A Survey of Bayes Methods in R

Maxwell Lovig

June 24, 2025


Prego
CLASSICO
WHITE CLAM SAUCE
SILVER PALATE
SAN MARZANO
MARINARA
TOMATO BASIL
ROASTED GARLIC
BERTOLLI
d'ITALIA
MID'S
NEWMAN'S OWN
MARINARA
SOCKAROONI
TOMATO & BASIL
RAGÚ
Simply
Garden
CHUNKY
Traditional
FIRE ROASTED
HOMESTYLE ALFREDO
NOW
2/$5
$4.99
$3.99
$2.29
$1.89
BOGO

CRAN Task View: Bayesian Inference

Applied researchers interested in Bayesian statistics are increasingly attracted to R because of the ease of which one can code algorithms to sample from posterior distributions as well as the significant number of packages contributed to the Comprehensive R Archive Network (CRAN) that provide tools for Bayesian inference. This task view catalogs these tools. In this task view, we divide those packages into four groups based on the scope and focus of the packages. We first review R packages that provide Bayesian estimation tools for a wide range of models. We then discuss packages that address specific Bayesian models or specialized methods in Bayesian statistics. This is followed by a description of packages used for post-estimation analysis. Finally, we review packages that link R to other Bayesian sampling engines such as JAGS, OpenBUGS, WinBUGS, Stan, and TensorFlow.

General Purpose Model-Fitting Packages

- The arm package contains R functions for Bayesian inference using lm, glm, mer and polr objects.
- BACCO is an R bundle for Bayesian analysis of random functions. BACCO contains three sub-packages: emulator, calibrator, and approximator, that perform Bayesian emulation and calibration of computer programs.
- bayesforecast provides various functions for Bayesian time series analysis using 'Stan' for full Bayesian inference. A wide range of distributions and models are supported, allowing users to fit Seasonal ARIMA, ARIMAX, Dynamic Harmonic Regression, GARCH, t-student innovation GARCH models, asymmetric GARCH, Random Walks, stochastic volatility models for univariate time series.
- bayesm provides R functions for Bayesian inference for various models widely used in marketing and micro-econometrics. The models include linear regression models, multinomial logit, multinomial probit, multivariate probit, multivariate mixture of normals (including clustering), density estimation using finite mixtures of normals as well as Dirichlet Process priors, hierarchical linear models, hierarchical multinomial logit, hierarchical negative binomial regression models, and linear instrumental variable models.
- BayesianTools *(archived)* is an R package for general-purpose MCMC and SMC samplers, as well as plot and diagnostic functions for Bayesian statistics, with a particular focus on calibrating complex system models. Implemented samplers include various Metropolis MCMC variants (including adaptive and/or delayed rejection MH), the T-walk, two differential evolution MCMCs, two DREAM MCMCs, and a sequential Monte Carlo (SMC) particle filter.
- LaplacesDemon seeks to provide a complete Bayesian environment, including numerous MCMC algorithms, Laplace Approximation with multiple optimization algorithms, scores of examples, dozens of additional probability distributions, numerous MCMC diagnostics, Bayes factors, posterior predictive checks, a variety of plots, elicitation, parameter and variable importance, and numerous additional utility functions.
- loo provides functions for efficient approximate leave-one-out cross-validation (LOO) for Bayesian models using Markov chain Monte Carlo. The approximation uses Pareto smoothed importance sampling (PSIS), a new procedure for regularizing importance weights. As a byproduct of the calculations, loo also provides standard errors for estimated predictive errors and for the comparison of predictive errors between models. The package also provides methods for using stacking and other model weighting techniques to average Bayesian predictive distributions.
- MCMCpack provides model-specific Markov chain Monte Carlo (MCMC) algorithms for wide range of models commonly used in the social and behavioral sciences. It contains R functions to fit a number of regression models (linear regression, logit, ordinal probit, probit, Poisson regression, etc.), measurement models (item response theory and factor models), changepoint models (linear regression, binary probit, ordinal probit, Poisson, panel), and models for ecological inference. It also contains a generic Metropolis sampler that can be used to fit arbitrary models.
- The mcmc package consists of an R function for a random-walk Metropolis algorithm for a continuous random vector.
- The nimble package provides a general MCMC system that allows customizable MCMC for models written in the BUGS/JAGS model language. Users can choose samplers and write new samplers. Models and samplers are automatically compiled via generated C++. The package also supports other methods such as particle filtering or whatever users write in its algorithm language.

Application-Specific Packages

ANOVA

- bayesanova provides a Bayesian version of the analysis of variance based on a three-component Gaussian mixture for which a Gibbs sampler produces posterior draws.
- AovBay provides the classical analysis of variance, the nonparametric equivalent of Kruskal Wallis, and the Bayesian approach.

Bayes factor/model comparison/Bayesian model averaging

- bain computes approximated adjusted fractional Bayes factors for equality, inequality, and about equality constrained hypotheses.
- BayesFactor provides a suite of functions for computing various Bayes factors for simple designs, including contingency tables, one- and two-sample designs, one-way designs, general ANOVA designs, and linear regression.
- BayesVarSel calculate Bayes factors in linear models and then to provide a formal Bayesian answer to testing and variable selection problems.
- The BMA package has functions for Bayesian model averaging for linear models, generalized linear models, and survival models. The complementary package ensembleBMA uses the BMA package to create probabilistic forecasts of ensembles using a mixture of normal distributions.
- BMS is Bayesian Model Averaging library for linear models with a wide choice of (customizable) priors. Built-in priors include coefficient priors (fixed, flexible and hyper-g priors), and 5 kinds of model priors.
- bridgesampling provides R functions for estimating marginal likelihoods, Bayes factors, posterior model probabilities, and normalizing constants in general, via different versions of bridge sampling (Meng and Wong, 1996).
- RoBMA implements Bayesian model-averaging for meta-analytic models, including models correcting for publication bias.

Bayesian tree models

- dbarts fits Bayesian additive regression trees (Chipman, George, and McCulloch 2010).
- The bartBMA offers functions for Bayesian additive regression trees using Bayesian model averaging.
- bartCause contains a variety of methods to generate typical causal inference estimates using Bayesian Additive Regression Trees (BART) as the underlying regression model (Hill 2012).

JAGS

JAGS

- Developed By Martyn Plummer
- Uses Gibbs sampler
- Interfaces with R using *rjags* and *R2Jags*

Laplace's Demon

- Created by Byron Hall/Statiscat, maintained by Henrik Singmann
- "Dealers Choice" (HARM, Metropolis within Gibbs, CHARM, etc.)
- Written in R/Ccp

Pick any number you like, but there's only one number appropriate for a demon[1]

```
>   set.seed(666)
```

[1]Demonic references are used only to add flavor to the software and its use, and in no way endorses beliefs in demons. This specific pseudo-random seed is often referred to, jokingly, as the 'demon seed'

Stan

- Started by Andrew Gelman, 54 current employees
- Uses Hamiltonian Monte Carlo
- Interfaces with R using *rstan*

Objective Bayes

Iris Dataset
x = sepal length
c = name of flower (1 = setosa, 2 = versicolor, 3 = viginica)

$$\mu_i \propto 1 \qquad (i \in \{1, 2, 3\})$$

$$\sigma_i \propto \frac{1}{\sigma} \quad \left(\sigma_i^2 \propto \frac{1}{\sigma^2}\right)$$

$$x_j \sim N(\mu_{c[j]}, \sigma_{c[j]}) \qquad (j \in \{1, \cdots, N\})$$

Iris: Stan

```
// In this example we have a vector of responses y with mixed grou
// We keep track of which responses are for which group by c
data {
  int N;
  real y[N];
  int c[N];
}

parameters {
  real mu[3];
  real<lower=0> sigma[3];
}

model {
  for (i in 1:3){
      target += -1 * log(sigma[i]);
      // Jeffries prior in the logged form
  }
  for (i in 1:N){
    y[i] ~ normal(mu[c[i]], sigma[c[i]]);
  }
}
```

Iris: Jags

```
model {
  # Priors
  for (i in 1:3) {
    mu[i] ~ dnorm(0,.0001)
    # Cannot use jeffries prior lol
  }

  for (i in 1:3) {
    sigma[i] ~ dgamma(.001, .001)
    # Almost Jeffries Prior
    tau[i] <- 1/(sigma[i] * sigma[i])
    # Jags uses the precision
    # parameterizaion for Normal
  }

  # Likelihood
  for (i in 1:N) {
    y[i] ~ dnorm(mu[c[i]], tau[c[i]])
  }
}
```

Aside

Published: 20 August 2013

Extending JAGS: A tutorial on adding custom distributions to JAGS (with a diffusion model example)

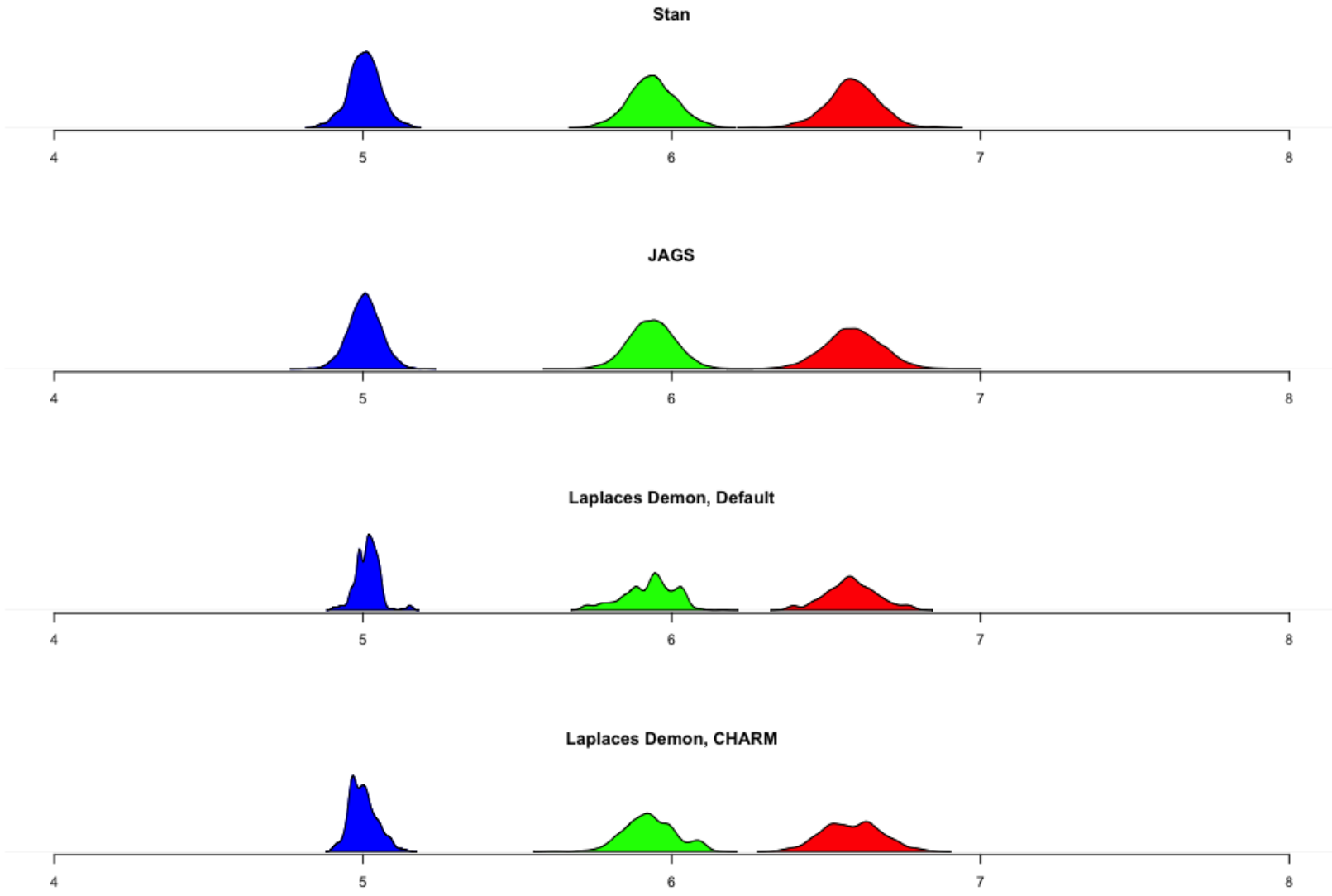
Dominik Wabersich & Joachim Vandekerckhove

Behavior Research Methods **46**, 15–28 (2014) | Cite this article

Iris: Laplaces Demon

```
Model <- function(parm, Data) {
  ## Parameters
  mu <- parm[1:3]
  sigma <- exp(parm[4:6])
  # Recall we must reparameterize log.sigma

  ## Priors
  sigma.prior <-  -2 * log(sigma)

  ## Likelihood and Posterior Predictivwe
  LL <- 0
  pp <- rep(0,length(Data$C))

  for (i in 1:length(Data$C)){
    LL <- LL + dnorm(Data$y[i], mu[Data$C[i]],
                     sigma[Data$C[i]], log=TRUE)
    pp[i] <- rnorm(1, mu[Data$C[i]], sigma[Data$C[i]])
  }

  ## Calculate the Posterior
  LP <- LL + sum(sigma.prior)

  Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP),
                   yhat=pp, parm=parm)
  return(Modelout) }
```

Iris: Results



Hierarchical Bayes

Jay's Dataset
x = Final enrollment, n = Maximum enrollment
c = class identifier (15 = 230, 44 = 662)
$i \in \{1, \cdots, 49\}$ and $j \in \{1, \cdots, N\}$.

$$\alpha \propto 1$$

$$\beta \propto 1$$

$$\theta_i \propto B(\alpha, \beta)$$

$$\lambda_i \propto \frac{1}{\sqrt{\lambda}}$$

$$x_j \sim Bi(n_j, \theta_{c[j]})$$

$$n_j \sim Poi(\lambda_{c[j]})$$

Student: Stan

```
data {
  int<lower = 0> N; int<lower = 0> M;
  int x[N]; int n[N]; int c[N];
}

parameters {
  real<lower = 0> lambda[M];  real<lower = 0> alpha;
  real<lower = 0> beta; real<lower = 0, upper = 1> theta[M];
}

model {
  for (i in 1:M){
    theta[i] ~ beta(alpha, beta);
    // The hierarchical bit is here
    target += -.5 * log(lambda[i]);
    // Jeffries for lambda
  }

  for (i in 1:N){
    n[i] ~ poisson(lambda[c[i]]);
    x[i] ~ binomial(n[i], theta[c[i]]);
  }
}
```

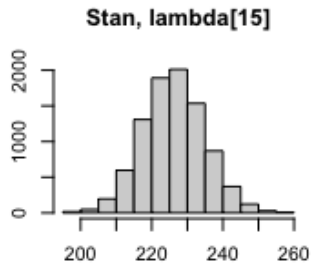
Student: JAGS

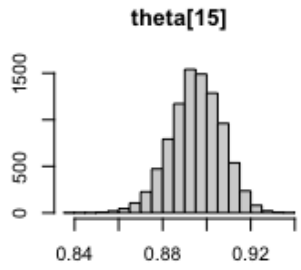
```
model {
  # Priors
  alpha ~ dunif(0,1000)
  beta ~ dunif(0,1000)

  for (i in 1:M) {
    lambda[i] ~ dgamma(.5, .0001)
    theta[i] ~ dbeta(alpha, beta)
    # Here is the hierarchical bit
  }

  # Likelihood
  for (i in 1:N) {
    n[i] ~ dpois(lambda[c[i]])
    x[i] ~ dbin(theta[c[i]], n[i])
  }
}
```

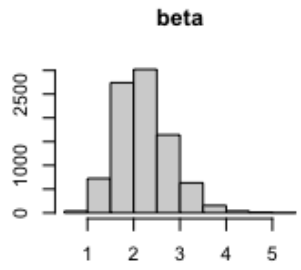
Student: Laplace's Demon

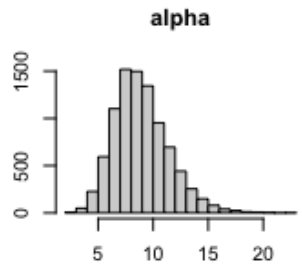
```
Model <- function(parm, Data) {
  ### Parameters
  parm[99] <- alpha <- interval(parm[99], a=1)
  parm[100] <- beta <- interval(parm[100], a=1)
  parm[1:49] <- pi <- interval(parm[1:49], 0.001, 0.999)
  parm[50:98] <- lambda <- interval(parm[50:98], a = 0)

  ### Log(Prior Densities)
  pi.prior <- dbeta(pi, alpha, beta, log=TRUE)
  lambda.prior <- dgamma(lambda, 1, .0001,
                         log = TRUE)

  ### Log-Likelihood
  LL <- 0
  for (i in 1:length(Data$y)){
    LL <- LL + dpois(Data$n[i], lambda[Data$c[i]], log = TRUE)
    LL <- LL + dbinom(Data$y[i], Data$n[i], pi[Data$c[i]],
                      log = TRUE) }

  ### Log-Posterior
  LP <- LL + sum(pi.prior) + sum(lambda.prior)
  Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP,
                   yhat=rbinom(length(Data$n), Data$pi),
                   parm=parm)
  return(Modelout)}
```

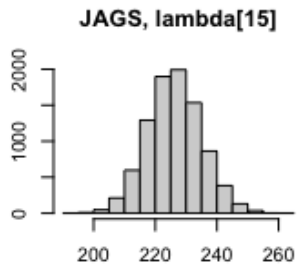
Student: S&DS 230

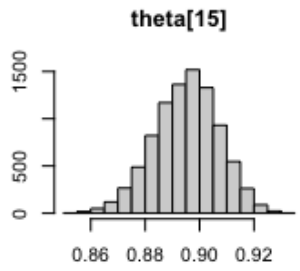


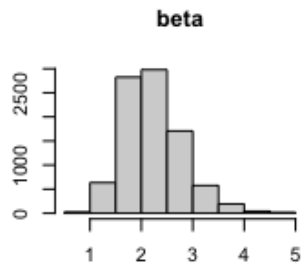


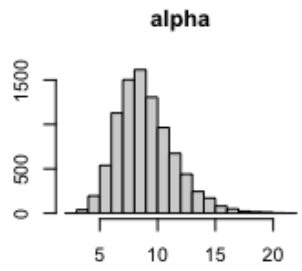


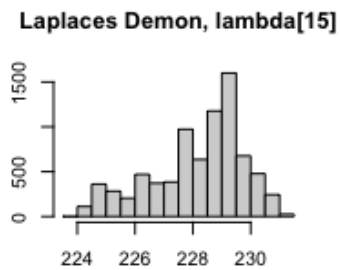


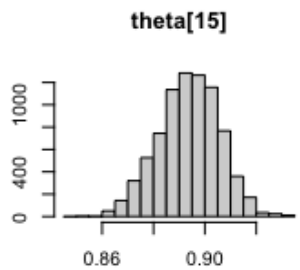


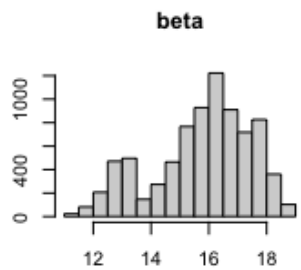


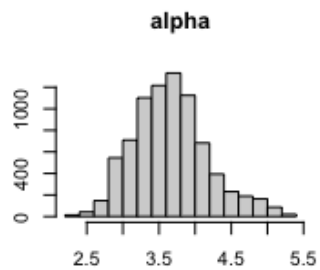




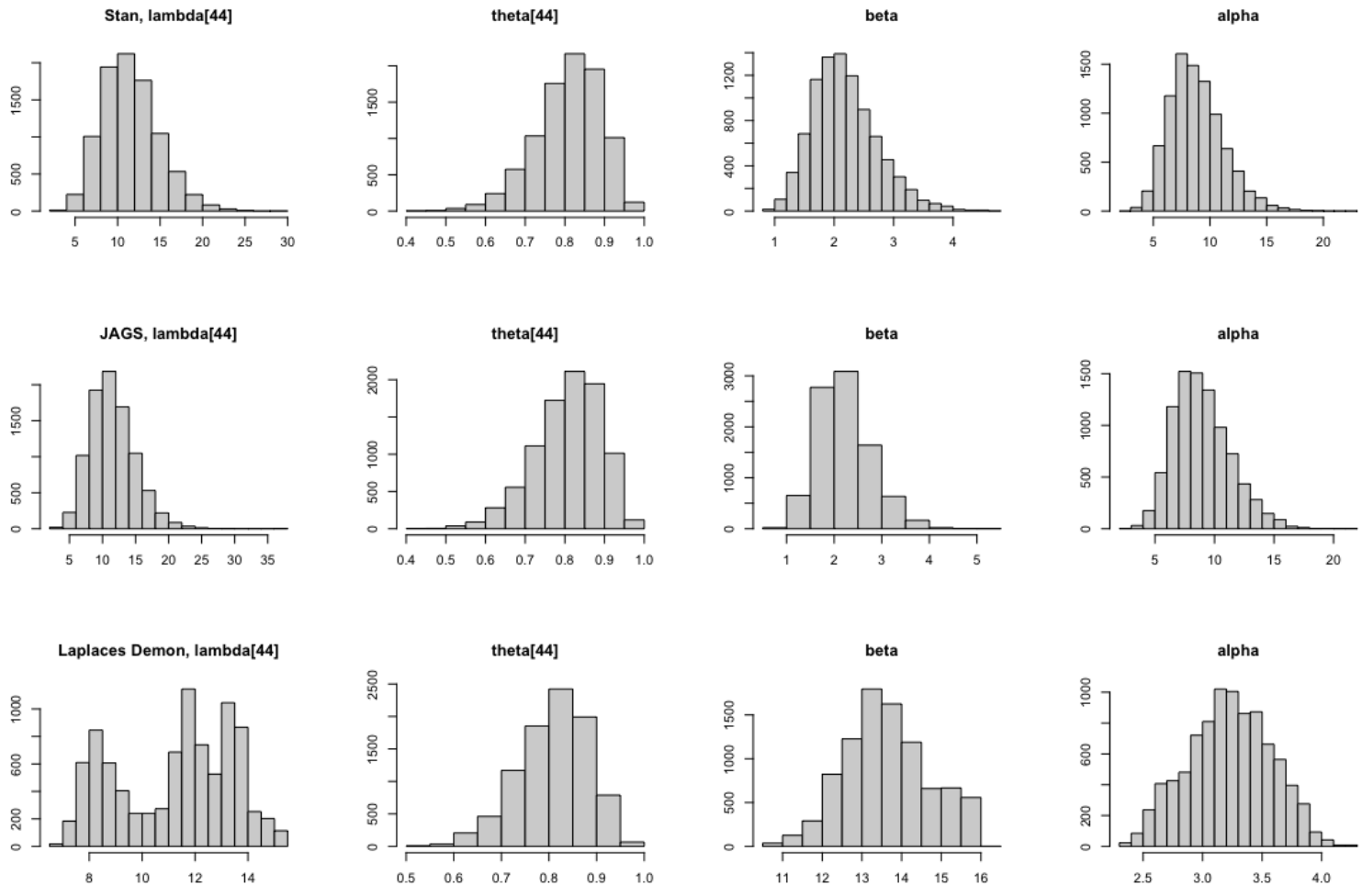








Student: S&DS 662



Speed

Isurance Dataset (Kaggle)
x = 9 predictor columns (age, sex, location, etc.)
y = Insurance claim amount
$j \in \{1, \cdots, N\}$.

$$\beta \sim N(\vec{0}, 10\mathrm{I})$$
$$\sigma^2 \sim G(1, 1)$$
$$y_j \sim N(x_j\beta, \sigma)$$

Insurance: Stan

```
data {
  int<lower=0> N;
  int<lower=0> P;
  matrix[N, P] x;
  vector[N] y;
}

parameters {
  vector[P] beta;
  real<lower=0> sigma;
}

model {
    beta ~ normal(0,10);
    sigma ~ gamma(1,1);
    y ~ normal(x * beta, sigma);
}
```

Insurance:

```
model {
  # Priors
  for (i in 1:P){
    beta[i]~dnorm(0,1/100)
  }

  sigma ~ dgamma(1,1)
  tau = 1 / (sigma)

  # Likelihood
  mu = x %*% beta
  for (i in 1:length(y)){
    y[i] ~ dnorm(mu[i], tau)
  }

}
```

Insurance:

```
Model <- function(parm, Data) {
  ### Parameters
  beta <- parm[Data$pos.beta]
  parm[Data$pos.sigma] <- sigma <-
                          interval(parm[Data$pos.sigma], 1e-100)
  ### Log-Prior
  beta.prior <- sum(dnormv(beta, 0, 100, log=TRUE))
  sigma.prior <- dgamma(sigma, 1, 1, log=TRUE)

  ### Log-Likelihood
  mu <- tcrossprod(Data$x, t(beta))
  LL <- sum(dnorm(Data$y, mu, sigma, log=TRUE))

  ### Log-Posterior
  LP <- LL + beta.prior + sigma.prior
  Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP,
                   yhat=rnorm(length(mu), mu, sigma), parm=parm)
  return(Modelout)
}
```

Insurance:

	test	replications	elapsed	relative	user.self	sys.self
2	JAGS	5	133.976	3.069	130.279	1.832
3	LD, CHARM	5	407.371	9.332	393.246	11.260
4	LD, HARM	5	43.653	1.000	41.533	1.547
1	Stan	5	169.697	3.887	166.614	1.641

JAGS