Comparison of ADMB-RE and JAGS Bayesian State-space length disaggregated population model: Estimating total mortality by decade for winter skate (*Leucoraja ocellata*)

Trevor Davies, Dalhousie University, Halifax, Canada
Steven J.D. Martell, International Pacific Halibut Commission, Seattle WA

March 20, 2013

```
> library(reshape2) ## for 'melt' function with ggplot2
> library(ggplot2)  ## pictures
> library(coda)     ## analysis of MCMC runs (JAGS \& ADMB)
> library(R2admb)   ## R interface to AD Model Builder
> library(R2jags)   ## R interface to JAGS (BUGS dialect)
> library(xtable)   ## Makes latex tables from R dataframes \& objects
> library(gdata)    ## Rename vars package
```

1 Summary

The model described here is a fully Bayesian state-space model implemented in both JAGS (Plummer, 2003) and AD Model Builder (ADMB; (Fournier et al., 2012)) with random effects. The goal of the model was to obtain decadal mortality estimates of three different size classes of winter skates (*Leucoraja ocellata*) on the eastern Scotian Shelf. The three time series of length disaggregated stages (Figure 1; Juvenile1, Juvenile2, and adult) are largely non-informative to the model parameters: catchability, recruitment rate, and stage transition probability. Consequently, it was necessary to include highly informative priors for these parameters to obtain reasonable mortality estimates. We did not implement this model in R due to the model containing 28 hyper parameters and 105 random effects which would have resulted in excessively long run times.

The full model description and alternative model formulations are fully described in Swain et al. (2009).

- ADMB-RE, ADMB in MCMC mode, and JAGS all gave similar results
- ADMB-RE was able to obtain maximum likelihood estimate (MLEs) much more; quickly than JAGS was able to converge to a stationary;
- ADMB-RE run in MCMC was unable to converge to stationary; posteriors even when twice the burn-in of the JAGS model was implemented
- Chain mixing appears to be much better in JAGS compared to ADMB-RE; however, the effect of this on parameter estimates is unknown.
- Coding of JAGS model was easier compared to ADMB-re

Table 1: Run times of ADMB-RE (Maximum Likelihood Estimate only; MLE); ADMB-RE with MCMC mode on (MCMC), and JAGS run times in seconds. Both JAGS and ADMB-RE were run using a single chain (a requirement of ADMB-RE) for 175 000 iterations, discarding the first 100 000 for burn-in, and sampling every 30th sample to reduce autocorrelation between samples.

ADMB	ADMBmcmc	JAGS
13.0	61.6	174.2

2 Introduction

The model implemented here is a stage-structured state-space model that was previously used to explore trends in natural mortality for winter skate (*Leucoraja ocellata*) in the eastern Scotian Shelf. Data on relative abundance for each of three length-classes (or stages) were obtained from bottom trawl surveys in July and September since 1970 (Figure 1). Age-disaggregated data are not available for this species.

3 The Data

Three time series of relative numbers of individuals by size class are as follows:

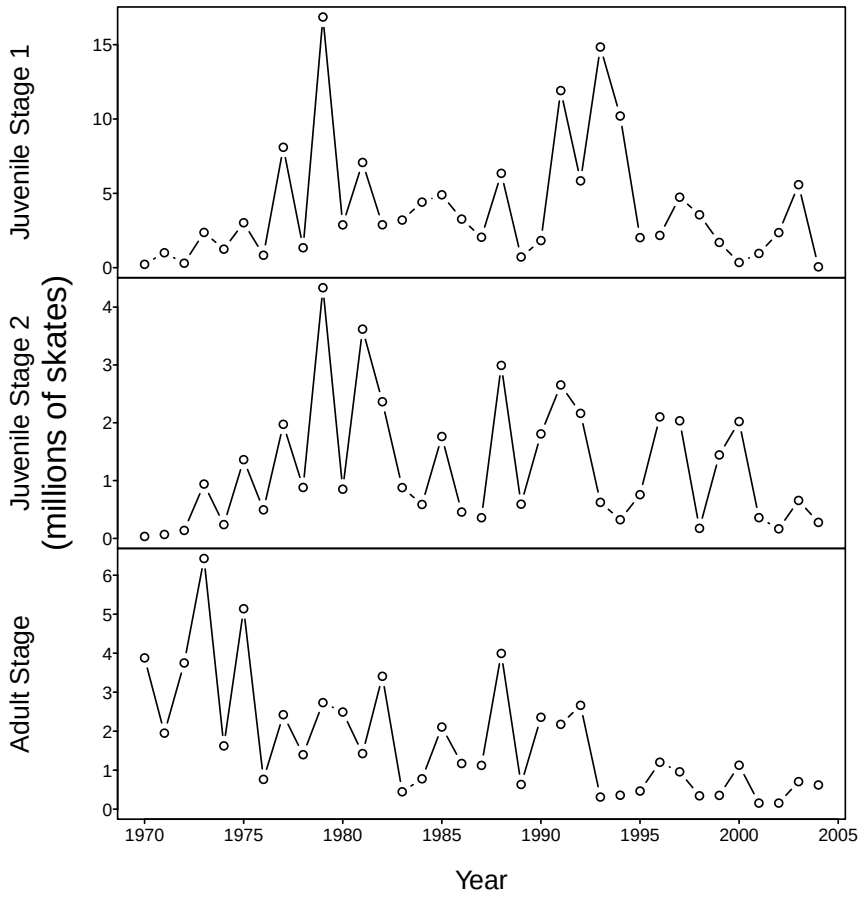

Figure 1: Survey abundance indices of three length classes of winter skate on the easter Scotian Shelf. Indices were scaled up to absolute abundance by catchability coefficients that were estimated outside the model. Note different scales on the y-axis. Data from Swain et al. (2009).

4 The Models

The model described here is a fully Bayesian state-space model and was implemented in both AD Model Builder with random effects (ADMB-RE) (Fournier et al., 2012) and JAGS (Plummer, 2003)

The population dynamics model for this species is divided into three stages, where the transition from small to larger stages occurs via a transition probability (θ_i), where i is the index for the stage. The state transition equation for the first stage is given by eq. 1, where $N_{1,t}$ is the number of individuals in stage 1 in year t, θ_1 is the transition probability from stage 1 to stage 2, r is the annual recruitment rate, $N_{3,t-a}$ is the number of stage-3 individuals in year lagged by a years to represent the time required to recruit to the first stage. The total mortality rate $Z_{1,d}$ by decade d which is the main parameter of interest.

$$N_{1,t} = \left[N_{1,t-1}(1-\theta_1) + \frac{1}{2}(rN_{3,t-a}) \right] e^{-Z_{1,d}} e^{\eta_{1,t}} \tag{1}$$

The state transition equations for stage two and three are given in eqs 2 and 3.

$$N_{2,t} = \left[N_{2,t-1}(1-\theta_2) + N_{1,t-1}\theta_1 \right] e^{-Z_{2,d}} e^{\eta_{2,t}} \tag{2}$$

$$N_{3,t} = \left[N_{3,t-1} + N_{2,t-1}\theta_2 \right] e^{-Z_{3,d}} e^{\eta_{3,t}} \tag{3}$$

where $N_{i,t}$ is the skate abundance (millions) for stage i in year t, $\theta_{1:2}$ is the transition probability between the first to second juvenile stage and second juvenile and adult age respectively, r is the recruitment rate, a accounts for the 5 year lag from the adult stage to entering the first juvenile stage, Z is the instantaneous mortality rate for each stage and each decadal period, d, and $\eta_{i,t}$ is an independent normal random variable for each stage and year with a mean of zero and process variance σ_i^2 to account for stochasticity in the population dynamics for each stage, $\sim LN(0, \sigma_i^2)$. All parameters are held constant with the exception of Z_i which is allowed to vary decadally.

The observation model relates the unobserved states, $N_{i,t}$ of the population model to the indices of abundance $y_{i,t}$, that are observed with error.

$$y_{i,t} = q_i N_{i,t} e^{\epsilon_{i,t}} \tag{4}$$

where $y_{i,t}$ is the scaled index of relative abundance of stage i at time t; q_i is the catchability coefficient for each stage i which describes the effectiveness of each unit of fishing effort; and $\epsilon_{i,t}$ is a normal random variable with a

mean of zero and variance τ_i^2 to account for error in the observations of index i, $\epsilon_{i,t} \sim LN(0, \tau_i^2)$.

The time series data is largely uninformatitve to many of the parameters in the model, therefore these data are supplied in the form of priors. The bayesian approach requires priors be used to supply additional external information to the model. Flat priors were used when prior information was not known about certain model parameters. Priors used in these analyses are summarized in Table 2.

Table 2: Summary of specified priors for Bayesian state-space model

Parameter	Prior
recrate (Recruitment rate)	log normal(1.18, 4)I(0.01,1000)
theta.1 (Transition probability 1)	beta(20, 80)
theta.2 (Transition probability 2)	beta(25, 75)
All z (Total mortality)	uniform(0, 3)
q (Catchability coefficient)	uniform(0,100)

5 AD Model Builder with random effects

ADMB has the benefit of being able to quickly obtain MLE of the parameters of interest compared to JAGS which must obtain the full stationary distribution profiles of all model parameters. This can result in a large time saving advantage of using ADMB over JAGS when analyzing large dataset and implimenting complex models. This feature of ADMB can be particularly useful during the model formulation and debugging phases of model development and full MCMC profiles can then be obtained once the model has reached a satisfactory level of development. Although this advantage is not a serious issue with the model explored here, this flexibility can potentially offset the generally larger time investment of formulating complex models in ADMB over those coded in JAGS.

We ran the ADMB -RE model in both MLE and MCMC modes. To implement MCMC in ADMB you use the arguments -mcmc2 *N* -mcsave *N2* commands to run the model with a chain of length *N* and save the output every *N2* steps. We use the argument -mcmc2 to generate a chain on the joint vector of hyper-parameters and random effects. A key difference between JAGS and ADMB is that ADMB does not begin sampling the

MCMC chains until a maximum likelihood estimate has been identified and therefore presumably a shorter burn-in period is required. The length of burn-in, however, is dependent upon model complexity and identifiability of parameter estimates. We therefore, ran ADMB for the same number of iterations and used the same burn-in as the JAGS model.

5.1 ADMB-RE Results: Maximum Likelihood estimates

The ADMB-RE MLE model obtained parameter estimates the most quickly and only required 13 seconds to run.

	Estimate	Std. Error	z value	Pr(>|z|)
recrate	3.11	0.01	350.29	0.00
theta.1	0.19	0.06	3.03	0.00
theta.2	0.25	0.06	3.91	0.00
z.juv1.1	1.24	0.12	10.77	0.00
z.juv1.2	0.56	0.13	4.50	0.00
z.juv1.3	0.52	0.10	5.12	0.00
z.juv2.1	0.28	0.16	1.76	0.08
z.juv2.2	0.29	0.15	2.03	0.04
z.juv2.3	0.47	0.11	4.44	0.00
z.adult.1	0.12	0.12	0.93	0.35
z.adult.2	0.22	0.11	2.04	0.04
z.adult.3	0.45	0.07	6.08	0.00

Table 3: Maximum likelihood estimates (MLEs) and associated standard errors of the recruitment rate (recrate), transition probability between size stages (theta 1 and 2) and mortality estimates for all stages and time periods. Subscripts to mortality parameter estimates ($Z_{s,d}$), are subscripted by stage and decade

5.2 ADMB Results: MCMC

The ADMB MCMC model took substantially longer than the ADMB-RE MLE model and finished in 61.6 seconds. Because ADMB only runs one MCMC chain we used Geweke's diagnostic to assess the convergence to a stationary distribution of the key model parameters (Figs 2 & 3). The existence of some Geweke z-scores outside 2 standard deviations of zero suggests that there may be some convergence issues with some of the parameters and the model may require a longer burn-in period to achieve satisfactory model convergence. The Geweke's convergence diagnostic is available in the R coda package (Plummer et al., 2006) and convergence failure is identified by a large number of Z-scores falling outside the 95% confidence intervals.

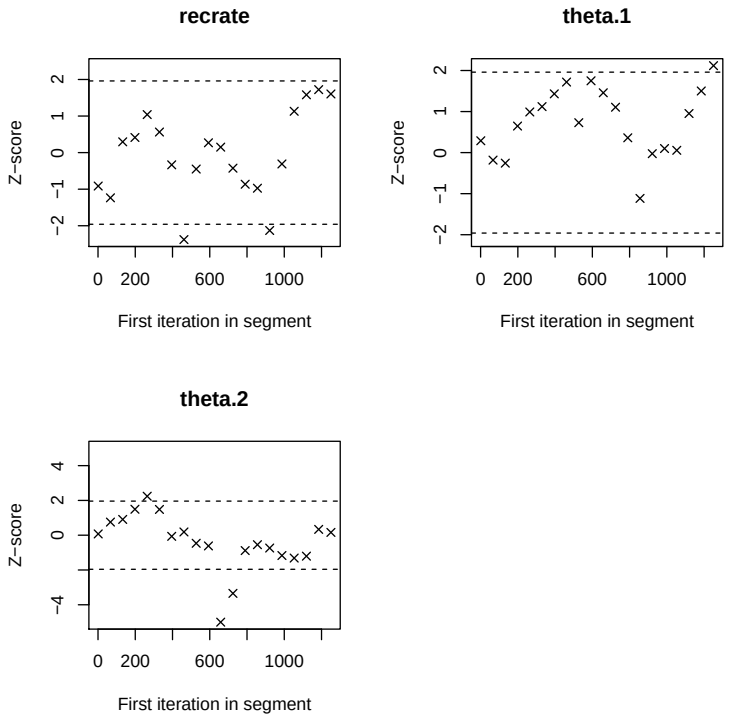

Figure 2: Geweke diagnostoc of select model paramters of ADMB-RE MCMC. Dashed lines identify 2 standard deviation boundaries around zero.

We examined trace plots of samples of select model parameters as another measure to assess model convergence (Fig. 4). Examination of the

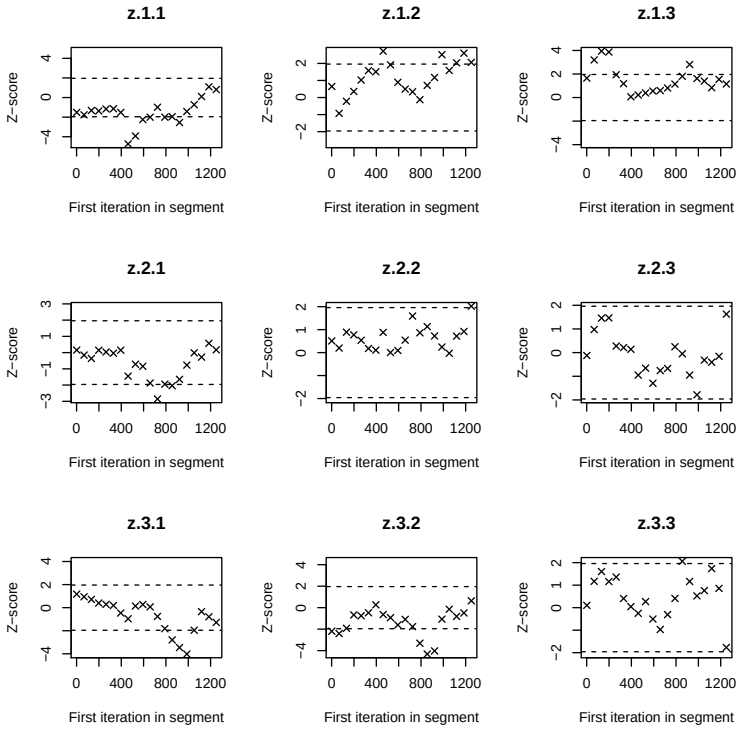

Figure 3: Geweke diagnostic of select model paramaters of ADMB-RE MCMC. Dashed lines identify 2 standard deviation boundaries around zero.

trace plots suggests that there is substantial autocorrelation between samples even though we specified a thinning for every 30 samples. This suggests that further burn-in and thinning may be necessary to achieve model convergence. We did, however, increase burn-in to 225 000 iterations and substantial autocorrelation was still present in the trace plots.

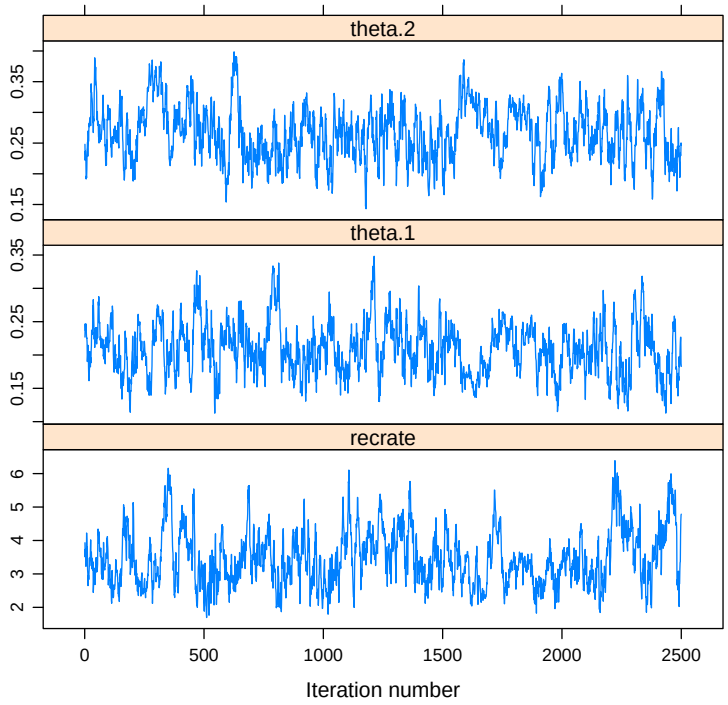

Figure 4: Traceplots of select ADMB-RE mcmc model parameters

	Estimate	Std.Error	50%	2.5%	97.5%
recrate	3.11	0.77	3.36	2.15	5.42
theta.1	0.19	0.04	0.21	0.14	0.29
theta.2	0.25	0.04	0.27	0.19	0.36
z.1.1	1.24	0.39	1.32	0.57	2.06
z.1.2	0.28	0.29	0.32	0.03	0.90
z.1.3	0.12	0.15	0.15	0.01	0.44
z.2.1	0.56	0.31	0.64	0.10	1.25
z.2.2	0.29	0.27	0.37	0.03	0.93
z.2.3	0.22	0.17	0.23	0.02	0.56
z.3.1	0.52	0.24	0.55	0.12	1.03
z.3.2	0.47	0.24	0.51	0.09	1.07
z.3.3	0.45	0.16	0.46	0.17	0.86

Table 4: Parameter estimates of select parameters, associated standard errors, and quantiles of ADMB MCMC results. Subscripts to total mortality parameter estimates estimates ($Z_{s,d}$), are subscripted by stage and decade

6 BUGS (using JAGS)

The JAGS model required 174.2 seconds to run.

There are a variety of methods to gauge convergence of an MCMC chain (Cowles and Carlin, 1996). Although JAGS can run multiple chains which allows for a variety of convergence diagnostics, ADMB can only run single chains and we therefore also ran the JAGS model with a single chain to maintain consistency between the two platforms. We also used the Geweke's convergence diagnostic and trace plots to gauge convergence.

Examination of the Geweke's diagnostic plots shows much better convergence of the JAGS model parameters (Figures 5 & refjags:geweke2 compared to the ADMB-RE model for the majority of parameters examined. Further, trace plots 7 shows good mixing of the select model parameters which is indicative that a stationary posterior distribution had been reached.

```
> jags.df <- as.data.frame(tfit_jags_mcmc)
> jags.parms1 <- c("deviance", "recrate",  "theta[1]", "theta[2]")
> sub.jags.df1 <- subset(jags.df,select=jags.parms1)
> geweke.plot(as.mcmc(sub.jags.df1))
```

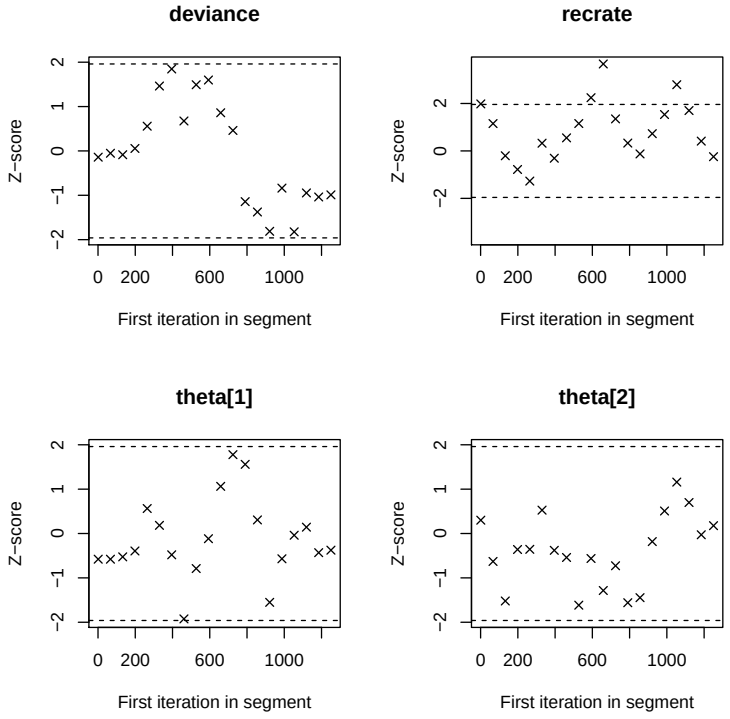

Figure 5: Geweke plots of select model parameters for model run in JAGS. Dashed lines are 95% confidence regions.

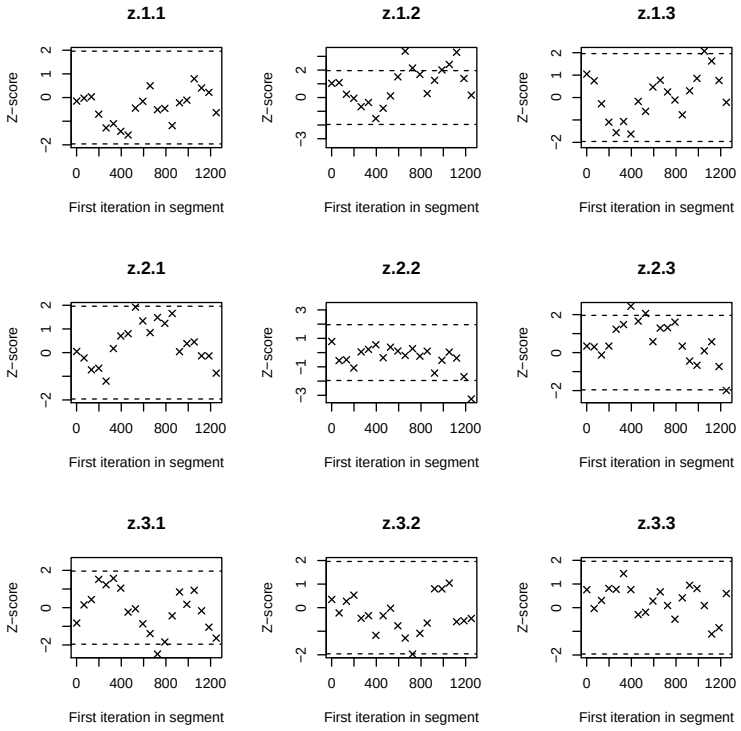

Figure 6: Geweke plots of decadal mortality estimates [stage,decade] for model run in JAGS. Dashed lines are 95% confidence regions.

```
> xyplot(as.mcmc(sub.jags.df1))
```

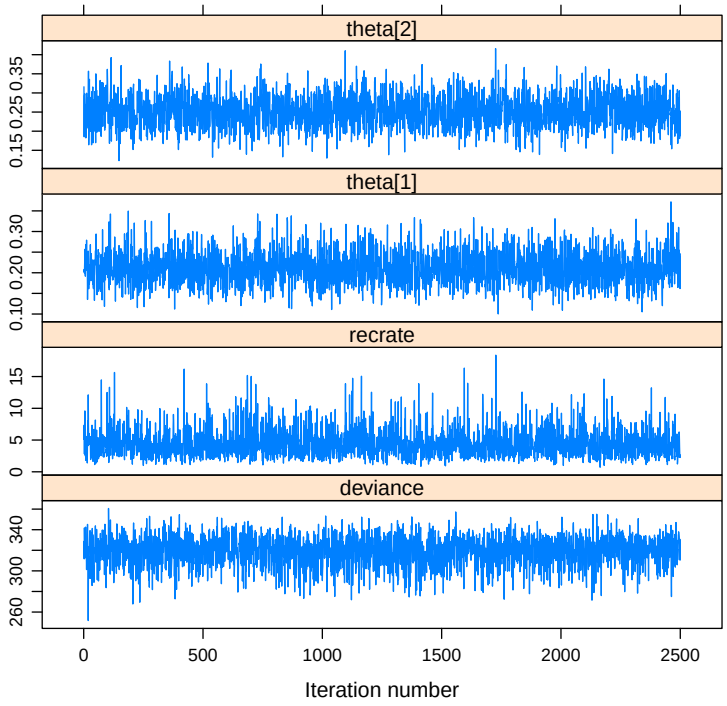

Figure 7: Traceplots of select JAGS model parameters after burning of 100000 iterations (single chain)

6.1 JAGS Results

Summary of JAGS results are as follows:

	Mean	SD	Naive SE	Time-series SE	50%	2.5%	97.5%
deviance	319.08	14.77	0.30	0.29	320.48	285.85	343.96
recrate	4.49	2.24	0.04	0.04	4.07	1.57	9.94
theta.1	0.21	0.04	0.00	0.00	0.21	0.14	0.30
theta.2	0.25	0.04	0.00	0.00	0.25	0.17	0.34
z.1.1	1.48	0.54	0.01	0.01	1.47	0.43	2.59
z.2.1	0.72	0.38	0.01	0.01	0.68	0.10	1.59
z.3.1	0.61	0.30	0.01	0.01	0.58	0.11	1.28
z.1.2	0.43	0.30	0.01	0.01	0.37	0.03	1.13
z.2.2	0.42	0.26	0.01	0.01	0.40	0.04	1.04
z.3.2	0.46	0.22	0.00	0.00	0.44	0.07	0.96
z.1.3	0.14	0.09	0.00	0.00	0.13	0.01	0.37
z.2.3	0.22	0.13	0.00	0.00	0.20	0.02	0.51
z.3.3	0.43	0.13	0.00	0.00	0.42	0.19	0.71

Table 5: Select Parameter and associated standard deviation from JAGS. Subscripts to total mortality parameter estimates estimates ($Z_{s,d}$), are subscripted by stage and decade

7 Comparison of ADMB-RE MCMC and JAGS bayesian posteriors

Posterior distributions of select model parameters of the ADMB-RE MCMC and JAGS models are compared in Fig 8. Posterior distributions between the two modelling platforms were similar, however, the suspect convergence diagnostics of the ADMB-RE MCMC model may be an explanation for the descrepencies between the two approaches.

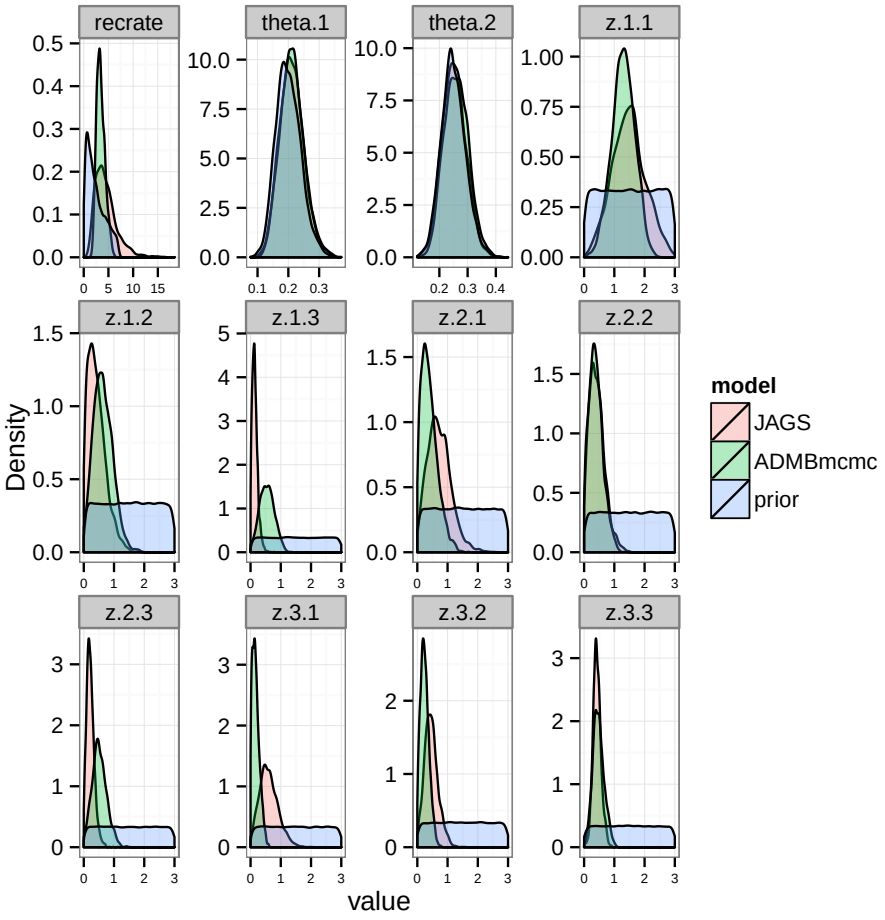

Figure 8: Comparison of posterior and prior densities of select model parameters from ADMB-RE Markov Chain Monte Carlo (MCMC) and JAGS analysis.

Time series predictions of the three modelling approaches indicate all three give similar estimates on population abundance through time (Fig. 9).

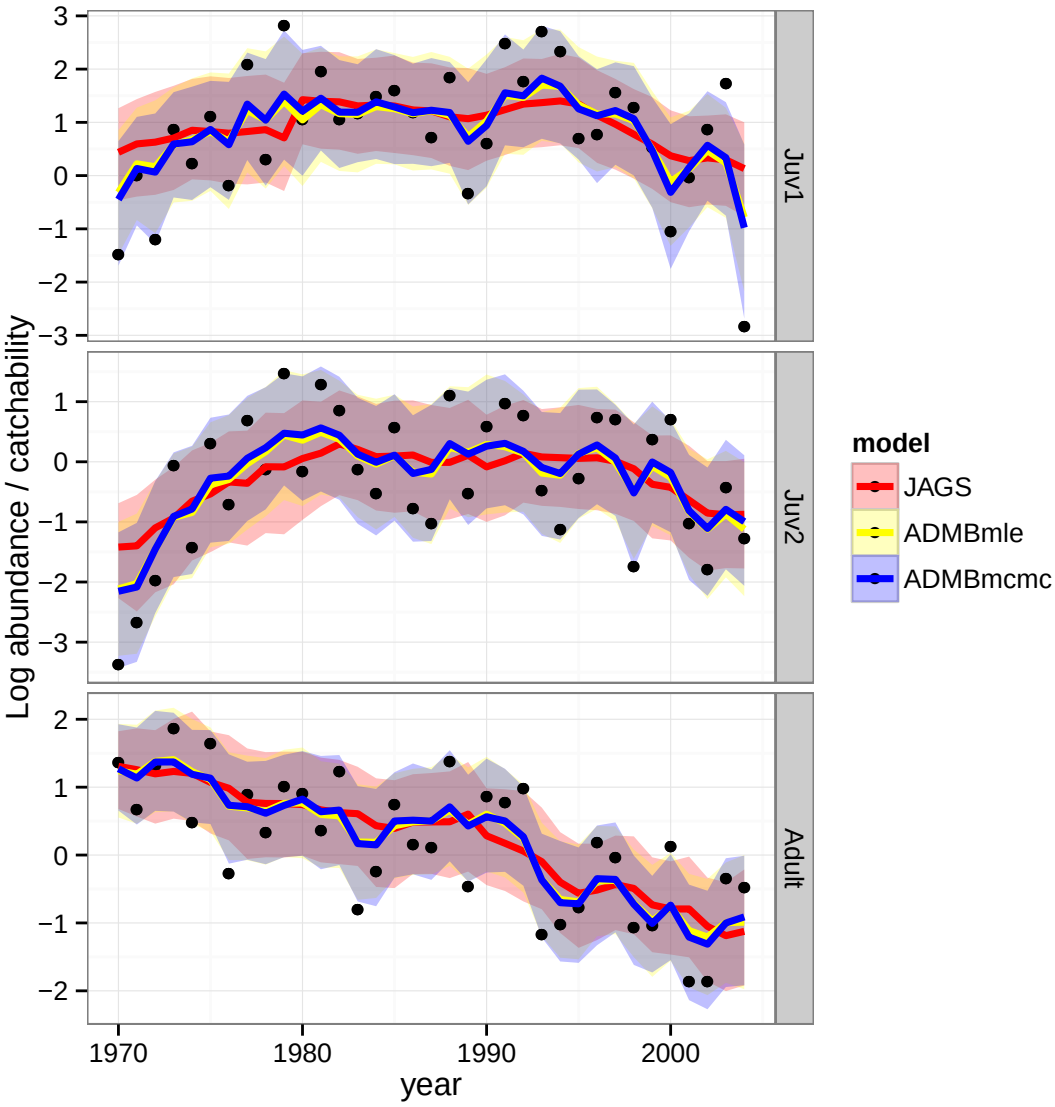

Figure 9: Comparison of the three model fits by JAGS, ADMB-RE maximum likelihood and ADMB-RE MCMC. Fit lines and 95% confidence limits are estimated abundance for each stage divided by catchability (logged).

8 Simulation results

We used a simulation approach to assess the ability of each modelling platform to accurately and precisely estimate model parameters. We simulated 25 datasets using fixed parameter values with error added in the form of both process and observation error (Table 7). Data was simulated in **R** and identical data sets were given to ADMB and JAGS for each simulation. ADMB-RE was run in MLE mode only for these tests.

JAGS appeared to be marginally better at estimating the mortality parameters in comparison to ADMB (Figure 10). The JAGS model, however, appeared to have a large bias in both transition probability parameters and consistently overestimated recruitment rate (Fig. 11).

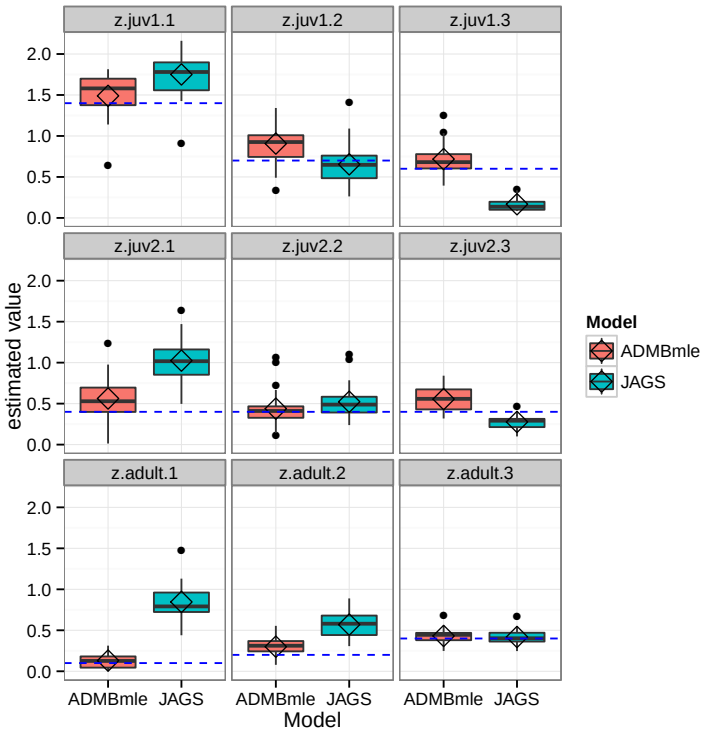

Figure 10: Simulation results of select parameters from obtained using JAGS and ADMB maximum likelihood estimation. Subscripts to total mortality parameter estimates estimates ($Z_{s,d}$), are subscripted by stage and decade. Diamonds are mean of the estimates and dashed blue line is the true value of each parameter.

.

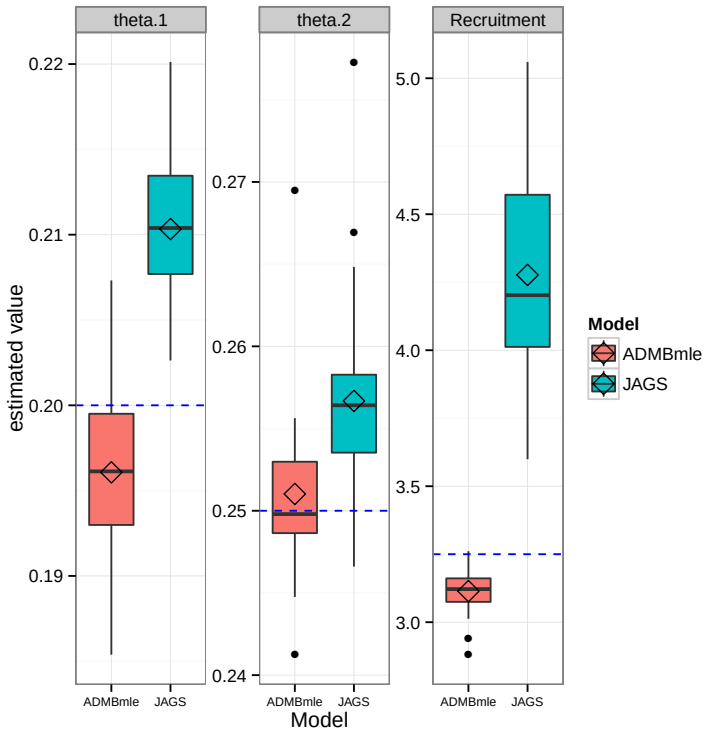

Figure 11: Simulation results of select parameters from obtained using JAGS and ADMB maximum likelihood estimation. Diamonds are mean of the estimates and dashed blue line is the true value of each parameter.

9 Data

Table 6: Winter skate (*Leucoraja ocellata*) abundance data from Swain et al. (2009), *Ecol. App.* Data are millions of individuals (both sexes)

Year	Juvenile stage 1	Juvenile stage 2	Adult stage
1970	0.226	0.034	3.880
1971	1.005	0.069	1.949
1972	0.300	0.139	3.750
1973	2.370	0.940	6.429
1974	1.247	0.239	1.622
1975	3.022	1.362	5.140
1976	0.837	0.494	0.764
1977	8.099	1.973	2.422
1978	1.341	0.881	1.397
1979	16.862	4.335	2.732
1980	2.880	0.851	2.491
1981	7.078	3.619	1.425
1982	2.887	2.364	3.407
1983	3.203	0.879	0.445
1984	4.409	0.587	0.778
1985	4.903	1.762	2.107
1986	3.268	0.456	1.171
1987	2.049	0.358	1.122
1988	6.347	2.992	3.994
1989	0.719	0.592	0.633
1990	1.825	1.808	2.357
1991	11.914	2.654	2.175
1992	5.842	2.163	2.663
1993	14.844	0.624	0.311
1994	10.202	0.323	0.357
1995	2.021	0.757	0.464
1996	2.170	2.102	1.203
1997	4.742	2.034	0.957
1998	3.559	0.174	0.342
1999	1.700	1.444	0.353
2000	0.351	2.021	1.127
2001	0.957	0.359	0.155
2002	2.360	0.166	0.155
2003	5.584	0.657	0.706
2004	0.059	0.277	0.620

Table 7: Fixed parameter values used to generate simulated data for comparison between software platforms.

Parameter	Juvenile stage 1	Juvenile stage 2	Adult stage
catchability scaling parameter (q)	1.0	1.0	1.0
recruitment rate (r)	3.2		
stage transition prob. (θ)		0.2	0.2
process error std. dev. (σ)	0.3	0.4	0.2
observation error std. dev. (τ)	1.0	0.9	0.8
inst. total mortality ($Z_{1970\text{-}1979}$)	1.4	0.4	0.1
inst. total mortality ($Z_{1980\text{-}1989}$)	0.7	0.4	0.2
inst. total mortality ($Z_{1990\text{-}2004}$)	0.6	0.4	0.4
adult abundance$_{1965-1969}$ (millions)			4.0
juvenile abundance$_{1969}$ (millions)	1.0	0.2	

10 ADMB Code

The AD Model Builder template code (`skate.tpl`)

```
DATA_SECTION
        init_int nobs;
        init_int a;             //age-at-maturity
        init_matrix data(1,nobs,0,3);
        matrix Iobs(1,3,1,nobs);
        ivector iyr(1,nobs);
        LOC_CALCS
                dmatrix tmp=trans(data).sub(1,3);
                Iobs=tmp;
                iyr=ivector(column(data,0));
        END_CALCS
        matrix sim_nu(1,3,1,nobs);
        !! sim_nu=0;
        //!! cout<<data<<endl;

PARAMETER_SECTION
        init_vector logq(1,3,1);
        init_bounded_matrix z(1,3,1,3,0.00,5.0,1);
        //Changed estimates of variance to the inverse to
        //improve mixing properties (a al. WinBugs)
        init_bounded_vector invSDpro(1,3,0.04,800,1);
        init_bounded_vector invSDobs(1,3,0.25,4,1);
        init_bounded_vector theta(1,2,0,1,1);
        init_bounded_number recrate(0.01,1000,1);
        init_vector logNad(1,a,1);
        init_vector logN69(1,2);
        //init_bounded_matrix nu(1,3,1,35,-5,5,2); // No Random effects
        random_effects_matrix nu(1,3,1,35,2); // for Random effects

        objective_function_value f;

        vector q(1,3);
        vector N69(1,2);
        vector Nad(1,a);
        vector sdpro(1,3);
        vector sdobs(1,3);
```

```
37
38          matrix st(1,3,1,35);
39          matrix nmod(1,3,1,35);
40          matrix logn(1,3,1,35);
41          matrix N(1,3,1,35);
42      // sdreport_matrix N(1,3,1,35);
43      sdreport_matrix sd_N(1,3,1,35);
44          matrix epsilon(1,3,1,35);
45          sdreport_number sd_var;
46      sdreport_matrix Ipred(1,3,1,35);
47
48  PROCEDURE_SECTION
49
50          calculate_process_model2();
51          calculate_observation_model();
52          calculate_objective_func();
53          sd_var=q(1);
54          sd_N = N;
55          for(int s=1; s<=3; s++){
56             Ipred(s) = logq(s)+log(N(s));
57            // Ipred(s) = exp(log(q(s)) + log(N(s)) + epsilon(s) );
58            //Ipred(s) = exp( log(Iobs(s)) - epsilon(s) );
59          }
60
61  FUNCTION calculate_process_model2
62          int s,y,d; // stage / year / decade
63          dvariable Adults;
64          q      = mfexp(logq);
65          N69    = mfexp(logN69);
66          Nad    = mfexp(logNad);
67          sdpro = sqrt(1./invSDpro);
68          sdobs = sqrt(1./invSDobs);
69          N(1,1) = (N69(1)*(1.-theta(1))+0.5*recrate*Nad(1))*exp(-z(1,1)+nu(1,1)*
                sdpro(1));
70          N(2,1) = (N69(2)*(1.-theta(2))+N69(1)*theta(1))*exp(-z(2,1)+nu(2,1)*sdpro
                (2));
71          N(3,1) = (N69(2)*theta(2)+Nad(a))*exp(-z(3,1)+nu(3,1)*sdpro(3));
72
73          d=0;
74          for(y=1;y<nobs;y++) /* Update d index for each decade */
75          {
76                  if( !((y-1)%10) && d<3) d++;
77                  if(y<a) /* Get adults for new recruitment */
78                  {
79                          Adults = Nad(y+1);
80                  }
81                  else
82                  {
83                          Adults = N(3,y-a+1);
84                  }
85                  /* Update numbers at each stage */
86                  N(1,y+1) = (N(1,y)*(1.-theta(1))+0.5*(recrate*Adults))
87                                          * exp(-z(1,d)+nu(1,y+1)
88                                          * sdpro(1)+sim_nu(1,y+1));
89
90                  N(2,y+1) = (N(2,y)*(1.-theta(2))+N(1,y)*theta(1))
91                                          * exp(-z(2,d)+nu(2,y+1)
92                                          * sdpro(2)+sim_nu(2,y+1));
93
94                  N(3,y+1) = (N(3,y) + N(2,y)*theta(2))
95                                          * exp(-z(3,d)+nu(3,y+1)
96                                          * sdpro(3)+sim_nu(3,y+1));
97          }
98
99  FUNCTION calculate_observation_model
100         int s,y;
101
102         for(s=1;s<=3;s++)
103         {
104                 epsilon(s) = log(Iobs(s)) - log(q(s)*N(s));
105         }
106
107 FUNCTION calculate_objective_func
108         dvar_vector fvec(1,10);
109         fvec.initialize();
110
```

```
111         /* Priors */
112         int s;
113         for(s=1; s<=3; s++)
114         {
115                 fvec(1)+=dlnorm(q(s),0.0,0.1);
116         }
117         fvec(2)=dbeta(theta(1),20,80);
118         fvec(2)+=dbeta(theta(2),25,75);
119         fvec(3)=dlnorm(N69(1),0.,0.4);
120         fvec(3)+=dlnorm(N69(2),-1.203973,0.4);
121         for(s=1; s<= a;s++)
122         {
123                 fvec(4)+=dlnorm(Nad(s),log(3.5), 0.3);
124         }
125         fvec(5)=dlnorm(recrate,log(3.25),0.25);
126
127         /* Likelihood of the survey data*/
128         for(s=1;s<=3;s++)
129         {
130                 fvec(6) += dnorm(epsilon(s),sdobs(s));
131                 fvec(8)+=dgamma(invSDobs(s),1.01,1.01);
132                 fvec(9)+=dgamma(invSDpro(s),1.01,1.01);
133         }
134
135         /* Random effects */
136         for(s=1;s<=3;s++)
137         {
138                 fvec(7) += norm2(nu(s));
139         }
140         f= sum(fvec);
141
142 GLOBALS_SECTION
143         #include <statslib/statsLib.h> // '#' preprocess command - put this in
                before compiling starts
144         #undef REPORT
145         #define REPORT(object) report << #object "\n" << object << endl;
146
147 REPORT_SECTION
148         REPORT(q);
149         //REPORT(sdobs);
150         //REPORT(sdpro);
151         REPORT(Iobs);
152         REPORT(N);
153         REPORT(Ipred);
154         //REPORT(epsilon);
155         REPORT(z);
156         REPORT(theta);
157         REPORT(recrate);
158
159 TOP_OF_MAIN_SECTION
160         arrmblsize = 50000000;
161         gradient_structure::set_GRADSTACK_BUFFER_SIZE(1.e7);
162         gradient_structure::set_CMPDIF_BUFFER_SIZE(1.e7);
163         gradient_structure::set_MAX_NVAR_OFFSET(11158);
164         gradient_structure::set_NUM_DEPENDENT_VARIABLES(5000);
```

11 BUGS Code

The JAGS (BUGS) template code (`skate_bugs.txt`)

```
1
2  model
3  {
4
5  for (s in 1:3) {
6    q[s] ~ dlnorm(0, 100)  # Informative Priors equivalent to q=1 (1/trawlable_
         units) with sd=0.1
7      for (d in 1:3) {
8        z[s,d] ~ dunif(0, 3)    # Priors for instantaneous mortality rates with
             decadal variation
9                      }
10   sdpro[s] ~ dunif(0, 5) # Process error variance
11   isigma2[s] <- pow(sdpro[s], -2)
12       }
13
14 # Priors for observation error variance
15   sdobs[1]~dunif(0.497,2)
16   sdobs[2]~dunif(0.586,2)
17   sdobs[3]~dunif(0.513,2)
18
19 for(i in 1:3){
20     itau2[i] <- pow(sdobs[i], -2)
21       }
22
23 # Priors (informative) for transistion probabilities
24   theta[1] ~ dbeta(20, 80)
25   theta[2] ~ dbeta(25, 75)
26
27 # Prior for recruitment rate
28   recrate ~ dlnorm(1.178655, 4)I(0.01,1000)
29
30 # Informative priors for inital abundance (based on trawlable abundance in 1000s)
31 # Nad[1] is adult abundance in 1965
32   N69[1] ~ dlnorm(0, 6.25)
33   N69[2] ~ dlnorm(-1.203973, 6.25)
34
35 for (i in 1:5) {Nad[i] ~ dlnorm(1.252763, 11.11111)}
36     # Process equation
37     for (s in 1:3) {
38       for (t in 1:10 ) {
39           st[s,t] <- exp(-z[s,1])
40           st[s,t+10] <- exp(-z[s,2])
41           st[s,t+20] <- exp(-z[s,3])
42                       }
43             for(t in 31:35){
44               st[s,t] <- exp(-z[s,3])
45                           }
46                   }
47
48   nmod[1,1] <- (Nad[1] * (recrate / 2) + N69[1] * (1 - theta[1]))* st[1,1]
49   nmod[2,1] <- (N69[1] * theta[1] + N69[2] * (1 - theta[2]))* st[2,1]
50   nmod[3,1] <- (N69[2] * theta[2] + Nad[5])* st[3,1]
51
52 for (s in 1:3) {
53   logn[s,1] <- log(nmod[s,1])
54   N[s,1] ~ dlnorm(logn[s,1], isigma2[s])
55                 }
56
57   nmod[1,2] <- (Nad[2] * (recrate / 2) + N[1,1]  * (1 - theta[1]))* st[1,2]
58   nmod[2,2] <- (N[1,1]  * theta[1] + N[2,1] * (1 - theta[2]))* st[2,2]
59   nmod[3,2] <- (N[2,1]  * theta[2] + N[3,1])* st[3,2]
60
61 for (s in 1:3) {
62   logn[s,2] <- log(nmod[s,2])
63   N[s,2] ~ dlnorm(logn[s,2], isigma2[s])
64                 }
65
66 for (t in 3:5) {
67   nmod[1,t] <- (Nad[t] * (recrate / 2) + N[1,t-1]  * (1 - theta[1]))* st[1,t]
```

```
  nmod[2,t] <- (N[1,t-1]  * theta[1] + N[2,t-1]  * (1 - theta[2]))* st[2,t]
  nmod[3,t] <- (N[2,t-1]  * theta[2] + N[3,t-1])* st[3,t]
       for (s in 1:3) {
         logn[s,t] <- log(nmod[s,t])
        N[s,t] ~ dlnorm(logn[s,t], isigma2[s])
                    }
                  }

for (t in 6:35 ) {
       nmod[1,t] <- (N[3,t-5] * (recrate / 2) + N[1,t-1]  * (1 - theta[1]))* st[1,
           t]
       nmod[2,t] <- (N[1,t-1]  * theta[1] + N[2,t-1]  * (1 - theta[2]))* st[2,t]
       nmod[3,t] <- (N[2,t-1]  * theta[2] + N[3,t-1])* st[3,t]

       for (s in 1:3) {
             logn[s,t] <- log(nmod[s,t])
             N[s,t] ~ dlnorm(logn[s,t], isigma2[s])
                    }
                   }

# Observation equation
for (t in 1:35 ) {
  for (s in 1:3) {
    Im[s,t] <- q[s] * N[s,t]
    logIm[s,t] <- log(Im[s,t])
    logIpred[s,t] <- logn[s,t]-log(q[s])
    Iobs[s,t] ~ dlnorm(logIm[s,t], itau2[s])
                  }
                  }
} # End model
```

12 Simulation Code

R simulation code skate_sims_run.R)

```
sim.data <- function(seed){

  if (!missing(seed)) set.seed(seed)

  q <- c(1.0,1.0,1.0)
  r <- 3.25
  theta <- c(0.20,0.25)
  sdpro <- c(0.35,0.40,0.20)
  sdobs <- c(1.0,0.9,0.75)
  ## Make z matrix for all years to make coding easy
  z.juv1 <- c(rep(c(1.4,0.7),each=10),rep(0.6,15))
  z.juv2 <- rep(0.4,35)
  z.adult <- c(rep(c(0.1,0.2),each=10),rep(0.4,15))
  z <- matrix(c(z.juv1,z.juv2,z.adult),nrow=3,ncol=35,byrow=TRUE)

  N.matrix <- data.frame(matrix(NA,ncol=40,nrow=3))
  names(N.matrix) <- paste('X',seq(1965,2004,1),sep='')

  N.matrix.1969 <- c(1.0,0.2,4.0)
  N.matrix$X1969 <- N.matrix.1969
  N.matrix[3,1:5] <- 4.0 # 4 million skates to start matrix calcs

  for (j in 6:40){
    N.matrix[1,j] <-  (N.matrix[3,j-5] * (r/2) + N.matrix[1,j-1] * (1 - theta[1])
        ) * exp(-z[1,j-5]) # Deterministic
    N.matrix[1,j] <- N.matrix[1,j]*exp(rnorm(1,0,sdpro[1]) - 0.5*sdpro[1]^2) ####
        ADD PROCESS ERROR

    N.matrix[2,j] <-  (N.matrix[1,j-1]  * theta[1] + N.matrix[2,j-1]  * (1 -
        theta[2]))* exp(-z[2,j-5])
    N.matrix[2,j] <- N.matrix[2,j]*exp(rnorm(1,0,sdpro[1]) - 0.5*sdpro[2]^2)

    N.matrix[3,j] <- (N.matrix[2,j-1]  * theta[2] + N.matrix[3,j-1])* exp(-z[3,j
        -5])
    N.matrix[3,j] <- N.matrix[3,j]*exp(rnorm(1,0,sdpro[3]) - 0.5*sdpro[3]^2)
  }
  N.matrix <- N.matrix[,-seq(1,5,1)]
  N.obs <- N.matrix
  for (i in 1:nrow(N.obs)){
    for (j in 1:ncol(N.obs)){
      N.obs[i,j] <- N.obs[i,j]*exp(rnorm(1,0,sdobs[i]) - 0.5*sdobs[i]^2)
    }}
  final.dat <- data.frame(year=seq(1970,2004,1),t(N.obs))
  names(final.dat) <- c('year','juv1','juv2','adult')

### set up true.dat file
  true.dat <- data.frame(q1=q[1],q2=q[2],q3=q[3],Recruitment=r,theta.1=theta[1],
      theta.2=theta[2],sdpro1=sdpro[1],sdpro2=sdpro[2],sdpro3=sdpro[3],sdobs1=
      sdobs[1],sdobs2=sdobs[2],sdobs3=sdobs[3],z.juv1.1=unique(z.juv1)[1],z.juv1
      .2=unique(z.juv1)[2],z.juv1.3=unique(z.juv1)[3],z.juv2.1=unique(z.juv2),z.
      juv2.2=unique(z.juv2),z.juv2.3=unique(z.juv2),z.adult.1=unique(z.adult)[1],
      z.adult.2=unique(z.adult)[2],z.adult.3=unique(z.adult)[3],N.juv1.1969=N.
      matrix.1969[1],N.juv2.1969=N.matrix.1969[2],N.adult.1969=N.matrix.1969[3])

true.dat2 <- as.data.frame(t(true.dat))
names(true.dat2) <- 'value'
return(list(final.dat,true.dat2))
}
final.dat <- sim.data()[[1]]
true.dat <- sim.data()[[2]]
#===== data files
  dat.name <- 'skate.dat'
  bugdat.name <- 'skate_bugs.dat'
  true.dat.name <- 'true.dat'
  write.table(final.dat,row.names=F, col.names=T,  file=bugdat.name)
  write.table(true.dat,row.names=T, col.names=T,  file=true.dat.name)

  unlink(dat.name)
  zz <- file(description = dat.name, open = "w", blocking = TRUE, encoding =
      getOption("encoding"))
  cat( "# number of years of data", file = zz, sep = "\n")
```

```
61    cat(nrow(final.dat), file = zz, sep = "\n")
62    cat("# Recruitment lag", file = zz, sep = "\n")
63    cat(5, file = zz, sep = "\n")
64    cat('#Data', file = zz, sep = "\n")
65    write.table(final.dat,row.names= FALSE, col.names=FALSE, append= TRUE, file=dat
          .name)
```

References

Cowles, M. and B. Carlin (1996). Markov chain monte carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, 883–904.

Fournier, D. A., H. J. Skaug, J. Ancheta, J. Ianelli, A. Magnusson, M. N. Maunder, A. Nielsen, and J. Sibert (2012). AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. *Optimization Methods & Software 27*(2), 233–249.

Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20–22, Vienna, Austria.

Plummer, M., N. Best, K. Cowles, and K. Vines (2006). Coda: Convergence diagnosis and output analysis for mcmc. *R News 6*(1), 7–11.

Swain, D., I. Jonsen, J. Simon, and R. Myers (2009). Assessing threats to species at risk using stage-structured state-space models: mortality trends in skate populations. *Ecological Applications 19*(5), 1347–1364.