Bayesian Hierarchical Modeling

A conceptual introduction

“P(A|B) = P(B|A) / P(B) * P(A)”

“when you have eliminated the impossible, whatever remains, however improbable, must be the truth”

Let's start with a simple problem:

You have a coin, and you're considering two hypotheses equally. Either:

- The coin is fair. p(H) = 0.5
- Or the coin will always land tails. p(H) = 0.0

We'll flip the coin once, observe the outcome, and evaluate the hypotheses.
Here's the table of all the equiprobable possibilities under consideration:
(rows 2 and 3 are duplicates of each other, because they must be equiprobable to rows 0 and 1)

	p(H)	possible result
0	0.5	H
1	0.5	T
2	0.0	T
3	0.0	T

You flip the coin and it lands heads (H). Which hypothesis is correct?

Let's start with a simple problem:

Obviously p(H) must have been 0.5. But let's make this procedure super explicit:

	p(H)
0	0.5
1	0.0

1. Consider the hypotheses.

	p(H)	possible result
0	0.5	H
1	0.5	T
2	0.0	T
3	0.0	T

2. Generate possible outcomes.

	p(H)	possible result
0	0.5	H
~~1~~	~~0.5~~	~~T~~
~~2~~	~~0.0~~	~~T~~
~~3~~	~~0.0~~	~~T~~

3. Eliminate the impossible.

4. Whatever remains must be the truth!

Let's start with a simple problem:

Note that this procedure is generalizable, and it completely obey's Bayes' rule.

As a test, try the problem again, except this time say the result turns out to be tails (T). What conclusions can we draw? What is the probability that p(H) = 0.5?

	p(H)
0	0.5
1	0.0

1. Consider the hypotheses.

	p(H)	possible result
0	0.5	H
1	0.5	T
2	0.0	T
3	0.0	T

2. Generate possible outcomes.

	p(H)	possible result
~~0~~	~~0.5~~	~~H~~
1	0.5	T
2	0.0	T
3	0.0	T

3. Eliminate the impossible.

4. Whatever remains must be the truth!

The posterior, or the "truth", can be read straight off from the remaining table. The probability of a "fair coin" is 1/3, and this can be double checked through Bayes' rule.

Let’s try a “standard” CTR problem.

You have an ad button, whose clickthrough rate you want to measure. Your initial assumption is that the CTR will be somewhere around 2 percent, so you model it as a Beta(2, 98) distribution.

We’ll then observe 100 impressions, see how many of those clicked the button (n_click), and draw conclusions about the actual CTR.

Say that 3 of the 100 impressions resulted in a click. Then...

Let’s try a “standard” CTR problem.

… here’s how the procedure works in this case: (recall: Beta(2,98) as the prior, 3/100 clicks as the result)

	alpha	beta
0	2	98

1. Consider the hypotheses.

sample from Beta(alpha, beta)

	alpha	beta	ctr
0	2	98	0.011704
1	2	98	0.026149
2	2	98	0.015570
3	2	98	0.013632
4	2	98	0.014818

2. Generate possible outcomes. First, the CTR...

sample from Binom(100, ctr)

	alpha	beta	ctr	n_click
0	2	98	0.011704	1
1	2	98	0.026149	3
2	2	98	0.015570	3
3	2	98	0.013632	2
4	2	98	0.014818	4

… then the number of clicks.

3. Eliminate the impossible.

	alpha	beta	ctr	n_click
~~0~~	~~2~~	~~98~~	~~0.011704~~	~~1~~
1	2	98	0.026149	3
2	2	98	0.015570	3
~~3~~	~~2~~	~~98~~	~~0.013632~~	~~2~~
~~4~~	~~2~~	~~98~~	~~0.014818~~	~~4~~

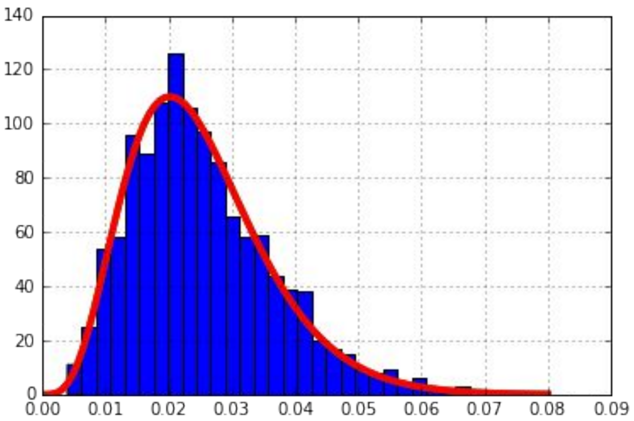
4. Whatever remains must be the truth!

You can draw as many samples as is practical, as long as they remain equiprobable. The more the better.

Let's try a "standard" CTR problem.

Once you have the final remaining "truth" table, you can again just read off the answers from it. For example, the "ctr" column in this table IS the posterior distribution of the CTR, subject only to having enough samples.

According to the usual conjugacy rules of Beta distributions, this should be Beta(2 + 3, 98 + 97) = Beta(5,195) - and indeed it is.



Blue histogram: from the ctr column
Red line: Beta(5, 195)

	alpha	beta	ctr	n_click
0	2	98	0.011704	1
1	2	98	0.026149	3
2	2	98	0.015570	3
3	2	98	0.013632	2
4	2	98	0.014818	4

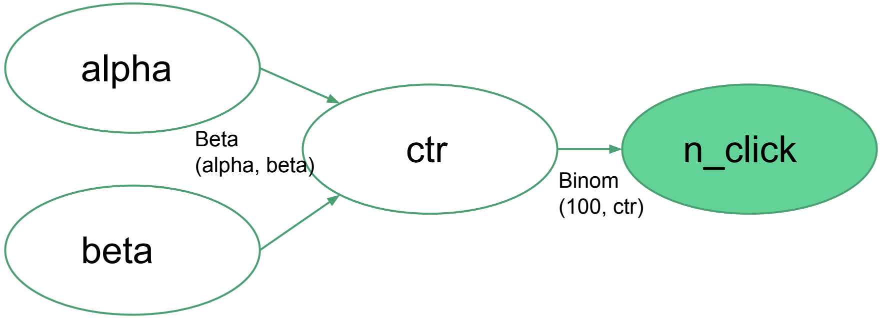
4. Whatever remains must be the truth!

You now understand hierarchical models!

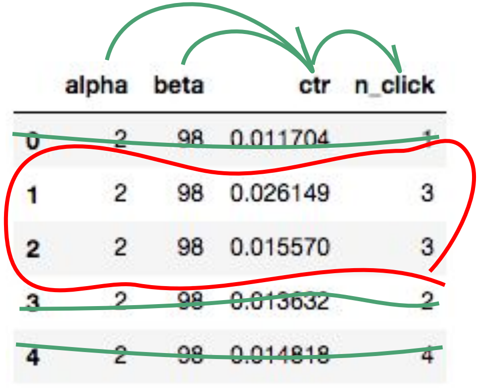
… wait, what?

Yes. Really. Anything with more than 2 columns in the final table can be considered a hierarchical model.

A hierarchical model is any model where the parameters and the data are generated in a stepwise fashion, so that one (or more) set of numbers is used to generate the next set of numbers.



Hierarchical model



	alpha	beta	ctr	n_click
0	2	98	0.011704	1
1	2	98	0.026149	3
2	2	98	0.015570	3
3	2	98	0.013632	2
4	2	98	0.014818	4

Our procedure

Hierarchical model for multiple comparisons

We'll do this problem in exactly the same way as the previous problems.

We have an ad button, whose clickthrough rate we want to measure. We're going to try out 3 different button colors - red, green, blue - to see which ones have the highest CTR.

Without multiple comparisons, we'd proceed by assuming that the prior is something like Beta(1,1), then adding in the observed clicks and impressions for each button color.

But here, we're going to assume that the prior will be of the form Beta(alpha, beta), and that this prior will serve as the prior for all three button types.

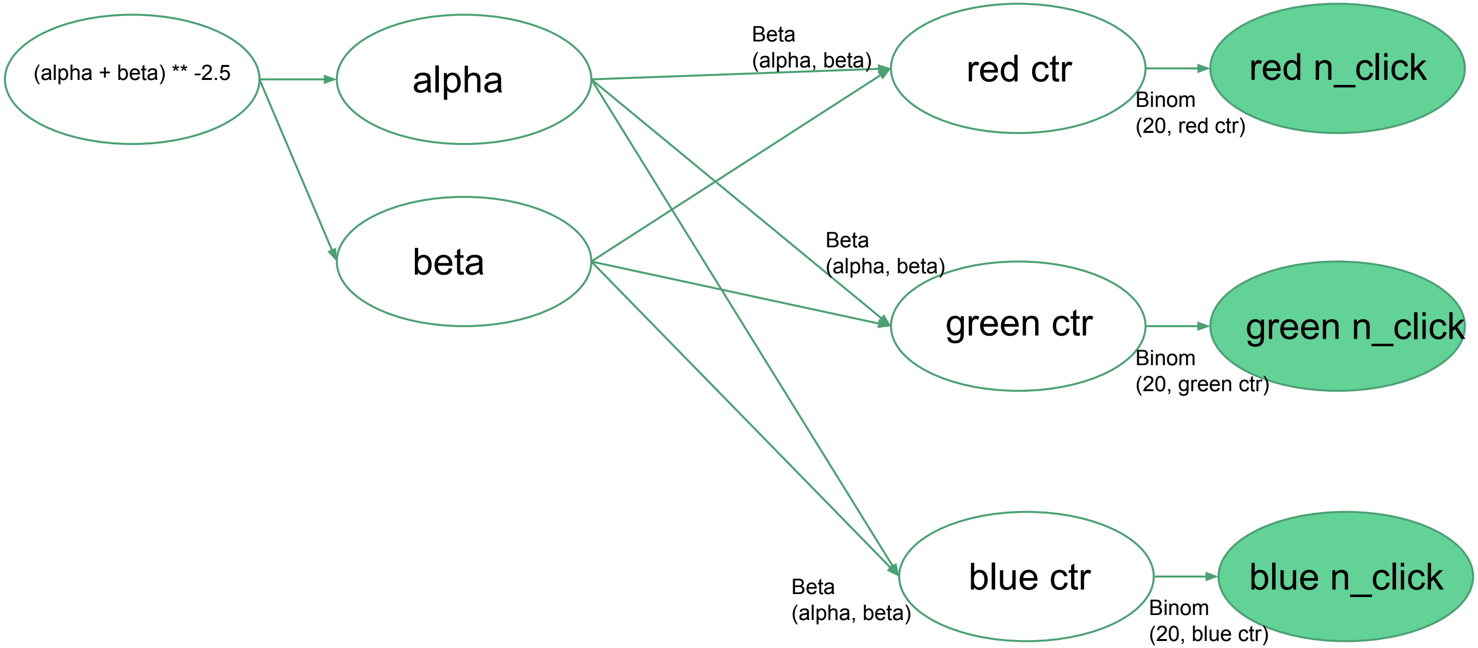
We don't know what alpha and beta are. To quantify our ignorance, we'll assume that they could be any numbers drawn from the uninformative prior of (alpha + beta) ** -2.5.

The observed outcomes are as follows:

- Red: 8 clicks /20 impressions
- Green: 3 clicks /20 impressions
- Blue: 4 clicks /20 impressions

Hierarchical model for multiple comparisons

Here's the hierarchical model:



Hierarchical model for multiple comparisons

And here's our "eliminate the impossible, whatever remains must be true" procedure.

(alpha + beta) ** -2.5

	a	b	red_ctr	green_ctr	blue_ctr	red_n_click	green_n_click	blue_n_click
0	27.560408	93.585110	0.158532	0.270765	0.170768	2	5	2
1	29.150653	94.174098	0.248105	0.315990	0.253719	3	5	4
2	29.150653	94.174098	0.216742	0.256373	0.232937	2	6	4
3	29.004853	91.192713	0.238933	0.232725	0.253625	8	4	7
4	31.208061	96.351843	0.375837	0.262979	0.157543	7	5	2

Filter to "eliminate the impossible", keeping only the rows that agree with observed outcomes:

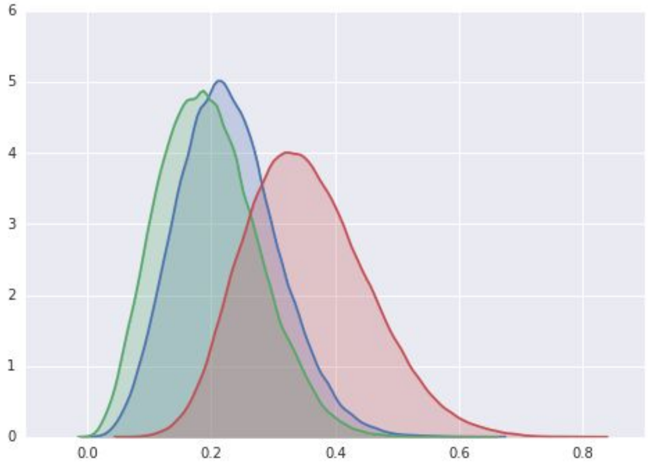
Red: 8 clicks /20
Green: 3 clicks /20
Blue: 4 clicks /20

	a	b	red_ctr	green_ctr	blue_ctr	red_n_click	green_n_click	blue_n_click
222	5.888692	16.568410	0.321374	0.240870	0.248961	8	3	4
2116	2.076052	7.913956	0.288322	0.204576	0.194890	8	3	4
5627	6.765367	11.730478	0.482734	0.270685	0.209131	8	3	4
9288	0.871883	3.667231	0.492877	0.180781	0.146215	8	3	4
10641	1.859267	7.828221	0.390246	0.143538	0.296354	8	3	4

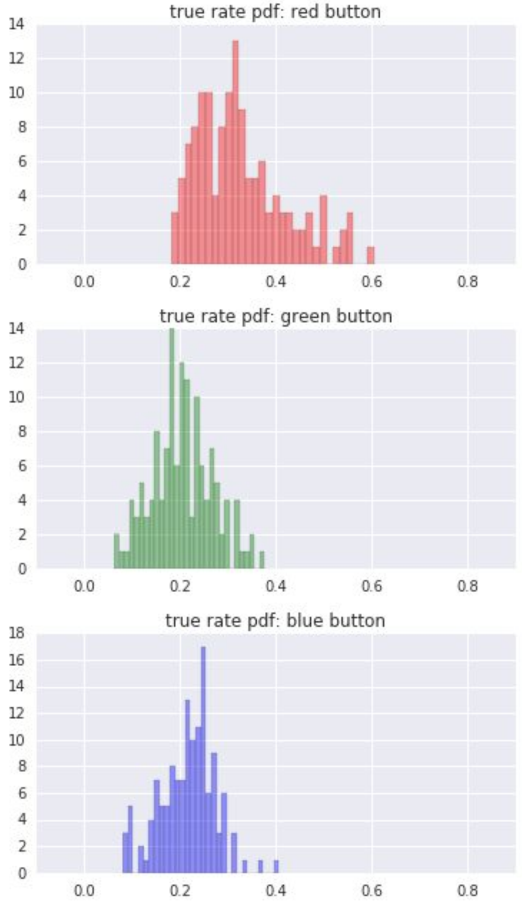
Hierarchical model for multiple comparisons: results

Results. Recall that the raw data was as follows:

- Red: 8 clicks /20 impressions
- Green: 3 clicks /20 impressions
- Blue: 4 clicks /20 impressions



True rate pdfs, using pymc. Note that each pdf is shifted towards their common mean, more so than the raw data would suggest. For example, for red, 8/20 = 0.4, but the pdf is centered more around 0.35 or so.



True rate pdf histograms, from brute-force simulation with pandas. Fewer samples, but note the good agreement with the pymc result.

Miscellaneous notes and “gotchas” I fell into

I kept wanting the prior distribution for all 3 button colors to be a single Beta distribution. But it’s not. It’s a combination of any Beta(alpha, beta) distribution which manages to generate the observed outcomes.

Similarly, I kept wanting to apply Bayes’ rule at each stage in the hierarchy. That is, I wanted to “fit”, or “eliminate the impossible”, multiple times, once at each stage. But you don’t. You “fit” or “eliminate the impossible” only once, to cut away all the rows which do not correspond to the observed outcomes after all of the random numbers have been generated.

The reason for these misconceptions was that I wanted to keep the Beta-Beta conjugacy of our pdfs. But you can’t. You generally have to give up on any clean, closed-form solutions.

That’s why everything is simulated, and that’s why using hierarchical models takes a long time.

Suggested resources:

http://sl8r000.github.io/ab_testing_statistics/use_a_hierarchical_model/
https://pymc-devs.github.io/pymc/tutorial.html

Conclusion: Holmes and Bayes, unified!

Questions? Comments? Suggestions?

Email me at naclhv at gmail dot com