introduction to Bayesian state space models

Ben Bolker

McMaster University
Departments of Mathematics & Statistics and Biology

20 June 2017

Outline

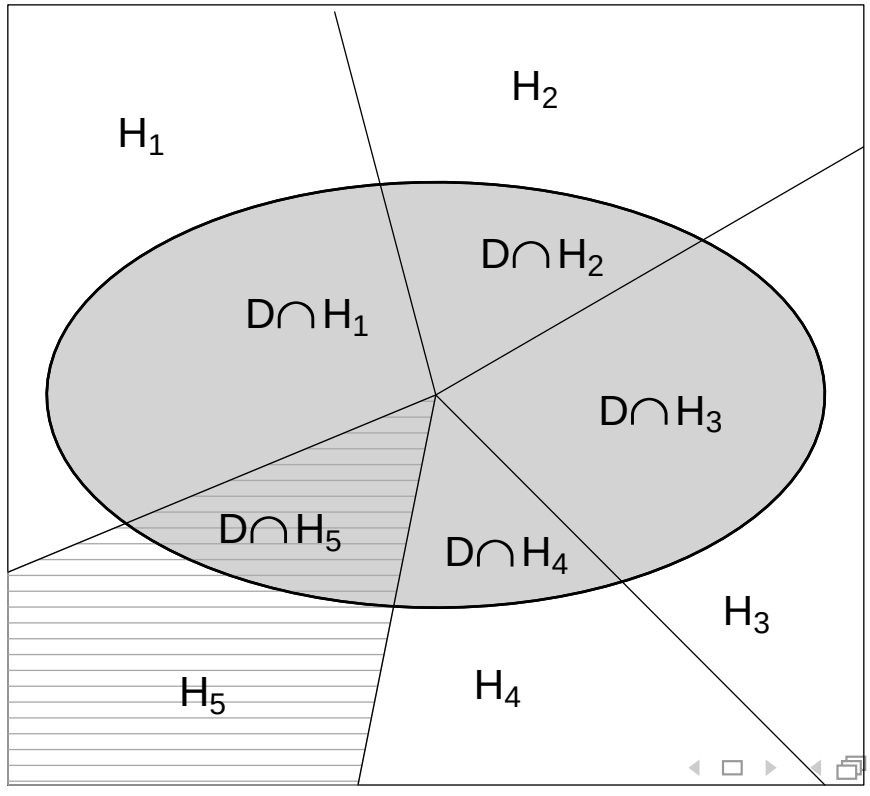
Outline

Big picture

- use Bayes' rule
- avoid frequentist contortions
- integrate prior knowledge
- make coherent decisions
- compute hard things

Bayes rule

$$\underbrace{P(H_i|D)}_{\text{posterior}} = \underbrace{P(D|H_i)}_{\text{likelihood}} \underbrace{P(H_i)}_{\text{prior}} / \underbrace{\sum_j P(H_j)P(D|H_j)}_{\text{P(data)}}$$



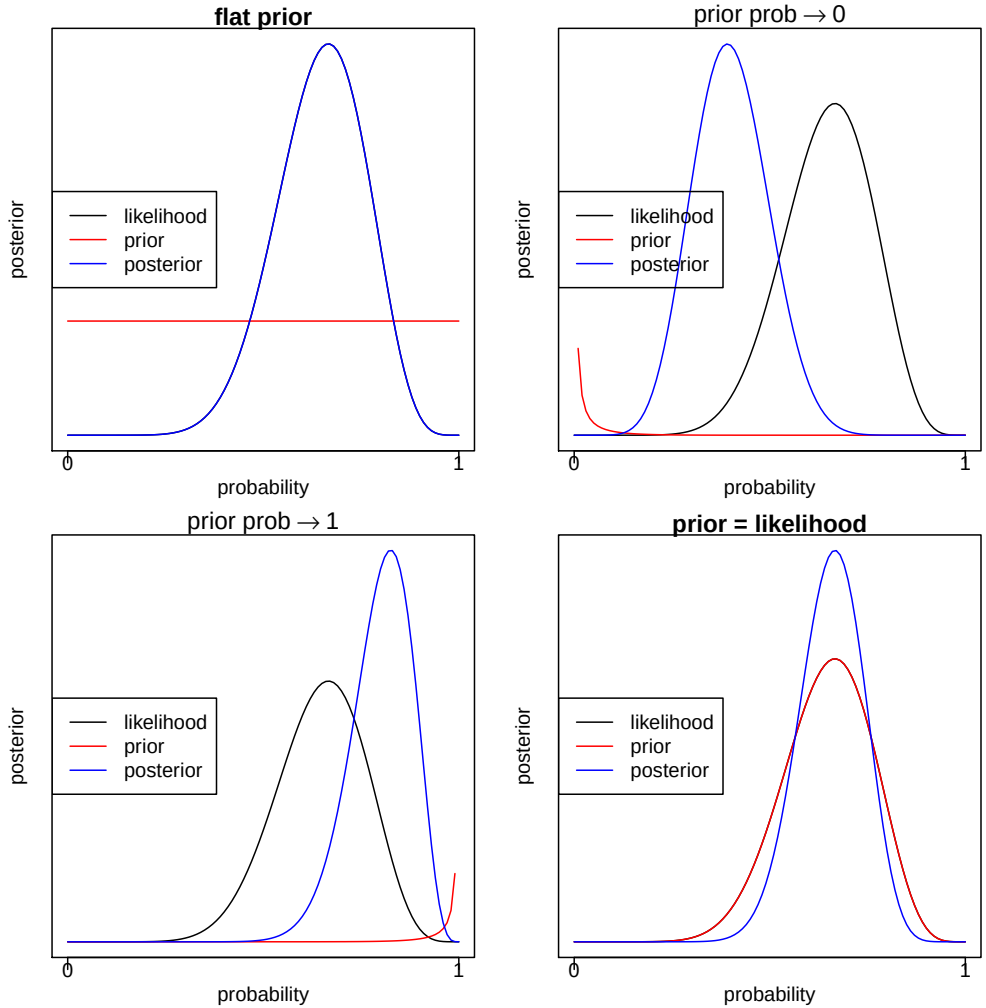
priors: $P(H_i)$

- usually framed as "prior belief"
- controversial because subjective
- if we set all $P(H_i)$ equal, $P(H_i|D) = P(D|H_i)/\sum P(D|H_j)$ (**scaled likelihood**)
- can't really be swept under the rug

more on priors

- **weak** or **diffuse**: little information
- **uninformative** or **flat**: no information
- **improper**: integral diverges (but sometimes OK)
- weak priors can cause problems with sparse data and/or weakly identifiable models
- **conjugate** priors: convenient functional forms

effects of priors



Outline

Markov chain Monte Carlo

- **general** method for sampling posterior probability densities
- construct a Markov chain whose stationary density equals the desired posterior probability density
- avoids computation of Bayes' rule denominator ($\int\int P(\boldsymbol{\theta})\, d\boldsymbol{\theta}$)

you won't believe these two MCMC tricks

Gibbs sampling sample parameters one at a time, exploiting conditioning

Rejection sampling (Metropolis-Hastings): pick new values of parameters at random, then pick a random number to decide whether to keep them

Gibbs sampling

Because $\mathrm{Prob}(A|B) = \mathrm{Prob}(A, B)/\mathrm{Prob}(B)$, we can say

$$\mathrm{Prob}(A, B, C, \ldots Z) \propto \mathrm{Prob}(A|B, \ldots, Z) \cdot \mathrm{Prob}(B|C, \ldots, Z) \cdot \ldots \cdot \mathrm{Prob}(Z)$$

This means that we can sample the conditional probabilities **sequentially** and get the right answer.
picture of sampling

Metropolis-Hastings

Jump, evaluate (prior $\times$ likelihood), decide whether to accept

$$\frac{\text{Prob}(A)}{\text{Prob}(B)} = \frac{P(\text{jump } B \to A) \cdot P(\text{accept } A|B)}{P(\text{jump } A \to B) \cdot P(\text{accept } B|A)}$$

In the long run our chain will converge to the right distribution

- **candidate distribution**: anything sensible
  (bad choices make sampling slow, but not incorrect)
- **acceptance rule**:

$$P(\text{accept } \theta_2) = \max\left(1, \frac{\Pr(\theta_2)L(\theta_2)}{\Pr(\theta_1)L(\theta_1)}\right)$$

i.e. "always accept if θ_2 better: sometimes if θ_2 worse"

Magic black boxes

- Can construct your own, customized samplers (Bolker et al., 2003)
- Use BUGS (Bayesian Inference Using Gibbs Sampling) language (`WinBUGS`, `OpenBUGS`, `JAGS`, `NIMBLE`)
- interfaces from R (https://CRAN.R-project.org/package=R2jags) or MATLAB (`https://github.com/msteyvers/matjags`).

Outline

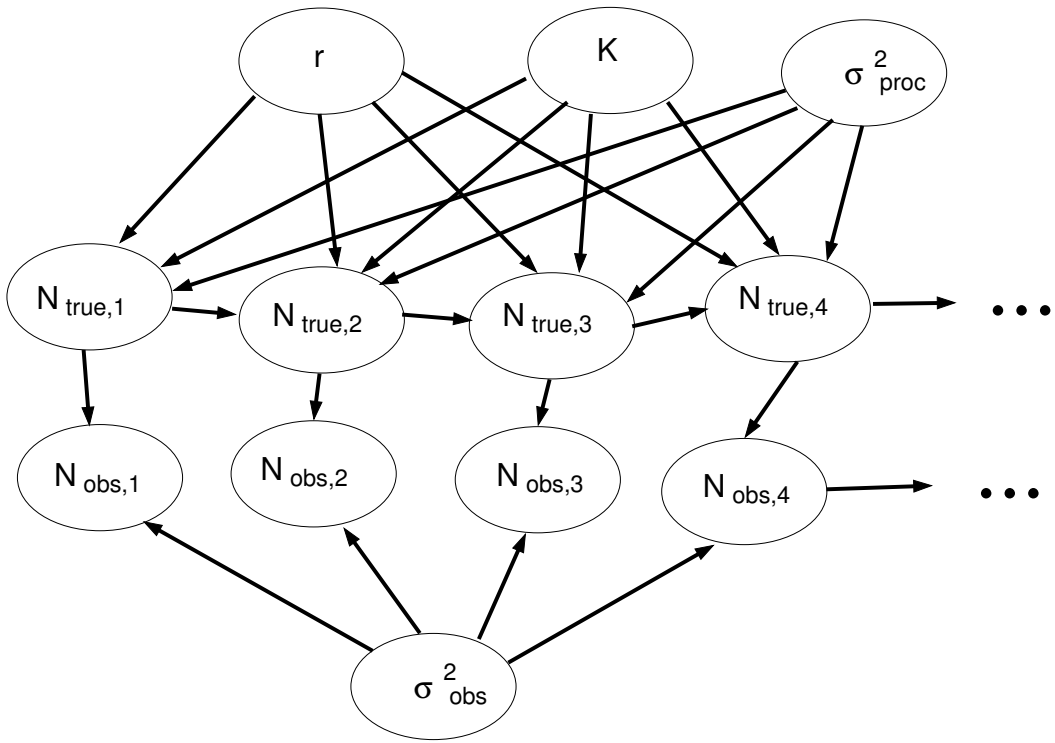
State space models

- address fundamental problem: prob(observations) depends on **unobserved** true values
- have to deal with/integrate over all possible values (**latent variables**)
- very high-dimensional: brute force fails
- use the previous two tricks

BUGS code for the logistic function

```
model <- function() {
  t[1] <- n0     ## initial values ...
  o[1] ~ dnorm(t[1],tau.obs)
  for (i in 2:N) {    ## step through observations ...
     v[i] <- t[i-1]+r*t[i-1]*(1-t[i-1]/K)
     t[i] ~ dnorm(v[i],tau.proc)
     o[i] ~ dnorm(t[i],tau.obs)
  }
  r ~ dunif(0.1,maxr) ## priors ...
  ## rate and scale of gamma
  K ~ dgamma(0.005,0.005)
  tau.obs ~ dgamma(0.005,0.005)
  tau.proc ~ dgamma(0.005,0.005)
  n0 ~ dgamma(1,n0rate)
}
```

Dependency structure for logistic model



BUGS vs R

- BUGS code is not sequential (!)
- BUGS is not vectorized (need `for` loops)
- BUGS: ~ means "distributed as" ("stochastic node")
  <- means assignment ("logical node")
- different distribution names and parameterizations (e.g. `dnorm(mean,prec)` for Normal, `dbin(size,prob)` for binomial): see LeBauer et al. (2013)

Running JAGS

- **Good news**: JAGS code is (relatively) intuitive
- **Bad news**:
  - Debugging is hard
  - Need to figure out how long to run chains (convergence diagnostics)
  - Poor mixing
  - Slow computation

Running JAGS (details)

- specify model and priors
- get model to compile
- run multiple chains
  - discard **burn-in**
  - thin results
- assess convergence

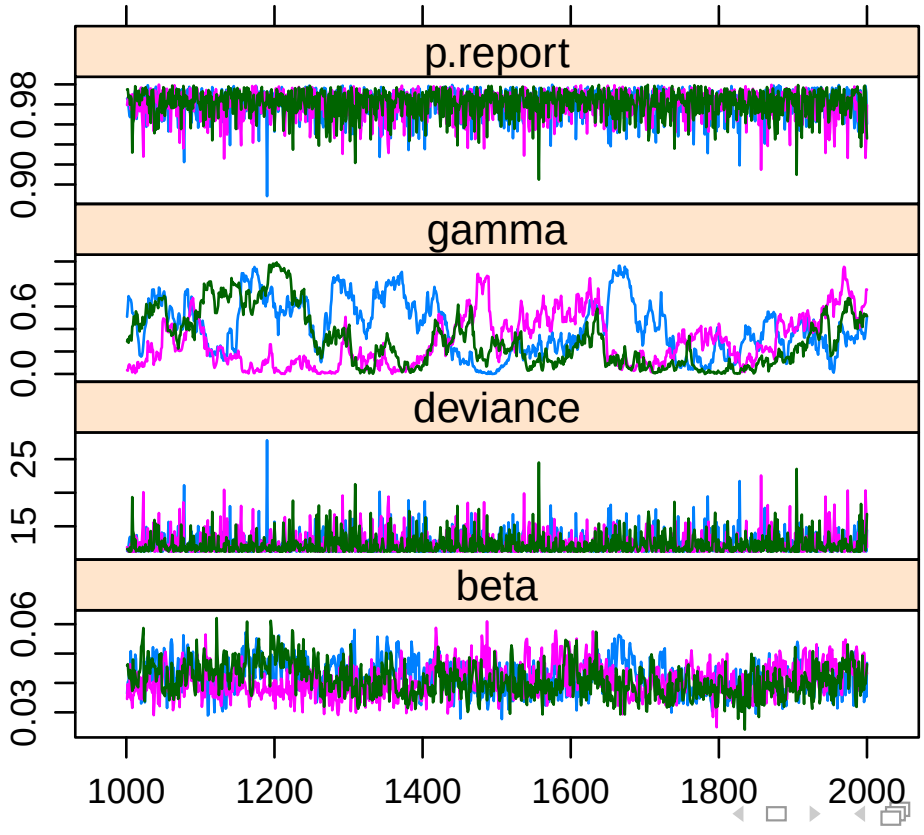
Troubleshooting JAGS

- simplify model
- specify initial values explicitly
- narrow priors and/or fix some parameters
- run longer and thin more

Diagnostics example

trace plots: should look like white noise

```
library(R2jags); library(coda); library(lattice)
xyplot(as.mcmc(jagsout))
```



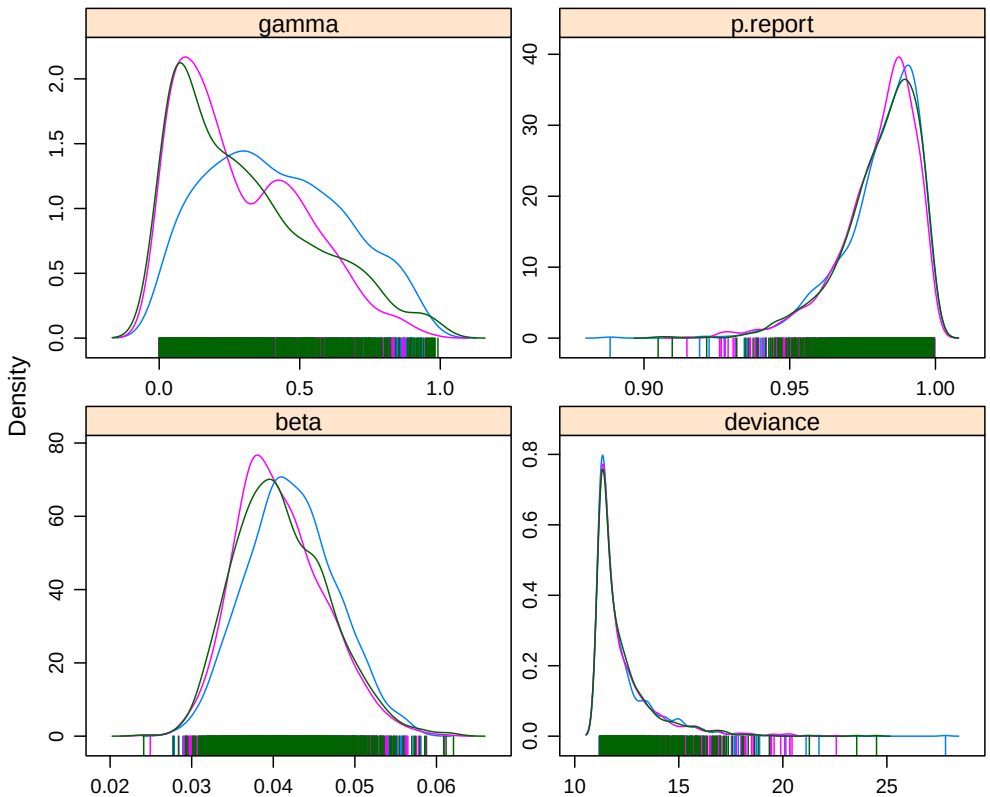
Diagnostics example

Gelman-Rubin statistic: want $\hat{R} < 1.2$

```
gelman.diag(as.mcmc(jagsout))

## Potential scale reduction factors:
##
##           Point est. Upper C.I.
## beta            1.01       1.05
## deviance        1.00       1.00
## gamma           1.06       1.20
## p.report        1.00       1.00
##
## Multivariate psrf
##
## 1.05
```

Density plots



Summary example

```
summary(as.mcmc(jagsout))
```

```
##                Mean        SD  Naive SE Time-series SE
## beta        0.04138 0.005566 0.0001016      0.0003900
## deviance   12.27502 1.506429 0.0275035      0.0290581
## gamma       0.33621 0.243174 0.0044397      0.0452345
## p.report    0.98090 0.013220 0.0002414      0.0002512
```

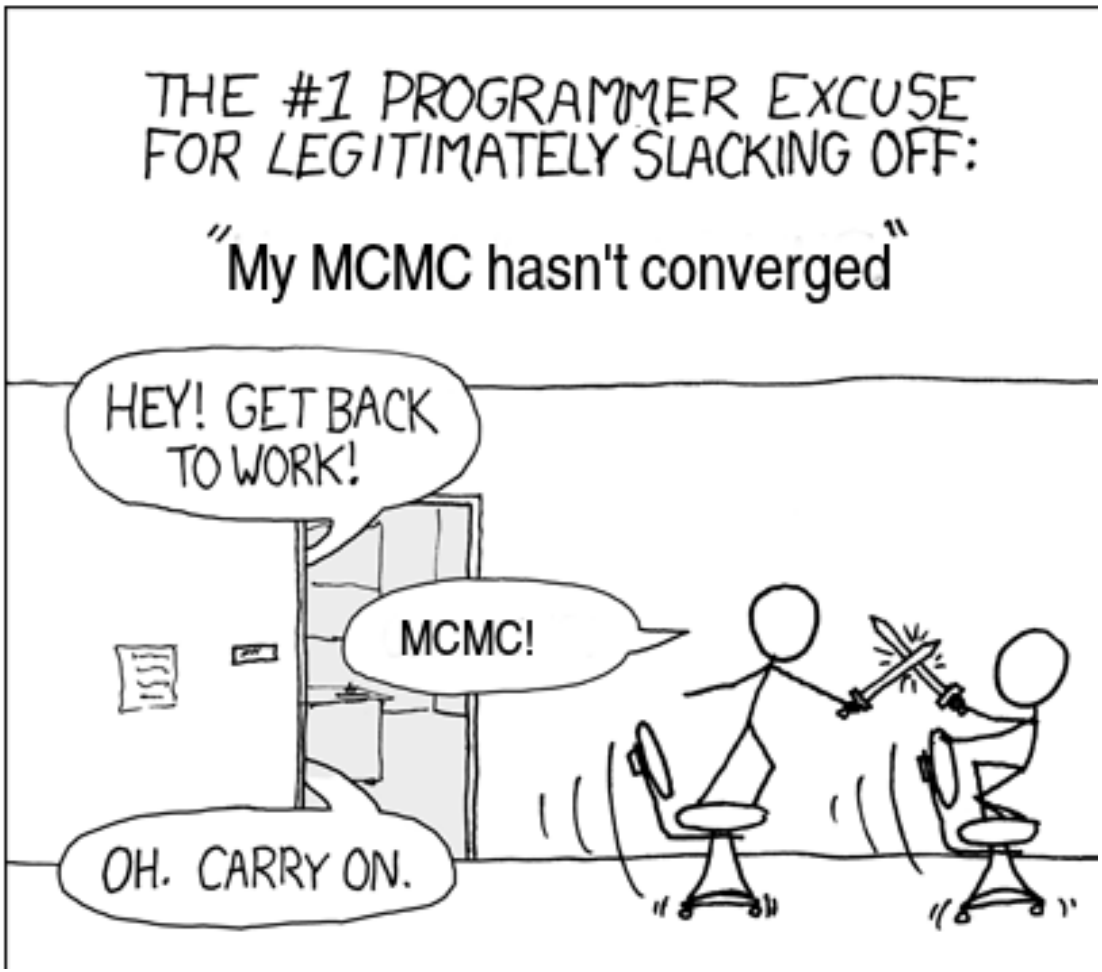
Inference from posteriors

- point estimates: mean, median (marginal)
- interval estimates: quantiles, **highest posterior density** (`coda::HPDinterval`)
- can summarize any quantity computed from samples (e.g. predictions)

Further resources

- Gelman et al. (2013) (fairly hard-core)
- Hobbs and Hooten (2015) (friendlier, ecologist-focused)
- Gelman and Hill (2006) (more regression-focused)

The problem with MCMC (xkcd)



Outline

Frequentist alternatives

- MCMC is usually Bayesian;
  opens various cans of worms
- there are many other related approaches, some classical
  - expectation-maximization
  - sequential Monte Carlo/particle filters (Ionides et al., 2006; Doucet et al., 2001; de Valpine, 2004): R `pomp`, `NIMBLE` packages, PyMC
  - data cloning (Lele et al., 2007): R `dclone` package

Estimation for continuous-time models

- Particle methods
  - simulate many trajectories ("particles") step-by-step
  - at each step, resample particles weighted by likelihood of current location
- Approximate Bayesian computation
  - sample parameters from prior
  - simulate trajectories for each parameter set
  - compute summary statistics ("probes")
  - save trajectories with summary stats near observed values

References

Bolker, B., Okuyama, T., et al., 2003. *Ecological Applications*, 13(3):763–775.

de Valpine, P., 2004. *Journal of the American Statistical Association*, 99:523–536.

Doucet, A., de Freitas, N., and Gordon, N.J., 2001. *Sequential Monte Carlo methods in practice*. Springer-Verlag, New York, USA.

Gelman, A., Carlin, J.B., et al., 2013. *Bayesian Data Analysis*. Chapman and Hall/CRC, Boca Raton, 3d edition. ISBN 978-1-4398-4095-5.

Gelman, A. and Hill, J., 2006. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, Cambridge, England.

Hobbs, N.T. and Hooten, M.B., 2015. *Bayesian Models: A Statistical Primer for Ecologists*. Princeton University Press, Princeton, New Jersey. ISBN 978-0-691-15928-7.

Ionides, E.L., Bretó, C., and King, A.A., 2006. *Proceedings of the National Academy of Sciences of the USA*, 103(49):18438–18443. doi:doi:10.1073pnas.0603181103.

LeBauer, D.S., Dietze, M.C., and Bolker, B.M., 2013. *R Journal*, 5(1):207–209.

Lele, S.R., Dennis, B., and Lutscher, F., 2007. *Ecology Letters*, 10:551–563. doi:doi:10.1111/j.1461-0248.2007.01047.x.