About Me

Ellen Campbell

Table of Contents

# Who I am and where I came from

I grew up in a small farm town near UC Davis. When I was about three I watched a PBS Nature program called “Incredible Suckers” on cephalopods and decided then and there that I wanted to study squid when I grew up. As I learned more, my interests broadened to math and science in general (although I still love squid!). I attended [UC Santa Cruz](https://www.ucsc.edu/), and while I toyed with the idea of majoring in math, I ultimately majored in biology with a minor in applied math and statistics.  
When I graduated in 2014, I joined Carlos Garza’s lab as one of his three full time lab staff. I attended Eric’s version of this class two years ago and found it super interesting and incredibly helpful for honing my UNIX/terminal and R skills and getting a deeper understanding of our sequencing processing pipeline.

When I’m not working, I love to:

1. Bake whatever interesting new food experiments I can find
2. Brew beers, ciders, and mead with Akshar
3. Go hiking!
4. Knit

Here’s a picture of my adorable cat, since apparently the only photos I have on my computer right now are of her



# Research Interests

I really enjoy the more trouble-shooting/practical work that we do in the lab. I’m less invested in a particular topic, and generally enjoy the more specific questions we approach in lab, like why a specific protocol has stopped working, or how do we assess the quality of genotypes for types of genotyping data that we haven’t generated before. I’ve really enjoyed being part of the lab’s transition towards sequencing-based genotyping. I’ve loved getting to handle GTseq data and the sneak peaks I got into the development of the snakemake pipeline and am really excited about our current dive into whole genome sequencing runs!

## Influential papers

Describe two papers that have been very influential in your research. The main purpose of this is learning how to cite literature in RMarkdown. You will have to add the citations to references.bib in BibTeX format, and then also cite them.

An easy way to get the paper in BibTeX format is to find it on Google Scholar. Then, under the paper’s link, click on the big fat quotation marks, and then choose the BibTex link at the bottom of the page. Then copy the contents into references.bib. For example you might paste this into references.bib.

@article{richardson1997bayesian,  
 title={On Bayesian analysis of mixtures with an unknown number of components (with discussion)},  
 author={Richardson, Sylvia and Green, Peter J},  
 journal={Journal of the Royal Statistical Society: series B (statistical methodology)},  
 volume={59},  
 number={4},  
 pages={731--792},  
 year={1997},  
 publisher={Wiley Online Library}  
}

Note the string before the comma on the first line. That is the cite key. Once you have done that, you can cite it by including @richardson1997bayesian if you don’t want the name in parenthesis, like, “As Richardson and Green (1997) introduced in their landmark paper.” On the other hand, you can use [@richardson1997bayesian] if you do want it all in parentheses, like, “Reversible jump MCMC has been previously applied to the analysis of mixtures (Richardson and Green 1997).”

## The mathematics behind my research

It is easy to write complex mathematics using a mathematical typesetting engine called TeX. Write two displayed equations that are relevant to your work. For a quick primer on TeX and LaTeX you can download [this cheatsheet](https://users.dickinson.edu/~richesod/latex/latexcheatsheet.pdf) and then fiddle with including things between the double dollar signs below:

or here:

## My computing experience

I’ve been using R since my undergrad days, but was only introduced to the tidyverse world of R after I graduated. I find it immensely useful for stripping through data and checking for common issues (assessing sample/genotype quality; assessing assay/reagent success) and less common issues (like the whole 9D debocle).

Here’s an ugly loopy chunk of R code that I want to rewrite in a tidier format, but every time I try, it slows things down!

#This input has a sample name column with a unique ID--lower the ID, the earlier the sample was run  
#I'll be arranging by sample name so that for each rerun, the earlier run comes first. You could arrange by run number, total number of reads, etc. Just make sure col 1 = NMFS\_ID, col 2 = what to sort reruns on, col3+ = genotypes  
genotypes <- as.data.frame(Together)  
#Create a vector containing each of the sample names only once  
samples <- as.vector(unique(genotypes[ , 1]))  
#Create a matrix to hold all of the consensus genotypes and the sample names  
consensus <- matrix(nrow = length(samples), ncol = (length(genotypes[1, ])+1)/2)  
#set the first column to be the sample names  
consensus[ , 1] <- samples  
#Start looping over samples  
for (i in c(1:length(samples))){  
 #Find the indicies for the two runs of sample i  
 indicies <- c(1:length(genotypes[,1]))[genotypes[ , 1] == samples[i]]  
 #Start looping over assays (increment by 2 as there are two columns per assay)  
 for (j in seq.int(3, length(genotypes[1, ]), 2)){  
 #We'll be comparing sums of the two columns to make things easy, so calculate the sums  
 a <- genotypes[indicies[1], j]+genotypes[indicies[1], j+1]  
 b <- genotypes[indicies[2], j]+genotypes[indicies[2], j+1]  
 #If the two runs result in the same genotype note as "S" (same), if mismatched note as   
 #"M" (mismatch), if one nocalled but other called note as "L" (loss)  
 if (a==b){  
 consensus[i,((j/2)+1)] <- "S"  
 } else if (a == 0){  
 consensus[i,((j/2)+1)] <- "C"  
 } else if (b == 0){  
 consensus[i,((j/2)+1)] <- "L"  
 } else if (genotypes[indicies[1], j]==genotypes[indicies[1], j+1]){  
 if (genotypes[indicies[2], j]==genotypes[indicies[2], j+1]){  
 consensus[i,((j/2)+1)] <- "XXtoYY"  
 } else {  
 consensus[i,((j/2)+1)] <- "XXtoXY"  
 }  
 } else if (genotypes[indicies[2], j]==genotypes[indicies[2], j+1]){  
 consensus[i,((j/2)+1)] <- "XYtoXX"  
 } else if (sum(genotypes[indicies[1], c(j, j+1)] %in% genotypes[indicies[2], c(j, j+1)]) > 0){  
 consensus[i,((j/2)+1)] <- "XYtoXZ"  
 } else {  
 consensus[i,((j/2)+1)] <- "XYtoWZ"  
 }  
 }  
}  
  
#Store the consensus score matrix as a data frame  
consensus <- as.data.frame(consensus)  
#Set the names of the columns to be the same names as the assays  
names(consensus) <- names(genotypes)[c(1, seq(3, length(genotypes[1, ]), 2))]  
#Write to file  
#write.csv(consensus, file = "Differences.csv", row.names = F)

I’ve been slowly dipping my toes into python programming over the past couple years. I was briefly introduced in my undergrad years, but only formally started learning in the past two years. I’ve found it’s useful for automating bits of our lab process (like merging metadata files with the code I’ve written below to replace our RDBMerge excel plugin which, sadly, has ceased working)

#Let's see if we can write a script that pulls the metadata we need out of  
#metadata excel files  
  
#Pull in libraries needed  
import pandas as pd  
import os  
  
#Avoid truncating long numbers by setting float format to display 12 digits  
pd.options.display.float\_format = "{:.12f}".format  
  
#Find all the files in the current directory  
files = sorted([file for file in os.listdir("./") if file.endswith(('.xlsm', '.xlsx', '.xls'))])  
  
#Read in files!!  
  
#Initialize data frame to hold merged metadata  
Repository = pd.DataFrame()  
Freshwater = pd.DataFrame()  
Marine = pd.DataFrame()  
  
  
#Loop over files  
for file in files:  
 #Read in repo data  
 AllTabs = pd.read\_excel(file, sheet\_name = ["Repository","Freshwater", "Marine"], index\_col = 0)  
 Repository = pd.concat([Repository,AllTabs["Repository"]])  
 Freshwater = pd.concat([Freshwater,AllTabs["Freshwater"]])  
 Marine = pd.concat([Marine,AllTabs["Marine"]])  
  
  
writer = pd.ExcelWriter("Merged.xlsx", engine = 'openpyxl')  
Repository.to\_excel(writer, sheet\_name = 'Repository', index = True, header = True)  
Freshwater.to\_excel(writer, sheet\_name = 'Freshwater', index = True, header = True)  
Marine.to\_excel(writer, sheet\_name = 'Marine', index = True, header = True)  
writer.save()  
writer.close()

I’ve been intermittently working on a python script that I’m hoping will eventually replace the microsatellite toolkit plugin in excel, which I’m worried may not work on our computers for much longer. I’ve currently stalled on reformatting genotype data into genepop format since apparently the original mstoolkit did not do this entirely correctly, but I’ll try to pick this up again the next time I have a lull in lab work.

## What I hope to get out of this class

Three things I’d like to get out of this class

* Hone my R and UNIX skills. I’ve used both enough now that I feel comfortable working in both settings, but I know there’s so much more they can do that I just don’t know of yet so excited to learn more things about what I can do with them!
* Become more comfortable on computing clusters and with SLURM. This was probably the part of the last class that I struggled with the most, and I haven’t used it much since then, so hoping I can become more comfortable just working in that environment.
* Get motivated to tackle new problems! Last class definitely motivated me to find new ways of tackling new problems (like how do we generate a reference VCF for coho in an organized/thoughtful fashion; what bits of excel plugins are breaking and how can we rewrite them in R (which ultimately drove me to learn how to try to start fixing them with python))

# Evaluating some R code

Here, please evaluate some R code to make a plot or a figure. The goal here is just to realize that you can imbed R code within fenced code blocks and get the output rendered into the document.

# Citations

Richardson, Sylvia, and Peter J Green. 1997. “On Bayesian Analysis of Mixtures with an Unknown Number of Components (with Discussion).” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 59 (4): 731–92.