Quiz 4

VERSION: 2018-03-12 · 09:33:49

Instructions:

- In this quiz you are asked to write an MCMC algorithm for the posterior distribution of a statistical model, and prepare an **R** Markdown document showing a number of tests to check your code.
- This is a take-home quiz due Tuesday, March 13 at 11:59pm.
- Submit your quiz on LEARN as follows:
  1. Submit three (3) files named **`uwname-quiz4.R`**, **`uwname-quiz4.Rmd`**, and **`uwname-quiz4.pdf`**, where **`uwname`** is your UW username. The files should contain (1) all the functions you are asked to write (2) the source code for the **R** Markdown document displaying your test code and (3) the compiled PDF version of your Markdown document.
  2. To upload these files to LEARN, navigate to **`Assessments/Dropbox`**, then click on **`Quiz 4`**.
- You may work on this quiz in groups. However, you **must include the names of all collaborators** in the `Comments` field during the LEARN file upload process.
- Organized, well-commented, and efficient code is required for full marks.
- Clarity and proper formating of the report and included figures are required for full marks.

Let $Y > 0$ denote the survival time of a patient after a given medical procedure. A popular model for Y is the Generalized Gamma Distribution[1]. A random variable Y is said to follow a Generalized Gamma distribution with shape, rate, and power parameters $\alpha, \beta, \lambda > 0$ if it is the power of a Gamma random variable:

$$Y \sim \text{Gen-Gamma}(\alpha, \beta, \lambda) \qquad \Longleftrightarrow \qquad Y = \beta^{-1} X^{1/\lambda}, \quad X \sim \text{Gamma}(\alpha/\lambda, 1). \tag{1}$$

The PDF of the Generalized Gamma distribution is

$$f(y \mid \alpha, \beta, \lambda) = \frac{\lambda \beta^{\alpha}}{\Gamma(\alpha/\lambda)} y^{\alpha-1} \exp\{-(\beta y)^{\lambda}\}.$$

The PDF, CDF, inverse-CDF, and a random number generator (i.e., the `d/p/q/r` functions) for the Generalized Gamma distribution are provided in the file **`ggamma-dpqr_functions.R`**.

Let $\boldsymbol{Y} = (Y_1, \ldots, Y_n) \overset{\text{iid}}{\sim} \text{Gen-Gamma}(\alpha, \beta, \lambda)$. In this Quiz, you are asked to write an MCMC sampler for the posterior distribution $p(\alpha, \beta, \lambda \mid \boldsymbol{Y})$ as described below.

[1] **Note:** The parametrization used in this Quiz is slighly different from that in the link.

Q1. Reparametrize model (1) with $\eta = \beta^\lambda$, and show that the loglikelihood in this parametrization is

$$\ell(\alpha, \eta, \lambda \,|\, \boldsymbol{Y}) = n \cdot \Big[\log \lambda - \log \Gamma(\alpha\lambda^{-1}) + \alpha\lambda^{-1} \log \eta \Big] + \alpha S - \eta T_\lambda, \tag{2}$$

where $S = \sum_{i=1}^n \log Y_i$ and $T_\lambda = \sum_{i=1}^n Y_i^\lambda$. That is, write an **R** function

```
ggamma.loglik <- function(alpha, eta, lambda, logy)
```

and verify that for fixed $\boldsymbol{Y}$ and different parameter vectors (α, η, λ), the simplified likelihood (2) and the unsimplified version $\ell(\alpha, \eta, \lambda \,|\, \boldsymbol{Y}) = \sum_{i=1}^n \log f(Y_i \,|\, \alpha, \eta, \lambda)$ differ only by a constant.

Note: It is roughly twice as efficient to precompute $\log(\boldsymbol{Y})$ and pass it as an argument (`logy`) to `ggamma.loglik`, than to pass in the data on the original scale. This is because the calculation of T_λ involves calculating each power Y_i^λ, which most programming languages implement as $\exp(\lambda \log Y_i)$. Since `exp` and `log` are relatively expensive operations which dominate the computations of the loglikelihood, precomputing $\log(\boldsymbol{Y})$ cuts the computation time roughly in half. A similar technique was employed in the file **`weibull-functions.R`**.

Q2. For Bayesian inference, suppose the prior distribution is of the form $\pi(\alpha, \eta, \lambda) = g(\alpha, \lambda)$, i.e., the prior is flat on η. Show that the conditional distribution

$$p(\eta \,|\, \alpha, \lambda, \boldsymbol{Y}) = \frac{p(\eta, \alpha, \lambda \,|\, \boldsymbol{Y})}{p(\alpha, \lambda \,|\, \boldsymbol{Y})} \propto p(\eta, \alpha, \lambda \,|\, \boldsymbol{Y}) \propto \mathcal{L}(\alpha, \eta, \lambda \,|\, \boldsymbol{Y}) \cdot g(\alpha, \lambda) \tag{3}$$

is a Gamma distribution:

$$\eta \,|\, \alpha, \lambda, \boldsymbol{Y} \sim \text{Gamma}(\hat{\kappa}, \hat{\gamma}) \qquad \Longleftrightarrow \qquad f(\eta \,|\, \alpha, \lambda, \boldsymbol{Y}) \propto \eta^{\hat{\kappa}-1} \cdot \exp\{-\hat{\gamma}\eta\},$$

where $\hat{\kappa}$ and $\hat{\gamma}$ are functions of α, λ, and $\boldsymbol{Y}$. To find the expressions for $\hat{\kappa}$ and $\hat{\gamma}$, throw out everything in (3) that doesn't depend on η until you recognize the unnormalized form of the Gamma distribution above.

Q3. Write an **R** function

```
ggamma.logmarg <- function(alpha, lambda, logy, Tlambda.out = FALSE)
```

which calculates the marginal posterior distribution of (α, λ) up to the factor of $g(\alpha, \lambda)$.

In other words, `ggamma.logmarg` returns the log of

$$\frac{p(\alpha, \lambda \mid \boldsymbol{Y})}{g(\alpha, \lambda)} \propto \frac{\mathcal{L}(\alpha, \eta, \lambda \mid \boldsymbol{Y})}{p(\eta \mid \alpha, \lambda, \boldsymbol{Y})} = \frac{\mathcal{L}(\alpha, \eta, \lambda \mid \boldsymbol{Y})}{\texttt{dgamma}(\texttt{x} = \eta, \texttt{shape} = \hat{\kappa}, \texttt{rate} = \hat{\gamma})},$$

with the expressions for $\hat{\kappa}$ and $\hat{\gamma}$ obtained in Q2.

Write the function `ggamma.logmarg` as efficiently as possible, as it will be called repeatedly in the MCMC algorithm defined below. In fact, the function should optionally return a length-two vector of both the log-PDF and $T_\lambda = \sum_{i=1}^n Y_i^\lambda$ (if `Tlambda.out = TRUE`) – an important precomputation to be used in the upcoming MCMC.

For fixed $\boldsymbol{Y}$ and different (α, η, λ), verify that the marginal posterior calculation is correct by checking that

```
ggamma.loglik(alpha, eta, lambda, logy) -              # joint distribution
  (ggamma.logmarg(alpha, lambda, logy) +               # marginal distribution
   dgamma(eta, shape = khat, rate = ghat, log = TRUE)) # conditional distribution
```

always returns the same constant.

Q4. Write an **R** function which implements a Collapsed Metropolis-Within-Gibbs MCMC algorithm to sample from the posterior distribution $p(\alpha, \eta, \lambda \mid \boldsymbol{Y})$ using the following guidelines:

★ The function signature must be in this format:

```
#' @title MCMC sampler for the Generalized Gamma posterior parameter distribution.
#'
#' @description Implements a Collapsed Metropolis-within-Gibbs sampler on the marginal
#' distribution \code{p(alpha, lambda | y)}, with analytic draws for the conditional
#' distribution \code{p(eta | alpha, lambda, y)}.
#'
#' @param nsamples Number of posterior iterations excluding burn-in.
#' @param y Vector of survival times on the regular scale (i.e., positive).
#' @param alpha0,lambda0 Initial values for parameters \code{alpha} and \code{lambda}
#' to start MCMC algorithm.
#' @param burn Number of burn-in samples.
#' @param mwg.sd Length-two vector giving the random walk jump size for
#' \code{alpha} and \code{lambda}.
#' @param acc.out Logical; whether or not to output the acceptance rate for each parameter.
#' @return A \code{nsamples x 3} matrix containing the posterior draws,
#' and optionally the acceptance rates if \code{acc.out = TRUE}.
ggamma.post <- function(nsamples, y, alpha0, lambda0, burn, mwg.sd, acc.out = FALSE)
```

Much of your code will be very similar to the examples in **`weibull-functions.R`** and **`weibull-mcmc.R`**.

★ Expanding on the `@description` in the function documentation above, let $\boldsymbol{\theta}^{(t)} = (\alpha^{(t)}, \eta^{(t)}, \lambda^{(t)})$ denote the state of the collapsed MWG sampler at step t. In order to get

to step $t+1$:

(1) Completely ignore $\eta^{(t)}$, and do a pair of MWG updates targeting the marginal distribution $p(\alpha, \lambda \,|\, \boldsymbol{Y})$:

$$\alpha^{(t+1)} \sim \text{MWG}(\alpha \,|\, \lambda^{(t)}),$$
$$\lambda^{(t+1)} \sim \text{MWG}(\lambda \,|\, \alpha^{(t+1)}).$$

(2) Once you have $(\alpha^{(t+1)}, \lambda^{(t+1)})$, calculate $(\hat{\kappa}^{(t+1)}, \hat{\gamma}^{(t+1)})$ as in Q2 and perform an exact conditional draw

$$\eta^{(t+1)} \sim p(\eta \,|\, \alpha^{(t+1)}, \lambda^{(t+1)}, \boldsymbol{Y}) \quad \Longleftrightarrow \quad \eta^{(t+1)} \,|\, \alpha^{(t+1)}, \lambda^{(t+1)}, \boldsymbol{Y} \sim \text{Gamma}(\hat{\kappa}^{(t+1)}, \hat{\gamma}^{(t+1)}).$$

Note: You can perform the conditional draws in **(2)** either at each step of the MWG sampler on $p(\alpha, \lambda \,|\, \boldsymbol{Y})$, or after all M samples $(\alpha^{(1)}, \lambda^{(1)}), \ldots, (\alpha^{(M)}, \lambda^{(M)})$ have been obtained, as in function `weibull.mmi` of **`weibull-functions.R`**. The former is slighly easier to program, but the latter is slightly more efficient, as draw from the conditional Gamma distribution $p(\eta \,|\, \alpha, \lambda, \boldsymbol{Y})$ can be vectorized over all pairs $(\hat{\kappa}^{(1)}, \hat{\gamma}^{(1)}), \ldots, (\hat{\kappa}^{(M)}, \hat{\gamma}^{(M)})$.

$\star$ Use the prior distribution which is flat on η and with $\log(\alpha), \log(\lambda) \overset{\text{iid}}{\sim} \mathcal{N}(0, 2)$. In other words, the prior distribution to use in your MWG sampler is

```
# marginal prior distribution for (alpha, lambda)
logprior <- function(alpha, lambda) {
  dlnorm(alpha, meanlog = 0, sdlog = 2, log = TRUE) +
    dlnorm(lambda, meanlog = 0, sdlog = 2, log = TRUE)
}
```

Q5. Perform a numerical verification of your MCMC algorithm as follows.

1. Simulate $n = 500$ observations from a Generalized Gamma distribution with parameters $\alpha = 2.3$, $\beta = 1.7$ (not η!), and $\lambda = 0.26$.

2. Using `ggamma.logmarg`, calculate $p(\alpha, \lambda \,|\, \boldsymbol{Y})$ on a fine 2D grid, and sum over rows or columns to obtain "analytic" marginal distributions $p(\alpha \,|\, \boldsymbol{Y})$ and $p(\lambda \,|\, \boldsymbol{Y})$.

3. Do a couple of short runs of the MCMC (e.g., $M \approx 500-1000$ iterations) to manually tune the MWG jump sizes to get the magic acceptance rate of 45% in each coordinate.

4. Once the MCMC is properly tuned, run it for $M = 200,000$ iterations.

5. Plot the histograms of the MCMC output for α and λ, and superimpose on them the corresponding analytic marginal posteriors obtained by the grid method.

Note: As we saw in class, the grid limits needs to be manually adjusted to capture the range of the posterior distribution $p(\alpha, \lambda \,|\, \boldsymbol{Y})$. However, **R** Markdown will generate a new random dataset every time you recompile the document, thus requiring you to reset the range each time. In order to prevent this, set the seed of **R**'s pseudo-random number generator so that it produces the same dataset $\boldsymbol{Y}$ on every run. To do this, simply include the command

```
set.seed(2018) # generate the same dataset on every run
```

right before generating your dataset[2].

Bonus Question: Use the MCMC output from Q5 to plot the histogram, posterior mean, and 95% credible interval for the Bayesian estimate of median of the survival distribution:

$$\tau = \tau(\alpha, \eta, \lambda) = F_Y^{-1}(.5 \,|\, \alpha, \eta, \lambda).$$

Hint: See the `qggamma` function in **`weibull-dpqr_functions.R`**.

[2]You may also wish to look into the caching option of code chunks containing lengthy calculations. However, in my experience this doesn't always work as expected...if you run into problems consider deleting the `cache` directory created by your `Rmd` file.