Work Log for September

Logan Brown

September 19, 2014

3 Week of September 15th-19th

3.1 Goals for the Week

1. Simulated Data Set
2. First Order Approximation
3. Potentially look at the c files in cubfits/cubfits/src
4. Check for problems in Wei-Chen's NSE code, specifically, a crash caused by running out of memory
5. Continue to improve the optimal-pessimal code as Deepika works with it.

3.2 Progress/Notes

3.2.1 Simulated Data Set

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

As I understand it, these fasta genomes are ones that have been modified by Codon Evolution Simulation (CES). A quick comparison of b-1/S.cerevisiae.S288c.REU.sim.b-1.ces.fasta and ../S.cerevisiae.S288c.fasta shows that they have different codons. Additionally b-1's fasta file and b-0.01's fasta are different as well.

The folder b-1 vs b-0.1, etc is the setting of the B parameter. Look at '/export/home/clandere/CodonUsageBias/NSE/ces3/branches/exchange/C/data_simulation/CES.DATA.SIM for more details

I chose to use b-0.001, it had the best signs of actually converging to something. The eta values of the genes actually changed. For some b values, there wasn't an eta change. b-0.001/S.cerevisiae.S288c.REU.sim.b-0.001.evol.summary.tsv was the highest B value that had changes at every genome (Evolution Time != nan).

I was not able to find X_obs values for the yeast data in the REU data. I found ORF data in /opt/big_scratch/work-my, which is data from Wei-Chen/Yassour. I ran a recursive search through those files looking for xobs values, the output is found in smallfindXobs.txt

3.2.2 cubappr SimuYeast Run

Launched a cubappr run of the simulated yeast genome from the REU data, both for single chain and ~~multichain~~ (nothing happened for 2 hours. either it crashed, or changing the config.r file messed with the actual inner workings). If either/both crashes, I'll try again with a smaller run, likely just singlechain.

3.2.3 First Order Approximation

$$\prod_{j=i+1}^{n}(1-p_j) = \prod_{j=i+2}^{n} -p_{i+1}\left(\prod_{j=i+2}^{n}(1-p_{i+1})\right)$$

Simply to make things easier to read, I'll restate $\prod_{j=q}^{n}(1-p_j)$ as t_q. Note that in general,

$$t_q = t_{q+1} - p_q(t_{q+1})$$

So

$$\begin{aligned}\prod_{j=i+1}^{n}(1-p_j) &= t_{i+2} - p_{i+1}t_{i+2} \\ &= t_{i+3} - p_{i+2}t_{i+3} - p_{i+1}\left(t_{i+3} - p_{i+2}t_{i+3}\right) \\ &= t_{i+3} - p_{i+2}t_{i+3} - p_{i+1}t_{i+3} - p_{i+1}p_{i+2}t_{i+3}\end{aligned}$$

Since $p_{i+1} * p_{i+2} \approx 0$.

$$\approx t_{i+3} - p_{i+2}t_{i+3} - p_{i+1}t_{i+3}$$

Continuing iteratively...

$$\begin{aligned}\prod_{j=i+1}^{n}(1-p_j) &\approx t_n - \sum_{k=i+1}^{n-1} p_k(t_n) \\ &\approx (1-p_n) - \sum_{k=i+1}^{n-1} p_k(1-p_n) \\ &\approx 1 - \sum_{k=i+1}^{n} p_k\end{aligned}$$

3.2.4 Potentially look at the c files in cubfits/cubfits/src

They still look like they aren't doing anything.

3.2.5 Investigate WeiChen NSE crash

- Ending crash caused by running out of memory?

Cedric also suggests it could be due to a problem with the serialization step. When the code is run in parallel (by SNOW activating multiple copies of the same program with different initial conditions), it comes back to the serial to be tested for convergence. If the data structures are too large at this point, it may crash due to "running out of memory", even though Gauley has waaay more memory. It may be related to memory.c

I'm running in single chain from here on out for testing this hypothesis.

Single chain crashed, however, it actually told the error!

```
Error in phi.New[accept] <- prop$phi.Prop[accept] :
  NAs are not allowed in subscripted assignments
Calls: system.time ... do.call -> <Anonymous> -> my.drawPhiConditionalAllPred
Timing stopped at: 6251.473 43.7 6298.546
Execution halted
```

3.2.6 Compare NSE code to ROC code

Here are all the files that use the NSE model (results of grep -il nse /cubfits/cubfits/R/*)

~~my.coef.r~~

my.estimatePhiOne.r
my.fitMultinomOne.r

Here is where the vglm concerns are. Mostly not concerned, it doesn't look like any additional vglm calls.

my.logdmultinomCodOne.r

Here's my biggest concern.

my.logdmultinomCodOne.roc has three lines that my.logdmultinomCodOne.nsef does not lp.c.raw ¡- yaa * lp.vec lp.c.raw[is.nan(lp.c.raw)] ¡- NA lp.c.raw ¡- rowSums(yaa * lp.vec, na.rm = TRUE)

my.drawPhiConditionalAllPred calls my.logPosteriorAllPred.lognormal, which calls my.logdmultinomCodOne.nsef, which does not have the stated lines. Without those lines, if lpProp - lpCurr - prop$lir becomes NaN (log of a negative value, most likely), it would not pass any errors until the line that actually complains, accept ¡- u ¡ exp(logAcceptProb).

However logAcceptProb is lpProp - lpCurr - prop$lir

My best hypothesis is that lpProp (proposed phi values from the Posterior distribution) has some NaN values.

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

3.2.7 Build Local Cubfits

1. cd path/to/cubfits/..
2. R CMD build cubfits
3. install.package("_____.tar.gz", lib ="place to install to", repos = NULL, type="source")

3.2.8 Disect NSE data Structure

3.2.9 Profile the ROC model (where is the time?)

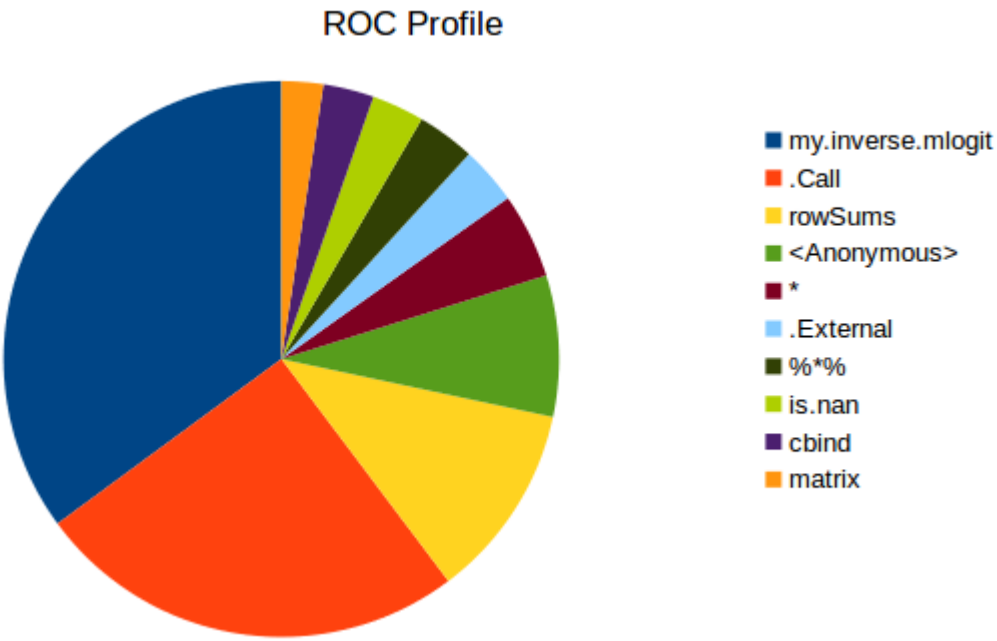
Look at the man page - ?Rprof

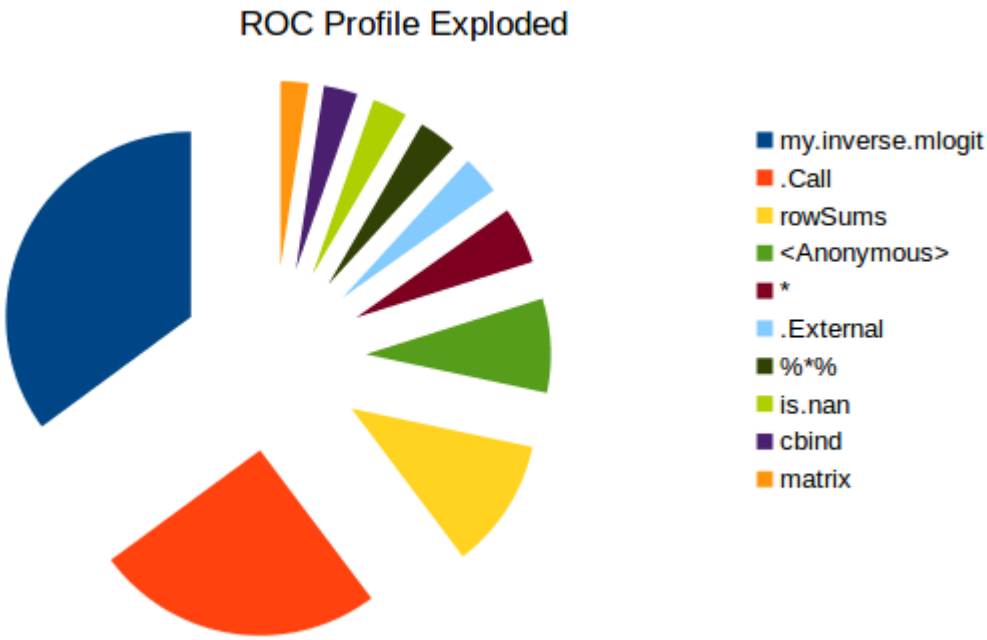
Run Rprof() before any code you want to profile, and Rprof(NULL) after the code, then run summaryRprof in R or R CMD Rprof from the command line to analyze the output.

Added Rprof() to the beginning of run_roc.r and Rprof(NULL) to the end of run_roc.r, then ran using ./workflow as usual.

These are the functions that took up more than 1% of the time on their own.

% self	self seconds	% total	total seconds	name
33.3	2123.86	60.5	3855.96	"my.inverse.mlogit"
23.9	1520.96	23.9	1520.98	".Call"
10.8	687.68	14.0	894.64	"rowSums"
7.8	496.74	98.7	6288.94	"<Anonymous>"
4.7	299.56	4.7	301.74	"*"
3.2	205.44	3.2	205.44	".External"
3.2	203.82	3.2	203.82	"%*%"
2.9	183.66	2.9	183.66	"is.nan"
2.8	178.62	2.8	178.84	"cbind"
2.3	143.80	4.6	294.34	"matrix"





.Call is only seen in the

3.2.10 Continue to improve the optimal-pessimal code as Deepika works with it.

3.3 Goals for next Week

1. Future Goal