UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Homework II - Bayesian Modelling

Advanced Statistical Modelling

October 23, 2023

Ximena Moure

1 Data Load

First I started by loading the data and plotting it to have a first look at it. I decided to remove the observation with NA values.

```r
library(R2jags)
# Data
x <- c(1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5)
y <- c(25, 31, 27, 28, 36, 35, NA, 34)

# Removing NA
not_na <- !is.na(y)
x <- x[not_na]
y <- y[not_na]
data_list <- list(x=x, y=y, n=length(y))

plot(x, y, pch=19, xlab="Fertilizer level", ylab="Yield")
```

Listing 1: Data load and plot

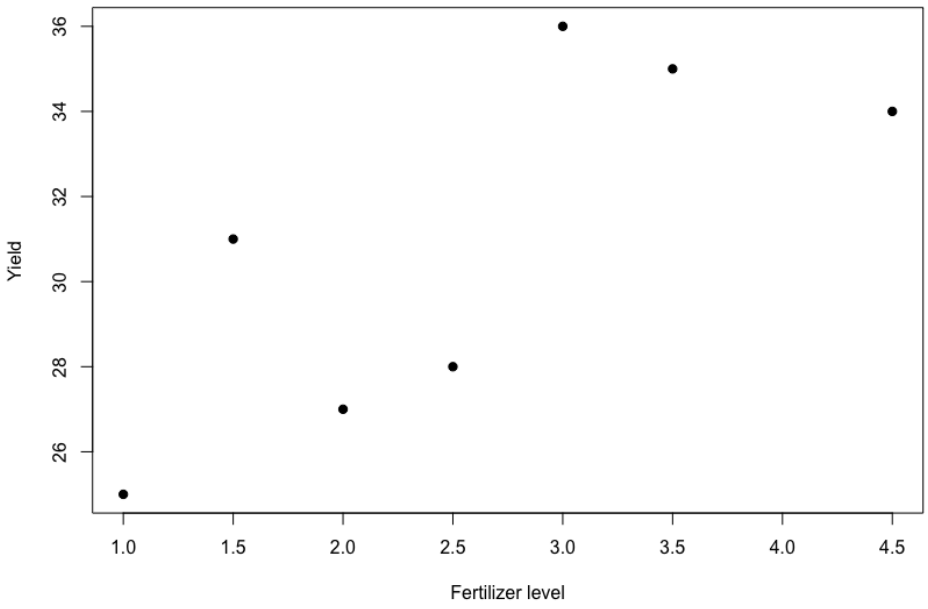

Figure 1: Data visualization

2 Model build

I use JAGS to build the model. The following is a description of it:

- $y[i]$ is the observed data point for the i-th observation.
- The model assumes that each observation $y[i]$ follows a Normal distribution (`dnorm`) with a mean of $b_0 + b_1 \cdot x[i]$ and a precision of τ.
- b_0 and b_1 are the intercept and slope of the linear regression, respectively.
- $x[i]$ is the i-th value of the predictor variable (in this case, the level of fertilizer).
- τ is the precision, which is the reciprocal of the variance ($\tau = \frac{1}{\sigma^2}$).
- b_0 and b_1 are given normal priors with mean 0 and a very small variance (1.0×10^{-7}). This essentially means that before observing the data, the model has a vague or non-informative belief about the values of b_0 and b_1.
- σ is given a uniform prior between 0 and 100000. This is a very wide range, making this prior quite uninformative as well. It represents a lack of strong prior knowledge about the value of the standard deviation.

The complete code for building the model is depicted in Listing 2

```
regression <- "
model {
 for (i in 1:n) {
   y[i]~ dnorm(b0+b1*x[i], tau)
 }
 b0 ~ dnorm(0, 1.0E-7)
 b1 ~ dnorm(0, 1.0E-7)
 sigma ~ dunif(0, 100000)
 tau <- pow(sigma, -2)
}
"

### discarded "Burn", save "Iter", chains "Chain"

Iter <- 10000
Burn <- 1000
Chain <- 2
Thin <- 1

data <- list(y=y, x=x, n=n)

parameters <- c("b0", "b1", "sigma")

potatoes.model  <-jags(data, inits=NULL, parameters.to.save=parameters,
                    model=textConnection(regression),
```

```
                    n.iter=(Iter*Thin+Burn), n.burnin=Burn, n.thin=Thin, n.chains=Chain
                        )

traceplot(potatoes.model, mfrow = c(length(parameters),1), varname = parameters)

print(potatoes.model)

attach.jags(potatoes.model)
b0 <- b0
b1 <- b1
sigma <- sigma
detach.jags()
```

Listing 2: Model

After running the above code we get the model in figure 2 and the plots depicted in figure 3. From the plots we can see that the process converges.

```
Inference for Bugs model at "3", fit using jags,
 2 chains, each with 11000 iterations (first 1000 discarded)
 n.sims = 20000 iterations saved
          mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat n.eff
b0         23.912   4.747 14.696 21.451 23.878 26.379 33.265 1.003 20000
b1          2.700   1.709 -0.660  1.829  2.699  3.582  6.018 1.003 20000
sigma       4.372   2.336  2.054  2.975  3.758  5.030 10.171 1.004   910
deviance   38.146   4.091 33.548 35.157 37.069 39.996 48.756 1.005   800

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 8.4 and DIC = 46.5
DIC is an estimate of expected predictive error (lower deviance is better).
```

Figure 2: Model result

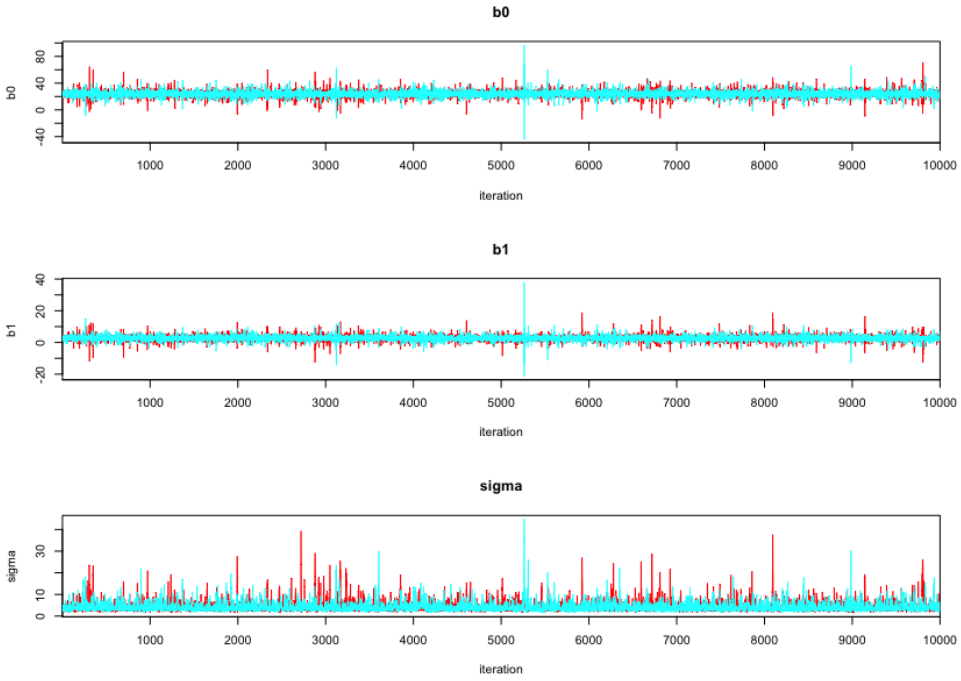

Figure 3: Model plots

3 Draw the posterior distribution for all of them

Now, we can plot the distributions. See Figure 4

```
par(mfrow=c(2,2))

plot(density(b0, adjust = 1.5), main = expression(paste(pi,"(",beta[0],"|y)")), xlab= "
    " )
plot(density(b1, adjust = 1.5), main = expression(paste(pi,"(",beta[1],"|y)")), xlab= "
    "  )
plot(density(sigma, adjust = 1.5), main = expression(paste(pi,"(",sigma,"|y)")), xlab=
    "" )
```

Listing 3: Posterior distribution

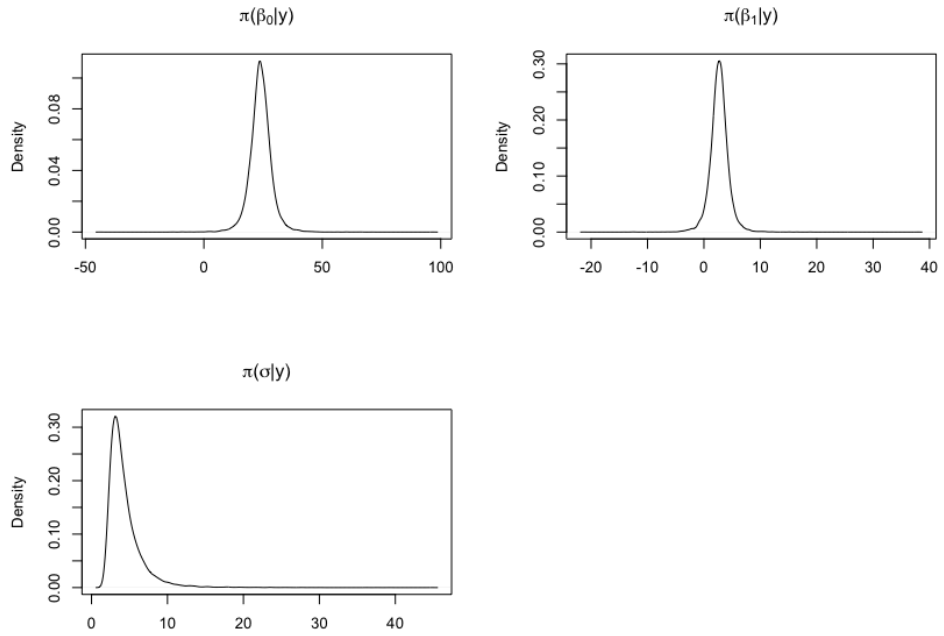

Figure 4: Posterior Distribution

4 Calculate a 95% credible interval for each parameter.

Parameter	Lower Bound	Upper Bound
β_0	14.696	33.265
β_1	-0.661	6.018
σ	2.055	10.171

Table 1: 95% Credible Intervals for each parameter

```r
# Calculate the 95% credible intervals for b0, b1, and sigma
CI_b0 <- quantile(b0, c(0.025, 0.975))
CI_b1 <- quantile(b1, c(0.025, 0.975))
CI_sigma <- quantile(sigma, c(0.025, 0.975))

# Print the 95% credible intervals
print(CI_b0)
print(CI_b1)
print(CI_sigma)
```

```
> # Print the 95% credible intervals
> print(CI_b0)
    2.5%    97.5%
14.69641 33.26484
> print(CI_b1)
      2.5%      97.5%
-0.6604218  6.0175296
> print(CI_sigma)
    2.5%    97.5%
 2.05405 10.17069
```

Figure 5: Results credible interval

5 Calculate a 95% credible interval for y given x = 4.

To compute a 95% credible interval for y given $x = 4$, we need to generate posterior predictive samples for y at $x = 4$ using the posterior samples of β_0 and β_1.

Percentile	Lower Bound	Upper Bound
2.5%	22.929	
97.5%		46.120

Table 2: 95% Credible Interval for y given $x = 4$

```r
# prediction

M <- length(b0)
y.x4 <- rnorm(M, b0+b1*4, sigma)

par(mfrow=c(2,1))
plot(density(y.x4))


quantile(y.x4, c(0.025,0.975))

summary(y.x4)
```