SSC PROBABILITY WORKSHOP
MAY 2025

PARTICLE FILTERING FOR STATE SPACE MODELS

For $t = 1, \ldots, n$, consider the state space model takes the form

$$\begin{aligned} Y_t &= f(X_t; \theta) + V_t \\ X_{t+1} &= g(X_t; \theta) + W_t \end{aligned}$$

where only the data $y_1, \ldots, y_n$, is observed. Sequences $\{V_t\}$ and $\{W_t\}$ are independent zero mean random variables. If $y_{1:t} = (y_1, \ldots, y_t)$, $x_{1:t} = (x_1, \ldots, x_t)$ we must carry out for $t = 1, 2, \ldots, n$ the following calculations.

- **Filtering:** Compute $p(x_t|y_{1:t})$: By Bayes theorem

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:(t-1)})}{p(y_t|y_{1:(t-1)})} \propto p(y_t|x_t)p(x_t|y_{1:(t-1)})$$

  where the function $p(y_t|x_t)$ is usually a straightforward conditional density, but the function $p(x_t|y_{1:(t-1)})$ is the density computed at the previous time point **prediction** step.

- **Prediction:** Compute $p(x_{t+1}|y_{1:t})$: we have that

$$p(x_{t+1}|y_{1:t}) = \int p(x_{t+1}|x_t)p(x_t|y_{1:t})\, dx_t$$

  where $p(x_{t+1}|x_t)$ is again a conditional density derived from the state equation, and $p(x_t|y_{1:t})$ is the posterior density computed at the time point **filtering** step.

We consider simulation-based versions of these calculations. The fundamental idea is that we may approximate any density $p(x)$ by the discrete approximation

$$\widehat{p}(x) = \sum_{i=1}^{N} w_i \delta_{x^{(i)}}$$

where $\{\delta_{x^{(i)}}, i = 1, \ldots, N\}$ is a collection of point masses at the values $\{x^{(i)}, i = 1, \ldots, N\}$ and $\{w_i, i = 1, \ldots, N\}$ is a collection of probabilities such that $0 \leq w_i \leq 1$ and

$$\sum_{i=1}^{n} w_i = 1.$$

More generally, we may obtain a *kernel* approximation in the same fashion: we estimate $p(x)$ as

$$\widehat{p}(x) = \sum_{i=1}^{N} w_i p(x|x^{(i)})$$

where $p(x|x^{(i)})$ is a specified conditional density. Note that the selection

$$w_i = \frac{1}{N} \qquad i = 1, \ldots, N$$

is legitimate.

For state space model calculations:

- In the **filtering** step, suppose we have a sample

$$x_t^{(1)}, \ldots, x_t^{(N)}$$

from $p(x_t|y_{1:(t-1)})$ obtained at the previous prediction step. We can obtain a kernel density estimate of the target density

$$\widehat{p}(x_t|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} p(y_t|x_t)\delta_{x_t^{(i)}}.$$

We can also produce a sample from the *particle approximation* to $p(x_t|y_{1:t})$

$$\widehat{p}(x_t|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} p(y_t|x_t^{(i)})\delta_{x_t^{(i)}}$$

by performing a *weighted resampling* from $(x_t^{(1)}, \ldots, x_t^{(N)})$ with weights proportional to $p(y_t|x_t^{(i)})$. That is, we consider the discrete distribution

$$\widehat{p}_t(v) = \sum_{i=1}^{n} P_t(v)\delta_{x_t^{(i)}}$$

where

$$P_t(v) = \frac{p(y_t|v)}{\sum_{k=1}^{N} p(y_t|x_t^{(k)})} \qquad v = x_t^{(1)}, \ldots, x_t^{(N)}.$$

This discrete distribution places mass $P_t(x_t^{(i)})$ at $x_t^{(i)}$ for each i, with zero mass elsewhere. We produce an independent sample from this distribution by resampling $x_t^{(i)}$ with probability $P_t(x_t^{(i)})$. Denote the resampled vector

$$u_t^{(1)}, \ldots, u_t^{(N)}$$

- For the **prediction** step, the particle approximation to $p(x_{t+1}|y_{1:t})$ is obtained as

$$\widehat{p}(x_{t+1}|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} p(x_{t+1}|u_t^{(i)})$$

A sample from this approximation can be obtained by propagating the samples $u_t^{(1)}, \ldots, u_t^{(N)}$ forward by sampling the transition densities $p(x_{t+1}|u_t^{(i)})$ for each i. This yields the sample

$$x_{t+1}^{(1)}, \ldots, x_{t+1}^{(N)}$$

and the recursion continues.

Example: AR(1) plus noise
Suppose

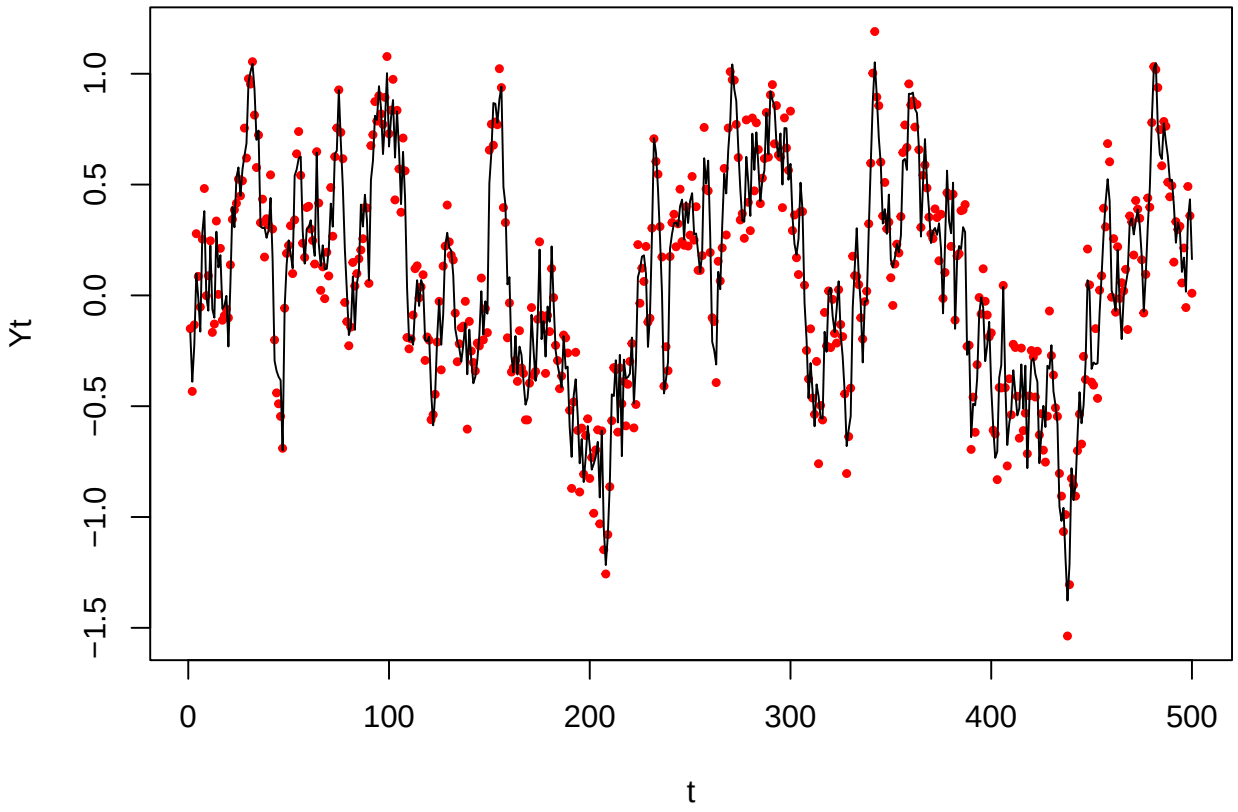
$$\begin{aligned} Y_t &= X_t + V_t \\ X_{t+1} &= \phi X_t + W_t \end{aligned}$$

where $-1 < \phi < 1$, with $V_t \sim Normal(0, \sigma_V^2)$ and $W_t \sim Normal(0, \sigma_V^2)$, so that

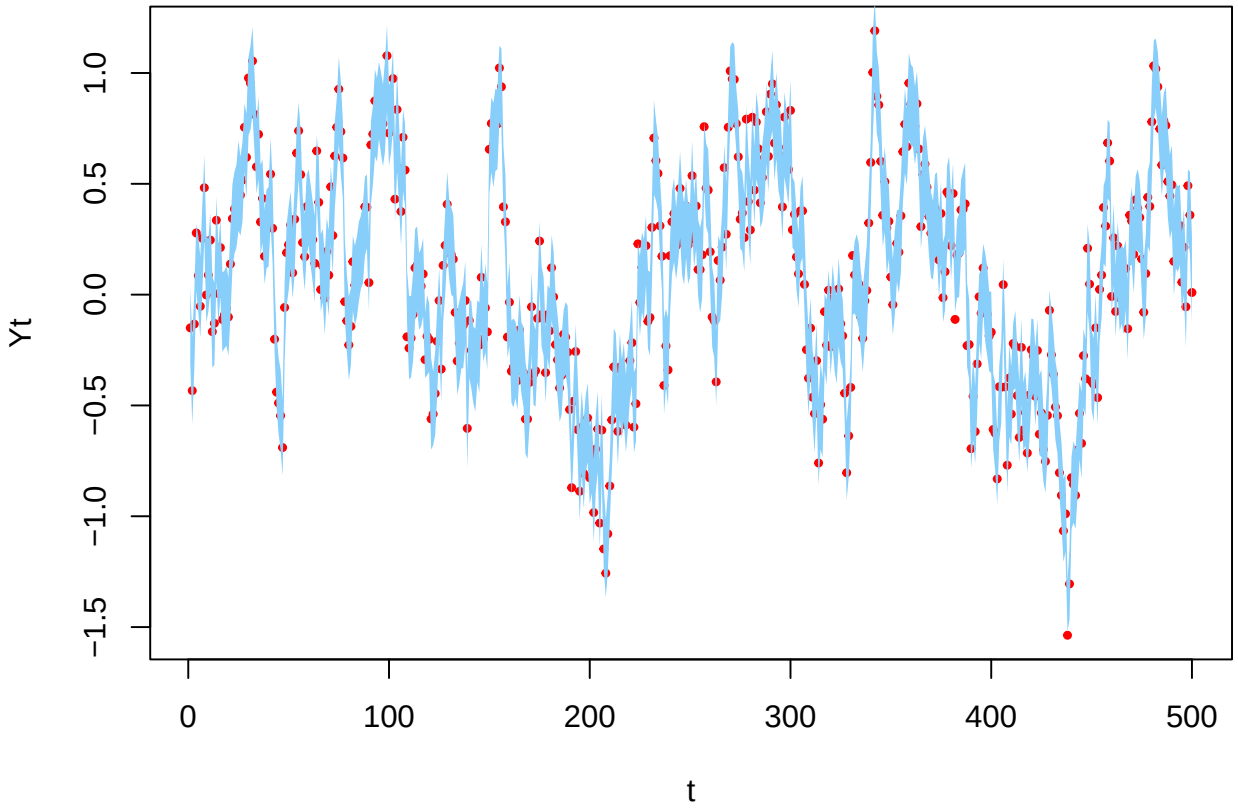
$$\begin{aligned} p(y_t|x_t) &= Normal(x_t, \sigma_V^2) \\ p(x_{t+1}|x_t) &= Normal(\phi x_t, \sigma_W^2) \end{aligned}$$

For illustration, suppose $\phi = 0.9$, $\sigma_V = 0.1$ and $\sigma_W = 0.2$.

```
set.seed(23)
n<-500
sigV<-0.1; sigW<-0.2; phi<-0.9
Vt<-rnorm(n)*sigV
Wt<-rnorm(n)*sigW
Xt<-rep(0,n)
Xt[1]<-rnorm(1,0,sigV/sqrt(1-phi^2))
for(s in 1:(n-1)){Xt[s+1]<-phi*Xt[s]+Wt[s]}
Yt<-Xt+Vt
par(mar=c(4,4,1,2))
plot(Yt,pch=19,col='red',cex=0.5,xlab='t')
lines(Xt)
```




```
N<-200
Xmat<-Umat<-wmat<-Wmat<-matrix(0,nrow=n,ncol=N)
Xmat[1,]<-rnorm(N,0,sigW/sqrt(1-phi^2))
wmat[1,]<-dnorm(Yt[1],Xmat[1,],sigV)
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
Umat[1,]<-sample(Xmat[1,],size=N,prob=Wmat[1,],rep=T)
for(j in 2:n){
        Xmat[j,]<-rnorm(N)*sigW+phi*Umat[j-1,]
        wmat[j,]<-dnorm(Yt[j],Xmat[j,],sigV)
        Wmat[j,]<-wmat[j,]/sum(wmat[j,])
    Umat[j,]<-sample(Xmat[j,],size=N,prob=Wmat[j,],rep=T)
}
Vhat<-apply(Umat,1,mean); V.se<-apply(Umat,1,sd)
par(mar=c(4,4,1,2))
plot(Yt,pch=19,col='red',cex=0.5,xlab='t')
polygon(c(1:500,500:1),c(Vhat-2*V.se,rev(Vhat+2*V.se)),col='lightskyblue',border=NA)
```



The blue shaded region is a 95% confidence interval for the unobserved state.

Sequential Importance Sampling: We aim to perform sequential importance sampling (SIS) for the complete data filtering distribution $p(x_{1:n}|y_{1:n})$. Note that for $j = 1, 2, \ldots, n$,

$$\begin{aligned} p(x_{1:j}, y_{1:j}) &= p(x_{1:(j-1)}, y_{1:(j-1)})p(x_j, y_j|x_{1:(j-1)}, y_{1:(j-1)}) \\ &= p(x_{1:(j-1)}, y_{1:(j-1)})p(x_j|x_{1:(j-1)}, y_{1:(j-1)})p(y_j|x_j, x_{1:(j-1)}, y_{1:(j-1)}) \\ &= p(x_{1:(j-1)}, y_{1:(j-1)})p(x_j|x_{j-1})p(y_j|x_j) \end{aligned}$$

by conditional independence. Therefore

$$\begin{aligned} p(x_{1:j}|y_{1:j}) &= p(x_{1:(j-1)}|y_{1:(j-1)})\frac{p(y_{1:(j-1)})}{p(y_{1:j})}p(x_j|x_{j-1})p(y_j|x_j) \\ &= p(x_{1:(j-1)}|y_{1:(j-1)})\frac{p(y_j|x_j)p(x_j|x_{j-1})}{p(y_j|y_{1:(j-1)})} \\ &= p(x_{1:(j-1)}|y_{1:(j-1)})p(x_j|x_{j-1}, y_j) \\ &= p(x_{1:(j-2)}|y_{1:(j-2)})p(x_{j-1}|x_{j-2}, y_{j-1})p(x_j|x_{j-1}, y_j) \qquad \text{recursion} \\ &\vdots \\ &= p(x_1|y_1)\prod_{k=2}^{j} p(x_k|x_{k-1}, y_k) \end{aligned}$$

Therefore

$$p_n(x_{1:n}) \equiv p(x_{1:n}|y_{1:n}) = p(x_1|y_1)\prod_{j=2}^{n} p(x_j|x_{j-1}, y_j) = p_1(x_1)\prod_{j=2}^{n} \widetilde{p}_j(x_j|x_{1:(j-1)})$$

say, where

$$\widetilde{p}_j(x_j|x_{1:(j-1)}) \equiv p(x_j|x_{j-1}, y_j) = \frac{p(y_j|x_j)p(x_j|x_{j-1})}{p(y_j|y_{1:(j-1)})}.$$

Consider the importance sampling (IS) density

$$f_n(x_{1:n}) = f_1(x_1)\prod_{j=2}^{n} \widetilde{f}_j(x_j|x_{1:(j-1)})$$

with weights given by $w_1(x_1) = p_1(x_1)/f_1(x_1)$ and for $j \geq 2$

$$w_j(x_{1:j}) = \frac{p_1(x_1)}{f_1(x_1)}\prod_{k=2}^{j}\frac{\widetilde{p}_k(x_k|x_{1:(k-1)})}{\widetilde{f}_k(x_k|x_{1:(k-1)})} = w_{j-1}(x_{1:(j-1)})\frac{\widetilde{p}_j(x_j|x_{1:(j-1)})}{\widetilde{f}_j(x_j|x_{1:(j-1)})} = w_{j-1}(x_{1:(j-1)})\frac{p_j(x_{1:j})}{p_{j-1}(x_{1:(j-1)})\widetilde{f}_j(x_j|x_{1:(j-1)})}$$

say, as by definition

$$\widetilde{p}_j(x_j|x_{1:(j-1)}) = \frac{p_j(x_{1:j})}{p_{j-1}(x_{1:(j-1)})} = \frac{p(x_{1:j}|y_{1:j})}{p(x_{1:(j-1)}|y_{1:(j-1)})}.$$

If

$$p_j(x_{1:j}) = p(x_{1:j}|y_{1:j}) = \frac{g_j(x_{1:j})}{Z_j}$$

the normalizing constants Z_j for the p_j for $j = 1, \ldots, n$ are unknown, these weights can be replaced by

$$w_1(x_1) = g_1(x_1)/p_1(x_1)$$

$$j \geq 2 \qquad w_j(x_{1:j}) = w_{j-1}(x_{1:(j-1)})\frac{g_j(x_{1:j})}{g_{j-1}(x_{1:(j-1)})\widetilde{f}_j(x_j|x_{1:(j-1)})} = w_{j-1}(x_{1:(j-1)})\alpha_j(x_{1:j})$$

say. The Monte Carlo computation is carried out as follows:

(i) for $j = 1$: sample $x_1^{(1)}, \ldots, x_1^{(N)}$ from p_1 and compute IS weights $w_1(x_1^{(i)})$.

(ii) for $j \geq 2$: for $i = 1, \ldots, N$, sample $x_j^{(i)}$ from $\widetilde{f}_j(x_j|x_{1:(j-1)}^{(i)})$ and compute IS weight

$$w_j(x_{1:j}^{(i)}) = w_{j-1}(x_{1:(j-1)}^{(i)})\alpha_j(x_{1:j}^{(i)})$$

We then have ultimately

$$\widehat{p}_n(x_{1:n}) = \sum_{i=1}^{N} W_n^{(i)}\delta_{x_{1:n}^{(i)}}(x_{1:n})$$

where

$$W_n^{(i)} = \frac{w_n(x_{1:n}^{(i)})}{N\widehat{Z}_n} \qquad\qquad \widehat{Z}_n = \frac{1}{N}\sum_{i=1}^{N} w_n(x_{1:n}^{(i)})$$

in the usual way. In order to assess the effectiveness of the procedure we track the *effective sample size* (ESS) where

$$\text{ESS} = \left(\sum_{i=1}^{N}\left\{W_j^{(i)}\right\}^2\right)^{-1}$$

We have $1 \leq \text{ESS} \leq N$, and large values of the ESS statistic indicate more effective performance.

In a simple state space model, write

$$\begin{aligned} \text{Observation density} &: p_{Y|X}(y_j|x_j) \\ \text{State density} &: p_X(x_j|x_{j-1}) \end{aligned}$$

with $X_1 \sim p_{X_1}$. In light of the observed data $y_{1:n} = (y_1, \ldots, y_n)$, the target density is

$$p_n(x_{1:n}) = p(x_{1:n}|y_{1:n}) = \frac{p(x_{1:n}, y_{1:n})}{p(y_{1:n})} = \frac{g_n(x_{1:n}, y_{1:n})}{Z_n(y_{1:n})}$$

say, where

$$g_n(x_{1:n}, y_{1:n}) = p_{X_1}(x_1) \left\{ \prod_{j=2}^{n} p_X(x_j|x_{j-1}) \right\} \left\{ \prod_{j=1}^{n} p_{Y|X}(y_j|x_j) \right\}$$

and

$$Z_n(y_{1:n}) = \int g_n(x_{1:n}, y_{1:n})\, dx_{1:n}.$$

The optimal choice of IS density $\widetilde{f}_j(x_j|x_{1:(j-1)})$ is $\widetilde{p}_j(x_j|x_{1:(j-1)})$, given by

$$\begin{aligned} \widetilde{p}_j(x_j|x_{1:(j-1)}) \equiv \widetilde{p}_j(x_j|y_{1:j}, x_{1:(j-1)}) = \frac{p_{Y|X}(y_j|x_{1:j}) p_X(x_j|x_{1:(j-1)})}{p(y_j|x_{1:(j-1)})} &= \frac{p_{Y|X}(y_j|x_j) p_X(x_j|x_{j-1})}{p(y_j|x_{j-1})} \\ &= \frac{p_{Y|X}(y_j|x_j) p_X(x_j|x_{j-1})}{\int p_{Y|X}(y_j|x) p_X(x|x_{j-1})\, dx} \end{aligned}$$

as, conditional on $x_{1:(j-1)}$, x_j is independent of $y_{1:(j-1)}$. If the optimal choice is not feasible – we need successive samples from the IS densities

$$\widetilde{f}_j(x_j|x_{1:(j-1)}) = \widetilde{p}_j(x_j|y_{1:j}, x_{1:(j-1)}) = \widetilde{p}_j(x_j|y_j, x_{(j-1)})$$

which might not be possible – we might choose instead the transition density as the proposal

$$\widetilde{f}_j(x_j|x_{1:(j-1)}) = p_X(x_j|x_{j-1})$$

which typically is straightforward to sample from, and yields the recursive (unnormalized) weight calculation

$$w_j(x_{1:j}) = w_{j-1}(x_{1:(j-1)}) p_{Y|X}(y_j|x_j)$$

Example: AR(1) plus noise

$$\begin{aligned} Y_t &= X_t + V_t \\ X_{t+1} &= \phi X_t + W_t \end{aligned}$$

where $-1 < \phi < 1$, with $V_t \sim Normal(0, \sigma_V^2)$ and $W_t \sim Normal(0, \sigma_V^2)$, so that

$$\begin{aligned} p(y_t|x_t) &= Normal(x_t, \sigma_V^2) \\ p(x_{t+1}|x_t) &= Normal(\phi x_t, \sigma_W^2) \end{aligned}$$

In this case, we have that

$$\widetilde{p}_j(x_j|y_j, x_{(j-1)}) \propto \exp\left\{ -\frac{1}{2\sigma_V^2}(y_j - x_j)^2 \right\} \exp\left\{ -\frac{1}{2\sigma_W^2}(x_j - \phi x_{j-1})^2 \right\}$$

and using the completing-the-square formula

$$A(x-a)^2 + B(x-b)^2 = (A+B)\left(x - \frac{Aa + Bb}{A+B} \right)^2 + \frac{AB}{A+B}(a-b)^2$$

with $A = 1/\sigma_V^2, a = y_j, B = 1/\sigma_W^2$, and $b = \phi x_{j-1}$, we have that

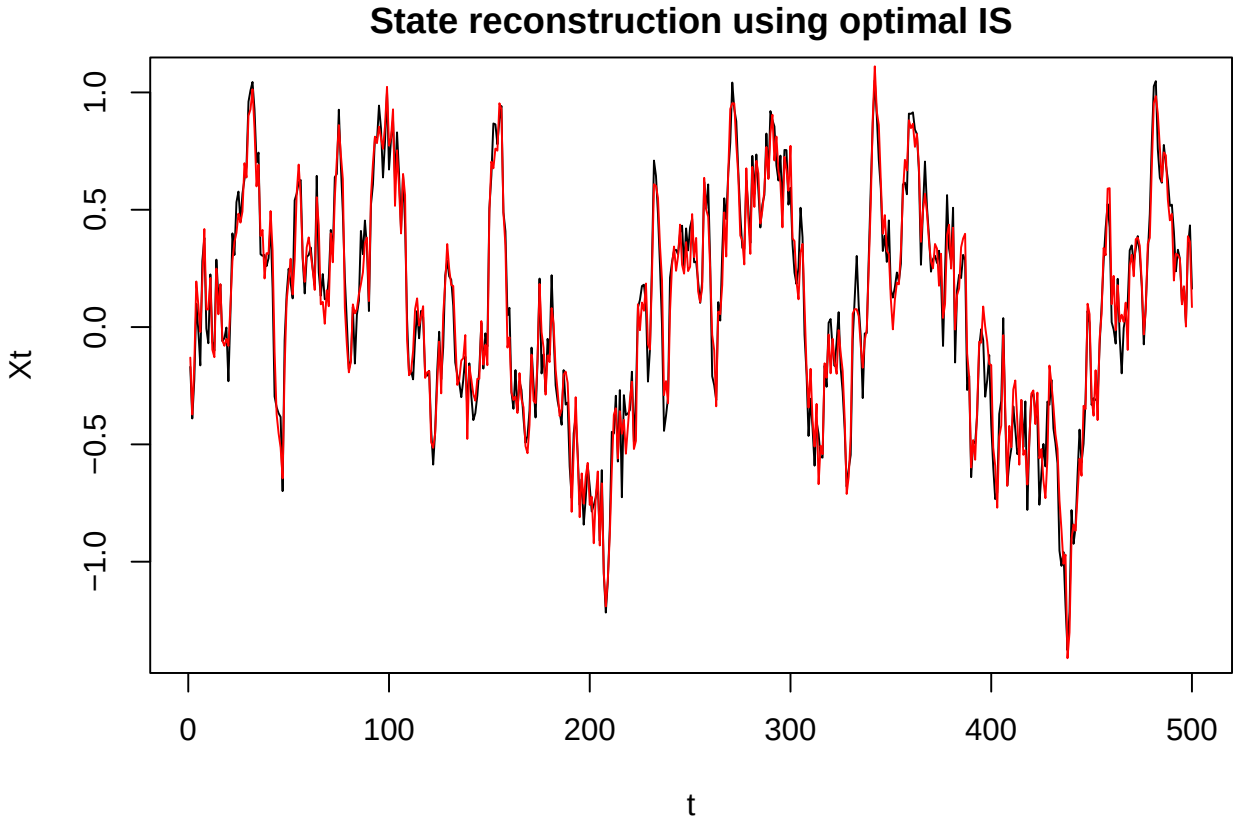
$$\widetilde{p}_j(x_j|y_j, x_{(j-1)}) \propto \exp\left\{-\frac{1}{2}\left[\frac{1}{\sigma_V^2} + \frac{1}{\sigma_W^2}\right]\left(x_j - \frac{y_j/\sigma_V^2 + \phi x_{j-1}/\sigma_W^2}{1/\sigma_V^2 + 1/\sigma_W^2}\right)^2\right\}$$

which means that

$$\widetilde{p}_j(x_j|y_j, x_{(j-1)}) \equiv Normal\left(\frac{y_j/\sigma_V^2 + \phi x_{j-1}/\sigma_W^2}{1/\sigma_V^2 + 1/\sigma_W^2}, \left[\frac{1}{\sigma_V^2} + \frac{1}{\sigma_W^2}\right]^{-1}\right)$$

Therefore, we can sample forward directly from $\widetilde{p}_j(x_j|y_j, x_{(j-1)})$, and the IS weights are always equal to $1/N$. This calculation follows by the exact same arguments that justify the Kalman Filter recursions.

```
N<-200
Xmat<-wmat<-Wmat<-matrix(0,nrow=n,ncol=N)
Xmat[1,]<-rnorm(N,Yt[1],sigW/sqrt(1-phi^2))
wmat[1,]<-rep(1,N)
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
ESS.0<-rep(0,n)
ESS.0[1]<-1/sum(Wmat[1,]^2)
sigj<-sqrt(1/(1/sigV^2+1/sigW^2))
for(j in 2:n){
    muj<-(Yt[j]/sigV^2+phi*Xmat[j-1,]/sigW^2)/(1/sigV^2+1/sigW^2)
        Xmat[j,]<-rnorm(N,muj,sigj)
    wmat[j,]<-rep(1,N); Wmat[j,]<-wmat[j,]/sum(wmat[j,])
    ESS.0[j]<-1/sum(Wmat[j,]^2)
}
Xhat<-rep(0,n)
for(j in 1:n){Xhat[j]<-sum(Wmat[j,]*Xmat[j,])}
par(mar=c(4,4,2,2))
plot(Xt,type='l',xlab='t',main='State reconstruction using optimal IS')
lines(Xhat,col='red')
```



Now we use the alternative proposal

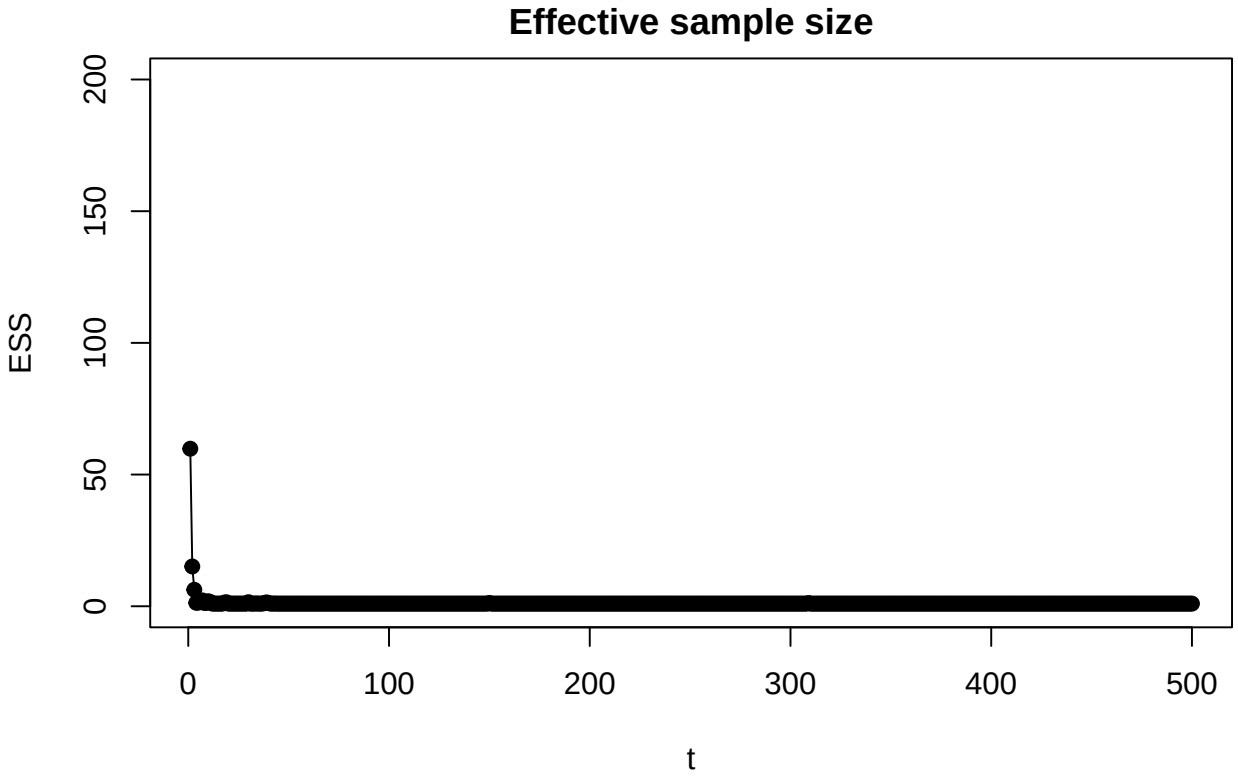
$$\widetilde{f}_j(x_j|x_{1:(j-1)}) = p_X(x_j|x_{j-1}) \equiv Normal(\phi x_{j-1}, \sigma_W^2)$$

which yields the recursive (unnormalized) weight calculation

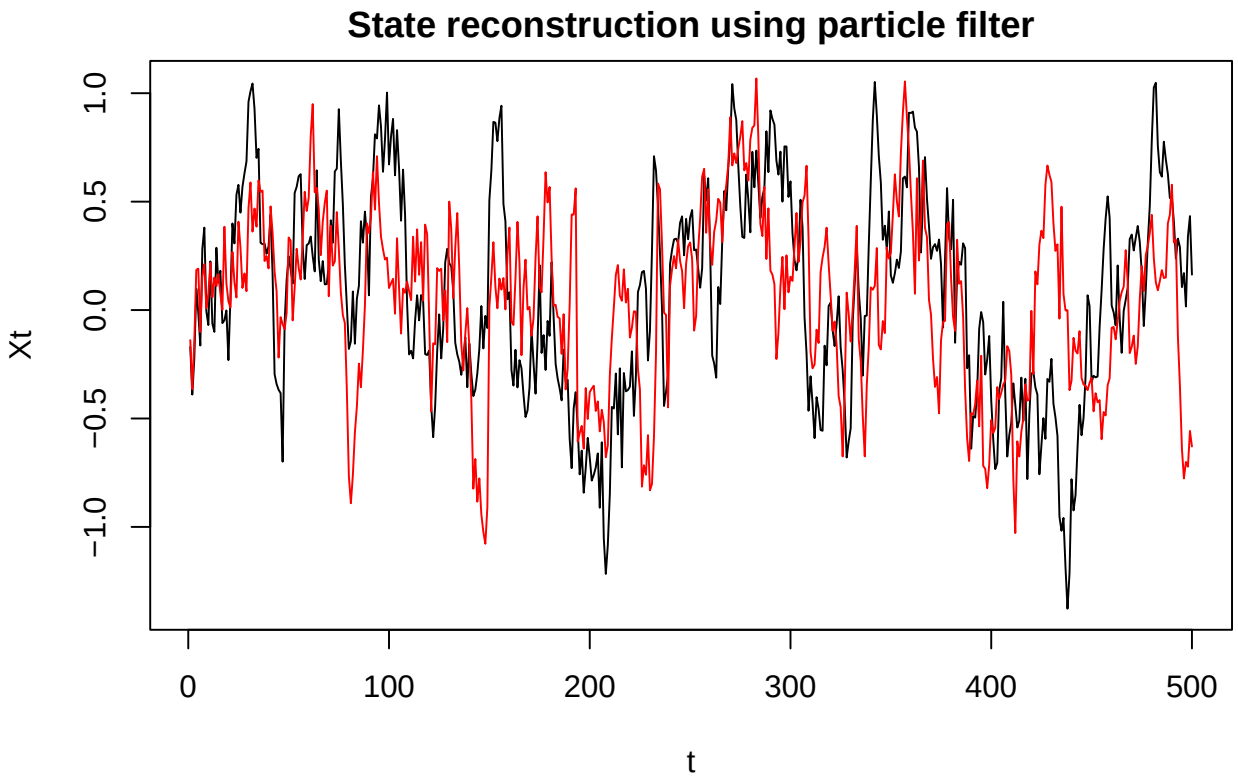
$$w_j(x_{1:j}) = w_{j-1}(x_{1:(j-1)}) \exp\left\{-\frac{1}{2\sigma_V^2}(y_j - x_j)^2\right\}.$$

In this case, there is numerical underflow in this weight calculation, so for each j the weights are scaled.

```
N<-200
Xmat<-wmat<-Wmat<-matrix(0,nrow=n,ncol=N)
Xmat[1,]<-rnorm(N,0,sigW/sqrt(1-phi^2))
wmat[1,]<-dnorm(Yt[1],Xmat[1,],sigV)
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
ESS.1<-rep(0,n)
ESS.1[1]<-1/sum(Wmat[1,]^2)
for(j in 2:n){
    Xmat[j,]<-rnorm(N)*sigW+phi*Xmat[j-1,]
    wv<-log(wmat[j-1,])+dnorm(Yt[j],Xmat[j,],sigV,log=T)
    wmat[j,]<-exp(wv-max(wv))    #Scaling
        Wmat[j,]<-wmat[j,]/sum(wmat[j,])
    ESS.1[j]<-1/sum(Wmat[j,]^2)
}
par(mar=c(4,4,2,2))
plot(c(1:n),ESS.1,type="l",xlab='t',ylab="ESS",ylim=range(0,N))
points(c(1:n),ESS.1,pch=19)
title('Effective sample size')
```




```
Xhat<-rep(0,n)
for(j in 1:n){Xhat[j]<-sum(Wmat[j,]*Xmat[j,])}
plot(Xt,type='l',xlab='t',main='State reconstruction using particle filter')
lines(Xhat,col='red')
```



Particle Degeneracy: A problem with this strategy is that the weights start to **degenerate** as j increases;

- the weights that define the Monte Carlo estimate $\widehat{p}_n(x_{1:n})$ are very small for most i, with only a few large weights.
- the empirical variance of the collection of unnormalized weights $\{w_j(x_j^{(i)})\}$ for $i = 1, \ldots, N$ increases
- this is mitigated, but not resolved by a better choice of $\widetilde{f}_j(x_j|x_{1:(j-1)})$

Particle degeneracy can be overcome by *resampling* that removes the particles with low weights. The commonest form of resampling is to use a non-parametric bootstrap, with multinomial sampling. At step j, the Monte Carlo estimate of p_j is

$$\widehat{\pi}_j(x_{1:n}) = \sum_{i=1}^{N} W_j(x_{1:j}^{(i)}) \delta_{x_{1:j}^{(i)}}(x_{1:n})$$

This discrete distribution on

$$x_{1:j}^{(1)}, \ldots, x_{1:j}^{(N)}$$

is resampled N times according to the weights

$$W_j(x_{1:j}^{(1)}), \ldots, W_j(x_{1:j}^{(N)}).$$

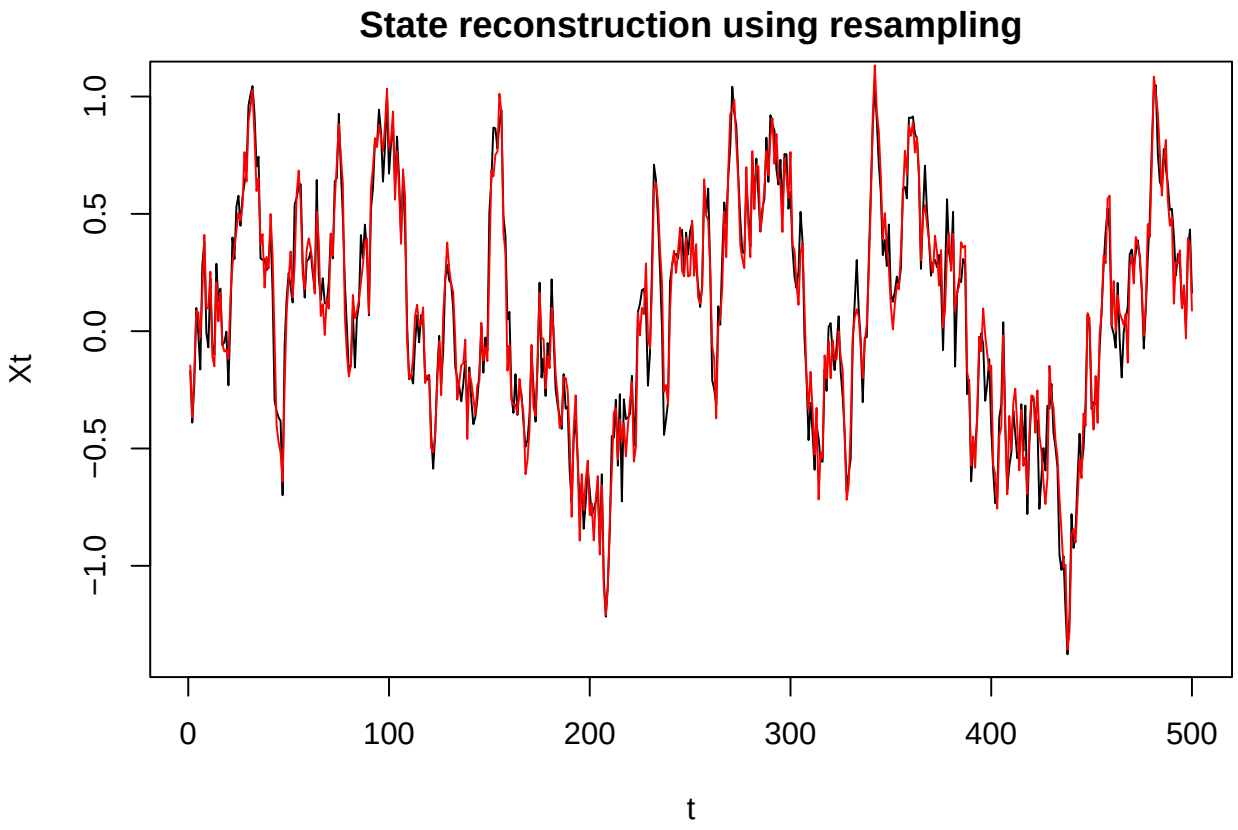
After resampling, the new resampled values replace the

$$x_{1:j}^{(1)}, \ldots, x_{1:j}^{(N)}$$

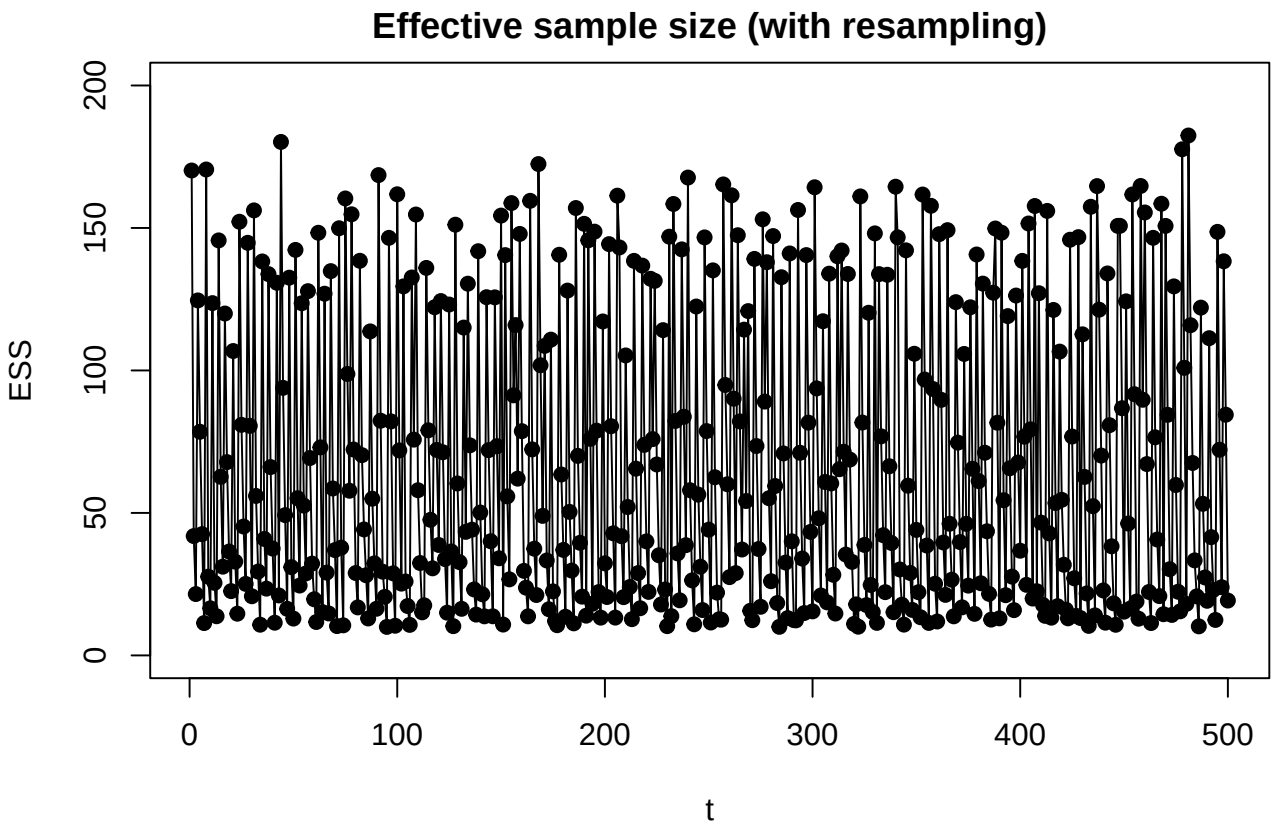
collection, and each resampled point carries with it its normalized resampling weight $W_j(x_{1:j}^{(i)})$.

Each original value $x_{1:j}^{(i)}$ may appear in the resampled collection more than once. We perform resampling to preserve particle diversity, but should not resample too often as that introduces excess variability. A common strategy is to resample only when the observed variance of the weights exceeds some threshold, or the ESS is too low.

```r
N<-200; index<-c(1:N)
Xmat<-wmat<-Wmat<-Imat<-matrix(0,nrow=n,ncol=N)
Xmat[1,]<-rnorm(N,0,sigW/sqrt(1-phi^2))
wmat[1,]<-dnorm(Yt[1],Xmat[1,],sigV)
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
Imat[1,]<-sample(index,prob=Wmat[1,],size=N,rep=T)
Xmat[1,]<-Xmat[1,Imat[1,]]
wmat[1,]<-Wmat[1,Imat[1,]]
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
ESS.3<-rep(0,n)
ESS.3[1]<-1/sum(Wmat[1,]^2)
for(j in 2:n){
        Xmat[j,]<-rnorm(N)*sigW+phi*Xmat[j-1,]
    wv<-log(wmat[j-1,])+dnorm(Yt[j],Xmat[j,],sigV,log=T)
    wmat[j,]<-exp(wv-max(wv))
        Wmat[j,]<-wmat[j,]/sum(wmat[j,])
        ESS.3[j]<-1/sum(Wmat[j,]^2)
        if(ESS.3[j]<10){  #Resample
                Imat[j,]<-sample(index,prob=Wmat[j,],size=N,rep=T)
                Xmat[j,]<-Xmat[j,Imat[j,]]
                wmat[j,]<-Wmat[j,Imat[j,]]
                Wmat[j,]<-wmat[j,]/sum(wmat[j,])
                ESS.3[j]<-1/sum(Wmat[j,]^2)
        }
}
Xhat<-rep(0,n)  #Filtering
for(j in 1:n){Xhat[j]<-sum(Wmat[j,]*Xmat[j,])}
par(mar=c(4,4,2,2))
plot(Xt,type='l',xlab='t', main='State reconstruction using resampling')
lines(Xhat,col='red')
```



```
par(mar=c(4,4,2,1))
plot(c(1:n),ESS.3[1:n],type="l",xlab='t',ylab="ESS",ylim=range(0,N))
points(c(1:n),ESS.3[1:n],pch=19)
title('Effective sample size (with resampling)')
```



Stochastic Volatility Model: We aim to carry out particle filtering for a

$$\text{Observation equation} \quad : \quad p_{Y|X}(y_j|x_j) \equiv Normal(0, \exp\{2x_j\})$$

$$\text{State equation} \quad : \quad p_X(x_j|x_{j-1}) \equiv Normal(\mu + \phi(x_{j-1} - \mu), \sigma^2)$$

with $p_{X_1}(x) \equiv Normal(\mu, \sigma^2/(1-\phi^2))$. We regard μ, ϕ and σ as known constants. In the simulation below, $\mu = -2, \phi = 0.95$ an $\sigma = 0.3$.

This model is used in the analysis of financial time series: it is a non-linear state space model, due to the non-linear dependence of the distribution of y_j on x_j. Here, for $j \geq 2$,

$$p_{Y|X}(y_j|x_j) = \left(\frac{e^{-2x_j}}{2\pi}\right)^{1/2} \exp\{-e^{-2x_j} y_j^2/2\}$$

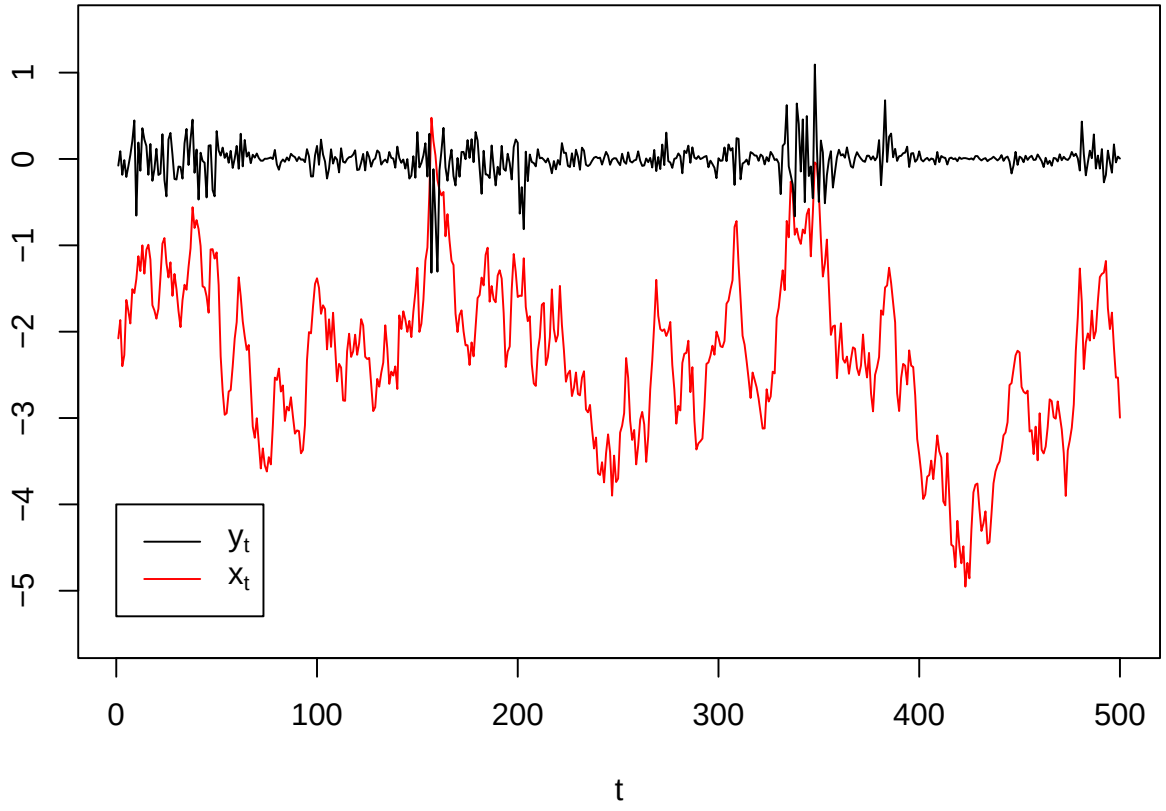
and the recursive weight calculation is based on

$$w_j(x_{1:j}) = \prod_{k=1}^{j} \frac{\widetilde{p}_k(x_k|x_{1:(k-1)})}{\widetilde{f}_k(x_k|x_{1:(k-1)})} = \prod_{k=1}^{j} p_{Y|X}(y_k|x_k)$$

that is, the likelihood up to observation j.

```
set.seed(1231)
n<-500
mu<--2; phi<-0.95; sig<-0.3
```

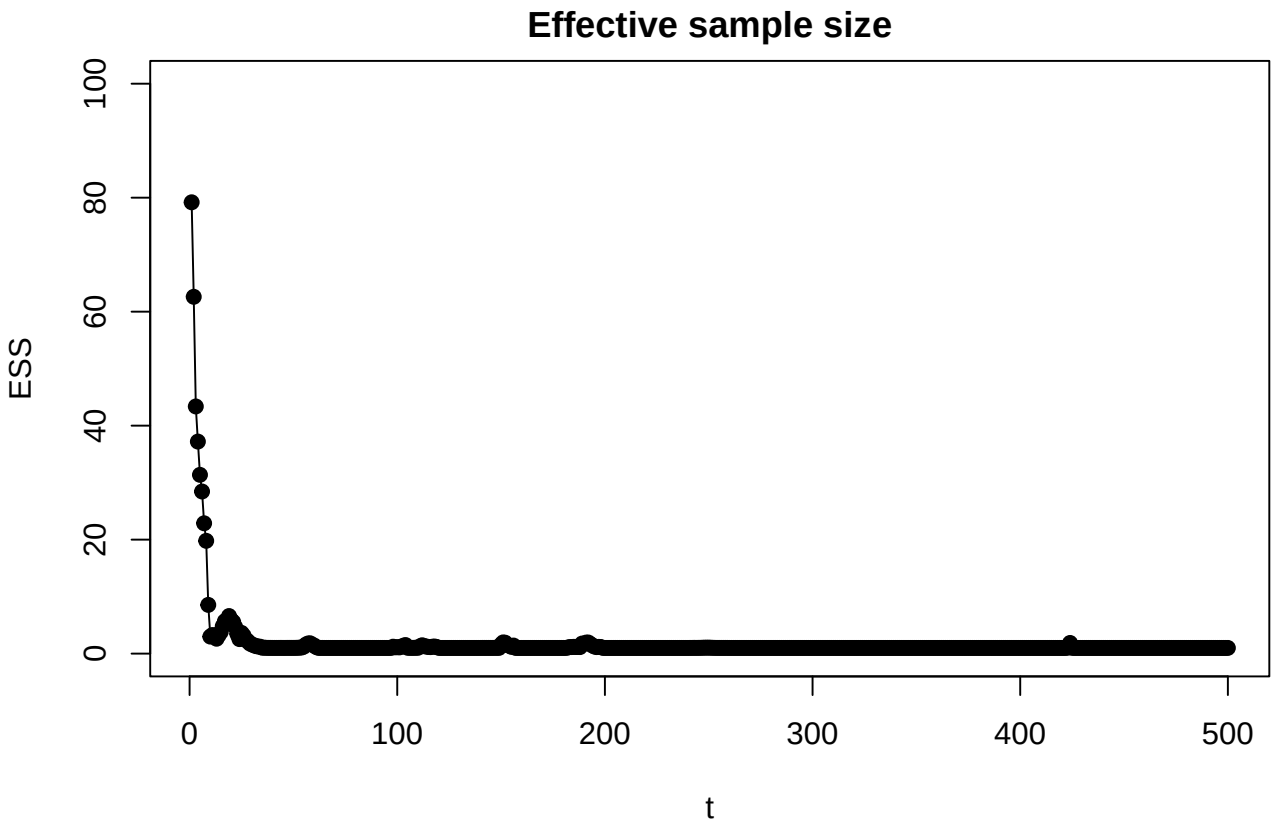
```
xvec<-yvec<-rep(0,n)
xvec[1]<-rnorm(1,mu,sig/sqrt(1-phi^2))
for(j in 2:n){xvec[j]<-rnorm(1,mu+phi*(xvec[j-1]-mu),sig)}
yvec<-rnorm(n)*exp(xvec)
par(mar=c(4,4,1,2))
plot(1:n,xvec,type="l",xlab="t",col='red',ylab=' ',ylim=range(-5.5,1.5))
lines(1:n,yvec,pch=19,cex=0.5)
legend(0,-4,c(expression(y[t]),expression(x[t])),col=c('black','red'),lty=1)
```




```
N<-100
Xmat<-wmat<-Wmat<-matrix(0,nrow=n,ncol=N)
Xmat[1,]<-rnorm(N,mu,sig/sqrt(1-phi^2))
wmat[1,]<-dnorm(yvec[1],0,exp(Xmat[1,]))
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
ESS.1<-rep(0,n)
ESS.1[1]<-1/sum(Wmat[1,]^2)
for(j in 2:n){
        Xmat[j,]<-rnorm(N)*sig+(mu+phi*(Xmat[j-1,]-mu))
        wmat[j,]<-wmat[j-1,]*dnorm(yvec[j],0,exp(Xmat[j,]))
        Wmat[j,]<-wmat[j,]/sum(wmat[j,])
    ESS.1[j]<-1/sum(Wmat[j,]^2)
}
```

Using this specific proposal, we observe quite poor performance.

```
par(mar=c(4,4,2,1))
plot(c(1:n),ESS.1[1:n],type="l",xlab='t',ylab="ESS",ylim=range(0,N))
points(c(1:n),ESS.1[1:n],pch=19)
title('Effective sample size')
```



We may also estimate the unknown state by $\widehat{x}_t$ computed from the sampled points. We compute the estimated (filtered) state by using the kernel density

$$\widehat{p}(x|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\phi(y_t, 0, \exp\{2x\})\mathbb{1}_{z_t^{(i)}}(x).$$

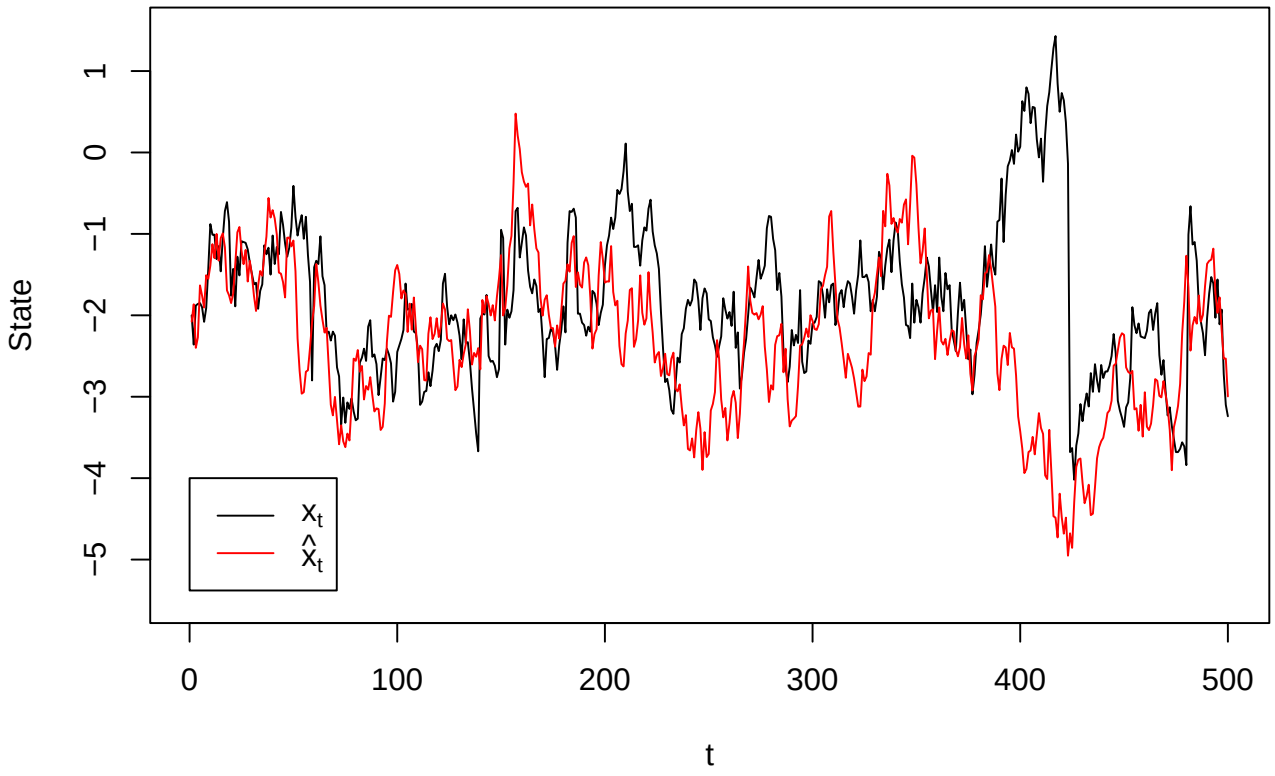
where $\phi(x;\mu,\sigma^2)$ is the Normal pdf with parameters μ and σ^2, or the mean estimate derived from the particles

$$\widehat{x}_t = \frac{1}{N}\sum_{i=1}^{N} W_t^{(i)} z_t^{(i)}.$$

```
xv<-yv<-seq(-5,5,by=0.01)
dnsv<-function(yval,xval){
        return(dnorm(yval,0,exp(xval)))
}
dns0<-function(x1,x0){
        return(dnorm(x1,mu,sig/sqrt(1-phi^2)))
}
dns<-function(x1,x0){
        return(dnorm(x1,mu+phi*(x0-mu),sig))
}

Xhat<-matrix(0,nrow=length(yv),ncol=n)  #Filtering
Yhat<-matrix(0,nrow=length(yv),ncol=n)  #Prediction for y
for(j in 1:n){
        if(j == 1){
                fmat<-outer(xv,Xmat[1,],dns0)
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }else{
                fmat<-outer(xv,Xmat[j-1,],dns)
```

```
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }
        fmat<-outer(yv,exp(Xmat[j,]),dnsv)
        Yhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
}
par(mar=c(4,4,2,1))
plot(xv[apply(Xhat,2,which.max)],xlab='t',type='l',ylab='State',ylim=range(-5.5,1.5))
lines(xvec,col='red')
legend(0,-4,c(expression(x[t]),expression(hat(x)[t])),col=c('black','red'),lty=1)
```



Particle Degeneracy: To address this, we seek a better choice of $\widetilde{f}_j(x_j|x_{1:(j-1)})$. Specifically, we try a normal approximation of

$$\begin{aligned} p_j(x_j|y_{1:j}, x_{1:(j-1)}) &\propto p_{Y|X}(y_j|x_j)p_X(x_j|x_{j-1}) \\ &= \exp\left\{-x_j - \frac{y_j^2}{2e^{2x_j}} - \frac{1}{2\sigma^2}(x_j - \mu_j)^2\right\} \end{aligned}$$

where $\mu_j = \mu + \phi(x_{j-1} - \mu)$. Here the mode of

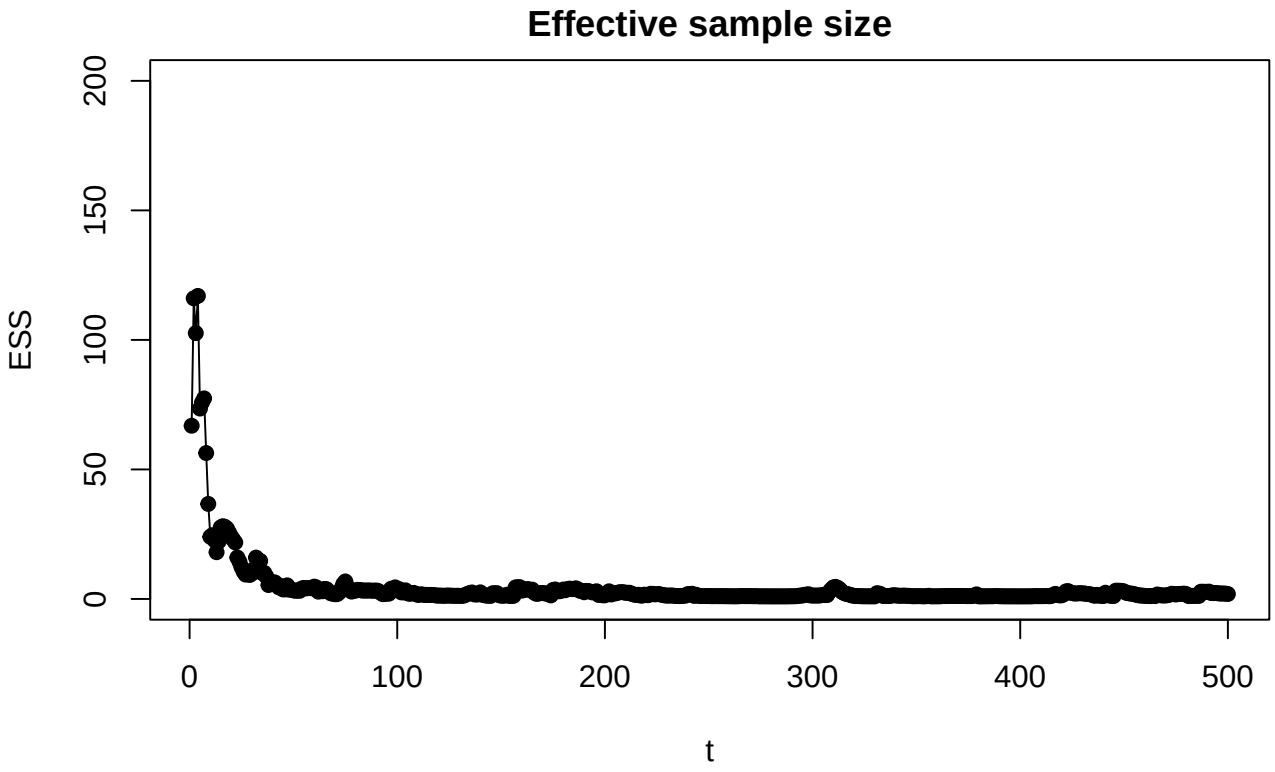
$$g_j(x|y_j, x_{j-1}) = \exp\left\{-x - \frac{y_j^2}{2e^{2x}} - \frac{1}{2\sigma^2}(x - \mu_j)^2\right\}$$

can be computed numerically, and the variance of the normal approximation can be computed by inspecting the curvature at the mode, given by

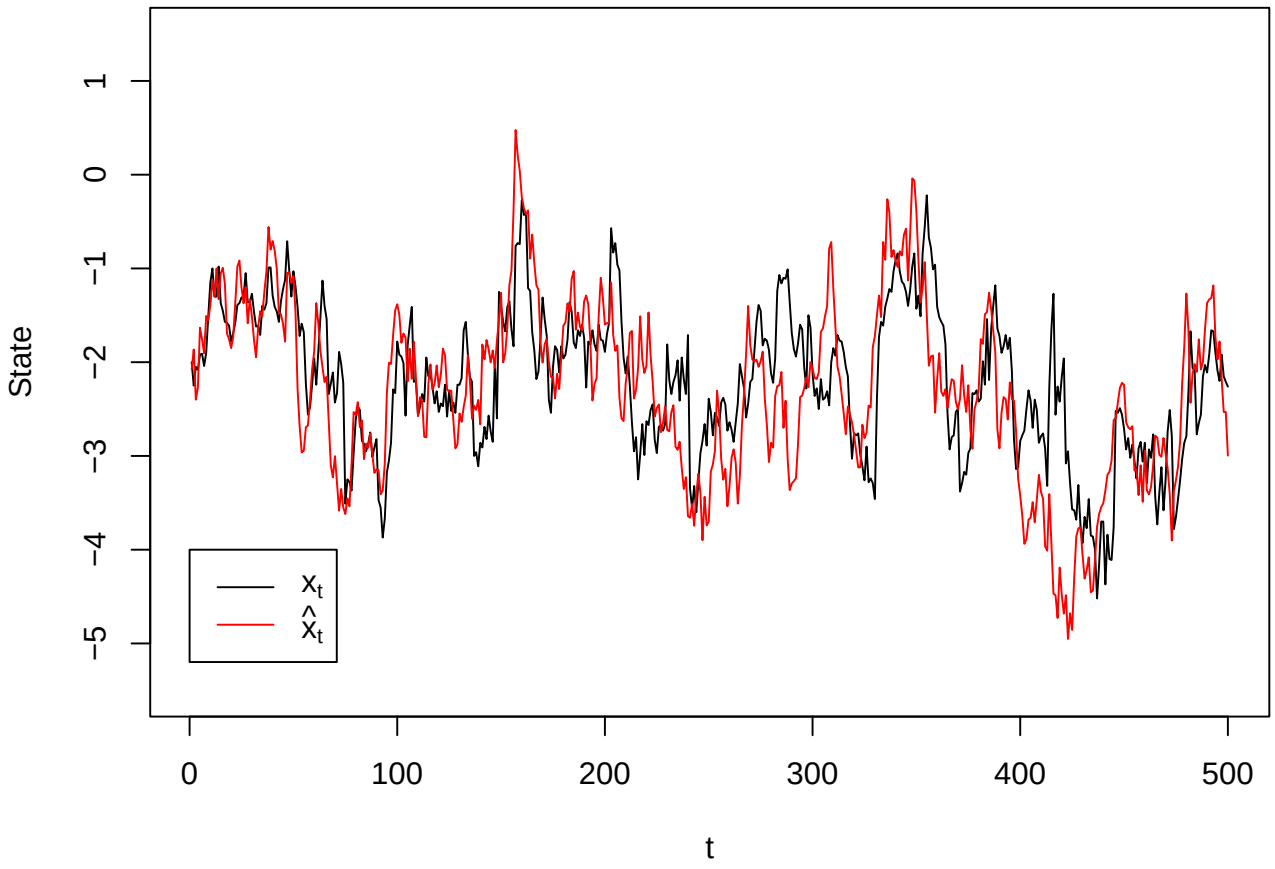
$$\left[-\frac{\partial^2 \log p(x|y_j, x_{j-1})}{\partial x^2}\right]^{-1} = \left[2y_j^2 e^{-2x} + 1/\sigma^2\right]^{-1}$$

```
N<-200
Xmat<-wmat<-Wmat<-matrix(0,nrow=n,ncol=N)
pfunc<-function(x,yv,x0,muv,phiv,sigv){
        muj<-muv+phiv*(x0-muv)
        return(-x-0.5*yv^2*exp(-2*x)-0.5*(x-muj)^2/sigv^2)
}
f1<-optimize(pfunc,yv=yvec[1],x0=mu,muv=mu,phiv=phi,sigv=sig/sqrt(1-phi^2),
             interval=c(-10,10),maximum=TRUE)
mu0<-f1$maximum
sig0<-1/sqrt(2*yvec[1]^2*exp(-2*mu0)+1/sig^2)
Xmat[1,]<-rnorm(N,mu0,sig0)
wmat[1,]<-dnorm(yvec[1],0,exp(Xmat[1,]))*dnorm(Xmat[1,],
                mu,sig/sqrt(1-phi^2))/dnorm(Xmat[1,],mu0,sig0)
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
ESS.2<-rep(0,n); ESS.2[1]<-1/sum(Wmat[1,]^2)
for(j in 2:n){
        for(i in 1:N){
                f1<-optimize(pfunc,yv=yvec[j],x0=Xmat[j-1,i],
            muv=mu,phiv=phi,sigv=sig,interval=c(-10,10),maximum=TRUE)
                mu0<-f1$maximum
                sig0<-1/sqrt(2*yvec[j]^2*exp(-2*mu0)+1/sig^2)
                x<-rnorm(1,mu0,sig0); Xmat[j,i]<-x
                wmat[j,i]<-wmat[j-1,i]*dnorm(yvec[j],0,exp(x))*
                                dnorm(x,mu+phi*(Xmat[j-1,i]-mu),sig)/
                                dnorm(x,mu0,sig0)
                Wmat[j,]<-wmat[j,]/sum(wmat[j,])
        }
        ESS.2[j]<-1/sum(Wmat[j,]^2)
}
```

```
par(mar=c(4,4,2,1))
plot(c(1:n),ESS.2[1:n],type="l",xlab='t',ylab="ESS",ylim=range(0,N))
points(c(1:n),ESS.2[1:n],pch=19); title('Effective sample size')
```



```
Xhat<-matrix(0,nrow=length(yv),ncol=n)
Yhat<-matrix(0,nrow=length(yv),ncol=n)
for(j in 1:n){
        if(j == 1){
                fmat<-outer(xv,Xmat[1,],dns0)
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }else{
                fmat<-outer(xv,Xmat[j-1,],dns)
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }
        fmat<-outer(yv,exp(Xmat[j,]),dnsv)
        Yhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
}
par(mar=c(4,4,2,1))
plot(xv[apply(Xhat,2,which.max)],xlab='t',type='l',ylab='State',ylim=range(-5.5,1.5))
lines(xvec,col='red')
legend(0,-4,c(expression(x[t]),expression(hat(x)[t])),col=c('black','red'),lty=1)
```

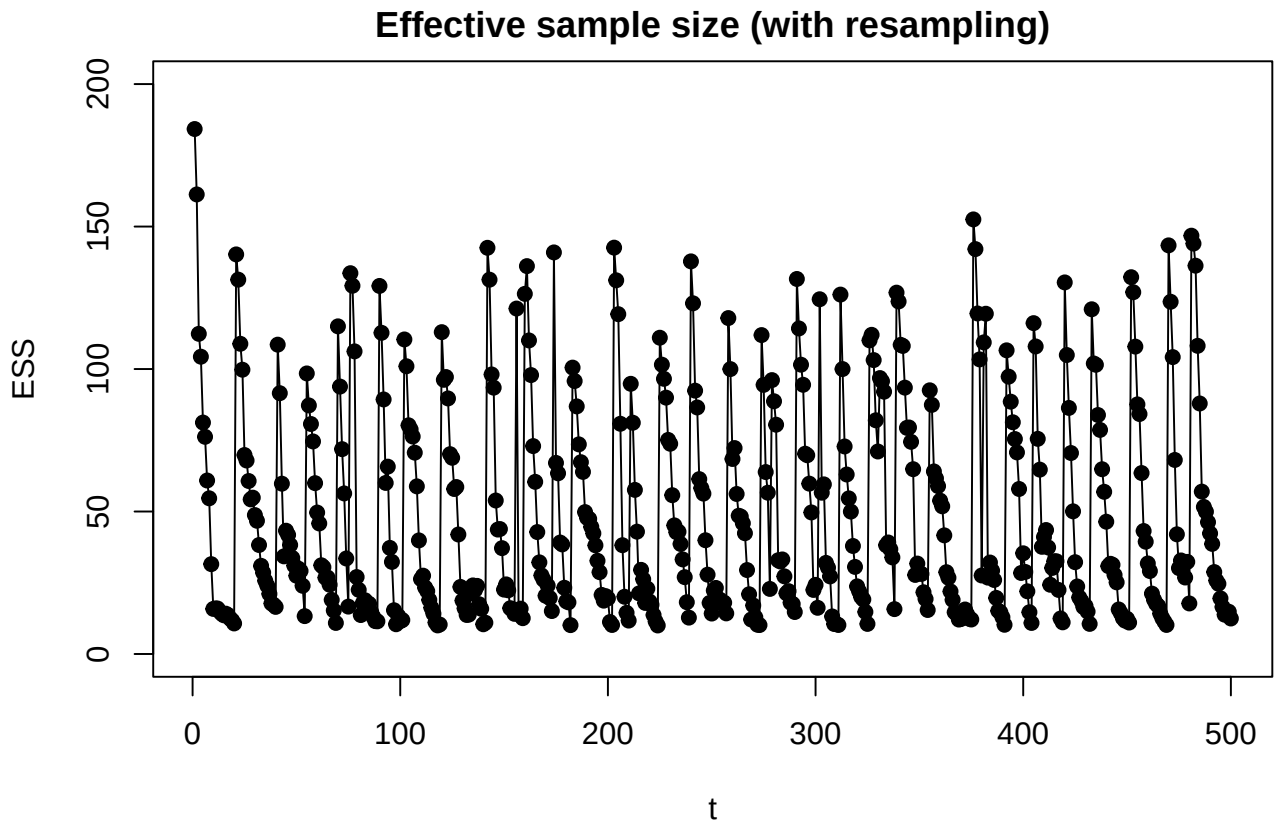


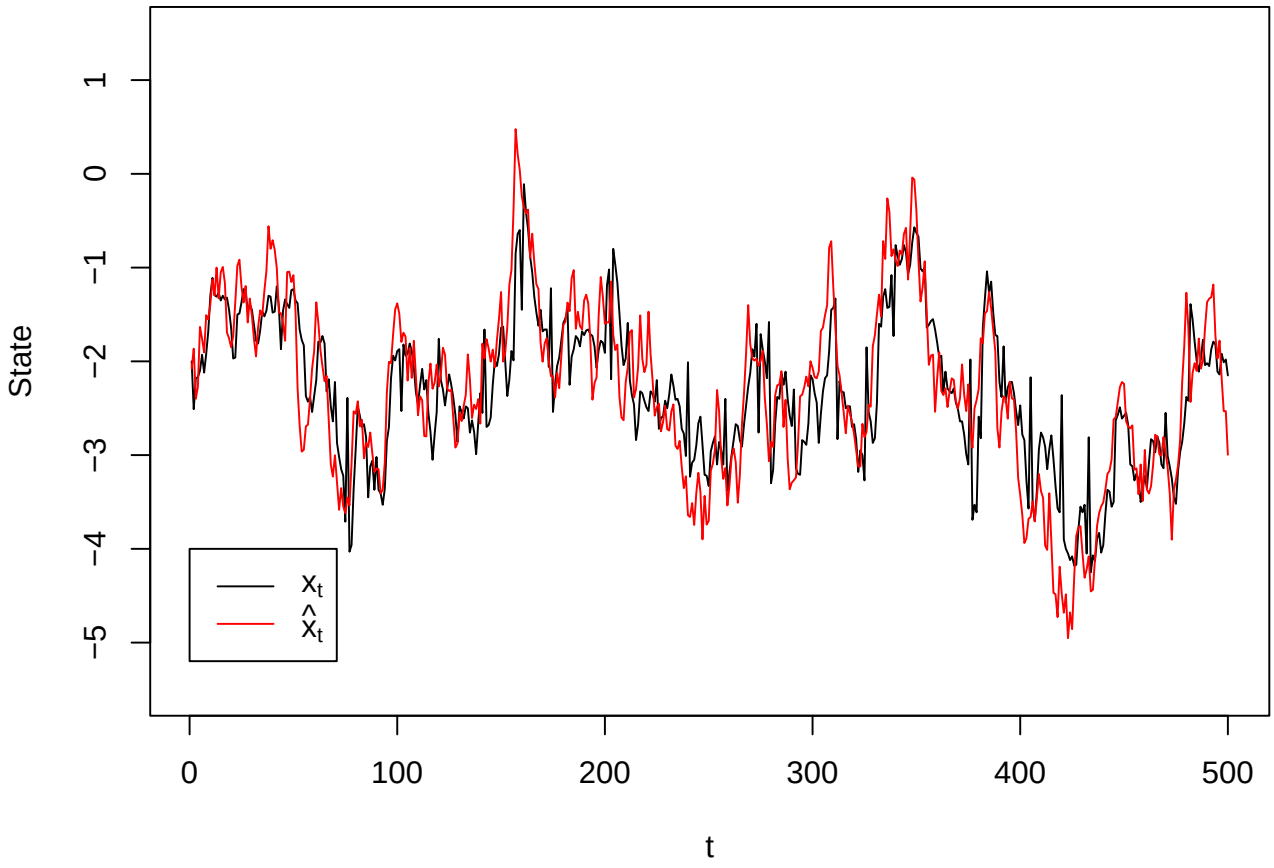
Resampling: We attempt the previous resampling approach.

```
N<-200
index<-c(1:N)
Xmat<-wmat<-Wmat<-Imat<-matrix(0,nrow=n,ncol=N)
Xmat[1,]<-rnorm(N,mu,sig/sqrt(1-phi^2))
wmat[1,]<-dnorm(yvec[1],0,exp(Xmat[1,]))
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
Imat[1,]<-sample(index,prob=Wmat[1,],size=N,rep=T)
Xmat[1,]<-Xmat[1,Imat[1,]]
```

```
wmat[1,]<-Wmat[1,Imat[1,]]
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
ESS.3<-rep(0,n)
ESS.3[1]<-1/sum(Wmat[1,]^2)
for(j in 2:n){
        Xmat[j,]<-rnorm(N)*sig+(mu+phi*(Xmat[j-1,]-mu))
        wmat[j,]<-wmat[j-1,]*dnorm(yvec[j],0,exp(Xmat[j,]))
        Wmat[j,]<-wmat[j,]/sum(wmat[j,])
        ESS.3[j]<-1/sum(Wmat[j,]^2)
        if(ESS.3[j]<10){  #Resample
                Imat[j,]<-sample(index,prob=Wmat[j,],size=N,rep=T)
                Xmat[j,]<-Xmat[j,Imat[j,]]
                wmat[j,]<-Wmat[j,Imat[j,]]
                Wmat[j,]<-wmat[j,]/sum(wmat[j,])
                ESS.3[j]<-1/sum(Wmat[j,]^2)
        }
}
```

```
par(mar=c(4,4,2,1))
plot(c(1:n),ESS.3[1:n],type="l",xlab='t',ylab="ESS",ylim=range(0,N))
points(c(1:n),ESS.3[1:n],pch=19)
title('Effective sample size (with resampling)')
```




```
Xhat<-matrix(0,nrow=length(yv),ncol=n)
Yhat<-matrix(0,nrow=length(yv),ncol=n)
for(j in 1:n){
        if(j == 1){
                fmat<-outer(xv,Xmat[1,],dns0)
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }else{
```

```
                fmat<-outer(xv,Xmat[j-1,],dns)
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }
        fmat<-outer(yv,exp(Xmat[j,]),dnsv)
        Yhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
}
par(mar=c(4,4,2,1))
plot(xv[apply(Xhat,2,which.max)],xlab='t',type='l',ylab='State',ylim=range(-5.5,1.5))
lines(xvec,col='red')
legend(0,-4,c(expression(x[t]),expression(hat(x)[t])),col=c('black','red'),lty=1)
```