Homework 1

Gaurav Kandlikar

October 9, 2016

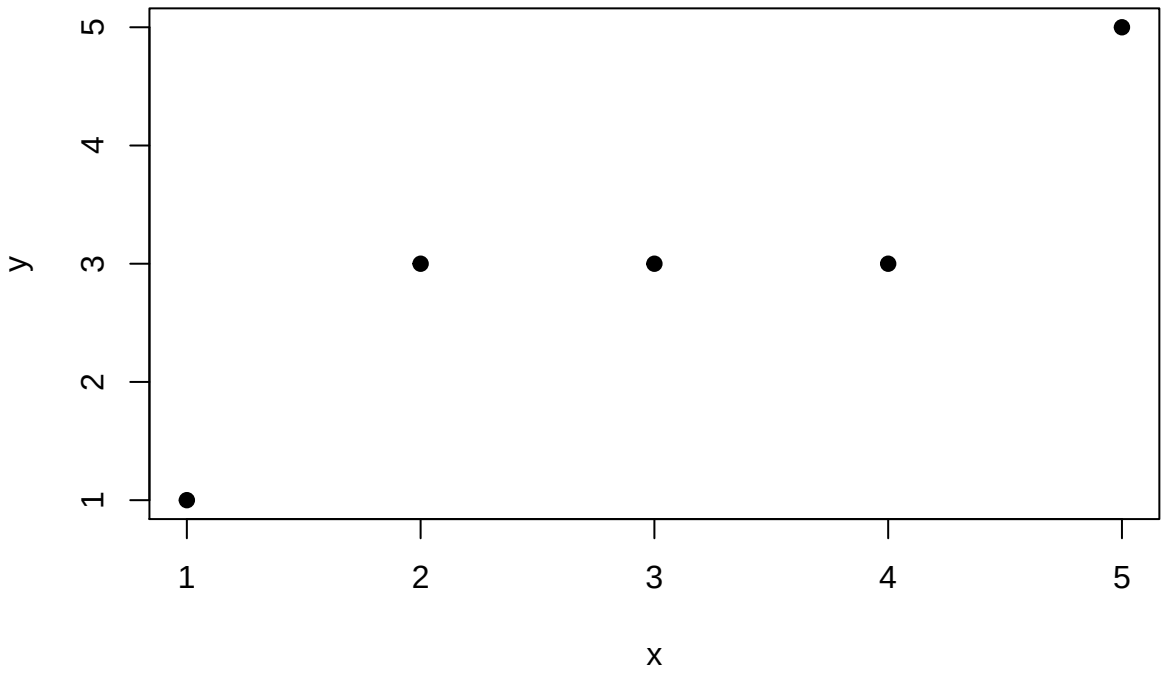
```
library(R2jags)
setwd("~/grad/courses/UCLA/biost234/homework/lab1")

## Create jags model file. This uses the bugs language
## to specify the prior times likelihood.
## Our model is simple linear regression
## This is virtually the same model in the jags manual section 2.1.
## The prior for beta is different here.
## Our prior for tau is "superior".
## What exactly does it mean that our prior is "superior?"

# sink("model1.txt")
cat("
    model
    {
    for(i in 1:N){
        y[i] ~ dnorm(mu[i],tau)
        mu[i] <- alpha + beta * (x[i]-x.bar)
        }
        alpha ~ dnorm(0, 0.0001)
        beta ~ dnorm(1,1)
        tau ~ dgamma(.25,.25)
    sigma <- 1/sqrt(tau)
    }
    ",fill = TRUE)
```

```
##
##     model
##  {
##  for(i in 1:N){
##      y[i] ~ dnorm(mu[i],tau)
##      mu[i] <- alpha + beta * (x[i]-x.bar)
##      }
##      alpha ~ dnorm(0, 0.0001)
##      beta ~ dnorm(1,1)
##      tau ~ dgamma(.25,.25)
##     sigma <- 1/sqrt(tau)
##  }
##
```

```
# sink()
```

```
# Set up dataset
## input data
## We specify a simple linear regression data set with 5 observations
## Here is the data:
```

```r
x     = c(1,2,3,4,5)
y     = c(1,3,3,3,5)
# Take a peek, because I'm bad at visualizing!
plot(x,y, pch = 19)
```




```r
N     = length(x)
x.bar = mean(x)
# Set up the things that we feed into the jags model
jags.data = list("x","y","N","x.bar")
```

JAGS parameters

```r
## Input parameter names
## These are the parameters whose posterior
## distribution(s) we are interested in.
## If we aren't interested in a particular parameter
## and don't list it, JAGS will
## not keep track of information about that parameter,
## though it is still part of the posterior.
## Here we list all parameters, including both tau and sigma, though
## they are functions of each other.

jags.params = c("alpha", "beta", "tau", "sigma") # things that we want JAGS to report
```

```
jags.inits = function(){
    list("alpha" = 0, "beta" = 1, "tau" = 1) # part of algorithm!
}
```

TO DO: identify where each parameter is used in the model.

In the model, α is the estimate for the intercept. In this case, because we have centered the covariate x, α will be an estimate of the mean of y. β will estimate the slope of the linear regression model. τ is the sampling precision, or the error term in the linear regression model. σ is simply a transformation of τ ($\sigma = 1/\sqrt{\tau}$), and is reported from the model for convenience- we could also calculate this in R outside the model if desired.

Run the simulation

```
lab1.sim = jags(jags.data, jags.inits, jags.params,
              model.file = "model1.txt",
              n.chains = 3, n.iter = 11000, n.burnin = 1000)
```

```
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 3
##    Total graph size: 40
##
## Initializing model
```

Question 1

What information is in the chapter 5 tables of the jags user manual?

Generally, the tables in chapter 5 list the functions that can be called inside jags models:

Table 5.1 lists the JAGS base functions- this base module contains some workhorse functions needed to run models. Table 5.2 lists scalar functions that are loaded with the bugs module- these functions take vectors and returns vectors of the same length.
Table 5.3 lists JAGS functions from the bugs module to calculate probability densities, probability functions, and quantliles of many distributions.
Table 5.4 lists the available link functions
Table 5.5 shows the bugs functions that return scalars
Table 5.6 shows the bugs functions that work on and return vectors/matrices
Table 5.7 lists functions that can be aliased to be R-compatible

Question 2

What information is in the distributions chapter of the jags user manual? Recite briefly the tables in this chapter.
Chapter 6 lists the distributions availble for us to use in JAGS. We can set parameters as following the d[xxx] distributions using the ~ notation.
Table 6.1 shows the univariate, real-valued distributions, along with the density equation, parameters, and range.
Table 6.2 shows discrete univariate distributions, and 6.3 shows multivariate distributions, both along with the ranges and density functions.
Table 6.4 shows BUGS aliases that we can are R-compatible (these are cases where the canonical bugs functions had slightly different names than the corresponding density functions in R, but we can now use the R names thanks to aliasing.)

Question 3

See the WinBUGS examples in the manual. What models (plural) does the Stacks example use? Specify each model. (There are 6 models).

In the Stacks example, six models are run with distinct error structures:

- The sampling of Y is normally distributed with mean calculated according to a multiple linear regression and τ being gamma distributed with shape and scale = 1≏3.
- Sampling of Y with error following a double exponential distribution
- Sampling of Y with error following a t-distributed error with df = 4.

They also ran the three error structures with a ridge-regression approach, which further shrinks parameter estimates to correct for multicolinearity in predictors.

- Ridge-regression with normally distributed error
- Ridge-regression with double-exponential distributed error
- Ridge-regression with t-distributed error

Question 4

a) **Turn in properly formatted output from your regression model.**

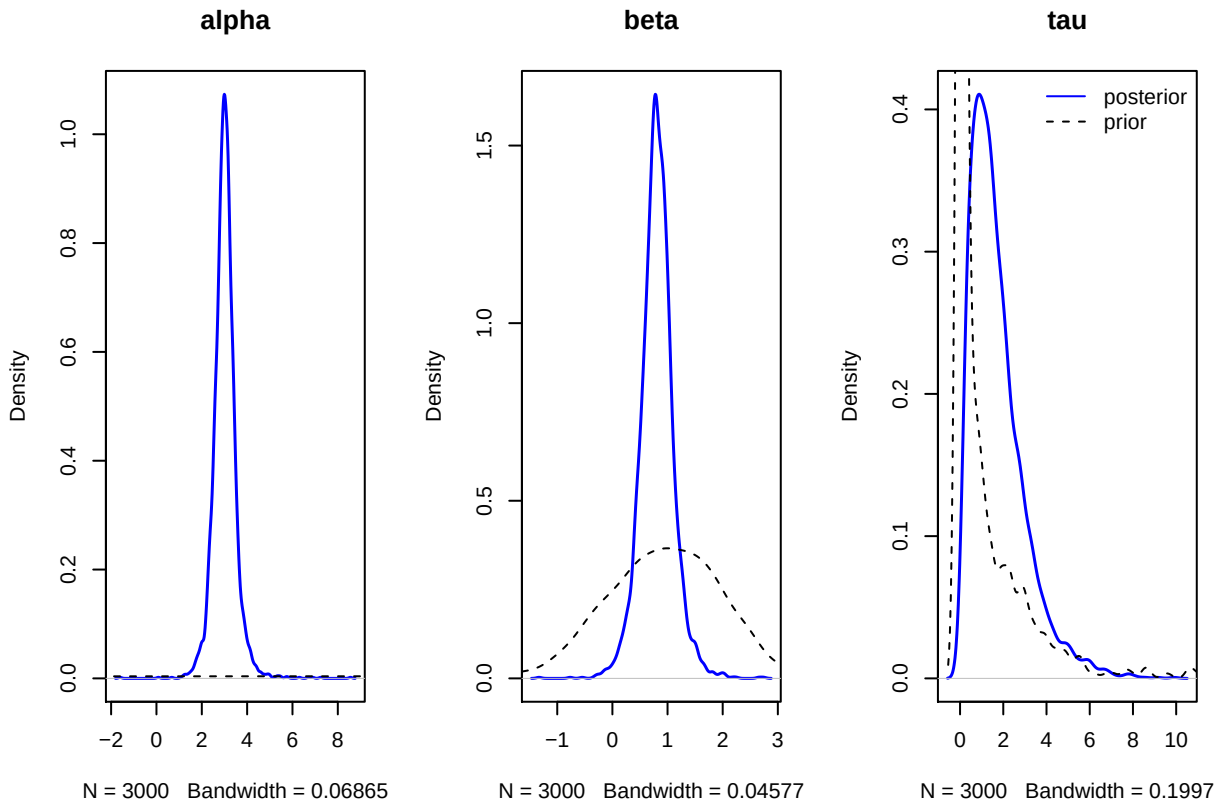
Recall that the initial linear regression model was saved in the R object `lab1.sim`. We retrieve relevant information from this object:

```
knitr::kable(lab1.sim$BUGSoutput$summary[,c("mean","sd","2.5%","97.5%")])
```

	mean	sd	2.5%	97.5%
alpha	3.013	0.512	2.038	4.04
beta	0.810	0.301	0.213	1.45
deviance	12.537	3.120	8.842	20.52
sigma	0.966	0.512	0.445	2.16
tau	1.722	1.270	0.215	5.06

Some plots of posteriors and priors

```
temp1 <- apply(lab1.sim$BUGSoutput$sims.array, 3, unlist)
alphas <- temp1[,"alpha"]
betas <- temp1[,"beta"]
tau <- temp1[,"tau"]

par(mfrow = c(1,3))
plot(density(alphas), main = "alpha", lwd = 1.5, col = "blue")
lines(density(rnorm(1000, 0, 1/sqrt(.0001))), lty = 2)
plot(density(betas), main = "beta", lwd = 1.5, col = "blue")
lines(density(rnorm(1000, 1, 1/sqrt(1))), lty = 2)
plot(density(tau), main = "tau", lwd = 1.5, col = "blue")
lines(density(rgamma(1000, 0.25, 0.25)), lty = 2)
legend("topright", bty = "n", lty = c(1,2), col = c("blue", "black"),
       legend = c("posterior", "prior"))
```



5. Change the prior precision for beta to 100, 10, 1, .1, .01, .001. The prior precision is the number 100 in the statement beta ~ dnorm(1,100) in your model program. Run the model for each of these values. What happens to the estimate of beta as the prior precision changes?

First, print out the six different models to the working directory:

```
# print out the seven models
# Uncomment these lines to print models out to files
# sink("model1_hw.txt")
# sink("model2_hw.txt")
# sink("model3_hw.txt")
# sink("model4_hw.txt")
# sink("model5_hw.txt")
# sink("model6_hw.txt")
# sink("model7_hw.txt")
cat("
    model
    {
    for(i in 1:N){
        y[i] ~ dnorm(mu[i],tau)
        mu[i] <- alpha + beta * (x[i]-x.bar)
        }
        alpha ~ dnorm(0, 0.0001)
    # uncomment one of these to make the model run:
        # beta ~ dnorm(1,1)
```

```
        # beta ~ dnorm(1,.001)
        # beta ~ dnorm(1,.01)
        # beta ~ dnorm(1,.1)
        # beta ~ dnorm(1,1)
        # beta ~ dnorm(1,10)
    # beta ~ dnorm(1,100)

        tau ~ dgamma(.25,.25)
    sigma <- 1/sqrt(tau)
    }
    ",fill = TRUE)
# sink()
```

Then, call the six models using the jags parameters, inits, and data from the previous example:

```
# Loop through the model files and run them
# The data, initial conditions, and paramaters to store
# have not changed since the first model run.
model_list <- list.files(pattern = "model\\d_hw.txt")
hw1.sims <- list()
for (ii in 1:length(model_list)) {
  current_model <- model_list[ii]
  hw1.sims[[ii]] = jags(jags.data, jags.inits, jags.params,
                  model.file = current_model,
                  n.chains = 3, n.iter = 11000, n.burnin = 1000)
}
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 3
##    Total graph size: 40
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 3
##    Total graph size: 40
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 3
```

```
##    Total graph size: 40
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 3
##    Total graph size: 40
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 3
##    Total graph size: 40
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 3
##    Total graph size: 40
##
## Initializing model
```

Question 5

a. Report a(n appropriately formatted) table of the posterior means and sds as a function of the prior precision.

```
beta_mean  <- unlist(lapply(hw1.sims, function(x) x$BUGSoutput$summary["beta", "mean"]))
beta_sd    <- unlist(lapply(hw1.sims, function(x) x$BUGSoutput$summary["beta", "sd"]))
beta_025   <- unlist(lapply(hw1.sims, function(x) x$BUGSoutput$summary["beta", "2.5%"]))
beta_975   <- unlist(lapply(hw1.sims, function(x) x$BUGSoutput$summary["beta", "97.5%"]))
to_print   <- cbind(beta_mean, beta_sd, beta_025, beta_975)
prior_tau <- c(0.001, 0.01, 0.1, 1, 10, 100)
to_print <- cbind(prior_tau, to_print)
knitr::kable(to_print)
```

prior_tau	beta_mean	beta_sd	beta_025	beta_975
1e-03	0.802	0.369	0.064	1.52
1e-02	0.793	0.381	0.100	1.56
1e-01	0.809	0.346	0.149	1.50
1e+00	0.819	0.312	0.193	1.44
1e+01	0.881	0.209	0.475	1.31
1e+02	0.972	0.094	0.789	1.16

b. As the prior precision goes to +infinity, what do you suppose the limit of the values of the estimate and sd are?
Our priors for β were that the parameter is normally distributed with mean 1 and τ ranging from 1^-3 to 10^2. Very low values of τ indicate that our prior distribution is quite "wide", whereas a high value of τ leads to a "narrow" prior centered around 1. It makes sense, therefore, that **the limit of the estimate of β as the prior τ approaches +infinity appears to be 1, and the limit of the sd appears to be 0**. This makes sense to me - by setting the prior with a precission approaching +infinity, we are essentially acting "telling" the analysis that we are very confident that the slope being estimated is 1 or very close to it.

c. The least squares estimate of beta is .8. What is the limit of the estimate as the prior precision goes to zero?
The limit of β as τ approaches 0 also appeares to be ~0.8. This too makes sense: when a prior has a very low precision (= high variance), the prior mean gets a low weight when determining the posterior. For the normal-normal model, the term $1/sd^2_{prior}$ appears in the numerator of the fraction determining the weight of the prior, so that the value of the weight tends to zero as sd_{prior} approaches +infinity.

Some plots I made for myself- not necessarily relevant to the questions

```
## Showing the posterior distributions, with priors overlayed
# Make a list of all parameter estimates
all_parameters <- lapply(hw1.sims, function(x) apply(x$BUGSoutput$sims.array, 3, unlist))
# Extract the betas
betas <- sapply(all_parameters, function(x) x[,"beta"])
colnames(betas) <- as.character(prior_tau)
priors <- (sapply(prior_tau, function(x) rnorm(1000, mean = 1, sd = sqrt(1/x))))
par(mfrow = c(2,3))
for (ii in 1:6) {
  plot(density(betas[, ii]), col = "blue", xlim = c(-2, 4), ylim = c(0, 4.5),
       main = paste("Tau =", prior_tau[ii]))
  lines(density(priors[,ii]), lty = 2)
  legend("topleft", col = c("blue", "black"), lty = c(1,2),
         legend = c("posterior", "prior"), bty = "n")
}
```