

1 Statistics for dynamic modelling

Statistical Model: a mathematical recipe for generating data. It typically has a structure and some *parameters* θ , which are not completely known.

Dynamic model: a model based on a process that evolves in time.

Statistics, tries to answer the questions:

1. Does a model fit data (for some θ)?
2. What values of θ are consistent with data.?
3. Does one model better fit the data than another?

To do statistics we need to be able to define fit/consistency.

Dynamic models can be statistically challenging, because their behaviour can change very abruptly as θ changes. But we'll start with smooth models.

1.1 Likelihood

In principle a model specifies $f_\theta(y)$: the *probability density function* (pdf) of data, y , given parameters θ .

We can assess the probability (density) of the observed y for any θ by plugging θ and y into $f_\theta(y)$.

Basic likelihood idea: θ values which make the observed data appear probable are more *likely* to be correct than those that make it appear improbable.

$f_\theta(y)$ with observed y plugged in, considered as a function of θ , is the *likelihood* of θ , often written $L(\theta)$. High $L(\theta)$ implies greater consistency of θ with data than low $L(\theta)$. i.e. better fit.

1.2 Maximum Likelihood Estimation

Maximum likelihood estimation: estimate θ as the maximizer of $L(\theta)$ to give $\hat{\theta}$.

Invariance: if g is a function, the MLE of $g(\theta)$ is $g(\hat{\theta})$.

In practice work with the log-likelihood $l(\theta) = \log(L)$ — $\hat{\theta}$ maximizes this too, but l is more convenient.

Now consider $\hat{\theta}$ as a function of random y , to obtain *sampling distribution* results about *estimator* $\hat{\theta}$.

Let $\mathcal{I}_{jk} = -\mathbb{E}(\partial^2 l / \partial \theta_j \partial \theta_k)$. As $n \rightarrow \infty$

1. $\hat{\theta} \rightarrow \theta$ and $\text{var}(\hat{\theta}) \rightarrow 0$ (consistency).
2. $\hat{\theta} \sim N(\theta, \mathcal{I}^{-1})$.
3. In practice if $\hat{\mathcal{I}}_{jk} = -\partial^2 l / \partial \theta_j \partial \theta_k$, then $\hat{\theta} \sim N(\theta, \hat{\mathcal{I}}^{-1})$.

1.3 Model comparison

Consider comparing a model with parameters θ to a simplified version defined by q restrictions $R(\theta) = 0$.

Let $\hat{\theta}_R$ be the maximizer of $l(\theta)$ subject to $R(\theta) = 0$.

If the simplified model is true then, as $n \rightarrow \infty$, $2[l(\hat{\theta}) - l(\hat{\theta}_R)] \sim \chi_q^2$. This is used to select between nested models by a Generalized Likelihood Ratio Test.

Can try to select model that is closest to truth in sense of minimizing the Kullbak-Leibler distance $K(\hat{\theta}, \theta_0) = \int [\log\{f_{\theta_0}(y)\} - \log\{f_{\hat{\theta}}(y)\}] f_{\theta_0}(y) dy$ (θ_0 is parameter vector of truth). Estimating the expected value of K (w.r.t. distribution of $\hat{\theta}$) leads to model selection by Akaike's Information Criterion (AIC).

Choose the model with the lowest value of $\mathcal{V}_{aic} = -2l(\hat{\theta}) + 2p$ where $p = \text{dim}(\theta)$.

2 Practical MLE

2.1 Numerical likelihood maximization

Newton's method: Guess $\theta^{[0]}$, set $k = 0$ and iterate following to convergence

1. Evaluate $g_i = \partial l / \partial \theta_i$ and $H_{ij} = \partial^2 l / \partial \theta_i \partial \theta_j \forall i, j$ at $\theta^{[k]}$, so $l(\theta)$ can be quadratically approximated

$$l(\theta^{[k]} + \Delta) \simeq l(\theta^{[k]}) + g^T \Delta + \frac{1}{2} \Delta^T H \Delta$$

2. Maximize the quadratic approx. $\Rightarrow \Delta = -H^{-1}g$ (if needed first perturb H to make +ve def.)
3. While $l(\theta^{[k]} + \Delta) < l(\theta^{[k]})$ set $\Delta = \Delta/2$.
4. Set $\theta^{[k+1]} = \theta^{[k]} + \Delta$, increment k by one and return to 1.

Quasi-Newton: build up an approximate H using successive g vectors — often very efficient.

May need to approximate g (and H) by finite-difference: e.g. $g_i \simeq \{l(\theta + I_i^T \epsilon) - l(\theta)\} / \epsilon$, if ϵ small.

Nelder-Mead: Successful and very simple geometric derivative free approach.

2.2 Maximizing likelihoods in R

Maximizing $l(\theta)$ w.r.t. $\theta \equiv$ minimizing $-l(\theta)$ w.r.t. θ .

Can numerically minimize $-l(\theta)$ in R using `optim`, `nlm` or `nlminb`.

First step: write an R function to *evaluate* negative log likelihood. Parameter vector must be first argument. e.g. model of data, x_i is that x_i are i.i.d. $N(\mu, \sigma^2)$.

```
normal.ll <- function(theta,x) { ## evaluate -ve log lik for normal model
  mu <- theta[1];sigma <- exp(theta[2]) ## notice trick to ensure sigma +ve
  ll <- -sum((x-mu)^2)/(2*sigma^2) - length(x)*log(sqrt(2*pi)*sigma) ## log lik
  -ll ## return -ve log likelihood
}
```

Second step: check that the function gives sane results. For example

```
> x <- rnorm(100) ## 100 iid N(0,1) r.v.s
> normal.ll(c(0,0),x) ## -ve log lik at "truth"
[1] 131.0876
> normal.ll(c(1,1),x) ## and away from truth
[1] 205.6841
```

...not obviously garbage!

Third step: optimize to find MLEs e.g. using `optim`, $\theta^{[0]}$ is first argument, and name of -ve log likelihood function is second, other arguments of -ve log likelihood are passed *named* at end.

```
fit <- optim(c(1,1),normal.ll,method="BFGS",hessian=TRUE,x=x)
```

`method="BFGS"` selects a quasi-newton method, `hessian=TRUE` indicates that finite difference approximation to H should be returned. `nlm` and `nlminb` are similar, but see R help for details.

Fourth step: extract results (also do model checking – see later). For example

```
mu.hat <- fit$par[1];sigma.hat <- exp(fit$par[2]) ## estimates
se <- diag(solve(fit$hessian))^0.5 ## standard errors (fitting scale)
```

3 Practical model checking and comparison

3.1 Residuals

Models make assumptions.

We should be sceptical about assumptions and at least check that they are ‘reasonable’.

In particular models make assumptions about the distribution of data. If these are badly wrong, then likelihood based inference will be invalid.

Can often define ‘residuals’ which should have approximately some known distribution if the model is correct. e.g.

1. If model $\mathbf{y} \sim \mathbf{N}(\boldsymbol{\mu}, \mathbf{I}\sigma^2)$ is correct then residuals $\hat{\epsilon}_i = (y_i - \hat{\mu}_i)/\hat{\sigma}_i \sim N(0, 1)$ (approx.)
2. If model $\mathbf{y} \sim \mathbf{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is correct then residuals $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\Sigma}^{-1/2}(\mathbf{y} - \hat{\boldsymbol{\mu}}) \sim N(\mathbf{0}, \mathbf{I})$ (approx.)
3. If model $\mathbf{y} \sim \text{Poi}(\boldsymbol{\mu})$ is correct then residuals $(y_i - \hat{\mu}_i)/\sqrt{\hat{\mu}_i}$ are somewhat like i.i.d. $N(0, 1)$ r.v.s

The conformity of the residuals to their theoretical behaviour is judged from plots. e.g.

1. A scatter plot of $\hat{\boldsymbol{\epsilon}}$ vs $\hat{\boldsymbol{\mu}}$ is checked for pattern on the mean of $\hat{\epsilon}_i$, and for pattern in the variance of the $\hat{\epsilon}_i$ — don’t want either.
2. A QQ-plot plots the ordered residuals against the appropriate number of quantiles of their ideal distribution. A straight line indicates conformity with assumptions.

3.2 Comparing models

Can compare nested model using GLRT. For example consider epidemic model

$$y_i \sim \text{Poi}\{n_0 \exp(rt - dt^2)\}, \quad n_0, r > 0$$

Want to compare model 0 with $d = 0$ (uncontrolled spread) to full model 1 (eventual control).

1. Write (negative) log-likelihood functions for the alternative models, and optimize.

```
y<-c(12,14,33,50,67,74,123,141,165,204,253,246,240); t<-1:13
```

```
l10 <- function(b,y,t) { n0 <- exp(b[1]);r <- exp(b[2]);  
  mu <- n0*exp(r*t);-sum(dpois(y,mu,log=TRUE)) }
```

```
l11 <- function(b,y,t) {n0 <- exp(b[1]);r <- exp(b[2]);d <- b[3]  
  mu <- n0*exp(r*t-d*t^2);-sum(dpois(y,mu,log=TRUE)) }
```

```
fit0 <- optim(c(1,-1),l10,method="BFGS",y=y,t=t)  
fit1 <- optim(c(1,-1,-2),l11,method="BFGS",y=y,t=t)
```

2. Form the log likelihood ratio statistics $2\{l(\hat{\theta}_1) - l(\hat{\theta}_0)\}$

```
llr <- 2*(fit0$value-fit1$value) ## ‘backwards’ since ‘values’ are -loglik
```

3. Work out **p-value**: probability of at least this large a log likelihood ratio *if simpler correct*.

```
> p.val <- pchisq(llr,df=1,lower.tail=FALSE);p.val  
[1] 3.722644e-17 ## data hugely improbably under null => accept model 1
```

AIC is an alternative (doesn’t require nested models)

```
> aic0 <- 2*fit0$value + 2*2;aic1 <- 2*fit1$value + 2*3;aic0;aic1  
[1] 166.3698  
[1] 97.4512 ## again model 1 is selected --- has lower AIC
```

4 Bayesian Statistics

Key idea: Parameters, θ , are treated as random variables, not fixed.

Recall: $f_{xy}(x, y) = f_{x|y}(x|y)f_y(y)$ and $f_x(x) = \int f_{xy}(x, y)dy$ ($f \cdot (\cdot)$ are pdfs).

Bayes rule/theorem: $f_{\theta|y}(\theta|y)f_y(y) = f_{y|\theta}(y|\theta)f_\theta(\theta)$, or $f_{\theta|y}(\theta|y) = f_{y|\theta}(y|\theta)f_\theta(\theta)/f_y(y)$.

Bayesian inference: plug in observed y , choose *prior* $f_\theta(\theta)$ and use *posterior* $f(\theta|y)$ in a (loosely) similar way to sampling distribution of $\hat{\theta}$. Note $f_{y|\theta}(y|\theta)$ is $L(\theta)$, but need to specify $f_\theta(\theta)$ is new.

Problem: for most interesting cases $f_y(y) = \int f_{y|\theta}(y|\theta)f_\theta(\theta)d\theta$ is intractable \Rightarrow can't get $f_{\theta|y}(\theta|y)$.

Solution: simulate a Markov Chain: $\theta_1^*, \theta_2^*, \dots$ with stationary distribution $f_{\theta|y}(\theta|y)$. Doesn't need $f_y(y)$!

Simulated values, $\theta_j^*, \theta_{j+1}^*, \dots$, used as (correlated) sample from $f_{\theta|y}(\theta|y)$.

$\mathbb{E}(g(\theta)) \simeq \text{mean}_j\{g(\theta_j^*)\}$, for any function g of θ .

$C\%$ credible interval for $g(\theta)$: based on $g(\theta_j^*)$ for $C\%$ of θ_j^* with highest posterior probability $f_{\theta|y}(\theta|y)$.

4.1 Constructing the chain: MCMC

Markov chain: sequence $\theta_1, \theta_2, \dots$ (dropped the \cdot^* s) where pdf of θ_j is determined entirely by θ_{j-1} .

Detailed balance (reversibility): let $P(\theta_i|\theta_j)$ be p.d.f. of θ_i given θ_j . We require

$$P(\theta_j|\theta_{j-1})f(\theta_{j-1}|y) = P(\theta_{j-1}|\theta_j)f(\theta_j|y) \quad (1)$$

LHS of (1) is joint pdf of θ_j, θ_{j-1} from chain. Integrating w.r.t. θ_{j-1} gives marginal of $\theta_j \dots$

$$\int P(\theta_j|\theta_{j-1})f(\theta_{j-1}|y)d\theta_{j-1} = \int P(\theta_{j-1}|\theta_j)f(\theta_j|y)d\theta_{j-1} = f(\theta_j|y)$$

i.e. given θ_{j-1} from $f(\theta_{j-1}|y)$, the chain generates θ_j also from $f(\theta_j|y)$ as result of (1).

Metropolis-Hastings method constructs chain with appropriate P :

1. Pick a *proposal distribution* $q(\theta_j|\theta_{j-1})$ (e.g. normal centred on θ_{j-1}). Iterate...
2. Generate θ_j^p from $q(\theta_j|\theta_{j-1})$.
3. Set $\theta_j = \theta_j^p$ with probability

$$\alpha = \min \left\{ 1, \frac{f_{y|\theta}(y|\theta_j^p)f_\theta(\theta_j^p)q(\theta_{j-1}|\theta_j^p)}{f_{y|\theta}(y|\theta_{j-1})f_\theta(\theta_{j-1})q(\theta_j^p|\theta_{j-1})} \right\}$$

otherwise set $\theta_j = \theta_{j-1}$. Increment j .

Note that q terms cancel if q depends only on $|\theta_j - \theta_{j-1}|$ (e.g. q normal centred on θ_{j-1}). Same goes for *priors* f_θ if they are uniform. Then

$$\alpha = \min \{1, L(\theta_j^p)/L(\theta_{j-1})\}$$

Burn-in: θ_1 may be very improbable \Rightarrow always need to discard first few hundred/thousand θ_j

4.2 Why Metropolis Hastings works

As we saw above MH will work if it satisfies detailed balance. Proof of detailed balance is easy. To simplify notation let $\pi(\theta) = f(\theta|y)$, so that MH acceptance probability from θ to θ^P is

$$\alpha(\theta^P, \theta) = \min \left\{ 1, \frac{\pi(\theta^P)q(\theta|\theta^P)}{\pi(\theta)q(\theta^P|\theta)} \right\}$$

Need to show that $\pi(\theta)P(\theta^P|\theta) = \pi(\theta^P)P(\theta|\theta^P)$. Trivial if $\theta^P = \theta$. Otherwise $P(\theta^P|\theta) = q(\theta^P|\theta)\alpha(\theta^P|\theta)$

$$\Rightarrow \pi(\theta)P(\theta^P|\theta) = \pi(\theta)q(\theta^P|\theta)\min \left\{ 1, \frac{\pi(\theta^P)q(\theta|\theta^P)}{\pi(\theta)q(\theta^P|\theta)} \right\} = \min \{ \pi(\theta)q(\theta^P|\theta), \pi(\theta^P)q(\theta|\theta^P) \} = \pi(\theta^P)P(\theta|\theta^P)$$

by symmetry of 3rd term above.

4.3 Gibbs sampling

Let θ_i denote i th element of θ , and θ_{-i} the other elements. Sometimes it easy to generate from $f(\theta_i|\theta_{-i})$. Given that $f(\theta) = f(\theta_i|\theta_{-i})f(\theta_{-i})$, it is clear that if we have a draw from $f(\theta)$ and update its i th component by replacing it with a draw from $f(\theta_i|\theta_{-i})$, the result is still a draw from $f(\theta)$. This provides a more efficient alternative to MH for some special model structures.

4.4 Link with maximum likelihood estimation

Uniform priors \Rightarrow modes of $f_{\theta|y}(\theta|y)$ are MLEs.

As $n \rightarrow \infty$ likelihood dominates prior (assuming informative) and posterior modes tend to MLEs.

As $n \rightarrow \infty$ Bayesian credible intervals achieve nominal frequentist coverage probability.

5 Practical MCMC: a toy example

MCMC is quite computer intensive, and for many problems it may be better to use a compiled language, such as C, or a combination of R and compiled code, rather than just R.

Consider a *very* simple example, where you know what sort of answers to expect. Suppose data x_1, x_2, \dots, x_n can be modelled as independent $N(\mu, \sigma^2)$ random variables.

We will work with parameters μ and $\beta = \log(\sigma^2)$, and assume (improper) uniform priors for both.

Simulate some example data `x <- rnorm(100)*.1+2`.

The log-likelihood for the parameters is

$$\log(f(x|\mu, \beta)) = l(\mu, \beta) = -n \log(2\pi\sigma^2)/2 - \sum_i (x_i - \mu)^2 / (2\sigma^2)$$

```
ll <- function(theta,y) {
  mu <- theta[1];sig2 <- exp(theta[2]);n <- length(y)
  -n/2*log(2*pi*sig2) - sum((y-mu)^2)/(2*sig2)
}
```

As a proposal distribution we could use a normal centered on the current μ and β values. Then the following implements Metropolis-Hastings sampling.

```
n.rep <- 10000          ## length of chain
theta<- matrix(0,n.rep,2) ## storage for samples
ll.t <- ll(theta[1,],x) ## log-likelihood of initial state
accept <- 0            ## keep track of acceptance rate
for (i in 2:n.rep) {   ## the MCMC loop
  theta[i,] <- theta[i-1,] + rnorm(2)*.02 ## proposal
  ll.p <- ll(theta[i,],x) ## log-likelihood of proposal
  alpha <- min(1,exp(ll.p-ll.t)) ## acceptance probability
  if (runif(1)<alpha) { ## accept proposal...
    ll.t <- ll.p;accept <- accept +1
  } else { ## reject (leave chain in previous state)...
    theta[i,] <- theta[i-1,]
  }
}
```

Check acceptance rate (sensitive to proposal). Too high \Rightarrow proposal too cautious; too low \Rightarrow bad proposal!

```
> accept/n.rep
[1] 0.518 ## excellent --- should converge quickly without wasting steps
```

Check convergence. As a bare minimum plot values against step:

```
par(mfrow=c(2,1),mar=c(6,6,1,1))
plot(1:n.rep,theta[,1],type="l",ylab=expression(mu),xlab="step")
plot(1:n.rep,exp(theta[,2])^.5,type="l",ylab=expression(sigma),xlab="step")
```

... apparently good convergence after 2000 steps (don't believe these notes — try it!).

Now examine the posterior distributions. (Sometimes you may need to *subsample* the chain at regular intervals, if it is important to have *independent* draws from the posteriors.) e.g. marginals of μ and σ :

```
par(mfrow=c(1,2));
hist(theta[2000:n.rep,1],xlab=expression(mu),main="")
hist(exp(theta[2000:n.rep,2])^.5,xlab=expression(sigma),main="")
```

... compare these to the comparable results from any first year statistics course.

6 Random effects

So far we have modelled data as being noisy observations of a *deterministic* process, represented by a deterministic (sub) model.

Often data are better modelled as noisy observations of a *stochastic* process, represented by a stochastic (sub) model.

For a given set of parameters a deterministic model gives the same results each time you simulate from it, whereas a stochastic model will give different (but hopefully somehow similar) results each time.

Random components of a model that are not simple 'independent residual error' terms are often known as *random effects*.

Example 1 A more plausible version of the urchin model (see lab 1) might be:

$$V_i = \begin{cases} v0_i \exp(c_i a_i) & a_i < a_{mi} \\ f_i/c_i + f_i(a_i - a_{mi}) & \text{otherwise} \end{cases}$$

where $v0_i \sim (V_r, \sigma_v^2)$, $c_i \sim N(\gamma, \sigma_\gamma^2)$ and $f_i \sim N(\phi, \sigma_\phi^2)$ and $a_{mi} = \log\{(f_i)/(c_i v0_i)\}/c_i$. i.e. each urchin follows the same basic model, but with its own particular coefficients. The model now has 6 parameters to estimate, but we can no longer write down the joint pdf of the V_i measurements. i.e we can't write down the likelihood.

Example 2: A simple deterministic Ricker type model for a time-series of population data might be

$$N_t = rN_t e^{-\alpha N_t}, \quad Y_t = N_t + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

so it is easy to write down the joint pdf of the Y_t and hence the likelihood of r , α and σ^2 . But

$$N_t = rN_t e^{-\alpha N_t + e_t}, \quad e_t \sim N(0, \sigma_e^2), \quad Y_t = N_t + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

is more plausible, and now we have to integrate out the e_t to get the likelihood — very difficult.

The problem. The likelihood of the parameters is the marginal pdf of the data (not the pdf of the data and the random effects). Usually the joint pdf of data and random effects can be written down, but the random effects have to be integrated out to get the likelihood, and this is not usually possible exactly.

Solution 1. Approximate the integral. e.g. we have $f(b, y)$ where b and y are random effects and data respectively. We want $f(y) = \int f(b, y) db$. Let \hat{b} be maximizer of $f(b, y)$ (for a given y) and $H_{ij} = -\partial^2 \log f / \partial b_i \partial b_j$ (evaluated at \hat{b}). Now

$$f(b, y) = \exp[\log\{f(b, y)\}] \simeq f(\hat{b}, y) \exp\{-(b - \hat{b})^\top H(b - \hat{b})/2\},$$

by Taylor's theorem, but the last expression is proportional to a multivariate Normal pdf, and can hence be integrated to give

$$f(y) \simeq (2\pi)^{n_b/2} |H|^{-1/2} f(\hat{b}, y)$$

... a *Laplace approximation* (higher order versions are also possible).

Solution 2. Use a Bayesian approach with MCMC to simulate from the joint posterior distribution of the parameters and random effects (we don't have to integrate out anything to do this). Discard the simulated random effects and you have the marginal posterior of the parameters.

Solution 3. Both the above tend to work well provided that the joint pdf of the random effects and the data, $f(b, y)$, is smooth and well behaved. If $f(b, y)$ is jagged and multi-modal then the Laplace approximation will be poor, and MCMC will give poor mixing. In this case it is usually better to base inference on trying to match statistics of the data which capture the relevant features of the data, but which vary in a relatively stable way from one replicate of the random effects to another.

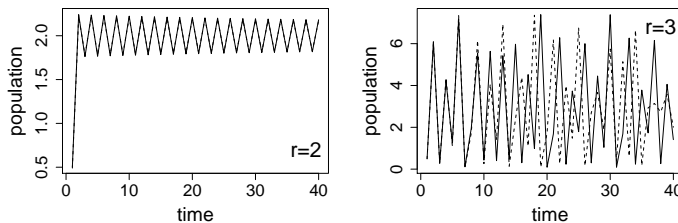
7 The problem with non-linear dynamics

Non-linear dynamic models can show extreme sensitivity to initial conditions.

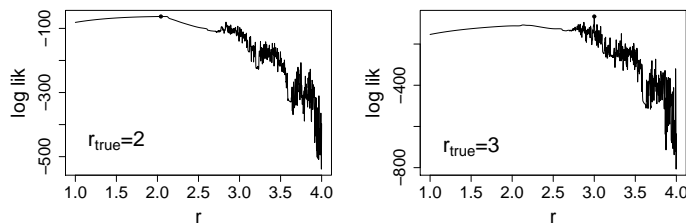
A simple example is provided by a scaled version of the Ricker population dynamic model

$$N_{t+1} = e^r N_t \exp(-N_t).$$

The following overlays trajectories starting from $N_1 = .5$ and $N_1 = .49$ (dashed), for two r values.



This sensitivity to initial conditions affects the model likelihood. Suppose we have a length 40 time series of observations for a population describable by a Ricker model, $Y_t \sim \text{Poi}(\phi N_t)$ (ϕ a scale parameter). The following two figures show transects through the log-likelihood, for different true r values.



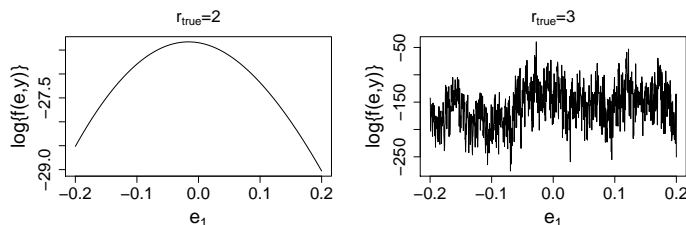
Clearly the likelihood will be difficult to optimize, especially in the $r = 3$ case. For the $r = 3$ case, large sample likelihood results will be useless.

The stochastic case is no better. For example consider the stochastic Ricker,

$$N_{t+1} = e^r N_t \exp(-N_t + e_t), \quad e_t \sim N(0, \sigma_e^2)$$

and assume the same Poisson sampling model as before.

Here are transects through the log joint density, $\log f(e, y)$, for two different r values



... So when $r = 2$ the log density is a nice smooth function of the random effects and Laplace approximation based maximum likelihood, or an MCMC approach, should work well. When $r = 3$ a Laplace approximation would fail, so the likelihood seems intractable. Similarly the posterior will be so multimodal that we can not expect MCMC to work properly.

8 Approximate likelihood based on ‘relevant’ statistics

The statistical difficulty with dynamically complex stochastic models is that the random effects are assumed to come from smooth well behaved uni-modal densities, but the density of random effect values given the data, is anything but smooth and uni-modal. This means that integrating out the random effects to get a likelihood, or doing MCMC, is very difficult.

To make progress we must focus on attributes of a data set which characterize its important dynamic features, but change relatively smoothly as the random effects change. In other words we should discard the very particular features of a set of data, which cause only a tiny and disjoint subset of random effects to be compatible with the data. If we do this then ‘integrating out’ the random effects should be feasible, somehow.

One way of looking at the issue is to recognise that we need data to be a reasonably typical replicate from the model. It should not be the case that the model can only simulate data that ‘look like’ the real data if a very unusual set of random effects occur, and neither should it be the case that a tiny perturbation of that unusual set results in a simulation that looks nothing like the real data. However, with near chaotic dynamic models then the usual approach to constructing a likelihood will result in exactly these two problems.

The problems are caused by small changes in the state at one time resulting in huge changes some time later. Hence we should focus on relationships between data points that are close in time.

One possible approach is to summarize the data using statistics which vary in a relatively stable manner from replicate to replicate of the model. If these summary statistics are chosen carefully, then they can capture most of the dynamically important information in the data, and can be used in place of the original data to construct an approximate likelihood. Note that such statistics should *not* be sufficient statistics, the whole point is that we want to *discard* some information because it is very sensitive to the exact values of the random effects.

e.g. consider a series Y_1, Y_2, \dots, Y_n describable by the Ricker model,

$$N_{t+1} = e^r N_t \exp(-\alpha N_t), \quad Y_t \sim \text{Poi}(N_t).$$

- We could model the relationship between neighboring observations using something like:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-1}^2 + \epsilon_t$$

where the ϵ_t are independent noise terms. The parameters β_j can be estimated by least squares. The structure of this linear model is suggested by the structure of the Ricker model, and so the resulting $\hat{\beta}_j$ plus the observed variance of the Y_t should summarize much of the dynamic information in the series. Hence we could use $S_{\text{obs}} = [\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \text{var}(Y_t)]^T$ in place of the original data. Note that the dimension of S_{obs} is larger than the number of parameters to be estimated.

- The Ricker model can be used to *simulate* a Y_t series and hence an S_{obs} for any given set of its parameters (θ , say) yielding an S_θ .
- So we would seek to find the value for θ that makes S_θ ‘best’ match S_{obs} . How?
 1. Suppose we assume that $S_{\text{obs}} \sim N(\mathbb{E}(S_\theta), \Sigma_\theta)$ (some transformation of S and/or the original data may be required to make this plausible). For any θ we can simulate a large number of S_θ vectors, and hence estimate $\mathbb{E}(S_\theta)$ and Σ_θ .
 2. Since S_{obs} is now ‘the data vector’ we evaluate its probability density according to the estimated normal distribution, and in so doing have evaluated an approximate likelihood of θ .
 3. In fact, if the dynamics are complicated, it is not usually possible to obtain a completely smooth approximate likelihood in this way, so the region of parameter space in the vicinity of the MLE is best explored using MCMC.

9 A practical approximate likelihood example

Consider a scaled version of the Ricker model, as a simple example with which to illustrate the previous section's proposal. Assume $N_{t+1} = e^r N_t \exp(-N_t + e_t)$, $e_t \sim N(0, \sigma_e^2)$ and that we observe $Y_t = N(N_t, \sigma^2)$. The following code simulates multiple replicate series, given a vector of parameters...

```

ricker.sim <- function(theta, noise, n.rep=nrow(noise[[1]]), n.t=ncol(noise[[1]]))
## simulates n.rep replicates of the Ricker model
## given the parameters in theta, and the noise
## matrices in 'noise'. noise[[1]] is process noise
## noise[[2]] is measurement.
{ n <- matrix(0, n.rep, n.t) ## to hold log N_t values
  n[,1] <- exp(theta[1]); ## initial state
  r <- theta[2] ## growth parameter
  sig.proc <- exp(theta[3]) ## proc. noise param.
  sig.meas <- exp(theta[4]) ## obs. error param.
  for (i in 1:(n.t-1)) { ## iterate on log scale. Note: all reps updated at once
    n[,i+1] = n[,i] + r - exp(n[,i]) + noise[[1]][,i]*sig.proc
  }
  y <- exp(n) + noise[[2]]*sig.meas
  y[y<.00001] <- .00001 ## can't observe -ve pop!
  y ## observed population
}

```

Now simulate some data (which we will subsequently try to fit!).

```

set.seed(1)
n <- 50 ## length of series
## setup noise matrices...
noise <- list(matrix(rnorm(n), 1, n), matrix(rnorm(n), 1, n))
n0 <- log(.10); r <- log(40); sig.p <- log(.3); sig.m <- log(.2)
theta <- c(n0, r, sig.p, sig.m) ## parameter vector
y <- ricker.sim(theta, noise) ## simulate population

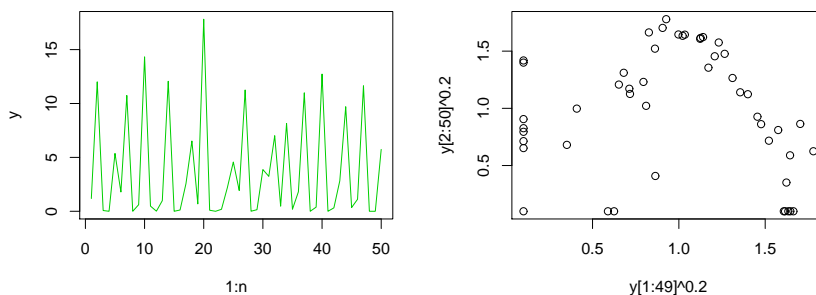
```

...and plot data and Y_t vs. Y_{t-1} ...

```

par(mfrow=c(1,2))
plot(1:n, y, type="l", col=3)
plot(y[1:49]^0.2, y[2:50]^0.2)

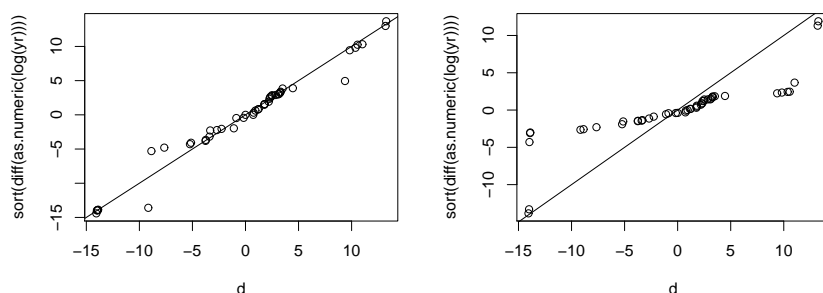
```



The right hand plot suggests that the coefficients, β , of the linear model $Y_t^{0.2} = \beta_0 + \beta_1 Y_{t-1}^{0.2} + \beta_2 Y_{t-1}^{0.4} + \epsilon_t$ might capture the dynamics quite well. The mean and variance of the predicted population might also be useful statistics. To characterize the distribution of the increments ($\log Y_t - \log Y_{t-1}$) we could compare

the observed order statistics for the observed increments, with the corresponding simulated increments. For example

```
d <- sort(diff(as.numeric(log(y))))
noise <- list(matrix(rnorm(n),1,n),matrix(rnorm(n),1,n))
## simulate from model with correct parameters...
yr <- ricker.sim(theta,noise)
plot(d,sort(diff(as.numeric(log(yr)))));abline(0,1)
## simulate from model with incorrect r...
thetar <- c(n0,r-1,sig.p,sig.m) ## parameter vector
yr <- ricker.sim(thetar,noise)
plot(d,sort(diff(as.numeric(log(yr)))));abline(0,1)
```



The left hand plot shows that a simulation from the model with the correct parameters gives increment order statistics very similar to the original, while data simulated with different parameters gives a much less close correspondence. So, as statistics, we could use the coefficients of the linear model relating the simulated ordered increments to the observed ordered increments. Then a function for transforming replicate simulations into replicate statistics might be ...

```
trans <- function(Y,y) {
## function to transform the replicates stored in rows
## of Y, into rows of appropriate statistics:
## 0. the coefficients of the linear regression of the
## ordered first differences on the ordered first
## differences of supplied data, y
## 1. coefficients of quadratic relationship between
## Y[i,] and lagged Y[i,] values
## 2. mean and standard deviations of Y[i,]

## following looks at distribution of first differences of
## log simulations, (rows of Y), in comparison to those of raw
## data, y. Distribution summarized by coefs of linear regression
## of replicate ordered differences of log Y[i,] on ordered diff
## of log y.
d <- sort(diff(as.numeric(log(y)))) ## ordered first differences of log y
X <- cbind(d*0+1,d) ## create model matrix
d <- diff(t(log(Y))) ## unordered first differences of rows of log Y
b <- solve(t(X)%*%X,t(X)%*%apply(d,2,sort)) ## regress ofd log Y on ofd log y

## Now look at auto-regression coefs...
n.rep <- nrow(Y);n.t <- ncol(Y)
X <- matrix(1,n.t-1,3) ## model matrix for auto-regression
```

```

zn <- matrix(0,n.rep,3) ## matrix to hold coefficients
for (i in 1:n.rep) {    ## loop through simulation replicates
  yi <- Y[i,]^2        ## get current rep.
  x <- yi - mean(yi)   ## 'centre' it for numerical stability
  ## fill in linear model matrix columns...
  X[,2] <- x[1:(n.t-1)]
  X[,3] <- x[1:(n.t-1)]^2
  ## following is quick way of obtaining coef(lm(y~X-1)) ....
  zn[i,] <- qr.solve(X,yi[2:n.t])
}

## Now combine all the statistics into one matrix. Each row contains the
## stats for one replicate
zn <- cbind(zn,t(b),apply(Y,1,mean),apply(Y,1,var)^.5) ## note mean, var on raw
zn
}

```

At this stage it might be a good idea to check that the statistics look roughly multivariate normal, using graphical checks — if they don't then some modification (e.g. transformation) might be needed. Let's assume that these checks are ok and move on to the approximate likelihood.

```

ll <- function(theta,y,noise) {
  ## calculates the simulated log likelihood of theta
  ## based on pseudodata derived from y, by simulating
  ## nrow(noise[[1]]) replicates of y using function
  ## 'sim'.

  Y <- ricker.sim(theta,noise) ## each row of Y is a replicate sim
  n.reps <- nrow(Y)

  ## transform sims and data to statistics...
  SY <- trans(Y,y)
  Sy <- trans(y,y)

  ## estimate mean and covariance matrix of statistics
  ms <- colMeans(SY)
  zz <- t(SY) - ms
  Vs <- zz%*%t(zz)/(n.reps-1)

  ## approximate log likelihood of model given 'data' Sy...
  as.numeric(-(Sy - ms)%*%solve(Vs,t(Sy-ms))/2 - determinant(Vs,log=TRUE)$modulus/2)
}

```

It is now worth plotting transects through the 'log-likelihood' ...

```

n.rep <- 200;
noise <- list(matrix(rnorm(n*n.rep),n.rep,n),matrix(rnorm(n*n.rep),n.rep,n))

## a transect through the log likelihood
llt <- th2 <- seq(20,80,length=100)
for (i in 1:100) {
  theta1 <- theta
  theta1[2] <- log(th2[i])
}

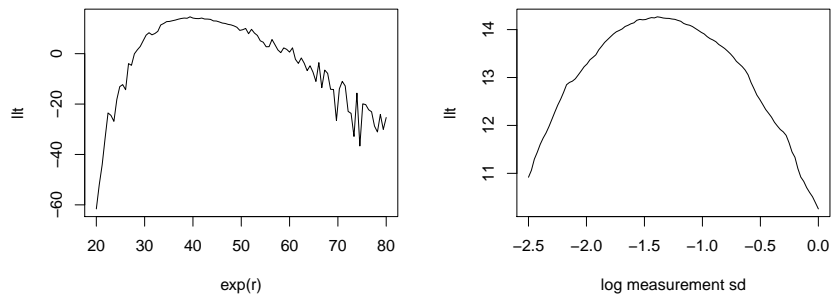
```

```

  llt[i] <- ll(theta1,y,noise)
}

plot(th2,llt,type="l",xlab="exp(r)")
...

```



Notice how the left had transect is still ‘noisy’, but the noise is now very modest in amplitude, relative to what was encountered in section 7 (the plots would be more noisy if the same fixed noise matrices were not used for each set of parameter values). It should be possible to explore this likelihood quite successfully by MCMC...

```

n.mc <- 15000

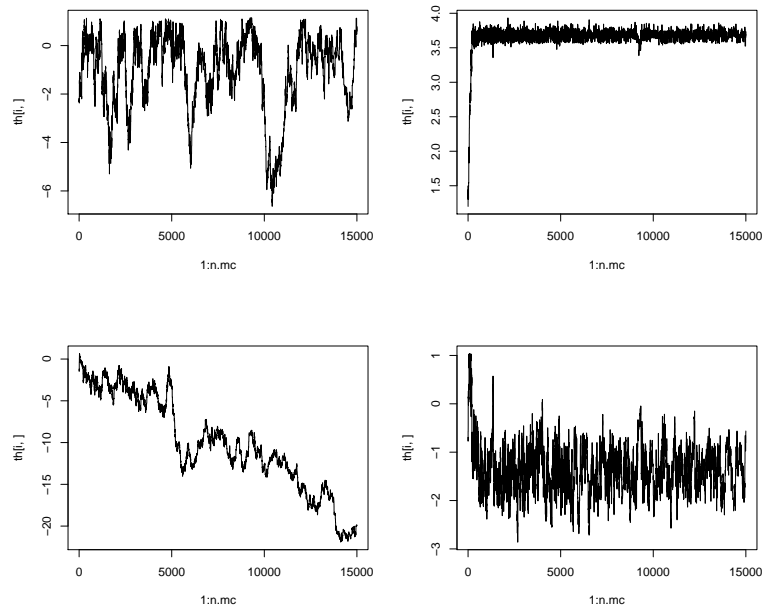
theta0 <- c(log(.1),1.2,log(.3),log(.5))
th <- matrix(0,4,n.mc)
llr <- rep(NA,n.mc)
th[,1] <- theta0
prop.sd <- c(.2,.1,0.2,0.2)

llr[1] <- ll(th[,1],y,noise)
reject <- 0
uni <- runif(n.mc)
for (i in 2:n.mc) {
  th[,i] <- th[,i-1] + rnorm(4)*prop.sd
  ## following trick avoids getting stuck in spikes away from optimum...
  noise <- list(matrix(rnorm(n*n.rep),n.rep,n),matrix(rnorm(n*n.rep),n.rep,n))
  ll.old <- ll(th[,i-1],y,noise)
  llr[i] <- ll(th[,i],y,noise)
  ## ...so we condition the likelihoods on the same noise (for a fair
  ## comparison), but change the noise each step (so spikes aren't fixed)
  alpha <- min(1,exp(llr[i]-ll.old))

  if (uni[i]>alpha) { ## reject
    th[,i] <- th[,i-1]
    llr[i] <- llr[i-1]
    reject <- reject + 1
  }
}
1-reject/n.mc ## acceptance rate

par(mfrow=c(2,2))
for (i in 1:4) plot(1:n.mc,th[i,],type="l")

```



Notice that while r is very well identified, the process noise parameter has effectively wandered to zero here. At this stage it would be possible to obtain a quadratic approximation to the log-likelihood in the vicinity of its maximum, by using linear regression to fit the stored log-likelihood values to a quadratic function of the stored parameter values. The quadratic could then be used to obtain the MLEs, and the Hessian of the log likelihood. For now, let's just look at some replicate simulations from chain ...

```

plot(1:n,y,type="l",col=3)
noise <- list(matrix(rnorm(n),1,n),matrix(rnorm(n),1,n))
y1 <- ricker.sim(th[,15000],noise)
plot(1:n,y1,type="l",col=3)
y1 <- ricker.sim(th[,14000],noise)
plot(1:n,y1,type="l",col=3)
y1 <- ricker.sim(th[,13000],noise)
plot(1:n,y1,type="l",col=3)

```

