NARRATOR: The following content is provided under a Creative Commons license. Your support will help MIT OpenCourseWare continue to offer high-quality educational resources for free. To make a donation, or view additional materials from hundreds of MIT courses, visit MIT OpenCourseWare at ocw.mit.edu.

PROFESSOR: All right, we should get started. So it's good to be back. We'll be discussing DNA sequence motifs. Oh yeah, we were, if you're wondering, yes, the instructors were at the awards on Sunday. It was great. The pizza was delicious.

So today, we're going to be talking about DNA and protein sequence motifs, which are essentially the building blocks of regulatory information, in a sense. Before we get started, I wanted to just see if there are any questions about material that Professor Gifford covered from the past couple days? No guarantees I'll be able to answer them, but just general things related to transcriptome analysis, or PCA? Anything?

Hopefully, you all got the email that he sent out about, basically, what you're expected to get. So at the level of the document that's posted, that's sort of what we're expecting. So if you haven't had linear algebra, that should still be accessible-- not necessarily all the derivations. Any questions about that?

OK, so as a reminder, team projects, your aims are due soon. We'll post a slightly-- there's been a request for more detailed information on what we'd like in the aims, so we'll post something more detailed on the website this evening, and probably extend the deadline a day or two, just to give you a little bit more time on the aims. So after you submit your aims-- this is students who are taking the project component of the course-- then your team will be assigned to one of the three instructors as a mentor/advisor, and we will schedule a time to meet with you in the next week or two to discuss your aims, just to assess the feasibility of the project and so forth, before you launch into it.

All right-- any questions from past lectures? All right, today we're going to talk about

modeling and discovery of sequence motifs. We'll give an example of a particular algorithm that's used in motif finding called the Gibbs Sampling Algorithm.

It's not the only algorithm, it's not even necessarily the best algorithm. It's pretty good. It works in many cases. It's an early algorithm. But it's interesting to talk about because it illustrates the problem in general, and also it's an example of a stochastic algorithm-- an algorithm where what it does is determined at random, to some extent. And yet still often converges to a particular answer. So it's interesting from that point of view.

And we'll talk about a few other types of motif finding algorithms. And we'll do a little bit on statistical entropy and information content, which is a handy way of describing motifs. And talk a little bit about parameter estimation, as well, which is critical when you have a motif and you want to build a model of it to then discover additional instances of that motif.

So some reading for today-- I posted some nature biotechnology primers on motifs and motif discovery, which are pretty easy reading. The textbook, chapter 6, also has some good information on motifs, I encourage you to look at that. And I've also posted the original paper by Bailey and Elkin on the MEME algorithm, which is kind of related to the Gibbs Sampling Algorithm, but is used as expectation maximization. And so it's a really nice paper-- take a look at that.

And I'll also post the original Gibbs Sampler paper later today. And then on Tuesday, we're going to be talking about Markov and hidden Markov models. And so take a look at the primer on HMMs, as well as there is some information on HMMs in the text. It's not really a distinct section, it's kind of scattered throughout the text. So the best approach is to look in the index for HMMs, and read the relevant parts that you're interested in.

And if you really want to understand the mechanics of HMMs, and how to actually implement one in depth, then I strongly recommend this Rabiner tutorial on HMMs, which is posted. So everyone please, please read that. I will use the same notation, to the extent possible, as the Rabiner paper when talking about some of the

algorithms used in HMMs in lecture. So it should synergize well.

So what is a sequence motifs? In general, it's a pattern that's common to a set of DNA, RNA, or protein sequences, that share a biological property. So for example, all of the binding sites of the Myc transcription factor-- there's probably a pattern that they share, and you call that the motif for Myc.

Can you give some examples of where you might get DNA motifs? Or protein motifs? Anyone have another example of a type of motif that would be interesting? What about one that's defined on function? Yeah, go ahead. What's your name?

AUDIENCE: Dan. [INAUDIBLE]

PROFESSOR: Yeah. So each kinase typically has a certain sequence motif that determines which proteins it phosphorylate. Right. Other examples? Yeah, so in that case, you might determine it functionally. You might purify that protein, incubate it with a pool of peptides, and see what gets phosphorylated, for example. Yeah, in the back?

AUDIENCE: I'm [INAUDIBLE], and promonocytes.

PROFESSOR: What was the first one?

AUDIENCE: Promonocytes? Oh, that one? Oh, that was my name.

PROFESSOR: Yeah, OK. And as to promoter motifs, sir? Some examples?

AUDIENCE: Like, [INAUDIBLE] in transcription mining site.

PROFESSOR: Ah. Yeah. And so you would identify those how?

AUDIENCE: By looking at sequences upstream of [INAUDIBLE], and seeing what different sequences have in common?

PROFESSOR: Right. So I think there's at least three ways-- OK, four ways I can think of identifying those types of motifs. That's probably one of the most common types of motifs encountered in molecular biology. So one way, you take a bunch of genes, where you've identified the transcription start site. You just look for patterns-- short sub-

sequences that they have in common. That might give you the TATA box, for example.

Another way would be, what about comparative genomics? You take each individual one, look to see which parts of that promoter are conserved. That can also help you refine your motifs. Protein binding, you could do ChIP-Seq, that could give you motifs.

And what about a functional readout? You clone a bunch of random sequences upstream of a luciferase reporter, see which ones actually drive expression, for example. So, that would be another. Yeah, absolutely, so there's a bunch of different ways to define them.

In terms of when we talk about motifs, there are several different models of increasing resolution that people use. So people often talk about talk about the consensus sequence so you say the TATA box, which, of course, describes the actual motif-- T-A-T-A-A-A, something like that. But that's really just the consensus of a bunch of TATA box motifs. You rarely find the perfect consensus in real promoters-- the real, naturally occurring ones are usually one or two mismatches away. So that doesn't fully captured it.

So sometimes you'll have a regular expression. So an example would be if you were describing mammalian 5 prime splice sites, you might describe the motif as GT, A or G, AGT, or sometimes abbreviated as GTR AGT, where R is shorthand for either appearing nucleotide-- either A or G. In some motifs you could have GT, NN, GT, or something like that. Those can be captured, often, by regular expressions in a scripting language like Python or Perl.

Another very common description in motifs, there would be a weight matrix. So you'll see a matrix where the width of the matrix is the number of bases in the motif. And then there are four rows, which are the four bases-- we'll see that in a moment. Sometimes these are described as position-specific probability matrices, or position-specific score matrices. We'll come to that in a moment. And then there are more complicated models. So it's increasingly becoming clear that the simple weight

matrix is too limited-- it doesn't capture all the information that's present in motifs.

So we talked about where do motifs come from. These are just some examples. I think I talked about all of these, except for in vitro binding. So in addition to doing a CLIP-seq, where you're looking at the binding of the endogenous protein, you could also make recombinant protein-- incubate that with a random pool of DNA molecules, pull down, and see what binds to it, for example.

So why are they important? They're important for obvious reasons-- that they can identify proteins that have a specific biological property of interest. For example, being phosphorylated by a particular kinase. Or promoters that have a particular property. That is, that they're likely to be regulated by a particular transcription factor, et cetera.

And ultimately, if you're very interested in the regulation of a particular gene, knowing what motifs are upstream and how strong the evidence is for each particular transcription factor that might or might not bind there, can be very useful in understanding the regulation of that gene. And they're also going to be important for efforts to model gene expression.

So, a goal of systems biology would be to predict, from a given starting point, if we introduce some perturbation-- for example, if we knock out or knock down a particular transcription factor, or over-express it, how will the system behave? So you'd really want to be able to predict how the occupancy of that transcription factor would change.

You'd want to know, first, where it is at an endogenous levels, and then how its occupancy at every promoter will change when you perturb its levels. And then, what effects that will have on expression of downstream genes. So these sorts of models all require really accurate descriptions of motifs.

OK, so these are some examples of protein motifs. Anyone recognize this one? What motif is that? So it says X's. X's would be degenerate oppositions, and C's would cysteines. And H's [INAUDIBLE]. What is this? What does this define? What

protein has this? What can you predict about its function?

AUDIENCE: Zinc finger.

PROFESSOR: Zinc finger, right. So it's a motif commonly seen in genome binding transcription factors, and it coordinates to zinc. What about this one? Any guesses on what this motif is? This quite a short motif. Yeah?

AUDIENCE: That's a phosphorylation.

PROFESSOR: Phosphorylation site. Yeah. And how do you know that?

AUDIENCE: The [INAUDIBLE] and the [INAUDIBLE] next to it means it's [INAUDIBLE].

PROFESSOR: OK, so you even know what kinase it is, yeah. Exactly. So that's sort of the view. So, serine, threonine, and tirocene are the residues that get phosphorylated. And so if you see a motif with a serine in the middle, it's a good chance it's a phosphorylation site.

Here are some-- you can think of them as DNA sequence motifs, because they occur in genes, but they, of course, function at that RNA level. These are the motifs that occur at the boundaries of mammalian introns. So this first one is the prime splicing motif.

So these would be the bases that occur at the last three bases of the exon. The first two of the intron here, are almost always GT. And then you have this position that I mentioned here-- it's almost always A or G position.

And then some positions that are bias for A, bias for G, and then slightly biased for T. And that is what you see when you look at a whole bunch of five prime ends of mammalian introns-- they have this motif. So some will have better matches, or worse, to this particular pattern. And that's the average pattern that you see.

And it turns out that in this case, the recognition of that site is not by a protein, per se, but it's by a ribonucleoprotein complex. So there's actually an RNA called U1 snRNA that base pairs with the five prime splice site. And its sequence, or part of its

sequence, is perfectly complimentary to the consensus five prime splice site. So we can understand why five prime splice sites have this motif-- they're evolving to have a certain degree of complementarity to U1, and in order to get officially recognized by the splicing machinery.

Then at the three prime end of introns, you see this motif here. So here's the last base of the intron, a G, and then an A before it. Almost all introns end with AG. Then you have a pyrimidine ahead of it. Then you have basically an irrelevant position here at minus four, which is not strongly conserved. And then a stretch of residues that are usually, but not always, pyrimidines-- called the pyrimidine track.

And in this case, the recognition is actually by proteins rather than RNA. And there are two proteins. One called U2AF65 that binds the pyrimidine track, and one, U2AF35 that binds that last YAG motif. And then there's an upstream motif here, that's just upstream of the 3 prime splice site that is quite degenerate and hard to find, called the branch point motif.

OK, so, let's take an example. So the five prime splice site is a nice example of a motif, because you can uniquely align them, right? You can sequence DNA, sequence genomes, align the CDNA to the genome, that tells you exactly where the splice junctions are. And you can take the exons that have a 5 prime splice site, and align the sequences aligned to the exon/intron boundary and get a precise motif. And then you can tally up the frequencies of the bases, and make a table like this, which we would call a position-specific probability matrix.

And what you could then do to predict additional, say, five prime splice-site motifs in other genes-- for example, genes where you didn't get a good CDNA coverage, because let's say they're not expressed in the cells that you analyzed-- you could then make this odds ratio here. So here we have a candidate sequence.

So the motif is nine positions, often numbered minus 3 to minus 1, would be the exonic parts of this. And then plus 1 to plus 6 would be the first six bases of the intron. That's just the convention that's used. I'm sure it's going to drive the computer scientists crazy because we're not starting at 0, but that's usually what's

used in the literature.

And so we have a nine-based motif. And then we're going to calculate the probability of generating that particular sequence as given plus-- meaning given our foreground, or motif model-- as the product of the probability of generating the first base in sequence, S1, using the column probability in the minus 3 position.

So if the first base is AC, for example, that would be 0.4. And then the probability of generating the second base in the sequence using the next column, and so forth. If you made a vector for each position that had a 1 for the base that occurred at that position, and a 0 for the other bases, and then you just did the dot product of that with the matrix, you get this.

So, we multiply probabilities. So that is assuming independence between positions. And so that's a key assumption-- weight matrices assume that each position in the motif contributes independently to the overall strength of that motif. And that may or may not be true-- they don't assume that it's homogeneous, that is you have usually in a typical case, different probabilities in different columns, so it's inhomogeneous, but assumes independence.

And then you often want to use a background model. For example, if your genome composition is 25% of each of the nucleotides, you could just have a background probability that was equally likely for each of the four, and then calculate the probability, S given minus, of generating that particular [INAUDIBLE] under the background model, and take the ratio of those two.

And the advantage of that is that then you can find sequences that are-- that ratio, it could be 100 times more like a 5 prime splice site than like background-- or 1,000 times. Or you have some sort of scaling on it. Whereas, if you just take the raw probability, it's going to be something that's on the order of 1/4 to a 1/9. So some very, very small number that's a little hard to work with.

So when people talk about motifs, they often use language like exact, or precise, versus degenerate, strong versus weak, good versus lousy, depending on the

context, who's listening. So an example of these would be a restriction enzyme. You often say restriction enzymes have very precise sequence specificity, they only cut-- echo R1 only cuts a GAA TTC. Whereas, a TATA binding protein is somewhat more degenerate. It'll bind to a range of things. So I use degenerate there, you could say it's a weaker motif.

You'll often-- if you want to try to make this precise, then the language of entropy information offers additional terminology, like high information content, low entropy, et cetera. So let's take a look at this as perhaps a more natural, or more precise way of describing what we mean, here.

So imagine you have a motif. We're going to do a motif of length one-- just keep the math super simple, but you'll see it easily generalizes. So you have probabilities of the four nucleotides that are Pk. And you have background probabilities, qk. And we're going to assume those are all uniform, they're all a quarter.

So then the statistical, or Shannon entropy of a probability distribution-- or vector of probabilities, if you will-- is defined here. So H of q, where q is a distribution or, in this case, vector, is defined as minus the summation of qk log qk, in general. And then if you wanted to be in units of bits, you'd use log base 2.

So how many people have seen this equation before? Like half, I'm going to go with. OK, good. So who can tell me why, first of all-- is this a positive quantity, negative quantity, non-negative, or what? Yeah, go ahead.

AUDIENCE: Log qk is always going to be negative. And so therefore you have to take the negative of the sum of all the negatives to get a positive number.

PROFESSOR: Right, so this, in general, is a non-negative quantity, because we have this minus sign here. We're taking logs of things that are between 0 and 1. So the logs are negative, right? OK. And then what would be the entropy if I say that the distribution q is this-- 0100, meaning, it's a motif that's 100% C? What is the entropy of that? What was your name?

AUDIENCE: William.

PROFESSOR: William.

AUDIENCE: So the entropy would be 0, because the vector is determined in respect of the known certainty.

PROFESSOR: Right. And we do the math-- you'll get, for the C term, you'll have a sum. You'll have three terms that are 0 log 0-- it might crash your calculator, I guess. And then you'll have one term that is 1 log 1. And so 1 log 1, that's easy. That's 0, right?

This, you could say, is undefined. But using L'Hospital's rule-- by continuity, x log x, you take the limit, as x gets small, is 0. So this is defined to be 0 in information theory. And this is always, always 0. So that comes out to be 0.

So it's deterministic. So entropy is a measure of uncertainty, and so that makes sense-- if you know what the base is, there's no uncertainty, entropy is 0. So what about this vector-- 1/4, 1/4, 25% of each of the bases, what is H of q? Anyone? I'm going to make you show me why, so-- Anyone want to attempt this? Levi?

AUDIENCE: I think it's 2.

PROFESSOR: 2, OK. Can you explain?

AUDIENCE: Because the log of the 1/4's is going to be negative 2. And then you're multiplying that by 1/4, so you're getting 1/2 for each and adding it up equals 2.

PROFESSOR: Right, in sum, there are going to be four terms that are 1/4 times log of a 1/4. This is minus 2, 1/4 times minus 2 is minus 1/2, 4 times minus 1/2 is minus 2, and then you change the sign, because there's this minus in front. So that equals 2.

And what about this one? Anyone see that one? This is a coin flip, basically. All right? It's either A or G. [INAUDIBLE]. Anyone? Levi, want to do this one again?

AUDIENCE: It's 1.

PROFESSOR: OK, and why?

AUDIENCE: Because you have two terms of 0 log 0, which is 0. And two terms of 1/2 times the log of 1/2, which is just negative 1. So you have 2 halves.

PROFESSOR: Yeah. So two terms like that. And then there's going to be two terms that are something that turns out to be 0-- 0 log 0. And then there's a minus in front. So that will be 1.

So a coin flip has one bit of information. So that's basically what we mean. If you have a fair coin and you don't know the outcome, we're going to call that one bit. And so a base that could be any of the four equally likely has twice as much uncertainty.

All right, and this is related to the Boltzmann entropy that you may be familiar with from statistical mechanics, which is the log of the number of states, in that if you have N states, and they're all equally likely, then it turns out that the Shannon entropy turns out to be log of the number states. We saw that here-- four states, equally likely, comes out to be log of 4 or 2. And that's true in general. All right, so you can think of this as a generalization of Boltzmann entropy, if you want to.

OK, so why did he call it entropy? So it turns out that Shannon, who was developing this in the late '40s, as developing a theory of communication, scratched his head a little bit. And he talked to his friend, John von Neumann-- none other than him, involved in inventing computers-- and he says, "My concern was what to call it. I thought of calling it information. But the word was overly used."

OK, so back in 1949, information was already overused. "And and so I decided to call it uncertainty." And then he discussed it with John von Neumann, and he had a better idea. He said, "You should call it entropy. In the first place, your certainly function has already been used in statistical mechanics under that name," so it already has a name. "And the second place, and more important, nobody knows what entropy really is, so in a debate, you always have the advantage."

So keep that in mind. After you've taken this class, just start throwing it around and you will win a lot of debates. So how is information related to entropy? So the way

we're going to define it here, which is how it's often defined, is information is reduction in uncertainty.

So, if I'm dealing with an unknown DNA sequence, the lambda phage genome, and it has 25% of each base, if you tell me, I'm going to send you two bases, I have no idea. They could be any pair of bases. My uncertainty is 2 bits per base, or 4 bits before you tell me anything. If you then tell me, it's the TA motif, which is always T followed by A, then now my uncertainty is 0, so the amount of information that you just gave me is 4 bits. You reduced my uncertainty from 4 bits to 0.

So we define the information at a particular position as the entropy before-- before meaning the background, the background a sort of your null hypothesis-- minus the entropy after-- so after you've told me that this is an instance of that motif, and it has a particular model. So, in this case, you can see the entropy is going to be entropy before. This is just H of q right here, this term. And then minus this term, which is H of p.

So, if it's uniform, we said H of q is 2 bits per position. And so, so the information content of the motif is just 2 minus the entropy of that motif model. In general, it turns out if the positions in the motif are independent, then the information content of the motif is 2w minus H of motif, where w is it width of the motif.

So for example, the entropy of the motif of-- we said the entropy of this is 2 bits, right? Therefore, the information content is what? If this is our-- let's say this is a P. This is our routine. Are you starting to generate? What is its information content?

AUDIENCE: 0?

PROFESSOR: 0. Why is it 0? Yeah, back row.

AUDIENCE: Because the information content of that is 0, and then the information content of the known hypothesis, so to say, is 0. Sorry, both of them are 2. So 2 minus 2 is 0.

PROFESSOR: The entropy of the background is 2, and the entropy if this is also 2. So 2 minus 2 is 0. And what about this? Let's say this was our motif, it's a motif that's either A or G.

We said the entropy of this is 1 bit, so what is the information content of this motif?

AUDIENCE: 1.

PROFESSOR: 1, and why is it 1?

AUDIENCE: Background is 2, and entropy here is 1.

PROFESSOR: Background is 2, entropy is 1. OK? And what about if I tell you it's the echo R1 restriction enzyme? So it's GAA TTC, a six-base motif precise-- it has to be those bases? What is the information content of that motif? In the back?

AUDIENCE: It's 12.

PROFESSOR: 12-- 12 what?

AUDIENCE: 12 bits.

PROFESSOR: 12 bits, and why is that?

AUDIENCE: Because the background is 2 times 6. So 6 bases, and 2 bits for each. And you have all the bases are determined at the specific [INAUDIBLE] enzyme site. So the entropy of that is 0, since 12 minus 0 is 12.

PROFESSOR: Right, the entropy of that motif is 0. You imagine 4,096 possible six-mers. One of them has probably 1. All the others have 0. You're going to have that big sum. It's going to come out to be 0, OK? Why is this useful at all, or is it?

One of the reasons why it's useful-- sorry, that's on a later slide. Well, just hang with me, and it will be clear why it's useful in a few slides. But for now, we have a description of information content.

So the echo R1 site has 12 bits of information, a completely random position has 0, and a short four-cutter restriction enzyme would have 2 times 4, 8 bits of information, right, and an eight-cutter. So you can see as the restriction enzyme gets longer, more information content.

So let's talk about the motif finding problem, and then we'll return to the usefulness of information content. So can everyone see the motif that's present in all these sequences?

If anyone can't, please let me know. You probably can't. Now, what now? These are the same sequences, but I've aligned them. Can anyone see a motif?

PROFESSOR: GGG GGG.

PROFESSOR: Yeah, I heard some G's. Right. so there's this motif that's over here. It's pretty weak, and pretty degenerate. There's definitely some exceptions, but you can definitely see that a lot of the sequences have at least GGC, possibly an A after that.

Right, so this is the problem we're dealing with. You have a bunch of promoters, and the transcription factor that binds may be fairly degenerate, maybe because it likes to bind cooperatively with several of its buddies, and so it doesn't have to have a very strong instance of the motif present. And so, it can be quite difficult to find. So that's why there's a real bio-informatics challenge.

Motif finding is not done by lining up sequences by hand, and drawing boxes-- although that's how the first motif was found, the TATA box. That's why it's called the TATA box, because someone just drew a box in a sequence alignment. But these days, you need a computer to find-- most motifs require some sort of algorithm to find. Like I said, it's essentially a local multiple alignment problem. You want multiple alignment, but it doesn't have to be global. It just can be local, it can be just over a sub-region.

There are basically at least three different sort of general approaches to the problem of motif finding. One approach is the so-called enumerative, or dictionary, approach. And so in this approach, you say, well, we're looking for a motif of length 6 because this is a leucine zipper transcription factor that we're modeling, and they usually have binding sites around 6, so we're going to guess 6. And we're going to enumerate all the six-mers, there's 4,096 six-mers.

We're going to count up their occurrences in a set of promoters that, for example,

are turned on when you over-express this factor, and look at those frequencies divided by the frequencies of those six-mers in some background set-- either random sequences, or promoters that didn't turn on. Something like that. You have two classes, and you look for statistical enrichment.

This approach, this is fine. There's nothing wrong with this approach. People use it all the time. One of the downsides, though, is that you're doing a lot of statistical tests. You're essentially testing each six-mer-- you're doing 4,096 statistical tests. So you have to adjust the statistical significance for the number of tests that you do, and that can reduce your power. So that's one main drawback.

The other reason is that maybe you don't see-- maybe this protein binds a rather degenerate motif, and a precise six-mer is just too precise. None of them will occur often enough. You really have to have a degenerate motif that's C R Y G Y. That's really the motif that it binds to, and so you don't see it unless you use something more degenerate. So you can generalize this to use regular expressions, et cetera. And it's a reasonable approach.

Another approach that we'll talk about in a moment is probabilistic optimization, where you wander around the possible space of possible motifs until you find one that looks strong. And we'll talk about that.

And then they're deterministic versions of this, like me. We're going to focus today on this second one. Mostly because it's a little bit more mysterious and interesting as an algorithm. And it's also [INAUDIBLE].

So, if the motif landscape looked like this, where imagine all possible motifs, you've somehow come up with a 2D lattice of the possible motif sequences. And then the strength of that motif, or the degree to which that motif description corresponds to the 2 motif is represented by the height here. Then, there's basically one optimal motif, and the closer you get to that, the better fit it is. Then our problem is going to be relatively easy.

But it's also possible that it looks something like this. There's a lot of sort of decoy

motifs, or weaker motifs that are only slightly enriched in the sequence space. And so you can easily get tripped up, if you're wandering around randomly. We don't know a priori, and it's probably not as simple as the first example. And so that's one of the issues that motivates these stochastic algorithms.

So just to sort of put this in context-- the Gibbs motif sampler that we're going to be talking about is a Monte Carlo algorithm, so that just means it's an algorithm that basically does some random sampling somewhere in it, so that the outcome that you get isn't necessarily deterministic. Your run it at different times, and you'll actually get different outputs, which can be a little bit disconcerting and annoying at times. But it turns out to be useful in some cases.

There's also a special case of a Las Vegas algorithm, where it knows when it got be optimal answer. But in general, not. In general, you don't know for sure.

So Gibbs motif simpler is basically a model where you have a likelihood for generating a set of sequences, S. So imagine you have 40 sequences that are bacterial promoters, each of 40 bases long, let's say. That's your S. And so what you want to do, then, is consider a model that there is a particular instance of a motif you're trying to discover, at a particular position in each one of those sequences. Not necessarily the same position, just some position in each sequence.

And we're going to describe the composition of that motif by a weight matrix. OK, one of these matrices that's of width, W, and then has the four rows specifying the frequencies of the four nucleotides at that position. The setup here is that you want to calculate or think about the probability of S comma A, S is the actual sequences, and A is basically a vector that specifies the location of the motif instance in each of those 40 sequences.

You want to calculate that, conditional on capital theta-- which is our weight matrix. So that's going to be, in this case, I think I made a motif of length 8, and it's shown there in red. There's going to be a weight matrix of length 8. And then there's going to be some sort of background frequency vector that might be the background composition in the genome of E.coli DNA, for example.

And so then the probability of generating those sequences together with that particular locations is going to be proportional to this. Basically, use the little theta background vector for all the positions, except the specific positions that are inside the motif, starting at position AK here. And then you use the particular column of the weight matrix for those 8 positions, and then you go back to using the background probabilities. Question, yeah?

AUDIENCE: Is this for finding motifs based on other known motifs? Or is this--

PROFESSOR: No, what we're doing-- I'm sorry, I should've prefaced that. We're doing de novo motif finding. We're going to tell the algorithm-- we're going to give the algorithm some sequences of a given length, or it can even be of variable lengths, and we're going to give it a guess of what the length of the motif is. So we're going to say, we think it's 8. That could come from structural reasons. Or often you really have no idea, so you just guess that you know, a lot of times it's kind of short, so we're going to go with 6 or 8, or you try different lengths. Totally de novo motif finding.

OK, so how does algorithm work? You have N sequences of length, L. You guessed that the motif has width, W. You choose starting positions at random-- OK, so this is a vector, of the starting position in each sequence, we're going to choose completely random positions within the end sequences. They have to be at least W before the end-- so we'll have a whole motif, that's just an accounting thing to make it simpler.

And then you choose one of the sequence at random. Say, the first sequence. You make a weight matrix model of width, W, from the instances in the other sequences. So for example-- actually, I have slides on this, so we'll just do it with the slides, you'll see what this looks like in a moment. And so you have instances here in the sequence, here in this one, here. You take all those, line them up, make a weight matrix out of those, and then you score the positions in sequence 1 for how well they match.

So, let me just do this. These are your motif instances. Again, totally random at the

beginning. Then you build a weight matrix from those by lining them up, and just counting frequencies. Then you pick a sequence at random-- yeah, your weight matrix doesn't include that sequence, typically. And then you take your theta matrix and you slide it along the sequence.

You consider every sub-sequence of length, W-- the one that goes from 1 to W, to one that goes from 2 to W plus 1, et cetera, all the way along the sequence, until you get to the end. And you calculate the probability of that sequence, using that likelihood that I gave you before. So, it's basically the probability generating sequence where you use the background vector for all the positions, except for the particular motif instance that you're considering, and use the motif model for that.

Does that make sense? So, if you happen to have a good looking occurrence of the motif at this position, here, in the sequence, then you would get a higher likelihood. So for example, if the motif was, let's say it's 3 long, and it happened to favor ACG, then if you have a sequence here that has, let's say, it's got TTT, that's going to have a low probability in this motif. It's going to be 0.1 cubed.

And then if you have an occurrence of, say, ACT, that's going to have a higher occurrence. It's going to be 0.7 times 0.7 times 0.1. So, quite a bit higher. So you start, it'll be low for this triplet here-- so I'll put a low value here. TTA is also going to be low. TAC, also low. But ACT, that matches 2 out of 3 to the motif. It's going to be a lot better. And then CT is going to be low again, et cetera.

So you just slide this along and calculate probabilities. And then what you do is you sample from this distribution. These probabilities don't necessarily sum to 1. But you re-normalize them so that they do sum to 1, you just add them up, divide by the sum. Now they sum to 1. And now you sample those sites in that sequence, according to that probability distribution.

Like I said, in this case you might end up sampling-- that's the highest probability site, so you might sample that. But you also might sample one of these other ones. It's unlikely you would sample this one, because that's very low. But you actually sometime sample one that's not so great.

So you sample a starting position in that sequence, and you basically-- wherever you would originally assign in sequence 1, now you move it to that new location. We've just changed the assignment of where we think the motif might be in that sequence. And then you choose another sequence at random from your list. Often you go through the sequences sequentially, and then you make a new weight matrix model.

So how will that weight matrix model differ from the last one? Well it'll differ because the instance of the motif in sequence 1 is now at a new location, in general. I mean, you might have sampled the exact same location you started, but in general it'll move. And so now, you'll got a slightly different weight matrix.

Most of the data going into it, N minus 1, is going to be the same. But one of them is going to be different. So it'll change a little bit.

You make a new weight matrix, and then you pick a new sequence. You slide that weight matrix along that sequence, you get this distribution, you sample from that distribution, and you keep going.

Yeah, this was described by Lorenz in 1993, and I'll post that paper. OK, so you sample a portion with that, and you update the location. So now we sampled that really high probably one, so we moved the motif over to that new orange location, there. I don't know if these animations are helping at all. And then you update your weight matrix.

And then you iterate until convergence. So you typically have a set of end sequences, you go through them once. You have a weight matrix, and then you go through them again. You go through a few times. And maybe at a certain point, you end re-sampling the same sites as you did in the last iteration-- same exact sites. You've converged.

Or, you keep track of the theta matrices that you get after going through the whole set of sequences, and from one iteration to the next, the theta matrix hasn't really changed much. You've converged.

So let's do an example of this. Here I made up a motif, and this is a representation where the four bases have these colors assigned to them. And you can see that this motif is quite strong. It really strongly prefers A at this position here, and et cetera.

And I put it at the same position in all the sequences, just to make life simple. And then a former student in the lab, [INAUDIBLE], he implemented the Gibb Sample in Matlab, actually, and made a little video of what's going on.

So the upper part shows the current weight matrix. Notice it's pretty random-looking at the beginning. And the right parts show where the motif is, or the position that we're currently considering. So this shows the position that was last sampled in the last round. And this shows the probability density along each sequence of what's the probability that the motif occurs at each particular place in the sequence.

And that's what happens over times. So it's obviously very fast, so I'll run it again and maybe pause it partway. We're starting from a very random-looking motif. This is what you get after not too many iterations-- probably like 100 or so. And now you can see your motif-- your weight matrix is now quite biased, and now favors A at this position, and so forth. And the locations of your motif, most of them are around this position, around 6 or 7 in the sequence-- that's where we put the motif in. But not all, some of them.

And then you can see the probabilities-- white is high, black is low-- in some sequences, it's very, very confident, the motif is exactly at that position, like this first sequence here. And others, it's got some uncertainty about where the motif might be. And then we let it run a little bit more, and it eventually converges to being very confident that the motif has the sequence, A C G T A G C A, and that it occurs at that particular position in the sequence.

So who can tell me why this actually works? We're choosing positions at random, updating a weight matrix, why does that actually help you find the real motif that's in these sequences? Any ideas? Or who can make an argument that it shouldn't work? Yeah? What was your name again?

AUDIENCE: Dan.

PROFESSOR: Dan, yeah, go ahead.

AUDIENCE: So, couldn't it, sort of, in a certain situation have different sub-motifs that are also sort of rich, and because you're sampling randomly you might be stuck inside of those boundaries where you're searching your composition?

PROFESSOR: Yeah, that's good. So Dan's point is that you can get stuck in sub-optimal smaller or weaker motifs. So that's certainly true. So you're saying, maybe this example is artificial? Because I had started with totally random sequences, and I put a pretty strong motif in a particular place, so there were no-- it's more like that mountain, that structure where there's just one motif to find. So it's perhaps an easy case.

But still, what I want to know is how does this algorithm, how did it actually find that motif? He implemented exactly that algorithm that I described. Why does it tend to go towards [INAUDIBLE]? After a long time, remember it's a long time, it's hundreds of iterations.

AUDIENCE: So you're covering a lot in the sequence, just the random searching of the sequence, when you're--

PROFESSOR: There are many iterations. You're considering many possible locations within the sequences, that's true. But why does it eventually-- why does it converge to something?

AUDIENCE: I guess, because you're seeing your motif more plainly than you're seeing other random motifs. So it will hit it more frequently-- randomly. And therefore, converge [INAUDIBLE].

PROFESSOR: Yeah, that's true. Can someone give a more intuition behind this? Yeah?

AUDIENCE: I just have a question. Is each iteration an independent test? For example, if you iterate over the same sequence base 100 times, and you're updating your weight matrix each time, does that mean it is the updating the weight matrix also taking into

account that the previous-- that this is the same sample space?

PROFESSOR: Yeah, the weight matrix, after you go through one iteration of all the sequences, you have a weight matrix. You carry that over, you don't start from scratch. You bring that weight matrix back up, and use that to score, let's say, that first sequence. Yeah, the weight matrix just keeps moving around. Moves a little bit every time you sample a sequence.

AUDIENCE: So you constantly get a strong [INAUDIBLE].

PROFESSOR: Well, does it?

AUDIENCE: Well, I guess--

PROFESSOR: Would it constantly get stronger? What's to make it get stronger or weaker? I mean, this is sort of-- you're on the track.

AUDIENCE: If it is random, then there's some probability that you're going to find this motif again, at which point it will get stronger. But, if it's-- given enough iterations, it gets stronger as long as you hit different spots at random.

PROFESSOR: Yeah, yeah. That's what I'm-- I think there was a comment. Jacob, yeah?

AUDIENCE: Well, you can think about it as a random walk through the landscape. Eventually, it has a high probability of taking that motif, and updating the [INAUDIBLE] direction, just from the probability of [INAUDIBLE].

PROFESSOR: OK.

AUDIENCE: And given the [INAUDIBLE].

PROFESSOR: OK, let's say I had 100 sequences of length, I don't know, 30. And the width of the motif is 6. So here's our sequences. We choose random positions for the start position, and let's say it was this example where the real motif, I put it right here, and all the sequences. That's where it starts.

Does this help? So it's 30 and 6, so there's 25 possible start positions. I did that to

make it a little easier.

So what would happen in that first iteration? What w can you say about what the weight matrix would look like? It's going to be a width, W, you know, columns 1, 2, 3, up to 6. We're going to give it 100 positions at random. The motif is here-- let's say it's a very strong motif, that's a 12-bit motif. So it's 100%-- it's echo R1. It's that.

What would that weight matrix look like, in this first iteration, when you first just sample the sites at random? What kind of probabilities would it have?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Equal? OK-- perfectly equal?

AUDIENCE: Roughly.

PROFESSOR: OK. Any box? Are we likely to hit the actual motif, ever, in that first encryption?

AUDIENCE: No, because you have a uniform probability, of sampling. Well, uniform at each one of the 25 positions?

PROFESSOR: Right.

AUDIENCE: Right now, you're not sampling proportional to the likelihood.

PROFESSOR: So the chance of hitting the motif in any given sequence is what?

AUDIENCE: 1/25.

PROFESSOR: 1/25. We have 100 sequences.

AUDIENCE: So that's four out of--

PROFESSOR: So on average, I'll hit the motif four times, right. The other 96 positions will be essentially random, right? So you initially said this was going to be uniform, right? On average, 25% of each base, plus or minus a little bit of sampling error-- could be 23, 24, 26. But now, you pointed out that it's going to be four. You're going to hit the motif four times, on average. So, can you say anything more?

AUDIENCE: Could you maybe have a slightly bias towards G on the first position? Slightly biased towards A on the second and third? Slightly biased towards T on the fourth and fifth. And slightly biased towards C in the sixth? So it would be slightly biased--

PROFESSOR: Right, so remind me of your name?

AUDIENCE: I'm Eric.

PROFESSOR: Eric, OK, so Eric says that because four of the sequences will have a G at the first position, because those are the ones where you sampled the motif, and the other 96 will have each of the four bases equally likely, on average you have like 24%-- plus 4 for G, right? Something like 28%-- this will be 28%, plus or minus a little bit. And these other ones will be whatever that works out to be, 23 or something like that-- 23-ish, on average.

Again, it may not come out exactly like-- G may not be number one, but it's more often going to be number one than any other base. And on average, it'll be more like 28% rather than 25%. And similarly for position two, A will be 28%, and three, and et cetera. And then the sixth will be-- C will have a little bit of a bias.

OK, so even in that first round, when you're sampling that first sequence, the matrix is going to be slightly biased toward the motif-- depending how the sampling went. You might not have hit any instances of motif, right? But often, it'll be a little bit--

Is that enough of a bias to give you a good chance of selecting the motif in that first sequence?

AUDIENCE: You mean in the first iteration?

PROFESSOR: Let's say the first random sequence size sample. No. You're shaking your head. Not enough of a bias because-- it's 0.28 over 0.25 to the sixth power, right? So it's like--

AUDIENCE: The likelihood is still close 1. Like, that's [INAUDIBLE] ratio.

PROFESSOR: So it's something like 1.1 to the sixth, or something like that. So it might be close to

2, might be twice as likely. But still, there's 25 positions. Does that make sense?

So it's quite likely that you won't sample the motif in that first-- you'll sample something else. Which will take it away in some random direction. So who can tell me how this actually ends up working? Why does it actually converge eventually, if you get it long enough?

AUDIENCE: [INAUDIBLE].

PROFESSOR: So the information content, what will happen to that? So the information content, if it was completely random-- we said that would be uniform. That would be zero information content, right? This matrix, which has around 28% at six different positions, will have an information content that's low, but non-zero. It might end up being like 1 bit, or something. And if you then sample motifs that are not the motif, they will tend to reduce the information content, tend to bring it back toward random.

If you sample locations that have the motif, what will that do to the information content? Boost it. So what would you expect if we were to plot the information content over time, what would that look like?

AUDIENCE: It should trend upwards, but it could fluctuate.

PROFESSOR: Yeah.

AUDIENCE: Over the number of iterations?

PROFESSOR: I think I blocked it here. Let me see if I can-- Let's try this. I think I plotted it. OK, never mind. I wanted to keep it very mysterious, so you guys have to figure it out. The answer is that it will-- basically what happens is you start with a weight matrix like this.

A lot of times, because the bias for the motif is quite weak, a lot of times you'll sample-- even for a sequence, what matters is-- like, if you had a sequence where the location, initially, was not the motif, and then you sample another location that's not the motif, that's not really going to change anything. It'll change things a little bit,

but not in any particular direction. What really matters is when you get to a sequence where you already had the motif, if you now sample one that's not the motif, your information content will get weaker. It will become more uniform.

But if you have a sequence where it wasn't the motif, but now you happen to sample the motif, then it'll get stronger. And when it gets stronger, it will then be more likely to pick the motif in the next sequence, and so on.

So basically what happens to the information content is that over many iterations-- it starts near 0. And can occasionally go up a little bit. And then once it exceeds the threshold, it goes like that. So what happens is it stumbles onto a few instances of the motif that bias the weight matrix. And if they don't bias it enough, it'll just fall off that. It's like trying to climb the mountain-- but it's walking in a random direction. So sometimes it will turn around and go back down.

But then when it gets high enough, it'll be obvious. Once you have a, say, 20 times greater likelihood of picking that motif than any other sequence, most of the time you will pick it. And very soon, it'll be stronger. And the next round, when it's stronger, you'll have a greater bias for picking the motif, and so forth. Question?

AUDIENCE: For this specific example, M is much greater than L minus W. How true is that for practical examples?

PROFESSOR: That's a very good question. There is sometimes-- depends on how commonly your motif occurs in the genome, and how good your data is, really, and what the source of your data is. So sometimes it can be very limited, sometimes-- If you do ChIP-Seq you might have 10,000 peaks that you're analyzing, or something. So you could have a huge number.

But on the other hand, if you did some functional assay that's quite laborious for a motif that drives luciferase, or something, and you can only test a few, you might only have 10. So it varies all over the map. So that's a good question. We'll come back to that in a little bit. Simona?

AUDIENCE: If you have a short motif, does it make sense, then, to reduce the number of sequences you have? Because maybe it won't converge?

PROFESSOR: Reduce the number of sequences? What do you people think about that? Is that a good idea or a bad idea? It's true that it might converge faster with a smaller number of sequences, but you also might not find it all. So generally you're losing information, so you want to have more sequences up to a certain point. Let's just do a couple more examples, and I'll come back. Those are both good questions.

OK, so here's this weak motif. So this is the one where you guys couldn't see it when I just put the sequences up. You can only see it when it's aligned-- it's this thing with GGC, here. And here is, again, the Gibbs Sampler. And what happened? Who can summarize what happened here? Yeah, David?

AUDIENCE: It didn't converge.

PROFESSOR: Yeah, it didn't quite converge. The motif is usually on the right side, and it found something that's like the motif. But it's not quite right-- it's got that A, it's G A G C, it should be G G C. And so it sampled some other things, and it got off track a little bit, because probably by chance, there were some things that looked a little bit like the motif, and it was finding some instances of that, and some instances of the real motif. And yeah, it didn't quite converge.

And you can see this probability vectors here, they have multiple white dots in many of the rows. So it doesn't know, it's uncertain. So it keeps bouncing around. So it didn't really converge, it was too weak, it was too challenging for the algorithm.

This is just a summary of the Gibb Sampler, how it works. It's not guaranteed to converge to the same motif every time. So what you generally will want to do is run it several times, and nine out of 10 times, you get the same motif. You should trust that. Go ahead.

AUDIENCE: Over here, are we optimizing for convergence of the value of the information content?

PROFESSOR: No, the information content is just describing-- it's just a handy single number description of how biased the weight matrix is. So it's not actually directly being optimized. But it turns out that this way of sampling tends to increase information content. It's sort of a self-reinforcing kind of a thing. But it's not directly doing that.

However MEME, more or less, directly does that. The problem with that is that, where do you start? Imagine an algorithm like this, but where you deterministically-- instead of sampling from the positions in the sequence, where it might have a motif in proportion to probabilities, you just chose the one that had the highest probability. That's more or less what MEME does. And so what are the pros and cons of that approach, versus this one? Any ideas?

OK, one of the disadvantages is that the initial choice of-- how you're initially seeding your matrix, matters a lot. That slight bias-- it might be that you had a slight bias, and it didn't come out being G was number one. It was actually-- T was number one, just because of the quirks of the sampling. So what would this be, 31 or something? Anyway, it's higher than these other guys.

And so then you're always picking the highest. It'll become a self-fulfilling prophecy. So that's the problem with MEME.

So the way that MEME gets around that, is it uses multiple different seeding, multiple different starting points, and goes to the end with all of them. And then it evaluates, how good a model did we get at the end? And whichever was the best one, it takes that. So it actually takes longer, but you only need to run it once because it's deterministic. You use a deterministic set of starting points, you run a deterministic algorithm, and then you evaluate.

The Gibbs, it can go off on a tangent, but because it's sampling so randomly, it often will fall off, then, and come back to something that's more uniform. And when it's a uniform matrix, it's really sampling completely randomly, exploring the space in an unbiased way. Tim?

AUDIENCE: For genomes that have inherent biases that you know going in, do you precalculate-

- do you just recalculate the weight matrix before, to [? affect those classes? ?] For example, if you had 80% AT content, then you're not looking for-- you know, immediately, that you're going to hit an A or a T off the first iteration. So how do you deal with that?

PROFESSOR: Good question. So these are some features that affect motif finding. I think that we've now hit at least a few of these-- number of sequences, length of sequences, information content, and motif, and basically whether the background is biased or not.

So, in general, higher information content motifs, or lower information content, are easier to find-- who thinks higher? Who thinks lower? Someone, can you explain?

AUDIENCE: I don't know. I just guessed.

PROFESSOR: Just a guess? OK, in back, can you explain? Lower?

AUDIENCE: Low information content is basically very uniform.

PROFESSOR: Low information means nearly uniform-- right, those are very hard to find. That's like that GGC one. The high information content motif, those are the very strong ones, like that first one. Those are much easier to find. Because when you stumble on to them, it biases the matrix more, and you rapidly converge to that. OK, high information is easy to find.

So if I have one motif per sequence, what about the length of the sequence? Is longer or shorter better? Is long better? Who thinks shorter is better? Shorter-- can you explain why short?

AUDIENCE: Shouldn't it be the smaller the search space, the fewer the problems?

PROFESSOR: Exactly, the shorter the search space, and your motif, there's less place for it to hide. You're more likely to sample it. Shorter is better.

If you think about-- if you have a motif like TATA, which is typically 30 bases from the TSS, if you happen to know that, and you give it plus 1 to minus 50, you're

giving it a small region, you can easily find the TATA box. If you give it plus 1 to minus 2,000 or something, you may not find it. It's diluted, essentially.

Number of sequences-- the more the better. This is a little more subtle, as Simona was saying. It affects convergence time, and so forth. But in general, the more the better.

And if you guessed the wrong length of your matrix, that makes it worse than if you guess the right length in either direction. For example, it's six-base motif, you guess three. The information content, even if it's a 12-bit motif, there's only six bits that you could hope to find, because you can only find three of those positions. So clearly, effectively it's a smaller information content, and much harder to find. And vice versa.

Another thing that occurs in practice is what's called shifted motifs. Your motif is G A A T T C. Imagine in your first iteration you happen to hit several of these sequences, starting here. You hit the motif, but off by two at several different places. That'll bias first position to be A, and the second position to be T, and so forth.

And then you tend to find other shifted versions of that motif. You may well converge to this-- A T C C N N, or something like that-- which is not quite right. It's close, you're very close, but not quite right. And it's not as information rich as the real motif. Because it's got those two N's at the end, instead of G A.

So one thing that's done in practice is a lot of times, every so often, the algorithm will say, what would happen if we shifted all of our positions over to the left by one or two? Or to the right by one or two? Would the information content go up? If so, let's do that.

So basically, shifted versions of the motif become local, near-optimal solutions. So you have to avoid them. And biased background composition is very difficult to deal with. So I will just give you one or two more examples of that in a moment, and continue.

So in practice, I would say the Gibbs Sampler is sometimes used, or AlignACE,

which is a version of Gibbs Sampler. But probably more often, people use an algorithm called MEME, which is this EM algorithm, which, like I said, is deterministic, so you always get the same answer, which makes you feel good. May or may not always be right, but you can try it out here at this website.

And actually, the Fraenkel Lab has a very nice website called WebMotifs that runs several different motif finders including, like I said, a MEME and AlignACE, which is similar to Gibbs, as well as some others. And it integrates the output, so that's often a handy thing to use. You can read about them there.

And then I just wanted to say a couple words-- this is related to Tim's comment about the biased background. How do you actually deal with that? And this related to this notion of a mean bit score of a motif.

So if I were to give you a motif model, P, and a background model, q, then the natural scoring system, if you wanted additives scores, instead of multiplicative, you would just take the log. So log P over q, I would argue, is natural additive scores. And that's often what you'll see in a weight matrix-- you'll see log probabilities, or logs of ratios of probabilities. And so then you just add them up, and it makes life a bit simpler.

And so then, if you were to calculate what's the mean bit score-- if I had a bunch of instances of a motif, it will be given by this formula that's here in the upper right. So that's your score. And this is the mean, where you're sampling over the probability in using the motif model, probabilities.

So it turns out, then, that if qk, your background, is uniform, motif of width w-- so its probability of any w-mer, is 1/4 to the w, then it's true that the mean bit-score is 2w minus the entropy of the motif, which is the same as the information content of the motif, using our previous definition. So that's just a handy relationship. And you can do a little algebra to show that, if you want.

So basically summation Pk log Pk over qk-- this log, you turn that into a difference-- so that summation Pk log Pk minus Pk log qk. And then you can do some

rearrangement, and sum them up, and you'll get this formula. I'll leave that as an exercise, and any questions on it, we can do it next time.

So what I wanted to get to is sort of this big question that I posed earlier-- what's the use of knowing the information content of a motif? And the answer is that one use is that it's true, in general, that the motif with n bits of information will occur about once every 2 to the n bases of random sequence. So we said a six-cutter restriction enzyme, echo R1, has an information content of 12 bits.

So by this rule, it should occur about once every to 2 to the 12th bases of sequence. And if you know your powers of 2, which you should all commit to memory, that's about 4,000. 2 to the 12th is 4 to the sixth, is 4,096. So it'll occur about once every 4 [? kb, ?] which if you've ever cut E. coli DNA, you know is about right-- your fragments come out to be about 4 [? kb. ?]

So this turns out to be strictly true for any motif that you can represent by a regular expression, like a precise motif, or something where you have a degenerate R or Y or N in it, still true. And if you have a more general motif that's described by weight matrix, then you have to define a threshold, and it's roughly true, but not exactly.

All right, so what do you do when the background composition is biased, like Tim was saying? What if it's 80%, A plus T? So then, it turns out that this mean bit-score is a good way to go. So like I said, the mean bit-score equals the information content in this special case, where the background is uniform.

But if the background is not uniform, then you can still calculate this mean bit-score, and it'll still be meaningful. But now it's called something else-- it's called relative entropy. Actually it has several names, relative entropy, Kullback-Leibler distance is another, and information for discrimination, depending whether you're reading the Double E literature, or statistics, or whatever.

And so it turns out that if you have a very biased composition-- so here's one that's 75% A T, probability of A and T are 3/8, C and G are 1/8. If your motif is just C 100% of the time, your information content by the original formula that I gave you,

would be 2 bits. However, the relative entropy will be 3 bits, if you just plug in these numbers into this formula, it will turn out to be 3 bits.

My question is, which one better describes the frequency of C in the background sequence? Frequency of this motif-- the motif is just a C. You can see that the relative entropy says that actually, that's stronger than it appears. Because it's a C, and that's a rare nucleotide, it's actually stronger than it appears. And so 2 to the 3rd is a better estimate of its frequency than 2 squared. So relative entropy.

So what you can do when you run a motif finder in a sequence of biased composition, you can say, what's the relative entropy of this motif at the end? And look at the ones that are strong. We'll come back to this a little more next time.

Next time, we'll talk about hidden Markov models, and please take a look at the readings. And please, those who are doing projects, look for more detailed instructions to be posted tonight. Thanks.