Hong Kong Export **Forecast**

2020

ECON 4120
Elon Chan

Table of Content

Introduction

In this report, I will demonstrate steps by steps on how to model and forecast Hong Kong Export data (in billion) using the techniques learned in class. I will first start with a simple trend model, then adding seasonality, finally incorporating the element of cycles into the model. In the end, we will see a pretty good in-sample as well as out-of-sample fit for the data. And eventually, use the model fitted to forecast Hong Kong's export in year 2020.

Setup, Loss Function and Date Preparation

The data used in this report is extracted from Bloomberg, the biggest data provider in the world. The period of the monthly time series starts from December 1999 and end in November 2019, having 239 observations in total, which is the maximum amount of data by the time I accessed the Bloomberg terminal. All the data shown are in billion, Hong Kong Dollar.

I choose to use the R language as my tools to analyses Hong Kong Export. The reason for this is that the R language is one of the most wildly used languages for statistically works. There are thousands of packages that we can install upon need and a massive supportive community which are helpful to the completion of this report. To make sure all my works can be replicated and checked by other people, I have uploaded the raw data file and R source code to a GitHub repository. The version of R and packages I used are:

```
> R.version
               _
platform       x86_64-pc-linux-gnu
arch           x86_64
os             linux-gnu
system         x86_64, linux-gnu
status
major          3
minor          6.1
year           2019
month          07
day            05
svn rev        76782
language       R
version.string R version 3.6.1 (2019-07-05)
nickname       Action of the Toes
```

```
library(readxl) # for importing data
library(forecast)
library(lmtest) # for dw test
library(ggplot2) # for plot graph
```

The choice for loss function is the mean square error (MSE) as it is optimal-forecast as introduced in lecture slides.

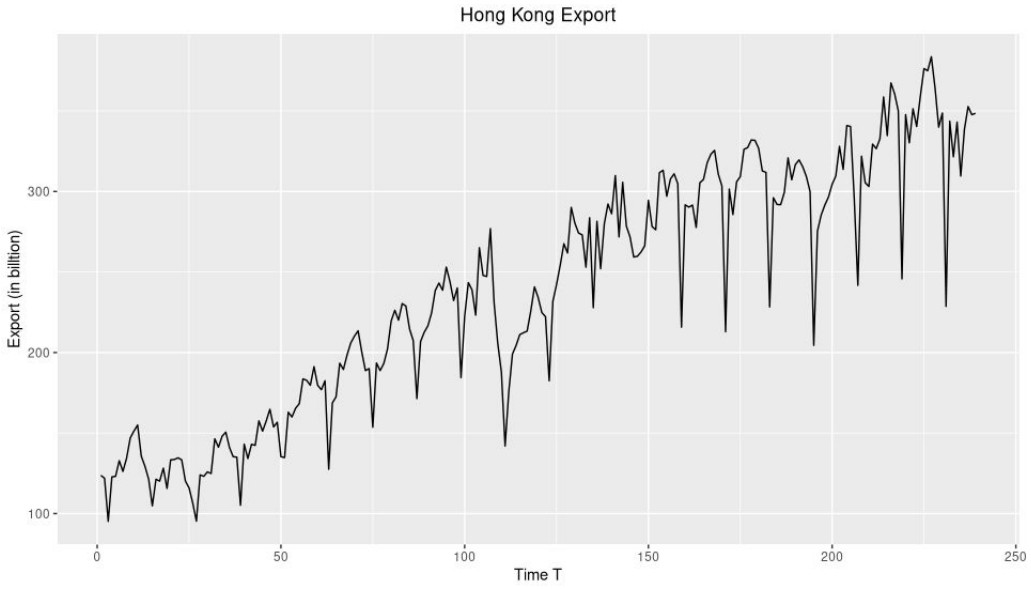
Before moving on into actual analysis, I would like to split the data into 2 piles, 80% as the training set and 20% as the testing set. This is because if we use all the data available to fit the model, then the problem of overfitting may occur and we will fail to select the model with the best forecasting ability. With overfitting, models' in-sample error is very small and satisfying but perform terribly in out-of-sample data. Therefore I will first use the training set to fit the model, then test the model's performance in the testing set.

```
hke <- read_excel("R/forecast/hke.xlsx", skip=5)
names(hke)[2] <- "hk_export" # Rs array starts at 1

hke$Date <- rev(hke$Date)
hke$seq <- seq(from=1, to = 239)
hke$hk_export <- rev(hke$hk_export)

n = round(length(hke$seq)*0.8)

train = head(hke, n)
test = tail(hke, -n)
```

Here I load the model into the data frame hke (skipping the first 5 header rows), which consist of the $date$ and the hk_export which are self-explanatory. the seq is the T in lecture slides. It is necessary because if we directly put $Date$ into the regression model, the result will not be interpretable. Now we are set for further analysis.

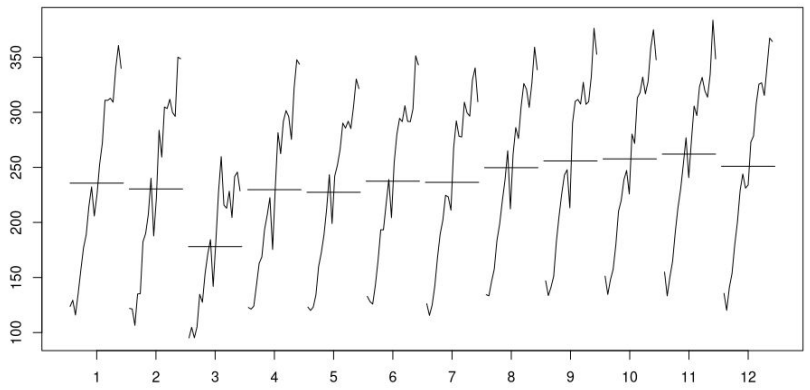
Data Exploration

Before we directly just into modelling, lets first take a look at the time series data we got and get a sense of it.



There are 5 characteristics.

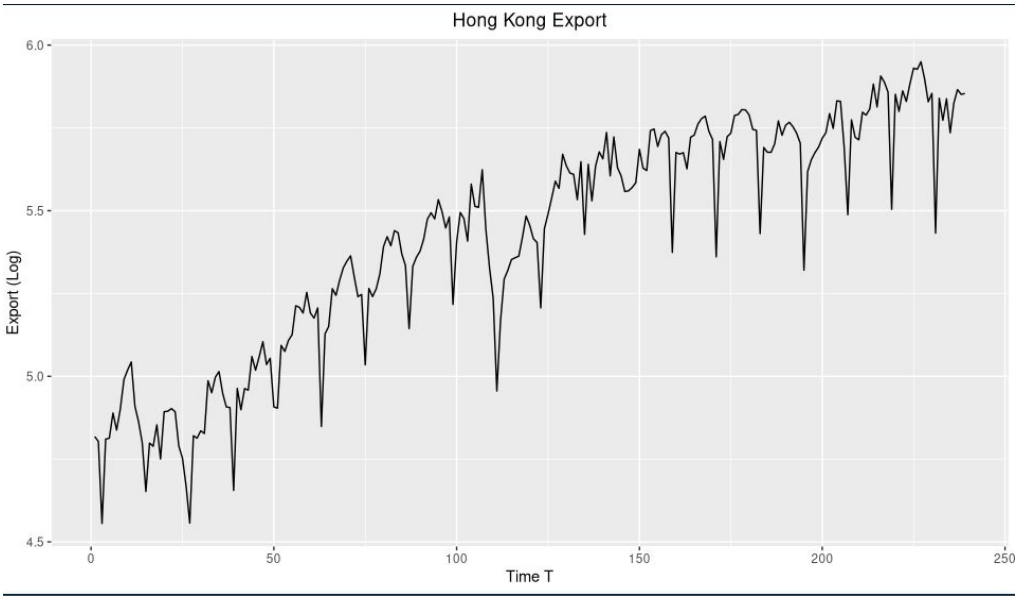
1. There is an upward trend in the data Which is not surprising as Hong Kong's economy has been growing steadily in the past 20 years.
2. The trend is a linear trend, we can model that easily.
3. Structural break exists, there is a sharp slope increase around $T = 120 \; to \; T = 140$, which correspond to the rapid recovery from the 2008 financial crisis.
4. Strong seasonal effect. As we can see there is a sharp decrease in export every February. This is reasonable since the lunar new year holiday is usually in February



As we can see, every February the export drops by 40% on average. Here February is represented by 3 because my data starts in December 1999.

5. Changing seasonal magnitude. The magnitude of each years' seasonal effect is increasing.

Characteristics 3 and 5 are gonna be problematic as they can not be handled easily with $trend, seasonal\ and\ cycle$ model. Luckily, we can try to take the logarithmic form of the original time series to smaller the difference introduced by the structural break and changing seasonal magnitude. Here is what the plot looks like:



However, after seeing the plot I have decided not to use the logarithmic form of the data but instead use the original one. There are 3 reasons

1. The structural break still exists. The same structural break incurred by the financial crisis still very visible by eyes, so the log technique is not that useful here
2. The seasonal magnitude is still changing, same as before the differences in magnitude is again obvious.
3. After taking the logarithmic for, observe that the trend is not linear anymore but having a curve like shapes. Which will take at least 1 extra parameter to model the non-linear trend.

Having considered the extra complexity and low effectiveness for taking log to the data, by the $KISS\ Principle$ I have decided to use the original data for building model.

Trend

The first part of the modelling is to model trend. Since we are dealing with a linear trend, I would choose to use the basic linear model.

$$y_t = \beta_0 + \beta_1 TIME_t + \varepsilon_t$$

In R, the command to build a model is pretty simple.

```
18  # Linear Model with Trend Line
19  linear_model <- lm(hk_export ~ seq, data=train)
20  plot(train$seq, train$hk_export, type="l", main="Hong Kong Export", xlab="TIME", ylab="Export", xlim=c(0,24
```

Here we fit the model with 1 variable, which is $TIME$. Remember we are first to do the fitting by training data. And here is the model's summary:

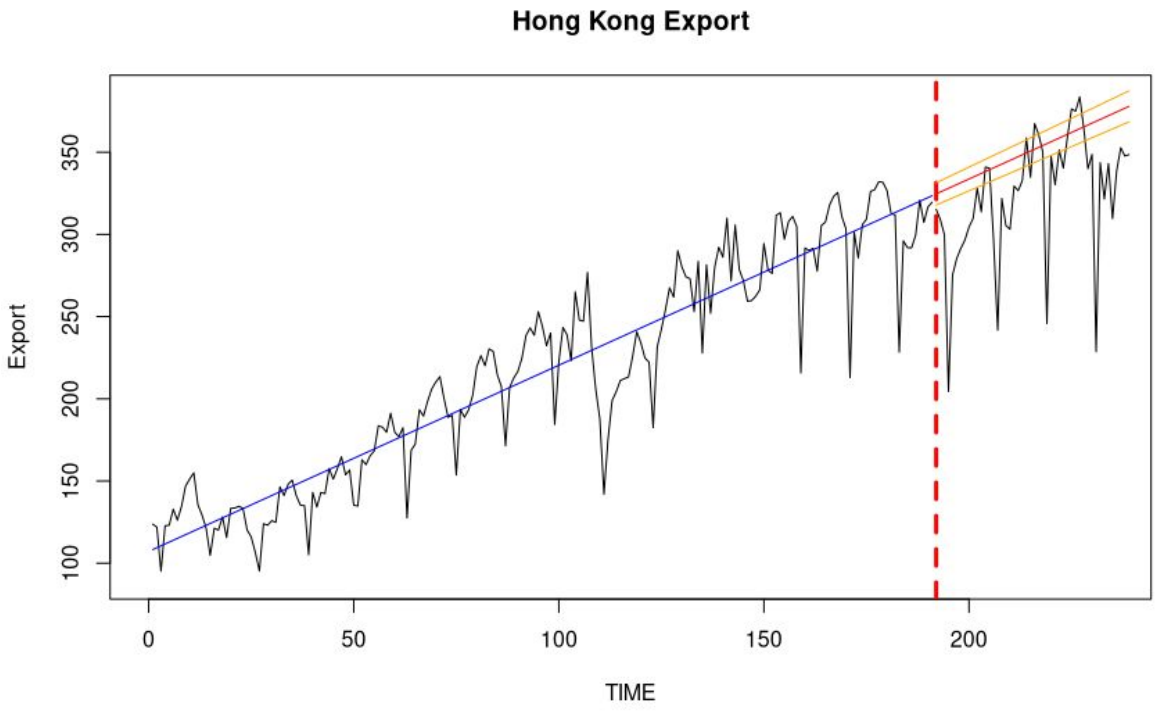
```
> summary(linear_model)

Call:
lm(formula = hk_export ~ seq, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-90.963 -13.164   1.645  16.496  48.537

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 107.18462    3.44190   31.14   <2e-16 ***
seq           1.13260    0.03109   36.43   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 23.69 on 189 degrees of freedom
Multiple R-squared:  0.8753,    Adjusted R-squared:  0.8747
F-statistic:  1327 on 1 and 189 DF,  p-value: < 2.2e-16
```

We can see a very significant result. β_0 is estimated to be 107.18462 and β_1 is 1.13260, each with an extremely low p-value. Which means both significantly differ away from 0. Now we turn our focus to adjusted R-squared: 0.8747, over 87% of the training data is explained by $TIME$. Lastly, we shall check the BIC (equivalent to SIC in the lecture slides).

```
> bic <- BIC(linear_model)
> bic
[1] 1764.842
```

BIC alone does not carry any meaning, so we should keep it in mind and compare with other models later.

Then we ask how does this simple model perform in out-of-sample data? Before jumping into the numbers directly, let us see the whole picture of this model.



The region left to red dash line is the training data and the blue line is the fitted line. The red solid line is the prediction made by the fitted model while the 2 orange lines are the 95% confidence interval. Now to see the detailed in the numbers.

Now let us see the MSE first

```
> test_error <- mean((data.frame(linear_predict)$fit - test$hk_export)^2)
> test_error
[1] 1869.949
```

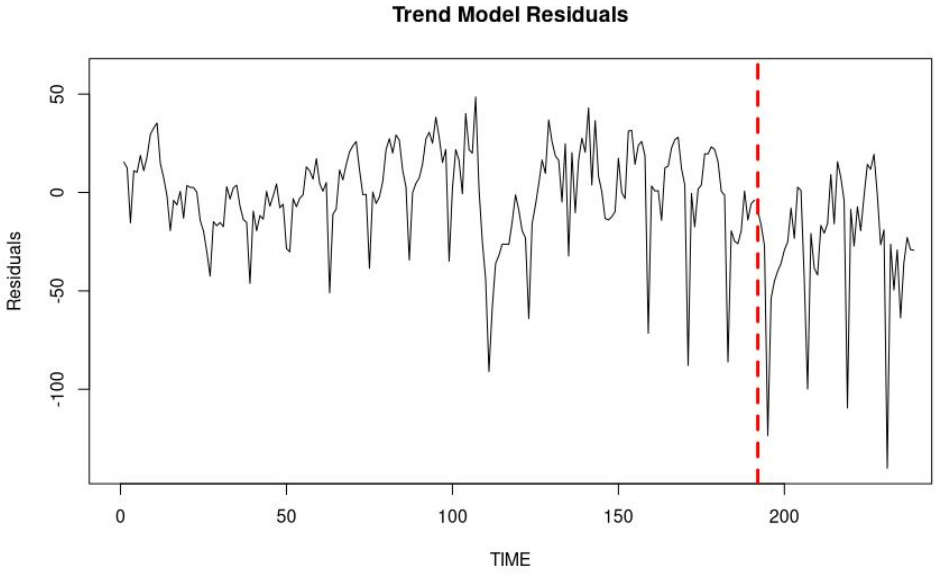
To be more rigorous, we shall examine the adjusted r square as well. Sadly, there is no convenient function in R to compute adjusted r square and BIC for out-of-sample data. I have written my own function to calculate adjusted r squared. However, building BIC myself but it is very easy to make an error and hard to debug as I won't know the correct value. We will simply use MSE and adjust r square to examine testing data. Afterall core of forecasting is about forecasting loss.

```r
# compute r squared for testing data
r_squared <- function(vals, preds){
  1-(sum((vals-preds)^2) / sum((vals-mean(preds))^2))
}

adj_r_squared <- function(vals, preds, k){
  1-((1-r_squared(vals, preds))*(length(preds)-1))/(length(preds)-k-1)
}

adj_r_squared(test$hk_export, data.frame(linear_predict)$fit, 1)
```

Here is the function I wrote

```
> adj_r_squared(test$hk_export, data.frame(linear_predict)$fit, 1)
[1] 0.1241984
```

The adjusted r squared is just 0.124! It is obvious that the model can not be generalized to unseen data. How can we do better than the basic trend model? Lets first look at what is left after we captured the trend in the data, namely, residuals.



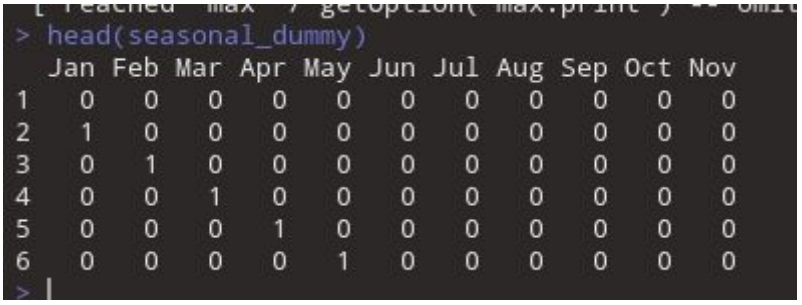
Not surprising, the residuals have very strong seasonality and incorporating that into our model should help to make a better forecast.

Seasonality

Recall that Hong Kong Export data is published every month and in February the export volume will experience a sharp decrease probably due to Lunar New Year Holiday, everyone is busying with celebration. To model seasonality, we add seasonal dummies into our model:

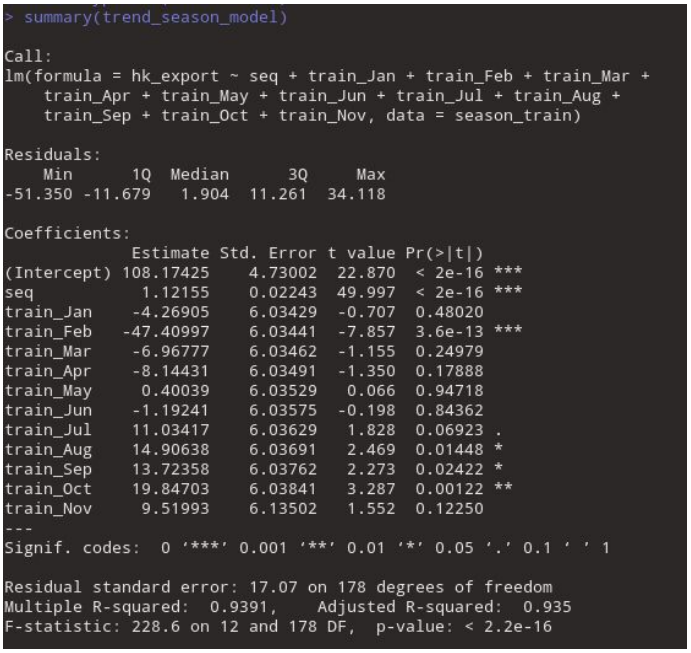
$$y_t = \beta_0 + \beta_1 TIME_t + \sum_{i=1}^{11} \alpha_i D_{it} + \varepsilon_t$$

The first two terms are exactly the same as the trend only. The last term consists of 11 (since we still keep beta 0) seasonal dummy variables and their coefficients. Each seasonal dummy variables is a vector having 1's in the time they represented. To show it more clearly:



```
> head(seasonal_dummy)
  Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
1   0   0   0   0   0   0   0   0   0   0   0
2   1   0   0   0   0   0   0   0   0   0   0
3   0   1   0   0   0   0   0   0   0   0   0
4   0   0   1   0   0   0   0   0   0   0   0
5   0   0   0   1   0   0   0   0   0   0   0
6   0   0   0   0   1   0   0   0   0   0   0
> |
```

After some transformation (please refer to the code for details), we fit the training data into the new model.

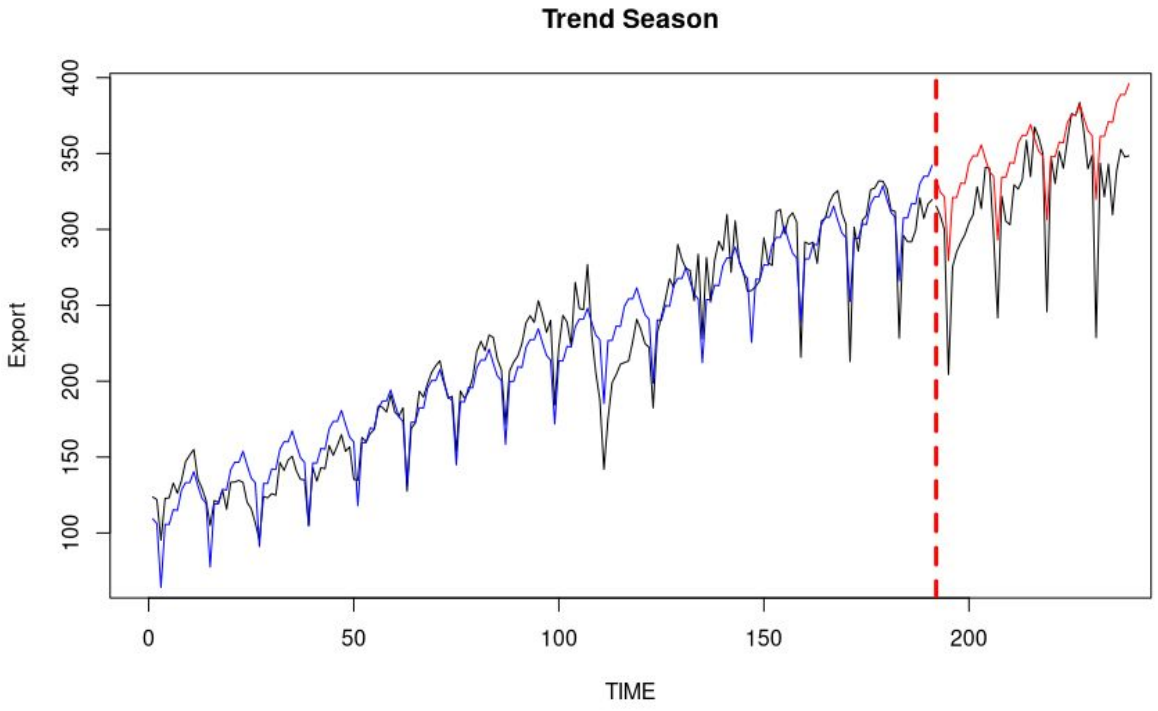


```
> summary(trend_season_model)

Call:
lm(formula = hk_export ~ seq + train_Jan + train_Feb + train_Mar +
    train_Apr + train_May + train_Jun + train_Jul + train_Aug +
    train_Sep + train_Oct + train_Nov, data = season_train)

Residuals:
    Min      1Q  Median      3Q     Max
-51.350 -11.679   1.904  11.261  34.118

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 108.17425    4.73002  22.870  < 2e-16 ***
seq           1.12155    0.02243  49.997  < 2e-16 ***
train_Jan    -4.26905    6.03429  -0.707  0.48020
train_Feb   -47.40997    6.03441  -7.857  3.6e-13 ***
train_Mar    -6.96777    6.03462  -1.155  0.24979
train_Apr    -8.14431    6.03491  -1.350  0.17888
train_May     0.40039    6.03529   0.066  0.94718
train_Jun    -1.19241    6.03575  -0.198  0.84362
train_Jul    11.03417    6.03629   1.828  0.06923 .
train_Aug    14.90638    6.03691   2.469  0.01448 *
train_Sep    13.72358    6.03762   2.273  0.02422 *
train_Oct    19.84703    6.03841   3.287  0.00122 **
train_Nov     9.51993    6.13502   1.552  0.12250
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.07 on 178 degrees of freedom
Multiple R-squared:  0.9391,    Adjusted R-squared:  0.935
F-statistic: 228.6 on 12 and 178 DF,  p-value: < 2.2e-16
```

The first thing we see is that not all dummies are significant, nevertheless, they are important to control the result. February is no doubt very significant since almost all the spikes occur in February. And summer period from July to September have relatively strong seasonal effect as well. Note that the adjusted R-squared is 0.935, much higher than in trend only model using training data. Let us check the BIC.

```
> BIC(trend_season_model)
[1] 1685.902
```

The BIC is indeed lower than the one in trend only model (1764.842). So we can say the new model provides a better fit for the Hong Kong Export time series data. Before we put the model into testing data, we shall first see the graph.



The region left to red dash line is the training data and the blue line is the fitted line. The red solid line is the prediction made by the fitted model. I don't plot the confidence interval here as the graph will look messy, you may refer to the code if you are interested in the confidence interval.

We then test it will out-of-sample to see the model's MSE and adjusted R-squared.

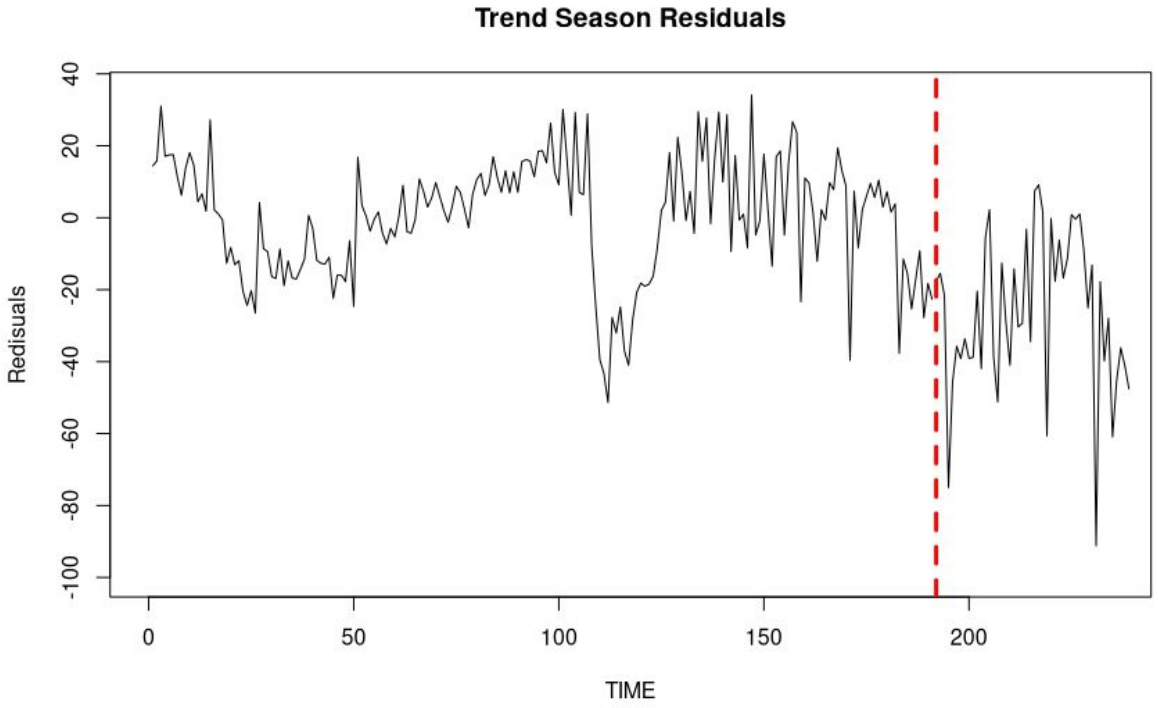
```
> season_test_error <- mean((data.frame(trend_season_predict)$fit - test$hk_export)^2)
> season_test_error
[1] 1153.708
```

The MSE is about 700 lower than trend only model prediction (1869.949). And for adjusted R-squared:

```
> adj_r_squared(test$hk_export, data.frame(trend_season_predict)$fit, 12)
[1] 0.2668577
```

0.2668677. Meaning that the trend season model can explain more than double of the variability than the trend only model.

The model is performing OK, but what if we want to do even better? Let us check what is left in the residuals.



At first glance, we might think that the residual is random and can do nothing about it. But luckily, we have Durbin-Watson test that we can use to check if there is any correlation in the residuals.

```
        Durbin-Watson test

data:  trend_season_model
DW = 0.77542, p-value < 2.2e-16
alternative hypothesis: true autocorrelation is greater than 0
```

DW test statistic will be around 2 if the residuals are not correlated, we have DW=0.77542, that is a sign of positive correlation. And we see the p-value is extremely small, that means we reject the null hypothesis. There is high chance the residuals are correlated.

Cycle

We have captured trend, seasonality and now only cycle is left in the residuals. If we can capture the cycle in the residuals of the previous model, we can then get a full model for Hong Kong Export. There are 3 major models to model the elements of the cycle in a time series. Namely, Moving Average (MA) model, Autoregressive (AR) model and the join of them, ARMA model. As discussed in class, ARMA is the most commonly used model as it can give a better and more parsimonious result. The drawback is that using the ARMA model, we can not tell how much lag in MA (q) and how much lag in AR (p) to use by seeing the ACF and PACF graph. Since both will have no clear cut-off.

Therefore, the method I use to search the best p and q is by brute force. In practice, the value of p and q won't be too large, so I set p and q in the range of 0 to 6, and calculate every possible combination of models' BIC, picking the lowest one.

```
# Cycle
max_p = 6
max_q = 6

result_list <- data.frame(matrix(ncol = 3, nrow = 0))
cat("p   q   BIC")
for (p in 0:max_p)
{
  for (q in 0:max_q)
  {
     result <- arima(trend_season_residuals, order=c(p,0,q))
     bic <- BIC(result)
     cat(p, q, bic)
     cat("\n")
     result_list <- rbind(result_list, c(p,q,bic))
  }
}
x <- c("p", "q", "BIC")
colnames(result_list) <- x
```

And the result is as fellow:

```
> result_list[which.min(result_list[,3]),]
  p q      BIC
9 1 1 1513.535
```

Turns out the best model to model residuals is ARMA(1, 1). Therefore our model will be:

$$(1)\ y_t = \beta_0 + \beta_1 TIME_t + \sum_{i=1}^{11} \alpha_i D_{it} + \varepsilon_t \quad where$$

$$(2)\ \varepsilon_t = \delta_t + \phi\varepsilon_{t-1} + \theta\zeta_{t-1} + \psi_t \ where\ \zeta is\ the\ residuals\ of\ \varepsilon$$

$$Let\ the\ residuals\ of\ (2)\ be\ \psi_t$$

Unfortunately, there is no strict forward to put all 3 elements $trend,\ season\ and\ cycle$ in 1 standard R lm function. That means I can not perform regression just as before. So I shall introduce how to get the fitted and predicted value in this model.

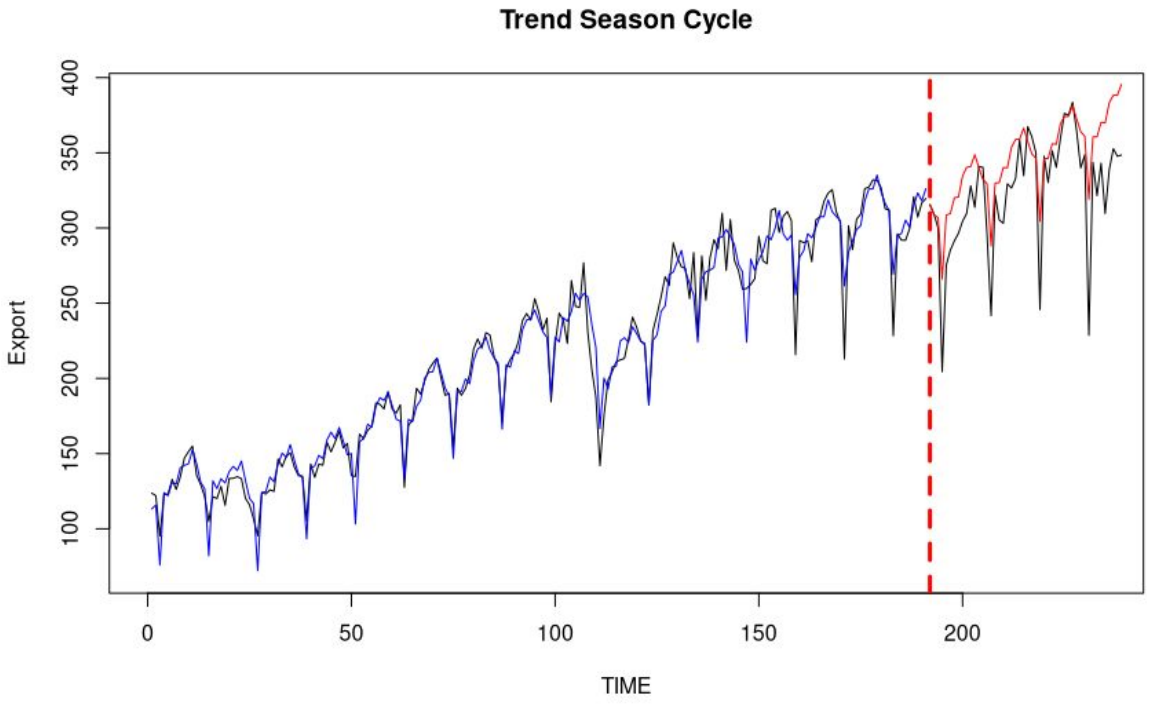
Note that the value for y_t is known, we also know the value for ε_t as well as the residuals for the (2) equation ψ_t. If we subtract ψ_t from y_t, we will get the fitted and prediction value!

$$y_t - \Psi_t = \beta_0 + \beta_1 TIME_t + \sum_{i=1}^{11} \alpha_i D_{it} + \delta_t + \phi\varepsilon_{t-1} + \theta\zeta_{t-1}$$

In R code (spent tons of time on this workaround…)

```
# Workaround
arma_model <- Arima(trend_season_residuals, order=c(1,0,1))
arma_residuals <- arma_model$residuals

full_past <- data.frame(trend_season_past)$fit + trend_season_residuals - arma_residuals
```

Before we see the statistic for the model, first see the graph for our final model.



Again, the region left to red dash line is the training data and the blue line is the fitted line. The red solid line is the prediction made by the fitted model. I don't plot the confidence interval here as the graph will look messy, you may refer to the code if you are interested in the confidence interval.

Visually, it is a very good fit! But what does the statistic tells us? Sadly since this result is not built by any R object, we can not effectively compute their BIC and other statistics. Therefore I could only report their MSE and adjusted R-squared.

For the training part. the MSE is:

```
> full_error <- mean((train$hk_export - full_past)^2)
> full_error
[1] 144.3154
```

and adjusted R squared:

```
> adj_r_squared(train$hk_export, full_past, 14)
[1] 0.9650298
```

Which both statistics are much better than the previous models!

Here comes the ultimate test for the model, the MSE of testing is:

```
> full_predict_error <- mean((test$hk_export - full_predict)^2)
> full_predict_error
[1] 930.58
```

and the adjusted R squared:

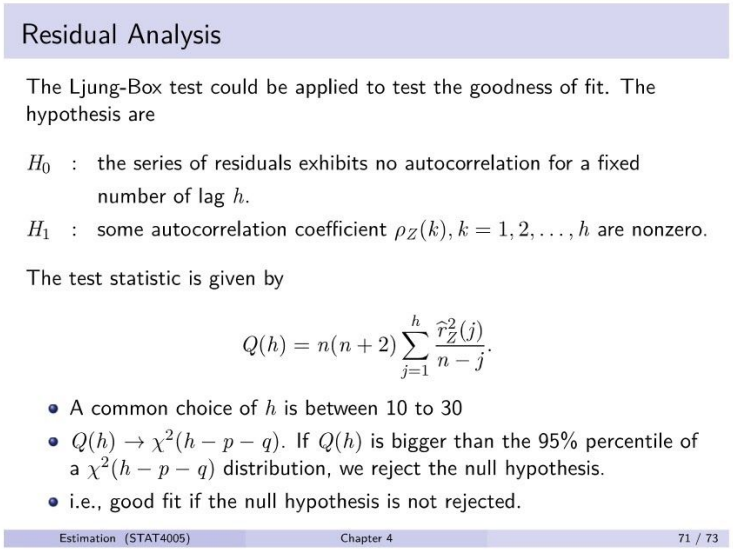
```
> adj_r_squared(test$hk_export, full_predict, 14)
[1] 0.3012955
```

Again, both statistics prove this model is more superior than the previous 2. However, if we look at the adjusted R squared, it is just over 0.3. Only 30% of the variability is explained by our model. Is it the best that we can do?

Can we have a better model?

To answer this question, first, recall what we can model and what we can not. We can model trend, seasonality and cycles. But if the data we have is pure white noise sequence, there is nothing we can do about it. Since for white noise, it's mean and covariance with its own lag term is both 0. So what if the residuals left in our trend season cycle is white noise? We can test it using the Ljung-Box Q-statistic.

Here we compute the Ljung-Box Q-statistic from lag 10 to lag 30 as suggested in this lecture slides from STAT4005



Residual Analysis

The Ljung-Box test could be applied to test the goodness of fit. The hypothesis are

H_0 : the series of residuals exhibits no autocorrelation for a fixed number of lag h.

H_1 : some autocorrelation coefficient $\rho_Z(k), k = 1, 2, \ldots, h$ are nonzero.

The test statistic is given by

$$Q(h) = n(n+2)\sum_{j=1}^{h}\frac{\hat{r}_Z^2(j)}{n-j}.$$

- A common choice of h is between 10 to 30
- $Q(h) \to \chi^2(h-p-q)$. If $Q(h)$ is bigger than the 95% percentile of a $\chi^2(h-p-q)$ distribution, we reject the null hypothesis.
- i.e., good fit if the null hypothesis is not rejected.

Estimation (STAT4005) Chapter 4 71 / 73

```
for (i in 10:30){
  temp <- Box.test( (test$hk_export - full_predict), type="Ljung-Box", lag=i, fitdf=1+1) # p+q
  cat(temp$statistic, temp$parameter, temp$p.value)
  cat("\n")
}
```

And the result:

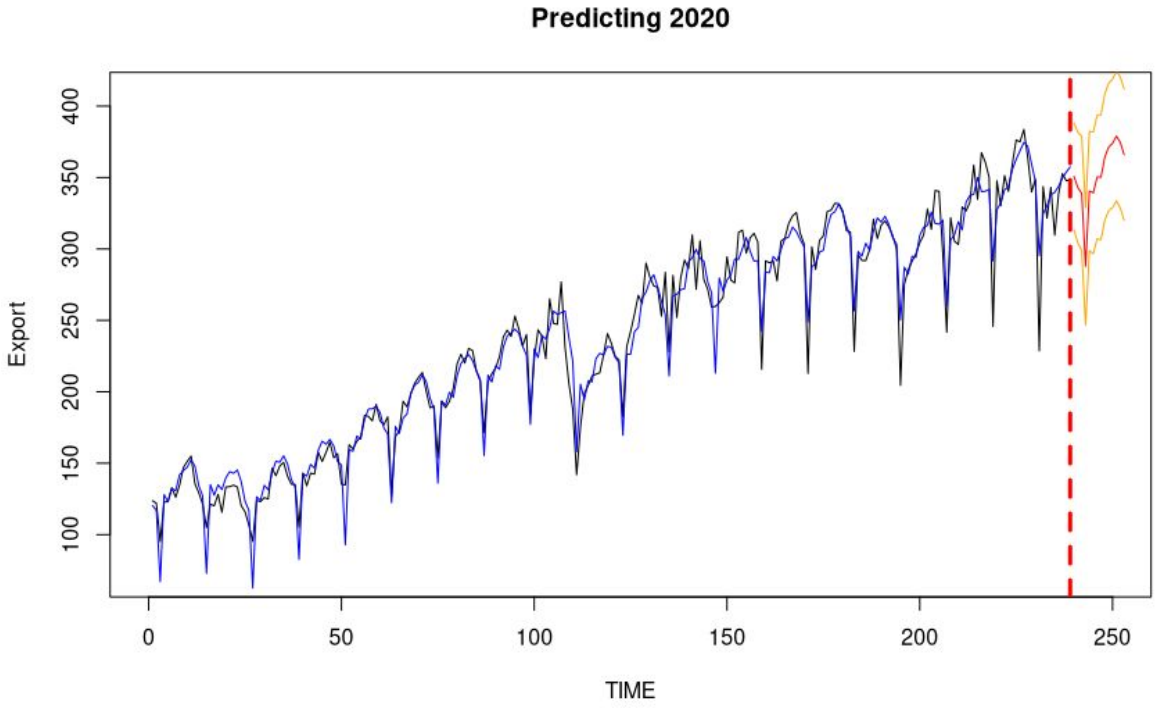
```
6.532643 8 0.5877892
7.534292 9 0.5816707
12.42774 10 0.2574485
14.39214 11 0.2120491
16.65033 12 0.1632308
20.38641 13 0.08598518
20.44257 14 0.1167941
23.1723 15 0.08054949
23.81151 16 0.09370292
25.58604 17 0.08233997
25.84493 18 0.1033568
26.66613 19 0.1126733
27.2812 20 0.1275363
27.35389 21 0.1594443
30.70395 22 0.1023345
30.70729 23 0.1301557
31.41695 24 0.1421209
33.61757 25 0.1163382
34.13627 26 0.1316275
34.18522 27 0.1607289
34.33193 28 0.1901576
```

First column: X-squared, second: df, third: p-value

As we can see, none of the p-values is smaller than 0.05, we accept the null hypothesis of the sequence is white noise. That means in the case of having over 0.3 adjusted R-squared, the prediction made by the trend season cycle model actually captured all the information it cans! And whats in the residuals is just white noise that we can not model them!

Forecasting 2020

Now that we have a model for Hong Kong Export. Let's make a prediction for the year 2020 using all the information in the information set we got!



Again, the region left to red dash line is the training data and the blue line is the fitted line. The red solid line is the prediction made by the fitted model while the 2 orange lines are the 95% confidence interval.

A table for the predicted value where the first row is the prediction for November 2019, the last row is for December 2020.

	TIME	Predicted Export Interval Up	Predicted Export	Predicted Export Interval Down
1	240	388.1185	350.7351	313.3517
2	241	381.6593	342.9255	304.1917
3	242	378.8749	338.9280	298.9811
4	243	328.7665	287.7961	246.8257
5	244	382.4102	340.5705	298.7307
6	245	381.8477	339.2657	296.6837
7	246	393.5323	350.3139	307.0955
8	247	393.9632	350.1974	306.4316
9	248	408.6207	364.3828	320.1449
10	249	415.9569	371.3108	326.6648
11	250	418.9151	373.9156	328.9161
12	251	424.2860	378.9799	333.6739
13	252	420.5036	374.6332	328.7627
14	253	412.0835	366.0706	320.0576

Reference

The R code used in this report can be found at

https://github.com/Elon-Chan/Forecasting-Hong-Kong-Export.git