Work Log for November

Logan Brown

November 5, 2014

Contents

1 Goals for the Month 2

2 Progress/Notes 2
- 2.1 Use Preston's Simulated Yeast, compare to REU yeast . . . 2
- 2.2 Visualization . . . 14
  - 2.2.1 'true' values vs simulated values . . . 14
- 2.3 Parallelize the Code . . . 14
  - 2.3.1 getOption("mc.cores") . . . 14
  - 2.3.2 Timing . . . 14
- 2.4 Add (a1-a2) as a parameter . . . 14
- 2.5 Wei Chen's Yeast / Real Yeast Genome . . . 15
- 2.6 Generate my own simulated yeast, using a reverse engineered cubfits . . . 15

3 Goals for next Month 15

1 Goals for the Month

As of October 31st

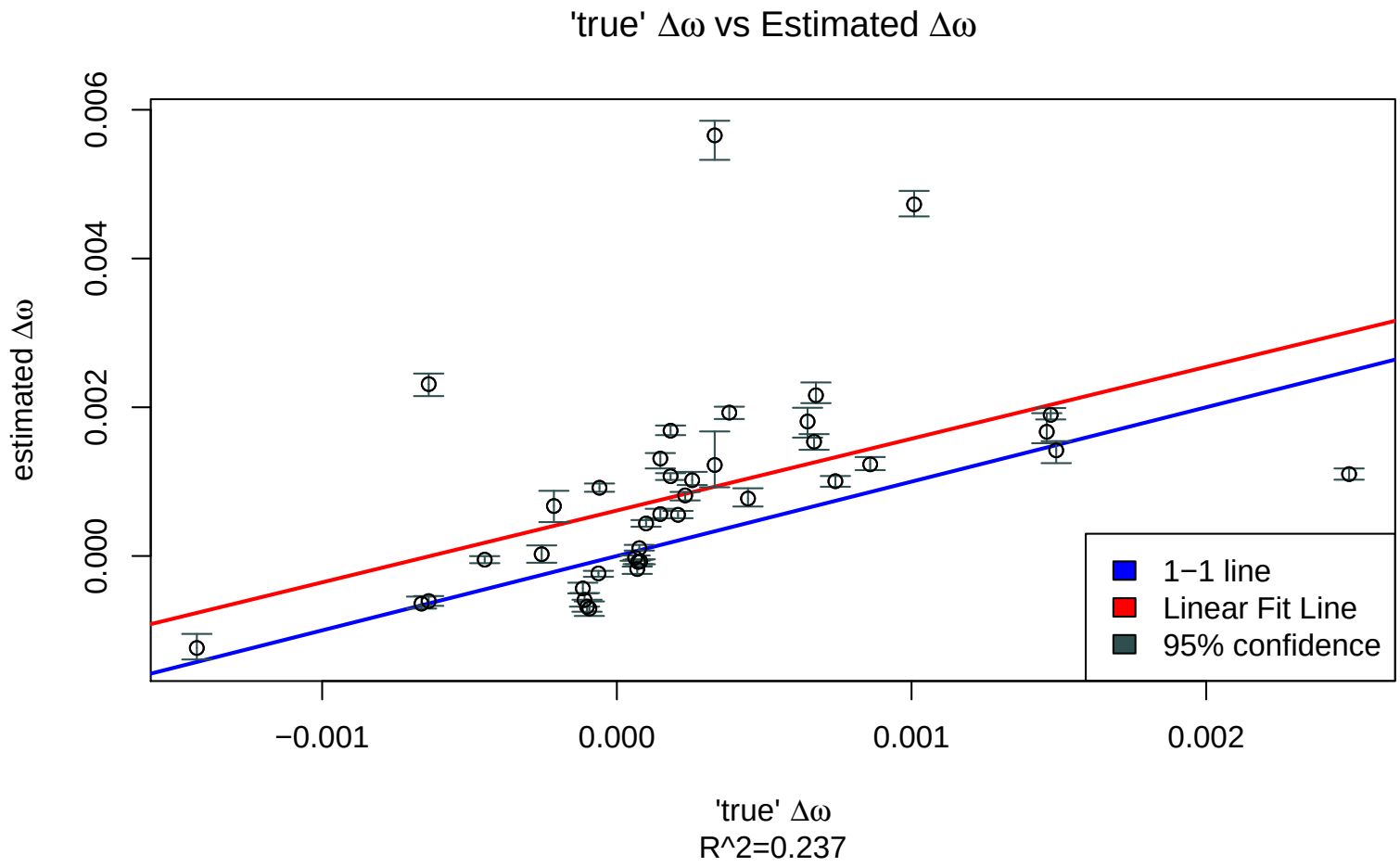
1. Use Preston's Simulated Yeast, compare to REU yeast
   look for estimated $\approx$ 4*true
2. Parallelize the Code
   mclapply, getOption("mc.cores")?
3. Wei Chen's Yeast / Real Yeast Genome
4. Generate my own simulated yeast, using a reverse engineered cubfits

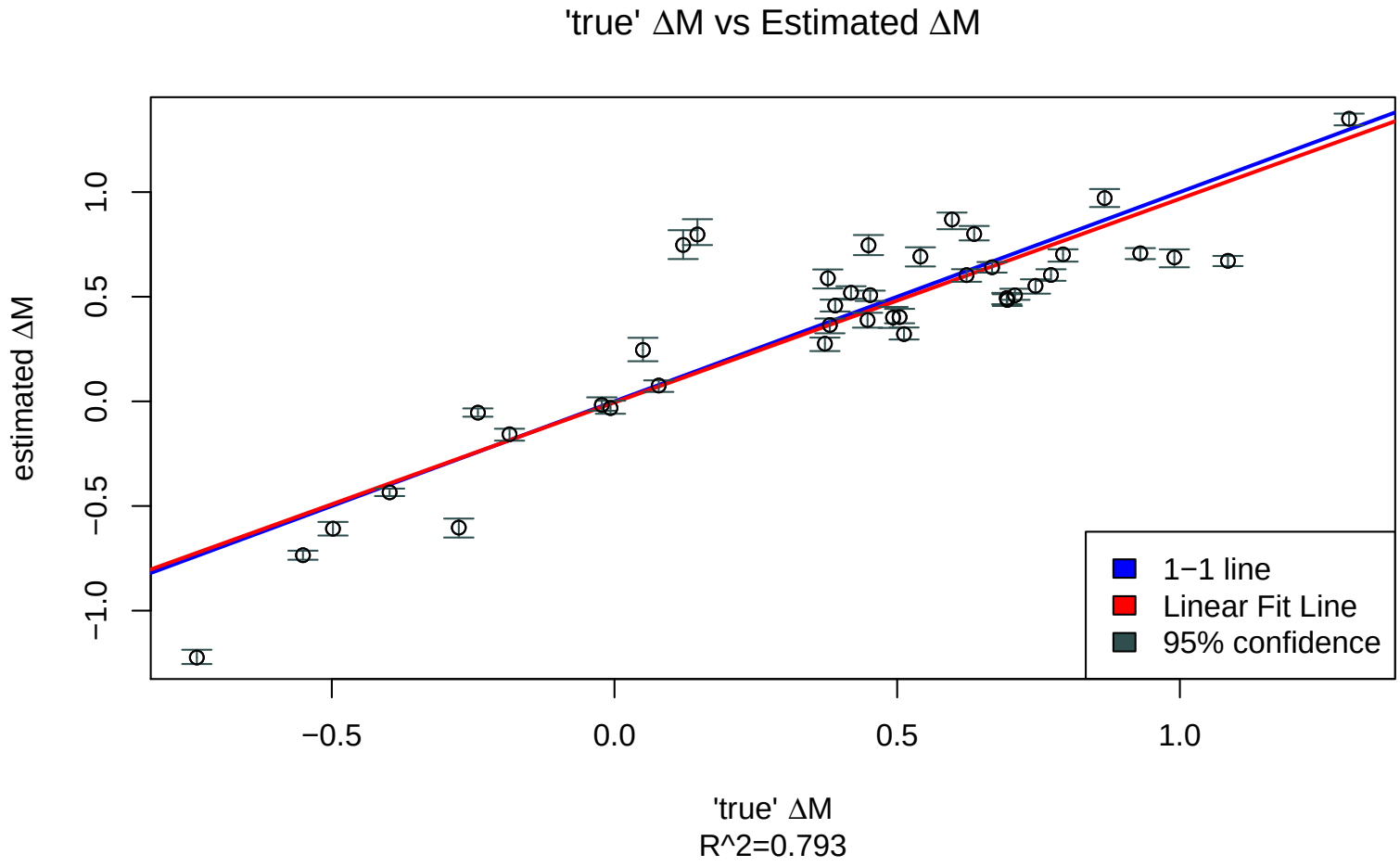
2 Progress/Notes

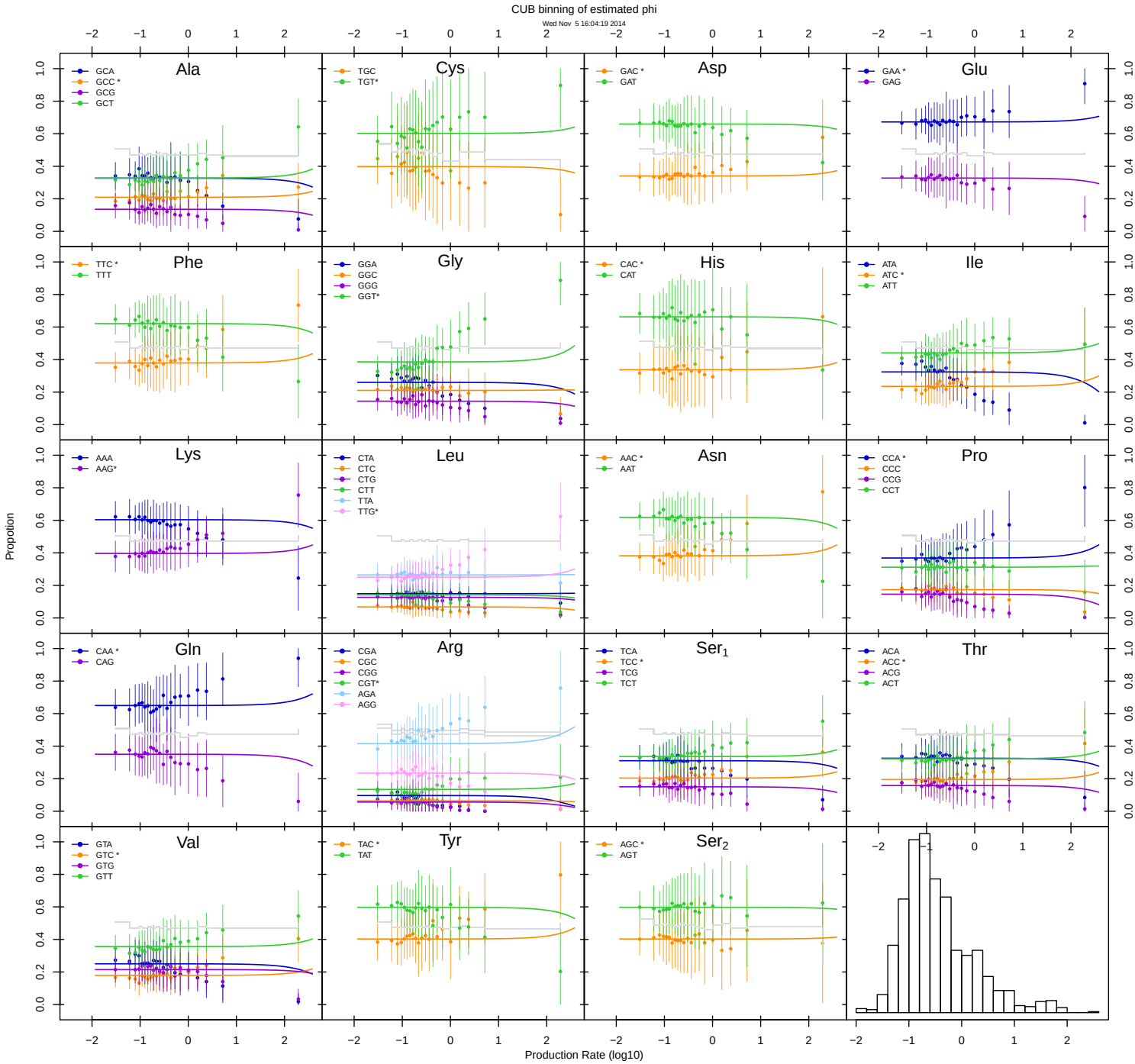
2.1 Use Preston's Simulated Yeast, compare to REU yeast

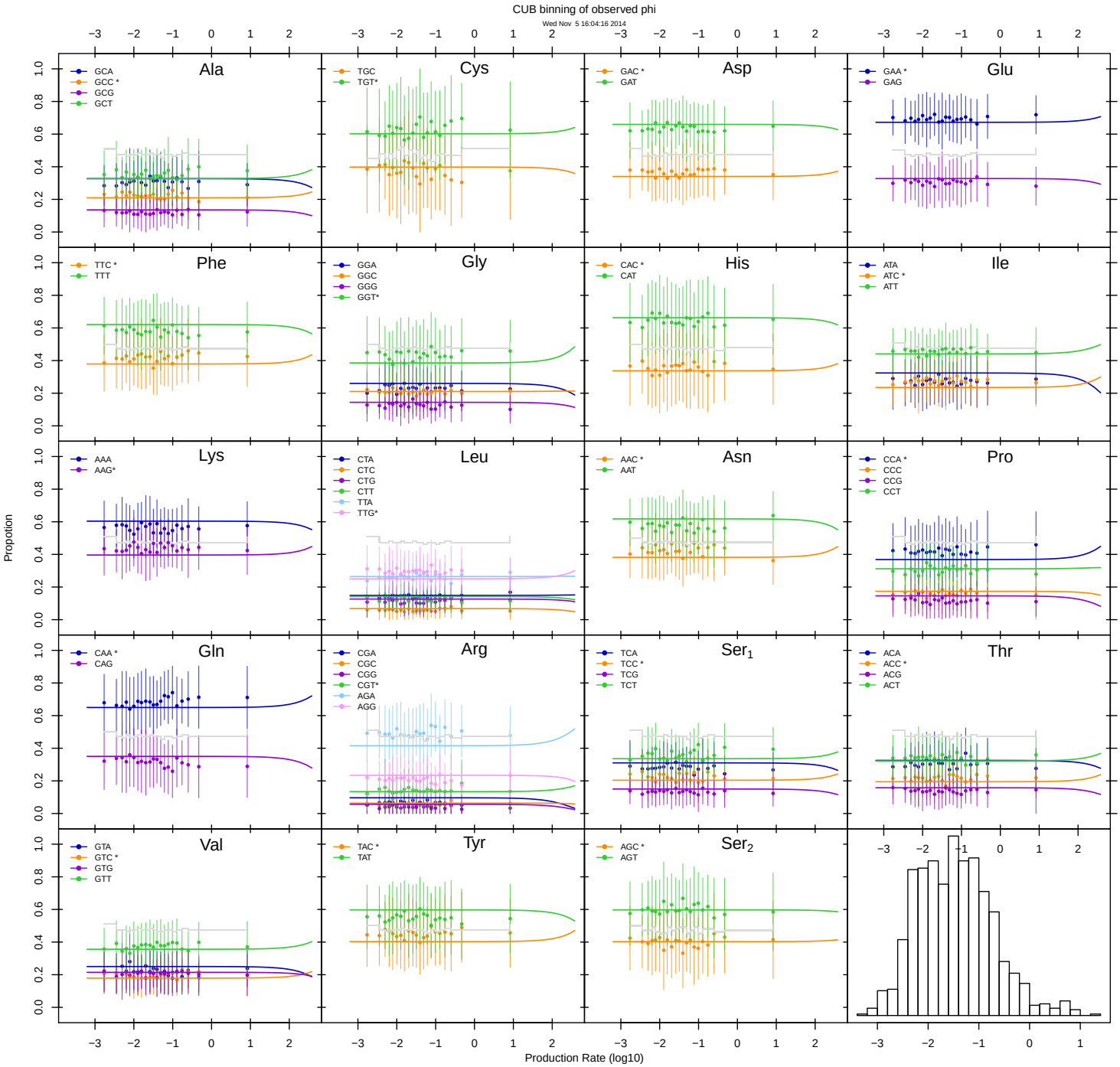
Here are the results from Preston's simulated yeast...

Notice that the $\Delta\omega$ values are not off by that $\approx$ 4 factor. We got a pretty good correlation on the logμ values, but the ϕ is pretty lousy, and the ω values leave something to be desired.


'true' Δω vs Estimated Δω
estimated Δω
0.000
0.002
0.004
0.006
−0.001
0.000
0.001
0.002
'true' Δω
R^2=0.237
1−1 line
Linear Fit Line
95% confidence

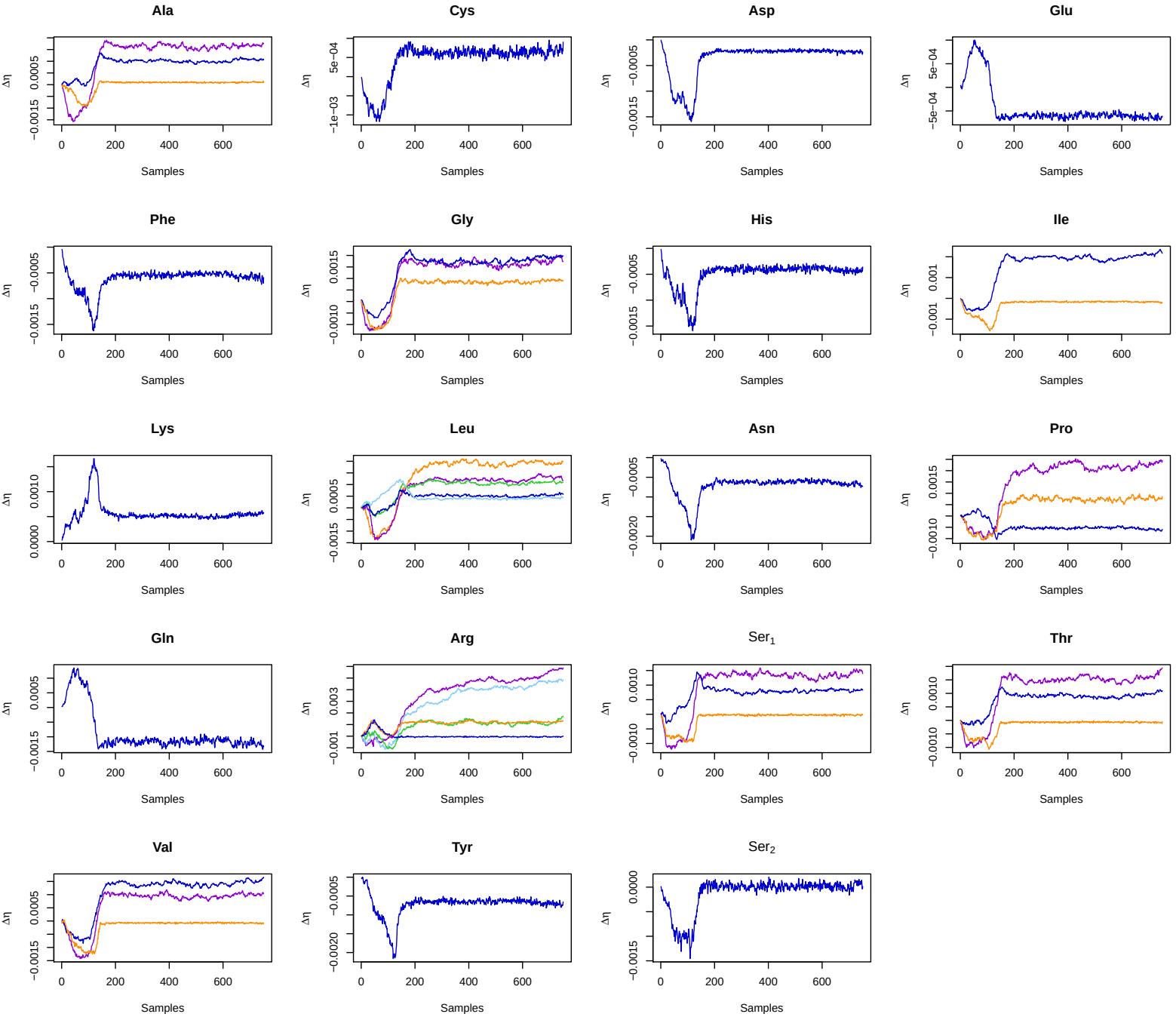

'true' ΔM vs Estimated ΔM
estimated ΔM
−1.0
−0.5
0.0
0.5
1.0
−0.5
0.0
0.5
1.0
'true' ΔM
R^2=0.793
1−1 line
Linear Fit Line
95% confidence

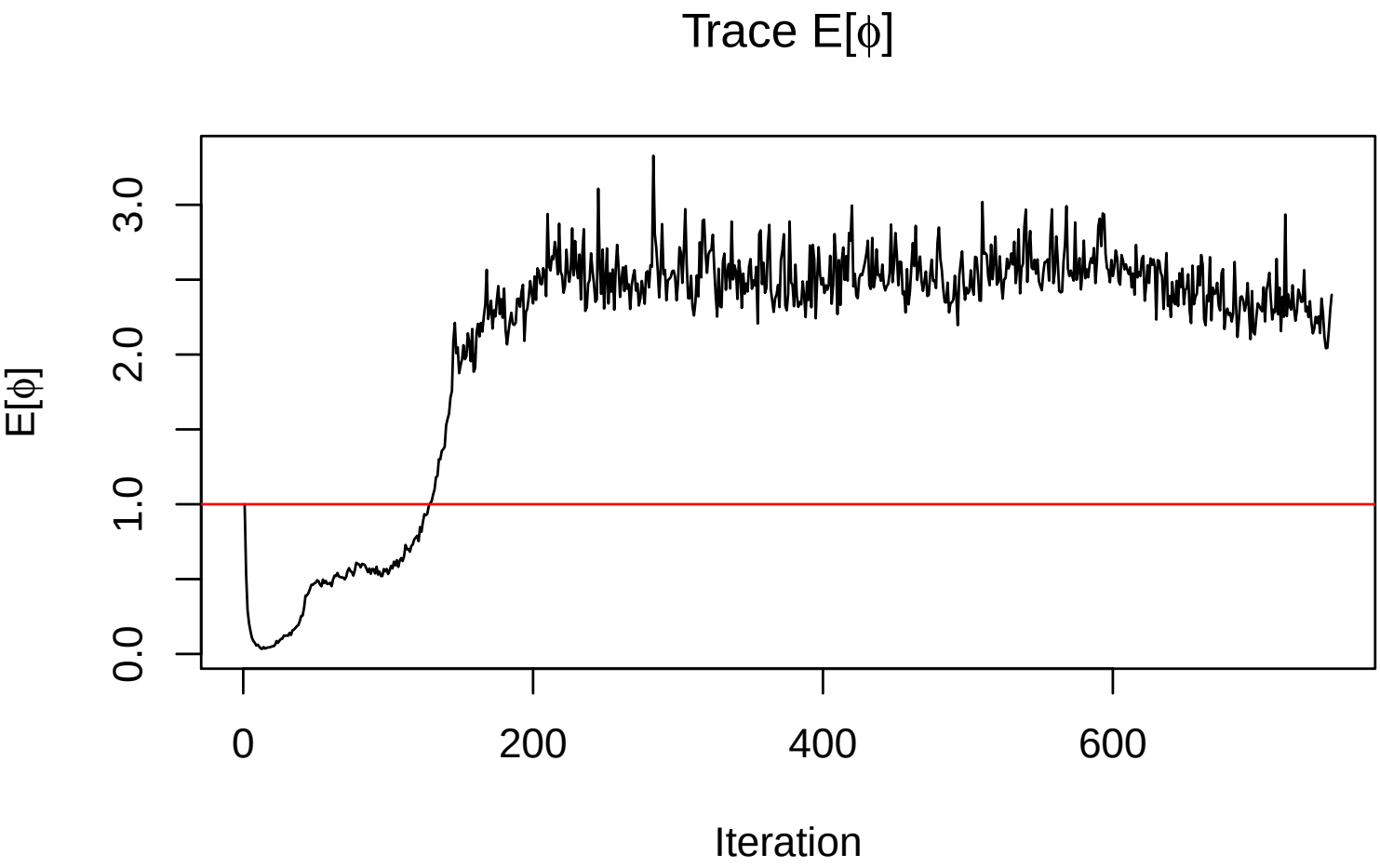

CUB binning of estimated phi
Wed Nov 5 16:04:19 2014
Ala
GCA
GCC *
GCG
GCT
Cys
TGC
TGT*
Asp
GAC *
GAT
Glu
GAA *
GAG
Phe
TTC *
TTT
Gly
GGA
GGC
GGG
GGT*
His
CAC *
CAT
Ile
ATA
ATC *
ATT
Lys
AAA
AAG*
Leu
CTA
CTC
CTG
CTT
TTA
TTG*
Asn
AAC *
AAT
Pro
CCA *
CCC
CCG
CCT
Gln
CAA *
CAG
Arg
CGA
CGC
CGG
CGT*
AGA
AGG
Ser1
TCA
TCC *
TCG
TCT
Thr
ACA
ACC *
ACG
ACT
Val
GTA
GTC *
GTG
GTT
Tyr
TAC *
TAT
Ser2
AGC *
AGT
Propotion
Production Rate (log10)

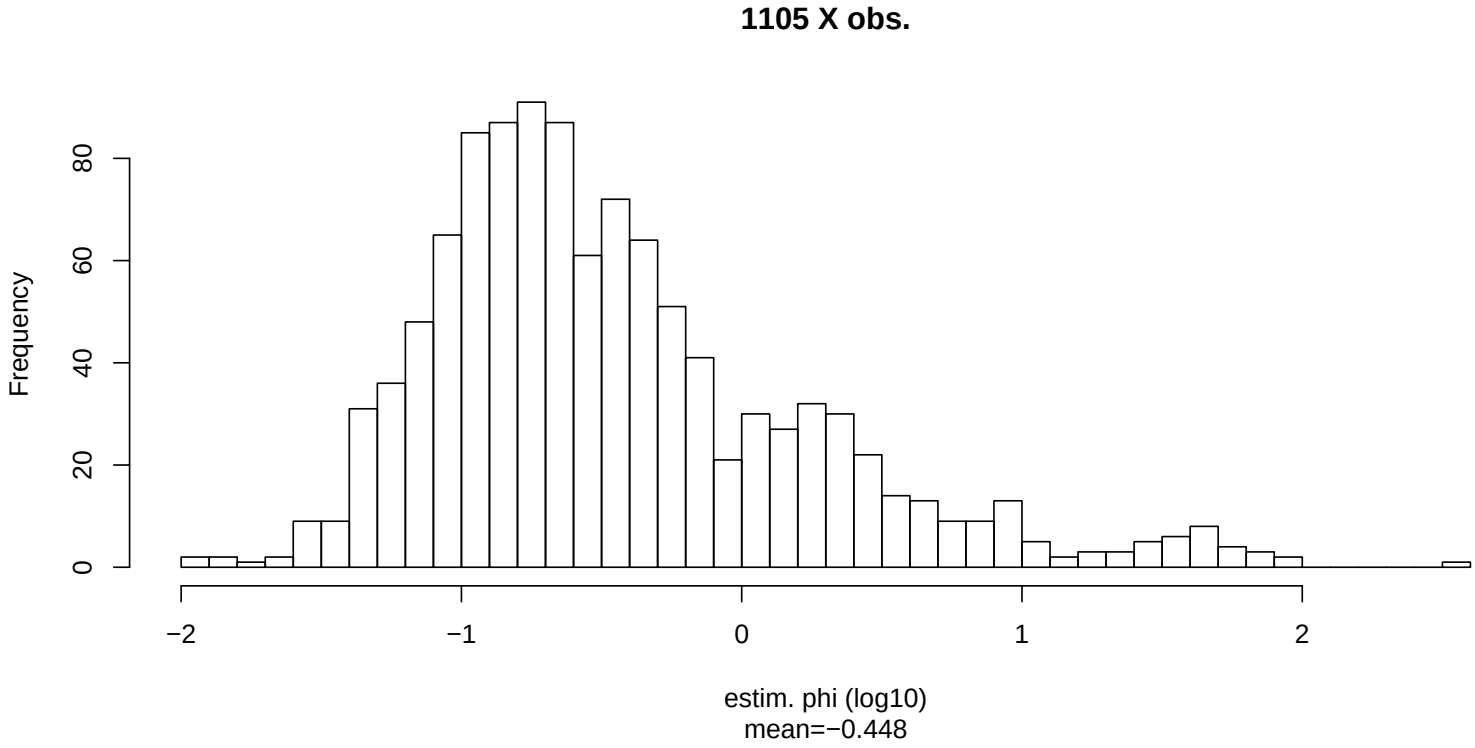

CUB binning of observed phi
Wed Nov 5 16:04:16 2014
Ala
GCA
GCC *
GCG
GCT
Cys
TGC
TGT*
Asp
GAC *
GAT
Glu
GAA *
GAG
Phe
TTC *
TTT
Gly
GGA
GGC
GGG
GGT*
His
CAC *
CAT
Ile
ATA
ATC *
ATT
Lys
AAA
AAG*
Leu
CTA
CTC
CTG
CTT
TTA
TTG*
Asn
AAC *
AAT
Pro
CCA *
CCC
CCG
CCT
Gln
CAA *
CAG
Arg
CGA
CGC
CGG
CGT*
AGA
AGG
Ser1
TCA
TCC *
TCG
TCT
Thr
ACA
ACC *
ACG
ACT
Val
GTA
GTC *
GTG
GTT
Tyr
TAC *
TAT
Ser2
AGC *
AGT
Propotion
Production Rate (log10)

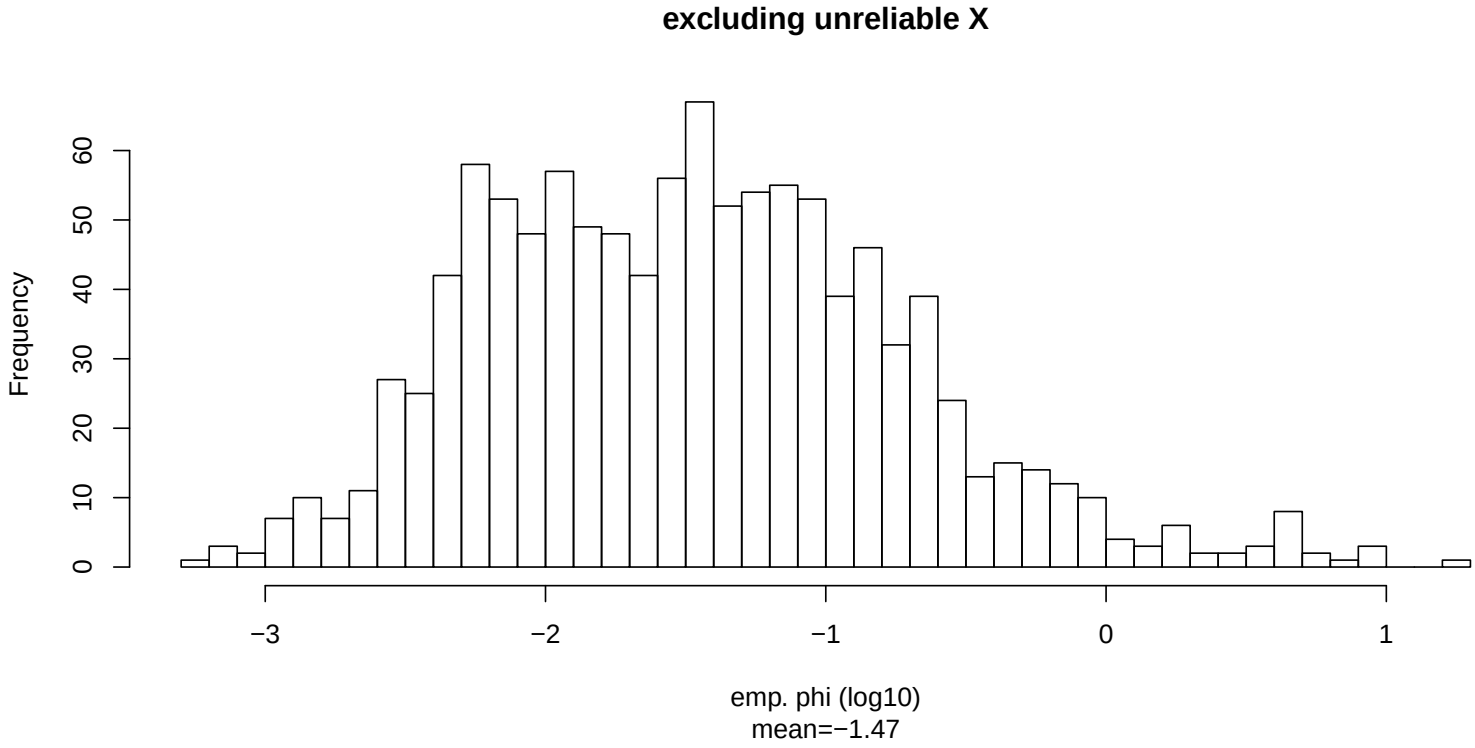
AA parameter trace 1105

Wed Nov 5 16:04:14 2014



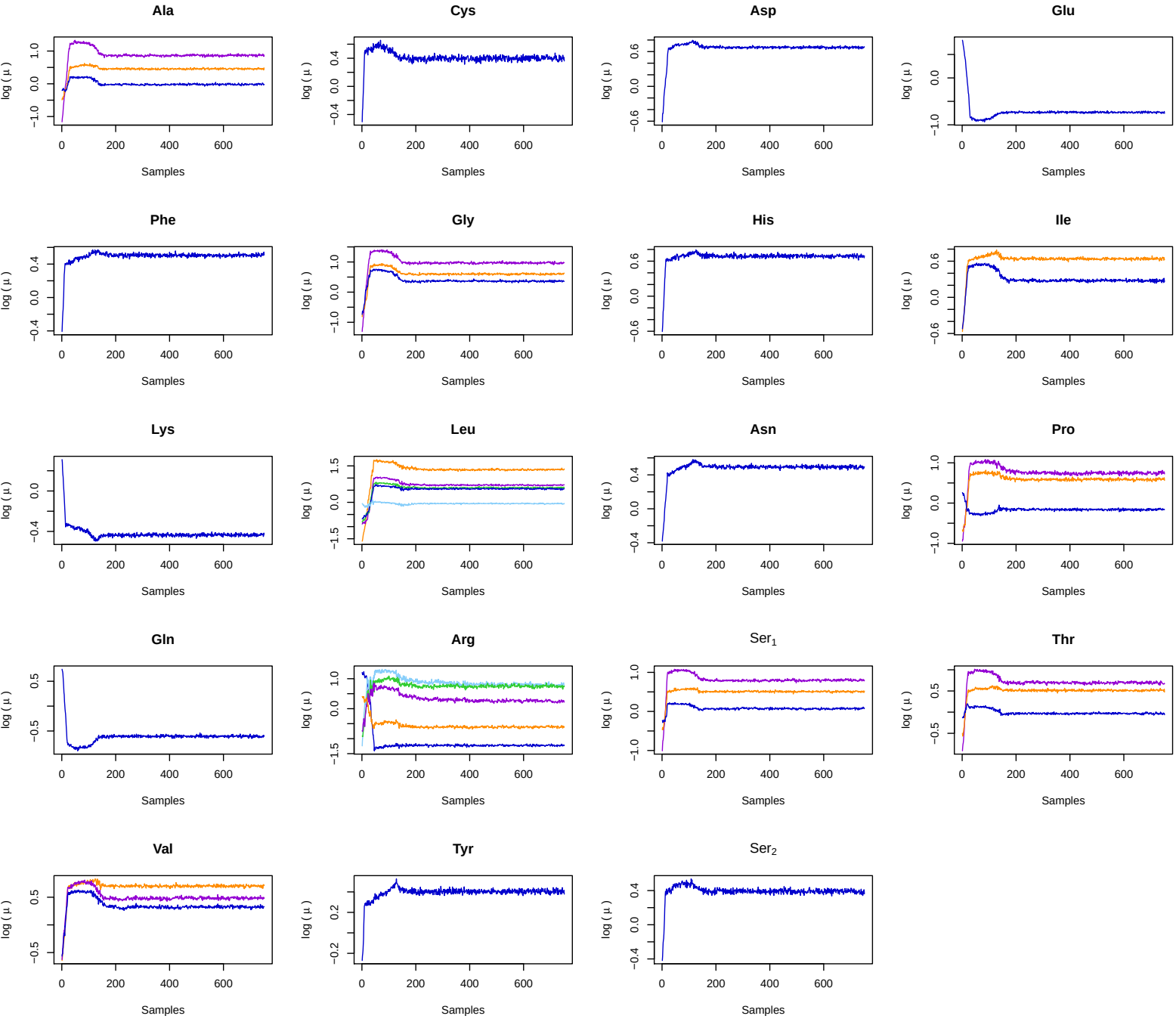

Trace E[ϕ]
E[ϕ]
0.0
1.0
2.0
3.0
0
200
400
600
Iteration

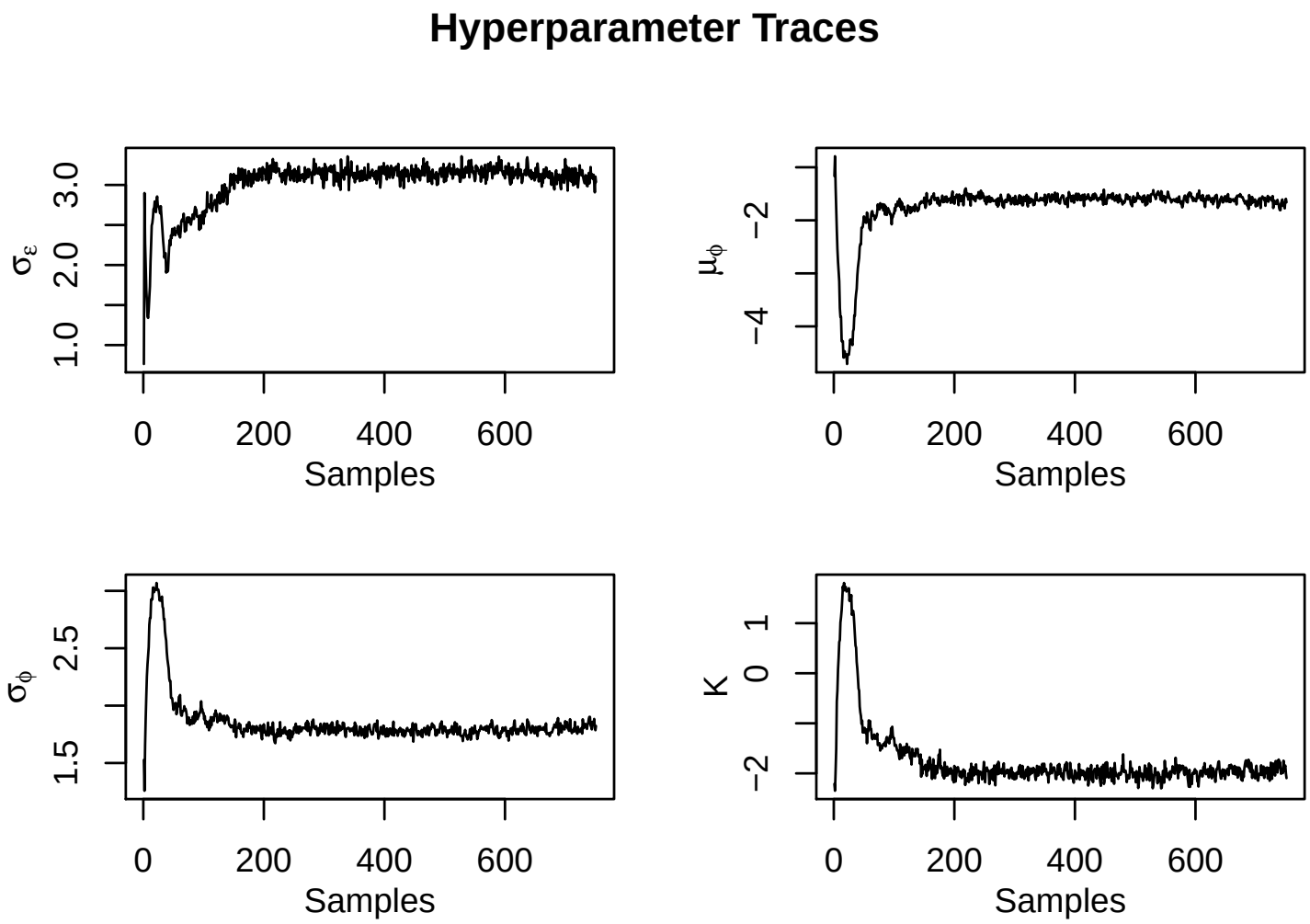

1105 X obs.
Frequency
0
20
40
60
80
−2
−1
0
1
2
estim. phi (log10)
mean=−0.448

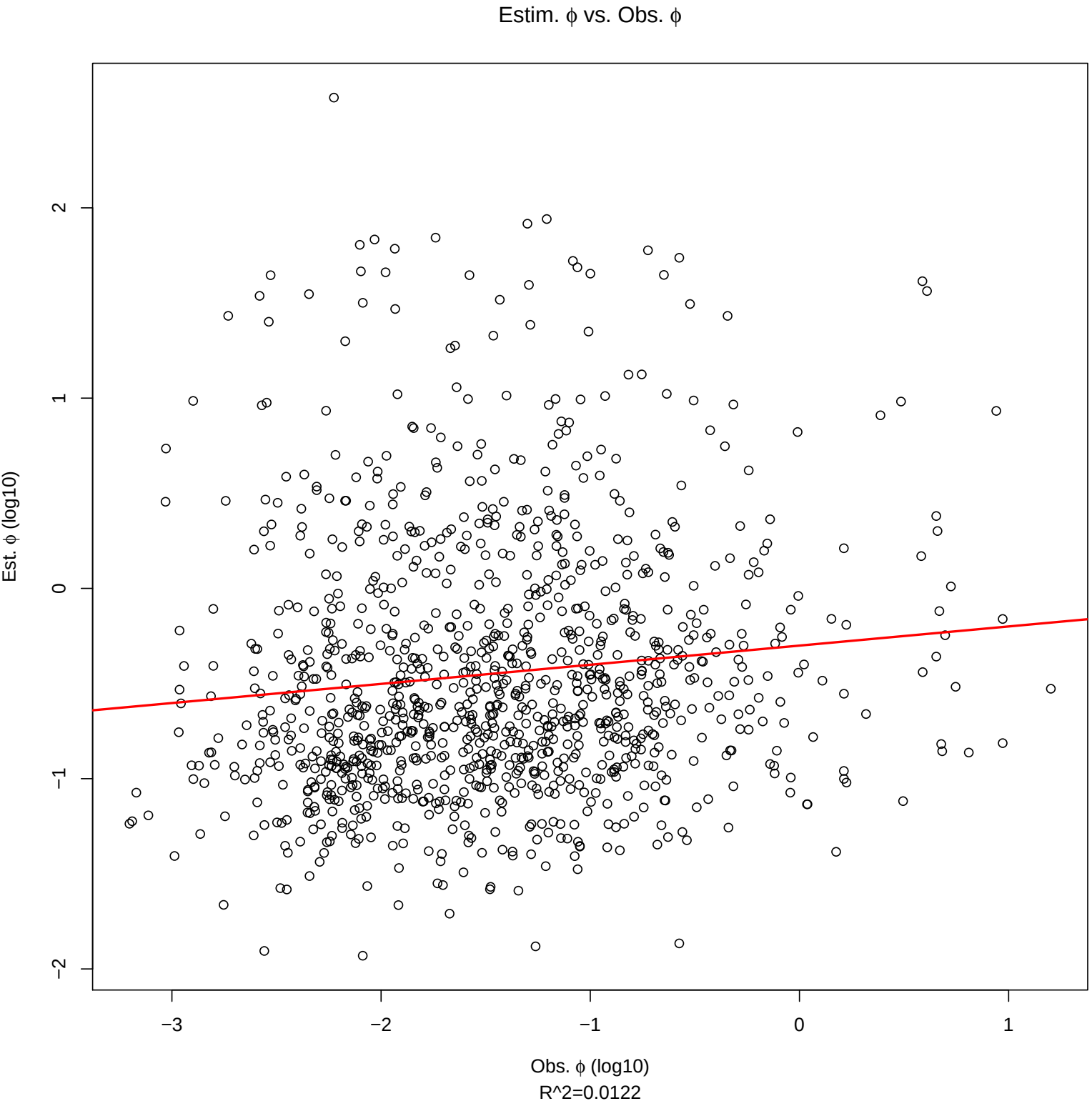

excluding unreliable X
Frequency
0
10
20
30
40
50
60
−3
−2
−1
0
1
emp. phi (log10)
mean=−1.47

AA parameter trace 1105

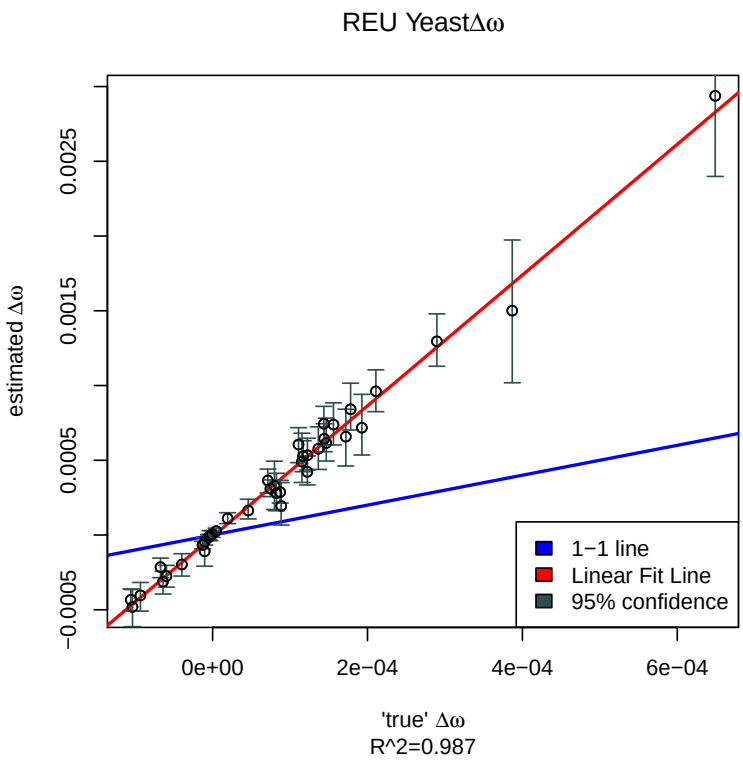
Wed Nov 5 16:04:13 2014

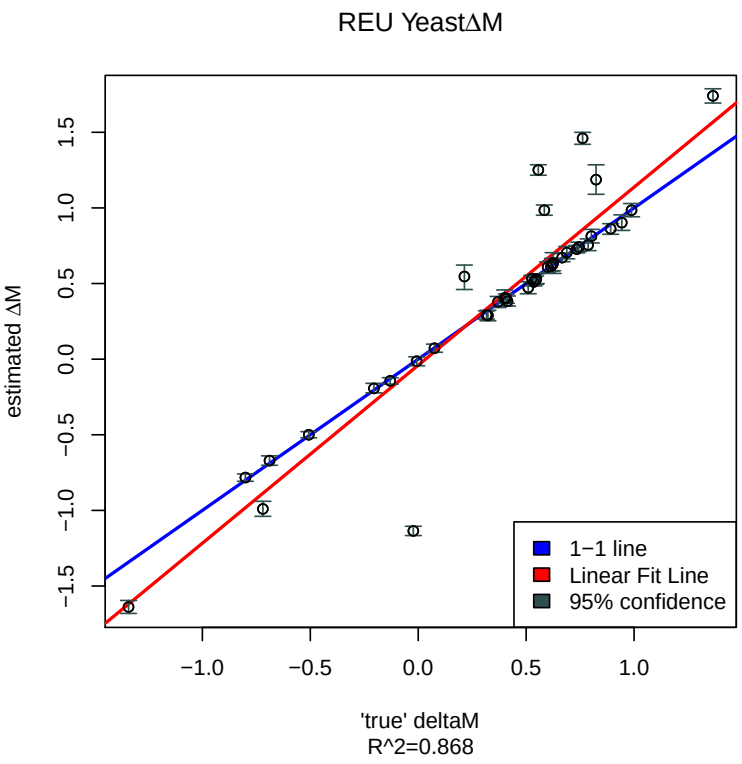


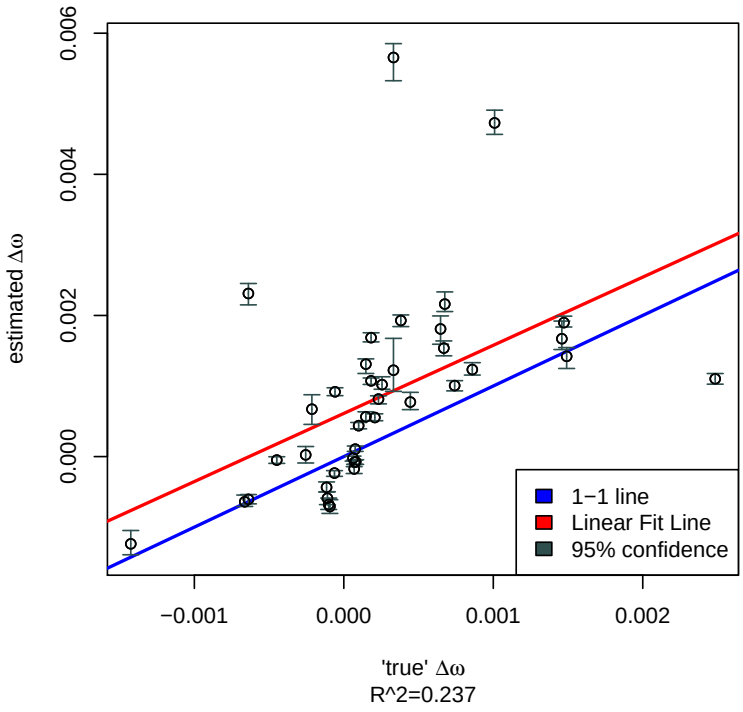

Hyperparameter Traces
σε
1.0
2.0
3.0
0
200
400
600
Samples
μϕ
−4
−2
0
200
400
600
Samples
σϕ
1.5
2.5
0
200
400
600
Samples
K
−2
0
1
0
200
400
600
Samples

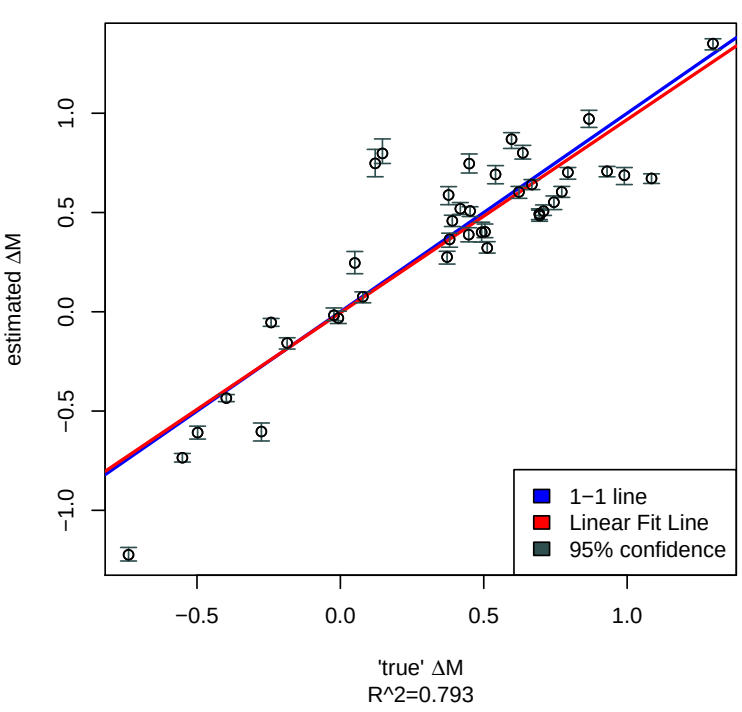

Estim. ϕ vs. Obs. ϕ
Est. ϕ (log10)
2
1
0
−1
−2
−3
−2
−1
0
1
Obs. ϕ (log10)
R^2=0.0122

Note how different the $\Delta\omega$ values are between Preston's yeast genome and the REU students' yeast genome.


REU YeastΔω
estimated Δω
'true' Δω
R^2=0.987
1−1 line
Linear Fit Line
95% confidence


REU YeastΔM
estimated ΔM
'true' deltaM
R^2=0.868
1−1 line
Linear Fit Line
95% confidence


Preston's YeastΔω
estimated Δω
'true' Δω
R^2=0.237
1−1 line
Linear Fit Line
95% confidence


Preston's YeastΔM
estimated ΔM
'true' ΔM
R^2=0.793
1−1 line
Linear Fit Line
95% confidence

2.2 Visualization

2.2.1 'true' values vs simulated values

Changes have been added to visualize.r, based on other plotting functions.

I've added confidence intervals (the scale of the interval can bet set in visualize.r). Cedric didn't have any functions to do so, but I was able to apply the "plotrix" package. I've also installed that package to "/home/lbrown/cubfits/Dependencies/plotrix". Everyone else should have permissions on that directory, in case someone wants to use my edited visualize.r function.

2.3 Parallelize the Code

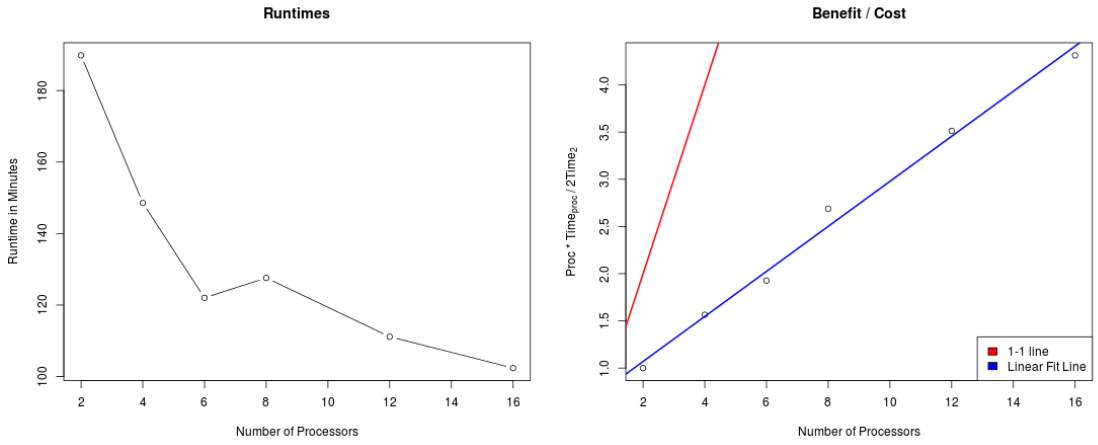
2.3.1 getOption("mc.cores")

How to set the number of cores for an mclapply call? mclapply's default number of cores is getOption("mc.cores",2L);

getOption("(option)", (value)) returns the value previously set to that (option), or otherwise it returns (value). mc.cores is not set by default. So first, set option("mc.cores"=Number_of_Cores Then mclapply should correctly get the number of cores.

2.3.2 Timing

As expected, we get diminshing returns on adding additional processors



2.4 Add (a1-a2) as a parameter

In the code, when the posterior probability of a codon is calculated, instead of calculating

$$\Pr(c_i|\phi,i) = \frac{\exp[\ln + \omega_i(a_1 - a_2)y_1 + \omega_i a_2 y_1 i]}{\exp[\ln \sum_{u=1}^{m} \exp[\ln + \omega_i(a_1 - a_2)y_1 + \omega_i a_2 y_1 i]}$$

Wei Chen calculates

$$\Pr(c_i|\phi, i) = \frac{\exp[\ln - \omega_i \phi i]}{\exp[\ln \sum_{u=1}^{m} \exp[\ln - \omega_i \phi i]}$$

This was done for a number of reasons. The y_1 term is just the aggregate of the effective population, ϕ, and a scaling term $-q$. Also, the assumption was that $a_1 \approx a_2 =$ 4ATP. To better account for the parameters of the model, we're going to add another parameter called $\Delta a_{12} = (a_1 - a_2)$, and use

$$\Pr(c_i|\phi, i) = \frac{\exp[\ln - \omega_i (\Delta a_{12})\phi - 4\omega_i \phi i]}{\exp[\ln \sum_{u=1}^{m} \exp[\ln - \omega_i (\Delta a_{12})\phi - 4\omega_i \phi i]}$$

The math part of the change takes place in my.logdmultinomCodOne.r, adding an extra row to xm and multiplying $\omega\phi\Delta a_{12}$, which is baa[2]*tmp.phi*(new parameter)

2.5 Wei Chen's Yeast / Real Yeast Genome

2.6 Generate my own simulated yeast, using a reverse engineered cubfits

3 Goals for next Month

1. Future Goal