Lecture 6

Introduction of Bayesian analysis using JAGS

WILD6900 (Spring 2020)

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

- Stochasitic 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 specificy 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

```r
## 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

```r
## 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)
```

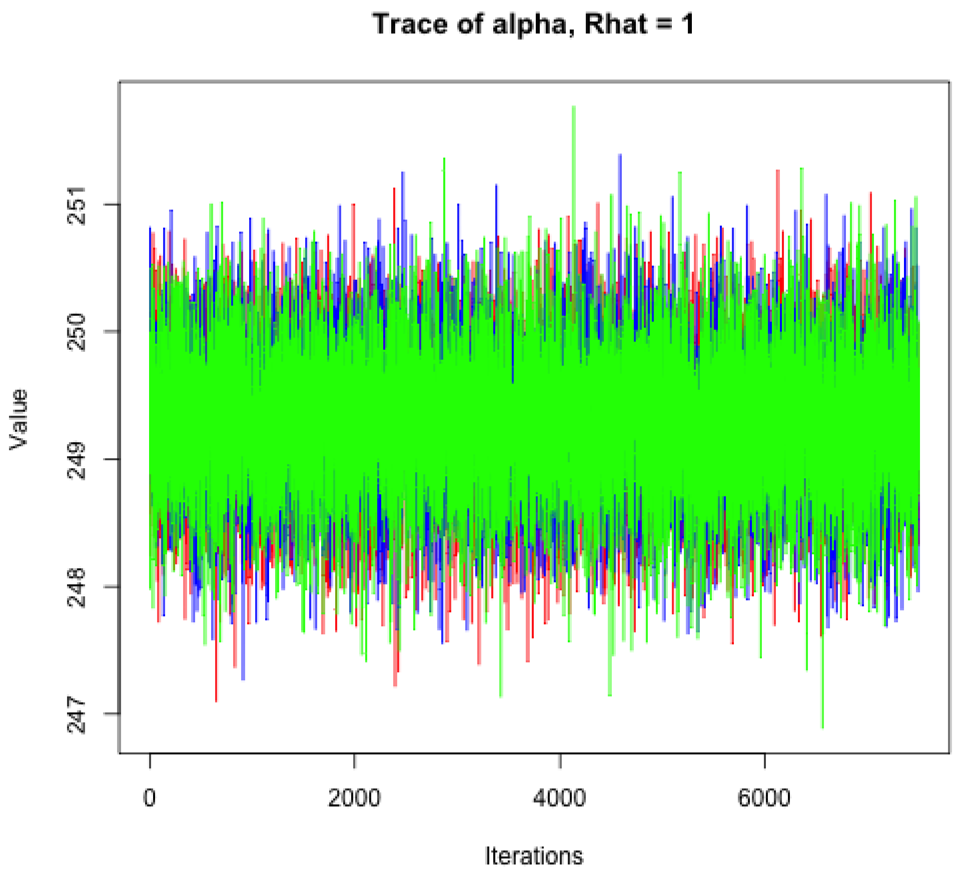
Diagnostics

```
print(jags_fit)
```

```
## JAGS output for model '/Library/Frameworks/R.framework/Versions/3.6/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 0.026 minutes at time 2020-01-04 20:36:37.
##
##              mean    sd     2.5%      50%    97.5% overlap0 f  Rhat n.eff
## alpha     249.300 0.551  248.198  249.302  250.367    FALSE 1 1.000 22500
## beta       50.104 0.558   48.999   50.104   51.184    FALSE 1 1.000 22500
## tau         0.019 0.002    0.015    0.019    0.023    FALSE 1 1.000 22500
## sigma       7.321 0.395    6.594    7.305    8.143    FALSE 1 1.000 22500
## deviance 1192.708 2.505 1189.884 1192.051 1199.172    FALSE 1 1.001  8610
##
## 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] 249 249 249 250 249 ...
##  $ beta    : num [1:22500] 50.4 49.3 50.9 51.2 50.7 ...
##  $ tau     : num [1:22500] 0.0218 0.0216 0.0206 0.0193 0.0208 ...
##  $ sigma   : num [1:22500] 6.77 6.8 6.97 7.2 6.94 ...
##  $ deviance: num [1:22500] 1192 1194 1192 1195 1191 ...
```

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

```
## [1] 249.4 249.5 249.2 250.1 249.3 248.2
```

Structure of the JAGS output

```
jags_fit$mean$alpha
```

```
## [1] 249.3
```

```
jags_fit$f$alpha
```

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

Structure of the JAGS output

```
jags_fit$summary
```

```
##               mean       sd      2.5%       25%       50%       75%
## alpha    2.493e+02 0.551199 2.482e+02 2.489e+02 2.493e+02 2.497e+02
## beta     5.010e+01 0.557579 4.900e+01 4.973e+01 5.010e+01 5.048e+01
## tau      1.882e-02 0.002019 1.508e-02 1.744e-02 1.874e-02 2.013e-02
## sigma    7.321e+00 0.394910 6.594e+00 7.048e+00 7.305e+00 7.573e+00
## deviance 1.193e+03 2.505359 1.190e+03 1.191e+03 1.192e+03 1.194e+03
##             97.5%  Rhat n.eff overlap0 f
## alpha     250.367 1.000 22500        0 1
## beta       51.184 1.000 22500        0 1
## tau         0.023 1.000 22500        0 1
## sigma       8.143 1.000 22500        0 1
## deviance 1199.172 1.001  8610        0 1
```

Structure of the JAGS output

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

```
## List of 3
##  $ : 'mcmc' num [1:7500, 1:5] 249 249 249 250 249 ...
##   ..- 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] 249 251 249 249 249 ...
##   ..- 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] 249 248 250 250 249 ...
##   ..- 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 5.04e+01 1.19e+03 7.85 1.62e-02
##   .. ..- attr(*, "names")= chr [1:5] "alpha" "beta" "deviance" "sigma" ...
##   ..$ : Named num [1:5] 2.50e+02 5.07e+01 1.19e+03 7.62 1.72e-02
##   .. ..- attr(*, "names")= chr [1:5] "alpha" "beta" "deviance" "sigma" ...
##   ..$ : Named num [1:5] 2.49e+02 5.10e+01 1.19e+03 7.31 1.87e-02
##   .. ..- attr(*, "names")= chr [1:5] "alpha" "beta" "deviance" "sigma" ...
##  $ elapsed.mins     : num 0.026
```

Saving model output

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