Lecture 6

Introduction of Bayesian analysis using JAGS

WILD6900 (Spring 2021)

Readings

> Kéry & Schaub 38-40; 58-64

From custom MCMC to BUGS

The BUGS language

Bayesian **A**nalysis **U**sing **G**ibbs **S**ampling

Language/program invented in the 1990's by epidemiologists in Cambridge

Later became WinBUGS

- First customizable MCMC software
- Revolutionized the use of Bayesian methods in applied statistics

The BUGS language

Since the development of WinBUGS, other Bayesian software programs have been developed:

- OpenBugs
- JAGS
- Nimble
- Stan

For the remainder of the course, we will fit models using JAGS

- **J**ust **A**nother **G**ibbs **S**ampler
- Uses BUGS language (easy to learn, lots of online documentation)
- Often out-performs WinBUGS
- Available for all operating systems

The BUGS language

Last week, we learned how to:

- Determine the *full conditional distributions* for a linear regression model
- Write a custom MCMC sampler to produce samples from the joint posterior distribution

Given a statistical model and user-specified prior distributions, JAGS does these steps for you!

- Possible to fit arbitrarily complex models [1]
- "Frees the modeler in you"

Running JAGS from R

JAGS is a stand alone software program

- Can be run from GUI or command line

JAGS can also be run from R using the `jagsUI` package (among others)

- Write model in R script and save as `.jags` file
- Provide `jagsUI` with data, initial values, and MCMC settings
- model run in JAGS
- model output brought back in to R for diagnostics/analysis/visualization

The BUGS language

Very similar to R (but more limited)

- Limited ability to vectorize operations

If you can write your model on the whiteboard, you can write it in JAGS

- Stochastic relationships represented by ~
- Deterministic relationships represented by <-

The BUGS language

Linear regression model

$$y_i = \alpha + \beta \times x_i + \epsilon_i$$

$$\epsilon_i \sim Normal(0, \tau)$$

The BUGS language

Linear regression model

$$y_i = \underbrace{\alpha + \beta \times x_i}_{Deterministic} + \underbrace{\epsilon_i}_{Stochastic}$$

The BUGS language

Linear regression model

$$\underbrace{\mu_i = \alpha + \beta \times x_i}_{Deterministic}$$

$$\underbrace{y_i \sim Normal(\mu_i, \tau)}_{Stochastic}$$

Remember that these equations define the *likelihood* of our data given values of α, β, and τ

The BUGS language

Linear regression model

To specify a fully Bayesian model, we also need to define the priors:

$$[\alpha] \sim Normal(\alpha|0, 0.001)$$

$$[\beta] \sim Normal(\alpha|0, 0.001)$$

$$[\tau] \sim Gamma(\tau|0.01, 0.01)$$

The BUGS language

Linear regression model

```
model{
  ## Priors
    alpha ~ dnorm(0, 0.001)
    beta ~ dnorm(0, 0.001)
    tau ~ dgamma(.001,.001) # Precision
    sigma <- 1/sqrt(tau)      # Calculate sd from precision

  ## Likelihood
    for(i in 1:N){
      mu[i] <- alpha + beta * x[i]
      y[i] ~ dnorm(mu[i], tau)
    }
} #end of model
```

Writing model files

```
sink(file="jags/linear_regression.jags")
cat("
  model{
    ## Priors
    alpha ~ dnorm(0, 0.001)
    beta ~ dnorm(0, 0.001)
    tau ~ dgamma(.001,.001) # Precision
    sigma <- 1/sqrt(tau)      # Calculate sd from precision
    ## Likelihood
    for(i in 1:N){
      mu[i] <- alpha + beta * x[i]
      y[i] ~ dnorm(mu[i], tau)
    }
  } #end of model
    ", fill=TRUE)
sink()
```

Preparing the data

```
## Read simulated data frame
dat <- readRDS("data/sim_seed_counts.rds")

## Store data for JAGS as list
jags_data <- list(y = dat$y, x = dat$visits.c, N = nrow(dat))

## Create function that returns random initial values
jags_inits <- function(){list(alpha = runif(1, 200, 300),
                              beta = runif(1, 25, 75),
                              tau = runif(1))}

## Tell JAGS which parameters we want to monitor
params <- c("alpha", "beta", "tau", "sigma")
```

Run the model

```
## Run the model
jags_fit <- jagsUI::jags(data = jags_data, inits = jags_inits,
                         parameters.to.save = params,
                         model.file = "jags/linear_regression.jags",
                         n.chains = 3, n.iter = 10000,
                         n.burnin = 2500, n.thin = 1)
```

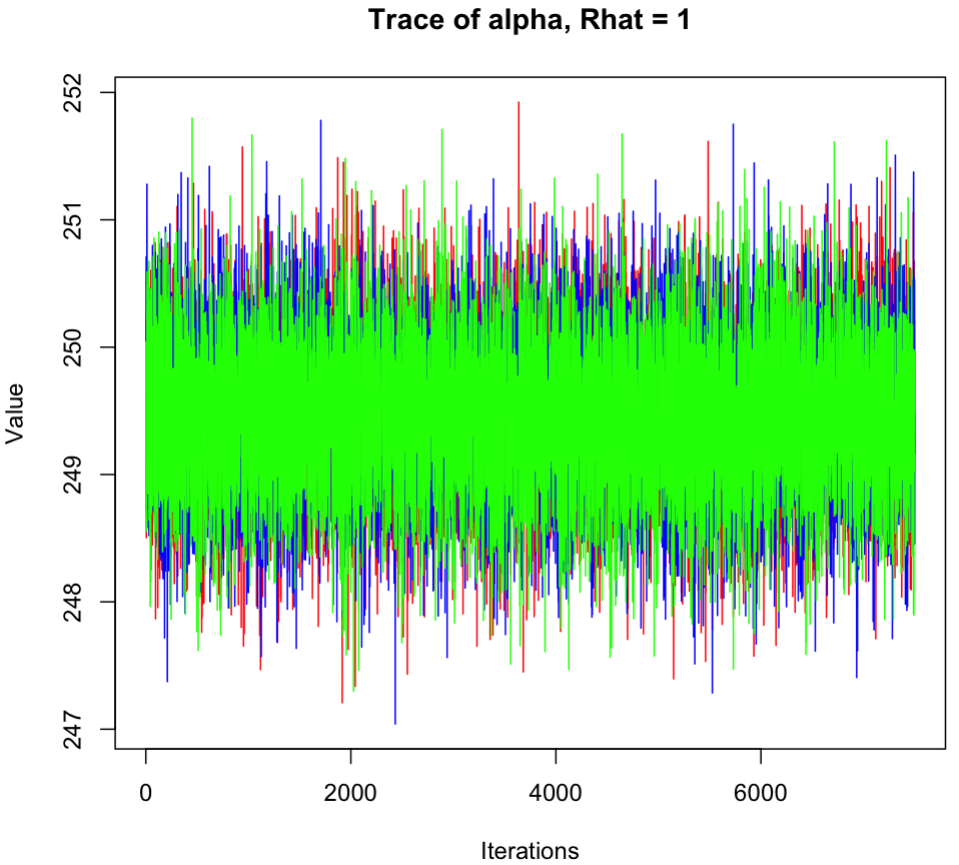
Diagnostics

```
print(jags_fit)
```

```
## JAGS output for model '/Library/Frameworks/R.framework/Versions/4.0/Resour
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 100 iterations (sufficient),
## burn-in = 2500 iterations and thin rate = 1,
## yielding 22500 total samples from the joint posterior.
## MCMC ran for 0.032 minutes at time 2021-01-13 09:41:26.
##
##               mean    sd     2.5%      50%    97.5% overlap0 f Rhat n.eff
## alpha      249.462 0.600  248.296  249.462  250.645    FALSE 1    1  9616
## beta        49.656 0.590   48.507   49.652   50.796    FALSE 1    1 22500
## tau          0.016 0.002    0.013    0.016    0.020    FALSE 1    1 22500
## sigma        7.884 0.428    7.098    7.866    8.783    FALSE 1    1 22500
## deviance  1218.809 2.454 1215.986 1218.177 1225.131    FALSE 1    1 22500
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size,
```

Diagnostics

```
jagsUI::traceplot(jags_fit)
```



Structure of the JAGS output

```
class(jags_fit)
```

```
## [1] "jagsUI"
```

```
names(jags_fit)
```

```
##  [1] "sims.list"   "mean"        "sd"          "q2.5"        "q25"
##  [6] "q50"         "q75"         "q97.5"       "overlap0"    "f"
## [11] "Rhat"        "n.eff"       "pD"          "DIC"         "summary"
## [16] "samples"     "modfile"     "model"       "parameters"  "mcmc.info"
## [21] "run.date"    "parallel"    "bugs.format" "calc.DIC"
```

Structure of the JAGS output

```
str(jags_fit$sims.list)
```

```
## List of 5
##  $ alpha   : num [1:22500] 249 249 250 249 249 ...
##  $ beta    : num [1:22500] 49.2 50.2 49.2 50.1 49.9 ...
##  $ tau     : num [1:22500] 0.0182 0.0181 0.0172 0.0187 0.0194 ...
##  $ sigma   : num [1:22500] 7.41 7.43 7.62 7.3 7.18 ...
##  $ deviance: num [1:22500] 1220 1219 1218 1218 1222 ...
```

```
head(jags_fit$sims.list$alpha)
```

```
## [1] 248.7 248.9 250.2 249.2 248.5 249.6
```

Structure of the JAGS output

```
jags_fit$mean$alpha
```

```
## [1] 249.5
```

```
jags_fit$f$alpha
```

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

Structure of the JAGS output

```
jags_fit$summary
```

```
##                  mean       sd      2.5%       25%       50%       75%     97
## alpha    2.495e+02 0.600132 2.483e+02 2.491e+02 2.495e+02 2.499e+02 2.506e
## beta     4.966e+01 0.589656 4.851e+01 4.926e+01 4.965e+01 5.006e+01 5.080e
## tau      1.623e-02 0.001753 1.296e-02 1.501e-02 1.616e-02 1.737e-02 1.985e
## sigma    7.884e+00 0.428339 7.098e+00 7.588e+00 7.866e+00 8.162e+00 8.783e
## deviance 1.219e+03 2.453575 1.216e+03 1.217e+03 1.218e+03 1.220e+03 1.225e
##               Rhat n.eff overlap0 f
## alpha    1.0003  9616        0 1
## beta     1.0000 22500        0 1
## tau      1.0000 22500        0 1
## sigma    0.9999 22500        0 1
## deviance 1.0000 22500        0 1
```

Structure of the JAGS output

```
str(jags_fit$samples)
```

```
## List of 3
##  $ : 'mcmc' num [1:7500, 1:5] 249 249 250 249 249 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:5] "alpha" "beta" "tau" "sigma" ...
##   ..- attr(*, "mcpar")= num [1:3] 2501 10000 1
##  $ : 'mcmc' num [1:7500, 1:5] 250 250 249 251 251 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:5] "alpha" "beta" "tau" "sigma" ...
##   ..- attr(*, "mcpar")= num [1:3] 2501 10000 1
##  $ : 'mcmc' num [1:7500, 1:5] 250 249 249 249 249 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:5] "alpha" "beta" "tau" "sigma" ...
##   ..- attr(*, "mcpar")= num [1:3] 2501 10000 1
##  - attr(*, "class")= chr "mcmc.list"
```

Structure of the JAGS output

```
str(jags_fit$mcmc.info)
```

```
## List of 9
##  $ n.chains        : num 3
##  $ n.adapt         : num 100
##  $ sufficient.adapt: logi TRUE
##  $ n.iter          : num 10000
##  $ n.burnin        : num 2500
##  $ n.thin          : num 1
##  $ n.samples       : num 22500
##  $ end.values      :List of 3
##   ..$ : Named num [1:5] 2.49e+02 5.02e+01 1.22e+03 7.70 1.69e-02
##   .. ..- attr(*, "names")= chr [1:5] "alpha" "beta" "deviance" "sigma" ...
##   ..$ : Named num [1:5] 2.48e+02 4.98e+01 1.22e+03 7.84 1.63e-02
##   .. ..- attr(*, "names")= chr [1:5] "alpha" "beta" "deviance" "sigma" ...
##   ..$ : Named num [1:5] 2.50e+02 4.93e+01 1.22e+03 8.09 1.53e-02
##   .. ..- attr(*, "names")= chr [1:5] "alpha" "beta" "deviance" "sigma" ...
##  $ elapsed.mins    : num 0.032
```

Saving model output

```
saveRDS(jags_fit, "output/regression_out.rds")
```