

MATH 545: INTRODUCTION TO TIME SERIES ANALYSIS
WINTER 2025
THE DLM PACKAGE

```
library(dlm)
library(forecast)
```

EXAMPLE: Fitting an AR(1) model.

Consider the model

$$Y_t = 0.8Y_{t-1} + Z_t$$

This model has a straightforward state space representation

OBSERVATION EQUATION	Y_t	$=$	$X_t + V_t$
STATE EQUATION	X_{t+1}	$=$	$\phi X_t + W_t$

where $W_t \equiv 0$ and $V_t \equiv Z_t$.

```
require(methods)
n <- 250
yt <- arima.sim(n=n,list(ar=.8,ma=0),sd=.5)

# estimate AR(1) for comparison
model10 <- Arima(yt,order=c(1,0,0),method="ML",include.mean=FALSE)
model10

+ Series: yt
+ ARIMA(1,0,0) with zero mean
+
+ Coefficients:
+      ar1
+    0.7755
+ s.e.  0.0397
+
+ sigma^2 = 0.2282:  log likelihood = -170
+ AIC=344.01  AICc=344.06  BIC=351.05

# Restrict to stationary causal case
parm_rest <- function(parm){
  return( c(exp(parm[1])/(1+exp(parm[1])),exp(parm[2])) )
}

# set up SS model for use in dlm
ssm1 <- function(parm){
  parm <- parm_rest(parm)
  return( dlm(FF=1,V=0,GG=parm[1],W=parm[2],m0=0,C0=solve(1-parm[1]^2)*parm[2]) )
}

fit1 <- dlmMLE(y=yt,parm=c(0,1),build=ssm1,hessian=T) # estimate parameters
coef <- parm_rest(fit1$par)
dg1 <- exp(fit1$par[1])/(1+exp(fit1$par[1]))^2 # get standard errors
dg2 <- exp(fit1$par[2])
dg <- diag(c(dg1,dg2))
var <- dg%*%solve(fit1$hessian)%*%dg
coef; sqrt(diag(var))

+ [1] 0.7754771 0.2272923
+ [1] 0.03969152 0.02033013
```

Using the ARMA `dlmModARMA` formulation in `dlm`

```
require(methods)
parm_rest <- function(parm){
  return( c(exp(parm[1])/(1+exp(parm[1])),exp(parm[2])) )
}

ssm2 <- function(parm){
  parm <- parm_rest(parm)
  dlm <- dlmModARMA(ar=parm[1], ma=NULL, sigma2=parm[2])
  dlm$C0 <- solve(1-parm[1]^2)*parm[2]
  return(dlm)
}

fit2 <- dlmMLE(yt,parm=c(0,1),build=ssm2,hessian=T)
coef <- parm_rest(fit2$par)
dg1 <- exp(fit2$par[1])/(1+exp(fit2$par[1]))^2
dg2 <- exp(fit2$par[2])
dg <- diag(c(dg1,dg2))
var <- dg%*%solve(fit2$hessian)%*%dg
coef; sqrt(diag(var))

+ [1] 0.7754771 0.2272923
+ [1] 0.03969152 0.02033013
```

EXAMPLE: Fitting an MA(1) model.

Consider the model

$$Y_t = Z_t + \theta Z_{t-1}$$

with $\theta = 0.6$. This model has a straightforward state space representation with $L = 2$

$$\begin{array}{lll} \text{OBSERVATION EQUATION} & Y_t & = \mathbf{F}\mathbf{X}_t + V_t \\ \text{STATE EQUATION} & \mathbf{X}_{t+1} & = \mathbf{G}\mathbf{X}_t + \mathbf{W}_{t+1} \end{array}$$

with state $\mathbf{X}_t = (X_{1t}, \theta Z_t)^\top$, where $V_t \equiv 0$ and

$$\mathbf{F} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \mathbf{W}_t = \begin{bmatrix} 1 \\ \theta \end{bmatrix} Z_{t+1}$$

that is, for the observation equation, we have $Y_t = X_{1t}$, and for the state equation we have

$$\begin{bmatrix} X_{1(t+1)} \\ \theta Z_{t+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_{1t} \\ \theta Z_t \end{bmatrix} + \begin{bmatrix} 1 \\ \theta \end{bmatrix} Z_{t+1}$$

With $\theta = 0.6$, we have the following results:

```
require(methods)
yt <- arima.sim(n=n,list(ma=0.6),sd=.5)

model01 <- Arima(yt,order=c(0,0,1),method="ML",include.mean=FALSE);model01

+ Series: yt
+ ARIMA(0,0,1) with zero mean
+
+ Coefficients:
+      ma1
+    0.5866
+ s.e. 0.0508
+
+ sigma^2 = 0.2427: log likelihood = -177.46
+ AIC=358.91 AICc=358.96 BIC=365.95
```

```

# set parameter restrictions
parm_rest <- function(parm){
  return( c(exp(parm[1])/(1+exp(parm[1])),exp(parm[2])) )
}

ssm3 <- function(parm){
  parm <- parm_rest(parm)
  return( dlmModARMA(ar=NULL, ma=parm[1], sigma2=parm[2]) )
}

# estimate parameters
fit3 <- dlmMLE(y=yt,parm=c(0,1),build=ssm3,hessian=T)

# get estimates
coef <- parm_rest(fit3$par)
# get standard errors using delta method
dg1 <- exp(fit3$par[1])/(1+exp(fit3$par[1]))^2
dg2 <- exp(fit3$par[2])
dg <- diag(c(dg1,dg2))
var <- dg%*%solve(fit3$hessian)%*%dg
# print results
coef; sqrt(diag(var))

+ [1] 0.5900676 0.2403592
+ [1] 0.03396860 0.01812424

```

EXAMPLE: Regression

The state space formulation can also be used for regression modelling. For a linear regression,

$$\begin{array}{ll}
 \text{OBSERVATION EQUATION} & Y_t = \mathbf{F}\mathbf{X}_t + V_t \\
 \text{STATE EQUATION} & \mathbf{X}_{t+1} = \mathbf{G}\mathbf{X}_t + \mathbf{W}_{t+1}
 \end{array}$$

where V_t is the residual error, and

$$\mathbf{F} = [\beta_0 \quad \beta_1] \quad \mathbf{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{W}_t = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

and $\mathbf{X}_1 = (1, 1)^\top$, so that

$$\begin{bmatrix} X_{1(t+1)} \\ X_{1t} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_{1t} \\ X_{1(t-1)} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

We use the function `dlmModReg`.

```

require(methods)
n <- 250
be0 <- 4; be1 <- 2
# simulate data
xt <- rnorm(n,mean=0,sd=1)
yt <- be0 + be1 * xt + rnorm(n,mean=0,sd=.5)

fit4 <- lm(yt~xt) #Fit using lm
summary(fit4)

+
+ Call:
+ lm(formula = yt ~ xt)
+
+ Residuals:
+      Min       1Q   Median       3Q      Max
+ -1.32969 -0.35712 -0.02736  0.37854  1.43011
+
+ Coefficients:

```

```

+               Estimate Std. Error t value      Pr(>|t|)
+ (Intercept)  3.98769    0.03212   124.16 <0.0000000000000002 ***
+ xt           1.97331    0.03254    60.65 <0.0000000000000002 ***
+ ---
+ Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
+
+ Residual standard error: 0.5076 on 248 degrees of freedom
+ Multiple R-squared:  0.9368, Adjusted R-squared:  0.9366
+ F-statistic: 3678 on 1 and 248 DF, p-value: < 0.00000000000000022

# set parameter restrictions
parm_rest <- function(parm){
  return( exp(parm[1]) )
}

ssm4 <- function(parm,x.mat){
  parm <- parm_rest(parm)
  return( dlmModReg(X=x.mat, dV=parm[1]) )
}
fit4 <- dlmMLE(y=yt,parm=1,x.mat=xt,build=ssm4,hessian=T) #Using dlmMLE
se2 <- parm_rest(fit4$par)# get residual variance estimate
sqrt(se2)

+ [1] 0.5076232

LSMod <- ssm4(fit4$par,xt)
LSf <- dlmFilter(yt,LSMod)

coef <- LSf$m[n+1,]
covar <- dlmSvd2var(LSf$U.C[[n+1]],LSf$D.C[n+1,])
coef.se <- c(sqrt(covar[1,1]),sqrt(covar[2,2]))
coef; coef.se

+ [1] 3.987694 1.973306
+ [1] 0.03211714 0.03253614

```

It is easy to combine models to obtain a linear regression with ARMA residual errors:

```

require(methods)
# simulate AR(2) process + time trend: AR(2) with phi1 = .6, phi2 = .2, sig2 = 1
n <- 250; tt <- as.numeric(seq(1,n))
be0 <- 1; be1 <- .1
yt <- be0 + be1*tt + arima.sim(n=n,list(ar=c(.6,.2),ma=0),sd=1)

# set parameter restrictions (only variances here)
parm_rest <- function(parm){
  return( c(parm[1],parm[2],exp(parm[3])) )
}
ssm6 <- function(parm,x.mat){
  parm <- parm_rest(parm)
  dlm <- dlmModReg(X=x.mat, addInt=TRUE, dV=1e-7) +
    dlmModARMA(ar=c(parm[1],parm[2]), ma=NULL, sigma2=parm[3])
  # get distribution variance of initial state
  tmp0 <- matrix(c(parm[1],parm[2],1,0),nr=2); tmp1 <- matrix(c(parm[3],0,0,0),nc=1)
  tmp <- solve(diag(4)-tmp0%x%tmp0)%*%tmp1
  dlm$C0[3:4,3:4] <- matrix(tmp,nr=2)
  return( dlm )
}
fit6 <- dlmMLE(y=yt,parm=c(0,0,1),x.mat=tt,build=ssm6,hessian=T)
coef <- parm_rest(fit6$par)

```

```

dg1 <- 1;dg2 <- 1;dg3 <- exp(fit6$par[3]);dg <- diag(c(dg1,dg2,dg3))
var <- dg%*%solve(fit6$hessian)%*%dg
# print results
coef; sqrt(diag(var))

+ [1] 0.59373903 0.07569234 0.99634144
+ [1] 0.06359096 0.06368269 0.08946985

# get parameter estimates (slope and intercept)
# these are the last filtered values of a
LSARMod <- ssm6(fit6$par,tt)
LSARf <- dlmFilter(yt,LSARMod)

# get parameters
coef <- LSARf$m[n+1,1:2]
covar <- dlmSvd2var(LSARf$U.C[[n+1]],LSARf$D.C[n+1,])
coef.se <- c(sqrt(covar[1,1]),sqrt(covar[2,2]))
coef; coef.se

+ [1] 0.8128630 0.1027248
+ [1] 0.37464564 0.00257624

# Same ARMA model + time trend using dlmModPoly + dlmModARMA
parm_rest <- function(parm){
  return( c(parm[1],parm[2],exp(parm[3])) )
}

ssm7 <- function(parm){
  parm <- parm_rest(parm)
  dlm <- dlmModPoly(2,dV=1e-7,dW=c(0,0)) +
    dlmModARMA(ar=c(parm[1],parm[2]), ma=NULL, sigma2=parm[3])
  # get distribution variance of initial state
  tmp0 <- matrix(c(parm[1],parm[2],1,0),nr=2);tmp1 <- matrix(c(parm[3],0,0,0),nc=1)
  tmp <- solve(diag(4)-tmp0%x%tmp0)%*%tmp1
  dlm$C0[3:4,3:4] <- matrix(tmp,nr=2)
  return( dlm )
}
fit7 <- dlmMLE(y=yt,parm=c(0,0,1),build=ssm7,hessian=T)

coef <- parm_rest(fit7$par)
dg1 <- 1;dg2 <- 1;dg3 <- exp(fit7$par[3]);dg <- diag(c(dg1,dg2,dg3))
var <- dg%*%solve(fit7$hessian)%*%dg
coef; sqrt(diag(var))

+ [1] 0.59373615 0.07570283 0.99633467
+ [1] 0.06360454 0.06371322 0.08949775

LSARMod2 <- ssm7(fit7$par) # get parameter estimates for slope and intercept
LSARf2 <- dlmFilter(yt,LSARMod2)
coef <- LSARf2$m[n+1,1:2]
coef[1] <- coef[1]-n*coef[2]
covar <- dlmSvd2var(LSARf2$U.C[[n+1]],LSARf2$D.C[n+1,])
coef.se <- c(sqrt(covar[1,1]),sqrt(covar[2,2]))
coef; coef.se

+ [1] 0.8128629 0.1027248
+ [1] 0.372431552 0.002576287

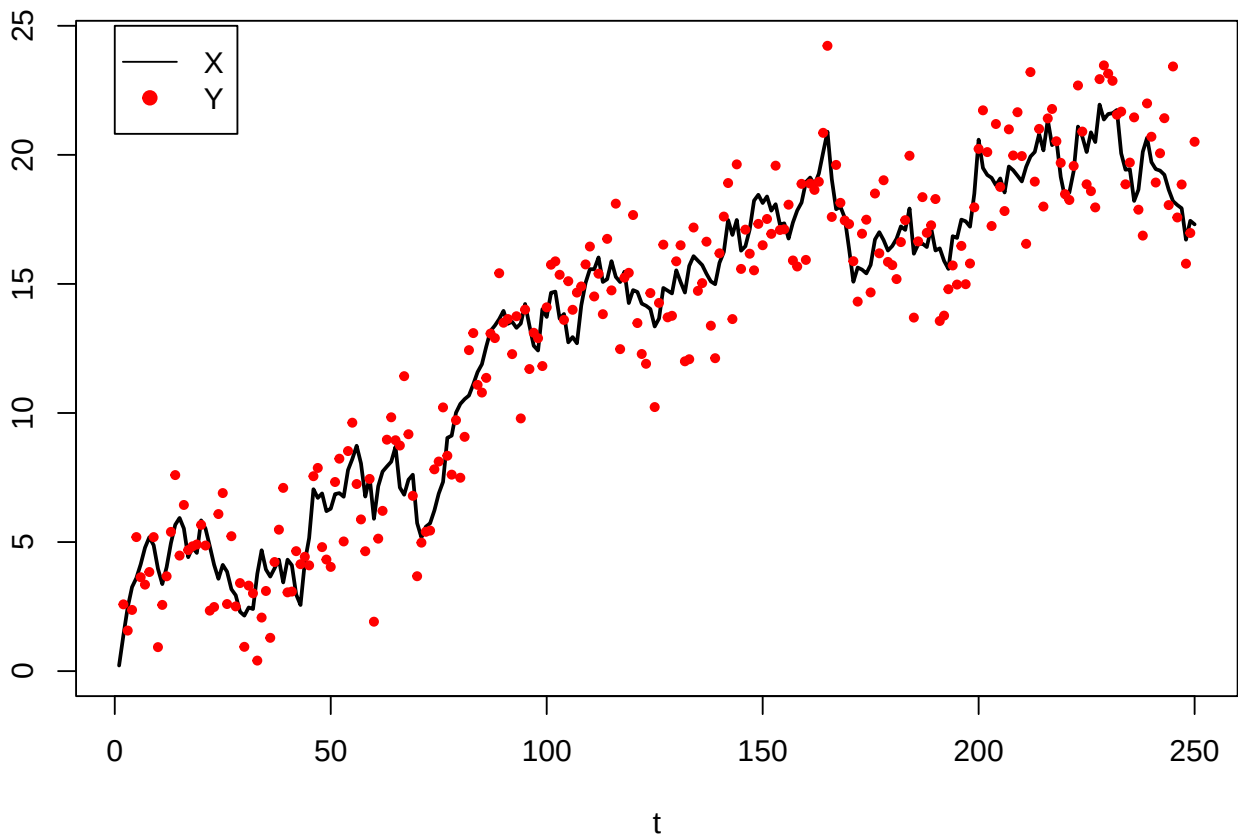
```

EXAMPLE: LOCAL LEVEL MODEL

For the local level model (random walk plus noise), we have

$$\begin{array}{lll} \text{OBSERVATION EQUATION} & Y_t & = X_t + V_t \\ \text{STATE EQUATION} & X_{t+1} & = X_t + W_t \end{array}$$

```
require(methods)
n <- 250; xt <- rep(n)
vt <- sqrt(3)*rnorm(n) ; wt <- sqrt(.5)*rnorm(n)
xt[1] <- wt[1]; for(i in 2:n){ xt[i] <- xt[i-1]+wt[i] }
yt <- as.ts(xt) + vt
par(mar=c(4,4,1,0))
plot(xt,type='l',main="",lwd=2,xlab='t',ylab='',ylim=range(0,max(yt)))
points(yt,pch=19,cex=0.6,col='red')
legend(0,25,c('X','Y'),col=c('black','red'),pch=c(-1,19),lty=c(1,-1))
```



```
parm_rest <- function(parm){ # set parameter restrictions (only variances here)
  return( exp(parm) )
}
ssm8 <- function(parm){
  parm <- parm_rest(parm)
  return( dlm(F=1,V=parm[2],GG=1,W=parm[1],m0=0,C0=10^7) )
}
fit8 <- dlmMLE(y=yt,parm=c(1,1),build=ssm8,hessian=T)
se2 <- parm_rest(fit8$par)
se2
+ [1] 0.4384317 2.8197198
```

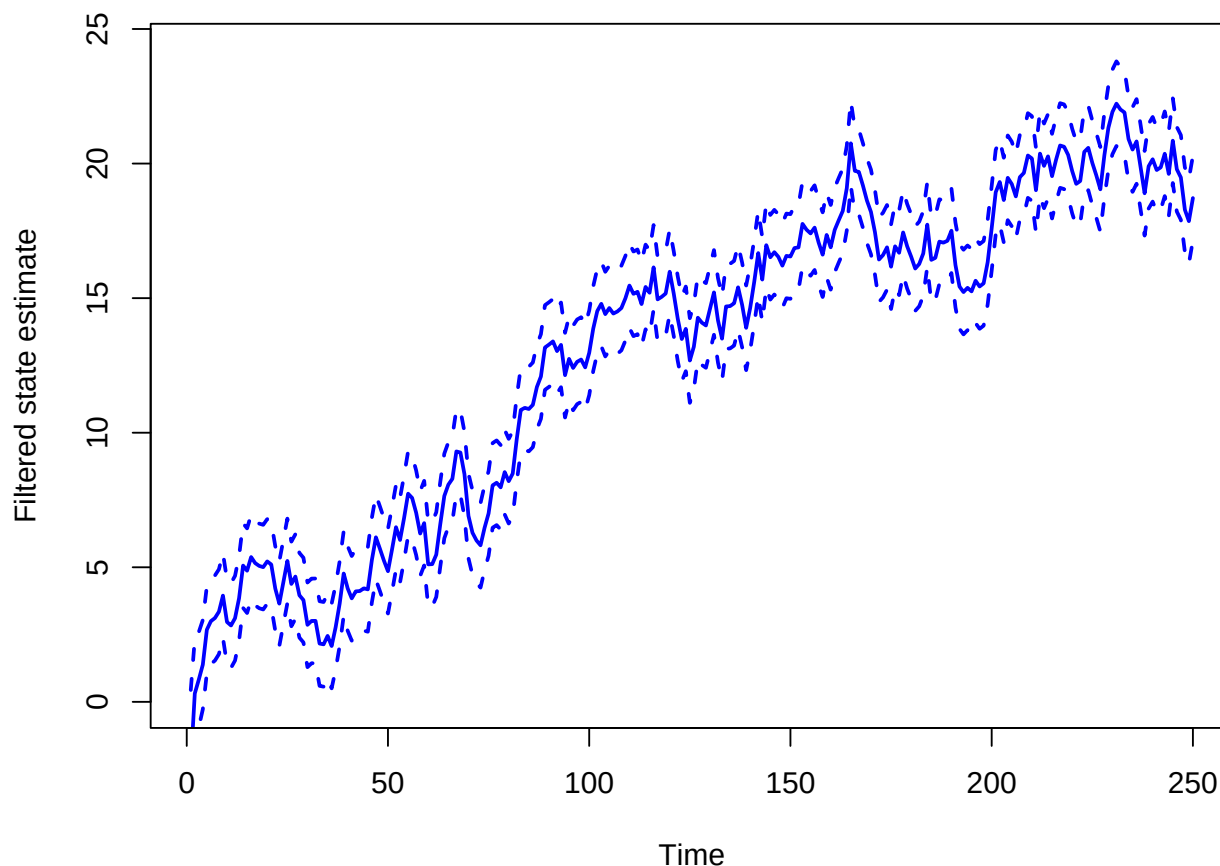
```

mod8 <- ssm8(fit8$par)           # filter and smooth and get standard errors
mod8f <- dlmFilter(yt,mod8)      # filtered estimates
mod8s <- dlmSmooth(mod8f)       # smoothed estimates
xtfilt <- ts(mod8f$m[-1],start=1) # filtered values as ts object
sefilt <- dlmSvd2var(mod8f$U.C,mod8f$D.C)[-1] # width of 90% confidence interval
width <- qnorm(.95) * sqrt(sapply(sefilt,diag))
xtfiltered <- cbind(xtfilt,as.vector(xtfilt)+width%o%c(1,-1))
xtsmooth <- ts(mod8s$m[-1],start=1) # smoothed values
sesmooth <- dlmSvd2var(mod8s$U.S,mod8s$D.S)[-1] # width of 90% confidence interval
width <- qnorm(.95) * sqrt(sapply(sesmooth,diag))
xtsmoothed <- cbind(xtsmooth,as.vector(xtsmooth)+width%o%c(1,-1))

plot(xtfiltered,plot.type='s',lwd=c(2,2,2), lty=c(1,2,2), ylab='Filtered state estimate',
     col=c("blue","blue","blue"),main="Filtered state + 90% interval", ylim=range(0,max(yt)))

```

Filtered state + 90% interval

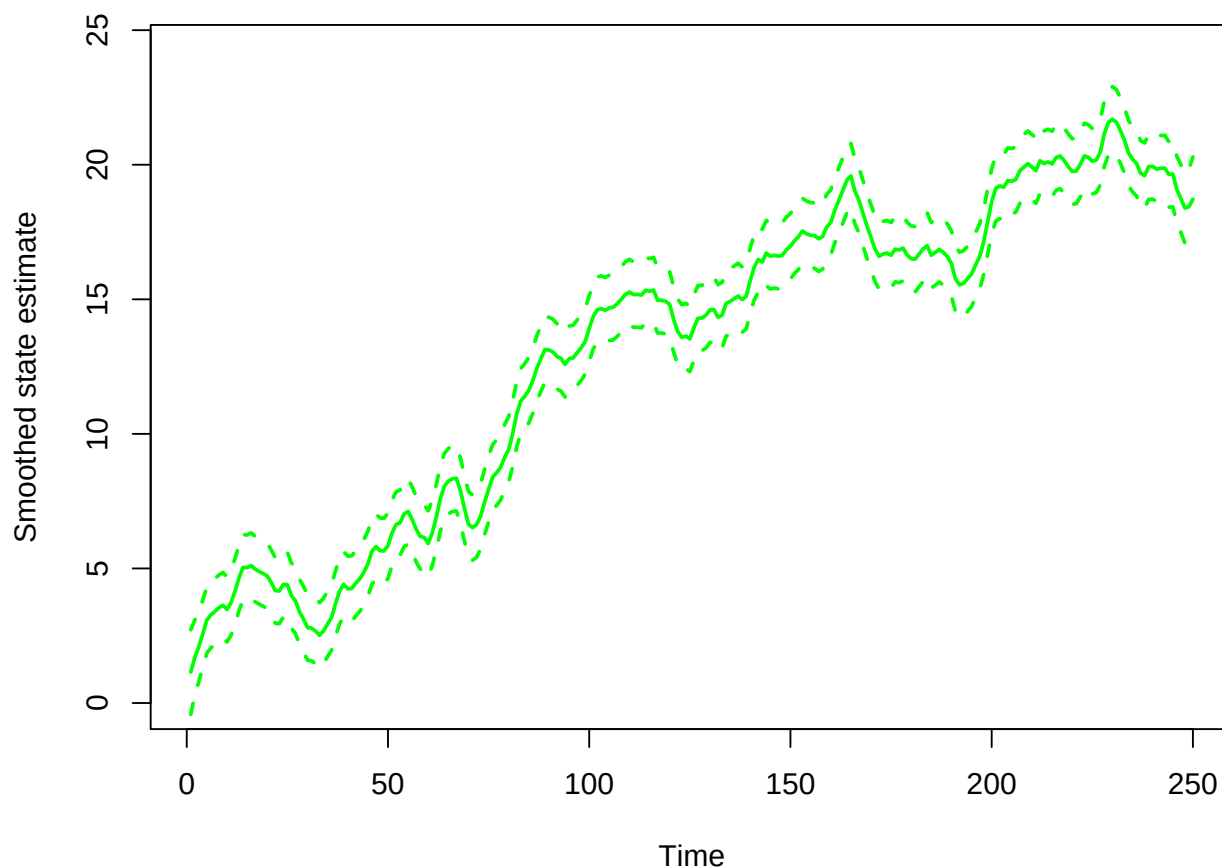


```

plot(xtsmoothed,plot.type='s',lwd=c(2,2,2), lty=c(1,2,2), ylab='Smoothed state estimate',
     col=c("green","green","green"),main="Smoothed state + 90% interval",ylim=range(0,max(yt)))

```

Smoothed state + 90% interval



It is also possible to get forecasts from the model by extrapolating from the state equation

```
# simulate AR(1) process
# phi = .8, sig2 = .25
require(methods)
yt <- arima.sim(n=n, list(ar=.8, ma=0), sd=.5)

parm_rest <- function(parm){
  return( c(exp(parm[1])/(1+exp(parm[1])), exp(parm[2])) )
}
ssm9 <- function(parm){
  parm <- parm_rest(parm)
  dlm <- dlmModARMA(ar=parm[1], ma=NULL, sigma2=parm[2])
  dlm$C0 <- solve(1-parm[1]^2)*parm[2]
  return(dlm)
}

# Fit model using 230 observations and forecast next 20 observations using dlmForecast
xt <- yt[1:230]
fit9 <- dlmMLE(y=xt, parm=c(0,1), build=ssm9, hessian=T)

mod9 <- ssm9(fit9$par)
mod9f <- dlmFilter(xt, mod9)
fore9 <- dlmForecast(mod9f, nAhead=20, method="plain")
```

It is also possible to handle forecasting for missing values:

```
# Case 2: fit model using 250 observations where the last 20 are missing
require(methods)
zt <- matrix(NA,250,1);zt[1:230] <- xt
fit10 <- dlmMLE(y=zt,param=c(0,1),build=ssm9,hessian=T)

mod10 <- ssm1(fit10$par)      # filter
mod10f <- dlmFilter(zt,mod10)
fore10 <- mod10f$f[231:250] # forecast are last observations
fore.all <- cbind(fore9$f,fore10,fore9$f-fore10) # compare results
fore.all

+               fore10
+ [1,] -0.393929116 -0.393929116 0
+ [2,] -0.310580252 -0.310580252 0
+ [3,] -0.244866625 -0.244866625 0
+ [4,] -0.193056910 -0.193056910 0
+ [5,] -0.152209272 -0.152209272 0
+ [6,] -0.120004316 -0.120004316 0
+ [7,] -0.094613394 -0.094613394 0
+ [8,] -0.074594770 -0.074594770 0
+ [9,] -0.058811754 -0.058811754 0
+ [10,] -0.046368163 -0.046368163 0
+ [11,] -0.036557429 -0.036557429 0
+ [12,] -0.028822484 -0.028822484 0
+ [13,] -0.022724125 -0.022724125 0
+ [14,] -0.017916077 -0.017916077 0
+ [15,] -0.014125332 -0.014125332 0
+ [16,] -0.011136646 -0.011136646 0
+ [17,] -0.008780317 -0.008780317 0
+ [18,] -0.006922548 -0.006922548 0
+ [19,] -0.005457851 -0.005457851 0
+ [20,] -0.004303061 -0.004303061 0
```