# About Me

Eric C. Anderson

## Contents



## Who I am and where I came from

I grew up in a small town in Southern California. When I was 11 years old, I spent five weeks living with a family friend who ran the hyperbaric chamber adjacent to the University of Southern California Marine Lab on Catalina Island. Days spent snorkeling the kelp beds, keeping intertidal organisms in an aquarium I was allowed to fill, and watching researchers tag blue sharks convinced me that I wanted to be a marine biologist.

In high school, however, I got into rock climbing and became a little math/physics nerd. I went to Stanford University planning to major in Mathematics, but got super intimidated by all the smart, wonky people in my first honors math class. I took a year-long detour to Prescott College, where I had my first class in evolution and wondered why the hell my high-school biology class was not taught in the context of evolution.

I ended up studying human biology at Stanford, and then went to University of Washington for graduate work. I started in Fisheries, but then got involved in genetics research that needed some new statistical developments. Thus, my math nerdiness was resurrected and I did my PhD in Quantitative Ecology and Resource Management, focusing on the use of Monte Carlo methods for inference from population genetic models.

After a two-year postdoc at Berkeley, I started working for the National Marine Fisheries Service in Santa Cruz, CA. I actually did become a marine biologist, sort of! I still work for NMFS, but five and a half years ago I moved to Fort Collins with Kristen, and have now have affiliate position in Biology and in FWCB.

When I am not working I love getting out and being active. My top four things to do are:

1. Snorkeling in rivers and creeks, backpacking, and hiking (all with my family),
2. Figure skating (I learned to skate after moving here. What a blast!),
3. Biking, both road and mountain,
4. Playing with our pair of awesome, springy cats.

Here is a picture of me with daughter Zoë, looking for aquatic invertebrates above the CSU Mountain Campus.

# Research Interests

I'm interested in all manner of statistical inference from genetic data. Lately I have been working on the genetic basis of run timing in Chinook salmon and other salmonids.

## Influential papers

When I was a graduate student, I heard Peter Green speak about his work on reversible jump MCMC for the analysis of finite mixture models. One of the problems I was working with at the time was estimating proportions of salmon from different rivers that were being caught in the ocean—the mixed stock fishery problem. I spent a lot of time working through Richardson and Green (1997) and learned a lot about MCMC and RJMCMC in the process.

Later on, much of my work on Bayesian inference of pedigrees from genetic data (Anderson and Ng 2016) builds upon the idea of factor graphs described by Kschischang et al. (2001).

## The mathematics behind my research

I have worked a lot with the coalescent process, so let's put down the expected time during which there are $k$ extant lineages in a population of size $N$.

$$\mathbb{E}T_k = \frac{4N}{k(k-1)}.$$

And, while we are at, let's throw down a description of one of the update steps in the sum-product algorithm for acyclic factor graphs:

$$\mu_{f_j \longrightarrow v_i}(x_i) = \sum_{x_{C\setminus i} \in \mathcal{X}_{C\setminus i}} h_j(x_{C\setminus i}, x_i) \prod_{k \in C\setminus i} \mu_{v_k \longrightarrow f_j}(x_k).$$

And, to really update this document, I am going to throw in some equations deriving Fisher information for the WGSassign paper that I did with Matt deSaix, Marina Rodriguez and Kristen Ruegg, that is set to come out in *Methods in Ecology and Evolution* sometime soon...

We start by finding the first derivative:

$$\frac{\partial L_i(\theta)}{\partial \theta} = \frac{\partial}{\partial \theta} \log\Big[ g_0(1-\theta)^2 + g_1 2\theta(1-\theta) + g_2\theta^2 \Big].$$

Let

$$\begin{aligned} u &= g_0(1-\theta)^2 + g_1 2\theta(1-\theta) + g_2\theta^2 \\ &= g_0(1 - 2\theta + \theta^2) + g_1(2\theta - 2\theta^2) + g_2\theta^2, \end{aligned}$$

and note that

$$\begin{aligned} \frac{\partial u}{\partial \theta} &= g_0(2\theta - 2) + g_1(2 - 4\theta) + g_2 2\theta \\ &= 2\theta(g_0 + g_2 - 2g_1) + 2(g_1 - g_0). \end{aligned}$$

Since $\partial \log(u)/\partial\theta = (\partial u/\partial\theta)u^{-1}$, we have that

$$\frac{\partial L_i(\theta)}{\partial \theta} = \Big( 2\theta(g_0 + g_2 - 2g_1) + 2(g_1 - g_0) \Big) \Big( g_0(1-\theta)^2 + g_1 2\theta(1-\theta) + g_2\theta^2 \Big)^{-1}.$$

Proceeding, define $v$ and $w$ as follows:

$$\begin{aligned} v &= 2\theta(g_{i,0} + g_{i,2} - 2g_{i,1}) + 2(g_{i,1} - g_{i,0}) &&= \frac{\partial u}{\partial \theta} \\ w &= \Big( g_{i,0}(1-\theta)^2 + g_{i,1} 2\theta(1-\theta) + g_{i,2}\theta^2 \Big)^{-1} &&= u^{-1}, \end{aligned}$$

and note that we can rewrite $\frac{\partial L_i(\theta)}{\partial \theta} = vw$, and take the derivative of that easily using the product rule: $(vw)' = v'w + w'v$. To do so, we first find the derivatives

$$v' = \frac{\partial v}{\partial \theta} = 2(g_0 + g_2 - 2g_1)$$
$$w' = \frac{\partial w}{\partial \theta} = -u^{-2}\frac{\partial u}{\partial \theta} = -u^{-2}v,$$

then we put them together with the product rule

$$\begin{aligned}\frac{\partial^2 L_i(\theta)}{\partial \theta^2} &= v'w + vw' = \frac{v'}{u} - \frac{v^2}{u^2} \\ &= \frac{2(g_0 + g_2 - 2g_1)}{g_0(1-\theta)^2 + g_1 2\theta(1-\theta) + g_2\theta^2} - \left(\frac{2\theta(g_0 + g_2 - 2g_1) + 2(g_1 - g_0)}{g_0(1-\theta)^2 + g_1 2\theta(1-\theta) + g_2\theta^2}\right)^2.\end{aligned}$$

Restoring the $_{k,\ell}$ subscript to $\theta$, and the $^{(i)}$ superscript and $\ell$ subscript to $g$, negating, taking the sum over the $n_k$ individuals and evaluating at the MLE yields $I_o^{(i)}(\theta_{k,\ell})$.

## My computing experience

I started programming in BASIC on our old Apple IIe in 1983. In high school I implemented a basic program to plot some fractal images. After that, I didn't really do any programming until grad school when I took a course in C. Here is some C code that I wrote:

```
    if(RU!=NULL) {
        RepUnitZSum = (int *)calloc(RU->NumRepUnits,sizeof(int));
        RepUnitPis = DvalVector(0,RU->NumRepUnits, 0.0, 1.0, .01);
        RepUnitPofZs = (dval ***)calloc(N,sizeof(dval **));
        for(i=0;i<N;i++) {
            RepUnitPofZs[i] = DvalVector(0,RU->NumRepUnits-1,  0.0,1.0,   -1);  /* no histograms for th
        }
        if(BO->PiTraceInterval>0) {
            repPi_tracef = OpenTraceFile("rep_unit_pi_trace.txt", "trace file of reporting unit Pi valu
        }
        if(BO->ZSumTraceInterval>0) {
            repZSumtracef = OpenTraceFile("rep_unit_zsum_trace.txt", "trace file of reporting unit ZSum
        }
    }
```

Wow! That is pretty ugly.

When I was a postdoc, John Novembre and the other members of Monty Slatkin's lab at Berkeley got me hooked on using the Unix shell, programming in bash, and writing short scripts in awk and sed. Here is a little awk script that takes the output of SGE's `qacct` command and makes a nice, tidy table of it

```
#! /usr/bin/awk -f

# an awk script.
# it expects the output of qacct like this:
# qacct -o eriq -b 09271925  -j ml

# make it executable and run it like this:
# qacct -o eriq -b 09271925  -j ml | tidy-qacct


# if you pass it a job number that was not one of your jobs it
# just skips the error message that comes up.
```

```
# note that the output of qacct is space delimited


/^==========/ {++n; next}  # increment run counter, then skip these lines
/^error:/ {next}  # skip it if you told it to get a wrong job number

# now, every data line it gets things.  It compiles the header
# all the way through, in case some reports have more columns...
NF > 1 {
  tag = $1;
  if(!(tag in header_vals)) {
    header[++h] = tag;
    header_vals[tag]++;
  }
  $1 = "";  # remove the tag from the full line of stuff
  values[n, tag] = $0;  # assign the values to the tag

}

# at the end of it all, we print the header and then all the values:
END {
  # print the header
  printf("%s", header[1]);
  for(i=2;i<=h;i++)
    printf("\t%s", header[i]);
  printf("\n");

  # cycle over individuals and fields and print em
  for(j=1;j<=n;j++) {
    printf("%s", values[j,header[1]]);
    for(i=2;i<=h;i++)
      printf("\t%s", values[j,header[i]]);
    printf("\n");
  }
}
```

I used to rather dislike the R programming language and felt it was dreadfully slow. It has gotten a lot better in the last two decades. The introduction of Hadley Wickham's tidy data analysis framework has really improved things.

### What I hope to get out of this class

I hope that I will:

- Help students understand enough about Unix and programming to lessen the pain of learning to do bioinformatics.
- Be able to advance students' own research.
- Impart to the students an appreciation of the importance of making research reproducible.

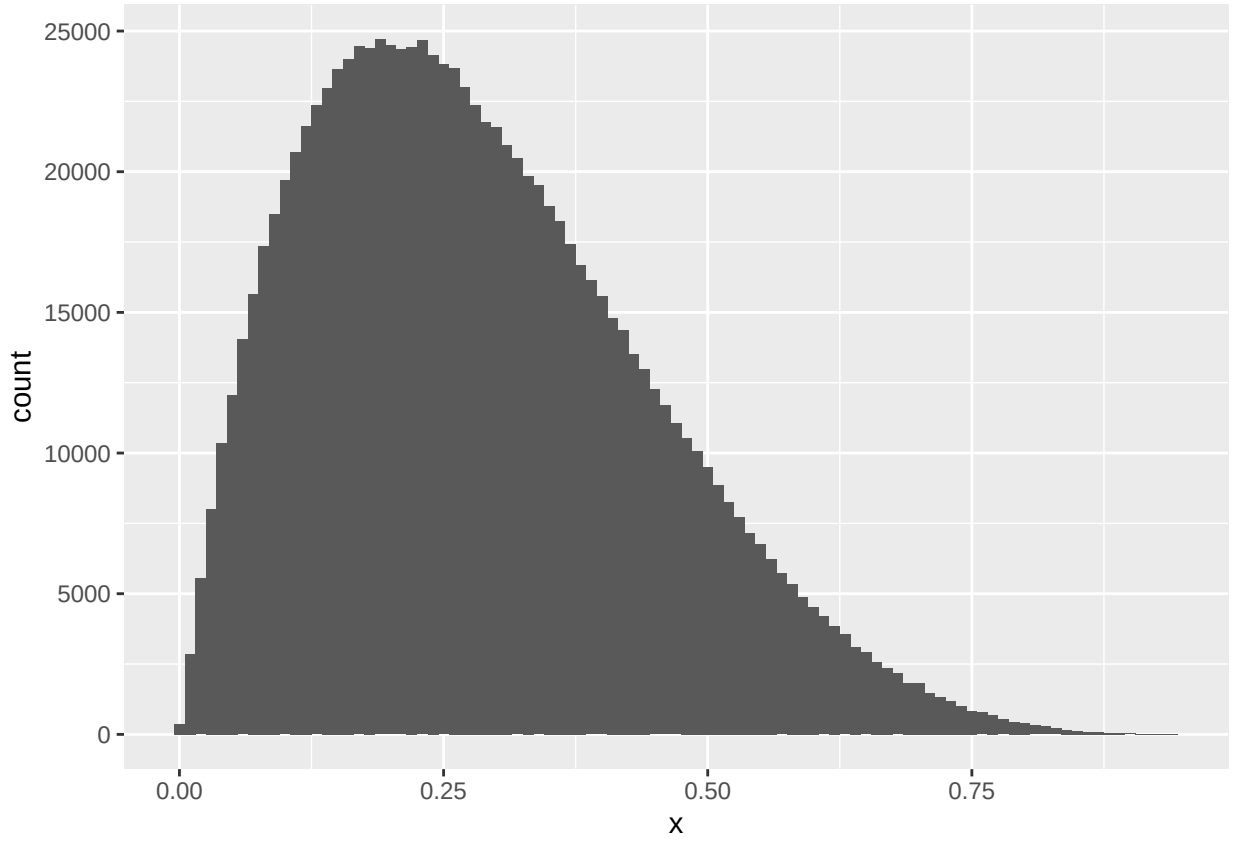
## Evaluating some R code

I'm going to just simulate one million beta random variables from a Beta$(2, 5)$ distribution and plot a histogram of it.

```r
library(tidyverse)

beta_rvs <- tibble(
  x = rbeta(1e06, shape1 = 2, shape2 = 5)
)

ggplot(beta_rvs, aes(x = x)) +
  geom_histogram(binwidth = 0.01)
```




## Citations


Anderson, Eric C, and Thomas C Ng. 2016. "Bayesian Pedigree Inference with Small Numbers of Single Nucleotide Polymorphisms via a Factor-Graph Representation." *Theoretical Population Biology* 107: 39–51.

Kschischang, Frank R, Brendan J Frey, Hans-Andrea Loeliger, et al. 2001. "Factor Graphs and the Sum-Product Algorithm." *IEEE Transactions on Information Theory* 47 (2): 498–519.

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.