DATA 624 Homework 4

Bin Lin

2018-3-26

7.8 1. Data set books contains the daily sales of paperback and hardcover books at the same store. The task is to forecast the next four days' sales for paperback and hardcover books (data set books).

a. Plot the series and discuss the main features of the data.

Approaches: I was using the Time Series plot to get a general graph of the daily sales for both paperback and hardcover books. Then I will use classical decomposition to investigate the trend, seasonal, and random components. Since books contains daily data, I would set the time frequency into 7 to search for any weekly seasonality

Interpretation: From the general Time Series graph, there is clear upward trend associated with both daily sales of paperback and hardcover books. The sale varies a lot from day to day. No apparent seasonality observed in this graph. In addition, it looks like there is a inverse relationship between two variables as sale of one peak, the sale of the other bottoms. From the decomposition graph, we can tell sales of both trending upwards. There is some sort of seasonality exist, as the sale then to peak once a week.

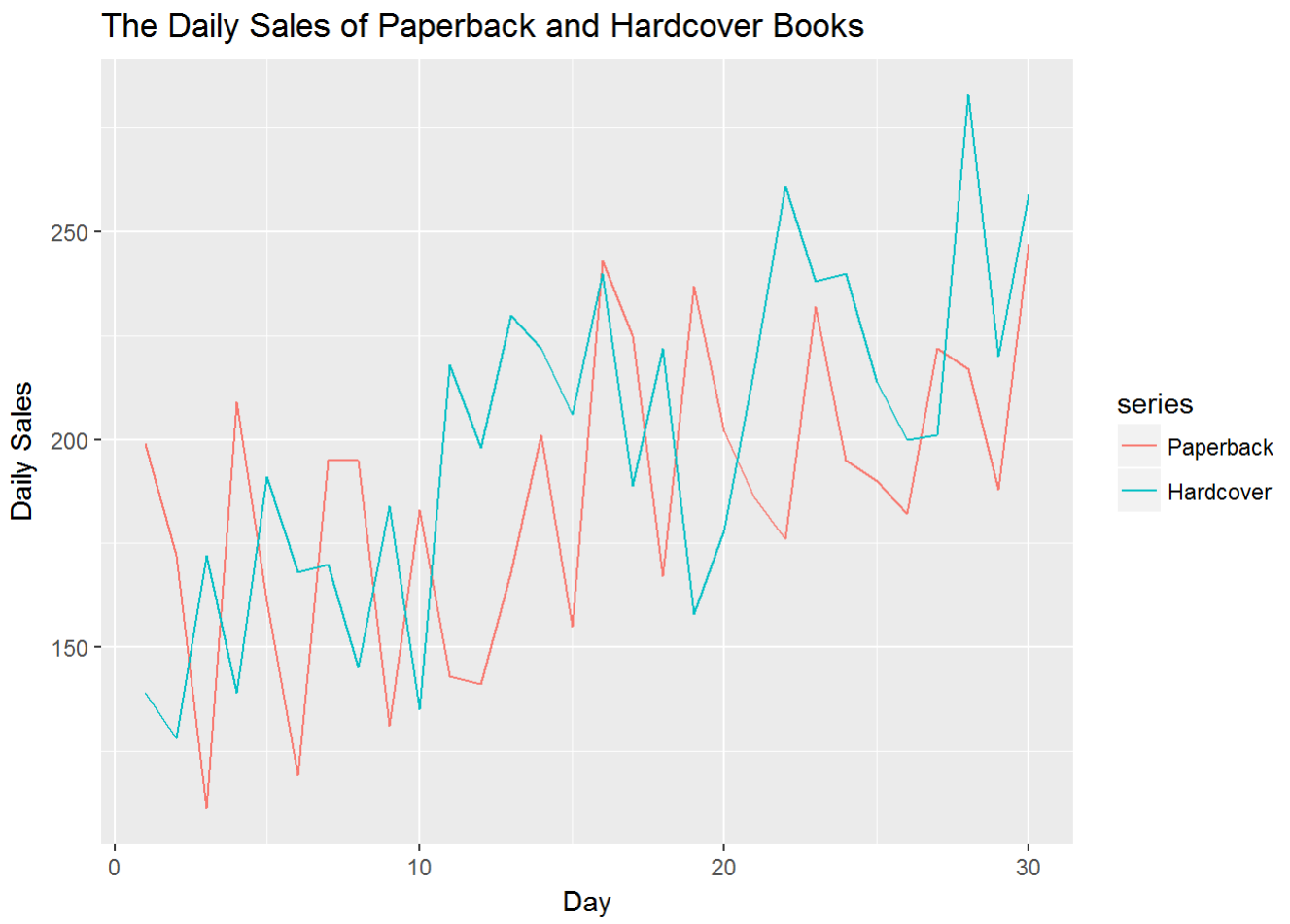
```
library(fma)
```

```
## Loading required package: forecast
```

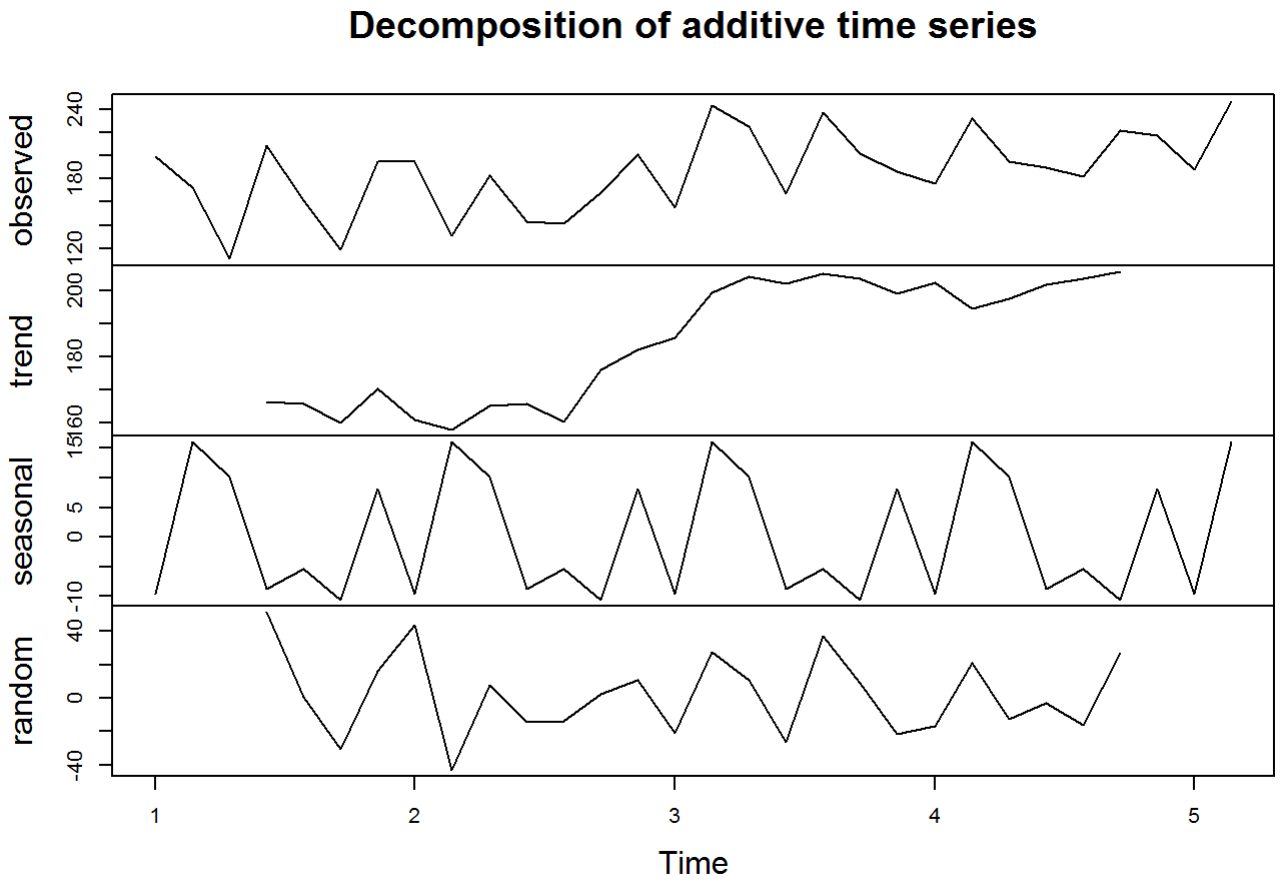
```
library(ggplot2)
data("books")
head(books)
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
##   Paperback Hardcover
## 1       199       139
## 2       172       128
## 3       111       172
## 4       209       139
## 5       161       191
## 6       119       168
```

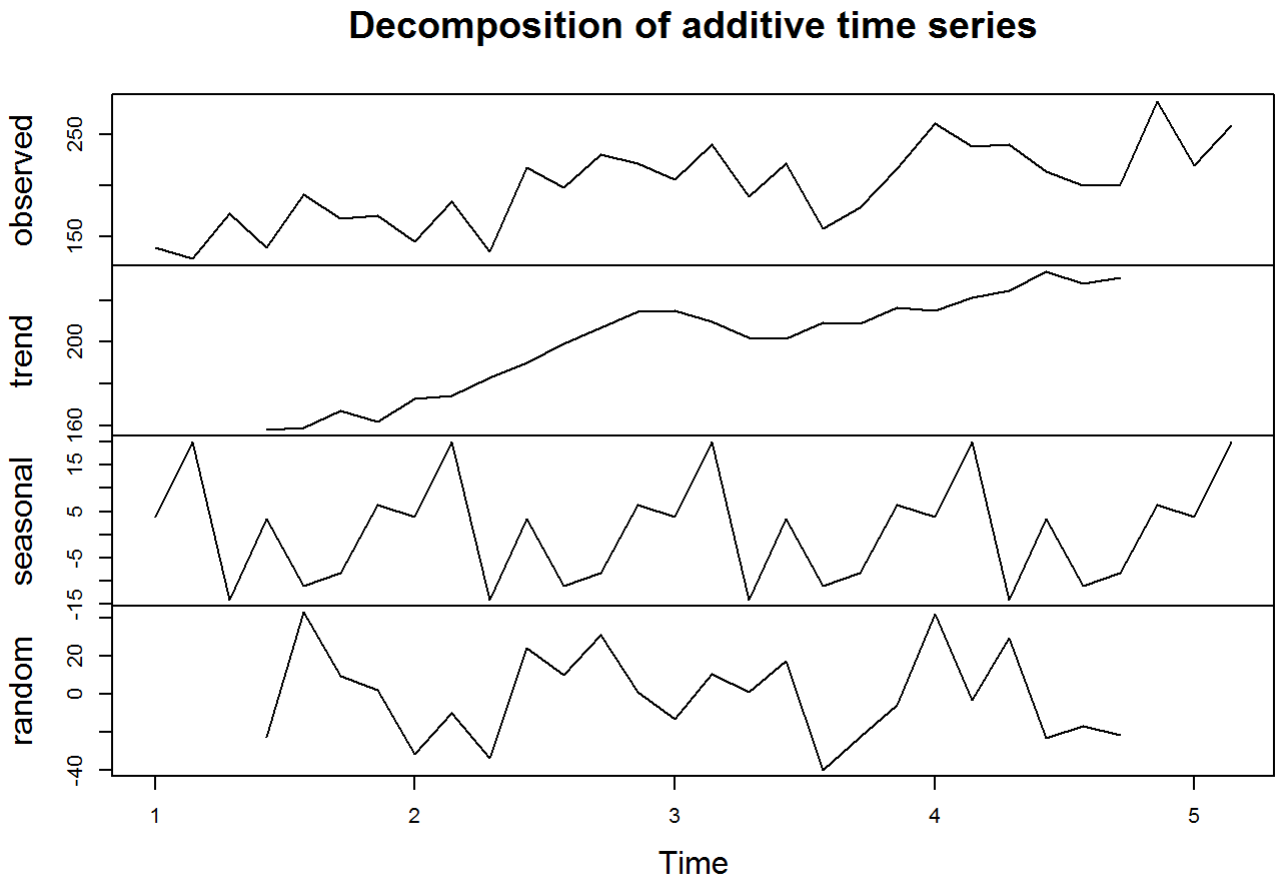
```
autoplot(books, main="The Daily Sales of Paperback and Hardcover Books", ylab="Daily Sales", xla
b="Day")
```




```
books_ts <- ts(books, frequency = 7)
fit_decom1 <- decompose(books_ts[, 1])
plot(fit_decom1)
```




```
fit_decom2 <- decompose(books_ts[, 2])
plot(fit_decom2)
```



b. Use simple exponential smoothing with the ses function (setting initial="simple") and explore different values of alpha for the paperback series. Record the within-sample SSE for the one-step forecasts. Plot SSE against alpha and find which value of alpha works best. What is the effect of alpha on the forecasts?

Approaches: Simple exponential smoothing is method suitable for forecasting data with no trend or seasonal pattern. Forecasts are calculated using weighted averages where the weights decrease exponentially as observations come from further in the past.

Interpretation:

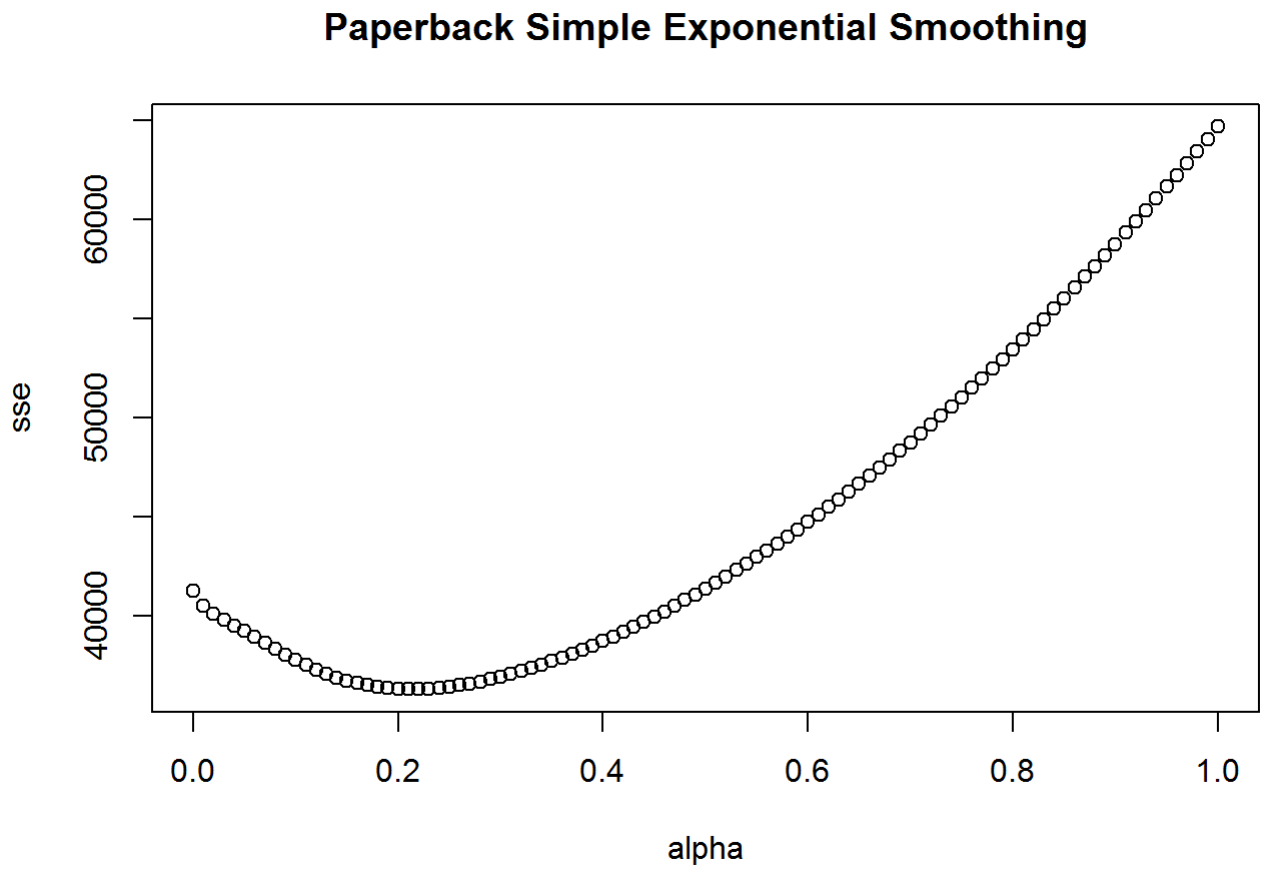
From the graph, we are able to tell that alpha the SSE reaches its minimum when alpha is around 0.21. Alpha is the smoothing parameter, it can only take values between 0 and 1. The larger the alpha value, more weight is given to the recent observation and vice versa.

```
alpha <- numeric()
sse <- numeric()

for(i in seq(0, 1, 0.01)) {

  fit <- ses(books[, 1], alpha = i, initial="simple", h = 4)
  alpha <- c(alpha, i)
  sse <- c(sse, fit$model$SSE)
}

result <- data.frame(alpha, sse)
plot(result, main="Paperback Simple Exponential Smoothing")
```



c. Now let ses select the optimal value of alpha. Use this value to generate forecasts for the next four days. Compare your results with 2.

We can use summary statistics of simple exponential smoothing to find out the optimal value of alpha, which minimize SSE. Or on the other way, we can extract the value from the model, we will get the optimal alpha value is 0.2125115.

```
fit1  <- ses(books[, 1], initial="simple", h = 4)
summary(fit1)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = books[, 1], h = 4, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.2125
##
##   Initial states:
##     l = 199
##
##   sigma:  34.7918
## Error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 1.749509 34.79175 28.64424 -2.770157 16.56938 0.7223331
##                     ACF1
## Training set -0.1268119
##
## Forecasts:
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 31       210.1537 165.5663 254.7411 141.9631 278.3443
## 32       210.1537 164.5706 255.7368 140.4404 279.8671
## 33       210.1537 163.5962 256.7112 138.9501 281.3573
## 34       210.1537 162.6418 257.6657 137.4905 282.8170
```

```
optimal_alpha1 <- fit1$model$par["alpha"]
optimal_alpha1
```

```
##     alpha
## 0.2125115
```

d. Repeat but with initial="optimal". How much difference does an optimal initial level make?

If we repeat the process with initial = "optimal", the optimal of alpha value is 0.1685384. The SSE values we obtain using three different methods are very closed to each other.

```
fit2  <- ses(books[, 1], initial="optimal", h = 4)
summary(fit2)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = books[, 1], h = 4, initial = "optimal")
##
##   Smoothing parameters:
##     alpha = 0.1685
##
##   Initial states:
##     l = 170.8257
##
##   sigma:  33.6377
##
##      AIC     AICc      BIC
## 318.9747 319.8978 323.1783
##
## Error measures:
##                     ME     RMSE     MAE       MPE     MAPE      MASE
## Training set 7.176212 33.63769 27.8431 0.4737524 15.57782 0.7021303
##                     ACF1
## Training set -0.2117579
##
## Forecasts:
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 31       207.1098 164.0013 250.2182 141.1811 273.0384
## 32       207.1098 163.3934 250.8261 140.2513 273.9682
## 33       207.1098 162.7937 251.4258 139.3342 274.8853
## 34       207.1098 162.2021 252.0174 138.4294 275.7901
```

```
optimal_alpha2 <- fit2$model$par["alpha"]
optimal_alpha2
```

```
##     alpha
## 0.1685384
```

```
method <- c("Graph", "Simple", "Optimal")
alphas <- c(0.21, optimal_alpha1, optimal_alpha2)
SSES <- c(min(result$sse), fit1$model$SSE, sum(residuals(fit2) ^ 2))

final <- data.frame(method,alphas, SSES)
final
```

```
##    method    alphas     SSES
## 1   Graph 0.2100000 36314.59
## 2  Simple 0.2125115 36313.98
## 3 Optimal 0.1685384 33944.82
```

e. Repeat steps (b)-(d) with the hardcover series.

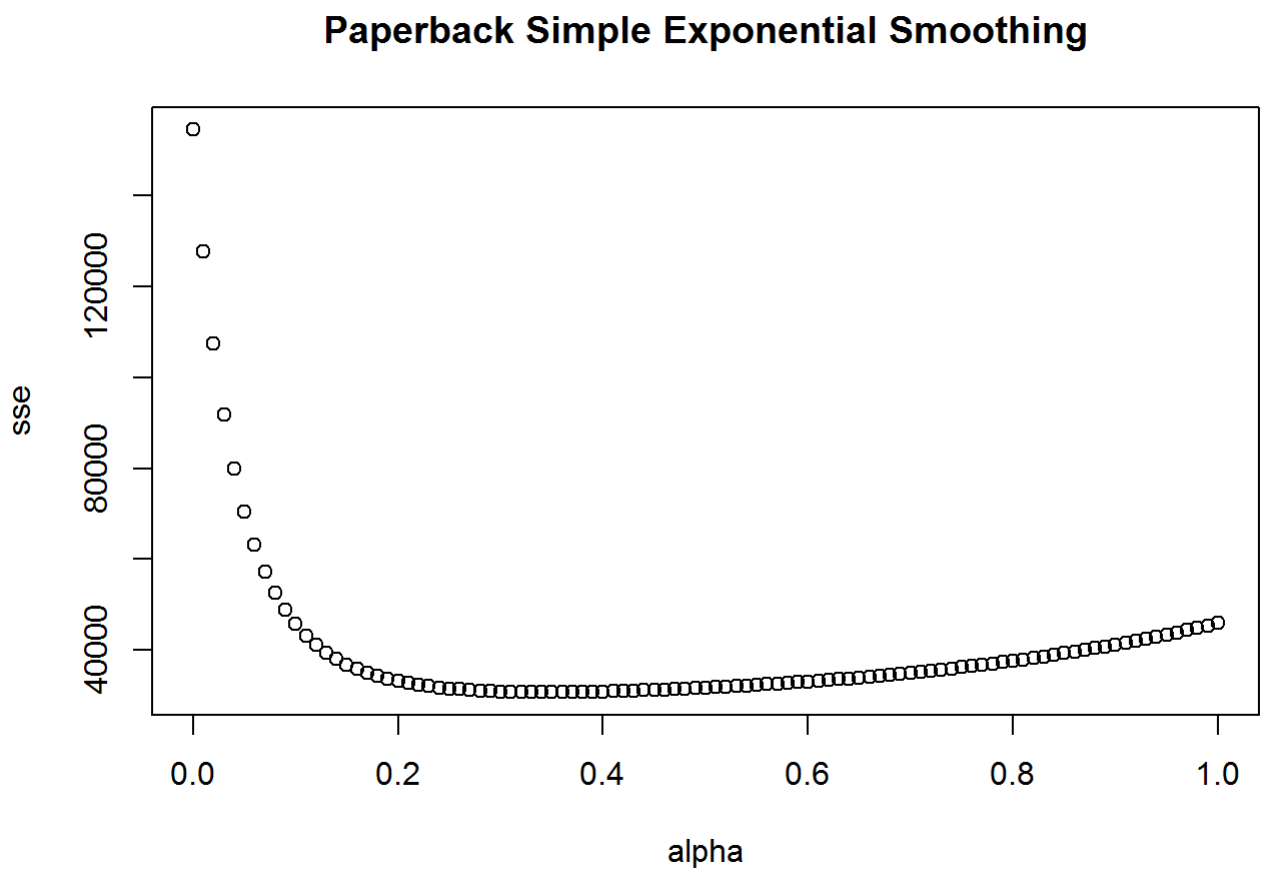
After repeating the steps, we can tell three different methods end up with different values of alpha. However, the SSE values are all very similar.

```
alpha <- numeric()
sse <- numeric()

for(i in seq(0, 1, 0.01)) {

  fit <- ses(books[, 2], alpha = i, initial="simple", h = 4)
  alpha <- c(alpha, i)
  sse <- c(sse, fit$model$SSE)
}

result <- data.frame(alpha, sse)
plot(result, main="Paperback Simple Exponential Smoothing")
```




```
fit1  <- ses(books[, 2], initial="simple", h = 4)
summary(fit1)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = books[, 2], h = 4, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.3473
##
##   Initial states:
##     l = 139
##
##   sigma:  32.0198
## Error measures:
##                    ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 9.72952 32.01982 26.34467 3.104211 13.05063 0.7860035
##                     ACF1
## Training set -0.1629042
##
## Forecasts:
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 31       240.3808 199.3457 281.4158 177.6231 303.1385
## 32       240.3808 196.9410 283.8206 173.9453 306.8162
## 33       240.3808 194.6625 286.0990 170.4608 310.3008
## 34       240.3808 192.4924 288.2691 167.1418 313.6197
```

```
optimal_alpha1 <- fit1$model$par["alpha"]
optimal_alpha1
```

```
##     alpha
## 0.3473305
```

```
fit2  <- ses(books[, 2], initial="optimal", h = 4)
summary(fit2)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = books[, 2], h = 4, initial = "optimal")
##
##   Smoothing parameters:
##     alpha = 0.3283
##
##   Initial states:
##     l = 149.2836
##
##   sigma:  31.931
##
##      AIC     AICc      BIC
## 315.8506 316.7737 320.0542
##
## Error measures:
##                     ME     RMSE     MAE      MPE     MAPE      MASE
## Training set 9.166918 31.93101 26.7731 2.636328 13.39479 0.7987858
##                     ACF1
## Training set -0.1417817
##
## Forecasts:
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 31       239.5602 198.6390 280.4815 176.9766 302.1439
## 32       239.5602 196.4905 282.6299 173.6908 305.4297
## 33       239.5602 194.4443 284.6762 170.5613 308.5591
## 34       239.5602 192.4869 286.6336 167.5677 311.5527
```

```
optimal_alpha2 <- fit2$model$par["alpha"]
optimal_alpha2
```

```
##     alpha
## 0.3282696
```

```
method <- c( "Simple", "Optimal")
alphas <- c(optimal_alpha1, optimal_alpha2)
SSES <- c(fit1$model$SSE, sum(residuals(fit2) ^ 2))

final <- data.frame(method,alphas, SSES)
final
```

```
##    method    alphas     SSES
## 1  Simple 0.3473305 30758.07
## 2 Optimal 0.3282696 30587.69
```

3. For this exercise, use the quarterly UK passenger vehicle production data from 1977:1–2005:1 (data set ukcars).

a. Plot the data and describe the main features of the series.

From the decomposition figure, we can tell that there is upward trend regarding to the vehicle production. The upward trend stopped starting the year of 2000. In addition, the vehicle production has very strong seasonality correlation.

```
#install.packages("fpp")
library(fpp)
```

```
## Loading required package: expsmooth
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

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

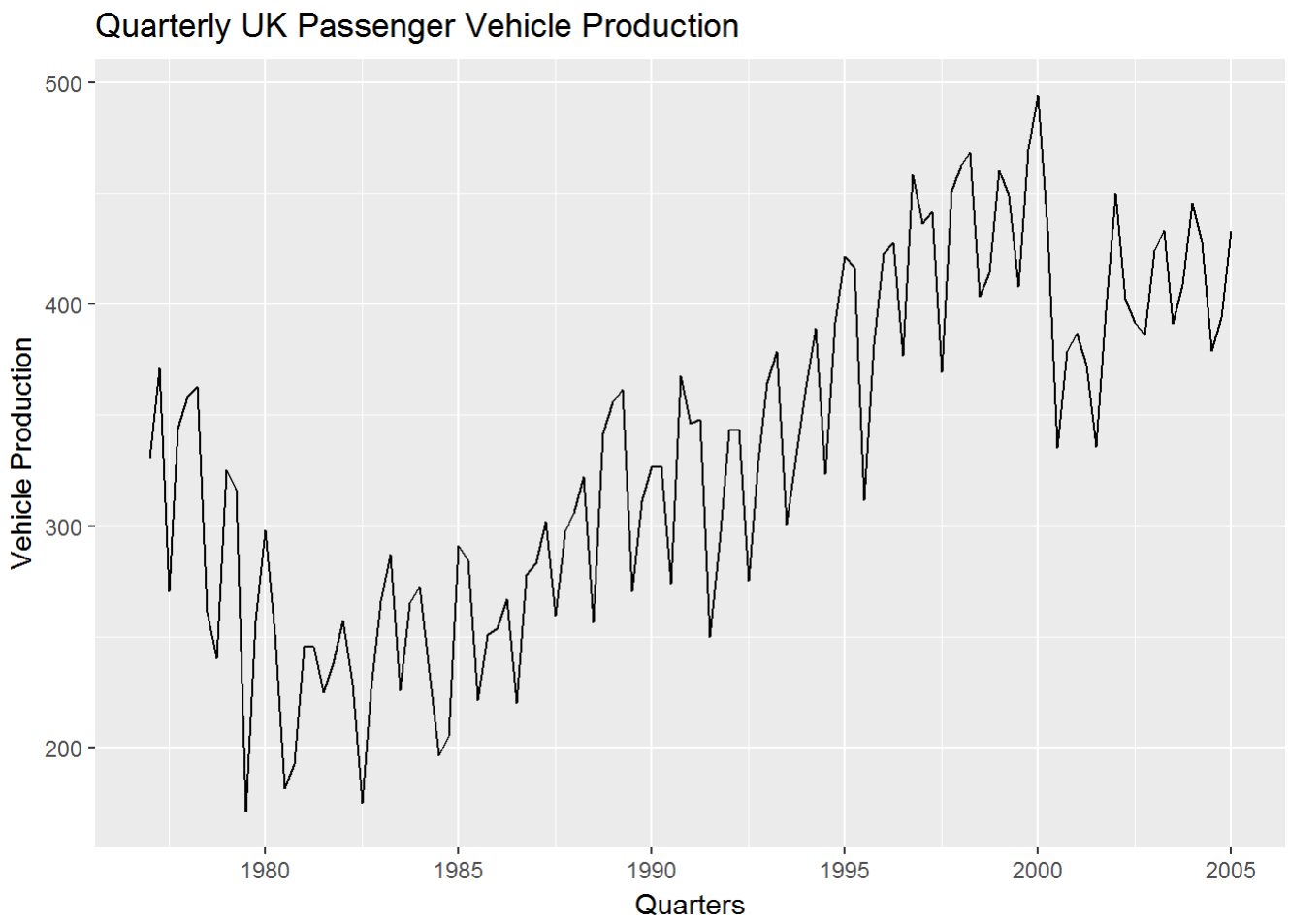
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: tseries
```

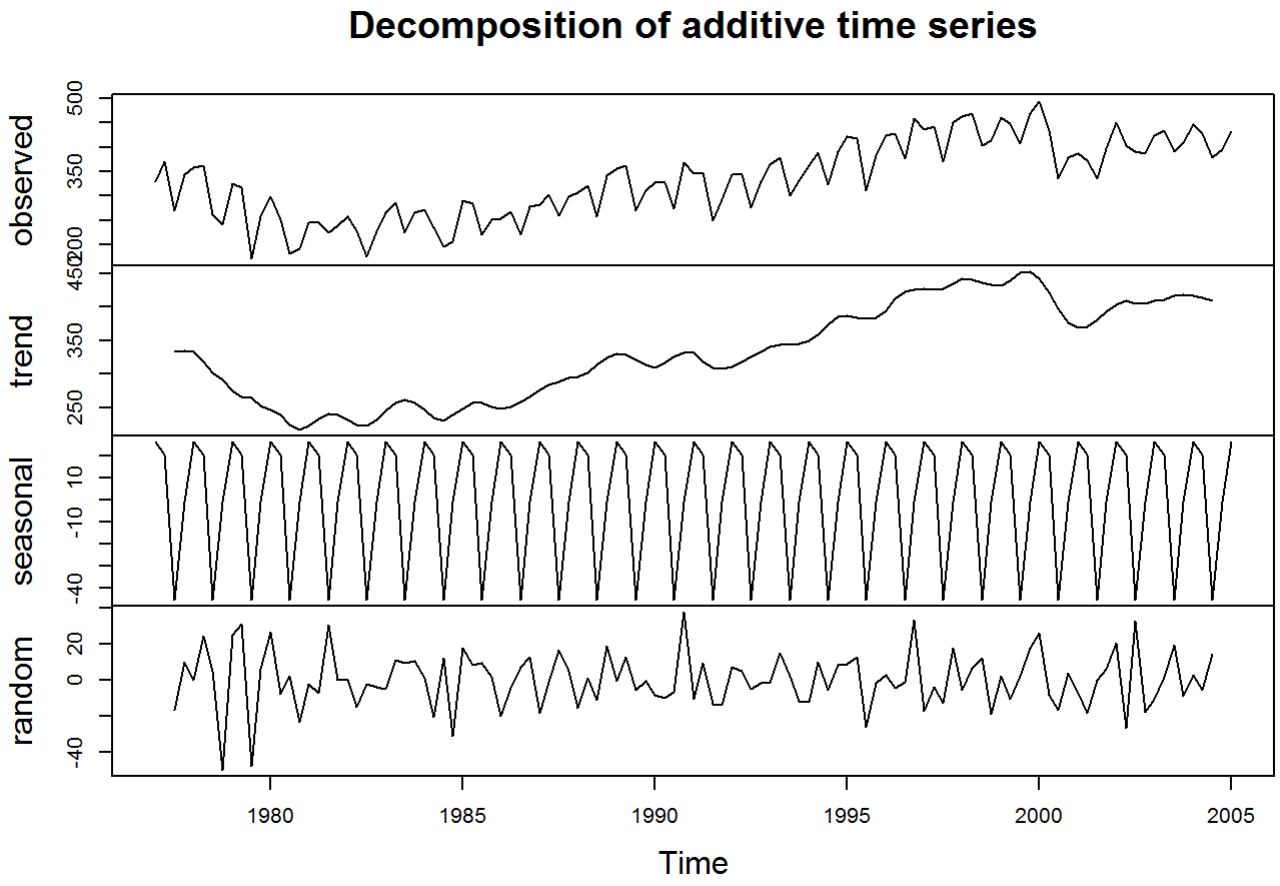
```
head(ukcars)
```

```
##         Qtr1    Qtr2    Qtr3    Qtr4
## 1977 330.371 371.051 270.670 343.880
## 1978 358.491 362.822
```

```
autoplot(ukcars, main ="Quarterly UK Passenger Vehicle Production", ylab = "Vehicle Production",
xlab = "Quarters")
```

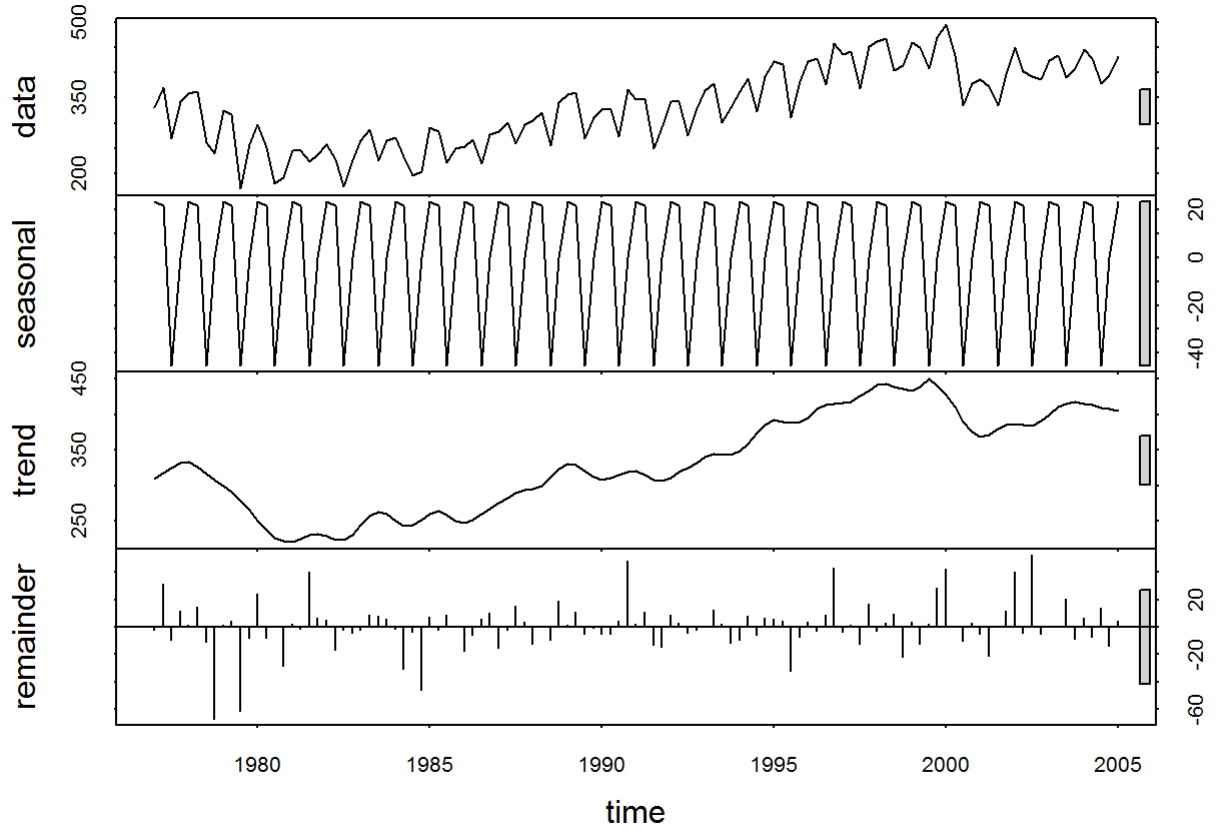



```
fit_decom <- decompose(ukcars)
plot(fit_decom)
```

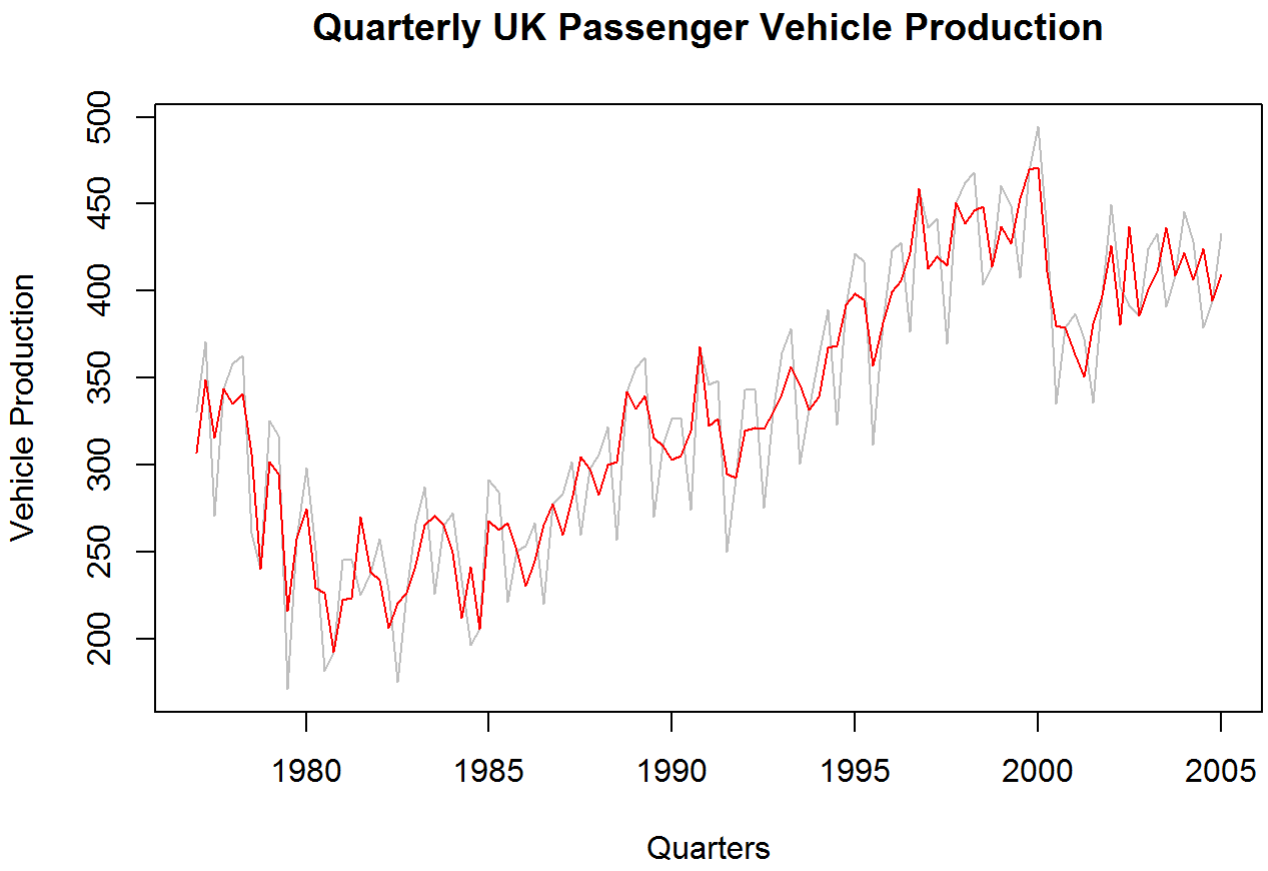


b. Decompose the series using STL and obtain the seasonally adjusted data.

```
fit <- stl(ukcars, s.window = "periodic", robust = TRUE)
plot(fit)
```




```
plot(ukcars, col="grey", main ="Quarterly UK Passenger Vehicle Production", ylab = "Vehicle Prod
uction", xlab = "Quarters")
lines(seasadj(fit), col="red", ylab="Seasonally adjusted")
```



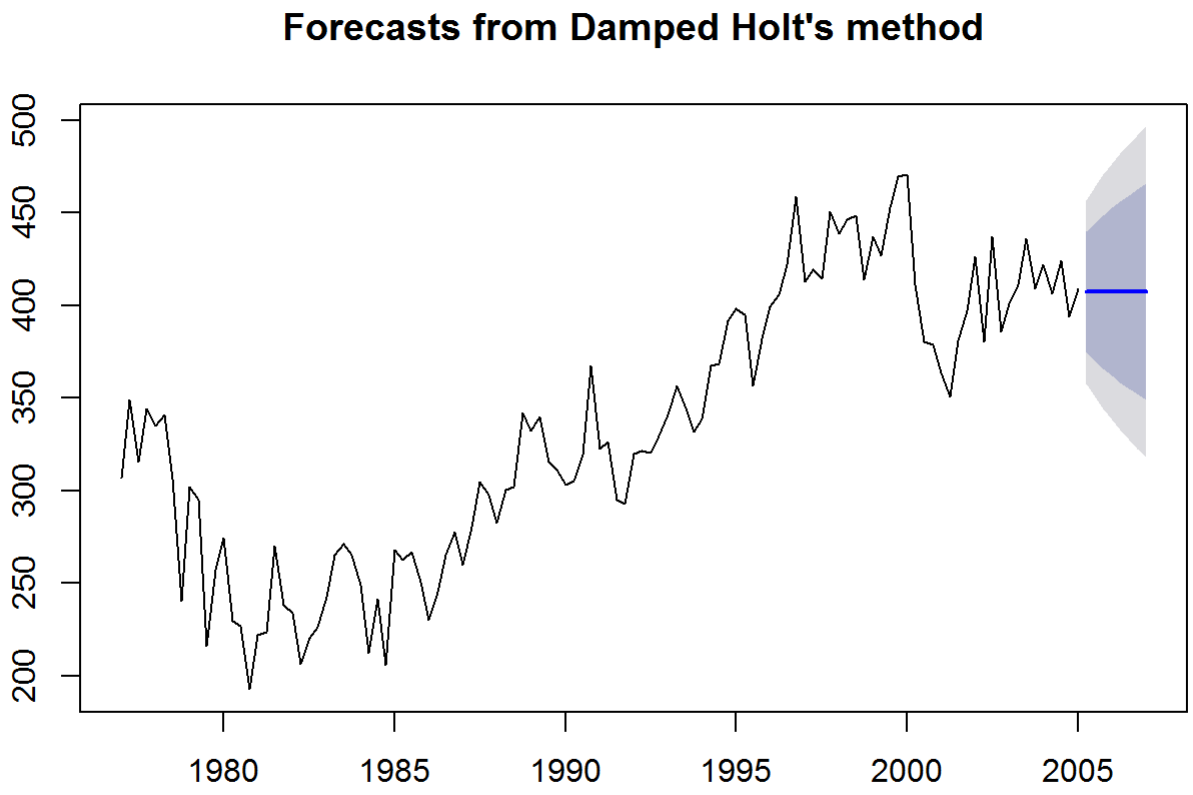
c. Forecast the next two years of the series using an additive damped trend method applied to the seasonally adjusted data. Then reseasonalize the forecasts. Record the parameters of the method and report the RMSE of the one-step forecasts from your method.

As showed in the following, I applied additive damped trend method to seasonally adjusted data. The RMSE is 25.20318. After the forecasts have been reseasonalized, the RMSE does not change.

```
fit2 <- holt(seasadj(fit), damped = TRUE, h = 8)
plot(fit2)
accuracy(fit2)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 2.518454 25.20318 20.5804 0.3038991 6.585405 0.6707052
##                    ACF1
## Training set 0.0353549
```

```
fit3 <- forecast(fit2, method = "naive")
plot(fit3)
```




```
accuracy(fit3)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 2.518454 25.20318 20.5804 0.3038991 6.585405 0.6707052
##                    ACF1
## Training set 0.0353549
```

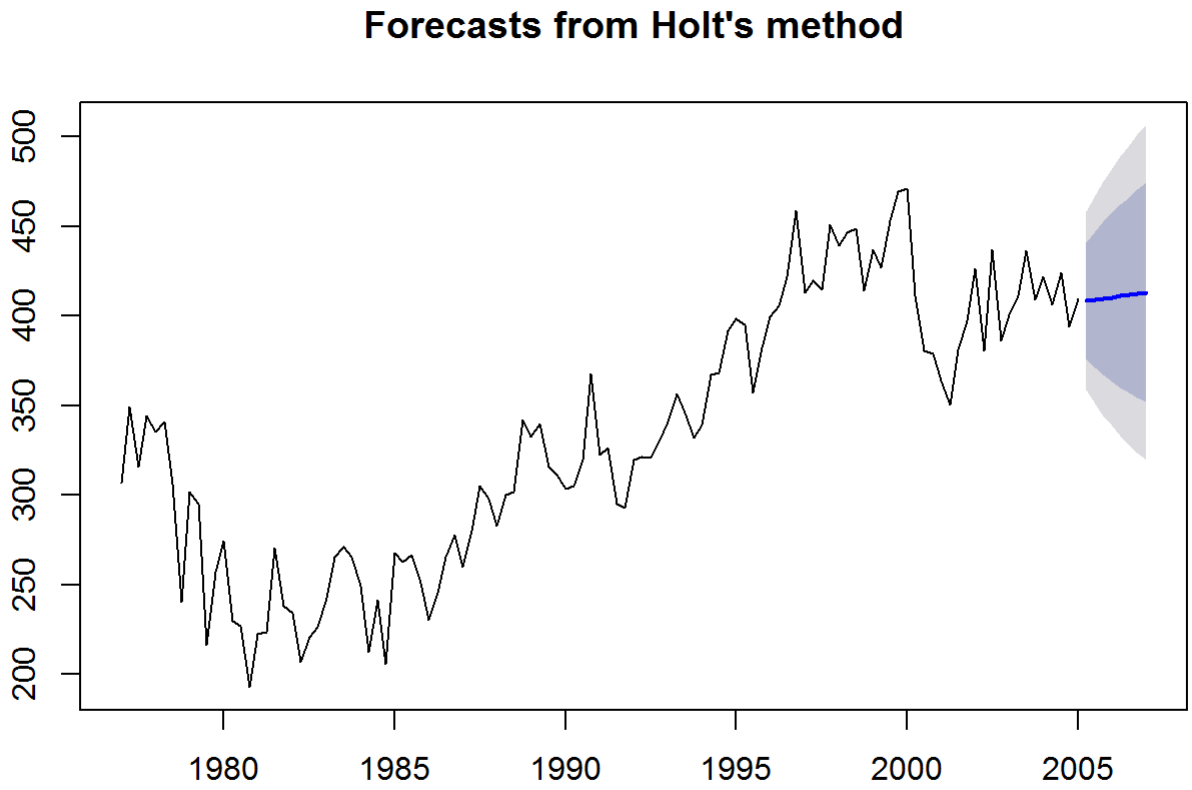
d. Forecast the next two years of the series using Holt's linear method applied to the seasonally adjusted data. Then reseasonalize the forecasts. Record the parameters of the method and report the RMSE of of the one-step forecasts from your method.

As showed in the following, I applied Holt's linear method to seasonally adjusted data. The RMSE is 25.39072. After the forecasts have been reseasonalized, the RMSE does not change.

```
fit4 <- holt(seasadj(fit), h = 8)
plot(fit4)
accuracy(fit4)
```

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1407116 25.39072 20.14514 -0.5931913 6.500319 0.6565204
##                     ACF1
## Training set 0.02953472
```

```
fit5 <- forecast(fit4, method = "naive")
plot(fit5)
```




```
accuracy(fit5)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1407116 25.39072 20.14514 -0.5931913 6.500319 0.6565204
##                    ACF1
## Training set 0.02953472
```

e. Now use ets() to choose a seasonal model for the data.

Approaches: There exist two tyoes of models one with additive errors and one with multiplicative errors. Each model also consists of three components or states: error, trend, seasonal. Possibilities for each component are: Error ={A,M}, Trend ={N,A,Ad,M,Md}, and Seasonal ={N,A,M}. Therefore, there are total 30 state space models. Function ets can be used to estimate the models just by customerizing the arguments.
Interpretation: According to the statistical summary, the estimated model is ETS(A,N,A) with RMSE value to be 25.25792. Model ETS(A,N,A) means this model had additive errors, and additive seasonalityl, without trend.

```
fit6 <- ets(ukcars)
summary(fit6)
```

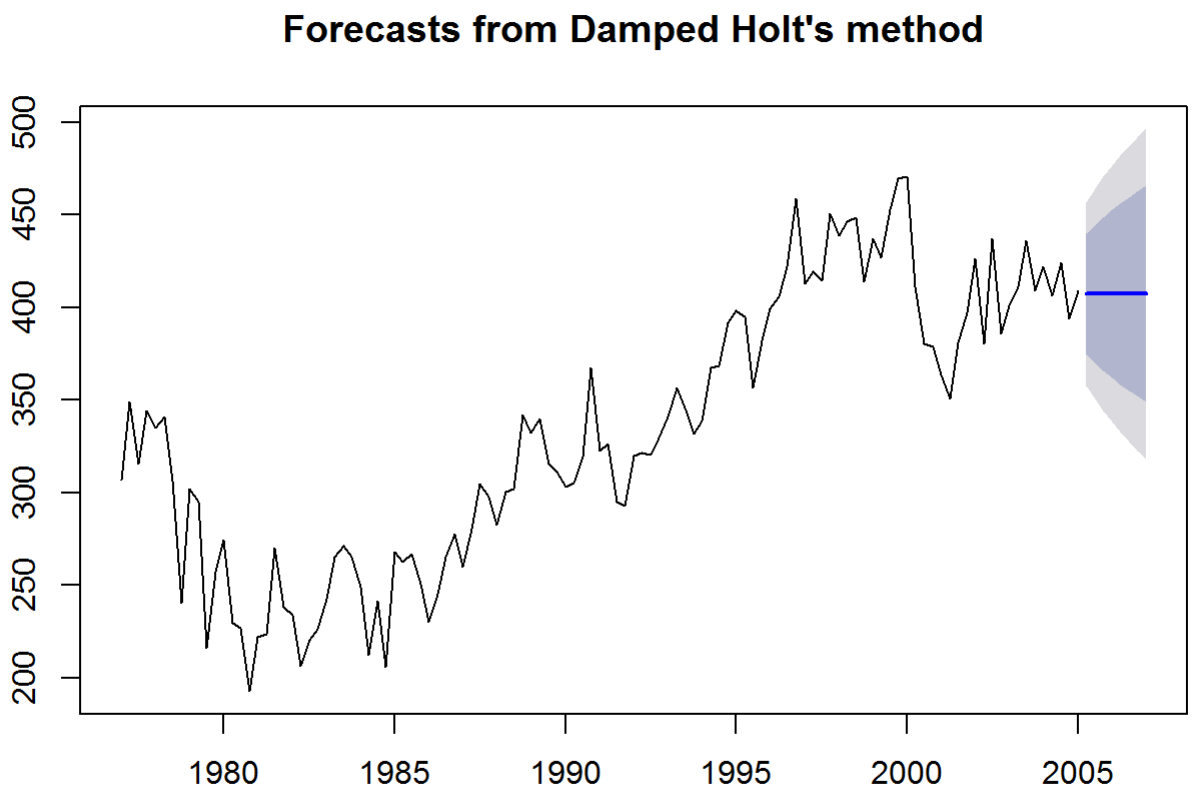
```
## ETS(A,N,A)
##
## Call:
##  ets(y = ukcars)
##
##   Smoothing parameters:
##     alpha = 0.6267
##     gamma = 1e-04
##
##   Initial states:
##     l = 313.0916
##     s=-1.1271 -44.606 21.5553 24.1778
##
##   sigma:  25.2579
##
##      AIC     AICc      BIC
## 1277.980 1279.047 1297.072
##
## Training set error measures:
##                     ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 1.324962 25.25792 20.12508 -0.1634983 6.609629 0.6558666
##                     ACF1
## Training set 0.01909295
```

f. Compare the RMSE of the fitted model with the RMSE of the model you obtained using an STL decomposition with Holt's method. Which gives the better in-sample fits?

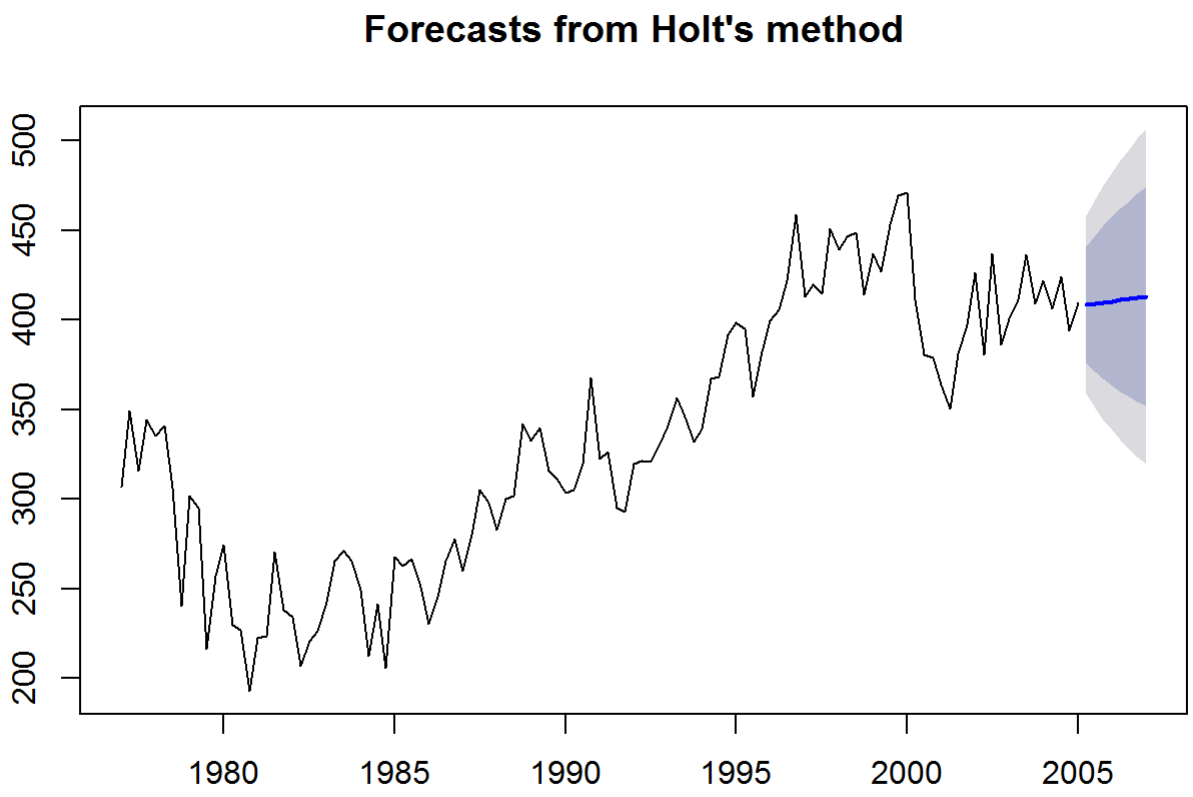
After we using additive damped trend method, Holt's linear method, and ets, we obtain three RMSE values. They are 25.20318, 25.39072, and 25.25792 respectively. All of them are not identical, but they are very closed to each other, which deem three models have similar accuracy to forecast the future vehicle productions.

g. Compare the forecasts from the two approaches? Which seems most reasonable?

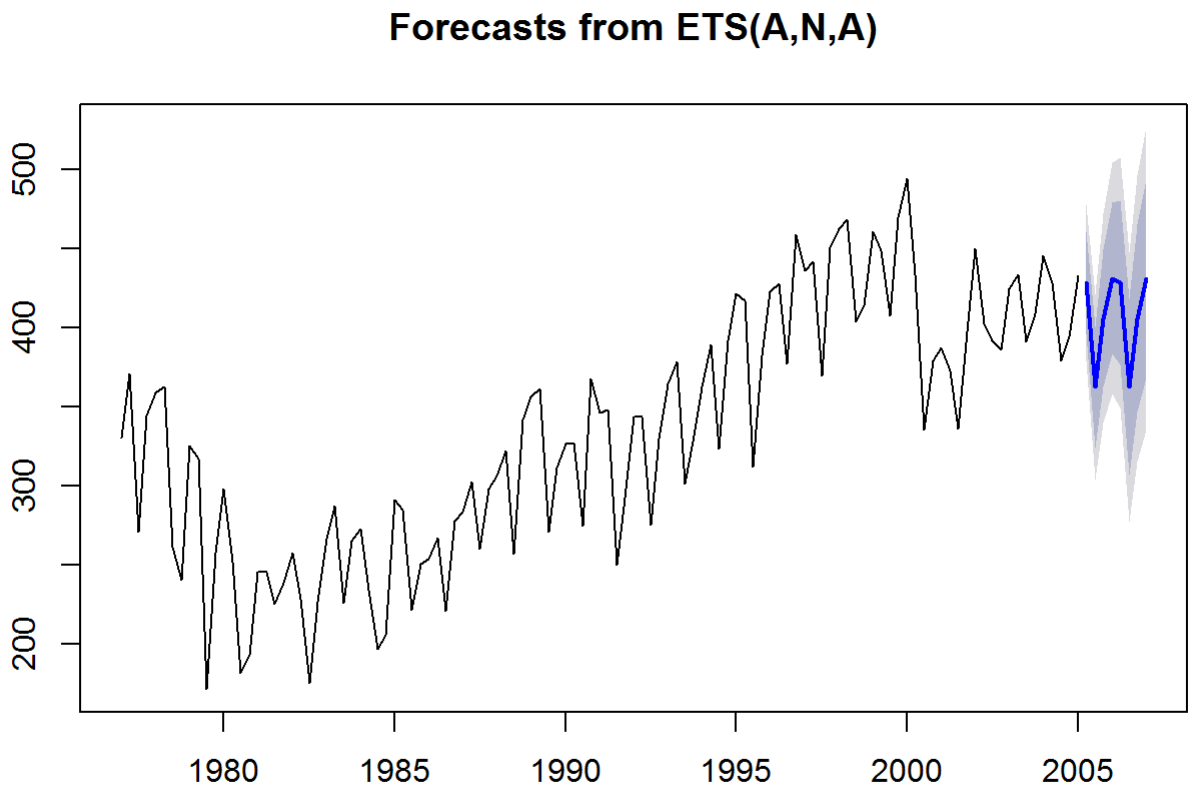
```
plot(fit2)
```


Forecasts from Damped Holt's method
200
250
300
350
400
450
500
1980
1985
1990
1995
2000
2005

```
plot(fit4)
```




```
plot(forecast(fit6,  h = 8))
```



I think ETS(A,N,A) is more reasonable, because it ocillate throughout the year which is a better presentation for the seasonality of the vehicle productions.