problem Set no. 2 — due Tuesday 10/7 at midnight

1 Evaluating frequency properties of Bayesian methods

Many Bayesian methods involve large amounts of computation and are difficult to evaluate analytically (e.g. models requiring MCMC sampling or other approximation techniques); however, frequency characteristics of these methods remain of interest. In particular, the frequency coverage of Bayesian interval estimates is often of interest to both practitioners and reviewers when examining new Bayesian methods. Without large-scale computation, such questions are difficult to address, but, with the availability of clusters like Odyssey, it is possible to evaluate some frequency properties of these methods through simulation.

In this problem set, you will work with a non-conjugate hierarchical Bayesian model. A MCMC algorithm to draw from its posterior is provided. Your task will be to evaluate this method via simulation using Odyssey and interpret your results.

2 Model

We will be using a simple, non-conjugate hierarchical model known as the log-Normal / Poisson model. It can be summarized as:

$$\begin{aligned} p(\mu, \sigma^2) &\propto 1/\sigma^2 \\ \log\theta_j &\sim \mathrm{N}(\mu, \sigma^2),\ j = 1, \cdots, J \\ Y_{jn} &\sim \mathrm{Pois}(w_j\theta_j),\ j = 1, \cdots, J;\ n = 1, \cdots, N \end{aligned}$$

In this model, we assume that each unit j has a basic intensity θ_j drawn from a common distribution. A set of N Poisson observations is then drawn for each unit, each of which has expectation $w_j\theta_j$; w_j is a constant, known exposure weight.

Models with this basic structure arise frequently in the analysis of astronomical, ecological, and medical data. One example of the former would be the `bayesstack` software for the analysis of x-ray data from the Chandra space telescope[1]. An ecological application can be found in, for example, Bulmer (1974)[2].

[1] Available at http://hea-www.cfa.harvard.edu/CHAMP/SOFT/bayesstack/

[2] Available at http://www.jstor.org.ezp-prod1.hul.harvard.edu/stable/2529621?seq=1

3 Provided Code & Data

You will find the file `poissonLogN_MCMC.R` in the `src` directory of this problem set. It contains the key function for your evaluations, `poisson.logn.mcmc`, as well as a set of functions used by this algorithm.

The function `poisson.logn.mcmc` takes 2 variables as required inputs. The first is `Y`, a $J \times N$ matrix of integer-valued observations. The second is `w`, a J-dimensional vector of exposure weights. This function also takes initial values `mu0` and `sigmasq0`; the function initializes θ internally. The last two parameters (`ndraws` and `burnin`) should be left at their default values.

The `poisson.logn.mcmc` function returns a list with several items. The most important for your purposes will be `logLambda`, which is a $J \times$ `ndraws` matrix for which each column contains a draw from the posterior of $\log \vec{\theta}$.

We have also provided the data file `weights.txt` consisting of 1000 exposure weights for use in tasks 4 and 5.

Finally, for use in task 5, we have provided the file `rASL.R` in the `src` directory of this problem set. It contains the the function `rASL`, which takes four arguments and returns draws from an asymmetric Laplace distribution. The first is `n`, the number of variates to draw. The remainder (`x0`, `b`, and `m`) are parameters for the asymmetric Laplace distribution used here. They can be understood via the representation:

$$\begin{aligned} Y &\sim x_0 + m\,X + \sqrt{b\,X}Z \\ X &\sim \text{Expo} \\ Z &\sim \text{N}(0,1) \\ X &\perp Z \end{aligned}$$

Here, $Y \sim \text{ASL}(x_0, m, b)$. The function `rASL` returns a vector of length `n` random variates, with the same structures as, for example, `rnorm`.

4 Tasks

4.1 Task 1: Examine the posterior (15 points)

Write out the joint posterior for μ, σ^2, and $\log \vec{\theta}$ for the model given in section 2 (ignoring normalizing constants). Write out the conditional posterior of $\log \theta_j$ given Y, $\vec{w}$, μ, and σ^2. Is this conditional posterior unimodal? Log-concave?

4.2 Task 2: Write functions to simulate data from the model (5 points)

Write the function `simYgivenTheta(theta, w, N)` to simulate data from the model given in section 2. Its required form is detailed in section 5.

4.3 Task 3: Evaluate coverage for a simple case (20 points)

You will design and execute simulations of data from the model to evaluate coverage for different parameter settings, using a two-stage design. First, you will first draw $\vec{\theta}$ from the given generative distribution. Then, you will repeatedly draw $Y \mid \vec{\theta}$ for a set of simulations. For example, if you are running 1000 simulations, you might choose to draw 40 unique $\vec{\theta}$'s, then draw 25 values of Y for each unique $\vec{\theta}$. This is necessary to estimate the coverage of posterior intervals for each value of θ_j, which would be quite difficult without replication.

For each of the following parameter settings, you will run a set of simulations, fixing w_j at 1 for all j, J at 1000, and N at 2.

μ	σ^2
1.6	0.7^2
2.5	1.3^2
5.2	1.3^2
4.9	1.6^2

For this and subsequent tasks, you should design your runs to use 12 Odyssey nodes for 1 hour (43200 seconds of total CPU time). This bound is per task (including all parameter values), but does not include post-processing, plotting, etc.

First, layout a design for these simulations with a structure analogous to that discussed above, running as many simulations as possible in the alloted time. This will require some testing to estimate the runtime of the MCMC algorithm. Your design should include runtime estimates and target sample sizes.

Second, run simulations according to your design, obtaining a posterior mean and standard deviation for each $\log \theta_j$ from each simulated dataset. You should also retain the actual value of each $\log \vec{\theta}$ as a basis for subsequent estimation of coverage. Also report the actual runtime for your simulations in your write-up.

Third, for each set of parameter values, obtain approximate 68% and 95% posterior intervals for each θ_j using the MCMC output. Estimate their frequency coverage (using the replication provided by the given type of design) and plot estimated coverage against the actual value of $\log \theta_j$. Details on desired output are included in section 6.

4.4 Task 4: Evaluate coverage with exposure weights (20 points)

We now add another layer of complexity. A set of exposure weights has been provided in the file `weights.txt` in the `dat` folder. For each set of parameter values from the previous task, simulate data with the new exposure weights. Maintain J at 1000 and N at 2; you should use the design laid out in task 3.

Perform a similar coverage evaluation, plotting estimated coverage against $\log \theta_j$ and $\log w_j$ for each set of parameter values. Details on desired output are included in section 6.

4.5 Task 5: Evaluate coverage with model misspecification (20 points)

Using the function `rASL.R`, simulate $\log \vec{\theta}$ using the following parameter values:

x_0	m	b
1.6	0	1.3
1.6	−0.7	1.3
1.6	0.7	1.3
1.6	0	2.6

Using your design from task 3 and the exposure weights from task 4, run a set of simulations to evaluate the coverage of the given inference method.

Produce plots analogous to those in tasks 4 for each set of parameter values. Details on desired output are included in section 6.

4.6 Task 6: Interpret & discuss results (20 points)

Interpret your results from tasks 3, 4, and 5. In particular, discuss the relationship between $\log \theta_j$, $\log w_j$, and coverage probabilities. For task 5, discuss how deviations from model assumptions affect the reliability of estimates based on the given model.

5 Code & data standards

Your solution should be submitted as a zip file `myfasusername_ps2.zip` (not `RAR` or another format) containing only the requested files (no sub-folders), where `myfasusername` is replaced by your username.

First, all functions for your work should be included in a file named `myfasusername_ps2_functions.R`. Your remaining code should call these functions, loading them via:

```
source("myfasusername_ps2_functions.R")
```

For tasks 3, 4, and 5, you should produce three files. For example, for task 3, you should have:

- `myfasusername_ps2_task3.R`
- `myfasusername_ps2_task3.slurm`
- `myfasusername_ps2_task3_post.R`

The first of these will contain your simulation code, the second will be you `slurm` script for the given set of simulations, and the last will contain any post-processing (e.g. plotting) you need to do.

All of your code should be designed to run in an arbitrary working directory containing the necessary files and subdirectories; do not use hard-coded file paths. Your code should contain default settings if no arguments (e.g. job indices) are supplied that will produce test output.

You should organize results and diagnostic output from Odyssey using two subdirectories. Raw output should be placed in a `out/` subdirectory of whichever working directory your code is running in. Diagnostic output from `slurm` should be sent to a `dump/` subdirectory, as demonstrated in the Odyssey user documentation. Do not include these subdirectories in your submission.

You should, however, include condensed versions of your coverage results in your submission. For each combination of task and parameter values, your post-processing code should produce files named as:

```
myfasusername_ps2_task4_par1_theta.dat
myfasusername_ps2_task4_par1_w.dat
```

changing the `task` and `par` indices as appropriate. The former file should contain a table with two columns: $\log \theta_j$ and the estimated coverage probability for each value. The latter (which should only be produced for tasks 4 and 5) should contain a column for $\log w_j$ and a column of the estimated coverage probability for each value. In total, you should have 5 such files: 1 for task 3, and 2 each for tasks 4 and 5. These should be placed in the root of your `myfasusername_ps2.zip` submission.

All plots should be named as myfasusername_ps2_task3_plot1.png, with task and plot indices changed as necessary. They should be included in the root of your zip submission.

Your simYgivenTheta(theta, w, N) function should take as input vectors of length J theta and w of intensities and exposure weights, respectively, along with the scalar N as discussed in section 2. It should return a matrix Y of the dimensions and forms discussed in the previously referenced section.

6 Write-up standards

You will submit a write-up in both source (LaTeX or Word) and PDF form. The former should be named myfasusername_ps2.tex or myfasusername_ps2.doc, and the latter should be named myfasusername_ps2.pdf. Please submit to stat221.harvard.fall2014@gmail.com including the keyword "stat221" in the subject.

The write-up should contain a section for each task with your responses. You should include plots in a separate appendix (titled "Figures"), referencing them from the body of the text.

For task 3, produce 4 plots (one for each set of parameter values). Each plot should include a scatterplot of coverage against $\log \theta_j$ with a smooth regression line (e.g. a LOESS or kernel smoother) through the dataset.

For each of tasks 4 and 5, produce 8 plots. The first 4 should be analogous to those for task 3; the other 4 should include a scatterplot of coverage against $\log w_j$ with a similar smooth regression line through the dataset.

In the body of your text, you should include answers for task 1 and your design from task 3 (including runtime estimates). Your answer to task 6 should require no more than a page of LaTeX, single-spaced with the fullpage package.

Best of luck and good coding!