Bayesian Survival Analysis with WinBUGS and R

Laura Graham

May 2015

Background

Frequentist vs. Bayesian inference

- Both approaches start with a model $y = f(x, \theta)$ and then attempt to estimate θ
- Frequentist approach estimates parameters using maximum likelihood estimation
- Bayesian approach uses Bayes Rule

Bayes Rule

Bayes Rule is given as:

- $p(A|B) = \frac{p(B|A)p(A)}{p(B)}$

When thinking about estimating parameters (θ) given a dataset (D):

- $p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$
- $p(\theta|D)$ is the posterior distribution of the parameters θ, $p(D|\theta)$ is the likelihood function and $p(\theta)$ is the prior distribution

Bayesian inference tends to be paraphrased as:

- posterior $\propto$ likelihood x prior

Frequentist vs. Bayesian inference

Frequentist

- Uncertainty expressed as probability statements about the data given fixed parameters
- Maximum likelihood estimation
- Standard error
- 95% confidence intervals
- Akaike information criterion

Bayesian equivalents

- Uncertainty expressed as probability that a parameter takes a certain value given the data
- Mode of the posterior distribution
- Standard deviation
- 95% credible interval (NB unlike 95% CI, 95% CRI can be interpreted as containing the parameter with 95% certainty)
- Deviance information criterion

Bayesian Computation

- Analytically solving Bayes Rule for anything more than the simplest models becomes difficult or impossible
- This means that computational/simulation methods are needed
- Markov Chain Monte Carlo samples from the posterior distribution keeping to areas of high probability given the data
- Many MCMC algorithms exist, the most commonly used software with these algorithms built in are WinBUGS, OpenBUGS and JAGS (all of which can be called from R)

Application

Cormack–Jolly–Seber model in WinBUGS

1. Write model in BUGS language (BUGS)
2. Get data in correct format (R)
3. Create function to generate initial values for parameters (R)
4. Select parameters to monitor (R)
5. Set MCMC characteristics (R)
6. Call WinBUGS to run model (R)

Example: CJS model with constant observation and survival dependent on habitat ($phi(habitat)p(.)$)

1. Write model in BUGS language

- Define model priors and constraints
- Define the likelihood functions of the state and observation processes

In our example, the groups are defined within the priors and constraints section. The state process is defined as:

- $z_{i,f_i} = 1$ (Individual alive at first capture)
- $z_{i,t+1} | z_{i,t} \sim Bernoulli(z_{i,t}\phi_{i,t})$

And the observation process as:

- $y_{i,t} | z_{i,t} \sim Bernoulli(z_{i,t} p_{i,t})$

1. Write model in BUGS language

Pt I: Priors and constraints

```
model {

  # Priors and constraints
  for (i in 1:nind){
    for (t in f[i]:(n.occasions-1)){
      phi[i,t] <- phi.g[group[i]]
      p[i, t] <- mean.p
    } #t
  } #i
  for (u in 1:g){
    # Priors for group-specific survival
    phi.g[u] ~ dunif(0, 1)
  }
  mean.p ~ dunif(0, 1)
```

1. Write model in BUGS language

Pt II: Likelihood function

```
  # Likelihood
  for (i in 1:nind){
    # Define latent state at first capture
    z[i,f[i]] <- 1
    for (t in (f[i]+1):n.occasions){
      # State process
      z[i,t] ~ dbern(mu1[i,t])
      mu1[i,t] <- phi[i,t-1] * z[i,t-1]
      # Observation process
      y[i,t] ~ dbern(mu2[i,t])
      mu2[i,t] <- p[i,t-1] * z[i,t]
    } #t
  } #i
}
```

2. Get data in correct format

Required data:

- Capture history
- Array containing the time period of first capture
- Number of individuals
- Number of time periods
- Matrix of known states
- Number of groups
- Grouping variable (habitat)

```
bugs.data <- list(y = CH, f = f, nind = nrow(CH),
                  n.occasions = ncol(CH),
                  z = known.state.cjs(CH),
                  g = length(unique(group)),
                  group = group)
```

3. Create function to generate initial values for the parameters

Markov chains need initial values to begin

```r
# Number of groups
g <- length(unique(group))

# Function to create initial values
inits <- function(){
          list(z = cjs.init.z(CH, f),
               # cjs.init.z() sets any known values
               # of z to NA (not initialised)
               phi.g = runif(g, 0, 1),
               mean.p = runif(g, 0, 1))
         }
```

4. Select parameters to monitor

When running the model in WinBUGS, we need to tell it which parameters we want it to keep track of.

```
parameters <- c("phi.g", "mean.p")
```

5. Set MCMC characteristics

WinBUGS needs to know how many chains to run, how many iterations on each chain, how many of the iterations should be discarded as burn-in and how many iterations to thin by.

```
ni <- 10000
nt <- 3
nb <- 2000
nc <- 3
```

6. Call WinBUGS to run model

The final stage of developing Bayesian CJS models is to run the model in WinBUGS. This can be done in R using the R2WinBUGS package.

```r
cjs.g <- bugs(bugs.data, inits, parameters,
              "cjs-group.bug", n.chains = nc,
              n.thin = nt, n.iter = ni,
              n.burnin = nb, debug = TRUE,
              working.directory='path/to/wd/')
```

Preliminary Results

Species and models

- 5 species modelled (those with the most captures in the data): *Metallura tyrianthina, Eriocnemis luciani, Diglossa humeralis, Coeligena iris, Diglossopis cyanea*
- Two model formulations: $\phi(habitat)p(.)$, $\phi(time)p(.)$ where habitat is a fixed effect and time is a random effect

Model fit results

Using time as a random effect gives a better fitting model for all species (but possibly needs thinking about penalising for increased parameter estimates). DIC values displayed.

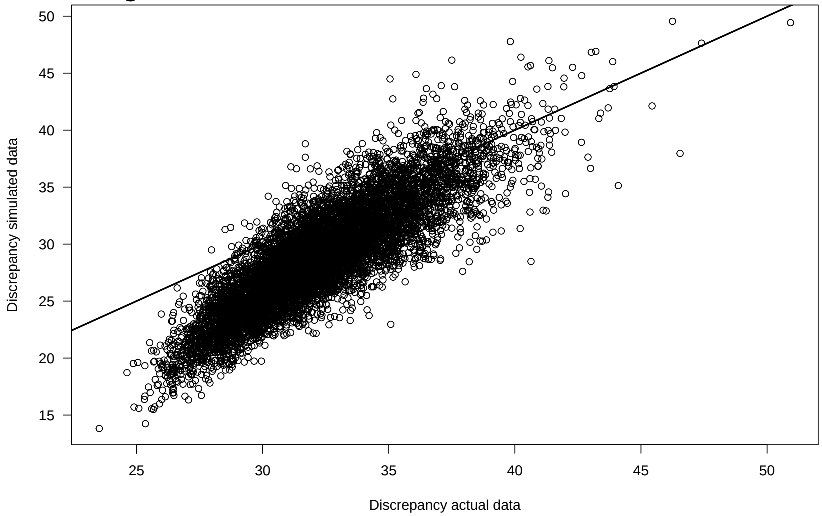
```
##                          time habitat
## COELIGENA IRIS          240.0   526.6
## DIGLOSSA HUMERALIS      287.9   679.1
## DIGLOSSOPIS CYANEA      151.8   241.2
## ERIOCNEMIS LUCIANI      365.7  1031.6
## METALLURA TYRIANTHINA 344.9   995.3
```

Random time effects model results

```
##                          mean.phi mean.p mean.sigma2
## COELIGENA IRIS               0.80   0.10        0.03
## DIGLOSSA HUMERALIS           0.77   0.10        0.00
## DIGLOSSOPIS CYANEA           0.85   0.04        0.02
## ERIOCNEMIS LUCIANI           0.77   0.19        0.03
## METALLURA TYRIANTHINA        0.79   0.08        0.04
```

Fit for D. cyanea random time effects model

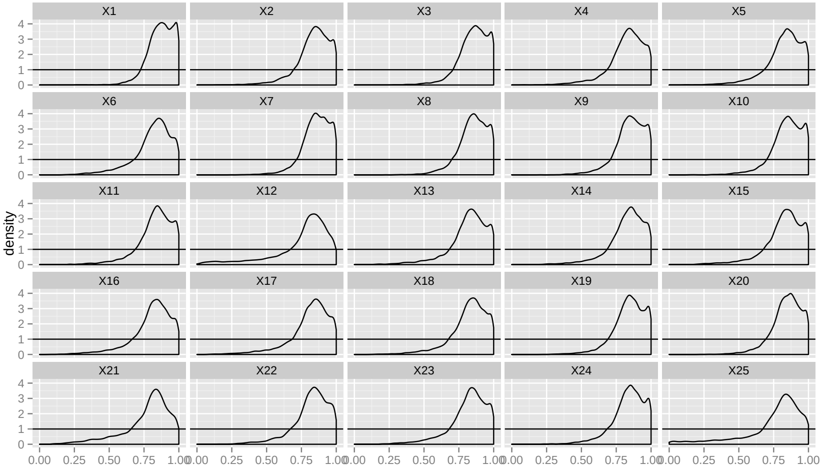
Well fitting model should be close to the 1:1 line



Bayesian p-value $= 0.1079$ (slightly confusing concept, perfect model would have $p = 0.5$)

Parameter identifiability for D. cynea random time effects model

Each panel represents the transition between survey periods.
Density plot shows the posterior estimates from the Bayesian model.
Priors are shown by horizontal line. Identified parameter
is one where there is not too much overlap between posterior and prior.


X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
X13
X14
X15
X16
X17
X18
X19
X20
X21
X22
X23
X24
X25
density