1. Consumers of broiler legs were given data loggers which measured the actual cooking time, t, and temperature, T, in the oven. The data show modestly negative correlation between t and T. Explain why this could be so. Compute the posterior distribution in WinBUGS assuming the following model for the logarithms. Compute also predictive distribution for a 'next' consumer. What is the probability that he/she will cook (A) less than 50 minutes, (B) over 50 min, but $T > 175$, (C) over 50 min, but $T \in [135, 175]$, (C) over 50 min, but $T < 135$? Is the model prediction realistic? Replace some of the datapoints with NAs and see how well/badly the model can predict the missing values. (You might also try with the original values without log-transform, but then you probably get some negative values in predictions).

$$\begin{bmatrix} \log(t) \\ \log(T) \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \rho\, \sigma_1\sigma_2 \\ \rho\, \sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix} \right)$$

$$\mu_i \sim \mathrm{N}(0, 0.001) \qquad \sigma_i^2 \sim \mathrm{Gamma}(0.01, 0.01) \qquad \rho \sim \mathrm{U}(-1, 1)$$

Hints: data transformation within code, `inverse()` for computing the inverse covariance matrix (which needs to be positive definite matrix, and this parametrization is. Not any self-made parametrization would produce pos.def. matrix), `step()` for the prediction probabilities.

As a second model, use the Wishart-prior construction (see e.g. 'Camel' in Examples Vol3 in OpenBUGS), in which we have τ as the precision matrix (inverse of covariance matrix) in the bivariate normal model for $(\log(t), \log(T))$ and $\tau \sim \mathrm{Wish}(R, k)$. A usual uninformative choice is to take $k = 2$ (for 2D problems) and R to represent a low precision matrix, e.g. Diag(0.001). In this prior: $E(\tau) = Rk$. Make both models run within the same code, compare with the previous results.

```
model{
mu[1] ~ dnorm(0,0.001);  mu[2] ~ dnorm(0,0.001)
T[1:2,1:2] <- inverse(C[1:2,1:2])
C[1,1]  <- s12 # sigma1 squared
s12 ~ dgamma(0.01,0.01)
C[2,2] <- s22 # sigma2 squared
s22 ~ dgamma(0.01,0.01)
C[1,2] <- C[2,1]
C[2,1] <- rho*sqrt(C[1,1]*C[2,2])
rho ~ dunif(-1,1)
#T[1:2,1:2] ~ dwish(R[1:2,1:2],2)
# the R in BUGS-Wishart needs to be a precision matrix, not covariance matrix
#R[1,1] <- 0.001
#R[1,2] <- 0
#R[2,1] <- 0
#R[2,2] <- 0.001

for(i in 1:N){
logtimetemp[i,1] <- log(timetemp[i,1])
logtimetemp[i,2] <- log(timetemp[i,2])
logtimetemp[i,1:2] ~ dmnorm(mu[1:2],T[1:2,1:2])  }
```

```
logttpred[1:2] ~ dmnorm(mu[1:2],T[1:2,1:2])
ttpred[1]<-exp(logttpred[1])
ttpred[2]<-exp(logttpred[2])
prisk[1] <- step(ttpred[1]-50)*step(ttpred[2]-175)
prisk[2] <- step(ttpred[1]-50)*step(ttpred[2]-132)*step(175-ttpred[2])
prisk[3] <- step(ttpred[1]-50)*step(132-ttpred[2])
prisk[4] <- step(50-ttpred[1])
}
list(N=19, timetemp=structure(.Data=c(
   43,   179,
   44,   217,
   47,   206,
   49,   185,
   49,   166,
   53,   193,
   53,   180,
   56,   176,
   58,   167,
   59,   180,
   61,   132,
   62,   152,
   62,   136,
   72,   178,
   73,   168,
   78,   149,
   84,   169,
   99,   161,
  105,   148),.Dim=c(19,2)))
```

2. Study the following BUGS code for implementing a hypergeometric distribution (by Andrew Millard, 2001). Check the posterior probability using some sample data. (Could be compared to the simple binomial model). This shows how new distributions might be a bit tricky to define within the BUGS code. But it's possible. The permutation model in the lecture notes produces a hypergeometric distribution too, when you take the sum of K elements of it (hypergeometric sample size K), but is not so applicable for learning the underlying unknown parameters from data.

```
model{ # hypergeometrical distribution
      Y1 <- Y+1  # allows for Y=0
      Y1 ~ dcat(prob[])
      for (k in 1:n+1){
              # likelihood function
              # using choice[k] means that the function using
              # factorials is not computed when it is undefined
              choice[k] <- step(Np-k+1) * step(k-1-(Np-N+n)) + 1
              probc[k,1] <- 0
```

```
                log(probc[k,2]) <-
 (logfact(Np)-logfact(k-1)-logfact(Np-k+1)) +
 (logfact(N-Np)-logfact(n-k+1)-logfact((N-Np)-(n-k+1))) -
 (logfact(N)-logfact(n)-logfact(N-n))

                prob[k] <- probc[k,choice[k]]
        }

        # uniform prior on underlying proportion in population of size N
        for (i in 1:N+1){
                pr.Np1[i] <- 1/(N+1)
        }
        Np1 ~ dcat(pr.Np1[])
        Np <- Np1 - 1
        p <- Np/N
}
 DATA:
 list(N=50, Y=15, n=20)
 ALTERNATIVE DATA:
 list(N=50, Y=0,n=20)
 INITS:
 list(Np1=30)
```

Variable Y is the outcome of the hypergeometric distribution ('sampling without replacement'), when the sample size is n, and the population size N. Possible values include zero, so we construct variable $Y_1 = Y + 1$ to be used in the categorical model for Y_1 which assumes outcomes in the range $1, 2, 3, \ldots, n + 1$. For each of these possible data values, probabilities are written according to the hypergeometric probability. The log of this is written in `log(probc[k,2])<-`. However, some combinations of its arguments give invalid results so that for those the probability should be defined zero. This is achieved by using `choice[k]` to rule out invalid values. The final correct probabilities are obtained in `prob[k]` for each possible outcome $k = 1, 2, \ldots, n + 1$. The prior is a discrete uniform distribution for `Np`, again using categorical distribution with parameters assigned as $1/(N + 1)$. So, when the outcome Y is observed, we obtain posterior distribution of `Np`, the unknown number of 'red balls' in a bag of N balls.

3. In the CO_2-model, use DIC to select the best model among those with different numbers of trigonometric terms with different periodicity ($12/k$ months with $k = 1, 2, 3, 4$). You might also include a model with exponential term.

```
 model{
 tau ~ dgamma(0.01,0.01);
 for(i in 1:2+2*K){ a[i] ~ dnorm(0,0.00001)}
 for(i in 1:N){   m[i] <-i
 x[i] ~ dnorm(mu[i],tau)
 mu[i] <- a[1]+a[2]*i
   +inprod(a[3:2+K],seasonCOS[i,1:K])
```

```
   +inprod(a[3+K:2+2*K],seasonSIN[i,1:K])
 for(k in 1:K){
 seasonCOS[i,k] <- cos(2*pi*i/(12/k))
 seasonSIN[i,k] <- sin(2*pi*i/(12/k))
 }}
 pi <- 3.1415926
}
 list(tau=1) # initials
 list(K=3,N=120,x=c(368.18,366.87,366.94,368.27,369.62,370.47,371.44,
 372.39,373.32,373.77,373.13,371.51,369.59,368.12,368.38,369.64,
 371.11,372.38,373.08,373.87,374.93,375.58,375.44,373.91,371.77,
 370.72,370.5,372.19,373.71,374.92,375.63,376.51,377.75,378.54,
 378.21,376.65,374.28,373.12,373.1,374.67,375.97,377.03,377.87,
 378.88,380.42,380.62,379.66,377.48,376.07,374.1,374.47,376.15,
 377.51,378.43,379.7,380.91,382.2,382.45,382.14,380.6,378.6,
 376.72,376.98,378.29,380.07,381.36,382.19,382.65,384.65,384.94,
 384.01,382.15,380.33,378.81,379.06,380.17,381.85,382.88,383.77,
 384.42,386.36,386.53,386.01,384.45,381.96,380.81,381.09,382.37,
 383.84,385.42,385.72,385.96,387.18,388.5,387.88,386.38,384.15,
 383.07,382.98,384.11,385.54,386.93,387.42,388.77,389.46,390.18,
 389.43,387.81,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA))
```

It seems that a model with periods 12 months and 6 months gives the lowest DIC.

4. Go through the steps needed to implement Zellner's prior in the 'soft drinks' example code. The main changes are:

```
# calculation of xtx
for (i in 1:P){ for (j in 1:P){
   inverse.V[i,j] <- inprod( x[,i] , x[,j] )
}}
# calculation of the elements of prior precision matrix
for(i in 1:P){ for (j in 1:P){
  prior.T[i,j] <- inverse.V[i,j] * tau /c2
}}
# multivariate prior for the beta vector
beta[1:P] ~ dmnorm( mu.beta[], prior.T[,] )
```

Full code file is on course pages. (Also from the book BMUW web pages).