STATT680_Final Project_Model 1

Group-1

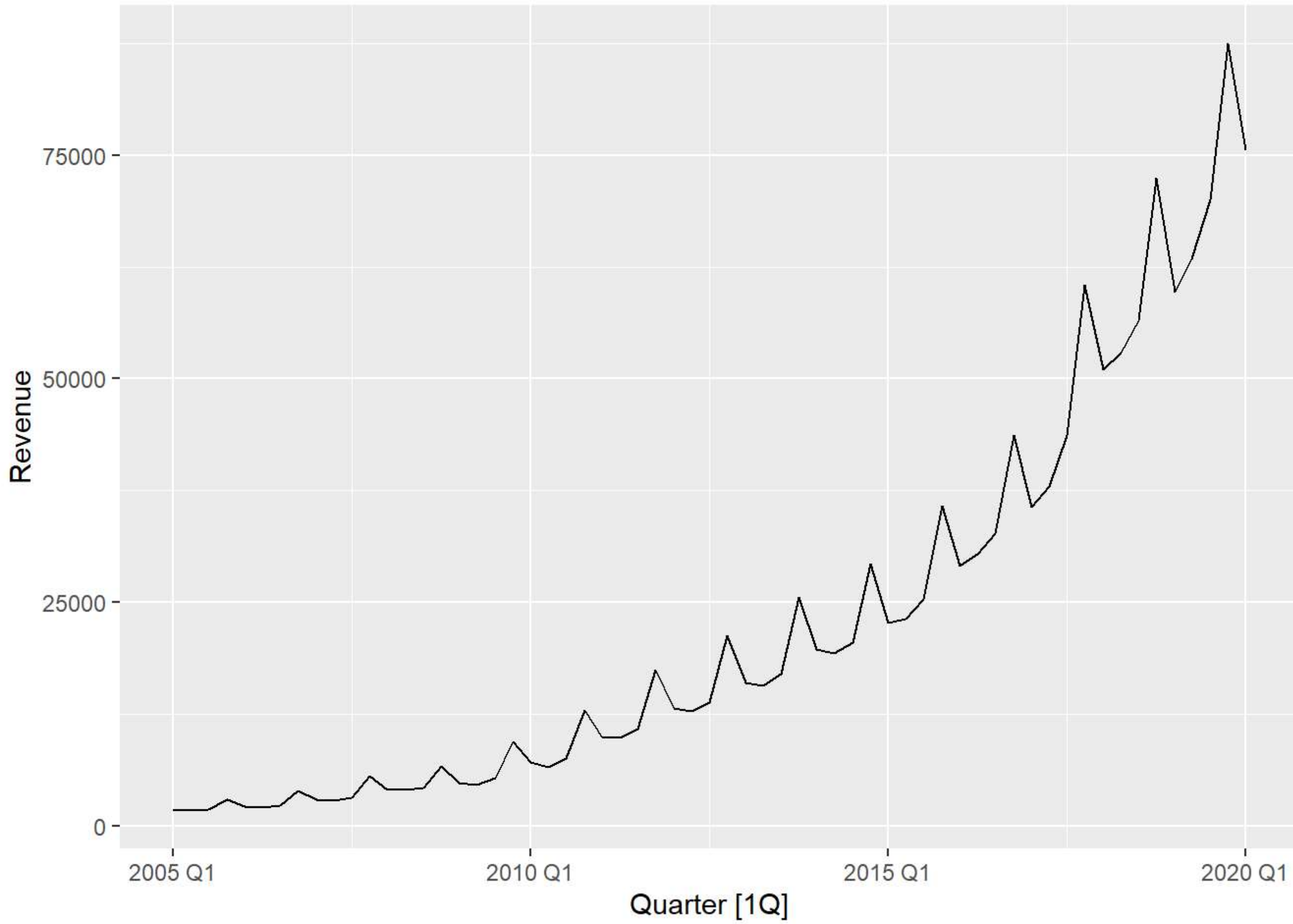
6/9/2022

```
library(fpp3)
library(readr)

# Read and convert the data in to tsibble

setwd("C:/Users/neela/Documents/Spring Quarter/Time series/Final Project")
amazon <- read.csv("Amazon.csv")

amazon <- amazon  %>%mutate(Quarter = yearquarter(Quarter)) %>% as_tsibble(index = 'Quarter')
```

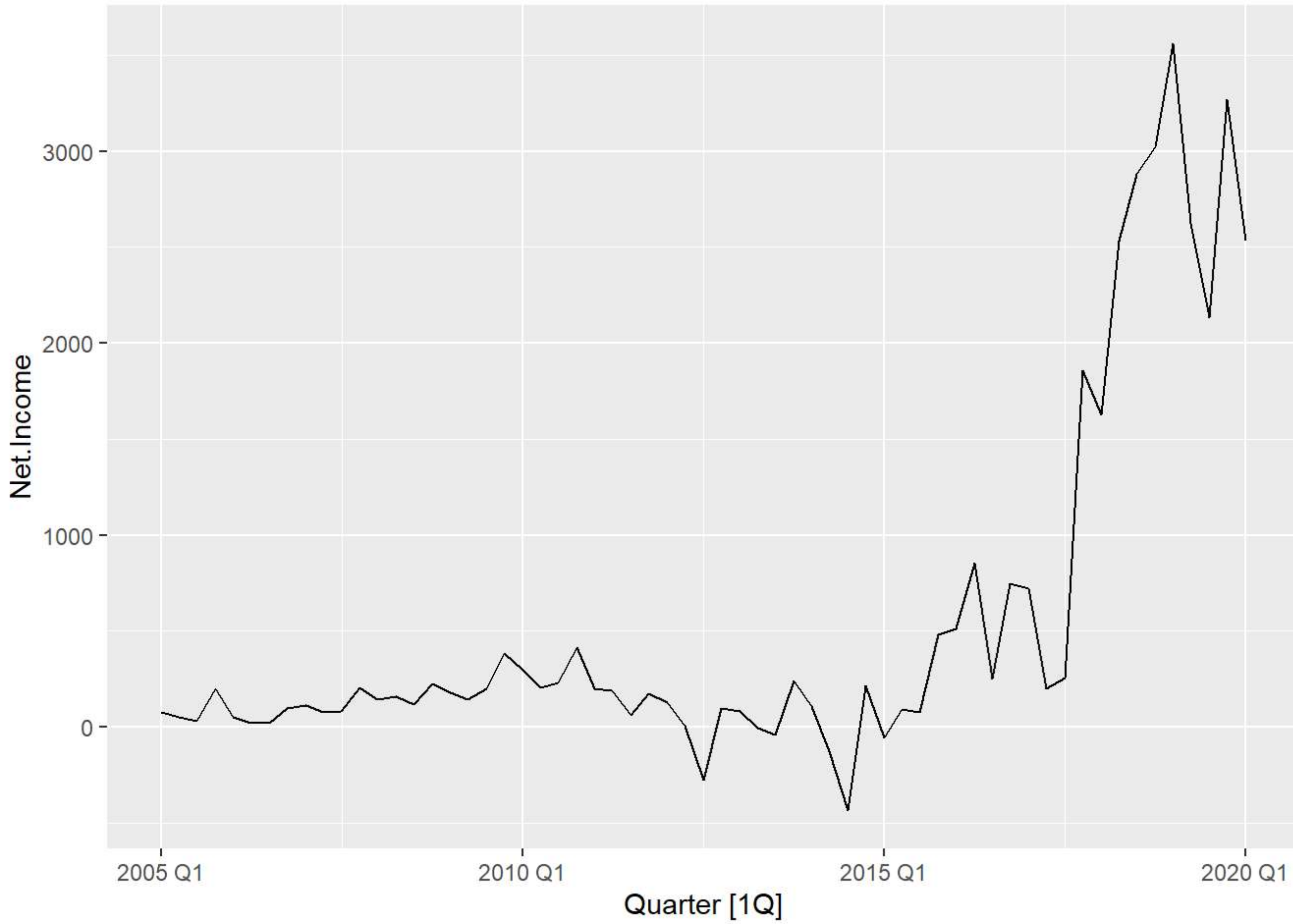
Data exploration:

In our data exploration, we wanted to see if Amazon Revenue and Net Income have a trend and/or seasonality. From plotting the data, we notice Revenue of Amazon is seasonal with a peak in Q4 and its revenue has a sharply upward trending. Both of those could be multiplicative. However, for Net Income, there seems no clear seasonality. Net Income of Amazon starts flat at the beginning of selected period, and suddenly increase around 2018.

```
amazon %>% autoplot(Revenue)
```




```
# There is seasonality and upward trending. Both could be multiplicative.

amazon %>% autoplot(Net.Income)
```




```
# There is no clear seasonality. Trend are flat at the beginning, and upward at around 2015 Q1

# As a first step of forecasting, we create a training set for period from Q1'2005 to Q4'2016.
amazon_train <- amazon %>% filter_index("2005 Q1" ~ "2016 Q4")
```

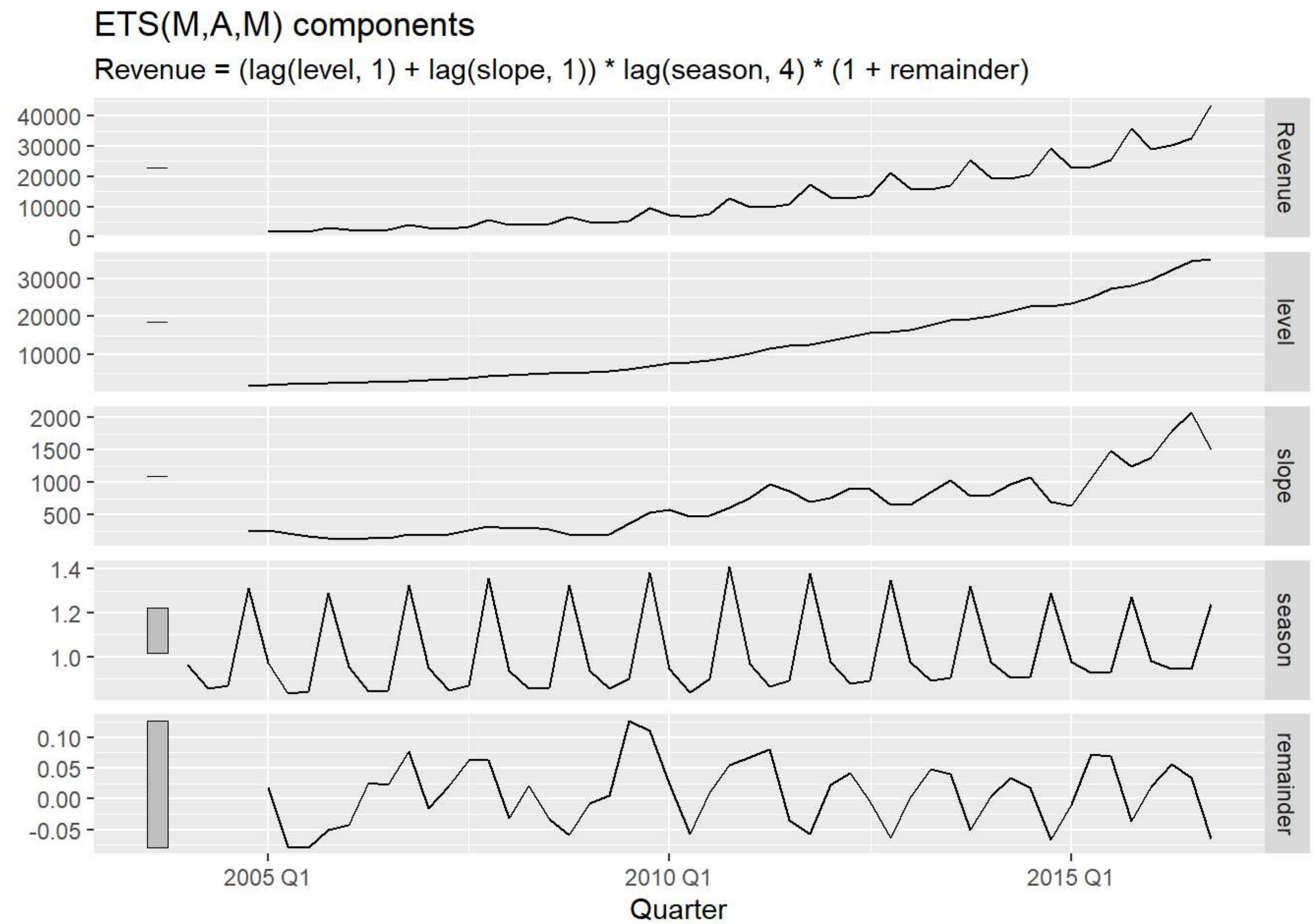
ETS Modelling

I. REVENUE FORECASTING:

```r
# R suggests that we should decide to use a model with components ETS(M,A,M)
fit_r <- amazon_train %>% model(ETS(Revenue))
report(fit_r)
```

```
## Series: Revenue
## Model: ETS(M,A,M)
##   Smoothing parameters:
##     alpha = 0.6252257
##     beta  = 0.2385308
##     gamma = 0.3747742
##
##   Initial states:
##      l[0]    b[0]     s[0]     s[-1]     s[-2]     s[-3]
##  1690.494 249.263 1.314536 0.8663746 0.8562489 0.9628407
##
##   sigma^2:  0.0032
##
##      AIC     AICc      BIC
## 792.2688 797.0057 809.1096
```

```r
# The following step is  to visualize components
components(fit_r) %>% autoplot() + labs(title = "ETS(M,A,M) components")
```

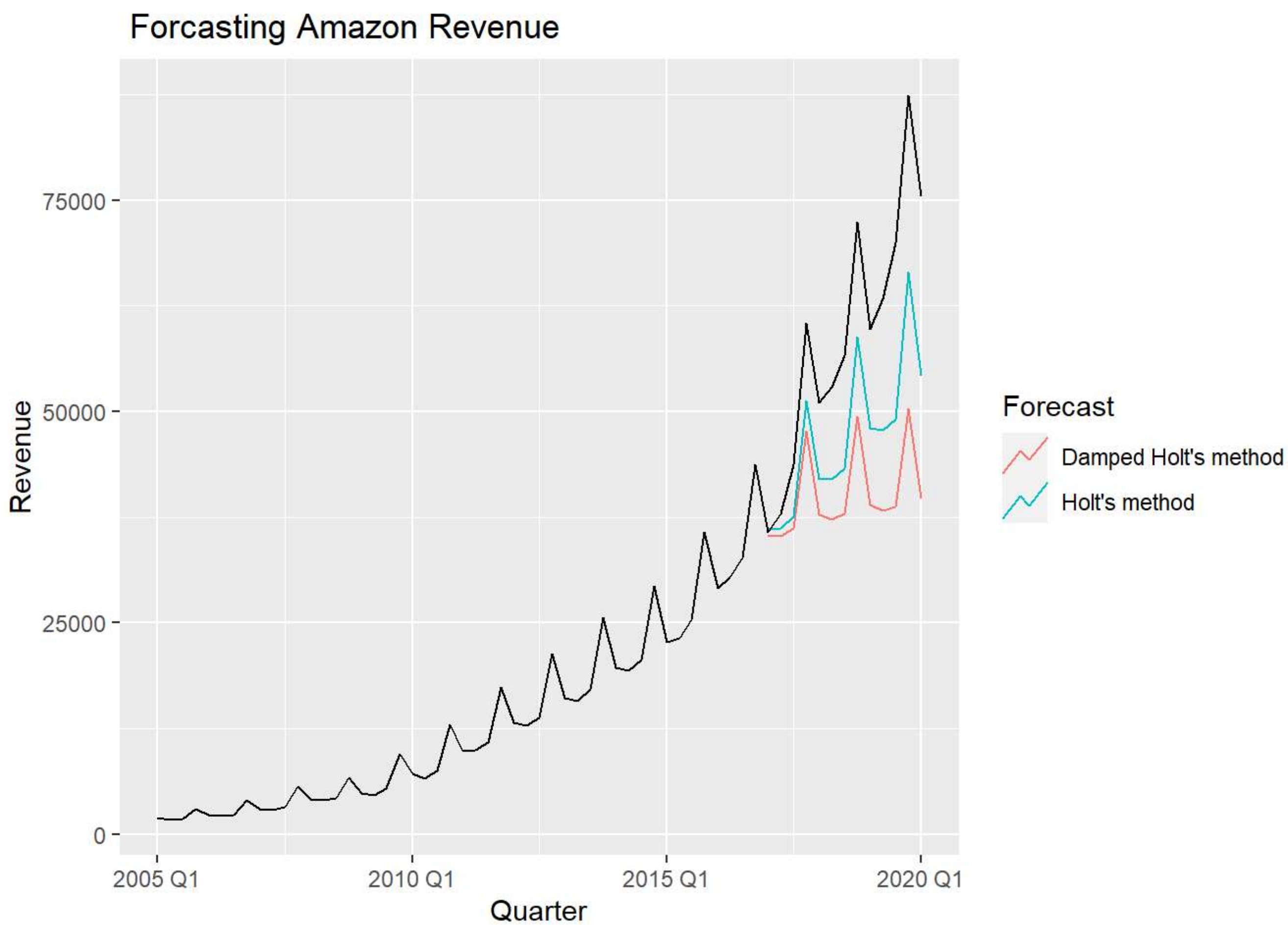


As the training dataset is set between Q1'2005 to Q4'2016. We will be forecasting the next 13 periods for Amazon's Revenue. Our forecast includes two methods: Holt's and Damped Holt's in which Damped Holt's uses phi = 0.8.

```
amazon_r_fc <- amazon_train %>%
  model(
    `Holt's method` = ETS(Revenue ~ error("M") +
                            trend("A") + season("M")),
    `Damped Holt's method` = ETS(Revenue ~ error("M") +
                                   trend("Ad", phi = 0.8) + season("M"))
  ) %>%
  forecast(h = 13)
```

By plotting forecast result, we come up our first conclusion that Holt's method performs better than Damped Holt's one as the forecast values of Holt's medthod are closer to the actual values. Also, the accuracy of Holt is also higher showing in its much lower RMSE and MAPE compared to the other method. However, both models seem to have significantly high forecast errors so exponential smoothing models do not appear to be the most accurate model to forecast Amazon's Revenue.

```
amazon_r_fc %>%  autoplot(amazon, level = NULL) +
  labs(title = " Forcasting Amazon Revenue",
       y = "Revenue") +
  guides(colour = guide_legend(title = "Forecast"))
```



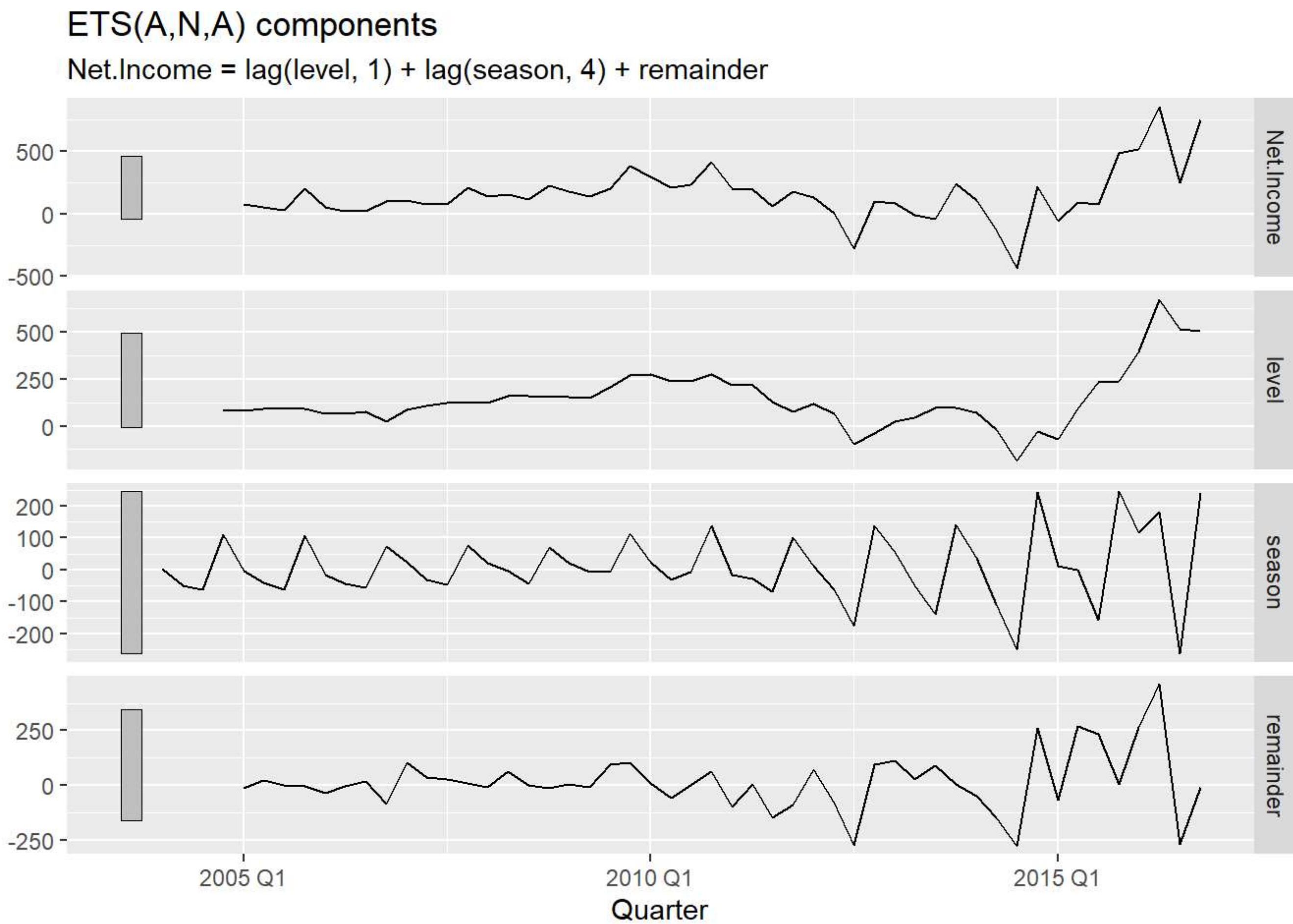
```r
# Accuracy
accuracy(amazon_r_fc,amazon)
```

```
## # A tibble: 2 x 10
##   .model               .type     ME   RMSE    MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>                <chr>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Damped Holt's method Test  18779. 21886. 18779.  28.9  28.9  6.48  6.27 0.752
## 2 Holt's method        Test  11853. 13586. 11908.  18.5  18.6  4.11  3.89 0.713
```

ETS MODELLING

II. NET INCOME FORECASTING:

```
## Series: Net.Income
## Model: ETS(A,N,A)
##   Smoothing parameters:
##     alpha = 0.6054508
##     gamma = 0.3945489
##
##   Initial states:
##      l[0]     s[0]     s[-1]     s[-2]    s[-3]
##  87.39701 110.3798 -62.94851 -50.17626 2.744993
##
##   sigma^2:  20703.63
##
##      AIC     AICc      BIC
## 670.4352 673.2352 683.5336
```

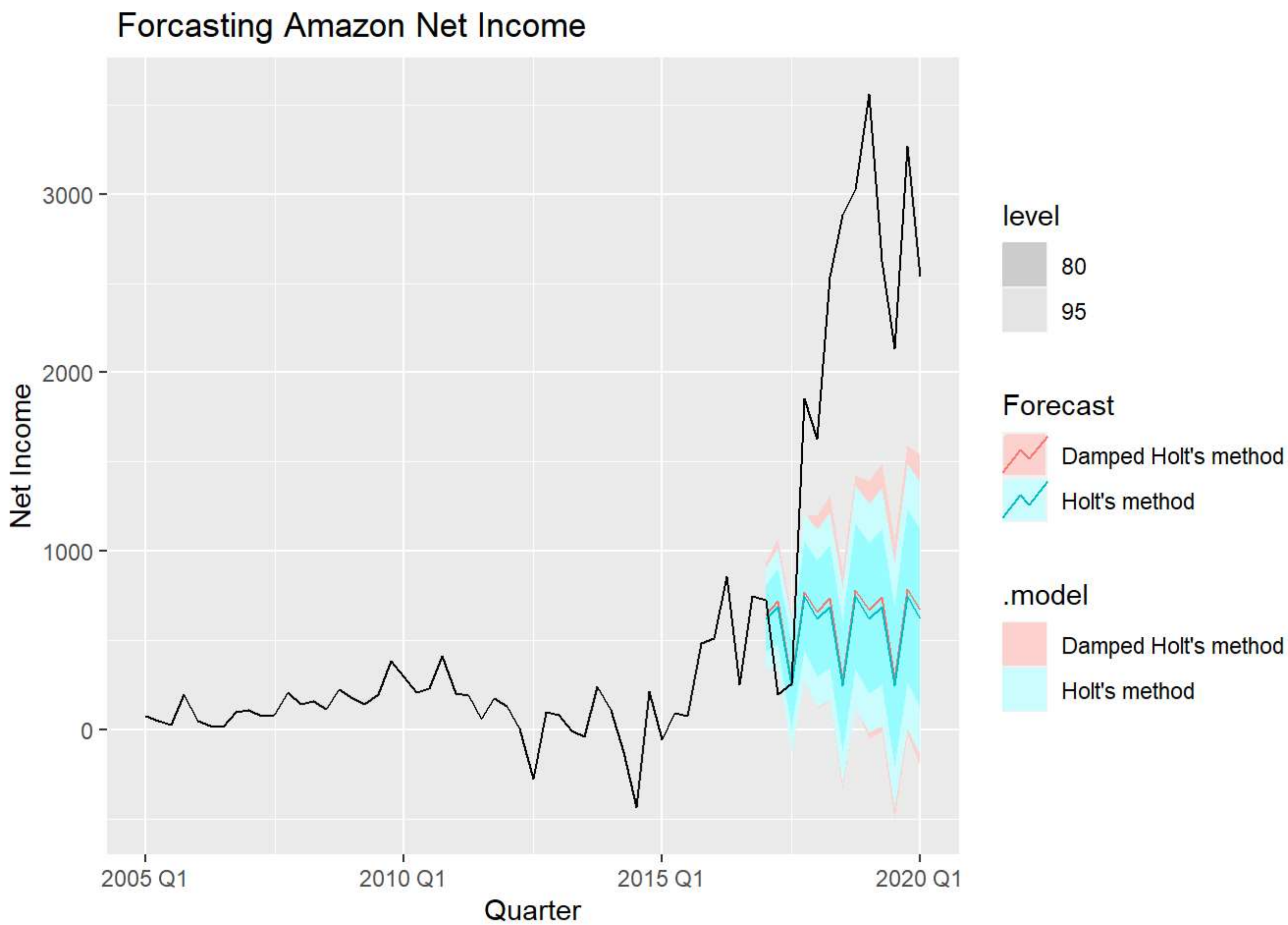


As the training dataset is set between Q1’2005 to Q4’2016. We will be forecasting the next 13 periods for Amazon’s Net Income. Our forecast includes two methods: Holt’s and Damped Holt’s in which Damped Holt’s uses phi = 0.8.

```
amazon_n_fc <- amazon_train %>%
  model(
    `Holt's method` = ETS(Net.Income ~ error("A") +
                             trend("N") + season("A")),
    `Damped Holt's method` = ETS(Net.Income ~ error("A") +
                                    trend("Ad", phi = 0.8) + season("A"))
  ) %>%
  forecast(h = 13)
```

By plotting forecast result, we observe that both of two method poorly perform on the testing dataset. None of those models is able to forecast a sharp increase in 2018. We would not recommend using this exponential smoothing models for forecasting Amazon's Net Income. We think forecasting should be more practical if we try to apply forecasting methods on the dataset from Q1'2018 until YTD.

```
amazon_n_fc %>%  autoplot(amazon) +
  labs(title = " Forcasting Amazon Net Income",
       y = "Net Income") +
  guides(colour = guide_legend(title = "Forecast"))
```




```
# Accuracy
accuracy(amazon_n_fc,amazon)
```

```
## # A tibble: 2 x 10
##   .model               .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>                <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Damped Holt's method Test  1479. 1807. 1561.  37.3  78.6  10.9  8.43 0.728
## 2 Holt's method        Test  1514. 1839. 1590.  40.5  78.9  11.1  8.57 0.730
```

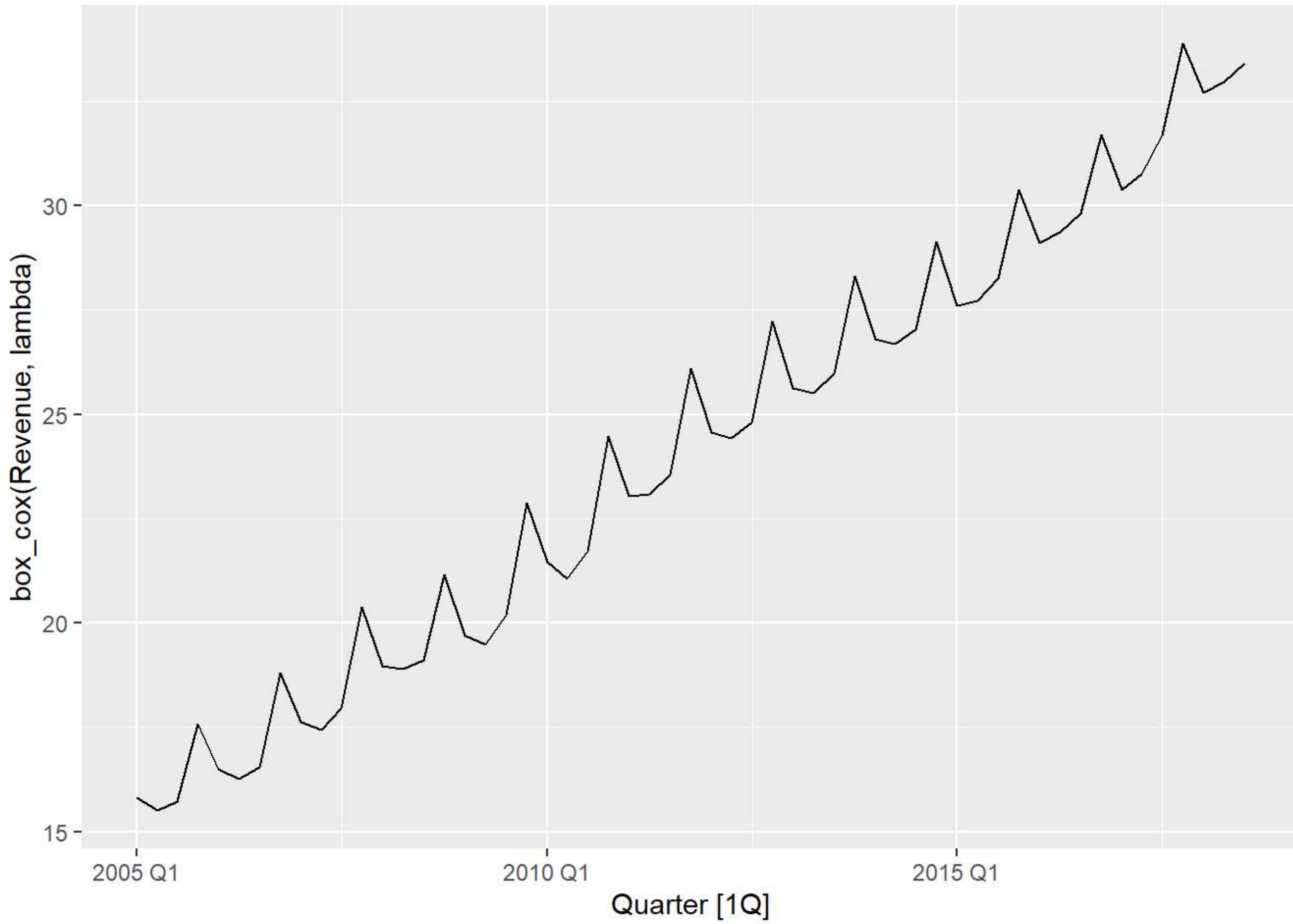
ARIMA Modelling

Data has a lot of variation, it increases as we compare initial years to later years, hence we are using box-cox transformation to curb that.

```
# creating a 90-10 split of test and training
AMZ_TS_train <- amazon %>% filter_index("2005 Q1" ~ "2018 Q3")

#checking optimal lambda for box-cox transformation
lambda <- AMZ_TS_train %>%
  features(Revenue, features = guerrero) %>%
  pull(lambda_guerrero)
lambda
```

```
## [1] 0.176449
```

```
#Checking auto-plot for transformed data
AMZ_TS_train %>%
  autoplot(box_cox(Revenue, lambda))
```



Checking if the datais stationary

```
#KPSS test to check whether data is stationary
AMZ_TS_train %>% features(box_cox(Revenue, lambda), unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1      1.47        0.01
```

As the p-value of KPSS test is 0.01, which is smaller than level of significance(0.05) so we reject the null hypothesis that says the data is stationary, therefore data is NOT stationary. So, we move on and apply appropriate differencing in the below code

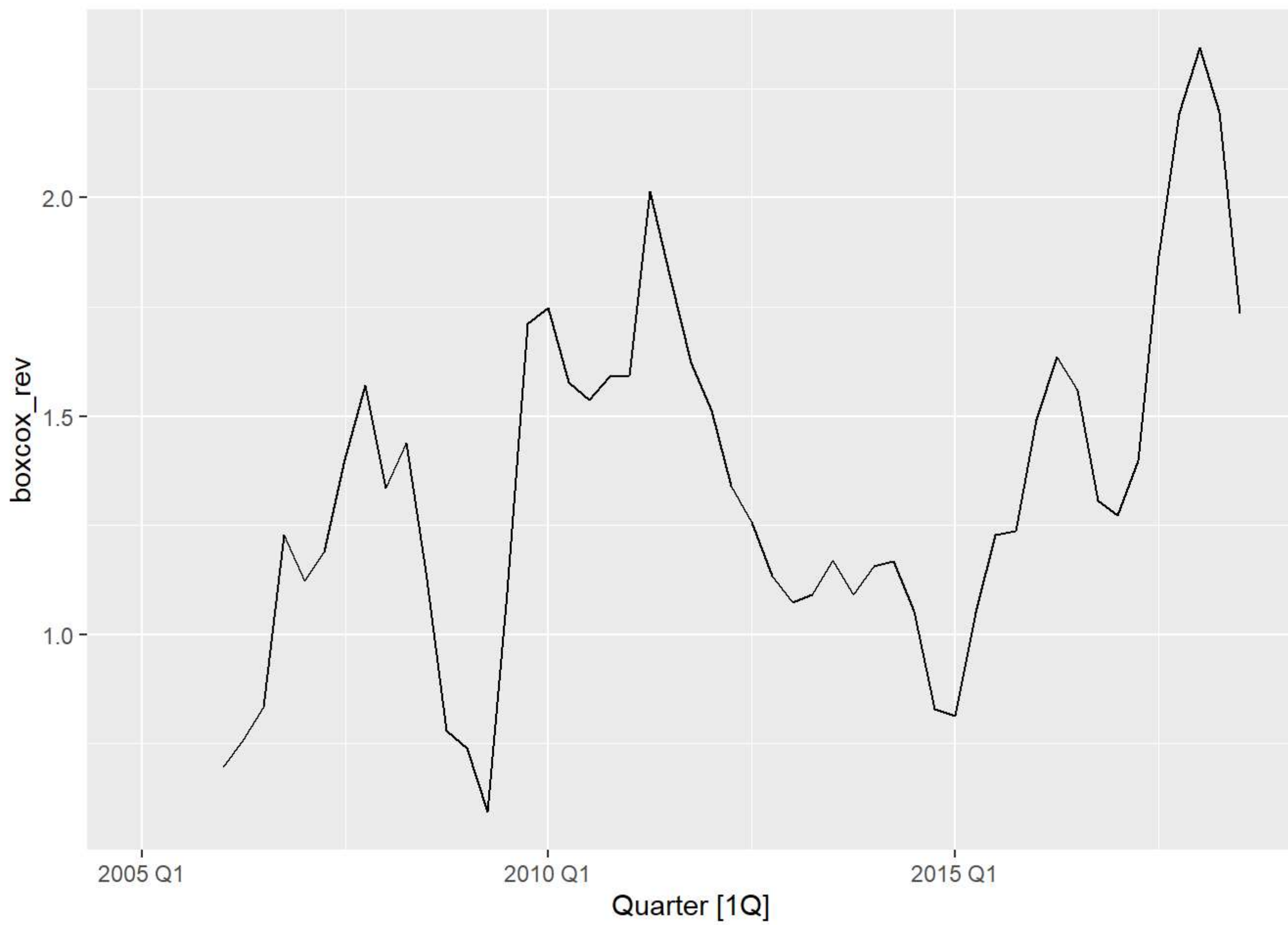
```r
# Checking how many non-seasonal differences required
AMZ_TS_train %>%
  features(box_cox(Revenue, lambda), unitroot_ndiffs)
```

```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      1
```

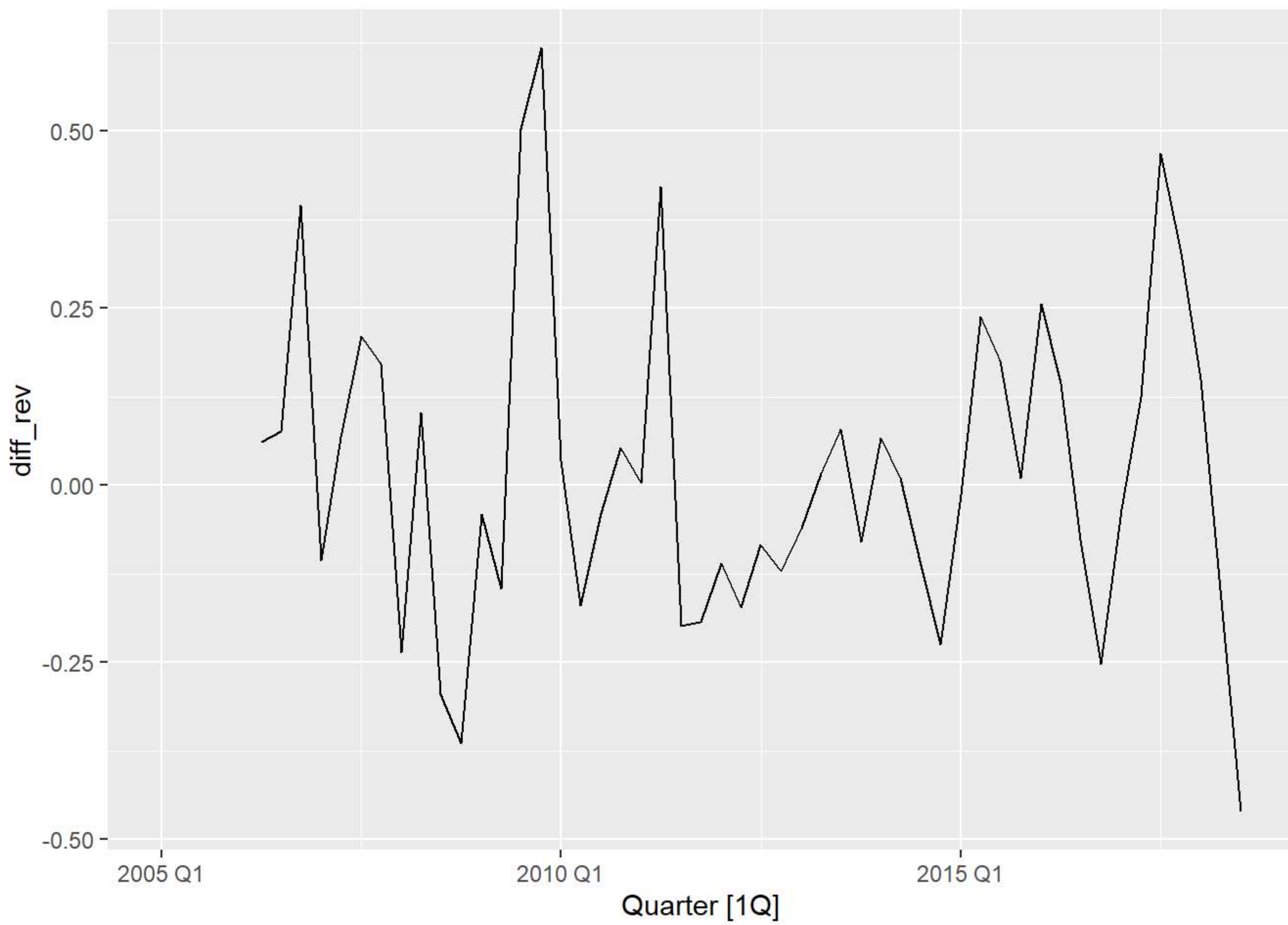
```r
# Checking how many seasonal differences required
AMZ_TS_train %>%
  features(box_cox(Revenue, lambda), unitroot_nsdiffs)
```

```
## # A tibble: 1 x 1
##   nsdiffs
##     <int>
## 1       1
```

```r
# Applying seasonal differences
AMZ_TS_train_mod <- AMZ_TS_train%>%
  mutate(boxcox_rev = difference(box_cox(Revenue, lambda), 4))
AMZ_TS_train_mod %>% autoplot(boxcox_rev)
```




```
# Applying non-seasonal differences
AMZ_TS_train_mod <- AMZ_TS_train_mod %>%
  mutate(diff_rev = difference(boxcox_rev))
AMZ_TS_train_mod %>% autoplot(diff_rev)
```




```
#KPSS test to check whether data is stationary
AMZ_TS_train_mod %>% features(diff_rev, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0554         0.1
```

The p-value is now, 0.1, which is larger than the level of significance(0.05) so we fail to reject the null hypothesis and the data is stationary now.

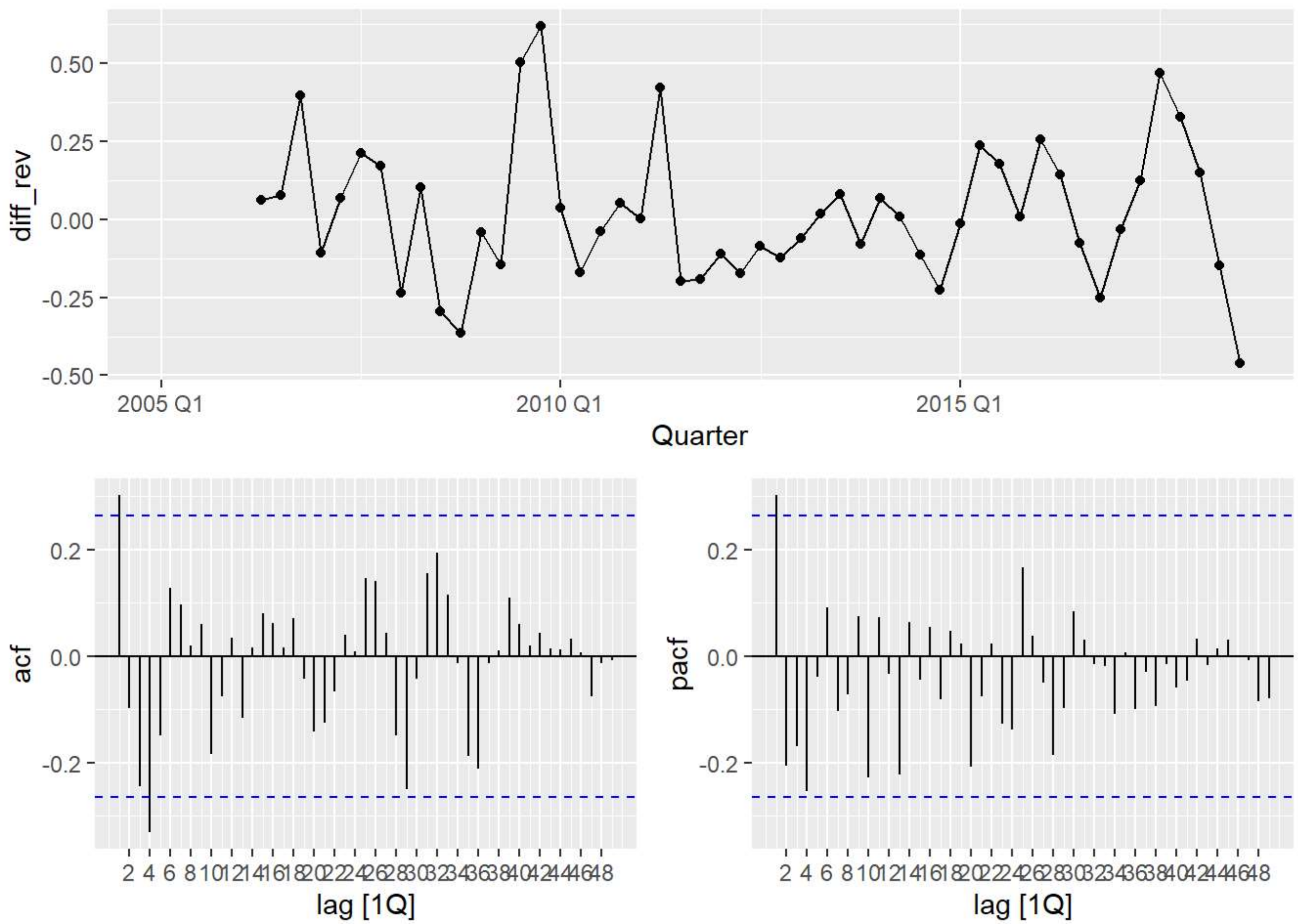
I. REVENUE FORECASTING:

```r
# checking best model suggested by R
fit_auto_rev <- AMZ_TS_train %>%
  model(ARIMA(Revenue))

# best ARIMA model suggested by R
fit_auto_rev
```

```
## # A mable: 1 x 1
##            `ARIMA(Revenue)`
##                     <model>
## 1 <ARIMA(3,1,2)(1,1,0)[4]>
```

We, now, see that the PACF lags are exponentially decaying so we checked for MA process, there were two significant non-seasonal lags in the ACF plot, so we chose q=2 and p=0, and there was one seasonal significant lag in ACF therefore we chose Q=1 and P=0. Therefore, we choose another model ARIMA(0,1,2)(0,1,1)4

```r
# Checking ACF and PCAF
AMZ_TS_train_mod %>%
  gg_tsdisplay(diff_rev, plot_type='partial', lag = 200)
```




```
#ARIMA (0,1,2)(0,1,1)4
fit_arima31 <- AMZ_TS_train %>%
  model(ARIMA(Revenue ~ pdq(0,1,2) + PDQ(0,1,1)))
```

Below is the code for checking AICc of both the models

```
#AICc for auto
report(fit_auto_rev)
```

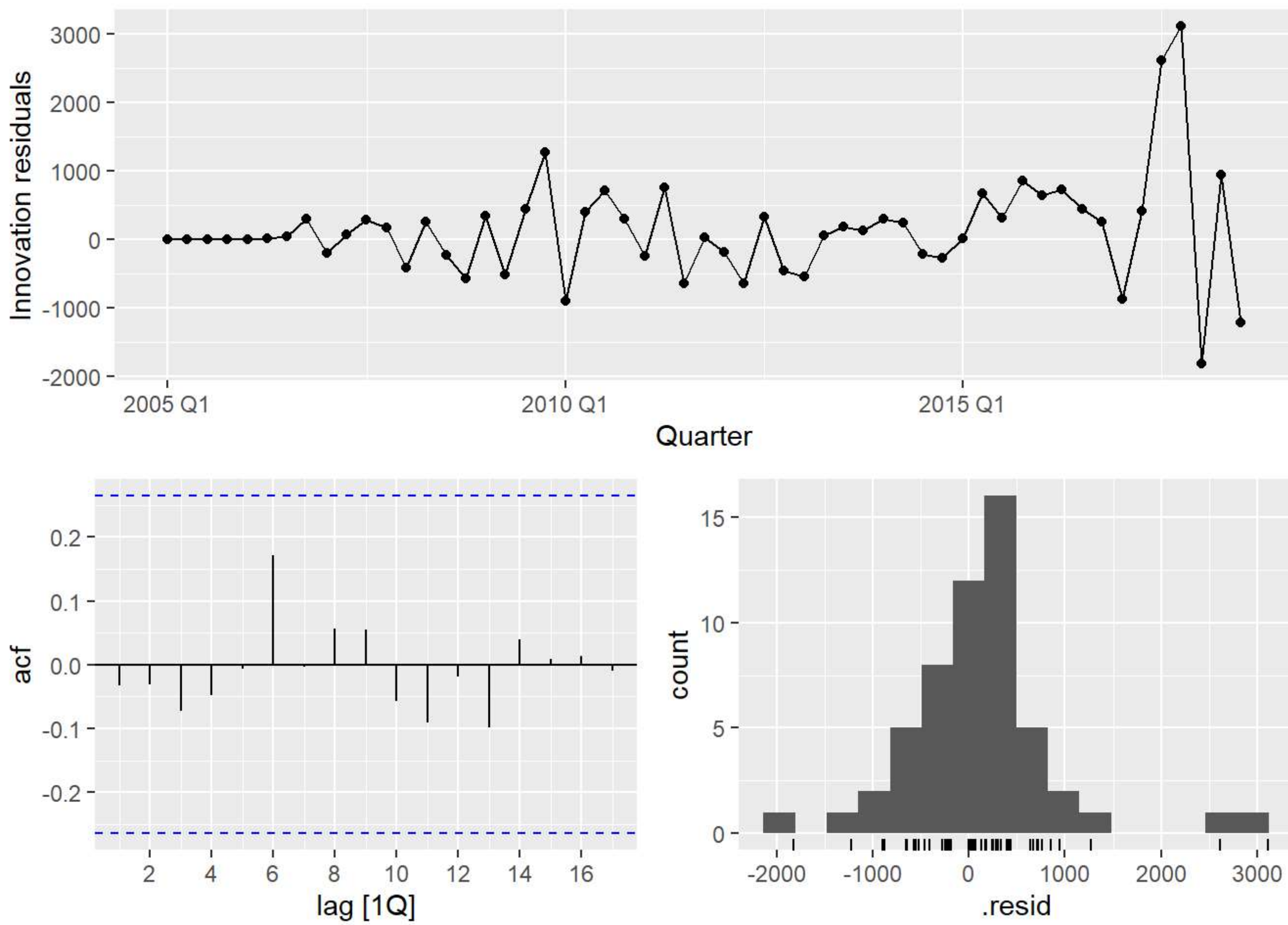
```
## Series: Revenue
## Model: ARIMA(3,1,2)(1,1,0)[4]
##
## Coefficients:
##          ar1      ar2      ar3      ma1     ma2    sar1
##       0.8225  -0.3332  -0.4448  -0.5013  0.0862  0.8577
## s.e.  0.3420   0.4096   0.2827   0.3557  0.2587  0.0815
##
## sigma^2 estimated as 742240:  log likelihood=-409.01
## AIC=832.02   AICc=834.69   BIC=845.41
```

```
#ARIMA (3,1,2)(1,1,0)4
report(fit_arima31)
```

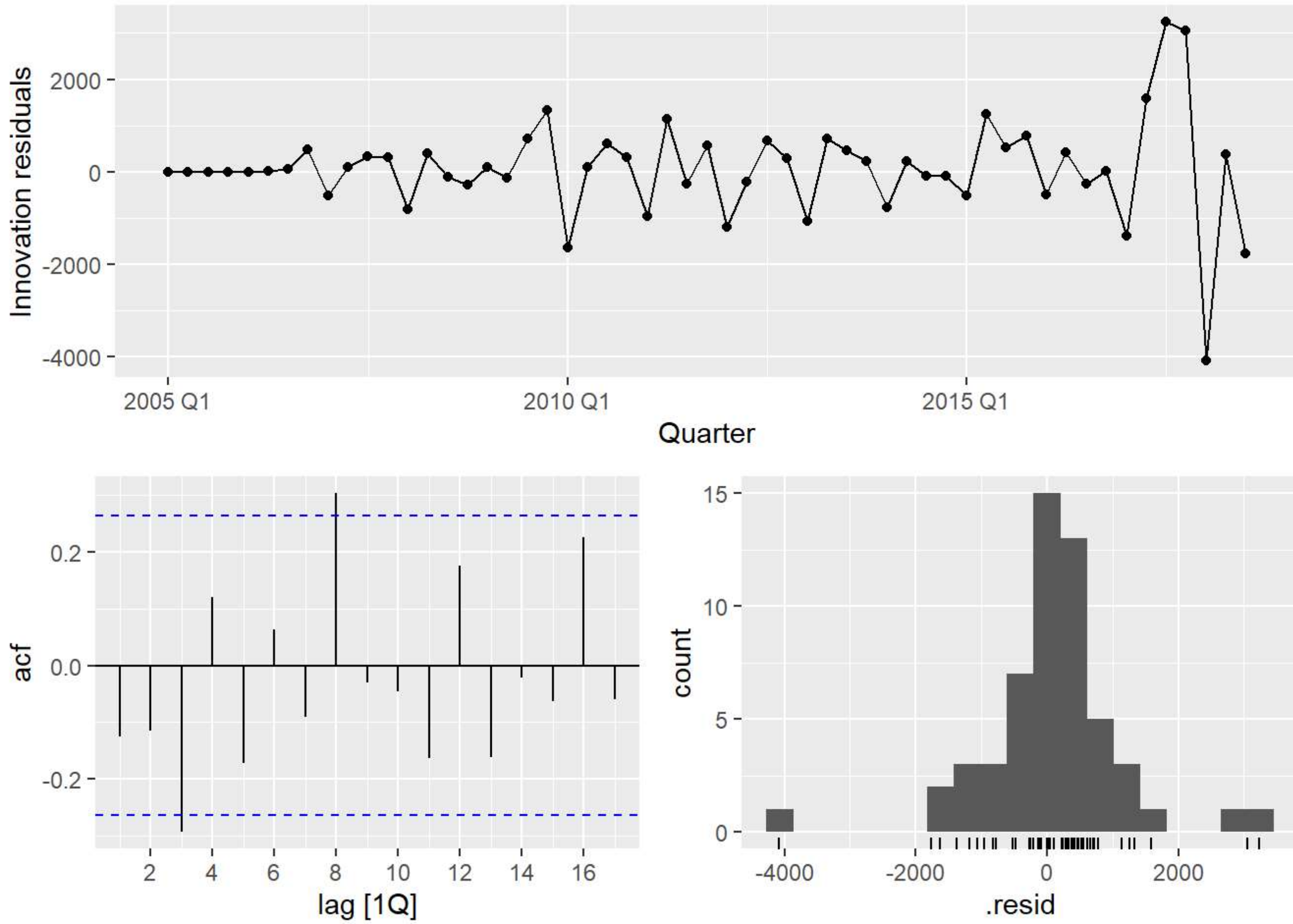
```
## Series: Revenue
## Model: ARIMA(0,1,2)(0,1,1)[4]
##
## Coefficients:
##          ma1     ma2    sma1
##       0.5884  0.4449  0.3568
## s.e.  0.1589  0.1830  0.1428
##
## sigma^2 estimated as 1312761:  log likelihood=-422.21
## AIC=852.41   AICc=853.3   BIC=860.06
```

AICc for auto model is smaller than arima31, therefore the auto model is better than arima31.

```
# Residual analysis for auto
fit_auto_rev %>%
  gg_tsresiduals()
```




```
# Residual analysis for arima01
fit_arima31 %>%
  gg_tsresiduals()
```




```
# White noise test auto
augment(fit_auto_rev) %>% features(.innov, ljung_box, lag=10, dof=2)
```

```
## # A tibble: 1 x 3
##   .model         lb_stat lb_pvalue
##   <chr>            <dbl>     <dbl>
## 1 ARIMA(Revenue)    3.10     0.928
```

```
# White noise test for arima01
augment(fit_arima31) %>% features(.innov, ljung_box, lag=10, dof=2)
```

```
## # A tibble: 1 x 3
##   .model                                          lb_stat lb_pvalue
##   <chr>                                             <dbl>     <dbl>
## 1 ARIMA(Revenue ~ pdq(0, 1, 2) + PDQ(0, 1, 1))    16.8    0.0326
```

P-values for white noise test for both auto model and arima31 model are larger than the level of significance 0.05, Therefore, residuals in auto model and arima31 model are white noise.

ARIMA MODELLING

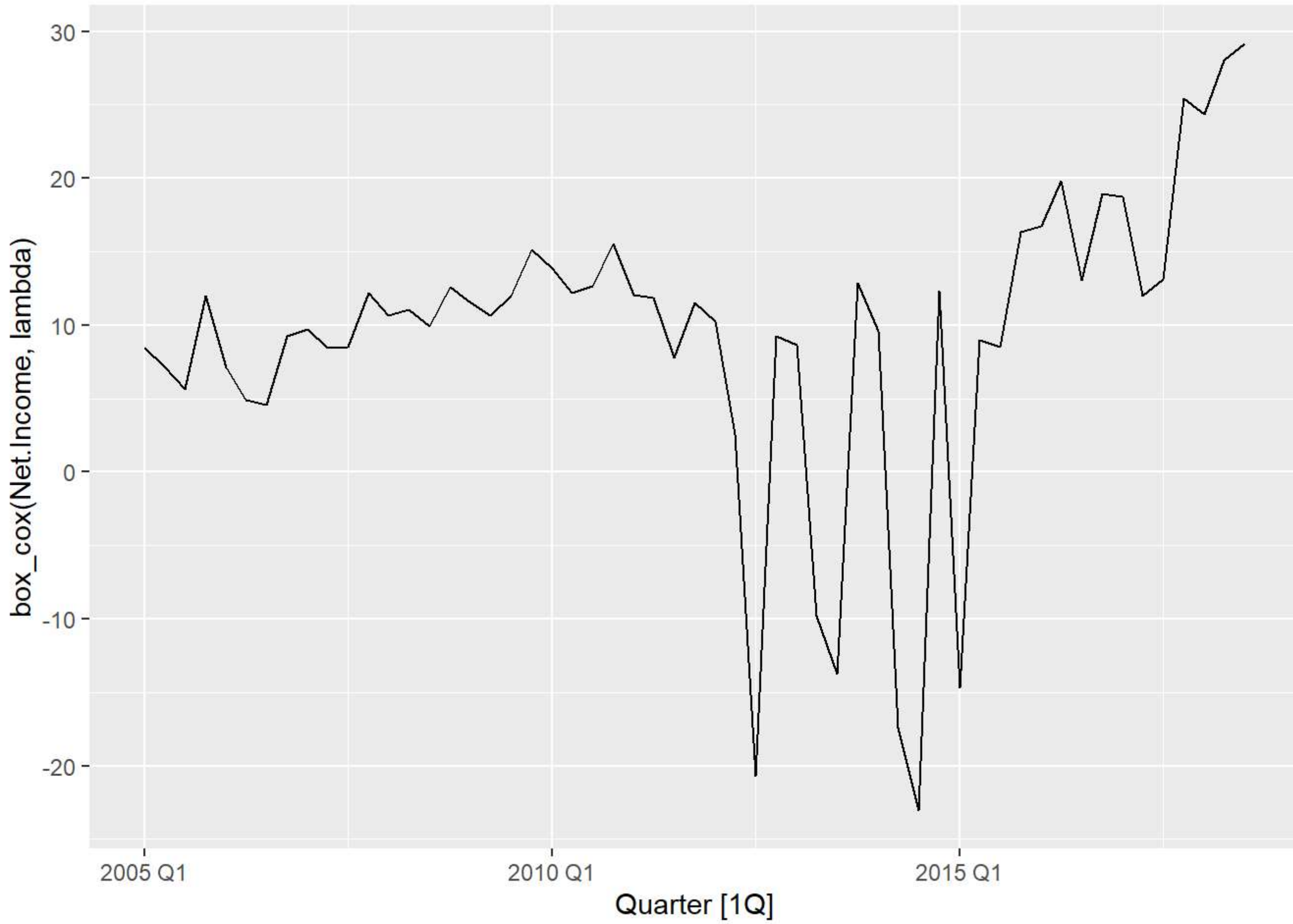
II. NET INCOME FORECASTING:

Data has a lot of variation, hence we are using box-cox transformation to curb that.

```
#checking optimal lambda for box-cox transformation
lambda <- AMZ_TS_train %>%
  features(Net.Income, features = guerrero) %>%
  pull(lambda_guerrero)
lambda
```

```
## [1] 0.2768266
```

```
#Checking auto-plot for transformed data
AMZ_TS_train %>%
  autoplot(box_cox(Net.Income, lambda))
```



Checking if the datais stationary

```
#KPSS test to check whether data is stationary
AMZ_TS_train %>% features(box_cox(Net.Income, lambda), unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1     0.224         0.1
```

As the p-value of KPSS test is 0.1, which is larger than level of significance(0.05) so we reject the null hypothesis that says the data is stationary, therefore data is stationary. However, we do not see a constant variation in data, so, we move on and apply appropriate differencing in the below code

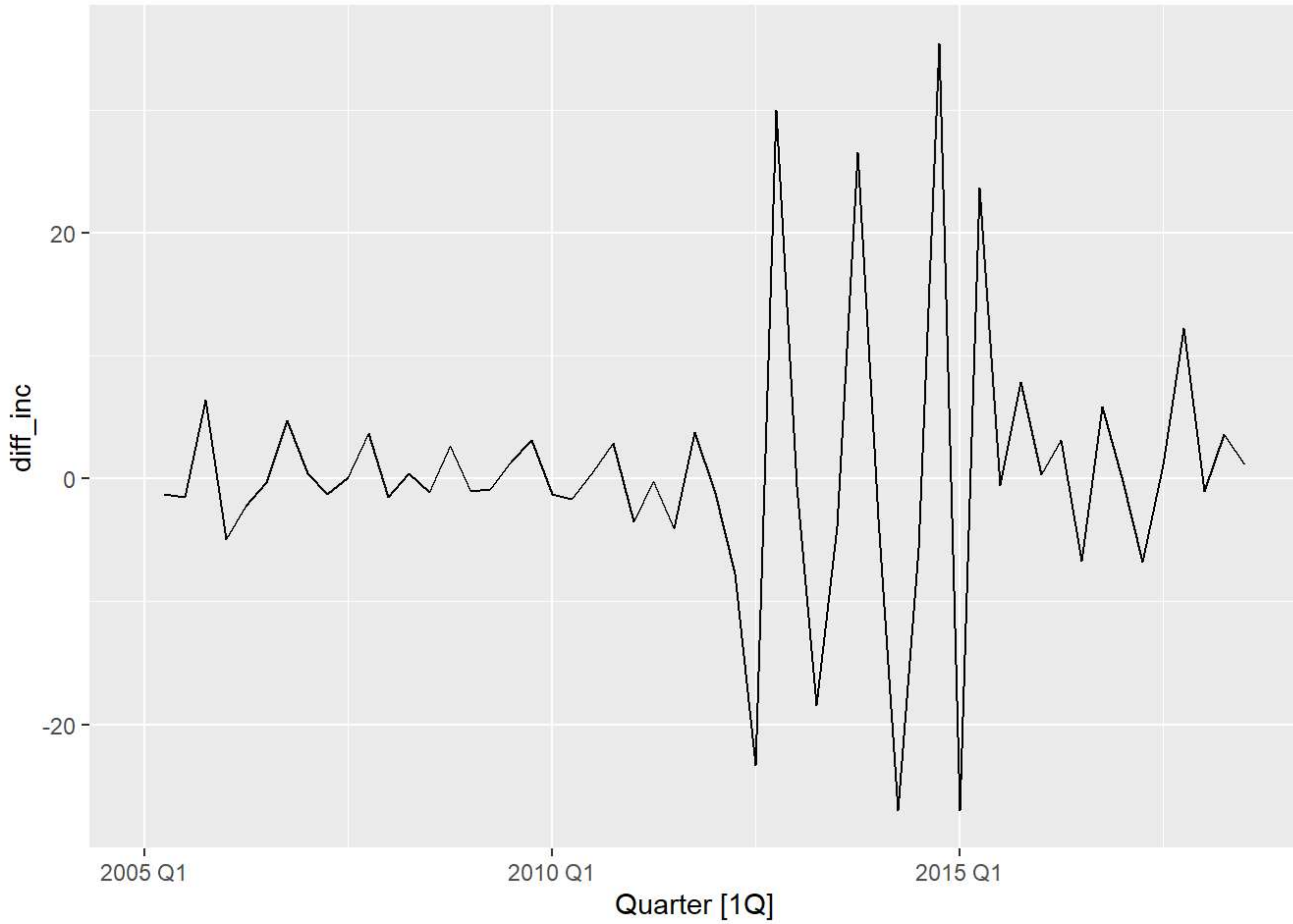
```
# Checking how many non-seasonal differences required
AMZ_TS_train %>%
  features(box_cox(Net.Income, lambda), unitroot_ndiffs)
```

```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      0
```

```
# Checking how many seasonal differences required
AMZ_TS_train %>%
  features(box_cox(Net.Income, lambda), unitroot_nsdiffs)
```

```
## # A tibble: 1 x 1
##   nsdiffs
##     <int>
## 1       0
```

```
# Applying non-seasonal differences
AMZ_TS_train_mod <- AMZ_TS_train_mod %>%
  mutate(diff_inc = difference(box_cox(Net.Income, lambda)))
AMZ_TS_train_mod %>% autoplot(diff_inc)
```



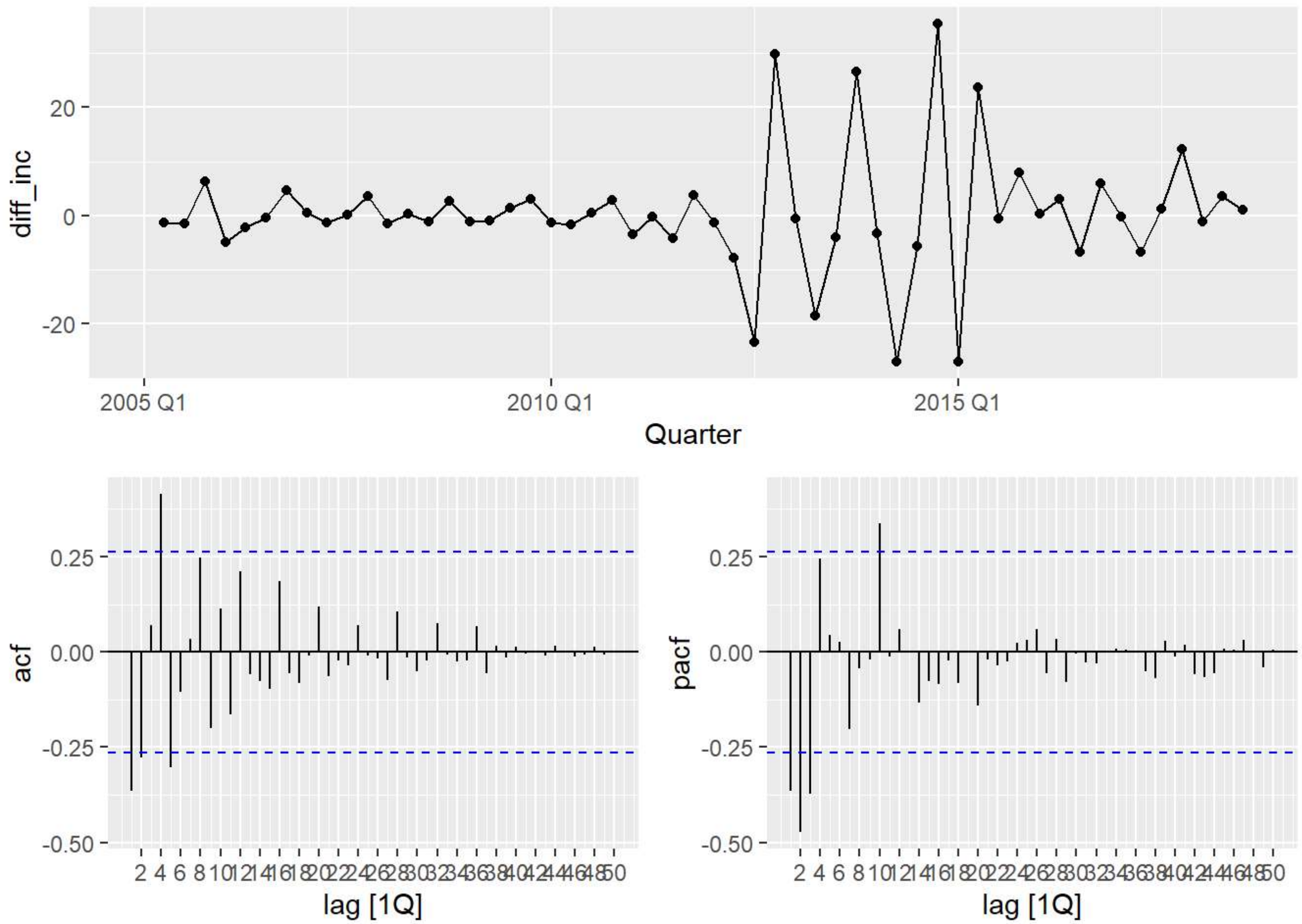
II. NET INCOME FORECASTING:

```
# checking best model suggested by R
fit_auto_inc <- AMZ_TS_train %>%
  model(ARIMA(Net.Income))

# best ARIMA model suggested by R
fit_auto_inc
```

```
## # A mable: 1 x 1
##                 `ARIMA(Net.Income)`
##                             <model>
## 1 <ARIMA(0,1,0)(2,0,0)[4] w/ drift>
```

We, now, see that the ACF lags are exponentially decaying so we checked for AR process, there were 4 significant non-seasonal lags in the PACF plot, so we chose p=4 and q=0. Therefore, we choose another model ARIMAARIMA(4,1,0), we are not using seasonal arima because we do not see any seasonality here.

```
# Checking ACF and PCAF
AMZ_TS_train_mod %>%
  gg_tsdisplay(diff_inc, plot_type='partial', lag = 50)
```




```
#ARIMA (4,1,0)
fit_arima410 <- AMZ_TS_train %>%
  model(ARIMA(Net.Income ~ pdq(4,1,0)))
```

Below is the code for checking AICc of both the models

```
#AICc for auto
report(fit_auto_inc)
```

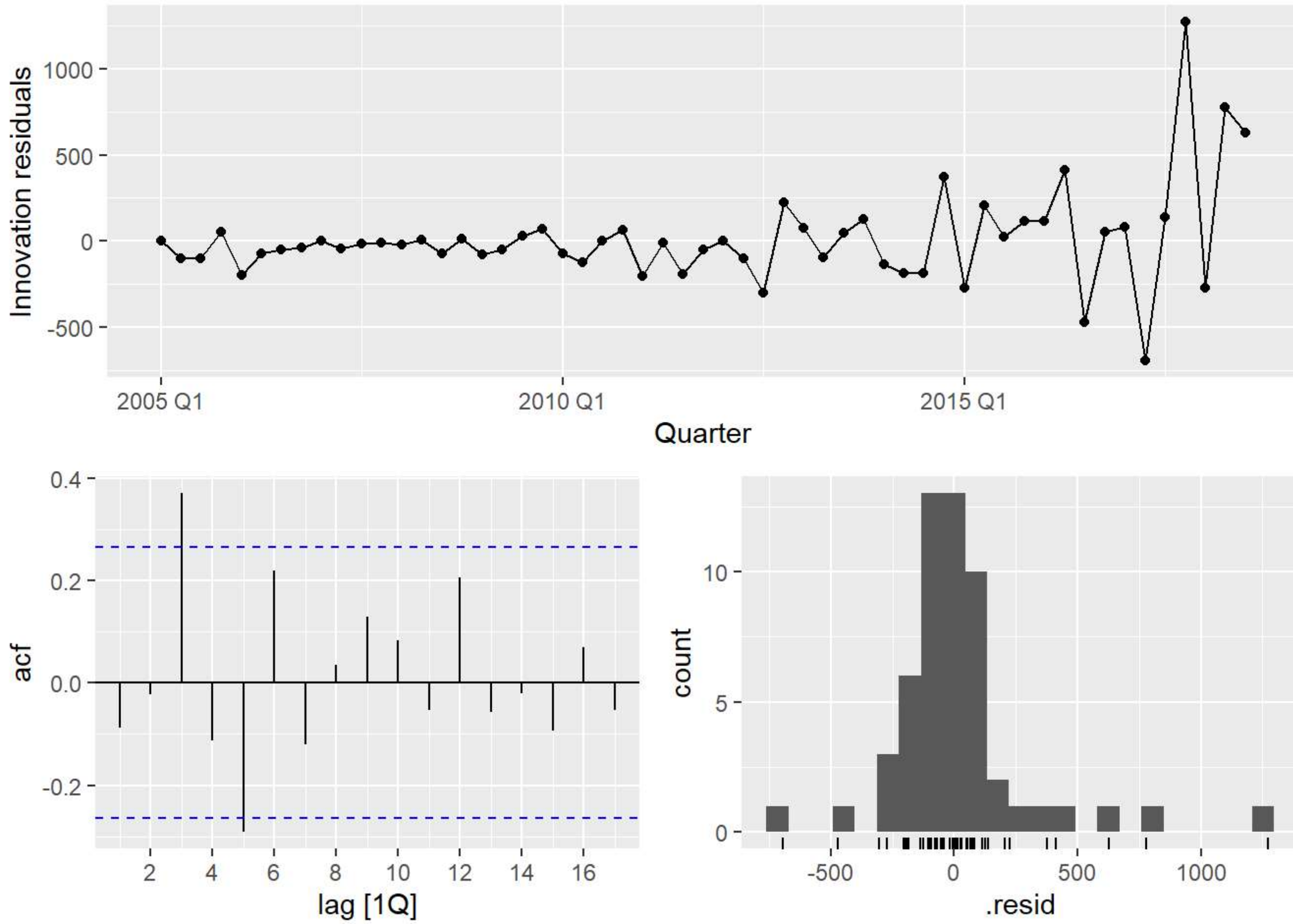
```
## Series: Net.Income
## Model: ARIMA(0,1,0)(2,0,0)[4] w/ drift
##
## Coefficients:
##         sar1    sar2  constant
##       0.1683  0.5304   31.0910
## s.e.  0.1537  0.1628   31.3839
##
## sigma^2 estimated as 85351:  log likelihood=-383.25
## AIC=774.5   AICc=775.31   BIC=782.45
```

```
#AICc ARIMA (4,1,0)
report(fit_arima410)
```

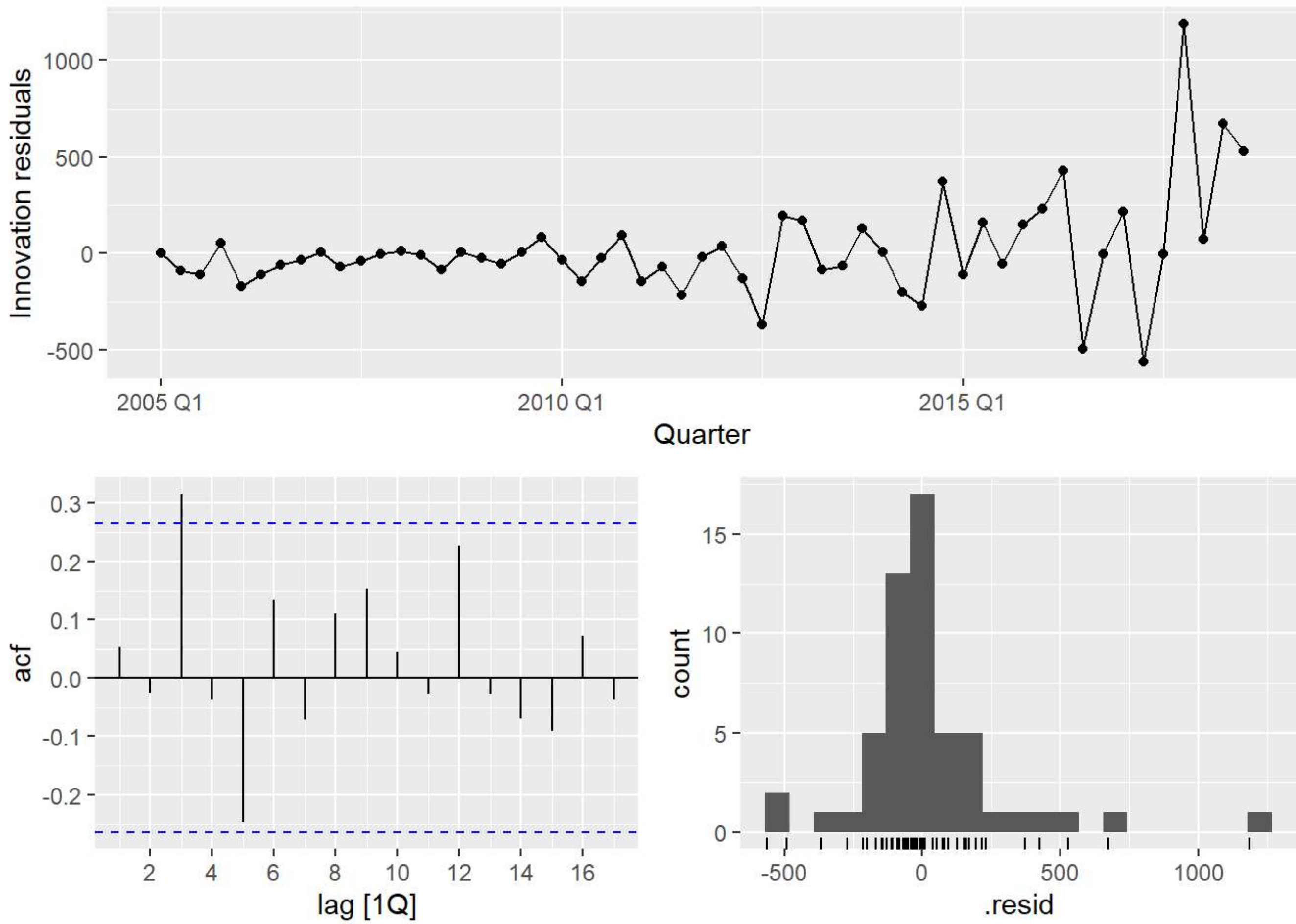
```
## Series: Net.Income
## Model: ARIMA(4,1,0)(1,0,1)[4] w/ drift
##
## Coefficients:
##           ar1     ar2     ar3     ar4     sar1     sma1  constant
##       -0.1181  0.0206  0.0819  0.8953  -0.5451  -0.3313   17.6081
## s.e.   0.0771  0.0599  0.0784  0.0746   0.2065   0.2162   18.0911
##
## sigma^2 estimated as 79669:  log likelihood=-379.59
## AIC=775.18   AICc=778.38   BIC=791.09
```

AICc for auto model is smaller than arima410, but they may not be comparable based on AICc as one is simple ARIMA model and the other is seasonal arima model.

```
# Residual analysis for auto
fit_auto_inc %>%
  gg_tsresiduals()
```




```
# Residual analysis for arima410
fit_arima410 %>%
  gg_tsresiduals()
```




```
# White noise test auto
augment(fit_auto_inc) %>% features(.innov, ljung_box, lag=10, dof=2)
```

```
## # A tibble: 1 x 3
##   .model            lb_stat lb_pvalue
##   <chr>               <dbl>     <dbl>
## 1 ARIMA(Net.Income)    20.5   0.00859
```

```
# White noise test for arima410
augment(fit_arima410) %>% features(.innov, ljung_box, lag=10, dof=2)
```

```
## # A tibble: 1 x 3
##   .model                           lb_stat lb_pvalue
##   <chr>                              <dbl>     <dbl>
## 1 ARIMA(Net.Income ~ pdq(4, 1, 0))    14.1    0.0787
```

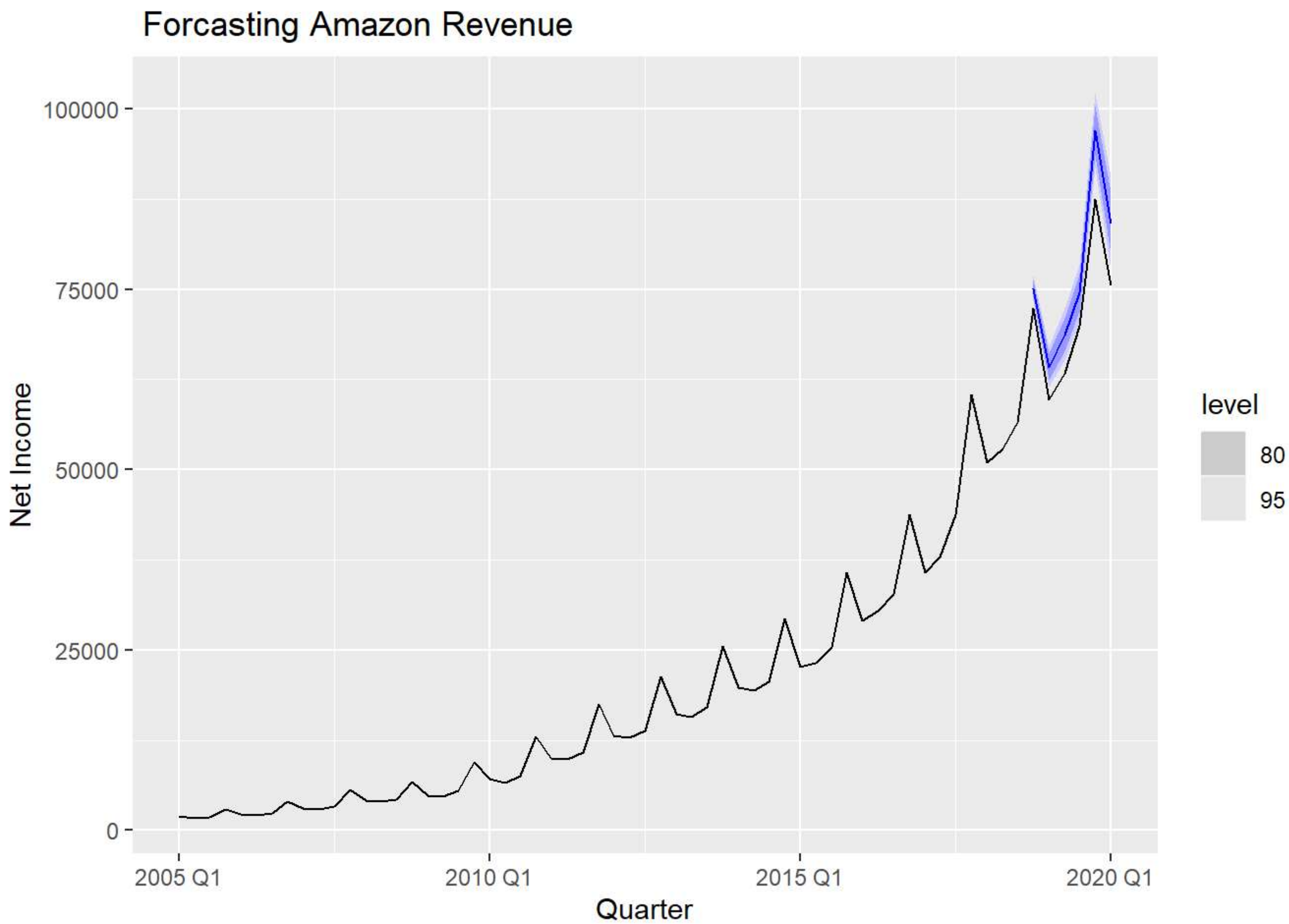
P-value for white noise test for arima410 model is larger than the level of significance 0.05, whereas P-value for white noise test for auto model is smaller than the level of significance 0.05. Therefore, residuals in arima410 model are white noise whereas the residuals in auto are not white noise, which shows that arima410 might be the better model.

Forecasting for Revenue using best ARIMA model - Auto Model

```
# Forecasting for other 6 quarters
fit_auto_fc <- fit_auto_rev %>% forecast(h = 6)

#Checking accuracy measure
fit_auto_fc %>% accuracy(amazon) %>% select(.model, RMSE:MAPE)
```

```
## # A tibble: 1 x 5
##   .model          RMSE   MAE   MPE  MAPE
##   <chr>          <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA(Revenue) 6385. 5904. -8.12  8.12
```

```
# Viewing forecasting plot
fit_auto_fc %>%  autoplot(amazon) +
  labs(title = " Forcasting Amazon Revenue",
       y = "Net Income") +
  guides(colour = guide_legend(title = "Forecast"))
```



Forecasting for Net Income using best ARIMA model - ARIMA(4,1,0)

```
# Forecasting for other 6 quarters
fit_arima410_fc <- fit_arima410 %>% forecast(h = 6)

#Checking accuracy measure
fit_arima410_fc %>% accuracy(amazon) %>% select(.model, RMSE:MAPE)
```

```
## # A tibble: 1 x 5
##   .model                            RMSE   MAE   MPE  MAPE
##   <chr>                            <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA(Net.Income ~ pdq(4, 1, 0))  718.  645. -19.7  24.3
```

```
# Viewing forecasting plot
fit_arima410_fc %>%  autoplot(amazon) +
  labs(title = " Forcasting Amazon Net Income",
       y = "Net Income") +
  guides(colour = guide_legend(title = "Forecast"))
```