R Notebook

Implementation

In this section we present a practical illustration of the algorithms 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 already mentioned before, in this case the filtering distribution can be computed in closed form solutions using the Kalman filter. However, this toy example will be used also to illustrate more involving filtering strategies described in this work. We believe indeed that it represents a useful starting point to understand the logic of the algorithms which may be eventually replicated when dealing with more complex models.
The filtering strategy is applied to 50 simulated data. Figure XX shows the simulated true states sequence assuming as data generating process the Equation (2) with $\tau^2 = 1$ and simulated observed sequence process form Equation (1) with $\sigma^2 = 1$.

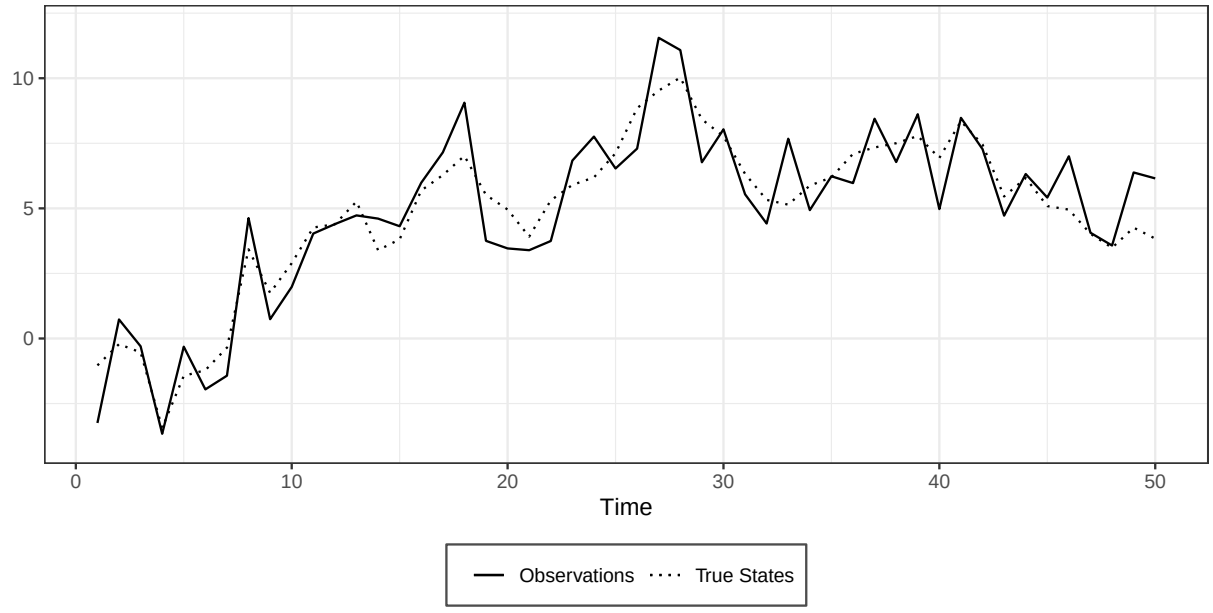

Figure 1: Simulated random walk plus noise model

The Kalman Filter for this model is implemented as described below.

Algorithm 0.1 *Kalman Filter for Random Walk plus Noise Model*

- *Initialize* $\theta_0 \sim N(m_0, C_0)$
- *For* $t = 1, ..., n$*:*

1. *Compute the one-step-ahead state predictive distribution at time $t-1$, notice that if $t=1$ no data have been observed yet and therefore $x_1|x_0 \sim N(a_1, R_1)$ otherwise*

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

2. *Compute the filtering distribution at time t as $p(x_t|y_{1:t}) \propto p(x_t|y_{1:t-1})p(y_t|x_t)$, i.e. the product of the one-step-ahead state predictive distribution and the likelihood*

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

Our `DLM` function implement in **R** replicate this steps.

`DLM(data,sig2,tau2,m0,C0)`

Arguments

`data`	the observed process. It has to be a vector or a univariate time series.
`sig2`	the variance σ^2 in Equation (1)
`tau2`	the variance τ^2 in Equation (2)
`m0`	central value of the normal prior state distribution
`C0`	variance of the normal prior state distribution

```
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 Figure XX below filtered states estimated using Kalman Filter with $x_0 \sim N(0, 100)$ and $\sigma^2 = \tau^2 = 1$ are compared to the true states values. Notice how closely the filtered states follow the observations and the goodness of the approximation of the true states. 95 percent credible intervals are computed as

$$[E(\theta_t|y_{1:t}) - z_{1-\alpha/2}\sqrt{V(\theta_t|y_{1:t})}, E(\theta_t|y_{1:t}) + z_{1-\alpha/2}\sqrt{V(\theta_t|y_{1:t})}]$$

The discussion of Kalman Filter will continue in Section XX where we compare it to the Particle Filter.

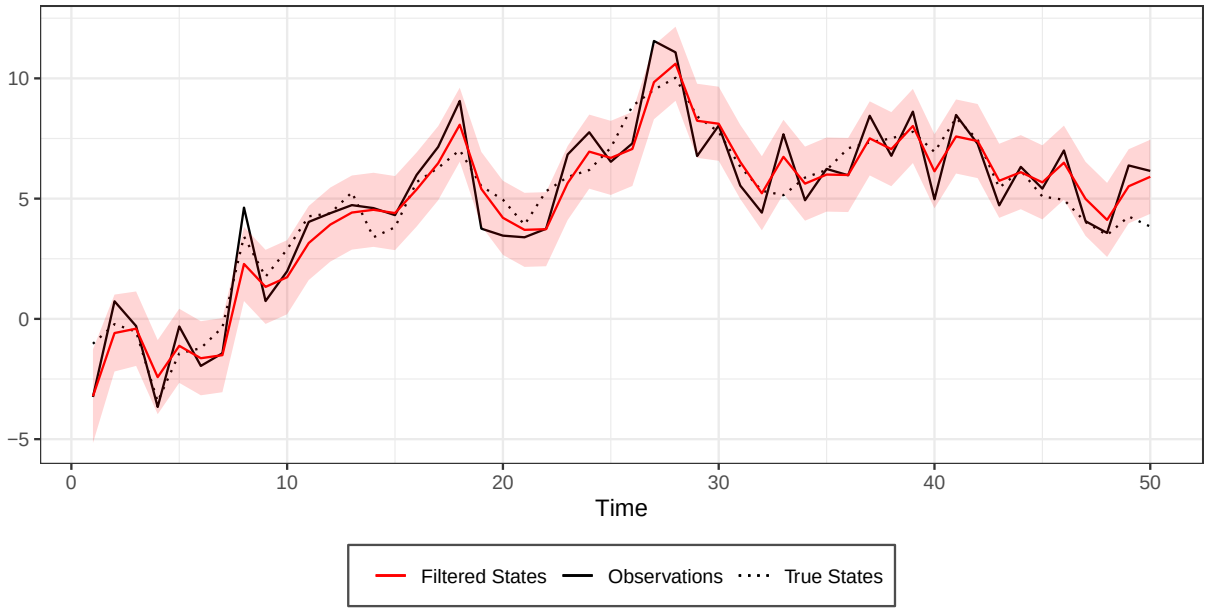

Figure 2: Kalman Filtered States with credible interval (in red)

Implementation Consider again the random walk plus noise model of section XX,the challenge here is to estimate the filtered states using a Sequential Importance Sampling algorithm. The main idea of the SIS applied to this univariate linear gaussian model is described in the following algorithm. We indicate with n the sample size of time observations and with N the generated sample size for each step of the Sequenial Monte Carlo.

Algorithm 0.2 *SIS filter for Random Walk plus Noise Model*

- *Let* $\{(x_0, w_0)^{(i)}\}_{i=1}^N$ *summarizes* $p(x_0|y_0)$ *such that, for example,* $E(g(x_0)|y_0) \approx \sum_{i=1}^N w_0^{(i)} g(x_0^{(i)})$. *In particular, initialize* $(x_0^{(1)}, ..., x_0^{(N)})$ *form* $N(m_0, C_0)$ *and set* $w_0^{(i)} = N^{-1}\ \forall\ i = 1, ..., N$.
- *For* $t = 1, ..., n$*:*
  1. *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* $p(x_t|y_{t-1})$
  2. *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* $p(x_t|y_t)$
  3. *Set* $p(x_t|y_t) = \sum_{i=1}^N w_t^{(i)} \delta_{x_t^{(i)}}$

Our `SISfun` function implemented in **R** replicate this steps.

`SISfun(data,N,m0,C0,tau,sigma)`

Arguments

`data` the observed process. It has to be a vector or a univariate time series.
`N` number of particles generated at each step
`m0` central value of the normal prior state distribution
`C0` variance of the normal prior state distribution
`tau` the standard deviation τ in Equation (2)
`sigma` the standard deviation σ in Equation (1)

```r
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))
}
```

We have already discussed the reasons why the SIS algorithm does not provide a good strategy in the filtering problem. We provide a graphical intuition of what happens when we use such filtering strategy on a simulated dataset. We decide to set $N = 1000, m_0 = 0, C_0 = 100$ and $\tau = \sigma = 1$. The results shown in the following two plots shows a clear degeneration of the effective sample size and bad fit of filtered states with respect to the true values.

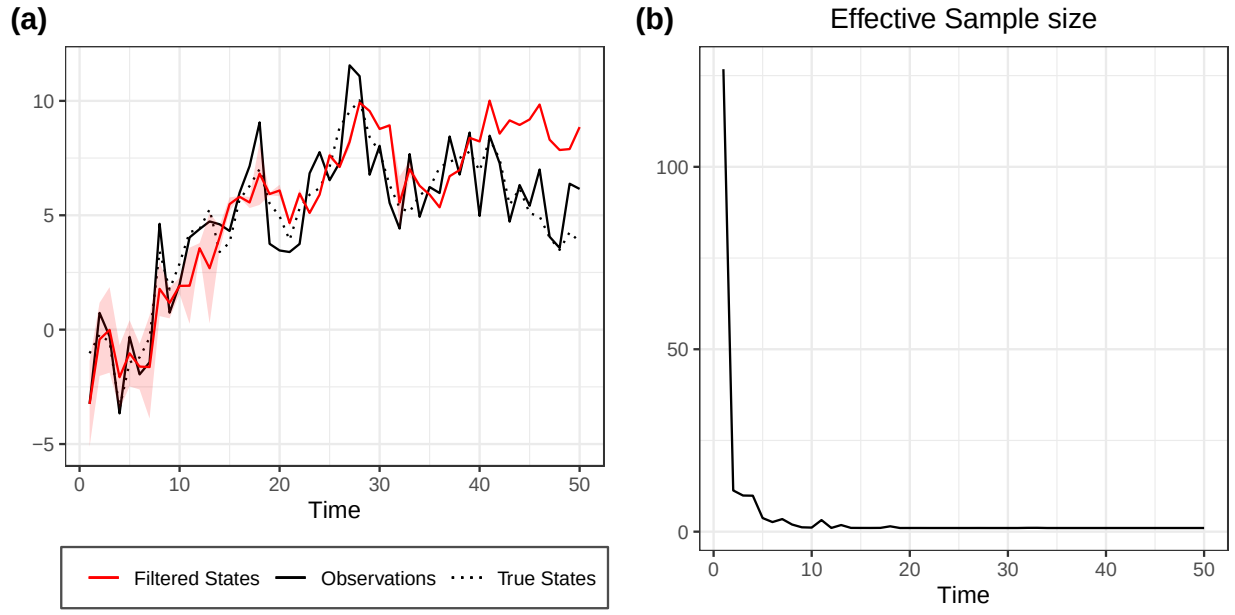

Figure 3: a) SIS Filtered States with credible interval (in red). b) Effective sample size.

Implementation In this section, Bootstrap Particle Filter will be used to estimate filtered states of the Random Walk plus Noise introduced in section XX. The overall strategy replicate the SIS filter with the addition of a ESS-based resampling step that applies when the effective sample size is smaller than a pre-determined threshold opportunely chosen (in our example we decide to follow a common rule of thumb

consisting in setting the threshold at $N/2$). The steps are presented in the following algorithm.

Algorithm 0.3 *BPF for Random Walk plus Noise Model*

- *Let* $\{(x_0, w_0)^{(i)}\}_{i=1}^N$ *summarizes* $p(x_0|y_0)$ *such that, for example,* $E(g(x_0)|y_0) \approx \sum_{i=1}^N w_0^{(i)} g(x_0^{(i)})$. *In particular, initialize* $(x_0^{(1)}, ..., x_0^{(N)})$ *form* $N(m_0, C_0)$ *and set* $w_0^{(i)} = N^{-1}\ \forall\ i = 1, ..., N$.
- *For* $t = 1, ..., n$:
  1. *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* $p(x_t|y_{t-1})$
  2. *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* $p(x_t|y_t)$
  3. *if* $ESS < N/2$ *then*
     (a) *Draw a sample of size N,* $(x_t^{(1)}, ..., x_t^{(N)})$, *from the discrete distribution* $P(x_t = x_t^{(i)}) = w_t^{(i)},\ \ i = 1, ..., N$
     (b) *Reset the weights:* $w_t^{(i)} = N^{-1}$, $i = 1, ..., N$.
  4. *Set* $p(x_t|y_t) = \sum_{i=1}^N w_t^{(i)} \delta_{x_t^{(i)}}$

These steps are resumed in our `PFfun` function.

`PFfun(data,N,m0,C0,tau,sigma,r)`

Arguments

`data` the observed process. It has to be a vector or a univariate time series.
`N` number of particles generated at each step
`m0` central value of the normal prior state distribution
`C0` variance of the normal prior state distribution
`tau` the standard deviation τ in Equation (2)
`sigma` the standard deviation σ in Equation (1)
`r` if present the threshold is set equal to N/r otherwise, if missing, the threshold is set equal to $N/2$

```
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)

    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))
}
```

The estimated states of the Boostrap Particle Filter togheter with the effective sample size are shown in Figure XX. We decide to set $N = 1000$,$m_0 = 0$,$C_0 = 100$ and $\tau = \sigma = 1$. Notice how the resampling step allows the effective sample size not to drop, improving results.

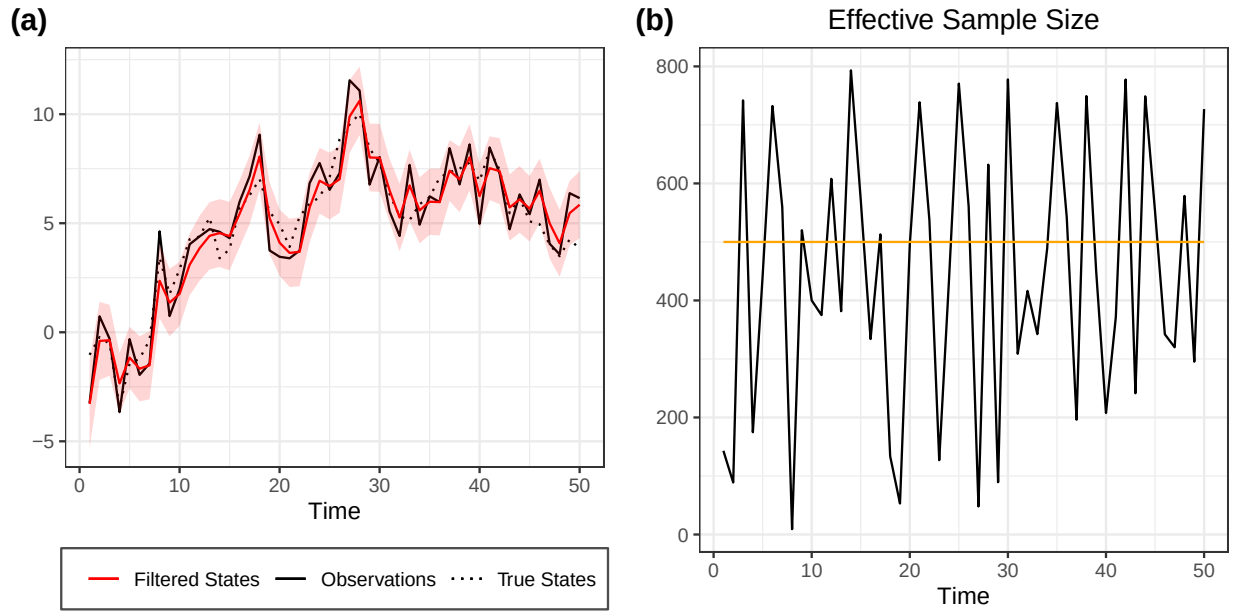

Figure 4: a) PF Filtered States with credible interval (in red). b) Effective sample size (in black) with threshold (in yellow).

Since the Random Walk plus Noise model allows for closed form solutions, we want to compare the results of Boostrap Particle Filter(BPF) with Kalman Filter(KF). As we can see from figure XX, when the number of particles generated at any interaction increases the results for both the estimated mean and the variance tend to converge [[SPIEGARE PERCHE']]. Moreover, comparing the Root Mean Square Errors of the filtered states with respect to the true state values, when the sample size increases the BPF decreases and, for large N, it reaches the accuracy of the Kalman Filter.

Table 1: Root Mean Square Errors

N	Threshold	KF	BPF
100	0.5	0.879	0.888
1000	0.5	0.879	0.886
10000	0.5	0.879	0.878

Implementation

The Bootstrap Particle Filter Approach for the Random Walk plus Noise Model described in Section XX

can be improved accounting for the observations in the importance transition density and it consists of generating x_t from its conditional distribution given x_{t-1} and y_t. In the Normal model we are considering, the optimal proposal will be a Normal density as well with mean and variance given by

$$\mu_{opt} = E(x_t|x_{t-1}, y_t) = x_{t-1} + \frac{\tau^2}{\tau^2+\sigma^2}(y_t - x_{t-1})$$
$$\sigma^2_{opt} = V(x_t|x_{t-1}, y_t) = \frac{\tau^2\sigma^2}{\tau^2+\sigma^2}$$

On the other hand, the incremental weights, using this importance transition density, are proportional to the conditional density of y_t given $x_{t-1} = x^{(i)}_{t-1}$, i.e $N(x^{(i)}_{t-1}, \tau^2+\sigma^2)$, evaluated at y_t. In other words the algorithm implemented in **R** is:

Algorithm 0.4 *GPF for Random Walk plus Noise Model*

- *Let* $\{(x_0, w_0)^{(i)}\}_{i=1}^N$ *summarizes* $p(x_0|y_0)$ *such that, for example,* $E(g(x_0)|y_0) \approx \sum_{i=1}^N w_0^{(i)} g(x_0^{(i)})$. *In particular, initialize* $(x_0^{(1)}, ..., x_0^{(N)})$ *from* $N(m_0, C_0)$ *and set* $w_0^{(i)} = N^{-1}\ \forall\ i = 1, ..., N$.
- *Compute* σ^2_{opt}
- *For* $t = 1, ..., n$*:*
  1. *Compute* μ_{opt}
  2. *Draw* $x_t^{(i)} \sim N(\mu_{opt}, \sigma^2_{opt})$ $i = 1, ..., N$ *such that* $\{(x_t, w_{t-1})^{(i)}\}_{i=1}^N$ *summarizes* $p(x_t|y_{t-1})$
  3. *Set* $w_t^{(i)} = w_{t-1}^{(i)} f_N(y_t; x_t^{(i)}, \sigma^2+\tau^2)$ $i = 1, ..., N$ *such that* $\{(x_t, w_t)^{(i)}\}_{i=1}^N$ *summarizes* $p(x_t|y_t)$
  4. *if* $ESS < N/2$ *then*
     (a) *Draw a sample of size N,* $(x_t^{(1)}, ..., x_t^{(N)})$*, from the discrete distribution* $P(x_t = x_t^{(i)}) = w_t^{(i)}$, $i = 1, ..., N$
     (b) *Reset the weights:* $w_t^{(i)} = N^{-1}$, $i = 1, ..., N$.
  5. *Set* $p(x_t|y_t) = \sum_{i=1}^N w_t^{(i)} \delta_{x_t^{(i)}}$

The `GPFfun` function resume this passages.

`GPFfun(data,N,m0,C0,tau,sigma,r)`

Arguments

`data` the observed process. It has to be a vector or a univariate time series.
`N` number of particles generated at each step
`m0` central value of the normal prior state distribution
`C0` variance of the normal prior state distribution
`tau` the standard deviation τ in Equation (2)
`sigma` the standard deviation σ in Equation (1)
`r` if present the threshold is set equal to N/r otherwise, if missing, the threshold is set equal to $N/2$

```
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))
  importancesd<-sqrt(tau - tau^2 /(tau + sigma))
```

```
  predsd <- sqrt(sigma+tau)
  w  = rep(1/N,N)

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

    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))
}
```

Let's provide directly a brief comparison between the Bootstrap Particle Filter (BPF) and the Guided Particle Filter (GPF). As we can see from the Figure XX, the Guided Particle Filter provides point estimates that are slightly better with respect to the ones of the BPF, and this is confirmed by the Table showing the RMSE. Moreover, also the variance is better, suggesting for higher precision.

Table 2: Root Mean Square Errors

N	Threshold	BPF	GPF
1000	0.50	0.916	0.880
1000	0.25	0.895	0.862
1000	0.10	0.869	0.881

Implementation

For illustration purposes, we are going implement an auxiliary particle filter with auxiliary function $g(x_{t-1}) = E(x_t|x_{t-1}) = x_{t-1}$.

Algorithm 0.5 *APF for Random Walk plus Noise Model*

- *Let* $\{(x_0, w_0)^{(i)}\}_{i=1}^N$ *summarizes* $p(x_0|y_0)$ *such that, for example,* $E(g(x_0)|y_0) \approx \sum_{i=1}^N w_0^{(i)} g(x_0^{(i)})$. *In particular, initialize* $(x_0^{(1)}, ..., x_0^{(N)})$ *from* $N(m_0, C_0)$ *and set* $w_0^{(i)} = N^{-1}\ \forall\ i = 1, ..., N$.
- *For* $t = 1, ..., n$*:*
  1. *For* $k = 1, ..., N$*:*
     (a) *Draw* I_k *with* $P(I_k) \propto w_{t-1}^{(i)} f(y_t|g(x_{t-1}^{(i)}))$

(b) *Draw* $x_t^{(k)} \sim N(x_{t-1}^{(I_k)}, \tau^2)$

(c) *Set* $\tilde{w}_t^{(k)} = \frac{f_N(y_t|x_t^{(k)})}{f_N(y_t|g(x_{t-1}^{(I_k)}))}$

2. *Normalize the weights:* $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N}(\tilde{w}_t^{(j)})}$

3. *Compute* $ESS = \left(\sum_{i=1}^{N} (w_t^{(i)})^2 \right)^{-1}$

4. *if* $ESS < N/2$ *then*

(a) *Draw a sample of size N,* $(x_t^{(1)}, ..., x_t^{(N)})$*, from the discrete distribution* $P(x_t = x_t^{(i)}) = w_t^{(i)}$, $i = 1, ..., N$

(b) *Reset the weights:* $w_t^{(i)} = N^{-1}$, $i = 1, ..., N$.

5. *Set* $p(x_t|y_t) = \sum_{i=1}^{N} w_t^{(i)} \delta_{x_t^{(i)}}$

The `APFfun` function resume this passages.

`APFfun(data,N,m0,C0,tau,sigma,r)`

Arguments

`data` the observed process. It has to be a vector or a univariate time series.
`N` number of particles generated at each step
`m0` central value of the normal prior state distribution
`C0` variance of the normal prior state distribution
`tau` the standard deviation τ in Equation (2)
`sigma` the standard deviation σ in Equation (1)
`r` if present the threshold is set equal to N/r otherwise, if missing, the threshold is set equal to $N/2$

```
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))
}
```

Let's compare the Auxiliary Particle Filter (APF) and the Bootstrap Particle Filter (BPF).

Table 3: Root Mean Square Errors

N	Threshold	BPF	APF
1000	0.50	0.885	0.875
1000	0.25	0.893	0.892
1000	0.10	0.855	0.870

Implementation
Consider the linear Gaussian example of section XX, but this time with unknown variances τ^2 and σ^2. Thus, let $\psi = (\sigma^2, \tau^2)$ be the unknown parameter vector and assign a gamma prior for its components,

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

Alternatively, assign them a uniform prior if we have no knowledge of the hyperparameters. The algorithm follows the following steps.

Algorithm 0.6 *LWF for Random Walk plus Noise Model*

- *Initialize* $(x_0^{(1)}, ..., x_0^{(N)})$ *from* $N(m_0, C_0)$, $(\sigma^{2^{(1)}}, ..., \sigma^{2^{(N)}})$ *from* $G(\alpha_v, \beta_v)$ *and* $(\tau^{2^{(1)}}, ..., \tau^{2^{(N)}})$ *from* $G(\alpha_w, \beta_w)$. *Set* $w_0^{(i)} = N^{-1} \ \forall \ i = 1, ..., N$. *Therefore* $\psi^{(i)} = (\sigma^{2^{(i)}}, \tau^{2^{(i)}})$, *and* $\hat{\pi}_0 = p(x_0|y_0) = \sum_{i=1}^N w_0^{(i)} \delta_{(x_0^{(i)}, \psi^{(i)})}$
- *For* $t = 1, ..., n$*:*
  1. *Compute* $\overline{\psi} = E_{\hat{\pi}_{t-1}}(\psi)$ *and* $\Sigma = V_{\hat{\pi}_{t-1}}(\psi)$. *For* $i = 1, ..., N$, *set*
  $$m^{(i)} = a\psi^{(i)} + (1-a)\hat{\psi}$$
  $$\hat{x}_t^{(i)} = E(x_t | x_{t-1} = x_{t-1}^{(i)}, \psi = \psi^{(k)})$$
  2. *For* $k = 1, ..., N$*:*
     - *Draw* I_k, *with* $P(I_k = i) \propto w_{t-1}^{(i)} f_N(y_t | g(x_{t-1}^{(i)}), \psi = m^{(i)})$
     - *Draw* $\sigma^{2^{(k)}}$ *from* $G(\alpha_v^{(I_k)}, \beta_v^{(I_k)})$
     - *Draw* $\tau^{2^{(k)}}$ *from* $G(\alpha_w^{(I_k)}, \beta_w^{(I_k)})$
     - *Draw* $x_t^{(k)}$ *from* $N(x_{t-1}^{(I_k)}, \psi = \psi^{(k)})$
     - *Set* $\tilde{w}_t^{(k)} = \frac{f_N(y_t | x_t^{(k)}, \psi = \psi^{(k)})}{f_N(y_t | g(x_{t-1}^{(I_k)}, \psi = m^{(I_k)}))}$
  3. *Compute* $ESS = \left(\sum_{i=1}^N (w_t^{(i)})^2 \right)^{-1}$

4. *if* $ESS < N/2$ *then*
   (a) *Draw a sample of size* N, $(x_t^{(1)}, ..., x_t^{(N)})$, *from the discrete distribution* $P(x_t = x_t^{(i)}) = w_t^{(i)}$, $i = 1, ..., N$
   (b) *Reset the weights:* $w_t^{(i)} = N^{-1}$, $i = 1, ..., N$.
5. *Set* $\hat{\pi}_t = p(x_t|y_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{(x_t^{(i)}, \psi^{(i)})}$

Our `LWfun` function goes through the illustrated steps.

`LWfun(data,N,m0,C0,alphav,betav,alphaw,betaw,delta,unif,r)`

Arguments

`data` the observed process. It has to be a vector or a univariate time series.
`N` number of particles generated at each step
`m0` central value of the normal prior state distribution
`C0` variance of the normal prior state distribution
`alphav, betav` Gamma prior hyperparameters on σ^2
`alphaw,betaw` Gamma prior hyperparameters on τ^2
`delta` hyperparameter delta value
`unif` if True then it sets a Uniform $(0, 10)$ prior on σ^2 and τ^2
`r` if present the threshold is set equal to N/r otherwise, if missing, the threshold is set equal to $N/2$

```
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[,2]+(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))
}
```

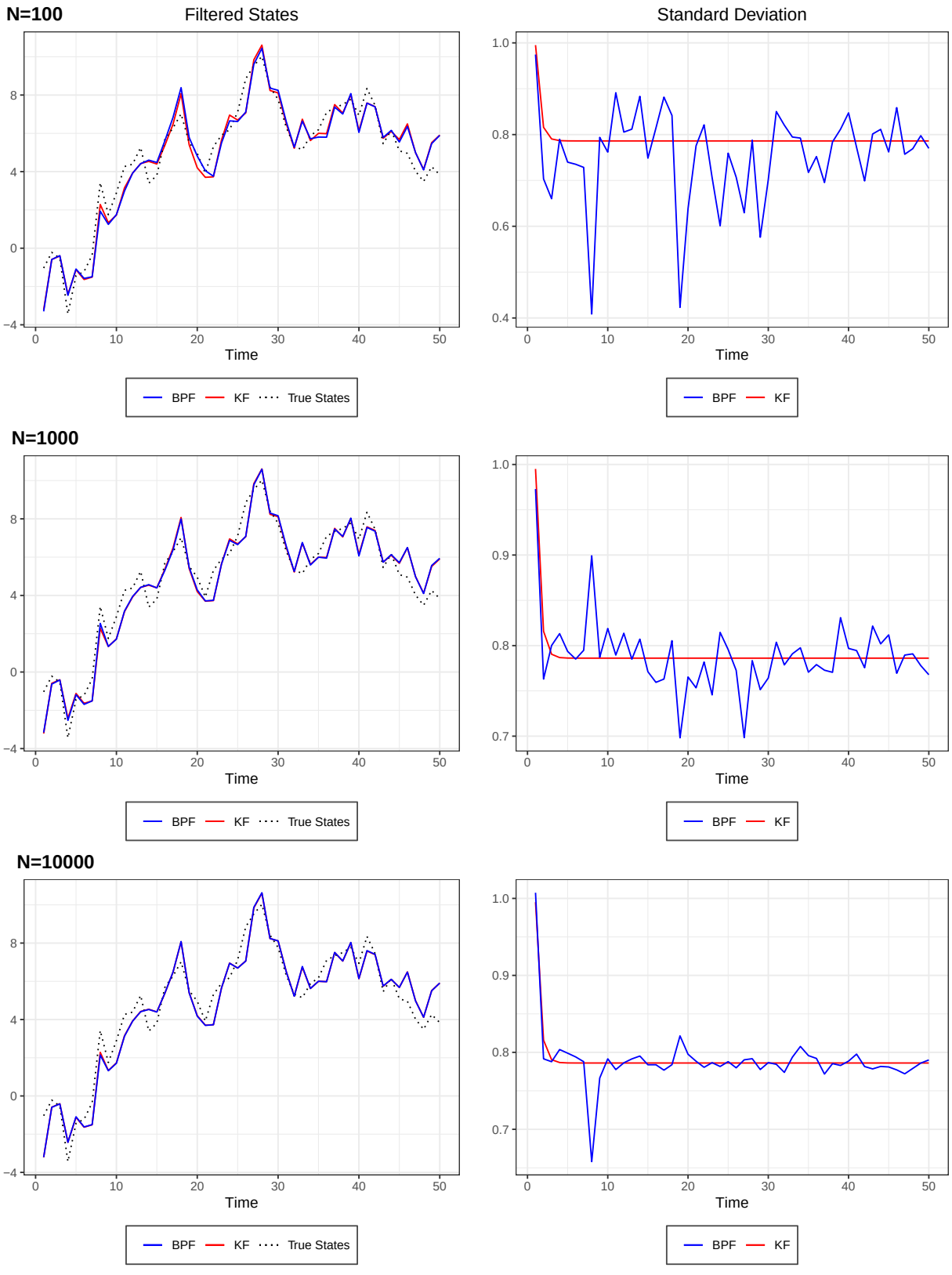

Figure 5: Comparison Bootstrap Particle Filter(BPF) and Kalman Filter (KF) for increasing number of generated particles (N)

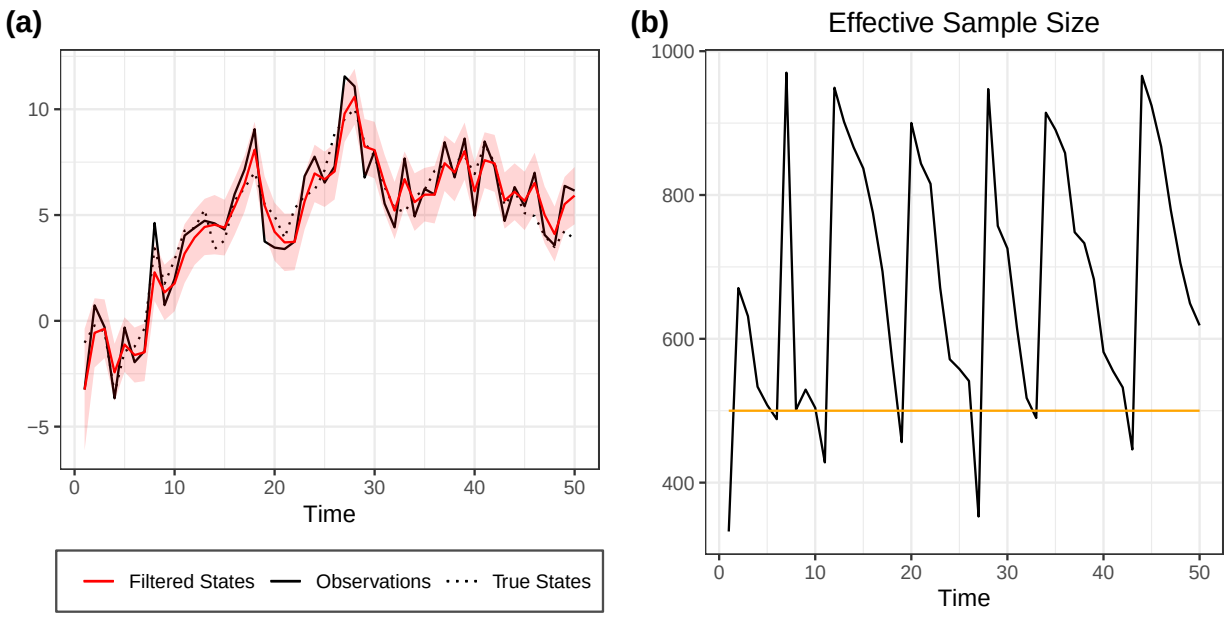

Figure 6: a) GPF Filtered States with credible interval (in red). b) Effective sample size (in black) with threshold (in yellow).

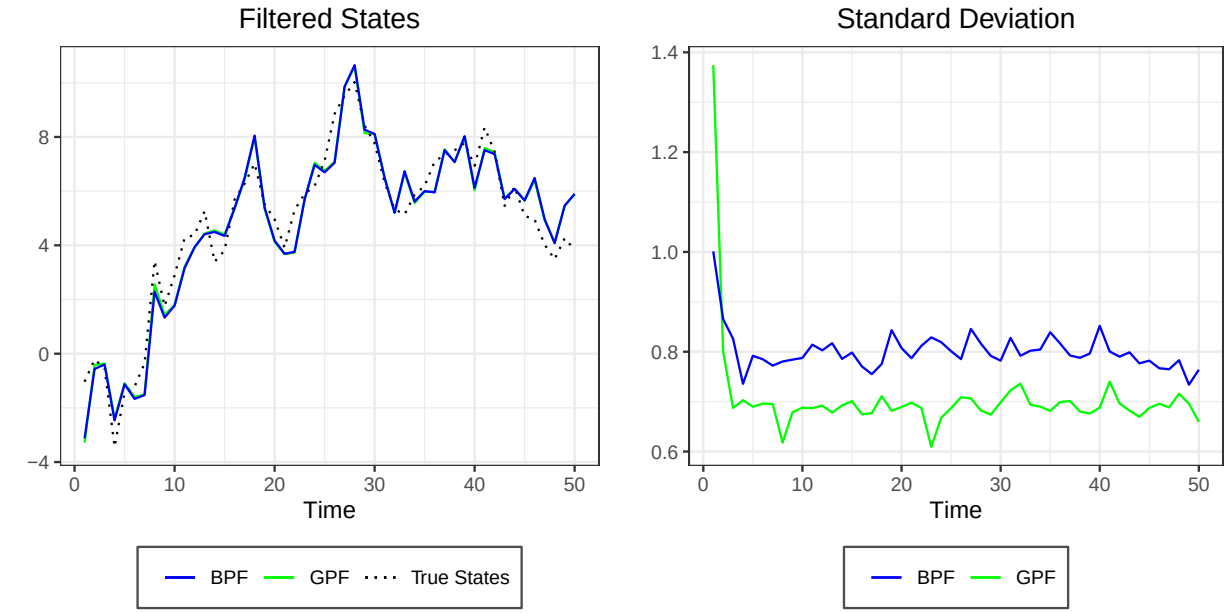

Figure 7: Comparison Bootstrap Particle Filter(BPF) and Guided Particle Filter (GPF), number of generated particles N=1000

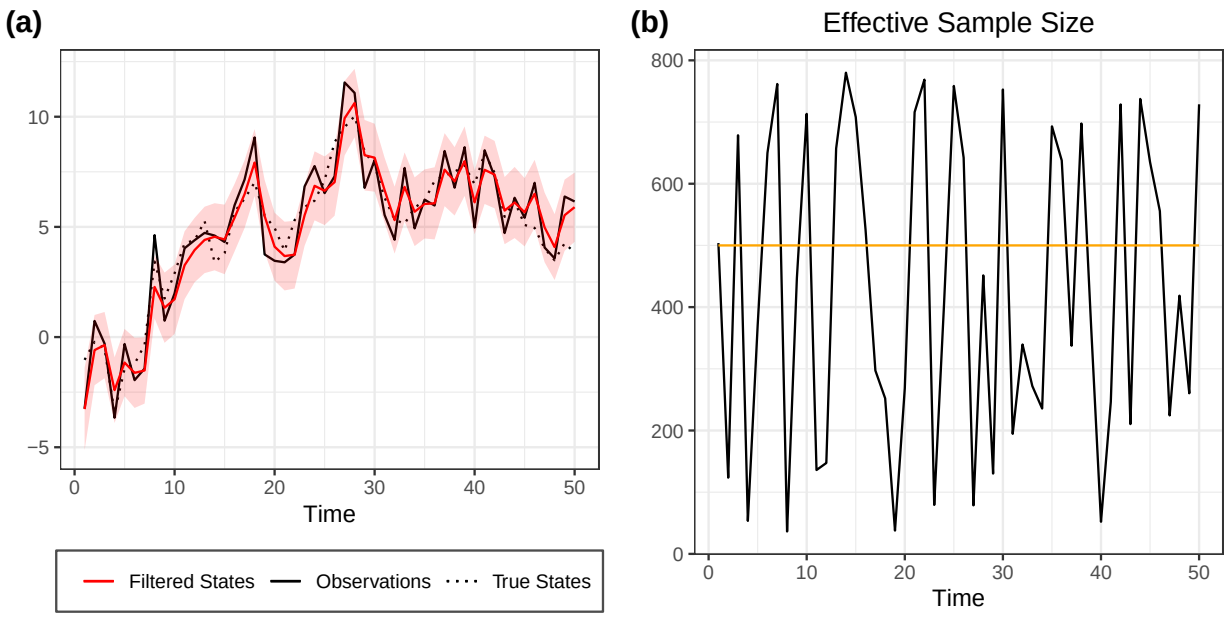

Figure 8: a) APF Filtered States with credible interval (in red). b) Effective sample size (in black) with threshold (in yellow).

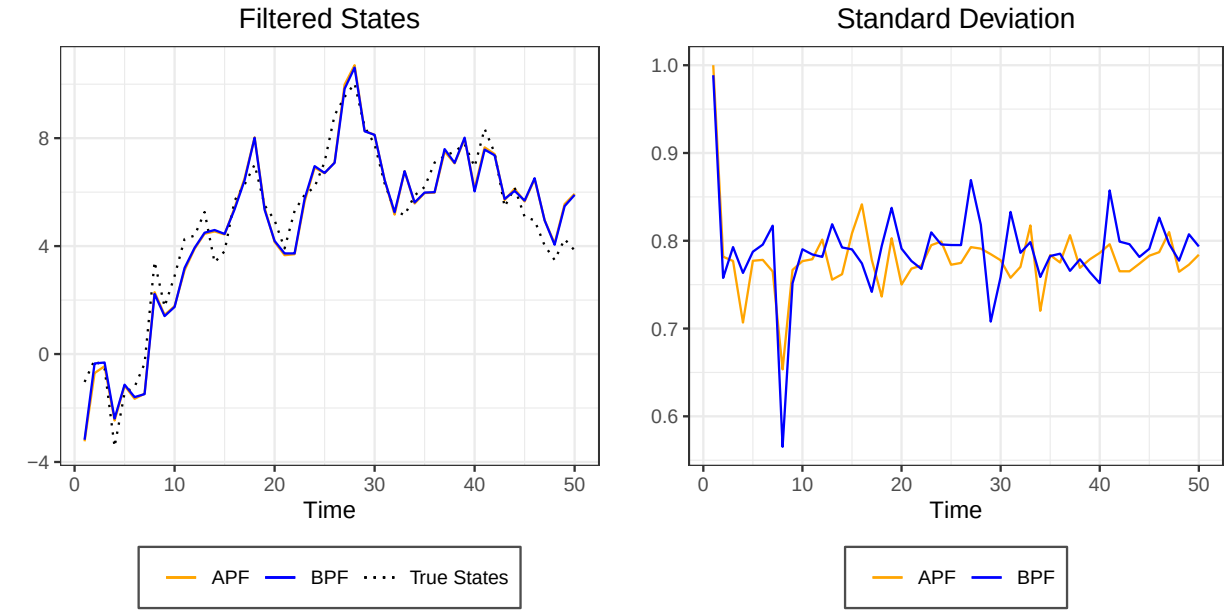

Figure 9: Comparison Bootstrap Particle Filter(BPF) and Guided Particle Filter (GPF), number of generated particles N=1000