example

Kwaku Peprah Adjei[1][2][3] Robert B. O'Hara[4]

Introduction

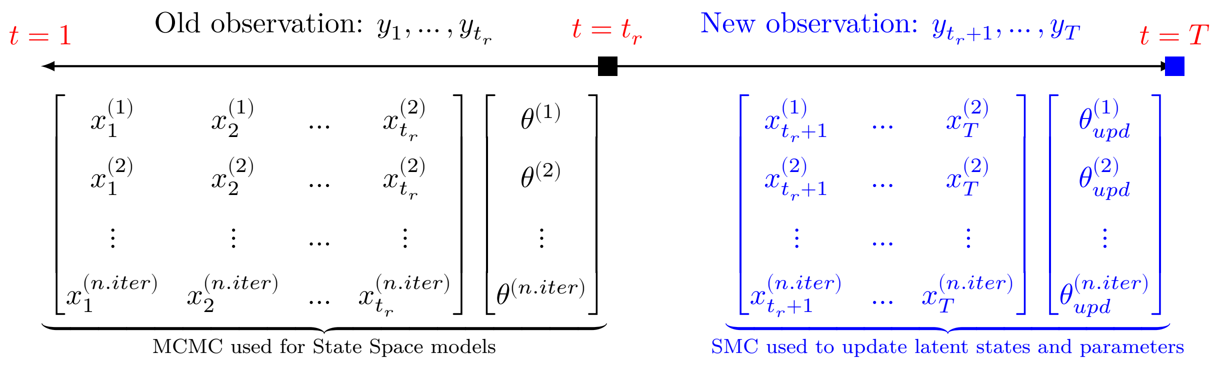
Ecological time series data are collected to explain spatio-temporal changes in species distribution. Species distribution models that explain the abundance or occupancy of the species need to be updated and the model parameters updated to reflect the current trends in the species.

In recent years, the state space models (SSMs) have become widely used in analyzing such data. These SSMs have been fitted in the Bayesian framework using the Markov chain Monte Carlo approach. This approach can be very computationally expensive and takes a relatively long time to fit. Fitting the SSMs with this approach can be infeasible in terms of computational load and time needed to run model which increases as the data gets larger in size.

A faster approach to fitting the SSMs is using the sequential Monte Carlo (SMC) approach. The SMC uses sequential importance sampling (SIS) to obtain importance weights that are used to generate posterior distributions of latent states at every time step and also update the other parameters in the model (using the particle MCMC approach).

This document seeks to demonstrate how once can use MCMC models already fitted to the SSMs and update them using the SMC approach. The bootstrap and auxiliary PFs were discussed in the main papar, but this document will focus on the boostrap particle filters. The reader is expected to be familiar with the nimbleSMC package and the various functions (the reader is referred to Chapter 8 of de Valpine et al. (2022) and Michaud et al. (2021)

for details on how to fit SSMs using SMC approach in NIMBLE). The first part provides a brief introduction to the state space models (SSMs), sequential monte carlo (SMC) approaches (specifically the bootstrap particle filter) and the particle MCMC.


$t = 1$
Old observation: $y_1, \dots, y_{t_r}$
$t = t_r$
New observation: $y_{t_r+1}, \dots, y_T$
$t = T$
MCMC used for State Space models
SMC used to update latent states and parameters

State space models

We assume we can obtain a (multivariate) observed time series data denoted by $y_{1:T} = \{y_1, y_2, \dots, y_T\}$ where y_t is a $(k \times 1)$ observed vector. These observations depend on latent (unobserved but in interest) states $x_{1:T} = \{x_1, x_2, \dots, x_T\}$, where x_t is an $(M \times 1)$ state vector and M is the number of particles at each time step t. These latent states are assumed to have a first order Markov structure (the latent state at time t depends on latent state at time $t-1$ only) and the observations at each time t, y_t, given the latent state at that time x_t are independent of previous observations and states.

A summary of the information we have for the SSM framework is given below:

$$\begin{aligned} \text{Intial state distribution}: &\quad p(x_0|\theta); \quad t = 0 \\ \text{State model}: &\quad p(x_t|x_{t-1}, \theta); \quad t = 1, 2, \dots, T \\ \text{Observation model}: &\quad p(y_t|x_t, \theta); \quad t = 1, 2, \dots, T \end{aligned} \tag{1}$$

where θ are top-level parameters (assumed to be deterministic in equation (1)).

Furthermore, we assume that we have already fitted a SSM, either with MCMC or SMC approaches, to the observed data from time $t = 1$ to $t = t_r$, where $t_r < T$. From the fitted SSM, the $M \times t_r$ latent state particles and weights at $t = t_r$ are stored, and used to update the posterior samples of the latent states $x_{t_r+1}, x_{t_r+2}, \dots, x_T$ and top -level parameters θ using SMC methods to be discussed below.

Sequential Monte Carlo (SMC) methods

The SMC methods used sequential importance sampling (SIS) technique to estimate the filtering distributions Michaud et al. (2021). At each time step t, the latent state x_t is proposed from the previous state x_{t-1} from a proposal distribution or importance function, $\pi(x_t|x_{t-1}, y_{1:t}, \theta)$, and posterior samples of x_t are drawn from the proposed samples using importance weights w_t:

$$w_t^{(i)} \propto \frac{p(x_t^{(i)}|x_{t-1}^{(i)}, y_{1:t}, \theta)}{\pi(x_t^{(i)}|x_{t-1}^{(i)}, y_{1:t}, \theta)} \tag{2}$$

and iteratively as:

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(x_t^{(i)}|x_{t-1}^{(i)}\theta)p(y_{1:t}|x_t^{(i)})}{\pi(x_t^{(i)}|x_{t-1}^{(i)}, y_{1:t}, \theta)} \tag{3}$$

for $i = 1, 2, \dots, M$ particles.

The importance function chosen for SIS is critical to the performance of the SMC method Michaud et al. (2021), the prior distribution of the latent states are chosen as the importance function. In this case, the importance weights in equation (3) simplifies to:

$$w_t^{(i)} \propto w_{t-1}^{(i)} p(y_t|x_t^{(i)}, \theta); \tag{4}$$

for $i = 1, 2, \dots, M$ particles. See Doucet, De Freitas, and Gordon (2001) for the details of the simplification.

This suggests that from time steps $t = t_{r+1}$ to $t = T$ (where we are have new data), we only need information about the weights at $t = t_r$ and the SSM distributions in equation (1) (which are assumed to be known), we can use any of the SMC algorithms to be briefly discussed below to estimating posterior distribution of latent states and update the top - level parameters. The saved posterior samples of latent state for time steps $t = 1, 2, \dots, t_r$ are assumed to be equally weighted samples (i.e. they have equal weights and we choose $w_t^{(i)} = 1, \forall i$). In summary, our proposed framework uses the following information for generating the posterior samples of the latent states:

$$\begin{cases} w_t^{(i)} = 1 & x_t^{(i)} \text{ from RM} \quad \text{for } 1 \leq t \leq t_r \\ w_t^{(i)} \text{ defined by equations } (2) - (4) & x_t^{(i)} \text{ from SMC} \quad \text{for } t_{r+1} \leq t \leq T \end{cases} \tag{5}$$

where RM denotes a reduced SSM fitted to the observed data $y_{1:t_r}$ using either MCMC or any SMC method to be discussed later.

Bootstrap particle filter

The bootstrap particle filter (hereafter, BPF) re-samples with replacement the M particles $(x_{0:t}^{(i)}; i = 1, 2, \dots, M)$ from the set of proposed samples $(\tilde{x}_{0:t}^{(i)}; i = 1, 2, \dots, M)$ according the importance weights $(w_t^{(i)}; i = 1, 2, \dots, M)$ defined in equations (2) to (4). This re-sampling approach mitigates the particle degeneracy problem, where unimportant particles are propagated through time Doucet, De Freitas, and Gordon (2001).

To implement the changes proposed in this paper; the following changes were made to the bootstrap PF algorithm implemented in nimble SMC:

Particle Markov Chain Monte Carlo

For all the SMC methods discussed so far, we have assumed that the parameters θ are deterministic. In most ecological applications, these parameters are stochastic. For example, these parameters can be effects we may be interested in making inferences about. Using the Bayesian framework, the joint likelihood of the latent states and parameters is $p(\theta, x_{1:t} | y_{1:t}) =$

Algorithm 1 Bootstrap filter with constant top-level nodes θ

```
for $t$ in $1 : t_r$ and $i$ in $1 : n.iter$ do
    for $m$ in $1 : M$ do
        Set $w_t^{(m)} := 1$; $x_t^{(m)} := x_t^{(i)}$
    end for
end for
for $t$ in $t_{r+1} : T$ do
    for $m$ in $1 : M$ do
        Generate $\tilde{x}_t^{(m)} \sim q(x_t|x_{t-1}^{(m), y_t})$

        Calculate unnormalised weight $w_t^{(m)} = \frac{f(\tilde{x}_t^{(m)}|x_{t-1}^{(m)})g(y_t|\tilde{x}_t^{(m)})}{q(x_t|x_{t-1}^{(m), y_t})}\pi_{t-1}^{(m)}$
    end for
    for $m$ in $1 : M$ do
        Normalize $w_t^{(m)}$ as $\pi_t^{(m)} := \frac{w_t^{(m)}}{\sum_{i=1}^{M} w_t^{(m)}}$
    end for
    for $m$ in $1 : M$ do
        Sample an index $j$ from the set of $i, \ldots, M$ with probabilities $\{\pi_t^{(m)}\}_{m=1}^{M}$
        Set $x_t^{(m)} = \tilde{x}_t^{(m)}$
        $\pi_t^{(m)} = \frac{1}{M}$
    end for
    Calculate $\tilde{p}(y_{t|1:t-1}) = \frac{1}{M}\sum_{m=1}^{M} w_t^{(m)}$
end for
```

$p_\theta(x_{1:t}|y_{1:t})p(\theta)$; where $p(\theta)$ is the prior distribution for the top-level parameters θ.

The particle MCMC (Andrieu, Doucet, and Holenstein 2010) makes it possible to jointly sample from the posterior distribution of the states and the top-level parameters θ. This algorithm first proposes a value for the top-level parameter ($\theta^\star$) from a proposal distribution fits a SMC method described above with the proposed value $\theta^\star$. This proposed value is accepted or rejected based on a Metropolis Hasting acceptance ratio (MHAR) defined as:

$$\begin{aligned} MHAR &= \frac{p_\theta(y_{1:t})p(\theta^\star)\pi(\theta|\theta^\star)}{p_{\theta^\star}(y_{1:t})p(\theta)\pi(\theta^\star|\theta)}; \\ \text{where} \quad p_\theta(y_{1:t}) &= p_\theta(y_1)\prod_{t=2}^{T} p_\theta(y_n|y_{n-1}) \\ &= p_\theta(y_1)\prod_{t=2}^{T}\sum_{i=1}^{M} w^i_{t|\theta} \end{aligned} \tag{6}$$

where $p_\theta(y_{1:t})$ is the marginal distribution of the observed data given the parameter θ, $w^i_{t|\theta}$ is the i^{th} importance weight at time t given the value of θ, $p(\theta^\star)$ and $p(\theta)$ is the prior distribution of $\theta^\star$ and θ respectively and $\pi(.|.)$ is the proposal distribution for the parameters θ.

Adapting the MHAR defined in equation (6) and weights in equation (5) for our proposed framework, the proposed parameter value ($\theta^\star$) is now accepted or rejected with MHAR:

$$MHAR_{upd} = \frac{p(\theta^\star)\pi(\theta|\theta^\star)\prod_{t=t_r+1}^{T}\sum_{i=1}^{M} w^i_{t|\theta^\star}}{p(\theta)\pi(\theta^\star|\theta)\prod_{t=t_r+1}^{T}\sum_{i=1}^{M} w^i_{t|\theta}}; \tag{7}$$

where $w^i_{t|\theta}$ is the i^{th} importance weight at time t given the value of θ, $p(\theta^\star)$ and $p(\theta)$ is the prior distribution of $\theta^\star$ and θ respectively and $\pi(.|.)$ is the proposal distribution for the parameters θ. See Supplementary Information 1 for details of the MHAR defined by equations (6) and (7).

Algorithm 2 Particle MCMC using the proposed updated model

```
for $i$ in $1 : n.iter$ do
    Generate $\theta^\star \sim q(\theta|\theta_r^{(i)})$
    Run a particle filter with algorithm one to estimate the marginal likelihood $\tilde{p}(y_{1:T}|\theta^\star)$
    At the last time step $T$, draw $x_{1:T}^\star \sim p(x_{1:T}|y_{1:T}, \theta^\star)$ from the full particle filter history
    Compute the Metropolis Hasting acceptance ratio in an equation 7 and choose $a^\star = min(1, MHARupd)$
    Generate $r \sim unif(0, 1)$
    if $a^\star > r$ then
        Set $\theta^{(i)} := \theta^\star$, $x_{1:T}^{(i)} = x^\star_{1:T|\theta^\star}$
    else
        Set $\theta^{(i)} := \theta_r^{(i)}$, $x_{1:T}^{(i)} = x_{\{1:T\}|\theta_r^{(i)}}$
    end if
end for
```

Packages needed 91

```
#devtools::install_github("Peprah94/nimMCMCSMCupdates")

library(nimMCMCSMCupdates)

library(nimble)

library(nimbleSMC)


set.seed(1)
```

Simulating data 94

```
# Setting up MCMC configuration values and variation of parameters

nIterations = 1000

nBurnin = 200

nChains = 3

nThin = 1

nyears = 50
```

```
aVars <- c(0.1, 0.8) # changing the intercept
#High and small values of a
iNodePrev <- c(49, 45, 20) # The number of years for reduced model
aVarstag = 2
iNodetag = 2
mcmcRun <- FALSE #use mcmc or nimbleSMC for reduced Model
pfTypeRun = "auxiliary"

sim2 <- function(a, b, c, t, mu0){
  x <- y <- numeric(t)
  x[1] <- rnorm(1, mu0, 1 )
  y[1] <- rnorm(1, x[1], 1)

  for(k in 2:t){
    x[k] <- rnorm(1, a*x[k -1] + b, 1)
    y[k] <- rnorm(1, x[k-1]*c, 1)# + (sigOE * (sqrt(df -2)/df) * rt(1, df))
  }
  return(list(x=x, y=y))
}

message("simulating data for a = ", aVars[aVarstag])
```

```
simulating data for a = 0.8
```

```
simData <- sim2(a = aVars[aVarstag],
                b = 1,
                c = 1.5,
                t = nyears,
                mu0 = 0.2)


str(simData)
```

```
List of 2
 $ x: num [1:50] -0.426 -0.177 1.188 2.438 3.526 ...
 $ y: num [1:50] -0.243 0.956 -1.086 2.52 3.351 ...
```

Define the NIMBLE model

```
stateSpaceCode <- nimbleCode({
  x[1] ~ dnorm(mu0, 1)
  y[1] ~ dnorm(x[1], 1)
  for(i in 2:t){
    x[i] ~ dnorm(x[i-1] * a + b, 1)
    y[i] ~ dnorm(x[i] * c, 1)
  }
  a ~ dunif(0, 1)
  b ~ dnorm(0, 1)
  c ~ dnorm(1,1)
  mu0 ~ dnorm(0, 1)
})
#
# ## define data, constants, and initial values
data <- list(
  #   #y = c(0.213, 1.025, 0.314, 0.521, 0.895, 1.74, 0.078, 0.474, 0.656, 0.802)
  y = simData$y
)
constants <- list(
  t = nyears
)
inits <- list(
  a = 0.1,
  b = 0,
  mu0= 0.2,
  c = 1
)
#
#
# ## build the model
stateSpaceModel <- nimbleModel(stateSpaceCode,
                               data = data,
                               constants = constants,
                               inits = inits,
                               check = FALSE)
```

Run baseline model

```
newModel <- stateSpaceModel$newModel(replicate = TRUE)


# Function to run the baseline model
 baselineModel <- nimMCMCSMCupdates::baselineSpartaEstimation(
   model = newModel,
   latent = "x",
   MCMCconfiguration = list(target = c('a', 'b', 'c', 'mu0'),
                            additionalPars = "x",
                            n.iter = nIterations,
                            n.chains = nChains,
                            n.burnin = nBurnin,
                            n.thin = nThin))
```

Reduced model

Either with

- MCMC
- SMC

```
data <- list(
  y = simData$y[-c((iNodePrev+1):50)]
)
constants <- list(
  t = iNodePrev
)
newModelReduced <- nimbleModel(stateSpaceCode,
                               data = data,
                               constants = constants,
                               inits = inits,
                               check = FALSE)
```

```r
example1ReducedModel <- spartaNimWeights(
  model = newModelReduced,
  latent = "x",
  mcmc = mcmcRun,
  pfType = pfTypeRun,
  MCMCconfiguration = list(target = c('a', 'b', 'c', 'mu0'),
                           additionalPars = "x",
                           n.iter = nIterations,
                           n.chains = nChains,
                           n.burnin = nBurnin,
                           n.thin = nThin)
)
```

Updated model

```r
example1UpdatedModel <- spartaNimUpdates(
  model = stateSpaceModel, #nimble model
  reducedModel = newModelReduced,
  latent = "x", #latent variable
  pfType = pfTypeRun,
  MCMCconfiguration = list(target = c('a', 'b', 'c', 'mu0'),
                           additionalPars = "x",
                           n.iter = (nIterations - nBurnin)/nThin,
                           n.chains = nChains,
                           n.burnin = 0,
                              n.thin = 1),  #saved loglikelihoods from reduced model
  postReducedMCMC = example1ReducedModel,# MCMC summary to use as initial values
  pfControl = list(saveAll = TRUE,
                   lookahead = "mean",
                   smoothing = FALSE,
                   mcmc = mcmcRun,
                   M = nyears - iNodePrev,
                   iNodePrev = iNodePrev)
)
```

References

Andrieu, Christophe, Arnaud Doucet, and Roman Holenstein. 2010. "Particle Markov Chain Monte Carlo Methods." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72 (3): 269–342.

Arulampalam, M Sanjeev, Simon Maskell, Neil Gordon, and Tim Clapp. 2002. "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking." *IEEE Transactions on Signal Processing* 50 (2): 174–88.

de Valpine, Perry, Christopher Paciorek, Daniel Turek, Nick Michaud, Cliff Anderson-Bergman, Fritz Obermeyer, Claudia Wehrhahn Cortes, Abel Rodrìguez, Duncan Temple Lang, and Sally Paganin. 2022. *NIMBLE User Manual* (version 0.13.1). https://doi.org/10.5281/zenodo.1211190.

Doucet, Arnaud, Nando De Freitas, and Neil Gordon. 2001. "An Introduction to Sequential Monte Carlo Methods." *Sequential Monte Carlo Methods in Practice*, 3–14.

Michaud, Nicholas, Perry de Valpine, Daniel Turek, Christopher J Paciorek, and Dao Nguyen. 2021. "Sequential Monte Carlo Methods in the Nimble and nimbleSMC r Packages." *Jour-*

nal of Statistical Software 100: 1–39.