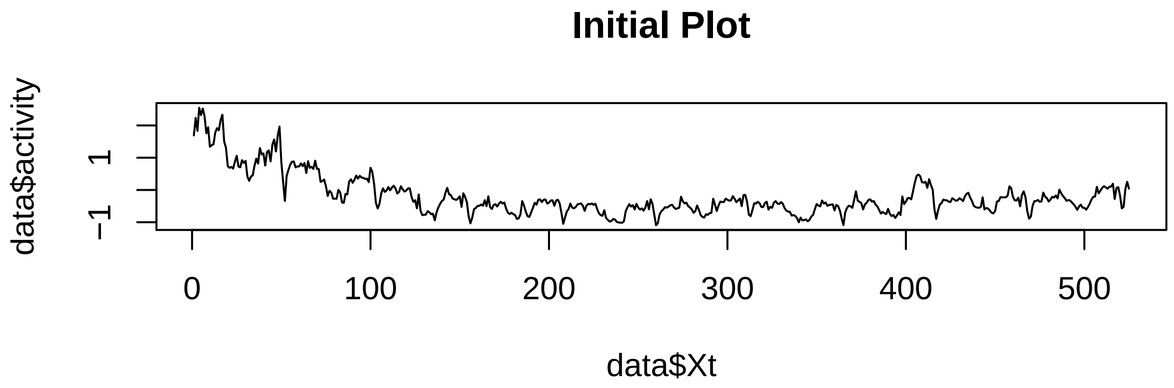
STAT153_MT2_Report

Jong Ha Lee, Binyi Cai

4/18/2017

1. Exploratory Data Analysis

In this report, we specifically use `q5_train.csv` dataset for our time series analysis.



In our initial plot, we see that the time series shows an overall parabolic mean trend, as well as sinusodial fluctuations. Furthermore, we see that there are some fluctuations that are larger than others, and thus a transformation to stabilize the variance may be necessary.

We conduct a log transformation after adding 2 to the original time series data, due to concerns in log-transforming negative numbers. In mathematical terms, we conduct a `log(Xt + 2)` transformation in order for variance stabilization. We continue to use this log transformed data, mainly because it provides to be more helpful in fitting parametric trend that is more conducive in forecasting.

---

2. Detrending Mean Structures

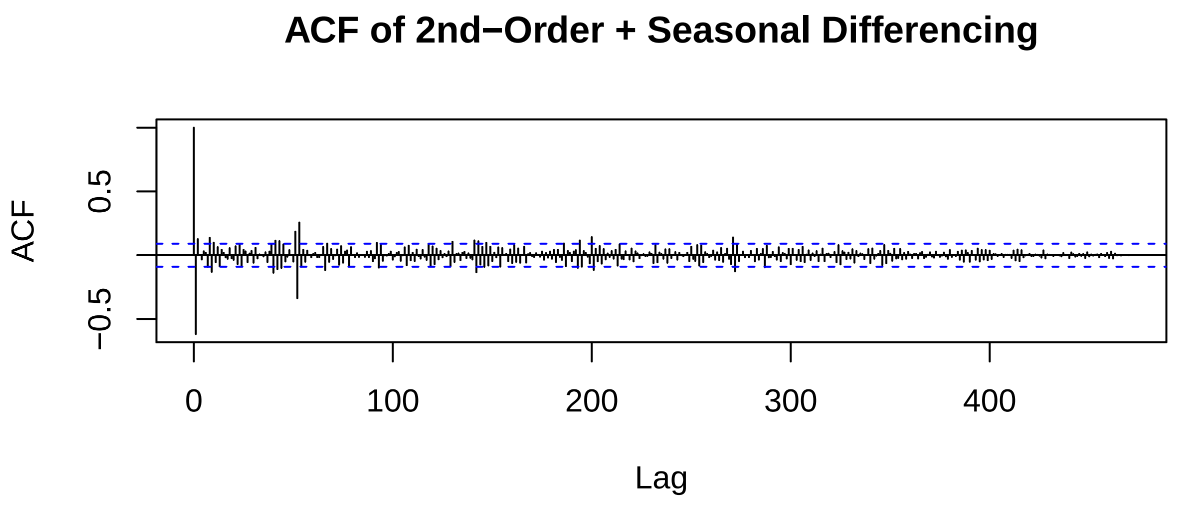
In this section, we discuss various detrending methods, and choose 2 detrending methods which seem to reduce our data to a stationary structure, so that we can fit Time Series models for the Covariance structures in our data.

2a) Detrending via Differencing

First, we try to detrend the mean via differencing. First, in order to remove the original mean trend, we 2nd-order difference it with lag = 1 mainly because the data seems to show a parabolic trend, or a polynomial with degree 2-trend. However, we noticed in the original plot that there were seasonal fluctuations. Thus, an order of seasonal differencing may be required. But what would our lag be?

Intuitively, we make an educated guess that there are 52 weeks in a year, and since our Google Trends data is weekly, these spikes occur due to the annual trends that are present in our dataset. Thus, we choose lag = 52 for our seasonal differencing, on top of the 2nd-order differencing we did. We can also verify this via looking at the ACF after first differencing, but we leave that to the reader.

The ACF of 2nd-order differencing, then seasonal differencing is shown below:



The differencing shows that the seasonal affects have been greatly reduced, though we still see larger-than-normal spikes at lags of multiples of 50, and unusually large spikes in both ways at lag 50-52. Furthermore, the ACFs seem to be getting smaller and smaller as lag increases, also pointing to the fact that our model may have reduced the data into stationary noise.

How can we check the model is stationary? We use the Augmented Dickey-Fuller Test to test stationarity of our model. In this test, the Null Hypothesis is: Data is not stationary. The Alternative Hypothesis is: Data is stationary. Thus, our desirable goal is that the model is stationary, meaning we reject our null hypothesis. We use the ADF-Test on both lag = 52, because this is the most likely lag which the time series data is strongly correlated, and we have also seen this in previous ACF plots as well.

```
## Warning in adf.test(diff.data, alternative = c("stationary"), k = 52): p-
## value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff.data
## Dickey-Fuller = -4.1105, Lag order = 52, p-value = 0.01
## alternative hypothesis: stationary
```

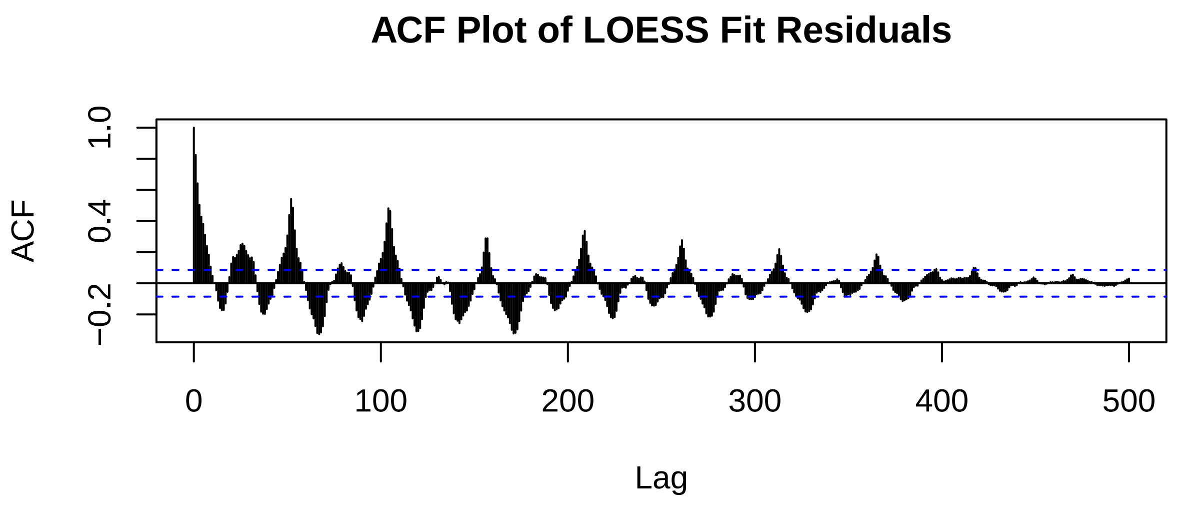
The ADF test shows that our differenced data is stationary. Thus, we have successfully de-trended our mean structure to stationary noise, and are now ready to do some ARIMA model fitting. However, is differencing really the best/only way to detrend?

In this next section, we explore another method of detrending through smoothing and sinusodial fitting.

2a) Detrending via Smoothing and Sinusodial Fitting

Going back to our original log-transformed plot, we saw a parabolic trend, as well as sinusodial fluctuations indicating seasonality. Can we systematically and in a stepwise fashion, reduce the data into stationary noise without differencing?

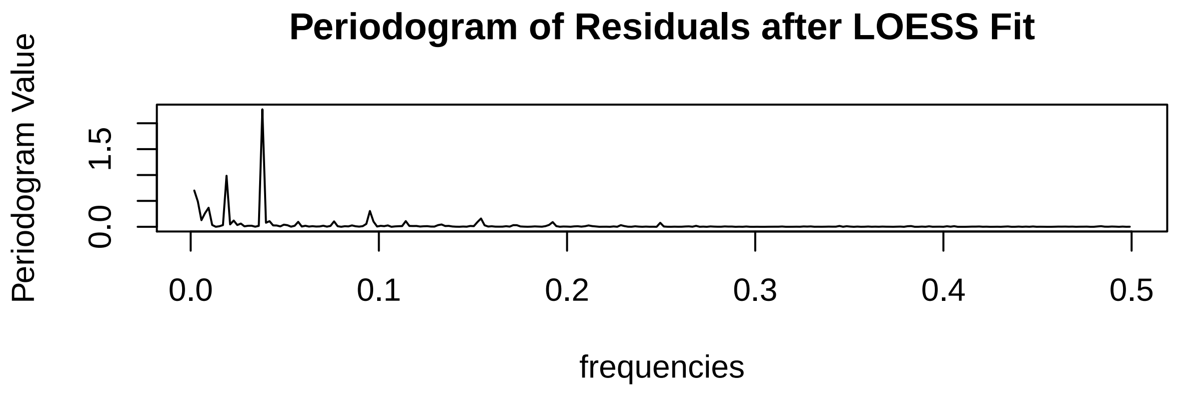
First, in order to remove the overall, global parabolic trend, we fit a LOESS smoothing model. How do the ACF of the residuals look?



It shows huge sinusodial fluctuations. If we stopped detrending via fitting here, this model is definitely not stationary noise and would be terrible for forecasting. However, if we were to fit sinusodial functions here *next*, this shows great promise in reducing our data to white noise.

Following our logic through, now we fit sinusodial functions via Discrete Fourier Transform. First, We find the most important frequencies to fit to these residuals via a periodogram.

The periodogram is shown below:

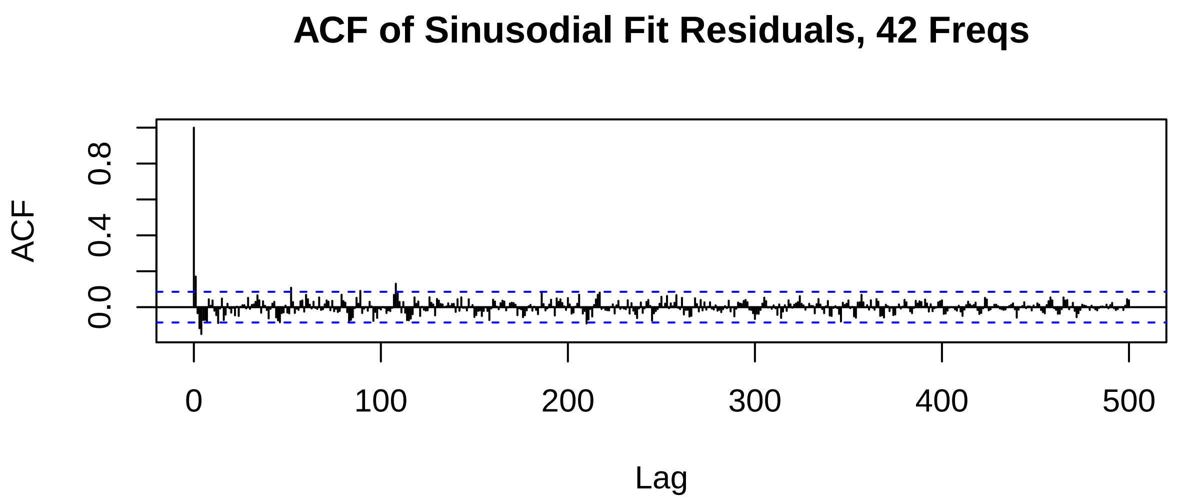


We see roughly 4-5 dominant frequencies. Converting the frequencies into periods for more interpretable results, what are those periods?

```
## [1]  26.25  52.50 525.00 262.50 105.00
```

We see some expected, and unexpected results here. We see that unsurprisingly, the periods of 26, 52, and 104 are included here, under the reasonable assumption that seasonal trend occurs semi-annually, annually, and every two years. However, we also see very surprising periods here, namely 10-year and 5-year periods. Perhaps this has to do with the fact that we were unable to completely detrend the global mean structure from Loess, and it still exists in the periodogram fitting.

When plotting the ACF after fitting 5 frequencies, it clearly still showed a seasonal fluctuation, and that we did not use enough frequencies, or in other words, underfitted our data. Let us try to fit a total number of frequencies of 8% of total number of datapoints = 42. Let us look at the ACF to see whether if it has improved.



The data looks much better, and much closer to stationary noise. However, we will check this again via ADF test to see if it is stationary noise not just by looking at the ACF plot.

```
## Warning in adf.test(season.fit$residuals, alternative = "s", k = 52): p-
## value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  season.fit$residuals
## Dickey-Fuller = -6.2344, Lag order = 52, p-value = 0.01
## alternative hypothesis: stationary
```

The ADF test determines that our data is significant enough to reject the null hypothesis that our residuals after sinusoidal fitting is not stationary.

In conclusion, in our second detrending method, we first fit a overall, global mean structure via LOESS, and accounted for the sinusodial fluctuations of the residuals, and was able to reduce the data to stationary noise. Now, we move on to fitting time series models to better model the covariance structure in our data.

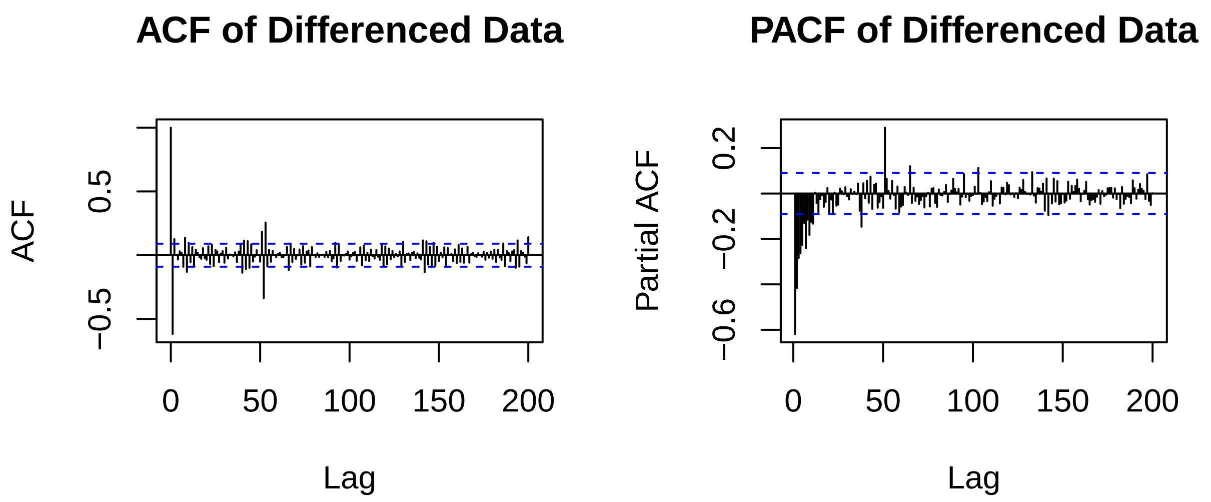
Note: From here onwards, we divide our analysis into two based on the two detrending methods, for holistic reasons. Then, we come back in analyzing which detrending method + covariance modeling worked best for forecasting.

3a. Differencing - Model Fitting & Diagnostics

3a.1) Determining ARIMA models based on ACF, PACF

We will first analyze covariance structure of our differenced data, our first detrending approach. Let us look at the ACF and PACF of the dataset to determine how many AR, MA, and Seasonal AR, Seasonal MA terms are best to fit this model. Note that our differencing has been already decided, in that we 2nd-order differenced in the normal ARIMA, and we needed to seasonal difference once to get a stationary noise to fit these models. Thus, d = 2, and D = 1.

Now, let's look at ACF and PACF of the differenced data.



For the seasonal part, there are significant spikes at 1 and around 52 in the ACF, and then it cuts off after that, which leads us to a seasonal MA(1) with period 52. PACF can either be regarded as tailing off at every seasonal lag, or cutting off after 2 seasonal lags(around 52,and 104).

For the non-seasonal part, ACF cuts off sharply after lag 1, suggesting an MA(1). PACF seems to be tailing off, and it's most likely to suggest a pure MA process, but it's also possible that p is 3 or more.

Thus, based on our eye heuristics, we determine a model of ARIMA(3,2,1) X ARIMA(1,1,1)[52] model. However, this is somewhat unreliable and there may be better models present. How can we better determine models that might be better for our data?

3a.2) Model Diagnostics - Local External Validity

In determing 2-3 top models to diagnose and analyze its performance, we create functions which help us compare different models. Given the seasonal ARIMA component P,D,Q, and the ARIMA differencing component d, the function tests different combinations of p and q in the normal ARIMA component. We chose AIC and BIC as the test to conduct mainly because these criterions tell us how well we fit signal, and ignore noise, which is important for forecasting as well.

Note that in this function, we should subtract -1 on both row and column to get the actual p and q we used to fit an ARIMA model. So [1,2] element would be the model's AIC where p = 0 and q = 1. Note that rows indicate the p (AR component), and columns indicate the q (MA component). Furthermore, we test two methods - CSS-ML and ML to get AIC/BICs. If both don't work due to optimization problems in R implementation, we ignore that model.

We note that this is part of Model Diagnostics as well, in that AIC and BIC tells us the performance of models in the aspect of Local External Validity. Once we decide on models to move forward based on AIC and BIC, we evaluate its Internal Validity and General External Validity, which are the other two model diagnostic criterions.

The AIC and BIC matrix is shown below, after running multiple different combinations of ARIMA models. NAs indicate that the ARIMA function was not able to utilize CSS-ML or ML methods to fit the model, and hence wasn't able to produce AIC or BIC. We show the AIC matrix below.

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] -420.4866 -866.5645 -943.7338 -955.3078
## [2,] -627.1848        NA        NA        NA
## [3,] -714.4250 -928.8134        NA -956.7649
## [4,] -754.5192 -935.7052        NA        NA
```

We list all the best models based on AIC and BIC. Note that we also incorporate the simplicity of the model, based on the number of AR,MA, Seasonal AR, and Seasonal MA terms: `ARIMA(2,2,3) X ARIMA(1,1,1)[52]`, `ARIMA(0,2,3) X ARIMA(1,1,1)[52]`, `ARIMA(0,2,1) X ARIMA(1,1,1)[52]`, and `ARIMA(0,2,2) X ARIMA(1,1,1)[52]`.

In cross-referencing both criterion, we get 4 models total.

Since this Local External Validity criterion is only one part of the whole model selection process, we create a weighing measure, where each model's AIC and BIC, respectively, are divided by the best (lowest) AIC and BIC. Thus, the best score would be 1, and the worst would be 0.

3a.3) Model Diagnostics - Local Internal Validity

Next, we evaluate whether the residuals from fitting these time series models are white noise. Our goal up until now was to find a model so that we would reduce and detrend our data into purely white noise - so that in forecasting, we only, and accurately fit trend and signal.

We evaluate whether the residuals are white noise or not based on the Ljung-Box-Pierce Test. In the R implementation, the Null Hypothesis is: The residuals are white noise. The Alternate Hypothesis is: The residuals are not white noise. Thus, we wish *not* to reject the null hypothesis. Furthermore, we subtract the number of coefficients we fitted from the degrees of freedom. Lastly, we evaluate white noise at lag = 104, because we note that to check white noise, we mustn't only check seasonality at when we predict, but also whether that seasonality continues in the long term, as our primary goal is forecasting. *We also cite our source of Rob J. Hyndman's website, who created the `forecast` package in R.*

```
##      sarima1      sarima2      sarima3      sarima4
## 0.7403398489 0.6943881110 0.0002131298 0.4785848447
```

We see that except for one model, the ARIMA(0,2,1) X ARIMA(1,1,1)[52], every model's residuals is not significant enough to reject the null hypothesis that the model is stationary. Again as done before, we calculate the scores based on the p-values, and store those scores into our score matrix.

3a.4) Model Diagnostics - General External Validity

This diagnostic/heuristic is probably the most important, mainly because our goal of this project was to forecast 2 years (104 observations) worth of data, and General External Validity exactly evaluates the performance our models in forecasting. By using time-series Cross Validation, which divides our original datset into training and testing datset, predict based on the model created from training set, and evaluate the performance by Mean Squared Error between the actual testing set values and the forecasted testing set values.

For all three models, we evaluate the CV MSEs, shown below. We use 4 folds.

3a.5) Model Diagnostics - Final Score

Finally, we have all the scores in all four criterions: AIC, BIC, Local Internal Validity (P-Value from Ljung-Box-Pierce test), and General External Validity (CV-MSE). We note that our most important criterion is CV-MSE, as it is the closest measure of forecasting that we will be doing, and then the next would be AIC/BIC, and lastly, the Local Internal Validity. We compute a weighted-average-score for each model, giving weights of CV = 0.5, mean(AIC+BIC) = 0.3, LIV = 0.2.

```
##      model       aic       bic         liv        cv  LEV_mean total_score
## 1 arima223 1.0000000 0.9926346 1.000000000 1.0000000 0.9963173   0.9988952
## 2 arima023 0.9984770 1.0000000 0.937931562 0.9283830 0.9992385   0.9515494
## 3 arima021 0.9057235 0.9135474 0.000287881 0.6927172 0.9096354   0.6193068
## 4 arima022 0.9863800 0.9920257 0.646439396 0.8152698 0.9892028   0.8336836
```

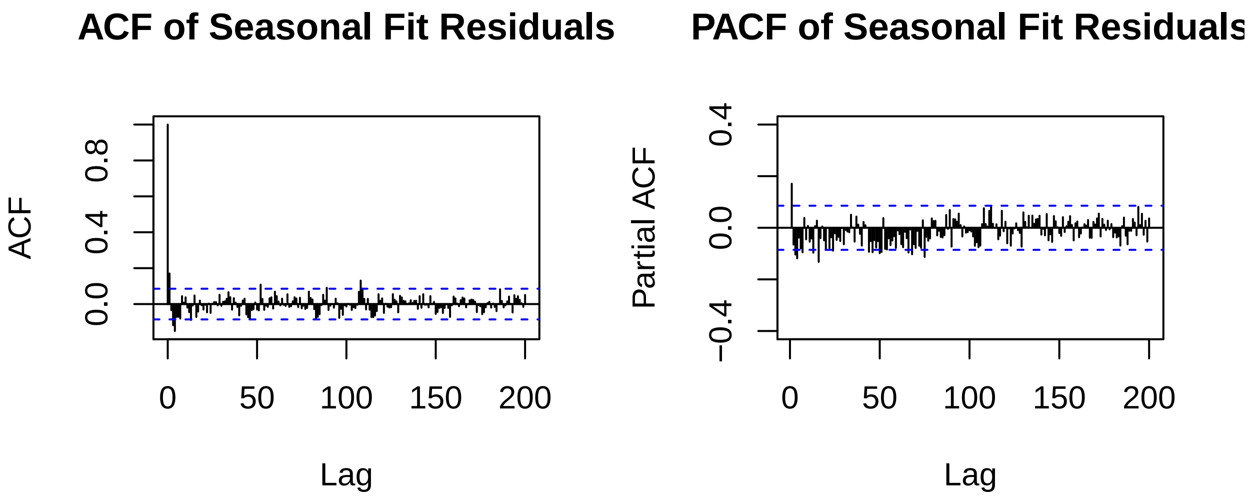
In conclusion, for the **detrending-via-differencing, ARIMA modeling** method, the model we choose is ARIMA(2,2,3) X SARIMA(1,1,1)[52]. However, we wonder if the MSE can be lower, because it seems relatively quite high of 0.6552226 for our ARIMA(2,2,3) X SARIMA(1,1,1)[52] model.

Perhaps seasonal fit will be better for us.

3b. LOESS + Sinusodial Fit - Model Fitting & Diagnostics

3b.1) Determining ARIMA models based on ACF, PACF

We follow the same structure as before, looking at ACF and PACF graphs to see which model might seem to be the best. We note that now we are looking at the covariance structure of the residuals after the sinusodial fit, which was fitting against the LOESS fit residuals. Thus, the ARIMA covariance structuring aspect will be of less importance compared to the differencing methodology.



Based on the ACF, it seems that the ACF rapidly decreases after 1 lag suggesting an MA(1), though there are multiple lags which are beyond the confidence interval for white noise. This could suggest a higher order MA model, but we can see this when we calculate the AIC and BIC matrices.

The PACF seems to be extremely random, and the PACF slowly tails off, though it is not as apparent for lag.max = 200 (very apparent for lag.max = 500). This may suggest an AR term is not necessary.

As for the seasonal component, there doesn't really seem to be a seasonal component as much, as we did reduce it to stationary noise via LOESS and sinusodial fitting.

3b.2) Model Diagnostics - Local External Validity

Again we compute the AIC and BIC matrix to see which combination gives us the lowest AIC, BIC, also taking into account the simplicity of the model.

Based on AIC and BIC, the best models are: `ARIMA(2,0,1)`, `ARIMA(2,0,2)`, `ARIMA(0,0,1)`, and `ARIMA(1,0,2)`.

3b.3) Model Diagnostics - Local Internal Validity

As before, calculating conducting Ljung-Box-Pierce Test and calculating p-values:

```
##    arima1    arima2    arima3    arima4
## 0.1589976 0.1303640 0.5824492 0.1951319
```

It seems that all models passed the white noise test based on Ljung-Box-Pierce Test.

3b.4) Model Diagnostics - General Internal Validity

Finally, running cross validation on all four models:

One important note here is that the mean Cross Validation Error of this model is 0.079886, compared to that of the differencing model which was 0.7776379. This is a shocking difference we see, and we can safely say that the LOESS + sinusodial fit model has a much better forecasting predictability.

3b.5) Model Diagnostics - Final Score

Finally, we have the final score for the seasonal fit models.

```
##      model       aic       bic       liv        cv  LEV_mean total_score
## 1 arima201 1.0000000 1.0000000 0.2729811 0.9998271 1.0000000   0.8545098
## 2 arima202 0.9990820 0.9961395 0.2238203 0.9977915 0.9976108   0.8429430
## 3 arima001 0.9688936 0.9742961 1.0000000 1.0000000 0.9715949   0.9914785
## 4 arima102 0.9969004 0.9968550 0.3350196 0.9991608 0.9968777   0.8656476
```

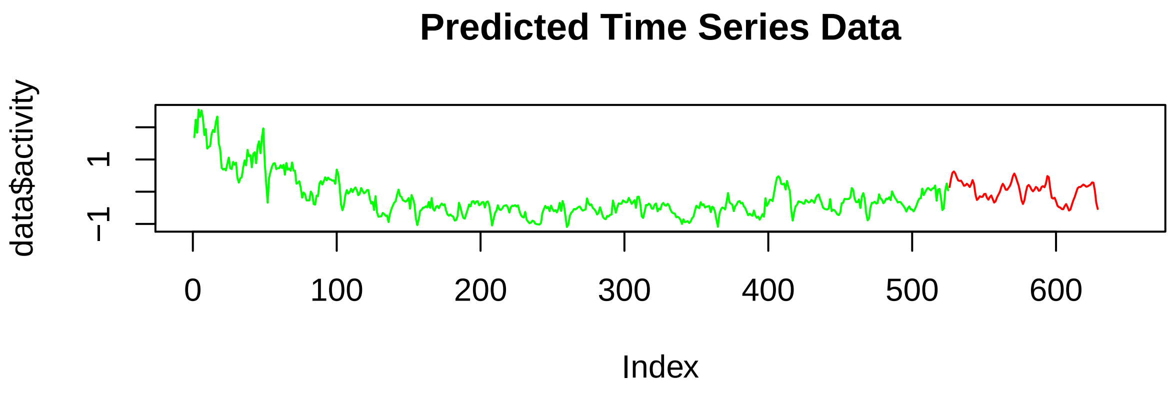
In conclusion, for the **detrending-via-LOESS & Sinusodial Fit, ARIMA modeling** method, the model we choose is ARIMA(0,0,1). Furthermore, we will choose this method to actually forecast the next two years given the training set.

---

4. Forecasting

We first fit an ARIMA(0,0,1) model to the residuals after sinusodial fit, and use that time series model to forecast the next 104 residuals. Then, we fit a sinusodial fit on the next 526 -> 629 observations using the frequencies we found via periodogram, and adding the two results in predicted LOESS fit's residuals. Lastly, we predict the overall mean trend via parametric fitting, and adding the predicted LOESS fit residuals + predicted LOESS fitted values = predicted fitted values for log.activity.

Finally, we convert the predicted values to actual activity values, and plotting predicted values.



Appendix

Question 5

```r
data <- read.csv("q5/q5_train.csv", stringsAsFactors = F)
data["Xt"] <- seq(1:nrow(data))
data['log.activity'] <- log(data$activity + 2)
plot(data$Xt, data$activity, main = "Initial Plot", type = "l")
```

```r
## 2a) Detrending via Differencing
diff.data <- diff(diff(data$log.activity, differences = 2), lag = 52)
acf(diff.data, lag.max = 500,
    main = "ACF of 2nd-Order + Seasonal Differencing")

adf.test(diff.data, alternative = c('stationary'), k = 52)
```

```r
## 2a) Detrending via Smoothing and Sinusodial Fitting
#plot(data$log.activity, type = "l",
#    main = "Log-Transformed Time Series Data, with LOESS Fit")
param.fit <- loess(log.activity ~ Xt, data = data,
                   control = loess.control(surface = "direct"),
                   span = 0.75, degree = 1)
#lines(param.fit$fitted, col = "red")
```

```r
acf(param.fit$residuals, lag.max = 500,
    main = "ACF Plot of LOESS Fit Residuals")
```

```r
periodos <-
  abs(fft(param.fit$residuals)/sqrt(length(param.fit$residuals)))^2
names(periodos) <- (1:length(periodos) - 1) / length(param.fit$residuals)
end.periodos <- ceiling(nrow(data) / 2)

plot(as.numeric(names(periodos[2:end.periodos])), periodos[2:end.periodos],
     main = "Periodogram of Residuals after LOESS Fit", type = "l",
     xlab = "frequencies", ylab = "Periodogram Value")
```

```r
1/as.numeric(names(sort(periodos[2:end.periodos], decreasing = T)[1:5]))
```

```r
coeffs <- sort(periodos[2:end.periodos], decreasing = T)[1:5]
freqs <- as.numeric(names(coeffs))

#Fitting sinusoid functions based on these frequencies
sinusoid <- function(start, end, freqs){
  t <- start:end
  df <- matrix(NA, nrow = length(t), ncol = length(freqs) * 2)
  for(i in 1:length(freqs)){
    df[ , 2*i] <- cos(2 * pi * freqs[i] * t)
    df[ , (2*i-1)] <- sin(2*pi*freqs[i]*t)
  }
  return(as.data.frame(df))
}
```

```r
season.fit.df <- sinusoid(start = 1, end = length(param.fit$residuals),
                          freqs = freqs)
season.fit.df["y"] <- param.fit$residuals
season.fit <- lm(y ~., data = season.fit.df)


# plot(data$log.activity, type = "l",
#      main = "Fitted Trend Line after LOESS and Seasonal Fit, 5 Freqs")
# lines(param.fit$fitted + season.fit$fitted.values, type = "l",
#       col = "red")
```

```r
acf(season.fit$residuals, lag.max = 500,
    main = "ACF of Sinusodial Fit Residuals, 5 Freqs")

sinusoid <- function(start, end, freqs){
  t <- start:end
  df <- matrix(NA, nrow = length(t), ncol = length(freqs) * 2)
  for(i in 1:length(freqs)){
    df[ , 2*i] <- cos(2 * pi * freqs[i] * t)
    df[ , (2*i-1)] <- sin(2*pi*freqs[i]*t)
  }
  return(as.data.frame(df))
}
num.periods <- floor(0.08*nrow(data))
coeffs <- sort(periodos[2:end.periodos], decreasing = T)[1:num.periods]
freqs <- as.numeric(names(coeffs))
season.fit.df <- sinusoid(start = 1, end = nrow(data),  freqs = freqs)
season.fit.df["y"] <- param.fit$residuals
season.fit <- lm(y ~., data = season.fit.df)

acf(season.fit$residuals, lag.max = 500,
    main = paste("ACF of Sinusodial Fit Residuals,", num.periods,"Freqs"))



adf.test(season.fit$residuals, alternative = "s", k = 52)
```

```r
## 3a.1) Determining ARIMA models based on ACF, PACF
par(mfrow = c(1,2))
acf(diff.data, lag.max = 200, main = "ACF of Differenced Data")
pacf(diff.data, lag.max = 200, main = "PACF of Differenced Data")
```

```r
## 3a.2) Model Diagnostics - Local External Validity
icTest = function(ts, p, d, q, P=0, D = 0, Q=0){
  aicmatrix = matrix(NA, (p+1), (q+1))
  bicmatrix = matrix(NA, (p+1), (q+1))
  for(a in 0:p){
    for(b in 0:q){
      model <- tryCatch(arima(ts, order = c(a,d,b),
                              seasonal = list(order = c(P,D,Q), period = 52)),
                        simpleError = function(e) e)
      if(inherits(model, "simpleError")){
        model <- tryCatch(arima(ts, order = c(a,d,b),
                                seasonal = list(order = c(P,D,Q), period = 52),
                                 method = "ML"),
                           simpleError = function(e) e)
      }
      if(inherits(model, "simpleError")){
        next
      }
      aicmatrix[a+1, b+1] = model$aic
      bicmatrix[a+1, b+1] = BIC(model)
    }

  }
  return(list(aic = aicmatrix, bic = bicmatrix))
}

#Due to long running time, load in previously saved
#ic.matrix <- icTest(data$log.activity, p = 3, d = 2, q = 3, P = 1, D = 1, Q = 1)
load("q5/finalSARicmat.RData")

#AIC Matrix
ic.matrix[[1]]

#BIC Matrix
#ic.matrix[[2]]

scorecard <- as.data.frame(matrix(NA, ncol = 5, nrow = 4))
colnames(scorecard) <- c("model","aic", "bic", "liv", "cv")
scorecard[ ,1] <- c("arima223", "arima023", "arima021", "arima022")

aic.scores <- c(ic.matrix[[1]][3, 4], ic.matrix[[1]][1, 4], ic.matrix[[1]][1, 2],
                ic.matrix[[1]][1, 3])
aic.scores <- aic.scores / min(aic.scores)

bic.scores <- c(ic.matrix[[2]][3, 4], ic.matrix[[2]][1, 4], ic.matrix[[2]][1, 2],
                ic.matrix[[2]][1, 3])
bic.scores <- bic.scores / min(bic.scores)

scorecard[ ,2] <- aic.scores
scorecard[ ,3] <- bic.scores
```

```r
## 3a.3) Model Diagnostics - Local Internal Validity
#ARIMA(2,2,3) X SARIMA(1,1,1)[52]
# sarima1 <- arima(data$log.activity, order = c(2,2,3),
#                  seasonal = list(order = c(1,1,1), period = 52))
# # acf(sarima1$residuals, lag.max = 500,
# #     main = "ACF of Residuals of ARIMA(1,1,1) X ARIMA(0,0,1)[52]")
#
#
# #ARIMA(0,2,3) X SARIMA(1,1,1)[52]
# sarima2 <- arima(data$log.activity, order = c(0,2,3),
#                  seasonal = list(order = c(1,1,1), period = 52))
# # acf(sarima2$residuals, lag.max = 500,
# #     main = "ACF of Residuals of ARIMA(3,2,1) X ARIMA(1,0,1)[52]")
#
# #ARIMA(0,2,1) X SARIMA(1,1,1)[52]
# sarima3 <- arima(data$log.activity, order = c(0,2,1),
#                  seasonal = list(order = c(1,1,1), period = 52))
# # acf(sarima3$residuals, lag.max = 500,
# #     main = "ACF of Residuals of ARIMA(0,2,2) X ARIMA(1,0,1)[52]")
#
# #ARIMA(0,2,2) X SARIMA(1,1,1)[52]
# sarima4 <- arima(data$log.activity, order = c(0,2,2),
#                  seasonal = list(order = c(1,1,1), period = 52))

#To Save time
load("q5/finalSARmodels.RData")
# all.models <- list("sarima1" = sarima1, "sarima2" = sarima2,
#                     "sarima3" = sarima3, "sarima4" = sarima4)
all.iv <-
  lapply(all.models, function(model){
    return(Box.test(model$residuals, lag = 104, fitdf = length(model$coef)))
  })

p.values <-
  sapply(all.models, function(model){
    return(Box.test(model$residuals, lag = 104, fitdf = length(model$coef))$p.value)
  })

scorecard[ ,4] <- p.values / max(p.values)

p.values
```

```r
## 3a.4) Model Diagnostics - General External Validity
tsCV <- function(ts, order, sorder = c(0,0,0), type, k){
  all_mses <- c()
  for(i in (10-k-1):8){
    train <- ts[1:(i*52), ]
    test <-  ts[(i*52+1):((i+2)*52), ]
    if(type == "resids"){
      param.fit <- loess(log.activity ~ Xt, data = train,
                         control = loess.control(surface = "direct"),
                         span = 0.75, degree = 1)

      periodos <-
        abs(fft(param.fit$residuals)/sqrt(length(param.fit$residuals)))^2
      names(periodos) <- (1:length(periodos) - 1) / length(param.fit$residuals)
      end.periodos <- ceiling(nrow(train) / 2)
      num.periods <- floor(0.08*nrow(train))
      coeffs <- sort(periodos[2:end.periodos], decreasing = T)[1:num.periods]
      freqs <- as.numeric(names(coeffs))
      season.fit.df <- sinusoid(start = 1, end = i*52,  freqs = freqs)
      season.fit.df["y"] <- param.fit$residuals
      season.fit <- lm(y ~., data = season.fit.df)

      sresids.model <- arima(season.fit$residuals, order = order,
                             seasonal = list(order = sorder, period  = 52))

      sresids.fcast <- predict(sresids.model, n.ahead = 104)
      s.fitted <- predict(season.fit,
                          newdata = sinusoid(start = (i*52+1), end = (i+2)*52,
                                             freqs = freqs))
      p.fitted <- predict(param.fit,
                          newdata = data.frame(Xt = test$Xt))
      predicted.val <- p.fitted + s.fitted + sresids.fcast$pred
    }
    if(type == "diff"){
      arima.model <- arima(train$log.activity, order = order,
                           seasonal = list(order = sorder, period = 52),
                           method = "CSS-ML")
      arima.fcast <- predict(arima.model, n.ahead = 104)
      predicted.val <- arima.fcast$pred
    }
    mses <- (test$activity - (exp(predicted.val) - 2))^2
    all_mses <- c(all_mses, mean(mses))
  }
  return(all_mses)
}
#
# sarima1.mse <- tsCV(ts = data, order = c(2,2,3), sorder = c(1,1,1),
#                     type = "diff", k  = 4)
# sarima2.mse <- tsCV(ts = data, order = c(0,2,3), sorder = c(1,1,1),
#                     type = "diff", k = 4)
# sarima3.mse <- tsCV(ts = data, order = c(0,2,1), sorder = c(1,1,1),
#                     type = "diff", k = 4)
# sarima4.mse <- tsCV(ts = data, order = c(0,2,2), sorder = c(1,1,1),
#                     type = "diff", k = 4)
#To Save time again
load("q5/finalSARmses.RData")

scorecard$cv <- min(c(mean(sarima1.mse), mean(sarima2.mse), mean(sarima3.mse),
                      mean(sarima4.mse))) /
  c(mean(sarima1.mse), mean(sarima2.mse), mean(sarima3.mse),
                  mean(sarima4.mse))
```

```
## 3a.5) Model Diagnostics - Final Score
scorecard["LEV_mean"] <- (scorecard$aic + scorecard$bic) / 2
weights <- c(0.2, 0.5, 0.3)
scorecard["total_score"] <- as.matrix(scorecard[ ,c(4,5,6)]) %*% weights

scorecard
```

```
## 3b.1) Determining ARIMA models based on ACF, PACF
par(mfrow = c(1,2))
acf(season.fit$residuals, lag.max = 200, main = "ACF of Seasonal Fit Residuals")

pacf(season.fit$residuals, lag.max = 200, main = "PACF of Seasonal Fit Residuals",
     ylim = c(-0.4,0.4))
```

```
## 3b.2) Model Diagnostics - Local External Validity
#AIC, BIC Matrix
sfit.ic.matrix <- icTest(season.fit$residuals, p = 3, d = 0, q = 3, P = 0, D = 0, Q = 0)

#AIC Matrix
```

```
#sfit.ic.matrix[[1]]

#BIC Matrix
#sfit.ic.matrix[[2]]


sfit.scorecard <- as.data.frame(matrix(NA, ncol = 5, nrow = 4))
colnames(sfit.scorecard) <- c("model","aic", "bic", "liv", "cv")
sfit.scorecard[ ,1] <- c("arima201", "arima202", "arima001", "arima102")


aic.scores <- c(sfit.ic.matrix[[1]][3, 2], sfit.ic.matrix[[1]][3, 3], sfit.ic.matrix[[1]][1, 2], sfit.i
aic.scores <- aic.scores / min(aic.scores)

bic.scores <- c(sfit.ic.matrix[[2]][3, 2], sfit.ic.matrix[[2]][3, 3], sfit.ic.matrix[[2]][1, 2], sfit.i
bic.scores <- bic.scores / min(bic.scores)

sfit.scorecard[ ,2] <- aic.scores
sfit.scorecard[ ,3] <- bic.scores
```

```
## 3b.3) Model Diagnostics - Local Internal Validity

#ARIMA(2,0,1)
arima1 <- arima(season.fit$residuals, order = c(2,0,1),
                 seasonal = list(order = c(0,0,0), period = 52))
#acf(arima1$residuals, lag.max = 500,
#    main = "ACF of Residuals of ARIMA(2,0,2)")

#ARIMA(2,0,2)
arima2 <- arima(season.fit$residuals, order = c(2,0,2),
                 seasonal = list(order = c(0,0,0), period = 52))
#acf(arima2$residuals, lag.max = 500,
#    main = "ACF of Residuals of ARIMA(3,0,1)")

#ARIMA(0,0,1)
arima3 <- arima(season.fit$residuals, order = c(0,0,1),
                 seasonal = list(order = c(0,0,0), period = 52))
#acf(sarima3$residuals, lag.max = 500,
#    main = "ACF of Residuals of ARIMA(0,0,2)")

arima4 <- arima(season.fit$residuals, order = c(1,0,2),
                seasonal = list(order = c(0,0,0), period = 52))

#Internal Validity Box.Test
all.models <- list("arima1" = arima1, "arima2" = arima2,
                   "arima3" = arima3, "arima4" = arima4)
all.iv <-
  lapply(all.models, function(model){
    return(Box.test(model$residuals, lag = 104, fitdf = length(model$coef)))
  })

#all.iv

sfit.p.values <- sapply(all.iv, function(test){
```

```r
  return(test$p.value)
})

sfit.scorecard[ ,4] <- sfit.p.values / max(sfit.p.values)
sfit.p.values
```

```r
## 3b.4) Model Diagnostics - General Internal Validity
sfit.mse1 <- tsCV(ts = data, order = c(2,0,1), sorder = c(0,0,0),
                  type = "resids", k= 4)
sfit.mse2 <- tsCV(ts = data, order = c(2,0,2), sorder = c(0,0,0),
                  type = "resids", k= 4)
sfit.mse3 <- tsCV(ts = data, order = c(0,0,1), sorder = c(0,0,0),
                  type = "resids", k= 4)
sfit.mse4 <- tsCV(ts = data, order = c(1,0,2), sorder = c(0,0,0),
                  type = "resids", k= 4)

sfit.scorecard$cv <- min(c(mean(sfit.mse1), mean(sfit.mse2), mean(sfit.mse3),
                      mean(sfit.mse4))) /
  c(mean(sfit.mse1), mean(sfit.mse2), mean(sfit.mse3),
                mean(sfit.mse4))
```

```r
## 3b.5) Model Diagnostics - Final Score
sfit.scorecard["LEV_mean"] <- (sfit.scorecard$aic + sfit.scorecard$bic) / 2
weights <- c(0.2, 0.5, 0.3)
sfit.scorecard["total_score"] <- as.matrix(sfit.scorecard[ ,c(4,5,6)]) %*% weights
sfit.scorecard
```

```r
# Forecasting
final.arima <- arima(season.fit$residuals, order = c(0,0,1),
                     seasonal = list(order = c(0,0,0), period = 52))
arima.fcast <- predict(final.arima, n.ahead = 104)

final.s.fitted <- predict(season.fit,
                          newdata = sinusoid(start = (526), end = 629,
                                             freqs = freqs))
final.p.fitted <- predict(param.fit, newdata = data.frame(Xt = 526:629))
final.predicted.val <-
  as.numeric(arima.fcast$pred + final.s.fitted + final.p.fitted)

final.predicted.val <- exp(final.predicted.val) - 2

plot(data$activity, type = "l", col = "green",
     main = "Predicted Time Series Data", xlim = c(0, 650))
lines(526:629, final.predicted.val, col = "red")
write(final.predicted.val, file = "q5/Q5_Jong_Lee_25344865.txt", sep = ",",
ncol = 1)
```

Question 1

```r
#Exploratory Data Analysis
data <- read.csv("q1/q1_train.csv", stringsAsFactors = F) #Note it is weekly data
data["Xt"] <- seq(1:nrow(data))
plot(data$Xt, data$activity, main = "Initial Plot", type = "l",
     ylab = "Activity", xlab = "Time")
```

```
local_maxes <- sapply(seq(1, nrow(data), by = 50), function(x){
  if(x + 50 > nrow(data)){
    return(which.max(data$activity[x:nrow(data)]) + x - 1)
  }
  else{
    return(which.max(data$activity[x:(x+50)]) + x - 1)
  }
})

#What's the average indice difference until local max happens again?
mean(diff(local_maxes))
```

```
#Log Transform
data["log.activity"] <- log(2 + data$activity)
plot(data$log.activity, type = "l", main = "Log+2-Transformed Activity")
```

```
#Differencing twice
trenddiff.logdata <- diff(data$log.activity)
acf(diff(trenddiff.logdata), lag.max = 500,
    main = "ACF of 2nd-Order Differenced Data")
```

```
#Seasonal Differencing after first differencing
acf(diff(trenddiff.logdata, lag = 52), lag.max= 500,
    main = "ACF of Lag-1 Differenced, then Seasonality Lag-52 Differenced Data")
```

```
#Parametric Fitting
param.fit <- loess(log.activity ~ Xt, data = data,
                   control = loess.control(surface = "direct"),
                   span = 0.75, degree = 1)
plot(data$log.activity, type = "l", main = "Parametric Mean-Trend Fitting")
lines(param.fit$fitted, col = "red")

plot(param.fit$residuals, type = "l",
     main = "Residuals of Parametric Fitting to Remove Linear Trend")
```

```
#Periodic, small number
periodos <- abs(fft(param.fit$residuals)/sqrt(length(param.fit$residuals)))^2
names(periodos) <- (1:length(periodos) - 1) / length(param.fit$residuals)
end.periodos <- ceiling(length(periodos) / 2)

plot(as.numeric(names(periodos))[2:end.periodos], periodos[2:end.periodos],
     type = "l", main = "Periodogram of Residuals after Parametric Fit",
     ylab = "Periodogram Value", xlab = "Frequency")
```

```
#Top three frequencies; note need to -1 because going from 1->n, change to 0->n-1
#Also we only go through only half of the frequencies, because the other half is simply the conjugate p
#Also don't look at frequency = 0, because that is not helpful

coeffs <- sort(periodos[2:end.periodos], decreasing = T)[1:3]
freqs <- as.numeric(names(coeffs))

#Fitting sinusoid functions based on these frequencies
sinusoid <- function(start, end, freqs){
  t <- start:end
```

```r
  df <- matrix(NA, nrow = length(t), ncol = length(freqs) * 2)
  for(i in 1:length(freqs)){
    df[ , 2*i] <- cos(2 * pi * freqs[i] * t)
    df[ , (2*i-1)] <- sin(2*pi*freqs[i]*t)
  }
  return(as.data.frame(df))
}
season.fit.df <- sinusoid(start = 1, end = length(param.fit$residuals),
                          freqs = freqs)
season.fit.df["y"] <- param.fit$residuals
season.fit <- lm(y ~., data = season.fit.df)


plot(data$log.activity, type = "l",
     main = "Fitted Trend Line after Parametric and Seasonal Fit, 3 Freqs")
lines(param.fit$fitted + season.fit$fitted.values, type = "l",
      col = "red")
```

```r
#ACF of seasonal fit residuals
acf(season.fit$residuals, lag.max = 500,
    main = "ACF of Seasonally Fitted Residuals, 3 Freqs")
```

```r
coeffs <- sort(periodos[2:end.periodos], decreasing = T)[1:floor(0.13*525)]
freqs <- as.numeric(names(coeffs))

season.fit.df <- sinusoid(start = 1, end = length(param.fit$residuals),
                          freqs = freqs)
season.fit.df["y"] <- param.fit$residuals
season.fit <- lm(y ~., data = season.fit.df)

#Fitted Values Plot
plot(data$log.activity, type = "l",
     main = "Fitted Trend Line after Parametric and Seasonal Fit, 34 Freqs")
lines(param.fit$fitted + season.fit$fitted.values, type = "l",
      col = "red")

#ACF
acf(season.fit$residuals, lag.max = 500,
    main = "ACF of Seasonally Fitted Residuals, 34 Freqs")
```

```r
#ACF and PACF of Seasonal Fit
par(mfrow = c(1,2))

#ACF and PACF
acf(season.fit$residuals, lag.max = 500)
pacf(season.fit$residuals, lag.max = 500)
```

```r
adf.test(season.fit$residuals, alternative = "stationary", k = 52)
```

```r
ic.matrix <- icTest(season.fit$residuals,
                    p = 3, d = 0, q = 3, P = 0, D = 0, Q = 0)
#AIC Matrix
ic.matrix[[1]]

#BIC Matrix
```

```r
ic.matrix[[2]]
```

```r
#ARIMA(1,0,2)
arima1 <- arima(season.fit$residuals, order = c(1,0,2))
acf(arima1$residuals, lag.max = 500,
    main = "ACF of Residuals of ARIMA(1,0,2)")

#ARIMA(2,0,2)
arima2 <- arima(season.fit$residuals, order = c(2,0,2))
acf(arima2$residuals, lag.max = 500,
    main = "ACF of Residuals of ARIMA(2,0,2)")

#ARIMA(0,0,1)
arima3 <- arima(season.fit$residuals, order = c(0,0,1))
acf(arima3$residuals, lag.max = 500,
    main = "ACF of Residuals of ARIMA(0,0,1)")
```

```r
all.models <- list("arima1" = arima1, "arima2" = arima2,
                   "arima3" = arima3)
all.iv <-
  lapply(all.models, function(model){
    return(Box.test(model$residuals, lag = 104, fitdf = length(model$coef)))
    })

all.iv
```

```r
tsCV <- function(ts, order, sorder = c(0,0,0), type, k){
  all_mses <- c()
  for(i in (10-k-1):8){
    train <- ts[1:(i*52), ]
    test <-  ts[(i*52+1):((i+2)*52), ]
    if(type == "resids"){
      param.fit <- loess(log.activity ~ Xt, data = train,
                         control = loess.control(surface = "direct"),
                         span = 0.75, degree = 1)

      periodos <-
        abs(fft(param.fit$residuals)/sqrt(length(param.fit$residuals)))^2
      names(periodos) <- (1:length(periodos) - 1) / length(param.fit$residuals)
      end.periodos <- ceiling(nrow(train) / 2)
      num.periods <- floor(0.13*nrow(train))
      coeffs <- sort(periodos[2:end.periodos], decreasing = T)[1:num.periods]
      freqs <- as.numeric(names(coeffs))
      season.fit.df <- sinusoid(start = 1, end = i*52,  freqs = freqs)
      season.fit.df["y"] <- param.fit$residuals
      season.fit <- lm(y ~., data = season.fit.df)

      sresids.model <- arima(season.fit$residuals, order = order,
                             seasonal = list(order = sorder, period  = 52))

      sresids.fcast <- predict(sresids.model, n.ahead = 104)
      s.fitted <- predict(season.fit,
                          newdata = sinusoid(start = (i*52+1), end = (i+2)*52,
                                             freqs = freqs))
      p.fitted <- predict(param.fit,
                          newdata = data.frame(Xt = test$Xt))
      predicted.val <- p.fitted + s.fitted + sresids.fcast$pred
    }
    if(type == "diff"){
      arima.model <- arima(train$log.activity, order = order,
                           seasonal = list(order = sorder, period = 52),
                           method = "CSS-ML")
      arima.fcast <- predict(arima.model, n.ahead = 104)
      predicted.val <- arima.fcast$pred
    }
    mses <- (test$activity - (exp(predicted.val) - 2))^2
    all_mses <- c(all_mses, mean(mses))
  }
  return(all_mses)
}
```

```r
#ARIMA(1,0,2)
arima1.mses <- tsCV(data, order = c(1,0,2), sorder = c(0,1,0),
                    k = 4, type = "resids")
mean(arima1.mses)

#ARIMA(2,0,2)
arima2.mses <- tsCV(data, order = c(2,0,2), sorder = c(0,1,0),
                    k = 4, type = "resids")
mean(arima2.mses)

#ARIMA(0,0,1)
arima3.mses <- tsCV(data, order = c(0,0,1), sorder = c(0,1,0),
                    k = 4, type = "resids")
mean(arima3.mses)
```

```r
final.arima <- arima(season.fit$residuals, order = c(2,0,2),
                     seasonal = list(order = c(0,1,0), period = 52))
arima.fcast <- predict(final.arima, n.ahead = 104)

final.s.fitted <- predict(season.fit,
                          newdata = sinusoid(start = (526), end = 629,
                                             freqs = freqs))
final.p.fitted <- predict(param.fit, newdata = data.frame(Xt = 526:629))
final.predicted.val <-
  as.numeric(arima.fcast$pred + final.s.fitted + final.p.fitted)
final.predicted.val <- exp(final.predicted.val) - 2

plot(data$activity, type = "l", col = "green",
     main = "Predicted Time Series Data", xlim = c(0, 650))
lines(526:629, final.predicted.val, col = "red")
write(final.predicted.val, file = "q1/Q1_Jong_Lee_25344865.txt", sep = ",",
ncol = 1)
```

Question 2

```
# q2
data2 <- read.csv("~/stat153/q2_train.csv", as.is = TRUE)
head(data2)
ts2 = ts(data2$activity)
data2$Date = (as.numeric(as.Date.character(data2$Date))-12414)/7
data2$log.ts2 = log(data2$activity+4)
log.ts2 = ts(data2$log.ts2)

#paramatric fit
# q2
#1. find the span for loess
findspan = function(span){
  loess_mse = numeric(length = 4)
  for(i in 1:4){

    fit = loess(log.ts2~Date, data = data2[(1:(52*(i+4))),],
                degree = 0, span = span,control=loess.control(surface="direct"))
    pred = predict(fit, newdata = data.frame(Date = seq((52*(i+4)+1),(52*(i+6)),length = 104)))
    loess_mse[i] = mean(((exp(pred)-4) - data2[((52*i+4)+1):(52*(i+6)),2])^2)
  }
  print(rbind(cbind(loess_mse), mean = mean(loess_mse)))
}

findspan(0.1)
findspan(0.2)
findspan(0.3)
findspan(0.4)
findspan(0.5)
findspan(0.6)

#2. fit parametric model
param.fit = loess(log.ts2~Date, data = data2,degree = 0, span = 0.2,control=loess.control(surface="dire
summary(param.fit)
plot(log.ts2, main = "log.ts2"); lines(param.fit$fitted, col = "red")

#3. look at the residuals
rsd = param.fit$residuals
plot(rsd)
adf.test(rsd)
acf(rsd, lag.max = 550); abline(v = c(52,104,156), col ="red")
pacf(rsd, lag.max = 550)
acf(diff(rsd, 52), lag.max = 250); abline(v = c(52,104,156), col ="red")
pacf(diff(rsd, 52), lag.max = 250)


#cross validation
# q2
CrossValidation_loess <- function(order,sorder){
  MSE = numeric(length = 4)
  for(i in 1:4){
    traindt = data2[(1:(52*(4+i))),3]
    testdt = data2[((52*(4+i)+1):(52*(4+i+2))),3]

    trend_fit = loess(log.ts2~Date, data = data2[(1:(52*(4+i))),],control=loess.control(surface="direct
    resid_fit = arima(trend_fit$residuals, order = order, seasonal = list(order = sorder, period = 52))

    trend_fcst = predict(trend_fit, newdata = data.frame(Date = seq((52*(4+i)+1),(52*(4+i+2)),length = 
    resid_fcst = predict(resid_fit, n.ahead = 104)


    MSE[i] = mean(((exp(trend_fcst + resid_fcst$pred)-4)-
                     (exp(testdt)-4))^2)
  }
  print(rbind(cbind(MSE), mean = mean(MSE)))
}
```

Question 3

```
### Q3.R
### Author: Jong Ha Lee
### Last Updated: 04/12/2017
setwd("~/Desktop/School/STAT153/midterm2")

### 1. Exploratory Data Analysis
data <- read.csv("q3/q3_train.csv", stringsAsFactors = F) #Note it is weekly data
data["Xt"] <- seq(1:nrow(data))

# Initial Plotting
plot(data$Xt, data$activity, main = "Initial Plot", type = "l")

#Log-Transformed Plot
data["log.activity"] <- log(2 + data$activity)
plot(data$log.activity, type = "l", main = "Log+2-Transformed Activity")

#Square-root transformation
data['sqrt.activity'] <- sqrt(data$activity + 2)
plot(data$sqrt.activity, type = "l")


#Based on plot, dividing into different sections to find local maximas
local_maxes <- sapply(seq(1, nrow(data), by = 50), function(x){
  if(x + 50 > nrow(data)){
    return(which.max(data$activity[x:nrow(data)]) + x - 1)
  }
  else{
    return(which.max(data$activity[x:(x+50)]) + x - 1)
  }
})

#What's the average indice difference until local max happens again?
```

```r
mean(diff(local_maxes))



##### 2. Transformation of Data and Removing Mean Structure #####
param.fit <- loess(log.activity ~ Xt, data = data,
                   span = 0.5, degree = 2,
                   control = loess.control(surface = "direct"))

plot(data$log.activity, type = "l")
lines(param.fit$fitted, col = "red")
plot(param.fit$residuals, type = 'l')
acf(param.fit$residuals, lag.max = 500)

#As expected, see a lot of seasonality. Let's remove seasonality by sinusodial fitting.
sinusoid <- function(start, end, freqs){
  t <- start:end
  df <- matrix(NA, nrow = length(t), ncol = length(freqs) * 2)
  for(i in 1:length(freqs)){
    df[ , 2*i] <- cos(2 * pi * freqs[i] * t)
    df[ , (2*i-1)] <- sin(2*pi*freqs[i]*t)
  }
  return(as.data.frame(df))
}
periodos <-
  abs(fft(param.fit$residuals)/sqrt(length(param.fit$residuals)))^2
names(periodos) <- (1:length(periodos) - 1) / length(param.fit$residuals)
end.periodos <- ceiling(nrow(data) / 2)
num.periods <- floor(0.3*nrow(data))
coeffs <- sort(periodos[2:end.periodos], decreasing = T)[1:num.periods]
freqs <- as.numeric(names(coeffs))
season.fit.df <- sinusoid(start = 1, end = nrow(data),  freqs = freqs)
season.fit.df["y"] <- param.fit$residuals
season.fit <- lm(y ~., data = season.fit.df)

plot(season.fit$fitted.values + param.fit$fitted, type = "l")
lines(data$log.activity, col = "red")
acf(season.fit$residuals, lag.max = 500)

#Take Differencing
acf(diff(season.fit$residuals, lag = 52),lag.max=500)

#Checking Staitonarity of Seasonal Fit Residuals
adf.test(diff(season.fit$residuals, lag = 52), alternative = "s", k = 52) #ok
adf.test(season.fit$residuals, alternative = "s", k = 52)

# ACF and PACF of Seasonal Fit Residuals
acf(diff(season.fit$residuals, lag = 52), lag.max = 200)
pacf(diff(season.fit$residuals, lag = 52), lag.max = 200)

icTest = function(ts, p, d, q, P=0, D = 0, Q=0){
  aicmatrix = matrix(NA, (p+1), (q+1))
  bicmatrix = matrix(NA, (p+1), (q+1))
  for(a in 0:p){
    for(b in 0:q){
      model <- tryCatch(arima(ts, order = c(a,d,b),
                              seasonal = list(order = c(P,D,Q), period = 52)),
                        simpleError = function(e) e)
      if(inherits(model, "simpleError")){
        model <- tryCatch(arima(ts, order = c(a,d,b),
                                seasonal = list(order = c(P,D,Q), period = 52),
                                method = "ML"),
                          simpleError = function(e) e)
      }
      if(inherits(model, "simpleError")){
        next
      }
      aicmatrix[a+1, b+1] = model$aic
      bicmatrix[a+1, b+1] = BIC(model)
    }

  }
  return(list(aic = aicmatrix, bic = bicmatrix))
}

sfit.ic.matrix <- icTest(season.fit$residuals, p = 3, d = 0, q = 3, P = 1, D = 1, Q = 0)


sfit.ic2.matrix <- icTest(season.fit$residuals, p = 3, d = 0, q = 3, P = 0, D = 0, Q = 0)
arima4 <- arima(season.fit$residuals, order = c(2,0,2),
                seasonal = list(order = c(0,0,0), period = 52))
arima5 <- arima(season.fit$residuals, order = c(1,0,3),
                seasonal = list(order = c(0,0,0), period = 52))
arima6 <- arima(season.fit$residuals, order = c(0,0,1),
                seasonal = list(order = c(0,0,0), period = 52))
new.all.models <- list(arima4, arima5, arima6)
new.all.iv <-
  lapply(new.all.models, function(model){
    return(Box.test(model$residuals, lag = 104, fitdf = length(model$coef)))
  })

new.all.iv
#Best Models for only seasonal differencing: ARIMA(1,0,2), ARIMA(2,0,3), ARIMA(0,0,1)
#Best Models for both non-seasonal and seasonal differencing: ARIMA(0,1,1), ARIMA(0,1,2), ARIMA(1,1,1)

#ARIMA(2,0,3)
arima1 <- arima(season.fit$residuals, order = c(0,1,2),
                seasonal = list(order = c(1,1,0), period = 52))
acf(arima1$residuals, lag.max = 500,
    main = "ACF of Residuals of ARIMA(2,0,3)")

#ARIMA(2,0,1)
arima2 <- arima(season.fit$residuals, order = c(0,1,1),
                seasonal = list(order = c(1,1,0), period = 52))
acf(arima2$residuals, lag.max = 500,
    main = "ACF of Residuals of ARIMA(2,0,1)")
```

```r
#ARIMA(1,0,2)
arima3 <- arima(season.fit$residuals, order = c(2,1,3),
                seasonal = list(order = c(1,1,0), period = 52))
acf(arima3$residuals, lag.max = 500,
    main = "ACF of Residuals of ARIMA(1,0,2)")

#Internal Validity Box.Test
all.models <- list("arima1" = arima1, "arima2" = arima2,
                   "arima3" = arima3)
all.iv <-
  lapply(all.models, function(model){
    return(Box.test(model$residuals, lag = 104, fitdf = length(model$coef)))
  })

all.iv

plot(data$log.activity, type = "l")
lines(param.fit$fitted, col = "red")
lines(417:520, p.fitted, col = "blue")

tsCV <- function(ts, order, sorder = c(0,0,0), type, k){
  all_mses <- c()
  for(i in (10-k-1):8){
    train <- ts[1:(i*52), ]
    test <-  ts[(i*52+1):((i+2)*52), ]
    if(type == "resids"){
      param.fit <- loess(log.activity ~ Xt, data = train,
                         control = loess.control(surface = "direct"),
                         span = 0.6, degree = 1)

      periodos <-
        abs(fft(param.fit$residuals)/sqrt(length(param.fit$residuals)))^2
      names(periodos) <- (1:length(periodos) - 1) / length(param.fit$residuals)
      end.periodos <- ceiling(nrow(train) / 2)
      num.periods <- floor(0.08*nrow(train))
      coeffs <- sort(periodos[2:end.periodos], decreasing = T)[1:num.periods]
      freqs <- as.numeric(names(coeffs))
      season.fit.df <- sinusoid(start = 1, end = i*52,  freqs = freqs)
      season.fit.df["y"] <- param.fit$residuals
      season.fit <- lm(y ~., data = season.fit.df)

      sresids.model <- arima(season.fit$residuals, order = order,
                             seasonal = list(order = sorder, period = 52))

      sresids.fcast <- predict(sresids.model, n.ahead = 104)
      s.fitted <- predict(season.fit,
                          newdata = sinusoid(start = (i*52+1), end = (i+2)*52,
                                             freqs = freqs))
      p.fitted <- predict(param.fit,
                          newdata = data.frame(Xt = test$Xt))
      predicted.val <- p.fitted + s.fitted + sresids.fcast$pred
    }
    if(type == "diff"){
      arima.model <- arima(train$log.activity, order = order,
                           seasonal = list(order = sorder, period = 52),
                           method = "CSS-ML")
      arima.fcast <- predict(arima.model, n.ahead = 104)
      predicted.val <- arima.fcast$pred
    }
    mses <- (test$activity - (exp(predicted.val) - 2))^2
    all_mses <- c(all_mses, mean(mses))
  }
  return(all_mses)
}

sfit.all_mses <- tsCV(ts = data, order = c(1,1,3), sorder = c(1, 1 ,0),
                      type = "resids", k= 4)


final.arima <- arima(season.fit$residuals, order = c(1,1,3),
                     seasonal = list(order = c(1,1,0), period = 52))

arima.fcast <- predict(final.arima, n.ahead = 104)

final.s.fitted <- predict(season.fit,
                          newdata = sinusoid(start = (526), end = 629,
                                             freqs = freqs))
final.p.fitted <- predict(param.fit, newdata = data.frame(Xt = 526:629))
final.predicted.val <-
  as.numeric(arima.fcast$pred + final.s.fitted + final.p.fitted)

final.predicted.val <- exp(final.predicted.val) - 2

plot(data$activity, type = "l", col = "green",
     main = "Predicted Time Series Data", xlim = c(0, 650))
lines(526:629, final.predicted.val, col = "red")
write(final.predicted.val, file = "q3/Q3_Jong_Lee_25344865.txt", sep = ",",
      ncol = 1)


#####*** OUTPUT OF PREDICTION RESULTS ***#####
#Assume predictions are in a vector called preds
preds <- sample(1:12312, 100, replace = F) #Random for now
write(preds, file = "Q1_Jong_Lee_25344865.txt", sep = ",",
      ncol = length(preds)) #As required format
```

Question 4

```r
# q4
data4 <- read.csv("~/stat153/q4_train.csv", as.is = TRUE)
head(data4)
ts4 = ts(data4$activity)
data4$Date = (as.numeric(as.Date.character(data4$Date))-12414)/7
```

```
# q4
#1. lm fit
lm.fit = lm(activity ~ Date + I(Date^2), data = data4 )
summary(lm.fit)

#2. look at the residuals of lm.fit
residual = lm.fit$residuals
adf.test(residual)
acf(residual, lag.max = 250)
pacf(residual, lag.max = 250)

#3. it arima models to the residuals
re.fit = arima( residual ,order = c(1,0,2), seasonal = list( order = c(0,1,1), period = 52))
acf(re.fit$residuals, lag.max = 250)



# mdoel comparing based on AIC and BIC
aicTest = function(ts, p, d, q, P, D, Q){
  aic = matrix(NA, (p+1), (q+1))
  for(a in 0:p){
    for(b in 0:q){
      aic[a+1, b+1] = arima(ts, order = c(a,d,b),seasonal = list(order = c(P,D,Q), period = 52))$aic
    }
  }
  return(aic)
}

bicTest = function(ts, p,d,q, P,D,Q){
  bic = matrix(NA, (p+1), (q+1))
  for(a in 0:p){
    for(b in 0:q){
      bic[a+1, b+1] = BIC(arima(ts, order = c(a,d,b), seasonal = list(order = c(P,D,Q), period = 52)))
    }
  }
  return(bic)
}

# goodness of fit check
plot(re.fit$residuals)
re.fit$aic;BIC(re.fit)
acf(re.fit$residuals, lag.max = 250)
Box.test(re.fit$residuals, lag = 20);Box.test(re.fit$residuals, lag = 104)

# q4
CrossValidation_lm <- function(order,sorder){
  MSE = numeric(length = 4)
  for(i in 1:4){

    traindt = data4[(1:(52*(4+i))),2]
    testdt = data4[((52*(4+i)+1):(52*(4+i+2))),2]

    trend_fit = lm(activity~Date+I(Date^2),data = data4)
    resid_fit = arima(trend_fit$residuals, order = order, seasonal = list(order = sorder, period = 52))

    trend_fcst = predict(trend_fit, newdata = data.frame(Date = seq((52*(4+i)+1),(52*(4+i+2)),length = 
    resid_fcst = predict(resid_fit, n.ahead = 104)


    MSE[i] = mean(((trend_fcst + resid_fcst$pred)-testdt)^2)
  }
  print(rbind(cbind(MSE), mean = mean(MSE)))
}

# forecast
lm.pred = predict(lm.fit, newdata = data.frame(Date = seq(526,629,length = 104)))
re.pred = predict(re.fit, n.ahead = 104)$pred

predict_q4 = lm.pred + re.pred

plot(ts4, type = "l", xlim = c(1, 629), ylim = c(-2,5))
lines(x = 1:525, y = fitted(re.fit)+lm.fit$fitted.values, col = "green")
lines(x =  1:525, y = lm.fit$fitted.values, type = "l", col = "blue")
lines(x =  526:629, y = lm.pred, type = "l", col = "blue")
lines(x = 526:629, y = predict_q4, type = "l", col = "red" )

# output
write.table(predict_q4,sep = ",", col.names = FALSE,row.names = FALSE,file = "Q_BinyiCai_3032593123.txt
```