Time series

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)

```
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following object is masked from 'package:tsibble':
##
##     key

## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

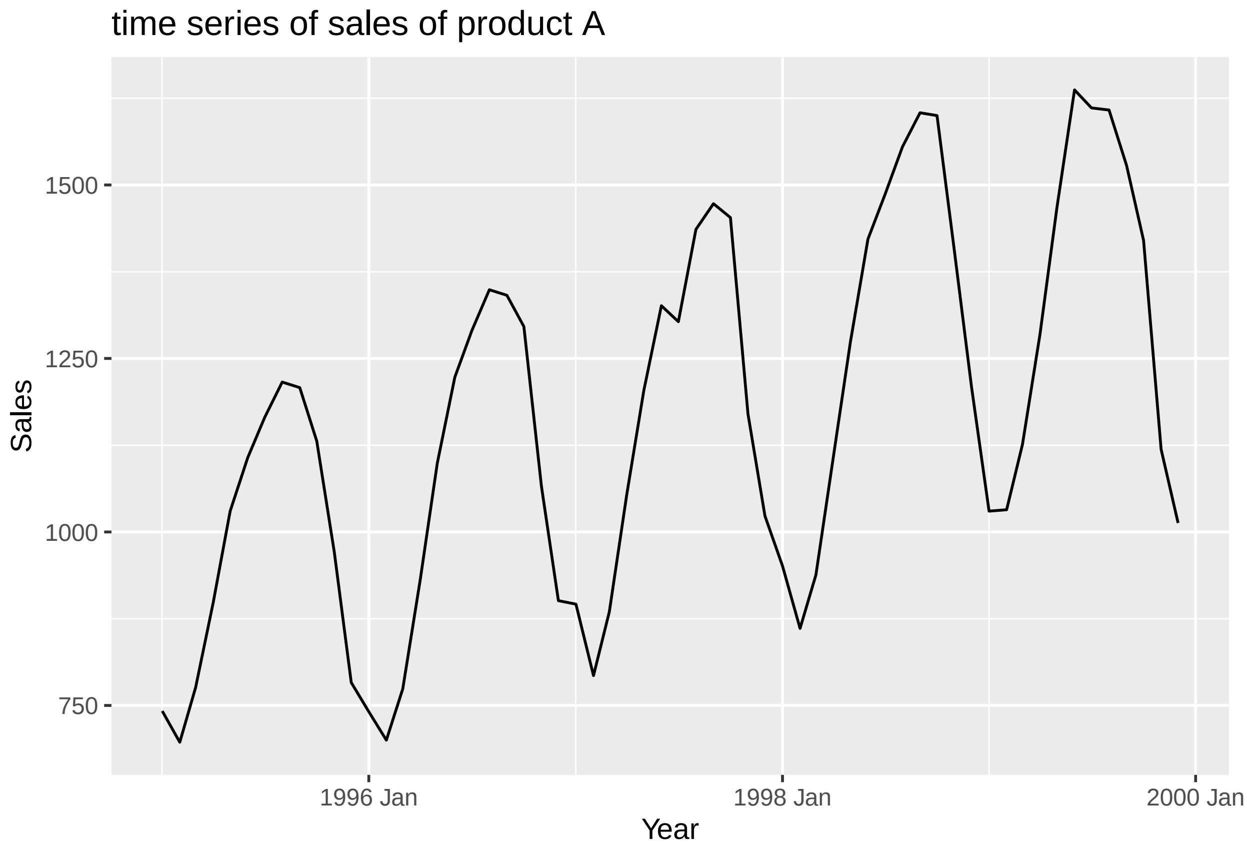
```
data <- fread("plastics.csv")
data %>% mutate(date = yearmonth(date)) %>% tsibble(index = date) -> datats
head(datats)
```

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

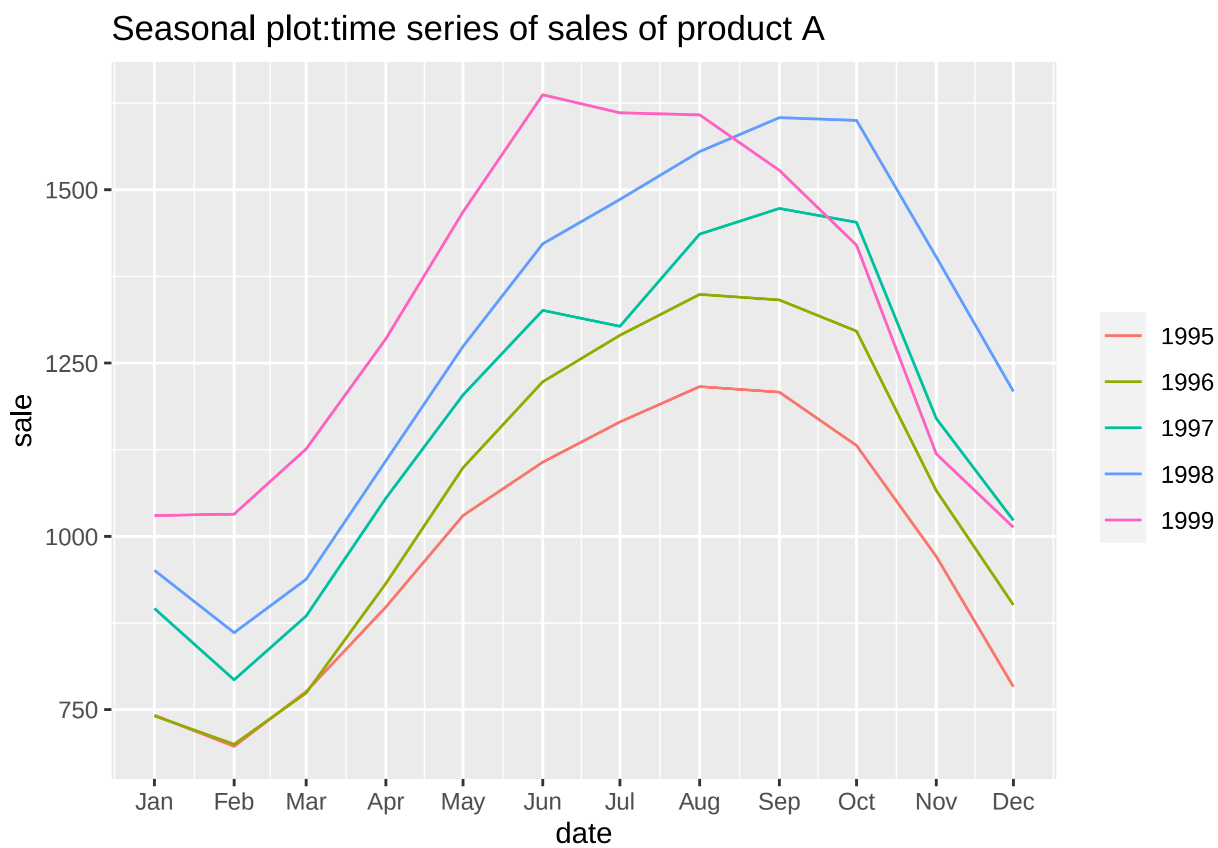
1.2 Plot the time series of sales of product A. Can you identify seasonal fluctuations and/or a trend-cycle? (2 points) library(ggplot2)

```
# Plot time series of sales
autoplot(datats)+ggtitle("time series of sales of product A") + ylab("Sales") + xlab("Year")
```

```
## Plot variable not specified, automatically selected `.vars = sale`
```




```
datats %>% gg_season(sale) +
  labs(title = "Seasonal plot:time series of sales of product A")
```

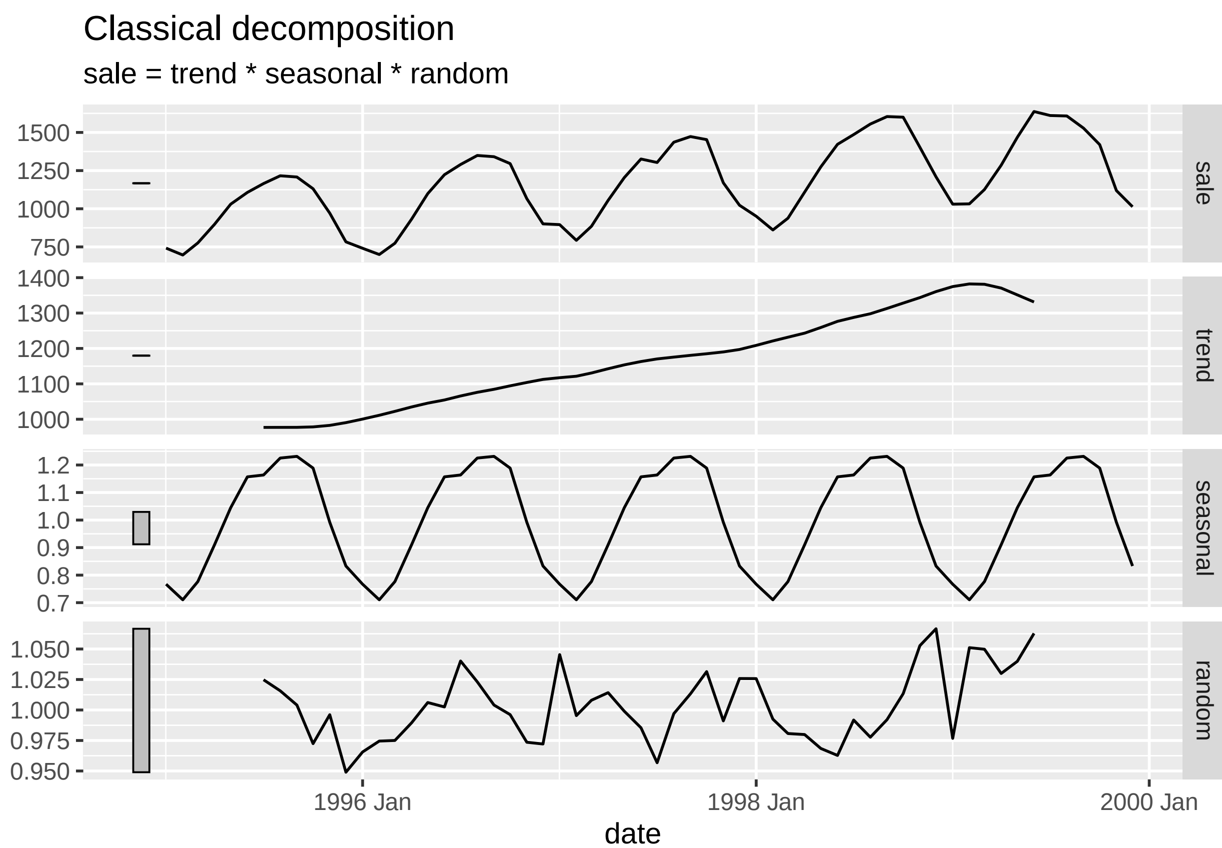


Seasonality can be observed in the time series of sales of product as the data in going up to peak then going down and pattern is repeating for equal intervals. The trend of the plot is increasing. The data is seasonal and increasing in nature.

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

```r
datats %>%
  model(classical_decomposition(sale, type = "multiplicative")) %>%
  components() %>%
  autoplot()
```

```
## Warning: Removed 6 rows containing missing values (`geom_line()`).
```



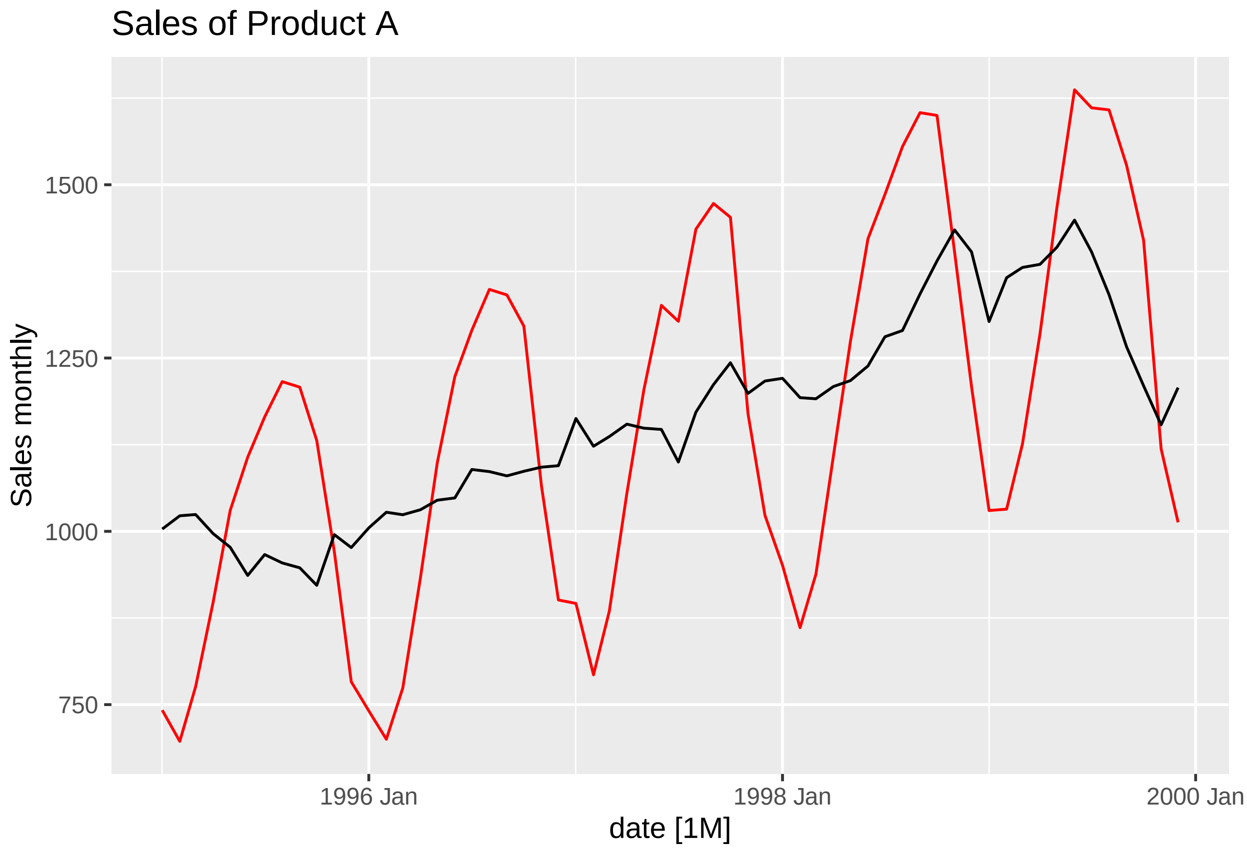
1.4 Do the results support the graphical interpretation from part a? (2 points)

Yes, the results support the graphical interpretation from part a.

From the classical multiplicative decomposition graphs, the trend graph shows increasing trend in the timeline. The seasonal graph also shows seasonality with a peaks at each interval.

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

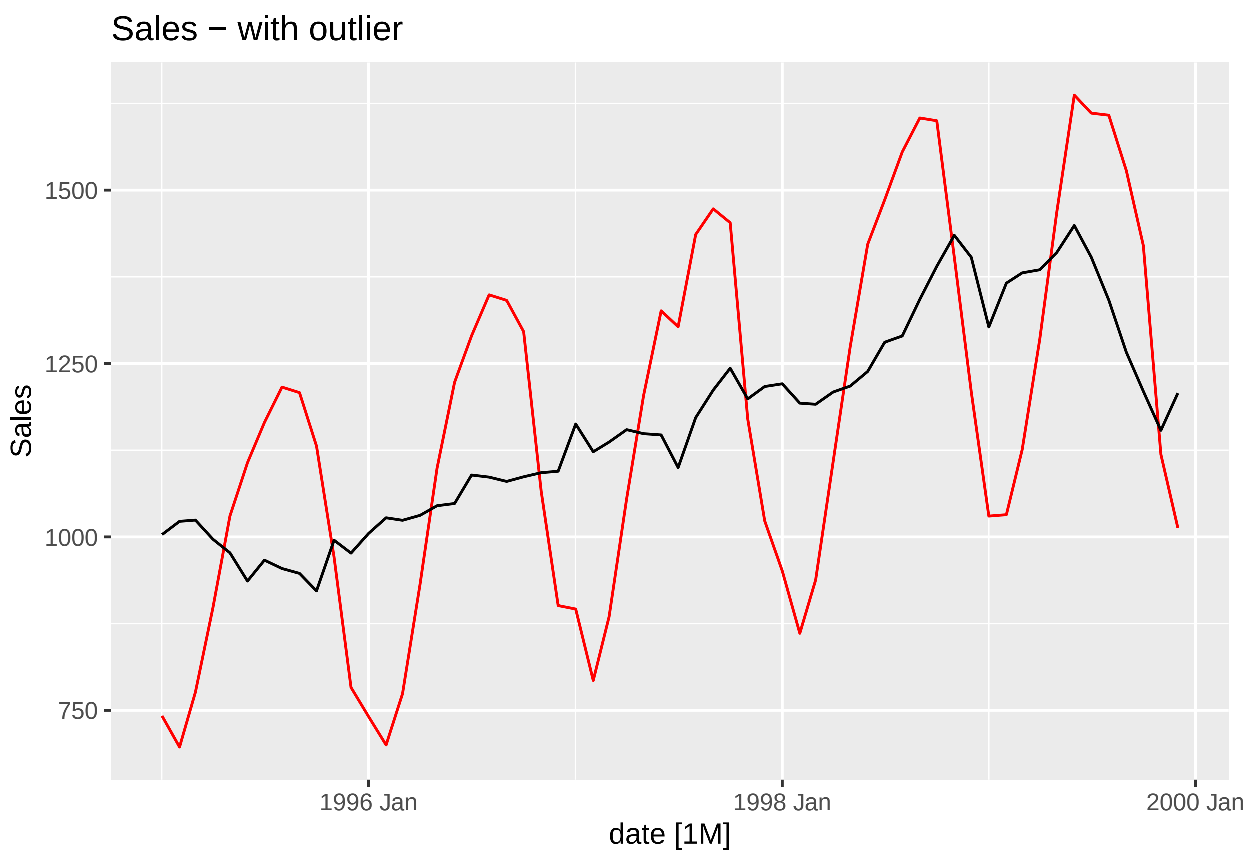
```
model1 <- datats %>%
  model(stl = STL(sale))

datats %>%
  autoplot(sale, color = "red") +
  autolayer(components(model1), season_adjust) +
  ylab("Sales monthly")+
  ggtitle("Sales of Product A")
```



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

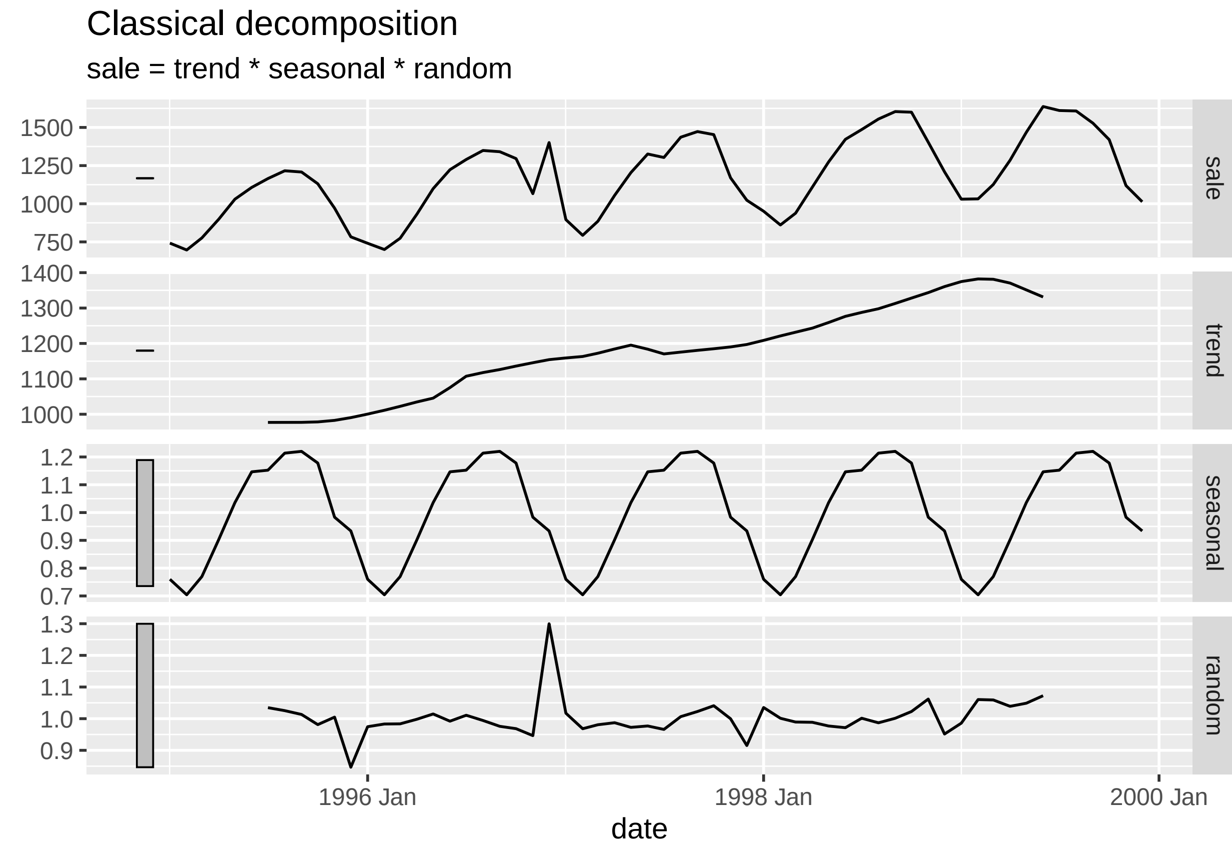
```
library(forecast)
library(ggplot2)

# make a copy of the data with an outlier
datats1 <- datats
datats1[24, "sale"] <- datats1[24, "sale"] + 500

# plot the original and modified data with the seasonally adjusted component
autoplot(datats, color = "red") +
  autolayer(components(model1), season_adjust) +
  ylab("Sales") +
  ggtitle("Sales - with outlier")
```

```
## Plot variable not specified, automatically selected `.vars = sale`
```




```
datats1 %>%
  model(classical_decomposition(sale, type = "multiplicative")) %>%
  components() %>%
  autoplot()
```

```
## Warning: Removed 6 rows containing missing values (`geom_line()`).
```

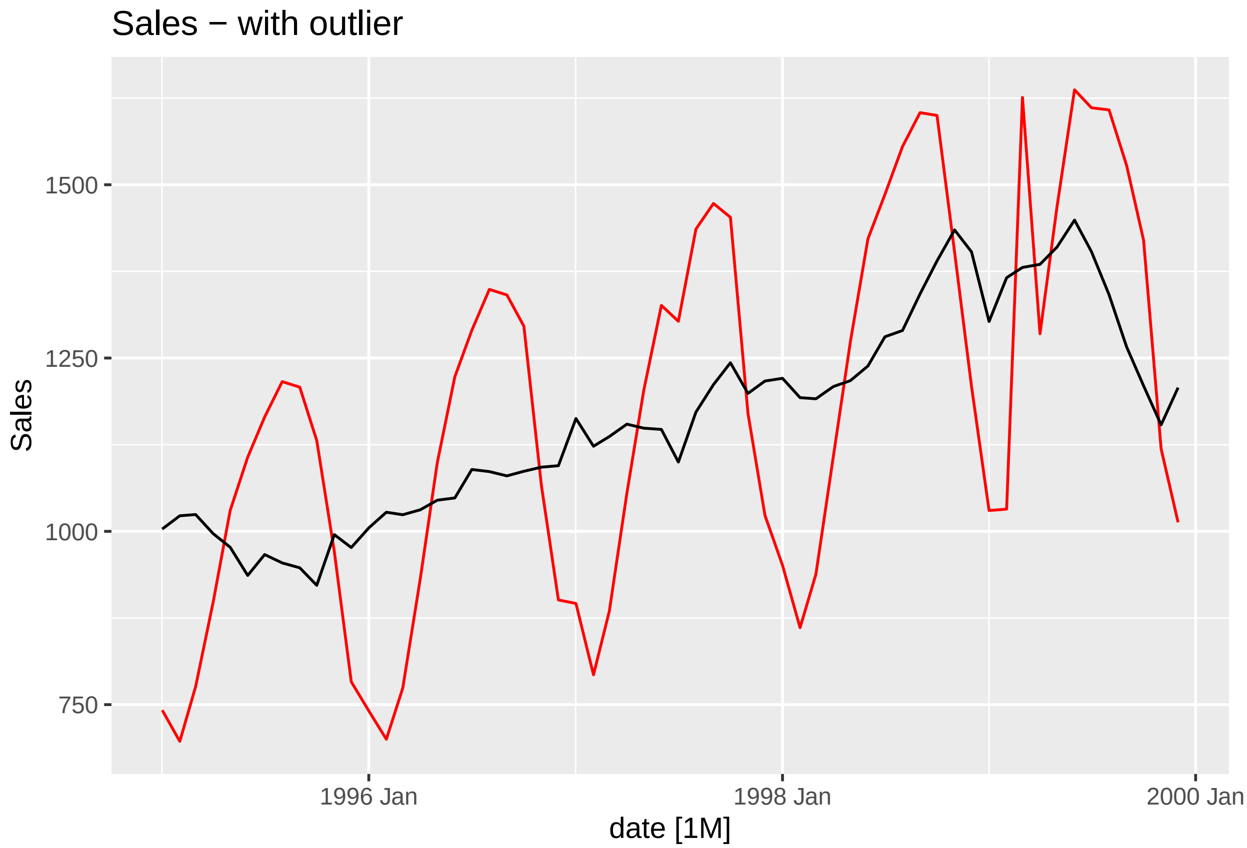


The outlier in the time series affects the trend of time series. The trend has a small peak around 1997 year end due to the outlier. The period of the seasonality does not change, but its shape has changed a little.

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)

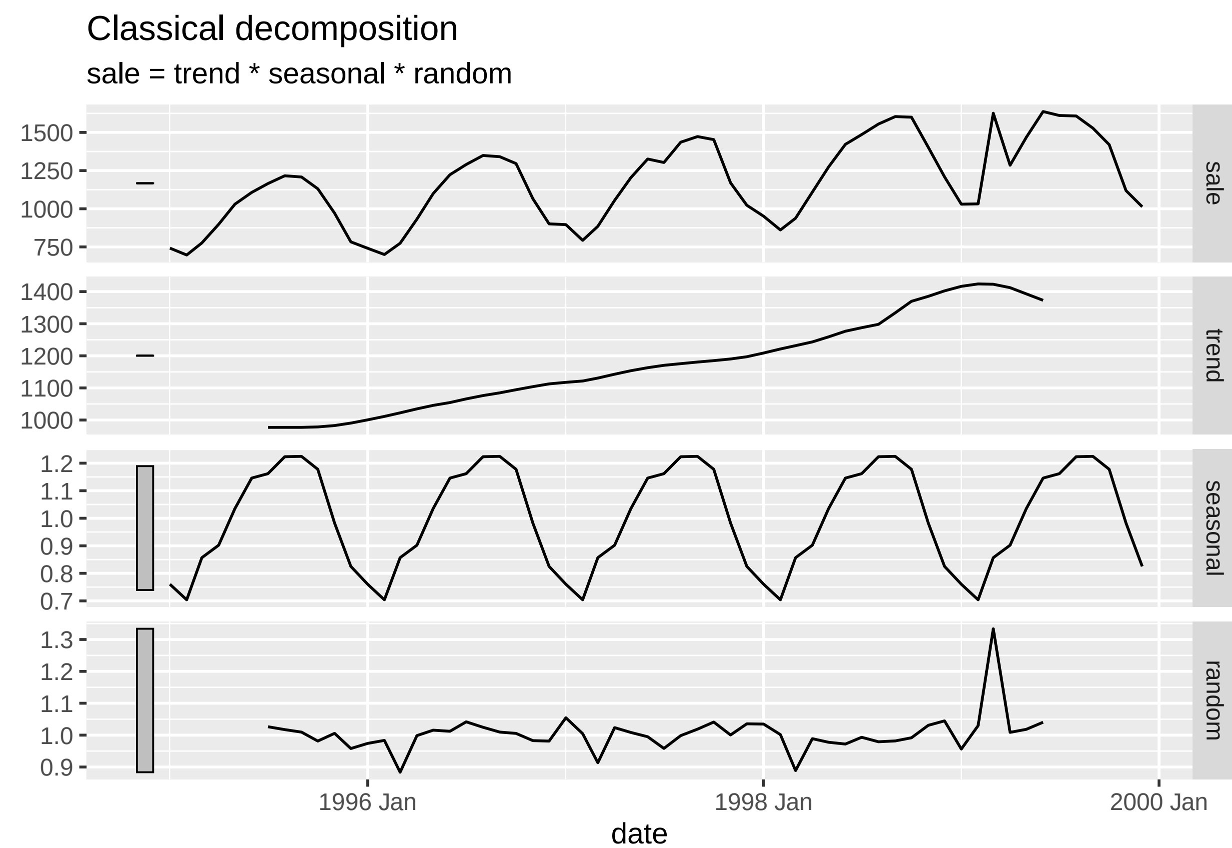
```
datats2 <- datats
datats2[51, "sale"] <- datats2[51, "sale"] + 500

# plot the original and modified data with the seasonally adjusted component
autoplot(datats2, color = "red") +
  autolayer(components(model1), season_adjust) +
  ylab("Sales") +
  ggtitle("Sales - with outlier")
```

```
## Plot variable not specified, automatically selected `.vars = sale`
```




```
datats2 %>%
  model(classical_decomposition(sale, type = "multiplicative")) %>%
  components() %>%
  autoplot()
```

```
## Warning: Removed 6 rows containing missing values (`geom_line()`).
```



The outlier near end of time series has a higher effect on trend, this can be seen in the graphs but has a smaller effect on the seasonality. The is similar to when the outlier is in the middle of time series. The trend increased and has a peak near the end of the time series due to the outlier. The period of the seasonality does not change, but its shape has changed a little.

The outlier location doesn't change any change in trend and seasonality. It can be seen in the decomposition trend graph that where the outlier is present the trend changes there. The shape of the seasonality has also changed a little.

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)

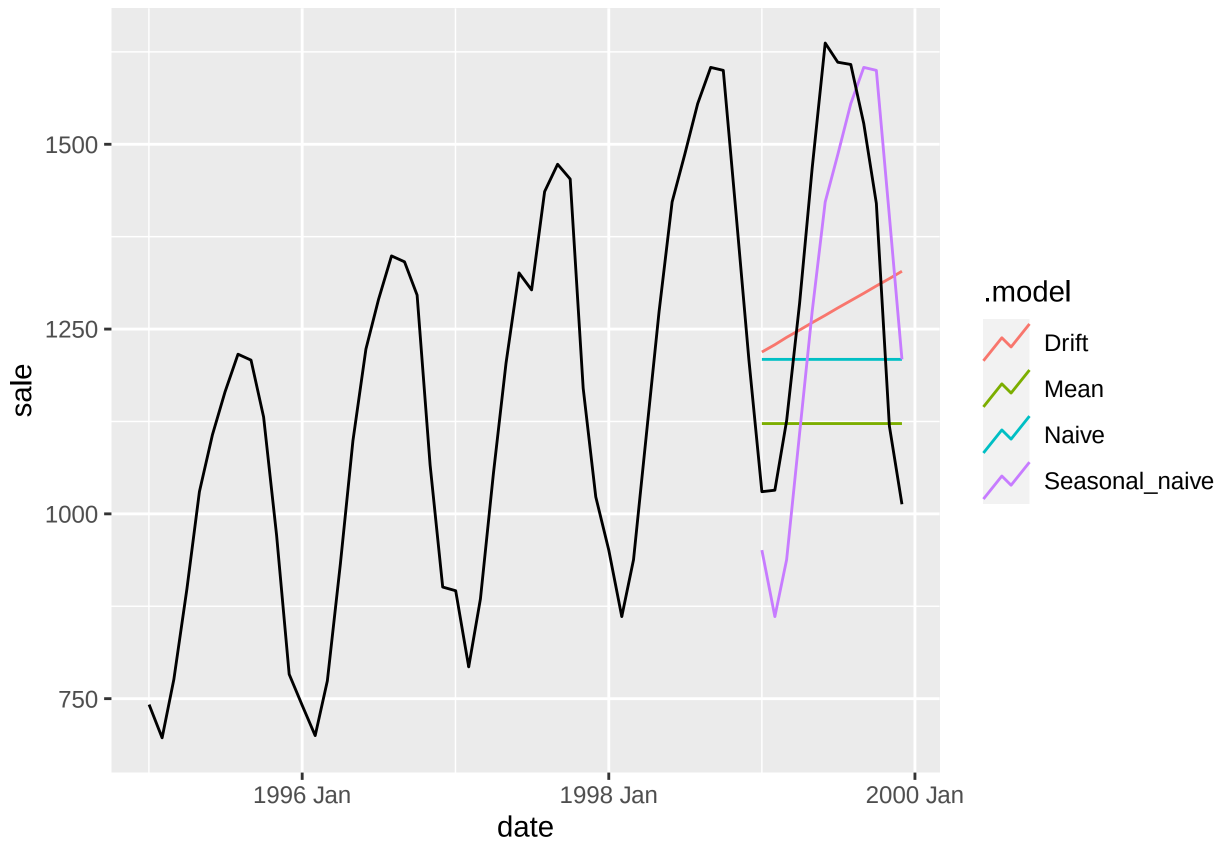
```
traindata <- datats %>% filter(date <= yearmonth("1998-12"))
testdata <- datats %>% filter(date > yearmonth("1998-12"))
```

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 <- traindata %>%
  filter(!is.na(sale)) %>%
  model(
    Seasonal_naive = SNAIVE(sale),
    Naive = NAIVE(sale),
    Drift = RW(sale ~ drift()),
    Mean = MEAN(sale)
  )

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 Seasonal_naive Training  1.02e+ 2  116.  103.  8.52   8.54 1      1    0.602
## 2 Naive          Training  9.94e+ 0  120.  101.  0.410  9.50 0.987  1.04 0.637
## 3 Drift          Training -4.35e-14  120.  101. -0.517  9.49 0.981  1.04 0.637
## 4 Mean           Training  0         255.  217. -5.64  21.0  2.11   2.21 0.867
```

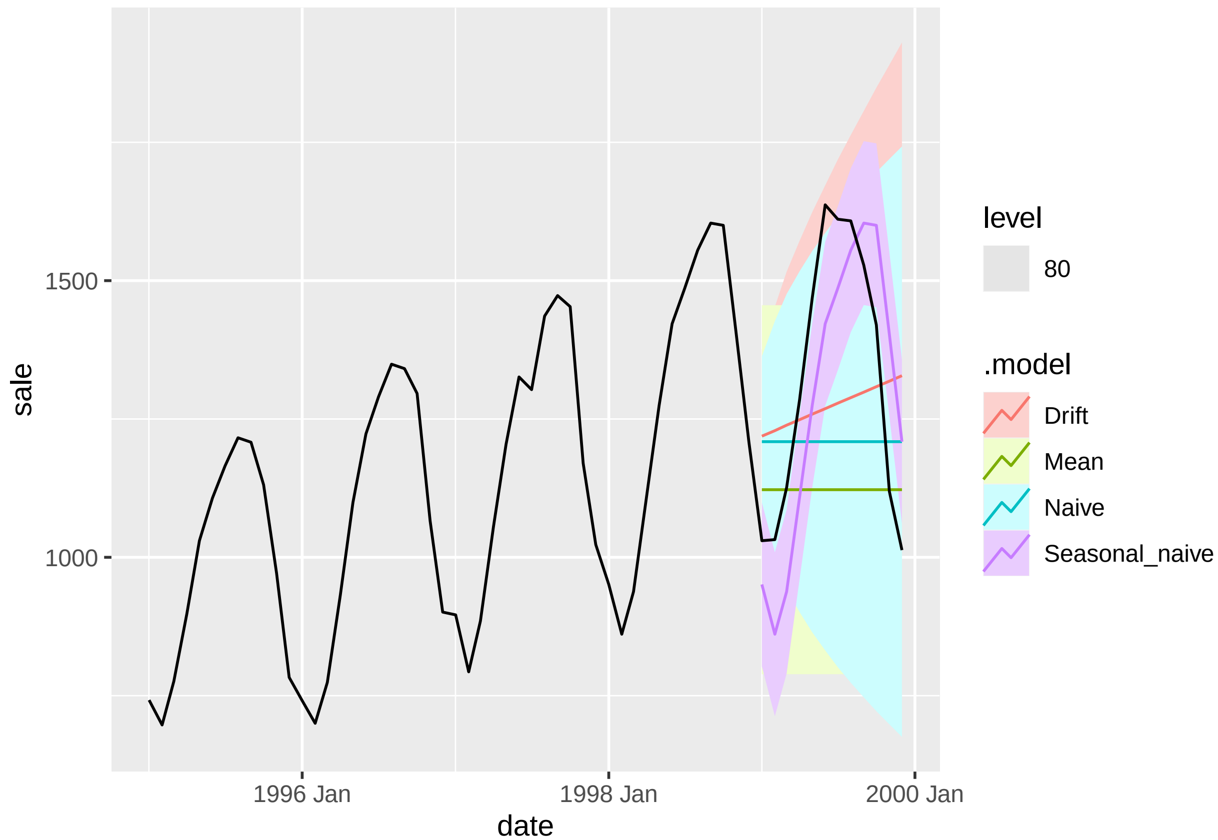
```r
# forecast next year (in training set)
fc <- fit %>% forecast(h = 12)
# plot forecasts and actual data
fc %>% autoplot(datats,level = NULL)
```




```r
# Calculate accuracy
accuracy(fc,datats)
```

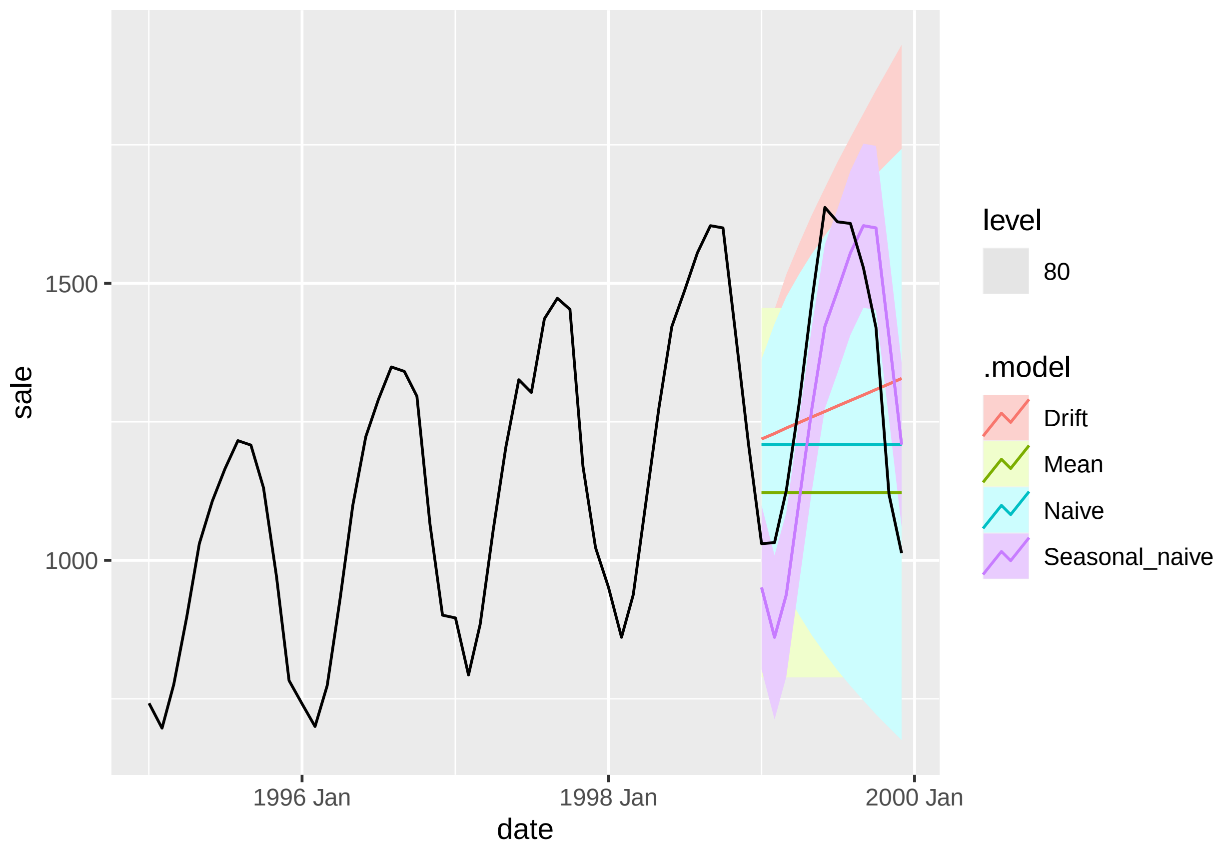
```
## # A tibble: 4 x 10
##   .model         .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>          <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Drift          Test   49.5  239.  218.  0.551  16.6  2.13  2.07 0.689
## 2 Mean           Test  201.   312.  250. 12.3    17.1  2.44  2.70 0.708
## 3 Naive          Test  114.   265.  235.  5.49   17.0  2.29  2.29 0.708
## 4 Seasonal_naive Test   38.8  174.  161.  2.47   12.9  1.57  1.50 0.817
```

```r
# forecast next year (in training set)
fc <- fit %>% forecast(h = 12)
#plot forecasts and actual data
fc %>% autoplot(datats,level = 80)
```

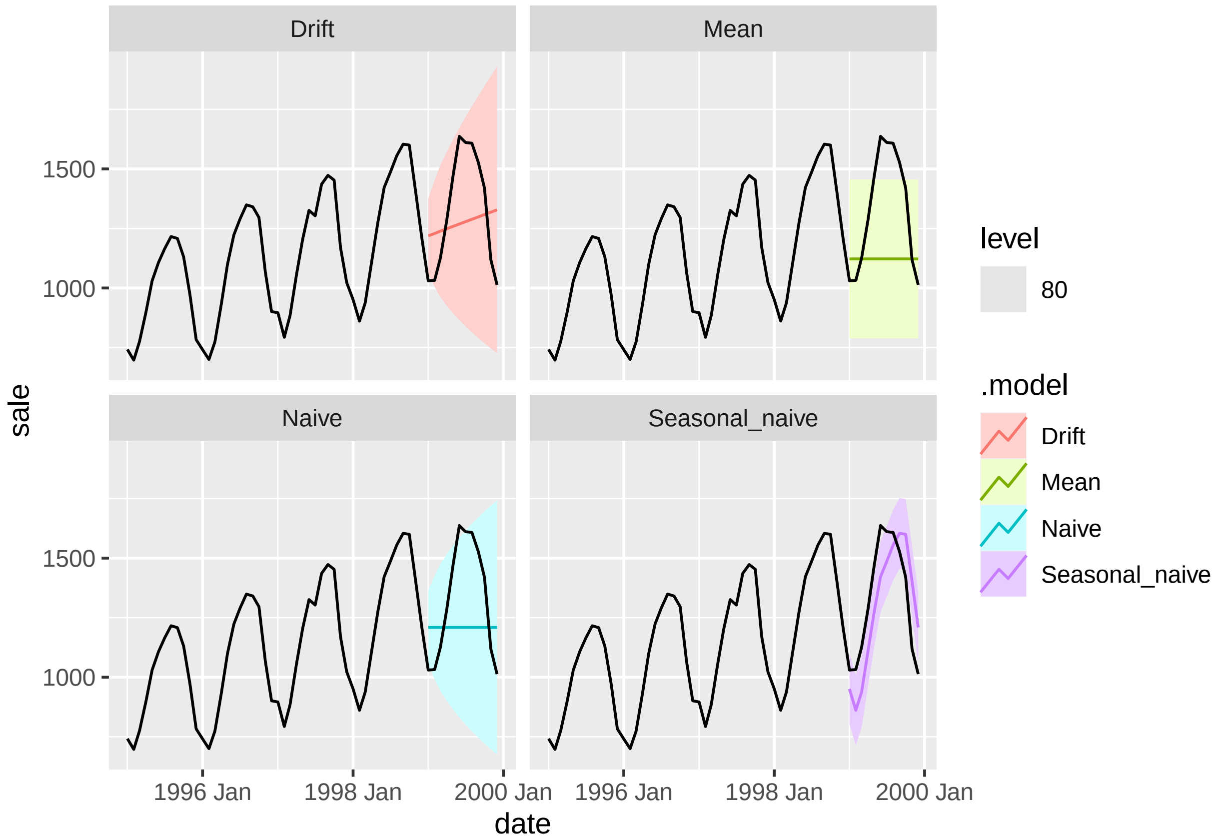



```
fc %>% autoplot(datats,level = 80, point_forecast = lst(mean, median))
```

```
## Warning in ggplot2::geom_point(mapping = mapping, data =
## dplyr::semi_join(object, : Ignoring unknown aesthetics: linetype
```



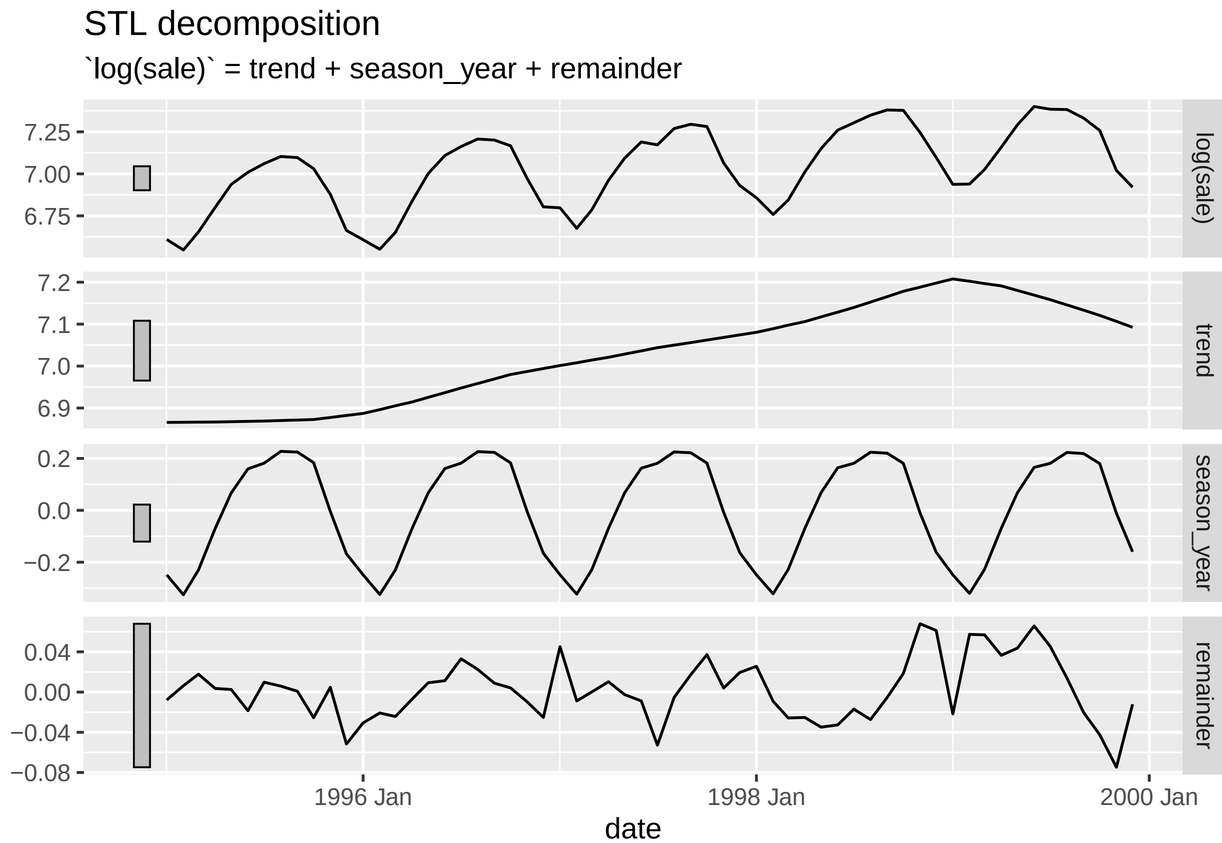

```
fc %>% autoplot(datats,level = 80) + facet_wrap(~.model)
```



It can be clearly seen that Seasonal naive model performance is the best. The data is seasonal and seasonal naive model can capture it.

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

```
data2 <- datats
data2 %>% model(STL(log(sale))) %>% components() %>% autoplot()
```




```r
fit <- data2 %>%
  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  4.51  40.1  30.6  0.336  2.56 0.261 0.303 0.356
## 2 ets      Training -1.15  39.1  28.7 -0.111  2.40 0.245 0.295 0.300
```

```r
report(fit)
```

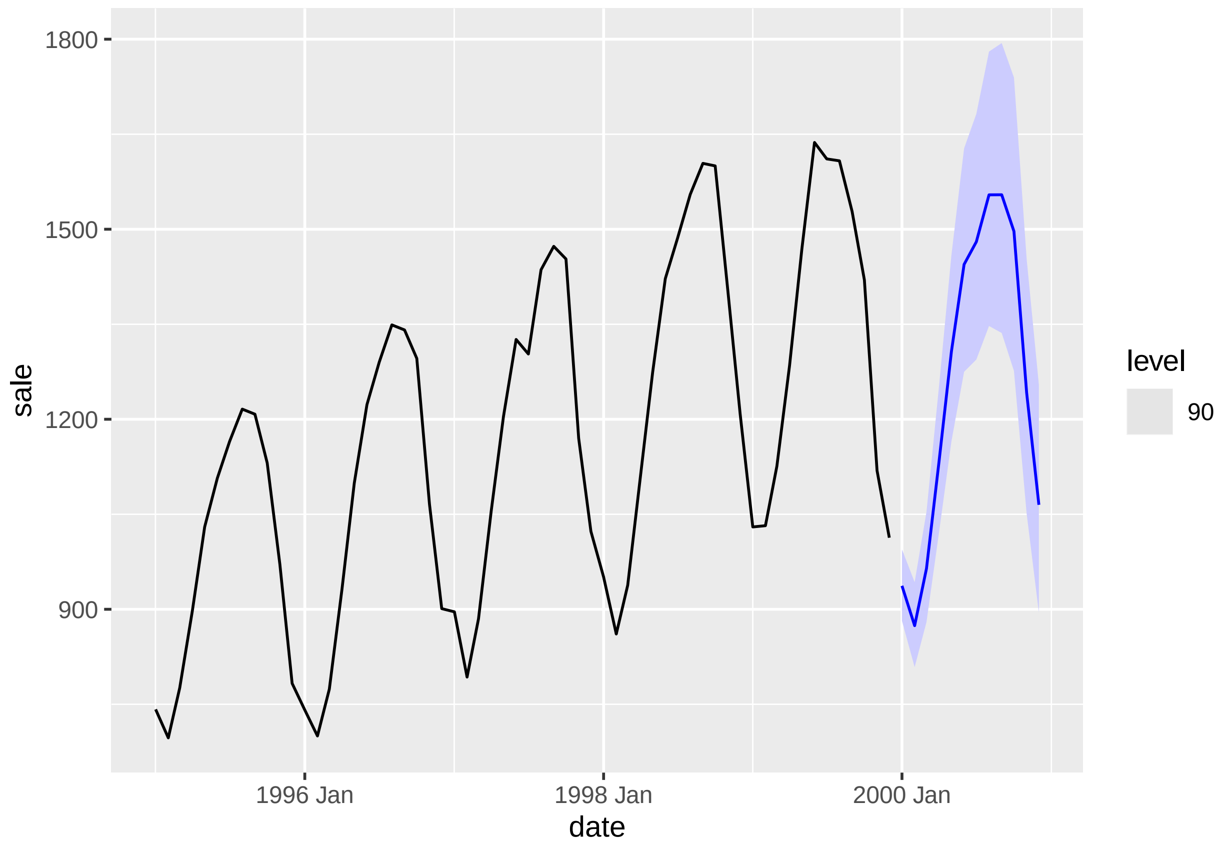
```
## Warning in report.mdl_df(fit): Model reporting is only supported for individual
## models, so a glance will be shown. To see the report for a specific model, use
## `select()` and `filter()` to identify a single model.

## # A tibble: 2 x 9
##   .model      sigma2 log_lik   AIC  AICc   BIC      MSE    AMSE     MAE
##   <chr>        <dbl>   <dbl> <dbl> <dbl> <dbl>    <dbl>   <dbl>   <dbl>
## 1 ets_auto 0.0000276    83.1 -136. -125. -105. 0.00105  0.00191 0.00364
## 2 ets      0.00134      84.9 -136. -121. -100. 0.000982 0.00179 0.0239
```

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

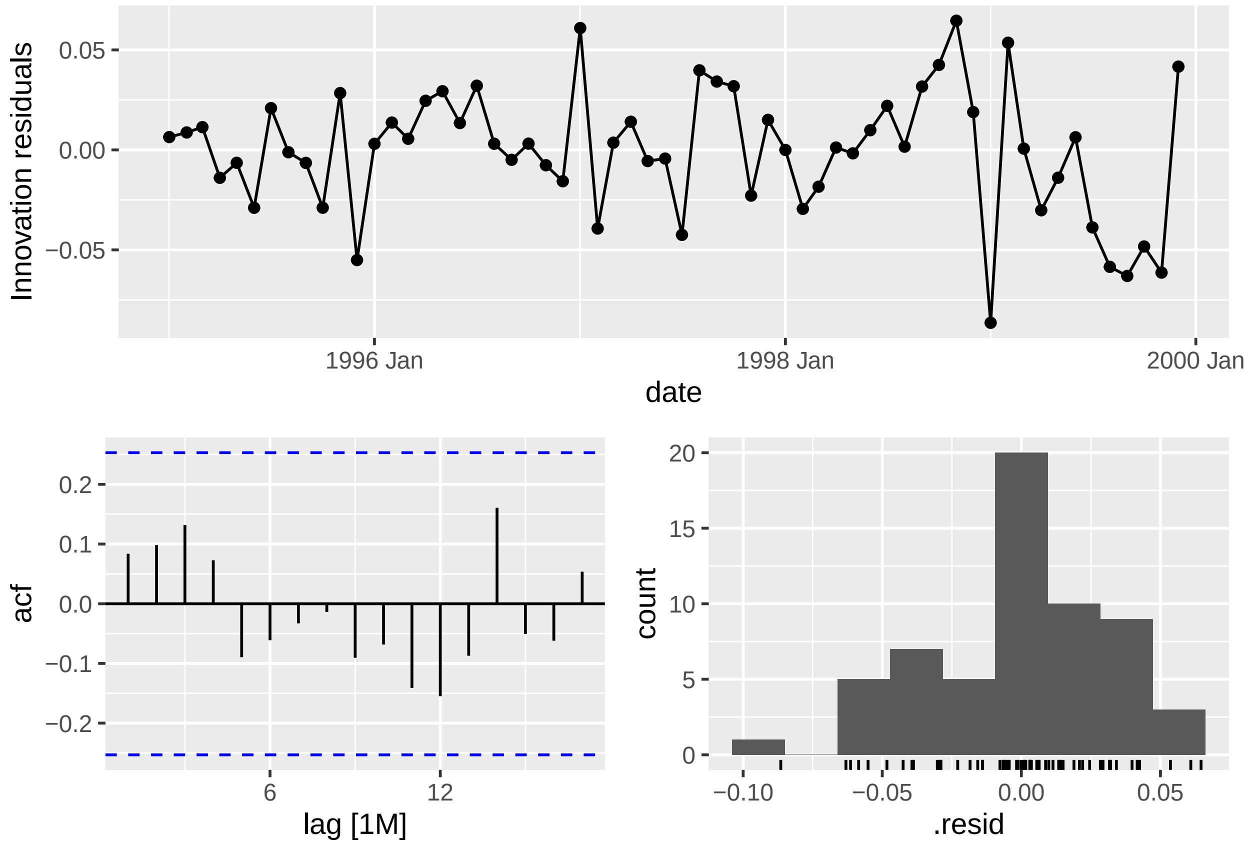
```
## Series: sale
```

```
## Model: ETS(M,N,A)
## Transformation: log(sale)
##   Smoothing parameters:
##     alpha = 0.7326626
##     gamma = 0.0001000079
##
##   Initial states:
##       l[0]       s[0]        s[-1]      s[-2]      s[-3]      s[-4]      s[-5]
##  6.904945 -0.1662107 -0.009129157 0.1830584 0.2227718 0.2262856 0.1827145
##       s[-6]      s[-7]       s[-8]      s[-9]     s[-10]     s[-11]
##  0.1625745 0.06895595 -0.06629676 -0.2260808 -0.3250813 -0.253562
##
##   sigma^2:  0
##
##       AIC      AICc       BIC
## -136.1679 -125.2588 -104.7527
```

```
fit <- fit %>% select(ets)
fc <- fit %>% forecast(h = "1 years")
fc %>% autoplot(datats,level = 90)
```




```
gg_tsresiduals(fit)
```



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

```
fit <- traindata %>%
  model(
    arima_auto = ARIMA(log(sale)),
    arima = ARIMA(log(sale)~0+pdq(3,0,3)+PDQ(1,1,0))
  )
```

```
## Warning in wrap_arima(y, order = c(p, d, q), seasonal = list(order = c(P, :
## possible convergence problem: optim gave code = 1
```

```
## Warning in sqrt(diag(best$var.coef)): NaNs produced
```

```
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 arima_auto Training  2.91  30.5  22.4 0.102  1.99 0.218 0.264 -0.0247
## 2 arima      Training 10.6   34.4  25.0 0.812  2.12 0.244 0.298 -0.00152
```

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

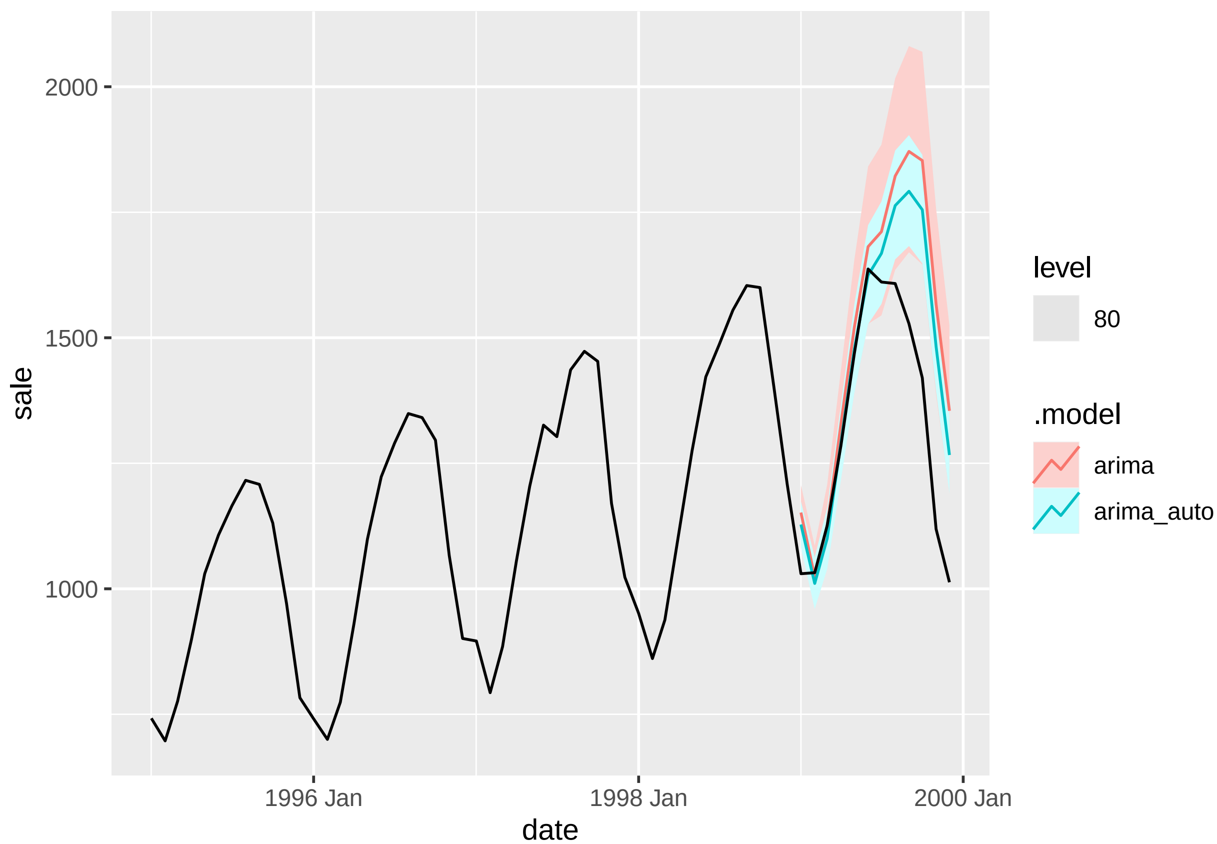
```
## Series: sale
## Model: ARIMA(1,0,0)(0,1,1)[12] w/ drift
## Transformation: log(sale)
##
## Coefficients:
```

```
##          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
```

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

```
## Series: sale
## Model: ARIMA(3,0,3)(1,1,0)[12]
## Transformation: log(sale)
##
## Coefficients:
##          ar1     ar2      ar3     ma1      ma2      ma3     sar1
##       0.4908  0.6272  -0.1394  0.2992  -0.1801  -0.0380  -0.3978
## s.e.     NaN  0.5023      NaN     NaN      NaN   0.2474   0.1801
##
## sigma^2 estimated as 0.00134:  log likelihood=69.86
## AIC=-123.71   AICc=-118.38   BIC=-111.04
```

```r
fc <- fit %>% forecast(h = "1 year")
fc %>% autoplot(datats,level = 80)
```

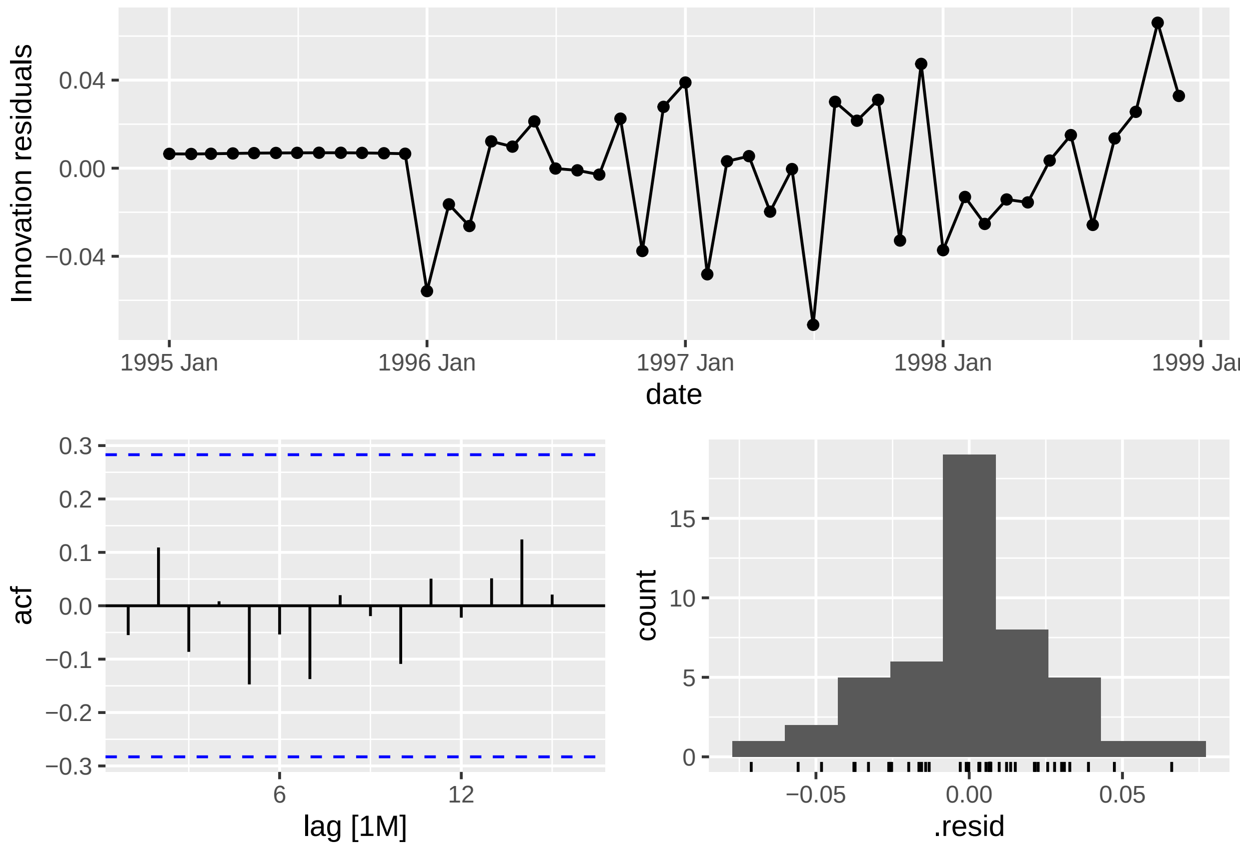



```r
accuracy(fc,datats)
```

```
## # 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      Test  -178.  241.  179. -13.9   14.0  1.75  2.09 0.842
## 2 arima_auto Test  -123.  186.  133.  -9.75  10.6  1.30  1.61 0.834
```

```r
gg_tsresiduals(fit %>% select(arima_auto))
```



1.12 Which model has best performance? (2 points)

By looking at the plots between the forecast and actual data, seasonal naive performed the best.

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)

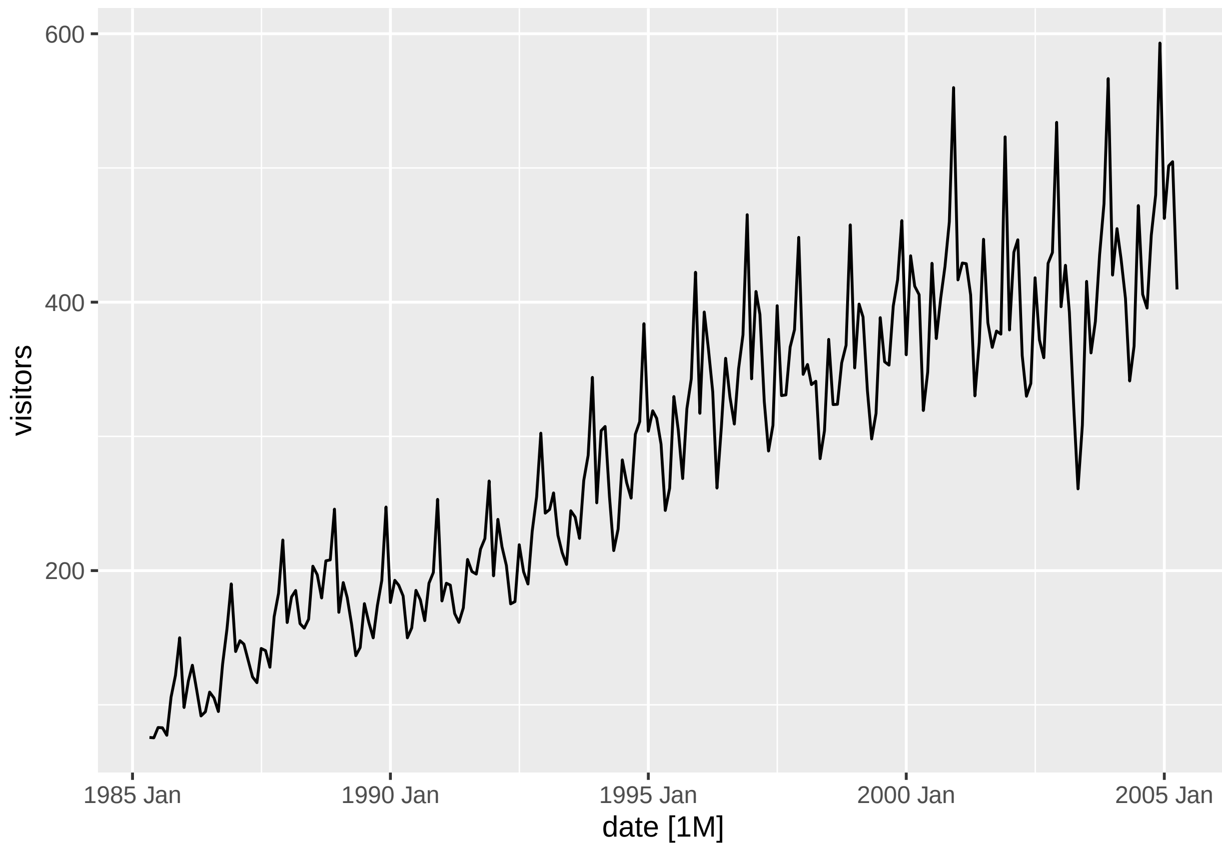
```r
#Reading the data file
df = read.csv("visitors.csv")
df %>%
  mutate(date=yearmonth(date)) %>%
  tsibble(index=date) ->
  df
head(df)
```

```
## # A tsibble: 6 x 2 [1M]
##       date visitors
##      <mth>    <dbl>
## 1 1985 May     75.7
## 2 1985 Jun     75.4
## 3 1985 Jul     83.1
```

```
## 4 1985 Aug     82.9
## 5 1985 Sep     77.3
## 6 1985 Oct    106.
```

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

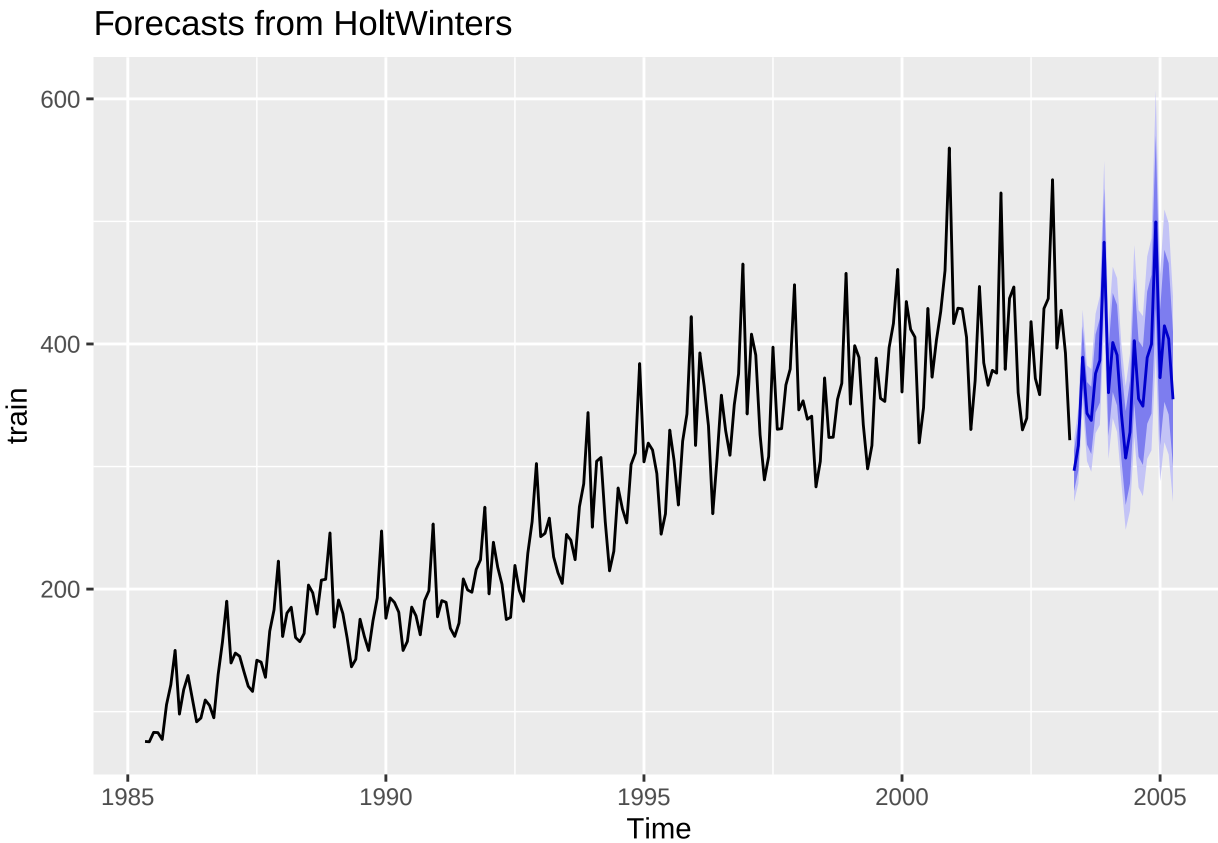
```r
#time plot of data
autoplot(df,visitors)
```



The time series of monthly Australian Overseas Visitors has a positive seasonal trend.Around 2003, there seems to be drop in number of visitors.

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)

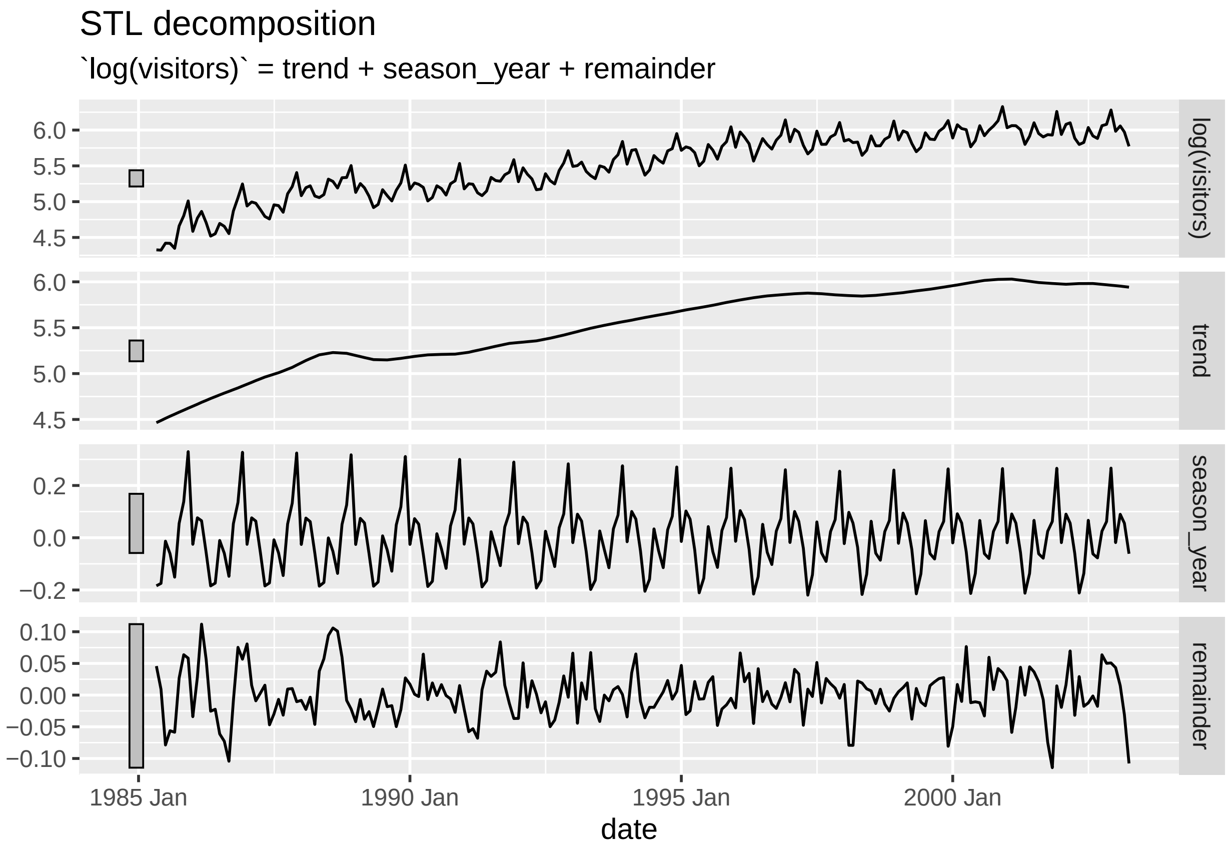
```r
train <- df %>%
  filter(date <= yearmonth("2003-04"))
test <- df %>%
  filter(date > yearmonth("2003-04"))

df1 <- HoltWinters(train, seasonal="multiplicative")
fc <- df1 %>% forecast::forecast(h = 24)
autoplot(fc)
```



2.3. Why is multiplicative seasonality necessary here? (6 points) Answer- The amplitude of the seasonal pattern in number of visitors increases as the level of time series increases, so multiplicative model is more appropriate choice.A multiplicative model would allow to capture this proportional relationship between the seasonal pattern and the level of the data, which could help understand the underlying trends and patterns in the number of visitors over time. 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;

```
# I. Forecast using ETS model
train %>% model(STL(log(visitors))) %>% components() %>% autoplot()
```




```
fit <- train %>%
  model(
    ets_auto = ETS(log(visitors)),
    ets = ETS(log(visitors) ~ 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  0.908  14.4  10.6  0.242  4.00 0.405 0.462 -0.0402
## 2 ets      Training -0.933  14.8  10.8 -0.279  4.07 0.412 0.476  0.0412
```

```
report(fit)
```

```
## Warning in report.mdl_df(fit): Model reporting is only supported for individual
## models, so a glance will be shown. To see the report for a specific model, use
## `select()` and `filter()` to identify a single model.
```

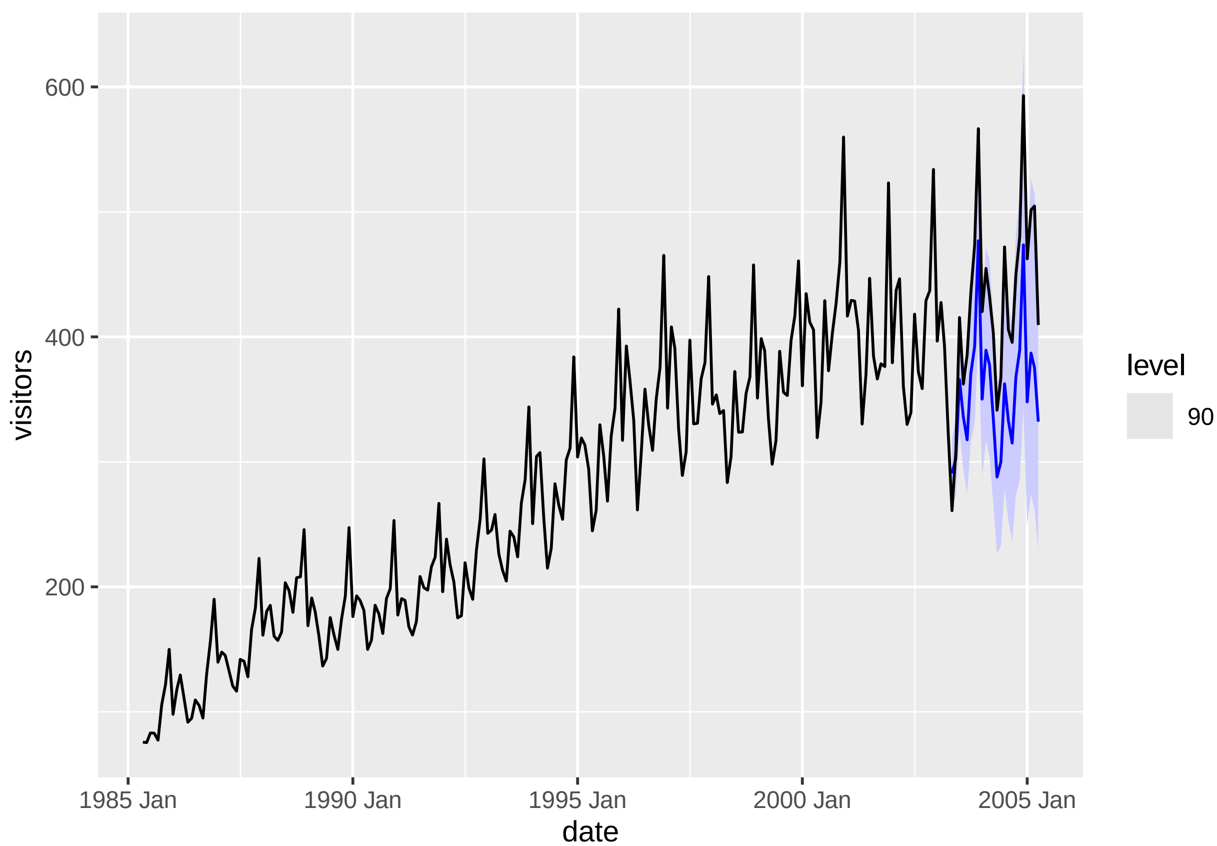
```
## # A tibble: 2 x 9
##   .model    sigma2 log_lik   AIC  AICc   BIC     MSE    AMSE    MAE
##   <chr>      <dbl>   <dbl> <dbl> <dbl> <dbl>   <dbl>   <dbl>  <dbl>
## 1 ets_auto 0.00288    60.0 -84.0 -80.5 -23.3 0.00266 0.00385 0.0402
## 2 ets      0.00301    54.8 -75.7 -72.6 -18.3 0.00279 0.00418 0.0407
```

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

```
## Series: visitors
```

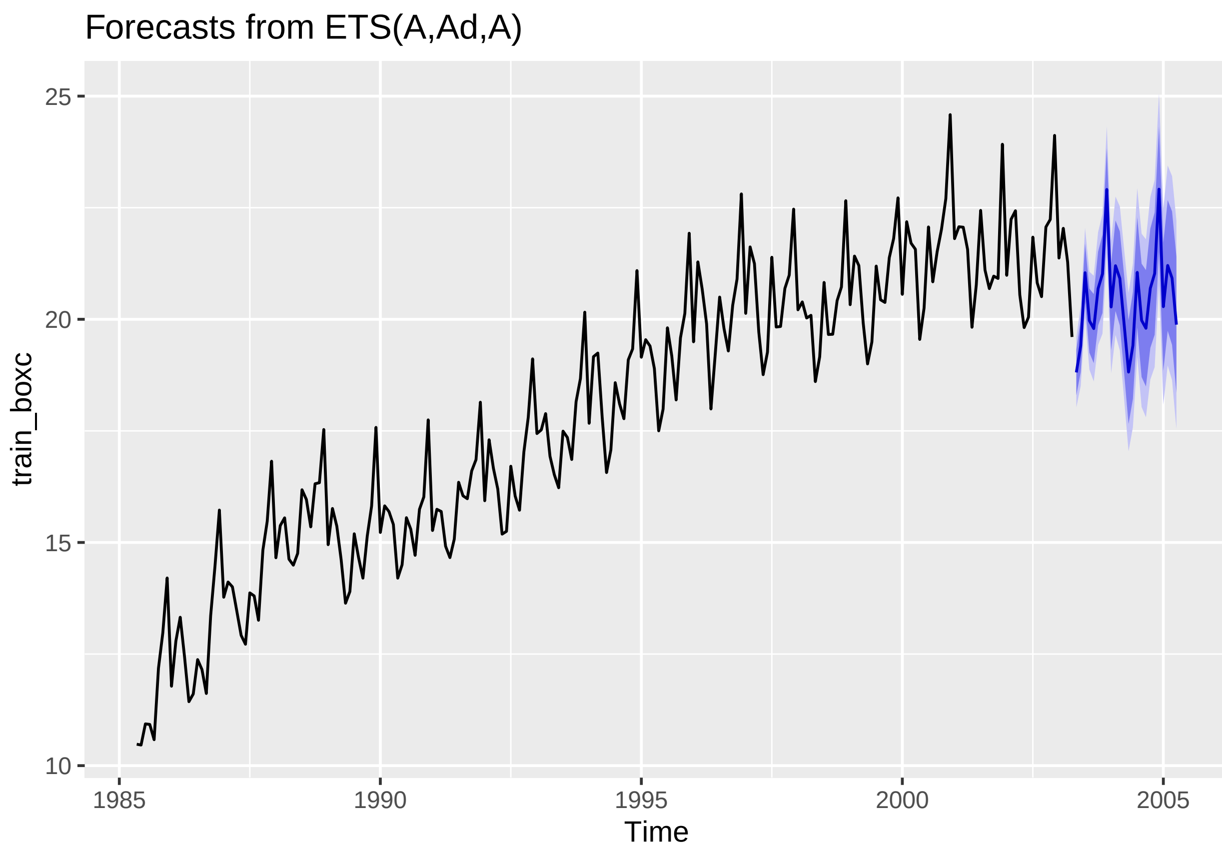
```
## Model: ETS(A,Ad,A)
## Transformation: log(visitors)
##   Smoothing parameters:
##     alpha = 0.6293372
##     beta  = 0.0006256646
##     gamma = 0.0001292349
##     phi   = 0.979999
##
##   Initial states:
##       l[0]       b[0]        s[0]       s[-1]      s[-2]      s[-3]      s[-4]
## 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

```r
fit <- fit %>% select(ets)
fc <- fit %>% forecast(h = 24)
fc %>% autoplot(df,level = 90)
```




```r
# Calculate accuracy of the forecast
RMSE_ets =accuracy(fc, test)[,"RMSE"]
```

```
# II. Forecast using Box-Cox transformed additive ETS model

tdata <- ts(train$visitors, start = c(1985, 5), frequency = 12)

# Apply the Box-Cox transformation to the training data
lambda <- BoxCox.lambda(tdata)
train_boxc <- BoxCox(tdata, lambda)

# Fit an additive ETS model to the transformed training data
fit <- ets(train_boxc, model = "AAA")
resid <- residuals(fit)
# Forecast the next 8 observations (i.e., the two-year test set)
pred <- forecast(fit, h = 24)
# Inverse transform the forecasts using the inverse Box-Cox transformation
pred_inv <- InvBoxCox(pred$mean, lambda)
autoplot(pred)
```




```
# Print the forecasts
pred_inv
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2003                                     290.9802 313.7044 381.8695 336.1874
## 2004 348.8481 388.8374 376.2235 332.3026 291.2614 313.9935 382.1907 336.4776
## 2005 349.1167 389.1195 376.4941 332.5477
##           Sep      Oct      Nov      Dec
## 2003 329.0056 366.3147 380.7326 470.2128
```

```
## 2004 329.2861 366.6091 381.0283 470.5443
## 2005
```

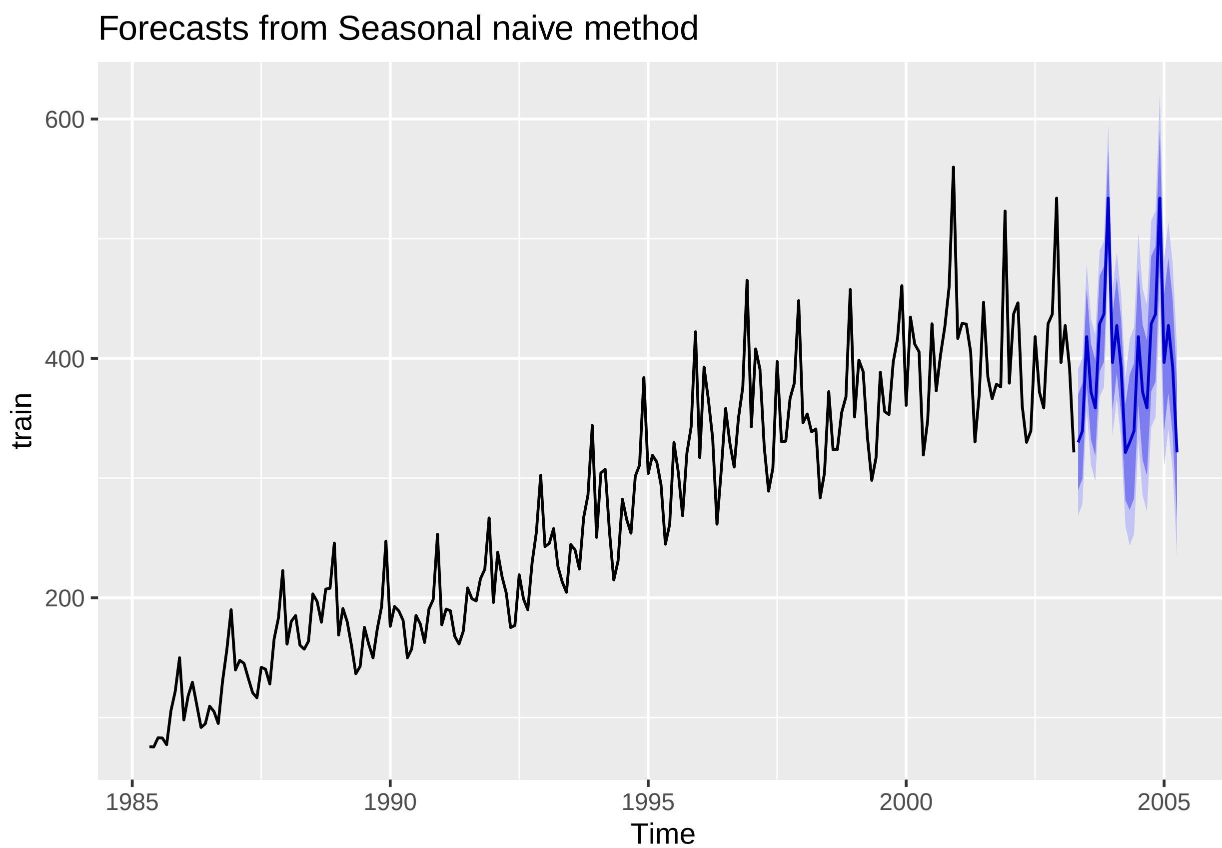
```
#Accuracy calculation for Box-Cox transformed additive ETS model
tedata <- ts(test$visitors, start = c(2003, 5), frequency = 12)
tedata
```

```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
## 2003                         260.9 308.3 415.5 362.2 385.6 435.3 473.3 566.6
## 2004 420.2 454.8 432.3 402.8 341.3 367.3 472.0 405.8 395.6 449.9 479.9 593.1
## 2005 462.4 501.6 504.7 409.5
```

```
# Calculate the RMSE
rmseETSbox <- sqrt(mean((pred_inv - tedata)^2))
rmseETSbox
```

```
## [1] 78.61032
```

```
# seasonal naïve method
fc3 <- snaive(train, h = 24)
autoplot(fc3)
```




```
# Accuracy calculation for seasonal_naïve_method

# Convert forecast and test sets to forecast time series class
fc3_ts <- ts(fc3$mean, start = c(2003, 5), frequency = 12)
test_ts <- ts(test$visitors, start = c(2003, 5), frequency = 12)
# Calculate the RMSE
rmse_seasonal_naïve_method <- sqrt(mean((fc3_ts - test_ts)^2))
```

```r
rmse_seasonal_naïve_method
```

```
## [1] 50.30097
```

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

```r
#ETS model- Accuracy-RMSE
RMSE_ets
```

```
## # A tibble: 1 x 1
##    RMSE
##   <dbl>
## 1  80.0
```

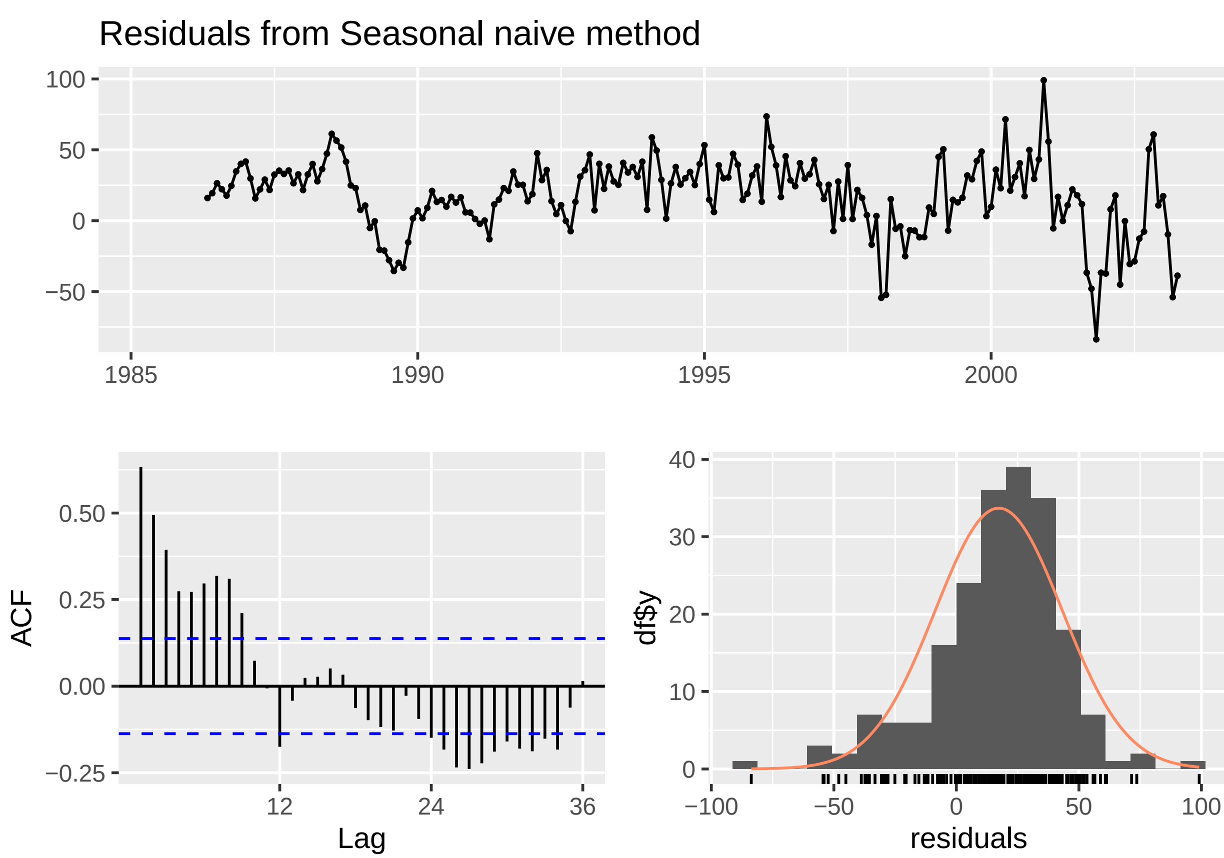
```r
#Box-Cox transformed additive ETS model- Accuracy-RMSE
rmseETSbox
```

```
## [1] 78.61032
```

```r
#Seasonal_naïve_method- Accuracy- RMSE
rmse_seasonal_naïve_method
```

```
## [1] 50.30097
```

```r
#Print Residual of Seasonal_naïve_method
print(checkresiduals(fc3))
```




```
##
##  Ljung-Box test
##
```

```
## data:  Residuals from Seasonal naive method
## Q* = 295.02, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
##
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 295.02, df = 24, p-value < 2.2e-16
```

It can be seen that the order of RMSE value is as follows: seasonal naïve method > additive ETS with BoxCox transformation > an ETS model

Seasonal naïve method gives the best performance and pass the residual test.

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)

```r
# Loading libraries
library(ggplot2)
library(zoo)
```

```
##
## Attaching package: 'zoo'

## The following object is masked from 'package:tsibble':
##
##     index

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(nortest)

library(urca)
library(forecast)
```

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

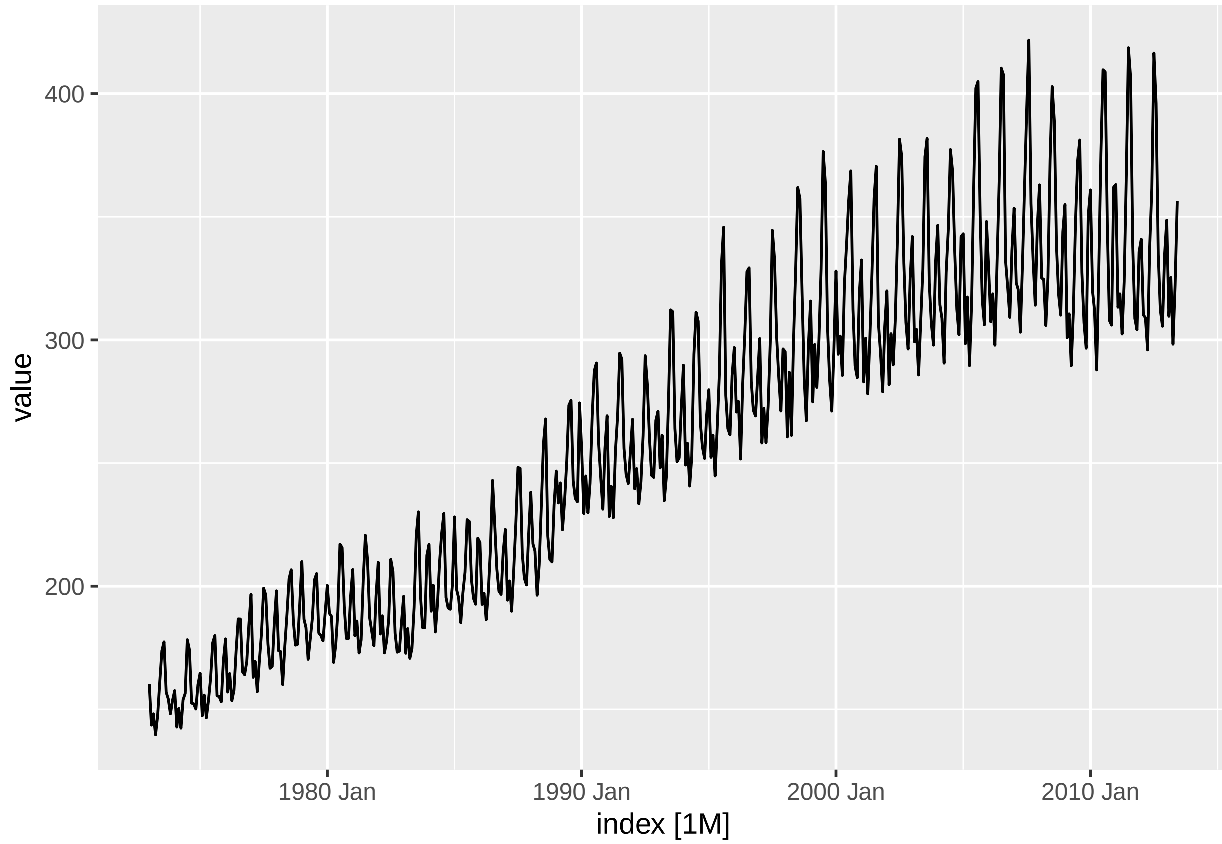
```r
usmelec <- readr::read_csv("usmelec.csv",show_col_types = FALSE)
usmelec %>%
  mutate(index=yearmonth(index)) %>%
  tsibble(index=index) ->
  usmelec
head(usmelec)
```

```
## # 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.
```

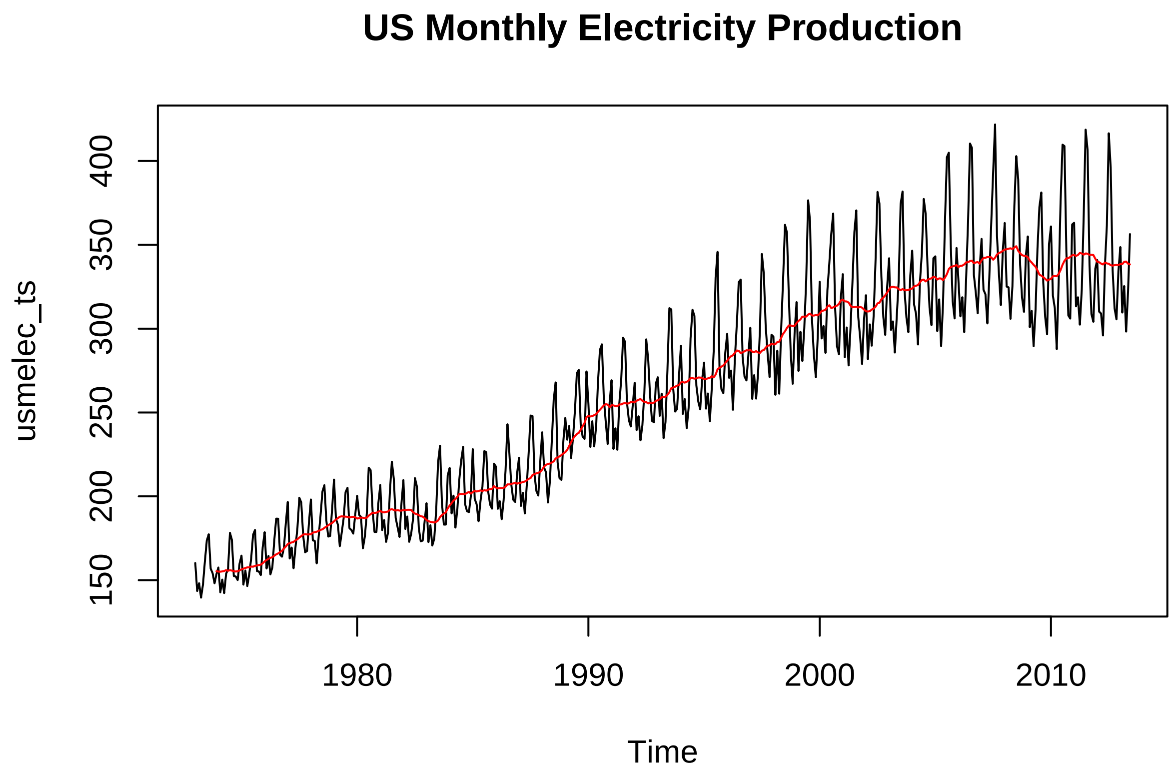
```
## 5 1973 May  147.
## 6 1973 Jun  161.
```

```
autoplot(usmelec, ylab = "Generation",
    xlab = "January 1973 - June 2013",
    ggtitle = "total net generation of electricity")
```

```
## Plot variable not specified, automatically selected `.vars = value`
## Warning in geom_line(...): Ignoring unknown parameters: `ylab`, `xlab`, and
## `ggtitle`
```




```
usmelec_ts <- ts(usmelec$value, frequency = 12, start = c(1973, 1))
usmelec_ma <- rollmean(usmelec_ts, k = 12, align = "right")

plot(usmelec_ts, main = "US Monthly Electricity Production")
lines(usmelec_ma, col = "red")
```



The 12-month moving average shows dip in the early-mid 1980s and after that trend is linearly increasing till around 2010 where it dips and flattens till year 2013.

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

```r
library(fpp2)
```

```
## -- Attaching packages ---------------------------------------------- fpp2 2.5 --

## v fma       2.5     v expsmooth 2.3

##

##
## Attaching package: 'fpp2'

## The following object is masked _by_ '.GlobalEnv':
##
##     usmelec
```

```r
library(urca)
library(forecast)
library(fBasics)
```

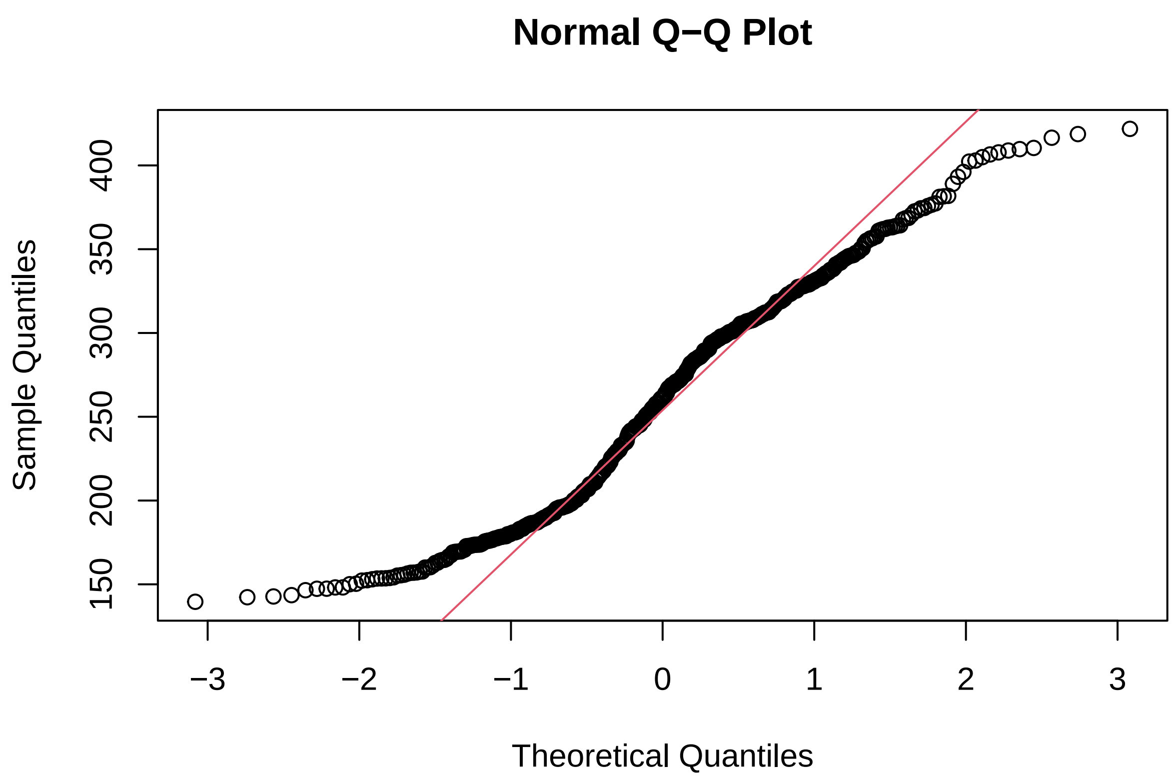
```
## Warning: package 'fBasics' was built under R version 4.2.3
```

```r
normalTest(usmelec$value, method = c("jb"))
```

```
##
## Title:
##  Jarque - Bera Normalality Test
```

```
##
## Test Results:
##   STATISTIC:
##     X-squared: 22.0343
##   P VALUE:
##     Asymptotic p Value: 1.642e-05
```

```
qqnorm(usmelec$value)
qqline(usmelec$value, col = 2)
```




```
skewness(usmelec$value)
```

```
## [1] 0.1444914
## attr(,"method")
## [1] "moment"
```

```
kurtosis(usmelec$value)
```

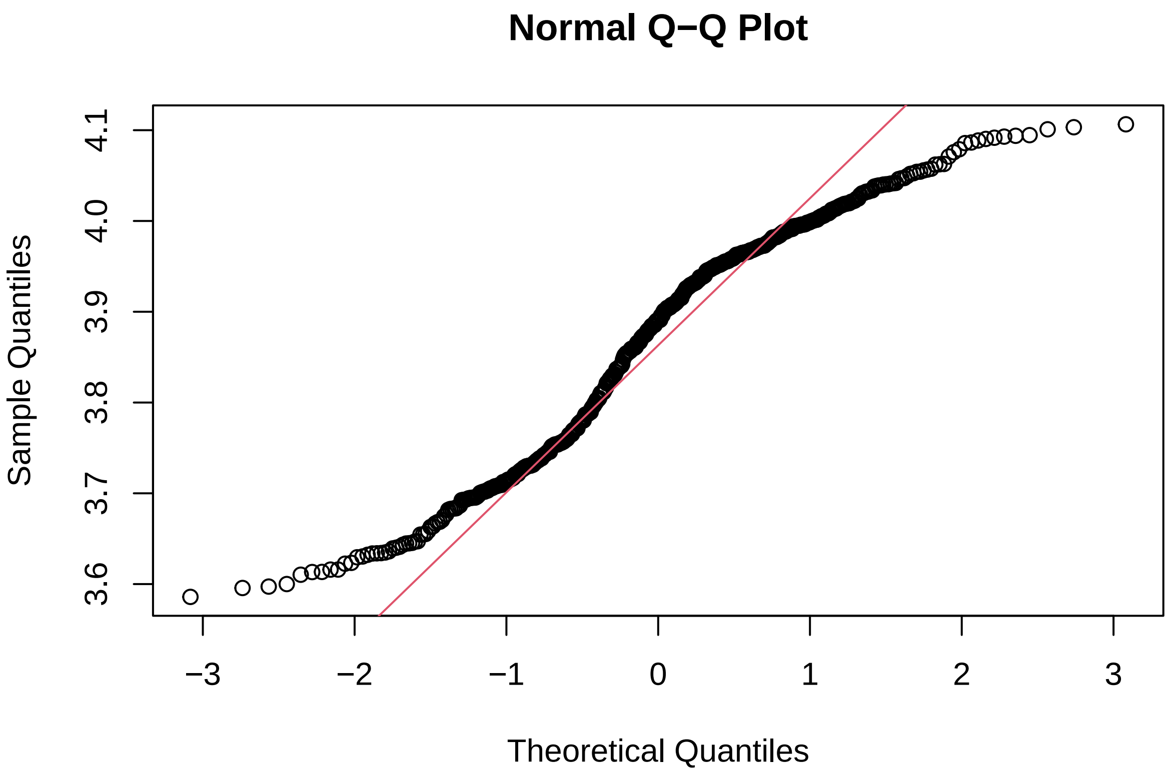
```
## [1] -1.010254
## attr(,"method")
## [1] "excess"
```

```
## apply Box-Cox transform with - lambda ='auto'
usmelec_box <- BoxCox(usmelec$value, lambda = "auto")
normalTest(usmelec_box, method = c("jb"))
```

```
##
## Title:
##  Jarque - Bera Normalality Test
```

```
##
## Test Results:
##   STATISTIC:
##     X-squared: 29.0037
##   P VALUE:
##     Asymptotic p Value: 5.034e-07
```

```r
qqnorm(usmelec_box)
qqline(usmelec_box, col = 2)
```




```r
skewness(usmelec_box)
```

```
## [1] -0.2692436
## attr(,"method")
## [1] "moment"
```

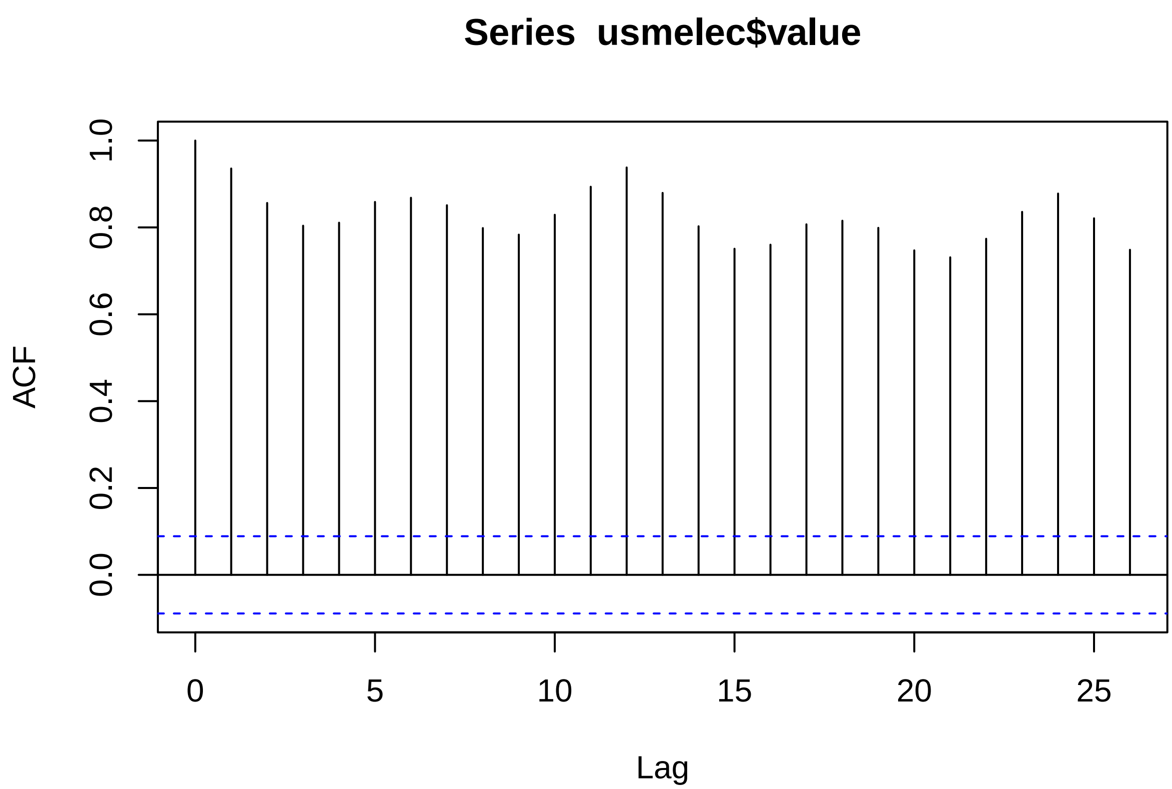
```r
kurtosis(usmelec_box)
```

```
## [1] -1.075892
## attr(,"method")
## [1] "excess"
```

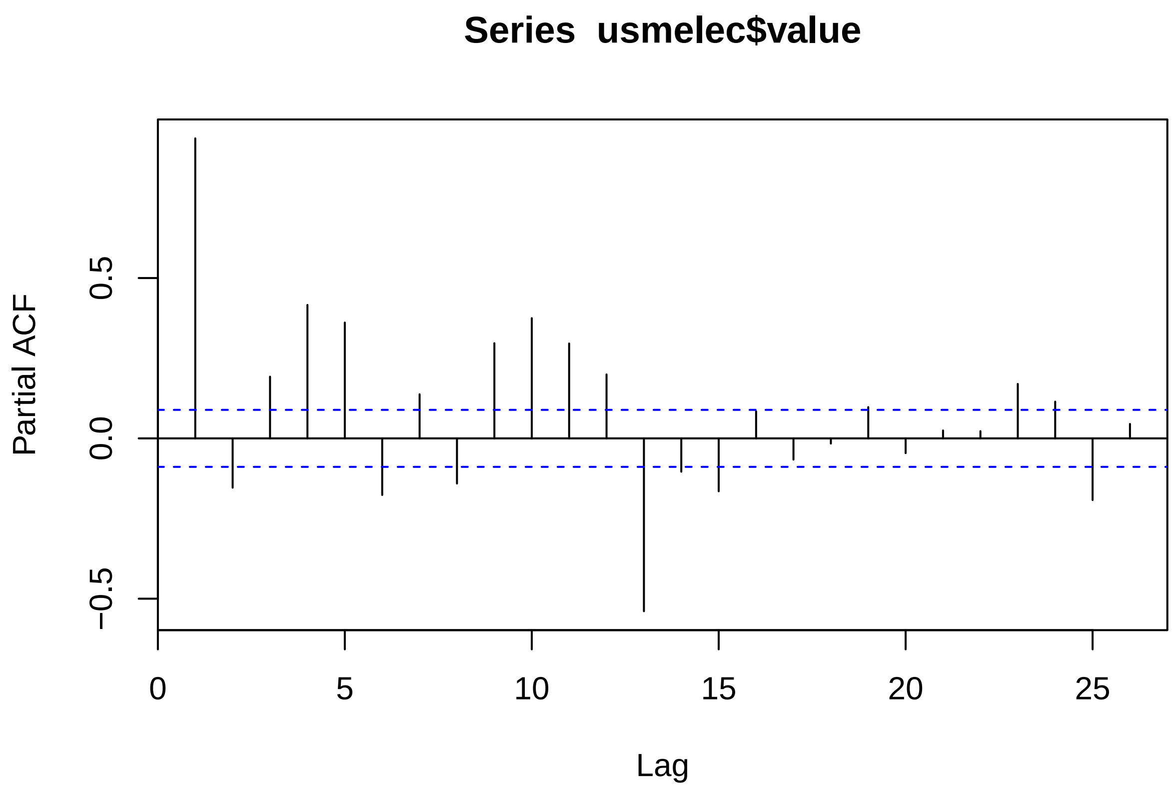
The Jarque-Bera (JB) test tells the data appears to follow a normal distribution. The test resulted in a statistic of X-squared: 22.0343 with an asymptotic p-value of 1.642e-05, and another test resulted in a statistic of X-squared: 29.0037 with an asymptotic p-value of 5.034e-07. The Q-Q plot shows a tight fit to the line, indicating normality. The skewness of the data is slightly right-skewed, with a value of 0.14, and the kurtosis is -1.01, which indicates less peakedness than a normal distribution or possibly less extreme outliers. Box-Cox and log transformations did not improve the kurtosis or skewness, so the non-transformed data will be used.

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

```r
acf(usmelec$value)
```




```r
pacf(usmelec$value)
```




```r
## check number of differences
ndiffs(usmelec$value, alpha = 0.05)
```

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

ACF plot doesn't show rapid decay in correlation and 1st diff lagged correlation of ACF plot and PACF appears to have significant residual variation. ndiff() gives 1, that tells taking first difference of time series is likely sufficient to make the series stationary.

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)

```r
# ARIMA #1
ARIMA1 <- Arima(usmelec_ts, order = c(10, 1, 0), seasonal = c(1,
    0, 0), lambda = 0)
ARIMA1
```

```
## Series: usmelec_ts
## ARIMA(10,1,0)(1,0,0)[12]
## Box Cox transformation: lambda= 0
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
##       -0.4508  -0.4600  -0.2608  -0.3087  -0.2334  -0.1232  -0.0976  -0.0725
## s.e.   0.0456   0.0497   0.0546   0.0565   0.0568   0.0562   0.0548   0.0547
##           ar9    ar10    sar1
##       -0.0740  0.0907  0.9204
## s.e.   0.0519  0.0482  0.0186
```

```
##
## sigma^2 = 0.001067:  log likelihood = 965.46
## AIC=-1906.92   AICc=-1906.26   BIC=-1856.71
```

```r
# ARIMA #2
ARIMA2 <- Arima(usmelec_ts, order = c(4, 1, 0), seasonal = c(4,
    1, 0), lambda = 0)
ARIMA2
```

```
## Series: usmelec_ts
## ARIMA(4,1,0)(4,1,0)[12]
## Box Cox transformation: lambda= 0
##
## Coefficients:
##           ar1      ar2      ar3      ar4     sar1     sar2     sar3     sar4
##       -0.4027  -0.3518  -0.1597  -0.1422  -0.7106  -0.5682  -0.4116  -0.1606
## s.e.   0.0458   0.0492   0.0489   0.0458   0.0471   0.0558   0.0548   0.0474
##
## sigma^2 = 0.0007652:  log likelihood = 1025.02
## AIC=-2032.04   AICc=-2031.65   BIC=-1994.61
```

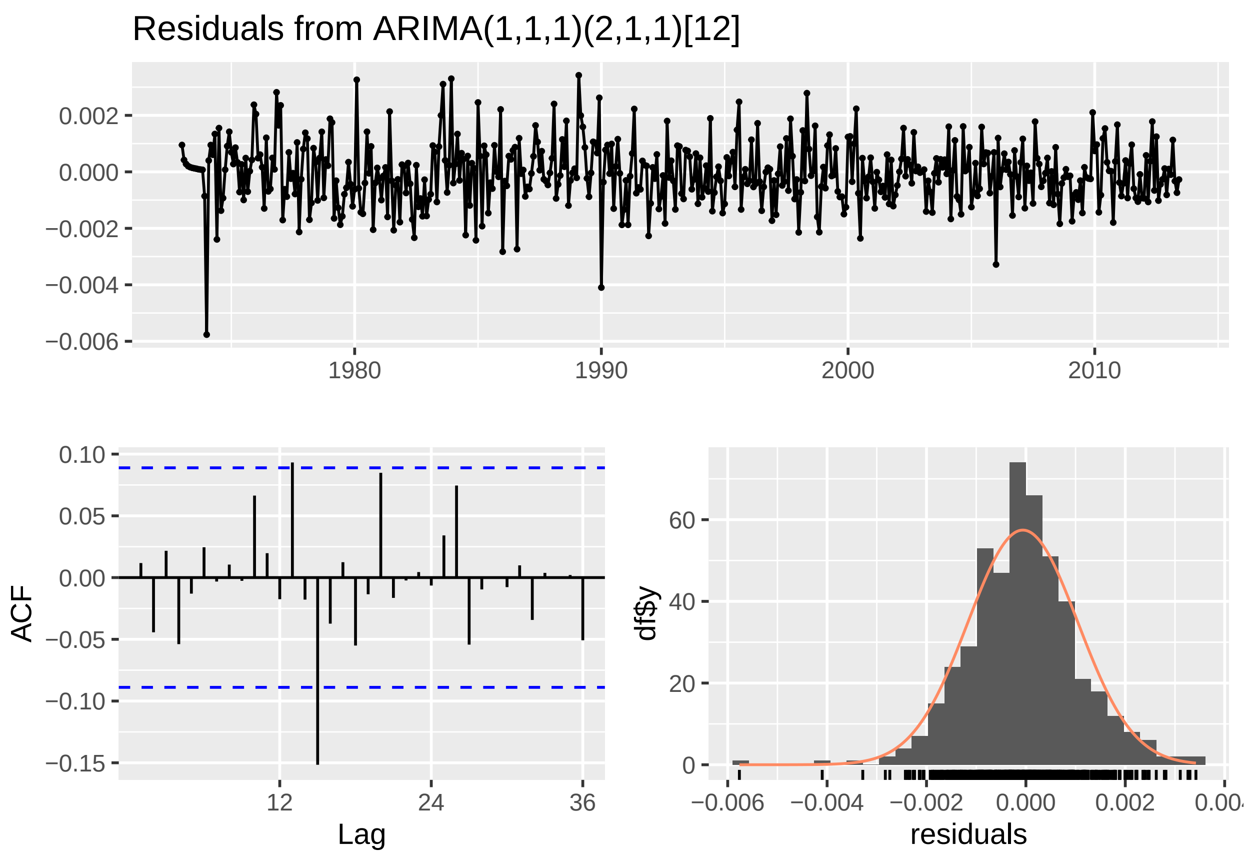
```r
# ARIMA #3
ARIMA3 <- auto.arima(usmelec_ts, seasonal = TRUE, stepwise = FALSE,
    approximation = FALSE, lambda = "auto")
ARIMA3
```

```
## Series: usmelec_ts
## ARIMA(1,1,1)(2,1,1)[12]
## Box Cox transformation: lambda= -0.5738168
##
## 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 = 1.274e-06:  log likelihood = 2547.32
## AIC=-5082.63   AICc=-5082.45   BIC=-5057.68
```

The best ARIMA model is the ARIMA3 model with AICc of -5842.31 which has values (ARIMA(1,1,1)(2,1,1)[12]) and Box Cox transformation: lambda= -0.4960396

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)

```r
checkresiduals(ARIMA3)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(2,1,1)[12]
## Q* = 28.013, df = 19, p-value = 0.08318
##
## Model df: 5.   Total lags used: 24
```

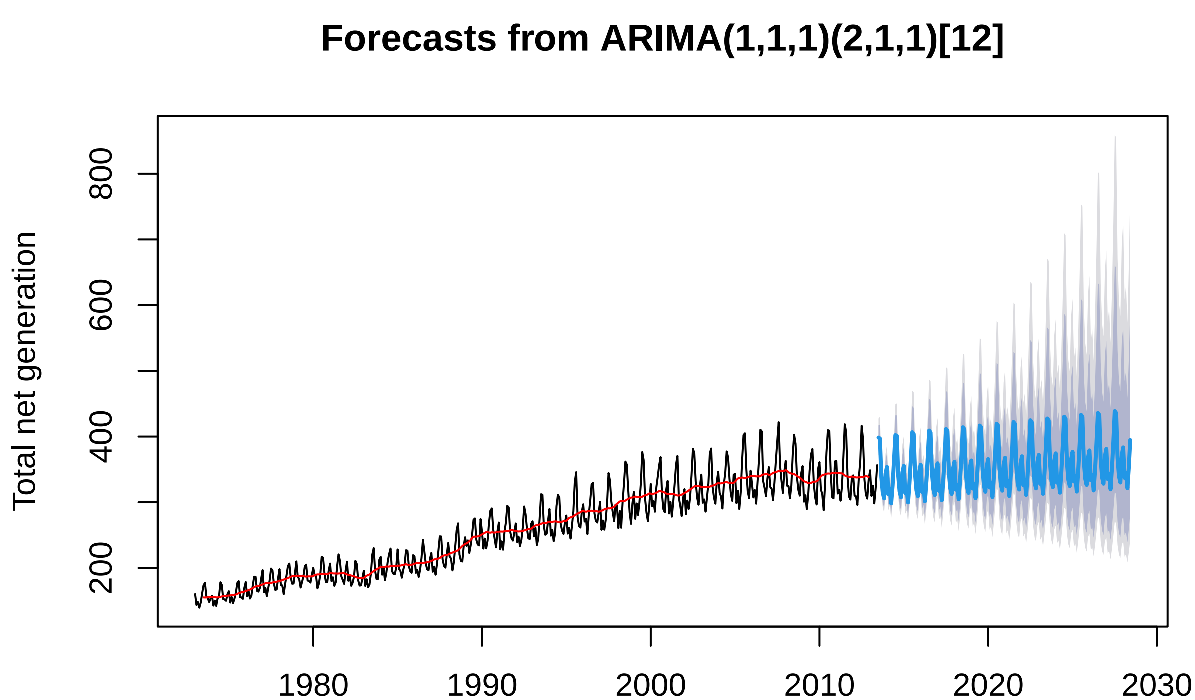
```
summary(ARIMA3)
```

```
## Series: usmelec_ts
## ARIMA(1,1,1)(2,1,1)[12]
## Box Cox transformation: lambda= -0.5738168
##
## 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 = 1.274e-06:  log likelihood = 2547.32
## AIC=-5082.63   AICc=-5082.45   BIC=-5057.68
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE  MAPE      MASE
## Training set -0.5001923 7.235643 5.302234 -0.1959052 2.002 0.5888268
##                     ACF1
## Training set -0.02860282
```

The Ljung-Box test was performed on the residuals of the model to check if they resemble white noise, and the test statistic is Q* = 26.322 with degrees of freedom (df) = 19 and a p-value of 0.1215. Since the p-value is greater than the significance level of 0.05, we fail to reject the null hypothesis that the residuals are white noise. This indicates that the ARIMA(1,1,1)(2,1,1)[12] model fits the data.The diagnostics training set errorand ACF plotsuggest that ARIMA3 model is a good fit for data, and the residuals resemble white noise.

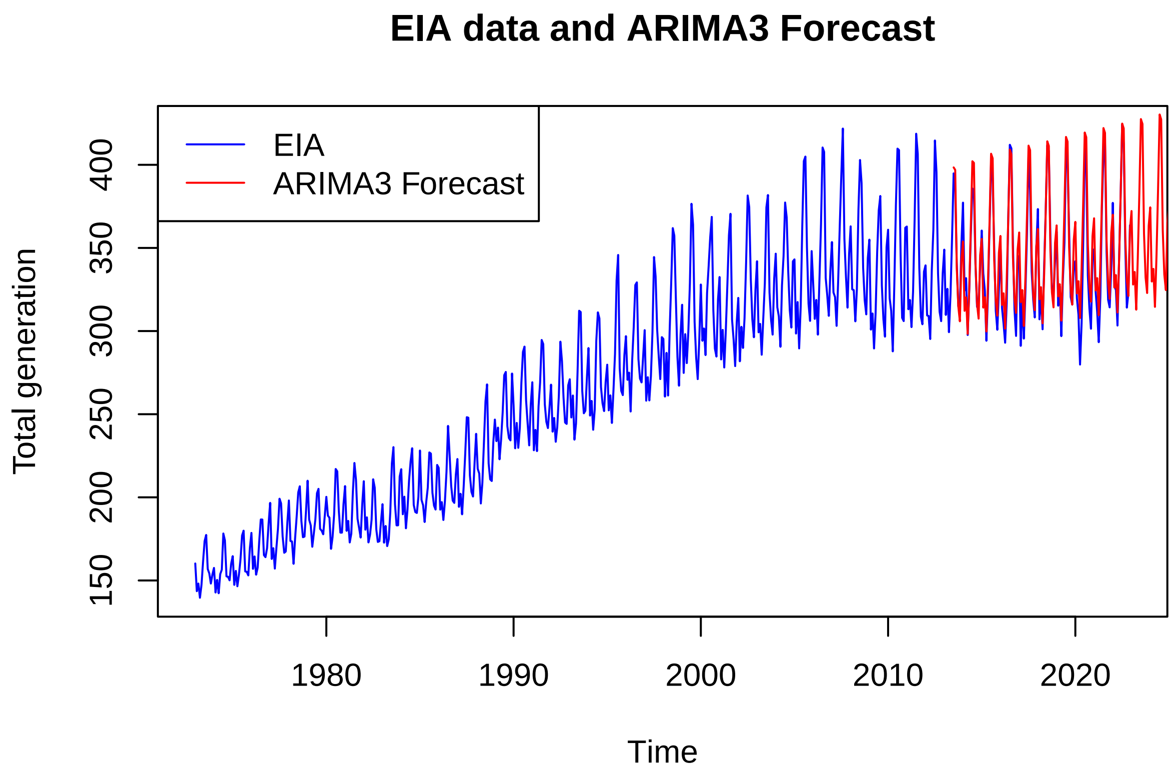
3.6 Forecast the next 15 years of electricity generation by the U.S. electric industry. Get the latest figures from the EIA (https://www.eia.gov/totalenergy/data/monthly/#electricity) to check the accuracy of your forecasts. (8 points)

```
plot(forecast(ARIMA3, h = 180), ylab = "Total net generation")
lines(ma(usmelec, 12), col = "red")
```




```
library(forecast)
library(tsibble)
library(dplyr)

# Read in the eia data
eia <- readr::read_csv("latest_data_eia.csv", show_col_types = FALSE)
eia %>%
  mutate(index=yearmonth(index)) %>%
  tsibble(index=index) ->
  eia

# Convert eia data to a time series
eia_ts <- ts(eia$value, frequency = 12, start = c(1973, 1))
```

```r
# ARIMA3 forecast
ARIMA3_forecast <- forecast(ARIMA3, h = 180)

# Plot the time series and forecast on the same plot
plot(eia_ts, main = "EIA data and ARIMA3 Forecast", col = "blue", ylab = "Total generation")
lines(ARIMA3_forecast$mean, col = "red")

# Add a legend
legend("topleft", legend = c("EIA", "ARIMA3 Forecast"), col = c("blue", "red"), lty = 1)
```




```r
# Extract ARIMA3 forecasted data
forecast <- window(ARIMA3_forecast$mean, start = c(2015, 1), end = c(2022, 12))

# Extract eia data
actual <- window(eia_ts, start = c(2015, 1), end = c(2022, 12))

# Calculate accuracy of ARIMA3 model
MAE <- mean(abs(forecast - actual))
MSE <- mean((forecast - actual)^2)
RMSE <- sqrt(MSE)

# Generate a sequence of dates from January 2015 to December 2022
dates <- seq(as.Date("2015-01-01"), as.Date("2022-12-01"), by = "month")


# Calculate percentage difference between forecast and actual values
```

```r
percentage_diff <- (forecast - actual) / actual * 100

# Create a data frame for percentage accuracy or difference
accuracy_df <- data.frame(date = dates,
                          actual = as.numeric(actual),
                          forecast = as.numeric(forecast),
                          percentage_diff = percentage_diff)

# Print percentage accuracy or difference for each month
print(accuracy_df)
```

```
##          date  actual forecast percentage_diff
## 1  2015-01-01 360.453 355.4698     -1.38247915
## 2  2015-02-01 334.596 314.1102     -6.12256187
## 3  2015-03-01 324.314 320.3750     -1.21455303
## 4  2015-04-01 294.177 299.7364      1.88981000
## 5  2015-05-01 322.189 328.8224      2.05884632
## 6  2015-06-01 362.493 365.4646      0.81976036
## 7  2015-07-01 400.535 406.6593      1.52904230
## 8  2015-08-01 392.242 403.9030      2.97289691
## 9  2015-09-01 350.190 342.6443     -2.15474715
## 10 2015-10-01 312.210 317.9797      1.84802310
## 11 2015-11-01 300.779 309.2488      2.81593924
## 12 2015-12-01 324.536 347.0240      6.92927513
## 13 2016-01-01 352.714 357.1958      1.27065932
## 14 2016-02-01 313.816 315.7260      0.60862855
## 15 2016-03-01 304.427 322.7060      6.00437879
## 16 2016-04-01 292.987 301.4523      2.88931481
## 17 2016-05-01 316.867 330.5706      4.32470155
## 18 2016-06-01 367.904 367.2575     -0.17572987
## 19 2016-07-01 412.043 409.1665     -0.69811816
## 20 2016-08-01 409.861 406.2566     -0.87942624
## 21 2016-09-01 351.560 344.5103     -2.00527167
## 22 2016-10-01 312.940 319.7116      2.16385528
## 23 2016-11-01 297.065 310.9091      4.66028712
## 24 2016-12-01 345.389 348.9196      1.02222062
## 25 2017-01-01 344.414 359.3015      4.32257084
## 26 2017-02-01 291.113 317.4070      9.03224838
## 27 2017-03-01 319.469 324.6643      1.62621755
## 28 2017-04-01 295.462 303.0602      2.57164317
## 29 2017-05-01 323.494 332.2390      2.70330928
## 30 2017-06-01 358.630 369.3269      2.98270505
## 31 2017-07-01 404.537 411.5394      1.73097109
## 32 2017-08-01 384.837 408.7703      6.21907807
## 33 2017-09-01 336.004 346.4803      3.11789648
## 34 2017-10-01 318.731 321.4637      0.85737225
## 35 2017-11-01 308.189 312.5652      1.41997820
## 36 2017-12-01 350.563 350.9814      0.11933885
## 37 2018-01-01 373.379 361.4308     -3.20002218
## 38 2018-02-01 307.058 319.1371      3.93382138
## 39 2018-03-01 321.765 326.4011      1.44082615
## 40 2018-04-01 301.057 304.6495      1.19329911
## 41 2018-05-01 339.228 334.0915     -1.51417591
## 42 2018-06-01 372.145 371.5462     -0.16089894
```

```
## 43 2018-07-01 411.617 414.1307      0.61069225
## 44 2018-08-01 408.352 411.3534      0.73499256
## 45 2018-09-01 356.558 348.4670     -2.26918268
## 46 2018-10-01 325.070 323.2220     -0.56848153
## 47 2018-11-01 322.466 314.2463     -2.54900520
## 48 2018-12-01 342.292 353.0119      3.13179438
## 49 2019-01-01 359.729 363.5443      1.06060967
## 50 2019-02-01 315.282 320.8784      1.77504542
## 51 2019-03-01 326.903 328.1817      0.39115119
## 52 2019-04-01 296.953 306.2627      3.13506565
## 53 2019-05-01 330.661 335.9764      1.60750139
## 54 2019-06-01 353.239 373.7663      5.81117248
## 55 2019-07-01 410.365 416.7713      1.56111450
## 56 2019-08-01 401.732 413.9505      3.04144724
## 57 2019-09-01 360.760 350.4632     -2.85419552
## 58 2019-10-01 320.518 324.9947      1.39669430
## 59 2019-11-01 315.897 315.9439      0.01485787
## 60 2019-12-01 338.536 355.0443      4.87637747
## 61 2020-01-01 342.019 365.6755      6.91673258
## 62 2020-02-01 319.698 322.6311      0.91747119
## 63 2020-03-01 309.870 330.0022      6.49698898
## 64 2020-04-01 279.846 307.8929     10.02225714
## 65 2020-05-01 304.837 337.8621     10.83368010
## 66 2020-06-01 351.967 375.9937      6.82641041
## 67 2020-07-01 409.871 419.4201      2.32979368
## 68 2020-08-01 398.536 416.5686      4.52471739
## 69 2020-09-01 333.493 352.4772      5.69252198
## 70 2020-10-01 313.703 326.7835      4.16969651
## 71 2020-11-01 301.403 317.6550      5.39212887
## 72 2020-12-01 344.523 357.0994      3.65038747
## 73 2021-01-01 349.241 367.8299      5.32265460
## 74 2021-02-01 323.899 324.3992      0.15443203
## 75 2021-03-01 311.377 331.8369      6.57076730
## 76 2021-04-01 293.322 309.5359      5.52767908
## 77 2021-05-01 320.174 339.7622      6.11799771
## 78 2021-06-01 373.872 378.2435      1.16924635
## 79 2021-07-01 405.649 422.0926      4.05364838
## 80 2021-08-01 412.886 419.2139      1.53260044
## 81 2021-09-01 347.712 354.5102      1.95513122
## 82 2021-10-01 318.754 328.5879      3.08511731
## 83 2021-11-01 314.254 319.3805      1.63132807
## 84 2021-12-01 337.162 359.1751      6.52894985
## 85 2022-01-01 377.106 370.0046     -1.88312980
## 86 2022-02-01 326.931 326.1829     -0.22881827
## 87 2022-03-01 324.772 333.6851      2.74442621
## 88 2022-04-01 303.324 311.1924      2.59404752
## 89 2022-05-01 342.215 341.6808     -0.15611430
## 90 2022-06-01 380.649 380.5159     -0.03496421
## 91 2022-07-01 424.013 424.7936      0.18409316
## 92 2022-08-01 412.710 421.8862      2.22340223
## 93 2022-09-01 350.722 356.5619      1.66509792
## 94 2022-10-01 314.111 330.4081      5.18831285
## 95 2022-11-01 322.959 321.1209     -0.56915217
## 96 2022-12-01 363.625 361.2696     -0.64776546
```

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)

The forcast interval is 15 years, it is wide and was following same trend for next 15 years so it was not useful. I think forecast of next 5 years could be helpful. It can be observed that after year 2019 the percentage difference is exceeding 10%.

```
plot(accuracy_df$date, accuracy_df$percentage_diff, type = "l",
     xlab = "Date", main = "Percentage difference of eia data and ARIMA3 model forecasted value")
```