Homework 4. Time series

The submitted files must include pdf-s with your answers along with all R scripts. For example:

- Student A submitted:
  - Homework4.pdf - final report containing all answers
  - Homework4.Rmd - R-markdown files with student solutions

No pdf report - no grade. If you experience difficulties with knitting, combine your answers in Word and any other editor and produce pdf-file for grading.

No R scripts - 50 % reduction in grade if relative code present in pdf- report, 100% reduction if no such code present.

Reports longer than 25 pages are not going to be graded.

Question1

1. The plastics data set (see plastics.csv) consists of the monthly sales (in thousands) of product A for a plastics manufacturer for five years. (Total 32 points)

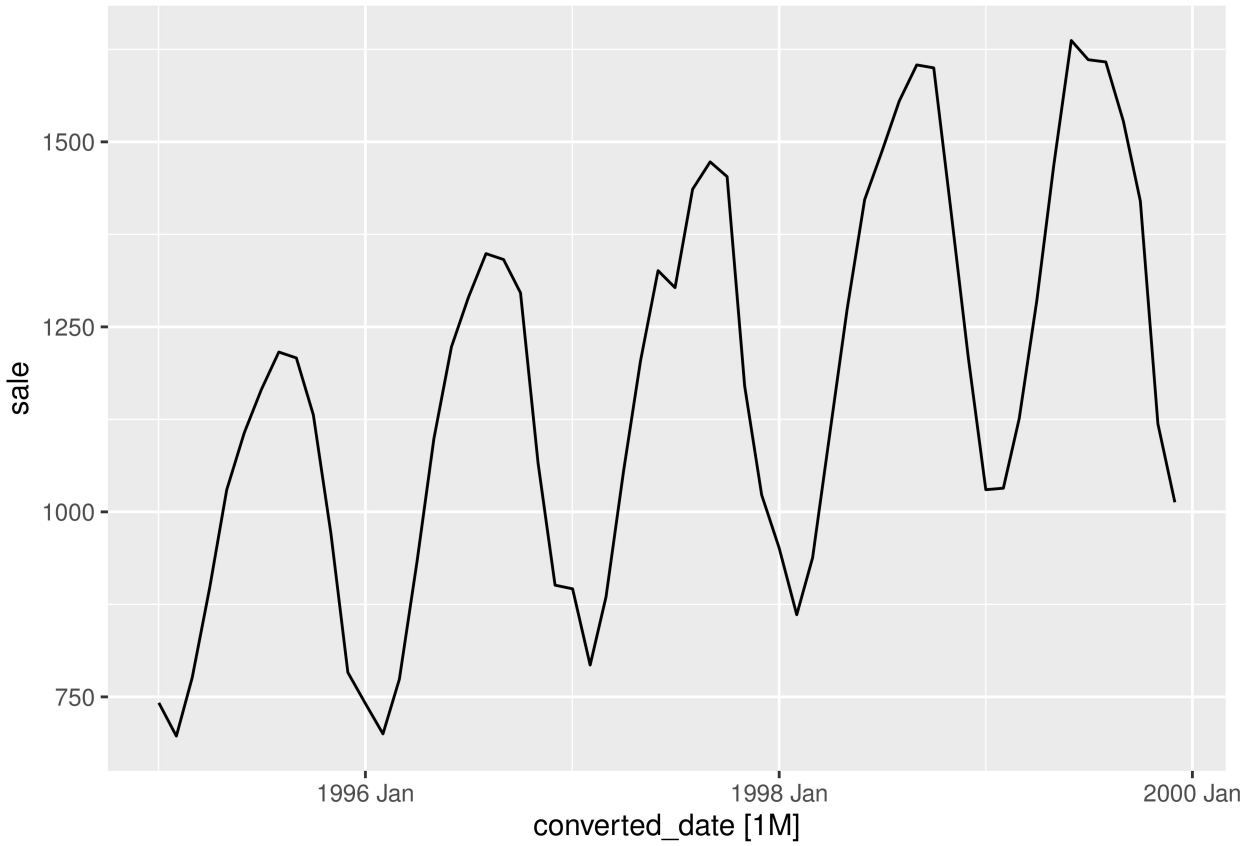
1.1 Read csv file and convert to tsible with proper index (2 points)

```
data <- readr::read_csv("plastics.csv",show_col_types = FALSE)

# Convert to tsibble with index at date
data %>%
  mutate(converted_date=yearmonth(date)) %>%
  tsibble(index=converted_date) ->
  data
head(data)
```

```
## # A tsibble: 6 x 3 [1M]
##   date      sale converted_date
##   <chr>    <dbl>          <mth>
## 1 1995 Jan   742       1995 Jan
## 2 1995 Feb   697       1995 Feb
## 3 1995 Mar   776       1995 Mar
## 4 1995 Apr   898       1995 Apr
## 5 1995 May  1030       1995 May
## 6 1995 Jun  1107       1995 Jun
```

1.2 Plot the time series of sales of product A. Can you identify seasonal fluctuations and/or a trend-cycle? (2 points)

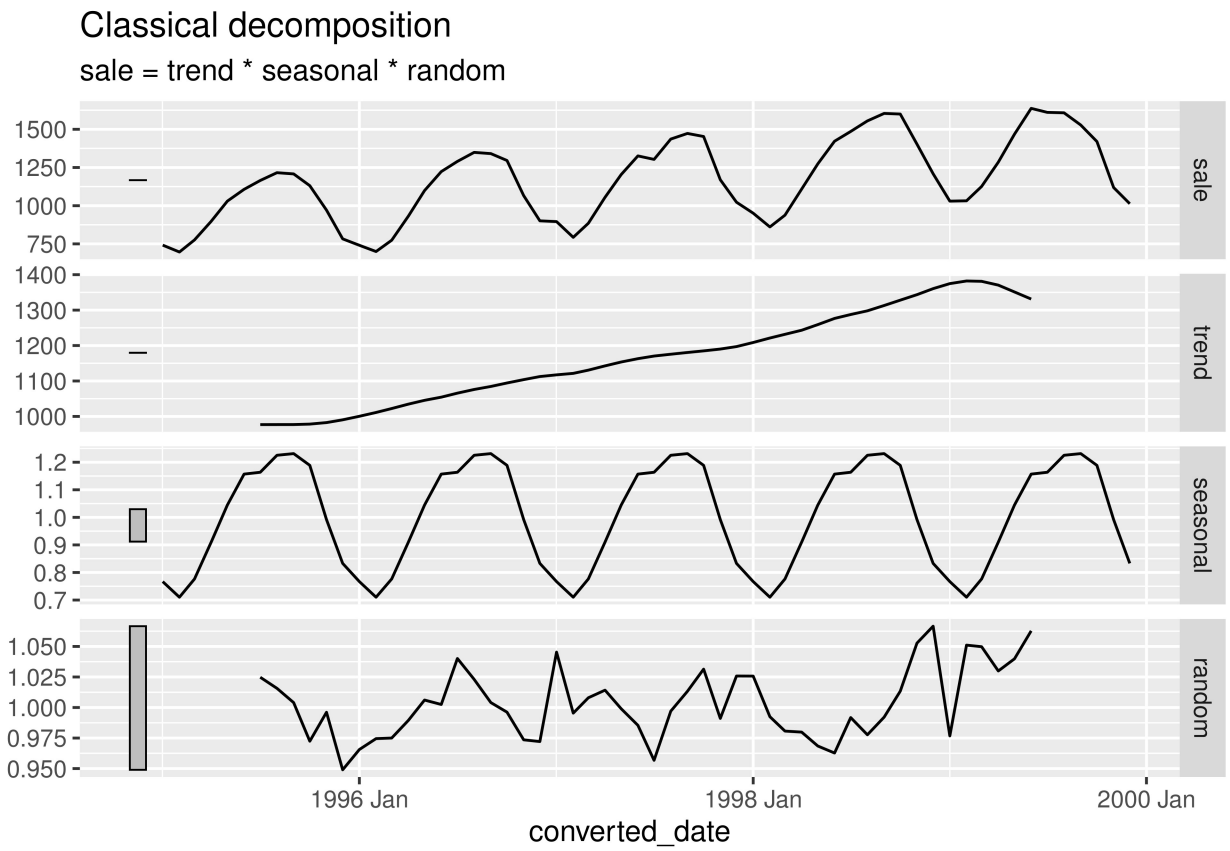
```
data %>% autoplot(sale)
```



Yes, we can see an overall increase in trend. And in the trend cycle we can observe that the trend increases over time.

1.3) Use a classical multiplicative decomposition to calculate the trend-cycle and seasonal components. Plot these components. (4 points)

```
decompose_data <- data %>%
  model(classical_decomposition(sale, type = "mult"))
decompose_data %>% components() %>% autoplot()
```

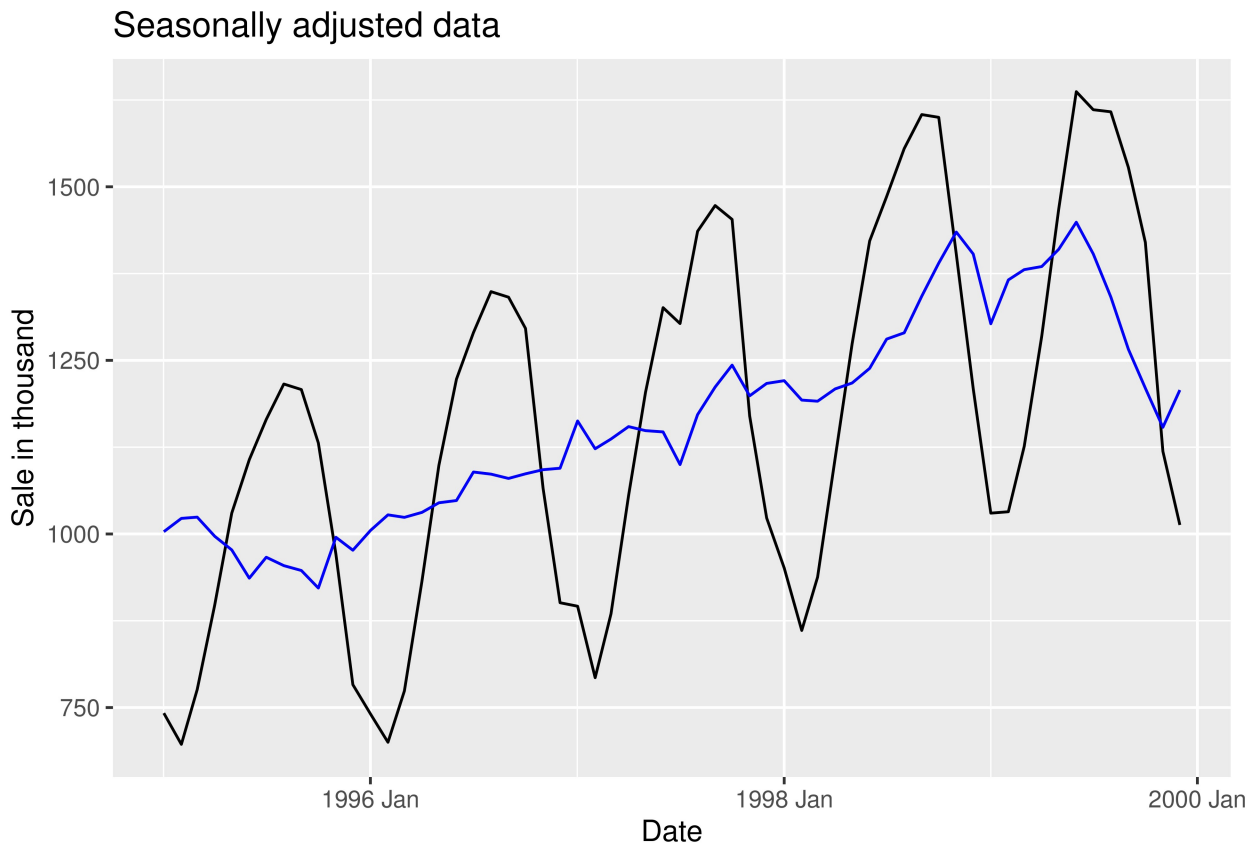


1.4 Do the results support the graphical interpretation from part a? (2 points)

Yes, the results support the graphical interpretation. The trend plot shows and increasing trend with time.We can also see the seasonal increase in trend probably in summer, depending on the location being in northern hemisphere.

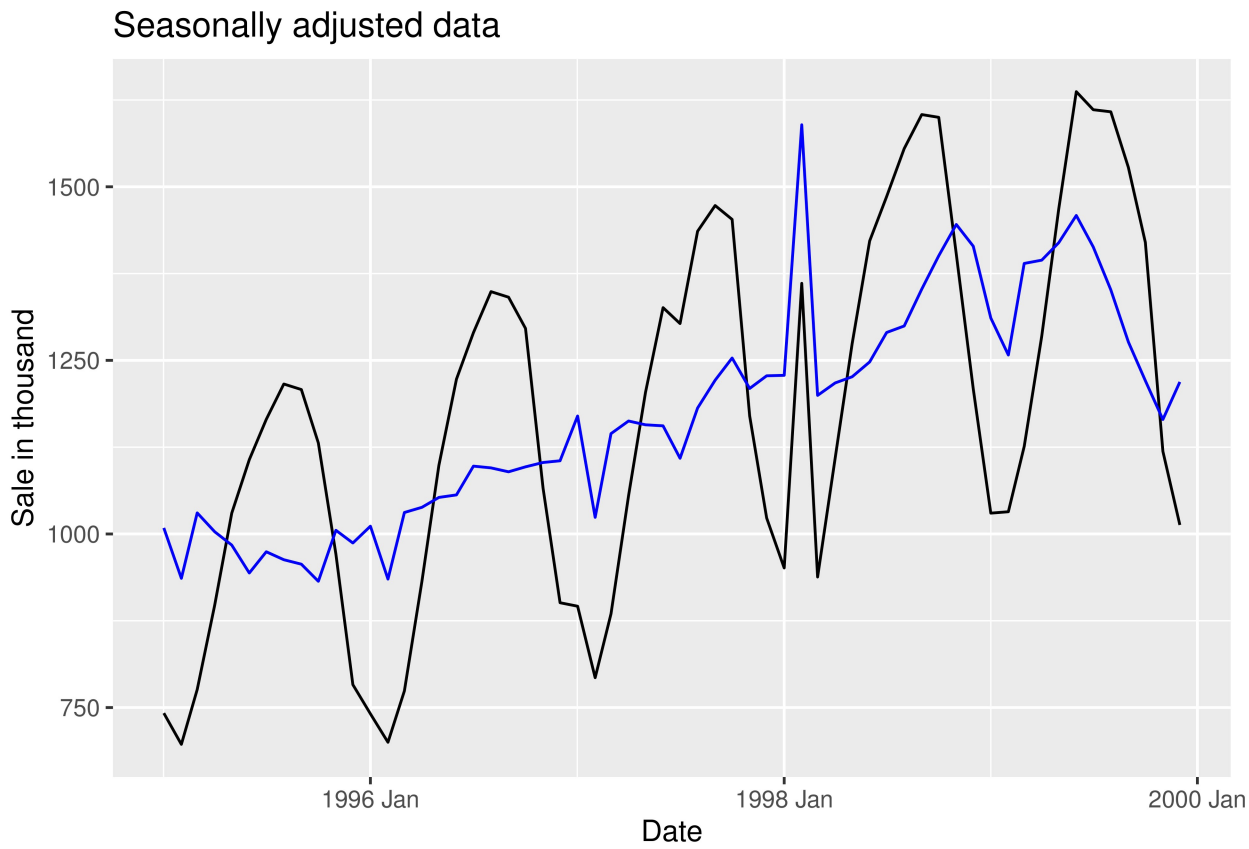
1.5 Compute and plot the seasonally adjusted data. (2 points)

```
dcmp = data%>%
  model(STL(sale))

data%>%autoplot(sale)+
  autolayer(components(dcmp), season_adjust, color= "blue")+
labs(x="Date",y = "Sale in thousand", title="Seasonally adjusted data")
```



1.6 Change one observation to be an outlier (e.g., add 500 to one observation), and recompute the seasonally adjusted data. What is the effect of the outlier? (2 points) tip: use autoplot to plot original and add outlier plot with autolayer

```
set.seed(200)
index = sample(1:60,1)
data_o <- data
data_o$sale[index] <- data_o$sale[index] + 500
dcmp_o = data_o%>%
  model(STL(sale))


data_o%>%autoplot(sale)+
  autolayer(components(dcmp_o), season_adjust, color= "blue")+
labs(x="Date",y = "Sale in thousand", title="Seasonally adjusted data")
```



The outlier gives a peak in seasonality after 1998.

1.7 Does it make any difference if the outlier is near the end rather than in the middle of the time series? (2 points)

The outlier in the oeak would affect the trend of the data rather than seasonality. Although placing the outlier anywhere will affect the time series.

1.8 Let's do some accuracy estimation. Split the data into training and testing. Let all points up to the end of 1998 (including) are training set. (2 points)

```
df_whole <- data
df_train <- df_whole %>% filter(converted_date < yearmonth("1999 Jan"))
```

1.9 Using training set create a fit for mean, naive, seasonal naive and drift methods. Forecast next year (in training set). Plot forecasts and actual data. Which model performs the best. (4 points)

```
fit <- df_train %>%
  model(
    Mean = MEAN(sale),
    Naive = NAIVE(sale),
    Seasonal_Naive = SNAIVE(sale),
    Drift = RW(sale ~ drift())
  )
accuracy(fit)
```

```
## # A tibble: 4 x 10
```

```
##   .model         .type           ME  RMSE   MAE    MPE  MAPE  MASE RMSSE  ACF1
##   <chr>          <chr>        <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Mean           Training  0         255.  217. -5.64  21.0  2.11   2.21 0.867
## 2 Naive          Training  9.94e+ 0  120.  101.  0.410  9.50 0.987  1.04 0.637
## 3 Seasonal_Naive Training  1.02e+ 2  116.  103.  8.52   8.54 1      1    0.602
## 4 Drift          Training -4.35e-14  120.  101. -0.517  9.49 0.981  1.04 0.637
```

```
#report(fit)
report(fit[1])
```

```
## Series: sale
## Model: MEAN
##
## Mean: 1122.1875
## sigma^2: 66318.879
```

```
report(fit[2])
```

```
## Series: sale
## Model: NAIVE
##
## sigma^2: 14655.8437
```

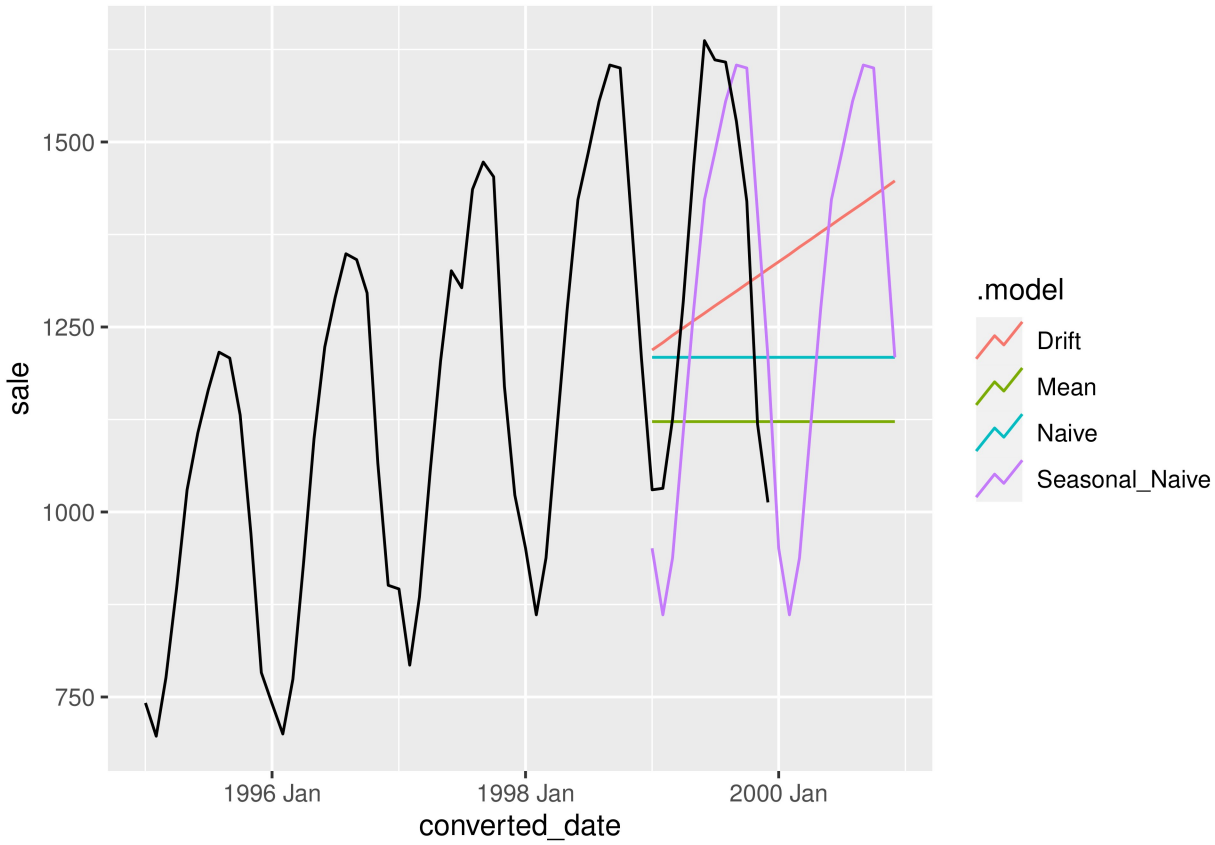
```
report(fit[3])
```

```
## Series: sale
## Model: SNAIVE
##
## sigma^2: 2937.5111
```

```
report(fit[4])
```

```
## Series: sale
## Model: RW w/ drift
##
## Drift: 9.9362 (se: 17.6586)
## sigma^2: 14655.8437
```

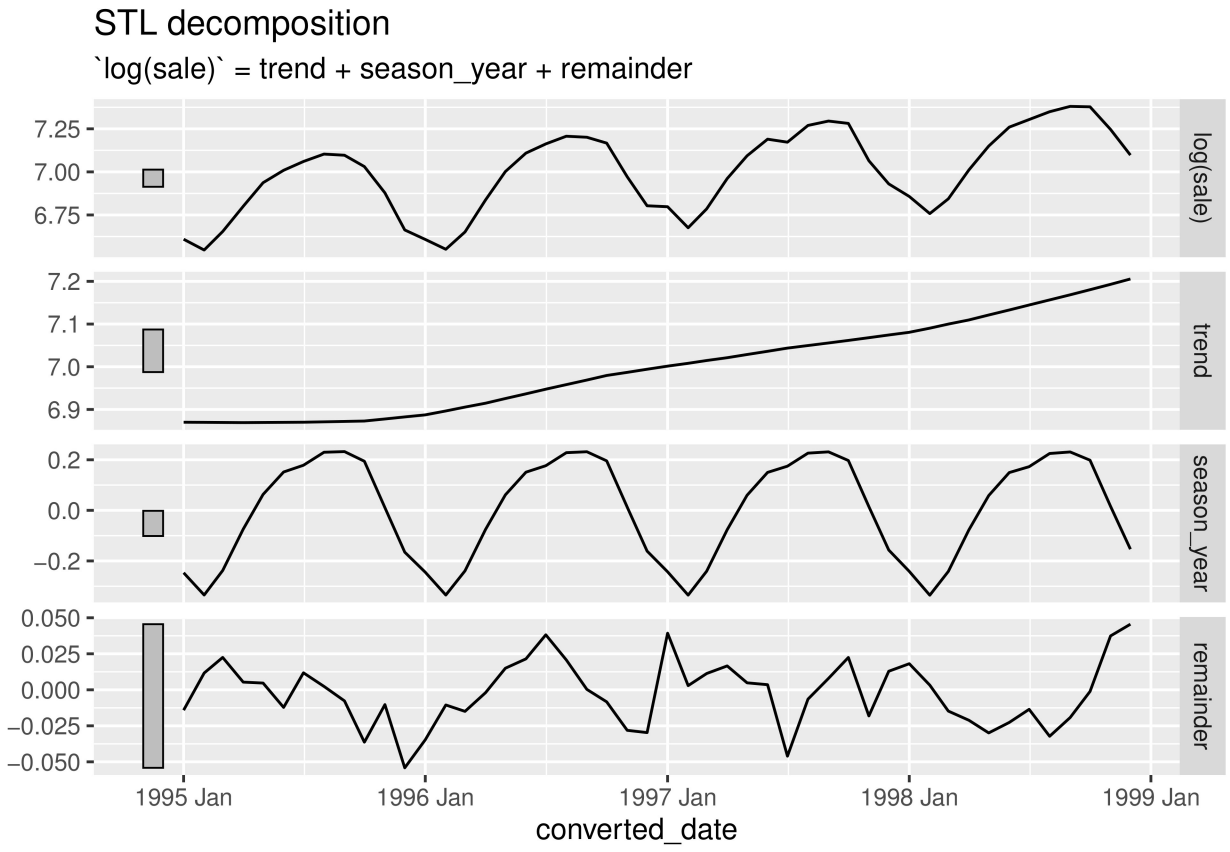
```
# forecast next year (in training set)
fc <- fit %>% forecast(h = 24)
# plot forecasts and actual data
fc %>% autoplot(df_whole,level = NULL)
```



The best model is the Seasonal_Naive as it has the lowest RMSE score.

1.10 Repeat 1.9 for appropriate EST. Report the model. Check residuals. Plot forecasts and actual data. (4 points)

```
df_train %>% model(STL(log(sale))) %>% components() %>% autoplot()
```




```
fit <- df_train %>%
  model(
    ets_auto = ETS(log(sale)),
    ets = ETS(log(sale) ~ error("A") + trend("A") + season("A"))
  )
accuracy(fit)
```

```
## # 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 ets_auto Training  2.36  26.8  22.0 0.127  1.95 0.215 0.232 0.100
## 2 ets      Training  2.36  26.8  22.0 0.127  1.95 0.215 0.232 0.100
```

```
report(fit[1])
```

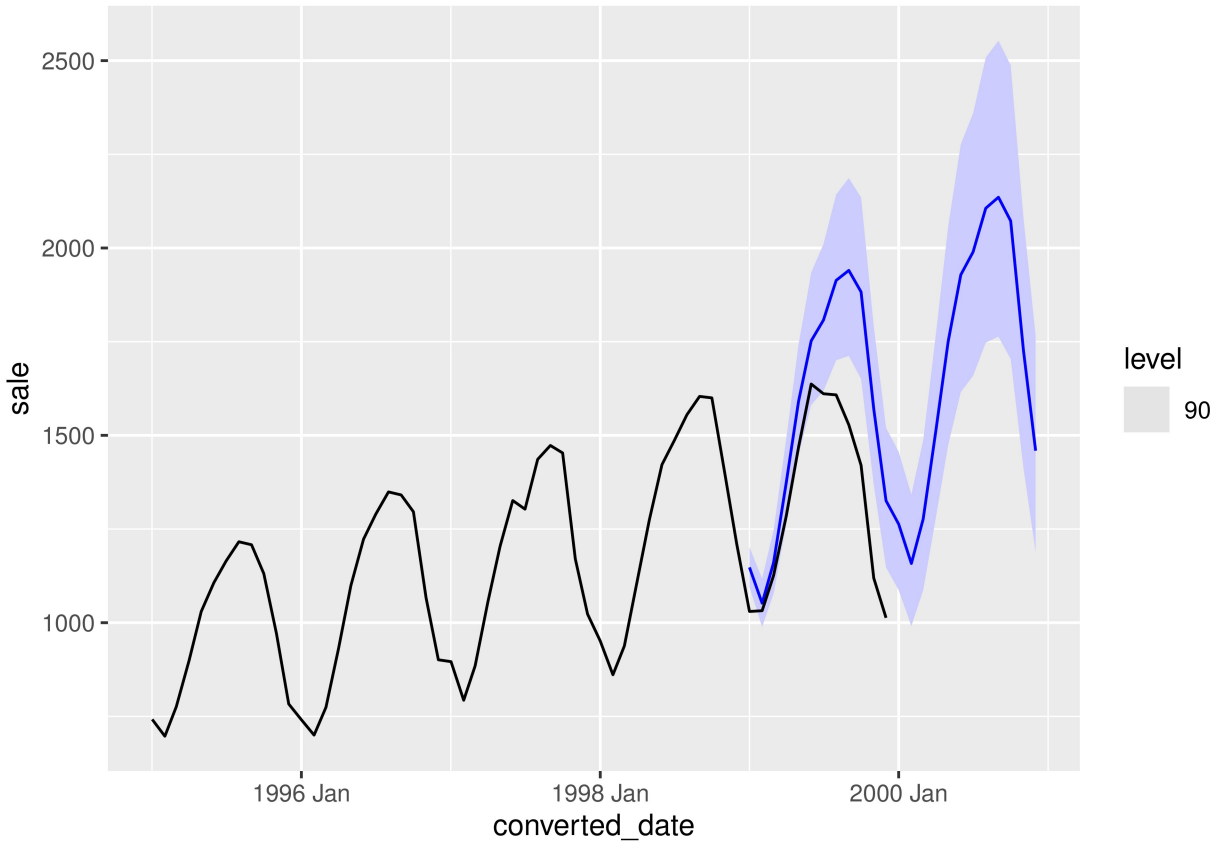
```
## Series: sale
## Model: ETS(A,A,A)
## Transformation: log(sale)
##   Smoothing parameters:
##     alpha = 0.8619105
##     beta  = 0.0001130351
##     gamma = 0.0001009812
##
##   Initial states:
```

```
##     l[0]        b[0]       s[0]       s[-1]     s[-2]    s[-3]     s[-4]
##  6.83579 0.007685797 -0.1745398 0.002879683 0.1923146 0.230409 0.2245478
##     s[-5]     s[-6]      s[-7]       s[-8]      s[-9]     s[-10]     s[-11]
##  0.175578 0.1524543 0.06326799 -0.07459685 -0.2357618 -0.3257154 -0.2308375
##
##   sigma^2:  8e-04
##
##       AIC      AICc       BIC
## -142.1542 -121.7542 -110.3438
```

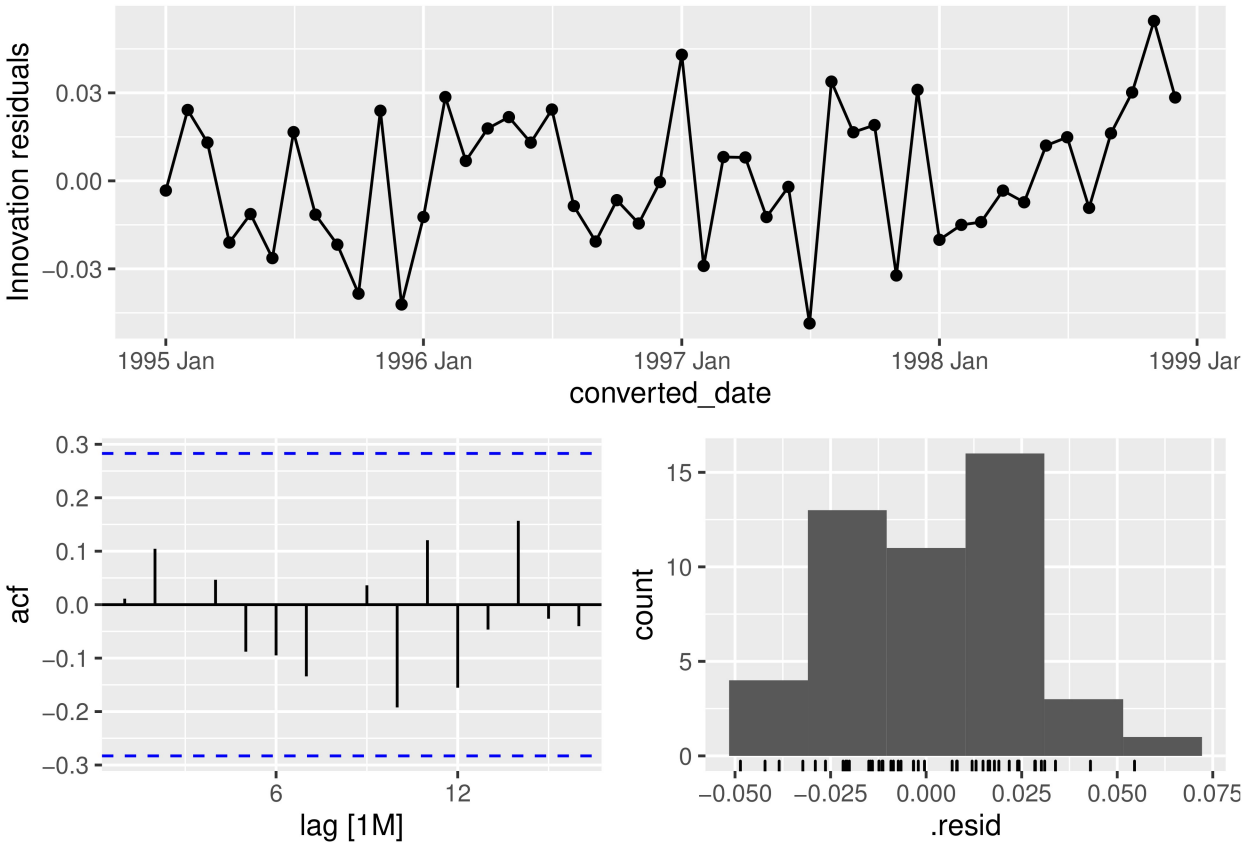
```
report(fit[2])
```

```
## Series: sale
## Model: ETS(A,A,A)
## Transformation: log(sale)
##   Smoothing parameters:
##     alpha = 0.8619105
##     beta  = 0.0001130351
##     gamma = 0.0001009812
##
##   Initial states:
##     l[0]        b[0]       s[0]       s[-1]     s[-2]    s[-3]     s[-4]
##  6.83579 0.007685797 -0.1745398 0.002879683 0.1923146 0.230409 0.2245478
##     s[-5]     s[-6]      s[-7]       s[-8]      s[-9]     s[-10]     s[-11]
##  0.175578 0.1524543 0.06326799 -0.07459685 -0.2357618 -0.3257154 -0.2308375
##
##   sigma^2:  8e-04
##
##       AIC      AICc       BIC
## -142.1542 -121.7542 -110.3438
```

```
#fit <- fit %>% select(ets)

# forecast next year (in training set)
fc <- fit[2] %>% forecast(h = 24)
# plot forecasts and actual data
fc %>% autoplot(df_whole,level = 90)
```

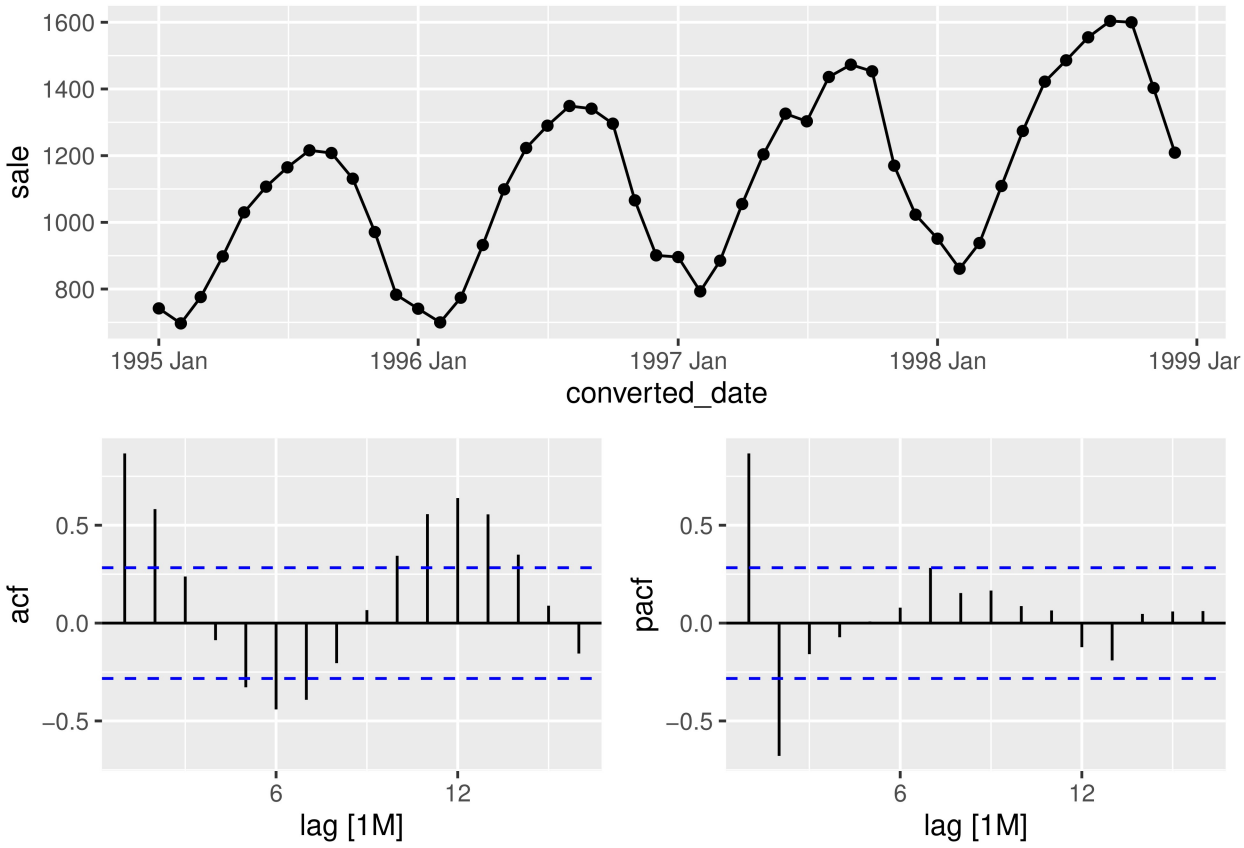



```
#Check residuals
gg_tsresiduals(fit[2])
```



1.11 Repeat 1.9 for appropriate ARIMA. Report the model. Check residuals. Plot forecasts and actual data. (4 points)

```
gg_tsdisplay(df_train, sale, plot_type='partial')
```

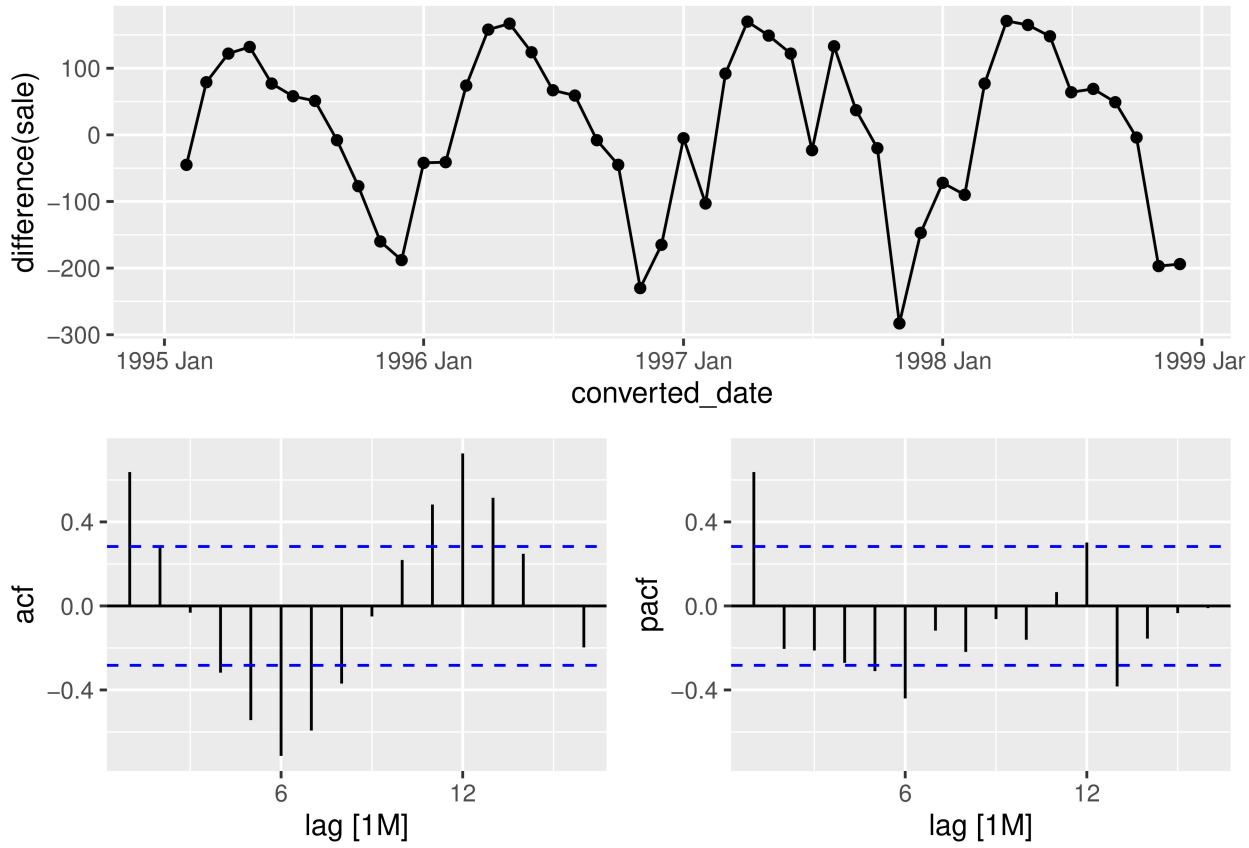


ACF: non seasonal: MA(1) seasonal: MA(12) (0,1,1),(0,1,1)base12 PACF:non seasonal: MA(2) seasonal: MA(12) (2,1,0),(0,1,1)base12

```
#library(tseries)
#adf.test(df_train$sale)#data stationary as p value less than .05
```

Data is not staionary

```
gg_tsdisplay(df_train, difference(sale), plot_type='partial')
```



We select first order difference as it has more noise From PACF: MA(6) so (13,1,0) & From ACF: AR(12) so (0,1,12)

```
df_train %>%
features(sale, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1     0.521      0.0369
```

```
df_train %>% features(sale, unitroot_nsdiffs)
```

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

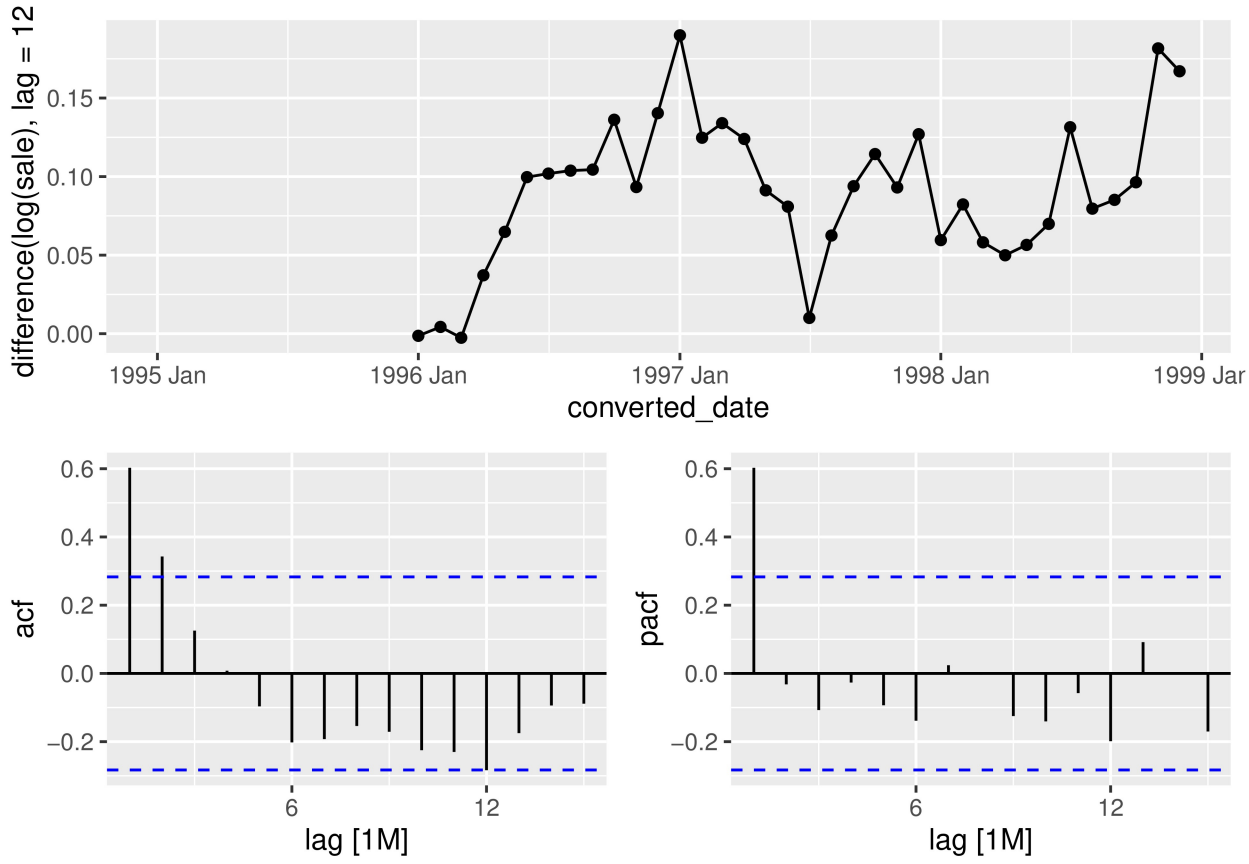
```
df_train %>% features(log(sale), unitroot_nsdiffs)
```

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

```
df_train %>% features( difference(log(sale), lag=12), unitroot_ndiffs)
```

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

```
gg_tsdisplay(df_train, difference(log(sale), lag=12), plot_type='partial')
```




```
fit <- (df_train) %>%
  model(
    arima_auto = ARIMA(log(sale), stepwise = FALSE, approx = FALSE),
    arima011011 = ARIMA(log(sale)~0+pdq(0,1,1)+PDQ(0,1,1)),
    arima2210011 = ARIMA(log(sale)~0+pdq(2,1,0)+PDQ(0,1,1))

  )
accuracy(fit)
```

```
## # A tibble: 3 x 10
##   .model       .type       ME  RMSE   MAE   MPE  MAPE  MASE RMSSE     ACF1
##   <chr>        <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1 arima_auto   Training  2.91  30.5  22.4 0.102  1.99 0.218 0.264 -0.0247
## 2 arima011011  Training  7.25  32.8  23.0 0.472  1.96 0.225 0.284  0.00618
## 3 arima2210011 Training  6.94  32.3  22.7 0.443  1.94 0.222 0.279  0.0266
```

```
report(fit[1])
```

```
## Series: sale
## Model: ARIMA(1,0,0)(0,1,1)[12] w/ drift
## Transformation: log(sale)
##
## Coefficients:
##          ar1     sma1  constant
##       0.7545  -0.5857    0.0235
## s.e.  0.1310   0.3406    0.0033
##
## sigma^2 estimated as 0.0009949:  log likelihood=72.39
## AIC=-136.77   AICc=-135.48   BIC=-130.44
```

```
report(fit[2])
```

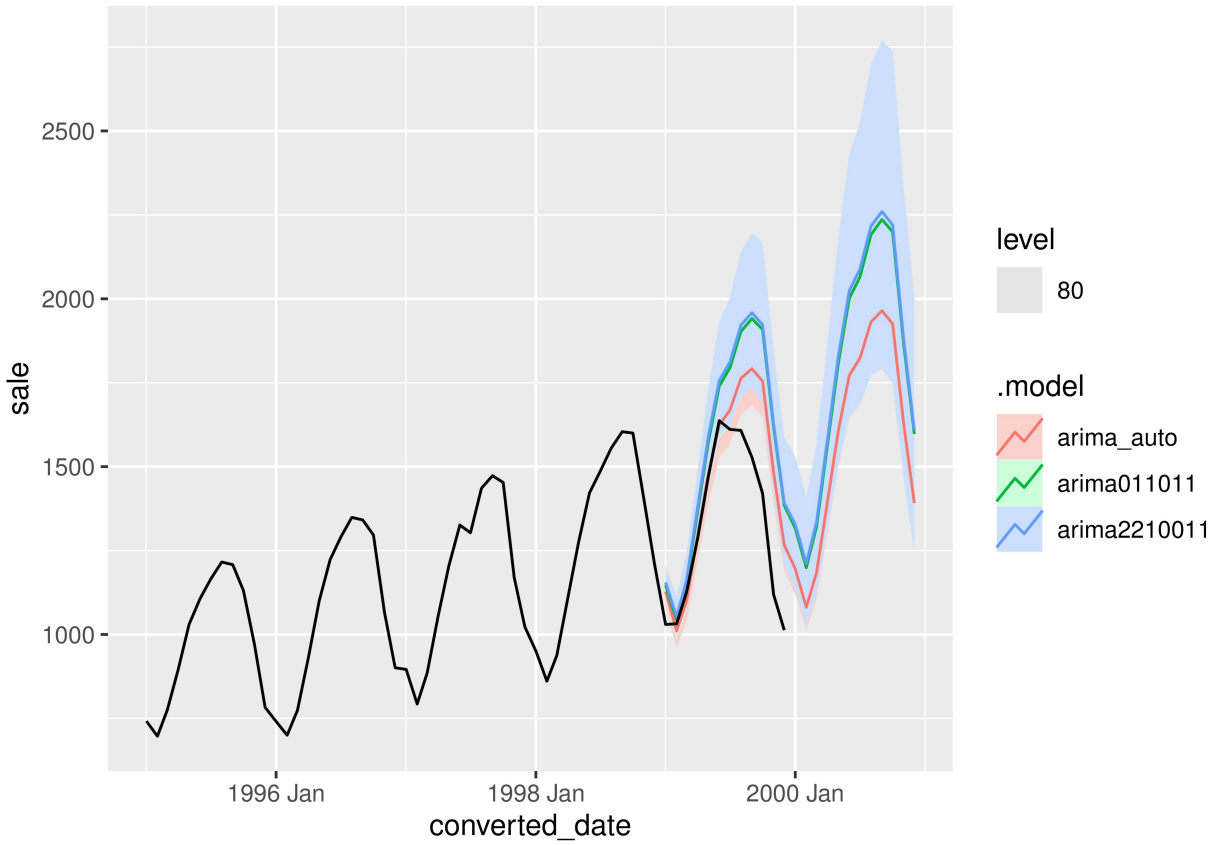
```
## Series: sale
## Model: ARIMA(0,1,1)(0,1,1)[12]
## Transformation: log(sale)
##
## Coefficients:
##           ma1     sma1
##       -0.1525  -0.5381
## s.e.   0.1653   0.3115
##
## sigma^2 estimated as 0.001079:  log likelihood=69.21
## AIC=-132.42   AICc=-131.65   BIC=-127.75
```

```
report(fit[3])
```

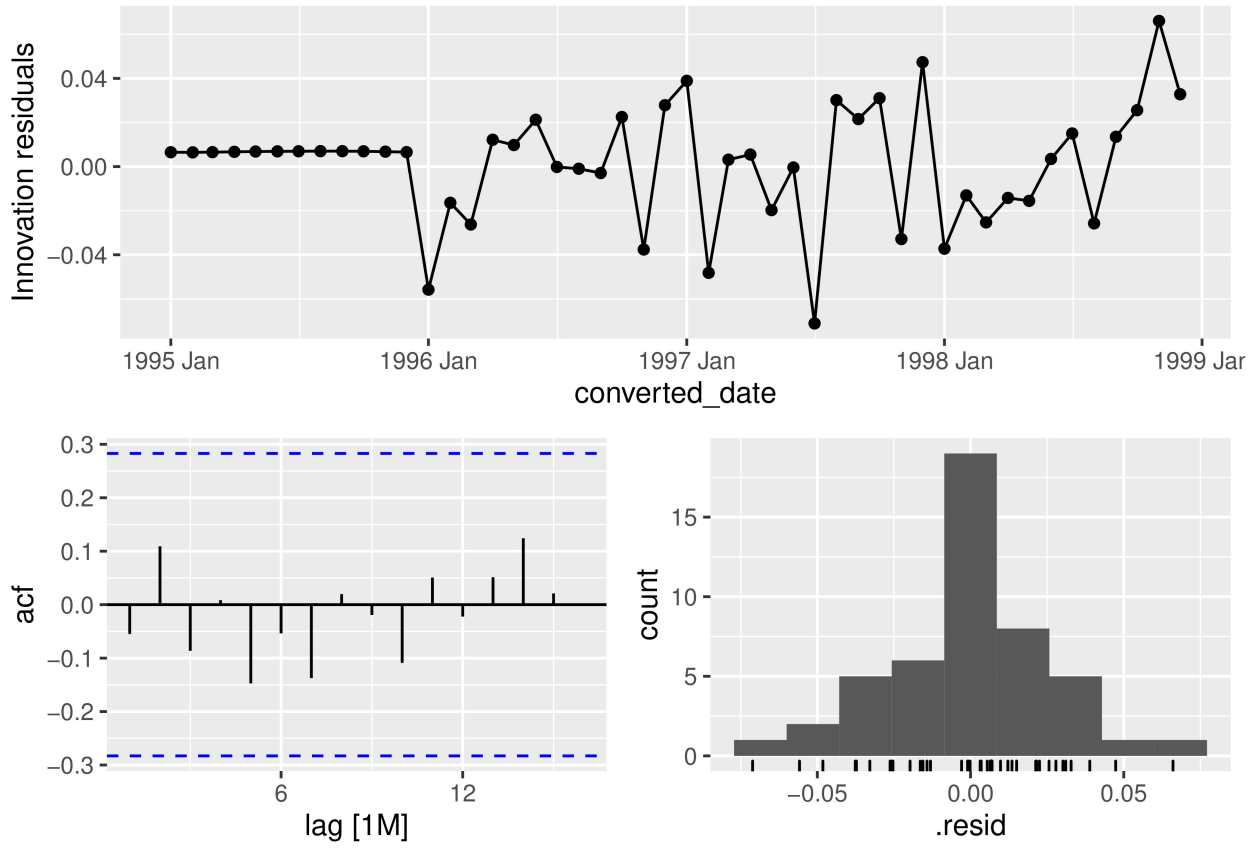
```
## Series: sale
## Model: ARIMA(2,1,0)(0,1,1)[12]
## Transformation: log(sale)
##
## Coefficients:
##           ar1     ar2     sma1
##       -0.1515  0.0764  -0.5733
## s.e.   0.1795  0.1853   0.3525
##
## sigma^2 estimated as 0.001084:  log likelihood=69.36
## AIC=-130.73   AICc=-129.4   BIC=-124.51
```

```
fc <- fit %>% forecast(h = "2 year")
```

```
fc %>% autoplot(df_whole,level = 80)
```




```
gg_tsresiduals(fit[1])
```



1.12 Which model has best performance? (2 points) Auto ARIMA has the best performance as it has the lowest RMSE values, although AIC values of our model arima2210011 is lower. ## Question 2

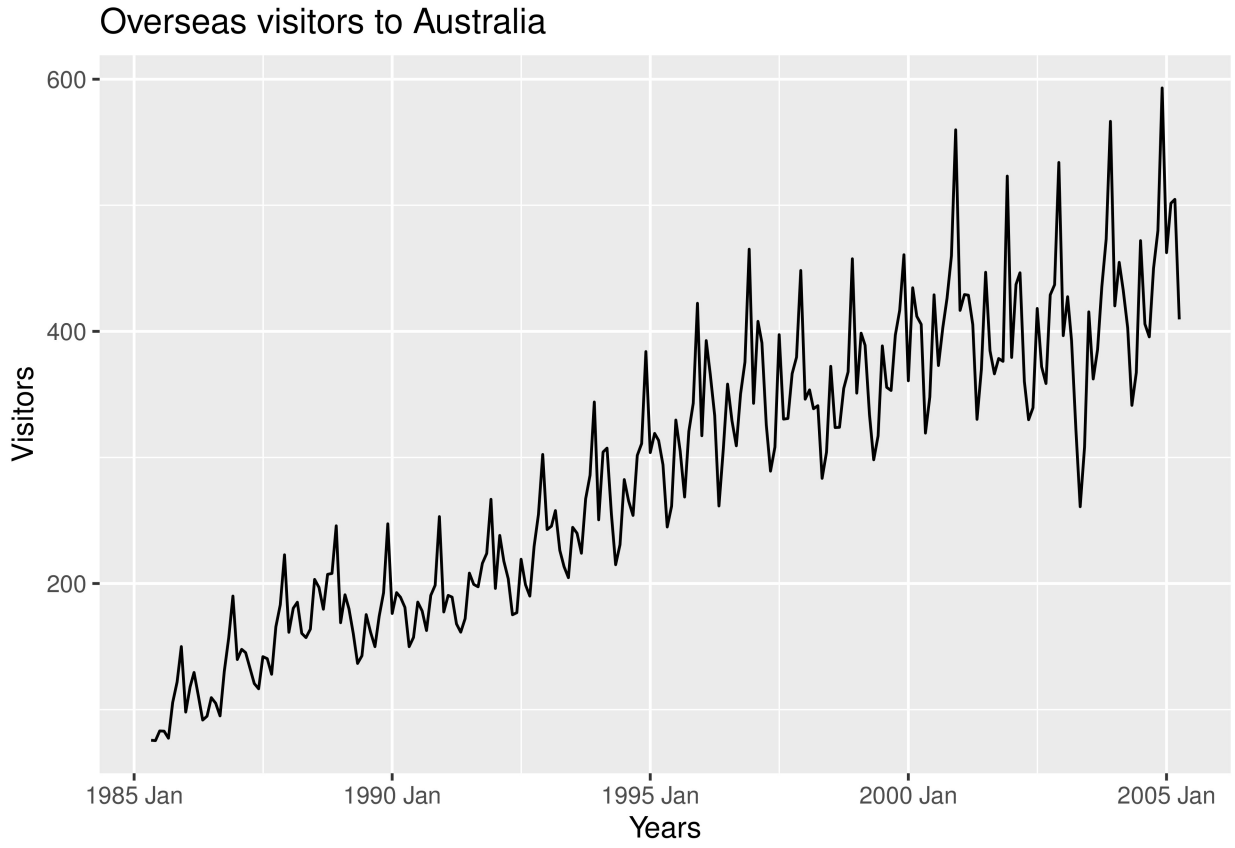
2 For this exercise use data set visitors (visitors.csv), the monthly Australian short-term overseas visitors data (thousands of people per month), May 1985–April 2005. (Total 32 points)

2.1 Make a time plot of your data and describe the main features of the series. (6 points)

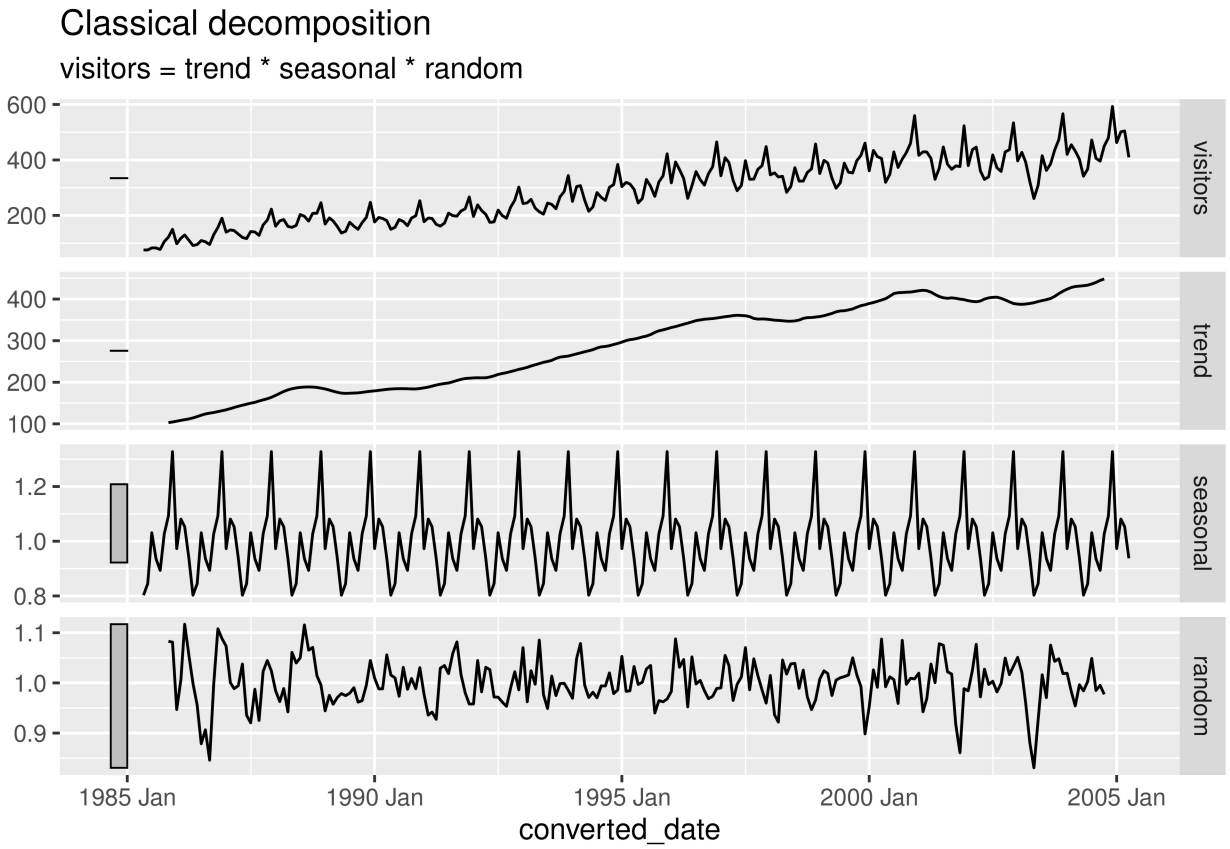
```
visitors_df <- readr::read_csv("visitors.csv",show_col_types = FALSE)

# Convert to tsibble with index at date
visitors_df %>%
  mutate(converted_date=yearmonth(date)) %>%
  tsibble(index=converted_date) ->
  vis_data
head(vis_data)
```

```
## # A tsibble: 6 x 3 [1M]
##   date     visitors converted_date
##   <chr>       <dbl>          <mth>
## 1 1985 May     75.7       1985 May
## 2 1985 Jun     75.4       1985 Jun
## 3 1985 Jul     83.1       1985 Jul
## 4 1985 Aug     82.9       1985 Aug
## 5 1985 Sep     77.3       1985 Sep
## 6 1985 Oct    106.        1985 Oct
```

```
#plot
autoplot(vis_data) +  ggtitle("Overseas visitors to Australia") +xlab("Years")+ ylab("Visitors")
```



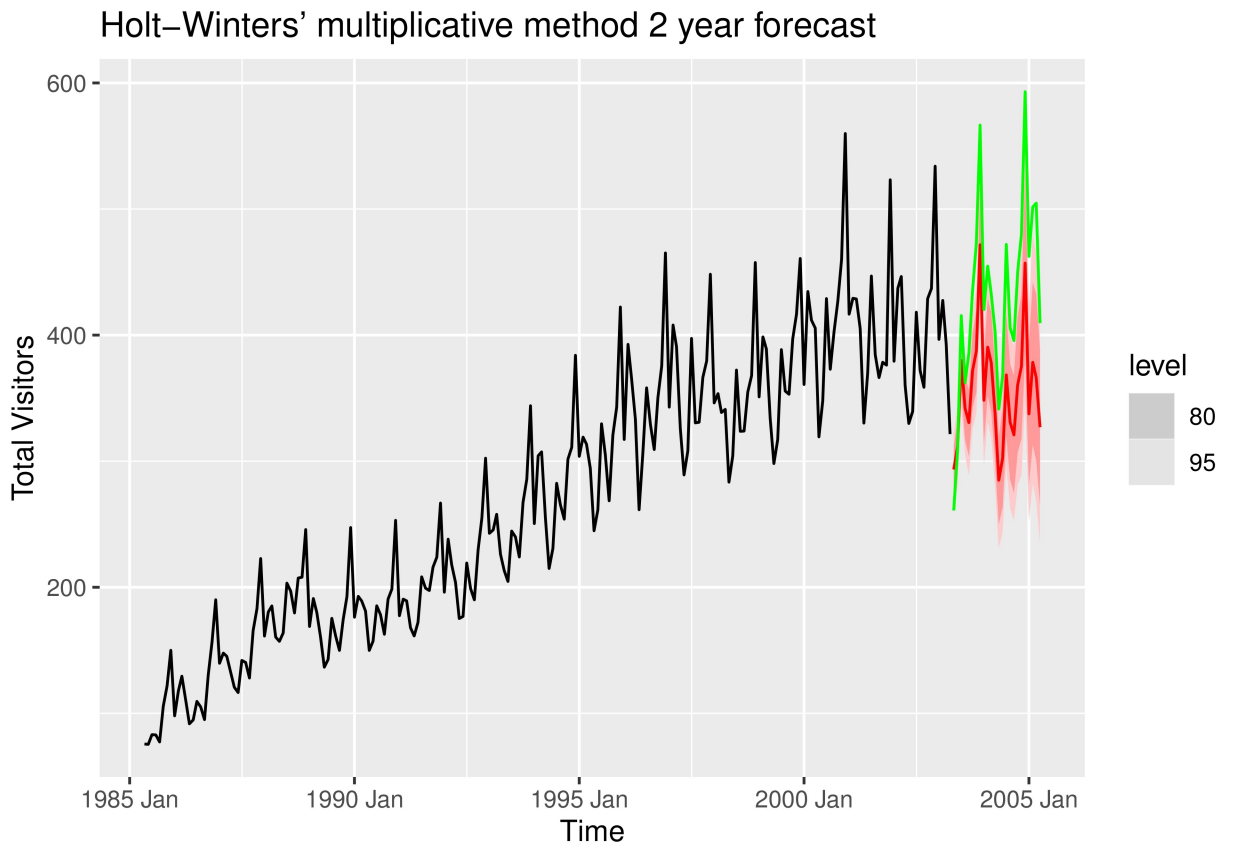

```
decompose_data <- vis_data %>%
  model(classical_decomposition(visitors, type = "mult"))
decompose_data %>% components() %>% autoplot()
```



The number if vistors increase every year and there is a good amount of seasonality to it.

2.2 Split your data into a training set and a test set comprising the last two years of available data. Forecast the test set using Holt-Winters' multiplicative method. (6 points)

```
vis_train <- vis_data %>% filter(converted_date < yearmonth("2003 May"))
vis_test <- vis_data %>% filter(converted_date > yearmonth("2003 Apr"))

#forecasting for next two years
fit <- vis_train %>%
  model(holt = ETS(visitors ~ error("A") + trend("A") + season("M"))) %>%
  forecast(h = 24)

autoplot(vis_train, level = 90, color = "black") +
  autolayer(fit, color = "red", series="Holt-Winters'") +
  autolayer(vis_test, color = "green", series = "Test") +
  labs(title="Holt-Winters' multiplicative method 2 year forecast", x = 'Time', y="Total Visitors")
```

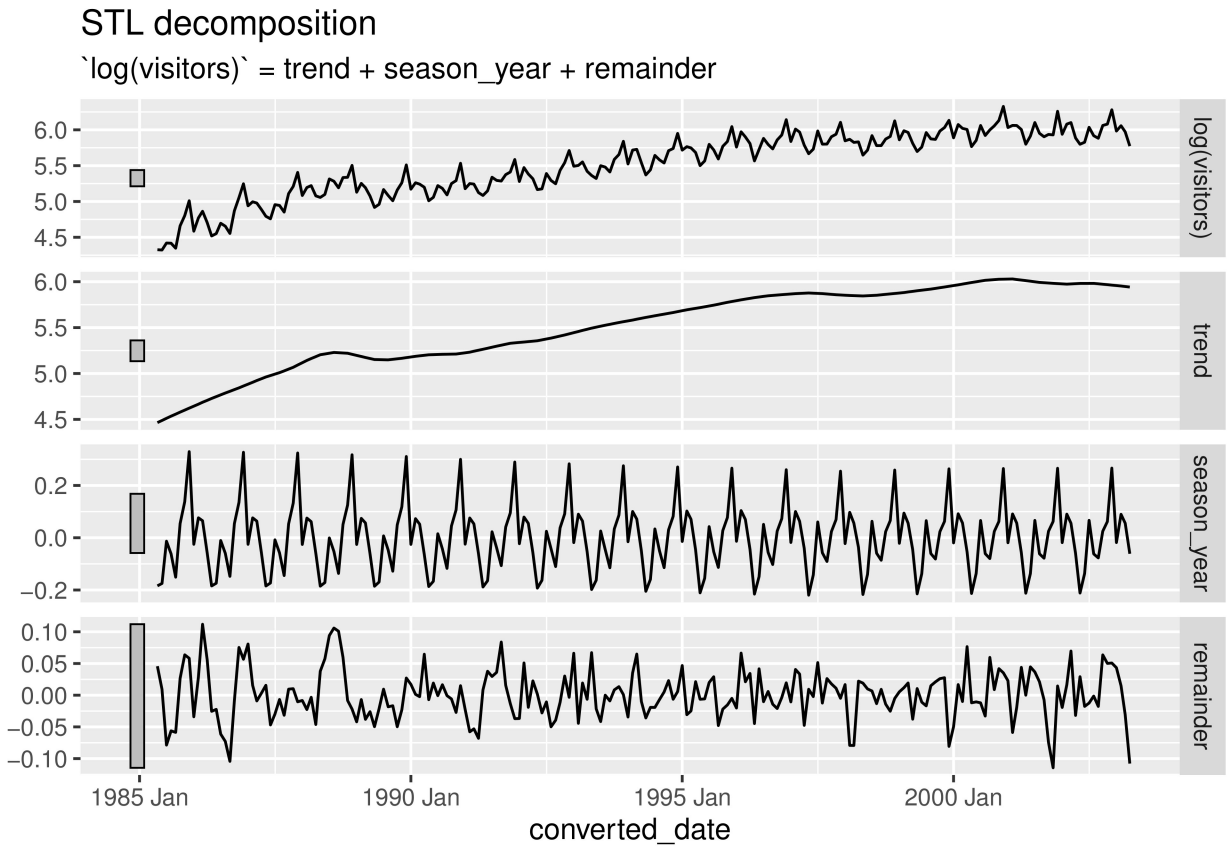


2.3. Why is multiplicative seasonality necessary here? (6 points) The magnitude of seasonal component changes with times, hence multiplicative seasonality.

2.4. Forecast the two-year test set using each of the following methods: (8 points)

I. an ETS model;
II. an additive ETS model applied to a Box-Cox transformed series;
III. a seasonal naïve method;

```
vis_train %>%
  model(STL(log(visitors))) %>%
  components() %>%
  autoplot()
```




```
ets_fit = vis_train %>%
  model(
    ets_auto = ETS(log(visitors)),
    Seasonal_Naive = SNAIVE(visitors)
  )
accuracy(ets_fit)
```

```
## # 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 ets_auto       Training  0.908  14.4  10.6 0.242  4.00 0.405 0.462 -0.0402
## 2 Seasonal_Naive Training 17.3    31.2  26.1 7.19  10.3  1     1       0.633
```

```
report(ets_fit[1])
```

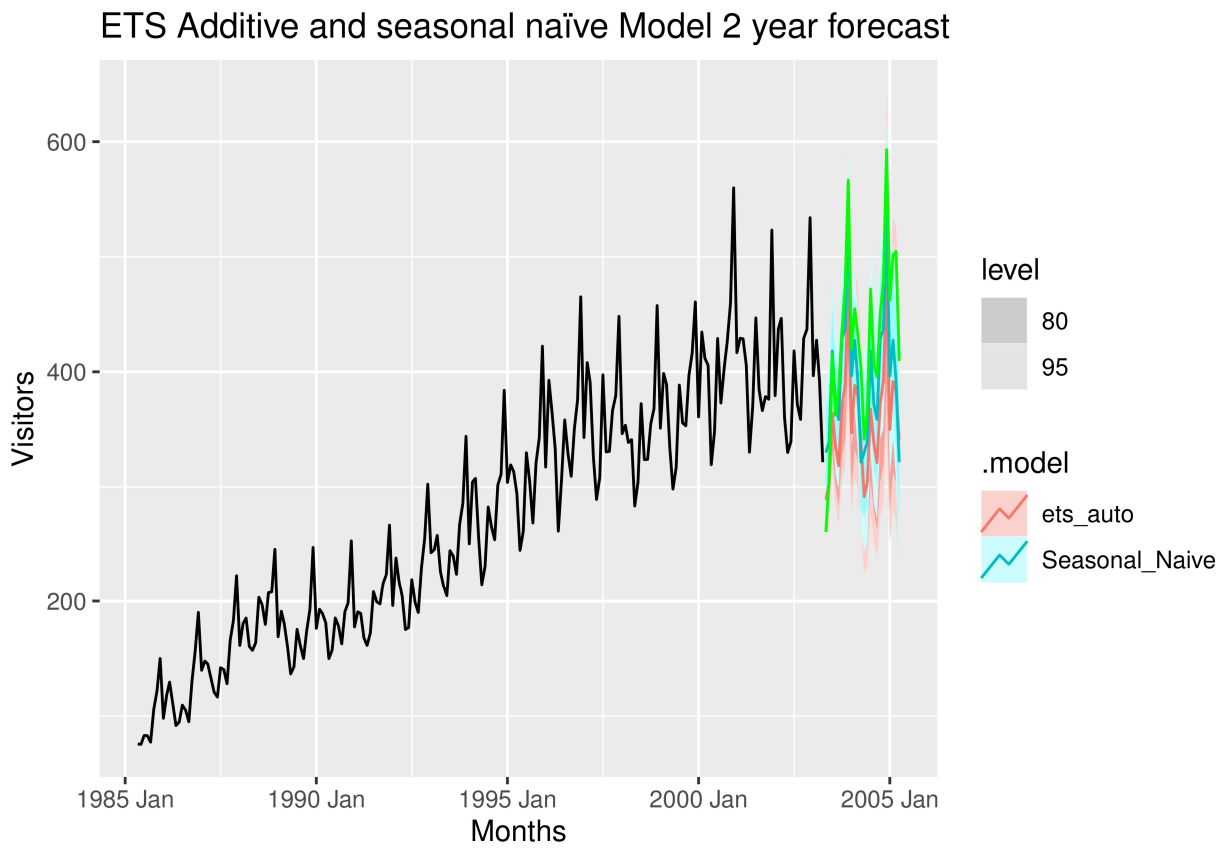
```
## Series: visitors
## Model: ETS(A,Ad,A)
## Transformation: log(visitors)
##   Smoothing parameters:
##     alpha = 0.6293372
##     beta  = 0.0006256646
##     gamma = 0.0001292349
##     phi   = 0.979999
##
```

```
##   Initial states:
##      l[0]       b[0]        s[0]      s[-1]      s[-2]      s[-3]     s[-4]
##  4.480591 0.01757652 -0.05231794 0.06496332 0.08712093 -0.0246398 0.2866596
##       s[-5]      s[-6]      s[-7]       s[-8]      s[-9]     s[-10]     s[-11]
##  0.09212879 0.04009071 -0.1080958 -0.05590228 0.02919788 -0.1576367 -0.2015687
##
##   sigma^2:  0.0029
##
##       AIC      AICc       BIC
## -84.01131 -80.53923 -23.25630
```

```
report(ets_fit[2])
```

```
## Series: visitors
## Model: SNAIVE
##
## sigma^2: 674.9436
```

```
ets_fc = ets_fit %>% forecast(h = "2 years")

autoplot(vis_train, level = 90, color = "black") +
  autolayer(ets_fc, color = "red", series="ETS") +
  autolayer(vis_test, color = "green", series = "Test") +
  ggtitle("ETS Additive and seasonal naïve Model 2 year forecast") +
  xlab("Months") +
  ylab("Visitors")
```



```
#box cox
lambda = vis_train %>% features(visitors, features = guerrero)
lambda
```

```
## # A tibble: 1 x 1
##   lambda_guerrero
##             <dbl>
## 1           0.362
```

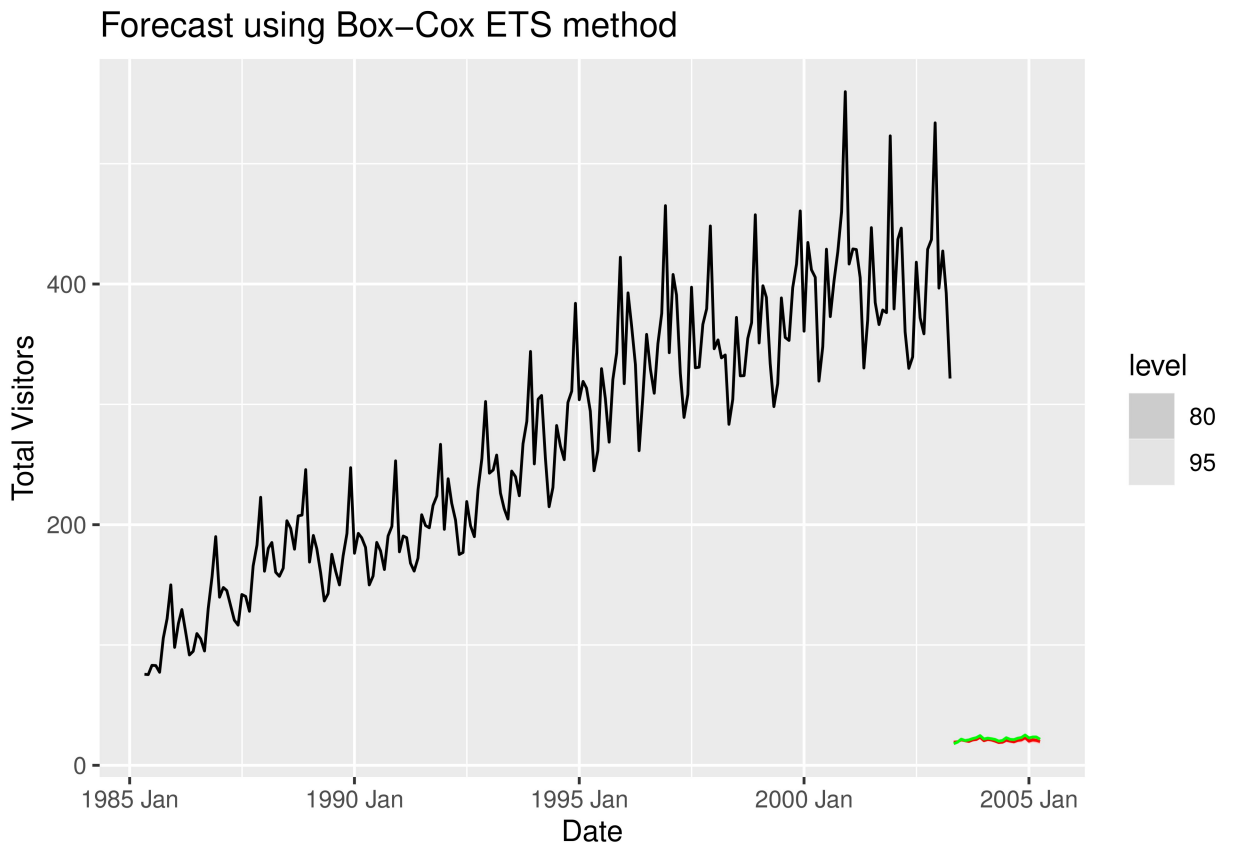
```
df_train_bc = vis_train %>% mutate(visitors = box_cox(visitors, lambda))
df_test_bc = vis_test %>% mutate(visitors = box_cox(visitors, lambda))

bc_fit = df_train_bc %>%
  model(
    Box_cox_ets = ETS(log(visitors) ~ error("A") + trend("A") + season("A")),
  )
accuracy(bc_fit)
```

```
## # A tibble: 1 x 10
##   .model      .type         ME  RMSE   MAE    MPE  MAPE  MASE RMSSE  ACF1
##   <chr>       <chr>      <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Box_cox_ets Training -0.0590 0.418 0.325 -0.386  1.91 0.417 0.452 0.357
```

```
report(bc_fit)
```

```
## Series: visitors
## Model: ETS(A,A,A)
## Transformation: log(visitors)
##   Smoothing parameters:
##     alpha = 0.2746539
##     beta  = 0.01655378
##     gamma = 0.0001006887
##
##   Initial states:
##      l[0]       b[0]        s[0]      s[-1]      s[-2]       s[-3]     s[-4]
##  2.386165 0.01137379 -0.02295132 0.02290911 0.03788298 -0.01057209 0.1199257
##       s[-5]      s[-6]     s[-7]       s[-8]     s[-9]      s[-10]     s[-11]
##  0.03588181 0.01414548 -0.046475 -0.01972517 0.0158953 -0.06509136 -0.0818254
##
##   sigma^2:  7e-04
##
##       AIC      AICc       BIC
## -403.0012 -399.9103 -345.6214
```

```
bc_fc = bc_fit %>% forecast(h = "2 years")

autoplot(vis_train, color = "black") +
  autolayer(bc_fc, color = "red", series ="BoxCox-ETS-forecasts") +
  autolayer(df_test_bc, color = "green", series = "Test") +
  ggtitle("Forecast using Box-Cox ETS method") +
  xlab('Date') +
  ylab("Total Visitors")
```



2.5. Which method gives the best forecasts? Does it pass the residual tests? (6 points)

```
#accuracy ets
# Accuracy and residuals for the ets and snaive model
accuracy(ets_fc, vis_data)
```

```
## # 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 ets_auto       Test   69.8  77.5  72.1 15.4  16.3   2.77  2.49 0.567
## 2 Seasonal_Naive Test   32.9  50.3  42.2  6.64  9.96  1.62  1.61 0.573
```

```
report(ets_fit[1])
```

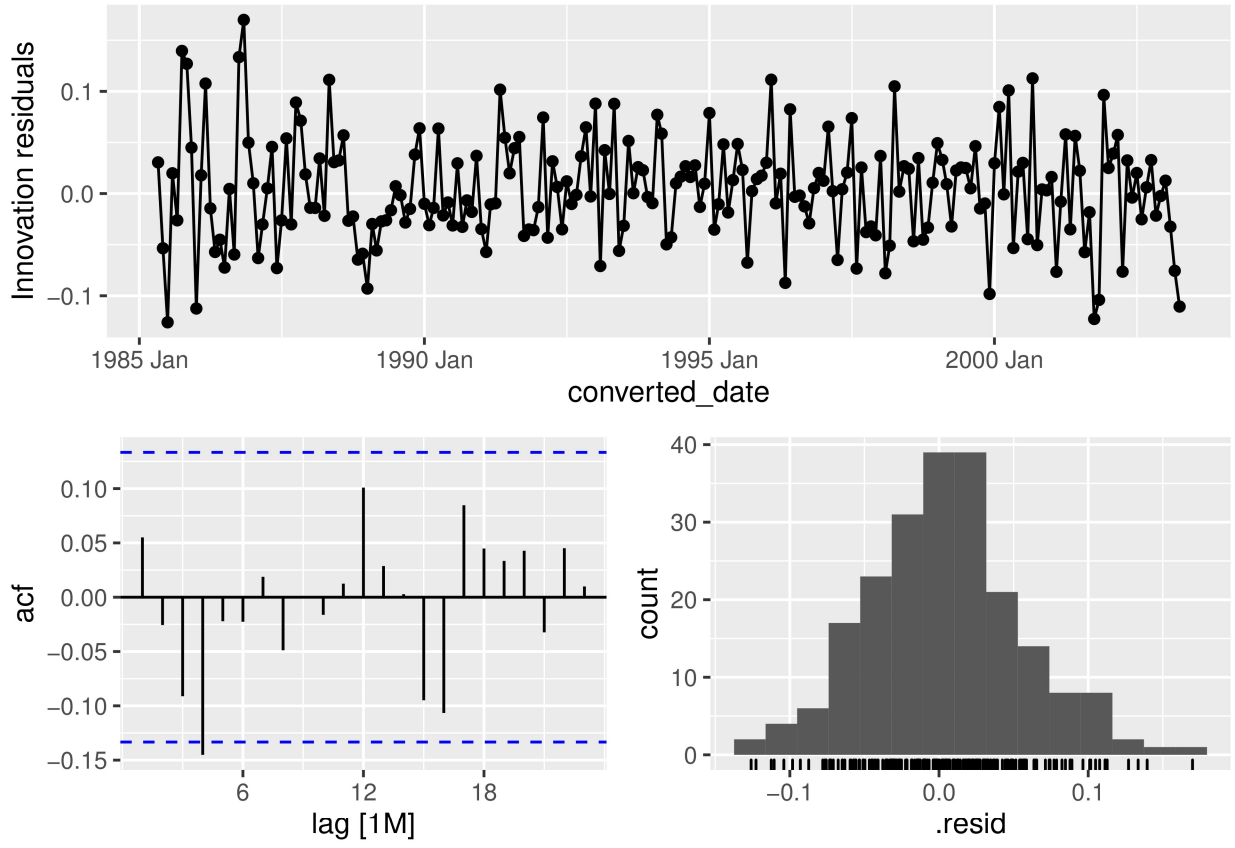
```
## Series: visitors
## Model: ETS(A,Ad,A)
## Transformation: log(visitors)
##   Smoothing parameters:
##     alpha = 0.6293372
##     beta  = 0.0006256646
##     gamma = 0.0001292349
##     phi   = 0.979999
##
##   Initial states:
##      l[0]       b[0]        s[0]      s[-1]      s[-2]      s[-3]      s[-4]
```

```
##  4.480591 0.01757652 -0.05231794 0.06496332 0.08712093 -0.0246398 0.2866596
##       s[-5]      s[-6]      s[-7]       s[-8]      s[-9]     s[-10]     s[-11]
##  0.09212879 0.04009071 -0.1080958 -0.05590228 0.02919788 -0.1576367 -0.2015687
##
##   sigma^2:  0.0029
##
##       AIC      AICc       BIC
## -84.01131 -80.53923 -23.25630
```

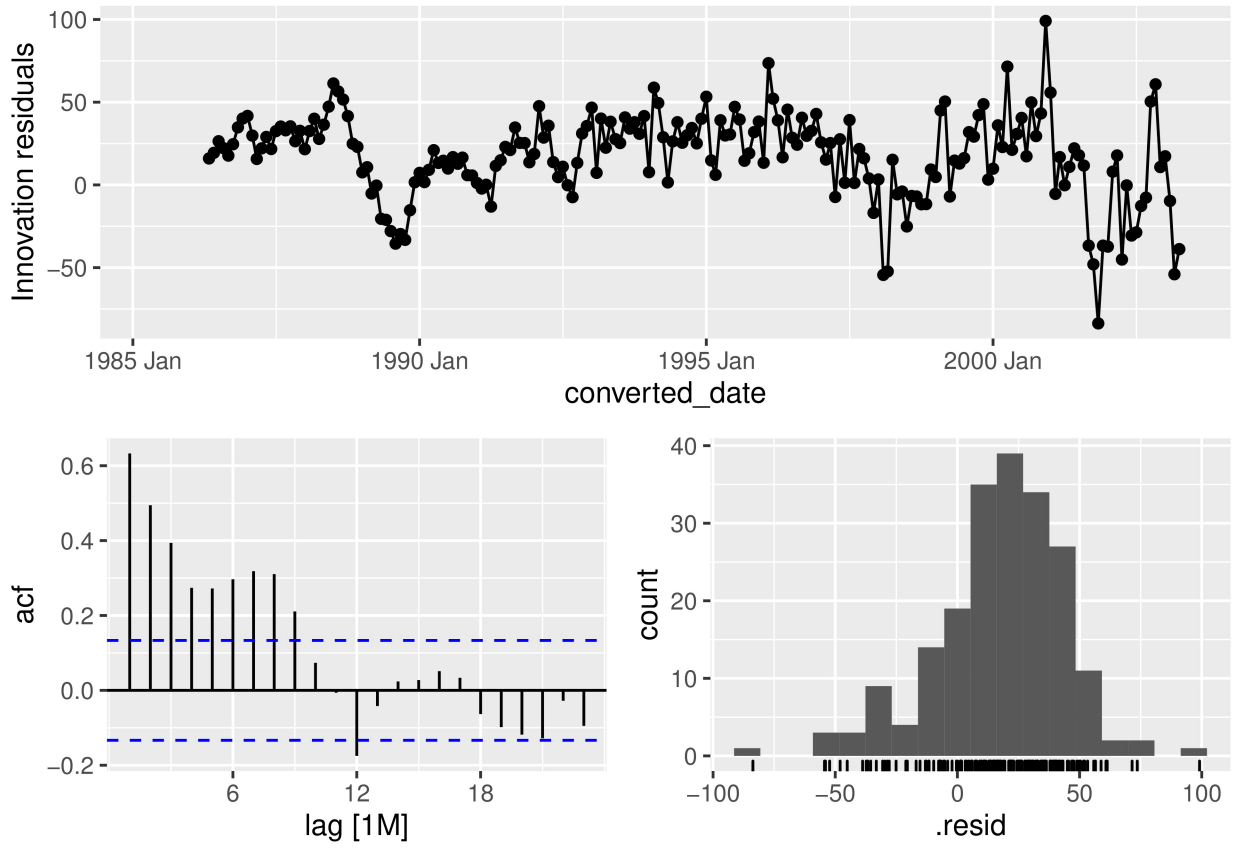
```
report(ets_fit[2])
```

```
## Series: visitors
## Model: SNAIVE
##
## sigma^2: 674.9436
```

```
gg_tsresiduals(ets_fit[1])
```




```
gg_tsresiduals(ets_fit[2])
```




```
augment(ets_fit[1]) %>%  features(.innov, ljung_box, lag=24, dof=7)
```

```
## # A tibble: 1 x 3
##   .model   lb_stat lb_pvalue
##   <chr>      <dbl>     <dbl>
## 1 ets_auto    20.1     0.268
```

```
augment(ets_fit[2]) %>%  features(.innov, ljung_box, lag=24, dof=7)
```

```
## # A tibble: 1 x 3
##   .model         lb_stat lb_pvalue
##   <chr>            <dbl>     <dbl>
## 1 Seasonal_Naive    295.         0
```

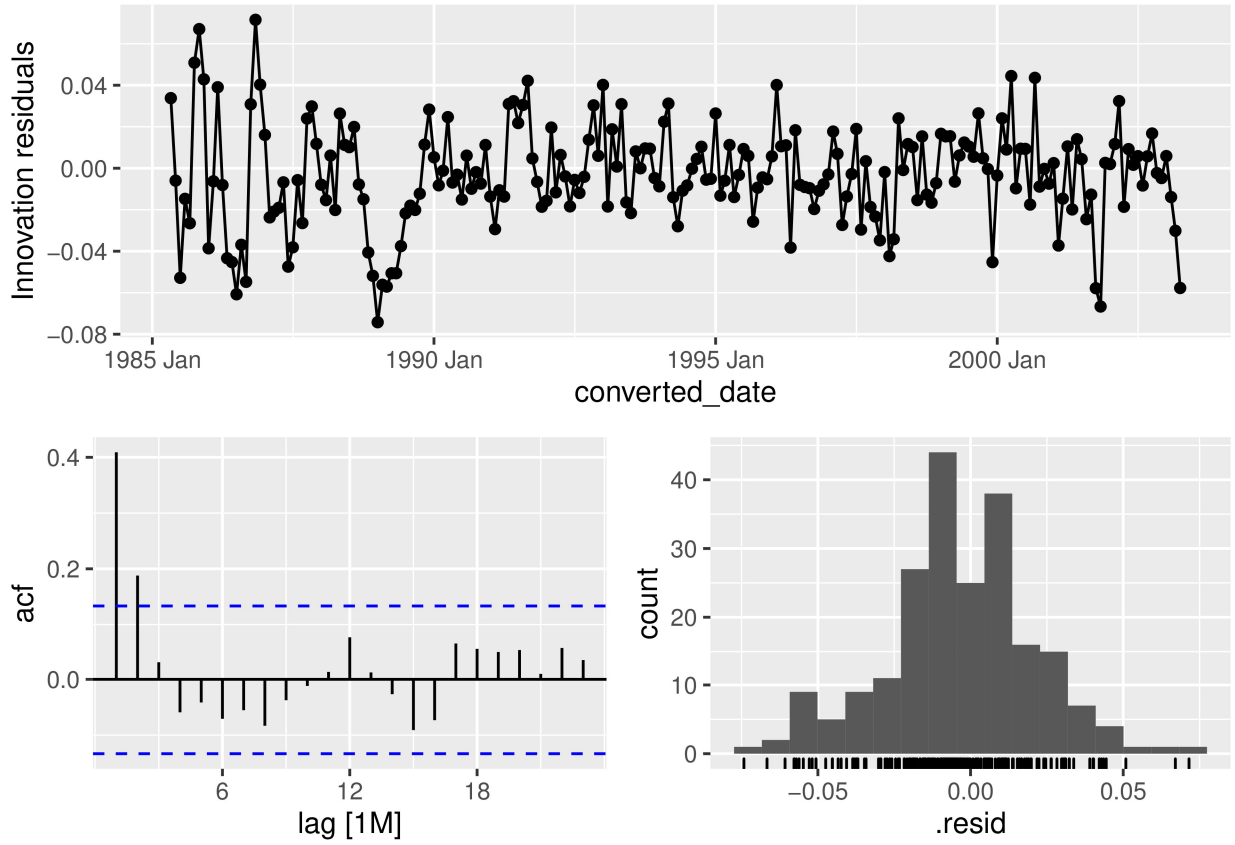
```
# Accuracy and residuals for the additive ets model
accuracy(bc_fc,vis_data)
```

```
## # A tibble: 1 x 10
##   .model      .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>       <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Box_cox_ets Test   409.  415.  409.  95.1  95.1  15.7  13.3 0.436
```

```
report(bc_fit)
```

```
## Series: visitors
## Model: ETS(A,A,A)
## Transformation: log(visitors)
##   Smoothing parameters:
##     alpha = 0.2746539
##     beta  = 0.01655378
##     gamma = 0.0001006887
##
##   Initial states:
##      l[0]       b[0]        s[0]      s[-1]      s[-2]       s[-3]     s[-4]
##  2.386165 0.01137379 -0.02295132 0.02290911 0.03788298 -0.01057209 0.1199257
##       s[-5]      s[-6]     s[-7]       s[-8]     s[-9]      s[-10]     s[-11]
##  0.03588181 0.01414548 -0.046475 -0.01972517 0.0158953 -0.06509136 -0.0818254
##
##   sigma^2:  7e-04
##
##       AIC      AICc       BIC
## -403.0012 -399.9103 -345.6214
```

```
gg_tsresiduals(bc_fit)
```



```
augment(bc_fit) %>%  features(.innov, ljung_box, lag=24, dof=7)
```

```
## # A tibble: 1 x 3
##   .model      lb_stat  lb_pvalue
##   <chr>         <dbl>      <dbl>
## 1 Box_cox_ets    58.9 0.00000158
```

Snaive is giving p value 0, that means model errors are autocorrelated and we cannot not trust the model. Further comparing the rmse and AIC values we can tell that the Auto ETS model is the best to use. ## Question 3

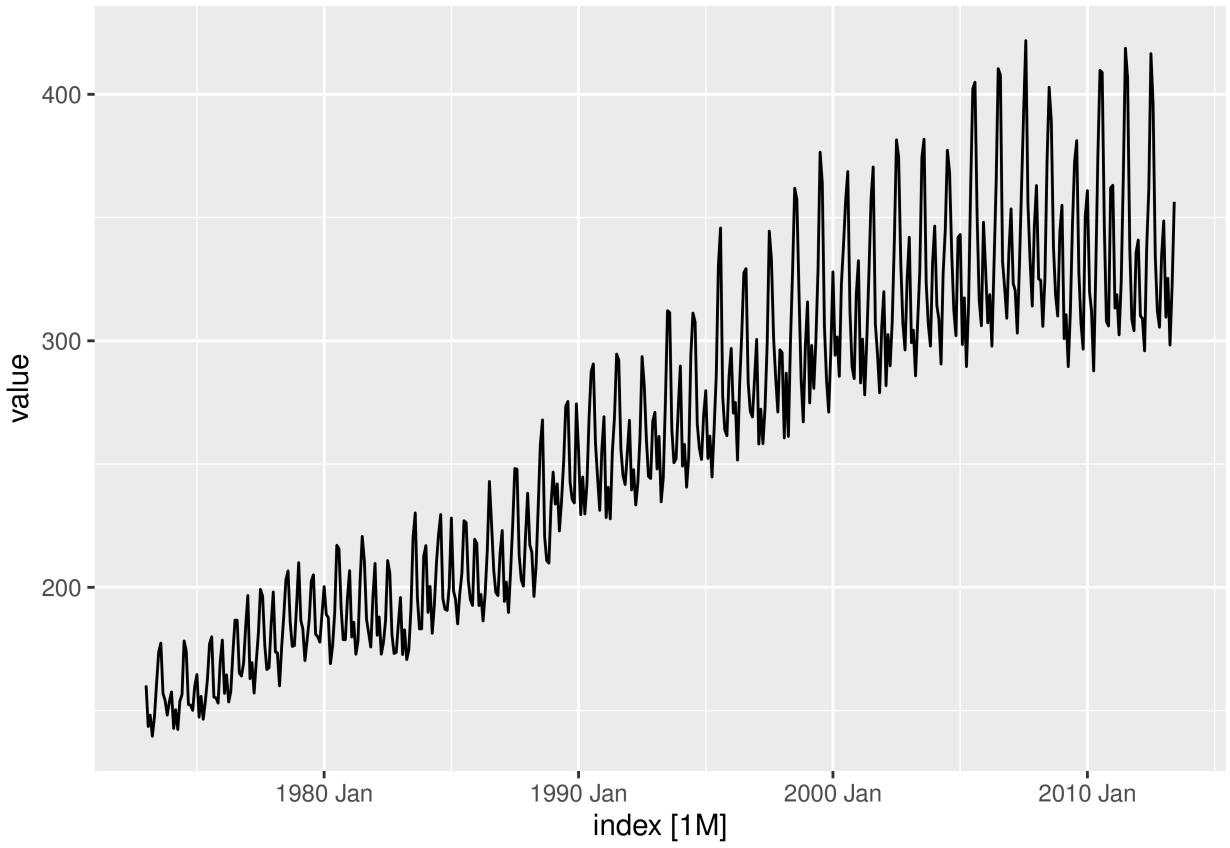
3. Consider usmelec (usmelec.csv), the total net generation of electricity (in billion kilowatt hours) by the U.S. electric industry (monthly for the period January 1973 – June 2013). In general there are two peaks per year: in mid-summer and mid-winter. (Total 36 points)

3.1 Examine the 12-month moving average of this series to see what kind of trend is involved. (4 points)

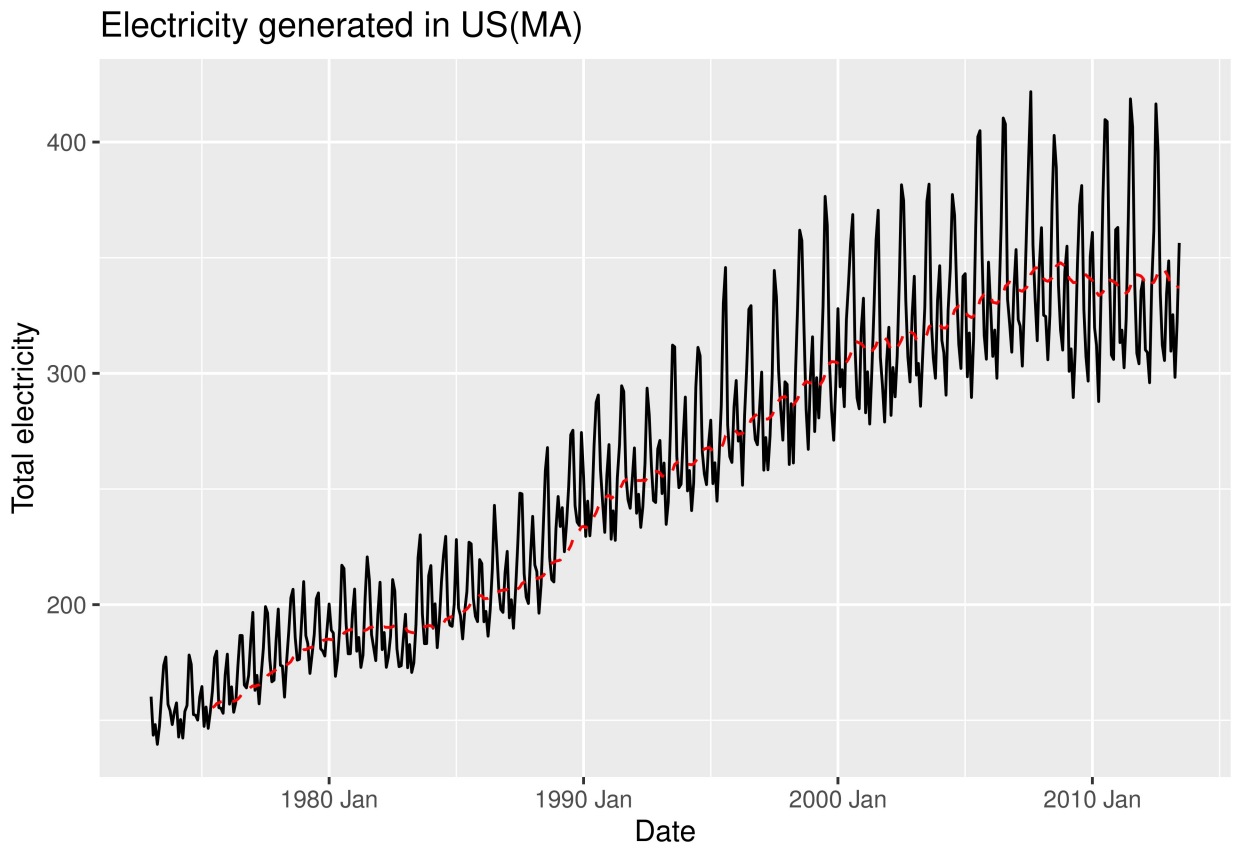
```
library(tidyquant)
library(ggplot2)
usmelec_df <- readr::read_csv("usmelec.csv",show_col_types = FALSE)

# Convert to tsibble with index at date
usmelec_df %>%
  mutate(index=yearmonth(index)) %>%
  tsibble(index= index) ->
  usmelec_data
head(usmelec_data)
```

```
## # A tsibble: 6 x 2 [1M]
##      index value
##      <mth> <dbl>
## 1 1973 Jan  160.
## 2 1973 Feb  144.
## 3 1973 Mar  148.
## 4 1973 Apr  140.
## 5 1973 May  147.
## 6 1973 Jun  161.
```

```
autoplot(usmelec_data)
```

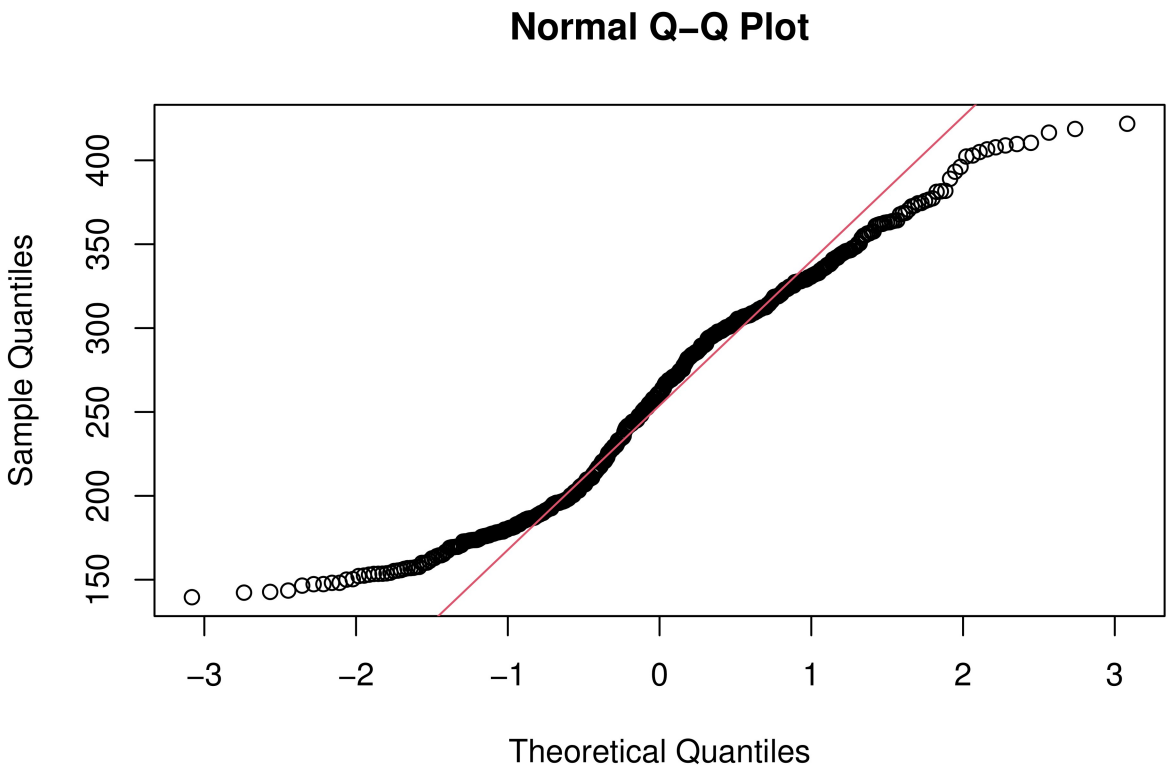



```
usmelec_ma <- usmelec_data %>%
  ggplot(aes(x = index, y = value)) +
  geom_line(color = "black") +
  geom_ma(ma_fun = SMA, n = 30, color = "red") +
  labs(x = "Date", y = "Total electricity", title = "Electricity generated in US(MA)")
usmelec_ma
```



3.2 Do the data need transforming? If so, find a suitable transformation. (4 points)

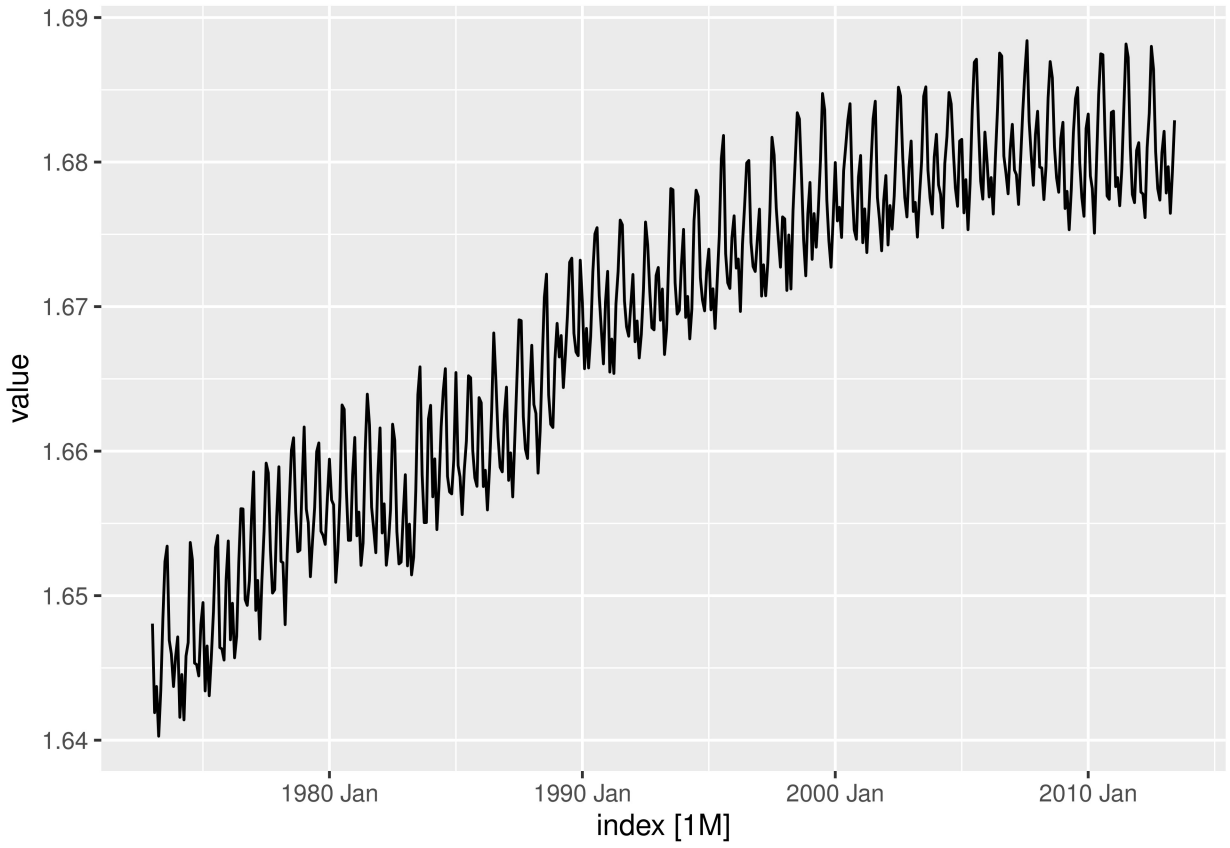
```
#QQ plot for the data
qqnorm(usmelec_data$value)
qqline(usmelec_data$value, col = 2)
```



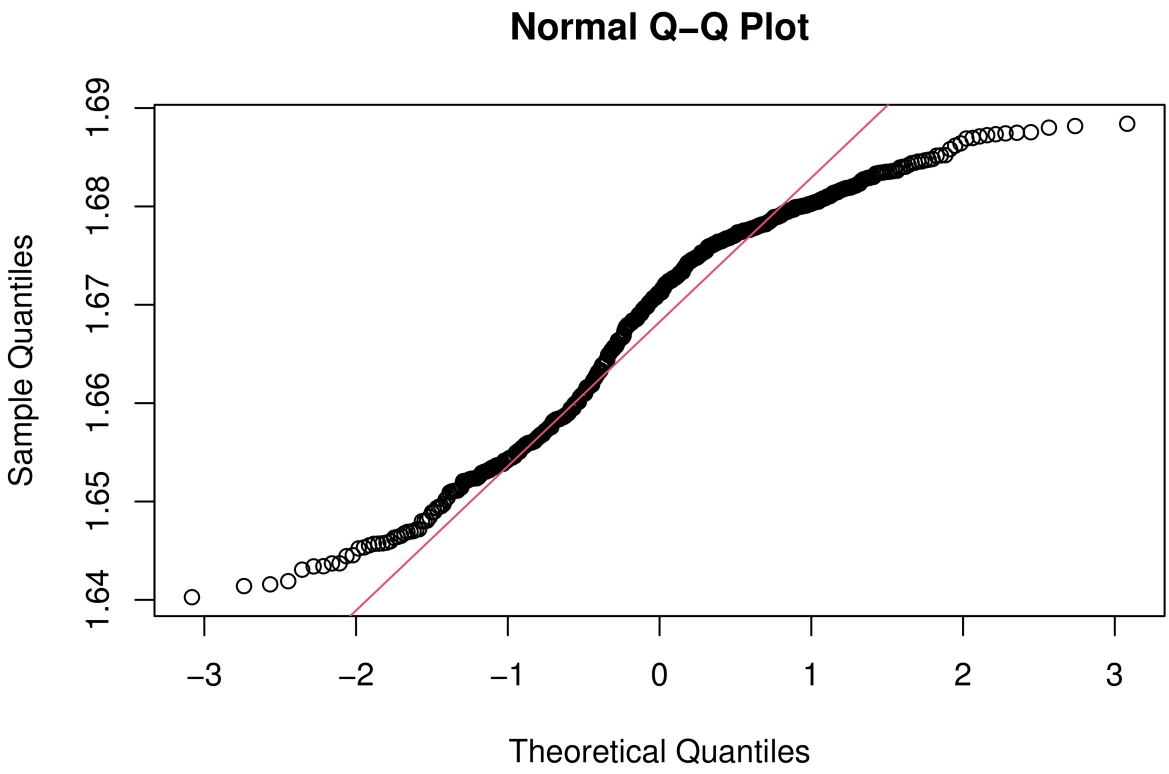

```
#box cox lambda variable
lambda = usmelec_data %>% features(value, features = guerrero)
lambda
```

```
## # A tibble: 1 x 1
##   lambda_guerrero
##             <dbl>
## 1          -0.574
```

```
#applying box cox transform on the data
usmelec_bc = usmelec_data %>% mutate(value = box_cox(value, lambda))
autoplot(usmelec_bc)
```




```
#QQ plot for the box cox tranform data
qqnorm(usmelec_bc$value)
qqline(usmelec_bc$value, col = 2)
```



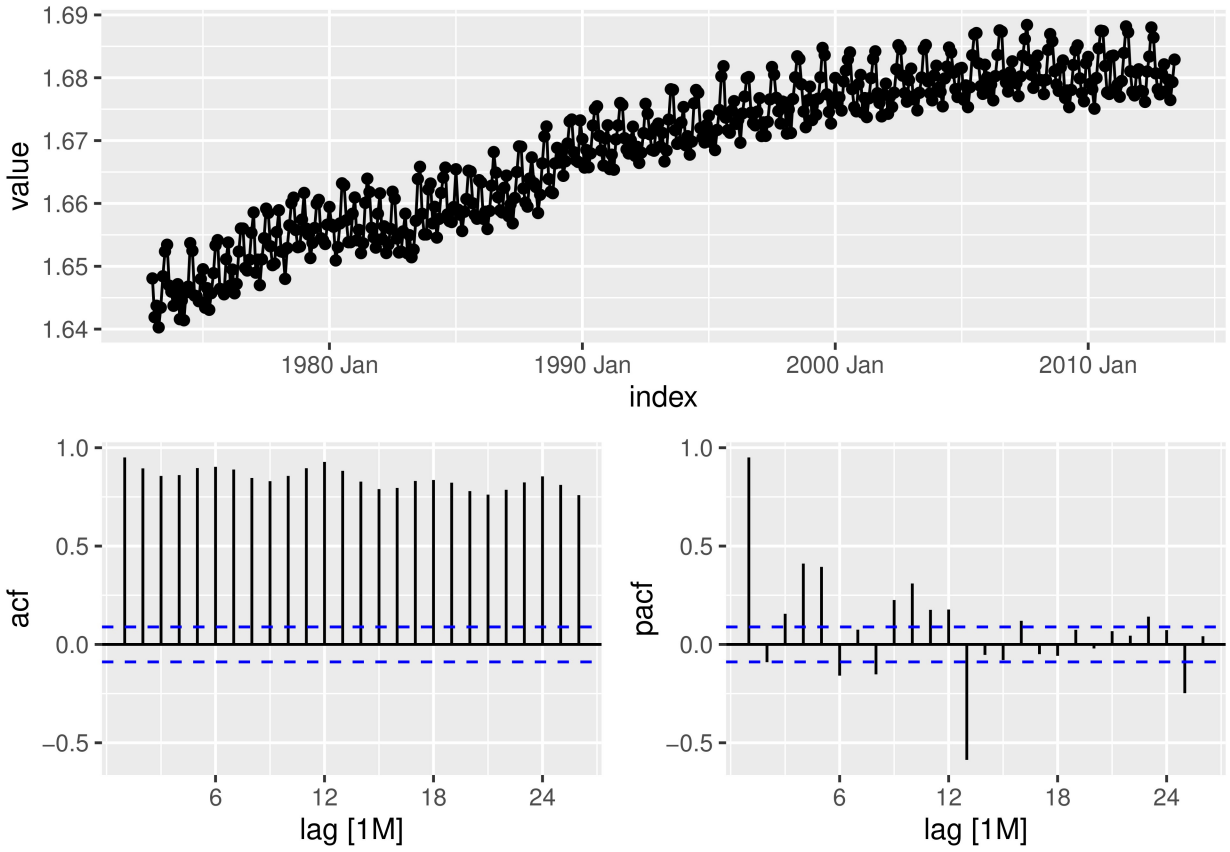
Yes, as we can observe from the qq plot, the box cox transformation eliminates non linearity, differeing variances and variability from the data. Our data has been transformed with a new normal distribution.

3.3 Are the data stationary? If not, find an appropriate differencing which yields stationary data. (4 points)

```
usmelec_bc %>% features(value, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1      7.83        0.01
```

```
usmelec_bc %>%gg_tsdisplay(value, plot_type='partial')
```

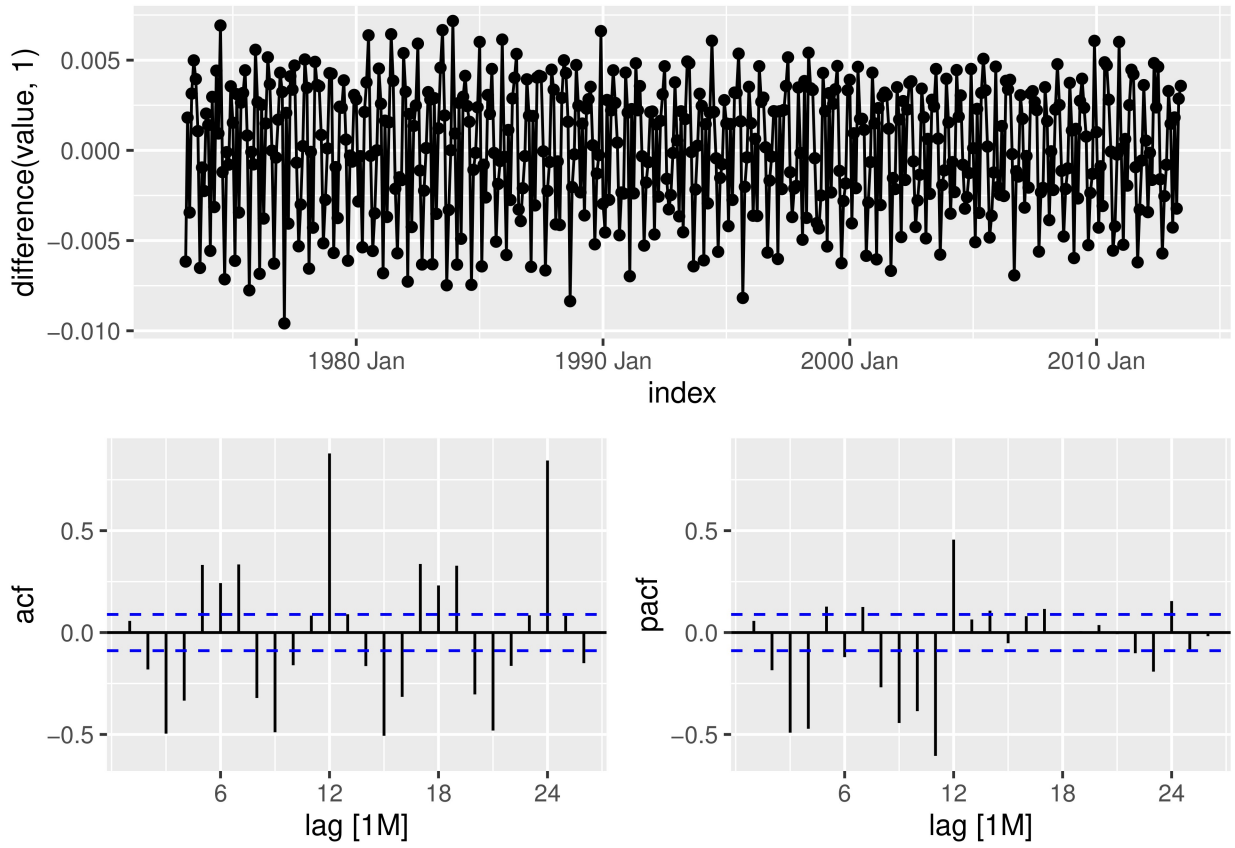


kpss value is less than 0.05 so we can tell that the data is not stationaty

```
usmelec_bc %>% features(difference(value,1), unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0194         0.1
```

```
usmelec_bc %>%gg_tsdisplay(difference(value,1), plot_type='partial')
```



Applying first diffference to the data makes it staionary(kpss value > 0.05)

3.4 Identify a couple of ARIMA models that might be useful in describing the time series. Which of your models is the best according to their AIC values? (6 points)

```
fit_usmelec_bc <- usmelec_bc %>%
  model(
    arima_auto_usmelec = ARIMA(value, stepwise = FALSE, approx = FALSE, ic='aic'),
    arima_elec = ARIMA(value~0+pdq(1,1,1)+PDQ(1,1,1))
  )
accuracy(fit_usmelec_bc)
```

```
## # 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 arima_auto_~ Trai~ -6.05e-5 0.00111 8.39e-4 -0.00363 0.0503 0.559 0.575 0.0117
## 2 arima_elec   Trai~ -5.95e-5 0.00111 8.44e-4 -0.00357 0.0506 0.563 0.577 0.0127
```

```
report(fit_usmelec_bc[1])
```

```
## Series: value
## Model: ARIMA(1,1,1)(2,1,1)[12]
##
## Coefficients:
##          ar1      ma1    sar1     sar2     sma1
##       0.3888  -0.8265  0.0403  -0.0958  -0.8471
```

```
## s.e.  0.0630    0.0375  0.0555    0.0531    0.0341
##
## sigma^2 estimated as 1.274e-06:  log likelihood=2547.32
## AIC=-5082.63   AICc=-5082.45   BIC=-5057.68
```

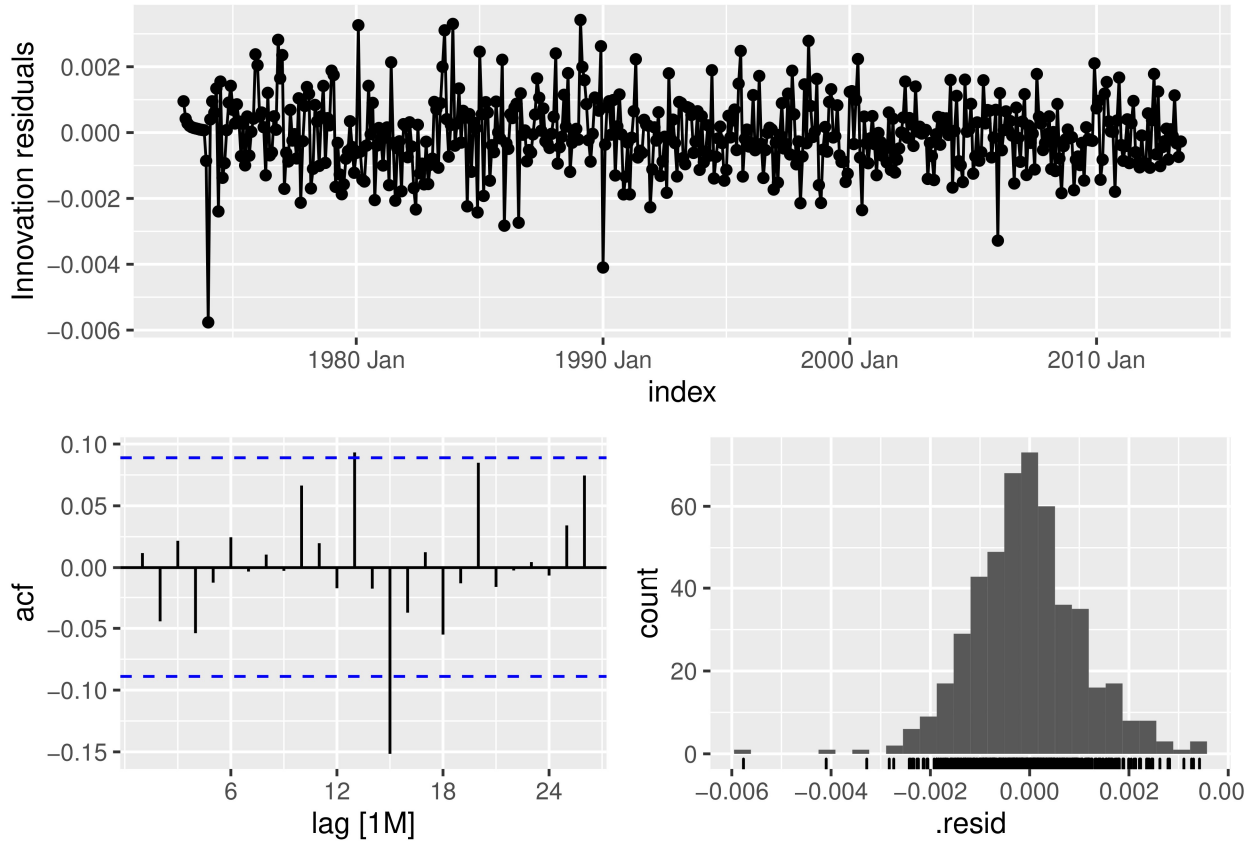
```
report(fit_usmelec_bc[2])
```

```
## Series: value
## Model: ARIMA(1,1,1)(1,1,1)[12]
##
## Coefficients:
##          ar1      ma1    sar1     sma1
##       0.3872  -0.8273  0.0646  -0.8744
## s.e.  0.0629   0.0374  0.0534   0.0270
##
## sigma^2 estimated as 1.28e-06:  log likelihood=2545.72
## AIC=-5081.44   AICc=-5081.31   BIC=-5060.64
```

The model with non season pdq(1,1,1) & seasonal pdq(1,1,1) has almost same but lower AIC value by a difference of order 10^-3 than Auto ARIMA but the rmse value of Auto ARIMA is better.

3.5 Estimate the parameters of your best model and do diagnostic testing on the residuals. Do the residuals resemble white noise? If not, try to find another ARIMA model which fits better. (4 points)

```
gg_tsresiduals(fit_usmelec_bc[1])
```



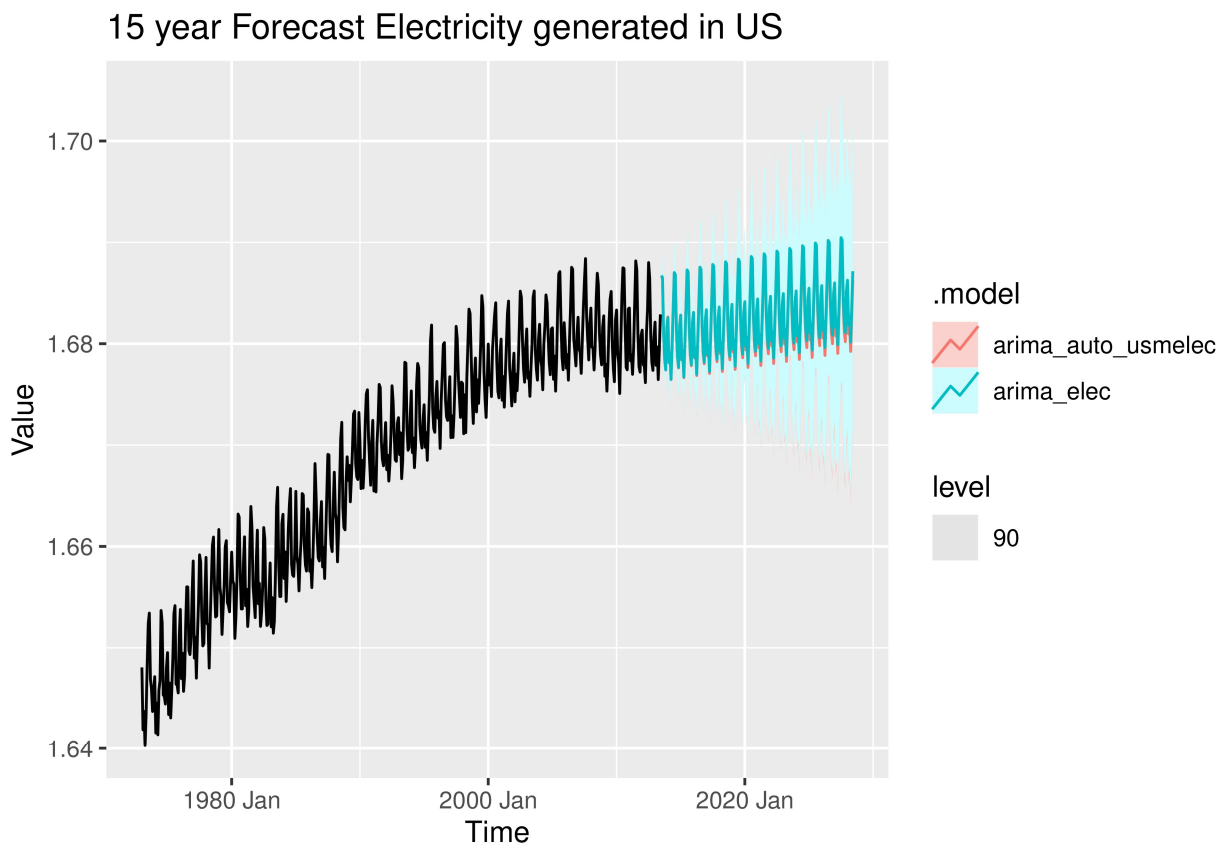
```
report(fit_usmelec_bc[1])
```

```
## Series: value
## Model: ARIMA(1,1,1)(2,1,1)[12]
##
## Coefficients:
##          ar1      ma1    sar1     sar2     sma1
##       0.3888  -0.8265  0.0403  -0.0958  -0.8471
## s.e.  0.0630   0.0375  0.0555   0.0531   0.0341
##
## sigma^2 estimated as 1.274e-06:  log likelihood=2547.32
## AIC=-5082.63   AICc=-5082.45   BIC=-5057.68
```

From the ACF plot we can tell that the residuals do not resemble white noise.

3.6 Forecast the next 15 years of electricity generation by the U.S. electric industry. Get the latest figures from the EIA (https://www.eia.gov/totalenergy/data/monthly/#electricity) to check the accuracy of your forecasts. (8 points)

```
auto_arima_usmelec_bc_fc <- fit_usmelec_bc %>% forecast(h = "15 years")
auto_arima_usmelec_bc_fc %>% autoplot(usmelec_bc,level = 90) +labs(title="15 year Forecast Electricity
```



3.7. Eventually, the prediction intervals are so wide that the forecasts are not particularly useful. How many years of forecasts do you think are sufficiently accurate to be usable? (6 points)

Comparing with the results on the government portal and considering the width of forecast we can only consider the next 4-5 year of predictions to be usable with a margin of error.

```
auto_arima_usmelec_bc_fc <- fit_usmelec_bc %>% forecast(h = "5 years")
auto_arima_usmelec_bc_fc %>% autoplot(usmelec_bc,level = 90) +labs(title="15 year Forecast Electricity
```