Work Log for September

Logan Brown

September 15, 2014

2 Week of September 8th-15th

2.1 Goals for the Week

1. NSE Model

2.2 Progress/Notes

2.2.1 NSE Model – run using workflow.sh

These are the changes I've made to cubfits/misc to try and run the NSE model. (run_nsef.r is just a copied version of run_roc.r with the following exceptions)

- In run_nsef.r, changed model="roc" to model="nsef" in cubsinglechain and cubmultichain for both "cubfits" and "cubappr"
- In run_nsef.r, added .CF.CT$model <- "nsef"; before each each call of cubsinglechain and cubmultichain
- In run_utility.r, changed get.logL <- function(ret, data, model="roc") to get.logL <- function(ret, data, model="nsef")
- Changed workflow.r from
  - Rscript run_roc.r -c $cubmethod -s "0.5 1 2 4" -f $genome -p $empphi -o $folder -n $foutname -i $pinit >> $logfile &

    to
  - Rscript run_nsef.r -c $cubmethod -s "0.5 1 2 4" -f $genome -p $empphi -o $folder -n $foutname -i $pinit >> $logfile &

I also changed to cubsinglechain instead of cubmultichain, to try and simplify matters, but then the MCMC started throwing "acceptance out of range" at every step of the iteration. Started at 10:44, ran until 11:44.

I'll retry with cubmultichain. I had run a multichain version on 9/5, and it just stalled. However, I just added run_utility.r change

Latest settings that worked!
n.samples = 10
use.n.samples = 10
n.chains = 4
n.cores = 4
min.samples=50
max.samples=100

```
==================================================================================
=============================== START HEADER =====================================
==================================================================================
Function call:
Rscript run_roc.r -c cubfits -s 0.5 1 2 4 -f ../data/ecoli_K12_MG1655_genome_filtered.fas

MCMC parameters:
Number of samples between checks: 10
Min samples: 50
Max samples: 100
Reset QR until: 0 samples
Thining: store every 10 iteration as sample
Swap 0 % of b matrix
Swap b matrix if L1-norm is < 0

Simulation parameters
Number of Cores 4
Number of Chains: 4
Parallel mode within chain: lapply
Samples used: 10
First 0  AAs removed from sequence
Sequences with less than 0 AAs ignored
List of AAs taken into account:
A C D E F G H I K L M N P Q R S T V W Y Z

Convergence criteria
Convergence test: Gelman & Rubin
Convergence criterium: Gelman Score < 0.15
Use every 1 sample for convergence test

==================================================================================
================================ END HEADER ======================================
==================================================================================


started at: 2014-09-08 13:56:25
```

```
using cubfits
reading sequences from file ../data/ecoli_K12_MG1655_genome_filtered.fasta
reading gene expression measurements (Xobs) from file
 ../data/ecoli_X_obs.csv
and compare to ORF list from FASTA file
 ../data/ecoli_K12_MG1655_genome_filtered.fasta
generating list of codon position in ORFs for each AA...
generating list of number of AA occurences per ORF...
generating list of codon counts per ORF...
generate initial phi using SCUO with sd(ln(phi)) values
- 0.5
- 1
- 2
- 4
running cubfits using cubmultichain
 with seeds: 68072 35014 78804 49768
4 slaves are spawned successfully. 0 failed.
Gelman score after sample: 11 5.24608844491114  test was performed on 6 samples
Gelman score after sample: 21 4.43002963732354  test was performed on 10 samples
Gelman score after sample: 31 4.15615605401198  test was performed on 16 samples
Gelman score after sample: 41 4.00798122092697  test was performed on 20 samples
Gelman score after sample: 51 3.87651221427067  test was performed on 26 samples
Gelman score after sample: 61 3.76115242256423  test was performed on 30 samples
Gelman score after sample: 71 3.68585493775202  test was performed on 36 samples
Gelman score after sample: 81 3.61050525774646  test was performed on 40 samples
Gelman score after sample: 91 3.52149378911106  test was performed on 46 samples
Gelman score after sample: 101 3.43585418976874  test was performed on 50 samples
Elapsed time for 4 chains doing  iterations on 4 cores was 30.2 min
process results...
saving results...
finished at: 2014-09-08 14:28:03
```

I was able to repeat these results using 1000 samples, (minimum of 500 samples), though it stopped after 500 samples. The Gelman score was 1.10581040617552. For brevity, the 500 sample log is not included in this pdf, but it is in the github.

2.2.2 Larger NSE Model

Ran from 17:22 to 19:12, then ended with no error messages. "Timing stopped at"

Is it possible that the model is OVERconverging? I had it doing 500 samples between checks, and it should succeed at doing 500 samples...

2.2.3 NSE Model – debug

Ran the contents of cubfits/demo/nsef.train.r in an R interactive session, adding in the line debugonce(cubfits)

As far as I can tell, the cubfits() function itself doesn't actually use the model or .CF.CT$model, I'll need ot look at drawP and drawPhi.

According to Cedric, it's worthwhile to look at cubfits/cubfits/R/my.logdmultinomCodOne.r. That function calls cubfits/cubfits/src/stable_exp.c and cubfits/cubfits/src/lib.c.

2.2.4 Optimal/Pessimal Code

Cedric made the change that had been discussed, where we switch from Δt values to $\Delta\eta$

In making that change, I also made several more changes.

- fixed a bug where the default codon for Q was wrong
- Changed from analyzing Δt to $\Delta\eta$ values. Made a mock up etaValues.bmat for example by just inverting the signs of the values
- made the language more clear for "optimal/best" versus "minimum/maximum"
- added a count and ratio for how many codons are actually being up/downgraded

Deepika contacted me about an error in the code. The beginning of the for loop (line 78 in the original, now line 79)

for(index in 1:length(sequence[[gene]]/3)){

to

for(index in 1:(length(sequence[[gene]])/3)){

The first one was causing a fractional value of index, which was making it do weird things, but mainly, it was making the code replace random chunks of genome, rather than accurately dividing the codon into groups of 3.

2.2.5 First Order Approximations

For completeness, I also wrote down how we derived the fitness function in fitnessHistory.tex

p_{ic} is the probability of a nonsense error at position i using codon c
p_j is the probability of a nonsense error at position j

First order approximation about $p_{ic} = 0$ is

$$f(p_{ic} \approx f(0) + f'(0) * (p_{ic} - 0) = p_{ic}\left(\left[\sum_{k=1}^{i} a_1 + a_2(k-1)\right]\left[\prod_{j=i+1}^{n}\left(\frac{1}{1-p_j}\right)\right]\right)$$

First order approximation about $p_{i+1} \approx p_{i+2} \cdots \approx p_n \approx 0$ is

$$f(p_j) \approx \left[\sum_{k=1}^{i} a_1 + a_2(k-1)\right] \left[\frac{p_{ic}}{1-p_{ic}}\right] + ((i+1)-n) \left[\sum_{k=1}^{i} a_1 + a_2(k-1)\right] \left[\frac{p_{ic}}{1-p_{ic}}\right] (p_j)$$

$$f(p_j) \approx f(0) + (f(0))(p_j)((i+1)-n)$$

Details are in approx.tex

2.2.6 Simulated Data Set from REU13

The data is in /home/lbrown/reucode/data/inputsimdata/REU_data

I think elongation rate refers to the odds $\frac{p_i}{1-p_i}$.

2.3 Goals for next Week

1. Simulated Data Set
2. Potentially look at the c files in cubfits/cubfits/src
3. ~~Code the first order approximations? Looking at my draw*.r for the fitness function~~
4. Check for problems in Wei-Chen's NSE code, specifically
   (a) ~~Unnecessary VGLM calls leading to longer run times~~ As far as I can tell, my.fitMultinomOne.nsef and my.fitMultinomOne.roc are using the same amount of vglm calls.
   (b) Ending crash caused by running out of memory
5. Continue to improve the optimal-pessimal code as Deepika works with it.