

Theta logistic population model writeup

Casper W. Berg

October 11, 2012

1 Summary

- ADMB is by far the fastest method, but it takes a little bit more coding than doing it in the BUGS language.
- HMM can be implemented in R, but it takes some effort and is quite slow.
- JAGS is slower but not too bad for this relatively simple problem.
- All methods produce quite similar results with respect to parameter estimates.
- JAGS produced much wider credible intervals for the estimated parameters than the corresponding confidence intervals for ADMB and HMM.
- For this application, choosing the mean of the MCMC chains from the BUGS runs instead of the mode leads to more bias in the parameter estimates.
- Coverage fractions are close to 0.95 for all the methods.

2 Introduction

This example is based on Pedersen et al. (2011), which contains much more details than this shorter writeup.

The example is a theta logistic population model, which is a nonlinear state-space model. The log-transformed population size is modelled as a nonlinear function of its previous size in the following way:

$$X_t = X_{t-1} + r_0 \left(1 - \left(\frac{\exp(X_{t-1})}{K} \right)^\theta \right) + e_t, \quad (1)$$

$$Y_t = X_t + u_t, \quad (2)$$

where $e_t \sim N(0, Q)$ and $u_t \sim N(0, R)$.

This example does not contain any real data, so instead we will simulate 200 timesteps of the population with an initial value substantially below the carrying capacity, such that in the beginning the population will grow, and in the end of the time series the population size will fluctuate around the carrying capacity. If we had only observed the population fluctuating around the carrying capacity it might have been impossible to identify all the parameters in the model, e.g. θ , and a simpler model should be chosen, for instance by setting $\theta = 1$.

3 Basics

A state-space model describes the dynamics of a latent state (\mathbf{X}_t) and how data (Y_t) relate to this state. An important feature of SSMs is their ability to model random variations in the latent state and in data. For $t \in \{1, \dots, N\}$ the general system and observation equations of the SSM are respectively $\mathbf{X}_t = g(t, \mathbf{X}_{t-1}, \mathbf{e}_t)$, and $\mathbf{Y}_t = h(t, \mathbf{X}_t, \mathbf{u}_t)$, where $\mathbf{e}_t \sim N(\mathbf{0}, \mathbf{Q}_t)$ is the system error and $u_t \sim N(0, \mathbf{R}_t)$ is the observation error. Here, “ $\sim N(\cdot)$ ” means Gaussian distributed. Because of the possible nonlinearity of g and h , advanced filtering and smoothing methods must be employed to gain meaningful estimates of \mathbf{X}_t . In this respect, the extended Kalman filter, the unscented Kalman filter, and Bayesian filtering e.g. using Markov chain Monte Carlo (MCMC) sampling or BUGS are common approaches. Alternative methods for nonlinear state estimation are hidden Markov models (HMMs, Zucchini and MacDonald, 2009) and mixed effects models using the software AD Model Builder (ADMB).

All parameters in this example except the carrying capacity, K , has been log transformed to ensure positive values. The carrying capacity might also have been log transformed as this should obviously also be positive, but since the chosen true value is far from zero we judged that log-transforming K was not critical. Besides being convenient for ensuring positive values, log transforming parameters also changes the shape of the likelihood surface. If standard confidence intervals based on quadratic approximations of the likelihood surface are used, it is of course important, that this quadratic approximation is reasonable, which it might often not be when dealing with nonlinear problems. In this problem, θ and r_0 are known to be highly correlated, and with a boomerang-shaped likelihood surface when not log transformed (Polansky et al., 2009), which is undesirable.

4 ADMB

The ADMB solution to this problem is the same basic template used for any type of state-space model: The states are declared as random effects, and the system and observation equations are put into `SEPARABLE_FUNCTIONS` in order gain huge speed improvements. ADMB will then be able to exploit the fact, that the covariance matrix for all the states in a state-space model is a banded matrix, and skip work on all the zeroes.

The data section of the ADMB solution contains the initial parameter values (this is just an alternative to using a `.pin` file), the number of data points N , and the observations y .

The parameter section contains all the fixed effects (starting with `init`), the random effects, which are the latent states (population size), and finally the objective function which is the joint negative log-likelihood.

The procedure section is quite simple for this problem, as all the likelihood contribution is done within the `SEPARABLE_FUNCTIONS`. The only thing we are doing outside the `SEPARABLE_FUNCTIONS` is to set the values of the `sdreport_numbers`. Notice, that we pass the log-transformed values (i.e. the parameters to be estimated) to the `SEPARABLE_FUNCTIONS`, and not the `sdreport_numbers`. This is because all computations used in the likelihood must be performed inside the `SEPARABLE_FUNCTIONS` and NOT in the procedure section.

```

1  //////////////////////////////////////
2  // ADMB code for the theta logistic population model.
3  //
4  // This model should be compiled with -r flag for random effects,
5  // I.e. admb -r admbmodel
6  // and should be executed with the -noinit flag,
7  // I.e. admbmodel -noinit
8  //
9  // Author: Casper W. Berg, 13-07-2010
10 //////////////////////////////////////
11 DATA_SECTION
12     init_vector initPars(1,5);
13     init_number N;
14     init_vector y(1,N)
15 PARAMETER_SECTION
16     init_number logSdU;
17     init_number logSdy;
18     init_number logr0;
19     init_number logTheta;
20     init_bounded_number K(100.0,2000.0);
21     random_effects_vector U(1,N);
22     objective_function_value jnll;
23
24     // do delta method to find confidence intervals
25     // for the following untransformed parameters
26     sdreport_number r0;
27     sdreport_number theta;
28     sdreport_number SdU;
29     sdreport_number Sdy;
30
31 PRELIMINARY_CALCS_SECTION
32     //initialize parameters
33     logr0 = log(initPars(1));
34     K = initPars(2);
35     logSdU = log(initPars(3));

```

```

36 logSdy = log(initPars(4));
37 logTheta = log(initPars(5));
38
39 PROCEDURE_SECTION
40 jnll=0.0; // joint negative log-likelihood to be minimized.
41 r0 = exp(logr0);
42 theta=exp(logTheta);
43 SdU = exp(logSdU);
44 Sdy = exp(logSdy);
45
46 // transition equation (Eq. 3)
47 for(int i=2; i<=N; ++i){
48     step(U(i-1),U(i),logSdU,logr0,K,i-1,logTheta);
49 }
50 // observation equation (Eq. 4)
51 for(int i=1; i<=N; ++i){
52     obs( logSdy, U(i), i);
53 }
54
55 SEPARABLE_FUNCTION void step(const dvariable& U1, const dvariable& U2,
56                             const dvariable& logSdU, const dvariable& logr0,
57                             const dvariable& K, int i, const dvariable& logTheta)
58 {
59     dvariable var2=exp(2.0*logSdU);
60     dvariable r0 = exp(logr0);
61     dvariable theta = exp(logTheta);
62     dvariable pred = U1 + r0*(1.0-pow( exp(U1)/K,theta ));
63     jnll+=0.5*(log(2.0*M_PI*var2)+square(pred-U2)/var2);
64 }
65
66 SEPARABLE_FUNCTION void obs(const dvariable& logSdy, const dvariable& x,
67                             int i)
68 {
69     dvariable var=exp(2.0*logSdy);
70     jnll+=0.5*(log(2.0*M_PI*var)+square(x-y(i))/var);
71 }
72
73 TOP_OF_MAIN_SECTION
74 // Set max n.o. independent variables to 1000 and increase memory.
75 gradient_structure::set_MAX_NVAR_OFFSET(1000);
76 arrmblsize=2000000;

```

5 BUGS

In Pedersen et al. (2011) we used OpenBUGS to run our model, which is possible from within R using the `rbugs` package. In this example we changed to JAGS and the `R2Jags` package, as OpenBUGS apparently is problematic to run using MacOS, and we wanted platform independent solutions if possible. The BUGS code was able to run unchanged using JAGS, and there were only minor differences between `rbugs` and `R2Jags`.

As to keep the results comparable between the frequentist and Bayesian

methods, we wanted vague (or uninformative) priors on our parameters, so we assigned uniform priors with wide support to all the fixed parameters, except the variance parameters for which we tried two different priors. It is common to assign vague inverse-gamma distributed priors to variance parameters (Spiegelhalter et al., 2003; Lambert et al., 2005). Gelman (2006), however, recommends using a uniform prior on the log-transformed standard deviation. Therefore, as in Pedersen et al. (2011), to assess the sensitivity of the estimation results to the choice of prior we perform BUGS estimation using both types of priors.

The OpenBUGS versions used in Pedersen et al. (2011) showed significant differences in running time, with the inverse-gamma distribution on the variances being six times faster than using a uniform distribution on the log-transformed standard deviations. JAGS however showed no noticeable difference in running speed using the two different priors.

```

1  # BUGS script for analyzing the theta logistic population model
2  # Authors: C.W. Berg and M.W. Pedersen, 13/7-2010
3
4  model{
5
6  # Define prior densities for parameters
7  K      ~ dunif(1.0, 22000.0)
8  logr0   ~ dunif(-4.0, 2.0)
9  logtheta ~ dunif(-4.0, 2.0)
10 iQ ~ dgamma(0.0001,0.0001)
11 iR ~ dgamma(0.0001,0.0001)
12
13 # Transform parameters to fit in the model
14 r0 <- exp(logr0)
15 theta <- exp(logtheta)
16
17 # Initial state
18 x[1] ~ dunif(0,10)
19
20 # Loop over all states, Eq. 3 in paper
21 for(t in 1:(N-1)){
22   mu[t] <- x[t] + r0 * ( 1 - pow(exp(x[t])/K, theta) )
23   x[t+1] ~ dnorm(mu[t],iQ)
24 }
25
26 # Loop over all observations, Eq. 4 in paper
27 for(t in 1:(N)){
28   y[t] ~ dnorm(x[t],iR)
29 }
30
31 }
```

```

1  model{
2
3  K      ~ dunif(1.0, 22000.0)
4  logr0  ~ dunif(-4.0, 2.0)
5  logtheta ~ dunif(-4.0, 2.0)
6  stdQ ~ dunif(0,100)
7  stdR ~ dunif(0,100)
8  iQ <- 1/(stdQ*stdQ);
9  iR <- 1/(stdR*stdR);
10
11 r0 <- exp(logr0)
12 theta <- exp(logtheta)
13
14
15 x[1] ~ dunif(0,10)
16
17 for(t in 1:(N-1)){
18   mu[t] <- x[t] + r0 * ( 1 - pow(exp(x[t])/K, theta) )
19   x[t+1] ~ dnorm(mu[t],iQ)
20 }
21
22
23 for(t in 1:(N)){
24   y[t] ~ dnorm(x[t],iR)
25 }
26
27 }

```

6 R

There is - to my knowledge - no easy way to do nonlinear state-space models in R. In Pedersen et al. (2011) it is described how it can be done by discretizing the continuous state-space and reformulating the state-space model as a hidden Markov model, and Matlab code was provided in the online supplementary material. I translated this code into R, and it is available on the svn-repository on the NCEAS website. I will not go into details about the R implementation in this writeup.

7 Simulation results

The simulation results show, that ADMB is many orders of magnitude faster than JAGS and the HMM implementation in R (1), though it should be noted, that the two latter methods might be able to produce similar results in shorter time, by choosing shorter mcmc-chains for JAGS and a coarser grid for the HMM solution.

From the distribution of parameter estimates (figure 2) we must note, that the estimates of K , $\log r_0$, and θ seems to be biased – K and $\log r_0$ positively and θ negatively. This is not too surprising considering that these parameters are quite correlated (this can be confirmed for instance by inspecting the matrix of parameter correlations produced by ADMB in the .cor file), and that the likelihood surface may be ugly in the sense that it is not well approximated by a quadratic form as discussed in Pedersen et al. (2011).

All of the methods seem to perform equally well with respect to bias in parameter estimates.

This is also evident in figure 3, which show both point estimates and CIs for each simulation and method.

Figure 4 summarises the root mean square error (rmse), length of the confidence intervals, and coverage fractions (the fraction of simulations where the CI contains the true value). It shows, that ADMB and the HMM method gives roughly the same results, whereas JAGS has a greater error and wider CIs. The coverage fractions are not too different – all of them are relatively close to the expected value of 0.95. The wider CIs produced by JAGS is consistent with the results obtained in Pedersen et al. (2011).

All the comparisons discussed so far has been using the mode of the MCMC chains as the parameter estimate for the BUGS methods. For posterior distributions obtained from MCMC that are approximately normal it does of course not matter whether the mean, mode or median is chosen as they are identical for the normal distribution. In this case however, the posterior distributions are not close to normal, and so the choice of estimator is important. Figure 5 is identical to figure 2 except that the mean of the chains instead of the mode was chosen for the BUGS runs. Choosing the mean in this case clearly results in more biased estimates of the parameters.

References

- Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models. *Bayesian analysis* 1(3), 515–533.
- Lambert, P., A. Sutton, P. Burton, K. Abrams, and D. Jones (2005). How vague is vague? A simulation study of the impact of the use of vague prior distributions in MCMC using WinBUGS. *Statistics in Medicine* 24(15), 2401–2428.
- Pedersen, M., C. Berg, U. Thygesen, A. Nielsen, and H. Madsen (2011). Estimation methods for nonlinear state-space models in ecology. *Ecological Modelling* 222(8), 1394–1400.
- Polansky, L., P. De Valpine, J. Lloyd-Smith, and W. Getz (2009). Likelihood ridges and multimodality in population growth rate models. *Ecology* 90(8), 2313–2320.

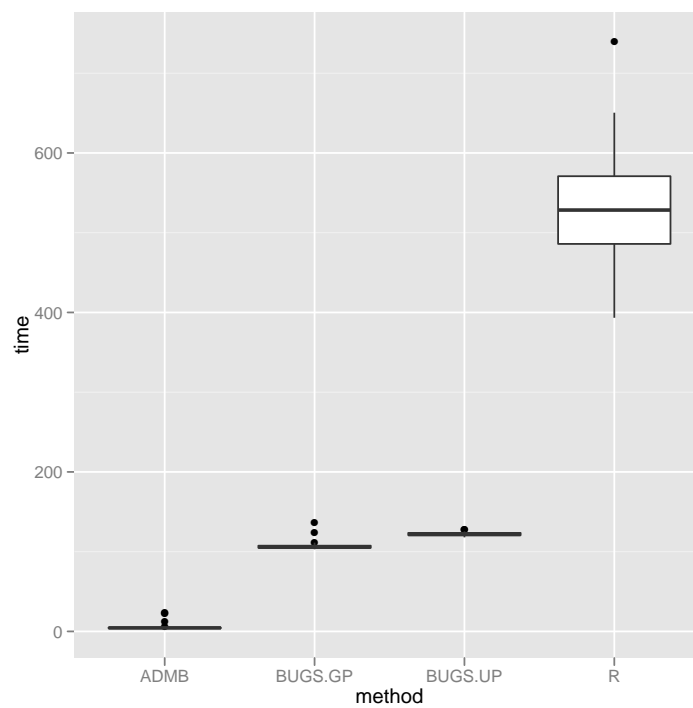


Figure 1: Timings

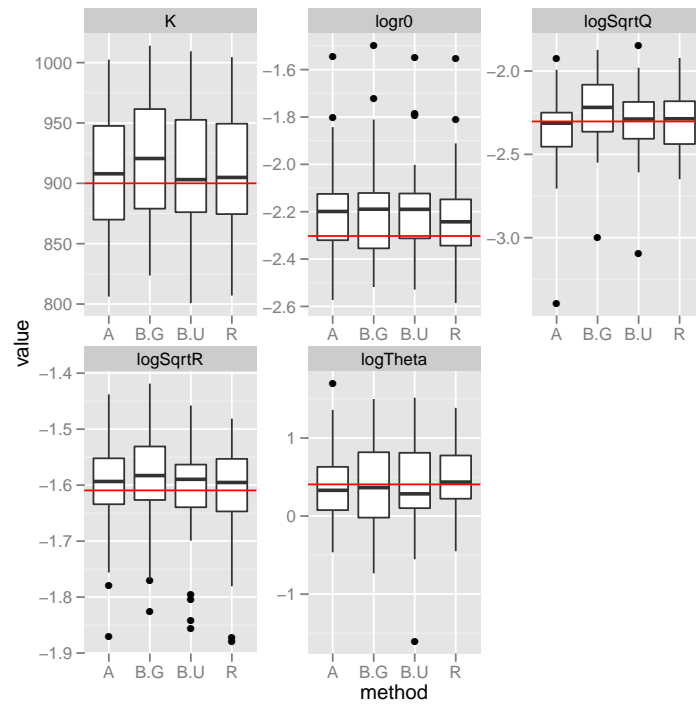


Figure 2: Distribution of parameter estimates

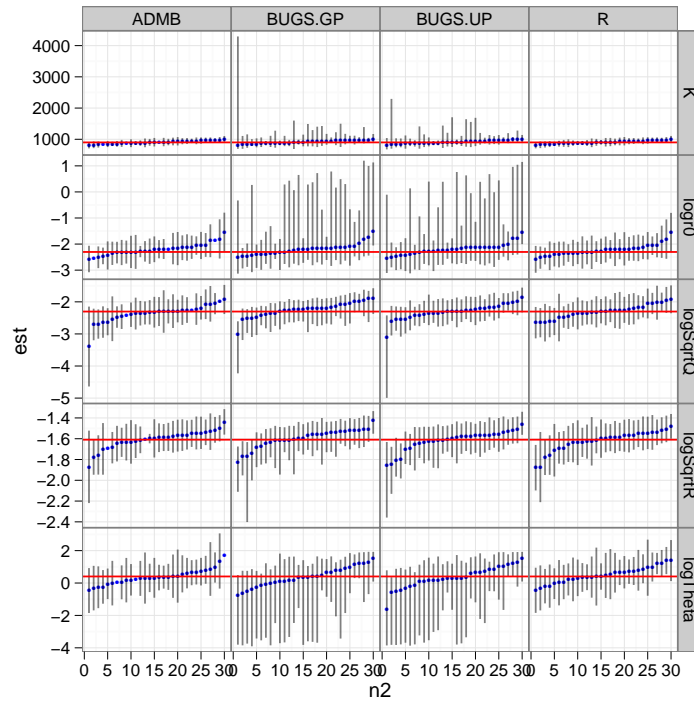


Figure 3: Full distribution of parameters+CIs

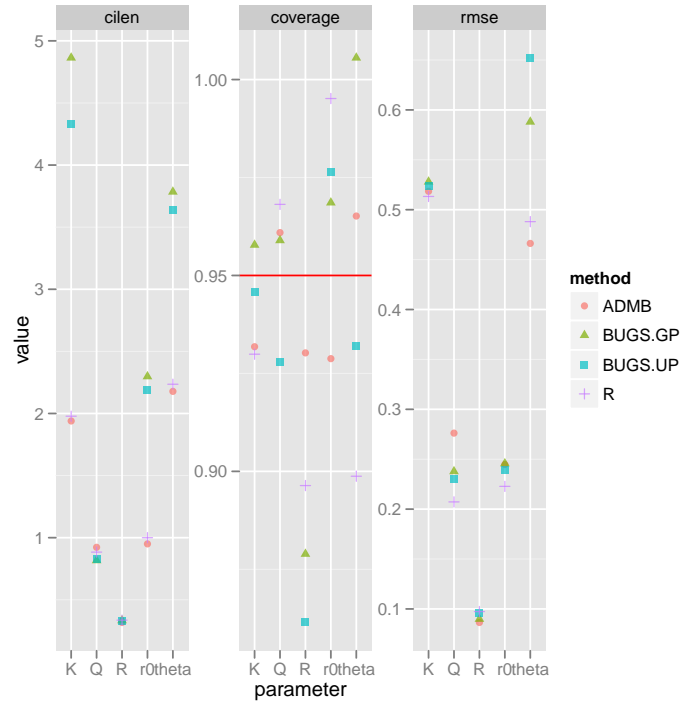


Figure 4: Confidence interval summaries. Rmse and interval lengths for the parameter K has been divided by 100 to get a better scaling on the y-axis. Coverage fractions has been added a small amount of jitter to avoid overlapping values.

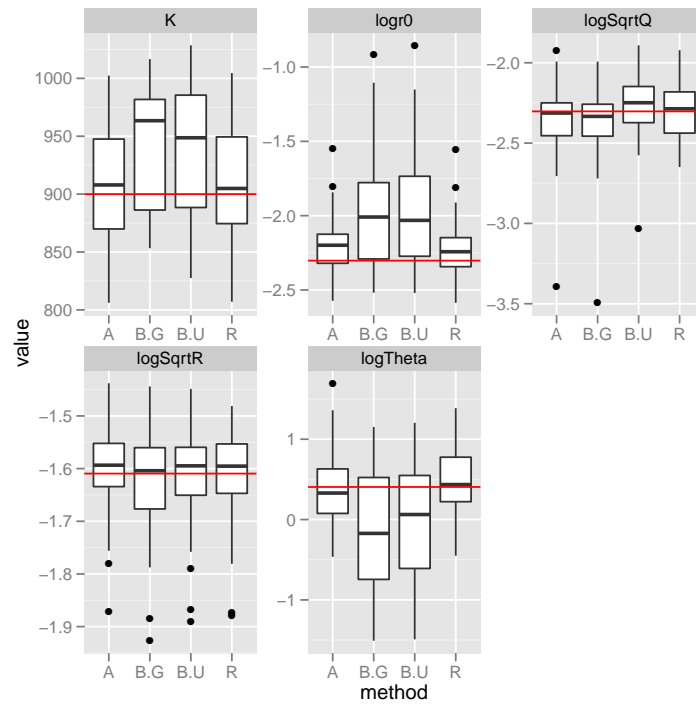


Figure 5: Distribution of parameter estimates using the mean value of the chains for BUGS runs instead of the mode.

- Spiegelhalter, D., A. Thomas, N. Best, and D. Lunn (2003). WinBUGS user manual. *Version 1.4. MRC Biostatistics Unit, Cambridge, UK.*
- Zucchini, W. and I. MacDonald (2009). *Hidden Markov Models for Time Series.* London: Chapman & Hall/CRC.