problem set no. 4 — due wednesday 11/9 before class starts

problem 1. the task is to implement the model in Tebaldi & West (JASA, 1998).

goals. the goals of this pset are for you to get familiar with derivations and implementation issues of MCMC algorithms that arise in the context of an ill-posed inverse problem, to become further acquainted with convergence diagnostics, to see an example of informative prior distributions and their impact, and to design a simple framework for simultaneous MCMC simulations across many datasets.

problem background. in a communication network, routers and switches connect subnetworks of users. we can measure traffic (packet counts) on ingoing and outgoing links at each router, every five minutes. these measurements are referred to as *link loads*. we want to infer the traffic on all the possible origin-destination (OD) routes, every 5 minutes. these non-observable quantities of interest are referred to as *OD flows*. in this setting, the number of link loads at each time t is smaller than the number of corresponding OD flows.

question 1.1 (20 point) implement the MCMC algorithm described in section 3.1 of Tebaldi & West. see the specifications at the end of the problem set for more details on the implementation.

question 1.2 (15 point) recall the router1 data from pset 4 – in this problem you are asked to fit the model of Tebaldi & West to the data from time point 5 only. you should use a uniform prior for the λ's and run 10 chains to perform the analysis. results from the 10 chains should be combined to produce all plots and summary statistics as follows:

- Compute a selection of MCMC convergence diagnostics, possibly numeric or graphical, and briefly discuss the results. Examples include autocorrelations (`acf`), effective sample-size (`effectiveSize`), and Gelman-Rubin statistic (`gelman.diag`)).
- Plot marginal posterior histograms of the x's on the same plot (the same format as Tebaldi & West's Figure 5)
- Plot marginal posterior densities of the λ's (it is recommend to use `densplot` in `library(coda)` to produce these)

Note: This job does not need to be run on Odyssey (although it is fine if you do).

question 1.3 (15 point) for the router1 data, time point 5, we now investigate the impact of the choice of prior distribution. perform the MCMC analysis as in question 1.1, but this time use an informative informative in the style of Tebaldi & West. To mimic the

online-updating, as described at the bottom of pg 566, you should use the following priors:

$$\lambda_i \sim \text{Gamma}\left(aX_i^{(t-1)}, a\right), \tag{1}$$

where the prior centered at the known true value from the previous time-point (time-point 4 in this case), but the variance is inflated by a factor of $1/a$ to stop the prior having too much influence. As in Tebaldi & West, set $a = 0.02$ (thus inflating the prior variance by a factor of 50). again, run 10 chains to perform the analysis and combine results. produce the output as above.

Note: This job does not need to be run on Odyssey (although it is fine if you do).

question 1.4 (10 point) Produce a side-by-side boxplot of the marginal posterior densities using the uniform and the informative priors, as in Figure 11 of Tebaldi & West. Comment on any differences you see, and explain possible reasons for these differences.

question 1.5 (15 point) Now, we apply the model of Tebaldi & West to each time-point (excluding time 1) of router 1 dataset using a uniform prior for the λ's. The analysis for each time-point does not depend on the other times. For each time-point, after the MCMC analysis has completed, you will need to:

1. Compute the marginal posterior median, 2.5th and 97.5th percentile for all of the x's and λ's. (this can be done immediately by computing `summary(foo)$quantiles` for an `mcmc` object). Write these values to individual files and then combine them (you may like the `Unix` command `cat`).
2. Plot the posterior 95% interval for the x's and λ's over time.

Note that this problem should only require a very small modification of your existing code to run the analysis (a few more lines to do the plots). Each time-point should be executed on a different Odyssey job (or subjob). Again, run 10 chains for each time-point – these can either be run within the same Odyssey job, or split up into more subjobs (it is recommend to do the 10 chains sequentially within the same job). All jobs must go to either the `short_serial` or `normal_serial` queues.

question 1.6 (15 point) Finally, use the true values for the previous time-points to form informative priors for the model as in equation (1). Run the same analysis as in question 1.4, the only change being the prior parameters. Again, this should require only trivial modification to the code you have already written. again, produce the 95% intervals plotted against time.

question 1.7 (10 point) Compare the posterior intervals over time with and without the informative prior. Again, discuss any differences and provide plausible explanations for them. Over the last two problem sets, which approach do you like the most?

Note: Part of the analysis you will perform for Questions 1.4 and 1.5 is identical to what you did for Questions 1.2 and 1.3, so this pset essentially boils down to 1.4 and 1.5 only. However, we recommend that you test and validate your code for single time points first (you could also use the toy example of section 3.2 to do this).

question 1.8 (extra credit) apply the model of Tebaldi & West to the link loads in file `2router_linkcount.dat`. you may use any prior for this problem. compare in a figure: these estimates, the estimates for x_t you get with your favorite model variant implemented for pset 4. again, discuss what you see and why.

* * *

what to submit. submit 1 zip file. the zip file should contain: the `R` and `lsf` scripts to run your MCMC computations on Odyssey (7-12 files); the figures described; a latex (or word) document with your answers to the questions (1 file).

please read the following instructions carefully and make sure your submission conforms to the guidelines. this is a computing exercise; if your code does not run it's a problem. any submitted homeworks that do not conform to the specifications will lose points.

please submit written answers (using latex or word) to all relevant questions on Canvas before 2:35 pm on Wednesday 11/9. Upload a separate PDF file named as `myfasusername_ps4.pdf` and a ZIP file named as `myfasusername_ps4.ZIP` to Canvas (where `myfasusername` is replaced by your actual FAS username). the ZIP file should contain three `R` scripts:

1. `myfasusername_mcmc.R`,
2. `myfasusername_1router_t5_uniform.R`,
3. `myfasusername_1router_t5_informative.R`,
4. `myfasusername_1router_all_times_uniform.R`,
5. `myfasusername_1router_all_times_informative.R`, and,
6. `myfasusername_2router.R`, (optional)

and your SLURM scripts:

7. `myfasusername_1router_t5_uniform.slurm`, (optional)
8. `myfasusername_1router_t5_informative.slurm`, (optional)
9. `myfasusername_1router_all_times_uniform.slurm`,
10. `myfasusername_1router_all_times_informative.slurm`, and,
11. `myfasusername_2router.slurm`. (optional)

your scripts must run on Odyssey and produce the plots as described above, **as always, assuming all input/output files are in the same folder**.

- the file `myfasusername_mcmc.R` should contain the function used to fit the MCMC algorithm to all datasets. this function must be named and have the following required arguments (although extra arguments are permitted):

  ```
  network_mcmc <- function(Y,A,prior,iter=1.2e5,burnin=2e4,verbose=FALSE){
    # Write the function
  }
  ```

  If switched on, the `verbose` argument should print out useful progress updates to help you debug your algorithm. Make sure to write this into your code – with comments – as you go, not just when it breaks!

- The application to each dataset (or time-point) should just involve some initial code to set-up the model inputs, then a simple call to `network_mcmc`, and finally, some additional lines of code to plot the graphs.

- You may want to consider returning your output from `network_mcmc` directly as an `mcmc` object, using `library(coda)`, although this is not required.

- the file `myfasusername_mcmc.R` should be loaded in using:

  ```
  source('myfasusername_mcmc.R')
  ```

- the figures should be named after the corresponding dataset and prior choice as follows:

  1. `myfasusername_1router_t5_uniform_*.pdf`,
  2. `myfasusername_1router_t5_informative_*.pdf`,
  3. `myfasusername_1router_all_times_uniform_*.pdf`,

4. `myfasusername_1router_all_times_informative_*.pdf`,
5. `myfasusername_2router_*.pdf` (extra credit only).

the `*` denotes a `Unix` wildcard i.e., you can put any text you like wherever you see one.

- the TEX or DOC file with the solution, named as `myfasusername_ps4.tex` or `myfasusername_ps4.doc`

as always, the latex source of this pset is provided for your convenience. if you have any questions about code formatting and organization then please ask Zhirui. happy coding!