UiO : Matematisk institutt
Det matematisk-naturvitenskapelige fakultet

STK-4051/9051 Computational Statistics Spring 2021
Sequential Monte Carlo

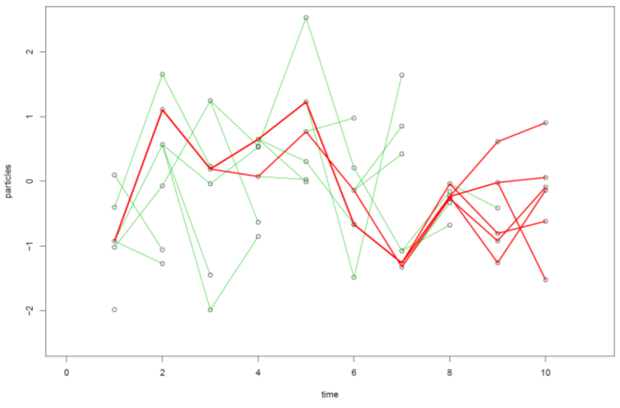
Instructor: Odd Kolbjørnsen, oddkol@math.uio.no

Simulation techniques

- Exact methods
  - Inversion/transformation methods
  - Rejection sampling
- Approximate methods
  - Sampling importance resampling
  - Sequential Monte Carlo
  - Markov chain Monte Carlo (Chapter 7 and 8)
- Variance reduction methods
  - Importance sampling
  - Antithetic sampling
  - Control variates
  - Rao-blackwellization
  - Common random numbers

Last time

- Want to sample from $f(x)$, but get sample from $g(x)$
  - The ratio: $f(x)/g(x)$ is important
- Rejection sampling
  - Bounding the ratio
- Importance sampling
  - Weighting with the ratio
  - Effective number of samples
- Sampling importance Resampling (SIR)
  - Resampling with the ratio
  - Proof: larger variance than importance sampling
- Sequential Monte Carlo
  - Weight decay
  - Resampling
    - Reduce variability at later time
    - Optimal resampling
  - Sample degeneracy
    - What are we able to approximate?



Properly weighted sample

- A weighted random pair (X, W) is properly weighted with respect to π if for any (square integrable) function h

$$E[Wh(X)] = c \cdot E_\pi[h(X)]$$

  for some constant c.

- A weighted random sample $\{(X^i, W^i), i = 1, ..., N\}$ is properly weighted with respect to π if each (X_i, W_i) are properly weighted.

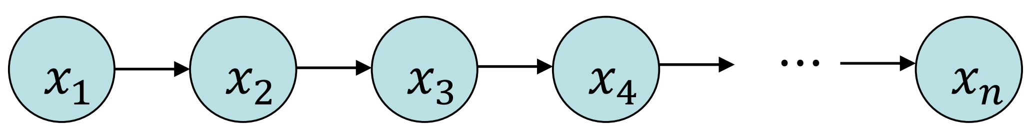
- Consequence: If $\{(X^i, W^i), i = 1, ..., N\}$ are properly weighted iid random pairs, then

$$\hat{\mu} = \frac{\sum_{i=1}^N W^i h(X^i)}{\sum_{i=1}^N W^i} \tag{1}$$

  is a consistent estimator of $\mu = E_\pi[h(X)]$ (with respect to increasing N).

Factoring into 1D distributions

- Target
  - $f(\boldsymbol{x}) = f(x_1)f(x_2|x_1)\cdots f(x_n|x_1, x_2, \dots, x_{n-1})$
- Simpler to sample from
  - $g(\boldsymbol{x}) = g(x_1)g(x_2|x_1)\cdots g(x_n|x_1, x_2, \dots, x_{n-1})$
- Sample each component sequentially as 1D
- Markov property: (same principle - simpler notation)
  - $f(\boldsymbol{x}) = f(x_1)f(x_2|x_1)f(x_3|x_2)\cdots f(x_n|x_{n-1})$
  - $g(\boldsymbol{x}) = g(x_1)g(x_2|x_1)g(x_3|x_2)\cdots g(x_n|x_{n-1})$



Sequential Monte Carlo in a Markov structure

- Sequential Monte Carlo: First high-dimensional setting
- Assume now $\mathbf{x} = \mathbf{x}_{1:t} = (x_1, \ldots, x_t)$ have a Markov structure

$$f_t(\mathbf{x}_{1:t}) = f_1(x_1)\prod_{i=2}^{t} f_i(x_i|x_{i-1})$$

- Also assume a proposal distribution with Markov property:

$$g_t(\mathbf{x}_{1:t}) = g_1(x_1)\prod_{i=2}^{t} g_i(x_i|x_{i-1})$$

- Importance weights:

$$w(\mathbf{x}_{1:t}) = \frac{f_t(\mathbf{x}_{1:t})}{g_t(\mathbf{x}_{1:t})} = \frac{f_1(x_1)}{g_1(x_1)}\prod_{i=2}^{t}\frac{f_i(x_i|x_{i-1})}{g_i(x_i|x_{i-1}))} = w(\mathbf{x}_{1:t-1})\frac{f_t(x_t|x_{t-1})}{g_t(x_t|x_{t-1}))}$$

- Opens up for sequential sampling/estimation
- Note: Easy to generalize to non-Markov settings as well
  - More computing at each step

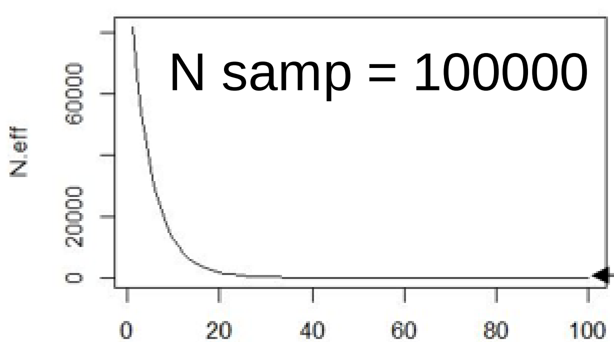
Sequential Monte Carlo

Algorithm

1. Sample $X_1 \sim g_1(\cdot)$. Let $w_1 = u_1 = f_1(x_1)/g_1(x_1)$. Set $t = 2$
2. Sample $X_t | x_{t-1} \sim g_t(x_t | x_{t-1})$.
3. Append x_t to $\mathbf{x}_{1:t-1}$, obtaining $\mathbf{x}_t$
4. Let $u_t = f_t(x_t|x_{t-1})/g_t(x_t|x_{t-1})$
5. Let $w_t = w_{t-1} u_t$, the importance weight for $\mathbf{x}_{1:t}$
6. Increment t and return to step 2

Can simulate m sequences in paralell!

Problem weight decay



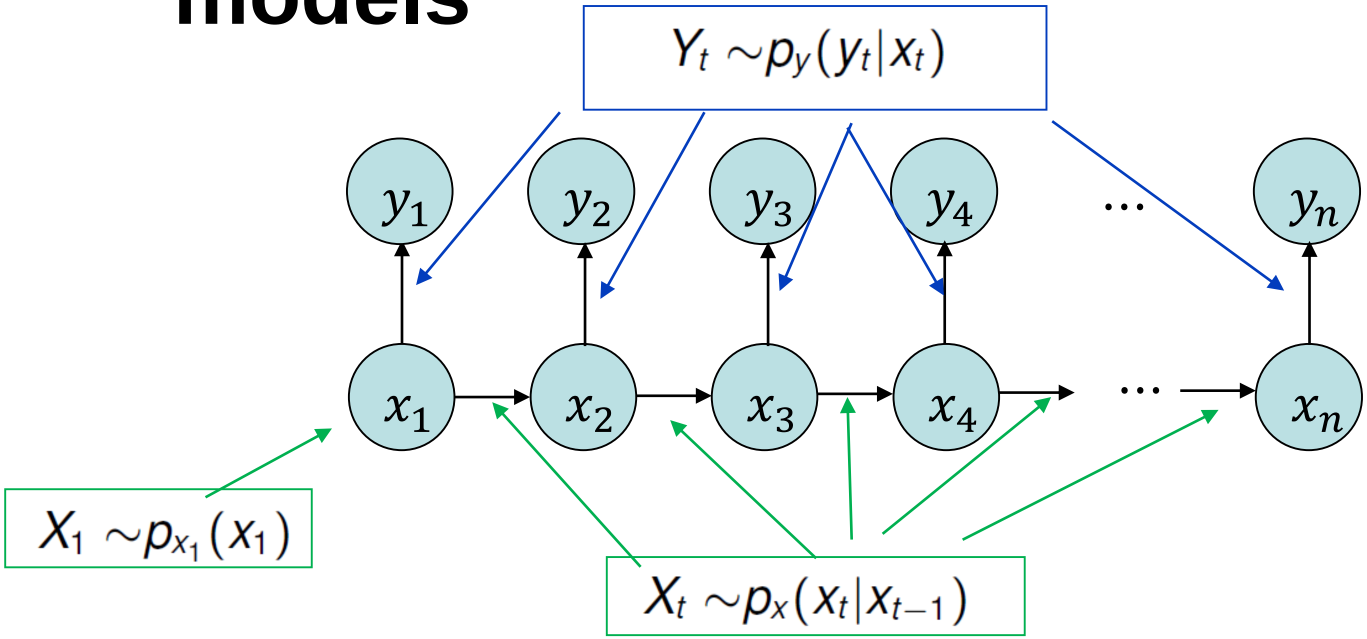
$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{n} w_i^2}$$

Neff: 3.78

Resampling to solve problem

- Degeneracy of weights a serious problem.
- Solution: Resampling (SIR idea)
- How:
  - Resample from $\{X_t^{(1)}, \ldots, X_t^{(n)}\}$ with normalized probabilities $w(X_t^{(1)}), \ldots, w(X_t^{(n)})$
  - Put all weights equal to 1
  - Either at each time step or when $\widehat{N}_{eff}$ is small
- Resampling will introduce extra random noise at the current time-point
- Can reduce noise at later time points
- Gives a good approximation to $f(x_n)$
- Does not give a good approximation to $f(\mathbf{x})$ or $f(x_1)$!

Origin of method in state space models

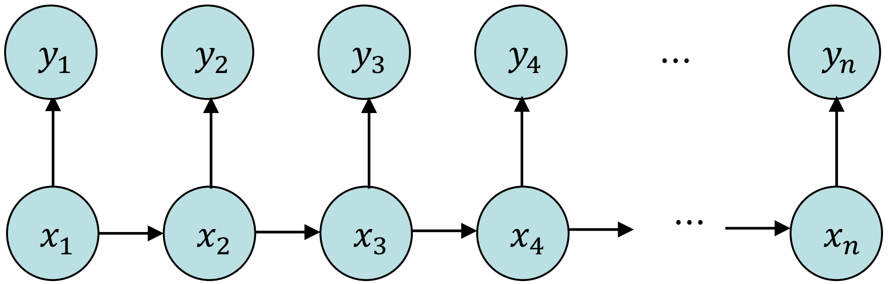


- For the filtering problem the target is the distribution $p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n})$
- This is good match for the resampling approach

Hidden Markov models - state space models

- Assume

$$X_1 \sim p_{x_1}(x_1)$$
$$X_t \sim p_x(x_t|x_{t-1})$$
$$Y_t \sim p_y(y_t|x_t)$$



- $\{y_t\}$ observed, $\{x_t\}$ hidden
- Chapter 4: $\{x_t\}$ discrete. Now possibly continuous
- Aim: $f(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$ or $f(\mathbf{x}_t|\mathbf{y}_{1:t})$
- Recursive relationship (misprint in book):

$$\begin{aligned} f(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) &= \frac{f(\mathbf{x}_{1:t}, y_t|\mathbf{y}_{1:t-1})}{f(y_t|\mathbf{y}_{1:t-1})} \\ &= \frac{f(\mathbf{x}_{1:t-1}|\mathbf{y}_{1:t-1})p_x(x_t|x_{t-1})p_y(y_t|x_t)}{f(y_t|\mathbf{y}_{1:t-1})} \\ &\propto f(\mathbf{x}_{1:t-1}|y_{1:t-1})p_x(x_t|x_{t-1})p_y(y_t|x_t) \end{aligned}$$

Hidden Markov model sequential Monte Carlo

$$f(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) \propto f(\mathbf{x}_{1:t-1}|y_{1:t-1})p_x(x_t|x_{t-1})p_y(y_t|x_t)$$

Sample next time step

Current sample

Update according to Markov dynamics

Weight adjustment

- Use : $g_t(x_t|x_{t-1}) = p_x(x_t|x_{t-1})$

$$\begin{aligned} w_t &= \frac{f(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})}{g(\mathbf{x}_{1:t})} \\ &\propto \frac{f(\mathbf{x}_{1:t-1}|y_{1:t-1})p_x(x_t|x_{t-1})p_y(y_t|x_t)}{p_{x_1}(x_1)\prod_{s=2}^{t} p_x(x_s|x_{s-1})} \\ &= \frac{f(\mathbf{x}_{1:t-1}|y_{1:t-1})}{g(\mathbf{x}_{1:t-1})}\frac{p_x(x_t|x_{t-1})p_y(y_t|x_t)}{p_x(x_t|x_{t-1})} \\ &= w_{t-1}p_y(y_t|x_t) \end{aligned}$$

Expand

«Summarize t-1»

«Recognize»

Algorithm SMC for HMM

First

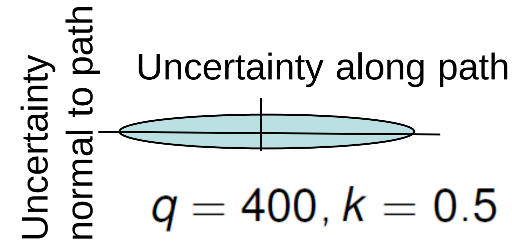
1. Sample $X_1^i \sim p_{X_1}(\cdot), i = 1, ..., n.$
2. Let $w_1^{*i} = u_1^i = p_y(y_1|x_1^i)$, normalize to $w_i^i = w_1^{*i} / \sum_j w_1^{*j}$.

Sequence

3. Sample $X_t^i|x_{t-1}^i \sim p_X(x_t|x_{t-1}^i), i = 1, ..., n.$
4. Append x_t^i to $\mathbf{x}_{1:t-1}^i$, obtaining $\mathbf{x}_t^i$
5. Let $u_t^i = p_y(y_t|x_t^i)$
6. Let $w_t^{*i} = w_{t-1}^i u_t^i$, normalize to $w_t^i = w_t^{*i} / \sum_j w_t^{*j}$.
7. If $\hat{N}_{eff}$ is small, perform resampling
8. Increment t and return to step 3

Terrain navigation

- Assume movement model for airplane



Model

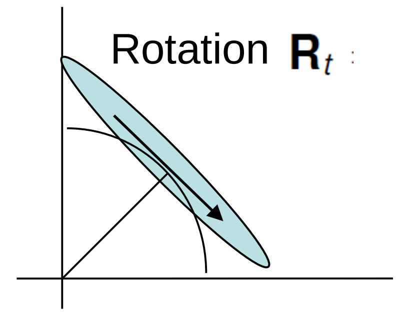
$$\mathbf{X}_t = \mathbf{x}_{t-1} + \mathbf{d}_t + \boldsymbol{\varepsilon}_t$$

$\mathbf{d}_t =$ Drift of plane measured by internal navigation system (assumed known)

$$\boldsymbol{\varepsilon}_t = \mathbf{R}_t^T \mathbf{Z}_t$$

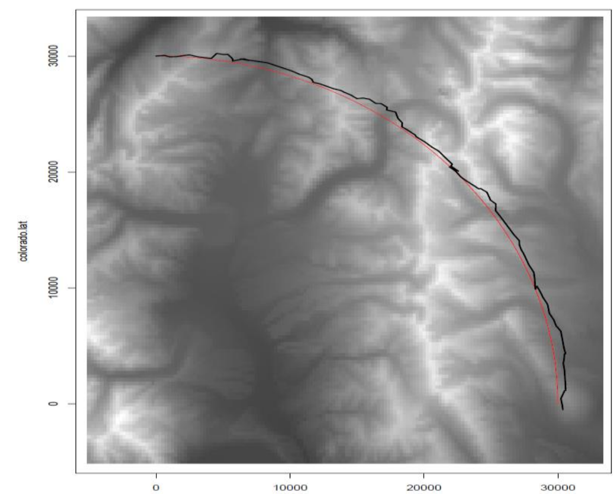
$$\mathbf{R}_t = \frac{1}{\sqrt{x_{1,t-1}^2 + x_{2,t-1}^2}} \begin{pmatrix} -x_{1,t-1} & x_{2,t-1} \\ -x_{2,t-1} & -x_{1,t-1} \end{pmatrix}$$

$$\mathbf{Z}_t \sim N_2\left(\mathbf{0}, q^2 \begin{pmatrix} 1 & 0 \\ 0 & k^2 \end{pmatrix}\right)$$



Data $\quad Y_t = m(\mathbf{x}_t) + \delta_t$

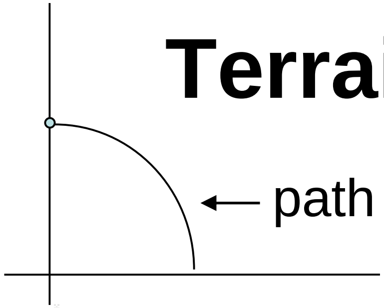
Map $\quad m(\mathbf{x}_t) =$ Elevation at point $\mathbf{x}_t$



- `Example_6_7.R`

Uncertainty $\boldsymbol{\varepsilon}_t, \delta_t$

Terrain navigation



← path (given)

```
# n = number of sampled trajectories
# Yt = observed elevation data
# mxti = map elevations
# xt.x, xt.y = current position of point
# uti = weight adjustment factors
# wti = weights
# Neff = effective sample size
# alpha = rejuvenation trigger
# dsubt.x, dsubt.y = true drift
# epst.x, epst.y = location error
```

```
##INITIAL VALUES
n=100
sigma=75
q.value=400
k=.5
sdx0=50
sqrtP0=sdx0     #My change
wti=rep(1/n,n)
```

Initial ensemble members equal weight

- Initialization
  - Path is constructed
  - Radius of curve 30 000

```
##SET UP TRUE STARTING POINT, 100 INITIAL POINTS
##AND FIRST ELEVATION OBSERVATION
x0hat.x=0 ; x0hat.y=30000  #start at true X0 here

truex=x0hat.x
truey=x0hat.y
xthat.x=truex
xthat.y=truey
xt.x=rnorm(n,x0hat.x,sqrtP0)    #was x.xold
xt.y=rnorm(n,x0hat.y,sqrtP0)    #was y.yold
```

Initial ensemble (n = 100)

$\mathbf{d}_t$

```
##SET UP TRUE DRIFT
route.theta=seq(0,pi/2,length=101)
route.x=30000*cos(route.theta)
route.y=30000*sin(route.theta)
route.x=rev(route.x)
route.y=rev(route.y)
dsubt.x=diff(route.x)
dsubt.y=diff(route.y)
```

Just one way to get it from the synthetic

Terrain navigation

```
# n = number of sampled trajectories
# Yt = observed elevation data
# mxti = map elevations
# xt.x, xt.y = current position of point
# uti = weight adjustment factors
# wti = weights
# Neff = effective sample size
# alpha = rejuvenation trigger
# dsubt.x, dsubt.y = true drift
# epst.x, epst.y = location error
```

Mimic observation process

```
#update truth and observed elevation data
  truex=truex+dsubt.x[i]
  truey=truey+dsubt.y[i]
  Yt=interp(colo.lon.rep,colo.lat.rep,
       colo.elev.interp,xo=truex,yo=truey)$z+rnorm(1,0,sigma)
```

Update weights from data

```
#find m(x_t^i)
  xord=rank(xt.x)
  yord=rank(xt.y)
  mapt=interp(colo.lon.rep,colo.lat.rep,colo.elev.interp,
     xo=sort(xt.x),yo=sort(xt.y))
  mxti=mapt$z[cbind(xord,yord)]
  extrap=is.na(mxti)
#weight the points
  uti=ifelse(extrap,0,dnorm(Yt,c(mxti),rep(sigma,n)))
  wti=uti*wti
  wti=wti/sum(wti)
  Neff=1/sum(wti^2)
  neff.record[i]=Neff
```

Current estimate

```
#preliminary calcs for drawing the plot
  xthat.old=c(xthat.x,xthat.y)
  xthat.x=sum(wti*xt.x)
  xthat.y=sum(wti*xt.y)
```

$$\mathbf{X}_t = \mathbf{x}_{t-1} + \mathbf{d}_t + \boldsymbol{\varepsilon}_t$$

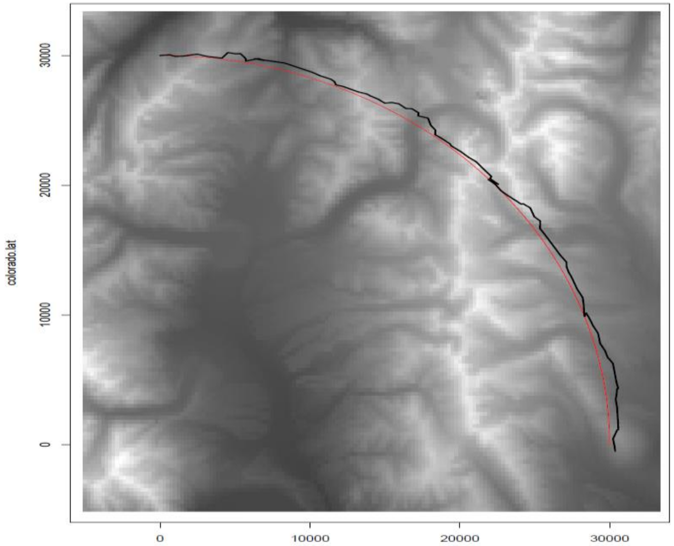
```
#update cloud
  tangent.slope=-truex/truey
  Zsigmamat=cbind(c(q.value^2,0),1*c(0,(k*q.value)^2))
  xtnext=rotnorm(n,tangent.slope,Zsigmamat)
  epst.x=xtnext[,1]
  epst.y=xtnext[,2]

rotnorm=function(N,slope,sigmamat) {
    v=rmvnorm(N,mean=c(0,0),sigma=sigmamat)
    xy=c(1,slope)
    Rot=cbind(c(-xy[1],-xy[2]),c(xy[2],-xy[1]))/sqrt(sum(xy^2))
    therot=t(Rot%*%t(v))
    therot }
```

Terrain navigation

```
# n = number of sampled trajectories
# Yt = observed elevation data
# mxti = map elevations
# xt.x, xt.y = current position of point
# uti = weight adjustment factors
# wti = weights
# Neff = effective sample size
# alpha = rejuvenation trigger
# dsubt.x, dsubt.y = true drift
# epst.x, epst.y = location error
```

Resample?

```
if (Neff<(alpha*n)) {
   idx=sample(1:n,n,replace=T,prob=wti)
   xtnew.x=xt.x[idx]
   xtnew.y=xt.y[idx]
   wti=rep(1/n,n)
   rejuvcount=rejuvcount+1
   reset=T }
```



Update according to dynamic model

```
if (!reset) {
   xtnext.x=xt.x+dsubt.x[i]+epst.x      #still using old points
   xtnext.y=xt.y+dsubt.y[i]+epst.y
} else {
   xtnext.x=xtnew.x+dsubt.x[i]+epst.x   #start with new points
   xtnext.y=xtnew.y+dsubt.y[i]+epst.y
   reset=F
}
xt.x=xtnext.x
xt.y=xtnext.y
```

SMC/ Particle filter / Bootstrap filter

- SMC with resampling usually called particle filters
- Some mix/confusion about terminology, mainly the same!
- Bootstrap filter: SMC for hidden Markov models with $g(x_t|x_{t-1}) = p_x(x_t|x_{t-1})$

Bootstrap filter

Algorithm 1 SMC

Simulate $x_1^i \sim p(x_1)$ for $i = 1, \ldots, N$. ▷ Initialization
Put weights $\boxed{w_1^i = p(y_1|x_1^i).}$
if $\hat{N}_{eff}$ is small **then** ▷ Resampling
    Resample x_1^i with probabilities proportional to w_1^i.
    Put $w_1^i = 1/N$.
end if

---

for $t = 2, 3, \ldots$ **do** ▷ Sequential Monte Carlo
    Simulate $\boxed{x_t^i \sim p(x_t|x_{t-1}^i)}$ for $i = 1, \ldots, N$.
    Put weights $\boxed{w_t^i = w_{t-1}^i p(y_t|x_t^i)}$.
    if $\hat{N}_{eff}$ is small **then** ▷ Resampling
        Resample $\mathbf{x}_{1:t}^i$ with probabilities proportional to w_t^i.
        Put $w_t^i = 1/N$.
    end if
end for

General algorithm

Algorithm 2 SMC

Simulate $x_1^i \sim q(x_1)$ for $i = 1, \ldots, N$. ▷ Initialization
Put weights $\boxed{w_1^i = p(x_1)p(y_1|x_1^i)/q(x_1^i).}$
if $\hat{N}_{eff}$ is small **then** ▷ Resampling
  Resample x_1^i with probabilities proportional to w_1^i.
  Put $w_1^i = 1/N$.
end if

---

for $t = 2, 3, \ldots$ **do** ▷ Sequential Monte Carlo
  Simulate $\boxed{x_t^i \sim q(x_t|x_{t-1}^i, y_t)}$ for $i = 1, \ldots, N$.
  Put weights $\boxed{w_t^i = w_{t-1}^i p(x_t^i|x_{t-1}^i)p(y_t|x_t^i)/q(x_t^i|x_{t-1}^i, y_t).}$
  if $\hat{N}_{eff}$ is small **then** ▷ Resampling
    Resample $\mathbf{x}_{1:t}^i$ with probabilities proportional to w_t^i.
    Put $w_t^i = 1/N$.
  end if
end for

Could also have: $q(x_1^i|y_1)$

Challenge: Find $q(x_t|x_{t-1}, y_t)$ a good approximation to $p(x_t|x_{t-1})p(y_t|x_t)$

SMC and parameter estimation

- Assume

$$X_1 \sim p(x_1;\theta)$$
$$X_t \sim p(x_t|x_{t-1};\theta)$$
$$Y_t \sim p(y_t|x_t;\theta)$$

Hidden Markov model
With unknown parameters
(we saw this type of model
in lecture 4, EM for HMM)

- Aim now: Simultaneous inference on θ and x_t (or $\mathbf{x}_{1:t}$)
- Two main approaches:
  - Maximum likelihood
  - Bayesian approach

SMC and maximum likelihood

- Interested in maximizing

$$L_t(\theta) = p(\mathbf{y}_{1:t}|\theta) = \int_{\mathbf{x}_{1:t}} p(\mathbf{y}_{1:t}|\mathbf{x}_{1:t};\theta)p(\mathbf{x}_{1:t}|\theta)d\mathbf{x}_{1:t}.$$

- Main problem: Calculation of the likelihood function (and possibly the score function in order to do optimization)
- Main approach: Use that

$$p(\mathbf{y}_{1:t}|\theta) = p(y_1|\theta)\prod_{s=2}^{t} p(y_s|\mathbf{y}_{1:s-1};\theta)$$

  and

$$\begin{aligned} p(y_s|\mathbf{y}_{1:s-1}) &= \int_{x_s} p(x_s|\mathbf{y}_{1:s-1})p(y_s|x_s;\theta)dx_s \\ &\approx \sum_{i=1}^{N} w_{t-1}^i p(y_s|x_s^i;\theta) \end{aligned}$$

  where $x_s^i \sim p(x_s|x_{s-1}^i)$ (Bootstrap filter).

- Poyiadjis et al. (2011): Algorithms for calculating the score function and information (matrix) recursively

SMC and Bayesian parameter estimation

- Assume

$$
\begin{aligned}
X_1 &\sim p(x_1;\theta)\\
X_t &\sim p(x_t|x_{t-1};\theta)\\
Y_t &\sim p(y_t|x_t;\theta)\\
\theta &\sim p(\theta)
\end{aligned}
$$

- Aim now: Simulate from $p(x_t, \theta|\mathbf{y}_{1:t})$
- Three approaches
  - Direct use of SMC
  - Introducing dynamics in θ
  - Using sufficient statistics

Direct use of SMC

- Assume at time $t-1$ the existence of a properly weighted sample $\{(x_{t-1}^i, \theta^i, w_{t-1}^i)\}$ with respect to $p(x_{t-1}, \theta | \mathbf{y}_{1:t-1})$.
- We have

$$\begin{aligned} p(x_t, \theta | \mathbf{y}_{1:t-1}) &= \int_{x_{t-1}} p(x_t | x_{t-1}, \theta) p(x_{t-1}, \theta | \mathbf{y}_{1:t-1}) dx_{t-1} \\ &\approx \sum_{i=1}^N w_{t-1}^i p(x_t | x_{t-1}^i, \theta^i) \delta_\theta(\theta^i) \end{aligned}$$

  and

$$p(x_t, \theta | \mathbf{y}_{1:t}) \approx c \cdot \sum_{i=1}^N w_{t-1}^i p(x_t | x_{t-1}^i, \theta^i) \delta_\theta(\theta^i) p(y_t | x_t, \theta^i)$$

- Updated samples $\{(\theta^i, x_t^i, w_t^i)\}$:
  1. Simulate $x_t^i \sim p(x_t | x_{t-1}^i, \theta^i)$
  2. Update the weights by $w_t^i \propto w_{t-1}^i p(y_t | x_t^i, \theta^i)$
- The sample $\{\theta^i\}$ do not change over time.
- With resampling, this will lead to degeneracy

Properly weighted sample

- A weighted random pair (X, W) is properly weighted with respect to π if for any (square integrable) function h

$$E[Wh(X)] = c \cdot E_\pi[h(X)]$$

  for some constant c.

- A weighted random sample $\{(X^i, W^i), i = 1, ..., N\}$ is properly weighted with respect to π if each (X_i, W_i) are properly weighted.

- Consequence: If $\{(X^i, W^i), i = 1, ..., N\}$ are properly weighted iid random pairs, then

$$\hat{\mu} = \frac{\sum_{i=1}^N W^i h(X^i)}{\sum_{i=1}^N W^i} \tag{1}$$

  is a consistent estimator of $\mu = E_\pi[h(X)]$ (with respect to increasing N).

Is direct use of SMC properly weighted?

- Proposal:

$$\theta^i \sim g(\theta) \qquad x_s^i \sim p(x_s|x_{s-1}^i, \theta^i), \quad s = 1, ..., t$$

- Weights at time $t = 1$:

$$w_1^i = \frac{p(\theta^i)p(x_1^i|\theta^i)p(y_1|x_1^i, \theta^i)}{g(\theta^i)p(x_1^i|\theta^i)} = \frac{p(\theta^i)p(y_1|x_1^i, \theta^i)}{g(\theta^i)}$$

giving properly weighted samples at time 1.

- At time t:

$$\begin{aligned} w_t^i =& \frac{p(\theta^i)p(x_1^i|\theta^i)p(y_1|x_1^i, \theta^i)\prod_{s=2}^t p(x_s^i|x_{s-1}^i, \theta^i)p(y_s|x_s^i, \theta^i)}{g(\theta^i)p(x_1^i|\theta^i)\ \prod_{s=2}^t p(x_s^i|x_{s-1}^i, \theta^i)} \\ =& \frac{p(\theta^i)p(x_1^i|\theta^i)p(y_1|x_1^i, \theta^i)\prod_{s=2}^t p(y_s|x_s^i, \theta^i)}{g(\theta^i)p(x_1^i|\theta^i)} \\ =& \frac{p(\theta^i)p(x_1^i|\theta^i)p(y_1|x_1^i, \theta^i)\prod_{s=2}^{t-1} p(y_s|x_s^i, \theta^i)}{g(\theta^i)p(x_1^i|\theta^i)} p(y_t|x_t{}^i, \theta^i) \\ =& w_{t-1}^i p(y_t|x_t{}^i, \theta^i) \end{aligned}$$

- Main problem: Now we need to resample $(\theta, \boldsymbol{x}_{1:t})$.
  Will result in degeneracy when $p(\theta, x_t|\boldsymbol{y}_{1:t})$ is of interest.

Lemmings data

- Interested in the dynamics of the lemmings populations
- From church books: Binary records on lemmings years or not.
- Define $x_t = \log(N_t)$, N_t population size at year t
- Model

$$
\begin{aligned}
x_t =& a x_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim N(0, \sigma^2) \\
y_t \sim& \text{Binom}\left(1, \frac{\exp(x_t)}{1+\exp(x_t)}\right)
\end{aligned}
$$

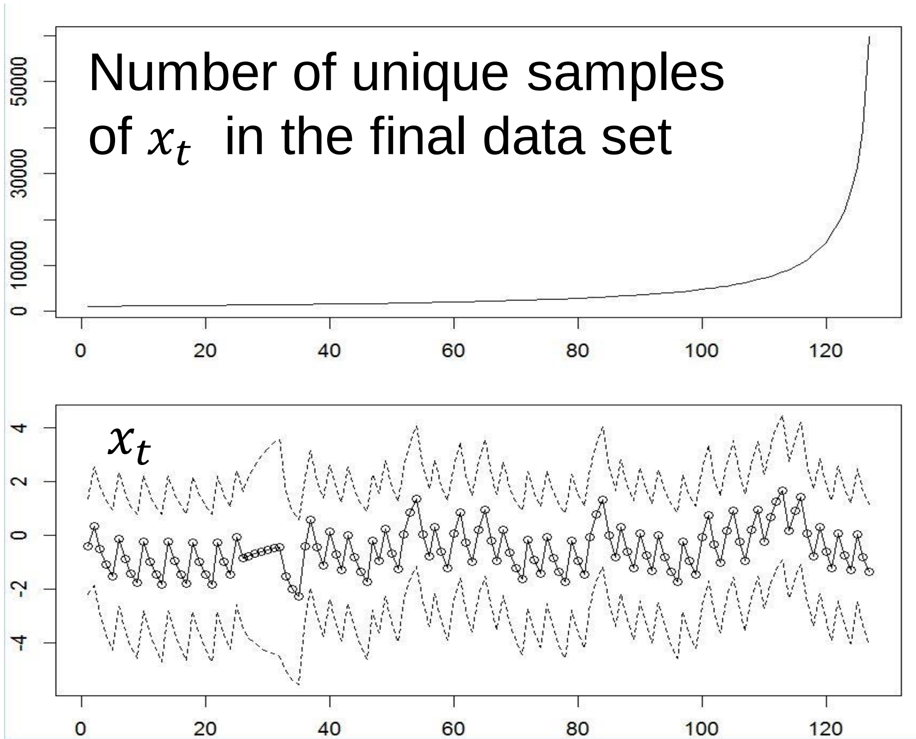
- Of interest: $p(x_t|\boldsymbol{y}_{1:t})$, $p(a|\boldsymbol{y}_{1:t})$
- `SMC_lin_bin.R`, `SMC_lin_bin_fixed.R`
- `SMC_lin_bin_parest_direct.R`

```
#SMC with parameter estimation
#Estimating parameters simultaneously using direct method
sig2=sig^2
sig2.a = 2
#Initialization
x.sim[1,]=rnorm(N,0,sig)
w = dbinom(y[1],1,exp(x.sim[1,])/(1+exp(x.sim[1,])))
#Resample
ind = sample(1:N,N,replace=T,prob=w)
x.sim[1,] = x.sim[1,ind]
w = rep(1/N,N)
x.hat[1,1] = mean(x.sim[1,])
x.hat[1,2:3] = quantile(x.sim[1,],c(0.025,0.975))
a.sim = rnorm(N,0,sqrt(sig2.a))
a.hat = matrix(nrow=nT,ncol=3)
a.hat[1,1] = mean(a.sim)
a.hat[1,2:3] = quantile(a.sim,c(0.025,0.975))
```

```
N.unique = rep(NA,nT)
for(i in 1:nT)
  N.unique[i] = length(unique(x.sim[i,]))
```

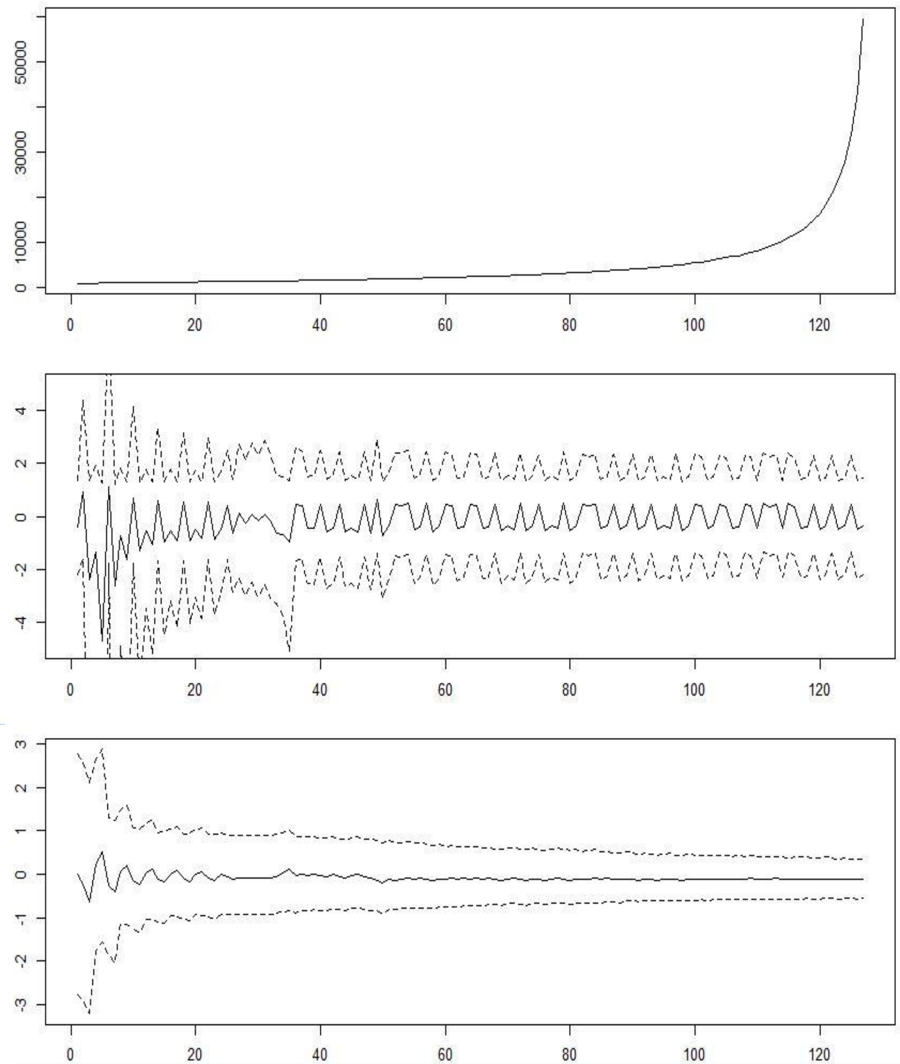
```
for(i in 2:nT)
{
  x.sim[i,]=rnorm(N,a.sim*x.sim[i-1,],sig)
  if(!is.na(y[i]))
    w = w*dbinom(y[i],1,exp(x.sim[i,])/(1+exp(x.sim[i,])))
  #Resample
  ind = sample(1:N,N,replace=T,prob=w)
  x.sim[1:i,] = x.sim[1:i,ind]            #Note: Resampling the whole path!
  a.sim = a.sim[ind]
  w = rep(1/N,N)
  x.hat[i,1] = mean(x.sim[i,])
  x.hat[i,2:3] = quantile(x.sim[i,],c(0.025,0.975))
  a.hat[i,1] = mean(a.sim)
  a.hat[i,2:3] = quantile(a.sim,c(0.025,0.975))
}
```

SMC_lin_bin_fixed.R



a=0.9

- SMC_lin_bin_parest_direct.R



Introducing dynamics in θ

- Liu and West (2001): Assume θ is (slowly) changing with time:

$$\theta_t = \theta_{t-1} + \zeta_t, \quad \zeta_t \sim N(0, q)$$

- Focus on $p(x_t, \theta_t|\mathbf{y}_{1:t})$.
- Assume a weighted sample $\{(x^i_{t-1}, \theta^i_{t-1}, w^i_{t-1})\}$

$$\begin{aligned}
p(x_t, \theta_t|\mathbf{y}_{1:t-1}) &= \int_{x_{t-1}} p(x_t|x_{t-1}, \theta_t)p(\theta_t|\theta_{t-1})p(x_{t-1}, \theta_{t-1}|\mathbf{y}_{1:t-1})dx_{t-1}d\theta_{t-1} \\
&\approx \sum_{i=1}^N w^i_{t-1}p(x_t|x^i_{t-1}, \theta_t)p(\theta_t|\theta^i_{t-1})
\end{aligned}$$

$$p(x_t, \theta_t|\mathbf{y}_{1:t}) \approx c \cdot \sum_{i=1}^N w^i_{t-1}p(x_t|x^i_{t-1}, \theta_t)p(\theta_t|\theta^i_{t-1})p(y_t|x_t, \theta_t).$$

- Update samples to $\{(\theta^i_t, x^i_t, w^i_t)\}$ by
  1. Simulate $\theta^i_t \sim p(\theta_t|\theta^i_{t-1})$,
  2. Simulate $x^i_t \sim p(x_t|x^i_{t-1}, \theta^i_t)$
  3. Update the weights by $w^i_t \propto w^i_{t-1}p(y_t|x^i_t, \theta^i_t)$.

«θ is like x»

Dynamics in θ continued

- New values $\{\theta_t^i\}$ are generated at each time point
- Main problem: Introduce extra variability in θ_t.
- Consequence: Estimation of θ_t mainly based on most recent observations
- The model

  $$\theta_t = \theta_{t-1} + \zeta_t, \quad \zeta_t \sim N(0, q)$$

  might be reasonable
- New problem: Estimate the static parameter q.
- `SMC_lin_bin_parest_dyn.R`

Sufficient statistics

- Example:

$$x_t = ax_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim N(0, \sigma^2), \quad \sigma \text{ known for simplicity}$$

- The distribution $p(y_t|x_t)$ can be arbitrary (but not depending on θ).
- $\theta = a$ needs to be estimated. Assume a prior $a \sim N(\mu_a, \sigma_a^2)$.
- Can be shown:

$$p(a|\mathbf{x}_{1:t}) = N(\mu_{a|t}, \sigma_{a|t}^2)$$

where

$$\mu_{a|t} = \frac{\sigma_a^2 \sum_{s=2}^t x_s x_{s-1} + \sigma^2 \mu_a}{\sigma_a^2 \sum_{s=2}^t x_{s-1}^2 + \sigma^2}; \quad \sigma_{a|t}^2 = \frac{\sigma^2 \sigma_a^2}{\sigma_a^2 \sum_{s=2}^t x_{s-1}^2 + \sigma^2}.$$

- Main point: Given $\mathbf{x}_{1:t}$, the distribution of a (and simulation) is simple.
- $p(a|\mathbf{x}_{1:t})$ only depend on $S_{t,1} = \sum_{s=2}^t x_s x_{s-1}$ and $S_{t,2} = \sum_{s=2}^t x_{s-1}^2$
- Both terms can be recursively updated through

$$S_{t,1} = S_{t-1,1} + x_t x_{t-1}, \quad S_{t,2} = S_{t-1,2} + x_{t-1}^2.$$

SMC and sufficient statistics

- Assume $p(y_t|x_t)$ do not depend on θ.
- Assume $p(\theta|\mathbf{x}_{1:t}) = p(\theta|S_t)$, S_t sufficient statistic.
- Assume $S_t = h(S_{t-1}, x_{t-1}, x_t)$
- Fearnhead (2002) and Storvik (2002): Focus on $p(x_t, S_t|\mathbf{y}_{1:t})$, not $p(x_t, \theta|\mathbf{y}_{1:t})$.
- Assume a properly weighted sample $\{(x_{t-1}^i, S_{t-1}^i, w_{t-1}^i), i = 1, \ldots, N\}$ with respect to $p(x_{t-1}, S_{t-1}|\mathbf{y}_{1:t-1})$
- Similar recursions as before:

$$
\begin{aligned}
p(x_t, S_t|\mathbf{y}_{1:t-1}) =& \int_{x_{t-1}} p(x_t, S_t|x_{t-1}, S_{t-1})p(x_{t-1}, S_{t-1}|\mathbf{y}_{1:t-1})dx_{t-1}dS_{t-1} \\
\approx& \sum_{i=1}^N w_{t-1}^i p(x_t, S_t|x_{t-1}^i, S_{t-1}^i) \\
p(x_t, S_t|\mathbf{y}_{1:t}) \approx& c \cdot \sum_{i=1}^N w_{t-1}^i p(x_t, S_t|x_{t-1}^i, S_{t-1}^i)p(y_t|x_t).
\end{aligned}
$$

- Simulation from $p(x_t, S_t|x_{t-1}^i, S_{t-1}^i)$ (possible proposal function)
  1. Simulate $\theta^i \sim p(\theta|x_{t-1}^i, S_{t-1}^i) = p(\theta|S_{t-1}^i)$.
  2. Simulate $x_t^i \sim p(x_t|x_{t-1}^i, \theta^i)$.
  3. Put $S_t^i = h(S_{t-1}^i, x_{t-1}^i, x_t^i)$.

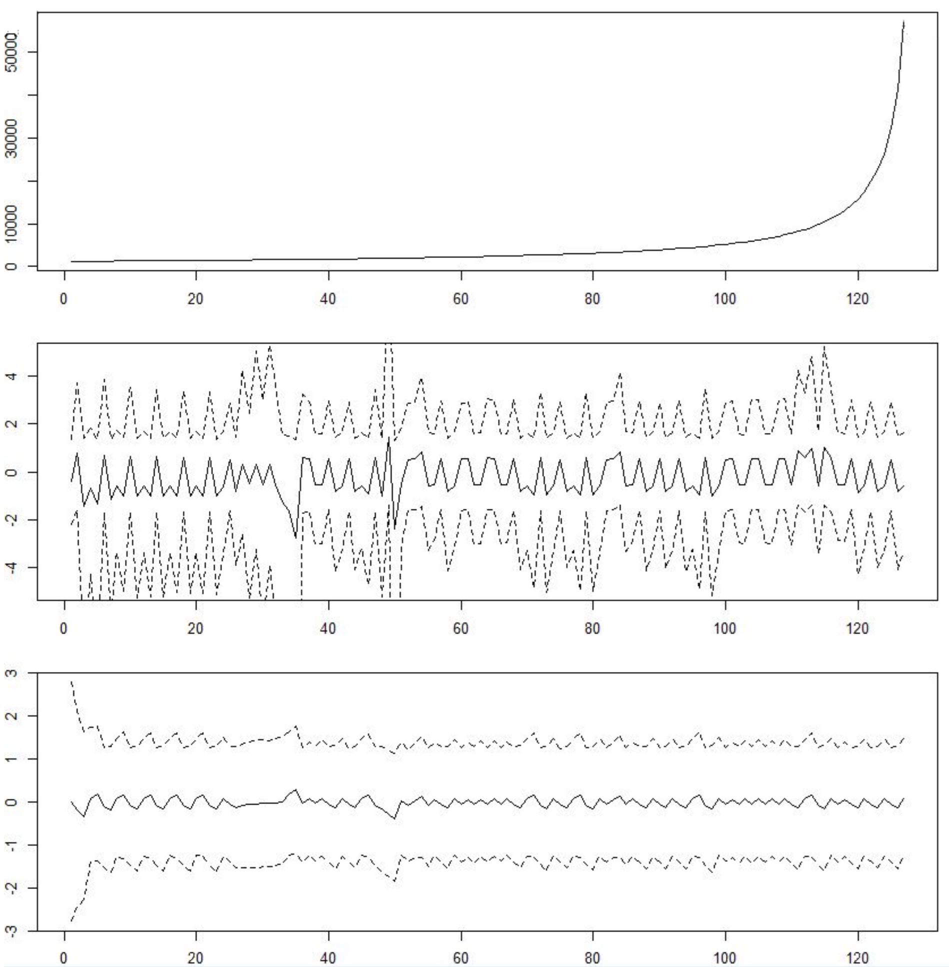
Algorithm Storvik filter

Algorithm 3 SMC with parameter updating

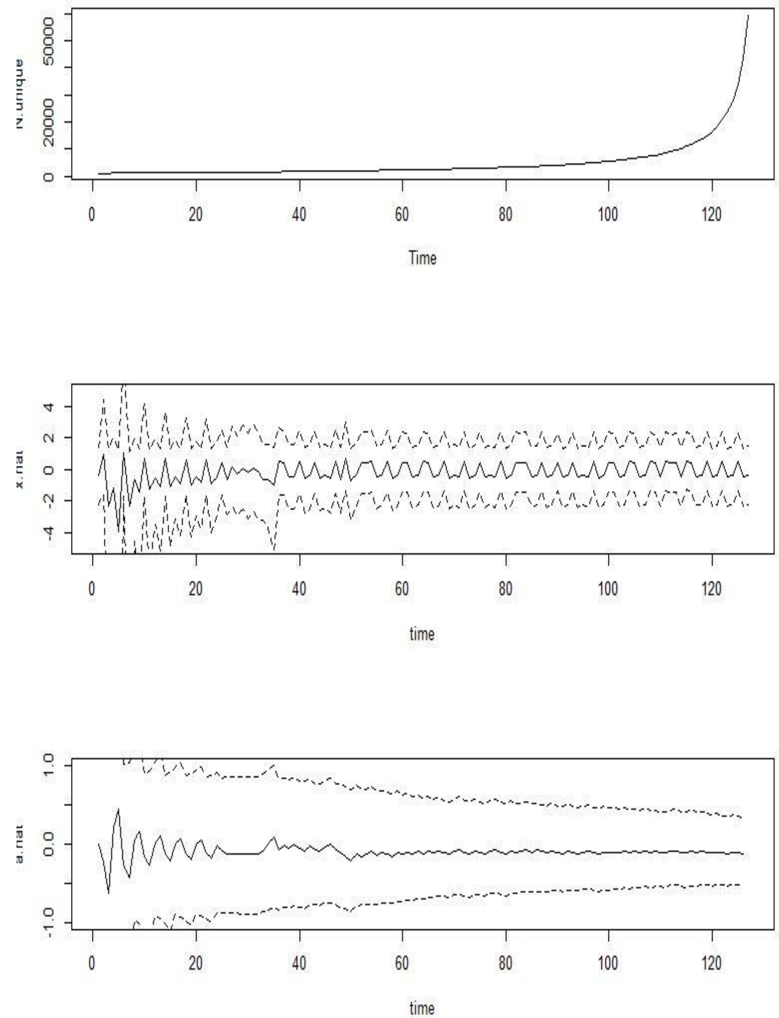
Simulate $\theta^i \sim p(\theta)$ for $i = 1, \ldots, N$. ▷ Initialization
Simulate $x_1^i \sim p(x_1|\theta^i)$ for $i = 1, \ldots, N$.
Put weights $w_1^i = p(y_1|x_1^i)$.
Put $S_1^i = 0$ for $i = 1, \ldots, N$.
for $t = 2, 3, \ldots$ **do** ▷ Sequential Monte Carlo
    Simulate $\theta^i \sim p(\theta|S_{t-1}^i)$ for $i = 1, \ldots, N$.
    Simulate $x_t^i \sim p(x_t|x_{t-1}^i, \theta^i)$ for $i = 1, \ldots, N$.
    Put weights $w_t^i = w_{t-1}^i p(y_t|x_t^i)$.
    Put $S_t^i = h(S_{t-1}^i, x_{t-1}^i, x_t^i)$.
    if $\hat{N}_{eff}$ is small **then** ▷ Resampling
        Resample (x_t^i, S_t^i) with probabilities proportional to w_t^i.
        Put $w_t^i = 1/N$.
    end if
end for

`SMC_lin_bin_parest_suff.R`

- SMC_lin_bin_parest_dyn.R



SMC_lin_bin_parest_suff.R



References

P. Fearnhead. Markov chain Monte Carlo, sufficient statistics, and particle filters. *Journal of Computational and Graphical Statistics*, 11(4):848–862, 2002.

J. Liu and M. West. Combined parameter and state estimation in simulation-based filtering. In *Sequential Monte Carlo methods in practice*, pages 197–223. Springer, 2001.

G. Poyiadjis, A. Doucet, and S. S. Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80, 2011. doi: 10.1093/biomet/asq062. URL +http://dx.doi.org/10.1093/biomet/asq062.

G. Storvik. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions on signal Processing*, 50(2):281–289, 2002.