Hi all, for the follow-up to question 2, where we are estimating models including process error, I am sending along a Stan program at the bottom of this email that should have what you need.

As a reminder, simulate multiple trajectories (my advice is 4-6) with fixed values for the parameters (r,K,dispersion constant, observation error), and fit to the following model. Try out a model with a) only process error, b) process plus observation error, and c) process plus observation error (with observation error fixed at “true” value). Make sure you have clearly saved the values used to generate the simulated data, and have plotted the simulated data.

For each version, in addition to plotting the model fit and histograms for inference on coefficients (as done in class), collect information about the sampling by saving an warning messages, and inspecting the MCMC by: a) use of pairs(model, pars = c()) for the 4 critical parameters, b) traceplot(model, pars = c()) for the same critical parameters. Play around with some options in specifying the priors.

Here is the call to Stan I use : logistic\_ss5b <- stan(file = "logistic5\_Bayes.stan", data = stan\_data5, chains = 6, iter = 2000)

Note that you can adjust the number of chains and iterations.

We will discuss all this, and fit some different models in class on Friday. **At this point, I would also recommend that you think about a possible application of these state-space models to a research question and data of interest to you!**

Best,

Chris

---------------

Stan Model:

data{

  int<lower = 0> N;

  int<lower = 0> M;

  int<lower=0> KK;

  int<lower = 0> times[KK];

  real y[M,KK];

}

parameters{

  real<lower = 0> r;

  real<lower = 0> K;

  real<lower = 0, upper = 2> k;

  real<lower = 0> obs\_error;

  vector<lower = 0>[M] mu\_init;

  real<lower = 0> l[M,N];

}

model{

  real mu[M,N];

   obs\_error ~ normal(0,10);

   K ~ normal(0,100);

   r ~ normal(0, 1);

   k ~ normal(0, 1);

  // kludgy fix for now

  mu\_init ~ normal(5,5);

  // tricky way of iterating through latent variable

// initial state

  for(i in 1:M){

  l[i,1] ~ gamma(mu\_init[i]/k,1/k);

  mu[i,1] = mu\_init[i];

  }

// rest of states

for(j in 1:M){

  for(i in 2:N){

    mu[j,i] = l[j,i-1] + r\*l[j,i-1]\*(1 - l[j,i-1]/K);

    l[j,i] ~ gamma(mu[j,i]/k,1/k);

  }

}

  // Note that we are constraining latent variable

  // with observations at specific times

  for(j in 1:KK){

    y[,j] ~ normal(mu[,times[j]],obs\_error);

  }

}