The following content is provided by MIT OpenCourseWare under a Creative Commons license. Additional information about our license, and MIT OpenCourseWare in general, is available at ocw.mit.edu.

GEORGE CHURCH: OK. Welcome back to the second half, where we'll talk about multisequence alignment, for starters. And I said that I would show this slide again. This time-- before, it was to introduce how we would go about getting an empirical substitution matrix from distantly related protein sequences, such as distantly related immunoglobulin family members. Now, we would like to ask, how did we get that multisequence alignment?

This is one way of thinking about it as a generalization of the two-dimensional array that we had before, where we would have, say, two sequences, one horizontal, one vertical. Now, the third dimension is the third sequence. This gets harder and harder to visualize as the number of sequences you put in, but let's think about it in three dimensions for just a moment here.

And when you have a multiple alignment, you can think of it as dynamic programming on this hyperlattice and that the indels for any pairwise combination may not be optimal for the triple. And let's go beyond triple, but to a very simple dinucleotide alignment. And we will say that this is the optimal multiple alignment.

You can see here that the multiple examples of AT anchor the A and T as being separate positions, even though normally, if you just did a pairwise alignment with a high gap penalty, there would be a tendency to line the A up with the T. You would not have these canceling indels.

But in the context of the multialignment, you now have a different interpretation. So we want to generalize the kind of algorithms we've been using. And again, this will be a recursive algorithm where the score of a two-character string is defined in terms of the maximum of various shorter strings.

So at the very top is the case where we have no insertions-- the simplest case, where we have no insertions or deletions. And we just ask what is the score of having a [? VSA, ?] that is to say, this triple single-amino-acid comparison, just like what was the score of having a V substituted for an s. Now, we're asking a V substituted for an S substituted for an A.

Now, the number of different cases we have here-- before, it was 3 for a global alignment, which was k, being the number of sequences, was 2. Now k is 3 for a three-way comparison. And all possible subsets is 2 to the k minus 1, in this case, so it's 7.

So seven cases, and you can just walk through them. You can see the first one is no insertions or deletions. The next three are two insertions or deletions in the three different ways that can happen. And then the last three are one placeholder, one of these dashes, which means that the other two sequences have insertions relative to the one dashed. So this is the seven cases for a three-way comparison.

Now, as k grows, then both the space complexity-- the amount of lattice points that you have to store somewhere, either in RAM, or disk, or somewhere-- grows by n to the k-th power where the sequences are roughly n long and the number of sequences is k. Now to compute each of those nodes-- well, I mean, what will be on the order of 2 to the k power, because remember I said that the number of subsets in general is going to be 2 to the k minus 1 or about roughly 2 to the k. And so the time complexity is have to do 2 to the k comparisons per node.

And there are n to the k nodes, so it's of the order 2 the k times n to the k. Now, this is not a straw man. This is not some naive algorithm. This is using all the power that we developed for the pairwise comparison and we're just generalizing it.

And so this is actually a hard problem. This does scale exponentially with k. And it's not like we only want to do k equals 2. There are very good reasons for inferring structure or function without experiments, just from sequence.

And the larger k is, the more you can explore. It's like doing a huge mutagenesis experiment and exploring viable mutants. So we want to do multisequence alignments, so how do we deal with this?

This is the way we deal with most non-polynomial calculations, that is to say, in this case, exponential, which is we approximate. Now, you can get something that's very close to the true optimum if how to prune this hyperlattice. Remember, one of the examples I showed was you could take this band.

If you know where the band should start and how wide it should be, you can essentially prune off many of the nodes without really losing any optimality. But you have to be very sure you know where to start it and how wide it should be. So it's optimal within those constraints.

Then there are others which are more heuristic. They are not guaranteed to be optimal, but on the other hand, they don't necessarily require arbitrary pruning. And the two that we'll illustrate in the next couple of slides is a tree alignment, as illustrated by ClustalW.

By the way, pruning is illustrated by a program called MSA, which is short for multisequence alignment. And we'll show a star alignment. And then when we get, later on into the transcriptome part of the course, we will talk about the Gibbs algorithm. So let's walk through ClustalW, and then a star algorithm.

So here's progressive multiple alignment. And I think most of you, if I had given you the luxury of just thinking this through during the break, how you would do the multialignment, this might be the algorithm you would come up with. Almost always it makes sense to start with the pairwise alignments because that is a solved problem, and we have fairly good scaling for that.

And so here, you take each of the, let's say four sequences and do all pairwise alignments. And you get this 4 by 4 matrix. It's going to be symmetric, so you only have to do the diagonal and the off diagonals on one-half of it and.

And you get the best score is S1 with S3, which has a score of 9. And so you can construct a tree. And this is-- basically, we're starting to describe the method by which we construct a tree, such as that tree of life that I've shown a couple of times now.

And so when you construct a tree, you take the two closest scoring sequences, and you indicate them as terminal branches of the tree. And you connect them to a fork, a branch point. And the distance of each from the common ancestor is indicated by the length of these lines.

And so the second-best score is S2 and S4. It's a little bit weaker similarity than S1 and S3. So you have these longer branches indicating greater divergence. And they're in their own cluster.

Now, it turns out that then the common ancestor for all the sequences, which would be the common ancestor of the common ancestors of the first two clusters, is represented by this final branching closest to the trunk of the tree or the roots of tree. And here, distance is this horizontal axis. And then, once you have this dendrogram, the next step, or the full steps, are aligning each of the sequences, which you already had to have done in order to calculate the similarity matrix-- and again, these are a pairwise alignment of S1, S2, and S4.

Steps 1 and 2 were already done to get a similarity matrix. Now, step 3 is new. You align this alignment, we'll call it the pair S1, S3, with the pair S2, S4.

And you could imagine keeping doing this hierarchical process. If there were additional sequences which are even more distant related, let's say S5, you would take this alignment of S1, S2, S3, S4, and align it with a single sequence S5. So you can see how you can align not only sequences, but you can align pseudosequences, which have these little indel bashes in them. So that's one method.

This is a different method. And here, the premise is that you've got one sequence which is sufficiently close to all the other sequences that you can use it as an anchor sequence. And whatever indels you put in individually pairwise, for that sequence, can be propagated throughout the entire multisequence alignment.

So here, we start the same way. Here, we have five sequences instead of four, but it's the same thing. You do all pairwise similarities, and you give a score.

These scores are the scores that would have come out at the end of that traceback in the pairwise alignments. So this is not a pairwise matrix. This is the results of 5 times 4 over 2 pairwise alignments.

Each of these boxes itself is the outcome of a full matrix on S1 versus S2, for example. And you can see from these set of scores that the best score, or the best set of scores for any sequence, is S1 has the best score to S2, and it has the best score overall to all of the sequences. And so we'll use S1 as the focus of the star geometry.

And we'll say OK we've already compared every sequence to S1. We compared every sequence to every sequence. But let's focus on that. And now take wherever the indels were that were required to get the best score for S1 with each of the others, and have S1 in red in each case, and use that as the anchor.

And so then in the multialignment, you take all the indels relative to the red one, and introduce them so that it's the anchor. So those are two radically different ways. And we'll get to the Gibbs sampling later.

But the Gibbs sampling, just in a nutshell, is in general when you have a hard problem, where you can't comprehensively go through the entire space, what you do is sample it. You say, let's try a few things, and try to randomly sample it, and maybe even develop locally. If, after randomly sampling in certain places look better, then look near there, and find other solutions, and keep optimizing. That's the Gibbs in a nutshell.

Now, we have explored the space-time accuracy trade-offs. You can improve time by having this storing, this pairwise or multi sequence in a matrix-- so that's actually you've done a trade-off where you've taken up computer memory in order to save time. And then if you're willing to sacrifice a little accuracy or a little comprehensiveness, then you can save even more time or memory.

Now we want to use motifs, which is the sort of thing that you get out of local alignments, to find genes. And we're going to use the motifs and the finding genes as a way of introducing a particular motif, which is a CG motif, as a simple example of a hidden Markov model. Now, how do we find genes?

Genes have little bits of sequence at the beginning, in the middle, or the end, which are distinctive. They have distinctive properties, typically sequence properties. So at the beginning of the gene, before the protein-coding region or RNA-coding region, you'll have regulatory elements such as promoters and so-called CG islands.

Now, remember the CG islands, because that's what we're going to use to illustrate the HMMs. The CG islands are basically an abundance of the CG dinucleotide. Of the 16 different dinucleotides, CG happens to be underrepresented in general invertebrate genomes, and over-represented in promoter regions upstream from genes. And the reason is probably that they bind to transcription factors, and the transcription factors protect them from methylation, and thereby protect them from a mutagenic process that would otherwise cause them to become a TG.

Now, that's the example of a distinctive sequence element that indicates the beginning of a gene or just before beginning of the gene. Within the gene, especially-- well, only-- if it's a protein-coding region, you'll have preferred codons. These are preferences that are set by the particular abundances of transfer RNAs in the cell, as well as other constraints on the sequence. If you're in an organism that does RNA splicing, you'll have RNA splice signals, and they'll have distinctive sequence features.

You will have-- if you have RNA splicing, then you will have to maintain the translational reading frame across the splice junctions. That's a hint. If you have multisequence alignments, then you can look for conserved positions and interspecies conservation.

The ultimate cheat is if you have a cDNA in the case of species that are spliced, then you can figure out the splicing just empirically by the presence of actually sequencing the messenger RNA that encodes your gene. So you know there's a gene there because you found it present in the messenger RNA population, and you sequenced it.

Now, there are problems with each of these approaches. Promoters and CG islands are sort of degenerate. They're weak sequence signatures. There's a high variety, and they're used in combinations.

When we're looking at preferred codons, we need a lot of codons in a row to see a preference over random sequences. Random sequences will also contain some of the same codons. And if you need longer ones, then you'll miss tiny proteins. And we'll talk about this in just a moment, specific examples.

Similarly for RNA splicing, you can have weak motifs, again. And alternative splicing-- it's not like there's one specific splice that occurs in a particular gene segment. There can be multiple kinds.

Conservation requires that you have the right species, that at least some of the species in your multisequence alignment are just the right distance-- not too close, not too far away. And cDNAs are great, if you have them. But if you have very rare [? trends, ?] you need to have the cell type and the rare [INAUDIBLE], rare messenger RNA within a cell type.

So let's talk about the sizes of proteins. If you look here, I plotted the sizes of proteins in annotated genomes-- two of the first annotated genomes is the smallest eukaryote yeast and the smallest prokaryote Mycoplasma-- and asked what were the sizes of the proteins that are annotated? Proteins in quotes, because this is what humans and computer programs together chose to represent. This is not truth, necessarily.

And you can see it goes out to over 900 amino acids. And if you go to humans, this would go out to 10s of thousands of amino acids long for the largest proteins. But let's focus attention on the smallest proteins.

How is it that it precipitously drops off at 100 amino acids? Why are there so few proteins that are short? And there are slightly more short proteins in Mycoplasma? Any guesses why they're so few? Why does it drop off at 100 amino acids?

STUDENT: There are more but we can't find them?

GEORGE CHURCH: Right, there probably are more. It's not that we can't, it's that the annotators chose not to. And why did they choose not to?

They just agreed that they would stop at 100. That was getting too short. And this is what kind of illustrates why.

Here, every genome has its own GC content, its own codon usage, and so forth. Here, we're just talking about just the first order percentage of GC versus AT. And the genetic code, theoretically and as observed, can restrict genomes so that they really can have a minimum of 25% GC content, or 28%, and a maximum of 75%. And essentially all genomes fall in that range, and yeast is around 39% or so.

And then if you plot-- stop codons tend to be made up of As and Ts. The stop codons are TAG, TGA, and TAA. So if you have an AT-rich genome, you're going to tend to have a lot. You tend to run into a stop codon at random quite frequently.

So if you have a long open reading frame in an AT-rich genome, that's very-- if you have a modestly long open reading frame, that's very significant, AT-rich genome. But you have a GC-rich genome, then you can go for a long time at random without running into the stop codon, so it's less significant. So you need to have more codons in a row in a CG-rich genome in order to convince yourself.

So it's usually somewhere in between. And you can see that there's this general trend. You need to have more codons in a row to convince yourself as the GC content on the horizontal axis goes higher.

And basically, the place where you start getting too many false positives is around 100 amino acids. And so that's why the community just decided to cut off there. When we get to proteomics, we'll talk about ways that you can empirically, by mass spectrometry and so forth, find those small proteins. And genetically, of course, you can find them.

Let's talk about the most extremely small ones, and ask whether these extremely small open reading frames are interesting. And I think they're very extreme examples are very interesting. So the smallest that I know of is a pentapeptide, which is actually encoded in not just one, but many different phylogenetically diverse, large ribosomal RNAs.

So here, ribosomal RNA normally acts as part of the translation apparatus, but here, it is acting as the messenger RNA, as well, presumably a separate molecule, maybe possibly a degraded version of it. But in some way or another, the 23 sRNA encodes this pentapeptide, which is not just some junk-- you can have junk DNA, you can have junk peptides. But this one actually confers erythromycin resistance at low levels in wild type. It is not a mutant kind of peptide. It's the normal pentapeptide.

Now, here's three examples that are related to one another. They have somewhere between 14 and 16 codons, and they have this very strange amino acid composition when you do the translation conceptually in the computer. Remember, tryptophan were a rare amino acid. Well, here's two of them in a row. That's pretty unusual.

Here's seven phenylalanines in a short stretch. And here's seven histidines right in a row. This is really bizarre.

And what furthermore gets even more conspiratorial because these seven histidines in a row happen to be-- the next gene down is a histidine biosynthetic gene. And not only that, but about eight histidine genes in a row come after that. And the same thing with phenylalanines upstream of phenylalanine biosynthetic genes, and this weird excess of tryptophan is upstream of tryptophan biosynthetic genes.

So what does this all mean? What it means, probably-- and there's actually quite a bit of experiments on this-- is that this is an excellent feedback loop, where you want to do feedback in the most relevant way. So here, if you want to know whether you need to make tryptophan, phenylalanine, or histidine, you ask whether there's enough of it around to do translation. That's very relevant.

And so this has to be sensing the translation process itself. It's asking whether the transfer RNAs are charged up with amino acids enough that you're getting efficient translation. If you're not, then you'll pause here. That ribosome will hesitate, waiting for the right transfer RNA.

And as it hesitates, this RNA changes. It's folding. And a series of events results in-- if it's hesitating, then it wants to make the biosynthetic genes downstream to make more of the amino acids. So the tRNA has to be charged up. So you get this nice, little feedback loop that the hesitation causes a change in RNA, which causes change of transcription, and you make more of what you need.

So I think these are interesting examples. And of course, if you knew in advance you were looking for runs of histidines, that would be great. But for other open reading frames, there may be a different story. And so you need to have methods for looking for very short motifs.

So let's go back to the bigger question of motifs, and ask, how do we deal with them more rigorously? And the way we deal with them more rigorously is these profiles. Now, what we're going to do is we're going to take a multisequence alignment. You now know how to do multisequence alignments.

And we now want to capture that information, and deal with these position-specific profiles. Remember I mentioned the PSI-BLAST and other algorithms. You acknowledge that you don't have a generic substitution matrix for all positions in all proteins or all nucleic acids.

You have a different substitution matrix for every position. Because one position might be, say, an alpha helix. We have one substitution matrix. And another one might be in a coil.

So here, this is all about what motifs are all about. Each position has a different set of rules. So the first position in this tetranucleotide-- it doesn't care what it is. It can be A, C, G, or T.

These are four different sequences, real start sites, that we've aligned, either manually or by computer. This is dead easy to do the alignment, but the interpretation here is the position upstream of the start codon doesn't matter. So your matrix down below is-- A, C, G and T each get a 1, which is a count.

We could do it in terms of frequencies, percentages. We're doing it in terms of counts here because that's just a restatement of the data. The T and the G position at the 3-prime end of the codons are, in this small sample, invariant. And so they get a count of 4 for the correct base and count of 0 for all the alternatives, A, C, and G, for example, instead of T.

And the A position is not quite invariant in this sample. GTG is a perfectly good start codon in, say, 1 sequence in 10 or 1 in 4 in this case. And so you get 3 and a 1.

So this is the weight matrix or position-sensitive substitution matrix. This is more precise than, say, a consensus sequence or a single sequence from the sample. But it's not the most precise way of representing this.

It's position sensitive, but we've lost the higher order correlations between positions. In other words, we've lost the dependencies of adjacent bases, or bases that are a few bases away. But let's see how this plays out, this position sensitive. This is another way of representing is in terms-- is an information theory version of this, where the full height of each of the bases is 2 bits.

And that's the same 2 bits we talked about in the first lecture, since there are four bases. And this is the same motif, ATG. The T and G were invariant in this larger sample, or nearly invariant in the sample size of now instead of just 4, but more than 1,000 sequences.

And again, A and G were the predominant ones. You can see a little bit of a T there in the first position. And then the base just upstream from the ATG is almost completely random. And so its information content is close to nil, and so it's 0 bits.

Now, this is easy enough that you can just do a big search aligning on the ATG, which is a very striking thing, and look to see if there's any other residual information content to the side. And sure enough, you find this little blip of Gs and As, mostly, at minus 9 relative to the A of ATG at 0. And it turns out that-- again, experimentally verified-- this motif-- so the ATG motif binds to transfer RNA, and the GA-rich motif actually binds to a ribosomal RNA sequence.

And so then basically, the messenger RNA is coaxed into the right position, to be in the right position of the ribosomes where the tRNA can bind the initiator. So here's an example where you can do a multisequence alignment. Here are 1,000 of sequences. k equals 1055-- remember, this is exponential of k. And you can find these motifs that have great biological significance.

Now, once you've done multisequence alignment and you've derived the weight matrix, this position-sensitive substitution matrix, now you want to be able to search the genome for these things. You know what a start motif looks like. You want to find them all. And it wouldn't just be the ATG, it would be this full, including the GA-rich motif.

And the way you do that is now take this weight matrix, and ask for each-- we're scanning the genome, and we run into the sequence [? AAT ?] AATG. Now you want to know, how good a match is that to this weight matrix, which was taken from either 4 sequences or 1,000 sequences?

And the way you do it is for each position, you ask what was the score in the whole learning set? And now this should be a now independent test set you're trying this out on. Here, the learning set and the test site are the same.

But you basically have the A is a score of 1, which is not going to be a big contribution because they were all the same. So then the second A is a score of 3, and the T and G are a score of 4, for a total score of 12 for this particular tetranucleotide instance of this motif represented by this weight matrix.

And then you can see that the top three sequences, which all have ATG, have the best scores. And the bottom one, GTG, even though it's a valid member of the learning set, it was something which was underrepresented statistically. GTG tended to be less frequently encountered than ATG, and so it gets a lower score when you search the genome for it. So if you prioritize these, they would be prioritized in this order by 12, 12, 12, 10.

So now the final topic, which talks about a very simple and short motif, which is the CG motif, which we claimed is over-represented in promoters in vertebrates. But before we talk about these very short motifs, let's talk about why we have probabilistic models in sequence analysis in general. And there are three main uses.

One of them is recognition-- for example, the recognition that we've been doing is, is a particular sequence of protein start? In other words, does it have a score which is statistically significant? That's basically what we were doing, very anecdotally, in the previous slides.

Or another task is discrimination. We ask questions like, is this protein more like a hemoglobin or like a myoglobin? The first question is about one sequence relative to, say, a weight matrix. The

Other one is about two sequences, asking how-- or three sequences-- whether a particular protein is more like one than another. And in a database search, we would go through. A question might be like, what are all sequences in [INAUDIBLE] that look like a serine protease? This would be asking for recognition multiple times, over and over.

So here is the basic idea-- which will be a Bayesian idea soon, in the next slide-- is assign a number to every possible sequence such that the probability of that sequence given a model-- so this jargon here, P of s/m, s bar m-- is the probability that you would get that sequence given a model. So the model might be this weight matrix we've been talking about or it could be something more complicated. So what's the probability that we get the sequence ATG, given the model, the full weight matrix model?

And as with any good probability, as we mentioned in the first class, they should sum to 1. If you sum up the sigmas of s, you sum over all sequences, then the probability given the models should sum to 1. Now, that will be true for the p of the sequences given a model summed over all sequences.

We can also have the probability of a sequence in your population of sequences irrespective of model. And those should sum to 1. And the probability of models in your collection of models, irrespective of sequence.

And here's a very useful theorem, called Bayes' theorem. And this is completely general. It doesn't depend on models and sequences. You could just call it m and s, where m and s are just two things.

And this is generally true, is that the probability that the model given the sequence is equal to the probability of the model times the probability of the sequence given the model divided by the probability of the sequence. And more jargon, but explanation of some of these terms here, is that the probability of the model and the probability of the sequence are prior probabilities. These are probabilities which are not conditional. They do not depend on something else.

Well, when you have this little bar in the middle, it means that you have the probability of the model given the sequence. It's called a posterior probability. Now let's see what all this Bayesian stuff is useful for.

We're going to be doing-- of the various applications, we had recognition discrimination and database search. So here's the example of a database search. We'll have two models, a model that we actually have a hydrolase and the model that we have randomness. So we call this the null model or n model, and m is the model that we're interested in, they're hydrolases.

So we have random bases or random amino acids. This is hydrolase and amino acids. So we want to report all the sequences where the probability that that sequence, given the model, is better than that sequence given a null model or random amino acids, that that is significant, and it's significant by the delta between just the null versus the probability of the model in general. So if we look, if we, let's say, do a database search where we have scoring metrics just as the ones we developed earlier in the talk, and we score for random sequences, we'll get one distribution in orange. And if we score for fide hydrolases, we might get this distribution in blue.

And we're asking whether the probability of getting a particular sequence given the model this is a hydrolase is better than probability of getting that sequence at random, the orange. And you want that to be statistically significant. So you can rephrase this in terms of bits, or in terms of significance level of probability of 5%, which is typically the case.

Now, when we're talking about the probability of a particular sequence, where we can have deviations from randomness at the mononucleotide level, at the dinucleotide level, and so on, and rather than just dump this on you as a mathematical fact, I want to give you some biological rationale for why you can have nonrandomness at every order of a Markov chain, meaning every length of sequence. So the first-order chain, the lowest-order chain, would be mononucleotides. And you might have a bias where C would be rare because the Cs mutate into Us.

And in organisms that lack a uracil glycosylase, which would then return it back to a C, Cs will change into Us because it's a very common chemical reaction. It's called cytosine deamination. But a deoxy U is an abnormal base. It's recognizable as an abnormal base, and there's repair in most organisms that [INAUDIBLE], but there's some that don't. And there's a tendency of those genomes to aim towards high content. The Cs disappear, and hence, take the Gs with them.

Similarly, many organisms repair-- well a T near a T in the presence of ultraviolet light will get mutated to something else. And if you can't repair that back to a T-T sequence, it gets repaired to something else or it gets mutated to something else. And so you'll lose that particular dinucleotide out of the 16 possible dinucleotides.

We've already mentioned that CG is rare. And the reason is that this is methylated for various regulatory reasons. And now, because it's methylated, even if you have uracil glycosylase, which would then take all the regular Cs that turn into Us, deoxy Us, and turn them back into deoxy Cs, now a 5-methyl C turns into a T, and you can't tell that it's abnormal. T is a perfectly reasonable thing to get. And so every place you've got a methyl CG turns into a TG, and you tend to lose the CGs, unless they're not methylated. And we'll get to that.

And similarly, you can have rare codons. And hence, these turn into rare triplets. You can have rare tetranucleotides if you, for example, have a methylase, the methylase is a pentanucleotide, and every time you see that-- every time the bacteria sees this related CTAG-G sequence, that says, oh, that must have been one of these methylation deamination problems. Let's fix it up. Let's make it this pentanucleotide. And the CTAG tends to be underrepresented as a consequence.

Similarly, very long stretches of As-- not just tetranucleotides, but you can get excesses of As due to the fact that messenger RNAs end in polyA. They get reverse transcribed, reinserted into the genome, and now you've got a polyA track. Or you can get polymers in general by polymerase slippage. So all these things can cause biases. And I just elaborated on one of them here, which is the triplet bias, documented here that this 10 times lower frequency of ATG than of some of the other arginine codons.

So now let's talk about a Markov model. This is not a hidden Markov model yet. In just a moment, it will be.

It's a Markov model because we're asking what is-- the columns that we had kept independent when we were making profiles or weight matrices, we said the two nucleotides, whether CG or AA or whatever, were independent. Now, we're no longer going to make them independent. We will allow them to recognize the co-dependence.

Forget the pluses right now. Just assume they'll be explained when we get to the hidden part of this. So they're hidden for now.

But what we're talking about is, what's the probability of getting an A given an A? We've got an A in the first, in the 5-prime position. What's the probability now of getting an A dependent on that one? So we're recognizing that dependence.

We've said that CGs are underrepresented in the genome as a whole, and they're over-represented in promoters. So this particular transition of what's the probability of getting a G given a C in the 5-prime position-- this is one of those conditional probabilities. This is a Bayesian that we had set up a couple of slides back.

And so this particular arrow going from a C to a G is represented by this probability. And you can see going the other way is a different probability. That would be p of C given G.

And these little arrows will refer to itself, is example of a p of an A given an A. So this is an AA dinucleotide. And you can see there's 16 possible transitions, including four homopolymers, AA, TT, CC, GG, and 12 transitions of the other dinucleotides.

Now, what do we mean by hidden? We've got CG islands where the CGs have been protected from methylation, and hence, protected from mutations. So they're fairly abundant.

They're involved in regulation in binding transcription factors. And these islands will be a variable length and just have an increased concentration of CGs. And then outside are the ocean, which are not protected.

They're not involved in transcription, and they mutate. And they are very low in CGs. And you want to know where the island begins and ends because that helps you know where the regulatory factors are.

So now, the hidden part is when you look at a new sequence, you won't know whether you're in an island or not. And so this Markov model that you have has to be different for whether you're in an island or not, but you don't know what you're in. So here is the hidden part.

So you've got a Markov model for the transitions within an island. And so in that case, you expect the CGs to be high, roughly the same as the other dinucleotides, possibly higher. And in the oceans where they're lost, you expect the CG, this particular transition from C to G, to be low, and most of the other transitions to be normal, maybe taking up some of the slots.

So there's 16 different dinucleotides in islands on the left. And there are 16 in oceans on the right. In addition, there's a whole set of transitions between islands and oceans.

The genome is not just blocks. They're all connected. And so you can make a transition from any nucleotide in an island to any nucleotide in an ocean.

And so here's one that's illustrated, this dotted, brown line, where it says probability of a C minus-- meaning in an ocean-- given that you have an A plus-- meaning in an island-- in the 5-prime position. So that would be a transition point going 5-prime to 3-prime, from an island into an ocean, going from an A to a C.

Aren't you glad that I picked a dinucleotide to illustrate this? OK, here's a real example. Here's an example where I've cut and pasted a very short sequence with only one ocean on the left, and one island on the right, in bold and capital letters.

You're given this as a learning set. Somebody has, by hand, decided that the boundary occurs at this first CG dinucleotide. There are no CGs to the left, and there are three CGs to the right.

And so when you make this table-- we'll call this an A table later on-- this A table has the transition from an A in the 5-prime position to an A in the 3-prime position. So that's the p A given A. And here's the CG dinucleotide, C to G transition, all in an island indicated by plus. And you can see that's quite frequent.

And then below it, let's look at the same CG dinucleotide going from C to G in an ocean. And here it's unobserved in this little toy example that I gave you, so it's a 0. So 43% in this actual example-- and you can work the numbers out because it's all here-- and there's only one transition between islands and oceans, and that happens to be a CC, a C in an ocean going to a C in an island. And that gives us 0.2.

And all the rest are 0s. Now, 0s are a problem, both for the CG dinucleotide in the ocean and for the transitions between oceans and islands. And the way you handle it is called pseudocounts.

You basically say, what if we just missed finding that thing? We're going to add 1 to it because however big the counts are, you can always add 1, and that would give you some feeling for the-- you don't really have 0s there. You can't trust 0s.

And there's even a more rigorous way of doing it called Dirichlet, where you can do these pseudocounts. And so you can see. You can actually calculate these conditional probabilities by hand in the privacy of your home, not while the hordes are waiting to get into the room.

And you can recreate these numbers with that simple formula there. Now this, is a real training set based on 48 known islands, again annotated by some person. And you can see those that this A matrix, focusing on those things that were 43 and 0 before, now more realistic numbers are 27% and 8% for an island and an ocean, respectively.

Now we're going to plug these numbers-- basically, I've cut off the transition tables, which are off to the right. Now let's use them to actually do an HMM. In the Viterbi algorithm-- remember we said dynamic program is a hero, and we're going to end on this.

The recursion we have here, the Viterbi score for-- so l and k are the states. There are two states, island plus, ocean minus. And i is the sequence.

Here, the sequence length is 4. i goes from 1 to 4. And the sequence we're testing is, is CGCG in an ocean or an island? What's your guess?

That's a pretty extreme case. But this is actually using the numbers from the previous slide, which were taken from real oceans and islands. And so you start out with the probabilities being just equally probable that you can start at the C.

So there are eight different states, and so we just divide 1 over 8 is a starting point, or 0.125. And so there are two possible places it can be, and they're equally probable. It's in an ocean or island, just given the C, 1/8.

Now you make a transition where you multiply this times the A matrix, A sub k l, so you're going from state 1 to state 1, from an island to an island. And if you look back one slide, you remember there is a 0.27 for going to a CG dinucleotide. So the recursion here is you multiply this-- is an emission, which is always 1.

You multiply the maximum of the previous Viterbi, so i plus 1 and i, times the A matrix, which in the previous slide is .27. So the previous one was 1/8, and then times 0.27, you get 0.034. And if you started in an ocean and stayed in an ocean, it would already drop to 0.01.

So you can see the better probability is already that you're in an island. And if you carry this all the way out to all four tetranucleotides, you get a much higher probability of being in the island, of 0.032, than being in the ocean, 0.002. Question.

STUDENT: Do you know the basis for thinking that the context for a dinucleotide is either an ocean or an island, in other words, only two states? Why couldn't the context be five states?

GEORGE CHURCH: OK.