R Notebook

Implementation

In this section we present some practical illustrations of the algorithm discussed. For the sake of simplicity we use a random walk plus noise model, i.e. the most basic form of a linear Gaussian state-space model.

$$y_t|x_t \sim N(x_t, \sigma^2) \tag{1}$$
$$x_t|x_{t-1} \sim N(x_{t-1}, \tau^2) \tag{2}$$
$$x_0 \sim N(m_0, C_0) \tag{3}$$

As we already know, in this case the filtering distribution can be computed in closed form solutions using the Kalman filter. However, this toy example will be the basis for the implementations of other filtering strategies since we think that it is useful to understand the logic of the algorithms and to compare their performances. The typical observed process for this kind of model is the one presented in Figure XX. We simulated 500 observation imposing $\sigma^2 = \tau^2 = 1$.

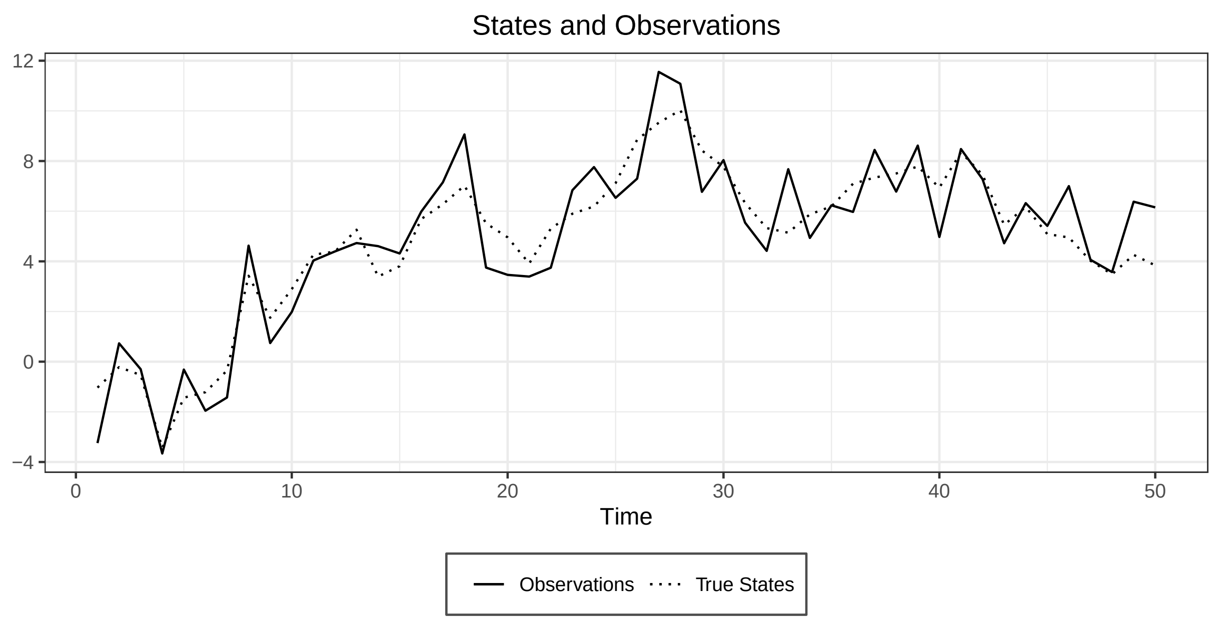


Figure 1: Toy example: the true states and the observation sequence

The Kalman Filter for this model can be easily implemented. Starting from the filtering distribution at period $t-1$, $x_{t-1}|y_{1:t-1} \sim N(m_{t-1}, C_{t-1})$, we compute:

- the one-step-ahead predictive distribution at time $t-1$

$$x_t|y_{1:t-1} \sim N(a_t, R_t)$$
$$a_t = m_{t-1}$$
$$R_t = C_{t-1} + \tau^2$$

- the filtering distribution at time t as $p(x_t|y_{1:t}) \propto p(x_t|y_{1:t-1})p(y_t|x_t)$

$$x_t|y_{1:t} \sim N(m_t, C_t)$$
$$m_t = (1 - \frac{R_t}{R_t + \sigma^2})a_t + \frac{R_t}{R_t + \sigma^2}y_t$$
$$C_t = \frac{R_t}{R_t + \sigma^2}\sigma^2$$

```
DLM<-function(data,sig2,tau2,m0,C0){
  n  = length(data)
  m  = rep(0,n)
  C  = rep(0,n)
  for (t in 1:n){
    if (t==1){
      a = m0
      R = C0 + tau2
    }else{
      a = m[t-1]
      R = C[t-1] + tau2
    }
    A = R/(R+sig2)
    m[t] = (1-A)*a + A*y[t]
    C[t] = A*sig2
  }
  return(list(m=m,C=C))
}
```

In the Figure below we show the filtered states states estimated using Kalman Filter with $x_0 \sim N(0, 100)$. The filtered states follow the observations closely and they provide a good approximation of the true states.

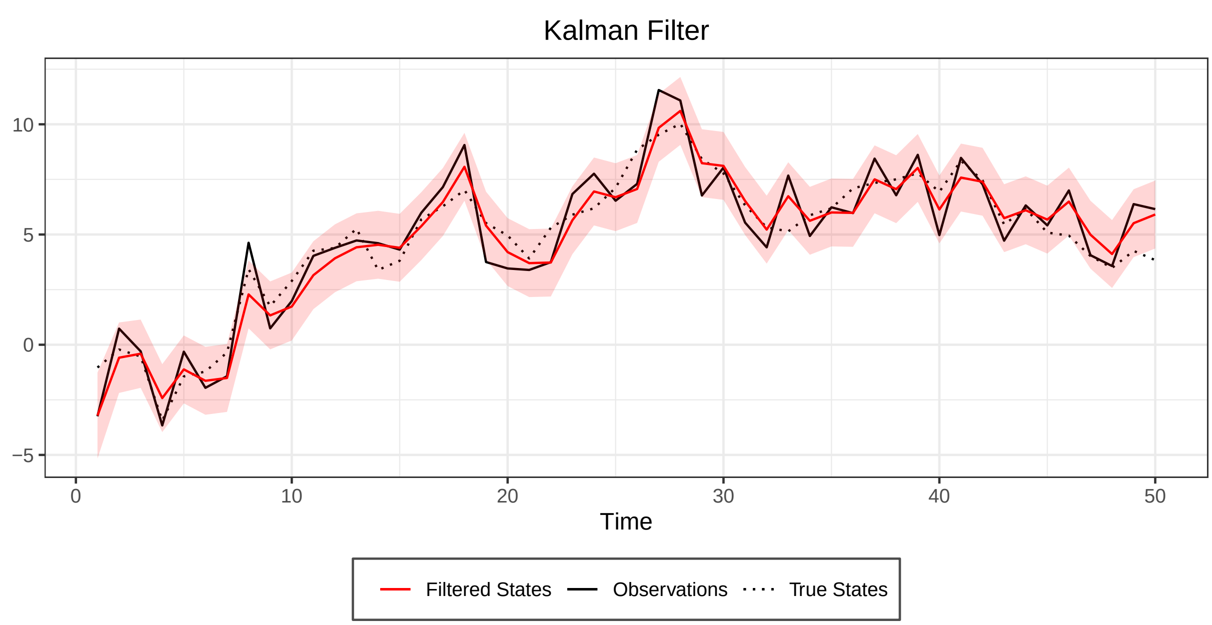
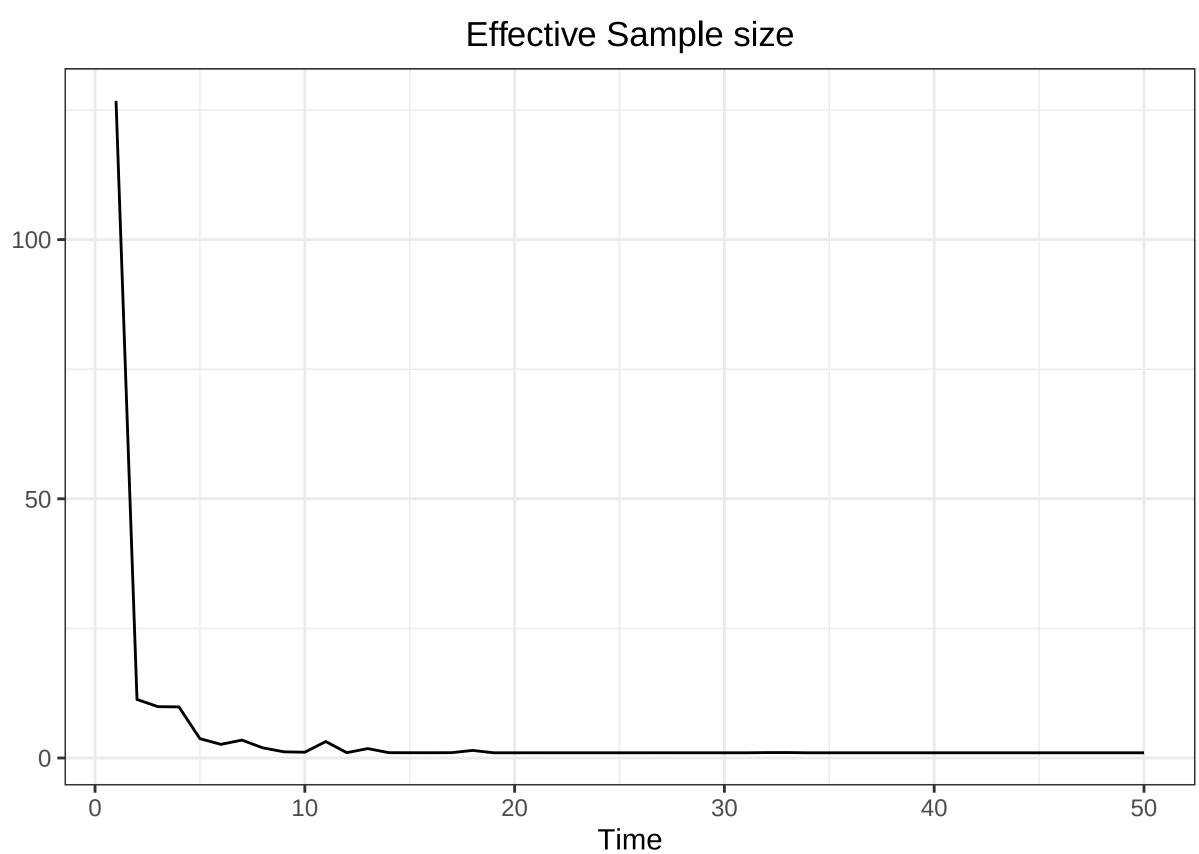


Figure 2: Kalman Filtered States with credible interval (in red)

Implementation With reference to the random walk plus noise of section XX, let $\{(x_0, w_0)^{(i)}\}_{i=1}^N$ summarizes $p(x_0|y_0)$ such that, for example, $E(g(x_0)|y_0) \approx \frac{1}{N}\sum_{i=1}^N w_0^{(i)} g(x_0^{(i)})$. For $t = 1, ..., n$ where n is the length of the sample, at any iteration

- Draw $x_t^{(i)} \sim N(x_{t-1}^{(i)}, \tau^2)$ $\; i = 1, ..., N$ such that $\{(x_t, w_{t-1})^{(i)}\}_{i=1}^N$ summarizes $p(x_t|y_{t-1})$
- Set $w_t^{(i)} = w_{t-1}^{(i)} f_N(y_t; x_t^{(i)}, \sigma^2)$ $\; i = 1, ..., N$ such that $\{(x_t, w_t)^{(i)}\}_{i=1}^N$ summarizes $p(x_t|y_t)$

```
SISfun<-function(data,N,m0,C0,tau,sigma){
  xs<-NULL
  ws<-NULL
  ess<-NULL
  x  = rnorm(N,m0,sqrt(C0))
  w  = rep(1/N,N)
  for(t in 1:length(data)){
    x    = rnorm(N,x,tau)                  #sample from N(x_{t-1},tau)
    w    = w*dnorm(data[t],x,sigma)        #update weight
    xs = rbind(xs,x)
    ws = rbind(ws,w)

    wnorm= w/sum(w)                        #normalized weight
    ESS  = 1/sum(wnorm^2)                  #effective sample size

    ess =rbind(ess,ESS)
  }

  return(list(xs=xs,ws=ws,ess=ess))
}
```



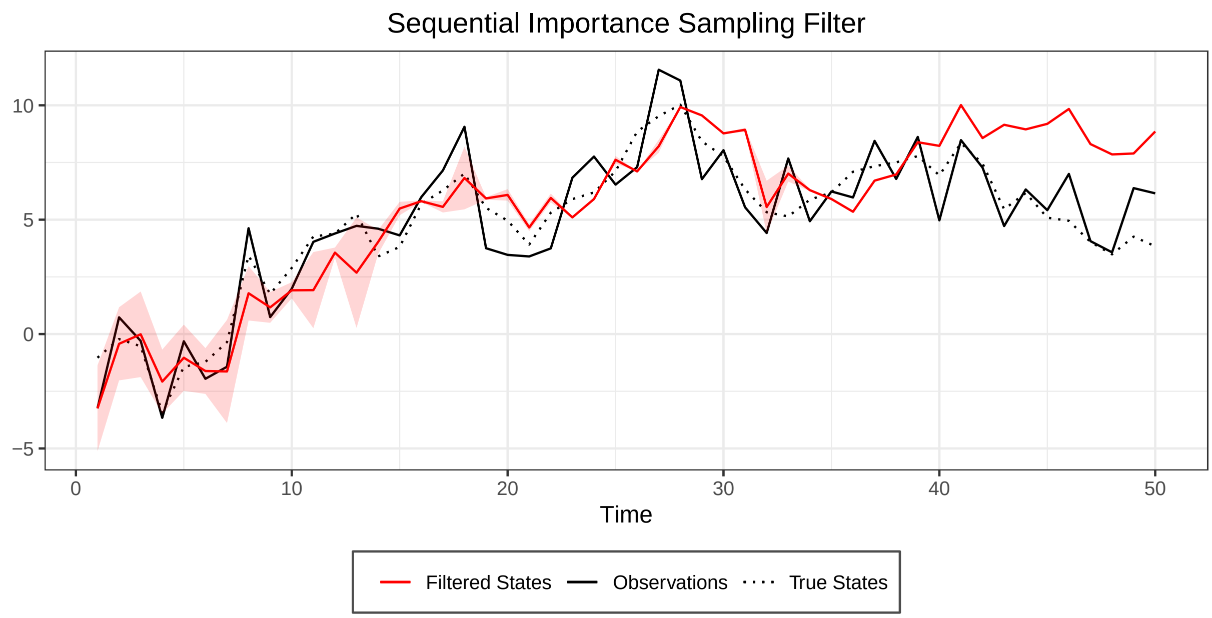


Figure 3: SIS Filtered States with credible interval (in red)

Particle Filter

With reference to the random walk plus noise of section XX, let $\{(x_0, w_0)^{(i)}\}_{i=1}^N$ summarizes $p(x_0|y_0)$ such that, for example, $E(g(x_0)|y_0) \approx \frac{1}{N}\sum_{i=1}^N w_0^{(i)} g(x_0^{(i)})$. For $t = 1, ..., n$ where n is the length of the sample, at any iteration

- Draw $x_t^{(i)} \sim N(x_{t-1}^{(i)}, \tau^2)\quad i = 1, ..., N$ such that $\{(x_t, w_{t-1})^{(i)}\}_{i=1}^N$ summarizes $p(x_t|y_{t-1})$
- Set $w_t^{(i)} = w_{t-1}^{(i)} f_N(y_t; x_t^{(i)}, \sigma^2)\quad i = 1, ..., N$ such that $\{(x_t, w_t)^{(i)}\}_{i=1}^N$ summarizes $p(x_t|y_t)$

In addition, when $ESS < ESS_0$ [1], resempling applies

- Draw a sample of size N, $x_t^{(1)}, ..., x_t^{(N)}$,from the discrete distribution $P(x_t = x_t^{(i)}) = w_t^{(i)},\quad i = 1, ..., N$
- Reset the weights: $w_t^{(i)} = N^{-1}$, $i = 1, ..., N$.

```
PFfun<-function(data,N,m0,C0,tau,sigma,r){
  if(missing(r)){r=2}else{}
  xs<-NULL
  ws<-NULL
  ess<-NULL
  x  = rnorm(N,m0,sqrt(C0))
  w  = rep(1/N,N)

  for(t in 1:length(data)){

    x<-rnorm(N,x,tau)
    w1<-w*dnorm(data[t],x,sigma)
```

[1]In our example we fix $ESS_0 = N/2$, this is an arbitrary common rule of thumb.

```r
    w = w1/sum(w1)
    ESS  = 1/sum(w^2)

    if(ESS<(N/r)){
      index<-sample(N,size=N,replace=T,prob=w)
      x<-x[index]
      w<-rep(1/N,N)
    }else{}

    xs = rbind(xs,x)
    ws = rbind(ws,w)
    ess =rbind(ess,ESS)
  }
  return(list(xs=xs,ws=ws,ess=ess))
}
```

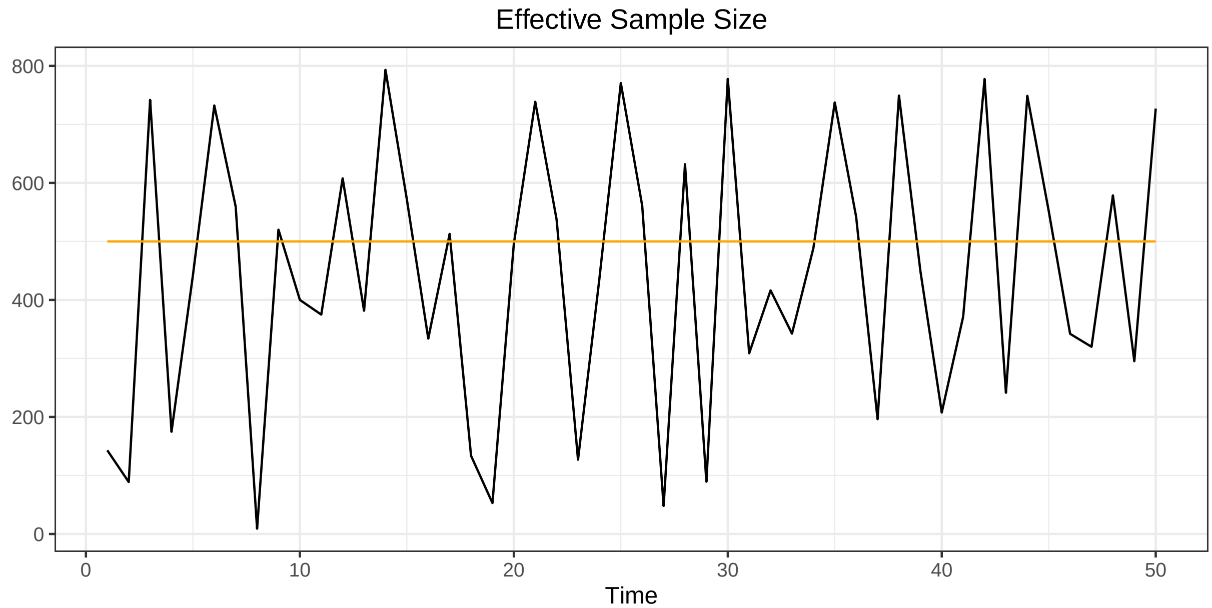


Figure 4: Effective Sample Size

Particle Filter Petroni more efficient

```r
PF2fun<-function(data,N,m0,C0,tau,sigma,r){
  if(missing(r)){r=2}else{}
  xs<-NULL
  ws<-NULL
  ess<-NULL
  x  = rnorm(N,m0,sqrt(C0))
  importancesd<-sqrt(tau - tau^2 /(tau + sigma))
  predsd <- sqrt(sigma+tau)
  w  = rep(1/N,N)

  for(t in 1:length(data)){
```

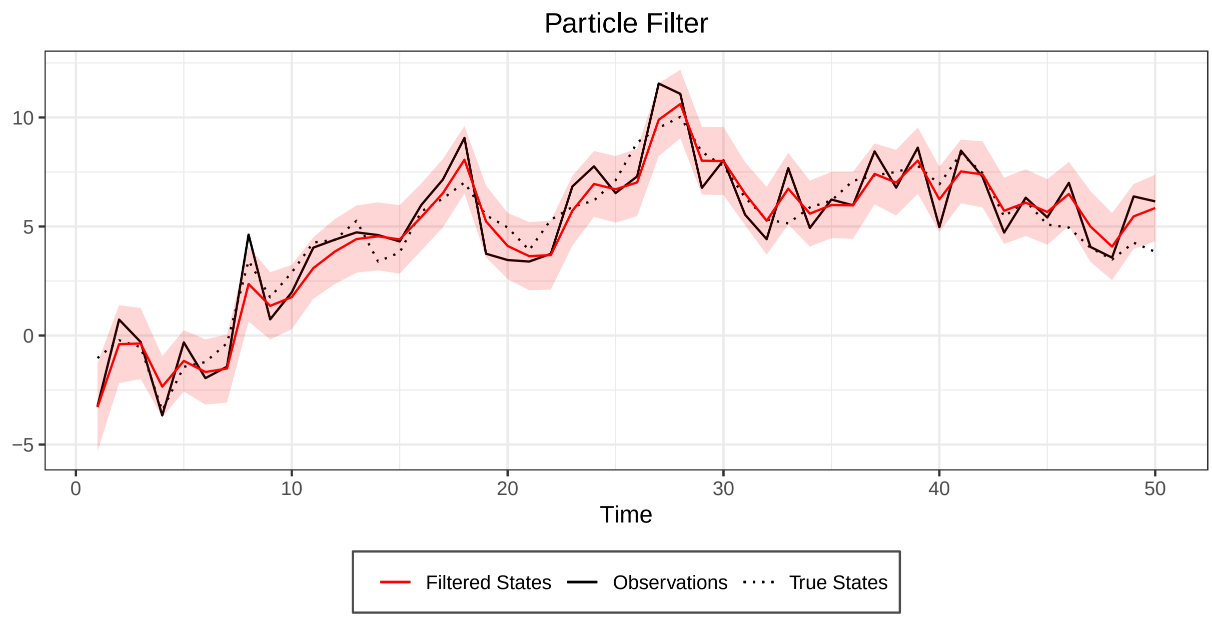
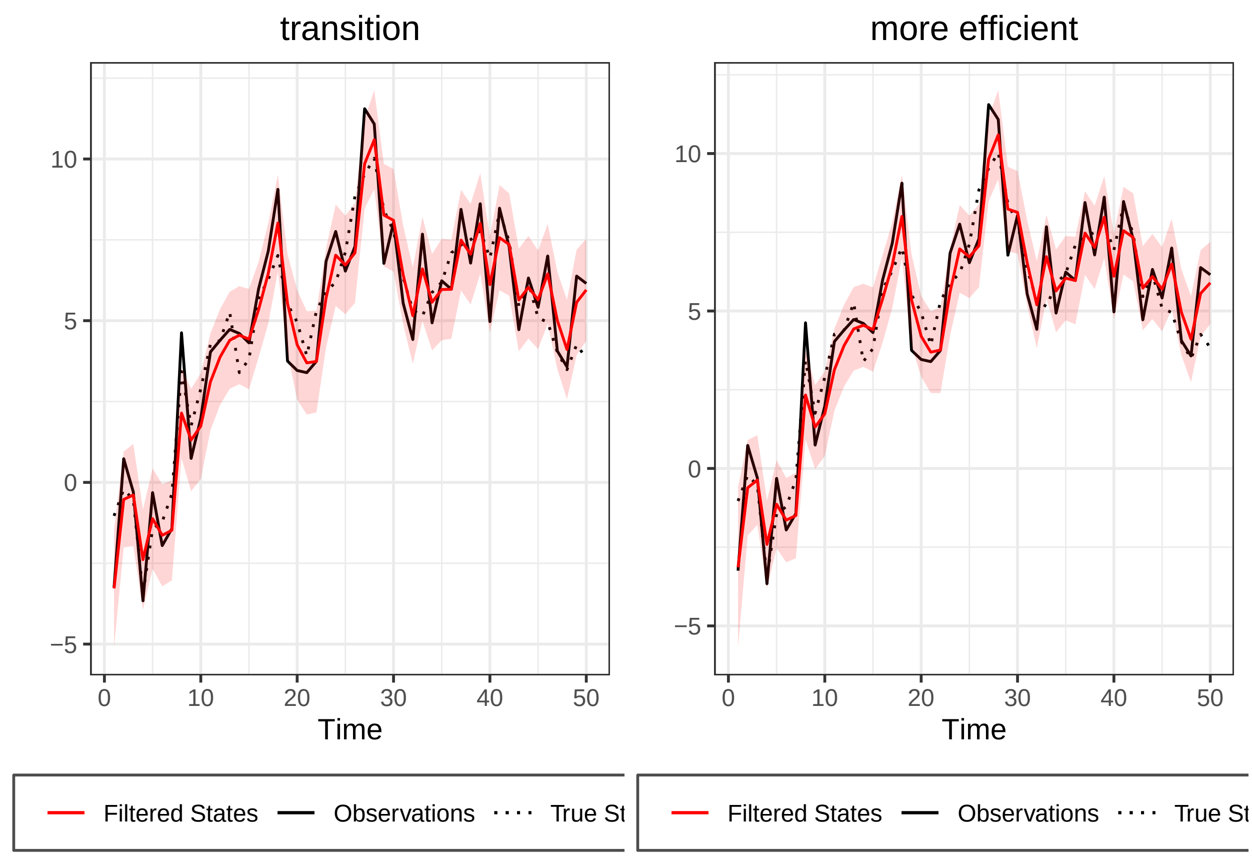


Figure 5: Particle Filtered States with credible interval (in red)

```
    means<-x+(tau/(tau+sigma))*(data[t]-x)
    x<-rnorm(N,means,importancesd)
    w1<-w*dnorm(data[t],x,predsd)

    w = w1/sum(w1)
    ESS  = 1/sum(w^2)

    if(ESS<(N/r)){
      index<-sample(N,size=N,replace=T,prob=w)
      x<-x[index]
      w<-rep(1/N,N)
    }else{}

    xs = rbind(xs,x)
    ws = rbind(ws,w)
    ess =rbind(ess,ESS)
  }
  return(list(xs=xs,ws=ws,ess=ess))
}
```

comparison between Particle Filters




```
Comparethetwo<-matrix(NA,nrow=2,ncol=2)
colnames(Comparethetwo)<-c("Credible Interval","RMSE")
rownames(Comparethetwo)<-c("transition","more efficient")
Comparethetwo[1,1]<-CredInt(pf1)
Comparethetwo[2,1]<-CredInt(pf2)
Comparethetwo[1,2]<-RMSE(truex,Filtervalues(pf1)$mean)
Comparethetwo[2,2]<-RMSE(truex,Filtervalues(pf2)$mean)
```

Table 1: Comparison

	Credible Interval	RMSE
transition	0.94	0.883
more efficient	0.90	0.873

Guided Particle Filter

Let's consider another toy example. Suppose that we have the following model

```
GPFfun<-function(data,N,m0,C0,tau,sigma,r){
 if(missing(r)){r=2}else{}
   xs<-NULL
  ws<-NULL
  ess<-NULL
  x  = rnorm(N,m0,sqrt(C0))
```

```
  w   = rep(1/N,N)

  for(t in 1:length(data)){

    xprev<-x
    x<-rnorm(N,x,tau)
    w1<-w*dnorm(data[t],x,sigma)*dnorm(x,xprev,tau)*I(x>0)/dnorm(x,xprev,tau)

    w = w1/sum(w1)
    ESS  = 1/sum(w^2)

    if(ESS<(N/r)){
      index<-sample(N,size=N,replace=T,prob=w)
      x<-x[index]
      w<-rep(1/N,N)
    }else{}

    xs = rbind(xs,x)
    ws = rbind(ws,w)
    ess =rbind(ess,ESS)
  }
  return(list(xs=xs,ws=ws,ess=ess))
}
```

With reference to the linear gaussian model of section XX, a very basic auxiliary particle sampling technique is

- Let $\{(x_{t-1}, w_{t-1})^{(i)}\}_{i=1}^{N}$ summarizes $p(x_{t-1}|y_{t-1})$

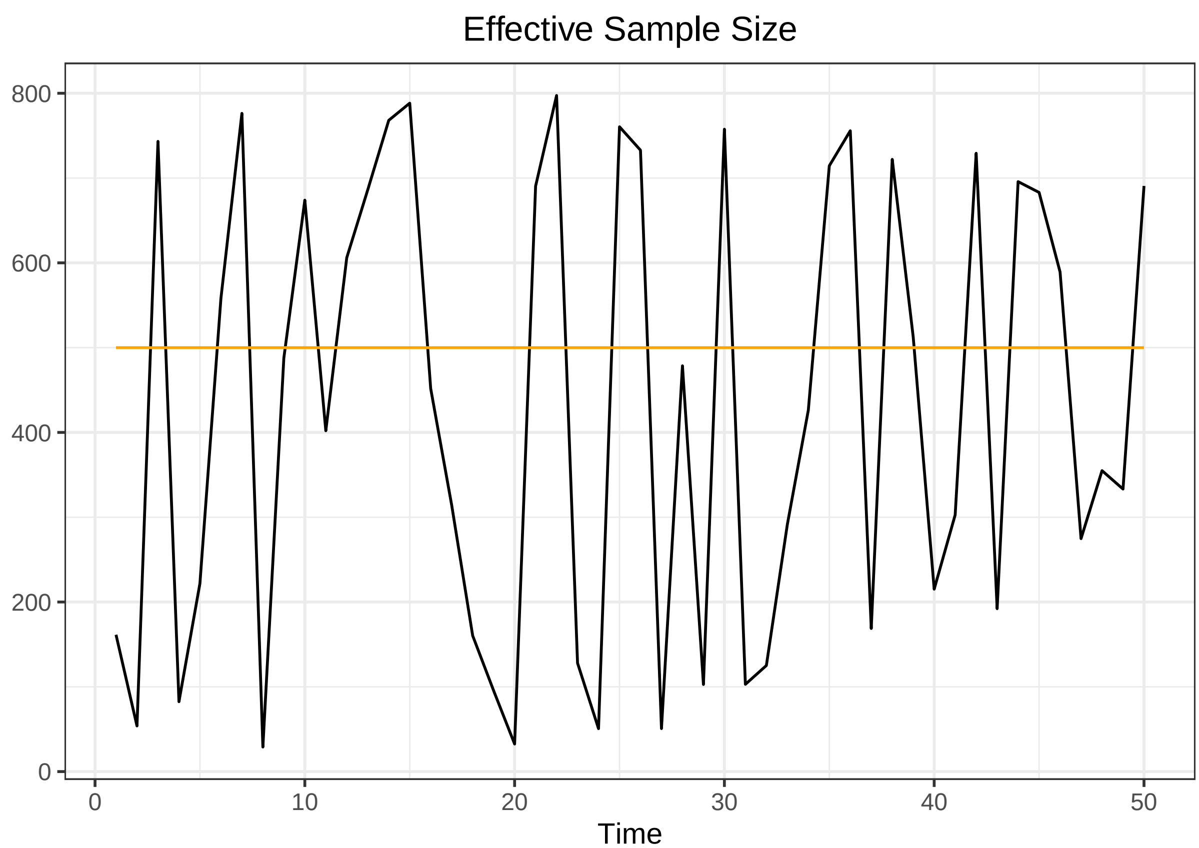
For $k = 1, ..., N$

- Draw I_k with $P(I_k) \propto w_{t-1}^{(i)} f(y_t|g(x_{t-1}^{(i)}))$ where $g(x_{t-1}^{(i)}) = E(X_t|X_{t-1})$
- Draw $x_t^{(k)} \sim N(x_{t-1}^{(I_k)}, \tau^2)$
- Set $w_t^{(k)} = \frac{f_N(y_t|x_t^{(k)})}{f_N(y_t|g(x_{t-1}^{(I_k)}))}$

The remaining steps are the same of the particle filter already seen in section XX.

```
APFfun<-function(data,N,m0,C0,tau,sigma,r){
  if(missing(r)){r=2}else{}
  xs<-NULL
  ws<-NULL
  ess<-NULL
  x   = rnorm(N,m0,sqrt(C0))
  w   = rep(1/N,N)

  for(t in 1:length(data)){

    weight = w*dnorm(data[t],x,sigma)
```

```
    k    = sample(1:N,size=N,replace=TRUE,prob=weight)
    x1    = rnorm(N,x[k],tau)
    lw   = dnorm(data[t],x1,sigma,log=TRUE)-dnorm(data[t],x[k],sigma,log=TRUE)
    w    = exp(lw)
    w    = w/sum(w)
    ESS   = 1/sum(w^2)

    if(ESS<(N/r)){
      index<-sample(N,size=N,replace=T,prob=w)
      x1<-x1[index]
      w<-rep(1/N,N)
    }else{}

    x <- x1
    xs = rbind(xs,x)
    ws = rbind(ws,w)
    ess =rbind(ess,ESS)

  }
  return(list(xs=xs,ws=ws,ess=ess))
}
```



Liu and West Filter
Consider the toy example of section XX. Let $\psi = (\sigma^2, \tau^2)$ be unknown and assign a gamma prior on these

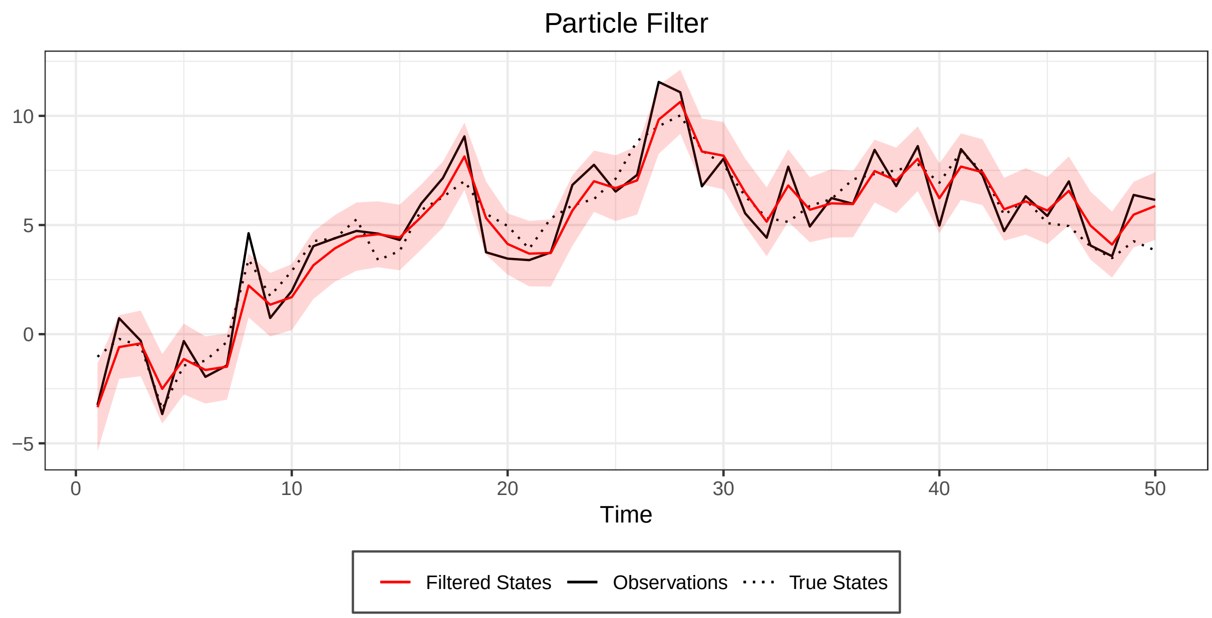


Figure 6: Particle Filtered States with credible interval (in red)

parameters

$$\sigma^2 \sim G(\alpha_v, \beta_v)$$
$$\tau^2 \sim G(\alpha_w, \beta_w)$$

or let assign them a uniform prior if we have no knowledge on hyperparameters. After having drown the hyperparameters indipendently form their priors and having set $w_0^{(i)} = N^{-1}$, $i = 1, ..., N$, and $\hat{\pi}_0 = \sum_{i=1}^{N} w_0^{(i)} \delta_{(x_0^{(i)}, \psi^{(i)})}$, for t=1,..T

- Compute $\hat{\psi} = E_{\hat{\pi}_{t-1}}(\psi)$ and $\Sigma = Var_{\hat{\pi}_{t-1}}(\psi)$. For $i = 1, ..., N$ set

$$m(\psi^{(i)}) = a\psi^{(i)} + (1-a)\overline{\psi}$$
$$v(\psi^{(i)}) = (1-a^2)\Sigma$$

  and

$$\alpha(\psi^{(i)}) = \frac{m(\psi^{(i)})^2}{v(\psi^{(i)})}$$
$$\beta(\psi^{(i)}) = \frac{m(\psi^{(i)})}{v(\psi^{(i)})}$$

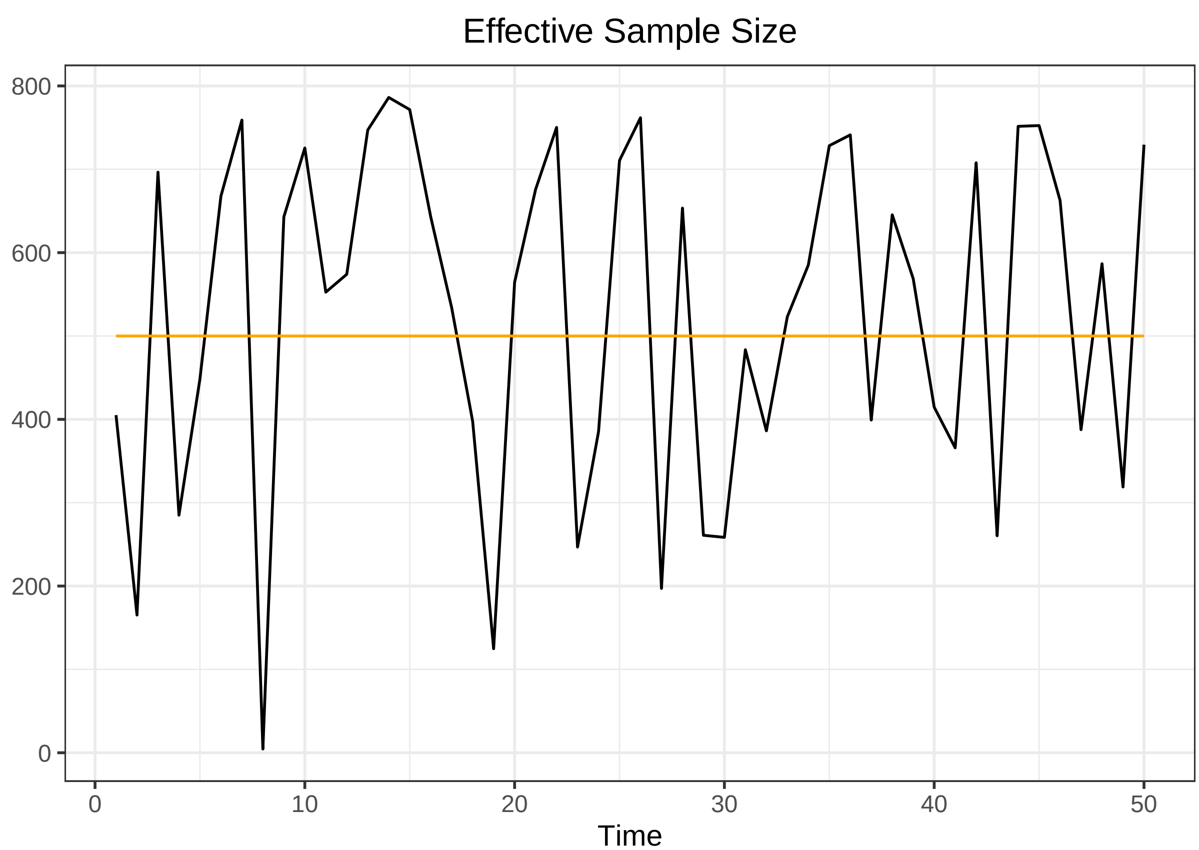
  For $k = 1, ..., N$

  - Draw I_k with $P(I_k = i) \propto w_{t-1^{(i)}} f_N(y_t | g(x_t^{(i)}), m(\psi^{(i)}))$ where for simplicity $g(x_t^{(i)}) = E(x_t | x_{t-1}, m(\psi^{(i)}))$
  - Draw $\psi^{(k)} \sim G(\alpha(\psi^{(I_k)}), \beta(\psi^{(I_k)}))$
  - Draw $x_t^{(k)} \sim N(x_{t-1}^{(I_k)}, \tau^{2^{(k)}})$
  - Set $\tilde{w}_t^k = \frac{f_N(y_t | x_t^{(k)}, \psi = \psi^{(k)})}{f_N(y_t | g(x_t^{(I_k)}), \psi = m(\psi)^{(I_k)})}$

- Normalize the weights
- Compute the effective sample size (ESS)
- If $ESS < N/2$, resample:
  - Draw a sample of size N, $x_t^{(1)}, ..., x_t^{(N)}$,from the discrete distribution $P((x_t, \psi) = (x_t^{(i)}, \psi^{(i)})) = w_t^{(i)}, \quad i = 1, ..., N$
  - Reset the weights: $w_t^{(i)} = N^{-1}$, $i = 1, ..., N$.

```
LWfun<-function(data,N,m0,C0,alphav,betav,alphaw,betaw,delta,unif,r){
  if(missing(r)){r=2}else{}
  xs     = rnorm(N,m0,sqrt(C0))
  if(unif==T){
  pars   = cbind(runif(N,0,10),runif(N,0,10))}else{}
  pars   = cbind(rgamma(N,shape=alphav,scale=betav),rgamma(N,shape=alphaw,scale=betaw))
  a      = (3*delta-1)/(2*delta)
  h2     = 1-a^2
  parss  = array(0,c(N,2,n))
  xss    = NULL
  ws     = NULL
  ess    = NULL
  w      = rep(1/N,N)
  for (t in 1:length(data)){
    meanV = weighted.mean(pars[,1],w)
    varV  = weighted.mean((pars[,1]-meanV)^2,w)
    meanW = weighted.mean(pars[,2],w)
    varW  = weighted.mean((pars[,2]-meanW)^2,w)

    muV = a*pars[,1]+(1-a)*meanV
    sigma2V = (1-a^2)*varV
    alphaV = muV^2/sigma2V
    betaV = muV/sigma2V

    muW = a*pars[,1]+(1-a)*meanW
    sigma2W = (1-a^2)*varW
    alphaW = muW^2/sigma2W
    betaW = muW/sigma2W

    weight      = w*dnorm(data[t],xs,sqrt(muV))
    k           = sample(1:N,size=N,replace=T,prob=weight)

    pars[,1]<-rgamma(N,shape=alphaV[k],rate=betaV[k])
    pars[,2]<-rgamma(N,shape=alphaW[k],rate=betaW[k])

    xsprevious<-xs[k]
    xs = rnorm(N,xs[k],sqrt(pars[,2]))

    w           = exp(dnorm( data[t],xs,sqrt(pars[,1]),log=T)-
                      dnorm( data[t],xsprevious,sqrt(muV[k]),log=T))
    w           = w/sum(w)
    ESS         = 1/sum(w^2)

    if(ESS<(N/r)){
```

```
      index<-sample(N,size=N,replace=T,prob=w)
      xs<-xs[index]
      pars<-pars[index,]
      w<-rep(1/N,N)
    }else{
      xs<-xs
      pars<-pars
    }


    xss          = rbind(xss,xs)
    parss[,,t]   = pars
    ws           = rbind(ws,w)
    ess          = rbind(ess,ESS)
  }
  return(list(xss=xss,parss=parss,ws=ws,ess=ess))
}
```



COMPARISON (ANOTHER SECTION)

Table 2: RMSE

N	Threshold	KF	PF	APF	LWF
20	0.5	0.879	0.936	0.783	0.954
100	0.5	0.879	0.870	0.880	0.899
500	0.5	0.879	0.859	0.867	0.889

Table 3: RMSE

N	Threshold	KF	PF	APF	LWF
1000	0.50	0.879	0.896	0.875	0.901
1000	0.25	0.879	0.869	0.897	0.881
1000	0.10	0.879	0.896	0.875	0.886

STOCHASTIC VOLATILITY

Basic specification of Stochastic Volatilty Model

$$y_t|x_t \sim N(0, e^{x_t}) \tag{4}$$
$$x_t|x_{t-1} \sim N(\alpha + \beta x_{t-1}, \tau^2) \tag{5}$$

Particle Filter

```
SVPFfun<-function(data,N,m0,C0,alpha,beta,tau,r){
  if(missing(r)){r=2}else{}
  xs<-NULL
  ws<-NULL
  ess<-NULL
  x   = rnorm(N,m0,sqrt(C0))
  w   = rep(1/N,N)

  for(t in 1:length(data)){

    x<-rnorm(N,alpha+beta*x,tau)
    w1<-w*dnorm(data[t],0,exp(x/2))

    w = w1/sum(w1)
    ESS  = 1/sum(w^2)

    if(ESS<(N/r)){
      index<-sample(N,size=N,replace=T,prob=w)
      x<-x[index]
      w<-rep(1/N,N)
    }else{}

    xs = rbind(xs,x)
    ws = rbind(ws,w)
    ess =rbind(ess,ESS)
  }
```

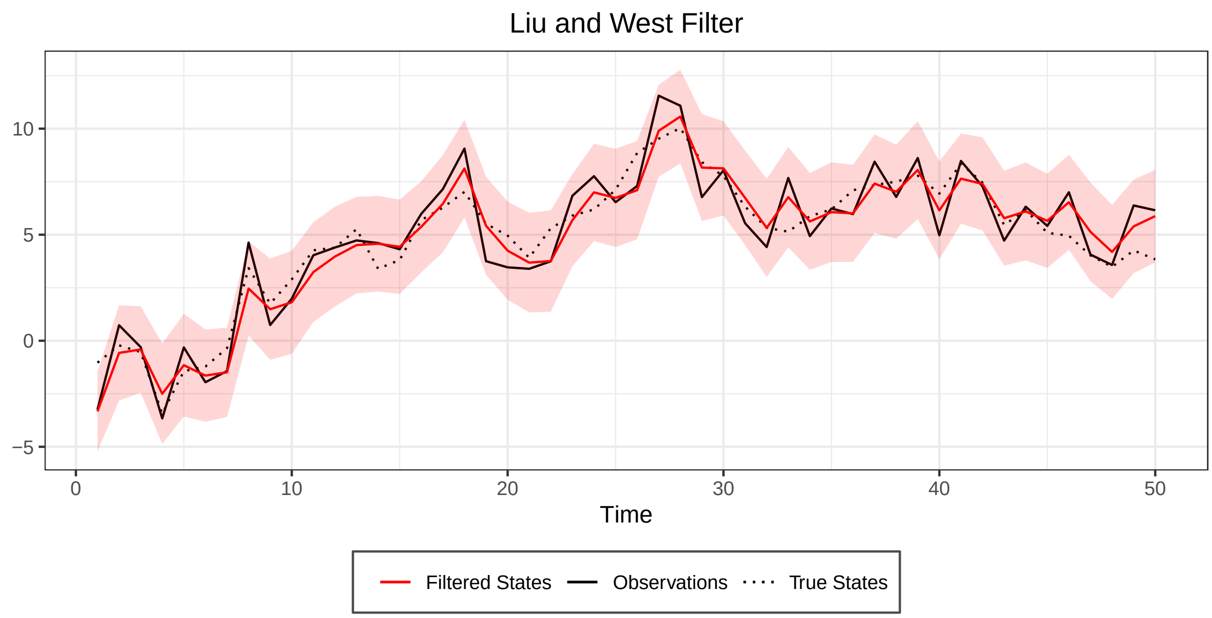


Figure 7: Particle Filtered States with credible interval (in red)

```
  return(list(xs=xs,ws=ws,ess=ess))
}
```

Auxiliary Particle Filter

```
SVAPFfun<-function(data,N,m0,C0,alpha,beta,tau,r){
  if(missing(r)){r=2}else{}
  xs<-NULL
  ws<-NULL
  ess<-NULL
  x  = rnorm(N,m0,sqrt(C0))
  w  = rep(1/N,N)

  for(t in 1:length(data)){

    weight = w*dnorm(data[t],0,exp(x/2))
    k    = sample(1:N,size=N,replace=TRUE,prob=weight)
    x1    = rnorm(N,alpha+beta*x[k],tau)
    lw   = dnorm(data[t],0,exp(x/2),log=TRUE)-dnorm(data[t],0,exp(x/2),log=TRUE)
    w    = exp(lw)
    w    = w/sum(w)
    ESS   = 1/sum(w^2)

    if(ESS<(N/r)){
      index<-sample(N,size=N,replace=T,prob=w)
      x1<-x1[index]
      w<-rep(1/N,N)
    }else{}

    x <- x1
```

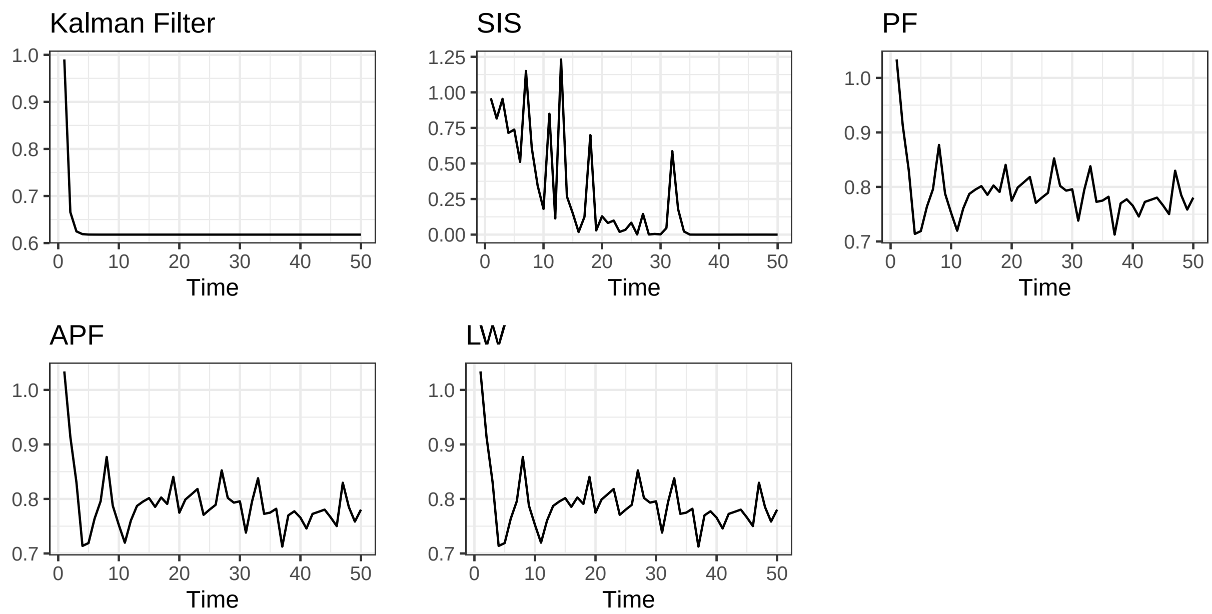


Figure 8: Variances

```
    xs = rbind(xs,x)
    ws = rbind(ws,w)
    ess =rbind(ess,ESS)

  }
  return(list(xs=xs,ws=ws,ess=ess))
}
```

Liu and West

```
SVLWfun<-function(data,N,m0,C0,ealpha,valpha,ebeta,vbeta,nu,lambda){

xs = rnorm(N,m0,sqrt(C0))
pars    = cbind(rnorm(N,ealpha,sqrt(valpha)),rnorm(N,ebeta,sqrt(vbeta)),
               log(1/rgamma(N,nu/2,nu*lambda/2)))
delta   = 0.75
a       = (3*delta-1)/(2*delta)
h2      = 1-a^2
parss   = array(0,c(N,3,n))
xss     = NULL
ws      = NULL
ESS     = NULL
w       = rep(1/N,N)
for (t in 1:length(data)){

  #mpar        = apply(pars,2,mean) #alternatively use this
  mpar<-c()
  for(i in 1:3){
  mpar[i]     = weighted.mean(pars[,i],w)
  }
```

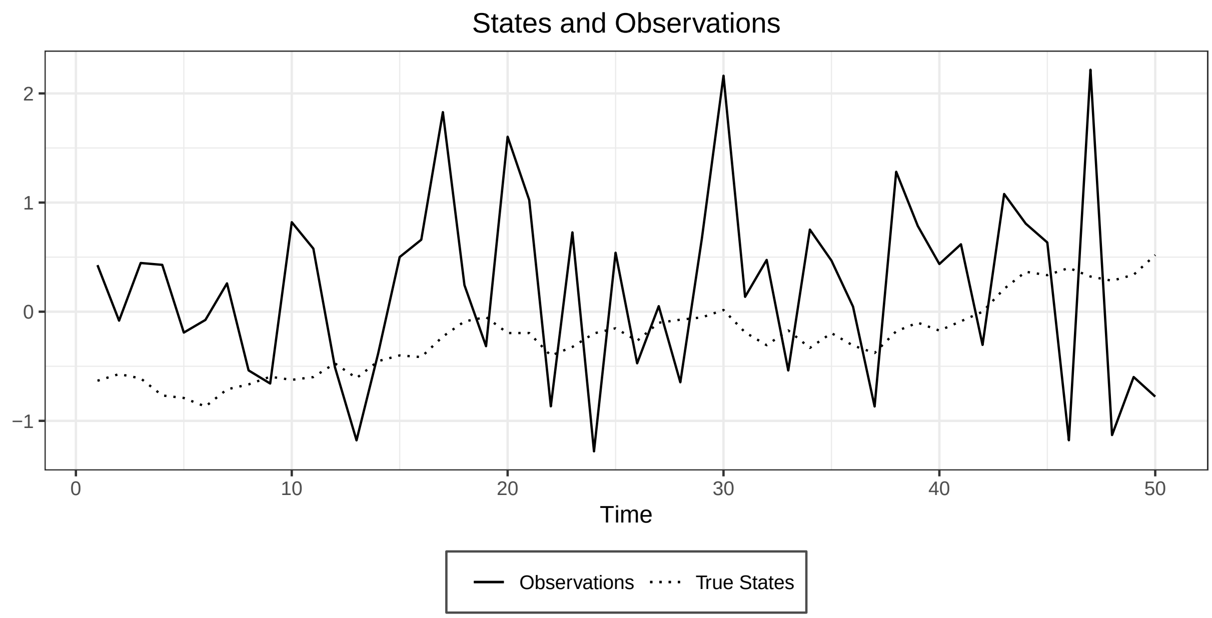


Figure 9: SV

```
  vpar         = var(pars)
  ms           = a*pars+(1-a)*matrix(mpar,N,3,byrow=T)
  mus          = pars[,1]+pars[,2]*xs
  weight       = w*dnorm(data[t],0,exp(mus/2))
  k            = sample(1:N,size=N,replace=T,prob=weight)
  ms1          = ms[k,] + matrix(rnorm(3*N),N,3)%*%chol(h2*vpar)
  xt           = rnorm(N,ms1[,1]+ms1[,2]*xs[k],exp(ms1[,3]/2))
  w            = dnorm(data[t],0,exp(xt/2))/dnorm(data[t],0,exp(mus[k]/2))
  w            = w/sum(w)
  ESS          = 1/sum(w^2)


  if(ESS<(N/2)){
    index<-sample(N,size=N,replace=T,prob=w)
    xs<-xt[index]
    pars<-ms1[index,]
    w<-rep(1/N,N)
  }else{
    xs<-xt
    pars<-ms1
  }

  xss          = rbind(xss,xs)
  parss[,,t]   = pars
  ws           = rbind(ws,w)
}
return(list(xs=xss,pars=parss,ws=ws))
}
```

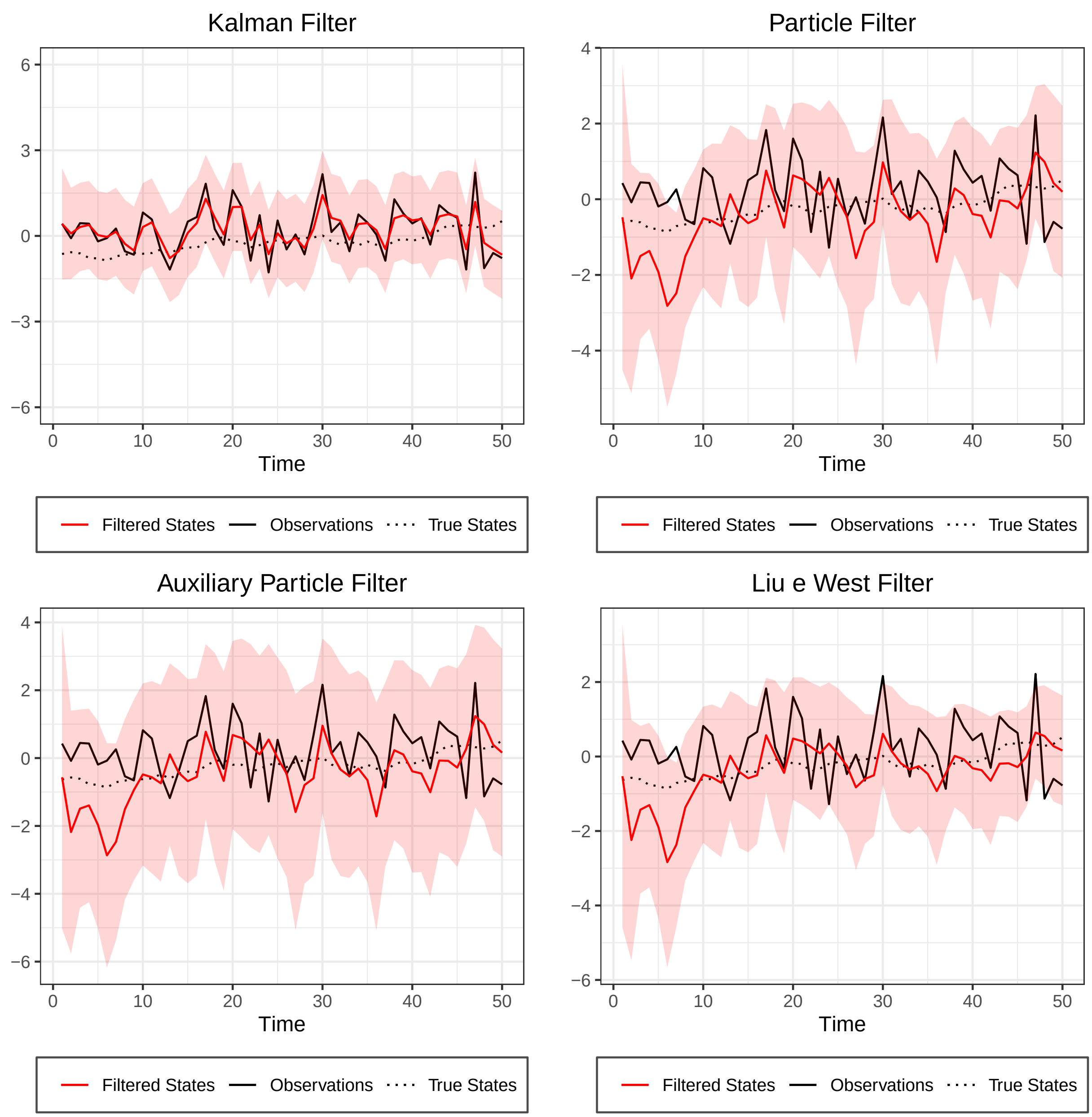


Figure 10: Comparison filter performances