Bayesian Data Analysis - Assignment 5

October 22, 2017

1 Generalized linear model: Bioassay with Metropolis

proporsal distribution: (based on $J_t(\theta^* \mid \theta^{t-1}) = N(\theta^* \mid \theta^{t-1}, c^2\Sigma)$)

$$N\left(\begin{pmatrix}\alpha\\ \beta\end{pmatrix}, \begin{pmatrix}3 & 0\\ 0 & 3\end{pmatrix}\right)$$

starting point: $\theta^0 = (\alpha,\beta)^0 = (0,5)$
number of chains: 10
the number of samples generated from each chain: 1000
the warm-up length: 200
Python code:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
from scipy.stats import norm

# data
x = np.array([-0.86, -0.30, -0.05, 0.73])
n = np.array([5, 5, 5, 5])
y = np.array([0, 1, 3, 5])

# log posterior distribution with parameter a, b
def log_posterior_distribution(a,b):
  ilogit_abx = 1 / (np.exp(-(a + b * x)) + 1)
  log_p =np.log(np.prod(ilogit_abx**y * (1 - ilogit_abx)**(n -
                              y)))
  return log_p

chains = 10  # number of chains
iterations = 1000  # number of iterations
starting_point =[0,5]  # starting points
a=np.zeros((iterations,chains))  # parameter a
```

```
b=np.zeros((iterations,chains))  # parameter b
a[0,0]=starting_point[0]
b[0,0]=starting_point[1]

# Metropolis algorithm
for i in range(iterations):
  for j in range(chains):
  # sample a set of proposal parameters.a~N(a[i-1,j],3),b~n(b[i
                              -1,j],3)
  a_new = norm.rvs(loc=a[i-1,j],scale=np.sqrt(3),size=1)
  b_new = norm.rvs(loc=b[i-1,j],scale=np.sqrt(3),size=1)
  # calculate the new and the previous one log posterior
                              distributions
  log_p_new = log_posterior_distribution(a_new,b_new)
  log_p_prev = log_posterior_distribution(a[i-1,j],b[i-1,j])
  # generate a random number between 0 and 1
  rand = np.random.uniform(0,1)
  # calculate the ratio of the densities
  # do comparision and set parameters (BDA3 P278)
  if min(np.log(1),log_p_new - log_p_prev)>np.log(rand):
    a[i,j]= a_new
    b[i,j]= b_new
  else:
    a[i,j]=a[i-1,j]
    b[i,j]=b[i-1,j]

# (remove the 200 warm-up samples)
a=a[200:]
b=b[200:]
# scatter plot
plt.figure()
plt.scatter(a,b,s=10,edgecolor='black')
plt.xlabel(r'$\alpha$')
plt.ylabel(r'$\beta$')
plt.title("Scatter plot with Metropolis algorithm")

# posterior density
plt.figure()
A = np.linspace(-2, 6, 100)
B = np.linspace(0, 25, 100)
ilogit_abx = 1 / (np.exp(-(A[:,None] + B[:,None,None] * x)) + 1
                              )
p = np.prod(ilogit_abx**y * (1 - ilogit_abx)**(n - y), axis=2)
plt.contourf(p, origin='lower', aspect='auto',
extent=(A[0], A[-1], B[0], B[-1]))
plt.xlim([-2,6])
plt.ylim([0,25])
plt.ylabel(r'$\beta$')
plt.xlabel(r'$\alpha$')
```

```python
plt.grid('off')
plt.title('posterior density')

# psrf
def psrf(samples):
  # Calculate means W of the variances
  W = np.mean(np.var(samples,axis=1,ddof=1),axis=0)
  # Calculate variances B (in fact B/n) of the means
  Bpn = np.var(np.mean(samples,axis=1),ddof=1,axis=0)
  B = Bpn*800
  Vh = (800-1)/800*W +Bpn
  R = np.sqrt(Vh/W)
  return R

print("The R of a: {:.6f}".format(psrf(a)))
print("The R of b: {:.6f}".format(psrf(b)))

plt.show()
```

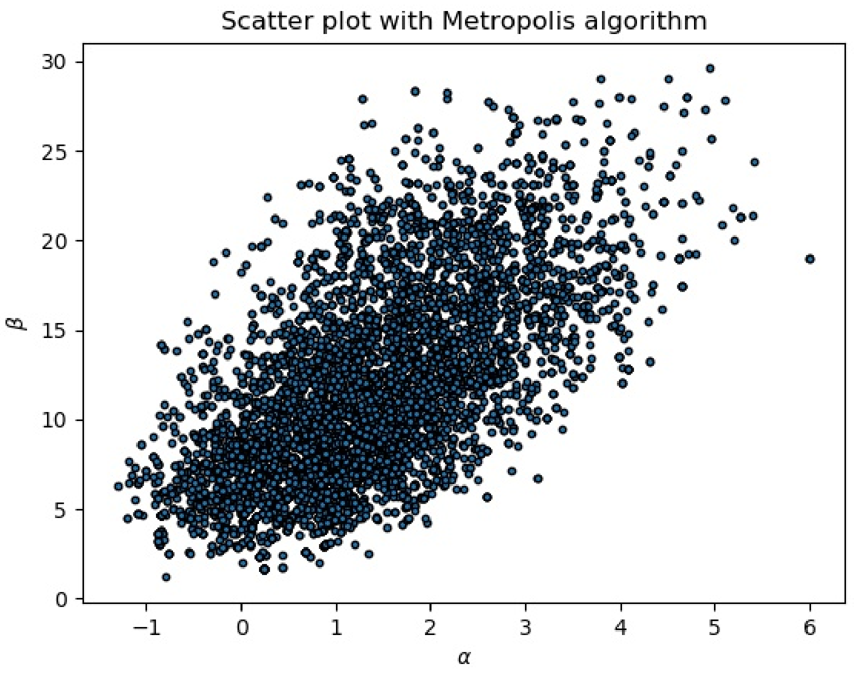

Figure 1: scatter plot with Metropolis algorithm

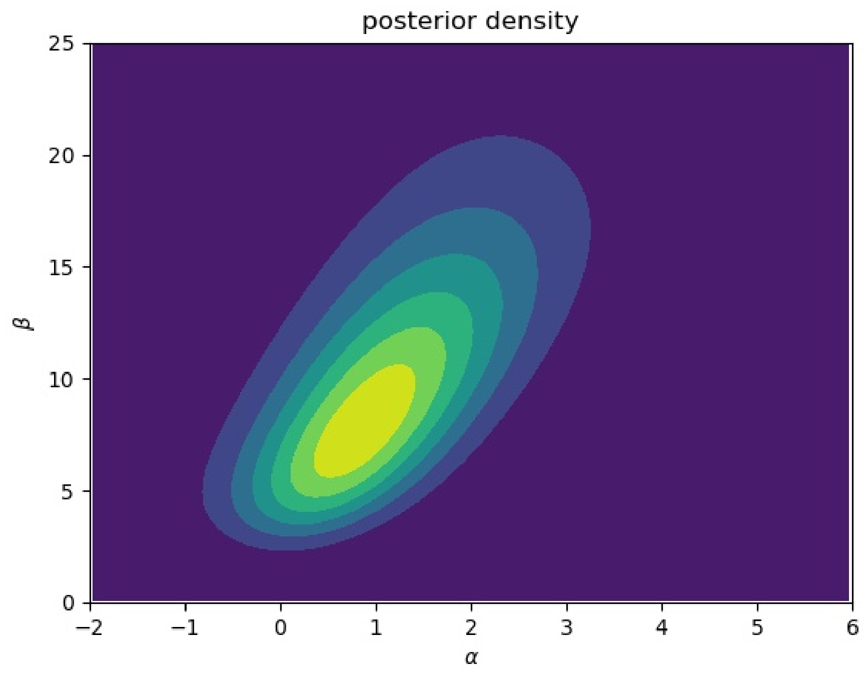

Figure 2: posterior density

$\hat{R}$ of α: 1.033702
$\hat{R}$ of β: 1.023332

The scatter plot matches the figure 2.
The $\hat{R}$ of α and the $\hat{R}$ of β are both very close to 1. So we can conclude that the chains have been converged (the samples were from the same distribution).

2 Generalized linea rmodel: Bioassay with Stan

biossary.stan:

```
data{
  int<lower=0> J; // number of doses
  vector[J] x; // values of doses
  int<lower=0> n[J]; // number of animals
  int <lower=0> y[J]; // number of deaths
  }
parameters{
  real alpha;
  real beta;
}
transformed parameters{
  vector[J] logits;
  logits = alpha + beta*x; // Link function
}
model{
  y~binomial_logit(n,logits);
}
```

R code:

```
library("rstan")
library("ggplot2")
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

biossary_data <- list(J=4,
                      x=c(-0.86,-0.30,-0.05,0.73),
                      n=c(5,5,5,5),
                      y=c(0,1,3,5))
biossary_fit<-stan(file="biossary.stan",data=biossary_data,iter=1000,chains=10)
print(biossary_fit)
biossary_result<-extract(biossary_fit)
p = data.frame(alpha=biossary_result$alpha,beta=biossary_result$beta)
ggplot(p,aes(alpha,beta))+geom_point(shape=21, fill="blue", color="darkred")
```

Output:

```
Inference for Stan model: biossary.
10 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=5000.

           mean se_mean   sd   2.5%    25%   50%   75% 97.5% n_eff Rhat
alpha      1.27    0.03 1.08  -0.67   0.51  1.19  1.94  3.64  1376 1.01
beta      11.40    0.15 5.62   3.40   7.08 10.39 14.73 24.71  1319 1.01
logits[1] -8.53    0.11 4.22 -18.50 -10.88 -7.77 -5.40 -2.52  1481 1.01
```

```
logits[2] -2.15    0.03 1.29  -5.26  -2.85 -1.96 -1.24 -0.11  2298 1.00
logits[3]  0.70    0.02 0.93  -1.05   0.09  0.67  1.28  2.67  1614 1.00
logits[4]  9.60    0.14 4.87   2.50   5.80  8.77 12.58 20.95  1238 1.01
lp__      -6.98    0.03 1.08  -9.83  -7.42 -6.64 -6.20 -5.93  1205 1.01

Samples were drawn using NUTS(diag_e) at Sun Oct 22 10:40:12 2017.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

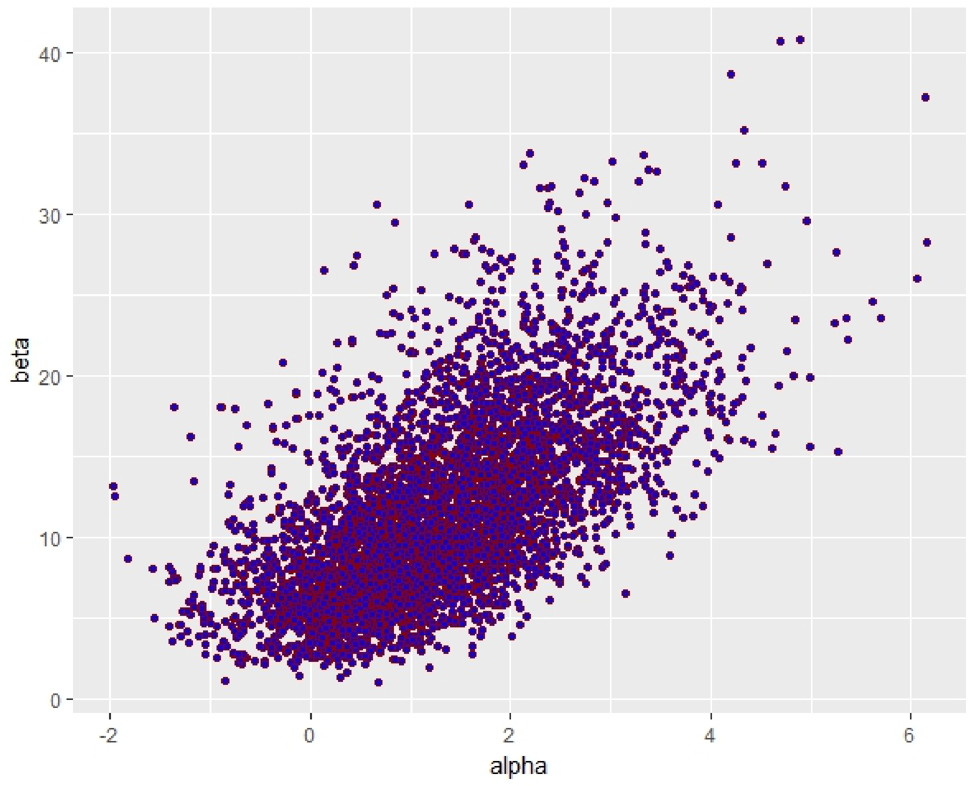

Figure 3: scatter plot with Stan

The scatter plot matches the Figure 2.
From the above results, we can see that the $\hat{R}$s of α and β are both 1.01, which is extremely close to 1. So we can conclude that the chains have been converged well.