



Examples Volume 3

[Camel: multivariate normal with structured missing data](#)

[Eye Tracking: dirichlet process prior](#)

[Fun shapes: general constraints](#)

[Hepatitis: random effects model with measurement error](#)

[Hips1: deterministic cost-effectiveness model](#)

[Hips2: stochastic cost-effectiveness model](#)

[Hips3: cost-effectiveness model](#)

[Hips4: cost-effectiveness model](#)

[Jama: radio carbon dating with phase information](#)

[Pigs smoothed histogram](#)

[Pines: bayes factors using pseudo priors](#)

[St Veit: radio carbon dating with stratification](#)

References:

Sorry - an on-line version of the references is currently unavailable.
Please refer to the existing Examples documentation available from
<http://www.mrc-bsu.cam.ac.uk/bugs>.



Camel: Multivariate normal with structured missing data

Tanner and Wong present a data set with missing values modeled as a bivariate normal. No closed form for the deviance of the model can be found and so no deviance node is created by the BUGS compiler. For the same reason the DIC menu is greyed-out

```

model
{
  for (i in 1 : N){
    Y[i, 1 : 2] ~ dnorm(mu[], tau[ , ])
  }
  mu[1] <- 0
  mu[2] <- 0
  tau[1 : 2, 1 : 2] ~ dwish(R[ , ], 2)
  R[1, 1] <- 0.001
  R[1, 2] <- 0
  R[2, 1] <- 0;
  R[2, 2] <- 0.001
  Sigma2[1 : 2, 1 : 2] <- inverse(tau[ , ])
  rho <- Sigma2[1, 2] / sqrt(Sigma2[1, 1] * Sigma2[2, 2])
}

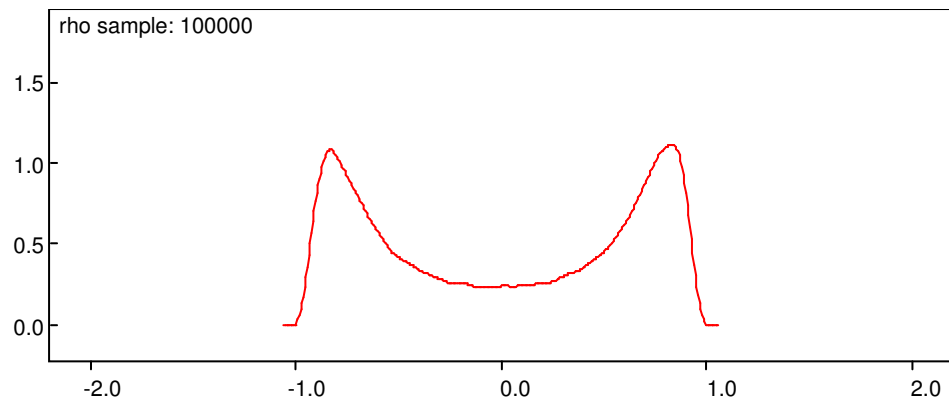
```

[Data](#) (click to open)

[Inits](#) (click to open)

Results

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
Sigma2[1,1]	3.193	2.081	0.01182	1.117	2.649	8.461	1001	100000
Sigma2[1,2]	0.04932	2.473	0.03685	-4.632	0.1154	4.667	1001	100000
Sigma2[2,1]	0.04932	2.473	0.03685	-4.632	0.1154	4.667	1001	100000
Sigma2[2,2]	3.209	2.164	0.01242	1.116	2.656	8.612	1001	100000
rho	0.01742	0.6582	0.01114	-0.9065	0.05492	0.9074	1001	100000
tau[1,1]	0.8593	0.5117	0.00325	0.2247	0.7394	2.179	1001	100000
tau[1,2]	-0.01712	0.7092	0.01099	-1.417	-0.02341	1.385	1001	100000
tau[2,1]	-0.01712	0.7092	0.01099	-1.417	-0.02341	1.385	1001	100000
tau[2,2]	0.8584	0.512	0.003143	0.2217	0.7406	2.172	1001	100000





Eye Tracking: dirichlet process prior

Adapted from Congdon (2001), example 6.27, page 263.

```

model{
  for( i in 1 : N ) {
    S[i] ~ dcat(pi[])
    mu[i] <- theta[S[i]]
    x[i] ~ dpois(mu[i])
    for( j in 1 : C ) {
      SC[i, j] <- equals(j, S[i])
    }
  }
  # Precision Parameter
  alpha <- 1
  # alpha~ dgamma(0.1,0.1)
  # Constructive DPP
  p[1] <- r[1]
  for( j in 2 : C ) {
    p[j] <- r[j] * (1 - r[j] - 1) * p[j - 1] / r[j] - 1]
  }
  p.sum <- sum(p[])
  for( j in 1:C){
    theta[j] ~ dgamma(A, B)
    r[j] ~ dbeta(1, alpha)
  }
  # scaling to ensure sum to 1
  pi[j] <- p[j] / p.sum
}
# hierarchical prior on theta[i] or preset parameters
A ~ dexp(0.1)  B ~dgamma(0.1, 0.1)
# A <- 1 B <- 1
# total clusters
K <- sum(cl[])
for( j in 1 : C ) {
  sumSC[j] <- sum(SC[ , j])
  cl[j] <- step(sumSC[j] -1)
}
}

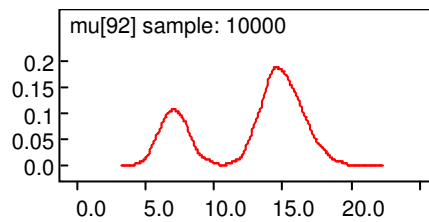
```

[Data](#) (click to open)

Results

a) fixed A and B, fixed $\alpha=1$, $C=10$ (max categories)

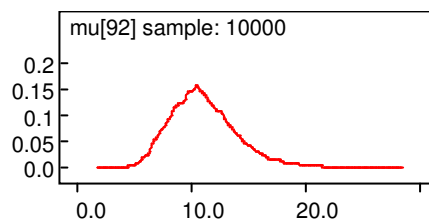
	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
K	8.498	0.9973	0.02544	7.0	9.0	10.0	1001	10000
deviance	295.7	15.34	0.4019	265.5	295.7	326.2	1001	10000
mu[92]	12.7	3.877	0.06945	5.72	14.17	18.0	1001	10000



Notice prior and data conflict.

b) variable A and B, fixed $\alpha=1$, $C=10$ (max categories)

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
A	0.7106	0.4467	0.02279	0.1639	0.6048	1.839	1001	10000
B	0.08661	0.06294	0.002122	0.008353	0.07175	0.2469	1001	10000
K	7.337	1.416	0.07718	5.0	7.0	10.0	1001	10000
deviance	279.9	16.83	0.7394	247.2	280.0	312.1	1001	10000
mu[92]	11.15	3.005	0.0478	6.275	10.8	18.11	1001	10000





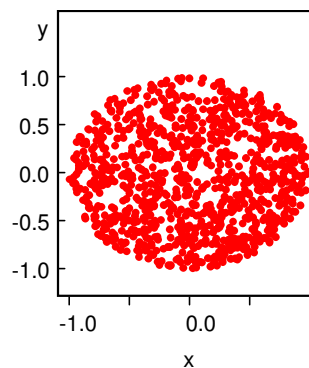
Fun shapes: general constraints

These toy examples illustrate how to implement general inequality constraints. They generate points uniformly over a restricted region of the square with corners at (1, 1) and (-1, -1). A dummy bernoulli variable is introduced and its corresponding proportion set to the step function of the required constraint. The only useful example is the parallelogram where $x + y$ is constrained to be less than or equal to one, this idea can be used to model proportions for binary data instead of say logistic regression.

Circle

```
model
{
  x ~ dunif(-1, 1)
  y ~ dunif(-1, 1)
  O <- 0
  O ~ dbern(constraint)
  constraint <- step(x * x + y * y - 1)
}
```

[Inits](#) (click to open)



Square minus circle

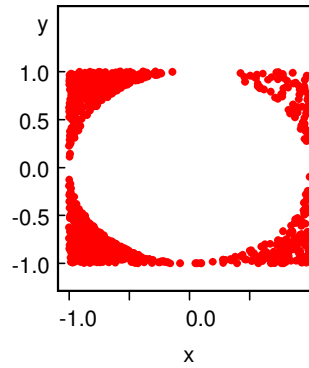
```
model
{
  x ~ dunif(-1, 1)
  y ~ dunif(-1, 1)
  O <- 1
```

```

O ~ dbern(constraint)
constraint <- step(x * x + y * y - 1)
}

```

[Inits](#) (click to open)



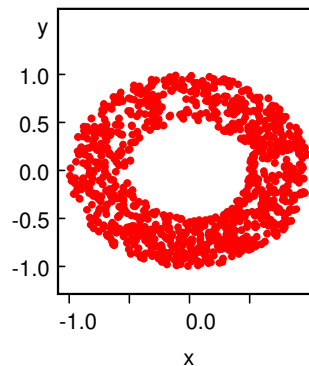
Ring

```

model
{
  x ~ dunif(-1, 1)
  y ~ dunif(-1, 1)
  O1 <- 0
  O1 ~ dbern(constraint1)
  constraint1 <- step(x * x + y * y - 1)
  O2 <- 1
  O2 ~ dbern(constraint2)
  constraint2 <- step(x * x + y * y - 0.25)
}

```

[Inits](#) (click to open)



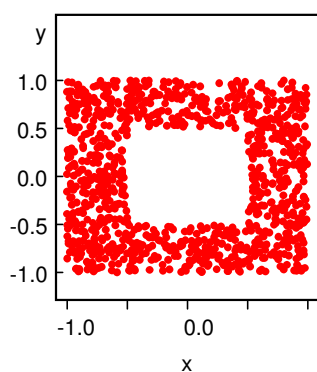
Hollow square

```

model
{
  x ~ dunif(-1, 1)
  y ~ dunif(-1, 1)
  O <- 0
  O ~ dbern(constraint)
  constraint <- (step(0.5 - abs(x)) * step(0.50 - abs(y)))
}

```

[Inits](#) (click to open)



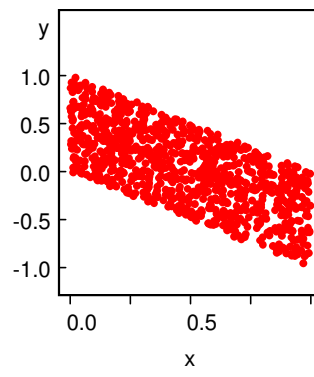
Parallelogram

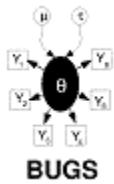
```

model
{
  x ~ dunif(0, 1)
  y ~ dunif(-1, 1.0)
  O1 <- 1
  O1 ~ dbern(constraint1)
  constraint1 <- step(x + y)
  O2 <- 0
  O2 ~ dbern(constraint2)
  constraint2 <- step(x + y - 1)
}

```

[Inits](#) (click to open)





Hepatitis: a normal hierarchical model with measurement error

This example is taken from Spiegelhalter et al (1996) (chapter in *Markov Chain Monte Carlo in Practice*) and concerns 106 children whose post-vaccination anti Hb titre was measured 2 or 3 times. Both measurements and times have been transformed to a log scale. One covariate $y_0 =$ log titre at baseline, is available.

The model is essentially a random effects linear growth curve

$$Y_{ij} \sim \text{Normal}(\alpha_i + \beta_i (t_{ij} - t_{\text{bar}}), \tau)$$

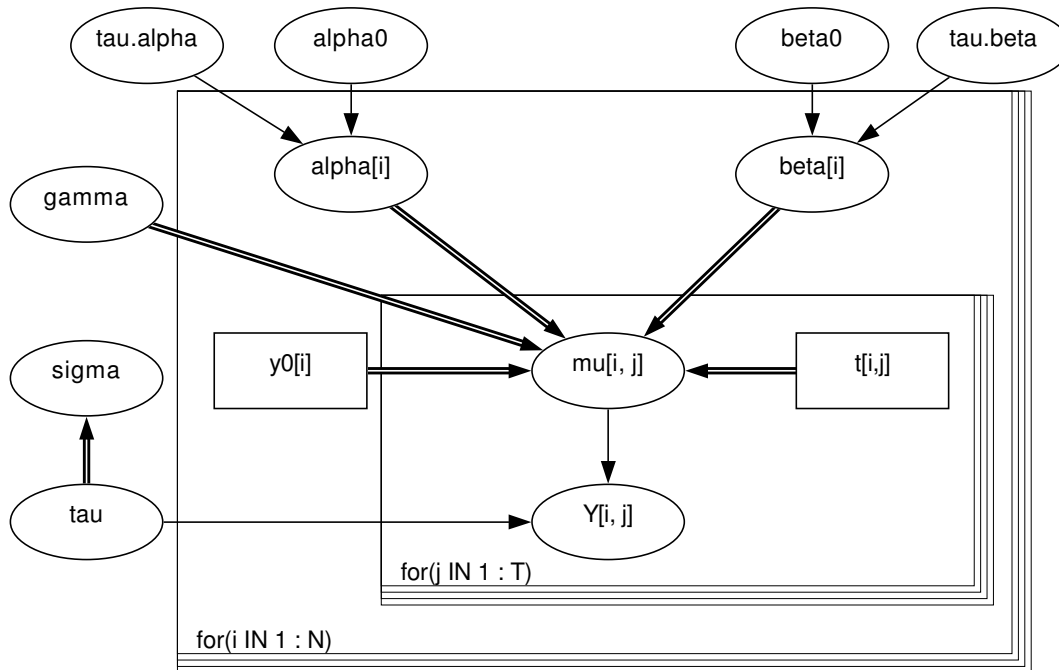
$$\alpha_i \sim \text{Normal}(\alpha_C, \tau_\alpha)$$

$$\beta_i \sim \text{Normal}(\beta_C, \tau_\beta)$$

where τ represents the *precision* (1/variance) of a normal distribution. We note the absence of a parameter representing correlation between α_i and β_i unlike in Gelfand *et al* 1990. However, see the `Birats` example in Volume 2 which does explicitly model the covariance between α_i and β_i .

$\alpha_C, \tau_\alpha, \beta_C, \tau_\beta, \tau$ are given independent "noninformative" priors.

Graphical model for hep example:



BUGS language for hep example:

```

model
{
  for( i in 1 : N ) {
    for( j in 1 : T ) {
      Y[i , j] ~ dnorm(mu[i , j],tau)
      mu[i , j] <- alpha[i] + beta[i] * (t[i,j] - 6.5) +
        gamma * (y0[i] - mean(y0[]))
    }
    alpha[i] ~ dnorm(alpha0,tau.alpha)
    beta[i] ~ dnorm(beta0,tau.beta)
  }
  tau ~ dgamma(0.001,0.001)
  sigma <- 1 / sqrt(tau)
  alpha0 ~ dnorm(0.0,1.0E-6)
  tau.alpha ~ dgamma(0.001,0.001)
  beta0 ~ dnorm(0.0,1.0E-6)
  tau.beta ~ dgamma(0.001,0.001)
  gamma ~ dnorm(0.0,1.0E-6)
}

```

Note the use of a very flat but conjugate prior for the population effects: a locally uniform prior could also have been used.

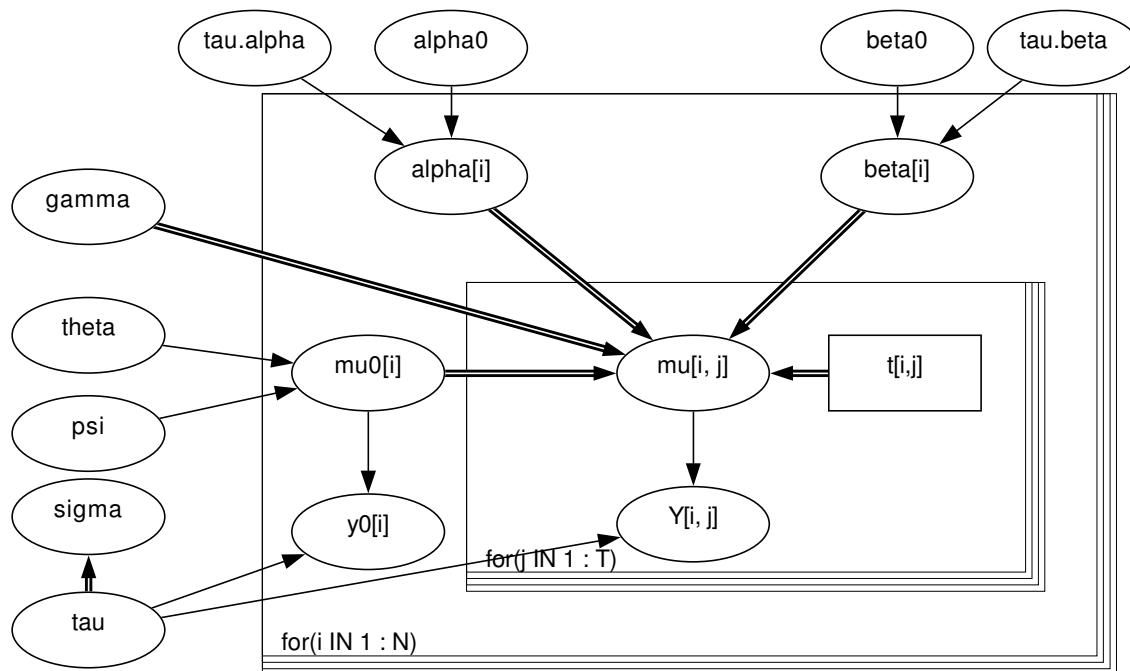
[Data](#) (click to open)

[Inits](#) (click to open)

Results

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
alpha0	6.138	0.1503	0.001816	5.846	6.137	6.436	1001	10000
beta0	-1.064	0.1206	0.008653	-1.315	-1.058	-0.8301	1001	10000
gamma	0.6691	0.08149	0.00228	0.5065	0.6687	0.8284	1001	10000
sigma	1.003	0.05471	0.001282	0.8997	1.001	1.115	1001	10000

With measurement error



```

model
{
  tau.alpha ~ dgamma(0.001,0.001)
  alpha0 ~ dnorm( 0.0,1.0E-6)
  beta0 ~ dnorm( 0.0,1.0E-6)
  tau.beta ~ dgamma(0.001,0.001)
  for( i in 1 : N ) {
    alpha[i] ~ dnorm(alpha0,tau.alpha)
    beta[i] ~ dnorm(beta0,tau.beta)
  }
}
    
```

```

y0[i] ~ dnorm(mu0[i],tau)
mu0[i] ~ dnorm(theta,psi)
}
for(j in 1 : T) {
  for(i in 1 : N) {
    Y[i , j] ~ dnorm(mu[i , j],tau)
    mu[i , j] <- alpha[i] + beta[i] * (t[i , j] - 6.5) +
      gamma * (mu0[i] - mean(y0[]))
  }
}
tau ~ dgamma(0.001,0.001)
sigma <- 1 / sqrt(tau)
gamma ~ dnorm( 0.0,1.0E-6)
theta ~ dnorm( 0.0,1.0E-6)
psi ~ dgamma(0.001,0.001)
}

```

[Data](#) (click to open)

[Inits for measurement error model](#) (click to open)

Results

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
alpha0	6.129	0.1586	0.004995	5.812	6.125	6.434	1001	10000
beta0	-1.075	0.1203	0.009152	-1.313	-1.063	-0.8381	1001	10000
gamma	1.082	0.2187	0.01668	0.7441	1.055	1.619	1001	10000
sigma	1.029	0.06513	0.003905	0.9147	1.024	1.166	1001	10000



Hips model 1: Closed form estimates for each strata - give results for "exact" columns in Table 2

Spiegelhalter, D.J. and Best, N.G. "Bayesian approaches to multiple sources of evidence and uncertainty in complex cost-effectiveness modelling". *Statistics in Medicine* 22, (2003), 3687-3709.

No updates are required for this model; just load data and compile, then obtain values for C, mean.C, sd.C, BL, mean.BL, sd.BL, BQ, mean.BQ and sd.BQ using the node tool from the Info menu.

```

model {

  for(k in 1 : K) { # loop over strata

    # Cost and benefit equations in closed form:
    #####

    # Costs
    for(t in 1 : N) {
      ct[k, t] <- inprod(pi[k, t, ], c[]) / pow((1 + delta.c), t - 1)
    }
    C[k] <- C0 + sum(ct[k, ])

    # Benefits - life expectancy
    for(t in 1:N) {
      blt[k, t] <- inprod(pi[k, t, ], bl[]) / pow((1 + delta.b), t - 1)
    }
    BL[k] <- sum(blt[k, ])

    # Benefits - QALYs
    for(t in 1 : N) {
      bqt[k, t] <- inprod(pi[k,t, ], bq[]) / pow((1 + delta.b), t - 1)
    }
    BQ[k] <- sum(bqt[k, ])

    # Markov model probabilities:
    #####

    # Transition matrix
    for(t in 2 : N) {
      Lambda[k, t, 1, 1] <- 1 - gamma[k, t] - lambda[k, t]
      Lambda[k, t, 1, 2] <- gamma[k, t] * lambda.op
      Lambda[k, t, 1, 3] <- gamma[k, t] * (1 - lambda.op)
      Lambda[k, t, 1, 4] <- 0
      Lambda[k, t, 1, 5] <- lambda[k, t]

      Lambda[k, t, 2, 1] <- 0

```

```

Lambda[k, t, 2, 2] <- 0
Lambda[k, t, 2, 3] <- 0
Lambda[k, t, 2, 4] <- 0
Lambda[k, t, 2, 5] <- 1

Lambda[k, t, 3, 1] <- 0
Lambda[k, t, 3, 2] <- 0
Lambda[k, t, 3, 3] <- 0
Lambda[k, t, 3, 4] <- 1 - lambda[k, t]
Lambda[k, t, 3, 5] <- lambda[k, t]

Lambda[k, t, 4, 1] <- 0
Lambda[k, t, 4, 2] <- rho * lambda.op
Lambda[k, t, 4, 3] <- rho * (1 - lambda.op)
Lambda[k, t, 4, 4] <- 1 - rho - lambda[k, t]
Lambda[k, t, 4, 5] <- lambda[k,t]

Lambda[k, t, 5, 1] <- 0
Lambda[k, t, 5, 2] <- 0
Lambda[k, t, 5, 3] <- 0
Lambda[k, t, 5, 4] <- 0
Lambda[k, t, 5, 5] <- 1

gamma[k, t] <- h[k] * (t - 1)
}

# Marginal probability of being in each state at time 1
pi[k,1,1] <- 1 - lambda.op pi[k,1, 2]<-0 pi[k,1,3] <- 0
pi[k,1, 4] <- 0 pi[k,1, 5] <- lambda.op

# Marginal probability of being in each state at time t>1
for(t in 2:N) {
  for(s in 1:S) {
    pi[k, t, s] <- inprod(pi[k, t - 1, ], Lambda[k, t, , s])
  }
}

}

# Mean and sd of costs and benefits over strata
#####

mean.C <- inprod(p.strata[], C[])
for(k in 1:12) {
  dev.C[k] <- pow(C[k] - mean.C, 2)
}
var.C <- inprod(p.strata[], dev.C[])
sd.C <- sqrt(var.C)

mean.BL <- inprod(p.strata[], BL[])
for(k in 1:12) {
  dev.BL[k] <- pow(BL[k] - mean.BL, 2)
}
var.BL <- inprod(p.strata[], dev.BL[])
sd.BL <- sqrt(var.BL)

```

```

mean.BQ <- inprod(p.strata[], BQ[])
for(k in 1:12) {
  dev.BQ[k] <- pow(BQ[k] - mean.BQ, 2)
}
var.BQ <- inprod(p.strata[], dev.BQ[])
sd.BQ <- sqrt(var.BQ)
}

```

[Data](#) (click to open)

Results

C[1]	5781.0
C[2]	5417.0
C[3]	4989.0
C[4]	4466.0
C[5]	4263.0
C[6]	4193.0
C[7]	5626.0
C[8]	5350.0
C[9]	5002.0
C[10]	4487.0
C[11]	4282.0
C[12]	4212.0
mean.C	4603.0
sd.C	403.1
BL[1]	14.48
BL[2]	12.7
BL[3]	10.34
BL[4]	7.737
BL[5]	5.405
BL[6]	4.101
BL[7]	15.14
BL[8]	13.7
BL[9]	11.65
BL[10]	9.1
BL[11]	6.46
BL[12]	4.988
mean.BL	8.687
sd.BL	2.591
BQ[1]	13.17
BQ[2]	11.59
BQ[3]	9.471
BQ[4]	7.158
BQ[5]	5.019
BQ[6]	3.813
BQ[7]	13.82
BQ[8]	12.53
BQ[9]	10.7
BQ[10]	8.431
BQ[11]	6.004
BQ[12]	4.641
mean.BQ	8.016
sd.BQ	2.338



Hips model 2: MC estimates for each strata - give results for "Monte Carlo" columns in Table 2

Spiegelhalter, D.J. and Best, N.G. "Bayesian approaches to multiple sources of evidence and uncertainty in complex cost-effectiveness modelling". *Statistics in Medicine* 22, (2003), 3687-3709.

$n = 10000$ updates (1 per simulated patient) are required for this model; monitor C, mean.C, BL, mean.BL, BQ, mean.BQ.

Sections of the code that have changed from Model 1 are shown in bold

```

model {

  for(k in 1 : K) { # loop over strata

    # Cost and benefit equations
    #####

    # Costs
    for(t in 1 : N) {
      ct[k, t] <- inprod(y[k, t, ], c[]) / pow(1 + delta.c, t - 1)
    }
    C[k] <- C0 + sum(ct[k, ])

    # Benefits - life expectancy
    for(t in 1 : N) {
      blt[k, t] <- inprod(y[k, t, ], bl[]) / pow(1 + delta.b, t - 1)
    }
    BL[k] <- sum(blt[k, ])

    # Benefits - QALYs
    for(t in 1:N) {
      bqt[k, t] <- inprod(y[k, t, ], bq[]) / pow(1 + delta.b, t - 1)
    }
    BQ[k] <- sum(bqt[k, ])

    # Markov model probabilities:
    #####

    # Transition matrix
    for(t in 1 : N) {
      Lambda[k, t, 1, 1] <- 1 - gamma[k, t] - lambda[k, t]
      Lambda[k, t, 1, 2] <- gamma[k, t] * lambda.op
      Lambda[k, t, 1, 3] <- gamma[k, t] *(1 - lambda.op)
      Lambda[k, t, 1, 4] <- 0
      Lambda[k, t, 1, 5] <- lambda[k, t]
    }
  }
}

```

```

Lambda[k, t, 2, 1] <- 0
Lambda[k, t, 2, 2] <- 0
Lambda[k, t, 2, 3] <- 0
Lambda[k, t, 2, 4] <- 0
Lambda[k, t, 2, 5] <- 1

Lambda[k, t, 3, 1] <- 0
Lambda[k, t, 3, 2] <- 0
Lambda[k, t, 3, 3] <- 0
Lambda[k, t, 3, 4] <- 1 - lambda[k, t]
Lambda[k, t, 3, 5] <- lambda[k, t]

Lambda[k, t, 4, 1] <- 0
Lambda[k, t, 4, 2] <- rho * lambda.op
Lambda[k, t, 4, 3] <- rho * (1 - lambda.op)
Lambda[k, t, 4, 4] <- 1 - rho - lambda[k, t]
Lambda[k, t, 4, 5] <- lambda[k, t]

Lambda[k, t, 5, 1] <- 0
Lambda[k, t, 5, 2] <- 0
Lambda[k, t, 5, 3] <- 0
Lambda[k, t, 5, 4] <- 0
Lambda[k, t, 5, 5] <- 1

gamma[k, t] <- h[k] * (t - 1)
}

# Marginal probability of being in each state at time 1
pi[k, 1, 1] <- 1 - lambda.op pi[k, 1, 2] <- 0 pi[k, 1, 3] <- 0 pi[k, 1, 4] <- 0
pi[k, 1, 5] <- lambda.op

# state of each individual in strata k at time t =1
y[k,1,1 : S] ~ dmulti(pi[k,1, ], 1)

# state of each individual in strata k at time t > 1
for(t in 2 : N) {
  for(s in 1:S) {
    # sampling probabilities
    pi[k, t, s] <- inprod(y[k, t - 1, ], Lambda[k, t, , s])
  }
  y[k, t, 1 : S] ~ dmulti(pi[k, t, ], 1)
}

}

# Mean of costs and benefits over strata
#####

mean.C <- inprod(p.strata[], C[])
mean.BL <- inprod(p.strata[], BL[])
mean.BQ <- inprod(p.strata[], BQ[])

}

```

[Data](#) (click to open)

Results

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
BL[1]	14.52	2.83	0.0317	5.212	15.5	16.91	1	10000
BL[2]	12.72	3.37	0.03352	2.833	13.78	16.31	1	10000
BL[3]	10.28	3.704	0.03809	1.943	11.11	15.37	1	10000
BL[4]	7.733	3.482	0.03394	1.0	7.802	13.78	1	10000
BL[5]	5.436	3.002	0.02759	1.0	5.212	12.16	1	10000
BL[6]	4.15	2.964	0.0296	1.0	3.673	11.48	1	10000
BL[7]	15.14	2.69	0.02495	6.582	15.95	17.13	1	10000
BL[8]	13.73	3.071	0.03025	4.465	14.76	16.81	1	10000
BL[9]	11.62	3.56	0.03392	1.943	12.76	16.05	1	10000
BL[10]	9.047	3.698	0.03798	1.0	9.853	15.08	1	10000
BL[11]	6.432	3.333	0.03159	1.0	5.917	13.78	1	10000
BL[12]	5.0	3.491	0.03316	1.0	4.465	13.04	1	10000
BQ[1]	13.19	2.574	0.02905	4.889	13.97	15.54	1	10000
BQ[2]	11.6	3.049	0.03095	2.658	12.54	14.96	1	10000
BQ[3]	9.421	3.358	0.03402	1.823	10.05	14.0	1	10000
BQ[4]	7.158	3.199	0.03128	0.938	7.318	12.71	1	10000
BQ[5]	5.049	2.768	0.02556	0.938	4.889	11.25	1	10000
BQ[6]	3.858	2.733	0.02729	0.938	3.445	10.53	1	10000
BQ[7]	13.82	2.467	0.02308	6.174	14.54	15.82	1	10000
BQ[8]	12.57	2.797	0.02713	4.188	13.38	15.44	1	10000
BQ[9]	10.66	3.242	0.03061	1.823	11.61	14.76	1	10000
BQ[10]	8.384	3.403	0.03406	0.938	9.242	14.0	1	10000
BQ[11]	5.978	3.077	0.02915	0.938	5.55	12.59	1	10000
BQ[12]	4.655	3.228	0.03051	0.938	4.188	12.0	1	10000
C[1]	5816.0	1925.0	18.04	4052.0	5359.0	10730.0	1	10000
C[2]	5426.0	1849.0	16.93	4052.0	4630.0	10350.0	1	10000
C[3]	4978.0	1660.0	18.1	4052.0	4052.0	9635.0	1	10000
C[4]	4451.0	1188.0	11.49	4052.0	4052.0	8242.0	1	10000
C[5]	4263.0	940.7	8.443	4052.0	4052.0	7781.0	1	10000
C[6]	4193.0	769.7	7.25	4052.0	4052.0	7006.0	1	10000
C[7]	5648.0	1844.0	15.12	4052.0	5087.0	10580.0	1	10000
C[8]	5329.0	1758.0	19.43	4052.0	4665.0	10200.0	1	10000
C[9]	5022.0	1667.0	17.31	4052.0	4052.0	9711.0	1	10000
C[10]	4474.0	1216.0	13.52	4052.0	4052.0	8242.0	1	10000
C[11]	4280.0	945.8	8.742	4052.0	4052.0	7781.0	1	10000
C[12]	4200.0	767.9	6.963	4052.0	4052.0	7006.0	1	10000
mean.BL	8.667	1.396	0.01388	5.915	8.672	11.4	1	10000
mean.BQ	7.998	1.287	0.01262	5.45	8.002	10.5	1	10000
mean.C	4600.0	463.0	4.481	4091.0	4469.0	5813.0	1	10000

Overall SD for Monte Carlo estimates at bottom of Table 2 is just the weighted SD of the strata-specific Monte Carlo means



Hips model 3: MC estimates for each strata, allowing for parameter uncertainty in revision hazard, h - gives results for Table 3

Spiegelhalter, D.J. and Best, N.G. "Bayesian approaches to multiple sources of evidence and uncertainty in complex cost-effectiveness modelling". *Statistics in Medicine* 22, (2003), 3687-3709.

$n = 10000$ updates (1 per simulated set of parameter values) are required for this model; monitor C, BL, BQ to get posterior mean and sd for each subgroup for results in top part of Table 3.

For results in bottom part of Table 3:

Approach 1: $p(s, \theta) = p(s | \theta) p(\theta)$

m_{θ} and v_{θ} are the mean and variance across subgroups for a given value of θ
 \Rightarrow within BUGS code, calculate mean and variance of $C[k]$, $BL[k]$, $BQ[k]$ across subgroups at each iteration, then take Monte Carlo expectation at end of run
 \Rightarrow monitor mean.C, mean.BL, mean.BQ, var.C, var.BL, var.BQ

Approach 2: $p(s, \theta) = p(\theta | s) p(s)$

overall mean, m = weighted mean of posterior means of $C[k]$, $BL[k]$, $BQ[k]$ \Rightarrow calculate after BUGS run var due to uncertainty, $vP2$ = weighted mean of posterior variances of $C[k]$, $BL[k]$, $BQ[k]$ \Rightarrow calculate after BUGS run var due to heterogeneity = $vH2$ = weighted variance of posterior means of $C[k]$, $BL[k]$, $BQ[k]$ \Rightarrow calculate after BUGS run

Sections of the code that have changed from Model 1 are shown in bold

```

model {

  for(k in 1 : K) { # loop over strata

    # Cost and benefit equations
    #####

    # Costs
    for(t in 1 : N) {
      ct[k, t] <- inprod(pi[k, t, ], c[]) / pow(1 + delta.c, t - 1)
    }
    C[k] <- C0 + sum(ct[k, ])

    # Benefits - life expectancy
    for(t in 1 : N) {
      blt[k, t] <- inprod(pi[k, t, ], bl[]) / pow(1 + delta.b, t - 1)
    }
    BL[k] <- sum(blt[k, ])
  }
}

```

```

# Benefits - QALYs
for(t in 1 : N) {
  bqt[k, t] <- inprod(pi[k, t, ], bq[]) / pow(1 + delta.b, t - 1)
}
BQ[k] <- sum(bqt[k, ])

# Markov model probabilities:
#####

# Transition matrix
for(t in 2 : N) {
  Lambda[k, t, 1, 1] <- 1 - gamma[k, t] - lambda[k, t]
  Lambda[k, t, 1, 2] <- gamma[k, t] * lambda.op
  Lambda[k, t, 1, 3] <- gamma[k, t] *(1 - lambda.op)
  Lambda[k, t, 1, 4] <- 0
  Lambda[k, t, 1, 5] <- lambda[k, t]

  Lambda[k, t, 2, 1] <- 0
  Lambda[k, t, 2, 2] <- 0
  Lambda[k, t, 2, 3] <- 0
  Lambda[k, t, 2, 4] <- 0
  Lambda[k, t, 2, 5] <- 1

  Lambda[k, t, 3, 1] <- 0
  Lambda[k, t, 3, 2] <- 0
  Lambda[k,t,3,3] <- 0
  Lambda[k, t, 3, 4] <- 1 - lambda[k, t]
  Lambda[k, t, 3, 5] <- lambda[k, t]

  Lambda[k, t, 4, 1] <- 0
  Lambda[k, t, 4, 2] <- rho * lambda.op
  Lambda[k,t,4,3] <- rho * (1 - lambda.op)
  Lambda[k, t, 4, 4] <- 1 - rho - lambda[k, t]
  Lambda[k, t, 4, 5] <- lambda[k, t]

  Lambda[k, t, 5, 1] <- 0
  Lambda[k, t, 5, 2] <- 0
  Lambda[k, t, 5, 3] <- 0
  Lambda[k, t, 5, 4] <- 0
  Lambda[k, t, 5,5 ] <- 1

  gamma[k, t] <- h[k] * (t - 1)
}

# Marginal probability of being in each state at time 1
pi[k,1,1] <- 1 - lambda.op  pi[k,1, 2] <- 0   pi[k,1, 3] <- 0 ;
pi[k,1, 4] <- 0  pi[k,1, 5] <- lambda.op

# Marginal probability of being in each state at time t > 1
for(t in 2 : N) {
  for(s in 1 : S) {
    pi[k, t, s] <- inprod(pi[k, t - 1, ], Lambda[k, t, , s])
  }
}
}

```

```
# age-sex specific revision hazard
for(k in 1 : K) {
  logh[k] ~ dnorm(logh0[k], tau)
  h[k] <- exp(logh[k])
}

# Calculate mean and variance across strata at each iteration
# (Gives overall mean and variance using approach 1)

mean.C <- inprod(p.strata[], C[])
mean.BL <- inprod(p.strata[], BL[])
mean.BQ <- inprod(p.strata[], BQ[])

for(k in 1:12) {
  C.dev[k] <- pow(C[k]-mean.C , 2)
  BL.dev[k] <- pow(BL[k]-mean.BL , 2)
  BQ.dev[k] <- pow(BQ[k]-mean.BQ , 2)
}
var.C <- inprod(p.strata[], C.dev[])
var.BL <- inprod(p.strata[], BL.dev[])
var.BQ <- inprod(p.strata[], BQ.dev[])

}
```

[Data](#) (click to open)

[Results](#)

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
BL[1]	14.48	0.005165	4.414E-5	14.47	14.48	14.49	1	10000
BL[2]	12.7	0.003677	3.675E-5	12.7	12.7	12.71	1	10000
BL[3]	10.34	0.002138	2.157E-5	10.33	10.34	10.34	1	10000
BL[4]	7.737	7.832E-4	7.703E-6	7.735	7.737	7.738	1	10000
BL[5]	5.405	3.239E-4	2.926E-6	5.405	5.405	5.406	1	10000
BL[6]	4.101	2.136E-4	2.185E-6	4.101	4.101	4.102	1	10000
BL[7]	15.13	0.00516	5.429E-5	15.12	15.14	15.14	1	10000
BL[8]	13.69	0.003836	3.473E-5	13.69	13.7	13.7	1	10000
BL[9]	11.65	0.002426	2.845E-5	11.64	11.65	11.65	1	10000
BL[10]	9.1	9.528E-4	9.131E-6	9.098	9.1	9.101	1	10000
BL[11]	6.46	4.226E-4	3.928E-6	6.459	6.46	6.461	1	10000
BL[12]	4.988	2.917E-4	2.814E-6	4.988	4.988	4.989	1	10000
BQ[1]	13.17	0.06026	5.153E-4	13.05	13.17	13.28	1	10000
BQ[2]	11.59	0.05198	5.195E-4	11.48	11.59	11.68	1	10000
BQ[3]	9.469	0.03841	3.874E-4	9.386	9.472	9.537	1	10000
BQ[4]	7.157	0.01873	1.842E-4	7.116	7.158	7.189	1	10000
BQ[5]	5.019	0.009925	8.967E-5	4.997	5.02	5.036	1	10000
BQ[6]	3.812	0.006747	6.9E-5	3.797	3.813	3.824	1	10000
BQ[7]	13.82	0.05718	6.004E-4	13.7	13.82	13.92	1	10000
BQ[8]	12.53	0.05005	4.53E-4	12.43	12.53	12.62	1	10000
BQ[9]	10.69	0.03914	4.591E-4	10.61	10.7	10.76	1	10000
BQ[10]	8.43	0.0199	1.906E-4	8.386	8.431	8.464	1	10000
BQ[11]	6.004	0.01083	1.007E-4	5.98	6.004	6.022	1	10000
BQ[12]	4.64	0.007624	7.354E-5	4.623	4.641	4.653	1	10000
C[1]	5787.0	230.8	1.974	5357.0	5781.0	6259.0	1	10000
C[2]	5425.0	202.1	2.02	5058.0	5414.0	5850.0	1	10000
C[3]	4997.0	151.6	1.529	4730.0	4984.0	5324.0	1	10000
C[4]	4472.0	74.94	0.7371	4342.0	4466.0	4634.0	1	10000
C[5]	4266.0	40.07	0.3621	4198.0	4263.0	4355.0	1	10000
C[6]	4196.0	27.27	0.2788	4149.0	4194.0	4257.0	1	10000
C[7]	5636.0	218.0	2.289	5230.0	5631.0	6079.0	1	10000
C[8]	5359.0	193.5	1.751	5010.0	5348.0	5768.0	1	10000
C[9]	5010.0	153.5	1.801	4734.0	5002.0	5332.0	1	10000
C[10]	4493.0	79.15	0.7582	4356.0	4487.0	4666.0	1	10000
C[11]	4285.0	43.46	0.4039	4210.0	4282.0	4380.0	1	10000
C[12]	4215.0	30.61	0.2953	4163.0	4212.0	4283.0	1	10000
mean.BL	8.687	4.508E-4	4.463E-6	8.686	8.687	8.688	1	10000
mean.BQ	8.015	0.008051	7.912E-5	7.998	8.015	8.03	1	10000
mean.C	4609.0	31.82	0.3126	4550.0	4608.0	4674.0	1	10000
var.BL	6.714	0.002979	2.98E-5	6.708	6.714	6.72	1	10000
var.BQ	5.466	0.03819	3.761E-4	5.389	5.467	5.539	1	10000
var.C	174400.0	28440.0	286.9	124400.0	172500.0	235400.0	1	10000

Note: results for the bottom panel of Table 3 (approach 1) for costs are given by

m = posterior mean of mean.C = 4609

vP1 = posterior variance of mean.C = 31.82 * 31.82

vH1 = posterior mean of var.C = 174400

'Model' to calculate overall mean (m), var due to uncertainty (vP2) and var due to heterogeneity (vH2) using approach 2

No updates needed - just compile model, load data, and gen inits, then use node tool from info menu to obtain values of mC, mBL, mBQ, vP2.C, vP2.BL, vP2.BQ, vH2.C, vH2.BL, vH2.BQ, TC, TBL, TBQ, pcC, pcBL, pcBQ.

```

model {

# overall mean outcome (m)
mC <- inprod(p.strata[], C[])
mBL <- inprod(p.strata[], BL[])
mBQ <- inprod(p.strata[], BQ[])

# variance due to uncertainty, vP
for(k in 1:12) {
  VC[k] <- sdC[k]*sdC[k]
  VBL[k] <- sdBL[k]*sdBL[k]
  VBQ[k] <- sdBQ[k]*sdBQ[k]
}
vP2.C <- inprod(p.strata[], VC[])
vP2.BL <- inprod(p.strata[], VBL[])
vP2.BQ <- inprod(p.strata[], VBQ[])

# variance due to heterogeneity, vH
for(k in 1:12) { devC[k] <- pow(C[k] - mC, 2) }
vH2.C <- inprod(p.strata[], devC[])
for(k in 1:12) { devBL[k] <- pow(BL[k] - mBL, 2) }
vH2.BL <- inprod(p.strata[], devBL[])
for(k in 1:12) { devBQ[k] <- pow(BQ[k] - mBQ, 2) }
vH2.BQ <- inprod(p.strata[], devBQ[])

# Percent of total variance due to heterogeneity
TC <- vP2.C + vH2.C
pcC <- vH2.C/TC
TBL <- vP2.BL + vH2.BL
pcBL <- vH2.BL/TBL
TBQ <- vP2.BQ + vH2.BQ
pcBQ <- vH2.BQ/TBQ

}

```

[Data](#) (click to open, posterior means and posterior sd of C, BL and BQ from running model 3)

Results

```

mC    4609.03
mBL   8.687390000000001
mBQ   8.01488
vP2.C 11472.793425
vP2.BL 3.3068250474E-6
vP2.BQ 7.493649773900001E-4
vH2.C 163953.4291
vH2.BL 6.713258897899999
vH2.BQ 5.464389485600001
TC    175426.222525
TBL   6.713262204725046
TBQ   5.465138850577391
pcC   0.9346004647431485

```

Hips3

Examples Volume III

pcBL 0.999999507419054
pcBQ 0.9998628827193821



Hips model 4: Comparative analysis of Stanmore & Charnley incorporating evidence

Spiegelhalter, D.J. and Best, N.G. “Bayesian approaches to multiple sources of evidence and uncertainty in complex cost-effectiveness modelling”. *Statistics in Medicine* 22, (2003), 3687-3709.

n = 10000 updates (1 per simulated set of parameter values) are required for this model;
 For hazard ratio estimates in bottom of table 4, monitor HR. For results in table 5, monitor C.incr, BQ.incr, ICER.strata, mean.C.incr, mean.BQ.incr, mean.ICER, P.CEA.strata[30,], P.CEA.strata[50,], P.CEA[30] and P.CEA[50]. To produce plots in Fig 2, use coda option to save samples of C.incr, BQ.incr, mean.C.incr, mean.BQ.incr. To produce plots in Fig 3, set summary monitors on P.CEA.strata and P.CEA to get posterior means

Sections of the code that have changed from Model 1 are shown in bold

```

model {

# Evidence
#####

  for (i in 1 : M){ # loop over studies
    rC[i] ~ dbin(pC[i], nC[i]) # number of revisions on Charnley
    rS[i] ~ dbin(pS[i], nS[i]) # number of revisions on Stanmore
    cloglog(pC[i]) <- base[i] - logHR[i]/2
    cloglog(pS[i]) <- base[i] + logHR[i]/2
    base[i] ~ dunif(-100,100)
    # log hazard ratio for ith study
    logHR[i] ~ dnorm(LHR,tauHR[i])
    tauHR[i] <- qualweights[i] * tauh # precision for ith study weighted by quality weights
  }
  LHR ~ dunif(-100,100)
  log(HR) <- LHR
  tauh <- 1 / (sigmah * sigmah)
  sigmah ~ dnorm( 0.2, 400)|(0, ) # between-trial sd = 0.05 (prior constrained to be positive)

  for(k in 1 : K) {
    logh[k] ~ dnorm(logh0[k], tau)
    h[1, k] <- exp(logh[k]) # revision hazard for Charnley
    h[2, k] <- HR * h[1, k] # revision hazard for Stanmore
  }

  # Cost-effectiveness model
#####

  for(k in 1 : K) { # loop over strata

    for(n in 1 : 2) { # loop over protheses

```

```

# Cost and benefit equations in closed form:
#####

# Costs
for(t in 1 : N) {
  ct[n, k, t] <- inprod(pi[n, k, t, ], c[n, ]) / pow(1 + delta.c, t - 1)
}
C[n,k] <- C0[n] + sum(ct[n, k, ])

# Benefits - life expectancy
for(t in 1 : N) {
  blt[n, k, t] <- inprod(pi[n, k, t, ], bl[]) / pow(1 + delta.b, t - 1)
}
BL[n, k] <- sum(blt[n, k, ])

# Benefits - QALYs
for(t in 1 : N) {
  bqt[n, k, t] <- inprod(pi[n, k, t, ], bq[]) / pow(1 + delta.b, t - 1)
}
BQ[n, k] <- sum(bqt[n, k, ])

# Markov model probabilities:
#####

# Transition matrix
for(t in 2:N) {
  Lambda[n, k, t, 1, 1] <- 1 - gamma[n, k, t] - lambda[k, t]
  Lambda[n, k, t, 1, 2] <- gamma[n, k, t] * lambda.op
  Lambda[n, k, t, 1, 3] <- gamma[n, k, t] *(1 - lambda.op)
  Lambda[n, k, t, 1, 4] <- 0
  Lambda[n, k, t, 1, 5] <- lambda[k, t]

  Lambda[n, k, t, 2, 1] <- 0
  Lambda[n, k, t, 2, 2] <- 0
  Lambda[n, k, t, 2, 3] <- 0
  Lambda[n, k, t, 2, 4] <- 0
  Lambda[n, k, t, 2, 5] <- 1

  Lambda[n, k, t, 3, 1] <- 0
  Lambda[n, k, t, 3, 2] <- 0
  Lambda[n, k, t, 3, 3] <- 0
  Lambda[n, k, t, 3, 4] <- 1 - lambda[k, t]
  Lambda[n, k, t, 3, 5] <- lambda[k, t]

  Lambda[n, k, t, 4, 1] <- 0
  Lambda[n, k, t, 4, 2] <- rho * lambda.op
  Lambda[n, k, t, 4, 3] <- rho * (1 - lambda.op)
  Lambda[n, k, t, 4, 4] <- 1 - rho - lambda[k, t]
  Lambda[n, k, t, 4, 5] <- lambda[k, t]

  Lambda[n, k, t, 5, 1] <- 0
  Lambda[n, k, t, 5, 2] <- 0
  Lambda[n, k, t, 5, 3] <- 0
  Lambda[n, k, t, 5, 4] <- 0
  Lambda[n, k, t, 5, 5] <- 1

```

```

    gamma[n, k, t] <- h[n, k] * (t - 1)
  }

  # Marginal probability of being in each state at time 1
  pi[n, k, 1, 1] <- 1 - lambda.op  pi[n, k, 1, 2] <- 0    pi[n, k, 1, 3] <- 0
  pi[n, k, 1, 4] <- 0  pi[n, k, 1, 5] <- lambda.op

  # Marginal probability of being in each state at time t>1
  for(t in 2 : N) {
    for(s in 1 : S) {
      pi[n, k, t, s] <- inprod(pi[n, k, t - 1, ], Lambda[n, k, t, , s])
    }
  }
}

# Incremental costs and benefits
#####

for(k in 1 : K) {
  C.incr[k] <- C[2, k] - C[1, k]
  BQ.incr[k] <- BQ[2, k] - BQ[1, k]
  ICER.strata[k] <- C.incr[k] / BQ.incr[k]
}

# Probability of cost effectiveness @ KK pounds per QALY
# (values of KK considered range from 200 to 20000 in 200 pound increments)
for(m in 1 : 100) {
  for(k in 1 : 12) {
    P.CEA.strata[m,k] <- step(KK[m] * BQ.incr[k] - C.incr[k])
  }
  P.CEA[m] <- step(KK[m] * mean.BQ.incr - mean.C.incr)
}

# overall incremental costs and benefit
for(n in 1 : 2) {
  mean.C[n] <- inprod(p.strata[], C[n, ])
  mean.BQ[n] <- inprod(p.strata[], BQ[n, ])
}
mean.C.incr <- mean.C[2] - mean.C[1]
mean.BQ.incr <- mean.BQ[2] - mean.BQ[1]
mean.ICER <- mean.C.incr / mean.BQ.incr
}

```

[Data](#) (click to open)

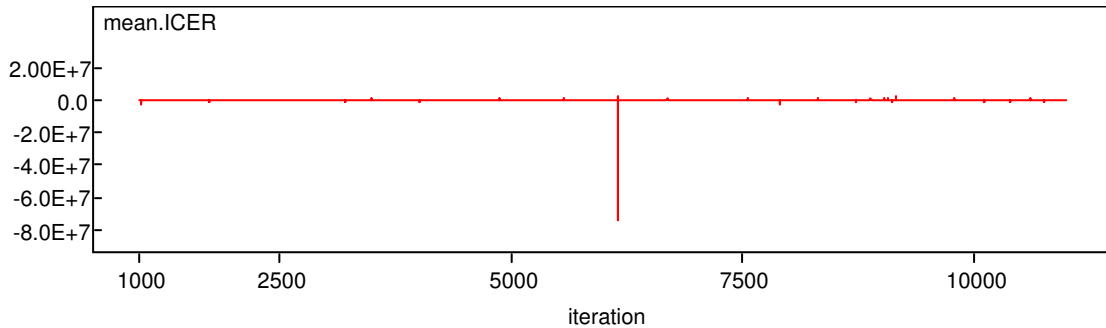
[Inits](#) (click to open)

[Results](#)

(quality weights c(0.5, 1, 0.2), delta.c = 0.06, delta.b = 0.06)

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
BQ.incr[1]	0.1344	0.062	0.001643	-0.001131	0.1383	0.2468	1001	10000
BQ.incr[2]	0.1117	0.0516	0.001338	-8.635E-4	0.115	0.2065	1001	10000
BQ.incr[3]	0.08049	0.03721	9.46E-4	-6.482E-4	0.08215	0.1488	1001	10000
BQ.incr[4]	0.03768	0.01761	4.254E-4	-3.335E-4	0.03852	0.07062	1001	10000
BQ.incr[5]	0.01952	0.009157	2.232E-4	-1.84E-4	0.01983	0.03648	1001	10000
BQ.incr[6]	0.01313	0.006165	1.5E-4	-1.255E-4	0.01333	0.02473	1001	10000
BQ.incr[7]	0.1263	0.05841	0.001528	-9.574E-4	0.1296	0.2334	1001	10000
BQ.incr[8]	0.1083	0.05013	0.001284	-8.994E-4	0.111	0.2003	1001	10000
BQ.incr[9]	0.0823	0.03811	9.646E-4	-7.132E-4	0.08406	0.1519	1001	10000
BQ.incr[10]	0.04001	0.01867	4.63E-4	-3.489E-4	0.04077	0.07491	1001	10000
BQ.incr[11]	0.02133	0.01	2.519E-4	-1.695E-4	0.02174	0.04026	1001	10000
BQ.incr[12]	0.01486	0.006956	1.715E-4	-1.24E-4	0.01515	0.02794	1001	10000
C.incr[1]	-85.33	251.9	6.739	-534.7	-102.8	470.3	1001	10000
C.incr[2]	-23.12	211.4	5.544	-401.6	-39.17	444.6	1001	10000
C.incr[3]	73.48	154.1	3.979	-202.7	64.23	411.4	1001	10000
C.incr[4]	216.9	73.45	1.815	83.44	212.5	378.4	1001	10000
C.incr[5]	280.2	38.55	0.956	210.2	278.1	365.2	1001	10000
C.incr[6]	303.0	25.94	0.6436	255.5	301.7	359.8	1001	10000
C.incr[7]	-59.73	235.1	6.211	-482.5	-74.64	460.2	1001	10000
C.incr[8]	-10.73	203.7	5.291	-378.4	-24.44	436.6	1001	10000
C.incr[9]	68.89	156.9	4.031	-212.5	59.23	415.5	1001	10000
C.incr[10]	209.6	77.43	1.954	67.46	205.2	379.5	1001	10000
C.incr[11]	274.2	41.76	1.068	196.7	271.8	366.3	1001	10000
C.incr[12]	297.2	29.11	0.7313	244.1	295.7	361.2	1001	10000
HR	0.6174	0.1616	0.004232	0.3696	0.595	1.004	1001	10000
ICER.strata[1]	-3892.0	364400.0	3529.0	-8165.0	-827.9	13570.0	1001	10000
ICER.strata[2]	-3767.0	399100.0	3867.0	-8614.0	-426.6	15460.0	1001	10000
ICER.strata[3]	-4304.0	575500.0	5596.0	-9641.0	647.9	20490.0	1001	10000
ICER.strata[4]	-1590.0	8.49E+5	8168.0	-14140.0	5267.0	43720.0	1001	10000
ICER.strata[5]	4637.0	1.164E+6	11050.0	-19170.0	13540.0	83620.0	1001	10000
ICER.strata[6]	3026.0	2.321E+6	22540.0	-27490.0	21940.0	124500.0	1001	10000
ICER.strata[7]	-4078.0	400100.0	3886.0	-8385.0	-665.1	14200.0	1001	10000
ICER.strata[8]	-3635.0	403100.0	3902.0	-8794.0	-312.2	16080.0	1001	10000
ICER.strata[9]	-3124.0	447300.0	4300.0	-9508.0	571.2	19410.0	1001	10000
ICER.strata[10]	-1372.0	7.46E+5	7208.0	-13610.0	4774.0	40700.0	1001	10000
ICER.strata[11]	-4258.0	1.891E+6	18190.0	-21410.0	12050.0	77440.0	1001	10000
ICER.strata[12]	-1725.0	2.396E+6	23380.0	-28200.0	18940.0	109600.0	1001	10000

P.CEA[30]	0.7233	0.4474	0.00952	0.0	1.0	1.0	1001	10000
P.CEA[50]	0.8481	0.3589	0.00717	0.0	1.0	1.0	1001	10000
P.CEA.strata[30,1]	0.9187	0.2733	0.005163	0.0	1.0	1.0	1001	10000
P.CEA.strata[30,2]	0.907	0.2904	0.005463	0.0	1.0	1.0	1001	10000
P.CEA.strata[30,3]	0.8702	0.3361	0.006467	0.0	1.0	1.0	1001	10000
P.CEA.strata[30,4]	0.5453	0.4979	0.01048	0.0	1.0	1.0	1001	10000
P.CEA.strata[30,5]	0.0303	0.1714	0.002634	0.0	0.0	1.0	1001	10000
P.CEA.strata[30,6]	4.0E-4	0.02	1.969E-4	0.0	0.0	0.0	1001	10000
P.CEA.strata[30,7]	0.9146	0.2795	0.005304	0.0	1.0	1.0	1001	10000
P.CEA.strata[30,8]	0.9046	0.2938	0.005463	0.0	1.0	1.0	1001	10000
P.CEA.strata[30,9]	0.8741	0.3317	0.006428	0.0	1.0	1.0	1001	10000
P.CEA.strata[30,10]	0.5896	0.4919	0.01106	0.0	1.0	1.0	1001	10000
P.CEA.strata[30,11]	0.0634	0.2437	0.004157	0.0	0.0	1.0	1001	10000
P.CEA.strata[30,12]	0.0019	0.04355	4.426E-4	0.0	0.0	0.0	1001	10000
P.CEA.strata[50,1]	0.9396	0.2382	0.004038	0.0	1.0	1.0	1001	10000
P.CEA.strata[50,2]	0.9342	0.2479	0.004377	0.0	1.0	1.0	1001	10000
P.CEA.strata[50,3]	0.9166	0.2765	0.005168	0.0	1.0	1.0	1001	10000
P.CEA.strata[50,4]	0.7654	0.4237	0.008298	0.0	1.0	1.0	1001	10000
P.CEA.strata[50,5]	0.2492	0.4325	0.008491	0.0	0.0	1.0	1001	10000
P.CEA.strata[50,6]	0.0204	0.1414	0.001758	0.0	0.0	0.0	1001	10000
P.CEA.strata[50,7]	0.9385	0.2402	0.004157	0.0	1.0	1.0	1001	10000
P.CEA.strata[50,8]	0.9322	0.2514	0.004412	0.0	1.0	1.0	1001	10000
P.CEA.strata[50,9]	0.9192	0.2725	0.005053	0.0	1.0	1.0	1001	10000
P.CEA.strata[50,10]	0.7887	0.4082	0.008194	0.0	1.0	1.0	1001	10000
P.CEA.strata[50,11]	0.332	0.4709	0.009844	0.0	0.0	1.0	1001	10000
P.CEA.strata[50,12]	0.053	0.224	0.003122	0.0	0.0	1.0	1001	10000
mean.BQ.incr	0.04811	0.02112	5.596E-4	-3.994E-4	0.05033	0.08318	1001	10000
mean.C.incr	184.6	88.62	2.351	38.94	175.1	388.7	1001	10000
mean.ICER	-2928.0	745700.0	7200.0	-12530.0	3271.0	33800.0	1001	10000





Jama River Valley Ecuador - Radiocarbon calibration with phase information

This example is from the book Buck CE, Cavanagh WG, & Litton CD (1996) *Bayesian approach to interpreting archaeological data*. Wiley: Chichester. p226-232 See also Zeidler, JA, Buck CE & Litton CD (1998) The integration of archaeological phase information and radiocarbon results from the Jama River Valley, Ecuador: a Bayesian approach. *Latin American Antiquity* **9** 160-179 .The model was set up by Andrew Millard.

© Andrew Millard 2001

```

model{
  for (i in 1 : nDate){
    theta[i] ~ dunif(beta[phase[i]], alpha[phase[i]] )
    X[i] ~ dnorm(mu[i], tau[i])
    tau[i] <- 1 / pow(sigma[i], 2)
    mu[i] <- interp.lin(theta[i], calBP[], C14BP[])
  }
# priors on phase ordering
alpha[1] ~ dunif(beta[1], theta.max)
beta[1] ~ dunif(alpha[2], alpha[1])
alpha[2] ~ dunif(beta[2], beta[1])
beta[2] ~ dunif(alpha[3], alpha[2])
alpha[3] ~ dunif(beta[3], beta[2])
beta[3] ~ dunif(alpha[4], alpha[3])
alpha[4] ~ dunif(alpha4min, beta[3])
alpha4min <- max(beta[4], alpha[5])
beta[4] ~ dunif(beta[5], alpha[4])
alpha[5] ~ dunif(alpha5min, alpha[4])
alpha5min <- max(beta[5], alpha[6])
beta[5] ~ dunif(beta[6], beta5max)
beta5max <- min(beta[4], alpha[5])
alpha[6] ~ dunif(beta[6], alpha[5])
beta[6] ~ dunif(beta[7], beta6max)
beta6max <- min(alpha[6], beta[5])
alpha[7] <- beta[6]
beta[7] ~ dunif(theta.min,alpha[7])

for (i in 1 : 7) {
  alpha.desc[i] <- 10 * round(alpha[i] / 10)
  beta.desc[i] <- 10 * round(beta[i] / 10)
}

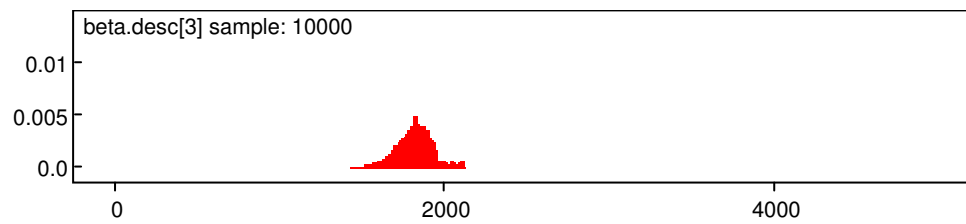
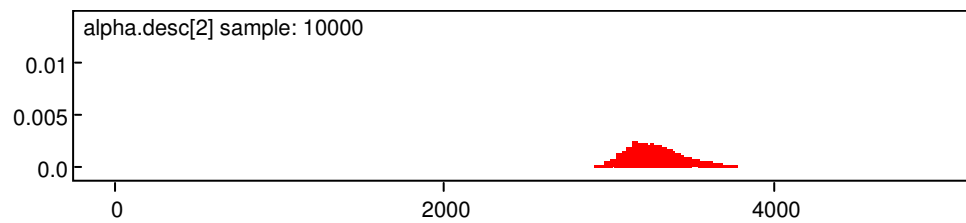
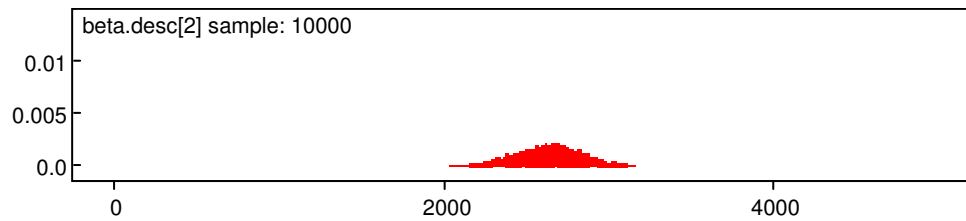
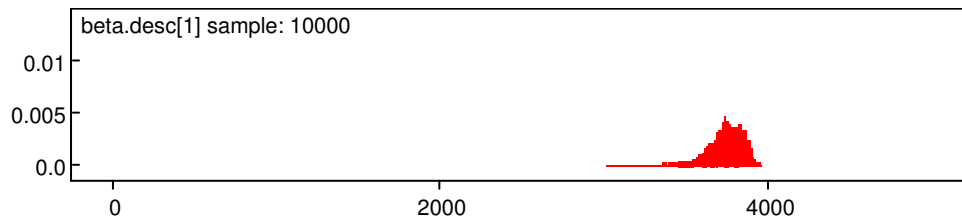
```

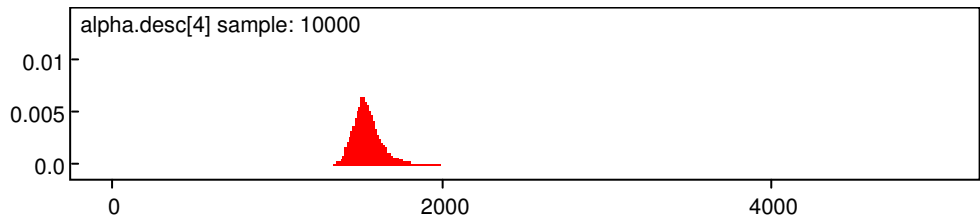
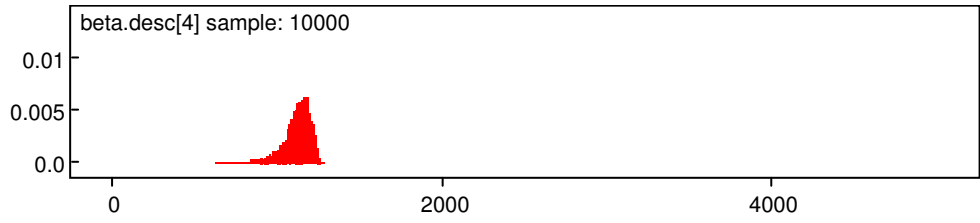
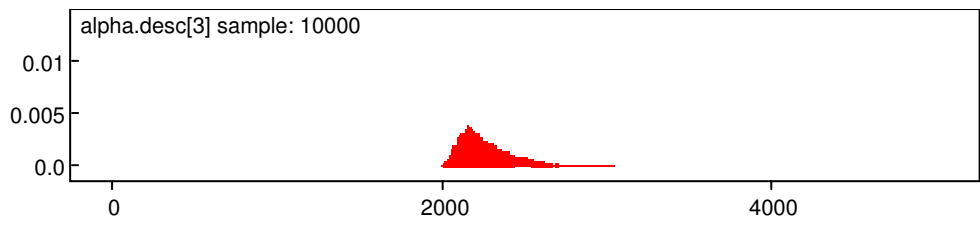
```
}  
}
```

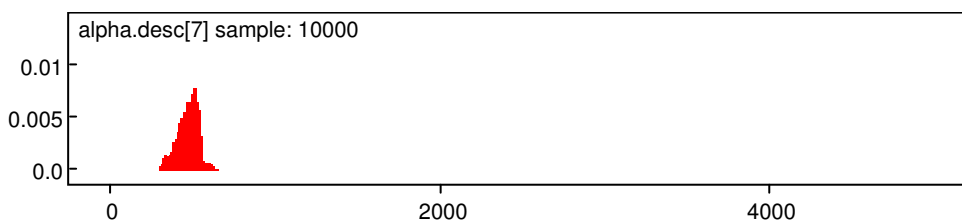
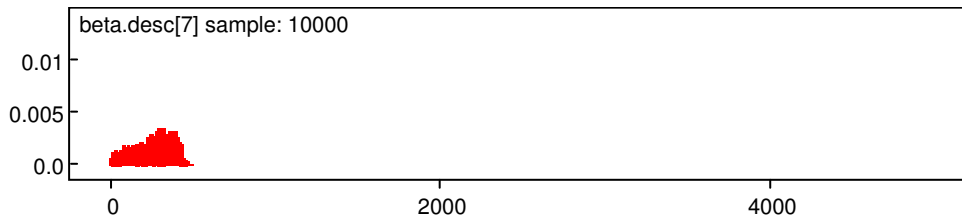
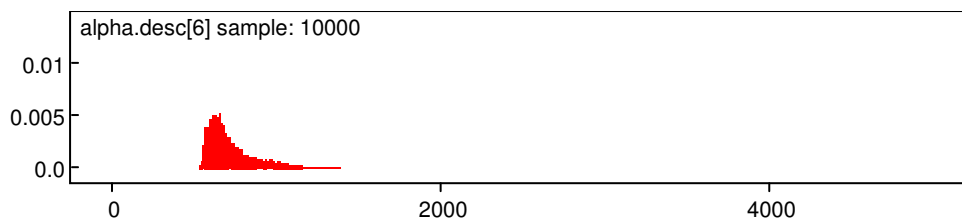
[Data1](#) [Data2](#) [Radio Carbon Calibration Curve](#) (click to open)

[Inits](#) (click to open)

Results









Pigs: genetic counseling and pedigree analysis

Spiegelhalter (1990) uses exact methods to analyse a small pedigree. This pedigree was previously used by Cannings and Thompson (1981) to illustrate their 'peeling' procedure to provide likelihoods for gene frequencies and probabilities for individuals being affected or carriers. We assume the pedigree refers to pigs which have the possibility of carrying a recessive gene: thus each pig has a genotype a_1a_1 , a_1a_2 or a_2a_2 , in which only those with a_2a_2 are affected with the trait, while those with a_1a_2 are carriers of the defective allele a_2 . We assume that Ian (the consequence of a mating between Fred and his Aunt Clare) is yet to be born, and all that is known is that Fred's niece Jane has the trait. We wish to estimate the prevalence p of the allele a_2 , and predict the chance of Ian being affected. The conditional probability distributions are as follows. For the genotype of the founder nodes Ann, Brian, Eric and Henry we assume a binomial distribution

$$\text{Founder} \sim \text{Binomial}(q, 2)$$

where Founder takes values 0, 1 or 2 for genotypes a_2a_2 , a_1a_2 and a_1a_1 respectively, and $q = 1 - p$ is the prevalence of the allele a_1 . This is equivalent to assuming Hardy Weinberg equilibrium, giving $P(a_1a_1) = q^2$, $P(a_1a_2) = 2q(1 - q)$, $P(a_2a_2) = (1 - q)^2$.

For the genotype of offspring we have the standard Mendelian inheritance probabilities given by the following table.

Genotype of parents	Genotype of offspring		
	a_1a_1	a_1a_2	a_2a_2
a_1a_1 a_1a_1	1.0	0.0	0.0
a_1a_1 a_1a_2	0.5	0.5	0.0
a_1a_1 a_2a_2	0.0	1.0	0.0
a_1a_2 a_1a_2	0.25	0.5	0.25
a_1a_2 a_2a_2	0.0	0.5	0.5
a_2a_2 a_2a_2	0.0	0.0	1.0

For a recessive gene the genotype-to-phenotype penetrance probabilities are given by:

Genotype of individual	Phenotype of individual	
	1 (normal)	2 (affected)
a_1a_1	1	
a_1a_2	1	
a_2a_2		1

The necessary inheritance probabilities are read in from the data file as an array

BUGS code for Pigs model:

```

model
{
  q ~ dunif(0,1)                # prevalence of a1
  p <- 1 - q                    # prevalence of a2
  Ann1 ~ dbin(q,2); Ann <- Ann1 + 1 # geno. dist. for founder
  Brian1 ~ dbin(q,2); Brian <- Brian1 + 1
  Clare ~ dcat(p.mendelian[Ann,Brian,]) # geno. dist. for child
  Diane ~ dcat(p.mendelian[Ann,Brian,])
  Eric1 ~ dbin(q,2)
  Eric <- Eric1 + 1
  Fred ~ dcat(p.mendelian[Diane,Eric,])
  Gene ~ dcat(p.mendelian[Diane,Eric,])
  Henry1 ~ dbin(q,2)
  Henry <- Henry1 + 1
  Ian ~ dcat(p.mendelian[Clare,Fred,])
  Jane ~ dcat(p.mendelian[Gene,Henry,])
  A1 ~ dcat(p.recessive[Ann,])      # phenotype distribution
  B1 ~ dcat(p.recessive[Brian,])
  C1 ~ dcat(p.recessive[Clare,])
  D1 ~ dcat(p.recessive[Diane,])
  E1 ~ dcat(p.recessive[Eric,])
  F1 ~ dcat(p.recessive[Fred,])
  G1 ~ dcat(p.recessive[Gene,])
  H1 ~ dcat(p.recessive[Henry,])
  I1 ~ dcat(p.recessive[Ian,])
  J1 ~ dcat(p.recessive[Jane,])
  a <- equals(Ann, 2)              # event that Ann is carrier
  b <- equals(Brian, 2)
  c <- equals(Clare, 2)
  d <- equals(Diane, 2)
  e <- equals(Eric, 2) ;
  f <- equals(Fred, 2)
  g <- equals(Gene, 2)
  h <- equals(Henry, 2)
  for (J in 1:3) {
    i[J] <- equals(Ian, J)        # i[1] = a1 a1
  }
}

```

```

# i[2] = a1 a2
# i[3] = a2 a2 (i.e. Ian affected)
}
}

```

[Data](#) (click to open)

[Inits](#) (click to open)

Results

Time for 100000 updates 18s on 200MHz Pentium Pro. A 10000 update burn in followed by a further 100000 updates gave the parameter estimates

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
i[1]	0.5727	0.4947	0.001848	0.0	1.0	1.0	10001	100000
i[2]	0.3669	0.482	0.001654	0.0	0.0	1.0	10001	100000
i[3]	0.06034	0.2381	7.803E-4	0.0	0.0	1.0	10001	100000
p	0.6266	0.1606	6.096E-4	0.2925	0.6377	0.9006	10001	100000

We note a number of important tricks. First, each genotype is a 3-valued categorical variable with conditional probabilities either determined by the binomial (Hardy-Weinberg equilibrium) distribution (for founder nodes) or from the Mendelian inheritance probabilities which are stored as a 3-dimensional array `p.mendelian`. In the latter case, the genotype of the parents picks which row of the matrix which row of the matrix is used for the distribution. However, the rows of this matrix are indexed by values 1, 2 or 3, whilst the genotypes of the founder nodes take values 0, 1 or 2. Since BUGS does not allow subscripts to be functions of variables, we must first add 1 to the genotype of the parents (for example, `Ann = Ann1 + 1`) and use these new variables as subscripts to the matrix `p.mendelian`. The genotype-to-phenotype distribution is handled similarly in a matrix `p.recessive`. Second, the `equals` function `equals(Ann, 2)` allows the calculation of $P(\text{Ann's genotype} = 2)$ (i.e. a carrier), whilst `equals(Ian, J)` calculates $P(\text{Ian's genotype} = J)$, where $J=3$ implies that Ian is affected.



Bayes factors using pseudo priors

Bayes factors using the Carlin and Chib method. For full description see Page 47 of Classic BUGS examples Vol 2.

```

model{
# standardise data
  for(i in 1:N){
    Ys[i] <- (Y[i] - mean(Y[])) / sd(Y[])
    xs[i] <- (x[i] - mean(x[])) / sd(x[])
    zs[i] <- (z[i] - mean(z[])) / sd(z[])
  }

# model node
  j ~ dcat(p[])
  p[1] <- 0.9995 p[2] <- 1 - p[1] # use for joint modelling
  # p[1] <- 1 p[2] <- 0 # include for estimating Model 1
  # p[1] <- 0 p[2] <- 1 # include for estimating Model 2
  pM2 <- step(j - 1.5)

# model structure
  for(i in 1 : N){
    mu[1, i] <- alpha + beta * xs[i]
    mu[2, i] <- gamma + delta * zs[i]
    Ys[i] ~ dnorm(mu[j, i], tau[j])
  }

# Model 1
  alpha ~ dnorm(mu.alpha[j], tau.alpha[j])
  beta ~ dnorm(mu.beta[j], tau.beta[j])
  tau[1] ~ dgamma(r1[j], l1[j])
  # estimation priors
  mu.alpha[1] <- 0 tau.alpha[1] <- 1.0E-6
  mu.beta[1] <- 0 tau.beta[1] <- 1.0E-4
  r1[1] <- 0.0001 l1[1] <- 0.0001
  # pseudo-priors
  mu.alpha[2] <- 0 tau.alpha[2] <- 256
  mu.beta[2] <- 1 tau.beta[2] <- 256
  r1[2] <- 30 l1[2] <- 4.5

# Model 2

```

```

gamma ~ dnorm(mu.gamma[j], tau.gamma[j])
delta ~ dnorm(mu.delta[j], tau.delta[j])
tau[2] ~ dgamma(r2[j], l2[j])
# pseudo-priors
mu.gamma[1] <- 0 tau.gamma[1] <- 400
mu.delta[1] <- 1 tau.delta[1] <- 400
r2[1] <- 46 l2[1] <- 4.5
# estimation priors
mu.gamma[2] <- 0 tau.gamma[2] <- 1.0E-6
mu.delta[2] <- 0 tau.delta[2] <- 1.0E-4
r2[2] <- 0.0001 l2[2] <- 0.0001
}

```

[Data](#) (click to open)

[Inits](#) (click to open)

Results

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
pM2	0.623	0.4846	0.007682	0.0	1.0	1.0	1001	10000

corresponding to a Bayes factor of $0.623 / 0.377 \times 0.9995 / 0.0005 = 3308$.



St Veit-Klinglberg, Austria - Radiocarbon calibration with stratification

This example is from the book Buck CE, Cavanagh WG & Litton CD (1996) *Bayesian approach to interpreting archaeological data*. Wiley: Chichester p218-226

See also Buck CE, Litton CD & Shennan SJ (1994) A case study in combining radiocarbon and archaeological information: the Early Bronze Age settlement of St Veit-Klinglberg, Lan Salzburg, Austria. *Germania* 2 427-447. The model was set up by Andrew Millard.

© Andrew Millard 2001

```

model{
  theta[1] ~ dunif(theta[2], theta.max)
  theta[2] ~ dunif(theta[3], theta[1])
  theta[3] ~ dunif(theta[9], theta[2])
  theta[4] ~ dunif(theta[9], theta.max)
  theta[5] ~ dunif(theta[7], theta.max)
  theta[6] ~ dunif(theta[7], theta.max)
  theta[7] ~ dunif(theta[9], theta7max)
  theta7max <- min(theta[5], theta[6])
  theta[8] ~ dunif(theta[9], theta.max)
  theta[9] ~ dunif(theta[10], theta9max)
  theta9max <- min(min(theta[3], theta[4]), min(theta[7], theta[8]))
  theta[10] ~ dunif(theta[11], theta[9])
  theta[11] ~ dunif(0, theta[10])

  bound[1] <- ranked(theta[1:8], 8)
  bound[2] <- ranked(theta[1:8], 1)
  bound[3] <- ranked(theta[9:11], 3)
  bound[4] <- ranked(theta[9:11], 1)

  for (j in 1 : 5){
    theta[j + 11] ~ dunif(0, theta.max)
    within[j, 1] <- 1 - step(bound[1] - theta[j + 11])
    for (k in 2 : 4){
      within[j, k] <- step(bound[k - 1] - theta[j + 11])
        - step(bound[k] - theta[j + 11])
    }
    within[j, 5] <- step(bound[4] - theta[j + 11])
  }
}

```

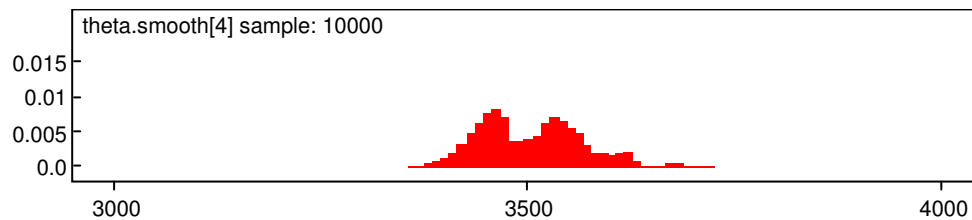
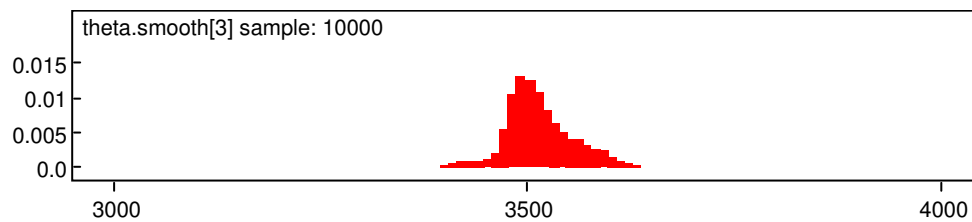
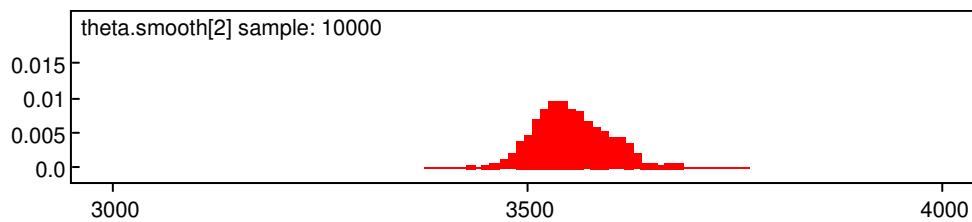
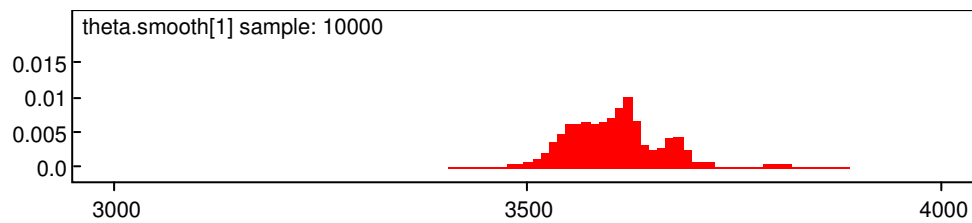
```
for (i in 1:nDate){
  X[i] ~ dnorm(mu[i], tau[i])
  tau[i] <- 1/pow(sigma[i],2)
  mu[i] <- interp.lin(theta[i], calBP[], C14BP[])

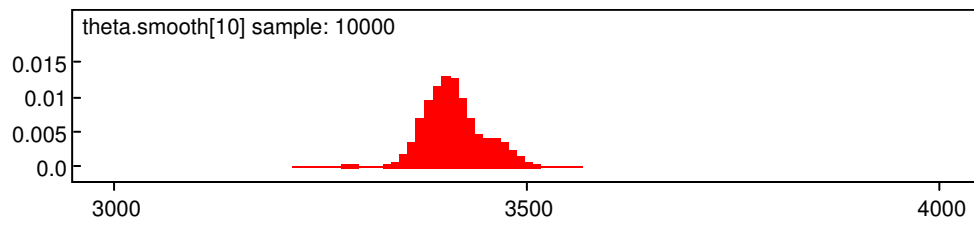
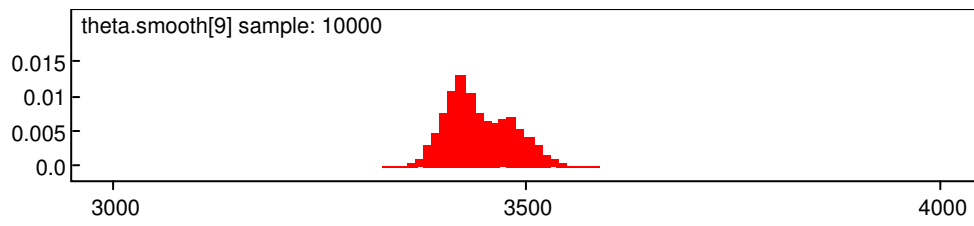
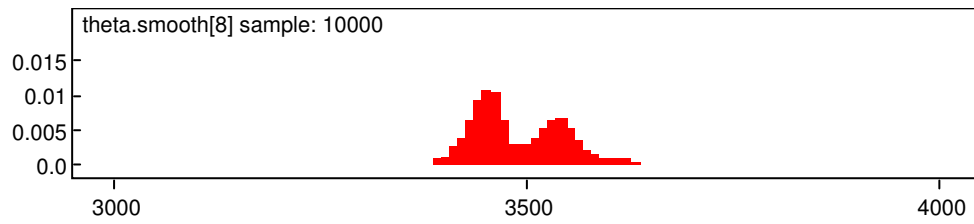
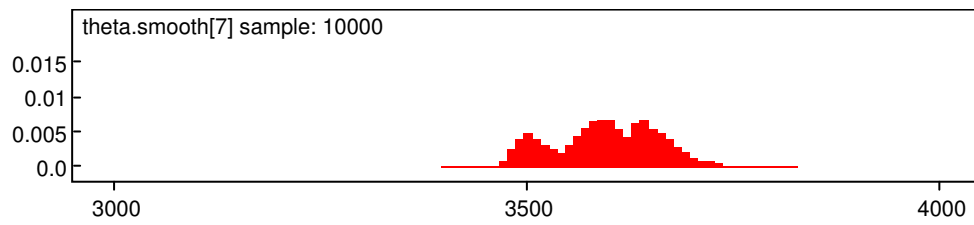
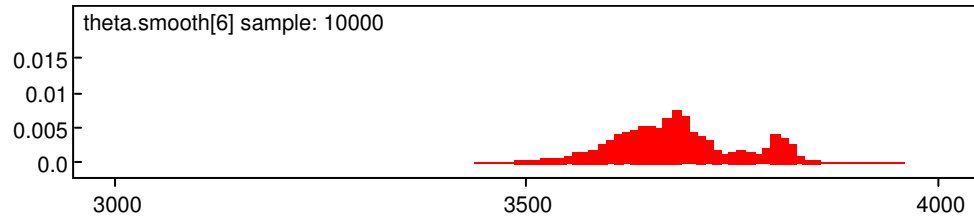
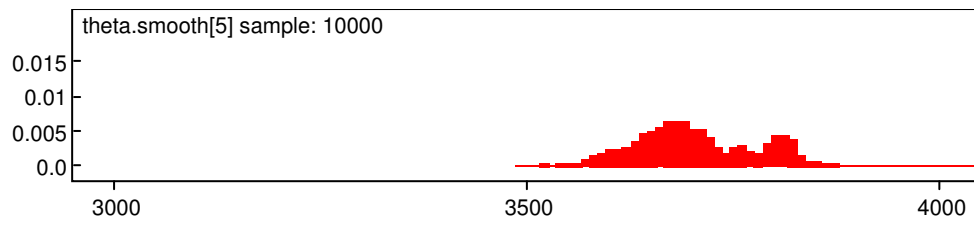
  # monitor the following variable to smooth density of theta
  theta.smooth[i] <- 10 * round(theta[i] / 10)
}
}
```

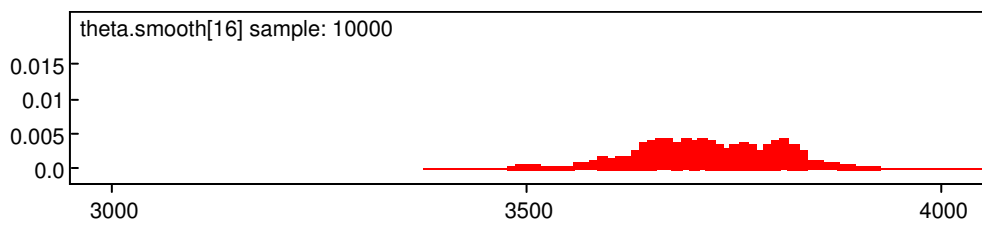
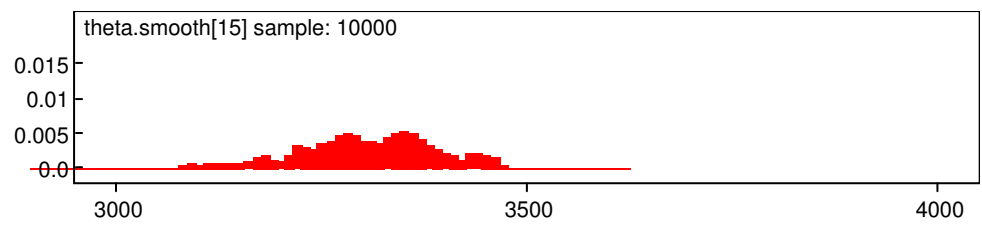
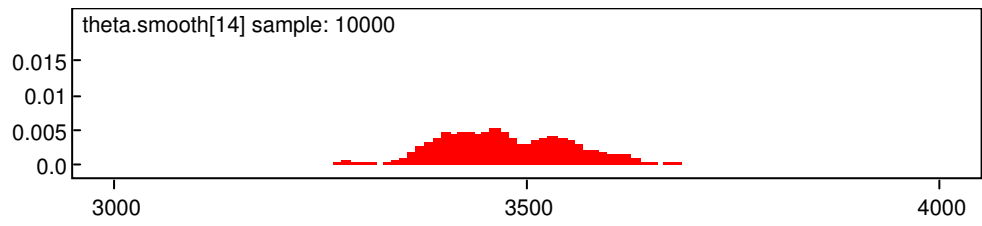
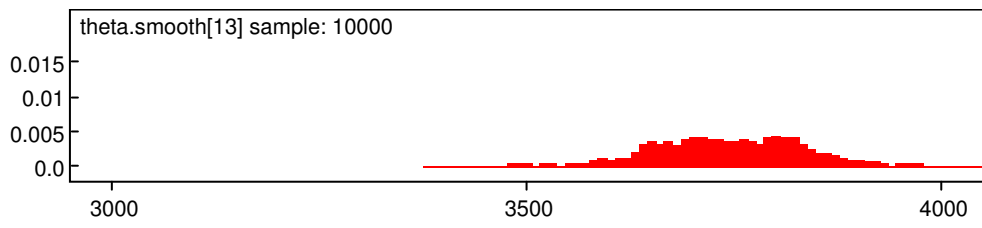
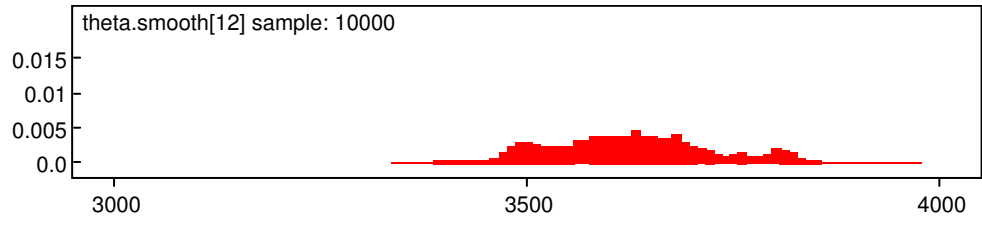
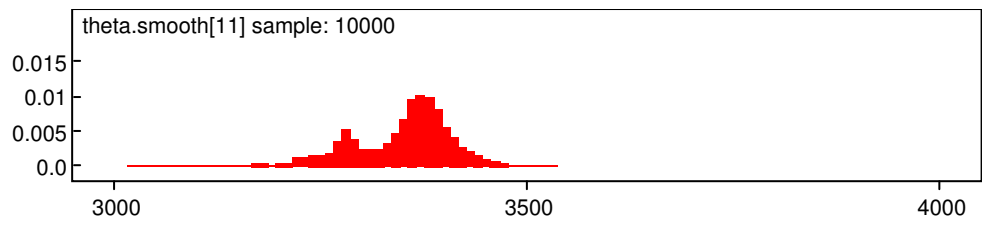
[Data Radio Carbon Calibration Curve](#) (click to open)

[Inits](#) (click to open)

Results







	mean	sd	MC_error
within[1,1]	0.1869	0.3898	0.004748
within[1,2]	0.7512	0.4323	0.005902
within[1,3]	0.0239	0.1527	0.001711
within[1,4]	0.0343	0.182	0.002705
within[1,5]	0.0037	0.06071	6.139E-4
within[2,1]	0.5228	0.4995	0.006329
within[2,2]	0.4718	0.4992	0.006467
within[2,3]	0.0019	0.04355	5.064E-4
within[2,4]	0.0034	0.05821	7.683E-4
within[2,5]	1.0E-4	0.009999	1.0E-4
within[3,1]	0.0057	0.07528	9.239E-4
within[3,2]	0.5221	0.4995	0.008287
within[3,3]	0.0996	0.2995	0.003068
within[3,4]	0.2748	0.4464	0.007434
within[3,5]	0.0978	0.297	0.003884
within[4,1]	0.0	0.0	1.0E-12
within[4,2]	0.0416	0.1997	0.002428
within[4,3]	0.0278	0.1644	0.001661
within[4,4]	0.2778	0.4479	0.006075
within[4,5]	0.6528	0.4761	0.006834
within[5,1]	0.4432	0.4968	0.00613
within[5,2]	0.5496	0.4975	0.006229
within[5,3]	0.0042	0.06467	7.132E-4
within[5,4]	0.0029	0.05377	6.243E-4
within[5,5]	1.0E-4	0.009999	1.0E-4