ORIE5550_HW7_Markdown

Luis Alonso Cendra Villalobos (lc2234)

2024-04-13

```r
#install.packages("tidyverse")
options(warn=-1) # turn off warnings
library(astsa)
library(perARMA)
library(forecast)
```

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

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:astsa':
##
##     gas
```

```r
library(urca)
library(xts)
```

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

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

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

```r
library(fGarch)
```

```
## NOTE: Packages 'fBasics', 'timeDate', and 'timeSeries' are no longer
## attached to the search() path when 'fGarch' is attached.
##
## If needed attach them yourself in your R script by e.g.,
##         require("timeSeries")
```

```r
library(tseries)
library(FinTS)
```

```
##
## Attaching package: 'FinTS'

## The following object is masked from 'package:forecast':
##
##     Acf
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::first()  masks xts::first()
## x dplyr::lag()    masks stats::lag()
## x dplyr::last()   masks xts::last()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(forecast)
options(warn=0) # turn on warnings
```

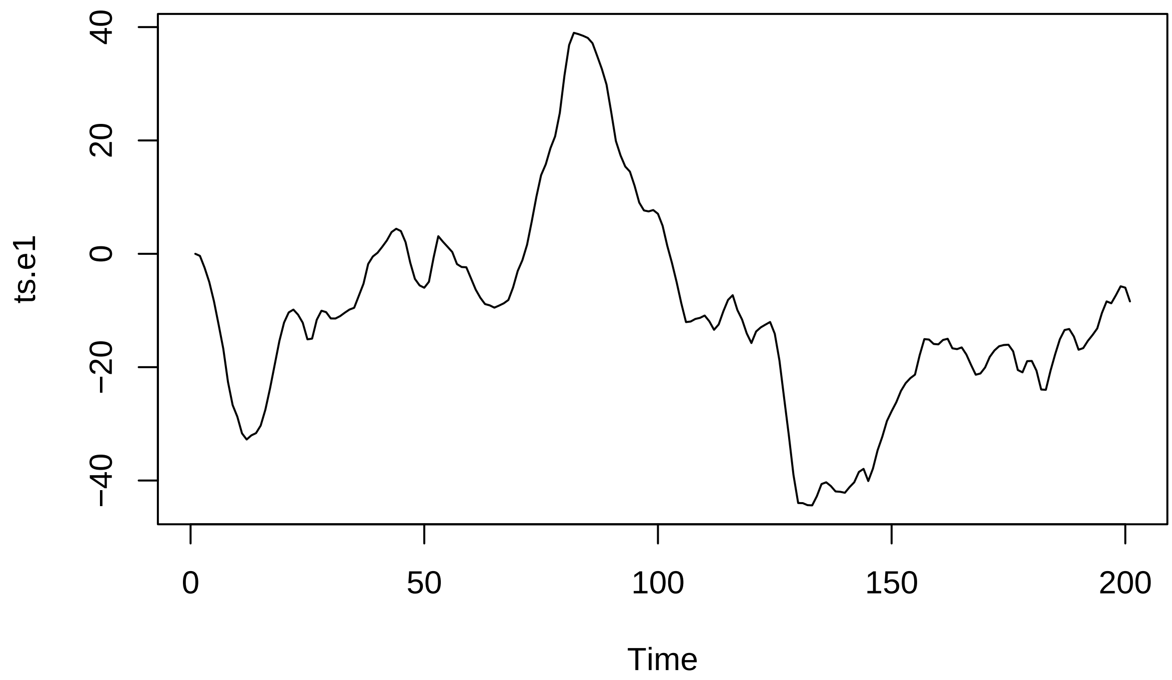
Problem 1

(a) Rewrite this model in a state-space form

See handwritten notes.

(b) Use set.seed(123) to generate a time series of length 200 from ARIMA(1,1,1) with $1 = 0.7$, $1 = 0.5$, and $2 Z = 2$ through arima.sim; Produce a time plot of the series; Use Arima to estimate the model parameters $1, 1, Z$ of the time series

```r
set.seed(123)

ts.e1 <- arima.sim(model=list(ar=0.7, ma=0.5, order=c(1,1,1)), n=200, sd = sqrt(2))
ts.e1 <- as.numeric(ts.e1)
plot.ts(ts.e1)
```




```
arima.model <- Arima(ts.e1,
                      order=c(1,1,1),
                      method = "ML",
                      include.mean = TRUE)
arima.model
```

```
## Series: ts.e1
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1     ma1
##       0.5800  0.6044
## s.e.  0.0717  0.0886
##
## sigma^2 = 1.77:  log likelihood = -340.63
## AIC=687.25   AICc=687.38   BIC=697.15
```

(c) Estimate the model parameters 1, 1, Z from the generated time series data via estimation for the state-space model; Report estimates and their standard errors [Note: You can use the estimates in (a) as the initial guess of the parameters].

```
library(astsa)

# Function to Calculate Likelihood
```

```r
Linn <- function(para){


 Phi <- diag(0,3)
 phi_1 <- para[1]
 Theta_1 <- para[2]
 Phi[1,1] <- 1
 Phi[1,2] <- 1
 Phi[1,3] <- Theta_1
 Phi[2,1] <- 0
 Phi[2,2] <- phi_1
 Phi[2,3] <- 0
 Phi[3,] <- c(0,1,0);

 A <- cbind(1,1,Theta_1)

 sigma_Z <- para[3] # sqrt sigma_Z^2
 QQ <- diag(0,3)
 QQ[1,] <- c(0,0,0)
 QQ[2,1] <- 0
 QQ[2,2] <- sigma_Z
 QQ[2,3] <- 0
 QQ[3,] <- c(0,0,0)

 RR <- 0

 kf <- Kfilter(ts.e1, A, mu0, Sigma0, Phi, QQ, RR,
             Ups=NULL, Gam=NULL,input=NULL,S=NULL,version=1)
 return(kf$like)

}

# Kfilter

# Initial Parameters
mu0      <- c(ts.e1[1],0,0)
Sigma0   <- diag(1,3)
init.par <- c(0.58, 0.6044, sqrt(1.77))  # G[1,1], the 2 Rs and Q

# Estimation
est <- optim(init.par, Linn, NULL, method="BFGS", hessian=TRUE, control=list(trace=0,REPORT=1))
SE  <- sqrt(diag(solve(est$hessian)))
u   <- cbind(estimate=est$par,SE)
rownames(u)=c("Phi1","Theta1","sigz2"); u
```

```
##          estimate         SE
## Phi1   0.5837906 0.07204385
## Theta1 0.6009904 0.08928380
## sigz2  1.3237167 0.06615694
```

```r
est
```

```
## $par
```

```
## [1] 0.5837906 0.6009904 1.3237167
##
## $value
## [1] 157.5489
##
## $counts
## function gradient
##       23        6
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##              [,1]       [,2]         [,3]
## [1,] 300.6356896 145.380729   0.7811095
## [2,] 145.3807291 195.758786   1.9282564
## [3,]   0.7811095   1.928256 228.5017488
```

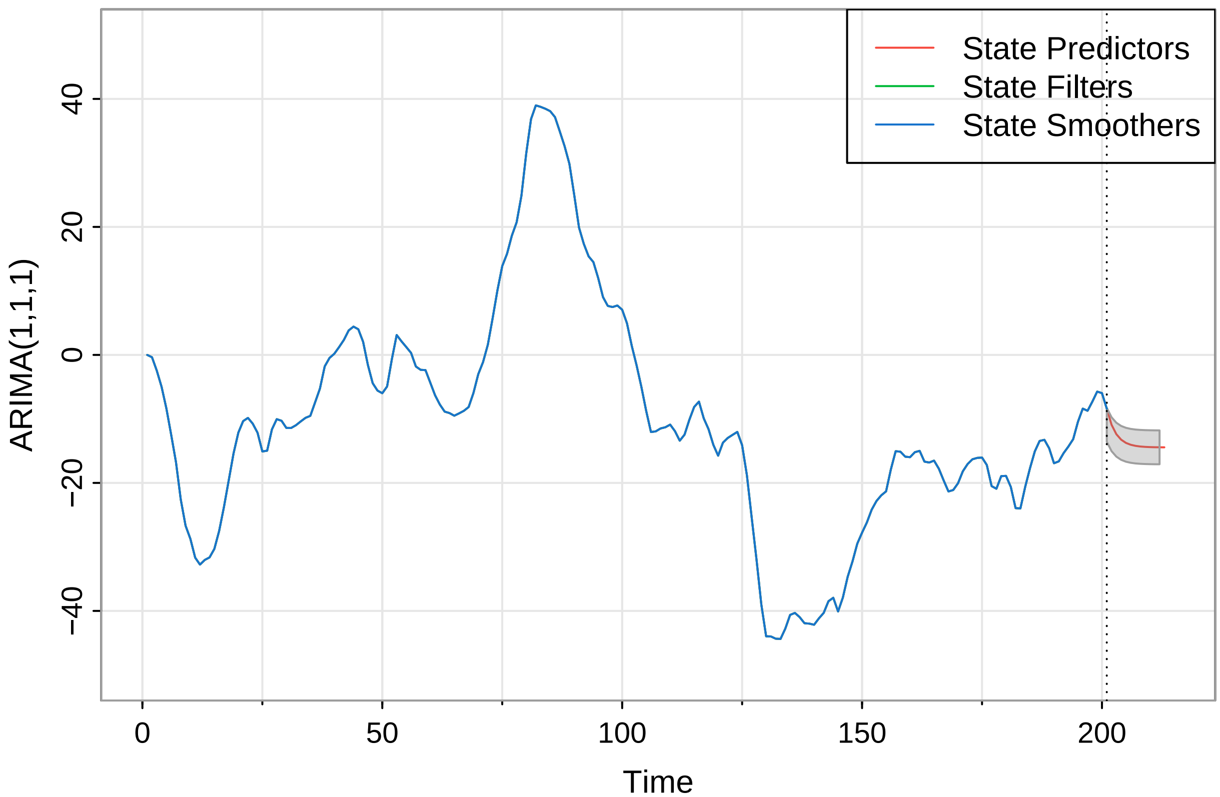
We have that $\phi_1 \approx 0.58$, $\theta_1 \approx 0.60$ and $\sigma_Z \approx 1.32$.

(d) Smooth the state variables X_t; Overlap the time series of $Y_t = AX_t$, $Y_{t|t} = AX_{t|t}$, and $Y_{t|T} = AX_{t|T}$ with different colors in a single figure [Hint: You may encounter computationally singular in this example. This is caused by the inverse computation. If so, add an arbitrarily small enough number to the parameter that is indeed zero.]

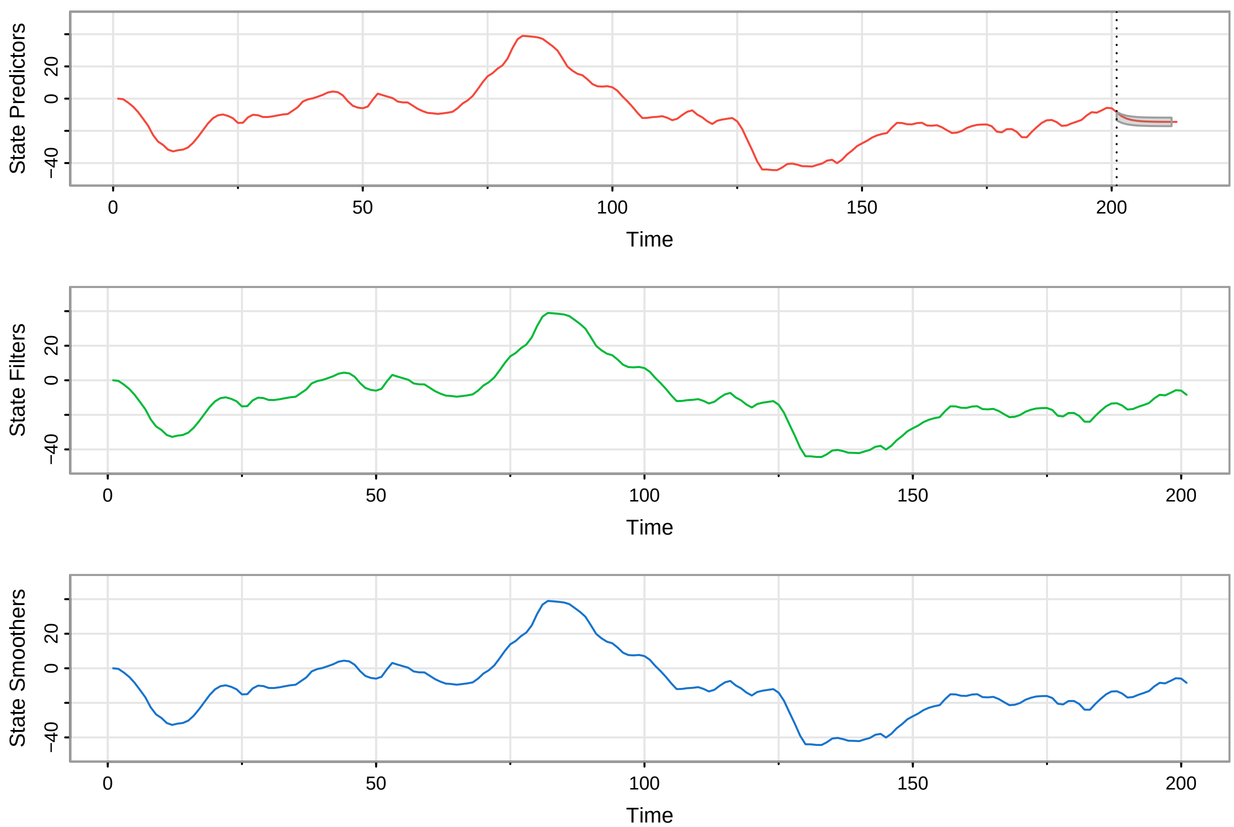
```
# Ksmooth

# Smoothing

Phi <- diag(0,3)
Phi[1,1] <- 1
Phi[1,2] <- 1
Phi[1,3] <- est$par[2]
Phi[2,1] <- 0
Phi[2,2] <- est$par[1]
Phi[2,3] <- 0
Phi[3,] <- c(0,1,0);

A <- cbind(1,1,est$par[2])

QQ <- diag(0,3)
QQ[1,] <- c(0,0,0)
QQ[2,1] <- 0
QQ[2,2] <- est$par[3]
QQ[2,3] <- 0
QQ[3,] <- c(0,0,0)

RR <- 0.0000001
```

```r
ks <- Ksmooth(ts.e1, A, mu0, Sigma0, Phi, QQ, RR)

# Plots

# Smoothers

Tsm   <- ts(as.numeric(ks$Xs[1,,]))
Ssm   <- ts(as.numeric(ks$Xs[2,,]))
Rsm   <- ts(as.numeric(ks$Xs[3,,]))

matrix_Smoothers <- matrix(nrow = 3, ncol = length(as.numeric(ks$Xs[1,,])))
matrix_Smoothers[1,] <- as.numeric(ks$Xs[1,,])
matrix_Smoothers[2,] <- as.numeric(ks$Xs[2,,])
matrix_Smoothers[3,] <- as.numeric(ks$Xs[3,,])

State_Smoothers   <- ts(as.numeric(A %*% matrix_Smoothers)) #Tsm + Ssm + Rsm

# Filters

Tsf   <- ts(as.numeric(ks$Xf[1,,]))
Ssf   <- ts(as.numeric(ks$Xf[2,,]))
Rsf   <- ts(as.numeric(ks$Xf[3,,]))

matrix_Filters <- matrix(nrow = 3, ncol = length(as.numeric(ks$Xf[1,,])))
matrix_Filters[1,] <- as.numeric(ks$Xf[1,,])
matrix_Filters[2,] <- as.numeric(ks$Xf[2,,])
matrix_Filters[3,] <- as.numeric(ks$Xf[3,,])

State_Filters   <- ts(as.numeric(A %*% matrix_Filters)) #Tsf + Ssf + Rsf

# Predictors

matrix_Predictors <- matrix(nrow = 3, ncol = length(as.numeric(ks$Xp[1,,])))
matrix_Predictors[1,] <- as.numeric(ks$Xp[1,,])
matrix_Predictors[2,] <- as.numeric(ks$Xp[2,,])
matrix_Predictors[3,] <- as.numeric(ks$Xp[3,,])

State_Predictors <- ts(as.numeric(A %*% matrix_Predictors))


p1    <- 3*sqrt(ks$Ps[1,1,]); p2 = 3*sqrt(ks$Ps[2,2,])

# Forecast
num <- length(ts.e1)
n.ahead <- 12
y       <- ts(append(ts.e1, rep(0,n.ahead)))
rmspe   <- rep(0,n.ahead)
x00     <- ks$Xf[,,num]
P00     <- ks$Pf[,,num]
Q       <- t(QQ) %*% QQ
R       <- RR^2

for (m in 1:n.ahead){
```

```
      xp <- Phi%*%x00
      Pp <- Phi%*%P00%*%t(Phi)+Q
     sig <- A%*%Pp%*%t(A)+R
       K <- Pp%*%t(A)%*%(1/sig)
     x00 <- xp
     P00 <- Pp-K%*%A%*%Pp
 y[num+m] <- A%*%xp
 rmspe[m] <- sqrt(sig)
}

# Single graph
par(mfrow=c(1,1))
tsplot(y, main='', ylab='ARIMA(1,1,1)', ylim=c(-50,50), xlim = c(0,215), col=2)
upp  <- ts(y[(num+1):(num+n.ahead)]+2*rmspe, start=num, freq=1)
low  <- ts(y[(num+1):(num+n.ahead)]-2*rmspe, start=num, freq=1)
 xx  <- c(time(low), rev(time(upp)))
 yy  <- c(low, rev(upp))
polygon(xx, yy, border=8, col=gray(.5, alpha = .3))
abline(v=num, lty=3)
lines(State_Filters, ylim=c(-50,50),col=3)
lines(State_Smoothers, ylim=c(-50,50),col=4)
legend("topright",
       c("State Predictors", "State Filters", "State Smoothers"),
       lty = 1,
       col = 2:4)
```



```r
# Several graphs
par(mfrow=c(3,1))
tsplot(y, main='', ylab='State Predictors', ylim=c(-50,50), xlim = c(0,215), col=2)
upp  <- ts(y[(num+1):(num+n.ahead)]+2*rmspe, start=num, freq=1)
low  <- ts(y[(num+1):(num+n.ahead)]-2*rmspe, start=num, freq=1)
 xx  <- c(time(low), rev(time(upp)))
 yy  <- c(low, rev(upp))
polygon(xx, yy, border=8, col=gray(.5, alpha = .3))
abline(v=num, lty=3)
tsplot(State_Filters, ylab='State Filters', ylim=c(-50,50),col=3)
tsplot(State_Smoothers, ylab='State Smoothers', ylim=c(-50,50),col=4)
```



Problem 2

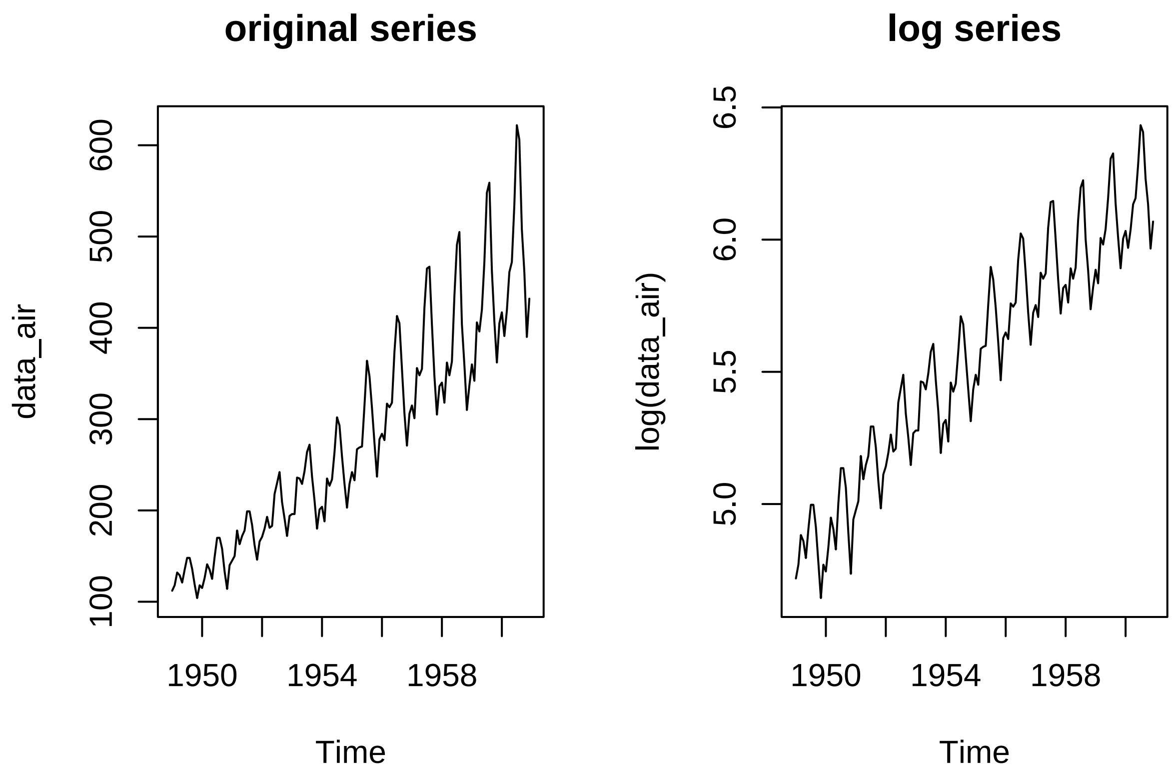
Consider the monthly totals of international airline passengers from 1949 to 1960 AirPassengers in the R package astsa. Do the following.

(a) Rewrite this model in a state-space form

See handwritten notes.

(b) Similar to Problem 1, estimate the model parameters V, W, Z, and U; Report estimates and their standard errors.

```
data_air = AirPassengers
par(mfrow = c(1, 2))
plot.ts(data_air, main="original series")
plot.ts(log(data_air), main="log series")
```




```
A <- cbind(1,0,1,0,0,0,0,0,0,0,0,0,0)

# Function to Calculate Likelihood
Linn <- function(para){
 Phi <- diag(0,13)
 Phi[1,1] <- 1
 Phi[1,2] <- 1
 Phi[2,2] <- 1
 Phi[3,] <- c(0,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1);
 Phi[4,3] <- 1;
 Phi[5,4] <- 1;
 Phi[6,5] <- 1;
 Phi[7,6] <- 1;
 Phi[8,7] <- 1;
 Phi[9,8] <- 1;
 Phi[10,9] <- 1;
```

```r
 Phi[11,10] <- 1;
 Phi[12,11] <- 1;
 Phi[13,12] <- 1;

 Q1 <- para[2] #Wt
 Q2 <- para[3] #Zt
 Q3 <- para[4] #Ut
 QQ <- diag(0,13)
 QQ[1,1] <- Q1
 QQ[2,2] <- Q2
 QQ[3,3] <- Q3

 RR <- para[1] #Vt

 kf <- Kfilter(data_air, A, mu0, Sigma0, Phi, QQ, RR,
              Ups=NULL, Gam=NULL,input=NULL,S=NULL,version=1)
 return(kf$like)
}
```

```r
# Kfilter

# Initial Parameters
mu0      <- c(rep(1.5, 13))
Sigma0   <- diag(1.5, 13)
init.par <- c(10,5,1,10)  # G[1,1], the 2 Rs and Q

# Estimation
options(warn=-1) # turn off warnings
est <- optim(init.par, Linn, NULL, method="BFGS", hessian=TRUE, control=list(trace=0,REPORT=1))
SE  <- sqrt(diag(solve(est$hessian)))
options(warn=0) # turn on warnings
u   <- cbind(estimate=est$par,SE)
rownames(u)=c("SigmaV^2","SigmaW^2","SigmaZ^2","SigmaU^2"); u
```

```
##              estimate        SE
## SigmaV^2 2.309055e-03 2.2334758
## SigmaW^2 1.843674e+01 2.1325930
## SigmaZ^2 2.325622e-07 0.2204203
## SigmaU^2 8.605274e+00 1.4436384
```

```r
est
```

```
## $par
## [1] 2.309055e-03 1.843674e+01 2.325622e-07 8.605274e+00
##
## $value
## [1] 552.3576
##
## $counts
## function gradient
##       25       15
##
```

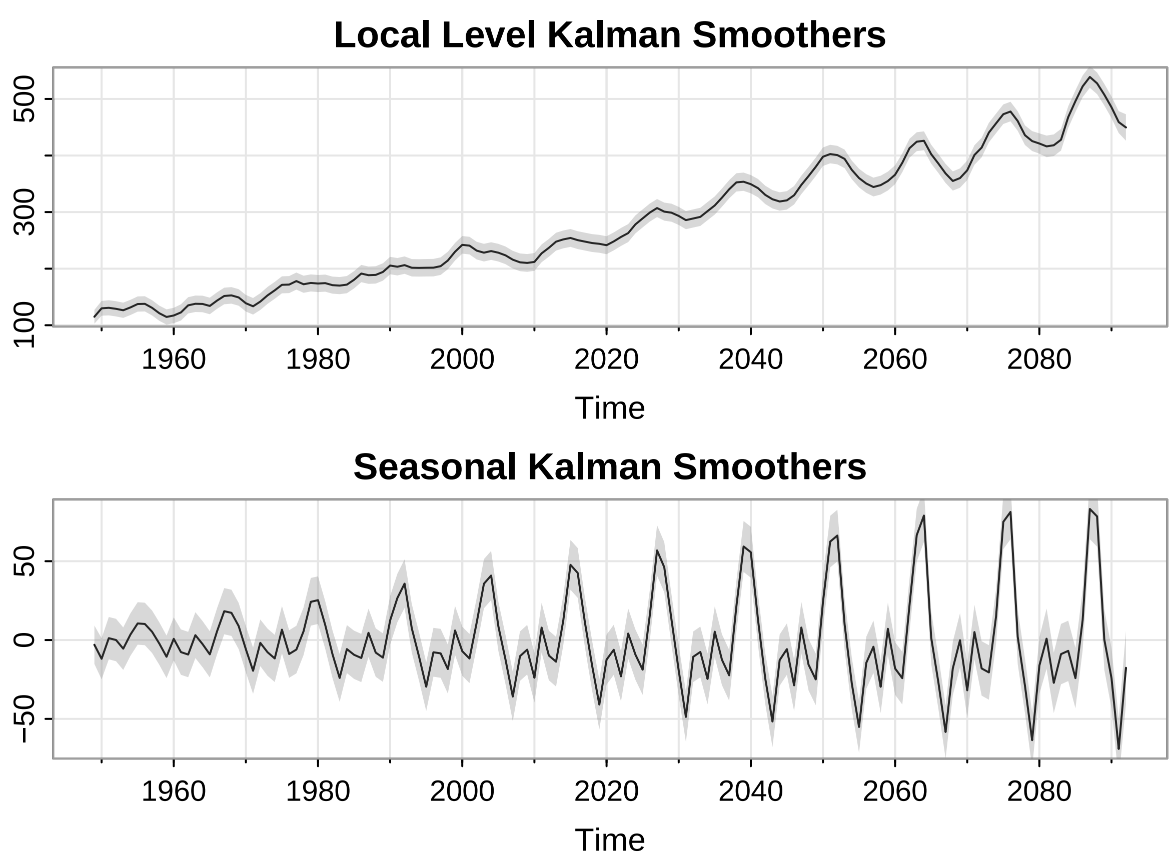
```
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##               [,1]          [,2]          [,3]          [,4]
## [1,]  2.004645e-01  3.751666e-06  2.842171e-08 -2.519585e-05
## [2,]  3.751666e-06  3.585902e-01 -4.121148e-07  3.294612e-01
## [3,]  2.842171e-08 -4.121148e-07  2.058244e+01  1.421085e-08
## [4,] -2.519585e-05  3.294612e-01  1.421085e-08  7.825237e-01
```

We have that $\sigma_V \approx sqrt(0.0023091)$, $\sigma_W \approx sqrt(18.4367429)$, $\sigma_Z \approx sqrt(2.3256222 \times 10^{-7})$ and $\sigma_U \approx sqrt(8.6052744)$.

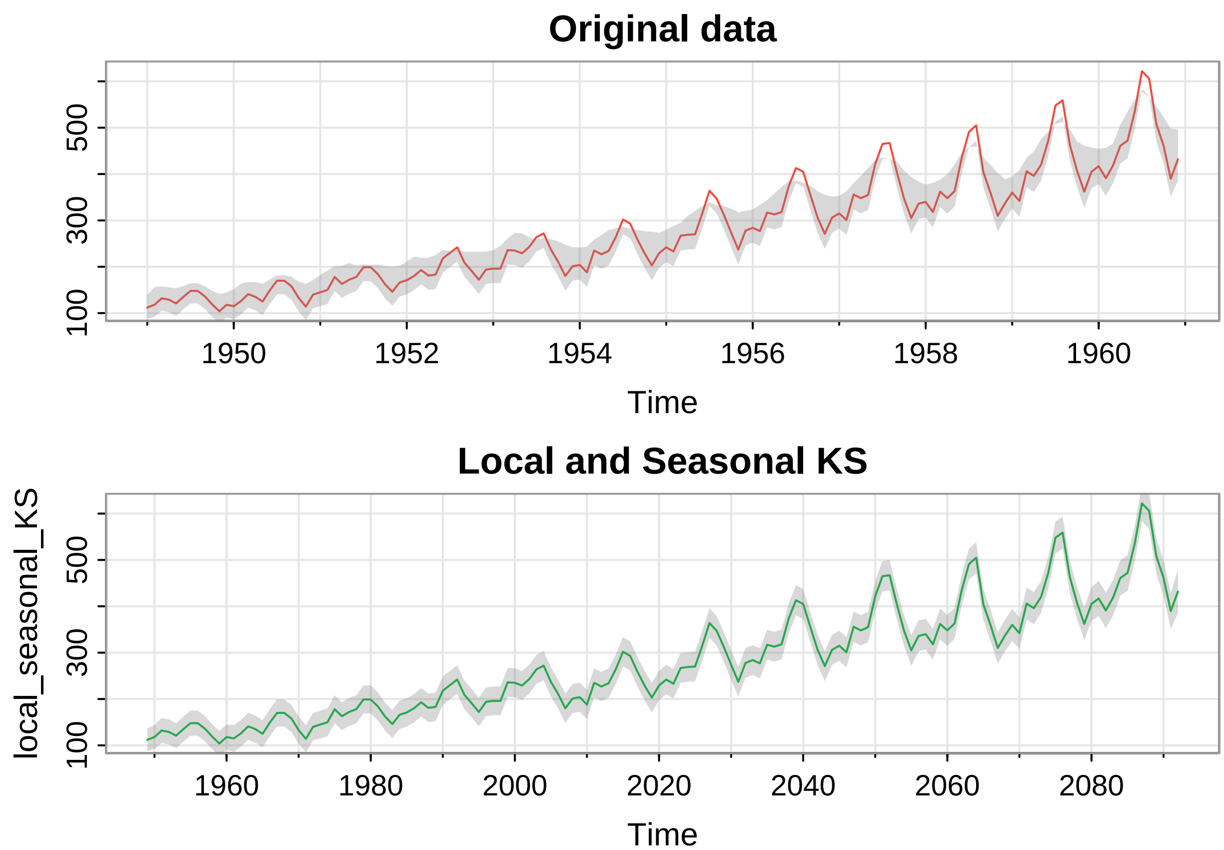
(c) Similar to Problem 1, smooth the state variables Xt; Draw three plots: The Kalman smoother estimators of (local level), the Kalman smoother estimators of (seasonal component), and the original data overlapped with the Kalman smoother estimators of (local level) & (seasonal component); Add confidence bands by 2× smoother mean square error each plot [Note: From (b), it may be not easy to find the initial estimates to make all estimates positive. In this case, use the absolute values of the estimates in (b)].

```r
# Ksmooth

# Smooth
A <- cbind(1,0,1,0,0,0,0,0,0,0,0,0,0)

Phi <- diag(0,13)
Phi[1,1] <- 1
Phi[1,2] <- 1
Phi[2,2] <- 1
Phi[3,] <- c(0,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1);
Phi[4,3] <- 1;
Phi[5,4] <- 1;
Phi[6,5] <- 1;
Phi[7,6] <- 1;
Phi[8,7] <- 1;
Phi[9,8] <- 1;
Phi[10,9] <- 1;
Phi[11,10] <- 1;
Phi[12,11] <- 1;
Phi[13,12] <- 1;

Q1 <- est$par[2]
Q2 <- est$par[3]
Q3 <- est$par[4]   # sqrt q11 and q22
QQ <- diag(0,13)
QQ[1,1] <- Q1 #Wt
QQ[2,2] <- Q2 #Zt
QQ[3,3] <- Q3 #Ut
```

```
RR <- est$par[1] # Vt

ks <- Ksmooth(data_air, A, mu0, Sigma0, Phi, QQ, RR)

local_KS = ts(as.numeric(ks$Xs[1,,]), start = 1949)
seasonal_KS = ts(as.numeric(ks$Xs[3,,]), start = 1949)

local_seasonal_KS = local_KS + seasonal_KS

# Plots
Tsm   <- ts(as.numeric(ks$Xs[1,,]), start = 1949)
Ssm   <- ts(as.numeric(ks$Xs[2,,]), start = 1949)

p1    <- 2*sqrt(ks$Ps[1,1,]); p2 = 2*sqrt(ks$Ps[3,3,])

par(mfrow=c(2,1))
tsplot(local_KS, main='Local Level Kalman Smoothers', ylab='')
xx  <- c(time(local_KS), rev(time(local_KS)))
yy  <- c(local_KS-p1, rev(local_KS+p1))
polygon(xx, yy, border=NA, col=gray(.5, alpha = .3))
tsplot(seasonal_KS, main='Seasonal Kalman Smoothers', ylab='')
xx  <- c(time(seasonal_KS), rev(time(seasonal_KS)))
yy  <- c((seasonal_KS)-(p2), rev((seasonal_KS)+(p2)))
polygon(xx, yy, border=NA, col=gray(.5, alpha = .3))
```

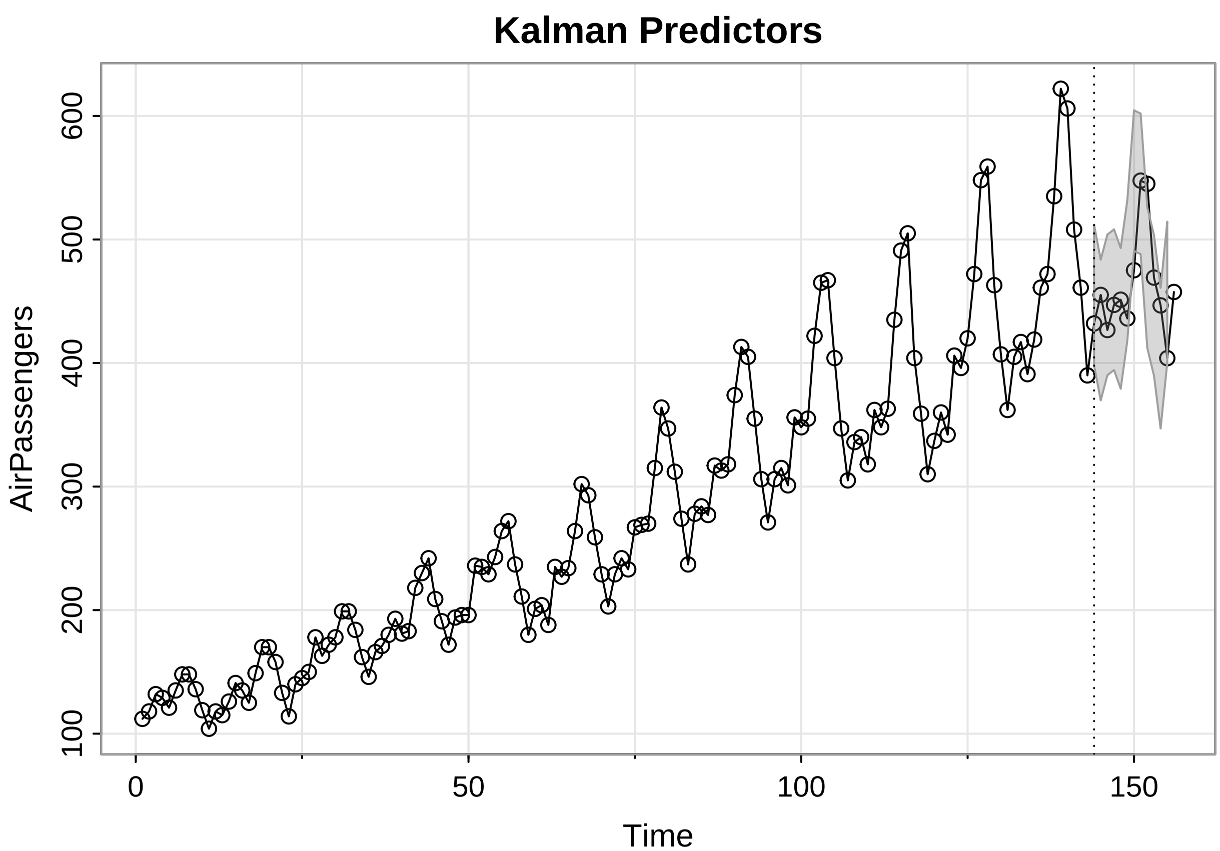


```
par(mfrow=c(2,1))
tsplot(data_air, main='Original data', ylab='', col=2)
xx  <- c(time(data_air), rev(time(data_air)))
yy  <- c(data_air-(p1+p2), rev(local_KS+(p1+p2)))
polygon(xx, yy, border=NA, col=gray(.5, alpha = .3))
tsplot(local_seasonal_KS,main='Local and Seasonal KS',col=3)
xx  <- c(time(local_seasonal_KS), rev(time(local_seasonal_KS)))
yy  <- c((local_seasonal_KS)-(p1+p2), rev((local_seasonal_KS)+(p1+p2)))
polygon(xx, yy, border=NA, col=gray(.5, alpha = .3))
```




```
# Forecast
num <- length(data_air)
n.ahead <- 12
y        <- ts(append(data_air, rep(0,n.ahead)))
rmspe    <- rep(0,n.ahead)
x00      <- ks$Xf[,,num]
P00      <- ks$Pf[,,num]
Q        <- t(QQ) %*% QQ
R        <- RR^2

for (m in 1:n.ahead){
       xp <- Phi%*%x00
       Pp <- Phi%*%P00%*%t(Phi)+Q
      sig <- A%*%Pp%*%t(A)+R
        K <- Pp%*%t(A)%*%(1/sig)
```

```
      x00 <- xp
      P00 <- Pp-K%*%A%*%Pp
 y[num+m] <- A%*%xp
 rmspe[m] <- sqrt(sig)
}

par(mfrow=c(1,1))
tsplot(y, type='o', main='Kalman Predictors', ylab='AirPassengers')
upp  <- ts(y[(num+1):(num+n.ahead)]+2*rmspe, start=num)
low  <- ts(y[(num+1):(num+n.ahead)]-2*rmspe, start=num)
 xx  <- c(time(low), rev(time(upp)))
 yy  <- c(low, rev(upp))
polygon(xx, yy, border=8, col=gray(.5, alpha = .3))
abline(v=length(data_air), lty=3)
```



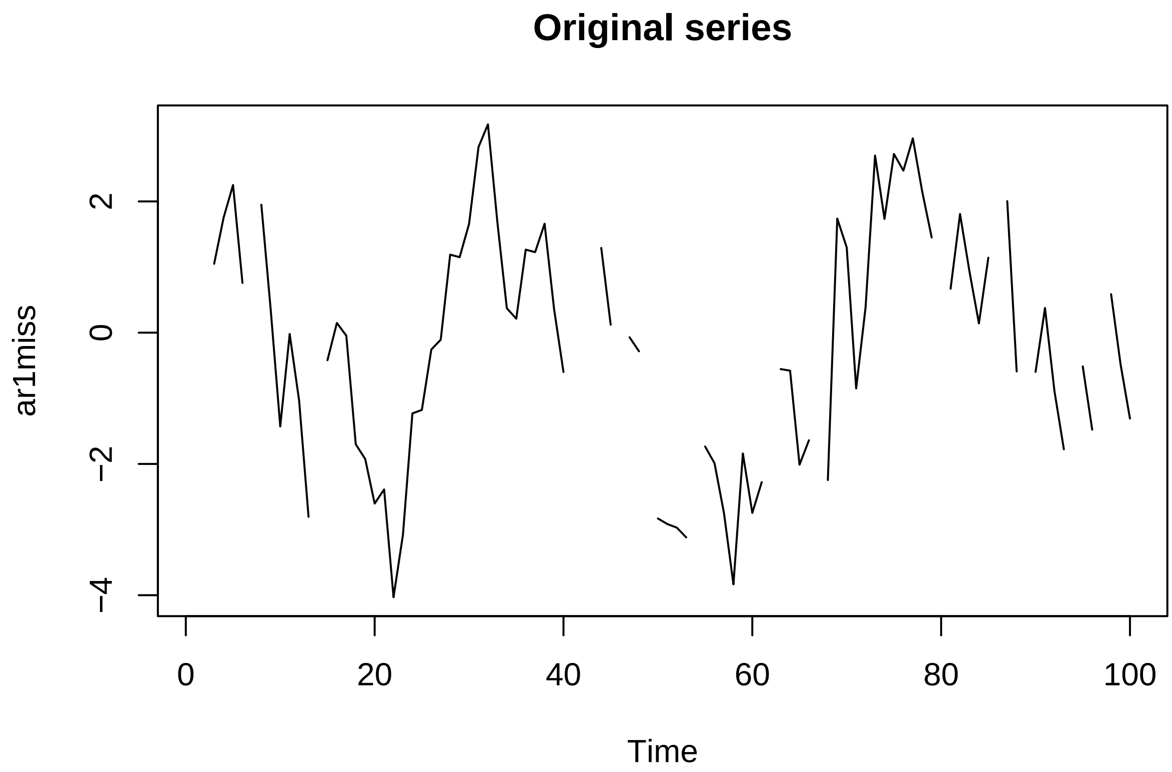
Question 3

Now consider the data ar1miss in the R package astsa. This data set has 100 observations generated from the AR(1) model with 1 = 0.9 and 2 Z = 1, where 10% of the observations have been deleted at random (replaced with NA). Use the EM algorithm and then estimate the missing values; Plot the Kalman smoother estimators, the original data, and the confidence bands by 3× smoother mean square error in a single plot; Verify for the time points that the observations are missing, the Kalman smoother estimators and the smoother mean square error are identical to the theoretical result for t = m in (2) and (3).

```
# Code prepared using the source:
# https://github.com/nickpoison/astsa/blob/master/fun_with_astsa/fun_with_astsa.md#8-state-space-models

ar1miss
```

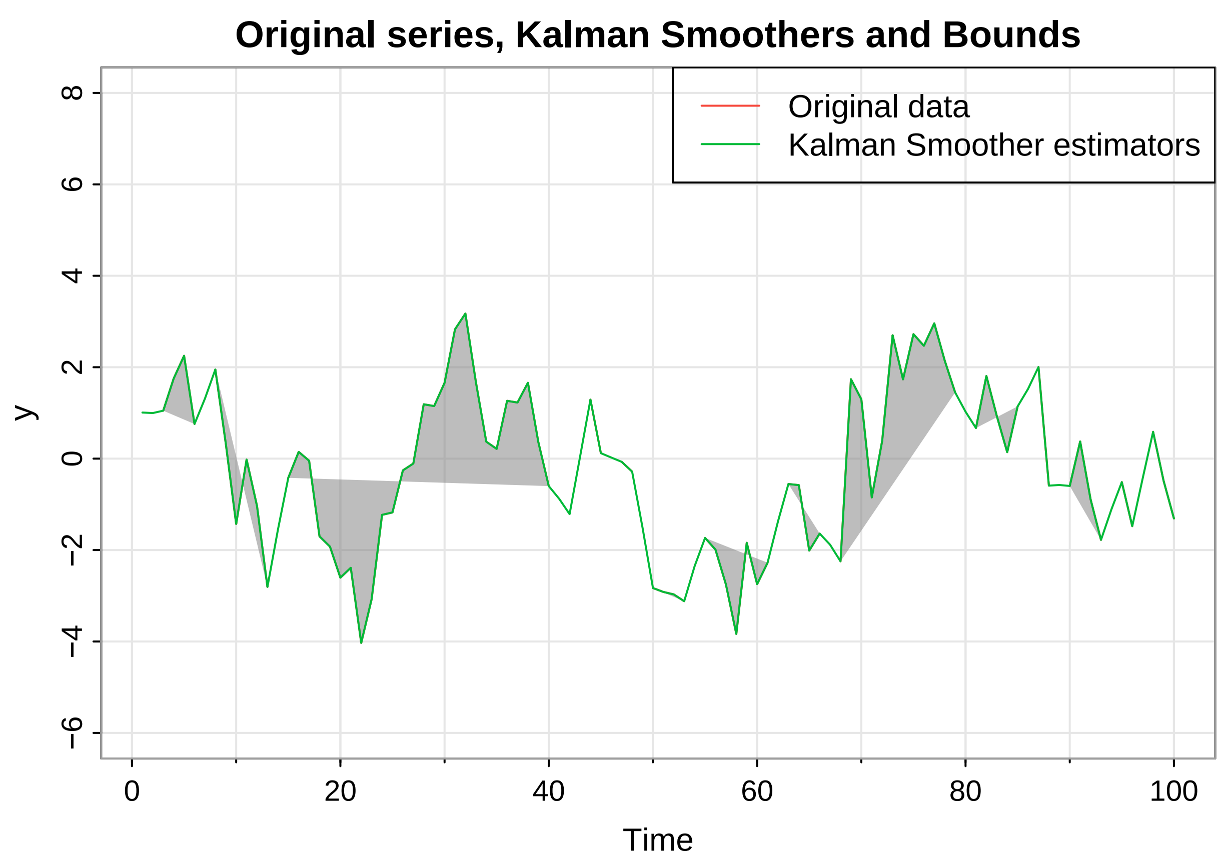
```
## Time Series:
## Start = 1
## End = 100
## Frequency = 1
##   [1]  1.008     NA  1.049  1.751  2.250  0.758     NA  1.950  0.336 -1.429
##  [11] -0.020 -1.035 -2.806     NA -0.420  0.148 -0.046 -1.701 -1.924 -2.604
##  [21] -2.388 -4.031 -3.076 -1.230 -1.177 -0.257 -0.107  1.189  1.150  1.658
##  [31]  2.828  3.174  1.683  0.371  0.213  1.265  1.227  1.660  0.363 -0.600
##  [41]     NA -1.214     NA  1.291  0.120     NA -0.070 -0.284     NA -2.831
##  [51] -2.916 -2.970 -3.119     NA -1.734 -1.990 -2.750 -3.835 -1.841 -2.746
##  [61] -2.277     NA -0.555 -0.579 -2.012 -1.639     NA -2.246  1.738  1.298
##  [71] -0.850  0.389  2.699  1.734  2.723  2.469  2.960  2.145  1.450     NA
##  [81]  0.669  1.808  0.931  0.141  1.143     NA  2.004 -0.591     NA -0.598
##  [91]  0.376 -0.890 -1.778     NA -0.513 -1.478     NA  0.587 -0.478 -1.309
```

```
plot(ar1miss, main="Original series")
```




```
y    <- ar1miss
num  <- length(y)
indicator <- array(1, dim=num)
A    <- array(0, dim=c(1,1,num))  # creates numxnum zero matrices

for(k in 1:num){
  if (!(is.na(y[k]))){
    A[1,1,k] = 1
    indicator[k] = 0
  }
}

# Initial values
mu0    <- 0
Sigma0 <- 1
Phi    <- 0.9
cQ     <- 0.01
cR     <- 0.00000000001 #R needs to be zero

for (i in 1:1000){
  invisible(capture.output(em <- EM(y, A, mu0, Sigma0, Phi, cQ, cR, max.iter = 1, tol = 0.1)))
  mu0    <- em$mu0
  Sigma0 <- em$Sigma0
  Phi    <- em$Phi
  cQ     <- em$Q
  cR     <- em$R
```

```
}

# Graph smoother
ks  <- Ksmooth(y, A, em$mu0, em$Sigma0, em$Phi, t(chol(em$Q)), t(chol(em$R)), NULL, NULL, NULL)

ys <- ks$Xs[1,,] # Kalman Smoothers
p1  <- 3*ks$Ps[1,1,] # smoother mean square error

par(mfrow=c(1,1))
tsplot(y, main='Original series, Kalman Smoothers and Bounds', col= 2, ylim=c(-6,8))
xx  <- c(time(y), rev(time(y)))
yy  <- c(y-p1, rev(y+p1))
polygon(xx, yy, border=NA, col=gray(.5, alpha = .3))
lines(ys, col = 3)
legend("topright",
       c("Original data", "Kalman Smoother estimators"),
       lty = 1,
       col = 2:5)
```




```
# comparisson Kalman smoother estimators

for (i in 1:num){
  if (indicator[i] == 1){
    cat(i, ys[i], (0.9/(1+0.9^2)) * (ys[i-1]+ys[i+1]), "\n")
  }
```

```
}
```

```
## 2 0.9968774 1.022818
## 7 1.312369 1.346519
## 14 -1.563406 -1.604088
## 41 -0.879113 -0.901989
## 43 0.03731626 0.03828729
## 46 0.02423134 0.02486188
## 49 -1.509613 -1.548895
## 54 -2.351894 -2.413094
## 62 -1.372463 -1.408177
## 67 -1.882775 -1.931768
## 80 1.026924 1.053646
## 86 1.525121 1.564807
## 89 -0.5762213 -0.5912155
## 94 -1.11028 -1.139171
## 97 -0.4318025 -0.4430387
```

We can see that for missing values, the Kalman Smoother estimators approximate their theoretical value.

```r
# comparisson smoother mean square errors

for (i in 1:num){
  if (indicator[i] == 1){
    cat(i, ks$Ps[1,1,i], (1^2/(1+0.9^2)), "\n")
  }
}
```

```
## 2 0.7446674 0.5524862
## 7 0.7446674 0.5524862
## 14 0.7446674 0.5524862
## 41 0.7446674 0.5524862
## 43 0.7446674 0.5524862
## 46 0.7446674 0.5524862
## 49 0.7446674 0.5524862
## 54 0.7446674 0.5524862
## 62 0.7446674 0.5524862
## 67 0.7446674 0.5524862
## 80 0.7446674 0.5524862
## 86 0.7446674 0.5524862
## 89 0.7446674 0.5524862
## 94 0.7446674 0.5524862
## 97 0.7446674 0.5524862
```

We can see that for missing values, the smoother mean square errors approximate their theoretical value.

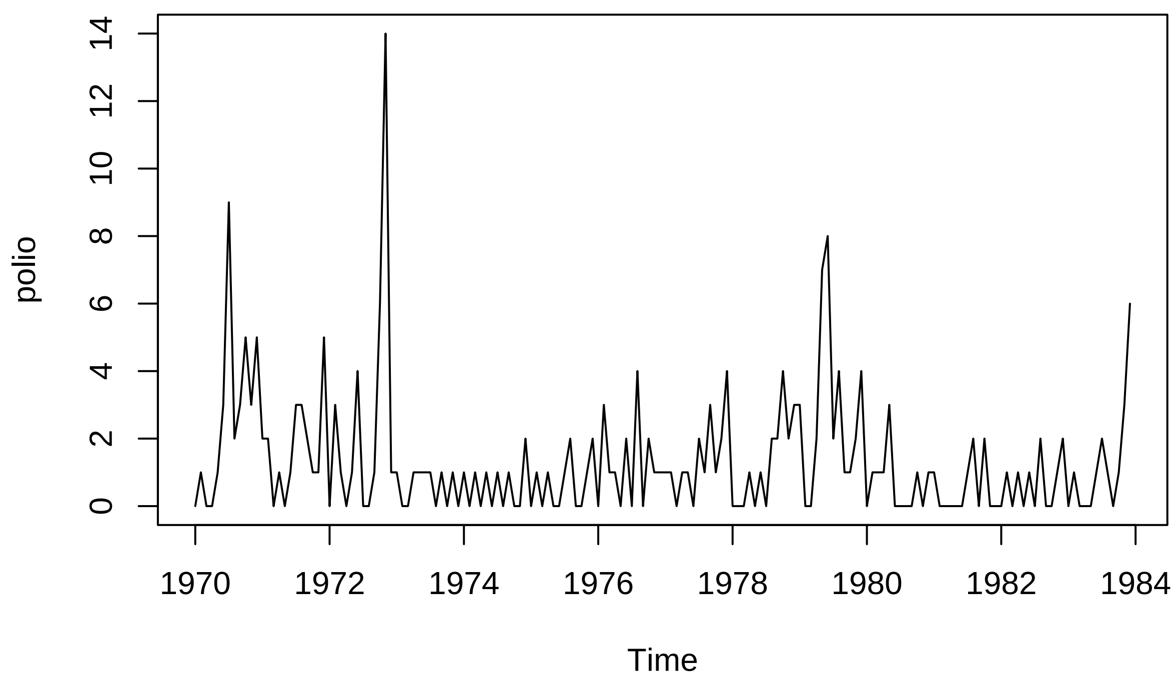
Question 4

Consider the dataset polio in the R package gamlss.data. Do the following:

(a) Draw a time plot of the data. Based on the plot, argue how many states of Xt seem to be required.

I argue that there are 2 states, eyeballing the means we set inital values of 1 and 2.

```r
plot.ts(polio)
```



(b) With proper starting values for the parameters of the response models, use set.seed(123) to fit a Poisson-HMM to the data; Compute stationary probabilities of the states; Check the overdispersion of the model numerically.

```r
# packageurl <- "https://cran.r-project.org/src/contrib/depmixS4_1.5-0.tar.gz"

# install.packages(packageurl)
# install.packages('depmixS4')

library(depmixS4)
```

```
## Warning: package 'depmixS4' was built under R version 4.3.3
```

```
## Loading required package: nnet
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 4.3.2

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## Loading required package: Rsolnp

## Warning: package 'Rsolnp' was built under R version 4.3.2

## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## The following object is masked from 'package:forecast':
##
##     getResponse
```

```
set.seed(123)

model <- depmix(polio ~ 1, nstates = 2, data=data.frame(polio), family=poisson('identity'),respstart=c(
fm <- fit(model)
```

```
## converged at iteration 38 with logLik: -260.0327
```

```
fm
```

```
## Convergence info: Log likelihood converged to within tol. (relative change)
## 'log Lik.' -260.0327 (df=5)
## AIC:  530.0655
## BIC:  545.6853
```

```
summary(fm)
```

```
## Initial state probabilities model
## pr1 pr2
##   1   0
##
## Transition matrix
##         toS1  toS2
## fromS1 0.932 0.068
## fromS2 0.330 0.670
```

```
##
## Response parameters
## Resp 1 : poisson
##     Re1.(Intercept)
## St1           0.790
## St2           4.178
```

```
standardError(fm)
```

```
##          par constr         se
## 1 1.00000000    bnd         NA
## 2 0.00000000    bnd         NA
## 3 0.93218283    inc 0.03300657
## 4 0.06781717    inc 0.03300657
## 5 0.33046261    inc 0.12501672
## 6 0.66953739    inc 0.12501672
## 7 0.79021754    inc 0.12028992
## 8 4.17819743    inc 0.69511952
```

```
##-- A little nicer display of the parameters --##
para.mle <- as.vector(getpars(fm))[3:8]
mtrans <- matrix(para.mle[1:4], byrow=TRUE, nrow=2)
lams   <- para.mle[5:6]
pi1    <- mtrans[2,1]/(2 - mtrans[1,1] - mtrans[2,2])
pi2    <- 1 - pi1

mean_Yt <- pi1*lams[1]+pi2*lams[2]

var_Yt <- mean_Yt+pi1*pi2*(lams[1]^2+lams[2]^2-2*lams[1]*lams[2])

c(mean_Yt, var_Yt)
```

```
## [1] 1.367106 2.988794
```

We have that the probability of the first state is 0.8297248 and the probability of the second state is 0.1702752. We can see that $Var(Y_t) \approx 2.9887938 > E(Y_t) \approx 1.3671065$, this phenomenon is called overdispersion.

(c) By referring to the counts of earthquakes example, draw three plots: A time plot of the data and estimated states, HMM smoothing probabilities of state 1, and a histogram of the data with the two estimated Poisson densities.

```
#-- Graphics --##
par(mfrow=c(3,1))
# data and states
tsplot(polio, main="", ylab='polio', type='h', col=gray(.7), ylim=c(0,50))
text(polio, col=6*posterior(fm)[,1]-2, labels=posterior(fm)[,1])
```

```
## Warning in .local(object, ...): Argument 'type' not specified and will default
## to 'viterbi'. This default may change in future releases of depmixS4. Please
## see ?posterior for alternative options.
```

```
## Warning in .local(object, ...): Argument 'type' not specified and will default
## to 'viterbi'. This default may change in future releases of depmixS4. Please
## see ?posterior for alternative options.
```

```r
# prob of state 2
tsplot(ts(posterior(fm)[,2], start=1900), ylab = expression(hat(pi)[~2]*'(t|n)'));  abline(h=.5, lty=2)
```

```
## Warning in .local(object, ...): Argument 'type' not specified and will default
## to 'viterbi'. This default may change in future releases of depmixS4. Please
## see ?posterior for alternative options.
```

```r
# histogram
hist(polio, breaks=30, prob=TRUE, main="")
xvals <- seq(1,45)
u1 <- pi1*dpois(xvals, lams[1])
u2 <- pi2*dpois(xvals, lams[2])
lines(xvals, u1, col=4)
lines(xvals, u2, col=2)
```