Just Another Gibbs Sampler: JAGS

Inglis, A., Ahmed, A., Wundervald, B. and Prado, E.
PhD students at Hamilton Institute

14th November of 2018

Agenda

Introduction

- JAGS is a program to perform inference for Bayesian Hierarchical models. It was proposed by Martyn Plummer in 2003 as an alternative to the BUGS software;
- The Gibbs Sampler function of JAGS is the ARMS, which is flexible for dealing with univariate target densities;
- The main advantages of JAGS, compared to BUGS, is the programming language and its interfaces with other software, such as R and Python;
- Also, JAGS does not require the specification of the conditional distributions.

Introduction

- Although JAGS is a really useful software to learn, there are few good examples online that explain both theory and code;
- The goals of this project were:
  - Learn how to use JAGS to perform Bayesian modelling and;
  - Write new codes;
- Basically, JAGS requires only the sampling distribution and the prior distribution for each parameter.

Introduction

```
model {
# Likelihood
    for (i in 1:n) {
        y[i] ~ dnorm(mu[i], precision)
        mu[i] <- alpha + beta * x
    }
# Prior distributions
    alpha ~ dnorm(0.0, 1.0E-3)
    beta ~ dnorm(0.0, 1.0E-3)
    aux <- dgamma(0.001, 0.001)
    precision ~ 1.0/ aux
}
```

R scripts

- Our main contributions were to add **mathematical details** and provide **real datasets examples** for **5** R scripts;
- The models involved were:
  - Random effect model;
  - Multivariate Normal model;
  - Beta regression;
  - Time series Beta Auto-Regressive model of order 1 and;
  - Mixture model.

R script - Beta regression

Let $\{Y_i\}_{i=1}^{n}$ be independent and identically distributed random variables and $X_i = (1, x_{i,1}, ..., x_{i,1})$ a line vector with all covariates of the individual i. We assume that Y_i is distributed according to a Beta distribution, denoted by $Y_i \sim \text{Beta}(\mu, \phi)$, which may be written in the form

$$f(Y_i|\mu, \phi) = \frac{\Gamma(\phi)\Gamma(\mu\phi)}{\Gamma((1-\mu)\phi)} Y_i^{\mu\phi-1}(1-Y_i)^{(1-\mu)/\phi},$$

where $0 < Y_i < 1$, $\mathbb{E}(Y_i) = \mu$, $\mathbb{V}(Y_i) = \mu(1-\mu)/(1+\phi)$, $0 < \mu < 1$ and $\phi > 0$. Thus, it is possible to model $g(\mu) = X_i\boldsymbol{\beta}$, where $g(\cdot)$ is the link function that maps the unit interval into $\mathbb{R}$.

This parametrization was proposed by Ferrari and Cribari-Neto (2004).

R script - Beta regression

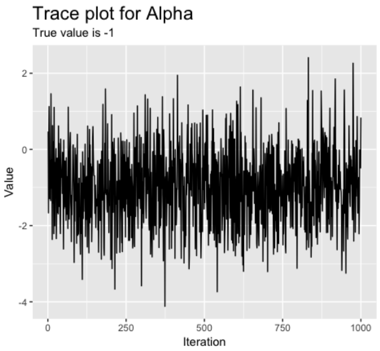
```
model
{
  # Likelihood
  for (t in 1:T) {
  y[t] ~ dbeta(a[t], b[t])
  a[t] <- mu[t] * phi
  b[t] <- (1 - mu[t]) * phi
  logit(mu[t]) <- alpha + beta * x[t]
  }
  # Priors
  alpha ~ dnorm(0, 10^-2)
  beta ~ dnorm(0, 10^-2)
  phi ~ dunif(0, 10)
}
```

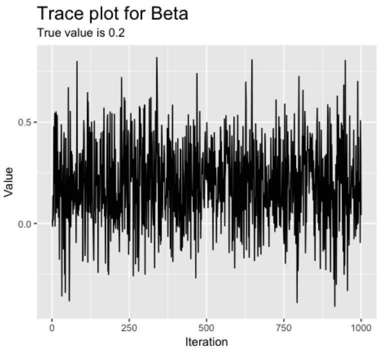
R script - Beta regression

```r
library(datasets)
head(attenu)

#Set up the data
acc=with(attenu,list(y=attenu$accel
                     ,T=nrow(attenu)))

# Set up jags model
jags_model=jags(acc,
                parameters.to.save = model_parameters,
                model.file = textConnection(model_code),
                n.chains=4,
                n.iter=1000,
                n.burnin=200,
                n.thin=2)
# Plot the jags output
print(jags_model)
```

R script - Beta regression

Simulation results for Beta regression, where three parameters were consider. The true values were set in $\alpha = -1$, $\beta = 0.2$ and $\phi = 5$.





Python scripts

- 3 Python scripts were created providing **mathematical details** and **simulated and real datasets examples**;
- The models involved were:
  - Bayesian linear regression;
  - Logistic regression;
  - Beta regression.

Python script - Logistic regression

The logistic regression model assumes that a sequence of independent and identically distributed random variables $\{Y_i\}_1^n$ has a Binomial distribution, denoted by $Y_i \sim \text{Binomial}(p_i)$, in the form of

$$f(Y_i|p_i) = \binom{n}{Y_i} p_i^{Y_i}(1-p_i)^{1-Y_i},$$

where $Y_i \in \{0,1\}$, $\log(\frac{p_i}{1-p_i}) = X_i\beta$, $X_i = (1, x_{i,1}, ..., x_{i,1})$ is the line vector of covariates associated to the individual i and β is the vector of unknown parameters.

Python script - Logistic regression

```
model
{
  # Likelihood
  for (t in 1:n) {
  y[t] ~ dbin(p[t], 1)
  logit(p[t]) <- beta_0 + beta_1 * x_1[t] +
                 beta_2 * x_2[t]
  }

  # Priors
  beta_0 ~ dnorm(0.0,0.01)
  beta_1 ~ dnorm(0.0,0.01)
  beta_2 ~ dnorm(0.0,0.01)
}
```

Python script - Logistic regression

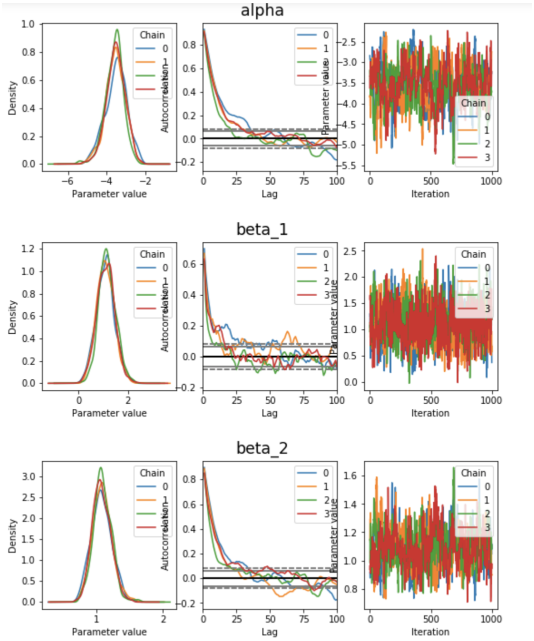
The data obtained for this section was adapted from data used to model logistic regression of moth mortalities when exposed to identical doses of a particular insecticide;

- Response variable: denotes the number of deaths observed in each batch;
- X_1 it is defined as the sexcode (i.e male or female);
- X_2 it is defined as the dose administered to each moth.

Python script - Logistic regression

```python
# Set up the data
model = pyjags.Model(code, data=dict(T = T, y = y,
                                     x_1 = x_1,
                                     x_2 = x_2, K = 1))

# Number of iterations to remove at start
model.sample(200, vars=[])

# Choose the parameters to watch and iterations:
samples = model.sample(1000, vars=['alpha', 'beta_1',
                                   'beta_2'])

print(jags_model)
```

Python script - Logistic regression



New models

Here we present 3 new models that we implemented both in R and Python;

The models are:

- Poisson regression model;
- Exponential survival model;
- Gaussian Mixture model.

Poisson regression model

A random variable Y is said to have a Poisson distribution with parameter λ if it takes integers $y = 0, 1, 2 \ldots$ with probability mass function

$$P(Y = y) = \frac{exp\{-\lambda\}\lambda^y}{y!},$$

where $\lambda > 0$. This mean can be modelled via a link function passed in a systematic component. For the Poisson regression case, the most widely used link function is the natural log, resulting in a equation that has the form

$$log(\hat{\lambda}) = \beta_0 + \beta_1\phi(x_1) + \cdots + \beta_n\phi(x_n).$$

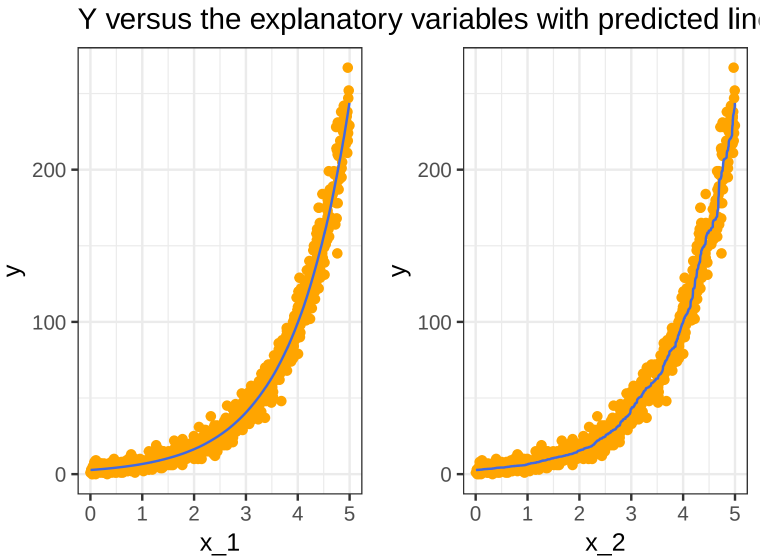
Poisson regression model

```
model
{
  # Likelihood
  for (i in 1:T) {
    y[i] ~ dpois(p[i])
    log(p[i]) <- alpha + beta_1 * x_1[i] +
                         beta_2 * x_2[i]
  }
  # Priors
  alpha ~ dnorm(0.0, 0.01)
  beta_1 ~ dnorm(0.0, 0.01)
  beta_2 ~ dnorm(0.0, 0.01)
}
'
```

Poisson regression model

```r
Simulate data -----------------------

# Some R code to simulate data from the Poisson model
T = 1000
set.seed(123)
x_1 = sort(runif(T, 0, 5))
x_2 = sort(runif(T, 0, 5))
alpha = 1
beta_1 = 1.2
beta_2 = -0.3
mu = alpha + beta_1 * x_1 + beta_2 * x_2
lambda = exp(mu)
y = rpois(n = T, lambda = lambda)
```

Poisson regression model



Exponential Survival models

In survival analysis, we are usually interested in modelling the time until a certain event occurs. Let T be a random variable representing the survival times of individuals in some population.

$$F(t) = P(T \leq t) = \int_0^t f(u)du$$

Survival data is also often censored. In this case, the likelihood is written as

$$L(t) = \prod_{i=1}^{n} [f(t_i)]^{\delta_i} [S(t_i)]^{1-\delta_i},$$

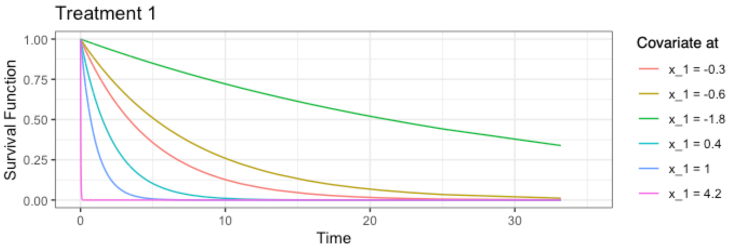
where δ_i is the indicator variable that takes 1 for the failures and 0 for censored observations. We consider for the failure time the Exponential distribution, given by

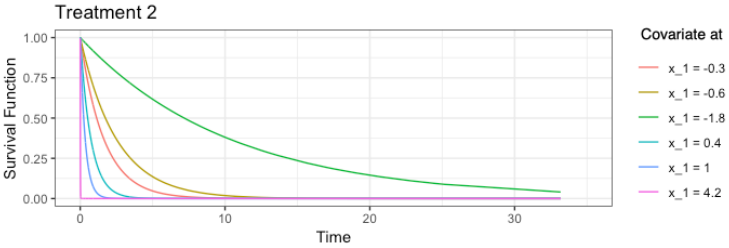
$$f(t) = \frac{1}{\alpha} exp\Big\{ -\frac{t}{\alpha} \Big\}, \quad \alpha > 0.$$

Exponential Survival models

```
model
{
  # Likelihood
  for (i in 1:T) {
    mu[i] = exp(beta_1 * x_1[i] + beta_2 * x_2[i])
    t[i] ~ dexp(mu[i] * lambda_0)
  }
  # Priors
  lambda_0 ~ dgamma(1, 1)
  beta_1 ~ dnorm(0.0, 0.01)
  beta_2 ~ dnorm(0.0, 0.01)
}
'
```

Exponential Survival models





Gaussian Mixture model

The mixture model is viewed hierarchically: the observations y are modeled conditionally on the vector z, having z itself a probabilistic specification.

$$p(y_i|\theta, \lambda) = \lambda_1 f(y_i|\theta_i) + \cdots + \lambda_H f(y_i|\theta_H),$$

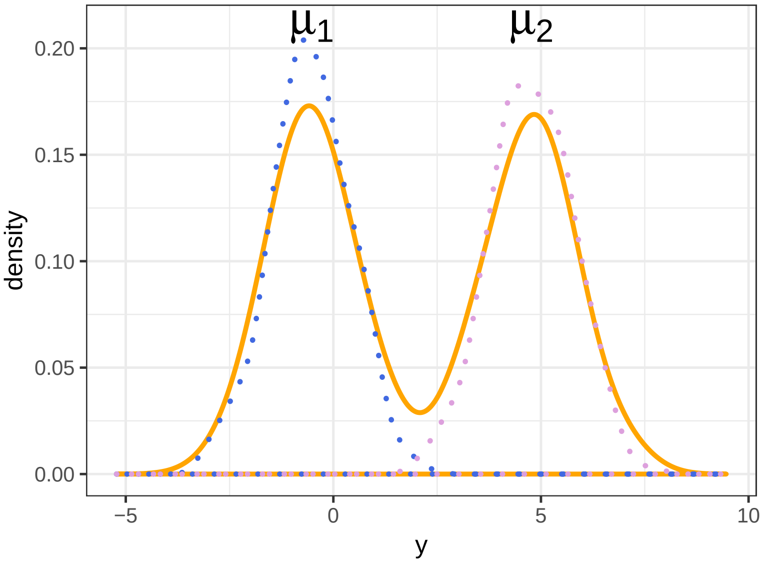
where the vector $\lambda = (\lambda_1, \ldots, \lambda_H)$ represents the proportions of the population taken as being drawn from each $f_h(y_i|\theta_h)$ distribution, for $h = 1, \ldots, H$, also that $\sum_{h=1}^{H} \lambda_h = 1$. Usually, the mixture components are assumed to be part of the same parametric family, such as the Gaussian, but with different parameter vectors. The unobserved variables is written as

$$z_{ih} = \begin{cases} 1, & \text{of the } i\text{th unite is drawn from the } h\text{th component} \\ 0, & \text{otherwise} \end{cases}$$

Gaussian Mixture model

```
model {
  # Likelihood:
  for(i in 1:N) {
    y[i] ~ dnorm(mu[i] , 1/sigma_inv)
    mu[i] <- mu_clust[clust[i]]
    clust[i] ~ dcat(lambda_clust[1:Nclust])
  }
  # Prior:
  sigma_inv ~ dgamma( 0.01 ,0.01)
  mu_clust[1] ~ dnorm(0, 10)
  mu_clust[2] ~ dnorm(5, 10)

  lambda_clust[1:Nclust] ~ ddirch(ones)
}
```

Gaussian Mixture model



Final remarks

- This project we learned how to use JAGS to perform Bayesian analysis;
- We had some challenges: i) JAGS was new for whole the group; ii) Difficulties to run the package **pyjags**;
- Our contributions were to provide mathematical details and codes for existing and new models. In the end, we produced:
  - 8 R scripts (3 new);
  - 3 Python scripts (all of them new);
- As future work other models can be implemented, such as other GLMs, Geostatistical models, more complex Survival models and other GLMMs with/without longitudinal structure.

Acknowledgments

This work was supported by a Science Foundation Ireland Career Development Award grant number: 17/CDA/4695

Science Foundation Ireland sfi For what's next

References

Denison, David G. T., Christopher C. Holmes, Bani K. Mallick, and Adrian F. M. Smith. 2002. Bayesian Methods for Nonlinear Classification and Regression. Wiley.

Gelman, Andrew, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2004. Bayesian Data Analysis. 2nd ed. Chapman; Hall/CRC.

Kalbfleisch, J. D., and R. L. Prentice. 2002. The Statistical Analysis of Failure Time Data. 2nd ed. John Wiley & Sons.

Ntzoufras, Ioannis. 2008. Bayesian Modeling Using WinBUGS. doi:10.1002/9780470434567.

Royle, J Andrew, and Robert M Dorazio. 2008. Hierarchical Modeling and Inference in Ecology: The Analysis of Data from Populations, Metapopulations and Communities. Elsevier.

Ferrari, Silvia, and Francisco Cribari-Neto. 2004. “Beta Regression for Modelling Rates and Proportions.” *Journal of Applied Statistics* 31 (7). Taylor & Francis: 799–815.