**Statistics 153 Midterm 2**

**My Dinh**

**Hyunouk(Andy) Ko**

1. **Introduction:**

The goal of the project is to predict 2 years of google trends data given that we only have information from 2004 to 2014. We base our modelling approach on the assumption that the data, after appropriate transformations, can be modelled as ARIMA model we learned in class. We consider trend, seasonality, and noise, but avoid parametric fitting

1. **Data:**

Five time series data each of length 525 that represent google trends queries for terms from the first week of January, 2004 to the second week of January were provided to build prediction models on. Each time series is a mixture of both negative and positive values with no missing values. The datasets come with no description, thus the analysis and prediction model are mainly based on data plots and time series knowledge.

1. **Process:**

In order to predict the next two years of the fith time series, we perform the following procedures:

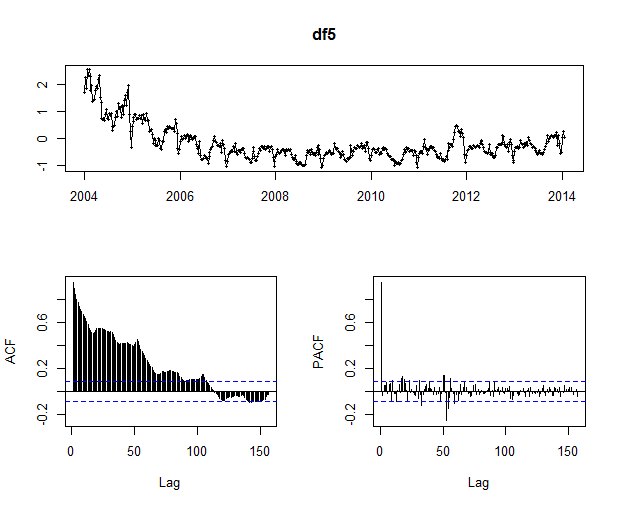
1. Data Exploration and Transformation
2. Model Selection
3. Model Diagnostic
4. Forecast

While the last step decides the outcome of the project, we do some research for step 1 and mainly focus on step 2 and 3.

1. Data Exploration and Transformation:

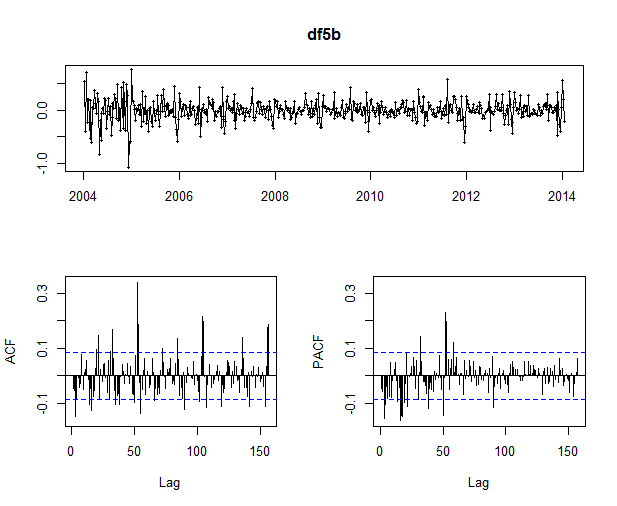
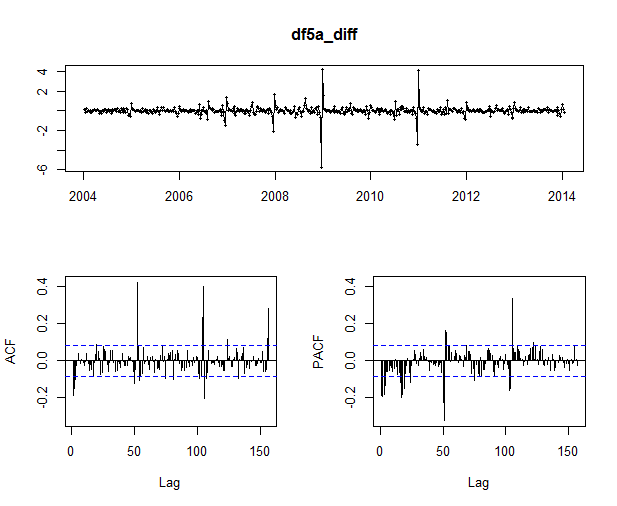
The goal of this step is to decide whether we should go with simple mean model or continue with more complex covariance models like ARIMA.

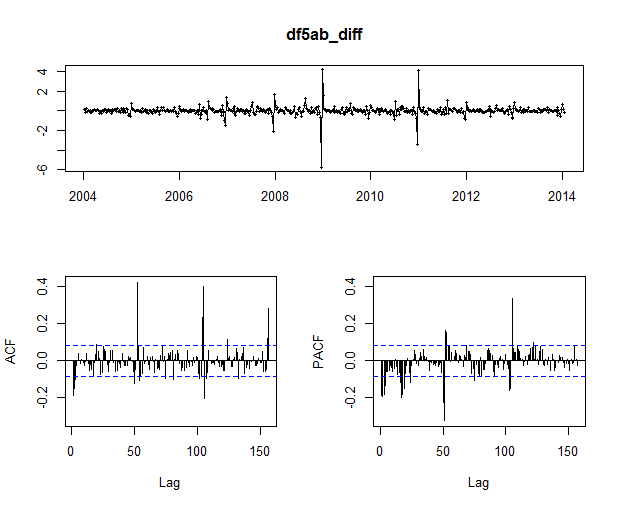
When looking at the data plot, we observe an upward trending, seasonality traits, and increasing variance over time. Lags at both ACF and PACF slowly decay, thus this time series is not stationary.



(Original Data)

Since the data seems to contain trend, we decide to use differencing method to remove this effect. The orders of differencing corresponds to the order of polynomial trend in the data set. In this dataset, only first order difference is needed to stabilize the mean. However, the unusual trait as long as high level shifts at some certain points in both time plot and correlogram suggest another transformation along with differencing.





df5a\_diff: log transform

df5b : difference

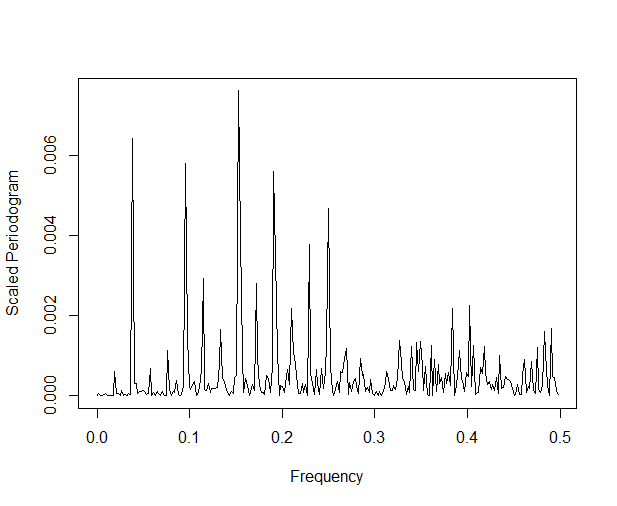
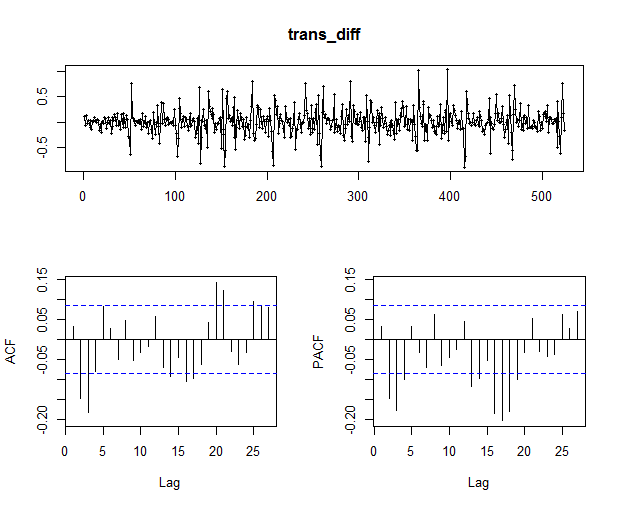
df5ab: difference on log transform

We try multiple transformations taking into consideration that the data contains both negative and positive values: translation to positive+ log transform + difference, translation + square root + difference, arcsinh + difference, translation + Box-Cox Transformation + difference, and Yeo-Johnson transformation + difference, which is an extended version of Box-Cox. Both Box-Cox and Yeo-Johnson belongs to family of power transformation and are popular used to attain the normality of the distributions. While BoxCox only performs on positive values, Yeo-Johnson can take in both negative and positive values. We notice that those power transformation help stabilize the variance of the time series, thus making it easier to attain stationary. These two transformation will be used to fit models that will be discussed in the next sections.

We examine each transformation by looking at their time plot, ACF, and PACF as well as perform statistical test (KPSS, Lijung-Box test) to check if the residuals are white noise. While other transformation still exhibits irregular patterns, the residuals of the last two transformation seems to be close to white noise. However, the choice of , which is power of the transformation, needs to be chosen carefully to attain the goal. Based on our research, we decide to choose for Box-Cox and Yeo-Johnson transformation using the following step:

* Use built-in search function to approximate the optimal that maximizes the log likelihood
* Get the 95% CI interval for estimation. If the optimal is not convenient value (for example 0.41), choose another value that is within the confidence interval that is more convenient (for example 0.5)

While the first rule is obvious, we come up with second rule after building the model. We realize that using with optimal value like 0.41 can induce approximated error for each data point when transforming and inverting data. After building model on the optimal lambda, we retest it with other convenient and achieve lower mean squared error.



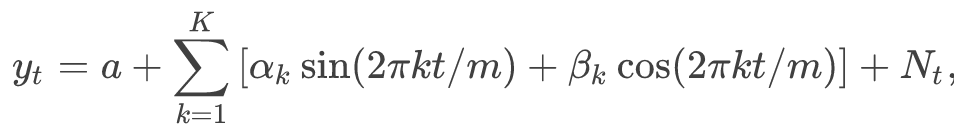
Box-Cox + Differenced data (Periodogram)

2. Model Selection and Prediction:

Even with Box-Cox and Yeo-Johnson transformations with differencing, the ACF and PACF plots still display more than 5% significant lags and seasonal pattern. Thus it is better that we consider SARIMA model instead of mean model for the fifth dataset. The goal now is to estimate six parameters p, d, q for the nonseasonal component and P, D, Q for seasonal component.

We originally start with `auto.arima()` function to come up with initial guess for our model. By default, `auto.arima` searches for all possible models and returns the one with lowest BIC score without any seasonal component. To force for seasonality, we set max.D =1. The function returns a model with period of 52, which also reconciles with our guess from the ACF and PACF residual plots as well as the weekly data. We can tune the parameters and look for the lowest BIC, AIC, and Cross Validation error for the final model. However, when plotting our prediction with the original time series, the prediction does not seem to address seasonality.

Taking a closer look at our data, we notice that the data are collected over ten years in which we have some leap years. In other word, some years will have 53 weeks instead of 52 weeks, thus fitting a model with a fixed period of 52 doesn’t do well for weekly data. After doing research, we decide to use Fourier regressors to address seasonality, which is better than naive approximation of seasonality=52. The model is given by the formula below:



(1)

Where K is chosen to minimize AIC and Nt is the ARIMA model.

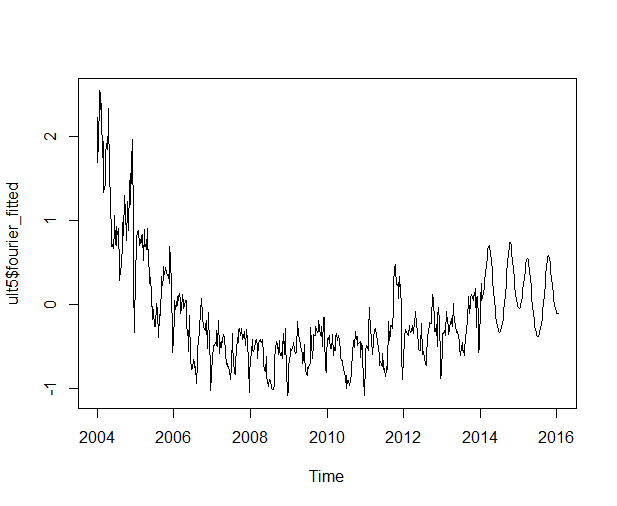
We test out this model and both its AIC an BIC score improves significantly. The plot of prediction also displays the seasonality trait of the data. We thus switch from SARIMA to ARIMA with error regression using Fourier approach and consider it as our first model as described below.

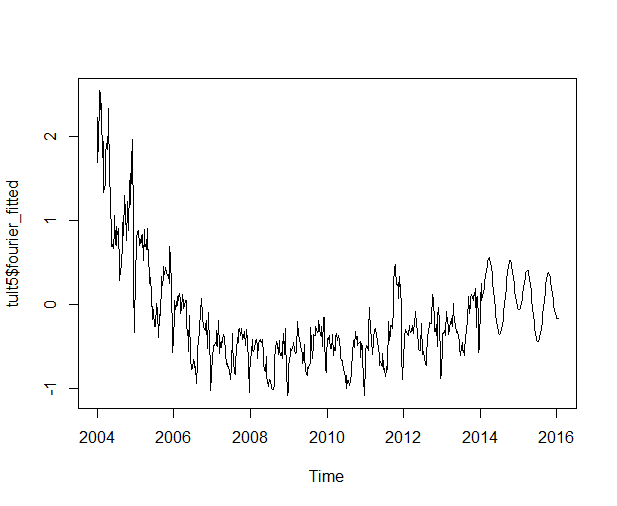
**Model 1: Fourier-regressed ARIMA Model**

We perform the following steps to select the best Fourier model:

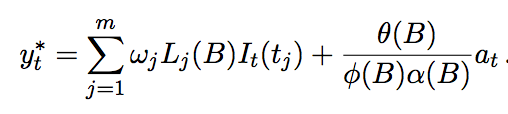
* Use `auto.arima` combined with Fourier regression to represent seasonal component as a starting guess. The number of linear terms of sin and cos in equation (1), K will be determined by the lowest AIC score. Attain p, d, q and coefficient component.
* Perform a linear search for p and q by increase and decrease each side by 1, refit it with `Arima` and fourier expansion where K is determined by the same rule as step 1.
* Re-evaluate model with cross validation.

We use this model for both transformed time series version that we come up with from part 1 for final comparison.

Though this model do a good job of seasonal representation, we also notice some outliers in the data set that cause unusual spikes. For example spikes around 2011 to 2012. This suggests an existence of additive or level shift outlier. This observations motivate us to seek for another model that can handle those situations. 

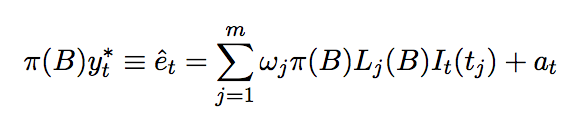


**Model 2 : Outlier-adjusted ARIMA Model with Fourier expansion:**

Since the first model prove to handle seasonality well, we build the second third model from the first model platform. Luckily, R already has a package that can detect outliers and forecast with ARIMA. 

Time series {Yt} is expressed as:

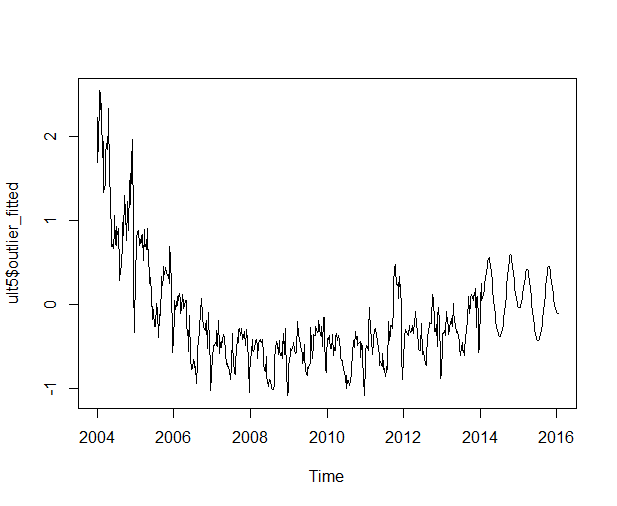
With estimated residuals:



The algorithm contains three stages:

* Fit the chosen model, use t-statistics to test for existence of outliers.
* Remove outliers with reasonable values. Refit ARIMA model. The significance of outliers may assert in new fitted model
* Reiterate step 1 and 2 until no outliers is detected.

We will not discuss further details on this processed because it goes beyond the scope of the materials we have learned. Moreover, the documentation by Javier L´opez-de-Lacalle for `tsoutliers` package delivers a detailed explanation of the algorithms. `tso` function can be incorporated with `auto.arima` or `arima` to fit the data. The hardest part of this model is perhap the trade off between speed of computation and model selection. The document suggests to use`auto.arima` to search for the best model; however, we notice it is very computationally expensive. We do not want to restrict our model with `arima` either since it does not allow any drift term if d > 0. Therefore we decide to modify `auto.arima` by setting the maximum order of p, d, q, and K for fourier term in the argument same as the final version of model 1. Since higher order of parameter will have a high chance of overfit data, thus we believe the bound for p, d, q and K is reasonable and help speed up the computation.



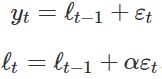
**Model 3 : Seasonal & Trend Decomposition + ETS(Error, Trend, Seasonality) Model**

Though we have not learn about this approach, we still include this model based on our research. This method uses STL decomposition using Loess on the data to remove any seasonality in the data. It assumes that the data could be written as:



and applies time series method ETS(Error, Trend, and Seasonality) on the seasonally-adjusted data. ETS is an exponential smoothing state space model often expressed with three parameters that describe the , for example (A,N,M), which indicates error is additive, trend is neither additive nor multiplicative, and seasonality is multiplicative. Possible choices of are N, A, A\_d, M, M\_d, which each stand for None, Additive, Additve Damped, Multiplicative, and Multiplicative Damped. “stlf” function which is part of the “forecast” package, takes in the original data, applies Box-Cox transformation based on lambda provided, decomposes using STL decomposition, models with ETS method, predicts on the transformed data, reseasonalizes, and returns final forecast results by inverting the same Box-Cox transformation used at the beginning.

For an example of how ETS is used, consider the case of ETS(A,N,N). Prediction is based on the model below:



Where

 and 

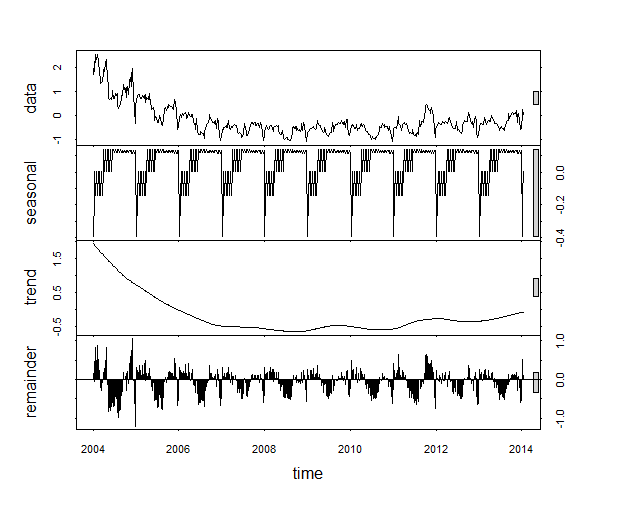
y\_t is the process ETS takes in and it recursively calculates the predictions.

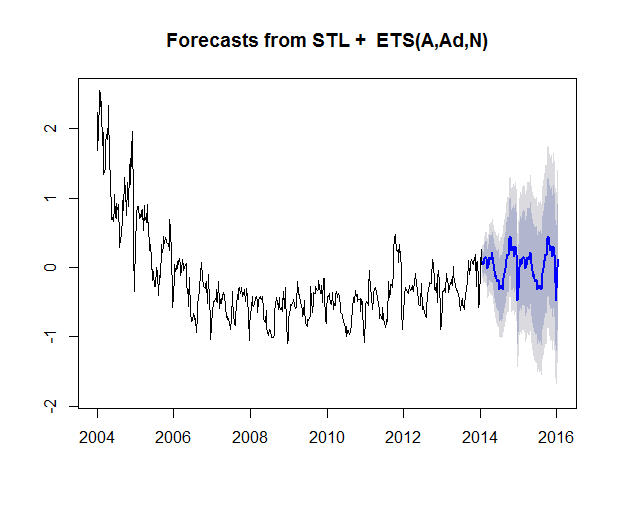
Note the error is additive as the A in the first parameter indicates.

What stlf does is that it searches for the best ETS model parameters to describe the (transformed and seasonally adjusted) data and estimates appropriate parameters according to information criterion specified(uses AIC as default).

Below are plots of the results obtained from apply stlf on q5\_train:

(STL Decomposition of the Data)



(Predicted Values using STL Forecast

3. Model diagnostic:

To determine which model would deliver the best prediction, we implement forward chaining cross values. Noticed that AIC and criteria is already taken into consideration when choosing initial guess for p, d, q and best K in model 1.

We set the minimum number of data as training set to be 161, and fold size of (525-161)/7 = 52. In other words, the first iteration takes the first 161 data and predicts the next 52, second iteration takes the first (161+52) data and predicts the next 52, and so on until the sixth iteration that takes the first 473 data and predicts the last 52. “cross\_validate” function runs “fourier\_model” with non-seasonal parameters for each of the 9 models in consideration. In other words, for each size of training set in CV, we let “fourier\_model” choose the best K in AIC, only keeping the non-seasonal order constant. Then, MSE of predictions in the 6 iterations are summed and returned. We compare these sums of MSE’s for each model and choose the final non-seasonal order that minimizes this quantity. With this final non-seasonal order determined, we finally fit the model on the entire data and run our final “fourier\_model” and make forecasts for the next 104 weeks. This of course, is the prediction on the transformed data so, we invert back the predictions using the inverse of whatever transformation that was used and produce the final predictions. Original data along with predicted data is plotted above.

1. **Results and Conclusion:**

We incorporate all of the steps that explained above into a wrapper function `ultimate`. Below is the summary of our `ultimate` function:

1. Use `auto.arima` in ‘fourier\_model’ to choose non-seasonal parameter and K that minimize AIC.
2. Use K to general model list with parameters (p, d, q) increased and decreased by 1.
3. Use cross validation(using 7-fold) on those parameters by manually testing all possible K’s to use for fourier expansion again. Repeat the process of CV for all 3 models explained in previous section. Return the list of all mean square error (MSE).
4. The candidate for final model will give the lowest MSEs from step 3.
5. Forecast based on that model. Reevaluate the model based on plots and interpretation of model and data.

With three different models and two type of transformed data, we have total of 5 predictions with two prediction for model 1, two prediction for model 2, and the last one for model 3. While it’s natural to go with the model with the lowest MSE, the error returned from each model doesn’t seem to be much different. Thus we add step 5 to choose the final model when plotting our prediction with original data. By examining the plots of the predictions, we consider the model that tend to give less variation as well as pick up the trending of the data. Moreover, we consider the interpretation of our model. Since this time series is generated by google search term, we try to guess the overall trending by reconciling with the unusual patterns in the data set. For this time, we decide to choose model 3 for our final prediction.

Q1\_train : Fourier-regressed ARIMA Model with Yeo-Johnson Transformation

Q2\_train : Fourier-regressed ARIMA Model with Box-Cox Transformation

Q3\_train : STL Decomposition + ETS Model with Box-Cox Transformation

Q4\_train : Fourier-regressed ARIMA Model with Yeo-Johnson Transformation

Q5\_train : STL Decomposition + ETS Model with Box-Cox Transformation

1. **Appendix:**

**# STATISTICS153 Midterm2**

**# Hyunouk(Andy) Ko & My Dinh**

**#==============================================================**

**# set directory and import packages**

**setwd("C:/Users/Andy Hyunouk Ko/Desktop/School/Probability&Statistics/Stat153/project")**

**library(caret)**

**library(tseries)**

**library(forecast)**

**library(stats)**

**library(car)**

**library(VGAM)**

**library(MASS)**

**library(tsoutliers)**

**#==============================================================**

**# Write an initial helper functions**

**# since we're dealing with long seasonality data, we consider Fourier transformation in fitting ARIMA model.**

**# reference: http://robjhyndman.com/hyndsight/longseasonality/**

**# K in the post is computed by choosing K that minimizes AICc.**

**# fourier\_model function gives the best fourier-regressed ARIMA model and K(#of fourier terms) in terms of AIC**

**fourier\_model = function (dat, method = c("auto", "manual"), non\_sea\_comp, drift = FALSE){**

**## Inputs**

**# dat : ts object**

**# method : if auto, it generates best BIC-yiedling nonseasonal parameters and if manual, user must specify non-seasonal component**

**# nonsea\_comp : must be specified as (AR, difference, MA) parameter form if method = 'manual'**

**# drift : boolean to include drift term or not**

**model = list(aic=Inf)**

**method = match.arg(method)**

**optim\_K = 0**

**for(i in 1:25)**

**{**

**if (method == "auto"){**

**fit = auto.arima(dat, xreg=fourier(dat, K = i), seasonal = FALSE)**

**}else fit = Arima(dat,order = non\_sea\_comp, xreg=fourier(dat, K = i), include.drift = drift)**

**if(fit$aic < model$aic){**

**model = fit**

**optim\_K = i**

**}else break;**

**}**

**return (list(model=model, K=optim\_K))**

**}**

**#==============================================================**

**## Data Exploration and Transformations for Stationarity**

**# Step 1: Identify unusual observations and transform data**

**df1 = read.csv("q1\_train.csv")**

**df1 = df1[,2]**

**df1 = ts(df1, freq = 365.25/7, start = 2004)**

**# detect and replace outlier.**

**df1\_out = tsclean(df1)**

**# plot dataset:**

**plot(df1, type = "l") #doesn't look stationary**

**acf(df1)**

**#log transform by the min:**

**df1a = df1**

**df1\_min = min(df1a)**

**df1a= log(df1a - df1\_min + 0.001)**

**df1a\_diff = diff(df1a)**

**tsdisplay(df1a\_diff)**

**# doesn't seem to work**

**# try difference on the original data:**

**df1b = df1**

**df1b= diff(df1b)**

**tsdisplay(df1b)**

**#try difference on the log transform:**

**df1ab\_diff = diff(df1a)**

**tsdisplay(df1ab\_diff)# doesn't seem to change anything**

**#try Yeo-Johnson transformation**

**df1e = read.csv("q1\_train.csv")**

**yw\_param = preProcess(df1e, method = c("YeoJohnson"))**

**yw\_param #search for optimal lambda, return #0.14 but 0.15 is still within CI so choose 0.15 for convenience**

**trans = yjPower(df1, 0.15, jacobian.adjusted = FALSE) #stabilize variance**

**trans\_diff = diff(trans) #stabilize mean**

**tsdisplay(trans\_diff) #pacf: there are apparently 3 clusters at 24, 48, 52,**

**#There are peak at 52, 106, 156 in acf.**

**# it seems that yeo\_johnson transformation didn't do well on outlier version**

**# Now check if we have stationary process**

**# Check using Ljung- Box test:**

**Box.test(trans\_diff) # p-value = 0.005694 indicates that the series is stationary**

**adf.test(trans\_diff, alternative = "stationary") #p-value = 0.01(small) indicates stationarity**

**kpss.test(trans\_diff) # null hypothesis is that data is stationary. p-value=0.1 --> data is stationary for alpha=0.05**

**# Check periodic diagram to see if there is any unusual frequency**

**n = length(trans\_diff)**

**I = abs(fft(trans\_diff))^2/n #perioddic**

**P = (4/n)\*I[1:(n/2)]**

**f = 0:round((n/2 -1))/n**

**plot(f, P,type = "l", xlab = "Frequency", ylab = "Scaled Periodogram")**

**#plot seems to have a lot of spikes.**

**#look at monthy mean:**

**mdf1 = df1e**

**mdf1$Date = as.Date(mdf1$Date)**

**mdf1$month = format(mdf1$Date, "%m")**

**monthly\_mean = aggregate(mdf1$activity, by = list(mdf1$month), FUN = mean)**

**plot(monthly\_mean,type="b", main="Monthly Means Plot for Trend", xlab="Month", ylab="Mean")**

**abline(h = mean(monthly\_mean$x)) #clearly not stationary**

**# Define some different transformations that combines BoxCox and Confidence Interval adjustments**

**transform = function(dat){**

**bool = (class(dat) == 'ts')**

**if(bool){**

**tinfo = tsp(dat)**

**}**

**dat = as.numeric(dat)**

**dat\_min = min(dat)**

**if (dat\_min < 0){**

**dat\_pre = dat - dat\_min + 0.001**

**}**

**lam= BoxCox.lambda(dat\_pre, method = "loglik")**

**out <- boxcox(dat\_pre~1)**

**x = range(out$x[out$y > max(out$y)-qchisq(0.95,1)/2])**

**print(paste0("Optimal lambda is ",lam ))**

**print(paste0("CI of lambda is ",x ))**

**dfbc = BoxCox(dat\_pre, lam)**

**if (bool){**

**dfbc = ts(dfbc,freq = tinfo[3], start = tinfo[1])**

**}**

**return (dfbc)**

**}**

**test = transform(df1)**

**# write a function that inverts the transformation done by 'transform' function above**

**inversebc = function(dat, lam, dat2){**

**bool = (class(dat) == 'ts')**

**if(bool){**

**tinfo = tsp(dat)**

**}**

**dat = as.numeric(dat)**

**dat2 = as.numeric(dat2)**

**dfmin = min(dat2)**

**inv = InvBoxCox(dat, lambda = lam)**

**if (dfmin <0){**

**inv = inv + dfmin -0.001**

**}**

**if(bool){**

**inv = ts(inv, freq = tinfo[3], start = tinfo[1])**

**}**

**return (inv)**

**}**

**# There maybe multiple seasonality patterns ( weekly, monthly): suggest to use tbats**

**#==============================================================**

**# Fit Model & Run Diagnostics.**

**#first initial guess using auto.arima**

**# may need to turn off approximation for long time series.**

**trants = ts(trans, freq = 365.25/7, start = 2004)**

**modelARIMA = auto.arima(trants, stepwise = FALSE, approximation = FALSE, D=1)**

**res\_plot(modelARIMA) #AIC=108.38**

**#ARIMA(0,1,3)(0,1,2)[52]**

**#improve model using fourier transform**

**m1 = fourier\_model(trants, method = "auto") #AIC = 67.5**

**res\_plot(m1$model)**

**# the acf of residual plot still show signficant spike at 1.**

**# Ljung-test to diagnosis it:**

**Box.test(residuals(m1$model), type = c("Ljung-Box")) # it seems to fails ljung-box test but it actually doens't matter.**

**# Write a function that calculates AIC and BIC given non-seasonal parameters(assuming model is fourier-regressed ARIMA)**

**information\_calculator = function(model\_list,drift=TRUE, ){**

**out = list()**

**index = 1**

**for (param in model\_list){**

**temp = fourier\_model(trants, method = c("manual"), param, drift=drift)**

**out[[index]]=c(temp$model$aic, temp$model$bic)**

**index = index + 1**

**}**

**return(out)**

**}**

**# Model generator generates a list of non-seasonal parameter candidates given one**

**# Used in ul**

**model\_generator = function(ns\_order){**

**model\_list=list()**

**for (ar in (ns\_order[1]-1):(ns\_order[1]+1)){**

**if (ar>=0){**

**for (ma in (ns\_order[3]-1):(ns\_order[3]+1)){**

**if (ma>=0){**

**model\_list[[length(model\_list)+1]] = c(ar, ns\_order[2], ma)**

**}**

**}**

**}**

**}**

**return(model\_list)**

**}**

**cross\_validate = function(drift=TRUE, data, trants, model\_list, inv, ...){**

**mse\_list = NULL**

**for (param in model\_list){**

**mse = 0**

**min\_st = 161**

**n = length(trants)**

**fold = (n-161)/7**

**for (i in 1:6){**

**train\_cut = min\_st + (fold\*(i))**

**train = ts(trants[1:train\_cut], freq = 365.25/7, start = 2004)**

**test\_cut = (train\_cut+1):(train\_cut+fold)**

**test = data[test\_cut]**

**train\_fit = fourier\_model(train, method = c("manual"), param, drift=drift)**

**fc = forecast(train\_fit$model, xreg=fourier(train, K=train\_fit$K, h=length(test\_cut)))**

**error = inv(as.numeric(fc$mean), ...) - test**

**mse = mse + mean(error^2)**

**}**

**mse\_list = c(mse\_list, mse)**

**}**

**return(mse\_list)**

**}**

**cross\_validate2 = function(drift=TRUE, data, trants, model\_list, inv, ...){**

**mse\_list = NULL**

**for (param in model\_list){**

**mse = 0**

**min\_st = 161**

**n = length(trants)**

**fold = (n-161)/7**

**for (i in 1:6){**

**train\_cut = min\_st + (fold\*(i))**

**train = ts(trants[1:train\_cut], freq = 365.25/7, start = 2004)**

**test\_cut = (train\_cut+1):(train\_cut+fold)**

**test = data[test\_cut]**

**test\_start = 2004 + 1/(365.25/7)\*(train\_cut+1)**

**npred = length(test\_cut) # number of periods ahead to forecast**

**train\_fit1 = fourier\_model(train, method = c("manual"), param, drift=drift)**

**ns\_order = train\_fit1$model$arma[c(1,6,2)]**

**p = ns\_order[1]; q = ns\_order[3];**

**k = train\_fit1$K**

**mo = tso(train, types=c("AO","LS","TC"), tsmethod = "auto.arima", remove.method = "bottom-up",args.tsmethod = list(max.p=p, max.q=q, seasonal=FALSE, xreg=fourier(train, K=k)))**

**newxreg = outliers.effects(mo$outliers, length(train)+ npred)**

**newxreg = newxreg[-seq\_along(train),]**

**newxreg = cbind(fourier(train, K=train\_fit1$K, h=length(test\_cut)),newxreg)**

**newxreg = ts(newxreg, start = test\_start, frequency = 365.25/7)**

**fc = forecast.Arima(mo$fit, xreg= newxreg, h = npred)**

**error = inv(as.numeric(fc$mean), ...) - test**

**mse = mse + mean(error^2)**

**}**

**mse\_list = c(mse\_list, mse)**

**}**

**return(mse\_list)**

**}**

**cross\_validate3 = function(data,...){**

**min\_st = 161**

**n = length(data)**

**fold = (n-min\_st)/7**

**mse=0**

**for (i in 1:6){**

**train\_cut = min\_st + (fold\*(i))**

**train = ts(data[1:train\_cut], freq = 365.25/7, start = 2004)**

**test\_cut = (train\_cut+1):(train\_cut+fold)**

**test = data[test\_cut]**

**train\_fit = stlf(train, h =length(test\_cut), ...)**

**error = as.numeric(train\_fit$mean) - test**

**mse = mse + mean(error^2)**

**}**

**print(paste0("MSE using stlf is ", mse))**

**return (list(model = stlf(data), mse = mse))**

**}**

**ultimate = function(drift=TRUE, data, trans, inv, ...){**

**##Inputs**

**# data should be a ts object with appropriate frequency for weekly data**

**# trans is the transformation to be applied to data. It must have parameter named "dat" which isthe original data.**

**# inv is the inverse of the transformation applied to original data. It must have parameter named "data" which is the transformed data to be inverted back.**

**# ... are arguments for trans and inv functions separated by delimiter '..'. Separtor '..' is a necessary argument.**

**##Output is a list whose elements are**

**# fourier\_fitted : ts object including the prediction for next two years using fourier-adjusted arima model**

**# outlier\_fitted : ts object including the prediction for next two years using fourier-adjusted arima model with additional outlier handling**

**# fourier\_model : forecast object that contains information of transformed data and predictions(on transformed data) for fourier model**

**# fourier\_fitted : forecast object that contains information of transformed data and predictions(on transformed data) for outlier-adjusted fourier\_model**

**# mse : sum of 6 mse's in 7-fold cross-validation for fourier model**

**# mse\_outlier : sum of 6 mse's in 7-fold cross-validation for outlier-adjusted fourier model**

**arguments <- list(...)**

**if('..' %in% arguments){**

**if(arguments[1] == '..'){**

**inv\_arg <- arguments[2:length(arguments)]**

**trans\_arg = NULL**

**} else if( arguments[length(arguments)]=='..'){**

**trans\_arg <- arguments[1:(length(arguments)-1)]**

**inv\_arg = NULL**

**} else{**

**i = which(arguments == '..')**

**trans\_arg <- arguments[1:(i-1)]**

**inv\_arg <- arguments[(i+1):length(arguments)]**

**}**

**} else error("You must include '..' as an argument")**

**# transform data**

**if (is.null(trans\_arg)){**

**trants = trans(data)**

**} else trants = do.call(trans, append(list(data), trans\_arg))**

**#n = length(trants)**

**# inital guess or parameters**

**init\_model = fourier\_model(trants, method = "auto", drift=drift)**

**ns\_order = init\_model$model$arma[c(1,6,2)] #non-seaonsal order**

**p = ns\_order[1]; q = ns\_order[3]**

**init\_model2 = tso(trants, tsmethod = "auto.arima", remove.method = "bottom-up",args.tsmethod = list(max.p=p, max.q=q, seasonal=FALSE, xreg=fourier(trants, K=init\_model$K)))**

**# generate model list**

**model\_list = model\_generator(ns\_order)**

**model\_list2 = list(init\_model2$fit$arma[c(1,6,2)])**

**# implement cross-validation and choose&fit final model(for fourier, outlier-adjusted fourier, and stlf)**

**if (is.null(inv\_arg)){**

**mse\_list2 = cross\_validate2(drift=TRUE, data=data, trants=trants, model\_list=model\_list2, inv=inv)**

**print(paste0("MSE with outlier effect is ", mse\_list2[1]))**

**mse\_list = cross\_validate(drift=TRUE, data, trants, model\_list, inv)**

**} else{**

**mse\_list2 = do.call(cross\_validate2, append(list(drift=TRUE, data=data, trants=trants, model\_list=model\_list2, inv=inv), inv\_arg))**

**print(paste0("MSE with outlier effect is ", mse\_list2[1]))**

**mse\_list = do.call(cross\_validate, append(list(drift=TRUE, data=data, trants=trants, model\_list=model\_list, inv=inv), inv\_arg))**

**}**

**stlf\_fc = cross\_validate3(data)**

**# implement final prediction with model with best MSE result**

**final\_model\_param = unlist(model\_list[mse\_list == min(mse\_list)])**

**print(paste0("Non-seasonal model parameter used is ","(", toString(final\_model\_param),")"))**

**print(paste0("MSE is ", mse\_list[mse\_list == min(mse\_list)]))**

**final\_model = fourier\_model(trants, method = "manual", final\_model\_param, drift=TRUE)**

**print(paste0("Number of coefficients in Fourier transformation is ", final\_model$K))**

**# set up for outlier preidction**

**ns\_order = final\_model$model$arma[c(1,6,2)]**

**p = ns\_order[1]; q = ns\_order[3];**

**k = final\_model$K**

**outlier\_model = tso(trants, types=c("AO","LS","TC"), tsmethod = "auto.arima", remove.method = "bottom-up",args.tsmethod = list(max.p=p, max.q=q, seasonal=FALSE, xreg=fourier(trants, K=k)))**

**newxreg = outliers.effects(outlier\_model$outliers, length(trants)+ 104)**

**newxreg = newxreg[-seq\_along(trants),]**

**newxreg = cbind(fourier(trants, K=final\_model$K, h=104), newxreg)**

**newxreg = ts(newxreg, start = 2014 + 1/(365.25/7)\*3, frequency = 365.25/7)**

**# predict and plot**

**fc = forecast(final\_model$model, xreg=fourier(trants, K=final\_model$K, h=104))**

**outlier\_fc = forecast.Arima(outlier\_model$fit, xreg= newxreg, h = 104)**

**if (is.null(inv\_arg)){**

**final\_fitted = inv(fc$mean)**

**outlier\_fitted = inv(outlier\_fc$mean)**

**} else {**

**final\_fitted = do.call(inv, append(list(fc$mean), inv\_arg))**

**outlier\_fitted = do.call(inv, append(list(outlier\_fc$mean), inv\_arg))**

**}**

**whole\_data = ts(c(data,final\_fitted), freq = 365.25/7, start = 2004)**

**outlier\_data = ts(c(data,outlier\_fitted), freq = 365.25/7, start = 2004)**

**stlf\_data = ts(c(data, stlf\_fc$mean), freq = 365.25/7, start = 2004)**

**robj = list(fourier\_fitted = whole\_data, outlier\_fitted = outlier\_data, stlf\_fitted = stlf\_data,**

**fourier\_model = fc, outlier\_model = outlier\_fc, stlf\_model = stlf\_fc$model,**

**mse=min(mse\_list), mse\_outlier = mse\_list2[1], mse\_stlf = stlf\_fc$mse)**

**return(robj)**

**}**

**# note: some model may have the smallest AIC but not RSME and vice versa**

**# when checking the model using ljung box test, sometimes the model doens't pass the test but**

**# it still gives the lowest RSME**

**#==============================================================**

**# calculate confidence interval to improve rounding when choosing the lambda for transformations**

**ci\_yj = function(dat){**

**out = boxCox(dat~1, family="yjPower", plotit = F)**

**x = range(out$x[out$y > max(out$y)-qchisq(0.95,1)/2])**

**print(x)**

**}**

**lam1 = ci\_yj(df1)**

**t1 = transform(df1)**

**ult1 = ultimate(drift=TRUE, df1, yeo.johnson, yeo.johnson, lambda=0.1,'..',lambda=0.1, inverse=TRUE)**

**#MSE is 1.525141, MSE\_outlier is 1.847089, MSE\_stlf is 1.454666**

**tult1=ultimate(drift=TRUE, df1, transform, inversebc, "..", lam=0.25, dat2=df1)**

**#MSE is 1.533934, MSE\_outlier is 2.028533, MSE\_stlf is 1.454666**

**m1 = list(c(5, 1, 4), c(3, 1, 1))**

**information\_calculator(m1)**

**submit1 = ult1$fourier\_fitted[526:629]**

**write.table("Q1\_Hyunouk\_Ko\_23837910.txt",x=submit1,row.names = FALSE, col.names=FALSE)**

**#==============================================================**

**#==============================================================**

**#==============================================================**

**## Question2**

**df2 = read.csv("q2\_train.csv")**

**df2 = df2[,2]**

**df2 = ts(df2,freq = 365.25/7, start = 2004)**

**#plot dataset:**

**plot(df2, type = "l") #doesn't look stationary**

**acf(df2)**

**pacf(df2)**

**#log transform by the min:**

**df2a = df2**

**df2\_min = min(df2a)**

**df2a= log(df2a - df2\_min + 0.001)**

**plot.ts(df2a) # doesn't seem to work**

**acf(df2a)**

**pacf(df2a)**

**# try difference on the original data:**

**df2b = df2**

**df2b= diff(df2b)**

**df2b\_min = min(df2b)**

**df2b= log(df2b - df2b\_min + 0.001)**

**plot.ts(df2b)**

**acf(df2b)**

**pacf(df2b)**

**#try difference on the log transform:**

**df2c = diff(df2a)**

**plot.ts(df2c)**

**acf(df2c)**

**pacf(df2c)**

**#try to take log transform of differenced data**

**df2d = df2b**

**df2d\_min = min(df2d)**

**df2d= log(df2d - df2d\_min + 0.001)**

**plot.ts(df2d) #very good except around 460**

**acf(df2d) #good**

**pacf(df2d) #good**

**#yeo\_johnson transformation**

**trants2 = yeo.johnson(df2, 0.35)**

**trants2\_diff = diff(trants2)**

**trants2\_diff2 = diff(trants2\_diff)**

**# Stationarity Check**

**Box.test(trants2\_diff) #p-value = 0.005694 indicates that the series is stationary**

**adf.test(trants2\_diff, alternative = "stationary") #p-value = 0.01(small) indicates stationarity**

**kpss.test(trants2\_diff) #null hypothesis is that data is stationary. p-value=0.1 --> data is stationary for alpha=0.05**

**# Predict using the same procedure as before**

**t2 = transform(df2)**

**lam2 = ci\_yj(df2)**

**ult2=ultimate(drift=TRUE, df2, yeo.johnson, yeo.johnson, lambda=0.4, "..", lambda=0.4, inverse=TRUE)**

**#MSE is 1.144993 and MSE\_outlier is 1.266753, MSE\_stlf is 0.672989739880507**

**tult2 =ultimate(drift=TRUE, df2, transform, inversebc, "..", lam=0.25, dat2=df2)**

**#MSE is 0.9990179, MSE\_outlieris 1.523128, MSE\_stlf is 0.672989739880507**

**#check AIC, BIC:**

**m2 = list(c(0, 1, 1), c(2, 1, 0))**

**information\_calculator(m2)**

**submit2 = tult2$fourier\_fitted[526:629]**

**write.table("Q2\_Hyunouk\_Ko\_23837910.txt",x=submit2, row.names = FALSE, col.names=FALSE)**

**#==============================================================**

**#==============================================================**

**#==============================================================**

**## Question 3**

**df3 = read.csv("q3\_train.csv")**

**df3 = df3[,2]**

**df3 = ts(df3, freq = 365.25/7, start = 2004)**

**t3 = transform(df3)**

**lam3 = ci\_yj(df3)**

**ult3 = ultimate(drift=TRUE, df3, yeo.johnson, yeo.johnson, lambda=-0.2, "..", lambda=-0.2, inverse=TRUE)**

**#MSE is 1.20442330289085, MSE\_outlier is 1.692021, MSE\_stlf is 0.47566059103331**

**tult3 = ultimate(drift=TRUE, df3, transform, inversebc, "..", lam=0.25, dat2=df3)**

**#MSE is 0.797214239292739, MSE\_outlier is 1.108866, MSE\_stlf is 0.47566059103331**

**#check AIC, BIC:**

**m3 = list(c(6, 1, 1), c(3, 1, 3))**

**information\_calculator(m3)**

**submit3 = as.numeric(cross\_validate3(df3, lambda=1.35, robust=TRUE)$model$mean)**

**write.table("Q3\_Hyunouk\_Ko\_23837910.txt",x=submit3, row.names = FALSE, col.names=FALSE)**

**#==============================================================**

**#==============================================================**

**#==============================================================**

**## Question 4**

**df4 = read.csv("q4\_train.csv")**

**df4 = df4[,2]**

**df4 = ts(df4,freq = 365.25/7, start = 2004)**

**t4 = transform(df4)**

**lam4 = ci\_yj(df4)**

**ult4=ultimate(drift=TRUE, df4, yeo.johnson, yeo.johnson, lambda=-0.08, "..", lambda=-0.08, inverse=TRUE)**

**#MSE is 1.14686676096389, MSE\_outlier is 1.319493, MSE\_stlf is 1.1690551322219**

**tult4 = ultimate(drift=TRUE, df4, transform, inversebc, "..", lam=0.35, dat2=df4)**

**#MSE is 1.179822, MSE\_outlier is 1.193635, MSE\_stlf is 1.1690551322219**

**#check AIC, BIC:**

**m4 =list(c(0, 1, 1),c(0, 1, 1))**

**information\_calculator(m4)**

**submit4 = ult4$fourier\_fitted[526:629]**

**write.table("Q4\_Hyunouk\_Ko\_23837910.txt",x=submit4,row.names = FALSE, col.names=FALSE)**

**#==============================================================**

**#==============================================================**

**#==============================================================**

**## QUestion 5**

**# Step 1: Identify unusual observations and transform data**

**df5 = read.csv("q5\_train.csv")**

**df5 = df5[,2]**

**df5 = ts(df5, freq = 365.25/7, start = 2004)**

**plot(df5)**

**tsdisplay(df5)**

**# detect and replace outlier.**

**df5\_out = tsclean(df5)**

**# plot dataset:**

**plot(df5, type = "l") #doesn't look stationary**

**acf(df5)**

**#log transform by the min:**

**df5a = df5**

**df5\_min = min(df5a)**

**df5a= log(df5a - df5\_min + 0.001)**

**df5a\_diff = diff(df5a)**

**tsdisplay(df5a\_diff)**

**# doesn't seem to work**

**# try difference on the original data:**

**df5b = df5**

**df5b= diff(df5b)**

**tsdisplay(df5b)**

**#try difference on the log transform:**

**df5ab\_diff = diff(df5a)**

**tsdisplay(df5ab\_diff)# doesn't seem to change anything**

**#try Yeo-Johnson transformation**

**df5e = read.csv("q5\_train.csv")**

**yw\_param = preProcess(df5e, method = c("YeoJohnson"))**

**yw\_param #search for optimal lambda, return #0.14 but 0.15 is still within CI so choose 0.15 for convenience**

**trans = yjPower(df5, -0.5, jacobian.adjusted = FALSE) #stabilize variance**

**trans\_diff = diff(trans) #stabilize mean**

**tsdisplay(trans\_diff) #pacf: there are apparently 3 clusters at 24, 48, 52,**

**#There are peak at 52, 106, 156 in acf.**

**# it seems that yeo\_johnson transformation didn't do well on outlier version**

**# Now check if we have stationary process**

**# Check using Ljung- Box test:**

**Box.test(trans\_diff) # p-value = 0.005694 indicates that the series is stationary**

**adf.test(trans\_diff, alternative = "stationary") #p-value = 0.01(small) indicates stationarity**

**kpss.test(trans\_diff) # null hypothesis is that data is stationary. p-value=0.1 --> data is stationary for alpha=0.05**

**# Check periodic diagram to see if there is any unusual frequency**

**n = length(trans\_diff)**

**I = abs(fft(trans\_diff))^2/n #perioddic**

**P = (4/n)\*I[1:(n/2)]**

**f = 0:round((n/2 -1))/n**

**plot(f, P,type = "l", xlab = "Frequency", ylab = "Scaled Periodogram")**

**#plot seems to have a lot of spikes.**

**#look at monthy mean:**

**mdf1 = df1e**

**mdf1$Date = as.Date(mdf1$Date)**

**mdf1$month = format(mdf1$Date, "%m")**

**monthly\_mean = aggregate(mdf1$activity, by = list(mdf1$month), FUN = mean)**

**plot(monthly\_mean,type="b", main="Monthly Means Plot for Trend", xlab="Month", ylab="Mean")**

**abline(h = mean(monthly\_mean$x)) #clearly not stationary**

**# Predict using the same procedure as before**

**t5 = transform(df5)**

**lam5 = ci\_yj(df5)**

**ult5=ultimate(drift=TRUE, df5, yeo.johnson, yeo.johnson, lambda=-0.5, "..", lambda=-0.5, inverse=TRUE)**

**#MSE is 0.3342673, MSE\_outlier is 0.4120022, MSE\_stlf is 0.327029169158465**

**tult5 = tult=ultimate(drift=TRUE, df5, transform, inversebc, "..", lam=0.3, dat2=df5)**

**#MSE is 0.3499961, MSE\_outlier is 0.3705752, MSE\_stlf is 0.327029169158465**

**#check AIC, BIC**

**m5 = list(c(1, 1, 0), c(0, 1, 1))**

**information\_calculator(m5)**

**submit5 = as.numeric(cross\_validate3(df5, lambda=1.5, robust=TRUE)$model$mean)**

**write.table("Q5\_Hyunouk\_Ko\_23837910.txt",x=submit5,row.names = FALSE, col.names=FALSE)**