NIMBLE PMCMC Comparison to POMP and Biips packages

```
## Derived CmodelBaseClass created by buildModelInterface for model code
```

We first compare runtimes for the NIMBLE PMCMC implementations to those of POMP and Biips. Below we provide specific information about each implementation:

- NIMBLE currently has two implementations of PMCMC - one version which resamples the log likelihood at each iteration (labeled Resamp), and one version which stores the log likelihood until a new sample is accepted (labeled NoResamp). We also consider NIMBLE PMCMC algorithms using adaptation (labled Adapt) and not using adaptation (labled NoAdapt). Each NIMBLE PMCMC algorithm uses a block sampler with a multivariate normal proposal distribution.
- POMP's PMCMC algorithm does not allow for adaptation, and follows the NoResamp method described above. POMP allows user specification of the proposal distribution. We set the proposals to again come from a multivariate normal distribution.
- The Biips PMCMC algorithm allows adaptation. It is not clear from the documentation whether Biips resamples log likelihood at each time point or not. In addition, Biips does not currently have block sampling implemented, so our proposal distributions are univariate normal for each parameter.

We again use Daniel's correlated model for analysis. Below is a table which gives a comparison of run-times for different combinations of n (the number of iterations to run the chain for) and m (the number of particles to use at each iteration). The initial values for each time the algorithm was run were set to the same values ($a = 0.95$, $b = 1.0$, $\sigma_{PN} = 0.2$, $\sigma_{OE} = 0.05$). For each combination of m and n, we ran the PMCMC algorithm five times for each algorithm, and took the median of those five run-times. All median times are provided in seconds.

The chart below shows a few interesting facts:

- Using adaptation in NIMBLE seems to add a very slight amount of time to the overall run-time of the algorithm. We used adaptation intervals of 200, so this effect is not seen until higher iteration values.
- Using a NoResamp algrthm vs. a Resamp algorithm in NIMBLE cuts run-time roughly in half for all numbers of iterations and particle sizes. However, as will be seen in the plots later in this document, the NoResamp algorithm seems to have issues with mixing well.
- NIMBLE runs faster than POMP when using the NoResamp algorithm, which POMP also uses. Using a Resamp algorithm for NIMBLE causes it to be around 50% slower than POMP.

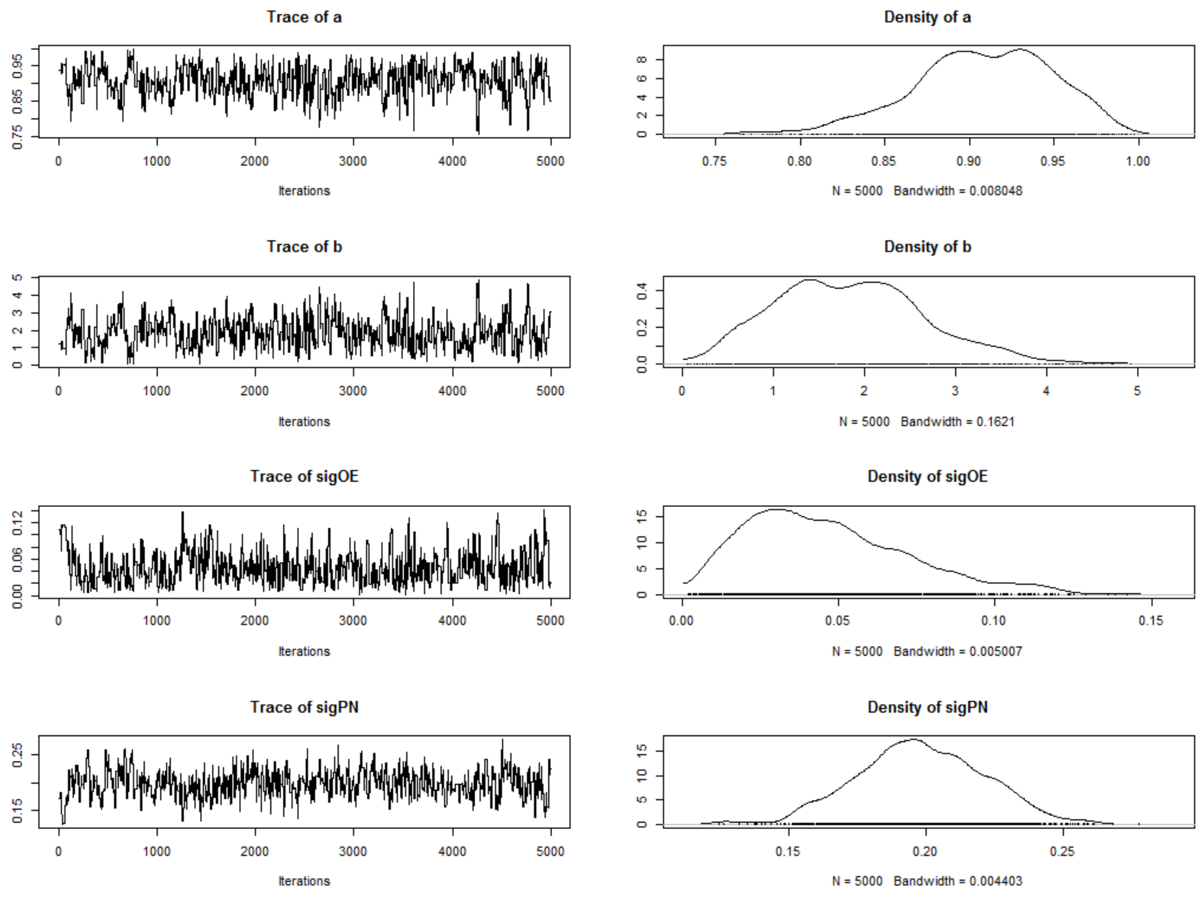
% latex table generated in R 3.2.0 by xtable 1.7-4 package % Thu Jun 25 00:58:19 2015

	Method	N..Particles	N..Iterations.	Median
2	NIMBLE PMCMC Resamp Adapt	100	100	0.96
3	NIMBLE PMCMC Resamp Adapt	100	1000	9.03
4	NIMBLE PMCMC Resamp Adapt	500	100	4.57
5	NIMBLE PMCMC Resamp Adapt	500	1000	43.28
6	NIMBLE PMCMC Resamp Adapt	1000	100	9.29
7	NIMBLE PMCMC Resamp Adapt	1000	1000	92.31
8	NIMBLE PMCMC Resamp NoAdapt	100	100	0.94
9	NIMBLE PMCMC Resamp NoAdapt	100	1000	9.61
10	NIMBLE PMCMC Resamp NoAdapt	500	100	4.52
11	NIMBLE PMCMC Resamp NoAdapt	500	1000	44.45
12	NIMBLE PMCMC Resamp NoAdapt	1000	100	8.78
13	NIMBLE PMCMC Resamp NoAdapt	1000	1000	87.72
14	NIMBLE PMCMC NoResamp Adapt	100	100	0.45
15	NIMBLE PMCMC NoResamp Adapt	100	1000	4.86
16	NIMBLE PMCMC NoResamp Adapt	500	100	2.22
17	NIMBLE PMCMC NoResamp Adapt	500	1000	22.22
18	NIMBLE PMCMC NoResamp Adapt	1000	100	4.53
19	NIMBLE PMCMC NoResamp Adapt	1000	1000	43.98
20	NIMBLE PMCMC NoResamp NoAdapt	100	100	0.52
21	NIMBLE PMCMC NoResamp NoAdapt	100	1000	4.66
22	NIMBLE PMCMC NoResamp NoAdapt	500	100	2.26
23	NIMBLE PMCMC NoResamp NoAdapt	500	1000	22.11
24	NIMBLE PMCMC NoResamp NoAdapt	1000	100	4.25
25	NIMBLE PMCMC NoResamp NoAdapt	1000	1000	43.09
26	POMP PMCMC	100	100	2.39
27	POMP PMCMC	100	1000	24.59
28	POMP PMCMC	500	100	3.75
29	POMP PMCMC	500	1000	38.91
30	POMP PMCMC	1000	100	5.69
31	POMP PMCMC	1000	1000	58.47
32	Biips PMCMC	100	100	8.22
33	Biips PMCMC	100	1000	75.53
34	Biips PMCMC	500	100	38.99
35	Biips PMCMC	500	1000	387.4

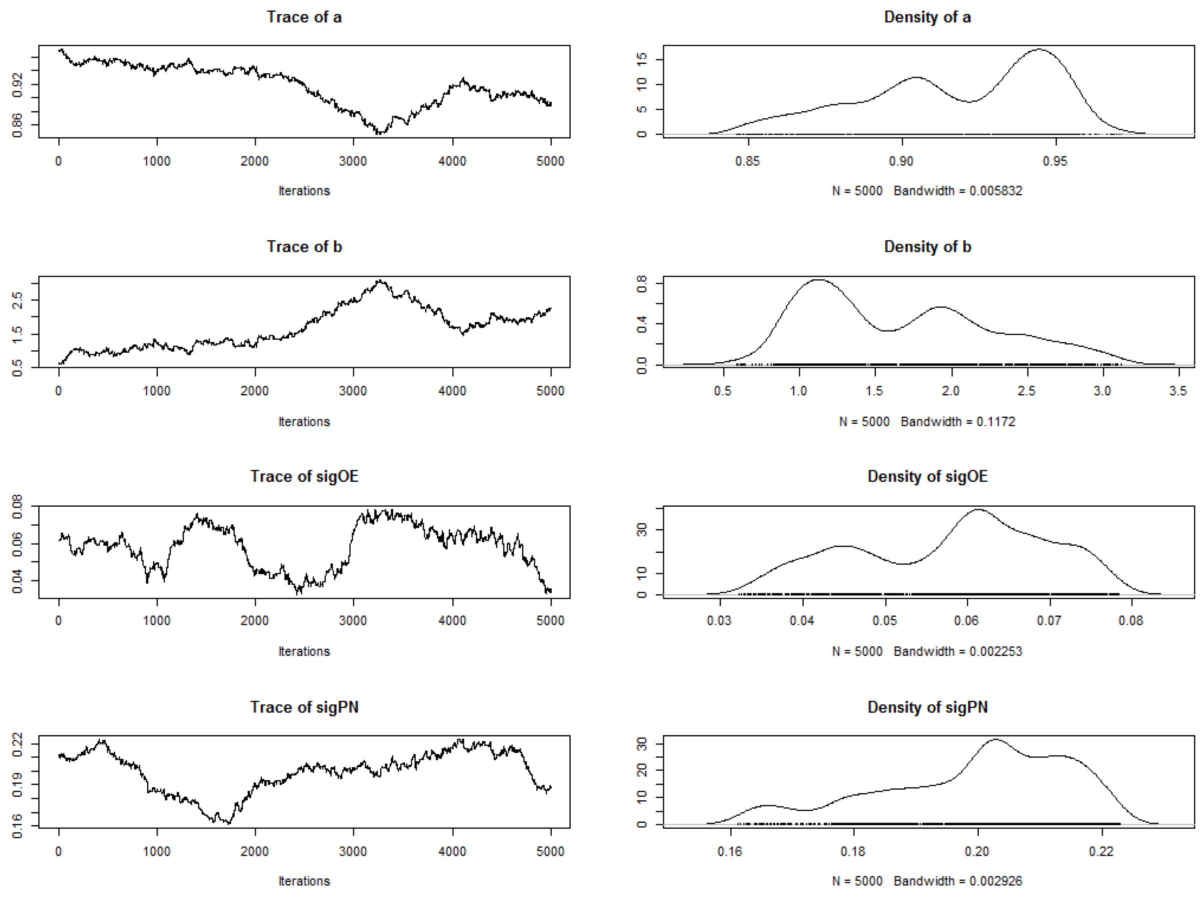
We next provide trace plots and posterior distributions for our parameters produced by the different PMCMC algorithms in NIMBLE and POMP. To produce these plots, we conducted a single run of 7,500 iterations (2,500 burn-in period) with 1,000 particles per iteration. We did not include the Biips PMCMC method, as Biips was running very slowly for such a large number of particles / iterations.

Looking at the plots below, only the NIMBLE Resamp Adapt PMCMC algorithm seems to be mixing well. As Perry guessed previously, running the PMCMC without resampling the log-likelihood at each time point seems to cause the algorithm to get stuck at certain high likelihood values. It is also possible / likely that the choice of a better proposal covariance matrix would lead to better mixing for the non-adaptive algorithms.

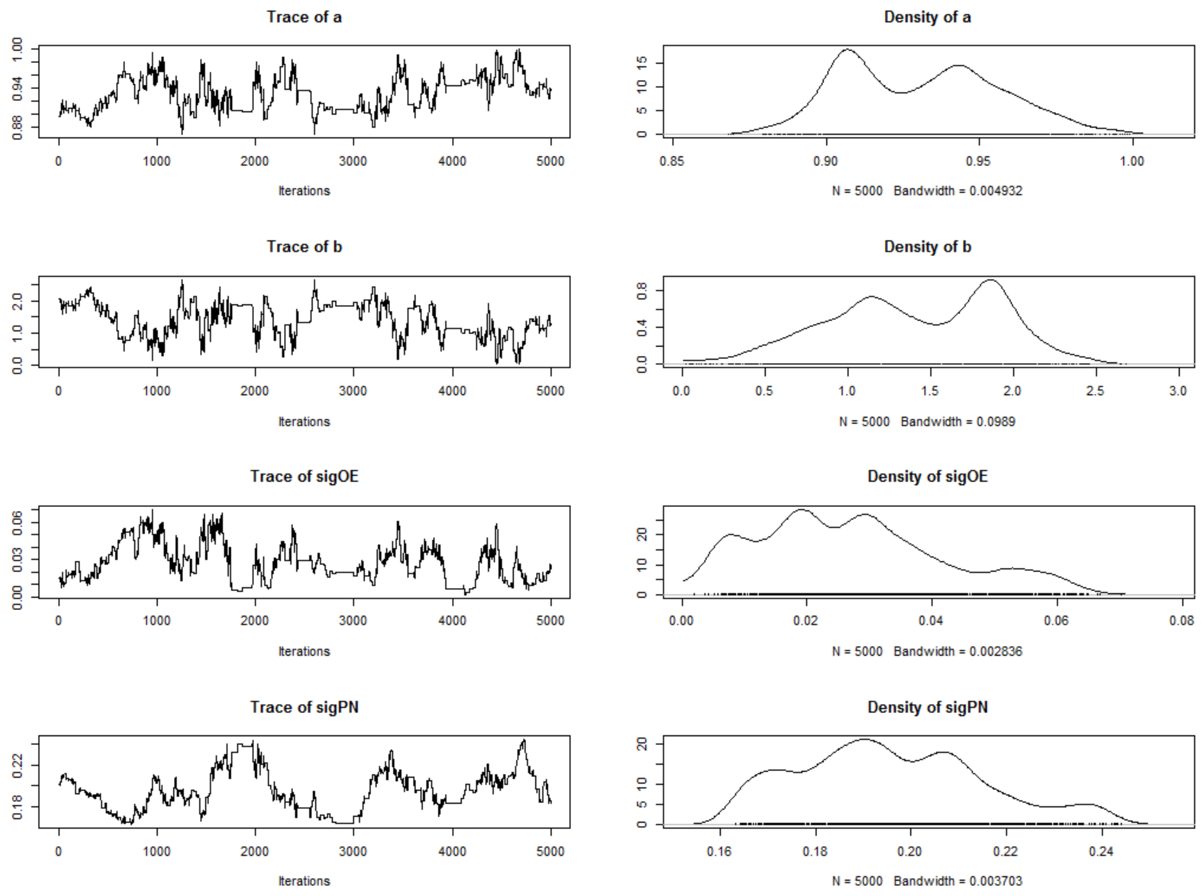
NIMBLE PMCMC Resamp Adapt



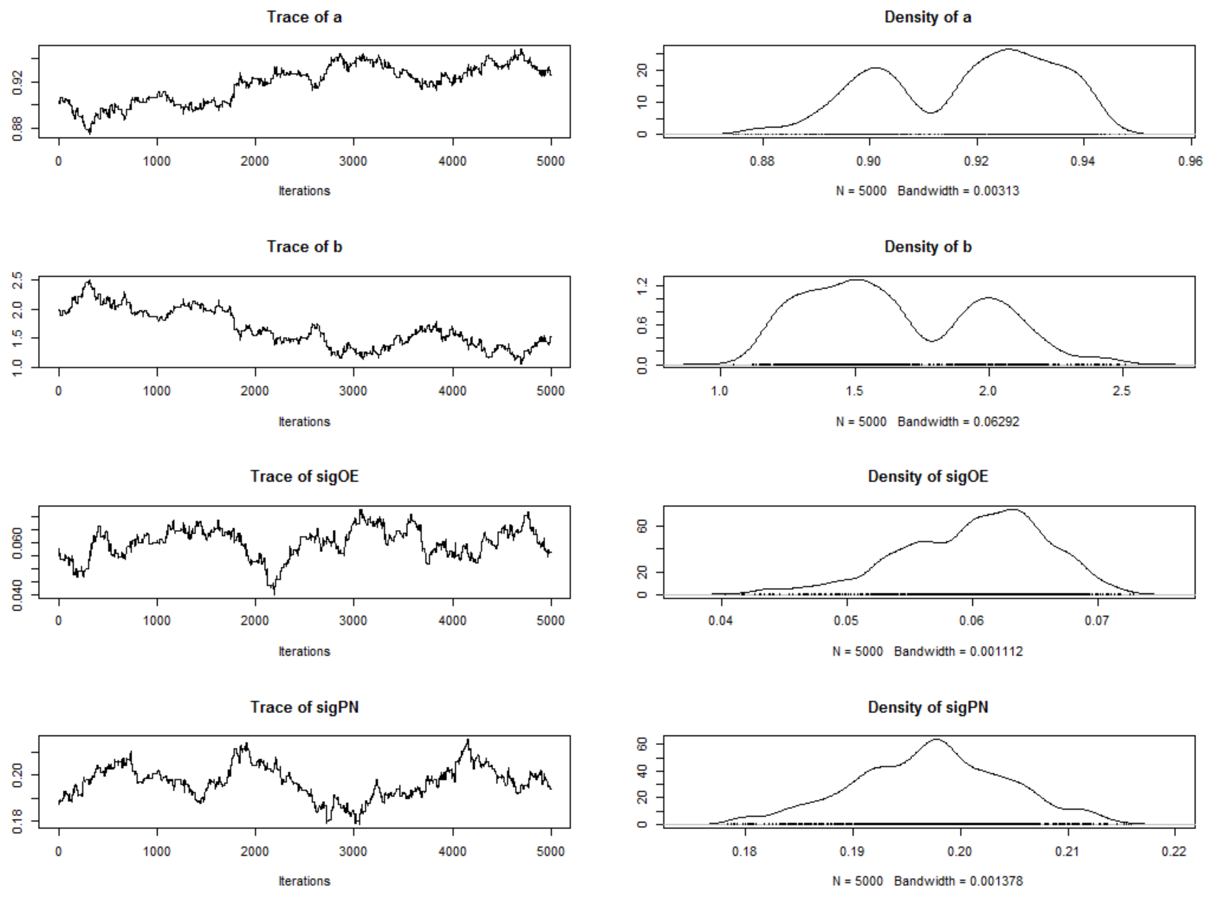
NIMBLE PMCMC Resamp NoAdapt



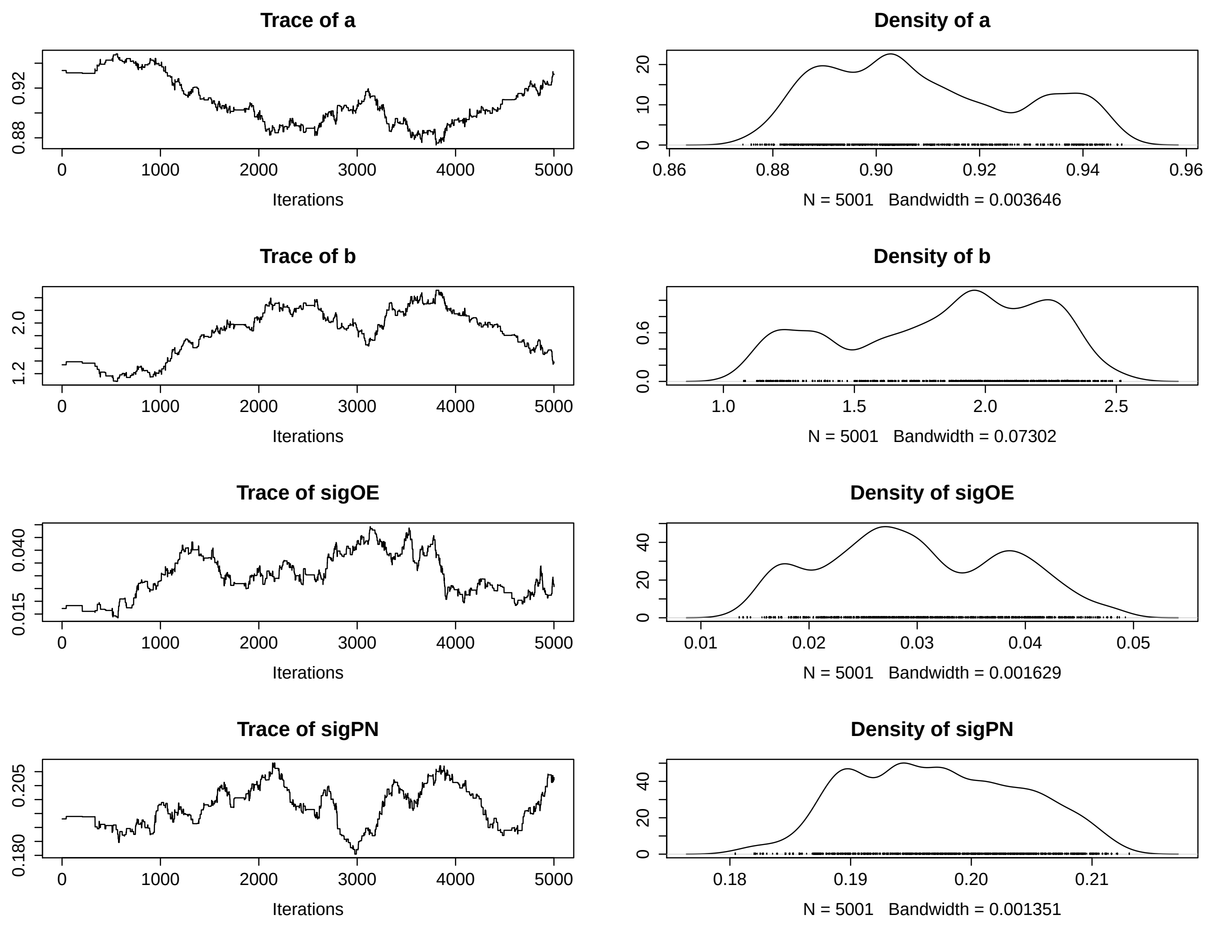
NIMBLE PMCMC NoResamp Adapt



NIMBLE PMCMC NoResamp NoAdapt



POMP PMCMC (NoResamp NoAdapt)



We finally provide a table listing the effective MCMC sample size for each parameter, for each of the different PMCMC algorithms we ran. As seen in the plots above, the NIMBLE Resamp Adapt algorithm gives much higher effective sample sizes than the other algorithms.

% latex table generated in R 3.2.0 by xtable 1.7-4 package % Thu Jun 25 00:58:19 2015

	Method	a	b	sigOE	sigPN
2	NIMBLE PMCMC Resamp Adapt	220.62	221.08	220.84	219.61
3	NIMBLE PMCMC Resamp NoAdapt	1.81	1.48	3.48	4.64
4	NIMBLE PMCMC NoResamp Adapt	28.14	27.67	35.39	52.93
5	NIMBLE PMCMC NoResamp NoAdapt	2.09	1.82	4.41	3.15
6	POMP PMCMC	2.29	2.24	4.43	5.19