Math 459: Bayesian Statistics – Homework 3

1. In this homework you will perform a Bayesian analysis of the gamma distribution using an uninformative prior and MCMC. The density of the gamma distribution is

$$f(x|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1}\exp(-\beta x), \quad \alpha,\beta > 0.$$

Derive Jeffreys prior (you must show your work to get credit). Some hints:

(a) To find the Fisher information matrix, you only need the log-likelihood for a sample of size 1.

(b) The derivatives of the natural logarithm of the gamma function are special functions. Note that $\partial \log(\Gamma(\alpha))/\partial\alpha$ is the `digamma` function (also in `R`) and $\partial^2 \log(\Gamma(\alpha))/\partial\alpha^2$ is the `trigamma` function.

Solution.

For Gamma distribution, the density function is

$$f(x|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1}\exp(-\beta x), \quad \alpha,\beta > 0.$$

To find the Fisher information matrix, we only need the log-likelihood for a sample of size 1. Then, the likelihood function is

$$L(\alpha,\beta|x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1}\exp(-\beta x)$$

and the log-likelihood function is

$$\log L(\alpha,\beta|x) = \alpha\log\beta - \log\Gamma(\alpha) + (\alpha-1)\log x - \beta x$$

First, calculate the first order derivatives for log-likelihood function $\log L(\alpha,\beta|x)$.

$$\frac{\partial \log L}{\partial \alpha} = \log\beta - \frac{\partial \log\Gamma(\alpha)}{\partial\alpha} + \log x$$

$$\frac{\partial \log L}{\partial \beta} = \frac{\alpha}{\beta} - x$$

Then, the second order derivatives are as follows:

$$\frac{\partial^2 \log L}{\partial \alpha^2} = -\frac{\partial^2 \log \Gamma(\alpha)}{\partial \alpha^2}$$
$$\frac{\partial^2 \log L}{\partial \alpha \partial \beta} = \frac{\partial^2 \log L}{\partial \beta \partial \alpha} = \frac{1}{\beta}$$
$$\frac{\partial^2 \log L}{\partial \beta^2} = -\frac{\alpha}{\beta^2}$$

Hence, we obtain the fisher information matrix:

$$I(\alpha, \beta) = \begin{bmatrix} \frac{\partial^2 \log \Gamma(\alpha)}{\partial \alpha^2} & -\frac{1}{\beta} \\ -\frac{1}{\beta} & \frac{\alpha}{\beta^2} \end{bmatrix}$$

Jeffreys' principle leads to defining the non-informative prior density as $\pi_J(\alpha, \beta) \propto \sqrt{\det I(\alpha, \beta)}$. Therefore, we have Jeffreys prior

$$\pi_J(\alpha, \beta) \propto \frac{1}{\beta} \sqrt{\alpha \frac{\partial^2 \log \Gamma(\alpha)}{\partial \alpha^2} - 1}$$

□

2. Write your own function to perform random walk Metropolis-Hastings sampling (with $10,000$ samples) from a density which is proportional to the above gamma density times Jeffreys prior. To get full credit, you must add comments to each step of the code to explain what is happening. Some guidelines:

(a) Remember that the default parameters for the `gamma` distribution in `R` are not the same as the usual gamma density above. Make sure you specify the gamma density as above in `R`.

(b) You can use any symmetric proposal density that you want.

(c) Make sure to save the output.

Solution.

See the following R code of random walk Metropolis-Hastings sampling and detailed comments included.

```
# generate sample data from a gamma distribution
set.seed(888)
alpha <- 1
beta <- 1
n <- 20
X <- rgamma(n, shape=alpha, rate=beta)

# define Jeffreys prior derived above
prior <- function(alpha,beta){
return (sqrt(trigamma(alpha)*alpha-1)/beta)
}

# define the target posterior function
target <- function(alpha,beta,X){
gamma <- dgamma(X,shape=alpha,rate=beta)
log_likelihood <- sum(log(gamma))
log_prior <- log(prior(alpha,beta))
return(exp(log_likelihood+log_prior))
}
```

```r
# use maximum-likelihood fitting function in r to generate initial values
require(MASS)

mle_fit <- fitdistr(X, "gamma",start=list(shape = 1, rate = 1))

alpha0 <- mle_fit$estimate[1]
beta0 <- mle_fit$estimate[2]
start <- unname(c(alpha0,beta0))

# spefiсy number of samples
n.sims <- 10000

# use 2-D normal as candidate distribution and sepcify variance-convariance matrix
cand.sd <- unname(diag(mle_fit$sd))

# define the random walk Metropolis-Hastings sampling function
rwmh <- function(target, n.sims, start, burnin, cand.sd, X)
{
require(MASS)
library(coda)
theta.cur <- start
draws <- matrix( NA, nrow = n.sims, ncol = 2)
for(i in 1:n.sims){
theta.can <- mvrnorm(1, theta.cur, cand.sd)
if (theta.can[1]>0 && theta.can[2]>0)
if(runif(1) <= min(1, target(theta.can[1], theta.can[2], X)
/target(start[1], start[2], X)))
theta.cur <- theta.can
draws[i, ] <- theta.cur
}
return(mcmc(unname(draws[(burnin + 1):n.sims, ])))
}

rwmh.draws <- rwmh(target, n.sims, start, burnin=0, cand.sd, X)
```

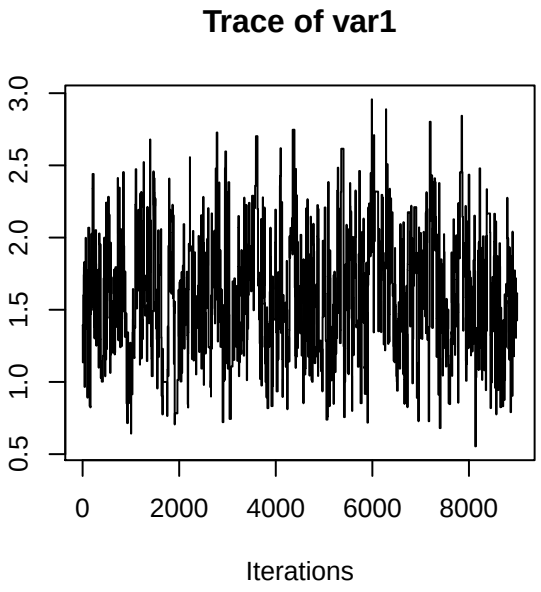
3. Use the `coda package` to give traceplots, autocorrelation function plots and perform all 4 diagnostic checks in the lecture notes (Gelman & Rubin, Geweke, Raftery & Lewis, and Heidelberg & Welch). Interpret these results.

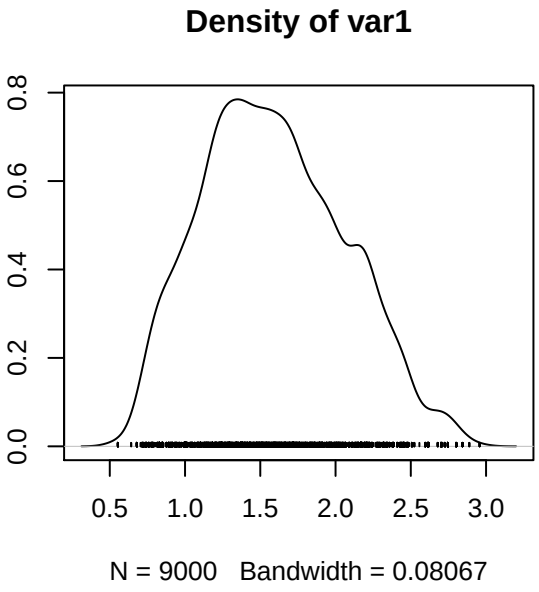
Solution.

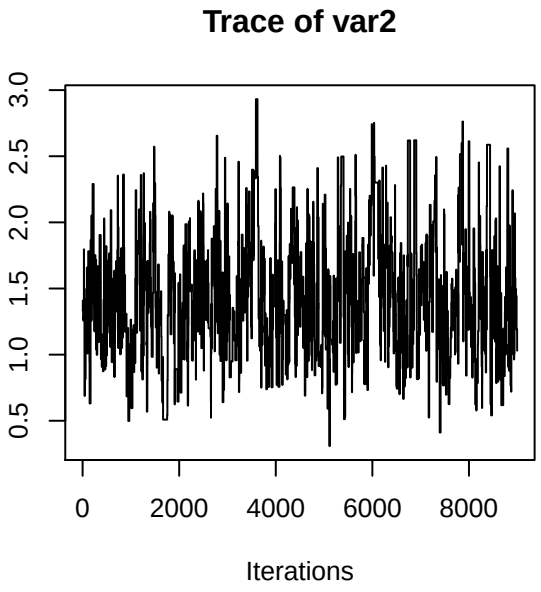
See the following R code for traceplots, autocorrelation plots and 4 different diagnostic checks.

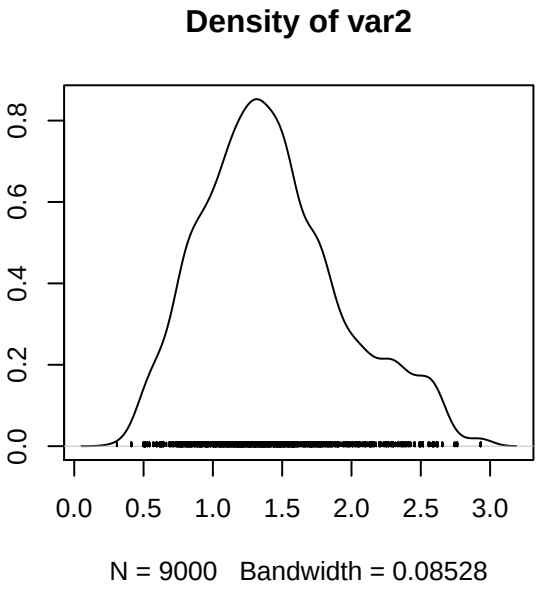
```
# generate samples by rwmh function defined above
library(coda)
set.seed(222)
rwmh.draws <- rwmh(target, n.sims, start, burnin=1000, cand.sd, X)

# Summarizing the Posterior Density
summary(rwmh.draws)

##
## Iterations = 1:9000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 9000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##       Mean     SD Naive SE Time-series SE
## [1,] 1.588 0.4702 0.004956        0.02886
## [2,] 1.428 0.5073 0.005348        0.03394
##
## 2. Quantiles for each variable:
##
##        2.5%   25%   50%   75% 97.5%
## var1 0.7822 1.240 1.552 1.916 2.485
## var2 0.5953 1.065 1.365 1.731 2.587

# traceplots of alpha and beta
```

```
plot(rwmh.draws)
```

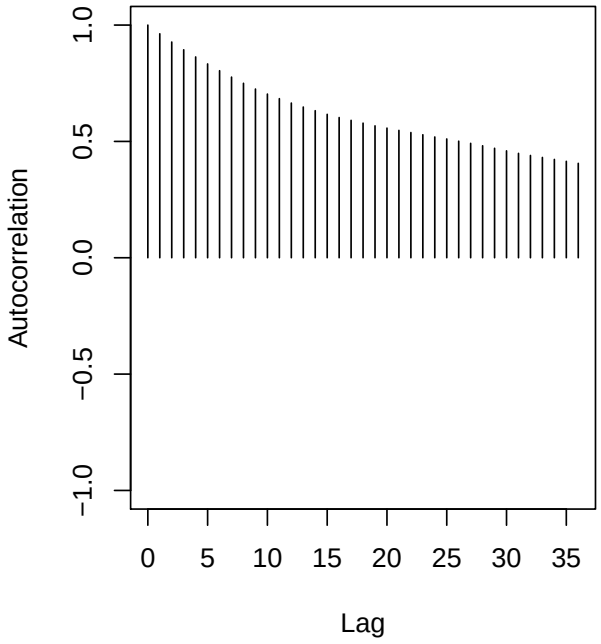


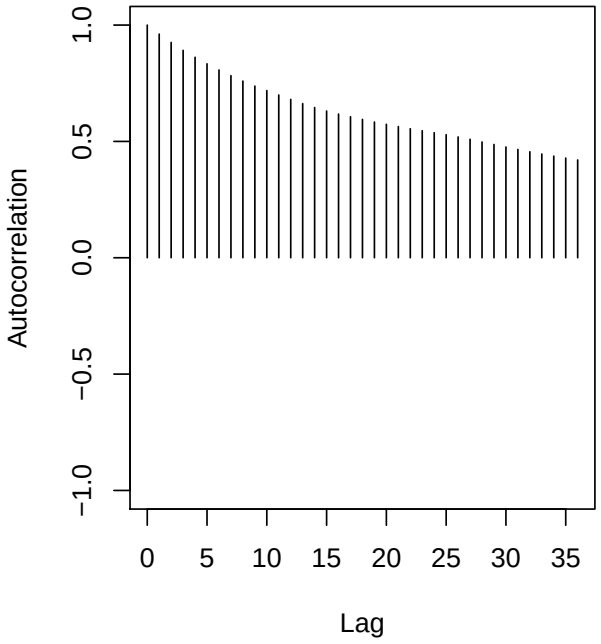






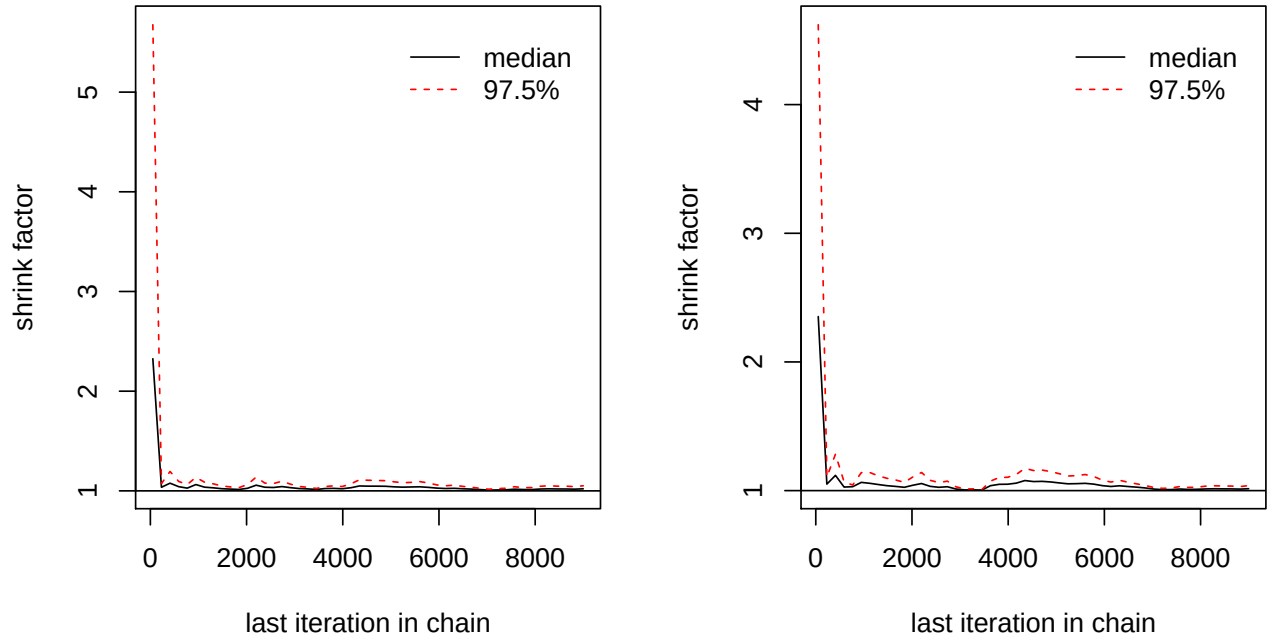

```
# autocorrelation plot of rwmh Draws
autocorr.plot(rwmh.draws)
```






```r
# (1) Gelman and Rubin diagnostic
rwmh.draws1 <- rwmh(target, n.sims, start, burnin=1000, cand.sd, X)
rwmh.draws2 <- rwmh(target, n.sims, start, burnin=1000, cand.sd, X)
rwmh.draws3 <- rwmh(target, n.sims, start, burnin=1000, cand.sd, X)
rwmh.draws4 <- rwmh(target, n.sims, start, burnin=1000, cand.sd, X)
rwmh.draws5 <- rwmh(target, n.sims, start, burnin=1000, cand.sd, X)
rwmh.list <- list(rwmh.draws1, rwmh.draws2, rwmh.draws3, rwmh.draws4, rwmh.draws5)

# Plotting how PSRF Changes through Iteration
gelman.diag(rwmh.list)

## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.01       1.01
## [2,]       1.00       1.01
##
```

```
## Multivariate psrf
##
## 1.01

gelman.plot(rwmh.list)
```




```
# (2) Geweke Diagnostic
geweke.diag(rwmh.draws)

##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    var1    var2
## -0.2430 -0.7624

# (3) Raftery and Lewis Diagnostic
raftery.diag(rwmh.draws,q=0.025,r=0.005,s=0.95)
```

```
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##  Burn-in  Total  Lower bound  Dependence
##  (M)      (N)    (Nmin)       factor (I)
##  109      132521 3746         35.4
##  127      141647 3746         37.8

# (4) Heidelberg and Welch Diagnostic
heidel.diag(rwmh.draws)

##
##      Stationarity start     p-value
##      test         iteration
## var1 passed       1         0.348
## var2 passed       1         0.197
##
##      Halfwidth Mean Halfwidth
##      test
## var1 passed    1.59 0.0566
## var2 passed    1.43 0.0665
```

Interpretation:

(1) Traceplots: The chain in the traceplot is stable, which indicates good mixing, i.e., the chain does not get stuck in a particular region of the parameter space.

(2) Autocorrelation Plot: Correlation between the parameters of a chain tends to decrease and produce faster convergence, but the decrease speed is very slow.

(3) Gelman and Rubin diagnostic: This diagnostic gives the median scale reduction factor and its 97.5% quantile (the PSRF is estimated with uncertainty since the chain lengths are finite), and also reports multivariate scale reduction factor. And we can see how the potential scale

reduction factor changes through the iterations using the *gelman.plot*() function. Results show a good convergence of the constructed chain.

(4) Geweke Diagnostic: This diagnostic takes two nonoverlapping parts (usually the first 0.1 and last 0.5 proportions) of the Markov chain and compares the means of both parts, using a difference of means test to see if the two parts of the chain are from the same distribution (null hypothesis). The z-values for each monitored variable in chain are rather small, hence suggesting that the first 10% of the samples do not arise from the same distribution as do the last 50%.

(5) Raftery and Lewis Diagnostic: This diagnostic was designed to test the number of iterations and burn-in needed by first running and testing a shorter 'pilot' chain. The resulting high dependence factors (35.4 and 37.8, > 5) cause concern: may be due to influential starting values, high correlations between parameters.

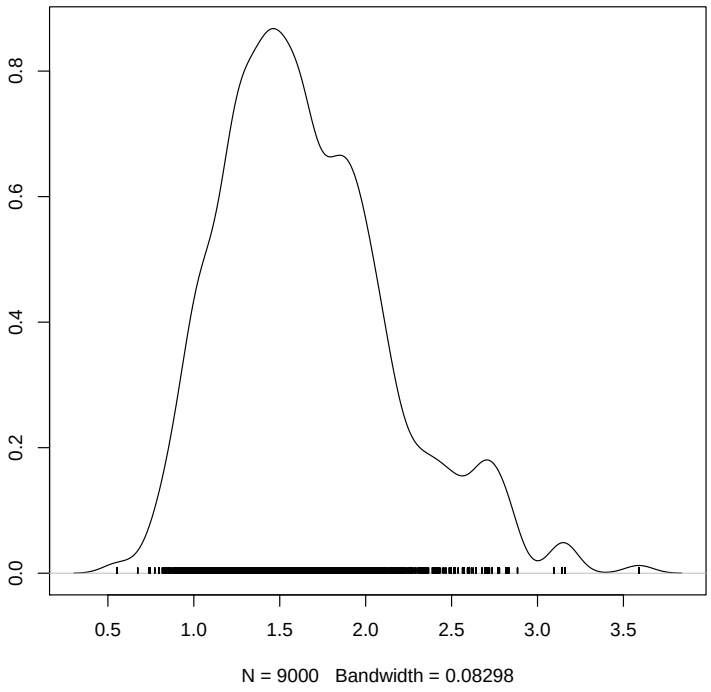
(6) Heidelberg and Welch Diagnostic: This disgnostic calculates a test statistic (based on Cramer-von Mises test statistic) to reject or fail to reject the null hypothesis that the Markov chain is from a stationary distribution. Results shows the chain pass all tests and indeed stable.

□

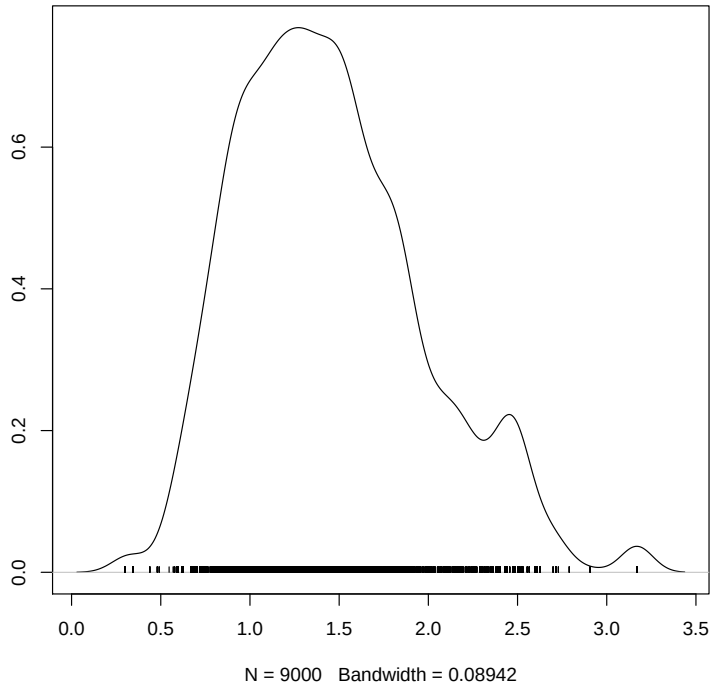
4. Plot the MCMC estimate of the marginal posterior density for each parameter (using the sampled values). Give an (estimated) 95% HPD interval for each parameter.

Solution.

```
# generate samples from the previous rwmh sampling function
library(coda)
set.seed(333)
rwmh.draws <- rwmh(target, n.sims, start, burnin=1000, cand.sd, X)

# plot the MCMC estimate of the marginal posterior density for each parameter
densplot(rwmh.draws[,1])
```




```
densplot(rwmh.draws[,2])
```




```
# calculate 95% HPD intervals
intervals = HPDinterval(rwmh.draws, prob = 0.95)
intervals

##          lower    upper
## var1 0.8520049 2.718169
## var2 0.5911429 2.473562
## attr(,"Probability")
## [1] 0.95
```

□