Assignment 2

2017-05-12

Question 1: Inference for the Poisson parameter λ:

(a) Given a general Gamma(a,b) prior, derive the posterior distribution of λ, given data $(x_1, \ldots, x_n)$, followed by the posterior predictive distribution of $(z_1, \ldots, z_m)$.

The Likelihood is given by:

$$L(\lambda|x) = \prod_{i=1}^{n} \frac{e^{-\lambda}\lambda^{x_i}}{x_i!} = \frac{e^{-n\lambda}\lambda^{\sum x_i}}{\prod_{i=1}^{n}(x_i!)}$$

The Prior is a $Gamma(a, b)$:

$$p(\lambda) = \frac{\beta^{\alpha}}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}, \lambda > 0$$

So the posterior is given by:

$$p(\lambda|x) \propto \frac{\beta^{\alpha}}{\Gamma(\alpha)}\lambda^{\sum x_i+\alpha-1}e^{-(n+\beta)\lambda}, \lambda > 0,$$

which is equivalent to a $Gamma(\sum x_i + \alpha, n + \beta)$, or $Gamma(n\bar{x} + \alpha, n + \beta)$.

The *Posterior Predictive Distribution* for a Poisson parameter λ with data $(z_1, \ldots, z_m)$ is given by:

$$\begin{aligned}
p(z|X) &= \int_0^{\infty} p(z|\lambda)p(\lambda|X)d\lambda \\
&= \int_0^{\infty} Poisson(z|\lambda) \cdot Gamma(\sum x_i + \alpha, n + \beta)d\lambda \\
&= \int_0^{\infty} \frac{e^{-\lambda}\lambda^{z}}{z!} \cdot \frac{(n+\beta)^{\sum x_i+\alpha}}{\Gamma(\sum x_i + \alpha)}\lambda^{\sum x_i+\alpha-1}e^{-(n+\beta)\lambda}d\lambda \\
&= \frac{(n+\beta)^{\sum x_i+\alpha}}{\Gamma(\sum x_i + \alpha)\Gamma(z+1)} \int_0^{\infty} \lambda^{z+\sum x_i+\alpha-1}e^{-(n+\beta+1)\lambda}d\lambda \\
&= \frac{(n+\beta)^{\sum x_i+\alpha}}{\Gamma(\sum x_i + \alpha)\Gamma(z+1)} \cdot \frac{\Gamma(z+\sum x_i+\alpha)}{(n+\beta+1)^{z\sum x_i+\alpha}} \\
&= \frac{\Gamma(z+\sum x_i+\alpha)}{\Gamma(\sum x_i + \alpha)\Gamma(z+1)} \left(\frac{n+\beta}{n+\beta+1}\right)^{\sum x_i+\alpha} \left(\frac{1}{n+\beta+1}\right)^{z}
\end{aligned}$$

(b) Assuming Gamma(a,b) priors for two Poisson parameters λ_1 and λ_2, derive the posterior for $\phi = \lambda_1/\lambda_2$. (Hint: use nuisance parameter $\mu = \lambda_2$).

Firstly, the likelihood function for two Poissons is:

$$p(x_1, x_2|\lambda_1, \lambda_2) = p(x_1|\lambda_1)p(x_2|\lambda_2) = \frac{e^{-\lambda_1}\lambda_1^{x_1}}{x_1!} \cdot \frac{e^{-\lambda_2}\lambda_2^{x_2}}{x_2!}.$$

Reparameterising by $\phi = \frac{\lambda_1}{\lambda_2}$, we get:

$$p(x_1, x_2|\phi, \lambda_2) = \frac{e^{-\phi\lambda_2}(\phi\lambda_2)^{x_1}}{x_1!} \cdot \frac{e^{-\lambda_2}\lambda_2^{x_2}}{x_2!}.$$

We then compute the Fisher Information Matrix:

$$I(\theta)_{ij} = E\left(-\frac{\partial^2 l}{\partial\theta_i \partial\theta_j}\right)$$
$$\implies F(\phi,\lambda_2) = \begin{bmatrix} \frac{\lambda_2}{\phi} & 1 \\ 1 & \frac{1+\phi}{\lambda_2} \end{bmatrix}$$
$$\implies S(\phi,\lambda_2) = F^{-1}(\phi,\lambda_2) = \begin{bmatrix} \frac{\phi(1+\phi)}{\lambda_2} & -\phi \\ -\phi & \lambda_2 \end{bmatrix}$$

Following the algorithm laid out by Bernardo we can define the marginal and conditional asympototic posteriors for ϕ.

$$d_0(\phi,\lambda_2) = \left[\frac{\phi(1+\phi)}{\lambda_2}\right]^{1/2}$$
$$d_1(\phi,\lambda_2) = \left(\frac{\lambda_2}{1+\phi}\right)^{1/2}$$

According to Corollary 1 of Proposition 2 in Barnardo's paper, because the nuisance parameter space $\Lambda(\phi) = \Lambda$ is independent of ϕ, we can factorise the above equations as

$$d_0^{-1}(\phi,\lambda_2) = \frac{1}{\sqrt{\phi(1+\phi)}} \cdot \sqrt{\lambda_2} = a_0(\phi)b_0(\lambda_2)$$
$$d_1^{-1}(\phi,\lambda_2) = \sqrt{\phi(1+\phi)} \cdot \frac{1}{\sqrt{\lambda_2}} = a_1(\phi)b_1(\lambda_2)$$

which implies that the marginal and conditional reference priors are

$$\pi(\phi) \propto a_0(\phi) = \frac{1}{\sqrt{\phi(1+\phi)}}$$
$$\pi(\lambda|\phi) \propto b_1(\lambda_2) = \frac{1}{\sqrt{\lambda_2}}$$

The joint posterior can be derived with the likelihood and joint prior ($d_1^{-1}(\phi,\lambda_2)$) previously derived.

$$\begin{aligned}\pi(\phi,\lambda_2|x_1,x_2) &\propto \pi(x_1,x_2|\phi,\lambda_2)\cdot\pi(\phi,\lambda_2) \\ &\propto e^{-(\phi+1)\lambda_2}\cdot\phi^{x_1-1/2}\,(1+\phi)^{1/2}\cdot\lambda_2^{x_1+x_2-1/2}\end{aligned}$$

which, I'm pretty sure, can be factored as

$$\pi(\phi,\lambda_2|x_1,x_2) \propto Gamma(\lambda_2|x_1+1/2,1)\cdot Gamma\left(\phi|x_1,\frac{1}{\lambda_2}\right)\cdot Beta\left(\frac{\phi}{1+\phi}|3/2,1\right).$$

(c) Derive the Jeffreys prior for (ϕ,μ), and the corresponding marginal posterior for ϕ.

The Jeffreys principle states that the Jeffreys prior is

$$\pi_\phi(\phi) \propto det(I(\phi))^{1/2} = \left(\frac{\lambda_2(1+\phi)}{\lambda_2\phi} - 1\right)^{1/2} = \phi^{-1/2}$$

The marginal posterior can be derived by multiplyin the likelihood by the Jeffrey's prior and then integrating out the nuissance parameter:

$$\begin{aligned}\pi(\phi|x_1,x_2) &\propto \int_0^\infty \frac{e^{-\phi\lambda_2}(\phi\lambda_2)^{x_1}}{x_1!}\cdot\frac{e^{-\lambda_2}\lambda_2^{x_2}}{x_2!}\cdot\frac{1}{\sqrt{\phi}}\quad d\lambda_2 \\ &\propto \phi^{x_1-1/2}\int_0^\infty e^{-(\phi+1)\lambda_2}\cdot\lambda_2^{x_1+x_2}\quad d\lambda_2\end{aligned}$$

Here we solve the integral:

$$\int_0^\infty e^{-(\phi+1)\lambda_2}\cdot\lambda_2^{x_1+x_2}\quad d\lambda_2 = (\phi+1)^{-(x_1+x_2+1)}\cdot\Gamma(x_1+x_2+1)$$

which leads us to

$$\begin{aligned}\pi(\phi|x_1,x_2) &\propto \phi^{x_1-1/2}(\phi+1)^{-(x_1+x_2+1)}\cdot\Gamma(x_1+x_2+1)\\ &\propto \frac{\phi^{x_1-1/2}}{(\phi+1)^{-(x_1+x_2+1)}}\end{aligned}$$

(d) Derive the reference prior for (ϕ,μ), and the corresponding marginal posterior for ϕ.

We can combine the marginal and conditional refererence priors derived above to determine the joint reference prior.

$$\pi(\phi,\lambda_2)=\pi(\phi)\pi(\lambda_2|\phi)=\frac{1}{\sqrt{\phi(1+\phi)}}\cdot\frac{1}{\sqrt{\lambda_2}}$$

Now that we have the marginal reference prior for ϕ, Bernardo states that we can calculate the marginal posterior of ϕ by integrating out the nuisance parameter λ_2.

$$\begin{aligned}\pi(\phi|x_1,x_2) &= \pi(\phi)\int_{\Lambda(\phi)} p(x_1,x_2|\phi,\lambda_2)\pi(\lambda_2|\phi)d\lambda_2\\ &= \pi(\phi)\int_{\Lambda}\frac{e^{-\phi\lambda_2}(\phi\lambda_2)^{x_1}}{x_1!}\cdot\frac{e^{-\lambda_2}\lambda_2^{x_2}}{x_2!}\cdot\frac{1}{\sqrt{\lambda_2}}d\lambda_2\\ &\propto \frac{1}{\sqrt{\phi(1+\phi)}}\int_{\Lambda}e^{-(\phi+1)\lambda_2}\phi^{x_1}\cdot\frac{\lambda_2^{x_1+x_2}}{\sqrt{\lambda_2}}d\lambda_2\\ &\propto \frac{\phi^{x_1}}{\sqrt{\phi(1+\phi)}}\int_{\Lambda}e^{-(\phi+1)\lambda_2}\cdot\frac{\lambda_2^{x_1+x_2}}{\sqrt{\lambda_2}}d\lambda_2\end{aligned}$$

Here we solve the integral

$$\int_{\Lambda}e^{-(\phi+1)\lambda_2}\cdot\frac{\lambda_2^{x_1+x_2}}{\sqrt{\lambda_2}}d\lambda_2=\frac{\Gamma(x_1+x_2-1/2)}{(1+\phi)^{(x_1+x_2+1/2)}}$$

which leaves

$$\pi(\phi|x_1,x_2)\propto\frac{\phi^{x_1-1/2}}{(1+\phi)^{(x_1+x_2+1)}}$$

which is identical to the marginal posterior derived by using the Jeffrey's prior.

(e) Based on (c), consider the posterior for ϕ based on uniform priors for λ_1 and λ_2, and comment on when inference based on this posterior could be quite different from that based on the reference posterior from (d). Give a data example, in terms of resulting intervals. (Hint: the above posteriors are related to known pdfs, and by transformation may be simplified even further.).

Before I begin, I should note that you deal with a lot of these issues yourself Frank, in *Consensus priors for multinomial and binomial ratios.*

Firstly, the parameter ϕ can be regarded as a multinomial ratio, a link established by both yourself – in *Section 2.3: Straitforward alternative derivations* – and Bernardo, who uses the transformation $\omega=\frac{phi}{1+phi}$ to arrive at an alternative expression for the marginal posterior density function given above:

$$pi(\phi|x_1,x_2)|x_1,x_2)\left|\frac{d\phi}{d\omega}\right|=\pi(\omega|x_1,x_2)\propto\omega^{x_1-1/2}(1-\omega)^{x_2-1/2}=Beta(\omega|x_1+1/2,x_2+1/2)$$

Given this is a multinomial case where the Jeffrey's/Reference posterior is given above, you give the Bayes posterior as

$$\pi_B(\phi|x_1,x_2)\propto\frac{\phi^{x_1}}{(1+\phi)^{(x_1+x_2+2)}}$$

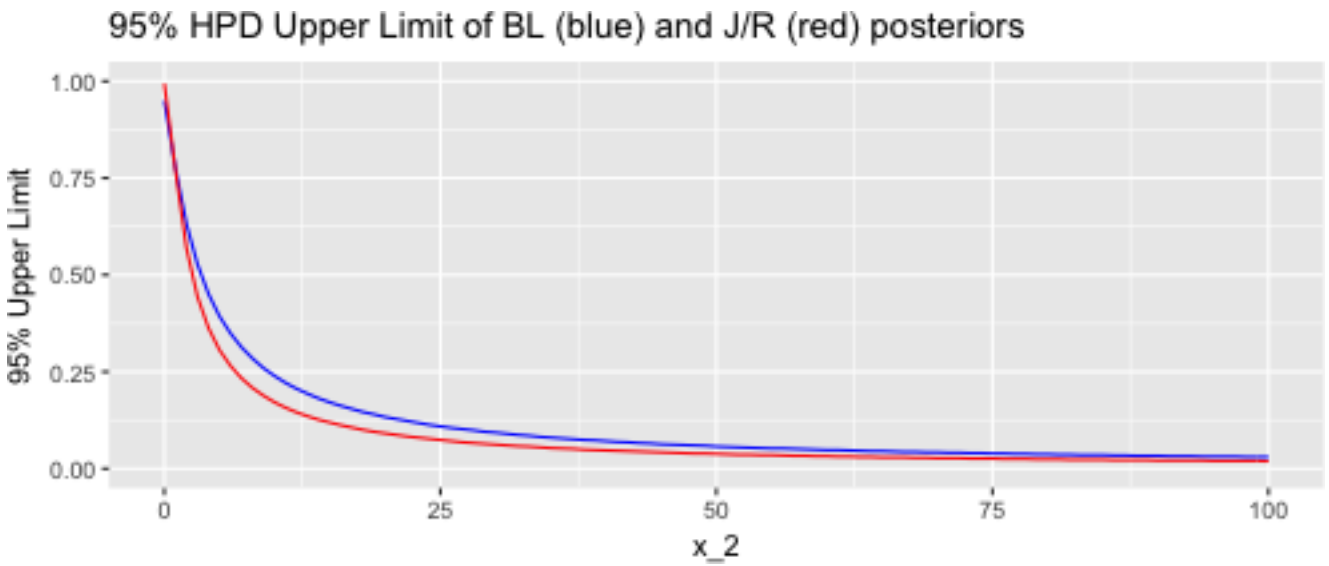
which we can derive a similar alternative expression for:

$$pi(\phi|x_1,x_2)|x_1,x_2)\left|\frac{d\phi}{d\omega}\right|=\pi(\omega|x_1,x_2)\propto\omega^{x_1}(1-\omega)^{x_2}=Beta(\omega|x_1+1,x_2+1)$$

As usual, the conditions where the difference between the Bayes and Jeffrey's/Reference marginal posteriors becomes apparent is when $x_1=0,x_2\to n$. The Jeffrey's/Reference posterior will have more weight closer to zero than the BL posterior, which we

studied more closely in Assignment 1. Also, we know that in terms of mean coverage, BL $\to 1-\alpha, n \to \infty$, which is not case for J/R.

In order to represent the performance of each posterior in the case of $x_n = 0$, I've reused some code from my previous Assignmet.



As we expect, the J/R is waited more towards 0 then the BL.

Question 2: Inference for variance components:

(a) Use e.g. SAS (PROC VARCOMP) to perform a classical analysis of the data in Table 5.1.4 of Box & Tiao (1973), based on finding point estimates only.

I'll be using R. First, let's confirm the results reported by Box & Tiao (1973), using frequentist methods.

```
data("Dyestuff2") #from the lme4 package
D2aov <- aov(Yield ~ Batch, Dyestuff2)
(D2summary <- summary(D2aov))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## Batch        5     42    8.34    0.56   0.73
## Residuals   24    359   14.95
```

```
(sigma2_with <- D2summary[[1]]$`Mean Sq`[2])
```

```
## [1] 14.95
```

```
(sigma2_btw <- (D2summary[[1]]$`Mean Sq`[1] - sigma2_with)/D2summary[[1]]$Df[1])
```

```
## [1] -1.322
```

These results match what was reported by Box & Tiao (1973). Next, we can attempt to fit a linear model using the REML method.

```
(D2REML <- lmer(Yield ~ 1 + (1|Batch), Dyestuff2, REML=TRUE))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: Yield ~ 1 + (1 | Batch)
##    Data: Dyestuff2
## REML criterion at convergence: 161.8
## Random effects:
##  Groups   Name        Std.Dev.
##  Batch    (Intercept) 0.00
##  Residual             3.72
## Number of obs: 30, groups:  Batch, 6
## Fixed Effects:
## (Intercept)
##        5.67
```

```
D2REML <- broom::tidy(D2REML)
(sigma2_with <- D2REML[which(D2REML$group == "Residual"),"estimate"]^2)
```

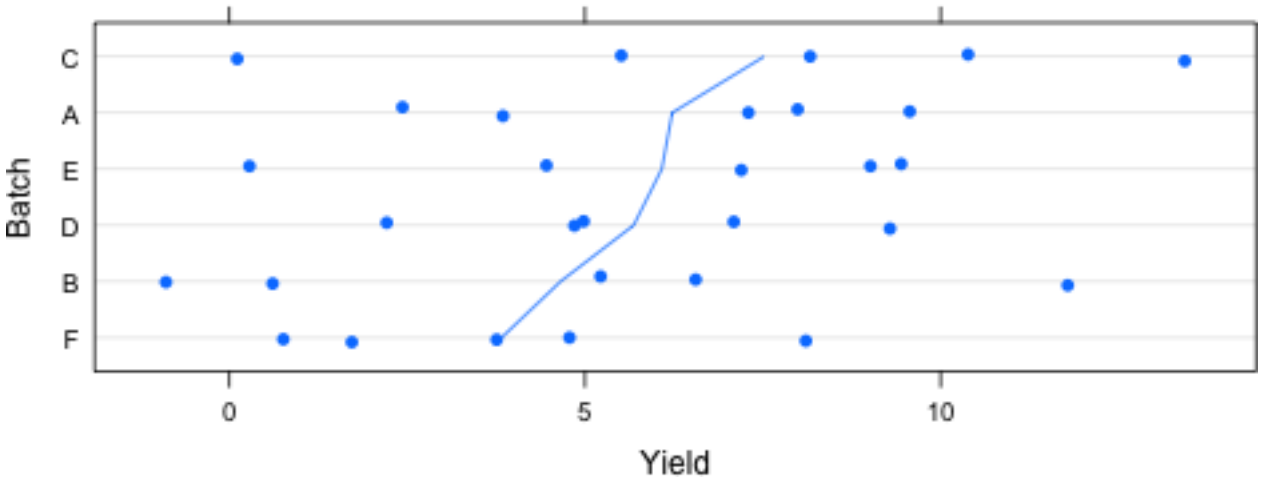
```
## [1] 13.81
```

```
(sigma2_btw <- D2REML[which(D2REML$group == "Batch"),"estimate"]^2)
```

```
## [1] 0
```

Despite the fact that the result for σ_{btw} is reported as 0, we can observe some slight variation between batches using a dotplot (reordering the Batches to produce a smoother average line).

```
dotplot(reorder(Batch, Yield) ~ Yield, Dyestuff2,
        ylab = "Batch", jitter.y = TRUE, aspect = 0.3,
        type = c("p","a"))
```



As a side note, the creator of the lme4 package that contains the Dyestuff2 data set explains the estimate of 0 away by saying that *"indicates that the level of "between-group" variability is not sufficient to warrant incorporating random effects in the model"* – page 25 here: http://lme4.r-forge.r-project.org/lMMwR/lrgprt.pdf.

(b) Use WinBUGS for a Bayesian analysis of (a), and find reasonable point and interval estimates for σ_1^2 and σ_2^2. Include graphs, including one of the joint posterior. [6 marks]

Before we begin, I've relabelled the ICC variable as ICC, instead of rho – which I had previously – just to keep things consistent. Also, I found methods for using the ICC method for models with more that two levels but didn't have time to go into them too much: I don't know if they automatically solve the issues we were discussing in class or not, although I assume they would.

I'll be using **OpenBUGS** and an **R** package called **R2OpenBUGS** to run these simulations.

```
#define the model
dyesmodel <- function(){
  for(i in 1 : batches) {
    m[i] ~ dnorm(theta,  tau.btw)
    for(j in 1 : samples) {
     y[j , i] ~ dnorm(m[i], tau.with)
}}
  sigma2.with <- 1/tau.with
  tau.with ~ dgamma(0.001, 0.001)
  ICC ~ dunif(0,1)
  sigma2.btw <- sigma2.with*(ICC/(1-ICC))
  tau.btw <- 1/sigma2.btw
  theta ~ dflat()
}

# write the model code out to a file
write.model(dyesmodel, "dyesmodel.txt")
model.file1 = paste(getwd(),"dyesmodel.txt", sep="/")

#prepare the data for input into OpenBUGS
batches <- 6; samples <- 5
y <- tbl_df(Dyestuff2) %>%
  mutate(samp = rep(seq(1,5,1),6)) %>%
  spread(Batch, Yield) %>%
  dplyr::select(-1)
y <- as.matrix(y)

data <- list ("batches","samples","y")

#initialization of variables
inits <- function(){list(theta=5, tau.with=1, ICC=0.5)}

#these are the parameters to save
parameters = c("sigma2.with","theta","sigma2.btw","ICC")
```

```
#run the model
dyes.sim <- bugs(
  data, inits, model.file = model.file1, parameters=parameters,
  n.burnin = 1000, n.chains = 3, n.iter = 20000,
  OpenBUGS.pgm=paste0("/Users/benjamin/Applications/wine/",
                      "drive_c/ProgramFiles/OpenBUGS/OpenBUGS323/OpenBUGS.exe"),
  WINE="/opt/local/bin/wine", WINEPATH="/opt/local/bin/winepath", useWINE=T)
attach.bugs(dyes.sim)
kable(dyes.sim$summary, digits = getOption("digits"))
```

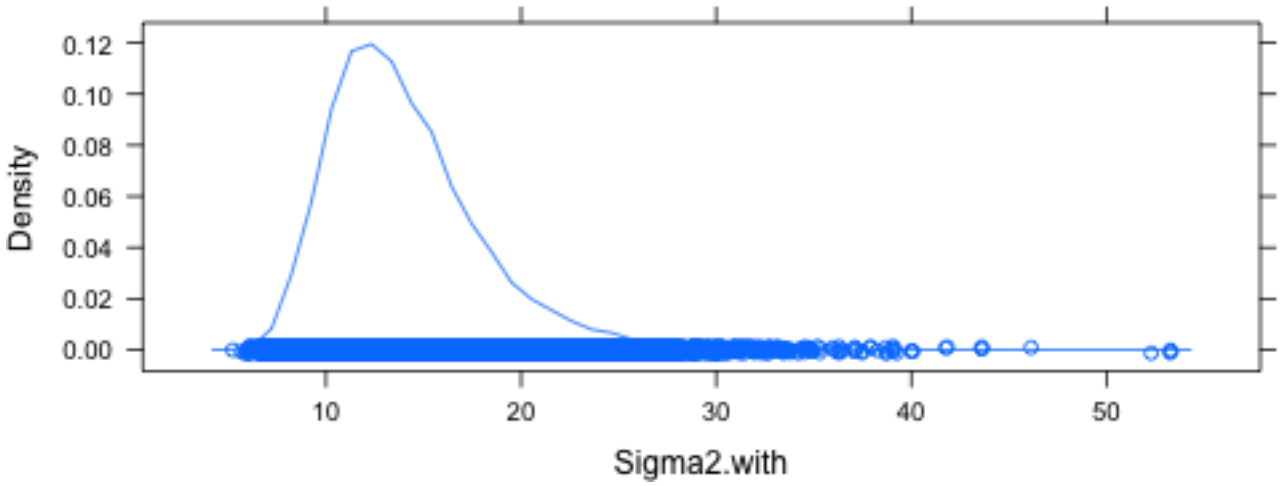
	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
sigma2.with	14.0861	3.9822	8.3640	11.2700	13.3800	16.0900	23.940	1.001	57000
theta	5.6655	1.0159	3.6260	5.0580	5.6660	6.2780	7.684	1.001	19000
sigma2.btw	3.2649	4.9286	0.0781	0.7035	1.7740	3.9422	15.120	1.002	1700
ICC	0.1565	0.1396	0.0057	0.0499	0.1164	0.2254	0.523	1.002	1800
deviance	165.0560	2.8285	161.1000	163.1000	164.5000	166.5000	172.000	1.001	57000

Here are the HPD intervals for each parameter.

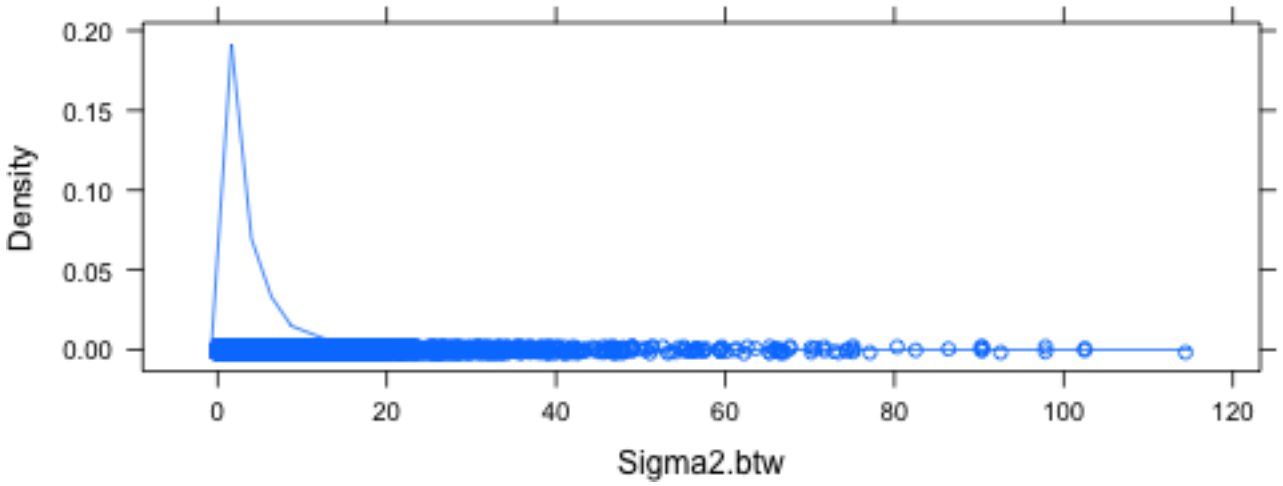
```
library(coda)
dye <- as.mcmc.list(dyes.sim)
HPDinterval(dye)[[1]]
```

```
##                 lower    upper
## deviance    1.605e+02 170.8000
## ICC         8.877e-04   0.4266
## sigma2.btw  8.574e-03  10.5800
## sigma2.with 7.357e+00  21.7200
## theta       3.725e+00   7.7390
## attr(,"Probability")
## [1] 0.95
```

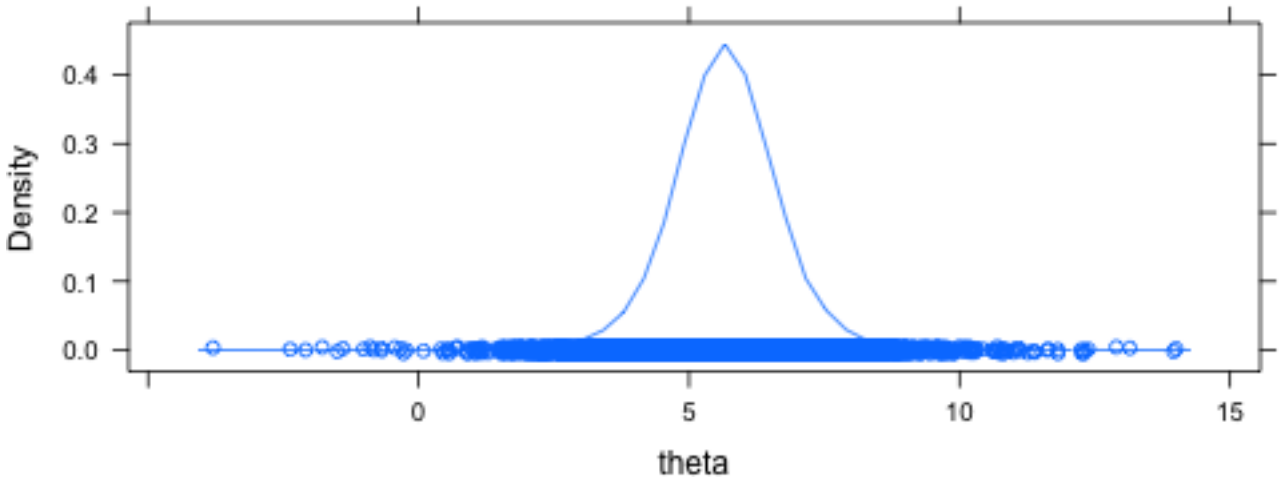
```
dyessimlist <- tbl_df(dyes.sim$sims.list)
densityplot(dyessimlist$sigma2.with, xlab = "Sigma2.with")
```



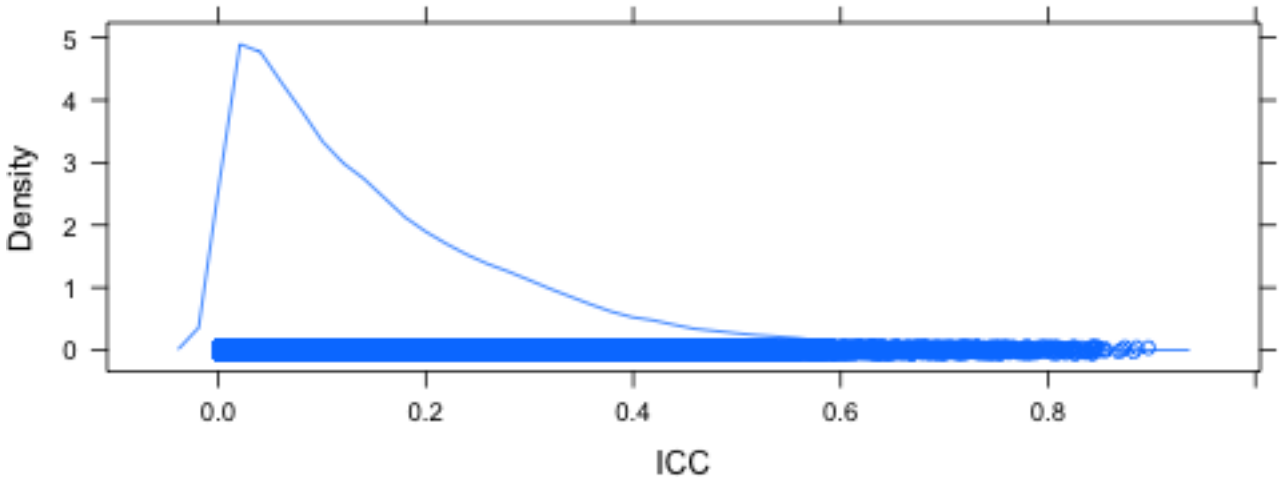

```
densityplot(dyessimlist$sigma2.btw, xlab = "Sigma2.btw")
```



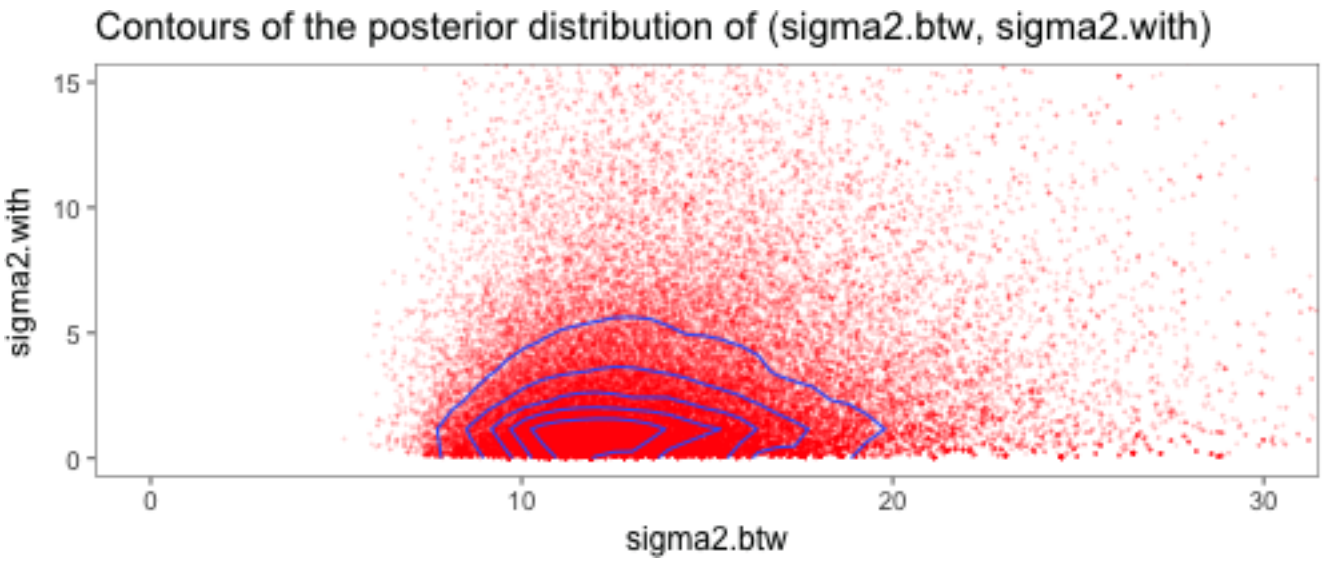

```
densityplot(dyessimlist$theta, xlab = "theta")
```




```
densityplot(dyessimlist$ICC, xlab = "ICC")
```




```
df <- tbl_df(cbind(rBtw = dyessimlist$sigma2.btw, rWith = dyessimlist$sigma2.with))
ggplot(data = df, aes(x=rWith,y=rBtw)) +
  geom_point(size=0.1, colour = "red", alpha = 0.1) +
  geom_density_2d() +
  coord_cartesian(xlim = c(0,30), ylim = c(0,15)) +
  theme_few() +
  labs(x="sigma2.btw",y="sigma2.with",
       title="Contours of the posterior distribution of (sigma2.btw, sigma2.with)")
```



(c) Box & Tiao also studied a 3-component model (Table 5.3.1).

i. Derive central credible intervals, for the 3 individual components, based on Table 5.3.3.

ii. Use WinBUGS to do the same, and include graphs.

iii. While not going as far as Box & Tiao's Figure 5.3.2, produce a graph of the joint posterior of σ_2^2 and σ_3^2, and one of σ_2^2/σ_3^2

Here is where my answers started to diverge somewhat more drastically from those reported by Box and Tiao, which I will discuss further on.

First, let's confirm the results in the text.

```
df <- readRDS("Question2/Data/Q2c.rds")
(BT3 <- summary(aov(Sample~(I/J/K),df)))
```

```
##             Df Sum Sq Mean Sq
## I            9  240.2   26.69
## I:J         10  122.9   12.29
## I:J:K       20   20.9    1.04
```

```
(m1 <- BT3[[1]]$`Mean Sq`[3])
```

```
## [1] 1.045
```

```
(m2 <- (BT3[[1]]$`Mean Sq`[2]-BT3[[1]]$`Mean Sq`[3])/2)
```

```
## [1] 5.624
```

```
(m3 <- (BT3[[1]]$`Mean Sq`[1]-BT3[[1]]$`Mean Sq`[2])/4)
```

```
## [1] 3.599
```

```
(mAll <- m1+m2+m3)
```

```
## [1] 10.27
```

```
(c(m1/mAll,m2/mAll,m3/mAll))
```

```
## [1] 0.1018 0.5477 0.3505
```

Next, the model for simulating in BUGS. My problems with the models that follow concern whether or not to simulate the mean of each level to use to centre the next. Here I have generated values for the mean of each Batch (I), Sample (J) and Test (K) and used them to generate samples from the joint posterior. In the next model I followed the model that (I believe?) was laid out by Box and Tiao and instead assumed that each mean was normally distributed about 0. Either way, each model type – applied to either problem – seemed to give me very different results from Box et. als original analysis, hence my confusion.

If I am wrong in both cases, my only thought is that I have entered the data in reverse: i.e. that instead of a $10x2x2$ sampling scheme, I've actually entered a $2x2X10$. Anyway, I digress.

```
#define the model
threecompmodel <- function(){
```

```
    for( i in 1 : I ) {
        a[i] ~ dnorm(theta, tauI)
        for( j in 1 : J ) {
            b[i , j] ~ dnorm(a[i], tauJ)
            for( k in 1 : K ) {
                e[i , j , k] <- mu0 + a[i] + b[i , j]
                y[i , j , k] ~ dnorm(e[i , j , k], tauK)
}}}

    ## Priors
  theta ~ dnorm(0.0,1.00E-04) # Batch Mean
    mu0   ~ dnorm(0.0,1.00E-04) # Sample Mean
    tauI  ~ dgamma(0.001, 0.001) # Between Batch variation
    tauJ  ~ dgamma(0.001, 0.001) # Between Sample variation
    tauK  ~ dgamma(0.001, 0.001) # Between Test variation
    sigma2I <- 1 / tauI
    sigma2J <- 1 / tauJ
    sigma2K <- 1 / tauK
    sigma2JK <- sigma2J / sigma2K
    denom <- sigma2I + sigma2J + sigma2K
    r1 <- sigma2I / denom
    r2 <- sigma2J / denom
    r3 <- sigma2K / denom
}
# write the model code out to a file
write.model(threecompmodel, "threecompmodelmodel.txt")
model.file.3comp = paste(getwd(),"threecompmodelmodel.txt", sep="/")

#prepare the data for input into OpenBUGS

data <- list(I=10,J=2,K=2,
             y = structure(
               .Data=c(2.004,  2.713,  0.602,  0.252,
                       4.342,  4.229,  3.344,  3.057,
                       0.869, -2.621, -3.896, -3.696,
                       3.531,  4.185,  1.722,  0.380,
                       2.579,  4.271, -2.101,  0.651,
                      -1.404, -1.003, -0.755, -2.202,
                      -1.676, -0.208, -9.139, -8.653,
                       1.670,  2.426,  1.834,  1.200,
                       2.141,  3.527,  0.462,  0.665,
                      -1.229, -0.596,  4.471,  1.606),
               .Dim=c(10,2,2)))

#initialization of variables
inits <- function(){
  list(mu0 = 0,theta=0,
       a = c(0,0,0,0,0,0,0,0,0,0),
         b = structure(.Data = c(0,0,0,0,0,0,0,0,0,0,
                                 0,0,0,0,0,0,0,0,0,0),
                       .Dim = c(10,2)),
       tauI = 1.0,tauJ = 1.0,tauK = 1.0)
}
#these are the parameters to save
parameters = c("sigma2I","sigma2J","sigma2K", "sigma2JK", "r1", "r2", "r3")

#run the model
threecomp.sim <- bugs(data, inits, model.file = model.file.3comp, parameters=parameters,
                      n.burnin = 1000, n.chains = 3, n.iter = 20000,
                      OpenBUGS.pgm=paste0("/Users/benjamin/Applications/wine/",
                                          "drive_c/ProgramFiles/OpenBUGS/OpenBUGS323/OpenBUGS.exe"),
```

```
                    WINE="/opt/local/bin/wine", WINEPATH="/opt/local/bin/winepath",useWINE=T)
attach.bugs(threecomp.sim)
kable(threecomp.sim$summary,digits = getOption("digits"))
```

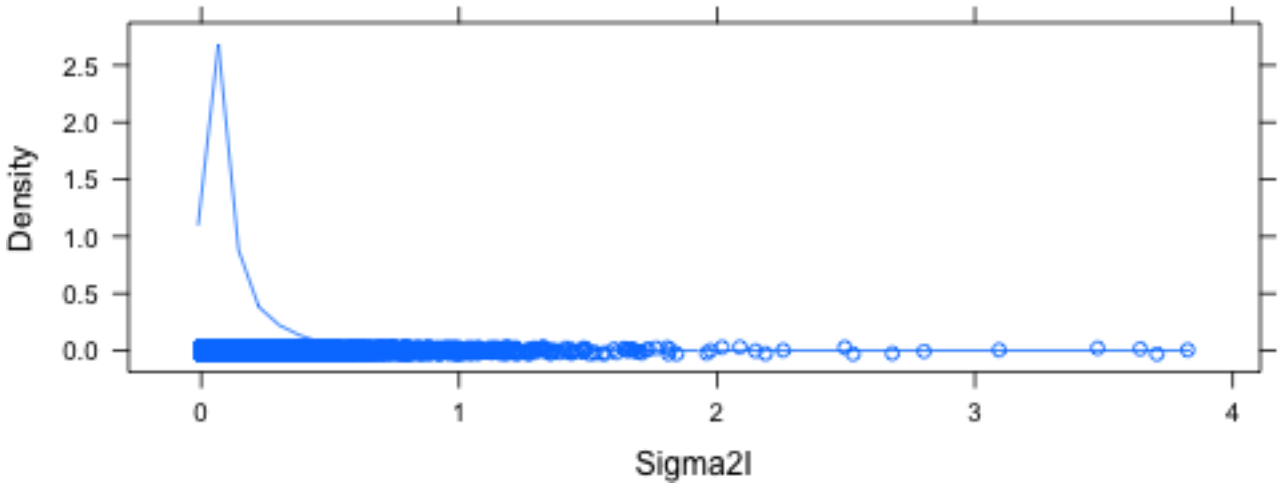
	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
sigma2I	0.0605	0.1309	0.0007	0.0045	0.0162	0.0584	0.4007	1.001	13000
sigma2J	0.1956	0.4667	0.0008	0.0066	0.0313	0.1569	1.5070	1.001	13000
sigma2K	10.4409	2.5275	6.6040	8.6450	10.0800	11.8200	16.4000	1.001	57000
sigma2JK	0.0197	0.0480	0.0001	0.0007	0.0031	0.0155	0.1520	1.001	12000
r1	0.0057	0.0114	0.0001	0.0004	0.0016	0.0057	0.0371	1.001	15000
r2	0.0175	0.0374	0.0001	0.0006	0.0031	0.0152	0.1313	1.001	14000
r3	0.9768	0.0390	0.8609	0.9745	0.9914	0.9973	0.9996	1.008	870
deviance	206.2200	2.5571	202.3000	204.6000	205.7000	207.4000	212.5000	1.001	30000

Here are the HPD intervals for each parameter.

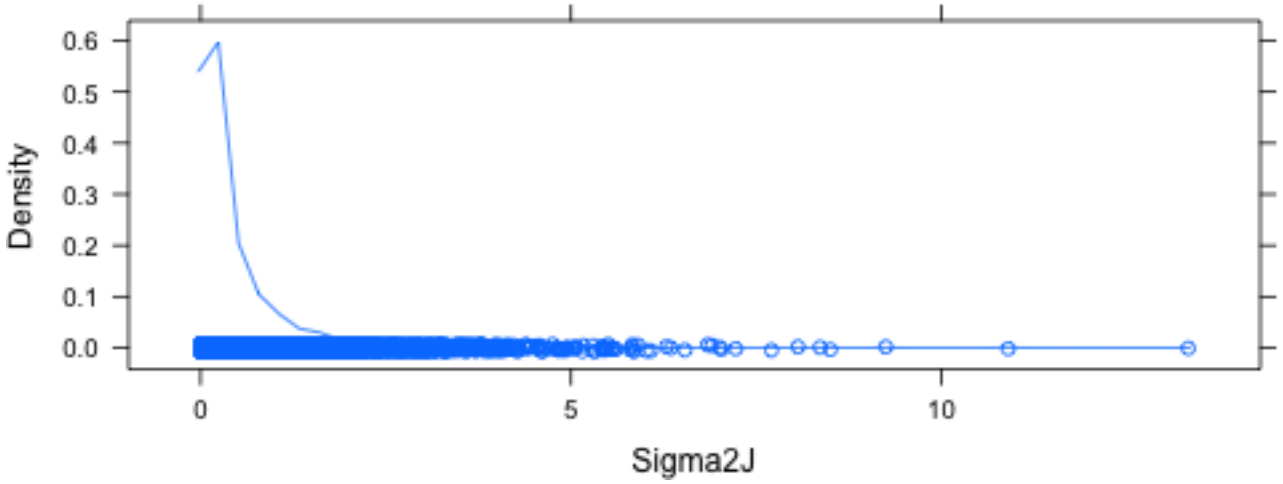
```
library(coda)
threec <- as.mcmc.list(threecomp.sim)
HPDinterval(threec)[[1]]
```

```
##              lower     upper
## deviance 2.019e+02 212.10000
## r1       1.424e-05   0.02792
## r2       1.460e-05   0.09769
## r3       8.944e-01   0.99990
## sigma2I  1.551e-04   0.29340
## sigma2J  1.366e-04   1.05800
## sigma2JK 1.460e-05   0.10880
## sigma2K  6.025e+00  15.42000
## attr(,"Probability")
## [1] 0.95
```

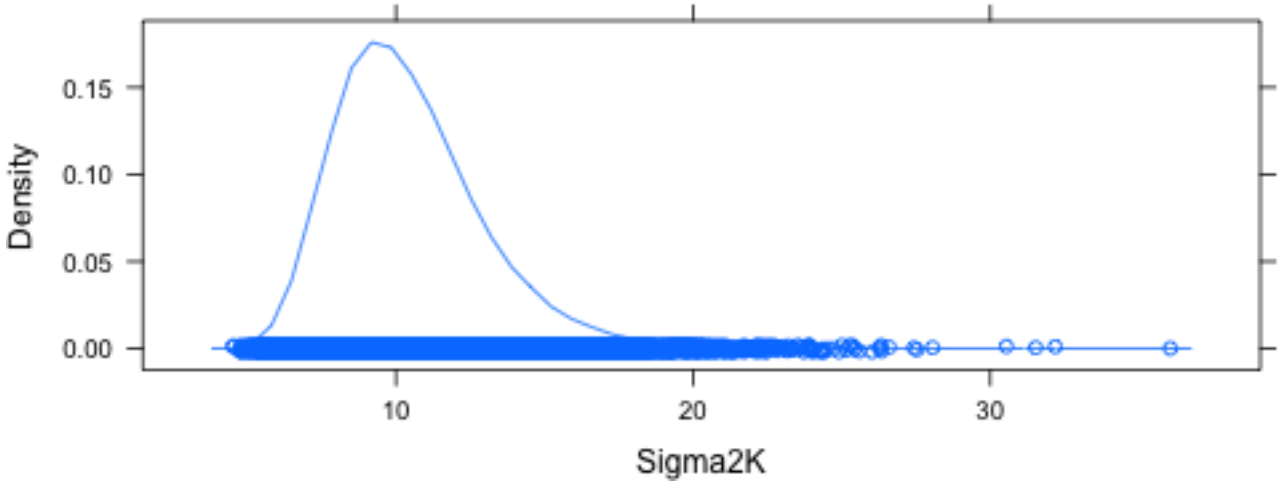
```
threecompsimlist <- tbl_df(threecomp.sim$sims.list)

densityplot(threecompsimlist$sigma2I, xlab = "Sigma2I")
```



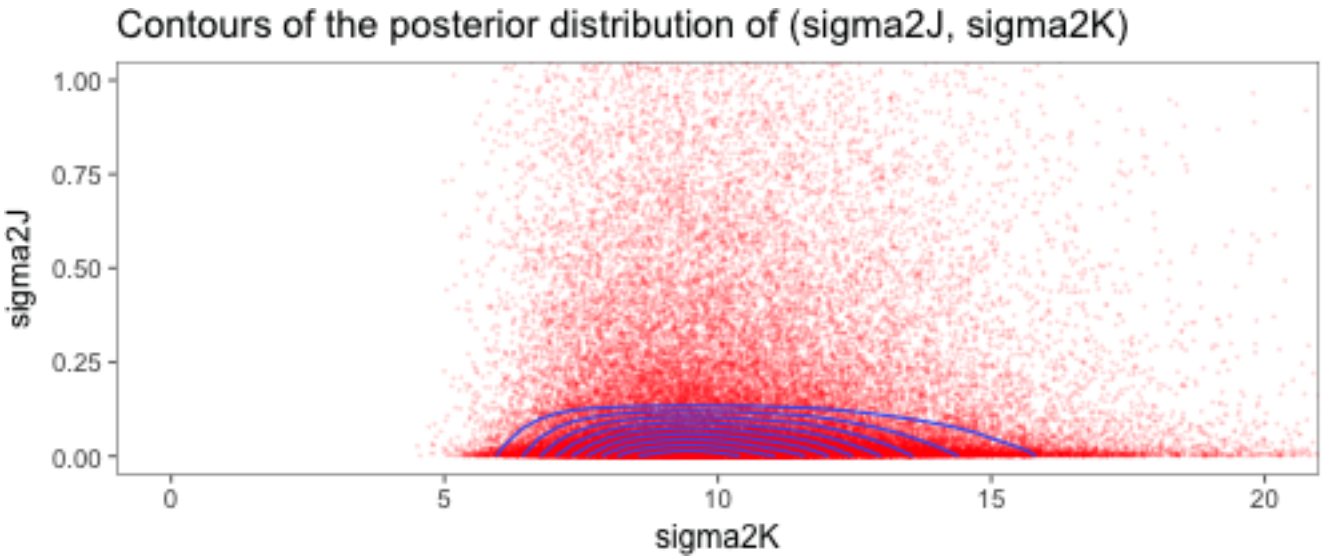

```
densityplot(threecompsimlist$sigma2J, xlab = "Sigma2J")
```



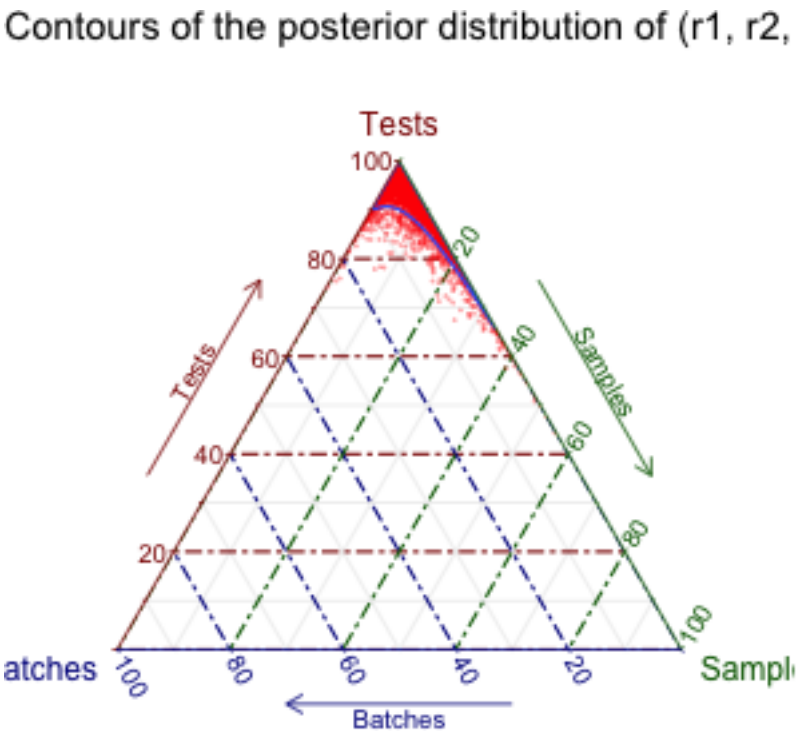

```
densityplot(threecompsimlist$sigma2K, xlab = "Sigma2K")
```




```
df <- tbl_df(cbind(rBatches = threecompsimlist$r1,
                   rSamples = threecompsimlist$r2,
                   rTests = threecompsimlist$r3))

ggplot(data = threecompsimlist, aes(x=sigma2K,y=sigma2J)) +
  geom_point(size=0.1, colour = "red", alpha = 0.1) +
  geom_density_2d() +
  coord_cartesian(xlim = c(0,20), ylim = c(0,1)) +
  theme_few() +
  labs(x="sigma2K",y="sigma2J",
       title="Contours of the posterior distribution of (sigma2J, sigma2K)")
```




```
ggtern(data = df, aes(x=rBatches,z=rSamples,y=rTests)) +
  theme_rgbw() +
  geom_point(size=0.1, colour = "red", alpha = 0.25) +
  geom_confidence_tern(breaks = c(0.95)) +
  theme_clockwise() +
  labs(x="Batches",z="Samples",y="Tests",
       title="Contours of the posterior distribution of (r1, r2, r3)")
```



(d) Box, Hunter & Hunter (1976, Chapter 17.3) studied a pigment paste example with three components, focusing on point estimates only. Use WinBUGS again to perform a Bayesian analysis. Include graphs.

Again, any issues that are present in the model above will be reproduced here.

```
#define the model
pigmentmodel <- function(){
    for( i in 1 : I ) {
        a[i] ~ dnorm(0.0, tauI)
        for( j in 1 : J ) {
            b[i , j] ~ dnorm(0.0, tauJ)
            for( k in 1 : K ) {
                e[i , j , k] <- theta + a[i] + b[i , j]
```

```
                y[i , j , k] ~ dnorm(e[i , j , k], tauK)
  }}}

    ## Priors
  theta ~ dflat() # sample mean
    tauI ~ dgamma(0.001, 0.001) # Between Batch variation
    tauJ ~ dgamma(0.001, 0.001) # Between Sample variation
    tauK ~ dgamma(0.001, 0.001) # Between Test variation
    sigma2I <- 1 / tauI
    sigma2J <- 1 / tauJ
    sigma2K <- 1 / tauK
    sigma2JK <- sigma2J / sigma2K
    denom <- sigma2I + sigma2J + sigma2K
    r1 <- sigma2I / denom
    r2 <- sigma2J / denom
    r3 <- sigma2K / denom
}
# write the model code out to a file
write.model(pigmentmodel, "dyes2model.txt")
model.file.pigment = paste(getwd(),"dyes2model.txt", sep="/")

#prepare the data for input into OpenBUGS

data <- list(I=15,J=2,K=2,
             y = structure(
               .Data=c(40,39,30,30,
                       26,28,25,26,
                         29,28,14,15,
                         30,31,24,24,
                         19,20,17,17,
                         33,32,26,24,
                         23,24,32,33,
                         34,34,29,29,
                         27,27,31,31,
                         13,16,27,24,
                         25,23,25,27,
                         29,29,31,32,
                         19,20,29,30,
                         23,23,25,25,
                         39,37,26,28),
               .Dim=c(15,2,2)))

#initialization of variables
inits <- function(){
  list(theta=0,
       a = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
         b = structure(
           .Data = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
           .Dim = c(15,2)),
  tauI = 1.0,tauJ = 1.0,tauK = 1.0)
}

WINE="/opt/local/bin/wine"
WINEPATH="/opt/local/bin/winepath"
OpenBUGS.pgm=paste0("/Users/benjamin/Applications/wine/",
                    "drive_c/ProgramFiles/OpenBUGS/OpenBUGS323/OpenBUGS.exe")

#these are the parameters to save
parameters = c("sigma2I","sigma2J","sigma2K", "sigma2JK", "r1", "r2", "r3")
```

```
#run the model
pigment.sim <- bugs(
  data, inits, model.file = model.file.pigment, parameters=parameters,
  n.burnin = 1000, n.chains = 3, n.iter = 20000,
  OpenBUGS.pgm=paste0("/Users/benjamin/Applications/wine/",
                      "drive_c/ProgramFiles/OpenBUGS/OpenBUGS323/OpenBUGS.exe"),
  WINE="/opt/local/bin/wine", WINEPATH="/opt/local/bin/winepath", useWINE = T)
attach.bugs(pigment.sim)
kable(pigment.sim$summary,digits = getOption("digits"))
```

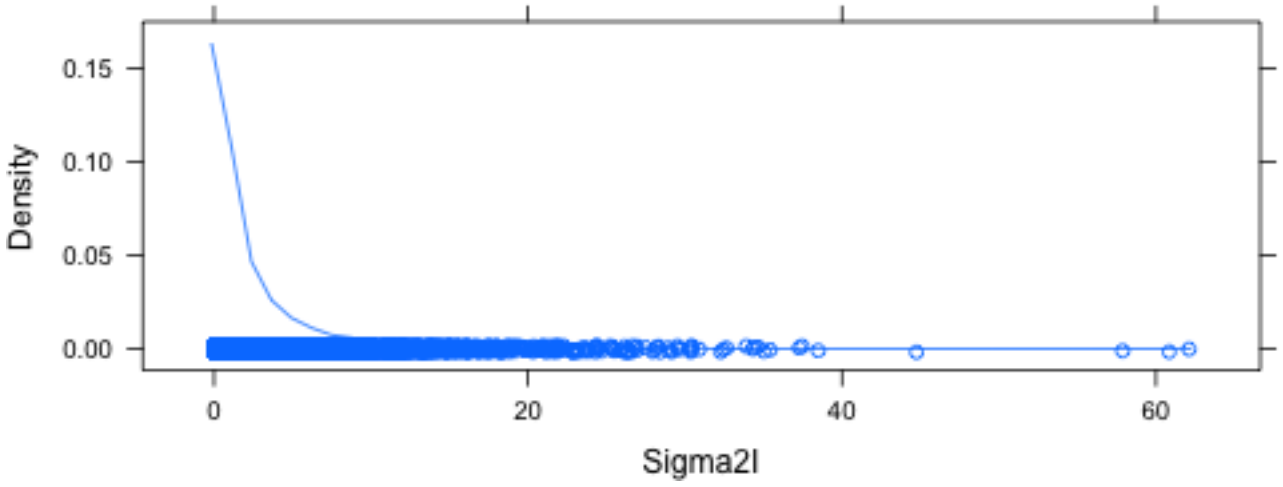
	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
sigma2I	0.9717	2.3667	0.0010	0.0112	0.0920	0.7396	7.5930	1.004	690
sigma2J	9.3620	8.4985	0.0025	1.4590	8.0850	14.5700	29.0502	1.036	110
sigma2K	27.8058	8.4188	15.0100	21.4200	26.6600	33.0600	46.7600	1.006	440
sigma2JK	0.4286	0.4483	0.0001	0.0431	0.3129	0.6632	1.5550	1.033	110
r1	0.0240	0.0514	0.0000	0.0003	0.0025	0.0201	0.1831	1.004	690
r2	0.2379	0.1904	0.0001	0.0400	0.2322	0.3911	0.6033	1.036	100
r3	0.7381	0.1890	0.3814	0.5869	0.7410	0.9175	0.9992	1.006	410
deviance	367.1263	14.1921	341.3000	355.6000	366.8000	380.5000	388.7000	1.008	290

Here are the HPD intervals for each parameter.

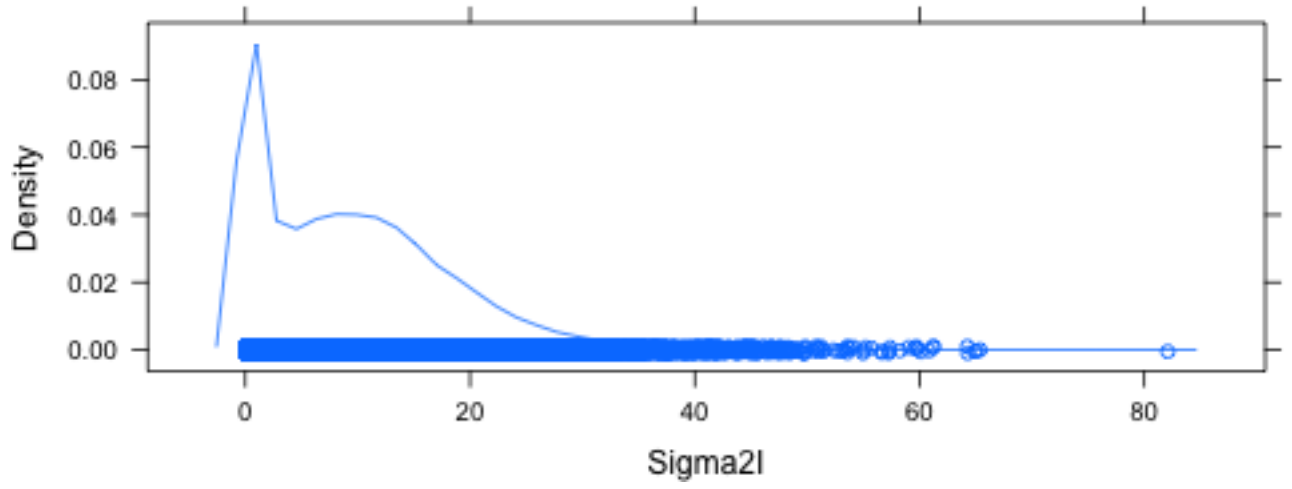
```
library(coda)
pig <- as.mcmc.list(pigment.sim)
HPDinterval(pig)[[1]]
```

```
##              lower    upper
## deviance 3.423e+02 389.5000
## r1       3.056e-06   0.1378
## r2       6.893e-06   0.5571
## r3       4.245e-01   0.9999
## sigma2I  1.289e-04   5.5490
## sigma2J  2.611e-04  25.0400
## sigma2JK 7.445e-06   1.3010
## sigma2K  1.365e+01  44.7100
## attr(,"Probability")
## [1] 0.95
```

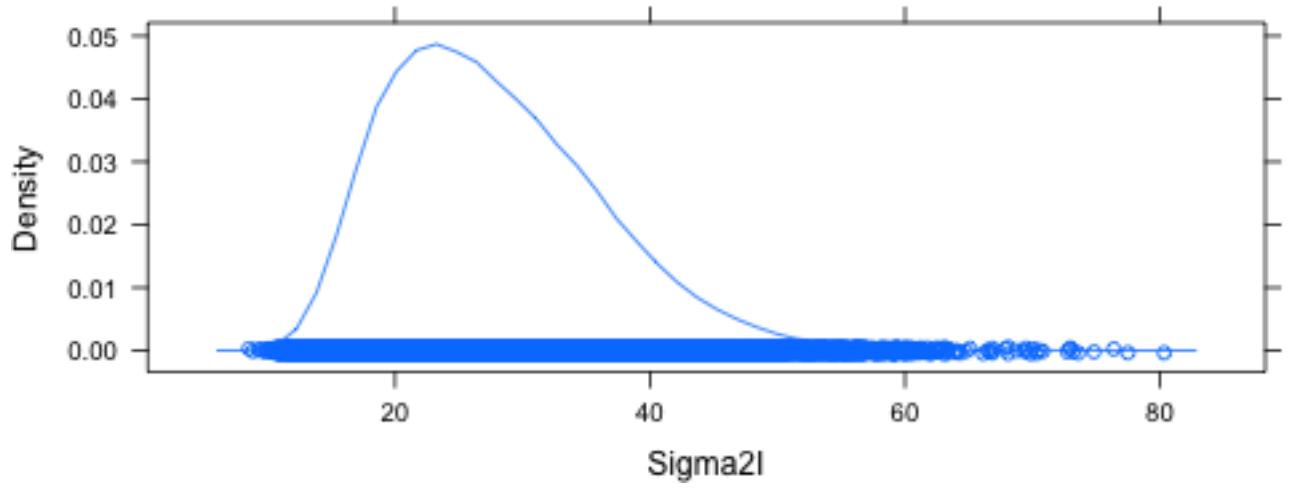
```
pigsimlist <- tbl_df(pigment.sim$sims.list)

densityplot(pigsimlist$sigma2I, xlab = "Sigma2I")
```




```
densityplot(pigsimlist$sigma2J, xlab = "Sigma2I")
```




```
densityplot(pigsimlist$sigma2K, xlab = "Sigma2I")
```




```
df <- tbl_df(cbind(rBatches = pigsimlist$r1, rSamples = pigsimlist$r2, rTests = pigsimlist$r3))
```

```
ggtern(data=df,aes(x=rBatches,z=rSamples,y=rTests)) +
  theme_rgbw() +
  geom_point(size=0.1, colour = "red") +
  geom_confidence_tern(breaks = c(0.5,0.7,0.9)) +
  theme_clockwise() +
  labs(x="Batches",z="Samples",y="Tests",
       title="Contours of the posterior distribution of (r1, r2, r3)")
```

Contours of the posterior distribution of (r1, r2, r3)