Week of May 11

Jeremy Rogers
jroger44@vols.utk.edu

May, 2015

Contents

1 Goals for the Month

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
10. Set up the printer for timesheets

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. I have read the first 3 chapters. I feel like I have a good grasp on the R language and environment now.

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. May 13 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 May 13. I left at 17:35 and it was still running on Gauley.

2. The next 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
```

This will be fun.

4. After speaking with Dr. Gilchrist, I am aware that my main objective at the moment is to run the first order approximation of the NSE model on the genomes, both simulated and actual. For the simulated genomes, I must find the values used to see if we can reconstruct them accurately using the first order NSE approximation.

5. I've been tweaking Logan's NSE code to parallelize a little, since his wasn't parallelized at all. Hopefully this will take less time to run now. In order to make it parallel, all I had to do was near the beginning of the code, add `config$parallel <- "mclapply"`. I did `mclapply` since I'm running this on Gauley with shared memory. If I move this to Newton, which I probably will at some point, I'll have to use `task.pull`

6. I've been tweaking some settings to try to get this to work properly. I had a few different errors that were quick fixes, but this one is a head scratcher. Running the code, I get this error, the same one from before:

```
running cubappr using cubsinglechain
with seeds: 84953
Running in parallel mode mclapplyError 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
Timing stopped at: 22.393 4.374 18.18
Execution halted
```

Length of p.DrawScale is incorrect? I looked at my.adaptive.DrawScale, and found some pertinent lines of code. However, there are 4 places which print the `length of p.DrawScale is incorrect` error. So, now I have to figure out which one it is.

My first thought was to add print statements in each one to see which one was getting triggered. I did that, but there were problems building it locally on Gauley. First, LaTeX wasn't installed, and for some reason, that caused the build to fail. So I got cedric to install `texlive` for me. Once that was installed, it built properly. However, when I would try to run it with the local library, R couldn't find it. Maybe I just don't know enough about R. I'll look into that later.

Cedric mentioned using R's `browser()` function to figure out where I am in the code. I'll begin looking through the R manual to learn that function.

I've narrowed it down significantly now. Logan's `run_roc.r` calls a function called `cubsinglechain`. I used `grep` to find it in the `cubfits` source, and it's in the file `cedric.convergence.r`. From there, it appears that it's in line 221, which has `res <- do.call(cubfits, c(input_list, list(phi.Init = init.phi), list(p.Init = p.init), list(b.RInit = b.rinit), list(b.Init = b.init), list(b.DrawScale = b.DrawScale), list(p.DrawScale = p.DrawScale), list(phi.DrawScale = phi.DrawScale)))`

I'm unsure of what's going on here for now. I'll ask Cedric at some point.

I decided to try to build `cubfits` locally, with a print statement after setting `p.DrawScale`. After adding that, it magically is working now. I began running it at 12:14, so I'll see where this goes.

Maybe this actually works sometimes, and I just got lucky with it working this time after adding a line. Or maybe the version of `cubfits` I built locally is different than the one I got from `CRAN`. Regardless, it's

still running now at 13:37, so hopefully it will stay this way. For the record, I'm running this with the Brewers' Yeast genome, with initial phi values given in the same directory.

2.10 Set up the printer

1. It was brought to my attention that I should set up the printer in the lab to print timesheets, and anything else I might want. This proved to be a daunting task. I printed out the network configuration on the printer to get the IP address. On my machine, I was able to ping the printer but unable to actually connect to it. So, I got the `.ppd` files from Brother's website to get the configuration. Eventually I got to a point where my machine would say it was sending the files, and then give a confirmation that it's printed, but the printer wouldn't react. I wonder if `CUPS` is sending the files to some other directory that isn't the actual printing spool, and that's what is giving the false positive printing confirmation. Either way, I think I'll just use my home printer for timesheets. I may come back to this later, but right now it's a low priority.