Work Log for August

Logan Brown

August 29, 2014

4 Week of August 23rd-30th

4.1 Goals for the Week

1. Finish analyzing cubappr.r (especially the MCMC after line 158)
2. Look at the consequences if model is "nsef" instead of "roc"
3. REU Results
4. Data Structures in the R manual (list?)
5. **Liz Howell Code**
6. **Run CUBFITS**

4.2 Progress / Notes

4.2.1 Finish analyzing cubappr.r (especially the MCMC after line 158)

4.2.2 Look at the consequences if model is "nsef" instead of "roc"

4.2.3 REU Results

4.2.4 Data Structures in the R manual (list?)

List ended up being very crucial. I studied lists, factors, and tables.

Lists are essentially C++ classes, in that they are a collection of smaller parts. Dereferenced with $ sign.

Factors are something like a struct, but not quite. They are like a char or number, but it also has "levels". Slightly obnoxious.

Tables are just AWESOME. No C++ comparison. They are like a matrix, except they hold mixed inputs, and you can just read.table() to generate a table from a file. So crisp.

The next thing I'd want from the user manual is a better way to read in the input. Right now, I just preprocess the file to chop out all of the commas with :%s/,/ /

4.2.5 Liz Howell Code

"*The code I mentioned to Liz Howell that I need you to write is very simple. Ideally it would be in a language that is more common than R, but the world is not ideal and I think doing it in R would be best for now.*

The code would take a observed DNA sequence for an given protein and output a pessimal (worst) and optimal sequence based on the delta eta values produced by cubfits. The pessimal sequence would use the codons with the longest pausing times (largest delta eta) and the optimal would use the codons with the shortest pausing times.

Note that in the current code, the table produced is always relative to the codon with the shortest pausing time, but if I would recommend not assuming that is always the case. That is used min() and max() functions to ID the codons rather than ==0 and max().

To summarize, the code will take in a FASTA file with one or more genes and a delta eta file produced by cubfits. The output will be a FASTA file with the pessimal versions of the genes and a second FASTA file with the optimal versions of the genes.produce up to two output."

The code seems approachable. I can use basic.r to set up the reu13.df and phi.df data structures. The problem is, I need phi.obs, which is unclear. I'm going to try phi.df[,1].

~~COMPUTERS WENT DOWN.~~

3 step plan.

1. Using a given genome, generate Phi values (expression levels)

2. Using given Phi values and cubfits, generate δt values (pausing times)

3. Using δt values, generate optimal and pessimal genomes.

First priority is to generate δt values given Phi values. Then I'll work on the coding to begin and end the process.

8/29: I've written the code to analyze the .bmat file from Cedric's run_roc.r. The problem is, it only has the δt values for a collection of synonyms, and doesn't relate it at all to the actual genome. That's bad. The code I've written so far finds the optimal and pessimal codons based on the delta eta values, but cannot tell which one of the synonyms it chose

I'm looking at run_roc.r, and the functions it calls (cubsinglechain and cubmultichain) to find this information. As far as I can tell, this is going to require substantial changes to the file i/o

4.2.6 Run Cubfits

~~seq.data <- read.seq(get.expath("seqi_200.fasta"))~~
~~phi.df <- read.phi.df(get.expath("phi_200.tsv"))~~
~~aa.names <- c("A", "C", "D")~~
~~seq.string <- convert.seq.data.to.string(seq.data)~~

~~reu13.df <- gen.reu13.df(seq.string, phi.df, aa.names)~~
~~reu13.list.new <- gen.reu13.list(seq.string, aa.names)~~
~~y <- gen.y(seq.string, aa.names)~~
~~n <- gen.n(seq.string, aa.names)~~
~~scuo <- gen.scuo(seq.string, aa.names)~~
~~phi.obs = phi.df[,2]~~

~~All except the last one came from demo/basic.r~~
~~The last one is added by me.~~

Last Error Error in .cubfitsEnv$my.stop(paste("mean(phi.Obs) =", mean(phi.Obs))) :

paste("mean(phi.Obs) =", mean(phi.Obs))) runs just fine, so it seems like an error with my.stop

R/my.stop.r has the problem. ~~It seems related to lapply, mclappaly, pdbMPI, and parallel?~~ phi.Obs must be NORMALIZED. (Mike claims it shouldn't be normalized though? This could be due to an older iteration of the code)

Cedric has some scripts which generate and mold the data appropriately before a run. I've checked out the repository to cubfitsLocal/misc

UPDATE: After correctly modifying workflow.sh, I seem to be successfully running cubfits.

Terminal crashed (details in Pictures/829terminalcrash.png). Rerunning cubfits.

4.2.7 FIXING GAULEY

1. Slow Login
2. Cannot input password at login screen
3. slow tab completion and non-commands

Fixed by Cedric, Mike, and Mike.

4.3 Goals for next Week

1. Connect codons with delta eta values
2. Liz Howell Code
3. Further readings in the R user manual