

## Laboratory Worksheet, Friday, March 26.

As always, there are several projects to work our way through today.

**I. Putative Highly Expressed Genes (PHX).** We are starting to get real data. Jayson Falkner's Java script is there, as well as several proofs that it works on *H. influenzae* datafiles. There is also the link to a Perl script which do much of the same.

Storage should be in the worksite *Math 548 001* which you should all belong to automatically at <http://ctng.umm.umich.edu>. There is a link on the 548 Labworksheets page where you get these weekly bulletins. Go to this page and make sure that you can access this site. You will find here under *Resources* (a link from the worksite home page) a set of empty folders. There are also folders for gathering the code you have been writing about component parts of the project.

As I understand this system (by analogy with the old system it replaces), student participants should be able to do anything within this Workspace except to add new participants. Let me know if you want somebody else to get involved with these projects.

**II. Training an HMM.** [Mainly a reprint.] This was mainly not finished last time. This will use the *new* command *dhmmem.m* in the *HMM* toolbox. There was a problem with this command because it contains an external, uncompiled C-command which wasn't contained in the earlier version (*learn\_dhmm.m*). There are two possible fixes. Either edit the command file *dhmmem.m* where it invokes "normalize" with a C-suffix, and convert it to the corresponding *normalize.m* from KPMtools. Or you can download the old *learn\_dhmm.m* from the 548 web resources page. This does not involve the C-form of the normalize command.

I will only sketch the procedure here. You should first pull up the file for the command from the *HMM* directory (that is, the HMM toolbox, which should be at /Desktop/KPM/HMM) and read it. What are the inputs? What is the structure of the *data* input? We will have data for training a model where we will fix upon 2 hidden states beforehand, each with 6 emissions. Given this simple architecture of the model we are looking for, how many parameters are we looking to estimate in order to train the HMM model?

We will get the data by simply taking our original data sequence *O* and permuting it. Matlab has a random permutation generator called *RANDPERM* (you can look it up in the help window to see what its arguments are). The proposal is to permute *O* randomly about 50 or 60 times, and build an array of data sequences which can be used for Baum-Welch (EM) training.

**Assuming** you have the above running, what do the trained values look like? Try looking at several initial estimates of the parameters, and see how the final estimates look depending on the initial conditions. Recall that this trainer does NOT have a randomizer, i.e., does not use a Monte Carlo method for taking a random alternate trial every so

often, so if you want to avoid *local* critical points, you have to try different initial points by hand. Try also running alternative numbers of repetitions in the EM algorithm. the method for specifying more iterations as an option is explained within the .m file for the learn command.

**III. Genetic code evolution, early amino acids.** This is more a warning for a future project. The 548 resource page contains a subdirectory on this project:

<http://www.math.lsa.umich.edu/dburns/548/rnasyn/>. There are a collection of materials there now concerning the temporal hierarchy and the evolution of the genetic code, especially by E. Trifonov. Roughly speaking, the synthase version of this project would be to take the amino acid synthase of a protein “recent” on the consensus dating scheme (cf., p.9 of Trifonov’s lecture slides [.../548/rnasyn/trifITP.pdf](http://www.math.lsa.umich.edu/dburns/548/rnasyn/trifITP.pdf) in the subdirectory), and find that the consensus a.a.-synthase for that residue only involved “older” residues. (The Miller-Urey “abiotic” residues are highlighted by a red dot in the reference above.) There is data there which last year’s class helped gather on the TrpB-synthase across species (TrpB = the beta subunit of tryptophan synthase, the component directly responsible for the synthesis of tryptophan from serine and indole). It is found under [.../rnasyn/TrpBsynthasedata](http://www.math.lsa.umich.edu/dburns/548/rnasyn/TrpBsynthasedata) in the 548 resources directory.

There is still a valuable reference for novices on this subject in the paper of Weber and Miller from 1981, “Reasons for the twenty coded protein amino acids”. I will get this old paper scanned and mounted in the 548 directory. Today I will give a brief intro to the problem.

Note also that some of the data collected in the subdirectory rnasyn deals with *aminoacyl-tRNA synthetases*. These, too, are ancient proteins, related to the RNA World Hypothesis, but their evolution is much more complicated and subtle than that of the aminoacyl synthases. We may look at those, but the data should be clearer and more interesting at first than that for the synthetases.

## Appendix: Software Fixes.

**A. The Matlab Fix.** Go to the *548 Resources* link from the course home page. Download the file *KPM.tar.gz* onto your desktop. In the Mac’s these should mount themselves on the desktop. Now call Matlab from the dock (it is the brown cone shaped icon in the bar at the bottom of the desktop). At the command line prompt (which looks a bit like `>>`) type `addpath ~/Desktop/KPM/HMM [return]`, and so on, according to the names of the four downloads (`.../KPM/KPMtools`, `KPMstats`, `netlab`). This sets the paths to these functions for the permanently mounted Matlab in the lab. These mounted toolboxes will *disappear* when you log off. You can save them in your personal space, if you have a way to use them within Matlab there. I have not yet double checked the new addpaths since incorporating all the components for the HMM download into one local download.

*Please note that there is a Licence agreement which you are accepting for the Nabney netlab package.*

**B. The SeqAln Fix.** The local systems people have stored the Waterman program suite *SeqAln* locally at <http://www.math.lsa.umich.edu/courses/seqaln-2.0.dmg> as a disk image. Download this file to your desktop, where it becomes an icon. From the systems person here: “This is a MacOSX Jaguar binary, and the executable files can be

found in that tree. So...students have to download and mount the dmg file and then type:  
/Volumes/seqaln - 2.0/seqaln/bin/SOMEPROGRAM to run a program. Hope this helps.”  
This fix has been tested and works.