PROFESSOR: All right. We should probably get started. So RNA plays important regulatory and catalytic roles in biology, and so it's important to understand its function. And so that's going to be the main theme of today's lecture.

But before we get to that, I wanted to briefly review what we went over last time. So we talked about hidden Markov models, some of the terminology, thinking of them as generative models, terminology of the different types of parameters, the initiation probabilities and transition probabilities and so forth. And Viterbi algorithm, just sort of the core algorithm used whenever you apply HMMs. Essentially, you always use the Viterbi algorithm.

And then we gave as an example the CpG Island HMM, which is admittedly a bit of a toy example. It's not really used in practice, that illustrates the principles. And then today we're going to talk about a couple of real world HMMs.

But before we get to that, I just wanted to-- sort of toward the end, we talked about the computational complexity of the algorithm, and concluded that if you have a case state HMM run on a sequence of length L, it's order k squared L. And this diagram is helpful to many people in sort of thinking about that.

So you can have transitions from any state-- for example, from this state-- to any of the other five states, and there's five-state HMM. And when you're doing the Viterbi, you have to maximize over the five possible input transitions into each state. And so the full set of computations that you have to do from going from position i to i plus 1 is k squared. Does that make sense? And then there's L different transitions you have to do, so it's k squared L.

Any questions about that? OK. All right and, so the example that we gave is shown

here. And what we did was to take an example sort of where you could sort of see the answer-- not immediately see it, but if we're thinking about it a little, figure out the answer. And then we talked about how the Viterbi algorithm actually works, and why it makes the transitions at the right place.

It seems to intuitively like it would make a transition later, but actually transitions at the right place. And one way to think about that is that these are not hard and fast decisions because you're optimizing two different paths. At every state, you're considering two possibilities.

And so you explore the possibility of-- the first time you hit a c, you explore the possibility of transitioning from genome to island, but you're not confirming whether you're going to do that yet until you get to the end and see whether that path ends up having a higher probability at the end of the sequence than the alternative. So that's sort of one way of thinking about that. Any questions about this sort of thing, how to understand when a transition will be made?

And I want to emphasize, for this simple HMM, we talked about you can kind of see what the answer's going to be. But if you have any HMM, any sort of interesting real world HMM with multiple states, there's no way you're going to be able to see it. Maybe you could guess what the answer might be, but you're not going to be able to be confident of what that is, which is why you have to actually implement it.

All right, good. Let's talk about a couple of real world HMMs. So I mentioned gene finding. That's been a popular application of HMMs, both in prokaryotes and eukaryotes. There's some examples discussed in the text.

Another very popular application are so-called profile HMMs. And so this is a hidden Markov model that's made based on a multiple alignment of proteins which have a related function or share a common domain. For example, there's a database called Pfam, which includes profile HMMs for hundreds of different types of protein domains.

And so once you have many dozens or hundreds or thousands of examples of a

protein domain, you can learn lots of things about it-- not just what the frequencies of each residue are in each position, but how likely you are to have an insertion at each position. And if you do have an insertion, what types of amino acid residues are likely to be inserted in that position, and how often you are likely to have a deletion at each position in the multiple alignment.

And so the challenge then is to take a query protein and to thread it through all of these profile HMMs and ask, does it have a significant match to any of them? And so that's basically how Pfam works. And the nice thing about HMMs is that they allow you to-- if you want to have the same probability of an insertion at each position in your multiple alignment, you can do that. But if you have enough data to observe that there's a five-fold higher likelihood of having an insertion at position three in a multiple alignment than there is at position two, you can put that in. You just change those probabilities.

So in this HMM, each of the hidden states is either an M state, which is a match state, or an I state, or an insert state. And so those will emit actual amino acid residues. Or it could be a delete state, which is thought of as emitting a dash, a placeholder in the multiple alignment. So these are also widely used.

And then one of my favorite examples-- it's fairly simple, but it turns out to be quite useful-- is the so-called TMHMM for prediction of transmembrane helices in protein. So we know that many, especially eukaryotic proteins, are embedded in membranes. And there's one famous family of seven transmembrane helix proteins, and there are others that have one or a few transmembrane helices. And knowing that a protein has at least one transmembrane helix is very useful in terms of predicting its function.

You predict it's localization. And knowing that it's a seven transmembrane helix protein is also useful. And so you want to predict whether the protein has transmembrane helices and what their orientation is. That is, proteins can have their end terminus either inside the cell or outside the cell. And then of course, where exactly those helices are.

And this program has about a 97% accuracy, according to [? the author. ?] So it works very well. So what properties do you think-- we said before that you have to have strongly different emission probabilities in the different hidden states to have a chance of being able to predict things accurately. So what properties do you think are captured in a model of transmembrane helices? What types of emission probabilities would you when you have for the different states in this model? Anyone?

So for this protein, what kind of residues would you have in here? Oops, sorry. I'm having trouble with this thing. All right, here in the middle of the membrane, what kind of residues are you going to see there?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Those are going to be hydrophobic. Exactly. And what about right where the helix emerges from the membrane? [INAUDIBLE] charge residue's there to kind of anchor it and prevent it from sliding back into membrane.

And then in general, both on the exterior and interior, you'll tend to have more hydrophilic residues. So that's sort of the basis of TMHMM.

So this is the structure. And you'll notice that these are not exactly the hidden states that correspond to individual amino acid residues. These are like meta states, just to illustrate the overall structure.

I'll show you the actual states on the next slide. But these were the types of states that the author, Anders [? Crow ?], decided to model. So he has sort of a-- focuses here on the helix core.

There's also a cytoplasmic cap and a non-cytoplasmic cap. Oops, didn't mean that. And then there's sort of a globular domain on each side-- both on the cytoplasmic side, or you could have one on the non-cytoplasmic side. OK, so there's going to be different compositions in each of these regions.

Now one of the things we talked about with HMMs is that if you were-- now let's

think about the helix core. The simplest model you might think of would be to have sort of a helix state, and then to allow that state to recur to itself. OK, so this type of thing where you then have some transition to some sort of cap state after, this would allow you to model helices of any length.

But now how long are transmembrane helices? What does that distribution look like? Anyone have an idea? There's a certain physical dimension. [INAUDIBLE]

It takes a certain number residues to get across here, and then that number is about 20-ish. So transmembrane helices tend to be sort of on the order of 20 plus or minus a few. And so it's totally unrealistic to have a transmembrane helix that's, like, five residues long.

So if you run this algorithm in generative mode, what distribution of helix lengths will you produce? We're running in generative mode where we're going to let, remember, to generate a series of hidden states and then associated amino acid sequences. It's coming from some, let's say-- I don't know. What kind of states are there here? [INAUDIBLE] plasmic.

Let's say goes into helix, hangs out here. I'm sorry, is there an answer to this question? Anyone? I don't know how long-- if I let it run, it'll generate a random number. It depends on what this probability is here.

Let's call this probability p, and then this would be 1 minus p. OK, so obviously if 1 minus p is bigger, it'll tend to produce longer helices. But in general, what is the shape of the distribution there of consecutive helical states that this model will generate?

AUDIENCE: Binomial.

PROFESSOR: Binomial. OK, can you explain why?

AUDIENCE: Because the helix would have to have probable-- the helix of length n would occur 1 minus p to the n power.

PROFESSOR: OK, so a helix of length 10 with a probability of then, say, let's call it L, for the length

of the helix, equals n is 1 minus p to the n, right? Is that binomial? Someone else?

AUDIENCE: Yeah. Is it a negative binomial?

PROFESSOR: Negative binomial. OK.

AUDIENCE: [INAUDIBLE] states and a helix state before moving out [INAUDIBLE].

PROFESSOR: Yeah. So the distribution is going to be like that. You have to stay in here for n and then leave. So this is the simplest-- you can have special cases of binomial and negative binomial. But in general, this distribution is called the geometric distribution. Or a continuous version would be the exponential distribution.

So what is the shape of this distribution? If I were to plot n down here on this axis, and the probability that L equals n on this axis, what kind of shape-- could someone draw in the air? So you had up and then down?

OK, so actually, it's going to be just down. Like that, right? Because as n increases, this goes down because 1 minus p is less than 1. So it just steadily goes down.

And what is the mean of this distribution? Anyone remember this? Yeah, so there's sort of two versions of this that you'll see.

One of them is the 1 minus p n minus 1 p, and one of them is this. And so this is the number of failures before a success, if you will. Successes lead to the helix.

And this is the number of trials till the first success. So one of them has a mean that's 1/p, and the other has a mean that's 1 minus p over p. So usually, p is small, and so those are about the same.

So 1/p. You could think that 1/p is roughly right. And so if we were to model transmembrane helices, and if transmembrane heresies are about-- I said about 20 residues long-- you would set p to what value to get the right mean?

AUDIENCE: 0.05.

PROFESSOR: Yeah. 0.05. 1/20, so that 1 over that will be about 20, right? And then 1 minus p

would, of course, be 0.9.

So if I were to do that, I would get a distribution that looks about like this with a mean of 20. But if I were to then look at real transmembrane helices and look at their distribution, I would see something totally different. It would probably look like that.

It would have a mean around 20. But the probability of anything less than 15 would be 0. That's too short. It can't go across the membrane.

And then again, you don't have ones that are 40. They don't kind of wiggle around in there and then come out. They tend to just go straight across.

So there's a problem here. You can see that if you want to make a more accurate model, you want to not only get the right emission probabilities with the right probabilities of hydrophobics and hydrophilics and the different states, but you also want to get the length right. And so the trick that-- well, actually, yeah. Can anyone think of tricks to get the right length distribution here?

How do we do better than this? Basically, hidden Markov models where you have a state that will recur to itself, it will always be a geometric distribution. The only choice you have is what is that probability. And so you can get any mean you want, but you always get this shape.

So if you want a more general shape, what are some tricks that you could do? How could you change the model? any ideas? Yeah, go ahead.

AUDIENCE: [INAUDIBLE] have multiple helix states.

PROFESSOR: Multiple helix states. OK. How many?

AUDIENCE: Proportional to the length we want, [INAUDIBLE].

PROFESSOR: Like one for each possible length.

AUDIENCE: It'd be less than one length.

PROFESSOR: Or less than one. OK. So you could have something like-- I mean, let's say you have like this. Helix begin-- or, helix 1, helix 2. You allow each of these to recur to themselves. What does that get you?

This actually gets you something a little bit better. It gives you a little bit about of-- it's more like that. So that's better.

But if I want to get the exact distribution, then actually one-- so this is the solution that the authors actually used. They made essentially 25 different helix states, and then they allowed various different transitions here. So it's a larger arbitrary here, but they have this special state three that can kind of take a jump.

So it can just continue on to four, and that'll make your maximum length helix core. Or it can skip one, go to five, and that'll make a helix core that's one residue shorter than that, or it can skip two, and so forth. And you can set any probabilities you want on these transitions.

As so you can fit basically an arbitrary distribution within a fixed range of lengths that's determined by how many states you have. OK, so they really wanted to get the length distribution right, and that's what they did. What's the cost of this? What's the downside? Simona?

AUDIENCE: I was just going to ask, it looks like from this your minimum helix length could be four.

PROFESSOR: Yeah. That's a good question. Well, we don't know what the probabilities-- they say said on that. Well, did they really mean that? And also, that's only the core, and maybe these cap things can be-- yeah, that seems a little short to me. So yeah, I agree. I'm not sure. It could just be for the sake of illustration, but they don't actually use those. But anyway, I'll probably have to read the paper. I haven't read this paper for many years so I don't remember exactly the answer to that.

But I have a citation. You can look it up if you're curious. But the main point I wanted to make with this is just that by setting an arbitrary number of states and putting in

possible transitions between them, you can actually construct any length of distribution you want. But there is a downside, and what is that downside?

AUDIENCE: Computational cost.

PROFESSOR: Yeah, the computational cost. Instead of having one helix state, now we've got 25 or something. So and the time goes up by the square of the number of states, so it's going to run slower. And you also have to estimate all these parameters.

OK, so here's an example of the output of the TMHMM program for a mouse chloride channel gene, CLC6. So the program predicts that there are seven transmembrane helices, as shown by these little red blocks here. You can see they're all about the same-- about 20 or so-- and that the program starts outside and ends inside.

So let's say you were going to do some experiments on this protein to test this prediction. So one of the types of experiments people do is they put some sort of modifiable or modified residue into one of the spaces between the transmembrane helices. And then you can test, by modifying this cell with something that's a non-permeable chemical, can you modify that protein? So only if that stretches on the outside of the cell will you be able to predict it.

So that's a way of testing the topology. So if you were doing those types of experiments, you might actually-- like maybe you're not sure if every transmembrane helix is correct. There could be some where the boundaries were a little off, or even a wrong helix.

And so one of the things that you often want with a prediction is not only to know what is the optimal or most likely prediction, but also how confident is the algorithm in each of the parts of its prediction. How confident is it in the location of transmembrane helix three or the probability that actually there is a transmembrane helix three. And so the way that this program does that is using something called the forward-backward algorithm.

So those of you who read the Rabener tutorial, it's described pretty well there. The

basic idea is that I mentioned that this Po-- the probability of the observable sequence summing over all possible HMM structures or all possible sequences of hidden states-- that is possible to calculate.

And the way that you do it is you run an algorithm that's similar to the Viterbi, but instead of taking the maximum entering each hidden state at intermediate positions, you sum those inputs. So you just do the sum at every point. And it turns out that will calculate the sum of the two values at the end-- or the k values at the end will be equal to the sum of the probabilities of generating the observable sequence over all possible sequences of hidden states. OK, so that's useful.

And then you can also run it backwards. There's no reason it has to be only going in one direction. And so what you do is you run these sort of summing versions of the Viterbi in both the forward direction and also run one in the backward direction.

And then you take a particular position here-- like let's say this is your helix state, for example. And we're interested in this position somewhere in the middle of the protein. Is that a helix or not?

And so basically you take the value that you get here from the forward in your forward algorithm and the value that you get here in the backward algorithm, and multiply those two together, and divide by this Po. And that gives you the probability. So that ends up being a way of calculating the sum of all the parses that go through this particular position i in the sequence in that particular state.

I mean, I realize that may not have been totally clear, and I don't want to take more time to totally go into it, but it is pretty well described and Rabener. And I'll just give you an example. So if you're motivated, please take a look at that. And if you have further questions, I'd be happy to discuss during office hours next week.

And this is what it looks like for this particular protein. So you get something called the posterior probability, which is the sum of the probabilities of all the parses. And they've plotted it for the particular state that is in the Viterbi path, that is in the optimal parse-- so for example, in blue here.

Well, actually, they've done it for all the different states here. So blue is the probability that you're outside. OK, so it's very, very confident that the end terminus of the protein is outside the cell. It's very, very confident in the locations of transmembrane helices one and two.

It actually more often than not thinks there's actually a third helix right here, but that didn't make it in the optional parse. That actually occurs in the majority of parses, but not in the optimal. And it's probably because it would then cause other things to be flipped later on if you had transmembrane helix there.

It's not sure whether there's a helix there or not, but then it's confident in this one. OK, so this gives you an idea. Now if you wanted to do some sort of test of the prediction, you want to test probably first the higher confidence predictions, so you might do something right here.

Or if maybe from experience you know that when it has a probability that's that high, it's always right, so there's no point testing it. So you should test one of these kind of less confident regions. So this actually makes the prediction much more useful to have some degree of confidence assigned to each part of the prediction.

So for the remainder of today, I want to turn to the topic of RNA secondary structure. So at the beginning, I will sort of get through some nomenclature. And then to motivate the topic, give some biological examples of RNA structure. Gives me an excuse to show some pretty pictures of structure.

And then we'll talk about two approaches which are two of the most widely used approaches toward predicting structure. So using evolution to predict structure by method of co-variations, which works well when you have many homologous sequences. And then using sort of first principles thermodynamics to predict secondary structure by energy minimization where obviously you don't need to have a homologous sequence present. And the nature biotechnology primer on RNA folding that I recommended is a good intro to the energy minimization approach.

So what is RNA secondary structure? So you all know that RNAs, like proteins, have

a three-dimensional tertiary fold structure that, in many cases, determines their function. But there's also sort of a simpler representation of this structure where you just describe which pairs of bases are hydrogen bonded to one other.

OK, and so for RNA-- so it's a famous example of an RNA structure, this sort of clover leaf structure that all tRNAs have. The secondary structure of the tRNA is the set of base pairs. So it's this base pair here between the first base and this one toward the end, and then base right here, and so forth.

And so if you specify all those base pairs, then you can then draw a picture like this, which gives you a good idea of what parts of the RNA molecule are accessible. So for example, it won't tell you where the anticodon loop is, which is sort of the business end of the tRNA. But it narrows it down to three possibilities.

You might consider that, or that, or down here. It's unlikely to be something in here because these bases are already paired. They can't pair to message. So it gives you sort of a first approximation toward the 3D structure, and so it's quite useful.

So how do we represent secondary structure? So there's a few different common representations that you'll see. So one is-- and this is sort of a computer-friendly but not terribly human-friendly representation, I would say-- is this sort of dot in parentheses notation here.

So the dot is an unpaired base and the parenthesis is a paired base. And how do you know-- chalk is sort of non-uniformly distributed here-- so if you have a structure like this and you have these three parentheses, what are they paired to? Well, you don't know yet until you get further down.

And then each left parenthesis has to have a right parenthesis somewhere. So now if we see this, then we know that there are two unpaired bases here, and then there's going to be three in a row that are paired-- these guys. We don't know what they're paired to yet.

Then there's going to be a five base pair loop, maybe a little pentagon type thing. Two, three, four-- oops-- four, five. And this one would be the right parentheses that

pair with the left parentheses over here. I should probably draw this coming out to make it clearer that it's not paired. So this notation you can convert to this. So after a while, it's relatively easy to do this, except when they're super long.

So that's what the left part of that would look like. So what about the right part? So the right part, we have something like one, two, three, four, bunch of dots, and then we have two, and then a dot, and then two. What does that thing look like?

So that's going to look like four bases here in a stem. Big loop, and then there's going to be two bases that are paired, and then a bulge, and then two more that are paired. These things happen in real structures.

OK and then the arced notation is a little more human-friendly. It actually draws an arc between each pair of bases that are hydrogen bonded. So I'm sure you can imagine what those structures would look like.

And it turns out that the arcs are very important. Like whether those arcs cross each other or not is sort of a fundamental classification of RNA secondary structures, into the ones that are tractable and the ones that are really difficult. So pretty pictures of RNA.

So this is a lower resolution cryo-EM structure of the bacterial ribosomes. Remember, ribosomes have two sub-units-- a large sub-unit, 50S, and a small sub-unit, 30S. And if you crack it open-- OK, so you basically split. You sort of break the ribosome like that, and you look inside, they're full of tRNAs.

So there are three pockets that are normally distinguished within ribosomes. The A site-- this is the site where the tRNA enters that's going to add a new amino acid to the growing peptide chain. The P site, which is this tRNA will have it [INAUDIBLE] with the actual growing peptide. And then the exit tunnel where this tRNA will eventually-- the exit, the E site, which is the one that was added a couple of residues ago.

So people often think of RNA structure just in terms of these secondary structures because they're much easier to generate than tertiary structures, and they give you-

- like for tRNA, it gives you some pretty good information about how it works. But for a large and complex structure like the ribosome, it turns out that RNA is actually not bad at building complex structures. I would say it's not as good as protein, but it is capable of constructing something like a long tube.

And in fact, in the ribosome, you find such a long tube right here. That is where the peptide that's been synthesized exits the ribosome. And you'll notice it's not a large cavity in which the protein might start folding.

It's a skinny tube that is thin enough that the polypeptide has to remain linear, cannot start folding back on itself. So you sort of extrude the protein in a linear, unfolded confirmation, and let it fold outside of the ribosome. If it could fold inside that, that might clog it up. That's probably one reason why it's not designed that way. I'm sure that was tried bye evolution and rejected.

So if you look at the ribosome-- now remember, the ribosome is composed of both RNA and protein-- you'll see that it's much more of one than the other. And so it's really much more of the fettuccine, which is the RNA part, than the linguini of the protein. And if you also look at the distribution of the proteins on the ribosome, you'll see that they're not in the core.

They're kind of decorated around the edges. It really looks like something that was originally made out of RNA, and then you sort of added proteins as accessories later. And that's probably what happened. This is based on the structures that were solved a few years ago.

If you then look at where the nearest proteins are to the active site-- actual catalytic site-- remember, the ribosome catalyzes peptide in addition to an amino acid to a growing peptide, so peptide bond formation-- you'll find that the nearest proteins are around 18 to 20 angstroms away. And this is too far to do any chemistry, so the active site residues or molecules need to be within a few angstroms to do any useful chemistry. And so this basically proves that the ribosome. Is a ribozyme. That is, it's an RNA enzyme. RNAs is [INAUDIBLE].

So here is the structure of a ribosome. It's very kind of beautiful, and it's impressive that somebody can actually solve the structure of something this big. But what is actually the practical use of this structure? Turns out there's quite an important practical application of knowing the structure. Any ideas?

AUDIENCE: Antibiotics.

PROFESSOR: Antibiotics. Exactly. So many antibiotics work by taking advantage of differences between the prokaryotic ribosome structure and eukaryotic ribosome structure. So if you can make a small molecule-- these are some examples-- that will inhibit prokaryotic ribosomes but hopefully not inhibit eukaryotic ribosome, then you can kill bacteria that might be infecting you.

So non-coding RNA. So there's many different families of non-coding RNAs, and I'm going to list some in a moment. And I'm going to actually challenge you, see if you can come up with any more families of non-coding RNAs.

But they're receiving increasing interest, I would say, ever since micro RNA's were discovered. Sort of a boom in looking at different types of non-coding RNAs. Link RNA is also important and interesting, as well as many of the classical RNA's like tRNAs and rRNAs and snoRNAs.

There may be new aspects of their regulation and function that will be interesting. And so when you're studying a non RNA, it's very, very helpful to know its structure. If it's going to base pair in trans with some other RNA-- as tRNAs do, as micro RNA's do, for example, or snRNAs and snoRNAs-- then you want to know which parts of the molecule are free and which are internally based paired.

And if you want to predict non RNAs genes in a genome, you may want to look for regions that are under selection for conservation of RNA structure, for conservation of the potential to base pair at some distance. If you see that, it's much more likely that that region of the genome encodes a non-coding RNA than it codes, for example-- there's a coding axon or that it's a transcription factor binding site or something like that that functions at the DNA level. So having this notion of

structure-- even just secondary structure-- is helpful for that application as well, and predicting functions as well, as I mentioned.

So co-variation. So let's take a look at these sequences. So imagine you've discovered a new class of mini micro RNA's. They're only eight bases long, and you've sequence five homologues from your five favorite mammals.

And these are the sequences that you get. And you know that they're homologous by [? a centimeter ?], they're in the same place in the genome, and they seem to have the same function. What could you say about their secondary structure based on this multiple alignment? You have to stare at it a little bit to see the pattern. There's a pattern here.

Any ideas? Anyone have a guess about what the structure is? Yeah, go ahead.

AUDIENCE: There's a two base pair stem, and then a four base loop.

PROFESSOR: Two base pair stem, four base loop, and you have of the stem. So how do you know that?

AUDIENCE: So if you look at the first two and last two bases of each sequence, the first and the eighths nucleotide can pair with each other, and so can the second and the seventh.

PROFESSOR: Yeah. Everyone see that? So in the first column you have AUACG, and that's complementary to UAUGC. Each base is complementary.

And the second position is CAGGU complementary to GUCUA. There's one slight exception there.

AUDIENCE: [INAUDIBLE]

PROFESSOR: Yeah. Well, it turns out that that RNA-- although the Watson Crick pairs GC and AU are the most stable-- GU pairs are only a little bit less stable than AU pairs, and they occur in natural RNA molecules. So GU is allowed in RNA even though you would never see that in DNA. OK, so everyone see that?

So the structure is-- I think I have it here. This would be co-variation You're changing the bases, but preserving the ability to pair. So when one base change-- when the first base changes from A to U, the last base changes from U to A in order to preserve that pairing.

You wouldn't know that if you just had two sequences, but once you get several sequences, it can be pretty compelling and allow you to make a pretty strong inference that that is the structure of that molecule. So how would you do this? So imagine you had a more realistic example where you've got a non-coding RNA that's 100 or a few hundred bases long, and you might have a multiple alignment of 50 homologous sequences.

You want something, you're not going to be able to see it by eye. You need sort of a more objective criterion. So one method that's commonly used is this statistic IX mutual information.

So if you look in your multiple alignment-- I'll just draw this here. You have many sequences. You consider every pair of columns-- this is a multiple alignment, so this column and this column-- and you calculate what we're going to call-- what are we going to call it? f ix.

That would be the frequency of a nucleotide x. You're in column i, so you just count how many A's, C's, G's, and T's there are. And similarly, f jy for all the possible values of x and all the possible values of y.

So these are the base frequencies in each column. And then you calculate the dinucleotide frequencies xy at each pair of columns. So in this colony, you say if there's an A here and a C here, and then there's another AC down here, and there's a total of one, two, three, four, five, six, seven sequences, then f AC ij is 2/7.

So you just calculate the frequency of each dinucleotide. These are no longer consecutive dinucleotides in a sequence necessarily there. They can be in arbitrary spacing.

OK, so you calculate those and then you throw them into this formula, and out comes a number. So what does this formula remind of? Have you seen a similar formula before?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Someone said [INAUDIBLE] Yeah, go ahead.

AUDIENCE: It reminds me of the Shannon entropy [INAUDIBLE].

PROFESSOR: Yeah, it looks like Shannon entropy, but there's a log of a ratio in there, so it's not exactly Shannon entropy. So what other formula has a log of a ratio in it?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Relative. So it actually looks like relative entropy. So relative entropy of what versus what? Who can sort of say more precisely if it's-- we'll say it's relative entropy of something versus a p versus q. And what is p and what is q? Yeah, in the back.

AUDIENCE: Is it relative entropy of co-occurrence versus independent occurrence?

PROFESSOR: Good. Yeah. co-occurence-- everyone get that? Co-occurrence of a pair of nucleotide xy at positions ij. Versus q is an independent occurrence. So if x and y occurred independently, they would have this frequency.

So if you think about it, you calculate the frequency of each base at each column in the multiple alignment. And this is like your null hypothesis. You're going to assume, what if they're evolving independently? So if it's not a folded RNA-- or if it's a folded RNA but those two columns don't happen to interact-- there's no reason to suspect that those bases would have any relationship to each other.

So this is like your expected value of the frequency of xy in position ij. And then this p is your observed value. So you're taking relative entropy of basically observed over expected.

And so relative entropy has-- I haven't proved this, but it's non-negative. It can be 0,

and then it goes up to some maximum, a positive value, but it's never negative. And what would it be if, in fact, p were equal to q? What would this formula give?

This is where we're saying suppose. Suppose this. In general, this won't be sure, but suppose it was equal to that. We've got mi ij equals summation of what?

That log of this, which is equal to this, so it's fx i fy j over the same thing-- hope you can see that-- log of-- log of 1 is 0, right? So it's just 0.

So if the nucleotides of the two columns occur completely independently, mutual information is 0. And that's one reason it's called mutual information. There's no information. Knowing what's in column i gives you no information about column j. So remember, relative entities are measures of information, not entropy.

And what is the maximum value that the mutual information could have? Any ideas on that? Any guesses? Joe, yeah.

AUDIENCE: You could have log base 2 log over f sub x, f sub y.

PROFESSOR: Of 1? OK, so you're saying if one of the particular dinucleotides had a frequency of 1?

AUDIENCE: Yeah. So if they're always the same whenever there's-- like an A, there's always going to be a T.

PROFESSOR: Right. So whenever there's an A, there's always a G or a T.

AUDIENCE: So then you'd get a 1 in the numerator, and they're relative probably in the bottom, which would be maximized if they were all even.

PROFESSOR: If they were all?

[INTERPOSING VOICES]

PROFESSOR: If they were uniform. Yeah. So did everyone get that? So the maximum occurs if fx i and j-- they're both uniform, so they're a quarter for every base at both positions. That's the maximum entropy in the background distribution.

But then if fx y ij equals 1/4, for example, x equals y-- or in our case, we're not interested in that. We're interested in x equals complement of y. C of y is going to be the complement of y. And 0 otherwise for x not equal complement of y.

OK, so for example, if we have only the dinucleotides AT, CG, GC, and TA occur, and each of them occurs with a frequency of 1/4, then you'll have four terms in the sum because, remember, the 0 log 0 is 0. So you'll have four terms in the sum, and each of them will look like 1/4 log 1/4 over a 1/4 times 1/4.

And so this will be 4, so log 2 of 4 4 is 2. And so you have four terms that are each 1/4 times 2. And so you'll get 2.

Well, this is not a sum. These are the four terms. These are the individual nonzero terms in that sum. Does that make sense? Everyone get this?

So that's why this is a useful measure of co-variation. If what's in one column really strongly influences what's in the other column, and there's a lot of variation in the two columns, and so you can really see that co-variation well, then mutual information is maximized. And that's basically what we just said, is written down here.

So it's maximal. They don't have to be complementary. It would achieve this maximum of 2 if they are complementary, but it would be also if they had some other very specific relationship between the nucleotides. So if you're going to use this, the way you would use it is take your multiple alignment, calculate the mutual information of each pair of columns-- so you actually have to make a table, i versus j, all possible pairs of columns-- and then you're going to look for the really high values.

And then when you find those high values, when you look at what actual bases are tending to occur together, you'll want to see that they're bases that are complementary to one another. And another thing that you'd want to see is you'd want to see that consecutive positions in one part of the alignment are co-varying with consecutive positions in another part of the alignment in the right way, in this

sort of inverse complementary way that RNA likes to pair.

Does that make sense? So in a sort of nested way in your multiple alignment, if you saw that this one co-varied with that, and then you also saw that the next base co-varied with the base right before this one, and this one co-varies with that one, that starts to look like a stem. It's much more likely that you have a three-base stem than that you just have some isolated base pair out in the middle of nowhere. It turns out it takes a few bases to make a good thermodynamically stable stem, and so you want to look for blocks of these things.

And so this works pretty well. Yeah, actually, one point I want to make first is that mutual information is nice because it's kind of a useful concept and it also relates to some of the entropy and relative entropy that we've been talking about in the course before. But it's not the only statistic that would work in practice. You can use any measure of basically non-independence between distributions. A chi square statistic would probably work equally well in practice.

And so here is a multiple alignment of a bunch of sequences. And what I've done is put boxes around columns that have significant

mutual information with other sets of columns. So for example, this set of columns here at the left-- the far left-- has significant mutual information with the ones at the far right. And these ones, these four positions co-vary with these four, and so forth. So can you tell, based on looking at this pattern of co-variation, what the structure is going to be?

OK, let's say we start up here. The first is going to pair with the last, with something at the end. Then we're going to have something here in the middle that pairs with something else nearby. Then we have something here that pairs with something else nearby, then we have another like that.

Does that make sense? So that there's these three pairs of columns in the middle-- these two, these two, and these two-- and then they're surrounded by this thing, the first pairing with the last. And so it's a clover leaf, so that's tRNA. Yeah?

AUDIENCE: So with that previous slide, this table here, you could create a co-variation matrix. How would that-- or, and it could be--

PROFESSOR: How does that co-variations matrix-- how do you convert it to this representations?

AUDIENCE: I'm just wondering how this would go up. Like let's say you took the co-variation matrix--

PROFESSOR: Oh, what would it look like?

AUDIENCE: --and visualized it as a heat map--

PROFESSOR: In the co-variation matrix.

AUDIENCE: Yeah. What would it look like in this particular example?

PROFESSOR: Yeah, that's a good question. OK, let's do that. I haven't thought about that before, so you'll have to help me on this. So here's the beginning.

We're going to write the sequence from 1 to n in both dimensions. And so here's the beginning, and it co-varies with the end. So this first would have a co-variation with the last, and then the second would co-vary with the second to last, and so forth. So you get a little diagonal down here. That's this top stem here.

And then what about the second stem? So then you have something down here that's going to co-vary with something kind of near by it. So block two is going to co-vary with block three. And again, it's going to be this inverse complementary kind of thing like that.

It's symmetrical, so you get this with that. But you only have to do one half, so you can just do this upper half here. So you get that. So it would look something like that.

AUDIENCE: So with the diagonal line orthogonal to the diagonal of the matrix--

PROFESSOR: Yeah, that's because they're inverse complementary.

AUDIENCE: OK.

PROFESSOR: That make sense? Good question. But we'll see an example like that later actually, as it turns out.

All right, so here's my question for you. You're setting this non-coding RNA. It has some length. You have some number of sequences.

They might have some structure. Is this method going to work for you, or is it not? What is required for it to work?

For example, would I want to isolate this gene-- this non-coding RNA gene-- just from primates, from like human, gorilla, chimp, orangutan, and do that alignment? Or would I want to go further? Would I want to go back to the rodents and dog, horse-- how far do you want to go? Yeah, question.

AUDIENCE: I think we a need a very strong sequence alignment for this, so we cannot go very far, because if you don't have a high percentage homology, then you will see all sorts of false positives.

PROFESSOR: Absolutely. So if you go too far, your alignment will suffer, and you need an alignment in order to identify the corresponding columns. So that puts an upper limit on how far you can go. But excellent point.

Is there a lower limit? Do you want to go as close as possible, like this example I gave with human, chimp, orangutan? Or is that too close? Why is too close bad? Tim?

AUDIENCE: Maybe if you're too close, then the sequence is having to [INAUDIBLE] to give you enough information [INAUDIBLE].

PROFESSOR: Yeah, exactly. They're all the same. Actually, you'll get 1 times 1 over 1 in that mutual information statistic, which log of that is going to be 0. There's zero mutual information if they're all the same.

So there has to be some variation, and the structure has to be conserved. That's

key. You have to assume that the structure is well conserved and you have to have a good alignment and there has to be some variation, a certain amount of variation.

Those are basically the three keys. Secondary structure has a more highly conserved sequence. Sufficient divergence so that you have these variations, and sufficient number of homologues you have to get good statistics, and not so far they your alignment is bad. Sorry about that. Sally?

AUDIENCE: It seems like another thing that we assume here is that you can project it onto a plane and it will lie flat. So if you have some very important, weird folding that allows you to, say, crisscross the rainbow thing.

PROFESSOR: Yeah, crisscross the rainbow. Yeah, very good question. So in the example of tRNA, if you were to do that arc diagram for tRNA, it would look like another big arc-- that's the first and last-- and then you have these three nested arcs. Nothing crisscrossing.

What if I saw-- [INAUDIBLE]-- two blocks of sequence that have a relationship like that? Is that OK? With this method, the co-variation, that's OK. There's no problem there. What does this structure look like?

So [INAUDIBLE] you have a stem, then you have a loop, and then a stem. So this is 1 pairs with 3. That's 1. That's 3.

Then you've got 2 up here, but 2 pairs with 4. So here's 4 over here, so 4 is going to have to come back up here and pair with 2. This is 2 over here.

So that is called a pseudoknot. It's not really a knot because this thing doesn't go through the loop, but it kind of behaves like a knot in some ways. And so do these actually occur in natural RNAs? Yes, Tim is nodding.

And are they important? Can you give me an example where they are important biologically?

AUDIENCE: [INAUDIBLE]

[INTERPOSING VOICES]

PROFESSOR: Riboswitches. We're going to come to what riboswitches are in a moment for those not familiar. And I think I have an example later of a pseudoknot that's important. So that's a good question.

I think I should have added to this list the point that you made in the back that they have to be close enough that you can get a good alignment. I should add that to this last. Thanks. It's a good point.

All right, so classes of non-coding RNAs. As promised, my favorites listed here. Everyone knows tRNAs, rRNAs. You can think of UTRs as being non RNAs. They often have structure that can be involved in regulating the message.

snRNAs involved splicing. snoRNAs-- small nucleolar RNAs-- are involved in directing modification of other RNAs, such as ribosomal RNAs and snRNAs, for example. Terminators of transcription in prokaryotes are like little stem loop structures.

RNaseP is an important enzyme. SRP is involved in targeting proteins with signal peptides to the export machinery. We won't go into tmRNA. micro RNAs and link RNAs, you probably know, and riboswitches. So Tim, can you tell us what a riboswitch is?

AUDIENCE: A riboswitch is any RNA structure that changes confirmation according to some stimulus [INAUDIBLE] or something in the cell. It could be an ion, critical changes in the structure. [INAUDIBLE].

PROFESSOR: Yeah, that was great. So just for those that may not have heard, I'll just say it again. So a riboswitch is any RNA that can have multiple confirmations, and changes confirmation in response to some stimulus-- temperature, binding of some ligand, small molecules, something like that, et cetera.

And often, one of those structures will block a particular regulatory element. I'll show an example in a moment. And so when it's in one confirmation, the gene will be

repressed. And when it's in the other, it'll be on. so it's a way of using RNA's secondary structure to sense what's going on in the cell and to appropriately regulate gene expression.

All right, so now we're going to talk about a second approach. So this would be the approach. You've got some RNA. It may not do something, and maybe you can't find any homologues.

It might be some newly evolved species-specific RNA, or your studying some obscure species where you don't have a lot of genomic sequence around. So you want to use the first principles, approach, the energy minimization approach. Or maybe you have the homologues, but you don't trust your alignment. You want a second opinion on what the structure is going to be.

So just in the way that protein folding-- you could think of an equilibrium model where it's determined by folding free energy, and enthalpy will favor base pairing. You get gain some enthalpy when you form a hydrogen bond, and entropy will tend to favor unfolding. So an RNA molecule that's linear has all this confirmational flexibility, and lose some of that when you form a stem. It forms a helix. Those things don't have as much flexibility.

And even the nucleotides in the loop are a little bit confirmationally-- they're not as flexible as they were when it was linear. So that means that at high temperatures, it'll favor unfolding. So the earliest approaches were approaches that sought to maximize the number of base pairs.

So they basically ignore entropy and focus on the enthalpy that you gain from forming base pairs. And so Ruth Nussinov described the first algorithm to figure out what is the maximum number of base pairs that you can form in an RNA. And so a way to think about this is imagine you've got this sequence.

What is the largest number of base pairs I can form with this sequence? I could just draw all possible base pairs. That A can pair with that T. This A can pair with that T.

They can't both pair simultaneously, right? And this C can pair with that G. So if we

don't allow crossing, which-- coming back to Sally's point-- this would cross this, right? So we're not going to allow that. So the best you could do be to have this A pair with this C and this C pair with this G and form this little structure.

This is not realistic because RNA loops can't be one base. They minimum is about three. But just for the sake of argument, you can list all these out, but imagine now you've got 100 bases here.

Every base will on average potentially be able to pair with 24 or 25 other bases. So you're just going to have just an incredible mishmash of possible lines all crisscrossing. So how do you figure out how to maximize that pairing? Any ideas? Don, yeah?

AUDIENCE: You look for sections of homology.

PROFESSOR: We're not using homology. We're doing [INAUDIBLE]

AUDIENCE: I'm sorry, not homology, but sections where--

PROFESSOR: Complementary?

AUDIENCE: Complementary. Yeah, that's the word I was thinking.

PROFESSOR: The blocks are complementary.

AUDIENCE: And then so--

PROFESSOR: You could blast the sequence against inverse complements itself and look for little blocks. You could do that. That's not what people generally do, mostly because the blocks of complementarity in real RNA structures are really short. They can be two, three, four, bases. Sally, yeah?

AUDIENCE: Could you use [INAUDIBLE] approach where you just start with a very small case and build up?

PROFESSOR: So we've seen that work for protein sequence alignment. We've seen it work for the Viterbi algorithm. So that is sort of the go-to approach in bioinfomatics, is to use

some sort of dynamic programming.

Now this one for RNA secondary structure that Nussinov came up with is a little bit different than the others. So you'll see it has a kind of different flavor. It turns out to be actually it's a little hard to get your head around at the beginning, but it's actually easier to do by hand. So let's take a look at that.

OK, so recursive maximization of base pairing. Now the thing about base pairing that's different from these other problems is that the first base in the sequence can base pair with the last. How do you chop up a sequence?

Remember with Needleman-Wunsch and with Viterbi we go from the beginning to the end, and that's a logical order. But with base pairing, that's actually not a logical order. You can't really do it that way.

So instead, you go from the inside out. You start in the middle of a sequence and work your way outwards in both directions. Or another way to think about it is you start with you write the sequence from 1 to n on both axes, and then actually we'll see that we initiate the diagonal all to 0's.

And then we think about these positions here next. So 1 versus 2. Could 1 pair with 2? And could 2 pair with 3?

Those are like little bits of possible RNA secondary structure. Again, we're ignoring this fact that loops have to be certain minimum. This is sort of a simplified case. And then you build outwards.

So you conclude that base 4 here could pair with base 5, so we're going to put a 1 there. And then we're going to build outward from that toward the beginning of the sequence and toward the end, adding additional base pairs when we can. That's basically the way the [INAUDIBLE] works.

And so that's one key idea, that we go from sort of close sequences, work outward, to faraway sequences. And the second key idea is that the relationship that, as you add more bases on the outside of what you've already got, that the optimal

structure in that larger portion of sequence space is related to the optimal structures of smaller portions of it in one of four different ways. And these are the four ways.

So let's look at these. So the first one is probably the simplest where if you're doing this, you're here somewhere, meaning you've compared sequences from position, let's say, i minus 1 to j minus 1 here. And then we're going to consider adding-- actually, it depends how you number your sequence. Let me see how this is done. Sorry. i plus 1.

i plus 1 to j minus 1. We figured out what the optimal structure is in here, let's suppose. And now we're going to consider adding one more base on either end. We're going to add j down here, and we're going to ask if it pairs with i.

And if so, we're going to take whatever the optimal structure was in here and we're going to add one base pair, and we're going to add plus 1 because now it's got one additional. We're counting base pairs. So that's that first case there.

And then the second case is you could also consider just adding one unpaired base onto whatever structure you had, and then you don't add one. And you could go in either direction. You can go sort of toward of the beginning of the sequence or toward the end of the sequence.

And then the third one is the tricky one, is what's called a bifurcation. You could consider that actually i and j are both paired, but not with each other. That i pairs with something that was inside here and j pairs with something that was inside here. So your optimal parse from i to j, if you will, is not going to come from the optimal parse from i plus 1 to j minus 1. It's going to come from rethinking this and doing the optimal parse from here to here and from here to here, and combining those two.

So you're probably confused by now, so let me try to do an example. And then I have an analogy that will confuse you further. So ask me for that one. This was the simplest one I could come up with that has this property.

OK, so we said before that if you were doing the optimal from 1 to 5, that it would be the AC pairing with the GT. We do that one. And now if you notice, this guy is kind of

a similar sequence. I just added a T at the beginning and an A at the end.

And so you can probably imagine that the best structure of this is here, those three. You've got three pairs of this sub-sequence here. That's as good as you can do with seven bases. You can only get three pairs. And this is as good as you can do with five, so these are clearly optimal.

So the issue comes that if you're starting from somewhere in the middle here-- let's say you are-- let's see, so how would you be doing this? You start here. Let's suppose the first two you consider are these two. You consider pairing that T with that A.

You can see this is not going to go well. You might end up with that as your optimal substructure of this region. Remember, you're working from the inside out, so you're going from here to here, and you end up with that.

And what do you do here? You don't have a G to pair the C to, so you add another unpaired base. Now you've got this optimal substructure of a sequence that's almost the whole sequence. It's just missing the first and last bases, but it only has three base pairs.

So when you go to add this, you can say, oh, I can't add any more base pairs, so I've only got three. But you should consider that we've already solved the optimal structure of that, and we had two nice pairs here. We had that pair and that pair, and we already solved the substructure of the optimal structure of this portion here, and you had those three pairs.

And so you can combine those two and all of a sudden you can do much better. So that's what that bifurcation thing is about. So this is the recursion working out, and you can see that's the base pairing one. You can add one, or you can just add an unpaired base and you don't add anything.

Or you consider all the possible locations of bifurcations in-between the two positions you're adding, i and j, and you consider all the possible pairs. And you just sum up each pair and go-- I'm sorry, you don't sum them up. You consider them all,

and then you take the maximum.

All right, so the algorithm is to take an n by n matrix, initialize the diagonal to 0, and initialize the sub-diagonal to 0 also. Just don't think too much about that. Just do it.

And then fill in this matrix recursively from the diagonal up and to the right. And it actually doesn't matter what order you fill it in as long as you're kind of working your way up into the right. You have to have the thing to the left and the thing below already filled in if you're going to fill in a box.

And then you keep track of the optimal score, which is going to be the sum of base pairs. And then you also keep track of how you got there. What base pair did you add so that you can trace back?

And then when you get up to the upper right corner of this matrix, you then trace back. So here is a partially filled in this matrix. This is from that the *Nature Biotechnology Review*. And the 0's are filled in.

So here's what I want you to do at home, is print out, photocopy or whatever-- make this matrix, or make a bigger version of it perhaps-- and look at the sequence and fill in this matrix, and fill in the little arrows every time you add a base pair. It's actually not that hard. There are no bifurcations in this, so that's the tricky one. Ignore that one.

You'll just be adding base pairs. It'll be pretty easy. And then you can reconstruct the sequence.

So here it is filled in. And the answer is given, so you can check yourself. But do it without looking at the answer. And then you go to the upper right corner.

That means that the optimal structure from the beginning of the sequence to the end-- which, of course, was our goal all along. And then you trace back and you can see whenever you're moving diagonally here, you're adding a base pair. Remember, you add one on each end, and so you're moving diagonally and adding the base pair, and you get this little structure here.

So computational complexity of the algorithm. You could think about this but I'll just tell you. It's memory n squared because you've got to fill in this matrix, so square of the length of the sequence.

Time n cubed. This is bad now. And why is it n cubed? It's n cubed because you have to fill in a matrix that's n by n. And then when you do that maximization step, that check for bifurcations, that's sort of of order n, as well.

So n cubed-- so this means that RNA folding is slow. And in fact, some of the servers won't allow you to fold anything more than a thousand bases because they'll take forever or something like that. And it cannot handle pseudoknots. If you think through the recursion, pseudoknots will be a problem.

I'm going to just show you-- yeah, I'll get to this-- that these are from the viruses. Real viruses, some of them have pseudoknots like these ones shown here, and some even have these kissing loops, which is another type where the two stem loops, the loops interact. And the pseudoknots in particular are important in the viral life cycle.

They can actually cause programmed ribosomal frame shifting. When the ribosomes hits one of the things, normally it just denatures RNA secondary structure. When it hits a pseudoknot, it'll actually get knocked back by one and will start translating in a different frame. And that's actually useful to the virus to do that under certain circumstances.

That's how HIV makes the replicated polymerase, is by doing a frame shift on the ribosome using a pseudoknot. So these things are important. And there's fancier methods that use more sophisticated thermodynamic models where GC counts more than AU.

And I won't go into the details, but I just wanted to show you some pretty pictures here that the Zuker algorithm-- this is a real world RNA folding algorithm-- calculates not only the minimum energy fold, but also sub-optimal folds, and the probabilities of particular base pairs, summing over all the possible structures that RNA could

form, weighted by their free energy. So it's the full partition function.

It's not perfectly accurate. It gets about 70% of base pairs correct, which means it usually gets things right, but occasionally totally wrong. And there's a website for the Mfold server, which is actually one of the most beautiful websites in bioinfomatics, I would say. And also if you want to run it locally, you should download the Vienna RNAfold package, which has a very similar algorithm.

And I just wanted to show you one or two examples. So this is the U5 snRNA. This is the output of Mfold. It predicts this structure. And then this what's called the energy dot plot, which shows the bases in the optimal structure down below here and then sort of these suboptimal structures here. And you can see there's no ambiguity. It's totally confident in this structure.

Then I ran the lysine riboswitch through this program, and I got this. I got the minimum for energy structure down in the lower left. And then you see there's a lot of other colored dots. Those are from the suboptimal structures.

So it looks like this thing has multiple structures, which of course it does. So the way that this one works is, in the absence of lysine, it forms this structure where the ribosome binding sequences-- this is prokaryotic-- is exposed. And so the ribosome can enter and translate these lysine biosynthetic enzymes.

But then when lysine accumulates to a certain level, it can interact with the RNA and shift it's structure so that you now form this stem, which sequesters the ribosome binding sequence and blocks lysine biosynthesis. So a very clever system.

And it turns out that there's dozens of these things in bacterial genomes, and they control a lot of metabolism. So they're very important. And there may be some in eukaryotes, too, and that would be good.

If anyone's looking for a product, not happy with their current project, you might think about looking for more riboswitches. So I'm going to have to end there. And thank you guys for your attention, and good luck on the midterm.