airline model(Kalman filter)

LinyiGuo

2019/8/8

Import Data

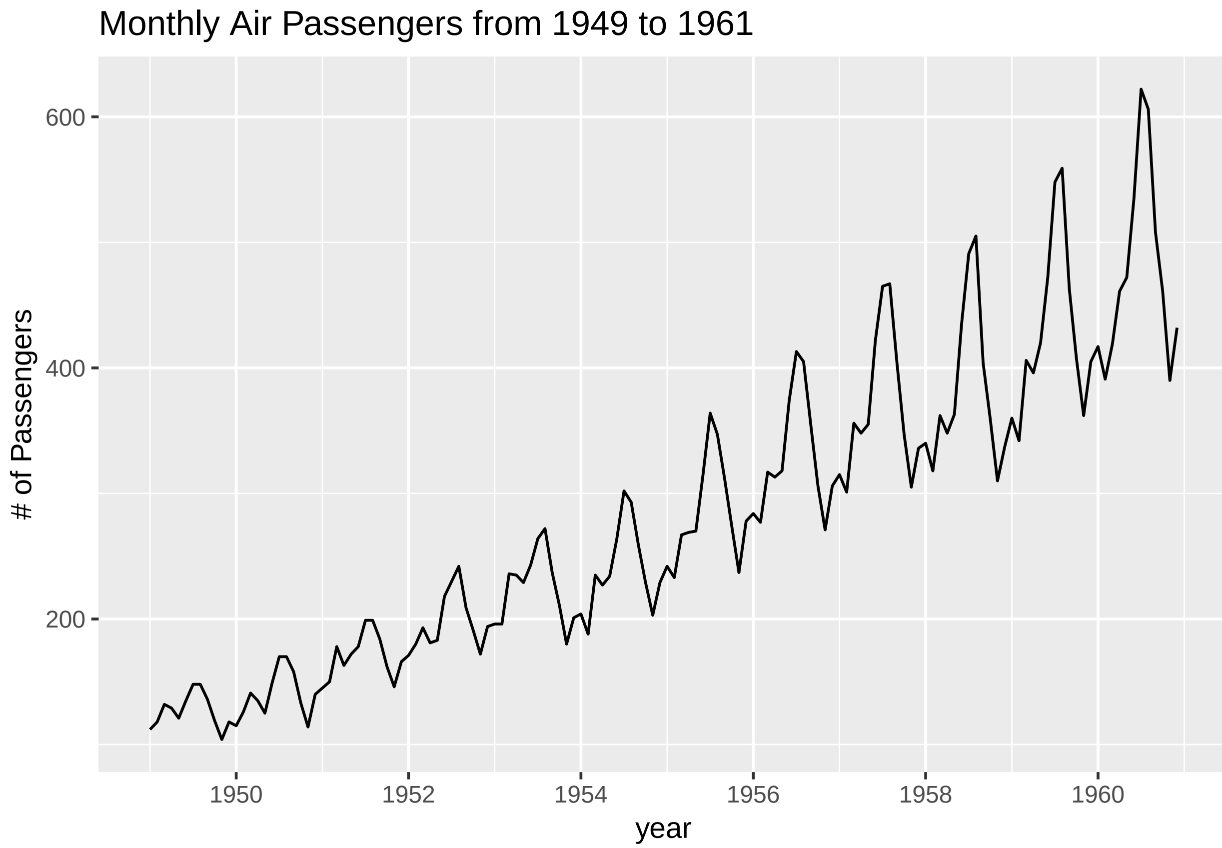
```
library(KFAS)
library(ggfortify)
```

```
## Loading required package: ggplot2

## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

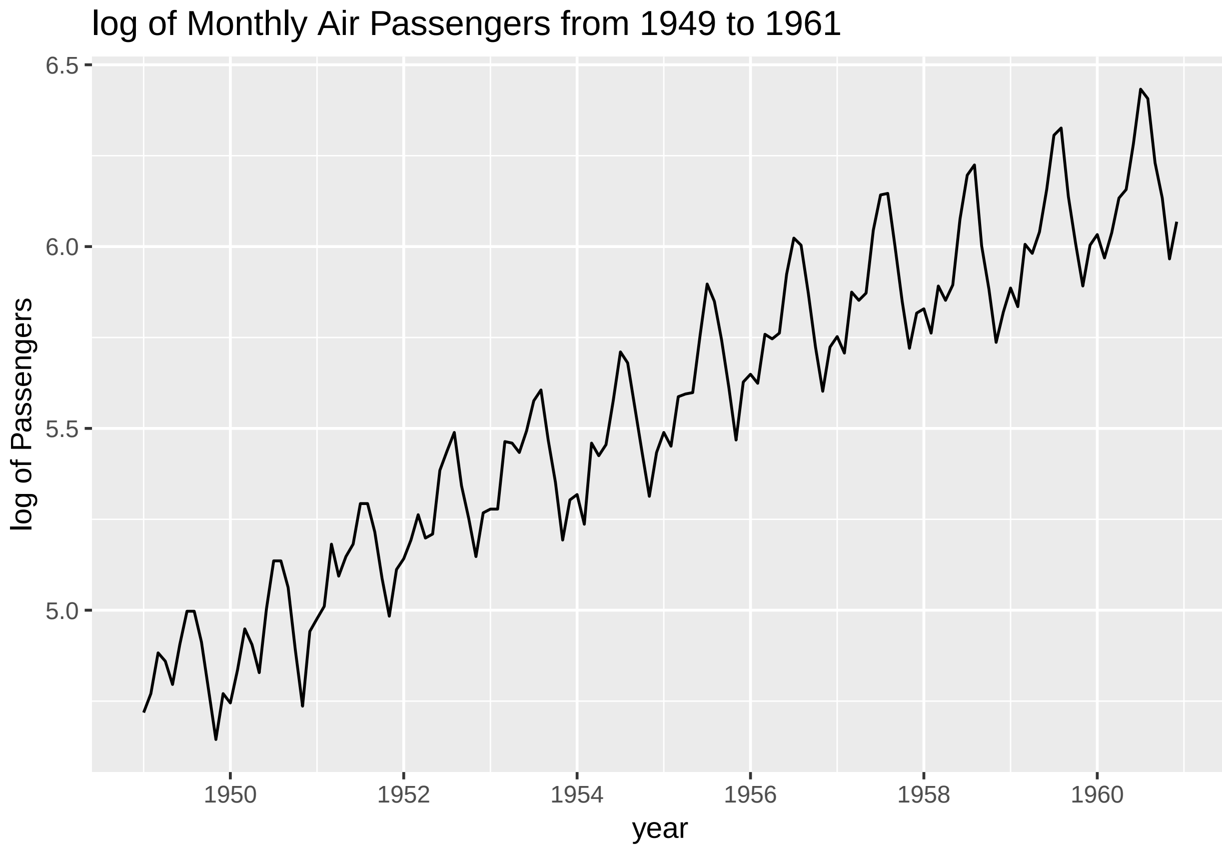
```
library(forecast)
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':
##   method                 from
##   autoplot.Arima         ggfortify
##   autoplot.acf           ggfortify
##   autoplot.ar            ggfortify
##   autoplot.bats          ggfortify
##   autoplot.decomposed.ts ggfortify
##   autoplot.ets           ggfortify
##   autoplot.forecast      ggfortify
##   autoplot.stl           ggfortify
##   autoplot.ts            ggfortify
##   fitted.ar              ggfortify
##   fitted.fracdiff        fracdiff
##   fortify.ts             ggfortify
##   residuals.ar           ggfortify
##   residuals.fracdiff     fracdiff
```

```
data("AirPassengers")
data_ap_log <- log(AirPassengers)
autoplot(AirPassengers)+labs(x = "year", y = "# of Passengers",
                             title = "Monthly Air Passengers from 1949 to 1961")
```



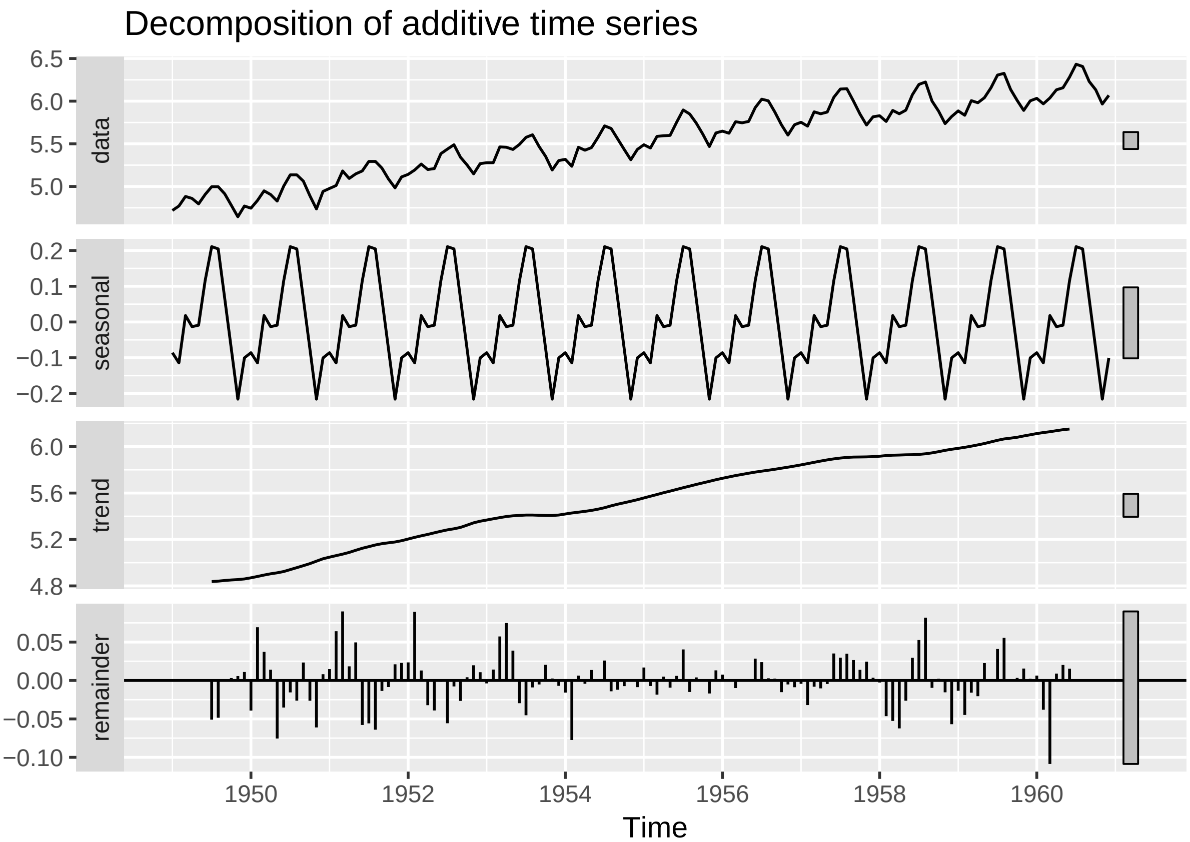

```
autoplot(data_ap_log)+labs(x = "year", y = "log of Passengers",
                             title = "log of Monthly Air Passengers from 1949 to 1961")
```



Again, since the difference is larger as time goes on, we take log first.

Results from ARIMA

```
decompose_ap1 <- decompose(data_ap_log, "additive")
autoplot(decompose_ap1)
```



Build SSModel and Apply Kalman Filter

According to paper 3.5, **KFAS** follows closely Durbin and Koopman's book and papers.

```
# create one SSModel object
mod_ap <- SSModel(data_ap_log~ SSMtrend(1, Q=list(1)) +
                    SSMseasonal(period = 12, sea.type = "dummy"),
                    H = 1)  # dummy here means we don't use trigonometric expression
print(mod_ap)
```

```
## Call:
## SSModel(formula = data_ap_log ~ SSMtrend(1, Q = list(1)) + SSMseasonal(period = 12,
##     sea.type = "dummy"), H = 1)
##
## State space model object of class SSModel
##
## Dimensions:
## [1] Number of time points: 144
## [1] Number of time series: 1
## [1] Number of disturbances: 1
## [1] Number of states: 12
## Names of the states:
##  [1]  level         sea_dummy1   sea_dummy2   sea_dummy3   sea_dummy4
##  [6]  sea_dummy5    sea_dummy6   sea_dummy7   sea_dummy8   sea_dummy9
```

```
## [11]  sea_dummy10  sea_dummy11
## Distributions of the time series:
## [1]  gaussian
##
## Object is a valid object of class SSModel.
```

```
# apply kalman filter
out_ap <- KFS(mod_ap, filtering = "state")
print(out_ap)
```

```
## Smoothed values of states and standard errors at time n = 144:
##              Estimate  Std. Error
## level         6.17782   0.83878
## sea_dummy1   -0.10274   0.40329
## sea_dummy2   -0.21733   0.40185
## sea_dummy3   -0.07434   0.40070
## sea_dummy4    0.06309   0.39983
## sea_dummy5    0.20700   0.39925
## sea_dummy6    0.21557   0.39896
## sea_dummy7    0.11091   0.39896
## sea_dummy8   -0.01196   0.39925
## sea_dummy9   -0.01031   0.39983
## sea_dummy10   0.02023   0.40070
## sea_dummy11  -0.11072   0.40185
```

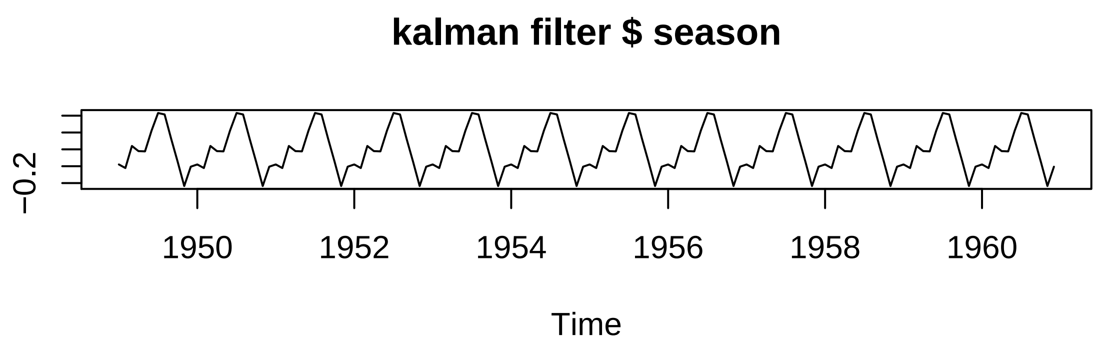
```
# get estimates of conditional means of states
coef(out_ap, states = "trend")
```

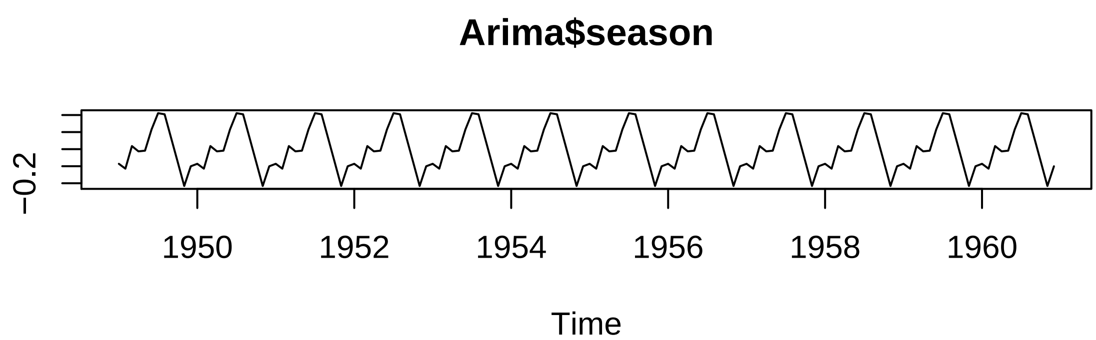
```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 1949 4.832206 4.856522 4.855955 4.848772 4.820233 4.804174 4.797922
## 1950 4.871068 4.910769 4.914237 4.903411 4.880407 4.897534 4.919155
## 1951 5.068616 5.106366 5.129127 5.119460 5.125189 5.096650 5.093885
## 1952 5.235669 5.261210 5.244282 5.229175 5.234433 5.252675 5.250005
## 1953 5.377370 5.398059 5.427971 5.442253 5.428888 5.398726 5.385137
## 1954 5.401213 5.390407 5.422845 5.438772 5.458206 5.468562 5.482440
## 1955 5.562748 5.567558 5.578168 5.599926 5.616585 5.639444 5.660083
## 1956 5.731809 5.737237 5.745164 5.759584 5.777070 5.797611 5.802414
## 1957 5.834317 5.834919 5.852610 5.868210 5.889502 5.916217 5.925050
## 1958 5.908226 5.887565 5.881697 5.886112 5.914122 5.949889 5.971106
## 1959 5.961656 5.964032 5.984909 6.004571 6.037076 6.054439 6.080169
## 1960 6.104473 6.086859 6.076676 6.125529 6.156198 6.174123 6.194812
##           Aug      Sep      Oct      Nov      Dec
## 1949 4.807954 4.835731 4.849678 4.859838 4.868112
## 1950 4.939708 4.971173 4.974310 4.987066 5.033356
## 1951 5.107276 5.141640 5.165802 5.193829 5.214744
## 1952 5.274834 5.292563 5.323613 5.351662 5.366545
## 1953 5.396309 5.404991 5.413698 5.409904 5.405722
## 1954 5.483905 5.496105 5.510675 5.527858 5.542358
## 1955 5.659225 5.675272 5.686680 5.697300 5.719825
## 1956 5.801759 5.805979 5.807155 5.817559 5.826069
## 1957 5.932471 5.933036 5.928316 5.928246 5.918776
## 1958 5.982560 5.959019 5.956174 5.951841 5.945442
## 1959 6.095367 6.086787 6.090360 6.101140 6.104082
## 1960 6.192947 6.184152 6.192122 6.184475 6.177823
```

```r
head(coef(out_ap, states = "seasonal"))
```

```
##           sea_dummy1  sea_dummy2  sea_dummy3  sea_dummy4  sea_dummy5
## Jan 1949 -0.08939058 -0.10274498 -0.21733422 -0.07434075  0.06309325
## Feb 1949 -0.11072116 -0.08939058 -0.10274498 -0.21733422 -0.07434075
## Mar 1949  0.02023021 -0.11072116 -0.08939058 -0.10274498 -0.21733422
## Apr 1949 -0.01031441  0.02023021 -0.11072116 -0.08939058 -0.10274498
## May 1949 -0.01196281 -0.01031441  0.02023021 -0.11072116 -0.08939058
## Jun 1949  0.11090764 -0.01196281 -0.01031441  0.02023021 -0.11072116
##           sea_dummy6  sea_dummy7  sea_dummy8  sea_dummy9 sea_dummy10
## Jan 1949  0.20700361  0.21557420  0.11090764 -0.01196281 -0.01031441
## Feb 1949  0.06309325  0.20700361  0.21557420  0.11090764 -0.01196281
## Mar 1949 -0.07434075  0.06309325  0.20700361  0.21557420  0.11090764
## Apr 1949 -0.21733422 -0.07434075  0.06309325  0.20700361  0.21557420
## May 1949 -0.10274498 -0.21733422 -0.07434075  0.06309325  0.20700361
## Jun 1949 -0.08939058 -0.10274498 -0.21733422 -0.07434075  0.06309325
##          sea_dummy11
## Jan 1949  0.02023021
## Feb 1949 -0.01031441
## Mar 1949 -0.01196281
## Apr 1949  0.11090764
## May 1949  0.21557420
## Jun 1949  0.20700361
```

```r
# compare seasonal components
par(mfcol=c(2,1))
plot.ts(coef(out_ap, states = "seasonal")[,1], ylab = "",
        main = "kalman filter $ season")
plot.ts(decompose_ap1$seasonal, ylab = '', main = "Arima$season")
```






```
par(mfcol=c(1,1))
```

Use trigonometric expression

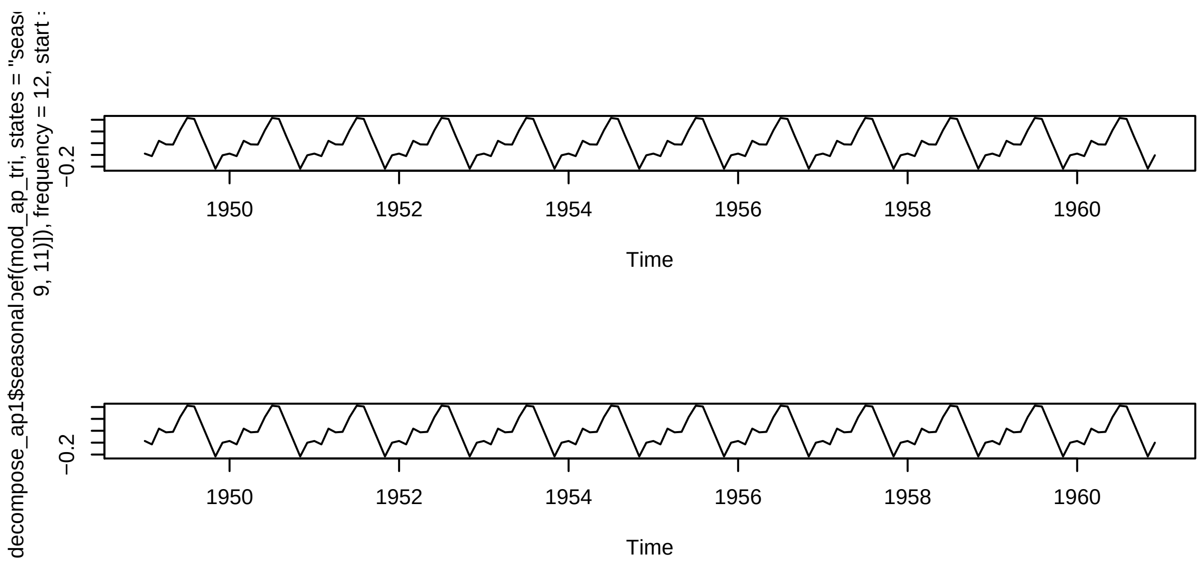
HINT: the seasonal component is the sum of column 1,3,5,7,9,11, when we use trigonometric seasonal.

```
mod_ap_tri <- SSModel(data_ap_log ~ SSMtrend(1, Q=list(5)) +
                    SSMseasonal(period = 12, sea.type = "trigonometric"),
                  H = 1)
out_ap_tri <- KFS(mod_ap_tri, filtering = "state")
print(out_ap_tri)
```

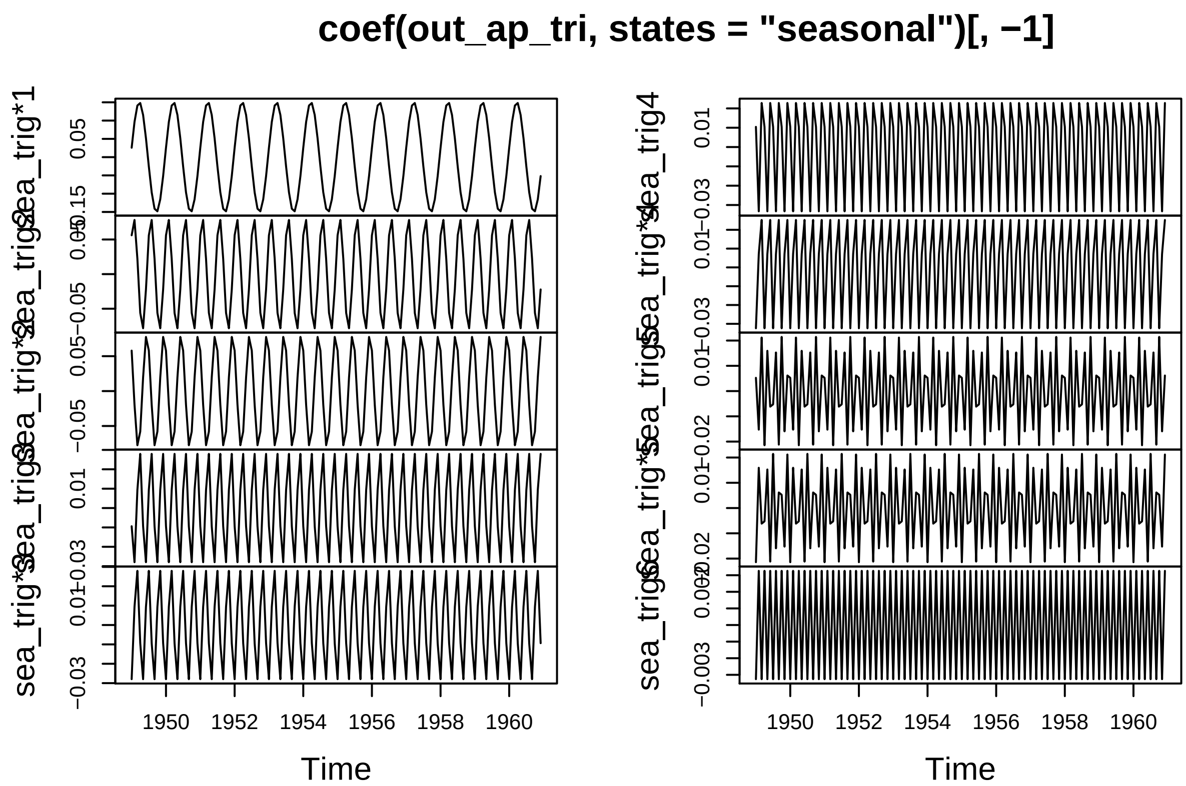
```
## Smoothed values of states and standard errors at time n = 144:
##              Estimate   Std. Error
## level         6.173755   1.130397
## sea_trig1    -0.141154   0.522885
## sea_trig*1   -0.051794   0.527186
## sea_trig2    -0.022160   0.289280
## sea_trig*2    0.077549   0.290485
## sea_trig3     0.027945   0.221270
## sea_trig*3   -0.009383   0.221270
## sea_trig4     0.022800   0.193356
## sea_trig*4    0.025234   0.192752
## sea_trig5     0.006168   0.181234
```

```
## sea_trig*5    0.021194    0.180337
## sea_trig6     0.003263    0.125349
```

```r
par(mfcol=c(3,1))
plot.ts(coef(out_ap_tri, states = "seasonal")[,1], ylab="")
plot.ts(ts(rowSums(coef(mod_ap_tri, states = "seasonal")[,c(1,3,5,7,9,11)]),
                   frequency = 12, start = 1949)) # sum the col's to get trigonometric season
plot.ts(decompose_ap1$seasonal)
```

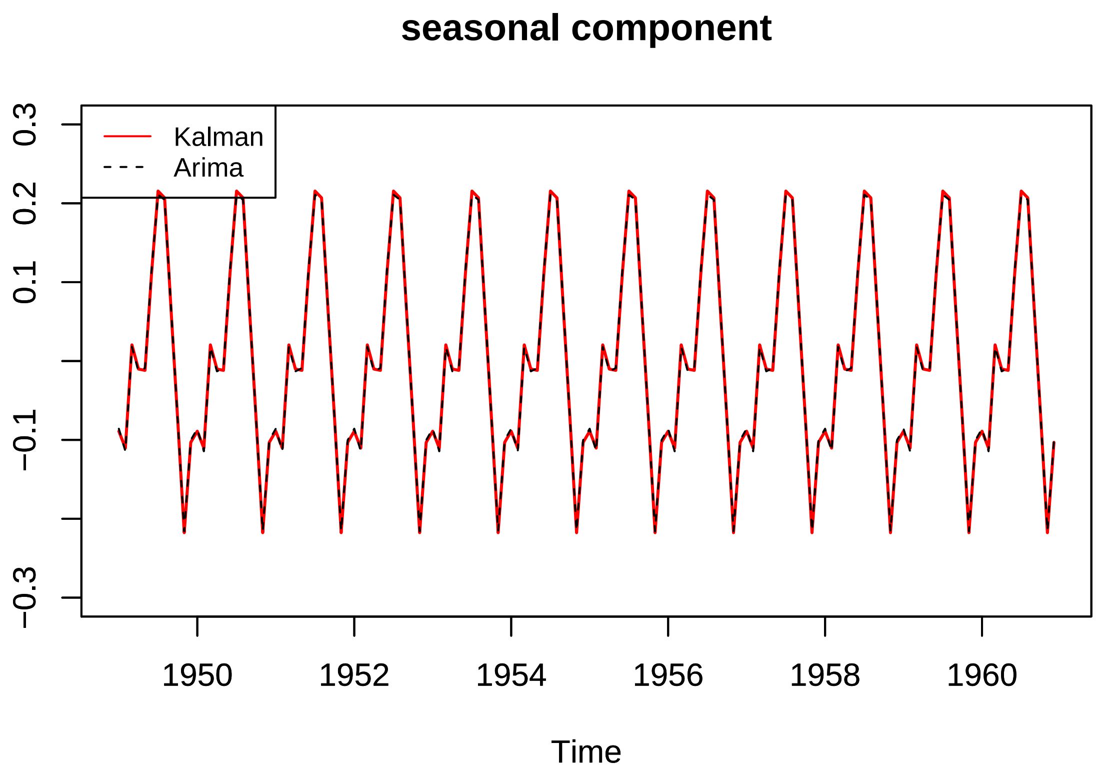



```r
par(mfcol=c(1,1))

plot.ts(coef(out_ap_tri, states = "seasonal")[,-1]) # maximum is 10 ts
```



compare the seasonal components from Arima and Kalman

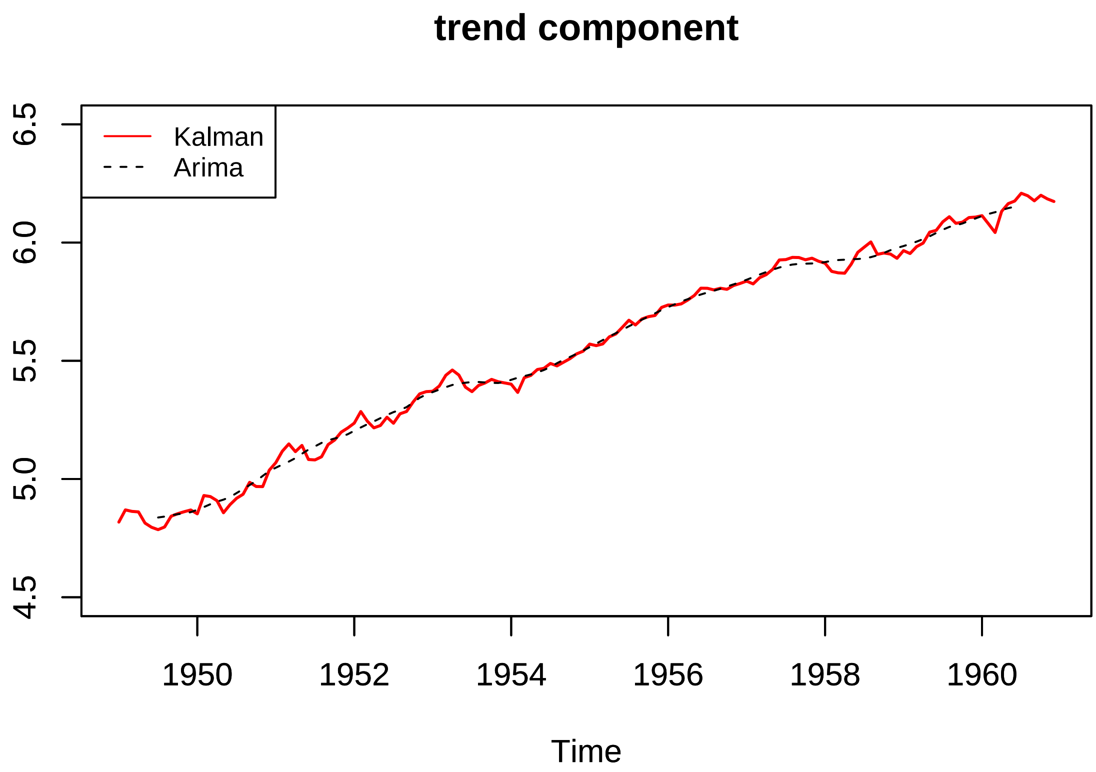
```
plot.ts(ts(rowSums(coef(mod_ap_tri, states = "seasonal")[,c(1,3,5,7,9,11)]),
           frequency = 12, start = 1949), col="red", lwd = 1.5,
        ylim=c(-0.3,0.3), ylab='')
par(new=TRUE)
plot.ts(ts(decompose_ap1$seasonal,frequency = 12, start = 1949),
        lty = 2, ylim=c(-0.3,0.3),ylab='')
title(main = 'seasonal component')
legend("topleft",c("Kalman", "Arima"),col=c("red","black"), lty=c(1,2),cex=0.8)
```



compare the trend components from Arima and Kalman

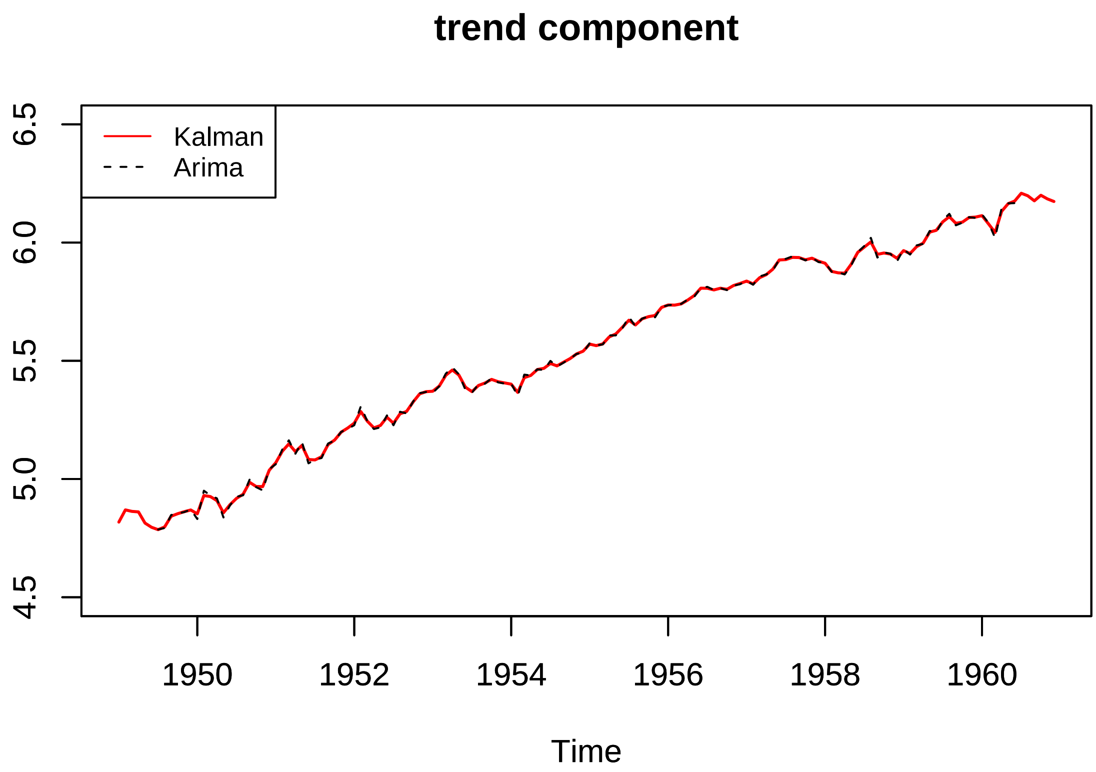
It shows that the *trend* from Kalman filter includes noise.

```
plot.ts(ts(coef(mod_ap_tri, states = "trend"),
           frequency = 12, start = 1949), col="red", lwd = 1.5,
       ylim=c(4.5,6.5), ylab='')
par(new=TRUE)
plot.ts(ts(decompose_ap1$trend,frequency = 12, start = 1949),
        lty = 2, ylim=c(4.5,6.5),ylab='') # only trend from arima
title(main = 'trend component')
legend("topleft",c("Kalman", "Arima"),col=c("red","black"), lty=c(1,2),cex=0.8)
```



Two curves fit almost perfectly after adding noise to arima's trend.

```
plot.ts(ts(coef(mod_ap_tri, states = "trend"),
           frequency = 12, start = 1949), col="red", lwd = 1.5,
        ylim=c(4.5,6.5), ylab='')
par(new=TRUE)
plot.ts(ts(decompose_ap1$trend+decompose_ap1$random,frequency = 12, start = 1949),
        lty = 2, ylim=c(4.5,6.5),ylab='') # sum trend and noise from arima
title(main = 'trend component')
legend("topleft",c("Kalman", "Arima"),col=c("red","black"), lty=c(1,2),cex=0.8)
```

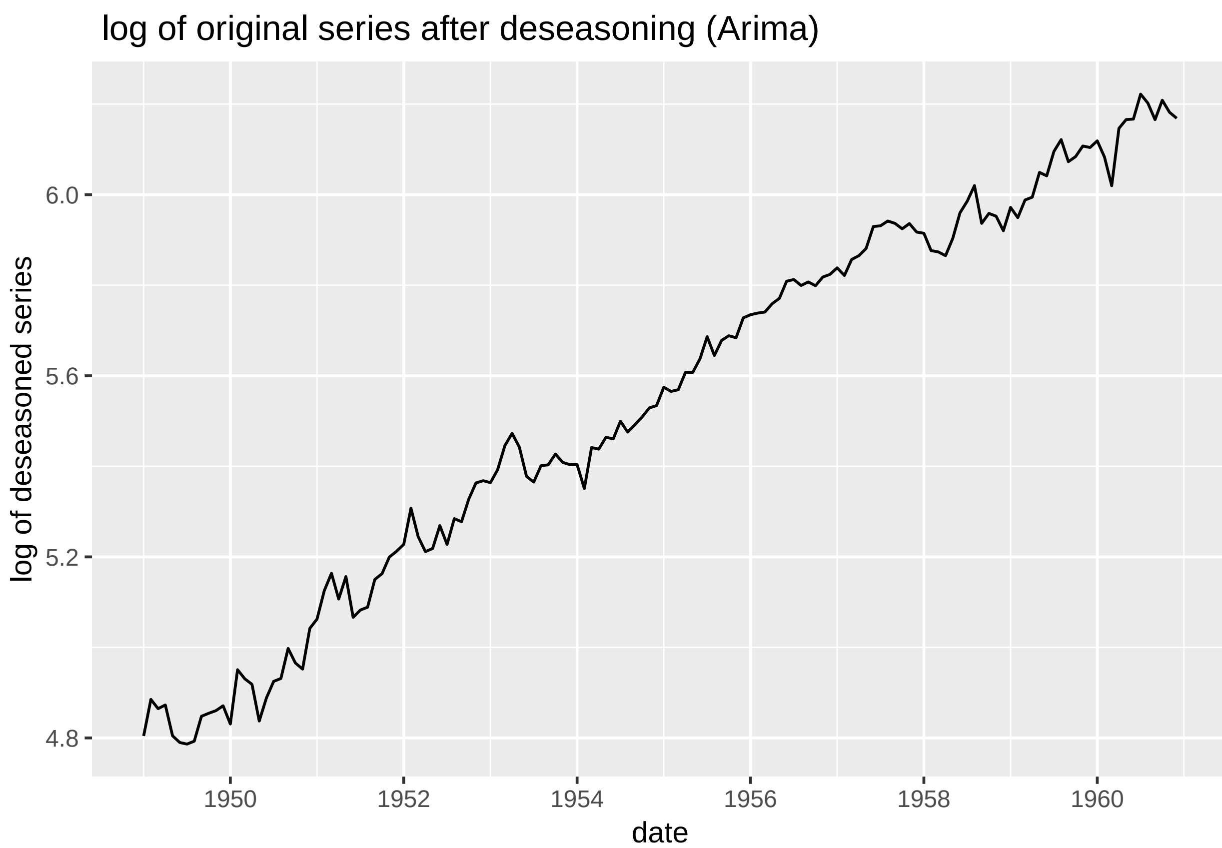


Seasonal Adjustment

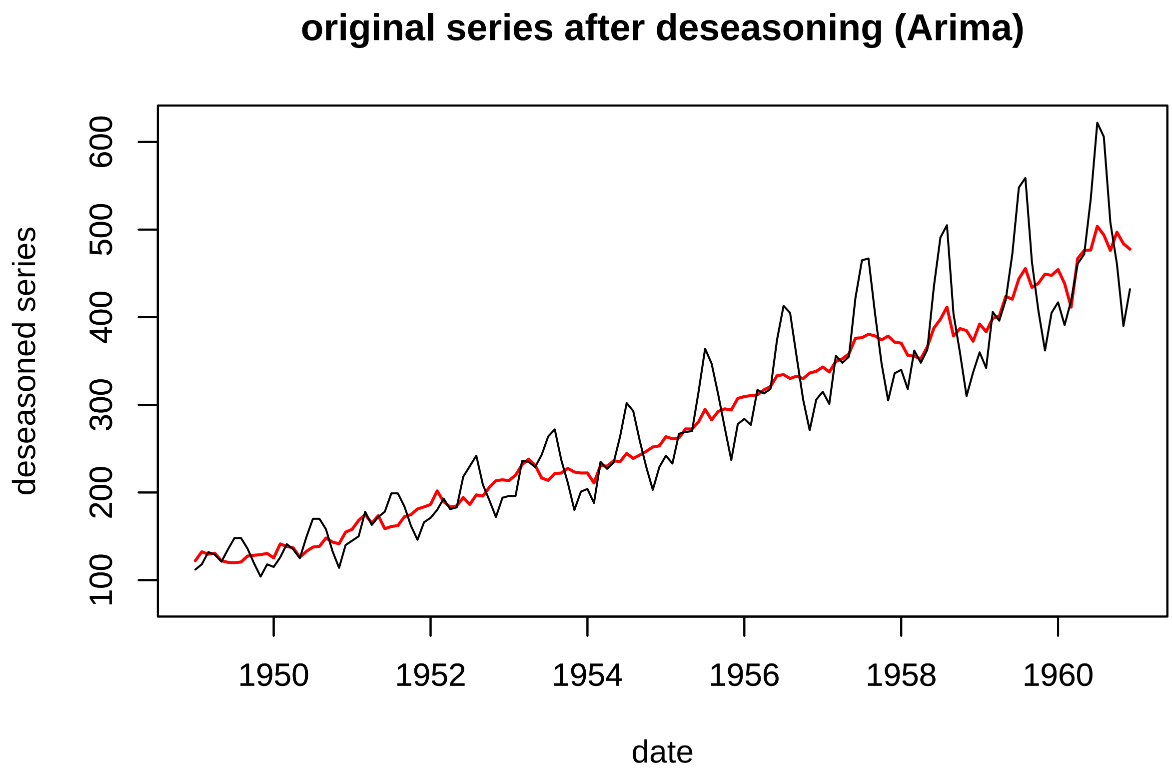
Since the results of seasonal components are the same, the deseasoned seires should be the same as well.

arima

```
autoplot(data_ap_log-decompose_ap1$seasonal)+labs(x="date", y="log of deseasoned series", title = " log
```

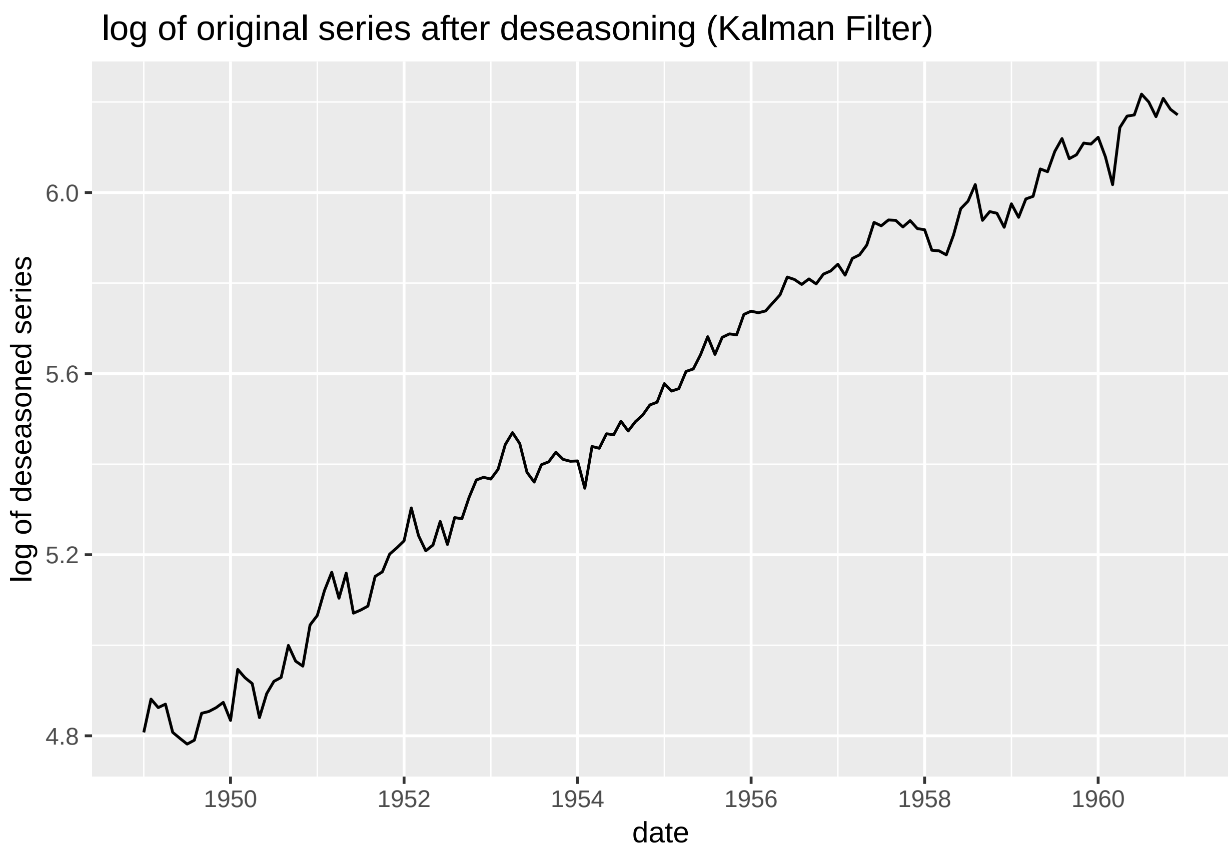



```r
plot.ts(exp(data_ap_log-decompose_ap1$seasonal),
        ylim=c(80,620),ylab='',xlab='', col="red",lwd=1.5)
par(new=TRUE)
plot.ts(AirPassengers,ylim=c(80,620),ylab='',xlab='')
title(xlab = "date", ylab="deseasoned series", main = "original series after deseasoning (Arima)")
```

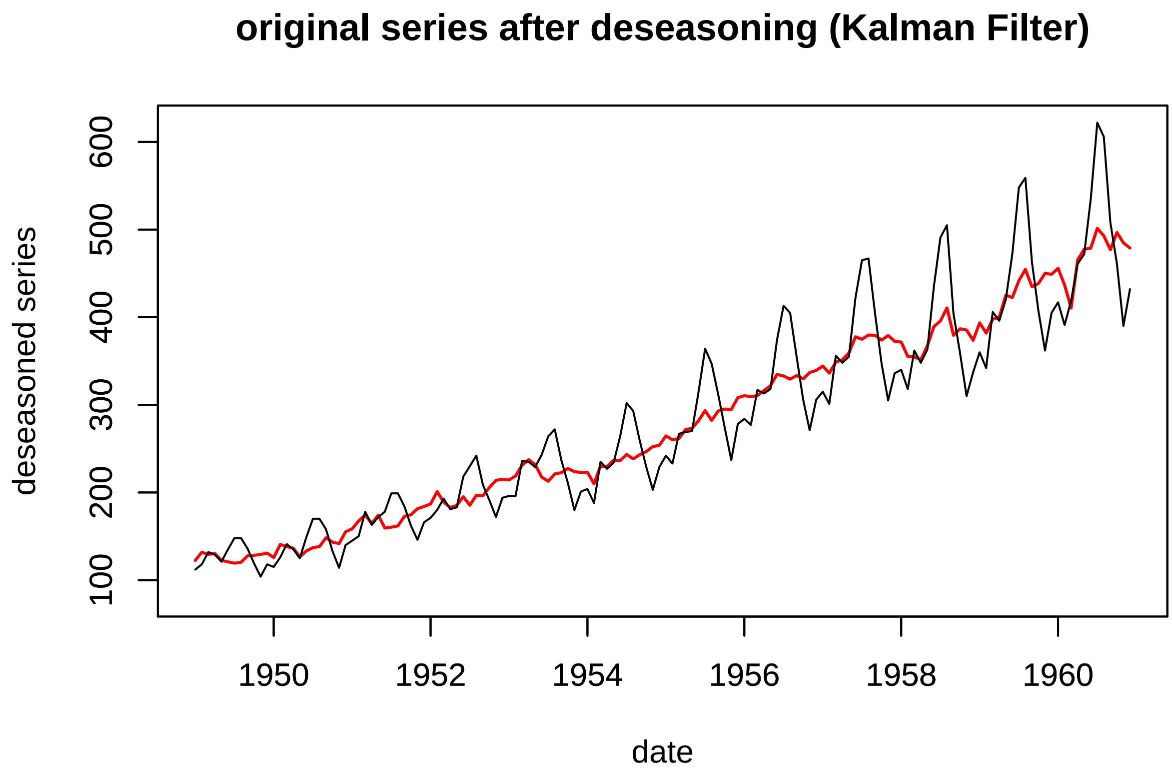


kalman filter

```
autoplot(data_ap_log - rowSums(coef(mod_ap_tri,
                                    states = "seasonal")[,c(1,3,5,7,9,11)])) + labs(x="date", y="log of
```




```r
plot.ts(exp(data_ap_log - rowSums(coef(mod_ap_tri,
                                   states = "seasonal")[,c(1,3,5,7,9,11)])),
        xlab='', ylab='', ylim=c(80,620), col='red', lwd=1.5)
par(new=TRUE)
plot.ts(AirPassengers,xlab='', ylab='', ylim=c(80,620))
title(xlab="date", ylab="deseasoned series", main = "original series after deseasoning (Kalman Filter)")
```



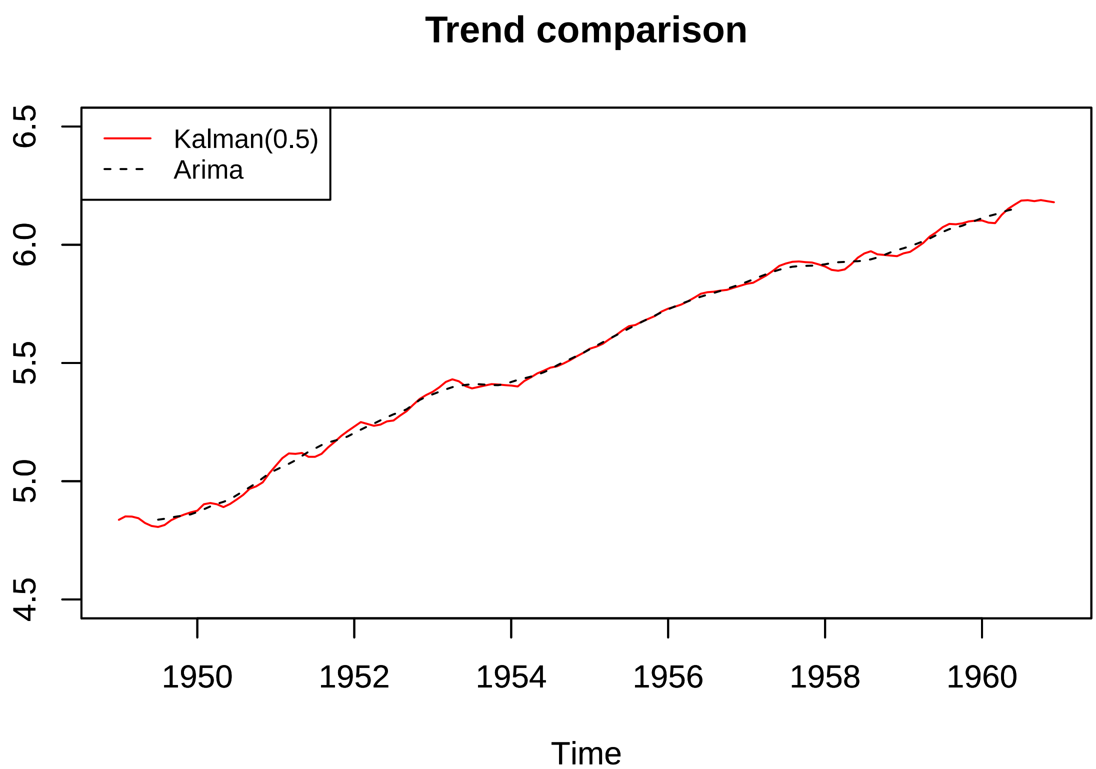
Some improvement

the variance of trend is 0.5

```
mod_ap_tri2 <- SSModel(data_ap_log ~ SSMtrend(1, Q=list(0.5)) +
                       SSMseasonal(period = 12, sea.type = "trigonometric"),
                     H = 1)
out_ap_tri2 <- KFS(mod_ap_tri2, filtering = "state")
print(out_ap_tri2)
```

```
## Smoothed values of states and standard errors at time n = 144:
##             Estimate   Std. Error
## level        6.179837   0.736846
## sea_trig1   -0.141064   0.199707
## sea_trig*1  -0.052128   0.202190
## sea_trig2   -0.022071   0.144605
## sea_trig*2   0.077395   0.145137
## sea_trig3    0.028034   0.132054
## sea_trig*3  -0.009473   0.132054
## sea_trig4    0.022889   0.127597
## sea_trig*4   0.025183   0.127395
## sea_trig5    0.006257   0.125805
## sea_trig*5   0.021170   0.125520
## sea_trig6    0.003307   0.088497
```

```r
plot(coef(out_ap_tri2, states = "trend"), ylim=c(4.5,6.5),ylab='',col=2)
par(new=TRUE)
plot(decompose_ap1$trend,ylim=c(4.5,6.5),ylab='', lty=2)
title(main='Trend comparison')
legend('topleft', c('Kalman(0.5)', 'Arima'), col=c(2,1), lty = c(1,2), cex = 0.8)
```



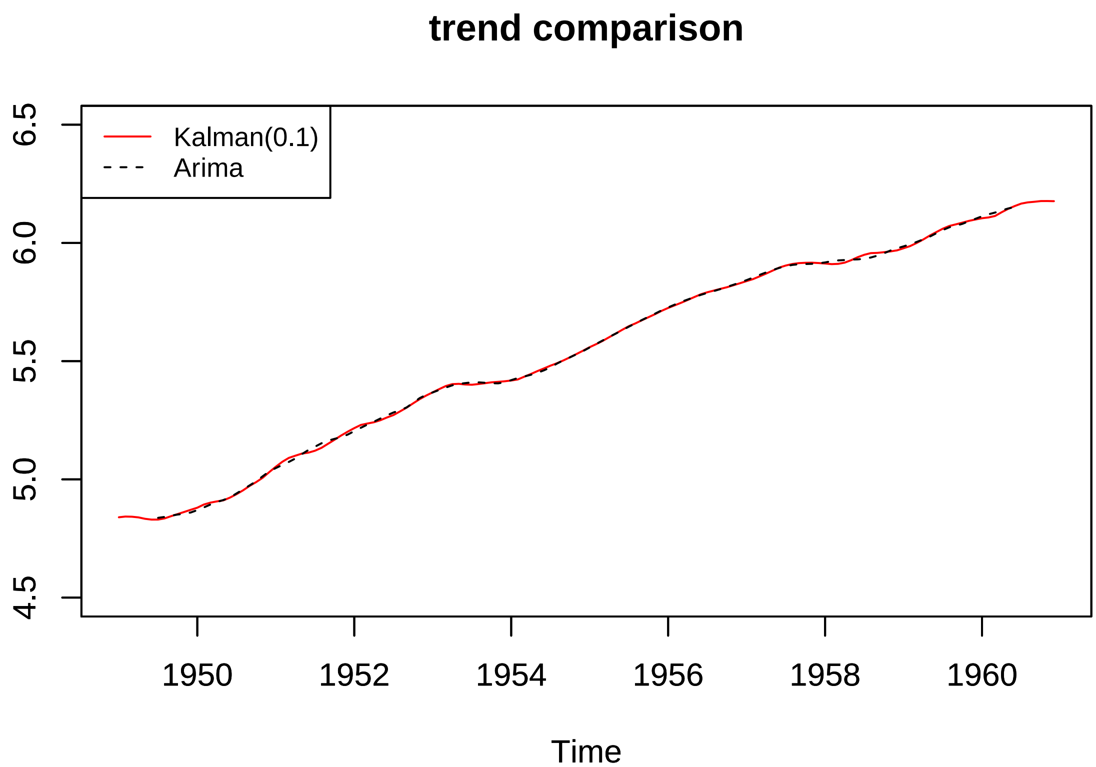
the variance of trend is 0.1

```r
mod_ap_tri3 <- SSModel(data_ap_log ~ SSMtrend(1, Q=list(0.1)) +
                     SSMseasonal(period = 12, sea.type = "trigonometric"),
                   H = 1)
out_ap_tri3 <- KFS(mod_ap_tri3, filtering = "state")
print(out_ap_tri3)
```

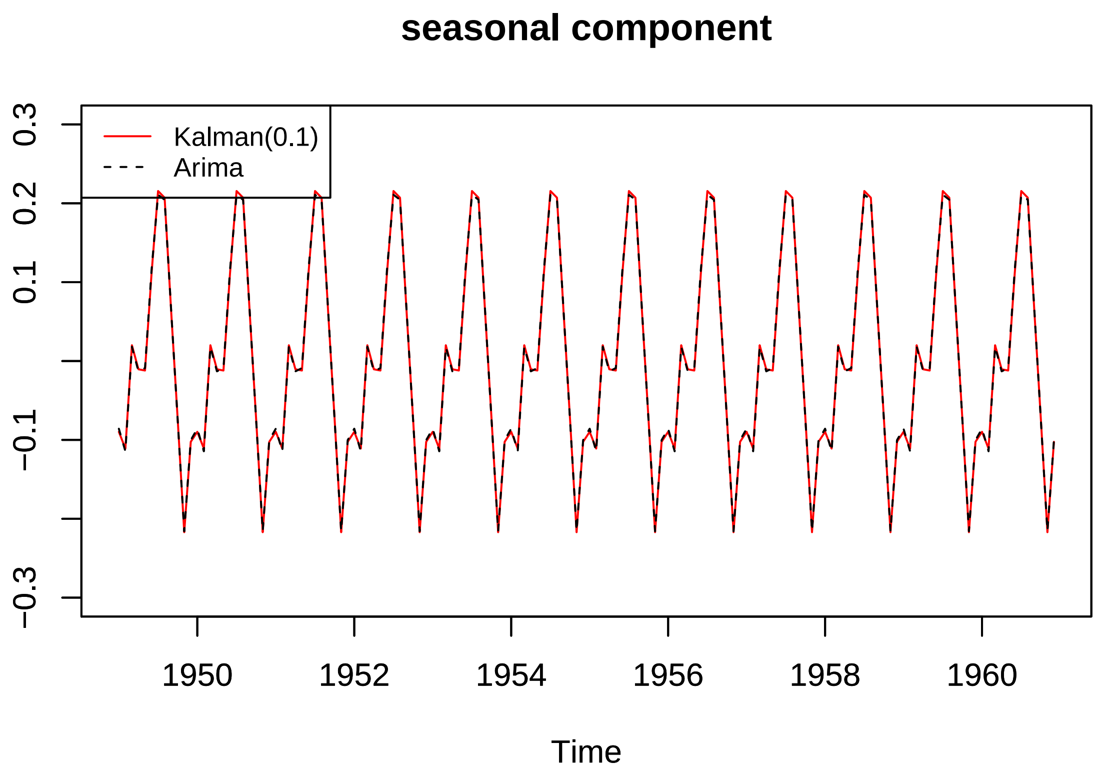
```
## Smoothed values of states and standard errors at time n = 144:
##             Estimate   Std. Error
## level        6.176531   0.527956
## sea_trig1   -0.141022   0.138219
## sea_trig*1  -0.052287   0.139720
## sea_trig2   -0.022028   0.123734
## sea_trig*2   0.077321   0.123994
## sea_trig3    0.028077   0.120895
## sea_trig*3  -0.009515   0.120895
## sea_trig4    0.022932   0.119934
## sea_trig*4   0.025158   0.119844
```

```
## sea_trig5    0.006300   0.119555
## sea_trig*5   0.021159   0.119429
## sea_trig6    0.003329   0.084416
```

```r
# compare trend
plot(coef(out_ap_tri3, states = "trend"), ylim=c(4.5,6.5),ylab='',col=2)
par(new=TRUE)
plot(decompose_ap1$trend,ylim=c(4.5,6.5),ylab='', lty=2)
title(main='trend comparison')
legend('topleft', c('Kalman(0.1)', 'Arima'), col=c(2,1), lty = c(1,2), cex = 0.8)
```




```r
# let's compare seasonal part under this case
plot(ts(rowSums(coef(out_ap_tri3, states = "seasonal")[,c(1,3,5,7,9,11)]),
        frequency = 12, start = 1949),
     ylim=c(-.3,.3),ylab='',col=2)
par(new=TRUE)
plot(decompose_ap1$seasonal,ylim=c(-.3,.3),ylab='', lty=2)
title(main =' seasonal component ' )
legend('topleft', c('Kalman(0.1)', 'Arima'), col=c(2,1), lty = c(1,2), cex = 0.8)
```



Until here, we still don't reproduce the results from statcan, where the seasonal adjustment curve is smooth.(**SOLVED IN KALMAN FILTER DOCUMENT**)