DATA624 - Excercise 6.2 and 6.6

Harpreet Shoker

Problem 6.2 The plastics data set consists of the monthly sales (in thousands) of product A for a plastics manufacturer for five years.

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

Loading the required libraries and plastics data

```r
library(fma)
```

```
## Loading required package: forecast

## Warning: package 'forecast' was built under R version 3.4.4
```

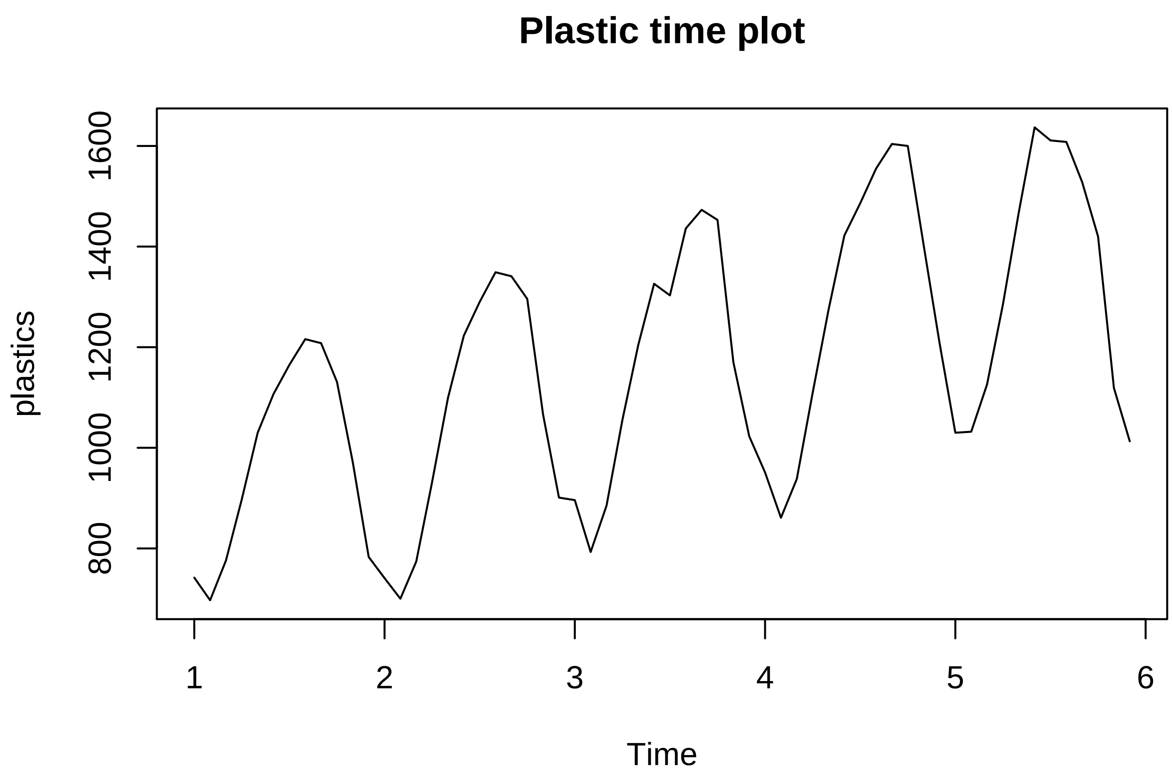
```r
library(ggplot2)
plastics
```

```
##    Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 1  742  697  776  898 1030 1107 1165 1216 1208 1131  971  783
## 2  741  700  774  932 1099 1223 1290 1349 1341 1296 1066  901
## 3  896  793  885 1055 1204 1326 1303 1436 1473 1453 1170 1023
## 4  951  861  938 1109 1274 1422 1486 1555 1604 1600 1403 1209
## 5 1030 1032 1126 1285 1468 1637 1611 1608 1528 1420 1119 1013
```

```r
ts(plastics)
```

```
## Time Series:
## Start = 1
## End = 60
## Frequency = 1
##  [1]  742  697  776  898 1030 1107 1165 1216 1208 1131  971  783  741  700
## [15]  774  932 1099 1223 1290 1349 1341 1296 1066  901  896  793  885 1055
## [29] 1204 1326 1303 1436 1473 1453 1170 1023  951  861  938 1109 1274 1422
## [43] 1486 1555 1604 1600 1403 1209 1030 1032 1126 1285 1468 1637 1611 1608
## [57] 1528 1420 1119 1013
```

```r
plot(plastics,main = 'Plastic time plot')
```



From the above Time plot we can see there are seasonal fluctuations and upward trend. Seasonal sales are peaking in summer. Overall plot shows positive trnd with sales increasing yearly.

b) Use a classical multiplicative decomposition to calculate the trend-cycle and seasonal indices.

```
plastic_model <- decompose(plastics, type="multiplicative")
trend <- plastic_model$trend #calculating trend
trend
```

```
##         Jan       Feb       Mar       Apr       May       Jun       Jul
## 1        NA        NA        NA        NA        NA        NA  976.9583
## 2 1000.4583 1011.2083 1022.2917 1034.7083 1045.5417 1054.4167 1065.7917
## 3 1117.3750 1121.5417 1130.6667 1142.7083 1153.5833 1163.0000 1170.3750
## 4 1208.7083 1221.2917 1231.7083 1243.2917 1259.1250 1276.5833 1287.6250
## 5 1374.7917 1382.2083 1381.2500 1370.5833 1351.2500 1331.2500        NA
##         Aug       Sep       Oct       Nov       Dec
## 1  977.0417  977.0833  978.4167  982.7083  990.4167
## 2 1076.1250 1084.6250 1094.3750 1103.8750 1112.5417
## 3 1175.5000 1180.5417 1185.0000 1190.1667 1197.0833
## 4 1298.0417 1313.0000 1328.1667 1343.5833 1360.6250
## 5        NA        NA        NA        NA        NA
```

```
seasonal <- plastic_model$seasonal # calculating seasonal indices
seasonal
```

```
##         Jan       Feb       Mar       Apr       May       Jun       Jul
## 1 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
## 2 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
## 3 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
## 4 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
```

```
## 5 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
##         Aug       Sep       Oct       Nov       Dec
## 1 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
## 2 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
## 3 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
## 4 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
## 5 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
```

c) Do the results support the graphical interpretation from part a?

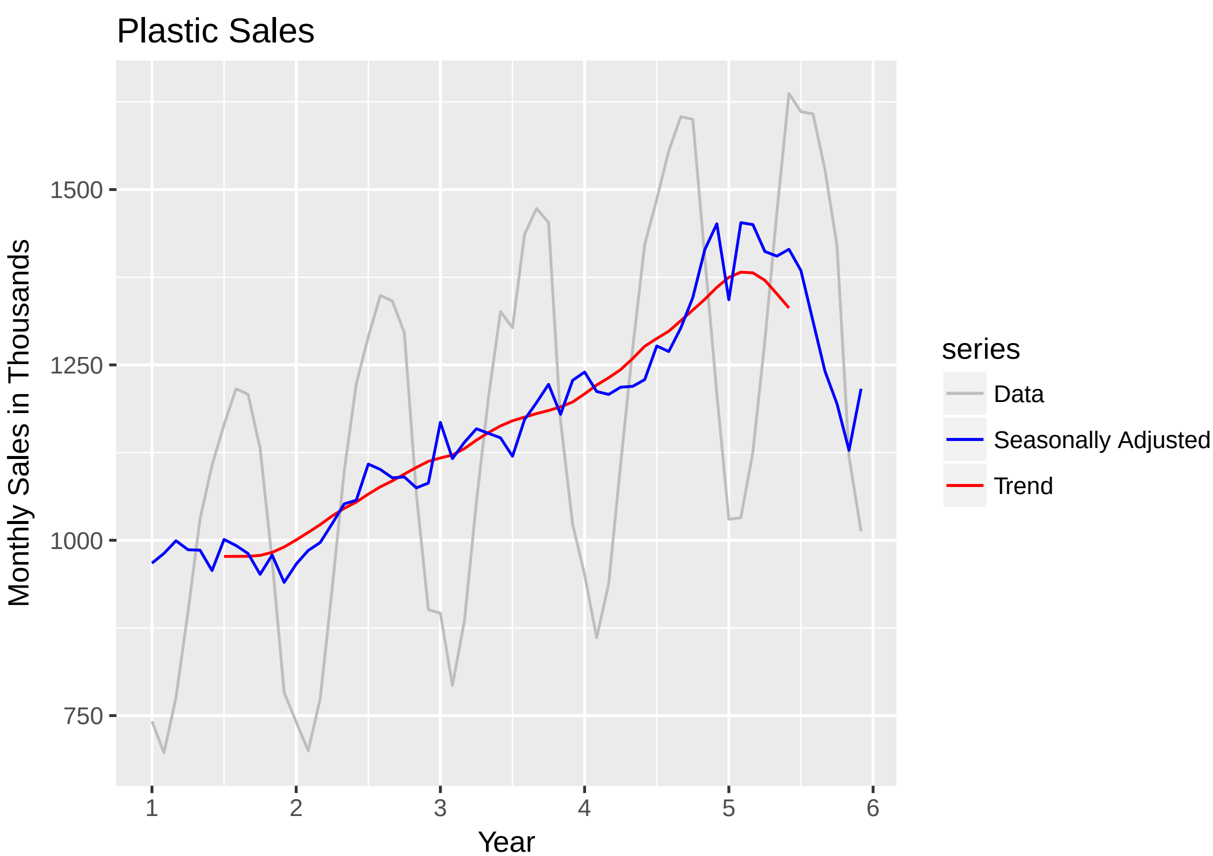
Yes, the results support the graphical interpretation.The graph indicates that the summer months have higher seasonal indices than the winter months

d) Compute and plot the seasonally adjusted data.

Here we are showing the trend-cycle component and the seasonally adjusted data, along with the original data.

```
autoplot(plastics, series="Data") +
  autolayer(trendcycle(plastic_model), series="Trend") +
  autolayer(seasadj(plastic_model), series="Seasonally Adjusted") +
  xlab("Year") + ylab("Monthly Sales in Thousands") +
  ggtitle("Plastic Sales") +
  scale_colour_manual(values=c("gray","blue","red"), breaks=c("Data","Seasonally Adjusted","Trend"))
```

```
## Warning: Removed 12 rows containing missing values (geom_path).
```



e) 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?

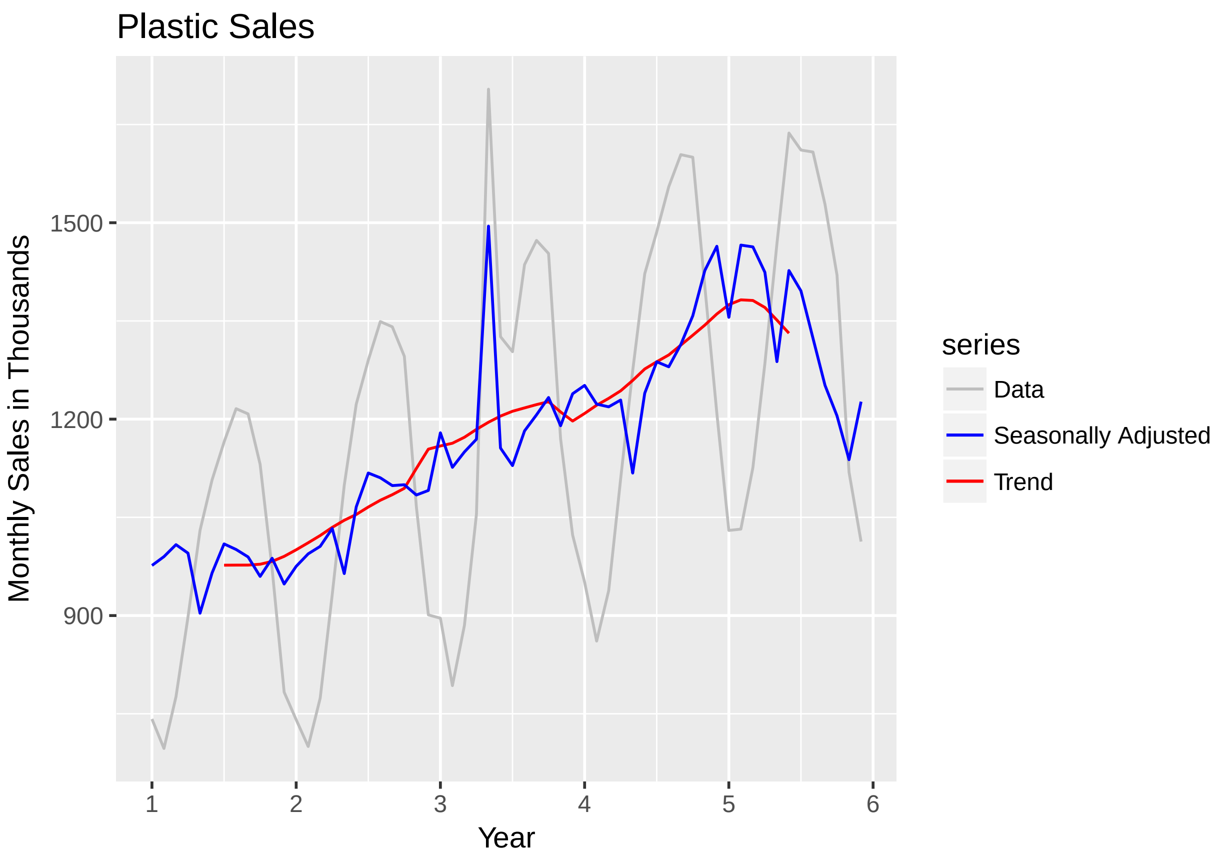
Here we are adding 500 to 29th observation

```
outlier_plastic <- plastics
outlier_plastic[29] <- outlier_plastic[29] + 500
outlier_model <- decompose(outlier_plastic, type = "multiplicative")
```

Plot showing the trend-cycle component and the seasonally adjusted data, along with the original data with modified outlier data.

```
autoplot(outlier_plastic, series = "Data") +
  autolayer(trendcycle(outlier_model), series = "Trend") +
  autolayer(seasadj(outlier_model), series = "Seasonally Adjusted") +
  xlab("Year") + ylab("Monthly Sales in Thousands") +
  ggtitle("Plastic Sales") +
  scale_color_manual(values=c("gray", "blue", "red"), breaks=c("Data", "Seasonally Adjusted", "Trend"))
```

```
## Warning: Removed 12 rows containing missing values (geom_path).
```



We can see from the above graph that outlier doesnot have much effects on Trend cycle but it is highly effecting the seasonal data

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

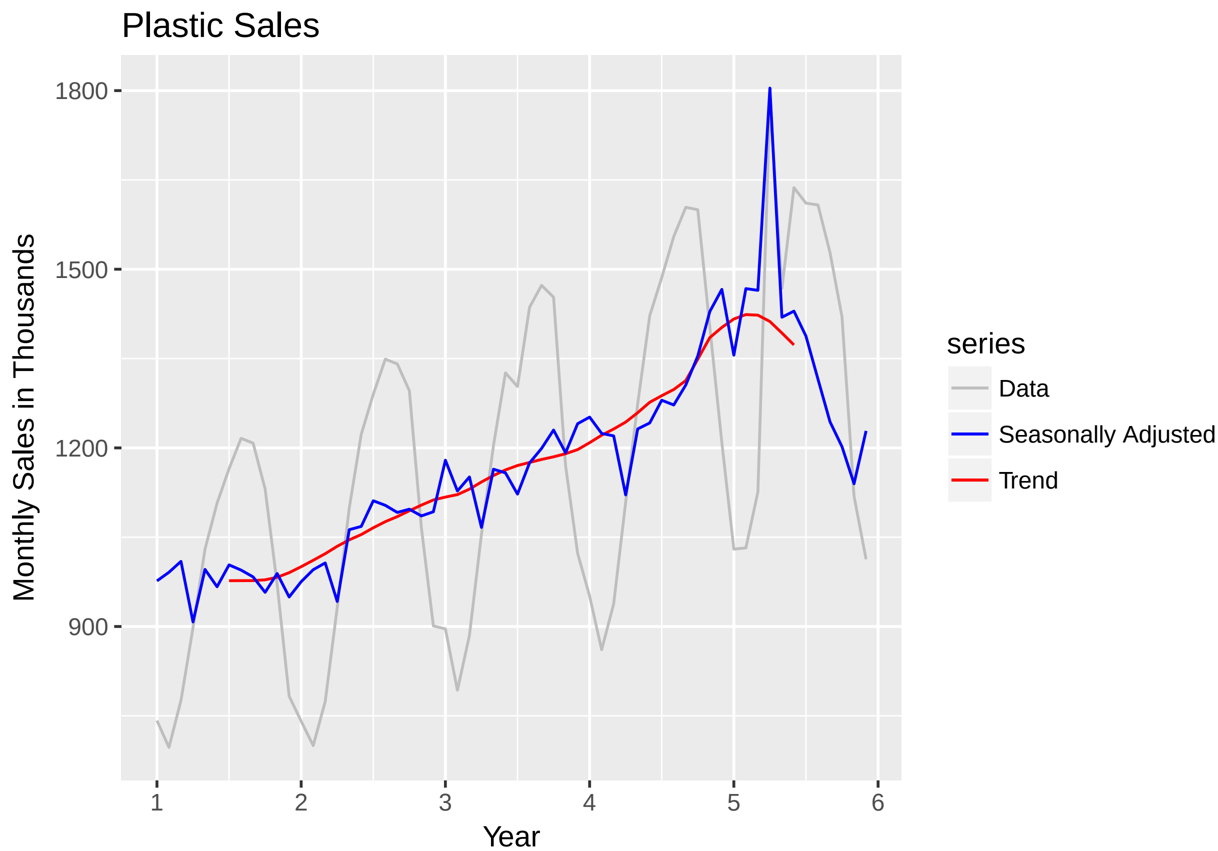
Here we are adding 500 to 52nd observation

```
outlierend_plastic <- plastics
outlierend_plastic[52] <- outlierend_plastic[52] + 500
outlierend_model <- decompose(outlierend_plastic, type = "multiplicative")
```

Plot showing the trend-cycle component and the seasonally adjusted data, along with the original data with modified outlier data

```
autoplot(outlierend_plastic, series = "Data") +
  autolayer(trendcycle(outlierend_model), series = "Trend") +
  autolayer(seasadj(outlierend_model), series = "Seasonally Adjusted") +
  xlab("Year") + ylab("Monthly Sales in Thousands") +
  ggtitle("Plastic Sales") +
  scale_color_manual(values=c("gray", "blue", "red"), breaks=c("Data", "Seasonally Adjusted", "Trend"))
```

```
## Warning: Removed 12 rows containing missing values (geom_path).
```

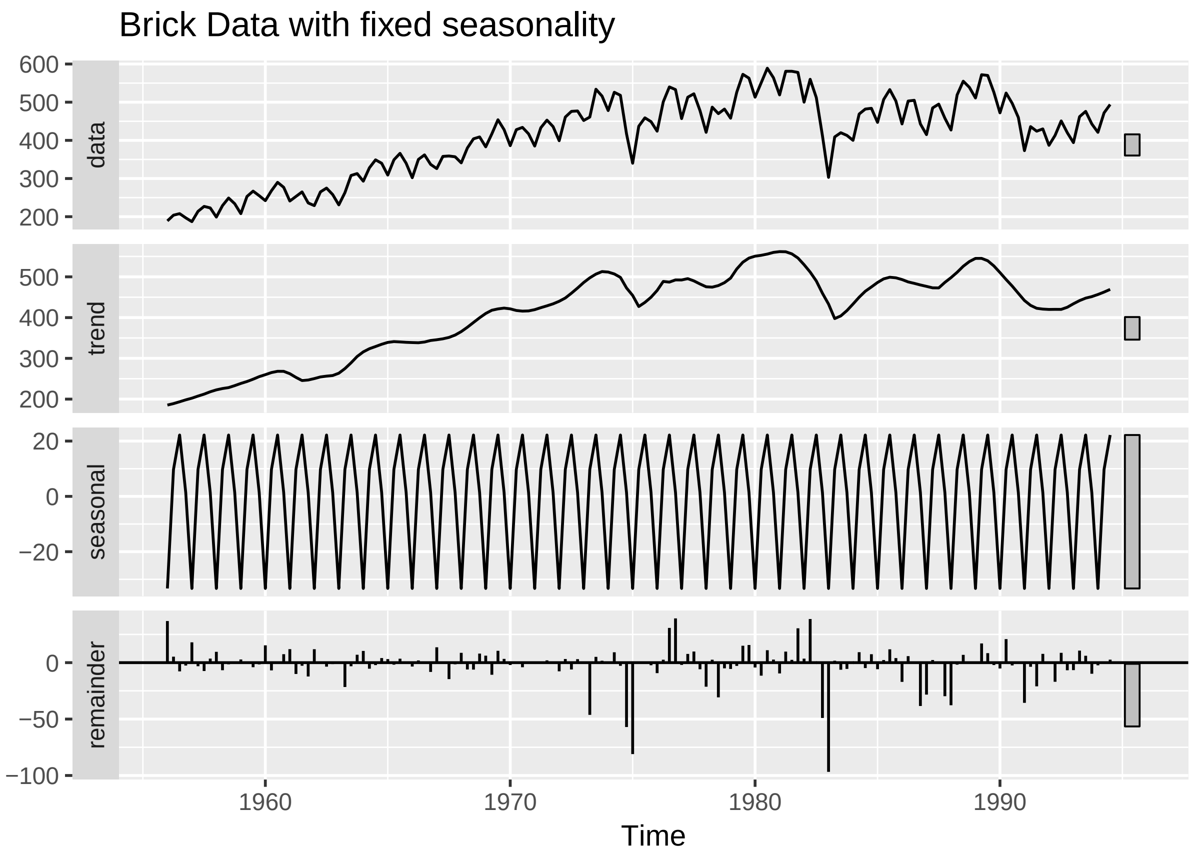


We can conclude that an outlier causes a spike in the month it is present by increasing seasonality index of that month.

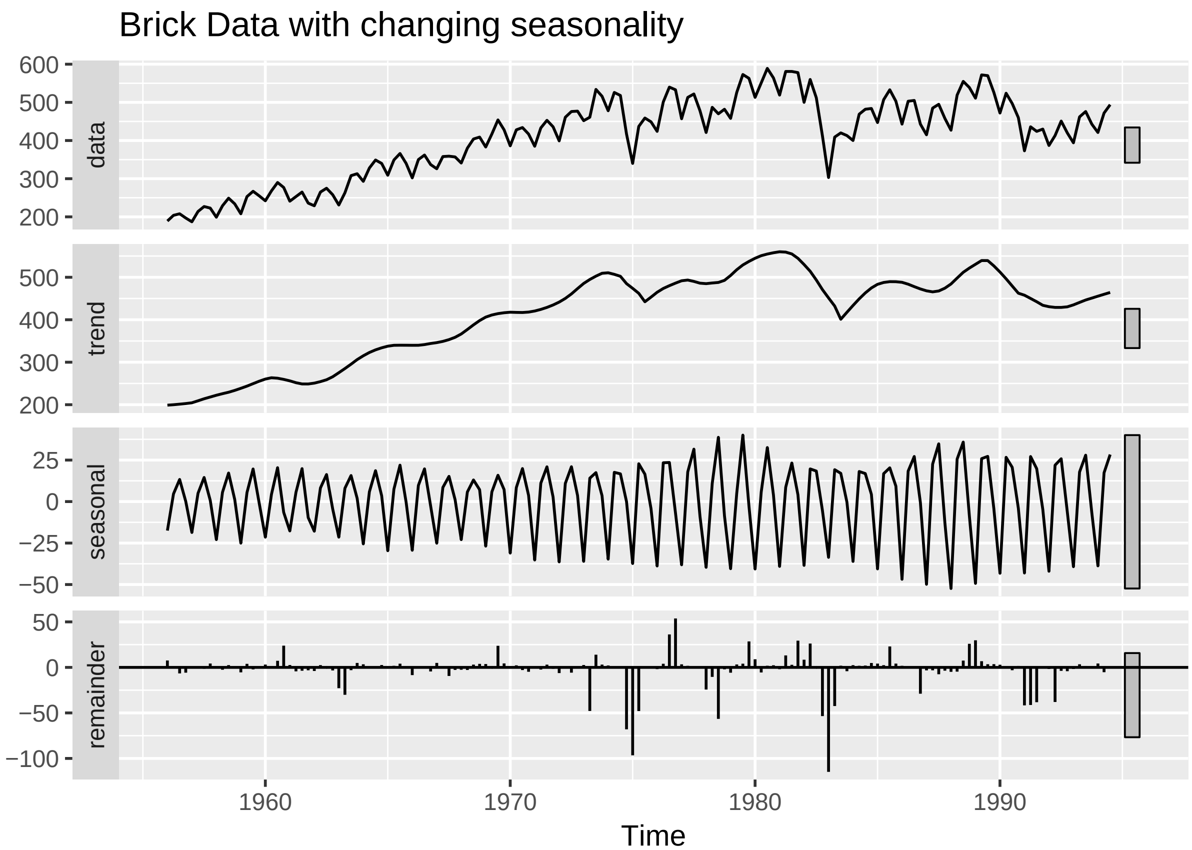
Problem 6.6 We will use the bricksq data (Australian quarterly clay brick production. 1956–1994) for this exercise.

a) Use an STL decomposition to calculate the trend-cycle and seasonal indices. (Experiment with having fixed or changing seasonality.)

```
stl_brickfixed <- stl(bricksq, s.window = "periodic",robust = TRUE)
autoplot(stl_brickfixed) +ggtitle("Brick Data with fixed seasonality")
```



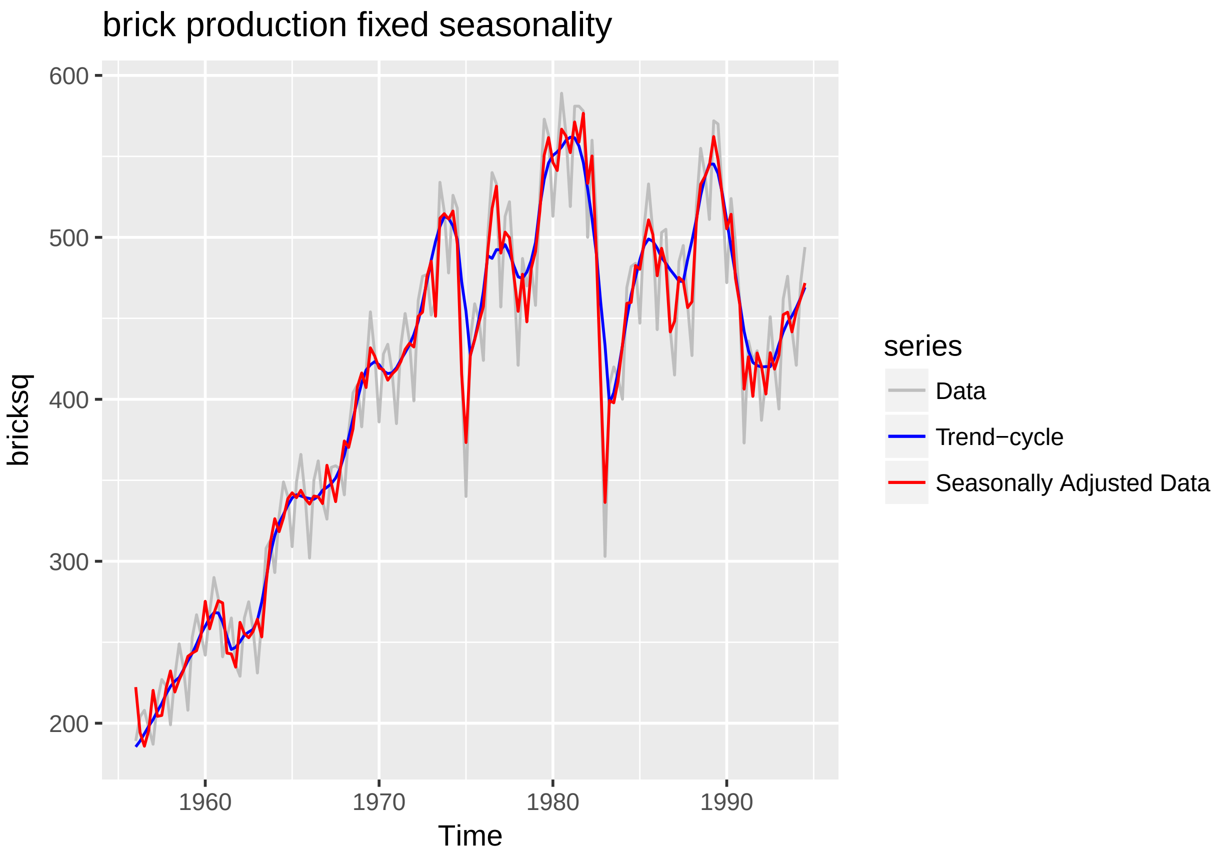

```
stl_brickchange <- stl(bricksq,s.window = 5,robust = TRUE)
autoplot(stl_brickchange) +ggtitle("Brick Data with changing seasonality")
```



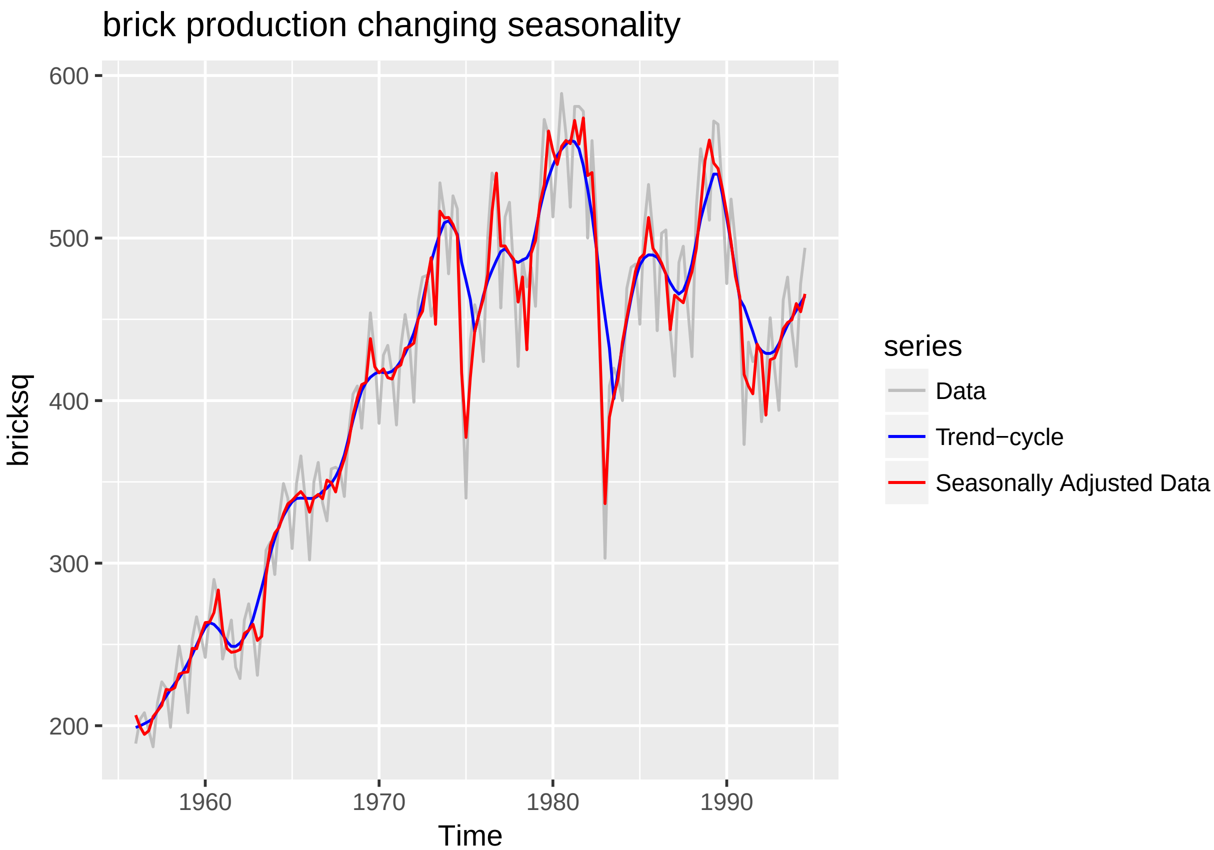
b) Compute and plot the seasonally adjusted data.

Here we are showing the trend-cycle component and the seasonally adjusted data, along with the original data.

```
# plot data which are decomposed by STL with fixed seasonality
autoplot(bricksq, series = "Data") +
  autolayer(trendcycle(stl_brickfixed),
            series = "Trend-cycle") +
  autolayer(seasadj(stl_brickfixed),
            series = "Seasonally Adjusted Data") +
  ggtitle("brick production fixed seasonality") +
  scale_color_manual(values = c("gray", "red", "blue"),
                     breaks = c("Data", "Trend-cycle", "Seasonally Adjusted Data"))
```

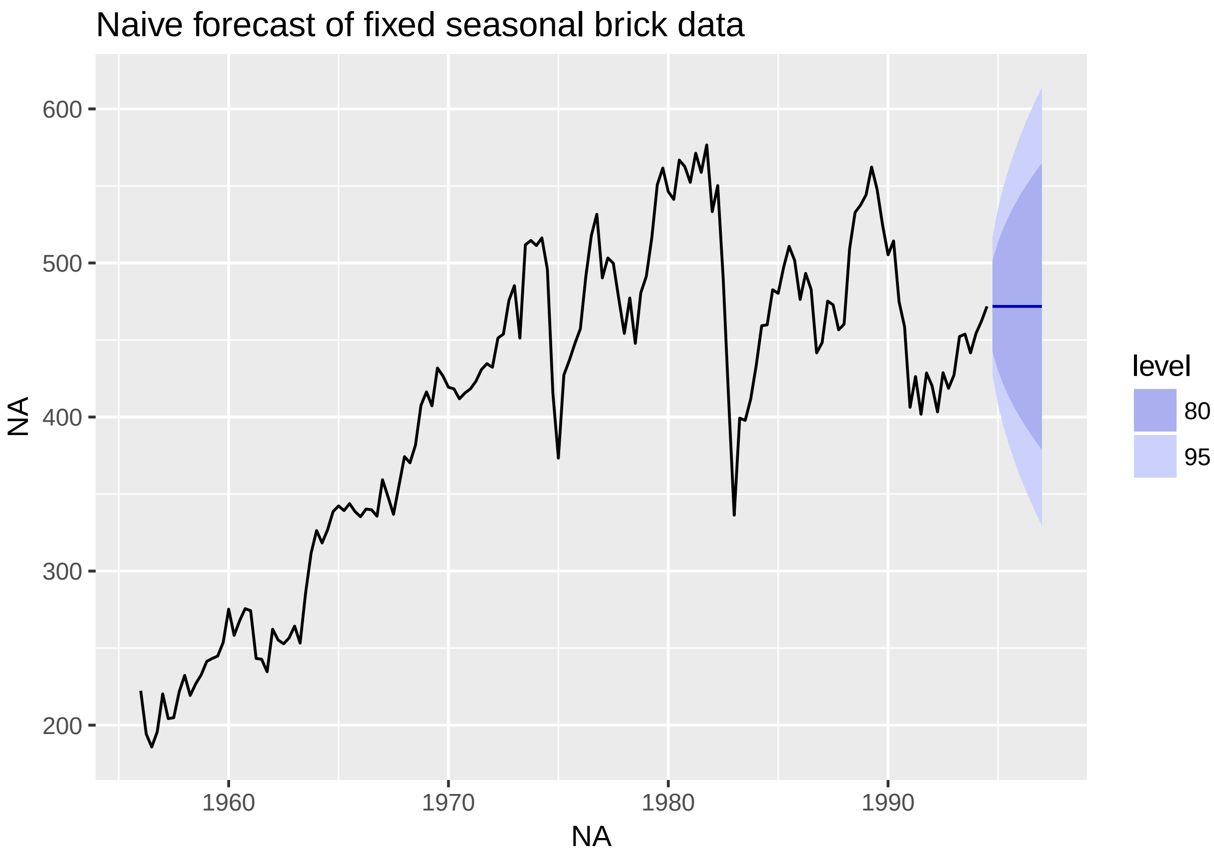



```r
# plot data which are decomposed by STL with changing seasonality
autoplot(bricksq, series = "Data") +
  autolayer(trendcycle(stl_brickchange),
            series = "Trend-cycle") +
  autolayer(seasadj(stl_brickchange),
            series = "Seasonally Adjusted Data") +
  ggtitle("brick production changing seasonality") +
  scale_color_manual(values = c("gray", "red", "blue"),
                     breaks = c("Data", "Trend-cycle", "Seasonally Adjusted Data"))
```

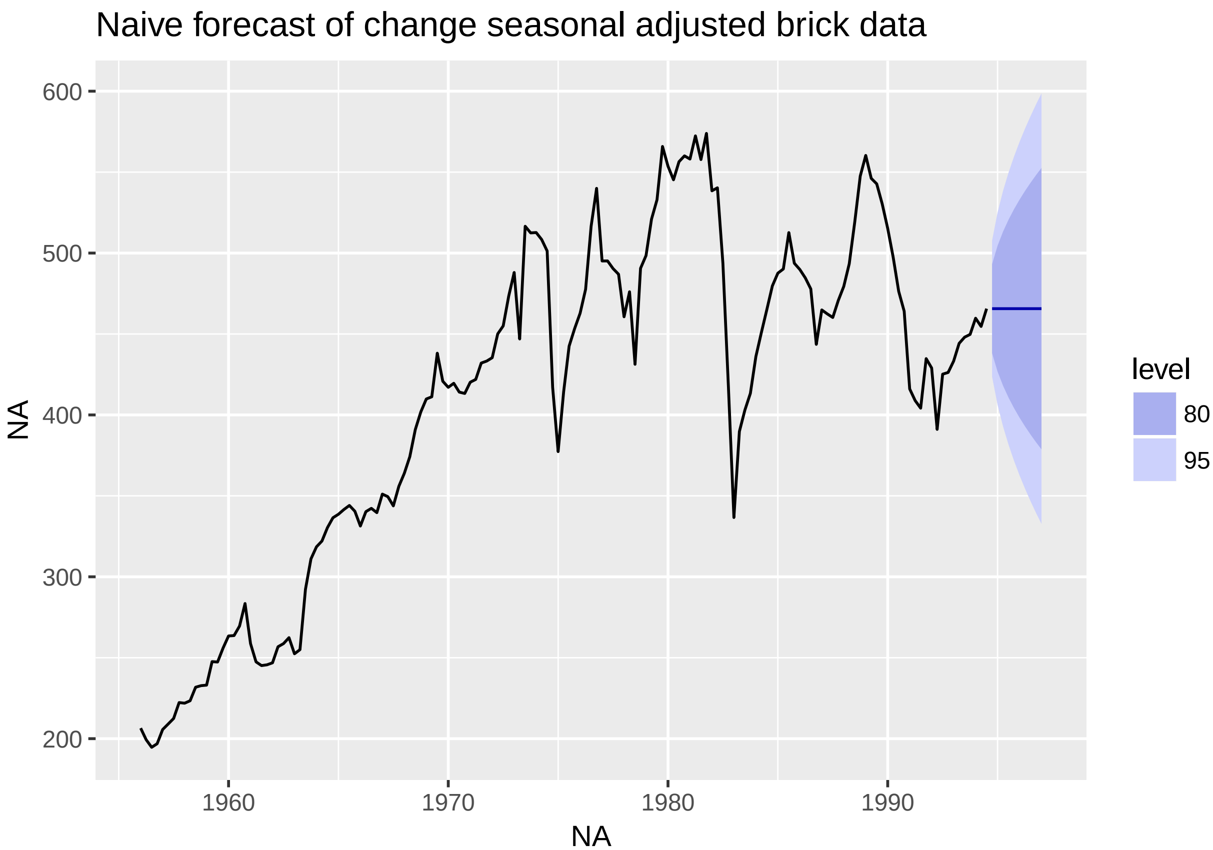


c) Use a naïve method to produce forecasts of the seasonally adjusted data.

```
stl_brickfixed %>% seasadj() %>% naive() %>% autoplot() +
  ggtitle(label = "Naive forecast of fixed seasonal brick data")
```



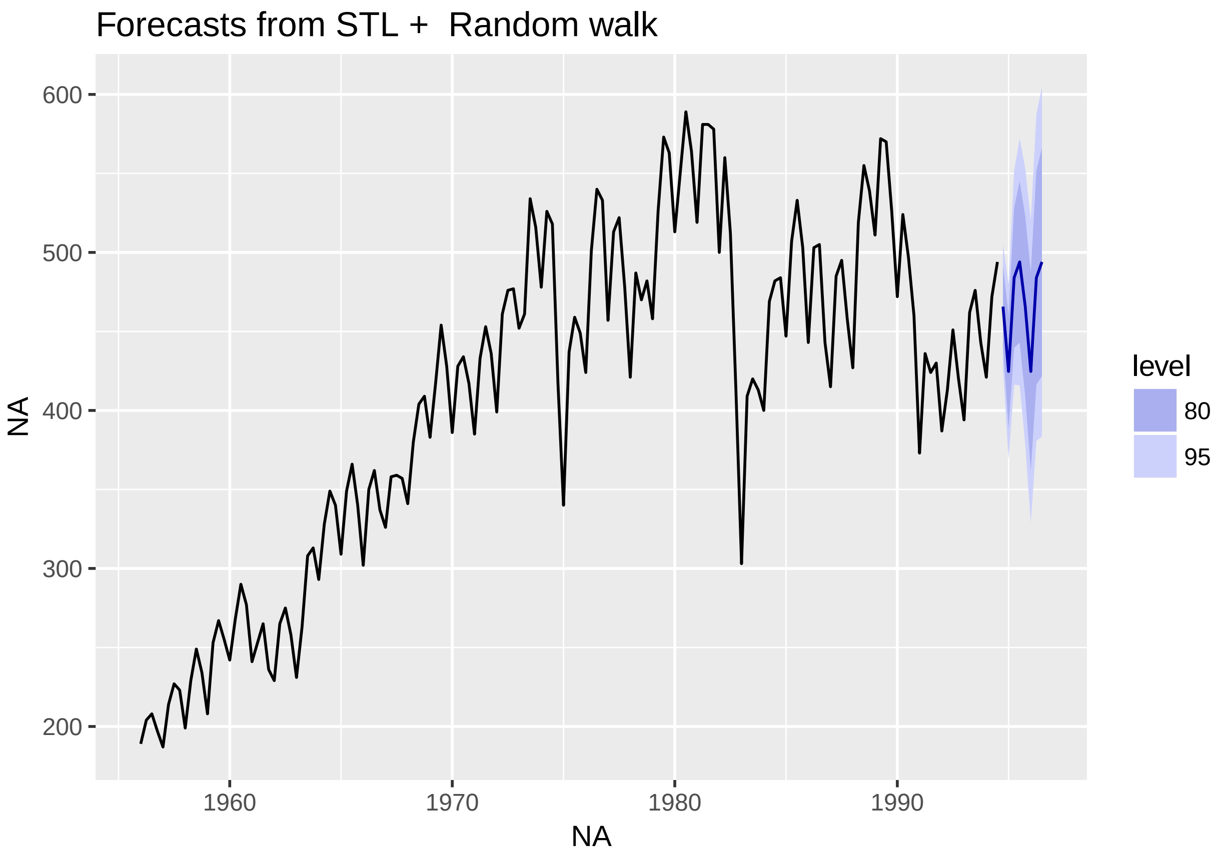

```
stl_brickchange %>% seasadj() %>% naive() %>% autoplot() +
  ggtitle(label = "Naive forecast of change seasonal adjusted brick data")
```



can see that the prediction intervals of seasonally adjusted data decomposed by STL with changing seasonality have smaller range than the one with fixed seasonality. It happened because the variance of the remainder component decreased when the seasonality can be changed.

d) Use stlf() to reseasonalise the results, giving forecasts for the original data.

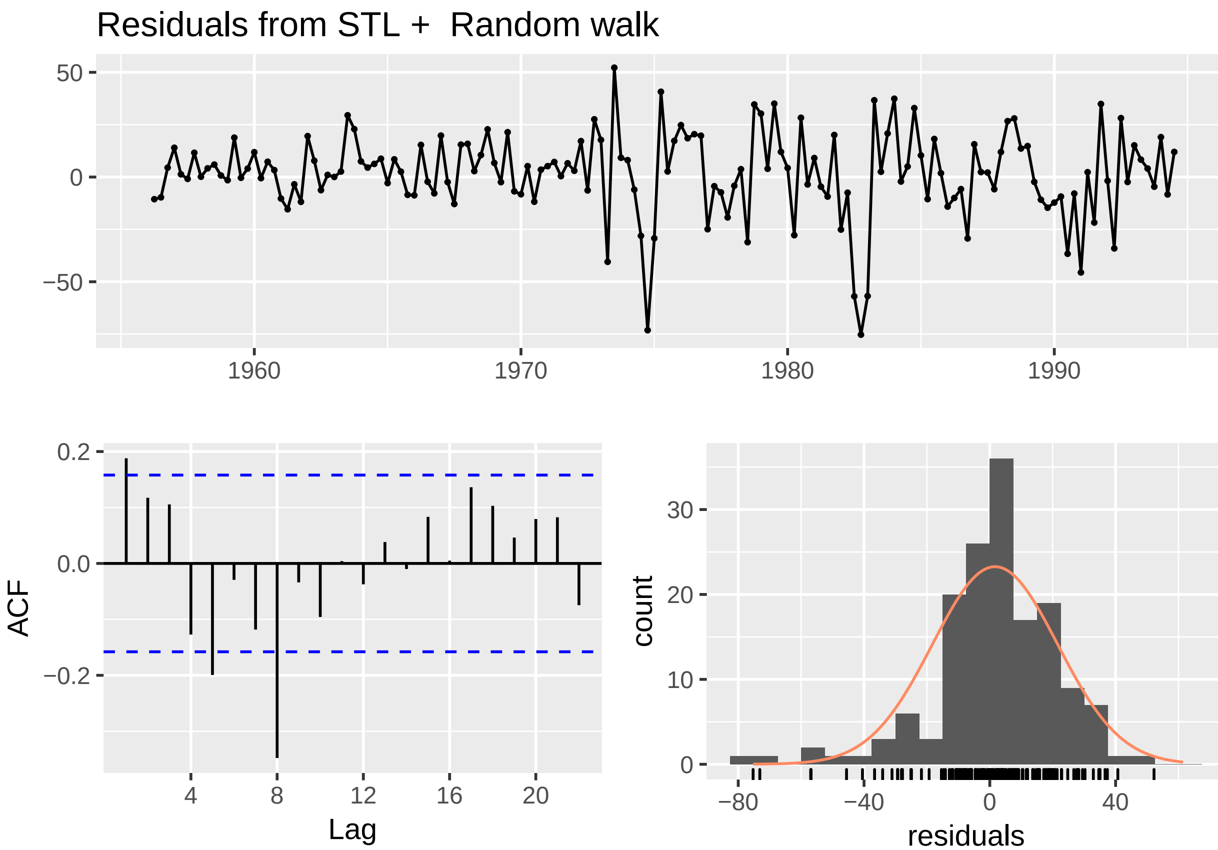
```
fcast <- stlf(bricksq, method='naive')
autoplot(fcast)
```



e) Do the residuals look uncorrelated?

```
checkresiduals(fcast)
```

```
## Warning in checkresiduals(fcast): The fitted degrees of freedom is based on
## the model used for the seasonally adjusted data.
```

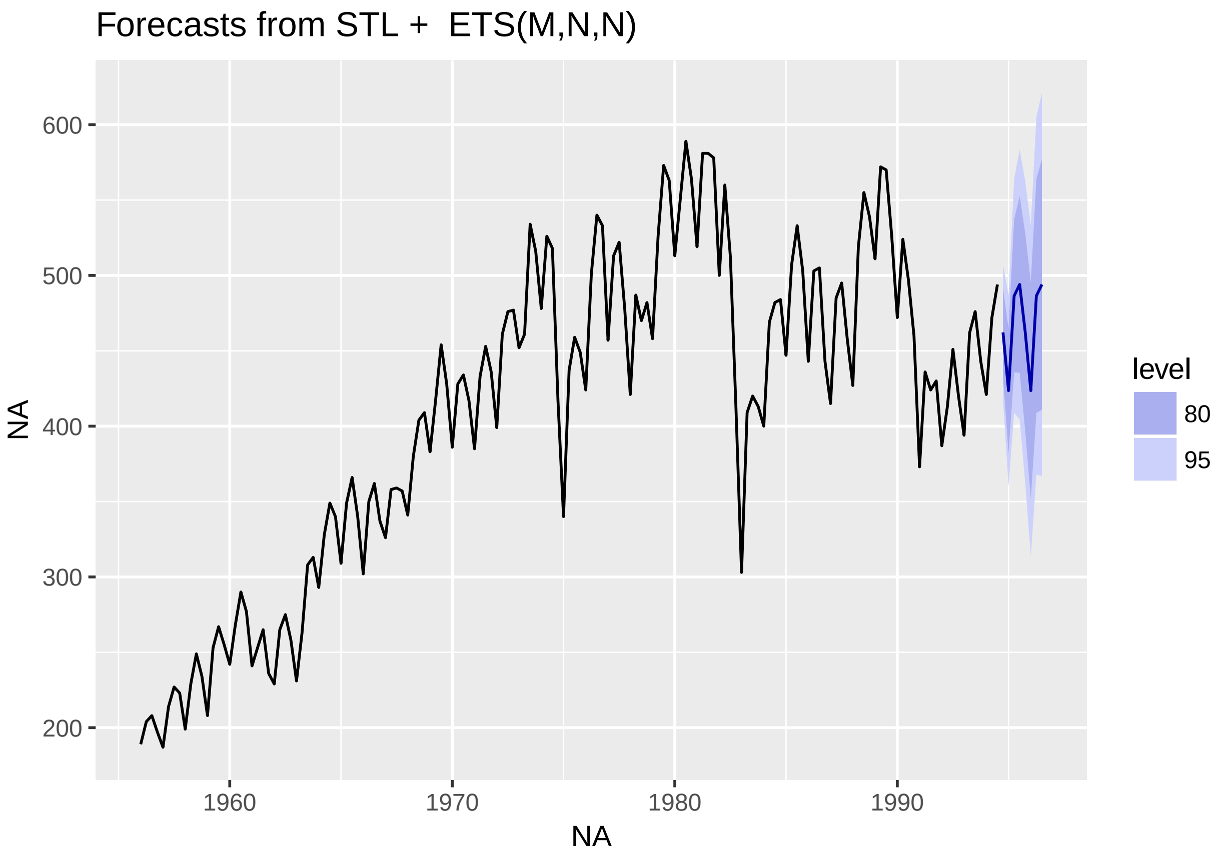



```
##
##  Ljung-Box test
##
## data:  Residuals from STL +  Random walk
## Q* = 40.829, df = 8, p-value = 2.244e-06
##
## Model df: 0.   Total lags used: 8
```

The residuals are correlated with each other.

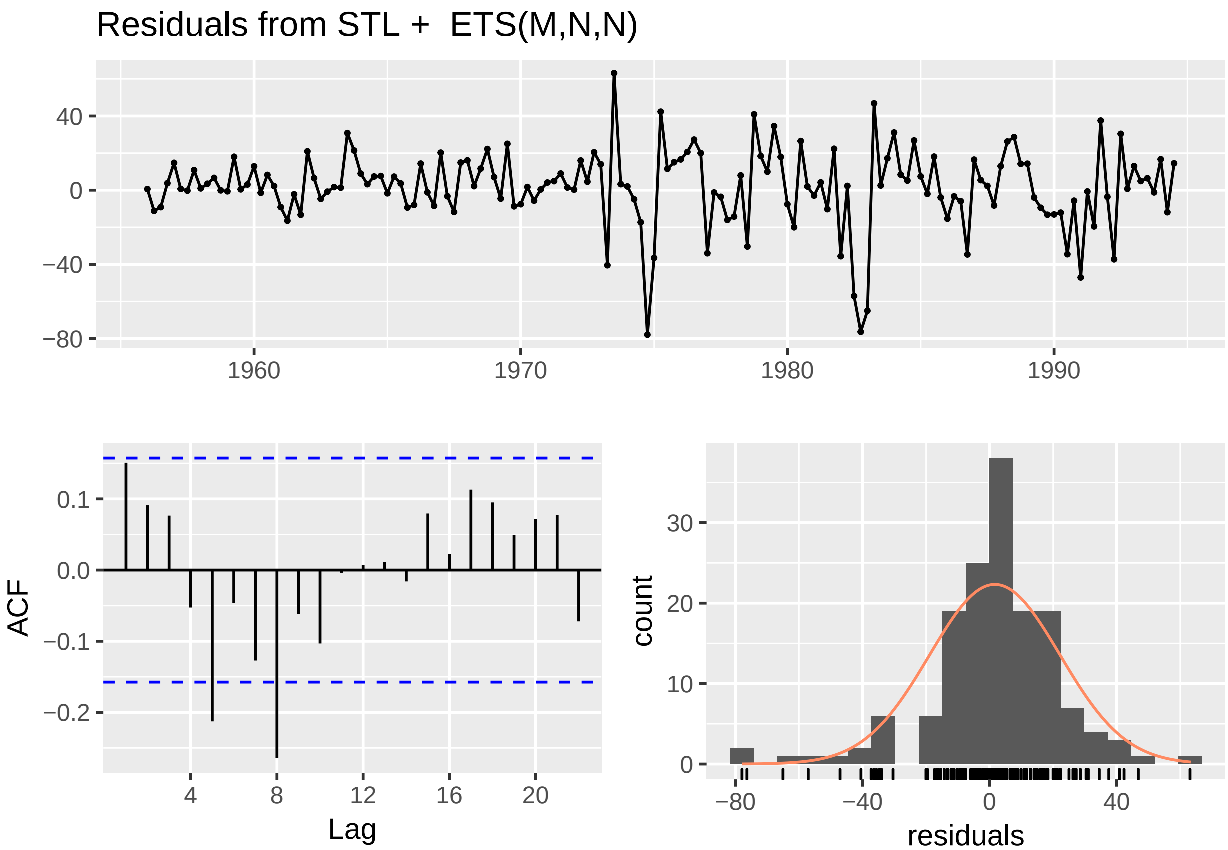
f) Repeat with a robust STL decomposition. Does it make much difference?

```
stlf_brickrobust <- stlf(bricksq, robust = TRUE)
autoplot(stlf_brickrobust)
```




```
checkresiduals(stlf_brickrobust)
```

```
## Warning in checkresiduals(stlf_brickrobust): The fitted degrees of freedom
## is based on the model used for the seasonally adjusted data.
```




```
##
##  Ljung-Box test
##
## data:  Residuals from STL +  ETS(M,N,N)
## Q* = 28.163, df = 6, p-value = 8.755e-05
##
## Model df: 2.   Total lags used: 8
```

It looked like the autocorrelations became lower generally, but there are still some high values left.

g) Compare forecasts from stlf() with those from snaive(), using a test set comprising the last 2 years of data. Which is better?

Splitting data into test and train data set and then applying stlf and snaive on the train data.

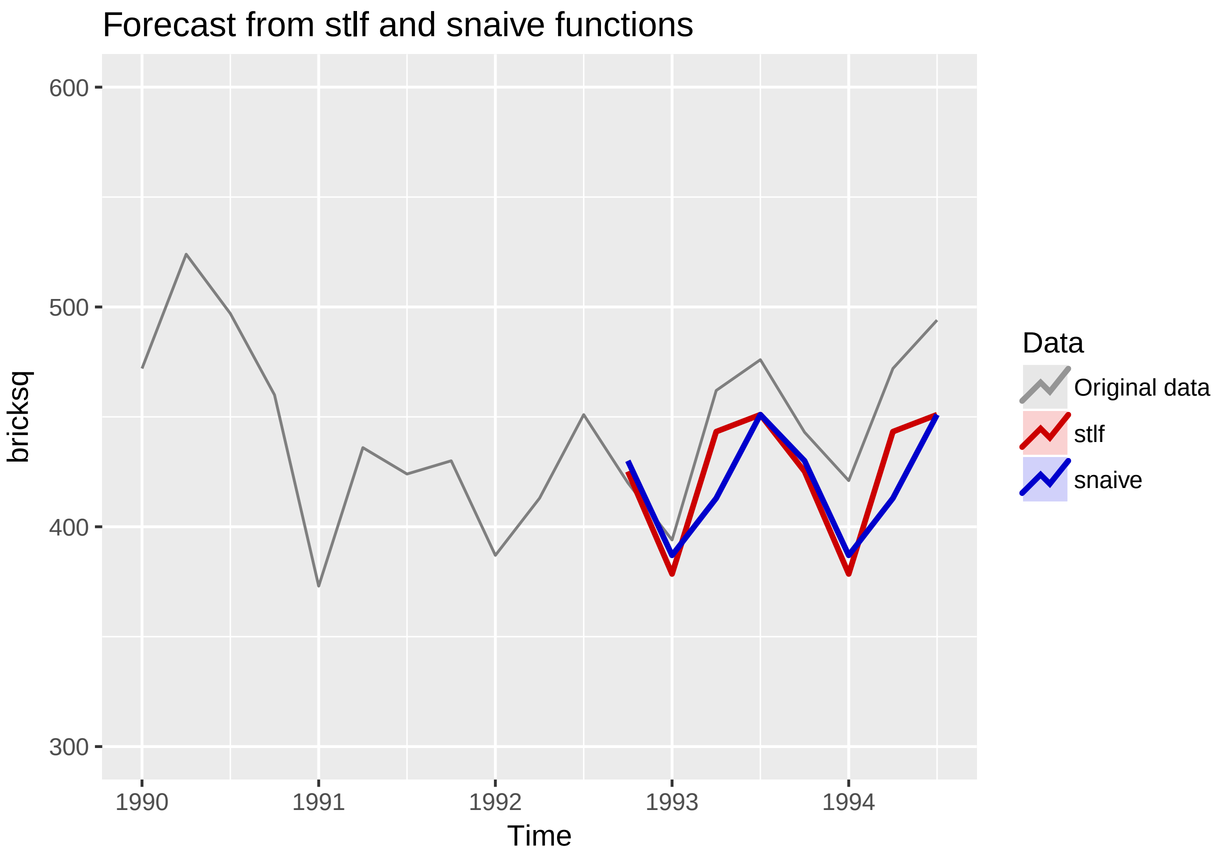
```
#subsetting train data set leaving last 2 years
train_brick <- subset(bricksq,
                      end = length(bricksq) - 8)
#subsetting test data set including only last 2 years data
test_brick <- subset(bricksq,
                     start = length(bricksq) - 7)

snaive_bricksq_train <- snaive(train_brick)

stlf_bricksq_train <- stlf(train_brick, robust = TRUE)
# plot data and forecast results
autoplot(bricksq, series = "Original data") +
  autolayer(stlf_bricksq_train, PI = FALSE, size = 1,
```

```r
           series = "stlf") +
  autolayer(snaive_bricksq_train, PI = FALSE, size = 1,
           series = "snaive") +
   scale_color_manual(values = c("gray50", "blue", "red"),
                     breaks = c("Original data", "stlf", "snaive")) +
  scale_x_continuous(limits = c(1990, 1994.5)) +
  scale_y_continuous(limits = c(300, 600)) +
  guides(colour = guide_legend(title = "Data")) +
  ggtitle("Forecast from stlf and snaive functions")
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```




```r
accuracy(snaive_bricksq_train,test_brick)
```

```
##                     ME     RMSE      MAE      MPE     MAPE      MASE
## Training set  6.174825 49.71281 36.41259 1.369661 8.903098 1.0000000
## Test set     27.500000 35.05353 30.00000 5.933607 6.528845 0.8238909
##                   ACF1 Theil's U
## Training set 0.8105927        NA
## Test set     0.2405423 0.9527794
```

```r
accuracy(stlf_bricksq_train,test_brick)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set  1.452849 20.90158 14.60718 0.3303597 3.599758 0.4011574
## Test set     23.261210 27.47526 24.55146 5.1141619 5.421365 0.6742575
##                   ACF1 Theil's U
```

```
## Training set 0.1652305        NA
## Test set     0.2030710 0.7163537
```

From the above forescast plot we can see that the forecasts from stlf function are more similar to the original data than the forecasts from snaive function.stlf function can also use trend, and its seasonality can change over time. The test set have trend with seasonality.Sometimes, different accuracy measures will lead to different results as to which forecast method is best. However, in this case, all of the results point to the stlf method as the best method for this data set.