Week of May 11

Jeremy Rogers
`jroger44@vols.utk.edu`

Week of May 11, 2015

Contents

1 Goals for the Week

1. Install LaTeX for these notes

2. Get access to the repository and Gauley

3. Read through all of Logan's previous notes

4. Brush up on R by looking through the user manual

5. Begin looking through the code on the repository

6. Create a repository for these notes
7. Get a Linux install working on my machine
8. Read up on MCMC
9. Begin running Logan's old code

2 Progress/Notes

2.1 Install LaTeX for these notes

1. LaTeXhas been installed and tested

2.2 Get access to the repository and Gauley

1. With Dr. Gilchrist, I was able to get access to both repositories (the one with Logan's notes and the one containing CUBfits). I also was granted an account on Gauley and Newton.

2.3 Read through all of Logan's previous notes

1. I should check out his script repository later, when I need to begin testing this. He says it's located at `https://github.com/ozway/cubmisc`
2. After reading through all the notes, it appears that his code is close to completion. There are a few things that he mentioned wanting to do, and I'm not sure if he got around to those.
3. He mentions moving some of the code to C from R, but he feels that it would be more work than it is worth.
4. He also mentions debugging the genome creation process by using a genome that is totally dominated by mutation bias, and see if the genome is correctly created across all phi values.
5. He also mentions changing some divisions to subtractions using logarithm rules, since that would be quicker.
6. Just some terminology that I need to remember:

(a) CUB – codon usage bias

(b) η – cost-benefit ratio of protein synthesis

(c) ϕ – protein synthesis rate

(d) N_e – population size

(e) ROC – ribosome overhead costs

(f) NSE – nonsense error

(g) ORFs – Open Reading Frames

(h) q – proportional decline in fitness per ATP wasted per unit time

(i) ΔM – mutation bias

(j) $E(\phi)$ – expected protein synthesis rate, should be 1 if time units are defined correctly

2.4 Read the R Manual

1. Read the first 3 chapters

2.5 Read through the code on the repository

1. I have begun to read through the main staples of the code (namely `my.cubappr.r` and `roc.appr.r`). This is a lot of code, in a language that I've only used in passing, so it could take a while to figure this out.

2.6 Create a repository for these notes

1. Repository has been created at `https://github.com/jeremyrogers/EEB`

2.7 Get a Linux Install working on my machine

1. This is a low priority, but I'd still like to get it done by the end of the week. According to the documentation on `CUBfits`, some of the parallel stuff might bug out on Windows.

2. Wednesday Update: I have a working linux install now, so that will dramatically help to test these things.

2.8 Read up on MCMC

1. Thanks to Stack Exchange's math section, I found a really nice explanation on MCMC. After reading that, I was able to parse through the rather technical Wikipedia page on the subject, and now I feel like I have a good grasp on this.

2.9 Run Logan's code

1. Logan's code from his `cubmisc` repository was pretty broken on my machine, but that could have just been my configuration. After fixing the initial errors I received, I began running his `run_roc.r` test script at 15:35 on Wednesday. I left at 17:35 and it was still running on Gauley.

2. Thursday morning at 10:55 when I got in, it had halted execution with this error:

   ```
   Error in my.set.adaptive(nIter + 1, n.aa = n.aa,
   b.DrawScale = b.DrawScale,  :
   length of p.DrawScale is incorrect.
   Calls: system.time ... cubsinglechain -> do.call ->
   <Anonymous> -> my.set.adaptive
   Execution halted
   ```

   I'm not really sure what any of this means, and it's entirely possible that this isn't his final code, being this other repository I found. However, this is currently the best lead I have on which code was Logan's. According to Cedric, there were 1 or 2 functions in the cubfits library that Logan wrote, but we're unsure of which ones they are.

3. I tried to run his `run_nsef.r`, but it halted with these errors:

   ```
   Error in phi.New[accept] <- prop$phi.Prop[accept] :
   NAs are not allowed in subscripted assignments
   Calls: system.time ... cubsinglechain -> do.call ->
   <Anonymous> -> my.drawPhiConditionalAll
   In addition: Warning messages:
   1: In dlnorm(phi.Obs, log(phi), sigmaW, log = TRUE) : NaNs produced
   2: In rnorm(1, mean = log.sigma.Phi.Curr, sd = sigma.Phi.DrawScale) :
   ```

```
NAs produced
3: In my.drawRestrictHP(proplist, list.Curr, phi.Curr) :
log acceptance probability not finite in hyperparam draw
4: In rnorm(1, mean = bias.Phi.Curr, sd = bias.Phi.DrawScale) :
NAs produced
5: In my.drawbiasPhi(proplist, list.Curr, log.phi.Obs, log.phi.Curr,  :
log acceptance probability not finite in hyperparam draw
Execution halted
```