Sea Surface Temperature Prediction Mini-Project Report

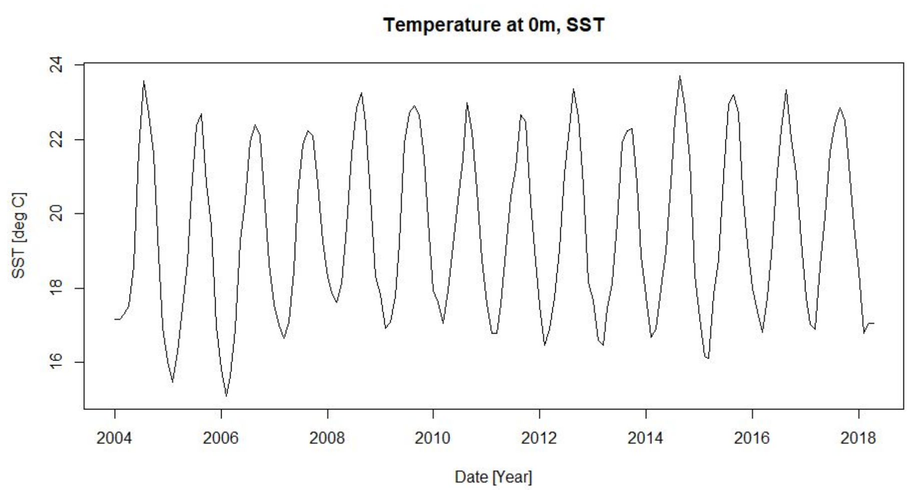
Chris Lee, Minghan Sun, Yishan Xu, Yege Zhang

March 15, 2020

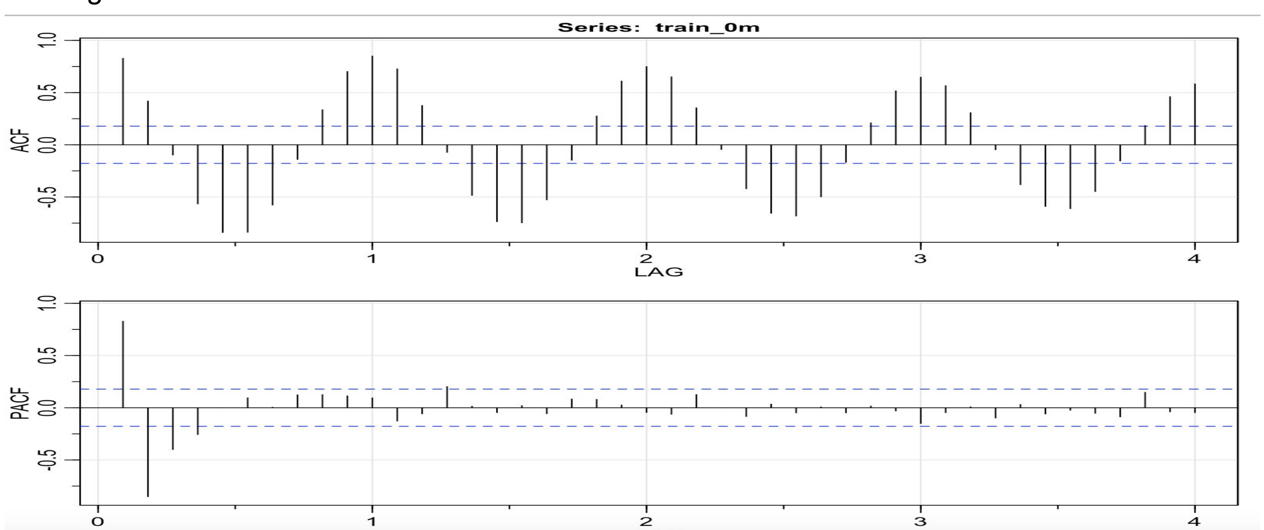
Objective (Yege)
To analyze the monthly sea temperature from 0 meters to 90 meters below the sea level, in order to predict sea surface temperature in the future.

Data (Yege)
For this project, we use the monthly sea temperature from 2014 to 2018, totaling 158 observations. According to the data, it has 33 days in one period, and 11 periods in a year.

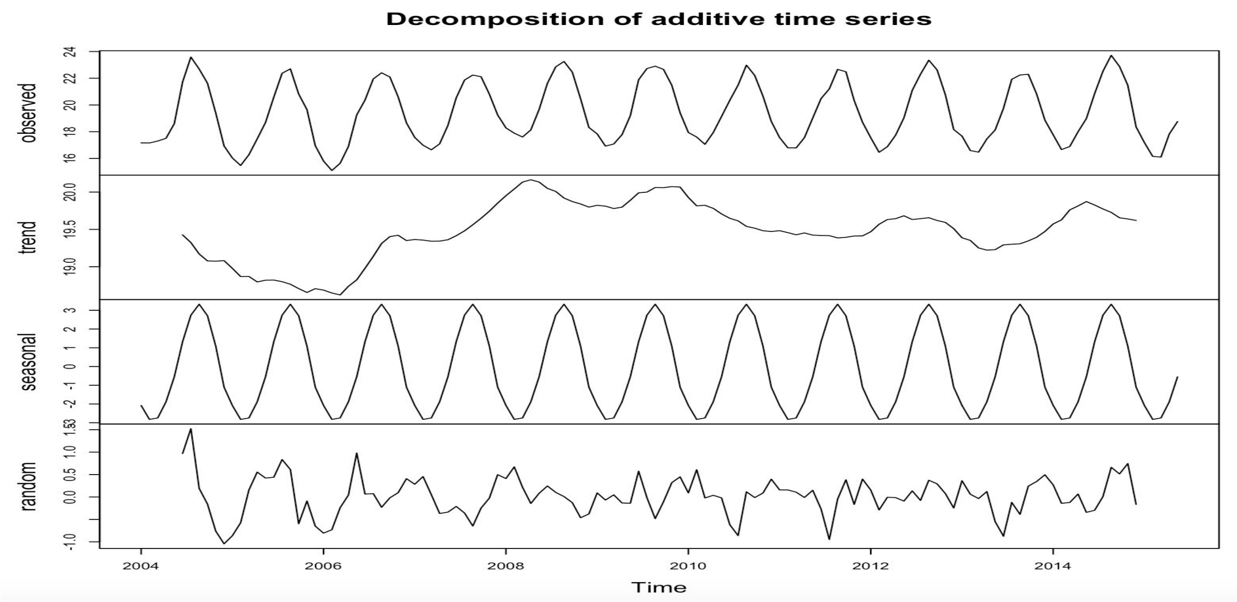
Data Exploration (Yege)
The plot below shows the original time series, which has the very obvious seasonality.



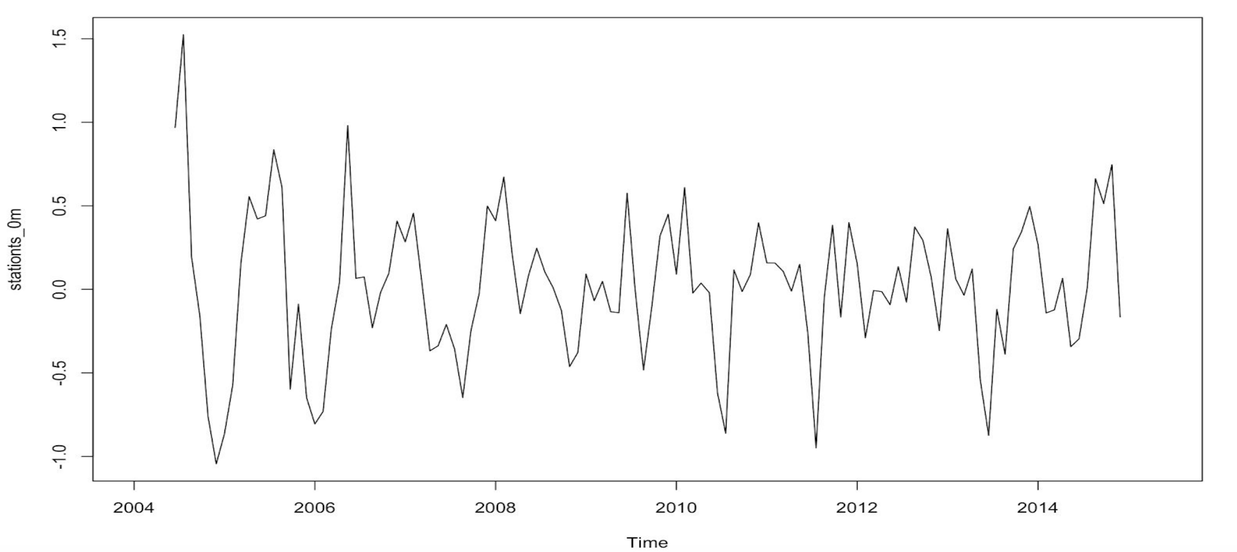
To further analyze, we plot the ACF and PACF. As we can see, the seasonality occurs in every time lag.



We then decompose the data into 3 parts: 1) trend, 2) seasonal,3) random. As we can see in the figure, our data has the upward and downward trend and the seasonality.



In order to have stationary data, we remove the trend and seasonality from the time series. However, we are still able to see the seasonality from the plot. To try to solve the issue, we decide to try the auto.arima function, which will be explained in the following model.



Model 1: SARIMA without Additional Variable (Minghan)

Ideally, we would like to find an optimal SARIMA model by constantly adjusting models until no correlation is found in residuals. However, the time series of sea surface temperature is more complicated than we thought. Manually trying and adjusting multiple models based on acf/pacf plot won't give a desired result. Ultimately, we turned to 'auto.arima' function for help, where we let the machine look for the best model that fits our sea surface temperature train series.

Below is the flow of how we find the appropriate SARIMA model based on train test.

Separating data as train/test:

```
#train test split
train_0m=ts(ndata$`0m`, start = c(2004,1,1), end = c(2015,5,15), frequency = 11)
test_0m=ts(ndata$`0m`, start = c(2015,6,17), end = c(2018,4,12), frequency = 11)
```

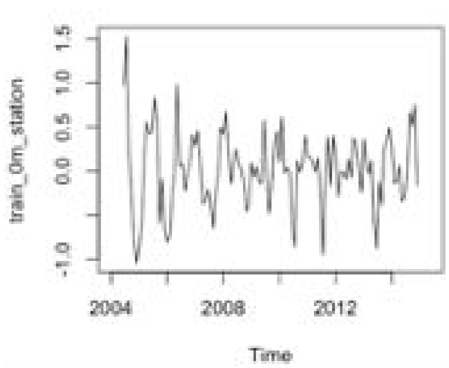
First 126 observations were put in the train test, last 32 observations were put in the test set.

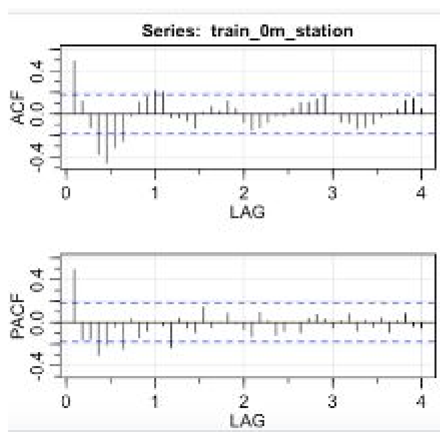
Trial 1: Use adjusted series to train models.

```
decompose<-decompose(train_0m)
train_0m_station<-train_0m-decompose$seasonal-decompose$trend
```

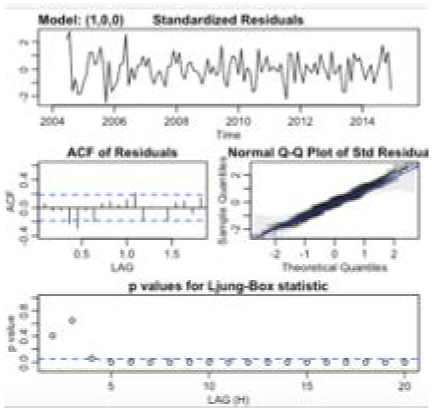
After we have taken out the trend and seasonality in the data, the series still displays uneven variance across the periods.

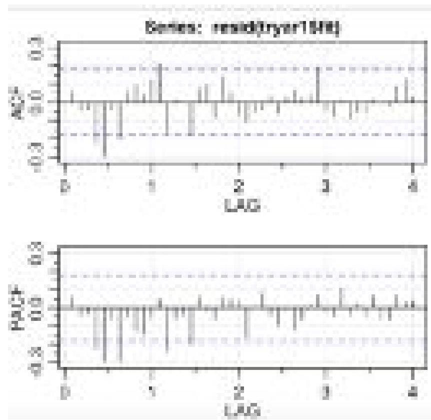
However, we decided to try AR1 model based on the acf/pacf graph below,



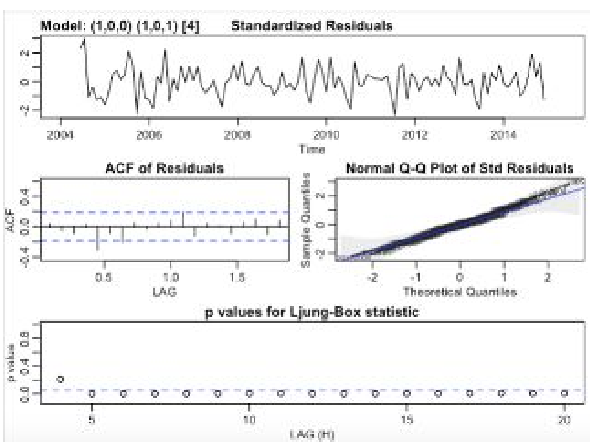


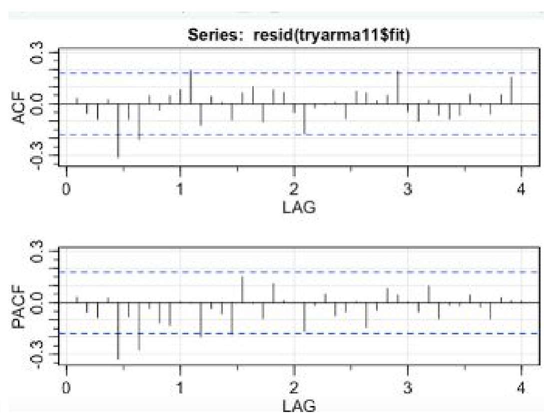
Diagnosis plot displayed below tells that there exists correlation among residuals starting from lag 4 .
Checking residual plot to find the appropriate model that fits residual, we have found no clear pattern, although both ACF/PACF start to stick out at lag=4.





Adding seasonal ARMA(1,1) with lag=4 to original mode. The diagnosis plot shown as below.

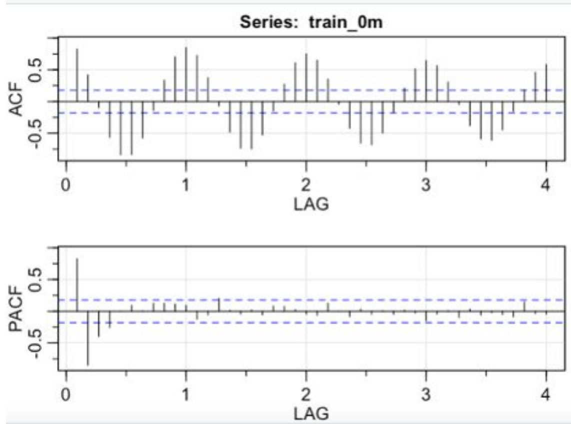




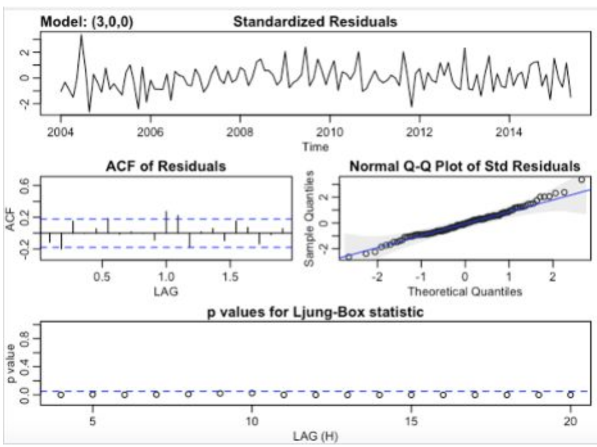
Q-statistic still shows correlation among residuals starting from lag=5. Residual plot for new model almost identical to previous model. After several 'try and error', no desired model was found.

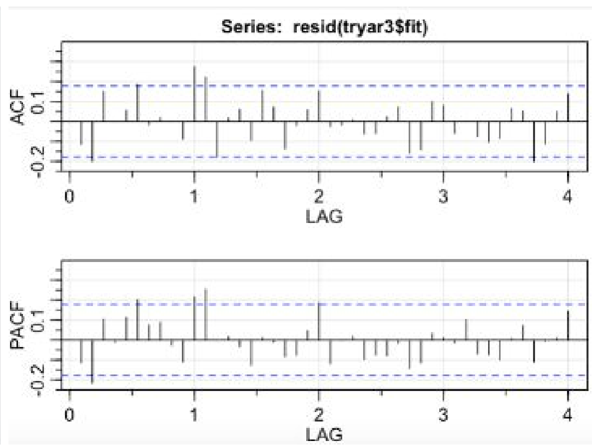
Trial 2: Use original series to fit models

According to the ACF/PACF graph below, AR(3)seems to be a good model to try first as pacf cut off at lag=3.

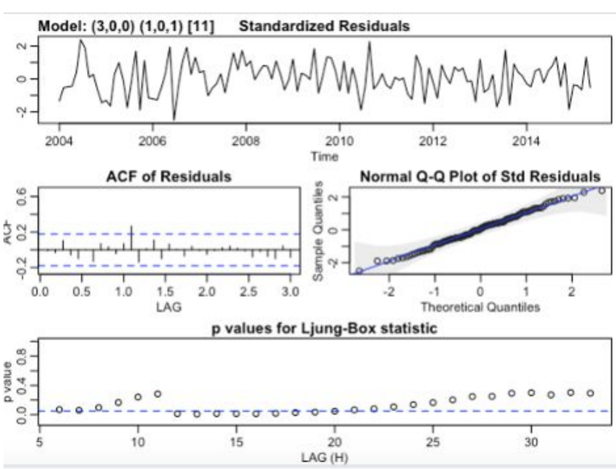


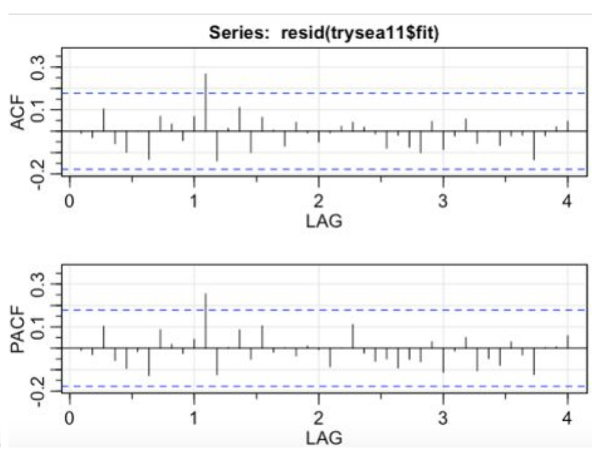
Diagnosis plot tells that there is correlation among residuals. Model needs to be adjusted.



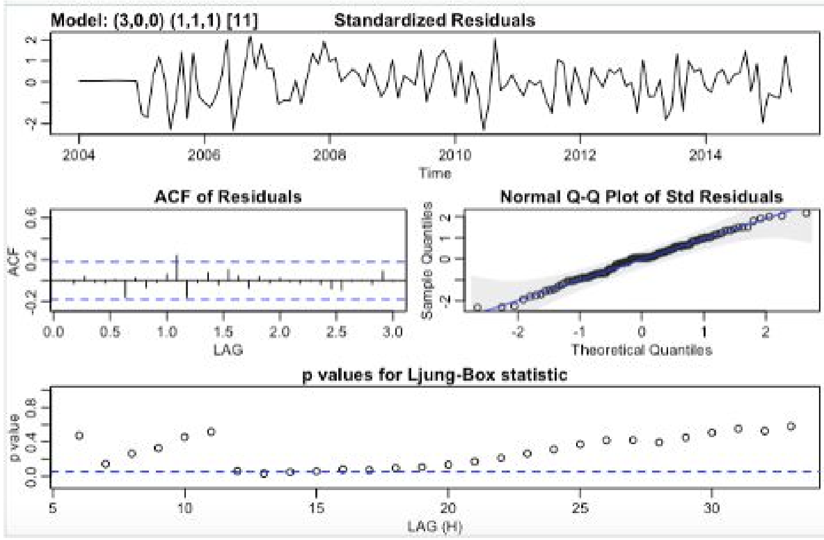


Since ACF/PACF stick out at lag=1 (11th period), adding seasonal ARMA(1,1) with s=11 to the original AR(3) may be appropriate. New diagnosis plot and residual check are displayed below.





Clear seasonality displayed, adding seasonal difference = 1 to the model.



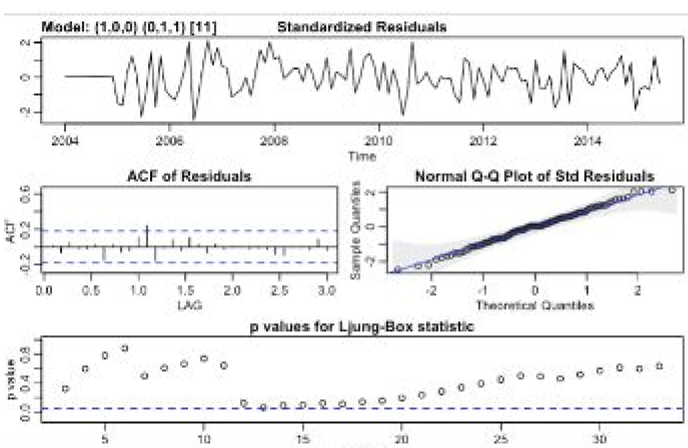
Model looks not bad, with great improvement compared to previous models. With a couple of p-value points almost below the blue line, we wanted to see if there is a better model than this. By adjusting several times, no model is better than the one above; SARIMA(3,0,0)*(1,1,1) S=11. Therefore, we decided to use 'auto.arima' function.

Trial 3: 'auto.arima' function

```
#Using auto arima to obtain the best model
fitauto_seasonal<-auto.arima(train_0m,seasonal=TRUE,trace=TRUE,test='kpss',ic='bic')
```

```
Best model: ARIMA(1,0,0)(0,1,1)[11]
```

By allowing seasonal component, auto.arima has selected SARIMA(1,0,0)*(0,1,1) S=11 as the best model based on the train set. Diagnosis plot is shown below.



P value plot looks good and better than the last model in general. In early lags, the p-value points are further to the blue line compared to the last model. No point is below the blue line. QQ plot is almost identical to the 45 degree line.

Now, we can forecast using SARIMA(1,0,0)*(0,1,1) S=11.

Forecast 1: using 'forecast' function

When using the 'forecast' function, we allow the model to use some predicted values to predict a certain stage. The MSPE of this method is equal to 22.18.

```
> mean((forecastedvalue$mean-test_0m)^2) #mspe=22.1746
[1] 22.1764
```

Forecast 2: using all the true value to perform one-step-ahead

In this method, we have created a function that will use all the true values prior to the predicted period to forecast. The MSPE of this method is equal to 21.99.

```
> mean((osapredictedvalue-test_0m)^2) #mspe=21.9856
[1] 21.9856
```

Conclusion: The MSPE from two forecast methods are very close. It indicates that, in this case, whether the series we use to perform one-step-ahead contains predicted value or not does not affect much on the prediction error. On average, using SARIMA without additional variables will give us mean squared predicted error of predicted sea surface temperature around 22.

Model 2: SARIMA with additional variables (Yishan)

Read the csv file into R, rename columns as “0m”, “10m”...

```
> data <- read.csv("gilbralter_time_series_r_2.csv")
> ndata <- data[,-c(1,3)]
> names(ndata) <- c("0m", "10m", "20m", "30m", "40m", "50m", "60m", "70m", "80m", "90m")
```

Split the data into training (first 80% observations) and the testing (remaining 20% observations). Set frequency equals to 11 because one period equals 33 days, and there are 11 periods in a year

```
> train_0m=ts(ndata$`0m`, start = c(2004,1,1), end = c(2015,5,15), frequency = 11)
> test_0m=ts(ndata$`0m`, start = c(2015,6,17), end = c(2018,4,12), frequency = 11)
>
```

As we can see from the initial model, the MSPE score is very high. Sea surface temperature in the past alone might not be able to accurately forecast the future sea surface temperature. There are other factors that impact the sea surface temperature. Thus, we added the exogenous variable to increase the prediction accuracy. The exogenous variable in this case is the temperature at 10m-90m below the sea surface.

```
> # adding addition variables without feature selection
> train_additional=ts(ndata[c(2:10)], start = c(2004,1,1), end = c(2015,5,15), frequency = 11)
> test_additional=ts(ndata[c(2:10)], start = c(2015,6,17), end = c(2018,4,12), frequency = 11)
>
```

Then, we utilized the auto.arima function to pick the best model. The best model turned out to be arima (0,1,0).

```
> fitauto_seasonal1<-auto.arima(train_0m,xreg=train_additional, seasonal=TRUE,trace=TRUE,test='kpss',ic='bic')

 Regression with ARIMA(2,1,2)(1,0,1)[11] errors : Inf
 Regression with ARIMA(0,1,0)            errors : 4.051812
 Regression with ARIMA(1,1,0)(1,0,0)[11] errors : 8.392856
 Regression with ARIMA(0,1,1)(0,0,1)[11] errors : 6.419628
 ARIMA(0,1,0)                                   : -0.7662402
 Regression with ARIMA(0,1,0)(1,0,0)[11] errors : 7.529331
 Regression with ARIMA(0,1,0)(0,0,1)[11] errors : 7.002356
 Regression with ARIMA(0,1,0)(1,0,1)[11] errors : 10.3817
 Regression with ARIMA(1,1,0)            errors : 5.697884
 Regression with ARIMA(0,1,1)            errors : 5.236153
 Regression with ARIMA(1,1,1)            errors : Inf

 Best model: Regression with ARIMA(0,1,0)            errors
```

With the best model picked, we used the forecast function to predict the sea surface temperature of next 32 months. Below is how the result looks.

```
> forecastedvalue1=forecast(fitauto_seasonal1,xreg=test_additional,h=32)
> forecastedvalue1
```

```
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2015.455       17.38864 17.12408 17.65319 16.98403 17.79324
2015.545       17.31676 16.94262 17.69090 16.74457 17.88896
2015.636       17.43723 16.97900 17.89545 16.73643 18.13802
2015.727       17.41792 16.88880 17.94703 16.60871 18.22712
2015.818       18.79294 18.20138 19.38451 17.88822 19.69766
2015.909       21.71286 21.06484 22.36089 20.72179 22.70393
2016.000       23.49975 22.79980 24.19970 22.42927 24.57023
2016.091       23.60324 22.85496 24.35152 22.45885 24.74763
2016.182       22.65849 21.86482 23.45215 21.44468 23.87230
2016.273       20.64776 19.81116 21.48436 19.36829 21.92723
2016.364       18.17316 17.29573 19.05060 16.83125 19.51508
2016.455       16.90580 15.98935 17.82225 15.50422 18.30739
2016.545       16.16461 15.21074 17.11848 14.70579 17.62343
2016.636       16.36347 15.37360 17.35335 14.84959 17.87736
2016.727       17.32547 16.30085 18.35009 15.75844 18.89249
2016.818       18.95617 17.89794 20.01439 17.33775 20.57458
2016.909       20.90323 19.81244 21.99402 19.23501 22.57145
2017.000       22.53833 21.41591 23.66074 20.82174 24.25491
2017.091       22.72026 21.56709 23.87343 20.95663 24.48388
2017.182       21.25678 20.07365 22.43991 19.44734 23.06622
2017.273       20.11404 18.90169 21.32638 18.25991 21.96816
2017.364       17.74546 16.50458 18.98634 15.84770 19.64322
2017.455       16.57032 15.30155 17.83909 14.62991 18.51073
2017.545       15.69027 14.39421 16.98632 13.70812 17.67241
2017.636       16.14639 14.82361 17.46917 14.12338 18.16941
2017.727       17.19401 15.84504 18.54299 15.13093 19.25709
2017.818       19.40085 18.02617 20.77552 17.29847 21.50323
2017.909       20.83899 19.43910 22.23889 18.69803 22.97995
2018.000       22.50521 21.08053 23.92988 20.32635 24.68406
2018.091       22.85289 21.40386 24.30193 20.63679 25.06900
2018.182       22.41432 20.94133 23.88730 20.16158 24.66706
2018.273       21.02429 19.52773 22.52084 18.73551 23.31307
```

The next step is to assess prediction accuracy. The MSPE is 0.3109 for ARIMAX, big improvement compared to initial model without exogenous variable

```
> mean((forecastedvalue1$mean-test_0m)^2)
[1] 0.3108831
>
```

Furthermore, we wanted to consider feature selection to see if we can improve prediction accuracy. We've run the linear regression, according to summary output, p-values of 10m, 80m, and 90m are low compared to other features.

```
> # adding addition variables with feature selection
> train<-ndata[1:126,]
> test<-ndata[127:158,]
> attach(train)
```

```
> linearreg =lm(`0m`~., data=train)
>
> summary(linearreg)

Call:
lm(formula = `0m` ~ ., data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-0.93251 -0.08856  0.00844  0.13463  0.46672

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.24818    0.58636  -2.129  0.03539 *
`10m`        1.36090    0.17045   7.984 1.14e-12 ***
`20m`       -0.17439    0.30943  -0.564  0.57412
`30m`       -0.25180    0.25254  -0.997  0.32081
`40m`        0.06504    0.16504   0.394  0.69426
`50m`        0.02976    0.19487   0.153  0.87887
`60m`       -0.12451    0.28781  -0.433  0.66609
`70m`        0.29390    0.25331   1.160  0.24833
`80m`       -0.64705    0.25798  -2.508  0.01352 *
`90m`        0.52371    0.16448   3.184  0.00187 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2411 on 116 degrees of freedom
Multiple R-squared:  0.9899,    Adjusted R-squared:  0.9891
F-statistic:  1262 on 9 and 116 DF,  p-value: < 2.2e-16
```

With selected features of 10m, 80m, and 90m, we reevaluated our model using auto.arima function, the best model is still arima(0,1,0)

```
> train_FeatureSelection<-ts(ndata[,c(2,9,10)], start = c(2004,1,1), end = c(2015,5,15), frequency = 11)
> test_FeatureSelection=ts(ndata[,c(2,9,10)], start = c(2015,6,17), end = c(2018,4,12), frequency = 11)
>
> fitauto_seasonal2<-auto.arima(train_0m,xreg=train_FeatureSelection, seasonal=TRUE,trace=TRUE,test='kpss',ic='bic')

 Regression with ARIMA(2,1,2)(1,0,1)[11] errors : Inf
 Regression with ARIMA(0,1,0)            errors : -3.999919
 Regression with ARIMA(1,1,0)(1,0,0)[11] errors : 3.532423
 Regression with ARIMA(0,1,1)(0,0,1)[11] errors : 3.181162
 ARIMA(0,1,0)                               : -8.823815
 Regression with ARIMA(0,1,0)(1,0,0)[11] errors : 0.1336801
 Regression with ARIMA(0,1,0)(0,0,1)[11] errors : 0.1310193
 Regression with ARIMA(0,1,0)(1,0,1)[11] errors : 4.95854
 Regression with ARIMA(1,1,0)            errors : -0.211681
 Regression with ARIMA(0,1,1)            errors : -0.4297958
 Regression with ARIMA(1,1,1)            errors : Inf

 Best model: Regression with ARIMA(0,1,0)            errors
```

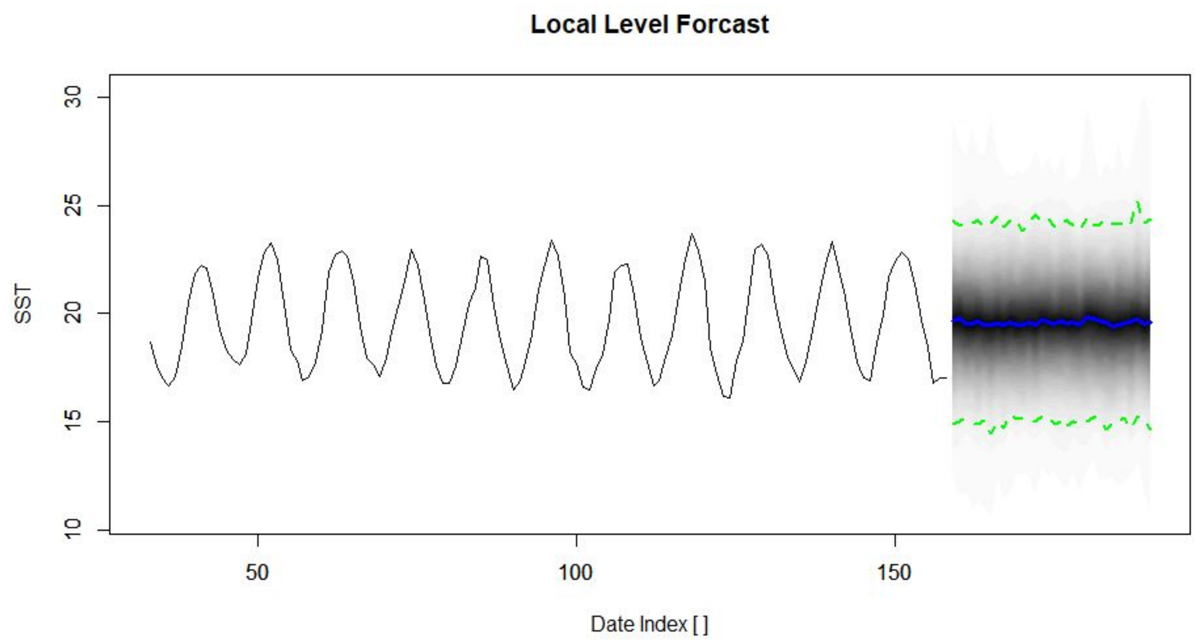
Again, we predicted sea surface temperature using forecast function, and assessed the MSPE. MSPE score turned out to be 0.2894. A slight improvement compared to the model without feature selection.

```
> forecastedvalue2=forecast(fitauto_seasonal2,xreg=test_FeatureSelection,h=32)
> mean((forecastedvalue2$mean-test_0m)^2)
[1] 0.2893556
>
```

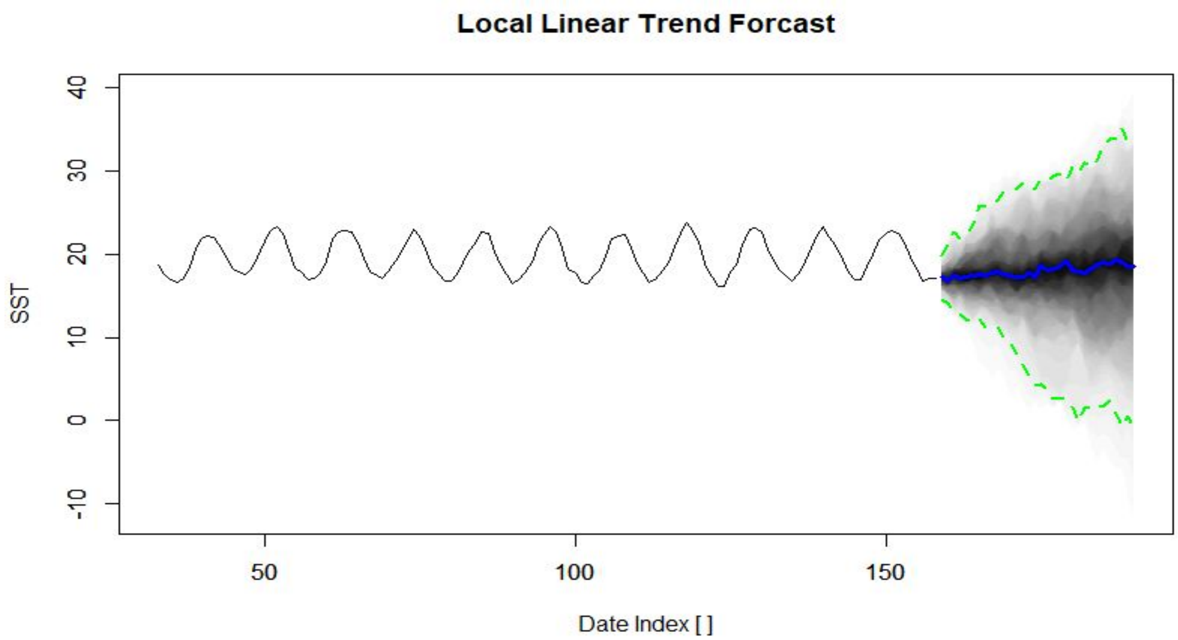
Model 3: Bayesian Structural Time Series (BSTS) (Chris)

BSTS is a time series regression method that uses dynamic linear models fit using Markov Chain Monte Carlo. With our data, we first started off with the most simple useful model in a structured time series and then worked our way up with increasing the complexity of the models. In the BSTS Models, the dark blue line is the median of the predictive distributions, the green lines indicate the prediction intervals (default is the 95% prediction interval), and the grey shaded area represents the posterior density distributions.

The most simple useful model in structured time series is the Local Level. BSTS Model of the Local Level forecasts is based around the average value of recent observations. The Local Level is also a random walk observed in noise.

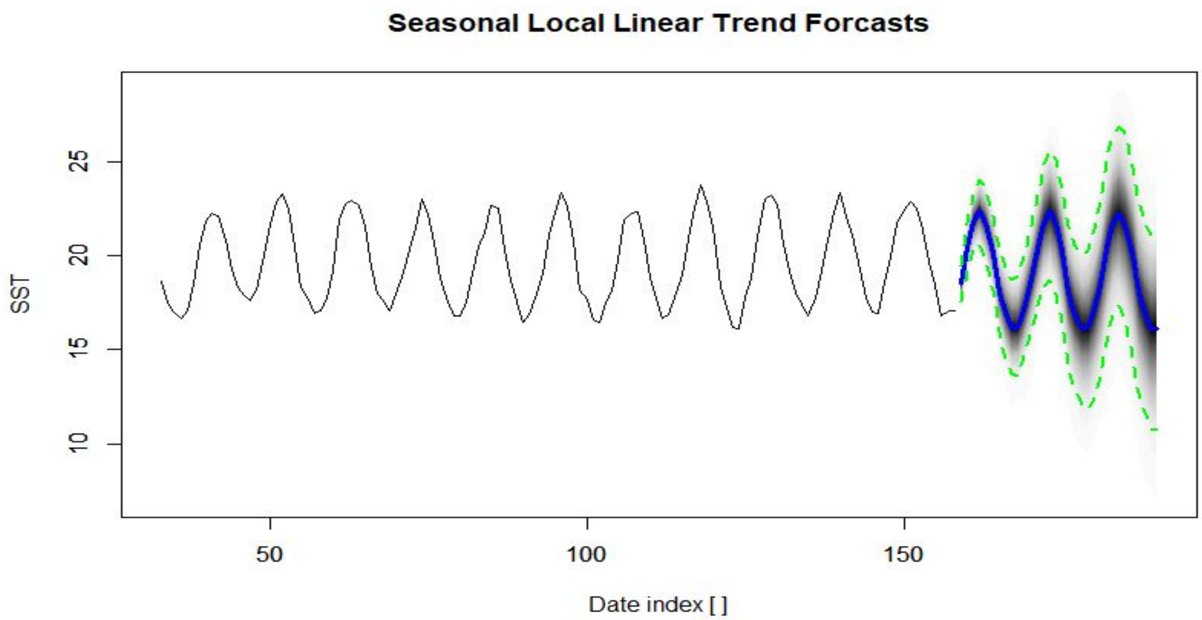


With the local level, we got a MSPE of 5.09. After attempting the local level model, we increased the complexity by adding a local linear trend component.



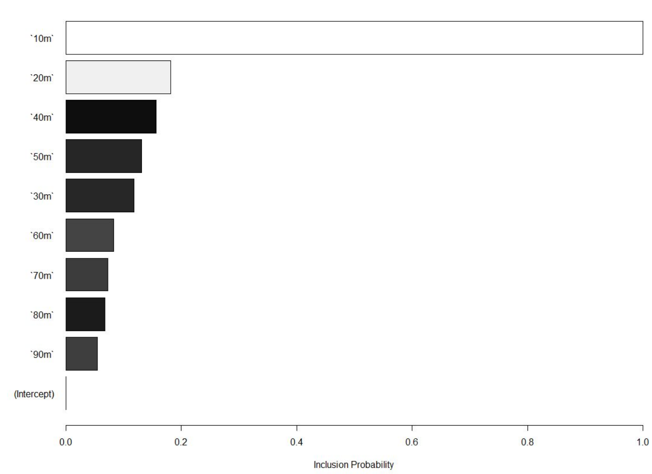
In our attempt of adding a local linear trend component, our MSPE score got worse, which makes sense. It was a MSPE of 9.955. As you can see in our data, we have an obvious seasonality. The Local Linear Trend model attempts to capture the trend at the tail ends of the time series. It is especially ideal for time series that have a particular direction and if we wanted the forecast to reflect on the past observations.

After adding on the local linear component, we then again increased our model's complexity by adding a seasonal component. With adding a seasonal component, we expect our model to fit pretty well.

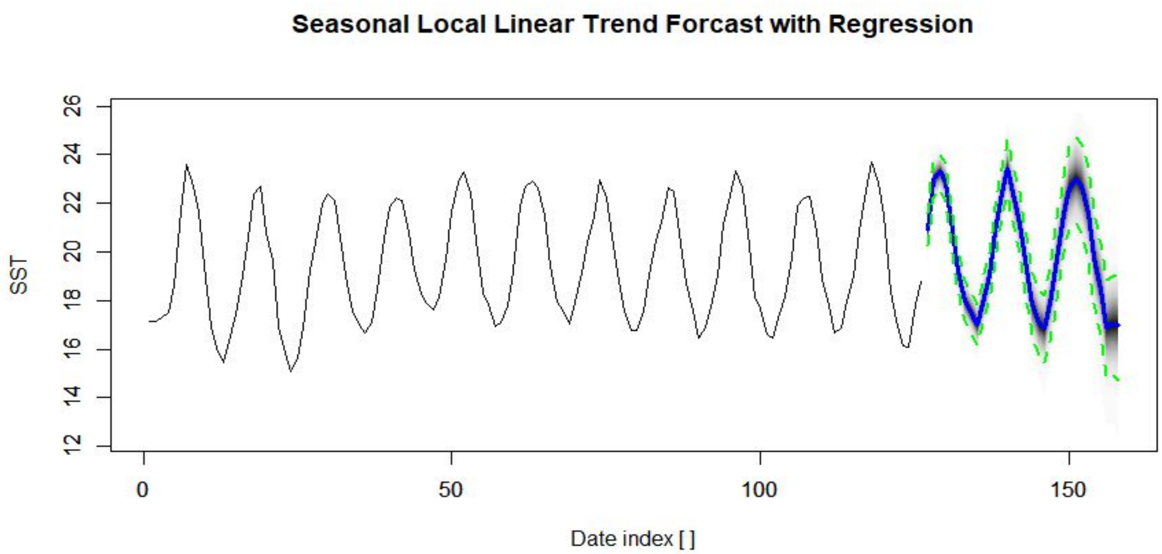


As we can see, after adding the seasonal component to make our forecasts seasonal, our new model is now much better than the previous two models since the two prior ones failed to characterize seasonality. We now have a MSPE of 2.25.

Now, after having a good model for our data, we wanted to attempt to make our model even better, and to do so, we created a regressional component by setting SST/0m, as our output, and the underlying layers (10m - 90m) as our features. We also look at the trend components of our data to let us know what may or may not be a good expected model size.



Looking at the trend components, the white bars represent positive betas, and the black bars represent negative betas for our regressional component. We see that 10m and 20m both are positive and have a high inclusion probability, so we expect that an expected model size of 1 and or 2 will be optimal. In our case, we set our expected model size to 1. The expected model size sets our spike and slab prior to having one spike, in other words, we are expecting our one depth to heavily influence the sea temperature, most likely the layer directly underneath (10m).



As we can see, adding a regressional component narrowed our prediction intervals improving our model. With the narrowed prediction intervals and improved performance, our new MSPE is now 0.024.

Conclusion (Yege)

The aim of the time series model is to predict the future values for the series by studying the past observations. In this project, we fitted the following three models, SARIMA model without including additional variable, SARIMA model with additional variable and BSTS. The SARIMA model performed much better after we added the additional variables. However, we only see a slight improvement when feature selection was applied. Among the most effective approaches for analyzing time series data, BSTS seasonal local linear trend forecast with regression is employed with the lowest MSPE and appropriate model was adaptively formed based on the given data.