Model Building and Selection for MCMC

Recap

- We can write simple models in JAGS and run them on data provided to us, remembering to handle:
    1. Convergence
    2. Effective sample size

- But so far we've not really considered:
    - How 'good' is our model?
    - How to choose between 'candidate models'

The Perfect Model

- Describes our data
  - Accounts for all known (important) biology
    - The error is i.i.d. – i.e. all correlations are modelled
  - Gives good parameter estimates for relevant effects

- Converges and runs quickly
  - Minimal autocorrelation
  - Parameters are as independent as possible

- Parsimonious
  - Sometimes at odds with describing biology well!

Model Formulation

- Start with the biology
  - Describe the processes that have resulted in your data
  - Simplify complex relationships using (good) approximations
  - Account for clustering but combine inseparable sources of variation together
  - Consider if simplifying your data would help…
  - Don't try to force the data into the wrong distribution!
- Are there any alternative ways you could write it?
  - Compare results from different models
- Are your priors having the effect you intend?
  - Compare results from different models

Equivalent Parameterisations

Gamma response:

```
model{
        for(i in 1:N){
                OpticalDensity[i] ~ dgamma(shape, rate)
        }
        shape ~ dgamma(0.001,0.001)
        rate ~ dgamma(0.001,0.001)
        mean <- shape/rate
        #monitor# shape, rate, mean
        #inits# shape, rate
        #data# N, OpticalDensity
}
```

# results1

Equivalent Parameterisations

Gamma response:

```
model{
        for(i in 1:N){
                OpticalDensity[i] ~ dgamma(shape, rate)
        }
        shape ~ dgamma(0.001,0.001)
        rate  <- shape/mean
        mean ~ dgamma(0.001,0.001)
        #monitor# shape, rate, mean
        #inits# shape, mean
        #data# N, OpticalDensity
}
```

# results2

Equivalent Parameterisations

```
>  results1

JAGS model summary statistics from 20000 samples (chains = 2; burnin = 5000):

             Lower95   Median  Upper95     Mean        SD      MCerr  MC%ofSD  SSeff      AC.10     psrf
mean    0.2999  0.42128  0.58469  0.42942  0.074782  0.0005869      0.8         16235  -0.0049401  1.0001
rate    1.8309   4.5216   7.8176   4.6952    1.5789   0.035277      2.2         2003      0.12786  1.0002
shape   0.9186   1.8971   3.1117   1.9591   0.57944     0.01342     2.3         1864      0.13751  1.0002

Total time taken: 1 seconds

>  results2

JAGS model summary statistics from 20000 samples (chains = 2; burnin = 5000):

             Lower95   Median  Upper95     Mean        SD        MCerr MC%ofSD  Sseff    AC.10     psrf
mean   0.29309  0.42139  0.57661  0.42925  0.074382  0.00074144       .         10064   0.0024101  1.0003
rate    1.8191    4.508   7.8397   4.6842    1.5875    0.01617        1         9638   -0.0058637       1
shape  0.89668   1.8877   3.1117   1.9541   0.58221    0.00601        1         9384   -0.0065946  1.0001

Total time taken: 2 seconds
```

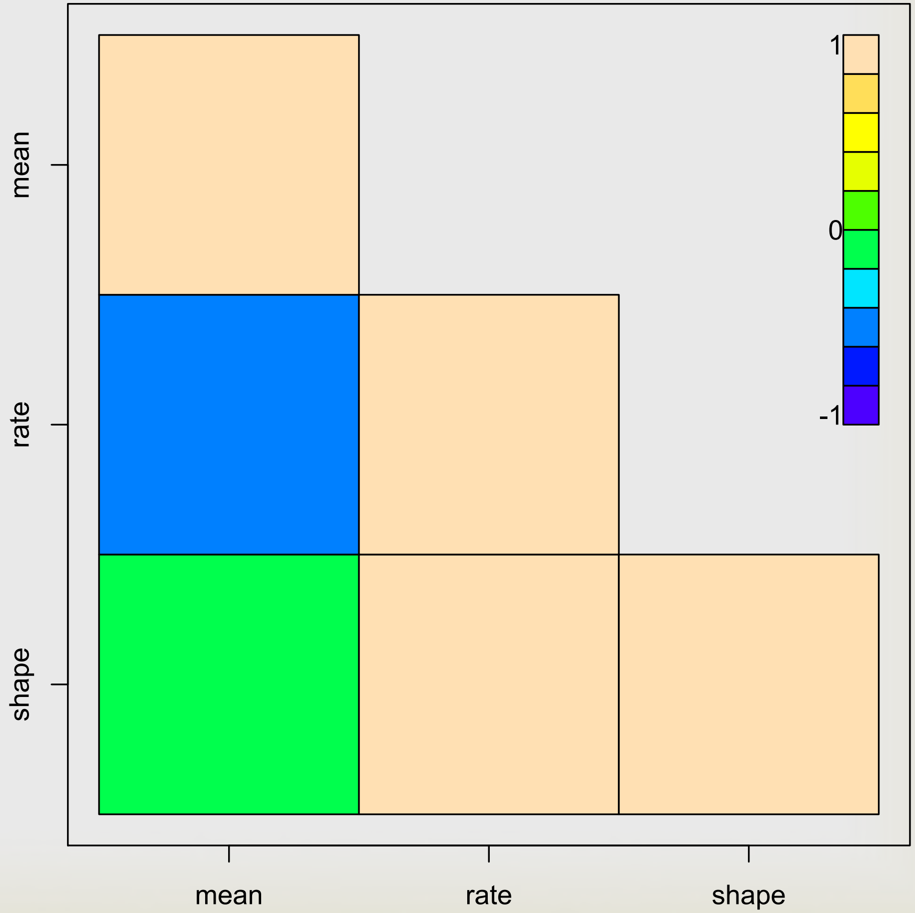
Cross-Correlation

- Cross-correlation is like autocorrelation but BETWEEN parameters
  - ie. How correlated is the current value of variable1 to the previous value of variable2?
  - Gives an indication about how much cross-dependence there is between variables
    - Will also have a knock-on effect on autocorrelation!
  - High cross-dependence between a stochastic and deterministic nodes is irrelevant – these aren't being sampled directly anyway
  - High cross-dependence between stochastic nodes is bad

Cross-Correlation

- shape and rate are heavily cross-correlated
  - so we don’t want both of these to be stochastic

- mean and shape are the most independent parameters
  - so make these the stochastic nodes



Independent Parameters

- To reduce cross-correlation we also want to make stochastic node parameters as independent as possible (i.e. the predictors are orthogonal)
  - Specially relevant to polynomials: ?poly – 'raw' argument

- The mean is nearly always the easiest parameter to estimate
  - Expect more autocorrelation for variance parameters – focus on making these independent to the mean
  - Relevant for parameters of some distributions eg Gamma and Beta

Independent Parameters

1. Re-center linear effects (and polynomials!) to a mean of zero to reduce cross-correlation between the effect(s) and/or the intercept

2. Code the most common category as the reference for fixed effects

3. Design the experiment to prevent quasi-separated data

4. Load the glm module for GLM models:

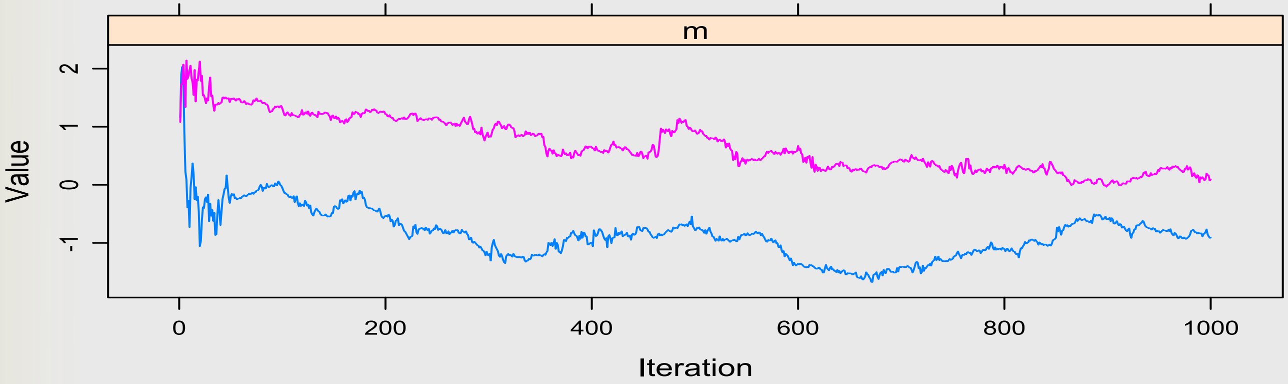
`#module# glm` or **`run.jags(…, modules=‘glm’)`**

Identifiability

- Sometimes we get a model that will not converge
  - Chains seem to have a 'random walk' through the parameter space, with no stable posterior
  - We say that the model is 'unidentifiable'

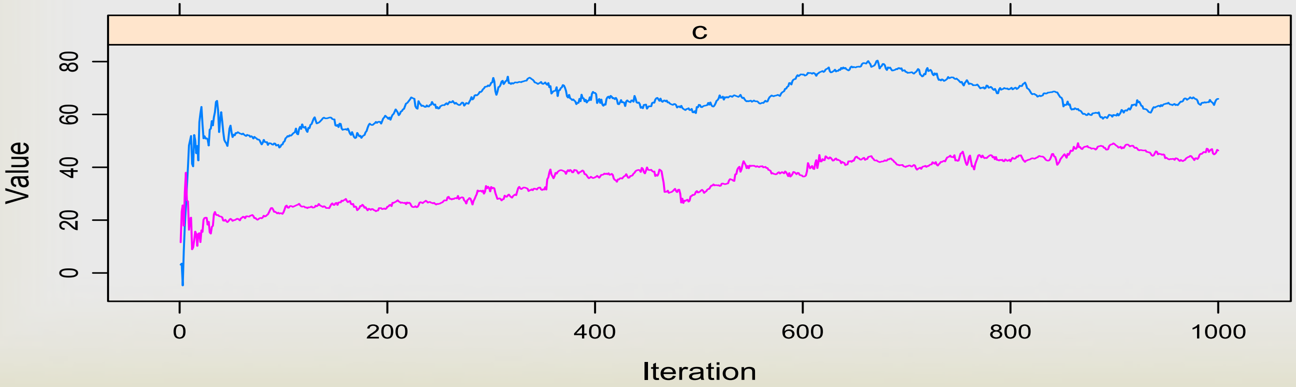
- Possible reasons:
  - Starting values too far away from the stationary posterior distribution for it to be found
  - Extreme conflict between prior and likelihood
    - No possible solution to satisfy both
  - Complete cross-correlation between parameters

Identifiability



m: mean of random effects

Not 0!



c: intercept

Reducing AutoCorr

- MAKE PARAMETERS INDEPENDENT!!!!
  - Center predictors on 0 (and standardise variance)
  - Reformulate model?
  - Use the GLM module in JAGS

- Try changing priors:
  - Weakly informative priors for variance parameters may help a lot!
  - Maybe put the prior on a different scale (e.g. sd vs tau)

Reducing AutoCorr

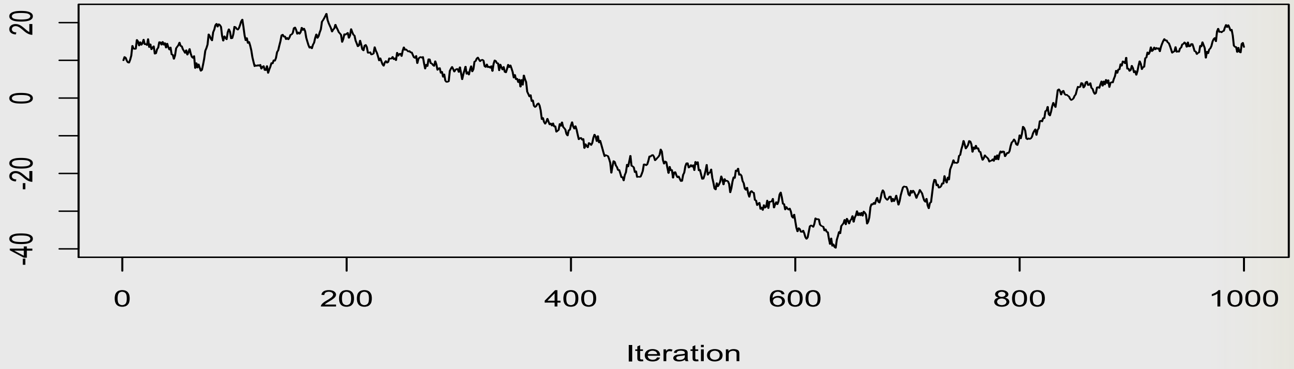
- But … we may still have a model with one or more auto-correlated parameters
  - We have tried and failed to find an alternative formulation
  - There are no other ways of reducing cross-dependence between parameters

- We will have to accept this autocorrelation and get on with life
  - BUT we need to accept that we will have to take (possibly) many more samples from the posterior to compensate for this

Autocorrelation

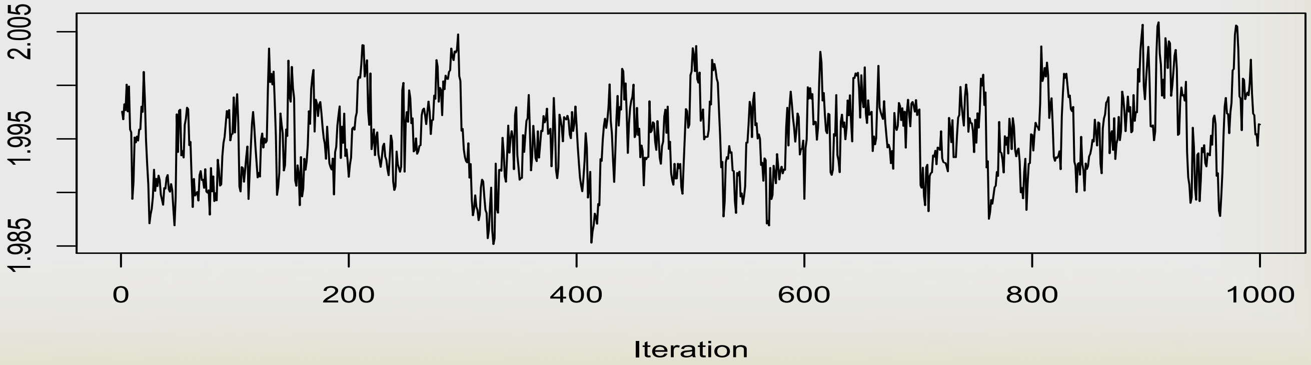
Severe

(SSe ~ 2)



Moderate

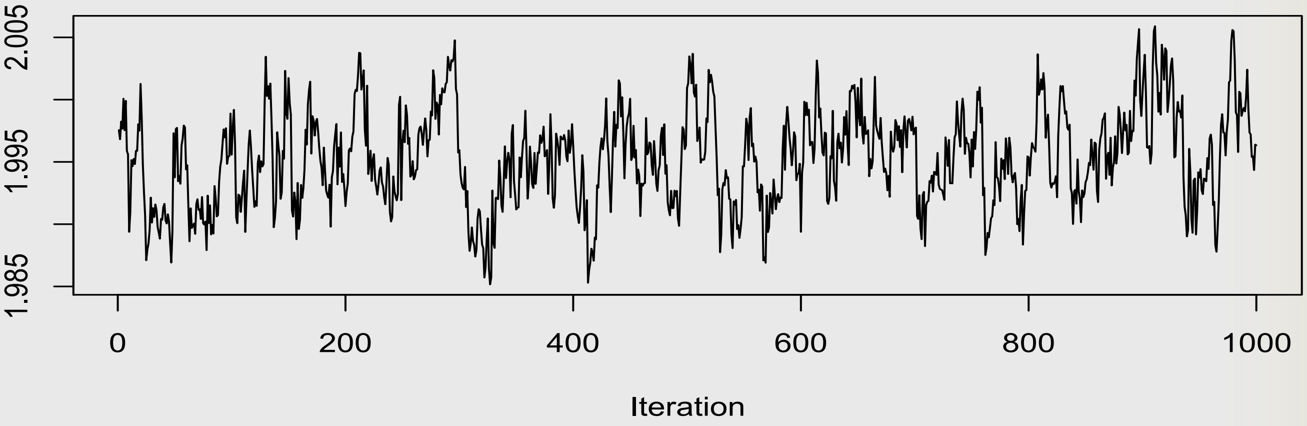
(SSe ~ 90)



Take More Samples!

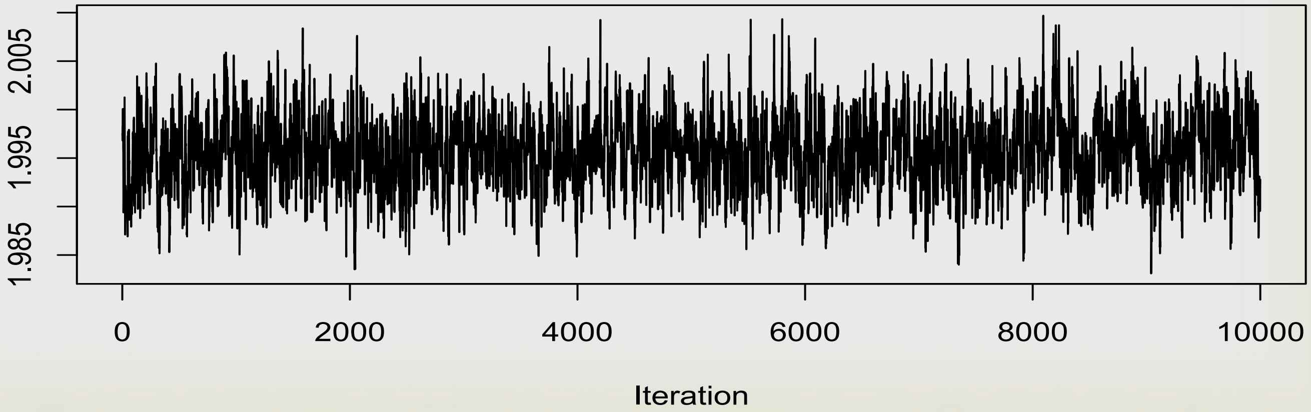
1000 iters

SSe ~ 90



10000 iters

SSe ~ 843



Thinning

- What if we need such a huge number of samples that we run out of memory?

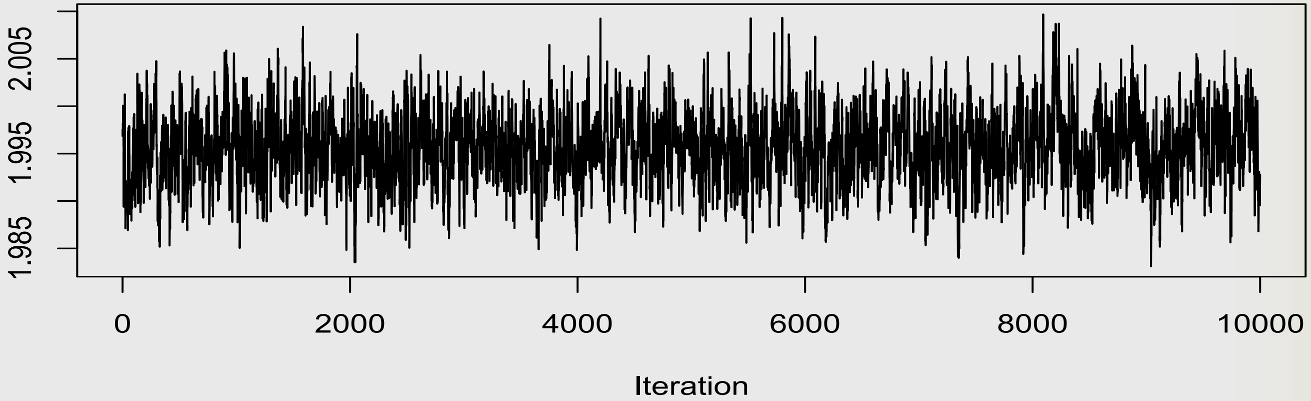
- We could thin the chains:
  - runjags gives us the option to thin during sampling
    - The ‘thin’ argument
  - We can also thin an existing MCMC chain:
    - combine.mcmc has ‘thin’ and ‘return.samples’ arguments

- This is often touted as a way of reducing autocorrelation, but in fact is JUST a way to reduce the number of MCMC samples our computer needs to store!

Thinning

10000 iters

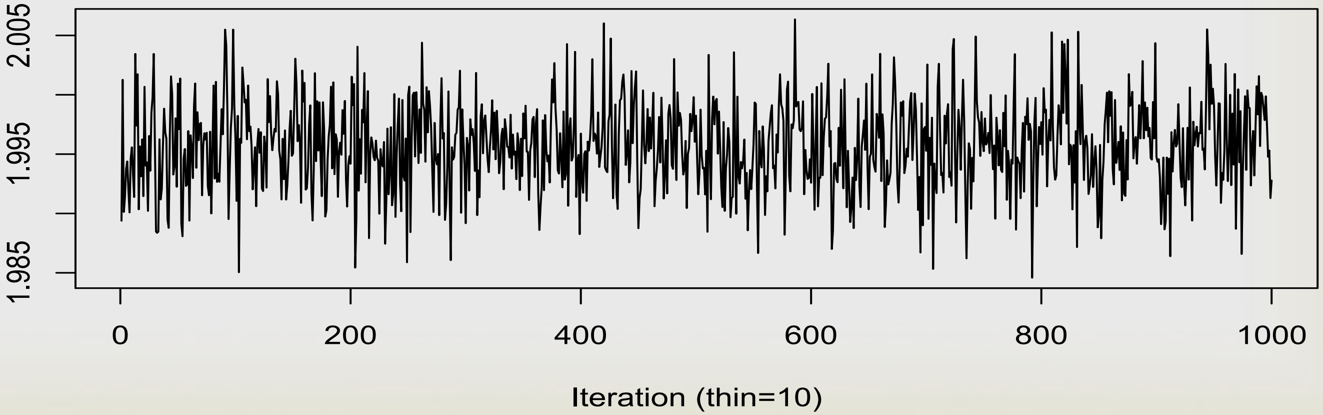
SSe ~ 843



10000 iters

(thin=10)

SSe ~ 740



Thinning

- Notice that we now have less autocorrelation in our thinned chains and therefore a better effective sample size (740) compared to the sample from 1000 iterations (90)

- But it is still not as good as it was before thinning (843) so don’t do it unless you have to!

- Can be useful when:
  - You have to take millions of samples and can’t store them
    - Use run.jags ‘thin’ argument of 10 or 100 or 1000
  - You want to do some computationally intensive post-processing of the samples
    - Use combine.mcmc ‘return.samples’ argument or 1000 or 10000

Bayesian Model Criticism

Parsimony

- Occam's (or Ockham's) razor;
  - *"entia non sunt multiplicanda praeter necessitatem"*
  - "entities must not be multiplied beyond necessity"
  - The simplest explanation that adequately describes the data should be preferred

- If we add more and more parameters to a model we (should) always get a better and better fit until the model is saturated

- So should we exclude some parameters from the model on the basis that they don't significantly improve model fit?
  - Often this doesn't apply to random effects if we know *a priori* that these are important clustering factors

Assessing Parsimony

- What is the likelihood of the model?
  - Usually use deviance rather than likelihood…

    deviance = - 2 x logLikelihood

    deviance = - 2 x logPosterior
  - There is a 'deviance' monitor built into JAGS – it can be handy for comparing multiple chains' solutions
    - i.e.: #monitor# deviance

- How many parameters are in the model?
  - It is easy for a model with lots of parameters to get a good deviance
  - A model with as many parameters as data points is called the *saturated model*

Bayes Factors

- Back to Bayes' theorem

Posterior α Likelihood x Prior

- Consider the 'parameter' as the model choice
  - Integrating over all parameter values within the model gives an automatic penalty for over-fitting of the entire model

- Calculate the posterior probability of Model A vs Model B given the data by multiplying the integrated likelihood of the data over all model parameter values by the prior belief in Model A vs Model B

- Problems
  - Integrating likelihood of data over all model parameter values!
  - Conceptually believes that one of Model A or B is correct

Frequentist Fit Statistics

- Likelihood ratio test
  - Does adding a new parameter give us a significantly better fit than expected by chance?
- AIC
  - Generalisation of LRT to non-nested models
  - AIC = 2k – 2ln(L)
- BIC
  - Similar to AIC but different parameter penalty

- Can all be used to select from a series of nested models
  - All require knowing how many parameters (*k*) are in each model

Defining a Bayesian *k*

- Consider four stochastic parameters:
  - mean ~ Norm(0, 10^-6)

    # Probably 1 parameter
  - mean ~ Norm(0, 1)

    # Maybe half a parameter?
  - mean ~ Norm(0, 10^6)

    # Roughly zero parameters?
  - mean <- 0

    # Definitely zero parameters!

- Do they allow equal flexibility for ‘mean’ to fit to the data?
  - Does ‘mean’ count as an equal parameter for all models?

p_D

- The ‘effective number of parameters’

- Caveats:
  - Accurate calculation depends on approximate normality in the posteriors
    - NOT POSSIBLE WITH MIXTURE MODELS
  - Not invariant to re-parameterisation
  - Requires sample size of data to be much larger than pD
  - Sometimes comes out negative….
    - Especially if strong prior/data conflict

- There are multiple ways to calculate p_D (and equivalents) with no real consensus on the ‘best’ approach

DIC

- Deviance Information Criterion

- Model deviance:

$$D = -2 \log p(y|\theta)$$

- Posterior mean deviance:

$$\overline{D} = \mathrm{E}\left[D\right]$$

- p_D = E[D] – deviance evaluated at posterior mean of the parameters

$$pD = \overline{D} - D\left(\overline{\theta}\right)$$

- DIC = Goodness of fit + complexity

$$DIC = \overline{D} + pD$$

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & Linde, A. van der. (2002). Bayesian Measures of Model Complexity and Fit. Journal of the Royal Statistical Society. Series B (Statistical Methodology), 64(4), 583-639. Blackwell Publishing for the Royal Statistical Society. Retrieved from http://www.jstor.org/stable/3088806

DIC variants

1. Original Spiegelhalter et al. definition of p_D
   - Used by WinBUGS and OpenBUGS

2. Plummer (2002) definition of p_D
   - Used by rjags and runjags

3. Gelman et al (2004) definition of p_D
   - Easy to calculate from any MCMC output
   - Used by r2jags and others

4. Plummer (2008) definition of penalized expected deviance (PED)
   - Also used by rjags and runjags

Plummer, M. (2002), Discussion of the paper by Spiegelhalter et al. Journal of the Royal Statistical Society Series B 64, 620.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian Data Analysis* (2nd ed.). Chapman and Hall/CRC.

Plummer, M. (2008) Penalized loss functions for Bayesian model comparison. Biostatistics doi: 10.1093/biostatistics/kxm049

Interpreting DIC

- Smaller is better
  - Smaller deviance and/or fewer parameters

- Only makes sense as a relative comparison of parameters FOR THE SAME DATA

- Rule of thumb:
  - A difference in DIC of < 5 is probably meaningless
  - A difference in DIC of 5-10 suggests the preferred model
  - A difference in DIC of >10 is more conclusive

- It is the ABSOLUTE DIFFERENCE that matters
  - The specific values and/or ratios are meaningless

Interpreting DIC

- Clearest for nested models with normal distributions
  - More hazy for very different candidate models

- Only ever a probability – never a certainty
  - The model deviance is a Monte Carlo approximation
  - All models are wrong anyway
  - Always look for potentially important differences in posterior inference from competing models
  - Always include variables on biological plausibility first

- Can be improved by priors that match the data!!!

- Always ask the question:
  - Are the posteriors for the common parameters of interest affected?
  - Would I be making a very different decision based on model A vs model B?

Using DIC

```
model{
        for(i in 1:N){
                Height[i] ~ dnorm(expected[i], tau)
                expected[i] <- intercept +
                   Weight[i]*weighteffect + sexeffect[Sex[i]]
        }
        intercept ~ dnorm(90, 0.01)
        weighteffect ~ dnorm(1, 4)
        sexeffect[1] <- 0
        sexeffect[2] ~ dunif(0, 100)
#monitor# ..., sexeffect,tau, deviance, dic, ped
}
```

The full model

```
results <- run.jags (“JAGSmodel.txt”)
results
# or:
# extract(results, ‘dic’)
plot(results, vars=“deviance”, type=“trace”)
```

Using DIC

```
JAGS model summary statistics from 20000 samples (chains=2; burnin=5000):

                                    Lower95  Median Upper95    Mean      SD
deviance                              394.95  419.58  440.23   418.6 11.502
####  OTHERWISE AS USUAL  ####

Model fit assessment:
DIC = 447.1893  (range between chains: 447.1592 - 447.2194)
PED = 506.562  (range between chains: 506.5319 - 506.5921)
Estimated effective number of parameters:
                          pD = 28.58059, pOpt = 87.95329

Total time taken: 1.2 minutes
```

- Remember:
  - The deviance is numerically approximated (i.e. a Monte Carlo estimate)!
  - As for AIC, a smaller DIC is better
    - BUT a difference of < 5 is marginal
    - A difference of between 5 and 10 is ‘suggestive’, > 10 is ‘conclusive’
  - There is a subtle difference between PED and the ‘standard’ DIC
    - Although they usually agree…

Using DIC

```
model{
       for(i in 1:N){
          Height[i] ~ dnorm(expected[i], tau)
          expected[i] <- intercept +
              Weight[i]*weighteffect + sexeffect[Sex[i]]
       }
       intercept ~ dnorm(90, 0.01)
       weighteffect <- 0
       sexeffect[1] <- 0
       sexeffect[2] <- 0
#monitor# intercept, weighteffect, sexeffect,tau, dic, ped
}

results <- run.jags (“JAGSmodel.txt”)
results
# or:
# extract(results, ‘dic’)
plot(results, vars=“deviance”, type=“trace”)
```

The simplest model

Using DIC

```
model{
        for(i in 1:N){
                Height[i] ~ dnorm(expected[i], tau)
                expected[i] <- intercept

        }
        intercept ~ dnorm(90, 0.01)


        #monitor# intercept, tau, dic, ped
}

results <- run.jags (“JAGSmodel.txt”)
results
plot(results, vars=“deviance”, type=“trace”)
```

The simplest model (equivalent)

Using DIC

```
model{
       for(i in 1:N){
              Height[i] ~ dnorm(expected[i], tau)
              expected[i] <- intercept +
                 Weight[i]*weighteffect + sexeffect[Sex[i]]
       }
       intercept ~ dnorm(90, 0.01)
       weighteffect ~ dnorm(1, 4)
       sexeffect[1] <- 0
       sexeffect[2] <- 0
#monitor# intercept, weighteffect, sexeffect,tau, dic, ped
}

results <- run.jags (“JAGSmodel.txt”)
results
plot(results, vars=“deviance”, type=“trace”)
```

An intermediate model (nested)

Using DIC

```
model{
       for(i in 1:N){
              Height[i] ~ dnorm(expected[i], tau)
              expected[i] <- intercept +
                 Weight[i]*weighteffect + sexeffect[Sex[i]]
       }
       intercept ~ dnorm(90, 0.01)
       weighteffect <- 0
       sexeffect[1] <- 0
       sexeffect[2] ~ dunif(0, 100)
#monitor# intercept, weighteffect, sexeffect,tau, dic, ped
}

results <- run.jags (“JAGSmodel.txt”)
results
plot(results, vars=“deviance”, type=“trace”)
```

Another intermediate model (nested)

Alternatives to DIC?

- Posterior predictive p-values
  - Take our parameter estimates and see if we can replicate the data
  - Preferably some aspect of the data that isn't modelled

- Bayesian Model Averaging
  - Average parameter estimates over all competing models
  - Needs some weighting of belief about models
    - Usually Bayes Factors

Alternatives to DIC?

- Reversible Jump MCMC
  - Introduce a MH step to switch between models
  - Currently can’t be done in BUGS
  - Convergence can be a nightmare!
  - Approximation: variable (de)activation using bernoulli selection

- Stochastic Variable Selection
  - Clever method for estimating parameter/model support within a single model run (similar principle to rjMCMC)
  - Convergence can be difficult

- Cross-validation
- WAIC

Cross-Validation

```
model{
    for(i in 1:N){
        Data[i] ~ dnorm(mu[i], tau)
        pred.data[i] ~ dnorm(mu[i], tau)
        mu[i] <- …..
    }
    #monitor# pred.data
}
```

```
Data <- real.data
for(i in 1:N){
    Data[i] <- NA       # Or multiple Data at once if preferred
    results <- run.jags(‘model.txt’, n.chains=2)
    plot(apply(as.mcmc(results),2,mean), real.data)
    summary(mean(as.mcmc(results)[,i]) - real.data)
}

# or:
?drop.k.jags
```

Cross-Validation

- Advantages
  - Robust
- Disadvantages
  - Computational cost
- Leave One Out (LOO) is approximated by WAIC

WAIC

- Widely Applicable Information Criterion

- Similar use/interpretation to DIC but with fewer drawbacks:
  - Theory is better understood (approximation to LOO)
  - WAIC is valid for singular models e.g. mixture models

- Requires the 'focus' of interest to be specified explicitly
  - Allows more specific 'tailoring' of the precise aspect of the model fit that we are assessing
  - Also requires an extra bit of thinking

WAIC

- Calculation is based on the mean and variance of the individual data-point contributions to the likelihood

- See also:

  Vehtari and Gelman, 2014:

  WAIC and cross-validation in Stan

  Vehtari, Gelman and Gaby, 2016:

  Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC

WAIC in JAGS

[currently in development]

- We need to specify the log likelihood (density) of interest:
  - This allows us to explicitly control the focus of the WAIC

```
for(i in 1:N){
       Obs[i] ~ dpois(lambda[i])
       log(lambda[i]) <- ...

       # To monitor the variance of the log likelihood:
       logdens_Obs[i] <- logdensity.pois(Obs[i], lambda[i])
       # And the mean of the likelihood:
       density_Obs[i] <- exp(logdens_Obs[i])
}
```

- Some additional R code is then needed: see the waic_example.R file

- NB: All distributions have a corresponding logdensity function
  - But JAGS 5 will remove the requirement for calculating logdens_Obs…

Model Adequacy

Residuals

```
model{
   for(i in 1:N){
      Data[i] ~ dnorm(mu[i], tau)
      mu[i] <- …..

      residual[i] <- Data[i] – mu[i]
      std.residual[i] <- (Data[i] – mu[i])
                           / (1/sqrt(tau))
   }
   #monitor# mu, residual, std.residual
}
```

But you will get a distribution of residuals for each data point!

Prediction

- As easy as monitoring a new variable!

```
for(i in 1:N){
      Data[i] ~ dnorm(mu[i], tau)
      predicted[i] ~ dnorm(mu[i], tau)
}
for(i in (N+1):(N*2)){
      predicted[i] ~ dnorm(mu[i], tau)
}
```

- Take great care with predictive models – model criticism techniques are quite different
  - DIC may not be suitable
  - Better to fit against ½ the dataset and test against the rest

Simulation Studies

- Does my model formulation do what I think it does?
  - ie. Can it retrieve good parameter estimates for data generated from the same model?

- Check it using simulated data
  - Simulate a dataset with known parameter values
    - From BUGS code by removing data, fixing parameter values as data and monitoring data
    - Using independent code in R – probably better
  - Run the model using this data
  - Compare known parameter values with estimates
  - Repeat 1000 times or so - see **`?run.jags.study`**