

ESTIMATING ALLELE FREQUENCIES

THE EM ALGORITHM

Estimating the frequency of alleles is easy when you can identify all of the alleles within genotypes, but let's work through it slowly anyway. Suppose this is what our sample looks like:

Genotype	A_1A_1	A_1A_2	A_2A_2
No. in sample	n_{11}	n_{12}	n_{22}

We all know that the frequency of A_1 , p , is

$$p = \frac{2n_{11} + n_{12}}{2(n_{11} + n_{12} + n_{22})} .$$

Suppose, however, that we're trying to estimate allele frequencies in the ABO blood group system in humans. Then we have a situation that looks like this:

Phenotype	A	B	AB	O
Genotype(s)	aa ao	bb bo	ab	oo
No. in sample	n_A	n_B	N_{AB}	N_O

Now we can't directly count the number of a , b , and o alleles. What do we do? Well, if we knew p_a , p_b , and p_o , we could figure out how many individuals with the A phenotype have the aa genotype and how many have the ao phenotype, namely

$$\begin{aligned} N_{aa} &= n_A \left(\frac{p_a^2}{p_a^2 + 2p_ap_o} \right) \\ N_{ao} &= n_A \left(\frac{2p_ap_o}{p_a^2 + 2p_ap_o} \right) . \end{aligned}$$

Obviously we could do the same thing for the B phenotype:

$$\begin{aligned} N_{bb} &= n_B \left(\frac{p_b^2}{p_b^2 + 2p_bp_o} \right) \\ N_{bo} &= n_B \left(\frac{2p_bp_o}{p_b^2 + 2p_bp_o} \right) . \end{aligned}$$

Notice that $N_{ab} = N_{AB}$ and $N_{oo} = N_O$ (lowercase subscripts refer to genotypes, uppercase to phenotypes). If we knew all this, then we could calculate p_a , p_b , and p_o from

$$\begin{aligned} p_a &= \frac{2N_{aa} + N_{ao} + N_{ab}}{2N} \\ p_b &= \frac{2N_{bb} + N_{bo} + N_{ab}}{2N} \\ p_o &= \frac{2N_{oo} + N_{ao} + N_{bo}}{2N} , \end{aligned}$$

where N is the total sample size.

Surprisingly enough we can actually estimate the allele frequencies by using this trick. Just take a guess at the allele frequencies. Any guess will do. Then calculate N_{aa} , N_{ao} , N_{bb} , N_{bo} , N_{ab} , and N_{oo} as described in the preceding paragraph.¹ That's the **E**xpectation part of the EM algorithm. Now take the values for N_{aa} , N_{ao} , N_{bb} , N_{bo} , N_{ab} , and N_{oo} that you've calculated and use them to calculate new values for the allele frequencies. That's the **M**aximization part of the EM algorithm. Chances are your new values for p_a , p_b , and p_o won't match your initial guesses, but² if you take these new values and start the process over and repeat the whole sequence several times, eventually the allele frequencies you get out at the end match those you started with. These are maximum-likelihood estimates of the allele frequencies.

USING WINBUGS

An alternative approach is to use a package called WinBUGS. That's short for "**W**indows **B**ayesian analysis **U**sing **G**ibbs **S**ampling." That's also a mouthful, but we won't worry about a lot of what that means (for now at least). Since picking priors for Bayesian analyses is a bit of an art, I'll always tell you what priors to pick for your problem. Your task is then reduced to figuring out how to describe the likelihood of your data.

In this case your sample of phenotypes is a multinomial sample from a set of unknown phenotype frequencies. It's actually pretty simple to describe this in WinBUGS, namely

```
x[1:4] ~ dmulti(pi[],n);
```

`x[1:4]` refers to your data.³ and `pi[]` refers to the unknown phenotype frequencies. The "`~`" means "is drawn from." So you can read the line above as "My phenotype data, `x`, is drawn from a multinomial distribution with unknown phenotype frequencies, `pi`, and the size of my sample is `n`."

Of course, what we're really interested in are the allele frequencies, not the phenotype frequencies, but we know how they're related. In WinBUGS we can describe the relationships as follows

```
pi[1] <- a*a + 2*a*o;      # The frequency of A
pi[2] <- b*b + 2*b*o;      # The frequency of B
pi[3] <- o*o;              # The frequency of O
pi[4] <- 2*a*b;           # The frequency of AB
```

Here you can read the "`<-`" as "is equal to." The value on the right side is assigned to the variable on the left side.⁴

The rest of what you need to know I'll tell you, namely the priors. When you put it all together, this is what the WinBUGS code looks like:

¹ Chances are N_{aa} , N_{ao} , N_{bb} , and N_{bo} won't be integers. That's OK. Pretend that there really are fractional animals or plants in your sample and proceed.

² Yes, truth *is* sometimes stranger than fiction.

³ More about this later.

⁴ Be careful. You're actually learning a little bit of computer programming here.

```

model ABO;
{
  # priors
  a1 ~ dgamma(1,0.01);
  b1 ~ dgamma(1,0.01);
  o1 ~ dgamma(1,0.01);
  a <- a1/(a1 + b1 + o1);
  b <- b1/(a1 + b1 + o1);
  o <- o1/(a1 + b1 + o1);

  # likelihood
  pi[1] <- a*a + 2*a*o;
  pi[2] <- b*b + 2*b*o;
  pi[3] <- o*o;
  pi[4] <- 2*a*b;
  x[1:4] ~ dmulti(pi[],n);
}

```

So what do we do now that we have this? Fire up WinBUGS. Open a new file and enter the code as shown above. Then enter the data on a line below the :

```
list(x=c(862, 365, 702, 131), n=2060)
```

Here $N_A = 862$, $N_B = 365$, $N_O = 702$, $N_{AB} = 131$, and $n = 2060$ is the total sample size (but you probably figured that out already). Once you're satisfied that you haven't made any typing errors, you're ready to proceed.

Double-click on the word "model." Then go to the menu at the top of your screen and select "Model->Specification." Then push the button that says "check model." If you've done everything correctly, you should see a small message in the status line at the bottom of WinBUGS that says "model is syntactically correct." Congratulations!

Now double-click on the word "list." Then push the button labeled select "load data." You should see "data loaded" in the status line. Now push "compile," and you should see "model compiled" in the status line.⁵ Push "gen inits," and you should see "initial values generated."

Select "Model->Update." You can change the number of updates if you'd like, but the default (1000) should be just fine. Click update and wait until the box labeled "iteration" reads "1000" (or whatever you selected for the number of updates).⁶

Select "Inference->Samples" and the "Sample Monitor Tool" will pop up. This is where you tell WinBUGS which variables you're interested in. We're interested in **a**, **b**, and **o**, so first enter **a** in the box that says "node," and click "set." Repeat for **b** and **o**. Find the "Update Tool" (you may have to move the sample monitor tool to find it), and click on "update" again. When it's finished running (the iteration box will read 2000), select "Options->Use Log" to turn on a log file. Go back to the Sample Monitor Tool, select **a** and press **stats**, and do the same for **b** and **o**. The result will be a window that looks something like this:

⁵ Hang on. You're almost there.

⁶ You might see a message like "can't bracket slice o1" in the status line. If so, don't worry about it. Push update again. You should get to 1000 before long.

The screenshot shows a window titled "Log" with a scroll bar on the right. It contains three sections of node statistics:

Node statistics								
node	mean	sd	MC error	2.5%	median	97.5%	start	sample
a	0.2815	0.007594	2.16E-4	0.2675	0.2813	0.2966	1001	1000
Node statistics								
node	mean	sd	MC error	2.5%	median	97.5%	start	sample
b	0.1289	0.00524	1.693E-4	0.1184	0.1292	0.1389	1001	1000
Node statistics								
node	mean	sd	MC error	2.5%	median	97.5%	start	sample
o	0.5896	0.008433	2.333E-4	0.5727	0.5899	0.6057	1001	1000

The columns in which we're interested are the ones labeled "mean," "sd," "2.5%," "median," and "97.5%".⁷ The mean and median are exactly what you probably think they are. The sd is the standard deviation. The 2.5% and 97.5% correspond are the "95% credibility intervals."

HOMEWORK PROBLEMS

1. Try the EM algorithm on the ABO data above, and compare your results with those presented from WinBUGS.
2. Try WinBUGS on the following data set:

Phenotypes	S	M	F	O	SM	MF	SF
Genotypes	ss so	mm mo	ff fo	oo	sm	mf	sf
No. in sample	1149	36	17	20	336	25	203

Use the following priors in WinBUGS:

```

s1 ~ dgamma(1,0.01);
m1 ~ dgamma(1,0.01);
f1 ~ dgamma(1,0.01);
o1 ~ dgamma(1,0.01);
s <- s1/(s1 + m1 + f1 +o1);
m <- m1/(s1 + m1 + f1 +o1);
f <- f1/(s1 + m1 + f1 +o1);
o <- o1/(s1 + m1 + f1 +o1);

```

You will probably want to write a small program, if you are familiar with a programming language, or set up a spreadsheet in Excel to do the EM calculations. They'll be pretty tedious otherwise.

⁷ The numbers you see in your output may be slightly different from those illustrated here, but they should be pretty close.