Work Log for September

Logan Brown

October 1, 2014

5 Week of September 29th - October 1st

5.1 Goals for the Week

1. What I know from Last Week (kept here for ease of access)
2. Find out what causes some of the probabilties to go outrageously high
3. Compare Min, Max, and Average Values
4. Use Sections of the Simulated Yeast Genome
5. If possible, NSE Patch
6. More information on Codon Position
7. Look into an R Vim extension or RStudio
8. Debugging Thoughts

5.2 Progress/Notes

5.2.1 What I know from Last Week (kept here for ease of access)

Here's what I know about the NaN error

1. ~~one~~ One or more of the elements of lpProp is going to NaN.
2. It is not always the same element. For my first run, it was zraS. The second was ecnA.
3. It is not because log(0)= $-\infty$. Multiple elements are going to -Inf (106 in the first run, 87 in the second run). Moreover, the roc code also tends to generate -Inf values (though not nearly as many in each run, only 1 or 2).
4. It doesn't appear that the scale is going out of control. Scale and acceptance rate stay similar to the values used in the ROC model.

5. It happens in cubfits and cubappr

6. It is not just happening for one amino acid. For the first run, it happened in Amino Acid 5 (F). In the run, it happened in both 8 and 9 (I and K). The third run happened in 11 (N). In the fourth run, it happened in 3 and 8 (D and I). In the fifth, it was 4 (E).

7. It is lp.vec, the return from my.inverse.mlogit.r

8. my.inverse.mlogit passes NON NaN values (though they are stupidly large like 1.452498e+18 instead of -0.5610390) to invmlogit, and it returns NaN values.

9. The code gets stuck in the following loop

   ```
   if(tmp_exp == HUGE_VAL || tmp_exp == 0.0){
   *flag_out_range = 1;
   *scale_exp = (tmp_exp == HUGE_VAL) ? max_exp : -max_exp;
   do{
   *scale_exp *= 0.5;
   tmp_exp = exp(*scale_exp);
   } while(tmp_exp == HUGE_VAL);
   *scale_exp = max_exp - *scale_exp;
   }
   ```

   That's what causes the slow down.

   But this means the problem comes earlier. At some point in the code, some probabilities are going to infinity, which causes the HUGE_VALUE loop (and the slowdown), and eventually causes NaN values.

   Cedric suggested that it may be caused by the covariance matrix exploding.

10. ~~All the values~~ The average values for lpProp are generally too low. mean(lpProp[is.finite(lpProp)]) returns -588.3597 in from the first run, and -554.7103 in the second run.

11. The C code sometimes FIXES the problem! When doing a second run with seed 83455, lp.vec has a NaN value at position 22893 (amino acid???)

12. The Phi values for codons with -Inf log probability are absurdly high.

   ```
   Browse[2]> mean(phi[is.finite(lp.c.raw)])
   [1] 15240.47
   Browse[2]> mean(phi[!is.finite(lp.c.raw)])
   [1] 3.256039e+20
   Browse[2]> testmat <- cbind(testvec[!is.finite(lp.c.raw)], lp.c.raw[!is.finite(lp.c.
   Browse[2]> colnames(testmat) <- c("#", "lp.c.raw", "phi", "naa")
   ```

```
Browse[2]> testmat
        # lp.c.raw          phi naa
cnu   194     -Inf 1.041789e+07   4
csrD  226     -Inf 1.091396e+04  18
frmR  467     -Inf 1.756417e+08   4
malQ  829     -Inf 1.057376e+04  36
moaC  915     -Inf 1.814127e+05  10
mqsR  934     -Inf 4.044010e+06   7
rsxA 1433     -Inf 6.201766e+04   5
shoB 1485      NaN 4.558455e+21   2
tesB 1551     -Inf 1.629425e+05  10
ybfA 1738     -Inf 8.534363e+05   4
ycfJ 1801     -Inf 3.949059e+04  10
yciB 1818     -Inf 2.038530e+05  12
ygeR 1993     -Inf 6.640161e+04  20
yhaJ 2020     -Inf 2.098447e+05  14

Browse[2]> yaa[!is.finite(lp.c.raw),]
     AAA AAG
cnu    3   1
csrD  14   4
frmR   2   2
malQ  24  12
moaC   7   3
mqsR   5   2
rsxA   3   2
shoB   2   0
tesB   8   2
ybfA   3   1
ycfJ   7   3
yciB   5   7
ygeR  17   3
yhaJ  13   1
```

5.2.2 Find out what causes some of the probabilties to go outrageously high

Theories:

One value is going to infinity, for its own reasons. This causes the other values to drop in response, which explains the drop in the average lpProp values. Through some means (addition of the bias? Seems unlikely, logdmultinomCodAllR returns NaN values before the bias comes into effect), I think one of the probabilities goes above 1. Then the odds ratio $\frac{p_{\vec{c}_{ij}}}{1-p_{\vec{c}_{ij}}} > 1$, which causes the probabilites to EXPLODE. But, it stays under a cap, because stable_exp.c has the HUGE_VALUE loop (included above) that keeps things "under control". In attempting to scale the probabilities, it just slows down the code (because it's trying to scale down exponential growth by halves). Eventually, the value becomes so high that the C code cannot process it and returns NaN, which the R code refuses to use for the acceptance vector, and the code crashes.

Alternately, the other values could be going to 0, which causes certain values to become certain (or beyond certain) in response, as opposed to the converse.

To test this theory, I ran one crash test run of the NSE code (documented), with random seed 83455 and got a NaN number for lp.vec at amino acid 4, in lp.vec[7472]. I'm now rerunning the code with that random seed, and tracking the values of lp.vec[7472]. If I'm correct, it should eventually reach above 0(log probability > 0 means probability is > 1), and then skyrocket from there.

One other possibility is that it's some kind of numerical error e.g. underflow, and the value lp.vec[7472] will suddenly jump from -30ish to 1×10^{18}

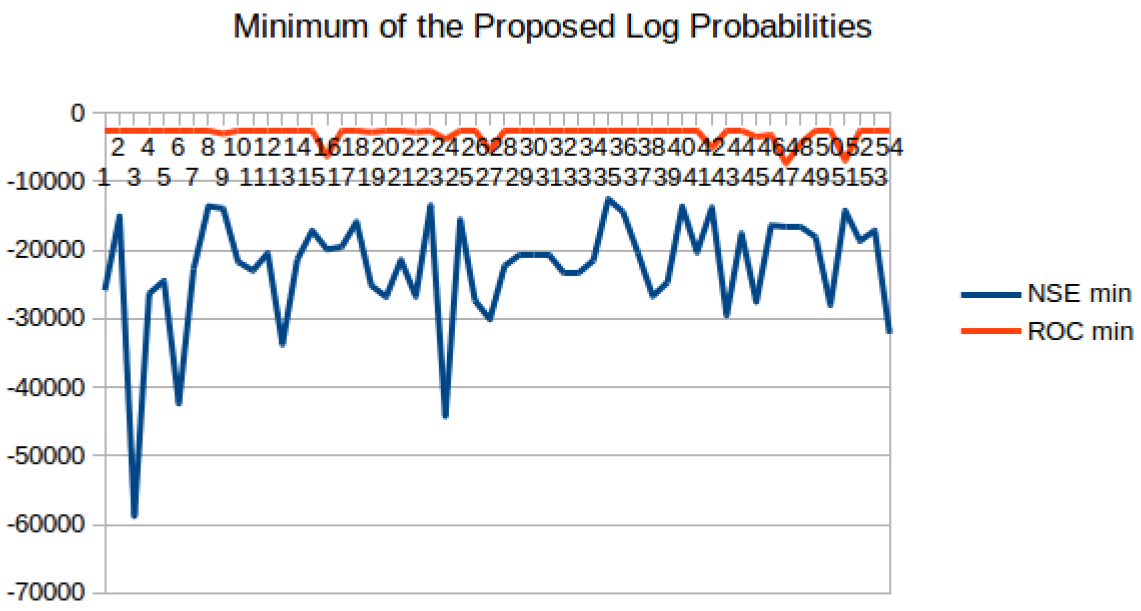
Results of test: Inconclusive

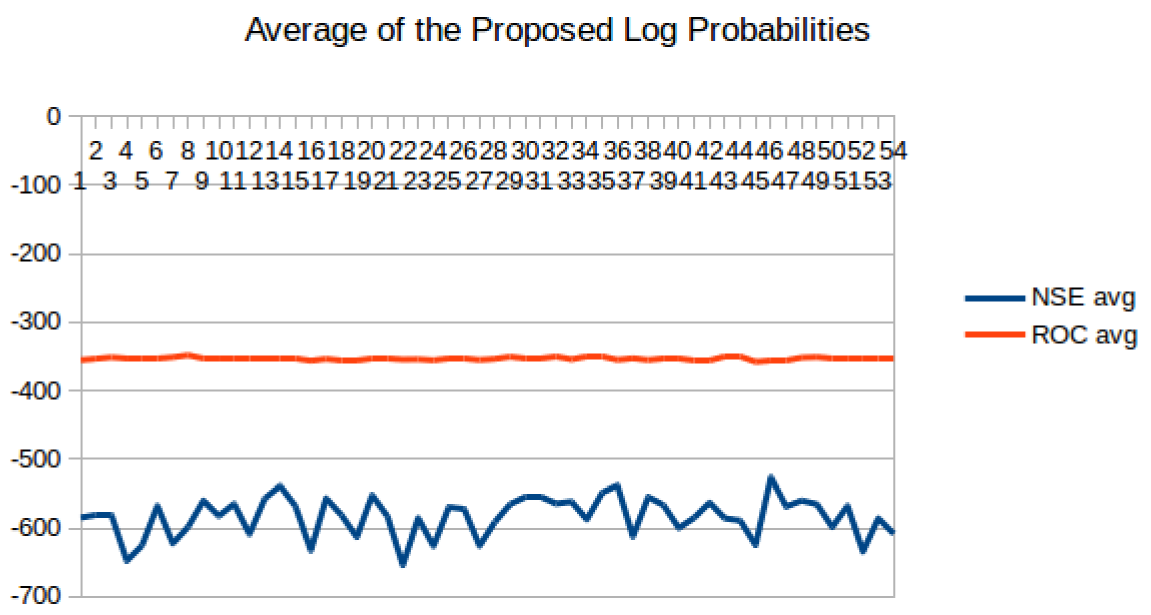
Setting the random seed to the same value generated DIFFERENT results. Cedric suggests that the MCMC may behave on a separate random seed. This is problematic.

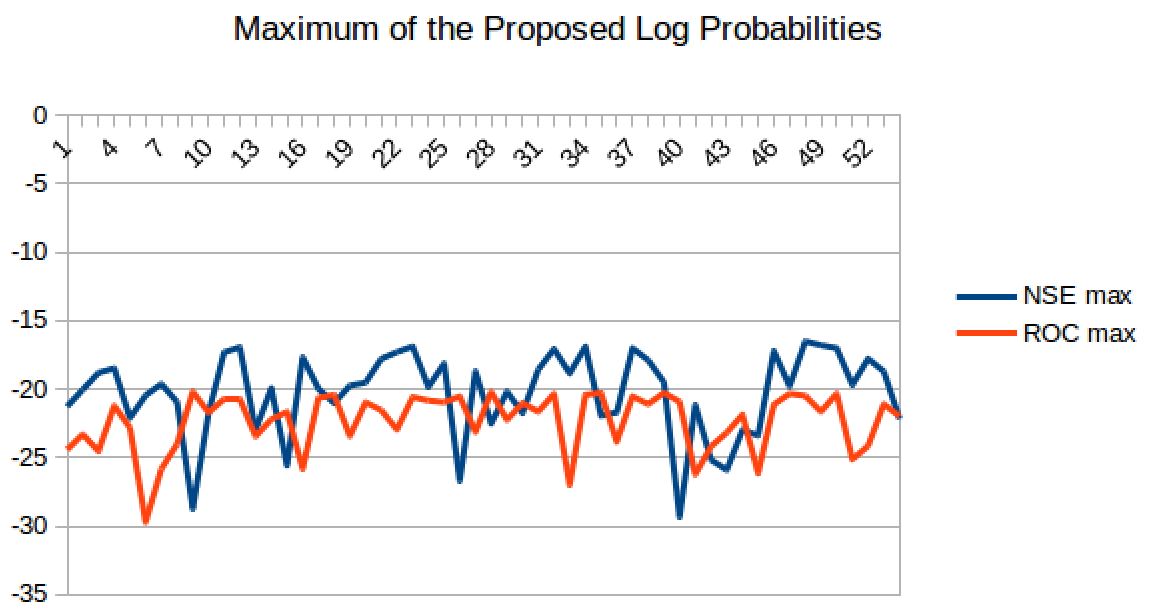
5.2.3 Compare Min, Max, and Average Values

grep (min/max/avg/NaN/inf) (file) | sort | (head/tail)

Wrote a small script, topbot.sh (min/max/avg/NaN/inf) (file)







5.2.4 Use Sections of the Simulated Yeast Genome

See cubfits/misc/S.cervisiae.REU13/random5ths.r

5.2.5 If possible, NSE Patch

One option to look into is to drop lp_c_raw from the code, and use rowsums like the Roc model. The two problems with that are that, as far as I can tell, lp.c.raw actually fixes the error in some cases, and using yaa * lp.vec complains for the NSE code. Not sure.

5.2.6 Look into an R Vim extension or RStudio

RStudio installed (also CMake), and it's VERY nice. Very intuitive, good use of screen real estate, and it holds a lot of information that I need at once. It keeps launching into /home/lbrown/PACKAGES/rstudio/bin though.

5.2.7 Debugging Thoughts

This was in last weeks summary, but it's included here for completeness.

How to GDB an R session

1. cd /home/lbrown/cubfits/misc/R
2. R -d gdb
3. run
4. source("debug_nsef.r")
5. Ctrl-C
6. continue

5.3 Goals for next Week

1. NSE Run Sections of Simulated Yeast Geome
2. Find out what causes some of the probabilties to go outrageously high
3. Compare Min, Max, and Average Values
4. Use Sections of the Simulated Yeast Genome
5. More information on Codon Position
6. Look into an R Vim extension or RStudio
7. Debugging Thoughts