BHao\_HW2

library(fma)  
# data(package='fma')

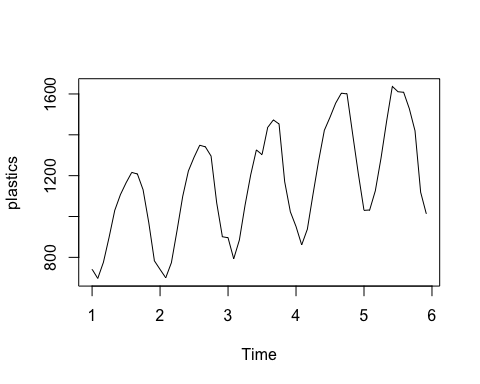
## 6.7.2 The data below represent the monthly sales (in thousands) of product A for a plastics manufacturer for years 1 through 5 (data set plastics).

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

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

plot(plastics)



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

fit = decompose(plastics, type='multiplicative')  
  
# seasonal indices   
fit$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

# trend-cycle indices   
fit$trend

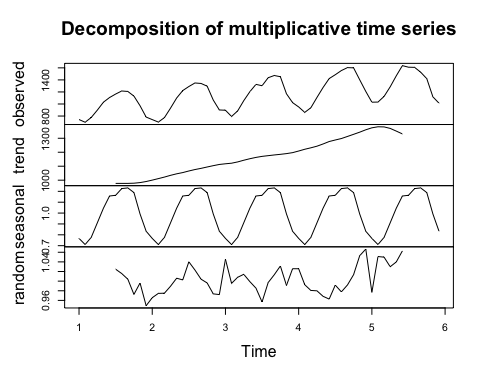
## Jan Feb Mar Apr May Jun Jul  
## 1 NA NA NA NA NA NA 976.9583  
## 2 1000.4583 1011.2083 1022.2917 1034.7083 1045.5417 1054.4167 1065.7917  
## 3 1117.3750 1121.5417 1130.6667 1142.7083 1153.5833 1163.0000 1170.3750  
## 4 1208.7083 1221.2917 1231.7083 1243.2917 1259.1250 1276.5833 1287.6250  
## 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

## 6.7.2.c Do the results support the graphical interpretation from part (a)?

Yes, the original data exhibits an upward trend with annual seasons, and the classical decomposition captures that.

## 6.7.2.d Compute and plot the seasonally adjusted data.

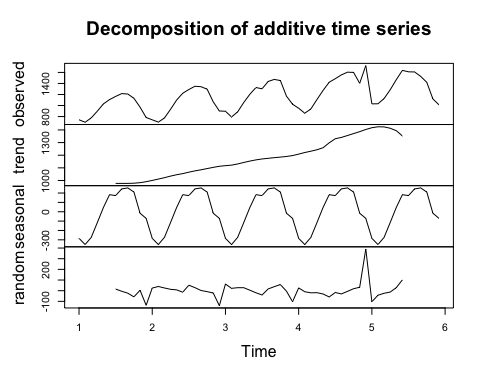
# plot of decomposition   
plot(fit)



## 6.7.2.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?

The outlier doesn't seem to affect the trend-cycle component, but it certainly changes the magnitudes of the seasonal and random components (although the shape of the seasonal component still looks reasonably the same).

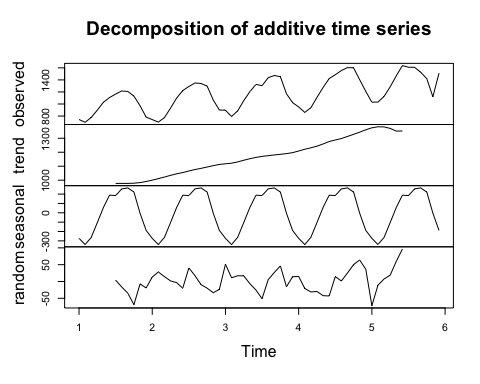
set.seed(123)  
plastics2 = plastics   
plastics2[sample.int(60, 1)] = plastics2[sample.int(60, 1)] + 500   
  
# plot decomposition   
fit2 = decompose(plastics2)  
plot(fit2)



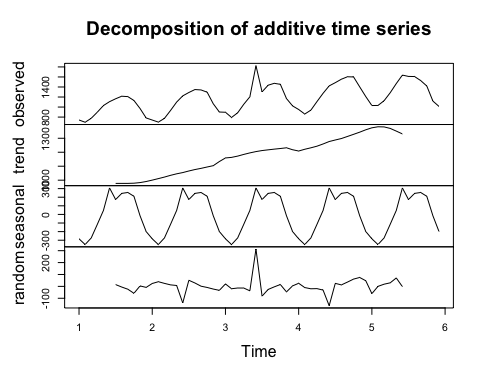
## 6.7.2.f Does it make any difference if the outlier is near the end rather than in the middle of the time series?

The location of the outlier does not seem to make much difference. In both cases, the outlier causes the seasonal and random components to be much larger than when no outlier exists.

plastics\_end = plastics  
plastics\_end[60] = plastics\_end[60] + 500  
  
plastics\_mid = plastics  
plastics\_mid[30] = plastics\_mid[30] + 500  
  
# plot decomposition   
fit\_end = decompose(plastics\_end)  
plot(fit\_end)

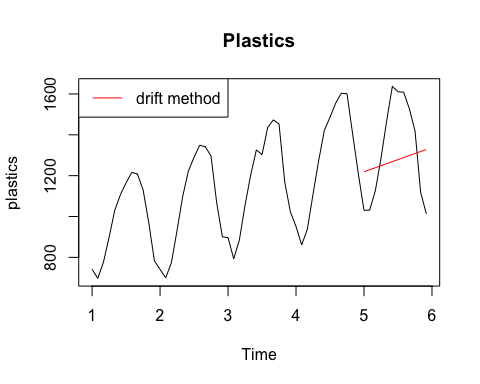


fit\_mid = decompose(plastics\_mid)  
plot(fit\_mid)



## 6.7.2.g Use a random walk with drift to produce forecasts of the seasonally adjusted data.

# create training and test sets   
plastics\_train = window(plastics, start=1, end=4.99)  
plastics\_test = window(plastics, start=5)  
  
# fit random walk with drift on training set   
plastics\_driftfit = rwf(plastics\_train, h = 12, drift = TRUE)  
  
# plot   
plot(plastics, main = 'Plastics')  
lines(plastics\_driftfit$mean, col = 2)  
legend('topleft', lty = 1, col = c(2),  
 legend = c('drift method'))



## 6.7.2.h Reseasonalize the results to give forecasts on the original scale.

# multiply by seasonality index   
plastics\_driftfitSeas = plastics\_driftfit$mean \* fit$seasonal[49:60]   
  
# plot   
plot(plastics, main = 'Plastics')  
lines(plastics\_driftfit$mean, col = 2)  
lines(plastics\_driftfitSeas , col = 3)  
legend('topleft', lty = 1, col = c(2, 3),  
 legend = c('drift method', 'drift with seasonality'))

