Stan and JAGS tricks

**JAGS**

1. JAGS using step and equals, for controlling
2. Storing values in a matrix, then summing these to handle cumulative sums:

for (i in 1:NTotal) {

for (j in 1:30)

{

for (k in 1:36)

{

A[i,j,k] <- equals(k,d[villageID[i]])\*step(Tp[villageID[i]]-j)\*(Tp[villageID[i]]-j+1)\*rain[captureDays[i]-j+1-d[villageID[i]]]

}

}

captureNumber[i] ~ dpois(nu[i])

nu[i] <- 0.04\*roomsSampled[villageID[i]]\*(lambda[villageID[i]]\*(2/(Tp[villageID[i]]\*(Tp[villageID[i]]+1)))\*sum(A[i,,])

+ nBase[villageID[i]])

}

1. How to make a discrete distribution over any support using the dcat(), an index variable and a probability density:

for (j in 1:NVillages){

indD[j] ~ dcat(punif[])

d[j] <- xD[indD[j]]

}

for (k in 1:35)

{

punif[k] <- 1

xD[k] <- k + 4

}

1. Illustrate what is actually meant by a for loop, given that it is not a procedural language.

**Stan**

1. Generated quantities part of code to sample parameters, things of interest.
2. Log\_sum\_exp explanation for marginalising parameters: 1D from manual, and 2D from FourVillagesExample.
3. Increment\_log\_prob: illustrate bias with simulated data.
4. Order of parameter declaration. It seems to be that you can only define parameters in blocks.
5. Show how to generate samples from any distribution (without any data).
6. Supply priors information/anything you want to change frequently in the data. This saves the Stan model having to be recompiled each time.
7. Supply functions for diagnosis: looking at the nDivergent, nLeapFrog and nTreeDepth
8. Show how to carry out PPCs via generated quantities
9. Include the table in ‘*stan-blocks-good-reference’* paper, which shows the locality of variables, and the number of times each block is called.
10. Include function to find cases (for meta-data) where the p values are less than 0.05, or greater than 0.95. (Perhaps adjust these to allow the user to specify the percentage).
11. Not so much a Stan point, but should illustrate PPCs for time series. Use Gelman’s dogs paper as a reference.
12. Centralised parameterisations vs non-central for speeding up Stan simulations. Reference Neals’s funnel and the paper Pppppss (Greek name). Chapter 20 in Stan manual.
13. How to specify random initial conditions for samplers.
14. Show how to use adapt\_delta to adjust the step size, and adapt\_stepsize as per Bob’s comment.
15. Show how to sample fake datasets properly using ‘generated quantities’ section. Look at the various Stan BUGS examples that I have translated.
16. Graphical checks of fake vs actual data.
17. Supply functions that do all the diagnostic checks on your model. Add suggestions to data if they fail tests.
18. Can I create a python function which converts a Stan sampling file into a fake data simulated file?
19. Mention not using Cauchy->Cauchy hyperparameters.
20. When creating fake data ensure that it looks at least a bit like the actual! CF Week spent looking at estimates of fake data for MRR data, where the between species variation was not great compared to within!
21. MLE estimation of quantities beforehand.
22. Make sure you understand what is meant by Stan error messages. For months I thought that the scale parameter of the NB was the kappa term, and couldn’t understand why I was getting a 0:0 error!
23. Multivariate t – only use for likelihood, not for prior (see <https://groups.google.com/forum/?fromgroups=#!topic/stan-users/0F0O4hfHA8g>)
24. Creating identity matrix – again see <https://groups.google.com/forum/?fromgroups=#!topic/stan-users/0F0O4hfHA8g>, although this makes the case that they aren’t needed! Maybe just make a point to say that we never need identity matrices in Stan.
25. Test model using iter=10-100 to see: is the output correct? How long will the model roughly take to run?
26. How to calculate WAIC in Stan? In other words how to do this by collecting log likelihood of all points, then using ‘loo’ R package,