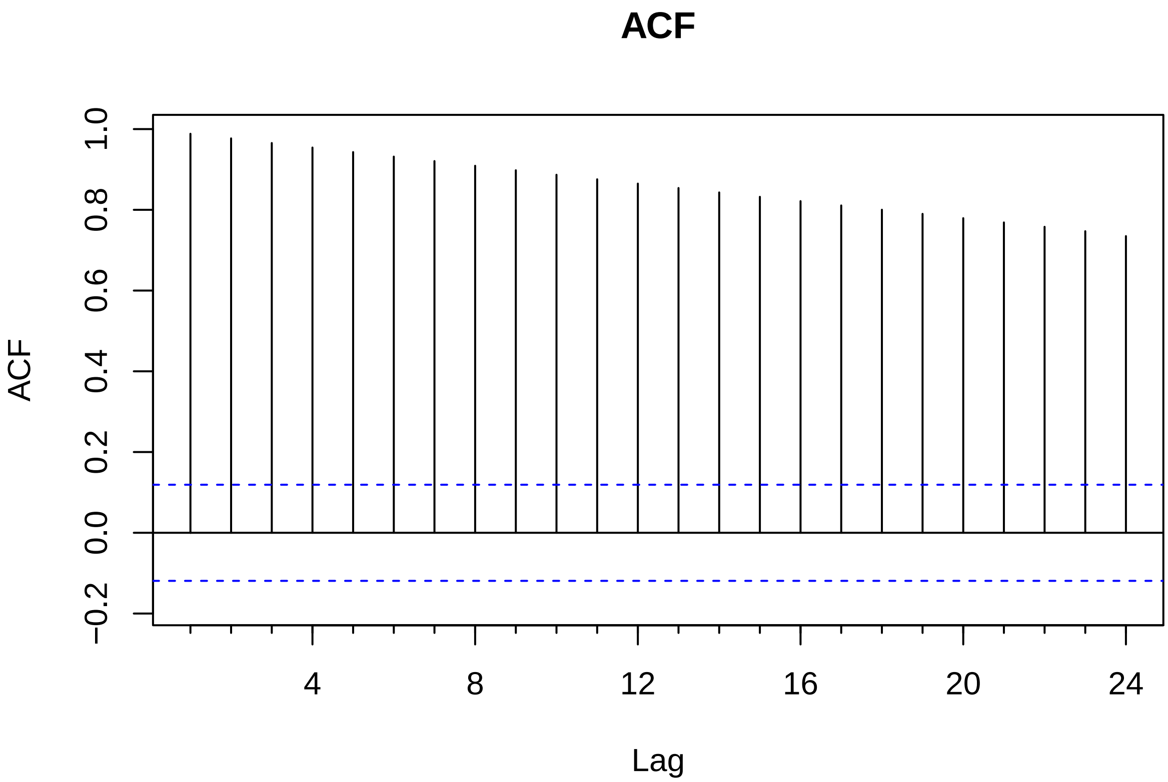


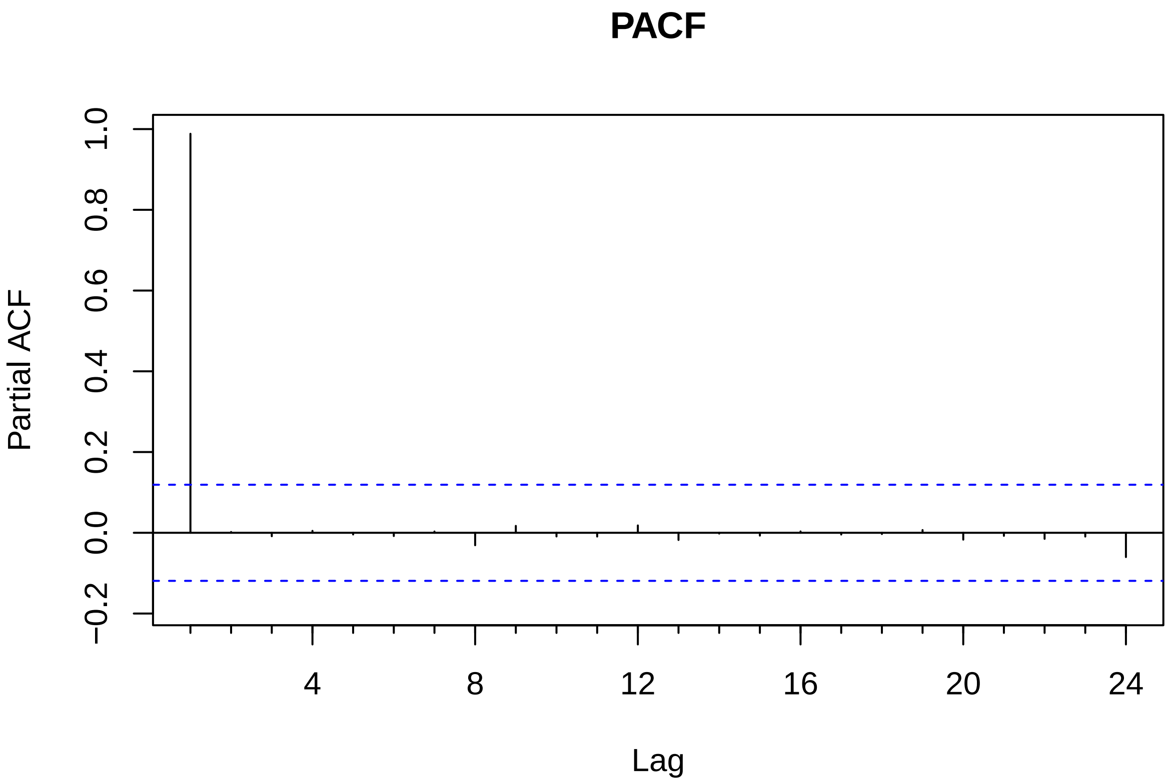


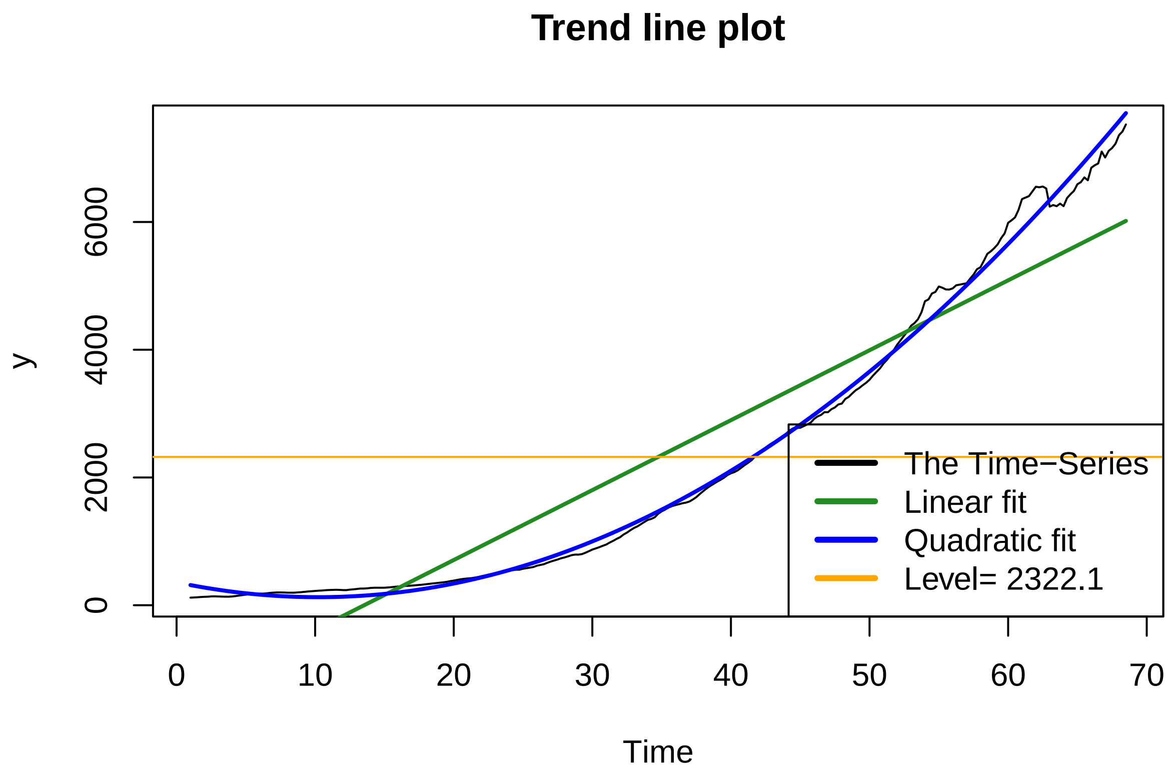
Decomposition of additive time series
observed
trend
seasonal
random
0
4000
0
4000
0.50
−0.5
−1.5
100
0
−100
0
10
20
30
40
50
60
70
Time


data
seasonal
trend
remainder
0
4000
1.0
0.0
−1.5
100
−100
0
10
20
30
40
50
60
70
time

Seasonal plot: ts_data
year
6000
4000
2000
0
Q1
Q2
Q3
Q4
Time
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68


ACF
ACF
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
4
8
12
16
20
24
Lag


PACF
Partial ACF
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
4
8
12
16
20
24
Lag




```r
#tsibble_data <- output_Second_Read_Data$tsibble_quarterly_train
```

Notes on the time-series:

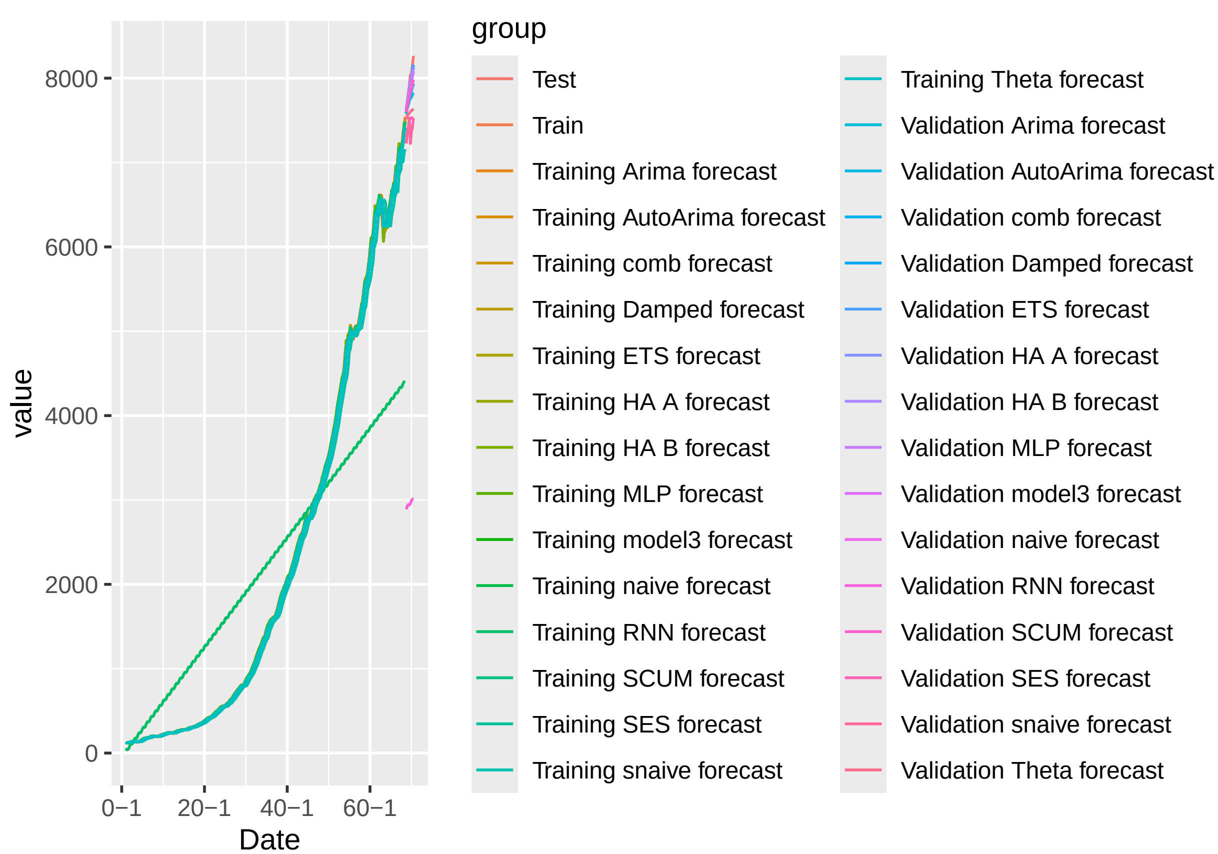
- We can see that there is **67 years of quarterly data**
- From the **Time plot** we can see that there is an overall increase trend
- From the **Decomposition graph** and the **seasonality graph** we can see there is seasonality in the series
- From the **ACF** and **Pacf** plots we can understand there **is an autocorrelation in the series**, the **Pacf** plot reveals that most of that autocorrelation originate from the 1st lag autocorrelation.

Task 2- Define the models + fit them + get the performance and plot

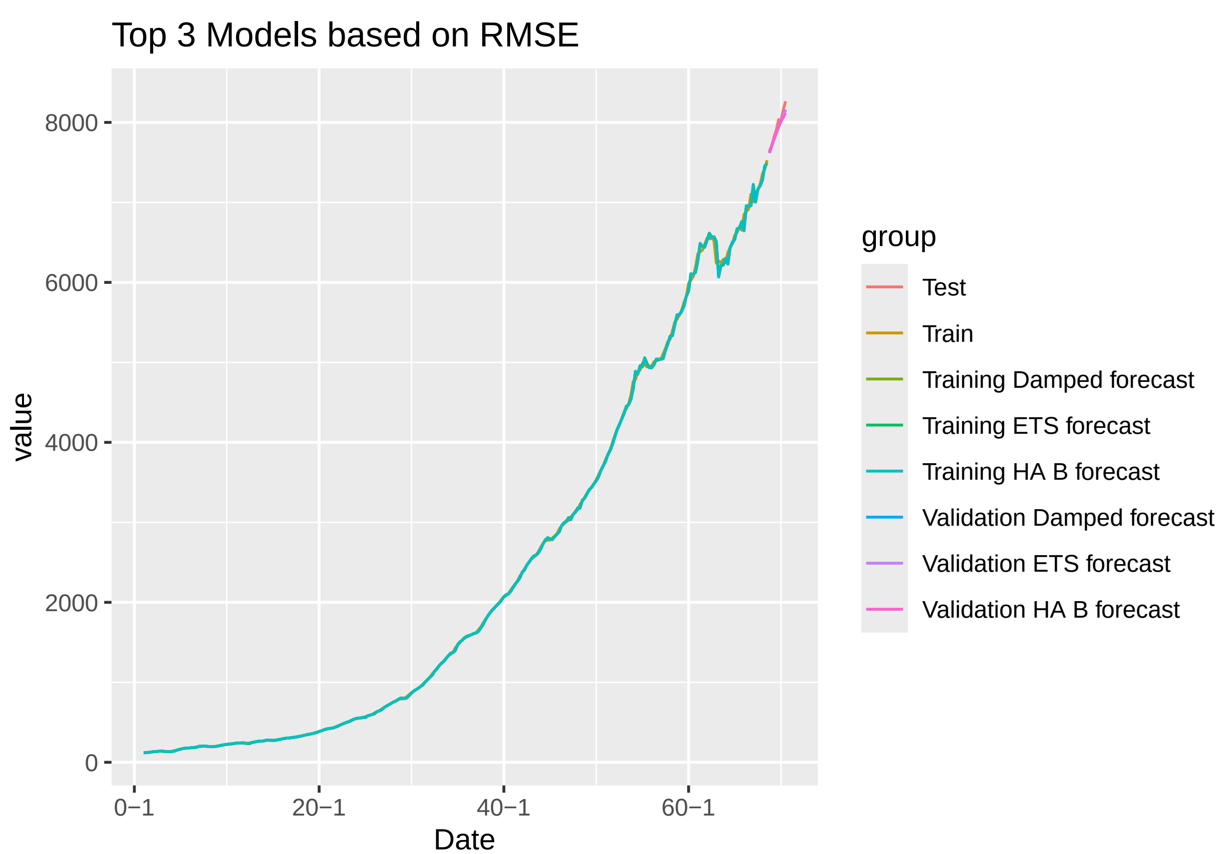
```r
#Madpis_Second_read_data_output <- output_Second_Read_Data

Madpis_Third_output <- Madpis_Third_models(
   Madpis_Second_read_data_output = output_Second_Read_Data,
   include_RNN = T,
   plot_all_models = TRUE,
   print_accuracy_all_models = TRUE,
   show_top_3_models = TRUE,
```

```
    plot_MLP = FALSE, write_datasets_for_RNN = F,
    dir_name_save_dataset = "datasets_for_RNN"
)
```

```
## New names:
## Rows: 271 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## New names:
## Rows: 8 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`

## [1] "KPSS Test p-value is lower than 0.05 Thus we need to reject\nH0: The data is **NOT Stationary**

## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
```



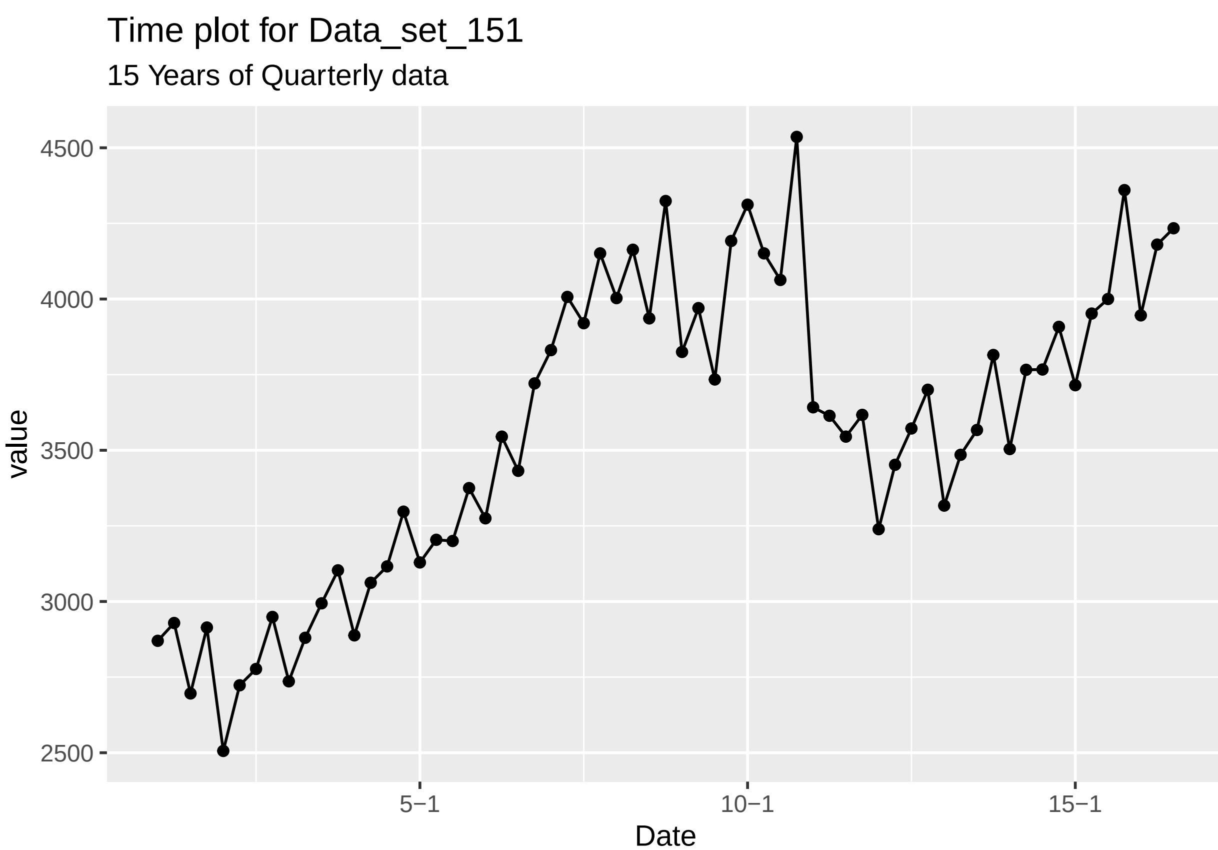
```
## # A tibble: 30 x 10
##    Model  Set            ME  RMSE   MAE   MPE  MAPE   MASE   ACF1 `Theil's U`
##    <chr>  <chr>       <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>       <dbl>
##  1 naive  Trainin set  27.4  49.1  31.8 1.52  1.65   0.270  0.281      NA
##  2 naive  Test Set    417.  463.  417.  5.18  5.18   3.54   0.597       4.83
##  3 snaive Trainin set 109.  159.  118.  5.89  6.14   1      0.901      NA
##  4 snaive Test Set    561.  590.  561.  7.02  7.02   4.77   0.475       6.01
##  5 model3 Trainin set  NA    49.1  31.9 NA    1.68   1.01  NA          NA
##  6 model3 Test Set     NA   462.  415.  NA    5.17 421.    NA          NA
##  7 SES    Trainin set  27.3  49.1  31.7 1.48  1.68   0.270  0.282      NA
##  8 SES    Test Set    417.  463.  417.  5.18  5.18   3.54   0.597       4.83
##  9 HA A   Trainin set   2.83  39.4  18.3 0.157 0.732  0.156 -0.409      NA
## 10 HA A   Test Set     60.1  75.8  60.1 0.744 0.744  0.511  0.201       0.785
## # i 20 more rows
## # A tibble: 3 x 10
##   Model  Set         ME  RMSE   MAE   MPE  MAPE  MASE    ACF1 `Theil's U`
##   <chr>  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>       <dbl>
## 1 Damped Test Set  45.0  57.9  45.4 0.558 0.563 0.386 -0.0240       0.598
## 2 ETS    Test Set  45.0  57.9  45.4 0.558 0.563 0.386 -0.0240       0.598
## 3 HA B   Test Set  51.4  71.7  54.5 0.633 0.674 0.463  0.239        0.744
```

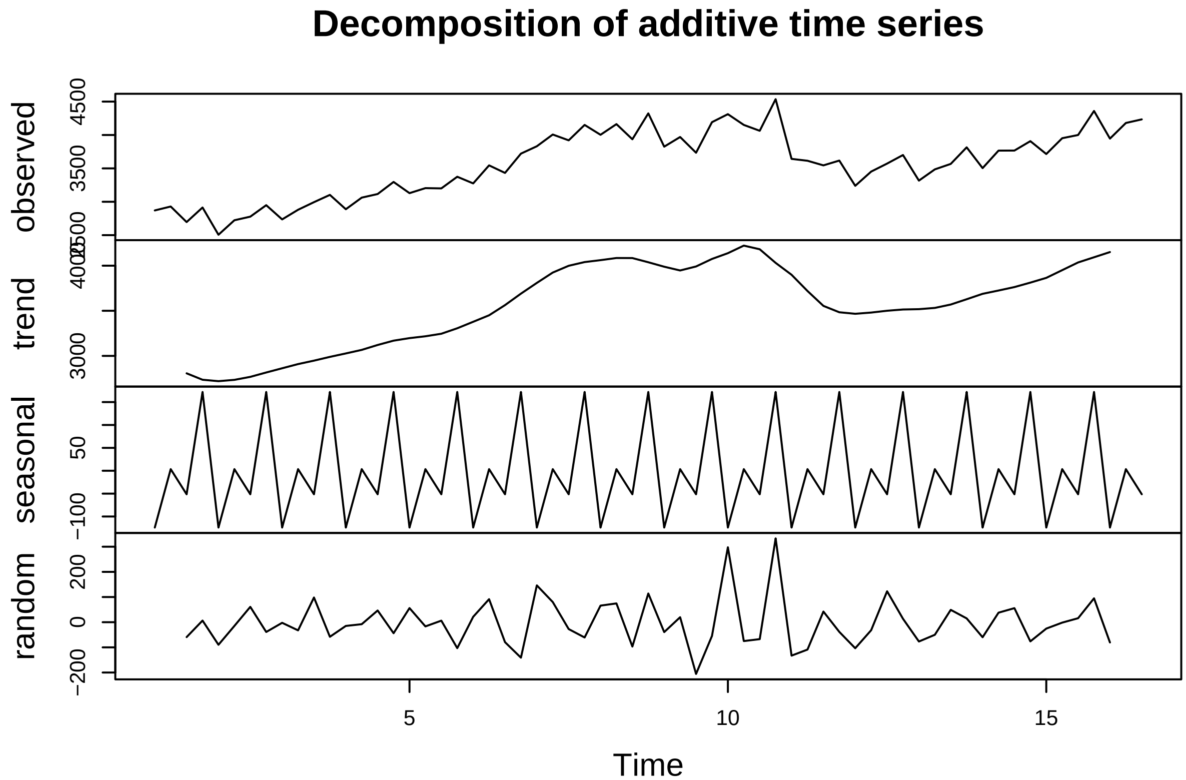


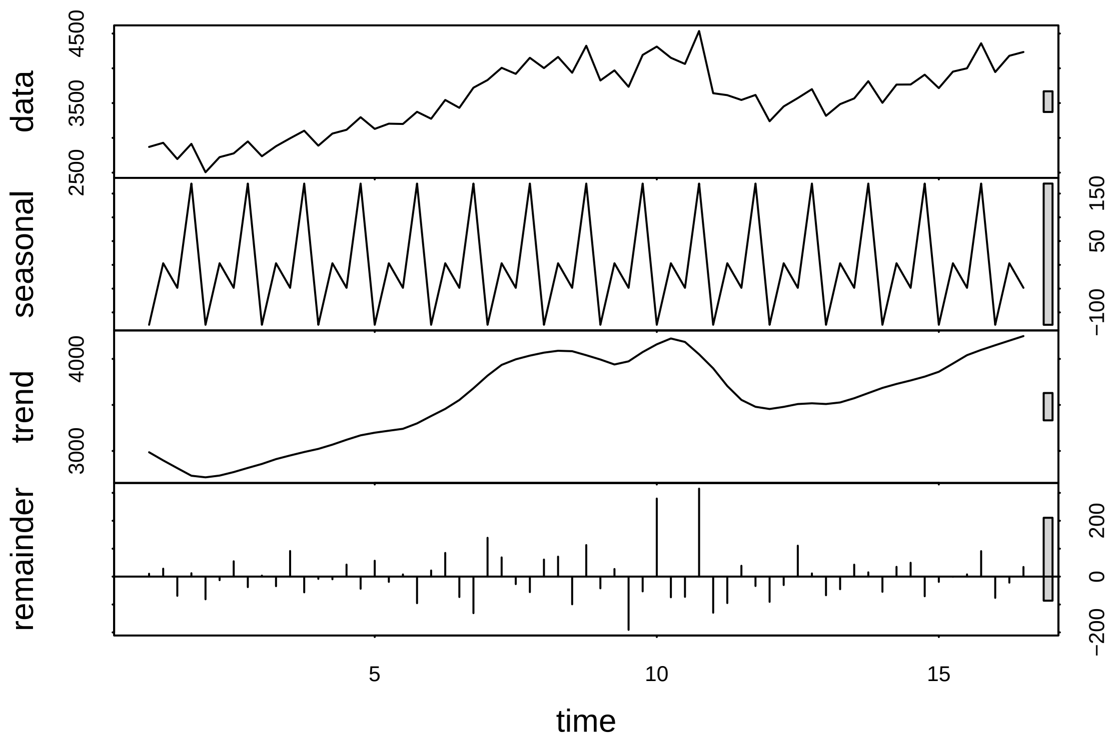
Dataset #5- Evaluate data set Q151:

Task 1- Select dataset Q-151 and preprocess it + Plot the data

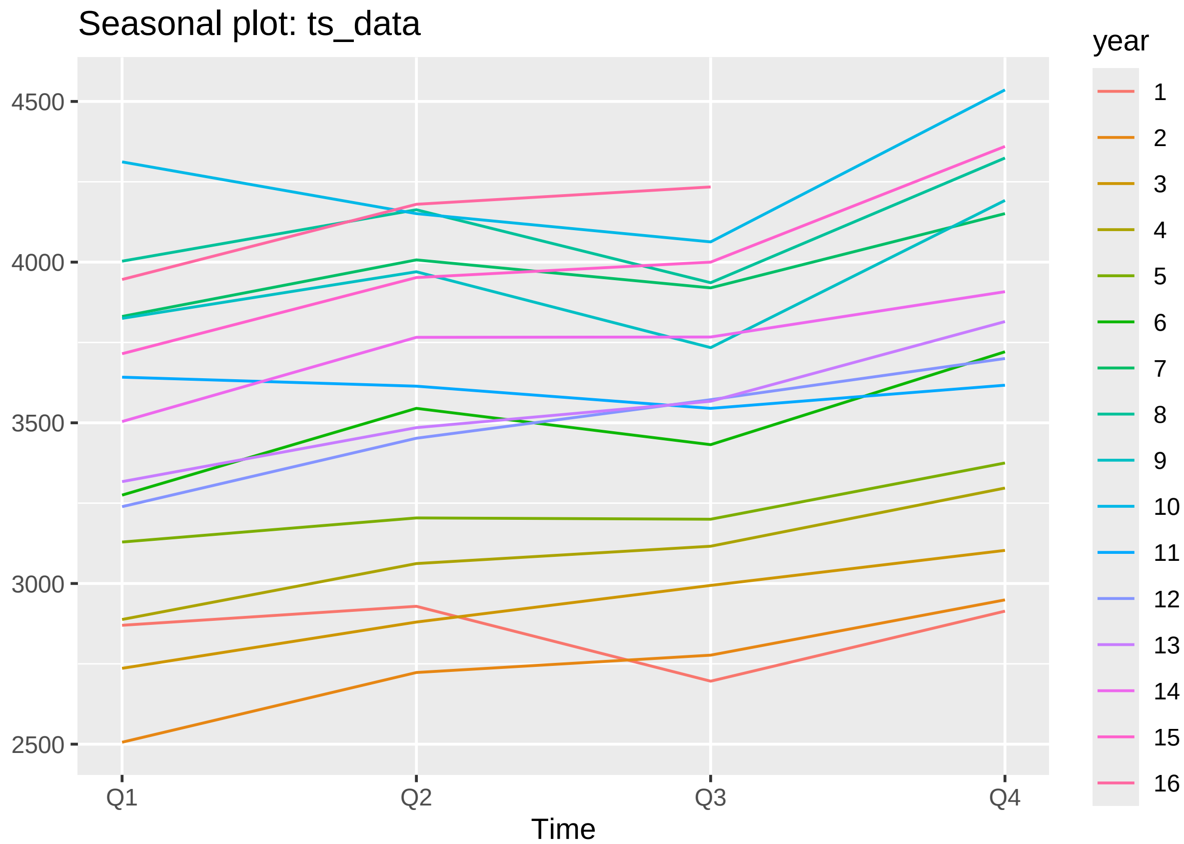
```
data_set_to_load <- c("Data_set_151")

output_Second_Read_Data <- Madpis_Second_read_data(
   Madpis_Initial_read_data_output = output_Inital_Read_Data,
   data_set_to_load = data_set_to_load,
   plot_dataset = TRUE,
   plot_decompose = TRUE, plot_seasonal = TRUE)
```

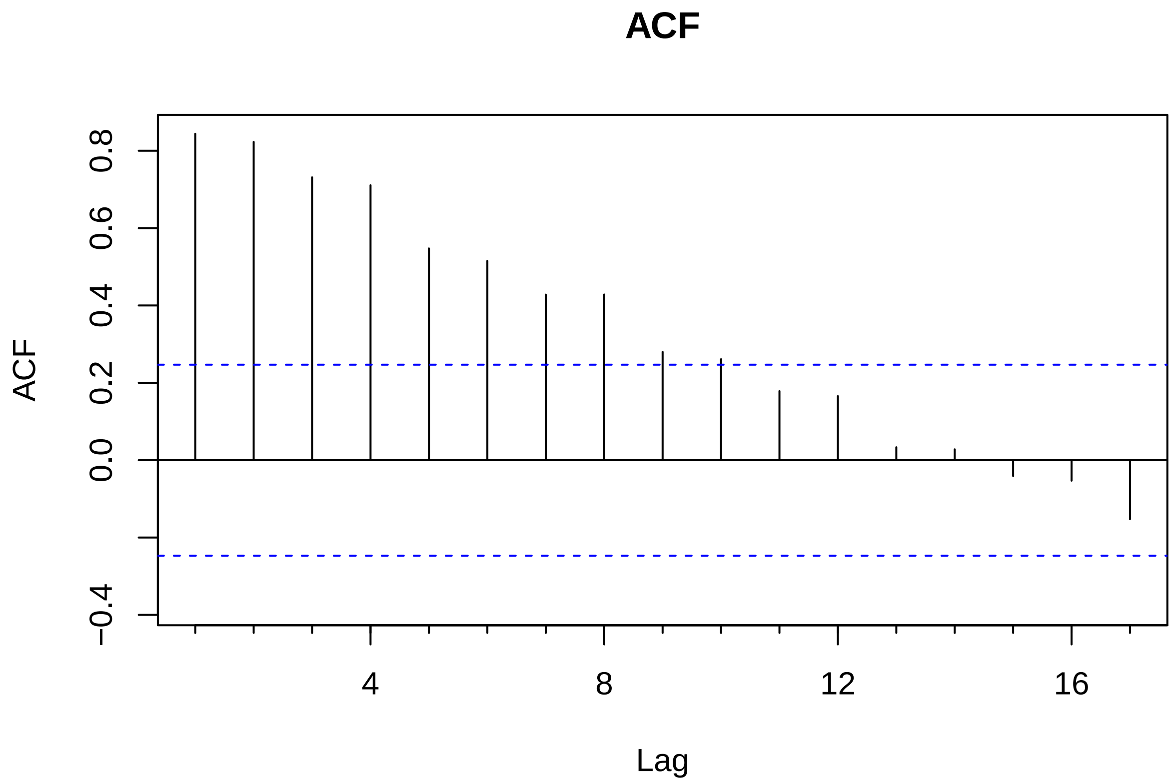


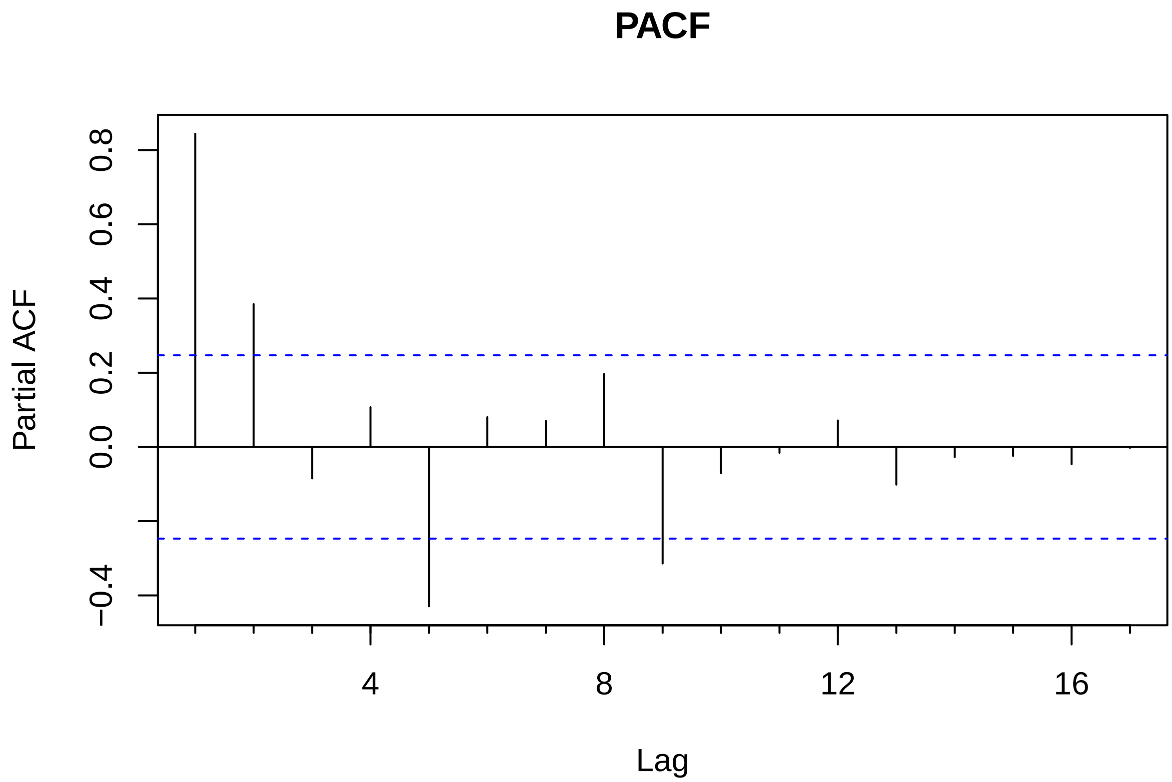

Decomposition of additive time series
observed
trend
seasonal
random
4500
3500
2500
4000
3000
50
−100
200
0
−200
5
10
15
Time



data
seasonal
trend
remainder
4500
3500
2500
150
50
−100
4000
3000
200
0
−200
5
10
15
time


Seasonal plot: ts_data
4500
4000
3500
3000
2500
Q1
Q2
Q3
Q4
Time
year
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


ACF
ACF
0.8
0.6
0.4
0.2
0.0
−0.4
4
8
12
16
Lag


PACF
Partial ACF
0.8
0.6
0.4
0.2
0.0
−0.4
4
8
12
16
Lag


```r
#tsibble_data <- output_Second_Read_Data$tsibble_quarterly_train
```

Notes on the time-series:

- We can see that there is **15 years of quarterly data**
- From the **Time plot** we can see that for the first 10 years there is an increase trend, afterwords - a sharp decrease trend and then again an increase trend.
- From the **Decomposition graph** and the **seasonality graph** we can see there is seasonality in the series
- From the **ACF** and **Pacf** plots we can understand there **is an autocorrelation in the series**, the **Pacf** plot reveals that most of that autocorrelation originate from the 1st lag autocorrelation and the 2nd and 5th

Task 2- Define the models + fit them + get the performance and plot

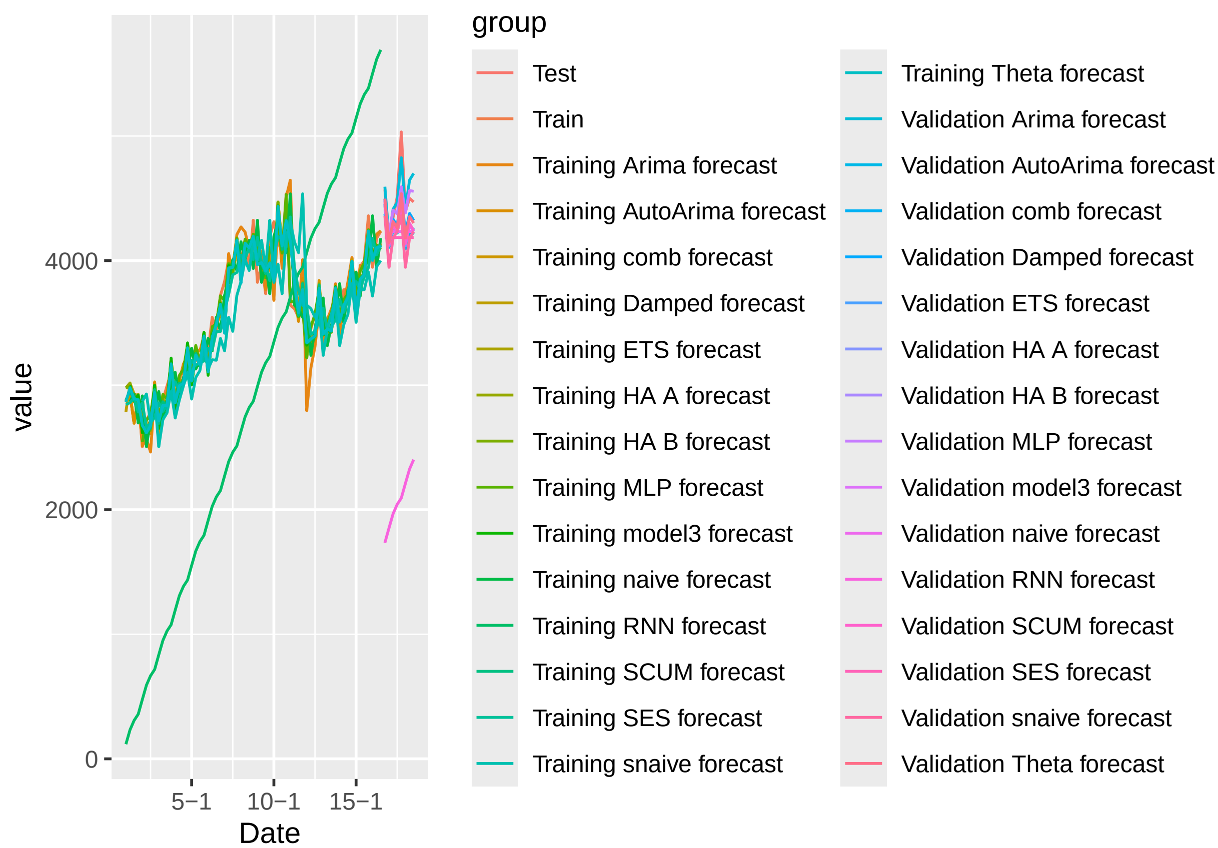
```r
#Madpis_Second_read_data_output <- output_Second_Read_Data

Madpis_Third_output <- Madpis_Third_models(
   Madpis_Second_read_data_output = output_Second_Read_Data,
   include_RNN = T,
   plot_all_models = TRUE,
```

```
  print_accuracy_all_models = TRUE,
  show_top_3_models = TRUE,
  plot_MLP = FALSE, write_datasets_for_RNN = F,
  dir_name_save_dataset = "datasets_for_RNN"
)
```

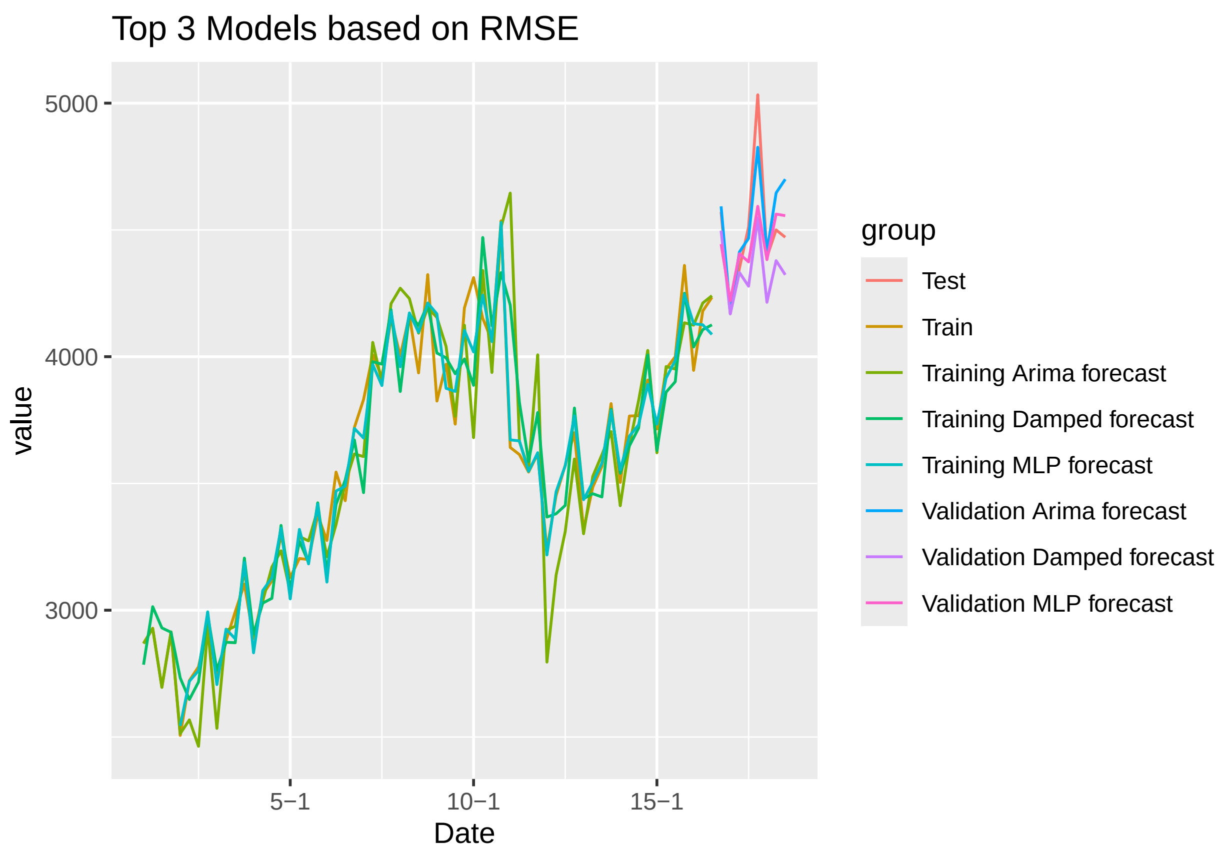
```
## New names:
## Rows: 63 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## New names:
## Rows: 8 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘
```

```
## [1] "KPSS Test p-value is lower than 0.05 Thus we need to reject\nH0: The data is **NOT Stationary**
```

```
## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
```



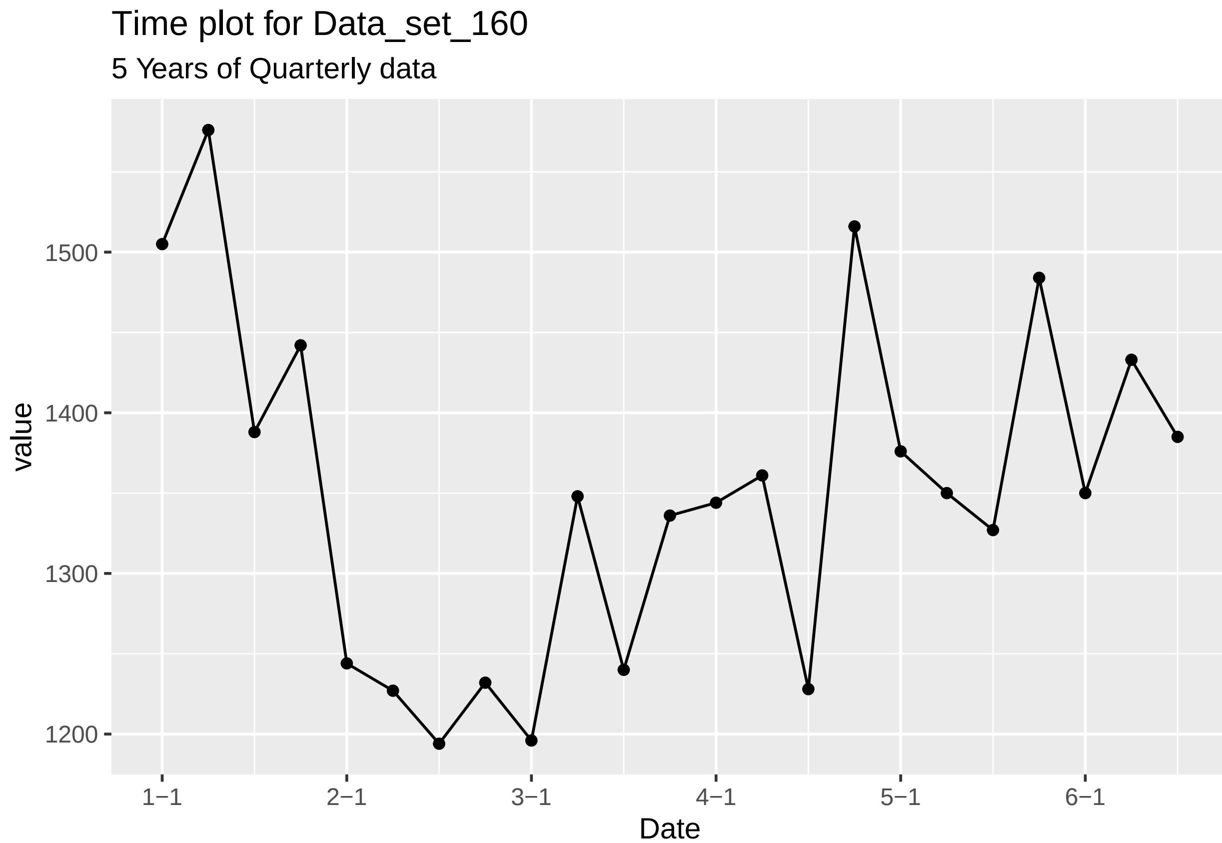

```
## # A tibble: 30 x 10
##    Model  Set            ME  RMSE   MAE    MPE  MAPE  MASE    ACF1 `Theil's U`
##    <chr>  <chr>       <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>       <dbl>
##  1 naive  Trainin set  22    250.  202.  0.390  5.68 0.833 -0.529        NA
##  2 naive  Test Set    266.   353.  282.  5.67   6.05 1.16  -0.0596        1.03
##  3 snaive Trainin set  90.0  298.  242.  2.26   6.69 1      0.683        NA
##  4 snaive Test Set    320.   356.  320.  7.00   7.00 1.32   0.316         1.06
##  5 model3 Trainin set  NA    157.  119. NA      3.30 0.587 NA            NA
##  6 model3 Test Set     NA    269.  214. NA      4.62 1.27  NA            NA
##  7 SES    Trainin set  37.7  219.  177.  0.826  4.94 0.730 -0.143        NA
##  8 SES    Test Set    314.   391.  318.  6.75   6.84 1.31  -0.0596        1.14
##  9 HA A   Trainin set  20.0  156.  116.  0.458  3.23 0.480 -0.0525       NA
## 10 HA A   Test Set    232.   282.  232.  5.01   5.01 0.957  0.346         0.857
## # i 20 more rows
## # A tibble: 3 x 10
##   Model  Set         ME  RMSE   MAE    MPE  MAPE  MASE  ACF1 `Theil's U`
##   <chr>  <chr>    <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>       <dbl>
## 1 Arima  Test Set -29.6  124.  92.6 -0.732  2.01 0.382 0.333       0.388
## 2 MLP    Test Set  57.5  175. 121.   1.12   2.58 0.501 0.142       0.527
## 3 Damped Test Set 157.   213. 157.   3.37   3.37 0.650 0.335       0.655
```



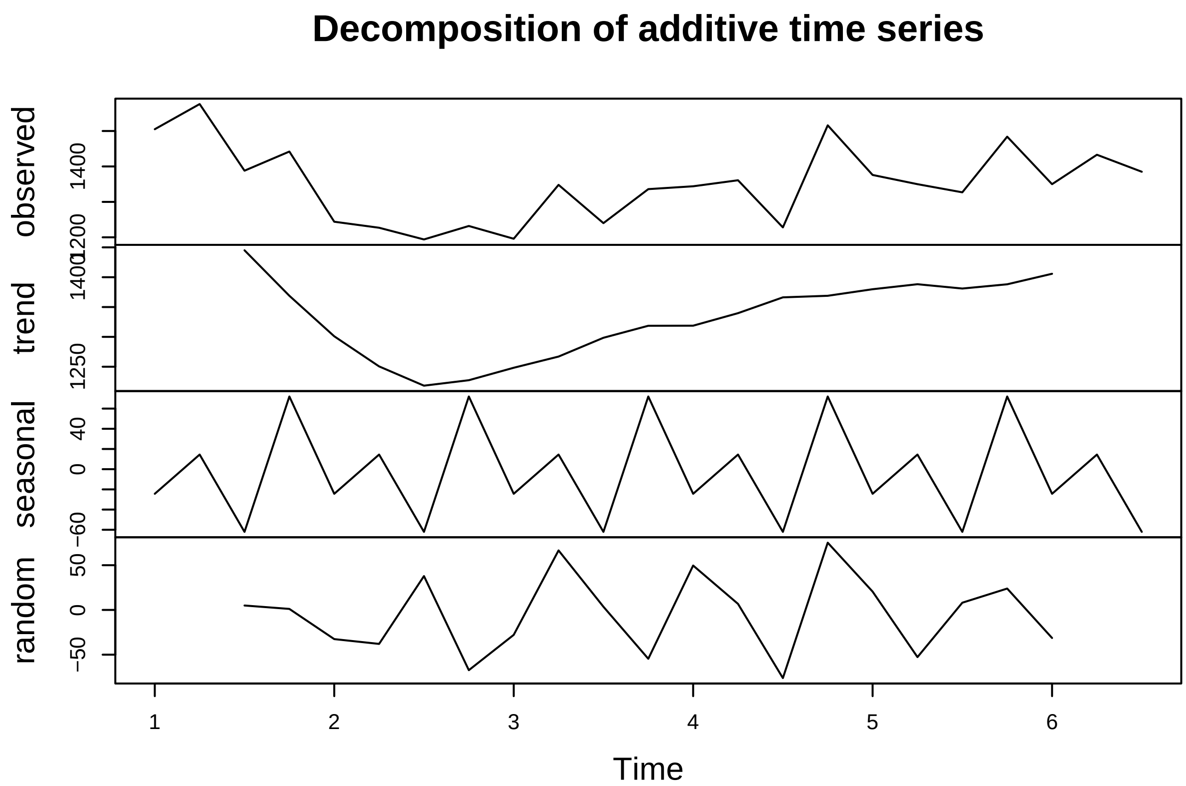
Dataset #6- Evaluate data set Q160:

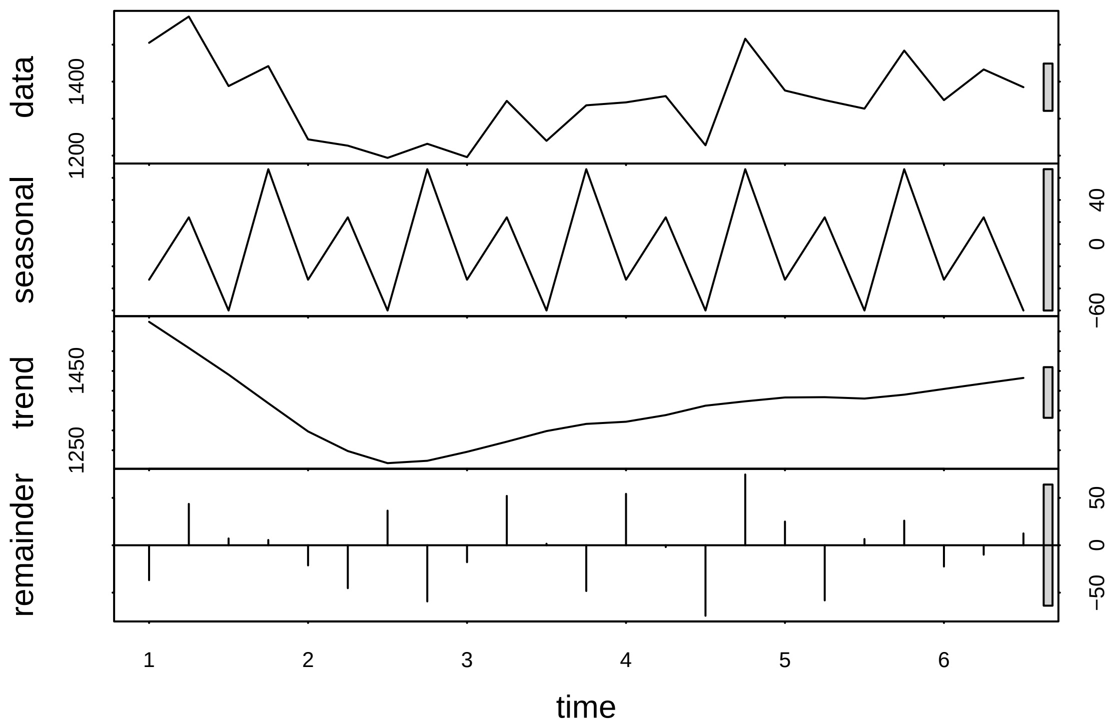
Task 1- Select dataset Q-160 and preprocess it + Plot the data

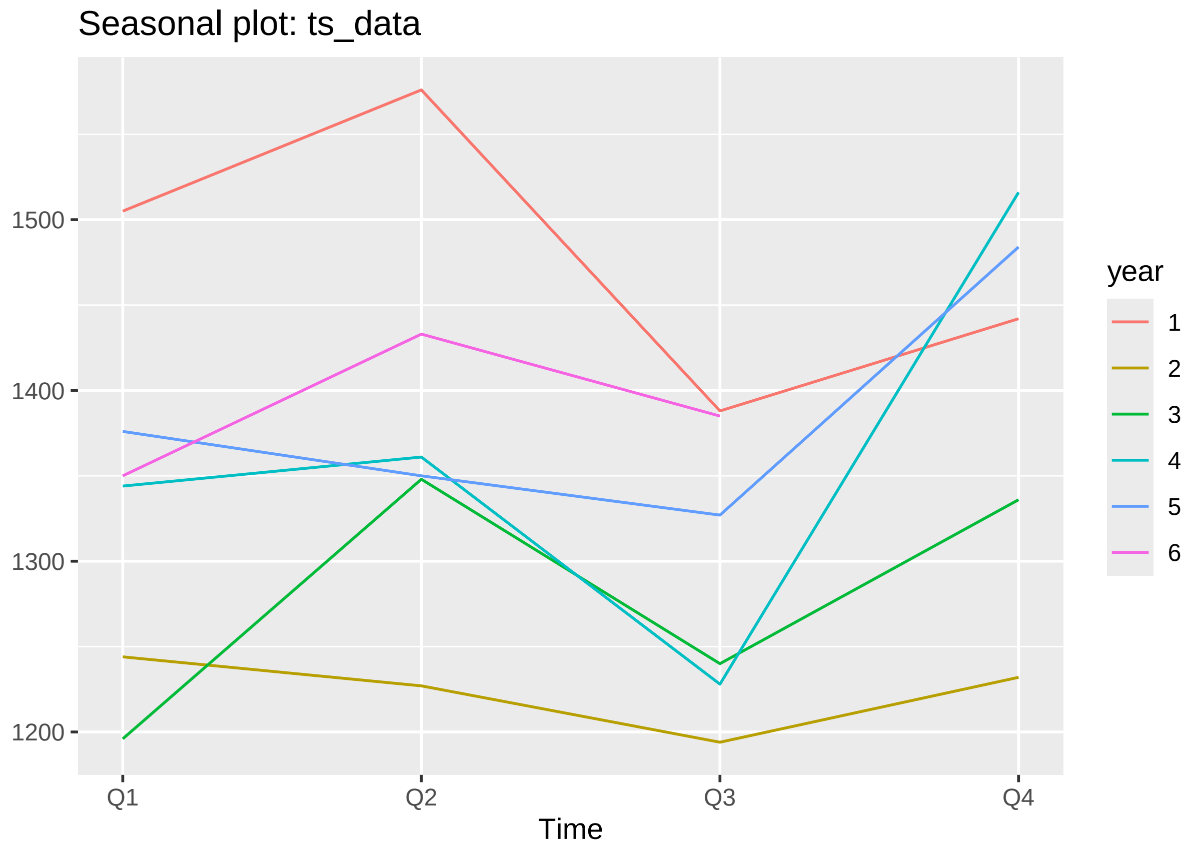
```
data_set_to_load <- c("Data_set_160")

output_Second_Read_Data <- Madpis_Second_read_data(
   Madpis_Initial_read_data_output = output_Inital_Read_Data,
   data_set_to_load = data_set_to_load,
   plot_dataset = TRUE,
   plot_decompose = TRUE, plot_seasonal = TRUE)
```

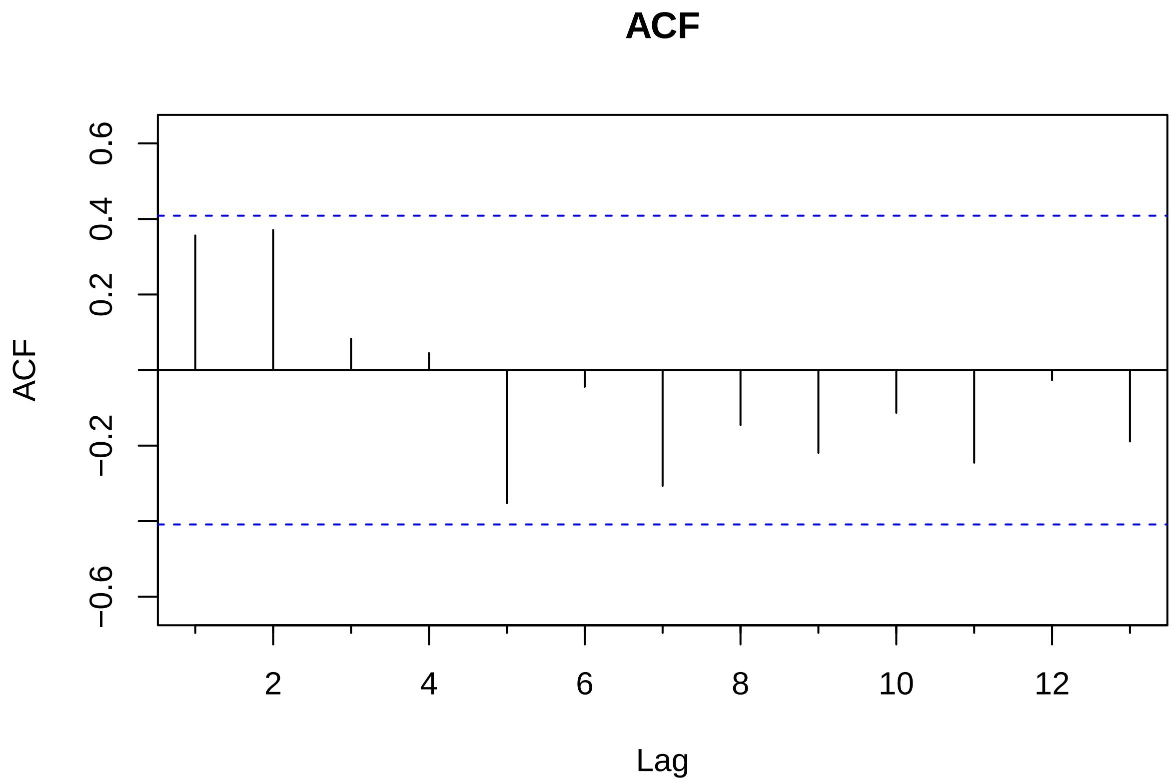


Time plot for Data_set_160
5 Years of Quarterly data
value
1500
1400
1300
1200
1−1
2−1
3−1
4−1
5−1
6−1
Date

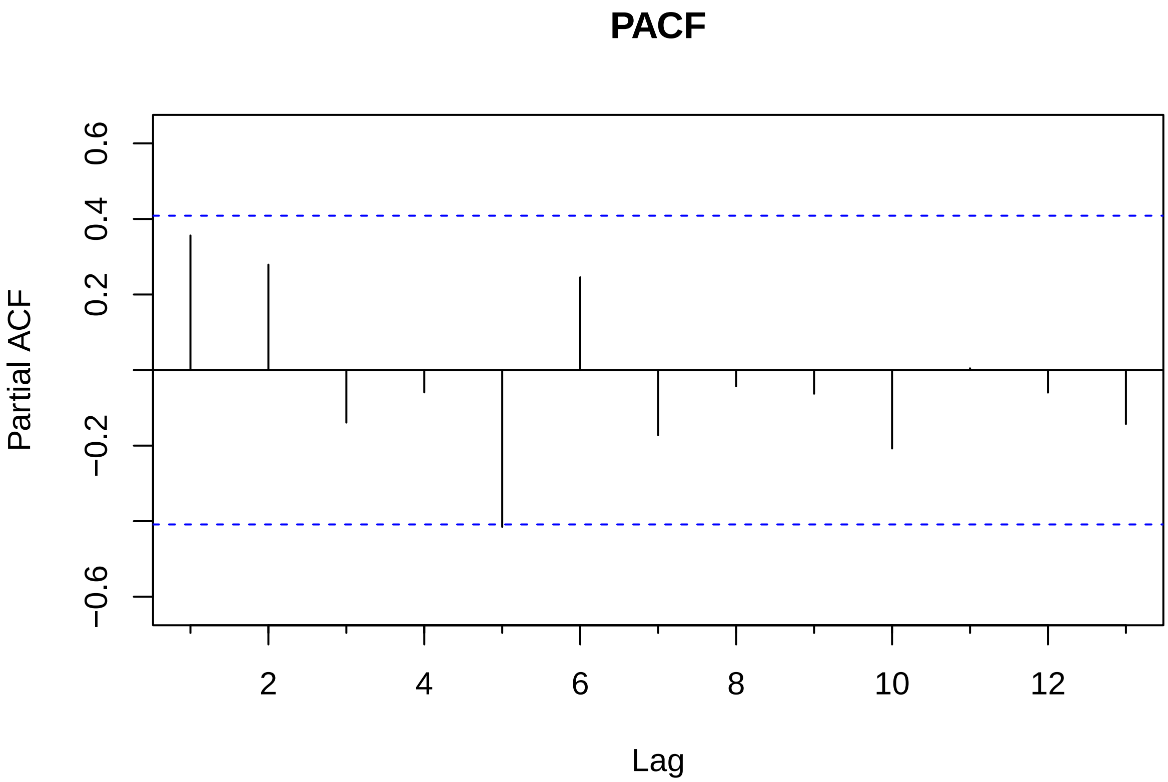

Decomposition of additive time series
observed
trend
seasonal
random
1200
1400
1250
1400
-60
0
40
-50
0
50
1
2
3
4
5
6
Time

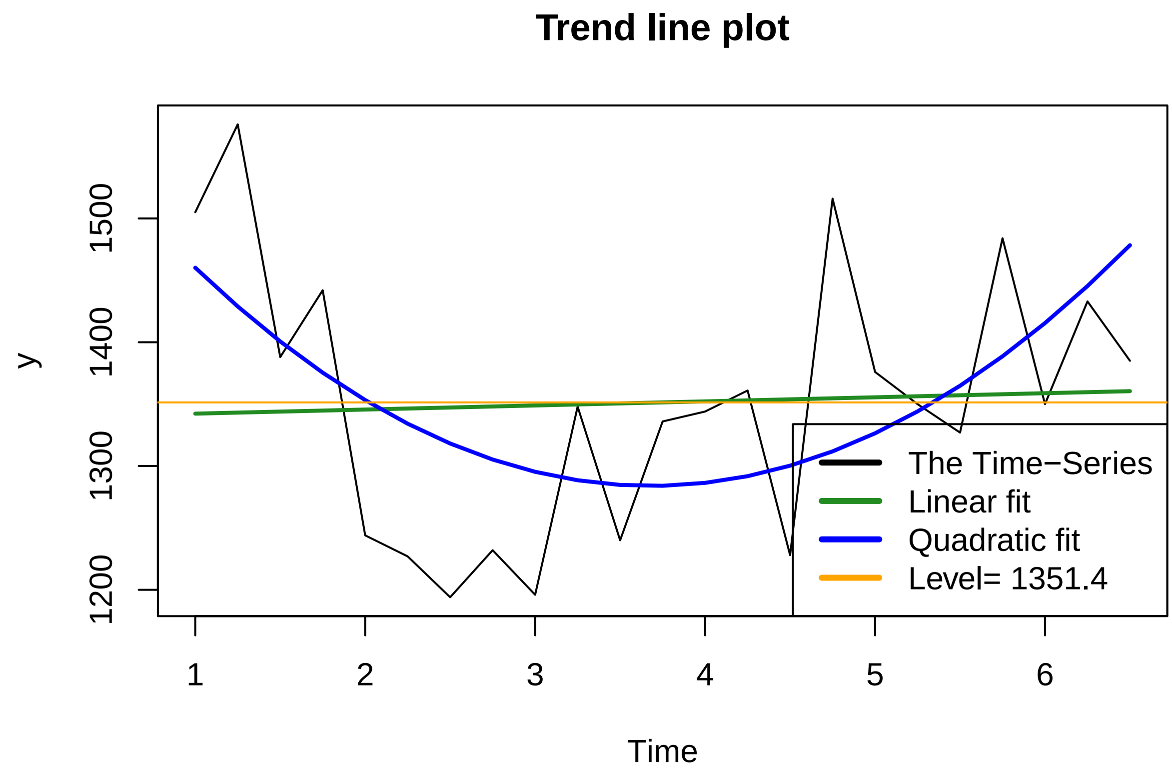

data
seasonal
trend
remainder
1200
1400
40
0
−60
1250
1450
50
0
−50
1
2
3
4
5
6
time


Seasonal plot: ts_data
1500
1400
1300
1200
Q1
Q2
Q3
Q4
Time
year
1
2
3
4
5
6



ACF
0.6
0.4
0.2
−0.2
−0.6
ACF
2
4
6
8
10
12
Lag


PACF
Partial ACF
0.6
0.4
0.2
−0.2
−0.6
2
4
6
8
10
12
Lag




```
#tsibble_data <- output_Second_Read_Data$tsibble_quarterly_train
```

Notes on the time-series:

- We can see that there is **5 years of quarterly data**
- From the **Time plot** we can see an overall decrease trend.
- From the **Decomposition graph** and the **seasonality graph** we can see there is seasonality in the series
- From the **ACF** and **Pacf** plots we can understand there **isn't significant autocorrelation in the series**

Task 2- Define the models + fit them + get the performance and plot

```
#Madpis_Second_read_data_output <- output_Second_Read_Data

Madpis_Third_output <- Madpis_Third_models(
   Madpis_Second_read_data_output = output_Second_Read_Data,
   include_RNN = T,
   plot_all_models = TRUE,
   print_accuracy_all_models = TRUE,
   show_top_3_models = TRUE,
```

```r
  plot_MLP = FALSE, write_datasets_for_RNN = F,
  dir_name_save_dataset = "datasets_for_RNN"
)
```

```
## New names:
## Rows: 23 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## New names:
## Rows: 8 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘
```

```
## Warning in tk_tbl.data.frame(as.data.frame(data), preserve_index, rename_index,
## : Warning: No index to preserve. Object otherwise converted to tibble
## successfully.
```

```
## Warning in tk_tbl.data.frame(as.data.frame(data), preserve_index, rename_index,
## : Warning: No index to preserve. Object otherwise converted to tibble
## successfully.
```

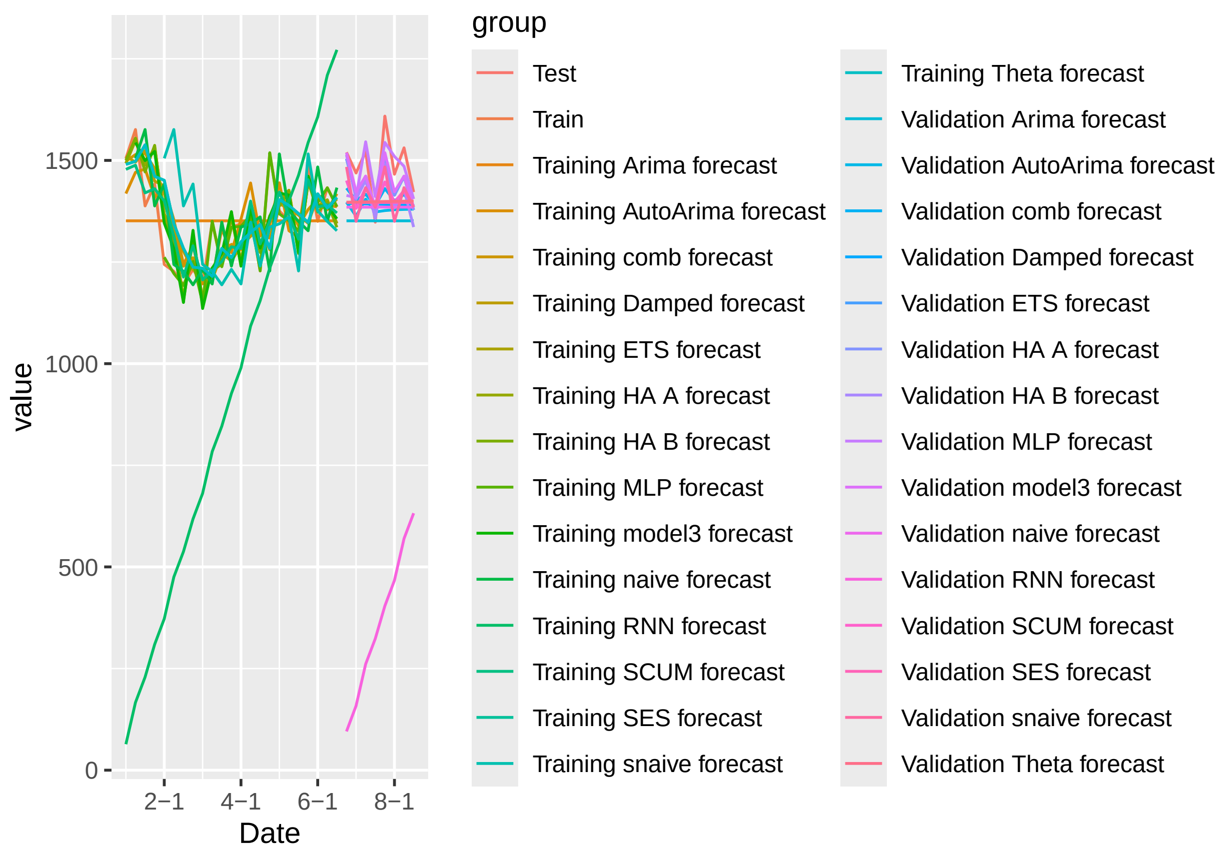
```
## [1] "KPSS Test p-value is higher than 0.05 Thus we need to Accept\nH0: The data is **Stationary**"
```

```
## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
## Joining with ‘by = join_by(Date)‘
```

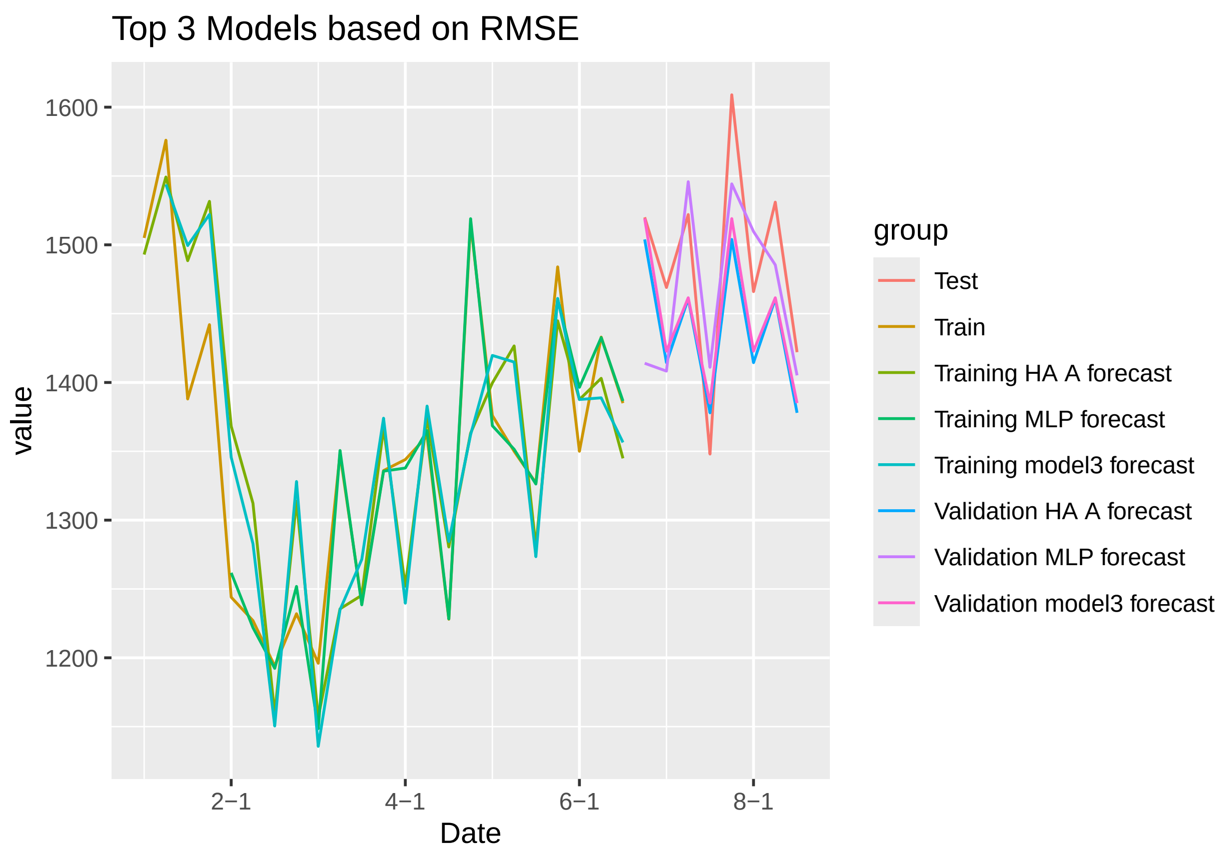
```
## Warning in tk_tbl.data.frame(as.data.frame(data), preserve_index, rename_index,
## : Warning: No index to preserve. Object otherwise converted to tibble
## successfully.
```

```
## Warning in tk_tbl.data.frame(as.data.frame(data), preserve_index, rename_index,
## : Warning: No index to preserve. Object otherwise converted to tibble
## successfully.
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (‘geom_line()‘).
```



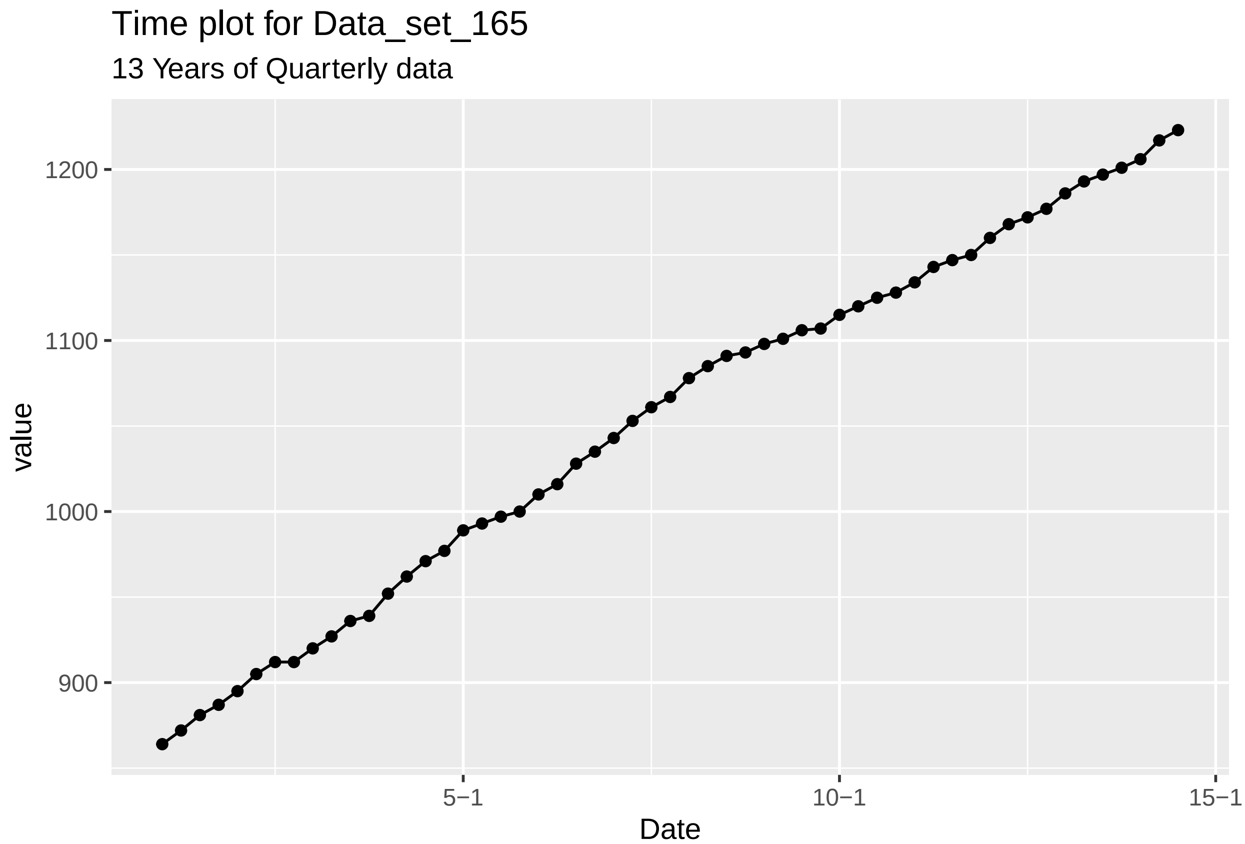

```
## # A tibble: 30 x 10
##    Model  Set             ME  RMSE   MAE    MPE  MAPE  MASE    ACF1 `Theil's U`
##    <chr>  <chr>        <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>       <dbl>
##  1 naive  Trainin set  -5.45 118.   93.1 -0.744  6.84 0.873 -0.598         NA
##  2 naive  Test Set    101.   125.  110.    6.56  7.24 1.03  -0.676          0.883
##  3 snaive Trainin set -13.6  141.  107.   -1.50  8.25 1      0.620         NA
##  4 snaive Test Set     72.9   90.0  82.1  4.78   5.47 0.770 -0.270          0.656
##  5 model3 Trainin set  NA     72.4  63.4 NA      4.74 0.666 NA             NA
##  6 model3 Test Set     NA     54.0  48.1 NA      3.20 0.603 NA             NA
##  7 SES    Trainin set  -7.91  98.2  77.7 -0.936  5.77 0.729 -0.152         NA
##  8 SES    Test Set     89.4  116.  102.    5.78  6.68 0.952 -0.676          0.823
##  9 HA A   Trainin set  -4.14  70.3  58.6 -0.439  4.36 0.549  0.0559        NA
## 10 HA A   Test Set     46.6   59.6  54.1  3.04   3.59 0.507 -0.485          0.448
## # i 20 more rows
## # A tibble: 3 x 10
##   Model  Set         ME  RMSE   MAE   MPE  MAPE  MASE   ACF1 `Theil's U`
##   <chr>  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>       <dbl>
## 1 model3 Test Set  NA    54.0  48.1 NA     3.20 0.603 NA            NA
## 2 MLP    Test Set  20.4  59.1  53.1  1.26  3.56 0.497 -0.117         0.337
## 3 HA A   Test Set  46.6  59.6  54.1  3.04  3.59 0.507 -0.485         0.448

## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



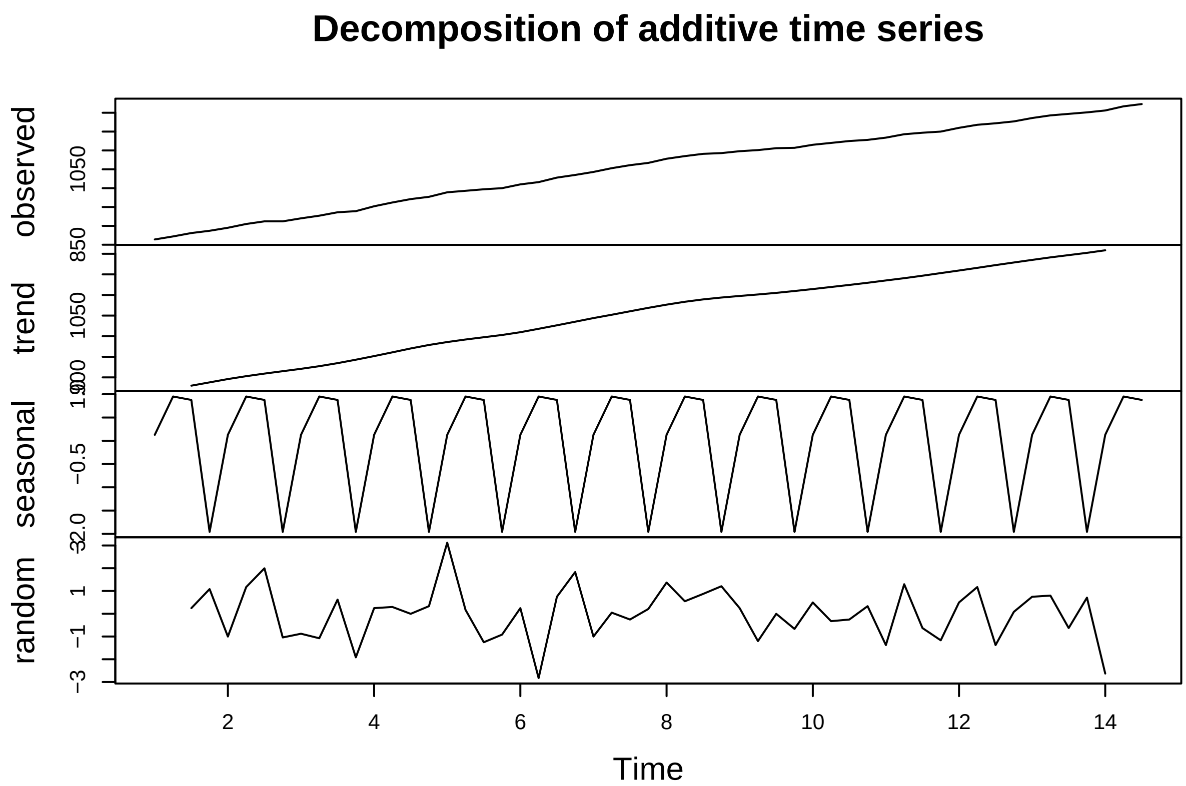
Dataset #7- Evaluate data set Q165:

Task 1- Select dataset Q-165 and preprocess it + Plot the data

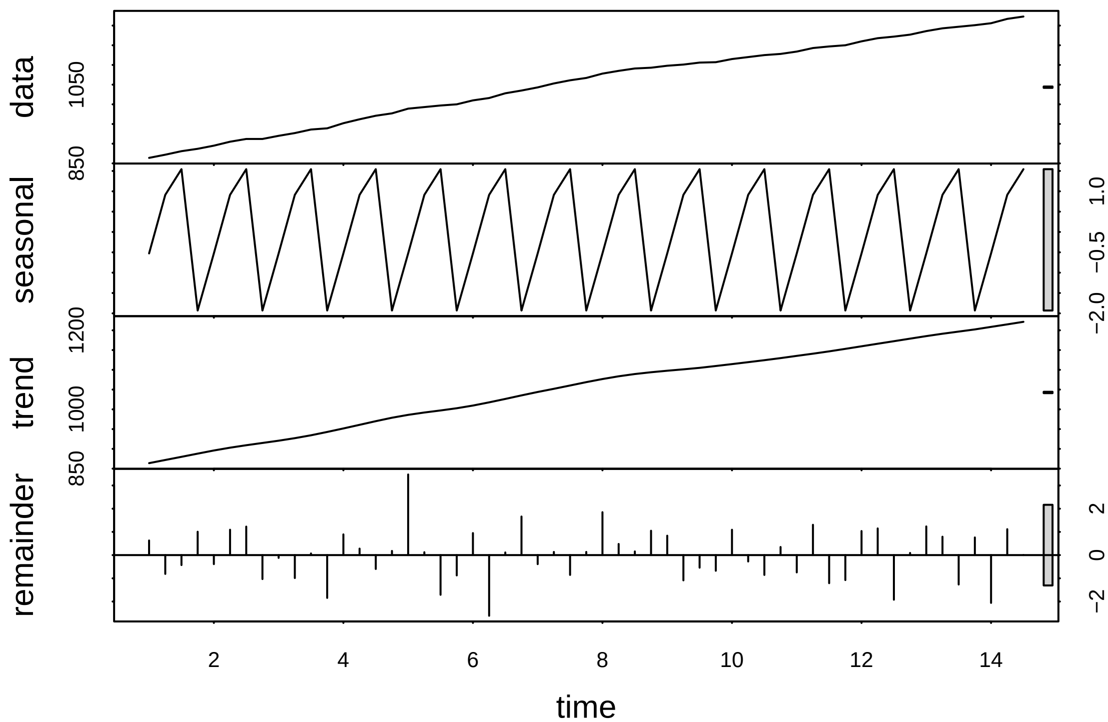
```
data_set_to_load <- c("Data_set_165")

output_Second_Read_Data <- Madpis_Second_read_data(
   Madpis_Initial_read_data_output = output_Inital_Read_Data,
   data_set_to_load = data_set_to_load,
   plot_dataset = TRUE,
   plot_decompose = TRUE, plot_seasonal = TRUE)
```



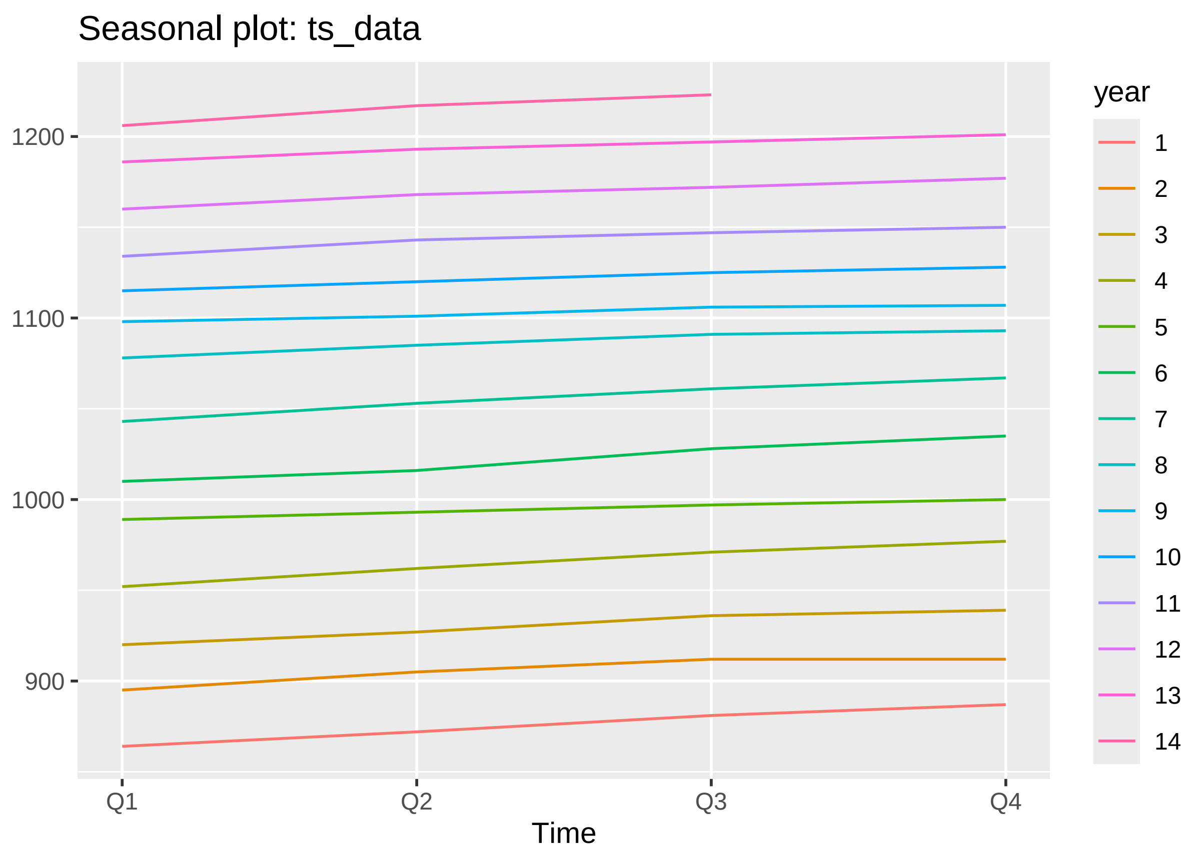
Time plot for Data_set_165
13 Years of Quarterly data
1200
1100
1000
900
value
5−1
10−1
15−1
Date



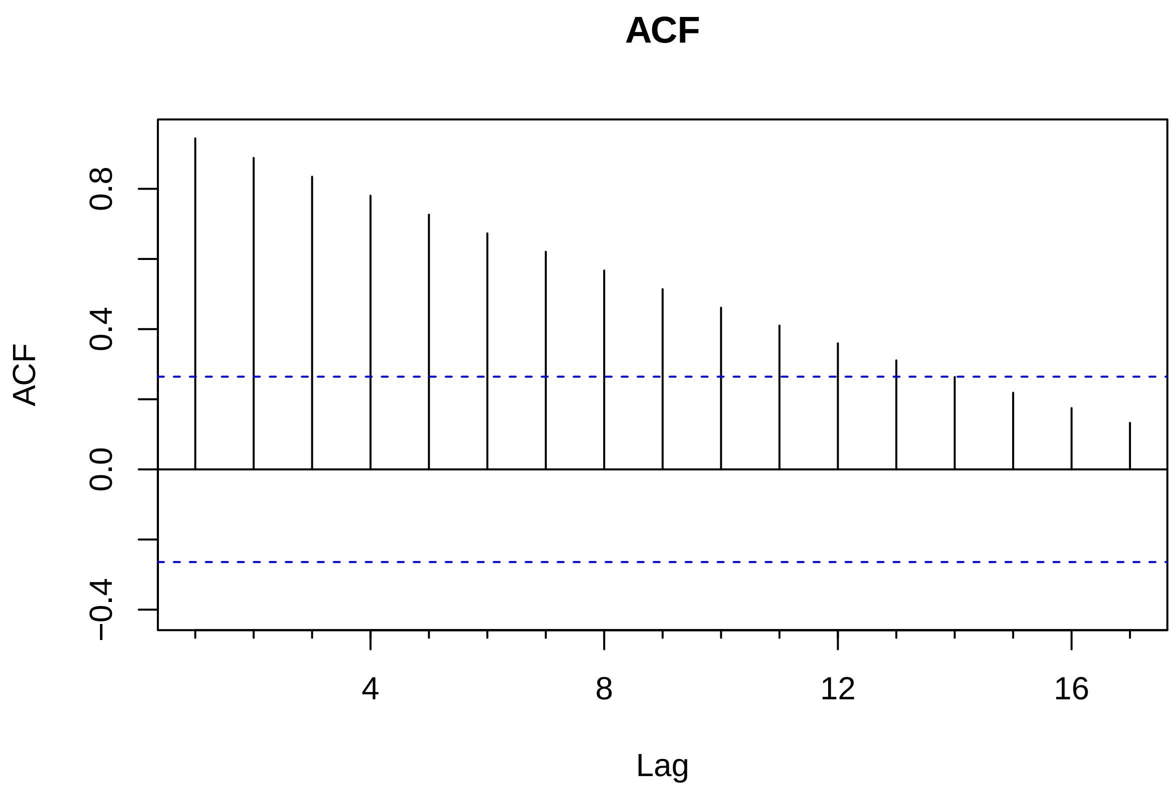
Decomposition of additive time series
observed
850
1050
trend
900
1050
seasonal
−2.0
−0.5
1.0
random
−3
−1
1
3
2
4
6
8
10
12
14
Time

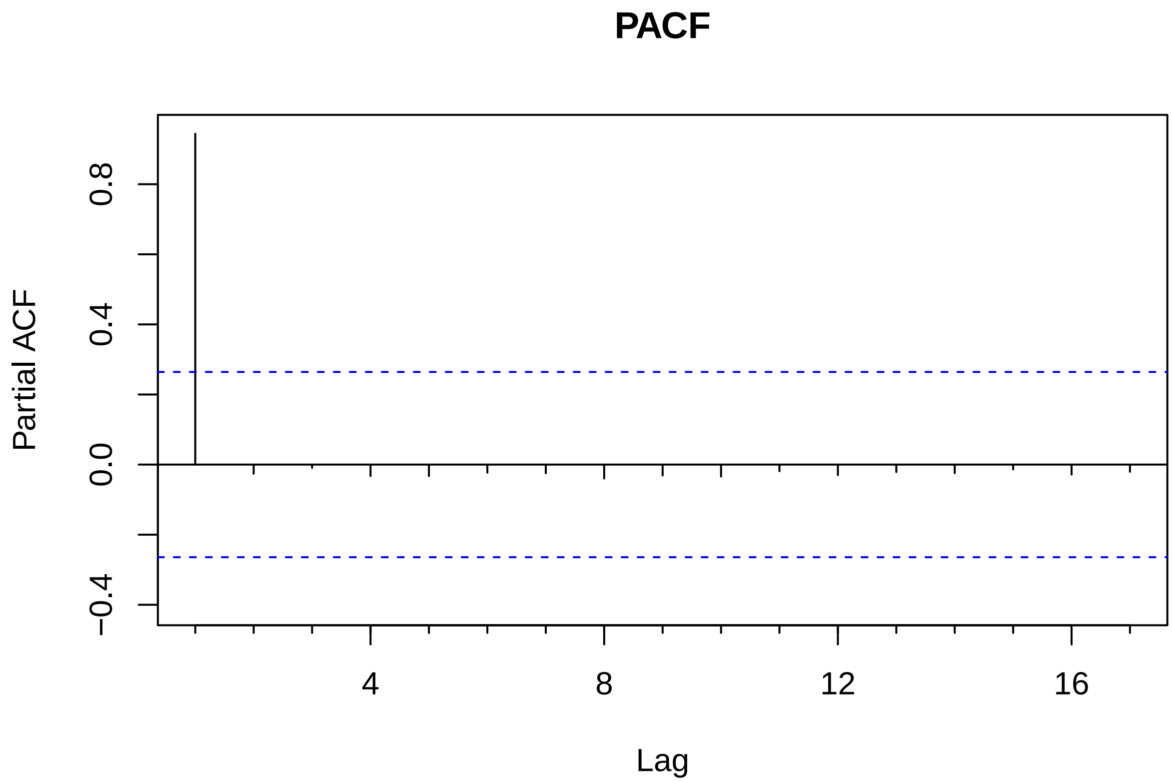


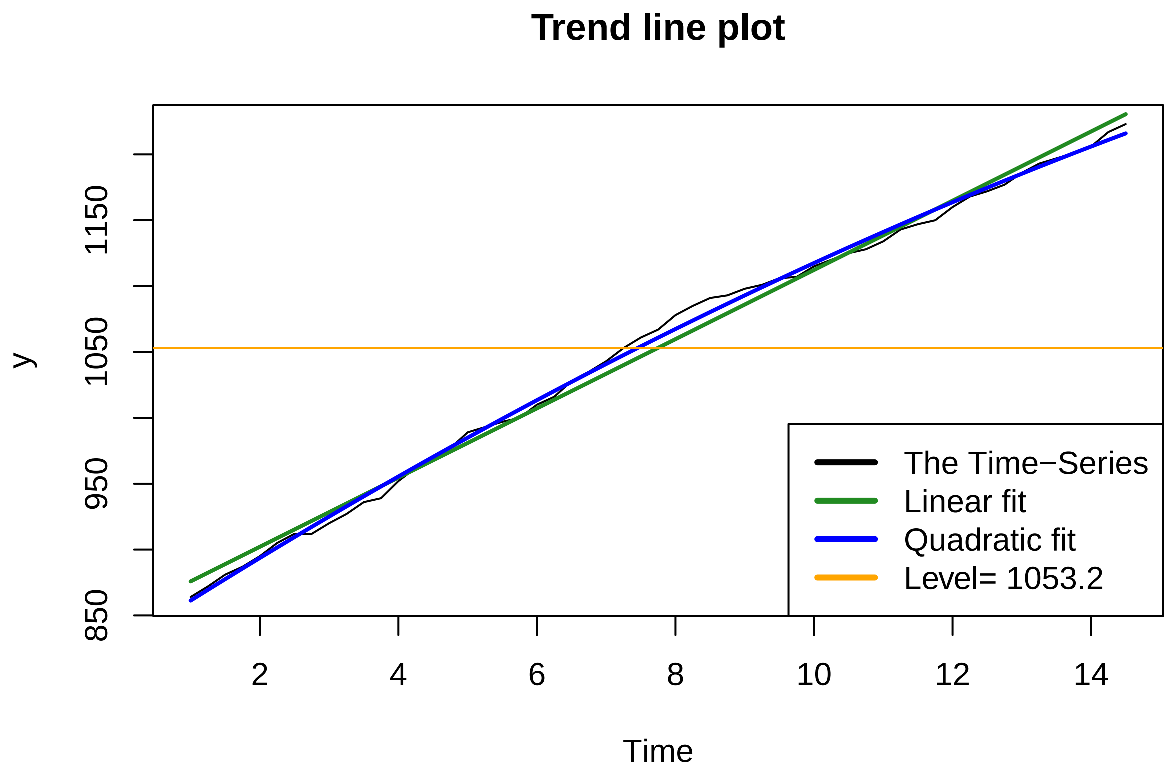
data
1050
850
seasonal
1.0
−0.5
−2.0
1200
trend
1000
850
remainder
2
0
−2
2
4
6
8
10
12
14
time



Seasonal plot: ts_data
1200
1100
1000
900
Q1
Q2
Q3
Q4
Time
year
1
2
3
4
5
6
7
8
9
10
11
12
13
14


ACF
0.8
0.4
0.0
−0.4
ACF
4
8
12
16
Lag


PACF
Partial ACF
0.8
0.4
0.0
−0.4
4
8
12
16
Lag




```
#tsibble_data <- output_Second_Read_Data$tsibble_quarterly_train
```

Notes on the time-series:

- We can see that there is **13 years of quarterly data**
- From the **Time plot** we can see that there is an overall increase trend.
- From the **Decomposition graph** and the **seasonality graph** we can see there is seasonality in the series
- From the **ACF** and **Pacf** plots we can understand there **is an autocorrelation in the series**, the **Pacf** plot reveals that most of that autocorrelation originate from the 1st lag autocorrelation.

Task 2- Define the models + fit them + get the performance and plot

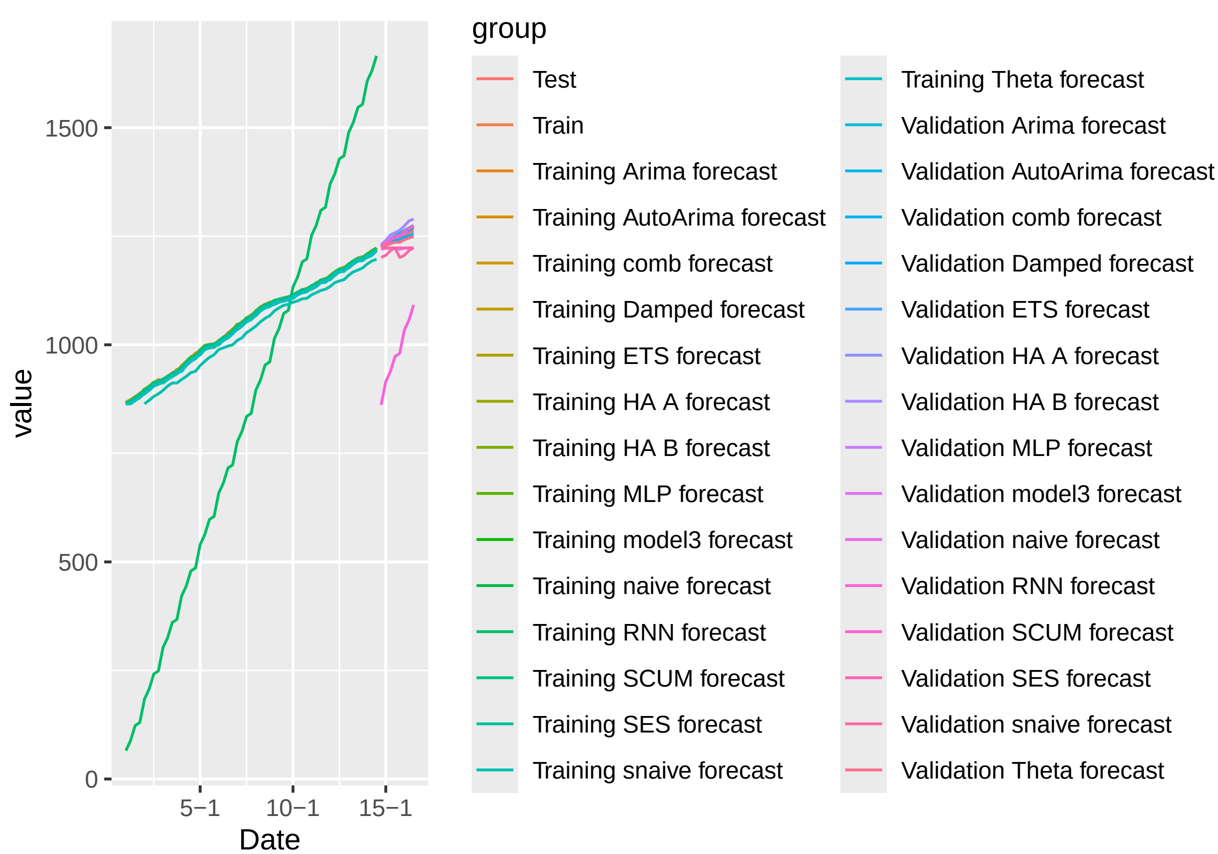
```
#Madpis_Second_read_data_output <- output_Second_Read_Data

Madpis_Third_output <- Madpis_Third_models(
   Madpis_Second_read_data_output = output_Second_Read_Data,
   include_RNN = T,
   plot_all_models = TRUE,
   print_accuracy_all_models = TRUE,
   show_top_3_models = TRUE,
```

```
    plot_MLP = FALSE, write_datasets_for_RNN = F,
    dir_name_save_dataset = "datasets_for_RNN"
)
```

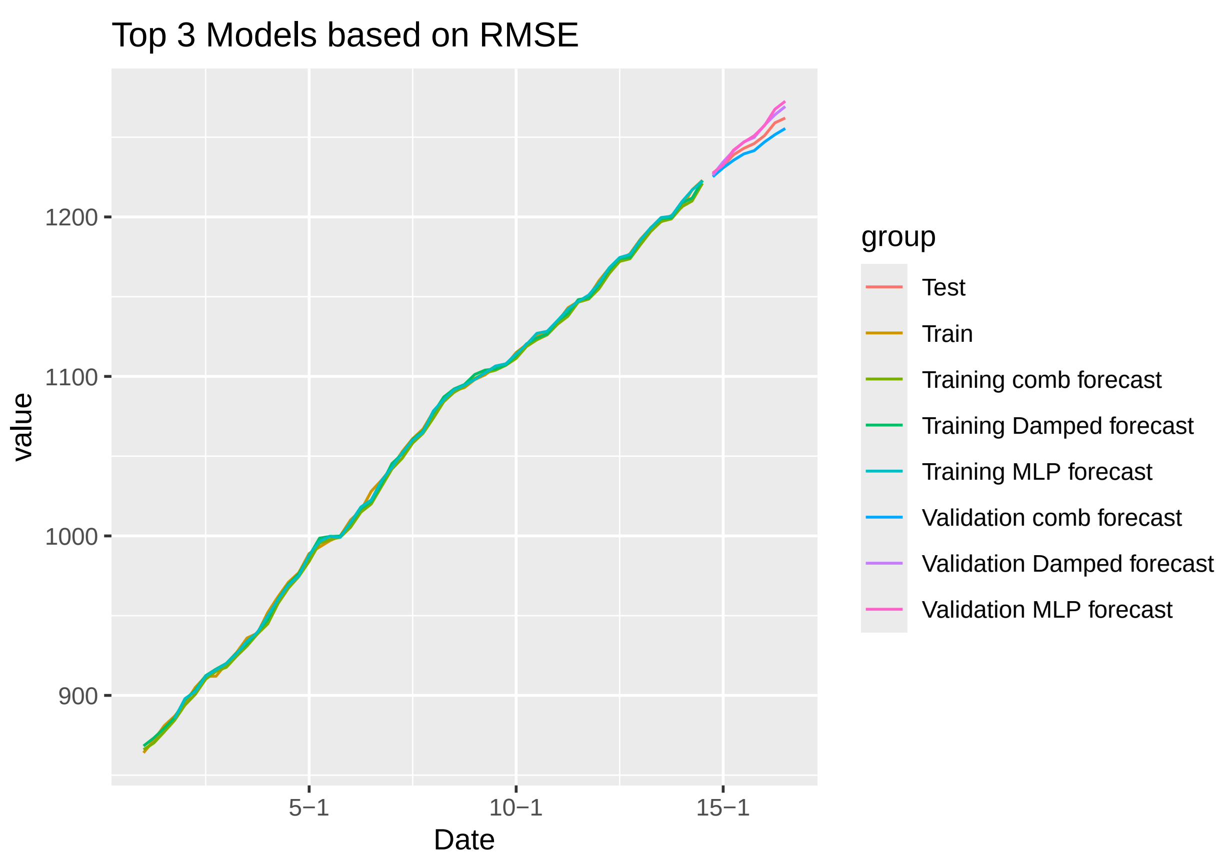
```
## New names:
## Rows: 55 Columns: 8
## -- Column specification
## ---------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## New names:
## Rows: 8 Columns: 8
## -- Column specification
## ---------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
## [1] "KPSS Test p-value is lower than 0.05 Thus we need to reject\nH0: The data is **NOT Stationary**
```

```
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
```



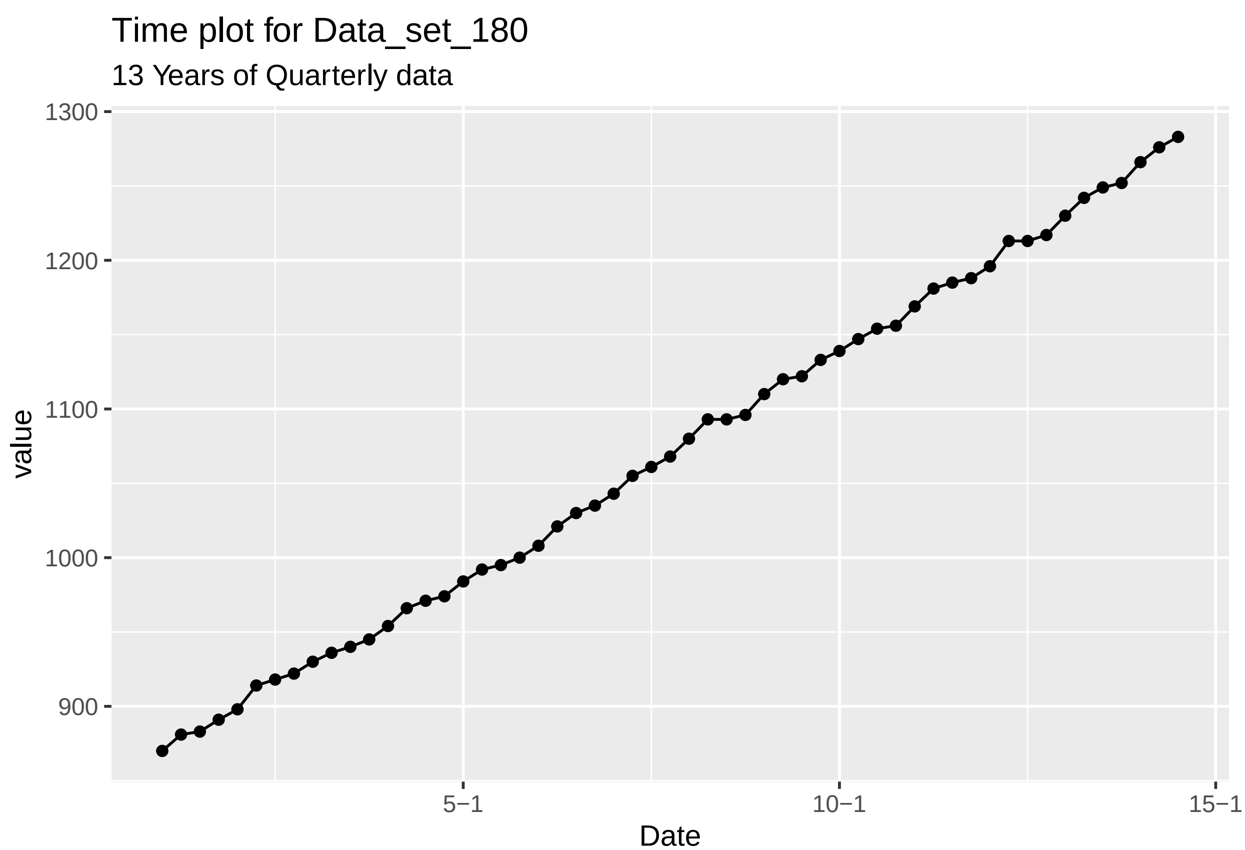
```
## # A tibble: 30 x 10
##    Model  Set          ME  RMSE   MAE     MPE  MAPE   MASE     ACF1 `Theil's U`
##    <chr>  <chr>     <dbl> <dbl> <dbl>   <dbl> <dbl>  <dbl>    <dbl>       <dbl>
##  1 naive  Trainin~  6.65   7.26  6.65  0.641  0.641  0.252   0.0179        NA
##  2 naive  Test Set 21.9   24.7  21.9   1.75   1.75   0.831   0.616          4.94
##  3 snaive Trainin~ 26.3   27.0  26.3   2.51   2.51   1       0.863         NA
##  4 snaive Test Set 33.1   34.6  33.1   2.66   2.66   1.26    0.528          6.70
##  5 model3 Trainin~ NA      7.02  6.63 NA      0.640  0.999  NA             NA
##  6 model3 Test Set NA     25.3  22.8  NA      1.82  18.1    NA             NA
##  7 SES    Trainin~  6.53   7.19  6.53  0.629  0.629  0.248  -0.00130       NA
##  8 SES    Test Set 21.9   24.7  21.9   1.75   1.75   0.831   0.616          4.94
##  9 HA A   Trainin~ -0.167  2.29  1.83 -0.0178 0.175  0.0694  0.0159        NA
## 10 HA A   Test Set -5.85   7.00  6.02 -0.467  0.481  0.229   0.472          1.40
## # i 20 more rows
## # A tibble: 3 x 10
##   Model  Set         ME  RMSE   MAE    MPE  MAPE  MASE   ACF1 `Theil's U`
##   <chr>  <chr>    <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>       <dbl>
## 1 comb   Test Set NA     4.51  4.05 NA     0.324 0.938 NA          NA
## 2 Damped Test Set -3.88  4.54  4.05 -0.310 0.324 0.154  0.338       0.909
## 3 MLP    Test Set -4.77  5.83  4.77 -0.381 0.381 0.181  0.612       1.16
```

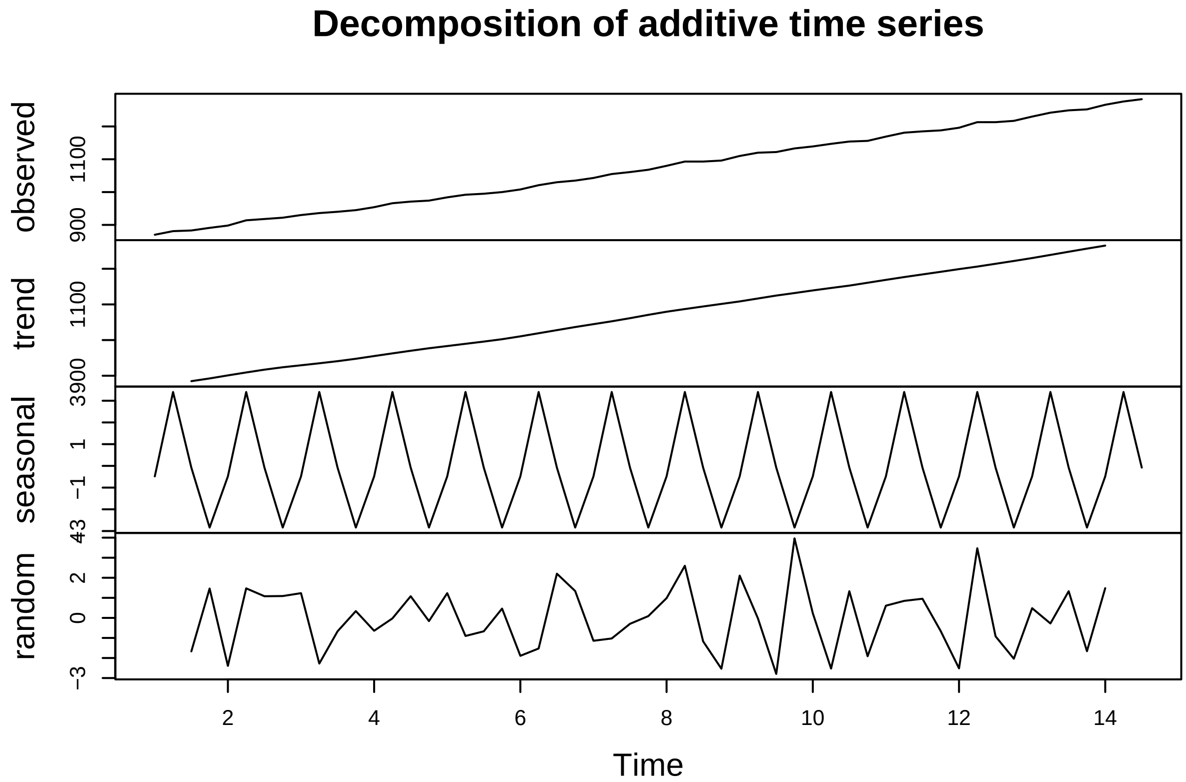


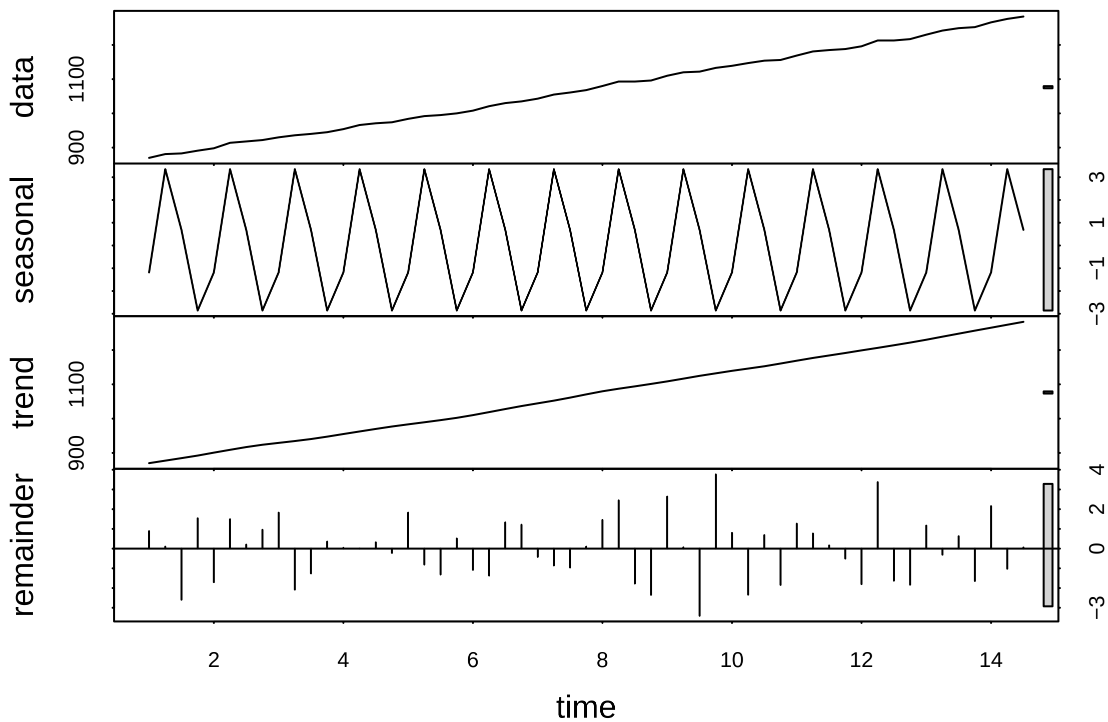
Dataset #8- Evaluate data set Q180:

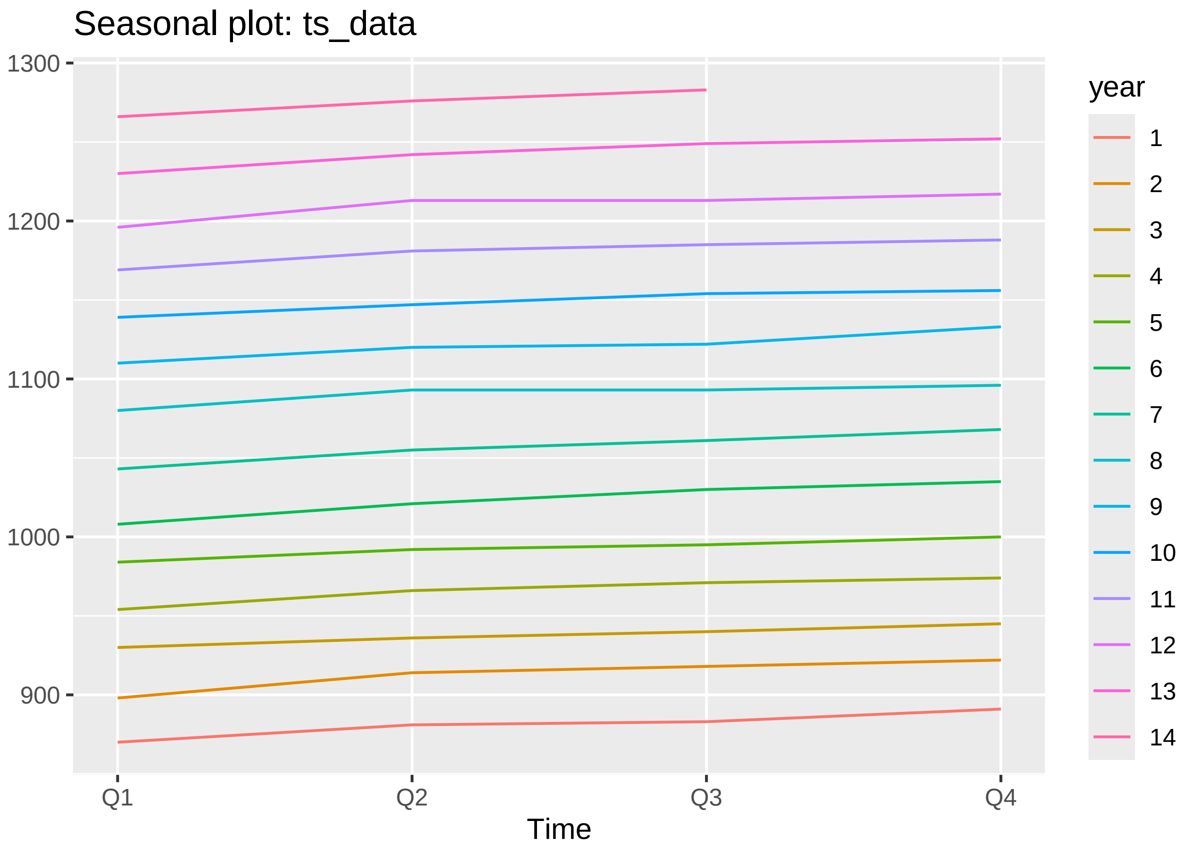
Task 1- Select dataset Q-180 and preprocess it + Plot the data

```
data_set_to_load <- c("Data_set_180")

output_Second_Read_Data <- Madpis_Second_read_data(
   Madpis_Initial_read_data_output = output_Inital_Read_Data,
   data_set_to_load = data_set_to_load,
   plot_dataset = TRUE,
   plot_decompose = TRUE, plot_seasonal = TRUE)
```

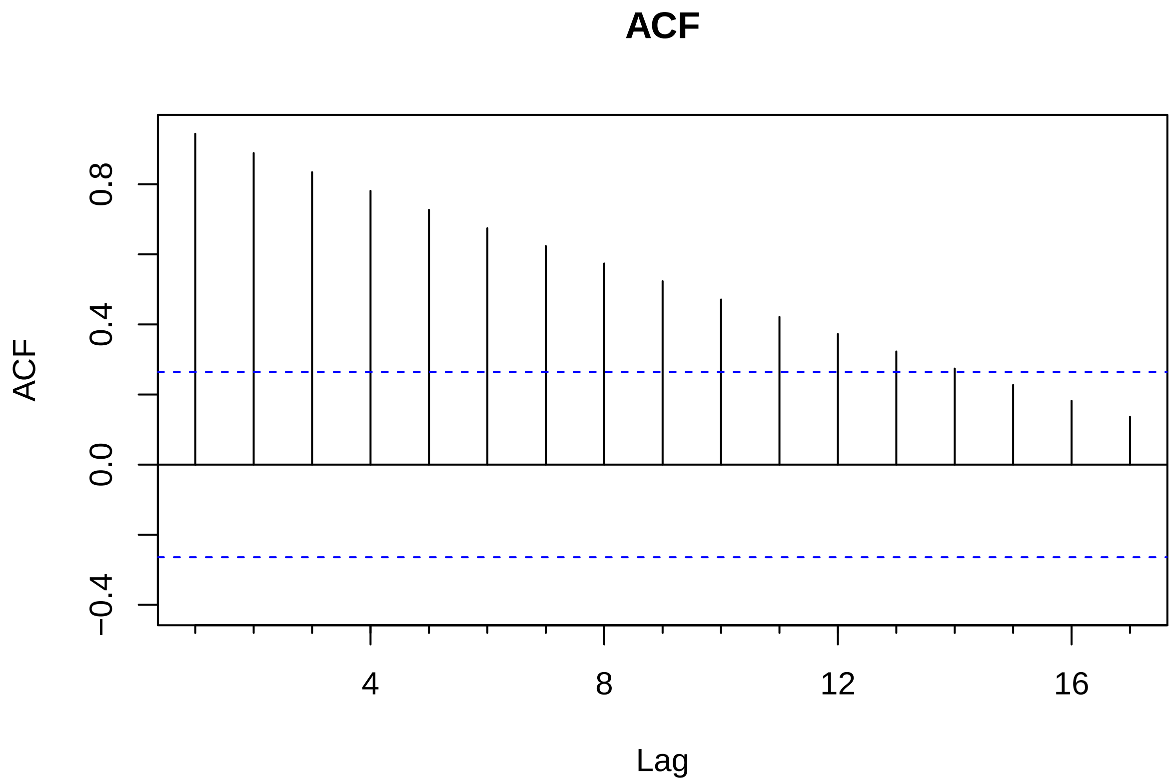


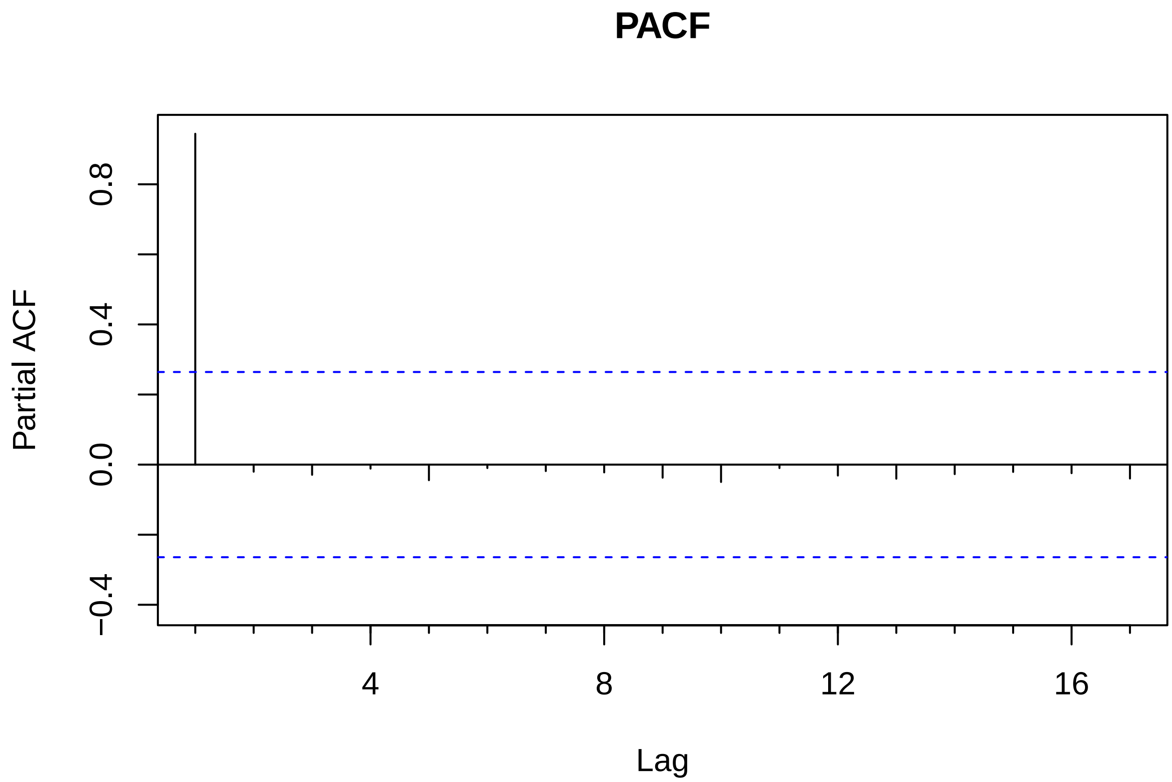

Decomposition of additive time series
observed
900
1100
trend
900
1100
seasonal
-3
-1
1
3
random
-3
0
2
4
2
4
6
8
10
12
14
Time

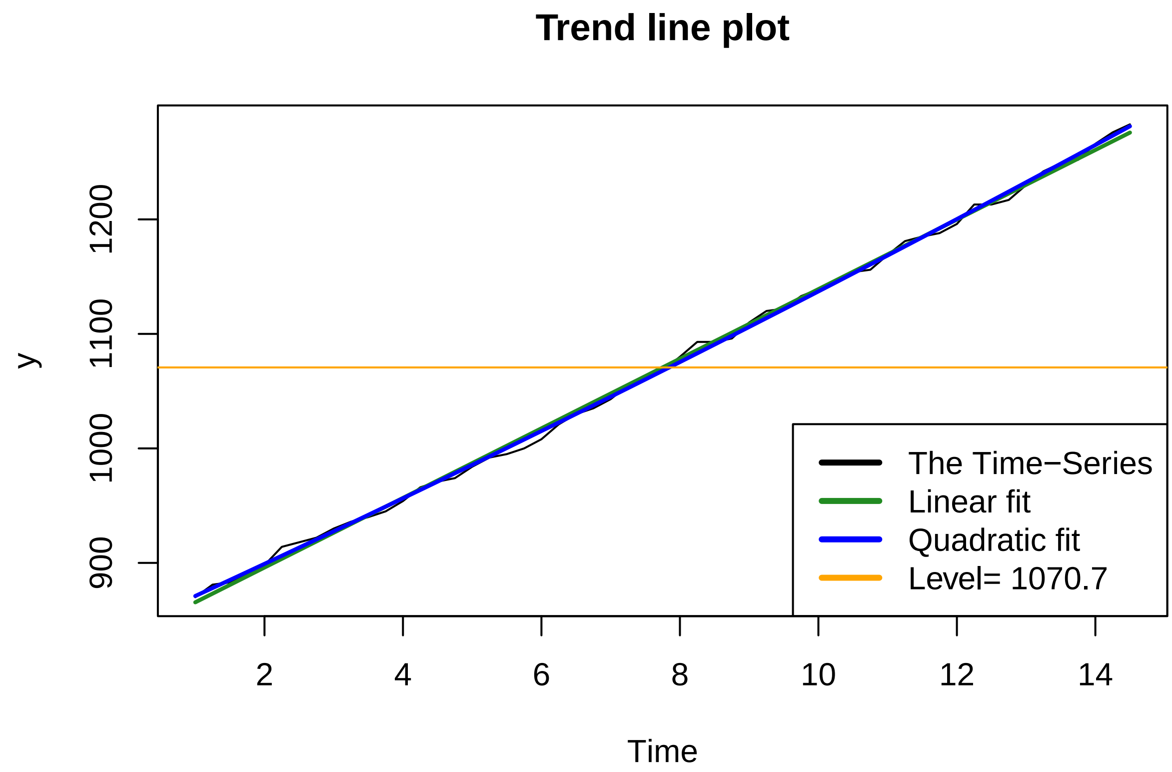

data
900
1100
seasonal
3
1
−1
−3
trend
900
1100
remainder
4
2
0
−3
2
4
6
8
10
12
14
time



Seasonal plot: ts_data
1300
1200
1100
1000
900
Q1
Q2
Q3
Q4
Time
year
1
2
3
4
5
6
7
8
9
10
11
12
13
14


ACF
ACF
0.8
0.4
0.0
−0.4
4
8
12
16
Lag


PACF
Partial ACF
0.8
0.4
0.0
−0.4
4
8
12
16
Lag




```
#tsibble_data <- output_Second_Read_Data$tsibble_quarterly_train
```

Notes on the time-series:

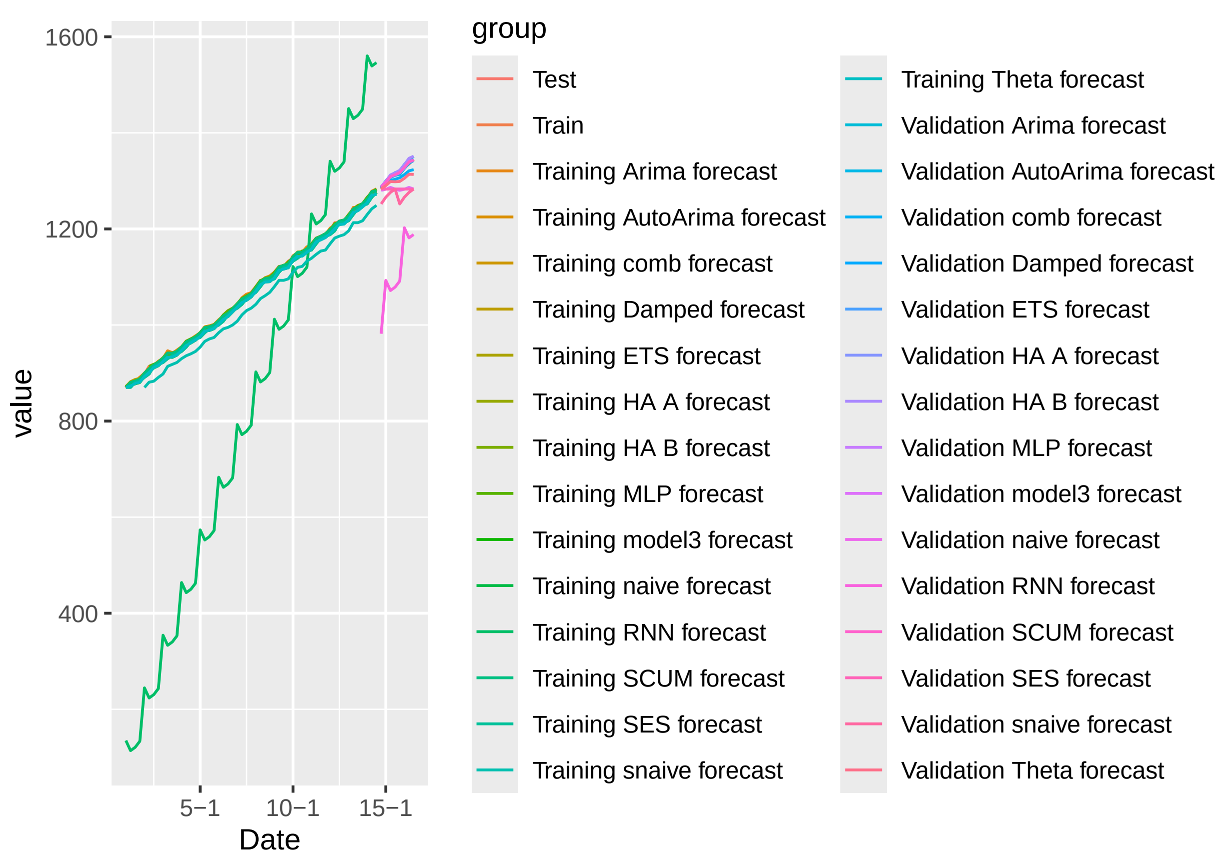
- We can see that there is **13 years of quarterly data**
- From the **Time plot** we can see that there is an overall increase trend.
- From the **Decomposition graph** and the **seasonality graph** we can see there is seasonality in the series
- From the **ACF** and **Pacf** plots we can understand there **is an autocorrelation in the series**, the **Pacf** plot reveals that most of that autocorrelation originate from the 1st lag autocorrelation.

Task 2- Define the models + fit them + get the performance and plot

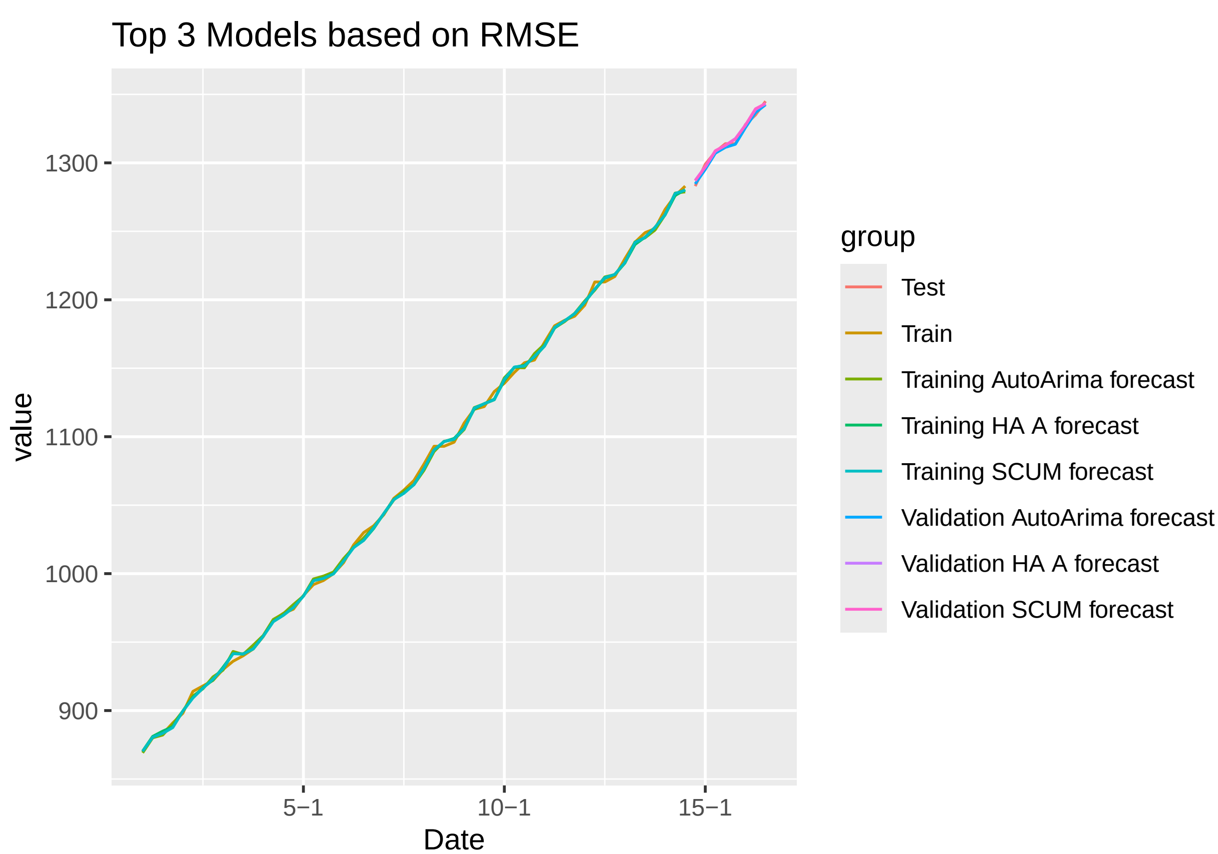
```
#Madpis_Second_read_data_output <- output_Second_Read_Data

Madpis_Third_output <- Madpis_Third_models(
   Madpis_Second_read_data_output = output_Second_Read_Data,
   include_RNN = T,
   plot_all_models = TRUE,
   print_accuracy_all_models = TRUE,
   show_top_3_models = TRUE,
```

```r
    plot_MLP = FALSE, write_datasets_for_RNN = F,
    dir_name_save_dataset = "datasets_for_RNN"
)
```

```
## New names:
## Rows: 55 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## New names:
## Rows: 8 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`

## [1] "KPSS Test p-value is lower than 0.05 Thus we need to reject\nH0: The data is **NOT Stationary**

## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
```



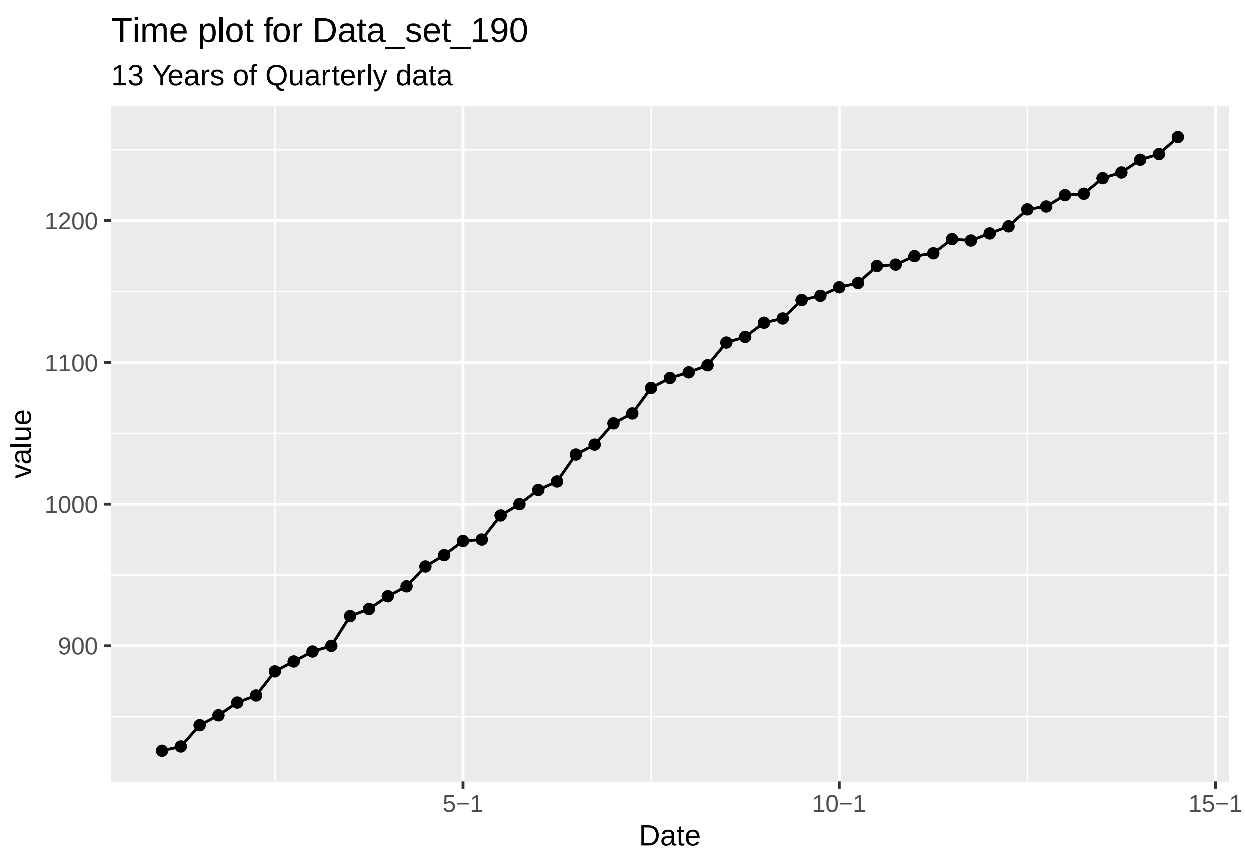
```
## # A tibble: 30 x 10
##    Model  Set          ME  RMSE   MAE     MPE  MAPE   MASE    ACF1 `Theil's U`
##    <chr>  <chr>     <dbl> <dbl> <dbl>   <dbl> <dbl>  <dbl>   <dbl>       <dbl>
##  1 naive  Trainin ~  7.65  8.69  7.65  0.716  0.716  0.251  -0.133         NA
##  2 naive  Test Set  32.8   37.7  32.8   2.47   2.47   1.08    0.527         3.93
##  3 snaive Trainin ~ 30.4   30.7  30.4   2.82   2.82   1       0.500        NA
##  4 snaive Test Set  46.5   48.8  46.5   3.52   3.52   1.53    0.581         4.97
##  5 model3 Trainin ~ NA      8.09  7.64 NA      0.716  1.01   NA            NA
##  6 model3 Test Set  NA     37.2  32.7  NA      2.46  10.3    NA            NA
##  7 SES    Trainin ~  7.51  8.61  7.51  0.703  0.703  0.247  -0.151         NA
##  8 SES    Test Set  32.8   37.7  32.8   2.47   2.47   1.08    0.527         3.93
##  9 HA A   Trainin ~  0.390  2.56  2.10  0.0322 0.195  0.0692 -0.0437       NA
## 10 HA A   Test Set  -0.903  2.67  2.22 -0.0693 0.169  0.0729 -0.581         0.230
## # i 20 more rows
## # A tibble: 3 x 10
##   Model     Set          ME  RMSE   MAE     MPE  MAPE   MASE   ACF1 `Theil's U`
##   <chr>     <chr>     <dbl> <dbl> <dbl>   <dbl> <dbl>  <dbl>  <dbl>       <dbl>
## 1 AutoArima Test Set  1.16   2.21  1.98  0.0877 0.151 0.0651 -0.597        0.226
## 2 SCUM      Test Set NA      2.66  2.27 NA      0.173 0.282  NA           NA
## 3 HA A      Test Set -0.903  2.67  2.22 -0.0693 0.169 0.0729 -0.581        0.230
```

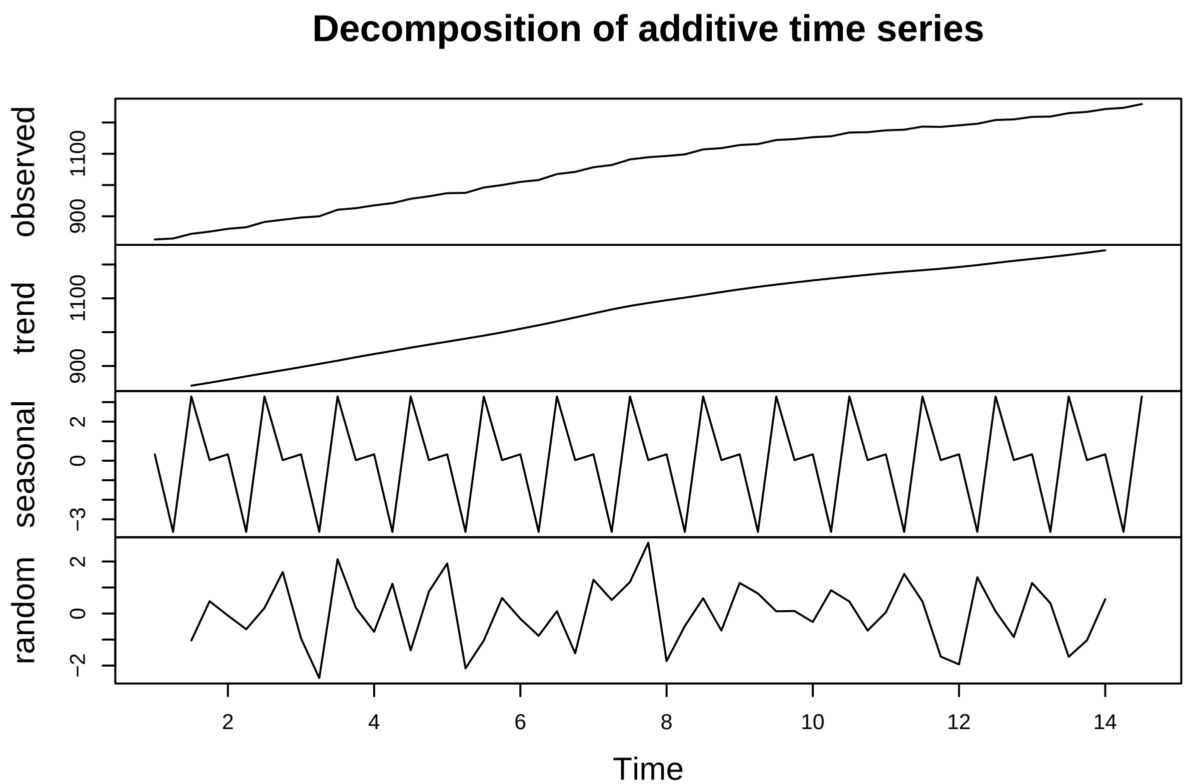


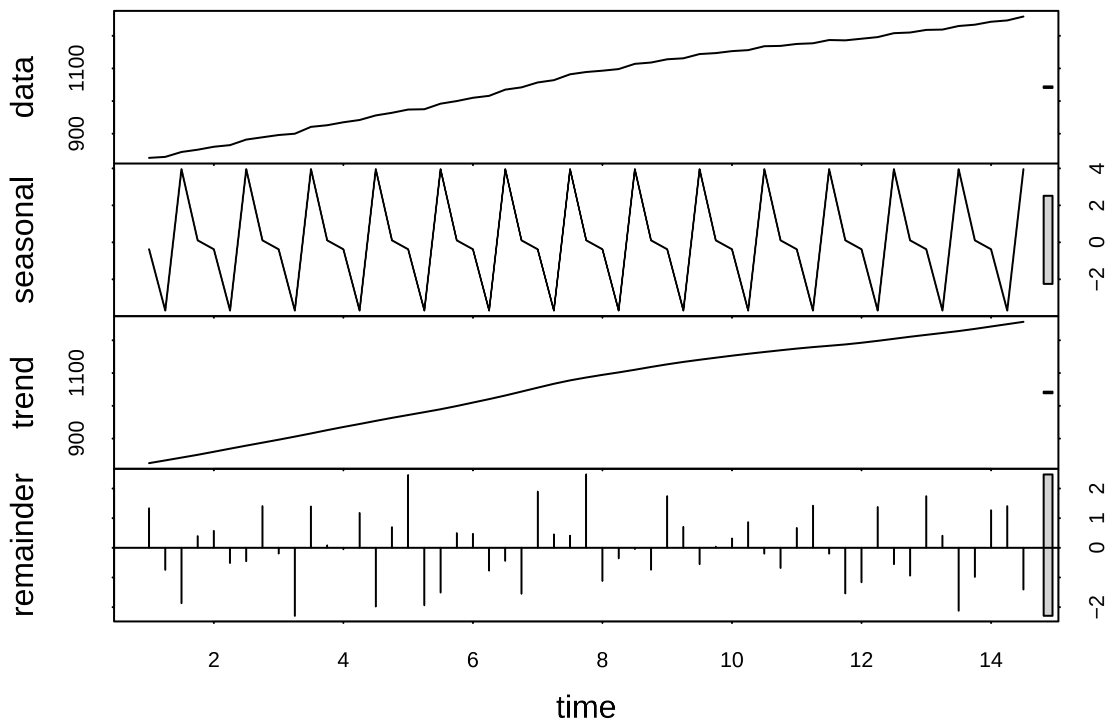
Dataset #9- Evaluate data set Q190:

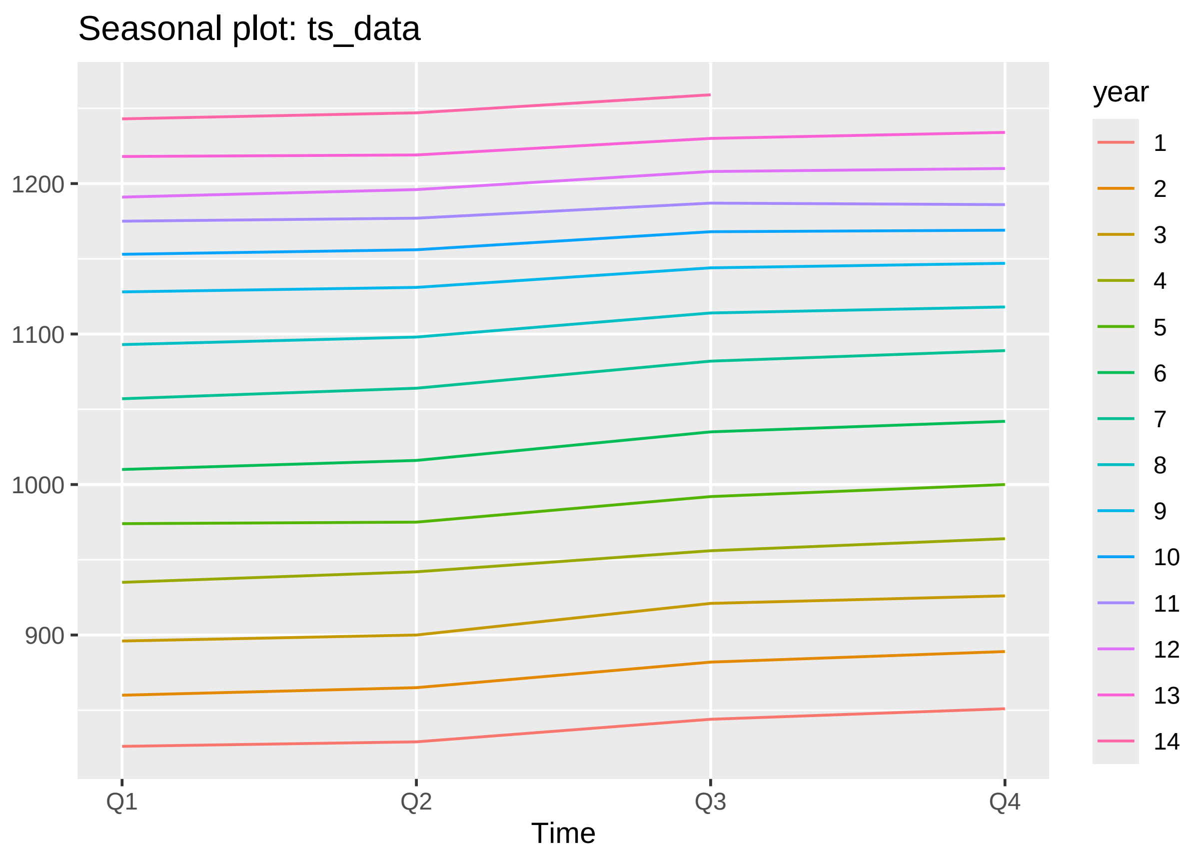
Task 1- Select dataset Q-190 and preprocess it + Plot the data

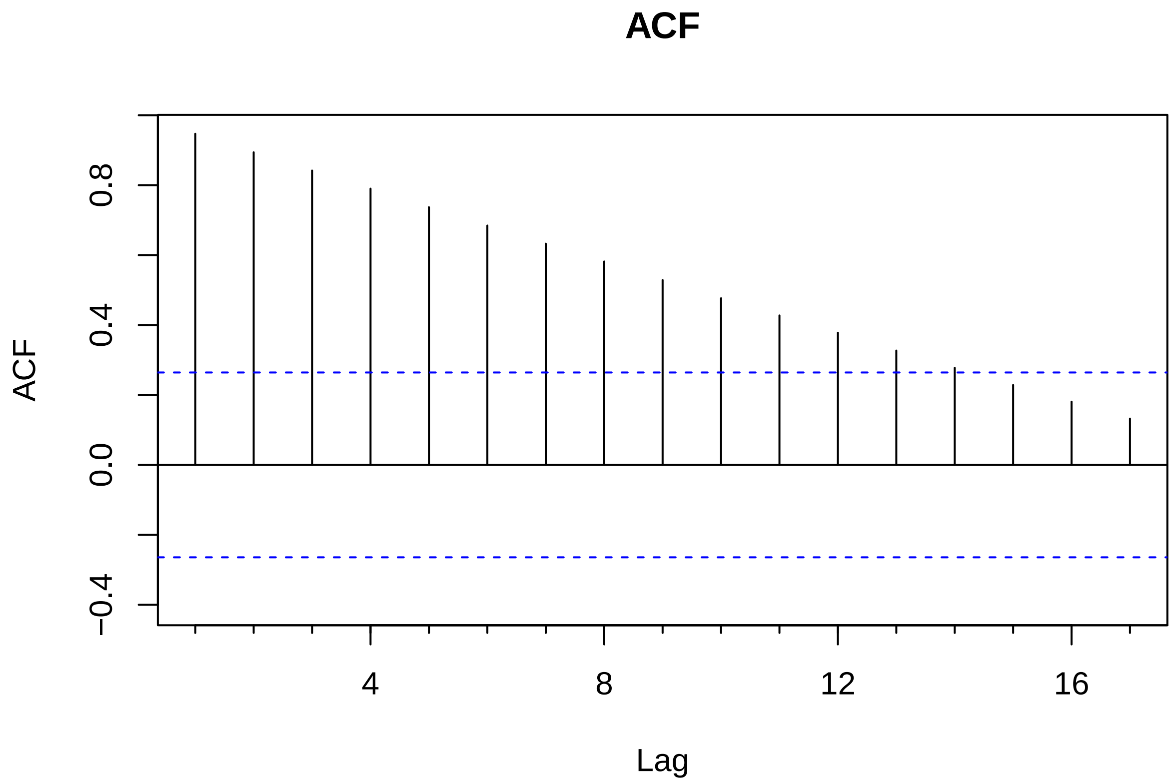
```
data_set_to_load <- c("Data_set_190")

output_Second_Read_Data <- Madpis_Second_read_data(
   Madpis_Initial_read_data_output = output_Inital_Read_Data,
   data_set_to_load = data_set_to_load,
   plot_dataset = TRUE,
   plot_decompose = TRUE, plot_seasonal = TRUE)
```

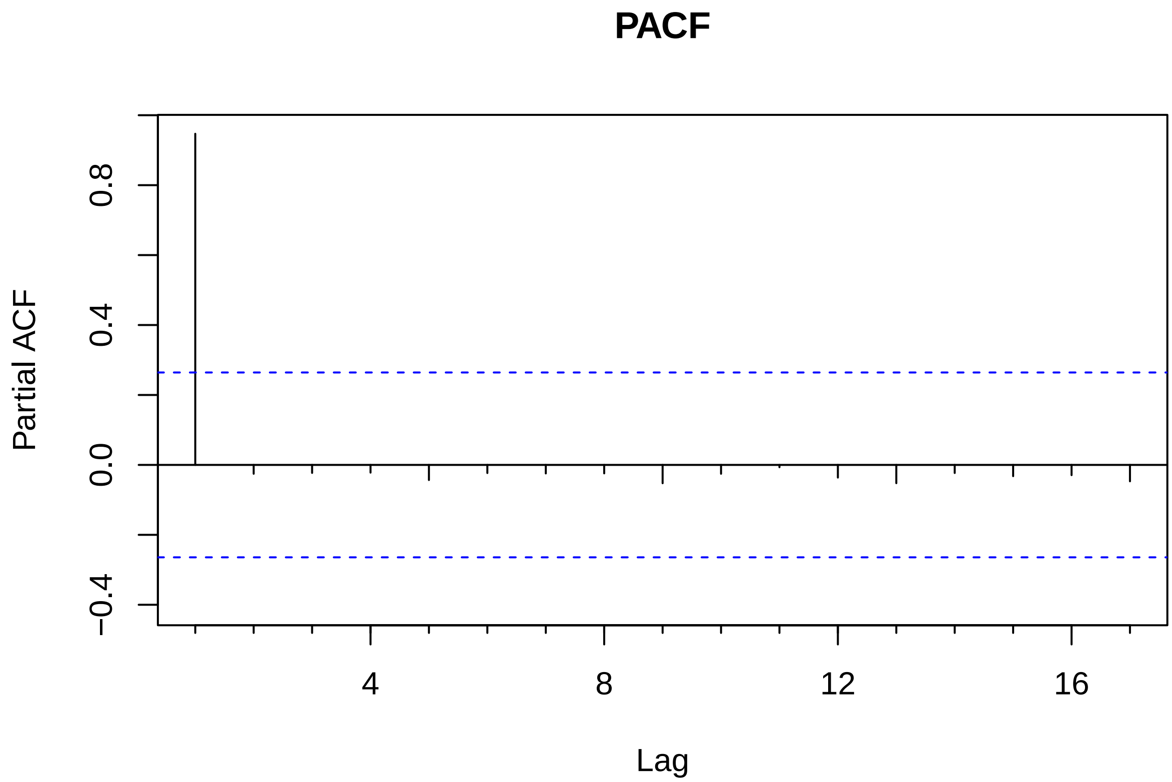



Decomposition of additive time series
observed
1100
900
trend
1100
900
seasonal
2
0
−3
random
2
0
−2
2
4
6
8
10
12
14
Time

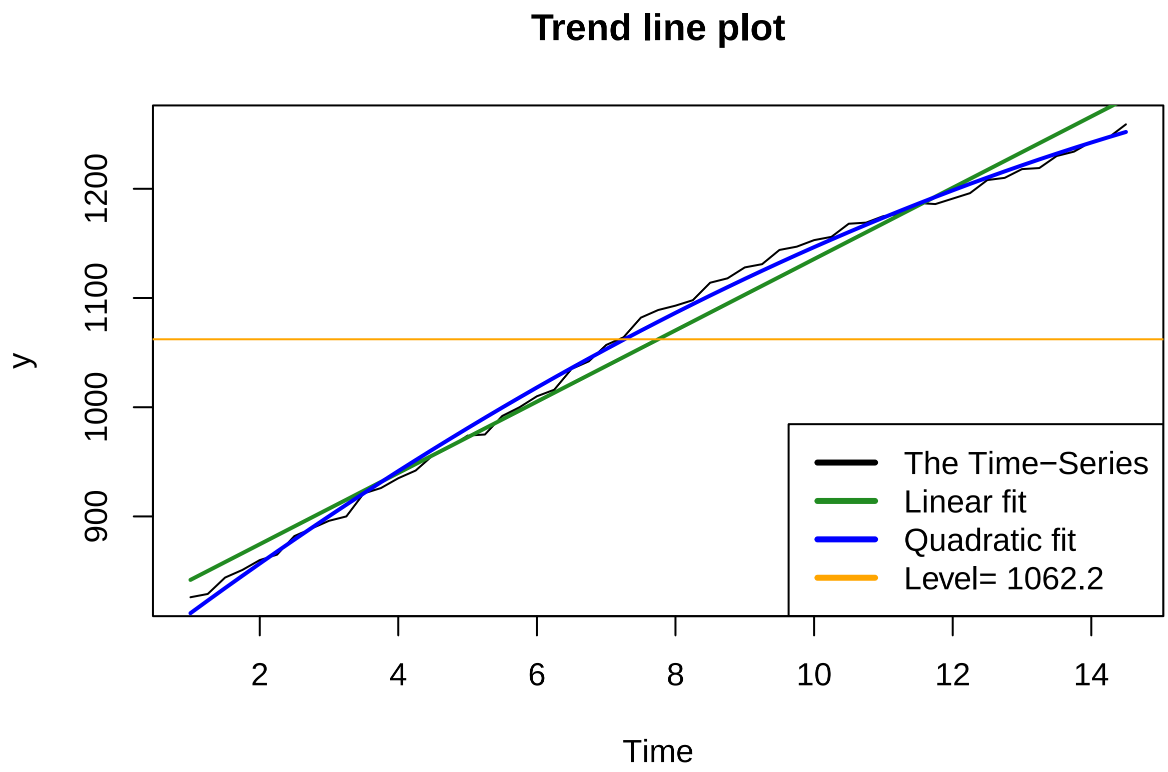

data
900
1100
seasonal
−2 0 2 4
trend
900
1100
remainder
−2 0 1 2
2
4
6
8
10
12
14
time


Seasonal plot: ts_data
1200
1100
1000
900
Q1
Q2
Q3
Q4
Time
year
1
2
3
4
5
6
7
8
9
10
11
12
13
14


ACF
ACF
0.8
0.4
0.0
−0.4
4
8
12
16
Lag



PACF
Partial ACF
0.8
0.4
0.0
−0.4
4
8
12
16
Lag




```r
#tsibble_data <- output_Second_Read_Data$tsibble_quarterly_train
```

Notes on the time-series:

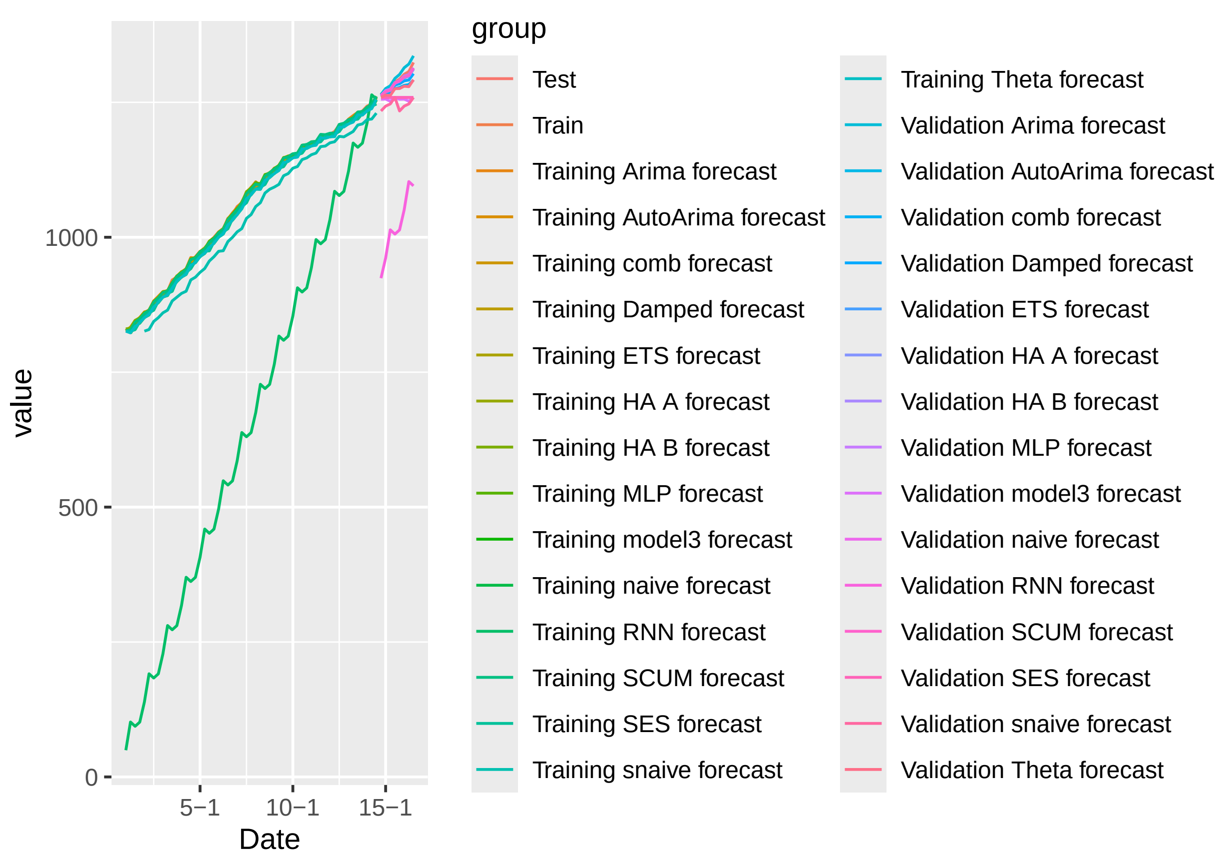
- We can see that there is **13 years of quarterly data**
- From the **Time plot** we can see that there is an overall increase trend.
- From the **Decomposition graph** and the **seasonality graph** we can see there is seasonality in the series
- From the **ACF** and **Pacf** plots we can understand there **is an autocorrelation in the series**, the **Pacf** plot reveals that most of that autocorrelation originate from the 1st lag autocorrelation.

Task 2- Define the models + fit them + get the performance and plot

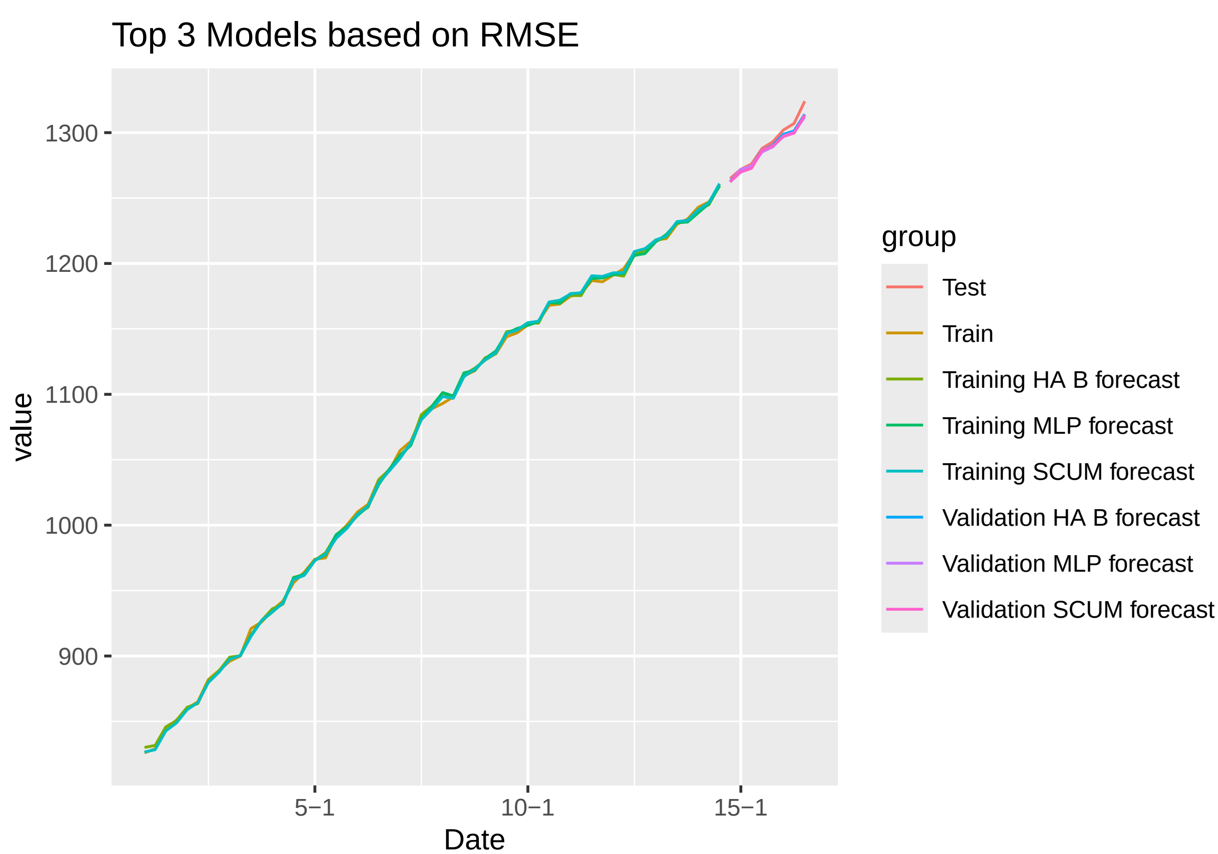
```r
#Madpis_Second_read_data_output <- output_Second_Read_Data

Madpis_Third_output <- Madpis_Third_models(
   Madpis_Second_read_data_output = output_Second_Read_Data,
   include_RNN = T,
   plot_all_models = TRUE,
   print_accuracy_all_models = TRUE,
   show_top_3_models = TRUE,
```

```r
    plot_MLP = FALSE, write_datasets_for_RNN = F,
    dir_name_save_dataset = "datasets_for_RNN"
)
```

```
## New names:
## Rows: 55 Columns: 8
## -- Column specification
## ---------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## New names:
## Rows: 8 Columns: 8
## -- Column specification
## ---------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`

## [1] "KPSS Test p-value is lower than 0.05 Thus we need to reject\nH0: The data is **NOT Stationary**

## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
```



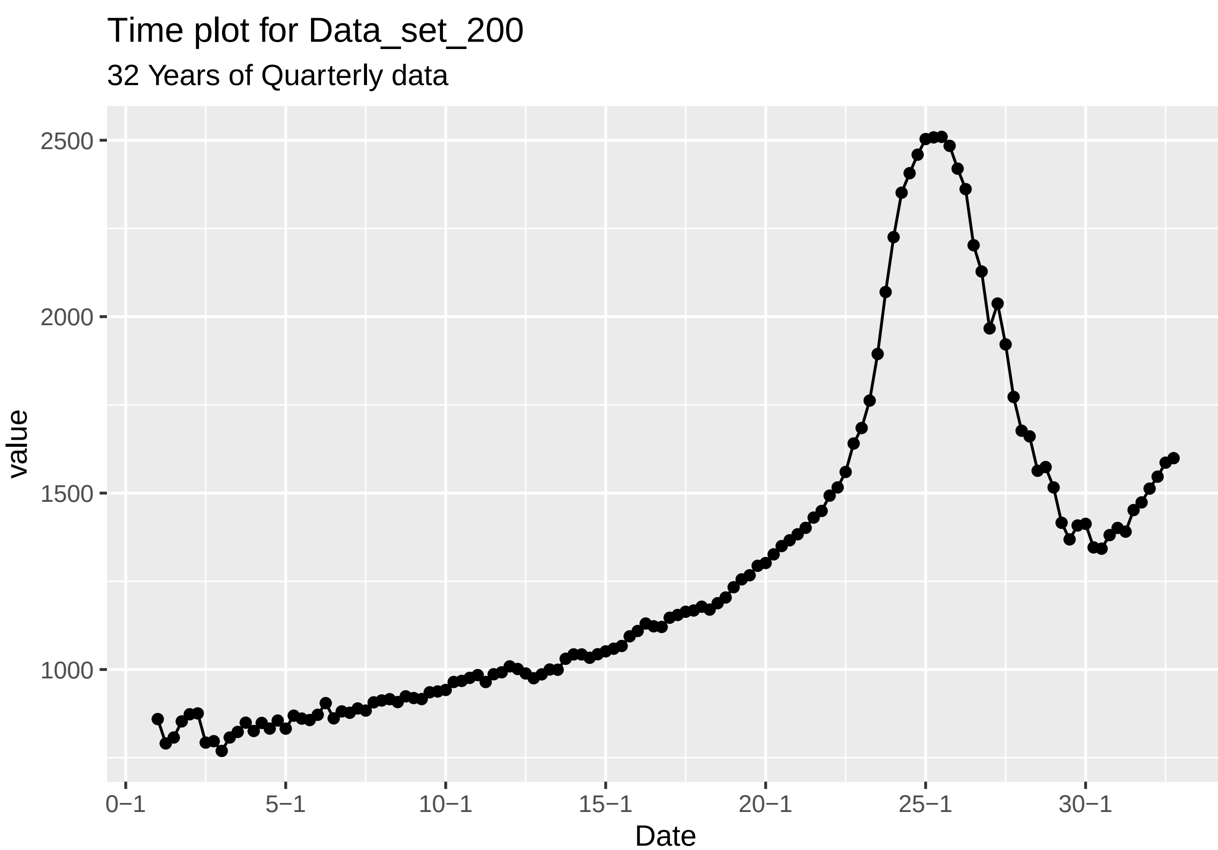
```
## # A tibble: 30 x 10
##    Model  Set            ME  RMSE   MAE      MPE  MAPE   MASE    ACF1 `Theil's U`
##    <chr>  <chr>       <dbl> <dbl> <dbl>    <dbl> <dbl>  <dbl>   <dbl>       <dbl>
##  1 naive  Trainin~   8.02    9.49  8.06  0.776   0.779 0.252  -0.375        NA
##  2 naive  Test Set 31.9     36.9  31.9   2.45    2.45  0.995   0.563         4.14
##  3 snaive Trainin~ 32.0     33.1  32.0   3.06    3.06  1       0.930        NA
##  4 snaive Test Set 45.1     47.8  45.1   3.48    3.48  1.41    0.615         5.24
##  5 model3 Trainin~ NA        8.40  7.96 NA       0.773 0.975  NA            NA
##  6 model3 Test Set NA       39.5  35.2  NA       2.70  9.59   NA            NA
##  7 SES    Trainin~  7.87     9.41  7.91  0.762   0.765 0.247  -0.331        NA
##  8 SES    Test Set 31.9     36.9  31.9   2.45    2.45  0.996   0.563         4.14
##  9 HA A   Trainin~ -0.0795  2.26  1.76 -0.00565 0.167 0.0549 -0.0428       NA
## 10 HA A   Test Set  5.36     6.28  5.36  0.412   0.412 0.167   0.505         0.698
## # i 20 more rows
## # A tibble: 3 x 10
##   Model Set          ME  RMSE   MAE   MPE  MAPE  MASE   ACF1 `Theil's U`
##   <chr> <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>       <dbl>
## 1 HA B  Test Set  3.50  4.47  3.50  0.268 0.268 0.109  0.416       0.493
## 2 SCUM  Test Set NA     5.28  4.43 NA     0.340 0.607 NA          NA
## 3 MLP   Test Set  4.29  5.49  4.29  0.329 0.329 0.134  0.436       0.605
```

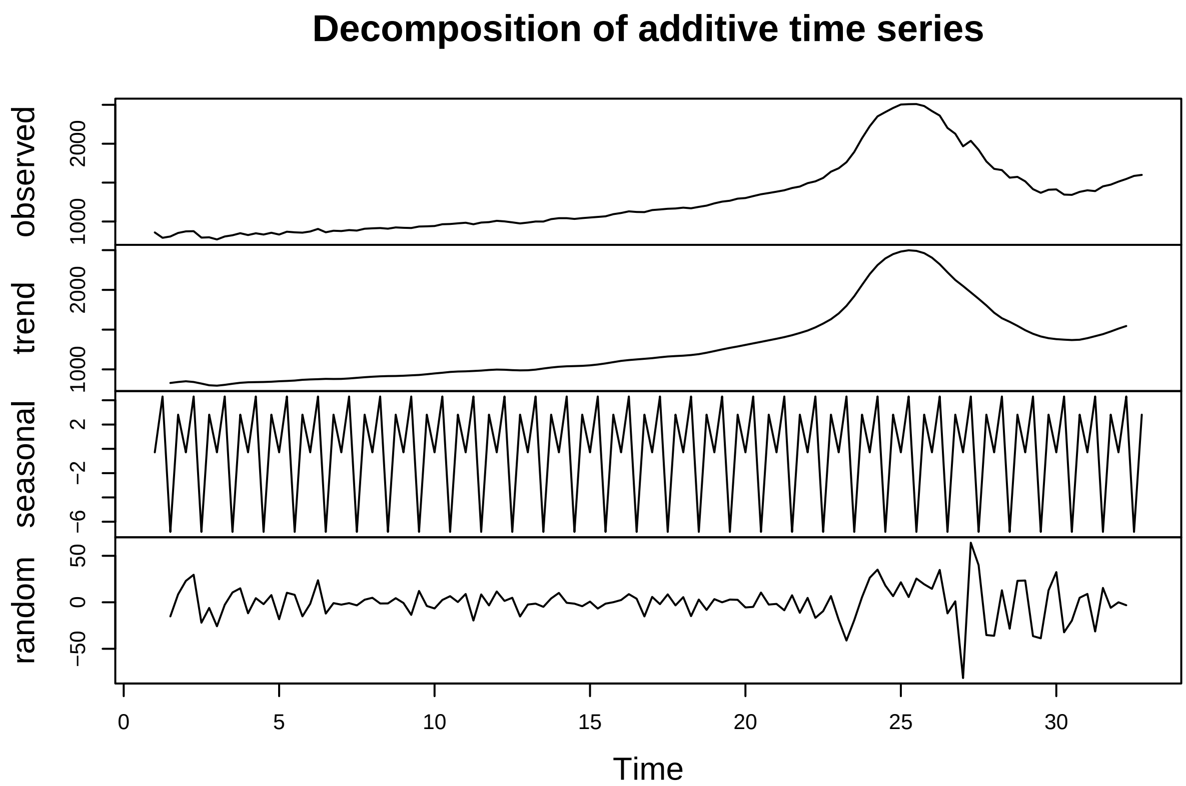


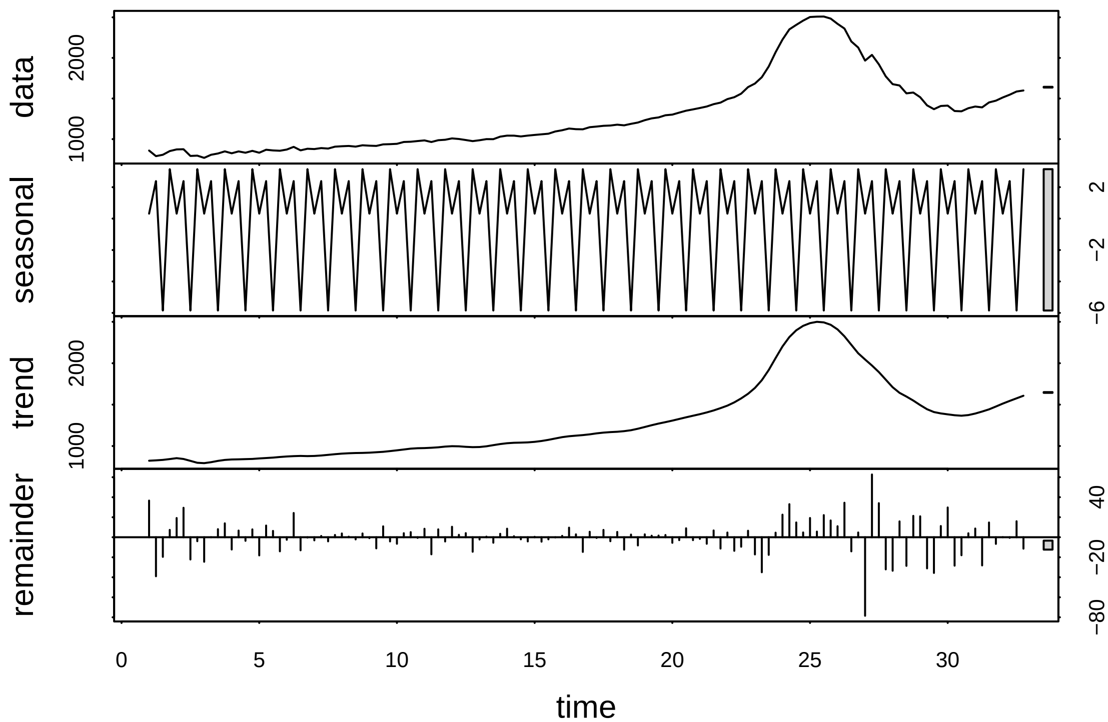
Dataset #10- Evaluate data set Q200:

Task 1- Select dataset Q-200 and preprocess it + Plot the data

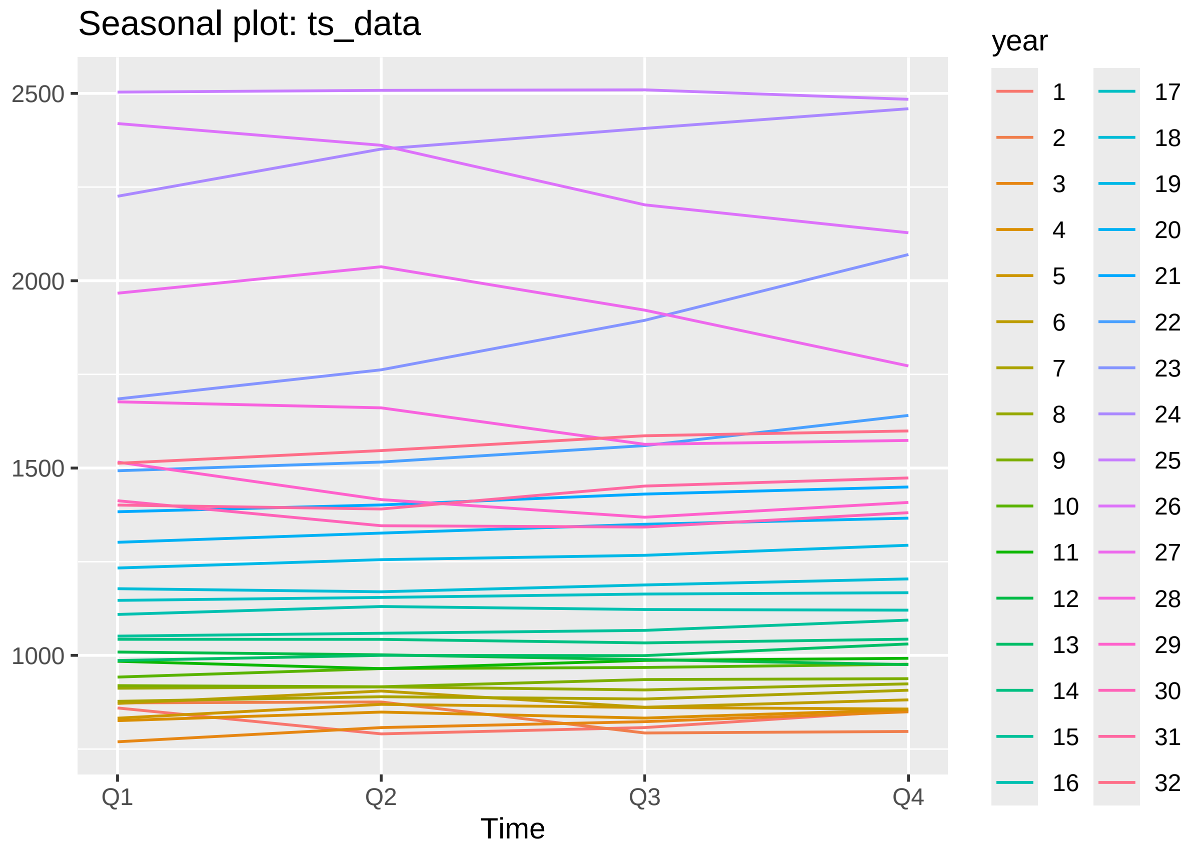
```
data_set_to_load <- c("Data_set_200")

output_Second_Read_Data <- Madpis_Second_read_data(
   Madpis_Initial_read_data_output = output_Inital_Read_Data,
   data_set_to_load = data_set_to_load,
   plot_dataset = TRUE,
   plot_decompose = TRUE, plot_seasonal = TRUE)
```

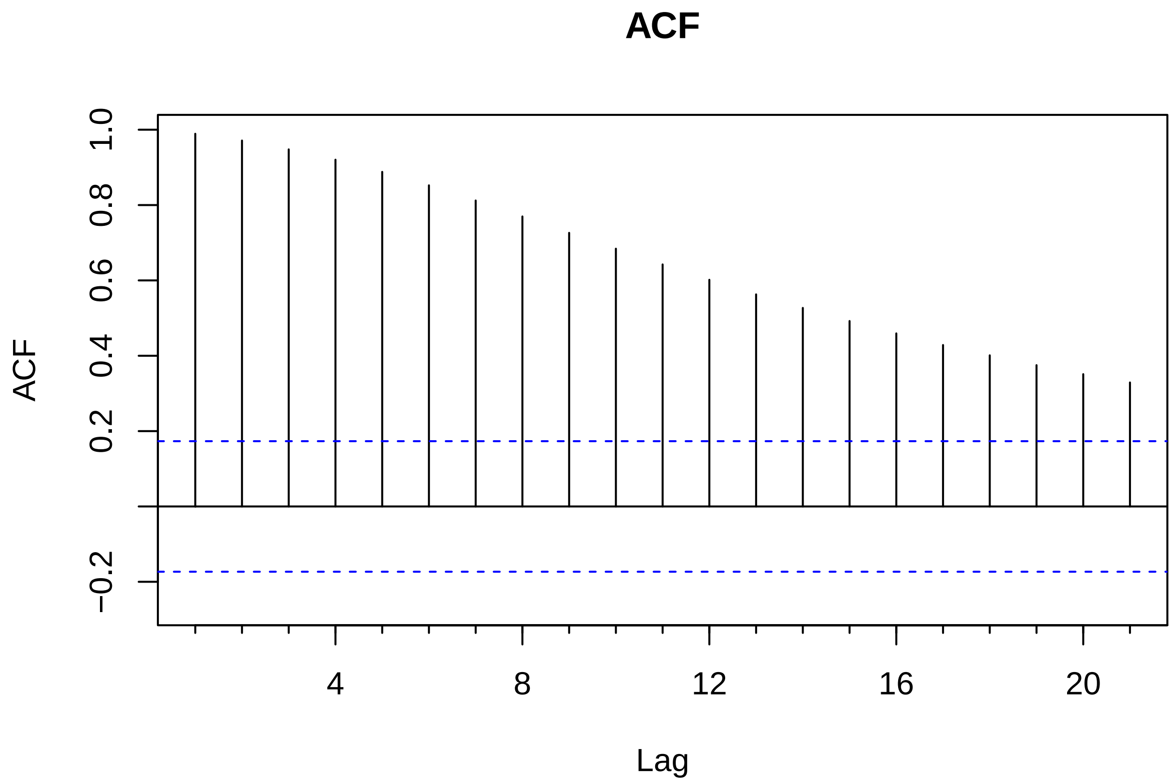


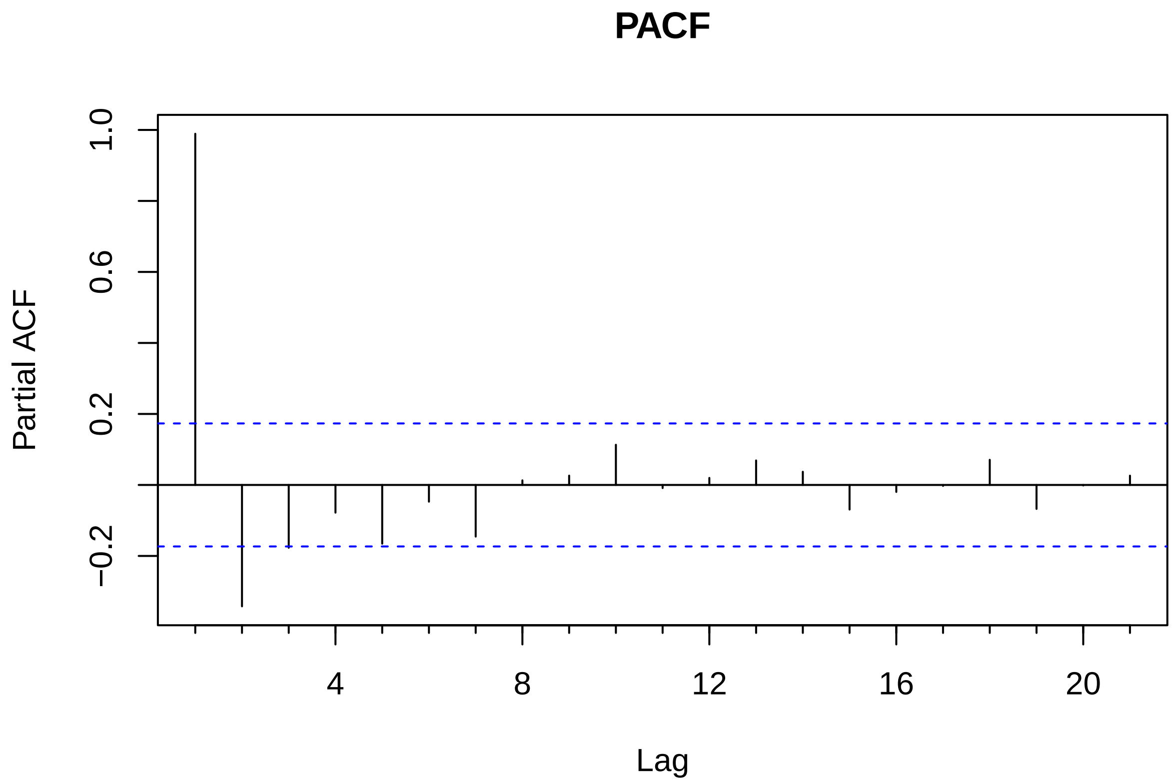

Decomposition of additive time series
observed
trend
seasonal
random
1000
2000
2
−2
−6
50
0
−50
0
5
10
15
20
25
30
Time

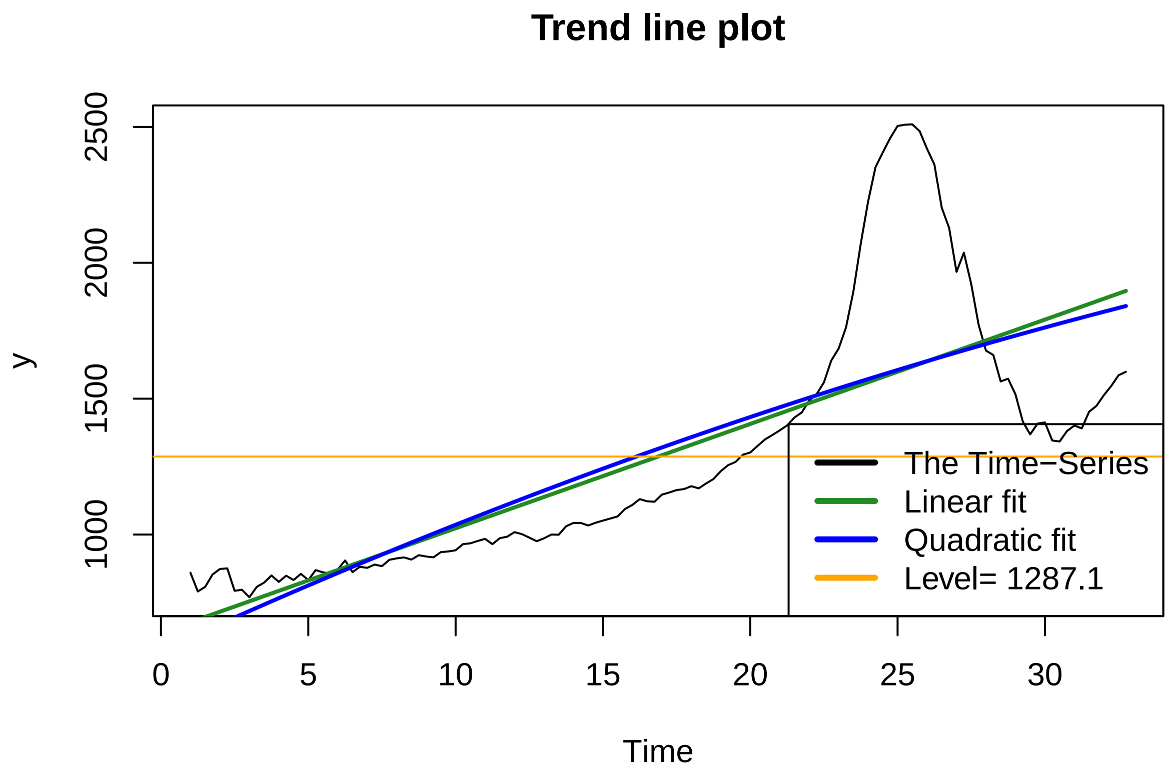


data
seasonal
trend
remainder
1000
2000
2
−2
−6
1000
2000
40
−20
−80
0
5
10
15
20
25
30
time


Seasonal plot: ts_data
2500
2000
1500
1000
Q1
Q2
Q3
Q4
Time
year
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32


ACF
1.0
0.8
0.6
0.4
0.2
−0.2
ACF
4
8
12
16
20
Lag


PACF
Partial ACF
1.0
0.6
0.2
−0.2
4
8
12
16
20
Lag




```
#tsibble_data <- output_Second_Read_Data$tsibble_quarterly_train
```

Notes on the time-series:

- We can see that there is **32 years of quarterly data**
- From the **Time plot** we can see that for the first 25 years there is an increase trend, afterwords - a decrease trend, at the end we can see a little increse trend.
- From the **Decomposition graph** and the **seasonality graph** we can see there is seasonality in the series
- From the **ACF** and **Pacf** plots we can understand there **is an autocorrelation in the series**, the **Pacf** plot reveals that most of that autocorrelation originate from the 1st and the 2nd lag autocorrelation.

Task 2- Define the models + fit them + get the performance and plot

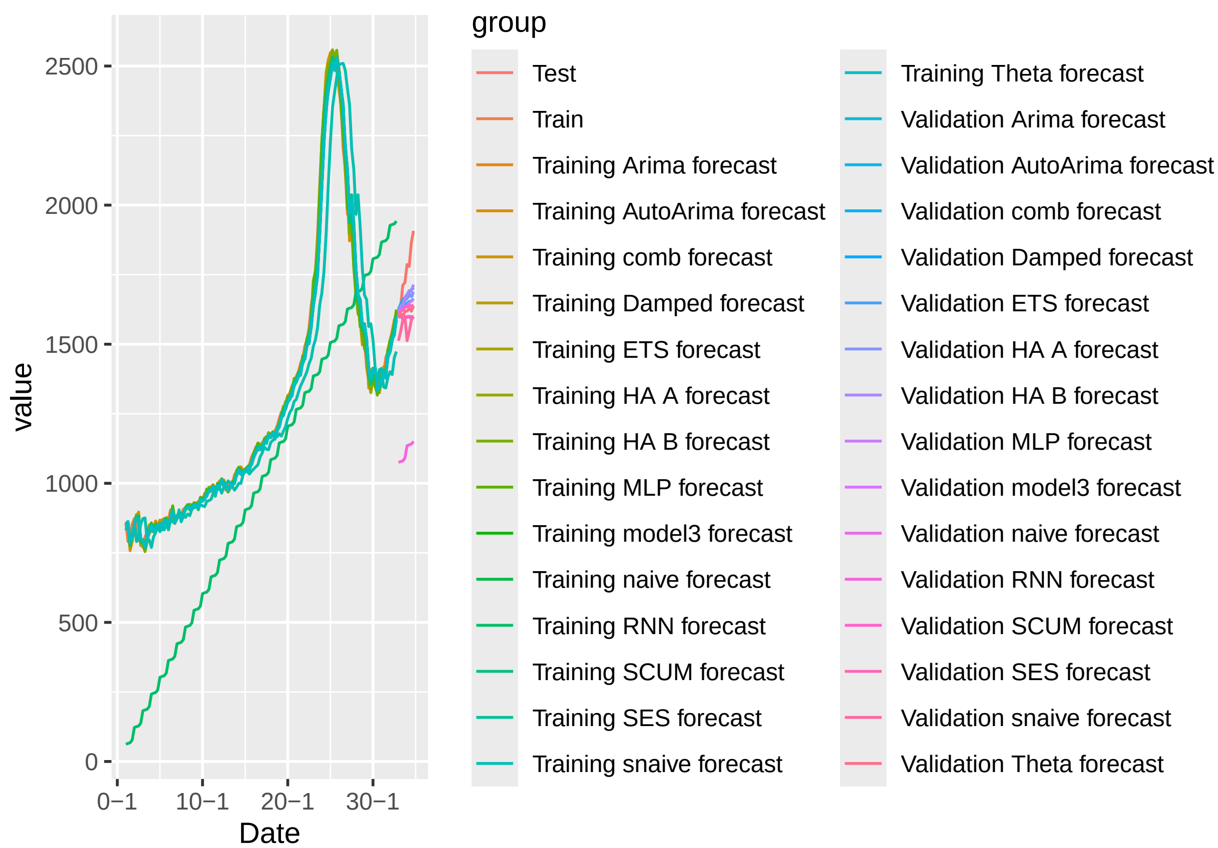
```
#Madpis_Second_read_data_output <- output_Second_Read_Data

Madpis_Third_output <- Madpis_Third_models(
   Madpis_Second_read_data_output = output_Second_Read_Data,
   include_RNN = T,
   plot_all_models = TRUE,
```

```
  print_accuracy_all_models = TRUE,
  show_top_3_models = TRUE,
  plot_MLP = FALSE, write_datasets_for_RNN = F,
  dir_name_save_dataset = "datasets_for_RNN"
)
```

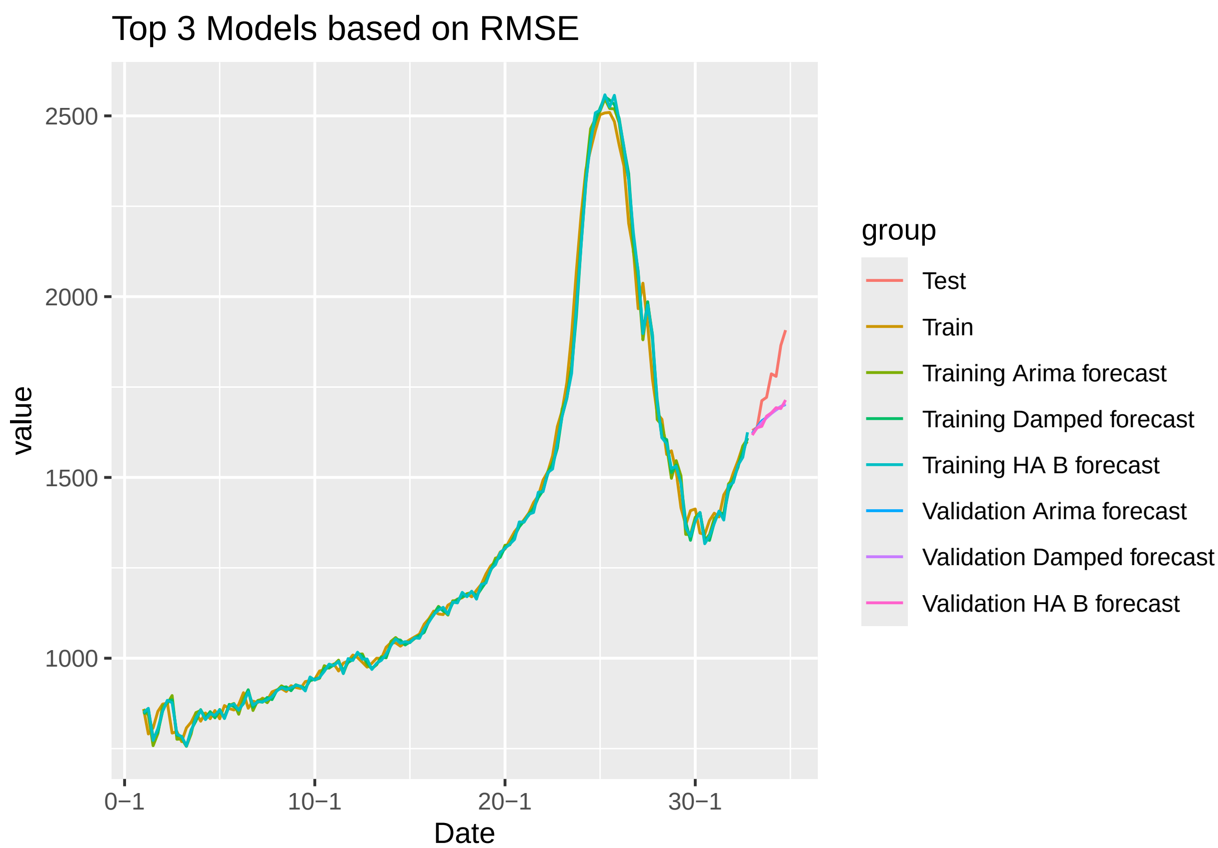
```
## New names:
## Rows: 128 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## New names:
## Rows: 8 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (8): ...1, Unnamed: 0, times, seasons_Q1, seasons_Q2, seasons_Q3, season...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
## [1] "KPSS Test p-value is lower than 0.05 Thus we need to reject\nH0: The data is **NOT Stationary**
```

```
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
## Joining with `by = join_by(Date)`
```




```
## # A tibble: 30 x 10
##    Model  Set            ME  RMSE   MAE   MPE  MAPE   MASE   ACF1 `Theil's U`
##    <chr>  <chr>       <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>       <dbl>
##  1 naive  Trainin set  5.82  49.5  32.7 0.440  2.33  0.322  0.547       NA
##  2 naive  Test Set    156.  182.  156.   8.64  8.64  1.54   0.581        3.59
##  3 snaive Trainin set  23.7  161.  102.   1.66  6.45  1      0.952       NA
##  4 snaive Test Set    194.  211.  194.  10.8  10.8   1.91   0.554        4.12
##  5 model3 Trainin set  NA    49.0  32.3 NA     2.28  0.939 NA           NA
##  6 model3 Test Set     NA   185.  159.  NA     8.80 20.6   NA           NA
##  7 SES    Trainin set  5.77  49.3  32.5 0.435  2.32  0.319  0.548       NA
##  8 SES    Test Set    156.  182.  156.   8.64  8.64  1.54   0.581        3.59
##  9 HA A   Trainin set  2.38  38.1  26.4 0.234  1.97  0.259  0.179       NA
## 10 HA A   Test Set    100.  124.  100.   5.52  5.52  0.987  0.530        2.43
## # i 20 more rows
## # A tibble: 3 x 10
##   Model  Set         ME  RMSE   MAE   MPE  MAPE  MASE  ACF1 `Theil's U`
##   <chr>  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>       <dbl>
## 1 HA B   Test Set  87.4  109.  87.4  4.80  4.80 0.860 0.498        2.15
## 2 Arima  Test Set  86.8  110.  86.9  4.76  4.76 0.855 0.545        2.17
## 3 Damped Test Set  88.0  111.  88.0  4.83  4.83 0.866 0.535        2.17
```



Task 3 - Article

As mentioned above, in this project, we chose the article: "A simple Combination Of Univariate models" by Fotios Petropoulos and Ivan Svetunkov.

3.1 A brief summary which describes the intent of the article. This can be a few sentences. It should not be a straight copying of a few sentences from the abstract or the article

The article present the method Fotios and Ivan used in the M4 competition. In short, they chose to **combine 4 different models** - ETS, CES, ARIMA and Theta and called the combined model - SCUM (Simple Combination of Univariate Models). In the article they elaborate on each one of the models and describe the way they combined them - using median on the outputs of the models. They explain that the performance of the combined model is better than the individual performance of each model (that comprise the big combined model), this is due to "the increase robustness of the final forecasts and the decrease in the risk of having a completely incorrect forecast". Moreover they used models that are diverse - each model is capable capturing something that the other models can't.

In addition they explain on the **computational time** issue and claim that one of the benefits of the SCUM model compared to other Machine learning models is the computational time, which in many cases is critical for a business that needs to get many forecasts within time constraints.

We have implemented their modeling approach here and used it to create forecasts for the 10 datasets we chose in Task 1.

The approach: The SCUM model takes median of the point forecasts and prediction intervals of four models:

1. **Exponential Smoothing** (using ETS)
2. **Complex exponential smoothing** (CES - produces non-linear trends with a slope that depends on the data characteristics. There are both non-seasonal and seasonal versions of this model. The former allows one to slide between the level and the trend without the need for a dichotomic selection of components that is appropriate for the time series. The latter captures the type of seasonality (additive or multiplicative) and produces the appropriate forecasts, once again without the need to switch between the two options. The combination of these two models allows us to capture complex dynamics in the data. (using auto.ces())
3. **Automatic autoregressive integrated moving average** (=using auto.arima())
4. **Dynamic optimized theta** (DOTM) (using dotm())

3.2 A discussion of which aspect of the article you decided to explore, and why:

We decided to implement the modeling approach presented in the article, namely to reproduce the SCUM method and use it with the time-series we selected to forecast them and preserve the SCUM method performance. The reason we decided to do so is both because this method is not too complicated and we can mimic it as well as in this project we are asked to do so. Also the idea of using an ensemble model is familiar to us from ML, using RandomForst, XGBoost, etc.

3.3 The results of your exploration - If it's a forecasting method - then the forecasts, a discussion of the pros and cons of the method,etc:

We presented in the code the prediction of the SCUM model and its performance. For convenience we also present them here:

```
### The predictions of the SCUM model on the training period: (column value)
#training_pred_tib_SCUM
Madpis_Third_output$training_pred_tib_all %>% filter(model == "SCUM" )
```

```
## # A tibble: 128 x 4
##    Date      value group                  model
##    <yearqtr> <dbl> <chr>                  <chr>
##  1 1 Q1       859. Training SCUM forecast SCUM
##  2 1 Q2       858. Training SCUM forecast SCUM
##  3 1 Q3       782. Training SCUM forecast SCUM
##  4 1 Q4       804. Training SCUM forecast SCUM
##  5 2 Q1       852. Training SCUM forecast SCUM
##  6 2 Q2       881. Training SCUM forecast SCUM
##  7 2 Q3       878. Training SCUM forecast SCUM
##  8 2 Q4       797. Training SCUM forecast SCUM
##  9 3 Q1       791. Training SCUM forecast SCUM
## 10 3 Q2       765. Training SCUM forecast SCUM
## # i 118 more rows
```

```
### The predictions of the SCUM model on the Validation period: (column value)
#validation_pred_tib_SCUM
Madpis_Third_output$Validation_pred_tib_all %>% filter(model == "SCUM" ) %>% select(c("Date","value", "
```

```
## # A tibble: 8 x 4
##   Date      value group                   model
##   <yearqtr> <dbl> <chr>                   <chr>
## 1 33 Q1     1607. Validation SCUM forecast SCUM
## 2 33 Q2     1619. Validation SCUM forecast SCUM
## 3 33 Q3     1623. Validation SCUM forecast SCUM
## 4 33 Q4     1634. Validation SCUM forecast SCUM
## 5 34 Q1     1635. Validation SCUM forecast SCUM
## 6 34 Q2     1641. Validation SCUM forecast SCUM
## 7 34 Q3     1637. Validation SCUM forecast SCUM
## 8 34 Q4     1641. Validation SCUM forecast SCUM
```

```
### The accuracy tibble of the SCUM model - containing the performance of the model on both the trainin
#accuracy_SCUM
Madpis_Third_output$accuracy_all %>% filter(Model == "SCUM" )
```

```
## # A tibble: 2 x 10
##   Model Set             ME  RMSE   MAE   MPE  MAPE   MASE  ACF1 `Theil's U`
##   <chr> <chr>        <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>       <dbl>
## 1 SCUM  Trainin set     NA  41.7  27.7    NA  2.04  0.820    NA          NA
## 2 SCUM  Test Set        NA 151.  125.     NA  6.92 21.8      NA          NA
```

Further comparison of the results: we can see that in terms of prediction power, when looking at the RMSE,at least for the last data set evaluated, the SCUM model ranks 9th from 15 in total, which is worse than ARIMA, and ETS that are both part of the ensemble. This might suggest that using one of them is better than SCUM (after testing which one is better).

On a personal note, we believe that the SCUM model's overall performance (averaged on all the published datasets) might be better, however examining the performance of the model on some specific datasets might reflect a wrong deceiving picture.

```
Madpis_Third_output$ accuracy_all %>%
     filter(Set == "Test Set" ) %>%
    arrange(RMSE) %>%
    mutate(rank = dense_rank(RMSE)) %>%
    select(rank, everything())
```

```
## # A tibble: 15 x 11
##     rank Model     Set      ME  RMSE   MAE   MPE  MAPE   MASE   ACF1 `Theil's U`
##    <int> <chr>     <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>       <dbl>
##  1     1 HA B      Test~  87.4  109.  87.4  4.80  4.80  0.860  0.498        2.15
##  2     2 Arima     Test~  86.8  110.  86.9  4.76  4.76  0.855  0.545        2.17
##  3     3 Damped    Test~  88.0  111.  88.0  4.83  4.83  0.866  0.535        2.17
##  4     3 ETS       Test~  88.0  111.  88.0  4.83  4.83  0.866  0.535        2.17
##  5     4 AutoArima Test~  92.8  122.  94.5  5.07  5.17  0.930  0.562        2.39
##  6     5 HA A      Test~ 100.   124. 100.   5.52  5.52  0.987  0.530        2.43
##  7     6 comb      Test~  NA    138. 115.  NA     6.33 15.4   NA           NA
##  8     7 MLP       Test~ 118.   141. 118.   6.48  6.48  1.16   0.561        2.78
##  9     8 SCUM      Test~  NA    151. 125.  NA     6.92 21.8   NA           NA
## 10     9 Theta     Test~ 138.   162. 138.   7.66  7.66  1.36   0.546        3.20
## 11    10 naive     Test~ 156.   182. 156.   8.64  8.64  1.54   0.581        3.59
## 12    11 SES       Test~ 156.   182. 156.   8.64  8.64  1.54   0.581        3.59
## 13    12 model3    Test~  NA    185. 159.  NA     8.80 20.6   NA           NA
## 14    13 snaive    Test~ 194.   211. 194.  10.8  10.8   1.91   0.554        4.12
## 15    14 RNN       Test~  NA    647. 644.  NA    36.6  59.7   NA           NA
```

F - U - N

We love time series forecasting

Roy Madpis & Michael Kobaivanov