Work Log for October

Logan Brown

October 22, 2014

Contents

1 Goals for the Month

1. Find out what causes some of the probabilties to go outrageously high
2. Use the Simulated Yeast Genome, compare to the Ecoli genome
3. If possible, NSE Patch
4. Debugging Thoughts

2 Progress/Notes

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

2.1.1 Compare Differences in the Code

```
my.coef.r
my.drawPhiConditionalAllPred.r
my.drawPhiConditionalAll.r
my.estimatePhiOne.r
my.fitMultinomOne.r
my.logdmultinomCodOne.r
my.objectivePhiOne.Lfp.r
my.objectivePhiOne.nlogL.r
my.objectivePhiOne.nlogphiL.r
my.objectivePhiOne.phiLfp.r
my.pPropTypeNoObs.lognormal_bias.r
plotbin.r
plotmodel.r
simu.orf.r
```

Codes where NSE and ROC are the same, or incredibly similar:

```
my.coef.r
my.estimatePhiOne.r
```

Codes where NSE and ROC differ:

2.2 Compare Genomes

2.2.1 Use Sections of the Simulated Yeast Genome

I ran one 5th of the yeast genome(section1.fasta, section1.csv) and it FINISHED. but... not well.

See the results in data/10.01.yeastnse.*

- It took just over 30 hours to finish
  started at: 2014-10-01 15:57:54
  finished at: 2014-10-02 22:08:56
- The proposed phi values are still outrageously too high. The final proposed phi had mean value 2766786.09545948
- Still proposing some 0 (-Inf) values for lpProp. At most, around 45-49 Inf per run.
- Xobs (the original phi values) are generally much lower than the results of run
- The values are not very accurate.

2.2.2 Crashed Yeast Run

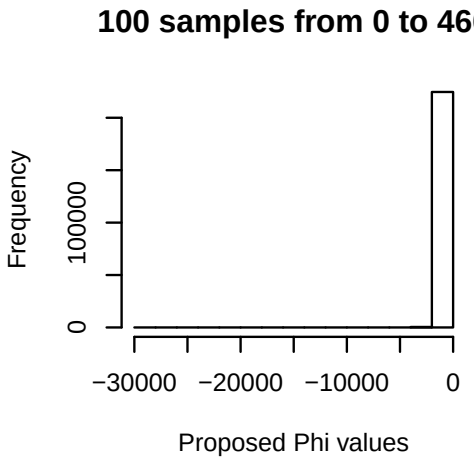
The vast vast majority of the phi values being proposed are fine. -Inf values are throwing it off. 10/22 Note: Due to the config file change (See Patch 1 and "look at stddev(phi)"), I'm discrediting these values.

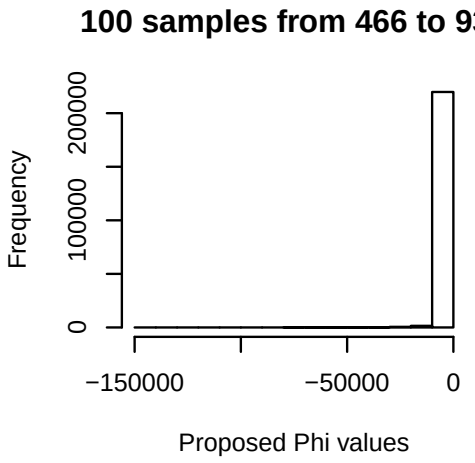
2.2.3 E. Coli genome

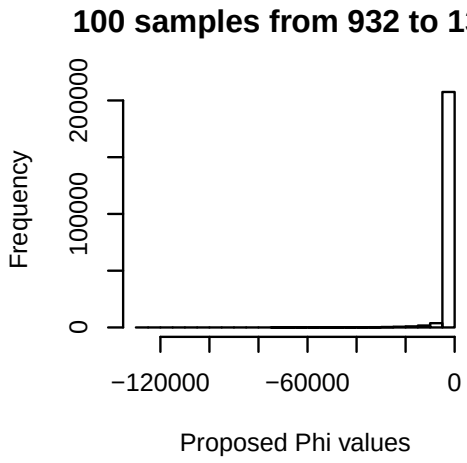
A LARGE NSE RUN OF THE E. COLI GENOME RAN TO COMPLETION. 10/22 Note: Due to the config file change (See Patch 1 and "look at stddev(phi)"), I'm discrediting these values.

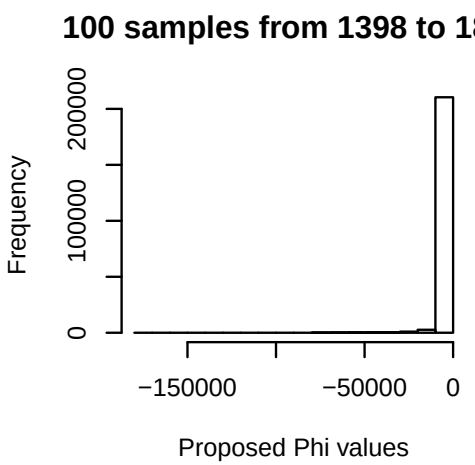
2.3 Look at the change in the proposed Phi distribution over time

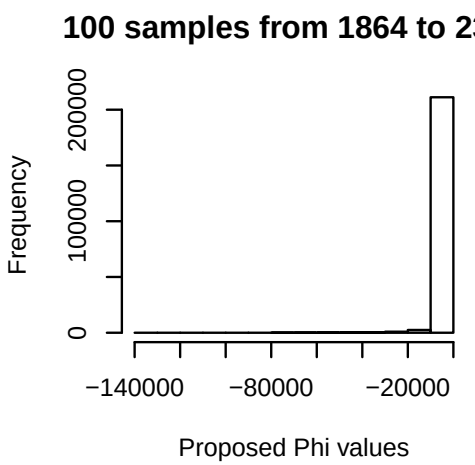
Trying to find a nice way to look at these.

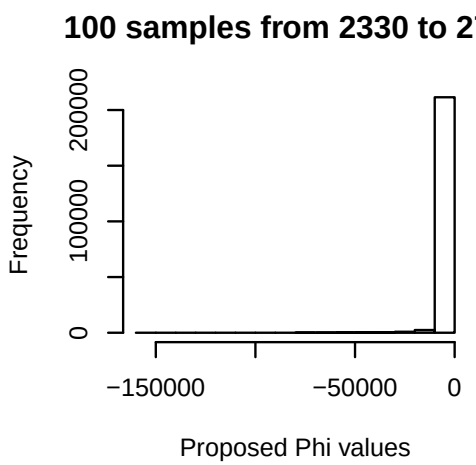

100 samples from 0 to 466
Frequency
0
100000
−30000
−20000
−10000
0
Proposed Phi values

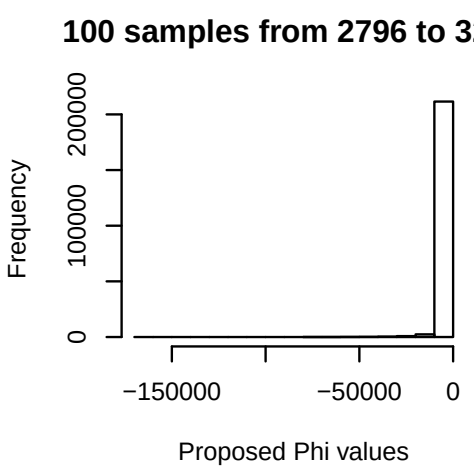

100 samples from 466 to 932
Frequency
0
100000
200000
−150000
−50000
0
Proposed Phi values

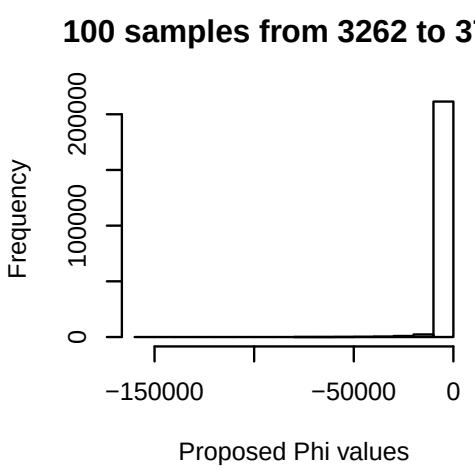

100 samples from 932 to 1398
Frequency
0
100000
200000
−120000
−60000
0
Proposed Phi values

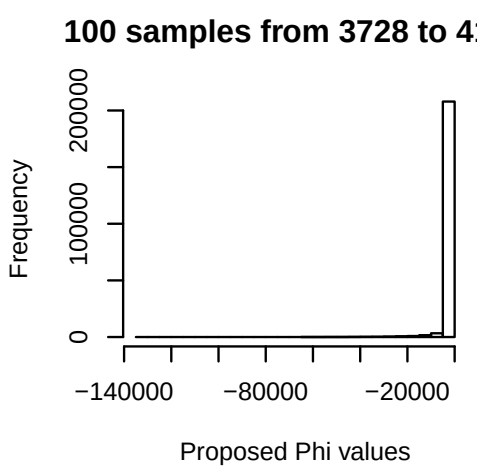

100 samples from 1398 to 1864
Frequency
0
100000
200000
−150000
−50000
0
Proposed Phi values


100 samples from 1864 to 2330
Frequency
0
100000
200000
−140000
−80000
−20000
Proposed Phi values


100 samples from 2330 to 2796
Frequency
0
100000
200000
−150000
−50000
0
Proposed Phi values


100 samples from 2796 to 3262
Frequency
0
100000
200000
−150000
−50000
0
Proposed Phi values


100 samples from 3262 to 3728
Frequency
0
100000
200000
−150000
−50000
0
Proposed Phi values


100 samples from 3728 to 4194
Frequency
0
100000
200000
−140000
−80000
−20000
Proposed Phi values

2.4 bmat being badly written

One concern that I have is that the bmatfiles are being badly written for the NSE model. They should look like

```
Parameter,Value,SD
A,0.601255504722893,0.0237033066067192
A,1.13452224653582,0.0267133181736158
A,1.30103073124409,0.0238480349000172
A,-0.318856124276582,0.0185921125022695
A,-0.638707250936925,0.0239128208157251
A,-0.430302895729612,0.0198676086998986
C,-0.0281672064573597,0.050965967492386
C,0.306346224638228,0.0485649142612776
D,-1.04923142474412,0.0247797820471684
D,0.548947123948893,0.0217844280815052
```

However, theyl ook like this

```
Parameter,Value,SD
(Intercept):1,-1.13368719411271,0.0163254262432022
(Intercept):2,0.474079558190101,0.0187917078911786
(Intercept):3,0.898302427488134,0.0273650691145686
tmp.phi:reu13.df.aa$Pos:1,0.000638113297651608,3.70900280679152e-05
tmp.phi:reu13.df.aa$Pos:2,2.66219626036282e-05,9.17596078252389e-06
tmp.phi:reu13.df.aa$Pos:3,0.000762697807068617,7.47917510033425e-05
(Intercept),0.407527027594279,0.0285214413558274
tmp.phi:reu13.df.aa$Pos,0.000319785184813285,8.90451644378058e-05
(Intercept),0.728372367207557,0.0119226668866337
tmp.phi:reu13.df.aa$Pos,-6.65911530134893e-05,1.09884115801929e-05
```

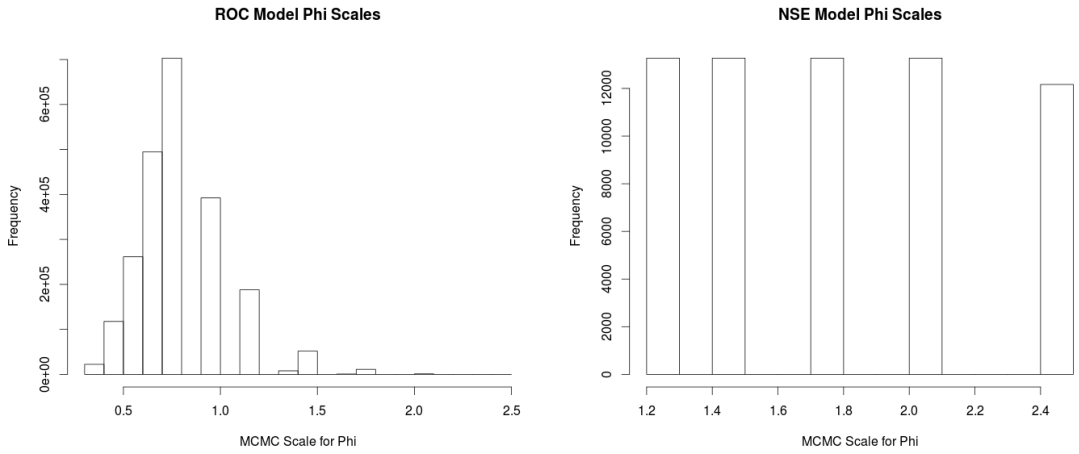
I'm fairly sure that this is some part of the code that is just getting the names of the codons wrong, however, I can't be sure. If deprecated part of the NSE code is reading the data structure incorrectly, then anything is possible. I want to fix this at one point

2.5 Look at stddev(phi)

I have a run going that is tracking the scales of the MCMC for the phi values.

Since the scale of the MCMC should be about the Standard Devation of Phi, we'll see what happens.

PROBLEM: The ROC model has different Phi scales for each gene. The NSE model keeps all the scales the same between different genes. That could be contributing to the problem? ~~I think this is caused by the opatch we wrote that replaces Wei-Chen's scaling by .5 with subtracting out log(DBL_MAX).~~



Every 6th time that the scale is updated, the code resets all the scales to 1. This definitely happens in Yeast (NSE and ROC), and it definitely happens for NSE (Yeast and Ecoli). Does it happen in ROC-Ecoli? Yes, confirmed.

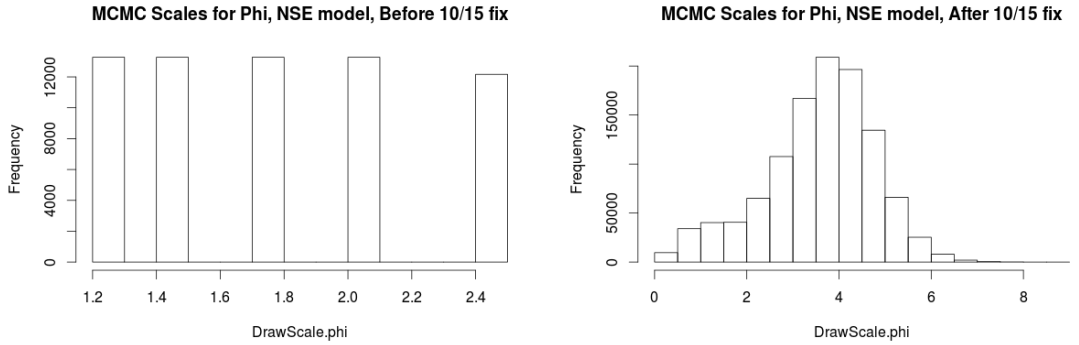
Also, Cedric says the number 6 is a result of the inputs. It has to do with the fact that my runs have been doing 50 samples between convergence checks and 10000 . I'm going to play with config.r to see this effect. (Confirmed).

Theory: length(.cubfitsEnv$DrawScale$phi) is

$$\frac{\text{Number of Samples between Convergence Checks}}{100} + 1.$$

Every time that the code checks for convergence, it reinitializes some values, and causes the problem.

By changing my values in config.r so that the code never checks for convergence (and therefore, never resets the scale), we get the following picture (left, before the change, right after the change)



- It's worthy of note that this is the second time a run of the NSE model on the Yeast genome has run to completion. Since this has happened in the past, I don't

want to entirely attribute the successful run to the removal of the scaling reset, but this is some evidence. This may have been the problem preventing a successful NSE run.

- I expect the ROC model to have a similar but less dramatic change. The former phiscale values were all ones that were within $(1.2)^i(0.8)^j$ where $i, j, i+j \in \{0, 1, ..., 6\}$. The ROC model should have more varied scaling terms (likely a Normal distribution)

- It appears that the "proper" distribution of the phi scales is roughly a normal distribution, that is slightly biased towards lower values. The whole set has mean=3.570621 and σ=1.196958.

- but this seems odd to me. If the scale should be something like 4, but it was being artificially held back to $[0, 2.48832]$, why was it therefore proposing NaN values? Shouldn't a LARGER scale make us more likely to propose impossible phi values? A small (and indeed, a very strangely behaving) scale should only prevent us from making progress, it shouldn't push us over some sort of cliff in the parameter space.

2.6 Make a git repository for my scripts

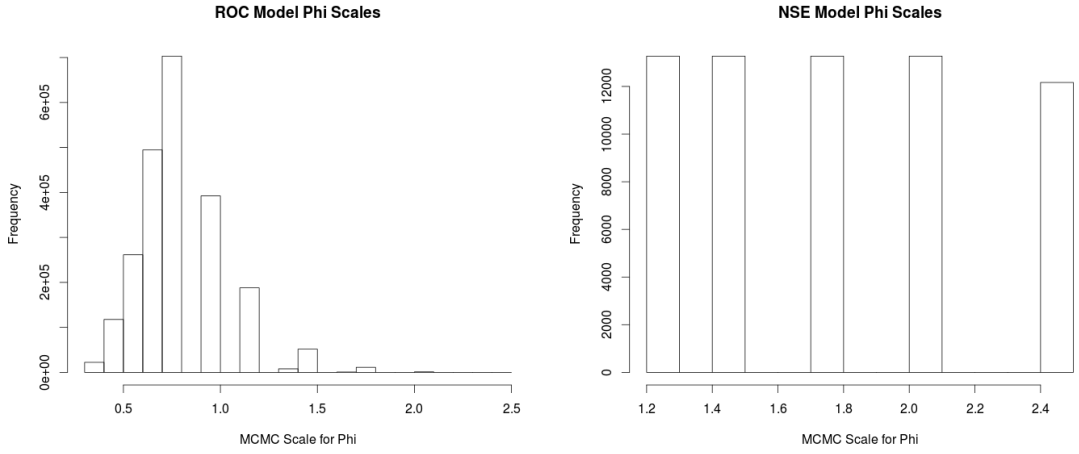
Done. https://github.com/ozway/cubmisc

2.7 NSE Patches

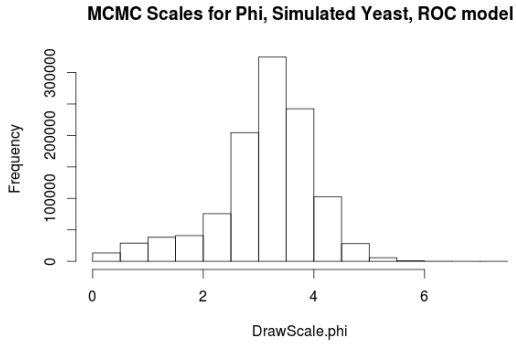
2.7.1 Patch 1 - Avoiding the 6th iteration reset

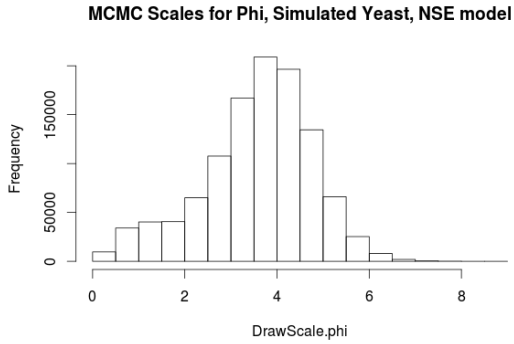
This is described in more detail in the section about stddev(phi), but I want to include the complete picture here.

These were the histograms of the Phi Scales before avoiding the 6th iteration reset. Note that the histogram on the left is actually of the Roc model for E Coli. Do not compare it to the Yeast histogram below.

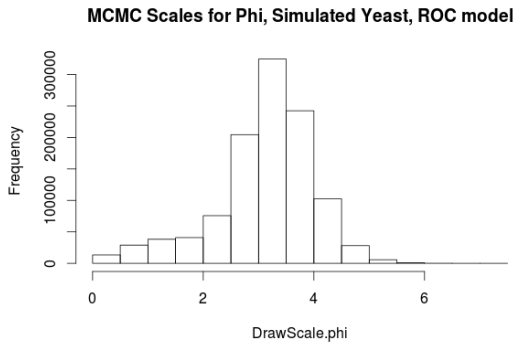


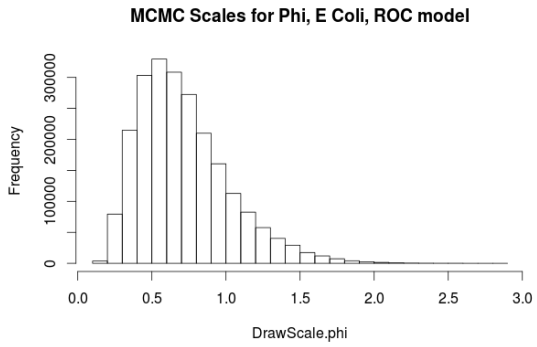
These are the updated histograms.





Here's a side-by-side comparison of RocYeast versus RocEcoli





2.7.2 Patch 2 - Avoiding the Unending Scaling Loop

I implemented C changes that replaced Wei-Chen's unending scaling loop with a one step subtraction. It seems that the code actually takes very slightly longer (.00013% longer). My best explanation is that these operations were not taking up much time in the first place, so any differences is runtime were overtaken by the innate differences in the runtime (due to randomness).

	Before Change	After Change
(total time)	124806.34	125446.36
my.inverse.mlogit	50367.20	50641.80
.Call	41140.64	41373.00
match	10364.14	10441.72
is.nan	4784.92	4809.60
%*%	4507.52	4536.28
<Anonymous>	3497.62	3548.08
cbind	3444.98	3448.48
matrix	2634.48	2633.28
cat	1750.56	1720.10

If further testing confirms this, I may revert the change, giving Wei-Chen's code the benefit of the doubt.

More problematic is that this still doesn't fix the important problem, values going to NaN causing the acceptance step to crash.

3 Patch 3 - Fix how the .Bmat file is written

4 Goals for next Month

1. Future Goal