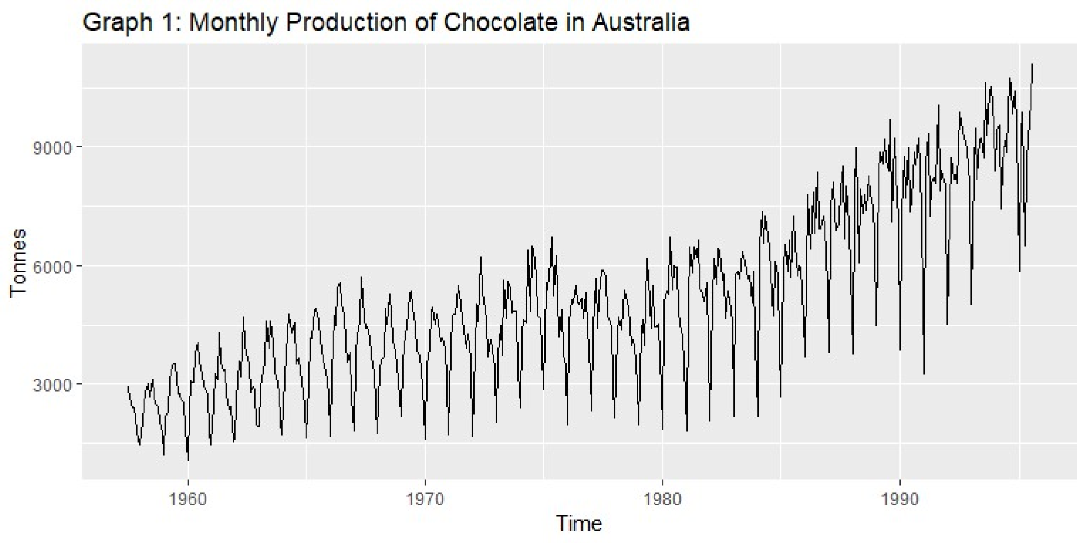
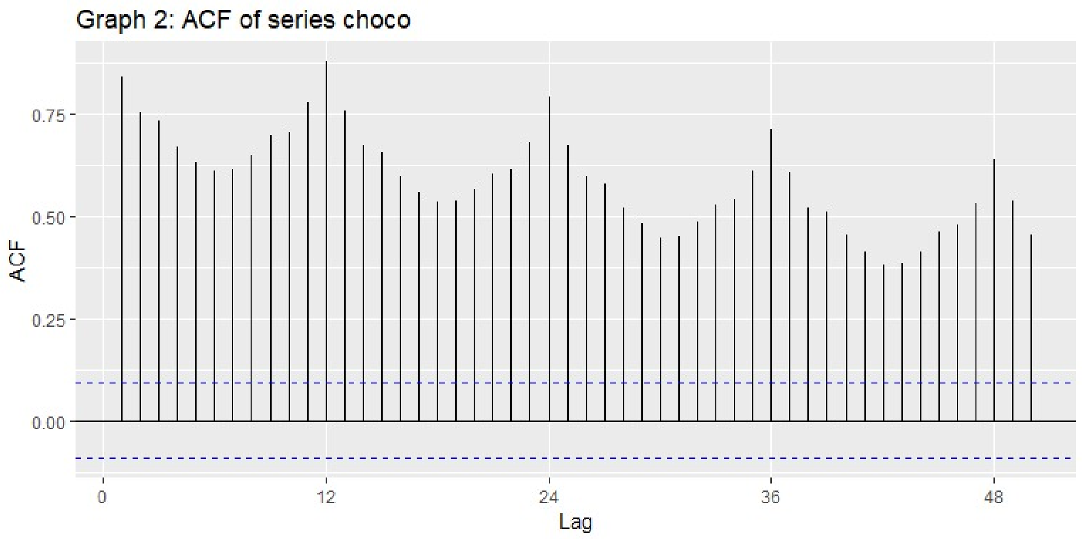
BAN430 Assignment1

Read data and summary statistics

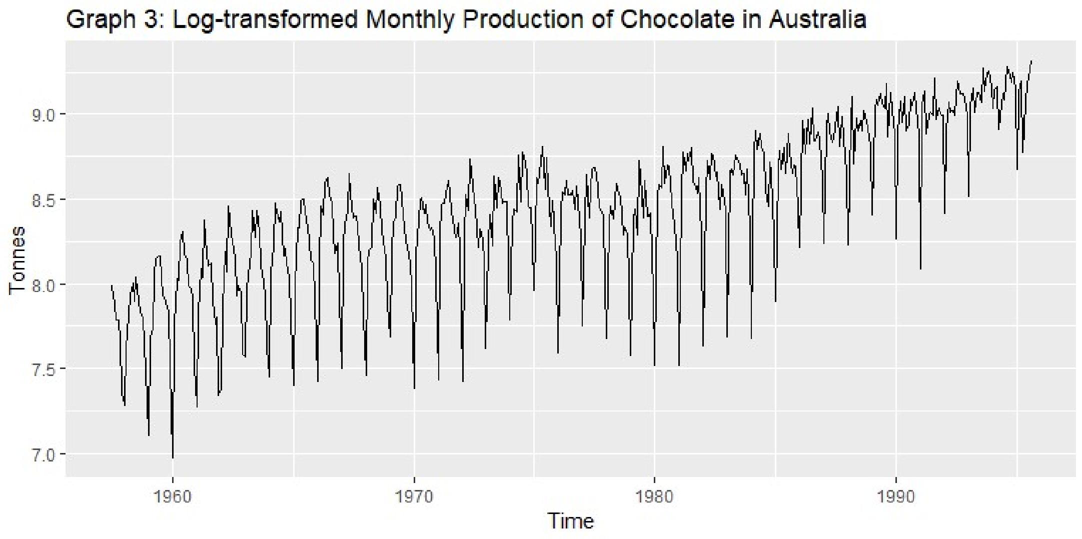
The data we have is a univariate time series of monthly production volume (in tonnes) of chocolate in Austrilia, from July 1957 to August 1995. We have 458 observations in total. We use R to make the data into a time series object, and split it into two parts, training data and test data. As required, we have the observations before August 1994 as training set, and the remaining observations as test set.



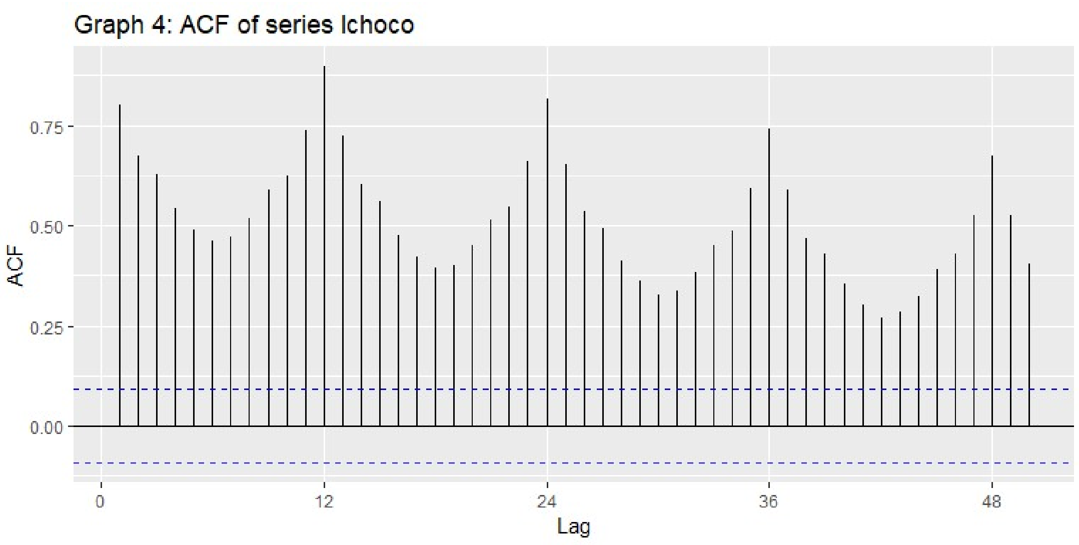
Firstly, we summarize the time series data and briefly describe the basic statistical features of it. The minimum and maximum observations are 1066 and 11095 respetively, with median of 4724 and mean of 5151. The number of observations is 458. We named this series as *choco*. Then we plot the data, we can see strong and regular seasonality from Graph 1. Overall the long-run trend of the data was steady growing. In addition, we draw the correlogram of the time series, and we set the lag length as 50. From the autocorrelated function graph (Graph 2), we can see that this series has strong autocorrelation, and the waved trend of the spikes also demonstrates obvious seasonality. Generally speaking, this series is nonstationary. To confirm our visual findings, we make a regression on trend and season (with results showed in Appendix). From the regression results, the trend variable and 11 seasonal dummy variables are all significant at 0.001 significance level. So we have strong evidence to say that the series *choco* is nonstationary, with significant trend, seasonality and autocorrelation.



Next, we try to make the series log-transfromed (*lchoco*), and we have similar findings on the new log series, with significant autocorrelation, seasonality and trend, as demonstrated in Graph 3, as conducting log-transformation to a time series does not change the property of data, but dose decreases the scale, and we can see the scale on y-axis has been reduced. In addition, the trend of log-transformed data is flatter than level data. However, there's still strong seasonality shown in Graph3.

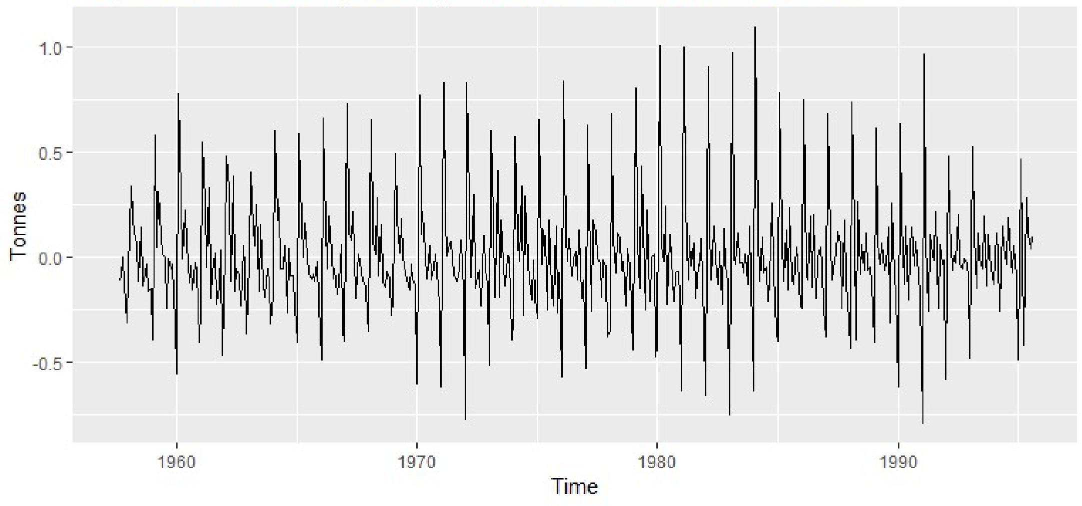


Next, we also plot the autocorrelation function of series *lchoco*. As demonstrated in Graph 4, the effects of autocorrelation diminish extremely slow and the spikes wave very regularly, again indicating significant autocorrelation and seasonality. We also conduct a regression of series *lchoco* on trend variable and 11 seasonal variables, and still all the variables are significant at 0.001 level, which can also verify the obvious trend and seasonality in this series. The regression results are also attached in the Appendix. Hence this series is not stationary.

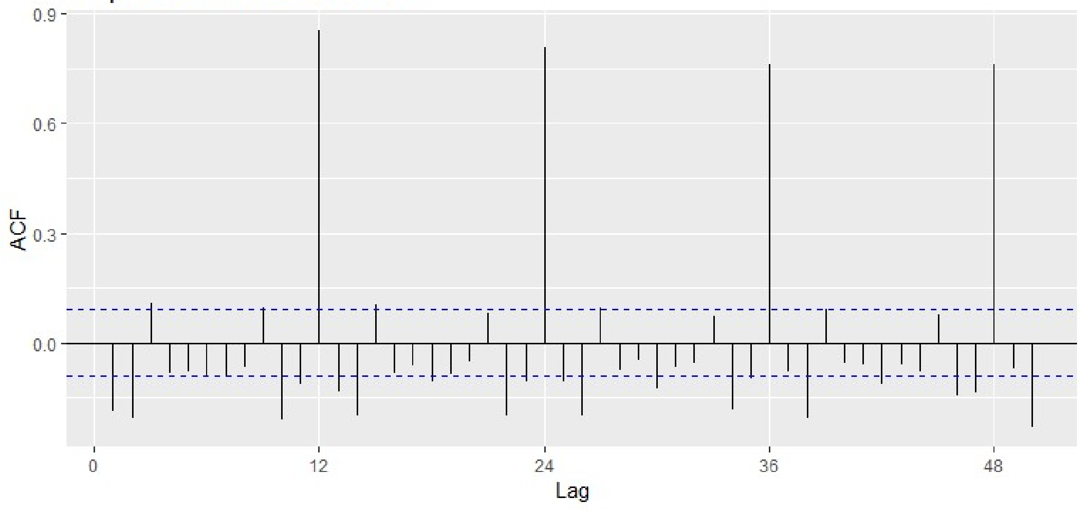


Furthermore, we make a first difference series of the log-transformed series (*dchoco*), which has a real economic meaning of monthly growth rate. This series still has obvious seasonality but the trend component has been effectively eliminated, according to regression on trend and seasons (results attached in Appendix). Hence, we can say the monthly growth rate is generally stable, and this finding can also be verified in Graph 5, there is no obvious trend. In addition, as shown in Graph 5, the series *dchoco* looks like a white noise, visually. In terms of autocorrelation, according to Graph 6, there are still some extremely significant spikes at lag12, lag24 and lag36 and so on, and those extreme observations may have something to do with yearly variance in chocolate production. So generally speaking, the first difference series of the log-transformed series (*dchoco*) is overall stationary.

Graph 5: Differencef Log Monthly Production of Chocolate in Australia



Graph 6: ACF of series dchoco

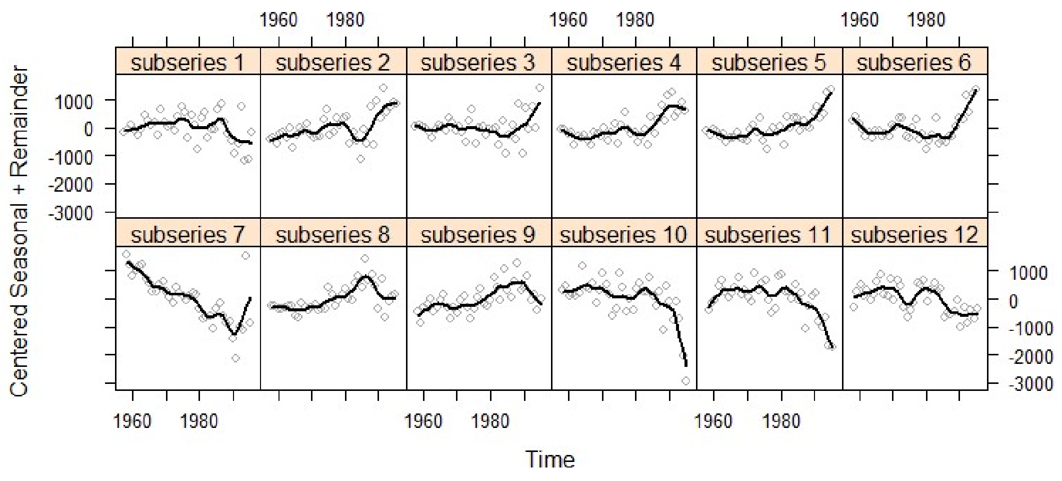


Decompose the series

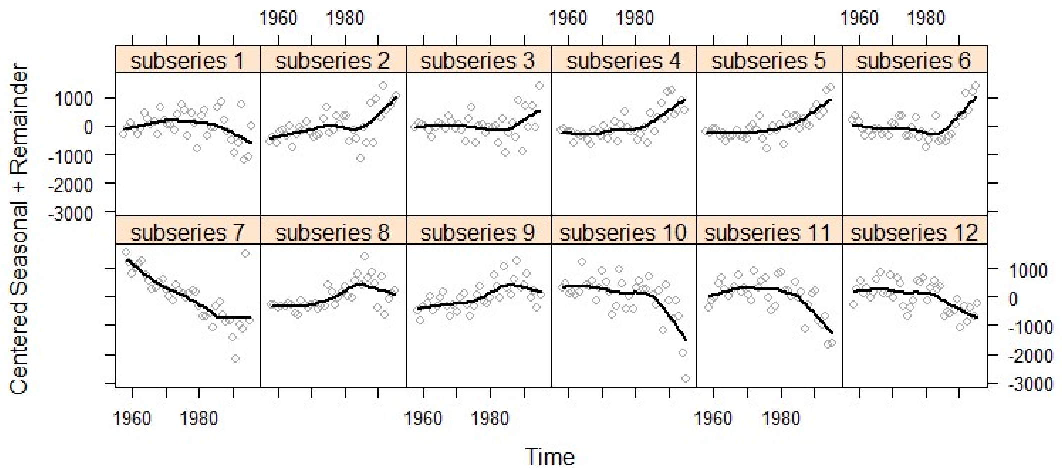
We choose STL decomposition when we decompose the series into its components, because STL decomposition is more versatile and robust when compared with other decomposition method such as SEATS and X11. It allows the seasonality to change through time. Also, we can control the rate of change of seasonal components and smoothness of trend-cycle. Although STL has disadvantages that it cannot deal with trading days and calendar variation automatically and can only provide facilities for additive decompositions, based on the data we have, those drawbacks do not matters at all.

In order to choose the seasonal smoothing parameter, we use *plot_seasonal()* function to draw the seasonal-diagnostic plot. (Cleveland, RB et.al., 1990. STL: A seasonal-trend decomposition approach using Loess. Journal of Official Statistics, v.6(1)). From Graph 7 we can see, the seasonal component is not perfectly periodic. When we set *s.window = 9*, an over-fitting occurs. However setting *s.window = 35* allows for some small change in the seasonality and avoids the problem of over-fitting (Graph 9). Hence, we set *s.window = 35* (http://www.gardner.fyi/blog/STL-Part-II/). To choose the relatively suitable t.window, we employ the formula *(1.5*period) / (1-(1.5/s.window))* and we get *t.window = 17*.

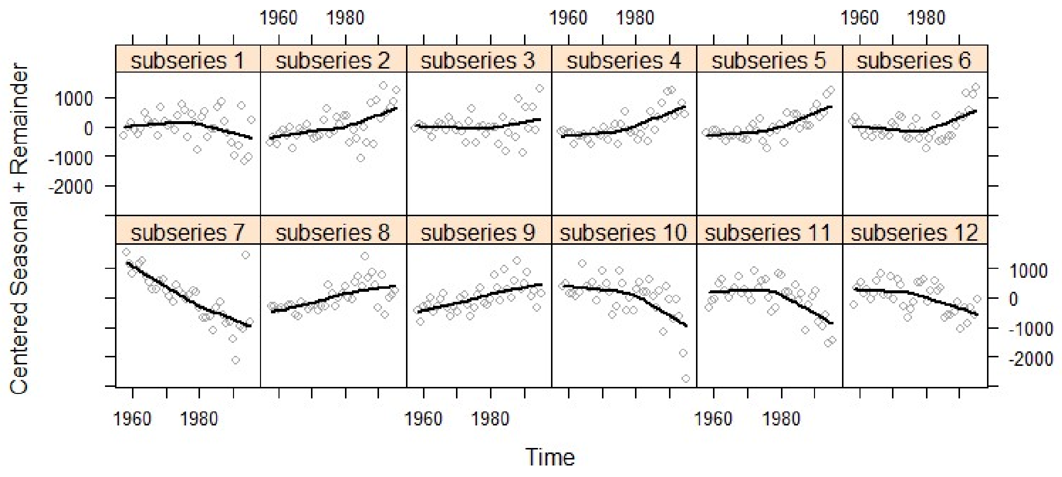
Graph 7: Seasonal-diagnostic plot (9)



Graph 8: Seasonal-diagnostic plot (21)

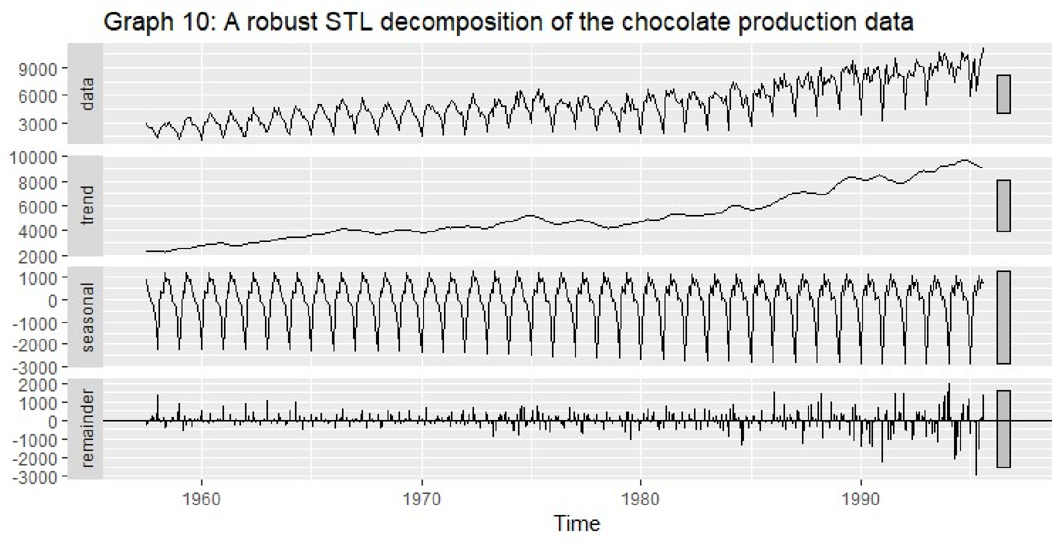


Graph 9: Seasonal-diagnostic plot (35)



Then we apply *stl()* function, with the parameter we get from previous step, to the chocolate production data set.

As Graph 10 shows, there is an obvious upward trend in the amout of chocolate produced. The variation in seasonal part becomes larger as time goes by but this change happens very slowly. Remainder part of Graph 10 indicates that after 1990, unusual production amount happens more frequently than before. The grey bars on the right side of each component in Graph 10 show the relative scales of the components. Each grey bar represents same length. Thus, the smaller the grey bar, the larger the scale. The grey bar of remainder component is much larger than that of data, suggesting the variance in remainder component is much smaller than that in data.



Forecast the components with STL

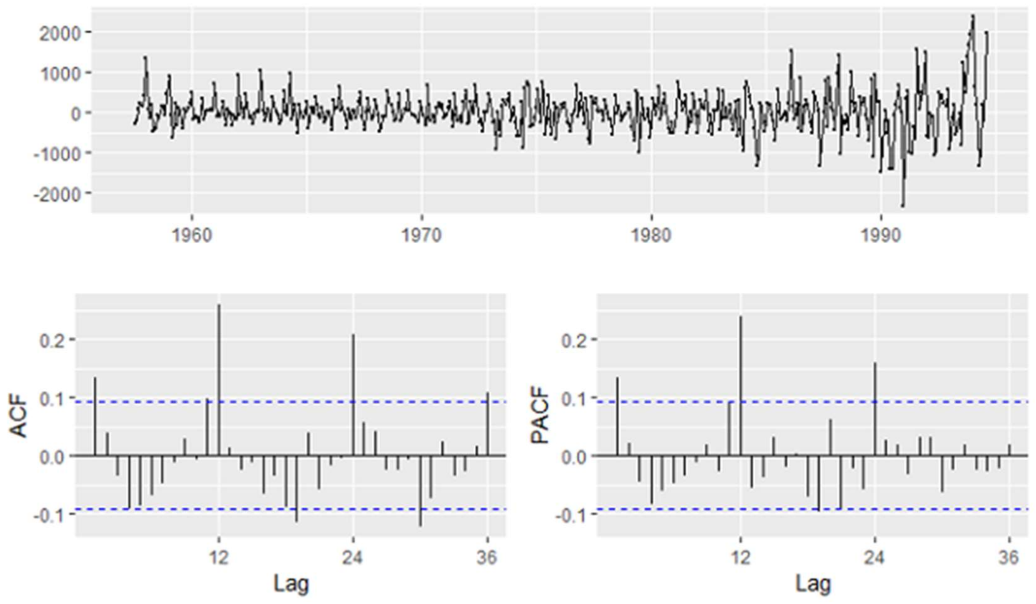
In order to forecast each component separately, we use *seasonal()*, *remainder()*, *trendcycle()* function to extract each component from *stl* decomposed time series data.

For seasonal component, since it is regularly and steadily changed according to seasons over time, we employ *snaive* method to forecast it. For remainder component, firstly we use *ggtsdisplay()* to check its stationarity. As shown in Graph 11, the mean of remainder series is constant at zero, and the variance increases as time goes by, and becomes quite violent after 1988. In addition, both ACF and PACF graphs show obvious seasonality in remainder data series. Hence, we suppose a seasonal ARIMA models to forecast the remainder component, and we use *auto.arima()* function to select an appropriate ARIMA model, and the model selected is ARIMA(2,0,2)(1,0,1)[12] with zero mean. For trend-and-cycle component, since it has an upward trend in the whole horizon and hard to determine the changes, we believe a method of random walk with drift suits this component best, and we use *rwf()* function. In addition, considering that trend-and-cycle component has obvious trend, we think may be ETS method can also apply to it. So we compare both methods of random walk with drift and ETS, testing the forecasting accuracy of them, and the method of random walk with drift has a relatively smaller RMSE (1396) than ETS (1858). So we decide to use method of random walk with drift to forecast trend-and-cycle component.

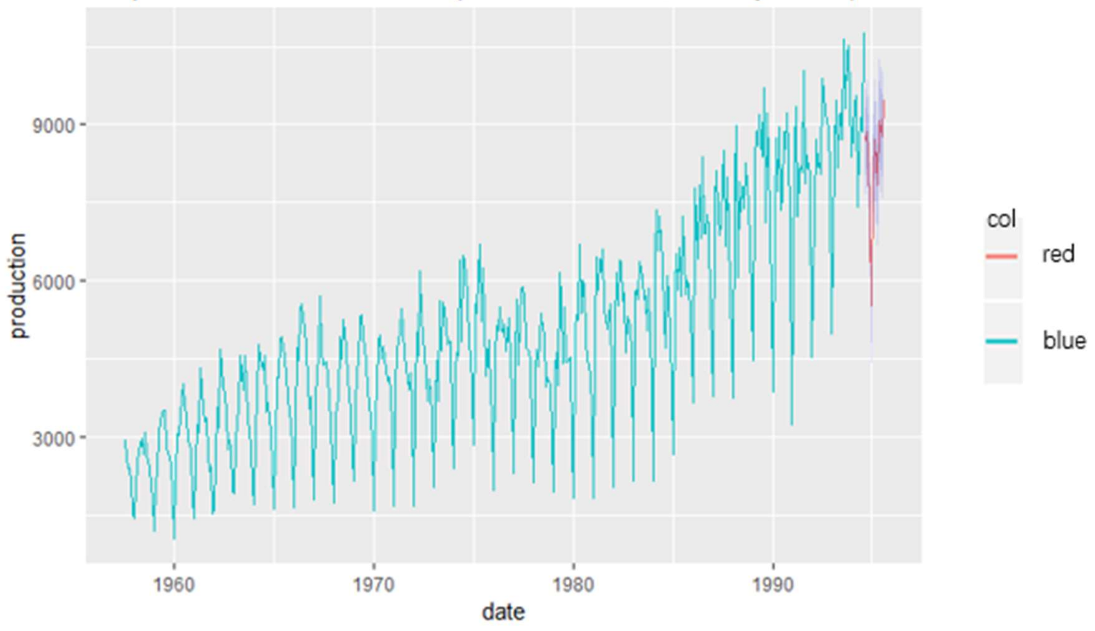
Next, we add the forecast results of the three components together to get a final forecast value. Because the uncertainty in seasonal component is much smaller than those of trend-and-cycle component and remainder component, so we can ignore the uncertainty in seasonal component when calculate prediction interval (Hyndman, R.J., & Athanasopoulos, G. 2018, Forecasting: principles and practice, 2nd edition). Hence, when calculating the prediction interval, we add the lower boundaries of the predictions in remainder and trend-and-cycle component and upper boundaries as well, then adding forecasting value of seasonal component respectively. Graph 12 shows forecast result and we zoom it in Graph 13.

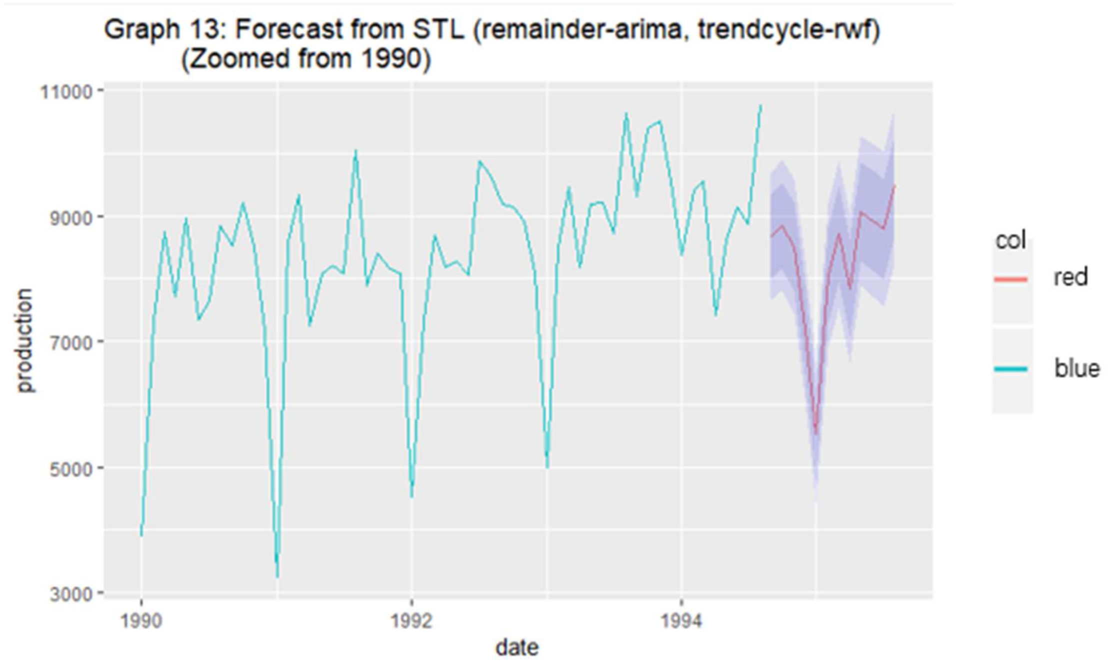
As we can see from Graph 13, the prediction interval is relatively narrow, making forecasting of chocolate production in Australia relatively easy and accurate.

Graph 11: Summary statistic of remainder component



Graph 12: Forecast from STL (remainder-arima, trendcycle-rwf)



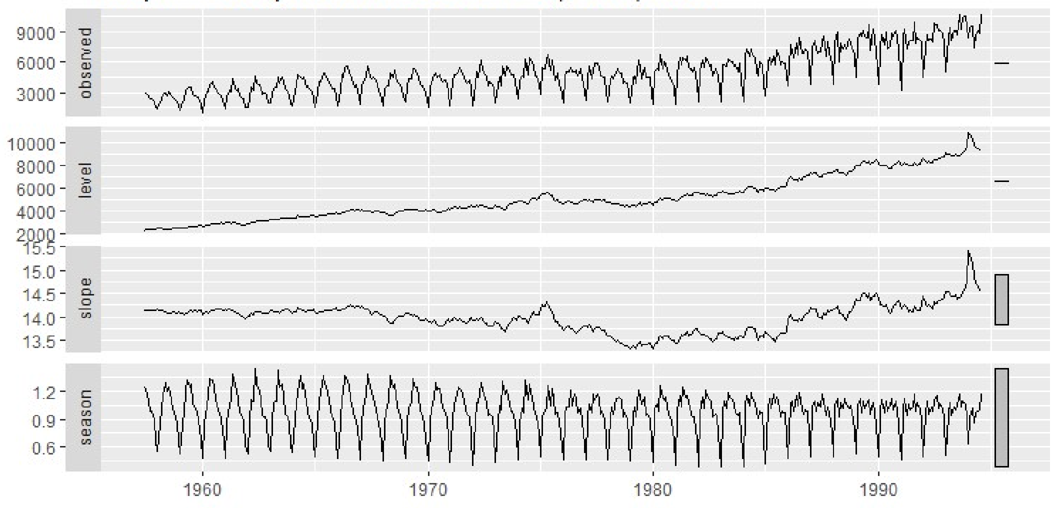

Graph 13: Forecast from STL (remainder-arima, trendcycle-rwf)
(Zoomed from 1990)
11000
9000
7000
5000
3000
production
1990
1992
1994
date
col
red
blue

Select an appropriate ETS model

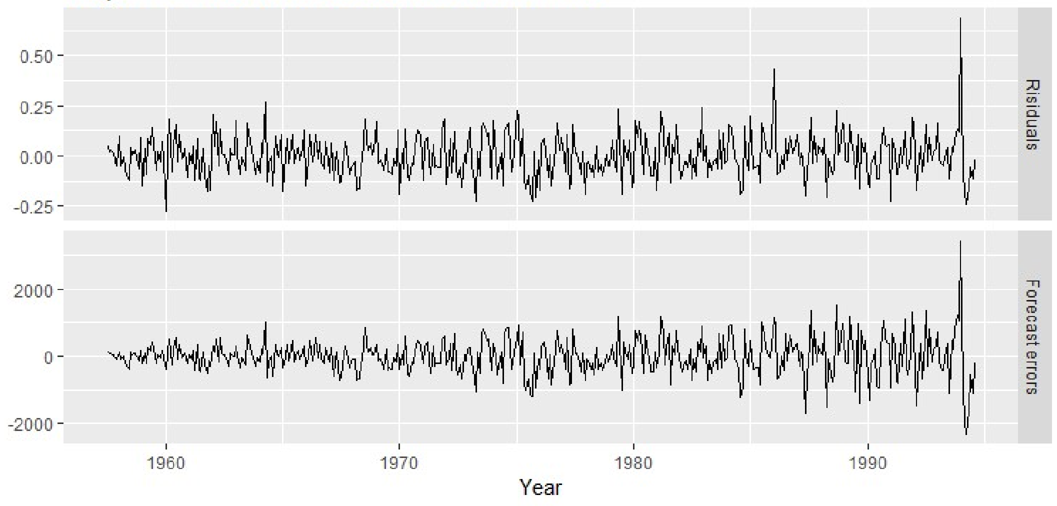
In this part we choose to fit a ETS model and do forecast. “ETS” means “Error, Trend and Seasonal”, and the model automatically selected by R function is *ETS(M,A,M)*, namely “Multiplicative Holt-Winters’ method with multiplicative errors”. The parameter estimates are *alpha-hat = 0.1921, beta-hat = 0.0001, gama-hat = 0.3473*. The small value of beta means that the slope component changes very little over time.

Graph 14 shows the development situation of each component over time. The grey bar of slope and season component is relatively wide, indicating small change in each component. The level component is relatively smooth, but it has some small variance due to changes in slope. Hence, it is relatively easy to forecast. Graph 16 indicates that the ETS model we chose can makes prediction easier because its prediction interval is relatively narrow. But the prediction interval is not narrow enough for us to determine the future very decisively, because the level component is not smooth enough. By only focusing on the time period after 1980, as Graph 17 shows, we can observe this more straightforward. Graph 15 indicates that the residuals in our model is not equivalent to the one-step training errors since our model has multiplicative errors, whose residuals are $\hat{\varepsilon}_t$ while residuals of one-step training errors are $y_t - \hat{y}_t|_{t-1}$. We also use *accuracy()* function to get the forecast error on test data set. We get the RMSE of around 694, which is much smaller than that of STL one in previous sub-question. Hence, based on the data we have, we think forecasting with ETS model outperforms STL.

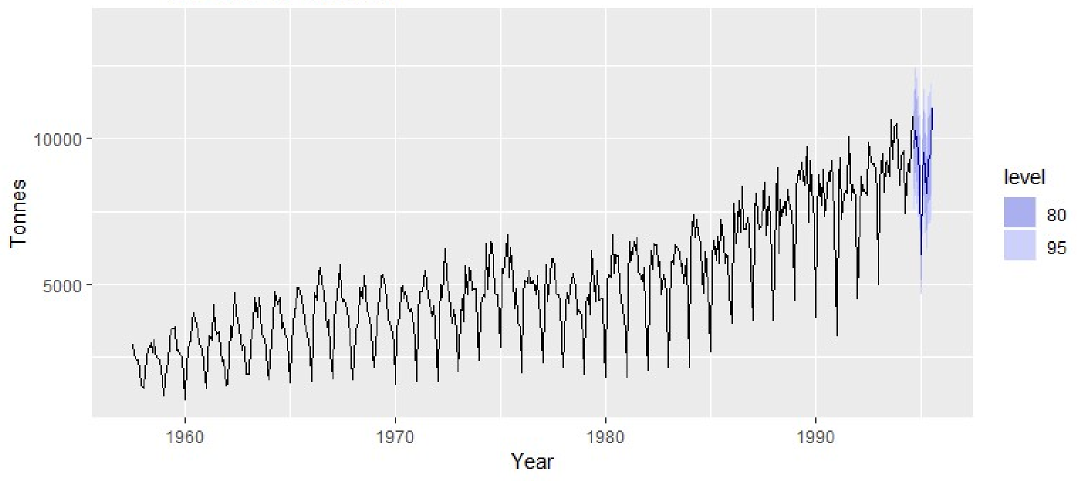
Graph 14: Components of selected ETS(M,A,M) model



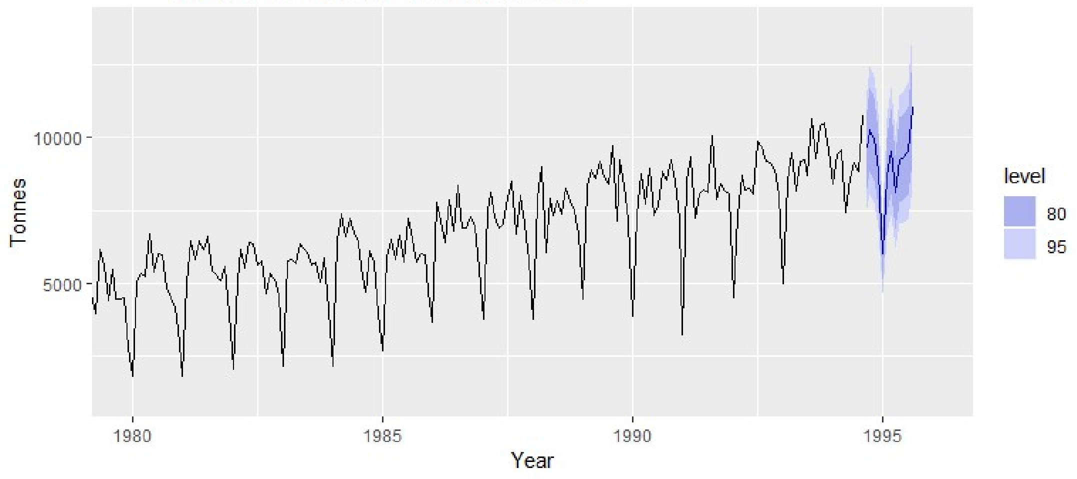
Graph 15: Plots of residuals and forecast errors



Graph 16: ETS forecasts of chocolate production for the period 1994:M9 to 1995:M8

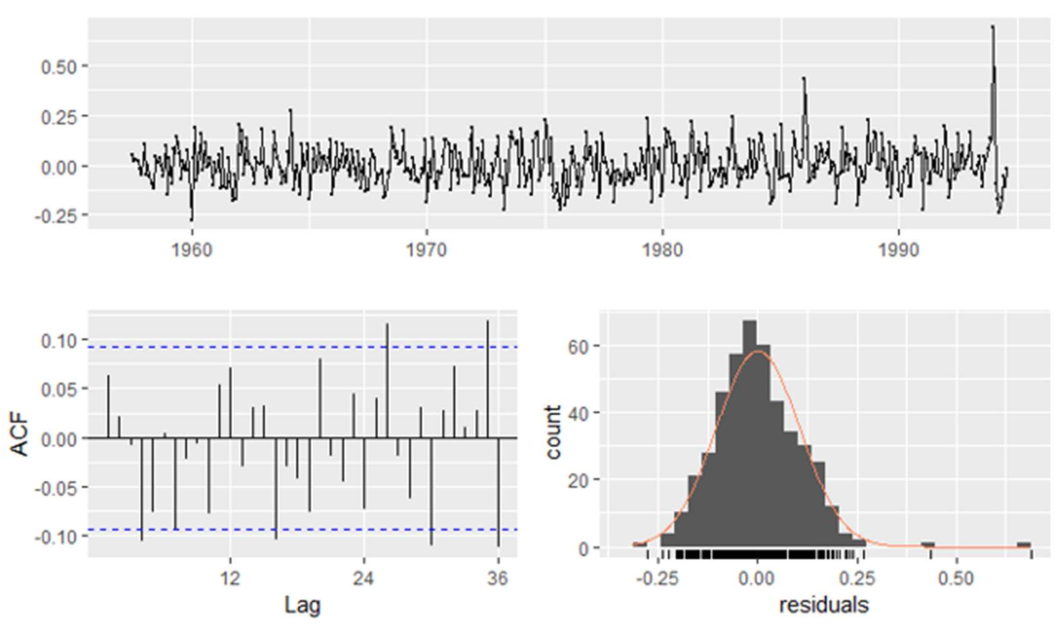


Graph 17: ETS forecasts of chocolate production for the period 1994:M9 to 1995:M8 (plot from 1980)



In addition, we test the residual series of our ETS model. According to Graph 18, the residuals series is close to white noise, though with some spikes out of bound in ACF plot. But generally, the ETS regression model is satisfactory and this means ETS model can fit our training data relatively well, and it indicates that we selected an appropriate ETS model. But the residual series cannot be used to evaluate the forecasting power of ETS model, as it is only focused on training data.

Graph 18: Residuals from ETS(M,A,M)



Reproduce tsCV()

We conducted the time series cross validation in both ways: using package function *tsCV()* and reproducing codes ourselves.

Firstly according to the requirement, the first prediction is on 1993M10, based on 1993M8. So the horizon (h) is 2 and the innitial number of observations in training data is 433. Here the 1 deducted represents the observation on 1993M8, we should not include that one in the minumum set. Then we use function *tsCV()* to test the RMSEs of two forecasted horizons, with the results of 1130.845 and 1154.918 respectively for $h = 1$ and $h = 2$.

```
# tsCV
length(window(choco, end = c(1993,8))) - 1 # 433
etscv <- tsCV(choco, snaive, h = 2, initial = 433)
sqrt(mean(etscv[,1]^2, na.rm = T))
sqrt(mean(etscv[,2]^2, na.rm = T))

# Reproduction
k <- 433 # minimum data length for fitting a model
n <- length(choco)
err <- matrix(NA, n-k-1, 2)
st_date <- tsp(choco)[1]+(k-1)/12
h <- 2
for(i in 1:(n-k-1))
{
  train <- window(choco, end = st_date + i/12)
  test <- window(choco, start = st_date + (i+1)/12, end = st_date + (i+2)/12)
  fit <- snaive(train,h=2)
  err[i,1] <- (test[1]-fit$mean[1])
  err[i,2] <- (test[2]-fit$mean[2])
}
sqrt(mean((err[,1]^2),na.rm = T))
sqrt(mean((err[,2]^2),na.rm = T))
```

Then we reproduce the function. We set $k = 433$, which is the minimum data length for fitting a model. We set n equals to the length of time series. We also prepare a 2-column empty matrix for storing the forecasting errors. We determine the starting date numerically by using function *tsp()*, because *tsp()[1]* is the numeric starting time of a series. Since we have k minimum data length in training data, we set the starting date of cross validation as *tsp(choco)[1]+(k-1)/12*, here the 1 deducted is the beginning observation of the series, and the 12 divided is the frequency of data series. We set prediction horizon as 2. We use a for loop to realize the cross validation process. In the loop horizon, we set it ends at *(n-k-1)*, where the 1 deducted is because last observation at 1995M8 cannot be included. That is the same reason why we set the row number of *(n-k-1)* in the previous matrix. In the loop, training data is the minimum data length adding the new observations, using command "*window(choco, end = st_date + i/12)*". Here "*i/12*" means the newly added numeric length of observations. We use similar command to set test data. Then we use s-naive function to do the forecasting, and calculate the difference between predicted value and actual value in test set, storing them in the matrix. Then we calculate the RMSEs for two columns in matrix separately, and we have the same results as using *tsCV*() function.

Appendix

Regression of *choco* on trend and season variables:

```
##
## Call:
## tslm(formula = choco ~ trend + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2005.26  -559.18   -29.02   495.57  2891.59
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -515.6422   151.9485  -3.394 0.000752 ***
## trend         13.6994     0.2963  46.240  < 2e-16 ***
## season2     2422.7479   192.2780  12.600  < 2e-16 ***
## season3     3080.0748   192.2786  16.019  < 2e-16 ***
## season4     2592.0859   192.2798  13.481  < 2e-16 ***
## season5     3505.6233   192.2814  18.232  < 2e-16 ***
## season6     3131.7660   192.2834  16.287  < 2e-16 ***
## season7     3301.5506   191.0412  17.282  < 2e-16 ***
## season8     3230.1589   191.0414  16.908  < 2e-16 ***
## season9     2512.6398   192.2814  13.068  < 2e-16 ***
## season10    2603.6246   192.2798  13.541  < 2e-16 ***
## season11    2390.3199   192.2786  12.432  < 2e-16 ***
## season12    1457.4626   192.2780   7.580 2.03e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 838.1 on 445 degrees of freedom
## Multiple R-squared:  0.8591, Adjusted R-squared:  0.8553
## F-statistic:   226 on 12 and 445 DF,  p-value: < 2.2e-16
```

Regression of *lchoco* on trend and season variables:

```
##
## Call:
## tslm(formula = lchoco ~ trend + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38949 -0.09226 -0.00245  0.09385  0.70710
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.133e+00  2.591e-02  275.35   <2e-16 ***
## trend       2.721e-03  5.051e-05   53.87   <2e-16 ***
## season2     6.776e-01  3.278e-02   20.67   <2e-16 ***
## season3     8.178e-01  3.278e-02   24.95   <2e-16 ***
## season4     7.574e-01  3.278e-02   23.10   <2e-16 ***
## season5     9.266e-01  3.278e-02   28.27   <2e-16 ***
## season6     8.582e-01  3.278e-02   26.18   <2e-16 ***
## season7     8.802e-01  3.257e-02   27.02   <2e-16 ***
## season8     8.423e-01  3.257e-02   25.86   <2e-16 ***
## season9     7.137e-01  3.278e-02   21.77   <2e-16 ***
## season10    7.167e-01  3.278e-02   21.86   <2e-16 ***
## season11    6.683e-01  3.278e-02   20.39   <2e-16 ***
## season12    4.298e-01  3.278e-02   13.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1429 on 445 degrees of freedom
## Multiple R-squared:  0.9053, Adjusted R-squared:  0.9027
## F-statistic: 354.5 on 12 and 445 DF,  p-value: < 2.2e-16
```

Regression of *dhoco* on trend and season variables:

```
##
## Call:
## tslm(formula = dchoco ~ trend + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.56655 -0.08045 -0.00693  0.07613  0.46805
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.275e-01  2.400e-02 -17.811  < 2e-16 ***
## trend        1.974e-06  4.696e-05   0.042    0.966
## season2      1.107e+00  3.038e-02  36.453  < 2e-16 ***
## season3      5.699e-01  3.038e-02  18.761  < 2e-16 ***
## season4      3.693e-01  3.038e-02  12.158  < 2e-16 ***
## season5      5.990e-01  3.038e-02  19.719  < 2e-16 ***
## season6      3.613e-01  3.038e-02  11.894  < 2e-16 ***
## season7      4.524e-01  3.038e-02  14.893  < 2e-16 ***
## season8      3.919e-01  3.018e-02  12.983  < 2e-16 ***
## season9      3.036e-01  3.038e-02   9.992  < 2e-16 ***
## season10     4.328e-01  3.038e-02  14.246  < 2e-16 ***
## season11     3.814e-01  3.038e-02  12.556  < 2e-16 ***
## season12     1.912e-01  3.038e-02   6.294 7.42e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1324 on 444 degrees of freedom
## Multiple R-squared:  0.7915, Adjusted R-squared:  0.7859
## F-statistic: 140.5 on 12 and 444 DF,  p-value: < 2.2e-16
```

Calculate of RMSE of both methods:

```
## [1] 1396.398
## [1] 1857.702
```

Regression of ETS model:

```
## ETS(M,A,M)
##
## Call:
##  ets(y = train)
##
##   Smoothing parameters:
##     alpha = 0.1921
##     beta  = 1e-04
##     gamma = 0.3473
##
##   Initial states:
##     l = 2312.142
##     b = 14.1286
##     s = 1.2455 1.3353 1.1819 1.0042 0.8577 0.5424
##            0.674 0.8941 0.9822 0.9819 1.0902 1.2106
##
##   sigma:  0.107
##
##      AIC     AICc      BIC
## 8269.384 8270.814 8339.090
```

Forecasting error of STL:

```
##                      ME      RMSE       MAE         MPE     MAPE
MASE
## Training set   18.01584  648.0922  472.9896  -0.1531076 10.37069
0.9103619
## Test set     -808.34898 1488.7873 1121.5660 -11.0529016 14.11813
2.1586750
##                    ACF1 Theil's U
## Training set -0.4193906        NA
## Test set      0.4023575 0.6925265
```

Forecasting error of ETS:

```
##                     ME     RMSE      MAE        MPE     MAPE
MASE
## Training set -15.34364 552.3410 403.2925 -0.9413968 8.146710
0.776216
## Test set     105.81581 694.0785 572.4448  0.1676528 6.648229
1.101783
##                    ACF1 Theil's U
## Training set 0.13643297        NA
## Test set     0.00634355 0.2855508
```

Results of *checkresiduals()*:

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,M)
## Q* = 38.616, df = 8, p-value = 5.791e-06
##
## Model df: 16.   Total lags used: 24
```

Calculation and reproduction of *tsCV()*:

```
## [1] 433

## [1] 1130.845

## [1] 1154.918

## [1] 1130.845

## [1] 1154.918
```