BHao\_HW4

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

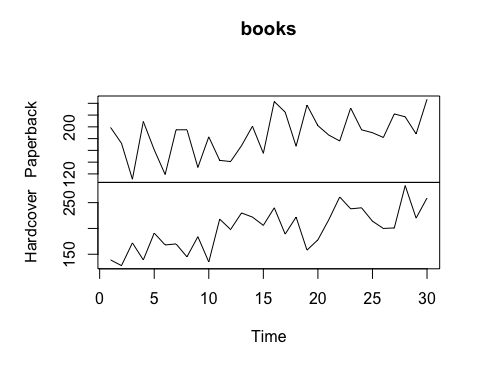
The data represent daily unit sales for paperback and hardcover books at the same store over 30 days. Optically, it appears that sales trend upward, but there are no obvious patterns in day-to-day sales.

library(fma)

## Loading required package: forecast

## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'zone/tz/2018c.  
## 1.0/zoneinfo/America/Los\_Angeles'

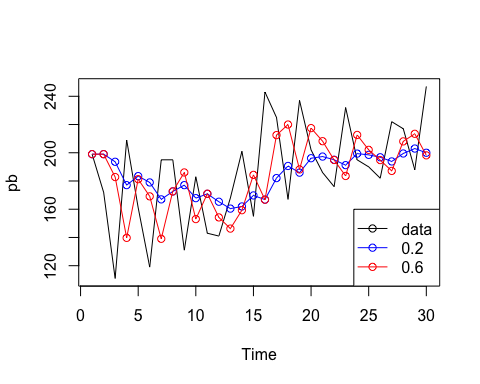
plot(books)



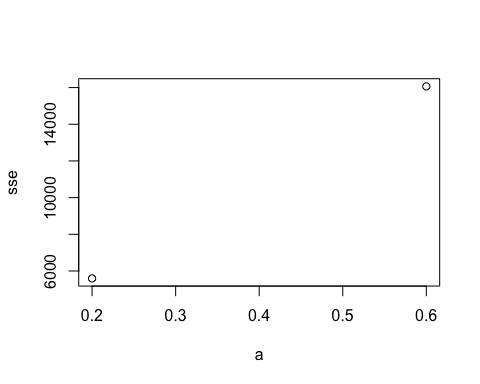
## 7.8.1.b. Use simple exponential smoothing with the ses function (setting initial="simple") and explore different values of αα for the paperback series. Record the within-sample SSE for the one-step forecasts. Plot SSE against αα and find which value of αα works best. What is the effect of αα on the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

In this case, the smaller alpha value 0.2 performed much better in terms of SSE than did the larger alpha value 0.6.

pb = books[, 'Paperback']  
  
fit1 = ses(pb, alpha = 0.2, initial = 'simple')  
fit2 = ses(pb, alpha = 0.6, initial = 'simple')  
plot(pb)  
lines(fitted(fit1), col='blue', type='o')  
lines(fitted(fit2), col='red' , type='o')  
legend('bottomright', lty=1, col=c(1,'blue','red'),  
 c('data', expression(a = 0.2), expression(a = 0.6)),  
 pch=1)



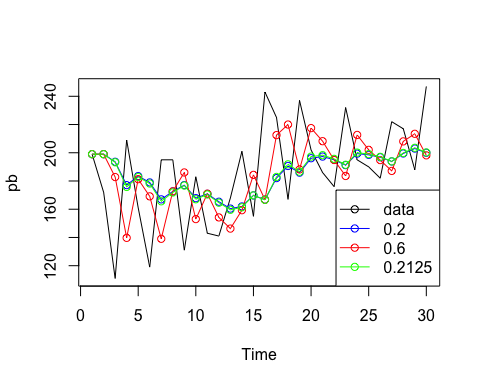
sse1 = sum((fitted(fit1) - mean(pb))^2)  
sse2 = sum((fitted(fit2) - mean(pb))^2)  
  
plot(data.frame(a = c(0.2, 0.6), sse = c(sse1, sse2)))



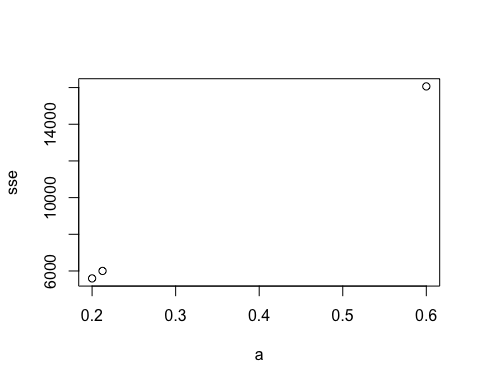
## 7.8.1.c. Now let ses select the optimal value of αα. Use this value to generate forecasts for the next four days. Compare your results with 2.

In this case, ses selected 0.1685 as the optimal alpha value; however, the SSE was still higher than the 0.2 alpha value above.

fit3 = ses(pb, initial = 'simple')  
  
plot(pb)  
lines(fitted(fit1), col='blue' , type='o')  
lines(fitted(fit2), col='red' , type='o')  
lines(fitted(fit3), col='green', type='o')  
legend('bottomright', lty=1, col=c(1,'blue','red','green'),  
 c('data', expression(a = 0.2), expression(a = 0.6), expression(a = 0.2125)),  
 pch=1)



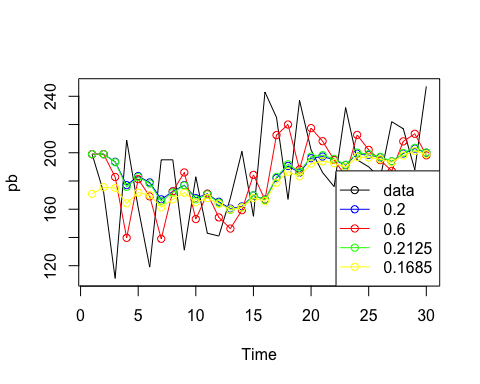
sse3 = sum((fitted(fit3) - mean(pb))^2)  
  
plot(data.frame(a = c(0.2, 0.6, 0.2125), sse = c(sse1, sse2, sse3)))



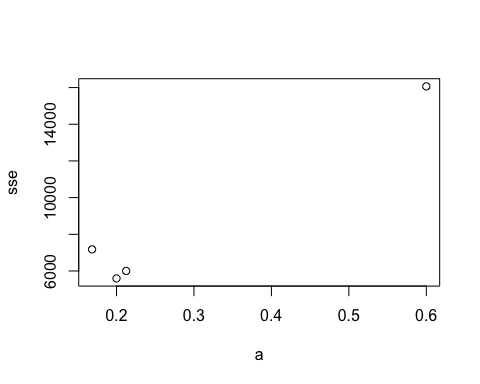
## 7.8.1.d. Repeat but with initial="optimal". How much difference does an optimal initial level make?

Interestingly, allowing ses to select the optimal initial value, the resulting SSE was higher than than above.

fit4 = ses(pb, initial = 'optimal')  
  
plot(pb)  
lines(fitted(fit1), col='blue' , type='o')  
lines(fitted(fit2), col='red' , type='o')  
lines(fitted(fit3), col='green' , type='o')  
lines(fitted(fit4), col='yellow', type='o')  
legend('bottomright', lty=1, col=c(1,'blue','red','green','yellow'),  
 c('data', expression(a = 0.2), expression(a = 0.6), expression(a = 0.2125), expression(a = 0.1685)),  
 pch=1)



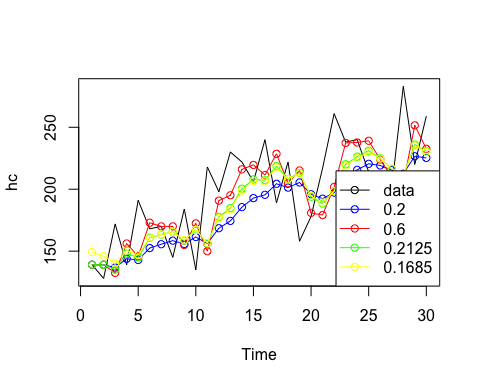
sse4 = sum((fitted(fit4) - mean(pb))^2)  
  
plot(data.frame(a = c(0.2, 0.6, 0.2125, 0.1685), sse = c(sse1, sse2, sse3, sse4)))



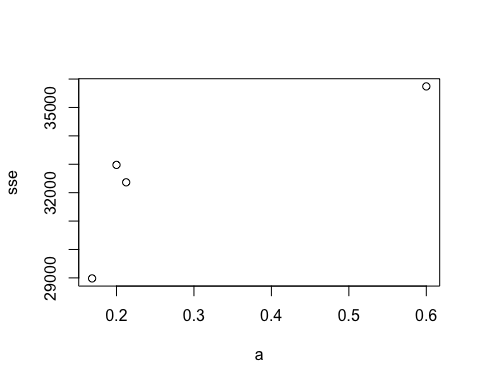
## 7.8.1.e Repeat steps (b)–(d) with the hardcover series.

In the case of hardcovers, allowing ses to select the optimal initial and alpha values resulted in the lowest SSE by far.

hc = books[, 'Hardcover']  
  
fit1 = ses(hc, alpha = 0.2, initial = 'simple')  
fit2 = ses(hc, alpha = 0.6, initial = 'simple')  
fit3 = ses(hc, initial = 'simple')  
fit4 = ses(hc, initial = 'optimal')  
  
plot(hc)  
lines(fitted(fit1), col='blue' , type='o')  
lines(fitted(fit2), col='red' , type='o')  
lines(fitted(fit3), col='green' , type='o')  
lines(fitted(fit4), col='yellow', type='o')  
legend('bottomright', lty=1, col=c(1,'blue','red','green','yellow'),  
 c('data', expression(a = 0.2), expression(a = 0.6), expression(a = 0.2125), expression(a = 0.1685)),  
 pch=1)



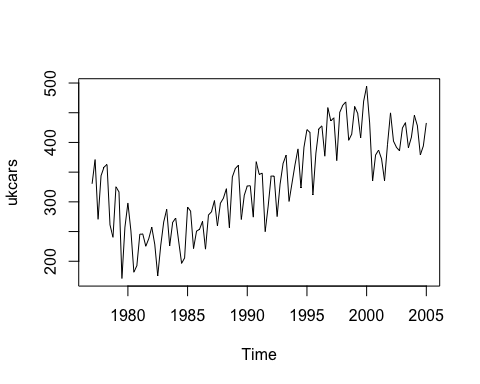
sse1 = sum((fitted(fit1) - mean(hc))^2)  
sse2 = sum((fitted(fit2) - mean(hc))^2)  
sse3 = sum((fitted(fit3) - mean(hc))^2)  
sse4 = sum((fitted(fit4) - mean(hc))^2)  
  
plot(data.frame(a = c(0.2, 0.6, 0.2125, 0.1685), sse = c(sse1, sse2, sse3, sse4)))



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

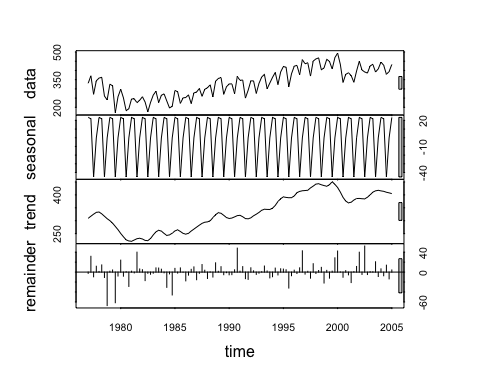
There is a clear upward trend in the data over time as well as clear seasonality.

library(expsmooth)  
plot(ukcars)



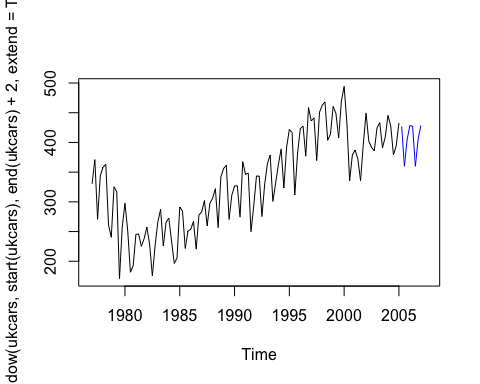
## 7.8.3.b. Decompose the series using STL and obtain the seasonally adjusted data.

fit\_stl = stl(ukcars, s.window = 'periodic', robust = TRUE)  
plot(fit\_stl)



## 7.8.3.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.

ukcars\_seasAdj = fit\_stl$time.series[, 'trend']  
reSeas = as.numeric(tail(fit\_stl$time.series[, 'seasonal'], 8)) # use prev 8 periods to reseasonalize forecast   
  
fit\_damped = ses(ukcars\_seasAdj, h = 8, damped = TRUE)  
fit\_damped\_reSeas = fit\_damped$mean + reSeas # combine with forecasted mean   
  
plot(window(ukcars, start(ukcars), end(ukcars) + 2, extend = TRUE)) # extend window to include full forecast   
lines(fit\_damped\_reSeas, col='blue')

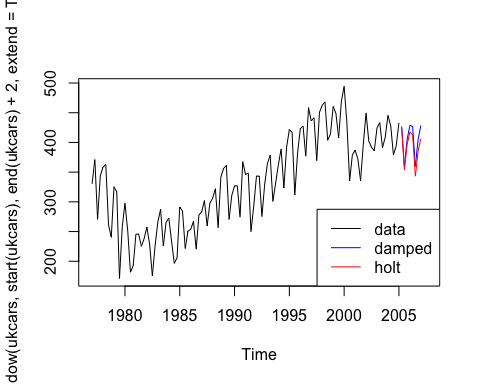


# output model parameters   
summary(fit\_damped)

##   
## Forecast method: Simple exponential smoothing  
##   
## Model Information:  
## Simple exponential smoothing   
##   
## Call:  
## ses(y = ukcars\_seasAdj, h = 8, damped = TRUE)   
##   
## Smoothing parameters:  
## alpha = 0.9999   
##   
## Initial states:  
## l = 309.5501   
##   
## sigma: 7.5104  
##   
## AIC AICc BIC   
## 995.8765 996.0967 1004.0587   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.8455707 7.510414 6.240617 0.2092296 1.972227 0.3181851  
## ACF1  
## Training set 0.766302  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2005 Q2 405.09 395.4650 414.7150 390.3698 419.8101  
## 2005 Q3 405.09 391.4789 418.7011 384.2736 425.9064  
## 2005 Q4 405.09 388.4201 421.7598 379.5957 430.5843  
## 2006 Q1 405.09 385.8415 424.3385 375.6519 434.5281  
## 2006 Q2 405.09 383.5696 426.6104 372.1774 438.0026  
## 2006 Q3 405.09 381.5157 428.6643 369.0362 441.1438  
## 2006 Q4 405.09 379.6269 430.5531 366.1475 444.0325  
## 2007 Q1 405.09 377.8688 432.3112 363.4588 446.7212

## 7.8.3.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.

fit\_holt = holt(ukcars\_seasAdj, h = 8)  
fit\_holt\_reSeas = fit\_holt$mean + reSeas   
  
plot(window(ukcars, start(ukcars), end(ukcars) + 2, extend = TRUE))  
lines(fit\_damped\_reSeas, col='blue')  
lines(fit\_holt\_reSeas , col='red')  
legend('bottomright', lty=1, col=c(1,'blue','red'),  
 c('data', 'damped', 'holt'))



# output model parameters   
summary(fit\_holt)

##   
## Forecast method: Holt's method  
##   
## Model Information:  
## Holt's method   
##   
## Call:  
## holt(y = ukcars\_seasAdj, h = 8)   
##   
## Smoothing parameters:  
## alpha = 0.9999   
## beta = 0.9999   
##   
## Initial states:  
## l = 299.8252   
## b = 14.2267   
##   
## sigma: 5.0564  
##   
## AIC AICc BIC   
## 910.4623 911.0231 924.0993   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -0.1504732 5.056389 3.901235 -0.0008004432 1.218107 0.1989089  
## ACF1  
## Training set 0.3659724  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2005 Q2 402.3147 395.8347 408.7947 392.4044 412.2251  
## 2005 Q3 399.5397 385.0511 414.0283 377.3813 421.6981  
## 2005 Q4 396.7647 372.5210 421.0084 359.6872 433.8422  
## 2006 Q1 393.9897 358.5007 429.4787 339.7140 448.2654  
## 2006 Q2 391.2147 343.1625 439.2669 317.7251 464.7042  
## 2006 Q3 388.4397 326.6306 450.2487 293.9108 482.9685  
## 2006 Q4 385.6646 308.9999 462.3293 268.4161 502.9132  
## 2007 Q1 382.8896 290.3460 475.4332 241.3564 524.4228

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

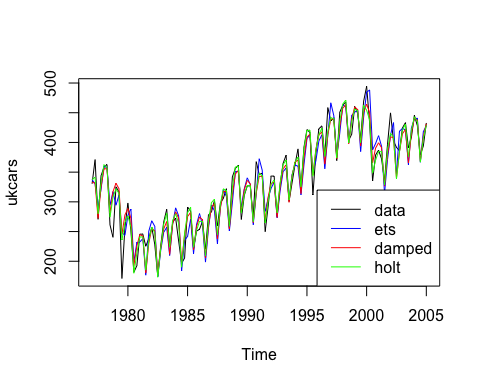
fit\_ets = ets(ukcars)  
  
# output model parameters  
summary(fit\_ets)

## 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

## 7.8.3.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?

The linear Holt method produced the lowest in-sample RMSE.

plot(ukcars)  
lines(fitted(fit\_ets), col='blue')  
# although the below two comparisons are not exactly fair since they are using the actual seasonality estimates   
lines(fitted(fit\_damped) + fit\_stl$time.series[, 'seasonal'], col='red')  
lines(fitted(fit\_holt) + fit\_stl$time.series[, 'seasonal'], col='green')  
legend('bottomright', lty=1, col=c(1,'blue','red','green'),  
 c('data', 'ets', 'damped', 'holt'))



data.frame(c('ets', 'damped', 'holt'),   
 c(sqrt(fit\_ets$mse), sqrt(fit\_damped$model$mse), sqrt(fit\_holt$model$mse)))

## c..ets....damped....holt..  
## 1 ets  
## 2 damped  
## 3 holt  
## c.sqrt.fit\_ets.mse...sqrt.fit\_damped.model.mse...sqrt.fit\_holt.model.mse..  
## 1 25.257919  
## 2 7.510414  
## 3 5.056389

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

The ets and damped forecasts are right on top of one another. The Holt model forecasts slightly lower values. It's not clear which is the most reasonable, as they all appear quite reasonable.

plot(forecast(fit\_ets, h = 8))  
lines(fit\_damped\_reSeas, col='red')  
lines(fit\_holt\_reSeas , col='green')  
legend('bottomright', lty=1, col=c(1,'blue','red','green'),  
 c('data', 'ets', 'damped', 'holt'))

